CoreLink™ MMU-400 Cycle Model

Version 9.0.0

User Guide

ARM®

ARM DUI 1009B (ID111116)

CoreLink™ MMU-400 Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
May 2016	A	Non-Confidential	Initial Restamp Release; r0p0
November 2016	B	Non-Confidential	Release 9.0.0

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 CoreLink MMU-400 System Memory Management Unit 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();`

Convention	Description	Example
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

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:

- *ARM CoreLink MMU-400 System Memory Management Unit Technical Reference Manual*
- *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.

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 *Cycle Model compiler*) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.
Cycle 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, 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	A 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

This chapter describes the functionality of the MMU-400 Cycle Model and how to use it in SoC Designer. It contains the following sections:

1.1 MMU-400 Memory Management Unit Functionality

The MMU-400 Cycle Model simulates the ARM® CoreLink™ System MMU-400 Memory Management Unit, which controls address translation, access permissions, memory attribute determination, and checking at a memory system level. The MMU-400 implements the AMBA® AXI3™, AXI4™ and ACE-Lite™ protocols. It supports a single master interface.

For details about the functionality of the hardware that the Cycle Model simulates, refer to the *ARM CoreLink MMU-400 System Memory Management Unit Technical Reference Manual*.

1.1.1 Hardware Features not Implemented

The following features of the MMU-400 hardware are not implemented in the MMU-400 Cycle Model:

- The register view is not available in SoC Designer.
- The MMU-400 Cycle Model currently has no debug support.
- Hardware and software profiling are not currently supported.

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 in Table 1-1:

Table 1-1 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.lib<*component_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

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<component_name>.conf` (for Linux)
   - `maxlib.lib<component_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. Note that the presence of certain ports and configuration options depends on the initial configuration of the Cycle Model (for example, ACE_Lite, AXI3, or AXI4).

1.3 Available Component ESL Ports

Table 1-2 describes the MMU-400 Cycle Model ESL ports that are exposed in SoC Designer.

Table 1-2 ESL Component Ports

ESL Port	Description	Direction
<*protocol*>_s[1]	Slave port for the protocol.	Transactor slave
<*protocol*>_m[1]	Master port for the protocol.	Transactor master
apb_ns	APB NonSecure port	Transactor slave
apb_s	APB Secure port	Transactor slave
<*protocol*>_ptw[2]	Page Table Walk bus master port	Transactor master
bclk	Clock that drives AXI ports "<*protocol*>_s" and "<*protocol*>_m"[1]	Input Clock
bresetn	Reset for AXI ports "<*protocol*>_s" and "<*protocol*>_m"	Signal Slave
cclk	Input clock that drives ports "*_ptw"[1], "apb_s", apb_ns"	Input Clock

Table 1-2 ESL Component Ports (continued)

ESL Port	Description	Direction
cresetn	Reset for ports "*_ptw"[1], "apb_s", apb_ns"	Input Reset
clk-in	This port is used internally. Leave unconnected.	N/A

1. Protocol name is configuration-dependent. It may be AXI3, AXI4, or ACE_Lite.
2. Protocol name may be AXI (indicates AXI3), AXI4, or ACE_Lite_DVM.

Pins not listed in this table have been tied or disconnected for performance reasons.

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 **Edit Parameters...**. You can also double-click the component. The *Edit Parameters* dialog box appears.
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 that the availability of certain parameters is dependent on your Cycle Model configuration.

Table 1-3 Component Parameters

Name	Description	Allowed Values	Default Value	Init/ 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_ns Base Address	Base address of address region 0.	Integer	0	Init
apb_ns Enable Debug Messages	Enable debug messages.	Bool	False	Runtime
apb_ns Size	Size of address region 0.	Integer	-	Init
apb_s Base Address	Base address of address region 0.	Integer	0	Init
apb_s Enable Debug Messages	Enable debug messages.	Bool	False	Runtime
apb_s Size	Size of address region 0.	Integer	-	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
CarbonDB Path	Sets the directory path to the database file.	Not used	Empty	n/a
Dump Waveforms	Whether SoC Designer dumps waveforms for this component.	True, False	False	Init
Enable Debug Messages	Determines whether debug messages are logged for the component.	True, False	False	Init
<*protocol*>_m Enable Debug Messages[1]	Enable debug messages.	True, False	False	Runtime
<*protocol*>_m Protocol Variant[1]	Protocol Variant for transactor.	String	N/A	N/A
<*protocol*>_ptw Enable Debug Messages[2]	Enable debug messages.	True, False	False	Runtime
<*protocol*>_ptw Protocol Variant[2]	Protocol Variant for transactor.	String	N/A	N/A
<*protocol*>_s Enable Debug Messages[1]	Enable debug messages.	True, False	False	Runtime
<*protocol*>_s Protocol Variant[1]	Protocol Variant for transactor.	String	N/A	N/A
<*protocol*>_s axi_size*n*[1]	Size of address region *n*, where *n* is equal to 0, 1, 2, 3, 4, or 5. **Note**: These parameters are obsolete and should be left at their default values.[3]	Integer	Default for address region 0 = 0x1000000 Default for address regions 1 to 5 = 0	Init
<*protocol*> axi_start*n*[1]	Start address of address region *n*, where *n* is equal to 0, 1, 2, 3, 4, or 5. **Note**: These parameters are obsolete and should be left at their default values.[2]	Integer	0	Init
Waveform file	Name of the waveform file.	String	arm_cm_pl470.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.	Set of values in pulldown menau.	1ns	Init

1. Protocol name is configuration-dependent. It may be AXI3, AXI4, or ACE_Lite.
2. Protocol name for _ptw parameters may be AXI (indicates AXI3), AXI4, or ACE_Lite_DVM.

3. ARM recommends using the Memory Map Editor (MME) in SoC Designer, which provides centralized viewing and management of the memory regions available to the components in a system. For information about migrating existing systems to use the MME, refer to Chapter 9 of the *SoC Designer User Guide*.

1.5 Debug Features

The MMU-400 Cycle Model currently has no debug support.

1.6 Available Profiling Data

Hardware and software profiling are not currently supported.