

Multi Channel DMA FPGA IP for PCI Express* User Guide

Updated for Quartus® Prime Design Suite: **25.1.1**



Online Version



Send Feedback

UG-20297

683821

2025.08.04

Contents

1. Before You Begin.....	5
1.1. Terms and Acronyms.....	5
1.2. Known Issues.....	6
2. Introduction.....	8
2.1. Multi Channel DMA IP for PCI Express Features.....	9
2.1.1. Endpoint Mode.....	9
2.1.2. Root Port Mode.....	10
2.2. Device Family Support.....	11
2.3. Recommended Speed Grades.....	12
2.4. Resource Utilization.....	12
2.5. Release Information.....	16
3. Functional Description.....	17
3.1. Multi Channel DMA.....	18
3.1.1. H2D Data Mover.....	18
3.1.2. D2H Data Mover.....	19
3.1.3. Descriptors.....	21
3.1.4. Avalon-MM PIO Master.....	25
3.1.5. Avalon-MM Write (H2D) and Read (D2H) Master.....	26
3.1.6. Avalon-ST Source (H2D) and Sink (D2H).....	27
3.1.7. User MSI-X.....	28
3.1.8. User Functional Level Reset (FLR).....	29
3.1.9. Control Registers.....	29
3.2. Bursting Avalon-MM Master (BAM).....	29
3.3. Bursting Avalon-MM Slave (BAS).....	30
3.4. MSI Interrupt.....	31
3.4.1. Endpoint MSI Support through BAS	31
3.4.2. MSI Interrupt Controller.....	32
3.5. Config Slave (CS).....	34
3.5.1. 29 bit AVMM address format.....	34
3.5.2. 14 bit AVMM address format.....	35
3.5.3. Configuration Access Mechanism.....	36
3.6. Root Port Address Translation Table Enablement.....	36
3.7. Hard IP Reconfiguration Interface.....	38
3.8. Config TL Interface.....	39
3.9. Configuration Intercept Interface (EP Only).....	39
3.10. Data Mover Only.....	39
3.10.1. H2D Data Mover.....	41
3.10.2. D2H Data Mover.....	44
3.10.3. Application Specific Bits.....	46
4. Interface Overview.....	47
4.1. Port List.....	48
4.1.1. Port List (H-Tile).....	48
4.1.2. Port List (P-Tile) (F-Tile) (R-Tile).....	49
4.2. Clocks.....	50
4.2.1. F-Tile System PLL Reference Clock Requirements.....	51

4.3. Resets.....	52
4.4. Multi Channel DMA.....	53
4.4.1. Avalon-MM PIO Master.....	53
4.4.2. Avalon-MM Write Master (H2D).....	54
4.4.3. Avalon-MM Read Master (D2H).....	55
4.4.4. Avalon-ST Source (H2D).....	56
4.4.5. Avalon-ST Sink (D2H).....	57
4.4.6. User Event MSI-X Interface.....	57
4.4.7. User Functional Level Reset (FLR) Interface.....	58
4.5. Bursting Avalon-MM Master (BAM) Interface.....	58
4.6. Bursting Avalon-MM Slave (BAS) Interface.....	59
4.7. Legacy Interrupt Interface.....	61
4.8. Hot Plug Interface (RP only).....	62
4.9. MSI Interface.....	63
4.10. Config Slave Interface (RP only)	64
4.11. Hard IP Reconfiguration Interface.....	64
4.12. Config TL Interface.....	65
4.13. Configuration Intercept Interface (EP Only).....	65
4.14. Data Mover Interface.....	66
4.14.1. H2D Data Mover Interface.....	66
4.14.2. D2H Data Mover Interface.....	67
4.15. Hard IP Status Interface.....	67
4.16. Precision Time Management (PTM) Interface.....	68
5. Parameters (H-Tile).....	70
5.1. IP Settings.....	70
5.1.1. System Settings.....	70
5.1.2. MCDMA Settings.....	71
5.1.3. Device Identification Registers.....	72
5.1.4. Multifunction and SR-IOV System Settings Parameters [Endpoint Mode].....	73
5.1.5. Configuration, Debug and Extension Options.....	74
5.1.6. PHY Characteristics.....	75
5.1.7. PCI Express / PCI Capabilities Parameters.....	75
5.2. Example Designs.....	78
6. Parameters (P-Tile) (F-Tile) (R-Tile).....	79
6.1. Top-Level Settings.....	79
6.2. PCIe0 Settings.....	82
6.2.1. Base Address Register.....	82
6.2.2. PCIe0 Configuration, Debug and Extension Options.....	84
6.2.3. PCIe0 Device Identification Registers.....	87
6.2.4. PCIe0 PCI Express / PCI Capabilities.....	89
6.2.5. MCDMA Settings.....	96
6.3. Example Designs.....	99
6.4. Analog Parameters (F-Tile MCDMA IP Only).....	101
6.5. PCIe1 Settings.....	102
6.5.1. PCIe1 Configuration, Debug and Extension Options.....	102
7. Designing with the IP Core.....	104
7.1. Generating the IP Core.....	104
7.2. Simulating the IP Core.....	105
7.3. IP Core Generation Output - Quartus Prime Pro Edition.....	106

7.4. Systems Integration and Implementation.....	109
7.4.1. Required Supporting IP.....	109
8. Software Programming Model.....	110
8.1. Multi Channel DMA IP Custom Driver.....	111
8.1.1. Architecture.....	111
8.1.2. libmqdma library details	114
8.1.3. Application	115
8.1.4. Software Flow.....	117
8.1.5. API Flow	118
8.1.6. libmqdma Library API List.....	120
8.1.7. Request Structures.....	127
8.2. Multi Channel DMA IP DPDK Poll-Mode based Driver.....	128
8.2.1. Architecture.....	128
8.2.2. MCDMA Poll Mode Driver.....	130
8.2.3. DPDK based application.....	132
8.2.4. Request Structures.....	132
8.2.5. Software Flow.....	133
8.2.6. API Flow.....	134
8.2.7. API List.....	136
8.3. Multi Channel DMA IP Kernel Mode Network Device Driver.....	138
8.3.1. Architecture.....	139
8.3.2. Driver Information.....	140
8.3.3. Software Flow.....	142
9. Registers.....	145
9.1. Queue Control (QCSR).....	146
9.2. MSI-X Memory Space.....	152
9.3. Control Register (GCSR).....	153
10. Troubleshooting/Debugging.....	155
10.1. Debug Toolkit.....	155
10.1.1. Overview.....	155
10.1.2. Enabling the Debug Toolkit.....	157
10.1.3. Launching the Debug Toolkit.....	157
10.1.4. Using the Debug Toolkit.....	160
11. Multi Channel DMA FPGA IP for PCI Express User Guide Archives.....	181
12. Revision History for the Multi Channel DMA FPGA IP for PCI Express User Guide.....	182

1. Before You Begin

1.1. Terms and Acronyms

Table 1. Acronyms

Term	Definition
API	Application Programming Interface
ATT	Address Translation Table
Avalon®-ST (or AVST)	Avalon Streaming Interface
Avalon-MM (or AVMM)	Avalon Memory-Mapped Interface
BAS	Bursting Avalon-MM Slave
BAM	Bursting Avalon-MM Master
CvP	Configuration via Protocol
D2H	Device-to-Host
D2HDM	Device-to-Host Data Mover
DMA	Direct Memory Access
DPDK	Data Path Development Kit
EOF	End of a File (or packet) for streaming
EP	End Point
FAE	Field Applications Engineer
FLR	Functional Level Reset
File (or Packet)	A group of descriptors defined by SOF and EOF bits of the descriptor for the streaming. At Avalon-ST user interface, a file (or packet) is marked by means of sof/eof.
GCSR	General Control and Status Register
Gen1	PCIe 1.0
Gen2	PCIe 2.0
Gen3	PCIe 3.0
Gen4	PCIe 4.0
Gen5	PCIe 5.0
H2DDM	Host-to-Device Data Mover
H2D	Host-to-Device
HIP	Hard IP
HIDX	Queue Head Index (pointer)
<i>continued...</i>	

© Altera Corporation. Altera, the Altera logo, the 'a' logo, and other Altera marks are trademarks of Altera Corporation. Altera reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

Term	Definition
IMMWR	Immediate Write Operation
IP	Intellectual Property
MCDMA	Multi Channel Direct Memory Access
MRRS	Maximum Read Request Size
MSI-X	Message Signaled Interrupt - Extended
MSI	Message Signaled Interrupt
PBA	Pending Bit Array
PD	Packet Descriptor
PCIe*	Peripheral Component Interconnect Express (PCI Express*)
PIO	Programmed Input/Output
PMD	Poll Mode Driver
QCSR	Queue Control and Status register
QID	Queue Identification
RP	Root Port
SOF	Start of a File (or packet) for streaming
SR-IOV	Single Root I/O Virtualization
TLP	Transaction Layer Packet
TIDX	Queue Tail Index (pointer)

1.2. Known Issues

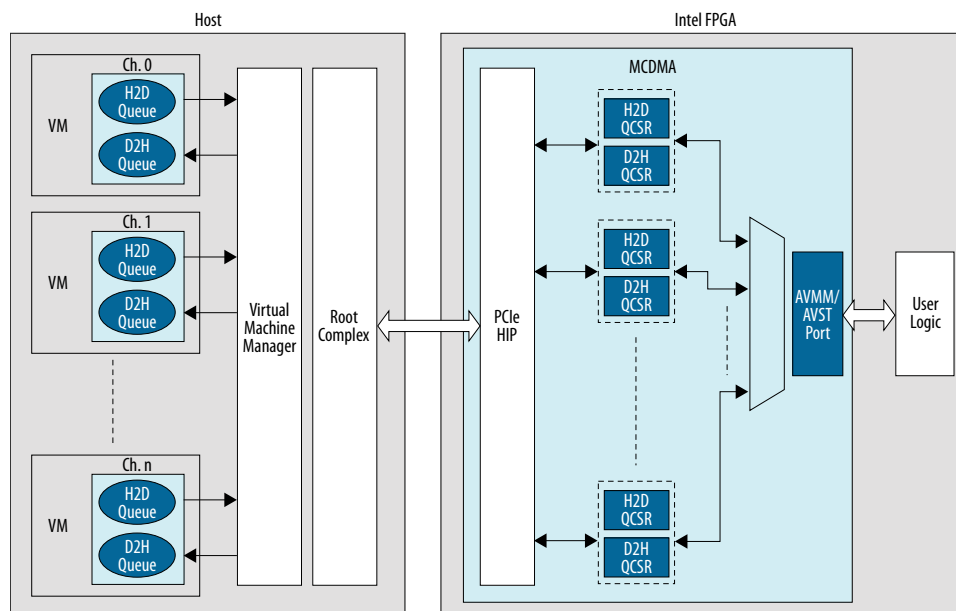
The following summarizes known issues in the current IP release:

1. Multichannel D2H AVST configuration has stability issues when total number of D2H channels configured is greater than 256
2. Design Example simulation in Quartus® Prime 23.4 release is failing for BAM+BAS +MCDMA user mode in H-Tile, when SRIOV is enabled.
3. MCDMA R-Tile Design Example simulations are not supported in Quartus Prime 23.4 release, except for PIO using MCDMA Bypass Mode Design Example.
4. Avalon-ST Source (H2D) interface data can get corrupted, if a Completion timeout event occurs for H2DDM descriptor fetch. This issue happens only if the SOP descriptor is fetched successfully and the EOP descriptor for a channel got missed because of the Completion timeout event.
5. FLR request is not completed gracefully, when assertion of Soft reset happens during FLR request is still under process.
6. MCDMA Root Port may drop multi-function Mem Rd Packets targeted to remote device as multi- function enabled End Point.
7. Few data drops may occur on a channel when a Q_RESET occurs on different channel, which is waiting for a Descriptor Fetch CPL after issuing a hardware instruction.

8. MCDMA P/F-Tile Design Example [Device-side Packet loopback, Packet Generate/Check and Traffic Generator/Checker] simulations are not supported for Gen3 Hard IP configurations.
9. MCDMA H/P/F/R-Tile does not support single physical function with one DMA Channel allocated and no SRIOV enabled configuration.
10. PIO read/write values do not match for certain addresses for PIO bypass mode DE with BAM+BAS or BAM+MCDMA or BAM+BAS+MCDMA modes.
11. MCDMA P-Tile 1x8 Hard IP mode with Debug Toolkit enabled does not load in system-console for Stratix[®] 10 DX Development Kit.
12. For the AXI Multi-PF configuration, stuck is being observed for both H2D and D2H paths in the MSIX mode with custom driver+VFIO driver only for a privileged PF. Meanwhile, the WB mode can be used instead of MSIX mode for privileged PFs in this configuration.

2. Introduction

Figure 1. Multi Channel DMA IP for PCI Express Usage in Server Hardware Infrastructure



The Multi Channel DMA IP for PCI Express enables you to efficiently transfer data between the host and device. The Multi Channel DMA IP for PCI Express supports multiple DMA channels between the host and device over the underlying PCIe link. A DMA channel consists of H2D (host to device) and D2H (device to host) queue pair.

As shown in the figure above, the Multi Channel DMA IP for PCI Express can be used in a server's hardware infrastructure to allow communication between various VM-clients and their FPGA-device based counterparts. The Multi Channel DMA IP for PCI Express operates on descriptor-based queues set up by driver software to transfer data between local FPGA and host. The Multi Channel DMA IP for PCI Express's control logic reads the queue descriptors and executes them.

The Multi Channel DMA IP for PCI Express integrates the Intel® PCIe Hard IP and interfaces with the host Root Complex via the PCIe serial lanes. On the user logic interface, Avalon-MM/Avalon-ST interfaces allow the designer easy integration of the Multi Channel DMA IP for PCI Express with other Platform Designer components.

Besides DMA functionality, Multi Channel DMA IP for PCI Express enables standalone Endpoint or Rootport functionality with Avalon-MM interfaces to the user logic. This functionality is described in more detail in the Functional Description chapter.

2.1. Multi Channel DMA IP for PCI Express Features

2.1.1. Endpoint Mode

- MCDMA P-Tile: PCIe Gen4/Gen3 x16/x8 and 2x8 ports in Stratix 10 DX and Agilex™ 7 devices.
- MCDMA H-Tile: PCIe Gen3 x16/x8 in Stratix 10 GX and Stratix 10 MX devices.
- MCDMA F-Tile: PCIe Gen4/Gen3 x16/x8/x4 and 2x8 ports in Agilex 7 device
- MCDMA R-Tile:
 - PCIe Gen5/Gen4/Gen3 2x8 in Agilex 7 devices.
 - PCIe Gen4/Gen3 x16 only in Agilex 7 I-Series FPGA Development Kit DK-DEV-AGI027R1BES R-Tile B0 revision.
 - PCIe Gen5/Gen4/Gen3 4x4 only in Agilex 7 I-Series FPGA Development Kit DK-DEV-AGI027R1BES R-Tile B0 revision. Port 2 and 3 don't support SRIOV, FLR, user event MSI-X and MSI capability. Port 2 and 3 only support BAM, BAS and BAM+BAS user mode.
- User Mode options:
 - Multi Channel DMA
 - Bursting Avalon Master (BAM)
 - Bursting Avalon Slave (BAS)
 - BAM and BAS
 - BAM and MCDMA
 - Data Mover Only (not available in MCDMA R/P/F-Tile IP x4 and MCDMA H-Tile IP)
 - BAM, BAS and MCDMA
- Supports up to 2,048 DMA channels.

Table 2. Maximum DMA channels

Device	MCDMA Interface Type	
	Avalon-MM	Avalon-ST
Stratix 10 GX Stratix 10 MX Stratix 10 DX Agilex 7	2,048†	2,048†

Note: † = Maximum 512 channels per Function

- Per Descriptor completion notification with MSI-X or Writebacks
- Option to select Avalon-MM or Avalon-ST DMA for user logic interface
- SR-IOV
- User MSI-X in MCDMA mode (Not supported in R-Tile MCDMA IP x4 Endpoint Ports 2 and 3)
- User FLR in MCDMA mode (Not supported in R-Tile MCDMA IP x4 Endpoint Ports 2 and 3)
- 10-bit tag feature

- Supports Precision Time Measurement (PTM). (Only available for R-Tile MCDMA IP Endpoint Ports 0 and 1)
- MSI Interrupt in BAS (only available for for all H/R/F/P Tiles MCDMA IP). It is not supported in R-Tile MCDMA IP x4 Endpoint Ports 2 and 3)
- H2D address and payload size alignment to byte granularity for AVST
- Ports subdivided 2x8 can run independently and concurrently (separately) instances on MCDMA and AVMM IP for P-Tile MCDMA Stratix 10 DX and R-Tile MCDMA Agilex 7 devices.
 - Example: Port0 -> MCDMA AVMMDMA DE & Port1 -> BAM+MCDMA Pkt Gen Checker DE.
 - Each instance runs independently with separate PERST
 - SCTH support
- Ports subdivided 4x4 can run independently and concurrently (separately) instances on MCDMA and AVMM IP for R-Tile MCDMA Intel Agilex 7 devices.
 - Examples: Port0 -> MCDMA, Port1 -> BAM+MCDMA, Port2 -> BAS & Port3 -> BAM+BAS.
 - Each instance runs independently with separate PERST
 - Only SCT support
- Example Design Simulation is only supported on Port0. Simulation is not support on Port0, Port1 and Port2
- Maximum payload size supported:
 - Stratix 10 GX and Stratix 10 MX devices: 512 bytes
 - Stratix 10 DX and Agilex 7 devices: 512 / 256 / 128 bytes

2.1.2. Root Port Mode

- MCDMA H-Tile: PCIe Gen3 x16/x8 in Stratix 10 GX and Stratix 10 MX devices
- MCDMA P-Tile: PCIe Gen4/Gen3 x16 and 4x4 ports in Stratix 10 DX and Agilex 7 devices
- MCDMA F-Tile: PCIe Gen4/Gen3 x16/x8, 2x4 and 4x4 in Agilex 7 device
- MCDMA R-Tile:
 - PCIe Gen5/Gen4/Gen3 2x8 ports in Agilex 7 devices.
 - PCIe Gen4/Gen3 x16 and Gen5/Gen4/Gen4 4x4 ports only in Agilex 7 I-Series FPGA Development Kit DK-DEV-AGI027R1BES R-Tile B0 revision.
- Configuration Slave (CS) interface for accessing Endpoint's config space
- Address Translation Table (ATT) is supported in BAS mode. MCDMA H-Tile IP does not support ATT.
- User mode options:
 - Bursting Avalon Master (BAM)
 - Bursting Avalon Slave (BAS)
 - BAM and BAS

- Ports subdivided 4x4 can run independently and concurrently (separately) instances on AVMM IP for R-Tile MCDMA Agilex 7 device
 - Examples: Port0 -> BAM, Port1 -> BAM+BAS, Port2 -> BAS & Port3 -> BAM+BAS.
 - Each instance runs independently with separate PERST
 - Only SCT is supported
- Ports subdivided 2x4 or 4x4 can run independently and concurrently (separately) instances on AVMM IP for F-Tile MCDMA Agilex 7 device
 - Each instance runs independently with separate PERST
 - Only SCT is supported
- Maximum payload size supported:
 - Stratix 10 GX and Stratix 10 MX devices: 512 bytes
 - Stratix 10 DX and Agilex 7 devices: 512 / 256 / 128 bytes

2.2. Device Family Support

The following terms define Multi Channel DMA IP for PCI Express core support levels for Intel FPGA IP cores in Stratix 10 devices:

- Advanced support:** 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 (data-path width, burst depth, I/O standards tradeoffs).
- Preliminary support:** 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 support:** 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. Device Family Support Table

Device Family	Support Level
Stratix 10	Final
Agilex 7	Preliminary
Agilex 9	Preliminary
Other device families	No support

Related Information

Timing and Power Models

Reports the default device support levels in the current version of the Intel Quartus Prime Pro Edition software.

2.3. Recommended Speed Grades

Table 4. Recommended Speed Grades

PCIe Gen.	Device	Tile Variant	PLD Clock Frequency	Recommended Fabric Speed Grade
Gen 3	Stratix 10 GX/MX	H-Tile	250 MHz	-1
	Stratix 10 DX	P-Tile	250 MHz	-1, -2, -3
	Agilex 7	P-Tile, F-Tile, R-Tile	250 MHz	-1, -2, -3
Gen4	Stratix 10 DX	P-Tile	350 MHz	-1
	Stratix 10 DX	P-Tile	250 MHz	-1, -2
	Agilex 7	P-Tile, F-Tile, R-Tile	400 MHz	-1, -2, -3
	Agilex 7	P-Tile, F-Tile, R-Tile	500 MHz	-1
Gen5	Agilex 7	R-Tile	400 MHz	-1, -2, -3
	Agilex 7	R-Tile	500 MHz	-1

Related Information

Intel Quartus Standard for Timing Closure and Optimization
Use this link for the Quartus Prime Pro Edition Software.

2.4. Resource Utilization

Table 5. Intel Stratix 10 H-Tile and P-Tile PCIe x16 [Avalon-MM Interface]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			H-Tile	P-Tile	H-Tile	P-Tile	H-Tile	P-Tile
MCDMA	PCIe Gen3 x16 for H-Tile PCIe Gen4 x16 for P-Tile	256	44,034	37,502	109,399	99,491	532	512
BAM_MCDMA	PCIe Gen4 x16 for P-Tile PCIe Gen3 x16 for H-Tile	256	48,447	41,835	120,555	110,600	616	596
BAM_BAS_MCDMA	PCIe Gen3 x16 for H-Tile	2,048	52,130	-	143,217	-	858	-
BAM	PCIe Gen4 x16 for P-Tile PCIe Gen3 x16 for H-Tile	n/a	25,162	17,567	53,976	42,111	307	285
BAS	PCIe Gen4 x16 for P-Tile	n/a	26,818	20,126	61,369	49,486	257	236
continued...								

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			H-Tile	P-Tile	H-Tile	P-Tile	H-Tile	P-Tile
	PCIe Gen3 x16 for H-Tile							
BAM+BAS	PCIe Gen4 x16 for P-Tile PCIe Gen3 x16 for H-Tile	n/a	33,655	25,104	78,809	65,025	372	346

Table 6. Intel Stratix 10 H-Tile and P-Tile PCIe x8 [Avalon-MM Interface]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			H-Tile	P-Tile	H-Tile	P-Tile	H-Tile	P-Tile
MCDMA	PCIe Gen3 x8 for H-Tile PCIe Gen4 x8 for P-Tile	256	22,914	25,822	61,888	69,774	397	372
BAM_MCDMA	PCIe Gen3 x8 for H-Tile PCIe Gen4 x8 for P-Tile	256	25,329	28,320	68,691	76,285	452	431
BAM_BAS_MCDMA	PCIe Gen3 x8 for H-Tile	2,048	33,957	-	97,495	-	544	-
BAM	PCIe Gen3 x8 for H-Tile PCIe Gen4 x8 for P-Tile	n/a	8,257	9,938	21,171	27,441	199	177
BAS	PCIe Gen3 x8 for H-Tile PCIe Gen4 x8 for P-Tile	n/a	9,227	11,374	24,973	31,260	169	149
BAM+BAS	PCIe Gen3 x8 for H-Tile PCIe Gen4 x8 for P-Tile	n/a	12,530	14,563	34,508	40,592	248	226

Table 7. Intel Stratix 10 H-Tile and P-Tile PCIe x16 [1 port Avalon-ST]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			H-Tile	P-Tile	H-Tile	P-Tile	H-Tile	P-Tile
MCDMA	PCIe Gen3 x16 for H-Tile PCIe Gen4 x16 for P-Tile	1 / 32 / 64	47,866 / 50,093 / 52,951	38,634 / 41,181 / 43,852	117,470 / 122,854 / 128,771	104,793 / 110,305 / 115,833	560 / 578 / 601	536 / 555 / 576
BAM_MCDMA	PCIe Gen3 x16 for H-Tile PCIe Gen4 x16 for P-Tile	2 / 32 / 64	51,976 / 54,300 / 57,132	42,155 / 43,745 / 45,118	128,208 / 133,935 / 139,874	113,660 / 117,292 / 120,406	643 / 662 / 684	615 / 625 / 638

Table 8. Agilex 7 P-Tile and F-Tile PCIe x16 [Avalon-MM Interface]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			P-Tile	F-Tile	P-Tile	F-Tile	P-Tile	F-Tile
MCDMA	Gen4 x16	256	33,805	37,445	97,557	103,143	512	521
BAM_MCDMA	Gen4 x16	256	38,546	42,198	108,328	113,886	595	605
BAM_BAS_MCDMA	Gen4 x16	2,048	43,907	44,000	139,591	139,552	855	855
BAM	Gen4 x16	n/a	17,246	20,780	42,097	47,680	285	295
BAS	Gen4 x16	n/a	19,164	22,677	49,327	54,854	236	246
BAM+BAS	Gen4 x16	n/a	24,955	28,562	64,885	70,342	346	356

Table 9. Agilex 7 P-Tile and F-Tile PCIe x8 [Avalon-MM Interface]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			P-Tile	F-Tile	P-Tile	F-Tile	P-Tile	F-Tile
MCDMA	Gen4 x8	256	22,254	23,864	67,551	69,063	372	383
BAM_MCDMA	Gen4 x8	256	24,440	26,085	74,195	75,716	431	441
BAM_BAS_MCDMA	Gen4 x8	2048	27,520	27,607	92,671	92,728	539	539
BAM	Gen4 x8	n/a	9,052	10,689	27,189	28,675	177	187
BAS	Gen4 x8	n/a	10,331	11,907	31,029	32,514	149	159
BAM+BAS	Gen4 x8	n/a	13,319	14,933	40,518	41,988	226	236

Table 10. Agilex 7 R-Tile PCIe x8 [Avalon-MM Interface]

User Mode	Link Conf	DMA Channels	ALMs	Logic Registers	M20Ks
MCDMA	Gen5 x8	2,048	36,360	115,373	857
BAM	Gen5 x8	-	15,023	45,060	327
BAM+BAS	Gen5 x8	-	14,933	45,036	327
BAM_MCDMA	Gen5 x8	2,048	35,937	113,178	843

Table 11. Agilex 7 P-Tile and F-Tile PCIe x16 [1 port Avalon-ST]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			P-Tile	F-Tile	P-Tile	F-Tile	P-Tile	F-Tile
MCDMA	Gen4 x16	1 / 32 / 64	33,913 / 36,373 / 39,480	37,567 / 40,071 / 43,078	102,712 / 108,215 / 114,039	108,303 / 113,764 / 119,553	537 / 554 / 576	546 / 564 / 587
BAM_MCDMA	Gen4 x16	2 / 32 / 64	38,247 / 39,448 / 41,041	41,880 / 43,115 / 44,686	112,445 / 115,445 / 118,806	118,007 / 120,995 / 124,434	620 / 625 / 639	629 / 636 / 648

Table 12. Agilex 7 P-Tile and F-Tile PCIe x8 [1 port Avalon-ST]

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			P-Tile	F-Tile	P-Tile	F-Tile	P-Tile	F-Tile
MCDMA	Gen4 x8	1 / 32 / 64	22,978 / 25,343 / 28,399	24,705 / 27,066 / 30,219	72,007 / 77,499 / 83,182	73,565 / 79,005 / 84,731	397 / 413 / 436	407 / 424 / 446
BAM_MCDMA	Gen4 x8	2 / 32 / 64	24,790 / 26,083 / 27,550	26,541 / 27,776 / 29,334	77,532 / 80,585 / 84,057	79,104 / 82,126 / 85,545	455 / 461 / 473	465 / 470 / 483

Table 13. Agilex 7 P-Tile and F-Tile Data Mover Only User Mode

User Mode	Link Conf	DMA Channels	ALMs		Logic Registers		M20Ks	
			P-Tile	F-Tile	P-Tile	F-Tile	P-Tile	F-Tile
Data Mover Only	Gen4 x16	n/a	31,528	41,514	83,773	91,219	522	532
Data Mover Only	Gen4 x8	n/a	18,163	25,718	56,890	60,391	381	391

Table 14. Intel Stratix 10 P-Tile Data Mover Only User Mode

User Mode	Link Configuration	DMA Channels	ALMs	Logic Registers	M20Ks
Data Mover Only	Gen4 x16	n/a	33,214	84,200	522
Data Mover Only	Gen4 x8	n/a	21,149	57,077	381

2.5. 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. If an IP core version is not listed, the user guide for the previous IP core version applies. The IP versioning scheme (X.Y.Z) number changes from one software version to another.

