AXI Memory Initialization v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG341 (v1.0) May 22, 2019

XILINX®

Table of Contents

Introduction

The AXI Memory Initialization core autonomously writes an initial value to all specified address locations after power-up and following each soft reset. This prevents spurious ECC errors that can occur when accessing an uninitialized memory. This could also provide security following a partial reconfiguration to prevent a process running in a new reconfigurable module from accessing data left over from an earlier run.

Features

- User-selectable starting address and address range size.
- User-selectable initial value string to aid debugging.
- Supports AXI4 and AXI3 protocols.

IP Facts

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™, UltraScale™, 7 series FPGA, and Zynq®-7000 SoC
Supported User Interfaces	AXI4 and AXI3
Resources	N/A
Provided with Core	
Design Files	SystemVerilog
Example Design	N/A
Test Bench	N/A
Constraints File	N/A
Simulation Model	Unencrypted SystemVerilog
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI Memory Init core implements the following modes:

- Initialization
- Operational

Initialization Mode

When the AXI Memory Init core is in its initial state, it is in Initialization Mode. The AXI Memory Init IP immediately re-enters Initialization Mode following a reset.

During Initialization Mode, the slave interface (SI) of the core is held in a quiescent state for write and read. No write or read commands requested on the SI are accepted (`s_axi_awready` and `s_axi_arready` output signals are deasserted). No write responses received on the master interface (MI) are propagated to the SI. Any read data transfers observed on the MI are ignored, as AXI protocol prohibits them from occurring when no read commands have been issued.

While in the Initialization Mode, the AXI Memory Init IP autonomously writes an initial value (default all-zeros) to all specified address locations accessible on the MI, according to the values of BASE_ADDR and ADDR_SIZE. That is, the IP assumes that the memory to be initialized is of size 2**ADDR_SIZE bytes. Throughout Initialization Mode, `init_complete_out` is driven to zero.

After writing all locations and waiting for all outstanding write responses (which are accepted and discarded), the IP transitions to Operational Mode.

For all write transactions issued autonomously during Initialization Mode:

- AWLEN = `h'0F` (16-beat bursts)
- AWSIZE = $\log_2$(DATA_WIDTH / 8), that is, full-width
- AWBURST = `2'h01` (INCR)
- AWID, AWPROT, AWCACHE, AWQOS, AWUSER, AWLOCK = all zeros
- WDATA = INIT_VALUE
- WSTRB = all ones

- WUSER = all zeros

For the first write transaction, AWADDR = BASE_ADDR. For each subsequent write transaction issued, AWADDR is incremented by 16 * (DATA_WIDTH / 8).

Operational Mode

During Operational Mode, transfers on all AXI channels are unconditionally propagated across the AXI Memory Init IP unchanged and with zero latency. All propagation pathways are extremely light-weight, and should incur minimal critical path delay.

Throughout the Operational Mode, `init_complete_in` is continually registered and propagated to `init_complete_out`. This allows the `init_complete` signals among multiple AXI Memory Init instances to be daisy-chained to produce a composite completion signal.

Applications

The AXI Memory Init core is connected in proximity to the data AXI slave interface (SI) of each memory controller IP, primarily in platform memory subsystem and/or static region.

Unsupported Features

The AXI4-Lite is not supported in the core.

Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about other Xilinx® LogiCORE™ IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx® LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

This core adheres to the AXI4 and AXI3 standards.

Performance

Maximum Frequencies

For all target devices, this IP core is expected to support the maximum clock frequency of the connected memory device.

Latency

After initialization is complete, this IP core introduces no latency in the patheway between the Slave and Master AXI interfaces.

Resource Use

For all target devices, this IP core is expected to consume less than 500 LUTs.

User Parameters

The following table shows the relationship between the fields in the Vivado® IDE and the user parameters (which can be viewed in the Tcl Console).

