altera
An Intel Company

GTS AXI Multichannel DMA IP for PCI Express User Guide

Updated for Quartus® Prime Design Suite: **25.1**

IP Version: **1.0.0**

Online Version **847470**
Send Feedback **2025.05.06**

Contents

1. Overview 5
1.1. About the GTS AXI Multichannel DMA IP for PCI Express* 5
1.1.1. IP Release Information 5
1.1.2. IP Features 6
1.1.3. IP High-level Block Diagram 8
1.1.4. Licensing the IP 8
1.1.5. Device Family Support 8
1.1.6. Device Speed Grade Support 9
1.1.7. Resource Utilization 9
1.1.8. IP and Design Example Support 10
1.2. About the Document 10
1.2.1. Target Audience 10
1.2.2. Conventions Used in the Document 11
1.2.3. Terminology 11
1.2.4. Acronyms 11
1.2.5. Reference Documents and Training 12
1.3. Applications 12
1.4. Key Concepts 12
1.4.1. PCI Express and DMA 12
1.4.2. Altera FPGA IPs for PCI Express and DMA Controller 13
1.4.3. Example Use Case 13
1.5. IP Design Flow 15
1.6. Design Guidelines 16
1.6.1. Supporting IPs 16
1.6.2. Aligning IP Settings with the GTS AXI Streaming IP for PCI Express 16
1.7. Design Flow Requirements 18
1.7.1. Software Requirements 18
1.7.2. Hardware Requirements 18
1.7.3. Supported EDA Tools 19

2. Quick Start Guide 20
2.1. Step-by-step IP Bring-up 20
2.1.1. Downloading and Installing Quartus Prime Software 20
2.1.2. Configuring and Generating the GTS AXI Multichannel DMA IP for PCI Express 20
2.1.3. Configuring and Generating the GTS AXI Streaming Intel FPGA IP for PCI Express 21
2.1.4. Configuring and Generating the GTS System PLL Clocks Intel FPGA IP 22
2.1.5. Configuring and Generating the GTS Reset Sequencer Intel FPGA IP 23
2.1.6. Configuring and Generating the Reset Release IP 24
2.1.7. Instantiating and Connecting the IP Interfaces 25
2.1.8. Simulate, Compile and Validate the Design on Hardware 26
2.2. Reset Sequence 27
2.2.1. Cold Reset Sequence Triggered by PERST# Signal 27
2.2.2. Warm Reset Sequence Triggered by Hot Reset 28
2.2.3. Cold Reset Triggered by User 28
2.2.4. Warm Reset Triggered by User 28

3. Configuring and Generating the GTS AXI Multichannel DMA IP for PCI Express..... 29
3.1. Creating the Quartus Prime Project..... 29
3.2. Configuring the GTS AXI Multichannel DMA IP for PCI Express..... 29
3.2.1. IP Settings..... 31
3.2.2. PCIe Settings..... 33
3.3. Generating HDL for Simulation and Synthesis..... 46
3.3.1. IP Core Generation Output..... 47
3.4. Generating the Design Example..... 49
3.5. Compiling the Design Example..... 53

4. Integrating the IP With Your Application..... 55
4.1. Overview..... 55
4.2. Implementing Required Clocking..... 57
4.3. Implementing Required Resets..... 57
4.4. Connecting the GTS AXI Streaming IP-Facing Interfaces..... 58
4.4.1. PCIe AXI-Stream TX Interface (ss_tx_st)..... 58
4.4.2. PCIe AXI-Stream RX Interface (ss_rx_st)..... 59
4.4.3. Control and Status Register Interface (ss_csr_lite)..... 59
4.4.4. Transmit Flow Control Credit Interface (ss_txcrdt)..... 60
4.4.5. Configuration Intercept Interface (CII)..... 61
4.4.6. Completion Timeout Interface (ss_cplto)..... 62
4.4.7. Function Level Reset (FLR) Interface..... 63
4.4.8. Control Shadow Interface (ss_ctrlshadow)..... 64
4.4.9. Error Interface..... 65
4.5. Connect User Logic-Facing Interfaces..... 66
4.5.1. H2D AXI-Stream Manager (h2d_st_initatr)..... 66
4.5.2. D2H AXI-Stream Subordinate (d2h_st_respndr)..... 67
4.5.3. H2D/D2H AXI-MM Manager (dma_mm_initatr)..... 67
4.5.4. BAM AXI-MM Manager (bam_mm_initatr)..... 69
4.5.5. BAS AXI-MM Subordinate (bas_mm_respndr)..... 72
4.5.6. PIO AXI-Lite Manager (pio_lite_initiatr)..... 74
4.5.7. HIP Reconfiguration AXI-Lite Subordinate (user_csr_lite)..... 75
4.5.8. User Event MSI-X (user_msix)..... 76
4.5.9. User Event MSI (user_msi)..... 77
4.5.10. User Function Level Reset (user_flr)..... 77
4.5.11. User Configuration Intercept Interface..... 78
4.5.12. Configuration Slave (cs_lite_respndr)..... 79

5. Simulating the IP..... 81
5.1. Design Example Overview..... 81
5.1.1. AXI-S Device-side Packet Loopback Design Example..... 81
5.2. Design Example Components..... 83
5.3. Simulating the Design Example..... 84

6. Validating the IP..... 85
6.1. Hardware and Software Requirements..... 87
6.2. Running the Design Example Application on a Hardware Setup..... 87
6.2.1. Program the FPGA..... 88
6.2.2. Configuration Changes from BIOS..... 89
6.2.3. Host Operating System Check (if Working with Ubuntu 22.04)..... 90
6.2.4. Installing the Required Kernel Version (if Working with Ubuntu 24.04)..... 90

6.2.5. Set the Boot Parameters....91
6.2.6. Custom Driver....93
6.2.7. DPDK Poll Mode Driver....97

A. Appendix A: Functional Description.... 102
A.1. High-level Functional Overview.... 102
A.1.1. Multichannel DMA.... 103
A.1.2. Bursting Master (BAM)....112
A.1.3. Bursting Slave (BAS)....113
A.1.4. MSI Interrupt.... 113
A.1.5. Configuration Slave (CS)....115
A.1.6. Root Port Address Translation Table Enablement....116
A.1.7. Control and Status Register Interface.... 118
A.1.8. Configuration Intercept Interface....119

B. Appendix B: Registers....120
B.1. Queue Control.... 121
B.2. MSI-X Memory Space.... 125
B.3. Control Register (GCSR).... 126

C. Document Revision History for the GTS AXI Multichannel DMA IP for PCI Express*... 128

1. Overview

The GTS AXI Multichannel DMA IP for PCI Express (GTS AXI MCDMA IP) allows you to implement Multichannel DMA using the industry standard AXI interface protocol. The GTS AXI MCDMA IP interfaces with the GTS AXI Streaming IP for PCI Express (GTS AXI Streaming IP). It facilitates streamlined and high-performance data transfers by furnishing independent DMA channels that operate seamlessly over the underlying PCIe link connecting the host and the device.

This document introduces the various Altera FPGA PCI Express and DMA IP offerings and details of GTS AXI MCDMA IP, including features and functional descriptions of the various blocks within the IP. This document also describes the design flow requirements, IP parameters, interfaces, and signals available to you when you use this IP.

1.1. About the GTS AXI Multichannel DMA IP for PCI Express*

1.1.1. IP Release Information

IP versions are the same as the Quartus® Prime Design Suite software versions up to v19.1. From Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 1. IP Release Information

Item	Description
IP Version	1.0.0
Quartus Prime Version	Quartus Prime Pro Edition Software v25.1
Release Date	2025.04.07

1.1.2. IP Features

GTS AXI Multichannel DMA IP Features

- Interfaces with GTS AXI Streaming Intel FPGA IP for PCI Express*.
- Supports PCIe 3.0/PCIe 4.0 1x4 Root Port and Endpoint modes.
- AXI Streaming/AXI Memory-Mapped user interface bandwidths: 128/256-bit @ up to 350 MHz.
- 10-bit tag and byte-aligned transfer are not supported.

1.1.2.1. Root Port Mode Features

- Address Translation Table (ATT) support in BAS and BAM + BAS modes
- Automatically enables Configuration Slave interface
- Completion reordering
- Completion timeout
- Error interface
- User mode options and supported user interfaces
  - Bursting Master (BAM)
    - BAM AXI4 Manager interface
    - Configuration Slave interface
    - Hard IP Reconfiguration interface
  - Bursting Slave (BAS)
    - BAS AXI4 Subordinate interface
    - Configuration Slave interface
    - Hard IP Reconfiguration interface
  - BAM + BAS
    - BAM AXI-MM Manager interface
    - BAS AXI-MM Subordinate interface
    - Configuration Slave interface
    - Hard IP Reconfiguration interface

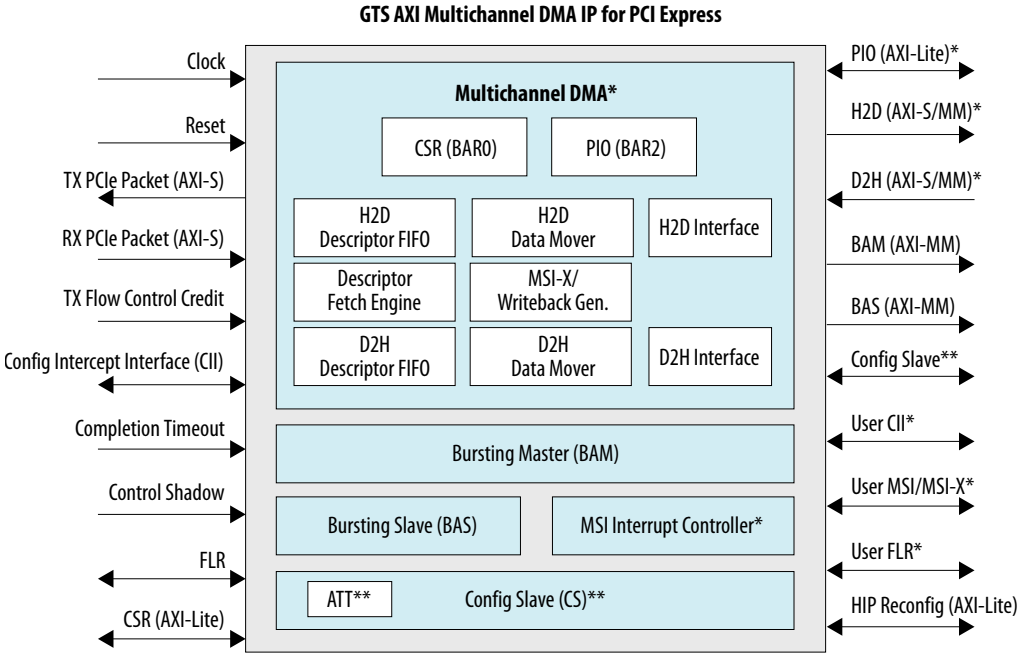
1.1.2.2. Endpoint Mode Features

- Supports up to 256 DMA channels
- H2D/D2H DMA data transfer via AXI-S/AXI-MM interface
- Integrated MSI-X for DMA operation and User MSI-X for user applications
- SR-IOV support with 4 PFs and 256 VFs
- Completion reordering
- Completion timeout
- Error interface
- User Function Level Reset

- 64-bit Metadata
- User mode and supported user interfaces
  - Multichannel DMA
    - H2D AXI-S/MM Manager interface
    - D2H AXI-S/MM Subordinate interface
    - PIO AXI4-Lite Manager interface
    - User MSI-X interface
    - User FLR interface
  - Bursting Master (BAM)
    - BAM AXI4 Manager interface
  - Bursting Slave (BAS)
    - BAS AXI4 Subordinate interface
    - User MSI interface
  - BAM + BAS
    - BAM AXI4 Manager interface
    - BAS AXI4 Subordinate interface
    - User MSI interface
  - BAM + MCDMA
    - BAM AXI4 Manager interface
    - H2D AXI-S/MM Manager interface
    - D2H AXI-S/MM Subordinate interface
    - User FLR interface
    - User MSI-X interface
  - BAM + BAS + MCDMA
    - BAM AXI4 Manager interface
    - BAS AXI4 Subordinate interface
    - H2D AXI-S/MM Manager interface
    - D2H AXI-S/MM Subordinate interface
    - User FLR interface
    - User MSI-X interface
- Optional User Interfaces
  - User Configuration Intercept interface

1.1.3. IP High-level Block Diagram

Figure 1. GTS AXI MCDMA IP High-level Block Diagram



Note: (*) Available in Endpoint Mode only.

Note: (**) Available in Root Port Mode only.

For more information on each of the modules, refer to Appendix A: Functional Description.

1.1.4. Licensing the IP

The Quartus Prime Pro Edition software installation includes the Altera FPGA IP library. This IP is available in the IP Catalog and Platform Designer.

Related Information

Altera® FPGA Software Installation and Licensing

1.1.5. Device Family Support

The following table defines the IP core support levels for the GTS AXI Multichannel DMA IP for PCI Express for the Agilex™ 5 devices and states the level of support available in the current release:

Table 2. GTS AXI Multichannel DMA IP for PCI Express Core Device Support Levels

Device Support Level	Definition
Advance	The IP core is available for simulation and compilation for this device family. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (datapath width, burst depth, I/O standards trade-offs).
Preliminary	The IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.
Final	The IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

Table 3. GTS AXI Multichannel DMA IP for PCI Express Core Device Family Support

Device Family	Support
Agilex 5	Preliminary Support

1.1.6. Device Speed Grade Support

Table 4. Device Speed Grade Support

Lane Rate	Link Configuration	PCIe* Streaming Interface Data Width	PLD Clock Frequency	Recommended Fabric Speed Grade
PCIe 4.0	X4	256-bit	300 MHz	-2
			250 MHz	-3
			200 MHz	-4
PCIe 3.0	X4	128-bit	250 MHz	-1, -2, -3, -4, -5, -6
			200 MHz	-1, -2, -3, -4, -5, -6

1.1.7. Resource Utilization

The following table provides the resource utilization numbers obtained by compiling the design using the Quartus Prime Pro Edition software targeting the device.

Table 5. AXI Streaming Port Configuration (in Endpoint Mode)

User Mode	Link Configuration	DMA Channels	ALMs	Logic Registers	M20Ks
Multichannel DMA	Gen3x4	1	21,288	65,612	565
		8	21,374	66,510	565
		256	21,183	67,044	599
	Gen4x4	1	23,102	77,438	532
		8	23,121	76,894	532
		256	22,927	75,073	566
BAM + BAS	Gen3x4	-	10,514	28,063	337

continued...

User Mode	Link Configuration	DMA Channels	ALMs	Logic Registers	M20Ks
	Gen4x4	-	13,250	38,807	397
BAM + BAS + MCDMA	Gen3x4	1	25,707	78,444	740
		8	25,617	77,661	740
		256	25,608	79,535	774
	Gen4x4	1	29,413	93,889	737
		8	29,463	94,681	737
		256	29,263	93,924	771

1.1.8. IP and Design Example Support

Table 6. GTS AXI Multichannel DMA IP Support Matrix for Agilex 5 Devices

Support level keys: S = simulation, C = compilation, T = timing, H = hardware, N/A = configuration not supported.

Configuration	IP Support		Design Example Support	
	EP	RP	EP	RP
PCIe 4.0 x4 256-bit	S, C, T, H	S, C, T, H	C, T	N/A
PCIe 3.0 x4 128-bit	S, C, T, H	S, C, T, H	C, T, H (*)	N/A

Note: (*) On Agilex 5 FPGA E-Series 065B Modular Development Kit with A5ED065BB32AE6SR0 device

1.2. About the Document

The GTS AXI Multichannel DMA IP for PCI Express* is a multi-channel Direct Memory Access (DMA) IP. Interfacing this IP with the GTS AXI Streaming IP for PCI Express* allows you to implement a PCI Express with DMA design on Agilex 5 FPGAs for high-bandwidth data transfers between the host or virtual machine and I/O devices.

This document introduces the details of the GTS AXI Multichannel DMA IP for PCI Express*, including features and functional descriptions of the various blocks within the IP. This document also describes the design requirements and guidelines, IP parameters, interfaces, and signals available to you when you use this IP.

Related Information

GTS AXI Streaming IP for PCI Express* User Guide

1.2.1. Target Audience

This document is intended for FPGA and system developers to use as a guide for evaluating the GTS AXI Multichannel DMA IP for PCI Express* in conjunction with the GTS AXI Streaming IP for PCI Express* on Agilex 5 FPGA devices.

1.2.2. Conventions Used in the Document

This document provides examples from both an operating system command-line interface (CLI) and a graphical user interface (GUI), and uses the following typographical conventions:

Table 7. Conventions

`codeblock`	Indicates CLI text, such as command prompts.
boldface	Text Indicates GUI fields or dialogue box title.
`monotype`	Indicates commands or file names.

1.2.3. Terminology

This section provides definitions for terms used in this document:

Table 8. Terminology

Term	Description
Channel	A DMA channel consists of a pair of Host-to-Device (H2D) and Device-to-Host (D2H) descriptor queues to handle bidirectional data transfer
Gen1	PCIe 1.0
Gen2	PCIe 2.0
Gen3	PCIe 3.0
Gen4	PCIe 4.0
GTS	High-speed transceiver of the Agilex 5 FPGA.
GTS AXI MCDMA IP	Refers to the GTS AXI Multichannel DMA IP for PCI Express.
GTS AXI Streaming IP	Refers to the GTS AXI Streaming Intel FPGA IP for PCI Express.

1.2.4. Acronyms

This section provides a list of acronyms used in this document and their expanded forms:

Table 9. Acronym

Acronym	Expanded Form
AXI	Advanced eXtensible Interface
AXI-MM	AXI Memory-Mapped
AXI-S	AXI Streaming *Note:* This acronym can also stand for AXI Stream. The terms "AXI Streaming" and "AXI Stream" are interchangeable.
BAM	Bursting Master
BAS	Bursting Slave
CSR	Control and Status Register
DMA	Direct Memory Access

continued...

Acronym	Expanded Form
D2H	Device-to-Host
H2D	Host-to-Device
HIP	Hard IP
MCDMA	Multichannel Direct Memory Access
QCSR	Queue Control and Status Register
TLP	Transaction Layer Packet

1.2.5. Reference Documents and Training

Table 10. Reference Documents

Document Names and Links
GTS AXI Streaming Intel® FPGA IP for PCI Express* User Guide
GTS Transceiver PHY User Guide
Scalable Scatter-Gather DMA Intel® FPGA IP User Guide
AMBA AXI4-Stream Protocol Specification
AMBA AXI Protocol Specification
Agilex 5 FPGA E-Series 065B Modular Development Kit

1.3. Applications

Agilex 5 FPGA devices serve a broad range of applications that require high performance, lower power, smaller form factors and lower logic densities. These characteristics make Agilex 5 ideal for midrange FPGA applications including:

- Data center
- Test and measurement equipment
- Video and broadcast equipment
- Industrial applications

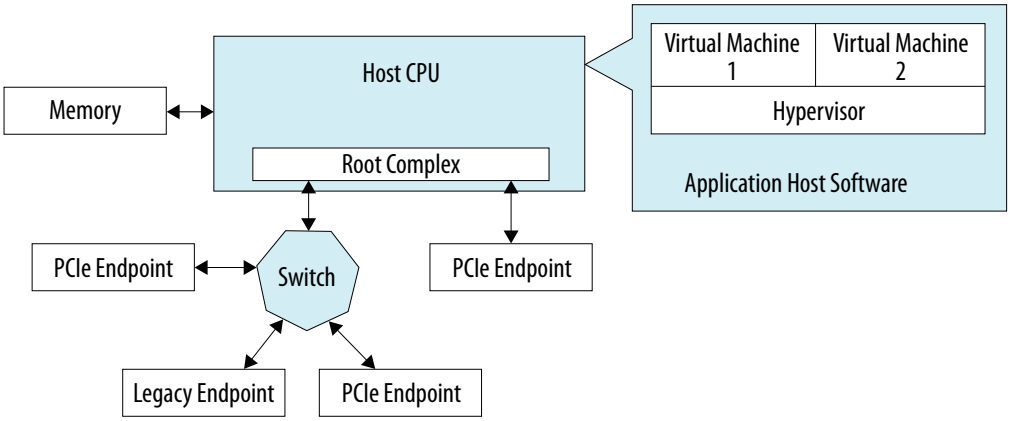
Most of the applications above involve underlying PCI Express connectivity. The GTS AXI Multichannel DMA IP for PCI Express together with the GTS AXI Streaming IP for PCI Express enable the implementation of efficient data transfers over a PCI Express connection for these applications.

1.4. Key Concepts

1.4.1. PCI Express and DMA

PCI Express* is a point-to-point, serial interconnect bus with a protocol stack that includes the Transaction, Data Link and Physical Layers. The protocol is scalable from 1 lane to 32 lanes per link, with data on the link serialized and sent from one device to another. It uses differential signaling with complementary pairs of signals for transmit and receive sides. It also uses packet-based transactions.

Figure 2. PCI Express Topology



Direct Memory Access (DMA) is a feature that enables data transfers between the Host memory and Device (Endpoint) memory without involving a processor. This feature offloads the processor during the data transfer process. A DMA controller can be used to facilitate such DMA operation, and it sends an interrupt to the processor once the transfer is completed.

The GTS AXI Multichannel DMA IP for PCI Express is one of the DMA IPs available in the Quartus Prime Pro Edition IP catalog to implement a DMA Controller for PCI Express applications.

1.4.2. Altera FPGA IPs for PCI Express and DMA Controller

Altera FPGA devices offer a wide variety of IPs (in addition to the GTS AXI Multichannel DMA IP for PCI Express and GTS AXI Streaming IP for PCI Express) for you to implement PCI Express in your designs on the respective FPGA device families depending on your design requirements.

- Avalon Streaming Intel FPGA IP for PCI Express:
  - R-Tile Avalon Streaming Intel FPGA IP for PCI Express
  - F-Tile Avalon Streaming Intel FPGA IP for PCI Express
  - P-Tile Avalon Streaming Intel FPGA IP for PCI Express
- AXI Streaming Intel FPGA IP for PCI Express
- Multi Channel DMA Intel FPGA IP for PCI Express
- AXI Multichannel DMA Intel FPGA IP for PCI Express
- Scalable Switch Intel FPGA IP for PCI Express

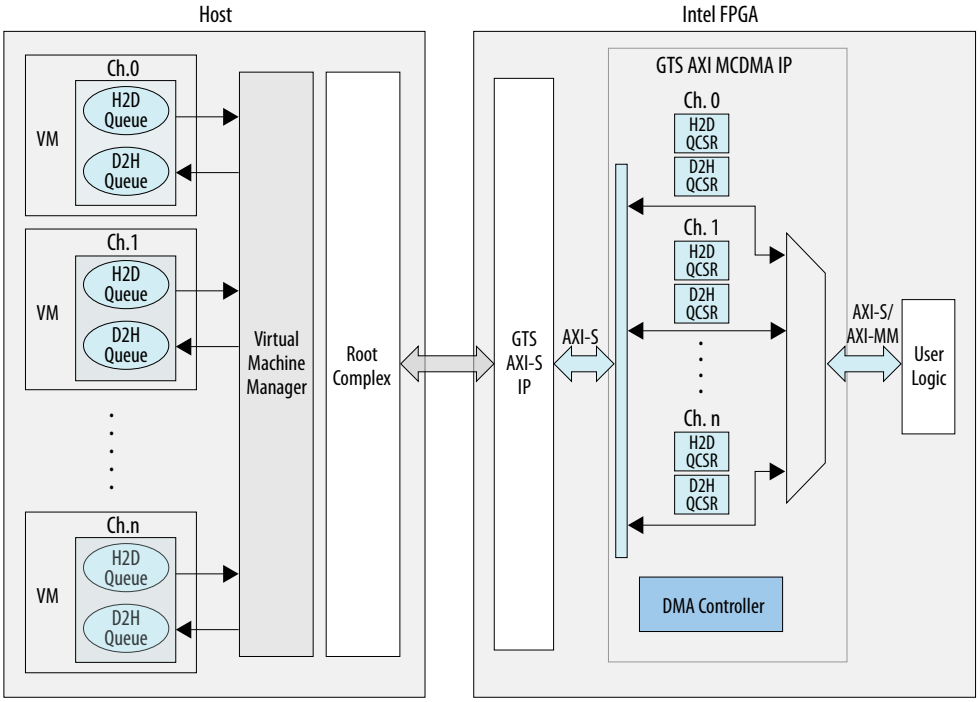
Refer to the Intel FPGA PCI Express IP Support Center for details on each IP.

1.4.3. Example Use Case

The figure below shows an example DMA application where the GTS AXI MCDMA IP plays a crucial role in a server's hardware infrastructure, enabling smooth communication between various VM clients and their FPGA-device counterparts. The GTS AXI MCDMA IP operates on descriptor-based queues, established by the driver software, facilitating efficient data transfers between the local FPGA and the Host. The

control logic embedded in the GTS AXI MCDMA IP intelligently interprets and executes the queued descriptors, ensuring the reliability and efficiency of data transfer operations.

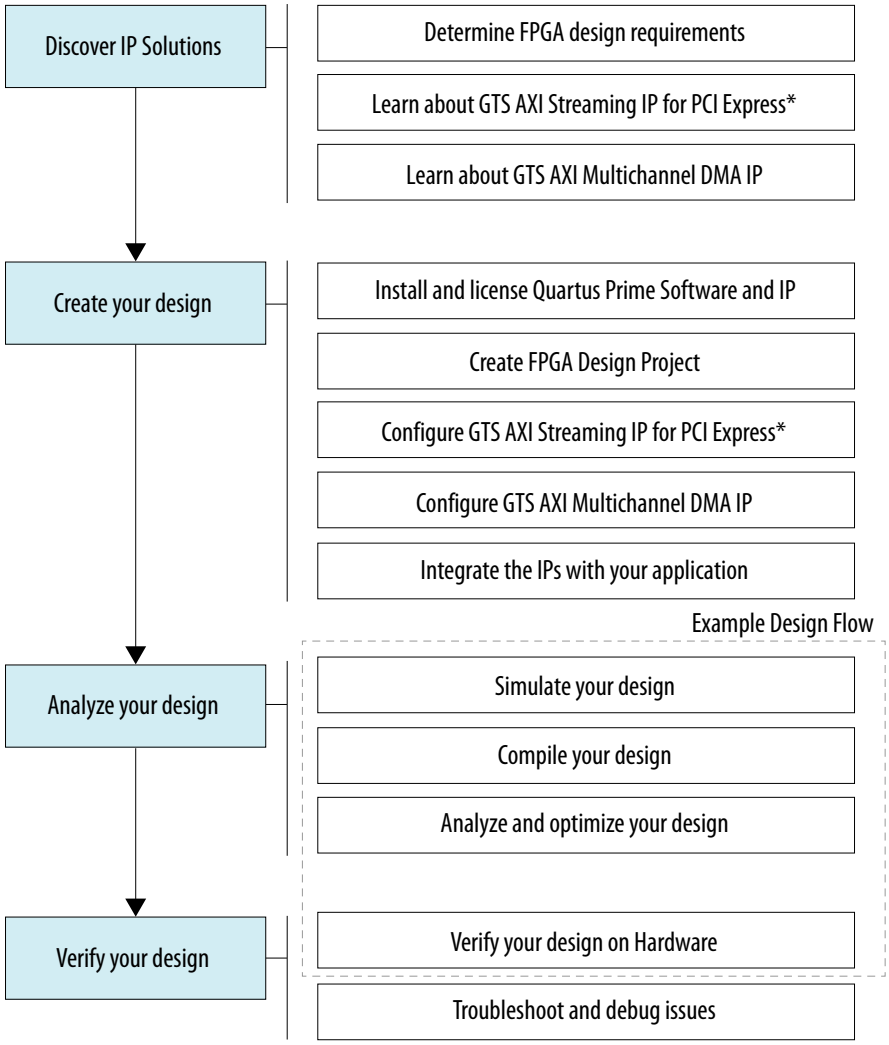
Figure 3. GTS AXI MCDMA IP for PCI Express: Server Hardware Infrastructure Example Use Case



1.5. IP Design Flow

The following flowchart illustrates the GTS AXI MCDMA IP design flow:

Figure 4. GTS AXI MCDMA IP Design Flow



The GTS AXI MCDMA IP also provides a hardware design example with drivers for design reference. When you generate the design example, the IP Parameter Editor automatically creates a design example with all necessary files for compilation. Altera recommends exploring the design example prior to creating your design to familiarize with the IP design flow. For more details, refer to the Generating the Design Example and Validating the IP sections.

Note: Design example simulation is not supported in the current release.

1.6. Design Guidelines

1.6.1. Supporting IPs

There are some essential IPs that are required in your design along with the GTS AXI MCDMA IP for Agilex 5 FPGA:

- GTS AXI Streaming IP for PCI Express - Implements a PCIe Endpoint or Root Port instance.
- GTS System PLL Clocks IP – Drives the system PLL of the GTS transceiver and provides a clock source for PCIe hard IP blocks. This is a mandatory IP for Agilex 5 designs that use system PLL clocking.
- GTS Reset Sequencer IP – Performs reset sequencing across all transceiver channels on one side of the device. This is a mandatory IP for Agilex 5 designs that use transceivers.
- Reset Release IP - Generates a signal to indicate the completion of device configuration. This is a mandatory IP for all Agilex 5 designs.

For more information on how to configure and connect these IPs, refer to the Quick Start Guide.

Related Information

GTS Transceiver PHY User Guide

1.6.2. Aligning IP Settings with the GTS AXI Streaming IP for PCI Express

As the GTS AXI MCDMA IP needs to interface with the GTS AXI Streaming IP for PCI Express, IP settings and interfaces of both IPs must be aligned to ensure that both IPs can operate smoothly.

The table below covers IP settings that must be aligned between the two IPs. For example, when the Hard IP Mode of the GTS AXI Streaming IP is configured as a Gen3 x4 Interface 128 bits, the PCIe Mode of the GTS AXI MCDMA IP must be aligned and configured to Gen3 1x4.

Table 11. IP Settings of the GTS AXI MCDMA IP and GTS AXI Streaming IP

GTS AXI MCDMA IP Settings		GTS AXI Streaming IP Settings	
PCIe Mode	Gen3 1x4	**Hard IP Mode**	Gen3 x4 Interface 128 bits
	Gen4 1x4		Gen4 x4 Interface 256 bits
Port Mode	Native Endpoint	**Port Mode**	Native Endpoint
	Root Port		Root Port
Multichannel DMA Mode			
BAR2 Address Width	4 KBytes - 12 bits ... 8 EBytes - 63 bits	**BAR2 Type**	64-bit prefetchable memory
		BAR2 Size	4 KBytes - 12 bits ... 8 EBytes - 63 bits
Enable Multiple Physical Functions	On/Off	**Enable Multiple Physical Functions**	On/Off

continued...

GTS AXI MCDMA IP Settings		GTS AXI Streaming IP Settings	
Total Physical Functions (PFs)	1-4	**Total Physical Functions (PFs)**	1-4
Enable SR-IOV Support	On/Off	**Enable SR-IOV Support**	On/Off
Total Virtual Functions of Physical Function	0 - 256	**Total Virtual Functions of Physical Function**	0 - 256
Enable MSI-X	On	**Enable MSI-X**	On
BAR0 Type	64-bit prefetchable memory	**BAR0 Type**	64-bit prefetchable memory
BAR0 Size	4 Mbytes – 22 bits	**BAR0 Size**	4 Mbytes – 22 bits
BAR1/3/4/5 Type	Disabled	**BAR1/3/4/5 Type**	Disabled
Expansion ROM Size	Disabled … 16 Mbytes – 24 bits	**Expansion ROM Size**	Disabled … 16 Mbytes – 24 bits
Bursting Slave, BAM + BAS User Modes			
Enable MSI Capability	On/Off	**PF0 Enable MSI**	On/Off
Enable ATT (Root Port mode only)	On/Off	**Enable Address Translation Services (ATS)**	On/Off
Endpoint Bursting Master, Bursting Slave, BAM + BAS User Modes			
BAR0/2/4 Type	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory	**BAR0/2/4 Type**	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory
BAR0/2/4 Size	4 KBytes - 12 bits … 16 EBytes - 64 bits	**BAR0/2/4 Size**	4 KBytes - 12 bits … 16 EBytes - 64 bits
BAR1/3/5 Type	Disabled	**BAR1/3/5 Type**	Disabled
Expansion ROM Size	Disabled … 16 Mbytes – 24 bits	**Expansion ROM Size**	Disabled … 16 Mbytes – 24 bits
Endpoint BAM + MCDMA, BAM + BAS + MCDMA User Modes			
BAR0 Type	64-bit prefetchable memory	**BAR0 Type**	64-bit prefetchable memory
BAR0 Size	4 MBytes - 22 bits	**BAR0 Size**	4 MBytes - 22 bits
BAR2/4 Type	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory	**BAR2/4 Type**	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory
BAR2/4 Size	4 KBytes - 12 bits … 16 EBytes - 64 bits	**BAR2/4 Size**	4 KBytes - 12 bits … 16 EBytes - 64 bits
BAR3/5 Type	Disabled	**BAR3/5 Type**	Disabled
Expansion ROM Size	Disabled … 16 Mbytes – 24 bits	**Expansion ROM Size**	Disabled … 16 Mbytes – 24 bits

For more details on each of the parameters, refer to the IP Settings section.

Some of the GTS AXI MCDMA IP interfaces are enabled by default. This requires the corresponding optional interface of the GTS AXI Streaming IP to be enabled so that signals from both IPs can be connected.

Table 12. Optional Interface to be Enabled on the GTS AXI Streaming IP

GTS AXI MCDMA IP Interfaces Enabled by Default	GTS AXI Streaming IP Settings	
Control Shadow Interface (Endpoint mode)	**Enable PCIe0 Control Shadow Interface**	On
Completion Timeout Interface	**Enable PCIe0 Completion Timeout Interface**	On
Error Interface	**Enable PCIe0 Error Interface**	On
Configuration Intercept Interface	**Enable PCIe0 Configuration Intercept interface**	On

As the `axi_st_clk` and `axi_mm_clk` of the GTS AXI MCDMA IP are expected to be driven by `coreclkout_hip_toapp` of the GTS AXI Streaming IP, the **PLD Clock Frequency** IP parameter must be configured to be aligned with the supported frequencies of the GTS AXI MCDMA IP.

Table 13. PLD Clock Frequency Settings

GTS AXI MCDMA IP Supported Clock Frequencies	GTS AXI Streaming IP Settings	
Gen4 1x4: 300/250/200 MHz Gen3 1x4: 250/200 MHz	**PLD Clock Frequency**	300/250/200 MHz

The GTS AXI Streaming IP supports both inband and sideband TLP header formats. However, the GTS AXI MCDMA IP only supports the sideband header format. Therefore, the **PCIe0 AXI-S Sideband Header** parameter must be enabled to ensure the packet streaming format of both IPs are aligned.

Table 14. AXI-S Sideband Header Settings

GTS AXI MCDMA IP Header Format	GTS AXI Streaming IP Settings	
Only the sideband TLP header format is supported.	**PCIe0 AXI-S Sideband Header**	On

1.7. Design Flow Requirements

1.7.1. Software Requirements

- Quartus Prime Pro Edition software version 25.1

1.7.2. Hardware Requirements

- Agilex 5 FPGA with transceiver
- Agilex 5 FPGA E-Series 065B Modular Development Kit (for design example run)
- Host system with PCIe 3.0 x4/x8/x16 slot

1.7.3. Supported EDA Tools

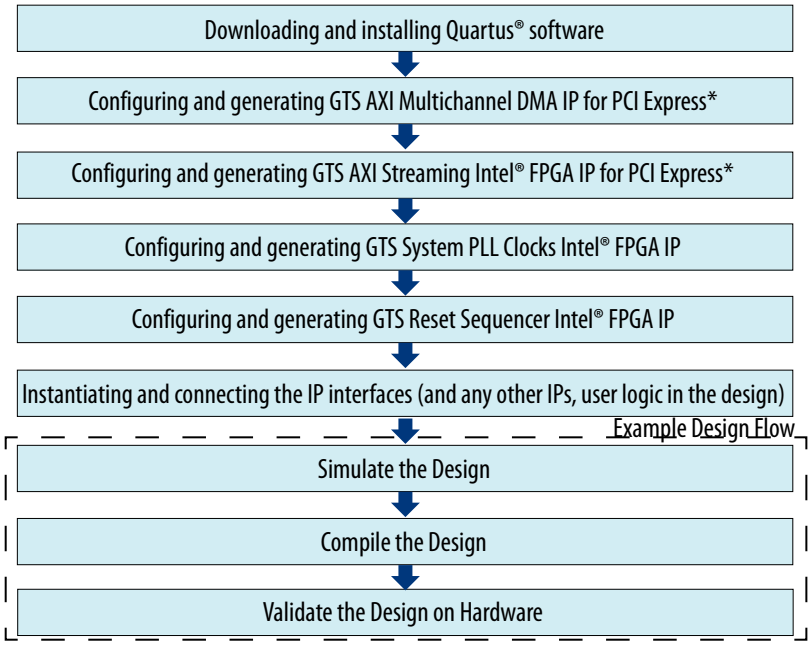
- Synopsys VCS MX* Simulator version V-2023.12-SP2-1
- Siemens EDA Questa* Simulator version 2024.3
- Aldec Riviera-PRO* Simulator version 2024.10
- Cadence Xcelium* Simulator version 23.09.004

2. Quick Start Guide

2.1. Step-by-step IP Bring-up

This chapter provides the steps to get started with the GTS AXI MCDMA IP, from installing the required software, to instantiating the IP, to simulating and compiling the IP(s) and verifying the functionality of this IP on hardware.

