Cortex®-R8 Processor Cycle Model

Version 9.6.0

User Guide

arm

101015_0906_00 (ID110817)

Cortex-R8 Processor Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Issue	Date	Confidentiality	Change
0903-00-01	June 2017	Non-Confidential	Release with 9.3.
0906-00	November 2017	Non-Confidential	Release with 9.6

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-R8 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 User Guide* (100996)

The following publications provide reference information about Arm products:

- *Arm Cortex-R8 Technical Reference Manual* (100400)
- *AMBA® Specification* (IHI 0011)

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-R8 Cycle Model, and how to use it in SoC Designer.

This chapter contains the following sections:

1.1 Functionality of the Cortex-R8 Cycle Model

The Arm Cortex-R8 Cycle Model simulates the Cortex-R8 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-R8 features
- Unsupported hardware features
- Features additional to the hardware
- Fast Models-related restrictions

1.1.1 Cortex-R8 features

This section describes the supported functionality of the Cortex-R8 Cycle Model:

- Configurations of up to 4 CPUs are supported.
- Configurable number of interrupts (0 to 480 in increments of 32).
- Single or dual AXI master port.
- Access to Tightly Couple Memory (TCM) via slave port.
- A subset of the available hardware registers is supported. Refer to “Register information” on page 24 for details.
- Variable ICache and DCache sizes.
- Variable ITCM and DTCM sizes.
- Accelerator Coherency Port (ACP) and ACP bridge with configurable ID size.
- 12, 16, 20, or 24 Memory Protection Unit (MPU) regions.
- Floating Point Unit (FPU).
- ETM interface, including register slice between the processor and ETM interface.
- Fast Peripheral Port (FPP) is supported.

1.1.2 Unsupported hardware features

The following features of the Cortex-R8 hardware are *not implemented* in this release of the Cortex-R8 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). The default is 512.
- Configurable size of the PREDictor (PRED) RAM. The default is 4096.
- 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-R8 hardware:

- Waveform dumping, including dumping of TCM memories, using the waveform-related parameters described in Table 1-3.
- Support for viewing register values (see Register information).
- Support for viewing memory spaces (see Memory information).
- Support for viewing disassembly data (see Disassembly view).
- Run to debug point feature. This feature forces the debugger to advance the processor to the debug state instead of having the Cycle Model get into a non-debuggable state. See Run To Debug Point feature for more information.

1.1.4 Fast Models-related restrictions

These restrictions and workarounds apply when using the Cortex-R8 Cycle Model with the SoC Designer Fast Models System Creator:

- The IRQS port is not automatically mapped to the Fast Models port INTS. Perform this mapping manually as a Signal Slave Array; for example, a Cortex-R7 with 32 IRQS would be a Signal Slave Array with Width set to 32.
- In the Fast Model Configuration GUI, on the Basic tab in the Clocks panel, manually map the Clock/2 port to Periphclk_in.
- The single AXI_Slave_TCM_Bus cannot be mapped to both ITCM and DTCM Fast Models ports. Only manual mapping to either ITCM or DTCM for one of the CPUs is supported. If you need to connect both ITCM and DTCM ports, contact Arm Technical Support.

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 Available component ports

Table 1-2 describes the ESL ports that are exposed in SoC Designer. Some ports may or may not appear depending on whether you are using the multiprocessor or uniprocessor version of the Arm hardware. See the *Arm Cortex-R8 Technical Reference Manual* (100400) for more information.

All pins that are not listed in this table have been either tied or disconnected for performance reasons.

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

Table 1-2 ESL component ports

ESL Port	Description	Type
ACLKENSC	Clock bus enable for the AXI bus that enables the AXI slave port to operate at integer ratios of the system clock	Signal Slave
ACLKENST	Clock bus enable for the AXI bus that enables the AXI interface to operate at either Integer ratios of the system clock or half integer ratios of the system clock.	Signal Slave
AFREADYMD*m*	ATB interface FIFO flush finished for data trace.	Signal Master
AFREADYMI*m*	ATB interface FIFO flush finished for instruction trace.	Signal Master
AFVALIDMD*m*	ATB interface FIFO flush request (data trace).	Signal Slave
AFVALIDMI*m*	ATB interface FIFO flush request (instruction trace).	Signal Slave
APB Slave DBG	APB Slave debug port	Transaction Slave
ATBYTESMD*m*	Size of ATDATA (data trace).	Signal Master
ATBYTESMI*m*	Size of ATDATA (instruction trace).	Signal Master
ATDATAMD*m*	ATB interface data (data trace).	Signal Master
ATDATAMI*m*	ATB interface data (instruction trace).	Signal Master
ATIDMD*m*	ATB interface trace source ID (data trace).	Signal Master
ATIDMI*m*	ATB interface trace source ID (instruction trace).	Signal Master
ATREADYMD*m*	ATDATA can be accepted (data trace).	Signal Slave
ATREADYMI*m*	ATDATA can be accepted (instruction trace).	Signal Slave
ATVALIDMD*m*	ATB interface data valid (data trace).	Signal Master
ATVALIDMI*m*	ATB interface data valid (instruction trace).	Signal Master
AXI3_Master_PERI	AXI3 Master peripheral port.	Transaction Master
AXI3_Master_PORT[0, 1]	AXI3 Master port (number of ports is configuration dependent).	Transaction Master
AXI3_Slave_ACP	AXI3 Slave port.	Transaction Slave
AXI3_Slave_TCM	AXI3 TCM Slave port.	Transaction Slave
AXI_Master_FPP*m*	AXI3 Master fast peripheral port.	Transaction Master
CIHSBYPASS	Channel interface HS bypass.	Signal Slave
CLK	Main clock of the Cortex-R8 MPCore processor.	Signal Slave

