IP CORE MANUAL

DDR4 SDRAM to PCI Express (PCIe) Direct Memory Access (DMA) IP

px_dma_ddr2pcie

PENTEK

Pentek, Inc.
One Park Way
Upper Saddle River, NJ 07458
(201) 818–5900
http://www.pentek.com/

Manual Part Number: 807.48403 Rev: 1.0 – August 11, 2017

Manual Revision History

Date	Version	Comments
8/11/17	1.0	Initial Release

Legal Notices

Copyright

Trademarks

Printed in the United States of America.

Table of Contents

Page

Table of Contents

Page

Chapter 3: Port Descriptions (continued)

Chapter 4: Register Space

Table of Contents

Page

Chapter 4: Register Space (continued)

Chapter 5: Designing with the Core

Chapter 6: Design Flow Steps

Table of Contents

Page

This page is intentionally blank

IP Facts

Description

Pentek's Navigator™ DDR4 SDRAM to PCI Express® (PCIe®) Memory Direct Memory Access (DMA) IP Core is a DMA engine that facilitates reading of data from the DDR4 SDRAM to the PCIe host by generating read requests to the DDR4 SDRAM and generating write requests to the Xilinx® Gen3 Integrated Block for PCI Express IP Core.

This core complies with the ARM® AMBA® *AXI4 Specification* and also provides a control/status register interface. This manual defines the hardware interface, software interface, and parameterization options for the DDR4 SDRAM to PCIe DMA Core.

Features

- Fully AXI4–compliant interfaces
- Supports generation of DDR4 SDRAM read request AXI4–Streams compatable with the Pentek **`px_axisrq2ddrctlr`** core used by Pentek Jade architectures.
- Compatible with the Xilinx Gen3 Integrated Block for PCI Express IP Core

Table 1–1: IP Facts Table

Core Specifics	
Supported Design Family[a]	Kintex® Ultrascale
Supported User Interfaces	AXI4–Lite and AXI4–Stream
Resources	See Table 2–1
Provided with the Core	
Design Files	VHDL
Example Design	Not Provided
Test Bench	VHDL
Constraints File	Not Provided[b]
Simulation Model	VHDL
Supported S/W Driver	HAL Software Support
Tested Design Flows	
Design Entry	Vivado® Design Suite 2017.1 or later
Simulation	Vivado VSim
Synthesis	Vivado Synthesis
Support	
Provided by Pentek fpgasupport@pentek.com	

a.For a complete list of supported devices, see the *Vivado Design Suite Release Notes*.
b.Clock constraints can be applied at the top level module of the user design.

This page is intentionally blank

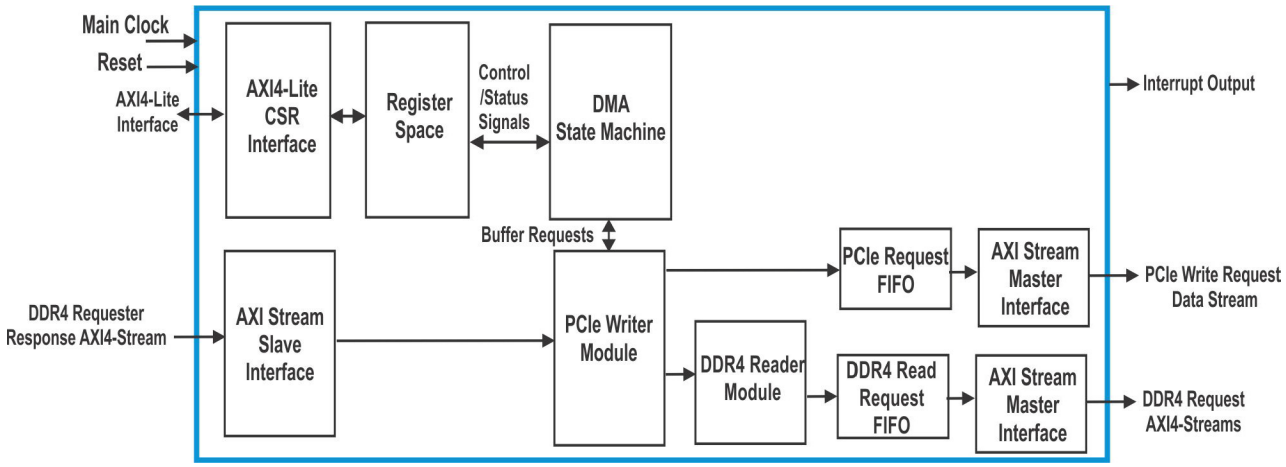
Chapter 1: Overview

1.1 Functional Description

The DDR4 SDRAM to PCIe DMA Core is a high−bandwidth, simple DMA engine used to read data from DDR4 memory and write it to a PCIe host through the Xilinx PCIe Core. This core generates read requests to the DDR4 SDRAM and then accepts the read responses and creates PCIe write requests.

This core generates PCIe write requests which are transferred to the Xilinx PCI Express IP Core through the PCIe Requester Request Interface. An AXI4−Lite Control/ Status Register (CSR) Bus accesses the control/ status registers within the Register Space of the core as shown in Figure 1−1. This core includes a DMA State Machine that is used to generate the buffer write requests based on the address and size of packet values defined by the control registers.

Figure 1−1: DDR4 SDRAM to PCIe DMA Core Block Diagram



1.1 Functional Description (continued)

Figure 1–1 is a top–level block diagram of the DDR4 SDRAM to PCIe DMA Core. The modules within the block diagram are described below.

- **AXI4–Lite Control/Status Register (CSR) Interface:** This module implements a 32–bit AXI4–Lite CSR Slave Interface to access the control/status registers within the Register Space of this core. For additional details about the AXI4–Lite CSR Interface, refer to Section 3.1, AXI4–Lite Core Interfaces.

- **Register Space:** This module contains control and status registers including Interrupt Enable, Interrupt Flag, and Interrupt Status registers. Registers are accessed through the AXI4–Lite Interface.

- **AXI4–Stream Interfaces:** The DDR4 SDRAM to PCIe DMA Core has three AXI4–Stream Interfaces to receive/ transfer AXI4–Streams from the user design. For more details about the AXI4–Stream Interface please refer to **Section 3.2 AXI4–Stream Core Interfaces**.

- **DMA State Machine:** The DMA State Machine is used to generate buffer write requests to the PCIe Writer module based on the source, destination addresses and packet size values defined in the control registers of the core.

- **PCIe Writer Module:** The PCIe Writer module generates the required PCIe Write Request AXI4–Streams from the buffer write requests received from the DMA State Machine. The generated PCIe Requester Request AXI4–Streams follow the standard AXI4–Stream format and must be connected to the Xilinx PCIe Core through a Pentek PCIe Requester Interface Gasket Core in order to convert the format of the **`tready`** and **`tkeep`** AXI4–Stream signals into a format compatible with the Requester Request Interface Bus of the Xilinx PCIe Core.

- **DDR4 SDRAM Reader Module:** This module generates DDR4 SDRAM read request AXI4–Streams from the data streams received from the PCIe Writer module of the core. The DDR4 read request AXI4–Streams are transferred to the user design across the DDR4 Request Interface of the core. Data is recieved back from the DDR4 Response Interface AXI4–Stream.

- **PCIe Request FIFO:** This is an AXI4–Stream FIFO used to store the generated outgoing PCIe write request data streams.

- **DDR4 Request FIFO:** This is an AXI4–Stream FIFO used to store the outgoing DDR4 read request AXI4–Streams.

1.2 Applications

The DDR4 SDRAM to PCIe DMA Core can be incorporated into a Kintex Ultrascale FPGA to perform DMA transfers of data from the DDR4 SDRAM to the Xilinx PCIe Core in the user design.

1.3 System Requirements

For a list of system requirements, see the Vivado Design Suite Release Notes.

1.4 Licensing and Ordering Information

This core is included with all Pentek Navigator FPGA Design Kits for Pentek Jade series board products. Contact Pentek for licensing and ordering information (www.pentek.com).

1.5 Contacting Technical Support

Technical Support for Pentek's Navigator FPGA Design Kits is available via e–mail (fpgasupport@pentek.com) or by phone (201–818–5900 ext. 238, 9 am to 5 pm EST).

1.6 Documentation

This user manual is the main document for this IP core. The following documents provide supplemental material:

1) *Vivado Design Suite User Guide: Designing with IP*

2) *Vivado Design Suite User Guide: Programming and Debugging*

3) *ARM AMBA AXI4 Protocol Version 2.0 Specification*
http://www.arm.com/products/system–ip/amba–specifications.php

4) *Xilinx Gen3 Integrated Block for PCI Express Product Guide*

This page is intentionally blank

Chapter 2: General Product Specifications

2.1 Standards

The DDR4 SDRAM to PCIe DMA Core has bus interfaces that comply with the *ARM AMBA AXI4–Lite Protocol Specification* and the *AMBA AXI4–Stream Protocol Specification*.

2.2 Performance

The performance of the DDR4 SDRAM to PCIe DMA Core is limited by the FPGA logic speed. The values presented in this section should be used as an estimation guideline. Actual performance can vary.

2.2.1 Maximum Frequencies

The DDR4 SDRAM to PCIe DMA Core has two incoming clock signals. The input main clock (**aclk**) has a maximum frequency of 250 MHz, and the DDR4 SDRAM clock has a maximum frequency of 300 MHz on a Kintex Ultrascale –2 speed grade FPGA.

2.3 Resource Utilization

The resource utilization of the DDR4 SDRAM to PCIe DMA Core is shown in Table 2–1. Resources have been estimated for the Kintex Ultrascale XCKU060 –2 speed grade device. These values were generated using the Vivado Design Suite.