Figure 5. Flow Diagram for Getting Started with the GTS AXI MCDMA IP



2.1.1. Downloading and Installing Quartus Prime Software

Refer to the Quartus Prime Pro Edition User Guides for details on downloading and installing the Quartus Prime software and the necessary patches. For information on the patches, refer to the *Design Flow Requirements* section.

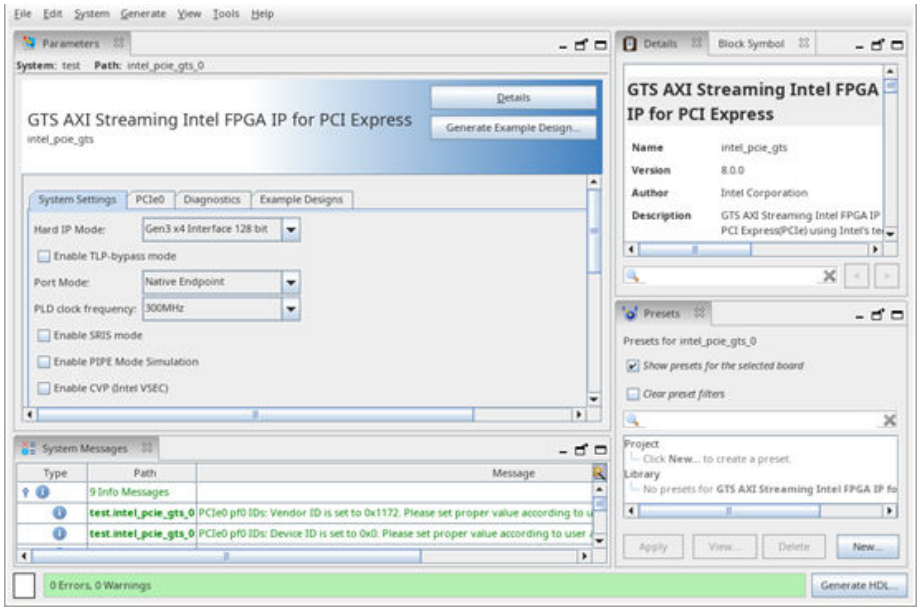
2.1.2. Configuring and Generating the GTS AXI Multichannel DMA IP for PCI Express

Refer to Configurating and Generating the GTS AXI Multichannel DMA IP for PCI Express to generate the IP.

2.1.3. Configuring and Generating the GTS AXI Streaming Intel FPGA IP for PCI Express

Adhere to the following procedure to generate the GTS AXI Streaming IP in standalone mode.

1. In the IP Catalog, select **GTS AXI Streaming Intel® FPGA IP for PCI Express** (**Library** → **Interface Protocols** → **PCI Express** → **GTS AXI Streaming Intel® FPGA IP for PCI Express**) and then click **Add**.
2. Specify a top-level name for your new custom IP variation and the directory for it. The IP Parameter Editor saves the IP variation settings in a file named <your_ip>.ip.
3. Click **Create**. The IP Parameter Editor appears as shown in the figure below. The GTS AXI Streaming Intel FPGA IP for PCI Express has **IP Settings**, **PCIe Settings**, and **Example Designs** tabs to allow you to quickly configure your custom IP variation.

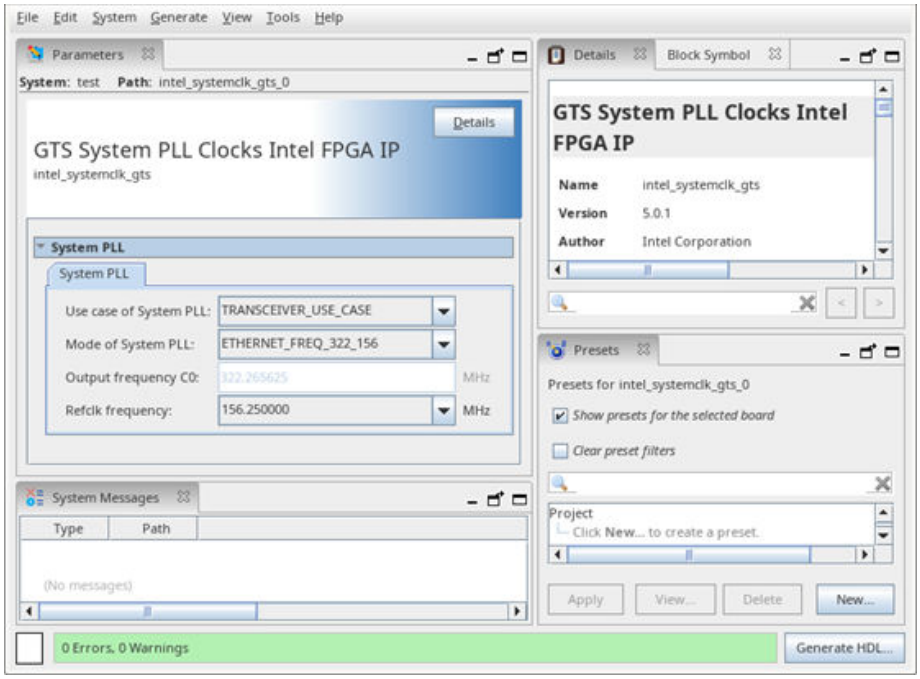


4. Generate the GTS AXI Streaming Intel FPGA IP for PCI Express.
   a. Click **Generate HDL**. The **Generation** dialogue box appears. Specify the output file generation options.
   b. Click **Generate**. This allows you to generate a GTS AXI Streaming IP in the standalone mode with the IP variation files generated according to your specifications.
   c. Click **Close** when the IP generation is complete. The IP Parameter Editor adds the top-level.ip file to the current project automatically. If you are prompted to manually add the .ip file to the project, click **Project** → **Add/Remove Files in Project** to add the file.

2.1.4. Configuring and Generating the GTS System PLL Clocks Intel FPGA IP

Following is the procedure to generate the GTS System PLL Clocks Intel FPGA IP.

1. Select **GTS System PLL Clocks Intel FPGA IP** in the IP Catalog.
2. Select **GTS System PLL Clocks Intel® FPGA IP** (**Library → Interface Protocols → Transceiver PHY → GTS System PLL Clocks Intel® FPGA IP**) and then click **Add**.
3. Specify a top-level name for your new custom IP variation and the directory for it. The IP Parameter Editor saves the IP variation settings in a file named <your_ip>.ip.
4. Click **Create**. The IP Parameter Editor appears as shown in the figure below.



5. Set the parameters while referring to the following table:

Parameter	Setting
Use case of System PLL	TRANSCEIVER_USER_CASE
Mode of System PLL	Select the setting that matches the PLD clock frequency in the GTS AXI Streaming IP. • PCIe 4.0 x8: PCIE_FREQ_500/PCIE_FREQ_450/PCIE_FREQ_400/PCIE_FREQ_350/ PCIE_FREQ_300/PCIE_FREQ_250/User_PCIE-based_Configuration_200. • PCIe 4.0 x4: PCIE_FREQ_350/PCIE_FREQ_300/PCIE_FREQ_250/User_PCIE-based_Configuration_200. • PCIe 3.0 x8: PCIE_FREQ_350/PCIE_FREQ_300/PCIE_FREQ_250/User_PCIE-based_Configuration_200. • PCIe 4.0 x2/x1, PCIe 3.0 x4/x2/x1: PCIE_FREQ_300/PCIE_FREQ_250/User_PCIE-based_Configuration_200.
Output frequency C0	Automatically set based on the Mode of System PLL setting.
Refclk frequency	100 MHz

6. Generate the GTS System PLL Clocks Intel FPGA IP.

a. Click **Generate HDL**. The **Generation** dialogue box appears. Specify the output file generation options.

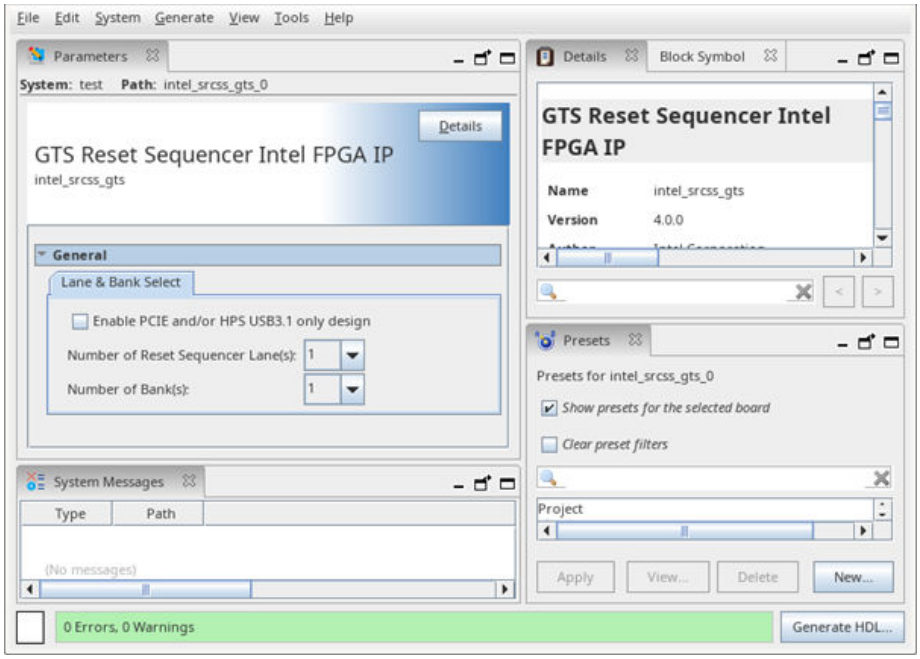
b. Click **Generate**. The IP variation files are generated according to your specifications.

c. Click **Close** when the IP generation is complete. The IP Parameter Editor adds the top-level.ip file to the current project automatically. If you are prompted to manually add the .ip file to the project, click **Project → Add/Remove Files in Project** to add the file.

2.1.5. Configuring and Generating the GTS Reset Sequencer Intel FPGA IP

Following is the process to configure and generate the GTS Reset Sequencer Intel FPGA IP. You have to instantiate only one GTS Reset Sequencer Intel FPGA IP for all the PCIe and non-PCIe channels on a side of the device.

1. Select **GTS Reset Sequencer Intel FPGA IP** in the IP Catalog.
2. Select **GTS Reset Sequencer Intel FPGA IP** (**Library → Interface Protocols → Transceiver PHY → GTS Reset Sequencer Intel FPGA IP**) and then click **Add**.
3. Specify a top-level name for your new custom IP variation and the directory for it. The IP Parameter Editor saves the IP variation settings in a file named <your_ip>.ip.
4. Click **Create**. The IP Parameter Editor appears as shown in the figure below.



5. Set the number of banks and lanes.

a. If the design only has PCIe channels on a side of the device, select the **Enable PCIE and/or HPS USB3.1 only design** option.

b. Set the **Number of Reset Sequencer Lane(s)** based on the total number of non-PCIe channels on the side of the device used in the design. The number of PCIe channels are not counted in the **Number of Reset Sequencer Lane(s)** parameter.

c. For a PCIe x8 design, set **Number of Bank(s)** to 2. For x4/x2/x1 designs, set **Number of Bank(s)** to 1.

6. Generate the GTS System PLL Clocks Intel FPGA IP.

a. Click **Generate HDL**. The **Generation** dialogue box appears. Specify the output file generation options.

b. Click **Generate**. The IP variation files are generated according to your specifications.

c. Click **Close** when the IP generation is complete. The IP Parameter Editor adds the top-level.ip file to the current project automatically. If you are prompted to manually add the .ip file to the project, click **Project → Add/Remove Files in Project** to add the file.

Note: For more descriptions of how to connect the GTS Reset Sequencer Intel FPGA IP, refer to Implementing the GTS Reset Sequencer Intel FPGA IP.

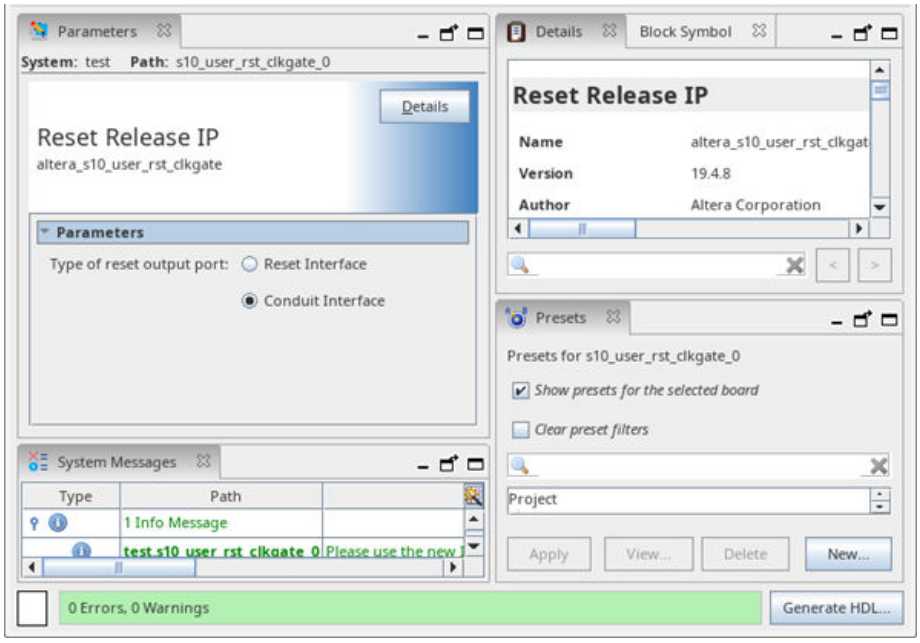
Related Information

Implementing the GTS Reset Sequencer Intel FPGA IP

2.1.6. Configuring and Generating the Reset Release IP

Following is the process to configure and generate the Reset Release IP for any design in an Agilex 5 device.

1. Select **Reset Release IP** in the IP Catalog.
2. Select **Reset Release IP** (**Library → Basic Functions → Configuration and Programming → Reset Release IP**) and then click **Add**.
3. Specify a top-level name for your new custom IP variation and the directory for it. The IP Parameter Editor saves the IP variation settings in a file named <your_ip>.ip.
4. Click **Create**. The IP Parameter Editor appears as shown in the figure below.

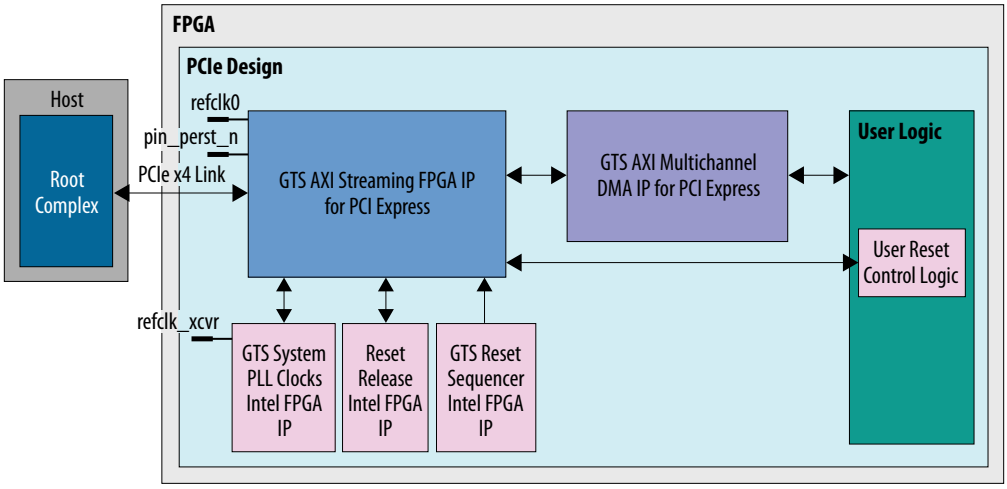


5. Select **Reset Interface** for the output port to match the GTS AXI Streaming IP.
6. Generate the GTS System PLL Clocks Intel FPGA IP.
   a. Click **Generate HDL**. The **Generation** dialogue box appears. Specify the output file generation options.
   b. Click **Generate**. The IP variation files are generated according to your specifications.
   c. Click **Close** when the IP generation is complete. The IP Parameter Editor adds the top-level.ip file to the current project automatically. If you are prompted to manually add the .ip file to the project, click **Project → Add/Remove Files in Project** to add the file.

2.1.7. Instantiating and Connecting the IP Interfaces

The GTS AXI Multichannel DMA IP enables the developer to build complex PCIe Endpoints. It requires the GTS AXI Streaming Intel FPGA IP for PCI Express, GTS System PLL Clocks Intel FPGA IP, GTS Reset Sequencer Intel FPGA IP, and Reset Release Intel FPGA IP for the implementation as shown in the figure below. The developer is required to create a customized reset controller for the reset interface in the GTS AXI Streaming IP. Otherwise, they can leverage the reset controller provided in the design example that can be generated in the Quartus Prime Pro Edition software. A top-level HDL file is required to instantiate and connect all the required IP components. Alternatively, the developer can consider using the Platform Designer tool available in the Quartus Prime Pro Edition software to instantiate, parameterize, connect and generate a subsystem that consists of all the required IPs.

Figure 6. An Example PCIe Subsystem Block Diagram



2.1.8. Simulate, Compile and Validate the Design on Hardware

The Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation optionally creates simulation files, including the functional simulation model, and vendor-specific simulator setup scripts for each IP core. You can use the functional simulation model and your own testbench or design example for simulation. IP generation output may also include scripts to compile and run any testbench. The scripts list all models or libraries you require to simulate your IP core.

The Quartus Prime software provides integration with many simulators and supports multiple simulation flows, including your own scripted and custom simulation flows. Whichever flow you choose, IP core simulation involves the following steps:

1. Generate the IP HDL, and simulator setup script files. Refer to Generating HDL for Simulation and Synthesis.
2. Set up your simulator environment and any simulation scripts.
3. Compile simulation model libraries.
4. Run your simulator.

It is recommended to use the design example provided along with the IP as a starting point where the testbench and Altera BFM are provided in the design example. Refer to Simulating the Design Example for more information.

The Quartus Prime Pro Edition Compiler supports a variety of flows to help you maximize performance and minimize compilation processing time. The modular Compiler is flexible and efficient, allowing you to run all modules in sequence with a single command, or to run and optimize each stage of compilation separately. As you develop and optimize your design, run only the Compiler stages that you need, rather than waiting for full compilation. Run full compilation only when your design is complete and you are ready to run all Compiler modules and generate a device programming image. Refer to Quartus® Prime Pro Edition User Guide: Design Compilation for more information.

The design example provided along with the IP has all the required settings like the pin assignments and timing constraints for full compilation and generating a device programming image for hardware validation. Please refer to Generating the Design

Example, Compiling the Design Example, and Validating the IP for guidance on how to generate, compile, and validate the design example on hardware using the software driver provided with the design example.

2.2. Reset Sequence

A user reset controller must be implemented in the user application logic, and it must follow the assertion and de-assertion sequence for graceful entry and exit for each of the resets (cold, warm, etc.) for the GTS AXI Streaming Intel FPGA IP for PCI Express, which is used along with the GTS AXI Multichannel DMA IP for PCI Express.

The focus of this section is to discuss the handling of the reset and handshake signals listed in the table below.

Signal Name	Direction	Description
p<n>_subsystem_cold_rst_n	Input	GTS AXI Streaming IP global reset. Active low signal. Resets sticky register bits. Can be implemented as a synchronous or asynchronous reset.
p<n>_subsystem_warm_rst_n	Input	GTS AXI Streaming IP warm reset. Active low signal. Does not reset sticky register bits. Can be implemented as a synchronous or asynchronous reset.
p<n>_subsystem_cold_rst_ack_n	Output	Indicates that a cold reset action is completed by the GTS AXI Streaming IP. Asynchronous handshake signal.
p<n>_subsystem_warm_rst_ack_n	Output	Indicates that a warm reset action is completed by the GTS AXI Streaming IP. Asynchronous handshake signal.
p<n>_subsystem_rst_req	Input	Reset entry indication from the user reset control logic. The GTS AXI Streaming IP queries the blocks in the design upon receiving this request and sends an acknowledgment back when the block is ready for reset entry. Asynchronous handshake signal.
p<n>_subsystem_rst_rdy	Output	Ready signal for the reset entry indication from the GTS AXI Streaming IP to the user reset control logic. Asynchronous handshake signal.
p<n>_initiate_warmrst_req	Output	Warm Reset entry required indication from the IP core to the user reset control logic. The initiator block cannot issue a new reset entry request until the previous reset sequence (entire reset operation) is completed. Asynchronous handshake signal.
p<n>_initiate_rst_req_rdy	Input	Indicates the user reset control logic has accepted initiation request and starts issuing resets. Asynchronous handshake signal.
p<n>_reset_status_n	Output	Active low signal. When low, it indicates the GTS AXI Streaming IP is in reset state. The application logic can use this signal to drive its reset network. Synchronous to coreclkout_hip of HIP.

2.2.1. Cold Reset Sequence Triggered by PERST# Signal

1. Cold Reset is initiated by the deassertion of the HIP input signal p<n>_pin_perst_n_i.
2. HIP asserts pld_link_reset_req to the GTS AXI Streaming IP.
3. The GTS AXI Streaming IP notifies the user reset controller by asserting p<n>_initiate_warmrst_req.

4. The user reset controller asserts p0_subsystem_rst_req.
5. The GTS AXI Streaming IP sequences its internal blocks for reset entry. When the internal blocks are ready for reset, the GTS AXI Streaming IP asserts p<n>_subsystem_rst_rdy to the user reset controller.
6. The user reset controller acknowledges to the GTS AXI Streaming IP that it is ready for reset by asserting p<n>_initiate_rst_req_rdy.

2.2.2. Warm Reset Sequence Triggered by Hot Reset

1. HIP asserts pld_link_reset_req to the GTS AXI Streaming IP.
2. The GTS AXI Streaming IP notifies the user reset controller by asserting p<n>_initiate_warmrst_req.
3. The user reset controller then asserts p<n>_subsystem_rst_req.
4. The GTS AXI Streaming IP sequences its internal blocks for reset entry. When the internal blocks are ready for reset, the GTS AXI Streaming IP asserts p<n>_subsystem_rst_rdy to the user reset controller.
5. The user reset controller acknowledges to the subsystem that it is ready for reset by asserting p<n>_initiate_rst_req_rdy.
6. The GTS AXI Streaming IP then asserts pld_warm_rst_rdy to HIP.
7. HIP asserts p<n>_reset_status_n indicating the application logic needs to be in reset.
8. The user reset controller asserts p<n>_subsystem_warm_rst_n, p<n>_axi_st_areset_n, and p<n>_axi_lite_areset_n.

2.2.3. Cold Reset Triggered by User

1. Cold Reset is initiated by the user reset controller by the assertion of the p<n>_subsystem_rst_req.
2. The GTS AXI Streaming IP sequences its internal blocks for reset entry. When the internal blocks are ready for reset, the GTS AXI Streaming IP asserts p<n>_subsystem_rst_rdy to the user reset controller.
3. User reset controller asserts p<n>_subsystem_cold_rst_n, p<n>_subsystem_warm_rst_n, p<n>_axi_st_areset_n, and p<n>_axi_lite_areset_n.

2.2.4. Warm Reset Triggered by User

The user reset controller triggers the Warm Reset flow the same way that it triggers the Cold Reset flow, with the exception that the p<n>_subsystem_cold_rst_n is not asserted for this flow.

3. Configuring and Generating the GTS AXI Multichannel DMA IP for PCI Express

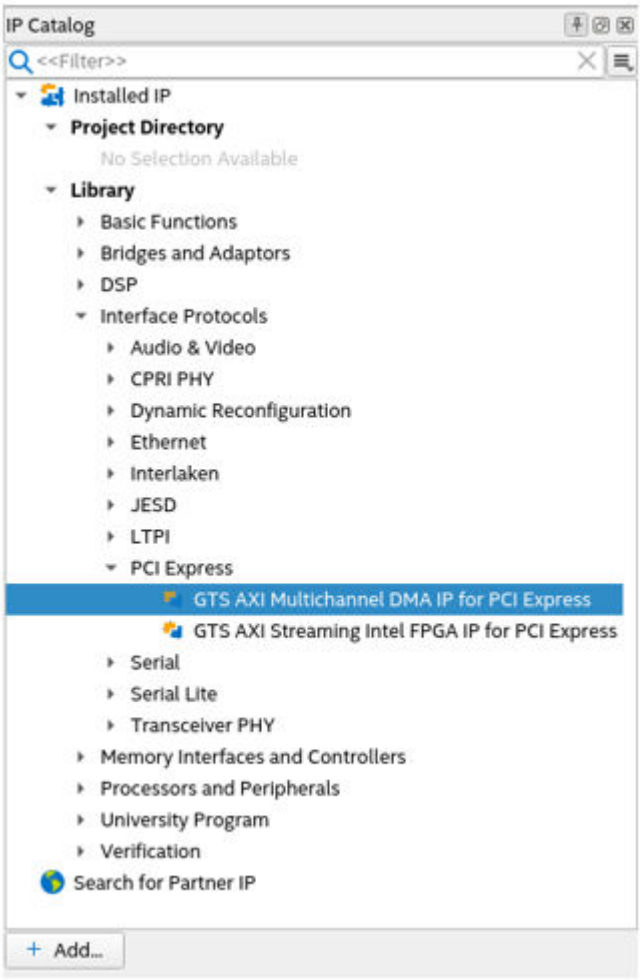
This chapter describes how to create a new project in the Quartus Prime Pro Edition software, add, parameterize, and generate the GTS AXI Multichannel DMA IP for simulation and synthesis.

3.1. Creating the Quartus Prime Project

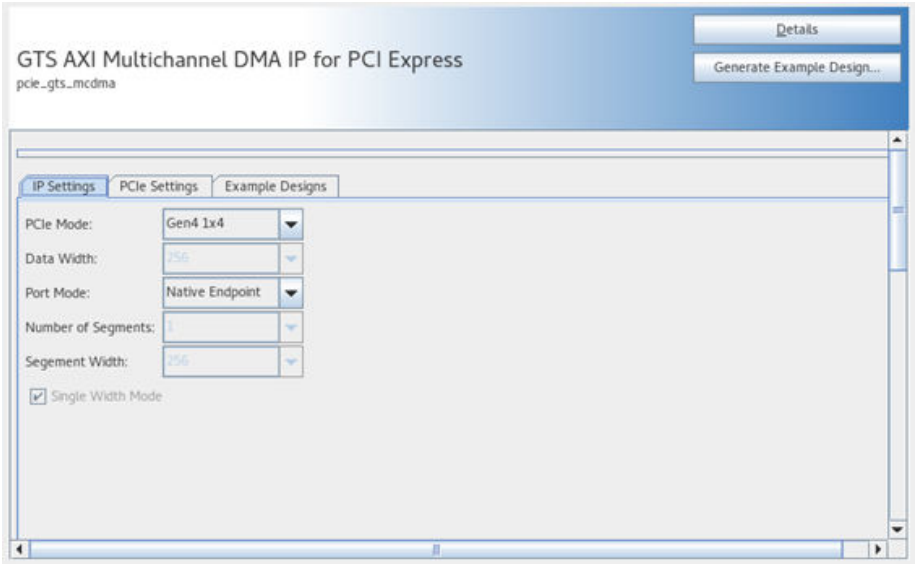
1. In the Quartus Prime Pro Edition software, create a new project by clicking **File** → **New Project Wizard**. Click **Next**.
2. Select **Empty Project** type and specify the **Directory, Name, and Top-Level Entity**. Click **Next**.
3. Specify the **Family, Device & Board Settings** as follows:
1. Select **Family**: Agilex 5 (E-Series/D-Series).
2. Select **Target Device** AD065BB32AE4SR0.
3. Click **Finish**.

3.2. Configuring the GTS AXI Multichannel DMA IP for PCI Express

1. Select from **Tools** → **IP Catalog** to open the IP Catalog.



2. Select **GTS AXI Multichannel DMA IP for PCI Express** (**Library** → **Interface Protocols** → **PCI Express** → **GTS AXI Multichannel DMA IP for PCI Express**) and then click **Add**.
3. In the New IP Variant dialogue box, specify a top-level name for your new custom IP variation and the directory for it. The IP Parameter Editor saves the IP variation settings in a file named <your_ip>.ip.
4. Click **Create**. The IP Parameter Editor appears as shown in the figure below. The GTS AXI Multichannel DMA IP for PCI Express has **IP Settings**, **PCIe Settings**, and **Example Designs** tabs to allow you to quickly configure your custom IP variation.



5. Specify the parameters for your IP core variation. Refer to the following sections for information about specific IP parameters.

Note: Ensure you apply settings that are aligned with the GTS AXI Streaming IP. Refer to Aligning IP Settings with the GTS AXI Streaming IP for PCI Express for more information.

3.2.1. IP Settings

Figure 7. IP Settings Tab Parameters

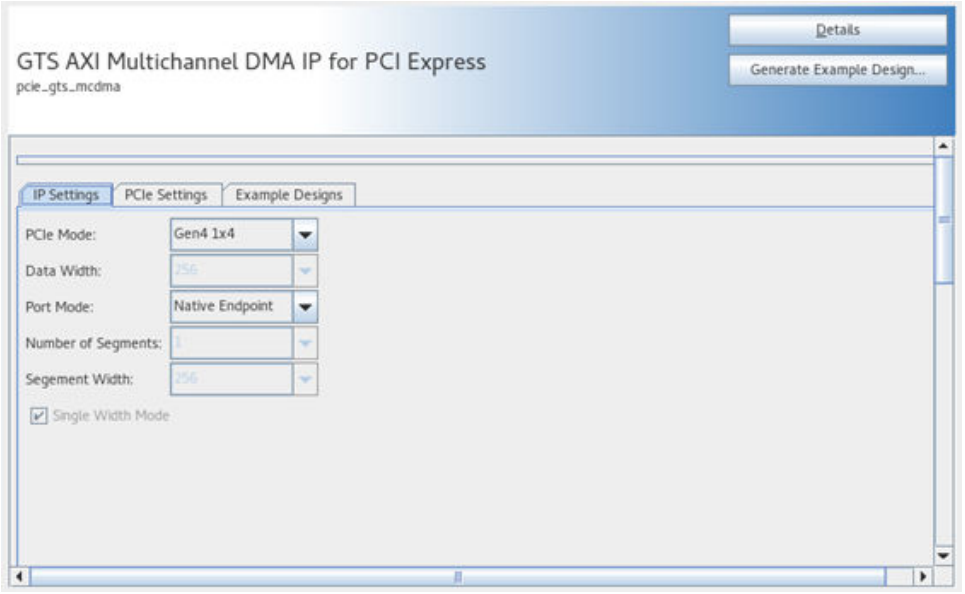


Table 15. IP Settings Parameters

Parameter	Value	Default Value	Description
PCIe Mode	**Gen4 1x4** **Gen3 1x4**	**Gen4 1x4**	Selects the width of the data interface between the transaction layer and the application layer implemented in the PLD fabric, the lane data rate and the lane rate. Selects the following elements: • Lane data rate: Gen3 and Gen4 are supported. • Lane width: x4 supports both Root Port and Endpoint modes.
Data Width	**256** **128**	**256**	Supported data widths per the PCIe mode: • Gen4 1x4: 256 bits • Gen3 1x4: 128 bits The IP Parameter Editor automatically selects a value per **PCIe Mode** settings.
Port Mode	**Native Endpoint** **Root Port**	**Native Endpoint**	Selects the port mode.
Number of Segments	**1**	**1**	Number of segments in the data interface. The IP Parameter Editor automatically selects a value per **Data Width**: • 256 bits: 1 • 128 bits: 1
Segment Width	**256** **128**	**256**	Segment data width. The IP Parameter Editor automatically selects a value based on the **Data Width** settings.
Single Width Mode	**On**	**On**	Sets the Single Width Mode. The IP Parameter Editor automatically sets this to **On** based on the **PCIe Mode** and **Data Width** settings.

3.2.2. PCIe Settings

3.2.2.1. MCDMA Settings

3.2.2.1.1. Multichannel DMA Mode

Figure 8. Multichannel DMA Mode Parameters

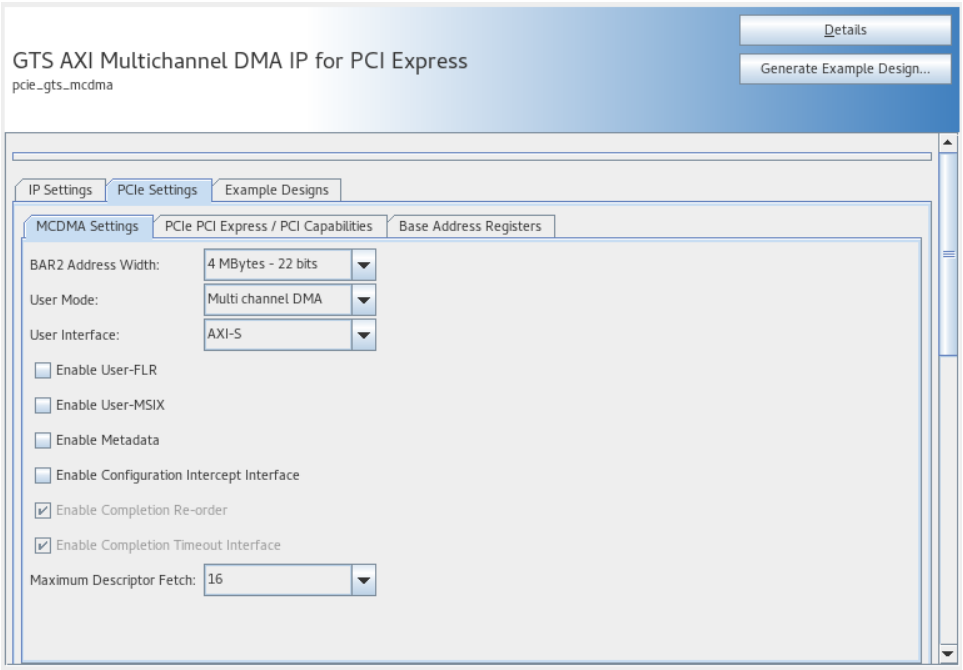


Table 16. Multichannel DMA Mode Parameters

Parameter	Value	Default Value	Description
BAR2 Address Width	**128 Bytes – 7 bits** **. . .** **8 EBytes – 63 bits**	**4 Mbytes – 22 bits**	AXI4-Lite PIO interface is available in Multichannel DMA mode for the Host to perform MMIO read/ write to user logic. This parameter sets the BAR2 address width used for the PIO functionality. *Note:* This parameter is available only in **Multichannel DMA** user mode.
User Mode	**Multichannel DMA** **Bursting Master** **Bursting Slave** **BAM+BAS** **BAM+MCDMA** **BAM+BAS+MCDMA**	**Multichannel DMA**	This parameter allows you to configure the mode of operation for the user application: • Bursting Master (BAM) and Bursting Slave (BAS) offer bursting AXI-MM capabilities without DMA functionality *Note:* Only Bursting Master, Bursting Slave and BAM+BAS options are available in Root Port Mode.
User Interface	**AXI-S** **AXI-MM**	**AXI-S**	Sets the type of H2D/D2H DMA user interface.
Enable User-FLR	**On / Off**	**Off**	Select to enable the User FLR interface, which allows the passing of FLR signals to the user side application.

continued...

Parameter	Value	Default Value	Description
			Note: User FLR is not supported in Bursting Master, Bursting Slave, and BAM+BAS modes.
Enable User-MSIX	**On / Off**	**Off**	The User MSI-X interface enables the user application to initiate interrupts through the MCDMA IP. *Note:* User MSI-X is not supported in Bursting Master, Bursting Slave, and BAM+BAS modes.
Enable Metadata	**On / Off**	**Off**	Enables or disables 8-byte metadata. Metadata is the 8-byte user application data that is transferred to and from the device via the DEST address field (for H2D DMA) and SRC address field (for D2H DMA) in the AXI Streaming mode.
Enable Configuration Intercept Interface	**On / Off**	**Off**	Select to enable the user application to detect the occurrence of a CFG request on the link and to modify its behavior via the User Configuration Intercept Interface.
Enable Completion Re-order	**On**	**On**	Enables completion reorder.
Enable Completion Timeout Interface	**On**	**On**	Enables the completion timeout interface.
Maximum Descriptor Fetch	**16** **32** **64**	**16**	Sets the maximum descriptors that are fetched per D2H prefetch channel. Applicable to the AXI-S interface type.

