

Cortex-R52 Cycle Model

Version 9.6

User Guide



Cortex-R52 Cycle Model

User Guide

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

Release Information

Document History

| Issue | Date | Confidentiality | Change |
|---------|------------------|------------------|-------------------------------------|
| 0906-00 | 17 November 2017 | Non-Confidential | Release 9.6 |
| 0906-01 | 28 November 2017 | Non-Confidential | Release 9.6, documentation update 1 |

Non-Confidential 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, third party 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 Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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.

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

<http://www.arm.com>

Contents

Cortex-R52 Cycle Model User Guide

Preface

| | |
|-----------------------|---|
| About this book | 7 |
|-----------------------|---|

Chapter 1

Introduction

| | |
|---|------|
| 1.1 Supported hardware features | 1-10 |
| 1.2 Unsupported hardware features | 1-11 |
| 1.3 Additional features for Cycle Model usability | 1-12 |

Chapter 2

Adding and configuring the SoC Designer component

| | |
|---|------|
| 2.1 SoC Designer component files | 2-14 |
| 2.2 Adding the Cycle Model to the component library | 2-15 |

Chapter 3

Available component ports

| | |
|------------------------------------|------|
| 3.1 Available ESL ports | 3-17 |
| 3.2 Reset behavior and ports | 3-18 |
| 3.3 Tied pins | 3-19 |

Chapter 4

Available component parameters

| | |
|---|------|
| 4.1 Changing parameter settings in SoC Designer | 4-21 |
| 4.2 Component parameters | 4-22 |
| 4.3 Dumping TCM waveforms | 4-27 |

Chapter 5

Debug features

| | |
|--------------------------------|------|
| 5.1 Register information | 5-29 |
|--------------------------------|------|

| | | |
|-----|---------------------------------|------|
| 5.2 | <i>Disassembly view</i> | 5-33 |
| 5.3 | <i>Memory view</i> | 5-34 |
| 5.4 | <i>Hardware profiling</i> | 5-35 |

Preface

This preface introduces the *Cortex-R52 Cycle Model User Guide*.

It contains the following:

- [About this book on page 7.](#)

About this book

This document describes how to use the Cortex-R52 Cycle Model in SoC Designer.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This section summarizes the functionality of the Cycle Model compared to that of the hardware, and describes the performance and accuracy of the Cycle Model. For details, see the *Cortex-R52 Processor Technical Reference Manual* (100026).

Chapter 2 Adding and configuring the SoC Designer component

This section provides basic information about using the Cycle Model component. See the *SoC Designer User Guide* (100996) for more information.

Chapter 3 Available component ports

This section describes the differences between the pins in the hardware and those on the Cycle Model.

Chapter 4 Available component parameters

This section describes the component parameters and how to set them.

Chapter 5 Debug features

This section describes the debug features supported by the Cycle Model CADI debug interface.

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.

Typographic conventions

italic

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.

monospace

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

monospace italic

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 *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Feedback

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 *Cortex-R52 Cycle Model User Guide*.
- The number 101016_0906_01_en.
- If applicable, the page number(s) to which your comments refer.
- 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.

Other information

- *Arm® Developer*.
- *Arm® Information Center*.
- *Arm® Technical Support Knowledge Articles*.
- *Technical Support*.
- *Arm® Glossary*.

Chapter 1

Introduction

This section summarizes the functionality of the Cycle Model compared to that of the hardware, and describes the performance and accuracy of the Cycle Model. For details, see the *Cortex-R52 Processor Technical Reference Manual* (100026).

It contains the following sections:

- [1.1 Supported hardware features on page 1-10.](#)
- [1.2 Unsupported hardware features on page 1-11.](#)
- [1.3 Additional features for Cycle Model usability on page 1-12.](#)

1.1 Supported hardware features

The following features of the Cortex-R52 hardware are fully implemented in the Cortex-R52 Cycle Model:

- Configurations of up to four CPUs are supported.
- Configurable number of interrupts (32 to 960 in increments of 32).
- AXI master port.
- Access to TCMs via slave port.
- Variable ICache and DCache sizes.
- Variable ITCM and DTCM sizes.
- 16, 20, or 24 EL1-controlled MPU regions per core configurable at build time.
- 0, 16, 20, or 24 EL2-controlled MPU regions per core configurable at build time.
- Floating Point Unit (FPU).
- ETM interface, including register slice between the processor and ETM interface.
- RAM protection.