Table 1: **User Parameters**

Parameter	Format/Range	Default Value	Description
ADDR_WIDTH	1 ≤ integer ≤ 64	32	Width of : • s_axi_araddr • m_axi_araddr • s_axi_awaddr • m_axi_awaddr
ID_WIDTH	0 ≤ integer ≤ 32	0	Width of : • s_axi_arid • m_axi_arid • s_axi_awid • m_axi_awid • s_axi_wid • m_axi_wid • s_axi_rid • m_axi_rid • s_axi_bid • m_axi_bid
DATA_WIDTH	integer = {32, 64, 128, 256, 512, 1024}	32	Width of : • s_axi_rdata • m_axi_rdata • s_axi_wdata • m_axi_wdata
ARUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of : • s_axi_aruser • m_axi_aruser
AWUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of : • s_axi_awuser • m_axi_awuser
WUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of : • s_axi_wuser • m_axi_wuser
RUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of : • s_axi_ruser • m_axi_ruser
BUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of : • s_axi_buser • m_axi_buser
PROTOCOL	string = {AXI4, AXI3}	AXI4	AXI sub-protocol of both SI and MI.
BASE_ADDR	0 ≤ bitstring ≤ 2**ADDR_WIDTH - 1; Bitstring width = ADDR_WIDTH;	0	Starting value of AWADDR during initialization. Low-order bits [ADDR_SIZE - 1:0] must be all zeros
ADDR_SIZE	$\log_2$(DATA_WIDTH / 8) + 4 ≤ integer ≤ ADDR_WIDTH	ADDR_WIDTH	Size of address range to be initialized, in power-of-2 bytes

Table 1: **User Parameters** *(cont'd)*

Parameter	Format/Range	Default Value	Description
INIT_VALUE	Bitstring width = DATA_WIDTH	0	Initialization value written to each memory location

Port Descriptions

AXI Global Signals

Table 2: **AXI Global Signals**

Signals	I/O	Width	Enablement	Description
aclk	I	1	Always	Clock for all interfaces
aclken	I	1	Optional	Clock enable for all interfaces
aresetn	I	1	Always	Active-Low reset for all interfaces (default tie-off = 1)

Slave Interface Signals

This table shows the AXI Memory Init IP Slave Interface signals.

Table 3: **Slave Interface Signals**

Signals	I/O	Default	Width	Description
s_axi_awid	I	AXI3, AXI4: 0	ID_WIDTH	Write Address Channel Transaction ID
s_axi_awaddr	I	REQ	ADDR_WIDTH	Write Address Channel Address
s_axi_awlen	I	AXI3, AXI4: 0	AXI4: 8 AXI3: 4	Write Address Channel Burst Length (0–255)
s_axi_awsize	I	AXI3, AXI4: REQ	3	Write Address Channel Transfer Size Code (0–7)
s_axi_awburst	I	AXI3, AXI4: REQ	2	Write Address Channel Burst Type Code (0–2).
s_axi_awlock	I	AXI3, AXI4: 0	AXI4: 1 AXI3: 2	Write Address Channel Atomic Access Type (0, 1)
s_axi_awcache	I	AXI3, AXI4: 0	4	Write Address Channel Cache Characteristics
s_axi_awprot	I	0b000	3	Write Address Channel Protection Bits
s_axi_awqos	I	AXI4: 0	4	AXI4 Write Address Channel Quality of Service
s_axi_awregion	I	AXI4: 0; AXI3: d/c	4	AXI4 Write Address Channel Address Region Index