Table 1-2 ESL component ports (continued)

ESL Port	Description	Type
clk_in	This port is used internally. Leave unconnected.	Clock Slave
COMMRX	Enable interrupt-driven communications over the DTR.	Signal Master
COMMTX		Signal Master
CPUCLKOFF	Individual processor clock enable. One bit per processor. N:0 where N is number of cores	Signal Slave
CTCLKOFF	Used to control the CoreSight™ debug logic clock, that is, CTI0, CTI1, CTM, and ROM table. Used to deassert the reset synchronously when leaving reset, but not used for clock enable.	Signal Slave
CTICHIN	Channel in.	Signal Slave
CTICHINACK	Channel in acknowledge.	Signal Master
CTICHOUT	Channel out.	Signal Master
CTICHOUTACK	Channel out acknowledge.	Signal Slave
DBGACK	Individual processor debug acknowledge signal.	Signal Master
DBGCLKOFF	Individual processor debug clock control, active-LOW: 0 - Processor debug clock is enabled. 1 - Processor debug clock is stopped.	Signal Slave
DBGCPUDONE	Acknowledges that corresponding processor has entered debug state and that all previous non-debug state memory accesses are complete.	Signal Master
DBGNOPWRDWN	Output reflecting the value of DBGPRCR[0].	Signal Master
DBGROMADDR	CoreSight System configuration. Specifies bits[31:12] of the ROM table physical address. If the address cannot be determined, tie this signal off to zero.	Signal Slave
DBGROMADDRV	Valid signal for DBGROMADDR. If the address cannot be determined, tie this signal LOW.	Signal Slave
DBGSELFADDR	Specifies bitsof the two's complement signed offset from the ROM Table physical address to the physical address where the debug registers are memory-mapped.	Signal Slave
DBGSELFADDRV	Valid signal for DBGSELFADDR.	Signal Slave
DBGSWENABLE	When LOW only the external debug agent can modify debug registers.	Signal Slave
EDBGRQ	Individual processor external debug request.	Signal Slave
ETM*m*CLKOFF	ETM reset signal.	Signal Slave
ETMACTIVE*m*	Trace is being output.	Signal Master
ETMIFEN*m*	Power control for ETM processor trace interface.	Signal Master
ETMPWRUPREQ*m*	Request to maintain power to ETM.	Signal Master
EVENTI	Event input for processor wake-up from WFE state.	Signal Slave
EVENTO	Macrocell standby and wait for event signal, event output.	Signal Master

Table 1-2 ESL component ports (continued)