Table 2–1: Resource Usage and Availability

Resource	# Used
LUTs	3184
Flip–Flops	8324
Memory LUTs	128
BRAMs	53.5

NOTE: Actual utilization may vary based on the user design in which the DDR4 SDRAM to PCIe DMA Core is incorporated.

2.4 Limitations and Unsupported Features

This section is not applicable to this IP core.

2.5 Generic Parameters

The generic parameters of the DDR4 SDRAM to PCIe DMA Core are described in Table 2–2. These parameters can be set as required by the user application while customizing the core.

Table 2-2: Generic Parameters

Port/Signal Name	Type	Description
pcie_channel	Integer 0–7	**PCIe Channel Number:** This parameter indicates the PCIe tag associated with a PCIe request. This must be unique for each DMA core incorporated into the user design. It can range from 0 to 7. For additional details about the PCIe tag field in the PCIe requester request descriptor of the Xilinx PCIe Core, please refer to Requester Request Descriptor formats in the *Xilinx PCIe Core Product Guide*.
ddr4_requester_id	Integer 0–255	**DDR4 Requester ID:** This parameter indicates the requester ID for the core. The ID is used to route DDR4 responses back to the correct requester in the system when there are multiple requesters. For this reason, each DDR4 requester in the system design must have a unique ID value. It can range from 0 to 255.

Chapter 3: Port Descriptions

This chapter provides details about the port descriptions for the following interface types:

- AXI4–Lite Core Interfaces
- AXI4–Stream Core Interfaces

3.1 AXI4–Lite Core Interfaces

The DDR4 SDRAM to PCIe DMA Core has the following two AXI4–Lite core interfaces to control, and receive status from, the core.

3.1.1 Control/Status Register (CSR) Interface

The CSR interface is an AXI4–Lite Slave Interface that can be used to access the control and status registers in the DDR4 SDRAM to PCIe DMA Core. Table 3–1 defines the ports in the CSR Interface. See Chapter 4 for a Control/Status Register memory map and bit definitions. See the *AMBA AXI4–Lite Specification* for more details on operation of the AXI4–Lite interfaces.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions

Port	Direction	Width	Description
aclk	Input	1	**Clock** (250 MHz)
s_axi_csr_aresetn	Input	1	**Reset:** Active low.
s_axi_csr_awaddr	Input	6	**Write Address:** Address used for write operations. It must be valid when **s_axi_csr_awvalid** is asserted and must be held until **s_axi_csr_awready** is asserted by the DDR4 SDRAM to PCIe DMA Core.
s_axi_csr_awprot	Input	3	**Protection:** The DDR4 SDRAM to PCIe DMA Core ignores these bits.
s_axi_csr_awvalid	Input	1	**Write Address Valid:** This input must be asserted to indicate that a valid write address is available on **s_axi_csr_awaddr**. The DDR4 SDRAM to PCIe DMA Core asserts **s_axi_csr_awready** when it is ready to accept the address. The **s_axi_csr_awvalid** must remain asserted until the rising clock edge after the assertion of **s_axi_csr_awready**.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)

Port	Direction	Width	Description
s_axi_csr_awready	Output	1	**Write Address Ready:** This output is asserted by the DDR4 SDRAM to PCIe DMA Core when it is ready to accept the write address.The address is latched when **s_axi_csr_awvalid** and **s_axi_csr_awready** are high on the same cycle.
s_axi_csr_wdata	Input	32	**Write Data:** This data will be written to the address specified by **s_axi_csr_awaddr** when **s_axi_csr_wvalid** and **s_axi_csr_wready** are both asserted. The value must be valid when **s_axi_csr_wvalid** is asserted and held until **s_axi_csr_wready** is also asserted.
s_axi_csr_wstrb	Input	4	**Write Strobes:** This signal, when asserted, indicates the number of bytes of valid data on the **s_axi_csr_wdata** signal. Each of these bits, when asserted, indicate that the corresponding byte of **s_axi_csr_wdata** contains valid data. Bit 0 corresponds to the least significant byte, and bit 3 to the most significant.
s_axi_csr_wvalid	Input	1	**Write Valid:** This signal must be asserted to indicate that the write data is valid for a write operation. The value on **s_axi_csr_wdata** is written into the register at address **s_axi_csr_awaddr** when **s_axi_csr_wready** and **s_axi_csr_wvalid** are high on the same cycle.
s_axi_csr_wready	Output	1	**Write Ready:** This signal is asserted by the DDR4 SDRAM to PCIe DMA Core when it is ready to accept data. The value on **s_axi_csr_wdata** is written into the register at address **s_axi_csr_awaddr** when **s_axi_csr_wready** and **s_axi_csr_wvalid** are high on the same cycle, assuming that the address has already or simultaneously been submitted.
s_axi_csr_bresp	Output	2	**Write Response:** The DDR4 SDRAM to PCIe DMA Core indicates success or failure of a write transaction through this signal, which is valid when **s_axi_csr_bvalid** is asserted; 00 = Success of normal access 01 = Success of exclusive access 10 = Slave Error 11 = Decode Error Note: For more details about this signal refer to the *AMBA AXI Specification*.
s_axi_csr_bready	Input	1	**Write Response Ready:** This signal must be asserted by the user logic when it is ready to accept the Write Response.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)

Port	Direction	Width	Description
s_axi_csr_bvalid	Output	1	**Write Response Valid:** This signal is asserted by the DDR4 SDRAM to PCIe DMA Core when the write operation is complete and the Write Response is valid. It is held until **s_axi_csr_bready** is asserted by the user logic.
s_axi_csr_araddr	Input	6	**Read Address:** Address used for read operations. It must be valid when **s_axi_csr_arvalid** is asserted and must be held until **s_axi_csr_arready** is asserted by the DDR4 SDRAM to PCIe DMA Core.
s_axi_csr_arprot	Input	3	**Protection:** These bits are ignored by the DDR4 SDRAM to PCIe DMA Core.
s_axi_csr_arvalid	Input	1	**Read Address Valid:** This input must be asserted to indicate that a valid read address is available on **s_axi_csr_araddr**. The core asserts **s_axi_csr_arready** when it ready to accept the Read Address. This input must remain asserted until the rising clock edge after the assertion of **s_axi_csr_arready**.
s_axi_csr_arready	Output	1	**Read Address Ready:** This output is asserted by the DDR4 SDRAM to PCIe DMA Core when it is ready to accept the read address. The address is latched when **s_axi_csr_arvalid** and **s_axi_csr_arready** are high on the same cycle.
s_axi_csr_rdata	Output	32	**Read Data:** This value is the data read from the address specified by the **s_axi_csr_araddr** when **s_axi_csr_arvalid** and **s_axi_csr_arready** are high on the same cycle.
s_axi_csr_rresp	Output	2	**Read Response:** The DDR4 SDRAM to PCIe DMA Core indicates success or failure of a read transaction through this signal, which is valid when **s_axi_csr_rvalid** is asserted; 00 = Success of normal access 01 = Success of exclusive access 10 = Slave Error 11 = Decode Error Note: For more details about this signal refer to the *AMBA AXI Specification*.
s_axi_csr_rvalid	Output	1	**Read Data Valid:** This signal is asserted by the DDR4 SDRAM to PCIe DMA Core when the read is complete and the read data is available on **s_axi_csr_rdata**. It is held until **s_axi_csr_rready** is asserted by the user logic.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)			
Port	**Direction**	**Width**	**Description**
s_axi_csr_rready	Input	1	**Read Data Ready:** This signal is asserted by the user logic when it is ready to accept the Read Data.
irq	Output	1	**Interrupt:** This is an active high, edge–type interrupt output.

3.2 AXI4–Stream Core Interfaces

The DDR4 SDRAM to PCIe DMA Core has the following AXI4–Stream interfaces, used to receive and transfer data streams:

- **PCIe Requester Request (PCIE_RQ) Interface:** This is the interface through which the DDR4 SDRAM to PCIe DMA Core transfers PCIe write requests. This interface is compatible with the Xilinx Gen3 Integrated Block for the PCIe core's Requester Request Interface in address–aligned mode.

- **DDR4 Response (RSP) Interface:** This is the interface through which the responses to read requests are received from the DDR4 SDRAM.

- **PCIe Miscellaneous Control (CNTL) Interface:** This is the interface through which static control signals from the Xilinx PCIe Core are received.

- **DDR4 Request (RQST) Interface:** This core has an DDR4 Memory Request AXI4–Stream Interface at the output of the core to transfer memory read requests to the user design.

3.2.1 PCIe Requester Request (PCIE_RQ) Interface

Table 3–2 defines the ports in the PCIE_RQ Interface of the DDR4 SDRAM to PCIe DMA Core. This is an AXI4–Stream Master Interface that can be used to submit Requester write requests to the Xilinx PCIe Core. The AXI4–Stream PCIe RQ Master Interface is compatible with the Xilinx PCIe Core's Requester Request Interface in 256–bit address–aligned configuration. See the Requester Request Interface Section of *Xilinx Gen3 Integrated Block for PCI Express Product Guide* for more details.