1.2 Unsupported hardware features

The following features of the Cortex-R52 hardware are not implemented in the Cortex-R52 Cycle Model:

- Semihosting.
- Split-lock mode.
- Error Correcting Code (ECC) on RAM blocks and buses.
- Memory Built-In Self Test (MBIST) interface.
- Memory Reconstruction Port (MRP).
- Use of Synopsys® DesignWare® library blocks rather than the Arm equivalents.
- Configurable size for Branch Target Address Cache (BTAC).
- Addition of one latency cycle to ITCM data read.
- Support for additional signals to control power (required for UPF).

1.3 Additional features for Cycle Model usability

The following features that are implemented in the Cycle Model do not exist in the hardware. These features have been added to the Cycle Model for enhanced usability.

- Waveform dumping, including dumping of TCM memories, using the waveform-related parameters described in [Chapter 4 Available component parameters on page 4-20](#)
- Support for viewing registers (see [5.1 Register information on page 5-29](#)).
- Support for viewing memory (see [5.3 Memory view on page 5-34](#)).
- Support for disassembly view (see [5.2 Disassembly view on page 5-33](#)).
- Support for hardware profiling (see [5.4 Hardware profiling on page 5-35](#)).

Chapter 2

Adding and configuring the SoC Designer component

This section provides basic information about using the Cycle Model component. See the *SoC Designer User Guide* (100996) for more information.

It contains the following sections:

- [2.1 SoC Designer component files](#) on page 2-14.
- [2.2 Adding the Cycle Model to the component library](#) on page 2-15.

2.1 SoC Designer component files

The component files are the final output from the Cycle Model Studio compile and are the input to SoC Designer.

There are two versions of the component: an optimized release version for normal operation, and a debug version.

On Linux, the debug version of the component is compiled without optimizations and includes debug symbols for use with gdb. The release version is compiled without debug information and is optimized for performance.

On Windows, the debug version of the component is compiled referencing the debug runtime libraries so it can be linked with the debug version of SoC Designer. The release version is compiled referencing the release runtime library. Both release and debug versions generate debug symbols for use with the Visual C++ debugger on Windows.

The component files provided with your Cycle Model are:

Table 2-1 SoC Designer component files

| Platform | File | Description |
|----------|---|-------------------------------------|
| Linux | maxlib.lib<component_name>.conf | SoC Designer configuration file |
| | lib.<component_name>.mx.so | SoC Designer component runtime file |
| | lib<component_name>.mx_DBG.so | SoC Designer component debug file |
| Windows | maxlib<component_name>.lib.windows.conf | SoC Designer configuration file |
| | lib<component_name>.mx.dll | SoC Designer component runtime file |
| | lib<component_name>.mx_DBG.dll | SoC Designer component debug file |

Additionally, this User Guide PDF is provided with the component.

2.2 Adding the Cycle Model to the component library

The compiled Cycle Model component is provided as a configuration file (.conf).

To make the component available in the Component Window in SoC Designer Canvas:

Procedure

1. Launch SoC Designer Canvas.
2. Select **File > Preferences**.
3. In the list on the left, click **Component Library**.
4. Under the **Additional Component Configuration Files** window, click **Add**.
5. Browse to the location where the Cycle Model is located and select the component configuration (.conf) file.
6. Click **OK**.
7. To save the preferences permanently, click the **OK & Save** button. The component is now available from the SoC Designer Component Window.
8. To pull the component onto the canvas, select and drag it.

The component's appearance may vary depending on your specific device configuration. Additional ports are provided depending on the model RTL configuration file (default.conf) used to create the Cycle Model.