ESL Port	Description	Type
extSemi_cpu*m*	Semihost port. Not enabled.	Transaction Master
fiq	Individual core legacy FIQ request input lines. Refer to the TRM for more information.	Signal Slave
fiqout	Active-LOW FIQ outputs from the internal GIC to the appropriate core. Refer to the TRM for more information.	Signal Master
FPUFLAGS*m*	Floating-Point Unit output flags for the corresponding core.	Signal Master
INCLKENM[0, 1]	Clock bus enable for the AXI bus. Refer to the TRM for more information.	Signal Slave
INCLKENMFP*m*	Clock enable.	Signal Slave
INCLKENMP	Clock enable.	Signal Slave
irq	IRQ Interrupt Port. One bit per processor. N:0 where N is number of cores.	Signal Slave
irqout	Active-LOW **IRQ** outputs from the internal GIC to the appropriate core. Refer to the TRM for more information.	Signal Master
IRQS	Interrupt distributor interrupt lines.	Signal Slave
nCPURESET	Individual core resets.	Signal Slave
nCTIIRQ	Active-LOW interrupt from CTI	Signal Master
nCTRESET	Reset for CoreSight debug logic, that is, CTI0, CTI1, CTI2, CTI3, CTM, and ROM table.	Signal Slave
nDBGRESET	Core debug logic resets.	Signal Slave
nETM*m*RESET	Reset for associated ETM.	Signal Slave
nPERIPHRESET	Timer and interrupt controller reset.	Signal Slave
nSCURESET	SCU global reset.	Signal Slave
nWDRESET	Core watchdog reset.	Signal Slave
OUTCLKENM[0, 1]	Clock bus enable for the AXI bus. See the TRM for details.	Signal Slave
OUTCLKENMFP*m*	Clock enable for AXI fast peripheral clock.	Signal Slave
OUTCLKENMP	Clock enable for AXI low-latency peripheral clock.	Signal Slave
PADDRDBG31	CoreSight APB devices Programming address.	Signal Slave
PERIPHCLK	Clock for the timer, watchdog, and interrupt controller. Drive PERIPHCLK with a CDIV component; the CDIV clock ratio must be set to 2. Refer to the *SoC Designer Standard Component Library Reference Manual* (101026) for information about the CDIV component. Refer to the *Arm Cortex-R8 Technical Reference Manual* (100400) for additional information about synchronizing the clocks.	Clock Slave

Table 1-2 ESL component ports (continued)

ESL Port	Description	Type
PERIPHCLKEN	Clock enable for timer, watchdog, and interrupt controller. Drive this signal using the MxSigDriver component, which is part of the SoC Designer Standard Model Library. To enable correct input timing for CLK, PERIPHCLK, and PERIPHCLKEN, ensure that the Initial Value parameter on the MxSigDriver component is set to 1. Refer to the *SoC Designer Standard Component Library Reference Manual* (101026) for more information.	Signal Slave
PMUEVENT*m*	PMU event bus for processor.	Signal Master
PMUIRQ	Processor interrupt requests by system metrics.	Signal Master
PMUPRIV	External Performance Monitoring Unit Gives the status of the Cortex-R8 MPCore processor.	Signal Master
PWRCTLO*m*	Reset value for core 0 status field, bits[1:0] of SCU CPU Power Status Register.	Signal Master
SCUEVABORT	Indicates an external abort has occurred during a coherency writeback.	Signal Master
SCUIDLE	Indicates the SCU is idle. See the TRM for details.	Signal Master
SMPnAMP	Indicates AMP or SMP mode for each processor.	Signal Master
SRENDM[0, 1]	Speculative read information from optional L2 Cache Controller.	Signal Slave
SRIDM[0, 1]	ID for speculative reads returned by L2 Cache Controller.	Signal Slave
STANDBYWFE	Indicates if processor is in Standby WFE.	Signal Master
STANDBYWFI	Indicates if processor is in Standby WFI.	Signal Master
SYNCREQD*m*	Synchronization request from data trace sink.	Signal Slave
SYNCREQI*m*	Synchronization request from instruction trace sink.	Signal Slave
TSSIZE	When HIGH (0b1), timestamp is 64 bits. When LOW (0b0), timestamp is 48 bits.	Signal Slave
TSVALUE	Timestamp value.	Signal Slave
WDRESETREQ	Processor watchdog reset requests.	Signal Master

1.4 Setting component parameters

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

- Changing parameter settings in SoC Designer
- Component parameters
- Dumping TCM waveforms

1.4.1 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'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.

For information about viewing TCM memory in FSDB using waveform-related parameters, see "Dumping TCM waveforms" on page 23.

1.4.2 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: In Table 1-3, "m" indicates CPU 0, 1, 2, or 3.

Table 1-3 Component parameters

Name	Description	Allowed Values	Default Value	Runtime/Init
ACLKENSC	Clock bus enable for the AXI bus that enables the AXI slave port to operate at integer ratios of the system clock.	0, 1	1	Runtime
ACLKENST	Clock bus enable for the AXI bus that enables the AXI interface to operate at integer ratios of the system clock.	0, 1	1	Runtime
AFVALIDMD*m*	ATB interface FIFO flush request (data trace)	0, 1	0	Runtime
AFVALIDMI*m*	ATB interface FIFO flush request (instruction trace)	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_DBG Base Address	APB Slave debug base address	Integer	0	Init
APB_Slave_DBG Enable Debug Messages	Whether debug messages are enabled on the APB Slave port.	true, false	false	Runtime
APB_Slave_DBG Protocol Variant	Protocol Variant in use for APB.	APB3	APB3	Init
APB_Slave_DBG Size	APB Slave debug size	Integer	0	Init
ATREADYMD*m*	ATDATA can be accepted (data trace)	0, 1	0	Runtime
ATREADYMI*m*	ATDATA can be accepted (instruction trace)	0, 1	0	Runtime
AXI3_Master_PERI Enable Debug Messages	Enables AXI3_Master_PERI port debug.	true, false	false	Runtime
AXI3_Master_PORT[0,1] Enable Debug Messages	Enables AXI3_Master port debug.	true, false	false	Runtime
AXI3_Slave_ACP axi_size[0-5]	These parameters should be left at their default values.	—	0	Runtime
AXI3_Slave_ACP axi_start [0-5]		—	0	Runtime

