AXI Protocol Firewall IP v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG293 May 22, 2019

Table of Contents

Introduction

The Xilinx® LogiCORE™ AXI Firewall IP has been developed to protect DMA/Bridge Subsystem for PCIe® from hangs and protocol violations downstream of it that can otherwise lead to host crashes.

Features

- Supports AXI3, AXI4, or AXI4-Lite interface
- Provides protection for the upstream network (SI connection) from failures of the downstream network (MI connection):
    - When a failure is detected, the firewall becomes blocked, preventing further transfers from propagating
    - When the firewall becomes blocked, the SI generates compliant responses for all outstanding and subsequent transactions
    - Separate blocks for read and write transactions
- Firewall blocking trigger conditions:
    - Timeout on expected MI responses
    - Fatal AXI protocol violations observed on MI
    - 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
    - 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	
Provided by Xilinx at the Xilinx Support web page	

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

Overview

The 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 response transfers, and blocks the offending transfer and any further transfers from propagating. This protects the upstream network, allowing it to keep operating in the event of a downstream failure. 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.

Feature Summary

- Supports AXI3, AXI4, or AXI4-Lite interface
- Provides protection for the upstream network (SI connection) from failures of the downstream network (MI connection):
    - When a failure is detected, the firewall becomes blocked, preventing further transfers from propagating.
    - When the firewall becomes blocked, the SI generates compliant responses for all outstanding and subsequent transactions.
    - Separate blocks for read and write transactions.
- Firewall blocking trigger conditions:
    - Timeout on expected MI responses
    - Fatal AXI protocol violations observed on MI
    - 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

- Controls read/write unblock requests (and soft block requests)
- Sets timeout limits

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.

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 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 Fault Status Register.

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

Block Condition

The 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.

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 all-ones **`rdata`** pattern. 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 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).

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 on the SI 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.

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.

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 MI Fault Status Register or responding to an interrupt triggered by the `mi_r_error` or `mi_w_error` output signals.

   Note: The BUSY bits in the Fault Status Register do not indicate that any fault has occurred.

2. 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 Fault Status Register. Do this regardless of whether the write or read traffic remains unblocked.

4. Reset all devices downstream of the firewall MI and release the downstream reset. 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 downstream devices recover from reset.

5. Request firewall unblock by writing to the MI 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.

Table 1-1: **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 don't 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: **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
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.

CAM Overflow

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.

Similarly, the **`NUM_{READ,WRITE}_OUTSTANDING`** parameters determine the size of the outstanding transaction counters to be implemented. 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, 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.

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.

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.

Applications

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

The AXI Firewall IP sits between the DMA controller of the PCIe channel in the static region, and some portion of the programmable region. AXI Firewall IP protects PCIe communications from hangs/protocol violations downstream of it that would otherwise lead to host crashes.

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
ADDR_WIDTH	$1 \le$ integer ≤ 64	32	Width of {s,m}_axi_{ar,aw}addr
ID_WIDTH	$0 \le$ 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	$0 \le$ integer ≤ 1024	0	Width of {s,m}_axi_aruser.
AWUSER_WIDTH	$0 \le$ integer ≤ 1024	0	Width of {s,m}_axi_awuser.
WUSER_WIDTH	$0 \le$ integer ≤ 1024	0	Width of {s,m}_axi_wuser.
RUSER_WIDTH	$0 \le$ integer ≤ 1024	0	Width of {s,m}_axi_ruser.
BUSER_WIDTH	$0 \le$ integer ≤ 1024	0	Width of {s,m}_axi_buser.
PROTOCOL	string = {AXI4, AXI3, AXI4LITE}	AXI4	Used for protocol-specific port widths/enablement.
NUM_READ_THREADS, NUM_WRITE_THREADS	$1 \le$ integer ≤ 16	1	Number of ID threads to implement in protocol check CAMs.

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

Parameter Name	Format/Range	Default Value	Description
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.

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
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

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

Signals	I/O	Default	Width	Description
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
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

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

Signals	I/O	Default	Width	Description
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
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

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

Signals	I/O	Default	Width	Description
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
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

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

Signals	I/O	Default	Width	Description
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)
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 firewall is blocked for writes due to a downstream write fault (sticky until unblocked).
mi_r_error	O	1	Always	Indicates firewall is blocked for reads due to a downstream read fault (sticky until unblocked).