Table 3: **Slave Interface Signals** *(cont'd)*

Signals	I/O	Default	Width	Description
s_axi_awuser	I	AXI3, AXI4: 0	AWUSER_WIDTH	User-defined AW Channel Signals
s_axi_awvalid	I	REQ	1	Write Address Channel Valid
s_axi_awready	O		1	Write Address Channel Ready
s_axi_wid	I	AXI3: 0 AXI4: d/c	ID_WIDTH	Write Data Channel Transaction ID for AXI3 Masters
s_axi_wdata	I	REQ	DATA_WIDTH	Write Data Channel Data
s_axi_wstrb	I	All ones	DATA_WIDTH/8	Write Data Channel Byte Strobes
s_axi_wlast	I	AXI3, AXI4: 0	1	Write Data Channel Last Data Beat
s_axi_wuser	I	AXI3, AXI4: 0	WUSER_WIDTH	User-defined W Channel Signals
s_axi_wvalid	I	REQ	1	Write Data Channel Valid
s_axi_wready	O		1	Write Data Channel Ready
s_axi_bid	O		ID_WIDTH	Write Response Channel Transaction ID
s_axi_bresp	O		2	Write Response Channel Response Code (0–3)
s_axi_buser	O		BUSER_WIDTH	User-defined B Channel Signals
s_axi_bvalid	O		1	Write Response Channel Valid
s_axi_bready	I	REQ	1	Write Response Channel Read
s_axi_arid	I	AXI3, AXI4: 0	ID_WIDTH	Read Address Channel Transaction ID
s_axi_araddr	I	REQ	ADDR_WIDTH	Read Address Channel Address
s_axi_arlen	I	AXI3, AXI4: 0	AXI4: 8 AXI3: 4	Read Address Channel Burst Length code (0–255)
s_axi_arsize	I	AXI3, AXI4: REQ	3	Read Address Channel Transfer Size Code (0–7)
s_axi_arburst	I	AXI3, AXI4: REQ	2	Read Address Channel Burst Type (0–2)
s_axi_arlock	I	AXI3, AXI4: 0	AXI4: 1 AXI3: 2	Read Address Channel Atomic Access Type (0, 1)
s_axi_arcache	I	AXI3, AXI4: 0	4	Read Address Channel Cache Characteristics
s_axi_arprot	I	0b000	3	Read Address Channel Protection Bits
s_axi_arregion	I	AXI4: 0; AXI3: d/c	4	AXI4 Read Address Channel Address Region Index
s_axi_arqos	I	AXI4: 0	4	AXI4 Read Address Channel Quality of Service
s_axi_aruser	I	AXI3, AXI4: 0	ARUSER_WIDTH	User-defined AR Channel Signals
s_axi_arvalid	I	REQ	1	Read Address Channel Valid
s_axi_arready	O		1	Read Address Channel Ready
s_axi_rid	O		ID_WIDTH	Read Data Channel Transaction ID
s_axi_rdata	O		DATA_WIDTH	Read Data Channel Data
s_axi_rresp	O		2	Read Data Channel Response Code (0–3)
s_axi_rlast	O		1	Read Data Channel Last Data Beat
s_axi_ruser	O		RUSER_WIDTH	User-defined R Channel Signals
s_axi_rvalid	O		1	Read Data Channel Valid

Table 3: **Slave Interface Signals** *(cont'd)*

Signals	I/O	Default	Width	Description
s_axi_rready	I	REQ	1	Read Data Channel Ready

Master Interface Signals

This table shows the AXI Memory Init IP Master Interface signals.

Table 4: **Master Interface Signals**

Signals	I/O	Default	Width	Description
m_axi_awid	O		ID_WIDTH	Write Address Channel Transaction ID
m_axi_awaddr	O		ADDR_WIDTH	Write Address Channel Address
m_axi_awlen	O		AXI4: 8 AXI3: 4	Write Address Channel Burst Length code. (0–255)
m_axi_awsize	O		3	Write Address Channel Transfer Size code (0–7)
m_axi_awburst	O		2	Write Address Channel Burst Type (0–2)
m_axi_awlock	O		AXI4: 1 AXI3: 2	Write Address Channel Atomic Access Type (0, 1)
m_axi_awcache	O		4	Write Address Channel Cache Characteristics
m_axi_awprot	O		3	Write Address Channel Protection Bits
m_axi_awregion	O		4	AXI4 Write Address Channel Address Region Index
m_axi_awqos	O		4	Write Address Channel Quality of Service
m_axi_awuser	O		AWUSER_WIDTH	User-defined AW Channel Signals
m_axi_awvalid	O		1	Write Address Channel Valid
m_axi_awready	I	REQ	1	Write Address Channel Ready
m_axi_wid	O		ID_WIDTH	Write Data Channel Transaction ID for AXI3 Slaves
m_axi_wdata	O		DATA_WIDTH	Write Data Channel Data
m_axi_wstrb	O		DATA_WIDTH/8	Write Data Channel Data Byte Strobes
m_axi_wlast	O		1	Write Data Channel Last Data Beat
m_axi_wuser	O		WUSER_WIDTH	User-defined W Channel Signals
m_axi_wvalid	O		1	Write Data Channel Valid
m_axi_wready	I	REQ	1	Write Data Channel Ready
m_axi_bid	I	AXI3, AXI4: REQ	ID_WIDTH	Write Response Channel Transaction ID
m_axi_bresp	I	0b00	2	Write Response Channel Response Code (0–3)
m_axi_buser	I	AXI3, AXI4: 0	BUSER_WIDTH	User-defined B Channel Signals
m_axi_bvalid	I	REQ	1	Write Response Channel Valid
m_axi_bready	O		1	Write Response Channel Ready
m_axi_arid	O		ID_WIDTH	Read Address Channel Transaction ID

