Cortex-R52 Cycle Model

Version 9.2.0

User Guide

ARM®

ARM 101016_0902_00 (ID050417)

Cortex-R52 Processor Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Issue	Date	Confidentiality	Change
0902-00-01	May 2017	Non-Confidential	Release with 9.2.

Non-Confidential Proprietary Notice

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

Preface

A Cycle Model component is a library developed from ARM intellectual property (IP) that is generated through Cycle Model Studio. The Cycle Model then can be used within a virtual platform tool — for example, SoC Designer.

About This Guide

This guide provides all the information needed to configure and use the Cortex-R52 Cycle Model in SoC Designer.

Audience

This guide is intended for experienced hardware and software developers who create components for use with SoC Designer. You should be familiar with the following products and technology:

- SoC Designer
- Hardware design verification
- Verilog or SystemVerilog programming language

Conventions

This guide uses the following conventions:

Convention	Description	Example
`courier`	Commands, functions, variables, routines, and code examples that are set apart from ordinary text.	`sparseMem_t SparseMemCreate-New();`
italic	New or unusual words or phrases appearing for the first time.	*Transactors* provide the entry and exit points for data ...
bold	Action that the user performs.	Click **Close** to close the dialog.
<text>	Values that you fill in, or that the system automatically supplies.	<platform>/ represents the name of various platforms.
[text]	Square brackets [ ] indicate optional text.	`$CARBON_HOME/bin/modelstudio [ <filename> ]`
[text1 | text2]	The vertical bar | indicates "OR," meaning that you can supply text1 or text 2.	`$CARBON_HOME/bin/modelstudio [<name>.symtab.db | <name>.ccfg ]`

Also note the following references:

- References to C code implicitly apply to C++ as well.
- File names ending in .cc, .cpp, or .cxx indicate a C++ source file.

Further reading

This section lists related publications. The following publications provide information that relate directly to SoC Designer:

- *SoC Designer Installation Guide*
- *SoC Designer User Guide*
- *SoC Designer Standard Model Library Reference Manual*

The following publications provide reference information about ARM products:

- *AMBA® Specification*

See http://infocenter.arm.com/help/index.jsp for access to ARM documentation.

The following publications provide additional information on simulation:

- IEEE 1666™ SystemC Language Reference Manual, (IEEE Standards Association)
- SPIRIT User Guide, Revision 1.2, SPIRIT Consortium.

Chapter 1

Using the Cycle Model in SoC Designer

This chapter describes the functionality of the ARM® Cortex-R52 Cycle Model, and how to use it in SoC Designer.

This chapter contains the following sections:

1.1 Functionality of the Cortex-R52 Cycle Model

The ARM Cortex-R52 Cycle Model simulates the Cortex-R52 MPCore processor. This section provides a summary of the functionality of the Cycle Model compared to that of the hardware, and the performance and accuracy of the Cycle Model.

This section describes:

- Cortex-R52 Features
- Unsupported Hardware Features
- Features Additional to the Hardware

1.1.1 Cortex-R52 Features

This section describes the supported functionality of the Cortex-R52 Cycle Model:

- Configurations of up to 4 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 Memory Protection Unit (MPU) regions.
- Floating Point Unit (FPU).
- ETM interface, including register slice between the processor and ETM interface.
- Configurable ICache and DCache sizes.
- RAM protection.

1.1.2 Unsupported Hardware Features

The following features of the Cortex-R52 hardware are *not implemented* in this release of 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).

1.1.3 Features Additional to the Hardware

To enhance usability, the following features have been added to the Cycle Model, which do not exist in the Cortex-R52 hardware:

- Waveform dumping using the waveform-related parameters described in Table 1-3.

1.2 Adding and Configuring the SoC Designer Component

The following topics briefly describe how to use the component. See the *SoC Designer User Guide* for more information.

- SoC Designer Component Files
- Adding the Cycle Model to the Component Library
- Adding the Component to the SoC Designer Canvas

1.2.1 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 provided component files are listed below:

Table 1-1 SoC Designer Component Files

Platform	File	Description
Linux	maxlib.lib<*model_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
	libR7_save_arch_handler.a	Library file for Architectural cache/restore support.
Windows	maxlib.lib<*model_name*>.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
	R7_save_arch_handler.lib	Library file for Architectural cache/restore support.