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/Init
AXI3_Slave_ACP Enable Debug Messages	Enables AXI3 Slave ACP port debug.	true, false	false	Runtime
AXI3_Slave_TCM axi_size[0-5]	These parameters should be left at their default values.	—	0	Runtime
AXI3_Slave_TCM axi_start[0-5]		—	0	Runtime
AXI3_Slave_TCM Enable Debug Messages	Enables AXI3_Slave TCM port debug.	true, false	false	Runtime
AXI_Master_FPP*m* Enable Debug Messages	Enables AXI_Master_FPP port debug.	true, false	false	Runtime
ARM CycleModels DB Path	Sets the directory path to the database file.	not used	empty	Init
CFGEND	Individual core endianness configuration. See the TRM for details.	0, 1	0	Init
CFGNMFI	Configuration of FIs to be nonmaskable. See the TRM for details.	0, 1	0	Init
CIHSBYPASS	Trace device interface HS bypass.	0-15	0	Init
CLUSTERID	Value read in Cluster ID field. See the TRM for details.	0 - f	0	Init
CTICHIN	Trace device Channel in.	0-15	0	Init
CTICHOUTACK	Trace device Channel out acknowledge.	0-15	0	Init
DBGROMADDR	External debug device CoreSight System configuration. Specifies bits[31:12] of the ROM.	Table physical address.	0	Init
DBGROMADDRV	Valid signal for DBGROMADDR.	0, 1	0	Init
DBGSELFADDR	Specifies bits[31:17] of the two's complement signed offset from the ROM Table physical address to the physical address where the debug registers are memory-mapped.	Integer	0	Init
DBGSELFADDRV	Valid signal for DBGSELFADDR.	0, 1	0	Runtime
DBGSWENABLE	When LOW only the external debug agent can modify debug registers.	0, 1	0	Runtime
Dump Waveforms	Whether SoC Designer dumps waveforms for this component.	true, false	false	Runtime
DumpMemFsdb_CPU*m*-ITCM-Bank[0-3]	Dump size parameters for TCM waveform dumping. See “Dumping TCM waveforms” on page 23 for instructions.	number of bits	0	Init
DumpMemFsdb_CPU*m*-DTCM-HighBank		number of bits	0	Init
DumpMemFsdb_CPU*m*-DTCM-LowBank		number of bits	0	Init

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/Init
EDBGRQ	Individual processor external debug request.	0 - f	0	Init
Enable Debug Messages	Whether debug messages are enabled for the component.	true, false	false	Runtime
FPFILTEREND*m*	For cores configured with an AXI fast peripheral port. Specifies the end address for address filtering at reset on the AXI fast peripheral port.	Integer	0x3100000	Init
FPFILTERSTART*m*	For cores configured with an AXI fast peripheral port. Specifies the start address for address filtering at reset on the AXI fast peripheral port.	Integer	0x3000000	Init
INCLKENM[0, 1]	Clock bus enable for the AXI bus. See the TRM for details.	0, 1	1	Runtime
INCLKENMFP*m*	Clock enable for theAXI fast peripheral clock.	0, 1	1	Runtime
INCLKENMP	AXI low-latency peripheral clock enable.	0, 1	1	Runtime
INITRAM*m*	Sets reset value of TCM enable bit for core *m*.	true, false	false	Init
MFILTEREN	For configurations with two master ports. Enables filtering of address ranges at reset for AXI Master port 1. See the TRM for details.	true, false	false	Init
MFILTEREND	For configurations with two master ports. Specifies the end address for address filtering at reset on AXI master port 1.	Integer	0xfff00000	Init
MFILTERSTART	For configurations with two master ports. Specifies the start address for address filtering at reset on AXI master port 1.	Integer	0x0	Init
nCPUHALT	Individual core input. See the TRM for details.	0 - f	0xF	Runtime
negLogic	Enables active low interrupts.	true, false	false	Init
OUTCLKENM[0, 1]	Clock bus enable for the AXI bus. See the TRM for details.	0, 1	1	Runtime
OUTCLKENMFP*m*	AXI fast peripheral clock enable.	0, 1	1	Runtime
OUTCLKENMP	AXI low-latency peripheral clock enable.	0, 1	1	Runtime
PADDRDBG31	CoreSight APB devices Programming address.	0x1f	0	Init