A change in:

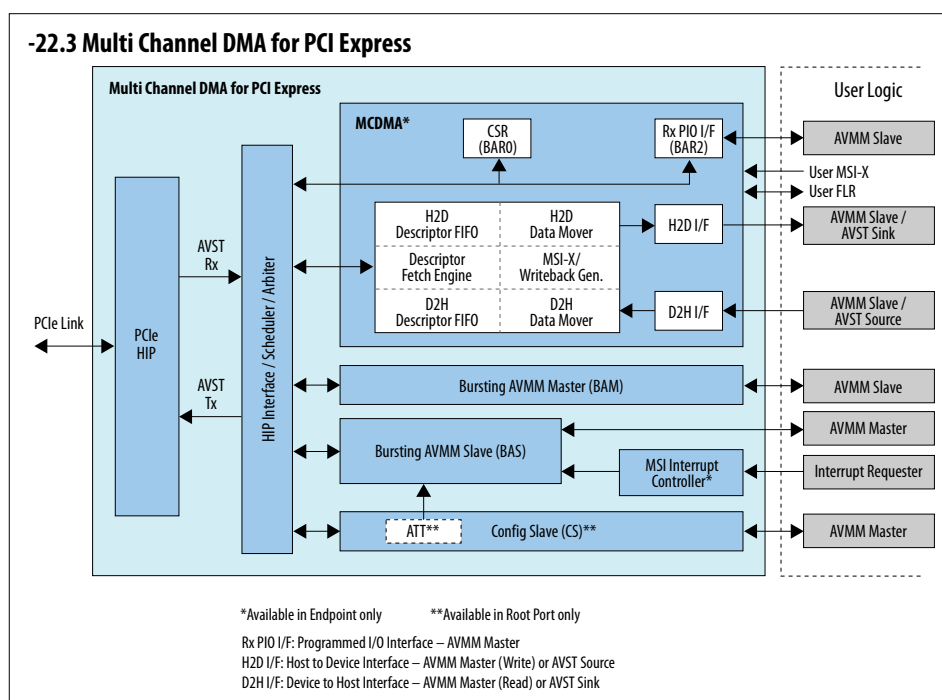
- X indicates a major revision of the IP. If you update your Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 15. Release Information for the Multi Channel DMA IP for PCI Express Core

Item	Description
IP Version	H-Tile IP version: 24.2.2 P-Tile IP version: 8.3.2 F-Tile IP version: 9.3.2 R-Tile IP version: 5.3.2
Quartus Prime Version	Quartus Prime Pro Edition 25.1.1 Software Release
Ordering Part Number (OPN)	H-Tile: IP-PCIEMCDMA P-Tile: IP-PCIEMCDMA F-Tile: IP-PCIEMCDMA R-Tile: IP-PCIEMCDMA

3. Functional Description

Figure 2. Multi Channel DMA IP for PCI Express Block Diagram



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 Multi Channel DMA IP for PCI Express supports. Each row indicates a user mode with required block(s).

Table 16. Valid user modes and required functional blocks

Mode		MCDMA	Bursting Master (BAM)	Bursting Slave (BAS)	Config Slave (CS)	Data Mover
Endpoint	MCDMA	✓	×	×	×	×
	BAM	×	✓	×	×	×
	BAS	×	×	✓	×	×
	BAM+BAS	×	✓	✓	×	×
	BAM+MCDMA	✓	✓	×	×	×
	BAM+BAS+MCDMA	✓	✓	✓	×	×
continued...						

Mode		MCDMA	Bursting Master (BAM)	Bursting Slave (BAS)	Config Slave (CS)	Data Mover
	Data Mover Only	x	√	x	x	√
Root Port	BAM	x	√	x	√	x
	BAS	x	x	√	√	x
	BAM+BAS	x	√	√	√	x

Note: Data mover only mode is not available for x4 topology in P/F/R-Tile MCDMA IPs.

3.1. Multi Channel DMA

Multi Channel DMA IP for PCI Express consists primarily of H2DDM & D2HDM blocks. It also offers a DMA-bypass capability to the Host for doing PIO Read/Writes to device memory.

The MCDMA engine operates on software DMA queue to transfer data between local FPGA and host. The elements of each queue are software descriptors that are written by driver/software. Hardware reads the queue descriptors and executes them. Hardware can support up to 2K DMA channels. For each channel, separate queues are used for read/write DMA operations.

Note: MCDMA requires the Source and Destination addresses be 64 byte aligned in D2H direction. This may not be required in future release.

3.1.1. H2D Data Mover

The Host-to-Device Data Mover (H2DDM) module transfers data from the host memory to local memory through the PCIe Hard IP and the Avalon-MM Write Master/ Avalon-ST Source interface.

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 Mem Rd TLPs based on descriptor information such as 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 re-ordered to ensure the read data is delivered to user logic in order.

When a descriptor is completed, that is, all read data has been received and forwarded to the Avalon-MM Write Master / Avalon-ST Source 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 write to host via the D2HDM to avoid 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.

3.1.2. D2H Data Mover

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

In AVMM mode, the D2HDM sends a series of AVMM reads via the master port based on PCIe address, MPS, and DMA transfer size. The AVMM read is generated as follows:

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

In AVST mode, D2HDM AVST sink de-asserts ready when descriptors are not available.

- Host sets up software descriptors for a port. Max payload count can be up to 1MB. SOF/EOF fields in the descriptor may not be set by the Host.
 - D2HDM uses descriptor update sequence to update SOF, EOF, Rx payload count fields in the software descriptor at Host location through a Memory Write request
- AVST `d2h_st_sof_i` signal assertion triggers a descriptor update sequence by D2HDM to mark start of AVST frame.
 - D2HDM issues a MWr to set the SOF field in the descriptor
 - WB/MSI-X, if set in the descriptor, is issued
- AVST `d2h_st_eof_i` signal assertion triggers a descriptor update sequence by D2HDM to mark end of AVST frame. The descriptor update sequence is as follows:
 - D2HDM terminates the descriptor at `d2h_st_eof_i` and initiates a descriptor update sequence.
 - During descriptor update sequence, a MWr is issued to set EOF field in the descriptor and update Rx payload count field with total bytes transferred.
 - WB/MSI-X if set in descriptor, is issued
- The descriptor immediately after EOF sequence, is considered as start of next AVST data frame and initiates a descriptor update sequence to set 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 write to host via the D2HDM to avoid 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.

3.1.2.1. D2H Descriptor Fetch

When you enable multiple channels over single port in AVST mode, the 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 (AVST port), there is no handshake between Host SW and User Logic through the 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 MCDMA IP implements segmentation of descriptor FIFO.

Below are the two IP parameters defined to provide user programmability.

- D2H Prefetch Channels: D2H descriptor memory is arranged in multiple segments to support user selectable number of prefetch queues (also known as Active Queues).
- Maximum Descriptor Fetch: Maximum number of D2H descriptors that can be fetched for each prefetch channel.

Since the MCDMA IP only implements N-prefetch channels, it is capable of handling user AVST data received for a non-prefetched channel. When D2H data mover receives data on AVST for a channel that does not have descriptors prefetched, D2HDM requests for descriptors to be fetched for that channel.

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

Note:

D2H does not support Tail pointer updates on a disabled channel in the current IP version. The host software must make sure a channel is enabled before doing Tail pointer updates.

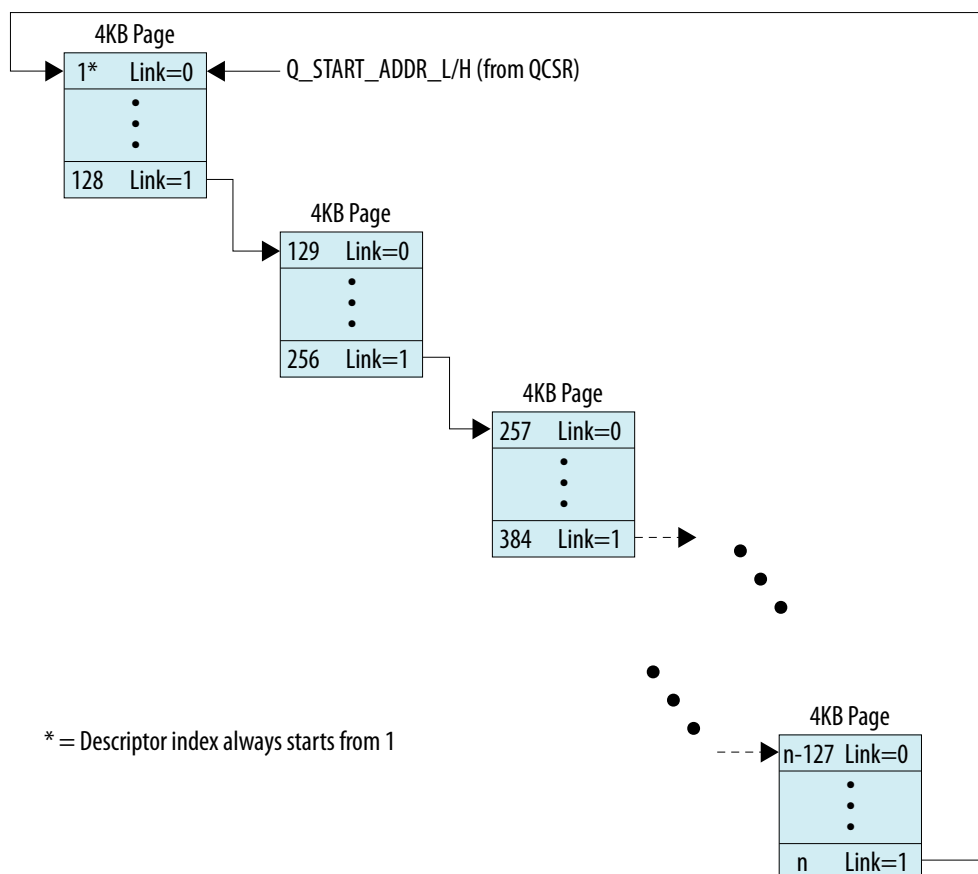
When all the segments are occupied and D2HDM receives data for a channel that does not have descriptors prefetched, the least recently used segment is cleared to accommodate descriptors fetched for this new channel. Descriptors in the least recently used segment that were cleared, are refetched whenever D2HDM receives data for the corresponding channel.

3.1.3. Descriptors

A DMA channel to support Multi Channel DMA data movement consists of a pair of the 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 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 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 3. Descriptor Linked-List



Software and hardware communicate and manage the descriptors using 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 4. Descriptor Ring Buffer

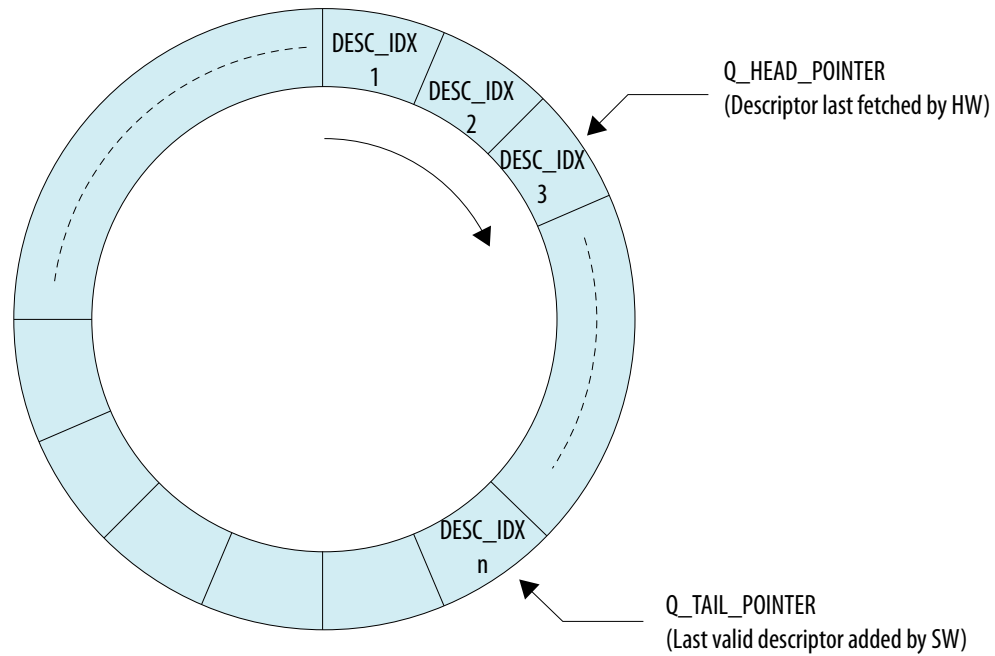


Table 17. Software Descriptor Format

Name	Width	Description
SRC_ADDR [63:0]	64	If Link bit =0, then this field contains the source address. Starting system address of 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 AVMM address in device memory. If the link bit is set, then this contains the address of the next 4 KB page in host memory containing the descriptors.
DEST_ADDR [127:64]	64	Provided link=0, this field means: Starting system address of 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 AVMM address in device memory.
PYLD_CNT [147:128]	20	Provided link=0, this field means DMA payload size in bytes. Max 1 MB, with 20'h0 indicating 1 MB. For a D2H queue in 1 port AVST mode, this field is a fixed value across all descriptors of the queue and must reflect the value in Q_PYLD_CNT (0x44) QCSR register.
RSRVD [159:148]	12	Reserved
continued...		

Name	Width	Description
DESC_IDX [175:160]	16	Unique Identifier for each descriptor, assigned by the software driver. This value is written to Q_COMPLETED_POINTER register when a descriptor data transfer is complete. <i>Note:</i> 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
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	SOF indicator for Avalon-ST streaming. In the H2D streaming, this bit causes the Avalon-ST Source interface to assert h2d_st_sof_o, indicating start of a file/packet. In the D2H streaming, this bit is set in the Descriptor itself, by MWr TLP when the user logic asserts d2h_st_sof_i, indicating start of a file/packet. <i>Note:</i> 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. <i>Note:</i> In the D2H streaming, if user logic prematurely ends the data transfer by asserting d2h_st_eof_i in the middle of a descriptor data move then starts a next file/packet, the SOF bit in the next descriptor is set by MWr TLP. <i>Note:</i> SOF bit is an optional feature for DMAs involving file data transfers using Avalon-ST interface.
EOF [223]	1	EOF indicator for Avalon-ST streaming. In the H2D streaming, this bit causes the Avalon-ST Source interface to assert h2d_st_eof_o, indicating 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_st_eof_i, indicating end of a packet. Along with the EOF bit, MWr TLP also updates the actual received payload count (RX_PYLD_CNT) field of the last descriptor.

continued...

Name	Width	Description
		<i>Note:</i> EOF bit is an optional feature for DMAs involving file data transfers using Avalon-ST interface.
RSRVD [253:224]	30	Reserved
DESC_INVALID [254]	1	Indicates if 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 host memory containing the descriptors.

3.1.3.1. Support for Unaligned (or byte-aligned) Data Transfer

MCDMA IP supports the following alignment modes for the descriptor source/destination address and payload count fields.

Table 18. Alignment Mode

Alignment Mode	IP GUI Parameter Setting	SRC/DEST Address Alignment	Payload Count Alignment	Note
Default alignment	Enable address byte aligned = FALSE	AVMM: DWORD aligned AVST: 64-byte aligned (or full data width aligned)	AVMM: DWORD aligned AVST: 64-byte aligned (exception: last descriptor of a packet/file)	<ul style="list-style-type: none"> Descriptors are 32-byte aligned Low resource utilization
Unaligned (or Byte aligned) Access	Enable address byte aligned = TRUE	Byte aligned	Byte aligned	<ul style="list-style-type: none"> Supported for H2D AVST Interface only Descriptors are 32-byte aligned DMA read requests use the PCIe First DWORD Byte Enable and Last DWORD Byte Enable to support byte granularity High resource utilization

3.1.3.2. Metadata Support

8 Byte Metadata

In Avalon Streaming mode, once you select 8 Byte metadata support during IP generation, the source and destination address field in the existing descriptor structure are repurposed for metadata support. The following fields of the existing descriptor defined above have revised properties.

Table 19.

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 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 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 Host Memory. If the queue is H2D, then this is 8 Byte Metadata

3.1.3.3. MSI-X/Writeback

MSI-X and Writeback block update the host with the current processed queue's head pointer and interrupt. Apart from a global MSI-X Enable and Writeback Enable, 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 relation between global and per-descriptor MSI-X/Writeback Enable.

Table 20. Multi Channel 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 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 Avalon-ST sink interface has received an sof/eof 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).

3.1.4. Avalon-MM PIO Master

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

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

The Avalon-MM PIO Master is present only if you select **Multi Channel DMA** User Mode for **MCDMA Settings** in the IP Parameter Editor GUI. The Avalon-MM PIO Master is always present irrespective of the Interface type (Avalon-ST/Avalon-MM) that you select.

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 SRIOV 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 ($\log_2(\text{Number of PFs})$) is the RTL design parameter selected by you such that Multi Channel DMA IP only allocates required number of the bits on Avalon-MM side to limit the number of the 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 ($\log_2(\text{Number of VFs})$) is the RTL design parameter selected by you such that Multi Channel DMA IP only allocates required number of the bits on Avalon-MM side to limit the number of the wires on the user interface.
4. **address**: Number of bits required for BAR2 size requested across all Functions (PFs and VFs) Example: If BAR2 is selected as 4 MB, the address size is 22 bits.

3.1.5. Avalon-MM Write (H2D) and Read (D2H) Master

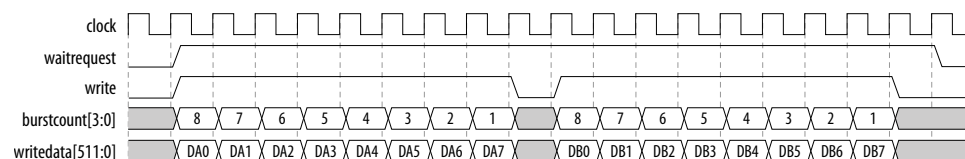
Avalon-MM Interface is used to transfer data between the host and device through the memory-mapped interface. You can enable the Memory-Mapped interface by selecting AVMM Interface type in the IP Parameter Editor. The Multi Channel DMA IP for PCI Express supports 1 write master port and 1 read master port.

Avalon-MM Write Master

The Avalon-MM Write Master is used to write H2D DMA data to the Avalon-MM slave in the user logic through the memory-mapped interface. The Write Master can issue AVMM write commands for up to 8/16/32 burst count for 512/256/128 data-width respectively. The `waitrequestAllowance` of this port is enabled, allowing the master to transfer up to N additional write command cycles after the `waitrequest` signal has been asserted. Value of <N> for H2D AVMM Master is as follows:

- 512-bit data-width is 16
- 256-bit data-width is 32
- 128-bit data-width is 64

Figure 5. Avalon-MM Write with waitrequestAllowance 16

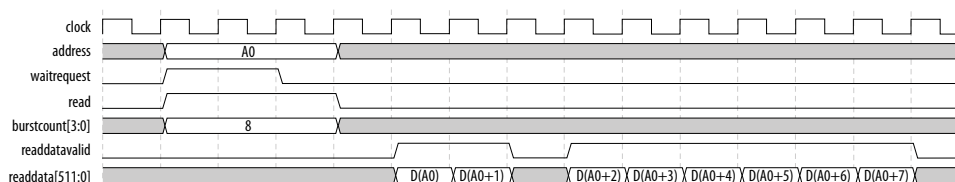


Avalon-MM Read Master

The Avalon-MM Read Master is used to read D2H DMA data from the Avalon-MM slave in the user logic through the memory-mapped interface. The Read Master can issue AVMM read commands for up to 8 bursts (burst count = 8). The waitrequestAllowance of this port is enabled, allowing the master to transfer up to N additional write command cycles after the waitrequest signal has been asserted. Value of <N> for H2D AVMM Master is as follows:

- 512-bit data-width is 16
- 256-bit data-width is 32
- 128-bit data-width is 64

Figure 6. Avalon-MM Read Master Timing Diagram



3.1.6. Avalon-ST Source (H2D) and Sink (D2H)

Multi Channel DMA provides Avalon Streaming Interfaces for transferring DMA data between the host and device. Avalon-ST Source interface is used to move H2D DMA data to external user logic. Avalon-ST Sink interface is used to move D2H DMA data to the host.

3.1.6.1. Avalon-ST 1-Port Mode

When you select AVST 1 port mode, the IP provides 1 AVST Source and Sink port for DMA. In this mode, you can enable up to 2K DMA channels.

Table 21. IP Parameters specific to D2H Descriptor Fetch in Avalon-ST 1 Port Mode

IP GUI Parameter	Description	Value for MCDMA
D2H Prefetch Channels	Number of prefetch channels	8, 16, 32, 64, 128, 256
Maximum Descriptor Fetch	Number of descriptors that can be fetched for each prefetch channels	16, 32, 64

Note: In the current Quartus Prime release, the D2H Prefetch Channels follows the total number of DMA channels that you select up to 256 total channels. When the total number of channels selected is greater than 256, then D2H Prefetch channels are fixed to 64. The resource utilization shall increase with the number of D2H prefetch channels.

For details about these parameters, refer to the *D2H Data Mover* section.

3.1.6.2. Packet (File) Boundary

When streaming the DMA data, the packet (file) boundary is indicated by the SOF and EOF bits of the descriptor and corresponding sof and eof signals of the Avalon-ST interface. Channel interleaving is not supported. A channel switch on AVST interface can only happen on packet boundary

Table 22. Multi Channel DMA Streaming Packet Boundary

<n>: 0 for 1 port

Packet Boundary	Descriptor Field	AVST Source (H2D) Signal	AVST Sink (D2H) Signal
Start of Packet	SOF	h2d_st_sof_<n>_o	d2h_st_sof_<n>_i
End of Packet	EOF	h2d_st_eof_<n>_o	d2h_st_eof_<n>_i

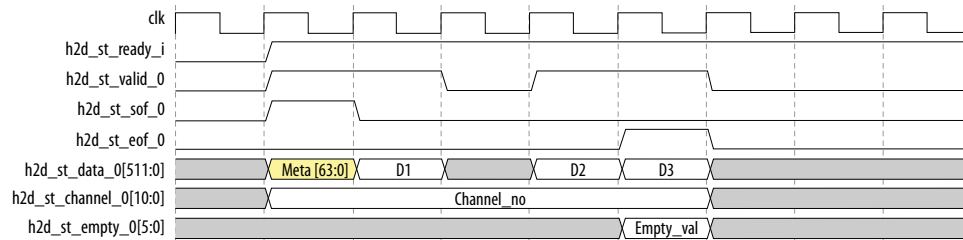
In Avalon-ST 1 port mode, a channel switch can only happen at packet boundary.

3.1.6.3. Metadata

When streaming DMA data, user can optionally enable 8-byte Metadata that contains metadata for user application. When enabled, the H2D descriptor destination address field is replaced with metadata and D2H descriptor source address field is replaced with Metadata.

With Metadata enabled, Avalon-ST SOF qualifies only the metadata and does not have any data. Since the metadata size is always 8 bytes with predefined property, user side does not expect an empty signal.

Figure 7. Avalon-ST Source Timing Diagram with Metadata enabled



3.1.7. User MSI-X

User MSI-X is arbitrated along with the H2D/D2H MSI-X/Writeback requests, and is handled exactly the same way as the others post the arbitration. The high-level diagram for the MSI-X handling mechanism is shown below.

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 to address the Descriptor completion related interrupts (DMA operation MSI-X) on both the paths.

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

Note: msix_queue_dir Queue direction. D2H = 0, H2D = 1

Note: MCDMA R-Tile IP Port 2 and Port 3 in Endpoint Mode do not support User MSI-X feature.

3.1.8. User Functional Level Reset (FLR)

When DMA engine receives Functional Level Resets from the PCIe Hard IP module, the reset requests are propagated to the downstream logic via this interface. In addition to performing resets to its internal logic, it waits for an acknowledgment from user logic for the reset request before it issues an acknowledgement to the PCIe Hard IP.

3.1.9. Control Registers

The Multi Channel DMA IP for PCI Express provides 4 MB of control register space that is internally mapped to PCIe BAR0. The control register block contains all the required registers to support the DMA operations. This includes QCSR space for individual queue control, MSI-X for interrupt generations, and GCSR for general global information.

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

Table 23. 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 2048 D2H and 2048 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 Control registers, refer to [Control Register \(GCSR\)](#) on page 153

3.2. Bursting Avalon-MM Master (BAM)

The BAM bypasses the Multi Channel DMA IP for PCI Express & provides a way for a Host to perform bursting PIO read/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 Avalon-MM burst read and write transactions, and sends back CplID TLPs for read requests it receives. Since the BAM user interface is Avalon-MM, the completions are always expected in order from user logic/Platform Designer fabric. The BAM supports bursts of up to 512 bytes and up to 32 outstanding read request.

BAM Address Mapping

You can select to map any BAR register other than the BAR0 of the physical function to BAM side 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 ($\log_2(\text{Number of PFs})$) is the RTL design parameter selected by the user such that Multi Channel DMA only allocates required number of the bits on Avalon-MM side to limit the number of the wires on the user interface. Example: If the number of PFs selected by user 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 ($\log_2(\text{Number of VFs})$) is the RTL design parameter selected by the user such that Multi Channel DMA only allocates required number of the bits on Avalon-MM side to limit the number of the wires on the user interface. Example: If the total number of VFs across all PFs selected by user is 32, the VF_NUM is 5.
4. **bar_num [2:0]:** This denotes the BAR number where the Avalon-ST 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 (max aperture of 4GB) of PF2/VF1 where only 3 PFs have been enabled by the user, 25 VFs have been enabled by the user, the BAM address is {1'b1, 2'b10, 5'b00001, 3'b011, bam_addr[31:0]}.