Chapter 3

Available component ports

This section describes the differences between the pins in the hardware and those on the Cycle Model.

It contains the following sections:

- [3.1 Available ESL ports on page 3-17.](#)
- [3.2 Reset behavior and ports on page 3-18.](#)
- [3.3 Tied pins on page 3-19.](#)

3.1 Available ESL ports

Table 1-2 describes the ESL ports that are exposed in SoC Designer. See the *Cortex-R52 Technical Reference Manual* for more information.

Table 3-1 SoC Designer component files

| ESL Port | Description | Type |
|------------------------|--|-------------------------------------|
| ACP_Slave_AXISlave | ACP Slave debug port (AXI) | Transaction Slave |
| APB_Slave_Debug_APB | APB Slave debug port | Transaction Slave |
| AXI4_Master_LLPP_AXI_x | AXI4 Master Low Latency Peripheral port | Transaction Master |
| AXI4_Master_Main_AXI_x | AXI4 Master Main port | Transaction Master |
| AXI4_RO_Flash_Master_x | CPU Flash Interface | Transaction Master |
| CLKIN | Main clock of the Cortex-R52 MPCore processor. | Main Clock Transactor (Clock Slave) |
| clk-in | This port is used internally. Leave unconnected. | Clock Slave |

3.2 Reset behavior and ports

The Cycle Model is reset internally each time SoC Designer Simulator is initialized. This behavior is standard and can not be changed. To view the internal reset sequence, set the **Align Waveforms** parameter to False, and this data appears in the waveform.

At simulation time zero and while simulation is running, you can generate a reset sequence. To do so, drive the reset pins on the component using external signals (for example, using the MxSigDriver component).

For information about reset pin names, bit ordering (for multiple cores), and required reset sequence, refer to the Technical Reference Manual for your IP.

3.3 Tied pins

The following signals are tied to a certain value:

- DFTCGEN (low)
- DFTRAMHOLD (low)
- DFTRSTDISABLE (low)
- DFTMCPHOLD (low)
- MBISTADDREXT (17 bits, all tied low)
- MBISTARRAYEXT (5 bits, all tied low)
- MBISTCFGEXT (low)
- MBISTINDATAEXT (78 bits, all tied low)
- MBISTREADENEXT (low)
- MBISTWRITEENEXT (low)
- MBISTREQEXT (high)
- 0NIDEN0 (high)
- NIDEN0 (high)
- DBGEN0 (high)
- HIDEN0 (high)
- HNIDEN0 (high)
- ACLKENF0 (high)
- ACLKENM0 (high)
- ACLKENP0 (high)
- PCLKENDBG (high)
- ACLKENS (high)
- CNTCLKEN (high)

Chapter 4

Available component parameters

This section describes the component parameters and how to set them.

It contains the following sections:

- [4.1 Changing parameter settings in SoC Designer on page 4-21.](#)
- [4.2 Component parameters on page 4-22.](#)
- [4.3 Dumping TCM waveforms on page 4-27.](#)

4.1 Changing parameter settings in SoC Designer

You can change the settings of all the component parameters in SoC Designer Canvas, and of some of the parameters in SoC Designer Simulator.

To modify the component parameters:

1. In SoC Designer Canvas, right-click on the component and select **Component Information**. You can also double-click the component. The **Edit Parameters** dialog box appears. The list of available parameters may differ slightly depending on the settings that you enabled in the configuration file (`default.conf`) when creating the component.
2. In the **Parameters** window, double-click the **Value** field of the parameter that you want to modify.
3. For text fields, type a new value. If a pulldown menu is available, select the desired value.

4.2 Component parameters

The following table describes the Cycle Model component parameters. Complete details about the parameters in this table are available in the Integration and Implementation Manual or Technical Reference Manual for your IP.

Note

m indicates CPU 0, 1, 2, or 3.

