JTAG to AXI Master v1.1

LogiCORE IP Product Guide

Vivado Design Suite

PG174 November 18, 2015

XILINX
ALL PROGRAMMABLE.™

Table of Contents

IP Facts

Chapter 1: Overview

Feature Summary 6
Applications 6
Unsupported Features 7
Licensing and Ordering Information 7

Chapter 2: Product Specification

Performance 8
Resource Utilization 8
Port Descriptions 9

Chapter 3: Designing with the Core

General Design Guidelines 12
Clocking 12
Resets 12
Protocol Description 12

Chapter 4: Design Flow Steps

Customizing and Generating the Core 13
Constraining the Core 16
Simulation 17
Synthesis and Implementation 17

Chapter 5: Example Design

Creating AXI Transactions 19
Issuing AXI Transactions 20
Directory and File Contents 22
<component name>_example/<component name>_example.srcs/ 22

Chapter 6: Test Bench

Appendix A: Migrating and Upgrading
Migrating to the Vivado Design Suite 24
Upgrading in the Vivado Design Suite 24

Appendix B: Debugging
Finding Help on Xilinx.com 25
Debug Tools 26
Hardware Debug 27

Appendix C: Additional Resources and Legal Notices
Xilinx Resources 28
References 28
Revision History 29
Please Read: Important Legal Notices 29

Introduction

The LogiCORE™ JTAG to AXI Master IP core is a customizable core that can generate the AXI transactions and drive the AXI signals internal to the FPGA in the system. The AXI bus interface protocol can be selected using a parameter in the IP customization Vivado® Integrated Design Environment (IDE). The width of the AXI data bus is customizable. This IP can drive AXI4-Lite or AXI4 Memory Mapped Slave through an AXI4 interconnect. Run time interaction with this core requires the use of the Vivado logic analyzer feature.

Features

- Provides AXI4 master interface
- Option to set AXI4 and AXI4-Lite interfaces
- User Selectable AXI data width – 32 and 64
- User Selectable AXI ID width up to four bits
- User Selectable AXI address width – 32 and 64
- Vivado logic analyzer Tcl Console interface to interact with hardware
- Supports AXI4 and AXI4-Lite transactions

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ Families, UltraScale™ Architecture Zynq®-7000 All Programmable SoC 7 Series
Supported User Interfaces	AXI4, AXI4-Lite
Resources	Performance and Resource Utilization web page.
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog
Test Bench	Not Provided
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows[2]	
Design Entry	Vivado® Design Suite Vivado
Simulation	Not Supported
Synthesis[3]	Vivado Synthesis
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.
3. The standard synthesis flow for Synplify is not supported for the core.

Overview

The JTAG to AXI Master is a customizable IP core which works as an AXI Master to drive AXI transactions. This IP can be used in Vivado® IP integrator or can be instantiated in HDL in a Vivado project.

Figure 1-1 shows an AXI system which uses the JTAG to AXI Master core as an AXI Master. The JTAG to AXI Master core does not have its own address space and responds to all the addresses you initiate. The JTAG to AXI Master core can communicate to all the downstream slaves (S0, S1, and S2 in this case) and can coexist with the other AXI Master in the system.