3.2.2.1.2. Bursting Master Mode

Figure 9. Root Port Bursting Master Mode Parameters

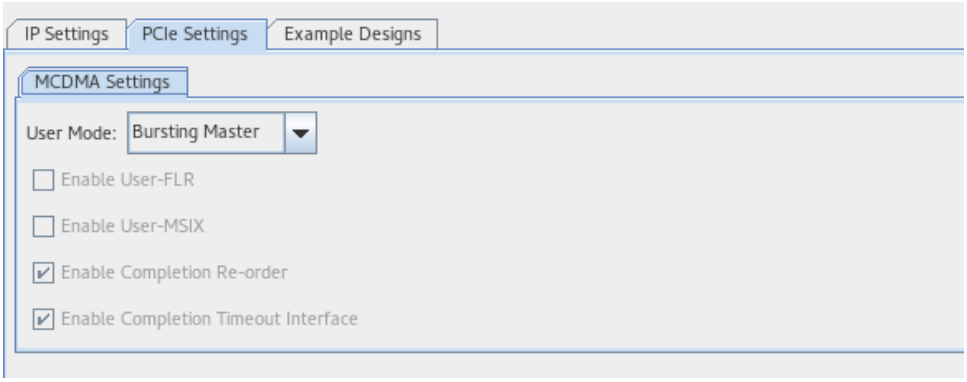


Figure 10. Endpoint Bursting Master Mode Parameters

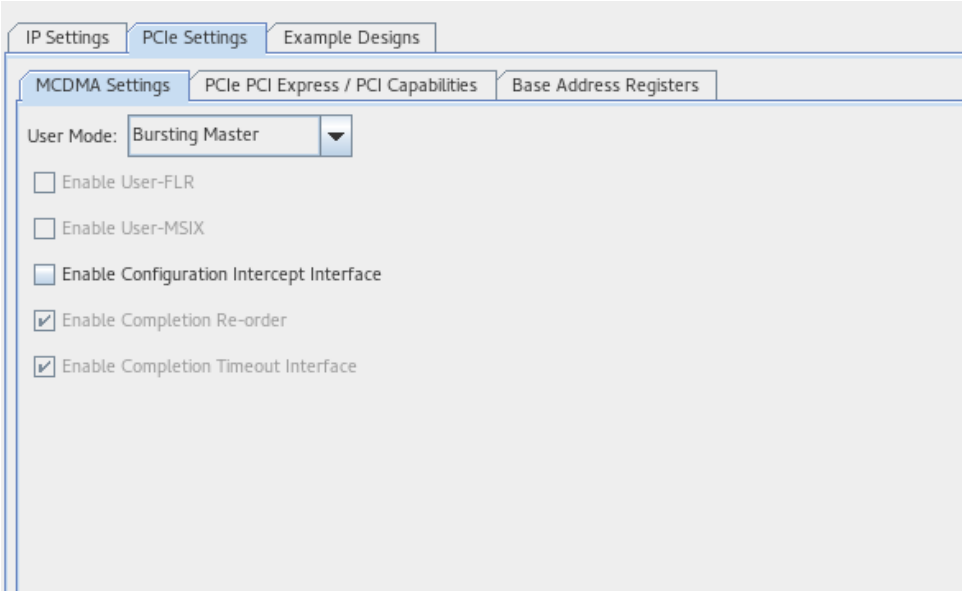


Table 17. Bursting Master Mode Parameters

Parameter	Value	Default Value	Description
Enable Configuration Intercept Interface	**On/Off**	**Off**	Select to enable the user application to detect the occurrence of a CFG request on the link and to modify its behavior via the configuration intercept interface. Applicable to Endpoint mode only.
Enable Completion Reorder	**On**	**On**	Enables completion reorder.
Enable Completion Timeout Interface	**On**	**On**	Enables the completion timeout interface.

3.2.2.1.3. Bursting Slave Mode

Figure 11. Root Port Bursting Slave Parameters

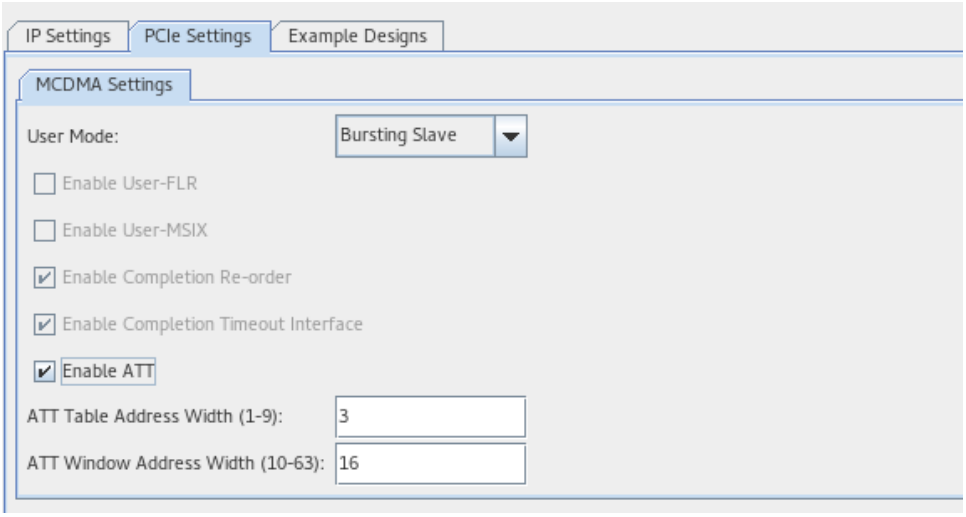


Table 18. Root Port Bursting Slave Parameters

Parameter	Value	Default Value	Description
Enable Completion Reorder	**On**	**On**	Enables completion reorder.
Enable Completion Timeout Interface	**On**	**On**	Enables the completion timeout interface.
Enable ATT	**On/Off**	**Off**	Enables the Address Translation Table for BAS.
ATT Table Address Width	**1 - 9**	**3**	Sets the depth of ATT. The depth is equal to 2 to the power of the number entered.
ATT Window Address Width	**10 - 63**	**16**	Sets the number of BAS address bits to be used directly.

Figure 12. Endpoint Bursting Slave Parameters

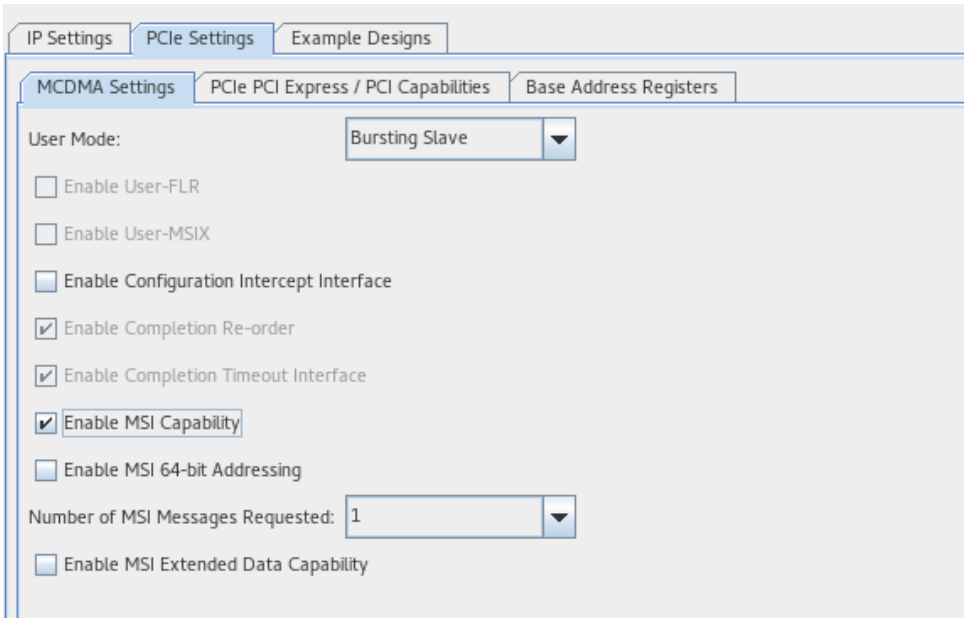


Table 19. Endpoint Bursting Slave Parameters

Parameter	Value	Default Value	Description
Enable Configuration Intercept Interface	**On/Off**	**Off**	Select to enable the user application to detect the occurrence of a CFG request on the link and to modify its behavior via the configuration intercept interface. Applicable to Endpoint only.
Enable Completion Reorder	**On**	**On**	Enables completion reorder.
Enable Completion Timeout Interface	**On**	**On**	Enables the completion timeout interface.
Enable MSI Capability	**On/Off**	**Off**	Enables or disables the MSI capability. *Note:* MSI is supported only in BAS and BAM + BAS modes.
Enable MSI 64-bit Addressing	**On/Off**	**Off**	Enables or disables 64-bit MSI addressing.
Number of MSI Messages Requested	**1, 2, 4, 8, 16, 32**	**1**	Sets the number of messages that the application can request in the multiple message capable field of the Message Control register.
Enable MSI Extended Data Capability	**On/Off**	**Off**	Enables or disables MSI extended data capability.

3.2.2.1.4. BAM + BAS Mode

Figure 13. Root Port BAM + BAS Mode Parameters

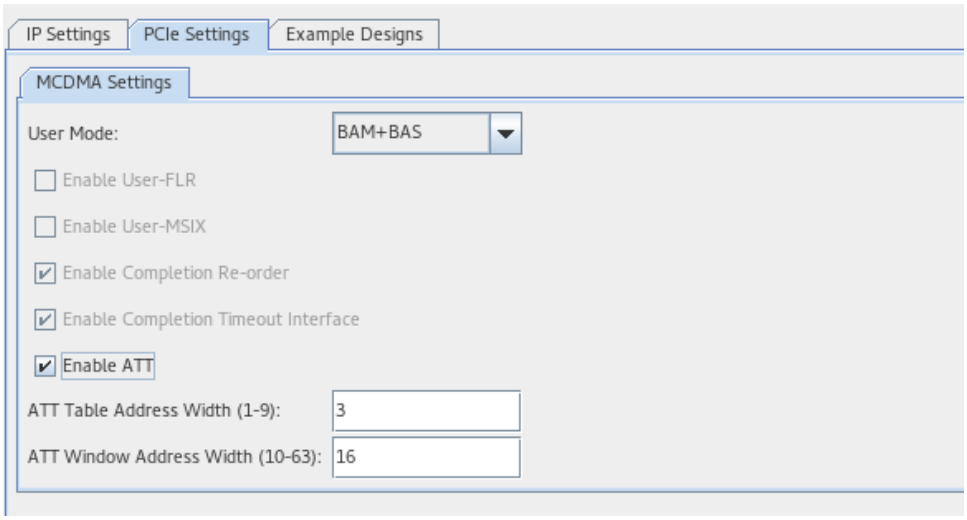


Table 20. Root Port BAM + BAS Mode Parameters

Parameter	Value	Default Value	Description
Enable Completion Re-order	**On**	**On**	Enables completion re-order.
Enable Completion Timeout Interface	**On**	**On**	Enables completion timeout interface.
Enable ATT	**On / Off**	**Off**	Enables Address Translation Table for BAS.
ATT Table Address Width	**1 - 9**	**3**	Sets the depth of ATT. Depth is equal to 2 to the power of the number entered.
ATT Window Address Width	**10 - 63**	**16**	Sets the number of BAS address bits to be used directly.

Figure 14. Endpoint BAM + BAS Mode Parameters

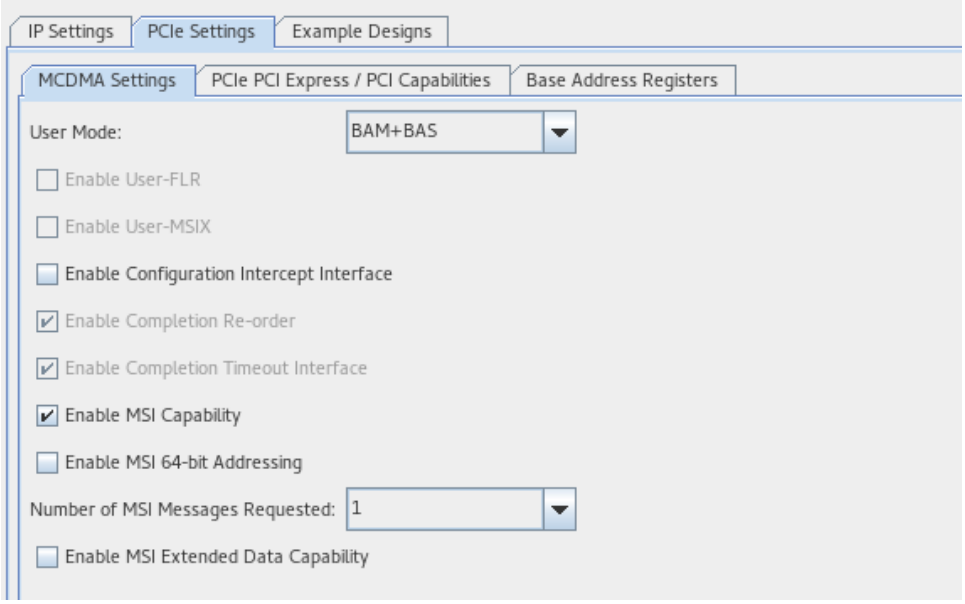


Table 21. Endpoint BAM + BAS Mode Parameters

Parameter	Value	Default Value	Description
Enable Configuration Intercept Interface	**On / Off**	**Off**	Select to enable the user application to detect the occurrence of a CFG request on the link and to modify its behavior via the configuration intercept interface. Applicable to Endpoint only.
Enable Completion Reorder	**On**	**On**	Enables completion reorder.
Enable Completion Timeout Interface	**On**	**On**	Enables completion timeout interface.
Enable MSI Capability	**On / Off**	**Off**	Enables or disables MSI capability. *Note:* MSI is supported only in the BAS and BAM + BAS modes.
Enable MSI 64-bit Addressing	**On / Off**	**Off**	Enables or disables 64-bit MSI addressing.
Number of MSI Messages Requested	**1, 2, 4, 8, 16, 32**	**1**	Sets the number of messages that the application can request in the multiple message capable field of the Message Control register.
Enable MSI Extended Data Capability	**On / Off**	**Off**	Enables or disables MSI extended data capability.

3.2.2.1.5. BAM + MCDMA Mode

Figure 15. BAM + MCDMA Mode Parameters

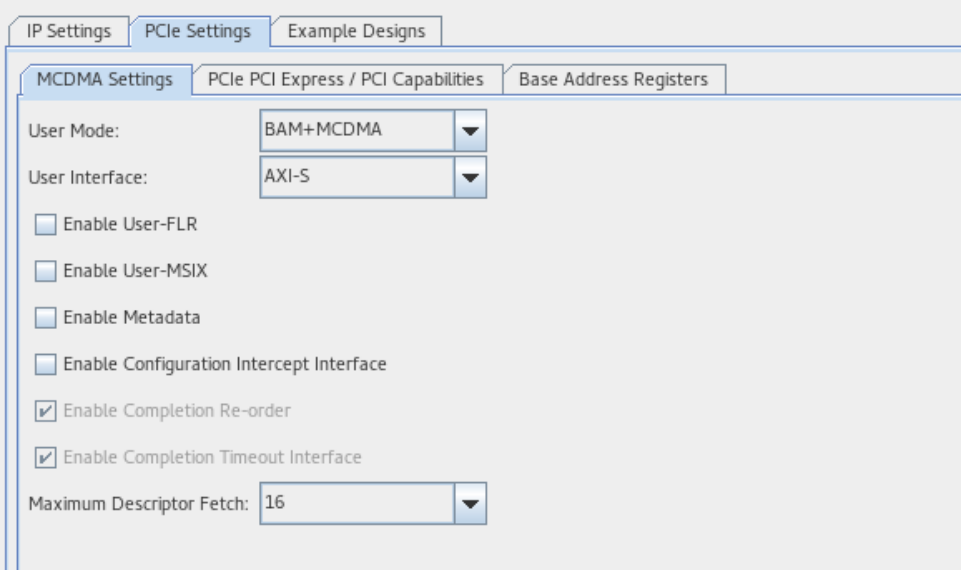


Table 22. BAM + MCDMA Mode Parameters

Parameter	Value	Default Value	Description
User Interface	**AXI-S** **AXI-MM**	**AXI-S**	Sets the type of DMA user interface.
Enable User-FLR	**On / Off**	**Off**	Select to enable the User FLR interface, which allows the passing of FLR signals to the user side application.
Enable User-MSIX	**On / Off**	**Off**	The User MSI-X interface enables the user application to initiate interrupts through the MCDMA IP.
Enable Metadata	**On / Off**	**Off**	Enables or disables 8-byte metadata. Metadata is the 8-byte user application data that is transferred to and from the device via the DEST address field (for H2D DMA) and SRC address field (for D2H DMA) in the AXI Streaming mode.
Enable Configuration Intercept Interface	**On / Off**	**Off**	Select to enable the user application to detect the occurrence of a CFG request on the link and to modify its behavior via the configuration intercept interface.
Enable Completion Re-order	**On**	**On**	Enables completion re-order.
Enable Completion Timeout Interface	**On**	**On**	Enables completion timeout interface.
Maximum Descriptor Fetch	**16** **32** **64**	**16**	Sets the maximum descriptors that are fetched per D2H prefetch channel. Applicable to the AXI-S interface type.

3.2.2.1.6. BAM + BAS + MCDMA Mode

Figure 16. BAM + BAS + MCDMA Mode Parameters

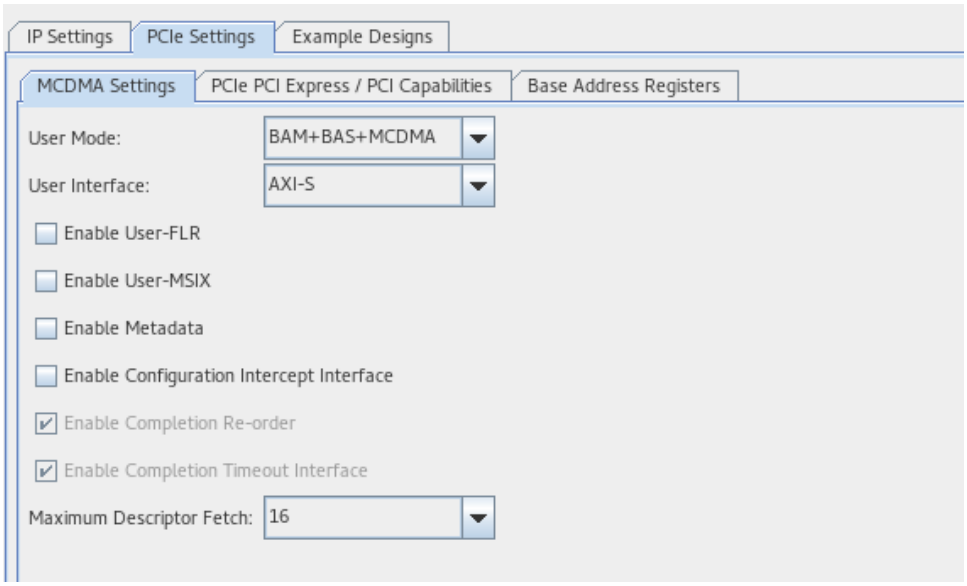


Table 23. BAM + BAS + MCDMA Mode Parameters

Parameter	Value	Default Value	Description
User Interface	**AXI-S** **AXI-MM**	**AXI-S**	Sets the type of DMA user interface.
Enable User-FLR	**On / Off**	**Off**	Select to enable the User FLR interface, which allows the passing of FLR signals to the user side application.
Enable User-MSIX	**On / Off**	**Off**	The User MSI-X interface enables the user application to initiate interrupts through the MCDMA IP.
Enable Metadata	**On / Off**	**Off**	Enables or disables 8-byte metadata. Metadata is the 8-byte user application data that is transferred to and from the device via the DEST address field (for H2D DMA) and SRC address field (for D2H DMA) in the AXI Streaming mode.
Enable Configuration Intercept Interface	**On / Off**	**Off**	Select to enable the user application to detect the occurrence of a CFG request on the link and to modify its behavior via the configuration intercept interface.
Enable Completion Re-order	**On**	**On**	Enables completion re-order.
Enable Completion Timeout Interface	**On**	**On**	Enables completion timeout interface.
Maximum Descriptor Fetch	**16** **32** **64**	**16**	Sets the maximum descriptors that are fetched per D2H prefetch channel. Applicable to the AXI-S interface type.

3.2.2.2. PCI Express / PCI Capabilities

This group of parameters defines various capability properties of the IP core. Some of these parameters are stored in the PCI Configuration Space - PCI Compatible Configuration Space. The byte offset indicates the parameter address.

3.2.2.2.1. PCIe Device

Figure 17. PCIe Multifunction and SR-IOV System Settings Parameters

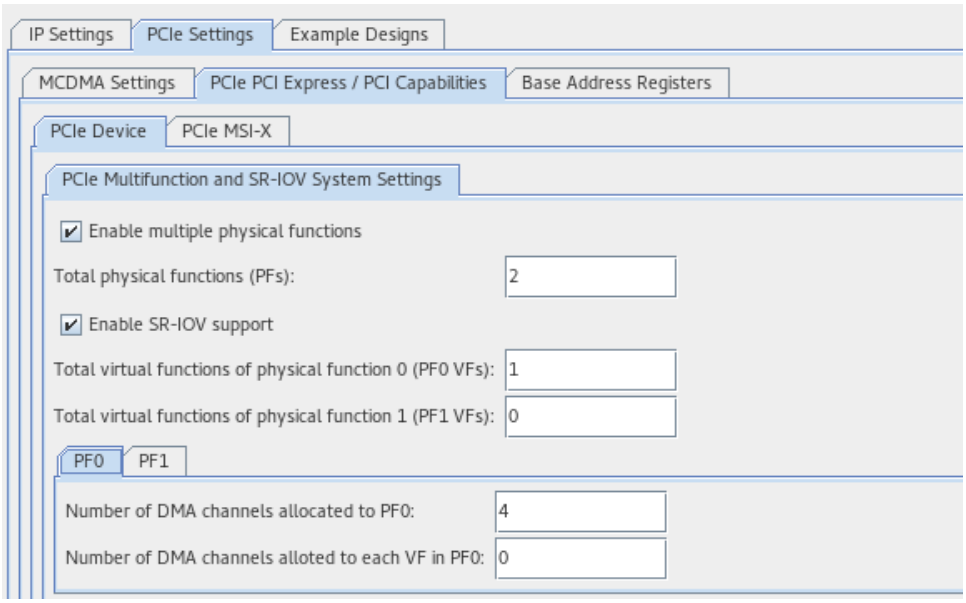


Table 24. PCIe Multifunction and SR-IOV System Settings Parameters

Parameter	Value	Default Value	Description
Enable multiple physical functions	**On / Off**	**Off**	Enables multiple physical functions.
Total physical functions (PFs)	**1 - 4**	**1**	Sets the number of physical functions. This parameter is available when **Enable multiple physical functions** is **On**.
Enable SR-IOV support	**On / Off**	**Off**	Enables SR-IOV.
Total virtual functions of physical function <n> (PF<n> VFs)	**0 - 256**	**0**	Sets the number of VFs to be assigned to Physical Function <n>. This parameter is available when **Enable SR-IOV support** is **On**. By default, only the parameter for physical function 0 appears. If you change the value of **Total physical functions (PFs)**, other parameters appear corresponding to the number of physical functions enabled.
Number of DMA channels allocated to PF<n>	**0 - 256**	**4**	Sets the number of DMA channels allocated to PF<n>. The GTS AXI MCDMA IP supports up to 256 DMA channels across all PFs and VFs.
Number of DMA channels allocated to each VF in PF<n>	**0 - 256**	**0**	Sets the number of DMA channels allocated to each VF in PF<n>. The GTS AXI MCDMA IP supports up to 256 DMA channels across all PFs and VFs.

3.2.2.2.2. PCIe MSI-X

PCIe PF MSI-X

The PF MSI-X capability parameters in Multichannel DMA, BAM+MCDMA and BAM+BAS +MCDMA modes are automatically set and cannot be modified.

Figure 18. PCIe PF MSI-X Parameters

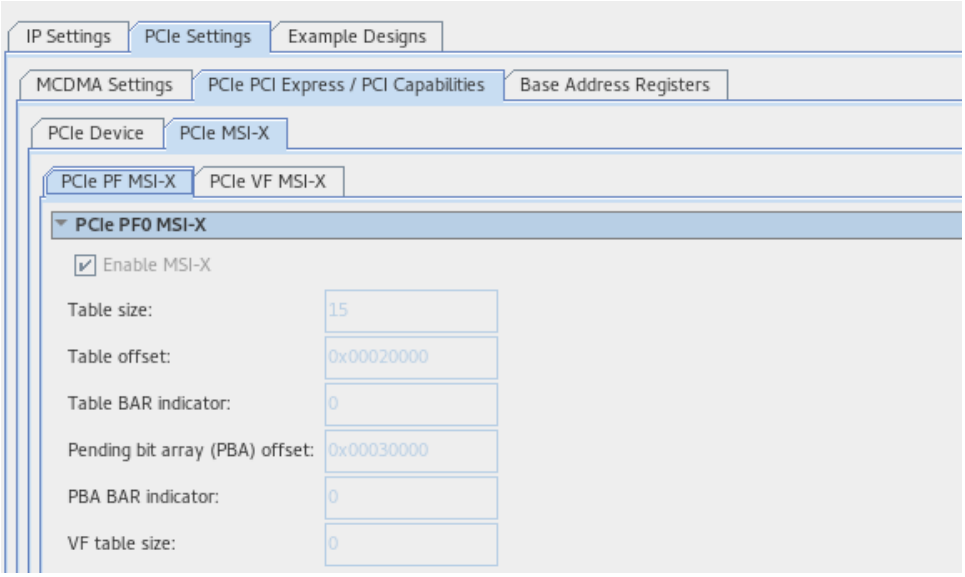


Table 25. PCIe PF MSI-X Parameters

Parameter	Value	Default Value	Description
Enable MSI-X	**On / Off**	See Description	Enables or disables the MSI-X capability. *Note:* In **Multichannel DMA**, **BAM+MCDMA** and **BAM+BAS+MCDMA** user modes, the MSI-X capability is automatically enabled, and the capability registers are set by the IP Parameter Editor. You cannot modify the capability register values.
Table Size	**0 - 1023**	**15**	Sets the number of entries in the MSI-X table. System software reads this field to determine the MSI-X table size N, which is encoded as N-1.
Table Offset	**0x00020000**	**0x00020000**	Sets the read-only base address of the MSI-X table. Points to the base of the MSI-X table. The lower 3 bits of the table BAR indicator (BIR) are set to zero by software to form a 64-bit qword-aligned offset.
Table BAR Indicator	**0**	**0**	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space.
Pending Bit Array (PBA) Offset	**0x00030000**	**0x00030000**	Used as an offset from the address contained in one of the function's Base Address registers to point to the base of the MSI-X PBA. The lower 3 bits of the PBA BIR are set to zero by software to form a 32-bit qword-aligned offset. This field is read-only after being programmed.
PBA BAR Indicator	**0**	**0**	Specifies the function Base Address register, located beginning at 0x10 in Configuration Space, that maps the MSI-X PBA into memory space. This field is read-only in the MSI-X Capability Structure.

PCIe VF MSI-X

The VF MSI-X capability is available in Multichannel DMA, BAM+MCDMA and BAM+BAS +MCDMA user modes if the SR-IOV support is enabled and total VFs are non-zero.

Figure 19. PCIe VF MSI-X Parameters

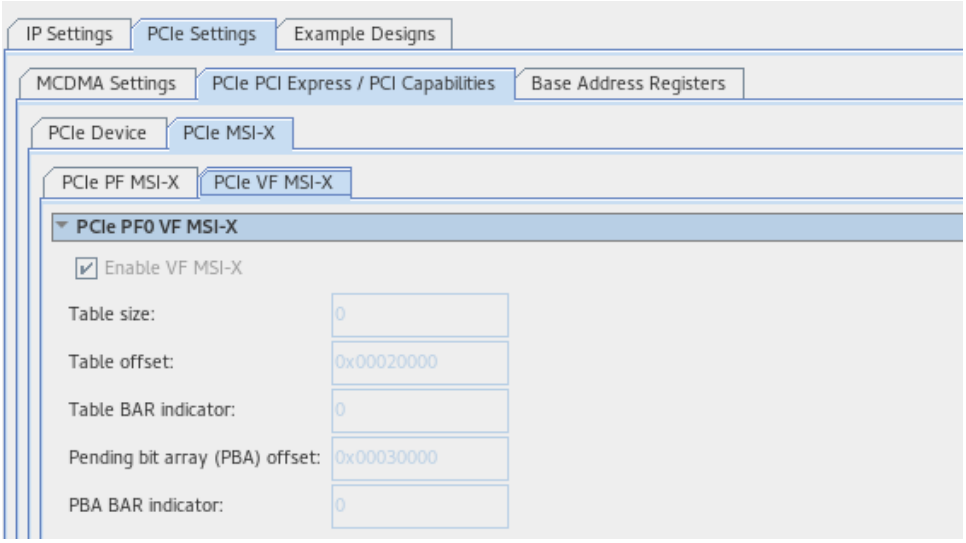
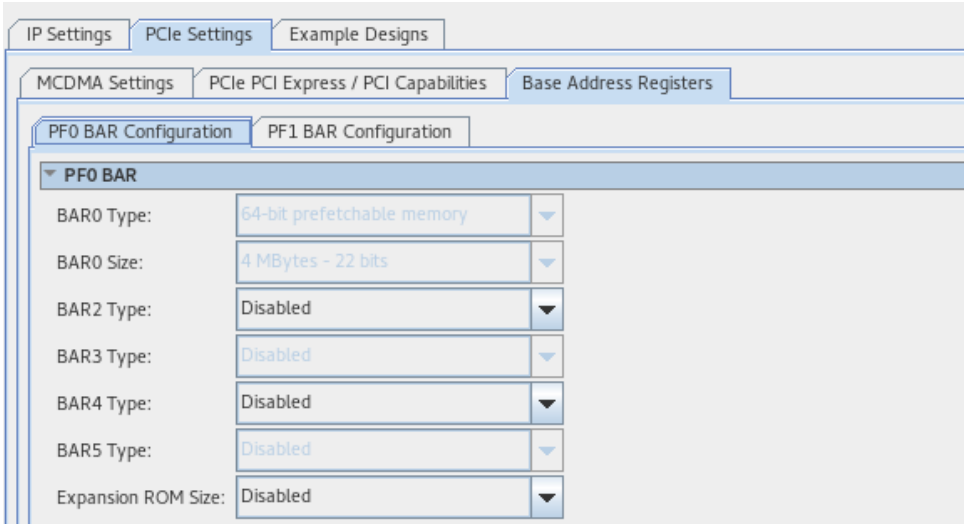


Table 26. PCIe VF MSI-X Parameters

Parameter	Value	Default Value	Description
Enable MSI-X	**On / Off**	See Description	Enables or disables the MSI-X capability. *Note:* In **Multichannel DMA**, **BAM+MCDMA** and **BAM+BAS+MCDMA** user modes, the MSI-X capability is automatically enabled, and the capability registers are set by the IP Parameter Editor. You cannot modify the capability register values.
Table Size	**0 - 1023**	**0**	Sets the number of entries in the MSI-X table. System software reads this field to determine the MSI-X table size N, which is encoded as N-1.
Table Offset	**0x00020000**	**0x00020000**	Sets the read-only base address of the MSI-X table. Points to the base of the MSI-X table. The lower 3 bits of the table BAR indicator (BIR) are set to zero by software to form a 64-bit qword-aligned offset.
Table BAR Indicator	**0**	**0**	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the MSI-X table into memory space.
Pending Bit Array (PBA) Offset	**0x00030000**	**0x00030000**	Used as an offset from the address contained in one of the function's Base Address registers to point to the base of the MSI-X PBA. The lower 3 bits of the PBA BIR are set to zero by software to form a 32-bit qword-aligned offset. This field is read-only after being programmed.
PBA BAR Indicator	**0**	**0**	Specifies the function Base Address register, located beginning at 0x10 in Configuration Space, that maps the MSI-X PBA into memory space. This field is read-only in the MSI-X Capability Structure.

3.2.2.3. Base Address Registers

Figure 20. PF BAR Configuration Parameters



The Base Address Registers settings are available for the Native Endpoint design. In Multichannel DMA, BAM+MCDMA and BAM+BAS+MCDMA modes, BAR0 is allocated to access a 4 MB space that internal IP registers are mapped to.

In Multichannel DMA mode, BAR2 size can be configured through the **BAR2 Address Width** parameter in **MCDMA Settings**.

Supported BAR types for the user application are as follows (vary by user mode):

- BAR 0/2/4: Disabled, 64-bit prefetchable memory, 64-bit non-prefetchable memory.
- BAR 1/3/5: Disabled.

If you select a 64-bit type for BAR 0/2/4, then the adjacent BAR 1/3/5 is disabled.

The following tables show the BAR and Expansion ROM parameters that can be configured in each user mode.

Table 27. PF BAR Configuration per User Mode [MCDMA, BAM, BAS]

Y = This BAR is available to the user application.

PF0 BAR	Multichannel DMA	Bursting Master	Bursting Slave
BAR0	64-bit prefetchable memory Size: 4 MB - 22 bits **Note:** BAR not available to user application.	Y	Y
BAR1	Disabled **Note:** BAR not available to user application.	Disabled	Disabled
BAR2	64-bit prefetchable memory Size: 4 MB - 22 bits **Note:** BAR not available to user application.	Y	Y
BAR3	Disabled	Disabled	Disabled

continued...

PF0 BAR	Multichannel DMA	Bursting Master	Bursting Slave
	Note: BAR not available to user application.		
BAR4	Disabled **Note:** BAR not available to user application.	Y	Y
BAR5	Disabled **Note:** BAR not available to user application.	Disabled	Disabled
Expansion ROM	Y	Y	Y

Table 28. PF BAR Configuration per User Mode [BAM+BAS, BAM+MCDMA, BAM+BAS +MCDMA]

Y = This BAR is available to the user application.

PF0 BAR	BAM+BAS	BAM+MCDMA	BAM+BAS+MCDMA
BAR0	Y	64-bit prefetchable memory Size: 4 MB - 22 bits *Note:* BAR not available to user application.	64-bit prefetchable memory Size: 4 MB - 22 bits *Note:* BAR not available to user application.
BAR1	Disabled	Disabled *Note:* BAR not available to user application.	Disabled *Note:* BAR not available to user application.
BAR2	Y	Y	Y
BAR3	Disabled	Disabled	Disabled
BAR4	Y	Y	Y
BAR5	Disabled	Disabled	Disabled
Expansion ROM	Y	Y	Y

3.3. Generating HDL for Simulation and Synthesis

1. Set the parameter values in the IP Parameter Editor and view the block diagram of the component. The **System Messages** tab at the bottom displays any errors in the IP parameters.
2. Click **Generate HDL**. The Generation dialogue box appears.

Figure 21. Generation Dialogue Box Options

Option	Description
Create HDL design files for synthesis	Allows you to specify **Verilog** or **VHDL** file type generation for the system's top-level definition and child instances. Select **None** to skip generation of synthesis files.
Create timing and resource estimates for each IP in your system to be used with third-party synthesis tools	Generates a non-functional Verilog Design File (`.v`) for use by supported third-party EDA synthesis tools. Estimates timing and resource usage for the IP component. The generated netlist file name is `<ip_component_name>_syn.v`.
Create Block Symbol File (.bsf)	Generates a Block Symbol File (`.bsf`) for use in a larger system schematic Block Diagram File (`.bdf`).
IP-XACT	Generates an IP-XACT file for the system, and adds the file to the IP Catalog. *Note:* Platform Designer supports importing and exporting files in IP-XACT 2009 format and exporting IP-XACT files in 2014 format.
Generate IP Core Documentation	Generates the IP user guide documentation for the components in your system (when available).
Create simulation model	Generates Verilog HDL or VHDL simulation models and simulator setup scripts. Enable the checkbox for one or more simulators to generate setup scripts for those tools in the following location: `<top-level system name>/<system name>/<sim>/<simulator>` If you do not specify a simulator, the setup scripts generate for all supported tools.
Clear output directories for selected generation targets	Clears previous synthesis and simulation file generation data for the current system.
Use multiple processors for faster IP generation (when available)	Disables or enables parallel IP generation for faster IP generation using multiple processors when available in your system.