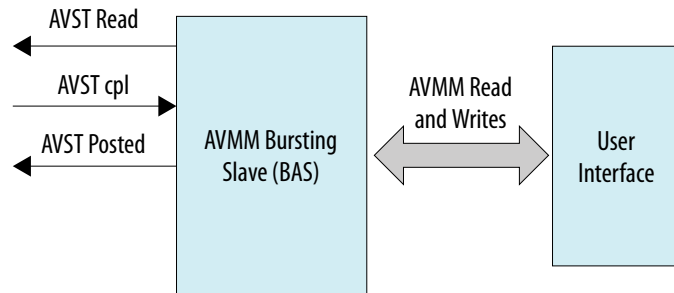
Note: For the Root Port: In the Root Port mode, the AVMM address output from BAM is the same as the one received on the Hard IP AVST.

3.3. Bursting Avalon-MM Slave (BAS)

The Avalon-MM TX Bursting slave module translates Avalon-MM Read and Write transactions from user logic to PCI Express Mrd and Mwr TLPs. The returned PCI Express Cpld packets is translated to Avalon-MM interface as response to Avalon-MM read transaction.

The BAS supports both 256 bit and 512 bit data widths to achieve bandwidths required for Gen4 x8 and Gen4 x16. It supports bursts up to 512 bytes and multiple outstanding read requests. The default support is only for the 64 NP outstanding.

Figure 8. Bursting Avalon-MM Slave Definition



Completion Re-ordering

Avalon-MM BAS interface is slave interface to the User Avalon-MM. The User AVMM can initiate AVMM reads to host interface and this translates to BAS Non-Posted 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 re-ordering module ensures the returned completions are re-ordered against the pending requests and send in the order on the AVMM interface since AVMM doesn't track out of order completions.

Note: BAS AVMM Slave waitrequestAllowance is 0, which means BAS can accept 0 additional command cycle after waitrequest is asserted.

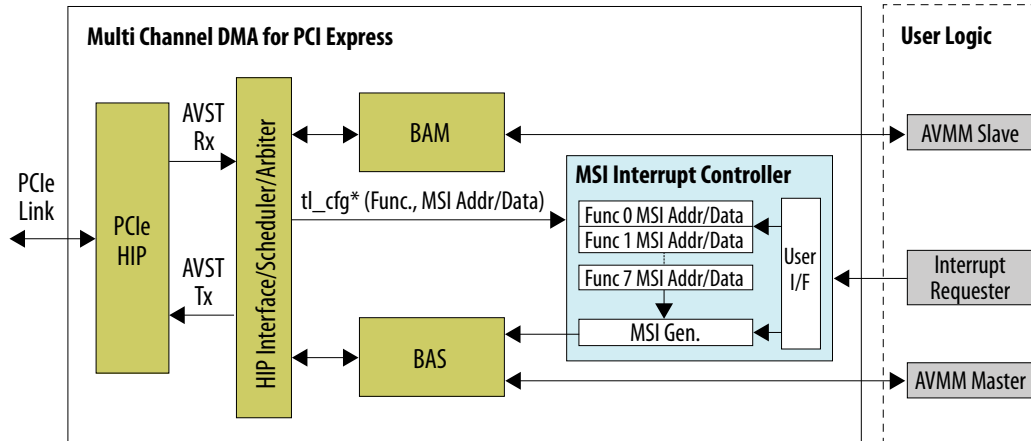
3.4. MSI Interrupt

3.4.1. 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 MSI Capability in Endpoint BAS or BAM+BAS mode, the IP core exposes MSI request interface to user logic. When user issues an MSI request through this interface, internal Interrupt Controller receives inputs such as function number and MSI number from user logic and generates AVMM Write to the BAS module as shown in the figure below. The BAS receives MSI signaling from interrupt controller and generating an MSI.

Figure 9. MSI Interrupt



* tl_cfg bus provides MSI Capability register and various config space register values.

Note: Endpoint MSI Interrupt is also available for H-Tile MCDMA IP starting with Quartus Prime Pro Edition 23.4 version.

Note: Endpoint MSI Interrupt is not supported for R-Tile MCDMA IP x4 Endpoint Ports 2 and 3.

3.4.2. MSI Interrupt Controller

The MSI Interrupt controller receives the necessary MSI information such as the message address and data from the PCIe Hard IP configuration output interface (tl_cfg_*). The MSI Interrupt controller uses this information to create Memory Write transaction. The MSI Interrupt controller has the necessary storage to hold the MSI addr/data for that specific function and user vector number.

The MSI Interrupt controller also gets the MSI Mask bits from the tl_cfg interface. If the MSI is masked for a specific function, the MSI Interrupt controller does not send the MSI for that function. In addition, it provides the Hard IP information on MSI Pending bits.

When the user requests the generation of the MSI, the user provides MSI vector (number) and user function information which the MSI Interrupt Controller indexes to get the MSI addr/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 (msi_num_i[4:0]) are used to manipulate the MSI data LSB bits as per the PCIe specification.

The BAS logic takes the MSI request from the MSI Interrupt controller and forms a Memory Write TLP. The internal request to Interrupt Gen is MSI_Pending & ~MSI_Mask. The MSI pending is User_MSI & MSI_En (from Hard IP).

Figure 10. MSI Controller

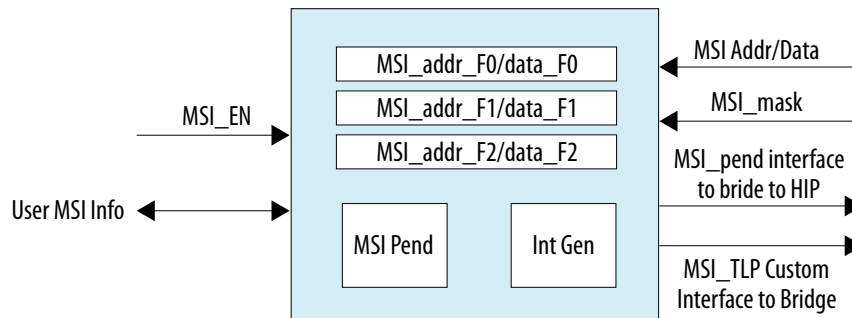


Figure 11. MSI Request Timing Diagram

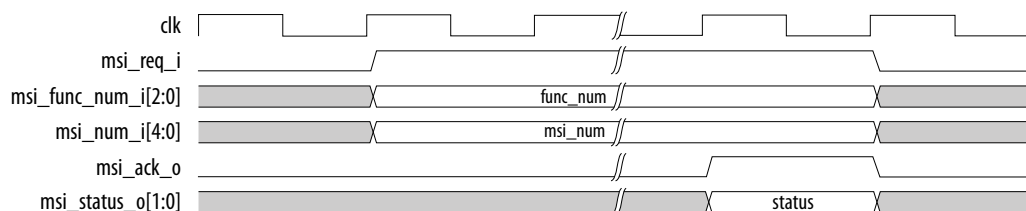
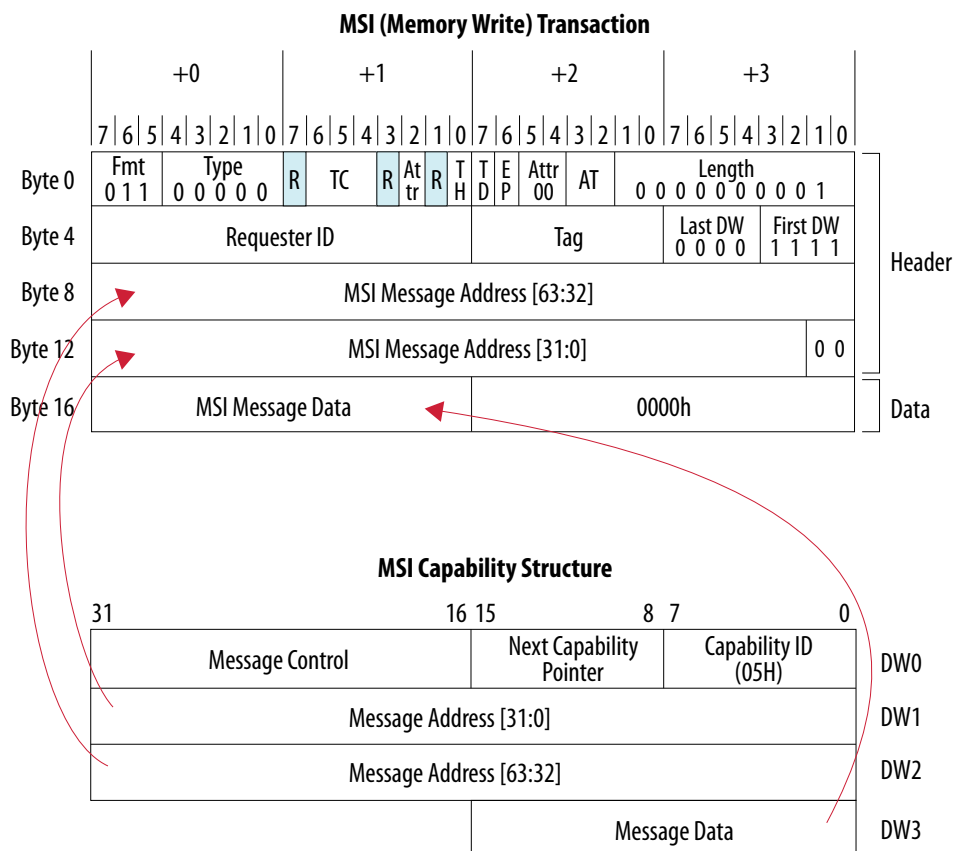


Figure 12. MSI Memory Write Transaction



The BAS logic Arbs for the Interrupt ping (on the Side band wires) from the Interrupt controller and sends the MSI on the AVST interface to the scheduler.

For MSI Request Interface signals, refer to [MSI Interface](#) on page 63

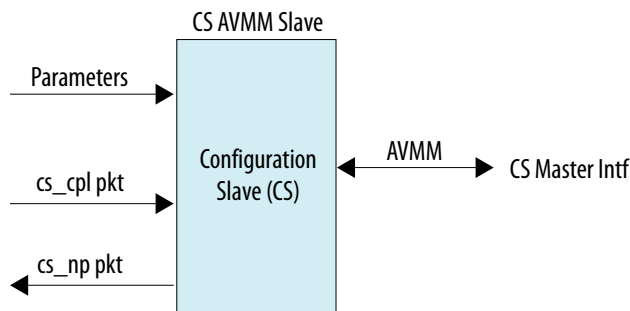
3.5. Config Slave (CS)

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

CS module converts the AVMM request into a configuration TLP with a fixed TAG value (decimal 255) assigned to it and sends it to scheduler. One unique TAG is enough as it doesn't support more than one outstanding transaction. This unique TAG helps in rerouting the completions to CS module.

Re-routing the completion is handled at the top level and since only 1 NP outstanding is needed, the TLP RX scheduler parses the completion field to decode the completion on a fixed TAG and route the transaction over to CS.

Figure 13. Avalon-MM Config Slave Module

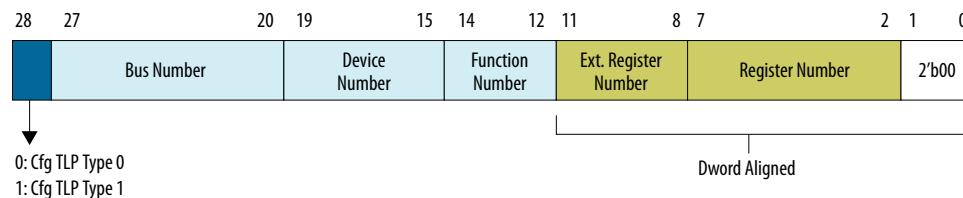


3.5.1. 29 bit AVMM address format

Config Slave AVMM Address

Config Slave interface supports 29-bit address format in Quartus Prime Pro Edition v21.1 and Quartus Prime Pro Edition v21.2.

Figure 14. 29 bit Address Format

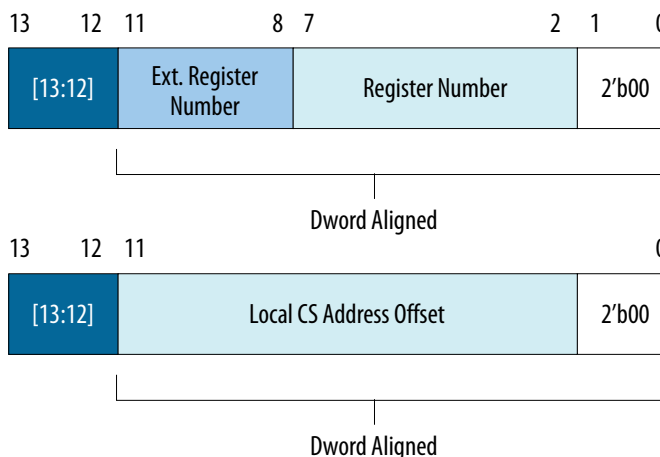


3.5.2. 14 bit AVMM address format

Config Slave supports 14-bit address format starting with Intel Quartus Prime Pro Edition v21.3. In this format, the AVMM slave address width is limited to 14 bits as shown in the figure below.

Note: Support for 29-bit address format is not available in Quartus Prime Pro Edition v21.3 onwards.

Figure 15. 14 bit Address Format



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

Table 24. Address Bits [13:12] Definition

Bits [13:12]	Description
2'b00	Config TLP Type 0
2'b01	Config 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 local CS registers that are supported in 14-bit address mode.

Table 25. 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 Address Translation Table
<i>continued...</i>			

Local CS Address Offset	Name	Access	Comment
			Note: Refer to Root Port Address Translation Table Enablement for information on an ATT programming example.

3.5.3. Configuration Access Mechanism

Table 26. Configuration Access Mechanism in 29-bit and 14-bit addressing

Access	29-bit Address	14-bit Address
EP Config Space Write	Single Write: AVMM write to EP Register address (AVMM address includes BDF+ Register) with actual data	Two Writes: <ul style="list-style-type: none"> Write BDF info to 0x0004 (with 13th bit set to 1) AVMM Write to EP Register address (with 13th bit set to 0) with actual data
EP Config Space Read	<ul style="list-style-type: none"> AVMM read to EP Register address (AVMM address includes BDF +Register) Type1/Type0 based on 28th bit CpID data is available on AVMM read data bus 	<ul style="list-style-type: none"> One AVMM write of BDF info to 0x0004 (with 13th bit set to 1) One AVMM read to EP Register address (with 13th bit set to 0) Type1/Type0 based on 12th bit CpID data is available on AVMM read data bus

Figure 16. Config Write Type 0 with 14-bit Address Format

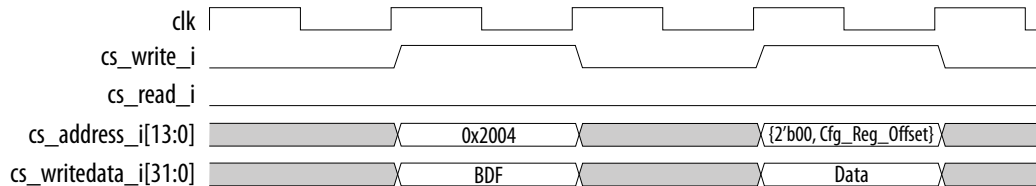
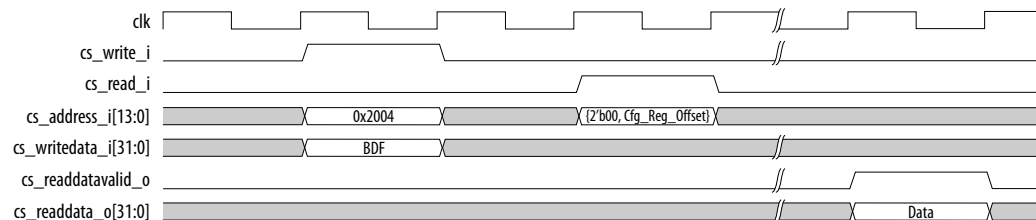


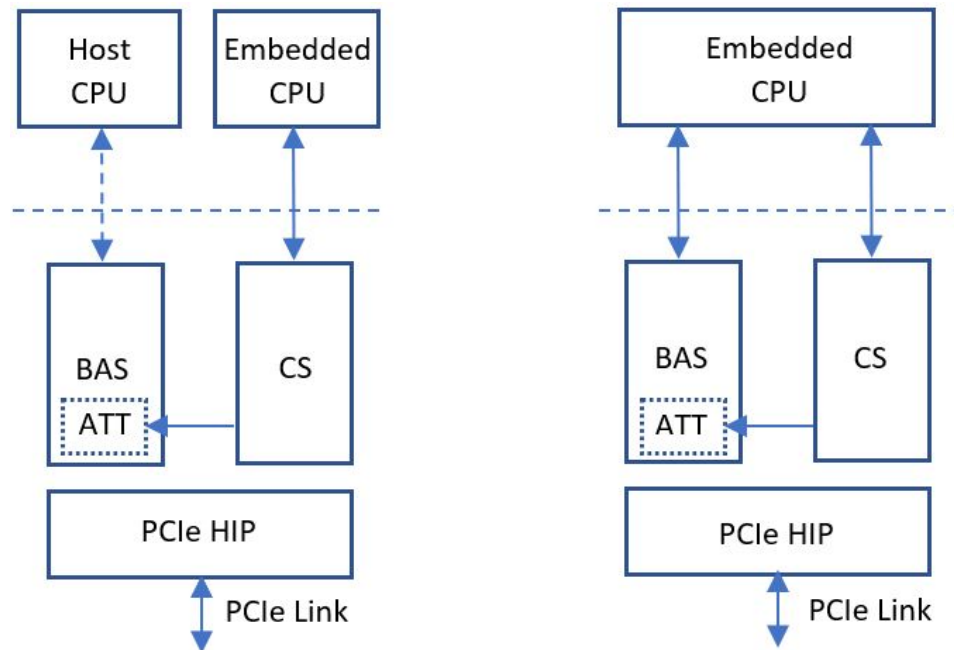
Figure 17. Config Read Type 0 with 14-bit Address Format



3.6. Root Port Address Translation Table Enablement

The Root Port Address Translation Table (ATT) translates the address driven by the Avalon-MM master connected to the Bursting Avalon-MM Slave (BAS). For example, if the embedded CPU with the smaller address bus is the master, 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 18. Example Use Cases with the Root Port Address Translation Table (ATT)



Note: For information about the CS register offset used to program ATT, refer to [Local CS Registers supported in 14-bit address mode](#).

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

- ATT Table Address Width (1-9): Sets the depth of ATT. Depth is equal to 2 to the power of 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 Avalon-MM slave address is used as-is in the resulting PCIe TLPs. When address mapping is enabled, burst of transactions on the Avalon-MM slave interfaces must not cross address mapping page boundaries. This requirement means $(\text{address} + 32 * \text{burst count}) \leq (\text{page base address} + \text{page size})$. Host software is expected to guarantee this in the Root Port mode and the BAS/CS does not have any mechanism to report the error if this requirement is violated.

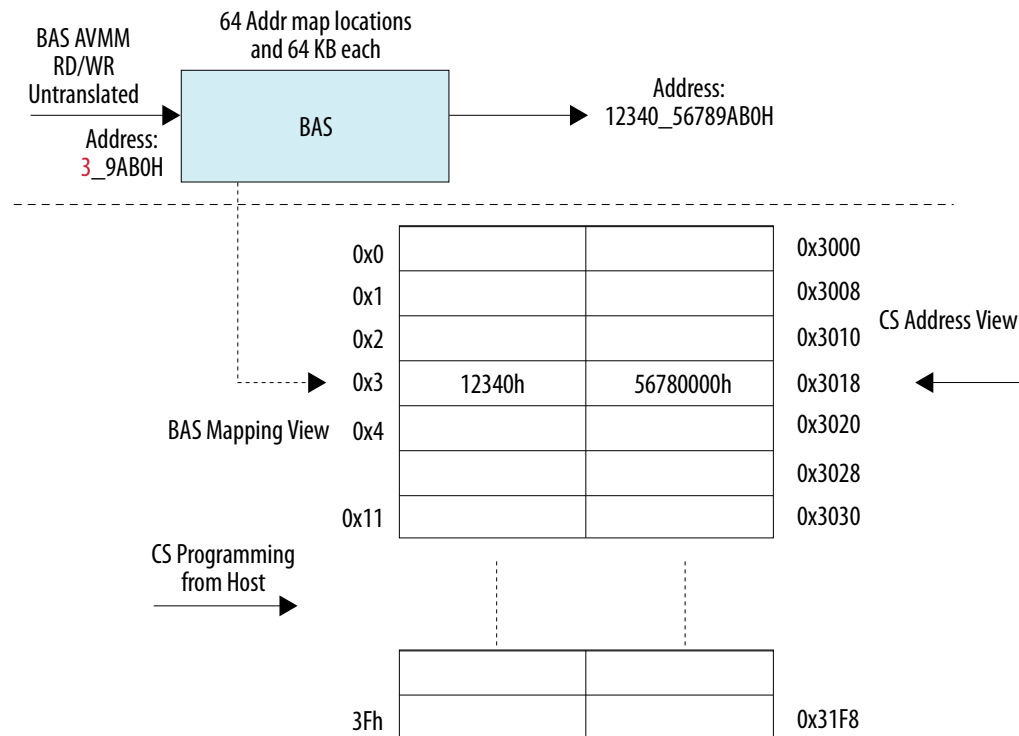
When address mapping is enabled, Avalon-MM lower-order address bits are passed through to the PCIe TLPs unchanged and are ignored in the address mapping table.

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

The number of lower-order address bits that are passed through defines the size of the page and is set by ATT Window Address Width parameter in the IP GUI. 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.

In the figure below, ATT depth is 64. This is set by ATT Table Address Width = 6 ($2^6 = 64$ deep). 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 19. 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 AVMM transaction is received on 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 above figure.

3.7. Hard IP Reconfiguration Interface

The Hard IP Reconfiguration Interface (usr_hip_reconfig_*) is supported on H-Tile, F-Tile, R-Tile and P-Tile. The Hard IP Reconfiguration interface is an Avalon-MM slave interface with a 21-bit address bus and an 8-bit data bus. You can use this bus to dynamically modify the value of configuration registers. This interface can be used in Endpoint and Root Port modes. It must be enabled if Root Port mode is selected. When you select Root Port mode, the IP Parameter Editor automatically enables this

interface. In Root Port mode, the application logic uses the Hard IP reconfiguration interface to access its PCIe configuration space to perform link control functions such as Hot Reset, Link Disable or Link Retrain.

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

Note: H-Tile MCDMA IP does not respond with completion to a Hard IP reconfiguration read or write request when those are targeted to **0x0FFC - 0xFF** address range. These addresses belong to the PCIe configuration space.

3.8. Config TL Interface

The Config TL interface extracts the required information stored in the PCIe Hard IP config space in order for the DMA to operate properly. Some of the example includes MPS, MRRS, and Bus Master Enable.

The configuration register extraction only occurs periodically, so the assumption is made that these are fairly static signals and there are significant delays after the config space is updated by software.

Note: MCDMA R-Tile IP does not support Config TL interface. The CII interface should be used as a replacement for similar functionality.

3.9. Configuration Intercept Interface (EP Only)

For detailed information about this interface, refer to *P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide (Chapter 4 Section 4.11)* or *F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide (Section 3.9 and Section 5.11)* or *R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide (Section 4.3.7)*

Related Information

- [P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)
- [F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)
- [R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)

3.10. Data Mover Only

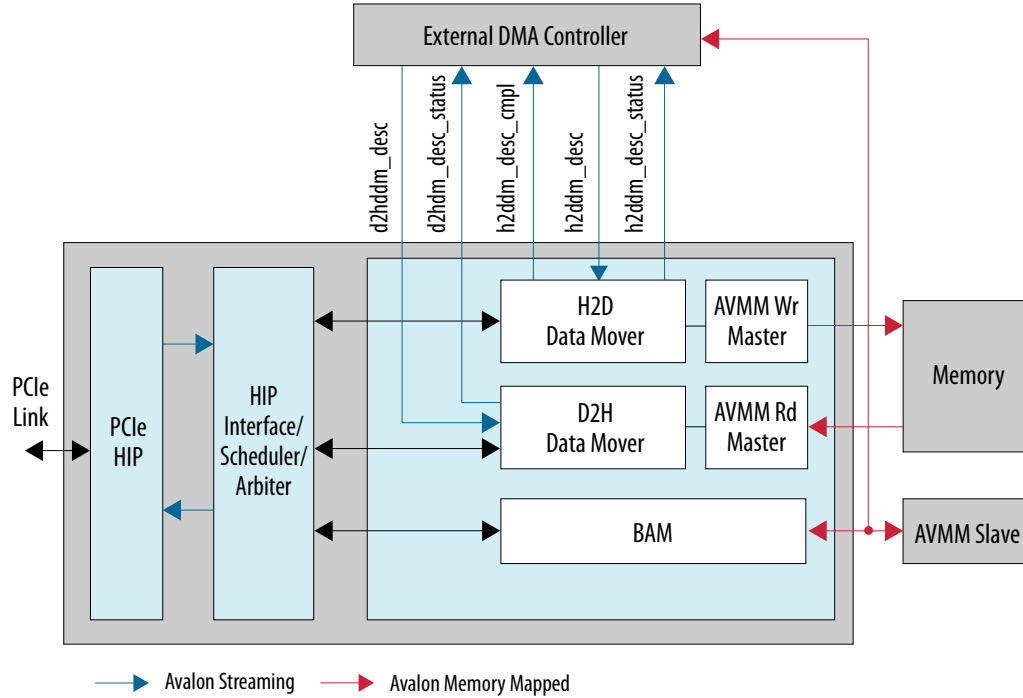
Data Mover Only mode is one of the MCDMA IP user modes for Endpoint MCDMA R-Tile, P-Tile and F-Tile IPs. This mode supports external descriptor fetch based on external descriptor controller. In this mode, the MCDMA IP core becomes a PCIe Data Mover subsystem implementing the usual PCIe semantics, but it does not have any blocks related to DMA descriptor functionality.

Note: Data Mover with the External Descriptor Controller only supports data movement over AVMM interface to the user logic. AVST interface support to user logic may be added in future.

Note: Data Mover only mode is not available for any of the x4 topologies in P/F/R-Tile MCDMA IPs.

The figure below is the top-level block diagram of the MCDMA IP in Data Mover Only mode with user descriptor controller.

Figure 20. PCIe Data Mover Subsystem connected to External DMA Controller



Data mover subsystem supports the following functionality:

- **HIP Interface:** Adapts to various PCIe HIP variants (P-Tile/F-Tile/R-Tile) the Data Mover subsystem is interfacing to.
- **Host to Device (H2D) Data Mover:** Provides AVST source and sink interfaces to external descriptor controller for both descriptor fetch and data move operations
- **Device to Host (D2H) Data Mover:** Provides AVST source and sink interfaces to external descriptor controller to initiate data move operations through descriptors. Also, you can use the same AVST interface to send writeback/ MSIx interrupt by setting the appropriate fields in the descriptor.

Note: Data mover IP does not implement the MSI-X controller (including MSIx-table and PBA-table) internally. You must implement the MSI-X controller in your logic and use this interface to only send the MSIx interrupt to PCIe Host using descriptors.

- Bursting Avalon Master (BAM): Performs non-bursting PIO operations for register programming.
- Support for PCIe semantics
 - Scheduler enforces PCIe ordering rules in both Tx and Rx direction leading to PCIe HIP
 - TLP chunking at MPS size
 - Check for the 4KB cross over and other error logging
- Completion re-ordering: Data Mover subsystem performs the re-ordering of the received completions before sending data to AVMM Write Master or sending the descriptor completion packets to external descriptor controller.