Table 3-2: PCIe Requester Request Interface Port Descriptions

Port	Direction	Width	Description
aclk	Input	1	**Clock:** 250MHz
aresetn	Input		**Reset:** Active Low.
m_axis_pcie_rq_tdata	Output	256	**Requester Request Data Bus:** This is the Requester request data from the DMA core to the Xilinx PCIe Core. It has a fixed width of 256 bits and is therefore only compatible with only the 256–bit wide version of the Xilinx PCIe Core. This data follows address–aligned format.
m_axis_pcie_rq_tlast	Output	1	**TLAST Indication for the Requester Request Data:** The DDR4 SDRAM to PCIe DMA Core asserts this signal in the last cycle of a data transfer to indicate the end of the packet.
m_axis_pcie_rq_tvalid	Output		**Requester Request Data Valid:** This core asserts this signal whenever it is driving valid data on the **m_axis_ pcie_rq_tdata** signal and keeps it asserted during the transfer of a packet. The Xilinx PCIe Core paces the data transfer using the **m_axis_pcie_rq_tready** signal.
m_axis_pcie_rq_tuser	Output	60	**Requester Request User Data:** This signal contains the sideband information for the TLP being transferred. This signal is valid when **m_axis_pcie_rq_tvalid** is High. **Table 3–3** defines the bit definitions of **m_axis_pcie_rq_ tuser[59:0].**

Table 3-2: PCIe Requester Request Interface Port Descriptions (Continued)			
Port	**Direction**	**Width**	**Description**
m_axis_pcie_rq_tkeep	Output	32	**TKEEP Indication for the Requester Request Data:** The assertion of bit *i* of this bus during a transfer indicates that dword *i* (in this case a dword is 8 bits) of the **m_axis_pcie_rq _tdata** bus contains valid data. This bit is set to 1 contiguously for all dwords, starting from the first dword of the descriptor to the last dword of the payload. Thus, **m_axis_pcie_rq_tkeep** is set to all 1s in all beats of a packet, except in the final beat when the total size of the packet is not a multiple of the width of data bus.
m_axis_pcie_rq_tready	Input	1	**Requester Request Ready:** This signal is asserted by the Xilinx PCIe Core to indicate that it is ready to accept data from the DDR4 SDRAM to PCIe DMA Core. Data is transferred across this interface when both **m_axis_ pcie_rq_tready** and **m_axis_pcie_rq _tvalid** are High on the same cycle. If the Xilinx PCIe core deasserts the ready signal when **m_axis _pcie_rq_tvalid** is High, the DMA core maintains the data on the bus and keeps the valid signal asserted until the PCIe core has asserted the ready signal. The standard 1–bit **tready** signal from the DMA core is converted by the Pentek PCIe Requester Interface Gasket core and transferred to the Xilinx PCIe core in the format compatible by it's Requester Request Interface.

3.2 AXI4–Stream Core Interfaces (continued)

3.2.1 PCIe Requester Request (PCIE_RQ) Interface (continued)

Table 3–3 shows the bit definitions of the PCIe Requester Request Interface user data (`m_axis_pcie_rq_tuser`).

Table 3–3: PCIe Requester Request Interface User Data Bit Definitions

Bit Index	Name	Width	Description
59:28	parity	32	**Parity:** Parity is not supported by the DDR4 SDRAM to PCIe DMA Core. The Xilinx PCIe Core must be configured to disable parity checking. These bits are always set to 0x000000.
27:24	seq_num	4	**Sequence Number:** Not supported. Set to 0x0.
23:16	tph_st_tag	8	**Transaction Processing Hint:** This is not supported by this core.
15	tph_indirect_tag_en	1	
14:13	tph_type	2	
12	tph_present	1	
11	discontinue		**Discontinue:** This signal is asserted during a transfer if a DMA abort has been requested. The Xilinx PCIe Core nullifies the corresponding TLP on the PCIe link to avoid data corruption.
10:8	addr_offset	3	**Address Offset:** The DDR4 SDRAM to PCIe DMA Core outputs the dword number where the payload data begins on the data bus through these address offset bits. This enables the Xilinx PCIe Core to determine the alignment of the data block being transferred. The Xilinx PCIe Core samples this field in the first beat of a packet when **m_axis_pcie_rq_tvalid** and **m_axis_pcie_rq_tready** are both asserted.
7:4	last_be	4	**Byte Enables for the Last dword:** The DDR4 SDRAM to PCIe DMA Core only supports data aligned to dword boundaries. Therefore, these four bits always output 0xF.
3:0	first_be		**Byte Enables for the First dword:** The DDR4 SDRAM to PCIe DMA Core only supports data aligned to dword boundaries. Therefore, these four bits always output 0xF.

3.2 AXI4–Stream Core Interfaces (continued)

3.2.2 DDR4 Response (RSP) Interface

The DDR4 SDRAM to PCIe DMA core implements a DDR4 Response Interface to receive read response data from the DDR4 SDRAM. This is an AXI4–Stream Slave Interface.

Table 3–4 defines the ports in the DDR4 Response Interface. See the AMBA AXI4–Stream Specification for more details on the operation of the AXI4–Stream Interface..

Table 3-4: DDR4 Response (RSP) Interface Port Descriptions

Port	Direction	Width	Description
s_axis_rsp_tdata	Input	512	**Response Data Bus:** This is the DDR4 Response data received by the core from the user design.
s_axis_rsp_tlast	Input	1	**Response Data Last:** Tlast indicates the end of a burst of responses and is used by AXI–Stream switches to arbitrate.
s_axis_rsp_tvalid	Input		**Response Data Valid:** Asserted when data is valid on the **s_axis_rsp_tdata** bus.
s_axis_rsp_tuser	Input	256	**Response User Data:** This is the sideband user information data which contains the DDR4 memory response packet header. **Table 3–5** defines the bit definitions of **s_axis_rsp_tuser[255:0].**
s_axis_rsp_tid	Input	8	**Response Data Stream Identifier:** This is the unique data stream identifier**.**

3.2 AXI4–Stream Core Interfaces (continued)

3.2.2 DDR4 Response Interface

Table 3–5 shows the bit definitions of the DDR4 Response Interface user data.

Table 3–5: DDR4 Response Interface User Data Bit Definitions

Bit Index	Name	Width	Description
255:128	DATA	128	**RAM Data:** These bits carry the upper 128 bits of the 640–bit RAM data read from the address location in the DDR4 SDRAM
127:120	RES	9	**Reserved**
119:40	MSK	80	**Byte Mask:** These bits indicate the byte masks of the data. (i.e. the data bytes on the data bus to be masked) This is an echo of the masks sent in the requests and so it marks what data is valid. A '1' indicates that the corresponding data byte is invalid or not needed.
39:36	RES	4	**Reserved**
35	OP	1	**Type of Request:** This bit indicates the memory request type that resulted in this response. Since only read requests generate responses, it is always set to '1.' 0 = Write 1 = Read
34:32	ADDRE	3	**Future Address Expansion**
31:0	ADDR	32	**DDR4 Memory Address:** This is the DDR4 SDRAM start address that the data was read from.

3.2 AXI4–Stream Core Interfaces (continued)

3.2.3 PCIe Miscellaneous Control (CNTL) Interface

This interface is used by the DDR4 SDRAM to PCIe DMA Core to receive PCIe link status data and byte swap data from the user design. **Table 3–6** defines the ports in the **CNTL** interface of the core. This interface is an AXI4–Stream Master Interface and is associated with **aclk** signal.

Table 3–6: PCIe Miscellaneous Control Interface Port Descriptions

Port	Direction	Width	Description
s_axis_cntl_tdata	Input	8	**PCIe DMA Control Data Bus:** This contains the information about the byte swap, PCIe link maximum payload size and maximum read request size. It has a fixed width of 8 bits. **s_axis_cntl_tdata[2:0]** – Maximum PCIe Packet Payload Size 000 – 128 Bytes maximum packet payload size 001 – 256 Bytes maximum packet payload size 010 – 512 Bytes maximum packet payload size 011 – 1024 Bytes maximum packet payload size 100 – 2048 Bytes maximum packet payload size 101 – 4096 Bytes maximum packet payload size **s_axis_cntl_tdata[6:4]** – Maximum PCIe Read Request Size 000 – 128 Bytes maximum read request size 001 – 256 Bytes maximum read request size 010 – 512 Bytes maximum read request size 011 – 1024 Bytes maximum read request size 100 – 2048 Bytes maximum read request size 101 – 4096 Bytes maximum read request size **s_axis_cntl_tdata[7]** – Byte Swap 0 – Not Swapped 1 – Swapped
s_axis_cntl_tvalid		1	**PCIe DMA Control Data Valid:** The core asserts this signal to indicate valid data on the **s_axis_cntl_tdata** signal. This signal is always High.

3.2 AXI4–Stream Core Interfaces (continued)

3.2.4 DDR4 Request (RQST) Interface

The DDR4 SDRAM to PCIe DMA Core implements a DDR4 Request Interface across the output to transfer memory write request data streams. This is an AXI4–Stream Master Interface.

Table 3–7 defines the ports in the DDR4 Request Interface. See the *AMBA AXI4–Stream Specification* for more details on the operation of the AXI4–Stream Interface.

Table 3-7: DDR4 Request Interface Port Descriptions

Port	Direction	Width	Description
m_axis_rqst_tdata	Output	512	**Request Data Bus:** This is the DDR4 Memory Request data sent to the core from the user design. For read requests it is unused and set to zeros.
m_axis_rqst_tvalid		1	**Request Data Valid:** Asserted when data is valid on **m_axis_rqst_tdata** bus. The user application can pace the data transfer using the **m_axis_rqst_tready** signal.
m_axis_rqst_tuser		256	**Request User Data:** This is the sideband user information data which contains the DDR4 memory request packet header. **Table 3–8** includes the bit definitions of the bits **tuser[255:0].**
m_axis_rqst_tlast		1	**Request Data Last:** This is provided mainly to allow arbitration in AXI Stream switches to use it to arbitrate on **tlast**.
m_axis_rqst_tid		8	**Request Data Stream Identifier:** This is the unique data stream identifier.
m_axis_rqst_tready	Input	1	**Request Data Ready:** Activation of this signal by the user design indicates that it is ready to accept data. Data is sent across this interface when both **m_axis_rqst _tvalid** and **m_axis_rqst_tready** are asserted in the same clock cycle.

3.2 AXI4–Stream Core Interfaces (continued)

3.2.4 DDR4 Request (RQST) Interface

Table 3–8 shows the bit definitions of the DDR4 Request Interface user data (**m_axis_rqst_tuser**).