3. Specify output file generation options, and then click **Generate**. This allows you to generate a GTS AXI MCDMA IP in the standalone mode with the IP synthesis and simulation files generated according to your specifications.
4. To generate a simulation testbench, click **Generate → Generate Testbench System** in the IP Parameter Editor. Specify testbench generation options, and then click **Generate**.
5. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate → Show Instantiation Template** in the IP Parameter Editor.
6. Click **Finish**. Click **Yes** if prompted to add files representing the IP variation to your project.
7. After generating and instantiating your IP variation, make appropriate pin assignments to connect the ports.

3.3.1. IP Core Generation Output

The Quartus Prime Pro Edition software generates the following output file structure for individual IP cores that are not part of a Platform Designer system.

Figure 22. IP Core Generation Output

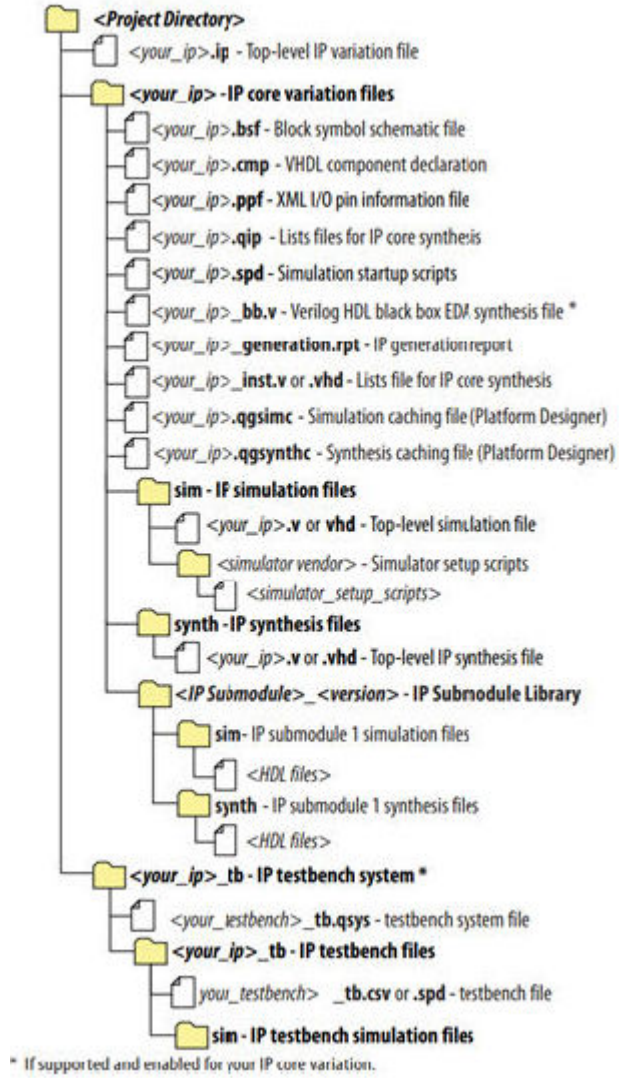


Table 29. Output Files of IP Generation

File Name	Description
<your_ip>.ip	Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Platform Designer system, the parameter editor also generates a .qsys file.
<your_ip>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.
<your_ip>_generation.rpt	IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.
<your_ip>.qgsimc (Platform Designer systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core.

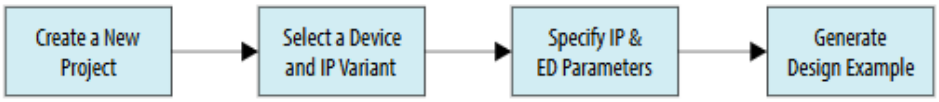
continued...

File Name	Description
	This comparison determines if Platform Designer can skip the regeneration of the HDL
<your_ip>.qgsynth (Platform Designer systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip the regeneration of the HDL.
<your_ip>.qip	Contains all information to integrate and compile the IP component.
<your_ip>.csv	Contains information about the upgrade status of the IP component.
<your_ip>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).
<your_ip>.spd	Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<your_ip>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<your_ip>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<your_ip>_inst.v or <your_ip>_inst.vhd	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<your_ip>.regmap	If the IP contains register information, the Quartus Prime Software generates the .regmap file. The .regmap file describes the register map information of master and slave interfaces. This file complements the .sopcinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
<your_ip>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system. During synthesis, the Quartus Prime Software stores the .svd files for slave interface visible to the System Console masters in the .sof file in the debug session. System Console reads this section, which Platform Designer queries for register map information. For system slaves, Platform Designer accesses the registers by name.
<your_ip>.v <your_ip>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
/mentor/	Contains a msim_setup.tcl script to set up and run a ModelSim® simulation.
/aldec/	Contains a Riviera-PRO* script rivierapro_setup.tcl to setup and run a simulation.
/synopsys/vcs/ /synopsys/vcsmx/	Contains a shell script vcs_setup.sh to set up and run a VCS* simulation. Contains a shell script vcsmx_setup.sh and synopsys_sim.setup file to set up and run a VCS* MX simulation.
/cadence/	Contains a shell script ncsim_setup.sh and other setup files to set up and run an NCSim simulation.
/submodules/	Contains HDL files for the IP core submodule.
/<IP submodule>/	Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.

3.4. Generating the Design Example

In the Quartus Prime Pro Edition software, you can generate a design example for the GTS AXI Multichannel DMA IP for PCI Express.

Figure 23. Design Example Generation

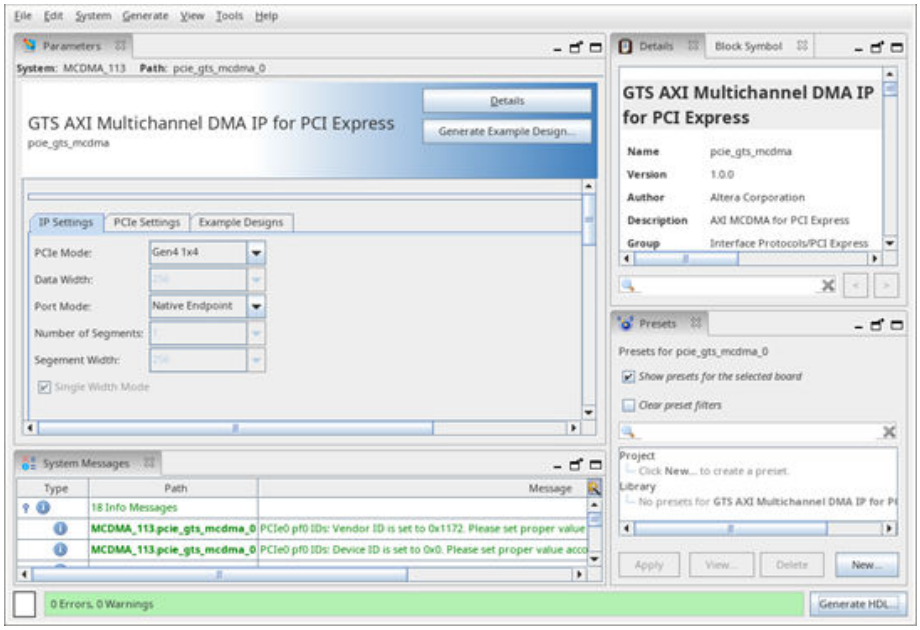


Following is the procedure to generate a design example:

1. In the Quartus Prime Pro Edition software, create a new project by clicking **File** → **New Project Wizard**. Click **Next**.
2. Select **Empty Project** type and specify the **Directory, Name, and Top-Level Entity**. Click **Next**.
3. Specify the **Family, Device & Board Settings** as follows:
   a. Select **Family**: Agilex 5 (E-Series/D-Series).
   b. Select **Target device**: A5ED065BB32AE4SR0.

      Note: The device used for the Quartus Prime-generated design example matches the device used in the target FPGA development kit board and may not be the same as the target device selected here.

   c. Click **Finish**.
4. Select from **Tools** → **IP Catalog** to open the IP Catalog.
5. Select **GTS AXI Multichannel DMA IP for PCI Express** (Library → Interface Protocols → PCI Express → GTS AXI Multichannel DMA IP for PCI Express) and then click **Add**.
6. In the **New IP Variant** dialogue box, specify a top-level name for your new custom IP variation and the directory for it. The IP Parameter Editor saves the IP variation settings in a file named <your_ip>.ip.
7. Click **Create**. The IP Parameter Editor appears as shown in the figure below.



8. Go to the **Example Designs** tab, and make the following selections:

a. For **Example Design Files**, the **Simulation** and **Synthesis** options are turned on by default. Leave the settings in default, unless you do not need the simulation or synthesis files, and to reduce the design example generation time.

   Note: The design example does not support simulation in this release. Turn off the **Simulation** option to reduce the design example generation time.

b. For **Generated HDL Format**, only Verilog is supported in the current release.

c. For **Target Development Kit**, select either the **Agilex 5 FPGA E-Series 065B Modular Development Kit (ES1)** or **NONE** to target the device selected for your current Quartus Prime software project.

   Note: If you select the development kit, the settings including the target device, pin assignments are included in the .qsf file of the generated design example, and you are not required to add them manually. These settings are board-specific for the development kit.

   Note: **Agilex 5 FPGA E-Series 065B Modular Development Kit** option is not supported for the Gen4 1x4 mode design example in the current release.

d. For **Currently Selected Example Design**, only the **AXI-S Device-side Packet Loopback** variant is supported in this release.

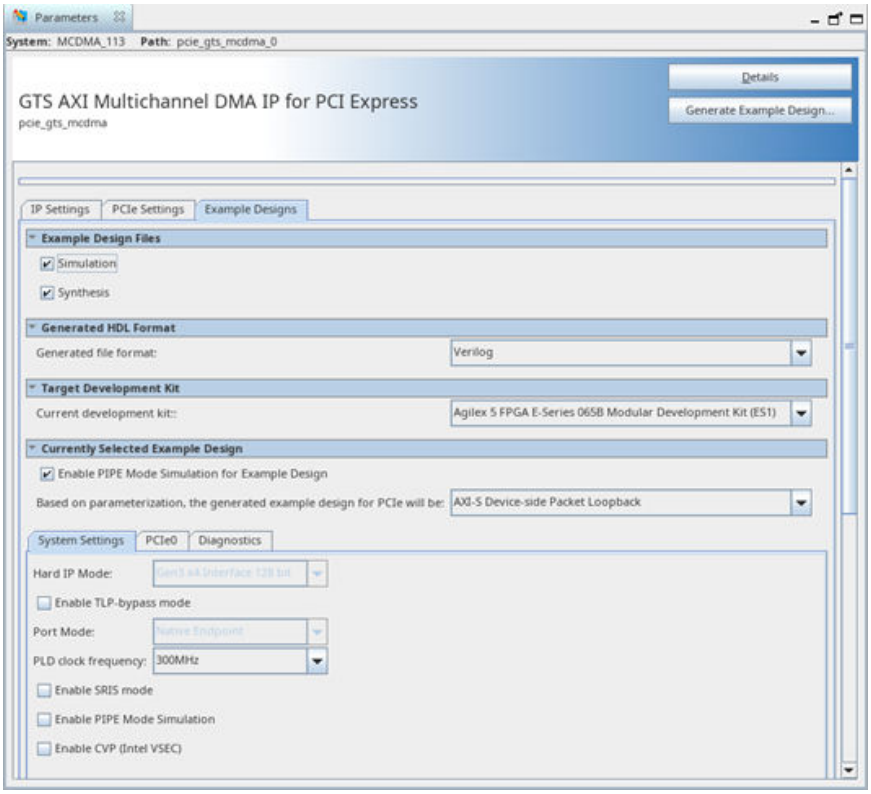


Table 30. Design Example Parameters

Parameter	Value	Default Value	Description
Simulation	**On / Off**	**On**	Design example simulation is not supported in the current release.
Synthesis		**On**	When the **Synthesis** box is checked, all necessary file sets for synthesis are generated. When unchecked, only the Platform Designer system is generated.
Generated file format	**Verilog**	**Verilog**	HDL format.
Current development kit	**Agilex 5 FPGA E-Series 065B Modular Development Kit (ES1)** **None**	**Agilex 5 FPGA E-Series 065B Modular Development Kit (ES1)**	Selects a target FPGA development kit board. *Note:* If an FPGA development board is selected, the target device used for generation is the one that matches the device on the development kit board. *Note:* **Agilex 5 FPGA E-Series 065B Modular Development Kit** option is not supported for the Gen4 1x4 mode design example in the current release.
Enable PIPE Mode Simulation for Example Design	**On / Off**	**On**	When selected, PIPE mode simulation is enabled.
Currently selected example design	**AXI-S Device-side Packet Loopback**	**AXI-S Device-side Packet Loopback**	Selects example design variant. Only the AXI-S Device-side Packet Loopback example design is supported in the current release. *Note:* • Design examples for BAM, BAS, BAM+BAS, BAM+MCDMA, and BAM +BAS+MCDMA modes are not supported. • Design example for Root Port mode is not supported.

Note: Design example generation includes the GTS AXI Streaming Intel FPGA IP for PCI Express (intel_pcie_gts). Hence, the **Example Designs** tab includes the GTS AXI Streaming IP Parameter Editor. Some of the PCIe Interface parameters are automatically configured per the GTS AXI MCDMA IP parameter values and greyed out.

Table 31. PCIe Interface Settings Parameters

Parameter	Value	Default Value	Description
Hard IP Mode	**Gen4 x4 Interface 256-bit** **Gen3 x4 Interface 128-bit**	**Gen4 x4 Interface 256-bit**	Selects the PCIe Hard IP mode. *Note:* The ode value is passed from the GTS AXI MCDMA IP settings.
Enable TLP-Bypass Mode	**On / Off**	**Off**	Enables the TLP Bypass feature for the GTS AXI Streaming IP. *Note:* The TLP Bypass design example is not supported.
Port Mode	**Root Port** **Native Endpoint**	**Native Endpoint**	Selects a PCIe mode for the GTS AXI Streaming IP. *Note:* The mode value is passed from the AXI MCDMA IP Settings. *Note:* The Root Port design example is not supported.

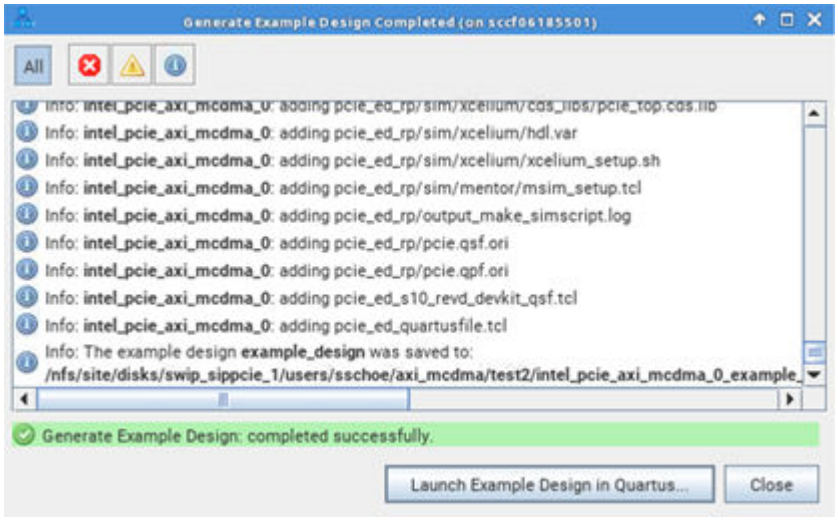
continued...

Parameter	Value	Default Value	Description
PLD Clock Frequency	**Gen4 1x4: 350/300/250/200 MHz** **Gen3 1x4: 300/250/200 MHz**	**300 MHz**	Selects the PLD clock frequency.
Enable SRIS Mode	**Off**	**Off**	Enables the Separate Reference Clock with Independent Spread Spectrum Clocking (SRIS) feature.
Enable PIPE Mode Simulation	**On / Off**	**Off**	When on, this parameter enables the PIPE mode simulation.
Enable CVP (Intel VSEC)	**On / Off**	**Off**	Enables the CVP feature for the GTS AXI Streaming IP.

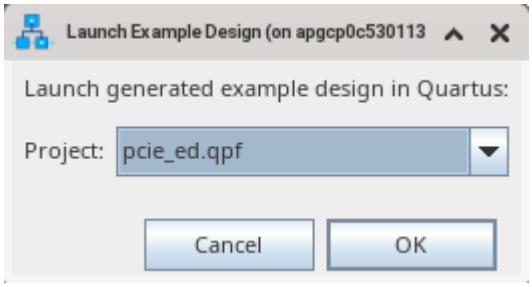
9. Click on **Generate Example Design** to generate the design example variant. When the prompt asks you to specify the directory for your design example, you can accept the default directory, `/pcie_gts_mcdma_0_example_design`, or choose another directory. Then, click **OK** to kick off the design example generation.

3.5. Compiling the Design Example

1. After the design example generation is completed, click **Launch Example Design in Quartus**.



2. Select `pcie_ed.qpf` and click **OK** to open the newly generated design example in the Quartus Prime Pro Edition software.



3. In the Quartus Prime Pro Edition software, click **Processing** → **Start Compilation** to compile the design example and generate the bitstream (.sof) file.

For more information on the example design, refer to the Design Example Overview section.

For more information on how to run the example design on Hardware, refer to the chapter Validating the IP.

4. Integrating the IP With Your Application

This chapter describes the integration of the GTS AXI MCDMA IP with your application. The focus is mainly on the signal interfaces that the GTS AXI MCDMA IP uses to communicate with the GTS AXI Streaming IP for PCI Express and user logic.

4.1. Overview

The following is a system block diagram of the GTS AXI MCDMA IP where one side of the IP interfaces with the GTS AXI Streaming IP and the other interfaces with the user logic. The following sections describe these interfaces in detail.

Figure 24. GTS AXI MCDMA IP System Block Diagram

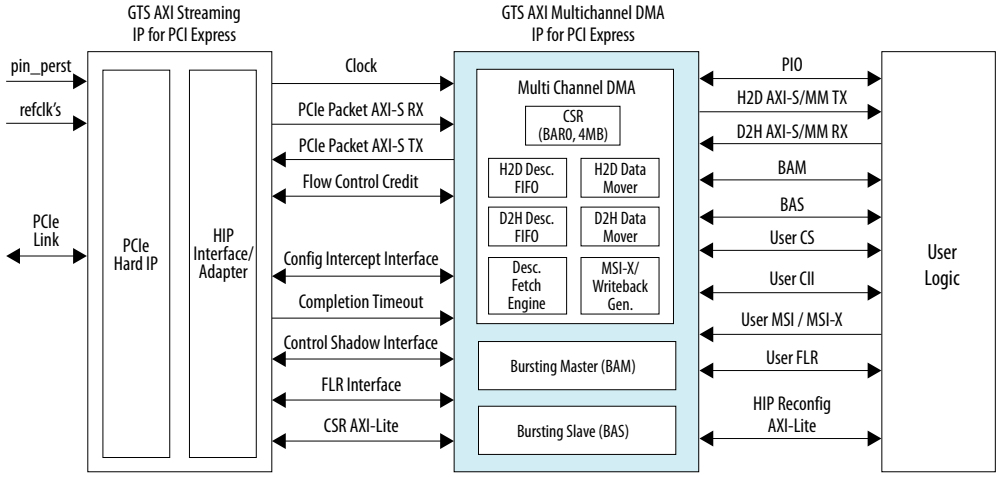
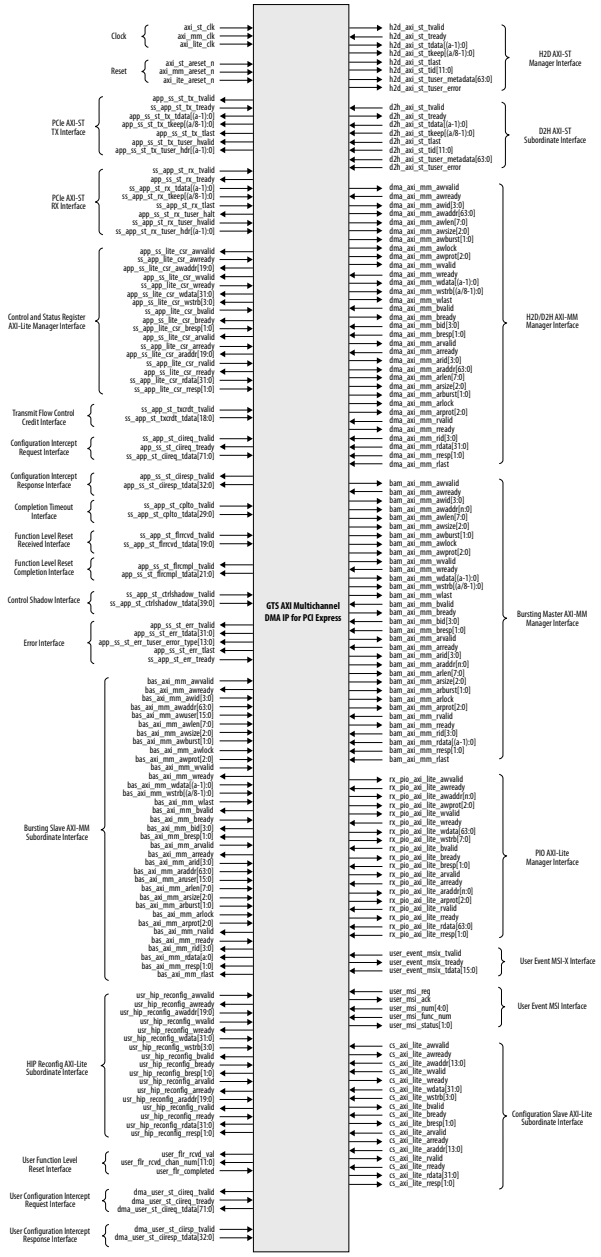


Figure 25. GTS AXI MCDMA IP Port List



a = 128 (Gen3 1x4); 256 (Gen4 1x4)

4.2. Implementing Required Clocking

This section describes the required clock connections and clock signals for the GTS AXI MCDMA IP. The IP requires three clock inputs:

- `axi_st_clk`
- `axi_mm_clk`
- `axi_lite_clk`

You can drive both the `axi_st_clk` and `axi_mm_clk` input clocks from the `coreclkout_hip_toapp` output clock of the GTS AXI Streaming IP. `axi_lite_clk` can be driven from any 100-250 MHz clock source available in your design.

Figure 26. Clock Connections of the GTS AXI MCDMA IP

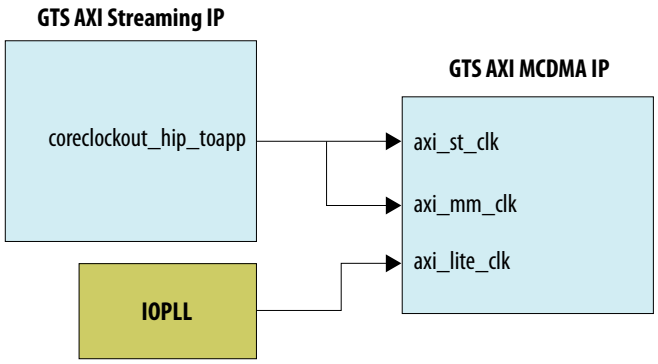


Table 32. Clock Signals

Clock Name	Direction	Description
`axi_st_clk`	Input	Global clock signal for the AXI Streaming interface. All AXI Streaming signals are sampled on the rising edge of this clock. This clock is typically derived from the `coreclkout_hip_toapp` output of the GTS AXI Streaming IP. Gen4x4: 350/300/250/200 MHz Gen3x4: 250/200 MHz
`axi_mm_clk`	Input	Global clock signal for the AXI Memory-Mapped interface. All AXI Memory-Mapped signals are sampled on the rising edge of this clock. This clock is typically derived from the `coreclkout_hip_toapp` output of the GTS AXI Streaming IP. Gen4x4: 350/300/250/200 MHz Gen3x4: 250/200 MHz
`axi_lite_clk`	Input	Global clock signal for AXI-Lite interface. All AXI-Lite signals are sampled on the rising edge of this clock. This clock drives the control and status register interfaces in the design. Frequency: 100-250 MHz

4.3. Implementing Required Resets

The GTS AXI MCDMA IP consists of three types of reset ports to reset the AXI-S, AXI-MM and AXI-Lite interfaces respectively. Altera recommends asserting resets during the initial device bring-up and PCIe cold/warm reset cycles.

Table 33. Reset Signals

Reset Signal Name	Direction	Description
axi_st_areset_n	Input	Active-low reset signal for the AXI Streaming interface.
axi_mm_areset_n	Input	Active-low reset signal for the AXI Memory-Mapped interface.
axi_lite_areset_n	Input	Active-low reset signal for the AXI-Lite interface.

4.4. Connecting the GTS AXI Streaming IP-Facing Interfaces

This section describes the required connections and interface signals between the GTS AXI MCDMA IP and GTS AXI Streaming IP. Ensure you apply IP settings that are aligned with the GTS AXI Streaming IP so that interface signals of both IPs are compatible and can be connected correctly. Refer to Aligning IP Settings with the GTS AXI Streaming IP for PCI Express for more information.

4.4.1. PCIe AXI-Stream TX Interface (ss_tx_st)

The outbound packet towards the link is transmitted through this PCIe AXI-Stream TX interface with separate header and data interfaces. These interfaces support single segments with data widths of 16 bytes (128 bits) and 32 bytes (256 bits). These interfaces should be connected to the corresponding AXI-Stream TX interface of the GTS AXI Streaming IP.

For information on this interface, refer to AXI4-Stream Transmit (TX) Interface.

Interface clock: axi_st_clk

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Table 34. PCIe AXI-Stream TX Interface

Signal Name	Direction	Description
app_ss_st_tx_tvalid	Output	Indicates that the source is driving a valid transfer.
ss_app_st_tx_tready	Input	Indicates that the sink can accept a transfer in the current cycle.
app_ss_st_tx_tdata[DWIDTH-1:0]	Output	Data bus used to provide the data that is passing across the interface.
app_ss_st_tx_tkeep[DWIDTH/8-1:0]	Output	A byte qualifier used to indicate whether the content of the associated byte is valid. The invalid bytes are allowed only during the app_ss_st_tx_tlast cycle. **Note:** Sparse tkeep is not supported.
app_ss_st_tx_tlast	Output	Indicates End of Data/Command Transmission.
app_ss_st_tx_tuser_hvalid	Output	Indicates the tuser_hdr is valid in the current cycle.
app_ss_st_tx_tuser_hdr[DWIDTH-1:0]	Output	Carries header format for the current cycle of data transfer. For the bit positions and mapping, refer to Header Format.

4.4.2. PCIe AXI-Stream RX Interface (ss_rx_st)

The packet from the link is received on this interface with separate header and data interfaces. These interfaces support single segments with data widths of 16 bytes (128 bits) and 32 bytes (256 bits). These interfaces should be connected to the corresponding AXI-Stream RX interface of the GTS AXI Streaming IP.

For information of this interface, refer to AXI4-Stream Receive (RX) Interface.

Interface clock: `axi_st_clk`

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Table 35. PCIe AXI-Stream RX Interface

Signal Name	Direction	Description
`ss_app_st_rx_tvalid`	Input	Indicates that the source is driving a valid transfer.
`app_ss_st_rx_tready`	Output	Indicates that the sink can accept a transfer in the current cycle.
`ss_app_st_rx_tdata[DWIDTH-1:0]`	Input	Data bus used to provide the data that is passing across the interface.
`ss_app_st_rx_tkeep[DWIDTH/8-1:0]`	Input	A byte qualifier used to indicate whether the content of the associated byte is valid. The invalid bytes are allowed only during the `ss_app_st_rx_tlast` cycle. **Note:** Sparse tkeep is not supported.
`ss_app_st_rx_tlast`	Input	Indicates End of Data/Command Transmission.
`ss_app_st_rx_tuser_hvalid`	Input	Indicates the `ss_app_st_rx_tuser_hdr` is valid in the respective segment.
`ss_app_st_rx_tuser_hdr[DWIDTH-1:0]`	Input	Carries the header format for the current cycle of data transfer.

4.4.3. Control and Status Register Interface (ss_csr_lite)

The AXI4-Lite Manager interface provides access to PCIe configuration space registers of all Functions as well as soft register space registers implemented in the GTS AXI Streaming IP. This interface does not differentiate between non-secure and secure accesses. All accesses are considered secure. This interface should be connected to the corresponding Control and Status Register Responder Interface of the GTS AXI Streaming IP.

For information of this interface, refer to Control and Status Register Responder Interface.

Interface clock: `axi_lite_clk`

Table 36. Control and Status Register Interface

Signal Name	Direction	Description
Write Address Channel		
app_ss_lite_csr_awvalid	Output	Indicates that the write address channel signals are valid.
ss_app_lite_csr_awready	Input	Indicates that a transfer on the write address channel can be accepted.
app_ss_lite_csr_awaddr[19:0]	Output	The address of the first transfer in a write transaction.
Write Data Channel		
app_ss_lite_csr_wvalid	Output	Indicates that the write data channel signals are valid.
ss_app_lite_csr_wready	Input	Indicates that a transfer on the write data channel can be accepted.
app_ss_lite_csr_wdata[31:0]	Output	Write data.
app_ss_lite_csr_wstrb[3:0]	Output	Write strobes, indicate which byte lanes hold valid data.
Write Response Channel		
ss_app_lite_csr_bvalid	Input	Indicates that the write response channel signals are valid.
app_ss_lite_csr_bready	Output	Indicates that a transfer on the write response channel can be accepted.
ss_app_lite_csr_bresp[1:0]	Input	Write response, indicates the status of a write transaction.
Read Address Channel		
app_ss_lite_csr_arvalid	Output	Indicates that the read address channel signals are valid.
ss_app_lite_csr_arready	Input	Indicates that a transfer on the read address channel can be accepted.
app_ss_lite_csr_araddr[19:0]	Output	The address of the first transfer in a read transaction.
Read Data Channel		
ss_app_lite_csr_rvalid	Input	Indicates that the read data channel signals are valid.
app_ss_lite_csr_rready	Output	Indicates that a transfer on the read data channel can be accepted.
ss_app_lite_csr_rdata[31:0]	Input	Read data.
ss_app_lite_csr_rresp[1:0]	Input	Read response, indicates the status of a read transfer.

4.4.4. Transmit Flow Control Credit Interface (ss_txcrdt)

Before a TLP can be transmitted, flow control logic verifies that the link partner's RX port has sufficient buffer space to accept it. The TX Flow Control interface reports the link partner's available RX buffer space to the GTS AXI MCDMA IP. The interface provides posted, non-posted, completion data, and header credit information. One data credit is equal to four dwords (DWs) and one header credit is equal to the maximum size header plus the optional digest field. The credits are advertised as the limit value as specified in the PCIe specification. This interface should be connected to the corresponding Transmit Flow Control Credit Interface of the GTS AXI Streaming IP.

For information on this interface, refer to Transmit Flow Control Credit Interface.

Interface clock: `axi_st_clk`

Table 37. Transmit Flow Control Credit Interface

Signal Name	Direction	Description
ss_app_st_txcrdt_tvalid	Input	Indicates that the credit information on tdata is valid.
ss_app_st_txcrdt_tdata[18:0]	Input	Carries the credit limit information and type of credit. [15:0]: Credit Limit Value • All zeros indicates infinite credit for P, NP, CPL. [18:16]: Credit Type • 3'b000 - Posted Header Credit • 3'b001 - Non-Posted Header Credit • 3'b010 - Completion Header Credit • 3'b011 - Reserved • 3'b100 - Posted Data Credit • 3'b101 - Non-Posted Data Credit • 3'b110 - Completion Data Credit • 3'b111 - Reserved

4.4.5. Configuration Intercept Interface (CII)

The Configuration Intercept Interface (CII) of the GTS AXI Streaming IP allows the application logic to detect the occurrence of a Configuration (CFG) request on the link and to modify the request. This interface propagates the CFG request to the User CII Interface and forwards the user response data to the GTS AXI Streaming IP if the User CII Interface is enabled. This interface should be connected to the corresponding Configuration Intercept Interface of the GTS AXI Streaming IP.

For information on this interface, refer to Configuration Intercept Interface.

4.4.5.1. Configuration Intercept Request Interface (ss_cii_req)

Interface clock: axi_lite_clk

Table 38. Configuration Intercept Request Interface

Signal Name	Direction	Description
ss_app_st_ciireq_tvalid	Input	When asserted, indicates a valid CFG request cycle is waiting to be intercepted. Deasserted when app_ss_st_ciireq_tready is asserted.
app_ss_st_ciireq_tready	Output	Application asserts this signal for one clock to acknowledge ss_app_st_ciireq_tvalid is seen.
ss_app_st_ciireq_tdata[0]	Input	hdr_poisoned: The poisoned bit in the received TLP header on the CII.
ss_app_st_ciireq_tdata[4:1]	Input	hdr_first_be: The first dword byte enable field in the received TLP header on the CII.
ss_app_st_ciireq_tdata[9:5]	Input	Reserved
ss_app_st_ciireq_tdata[12:10]	Input	func_num: The PF number in the received TLP header on the CII.
ss_app_st_ciireq_tdata[23:13]	Input	vf_num: The child VF number of parent PF in the received TLP header on the CII.
ss_app_st_ciireq_tdata[24]	Input	vf_active: Indicates VF number is valid in the received TLP header on the CII.

continued...

Signal Name	Direction	Description
ss_app_st_ciireq_tdata[25]	Input	wr: Indicates a configuration write request detected in the received TLP header on the CII. Also, indicates that ss_app_st_ciireq_tdata[67:36] is valid.
ss_app_st_ciireq_tdata[35:26]	Input	addr: The double word register address in the received TLP header on the CII.
ss_app_st_ciireq_tdata[67:36]	Input	dout: Received TLP payload data from the link partner to your application client. The data is in little endian format. The first received payload byte is in [43:36].
ss_app_st_ciireq_tdata[71:68]	Input	Reserved

4.4.5.2. Configuration Intercept Response Interface (ss_cii_resp)

This interface is meant to provide the response for any request received on the **ss_cii_req** interface. The response data is qualified by `tvalid`.

Interface clock: `axi_lite_clk`

Table 39. Configuration Intercept Response Interface

Signal Name	Direction	Description
app_ss_st_ciiresp_tvalid	Output	This signal will be asserted for one clock to indicate that valid data is driven on the `app_ss_st_ciiresp_tdata` bus.
app_ss_st_ciiresp_tdata[31:0]	Output	Override data for the intercepted configuration request on the Configuration Intercept Request interface. For CfgWr: Override the write data to the Configuration register. For CfgRd: Override the data payload of the completion TLP.
app_ss_st_ciiresp_tdata[32]	Output	Override Data Enable: when asserted, the CfgWr payload or CfgRd completion will be overridden by the data supplied on the `app_ss_st_ciiresp_tdata[31:0]` bus.

4.4.6. Completion Timeout Interface (ss_cplto)

When a completion timeout happens in the GTS AXI Streaming IP, the information is shared with the GTS AXI MCDMA IP through this interface. The interface provides the function number and tag number of the outstanding request that was timed out. This interface should be connected to the corresponding Completion Timeout Interface of the GTS AXI Streaming IP.

For information on this interface, refer to Completion Timeout Interface.

Interface clock: `axi_lite_clk`

Table 40. Completion Timeout Interface

Signal Name	Direction	Description
ss_app_st_cplto_tvalid	Input	Indicates that a completion timeout is received for an outstanding non-posted request.
ss_app_st_cplto_tdata[29:0]	Input	Carries completion timeout information. [9:0] - Tag Number

continued...

Signal Name	Direction	Description
		[12:10] - PF Number, indicates parent PF number of VF when VF Active is high, else PF Number of function. [23:13] - VF Number, indicates VF number when VF Active is high. [24] - VF Active, indicates timeout is for VF. [31:25]: Reserved. [43:32]: Transfer length in bytes (least significant 12-bits) of the expected completion that timed out for the non-posted transaction. For a split completion, it indicates the number of bytes remaining to be delivered when the completion timed out (the maximum length is the maximum read request size. Example: 4K Bytes = 2^12 bytes). [46:44]: Traffic class of the expected completion that timed out for the non-posted transaction. [48:47]: Attribute of the expected completion that timed out for the non-posted transaction. ID based ordering is not supported. • [47]: No snoop. • [48]: Relaxed ordering.

4.4.7. Function Level Reset (FLR) Interface

The FLR (Function Level Reset) interface is used to reset individual Single Root I/O Virtualization (SR-IOV) functions, for both Physical Functions (PFs) and Virtual Functions (VFs). When an FLR is executed for a specific VF, the packets received for that VF become invalid. The assertion of the `ss_app_st_flrrcvd_tvalid` signal indicates that an FLR has been received for a particular PF or VF. This FLR is then propagated to the User FLR interface for user logic to execute its FLR routine and report the completion status back to the Function Level Reset Completion interface.