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/Init
PERIPHBASE	Specifies the base address for Timers, Watchdogs, Interrupt Controller, and SCU registers. Set to the base address shifted right by 13 bits (bits 12:0 are ignored).	Integer	0xd3080000	Init
PERIPHCLKEN	Clock enable for the interrupt controller and timers.	0, 1	1	Runtime
PFILTEREND	For use with configurations with the AXI Master peripheral port. Specifies the end address for address filtering at reset on the AXI peripheral port.	0, 1	0xd3100000	Init
PFILTERSTART	For use with configurations with the AXI Master peripheral port. Specifies the start address for address filtering at reset on the AXI peripheral port.	Integer	0xd3000000	Init
SRENDM[0, 1]	Speculative read information from optional L2 Cache Controller.	0 - f	0	Init
SRIDM[0, 1]	ID for speculative reads returned by L2 Cache Controller.	If the ACP is implemented, [n] is [AXISC_ID_BIT:0], that is, the number of ACP ID bits + 1. If the ACP is not implemented, [n] is [4:0].	0	Init
SYNCREQD*m*	Synchronization request from data trace sink.	0, 1	0	Runtime
SYNCREQI*m*	Synchronization request from instruction trace sink.	0, 1	0	Runtime
TEINIT	Individual processor out-of-reset default exception handling state.	Integer	0	Init
TSSIZE	When HIGH (0b1), timestamp is 64 bits. When LOW (0b0), timestamp is 48 bits.	0, 1	0	Runtime
TSVALUE	Timestamp value	Integer	0	Runtime
VINITH*m*	Use high vector addresses. (Individual processor control of the location of the exception vectors at reset.) *n* is the max CPU ID.	Integer	0	Init

Table 1-3 Component parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime/Init
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-R8.v	Init
Waveform Format	Format of the waveform dump file.	VCD, FSDB	VCD	Init
Waveform Time-scale	Sets the timescale to be used in the waveform.	1 ns	1 ns	Init

1.4.3 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_BTCMm-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:
   - **DumpMemFsdb_CPUm-ITCM-Bank** for CPU 0, 1, 2, and 3
   - **DumpMemFsdb_CPUm-DTCM-HighBank**
   - **DumpMemFsdb_CPUm-DTCM-LowBank**

1.5 Debug features

The Cortex-R8 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the registers and memory, and display disassembly for programs running on the Cycle Model in the SoC Designer Simulator or any debugger that supports CADI. A view can be accessed in SoC Designer Simulator by right clicking on the Cycle Model and choosing the appropriate menu entry.

The following topics are discussed in this section:

- Register information
- Run To Debug Point feature
- Memory information
- Disassembly view

1.5.1 Register information

This section describes the registers supported with this release. Registers are grouped into sets according to functional area, as described in the following sections:

- Core registers
- Control registers
- GIC_Core registers
- GIC_Distributor registers
- Global_Timer registers
- ID registers
- MPU registers
- Perf registers
- SCU registers
- Timer_WD registers
- VFP registers

For detailed descriptions of these registers, refer to the *Arm® Cortex™-R8 Technical Reference Manual* (100400) for the appropriate processor core.

Note: Registers are accurate only at debuggable points. While SoC Designer grays out the register view when the processor is not at a debuggable point, values are still visible. Due to the speculative nature of the processor pipeline, these values are not guaranteed to be accurate.

In general, you can write to a register only at a debuggable point. If a value is deposited at any other point, it may not be correctly propagated.

1.5.1.1 Core registers

The Core group contains the Arm architectural registers.

Table 1-4 Core registers

Name	Description	Type
R0-R14	General purpose registers	Read/Write
R15	PC	Read/Write
CPSR	Current Program Status Register	Read/Write
R8_usr — R14_usr	R8-R14 in USR mode	Read/Write
R13_irq — R14_irq	R13/R14 in IRQ mode	Read/Write
SPSR_irq	Saved program status register in IRQ mode	Read/Write
R8_fiq — R14_fiq	R8-R14 in FIQ mode	Read/Write
SPSR_fiq	Saved program status register in FIQ mode	Read/Write
R13_svc — R14_svc	R13/R14 in SVC mode	Read/Write
SPSR_svc	Saved program status register in SVC mode	Read/Write
R13_abt — R14_abt	R13/R14 in ABT mode	Read/Write
SPSR_abt	Saved program status register in ABT mode	Read/Write
R13_und — R14_und	R13/R14 in UND mode	Read/Write
SPSR_und	Saved program status register in UND mode	Read/Write
SPSR	SPSR value in current mode	Read/Write

1.5.1.2 Control registers

The Control group (Table 1-5) is used for system control and configuration using the CP15 Control registers.

