Table of Contents

IP Facts

Chapter 1: Overview
   Applications ................................................................. 5
   Licensing and Ordering .................................................. 6

Chapter 2: Product Specification
   Standards ........................................................................ 7
   Performance .................................................................... 7
   Resource Utilization ........................................................ 7
   Port Descriptions ............................................................ 8
   Register Space .................................................................. 11

Chapter 3: Designing with the Core
   General Design Guidelines ................................................ 12
   Clocking ........................................................................ 13
   Resets ........................................................................... 13

Chapter 4: Customizing and Generating the Core
   Vivado Integrated Design Environment ............................. 14
   Output Generation ........................................................... 16
   Constraining the Core ...................................................... 16

Chapter 5: Simulation
   Overview ........................................................................ 17
Chapter 6: Synthesis and Implementation

Chapter 7: Example Design

Chapter 8: Test Bench

Appendix A: Upgrading
  Migrating to the Vivado Design Suite ..................................................... 22
  Upgrading in the Vivado Design Suite .................................................... 22

Appendix B: Debugging
  Finding Help on Xilinx.com ........................................................................ 23
  General Checks .......................................................................................... 24
  Debug Tools .............................................................................................. 24
  Clocks and Resets ..................................................................................... 25
  Core Size and Optimization ..................................................................... 25
  Flags ......................................................................................................... 25

Appendix C: Additional Resources
  Xilinx Resources ....................................................................................... 27
  Documentation Navigator and Design Hubs .............................................. 27
  References ................................................................................................. 27
  Revision History ....................................................................................... 28
  Please Read: Important Legal Notices ................................................... 29
Introduction

The LogiCORE IP AXI4-Stream Protocol Checker core monitors AXI4-Stream interfaces for protocol violations and provides an indication of which violation occurred.

The checks are synthesizable versions of the System Verilog protocol assertions provided by ARM in the AMBA 4 AXI4, AXI4-Lite, and AXI4-Stream Protocol Assertion User Guide [Ref 2].

Features

- Supports checking for AXI4-Stream protocol.
- Supports interface widths:
  - TDATA width: 1 to 512 bytes
  - TUSER width: 0 to 4096 bits
  - TID width: 0 to 32 bits
  - TDEST width: 0 to 32 bits
- Supports optional signals:
  - TREADY
  - TSTRB
  - TLAST
  - TKEEP
- Programmable messaging levels for simulation operation.
- Instrumented to support Vivado Debug Nets and connections to Vivado Logic Analyzer monitoring.
- AXI4-Lite control register slave interface reads protocol check status

LogiCORE IP Facts Table

Core Specifics

<table>
<thead>
<tr>
<th>Supported Device Family(1)</th>
<th>UltraScale+™ Families, UltraScale™ Architecture, Xilinx® 7 Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported User Interfaces</td>
<td>AXI4-Stream</td>
</tr>
<tr>
<td>Resources</td>
<td>See Table 2-1.</td>
</tr>
</tbody>
</table>

Provided with Core

<table>
<thead>
<tr>
<th>Design Files</th>
<th>RTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Design</td>
<td>Verilog</td>
</tr>
<tr>
<td>Test Bench</td>
<td>Verilog</td>
</tr>
<tr>
<td>Constraints File</td>
<td>Not Provided</td>
</tr>
<tr>
<td>Simulation Model</td>
<td>Verilog Source HDL</td>
</tr>
<tr>
<td>Supported S/W Driver</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Tested Design Flows(2)

<table>
<thead>
<tr>
<th>Design Entry</th>
<th>Vivado Design Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>For support simulators, see the Xilinx Design Tools: Release Notes Guide.</td>
</tr>
<tr>
<td>Synthesis</td>
<td>Vivado Synthesis</td>
</tr>
</tbody>
</table>

Support

Provided by Xilinx at the Xilinx Support web page

Notes:

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

The AXI4-Stream Protocol Checker is used to debug interface signals in systems using the AXI4-Stream protocol. When placed in an AXI4-Stream system, the connection of the AXI4-Stream Protocol Checker monitors the traffic between the AXI4-Stream Master and AXI4-Stream Slave core.

