

SystemC Cycle Models

Version 3.0.0

User Guide

Non-Confidential

arm

Proprietary Notice

This document is protected by copyright and other related rights and the practice 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. **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 implementations infringe any third party patents.

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, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

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.

This document consists solely of commercial items. You shall be responsible for ensuring that any 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.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. 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 the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2017 Arm. All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

Contents

Arm SystemC Cycle Models User Guide

	Preface	
	About this book	6
	Implementation obligations	6
	Intended audience	6
	Glossary	6
	Conventions	6
	Additional reading	7
	Feedback	8
	Feedback on this product	8
	Feedback on content	8
	Customer support	8
Chapter 1	Introduction	
	1.1 About SystemC Cycle Models	10
	1.2 Requirements and supported platforms	11
	1.2.1 Prerequisites	11
	1.2.2 Supported platforms	11
	1.2.3 Supported compilers	11
	1.3 Package contents	12
Chapter 2	Using SystemC Cycle Models	
	2.1 Replacing a SystemC Cycle Model in a CPAK	16
	2.2 Adding a SystemC Cycle Model to a CPAK	17
	2.3 Dumping Waveforms	18
	2.4 Configuring TARMAC Trace	19
	2.5 Configuring PMU events	20
	2.6 Configuring Cache and TCM Sizes (Cortex-R52 SystemC Cycle Model only)	21
	2.7 Loading TCMs (Cortex-R8 SystemC Cycle Model only)	22
	2.8 Loading TCMs (Cortex-R52 SystemC Cycle Model Only)	23
	2.9 Resetting the SystemC Cycle Model	24

Preface

This preface introduces the *Arm® SystemC Cycle Models User Guide*. It contains the following sections:

- *About this book on page 16.*
- *Feedback on page 18.*

About this book

This book is for the Arm SystemC Cycle Models.

Implementation obligations

This book is designed to help you implement an Arm® product. The extent to which the deliverables may be modified or disclosed is governed by the contract between Arm and Licensee. There may be validation requirements, which if applicable will be detailed in the contract between Arm and Licensee and which if present must be complied with prior to the distribution of any silicon devices incorporating the technology described in this document. Reproduction of this document is only permitted in accordance with the licences granted to Licensee.

Arm assumes no liability for your overall system design and performance, the verification procedures defined by Arm are only intended to verify the correct implementation of the technology licensed by Arm, and are not intended to test the functionality or performance of the overall system. You or the Licensee will be responsible for performing any system level tests.

You are responsible for any applications which are used in conjunction with the Arm technology described in this document, and in order to minimise risks adequate design and operating safeguards should be provided for by you. Arm's publication of any information in this document of information regarding any third party's products or services is not an express or implied approval or endorsement of the use thereof.

Intended audience

This book is written for experienced hardware engineers, software engineers and System-on-Chip (SoC) designers who might have experience of Arm products. You are expected to have experience of SystemC.

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 *Arm® Glossary* <http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html>.

Conventions

This book uses the conventions that are described in:

- *Typographical conventions*.

Typographical conventions

The following table describes the typographical conventions:

Typographical conventions

Style	Purpose
<i>italic</i>	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
monospace <i>italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
monospace bold	Denotes language keywords when used outside example code.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>Arm® Glossary</i> . For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Additional reading

This section lists publications by Arm and by third parties.

See *Infocenter* <http://infocenter.arm.com>, for access to Arm documentation.

Arm publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Cycle Model SystemC Runtime Installation Guide* (Arm 101146)
- *Cycle Model Studio SystemC User Manual* (Arm DUI 1057)

Feedback

Arm welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title.
- The number, ARM 101124.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

———— **Note** —————

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Customer support

If you have an active support contract then go to *DesignStart™ - Online Access to Arm IP* <http://www.arm.com/support/designstart.php> and then login to obtain prompt attention to issues and questions about your Arm Physical Intellectual Property product.

Go to *Support and Maintenance*

<http://www.arm.com/support/services/support-maintenance.php> and select the **Physical IP** tab for information about support contract options.

If you cannot contact Arm through the web support channel then send an e-mail to support-pipd@arm.com.

Chapter 1

Introduction

This chapter introduces the Arm SystemC Cycle Models. It contains the following sections:

- *About SystemC Cycle Models on page 1-10*
- *Requirements and supported platforms on page 1-11*
- *Package contents on page 1-12*

1.1 About SystemC Cycle Models

Arm SystemC Cycle Models are compiled directly from the RTL code. The SystemC model wrappers are provided in source form to enable compiling for any SystemC 2.3.1-compliant simulator. You can integrate these models directly into any IEEE 1666-compliant SystemC environment.

SystemC Cycle Models may be used outside the scope of a CPAK.

1.2 Requirements and supported platforms

This section describes prerequisites to using Arm SystemC Cycle Models and supported platforms and compilers.

