

# AXI to AXI Asynchronous Bridge Cycle Model

**Version 9.0.0**

## **User Guide**

**Non-Confidential**



# AXI to AXI Asynchronous Bridge Cycle Model

## User Guide

Copyright © 2016 ARM Limited. All rights reserved.

### Release Information

The following changes have been made to this document.

Change History			
Date	Issue	Confidentiality	Change
November 2016	A	Non-Confidential	First release

### Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM Limited (“ARM”). **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version shall prevail.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to ARM’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement specifically covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. You must follow the ARM trademark usage guidelines <http://www.arm.com/about/trademarks/guidelines/index.php>.

Copyright © ARM Limited or its affiliates. All rights reserved.  
ARM Limited. Company 02557590 registered in England.  
110 Fulbourn Road, Cambridge, England CB1 9NJ.

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

**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

## Chapter 1.

### Using the Model in SoC Designer

Asynchronous AXI to AXI Bridge Model Overview .....	1-2
Asynchronous AXI to AXI Bridge .....	1-2
Model Features .....	1-3
Component Ports .....	1-3
Component Parameters .....	1-4
Debug Features .....	1-5
Register Information .....	1-5
Available Profiling Data .....	1-8



# Preface

A Cycle Model component is a library 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 this 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
<code>courier</code>	Commands, functions, variables, routines, and code examples that are set apart from ordinary text.	<code>sparseMem_t SparseMemCreateNew();</code>
<i>italic</i>	New or unusual words or phrases appearing for the first time.	<i>Transactors</i> provide the entry and exit points for data ...
<b>bold</b>	Action that the user performs.	Click <b>Close</b> 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.	<code>\$CARBON_HOME/bin/modelstudio [ &lt;filename&gt; ]</code>
[ text1   text2 ]	The vertical bar   indicates “OR,” meaning that you can supply text1 or text 2.	<code>\$CARBON_HOME/bin/modelstudio [ &lt;name&gt;.symtab.db   &lt;name&gt;.ccfg ]</code>

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*
- *SoC Designer AHBv2 Protocol Bundle User Guide*

The following publications provide reference information about ARM® products:

- *AMBA Design Kit Technical Reference Manual*
- *AMBA Specification*
- *AMBA AHB-Lite Protocol Specification*
- *AMBA AHB Protocol Specification*
- *Architecture Reference Manual*

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

## Glossary

**Table 1:**

AMBA	<i>Advanced Microcontroller Bus Architecture</i> . 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	<i>Advanced High-performance Bus</i> . 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	<i>Advanced Peripheral Bus</i> . 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	<i>Advanced eXtensible Interface</i> . 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 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 <i>Cycle Model</i> , 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	<i>Cycle Accurate Simulation Interface</i> , is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	<i>Cycle Accurate Debug Interface</i> , enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	<i>Cycle Accurate Profiling Interface</i> , 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	<i>Electronic System Level</i> . 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	<i>Hardware Description Language</i> . A language for formal description of electronic circuits, for example, Verilog.
RTL	<i>Register Transfer Level</i> . A high-level hardware description language (HDL) for defining digital circuits.

**Table 1:**

SoC Designer	The full name is <i>SoC Designer</i> . 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	<i>Transaction adaptors</i> . You add transactors to your component to connect your component directly to transaction level interface ports for your particular platform.



# Chapter 1

## Using the Model in SoC Designer

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

- [Asynchronous AXI to AXI Bridge Model Overview](#)
- [Component Ports](#)
- [Component Parameters](#)
- [Debug Features](#)
- [Available Profiling Data](#)

## 1.1 Asynchronous AXI to AXI Bridge Model Overview

The Asynchronous AXI to AXI Bridge Cycle Model enables two AXI clock domains to communicate.

This is illustrated in Figure 1-1.

