Cortex®-M23 Processor Cycle Model

Version 9.6.0

User Guide

Non-Confidential

arm

101162_0906_00 (ID103017)

Cortex-M23 Processor Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
November 2017	0906-00	Non-Confidential	Release with 9.6

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Confidentiality Status

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

Product Status

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

Web Address

http://www.arm.com

Contents

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™ Plus.

About This Guide

This guide provides all the information needed to configure and use the Cortex™-M23 Cycle Model in SoC Designer Plus.

Audience

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

- SoC Designer Plus
- 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 Plus:

- *SoC Designer Plus User Guide* (100996)

The following publications provide reference information about Arm products:

- *Cortex-M23 Processor Technical Reference Manual* (DDI 0550)
- *Cortex-M23 Processor Integration and Implementation Manual* (DIT 0059)
- *AMBA Specification (Rev 2.0)* (IHI0011)

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.

Glossary

AMBA	*Advanced Microcontroller Bus Architecture*. The Arm open standard on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a System-on-Chip (SoC).
AHB	*Advanced High-performance Bus*. A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol.
APB	*Advanced Peripheral Bus*. A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports.
AXI	*Advanced eXtensible Interface*. A bus protocol that is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.
Cycle Model	A software object created by the Cycle Model Studio (or *Carbon compiler*) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.
Carbon Model Studio	Graphical tool for generating, validating, and executing hardware-accurate software models. It creates a Cycle Model, and it also takes a Cycle Model as input and generates a component that can be used in SoC Designer Plus, Platform Architect, or Accellera SystemC for simulation.
CASI	*ESL API Simulation Interface*, is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	*ESL API Debug Interface*, enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	*ESL API Profiling Interface*, enables collecting historical data from a component and displaying the results in various formats.
Component	Building blocks used to create simulated systems. Components are connected together with unidirectional transaction-level or signal-level connections.
ESL	*Electronic System Level*. A type of design and verification methodology that models the behavior of an entire system using a high-level language such as C or C++.
HDL	*Hardware Description Language*. A language for formal description of electronic circuits, for example, Verilog.
RTL	*Register Transfer Level*. A high-level hardware description language (HDL) for defining digital circuits.
SoC Designer	High-performance, cycle accurate simulation framework which is targeted at System-on-a-Chip hardware and software debug as well as architectural exploration.
SystemC	SystemC is a single, unified design and verification language that enables verification at the system level, independent of any detailed hardware and software implementation, as well as enabling co-verification with RTL design.
Transactor	*Transaction adaptors*. You add transactors to your component to connect your component directly to transaction level interface ports for your particular platform.

Chapter 1

Using the Cycle Model Component in SoC Designer Plus

This chapter describes the functionality of the Cycle Model component, and how to use it in SoC Designer Plus. It contains the following sections:

1.1 Cortex-M23 Functionality

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. For details, see the *Cortex-M23 Processor Technical Reference Manual* (DDI 0550).

- Supported Hardware Features
- Unsupported Hardware Features

1.1.1 Supported Hardware Features

The following features of the Cortex-M23 hardware are fully implemented in the Cortex-M23 Cycle Model:

- Cortex-M23 Integer Core
- NVIC – Nested Vectored Interrupt Controller
- WIC – Wakeup Interrupt Controller Interface Support (support is for the interface only).
- AMBA5 AHB Master Interface
- Single-cycle I/O port (optional)
- FPB – Flash Patch and Breakpoint unit (optional)
- DWT – Data Watchpoint and Trace (optional)
- MPU_S – Secure Memory Protection Unit (optional)
- MPU_NS – Non-Secure Memory Protection Unit (optional)
- SAU - Secureity Attribution Unit (optional)
- ETM supported to enable software profiling.

1.1.2 Unsupported Hardware Features

The following features of the Cortex-M23 hardware are not implemented in the Cortex-M23 Cycle Model:

- SW/JTAG-DP
- Semihosting
- Register views
- Hardware profiling
- Debugger integration

1.2 Adding and Configuring the SoC Designer Plus Component

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

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

1.2.1 SoC Designer Plus Component Files

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

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

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 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 Plus *Component Window.*

1.2.3 Adding the Component to the SoC Designer Plus 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, *default.conf*, used to create the Cycle Model.

1.3 ESL Ports

This section describes the differences between the pins listed in the *Cortex-M23 Processor Technical Reference Manual* (DDI 0550) and those on the Cycle Model. Certain hardware pins have been converted to init-time Cycle Model parameters.

This section includes the following subsections:

- Available ESL Ports
- Reset Behavior and Ports
- Tied Pins

1.3.1 Available ESL Ports

Table 1-2 describes the ESL ports that are exposed in SoC Designer Plus. See the *Cortex-M23 Processor Technical Reference Manual* (DDI 0550) for more information.

Table 1-2 ESL Component Ports

ESL Port	Description	Type
AHB5Initiator_Slave_Debug	AHB5 Slave debug port.	Transaction Slave
AHB5Initiator_master	AHB5 Master port.	Transaction Master
HCLK	Clock for the majority of the non-debug logic in the processor system domain.	Clock Slave
SCLK	Free running clock that clocks a small amount of logic in the processor system domain.	Clock Slave
clk-in	This port is used internally. Leave unconnected.	Clock Slave

1.3.2 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 (see Setting Component Parameters), 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.

1.3.3 Tied Pins

The following pins are tied High:

- STCLKEN
- STCLKENNS

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 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. 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.