3.10.1. H2D Data Mover

Table 27. H2D Data Mover Interface

* = Interface name in IP Parameter Editor Block Symbol

Interface Name*	Type	Description
h2ddm_desc	AVST Sink	H2D data mover gets the descriptor from external descriptor controller. Data mover throttles external descriptor controller with ready signal.
h2ddm_desc_cmpl	AVST Source	H2D data mover sends the completion (descriptor data from host memory) to external descriptor controller on this interface.
h2ddm_desc_status	AVST Source	Once a descriptor is complete, H2D data mover sends the status information (success, error, descriptor ID, application specific bits) to external descriptor controller
h2ddm_master	AVMM Write Master	This interface provides the read data from the host memory to the user application.

3.10.1.1. H2D Descriptor Format (h2ddm_desc)

Table 28. H2D Descriptor Format

Name	Width	Description
SRC_ADDR [63:0]	64	Starting system address of allocated transmit buffer read by DMA.
DEST_ADDR [127:64]	64	Starting local AVMM address written by DMA. Valid only when MM_mode=1. RSVD when MM_mode = 0.
PYLD_CNT [147:128]	20	Payload size in bytes. MM_mode=0: The DMRd is meant for descriptor fetch and completion is returned on the Descriptor completion interface. In this case, upper 10 bits are reserved.
continued...		

Name	Width	Description
		MM_Mode=1: The DMRd is for data fetch and completion is sent to AVMM Write master interface. In this case, entire 20 bits are valid. 20'h0 means 1MB.
RSVD	1	Reserved
App_specific_bits [151:149]	3	Application-specific bits. Reserved for H2D (0x0).
DESC_IDX1 [167:152]	16	Unique Identifier for each descriptor. The same ID is applied to the AVST source status signaling interface (h2ddm_desc_status) returning the status of the Data mover completion to DMA controller.
RSVD [173:168]	6	Reserved
MCDMA mode [174:174]	1	Hardwired to 0. Indicates Data Mover mode, i.e., external DMA.
MM_mode [175:175]	1	MM_mode=0: Indication to H2D Data Mover to transfer the completion to AVST source CMPL interface (h2dm_desc_cmpl) to external DMA controller. Descriptor fetches happen only on the H2D Data Mover with MM_mode=0. MM_mode=1: Indication to H2D Data Mover to transfer the data to AVMM interface. Indicates that the descriptor is a data movement command. DMRd command with MM_mode=1 reads the data from host memory and write it to local FPGA memory.
DM_FmtType [183:176]	8	'h20=DMRd
PFVF [199:184]	16	{VF_ACTIVE, VFNUM[10:0], PF[3:0]}
RSVD [200:200]	1	Reserved
RSVD [216:201]	16	Reserved
DESC_IDX2 [228:217]	12	Optional. Descriptor ID field providing additional provision for descriptor fetch engine to embed information such as channel number, etc. which the completion status on AVST will return unedited for the response completion packet. If not used, the user external DMA controller is expected to drive this field to zero. The data mover subsystem will treat the descriptor ID field as {DESC_IDX2, DESC_IDX1}.
RSVD [255:229]	27	Reserved

3.10.1.2. H2D Descriptor Completion Packet Format (h2ddm_desc_cmpl)

The H2D DM descriptor completion data which is the returned completion from the host when original descriptor request was with MM_Mode=0 is as follows.

Table 29. H2D Descriptor Completion Packet Format

Name	Width	Description
CMPL_LEN [13:0]	14	Length of the completion in bytes. Completion data are returned from next cycle until the end of packet when the EOP is asserted.
LOWER_ADDR [37:14]	24	The lower address indicates lower 24 bits of starting byte address in current completion corresponding to first completion only.
RSVD	1	Reserved
DM_FmtType [46:39]	8	'h4A: DMCmpl
PFVF [62:47]	16	{VF_ACTIVE, VFNUM[10:0], PF[3:0]} copied from original request.
Cmpl_sts [65:63]	3	3'b000: Success 3'b001: Not Successful. All other values are reserved.
DESC_IDX1 [81:66]	16	Unique Identifier for each descriptor. This is the same ID copied from original request.
En_partial_cmpl_data [82:82]	1	<i>Note:</i> This parameter is not supported in the Quartus Prime 22.1 release.
DESC_IDX2 [94:83]	12	AVST completion DESC_IDX2 value which is copied from original request.
RSVD [255:95]	161	Reserved
Completion data	256b / 512b	<i>Note:</i> This parameter is not supported in the Quartus Prime 22.1 release. Partial completion data (En_Partial_cmpl_data=1) <i>Note:</i> This is Param based data width

3.10.1.3. H2D Descriptor Status (h2ddm_desc_status)

The table below is the H2D Data Mover descriptor status sent to the external DMA controller when it completes the execution of a descriptor.

Note: This is intended to be used only when MM_mode=1 in original H2D Data Mover descriptor.

Table 30. H2D Descriptor Status

Name	Width	Description
DESC_IDX1[15:0]	16	Unique Identifier for each descriptor. This is copied from original request.
App_specific_bits[18:16]	3	Application specific bits
Error[19]	1	Error Status [CISE to elaborate on error condition]
DESC_IDX2[31:20]	12	DESC_IDX2 copied from original request

3.10.2. D2H Data Mover

Table 31. D2H Data Mover Interface

* = Interface name in IP Parameter Editor Block Symbol

Interface Name*	Type	Description
d2hdm_desc	AVST Sink	D2H data mover gets the descriptor from external descriptor controller. This interface is also used to send the DMWr or DWIntr command (MM_mode=0) to send the writeback to host.
d2hdm_desc_status	AVST Source	Once a descriptor is complete, D2H data mover returns the status of the D2H descriptors.
d2hdm_master	AVMM Read Master	D2H data mover reads the data from local memory and writes the data to host memory location.

3.10.2.1. D2H Descriptor Format (d2hdm_desc)

Table 32. D2H Descriptor Format

Name	Width	Description
SRC_ADDR [63:0]	64	Starting local memory address of allocated transmit buffer read by DMA and instruction must be DMWr for DMA operation with MM_mode=1. Application specific bit indicates whether this command is an Interrupt. If App_specific_bit=001, MM_mode=0 MM_mode=1: Starting local memory address of allocated transmit buffer read by DMA and instruction must be DMWr for DMA operation. MM_mode=0: Expectations is to use the DMWr command for WB / MSIX interrupt.
DEST_ADDR [127:64]	64	Destination system address where the DM sends the data to in the host memory. When MM_mode=0, instruction is DMWr, app_specific_bits=001 <ul style="list-style-type: none"> When app_specific_bits=010, contains the WB address H/L from external DMA's QCSR
PYLD_CNT [147:128]	20	DMA payload size in bytes. Max 1 MB, with 20'h0 indicating 1 MB
DM_FmtType [155:148]	8	'h60: DMWr
RSVD [159:156]	1	Reserved
PFVF [175:160]	16	{VF_ACTIVE, VFNUM[10:0], PF[3:0]}
MM_mode [176:176]	1	MM_mode=0: Only when Write back is intended. It is illegal and behavior is undefined if a DMRd is instructed with MM_Mode=0;
continued...		

Name	Width	Description
		MM_mode=1: Indication to D2H DM to transfer the data to Host side. MM_mode=1 with DMWr command enables reading local FPGA memory and writing to host memory.
App_specific_bits [179:177]	3	Application-specific bits. An example use case is that the external DMA controller set these bits to generate the Interrupt.
DESC_IDX1 [195:180]	16	Unique Identifier for each descriptor, the same ID will be applied AVST source status signaling (d2hdm_desc_status) returning the status of the data mover completion to DMA controller.
RSVD [196:196]	1	Reserved
RSVD [212:197]	16	Reserved
DESC_IDX2 [224:213]	12	Optional. Descriptor ID field providing additional provision for descriptor fetch engine to embed information such as channel number, etc. which the completion status on AVST will return unedited for the response completion packet. If not used, the user external DMA controller is expected to drive this field to zero. The data mover subsystem will treat the descriptor ID field as {DESC_IDX2, DESC_IDX1}.
RSVD [255:225]	31	Reserved

Table 33. Writeback and MSI-X Format

D2H Data Mover Descriptor Data Bus	Write Back	MSI-X
d2hdm_desc_data_i [63:0]	64-bits Writeback Data: Write-back Data [63:0] 32-bits Writeback Data: {32'h0, Write-back Data [31:0]}	32-bits MSI-X Data: {32'h0, MSI-X Data [31:0]}
d2hdm_desc_data_i [127:64]	Write-back Host Address [63:0]	MSI-X Address [63:0]
d2hdm_desc_data_i [147:128]	Payload size in bytes 64-bits Write back Data: 'd8 32-bits Write back Data: 'd4	Payload size in bytes 32-bits MSI-X Data: 'd4
d2hdm_desc_data_i [155:148]	Data Mover Write: 'h60	Data Mover Write: 'h60
d2hdm_desc_data_i [159:156]	Reserved : '0	Reserved: '0
d2hdm_desc_data_i [175:160]	{VF_ACTIVE, VFNUM [10:0], PF[3:0]}	{VF_ACTIVE, VFNUM [10:0], PF[3:0]}
d2hdm_desc_data_i [176:176]	MM_mode : 1'b0	MM_mode: 1'b0
d2hdm_desc_data_i [255:177]	Reserved : '0	Reserved: '0

3.10.2.2. D2H Descriptor Status (d2hdm_desc_status)

The table below is the D2H Data Mover descriptor status sent to the external DMA controller when it completes the execution of a descriptor.

Table 34. D2H Descriptor Status

Name	Width	Description
DESC_IDX1 [15:0]	16	Unique Identifier for each descriptor. This is copied from original request.
App_specific_bits [18:16]	3	Application specific bits
Error [19]	1	Error Status Reserved field only.
DESC_IDX2 [31:20]	12	DESC_IDX2 copied from original request

3.10.3. Application Specific Bits

Table 35. Application Specific Bits

Bit [2]	Bit [1]	Bit [0]	Description
0	1	1	Reserved
0	1	0	Reserved
0	0	1	Data mover considers this as WB and picks the address/ data from AVST sink interface in D2H direction (d2hdm_desc). Not applicable for H2D, and external DMA controller must drive 0.
0	0	0	Reserved



4. Interface Overview

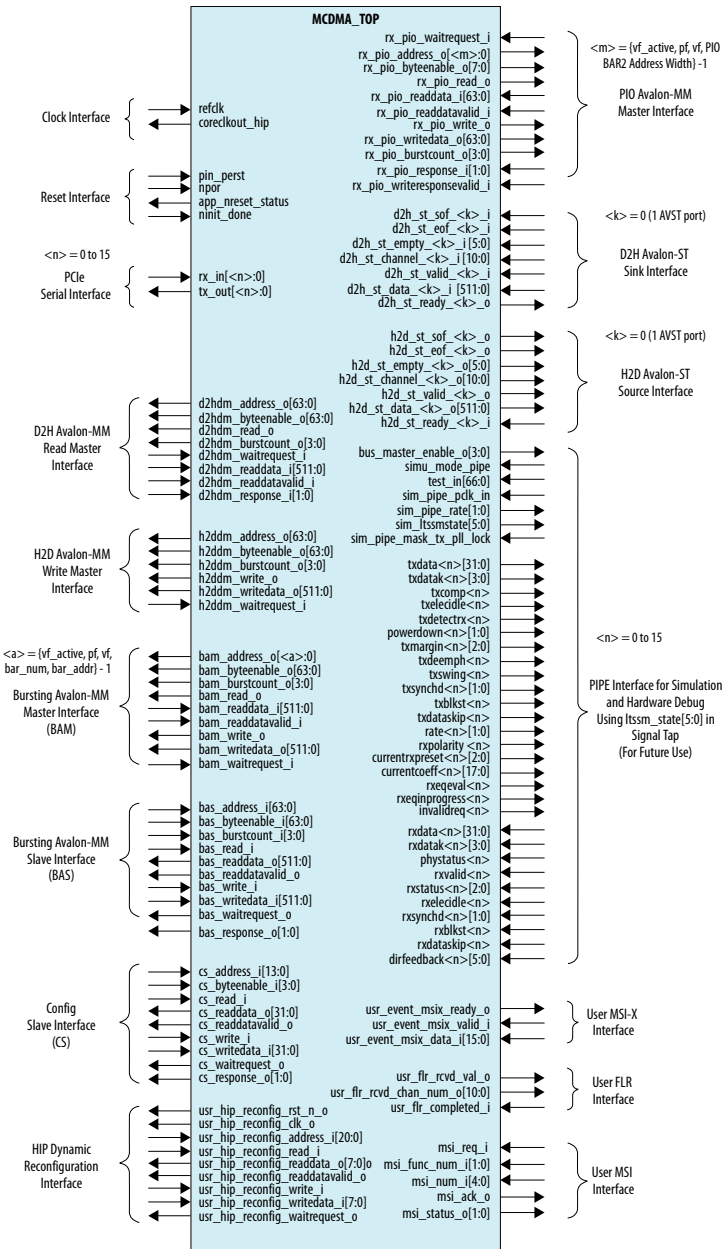
Interfaces for the Multi Channel DMA IP for PCI Express are:

- Clocks
- Resets
- Multi Channel DMA mode interfaces (EP only):
 - Avalon-MM PIO Master Interface
 - Avalon-MM Write Master Interface
 - Avalon-MM Read Master Interface
 - Avalon-ST Source Interface
 - Avalon-ST Sink Interface
 - User MSI-X
 - User FLR
- Bursting Avalon-MM Master Interface (BAM)
- Bursting Avalon-MM Slave Interface (BAS)
- MSI Interface (in BAS mode & BAM+BAS mode for EP H/P/F/R-Tile MCDMA IP)
- Config Slave Interface (RP only)
- Hard IP Reconfig Interface
- Config TL Interface
- Data Mover Mode (available in MCDMA P-Tile, R-Tile and F-Tile IPs)

4.1. Port List

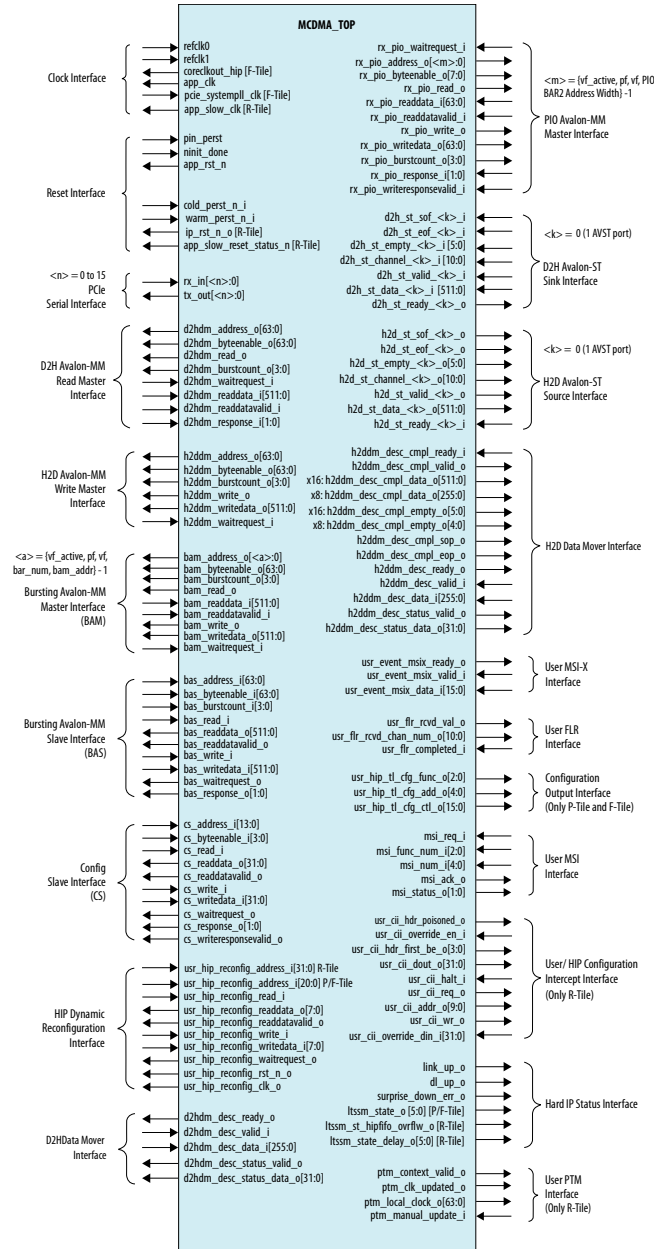
4.1.1. Port List (H-Tile)

Figure 21. Multi Channel DMA IP for PCI Express Port List (H-Tile)



4.1.2. Port List (P-Tile) (F-Tile) (R-Tile)

Figure 22. Multi Channel DMA IP for PCI Express Port List (P-Tile) (F-Tile) (R-Tile)



4.2. Clocks

Table 36. Multi Channel DMA IP for PCI Express Clock Signals

Signal Name	I/O Type	Description	Clock Frequency
H-Tile			
refclk	Input	PCIe reference clock defined by the PCIe specification. This input reference clock must be stable and free-running at device power-up for a successful device configuration.	100 MHz ± 300 ppm
coreclkout_hip	Output	This is an output clock provided to user logic. Avalon-MM / Avalon-ST user interfaces are synchronous to this clock.	250 MHz
P-Tile and F-Tile and R-Tile			
refclk0	Input	PCIe reference clock defined by the PCIe specification. These clocks must be free-running and driven by the single clock source. For F-Tile, connect <code>outrefclk_fgt_i</code> (i = 0 to 7) from "F-Tile Reference and SystemPLL Clocks" IP to this port. Drive <code>refclk1</code> input port with the same clock for <code>refclk0</code> input port if your design does not need a separate <code>refclk</code> .	100 MHz ± 300 ppm
refclk1	Input		
coreclkout_hip	Output	Clock <i>Note:</i> Not available for P-Tile. <i>Note:</i> Not available for R-Tile. <i>Note:</i> In earlier versions, this signal was present. Manual upgrade is required.	
app_clk	Output	Application clock	Gen3: 250 MHz Gen4: 400 MHz (Intel Stratix 10 DX), 500 MHz (Agilex 7)
app_slow_clk	Output	Clock for sideband signals. <i>Note:</i> Only available for R-Tile on Quartus Prime 22.4 version and onwards.	
<i>continued...</i>			

Signal Name	I/O Type	Description	Clock Frequency
		<i>Note:</i> Divide-by-2 or Divide-by-4 clock derived from <code>coreclkout_hip</code> . Use the Slow Clock Divider option in the Parameter Editor to choose between the divide by 2 or 4 versions of <code>coreclkout_hip</code> for this clock.	
<code>pcie_systempll_clk</code>	Input	<p>System PLL clock from "F-Tile Reference and SystemPLL Clocks" IP. Connect <code>out_systempll_clk_0</code> from the "F-Tile Reference and System PLL Clocks" IP to this port.</p> <p>For Mode of System PLL setting, select frequency that is two times of the selected PLD Clock Frequency. For example, if the selected PLD clock frequency is 500 MHz, use the "PCIE_FREQ_1000" setting.</p> <p><i>Note:</i> Only available for F-Tile.</p>	

4.2.1. F-Tile System PLL Reference Clock Requirements

The F-Tile Reference and System PLL Clocks Intel FPGA IP is required IP for MCDMA F-Tile designs to configure the reference clock for the PCI Express channels and configure the System PLL.

Reference clock to the System PLL can be driven to any one of the F-Tile reference clock pins, `refclk[0]` to `refclk[7]`. The reference clock must adhere to the following requirements:

- If compliance to PCI Express link training timing specifications are required, the reference clock to System PLL must be available and stable before device configuration begins. You must set the `Refclk` is available at power-on parameter in the System PLL IP to **On**. Derive the reference clock from an independent and free running clock source. Alternately, if the reference clock from the PCIe link is guaranteed available before device configuration begins, you can use it to drive the System PLL. Once the PCIe link `refclk` is alive, it is never allowed to go down.
- If compliance to PCI Express link training timing specifications are not required and the reference clock to System PLL may not be available before device configuration begins, you must set the `Refclk` is available at power-on parameter in the System PLL IP to **Off**. In this case, you may use the reference clock from the PCI Express link to drive the System PLL. The System PLL does not lock to the reference until you perform the Global Avalon memory-mapped interface write operations signaling that the reference clock is available.

Once the reference clock for the System PLL is up, it must be stable and present throughout the device operation and must not go down. If you are not able to adhere to this requirement, you must reconfigure the device.

Note: Refer to *Implementing the F-Tile Reference and System PLL Clocks Intel FPGA IP* section in *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide* for information about this IP.

Note: Refer to *Example Flow to Indicate All System PLL Reference Clocks are Ready* section in *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide* to trigger the System PLL to lock to reference clock.

Related Information

- [F-Tile Architecture and PMA and FEC Direct PHY IP User Guide](#)
- [Implementing the F-Tile Reference and System PLL Clocks Intel® FPGA IP](#)
- [Example Flow to Indicate All System PLL Reference Clocks are Ready](#)

4.3. Resets

Table 37. Multi Channel DMA IP for PCI Express Reset Signals

Signal Name (<n> = Port Number)	I/O Type	Description
H-Tile		
pin_perst_n	Input	This is an active-low input to the PCIe Hard IP, and implements the PERST# function defined by the PCIe specification.
npwr	Input	Application drives this active-low reset input to the PCIe Hard IP. This resets entire PCIe Hard IP. If not used, you must tie this input to 1.
app_nreset_status	Output	This is an active low reset status. This is deasserted after the PCIe Hard IP has come out of reset.
ninit_done	Input	This is an active low input signal. A "1" indicates that the FPGA device is not yet fully configured. A "0" indicates the device has been configured and is in normal operating mode. To use the ninit_done input, instantiate the Reset Release Intel FPGA IP in your design and use its ninit_done output. The Reset Release IP is required in Intel Stratix 10 design. It holds the Multi Channel DMA for PCI Express IP in reset until the FPGA is fully configured and has entered user mode.
P-Tile and F-Tile and R-Tile		
pin_perst_n	Input	See H-Tile pin_perst description
ninit_done	Input	See H-Tile ninit_done description
continued...		

Signal Name (<n> = Port Number)	I/O Type	Description
app_rst_n	Output	For EP mode: Resets the MCDMA soft IP blocks and user logic. app_rst_n is asserted when software writes to the SW_RESET register bit[0], when in hot reset by the input app_nreset_status. For RP mode: The output app_rst_n follows the input app_nreset_status.
i_gpio_perst#<n>_n	Input	This is an active-low reset to each port when Enable Independent Perst option is enabled. <i>Note:</i> This signal is only enable for F-Tile.
p<n>_app_slow_reset_status_n	Output	This is the equivalent signal for app_rst_n in the slow_clk domain. <i>Note:</i> This signal is only enabled for R-Tile.
p<n>_cold_perst_n_i	Input	Only available when the Enable Independent GPIO Perst parameter in the Top-Level Settings tab is selected, these active-low signals independently trigger cold resets to individual PCIe Controllers. If these inputs are not used, they should be tied off to 1.
p<n>_warm_perst_n_i	Input	Only available when the Enable Independent GPIO Perst parameter in the Top-Level Settings tab is selected, these active-low signals independently trigger cold resets to individual PCIe Controllers. If these inputs are not used, they should be tied off to 1.
p<n>_ip_rst_n_o	Output	Only available when the Enable Independent GPIO Perst parameter in the Top-Level Settings tab is selected, these active-low output signals are exposed to the application logic and indicate the status of the Hard Reset Controller triggering resets to individual PCIe Controllers. <i>Note:</i> This signal is only enabled for R-Tile.

4.4. Multi Channel DMA

4.4.1. Avalon-MM PIO Master

The Avalon-MM PIO Master interface is used to write to /read from external registers implemented in the user logic.

Table 38. Avalon-MM PIO Master

Interface Clock Domain for H-Tile: coreclkout_hip

Interface Clock Domain for P-Tile, F-Tile and R-Tile: app_clk

Signal Name	I/O Type	Description
rx_pio_address_o[n:0]	Output	PIO Read/Write Address. For PIO interface address mapping, refer to Avalon-MM PIO Master on page 25
rx_pio_writedata_o[63:0]	Output	PIO Write Data Payload.
rx_pio_byteenable_o[7:0]	Output	PIO Write Data Byte Enable. <i>Note:</i> Not applicable for read transfers
rx_pio_write_o	Output	PIO Write.
rx_pio_read_o	Output	PIO Read
rx_pio_burstcount_o[3:0]	Output	PIO Write Burst Count.
rx_pio_waitrequest_i	Input	PIO Write WaitRequest.
rx_pio_writeresponsevalid_i	Input	PIO response valid to a write request
rx_pio_readdata_i[63:0]	Input	PIO Read Data.
rx_pio_readdatavalid_i	Input	PIO Read data valid
rx_pio_response_i[1:0]	Input	PIO response. Reserved for future release. Tie to 0.

4.4.2. Avalon-MM Write Master (H2D)

The H2D Avalon-MM Write Master interface is used to write H2D DMA data to the external Avalon-MM slave. This port is 128-bit (x4) / 256-bit (x8/x4*)/ 512-bit (x16) write master that is capable of writing maximum 512 bytes of data per AVMM transaction. The `WaitRequestAllowance` of this port is enabled allowing the master to transfer continuously N data phases after the `WaitRequest` signal has been asserted.

Value of <N> for H2D AVMM Master is as follows:

- 512-bit data-width is 16
- 256-bit data-width is 32
- 128-bit data-width is 64

Note: In R-Tile, the Port x4 can be 256 bit write master when Gen4 4x4 Interface - 256 bit is selected in PCI Express Hard IP Mode.

Table 39. Avalon-MM Write Master (H2D)

Interface Clock Domain for H-Tile: coreclkout_hip

Interface Clock Domain for P-Tile, F-Tile and R-Tile: app_clk

Signal Name	I/O Type	Description
h2ddm_waitrequest_i	Input	H2D Wait Request
h2ddm_write_o	Output	H2D Write
h2ddm_address_o[63:0]	Output	H2D Write Address
x16: h2ddm_burstcount_o[3:0] x8/x4*: h2ddm_burstcount_o[4:0] x4 (128-bit): h2ddm_burstcount_o[5:0]	Output	H2D Write Burst Count
x16: h2ddm_writedata_o[511:0] x8/x4*: h2ddm_writedata_o[255:0] x4 (128-bit): h2ddm_writedata_o[127:0]	Output	H2D Write Data Payload
x16: h2ddm_byteenable_o[63:0] x8/x4*: h2ddm_byteenable_o[31:0] x4 (128-bit): h2ddm_byteenable_o[15:0]	Output	H2D Byte Enable

4.4.3. Avalon-MM Read Master (D2H)

The D2H Avalon-MM Read Master interface is use to read D2H DMA data from the external AVMM slave. This port is 128-bit (x4) / 256-bit (x8/x4*) / 512-bit (x16) read master that is capable of reading maximum 512 bytes of data per AVMM transaction. The waitrequestAllowance of this port is enabled, allowing the master to transfer up to N additional write command cycles after the waitrequest signal has been asserted. Value of <N> for H2D AVMM Master is as follows:

- 512-bit data-width is 16
- 256-bit data-width is 32
- 128-bit data-width is 64

Note: In R-Tile, the Port x4 can be the 256-bit read master when Gen4 4x4 Interface - 256-bit is selected in the PCI Express Hard IP Mode.