Table 1-5 Control registers

Name	Description	Type
SCTLR	System Control Register	Read/Write
ACTLR	Auxiliary Control Register	Read/Write
CPACR	Coprocessor Access Control Register	Read/Write[1]
DFSR	Data Fault Status Register	Read/Write
IFSR	Instruction Fault Status Register	Read/Write
DFAR	Data Fault Address Register	Read/Write
IFAR	Instruction Fault Address Register	Read/Write
ITCMRR	Instruction TCM Region Register	Read/Write[2]
DTCMRR	Data TCM Region Register	Read/Write[2]
CONTEXTIDR	Context ID Register	Read/Write
TPIDRURW	User Read/Write Software Thread Register	Read/Write

Table 1-5 Control registers (continued)

Name	Description	Type
TPIDRURO	User Read Only Software Thread Register	Read/Write
TPIDRPRW	Privileged Only Software Thread Register	Read/Write
PCR	Power Control Register	Read/Write
CBAR	Configuration Base Address Register	Read/Write

1. Read/Write for FPU variants. Otherwise, Read Only.
2. Read/Write if TCM size does not equal 0. Otherwise, Read Only.

1.5.1.3 GIC_Core registers

Table 1-6 describes the supported interrupt interface registers.

Table 1-6 GIC_Core registers

Name	Description	Type
ICCIIDR	CPU Interface Implementer Identification Register	Read Only
ICCICR	CPU Interface Control Register	Read/Write
ICCPMR	Interrupt Priority Mask Register	Read/Write
ICCBPR	Binary Point Register	Read/Write

1.5.1.4 GIC_Distributor registers

Table 1-7 describes the supported distributor registers.

Table 1-7 GIC_Distributor registers

Name	Description	Type
ICDDCR	Distributor Control Register	Read/Write
ICDICTR	Interrupt Controller Type Register	Read Only
ICDISER_sw	ICDISER0 Soft Interrupt Set-Enable Registers	Read/Write
ICDICER_sw	ICDICER0 Soft Interrupt Clear-Enable Registers	Read/Write
ICDISPR_sw	ICDISPR0 Soft Interrupt Set-Pending Registers	Read/Write
ICDICPR_sw	ICDICPR0 Soft Interrupt Clear-Pending Registers	Read/Write
ICDISER_spi_set*x*[1]	External SPI registers.	Read/Write
ICDICER_spi_set*x*[1]		
ICDISPR_spi_set*x*[1]		
ICDICPR_spi_set*x*[1]		
ICDIPR_*n*_*m*_sw[2]	Interrupt Priority distributor registers for soft interrupts.	Read/Write
ICDIPR_*n*_*m*_spi_set*x*[1, 2]	Interrupt Priority distributor register sets.	Read/Write

Table 1-7 GIC_Distributor registers (continued)

Name	Description	Type
ICDIPTR_*n*_*m*_spi_set*x*[1, 2]	Distributor registers for interrupt target ICDIPTR.	Read/Write
ICDICFR_*n*_*m*_spi_set*x*[1, 2]	Interrupt Configuration Registers.	Read/Write

1. Set0 corresponds to the first 32 external SPIs (31-0). Set1 corresponds to the next 32 external SPIs (63-32), and so on. Number of sets depends on configured number of SPIs.
2. There are four interrupts per 32-bit register; the digits in the register name denote the location of the interrupt block; i.e., ICDIPR_7_4_sw denotes the interrupts at locations 7, 6, 5, and 4.

1.5.1.5 Global_Timer registers

Table 1-8 describes the Global_Timer registers.

Table 1-8 Global_Timer registers

Name	Description	Type
GT_TIMER_LO	Lower Timer Counter Register	Read/Write
GT_TIMER_HI	Upper Timer Counter Register	Read/Write
GT_TCR	Global Timer Control register	Read/Write
GT_TSR	Global Timer Interrupt Status Register	Read/Write
GT_COMP_LO	Lower Comparator Value Register	Read/Write
GT_COMP_HI	Upper Comparator Value Register	Read/Write
GT_AUTOINC	Auto-increment Register	Read/Write

1.5.1.6 ID registers

The ID group (Table 1-8) contains registers that identify the device.