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
0x0	RO	MI Fault Status Register
0x4	WO	MI Soft Fault Control Register
0x8	WO	MI Unblock Control Register
0x30	RW	MAX_CONTINUOUS_RTRANSFERS_WAITS Register
0x34	RW	MAX_WRITE_TO_BVALID_WAITS Register
0x38	RW	MAX_ARREADY_WAITS Register
0x3c	RW	MAX_AWREADY_WAITS Register
0x40	RW	MAX_WREADY_WAITS Register

MI Fault Status Register

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

Table 2-8: **MI Fault Status Register**

Bit(s)	Name	Core Access	Default Value	Description
31:21		RO		All zeros
20	ERRS_BRESP	RO		
19	RECS_WRITE_TO_BVALID_MAX_WAIT	RO		
18	RECS_WREADY_MAX_WAIT	RO		
17	RECS_AWREADY_MAX_WAIT	RO		
16	WRITE_RESPONSE_BUSY	RO		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.)
15:5		RO		All zeros
4	ERRS_RID	RO		
3	ERRS_RDATA_NUM	RO		
2	RECS_CONTINUOUS_RTRANSFERS_ MAX_WAIT	RO		

Table 2-8: **MI Fault Status Register**

Bit(s)	Name	Core Access	Default Value	Description
1	RECS_ARREADY_MAX_WAIT	RO		
0	READ_RESPONSE_BUSY	RO		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 Soft Fault Control Register

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

Table 2-9: **MI Soft Fault Control Register**

Bit(s)	Name	Core Access	Default Value	Description
31:21		WO		Reserved
20:17	Soft write faults	WO		Trigger Write block; copy bit pattern to MI Fault Status Register bits 20-17.
16		WO		Reserved
15:5		WO		Reserved
4:1	Soft read faults	WO		Trigger Read block; copy bit pattern to MI Fault Status Register bits 4-1.
0		WO		Reserved

MI Unblock Control Register

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

Table 2-10: **MI Unblock Control Register**

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

MAX_CONTINUOUS_RTRANSFERS_WAITS Register

The MAX_CONTINUOUS_RTRANSFERS_WAITS register is shown in Table 2-11.

Table 2-11: **MAX_CONTINUOUS_RTRANSFERS_WAITS Register**

Bit(s)	Name	Core Access	Default Value	Description
31:16				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 2-12.

Table 2-12: **MAX_WRITE_TO_BVALID_WAITS Register**

Bit(s)	Name	Core Access	Default Value	Description
31:16				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 2-13.

Table 2-13: **MAX_ARREADY_WAITS Register**

Bit(s)	Name	Core Access	Default Value	Description
31:16				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 2-14.

Table 2-14: **MAX_AWREADY_WAITS Register**

Bit(s)	Name	Core Access	Default Value	Description
31:15				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 2-15.

Table 2-15: **MAX_WREADY_WAITS Register**

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

Designing with the Core

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

General Design Guidelines

The AXI Firewall IP core should be inserted into a system as shown in Figure 3-1.

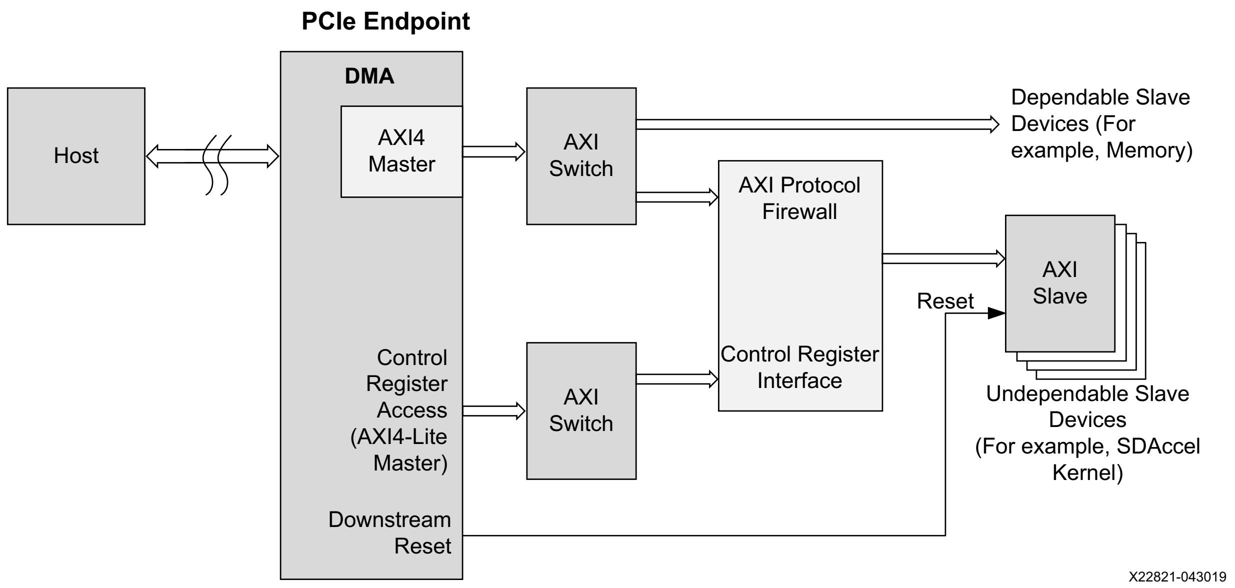

PCIe Endpoint
DMA
Host
AXI4 Master
AXI Switch
Dependable Slave Devices (For example, Memory)
AXI Protocol Firewall
AXI Slave
Reset
Control Register Access (AXI4-Lite Master)
AXI Switch
Control Register Interface
Undependable Slave Devices (For example, SDAccel Kernel)
Downstream Reset
X22821-043019

Figure 3-1: **AXI Firewall IP 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.