The interface is checked against the rules outlined in the AXI Specification to determine if a violation has occurred [Ref 2]. These violations are reported in a simulation log file message and as a debug net in the Vivado Logic Analyzer. In addition, the violations appear on the status vector output port from the core, as shown in Figure 1-1.

Applications

The AXI4-Stream Protocol Checker is typically used to ensure that traffic on a given AXI4-Stream connection complies with the AXI4-Stream protocol.
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 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.
Chapter 2

Product Specification

The AXI4-Stream Protocol Checker monitors the connection for AXI4-Stream protocol violations. The AXI4-Stream Protocol Checker is designed around the ARM System Verilog assertions that have been converted into synthesizable HDL. When a protocol violation occurs, the AXI4-Stream Protocol Checker asserts the corresponding bit on the \texttt{pc\_status} output vector. The output vector bit mapping can be found in Table 2-5. The value of the status vector can also be read using the optional AXI4-Lite control register slave interface.

Bits of the \texttt{pc\_status} vector are synchronously set when a protocol violation occurs. Multiple bits can be triggered on the same or different cycles. When the bit within the \texttt{pc\_status} vector has been set, it remains asserted until either the connection has been reset with \texttt{aresetn} or the core has been reset with \texttt{system\_resetn}.

---

Standards

The AXI interfaces conform to the Advanced Microcontroller Bus Architecture (AMBA®) AXI version 4 specification from Advanced RISC Machine (ARM®), including the AXI4-Lite control register interface subset [Ref 2].

---

Performance

This section details the performance information for various core configurations.

Maximum Frequencies

The maximum frequency for largest configuration listed in Table 2-1 is 225MHz.

---

Resource Utilization

The resources listed in Table 2-1 have been estimated for the Virtex-7 XC7VX485T FPGA. These values were generated using the Vivado IP catalog. They were derived from post-synthesis reports, and might change during the implementation stages.
**Note:** UltraScale device resource utilization is expected to be similar to Virtex-7 device results.

**Table 2-1: Virtex-7 Resource Utilization (XC7VX485T)**

<table>
<thead>
<tr>
<th>Range</th>
<th>HAS_TKEEP</th>
<th>HAS_TLAST</th>
<th>HAS_TREADY</th>
<th>HAS_TSTRB</th>
<th>MAXWAITS</th>
<th>TDATABYTES</th>
<th>TDEST_WIDTH</th>
<th>TID_WIDTH</th>
<th>TUSER_WIDTH</th>
<th>LUTs</th>
<th>FFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Largest</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>128</td>
<td>512</td>
<td>32</td>
<td>32</td>
<td>1024</td>
<td>0</td>
<td>14362</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>467</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>32</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>212</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>32</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>212</td>
</tr>
<tr>
<td>Smallest</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>22</td>
</tr>
</tbody>
</table>

### Port Descriptions

*Table 2-2* lists the common interface signals. *Table 2-3* lists the AXI4-Stream ports.

**IMPORTANT:** Due to the flexible nature of the AXI4-Stream protocol definition, all ports except ACLK, ARESETn and TVALID are optional ports.

**Table 2-2: Protocol Independent Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Default</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aclk</td>
<td>Input</td>
<td>Required</td>
<td>1</td>
<td>Interface clock input: Used by both the AXIS PC Monitor interface and the optional AXI4-Lite Control register slave interface.</td>
</tr>
<tr>
<td>aresetn</td>
<td>Input</td>
<td>Required</td>
<td>1</td>
<td>Interface reset input (active-Low): Resets both the AXIS PC Monitor interface and the optional AXI4-Lite Control register slave interface.</td>
</tr>
<tr>
<td>acken</td>
<td>Input</td>
<td>1</td>
<td>1</td>
<td>Interface clock enable input (active-High).</td>
</tr>
<tr>
<td>system_resetn</td>
<td>Input</td>
<td>Optional</td>
<td>1</td>
<td>System reset (active-Low). This signal can be enabled through the configuration Vivado IDE.</td>
</tr>
<tr>
<td>pc_status</td>
<td>Output</td>
<td>32</td>
<td></td>
<td>Active-High vector of protocol violations or warnings.</td>
</tr>
<tr>
<td>pc_asserted</td>
<td>Output</td>
<td>1</td>
<td></td>
<td>Active-High signal is asserted when any bit of the pc_status vector is asserted.</td>
</tr>
</tbody>
</table>

**Table 2-3: AXI4-Stream Protocol Ports**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Default</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc_axis_tlast</td>
<td>Input</td>
<td>1'b1</td>
<td>1</td>
<td>Stream Channel Last Data Beat</td>
</tr>
<tr>
<td>pc_axis_tdata</td>
<td>Input</td>
<td>DATA_WIDTH</td>
<td></td>
<td>Stream Channel Data</td>
</tr>
<tr>
<td>pc_axis_tstrb</td>
<td>Input</td>
<td>All Ones</td>
<td>DATA_WIDTH/8</td>
<td>Stream Channel Byte Strobes</td>
</tr>
</tbody>
</table>
Control Register Slave Port Descriptions

Table 2-4 lists the interface signals for the AXI4-Lite control register slave interface, when enabled. The control register slave interface is read-only.

Table 2-4: Control Register Slave Port Descriptions

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Default</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_axi_araddr</td>
<td>Input</td>
<td>Required</td>
<td>10</td>
<td>Read Address</td>
</tr>
<tr>
<td>s_axi_arvalid</td>
<td>Input</td>
<td>Required</td>
<td>1</td>
<td>Read Address Channel Valid</td>
</tr>
<tr>
<td>s_axi_arready</td>
<td>Output</td>
<td>Required</td>
<td>1</td>
<td>Read Address Channel Ready</td>
</tr>
<tr>
<td>s_axi_rdata</td>
<td>Output</td>
<td>Required</td>
<td>32</td>
<td>Read Data</td>
</tr>
<tr>
<td>s_axi_rresp</td>
<td>Output</td>
<td></td>
<td>2</td>
<td>Read Response code (always 0)</td>
</tr>
<tr>
<td>s_axi_rvalid</td>
<td>Output</td>
<td>Required</td>
<td>1</td>
<td>Read Data Channel Valid</td>
</tr>
<tr>
<td>s_axi_rready</td>
<td>Input</td>
<td>Required</td>
<td>1</td>
<td>Read Data Channel Ready</td>
</tr>
</tbody>
</table>

Protocol Checks and Descriptions

The AXI4-Stream Protocol checks and descriptions listed in Table 2-5 are the same as the assertions that are found in the ARM AXI Assertions [Ref 2] with minor differences.

Table 2-5: Checks and Descriptions

<table>
<thead>
<tr>
<th>Name of Protocol Check</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXI4STREAM_ERRM_TVALID_RESET</td>
<td>0</td>
<td>TVALID is Low for the first cycle after aresetn goes High. This assertion is not available when the system_resetn port is not enabled.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TID_STABLE</td>
<td>1</td>
<td>TID remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if both TREADY and TID are enabled on the interface.</td>
</tr>
</tbody>
</table>
Chapter 2: Product Specification

Table 2-5: Checks and Descriptions (Cont’d)

<table>
<thead>
<tr>
<th>Name of Protocol Check</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXI4STREAM_ERRM_TDEST_STABLE</td>
<td>2</td>
<td>TDEST remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if both TREADY and TDEST are enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TKEEP_STABLE</td>
<td>3</td>
<td>TKEEP remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if TDATA, TREADY and TKEEP are enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TDATA_STABLE</td>
<td>4</td>
<td>TDATA remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if both TREADY and TDATA are enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TLAST_STABLE</td>
<td>5</td>
<td>TLAST remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if both TREADY and TLAST are enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TSTRB_STABLE</td>
<td>6</td>
<td>TSTRB remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if TDATA, TREADY and TSTRB are enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TVALID_STABLE</td>
<td>7</td>
<td>When TVALID is asserted, it must remain asserted until TREADY is High. This assertion is only valid if TREADY is enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_RECS_TREADY_MAX_WAIT</td>
<td>8</td>
<td>It is recommended that TREADY is asserted within MAXWAITS cycles of TVALID being asserted. This assertion is only valid if TREADY is enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TUSER_STABLE</td>
<td>9</td>
<td>TUSER remains stable when TVALID is asserted, and TREADY is Low. This assertion is only valid if both TREADY and TUSER are enabled on the interface.</td>
</tr>
<tr>
<td>AXI4STREAM_ERRM_TKEEP_TSTRB</td>
<td>10</td>
<td>If TKEEP is de-asserted, then TSTRB must also be de-asserted. This assertion is only valid if TDATA, TSTRB and TKEEP are enabled on the interface.</td>
</tr>
<tr>
<td>XILINX_ARESETN_PULSE_WIDTH</td>
<td>11</td>
<td>ARESETn must be low for at least 16 ACLKn cycles.</td>
</tr>
<tr>
<td>XILINX_SPARSE_TKEEP</td>
<td></td>
<td>Simulation only check Detects the Sparse Tkeep in the transaction</td>
</tr>
</tbody>
</table>
Register Space

The registers listed in Table 2-6 can be read from the AXI4-Lite control register slave interface.

The Current PC registers indicate the current state of the pc_status vector, which may accumulate more checks over time.

The Snapshot PC registers indicate the state of the pc_status vector at the time of the first check assertion (when pc_asserted first becomes asserted). The Snapshot registers are reset by either aresetn or system_retn.

Table 2-6: Control Register Map

<table>
<thead>
<tr>
<th>Address Space Offset</th>
<th>Register Name</th>
<th>Access Type</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>PC asserted</td>
<td>R</td>
<td>0</td>
<td>Bit 0 = Current value of pc_asserted</td>
</tr>
<tr>
<td>0x100</td>
<td>Current PC[31:0]</td>
<td>R</td>
<td>0</td>
<td>Current value of pc_status[31:0]</td>
</tr>
<tr>
<td>0x200</td>
<td>Snapshot PC[31:0]</td>
<td>R</td>
<td>0</td>
<td>First event value of pc_status[31:0]</td>
</tr>
</tbody>
</table>
Designing with the Core

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

General Design Guidelines

The AXI4-Stream Protocol Checker should be inserted into a system as shown in Figure 3-1.

Figure 3-1: AXI4-Stream Connections
It is typically easier to find protocol violations during simulation. In simulation, it is possible to instantiate protocol checkers on all AXI4-Stream interfaces.

**TIP:** *It is highly recommended to debug a system in simulation rather than in hardware. However, when debugging in hardware, a processor can be used to monitor the pc_status port by connecting the signals to an AXI GPIO core.*

---

**Clocking**

The same aclk that is connected to both the AXI4-Stream Master and AXI4-Stream Slave should also be connected to the AXI4-Stream Protocol Checker.

---

**Resets**

At a minimum, the AXI4-Stream Protocol Checker requires the aresetn signal. A system_resetn can be configured to clear the pc_status vector without resetting the AXI4-Stream interface. Both reset inputs are synchronous to aclk. The assertion of either reset clears the pc_status vector.

**TIP:** *Xilinx recommends asserting aresetn for a minimum of 16 clock cycles.*

If system_resetn is enabled, the AXI4-Stream Protocol Checker can check the AXI4-Stream Protocol specification that defines the state of the interface following the de-assertion of aresetn. This check Protocol violation notifications related to the required behavior of the interface with respect to aresetn are cleared using system_resetn. When a system reset is not available, system_resetn should be disabled. When this is done, the AXI4STREAM_ERRM_TVALID_RESET check is not possible.
Chapter 4

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado™ Design Suite environment.

Vivado Integrated Design Environment

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 popup menu.

For details, see the sections, “Working with IP” and “Customizing IP for the Design” in the Vivado Design Suite User Guide: Designing with IP (UG896) [Ref 5] and the “Working with the Vivado IDE” section in the Vivado Design Suite User Guide: Getting Started (UG910) [Ref 6].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.
Modify the parameters to meet the requirements of the larger project into which the core is integrated. The following subsections detail the options.

**Global Parameters**

- **Component Name**: The base name of the output files generated for the core. Names must begin with a letter and can be composed of any of the following characters: a to z, 0 to 9, and ".".

  **IMPORTANT**: When using IP Integrator, the Vivado IDE automatically computes the value of the parameters in this section. Automated parameter values are typically passed from AXI4-Stream master to AXI4-Stream slave.

- **TDATA Width**: Specifies the width of the TDATA data path and its companion signals if enabled to be monitored by the protocol checker. If set to 0, the signal is omitted.

- **TID Width**: Specifies the width of the TID signal (if any) to be monitored by the protocol checker. If set to 0, the signal is omitted.
• **TDEST Width**: Specifies the width of the TDEST signal (if any) to be monitored by the protocol checker. If set to 0, the signal is omitted.

• **TUSER Width**: Specifies the width of the TUSER signal (if any) to be monitored by the protocol checker. If set to 0, the signal is omitted.

• **TREADY Enabled**: When checked, the TREADY signal is monitored by the protocol checker. If unchecked, the signal is omitted.

• **TSTRB Enabled**: When checked, the TSTRB signal is monitored by the protocol checker. If unchecked, the signal is omitted. This box is not available when the TDATA width is 0.

• **TKEEP Enabled**: When checked, the TKEEP signal is monitored by the protocol checker. If unchecked, the signal is omitted. This box is not available when the TDATA width is 0.

• **TLAST Enabled**: When checked, the TLAST signal will be monitored by the protocol checker. If unchecked the signal is omitted.

• **Enable_MARK_DEBUG**: When enabled, the signals appear as debug signals in the system debug wizard

• **Enable_Control**: Enable AXI Lite slave interface (S_AXI) to access the read only internal registers of the pc_status

**Parameter Checker Options**

• **Maximum number of idle cycles for TREADY monitoring**: Specifies the maximum number of cycles between the assertion of TVALID to the assertion of TREADY before an ERROR is generated. When the value is set to 0, this check is disabled.

• **Simulation Log Messaging Level**: When a violation is triggered this parameter allows the core to indicate to the simulation log file different error levels or to disable all messaging entirely. It is also possible via this parameter, to stop or finish the simulation upon a protocol violation occurrence.

• **Enable system reset interface**: Enables the system_reseth port. When disabled the port is tied High.

---

**Output Generation**

For details, see “Generating IP Output Products” in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

---

**Constraining the Core**

There are no required constraints for this core.
Chapter 5

Simulation

This chapter contains information about simulating in the Vivado® Design Suite environment. For details, see the “Simulating IP” section in the Vivado Design Suite User Guide: Logic Simulation (UG900) [Ref 8].

Overview

When simulating, the AXI4-Stream Protocol Checker can be configured to print display messages as follows:

\[ \text{<time>ns : <instance_path> : BIT( <bit_number>) : <log_level> : <violation_message>} \]

The fields shown are defined as:

- **instance_path**: The simulation hierarchy of the protocol checker instance that is issuing the message.
- **bit_number**: Indicates which bit in the pc_status vector is asserted. This bit can be used to look up the violation in Table 2-5.
- **log_level**: Messaging-level text that can be one of the following values: INFO, WARNING, or ERROR with the ability to stop or finish the simulation. This value is set via the configuration Vivado IDE.
- **violation_message**: Descriptive message indicating the name of the violation (for example, AXI4STREAM_ERRM_TVALID_RESET) and violation. In most cases, the location of the full violation description in the AXI specification is included. The violation message name can be used to look up the violation in Table 2-5.

For example:

\[ 73717.00ns : \text{tb.top.AXI4STREAM_0.REP : BIT(0) : ERROR : AXI4STREAM_ERRM_TVALID_RESET. TVALID must be low for the first clock edge that ARESETn goes high. Spec: section 2.7.2, and figure 2-4 on page 2-11.} \]

The core provides a debugging capability for simulation. Based on the messaging-level configuration, the AXI4-Stream Protocol Checker can either, continue, finish, or stop the simulation at the point of detecting the violation.
Chapter 6

Synthesis and Implementation

This chapter contains information about synthesis and implementation in the Vivado® Design Suite environment.

For details about synthesis and implementation, see “Synthesizing IP” and “Implementing IP” in the Vivado Design Suite User Guide: Designing with IP (UG896) [Ref 5].
Example Design

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

The example design demonstrates basic core functionality for the customized IP core. The example design is an independent Vivado Design Suite project populated with the customized IP along with additional IPs including example master(s), example slave(s), clocking and reset blocks. A synthesizable top-level HDL file is provided that instantiates and wires together the IPs, as shown in Figure 7-1. If the parent Vivado Design Suite project is configured for a Xilinx supported board, the physical board constraints are also provided. A simulation-only demonstration test bench for the example design is also provided and detailed in Chapter 8, Test Bench.

**IMPORTANT:** The example design does not exhaustively demonstrate all the features of the IP. It is not a verification test bench.

---

**Figure 7-1: Example Design Block Diagram**

The example design demonstrates basic functionality by instantiating a synthesizable AXI4-Stream master which sends transfers to an AXI4-Stream slave. After a fixed number of transfers, the master completes and asserts the done output. The slave receives the transfers and performs a meaningless operation on the payload to emulate data processing. After all transfers are received by the slave, the idle output is asserted. The master and slave will not violate AXI4-Stream protocol, and therefore no error statuses will be asserted.
When the master done output and the slave idle output are both asserted, the done output pin is asserted. If this project is configured for a Xilinx reference board, the done signal will be tied to an LED output.

The example design includes a Clocking Wizard IP core to interface with an external differential clock input and to provide a clock output to easily meet timing closure. If the project is configured for a Xilinx reference board, the Clocking Wizard core will be configured to specify the differential clock input constraints for the board.

The example design also includes a reset block IP to interface with an external reset input and to provide a de-bounced active-Low system reset. If the project is configured for a Xilinx reference board, the Proc Sys Reset IP will be configured to specify the reset input constraint for the board.
Test Bench

This chapter contains information about the provided test bench in the Vivado® Design Suite environment.

The example design generates a behavioral Verilog test bench that wraps around the top level of the example design. The test bench includes clocking and reset stimuli to the example design to run simulations. It monitors the done output to signal simulation completion. The test bench is useful for learning the signaling on the core by observing the simulation waveforms. The test bench will work with all simulation outputs from behavioral RTL through post-implementation timing.

In the example design, the simulation sources file-set includes the test bench. To run the test bench, select Run Simulation in the Vivado Flow Navigator. After the simulation is open, issue the "run all" command to run the simulation to completion. The following code shows an example successful completion of the test bench simulation.

```
1937.60ns: exdes_tb: Starting testbench
2017.60ns: exdes_tb: Asserting reset for 16 cycles
2097.60ns: exdes_tb: Reset complete
68480.00ns: exdes_tb: SIMULATION PASSED
68480.00ns: exdes_tb: Test Completed Successfully
```

For more information with running the simulation test bench, the Vivado Design Suite User Guide: Logic Simulation (UG900) [Ref 8].
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 about migrating to the Vivado Design Suite, see the ISE to Vivado Design Suite Migration Guide (UG911) [Ref 7].

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.

Parameter Changes

There were no parameter changes in this version.

Port Changes

There were no port changes in this version.
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 AXI4-Stream Protocol Checker, the Xilinx Support web page (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 AXI4-Stream Protocol Checker. 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.

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.

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.
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 also 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 AXI4-Stream Protocol Checker**

AR: 54424

---

General Checks

The AXI4-Stream Protocol Checker design limits the types of problems one may encounter when using the core. In the case where the interface is not fully specified, the system designer must ensure that any unused inputs to the AXI4-Stream Protocol Checker have been correctly tied off based on the AXI4-Stream Protocol. See Table 2-3 for the correct tie-off values.