This interface should be connected to the corresponding Function Level Reset Interface of the GTS AXI Streaming IP.

For information on this interface, refer to Function Level Reset Interface.

4.4.7.1. FLR Received Interface (ss_flrrcvd)

Interface clock: `axi_lite_clk`

Table 41. FLR Received Interface

Signal Name	Direction	Description
`ss_app_st_flrrcvd_tvalid`	Input	When asserted, indicates a FLR request received from the Host. The signal is valid for one clock cycle.
`ss_app_st_flrrcvd_tdata[19:0]`	Input	Valid when `ss_app_st_flrrcvd_tvalid` is asserted. [2:0] - The PF number of the FLR Request. [13:3] - Indicates child VF number of parent PF indicated by PF number. [14] - Indicates request is for Virtual Function implemented in controller's Physical Function. [19:15] – Reserved.

4.4.7.2. FLR Completion Interface (ss_flrcmpl)

Interface clock: `axi_lite_clk`

Table 42. FLR Completion Interface

Signal Name	Direction	Description
app_ss_st_flrcmpl_tvalid	Output	When asserted, indicates a FLR request is completed. The signal is valid for one clock cycle.
app_ss_st_flrcmpl_tdata [21:0]	Output	Valid when app_ss_st_flrcmpl_tvalid is asserted. [2:0] - The PF number of the FLR Completion. [13:3] - Indicates child VF number of parent PF indicated by the PF number. [14] - Indicates completion is from the Virtual Function implemented in the controller's Physical Function. [21:15] – Reserved.

4.4.8. Control Shadow Interface (ss_ctrlshadow)

The control shadow interface brings out the settings of the various configuration register fields of the function. The interface provides update to primary control signals only. This interface should be connected to the corresponding Control Shadow Interface of the GTS AXI Streaming IP.

For information of this interface, refer to Control Shadow Interface.

Interface clock: axi_lite_clk

Table 43. Control Shadow Interface

Signal Name	Direction	Description
ss_app_st_ctrlshadow_tvalid	Input	This signal is asserted for one clock cycle when there is an update to the register fields being monitored due to a Configuration Write performed by the Root Complex. The new settings of the register fields are available on the tdata bus.
ss_app_st_ctrlshadow_tdata[39:0]	Input	When ss_app_st_ctrlshadow_tvalid is asserted, this signal provides the current settings of the register fields of the associated Function. [2:0] - Identifies the Physical Function Number of configuration register. [13:3] - Identifies the Virtual Function Number of the configuration register. [14] - Indicates information is for the Virtual Function implemented in the slot's Physical Function. [19:15] - Identifies the slot Number of the configuration register. [20] - Bus Master Enable [21] - MSI-X Mask [22] - MSI-X Enable [23] - Mem Space Enable [24] - ExpRom Enable [25] - TPH Req Enable [26] - ATS Enable [27] - MSI Enable [28] - MSI Mask [29] - Extended Tag [30] - 10 Bit Tag Req Enable [31] - PTM Enable [34:32] - MPS Size [37:35] - MRRS Size

continued...

Signal Name	Direction	Description
		[38] - VF Enable [39] - Page Request Enable

4.4.9. Error Interface

This interface allows the GTS AXI MCDMA IP to report application errors to the GTS AXI Streaming IP. This interface should be connected to the corresponding Error Interface of the GTS AXI Streaming IP.

For information of this interface, refer to Error Interface.

Interface clock: `axi_lite_clk`

Table 44. Error Interface

Signal Name	Direction	Description
`app_ss_st_err_tvalid`	Output	When asserted, indicates the MCDMA IP is reporting an error.
`app_ss_st_err_tdata[31:0]`	Output	Has the function number information, 128-bit header and 32-bit TLP prefix over multiple cycles (32 bits of information are sent in each clock cycle). Cycle 1: Carries the following information: • Bit[0]: Reserved • Bit[5:1]: PF Number of function • Bit[16:6]: Reserved • Bit[17]: Indicates TLP Header follows in subsequent cycles. • Bit[18]: Indicates TLP Header Prefix field follows in subsequent cycles. • Bit[31:19] : Reserved Cycle 2: TLP header[31:0] Cycle 3: TLP header[63:32] Cycle 4: TLP header[95:64] Cycle 5: TLP header[127:96] Cycle 6: TLP prefix Depending on Bit[17] and Bit[18] of cycle 1, tdata is valid for 1/5/6 cycles.
`app_ss_st_err_tuser_error_type[13:0]`	Output	Indicates the error type: • Bit[0]: Malformed TLP • Bit[1]: Receiver overflow • Bit[2]: Unexpected completion • Bit[3]: Completer abort • Bit[4]: Completion timeout • Bit [5]: Unsupported request • Bit[6]: Poisoned TLP received • Bit[7]: AtomicOp egress blocked • Bit[8]: Uncorrectable internal error • Bit[9]: Correctable internal error • Bit[10]: Advisory error

continued...

Signal Name	Direction	Description
		• Bit[11]: TLP prefix blocked • Bit[12]: ACS violation • Bit[13]: ECRC check failed
app_ss_st_err_tlast	Output	Indicates the last cycle of tdata transfer. tlast is asserted on 1st/5th/6th cycle of tdata depending on Bit[17] and Bit[18] of tdata in cycle 1.
app_st_err_tready	Input	When de-asserted, this signal indicates back-to-back outputs cannot be processed.

4.5. Connect User Logic-Facing Interfaces

This section describes the required connections and interface signals between the GTS AXI MCDMA IP and your application logic. Ensure the application logic interface signals are compatible with the GTS AXI MCDMA IP, including the number of signals and signal widths, and can be connected correctly.

4.5.1. H2D AXI-Stream Manager (h2d_st_initatr)

The H2D AXI Manager interface is used to send H2D DMA data to the external AXI-Stream Subordinate logic. This interface is available by selecting the AXI-S **User Interface** type in the IP Parameter Editor. This interface should be connected to the corresponding AXI-Stream Subordinate Interface of the application logic.

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Interface clock: `axi_st_clk`

Table 45. H2D AXI-Stream Manager Interface

Signal Name	Direction	Description
h2d_axi_st_tvalid	Output	Indicates that the manager interface is driving a valid transfer.
h2d_axi_st_tready	Input	Indicates that the sink can accept a transfer in the current cycle.
h2d_axi_st_tdata[DWIDTH-1:0]	Output	Data interface.
h2d_axi_st_tkeep[DWIDTH/8-1:0]	Output	A byte qualifier used to indicate whether the content of the associated byte is valid. The invalid bytes are allowed only during the h2d_axi_st_tlast cycle. Note: sparse tkeep is not allowed.
h2d_axi_st_tlast	Output	Indicates end of data transmission.
h2d_axi_st_tid[11:0]	Output	Stream ID, indicates channel number.
h2d_axi_st_tuser_metadata[63:0]	Output	Descriptor 8-byte metadata. Available only when metadata support is enabled through the IP Parameter Editor.
h2d_axi_st_tuser_error	Output	Error Status.

4.5.2. D2H AXI-Stream Subordinate (d2h_st_respndr)

The D2H AXI-Stream Subordinate interface is used to sink D2H DMA data from the external AXI-Stream Manager logic. This interface is available by selecting the AXI-S **User Interface** type in the IP Parameter Editor. This interface should be connected to the corresponding AXI-Stream Manager Interface of the application logic.

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Interface clock: `axi_st_clk`

Table 46. D2H AXI-Stream Subordinate Interface

Signal Name	Direction	Description
`d2h_axi_st_tvalid`	Input	Indicates that the manager interface is driving a valid transfer
`d2h_axi_st_tready`	Output	Indicates that the subordinate interface can accept a transfer in the current cycle.
`d2h_axi_st_tdata[DWIDTH-1:0]`	Input	Data interface.
`d2h_axi_st_tkeep[DWIDTH/8-1:0]`	Input	A byte qualifier used to indicate whether the content of the associated byte is valid. The invalid bytes are allowed only during the `d2h_axi_st_tlast` cycle. Note: sparse tkeep is not allowed.
`d2h_axi_st_tlast`	Input	Indicates end of data transmission.
`d2h_axi_st_tid[11:0]`	Input	Stream ID, indicates channel number.
`d2h_axi_st_tuser_metadata[63:0]`	Input	Descriptor 8-byte metadata. Available only when metadata support is enabled through the IP Parameter Editor.
`d2h_axi_st_tuser_error`	Input	Error Status.

4.5.3. H2D/D2H AXI-MM Manager (dma_mm_initatr)

The AXI-MM Manager interface is used to send H2D DMA data and sink D2H DMA data from the external AXI-MM Subordinate logic. The H2D path is connected to write channels while the D2H path is connected to read channels of the interface. This interface is available by selecting the AXI-MM **User Interface** type in the IP Parameter Editor. This interface should be connected to the corresponding AXI-MM Subordinate Interface of the application logic.

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Interface clock: `axi_mm_clk`

Table 47. H2D/D2H AXI-MM Manager Interface

Signal Name	Direction	Description
Write Address Channel		
dma_axi_mm_awvalid	Output	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
dma_axi_mm_awready	Input	Write address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals.
dma_axi_mm_awid[3:0]	Output	Write address ID. This signal is the identification tag for the write address group of signals. The default value is 0.
dma_axi_mm_awaddr[63:0]	Output	Write address. The write address gives the address of the first transfer in a write burst transaction.
dma_axi_mm_awlen[7:0]	Output	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. Maximum burst is 512B.
dma_axi_mm_awsize[2:0]	Output	Burst size. This signal indicates the size of each transfer in the burst.
dma_axi_mm_awburst[1:0]	Output	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated.
dma_axi_mm_awlock	Output	Lock type. This signal is tied to '0'.
dma_axi_mm_awprot[2:0]	Output	Protection type. This interface does not use protection attributes.
Write Data Channel		
dma_axi_mm_wvalid	Output	Write Data Valid. This signal indicates that the write data is valid.
dma_axi_mm_wready	Input	Write Data Ready. This signal indicates that the receiver can accept write data.
dma_axi_mm_wdata[DWIDTH-1:0]	Output	Write data.
dma_axi_mm_wstrb[DWIDTH/8-1:0]	Output	Write strobes. This signal indicates which byte lanes hold valid data.
dma_axi_mm_wlast	Output	Write last. This signal indicates the final transfer in a write burst.
Write Response Channel		
dma_axi_mm_bvalid	Input	Write Response Valid. This signal indicates that the write response is valid.
dma_axi_mm_bready	Output	Write Response Ready. This signal indicates that a transfer of the write response channel can be accepted.
dma_axi_mm_bid[3:0]	Input	Response ID. This signal is the ID tag of the write response.

continued...

Signal Name	Direction	Description
dma_axi_mm_bresp[1:0]	Input	Write Response. This signal indicates the status of a write transaction.
Read Address Channel		
dma_axi_mm_arvalid	Output	Read address valid. This signal indicates that the channel is signaling valid read address and control information.
dma_axi_mm_arready	Input	Read address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals
dma_axi_mm_arid[3:0]	Output	Read address ID. This signal is the identification tag for the read address group of signals. The default value is 0.
dma_axi_mm_araddr[63:0]	Output	Read address. The read address gives the address of the first transfer in a read burst transaction.
dma_axi_mm_arlen[7:0]	Output	Burst length. The burst length gives the exact number of transfers in a burst. Maximum burst is 512B.
dma_axi_mm_arsize[2:0]	Output	Burst size. This signal indicates the size of each transfer in the burst.
dma_axi_mm_arburst[1:0]	Output	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated.
dma_axi_mm_arlock	Output	Lock type. This signal is tied to '0'.
dma_axi_mm_arprot[2:0]	Output	Protection type. This interface does not use protection attributes.
Read Data Channel		
dma_axi_mm_rvalid	Input	Read data valid. This signal indicates that the channel is signaling the read data is valid.
dma_axi_mm_rready	Output	Read data ready. This signal indicates that the manager can accept the read data and response information.
dma_axi_mm_rid[3:0]	Input	Read ID tag. This signal is the identification tag for the read data group of signals generated by the subordinate.
dma_axi_mm_rdata[DWIDTH-1:0]	Input	Read Data.
dma_axi_mm_rresp[1:0]	Input	Read response. This signal indicates the status of the read transfer. EXOKAY is not supported.
dma_axi_mm_rlast	Input	Read last. This signal indicates the last transfer in a read burst.

4.5.4. BAM AXI-MM Manager (bam_mm_initatr)

The Bursting Master (BAM) bypasses the DMA engine of the GTS AXI MCDMA IP and provides a way for the Host to perform bursting PIO reads/writes to the user logic. This interface is available in **Bursting Master**, **BAM+BAS**, **BAM + MCDMA** and **BAM +BAS+MCDMA** modes. This interface should be connected to the corresponding AXI-MM Subordinate Interface of the application logic.

Interface Clock: `axi_mm_clk`

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Table 48. BAM AXI-MM Manager Interface

Signal Name	Direction	Description
Write Address Channel		
bam_axi_mm_awvalid	Output	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
bam_axi_mm_awready	Input	Write address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals.
bam_axi_mm_awid[3:0]	Output	Write address ID. This signal is the identification tag for the write address group of signals. The default value is 0.
bam_axi_mm_awaddr[n:0]	Output	Write address. The write address gives the address of the first transfer in a write burst transaction. Refer to the Bursting Master (BAM) BAM Address Mapping section for more information on the actual address width.
bam_axi_mm_awlen[7:0]	Output	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
bam_axi_mm_awsize[2:0]	Output	Burst size. This signal indicates the size of each transfer in the burst.
bam_axi_mm_awburst[1:0]	Output	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated.
bam_axi_mm_awlock	Output	Lock type. This signal is tied to '0'.
bam_axi_mm_awprot[2:0]	Output	Protection type. This interface does not use protection attributes.
Write Data Channel		
bam_axi_mm_wvalid	Output	Write Data Valid.
bam_axi_mm_wready	Input	Write Data Ready. This signal indicates that the receiver can accept write data.
bam_axi_mm_wdata[DWIDTH-1:0]	Output	Write data.
bam_axi_mm_wstrb[DWIDTH/8-1:0]	Output	Write strobes. This signal indicates which byte lanes hold valid data.
bam_axi_mm_wlast	Output	Write last. This signal indicates the final transfer in a write burst.
Write Response Channel		
bam_axi_mm_bvalid	Input	Write Response Valid.
bam_axi_mm_bready	Output	Write Response Ready.

continued...

Signal Name	Direction	Description
bam_axi_mm_bid[3:0]	Input	Response ID. This signal is the ID tag of the write response.
bam_axi_mm_bresp[1:0]	Input	Write Response.
Read Address Channel		
bam_axi_mm_arvalid	Output	Read address valid. This signal indicates that the channel is signaling valid read address and control information.
bam_axi_mm_arready	Input	Read address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals
bam_axi_mm_arid[3:0]	Output	Read address ID. This signal is the identification tag for the read address group of signals. The default value is 0.
bam_axi_mm_araddr[n:0]	Output	Read address. The read address gives the address of the first transfer in a read burst transaction. Refer to the Bursting Master (BAM) BAM Address Mapping section for more information on the actual address width.
bam_axi_mm_arlen[7:0]	Output	Burst length. The burst length gives the exact number of transfers in a burst.
bam_axi_mm_arsize[2:0]	Output	Burst size. This signal indicates the size of each transfer in the burst.
bam_axi_mm_arburst[1:0]	Output	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated.
bam_axi_mm_arlock	Output	Lock type. This signal is tied to '0'.
bam_axi_mm_arprot[2:0]	Output	Protection type. This interface does not use protection attributes.
Read Data Channel		
bam_axi_mm_rvalid	Input	Read data valid. This signal indicates that the channel is signaling the read data is valid.
bam_axi_mm_rready	Output	Read data ready. This signal indicates that the manager can accept the read data and response information.
bam_axi_mm_rid[3:0]	Input	Read ID tag. This signal is the identification tag for the read data group of signals generated by the subordinate.
bam_axi_mm_rdata[DWIDTH-1:0]	Input	Read Data.
bam_axi_mm_rresp[1:0]	Input	Read response. This signal indicates the status of the read transfer. EXOKAY is not supported on Altera FPGAs.
bam_axi_mm_rlast	Input	Read last. This signal indicates the last transfer in a read burst.

4.5.5. BAS AXI-MM Subordinate (bas_mm_respndr)

The Bursting Slave (BAS) translates AXI-MM Read and Write transactions from the user logic to PCI Express Mrd and Mwr TLPs. This AXI-MM interface is available in **Bursting Slave**, **BAM+BAS** and **BAM+BAS+MCDMA** User Modes. This interface should be connected to the corresponding AXI-MM Manager Interface of the application logic.

Interface Clock: `axi_mm_clk`

DWIDTH depends on the PCIe mode selected:

- 256 bits for Gen4 1x4
- 128 bits for Gen3 1x4

Table 49. BAS AXI-MM Subordinate Interface

Signal Name	Direction	Description
Write Address Channel		
`bas_axi_mm_awvalid`	Input	Write address valid. This signal indicates that the channel is signaling valid write address and control information.
`bas_axi_mm_awready`	Output	Write address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals.
`bas_axi_mm_awid[3:0]`	Input	Write address ID. This signal is the identification tag for the write address group of signals.
`bas_axi_mm_awaddr[63:0]`	Input	Write address. The write address gives the address of the first transfer in a write burst transaction. The addresses of the Bursting Slave must be aligned to the width of the data bus. For example, if the data width is 64B, the addresses must align to 64B.
`bas_axi_mm_awuser[15:0]`	Input	The write address user information encodes the following information: • EP mode: {1'b0, `vf_num[11:0]`, `vf_active`, `pf_num[2:0]`}. • RP mode: to be used as the Requester ID in the Memory Write TLPs.
`bas_axi_mm_awlen[7:0]`	Input	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
`bas_axi_mm_awsize[2:0]`	Input	Burst size. This signal indicates the size of each transfer in the burst.
`bas_axi_mm_awburst[1:0]`	Input	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated. Only the INCR burst type is supported.
`bas_axi_mm_awlock`	Input	Lock type. This signal is tied to '0'.
`bas_axi_mm_awprot[2:0]`	Input	Protection type. This interface does not use protection attributes.

continued...

Signal Name	Direction	Description
Write Data Channel		
bas_axi_mm_wvalid	Input	Write Data Valid.
bas_axi_mm_wready	Output	Write Data Ready. This signal indicates that the receiver can accept write data.
bas_axi_mm_wdata[DWIDTH-1:0]	Input	Write data.
bas_axi_mm_wstrb[DWIDTH/8-1:0]	Input	Write strobes. This signal indicates which byte lanes hold valid data.
bas_axi_mm_wlast	Input	Write last. This signal indicates the final transfer in a write burst.
Write Response Channel		
bas_axi_mm_bvalid	Output	Write Response Valid.
bas_axi_mm_bready	Input	Write Response Ready.
bas_axi_mm_bid[3:0]	Output	Response ID. This signal is the identification tag of the write response.
bas_axi_mm_bresp[1:0]	Output	Write Response.
Read Address Channel		
bas_axi_mm_arvalid	Input	Read address valid. This signal indicates that the channel is signaling valid read address and control information.
bas_axi_mm_arready	Output	Read address ready. This signal indicates that the subordinate is ready to accept an address and associated control signals
bas_axi_mm_arid[3:0]	Input	Read address ID. This signal is the identification tag for the read address group of signals.
bas_axi_mm_araddr[63:0]	Input	Read address. The read address gives the address of the first transfer in a read burst transaction. The addresses of the Bursting Slave must be aligned to the width of the data bus. For example, if the data width is 64B, the addresses must align to 64B.
bas_axi_mm_aruser[15:0]	Input	The read address user signal encodes the following information: • EP mode: {1'b0, vf_num[11:0], vf_active, pf_num[2:0]}. • RP mode: to be used as the Requester ID in the Memory Read TLPs.
bas_axi_mm_arlen[7:0]	Input	Burst length. The burst length gives the exact number of transfers in a burst.
bas_axi_mm_arsize[2:0]	Input	Burst size. This signal indicates the size of each transfer in the burst.
bas_axi_mm_arburst[1:0]	Input	Burst type. The burst type and the size information determine how the address for each transfer within the burst is calculated. Only the INCR burst type is supported.
bas_axi_mm_arlock	Input	Lock type. Tie this signal to '0'.
bas_axi_mm_arprot[2:0]	Input	Protection type. This interface does not use protection attributes.
Read Data Channel		

continued...

Signal Name	Direction	Description
bas_axi_mm_rvalid	Output	Read data valid. This signal indicates that the channel is signaling the required read data.
bas_axi_mm_rready	Input	Read data ready. This signal indicates that the manager can accept the read data and response information.
bas_axi_mm_rid[3:0]	Output	Read ID tag. This signal is the identification tag for the read data group of signals generated by the subordinate.
bas_axi_mm_rdata[DWIDTH-1:0]	Output	Read Data.
bas_axi_mm_rresp[1:0]	Output	Read response. This signal indicates the status of the read transfer. EXOKAY is not supported.
bas_axi_mm_rlast	Output	Read last. This signal indicates the last transfer in a read burst.

4.5.6. PIO AXI-Lite Manager (pio_lite_initiatr)

The AXI4-Lite PIO Manager is present only if you select the **Multi Channel DMA** User Mode for **MCDMA Settings** in the IP Parameter Editor. The AXI4-Lite PIO Manager is always present irrespective of the user interface type (AXI-S/AXI-MM) that you select. This interface should be connected to the corresponding AXI-MM Subordinate Interface of the application logic.

Interface clock: axi_lite_clk

Table 50. PIO AXI-Lite Manager Interface

Signal Name	Direction	Description
Write Address Channel		
rx_pio_axi_lite_awvalid	Output	Write address valid.
rx_pio_axi_lite_awready	Input	Write address ready. This signal indicates that the subordinate is ready to accept an address.
rx_pio_axi_lite_awaddr[n:0]	Output	Write address. The write address gives the address of the first transfer in a write burst transaction. Refer to the AXI4-Lite PIO Manager address mapping section for more information on the actual address width.
rx_pio_axi_lite_awprot[2:0]	Input	Protection type. This interface does not use protection attributes.
Write Data Channel		
rx_pio_axi_lite_wvalid	Output	Write Data Valid.
rx_pio_axi_lite_wready	Input	Write Data Ready. This signal indicates that the subordinate is ready to accept data.
rx_pio_axi_lite_wdata[63:0]	Output	Write data.

continued...

Signal Name	Direction	Description
rx_pio_axi_lite_wstrb[7:0]	Output	Write strobes. This signal indicates which byte lanes hold valid data.
Write Response Channel		
rx_pio_axi_lite_bvalid	Input	Indicates that the write response channel signals are valid.
rx_pio_axi_lite_bready	Output	Indicates that a transfer on the write response channel can be accepted.
rx_pio_axi_lite_bresp[1:0]	Input	Write Response. Indicates the status of a write transaction.
Read Address Channel		
rx_pio_axi_lite_arvalid	Output	This signal indicates that the read address channel signals are valid.
rx_pio_axi_lite_arready	Input	This signal indicates that a transfer on the read address channel can be accepted.
rx_pio_axi_lite_araddr[n:0]	Output	The address of the first transfer in a read transaction. Refer to the AXI4-Lite PIO Manager address mapping section for more information on the actual address width.
rx_pio_axi_lite_arprot[2:0]	Output	Protection type. This interface does not use protection attributes.
Read Data Channel		
rx_pio_axi_lite_rvalid	Input	Read data valid.
rx_pio_axi_lite_rready	Output	This signal indicates that the manager is ready to accept the read data.
rx_pio_axi_lite_rdata[63:0]	Input	Read Data.
rx_pio_axi_lite_rresp[1:0]	Input	Read response. This signal indicates the status of the read transfer. EXOKAY is not supported.

4.5.7. HIP Reconfiguration AXI-Lite Subordinate (user_csr_lite)

The Hard IP Reconfiguration interface is an AXI-Lite subordinate interface with a 21-bit address and a 32-bit data bus. You can use this bus to dynamically modify the value of the PCIe configuration space registers of all Functions as well as the soft register space registers implemented in the GTS AXI Streaming IP. This interface should be connected to the corresponding AXI-Lite Manager Interface of the application logic.

Note: This interface is supported only in Root Port mode in the current release.

Interface clock: axi_lite_clk

Table 51. HIP Reconfiguration AXI-Lite Slave Interface

Signal Name	Direction	Description
Write Address Channel		
usr_hip_reconfig_awvalid	Input	This signal indicates that the write address channel signals are valid.
usr_hip_reconfig_awready	Output	This signal indicates that a transfer on the write address channel can be accepted.

continued...

Signal Name	Direction	Description
usr_hip_reconfig_awaddr[19:0]	Input	The address of the first transfer in a write transaction.
Write Data Channel		
usr_hip_reconfig_wvalid	Input	Indicates that the write data channel signals are valid.
usr_hip_reconfig_wready	Output	This signal indicates that a transfer on the write data channel can be accepted.
usr_hip_reconfig_wdata[31:0]	Input	Write data.
usr_hip_reconfig_wstrb[3:0]	Input	Write strobes. This signal indicates which byte lanes hold valid data.
Write Response Channel		
usr_hip_reconfig_bvalid	Output	Indicates that the write response channel signals are valid.
usr_hip_reconfig_bready	Input	Indicates that a transfer on the write response channel can be accepted.
usr_hip_reconfig_bresp[1:0]	Output	Write Response. Indicates the status of a write transaction.
Read Address Channel		
usr_hip_reconfig_arvalid	Input	This signal indicates that the read address channel signals are valid.
usr_hip_reconfig_arready	Output	This signal indicates that a transfer on the read address channel can be accepted.
usr_hip_reconfig_araddr[19:0]	Input	The address of the first transfer in a read transaction.
Read Data Channel		
usr_hip_reconfig_rvalid	Output	This signal indicates that the read data channel signals are valid.
usr_hip_reconfig_rready	Input	This signal indicates that a transfer on the read data channel can be accepted.
usr_hip_reconfig_rdata[31:0]	Output	Read Data.
usr_hip_reconfig_rresp[1:0]	Output	Read response. This signal indicates the status of a read transfer. • 2'b00: OKAY • 2'b01: EXOKAY • 2'b10: SLVERR • 2'b11: DECERR

4.5.8. User Event MSI-X (user_msix)

User logic can request the DMA engine to send an event interrupt for a queue associated with a PF/VF. This interface is available by enabling the **Enable User MSI-X** parameter in the IP Parameter Editor.

Interface clock: `axi_lite_clk`

Table 52. User Event MSI-X Interface

Signal Name	Direction	Description
user_event_msix_tvalid	Input	The valid signal qualifies valid data on any cycle with data transfer.
user_event_msix_tready	Output	Asserted when the interface is ready for transfers to take place.
user_event_msix_tdata[15:0]	Input	{msix_queue_dir, rsvd[3:0], msix_queue_num_i[10:0]} **Note:** msix_queue_dir denotes the DMA direction. • 0: Device-to-Host (D2H) DMA Queue • 1: Host-to-Device (H2D) DMA Queue

4.5.9. User Event MSI (user_msi)

This interface is available by enabling the **Enable MSI Capability** parameter in the IP Parameter Editor.

Interface clock: `axi_lite_clk`

Table 53. User Event MSI Interface

Signal Name	Direction	Description
user_msi_req	Input	MSI request signal. Assertion causes MemWrite TLP to be generated to the Host based on the MSI Capability register values and other MSI input ports. You can deassert the MSI request any time after Acknowledge (user_msi_ack) has been asserted.
user_msi_ack	Output	Acknowledge signal for MSI request. Asserted for 1 clock cycle. User logic can deassert the MSI request as soon as this signal is asserted.
user_msi_num[4:0]	Input	MSI number that indicates the offset between the base message data and the MSI to be sent. When multiple messages mode is enabled, this signal sets the lower five bits of the MSI Data register. The user application provides the MSI vector information when multiple messages are enabled. For more information, refer to the Message Data Register for MSI in the PCIe Base Specification.
user_msi_func_num	Input	Specifies the function number requesting an MSI transmission.
user_msi_status[1:0]	Output	Indicates the execution status of the requested MSI. Valid when user_msi_ack is asserted. • 00: MSI message sent. • 01: Pending bit is set and no message sent (MSI is masked). • 10: Error (MSI is not enabled or not allocated). • 11: Reserved.

4.5.10. User Function Level Reset (user_flr)

When the DMA engine receives Functional Level Resets from the Host, the reset requests are propagated to the downstream logic via this interface. In addition to performing resets to its internal logic, the FLR interface waits for an acknowledgment from the user logic for the reset request before it issues an acknowledgement to the GTS AXI Streaming IP. This interface is available by enabling the **Enable User-FLR** parameter in the IP Parameter Editor.

Interface clock: `axi_lite_clk`

Table 54. User Function Level Reset Interface

Signal Name	Direction	Description
user_flr_rcvd_val	Output	Indicates the user logic to begin FLR for the specified channel in usr_flr_rcvd_chan_num. Remains asserted until usr_flr_completed input is sampled high (1'b1).
user_flr_rcvd_chan_num[11:0]	Output	Indicates the channel number for which FLR has to be initiated by the user logic.
user_flr_completed	Input	One-cycle pulse from the user logic indicating the completion of FLR activity for the channel in usr_flr_rcvd_chan_num.

4.5.11. User Configuration Intercept Interface

This interface is available by enabling the **Enable Configuration Intercept Interface** parameter in the IP Parameter Editor.

4.5.11.1. User Configuration Intercept Request Interface (user_cii_req)

Interface clock: `axi_lite_clk`

Table 55. User Configuration Intercept Request Interface (user_cii_req)

Signal Name	Direction	Description
dma_user_st_ciireq_tvalid	Output	When asserted, indicates a valid CFG request cycle is waiting to be intercepted. Deasserted when user_dma_st_ciireq_tready is asserted.
user_dma_st_ciireq_tready	Input	Application asserts this signal for one clock to acknowledge that dma_user_st_ciireq_tvalid is seen by the responder.
dma_user_st_ciireq_tdata[0]	Output	hdr_poisoned: The poisoned bit in the received TLP header on the CII.
dma_user_st_ciireq_tdata[4:1]	Output	hdr_first_be: The first dword byte enable field in the received TLP header on the CII.
dma_user_st_ciireq_tdata[9:5]	Output	Reserved.
dma_user_st_ciireq_tdata[12:10]	Output	func_num: The PF number in the received TLP header on the CII.
dma_user_st_ciireq_tdata[23:13]	Output	vf_num: The child VF number of the parent PF in the received TLP header on the CII.
dma_user_st_ciireq_tdata[24]	Output	vf_active: Indicates the VF number is valid in the received TLP header on the CII.
dma_user_st_ciireq_tdata[25]	Output	wr: Indicates a configuration write request is detected in the received TLP header on the CII. Also, indicates that ss_app_st_ciireq_tdata[67:36] is valid.
dma_user_st_ciireq_tdata[35:26]	Output	addr: The double-word register address in the received TLP header on the CII.
dma_user_st_ciireq_tdata[67:36]	Output	dout: Received TLP payload data from the link partner to your application client. The data is in little endian format. The first received payload byte is in [43:36].
dma_user_st_ciireq_tdata[71:68]	Output	Reserved.

4.5.11.2. User Configuration Intercept Response Interface (user_cii_resp)

Interface clock: `axi_lite_clk`

Table 56. User Configuration Intercept Response Interface (user_cii_resp)

Signal Name	Direction	Description
user_dma_st_ciiresp_tvalid	Input	Application asserts this signal for one clock to indicate that valid data is driven on the dma_user_st_ciiresp_tdata bus.
dma_user_st_ciiresp_tdata[31:0]	Input	Override data. • CfgWr: this bus contains the data supplied by the application logic to override the write data. • CfgRd: this bus contains the data supplied by the application logic to override the data payload of the Completion TLP.
dma_user_st_ciiresp_tdata[32]	Input	Override Data Enable. Application asserts this signal to override the CfgWr payload or CfgRd completion using the data supplied by the application logic on dma_user_st_ciiresp_tdata[31:0].

4.5.12. Configuration Slave (cs_lite_respndr)

The Configuration Slave (CS) is an AXI-lite interface that converts read and write transactions into configuration TLPs to be sent over the PCIe link. This interface is applicable only in Root Port mode.

Interface clock: `axi_lite_clk`

Table 57. Configuration Slave Interface

Signal Name	Direction	Description
Write Address Channel		
`cs_axi_lite_awvalid`	Input	Write address valid.
`cs_axi_lite_awready`	Output	Write address ready.
`cs_axi_lite_awaddr[13:0]`	Input	Write address.
Write Data Channel		
`cs_axi_lite_wvalid`	Input	Write data valid.
`cs_axi_lite_wready`	Output	Write data ready.
`cs_axi_lite_wdata[31:0]`	Input	Write data.
`cs_axi_lite_wstrb[3:0]`	Input	Write data strobes. This signal indicates which byte lanes are valid.
Write Response Channel		
`cs_axi_lite_bvalid`	Output	Write response valid.
`cs_axi_lite_bready`	Input	Write response ready.
`cs_axi_lite_bresp[1:0]`	Output	Write Response. Indicates the status of the write transaction.
Read Address Channel		
`cs_axi_lite_arvalid`	Input	Read address valid.

continued...

Signal Name	Direction	Description
cs_axi_lite_arready	Output	Read address ready.
cs_axi_lite_araddr[13:0]	Input	Read address.
Read Data Channel		
cs_axi_lite_rvalid	Output	Read data valid.
cs_axi_lite_rready	Input	Read data ready.
cs_axi_lite_rdata[31:0]	Output	Read Data.
cs_axi_lite_rresp[1:0]	Output	Read response. This signal indicates the status of the read transfer. EXOKAY is not supported.

5. Simulating the IP

5.1. Design Example Overview

The GTS AXI Multichannel DMA IP for PCI Express design example demonstrates a multichannel DMA solution for Agilex 5 devices using the GTS AXI Streaming IP implemented in the FPGA fabric.

You can generate the design example from the **Example Designs** tab of the GTS AXI Multichannel DMA IP for PCI Express Parameter Editor. Choose the desired user interface type, either AXI-S or AXI-MM. You can allocate up to 256 DMA channels (with a maximum of 256 channels per function) when the AXI-MM interface type or AXI-S 1-port interface is selected.

The following table summarizes the GTS AXI Multichannel DMA IP design example variants, simulation and hardware support status.

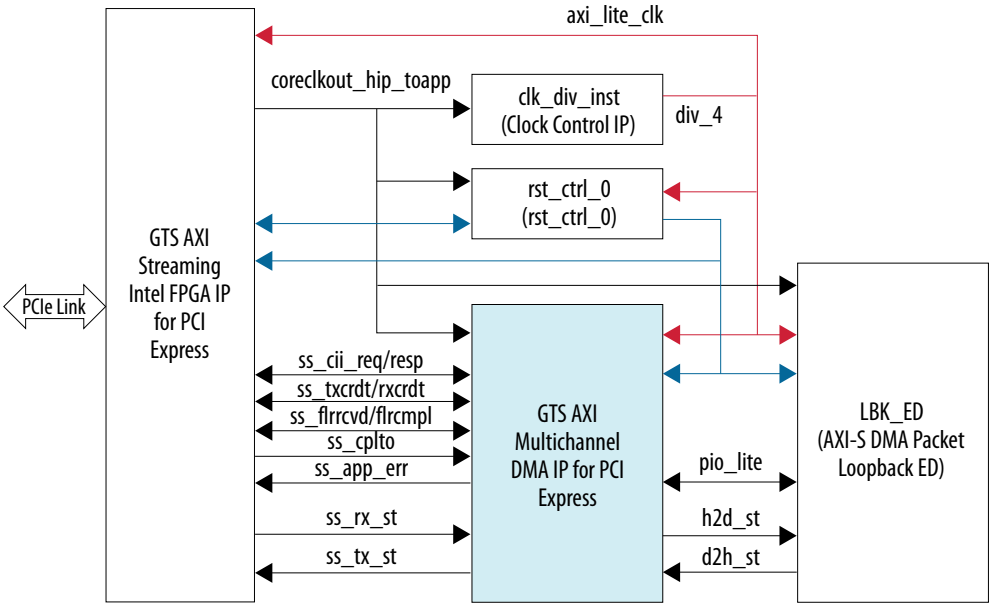
Design Example	MCDMA Settings		PCIe Mode	Simulation	Hardware
	User Mode	Interface Type			
AXI-S Device-side Packet Loopback	Multichannel DMA	AXI-S	Gen4 x4	No Support	No Support
AXI-S Device-side Packet Loopback	Multichannel DMA	AXI-S	Gen3 x4	No Support	Agilex 5 FPGA E-Series 065B Modular Development Kit

5.1.1. AXI-S Device-side Packet Loopback Design Example

In this device-side packet loopback design example, the host initially sets up specific memory locations within its memory. Data from the host memory is then transferred to the device-side memory by the GTS AXI Multichannel DMA IP through Host-to-Device (H2D) DMA operations. Subsequently, the IP loops this data back to the host memory using Device-to-Host (D2H) DMA operations.