Table 3-8: DDR4 Request Interface User Data Bit Definitions

Bit index	Name	Width	Description
255:128	**DATA**	128	**RAM Data:** These bits are not used for read requests. Set to all zeros.
127:120	**RES**	9	**Reserved**
119:40	**MSK**	80	**Byte Mask:** These bits indicate the byte masks of the data i.e., the data bits on the data bus to be masked. Set byte mask bits to '1' to mark data bytes in the response as invalid data.
39:36	**RES**	4	**Reserved**
35	**OP**	1	**Type of Request:** This bit indicates the type of Memory request.This core always sets this to '1' for Read. 0 = Write 1 = Read
34:32	**ADDRE**	3	**Future Address Expansion:** These bits indicate the address expansion of the DDR4 SDRAM memory location.
31:0	**ADDR**	32	**DDR4 Memory Address:** This is the address location in the DDR4 SDRAM where the read/ write operation is to be performed. The address must be aligned to 64 byte request size boundaries.

Chapter 4: Register Space

This chapter provides the memory map and register descriptions for the register space of the DDR4 SDRAM to PCIe DMA Core. The memory map is provided in Table 4–1.

Table 4–1: Register Space Memory Map

Register Name	Address (Base Address +)	Access	Description
DMA Control	0x00	R/W	Controls the reset of the DMA engine, PCIe address type, and the number of consecutive accesses to the DDR4 SDAM memory.
DMA Advance	0x04		Controls the start of link chain execution after a DMA restart.
DMA Abort	0x08		Controls the abort operation of the DMA.
DMA Transfer Length	0x0C		Controls the DMA transfer length.
FIFO Flush	0x10		Controls the reset of Input FIFO.
PCIe Destination Address (Lower)	0x14		Controls the lower dword of the PCIe destination address [31:0] where a read operation is to be performed.
PCIe Destination Address (Upper)	0x18		Controls the upper dword of the PCIe destination address [63:32] where a read operation is to be performed.
DDR4 SDRAM Source Address	0x1C		Controls the DDR4 SDRAM source address where data is to be read.
DMA Status	0x20	R	Indicates the DMA Status.
Reserved	0x24	N/A	Reserved
	0x28	N/A	
Bytes Last Transferred	0x2C	R	Indicates the number of bytes transferred in the last executed DMA transfer.
Reserved	0x30	N/A	Reserved
Interrupt Enable	0x34	R/W	Interrupt enable bits
Interrupt Status	0x38	R	Interrupt status bits
Interrupt Flag	0x3C	R/Clr	Interrupt flag bits

4.1 DMA Control Register

This register controls the reset of the DMA engine. It resets the DMA State Machine of the core. This register also controls the PCIe address type, and the number of consecutive accesses to the DDR4 SDRAM memory. The DMA Control Register is illustrated in Figure 4–1 and described in Table 4–2. The restart control bit of this register must be toggled once to start execution of DMA after power–up and DMA configuration, or after a DMA abort sequence.

Figure 4–1: DMA Control Register

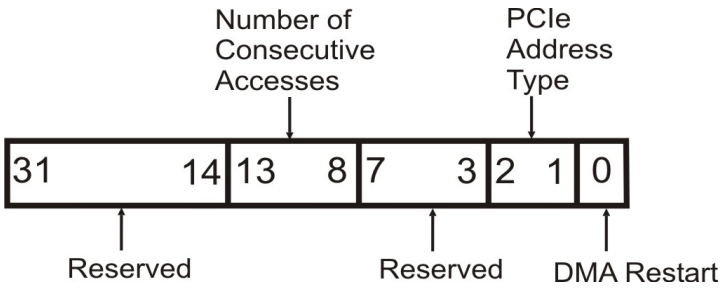


Table 4–2: DMA Control Register (Base Address + 0x00)

Bits	Field Name	Default Value	Access Type	Description
31:14	Reserved	N/A	N/A	**Reserved**
13:8	conseq_ access	0x2F	R/W	**Number of Consecutive Accesses:** These bits control the number of consecutive accesses to the DDR4 SDRAM memory. The optimal value for multi–requester systems for highest bandwidth is 0x2F.
7:3	Reserved	N/A	N/A	**Reserved**
2:1	pcie_at	00	R/W	**PCIe Address Type:** These bits define the address type of the memory transaction. These bits are used by the Xilinx PCIe Core to define the address type in the header of the request TLP. Please refer to the *Xilinx PCIe Core Product Guide* for more details. 00 : Address in the request is untranslated 01 : Transaction is a transaction request 10 : Address in the request is a translated address 11 : Reserved
0	dma_restart	0	R/W	**DMA Restart:** When this bit is toggled ‘1’ then ‘0’, the DMA is reset. This must be done before each DMA execution.

4.2 DMA Advance Register

The DMA Advance Register is used to advance the DMA. The DMA Advance Register control bit must be toggled to start execution of the DMA. This register is illustrated in Table 4–2 and described in Table 4–3.

Figure 4–2: DMA Advance Register

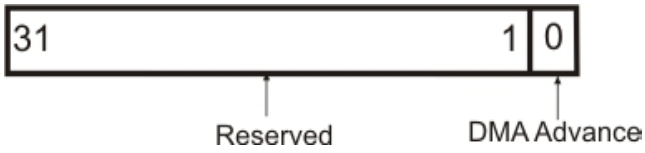


Table 4–3: DMA Advance Register (Base Address + 0x04)

Bits	Field Name	Default Value	Access Type	Description
31:1	Reserved	N/A	N/A	**Reserved**
0	dma_advance	0	R/W	**DMA Advance:** When toggled '1' then '0', it starts the execution of the DMA after reset.

4.3 DMA Abort Register

The DMA Abort Register is used to control the generation of a DMA engine abort sequence. When an abort sequence is enabled, the DMA packets that have already been constructed by the core will be allowed to be transmitted while new DMA activity will be inhibited. When all packets have been transmitted, a DMA reset will automatically be generated. After a DMA abort, the DMA Restart Register control bit must be toggled to start execution of another DMA. This register is illustrated in Figure 4–3 and described in Table 4–4.

Figure 4–3: DMA Abort Register

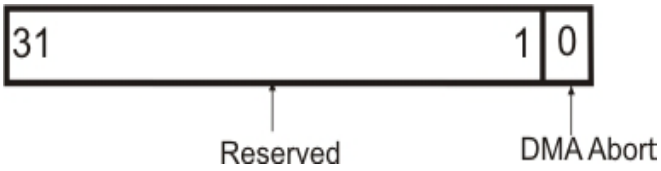


Table 4–4: DMA Abort Register (Base Address + 0x08)

Bits	Field Name	Default Value	Access Type	Description
31:1	Reserved	N/A	N/A	**Reserved**
0	dma_abort	0	R/W	**DMA Abort:** When toggled '1' then '0' a DMA abort sequence will commence.

4.4 DMA Transfer Length Register

The DMA Transfer Length Register is used to control the number of bytes of data to be written to the PCIe address. This register is illustrated in Figure 4–4 and described in Table 4–5.

Figure 4–4: DMA Transfer Length Register

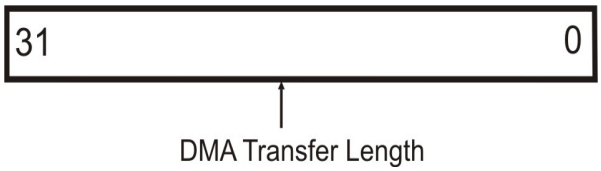


Table 4–5: DMA Transfer Length Register (Base Address + 0x0C)

Bits	Field Name	Default Value	Access Type	Description
31:0	length	0x00000000	R/W	**DMA Transfer Length:** These bits indicate the number of bytes of data to be read from the DDR4 SDRAM and written to the PCIe address.

4.5 FIFO Flush Register

The FIFO Flush Register is used to reset the PCIe Request FIFO and the DDR4 Request FIFO within the DDR4 SDRAM to PCIe DMA Core. Typically, it is a good practice to flush the FIFO before starting the execution of a new DMA transfer in case any extraneous requests or data was left in the FIFO. This register is illustrated in Figure 4–5 and described in Table 4–6.

Figure 4–5: FIFO Flush Register

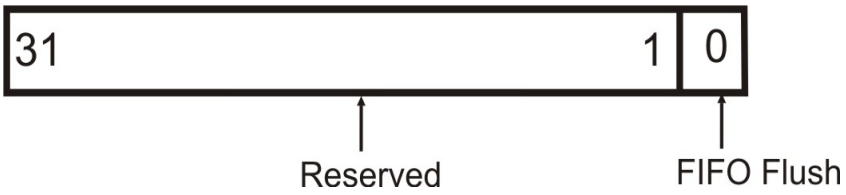


Table 4–6: FIFO Flush Register (Base Address + 0x10)

Bits	Field Name	Default Value	Access Type	Description
31:1	Reserved	N/A	N/A	**Reserved**
0	fifo_flush	0	R/W	**FIFO Flush:** This bit is used to reset the PCIe Request FIFO and DDR4 Request FIFO. 0 = Run 1 = Reset

4.6 PCIe Destination Address (Lower) Register

This register controls the lower dword (32–bits) of the PCIe destination address [31:0] of the PCIe host where a write operation is to be performed. This register is illustrated in Figure 4–5 and described in Table 4–6.