Table 40. Avalon-MM Read Master (D2H)

Interface Clock Domain for H-Tile: coreclkout_hip

Interface Clock Domain for P-Tile, F-Tile and R-Tile: app_clk

Signal Name	I/O Type	Description
d2hdm_read_o	Output	D2H Read.
d2hdm_address_o[63:0]	Output	D2H Read Write Address.
x16: d2hdm_byteenable_o[63:0]	Output	D2H Byte Enable

continued...

Signal Name	I/O Type	Description
x8/x4*: d2hdm_byteenable_o[31:0] x4 (128-bit): d2hdm_byteenable_o[15:0]		
x16: d2hdm_burstcount_o[3:0] x8/x4*: d2hdm_burstcount_o[4:0] x4 (128-bit): d2hdm_burstcount_o[5:0]	Output	D2H Burst Count.
d2hdm_waitrequest_i	Input	D2H Write WaitRequest.
d2hdm_readdatavalid_i	Input	D2H Read Data Valid.
x16: d2hdm_readdata_i[511:0] x8/x4*: d2hdm_readdata_i[255:0] x4 (128-bit): d2hdm_readdata_i[127:0]	Input	D2H Read Data.
d2hdm_response_i[1:0]	Input	Tied to 0

4.4.4. Avalon-ST Source (H2D)

The H2D Avalon-ST source interface is used to send H2D DMA data to the external Avalon-ST sink logic.

Table 41. Avalon-ST Source (H2D)

<n> = 0 (1 port mode)

Signal Name	I/O Type	Description
x16: h2d_st_data_<n>_o[511:0] x8/x4 (256-bit): h2d_st_data_<n>_o[255:0] x4 (128-bit): h2d_st_data_<n>_o[127:0]	Output	H2D Streaming data from host to device
h2d_st_valid_<n>_o	Output	Valid for all outgoing signals. A '1' represents the readiness of data to be sent.
h2d_st_ready_<n>_i	Input	Backpressure from device. A '1' represents, device readiness for receiving data.
h2d_st_sof_<n>_o	Output	Start of file (or packet) as instructed in host descriptor.
h2d_st_eof_<n>_o	Output	End of file (or packet) as instructed in host descriptor.
x16: h2d_st_empty_<n>_o[5:0] x8/x4 (256-bit): h2d_st_empty_<n>_o[4:0] x4 (128-bit): h2d_st_empty_<n>_o[3:0]	Output	Represents the number of empty bytes in h2d_st_data_<n>_o, and valid only when both h2d_st_valid_<n>_o and h2d_st_eof_<n>_o is '1'.
h2d_st_channel_<n>_o[10:0]	Output	To support multi-Channel per port.

4.4.5. Avalon-ST Sink (D2H)

The D2H Avalon-ST Sink interface is used to read D2H DMA data from the external Avalon-ST source logic.

Table 42. Avalon-ST Sink (D2H)

<n> = 0 (1 port mode)

Signal Name	I/O Type	Description
d2h_st_valid_<n>_i	Input	Valid for all incoming signals. A '1' represents the device readiness for data to be sent.
x16: d2h_st_data_<n>_i[511:0] x8/x4 (256-bit): d2h_st_data_<n>_i[255:0] x4 (128-bit): d2h_st_data_<n>_1[127:0]	Input	D2H Streaming data from device to host.
d2h_st_ready_<n>_o	Output	Backpressure from Multi Channel DMA IP for PCI Express. A '1' represents, IP readiness for receiving data.
x16: d2h_st_empty_<n>_i[5:0] x8/x4 (256-bit): d2h_st_empty_<n>_i[4:0] x4 (128-bit): d2d_st_empty_<n>_i[3:0]	Input	Represents the number of empty bytes in d2h_st_data_<n>_i, and valid only when both d2h_st_valid_<n>_i and d2h_st_eop_<n>_i is '1'.
d2h_st_sof_<n>_i	Input	Start of file (or packet) as instructed by the user logic.
d2h_st_eof_<n>_i	Input	End of file (or packet) as instructed by the user logic.
d2h_st_channel_<n>_i[10:0]	input	To support multi-Channel per port.

4.4.6. User Event MSI-X Interface

User logic requests DMA engine to send an event interrupt for a queue associated with a PF/VF.

Table 43. User MSI-X Interface

Interface Clock Domain for H-Tile: coreclkout_hip

Interface Clock Domain for P-Tile, F-Tile and R-Tile: app_clk

Signal Name	I/O	Description
usr_event_msix_valid_i	Input	The valid signal qualifies valid data on any cycle with data transfer.
usr_event_msix_ready_o	Output	On interfaces supporting backpressure, the sink asserts ready to mark the cycles where transfers may take place.
usr_event_msix_data_i [15:0]	Input	{rsvd[3:0],msix_queue_dir,msix_queue_num_i[10:0]} <i>Note:</i> msix_queue_dir Queue direction. D2H = 0, H2D =1

4.4.7. User Functional Level Reset (FLR) Interface

When the DMA engine receives Functional Level Resets from the PCIe Hard IP module, 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 user logic for the reset request before it issues an acknowledgement to the PCIe Hard IP.

Table 44. User FLR Interface

Interface Clock Domain for H-Tile: coreclkout_hip

Interface Clock Domain for P-Tile, F-Tile and R-Tile: app_clk

Signal Name	I/O	Description
usr_flr_rcvd_val_o	Output	Indicates user logic to begin flr for the specified channel in <code>usr_flr_rcvd_chan_num_o</code> . asserted until <code>usr_flr_completed_i</code> input is sampled 1'b1.
usr_flr_rcvd_chan_num_o[10:0]	Output	Indicates Channel number for which flr has to be initiated by user logic.
usr_flr_completed_i	Input	One-cycle pulse from the application indicates completion of flr activity for channel in <code>usr_flr_rcvd_chan_num_o</code>

4.5. Bursting Avalon-MM Master (BAM) Interface

Table 45. BAM Signals

Signal Name	I/O Type	Description
bam_address_o[<n>:0]	Output	Represents a byte address. The value of address must align to the data width. <n>: {vfactive+ $\lceil \log_2(\text{PF_NUM}) \rceil + \lceil \log_2(\text{VF_NUM}) \rceil + 3 + \text{BAR_addr_width} \} - 1$, where vfactive=1, PF_NUM=number of PFs enabled, VF_NUM=number of VFs enabled, 3=bar_num width, BAR_addr_width= 22 bits (H-Tile) / max(BAR_addr_width) (P-Tile and F-Tile and R-Tile)
x16: bam_byteenable_o[63:0] x8/x4 (256-bit): bam_byteenable_o[31:0] x4 (128-bit): bam_byteenable_o[15:0]	Output	Enables one or more specify the valid bytes of write data during transfer on interfaces. For x16, each <code>bas_byteenable</code> bit correspond to a byte in <code>bam_writedata_o[511:0]</code> . For single-cycle read bursts and for all write bursts, all contiguous sets of enabled bytes are supported. For multi-cycle read bursts, all bits of <code>bam_byteenable_o[63:0]</code> are asserted, regardless of the First Byte Enable (BE) and Last BE fields of the corresponding TLP.
x16: bam_burstcount_o[3:0]	Output	Used by a bursting master to indicate the number of transfers in each burst.

continued...

Signal Name	I/O Type	Description
x8/x4(256-bit): bam_burstcount_o[4:0] x4 (128-bit): bam_burstcount_o[5:0]		
bam_read_o	Output	Asserted to indicate a read transfer.
x16: bam_readdata_i[511:0] x8/x4(256-bit): bam_readdata_i[255:0] x4 (128-bit): bam_readdata_i[127:0]	Input	Read data from the user logic in response to a read transfer
bam_readdatavalid_i	Input	When asserted, indicates that the bam_readdata signal contains valid data in response to a previous read request. For a read burst with burstcount value <n>, the readdatavalid signal must be asserted <n> times, once for each readdata item.
bam_write_o	Output	Asserted to indicate a write transfer
x16: bam_writedata_o[511:0] x8/x4 (256-bit): bam_writedata_o[255:0] x4 (128-bit): bam_writedata_o[127:0]	Output	Data for write transfers
bam_waitrequest_i	Input	When asserted, indicates that the Avalon-MM slave is not ready to respond to a request. WaitRequestAllowance for BAM AVMM Master Interface : <ul style="list-style-type: none"> • 512-bit data-width is 16 • 256-bit data-width is 32 • 128-bit data-width is 64

4.6. Bursting Avalon-MM Slave (BAS) Interface

Table 46. BAS Signals

Signal Name	I/O Type	Description
bas_vfactive_i	Input	When asserted, this signal indicates AVMM transaction is targeting a virtual function
H-Tile: bas_pfnum_i[1:0] P-Tile and F-Tile and R-Tile: bas_pfnum_i[2:0]	Input	Specifies a target PF number
bas_vfnum_i[10:0]	Input	Specifies a target VF number
H-Tile: bas_address_i [63:0] P-Tile, F-Tile and R-Tile : bas_address_i[<n>-1:0]	Input	Specify the byte address regardless of the data width of the master. The Bursting Slave's addresses must be aligned to the width of the data bus. For example, if the data width is 64B, the addresses must align to 64B.
continued...		

Signal Name	I/O Type	Description
		In P/F/R-Tile End Point mode the BAS address width <n> is always 64. In P/F/R-Tile Root Port mode, if ATT is enabled then <n> : (ATT Table Address Width) + (ATT Window Address Width), else <n> : 64
x16: bas_byteenable_i[63:0] x8/x4 (256-bit): bas_byteenable_i[31:0] x4 (128-bit): bas_byteenable_i[15:0]	Input	Enables one or more specific bytes of write data during transfers on interfaces. For x16, each bas_byteenable bit corresponds to a byte in bas_writedata_i[511:0]. For single-cycle read bursts and for all write bursts, all contiguous sets of enabled bytes are supported. For burst read transactions, the bas_byteenable_i[63:0] must be 64'hFFFF_FFFF_FFFF_FF
x16: bas_burstcount_i[3:0] x8/x4 (256-bit): bas_burstcount_i[4:0] x4 (128-bit): bas_burstcount_i[5:0]	Input	Used by a bursting master to indicate the number of transfers in each burst.
bas_read_i	Input	Asserted to indicate a read transfer.
x16: bas_readdata_o[511:0] x8/x4 (256-bit): bas_readdata_o[255:0] x4 (128-bit): bas_readdata_o[127:0]	Output	Read data to the user logic in response to a read transfer
bas_readdatavalid_o	Output	When asserted, indicates that the readdata signal contains valid data. For a read burst with burstcount value <n>, the readdatavalid signal must be asserted <n> times, once for each readdata item.
bas_write_i	Input	Asserted to indicate a write transfer
continued...		

Signal Name	I/O Type	Description
x16: bas_writedata_i[511:0] x8/x4(256-bit): bas_writedata_i[255:0] x4 (128-bit): bas_writedata_i[127:0]	Input	Data for write transfers
bas_waitrequest_o	Output	When asserted, indicates that the Avalon-MM slave is not ready to respond to a request. waitrequestAllowance = 0 The master cannot issue any transfer after bas_waitrequest_o is asserted.
bas_response_o[1:0]	Output	Carries the response status: <ul style="list-style-type: none"> 00: OKAY. Successful response for a transaction. 01: RESERVED. Encoding is reserved. 10: SLAVEERROR. Error from an endpoint agent. Indicates an unsuccessful transaction. 11: DECODEERROR. Indicates attempted access to an undefined location.

4.7. Legacy Interrupt Interface

Legacy interrupts mimic the original PCI level-sensitive interrupts using virtual wire messages. The MCDMA in Endpoint mode signals legacy interrupts on the PCIe link using Message TLPs. The term INTx refers collectively to the four legacy interrupts INTA#, INTB#, INTC# and INTD#. You can assert app_int_i to cause an Assert_INTx Message TLP to be generated and sent upstream. A deassertion of app_int_i, i.e a transition of this signal from high to low, causes a Deassert_INTx Message TLP to be generated and sent upstream.

To use legacy interrupts, you must clear the Interrupt Disable bit, which is bit 10 of the Command Register in the configuration header. Then, you must turn off the MSI Enable bit.

This interface is available when the **Enable Legacy Interrupt** checkbox is enabled under the **PCIe Settings > PCIe Avalon Settings** tab in the IP Parameter Editor.

Note: The Root Port mode supports only the legacy interrupt via the p#_irq_status_o signal.

Table 47. Legacy Interrupt Interface Signals

Signal Name	I/O Type	Description
p#_app_int_i[7:0]	Input	When asserted, these signals indicate an assertion of an INTx message is requested. A transition from high to low indicates a deassertion of the INTx message is requested. Each bit is associated with a corresponding physical function. Only available in EP mode.
<i>continued...</i>		

Signal Name	I/O Type	Description
		<i>Note:</i> Legacy interrupts are currently not supported by the MCDMA IP. Tie these unused inputs to 0.
p#_app_int_ready_o[7:0]	Output	One bit per physical function. The app_int_i value should be held until app_int_ready_o=1. Only available in EP mode.
p#_irq_status_o	Output	These signals drive legacy interrupts to the Application Layer in Root Port mode. The source of the interrupt is logged in the Root Port Interrupt Status registers (x4, x8: rp_irq_status, x16: root_port_irq_status) in the Port Configuration and Status registers. Only available in RP mode.

4.8. Hot Plug Interface (RP only)

Note: The Hot Plug Interface is supported only in the R-Tile MCDMA RP mode.

Hot Plug support means that the device can be added to or removed from a system during runtime. The Hot Plug Interface in the R-Tile MCDMA IP enables an Altera FPGA with this IP to safely provide this capability. To implement hot plug in an R-tile RP, you must enable the parameter **Enable Hot Plug** under the **PCIe Avalon Settings** tab of the IP Parameter Editor. This enables the Hot Plug Interface within the IP.

The interface displays a list of signals reported by the on-board hot plug components in the Downstream Port. This interface is available only if the Slot Status Register of the PCI Express Capability Structure is enabled. Refer to the Slot Status Register of the PCI Express Capability Structure for additional information. You must implement the required capabilities and user logic to drive this interface and monitor the pX_irq_status_o signal. The pX_irq_status_o signal is available when the **Enable Legacy Interrupt** checkbox is enabled under the **PCIe Avalon Settings** tab of the IP Parameter Editor.

Table 48. Hot Plug Interface Signals

Signal Name	I/O Type	Description
pX_sys_attn_button_pressed_i	Input	Attention Button Pressed. Indicates that the system attention button was pressed, and sets the Attention Button Pressed bit in the Slot Status Register.
pX_sys_pwr_fault_det_i	Input	Power Fault Detected. Indicates the power controller detected a power fault at this slot.
pX_sys_cmd_cpled_int_i	Input	Command Completed Interrupt. Indicates that the Hot Plug controller completed a command.
<i>continued...</i>		

Signal Name	I/O Type	Description
pX_sys_pre_det_state_i	Input	Indicates whether or not a card is present in the slot. <ul style="list-style-type: none"> 0: slot is empty. 1: card is present in the slot.
pX_sys_mrl_sensor_state_i	Input	MRL Sensor State. Indicates the state of the manually operated retention latch (MRL) sensor. <ul style="list-style-type: none"> 0: MRL is closed. 1: MRL is open.
pX_sys_eml_interlock_engaged_i	Input	Indicates whether the system electromechanical interlock is engaged, and controls the state of the electromechanical interlock status bit in the Slot Status Register.

4.9. MSI Interface

Note: MSI Interface is supported only in EP mode.

Table 49. MSI Interface Signals

Signal Name	I/O Type	Description
msi_req_i	Input	MSI request signal. Assertion causes Mem Write TLP to be generated on the AVST interface to the HIP based on the MSI Capability register values and other MSI input ports. You can deassert MSI request any time after Ack (msi_ack_o) has been asserted.
msi_func_num_i[2:0]	Input	Specifies the function number requesting an MSI transmission
msi_num_i[4:0]	Input	MSI number that indicates the offset between the base message data and the MSI to send. When multiple message mode is enabled, this signal sets the lower five bits of the MSI Data register. For more information, refer to PCIe base specification.
msi_ack_o	Output	Ack for MSI request. Asserted for 1 clock cycle. Your logic can deassert the MSI request as soon as this signal is asserted.
msi_status_o[1:0]	Output	Indicates the execution status of requested MSI. Valid when msi_ack_o is asserted. <ul style="list-style-type: none"> 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

4.10. Config Slave Interface (RP only)

Table 50. Config Slave Interface Signals

Signal Name	I/O Type	Description
cs_address_i[13:0]	Input	Represents a byte address. The value of address must align to the data width.
cs_byteenable_i[3:0]	Input	Enables one or more specific byte lanes during transfers on interfaces
cs_read_i	Input	Asserted to indicate a read transfer.
cs_readdata_o[31:0]	Output	Read data to the user logic in response to a read transfer
cs_readdatavalid_o	Output	When asserted, indicates that the readdata signal contains valid data.
cs_write_i	Input	Asserted to indicate a write transfer
cs_writedata_i[31:0]	Input	Data for write transfers
cs_writeresponse_valid_o	Output	Write responses for write commands. When asserted, the value on the response signal is a valid write response.
cs_waitrequest_o	Output	When asserted, indicates that the Avalon-MM slave is not ready to respond to a request.
cs_response_o[1:0]	Output	Carries the response status: 00: OKAY—Successful response for a transaction. 01: RESERVED—Encoding is reserved. 10: SLAVEERROR—Error from an endpoint agent. Indicates an unsuccessful transaction. 11: DECODEERROR—Indicates attempted access to an undefined location.

4.11. Hard IP Reconfiguration Interface

Table 51. Hard IP Reconfiguration Interface

Signal Name	I/O	Description
usr_hip_reconfig_clk_o	Output	Use this clock to drive the usr_hip_reconfig interface.
usr_hip_reconfig_rst_n_o	Output	Active low Reset w.r.t usr_hip_reconfig_clk_o.
usr_hip_reconfig_readdata_o[7:0]	Output	read data out
usr_hip_reconfig_readdatavalid_o	Output	When asserted, the data on hip_reconfig_readdata[7:0] is valid.
usr_hip_reconfig_write_i	Input	Write enable
usr_hip_reconfig_read_i	Input	Read enable
continued...		

Signal Name	I/O	Description
H-Tile, P-Tile and F-Tile: usr_hip_reconfig_address_i[20:0] R-Tile: usr_hip_reconfig_address_i[31:0]	Input	Reconfig register address
usr_hip_reconfig_writedata_i[7:0]	Input	Write data
usr_hip_reconfig_waitrequest_o	Output	When asserted, this signal indicates that the IP core is not ready to respond to a request.

4.12. Config TL Interface

Table 52. Config TL Interface Signals

Signal Name	I/O Type	Description
H-Tile: usr_hip_tl_config_func_o [1:0] P-Tile and F-Tile: usr_hip_tl_config_func_o [2:0]	Output	Specifies the function whose Configuration Space register values are being driven out on t1_cfg_ctl_o bus.
H-Tile: usr_hip_tl_config_add_o[3:0] P-Tile and F-Tile: usr_hip_tl_config_add_o[4:0]	Output	This address bus contains the index indicating which Configuration Space register information is being driven onto the t1_cfg_ctl_o bus. For detailed information for Config Space registers, refer to the Configuration Output Interface of the P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide or Configuration Output Interface of the F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide
H-Tile: usr_hip_tl_config_ctl_o[31:0] P-Tile and F-Tile: usr_hip_tl_config_ctl_o[15:0]	Output	Multiplexed data output from the register specified by t1_cfg_add_o[4:0]. For detailed information for Config Space registers, refer to Configuration Output Interface of the P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide or Configuration Output Interface of the F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide

4.13. Configuration Intercept Interface (EP Only)

For detailed information about this interface, refer to

- *P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide [Configuration Intercept Interface (EP Only)]*
- *F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide [Configuration Intercept Interface (EP Only)]*
- *R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide [Configuration Intercept Interface (EP Only)]*

Related Information

- [P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)
- [F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)
- [R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)

4.14. Data Mover Interface

4.14.1. H2D Data Mover Interface

Table 53. H2D Data Mover Descriptor Completion Interface (h2ddm_desc_cmpl) Signals

Signal Name	I/O	Description
h2ddm_desc_cmpl_ready_i	Input	Indicates external descriptor controller is ready to accept data.
h2ddm_desc_cmpl_valid_o	Output	Qualifies all output signals
x16: h2ddm_desc_cmpl_data_o[511:0] x8/x4 (256-bit): h2ddm_desc_cmpl_data_o[255:0] x4 (128-bit): h2ddm_desc_cmpl_data_o[127:0]	Output	H2D Data Mover descriptor completion data
x16: h2ddm_desc_cmpl_empty_o[5:0] x8/x4 (256-bit): h2ddm_desc_cmpl_empty_o[4:0] x4 (128-bit): h2ddm_desc_cmpl_empty_o[3:0]	Output	Specifies number of bytes that are empty during the cycle when h2ddm_desc_cmpl_eop_o is asserted. These signals are not valid when the h2ddm_desc_cmpl_eop_o is not asserted.
h2ddm_desc_cmpl_sop_o	Output	Signals the first cycle of data transfer when used in conjunction with h2ddm_desc_cmpl_valid_o.
h2ddm_desc_cmpl_eop_o	Output	Signals the last cycle of data transfer when used in conjunction with h2ddm_desc_cmpl_valid_o.

Table 54. H2D Data Mover Descriptor Status Interface (h2ddm_desc_status) Signals

Signal Name	I/O	Description
h2ddm_desc_ready_o	Output	Indicates MCDMA IP Core is ready to accept H2D Data Mover descriptor data
h2ddm_desc_valid_i	Input	Qualifies H2D Data Mover descriptor data signals
h2ddm_desc_data_i[255:0]	Input	H2D Data Mover descriptor data

Table 55. H2D Data Mover Descriptor Status Interface (h2ddm_desc_status) Signals

Signal Name	I/O	Description
h2ddm_desc_status_valid_o	Output	Qualifies H2D Data Mover descriptor status
h2ddm_desc_status_data_o[31:0]	Output	H2D Data Mover descriptor status data

4.14.2. D2H Data Mover Interface

Table 56. D2H Data Mover Descriptor Interface (d2hdm_desc) Signals

Signal Name	I/O	Description
d2hdm_desc_ready_o	Output	Indicates MCDMA IP Core is ready to accept D2H Data Mover descriptor data
d2hdm_desc_valid_i	Input	Qualifies D2H Data Mover descriptor data signals
d2hdm_desc_data_i[255:0]	Input	D2H Data Mover descriptor data

Table 57. D2H Data Mover Descriptor Status Interface (d2hdm_desc_status) Signals

Signal Name	I/O	Description
d2hdm_desc_status_valid_o	Output	Qualifies D2H Data Mover descriptor status
d2hdm_desc_status_data_o[31:0]	Output	D2H Data Mover descriptor status data

4.15. Hard IP Status Interface

Table 58. Hard IP Status Interface Signals

Signal Name	I/O	Description
p0_link_up_o	Output	When asserted, this signal indicates the link is up.
p0_dl_up_o	Output	When asserted, this signal indicates the Data Link (DL) Layer is active.
p0_surprise_down_err_o	Output	This signal is the Surprise Down error indicator. This error event is triggered when the PHY layer reports to the Data Link Layer that the link is down.
p0_ltssm_state_o[5:0] p0_ltssm_state_delay_o[5:0] (only R-Tile)	Output	Indicates the LTSSM state: <ul style="list-style-type: none"> 6'h00: S_DETECT_QUIET 6'h01: S_DETECT_ACT 6'h02: S_POLL_ACTIVE 6'h03: S_POLL_COMPLIANCE 6'h04: S_POLL_CONFIG 6'h05: S_PRE_DETECT_QUIET 6'h06: S_DETECT_WAIT 6'h07: S_CFG_LINKWD_START 6'h08: S_CFG_LINKWD_ACCEPT 6'h09: S_CFG_LANENUM_WAIT

continued...

Signal Name	I/O	Description
		<ul style="list-style-type: none"> 6'h0A: S_CFG_LANENUM_ACCEPT 6'h0B: S_CFG_COMPLETE 6'h0C: S_CFG_IDLE 6'h0D: S_RCVRY_LOCK 6'h0E: S_RCVRY_SPEED 6'h0F: S_RCVRY_RCVRCFG 6'h10: S_RCVRY_IDLE 6'h11: S_L0 6'h12: S_L0S 6'h13: S_L123_SEND_IDLE 6'h14: S_L1_IDLE 6'h15: S_L2_IDLE 6'h16: S_L2_WAKE 6'h17: S_DISABLED_ENTRY 6'h18: S_DISABLED_IDLE 6'h19: S_DISABLED 6'h1A: S_LPBK_ENTRY 6'h1B: S_LPBK_ACTIVE 6'h1C: S_LPBK_EXIT 6'h1D: S_LPBK_EXIT_TIMEOUT 6'h1E: S_HOT_RESET_ENTRY 6'h1F: S_HOT_RESET 6'h20: S_RCVRY_EQ0 6'h21: S_RCVRY_EQ1 6'h22: S_RCVRY_EQ2 6'h23: S_RCVRY_EQ3"
p0_ltssm_st_hipfifo_ovrflw_o (only R-Tile)	Output	PCIe Hard IP FIFO storing ltssm_state changes is full. State changes may have been dropped prior to the current ltssm_state value change.

4.16. Precision Time Management (PTM) Interface

The following considerations must be taken into account when implementing the PTM feature with R-Tile MCDMA IP:

- PTM interface is only supported for Endpoint in Port 0 and 1.
- PTM accuracy in the common clock scheme is +/-50 ns.
- PTM accuracy in the separate clock scheme is +/-100 ns.

Table 59. PTM Interface Signals

Signal Name	I/O	Description
ptm_context_valid_o	Output	When this signal is asserted, it indicates that the value present on the ptm_time bus is valid. Hardware will deassert this bit whenever a PTM dialogue is requested and an update is in progress.
ptm_clk_updated_o	Output	This one clock pulse is an indication that the PTM dialogue has completed and the results of that operation have been driven on the ptm_time bus.
ptm_local_clock_o[63:0]	Output	This bus contains the calculated master time at t1' as indicated in the PCIe spec plus any latency to do the calculation and to drive the value to the requester.
ptm_manual_update_i	Input	Asserted high for one coreclkout_hip clock when the user application wants to request a PTM handshake to get a snapshot of the latest time. For detailed information about topic refer to <i>R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide</i> .

5. Parameters (H-Tile)

This chapter provides a reference for all the H-Tile parameters of the Multi Channel DMA IP for PCI Express.

Table 60. Design Environment Parameter

Starting in Quartus Prime 18.0, there is a new parameter **Design Environment** in the parameters editor window.

Parameter	Value	Description
Design Environment	Standalone System	Identifies the environment that the IP is in. <ul style="list-style-type: none"> The Standalone environment refers to the IP being in a standalone state where all its interfaces are exported. The System environment refers to the IP being instantiated in a Platform Designer system.