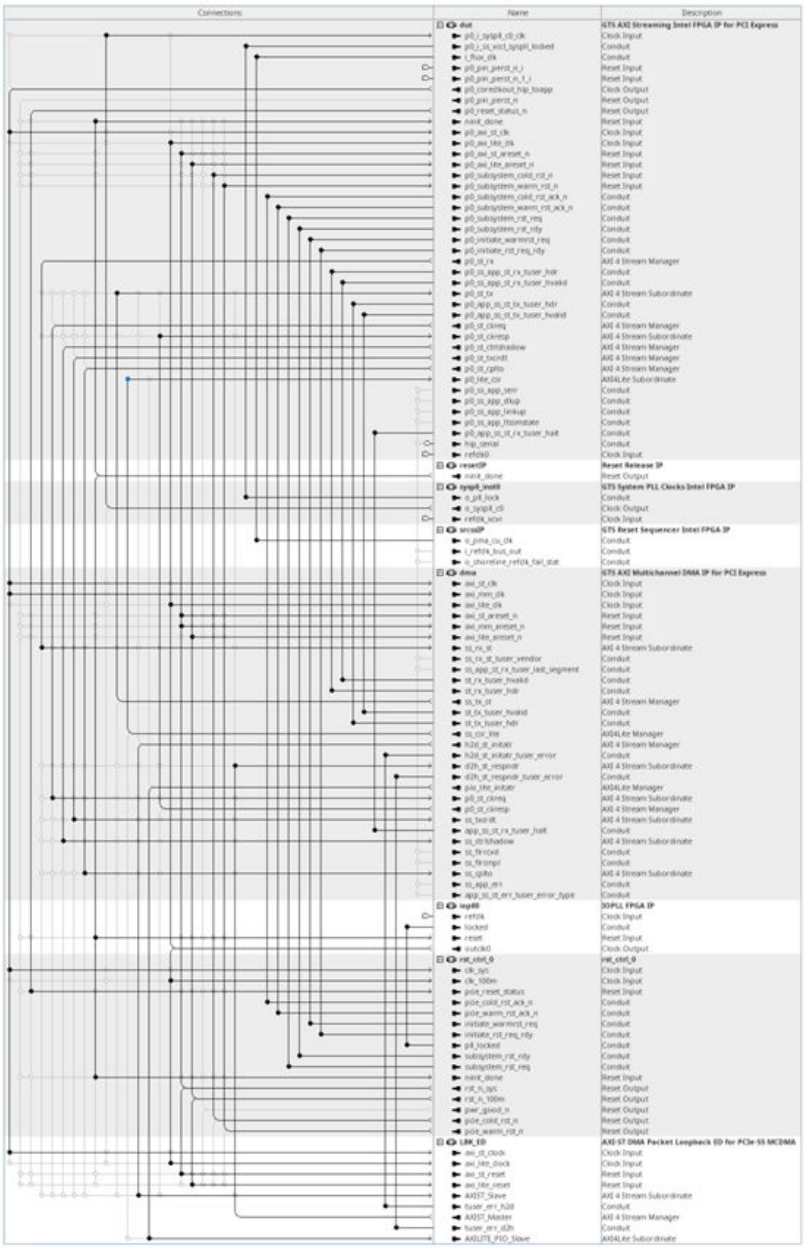
Additionally, the design example enables the AXI-Lite PIO master, which bypasses the DMA path. This allows the application running in the host to perform single, non-bursting register read/write operations with the on-chip memory block.

Figure 27. AXI Streaming Device-side Packet Loopback Design Example



5.2. Design Example Components

Figure 28. Platform Designer System Contents for the AXI-S Device-side Packet Loopback Design Example



The AXI-S device-side packet loopback design example includes the following components:

Design Components	Description
GTS AXI Multichannel DMA IP for PCI Express	Incorporates PCI Express (PCIe) into your design utilizing Intel's advanced PCIe hardened protocol stack, which encompasses the transaction, data link, and physical layers. It also includes optional components like Single Root I/O Virtualization (SR-IOV) for virtualization applications that demand high-bandwidth data transfer to and from the host memory. This component converts the PCIe serial link transfer to the AXI Stream interface and directs the TLP data received to the GTS AXI Multichannel DMA IP.
GTS AXI Streaming Intel FPGA IP for PCI Express	Facilitates efficient data transfer between the local FPGA and the host through multiple DMA channels over the PCIe link. Each DMA channel comprises a host-to-device (H2D) and a device-to-host (D2H) queue pair, operating on descriptor-based queues established by driver software for data transfer. It is engineered to support standalone Endpoint or Rootport functionality, offering AXI-S and AXI-MM interfaces to the user logic.
GTS System PLL Clocks Intel FPGA IP	This IP is required for PCIe interface implementation on Agilex 5 devices to configure the reference clock for the System PLL and provides the clock signal for the p<n>_i_syspll_c0_clk of the GTS AXI Streaming IP. *Note:* For more information, refer to the *Implementing the GTS System PLL Clocks Intel FPGA IP* section in the GTS Transceiver PHY User Guide.
GTS Reset Sequencer Intel FPGA IP	This IP must be instantiated for each Agilex 5 FPGA side that uses transceivers for simulation and proper device operation. It provides the PMA Control Unit clock to the i_flux_clk clock of the GTS AXI Streaming IP. *Note:* For more information, refer to the *Implementing the GTS Reset Sequencer Intel FPGA IP* section in the GTS Transceiver PHY User Guide.
Reset Release IP	This IP holds a control circuit in reset until the device has fully entered user mode. The FPGA asserts the INIT_DONE output to signal that the device is in user mode. It is required when using the GTS AXI Streaming IP. *Note:* For more information on the Reset Release Intel FPGA IP, refer to the Device Configuration User Guide: Agilex 5 FPGAs and SoCs.
AXI-S DMA Packet Loopback ED for PCIe-SS MCDMA	This design example module implements the device-side memory and allows data from the host memory to transfer to it by the AXI Multichannel DMA IP through Host-to-Device (H2D) DMA operations. Subsequently, the IP loops this data back to the host memory using Device-to-Host (D2H) DMA operations. In addition, it provides an AXI-Lite PIO interface that allows the application to perform single, non-bursting register read/write operations with the on-chip memory block.
rst_ctrl_0	This module handles the reset and handshake signals of the GTS AXI Streaming IP for graceful entry and exit for each of the resets (cold, warm, etc.) especially when initiated by the host system.
IOPLL FPGA IP	This IP is required to configure the settings of the Agilex 5 I/O PLL to provide a 100 MHz clock signal for the AXI-Lite interface in the design.

5.3. Simulating the Design Example

Note: Design example simulation is not supported in the current release.

6. Validating the IP

Using the Quartus Prime software, you can generate a design example for the GTS AXI Multichannel DMA IP core. The generated design example reflects the parameters that you specify. The design example automatically creates the files necessary to simulate and compile in the Quartus Prime software. You can download the compiled design to your FPGA Development Board. To download to custom hardware, update the Quartus Prime Settings File (.qsf) with the correct pin assignments.

Figure 29. Design Example Development Steps

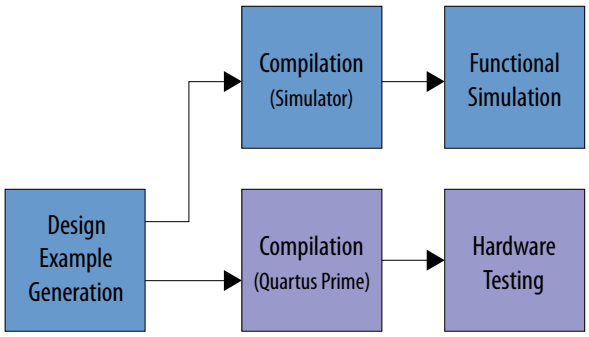
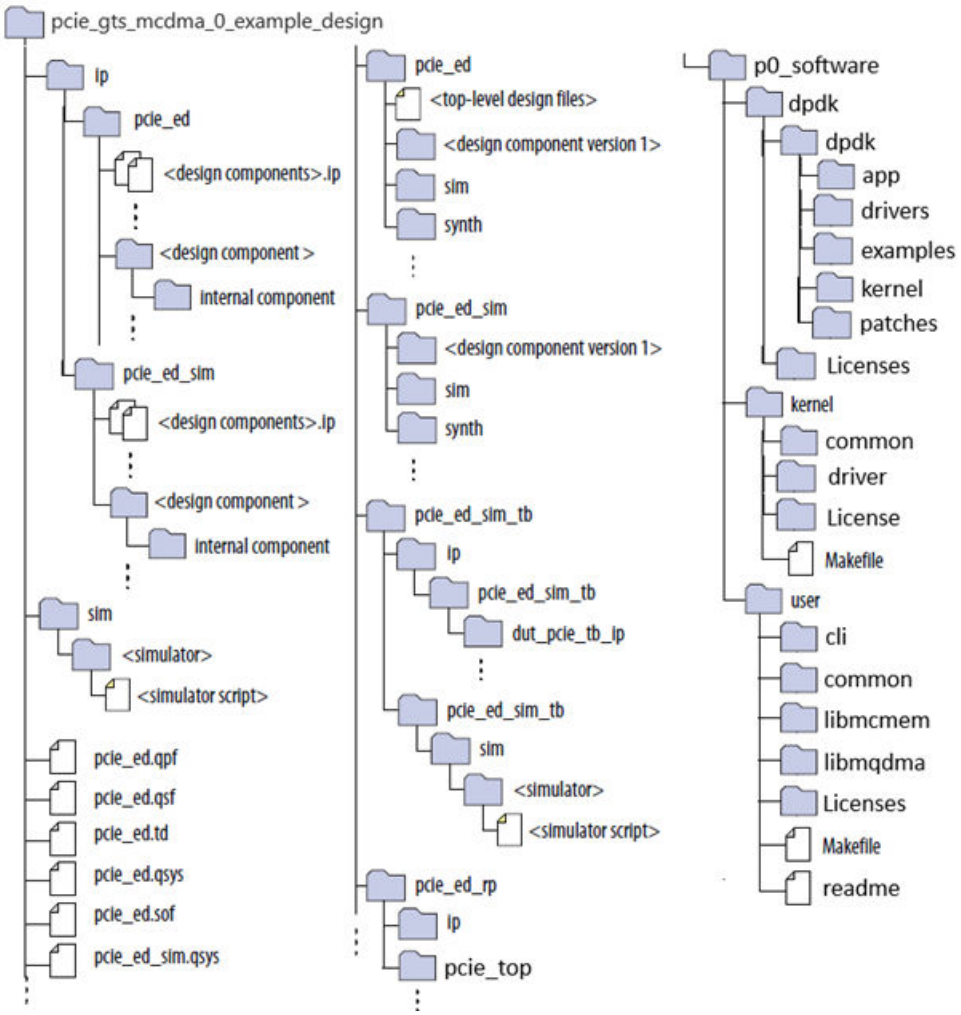


Figure 30. Design Example Directory Structure



6.1. Hardware and Software Requirements

- Quartus Prime Pro Edition software version 25.1
- Host system with PCIe 3.0 x16 slot
  - Operating System: Ubuntu 22.04 LTS Kernel: 5.15.0-134-generic

Table 58. Operating System: Ubuntu 22.04 LTS

Operating System	Ubuntu 22.04 LTS
GCC Version	11.4
Kernel	v5.15

  - Operating System: Ubuntu 24.04.1 LTS Kernel: 6.6.65-060665-generic

Table 59. Operating System: Ubuntu 24.04.1 LTS

Operating System	Ubuntu 24.04 LTS
GCC Version	13
Kernel	v6.6

Note: There is a known issue with the DPDK driver compilation in Ubuntu 24.04 LTS. Refer to the Knowledge Database (KDB) for the solution.

- Agilex 5 FPGA E-Series 065B Modular Development Kit
  - By default, the PCIe reference clock source for the *Agilex 5 FPGA E-Series 065B Modular Development Kit (ES1)* is set to use a local clock source. To test the PCIe example design in the common clock scheme, you must set switch S13.1 of the development kit to the ON position.

For more information on development kits, refer to Altera FPGA Development Kits on the Intel website.

6.2. Running the Design Example Application on a Hardware Setup

Before you can test the design example in hardware, you must configure the FPGA and then restart your computer to allow the enumeration to take place for the PCIe device. After that, install the Linux kernel driver.

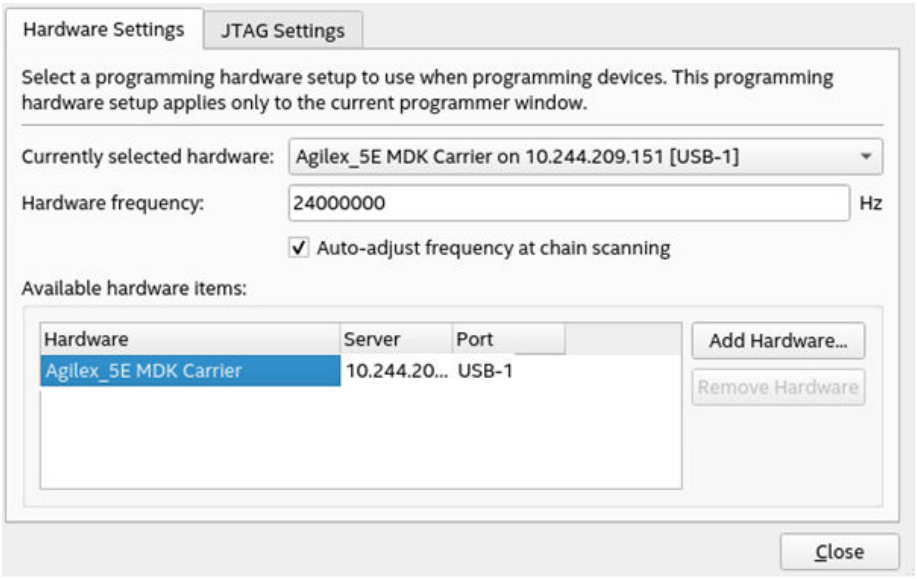
Design Example	MCDMA Settings		Driver Support	Application Support
	User Mode	Interface Type		
AXI-S Device-side Packet Loopback	Multichannel DMA	AXI-S	Custom	Perfq app PIO Test Read Custom PIO Read Write Test Device-side Packet Loopback
			DPDK	MCDMA test PIO Test Device-side Packet Loopback

6.2.1. Program the FPGA

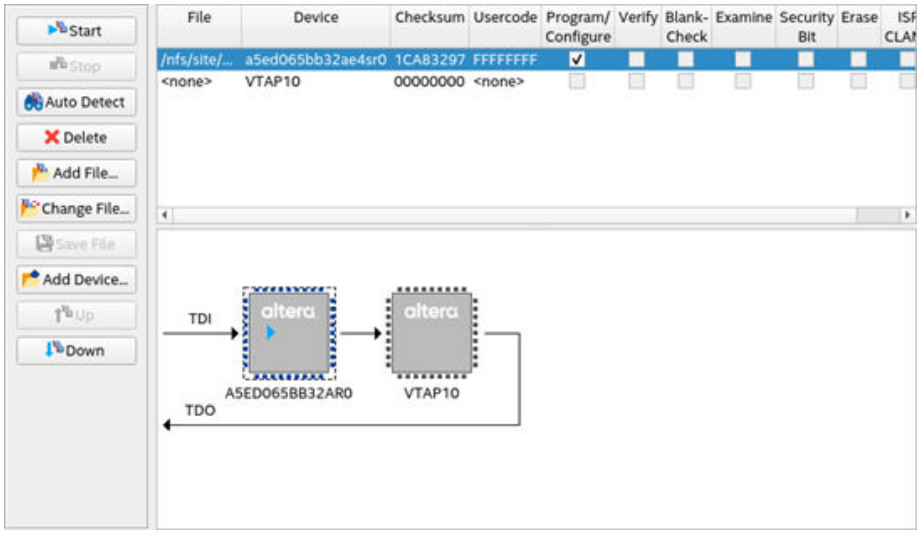
Prerequisite: Generate and compile the design example in the Quartus Prime Pro Edition software before starting to test the design example in hardware.

This section describes how to configure the Agilex 5 FPGA on the Agilex 5 FPGA E-Series 065B Modular Development Kit (ES1).

1. Install the Agilex 5 FPGA E-Series 065B Modular Development Kit in a PCIe Gen3 x16 slot in the host system, connected to the ATX 6-pin power supply.
2. Connect the Agilex 5 FPGA E-Series 065B Modular Development Kit to a computer system in which the Quartus Prime Pro Edition software is installed using the USB cable shipped along with the development kit for FPGA configuration.
3. Power on the host system and turn on the power switch on the development kit.
4. In the Quartus Prime Pro Edition software, invoke the programmer by clicking **Tools** > **Programmer**.
5. In the Programmer, click **Hardware Setup** and verify the Agilex 5 FPGA E-Series 065B Modular Development Kit is detected in the **Hardware Settings** tab.



6. For **Currently selected hardware**, select the Agilex 5 FPGA E-Series 065B Modular Development Kit and then click **Close**.
7. Click **Auto Detect** to detect the JTAG device chain.



8. Select the target FPGA device in the JTAG chain, click **Change File**, and select the FPGA configuration file pcie_ed.sof. Then, click **Open**.
9. Check the Program/Configure option and click **Start** to start the FPGA configuration.
10. Perform a warm reboot for the host system once the Agilex 5 FPGA is successfully configured.
11. Check the enumeration of the PCIe Endpoint device (Agilex 5 FPGA E-Series 065B Modular Development Kit) on the host system by running the "`lspci -d 1172:`" command in a Linux Terminal.

    Expected result:

    BDF Unassigned class [ff00]: Altera Corporation Device 0000 (rev 01)

6.2.2. Configuration Changes from BIOS

Enable the following parameters from the BIOS:

VT-d (or **AMD-V** for AMD processors)

Make sure the IOMMU is enabled on the host by using the following command:

```
$ sudo dmesg | grep IOMMU
```

Expected output:

```
DMAR: IOMMU enabled
```

Enable the following parameters in BIOS if using SRIOV:

- **SRIOV Enable**
- **ARI (Alternative Routing ID Interpretation)**

6.2.3. Host Operating System Check (if Working with Ubuntu 22.04)

Follow the steps below to check if your host operating system version matches the Ubuntu 22.04 as mentioned in Hardware and Software Requirements.

1. Check the kernel version using this command:

   ```
   $ uname -r
   ```

   Expected output:

   ```
   5.15.0-xx-generic
   ```

2. If this is not the kernel version in your Ubuntu 22.04 system, follow the steps below. These steps will change the kernel from Hardware Enablement (HWE) to General Availability (GA) Linux 5.15.0-xx-generic and install Linux headers and gcc required for the MCDMA drivers.

   // Install GA.

   ```
   sudo apt install linux-image-generic
   ```

   // Reboot into the newly-installed older kernel.

   ```
   sudo apt remove linux-generic-hwe-22.04
   ```

   // Remove HWE.

   ```
   sudo apt autoremove
   ```

   // Install required packages

   ```
   sudo apt-get install linux-headers-generic gcc
   ```

3. Check the gcc version using the command below:

   ```
   $ gcc --version
   ```

4. If the gcc version is not gcc-11, install gcc-11 using these commands:

   ```
   $ sudo apt-get update
   ```

   ```
   $ sudo apt install gcc-11
   ```

5. To switch between the installed gcc versions, use the update-alternatives tool and select gcc-11.

   ```
   $ sudo update-alternatives --config gcc
   ```

6.2.4. Installing the Required Kernel Version (if Working with Ubuntu 24.04)

Ubuntu 24.04.2 runs on kernel v6.8 while the supported kernel version is up to v6.6. Follow the steps below to install kernel v6.6.

1. Get the following files from https://kernel.ubuntu.com/mainline/v6.6.65/ and copy them into the /root directory.

- linux-headers-6.6.65-060665-generic_6.6.65-060665.202412111549_amd64.deb
- linux-headers-6.6.65-060665_6.6.65-060665.202412111549_all.deb
- linux-image-unsigned-6.6.65-060665-generic_6.6.65-060665.202412111549_amd64.deb
- linux-modules-6.6.65-060665-generic_6.6.65-060665.202412111549_amd64.deb

2. Run the following command to install the kernel package.

   ```
   $ sudo dpkg -i linux-*.deb
   ```

3. Run the following commands to update the bootloader and reboot the machine to load the new kernel version.

   ```
   $ sudo update-grub
   ```

   ```
   $ sudo reboot
   ```

4. If the system does not boot up Ubuntu with kernel v6.6.65, set the GRUB_DEFAULT as shown below in `/etc/default/grub`:

   ```
   GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux
   6.6.63-060663-generic"
   ```

5. Update the bootloader and reboot by running the following commands.

   ```
   $ sudo update-grub
   ```

   ```
   $ sudo reboot
   ```

6. Run the following commands to check the kernel version:

   ```
   $ sudo uname -r
   ```

   Expected output:

   ```
   6.6.65-060665-generic
   ```

6.2.5. Set the Boot Parameters

Follow the steps below to modify the default **hugepages** setting in the grub files:

1. Edit the **/etc/default/grub** file.

   Append the following parameters to the GRUB_CMDLINE_LINUX line in the /etc/default/grub file:

   For Intel CPU:

   ```
   GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=20
   intel_iommu=on
   iommu=pt processor.max_cstate=0 intel_idle.max_cstate=0 intel_pstate=disable
   panic=1 quiet splash
   vt.handoff=7"
   ```

   For AMD CPU:

   ```
   GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=20
   amd_iommu=on
   iommu=soft processor.max_cstate=0 amd-pstate panic=1 quiet splash
   vt.handoff=7"
   ```

An Intel CPU example of the /etc/default/grub file on Ubuntu after the edits can be seen below:

```
root@bapvecise042:~# cat /etc/default/grub # If you change this file, run
'update-grub'
afterwards to update # /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'
GRUB_DEFAULT="1>2" GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0 GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo
Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=20
intel_iommu=on iommu=pt
nprocessor.max_cstate=0 intel_idle.max_cstate=0 intel_pstate=disable panic=1
quiet splash vt.handoff=7"
# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information
# from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"
# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console
# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB
# with the command `vbeinfo'
#GRUB_GFXMODE=640x480
# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to
Linux
#GRUB_DISABLE_LINUX_UUID=true # Uncomment to disable generation of recovery
mode menu entries
#GRUB_DISABLE_RECOVERY="true" # Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
```

2. Generate GRUB configuration files by running the following command:

   ```
   $ sudo update-grub
   ```

3. Reboot the system.
4. Verify the changes above:

   ```
   $ cat /proc/cmdline
   ```

5. Set the huge pages:

   ```
   $ echo 40 > sudo tee /proc/sys/vm/nr_hugepages
   ```

6. If the host supports multiple NUMAs, follow the following steps:
   a. Check how many NUMAs are enabled on the host.

      ```
      $lscpu | grep NUMA
      NUMA node(s): 2
      NUMA node0 CPU(s): 0-15, 32-47
      NUMA node1 CPU(s): 16-31, 48-63
      ```

      In this example, we have 2 NUMAs. If only one NUMA is present, ignore this step.

   b. Check which device is provisioned:

      ```
      $ cat /sys/class/pci_bus/<Domain:Bus>/device/numa_node
      ```

   c. Enable the huge pages, whichever NUMA device is located:

      ```
      $ echo 40> sudo tee /sys/devices/system/node/node<nodenum>/hugepages/
      hugepages-1048576kB/nr_hugepages
      ```

d. Configure the thread sequence in software /user/cli/perfq_app/perfq_app.h based on which NUMA device is located. For example:

```
#define THREAD_SEQ "0-15"
```

7. Set PCIe_SLOT based on your design example link width configuration. Modify the macro below in the `user/common/include/ifc_libmqdma.h` and `dpdk/dpdk/drivers/net/mcdma/rte_pmd_mcdma.h` files:

```
#define PCIe_SLOT 0 /* 0 - x16, 1 - x8, 2 – x4 */
```

Example of macro setting for x4 link width:

```
#define PCIe_SLOT 2
```

6.2.6. Custom Driver

6.2.6.1. Prerequisites

Prepare the host system for the driver installation and testing.

6.2.6.1.1. Steps for UIO Driver Installation

1. In the host system, install the UIO driver using the command below:

```
$ sudo modprobe uio
```

2. Install "make":
   a. Check if "make" is installed using the following command:

```
$ make --version
```

   b. Install "make" using the following command:

```
$ sudo apt install make
```

   c. Build the mcdma kernel driver and load:

```
$ make clean all -C p0_software/kernel/driver/kmod/mcdma-custom-driver
$ sudo insmod p0_software/kernel/driver/kmod/mcdma-custom-driver/ifc_uio.ko
```

3. Verify whether the driver is loaded or not by running the command below:

```
$ lspci -d 1172:000 -v | grep ifc_uio
```

Expected output:

(Kernel driver in use: `ifc_uio`)

6.2.6.1.2. Enable Virtual Function

When the SR-IOV feature is enabled in the IP for the design example generation, the virtual functions can be enabled in the host system for the tests outlined in the subsequent sections in the same way as the physical function, except the External Descriptor Controller design example variant. This step can be bypassed if you do not intend to test the virtual function.

Here is the command to enable the VF for testing:

```
$ echo <num_of_vfs> | sudo tee /sys/bus/pci/devices/<bdf>/sriov_numvfs
```

For example: Enable 4 VFs on 0000:01:00.0

```
$ echo 4 | sudo tee /sys/bus/pci/devices/0000\:01\:00.0/
sriov_numvfs
```

6.2.6.1.3. Build and Install User Space Library

1. Build the library by running the command below:

   ```
   $ make clean all -C p0_software/user/libmqdma/
   ```

2. Load the library by running the command below:

   ```
   $ sudo cp p0_software/user/libmqdma/libmqdmasoc.so /usr/
   local/lib
   ```

3. Run the command below and make sure that the printout contains `libmqdma`. Look for "`libmqdmasoc.so -> libmqdmasoc.so`" in the printout.

   ```
   $ sudo ldconfig -v | grep libmqdmasoc.so
   ```

6.2.6.1.4. Build the Reference Application

1. Build the reference application by running the commands below:

   ```
   $ cd p0_software/user/cli/perfq_app/
   $ make clean && make all
   $ ./perfq_app -h
   ```

   This command displays the available options in the application, as shown in the image below:

   ```
   [root@BAPVEMB001T perfq_app]#  ./perfq_app -h
   --bas_perf      Enable BAS Performance Mode
   -h              Show Help
   -a <threads>    Number of Threads to be used
   -b <bdf>        BDF of Device
   -c <chnls>      Number of Channels to be used
   -d <seconds>    Refresh rate in seconds for Performance logs
   -e              Enable BAS Mode
   -p <bytes>      Payload Size of each descriptor
   -s <bytes>      Request Size in Bytes
   -g <start chno> Starting channel number while acquiring
   -f <#dsriptrs>  File Size in Descriptors
   -o              PIO Test
   -v              Enable Data Validation
   -n              configure channel count in HW incase of Performance mode
   --files <files> Packet Gen files
   -t              Transmit Operation
   -r              Receive Operation
   -z              Bidirectional Transfer
   -l <seconds>    Time limit in Seconds
   Required parameters: b & ((p & (t | r | z ) & (s | l) & c & a) | o)
   Required parameters BAS Mode:
          Perf:           b & z & s & --bas_perf
          Non perf:       b & (t | r | z) & s & e
   Output format:
   Ch_ID            - Channel ID and Direction
   Req              - Number of Descriptors submitted for DMA Transfers
   Rsp              - Number of Descriptors procesed
   Time             - Total time Elapsed
   Good Descriptors - Number of good descriptors which match the expected data
   Bad Descriptors  - Number of bad descriptors which doesn't match the expected data
   Good Files       - Number of files which had SOF and EOF matched
   Bad Files        - Number of files which didn't match EOF/SOF/Data
   ```

   Refer to the README file located in the software/user/cli/perfq_app directory for more information.

6.2.6.2. PIO Test

This test writes and reads from the PIO memory and validates the data. If data validation passes, then the result is Pass; else, it is Fail.

1. Perform a **PIO test** after completing the instructions outlined in Prerequisites to check if the setup works correctly by running the command below:

   ```
   $ sudo ./perfq_app -b 0000:08:00.0 -o
   ```

   Note: Here the -b option should be provided with the correct BDF of the FPGA endpoint card in the system. It can be checked using the lspci command. If successful, the application displays a Pass status as shown in the image below:

   ```
   iapps@pg-archer-u22045:perfq_app$ sudo ./perfq_app -b 0000:98:00.0 -o
   PIO Write and Read Test ...
   Pass
   iapps@pg-archer-u22045:perfq_app$
   ```

2. The design example can be reset using the utility below when needed:

   a. Build the devmem utility.

   ```
   $ cd software/user/cli/devmem
   $ make clean all
   ```

   b. Perform a reset by running the command below:

   ```
   $ sudo ./devmem 0000:01:00.0 0 0x00200120 0x1
   ```

   The expected printout is as shown below:

   ```
   iapps@pg-archer-u22045:devmem$ sudo ./devmem 0000:98:00.0 0 0x00200120 0x1
   iapps@pg-archer-u22045:devmem$
   ```

6.2.6.3. Custom PIO Read Write Test

You can read and write from the PIO address range in BAR 2 from any valid custom memory.

1. A custom PIO write or read can be performed after completing the instructions outlined in 3.5.7.1. Prerequisites section.
2. Perform a custom write by running the commands below.

   ```
   $cd software/user/cli/perfq_app
   ```

   ```
   $ sudo ./perfq_app -b 0000:01:00.0 -o --pio_w_addr=0x1010 --
   pio_w_val=0x30 --bar=2
   ```

   Expected print out:

   ```
   WRITE: PIO Address = 0x1010 Value = 0x30, bar = 2
   ```

   Parameters for Write operation

   -b <bdf>

   -o

   --pio_w_addr=<address>

   --pio_w_val=<value to write>

--bar=<bar number>

3. Perform a custom read by running the commands below.

   ```
   $ cd software/user/cli/perfq_app
   ```

   ```
   $ sudo ./perfq_app -b 0000:01:00.0 -o --pio_r_addr=0x1010 --bar=2
   ```

   Expected print out:

   ```
   READ: PIO Address = 0x1010 Value = 0x30, bar = 2
   ```

   Parameters for Read operation

   -b <bdf>

   -o

   --pio_r_addr=<address>

   --bar=<bar number>

6.2.6.4. DMA Test

6.2.6.4.1. AXI-S Device-side Packet Loopback (Device-side Packet Loopback)

Example of testing a **loopback design** with the following configuration:

Command:

```
$ cd software/user/cli/perfq_app
$ ./perfq_app -b 0000:01:00.0 -p 32768 -l 5 -i -c 2 -d 2 -a 4
```

Configuration:

- BDF (-b 0000:01:00.0)
- 2 DMA channels (-c 2)
- Bidirectional H2D-D2H loopback (-i)
- Payload length of 32,768 bytes in each descriptor (-p 32768)

- Transfer the data for 5 seconds (-l 5)
- Dump the progress log every 2 seconds (-d 2)
- Total of four threads, with one thread per queue x 2 queues per channel x 2 channels (-a 4)

```
iapps@pg-archer-u22045:perfq_app$ sudo ./perfq_app -b 0000:98:00.0 -p 32768 -l 5 -i -c 2 -d 2 -a 4
Allocating 2 Channels...

--------------------------------------------------------------------
BDF: 0000:98:00.0
Channels Allocated: 2
QDepth 508
Number of pages: 8
Completion mode: WB
Payload Size per descriptor: 32768 Bytes
SOF on descriptor: 1
EOF on descriptor: 1
File Size: 32768 Bytes
Tx Batch Size: 127 Descriptors
Rx Batch Size: 127 Descriptors
DCA: OFF
--------------------------------------------------------------------
Thread initialization in progress ...
Thread initialization done
-------------------------------------------------------------------------------------------------------------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd      TBW        IBW        MIBW       HIBW       LIBW      MPPS      #stuck
Tx      2       00:02:000     6205728.00KB  02.96GBPS  02.96GBPS  01.48GBPS  01.48GBPS  01.48GBPS 00.10MPPS     0
Rx      2       00:02:000     6205728.00KB  02.96GBPS  02.96GBPS  01.48GBPS  01.48GBPS  01.48GBPS 00.10MPPS     0
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd      TBW        IBW        MIBW       HIBW       LIBW      MPPS      #stuck
Tx      2       00:04:000     12399264.00KB  02.96GBPS  02.95GBPS  01.48GBPS  01.48GBPS  01.47GBPS 00.10MPPS     0
Rx      2       00:04:000     12399264.00KB  02.96GBPS  02.95GBPS  01.48GBPS  01.48GBPS  01.47GBPS 00.10MPPS     0
-------------------------------------------------------------------------------------------------------------------------------------
All Threads exited
-----------------------------------OUTPUT SUMMARY-----------------------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd           TBW      MPPS    d_drop_cnt  Passed  Failed  %passed
Tx      2       00:04:029     12492736.00KB    02.96GBPS 00.10MPPS        0      2       0     100.00%
Rx      2       00:04:029     12492736.00KB    02.96GBPS 00.10MPPS        0      2       0     100.00%
-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
Total Bandwidth: 5.92GBPS, 0.19MPPS
-------------------------------------------------------------------------------------------
Full Forms:
TBW:            Total Bandwidth
IBW:            Interval Bandwidth
MIBW:           Mean Interval Bandwidth
HIBW:           Highest Interval Bandwidth
LIBW:           Lowest Interval Bandwidth
Please refer to perfq_log_20250428_163556.txt for more details
iapps@pg-archer-u22045:perfq_app$
```

6.2.7. DPDK Poll Mode Driver

6.2.7.1. Prerequisites

Prepare the host system for the driver installation and testing.

External Packages

To run the DPDK software driver, you must install the `numactl-devel` package:

```
$ sudo apt-get install numactl libnuma-dev meson python3-pyelftools  netperf
iperf iperf3 ethtool dwarves git
```

6.2.7.1.1. Install PMD Driver

1. In the host system, give execute permission to the v21.11.2 patch script.

```
$ cd p0_software/dpdk/dpdk/patches/v21.11.2/
$ chmod 777 apply-patch.sh
```

2. Delete the dpdk-stable folder if it was already created. Create a new one using the patch script.

```
$ rm -rf dpdk-stable
$ sh ./apply-patch.sh
```

3. Configure and build dpdk.

```
$ cd dpdk-stable
$ meson build
$ DESTDIR=install ninja -C build install
$ cd build
```

4. Build the igb_uio kernel driver and mcdma-test/perfq app.

```
$ meson configure -Dexamples=mcdma-test/perfq -Denable_kmods=true
$ ninja
```

6.2.7.1.2. Bind the Device

1. Install the UIO base module.

   ```
   $ sudo modprobe uio
   ```

2. Install uio_igb:

   ```
   $ sudo insmod ./kernel/linux/igb_uio/igb_uio.ko
   ```

3. To bind the driver:

   ```
   $ cd ../../
   $ sudo dpdk-stable/usertools/dpdk-devbind.py -b igb_uio <BDF>
   ```

   Example:

   ```
   $ sudo dpdk-stable/usertools/dpdk-devbind.py -b igb_uio
   01:00.0
   ```

   To unbind the driver:

   ```
   $ sudo -c "echo <BDF> > sys/bus/pci/devices/<BDF>/driver/
   unbind"
   ```

   Example:

   ```
   $ sudo -c "echo 0000:01:00.0 > /sys/bus/pci/devices/
   0000:01:00.0/driver/unbind"
   ```

6.2.7.1.3. Enable Virtual Function

When the SR-IOV feature is enabled in the IP for the design example generation, the virtual functions can be enabled in the host system for the tests outlined in the subsequent sections, except for the External Descriptor Controller design example variant. This step can be bypassed if you do not intend to test the virtual function.