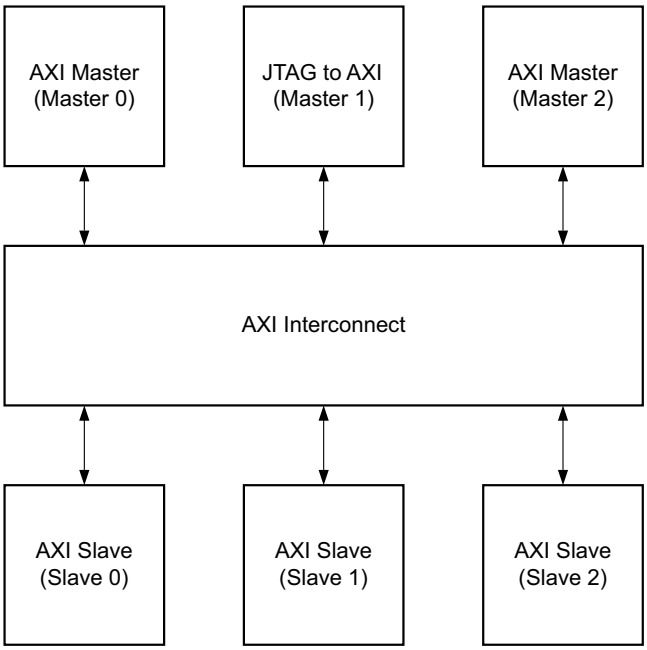

Figure 1-1: **JTAG to AXI Master System**

Feature Summary

- Parametrized protocol choice:
  - AXI4
  - AXI4-Lite
- Parametrized Address width of 32 and 64
- ID Width (up to four bits) which allows user-defined ID signals
- Fixed AXI4-Lite data width of 32
- Parametrized AXI4 data width of 32 and 64
- Supports all memory mapped AXI interface transactions including:
  - Burst Type – INCR, FIXED, and WRAP
  - Burst Length:
    - 1 to 256 for INCR and FIXED
    - 2, 4, 8, and 16 for WRAP
- Supports cache signals
- Hardware debug run time Tcl Console control to simultaneous read/write

Applications

The JTAG to AXI Master core can be used in embedded and non-embedded systems where AXI-based IP or systems need to be debugged. Also, this can be used during testing to drive the AXI transaction as test vectors on hardware and can serve as an AXI traffic generator. The ILA IP core can be used to monitor the traffic on the AXI port of the JTAG to AXI Master core.

Unsupported Features

- Narrow transfers
- Security features
- Address pipelining
- Out-of-order transaction

Licensing and Ordering Information

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

Product Specification

The JTAG to AXI Master core is used to drive data into your design through the AXI interface and also read data from your design through the same AXI interface. You write to the Vivado® Tcl Console that drives this IP through JTAG and this IP drives the AXI transaction on your AXI port. Because this does not have its own address space, it is transparent for all the AXI transactions generated from JTAG.

Along with Vivado logic analyzer this core can be used as an AXI System Debug and Testing tool.

Performance

For details about performance, visit the Performance and Resource Utilization web page.

Maximum Frequencies

The JTAG to AXI Master core is designed to run at design clock frequencies up to 200 MHz, but maximum clock frequency may be limited by other factors in the design such as overall utilization or routing congestion.

Resource Utilization

For details about resource utilization, visit the Performance and Resource Utilization web page.

Port Descriptions

Table 2-1 shows the JTAG to AXI Master core I/O port signals.

Table 2-1: **JTAG to AXI Master I/O Signal Description**

Signal Name	Interface	I/O	Initial State	Description
m_axi_awaddr (C_M_AXI_ADDR_WIDTH – 1: 0)	M_AXI4/ M_AXI4_LITE	O	0s	Write Address Channel Address Bus
m_axi_awlen[7:0]	M_AXI4	O	0s	Write Address Channel Burst Length. In data beats – 1.
m_axi_awsize[2: 0]	M_AXI4	O	0s	Write Address Channel Burst Size. Indicates with of burst transfer. 000b = 1-byte (8-bit wide burst) 001b = 2 bytes (16-bit wide burst) 010b = 4 bytes (32-bit wide burst) 011b = 8 bytes (64-bit wide burst). 100b = 16 bytes (128-bit wide burst) 101b = 32 bytes (256-bit wide burst) 110b = 64 bytes (512-bit wide burst) 111b = 128 bytes (1,024-bit wide burst)
m_axi_awburst[1:0]	M_AXI4	O	0s	Write Address Channel Burst Type. Indicates type burst. 00b = FIXED – Fixed address 01b = INCR – Incrementing address 10b = WRAP – Not supported 11b = reserved
m_axi_awprot[2:0]	M_AXI4	O	010b	Write Address Channel Protection. This is always driven with a constant output of 0010b.
m_axi_awcache[3:0]	M_AXI4	O	0011b	Write Address Channel Cache
m_axi_awvalid	M_AXI4/ M_AXI4_LITE	O	0	Write Address Channel Write Address Valid. Indicates if m_axi_awaddr is valid. 0 = write address is not valid 1 = write address is valid
m_axi_awready	M_AXI4/ M_AXI4_LITE	I	–	Write Address Channel Write Address Ready. Indicates target is ready to accept the write address. 0 = target not ready to accept address 1 = target ready to accept address
m_axi_wdata (C_M_AXI_DATA_WIDTH – 1:0)	M_AXI4/ M_AXI4_LITE	O	0s	Write Data Channel Write Data Bus
m_axi_wstrb (C_M_AXI_DATA_WIDTH/ 8 – 1:0)	M_AXI4/ M_AXI4_LITE	O	0s	Write Data Channel Write Strobe Bus. Indicates which bytes are valid in the write data bus. This value is passed from the stream side strobe bus.
m_axi_wlast	M_AXI4/ M_AXI4_LITE	O	0	Write Data Channel Last. Indicates the last data beat of a burst transfer. 0 = not last data beat 1 = last data beat