Figure 4–6: PCIe Destination Address (Lower) Register

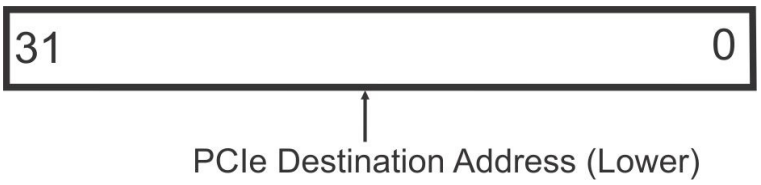


Table 4–7: PCIe Destination Address (Lower) Register (Base Address + 0x14)

Bits	Field Name	Default Value	Access Type	Description
31:0	dest_addr	0x0000 0000	R/W	**PCIe Destination Address:** These bits hold the value of the PCIe address [31:0] where the PCIe write request data is to be sent.

4.7 PCIe Destination Address (Upper) Register

This register controls the upper dword (32–bits) of the PCIe destination address [63:32] of the PCIe host where a write operation is to be performed. This register is illustrated in Figure 4–5 and described in Table 4–6.

Figure 4–7: PCIe Destination Address (Upper) Register

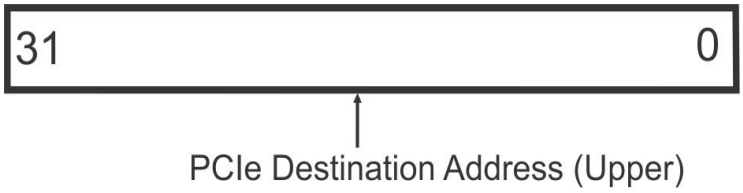


Table 4–8: PCIe Destination Address (Upper) Register (Base Address + 0x18)

Bits	Field Name	Default Value	Access Type	Description
31:0	dest_addr	0x0000 0000	R/W	**PCIe Destination Address:** These bits hold the value of the PCIe address [63:32] where the PCIe write request data is to be sent.

4.8 DDR4 SDRAM Source Address Register

This register controls the DDR4 SDRAM address where the read request data is to be written. This register is illustrated in Figure 4–5 and described in Table 4–6.

Figure 4–8: DDR4 SDRAM Source Address Register

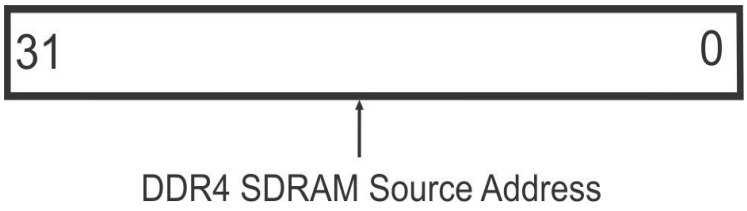


Table 4–9: DDR4 SDRAM Source Address Register (Base Address + 0x1C)

Bits	Field Name	Default Value	Access Type	Description
31:0	src_addr	0x0000 0000	R/W	**DDR4 SDRAM Source Address:** These bits hold the value of the DDR4 address where the write request data is to be written.

4.9 DMA Status Register

The DMA Status Register indicates the status of the DMA engine and the input FIFO. This register is illustrated in Figure 4–9 and described in Table 4–10.

Figure 4–9: DMA Status Register

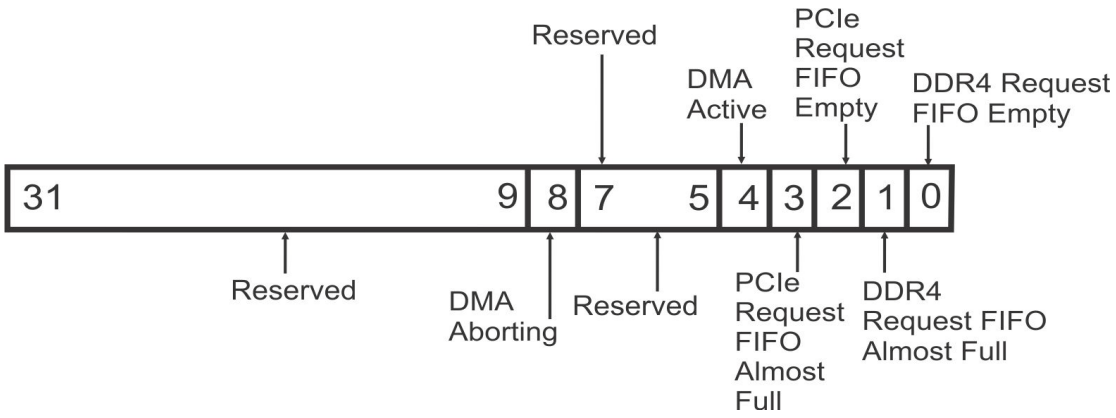


Table 4-10: DMA Status Register (Base Address + 0x20)

Bits	Field Name	Default Value	Access Type	Description
31:9	Reserved	N/A	N/A	**Reserved**
8	dma_aborting	0	R	**DMA Aborting:** When this bit '0', the DMA is not in an abort cycle. When this bit '1', a DMA abort is in progress. 0 = Normal 1 = DMA abort
7:5	Reserved	N/A	N/A	**Reserved**
4	dma_active	0	R/W	**DMA Active:** This bit indicates that the DMA is executing a DMA transfer. 0 = DMA inactive 1 = DMA active

Table 4-10: DMA Status Register (Base Address + 0x20) (Continued)

Bits	Field Name	Default Value	Access Type	Description
3	pcie_rqst_fifo_afl	0	R	**PCIe Request FIFO Almost Full:** This bit indicates that the PCIe request FIFO of the DDR4 SDRAM to PCIe DMA Core is almost full. 0 = FIFO not full 1 = FIFO almost full
2	pcie_rqst_fifo_empty			**PCIe Request FIFO Empty:** This bit indicates that the PCIe request FIFO of the DDR4 SDRAM to PCIe DMA Core is empty. 0 = FIFO not empty 1 = FIFO empty
1	ddr_rqst_fifo_afl			**DDR4 Request FIFO Almost Full:** This bit indicates that the DDR4 Request FIFO is almost full. 0 = FIFO not full 1 = FIFO almost full
0	ddr_rqst_fifo_empty			**DDR4 Request FIFO Empty:** This bit indicates that the DDR4 Request FIFO of the DDR4 SDRAM to PCIe DMA Core is empty. 0 = FIFO not empty 1 = FIFO empty

4.10 Bytes Last Transferred Register

The Bytes Last Transferred Register is a status register which indicates the number of bytes transferred in the last executed descriptor. This register is illustrated in Figure 4–10 and described in Table 4–11.

Figure 4–10: Bytes Last Transferred Register

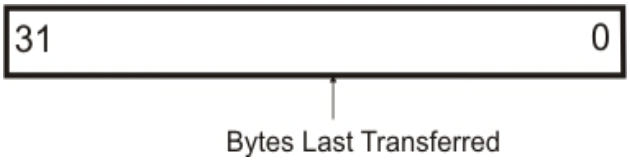


Table 4–11: Bytes Last Transferred Register (Base Address + 0x2C)

Bits	Field Name	Default Value	Access Type	Description
31:0	bytes_last_ transferred	0x00000000	R	**Bytes Last Transferred:** These bits indicate the number of bytes transferred in the last DMA transfer.

4.11 Interrupt Enable Register

The bits in the interrupt enable register are used to enable (or disable) the generation of interrupts based on the condition of certain circuit elements, known as interrupt sources. When a bit in this register associated with a given interrupt source is High, an interrupt will be generated by the rising edge of that source's Interrupt Status Register bit (See Section 4.12). This register is illustrated in Figure 4–11 with the bits described in Table 4–12.

Figure 4–11: Interrupt Enable Register

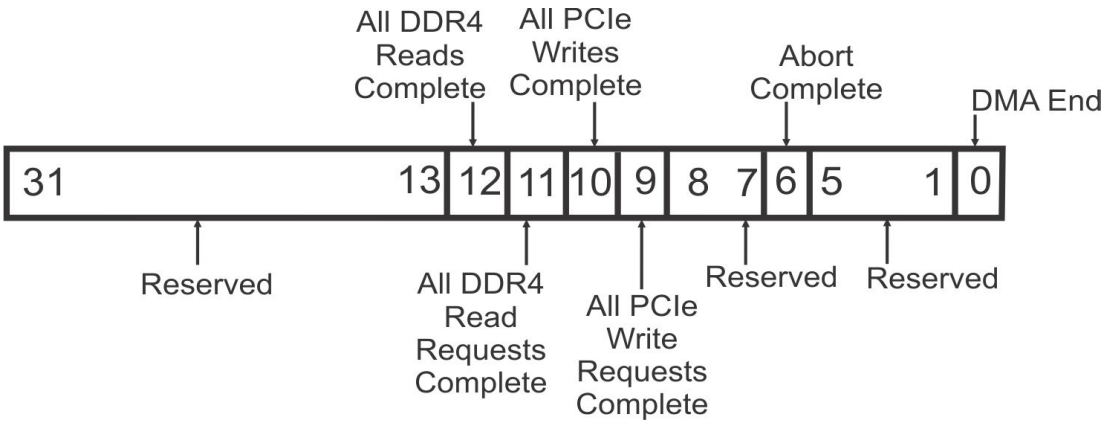


Table 4–12: Interrupt Enable Register (Base Address + 0x34)

Bits	Field Name	Default Value	Access Type	Description
31:13	Reserved	N/A	N/A	**Reserved**
12	ddr_rd_ complete	0	R/W	**All DDR4 Reads Complete:** This bit enables/ disables the all DDR4 reads complete interrupt source. The all DDR4 reads complete interrupt source indicates that all DDR4 read requests have been executed. 0 = Disable interrupt 1 = Enable interrupt
11	ddr_rqsts_ complete	0	R/W	**All DDR4 Read Requests Complete:** This bit enables/ disables the all DDR4 read requests complete interrupt source. The all DDR4 read requests complete interrupt source indicates that all DDR4 read requests have been generated by the DDR4 SDRAM reader module of the DMA core. 0 = Disable interrupt 1 = Enable interrupt