Table 1-9 ID registers

Name	Description	Type
MIDR	Main ID	Read Only
MPIDR	Multiprocessor Affinity Register	Read Only
CLIDR	Cache Level ID Register	Read Only
CSSELR	Cache Size Selection Register	Read/Write
CTR	Cache Type Register	Read Only
TCMTR	TCM Type Register	Read Only
MPUIDR	MPU Type Register	Read Only
REVIDR	Revision ID Register	Read Only
ID_PFR0	Processor Feature Register 0	Read Only
ID_PFR1	Processor Feature Register 1	Read Only
ID_DFR0	Debug Feature Register	Read Only

Table 1-9 ID registers (continued)

Name	Description	Type
ID_AFR0	Auxiliary Feature Register 0	Read Only
ID_MMFR[0-3]	Memory Model Feature Registers	Read Only
ID_ISAR[0-4]	Instruction Set Attributes Registers	Read Only
CCSIDR	Cache Size ID Register	Read Only
AIDR	Auxiliary ID Register	Read Only

1.5.1.7 MPU registers

Table 1-10 describes the Memory Protection Unit Registers.

Table 1-10 MPU registers

Name	Description	Type
R[0-23]_DRBAR	MPU Region Base Address Registers	Read/Write
R[0-23]_DRSR	MPU Region Size and Enable Registers	Read/Write
R[0-23]_DRACR	MPU Region Access Control Registers	Read/Write
RGNR	MPU Memory Region Number Registers	Read/Write
DRBAR	MPU Region Base Address Registers	Read/Write
DRSR	MPU Region Size and Enable Registers	Read/Write
DRACR	MPU Region Access Control Registers	Read/Write

1.5.1.8 Perf registers

The Perf group (Table 1-11) contains performance-related registers.

Table 1-11 Performance registers

Name	Description	Type
PMCR	Performance Monitor Control Register	Read/Write
PMCNTENSET	Count Enable Set Register	Read/Write
PMCNTENCLR	Count Enable Clear Register	Read/Write
PMOVSR	Overflow Flag Status Register	Read/Write
PMSELR	Event Counter Selection Register	Read/Write
PMCCNTR	Cycle Count Register	Read/Write
PMXEVTYPER	Event Selection Register	Read/Write
PMXEVCNTR	Performance Monitor Count Registers	Read/Write
PMUSERENR	User Enable Register	Read/Write
PMINTENSET	Interrupt Enable Set Register	Read/Write
PMINTENCLR	Interrupt Enable Clear Register	Read/Write

1.5.1.9 SCU registers

Table 1-12 describes the SCU Registers. *m* indicates processor (0, 1, 2, 3).

Table 1-12 SCU registers

Name	Description	Type
SCU_CTRL	SCU Control Register	Read/Write
SCU_CONFIG	SCU Configuration Register	Read Only
SCU_CPU_POW_ST	SCU CPU Power Status Register	Read Only
SCU_FILT_START	Master Filtering Start Address Register	Read/Write[1]
SCU_FILT_END	Master Filtering End Address Register	Read/Write[1]
SCU_PERIPH_FILT_ START	Peripherals Filtering Start Address Register	Read Only
SCU_PERIPH_FILT_END	Peripherals Filtering End Address Register	Read Only
SCU_FPP_FILT_START*m*	FPP Filtering Start	
SCU_FPP_FILT_END*m*	FPP Filtering End	
SCU_INV_ALL	SCU Invalidate All Registers	Read/Write
SCU_SAC	SCU Access Control Register	Read/Write

1. Read/Write for the two AXI variants; Read Only for other protocols.

1.5.1.10 Timer_WD registers

Table 1-13 describes the Timer_WD registers group.

Table 1-13 Timer_WD registers

Name	Description	Type
TWD_T_LOAD	Private Timer Load Register	Read/Write
TWD_T_CNT	Private Timer Counter Register	Read/Write
TWD_T_CTRL	Private Timer Control Register	Read/Write
TWD_T_IS	Private Timer Interrupt Status Register	Read/Write
TWD_WD_LOAD	Watchdog Load Register	Read/Write
TWD_WD_CNT	Watchdog Counter Register	Read/Write
TWD_WD_CTRL	Watchdog Control Register	Read/Write
TWD_WD_IS	Watchdog Interrupt Status Register	Read/Write
TWD_WD_RST	Watchdog Reset Status Register	Read/Write

1.5.1.11 VFP registers

The VFP group (Table 1-14) contains the floating point-related registers. These registers are present only if the Cortex-R8 Cycle Model was configured with an FPU coprocessor.