Here is the command to enable the VF for testing:

```
$echo <num_of_vfs> | sudo tee /sys/bus/pci/devices/<bdf>/
sriov_numvfs
```

For example: Enable 4 VFs on 0000:01:00.0

```
echo 4 | sudo tee /sys/bus/pci/devices/0000\:01\:00.0/
sriov_numvfs
```

6.2.7.2. PIO Test

1. Complete the instructions outlined in Prerequisites before running the reference application mcdma-test:

```
$ sudo ./build/mcdma-test -- -h
```

This command displays the available options in the application as shown in the image below:

```
iapps@pg-archer-u22045:perfq$ sudo ./build/mcdma-test -- -b 0000:98:00.0 -o
EAL: Detected CPU lcores: 96
EAL: Detected NUMA nodes: 2
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available 2048 kB hugepages reported
EAL: VFIO support initialized
EAL: Probe PCI driver: net_mcdma (1172:0) device: 0000:98:00.0 (socket 1)
PMD: ifc_mcdma_get_hw_version(): MCDMA RTL VERSION : 0x30000
PMD: ifc_mcdma_get_device_caps(): Max Supported by DPDK: 1024
TELEMETRY: No legacy callbacks, legacy socket not created
PIO Write and Read Test on port 0
Pass
iapps@pg-archer-u22045:perfq$ sudo ./build/mcdma-test -- -h
EAL: Detected CPU lcores: 96
EAL: Detected NUMA nodes: 2
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available 2048 kB hugepages reported
EAL: VFIO support initialized
EAL: Probe PCI driver: net_mcdma (1172:0) device: 0000:98:00.0 (socket 1)
PMD: ifc_mcdma_get_hw_version(): MCDMA RTL VERSION : 0x30000
PMD: ifc_mcdma_get_device_caps(): Max Supported by DPDK: 1024
TELEMETRY: No legacy callbacks, legacy socket not created
--bar <num>     BAR number to be configured for BAM/BAS
--bas_perf      Enable BAS Performance Mode
--bas           Enable BAS Non-perf Mode
--bam_perf      Enable BAM Performance Mode
--bam           Enable BAM Non-perf Mode
-h              Show Help
-a <threads>    Number of Threads to be used
-b <bdf>        BDF of Device
-c <chnls>      Number of Channels to be used
-d <seconds>    Refresh rate in seconds for Performance logs
-e              IP Reset
-p <bytes>      Payload Size of each descriptor
-s <bytes>      Request Size in Bytes
-f <#dsriptrs>  File Size in Descriptors
-l <seconds>    Time limit in Seconds
-o              PIO Test
-v              Enable Data Validation
-j              D2H Payload per Descriptor
-k              D2H expected File Size in bytes
-g              Clear the allocated Channel
-x <#dscriptrs> Batch Size in Descriptors
-u              Loop back Transfer
Required parameters: b & ((p & u) & (s | l) & c & a) | o)
-i              Independent Loop back
Required parameters: b & (i & (s | l) & c & a) | o | e)
Required parameters BAS Mode:
        Perf:           b & z & s & --bas_perf
        Non perf:       b & (t | r | z) & s & --bas
CLI format:
        ./build/mcdma-test <EAL_OPTIONS> -- <MCDMA_OPTIONS>
Output format:
Ch_ID            - Channel ID and Direction
Req              - Number of Descriptors submitted for DMA Transfers
Rsp              - Number of Descriptors procesed
Time             - Total time Elapsed
Good Descriptors - Number of good descriptors which match the expected data
Bad Descriptors  - Number of bad descriptors which doesn't match the expected data
Good Files       - Number of files which had SOF and EOF matched
Bad Files        - Number of files which didn't match EOF/SOF/Data
iapps@pg-archer-u22045:perfq$
```

2. Perform a PIO test to check if the setup works correctly. If successful, the application shows a Pass status.

   ```
   $ ./build/mcdma-test -- -b 0000:01:00.0 -o
   ```

   The -b option should be updated with the correct BDF of the endpoint device in the system.

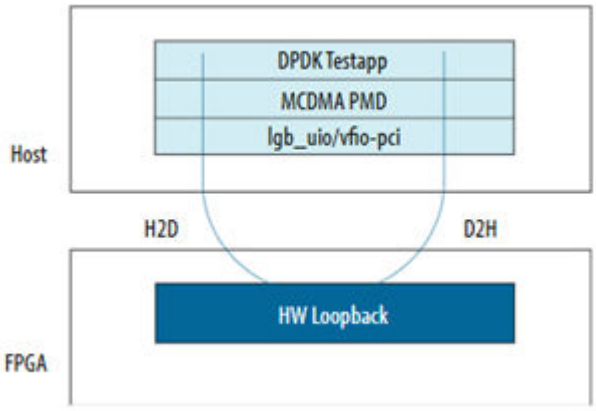
```
iapps@pg-archer-u22045:perfq$ sudo ./build/mcdma-test -- -b 0000:98:00.0 -o
EAL: Detected CPU lcores: 96
EAL: Detected NUMA nodes: 2
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available 2048 kB hugepages reported
EAL: VFIO support initialized
EAL: Probe PCI driver: net_mcdma (1172:0) device: 0000:98:00.0 (socket 1)
PMD: ifc_mcdma_get_hw_version(): MCDMA RTL VERSION : 0x30000
PMD: ifc_mcdma_get_device_caps(): Max Supported by DPDK: 1024
TELEMETRY: No legacy callbacks, legacy socket not created
PIO Write and Read Test on port 0
Pass
iapps@pg-archer-u22045:perfq$
```

6.2.7.3. DMA Test

6.2.7.3.1. Device-side Packet Loopback

The DPDK driver can also be used with the AXI-S Device-side Packet Loopback design example for loopback test.

The following diagram shows the testing strategy.



Complete the instructions outlined in Prerequisites before running the following command for the loopback test:

```
$ ./build/mcdma-test -m 8192 --file-prefix=pf0 -l 0-15 -- -b 0000:01:00.0
-p 32768 -d 2 -c 1 -a 2 -l 5 -i
```

Configuration:

- Memory allocation size (-m 8192)
- Physical function number (--file-prefix=pf0)
- Number of CPU cores allocated for the test (-l 0-15), to match the output from the "`sudo lscpu | grep NUMA`" command
- BDF (-b 0000:01:00.0)
- 1 DMA channel (-c 1)
- Bidirectional H2D-D2H Loopback (-i)
- Payload length of 32,768 bytes in each descriptor (-p 32768)

- Transfer the data for 5 seconds (-l 5)
- Dump the progress log every 2 seconds (-d 2)
- Total of two threads (-a 2)
- Transfer data for 5 seconds (-l 5)

```
iapps@pg-archer-u22045:perfq$ sudo ./build/mcdma-test -m 8192 --file-prefix=pf0 -l 0-15 -- -b 0000:98:00.0 -p 32768 -d 2 -c 1 -a 2 -l 5 -i
EAL: Detected CPU lcores: 96
EAL: Detected NUMA nodes: 2
EAL: Detected static linkage of DPDK
EAL: Multi-process socket /var/run/dpdk/pf0/mp_socket
EAL: Selected IOVA mode 'PA'
EAL: No available 2048 kB hugepages reported
EAL: VFIO support initialized
EAL: Probe PCI driver: net_mcdma (1172:0) device: 0000:98:00.0 (socket 1)
PMD: ifc_mcdma_get_hw_version(): MCDMA RTL VERSION : 0x30000
PMD: ifc_mcdma_get_device_caps(): Max Supported by DPDK: 1024
TELEMETRY: No legacy callbacks, legacy socket not created
Allocating 1 Channels...

-------------------------------------------------
BDF: 0000:98:00.0
Channels Allocated: 1
QDepth 508
Number of pages: 8
Completion mode: WB
H2D Payload Size per descriptor: 32768 Bytes
D2H Payload Size per descriptor: 32768 Bytes
H2D SOF on descriptor: 1
H2D EOF on descriptor: 1
H2D File Size: 32768 Bytes
D2H SOF on descriptor: 1
D2H EOF on descriptor: 1
D2H File Size: 32768 Bytes
File Size: 32768 Bytes
Tx Batch Size: 254 Descriptors
Rx Batch Size: 254 Descriptors
TID FIFO Checks: OFF
AVST Interface
DCA: OFF
-------------------------------------------------------------
Thread initialization in progress ...
Thread is in READY state...
Thread initialization done
-------------------------------------------------------------------------------------------------------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd      TBW        IBW        MIBW       HIBW       LIBW      MPPS      #stuck
Tx      1       00:02:000     6242304.00KB  02.98GBPS  02.98GBPS  02.98GBPS  02.98GBPS  02.98GBPS 00.10MPPS     0
Rx      1       00:02:000     6242304.00KB  02.98GBPS  02.98GBPS  02.98GBPS  02.98GBPS  02.98GBPS 00.10MPPS     0
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd      TBW        IBW        MIBW       HIBW       LIBW      MPPS      #stuck
Tx      1       00:04:000     12476480.00KB  02.97GBPS  02.97GBPS  02.97GBPS  02.97GBPS  02.97GBPS 00.10MPPS     0
Rx      1       00:04:000     12476480.00KB  02.97GBPS  02.97GBPS  02.97GBPS  02.97GBPS  02.97GBPS 00.10MPPS     0
-------------------------------------------------------------------------------------------------------------------------------
All Threads exited
-------------------------------------------------------------------------------------------------------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd      TBW        IBW        MIBW       HIBW       LIBW      MPPS      #stuck
Tx      1       00:06:000     15240000.00KB  03.63GBPS  01.32GBPS  01.32GBPS  01.32GBPS  01.32GBPS 00.04MPPS     0
Rx      1       00:06:000     15240000.00KB  03.63GBPS  01.32GBPS  01.32GBPS  01.32GBPS  01.32GBPS 00.04MPPS     0
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------OUTPUT SUMMARY--------------------------------
Dir  #queues   Time_elpsd     B_trnsfrd           TBW     d_drop_cnt  Passed  Failed  %passed
Tx      1       00:06:879     15240000.00KB    02.98GBPS       0         1      0     100.00%
Rx      1       00:06:879     15240000.00KB    02.98GBPS       0         1      0     100.00%
-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
Total Bandwidth: 5.96GBPS, 0.20MPPS
Total TX Bandwidth: 2.98GBPS, 0.10MPPS
Total RX Bandwidth: 2.98GBPS, 0.10MPPS
Total data drop count :0
-------------------------------------------------------------------------------------------------------
Full Forms:
TBW:            Total Bandwidth
IBW:            Interval Bandwidth
MIBW:           Mean Interval Bandwidth
HIBW:           Highest Interval Bandwidth
LIBW:           Lowest Interval Bandwidth
Please refer to perfq_log_20250428_112931.txt for more details
        total_drops:0
iapps@pg-archer-u22045:perfq$
```

Configuration for AXI-S LB Undefined single function mode in **`examples/mcdma-test/perfq/meson.build`**.

```
-UIFC_MCDMA_SINGLE_FUNC
```

Command for loopback:

```
./build/mcdma-test –m 16384 –l 0-8 - -b -p 64 -l 2 -i -d 1 -c 2048
```

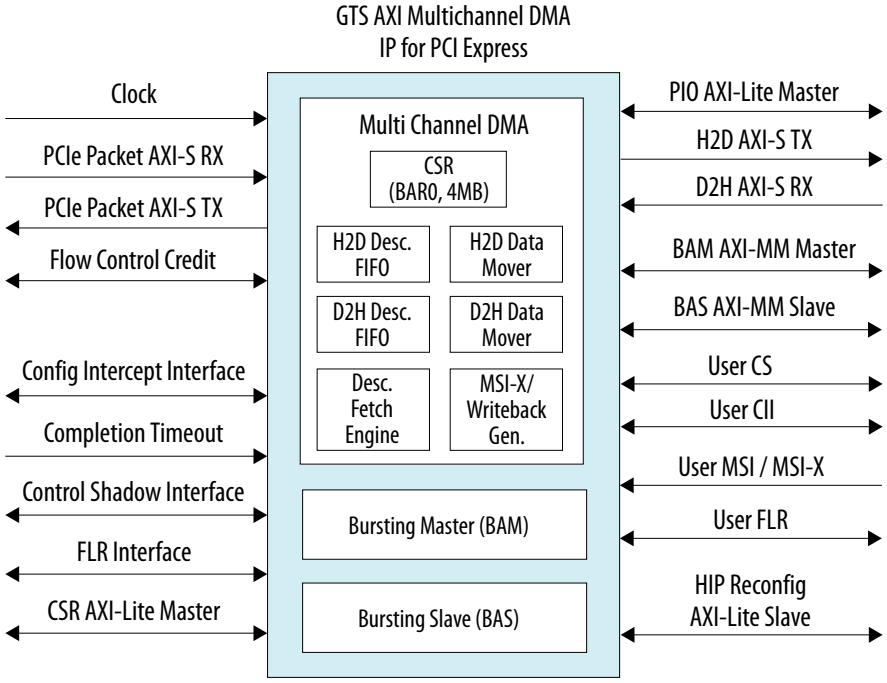
Note: In the current release, a single page is supported in DIDF mode.

Note: In the current release, a simultaneous process is not supported in DIDF mode. You can run one process with 2K channels.

A. Appendix A: Functional Description

A.1. High-level Functional Overview

Figure 31. GTS AXI MCDMA IP Block Diagram



Note that not all the blocks co-exist in a design. Required functional blocks are enabled based on the user mode that you select when you configure the IP. The following table shows valid user modes that the GTS AXI Multichannel DMA IP for PCI Express supports. Each row indicates a user mode with required block(s).

Table 60. Valid User Modes and Required Functional Blocks

Mode		MCDMA	Bursting Master (BAM)	Bursting Slave (BAS)	Configuration Slave (CS)
Endpoint	MCDMA	√	x	x	x
	BAM	x	√	x	x
	BAS	x	x	√	x
	BAM + BAS	x	√	√	x
	BAM + MCDMA	√	√	x	x

continued...

Mode		MCDMA	Bursting Master (BAM)	Bursting Slave (BAS)	Configuration Slave (CS)
	BAM + BAS + MCDMA	√	√	√	x
Root Port	BAM	x	√	x	√
	BAS	x	x	√	√
	BAM + BAS	x	√	√	√

A.1.1. Multichannel DMA

The GTS AXI Multichannel DMA IP for PCI Express consists primarily of Host-to-Device Data Mover (H2DDM) and Device-to-Host Data Mover (D2HDM) blocks. It also offers a DMA-bypass capability to the Host for doing PIO Reads/Writes to the device memory.

The MCDMA engine operates on a software DMA queue to transfer data between the local FPGA and Host. The elements of each queue are software descriptors that are written by the driver/software. Hardware reads the queue descriptors and executes them. Hardware can support up to 256 DMA channels. For each channel, separate queues are used for read/write DMA operations.

A.1.1.1. H2D Data Mover

In Multichannel DMA mode, the Host-to-Device Data Mover (H2DDM) module transfers data from the host memory to local memory through the GTS AXI Streaming IP and the AXI-MM Manager/AXI-Stream Manager interface of the GTS AXI MCDMA IP.

There are two modes of usage for the H2DDM: queue descriptors fetching and H2D data payload transfer.

When used for descriptor fetching, the destination of the completion data is internal descriptor FIFOs where descriptors are stored before being dispatched to the H2DDM or D2HDM for actual data transfer.

When used for data payload transfer, the H2DDM generates MemRd TLPs based on descriptor information such as the PCIe address (source), data size, and MRRS value as follows:

- First MemRd to the MRRS address boundary.
- Following with MemRd's of full MRRS size.
- Last MemRd of the remaining partial MRRS.

The received completions are reordered to ensure the read data is delivered to the user logic in order.

When a descriptor is completed, that is, all read data has been received and forwarded to the AXI-MM Manager/AXI-Stream Manager interface, the H2DDM performs the housekeeping tasks that include:

- Schedule MSI-X for a completed queue, if enabled.
- Schedule Writeback Consumed Head Pointer for a completed queue, if enabled.
- Update Consume Head Pointer for software polling.

MSI-X and Writeback are memory writes to the Host via the D2HDM to avoid a race condition due to out-of-order writes. Based on the updated status, software can proceed with releasing the transmit buffer and reuse the descriptor ring entries.

A.1.1.2. D2H Data Mover

The D2H Data Mover (D2HDM) transfers data from the device memory to host memory. It receives the data from the user logic through the AXI-MM Subordinate/ AXI-Stream Subordinate interface and generates MemWr TLPs to move the data to the host based on descriptor information such as the PCIe address (destination), data size, and MPS value to transmit data to the receive buffer in the host memory.

In AXI-MM mode, the D2HDM sends a series of AXI-MM reads based on PCIe address, MPS, and DMA transfer size. The AXI-MM read is generated as follows:

- First AXI-MM read to the 64-byte address boundary. Multiple bursts are read on first AXI-MM read if:
  - AXI-MM address is 64-byte aligned.
  - Total payload count of the descriptor is 64-byte aligned and less than the maximum supported MPS.
- Following with AXI-MM reads with the maximum supported MPS size.
- Last AXI-MM Read of the remaining size.

In AXI-S mode, D2HDM deasserts ready when descriptors are not available.

- Host sets up software descriptors for a port. Maximum payload count can be up to 1MB. SOF/EOF fields in the descriptor may not be set by the Host.
  - D2HDM uses a descriptor update sequence to update SOF, EOF, Rx payload count fields in the software descriptor at a Host location through a Memory Write request.
- Once there is an available descriptor, incoming data from the user logic triggers a descriptor update sequence by D2HDM to mark the start of the AXI-Stream frame.
  - D2HDM issues a MWr to set the SOF field in the descriptor.
  - WB/MSI-X, if set in the descriptor, is issued.
- AXI-Stream d2h_axi_st_tlast signal assertion triggers a descriptor update sequence by D2HDM to mark the end of the AXI-Stream frame. The descriptor update sequence is as follows:
  - D2HDM terminates the descriptor and initiates a descriptor update sequence.
  - During the descriptor update sequence, a MWr is issued to set the EOF field in the descriptor and update the Rx payload count field with the total bytes transferred.
  - WB/MSI-X, if set in the descriptor, is issued.
- The descriptor immediately after the EOF sequence is considered as the start of the next AVST data frame and initiates a descriptor update sequence to set the SOF field.

When a descriptor is completed, that is, all DMA data corresponding to the descriptor has been sent to the host, the D2HDM performs housekeeping tasks that include:

- Schedule MSI-X for a completed queue, if enabled in the descriptor.
- Schedule Writeback Consumed Head Pointer for a completed queue, if enabled in the descriptor.
- Update Consume Head Pointer for software polling.
- MSI-X and Writeback are memory writes to the host via the D2HDM to avoid a race condition due to out-of-order writes.

Based on the updated status, software can proceed with releasing the receive buffer and reuse the descriptor ring entries.

A.1.1.2.1. D2H Descriptor Fetch

When you enable multiple DMA channels in AXI-S mode, the GTS AXI MCDMA IP limits the number of the channels that can be active or can prefetch the descriptors for the data movement to avoid implementing the larger memory to hold descriptors simultaneously for all channels.

The descriptor FIFO is designed to hold descriptors only for a defined number of channels. When the data is received on the user interface, there is no handshake between the host software and User Logic through the GTS AXI MCDMA IP to control the order of descriptor fetch or data movement of multiple channels. To enable easy access to descriptors of multiple channels, the GTS AXI MCDMA IP implements segmentation of the descriptor FIFO.

In AXI-S mode, when data is received for a channel that does not have Tail pointer (TID) updates from the host, the corresponding AXI-Stream packet from SOF to EOF is dropped.

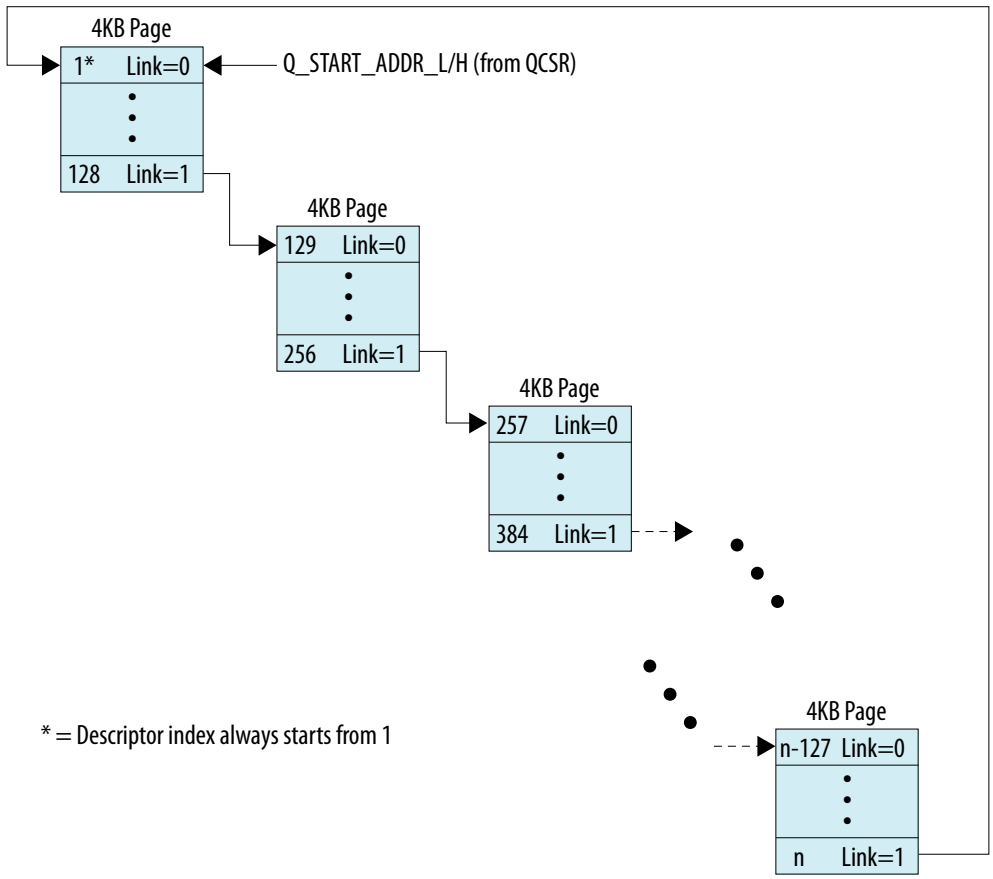
Note: D2H does not support Tail pointer updates on a disabled channel. The host software must make sure a channel is enabled before doing Tail pointer updates.

A.1.1.3. Descriptors

A DMA channel to support Multichannel DMA data movement consists of a pair of descriptor queues: one H2D descriptor queue and one D2H descriptor queue. Descriptors are arranged contiguously within a 4 KB page.

Each descriptor is 32 bytes in size. The descriptors are kept in the host memory in a linked-list of 4 KB pages. For a 32-byte descriptor and a 4 KB page, each page contains up to 128 descriptors. The last descriptor in a 4 KB page must be a “link descriptor” – a descriptor containing a link to the next 4 KB page with the link bit set to 1. The last entry in the linked list must be a link pointing to the base address (Q_START_ADDR_L/H registers) programmed in the QCSR, in order to achieve a circular buffer containing a linked-list of 4 KB pages. The figure below shows the descriptor linked list.

Figure 32. Descriptor Linked-List



Software and hardware communicate and manage the descriptors using the tail index pointer (Q_TAIL_POINTER) and head index pointer (Q_HEAD_POINTER) QCSR registers as shown in the following figure. The DMA starts when software writes the last valid descriptor index to the Q_TAIL_POINTER register.

Figure 33. Descriptor Ring Buffer

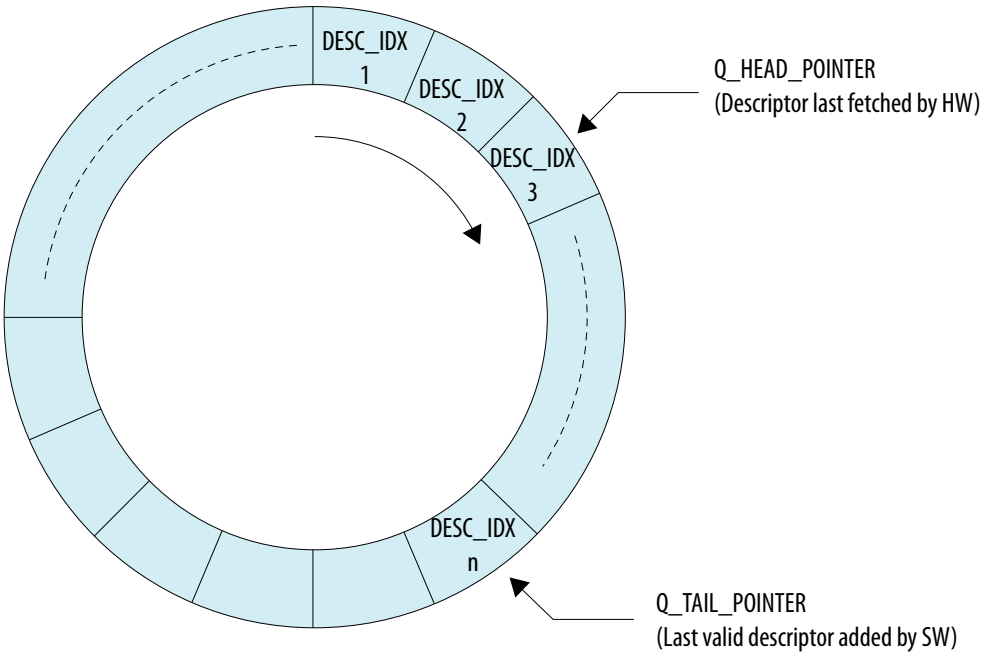


Table 61. Software Descriptor Format

Name	Width	Description
SRC_ADDR [63:0]	64	If Link bit = 0, then this field contains the source address. Starting system address of the allocated transmit buffer read by DMA. If the queue is H2D, then this field contains the address in Host Memory. If the queue is D2H, then this is the device memory address (AXI-MM mode) or don't-care (AXI-S mode). If the link bit is set, then this contains the address of the next 4 KB page in the host memory containing the descriptors.
DEST_ADDR [127:64]	64	Provided link=0, this field means: Starting system address of the allocated receive buffer written by DMA. If the queue is D2H, then this field contains the address in Host Memory. If the queue is H2D, then this is the device memory address (AXI-MM mode) or don't-care (AXI-S mode).
PYLD_CNT [147:128]	20	Provided link=0, this field means DMA payload size in bytes. Maximum 1 MB, with 20'h0 indicating 1 MB. For a D2H queue in AXI-S mode, this field is a fixed value across all descriptors of the queue and must reflect the value in the Q_PYLD_CNT (0x44) QCSR register.
RSRVD [159:148]	12	Reserved
DESC_IDX [175:160]	16	Unique Identifier for each descriptor, assigned by the software driver. This value is written to the Q_COMPLETED_POINTER register when a descriptor data transfer is complete. *Note:* First descriptor DESC_IDX value is 1, not 0.
MSIX_EN [176]	1	Enable MSI-X per descriptor.
WB_EN [177]	1	Enable Write Back per descriptor.

continued...

Name	Width	Description
RSRVD [191:178]	14	Reserved
RX_PYLD_CNT [211:192]	20	Received actual payload for D2H data movement (upstream).
RSRVD [221:212]	10	Reserved
SOF [222]	1	Start of file/packet indicator for AXI-Stream. In H2D streaming, this bit triggers the start of data transfer on the AXI-Stream Manager interface. In D2H streaming, this bit is set in the Descriptor itself by the MWr TLP. *Note:* In the H2D streaming, both SOF and EOF can be set in the same descriptor (file size = payload count) or it can span multiple descriptor pages. *Note:* In the D2H streaming, if the user logic prematurely ends the data transfer by asserting d2h_axi_st_tlast in the middle of a descriptor data move then starts a next file/packet, the SOF bit in the next descriptor is set by the MWr TLP.
EOF [223]	1	End of file/packet indicator for AXI-Stream. In the H2D streaming, this bit causes the AXI-Stream Manager interface to assert h2d_axi_st_tlast, indicating the end of a file/packet. In the D2H streaming, this bit is set within the descriptor itself by a Writeback (if Writeback is enabled) when the user logic asserts d2h_axi_st_tlast, indicating the end of a packet. Along with the EOF bit, the MWr TLP also updates the actual received payload count (RX_PYLD_CNT) field of the last descriptor.
RSRVD [253:224]	30	Reserved
DESC_INVALID [254]	1	Indicates if the current descriptor content is valid or stale.
LINK [255]	1	Link = 0 Descriptor contains the source address, destination address and length. Link = 1 Descriptor contains the address of the next 4 KB page in the host memory containing the descriptors.

A.1.1.3.1. Support for Unaligned (or Byte-Aligned) Data Transfer

The GTS AXI MCDMA IP supports the following alignment modes for the descriptor source/destination address and payload count fields. Unaligned (or Byte-aligned) data transfer is not supported.

Table 62. Alignment Mode

SRC/DEST Address Alignment	Payload Count Alignment	Note
AXI-MM: Dword aligned. **AXI-S**: 64-byte aligned (or full data width aligned).	**AXI-MM**: Dword aligned. **AXI-S**: 64-byte aligned (exception: last descriptor of a packet/file).	Descriptors are 32-byte aligned.

A.1.1.3.2. Metadata Support

8-Byte Metadata

In AXI Streaming mode, once you select **Enable Metadata** support during the IP generation, the source and destination address fields in the existing descriptor structure are repurposed for metadata support. The following fields of the existing descriptor defined above have revised properties.

Table 63. Metadata for SRC_ADDR and DEST_ADDR Fields

Name	Width	Description
SRC_ADDR[63:0]	64	If Link bit = 0, then this field contains the source address. If the queue is H2D, then this field contains the address in the host memory. If the queue is D2H, then this is 8-Byte Metadata. If the link bit is set, then this contains the address of the next 4KB page in the host memory containing the descriptors.
DEST_ADDR[127:64]	64	Provided link = 0, this field means: If the queue is D2H, then this field contains the address in the host memory. If the queue is H2D, then this is 8 Byte-Metadata.

A.1.1.3.3. MSI-X/Writeback

The MSI-X and Writeback block updates the host with the current processed queue's head pointer and interrupt. Apart from a global MSI-X Enable and Writeback Enable for each queue, there is a provision to selectively enable or disable the MSI-X and Writeback on a per-descriptor basis. This feature can be used by applications to throttle the MSI-X/Writeback.

The table below shows the relationship between the global and per-descriptor MSI-X/Writeback Enables.

Table 64. Multichannel DMA Per-Descriptor Enable vs. Global MSI-X/Writeback Enable

Global Enable	Per-Descriptor Enable	MSI-X/Writeback Generation
1	1	On
1	0	Off
0	1	Off
0	0	Off

If enabled, a Writeback is sent to the host to update the status (completed descriptor ID) stored in the Q_CONSUMED_HEAD_ADDR location. In addition, for D2H streaming DMA, an additional MWr TLP is issued to the D2H descriptor itself when the IP's AXI-Stream Subordinate interface has received the first/last data from the user logic. It updates the D2H descriptor packet information fields such as start of a file/packet (SOF), end of a file/packet (EOF), and received payload count (RX_PYLD_CNT).

A.1.1.4. AXI4-Lite PIO Manager

The AXI4-Lite PIO Manager bypasses the DMA block and provides a way for the Host to do MMIO reads/writes to the CSR registers of the user logic. PCIe BAR2 is mapped to the AXI4-Lite PIO Manager. Any TLP targeting BAR2 is forwarded to the user logic. TLP address targeting the PIO interface should be 8-byte aligned. The PIO interface supports non-bursting 64-bit write and read transfers only.

Note: Do not attempt to perform 32-bit transactions on the PIO interface. Only 64-bit transactions are supported.

The AXI4-Lite PIO Manager is present only if you select the **Multi Channel DMA** User Mode for **MCDMA Settings** in the IP Parameter Editor. The AXI4-Lite PIO Manager is always present irrespective of the user interface type (AXI-S/AXI-MM) that you select.

PIO Address Mapping

The PIO interface address mapping is as follows: PIO address = {vf_active, pf [PF_NUM_W-1:0], vf [VF_NUM_W-1:0], address}

1. **vf_active**: This indicates that SR-IOV is enabled.
2. **pf [PF_NUM_W-1:0]:** Physical function number decoded from the PCIe header received from the HIP; PF_NUM_W, which is ($clog2(Number of PFs)), is the RTL design parameter selected by you so that the GTS AXI MCDMA IP only allocates the required number of bits on the AXI-MM side to limit the number of wires on the user interface.
3. **vf [VF_NUM_W-1:0]:** Virtual function number decoded from the PCIe header received from the HIP; VF_NUM_W, which is ($clog2(Number of VFs)), is the RTL design parameter selected by you so that the GTS AXI MCDMA IP only allocates the required number of bits on the AXI-MM side to limit the number of wires on the user interface.
4. **address:** Number of bits required for the BAR2 size requested across all Functions (PFs and VFs). Example: If BAR2 is selected as 4 MB, the address size is 22 bits.

A.1.1.5. AXI-MM Write (H2D) and Read (D2H) Manager

The AXI-MM interface is used to transfer data between the host and device through the memory-mapped interface. You can enable the Memory-Mapped interface by selecting the AXI-MM **User Interface** type in the IP Parameter Editor. The GTS AXI MCDMA IP supports a single AXI4 interface, where the H2D path is connected to the write channels while the D2H path is connected to the read channels of the interface.

AXI-MM Write (H2D)

The AXI-MM Write interface is used to write H2D DMA data to the AXI-MM subordinate in the user logic through the memory-mapped interface. The Write interface can issue AXI-MM write commands with a maximum of 512B burst size. The data width of this interface varies based on the **PCIe Mode** IP parameter configuration: 128-bit (Gen3 1x4) or 256-bit (Gen4 1x4). The readyAllowance of this port is enabled, allowing the interface to transfer data up to N additional write command cycles after the ready signal has been deasserted. The corresponding AXI-MM subordinate interface in the user logic supports this attribute to avoid data loss. The value of <N> for the H2D AXI-MM Interface is as follows:

- 256-bit data width is 32.
- 128-bit data width is 64.

AXI-MM Read (D2H)

The AXI-MM Read interface is used to read D2H DMA data from the AXI-MM subordinate in the user logic through the memory-mapped interface. The Read interface can issue AXI-MM read commands with a maximum of 512B burst size. The data width of this interface varies based on the **PCIe Mode** IP parameter configuration: 128-bit (Gen3 1x4) or 256-bit (Gen4 1x4). The readyAllowance of this port is enabled, allowing the interface to transfer up to N additional write command cycles after the ready signal has been deasserted. The corresponding AXI-MM subordinate interface in the user logic supports this attribute to avoid data loss. The value of <N> for the D2H AXI-MM Interface is as follows:

- 256-bit data width is 32.
- 128-bit data width is 64.

A.1.1.6. AXI-Stream Manager (H2D) and Subordinate (D2H)

The GTS AXI MCDMA IP provides AXI-Stream interfaces for transferring DMA data between the host and device. The AXI-Stream Manager interface is used to move H2D DMA data to the external user logic. The AXI-Stream Subordinate interface is used to move D2H DMA data to the host. You can enable the Streaming interface by selecting the AXI-S **User Interface** type in the IP Parameter Editor.

Note: As the GTS AXI MCDMA IP implements separate header and data interfaces, the **PCIe AXI-S Sideband Header** parameter of the GTS AXI Streaming IP must be enabled.

When streaming the DMA data, the packet (file) boundary is indicated by the SOF and EOF bits of the descriptor. Channel interleaving is not supported. A channel switch on the AXI-Stream interface can only happen on a packet boundary.

For the AXI-Stream interface, you can also optionally enable 8-byte metadata that contains the metadata for your user application. When enabled in the IP Parameter Editor, the H2D descriptor destination address field is replaced with metadata and the D2H descriptor source address field is replaced with metadata. The metadata is presented on the h2d/d2h_axi_st_tuser_metadata signal and valid on the first clock cycle of the data transfer.

A.1.1.7. User MSI-X

User MSI-X is arbitrated along with the H2D/D2H MSI-X/Writeback requests and is handled the same way as the others post the arbitration.

Each DMA Channel is allocated 4 MSI-X vectors:

- 2'b00: H2D DMA Vector
- 2'b01: H2D Event Interrupt
- 2'b10: D2H DMA Vector
- 2'b11: D2H Event Interrupt

2'b00 and 2'b10 are used to address Descriptor completion related interrupts (DMA operation MSI-X) on both H2D and D2H paths.

2'b01 and 2'b11 are used for the user MSI-X.

User MSI-X can be requested via the user event MSI-X request interface.

A.1.1.8. User Function Level Reset (FLR)

When the DMA engine receives Function Level Resets from the Host via the GTS AXI Streaming IP, the reset requests are propagated to the downstream logic via this interface. In addition to performing resets to its internal logic, the FLR interface waits for an acknowledgment from the user logic for the reset request before it issues an acknowledgement to the GTS AXI Streaming IP.

A.1.1.9. Control and Status Registers

The GTS AXI MCDMA IP provides 4 MB of register space that is internally mapped to PCIe BAR0. There is no user interface to access the register space. The CSR block contains all the required registers to support the DMA operations. This includes the QCSR space for individual queue control, MSI-X for interrupt generations, and GCSR for general global information.

The following table shows the 4 MB space mapped for each function in PCIe configuration space through BAR0.