Table 4–12: Interrupt Enable Register (Base Address + 0x34) (Continued)

Bits	Field Name	Default Value	Access Type	Description
10	all_pcie_wr _cmplt	0	R/W	**All PCIe Writes Complete:** This bit enables/ disables the all PCIe writes complete interrupt source. The all writes complete interrupt source indicates that all write requests have been sent to the Xilinx PCIe Core. 0 = Disable interrupt 1 = Enable interrupt
9	all_pcie_rqsts _cmplt	0	R/W	**All PCIe Write Requests Complete:** This bit enables/ disables the all PCIe write requests complete interrupt source. The all write requests complete interrupt source indicates that all write requests to the Xilinx PCIe Core have been transferred. 0 = Disable interrupt 1 = Enable interrupt
8	Reserved	0	N/A	**Reserved**
7	Reserved	N/A	N/A	**Reserved**
6	abort_cmplt	0	R/W	**Abort Complete:** This bit enables/ disables the DMA abort complete interrupt source. 0 = Disable interrupt 1 = Enable interrupt
5:1	Reserved	N/A	N/A	**Reserved**
0	dma_end	0	R/W	**DMA End Interrupt:** This bit enables/ disables the DMA end interrupt source. The link end interrupt source indi–cates that the current DMA transfer is complete. 0 = Disable interrupt 1 = Enable interrupt

4.12 Interrupt Status Register

The Interrupt Status Register has read−only access associated with each interrupt condition. A status bit changes to '1' when the source interrupt occurs. When a status bit in this register changes to '1' the corresponding flag bit in the Interrupt Flag Register is set to '1'. A status bit in this register clears to '0' when that interrupt condition clears, whereas the associated flag bit in the Interrupt Flag Register remains at logic '1' until it is explicitly cleared by the user.

Some of the interrupt sources are transient and so may not appear in the Interrupt Status Register at the time it is read. In such cases use the Interrupt Flag Register to see the interrupt conditions that have occurred. This register is illustrated in Figure 4−11 with the bits described in Table 4−12.

Figure 4−12: Interrupt Status Register

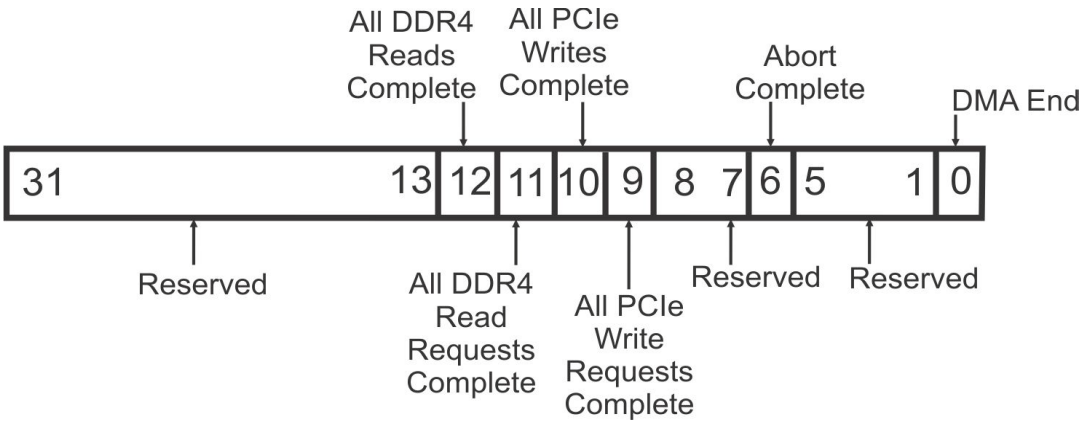


Table 4−13: Interrupt Status Register (Base Address + 0x38)

Bits	Field Name	Default Value	Access Type	Description
31:13	Reserved	N/A	N/A	**Reserved**
12	ddr_rd_ complete	0	R	**All DDR4 Reads Complete:** This bit indicates the status of the all DDR4 reads complete interrupt source. The all DDR4 reads complete interrupt source indicates that all DDR4 read requests have been executed. 0 = No interrupt 1 = Interrupt condition asserted

Table 4–13: Interrupt Status Register (Base Address + 0x38) (Continued)

Bits	Field Name	Default Value	Access Type	Description
11	ddr_rqsts_ complete	0	R	**All DDR4 Read Requests Complete:** This bit indicates the status of the all DDR4 read requests complete interrupt source. The all DDR4 read requests complete interrupt source indicates that all DDR4 read requests have been generated by the DDR4 SDRAM reader module of the DMA core. 0 = No interrupt 1 = Interrupt condition asserted
10	all_pcie_rd _cmplt	0	R	**All PCIe Writes Complete:** This bit indicates the status of the all PCIe writes complete interrupt source. The all writes complete interrupt source indicates that all write requests have been sent to the Xilinx PCIe Core. 0 = No interrupt 1 = Interrupt condition asserted
9	all_pcie_rqsts _cmplt	0	R	**All PCIe Read Requests Complete:** This bit indicates the status of the all PCIe read requests complete interrupt source. The all read requests complete interrupt source indicates that all read requests to the Xilinx PCIe Core have been transferred. 0 = No interrupt 1 = Interrupt condition asserted
8	Reserved	0	N/A	**Reserved**
7	Reserved	N/A	N/A	**Reserved**
6	abort_cmplt	0	R	**Abort Complete:** This bit indicates the status of the DMA abort complete interrupt source. 0 = No interrupt 1 = Interrupt condition asserted
5:1	Reserved	N/A	N/A	**Reserved**
0	dma_end	0	R	**DMA End Interrupt:** This bit indicates the status of the DMA end interrupt source. The link end interrupt source indicates that the current DMA transfer is complete. 0 = No interrupt 1 = Interrupt condition asserted

4.13 Interrupt Flag Register

The Interrupt Flag Register has read/clear access associated with each interrupt condition. When reset, this register has all bits set to '0' (cleared). Each flag bit in this register latches an interrupt occurrence. A '1' in any flag bit in this register indicates that an interrupt has occurred.

Note that when any status bit in the Interrupt Status Register, changes to '1' the corresponding flag bit in this register will also be set to '1'. However, when a status bit in the Interrupt Status Register clears from '1' to '0', the corresponding latched flag bit in this register does not clear, but remains at '1'. To clear the flag bits, write '1's to the desired bits. The flags are not affected by the Interrupt Enable Register. This register is illustrated in Figure 4–11 with the bits described in Table 4–12.

Figure 4–13: Interrupt Flag Register

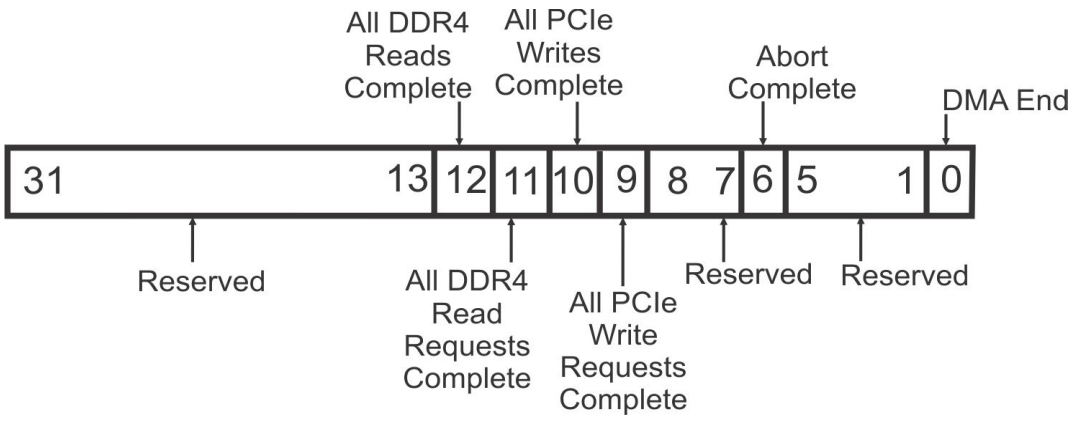


Table 4–14: Interrupt Flag Register (Base Address + 0x3C)

Bits	Field Name	Default Value	Access Type	Description
31:13	Reserved	N/A	N/A	**Reserved**
12	ddr_rd_ complete	0	R/Clr	**All DDR4 Reads Complete:** This bit indicates the all DDR4 reads complete interrupt flag. The all DDR4 reads complete interrupt source indicates that all DDR4 read requests have been executed. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

Table 4–14: Interrupt Flag Register (Base Address + 0x3C) (Continued)

Bits	Field Name	Default Value	Access Type	Description
11	ddr_rqsts_ complete	0	R/Clr	**All DDR4 Read Requests Complete:** This bit indicates the all DDR4 read requests complete interrupt flag. The all DDR4 read requests complete interrupt source indicates that all DDR4 read requests have been generated by the DDR4 SDRAM reader module of the DMA core. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
10	all_pcie_rd _cmplt	0	R/Clr	**All PCIe Writes Complete:** This bit indicates the all PCIe writes complete interrupt flag. The all writes complete interrupt source indicates that all write requests have been sent to the Xilinx PCIe Core. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
9	all_pcie_rqsts _cmplt	0	R/Clr	**All PCIe Write Requests Complete:** This bit indicates the all PCIe write requests complete interrupt flag. The all write requests complete interrupt source indicates that all write requests to the Xilinx PCIe Core have been transferred. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
8	Reserved	0	N/A	**Reserved**
7	Reserved	N/A	N/A	**Reserved**
6	abort_cmplt	0	R/Clr	**Abort Complete:** This bit indicates the DMA abort complete interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
5:1	Reserved	N/A	N/A	**Reserved**
0	dma_end	0	R/Clr	**DMA End Interrupt:** This bit indicates the DMA end inter–rupt flag. The link end interrupt source indicates that the current DMA transfer is complete. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

Chapter 5: Designing with the Core

This chapter provides guidelines and additional information to facilitate designing with the DDR4 SDRAM to PCIe DMA Core.