Table 4-1 Component parameters

| Parameter name | Description | Allowed values | Default value | Init/ Runtime |
|---|--|----------------|---------------|---------------|
| ACLKENF _m | Flash port clock enable. | 0, 1 | 1 | Runtime |
| ACLKENM _m | AXI master port clock enable. | 0, 1 | 1 | Runtime |
| ACLKENP _m | Low Latency Peripheral Port (LLPP) clock enable. | 0, 1 | 1 | Runtime |
| ACLKENS | AXI slave port clock enable. | 0, 1 | 0 | Runtime |
| ACP_Slave_AXISlave axi_size[0-5] | This parameter should be left at its default value. | — | 0 | Init |
| ACP_Slave_AXISlave axi_start[0-5] | This parameter should be left at its default value. | — | 0 | Init |
| ACP_Slave_AXISlave Enable Debug Messages | Whether debug messages are enabled on the AXI Slave port. | true, false | false | Runtime |
| ACP_Slave_AXISlave Protocol Variant | Protocol Variant in use for AXI. | AXI4 | AXI4 | Init |
| AFVALIDD _m | FIFO flush request (ATB Data). | 0, 1 | 0 | Runtime |
| AFVALIDI _m | FIFO flush request (ATB Instruction). | 0, 1 | 0 | Runtime |
| Align Waveforms | When set to true, waveforms dumped by the component are aligned with the SoC Designer simulation time. The reset sequence, however, is not included in the dumped data. When set to false, the reset sequence is dumped to the waveform data, however, the component time is not aligned with SoC Designer time. | true, false | true | Init |
| APB_Slave_Debug_APB Base Address | APB Slave debug base address. | Integer | 0 | Init |
| APB_Slave_Debug_APB Enable Debug Messages | Whether debug messages are enabled on the APB Slave port. | true, false | false | Runtime |
| APB_Slave_Debug_APB Protocol Variant | Protocol Variant in use for APB. | APB3 | APB3 | Init |
| APB_Slave_Debug_APB Size | APB Slave debug size. | Integer | 0 | Init |
| ATCLKEND | Clock enable for the ATB interfaces. | 0, 1 | 0 | Runtime |

Table 4-1 Component parameters (continued)

| Parameter name | Description | Allowed values | Default value | Init/ Runtime |
|--|---|---|---------------|---------------|
| ATCLKENI | ATB clock enable and clock enable for TSVALUEB[63:0]. | 0, 1 | 0 | Runtime |
| ATCMSIZE _m | Sets ATCM Size. | 0 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB | 32 KB | Init |
| ATCM_WAIT_STATES _m | Sets ATCM wait state. | 0, 1 | 0 | Init |
| ATREADYD _m | ATB device ready (ATB Data). | 0, 1 | 0 | Runtime |
| ATREADYI _m | ATB device ready (ATB Instruction). | 0, 1 | 0 | Runtime |
| AXI4_Master_LLPP_AXI_m Enable Debug Messages | Enables AXI4 Master LLPP port debug. | true, false | false | Runtime |
| AXI4_Master_LLPP_AXI_m Protocol Variant | Specifies variant in use on AXI4 Master LLPP port. | AXI4 | AXI4 | Init |
| AXI4_Master_Main_AXI_m Enable Debug Messages | Enables AXI4 Master Main port debug. | true, false | false | Runtime |
| AXI4_Master_Main_AXI_m Protocol Variant | Specifies variant in use on AXI4 Main port. | AXI4 | AXI4 | Init |
| AXI4_RO_Flash_Master_0 Enable Debug Messages | Enables AXI4 RO Flash Master port debug. | true, false | false | Runtime |
| AXI4_RO_Flash_Master_0 Protocol Variant | Specifies variant in use on AXI4 RO Flash Master port. | AXI4 | AXI4 | Init |
| BTCMSIZE _m | Sets BTCM size. | 0 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB | 32 KB | Init |
| BTCM_WAIT_STATES _m | Sets BTCM wait state. | 0, 1 | 0 | Init |
| ARM_CM DB Path | Sets the directory path to the database file. | not used | empty | Init |
| CFGAXISTCMBASEADDR | Base address of the TCMs on the AXI-slave interface. | Integer | 0x1000000 | Init |
| CFGCLUSTERUTID | Cluster Unique Transaction IDentifier for the purpose of interconnect protection. | 0 - 3 | 0 | Init |
| CFGDBGROMADDR | Debug ROM table address. | Integer | 0x12000 | Init |
| CFGDMBROMADDRV | Debug ROM table address enable. | true, false | false | Init |
| CFGENDIANESS _m | Data endianness. | 0, 1 | 0 | Init |
| CFGFLASHBASEADDR | Base address of the flash interface. | Integer | 0x8000000 | Init |
| CFGFLASHEN _m | Flash interface enabled or disabled out of reset. | 0, 1 | 1 | Init |