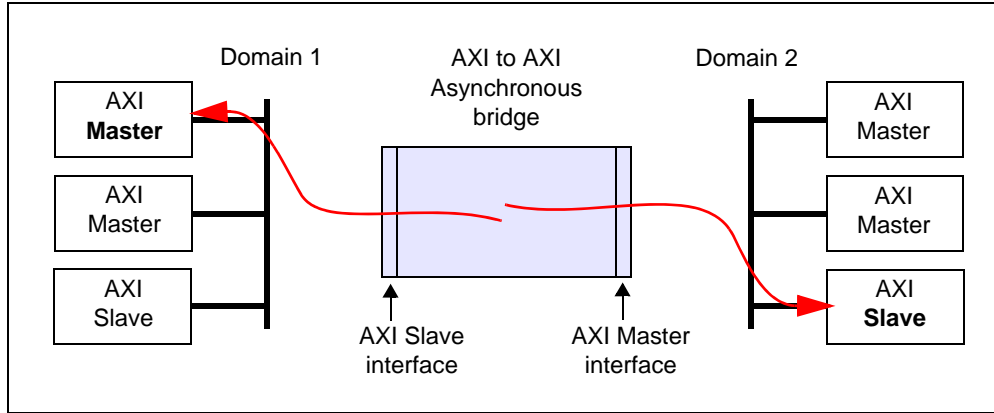


Figure 1-1 Connection of AXI Subsystems through an Asynchronous AXI to AXI Bridge

The available models are described in the next section.

### 1.1.1 Asynchronous AXI to AXI Bridge

The AXI to AXI Asynchronous Bridge is provided in three versions that support different data bus widths:

**Table 1-1 Asynchronous AXI to AXI Bridge Components**

Component	Description
CM_Bridge_Axi_Axi_ASync_32	Converts transactions for a 32-bit wide data bus.
CM_Bridge_Axi_Axi_ASync_64	Converts transactions for a 64-bit wide data bus.
CM_Bridge_Axi_Axi_ASync_128	Converts transactions for a 128-bit wide data bus.

Figure 1-2 shows a simple configuration using the Asynchronous AXI to AXI Bridge.

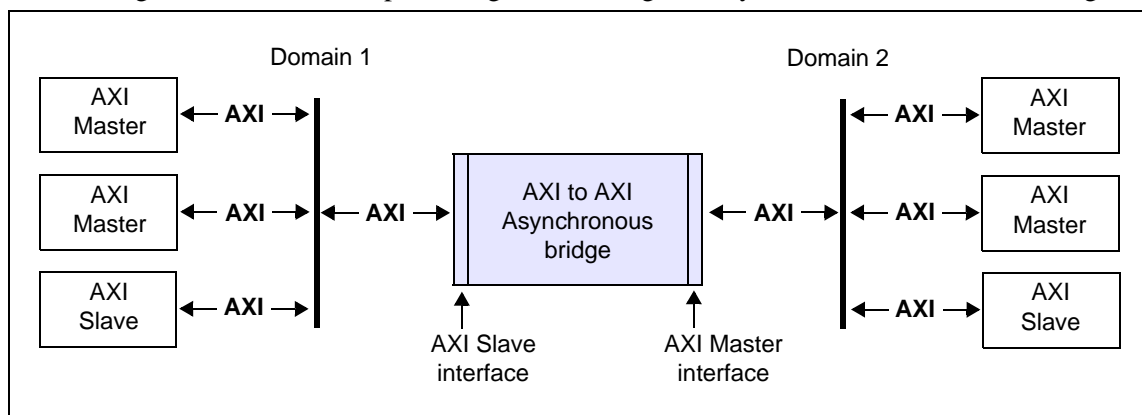


Figure 1-2 AXI to AXI Asynchronous Bridge Configuration

## 1.1.2 Model Features

The Asynchronous AXI to AXI Bridges provide the following features:

- Support for 32-, 64, and 128-bit data bus widths.
- Channel-specific variable-depth payload buffers.
- FIFO buffer depth set to 8 for each AXI channel.

## 1.2 Component Ports

Table 1-2 describes the ESL ports that are exposed in SoC Designer for the Asynchronous AXI to AXI Bridge.

**Table 1-2 ESL Component Ports**

ESL Port	Description	Direction	Type
ACLKS	Slave interface clock. This clock times all bus transfers. All signal timings on the Slave interface are related to the rising edge of ACLKS.	Input	Clock slave
ARESETSn	Reset port for resetting the Slave interface.	Input	Signal slave
axi_s	AXIv2 transaction slave port.	Input	Transaction slave
ACLKM	Master interface clock. This clock times all bus transfers. All signal timings on the Master interface are related to the rising edge of ACLKM.	Input	Clock slave
ARESETMn	Reset port for resetting the Master interface.	Input	Signal slave
axi_m	AXIv2 transaction master port.	Output	Transaction master
clk-in	Input clock. The component is clocked at the frequency of the clock connected to the <i>clk-in</i> port. If the <i>clk-in</i> port is not connected, clocking is taken from SoC Designer System Properties.  If it is connected, then it must be connected to the fastest clock. For example, if <i>ACLKM</i> is the faster domain, then <i>ACLKM</i> and <i>clk-in</i> should be connected to the same source.	Input	Clock slave

## 1.3 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. Table 1-3 describes the component parameters that are available for the Asynchronous AXI to AXI Bridge.