Table 2-1: **JTAG to AXI Master I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
m_axi_wvalid	M_AXI4/ M_AXI4_LITE	O	0	Write Data Channel Data Valid. Indicates m_axi_wdata is valid. 0 = not valid write data 1 = valid write data
m_axi_wready	M_AXI4/ M_AXI4_LITE	I	–	Write Data Channel Ready. Indicates the write channel target is ready to accept write data. 0 = target is not ready 1 = target is ready
m_axi_bresp[1:0]	M_AXI4/ M_AXI4_LITE	I	–	Write Response Channel Response. Indicates results of the write transfer. 00b = OKAY – Normal access has been successful 01b = EXOKAY – Not supported 10b = SLVERR – Slave returned error on transfer 11b = DECERR – Decode error, transfer targeted unmapped address
m_axi_bvalid	M_AXI4/ M_AXI4_LITE	I	–	Write Response Channel Response Valid. Indicates response, m_axi_bresp, is valid. 0 = response is not valid 1 = response is valid
m_axi_bready	M_AXI4/ M_AXI4_LITE	O	0	Write Response Channel Ready. Indicates write channel is ready to receive response. 0 = not ready to receive response 1 = ready to receive response\
m_axi_araddr (C_M_AXI_ADDR_WIDTH – 1:0)	M_AXI4/ M_AXI4_LITE	O	0s	Read Address Channel Address Bus
m_axi_arlen[7:0]	M_AXI4	O	0s	Read Address Channel Burst Length. In data beats – 1.
m_axi_arsize[2:0]	M_AXI4	O	0s	Read Address Channel Burst Size. Indicates with of burst transfer. 000b = 1-byte (8-bit wide burst) 001b = 2 bytes (16-bit wide burst) 010b = 4 bytes (32-bit wide burst) 011b = 8 bytes (64-bit wide burst). 100b = 16 bytes (128-bit wide burst) 101b = 32 bytes (256-bit wide burst) 110b = 64 bytes (512-bit wide burst) 111b = 128 bytes (1,024-bit wide burst)
m_axi_arburst[1:0]	M_AXI4	O	0s	Read Address Channel Burst Type. Indicates type burst. 00b = FIXED – Fixed address 01b = INCR – Incrementing address 10b = WRAP – Not supported 11b = reserved
m_axi_arprot[2:0]	M_AXI4	O	010b	Read Address Channel Protection. This is always driven with a constant output of 0010b.
m_axi_arcache[3:0]	M_AXI4	O	0011b	Read Address Channel Cache

Table 2-1: **JTAG to AXI Master I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
m_axi_arvalid	M_AXI4/ M_AXI4_LITE	O	0	Read Address Channel Read Address Valid. Indicates if m_axi_araddr is valid. 0 = write address is not valid 1 = write address is valid
m_axi_arready	M_AXI4/ M_AXI4_LITE	I	0	Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. 0 = target not ready to accept address 1 = target ready to accept address
m_axi_rdata (C_M_AXI_DATA_WIDTH – 1:0)	M_AXI4/ M_AXI4_LITE	I	–	Read Data Channel Read Data Bus
m_axi_rlast	M_AXI4/ M_AXI4_LITE	I	–	Read Data Channel Last. Indicates the last data beat of a burst transfer. 0 = not last data beat 1 = last data beat
m_axi_rvalid	M_AXI4/ M_AXI4_LITE	I	–	Read Data Channel Data Valid. Indicates m_axi_rdata is valid. 0 = not valid write data 1 = valid write data
m_axi_rready	M_AXI4/ M_AXI4_LITE	O	0	Read Data Channel Ready. Indicates the JTAG to AXI Master is ready to accept read data. 0 = target is not ready 1 = target is ready
m_axi_rresp[1:0]	M_AXI4/ M_AXI4_LITE	I	–	Read Response Channel Response. Indicates results of the write transfer. 00b = OKAY – Normal access has been successful 01b = EXOKAY – Not supported 10b = SLVERR – Slave returned error on transfer 11b = DECERR – Decode error, transfer targeted unmapped address

Designing with the Core

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

General Design Guidelines

The JTAG to AXI Master core can be used for AXI System Debug and Testing.