Table 4-1 Component parameters (continued)

| Parameter name | Description | Allowed values | Default value | Init/ Runtime |
|---------------------------------|---|----------------|---------------|---------------|
| CFGFLASHIMP | Flash region present in memory map. | 0, 1 | 1 | Init |
| CFGFLASHPROTEN | Flash memory protection enable out of reset. | true, false | false | Init |
| CFGFLASHPROTIMP | Flash memory protection support. | 0, 1 | 0 | Init |
| CFGINITREG | Program-visible registers initialized to known value out of reset. | 0, 1 | 0 | Init |
| CFGL1CACHEINVDIS _m | Automatic post-reset L1 cache invalidate disable. | 0, 1 | 1 | Init |
| CFGLLPPBASEADDR | Base address of the LLPP. | Integer1 | 0xB0000 | Init |
| CFGLLPPIMP | LLPP region present in memory map. | 0, 1 | 1 | Init |
| CFGLLPPSIZE | Region size of the LLPP. See the Cortex-R52 TRM for more information. | 4 bits | 0xD | Init |
| CFGMPIDRAFF1 | Cluster ID at affinity level 1. | 8 bits | 0 | Init |
| CFGMPIDRAFF2 | Cluster ID at affinity level 2. | 8 bits | 0 | Init |
| CFGMRPEN | MRP enable. | 0, 1 | 0 | Runtime |
| CFGPERIPHBASE | Base address of the memory-mapped registers, which is principally the interrupt distributor control. | Integer1 | 0xE1E00000 | Init |
| CFGGRAMPROTEN | RAM (caches and TCMs) memory protection enable out of reset. | true, false | false | Init |
| CFGTCMBOOT _m | ATCM enabled and at address 0x0 out of reset. | 0, 1 | 0 | Init |
| CFGTHUMBEXCEPTIONS _m | Instruction set state (A32 or T32) and value of HSCTLR.TE out of reset. | 0, 1 | 0 | Init |
| CFGVECTABLE _m | Vector table base address out of reset. | Integer1 | 0x20 | Init |
| CLREXMONREQ | Clearing of the external global exclusive monitor request. When asserted, this signal acts as a WFE wake-up event to all cores. | 0, 1 | 0 | Runtime |
| CNTVALUEB | Global system counter value in binary format. | Integer1 | 0 | Runtime |
| COREPREQ _m | P-channel request. | 0, 1 | 0 | Runtime |
| COREPSTATE _m | P-channel state. | 0, 1 | 0 | Runtime |
| CPUHALT _m | Core waits out of reset before going through reset sequence. Reset to false to fetch instructions. | true, false | false | Runtime |

Table 4-1 Component parameters (continued)