Table 4: **Master Interface Signals** *(cont'd)*

Signals	I/O	Default	Width	Description
m_axi_araddr	O		ADDR_WIDTH	Read Address Channel Address
m_axi_arlen	O		AXI4: 8 AXI3: 4	Read Address Channel Burst Length Code (0–255)
m_axi_arsize	O		3	Read Address Channel Transfer Size Code (0–7)
m_axi_arburst	O		2	Read Address Channel Burst Type (0–2)
m_axi_arlock	O		AXI4: 1 AXI3: 2	Read Address Channel Atomic Access Type (0,1)
m_axi_arcache	O		4	Read Address Channel Cache Characteristics
m_axi_arprot	O		3	Read Address Channel Protection Bits
m_axi_arregion	O		4	AXI4 Read Address Channel Address Region Index
m_axi_arqos	O		4	AXI4 Read Address Channel Quality of Service
m_axi_aruser	O		ARUSER_WIDTH	User-defined AR Channel Signals
m_axi_arvalid	O		1	Read Address Channel Valid
m_axi_arready	I	REQ	1	Read Address Channel Ready
m_axi_rid	I	AXI3, AXI4: REQ	ID_WIDTH	Read Data Channel Transaction ID
m_axi_rdata	I	REQ	DATA_WIDTH	Read Data Channel Data
m_axi_rresp	I	0b00	2	Read Data Channel Response Code (0–3)
m_axi_rlast	I	AXI3, AXI4: REQ	1	Read Data Channel Last Data Beat
m_axi_ruser	I	AXI3, AXI4: 0	RUSER_WIDTH	User-defined R Channel Signals
m_axi_rvalid	I	REQ	1	Read Data Channel Valid
m_axi_rready	I		1	Read Data Channel Ready

Non-AXI Signals

Table 5: **Non-AXI Signals**

Signals	I/O	Width	Enablement	Description
init_complete_in	I	1	Always, tie-off = 1	Daisy-chain input
init_complete_out	O	1	Always	During Initialization Mode, drive to zero. During Operational Mode, register and propagate init_complete_in.

Designing with the Core

This section includes guidelines and additional information to facilitate designing with the core.