Clocking

The `aclk` input port is used as clock port on the AXI interface by the JTAG to AXI Master core. All of the AXI signals are generated or sampled based on the rising edge of the `aclk`. You must connect this to the proper clock source in the design.

Resets

The `aresetn` input port is used as reset port on the AXI interface by the JTAG to AXI Master core. This is an active-Low, synchronous signal and sampled with respect to `aclk`. The AXI side of logic is reset when JTAG to AXI Master core samples this as Low on the rising edge of `aclk`.

Protocol Description

For more details on the AXI specifications, see *Vivado AXI Reference Guide* (UG1037) [Ref 1].

Design Flow Steps

This chapter 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 Vivado 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) [Ref 2]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4]

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 *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 2] for detailed information. Vivado Integrated Design Environment (IDE) might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of `validate_bd_design` command.

The JTAG to AXI Master core can be found in `/Debug & Verification/Debug/` in the Vivado IP Catalog (Figure 4-1).

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) [Ref 3] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment. This layout might vary from the current version.

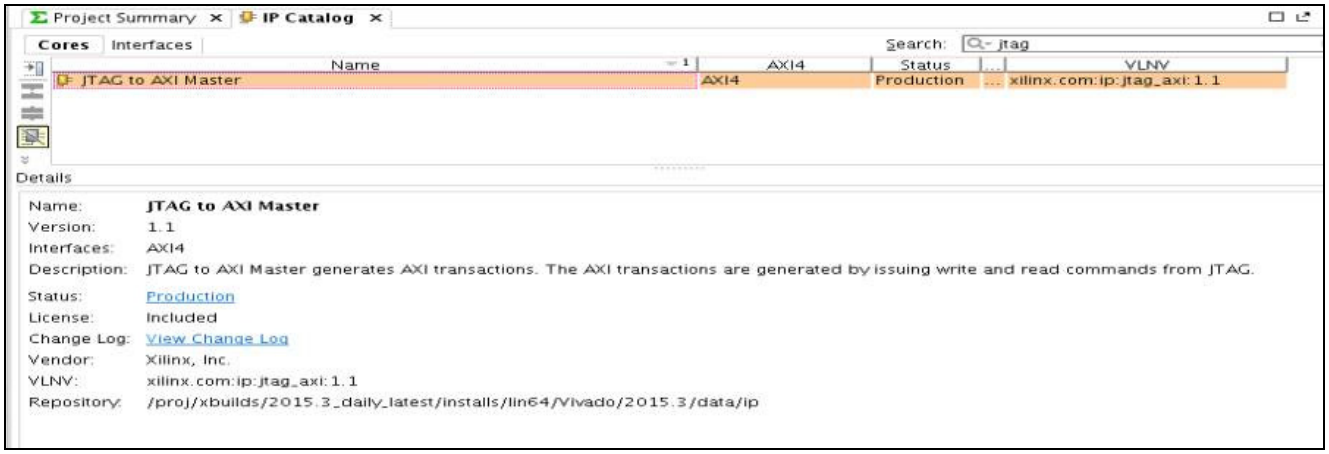

Figure 4-1: **JTAG to AXI Master Core in Vivado IP Catalog**

Vivado IDE Field

Figure 4-2 shows the Customize IP window when you click JTAG to AXI Master in the Vivado IP catalog as shown in Figure 4-1.