5.1. IP Settings

5.1.1. System Settings

Figure 23. Multi Channel DMA IP for PCI Express Parameter Editor

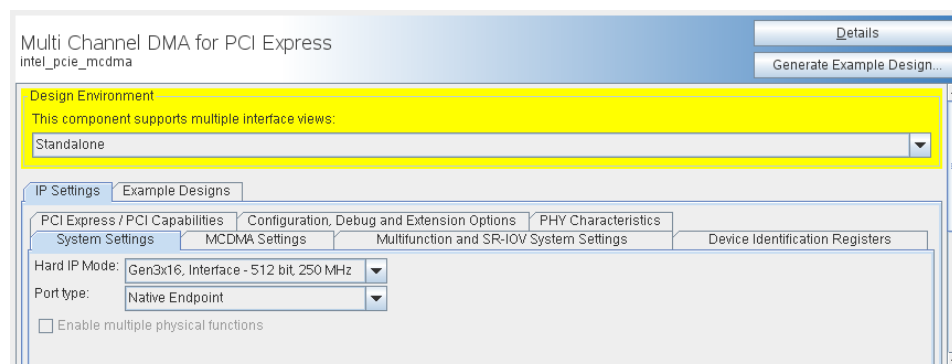


Table 61. System Settings

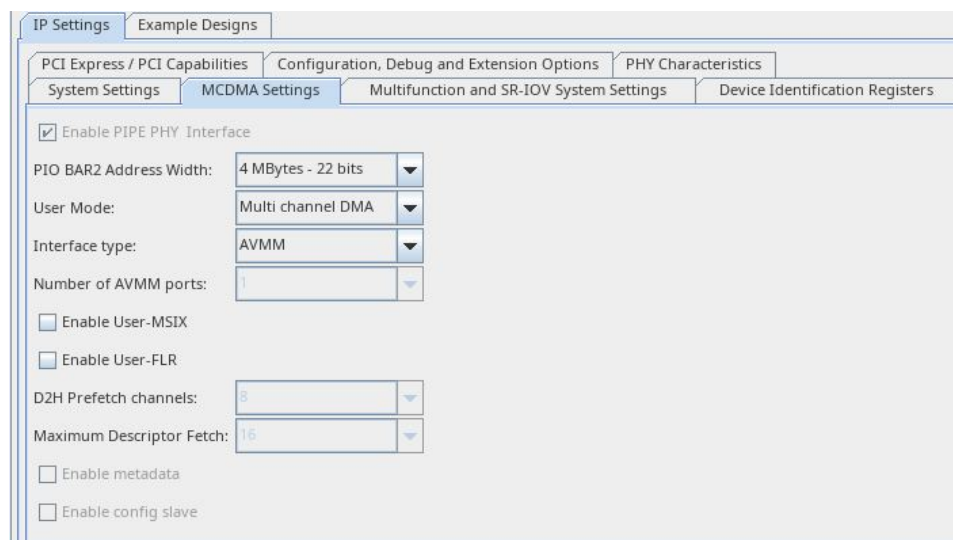
Parameter	Value	Description
Hard IP mode	Gen3x16, Interface - 512-bit, 250 MHz Gen3x8, Interface - 256 bit, 250 MHz	Selects the following elements: <ul style="list-style-type: none"> The lane data rate. Gen3 is supported The Application Layer interface frequency The width of the data interface between the hard IP Transaction Layer and the Application Layer implemented in the FPGA fabric.
Port type	Native Endpoint	Specifies the port type.

continued...

Parameter	Value	Description
	Root Port	
Enable multiple physical functions	On / Off	This parameter is not user configurable. This is automatically turned on when you select Total physical functions (PFs) greater than 1 in Multifunction and SR-IOV System Settings tab.

5.1.2. MCDMA Settings

Figure 24. MCDMA Settings Parameters



The screenshot shows the 'MCDMA Settings' tab in the Altera IDE. The settings are as follows:

- ☒ Enable PIPE PHY Interface
- PIO BAR2 Address Width: 4 MBytes - 22 bits
- User Mode: Multi channel DMA
- Interface type: AVMM
- Number of AVMM ports: 1
- ☐ Enable User-MSIX
- ☐ Enable User-FLR
- D2H Prefetch channels: 8
- Maximum Descriptor Fetch: 16
- ☐ Enable metadata
- ☐ Enable config slave

Table 62. MCDMA Settings

Parameter	Value	Description
PIO BAR2 Address Width	NA 128 Bytes - 7 bits ~ 8 EBytes - 63 bits	Address width for PIO AVMM port. Default address width is 22 bits
User Mode	Multi channel DMA Bursting Master Bursting Slave BAM+BAS BAM+MCDMA BAM+BAS+MCDMA Data Mover Only	This option allows you to configure the mode of operation for MCDMA IP. MCDMA mode has the DMA functionality. BAM and BAS offer Bursting Master and Slave AVMM capabilities without the DMA functionality.
Interface type	AVMM AVST	User logic interface type for D2HDM and H2DDM. Default: Avalon-MM Interface
Number of ports	1	Number of ports for AVMM and AVST interface is 1.
Enable User-MSIX	On / Off	User MSI-X enables user application to initiate interrupts through MCDMA, this option is available only if the user selects MCDMA mode
continued...		

Parameter	Value	Description
Enable User-FLR	On / Off	User FLR interface allows passing of FLR signals to the user side application
D2H Prefetch channels	8 16 32 64 128 256	Sets the D2H Prefetch channels. Applicable to AVST 1 port interface only. In the current release, the D2H Prefetch Channels parameter follows the total number of DMA channels that you select in the IP Parameter Editor up to 256 total channels. When the total number of channels selected is greater than 256, then D2H Prefetch channels are fixed to 64.
Maximum Descriptor Fetch	16 32 64	Sets the maximum descriptors that are fetched per D2H prefetch channel. Applicable to AVST 1 port interface only.
Enable Metadata	On / Off	Enable Metadata. Applicable to AVST 1 port interface only.
Enable config slave	On / Off	This parameter is not user configurable. This is turned on automatically when a Root Port mode is selected. Not applicable to Endpoint mode.
Enable MSI Capability Value	On / Off	Enables or disables MSI capability for BAS. Note: This parameter is only available when User Mode is set to BAS or BAM+BAS in Endpoint mode. Not supported in Root Port mode.
Number of MSI Messages Requested	1 2 4 8 16 32	Sets the number of messages that the application can request in the multiple message capable field of the Message Control register.
Enable MSI 64-bit addressing Value	On / Off	Enables or disables 64-bit MSI addressing.
Enable MSI Extended Data Capability	On / Off	Enables or disables MSI extended data capability.
Enable address byte aligned transfer	On / Off	This option allows you to enable the Byte aligned address mode support needed for Kernel or DPDK drivers and DMA makes no assumption on the alignment of data with respect to the address. <i>Note:</i> This parameter is only available when the Interface type is set to AVST.

5.1.3. Device Identification Registers

The following table lists the default values of the read-only registers in the PCI* Configuration Header Space. You can use the parameter editor to set the values of these registers.

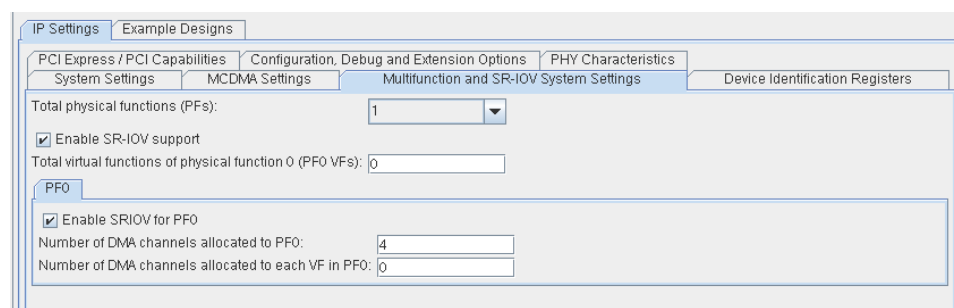
You can specify Device ID registers for each Physical Function.

Table 63. PCIe0 Device Identification Registers

Parameter	Value	Description
Vendor ID	0x00001172	Sets the read-only value of the Vendor ID register. This parameter cannot be set to 0xFFFF per the PCI Express Base Specification. Address offset: 0x000.
Device ID	0x00000000	Sets the read-only value of the Device ID register. Address offset: 0x000.
Revision ID	0x00000001	Sets the read-only value of the Revision ID register. Address offset: 0x008.
Class Code	0x00ff0000	Sets the read-only value of the Class Code register. You must set this register to a non-zero value to ensure correct operation. Address offset: 0x008.
Subsystem Vendor ID	0x00000000	Address offset: 0x02C. Sets the read-only value of Subsystem Vendor ID register in the PCI Type 0 Configuration Space. This parameter cannot be set to 0xFFFF per the PCI Express Base Specification. This value is assigned by PCI-SIG to the device manufacturer. This value is only used in Root Port variants.
Subsystem Device ID	0x00000000	Sets the read-only value of the Subsystem Device ID register in the PCI Type 0 Configuration Space. This value is only used in Root Port variants. Address offset: 0x02C

5.1.4. Multifunction and SR-IOV System Settings Parameters [Endpoint Mode]

Figure 25. Multifunction and SR-IOV System Settings Parameters



IP Settings Example Designs

PCI Express / PCI Capabilities Configuration, Debug and Extension Options PHY Characteristics

System Settings MCDMA Settings Multifunction and SR-IOV System Settings Device Identification Registers

Total physical functions (PFs): 1

☒ Enable SR-IOV support

Total virtual functions of physical function 0 (PF0 VFs): 0

PF0

☒ Enable SRIOV for PF0

Number of DMA channels allocated to PF0: 4

Number of DMA channels allocated to each VF in PF0: 0

Table 64. PCIe0 Multifunction and SR-IOV System settings

Parameter	Value	Description
Total physical functions (PFs)	1-4	Sets the number of physical functions
Enable SR-IOV support	On / Off	Enable SR-IOV support
Total virtual functions of physical function (PF VFs)	0 (default)	Set the number of VFs to be assigned to each Physical Function
Enable SRIOV for PF0	On / Off	Enable SR-IOV support on Physical Function
Number of DMA channels allocated to PF0	0 - 512	Number of DMA Channels between the host and device PF Avalon-ST / Avalon-MM ports.
Number of DMA channels allocated to each VF in PF0	0 - 512	When SRIOV support is turned on for the PF, this parameter sets the number of DMA channels allocated to each VF in the PF <i>Note:</i> This parameter is active when 'Enable SR-IOV support' is set to ON and 'Enable SRIOV for PF' is also set to ON.

5.1.5. Configuration, Debug and Extension Options

Table 65. PCIe0 Configuration, Debug and Extension Options

Parameter	Value	Description
Enable HIP dynamic reconfiguration of PCIe read-only registers	On / Off	When on, creates an Avalon-MM slave interface that software can drive to update global configuration registers which are read-only at run time.
Enable transceiver dynamic reconfiguration	On / Off	When on, creates an Avalon-MM slave interface that software can drive to update Transceiver reconfiguration registers
Enable Native PHY, LCPLL, and fPLL ADME for Toolkit	On / Off	When on, Native PHY and ATXPLL and fPLL ADME are enabled for Transceiver Toolkit. Must enable transceiver dynamic reconfiguration before enabling ADME
Enable PCIe Link Inspector	On / Off	When on, PCIe link inspector is enabled. Must enable HIP dynamic reconfiguration, transceiver dynamic reconfiguration and ADME for Toolkit to use PCIe link inspector
Enable PCIe Link Inspector AVMM Interface	On / Off	When on, PCIe link inspector AVMM interface is exported. When on, JTAG to Avalon Bridge IP instantiation is included in the Example Design generation for debug

5.1.6. PHY Characteristics

Table 66. PHY Characteristics

Parameter	Value	Description
Gen2 TX de-emphasis	3.5dB 6dB	Specifies the transmit de-emphasis for Gen2. Intel recommends the following settings: <ul style="list-style-type: none"> 3.5 dB: Short PCB traces 6.0 dB: Long PCB traces.
VCCR/VCCT supply voltage for the transceiver	1_1V 1_0V	Allows you to report the voltage supplied by the board for the transceivers.

5.1.7. PCI Express / PCI Capabilities Parameters

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.

5.1.7.1. Device

Figure 26. Device Table

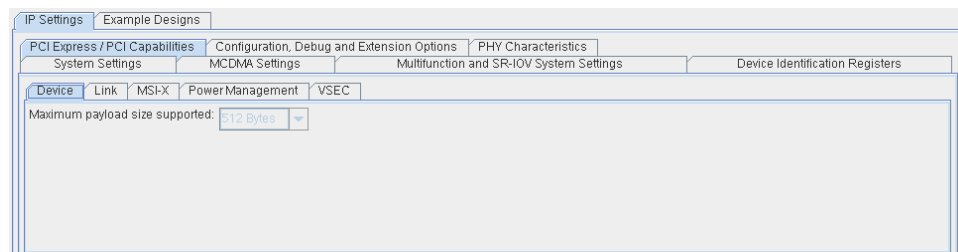


Table 67. Device

Parameter	Possible Values	Default Value	Address	Description
Maximum payload sizes supported	512 bytes <i>Note: Value is fixed at 512 bytes</i>	512 bytes	0x074	Specifies the maximum payload size supported. This parameter sets the read-only value of the max payload size supported field of the Device Capabilities register.

5.1.7.2. Link

Table 68. Link Table

Parameter	Value	Description
Link port number (Root Port only)	0x01	Sets the read-only value of the port number field in the Link Capabilities register. This parameter is for Root Ports only. It should not be changed.
Slot clock configuration	On/Off	When you turn this option On , indicates that the Endpoint uses the same physical reference clock that the system provides on the connector. When Off , the IP core uses an independent clock regardless
continued...		

Parameter	Value	Description
		of the presence of a reference clock on the connector. This parameter sets the Slot Clock Configuration bit (bit 12) in the PCI Express Link Status register.

5.1.7.3. MSI-X

Note: The MSI-X capability parameters cannot be set or modified if you select the MCDMA mode.

Note: The MSI-X capability parameters can be modified on BAS/BAM/BAM+BAS mode.

Table 69. MSI-X

Parameter	Value	Description
Enable MSI-X	On / Off	When On, adds the MSI-X capability structure, with the parameters shown below.
Table size	15	System software reads this field to determine the MSI-X table size <n>, which is encoded as <n-1>.
Table offset	0x0000000000020000	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. This field is read-only after being programmed.
Table BAR indicator	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. This field is read-only.
Pending bit array (PBA) offset	0x0000000000030000	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	Specifies the function Base Address registers, 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.
VF Table size	0	Sets the number of entries in the MSI-X table for PF0's VFs. MSI-X cannot be disabled for VF. Set to 1 to save resources.

5.1.7.4. Power Management

Table 70. Power Management Parameters

Parameter	Value	Description
Endpoint L0s acceptable latency	Maximum of 64 ns Maximum of 128 ns Maximum of 256 ns Maximum of 512 ns Maximum of 1 us Maximum of 2 us Maximum of 4 us No limit	<p>This design parameter specifies the maximum acceptable latency that the device can tolerate to exit the L0s state for any links between the device and the root complex. It sets the read-only value of the Endpoint L0s acceptable latency field of the Device Capabilities Register (0x084).</p> <p>This Endpoint does not support the L0s or L1 states. However, in a switched system there may be links connected to switches that have L0s and L1 enabled. This parameter is set to allow system configuration software to read the acceptable latencies for all devices in the system and the exit latencies for each link to determine which links can enable Active State Power Management (ASPM). This setting is disabled for Root Ports.</p> <p>The default value of this parameter is 64 ns. This is a safe setting for most designs.</p>
Endpoint L1 acceptable latency	Maximum of 1 us Maximum of 2 us Maximum of 4 us Maximum of 8 us Maximum of 16 us Maximum of 32 us Maximum of 64 ns No limit	<p>This value indicates the acceptable latency that an Endpoint can withstand in the transition from the L1 to L0 state. It is an indirect measure of the Endpoint's internal buffering. It sets the read-only value of the Endpoint L1 acceptable latency field of the Device Capabilities Register.</p> <p>This Endpoint does not support the L0s or L1 states. However, a switched system may include links connected to switches that have L0s and L1 enabled. This parameter is set to allow system configuration software to read the acceptable latencies for all devices in the system and the exit latencies for each link to determine which links can enable Active State Power Management (ASPM). This setting is disabled for Root Ports.</p> <p>The default value of this parameter is 1 μs. This is a safe setting for most designs.</p>

The Stratix 10 Avalon-ST Hard IP for PCI Express and Stratix 10 Avalon-MM Hard IP for PCI Express do not support the L1 or L2 low power states. If the link ever gets into these states, performing a reset (by asserting `pin_perst`, for example) allows the IP core to exit the low power state and the system to recover.

These IP cores also do not support the in-band beacon or sideband WAKE# signal, which are mechanisms to signal a wake-up event to the upstream device.

5.1.7.5. Vendor Specific Extended Capability (VSEC)

Table 71. VSEC

Parameter	Value	Description
User ID register from the Vendor Specific Extended Capability	Custom value	Sets the read-only value of the 16-bit User ID register from the Vendor Specific Extended Capability. This parameter is only valid for Endpoints.

5.2. Example Designs

Table 72. Example Designs

Parameter	Value	Description
Currently Selected Example Design	PIO using MQDMA Bypass mode AVMM DMA Device-side Packet Loopback Packet Generate/ Check	Select an example design available from the pulldown list. User Mode and Avalon-ST/Avalon-MM Interface type setting determines available example designs
Simulation	On/Off	When On , the generated output includes a simulation model.
Select simulation Root Complex BFM	Intel FPGA BFM Third-party BFM	Choose the appropriate BFM for simulation. Intel FPGA BFM: Default. This bus functional model (BFM) supports x16 configurations by downtraining to x8. Third-party BFM: Select this If you want to simulate all 16 lanes using a third-party BFM.
Synthesis	On/Off	When On , the generated output includes a synthesis model.
Generated HDL format	Verilog/VHDL	Only Verilog HDL is available in the current release.
Target Development Kit	None Stratix 10 GX H-Tile Production FPGA Development Kit Stratix 10 MX H-Tile Production FPGA Development Kit	Select the appropriate development board. If you select one of the development boards, system generation overwrites the device you selected with the device on that development board. <i>Note:</i> If you select None , system generation does not make any pin assignments. You must make the assignments in the .qsf file.

Note: For more information about example designs, refer to the *Multi Channel DMA Intel FPGA IP for PCI Express Design Example User Guide*.

Related Information

[Multi Channel DMA Intel FPGA IP for PCI Express Design Example User Guide](#)

6. Parameters (P-Tile) (F-Tile) (R-Tile)

This chapter provides a reference for all the R-Tile, P-Tile and F-Tile parameters of the Multi Channel DMA IP for PCI Express.

Table 73. Design Environment Parameter

Starting in Quartus Prime 18.0, there is a new parameter **Design Environment** in the parameters editor window.

Parameter	Value	Description
Design Environment	Standalone System	<p>Identifies the environment that the IP is in.</p> <ul style="list-style-type: none"> The Standalone environment refers to the IP being in a standalone state where all its interfaces are exported. The System environment refers to the IP being instantiated in a Platform Designer system.

6.1. Top-Level Settings

Figure 27. Multi Channel DMA IP for PCI Express Parameter Editor

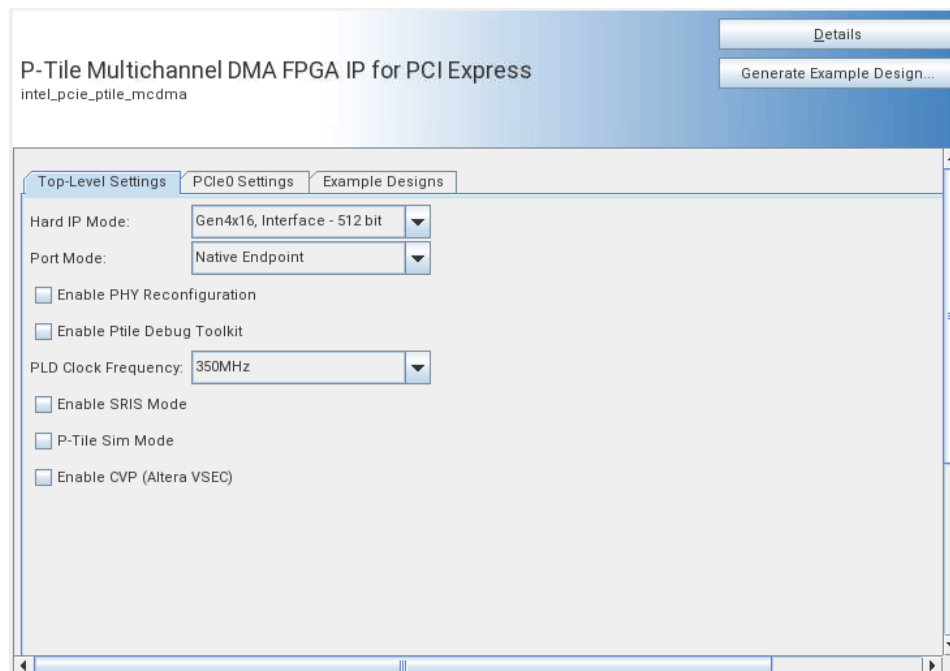


Table 74. Top-Level Settings

Parameter	Value	Default Value	Description
Hard IP mode (P-Tile)	Gen4 x16, Interface – 512 bit Gen3 x16, Interface – 512 bit Gen4 2x8, Interface – 256 bit Gen3 2x8, Interface – 256 bit Gen4 1x8, Interface – 256 bit Gen3 1x8, Interface – 256 bit Gen4 4x4, Interface – 128 bit Gen3 4x4, Interface – 128 bit	Gen4x16, Interface – 512 bit	Selects the following elements: <ul style="list-style-type: none"> Lane data rate: Gen3 and Gen4 are supported. Lane width: x16 supports both Root Port and Endpoint modes. 1 x8, 2 x8 support only Endpoint mode. 4 x4 mode only supports Root Port mode.
Hard IP mode (F-Tile)	Gen4 x16, Interface – 512 bit Gen3 x16, Interface – 512 bit Gen4 2x8, Interface – 256 bit Gen3 2x8, Interface – 256 bit Gen4 1x8, Interface – 256 bit Gen3 1x8, Interface – 256 bit Gen4 1x4, Interface – 128 bit Gen3 1x4, Interface – 128 bit Gen4 2x4, Interface – 128 bit Gen3 2x4, Interface – 128 bit Gen4 4x4, Interface – 128 bit Gen3 4x4, Interface – 128 bit	Gen4x16, Interface – 512 bit	Selects the following elements: <ul style="list-style-type: none"> Lane data rate: Gen3 and Gen4 are supported. Lane width: x16, 1 x8, and 2 x8 support both Root Port and Endpoint modes. 1 x4 and 2 x4 support both Root Port and Endpoint modes. 4 x4 supports only the Root Port mode.
Hard IP mode (R-Tile)	Gen4 x16, Interface – 512 bit Gen3 x16, Interface – 512 bit Gen5 2x8, Interface – 512 bit Gen4 2x8, Interface – 512 bit Gen3 2x8, Interface – 512 bit Gen4 2x8, Interface – 256 bit Gen3 2x8, Interface – 256 bit Gen5 4x4, Interface – 256 bit Gen4 4x4, Interface – 256 bit Gen3 4x4, Interface – 256 bit Gen4 4x4, Interface – 128 bit Gen3 4x4, Interface – 128 bit	Gen5 2x8, Interface – 512 bit	Selects the following elements: <ul style="list-style-type: none"> Link data rate: Gen5, Gen4 and Gen3 are supported. Link width: x16, x8 and x4 modes supported for both Root Port and Endpoint
Number of PCIe	1 / 2	1	Display total number of MCDMA IP cores in x8 mode.
Port Mode	Native Endpoint Root Port	Native Endpoint	Specifies the port type. Root Port mode: x16,1x8 (F-Tile only), 2x8 (R-Tile only), 2x4 (F-Tile only), 4x4 Endpoint mode: x16, 1x8, 2x8, 1x4 (F-Tile only), 4x4 (R-Tile only) <i>Note:</i> 1x8 is not supported for R-Tile <i>Note:</i> In R-Tile 4x4, Port mode has several options EP/EP/EP/EP or EP/EP/RP/RP or EP/RP/RP/RP or RP/RP/RP/RP.
Enable Ptile Debug Toolkit (P-Tile) Enable Debug Toolkit (F-Tile and R-Tile)	On / Off	Off	Enable the Debug Toolkit for JTAG-based System Console debug access.
<i>continued...</i>			

Parameter	Value	Default Value	Description
Enable PHY Reconfiguration	On / Off	Off	<p>When on, creates an Avalon-MM slave interface that software can drive to update Transceiver reconfiguration registers</p> <p>Enable the transceiver PMA registers access through a dedicated an Avalon-MM slave interface.</p> <p><i>Note:</i> In F-Tile, this option has renamed as Enable PMA registers access</p> <p><i>Note:</i> This parameter is not supported for R-Tile</p>
PLD Clock Frequency (P-Tile and F-Tile)	500 MHz 450 MHz 400 MHz 350 MHz 250 MHz 225 MHz 200 MHz 175 MHz	350 MHz (for Gen4 modes) 250 MHz (for Gen3 modes)	<p>Select the frequency of the Application clock. The options available vary depending on the setting of the Hard IP Mode parameter.</p> <p>For Gen4 modes, the available clock frequencies are 500 MHz / 450 MHz / 400 MHz / 350 MHz / 250 MHz / 225 MHz / 200 MHz / 175 MHz (for Agilex 7) and 400 MHz / 350 MHz / 200 MHz / 175 MHz (for Intel Stratix 10 DX).</p> <p>For Gen3 modes, the available clock frequency is 250 MHz (for Agilex 7 and Intel Stratix 10 DX).</p>
PLD Clock Frequency (R-Tile)	500 MHz 475 MHz 450 MHz 425 MHz 400 MHz 350 MHz 275 MHz 250 MHz	500 MHz (for Gen5 mode) 500 MHz or 300 MHz (for Gen4 mode) 250 MHz (for Gen3 mode)	<p>Selects the frequency of the Application clock. The options available vary depending on the setting of the Hard IP Mode parameter.</p> <p>For Gen5 modes, the available clock frequencies are 500 MHz / 475 MHz / 450 MHz / 425 MHz / 400 MHz</p> <p>For Gen4 modes, the available clock frequencies are 500 MHz / 475 MHz / 450 MHz / 425 MHz / 400 MHz / 300 MHz / 275 MHz / 250 MHz</p> <p>For Gen3 modes, the available clock frequency are 300 MHz / 275 MHz / 250 MHz</p>
Enable SRIS Mode	On / Off	Off	<p>Enable the Separate Reference Clock with Independent Spread Spectrum Clocking (SRIS) feature.</p> <p>When you enable this option, the Slot clock configuration option under the PCIe Settings → PCIe PCI Express/PCI Capabilities → PCIe Link tab will be automatically disabled.</p>
P-Tile Sim Mode	On / Off	Off	<p>Enabling this parameter reduces the simulation time of Hot Reset tests by 5 ms.</p> <p><i>Note:</i> Do not enable this option if you need to run synthesis.</p> <p><i>Note:</i> This parameter is not supported for R-Tile and F-Tile.</p>
Enable PIPE Mode Simulation	On / Off	Off	<p>When you set this parameter, the PIPE interface is exposed which can be used to improve the simulation time.</p> <p><i>Note:</i> This parameter is not supported by F-Tile MCDMA IP or by F-Tile MCDMA Design Examples.</p> <p><i>Note:</i> This parameter is not supported by R-Tile MCDMA IP or by R-Tile MCDMA Design Examples.</p>

continued...