| Parameter name | Description | Allowed values | Default value | Init/ Runtime |
|--------------------------------|--|---|---------------|---------------|
| CTCMSIZE m | Sets CTCM size. | 0 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB | 32 KB | Init |
| CTCM_WAIT_STATES m | Sets CTCM wait state. | 0, 1 | 0 | Init |
| CTMCHIN | CTM input channel interface. | Integer1 | 0 | Runtime |
| CTMCHOUTACK | CTM input channel interface acknowledge. | Integer1 | 0 | Runtime |
| CTMCIHSBYPASS | CTM channel interfaces handshake bypass for each channel. | Integer1 | 0 | Runtime |
| CTMCISBYPASS | CTM channel interfaces synchronization bypass. Same for all channels. | 0, 1 | 0 | Runtime |
| DCACHESIZE m | Sets data cache size. | 4 KB, 8 KB, 16 KB, 32 KB | 32 KB | Init |
| Dump Waveforms | Whether SoC Designer dumps waveforms for this component. | true, false | false | Runtime |
| Dump Waveforms | Whether SoC Designer dumps waveforms for this component. | true, false | false | Runtime |
| DumpMemFsdB_ATCM[0-3] | Dump size parameter for TCM waveform dumping from CPU number 0, 1, 2, or 3. See 4.3 Dumping TCM waveforms on page 4-27 for instructions. | number of bits | 0 | Init |
| DumpMemFsdB_BTCM m -HighBank | Dump size parameter for TCM waveform dumping. | number of bits | 0 | Init |
| DumpMemFsdB_BTCM m -LowBank | Dump size parameter for TCM waveform dumping. | number of bits | 0 | Init |
| DumpMemFsdB_CTCM m -HighBank | Dump size parameter for TCM waveform dumping. | number of bits | 0 | Init |
| DumpMemFsdB_CTCM m -LowBank | Dump size parameter for TCM waveform dumping. | number of bits | 0 | Init |
| EDBGRQ m | Individual processor external debug request. | 0 - f | 0 | Init |
| Enable Debug Messages | Whether debug messages are enabled for the component. | true, false | false | Runtime |
| EVENTI | Event input for processor wake-up from WFE low-power state. | 0, 1 | 0 | Runtime |
| EXTPPI m | External private peripheral interrupts into the GDU. | Integer1 | 0 | Runtime |

Table 4-1 Component parameters (continued)

| Parameter name | Description | Allowed values | Default value | Init/ Runtime |
|---------------------------|---|--------------------------|-----------------------|---------------|
| ICACHESIZE _m | Sets Instruction Cache Size. | 4 KB, 8 KB, 16 KB, 32 KB | 32 KB | Init |
| LATEERRF _m | Late data error. See the Cortex-R52 TRM for details. | 0, 1 | 0 | Runtime |
| MRPBACKPRESS _m | Driven by the slave connected to the MRP. See the Cortex-R52 TRM for details. | 0, 1 | 0 | Runtime |
| PADDRDBG | APB address bus bits[22:2]. | Integer1 | 0 | Runtime |
| PCLKENDBG | APB clock enable. | 0, 1 | 1 | Runtime |
| RAM_PROT _m | RAM protection supported. | 0, 1 | 1 | Init |
| RDATACODEF _m | Read data. | Integer1 | 0 | Runtime |
| SEI _m | Physical system error interrupt into the core. Active HIGH, edge-sensitive. | 0, 1 | 0 | Runtime |
| SPI | Shared peripheral interrupts into the GDU. | Integer1 | 0 | Runtime |
| SYNCREQD _m | Synchronization request from data trace sink. | 0, 1 | 0 | Runtime |
| SYNCREQI _m | Synchronization request from instruction trace sink. | 0, 1 | 0 | Runtime |
| TSVALUEB | Timestamp value. | Integer1 | 0 | Runtime |
| VSEI _m | Virtual system error interrupt into the core. | 0, 1 | 0 | Runtime |
| Waveform File | Name of the waveform file. When enabled, SoC Designer writes accumulated waveforms to the waveform file in the following situations: when the waveform buffer fills, when validation is paused and when validation finishes, and at the end of each validation run. | string | arm_cm_Cortex-R52.vcd | Init |
| Waveform Format | Format of the waveform dump file. | VCD | VCD | Init |
| Waveform Timescale | Sets the timescale to be used in the waveform. | 1 ns | 1 ns | Init |

4.3 Dumping TCM waveforms

The Cycle Model supports views into the TCM to dump TCM waveforms.

Note

Enabling TCM dumping has additional memory requirements for the system running the simulation. Simulation memory usage increases by approximately 1600 times the size of the TCM data being dumped. For example, if you set `DumpMemFsdb_BTCTMm-HighBank` to 200 bits, the simulation system must have an additional 32,000 bits of memory space.