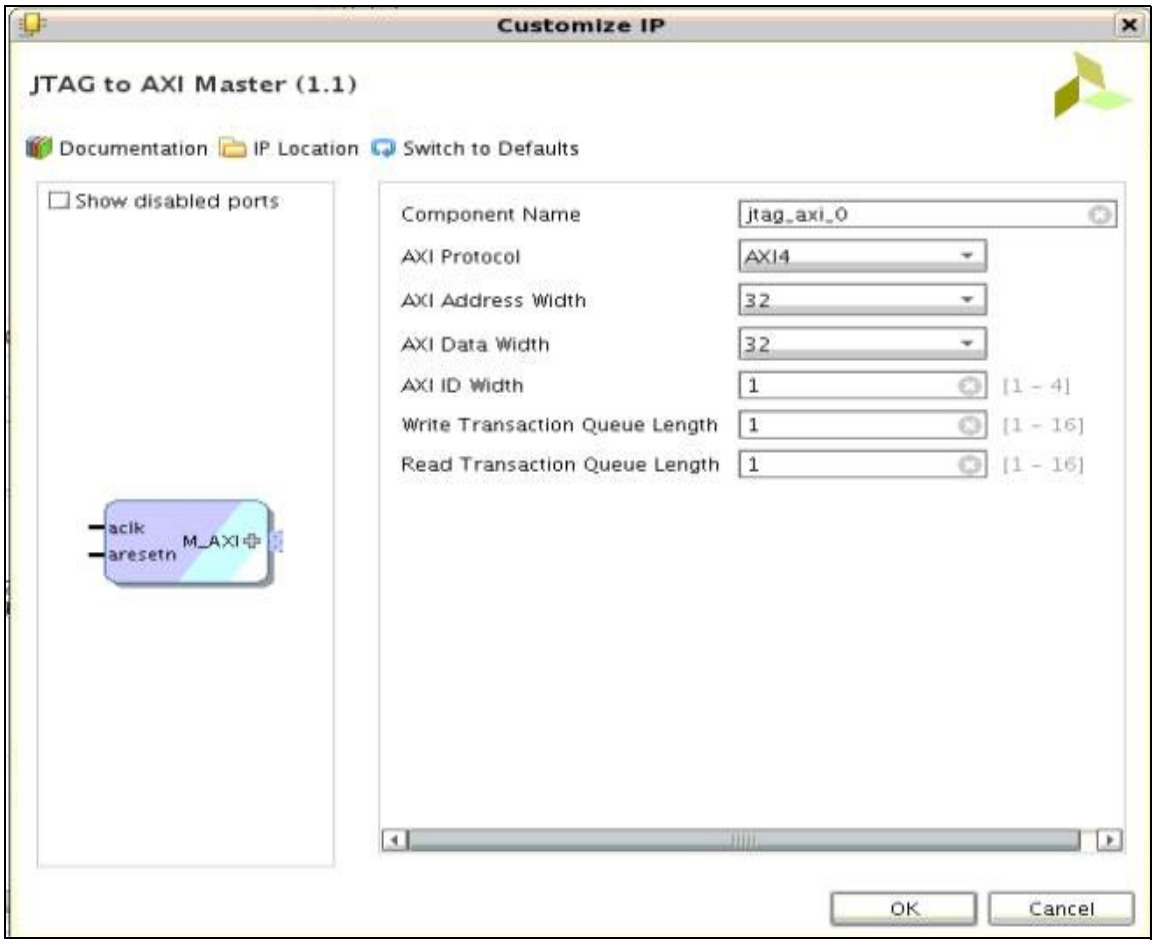

Figure 4-2: **JTAG to AXI Master Customize IP Window**

- **Component Name** – Use this text field to provide a unique module name for the ILA core.
- **AXI Protocol** – Selects the AXI4 interface protocol.
- **AXI Address Width** – Selects the AXI4 address width (32 or 64).
- **AXI Data Width** – Selects the data width (32 or 64).
- **AXI ID Width** – Selects the ID width with range of 1 to 4.
- **Write Transaction Queue Length** – Selects the maximum number of write transactions in the queue. All the queued transactions are issued back-to-back. Default is 1 with a range of 1 to 16.

- **Read Transaction Queue Length** – Selects the maximum number of read transactions in the queue. All the queued transactions are issued back-to-back. Default is 1 with a range of 1 to 16.