Chapter 4

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.

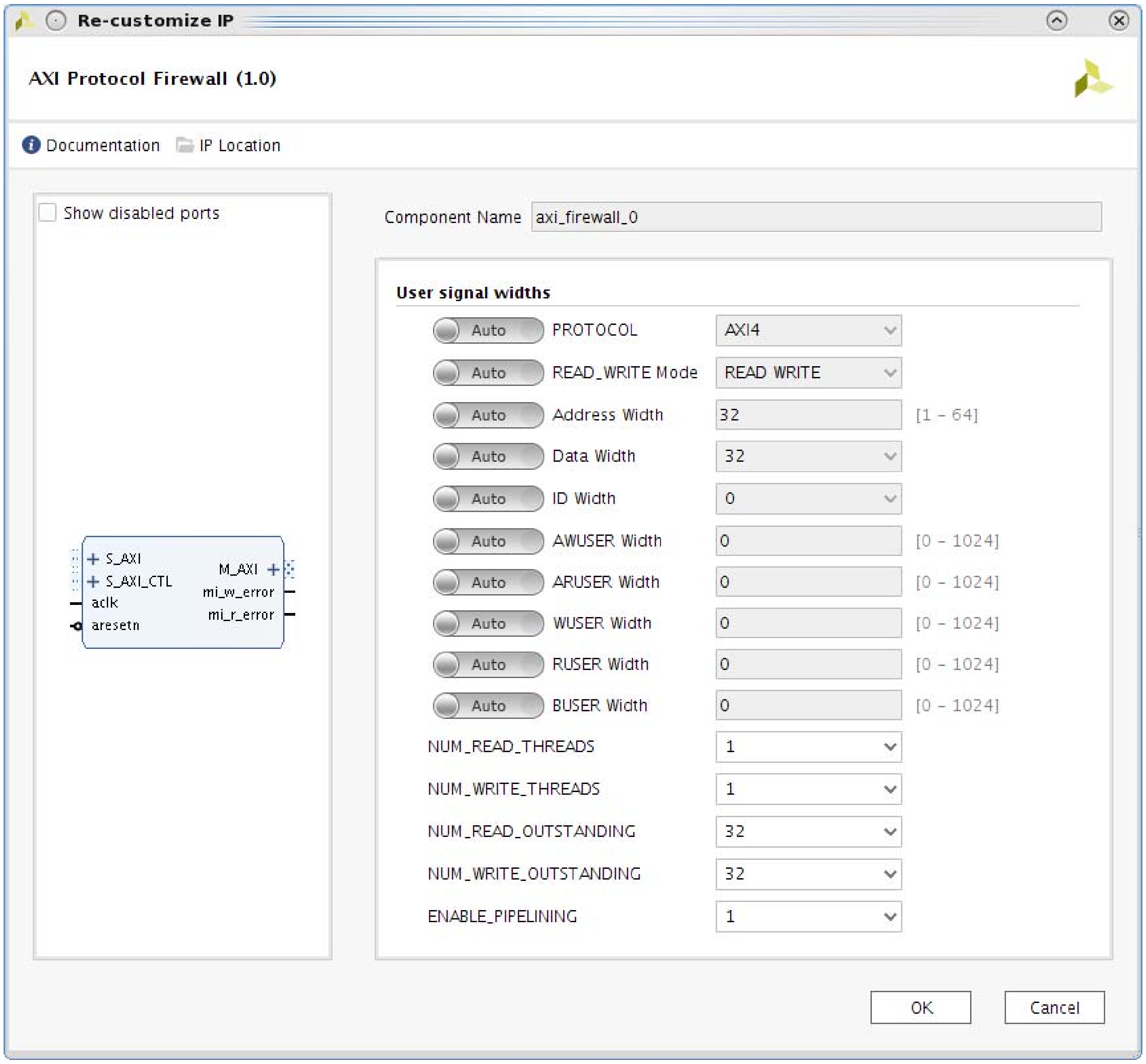
If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. 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.



Component Name axi_firewall_0

User signal widths

Auto	PROTOCOL	AXI4	
Auto	READ_WRITE Mode	READ WRITE	
Auto	Address Width	32	[1 - 64]
Auto	Data Width	32	
Auto	ID Width	0	
Auto	AWUSER Width	0	[0 - 1024]
Auto	ARUSER Width	0	[0 - 1024]
Auto	WUSER Width	0	[0 - 1024]
Auto	RUSER Width	0	[0 - 1024]
Auto	BUSER Width	0	[0 - 1024]
	NUM_READ_THREADS	1	
	NUM_WRITE_THREADS	1	
	NUM_READ_OUTSTANDING	32	
	NUM_WRITE_OUTSTANDING	32	
	ENABLE_PIPELINING	1	

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

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: 68234

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 B

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. *LogiCORE IP AXI Interconnect Product Guide* (PG059)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
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