Parameters that may be modified at runtime are identified with *Yes* in the *Runtime* column, otherwise the parameter values are fixed and must be set before the start of simulation.

**Table 1-3 Component Parameters**

Name	Description	Allowed Values	Default Value	Runtime
Align Waveforms	When set to <i>true</i> , waveforms dumped from Cycle Model components are aligned with the SoC Designer simulation time. The reset sequence, however, is not included in the dumped data.  When <i>false</i> , the reset sequence is dumped to waveform data, however, the component time is not aligned with the SoC Designer time.	true, false	true	No
axi_m Enable Debug Messages	When set to <i>true</i> , writes AXI master debug messages to the SoC Designer output window.	true, false	false	Yes
axi_s axi_size[0-5] <sup>1</sup>	These parameters are obsolete and should be left at their default values. <sup>2</sup>	0x0 - 0xFFFFFFFF	size 0 default is 0x100000000, size 1-5 default is 0x0	No
axi_s axi_start[0-5]		0x0 - 0xFFFFFFFF	0x0	No
axi_s Enable Debug Messages	When set to <i>true</i> , writes AXI slave debug messages to the SoC Designer output window.	true, false	false	Yes
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	Enable or disable the capture of debug messages for the component.	true, false	false	Yes
Waveform File <sup>3</sup>	Name of the waveform file.	<i>string</i>	arm_CM_Bridge_Axi_Axi_ASync_<data_width>.vcd	No
Waveform Format	The 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. The square brackets specify that a range of numbers that are available. For example, the parameter name for the start addresses “axi\_s axi\_start[0-5]” will be expanded to 6 parameter name combinations that range from “axi\_s axi\_start 0” to “axi\_s axi\_start 5”. The size of a memory region depends on the “axi\_s axi\_start” and “axi\_s axi\_size” parameters. The end address is calculated as Start + Size -1. The size of the memory region must not exceed the value of 0x100000000. If the sum of Start+Size is greater than 0x100000000, the size of the memory region is reduced to the difference: 0x100000000-Start.
2. ARM recommends using the Memory Map Editor (MME) in SoC Designer, which provides centralized view-ing 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*.
3. When enabled, SoC Designer writes waveforms to the waveform file at the following times: when the wave-form buffer fills, when validation is paused and when validation finishes, and at the end of each validation run.

## 1.4 Debug Features

The Asynchronous AXI to AXI Bridge has a debug interface (CADI) that allows you to track the values of the applicable AXI signals. A view can be accessed in SoC Designer by right-clicking on the component and choosing the appropriate menu entry.

Transactions can be visualized using the transaction monitors attached to connections. By right clicking on any of the connections in SoC Designer, a transaction monitor probe can be attached.

### 1.4.1 Register Information

The available signals are listed in the following sections:

- [AXI Slave Port Signals Registers](#)
- [AXI Master Port Signals Registers](#)

#### 1.4.1.1 AXI Slave Port Signals Registers

Table 1-4 shows the AXI slave port signals. See the *ARM AMBA AXI Protocol Specification* for more information about these signals.

**Table 1-4 AXI Slave Port Signal Registers**

Name	Description	Type
ARIDS	The read address ID.	read-only
ARADDRS	The read address.	read-only
ARVALIDS	Indicates whether the read address is available.	read-only
ARREADYS	Indicates whether the slave is ready to accept the read address.	read-only
ARLENS	The burst length.	read-only
ARSIZEs	The burst size.	read-only
ARBURSTS	The burst type.	read-only
ARLOCKS	The lock type.	read-only
ARCACHES	The cache type.	read-only
ARPROTS	The protection type.	read-only
AWIDS	The write address ID.	read-only