User Parameters

Table 4-1 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[(1)]	User Parameter/Value[(1)]	Default Value[(1)]
AXI Address Width	M_AXI_ADDR_WIDTH	32
AXI Data Width	M_AXI_DATA_WIDTH	32
AXI ID Width	M_AXI_ID_WIDTH	1
Protocol	PROTOCOL	AXI
Write Transaction Queue Length	WR_TXN_QUEUE_LENGTH	1
Read Transaction Queue Length	RD_TXN_QUEUE_LENGTH	1

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

The JTAG to AXI Master core includes an XDC file that contains appropriate multicycle path constraints to prevent the over-constraining of clock domain crossing synchronization paths. It is also expected that the clock signal connected to the `aclk` input port of the JTAG to AXI Master core is properly constrained in your design constraints.

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

This core does not support simulation.

Synthesis and Implementation

This section contains information about synthesis and implementation in the Vivado Design Suite. For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

IMPORTANT: *The standard synthesis flow for Synplify is not supported for the core.*

Interacting with the JTAG to AXI Master Core in Hardware

The JTAG to AXI Master core can only be communicated with using Tcl console commands. You can create and run AXI read and write transactions using these Tcl console commands.

A complete list of these Tcl console commands and methodology to interact with the core can be found at "Hardware System Communication Using the JTAG-to-AXI Master Debug Core" section of the chapter titled "Debugging Logic Designs in Hardware" in the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Creating AXI Transactions

Listed below are two examples for creating AXI transactions in AXI4 and AXI4-lite:

AXI4 Example

Create a write AXI burst transaction with eight 32-bit data:

```
create_hw_axi_txn wr_txn [get_hw_axis hw_axi_1] -address 00000000 -data
{11111111_22222222_33333333_44444444_55555555_66666666_77777777_88888888} -len 8
-size 32 -type write
```

Create a read AXI burst transaction with eight 32-bit data:

```
create_hw_axi_txn rd_txn [get_hw_axis hw_axi_1] -address 00000000 -len 8 -size 32
-type read
```

Create a write AXI burst transaction with eight 32-bit data and 64 bit address:

```
create_hw_axi_txn wr_txn64 [get_hw_axis hw_axi_1] -address 0000000000000000 -data
{11111111_22222222_33333333_44444444_55555555_66666666_77777777_88888888} -len
8-size 32 -type write
```

Create a read AXI burst transaction with eight 32-bit data and 64 bit address:

```
create_hw_axi_txn rd_txn64 [get_hw_axis hw_axi_1] -address 0000000000000000 -len 8
-size 32-type read
```

AXI4-Lite Example

Create a write AXI burst transaction with eight 32-bit data:

```
create_hw_axi_txn wr_txn_lite [get_hw_axis hw_axi_1] -address 00000000 -data
12345678 -type write
```

Create a read AXI burst transaction with eight 32-bit data:

```
create_hw_axi_txn rd_txn_lite [get_hw_axis hw_axi_1] -address 00000000 -type read
```

Issuing AXI Transactions

In this example, four transactions are issued back-to-back by setting the queue to four.

```
run_hw_axi txn_1 txn_2 txn_3 txn_4 -queue
```

create_hw_axi_txn

Description

Create a hardware AXI transaction object where:

```
create_hw_axi_txn  [-address <arg>] [-data <arg>] [-size <arg>] -type <arg>
                   [-len <arg>] [-burst <arg>] [-cache <arg>] [-id <arg>]
                   [-quiet] [-verbose] <name> <hw_axi>
```

Returns

New hardware AXI transaction object.

Usage

Name	Description	Default
[-address]	AXI read or write address.	Address 0
[-data]	Transaction data.	All 0s
[-size]	The number of bytes in each data transfer, or beat, in a burst.	32
-type	Read or Write transaction.	
[-len]	AXI4 Memory mapped burst lengths of 1-256 beats for incrementing bursts and 2, 4, 8, 16 beats for wrap bursts.	1
[-burst]	Burst type: INCR, FIXED or WRAP: • FIXED: The address is the same for every transfer in the burst. • INCR: The address for each transfer in the burst is an increment of the address for the previous transfer. The increment value depends on the size of the transfer. • WRAP: Similar to an INCR burst, except that the address wraps around to a lower address if an upper address limit is reached.	INCR
[-cache]	Cache type. The cache type specifies the attributes of a transaction. The transaction attributes are: • Bit 0 = Bufferable • Bit 1 = Modifiable • Bit 2 = Read Allocation • Bit 3 = Write Allocation	3
[-id]	Read or Write Transaction ID.	0

Name	Description	Default
[-quiet]	Ignore command errors.	
[-verbose]	Suspend message limits during command execution.	
<name>	Name of new object.	
<hw_axi>	Associated hardware AXI core object.	

run_hw_axi

Description

Run a hardware AXI read/write transaction(s) and update transaction status in `hw_axi` object.