Table 1-14 VFP registers

Name	Description	Type
FPSID	Floating-point System ID Register	Read Only
MVFR0	Media and VFP Feature Register0	Read Only
MVFR1	Media and VFP Feature Register1	Read Only
FPEXC	Floating-point Exception Register	Read/Write
FPSCR	Floating-point Status and Control Register	Read/Write
S0-S31	Advanced SIMD and VFP extension registers (Single word)	Read/Write
S32-S63	Additional SIMD and VFP extension registers. Only present with Neon	Read/Write
D0-D15	Advanced SIMD and VFP extension registers (Double word)	Read/Write
D16-D31	Additional SIMD and VFP extension registers (Double word). Only present with Neon	Read/Write

1.5.2 Run To Debug Point feature

The “run to debug point” feature has been added to enhance Cycle Model debugging. This feature forces the processor into a coherent state called a debug point. The Cycle Model is brought to the debug point automatically whenever a software breakpoint is hit (including single stepping). However, if a hardware breakpoint is reached, or the system is advanced by cycles within SoC Designer, the Cycle Model can get to a non-debuggable state. In this event, the *run to debug point* will advance the processor to the debug state. It does this by stalling the instruction within the decode stage and allowing all earlier instructions to complete. Once that has been accomplished, the Cycle Model will cause the system to stop simulating.

The run to debug point is available as a context menu item for the component within SoC Designer Simulator. It is also available in the disassembler view.

1.5.3 Memory information

Table 1-15 describes the available memory space views. Note that address range and access size are configuration-dependent.

Table 1-15 Memory Spaces

Name	Address Range	Access Size
memory[1]	configuration-dependent	8
axi_m0	configuration-dependent	8
axi_m1	configuration-dependent	8

1. Includes caches & TCMs.

1.5.4 Disassembly view

To display the disassembly view in the SoC Designer Simulator, right-click on the Cortex-R8 Cycle Model and select **View Disassembly…** from the context menu.

All CADI windows support breakpoints – when double-clicking on the proper location a red dot will indicate that a breakpoint is currently active. To remove the breakpoints simply double-click on the same location again.

1.6 Available Profiling data

Profiling data is enabled, and can be viewed using the Profiling Manager, which is accessible via the Debug menu in the SoC Designer Simulator. Both hardware- and software-based profiling are available.

1.6.1 Hardware profiling

Table 1-16 shows the profiling streams and events supported by the Cortex-R8 Cycle Model.

Table 1-16 Cortex-R8 profiling events

Stream	Event Name	Comments
Instructions	0x00_SW_INCREMENT	Software Increment
	0x08_INST_ARCHITECTURALLY_EXECUTED	Instructions architecturally executed
	0x09_EXC_TAKEN	Exception taken
	0x0A_EXC_RETURN	Exception return architecturally executed
	0x0B_CID_WRITE	Counts the number of instructions architecturally executed writing into the ContextID Register
	0x06_DATA_READ	Data read architecturally executed
	0x07_DATA_WRITE	Data write architecturally executed
Pipeline	0x0C_SW_PC_CHANGE	Software change of the PC
	0x0D_BR_IMMED	Immediate branch architecturally executed
	0x0E_BR_RETURN	Procedure return (other than exception returns) architecturally executed.
	0x0F_UNALIGNED	Unaligned load-store
	0x10_BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed
	0x12_BR_PRED	Predictable branch speculatively executed
	0x80_STREX_PASS	Exclusive instruction speculatively executed - STREX pass
	0x81_STREX_FAIL	Exclusive instruction speculatively executed - STREX fail
I-Cache	0x01_I_CACHE_MISS	Instruction cache miss
	0x14_I_CACHE_ACCESS	Level 1 instruction cache access
D-Cache	0x03_D_CACHE_MISS	Data cache miss
	0x04_D_CACHE_ACCESS	Data read or write operation that causes a cache access at (at least) the lowest level of data or unified cache

Table 1-16 Cortex-R8 profiling events (continued)

Stream	Event Name	Comments
Cycle	0x11_CPU_CYCLES	Cycle
	0x50_NUM_CYCLES_IRQs _INTERRUPTED	Number of cycles IRQs are interrupted
	0x51_NUM_CYCLES_FIQs _INTERRUPTED	Number of cycles FIQs are interrupted
Microarchitecture	0x90_DMB_STALL	DMB stall
	0x91_ITCM_ACCESS	ITCM access
	0x92_DTCM_ACCESS	DTCM access
	0x93_DATA_EVICTION	Data eviction
	0x94_SCU_COHERENCY	SCU coherency operation (CCB request)
	0x95_ICACHE_STALL	Instruction cache dependent stall
	0x96_DCACHE_STALL	Data cache dependent stall
	0x97_NOCACHE_NOPERIPH _STALL	Non-cacheable no peripheral dependent stall
	0x98_NOCACHE_PERIPH_STALL	Non-cacheable peripheral dependent stall
	0x99_DCACHE_HI_PRIO_STALL	Data cache high priority dependent stall
	0x9_FPP_ACCESS	Access to AXI fast peripheral port (reads and writes)

1.6.2 Software profiling

Software-based profiling is provided by SoC Designer. Profiling information is also available in the SoC Designer Profiler. See the *SoC Designer User Guide* (100996) for more information.

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.