To dump TCM waveforms:

1. Set the parameter `Dump Waveforms` to `True`.
2. Set the parameter `Waveform Format` to `FSDB`.
3. In the `waveform File` parameter, specify the filename, such as `arm_cm_Core.fsdb`.
4. Using the following parameters, set the number of bits to dump from memory (*m* represents the CPU number):
 - `DumpMemFsdb_ATCTMm` for CPU 0, 1, 2, and 3
 - `DumpMemFsdb_BTCTMm-HighBank`
 - `DumpMemFsdb_BTCTMm-LowBank`
 - `DumpMemFsdb_CTCTMm-HighBank`
 - `DumpMemFsdb_CTCTMm-LowBank`

Chapter 5

Debug features

This section describes the debug features supported by the Cycle Model CADI debug interface.

It contains the following sections:

- [5.1 Register information on page 5-29.](#)
- [5.2 Disassembly view on page 5-33.](#)
- [5.3 Memory view on page 5-34.](#)
- [5.4 Hardware profiling on page 5-35.](#)

5.1 Register information

This section describes the supported register views.

AArch32_Core Registers

Table 5-1 AArch32_Core Registers

| Name | Access |
|---------|------------|
| R0 | Read-Write |
| R1 | Read-Write |
| R2 | Read-Write |
| R3 | Read-Write |
| R4 | Read-Write |
| R5 | Read-Write |
| R6 | Read-Write |
| R7 | Read-Write |
| R8_usr | Read-Write |
| R9_usr | Read-Write |
| R10_usr | Read-Write |
| R11_usr | Read-Write |
| R12_usr | Read-Write |
| R15 | Read-Only |
| R13_usr | Read-Write |
| R14_usr | Read-Write |
| R13_svc | Read-Write |
| R14_svc | Read-Write |
| R13_irq | Read-Write |
| R14_irq | Read-Write |
| R8_fiq | Read-Write |
| R9_fiq | Read-Write |
| R10_fiq | Read-Write |
| R11_fiq | Read-Write |
| R12_fiq | Read-Write |
| R8 | Read-Write |
| R9 | Read-Write |
| R10 | Read-Write |
| R11 | Read-Write |
| R12 | Read-Write |
| R13_fiq | Read-Write |

Table 5-1 AArch32_Core Registers (continued)

| Name | Access |
|------------------------|---------------|
| R14_fiq | Read-Write |
| R13_und | Read-Write |
| R14_und | Read-Write |
| R13_abt | Read-Write |
| R14_abt | Read-Write |
| R13_hyp | Read-Write |
| ELR_hyp | Read-Write |
| R13 | Read-Write |
| R14 | Read-Write |
| CPSR_nodbg | Read-Write |
| DSPSR_dbg | Read-Write |
| CPSR | Read-Write |
| SPSR_svc | Read-Write |
| SPSR_irq | Read-Write |
| SPSR_fiq | Read-Write |
| SPSR_und | Read-Write |
| SPSR_abt | Read-Write |
| SPSR_hyp | Read-Write |
| SPSR | Read-Write |
| ExtendedTargetFeatures | Read-Only |
| PC_MEMSPACE | Read-Write |

Aarch32 ID Registers

Table 5-2 Aarch32 ID Registers

| Name | Access |
|-------------|---------------|
| MIDR | Read-Only |
| CTR | Read-Only |
| TCMTR | Read-Only |
| TLBTR | Read-Only |
| MPIDR | Read-Only |
| MPUIR | Read-Only |
| REVIDR | Read-Only |
| ID_PFR0 | Read-Only |
| ID_PFR1 | Read-Only |
| ID_DFR0 | Read-Only |