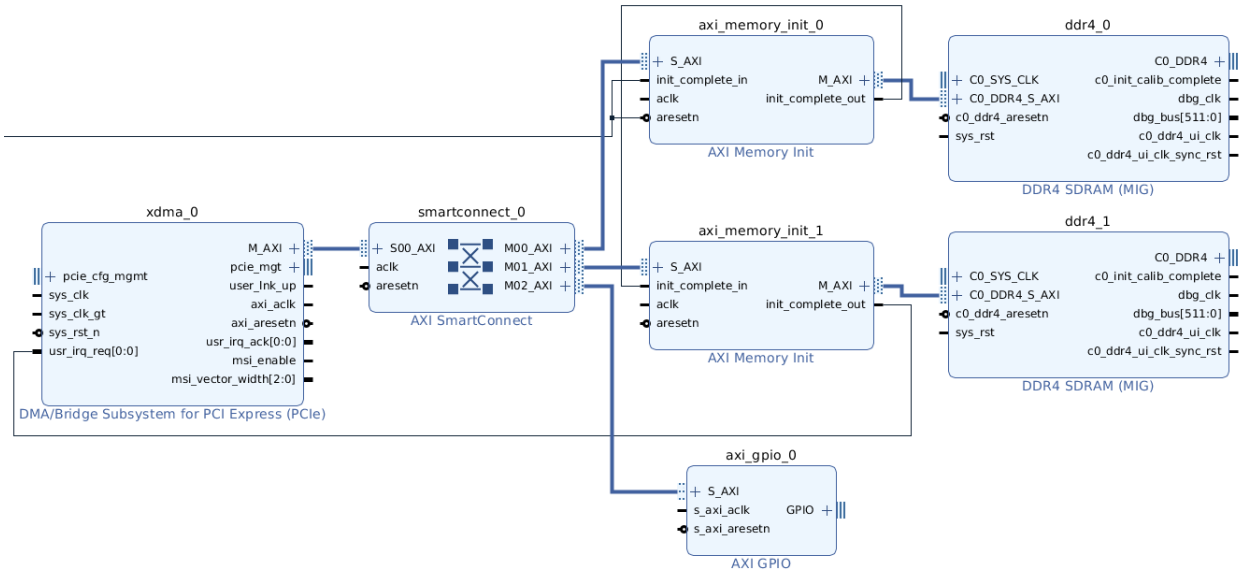
General Design Guidelines

The AXI Memory Init IP core can be inserted along any AXI pathway that accesses the memory device to be initialized. The example design shown in the following figures contains two DDR4 memory instances (4 GB each) accessed by a DMA master through a SmartConnect switch.

In the figure, an instance of the AXI Memory Init is inserted immediately before each DDR4 instance. Following reset, both AXI Memory Init cores begin initializing their respective memories in parallel.

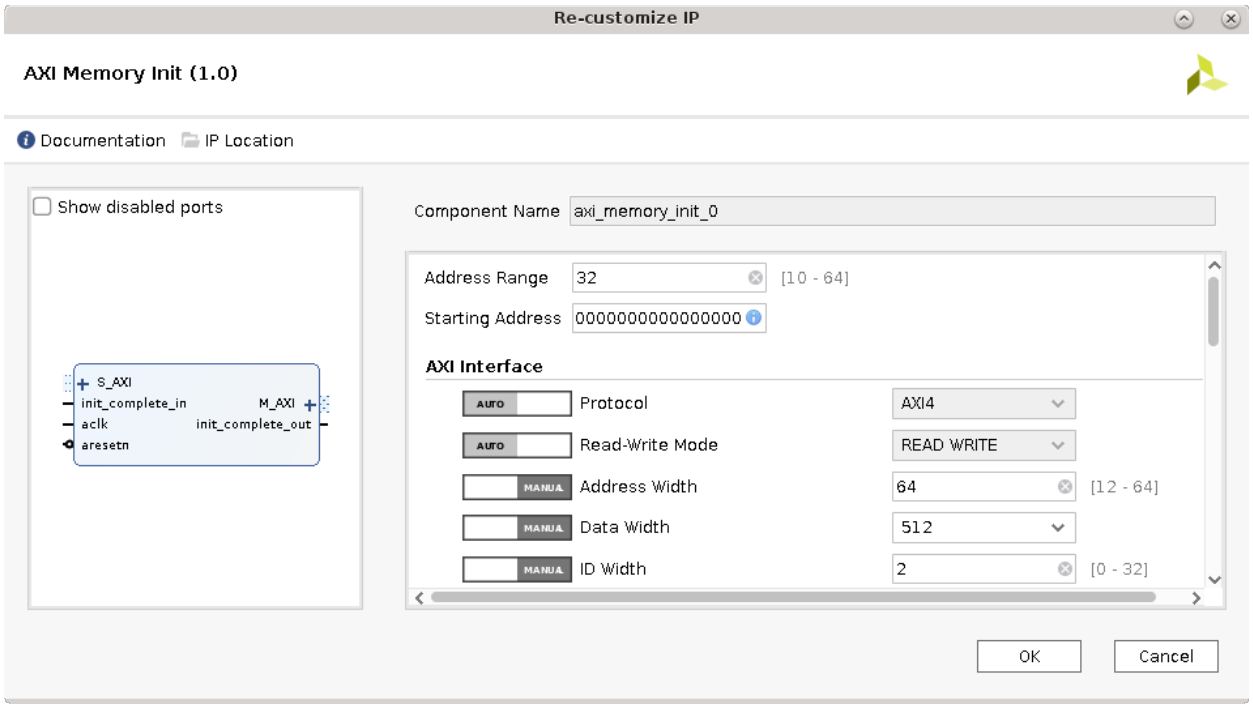
Notice how the `init_complete` signals of the two AXI Memory Init instances are daisy-chained to provide a composite completion signal to an interrupt request input. The `init_complete_in` of the first instance can be left unconnected (default tie-off High), tied High or driven by another initialization completion signal, such as a calibration completion output from a memory controller. In the figure, `init_complete_in` is driven by the same signal that drives the `aresetn` input of the IP, for convenience, as it remains High during operation.