Parameter	Value	Default Value	Description
			<p><i>Note:</i> When running simulations with the PIPE interface, the following macro is required: "+define +RTILE_PIPE_MODE"</p> <p>For more information regarding PIPE Mode Simulation feature and its usage, refer to R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide.</p>
Enable Independent Perst (P-Tile and F-Tile) Enable Independent GPIO Perst (R-Tile)	On / Off	Off	<p>Enable the reset of PCS and Controller in User Mode for Endpoint 2x8 mode.</p> <p>When this parameter is On, new signals <code>p<n>_cold_perst_n_i</code> and <code>p<n>_warm_perst_n_i</code> are exported to the user application for P/R-Tiles. In case of F-Tile, <code>i_gpio_perst#_n</code> is exported to the user application..</p> <p>When this parameter is Off (default), the IP internally ties off these signals instead of exporting them.</p> <p><i>Note:</i> This parameter is required for the independent reset feature, which is only supported in the x8x8 Endpoint/Endpoint mode. In F-Tile, the Hard IP Reconfiguration Interface must be enabled and <code>p0_hip_reconfig_clk</code> port must be connected to a clock source when it is using this reset signal or Enable Independent Perst option is turned on.</p> <p><i>Note:</i> For more information regarding the independent resets feature and its usage, refer to</p> <ul style="list-style-type: none"> • P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide Appendix E • R-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide • F-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide
Enable CVP (Intel VSEC)	On / Off	Off	<p>Enable support for CVP flow for single tile only</p> <p>Refer to Agilex 7 Device Configuration via Protocol (CvP) Implementation User Guide for more information</p>
Slow Clock Divider	2 & 4	4	<p>Allows you to set the <code>slow_clk</code> to be divided by 2 or 4 from the <code>coreclkout_hip</code>.</p> <p><i>Note:</i> This parameter is supported for R-Tile only.</p>

6.2. PCIe0 Settings

6.2.1. Base Address Register

Note: This tab is only available for Bursting Master, Bursting Slave, BAM+BAS, BAM+MCDMA, BAM+BAS+MCDMA and Data Mover Only user modes. This tab is not available when only MCDMA user mode is selected. Options of setting BAR0/BAR1 Type is not available if MCDMA is selected with BAM.

Table 75. Base Address Registers

Parameter	Value	Description
BAR0 Type	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory 32-bit non-prefetchable memory	If you select 64-bit prefetchable memory, 2 contiguous BARs are combined to form a 64-bit prefetchable BAR; you must set the higher numbered BAR to Disabled . Defining memory as prefetchable allows contiguous data to be fetched ahead. Prefetching memory is advantageous when the requestor may require more data from the same region than was originally requested. If you specify that a memory is prefetchable, it must have the following 2 attributes: <ul style="list-style-type: none"> • Reads do not have side effects such as changing the value of the data read. • Write merging is allowed.
BAR1 Type	Disabled 32-bit non-prefetchable memory	For a definition of prefetchable memory, refer to the BAR0 Type description.
BAR2 Type	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory 32-bit non-prefetchable memory	For a definition of prefetchable memory and a description of what happens when you select the 64-bit prefetchable memory option, refer to the BAR0 Type description.
BAR3 Type	Disabled 32-bit non-prefetchable memory	For a definition of prefetchable memory, refer to the BAR0 Type description.
BAR4 Type	Disabled 64-bit prefetchable memory 64-bit non-prefetchable memory 32-bit non-prefetchable memory	For a definition of prefetchable memory and a description of what happens when you select the 64-bit prefetchable memory option, refer to the BAR0 Type description.
BAR5 Type	Disabled 32-bit non-prefetchable memory	For a definition of prefetchable memory, refer to the BAR0 Type description.
BARn Size	128 Bytes - 16 EBytes	Specifies the size of the address space accessible to BARn when BARn is enabled. n = 0, 1, 2, 3, 4 or 5
Expansion ROM	Disabled 4 KBytes - 12 bits 8 KBytes - 13 bits 16 KBytes - 14 bits 32 KBytes - 15 bits 64 KBytes - 16 bits 128 KBytes - 17 bits 256 KBytes - 18 bits 512 KBytes - 19 bits 1 MByte - 20 bits 2 MBytes - 21 bits 4 MBytes - 22 bits 8 MBytes - 23 bits 16 MBytes - 24 bits	Specifies the size of the expansion ROM from 4 KBytes to 16 MBytes when enabled.

6.2.2. PCIe0 Configuration, Debug and Extension Options

Figure 28. Endpoint PCIe0 Configuration, Debug and Extension Options [P-Tile]

P-Tile Multichannel DMA FPGA IP for PCI Express
intel_pcie_ptile_mcdma

Details
Generate Example Design...

Top-Level Settings | **PCIe0 Settings** | PCIe1 Settings | Example Designs

PCIe0 IP Settings

PCIe0 Device Identification Registers | PCIe0 PCI Express / PCI Capabilities | MCDMA Settings

PCIe0 Avalon Settings | **PCIe0 Configuration, Debug and Extension Options**

Gen 3 Requested equalization far-end TX preset vector: 0x00000004

Gen 4 Requested equalization far-end TX preset vector: 0x00000270

☒ Enable Rx Buffer Limit Ports

Figure 29. Endpoint PCIe0 Configuration, Debug and Extension Options [F-Tile]

System: FTile_MCDMA Path: intel_pcie_ftile_mcdma_0

F-Tile Multichannel DMA FPGA IP for PCI Express
intel_pcie_ftile_mcdma

Details
Generate Example Design...

Top-Level Settings | **PCIe0 Settings** | PCIe1 Settings | Example Designs | Analog Parameters

PCIe0 Device Identification Registers | PCIe0 PCI Express / PCI Capabilities | MCDMA Settings

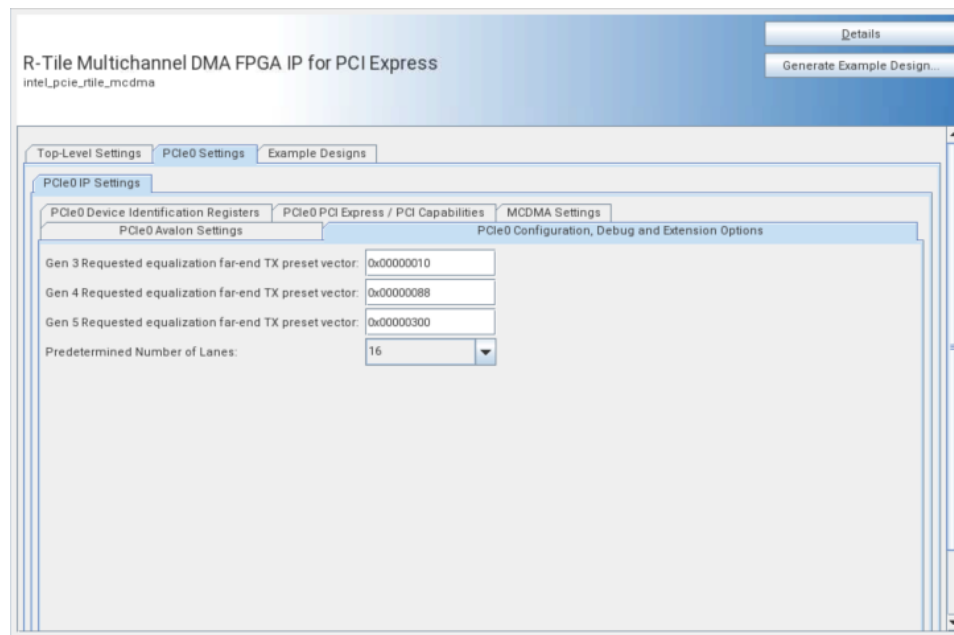
PCIe0 Avalon Settings | **PCIe0 Configuration, Debug and Extension Options**

Gen 3 requested equalization far-end TX preset vector: 0x0000030a

Gen 4 requested equalization far-end TX preset vector: 0x00000360

Predetermined number of lanes: 8

☒ Enable Rx Buffer Limit Ports

Figure 30. Endpoint PCIe0 Configuration, Debug and Extension Options [R-Tile]


R-Tile Multichannel DMA FPGA IP for PCI Express
intel_pcie_rtile_mcdma

Details
Generate Example Design...

Top-Level Settings | **PCIe0 Settings** | Example Designs

PCIe0 IP Settings

PCIe0 Device Identification Registers | PCIe0 PCI Express / PCI Capabilities | MCDMA Settings

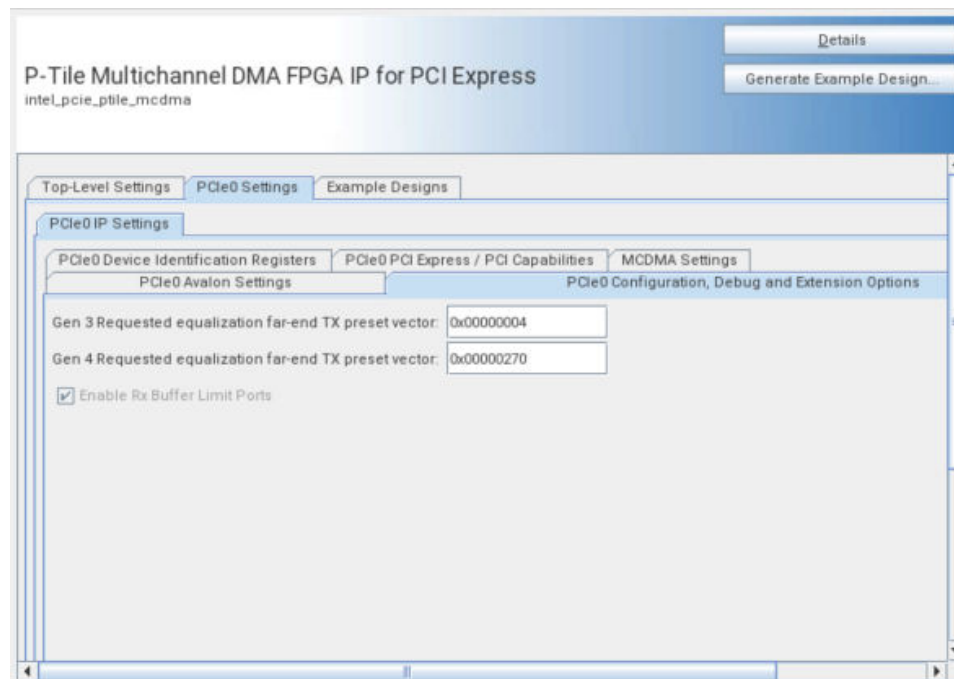
PCIe0 Avalon Settings | **PCIe0 Configuration, Debug and Extension Options**

Gen 3 Requested equalization far-end TX preset vector: 0x00000010

Gen 4 Requested equalization far-end TX preset vector: 0x00000088

Gen 5 Requested equalization far-end TX preset vector: 0x00000300

Predetermined Number of Lanes: 16

Figure 31. Rootport PCIe0 Configuration, Debug and Extension Options [P-Tile]


P-Tile Multichannel DMA FPGA IP for PCI Express
intel_pcie_ptile_mcdma

Details
Generate Example Design...

Top-Level Settings | **PCIe0 Settings** | Example Designs

PCIe0 IP Settings

PCIe0 Device Identification Registers | PCIe0 PCI Express / PCI Capabilities | MCDMA Settings

PCIe0 Avalon Settings | **PCIe0 Configuration, Debug and Extension Options**

Gen 3 Requested equalization far-end TX preset vector: 0x00000004

Gen 4 Requested equalization far-end TX preset vector: 0x00000270

☒ Enable Rx Buffer Limit Ports

Figure 32. RootPort PCIe2 Configuration, Debug and Extension Options [R-Tile]

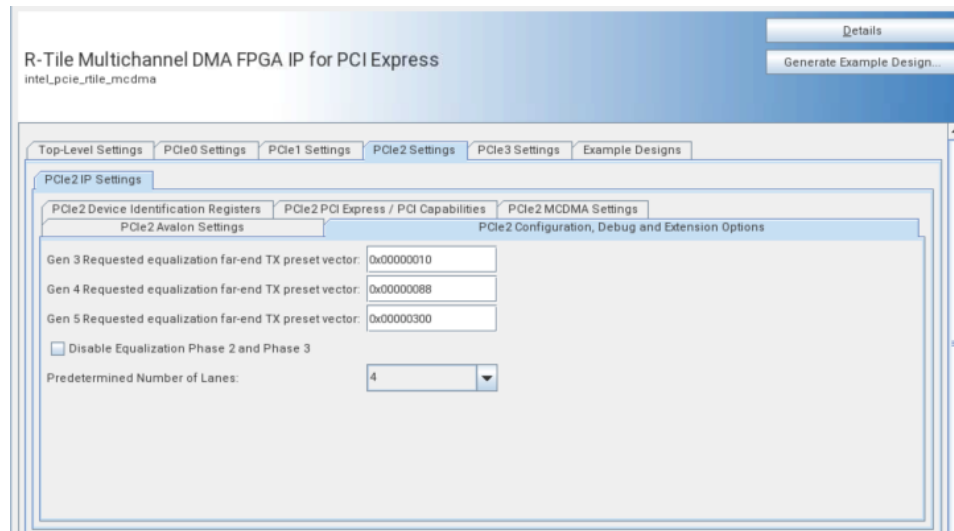


Table 76. PCIe0 Configuration, Debug and Extension Options

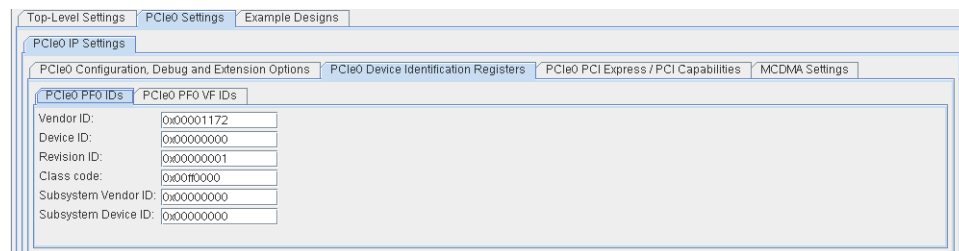
Parameter	Value	Default Value	Description
Gen 3 Requested equalization far-end TX preset vector	0 - 65535	0x00000004 (for P-Tile) 0x0000030a (for F-Tile) 0x00000200 (for R-Tile Rev. A) 0x00000010 (for R-Tile Rev. B)	Specifies the Gen 3 requested phase 2/3 far-end TX preset vector. Choosing a value different from the default is not recommended for most designs.
Gen 4 Requested equalization far-end TX preset vector	0 - 65535	0x00000270 (for P-Tile) 0x00000360 (for F-Tile) 0x00000008 (for R-Tile Rev. A) 0x00000088 (for R-Tile Rev. B)	Specifies the Gen 4 requested phase 2/3 far-end TX preset vector. Choosing a value different from the default is not recommended for most designs.
Gen5 Requested equalization far-end TX preset vector	0 - 65535	0x00000200 (for R-Tile Rev. A) 0x00000300 (for R-Tile Rev. B)	Specifies the Gen 5 requested phase 2/3 far-end TX preset vector. Choosing a value different from the default is not recommended for most designs.
Predetermined number of lanes (for R-Tile and F-Tile)	16 8 4 2 1	Maximum link width	Defines the number of lanes which are connected and good.
Enable HIP Reconfig interface	On / Off	Off	Enables HIP reconfiguration interface

continued...

Parameter	Value	Default Value	Description
			<i>Note:</i> This interface is automatically enabled in Root Port mode. Hence, the parameter is not available for user modification in Root Port mode.
Enable Prefetchable Memory 64-bit address support (Root Port mode only)	On / Off	Off	When in RP mode, indicates that the prefetchable memory range supported is 64-bit. <i>Note:</i> This feature is supported for P-Tile and F-Tile only.
Disable Equalization Phase 2 and Phase3	On / Off	Off	When this parameter is selected, there is no Equalization in Phase 2 and 3 during PCI Express link training. <i>Note:</i> This feature is only supported for R-Tile and with Root Port mode.

6.2.3. PCIe0 Device Identification Registers

Figure 33. PCIe0 Device Identification Registers



6.2.3.1. PCIe0 PF0 IDs

Table 77. PCIe0 PF0 IDs Settings

Parameter	Range	Default Value	Description
Vendor ID	16 bits	0x00001172	Sets the read-only value of the Vendor ID register. This parameter cannot be set to 0xFFFF per the PCI Express Base Specification. Address offset: 0x000.

continued...

Parameter	Range	Default Value	Description
			<i>Note:</i> Set your own Vendor ID by changing this parameter.
Device ID	16 bits	0x00000000	Sets the read-only value of the Device ID register. This register is only valid in the Type 0 (Endpoint) Configuration Space. Address offset: 0x000. <i>Note:</i> Set your own Device ID by changing this parameter.
Revision ID	8 bits	0x00000001	Sets the read-only value of the Revision ID register. Address offset: 0x008. <i>Note:</i> Set your own Revision ID by changing this parameter.
Class Code	24 bits	0x00ff0000	Sets the read-only value of the Class Code register. This parameter cannot be set to 0x0 per the PCI Express Base Specification. Address offset: 0x008. <i>Note:</i> Set your own Class Code by changing this parameter.
Subsystem Vendor ID	16 bits	0x00000000	Sets the read-only value of Subsystem Vendor ID register in the PCI Type 0 Configuration Space. This parameter cannot be set to 0xFFFF per the PCI Express Base Specification. This value is assigned by PCI-SIG to the device manufacturer. This value is only used in Root Port variants. Address offset: 0x02C <i>Note:</i> Set your own Subsystem Vendor ID by changing this parameter.
Subsystem Device ID	16 bits	0x00000000	Sets the read-only value of the Subsystem Device ID register in the PCI Type 0 Configuration Space. This value is only used in Root Port variants. Address offset: 0x02C <i>Note:</i> Set your own Subsystem Device ID by changing this parameter.

6.2.3.2. PCIe0 PF0 VF IDs

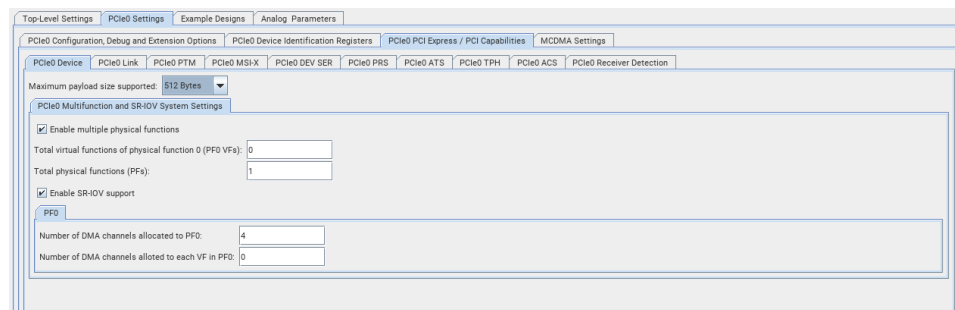
Table 78. PCIe0 PF0 VF IDs settings

Parameter	Range	Value	Description
Device ID	16 bits	0x00000000	Sets the read-only value of the Device ID register for the virtual functions
Subsystem ID	16 bits	0x00000000	Sets the read-only value of the Subsystem ID register for the virtual functions

6.2.4. PCIe0 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.

Figure 34. PCIe0 PCI Express / PCI Capabilities Parameters



6.2.4.1. PCIe0 Device

Table 79. PCIe0 Device

Parameter	Value	Default Value	Description
Maximum payload size supported	512 Bytes 256 Bytes 128 Bytes	512 Bytes	Specifies the maximum payload size supported. This parameter sets the read-only value of the max payload size supported field of the Device Capabilities registers.
Support Extended Tag Field	On / Off	On	Sets the Extended Tag Field Supported bit in Configuration Space Device Capabilities Register (P-Tile) <i>Note:</i> This parameter is not user configurable. It is set to On by the IP.
Enable multiple physical functions	On / Off	Off	Enables multiple physical functions.
Total virtual functions of physical function 0 (PF0 VFs)	0 - 2048	0	Sets the number of VFs to be assigned to physical function x. This parameter

continued...

Parameter	Value	Default Value	Description
			only appears when Enable SR-IOV support is set to 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. For example, if you set Total physical functions (PFs) to 2, Total virtual functions of physical function 0 (PF0 VFs) and Total virtual functions of physical function 1 (PF1 VFs) appear, allowing you to set the number of virtual functions for each of those physical functions.
Total physical functions (PFs)	1-8	1	Sets the number of physical functions. This parameter only appears when Enable Multiple Physical Functions is set to True .
Enable SR-IOV support	On / Off	Off	Enable SR-IOV support
Number of DMA channels allocated to PF0	0-512	4	Set the number of DMA channels allocated to the physical function
Number of DMA channels allocated to each VF in PF0	0-512	0	Set the number of DMA channels allocated to each virtual function in the physical function

6.2.4.2. PCIe0 Link

Table 80. PCIe0 Link

Parameter	Value	Default Value	Description
Link port number (Root Port only)	0-255	1	Sets the read-only value of the port number field in the Link Capabilities register. This parameter is for Root Ports only. It should not be changed.
Slot clock configuration	On/Off	On	When you turn this option On , indicates that the Endpoint uses the same physical reference clock that the system provides on the connector. When Off , the IP core uses an independent clock regardless of the presence of a reference clock on the connector. This parameter sets the Slot Clock Configuration bit (bit 12) in the PCI Express Link Status register. You cannot enable this option when the Enable SRIS Mode option is enabled.
Enable Modified TS	On/Off	On	Enables the controller to send the Modified TS OS if both sides of the link agree when the SUPPORT_MOD_TS register is set to 1.
continued...			

Parameter	Value	Default Value	Description
			<p>If the port negotiates alternate protocols or passes a Training Set Message, the bit must be set to 1.</p> <p><i>Note:</i> This feature is only supported for R-Tile.</p>

6.2.4.3. PCIe0 MSI-X

Note: The PCIe0 MSI-X feature parameters cannot be set or modified if you select the MCDMA mode.

Note: The PCIe0 MSI-X feature parameters can be modified on BAS/BAM/BAM+BAS mode.

Table 81. PCIe0 PF MSI-X

Parameter	Value	Description
Enable MSI-X	On / Off	When On, adds the MSI-X capability structure, with the parameters shown below.
Table size	15	System software reads this field to determine the MSI-X table size <n>, which is encoded as <n-1>.
Table offset	0x0000000000020000	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. This field is read-only after being programmed.
Table BAR indicator	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. This field is read-only.
Pending bit array (PBA) offset	0x0000000000030000	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	Specifies the function Base Address registers, 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.
VF Table size	0	<p>Sets the number of entries in the MSI-X table for PF0's VFs.</p> <p>MSI-X cannot be disabled for VF. Set to 1 to save resources.</p>

Table 82. PCIe0 VF MSI-X

Parameter	Value	Description
Enable VF MSI-X	On / Off	When On, adds the MSI-X capability structure to the VF, with the parameters shown below.
Table size	15	System software reads this field to determine the VF MSI-X table size <n>, which is encoded as <n-1>.
Table offset	0x0000000000002000	Points to the base of the VF 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. This field is read-only after being programmed.
Table BAR indicator	0	Specifies which one of a function's base address registers, located beginning at 0x10 in the Configuration Space, maps the VF MSI-X table into memory space. This field is read-only.
Pending bit array (PBA) offset	0x0000000000030000	Used as an offset from the address contained in one of the function's Base Address registers to point to the base of the VF 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	Specifies the function Base Address registers, located beginning at 0x10 in Configuration Space, that maps the VF MSI-X PBA into memory space. This field is read-only in the MSI-X Capability Structure.

6.2.4.4. PCIe0 DEV SER

Table 83. PCIe0 DEV SER

Parameter	Value	Default Value	Description
Enable Device Serial Number Capability	On / Off	Off	Capability Enable Device Serial Number Capability (DEV SER)
Device Serial Number (DW1)	32 bits	0x0000000000000000	Set the lower 32 bits of the IEEE 64-bit Device Serial Number (DW1)
Device Serial Number (DW2)	32 bits	0x0000000000000000	Set the upper 32 bits of the IEEE 64-bit Device Serial Number (DW2)

6.2.4.5. PCIe0 PRS

Table 84. PCIe0 PRS

Parameter	Value	Default Value	Description
PF0 Enable PRS	On / Off	Off	Enable PF0Page Request Service (PRS) capability.
PF0 Page Request Services Outstanding Capacity	32 bits	0x0000000000000000	This parameter is the upper limit on the number of pages that can be usefully allocated to the Page Request Interface (PRS).

6.2.4.6. PCIe0 PTM

Note: This capability is only available for R-Tile MCDMA IP Endpoint mode in Ports 0 and 1.

Table 85. PCIe0 PTM

Parameter	Value	Default Value	Description
Enable PTM	On / Off	Off	When you select this parameter, Precision Time Measurement (PTM) Capability is available.
Period between each automatic update of PTM context (in ms)	Disable 1 2 3 4 5 6 7 8 9 10	Disable	Determines the PTM context auto-update interval. Selecting Disable prevents PTM context auto-update.

6.2.4.7. PCIe0 ACS Capabilities

Note: The parameters in this feature are automatically set by the IP. You have no control over the selection.

6.2.4.7.1. PCIe0 ACS for Physical Functions

Table 86. PCIe0 ACS for Physical Functions

Parameter	Value	Default Value	Description
Enable Access Control Services (ACS)	On / Off	On	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected. <i>Note:</i> This feature is available if the IP is in Root Port mode, or if the IP is in Endpoint mode and the Enable multiple physical functions and/or the Enable SR-IOV support in the PCIeN Device tab is set to True .
Enable ACS P2P Traffic Support	On / Off	Off	Indicates if the component supports Peer to Peer Traffic
continued...			

Parameter	Value	Default Value	Description
			<i>Note:</i> This feature is available if the IP is in Root Port mode, or if the IP is in Endpoint mode and the Enable multiple physical functions and/or the Enable SR-IOV support in the PCIeN Device tab is set to True .

6.2.4.7.2. PCIe0 ACS for Virtual Functions

Table 87. PCIe0 ACS for Virtual Functions

Parameter	Value	Default Value	Description
Enable Access Control Services (ACS)	On / Off	Off	ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected. <i>Note:</i> This feature is available if the MCDMA IP is in Endpoint mode and the Enable SRIOV support in the PCIeN Device tab is set to True .