Table 5-2 Aarch32 ID Registers (continued)

| Name | Access |
|-------------|---------------|
| ID_AFR0 | Read-Only |
| ID_MMFR0 | Read-Only |
| ID_MMFR1 | Read-Only |
| ID_MMFR2 | Read-Only |
| ID_MMFR3 | Read-Only |
| ID_MMFR4 | Read-Only |
| ID_ISAR0 | Read-Only |
| ID_ISAR1 | Read-Only |
| ID_ISAR2 | Read-Only |
| ID_ISAR3 | Read-Only |
| ID_ISAR4 | Read-Only |
| ID_ISAR5 | Read-Only |
| CCSIDR | Read-Only |
| CLIDR | Read-Only |
| AIDR | Read-Only |

AArch32 Control Registers

Table 5-3 AArch32 Control Registers

| Name | Access |
|-------------|---------------|
| SCTLR | Read-Write |
| ACTLR | Read-Only |
| ACTLR2 | Read-Only |
| DFSR | Read-Write |
| IFSR | Read-Write |

AArch32 PERF Registers

Table 5-4 AArch32 PERF Registers

| Name | Access |
|-------------|---------------|
| PMCR | Read-Write |
| PMCNTENSET | Read-Write |
| PMCNTENCLR | Read-Write |
| PMOVSr | Read-Write |
| PMSELR | Read-Write |
| PMCEID0 | Read-Only |
| PMCEID1 | Read-Only |

Table 5-4 AArch32 PERF Registers (continued)

| Name | Access |
|-------------|---------------|
| PMCCNTR | Read-Write |
| PMXEVTYPER | Read-Write |
| PMXVCNTR | Read-Write |
| PMUSERENR | Read-Write |
| PMINTENSET | Read-Write |
| PMINTENCLR | Read-Write |
| PMEVCNTR0 | Read-Write |
| PMEVTYPER0 | Read-Write |
| PMEVCNTR1 | Read-Write |
| PMEVTYPER1 | Read-Write |
| PMEVCNTR2 | Read-Write |
| PMEVTYPER2 | Read-Write |
| PMEVCNTR3 | Read-Write |
| PMEVTYPER3 | Read-Write |
| PMCCFILTR | Read-Write |
| PMOVSSET | Read-Write |

5.2 Disassembly view

SoC Designer Simulator supports a Disassembly view of a program running on the Cycle Model.

To display the Disassembly view in SoC Designer Simulator, right-click on the Cycle Model and select **View Disassembly...** from the context menu. Refer to the *SoC Designer User Guide* (100996) for more information.

By default, the Disassembly view uses a memory space called `Memory`. You can select the memory space to use by setting the `PC_MEMSPACE` register and re-opening the Disassembly view. Set `PC_MEMSPACE` as follows (`Memory` and `AXI_Main` currently show the same data):

- 0 - `Memory`
- 1 - `AXI_Main`
- 2 - `AXI_Flash`. Displays contents of memory attached to the CPU Flash Interface port.
- 3 - `AXI_LLPP`. Displays contents of memory attached to the AXI4 Master Low Latency Peripheral port.

All CADI windows support breakpoints. When double-clicking on the proper location a red dot indicates that a breakpoint is currently active. To remove the breakpoints simply doubleclick on the same location again.

5.3 Memory view

SoC Designer Simulator supports a memory view of a program running on the Cycle Model.

To display the memory view in SoC Designer Simulator, right-click on the Cycle Model and select **View Child Memory for...** from the context menu. Refer to the *SoC Designer User Guide* (100996) for more information.

The following memory views are available (Memory and AXI_Main currently show the same data):

- 0 - Memory
- 1 - AXI_Main
- 2 - AXI_Flash. Displays contents of memory attached to the CPU Flash Interface port.
- 3 - AXI_LLPP. Displays contents of memory attached to the AXI4 Master Low Latency Peripheral port.

Memory views do not include TCM. Memory views are not coherent views; cache is not taken into account.

5.4 Hardware profiling

Profiling data is enabled, and can be viewed using the Profiling Manager, which is accessible via the **Debug** menu in the SoC Designer Simulator. Software profiling is not supported.

The Cycle Model supports the full set of hardware profiling events. See the *Cortex-R52 Processor Technical Reference Manual* (100026) for the list of supported profiling events.