Figure 3: **Initializing Multiple Memories in Parallel**



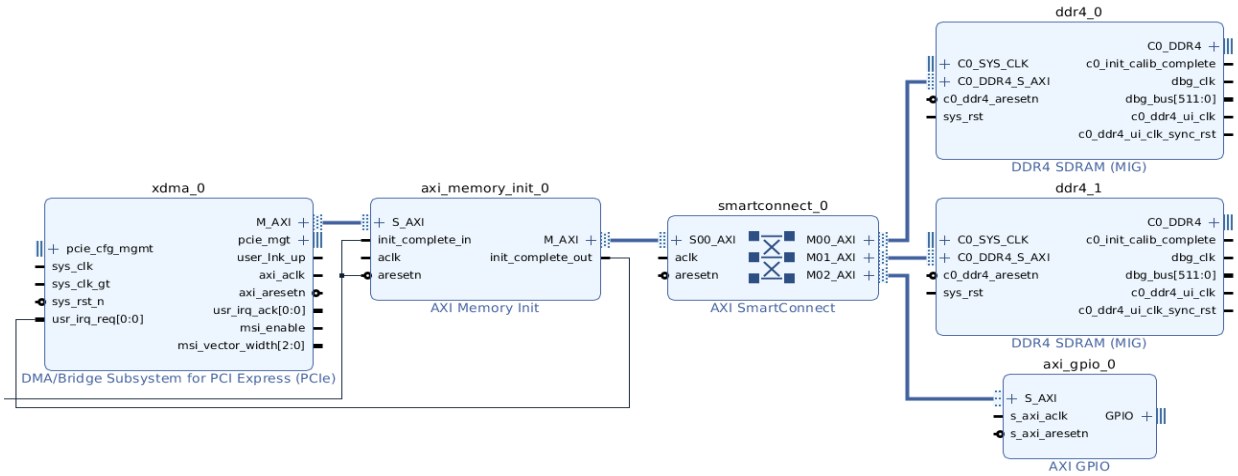
Both instances of AXI Memory Init are configured the same manner, as shown in the following figure. Notice that the base address of 0 works for both memories, because the AXI Memory Init instances are placed immediately before the memories.

Figure 4: **Configuring AXI Memory Init IP Inserted Directly Before a Memory**



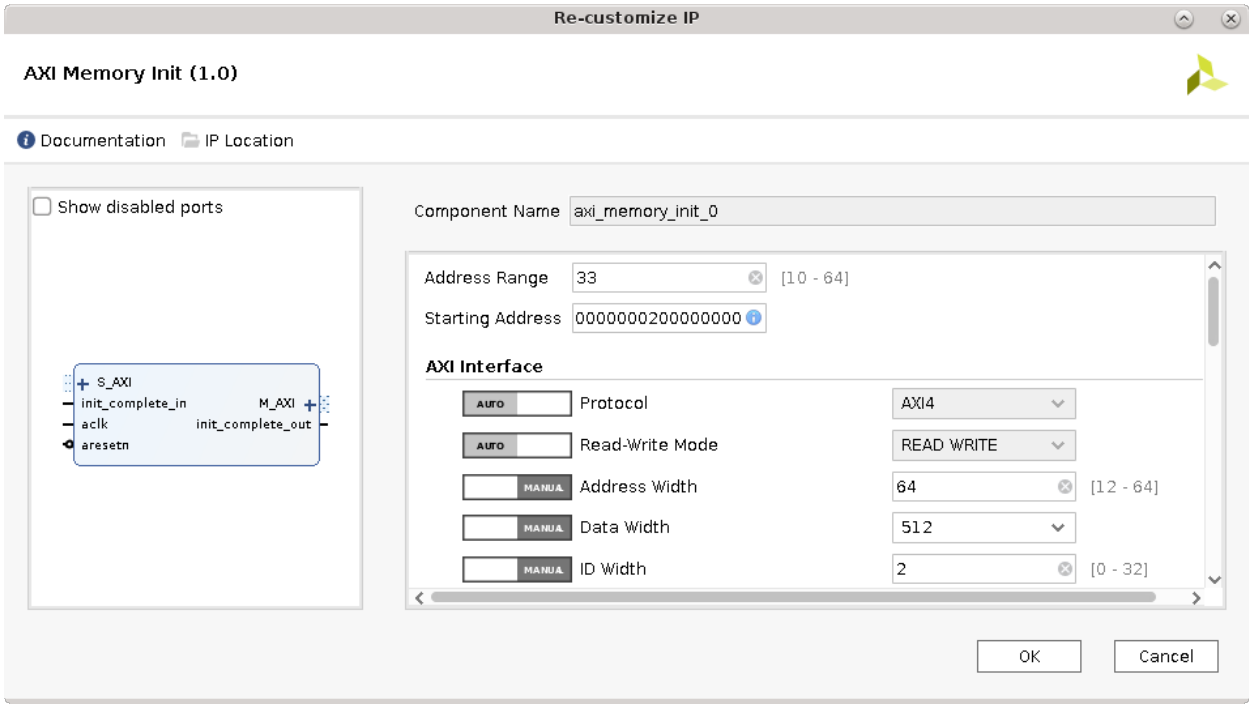
In the following figure, one instance of AXI Memory Init is inserted before the SmartConnect switch that accesses both memories, plus other slave devices. The memories are mapped to adjacent address ranges in the system address map. Following reset, the AXI Memory Init core sweeps across their combined address space, initializing both memories.