5.1 General Design Guidelines

The DDR4 SDRAM to PCIe DMA Core provides the required logic to generate PCIe write requests to the Xilinx PCIe Core which interacts with the PCIe host. The core generates DDR4 SDRAM read requests to read the SDRAM so that the data can be written to the PCIe host. The user can control the DMA operation by setting the desired values to the control registers in the Register Space as described in Chapter 4.

5.2 Clocking

Main Clock: **aclk**

This clock is used to clock all the ports on the DDR4 SDRAM to PCIe DMA Core except the DDR4 SDRAM Interfaces.

DDR4 SDRAM Clock: **axis_ddr_aclk**

This is used to clock the DDR4 SDRAM Interface of the DMA Core. (RQST and RSP)

5.3 Resets

Main reset: **aresetn**

This is an active low synchronous reset associated with **aclk**. When this reset is asserted, all state machines in the core are reset and FIFOs are flushed. Caution should be exercised that this reset is not asserted while the DMA is running, in order to avoid generation of incomplete or malformed packets at the PCIe Requestor Request Inter–face. When possible, an abort sequence should be used instead when the user design needs to stop the DMA operation.

CSR Reset: **s_axi_csr_aresetn**

This is an active low synchronous reset associated with **aclk**. When asserted, this reset will clear all control registers back to their initialized default states. It does not reset any of the state machines or flush any of the FIFOs in the core.

5.4 Interrupts

This core has an edge−type (rising edge−triggered) interrupt output. It is synchronous with the **aclk**. On the rising edge of any interrupt signal, a one clock cycle wide pulse is output from the core on its **irq** output. Each interrupt event is stored in two registers, accessible on the **s_axi_csr** bus. The Interrupt Status Register always reflects the cur−rent state of the interrupt condition, which may have changed since the generation of the interrupt. The Interrupt Flag Register latches the occurrence of each interrupt, in a bit that retains its state until explicitly cleared.The interrupt flags can be cleared by writing '1' to the associated bit's location. All interrupt sources that are enabled (via the Interrupt Enable Register) are "OR ed" onto the **irq** output.

NOTE: All interrupt sources are latched in the Interrupt Flag Register, even when an interrupt source is not enabled to create an interrupt.

NOTE: Because this core uses edge−triggered interrupts, the fact that an interrupt condition may remain active after servicing will not cause the generation of a new interrupt. A new interrupt will only be generated by another rising edge on an interrupt source.

5.5 Interface Operation

- **CSR Interface:** This is the control/status register interface. It is associated with **aclk**. It is a standard AXI4−Lite Slave interface. Typically, this interface is connected along with other cores' AXI4−Lite interfaces through an AXI Lite Crossbar core or a series of AXI Lite Crossbar cores that route AXI Lite accesses through to the desired core based on the address range.

- **PCIe Requester Request Interface:** This is the PCIe Requester Request Interface which is associated with **aclk** and is used to transfer PCIe write requests from the DDR4 SDRAM to PCIe DMA Core to the Xilinx PCIe Core. This is a standard AXI4−Stream Master interface which is compatible with the Xilinx PCIe Core's Requester Request Bus when it is set up to be 256 bits wide and operating in address−aligned mode.

  Typically, this interface is connected along with other DMA cores through an AXI4−Stream Switch Core that arbitrates multiple input streams into one output stream, to the Xilinx PCIe Core's Requester Request Bus.

  This interface must be connected to Xilinx PCIe Core through a Pentek PCIe Requester Interface Gasket Core in order to convert the standard **tkeep** and **tready** signals of the PCIe request from the DDR4 SDRAM DMA Core into format compati−ble with the Xilinx PCIe Core's Requester Request Bus signals. For more details about this interface, refer to Section 3.2.1.

- **DDR4 Request (RQST) Interface:** This is an AXI4−Stream Master Interface used to transfer DDR4 read request data streams to the user design, and is associated with **axis_ddr_aclk**. For more details about this interface, refer to **Section 3.2.4**.

5.5 Interface Operation (continued)

- **DDR Response (RSP) Interface:** This is an AXI4−Stream Slave Interface used to receive DDR4 read responses and is associated with **axis_ddr_aclk**.

- **PCIe Miscellaneous Control Interface:** This interface is associated with **aclk** and is used to receive static control signals from the Xilinx PCIe Core. For more details about this interface, refer to Section 3.2.3.

5.6 Programming Sequence

This section briefly describes the programming sequence for the DDR4 SDRAM to PCIe DMA Core.

1) Set the control registers with the required values.

2) Enable DMA interrupts.

3) Reset the DMA.

4) Start the DMA.

5) When done, check the interrupt flag register and clear the interrupts.

5.7 Timing Diagrams

The timing diagram for the DDR4 SDRAM to PCIe DMA Core is shown in Figure 6−3. This timing diagram is obtained by running the simulation of the test bench of the core in Vivado VSim environment. For more details about the test bench, refer to Section 6.5.

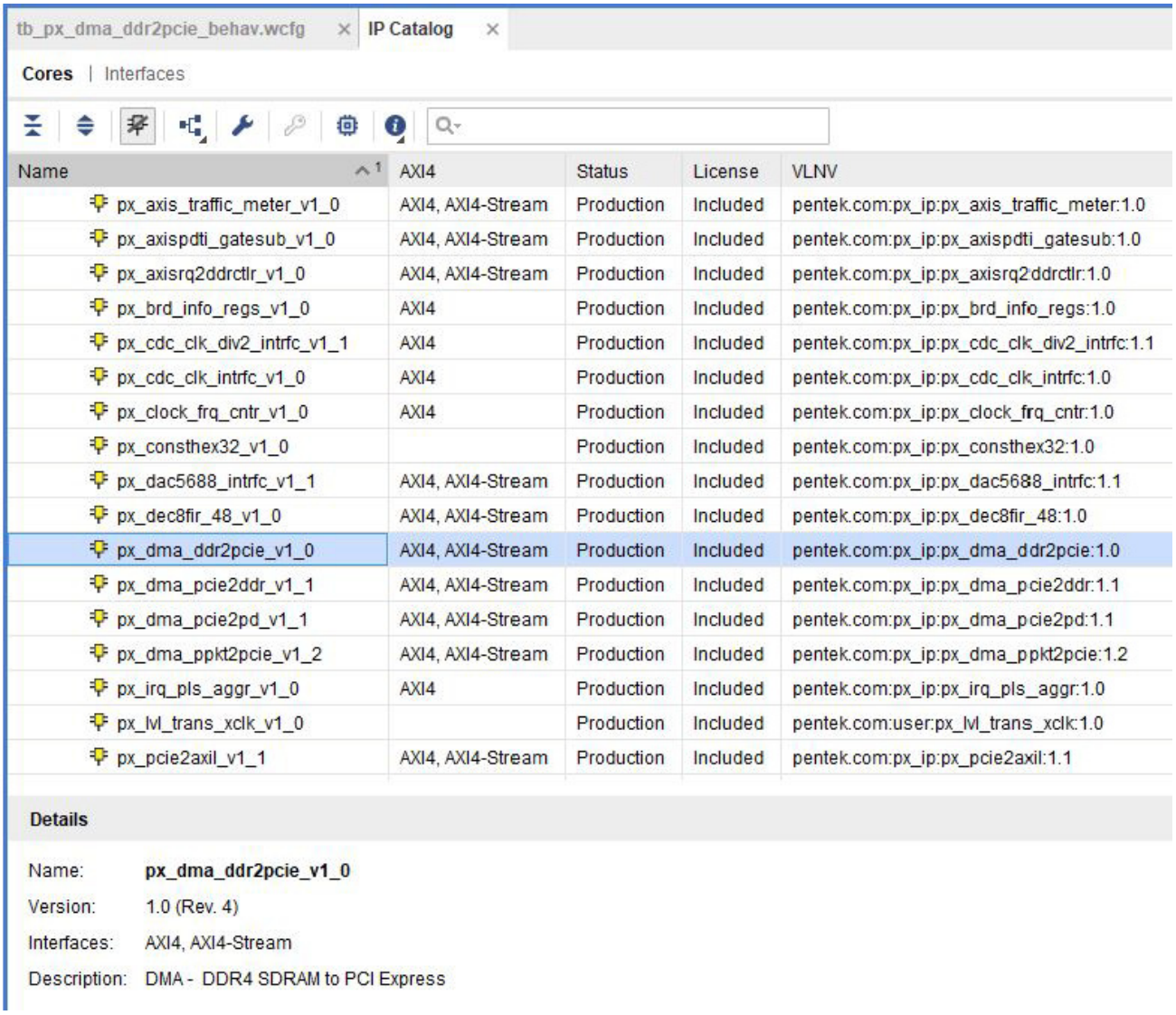
This page is intentionally blank

Chapter 6: Design Flow Steps

6.1 Pentek IP Catalog

This chapter describes customization and generation of the Pentek DDR4 SDRAM to PCIe DMA Core. It also includes simulation, synthesis, and implementation steps that are specific to this IP core. This core can be generated from the Vivado IP Catalog when the Pentek IP Repository has been installed. It will appear in the IP Catalog list as **px_dma_ddr2pcie_v1_0** as shown in Figure 6–1.