1.2.1 Prerequisites

The following are required to use Arm SystemC Cycle Models:

- Simulation and recompilation require the Cycle Model Studio Runtime. Linux and Windows versions of this runtime are available.
- The Cycle Model SystemC Runtime. See the *Cycle Model SystemC Runtime Installation Guide* (Arm 101146) for more information.
- You must also have a SystemC environment configured. See the *Cycle Model SystemC Runtime Installation Guide* (Arm 101146) for supported versions.

The Cycle Model Studio Runtime Library and the Cycle Model SystemC Runtime are available in the Support area of Arm IP Exchange (<https://www.armipexchange.com/>).

1.2.2 Supported platforms

Arm SystemC Cycle Models are supported on Red Hat Enterprise Linux version 6.6 (64-bit) and above.

1.2.3 Supported compilers

The SystemC Cycle Models have been tested on Linux with GCC 4.8.3. Newer versions of this compiler may also work.

The SystemC Cycle Models include C++ Version 11 code, so the GCC in use must support this.

1.3 Package contents

Each SystemC Cycle Model contains the following files:

———— **Note** ————

All files may not be present for all models.

`<component>ResetModule.h`

Reset module used to drive the SystemC pin-level wrapper for the Reset sequence of the IP.

`<component>.xmlAnswers`

Shows the configuration of the Cycle Model as requested when the model was built on Arm IP Exchange.

`lib<component>.a`

RTL-based core of the Cycle Model. This can be compiled into the system executable.

`lib<component>.h`

Base function header exposed by the core Cycle Model. This is required to access functions in the core Cycle Model. `lib<component>.systemc.cpp` and `lib<component>.systemc.h` contain a reference to `lib<component>.h`.

`lib<component>.so` **and** `lib<component>.a`

Compiled version of the pin-level Cycle Model implementation, which can be linked into the system-level model.

`lib<component>.systemc.cpp`

Pin-level SystemC wrapping implementation around the core Cycle Model. This can be compiled to generate a signal-level, linked SystemC model.

`lib<component>.systemc.h`

Pin-level SystemC wrapping header for the core Cycle Model. This can be compiled to generate a signal-level, linked SystemC model.

`loadTCMUtil/*` (Availability is model-dependent)

Set of files that support direct loading of the TCM.

`Makefile`

Compiles the pin-level model into the shared libraries included with the installation.

`<component>_tarmac.h` (Availability is model-dependent)

Cycle Model parameter definition to generate univent traces.

`<component>_params.h` (Availability is model-dependent)

Cycle Model-specific parameter definitions.

`<component>_pmu.h` (Availability is model-dependent)

Cycle Model hardware profiling implementation to generate profiling events.

uivent_tarmac.cpp (Availability is model-dependent)

Univent interface header which can be hooked into the pin level Cycle Model to generate univent traces.

uivent_tarmac.h (Availability is model-dependent)

Univent interface implementation which can be hooked into the pin level Cycle Model to generate univent traces.

uiventUtil/* (Availability is model-dependent)

Contains model-specific Univent libraries which are needed to compile the univent_tarmac.cpp and univent_tarmac.h into the model.

Chapter 2

Using SystemC Cycle Models

This chapter describes:

- *Replacing a SystemC Cycle Model in a CPAK on page 2-16*
- *Adding a SystemC Cycle Model to a CPAK on page 2-17*
- *Dumping Waveforms on page 2-18*
- *Configuring TARMAC Trace on page 2-19*
- *Configuring PMU events on page 2-20*
- *Configuring Cache and TCM Sizes (Cortex-R52 SystemC Cycle Model only) on page 2-21*
- *Loading TCMs (Cortex-R8 SystemC Cycle Model only) on page 2-22*
- *Loading TCMs (Cortex-R52 SystemC Cycle Model Only) on page 2-23*
- *Resetting the SystemC Cycle Model on page 2-24*

Note

Pin-based SystemC Cycle Models may be used outside the scope of a CPAK; it is strongly suggested that you download an Arm SystemC CPAK, which provides a valuable reference when you are working with SystemC Cycle Models. Access Arm System Exchange (<https://www.armssystemexchange.com/cpaks/>) to download a CPAK.

2.1 Replacing a SystemC Cycle Model in a CPAK

To replace IP in an existing Arm SystemC CPAK with an alternate configuration of the same IP — for example, swapping a single-core CPU for a dual-core CPU — replace the contents of the MODELS directory with the files for the new SystemC Cycle Model.

2.2 Adding a SystemC Cycle Model to a CPAK

If you want to add new SystemC Cycle Models (for example, additional cores, memory, or peripheral components) to your SystemC CPAK:

1. Access Arm IP Exchange (<https://www.armipexchange.com/>) to configure, build and download the new model.
2. In your CPAK directory, add the files for the new SystemC Cycle Model to a new directory in CPAK/MODELS.