Figure 5: **Using One AXI Memory Init IP to Initialize Multiple Memories**



The figure shows the configuration of the AXI Memory Init core. Its base address must be set to the base address of the first memory. Its Address Range parameter is set to 33, indicating that it initializes a total memory space of 8 GB.

Figure 6: **Configuring an AXI Memory Init IP Inserted Before a Switch**



Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Make Only Allowed Modifications

You should not modify the core. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the core can only be made by selecting the options in the customization IP dialog box when the core is generated.

Clocking

All I/O signals on the IP are synchronized to the `aclk` input.

Resets

The AXI Memory Init IP requires one active-Low reset for all interfaces, `aresetn`. The reset is synchronous to `aclk`. AXI networks connected to the SI and MI interfaces should be reset concurrently with this IP.

Design Flow Steps

This section describes customizing and generating the core, constraining the core, and the simulation, synthesis, and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

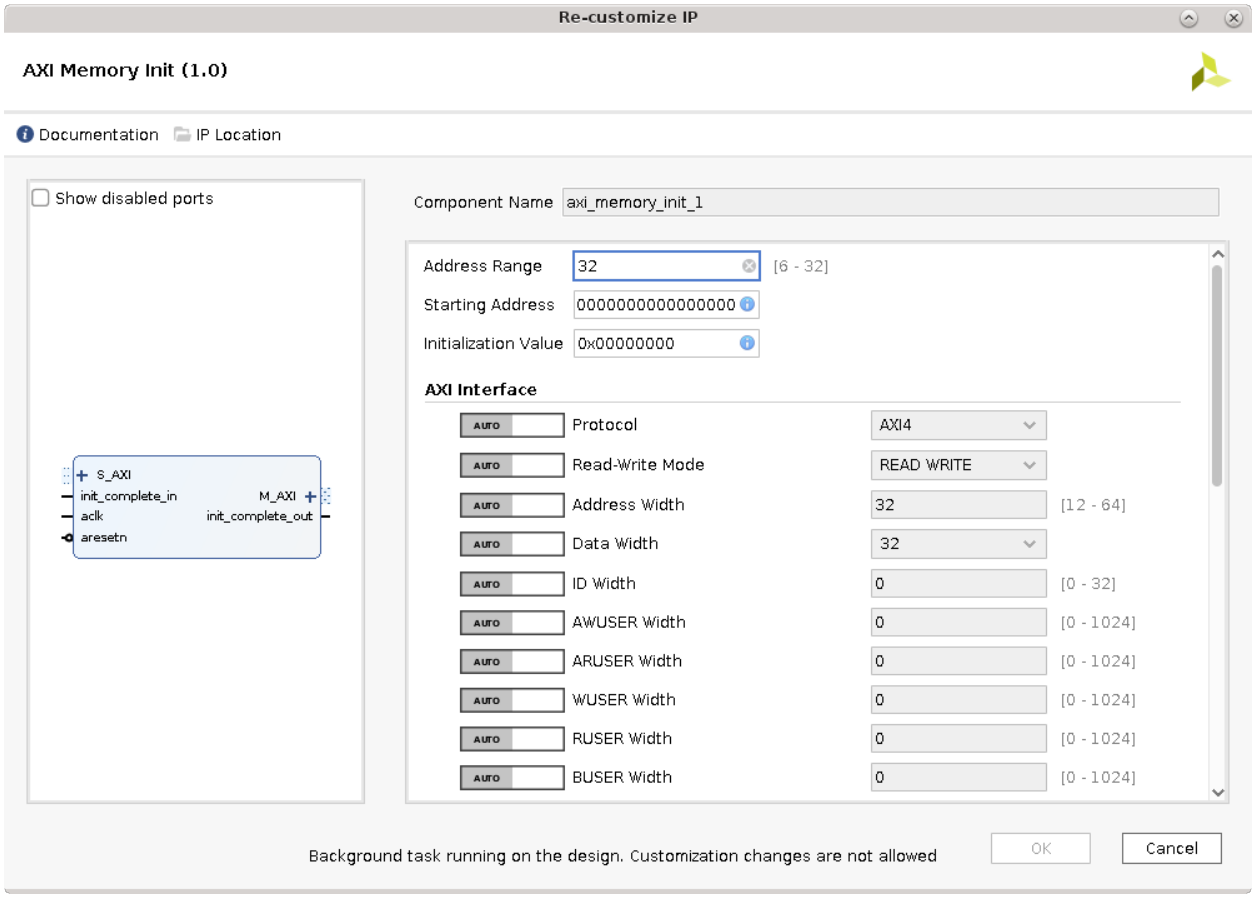
1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

AXI Memory Init Re-customize IP

Figure 7: **AXI Memory Init Re-customize IP**



- **Address Range:** Size of address range to be initialized, in power-of-2 bytes.
- **Starting Address:** Low-order bits [ADDR_SIZE - 1:0] must be all zeros.
- **Initialization Value:** Value written to all memory locations.
- **Protocol:** Selects between AXI4 and AXI3 protocols.
- **Address Width:** Width of `s_axi_araddr`, `m_axi_araddr`, `s_axi_awaddr`, and `m_axi_awaddr`. Default value is 32.
- **Data Width:** Width of `s_axi_rdata`, `m_axi_rdata`, `s_axi_wdata`, and `m_axi_wdata`. Default value is 32.