Table 1-3 Component Parameters

Name	Description	Allowed Values	Default Value	Init/ Runtime
AHB5Initiator_Master Align Data	AHB5 Master Transactor Data Alignment.	true, false	false	Init
AHB5Initiator_Master Big Endian	Endianness of data in AHB5 Master Transactor.	true, false	false	Init
AHB5Initiator_Master Enable Debug Messages	Enables/disables AHB5 Master port debug.	true, false	false	Runtime
AHB5Initiator_Slave_Debug Align Data	AHB5 Slave Transactor Data Alignment.	true, false	false	Init
AHB5Initiator_Slave_Debug Big Endian	Endianness of data in AHB5 Slave Transactor.	true, false	false	Init
AHB5Initiator_Slave_Debug Enable Debug Messages	Enables/disables AHB5Initiator Slave port debug.	true, false	false	Runtime
AHB5Initiator_Slave_Debug Filter HREADYIN	AHB5 Slave Transactor HREADY Signal.	0, 1	0	Init
AHB5Initiator_Slave_Debug region size [0-5]	AHB5 Slave Transactor region size.	0 - 0x100000000	0 — 0x100000000 1 - 5 — 0x0	Init
AHB5Initiator_Slave_Debug region start [0-5]	AHB5 Slave Transactor region start address	0 - 0x100000000	0x0	Init
Align Waveforms	When set to *true*, waveforms dumped from the component are aligned with the SoC Designer Plus 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 the SoC Designer Plus time.	true, false Not settable in Canvas	true	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
Carbon DB Path	Sets the directory path to the database file.	Not Used	empty	Init
CFGSECEXT	Enables support for Armv8-M Security Extension.	0, 1	0	Runtime
CFGSTCALIB	SysTick calibration; used when one or two SysTick timers are configured. When there are two, due to enabled Security Extensions, CFGSTCALIB calibrates the secure SysTick timer.	Refer to the *Cortex-M23 Processor Integration and Implementation Manual* (DIT 0059).	0x207A11F	Init
CFGSTCALIBNS	SysTick calibration; only used when Security Extensions are enabled and there are two SysTick timers. In this case, CFGSTCALIBNS calibrates the non-secure SysTick timer.	Refer to the *Cortex-M23 Processor Integration and Implementation Manual* (DIT 0059).	0x207A11F	Init
CPUWAIT	Drive HIGH to cause the processor to wait for the signal to be LOW before coming out of reset.	0, 1	0	Runtime
DBGEN	Debug authentication signal.	0, 1	0	Runtime
DBGRESTART	External restart request.	0, 1	0	Runtime
DCLK	Debug clock.	0, 1	0	Runtime
DFTCGEN	Enables the clock gating cells during scanning in or out of test patterns.	0, 1	0	Runtime
DFTRSTDISABLE	Enalbes the reset synchronizers during scan testing.	0, 1	0	Runtime
Dump Waveforms	Determines whether SoC Designer Plus dumps waveforms for this component.	true, false	false	Runtime
ECOREVNUM	Implements engineering change order modification for certain bits in the architected ID registers. See the TRM for details.	0, 1	0	Runtime
EDBGRQ	External debug request.	0, 1	0	Runtime
Enable Debug Messages	Determines whether debug messages are logged for the component.	true, false	false	Runtime
ETMPWRUP	ETM is enabled.	0, 1	1	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
HEXOKAY	Bus Exclusive Answer.	0, 1	1	Runtime
HNONSECD	Used for Secure and Non-secure transactions with peripherals. See the TRM for details.	0, 1	1	Runtime
IDAUIDA	8-bit region identifier associated with the IDAU region. See the TRM for details.	0, 1	1	Runtime
IDAUIDVA	Region number valid.	0, 1	1	Runtime
IDAUNCHKA	Region exempt fromattribution check.	0, 1	1	Runtime
IDAUNCHKB		0, 1	1	Runtime
IDAUNSA	Non-secure region response	0, 1	1	Runtime
IDAUNSB		0, 1	1	Runtime
IDAUNSCA	Non-secure-callable region response	0, 1	1	Runtime
IDAUNSCB		0, 1	1	Runtime
INITVTOR	NO DESCRIPTION AVAILABLE	0, 1	1	Runtime
INITVTORNS	NO DESCRIPTION AVAILABLE	0, 1	1	Runtime
IOMATCH	I/O address decoder response: LOW Address on IOCHECK uses AHB. HIGH Address on IOCHECK uses I/O port.	0, 1	1	Runtime
IORDATA	I/O port read data, for reads.	0, 1	1	Runtime
IRQ	External interrupt signals.	Configura-tion-depen-dent	0	Runtime
IRQLATENCY	Specifies the minimum number of cycles between an interrupt that becomes pended in the NVIC, and the vector fetch for that interrupt being issued on the AHB-Lite inter-face. See the TRM for details.	0, 1	1	Runtime
nDBGRESET	APB MCU interface reset.	0, 1	1	Runtime
NIDEN	Configures event tracing.	0, 1	1	Runtime
NMI	Non Maskable Interrupt.	0, 1	0	Runtime
RXEV	Event in.	0, 1	0	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
SLEEPHOLDREQn	Request to extend the processor sleeping state regardless of wake-up events. If the processor acknowledges this request driving SLEEPHOLDACKn LOW, this guarantees the processor remains idle even on receipt of a wake-up event.	0, 1	1	Runtime
SPIDEN	Configures secure invasive debug.	0, 1	1	Runtime
SPNIDEN	Configures secure non-invasive debug	0, 1	1	Runtime
Waveform File[1]	Name of the waveform file.	*string*	arm_cm_CortexM23.vcd	Init
Waveform Format	The format of the waveform dump file.	VCD, FSDB	VCD	Init
Waveform Timescale	Sets the timescale to be used in the waveform.	Many values in drop-down	1 ns	Init
WICENREQn	Active HIGH request for deep sleep to be WIC-based deep sleep. Driven from the power management unit.	0, 1	0	Runtime

1. When enabled, SoC Designer Plus 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.

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.