Syntax

```
run_hw_axi  [-queue] [-quiet] [-verbose] <hw_axi_txns>...
```

Usage

Name	Description	Default
[-queue]	Queued mode: This allows a maximum of 16 read and 16 write transactions to be combined and executed in a local queue for lower latency and higher performance between AXI transactions. The read and write queues are independent of each other.	
[-quiet]	Ignore command errors	
[-verbose]	Suspend message limits during command execution	
<hw_axi_txns>	Hardware AXI transaction object to execute on the AXI bus	

Directory and File Contents

This section describes the files and directory structure generated for the IP example design. For the purposes of this document, assume the name of the project is the default "project_1."

project_1/<component name_example>

Top-level project directory; name is user-defined

- <component name_example.srcs>
- constrs_1/imports/<component name>/example_<component name>.xdc
- sources_1/imports/<component name>/example_<component name>.v
- sources_1/ip/axi_bram_ctrl_0
  - axi_bram_ctrl_0.xci
  - axi_bram_ctrl_0.xml
- sources_1/ip/<component name>

<component name>_example/<component name>_example.srcs/

This directory contains the source files needed to synthesize the JTAG to AXI Master core whose name is <component name>.

Table 5-1 shows the files associated with the core.

Table 5-1: **JTAG to AXI Master Example Design Source Files**

Name	Description
constrs_1/imports/<component name>/example_<component name>.xdc	Constraints file for the example design
sources_1/imports/<component name>/example_<component name>.v	Verilog (.v) source file for the example design

Implementation

To implement the example design, select **Run Implementation** in the Vivado Project Manager window. For further information on setting up the implementation, see the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 7].

Test Bench

There is no test bench for this IP core release.

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 8].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

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 JTAG to AXI Master, 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.

Documentation

This product guide is the main document associated with the JTAG to AXI Master. 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 proper 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 JTAG to AXI Master

AR: 57014

Technical Support

Xilinx provides technical support in the Xilinx Support web page 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 contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address IP core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)
- JTAG to AXI Master 1.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6]. This document gives details and an example Tcl Console command that is used to perform any AXI transaction through JTAG to AXI Master.

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado Design Suite debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado Design Suite debug feature for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

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

References

These documents provide supplemental material useful with this product guide:

1. *Vivado® 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. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
7. *Vivado Design Suite User Guide: Implementation* (UG904)
8. *ISE® to Vivado Design Suite Migration Guide* (UG911)
9. *ARM® AMBA® AXI Protocol v2.0 Specification* (ARM IHI 0022C)
10. *AMBA AXI4-Stream Protocol Specification*
11. *AXI Interconnect LogiCORE IP Product Guide* (PG059)
12. *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898)
13. *Vivado Design Suite Tutorial: Embedded Processor Hardware Design* (UG940)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/18/2015	1.1	Added support for UltraScale+ families.
09/30/2015	1.1	• Added feature: User selectable AXI address width–32 and 64. • Added User Parameter to table in *Design Flow Steps* chapter. • Updated GUI images in *Design Flow Steps* for latest version of IP Catalog.
04/01/2015	1.0	• Updated m_axi_awvalid, m_axi_arvalid, and m_axi_arready in Table 2-3: JTAG to AXI Master I/O Signal Description.
10/01/2014	1.0	• Updated to latest template. • Updated Feature Summary. • Updated Resource Utilization section. • Updated Fig. 4-1. • Updated Vivado IDE Field section. • Added User Parameter table in Design Flow Steps chapter. • Added Creating AXI Transactions and Issuing AXI Transactions sections in Example Design chapter.
12/18/2013	1.0	• Added UltraScale support. • Added Interacting with the JTAG-to-AXI Master Debug Core in Hardware section in Synthesis chapter. • Updated description to UG908 in Vivado Lab Tools section.
10/02/2013	1.0	Initial Xilinx release.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://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 http://www.xilinx.com/legal.htm#tos.

© Copyright 2013–2015 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.