- **ID Width:** Width of `s_axi_arid`, `m_axi_arid`, `s_axi_awid`, `m_axi_awid`, `s_axi_wid`, `m_axi_wid`, `s_axi_rid`, `m_axi_rid`, `s_axi_bid`, and `m_axi_bid`. Default value is 0 (ID ports disabled).

- **AWUSER Width:** Width of `s_axi_awuser` and `m_axi_awuser`. Default value is 0 (USER ports disabled).
- **ARUSER Width:** Width of `s_axi_aruser` and `m_axi_aruser`. Default value is 0 (USER ports disabled).
- **WUSER Width:** Width of `s_axi_wuser` and `m_axi_wuser`. Default value is 0 (USER ports disabled).
- **RUSER Width:** Width of s_axi_ruser and m_axi_ruser. Default value is 0 (USER ports disabled).
- **BUSER Width:** Width of `s_axi_buser` and `m_axi_buser`. Default value is 0 (USER ports disabled).

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Constraining the Core

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900).

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Upgrading

This appendix is not applicable for the first release of the core.

Appendix B

Debugging

This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator. Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the Core

AR 72129

Technical Support

Xilinx provides technical support on the Xilinx Community Forums for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To ask questions, navigate to the Xilinx Community Forums.

XILINX

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite: AXI Reference Guide* (UG1037)
2. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
3. *Vivado Design Suite User Guide: Designing with IP* (UG896)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
05/22/2019 Version 1.0	
Initial Xilinx release.	N/A

Please Read: Important Legal Notices

www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright