AXI Protocol Firewall IP v1.1

LogiCORE IP Product Guide

Vivado Design Suite

PG293 January 21, 2021

XILINX

Table of Contents

Introduction

The Xilinx® LogiCORE™ AXI Firewall IP is a bridge between two portions of an AXI memory-mapped network that protects one portion from issues caused by the opposite portion, such as protocol violations or timeout hangs. The AXI Firewall IP monitors either the MI-side or SI-side for severe error conditions, preventing such an event from disrupting network operation on the opposite side. After the AXI Firewall IP blocks problematic traffic, the monitored portion of the AXI network can be independently reset so that normal operation can be restored without resetting the protected portion of the network.

Features

- Supports AXI3, AXI4, or AXI4-Lite interface
- Protects either the upstream or downstream network from failures caused by the opposite network:
  - Configurable to monitor either the MI side or SI side for error conditions
  - When a failure is detected, the firewall becomes blocked, preventing further transfers from propagating
  - When the firewall becomes blocked, the AXI Firewall IP generates compliant transfers out to the protected side, as needed, to terminate any outstanding transactions
  - Separate blocks for read and write transactions
- Firewall blocking trigger conditions:
  - Timeout on expected transfers
  - Fatal AXI protocol violations observed on monitored interface
  - DECERR or SLVERR response observed on the MI (optional)
  - Soft block requested through the AXI4-Lite control interface
- Firewall unblocking trigger conditions:
  - Unblock requested through the AXI4-Lite control interface
  - Global `aresetn`
- AXI4-Lite control interface:
  - Indicates read/write block status
  - Indicates cause of block
  - Indicates last observed read or write address, when applicable
  - Controls read/write unblock requests (and soft block requests)
  - Sets timeout limits

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™, UltraScale™, Zynq®-7000 SoC, 7 series FPGAs
Supported User Interfaces	AXI4, AXI4-Lite, AXI3
Resources	N/A
Provided with Core	
Design Files	SystemVerilog
Example Design	SystemVerilog
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 Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 76037
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
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 third-party tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado timing, resource and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:

Core Overview

The Xilinx LogiCORE™ AXI Firewall IP core propagates AXI traffic from its slave interface (SI) to its master interface (MI), and propagates responses back from the MI. While doing so, the core actively checks for certain potentially-fatal protocol violations in the transfers, and blocks the offending transfer and any further transfers from propagating. The core can be configured to monitor either the MI-side (responses from downstream) or the SI-side (transfers from upstream) for error conditions, allowing the AXI network connected to the opposite interface to continue operating. (MI-side and SI-side firewalls cannot both be enabled in the same IP instance.) A control-register interface on the core can then be used to read information about the error status and to initiate recovery of the firewall.

Operation

Following reset, the AXI Firewall IP wakes up in the normal operating mode, in which transfers received on either interface are propagated to the other side at full bandwidth.

MI-Side Mode Normal Operation

The AXI Firewall IP actively checks for certain potentially fatal problems in the AXI response transfers received on the R and B channels of the MI. When any of the enabled fault conditions are detected on the R or B channel of the MI, the firewall blocks further read and write transfers between the SI and MI. Separate firewall blocking conditions are imposed for read and write faults.

If the block is triggered by a protocol violation fault in a response transfer, the transfer exhibiting the violation is not propagated to the SI. The block condition is indicated through the AXI4-Lite control interface and through the non-AXI error output signals (**mi_r_error** and **mi_w_error**), both of which remain sticky until the block is cleared.

Only problematic conditions observed on the MI of the firewall trigger faults. No conditions observed on the SI of the firewall trigger faults.

SI-Side Mode Normal Operation

The AXI Firewall IP actively checks for certain potentially fatal issues in the AXI forward transfers received on the AR, AW, and W channels of the SI. When any of the enabled read or write fault conditions are detected on the associated forward channel of the SI, the AXI Firewall IP blocks further read or write transfers between the SI and MI. Separate firewall blocking conditions are imposed for read and write faults.

If the block is triggered by a protocol violation fault in a forward transfer, the transfer exhibiting the violation is not propagated to the MI. The block condition is indicated through the AXI4-Lite control interface and through the non-AXI error output signals (**si_r_error** and **si_w_error**), both of which remain sticky until the block is cleared.

Only problematic conditions observed on the SI of the firewall trigger faults. No conditions observed on the MI of the firewall trigger faults.

When configured for SI-side mode, the AXI Firewall IP does not allow W-channel transfers to propagate ahead of corresponding AW commands; any premature W-transfers are stalled (**s_axi_wready** deasserted) until the AW command is received.

Block Condition

The AXI Firewall IP is designed to detect potentially fatal conditions on the monitored interface. An AXI protocol violation is considered potentially fatal if there is a high likelihood that it would corrupt the state of the AXI interface and interfere with ongoing operation. There are additional protocol violations listed in the *AXI Protocol Specification* [Ref 1] that are not considered potentially fatal if they are likely to only corrupt the transported payload without interfering with the on-going operation of the AXI interface. The AXI Firewall IP core does not implement checks for non-fatal protocol violations, to save area.

Note: Refer to the *AXI Protocol Checker LogiCORE IP Product Guide* (PG101) [Ref 2] if you wish to monitor for all documented AXI protocol violations.

The fault(s) causing the block are readable through the control interface, and remain unchanged until the block is cleared. Specifically, during the first cycle in which any read or write fault is triggered, the MI-Side Fault Status Register/SI-Side Fault Status Register captures all checks that are asserted during that cycle, but no subsequently triggered checks (within read or write) will change the value of the read/write field of the MI-Side Fault Status Register/SI-Side Fault Status Register.

MI-Side Mode Block Condition

The AXI Firewall IP internally records all outstanding transactions, including ID thread and beat count. When the IP activates a block, all MI-side forward-channel **`valid`** outputs are deasserted and no further commands will issue on the MI.

Note: Deassertion of **`m_axi_{ar,aw,w}valid`** without corresponding **`m_axi_*ready`** is an AXI protocol violation, however the affected channels of the MI are assumed to be in an operationally invalid state when a block occurs, requiring a downstream reset before returning to normal operation.

Upon blocking, the IP also asserts the MI-side response channel **`m_axi_{r,b}ready`** outputs for the duration of the block.

Once blocked, the firewall autonomously issues protocol-compliant response transfers on the SI for all outstanding (read or write) transactions. Ordering among transaction responses per thread remains protocol-compliant, but the ordering among multiple threads is non-deterministic. Read response transfers convey the repeating **`rdata`** pattern **`0xDEADFA11`**. All response transfers indicate SLVERR response code.

Any further command transfers received on the SI during the block period are appended to the internal command queue and responded to in turn. That is, there is no requirement for upstream masters to promptly withhold subsequent commands after a block is triggered. However, no commands (or accompanying W-payload transfers) received prior to an unblock request get propagated to the MI.

The RESPONSE_BUSY bits in the MI-Side Fault Status Register indicate whether there are outstanding transactions that still need to be completed on the SI. During all modes of operation, the BUSY bits assert as soon as any activity associated with a new transaction are observed (**`arvalid=1, awvalid=1 or wvalid=1`** associated with a new transaction). The BUSY bits deassert as soon as the outstanding transaction counters decrement to zero upon receiving a completed R or B channel handshake of the last outstanding read/write transaction. If any further command transfers are received on the SI after the BUSY bit is deasserted, the BUSY bit will become asserted again until all associated activity is again completed.

The BUSY bits are not masked while reads or writes remain in normal operating mode. While read or write traffic is blocked, deassertion of the BUSY bit indicates that the autonomous

flushing operation of the firewall has completed for read or write. When read or write traffic is in normal operating mode, deassertion of the BUSY bit indicates that all outstanding transactions have completed normally. In either case, observing both the READ_RESPONSE_BUSY and WRITE_RESPONSE_BUSY low indicates that it is safe to reset all downstream (MI-side) devices and request unblocking of the firewall through the control interface (provided no new transactions arrive at the SI after BUSY deasserts).

SI-Side Mode Block Condition

The AXI Firewall IP internally records all outstanding transactions and beat counts (it is not necessary to track ID threads in SI-side mode). When the IP activates a block for a read or write fault, the SI-side read or write response-channel **`valid`** output is deasserted and no further response transfers will propagate to the SI.

Note: Deassertion of **`s_axi_{r,b}valid`** without corresponding **`s_axi_*ready`** is an AXI protocol violation. However, the affected channels of the SI are assumed to be in an operationally invalid state when a block occurs, requiring an upstream reset before returning to normal operation.

Upon blocking for a read or write fault, the IP also asserts the SI-side and MI-side **`ready`** output signals on the affected channels for the duration of the block. No further read or write commands are issued on the MI-side AR or AW channel.

Once blocked for a write fault, the firewall autonomously issues protocol-compliant W-channel transfers on the MI for any outstanding write transactions, in the order in which the prior AW commands were issued, with the repeating **`wdata`** pattern **`0xDEADFA11`**.

The RESPONSE_BUSY bits in the SI-Side Fault Status Register indicate whether there are outstanding transactions that still need to be completed on the MI. The BUSY bits deassert as soon as the outstanding transaction counters decrement to zero upon receiving a completed R or B channel handshake of the last outstanding read/write transaction.

The BUSY bits are not masked while reads or writes remain in normal operating mode. While read or write traffic is blocked, deassertion of the BUSY bit indicates that the autonomous flushing operation of the firewall has completed for read or write. When read or write traffic is in normal operating mode, deassertion of the BUSY bit indicates that all outstanding transactions have completed normally. In either case, observing both the READ_RESPONSE_BUSY and WRITE_RESPONSE_BUSY low indicates that it is safe to reset all upstream (SI-side) devices and request unblocking of the firewall through the control interface.

Recovery

A block condition can be cleared either through an AXI4-Lite control interface unblock request or global **`aresetn`**. If unblock is requested through the control interface while a BUSY bit is *deasserted* (recommended), the firewall unblocks immediately. If unblock is requested while the read/write BUSY bit is *asserted* and while the corresponding read/write traffic is blocked (not recommended), the IP waits until autonomous flushing is completed

for any remaining outstanding read/write transactions, then the block is immediately released and new command transfers begin propagating from SI to MI.

Assertion of the **bvalid** output on the control interface after writing the unblock request guarantees that it is safe to issue a new command to the SI that will propagate to the MI. An unblock request has no effect on read/write channels that are not blocked.

For MI-side mode, the downstream network must be in a state in which it can accept new commands before the firewall is unblocked. This is typically done by locally resetting all downstream AXI IP.

For SI-side mode, the upstream network must be in a state in which it can begin issuing new commands (with no old transactions pending) before the firewall is unblocked. This is typically done by locally resetting all upstream AXI IP.

The global **aresetn** on the firewall IP resets the entire IP to the unblocked state. When the firewall IP is reset, both the upstream and downstream networks should be reset concurrently. Command propagation commences soon after **aresetn** is deasserted.

The recommended recovery sequence is as follows:

1. Detect a blocked read and/or write condition by either sampling the fault status bits of the appropriate MI-Side Fault Status Register/SI-Side Fault Status Register or responding to an interrupt triggered by the **mi/si_r/w_error** output signals.

   Note: The BUSY bits in the MI-Side Fault Status Register/SI-Side Fault Status Register can transition during normal operation and do not indicate that any fault has occurred.

2. For MI-side mode, discontinue further issuing of both read and write transactions into the firewall SI. If necessary, wait for any latency to lapse for new transactions to reach the firewall SI. Do this regardless of whether either the write or read traffic remains unblocked.

3. Wait for both read and write BUSY bits to become deasserted in the MI-Side Fault Status Register/SI-Side Fault Status Register. Do this regardless of whether the write or read traffic remains unblocked.

4. Reset all devices on the side of the firewall being monitored for faults and for a minimum recommended duration of 16 clock cycles, then release the reset. For MI-side mode, if necessary, wait for any latency to lapse for any outstanding R or B channel response transfers to reach the firewall MI.

   Note: It is not important to wait until the connected devices recover from reset.

5. Request firewall unblock by writing to the appropriate MI-Side Unblock Control Register/SI-Side Unblock Control Register. Wait for the write to complete (**s_axi_ctl_bvalid** becomes asserted).

6. Resume issuing new transactions to the firewall SI.

Operating States

Operating states are described in Table 1-1 and Table 1-2.

Table 1-1: **MI-Side Mode Operating States**

State	mi_{r,w}_error	{READ,WRITE}_RESPONSE_BUSY	s_axi_{aw,ar,w}ready	m_axi_{aw,ar,w}valid	m_axi_{r,b}ready	Description
Normal	0	varies	from m_axi {aw,ar,w}ready	from s_axi {aw,ar,w}valid	from s_axi {r,b}ready	Commands received on SI propagate to MI. Responses received on MI propagate to SI if they do not violate protocol.
Flushing	1	1	1	0	1	Firewall is blocked; outstanding transactions to be flushed on SI. Commands received on SI do not propagate to MI and are added to the flush queue. Responses received on MI do not propagate to SI and are ignored.

Table 1-1: **MI-Side Mode Operating States *(Cont'd)***

State	mi_{r,w}_error	{READ,WRITE}_RESPONSE_BUSY	s_axi_{aw,ar,w}ready	m_axi_{aw,ar,w}valid	m_axi_{r,b}ready	Description
Blocked	1	0	1	0	1	Firewall is still blocked; no further transactions remain to be flushed on SI. Commands received on SI do not propagate to MI and are added to the flush queue, causing the IP to revert to the Flushing state. Responses received on MI do not propagate to SI and are ignored. When an unblock request is received, the IP transitions to the Normal state immediately.
Unblock Pending	1	1	1	0	1	Unblock requested while still busy (not recommended). Commands received on SI do not propagate to MI and are added to the flush queue, causing the IP to remain in the Unblock Pending state longer. Responses received on MI do not propagate to SI and are ignored. When all outstanding read and write transactions have been flushed, the IP transitions to the Normal state.

Table 1-2: **SI-Side Mode Operating States**

State	si_{r,w}_error	{READ,WRITE}_RESPONSE_BUSY	s_axi_{r,b}valid	s_axi_{aw,ar,w}ready	m_axi_{r,b}ready	Description
Normal	0	varies	from m_axi_{r,b}valid	from m_axi_{aw,ar,w}ready	from s_axi{r,b}ready	Commands received on SI propagate to MI if they do not violate protocol. Responses received on MI propagate to SI.
Flushing	1	1	0	1	1	Firewall is blocked; outstanding transactions to be flushed on MI. Commands received on SI do not propagate to MI and are ignored. Responses received on MI do not propagate to SI.
Blocked	1	0	0	1	1	Firewall is still blocked; no further transactions remain to be flushed on MI. Commands received on SI do not propagate to MI and are ignored. After flushing is done, no additional responses are expected to be received on the MI for the affected channel. When an unblock request is received, the IP transitions to the Normal state immediately.
Unblock Pending	1	1	0	1	1	Unblock requested while still busy (not recommended). Commands received on SI do not propagate to MI and are ignored. After flushing is done, no additional responses are expected to be received on the MI for the affected channel. When all outstanding read and write transactions have been flushed, the IP transitions to the Normal state.

CAM Overflow

When operating in MI-side mode, the **NUM_{READ,WRITE}_THREADS** parameters determine the number of CAM entries to be implemented in the IP. Once all CAM locations are filled, any further commands having a non-matching ID cannot be propagated to the MI; otherwise protocol checking becomes unreliable. If a transaction is received on the SI with a non-matching ID while the CAM is full, the corresponding **s_axi_{ar,aw}ready** is deasserted and the SI stalls (for read or write) until a vacancy is created in the CAM. The transaction causing the CAM overflow stall may be accepted (handshake completion) on the SI, even though it is not propagated to the MI. This CAM-full stall condition does not trigger any fault. Thread-tracking CAMs are not needed for SI-side mode operation; the NUM_*THREADS parameters are ignored.

Similarly, the **NUM_{READ,WRITE}_OUTSTANDING** parameters determine the size of the outstanding transaction counters to be implemented for both MI-side and SI-side modes. Once a counter reaches its maximum value, further commands are also stalled by deasserting **s_axi_{ar,aw}ready**. (Again, no fault is triggered by reaching the maximum limit of an outstanding transaction counter.)

Timeout Faults

The following timeout conditions block the firewall in MI-side mode, depending on corresponding **MAXWAIT** register:

- RECS_AWREADY_MAX_WAIT: AWREADY should be asserted within MAXWAITS cycles of AWVALID being asserted.
- RECS_WREADY_MAX_WAIT: WREADY should be asserted within MAXWAITS cycles of WVALID being asserted.
- RECS_ARREADY_MAX_WAIT: ARREADY should be asserted within MAXWAITS cycles of ARVALID being asserted.
- RECS_CONTINUOUS_RTRANSFERS_MAX_WAIT: RVALID should be asserted within MAXWAITS cycles of either AR command transfer or previous R transfer while there are outstanding AR commands.
- RECS_WRITE_TO_BVALID_MAX_WAIT: BVALID should be asserted within MAXWAITS cycles of AW command transfer or WLAST transfer (whichever is later), or previous B transfer if there are yet more AW and WLAST transfers outstanding.

The following timeout conditions block the firewall in SI-side mode, depending on corresponding **MAXWAIT** register:

- RECM_BREADY_MAX_WAIT: BREADY should be asserted within MAXWAITS cycles of BVALID being asserted.
- RECM_RREADY_MAX_WAIT: RREADY should be asserted within MAXWAITS cycles of RVALID being asserted.

- RECM_CONTINUOUS_WTRANSFERS_MAX_WAIT: WVALID should be asserted within MAXWAITS cycles of either AWVALID or previous W transfer while there are outstanding AW commands.
- RECM_WVALID_TO_AWVALID_MAX_WAIT: AWVALID should be asserted within MAXWAITS cycles of a premature WVALID.

For all timeout checks, any change in value written to any MAX_WAIT control register takes effect only while the watchdog timer is not actively counting. That is, while the earlier qualifying condition (such as **`arvalid`** assertion) is false or while the later triggering condition (such as **`arready`** assertion) is true, watchdog timer counting for the corresponding check is disabled and the counter is continually re-loaded with the value currently stored in the corresponding MAX_WAIT register. Once timeout count begins, the last MAX_WAIT value loaded is used to determine the triggering of the fault.

MI-Side Mode Protocol Violation Faults

The following fatal protocol violations always block the firewall. The MI-side transfer that triggers the block is not propagated to the SI. Once blocked, the SI autonomously issues protocol-compliant responses to all incomplete or partially-incomplete outstanding transactions.

- ERRS_BRESP: A slave must only give a write response after both the write address and the last write data item are transferred and the BID, if any, must match an outstanding AWID.
- ERRS_RDATA_NUM: The number of read data items must match the corresponding ARLEN (does not apply to AXI4-Lite protocol).
- ERRS_RID: Slave can only give read data in response to an outstanding read transaction, and the RID, if any, must match an outstanding ARID.
- ERRS_BVALID_STABLE: Once BVALID is asserted, it must remain asserted until BREADY is High.
- ERRS_RVALID_STABLE: Once RVALID is asserted, it must remain asserted until RREADY is High.

SI-Side Mode Protocol Violation Faults

The following fatal protocol violations always block the firewall. The SI-side transfer that triggers the block is not propagated to the MI. Once blocked, the MI autonomously issues protocol-compliant write data transfers for any incomplete or partially-incomplete outstanding write transactions.

- ERRM_AWSIZE: The size of a write transfer must not exceed the width of the data interface.

- ERRM_ARSIZE: The size of a read transfer must not exceed the width of the data interface.
- ERRM_WDATA_NUM: The number of write data items must match the corresponding AWLEN.
- ERRM_AWADDR_BOUNDARY: A write burst must not cross a 4 KB boundary.
- ERRM_ARADDR_BOUNDARY: A read burst must not cross a 4 KB boundary.
- ERRM_AWVALID_STABLE: Once AWVALID is asserted, it must remain asserted until AWREADY is High.
- ERRM_WVALID_STABLE: Once WVALID is asserted, it must remain asserted until WREADY is High.
- ERRM_ARVALID_STABLE: Once ARVALID is asserted, it must remain asserted until ARREADY is High.

Error Response Faults

The following in-band AXI error responses, observed on the MI, optionally block the firewall and/or signal an interrupt, when enabled in the SI-Side Interrupt Enable Register/MI-Side Interrupt Enable Register. The offending command cannot be prevented from propagating to the MI.

- XILINX_RD_SLVERR: Detected Read SLVERR.
- XILINX_RD_DECERR: Detected Read DECERR.
- XILINX_WR_SLVERR: Detected Write SLVERR.
- XILINX_WR_DECERR: Detected Write DECERR.

SLVERR and DECERR Summary

The behavior of AXI Firewall IP v1.1, regarding the observation of SLVERR or DECERR response code events, is affected by the following conditions:

- In the MI-Side Interrupt Enable Register/SI-Side Interrupt Enable Register, the "Enable firewall blocking" (Bits[31:30, 15:14])
- In the MI-Side Interrupt Enable Register, the "Enable interrupt" bits for SLVERR/DECERR (Bits[23:22, 7:6])
- In the SI-Side Interrupt Enable Register, the "Enable interrupt" bits for SLVERR/DECERR (Bits[26:25, 6:5])
- Device Global Interrupt Enable Register
- MASK_ERR_RESP parameter

The AXI Firewall IP behaviors that are affected by SLVERR or DECERR response code events are:

- Blocking the firewall (and asserting the **{si,mi}_{w,r}_err** output).
  - Blocking due to SLVERR/DECERR is enabled by the "Enable firewall blocking" (Bits[31:30, 15:14]) in the MI-Side Interrupt Enable Register/SI-Side Interrupt Enable Register.
- Asserting an interrupt request (**ip2intc_irpt** output).
  - Interrupt requests due to SLVERR/DECERR are enabled by the "Enable interrupt" bits in the MI-Side Interrupt Enable Register/SI-Side Interrupt Enable Register, in conjunction with the Device Global Interrupt Enable Register.
- Asserting a SLVERR/DECERR syndrome bit in the Fault Status Register.
  - Asserting a SLVERR/DECERR syndrome bit is enabled by setting either the "Enable firewall blocking" bit or the "Enable interrupt" bit for the corresponding response code in the MI-Side Interrupt Enable Register/SI-Side Interrupt Enable Register (regardless of the Device Global Interrupt Enable Register).
- Propagating SLVERR/DECERR values on the **RRESP/BRESP** signal outputs on the SI, including driving those outputs to SLVERR while flushing outstanding transactions by a MI-side Firewall.
  - Propagating SLVERR/DECERR on the SI is disabled by the MASK_ERR_RESP parameter.

The following table describes the detailed behavior when a SLVERR (**01**) or DECERR (**11**) value is sampled on the **RRESP** or **BRESP** signal on the MI interface (regardless of whether the Firewall is in MI-side Firewall Mode or SI-side Firewall Mode).

Table 1-3: **SLVERR/DECERR Behavior**

Input Condition				Resulting Behavior			
MASK_ERR_RESP	**Enable Firewall Blocking Bit**	**Device Global Interrupt Enable**	**Enable Interrupt Bit for SLVERR/DECERR**	**s_axi_{r,b}resp Output**	**{si,mi}_{w,r}_err Output**	**Fault Status Register**	**ip2intc_irpt Output**
0 (default)	x	x	x	Propagate m_axi_{r,b}resp input			
1	x	x	x	Drive OKAY ("00") if m_axi_{r,b}resp=={OKAY,SLVERR,DECERR} (Propagate EXOKAY)			

Table 1-3: **SLVERR/DECERR Behavior *(Cont'd)***

Input Condition				Resulting Behavior			
MASK_ERR_RESP	**Enable Firewall Blocking Bit**	**Device Global Interrupt Enable**	**Enable Interrupt Bit for SLVERR/ DECERR**	**s_axi_{r,b}resp Output**	**{si,mi}_{w,r}_err Output**	**Fault Status Register**	**ip2intc_irpt Output**
x	0 (default)	x	0 (default)		0 (don't block)	SLVERR/ DECERR bits = 0 (indicate no fault)	
x	0 (default)	x	1		0 (don't block)	SLVERR/ DECERR bit = 1 (indicate correspon ding fault)	
x	1	x	x		1 (block firewall)	SLVERR/ DECERR bit = 1 (indicate correspon ding fault)	
x	x	0 (default)	x				0 (no interrupt)
x	x	1	0 (default)				0 (no interrupt)
x	x	1	1				1 (interrupt request)

Undetectable Protocol Violations

There is a common master device design flaw listed as a protocol violation in the *AMBA AXI Protocol Specification* [Ref 1] that is not possible to detect by the firewall. The protocol specification allows slave interfaces to delay asserting AWREADY until after it receives both AWVALID and WVALID. This is a useful way to streamline the slave interface design, especially for AXI4-Lite slaves. To avoid the possibility of deadlock, the protocol specification prohibits any master interface from delaying WVALID until after it receives AWREADY. If a firewall configured in SI-side mode (or an AXI Protocol Checker IP) is placed downstream of a master device that violates this rule, it has no way to detect this violation. It is not illegal for a master to assert AWVALID before WVALID, and there is no way for the firewall to determine whether the delay is due to the master waiting for AWREADY to arrive. If a firewall configured in MI-side mode is placed downstream of a master device that violates this rule, it can trigger a timeout fault on the expected AWREADY assertion that never occurs, but its root-cause would be due to a design flaw in the master.

Applications

The AXI Firewall IP is deployed in MI-side mode when a reliable upstream system accesses a downstream subsystem of questionable reliability. A common example is in hardware acceleration systems, in which user-defined acceleration kernels are loaded into a re-programmable region of the FPGA. Potentially fatal conditions that might arise in the user region must be prevented from blocking host access to the FPGA, typically through the PCIe® bridge, which must remain operational.

The AXI Firewall IP is deployed in SI-side mode when a reliable downstream system is accessed by an upstream subsystem of questionable reliability. A common example is in hardware acceleration systems, in which a user-defined acceleration kernel loaded into a re-programmable region of the FPGA acts as a master to access remote memory locations through the PCIe bridge. Potentially fatal conditions that might arise in the kernel must be prevented from corrupting PCIe traffic or the remote system.

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

The AXI Firewall IP core supports the AXI3, AXI4, and AXI4-Lite protocols.

Standards

The AXI interfaces conform to the Arm® Advanced Microcontroller Bus Architecture (AMBA®) AXI version 4 specification [Ref 1], including the AXI4-Lite control register interface subset.

User Parameters

Table 2-1 shows the AXI Firewall core user parameters.

Table 2-1: **AXI Firewall User Parameters**

Parameter Name	Format/Range	Default Value	Description
FIREWALL_MODE	string = {MI, SI}	MI	MI-side mode monitors responses received on the MI for fatal conditions and prevents them from impacting upstream operation. SI-side mode monitors forward transfers received on the SI for fatal conditions and prevents them from impacting downstream operation.
ADDR_WIDTH	1 ≤ integer ≤ 64	32	Width of {s,m}_axi_{ar,aw}addr
ID_WIDTH	0 ≤ integer ≤ 32	0	Width of {s,m}_axi_{ar,aw,w,r,b}id.
DATA_WIDTH	integer = {32, 64, 128, 256, 512, 1024}	32	Width of {s,m}_axi_{r,w}data
ARUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of {s,m}_axi_aruser.
AWUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of {s,m}_axi_awuser.
WUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of {s,m}_axi_wuser.
RUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of {s,m}_axi_ruser.
BUSER_WIDTH	0 ≤ integer ≤ 1024	0	Width of {s,m}_axi_buser.

Table 2-1: **AXI Firewall User Parameters *(Cont'd)***

Parameter Name	Format/Range	Default Value	Description
PROTOCOL	string = {AXI4, AXI3, AXI4LITE}	AXI4	Used for protocol-specific port widths/enablement.
NUM_READ_THREADS, NUM_WRITE_THREADS	1 ≤ integer ≤ 16	1	Number of ID threads to implement in protocol check CAMs (used only in MI-side mode).
NUM_READ_OUTSTANDING, NUM_WRITE_OUTSTANDING	0 ≤ integer ≤ 32	1	Number of outstanding transactions allowed before stalling the SI. 0 = IP is disabled for read/write.
ENABLE_PIPELINING	integer = {0,1}	1	Enable conditional boundary register slices.
MASK_ERR_RESP	integer = {0,1}	0	0 = Propagate m_axi_{b,r}resp to SI. Drive s_axi_{b,r}resp = SLVERR ("10") while flushing {B,R}-channel responses during MI-Firewall block. 1 = Force s_axi_{b,r}resp =OKAY ("00") when m_axi_{b,r}resp==={SLVERR,DECERR}. Drive s_axi_{b,r}resp = OKAY while flushing {B,R}-channel responses during MI-Firewall block. Continue to propagate EXOK ("01") from MI.

Port Descriptions

Table 2-2 shows the AXI Firewall IP global signals.

Table 2-2: **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)

Table 2-3 shows the AXI Firewall IP Slave Interface signals.

Table 2-3: **Slave Interface Signals**

Signals	I/O	Default	Width	Description
s_axi_awid	I	AXI3, AXI4: 0 AXI4-Lite: d/c	ID_WIDTH	Write Address Channel Transaction ID
s_axi_awaddr	I	REQ	ADDR_WIDTH	Write Address Channel Address

Table 2-3: **Slave Interface Signals *(Cont'd)***

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

Table 2-3: **Slave Interface Signals *(Cont'd)***

Signals	I/O	Default	Width	Description
s_axi_arid	I	AXI3, AXI4: 0 AXI4-Lite: d/c	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-Lite: d/c	AXI4: 8 AXI3: 4	Read Address Channel Burst Length code (0–255)
s_axi_arsize	I	AXI3, AXI4: REQ AXI4-Lite: d/c	3	Read Address Channel Transfer Size Code (0–7)
s_axi_arburst	I	AXI3, AXI4: REQ AXI4-Lite: d/c	2	Read Address Channel Burst Type (0–2)
s_axi_arlock	I	AXI3, AXI4: 0 AXI4-Lite: d/c	AXI4: 1 AXI3: 2	Read Address Channel Atomic Access Type (0, 1)
s_axi_arcache	I	AXI3, AXI4: 0 AXI4-Lite: d/c	4	Read Address Channel Cache Characteristics
s_axi_arprot	I	0b000	3	Read Address Channel Protection Bits
s_axi_arregion	I	AXI4: 0; AXI3, AXI4-Lite: d/c	4	AXI4 Read Address Channel address Region Index
s_axi_arqos	I	AXI4: 0 AXI4-Lite: d/c	4	AXI4 Read Address Channel Quality of Service
s_axi_aruser	I	AXI3, AXI4: 0 AXI4-Lite: d/c	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
s_axi_rready	I	REQ	1	Read Data Channel Ready

Table 2-4 shows the AXI Firewall IP Master Interface signals.

Table 2-4: **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

Table 2-4: **Master Interface Signals *(Cont'd)***

Signals	I/O	Default	Width	Description
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 AXI4-Lite: d/c	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 AXI4-Lite: d/c	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
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

Table 2-4: **Master Interface Signals *(Cont'd)***

Signals	I/O	Default	Width	Description
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 AXI4-Lite: d/c	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 AXI4-Lite: d/c	1	Read Data Channel Last Data Beat
m_axi_ruser	I	AXI3, AXI4: 0 AXI4-Lite: d/c	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

Table 2-5 shows the AXI Firewall IP Control Register Interface signals.

Table 2-5: **Control Register Interface Signals**

Signals	I/O	Default	Width	Description
s_axi_ctl_awaddr	I	REQ	12	Write Address Channel Address
s_axi_ctl_awvalid	I	REQ	1	Write Address Channel Valid
s_axi_ctl_awready	O		1	Write Address Channel Ready
s_axi_ctl_wdata	I	REQ	32	Write Data Channel Data
s_axi_ctl_wstrb	I	All ones	4	Write Data Channel Byte Strobes
s_axi_ctl_wvalid	I	REQ	1	Write Data Channel Valid
s_axi_ctl_wready	O		1	Write Data Channel Ready
s_axi_ctl_bresp	O		2	Write Response Channel Response Code (0–3)
s_axi_ctl_bvalid	O		1	Write Response Channel Valid
s_axi_ctl_bready	I	REQ	1	Write Response Channel Ready
s_axi_ctl_araddr	I	REQ	12	Read Address Channel Address
s_axi_ctl_arvalid	I	REQ	1	Read Address Channel Valid
s_axi_ctl_arready	O		1	Read Address Channel Ready
s_axi_ctl_rdata	O		32	Read Data Channel Data
s_axi_ctl_rresp	O		2	Read Data Channel Response Code (0–3)

Table 2-5: **Control Register Interface Signals**

Signals	I/O	Default	Width	Description
s_axi_ctl_rvalid	O		1	Read Data Channel Valid
s_axi_ctl_rready	I	REQ	1	Read Data Channel Ready

Table 2-6 shows the AXI Firewall IP Non-AXI signals.

Table 2-6: **Non-AXI Signals**

Signals	I/O	Width	Enablement	Description
mi_w_error	O	1	Always	Indicates MI-side firewall is blocked for writes due to a downstream write fault (sticky until unblocked).
mi_r_error	O	1	Always	Indicates MI-side firewall is blocked for reads due to a downstream read fault (sticky until unblocked).
si_w_error	O	1	Always	Indicates SI-side firewall is blocked for writes due to an upstream write fault (sticky until unblocked).
si_r_error	O	1	Always	Indicates SI-side firewall is blocked for reads due to an upstream read fault (sticky until unblocked).
ip2intc_irpt	O	1	Always	IRQ, as enabled by interrupt control registers.

Register Space

The registers shown in Table 2-7 are accessible through the AXI4-Lite control interface:

Table 2-7: **AXI Firewall IP Register Map**

Offset	Access Type	Register	MI-side Mode Usage	SI-side Mode Usage
0x0	RO	MI-Side Fault Status Register	✓	
0x4	WO	MI-Side Soft Fault Control Register	✓	
0x8	WO	MI-Side Unblock Control Register	✓	
0x10	RO	IP Version Register	✓	✓
0x30	RW	MAX_CONTINUOUS_RTRANSFERS_WAITS Register (0x30)	✓	
0x34	RW	MAX_WRITE_TO_BVALID_WAITS Register	✓	
0x38	RW	MAX_ARREADY_WAITS Register (0x38)	✓	
0x3C	RW	MAX_AWREADY_WAITS Register (0x3C)	✓	
0x40	RW	MAX_WREADY_WAITS Register (0x40)	✓	
0x100	RO	SI-Side Fault Status Register (0x100)		✓
0x104	WO	SI-Side Soft Fault Control Register (0x104)		✓
0x108	WO	SI-Side Unblock Control Register (0x108)		✓
0x130	RW	MAX_CONTINUOUS_WTRANSFERS_WAITS Register (0x130)		✓
0x134	RW	MAX_WVALID_TO_AWVALID_WAITS Register (0x134)		✓
0x138	RW	MAX_RREADY_WAITS Register (0x138)		✓
0x13C	RW	MAX_BREADY_WAITS Register (0x13C)		✓
0x200	RW	Device Global Interrupt Enable Register (0x200)	✓	✓
0x204	RW	MI-Side Interrupt Enable Register (0x204)	✓	
0x208	RW	SI-Side Interrupt Enable Register (0x208)		✓
0x210	RO	Final ARADDR Low Register (0x210)	✓	✓
0x214	RO	Final ARADDR High Register (0x214)	✓	✓
0x218	RO	Final AWADDR Low Register (0x218)	✓	✓
0x21C	RO	Final AWADDR High Register (0x21C)	✓	✓
0x220	RO	Final ARUSER Register (0x220)	✓	✓
0x224	RO	Final AWUSER Register (0x224)	✓	✓
0x230	RW	Timeout Prescaler Register (0x230)	✓	✓
0x234	RW	Timeout Initial Delay Register (0x234)	✓	✓

MI-Side Fault Status Register

The MI-Side Fault Status register is shown in Table 2-8.

Table 2-8: **MI-Side Fault Status Register (0x0)**

Bit(s)	Name	Core Access	Default Value	Description
31:24	Reserved	RO		All zeros
23	XILINX_WR_DECERR	RO	0	
22	XILINX_WR_SLVERR	RO	0	
21	ERRS_BVALID_STABLE	RO	0	
20	ERRS_BRESP	RO	0	
19	RECS_WRITE_TO_BVALID_MAX_WAIT	RO	0	
18	RECS_WREADY_MAX_WAIT	RO	0	
17	RECS_AWREADY_MAX_WAIT	RO	0	
16	WRITE_RESPONSE_BUSY	RO	0	Indicates whether there are outstanding write transactions that still need to be completed on the SI; deasserts as soon as outstanding writes decrement to zero. (Not masked while writes are in normal operating mode.)
15:8	Reserved	RO		All zeros
7	XILINX_RD_DECERR	RO	0	
6	XILINX_RD_SLVERR	RO	0	
5	ERRS_RVALID_STABLE	RO	0	
4	ERRS_RID	RO	0	
3	ERRS_RDATA_NUM	RO	0	
2	RECS_CONTINUOUS_RTRANSFERS_MAX_WAIT	RO	0	
1	RECS_ARREADY_MAX_WAIT	RO	0	
0	READ_RESPONSE_BUSY	RO	0	Indicates whether there are outstanding read transactions that still need to be completed on the SI; deasserts as soon as outstanding reads decrements to zero. (Not masked while reads are in normal operating mode.)

MI-Side Soft Fault Control Register

The MI-Side Soft Fault Control register is shown in Table 2-9.

Table 2-9: **MI-Side Soft Fault Control Register (0x4)**

Bit(s)	Name	Core Access	Default Value	Description
31:24	Reserved	WO		Reserved
23:17	Trigger Write Block	WO	0	Copy bit pattern to MI-side Fault Status Register Bits[23:17]. Self-clearing when block is triggered.
16:8	Reserved	WO		Reserved
7:1	Trigger Read Block	WO	0	Copy bit pattern to MI-side Fault Status Register Bits[7:1]. Self-clearing when block is triggered.
0	Reserved	WO		Reserved

MI-Side Unblock Control Register

The MI-Side Unblock Control Register is shown in Table 2-10.

Table 2-10: **MI-Side Unblock Control Register (0x8)**

Bit(s)	Name	Core Access	Default Value	Description
31:1	Reserved	WO		Reserved
0	Unblock request	WO	0	Request MI unblock (both read and write). Self-clearing when unblock completes.

IP Version Register

The IP Version Register is shown in Table 2-11.

Table 2-11: **IP Version Register (0x10)**

Bit(s)	Name	Core Access	Default Value	Description
31:0	IP Version	RO	0x1	0x0 = axi_firewall_v1_0 0x1 = axi_firewall_v1_1

MAX_CONTINUOUS_RTRANSFERS_WAITS Register

The MAX_CONTINUOUS_RTRANSFERS_WAITS Register is shown in Table 2-12.

Table 2-12: **MAX_CONTINUOUS_RTRANSFERS_WAITS Register (0x30)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0	MAX_CONTINUOUS_RTRANSFERS_WAITS	RW	16'hFFFF	Run-time MAXWAITS value. Setting to zero disables this timeout check.

MAX_WRITE_TO_BVALID_WAITS Register

The MAX_WRITE_TO_BVALID_WAITS Register is shown in Table 2-13.

Table 2-13: **MAX_WRITE_TO_BVALID_WAITS Register (0x34)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0	MAX_WRITE_TO_BVALID_WAITS	RW	16'hFFFF	Run-time MAXWAITS value Setting to zero disables this timeout check.

MAX_ARREADY_WAITS Register

The MAX_ARREADY_WAITS Register is shown in Table 2-14.

Table 2-14: **MAX_ARREADY_WAITS Register (0x38)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0	MAX_ARREADY_WAITS	RW	16'hFFFF	Run-time MAXWAITS value Setting to zero disables this timeout check.

MAX_AWREADY_WAITS Register

The MAX_AWREADY_WAITS Register is shown in Table 2-15.

Table 2-15: **MAX_AWREADY_WAITS Register (0x3C)**

Bit(s)	Name	Core Access	Default Value	Description
31:15	Reserved			Reserved
15:0	MAX_AWREADY_WAITS	RW	16'hFFFF	Run-time MAXWAITS value. Setting to zero disables this timeout check.

MAX_WREADY_WAITS Register

The MAX_WREADY_WAITS Register is shown in Table 2-16.

Table 2-16: **MAX_WREADY_WAITS Register (0x40)**

Bit(s)	Name	Core Access	Default Value	Description
31:15	Reserved			Reserved
15:0	MAX_WREADY_WAITS	RW	16'hFFFF	Run-time MAXWAITS value. Setting to zero disables this timeout check.

SI-Side Fault Status Register

The SI-Side Fault Status register is shown in Table 2-17.

Table 2-17: **SI-Side Fault Status Register (0x100)**

Bit(s)	Name	Core Access	Default Value	Description
31:27	Reserved	RO		All zeros
26	XILINX_WR_DECERR	RO	0	
25	XILINX_WR_SLVERR	RO	0	
24	ERRM_WVALID_STABLE	RO	0	
23	ERRM_AWVALID_STABLE(1)	RO	0	
22	ERRM_AWADDR_BOUNDARY(1)	RO	0	
21	ERRM_WDATA_NUM	RO	0	
20	ERRM_AWSIZE(1)	RO	0	
19	RECM_WVALID_TO_AWVALID_MAX_WAIT	RO	0	
18	RECM_CONTINUOUS_WTRANSFERS_MAX_WAIT	RO	0	
17	RECM_BREADY_MAX_WAIT	RO	0	

Table 2-17: **SI-Side Fault Status Register (0x100) *(Cont'd)***

Bit(s)	Name	Core Access	Default Value	Description
16	WRITE_RESPONSE_BUSY	RO	0	Indicates whether there are outstanding write transactions that still need to be completed on the MI; deasserts as soon as outstanding writes decrement to zero. (Not masked while writes are in normal operating mode.)
15:7	Reserved	RO		All zeros
6	XILINX_RD_DECERR	RO	0	
5	XILINX_RD_SLVERR	RO	0	
4	ERRM_ARVALID_STABLE(2)	RO	0	
3	ERRM_ARADDR_BOUNDARY(2)	RO	0	
2	ERRM_ARSIZE(2)	RO	0	
1	RECM_RREADY_MAX_WAIT	RO	0	
0	READ_RESPONSE_BUSY	RO	0	Indicates whether there are outstanding read transactions that still need to be completed on the MI; deasserts as soon as outstanding reads decrements to zero. (Not masked while reads are in normal operating mode.)

Notes:

1. When this violation is asserted, the Final AWADDR Low/High Registers and Final AWUSER Register contain the actual values of the offending command.
2. When this violation is asserted, the Final ARADDR Low/High Registers and Final ARUSER Register contain the actual values of the offending command.

SI-Side Soft Fault Control Register

The SI-Side Soft Fault Control register is shown in Table 2-18.

Table 2-18: **SI-Side Soft Fault Control Register (0x104)**

Bit(s)	Name	Core Access	Default Value	Description
31:27	Reserved	WO		Reserved
26:17	Trigger Write Block	WO	0	Copy bit pattern to SI-side Fault Status Register Bits[26:17]. Self-clearing when block is triggered.
16:7	Reserved	WO		Reserved
6:1	Trigger Read Block	WO	0	Copy bit pattern to SI-side Fault Status Register Bits[6:1]. Self-clearing when block is triggered.
0	Reserved	WO		Reserved

SI-Side Unblock Control Register

The SI-Side Unblock Control Register is shown in Table 2-19.

Table 2-19: **SI-Side Unblock Control Register (0x108)**

Bit(s)	Name	Core Access	Default Value	Description
31:1	Reserved	WO		Reserved
0	Unblock request	WO	0	Request SI unblock (both read and write). Self-clearing when unblock completes.

MAX_CONTINUOUS_WTRANSFERS_WAITS Register

The MAX_CONTINUOUS_WTRANSFERS_WAITS Register is shown in Table 2-20.

Table 2-20: **MAX_CONTINUOUS_WTRANSFERS_WAITS Register (0x130)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0	MAX_CONTINUOUS_WTRANSFERS_WAITS	RW	16'hFFFF	Run-time MAXWAITS value. Setting to zero disables this timeout check.

MAX_WVALID_TO_AWVALID_WAITS Register

The MAX_WVALID_TO_AWVALID_WAITS Register is shown in Table 2-21.

Table 2-21: **MAX_WVALID_TO_AWVALID_WAITS Register (0x134)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0	MAX_WVALID_TO_AWVALID_WAITS	RW	16'hFFFF	Run-time MAXWAITS value Setting to zero disables this timeout check.

MAX_RREADY_WAITS Register

The MAX_RREADY_WAITS Register is shown in Table 2-22.

Table 2-22: **MAX_RREADY_WAITS Register (0x138)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0	MAX_RREADY_WAITS	RW	16'hFFFF	Run-time MAXWAITS value Setting to zero disables this timeout check.

MAX_BREADY_WAITS Register

The MAX_BREADY_WAITS Register is shown in Table 2-23.

Table 2-23: **MAX_BREADY_WAITS Register (0x13C)**

Bit(s)	Name	Core Access	Default Value	Description
31:15	Reserved	RW		Reserved
15:0	MAX_BREADY_WAITS	RW	16'hFFFF	Run-time MAXWAITS value. Setting to zero disables this timeout check.

Device Global Interrupt Enable Register

The Device Global Interrupt Enable Register is shown in Table 2-24.

Table 2-24: **Device Global Interrupt Enable Register (0x200)**

Bit(s)	Name	Core Access	Default Value	Description
31:1	Reserved	RW		Reserved
0	Global Interrupt Enable	RW	0	When FIREWALL_MODE = MI, asserts ip2intc_irpt output when any of the faults enabled by the MI-side Interrupt Enable Register are active. When FIREWALL_MODE = SI, asserts ip2intc_irpt output when any of the faults enabled by the SI-side Interrupt Enable Register are active.

MI-Side Interrupt Enable Register

The MI-Side Interrupt Enable Register is shown in Table 2-25.

Table 2-25: **MI-Side Interrupt Enable Register (0x204)**

Bit(s)	Name	Core Access	Default Value	Description
31		RW	0	Enable firewall blocking due to XILINX_WR_DECERR.(1)
30		RW	0	Enable firewall blocking due to XILINX_WR_SLVERR.(1)
29:24	Reserved	RW		Reserved
23		RW	0	Enable interrupt for XILINX_WR_DECERR events.
22		RW	0	Enable interrupt for XILINX_WR_DECERR events.
21:17		RW		Enable interrupt for corresponding fault in MI-side Fault Status Register Bits[21:17].
16	Reserved	RW		Reserved (Always zero)
15		RW	0	Enable firewall blocking due to XILINX_RD_DECERR.(1)
14		RW	0	Enable firewall blocking due to XILINX_RD_SLVERR.(1)
13:8	Reserved	RW		Reserved
5:1		RW	0	Enable interrupt for corresponding fault in MI-side Fault Status Register Bits[5:1].
0	Reserved	RW		Reserved (Always zero)

Notes:

1. When any of Bits[31:30, 15:14] are disabled (0), the corresponding error response event can be recorded in the MI-side Fault Status Register and can still issue an interrupt if the corresponding "Enable interrupt" bit is set, but will not cause the firewall to become blocked.

SI-Side Interrupt Enable Register

The SI-Side Interrupt Enable Register is shown in Table 2-26.

Table 2-26: **SI-Side Interrupt Enable Register (0x208)**

Bit(s)	Name	Core Access	Default Value	Description
31		RW	0	Enable firewall blocking due to XILINX_WR_DECERR.(1)
30		RW	0	Enable firewall blocking due to XILINX_WR_SLVERR.(1)
29:27	Reserved	RW		Reserved
26		RW	0	Enable interrupt for XILINX_WR_DECERR events.
25		RW	0	Enable interrupt for XILINX_WR_SLVERR events.
24:17		RW	0	Enable interrupt for corresponding fault in SI-side Fault Status Register Bits[24:17].
16	Reserved	RW		Reserved (Always zero)
15		RW	0	Enable firewall blocking due to XILINX_RD_DECERR.(1)

Table 2-26: **SI-Side Interrupt Enable Register (0x208) *(Cont'd)***

Bit(s)	Name	Core Access	Default Value	Description
14		RW	0	Enable firewall blocking due to XILINX_RD_SLVERR.(1)
13:7	Reserved	RW		Reserved
6		RW	0	Enable interrupt for XILINX_RD_DECERR events.
5		RW	0	Enable interrupt for XILINX_RD_SLVERR events.
4:1		RW	0	Enable interrupt for corresponding fault in SI-side Fault Status Register Bits[4:1].
0	Reserved	RW		Reserved (Always zero)

Notes:

1. When any of Bits[31:30, 15:14] are disabled (0), the corresponding error response event can be recorded in the SI-side Fault Status Register and can still issue an interrupt if the corresponding "Enable interrupt" bit is set, but will not cause the firewall to become blocked.

Final ARADDR Low Register

The Final ARADDR Low Register is shown in Table 2-27.

Table 2-27: **Final ARADDR Low Register (0x210)**

Bit(s)	Name	Core Access	Default Value	Description
31:0		RO	0	Final value observed on s_axi_araddr[31:0]. For SI-Firewall protocol violations of the AR command, this is the address of the offending command. For all other conditions, this is the most recent address observed prior to triggering a block.

Final ARADDR High Register

The Final ARADDR High Register is shown in Table 2-28.

Table 2-28: **Final ARADDR High Register (0x214)**

Bit(s)	Name	Core Access	Default Value	Description
31:0		RO	0	Final value observed on s_axi_araddr[63:32]. For SI-Firewall protocol violations of the AR command, this is the address of the offending command. For all other conditions, this is the most recent address observed prior to triggering a block.

Final AWADDR Low Register

The Final AWADDR Low Register is shown in Table 2-29.

Table 2-29: **Final AWADDR Low Register (0x218)**

Bit(s)	Name	Core Access	Default Value	Description
31:0		RO	0	Final value observed on s_axi_awaddr[31:0]. For SI-Firewall protocol violations of the AW command, this is the address of the offending command. For all other conditions, this is the most recent address observed prior to triggering a block.

Final AWADDR High Register

The Final AWADDR High Register is shown in Table 2-30.

Table 2-30: **Final AWADDR High Register (0x21C)**

Bit(s)	Name	Core Access	Default Value	Description
31:0		RO	0	Final value observed on s_axi_awaddr[63:32]. For SI-Firewall protocol violations of the AW command, this is the address of the offending command. For all other conditions, this is the most recent address observed prior to triggering a block.

Final ARUSER Register

The Final ARUSER Register is shown in Table 2-31.

Table 2-31: **Final ARUSER Register (0x220)**

Bit(s)	Name	Core Access	Default Value	Description
31:0		RO	0	Final value observed on s_axi_aruser[31:0]. For SI-Firewall protocol violations of the AR command, this is the aruser value of the offending command. For all other conditions, this is the most recent value observed prior to triggering a block.

Final AWUSER Register

The Final AWUSER Register is shown in Table 2-32.

Table 2-32: **Final AWUSER Register (0x224)**

Bit(s)	Name	Core Access	Default Value	Description
31:0		RO	0	Final value observed on s_axi_awuser[31:0]. For SI-Firewall protocol violations of the AW command, this is the awuser value of the offending command. For all other conditions, this is the most recent value observed prior to triggering a block.

Timeout Prescaler Register

The Timeout Prescaler Register is shown in Table 2-33.

Table 2-33: **Timeout Prescaler Register (0x230)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0		RW	0	Number of aclk cycles to wait before incrementing each of the MAX_WAITS timers. For each value n, each MAX_WAIT register (if enabled) prescribes a timeout period of MAX_WAIT * (n+1) aclk cycles.

Timeout Initial Delay Register

The Timeout Initial Delay Register is shown in Table 2-34.

Table 2-34: **Timeout Initial Delay Register (0x234)**

Bit(s)	Name	Core Access	Default Value	Description
31:16	Reserved	RW		Reserved
15:0		RW	0	Number of MAX_WAIT timer cycles (after prescaling) to delay incrementing all MAX_WAITS timers after reset/ power-on recovery. Following reset, incrementing of all MAX_WAITS timers is suppressed for TimeoutInitialDelay * (n+1) aclk cycles, where n is the Timeout Prescaler value.

Designing with the Core

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

General Design Guidelines

Example system design showing in AXI Firewall IP MI-side mode.

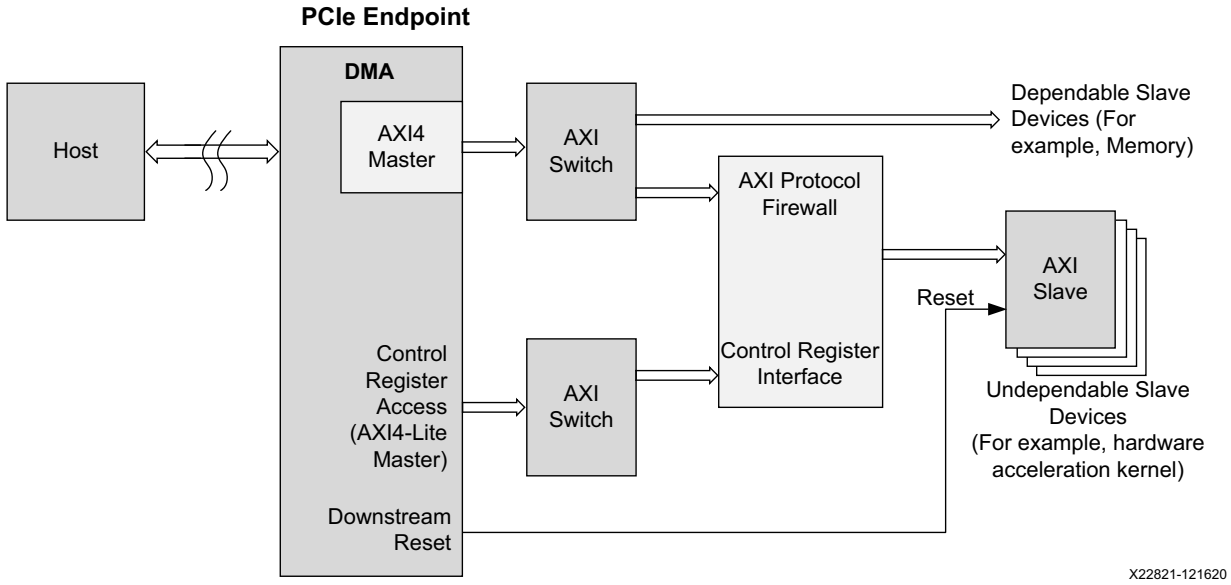

Figure 3-1: **AXI Firewall IP MI-Side Mode Example System Topology**

Clocking

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

Resets

The AXI Firewall 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 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 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 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

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) [Ref 3] 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 Vivado 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 4] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

Figure 4-1 shows the customization options for the AXI Protocol Firewall IP.

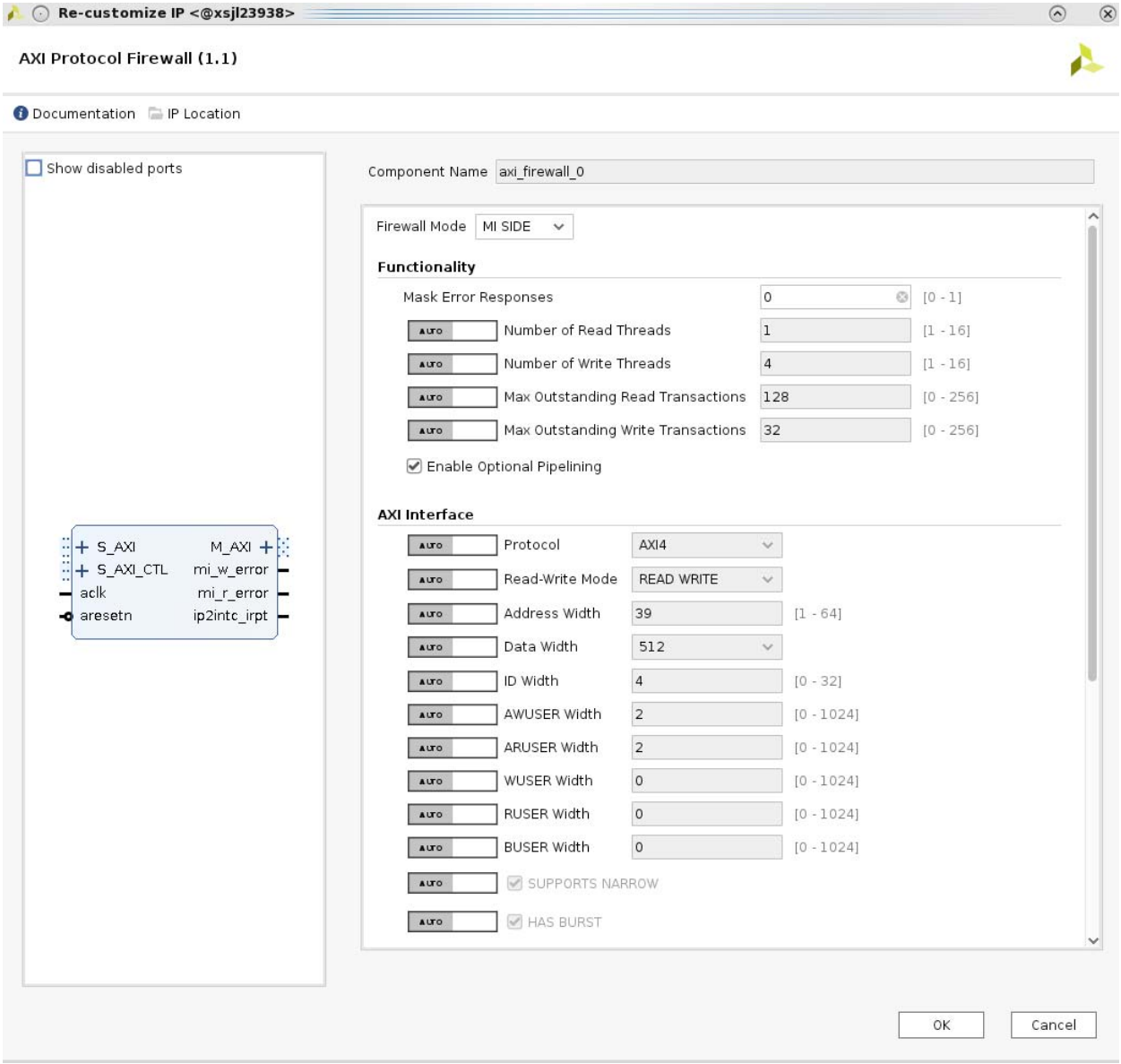

Figure 4-1: **AXI Protocol Firewall: Re-customize IP**

Upgrading

AXI Firewall IP v1.1 is backward-compatible replacement for AXI Firewall IP v1.0, in regards to both hardware design and run-time software (register map). Any existing system design containing AXI Firewall IP v1.0 will have the same behavior after allowing Vivado® to automatically upgrade to v1.1.

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 AXI Firewall IP, 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 AXI Firewall IP. 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 AXI Firewall IP

AR: 76037

Technical Support

Xilinx provides technical support at 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.

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 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. Instructions on how to download the Arm AMBA AXI specifications are at Arm AMBA Specifications. See the:
   - AMBA AXI4-Stream Protocol Specification
   - AMBA AXI Protocol v2.0 Specification
2. *AXI Protocol Checker LogiCORE™ IP Product Guide* (PG101)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *Vivado Design Suite User Guide: Designing with IP* (UG896)
5. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
7. *ISE to Vivado Design Suite Migration Guide* (UG911)
8. *Vivado Design Suite User Guide: Implementation* (UG904)
9. *SmartConnect v1.0 LogiCORE IP Product Guide* (PG247)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/21/2021	1.1	• Updated IP Facts. • Added Navigating Content by Design Process. • Updated Overview. • Updated Table 2-1. • Updated Register Space. • Updated Figure 3-1. • Updated Figure 4-1. • Added description in Upgrading. • Updated AR link.
05/22/2019	1.0	Added DMA/Bridge Subsystem for PCIe in Introduction.
10/04/2017	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 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.