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

1.2.2 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, perform the following steps:

1. Launch SoC Designer Canvas.
2. From the *File* menu, select **Preferences**.
3. Click on **Component Library** in the list on the left.
4. Under the *Additional Component Configuration Files* window, click **Add**.
5. Browse to the location where the SoC Designer Cycle Model is located and select the component configuration file:
   - `maxlib.lib<model_name>.conf` (for Linux)
   - `maxlib.lib<model_name>.windows.conf` (for Windows)
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.*

1.2.3 Adding the Component to the SoC Designer Canvas

Locate the component in the *Component Window* and drag it out to the Canvas. The component's appearance may vary depending on your specific device configuration.

Additional ports are provided depending on the model RTL configuration file used to create the Cycle Model.

1.3 ESL Ports

This section describes the differences between the pins listed in the *ARM Cortex-R52 Technical Reference Manual* (TRM) and those on the Cortex-R52 Cycle Model. Certain hardware pins have been converted to init-time Cycle Model parameters.

- Available Component ESL Ports — Describes ports that have been added to the Cycle Model, such as clocks and resets required by SoC Designer, or those created by wrapping multiple hardware pins into transactors.
- Tied Pins — Describes pins that are tied under certain conditions.

1.3.1 Available Component ESL Ports

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

Note: Most ESL component port values can be set using a component parameter. In these cases, the parameter value is used whenever the ESL port is not connected. If the port is connected, the connection value takes precedence over the parameter value.

Table 1-2 ESL Component Ports

ESL Port	Description	Type
ACP_Slave_AXISlave	ACP Slave debug port (AXI)	Transaction Slave
APB_Slave_Debug_APB	APB Slave debug port (APB)	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

1.3.2 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 0NIDEN0 (high)
- NIDEN0 (high)
- DBGEN0 (high)
- HIDEN0 (high)
- HNIDEN0 (high)
- ACLKENF0 (high)
- ACLKENM0 (high)
- ACLKENP0 (high)
- PCLKENDBG (high)
- ACLKENS (high)
- CNTCLKEN (high)

1.4 Setting Component Parameters

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's parameters:

1. In the 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 you enabled when creating the component.
2. In the *Parameters* window, double-click the **Value** field of the parameter that you want to modify.
3. If it is a text field, type a new value in the *Value* field. If a menu choice is offered, select the desired option. The parameters are described in Table 1-3.

Note: In Table 1-3, "m" indicates the CPU number (0, 1, 2, or 3).

Table 1-3 Component Parameters

Name	Description	Allowed Values	Default Value	Runtime/ Init
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]	These parameters should be left at their default values.	—	0	Init
ACP_Slave_AXISlave axi_start[0-5]		—	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

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/ 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
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

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/ Init
Carbon 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[1]	0x1000000	Init
CFGCLUSTERUTID	Cluster Unique Transaction IDentifier for the purpose of interconnect protection.	0 - 3	0	Init
CFGDBGROMADDR	Debug ROM table address.	Integer[1]	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[1]	0x8000000	Init
CFGFLASHEN*m*	Flash interface enabled or disabled out of reset.	0, 1	1	Init
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.	Integer[1]	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.	Integer[1]	0xE1E00000	Init
CFGRAMPROTEN	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.	Integer[1]	0x20	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/ 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.	Integer[1]	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
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.	Integer[1]	0	Runtime
CTMCHOUTACK	CTM input channel interface acknowledge.	Integer[1]	0	Runtime
CTMCIHSBYPASS	CTM channel interfaces handshake bypass for each channel.	Integer[1]	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
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.	Integer[1]	0	Runtime
ICACHESIZE*m*	Sets Instruction Cache Size.	4 KB, 8 KB, 16 KB, 32 KB	32 KB	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/ 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].	Integer[1]	0	Runtime
PCLKENDBG	APB clock enable.	0, 1	1	Runtime
RAM_PROT*m*	RAM protection supported.	0, 1	1	Init
RDATACODEF*m*	Read data.	Integer[1]	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.	Integer[1]	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.	Integer[1]	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 Time-scale	Sets the timescale to be used in the waveform.	1 ns	1 ns	Init

1. See the *Cortex-R52 Technical Reference Manual* for details.

1.5 Debug Features

The Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the memory. In SoC Designer Simulator, right-click on the Cycle Model and select the appropriate menu entry.

The following topics are discussed in this section:

- Memory Information

1.5.1 Memory Information

Table 1-4 describes the available memory space views. Note that address range and access size are configuration-dependent.

Table 1-4 Memory Spaces

Name	Address Range	Access Size
AXI_Main	configuration-dependent	8
AXI_Flash	configuration-dependent	8
AXI_LLPP	configuration-dependent	8

Third Party Software Acknowledgement

ARM acknowledges and thanks the respective owners for the following software that is used by our product:

- **ELF (Executable and Linking Format) Tool Chain Product**

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.