Table 65. Control Registers

Address Space	Range	Size	Description
QCSR (D2H, H2D)	22'h00_0000 - 22'h0F_FFFF	1 MB	Individual queue control and status registers, up to 256 D2H and 256 H2D queues.
MSI-X (Table and PBA)	22'h10_0000 - 22'h1F_FFFF	1 MB	MSI-X Table and PBA space.
GCSR	22'h20_0000 - 22'h2F_FFFF	1 MB	General DMA control and status registers. Only for PF0.
Reserved	22'h30_0000 – 22'h3F_FFFF	1MB	Reserved

Note: For more information on the Control and Status registers, refer to the *Registers* Appendix.

A.1.2. Bursting Master (BAM)

The BAM bypasses the DMA engine of the GTS AXI MCDMA IP and provides a way for the Host to perform bursting PIO reads/writes to the user logic. The BAM converts memory read and write TLPs initiated by the remote link partner and received over the PCIe link into AXI-MM burst read and write transactions and sends back CplD TLPs for read requests it receives. The BAM supports both 128-bit and 256-bit data widths to achieve bandwidths required for Gen3 x4 and Gen4 x4. The completions are always expected in order from the user logic/Platform Designer fabric. The BAM supports bursts of up to 512 bytes and up to 32 outstanding read requests.

BAM Address Mapping

You can choose to map any BAR register other than BAR0 of the physical function to the BAM for the user application. The BAM interface address mapping is as follows:

BAM address = {vf_active, pf, vf, bar_num, bam_addr}

1. **vf_active:** This indicates that SRIOV is enabled.
2. **pf [PF_NUM-1:0]:** Physical function number decoded from the PCIe header received from the HIP. PF_NUM, which is ($clog2(Number of PFs)), is the RTL design parameter you select such that the IP only allocates the required number of bits on the AXI-MM side to limit the number of wires on the user interface. Example: If the number of PFs you selected is 4, the PF_NUM is 2.
3. **vf [VF_NUM-1:0]:** Virtual function number decoded from the PCIe header received from the HIP. VF_NUM, which is ($clog2(Number of VFs)), is the RTL design parameter you select such that the IP only allocates the required number

of bits on the AXI-MM side to limit the number of wires on the user interface. Example: If the total number of VFs across all PFs you selected is 32, the VF_NUM is 5.

4. **bar_num [2:0]:** This denotes the BAR number where the AXI-Stream transaction was received.
5. **bam_addr [ADDR_SIZE-1:0]:** Lower address based on the maximum aperture size amongst all the BARs. Example: If BAR3 is selected as 16 MB and BAR2 is 4 GB, the ADDR_SIZE = 32 corresponding to BAR2.

Example: If the transaction was received for BAR3 (maximum aperture of 4 GB) of PF2/VF1 where you have enabled only 3 PFs and 25 VFs, the BAM address is {1'b1, 2'b10, 5'b00001, 3'b011, bam_addr[31:0]}.

Note: In Root Port mode, the AXI-MM address output from BAM is the same as the one received on the GTS AXI Streaming IP AXI Streaming interface.

A.1.3. Bursting Slave (BAS)

The AXI-MM Bursting Slave module translates AXI-MM Read and Write transactions from the user logic to PCI Express Mrd and Mwr TLPs. The returned PCI Express CplD packets are translated to AXI-MM Read Data channel as a response to the AXI-MM read request transaction.

The BAS supports both 128-bit and 256-bit data widths to achieve the bandwidths required for Gen3 x4 and Gen4 x4. It supports bursts of up to 512 bytes and up to 64 outstanding read requests.

Completion Reordering

The AXI-MM BAS interface is a subordinate interface to the User AXI-MM interface. The User AXI-MM can initiate AXI-MM reads to the host interface and this translates to BAS Non-Posted (NP) packet interface signals. The BAS module keeps track of the initiated NP requests and tracks against the completions received from the PCIe on the scheduler completion packet interface.

Since the completion from the PCIe can come out of order, the completion reordering module ensures the returned completions are reordered against the pending requests and sent in the same order on the AXI-MM interface.

A.1.4. MSI Interrupt

A.1.4.1. Endpoint MSI Support Through BAS

MSI enables a device Function to request service by writing a system-specified data value to a system-specified address using a single-dword Memory Write transaction. System software initializes the message address and message data (referred to as the "vector") during device configuration, allocating one or more vectors to each MSI-capable Function.

When you enable the MSI Capability in the Endpoint BAS or BAM+BAS mode, the IP exposes the MSI request interface to the user logic. When you issue an MSI request through this interface, the internal Interrupt Controller receives inputs such as function number and MSI number from the user logic and forwards the request to the BAS module. The BAS receives MSI signaling from the interrupt controller and generates an MSI.

A.1.4.2. MSI Interrupt Controller

The MSI Interrupt Controller has the necessary storage to hold the MSI address/data for that specific function and user vector number. If the MSI is masked for a specific function, the MSI Interrupt Controller does not send the MSI for that function.

When you request the generation of the MSI, you need to provide the MSI vector (number) and user function information which the MSI Interrupt Controller indexes to get the MSI address/data and send this information to the BAS. The MSI capability message control register [6:4] selects the number of the MSI vectors per function. The MSI vector input is used to manipulate the MSI data LSB bits as per the PCIe Specification.

Figure 34. MSI Request Timing Diagram

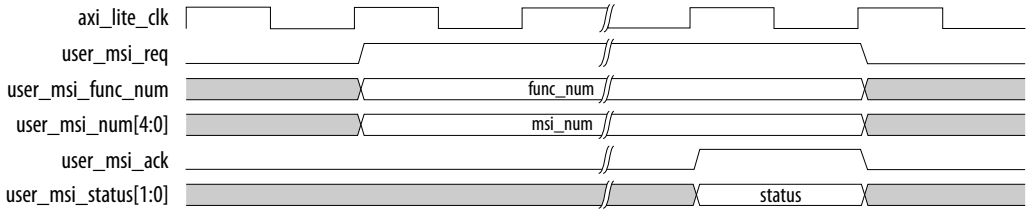
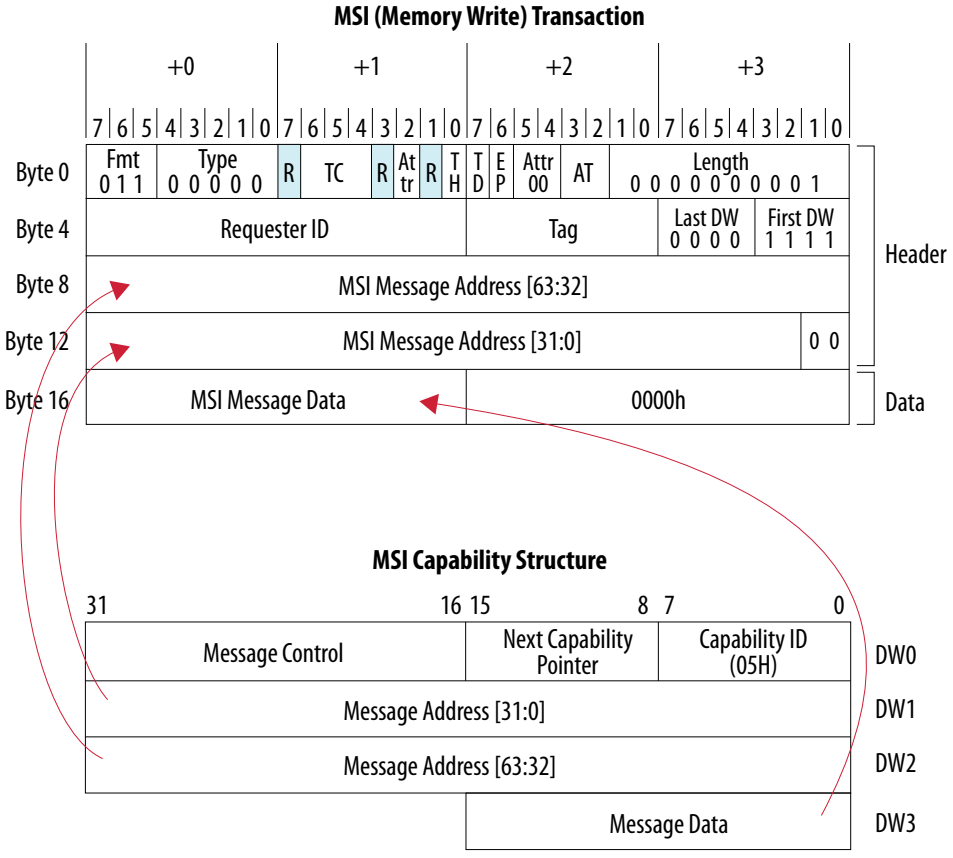


Figure 35. MSI Memory Write Transaction



Refer to User Event MSI for more information on the interface signals.

A.1.5. Configuration Slave (CS)

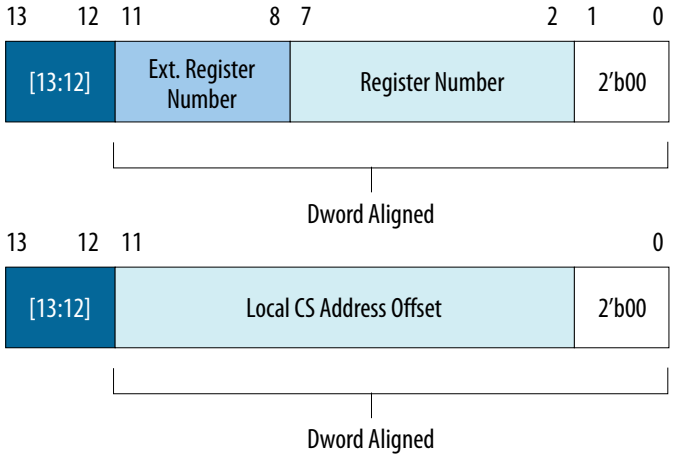
This interface is applicable only in Root Port mode. The Configuration Slave (CS) is an AXI-MM non-bursting interface and essentially converts single-cycle, AXI-MM read and write transactions into AXI-Stream reads and writes of PCIe configuration TLPs to be sent to the GTS AXI Streaming IP (to be sent over the PCIe link). This module also processes the completion TLPs (Cpl and CplD) it receives in return.

The CS module converts the AXI-MM request into a configuration TLP with a fixed TAG value (decimal 255) assigned to it and sends it to scheduler. One unique TAG is sufficient as it does not support more than one outstanding transaction. This unique TAG helps in rerouting the completions to the CS module. On the receiving side, the TLP RX scheduler parses the completion field to decode the completion TLP on a fixed TAG and route the transaction over to CS.

A.1.5.1. 14-bit AXI-MM Address Format

The Configuration Slave supports a 14-bit address format as shown in the figure below.

Figure 36. 14-bit CS Address Format



The two most significant bits [13:12] determine whether address [11:0] are used to form a Configuration TLP sent downstream or used to write to/read from the local Configuration Slave registers.

Table 66. CS Address Bits [13:12] Definition

Bits [13:12]	Description
2'b00	Configuration TLP Type 0
2'b01	Configuration TLP Type 1
2'b10	Local CS address space 14'h2000 – 14'h2FFF (BDF register, etc.)
2'b11	Local CS address space 14'h3000 – 14'h3FFF (ATT tables)

The following is a list of the local CS registers.

Table 67. Local CS Registers Supported in 14-bit Address Mode

Local CS Address Offset	Name	Access	Comment
14'h2000	Scratch Pad Register	RW	
14'h2004	BDF Register	RW	{Bus[7:0], Device[4:0], Function[2:0]}
14'h3000 – 14'h3FFF	ATT for BAS	RW	Address range for the Address Translation Table. **Note:** Refer to Root Port Address Translation Table Enablement for information on the ATT programming example.

A.1.5.2. Configuration Access Mechanism

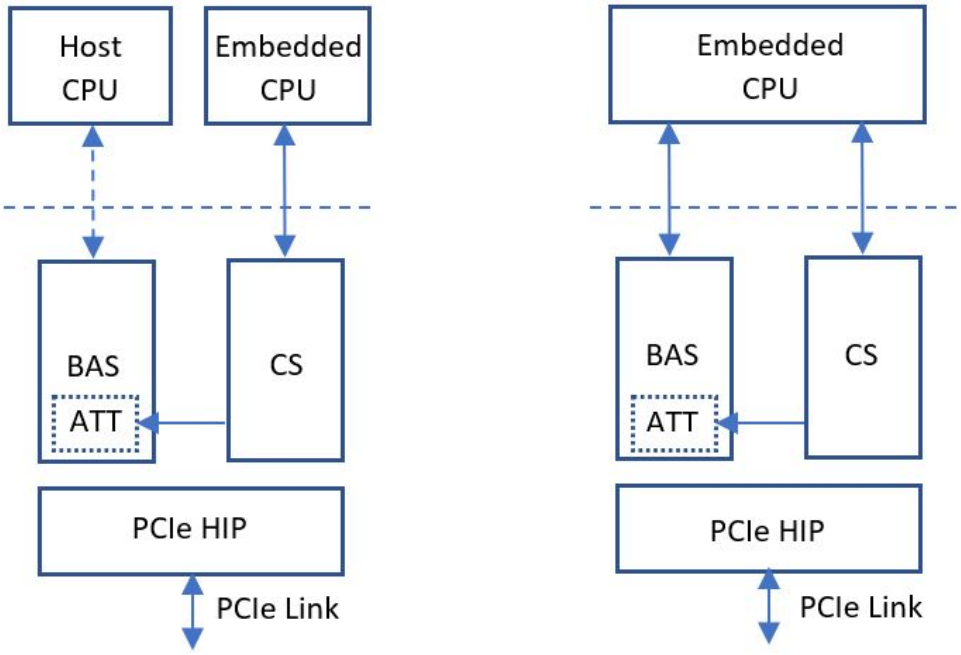
Table 68. Configuration Access Mechanism

Access	Access Mechanism
EP Configuration Space Write	Two Writes: • Write BDF information to 0x2004 (with address bit[13] set to 1). • Write to EP Register address (with address bit[13] set to 0) with the actual data.
EP Configuration Space Read	• Write BDF information to 0x2004 (with address bit[13] set to 1). • Read from EP Register address (with address bit[13] set to 0). • Configuration TLP Type1/Type0 is based on the address bit[12]. • CplD data is available on the read data channel.

A.1.6. Root Port Address Translation Table Enablement

The Root Port Address Translation Table (ATT) translates the address driven by the AXI-MM Manager connected to the Bursting Slave (BAS). This allows the AXI-MM Manager interface with a smaller address bus to access a larger address space. For example, if the embedded CPU with the smaller address bus is the AXI-MM Manager, ATT enables the embedded CPU to access the entire 64-bit PCIe address space. If the host CPU is the master, ATT can be used to translate the host address bus. The following figure shows example use cases.

Figure 37. Example Use Cases with the Root Port Address Translation Table (ATT)



Note: For information about the CS register offset used to program the ATT, refer to Local CS Registers Supported.

The IP provides the following parameters in the IP Parameter Editor (**MCDMA Settings**) that allow you to select the address mapping when you enable ATT.

- **ATT Table Address Width (1-9)**: Sets the depth of the ATT. Depth is equal to 2 to the power of the number entered.
- **ATT Window Address Width (10-63)**: Sets the number of BAS address bits to be used directly.

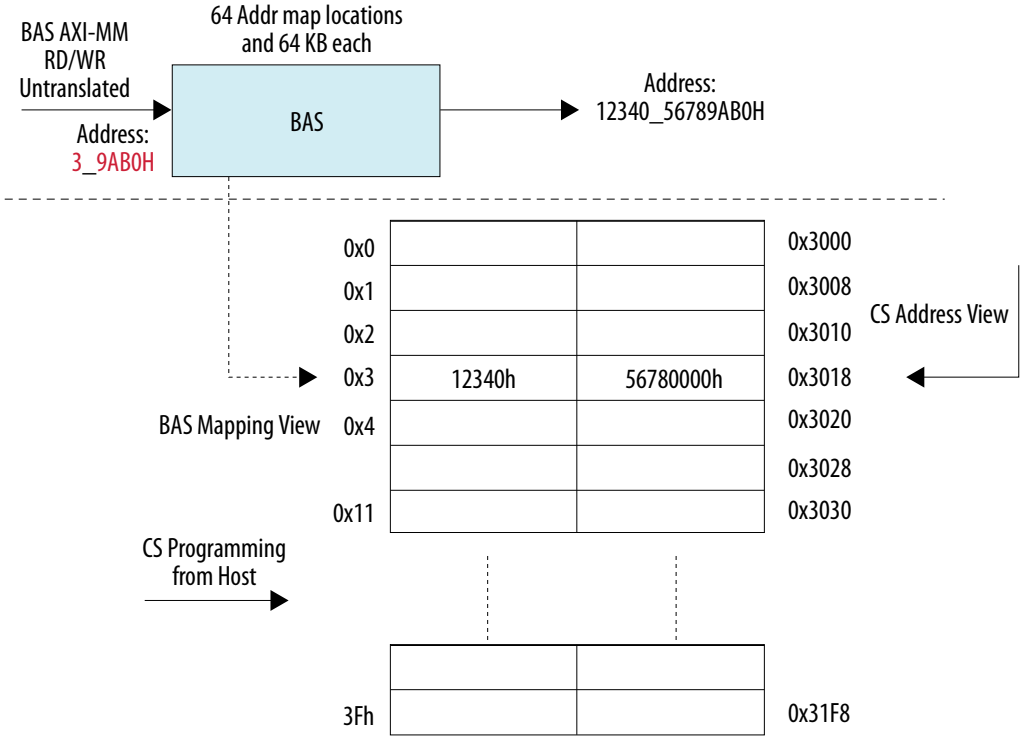
When address mapping is disabled, the AXI-MM subordinate address is used as-is in the resulting PCIe TLPs. When address mapping is enabled, burst of transactions on the AXI-MM subordinate interfaces must not cross address mapping page boundaries. This requirement means (address + 32 * burst count) <= (page base address + page size). Host software is expected to guarantee this in the Root Port mode. The BAS/CS does not have any mechanism to report the error if this requirement is violated.

When address mapping is enabled, the AXI-MM lower-order address bits are passed through to the PCIe TLPs unchanged and are ignored in the address mapping table. The number of lower-order address bits that are passed through defines the size of the page and is set by the **ATT Window Address Width** parameter in the IP Parameter Editor. If bits [63:32] of the resulting PCIe address are zero, TLPs with 32-bit wide addresses are created as required by the PCI Express standard.

For example, if you define 16 address mapping windows of 64 KB each at configuration time and program the local CS addresses 0x3018 and 0x301C with 0x56780000 and 0x00012340 respectively, a read or write transaction to address 0x39AB0 on the bursting AXI-MM subordinate interface gets transformed into a memory read or write TLP accessing PCIe address 0x0001234056789AB0.

In the figure below, the ATT depth is 64. This is set by **ATT Table Address Width =** 6 (2^6= 64 deep). The address pass-through width is 16. This is set by **ATT Window Address Width** = 16. This means BAS forwards the lower 16-bit address as is. The upper 6 bits are used to select the ATT entry. In the example, the ATT entry selection is 0x03.

Figure 38. ATT Enablement Example



The Flow is:

1. The host software is expected to program all the ATT registers in the CS for all enabled locations.
2. When the AXI-MM transaction is received by BAS, it detects the address being 32-bit or 64-bit and enables the translation logic in CS as shown above. If the address translation gets enabled, the BAS logic retrieves the translated data from the ATT table as shown in the figure above.

A.1.7. Control and Status Register Interface

The Control and Status Register interface is an AXI4-lite Manager interface with a 20-bit address bus and a 32-bit data bus. The GTS AXI MCDMA IP uses this interface to access registers implemented in the GTS AXI Streaming IP, allowing the user application to access PCIe configuration space registers of all Functions as well as soft register space registers implemented in the GTS AXI Streaming IP. This interface can be used in Endpoint and Root Port modes. This interface is connected to the Control and Status Register Responder Interface of the GTS AXI Streaming IP.

For the user logic to access the registers of the GTS AXI Streaming IP, the **Enable HIP Reconfiguration Interface** parameter can be enabled to expose an AXI4-lite Subordinate interface. The application accesses PCIe configuration space registers through this interface by adding an offset of 0x80000 to the actual physical address of the PCIe configuration space address. Refer to the Register Address Map in the GTS AXI Streaming Intel FPGA IP for PCI Express User Guide.

Note: After a warm reset or cold reset, changes made to the configuration registers of the GTS AXI Streaming IP via this interface are lost and these registers revert to their default values.

A.1.8. Configuration Intercept Interface

The Configuration Intercept Interface (CII) allows the application logic to detect the occurrence of a Configuration (CFG) request on the link and to modify its behavior. The application logic can use the CII to:

- Delay the processing of a CFG request by the controller. This allows the application to perform any housekeeping task first.
- Overwrite the data payload of a CfgWr request. The application logic can also overwrite the data payload of a CfgRd Completion TLP.

The CII interface gives an update only when reconfiguration happens to the writable PCIe configuration space registers. If you want to get the values of the PCIe configuration space registers which are read-only, you must use the HIP reconfiguration interface.

This interface is applicable only when operating in Endpoint mode and can be enabled with the **Enable Configuration Intercept Interface** parameter. You must also make sure that the corresponding **Enable Configuration Intercept Interface** parameter of the GTS AXI Streaming IP is enabled, and the interface signals are connected to this interface.

B. Appendix B: Registers

The GTS AXI Multichannel DMA IP for PCI Express provides configuration, control and status registers to support the DMA operations including:

- D2H and H2D Queue control and status (QCSR)
- MSI-X Table and PBA for interrupt generation
- General/global DMA control (GCSR)

The register space is internally mapped to PCIe BAR0 of a function. There is no user interface to access the register space.

Note: GCSR is only for PF0.

Note: Read/Write access to the CSR address space is limited to 32 bits at a time through the Mrd/Mwr commands from the host.

The following table shows a 4 MB aperture space mapped for PF0 in the PCIe configuration space through BAR0.

Table 69. Multichannel DMA CSR Address Space

Address Space Name	Range	Size	Description
QCSR (D2H, H2D)	22'h00_0000 - 22'h0F_FFFF	1MB	Individual queue control registers. Up to 256 D2H and 256 H2D queues.
MSI-X (Table and PBA)	22'h10_0000 - 22'h1F_FFFF	1MB	MSI-X Table and PBA space.
GCSR	22'h20_0000 - 22'h2F_FFFF	1MB	General DMA control and status registers.
Reserved	22'h30_0000 – 22'h3F_FFFF	1MB	Reserved

The following table shows how QCSR registers for each DMA channel are mapped with 1 MB space of QCSR.

Table 70. QCSR Address Space

Address Space Name	Size	DMA Channel	Size	Description
QCSR (D2H)	512 KB	DMA Channel 0	256 B	QCSR for DMA channel 0
		DMA Channel 1	256 B	QCSR for DMA channel 1
		….	….	….
		DMA Channel N	256 B	QCSR for DMA channel N
QCSR (H2D)	512 KB	DMA Channel 0	256 B	QCSR for DMA channel 0
		DMA Channel 1	256 B	QCSR for DMA channel 1
		….	….	….
		DMA Channel N	256 B	QCSR for DMA channel N

B.1. Queue Control

The QCSR space contains queue control and status information. This register space of 1 MB can support up to 256 H2D and 256 D2H queues, where each queue is allocated 256 bytes of register space. The memory space allocated to each function is enough for each function to have allocated all the DMA Channels. However, the actual number depends on the parameters input at IP generation time.

Address [7:0]: Registers for the queues

Address [18:8]: Queue number

Address [19]: 0 = D2H, 1=H2D

The following registers are defined for the H2D/D2H queues. The base addresses for H2D and D2H are different, but registers (H2D and D2H) have the same address offsets.

Table 71. Queue Control Registers

Register Name	Address Offset	Access Type	Description
Q_CTRL	8'h00	R/W	Control Register
RESERVED	8'h04		Reserved
Q_START_ADDR_L	8'h08	R/W	Lower 32 bits of the queue base address in system memory. This is the beginning of the linked-list of 4KB pages containing the descriptors.
Q_START_ADDR_H	8'h0C	R/W	Upper 32 bits of the queue base address in system memory. This is the beginning of the linked-list of 4KB pages containing the descriptors.
Q_SIZE	8'h10	R/W	Number of maximum entries in a queue. Powers of 2 only.
Q_TAIL_POINTER	8'h14	R/W	Current pointer to the last valid descriptor queue entry in the host memory.
Q_HEAD_POINTER	8'h18	RO	Current pointer to the last descriptor that was fetched. Updated by the Descriptor Fetch Engine.
Q_COMPLETED_POINTER	8'h1C	RO	Last completed pointer after DMA is done. Software can poll this for status if Writeback is disabled.
Q_CONSUMED_HEAD_ADDR_L	8'h20	R/W	Lower 32 bits of the system address where the ring consumed pointer is stored. This address is used for consumed pointer writeback.
Q_CONSUMED_HEAD_ADDR_H	8'h24	R/W	Upper 32 bits of the system address where the ring consumed pointer is stored. This address is used for consumed pointer writeback.
Q_BATCH_DELAY	8'h28	R/W	Delay the descriptor fetch until the time elapsed from a prior fetch exceeds the delay value in this register to maximize fetching efficiency.

continued...

Register Name	Address Offset	Access Type	Description
Q_DATA_DRP_ERR_CTR	8'h40	RW	Data drop error counter.
Q_PYLD_CNT	8'h44	R/W	20-bit payload count. DMA payload size in bytes and must be 64-byte aligned. Maximum 1 MB, with 20'h0 indicating 1 MB. The value set in this register must be the same as used by the Host software to populate the PYLD_CNT field of descriptors for the respective channel. Applicable only for D2H AXI-S port mode. Unused in all other modes.
Q_RESET	8'h48	R/W	Request reset for the queue by writing 1'b1 to this register, and poll for a value of 1'b0 when reset has been completed by hardware. Hardware clears this bit after completing the reset of a queue. Similar process occurs when FLR reset is detected for a VF.

The following registers are defined for each implemented H2D and D2H queue. The total QCSR address space for each H2D/D2H is 256B and requires 8 bits of address.

Table 72. Q_CTRL (Offset 8'h00)

Bits[31:0]	Name	R/W	Default	Description
[31:10]	rsvd			Reserved
[9]	q_intr_en	R/W	0	If set, upon completion, generate an MSI-X interrupt for the queue.
[8]	q_wb_en	R/W	0	If set, upon completion, do a write-back for the queue.
[7:1]	rsvd			Reserved
[0]	q_en	R/W	0	Enable. Once it is enabled, the DMA starts fetching pending descriptors and executing them.

Table 73. Q_START_ADDR_L (Offset 8'h08)

Bits[31:0]	Name	R/W	Default	Description
[31:0]	q_strt_addr_l	R/W	0	After software allocates the descriptor ring buffer, it writes the lower 32-bit allocated address to this register. The descriptor fetch engine uses this address and the pending head/tail pointer to fetch the descriptors.

Table 74. Q_START_ADDR_H (Offset 8'h0C)

Bits[31:0]	Name	R/W	Default	Description
[31:0]	q_strt_addr_h	R/W	0	After software allocates the descriptor ring buffer, it writes the upper 32-bit allocated address to this register. The descriptor fetch engine uses this address and the pending head/tail pointer to fetch the descriptors.

Table 75. Q_SIZE (Offset 8'h10)

Bits[31:0]	Name	R/W	Default	Description
[31:5]	rsvd			Reserved
[4:0]	q_size	R/W	1	Size of the descriptor ring in power of 2 and maximum value of 16. The unit is number of descriptors. Hardware defaults to using a value of 1 if an illegal value is written. A value of 1 means a queue size of 2 (2^1). A value of 16 (0x10) means a queue size of 64K (2^16).

Table 76. Q_TAIL_POINTER (Offset 8'h14)

Bits[31:0]	Name	R/W	Default	Description
[31:16]	rsvd			Reserved
[15:0]	q_tl_ptr	R/W	0	After software sets up a last valid descriptor in the descriptor buffer, it programs this register with the position of the last (tail) valid descriptor that is ready to be executed. The DMA Descriptor Engine fetches descriptors from the buffer up to this position of the buffer. *Note:* Writing 0x0 to Q_TAIL_POINTER is illegal.

Table 77. Q_HEAD_POINTER (Offset 8'h18)

Bits[31:0]	Name	R/W	Default	Description
[31:25]	rsvd			Reserved
[24]	q_err_during_desc_fetch	RO	0	Error during descriptor fetch, e.g. Cmplto/UR.
[23:16]	rsvd			Reserved
[15:0]	q_hd_ptr	R/W	0	After the DMA Descriptor Fetch Engine fetches the descriptors from the descriptor buffer, up to the tail pointer, it updates this register with that last fetched descriptor position. The fetch engine only fetches descriptors if the head and tail pointers are not equal.

Table 78. Q_COMPLETED_POINTER (Offset 8'h1C)

Bits[31:0]	Name	R/W	Default	Description
[31:16]	rsvd			Reserved
[15:0]	q_cmpl_ptr	R/W	0	This register is updated by hardware to store the last descriptor position (pointer) that DMA has completed, and all data for that descriptor and previous descriptors has arrived at the intended destinations. Software can poll this register to find out the status of the DMA for a specific queue.

Table 79. Q_CONSUMED_HEAD_ADDR_L (Offset 8'h20)

Bits[31:0]	Name	R/W	Default	Description
[31:0]	q_cnsm_hd_addr_l	R/W	0	Software programs this register with the lower 32-bit address location where the writeback targets after DMA is completed for a descriptor with writeback bit enabled.

Table 80. Q_CONSUMED_HEAD_ADDR_H (Offset 8'h24)

Bits[31:0]	Name	R/W	Default	Description
[31:0]	q_cnsm_hd_addr_h	R/W	0	Software programs this register with the upper 32-bit address location where the writeback targets after DMA is completed for a set of descriptors.

Table 81. Q_BATCH_DELAY (Offset 8'h28)

Bits[31:0]	Name	R/W	Default	Description
[31:20]	rsvd			Reserved
[19:0]	q_batch_dscr_delay	R/W	0	Software programs this register with the amount of time between fetches for descriptors. Each unit is 2ns.

Table 82. Q_DATA_DRP_ERR_CTR (Offset 8'h40)

Bits[31:0]	Name	R/W	Default	Description
[31:21]	rsvd			Reserved
[20]	q_d2h_data_drop_err_st	R/W	0	D2H data drop error status for a channel. This bit is set when packets are received for a channel that is not enabled or when packets are dropped for a channel that is enabled.
[19]	rsvd			Reserved
[18]	rsvd			Reserved
[17]	q_data_err_mm	R/W	0	Data error status for a channel in AXI-MM mode.
[16]	q_data_err_st	R/W	0	Data error status for a channel in AXI-S mode.
[15:0]	q_data_drp_err_cnt	R/W	0	Data drop error count. The error count is updated by the hardware and count increments whenever the D2H side drops a packet. The error count saturates at all 1's and needs software to clear. Data drop example cases: • Packets received for a channel that is not enabled. • Packets received for a channel that is enabled, but does not have TID update. *Note:* For packets received for a channel that is enabled, TID update is done by software. However, the descriptors that are not yet fetched are not dropped.

Table 83. Q_PYLD_CNT (Offset 8'h44)

Bits[31:0]	Name	R/W	Default	Description
[31:20]	rsvd			Reserved
[19:0]	q_pyld_cnt	R/W	0	20-bit payload count. DMA payload size in bytes. Maximum 1 MB, with 20'h0 indicating 1 MB. This value must be the same as set in the descriptors payload count field. Applicable only for D2H AXI-S port mode. Unused in all other modes.

Table 84. Q_RESET (Offset 8'h48)

Bits[31:0]	Name	R/W	Default	Description
[31:1]	rsvd			Reserved
[0]	q_reset	R/W	0	Request reset for the queue by writing 1'b1 to this register, and poll for a value of 1'b0 when reset has been completed by hardware. Hardware clears this bit after completing the reset of a queue.

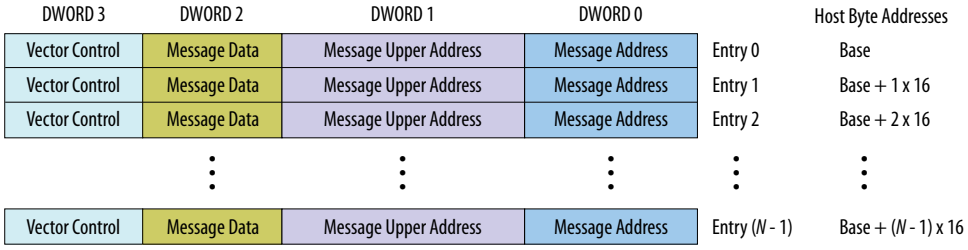
B.2. MSI-X Memory Space

The MSI-X Table and Pending Bit Array (PBA) memory are mapped to the second MB space of the register address space. The actual amount of memory depends on the GTS AXI MCDMA IP configuration.

MSI-X Table

Each entry (vector) is 16 bytes (4 DWORDs) and is divided into Message Address, Data, and Mask (Vector Control) fields as shown in the figure below. Though the actual required space is smaller, the MSI-X Table is mapped to a 512 KB of space.

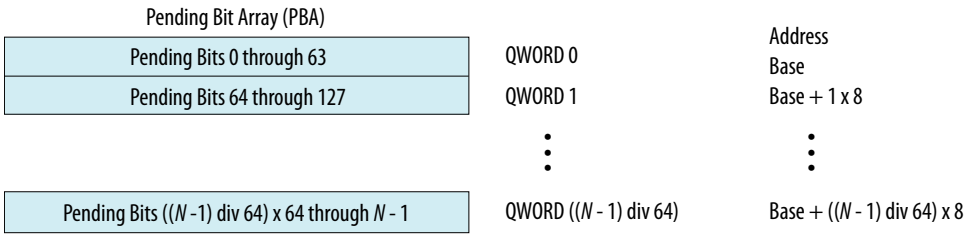
Figure 39. MSI-X Table Structure



MSI-X PBA

The MSI-X PBA memory space is mapped to a 512 KB region. The actual amount of memory depends on the IP configuration. The Pending Bit Array contains the Pending bits, one per MSI-X Table entry, in an array of QWORDs (64 bits). The PBA format is shown below.

Figure 40. MSI-X PBA Structure



Each DMA Channel is allocated 4 MSI-X vectors:

- 2'b00: H2D DMA Vector
- 2'b01: H2D Event Interrupt
- 2'b10: D2H DMA Vector
- 2'b11: D2H Event Interrupt

B.3. Control Register (GCSR)

This space contains global control/status registers that control the DMA operation. Access to this register set is restricted to PF0 only.

Table 85. Control Registers

Register Name	Address Offset	Access Type	Description
RESERVED	8'h00 - 8'h04		Reserved
WB_INTR_DELAY	8'h08	R/W	Delay the writeback and/or the MSI-X interrupt until the time elapsed from a prior writeback/ interrupt exceeds the delay value in this register.
RESERVED	8'h0C – 8'h6F		Reserved
VER_NUM	8'h70	RO	GTS AXI Multichannel DMA IP for PCI Express version number.
SW_RESET	9'h120	RW	Write this register to issue a GTS AXI MCDMA IP reset without disturbing the PCI Express link. This resets all queues and erases all the context. Can be issued only from PF0.

Table 86. WB_INTR_DELAY (Offset 8'h08)

Bits[31:0]	Name	R/W	Default	Description
[31:20]	rsvd			Reserved
[19:0]	wb_intr_delay	R/W	0	Delay the writeback and/or the MSI-X interrupt until the time elapsed from a prior writeback/interrupt exceeds the delay value in this register. Each unit is 2ns.

Table 87. VER_NUM (Offset 9'h070)

Bits[31:0]	Name	R/W	Default	Description
[31:24]	rsvd			Reserved
[23:16]	MAJOR_VER	RO	0	Major version number of the GTS AXI MCDMA IP.
[15:8]	UPDATE_VER	RO	0	Update version number of the GTS AXI MCDMA IP.
[7:0]	PATCH_VER	RO	0	Patch version number of the GTS AXI MCDMA IP.

The IP version number is defined using MAJOR_VER.UPDATE_VER.PATCH_VER format. For information about the GTS AXI MCDMA IP version number, refer to the IP Revision History.

Table 88. SW_RESET (Offset 9'h120)

Bits[31:0]	Name	R/W	Default	Description
[31:1]	rsvd			Reserved
[0]	SW_RESET	RW	0	Set this bit to issue a GTS AXI MCDMA IP reset without disturbing the PCIe link. This resets all queues and erases all the context. Issued only from PF0.

C. Document Revision History for the GTS AXI Multichannel DMA IP for PCI Express*

Document Version	Quartus Prime Version	IP Version	Changes
2025.05.06	25.1	1.0.0	Initial release.