CoreLink™ ADB-400 AMBA® Domain Bridge Cycle Model

Version 9.1.0

User Guide

ARM®

ARM DUI1081A (ID121516)

CoreLink ADB-400 AMBA Domain Bridge Cycle Model
User Guide

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
February 2017	A	Non-Confidential	Restamp release.

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 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 Component Library Reference Manual*

The following publications provide reference information about ARM® products:

- *AMBA 3 AHB-Lite Overview*
- *AMBA Specification (Rev 2.0)*
- *AMBA AHB Transaction Level Modeling Specification*
- *Architecture Reference Manual*

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	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 in SoC Designer

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

1.1 ADB-400 Cycle Model Functionality

The AMBA® Domain Bridge (ADB) integrates multiple power or clock domains, or different voltage levels in an AMBA system. It provides an asynchronous bridge between two components or systems. It also supports clock availability management, simple reset requirements, complex power management, and Dynamic Voltage and Frequency Scaling (DVFS).

This component supports full system coherent memory views by implementing upstream data forwarding. Refer to the *SoC Designer User Guide* for details about full system coherent memory views.

You can configure the ADB-400 to provide the optimum features, performance, and gate count required for your intended application.

1.1.1 Supported Protocols

The AMBA Domain bridge supports the following protocols:

- AXI3™
- AXI4™
- ACE-Lite™
- ACE-Lite with Distributed Virtual Memory (DVM) transactions (ACE-Lite+DVM)
- ACE™

Note: The AMBA Domain Bridge does not perform protocol translation. The two components or systems on the master and slave side must use the same protocol.

1.1.2 Unsupported Protocols

- The ADB Cycle Model does not support the AXI4Stream configuration.

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

Table 1-1 SoC Designer Component Files

Platform	File	Description
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

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.

1.3 Available Component ESL Ports

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

Table 1-2 AXI ESL Component Ports

ESL Port	Description	Direction	Type
CIT_aclks	Clock slave port	Input	Clock slave
CIT_aclkm	Clock slave port	Input	Clock slave
cactives	Clock activity indicator signal	Output	Signal master
cactivem	Clock activity indicator signal	Output	Signal master
pwrdnackn	Acknowledge signal for powerdown interface	Output	Signal master
pwrdnreqn	Request signal for powerdown interface	Input	Signal slave
<*variant*>_s[1]	Slave interface port	Input	Transaction slave
<*variant*>_m[1]	Master interface port	Output	Transaction master
clk_in	Clock slave input	Input	Clock slave

1. <*variant*> may be one of the following: ACE, ACE_Lite, ACE_LITE_DVM, AXI3, or AXI4.

All pins that are not listed in this table have been either 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 dialog box, 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 that are available for the ADB-400 are described in Table 1-3.

Table 1-3 ADB-400 Component Parameters

Parameter Name	Description	Allowed Values	Default Value	Runtime[1]
Align Waveforms	When set to *true*, waveforms dumped from 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 the SoC Designer time.	true, false	true	No
Carbon DB Path	Sets the directory path to the database file.	not used	empty	No
Dump Waveforms	Whether SoC Designer dumps waveforms for this component.	true, false	false	Yes
Enable Debug Messages	When set to *true* writes the debug messages onto the SoC Designer output window.	true, false	false	Yes
<*variant*>_m Enable Debug Messages[2]	When set to true, writes debug messages to the SoC Designer output window.	true, false	false	Yes
<*variant*>_m Protocol Variant[2]	Variant of the AXI4 or AXI3 protocol to use.	Not configurable.	Variant specified during Cycle Model configuration	Yes
<*variant*>_s Enable Debug Messages[2]	When set to true, writes debug messages to the SoC Designer output window.	true, false	false	Yes
<*variant*>_s Protocol Variant[2]	Variant of the AXI4 or AXI3 protocol to use.	Not configurable.	Variant specified during Cycle Model configuration	Yes

Table 1-3 ADB-400 Component Parameters (continued)

Parameter Name	Description	Allowed Values	Default Value	Runtime[1]
<*variant*>_s axi_size[0...5][2]	These parameters are obsolete and should be left at their default values.[3]	N/A	0x10000000	No
<*variant*>_s axi_start[0...5][2]			0	No
pwrdnreqn	When set to 0, causes the component to power down.	integer	1	Yes
Waveform File [4]	Name of the waveform file.	*string*	arm_cm_ADB-400.vcd	No
Waveform Format	Format of the waveform dump file.	VCD, FSDB	VCD	No
Waveform Timescale	Sets the timescale to be used in the waveform.	Many values in drop-down	1 ns	No

1. *Yes* means the parameter can be dynamically changed during simulation, *No* means it can be changed only when building the system, *Reset* means it can be changed during simulation, but its new value will be taken into account *only* at the next reset.
2. <*variant*> may be one of the following: ACE, ACE_Lite, ACE_LITE_DVM, AXI3, or AXI4.
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*.
4. 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.

1.5 Debug Features

The ADB-400 Cycle Model has no debug features.

1.6 Available Profiling Data

The ADB-400 Cycle Model component has no profiling capabilities.