To connect two ports on different models:

1. Declare an `sc_signal` in the `system_test.cpp` file. This signal needs to be the same type and width as the two ports. If the ports are the same type but different widths, use `scx_signal_sizer` instead of `sc_signal`.
2. Edit the `<component>ResetImp.cpp` file in the MODELS directory for both models and comment out the signal-to-port binding in the `bind_nontlm_ports_signals` method.
3. Bind the signal to both ports in the `system_test.cpp` file. For example:


```
sc_signal<bool> signal1;
Inst1.port1.bind(signal1);
Inst2.port1.bind(signal2);
```
4. Recompile the system. Models are recompiled automatically as part of the system recompile.

2.3 Dumping Waveforms

You can instrument waveform dumping using the APIs documented in the *Cycle Model Studio SystemC User Manual* (Arm DUI 1057). If you do not have a full Cycle Model Studio installation that includes this document, it is available on Arm Developer (<https://developer.arm.com>).

You can also enable and disable waveform dumping by setting parameter values within the system executable code. Set the following parameters:

Waveform Parameters

Parameter	Available settings	Default setting
WAVEFORMS_ENABLED	True, False	
WAVEFORM_TIMEUNIT	Units defined by <code>sc_time_unit()</code> : - SC_FS - SC_PS - SC_NS - SC_US - SC_MS - SC_SEC	SC_PS
WAVEFORM_TYPE	FSDB, VCD	VCD

For example:

```
scx::scx_set_parameter("<sc-module-name>.WAVEFORMS_ENABLED",True);
scx::scx_set_parameter("<sc-module-name>.WAVEFORM_TIMEUNIT",SC_NS);
scx::scx_set_parameter("<sc-module-name>.WAVEFORMS_TYPE","FSDB");
```

`sc-module-name` is the name given to the model when it is instantiated in the system executable.

2.4 Configuring TARMAC Trace

The following SystemC Cycle Models support TARMAC Trace:

- Cortex-A32
- Cortex-A35
- Cortex-A53
- Cortex-A55
- Cortex-A75
- Cortex-M7
- Cortex-M23
- Cortex-M33
- Cortex-R8
- Cortex-R52

By default, TARMAC Trace is disabled. You can enable TARMAC tracing by setting parameter values in the system executable code. These parameters are:

- TARMAC_ENABLED = *True* or *False*.
- TARMAC_LOGFILE_PREFIX = *""*. This is used to distinguish multiclusters. The default is *""*.

For example:

```
scx::scx_set_parameter("<sc-module-name>.TARMAC_ENABLED",True);
```

```
scx::scx_set_parameter("<sc-module-name>.TARMAC_LOGFILE_PREFIX","CLUSTER0");
```

sc-module-name is the name given to the model when it is instantiated in the system executable.

2.5 Configuring PMU events

SystemC Cycle Model PMU events are stored in C++ variables. For specifics about variable names for PMU events, refer to the following file:

- `<component>_pmu.h`

Refer to the Arm *Technical Reference Manual* for your IP for information about profiled events.

By default, calculations of PMU events are disabled in the SystemC Cycle Model. You can turn PMU events on by setting a parameter value in the system executable code. Use the following parameter:

- `PMU_ENABLED = True` or `False`

For example:

```
scx::scx_set_parameter("<sc-module-name>.PMU_ENABLED",True);
```

`sc-module-name` is the name given to the model when it is instantiated in the system executable.

2.6 Configuring Cache and TCM Sizes (Cortex-R52 SystemC Cycle Model only)

You can set the instruction cache, data cache, and TCM sizes by setting parameter values within the system executable. For parameter names and possible values, refer to the `<component>_params.h` file.

For example:

```
scx::scx_set_parameter("<sc-module-name>.ICACHE_SIZE_CPU0",7)
```

`sc-module-name` is the name given to the model when it is instantiated in the system executable.

2.7 Loading TCMs (Cortex-R8 SystemC Cycle Model only)

You can load the DTCMs and ITCMs memories by setting parameter values within the system executable. Set the following parameters:

- `LOAD_DTCMS` = *True* or *False*
- `LOAD_ITCMS` = *True* or *False*

To see the parameter names for the data files that are to be loaded, see the files `<component>_dcm.h` and `<component>_itcm.h`. You can change these files by setting the parameters in the same way.

2.8 Loading TCMs (Cortex-R52 SystemC Cycle Model Only)

You can load the ATCMs, BTCMs and CTCMs memories by setting parameter values within the system executable. Set the following parameters:

- `LOAD_ATCMS` = *True* or *False*
- `LOAD_BTCMS` = *True* or *False*
- `LOAD_CTCMS` = *True* or *False*

For the parameter names for the data files to be loaded, see the files `<component>_atcm.h`, `<component>_btcm.h` and `<component>_ctcm.h`. You can change these files by setting the parameters in the same way.

2.9 Resetting the SystemC Cycle Model

A default reset sequence is provided in source form in `<component>ResetModule.h`. Modify this file as needed, then recompile the model after making your changes.

Ensure that the reset module is connected to the model.

Refer to the Arm Technical Reference Manual for your IP for details about its reset sequence.