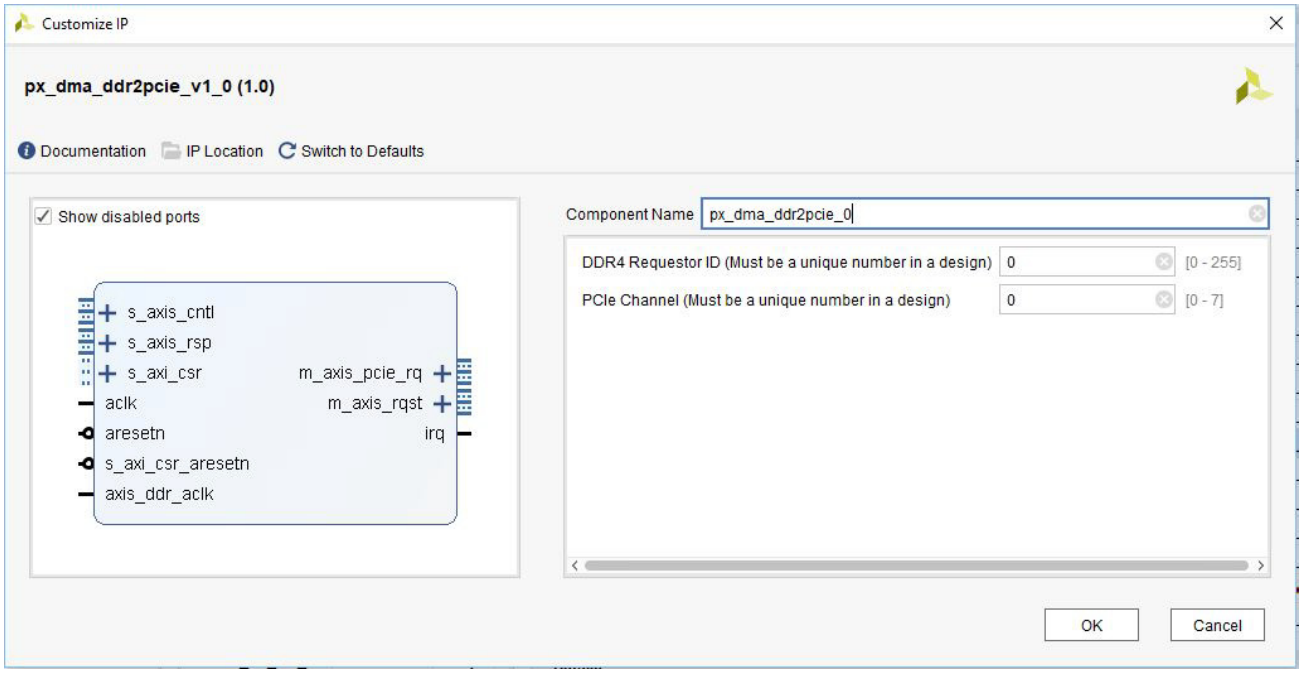
Figure 6–1: DDR4 SDRAM to PCIe DMA Core in Pentek IP Catalog



6.1 Pentek IP Catalog (continued)

When you select the **px_dma_ddr2pcie_v1_0** core, a screen appears that shows the core's symbol and the core's parameters (see Figure 6–2). The core's symbol is the box on the left side.

Figure 6–2: DDR4 SDRAM to PCIe DMA Core IP Symbol



6.2 User Parameters

The user parameters of this DDR4 SDRAM to PCIe DMA Core are explained in Section 2.5 of this user manual.

6.3 Output Generation

For more details about generating and using IP in the Vivado Design Suite, refer to the *Vivado Design Suite User Guide – Designing with IP*.

6.4 Constraining the Core

This section contains information about constraining the DDR4 SDRAM to PCIe DMA Core in the Vivado Design Suite.

Required Constraints

The XDC constraints are not provided with the DDR4 SDRAM to PCIe DMA Core. Clock constraints can be applied in the top−level module of the user design.

Device, Package, and Speed Grade Selections

This IP works for the Kintex Ultrascale FPGAs.

Clock Frequencies

The main clock (**aclk**) of the DDR4 SDRAM to PCIe DMA Core has maximum fre−quency of 250 MHz. The DDR4 SDRAM clock (**ais_ddr_aclk**) has a maximum fre−quency of 300 MHz.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking and Placement

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

6.5 Simulation

The DDR4 SDRAM to PCIe DMA Core has a test bench which generates output wave–forms using the Vivado VSim environment. This test bench is designed to run at 250 MHz main clock frequency (**aclk**) and 300 MHz DDR4 SDRAM clock (**axis_ddr_aclk**) frequency.

The control registers in the Register Space of the DMA Core are written to based on the values defined in a **test_parameters.txt** file. The DMA is made operational by toggling the control bit of the DMA Advance control register. The programming sequence of the DMA is the same as described in Section 5.6.

The contents of the **test_parameters.txt** file along with descriptions of the parameters are provided in Table 6–1..

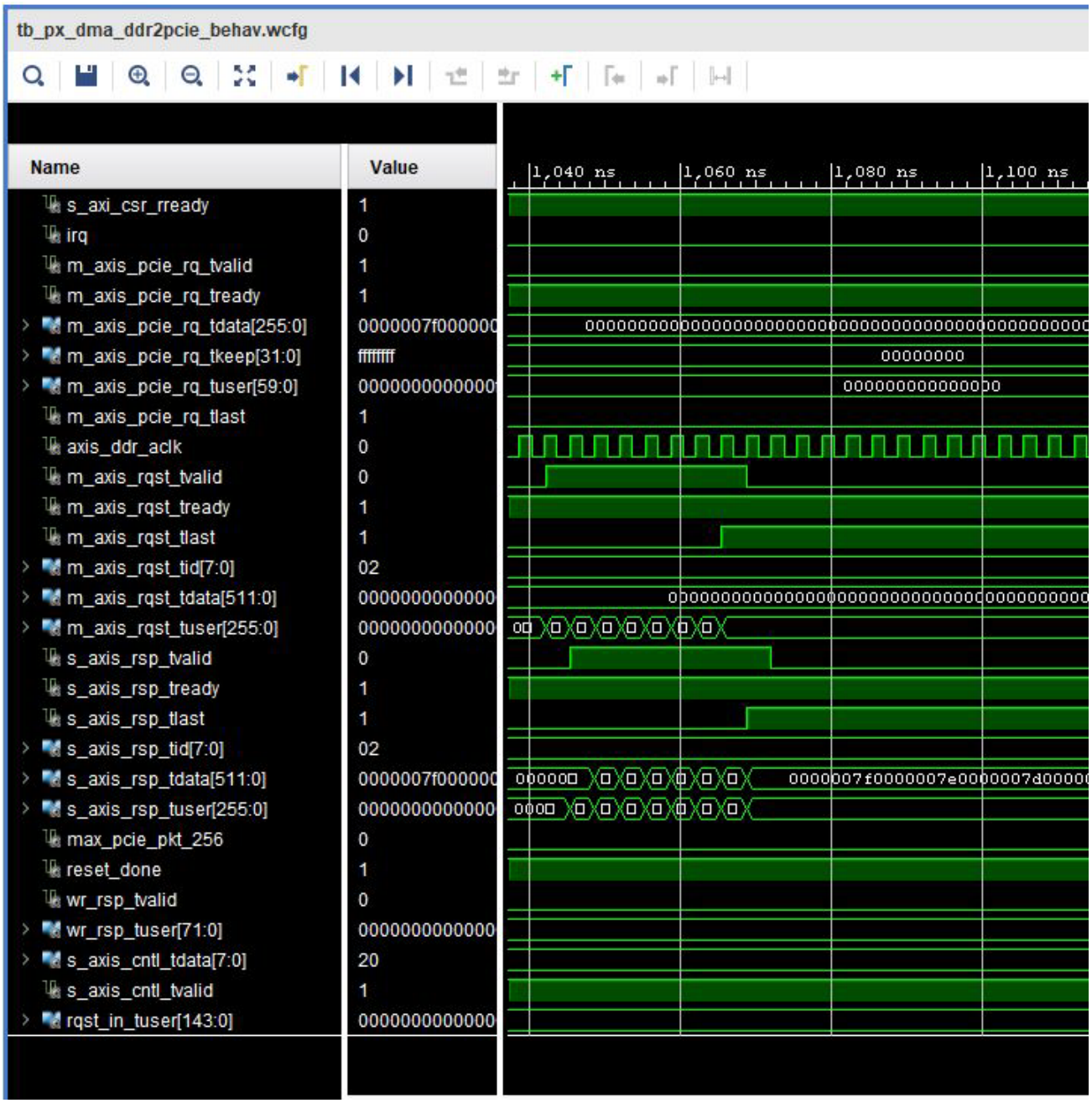
Table 6-1: Test Parameters File Contents and Parameter Descriptions

Parameter	Type	Value	Description
pcie_rqst_size	std_logic_ vector	2	**Maximum PCIe Write Request Size:** This value is determined by the PCIe host. 000 – 128 Bytes maximum packet payload size 001 – 256 Bytes maximum packet payload size 010 – 512 Bytes maximum packet payload size 011 – 1024 Bytes maximum packet payload size 100 – 2048 Bytes maximum packet payload size 101 – 4096 Bytes maximum packet payload size
pcie_address	std_logic_ vector	0x00000000 C0000000	**PCIe Destination Address:** This is the PCIe address where a data write operation is to be performed.
ddr4_address	std_logic_ vector	0xD0000000	**DDR4 SDRAM Source Address:** This is the DDR4 SDRAM source address where the data is to be read.
buffera_bytesize	Integer	516	**Bytes to Transfer:** This parameter indicates the number of bytes to be written to the PCIe during the execution of the DMA transfer.

When the test bench is run, the simulation produces the results shown in Figure 6–3. The generated PCIe write request data streams generated are stored in **test_results.txt** file.

6.5 **Simulation** (continued)

Figure 6–3: DDR4 SDRAM to PCIe DMA Core Test Bench Simulation Output



6.6 Synthesis and Implementation

For details about synthesis and implementation see the *Vivado Design Suite User Guide – Designing with IP*.