---

Debug Tools

**Vivado Design Suite Debug Feature**

Vivado® lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

The AXI4-Stream Protocol Checker core supports probing using the Vivado ILA 2.0 core and the Vivado Logic Analyzer. All of the protocol checks named in Table 2-5 are available as
Unassigned Debug Nets in the synthesized design. See *Vivado Design Suite Tutorial: Programming and Debugging* (UG936) [Ref 4].

It is also possible to monitor all or one bit of the `pc_status` vector via any method of the designer's choosing.

---

**Clocks and Resets**

To resolve clocking and reset issues, verify these items:

- Check that `aclk` is connected to the same clock that is driving both the Master and Slave interfaces.
- Check that `aresetn` is connected to the same reset that is driving both the Master and Slave interfaces.
- Check that `aclken` is connected to the same reset that is driving both the Master and Slave interfaces.
- Ensure that both `aresetn` and `system_resetn` (if enabled) are connected to active-Low polarity.
- Ensure that `aresetn` is both synchronously asserted and released on `aclk`.

---

**Core Size and Optimization**

In some cases the size of the core can become very large. The following tips can reduce the size of the core:

- Set the maximum number of idle cycles for READY monitoring to 0. This disables a recommended *VALID to *READY wait checks.
- Each bit of the `pc_status` vector consumes some amount of resources; therefore, fewer bits observed reduces the overall footprint of the design.

---

**Flags**

- **No Flags Asserted:** One simple test to check to see if the AXI4-Stream Protocol Checker is correctly connected to the interface is to not connect the `pc_axis_tready` input into the protocol checker and to tie that port to 0. This causes multiple bits in the `pc_status` vector to be asserted when AXI4-Stream traffic begins because this will
violate all the AXI4STREAM_ERRM_T*_STABLE protocol checks (bits [1:7] and 9). See Table 2-5 for the descriptions of these checks.

- **Flags Asserted**: If there are bits asserted in the pc_status vector and the source/reason of the violation using Table 2-5 is not clear, move the AXI4-Stream Protocol Checker “upstream” toward the AXI4-Stream Master generating the transactions.
Additional Resources

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (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 the Xilinx Documentation Navigator, click the Design Hubs View tab.
- On the Xilinx website, see the Design Hubs page.

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

References

These documents provide supplemental material useful with this product guide:

1. AMBA AXI4-Stream Protocol Specification
2. AMBA 4 AXI4, AXI4-Lite, and AXI4-Stream Protocol Assertion User Guide
3. Xilinx AXI Reference Guide (UG761)
4. Vivado Design Suite Tutorial: Programming and Debugging (UG936)
7. ISE to Vivado Design Suite Migration Methodology Guide (UG911)

Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/04/2018</td>
<td>2.0</td>
<td>• Added two Global Parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>◦ Enable_MARK_DEBUG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>◦ Enable_Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated core version to 2.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Register Space section under Product Specification chapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the block diagram in the Overview chapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the Port Descriptions table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Control Register Slave Port Descriptions table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the protocol check Xilinx Sparse Tkeep.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the GUI screen capture.</td>
</tr>
<tr>
<td>10/04/2017</td>
<td>1.2</td>
<td>Added ARESETn for reset pulse width.</td>
</tr>
<tr>
<td>11/18/2015</td>
<td>1.1</td>
<td>Added support for UltraScale+ families.</td>
</tr>
<tr>
<td>12/18/2013</td>
<td>1.1</td>
<td>• Added support for UltraScale™ architecture.</td>
</tr>
<tr>
<td>10/02/2013</td>
<td>1.1</td>
<td>• Changed all signals and ports to lowercase.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Example Design and Test Bench chapters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Migration and Upgrading appendix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for IP Integrator.</td>
</tr>
<tr>
<td>03/20/2013</td>
<td>1.0</td>
<td>Initial Xilinx release.</td>
</tr>
</tbody>
</table>
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 https://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.

This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

© Copyright 2013-2018 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, 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.