**Table 1-4 AXI Slave Port Signal Registers (continued)**

Name	Description	Type
AWADDRS	The write address.	read-only
AWVALIDS	Indicates whether the write address is available.	read-only
AWREADYS	Indicates whether the slave is ready to accept the write address.	read-only
AWLENS	The burst length.	read-only
AWSIZES	The burst size.	read-only
AWBURSTS	The burst type.	read-only
AWLOCKS	The lock type.	read-only
AWCACHES	The cache type.	read-only
AWPROTS	The protection type.	read-only
WIDS	The write ID tag.	read-only
WDATAS	The write data.	read-only
WSTRBS	The write strobes.	read-only
WLASTS	The last transfer in a write burst.	read-only
WVALIDS	Indicates whether the write data and strobes are available.	read-only
WREADYS	Indicates whether the slave is ready to accept the write data.	read-only
RIDS	The read ID tag.	read-only
RDATAS	The read data.	read-only
RLASTS	The last transfer in a read burst.	read-only
RVALIDS	Indicates whether the read data is available.	read-only
RREADYS	Indicates whether the master is ready to accept the read data and response information.	read-only
RRESPS	The read response.	read-only
BIDS	The response ID.	read-only
BRESPS	The write response.	read-only
BVALIDS	Indicates whether the write response is available.	read-only
BREADYS	Indicates whether the master is ready to accept the response information.	read-only
AWUSERS	Additional master interface signals for the write address channel.	read-only
WUSERS	Additional master interface signals for the write data channel.	read-only
BUSERS	Additional master interface signals for the write response channel.	read-only
ARUSERS	Additional master interface signals for the read address channel.	read-only
RUSERS	Additional master interface signals for the read data channel.	read-only

### 1.4.1.2 AXI Master Port Signals Registers

Table 1-5 shows the AXI master port signals. See the *ARM AMBA AXI Protocol Specification* for more information about these signals.

**Table 1-5 AXI Master Port Signal Registers**

Name	Description	Type
ARIDM	The read address ID.	read-only
ARADDRM	The read address.	read-only
ARVALIDM	Indicates whether the read address is available.	read-only
ARREADYM	Indicates whether the slave is ready to accept the read address.	read-only
ARLENM	The burst length.	read-only
ARSIZEM	The burst size.	read-only
ARBURSTM	The burst type.	read-only
ARLOCKM	The lock type.	read-only
ARCACHEM	The cache type.	read-only
ARPROTM	The protection type.	read-only
AWIDM	The write address ID.	read-only
AWADDRM	The write address.	read-only
AWVALIDM	Indicates whether the write address is available.	read-only
AWREADYM	Indicates whether the slave is ready to accept the write address.	read-only
AWLENM	The burst length.	read-only
AWSIZEM	The burst size.	read-only
AWBURSTM	The burst type.	read-only
AWLOCKM	The lock type.	read-only
AWCACHEM	The cache type.	read-only
AWPROTM	The protection type.	read-only
WIDM	The write ID tag.	read-only
WDATAM	The write data.	read-only
WSTRBM	The write strobes.	read-only
WLASTM	The last transfer in a write burst.	read-only
WVALIDM	Indicates whether the write data and strobes are available.	read-only
WREADYM	Indicates whether the slave is ready to accept the write data.	read-only
RIDM	The read ID tag.	read-only
RDATAM	The read data.	read-only
RLASTM	The last transfer in a read burst.	read-only
RVALIDM	Indicates whether the read data is available.	read-only

**Table 1-5 AXI Master Port Signal Registers (continued)**

Name	Description	Type
RREADYM	Indicates whether the master is ready to accept the read data and response information.	read-only
RRESPM	The read response.	read-only
BIDM	The response ID.	read-only
BRESPM	The write response.	read-only
BVALIDM	Indicates whether the write response is available.	read-only
BREADYM	Indicates whether the master is ready to accept the response information.	read-only
AWUSERM	Additional master interface signals for the write address channel.	read-only
WUSERM	Additional master interface signals for the write data channel.	read-only
BUSERM	Additional master interface signals for the write response channel.	read-only
ARUSERM	Additional master interface signals for the read address channel.	read-only
RUSERM	Additional master interface signals for the read data channel.	read-only

## 1.5 Available Profiling Data

This model does not provide profiling streams, and hence, does not provide profiling information. Transaction related information can be retrieved from the respective bus components.