6.2.4.8. PCIe0 ATS

6.2.4.8.1. PCIe0 ATS for Physical Functions

Table 88. PCIe0 ATS for Physical Functions

Parameter	Value	Default Value	Description
Enable Address Translation Services (ATS)	On / Off	Off	Enable or disable Address Translation Services (ATS) capability. When ATS is enabled, senders can request, and cache translated addresses using the RP memory space for later use.

6.2.4.8.2. PCIe0 ATS for Virtual Functions

Table 89. PCIe0 ATS for Virtual Functions

Parameter	Value	Default Value	Description
Enable Address Translation Services (ATS)	On / Off	Off	Enable or disable Address Translation Services (ATS) capability. When ATS is enabled, senders can request, and cache translated addresses using the RP memory space for later use.

6.2.4.9. PCIe0 TPH

6.2.4.9.1. PCIe0 TPH for Physical Functions

Table 90. PCIe0 TPH for Physical Functions

Parameter	Value	Description
Enable TLP Processing Hints (TPH)	On / Off	Enable or disable TLP Processing Hints (TPH) capability.

Parameter	Value	Description
		Using TPH may improve the latency performance and reduce traffic congestion.

6.2.4.9.2. PCIe0 TPH for Virtual Functions

Table 91. PCIe0 TPH for Virtual Functions

Parameter	Value	Description
Enable TLP Processing Hints (TPH)	On / Off	Enable or disable TLP Processing Hints (TPH) capability. Using TPH may improve the latency performance and reduce traffic congestion.

6.2.4.10. Receiver Detection (F-Tile MCDMA IP Only)

Table 92. Receiver Detection Parameters (F-Tile MCDMA IP Only)

Parameter	Value	Default Value	Description
PF0 FORCE_DETECT_LANE enable	True/False	False	Force Detect Lane Enable When this option is set, the controller ignores receiver detection from PHY during LTSSM Detect state and uses FORCE_DETECT_LANE value
PF0 FORCE_DETECT_LANE value	0x0000ffff - 0x00000000	0x00000000	When the FORCE_DETECT_LANE_EN field is set, the controller ignores receiver detection from PHY during LTSSM Detect state and uses this value instead: <ul style="list-style-type: none"> For x4 lane: 0x0000000f For x8 lane: 0x000000ff For x16 lane: 0x0000ffff

6.2.5. MCDMA Settings

Figure 35. MCDMA Settings Parameters

R-Tile Multichannel DMA FPGA IP for PCI Express
intel_pcie_rtile_mcdma

Details
Generate Example Design...

Top-Level Settings | PCIe0 Settings | PCIe1 Settings | Example Designs

PCIe0 IP Settings

PCIe0 Device Identification Registers | PCIe0 PCI Express / PCI Capabilities | **MCDMA Settings**

PCIe0 Avalon Settings | PCIe0 Configuration, Debug and Extension Options

BAR2 Address Width: 4 Mbytes - 22 bits

User Mode: Multi channel DMA

Interface type: AVMM

Number of ports: 1

☐ Enable 32-bit Address Routing

☐ Enable 32-bit PIO

☐ Enable User-MSIX

☐ Enable User-FLR

D2H Prefetch channels: 8

Maximum Descriptor Fetch: 16

☐ Enable Metadata

☐ Enable HIP Reconfig interface

☐ Enable address byte aligned transfer

☐ Enable Configuration Intercept interface

Table 93. MCDMA Settings

Parameter	Value	Default Value	Description
BAR2 Address Width	128 Bytes - 8 Bytes	4 Mbytes – 22 bits	Address width for PIO AVMM port. Default address width is 22 bits
User Mode	Multi channel DMA Bursting Master Bursting Slave BAM+BAS BAM+MCDMA BAM+BAS+MCDMA Data Mover Only	Multi channel DMA	This option allows user to configure the mode of operation for MCDMA IP. MCDMA mode has the DMA functionality. BAM and BAS offer Bursting Master and Slave AVMM capabilities without DMA functionality <i>Note:</i> In R-Tile 4x4, Port 2 and 3 only support BAM, BAS and BAM +BAS user mode. <i>Note:</i> MCDMA R/P/F-Tile IP in x4 topology does not support Data Mover Only Mode.
Interface type	AVMM AVST	AVMM	User logic interface type for D2HDM and H2DDM. Default: Avalon-MM Interface

continued...

Parameter	Value	Default Value	Description
Number of ports	1	1	Number of ports for AVMM and AVST interface is 1.
Enable 32-bit PIO	On / Off	Off	Enables or disables the 32-bit PIO design examples. <i>Note:</i> This parameter is only available for the R-Tile MCDMA IP and F-Tile MCDMA IP.
Enable User-MSIX	On / Off	Off	User MSI-X is enables user application to initiate interrupts through MCDMA, this option is available only if the user selects MCDMA mode
Enable User-FLR	On / Off	Off	User FLR, interface allows passing of FLR signals to the user side application <i>Note:</i> This parameter is not supported for R-Tile MCDMA IP x4 Endpoint Ports 2 and 3
D2H Prefetch channels	8 16 32 64 128 256	8	Sets the D2H Prefetch channels In the current release, the D2H Prefetch Channels parameter follows the total number of DMA channels that you select in the IP Parameter Editor up to 256 total channels. When the total number of channels selected is greater than 256, then D2H Prefetch channels are automatically fixed to 64. <i>Note:</i> This parameter applicable to AVST 1 port interface only.
Maximum Descriptor Fetch	16 32 64	16	Sets the maximum descriptors that are fetched per D2H prefetch channel. <i>Note:</i> This parameter is applicable to AVST 1 port interface only.
Enable Metadata	On / Off	Off	Enables Metadata <i>Note:</i> This parameter is only available when the Interface Type is set to AVST.
Enable Configuration Intercept Interface	On / Off	Off	Select to enable configuration intercept interface.
Enable 10-bit tag support	On / Off	Off	This parameter is available for the following user modes: MCDMA, Bursting
continued...			

Parameter	Value	Default Value	Description
			Slave, BAM+BAS, BAM +MCDMA and BAM+BAS +MCDMA <i>Note:</i> This parameter is not supported for R-tile. 10-bit tag is set to On by the IP.
Enable address byte aligned transfer	On / Off	Off	This is the option to enable the Byte aligned address mode support needed for Kernel or DPDK drivers and DMA makes no assumption on the alignment of data w.r.t to address. <i>Note:</i> This parameter is only available when the Interface Type is set to AVST.
Enable MSI Capability	On / Off	Off	Enables or disables MSI capability for BAS. <i>Note:</i> This parameter is only available when User Mode is set to BAS or BAM+BAS. <i>Note:</i> This parameter is not supported for R-Tile MCDMA IP x4 Endpoint Ports 2 and 3
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.
Export pld_warm_rst_rdy and link_req_rst_n interface to top level	On / Off	Off	Exports pld_warm_rst_rdy and link_req_rst_n interface to top level. <i>Note:</i> This parameter is not available in R-Tile MCDMA IP.

Table 94. Root Port MCDMA Settings ATT Parameters

Parameter	Value	Default Value	Description
Enable ATT	On / Off	Off	Enables ATT for BAS
ATT Table Address Width	1 - 9	3	Sets depth of ATT. Depth is equal to 2 to the power of number entered.
ATT Window Address Width	10 - 63	16	Sets the number of BAS address bits to be used directly.

6.3. Example Designs

Figure 36. Example Design Settings [for x16 mode]

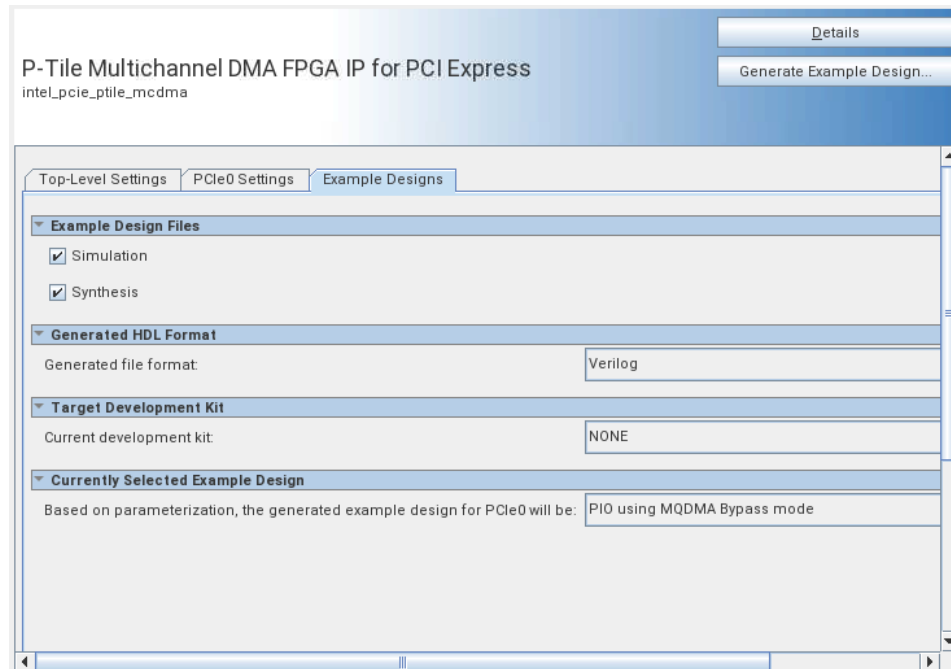


Figure 37. Example Design Settings [for 2x8 mode]

Table 95. Example Designs

Parameter	Value	Default Value	Description
Simulation	On / Off	True	When Simulation box is checked, all necessary filesets required for simulation are generated. When this box is NOT checked, filesets required for Simulation are NOT generated. Instead a Platform Designer example design system is generated. <i>Note:</i> Multiple physical functions and SR-IOV are not supported for simulation. When you generate an example design, turn off Simulation.
Synthesis	On / Off	True	When Synthesis box is checked, all necessary filesets required for synthesis are generated. When Synthesis box is NOT checked, filesets required for Synthesis are NOT

continued...

Parameter	Value	Default Value	Description
			generated. Instead a Platform Designer example design system is generated
Generated file format	Verilog	Verilog	HDL format
Current development kit	<p>None</p> <p>MCDMA P-Tile IP:</p> <ul style="list-style-type: none"> Intel Agilex 7 F-Series P-Tile ESO FPGA Development Kit Intel Agilex 7 F-Series P-Tile Production FPGA Development Kit Stratix 10 DX P-Tile Production FPGA Development Kit <p>MCDMA F-Tile IP:</p> <ul style="list-style-type: none"> None Intel Agilex 7 F-Series F-Tile FPGA Devkit DK-DEV-AGF027F1ES <p>MCDMA R-Tile IP:</p> <ul style="list-style-type: none"> None Intel Agilex 7 I-Series FPGA DevKit DK-DEV-AGI027RES Intel Agilex 7 I-Series FPGA DevKit DK-DEV-AGI027R1BES 		<p>This option provides supports for various Development Kits listed. The details of Intel FPGA Development kits can be found on Intel FPGA website.</p> <p>If this menu is grayed out, it is because no board is supported for the options selected (for example, synthesis deselected).</p> <p>If an Intel FPGA Development board is selected, the Target Device used for generation is the one that matches the device on the Development Kit</p>
Currently Selected Example Design	<p>PIO using MQDMA Bypass mode</p> <p>Device-side Packet Loopback</p> <p>Packet Generate/Check AVMM DMA Traffic Generator/Checker</p> <p>External Descriptor Controller</p>		<p>Based on MCDMA setting for "User Mode" and "Interface Type" different Example Designs are supported. List of Example design options are:</p> <p>User Mode=MCDMA, BAM +MCDMA and BAM+BAS +MCDMA* Interface Type=AVST:</p> <ul style="list-style-type: none"> PIO using MCDMA Bypass mode Device-side Packet Loopback Packet Generate/Check <p>User Mode=MCDMA, BAM +MCDMA and BAM+BAS +MCDMA*, Interface Type=AVMM:</p> <ul style="list-style-type: none"> PIO using MCDMA Bypass mode AVMM DMA <p>User Mode=Bursting Master:</p> <ul style="list-style-type: none"> PIO using MCDMA Bypass mode <p>User Mode=BAM+BAS:</p> <ul style="list-style-type: none"> PIO using MCDMA Bypass mode Traffic Generator/Checker <p>User Mode=Data Mover Only:</p>

Parameter	Value	Default Value	Description
			<ul style="list-style-type: none"> PIO using MCDMA Bypass mode External descriptor controller

Note: For more information about example designs, refer to the *Multi Channel DMA Intel FPGA IP for PCI Express Design Example User Guide*.

Related Information

[Multi Channel DMA Intel FPGA IP for PCI Express Design Example User Guide](#)

6.4. Analog Parameters (F-Tile MCDMA IP Only)

Figure 38. Analog Parameters Tab (F-Tile MCDMA IP Only)

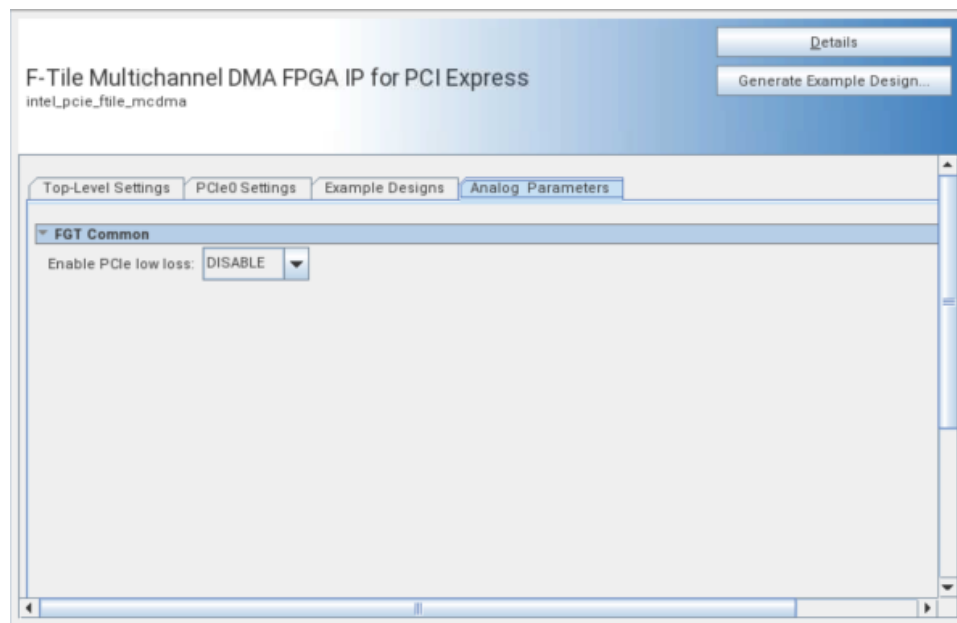


Table 96. Analog Parameters Tab (F-Tile MCDMA IP Only)

Parameter	Value	Default Value	Description
Enable PCIe low loss	DISABLE/ENABLE	DISABLE	<p>When you select ENABLE, the parameter enables the transceiver analog settings for low loss PCIe design.</p> <p>This parameter should only be enabled for chip-to-chip design where the insertion loss from endpoint silicon pad to root port silicon pad including the package insertion loss is below 8 dB at 8 GHz.</p>

6.5. PCIe1 Settings

PCIe1 Settings tab is exposed in the IP GUI when you select Gen4 2x8 or Gen3 2x8 mode. It allows you to configure a second x8 MCDMA instance. It provides the same set of parameters as PCIe0 Settings. In addition, PCIe1 Settings tab provides a special parameter that supports independent PERST use case.

6.5.1. PCIe1 Configuration, Debug and Extension Options

Figure 39. PCIe1 Configuration, Debug and Extension Options

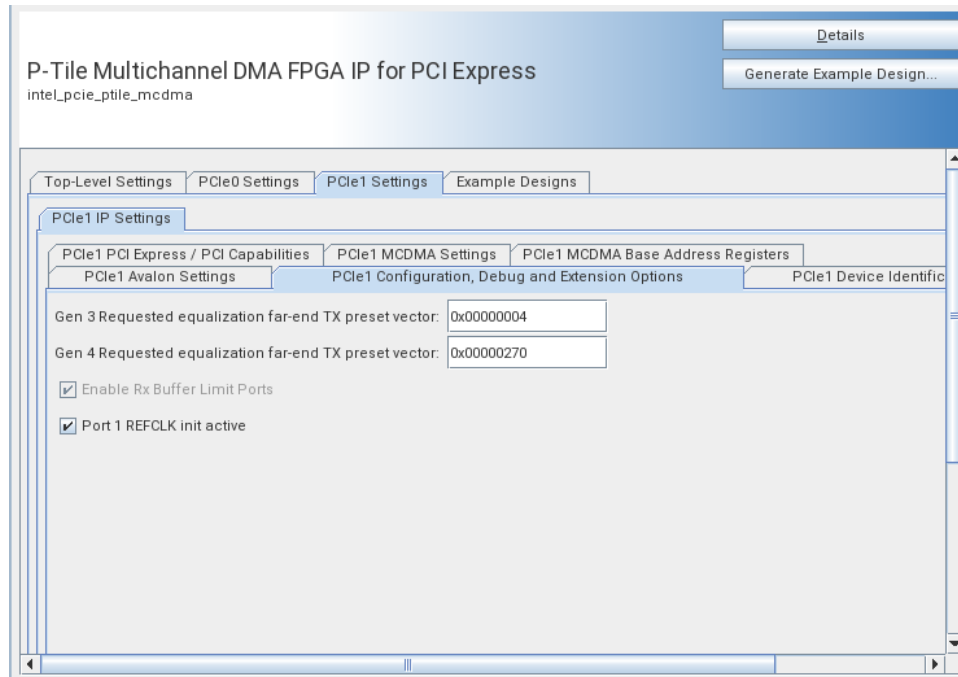


Table 97. PCIe1 Configuration, Debug and Extension Options Settings Table

Parameter	Value	Default Value	Description
Port 1 REFCLK init active (only P-Tile)	On / Off	On	<p>If this parameter is On (default), the <code>refclk1</code> is stable after <code>pin_perst</code> and is free-running. This parameter must be set to On for Type A/B/C systems.</p> <p>If this parameter is Off, <code>refclk1</code> is only available later in User Mode. This parameter must be set to Off for Type D systems.</p> <p>This parameter is only available in the PCIe1 Settings tab for a 2x8 topology.</p> <p>For more details regarding the link subdivision feature and its usage, refer to P-Tile</p>

6. Parameters (P-Tile) (F-Tile) (R-Tile)

683821 | 2025.08.04



Parameter	Value	Default Value	Description
			Avalon Streaming Intel FPGA IP for PCI Express User Guide Appendix E.

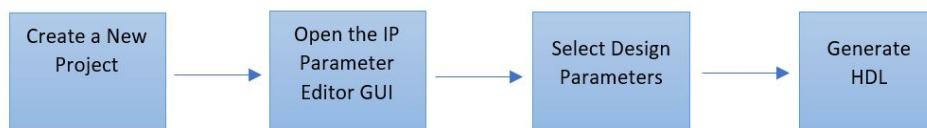
7. Designing with the IP Core

7.1. Generating the IP Core

You can use the Quartus Prime Pro Edition IP Catalog or Platform Designer to define and generate a Multi Channel DMA IP for PCI Express custom component.

Follow the steps shown in the figure below to generate a custom Multi Channel DMA IP for PCI Express component.

Figure 40. IP Generation Flowchart



You can select the Multi Channel DMA IP for PCI Express in the Quartus Prime Pro Edition IP Catalog or Platform Designer as shown below.

Figure 41. Quartus Prime Pro Edition IP Catalog (with filter applied)

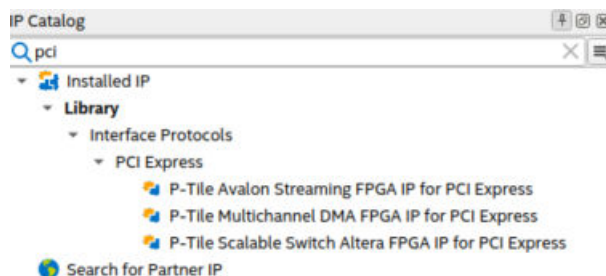
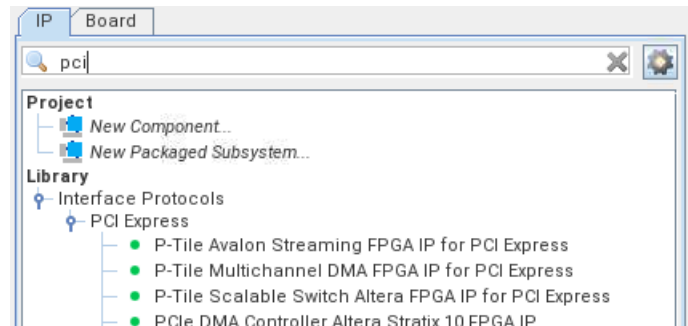


Figure 42. Platform Designer IP Library (with filter applied)

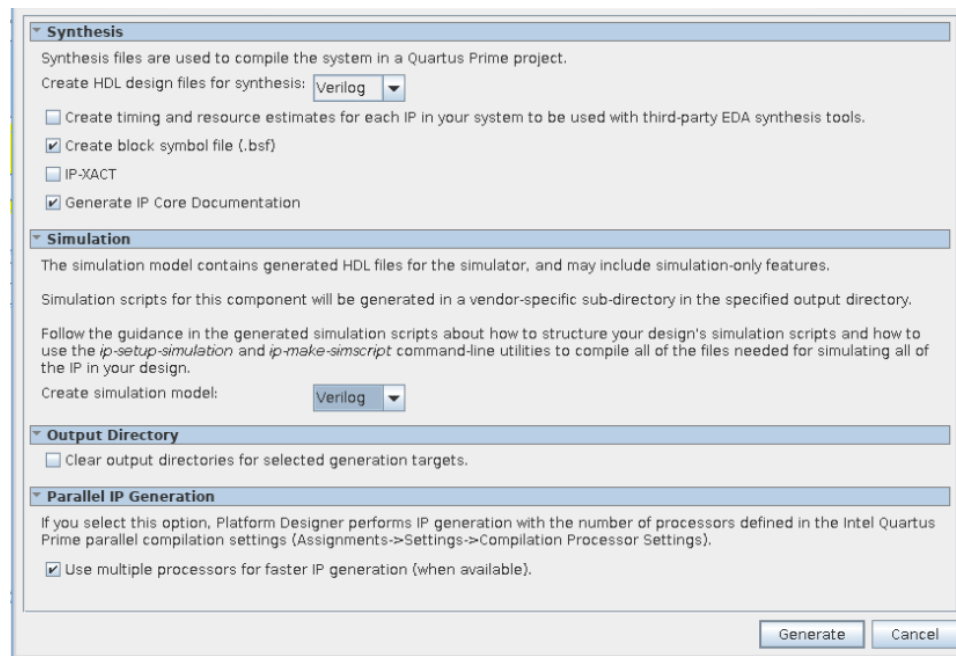


7.2. Simulating the IP Core

The Quartus Prime Pro Edition software optionally generates a functional simulation model, a testbench or design example, and vendor-specific simulator setup scripts when you generate your parameterized Multi Channel DMA for PCI Express IP core. For Endpoints, the generation creates a Root Port BFM. There is no support for Root Ports in this release of the Quartus Prime Pro Edition.

To enable IP simulation model generation, set **Create simulation model** to **Verilog** or **VHDL** when you generate HDL:

Figure 43. Multi Channel DMA IP for PCI Express Simulation in Quartus Prime Pro Edition



For information about supported simulators, refer to the *Multi Channel DMA for PCI Express Intel FPGA IP Design Example User Guide*.

Note: Root Port simulation is supported by VCS only.

Note: The Intel testbench and Root Port BFM provide a simple method to do basic testing of the Application Layer logic that interfaces to the PCIe IP variation. This BFM allows you to create and run simple task stimuli with configurable parameters to exercise basic functionality of the example design. The testbench and Root Port BFM are not intended to be a substitute for a full verification environment. Corner cases and certain traffic profile stimuli are not covered. To ensure the best verification coverage possible, Intel strongly recommends that you obtain commercially available PCIe verification IP and tools, or do your own extensive hardware testing, or both.

Related Information

- [Introduction to Intel FPGA IP Cores](#)
- [Simulating Intel FPGA IP Cores](#)
- [Simulation Quick-Start](#)
- [Multi Channel DMA for PCI Express Design Example User Guide](#)

7.3. IP Core Generation Output - Quartus Prime Pro Edition

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 44. Individual IP Core Generation Output (Quartus Prime Pro Edition)

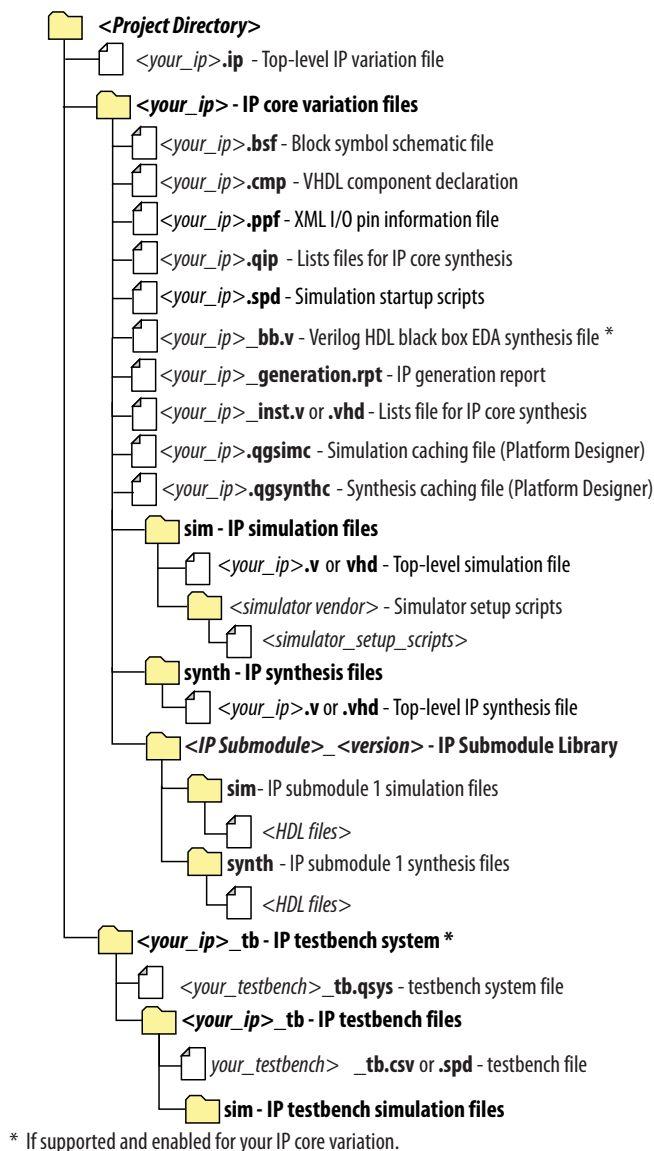


Table 98. Output Files of Intel FPGA IP Generation

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.
continued...	

File Name	Description
<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. This comparison determines if Platform Designer can skip 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 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 _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.

7.4. Systems Integration and Implementation

7.4.1. Required Supporting IP

Stratix 10 and Agilex 7 devices use a parallel, sector-based architecture that distributes the core fabric logic across multiple sectors. Device configuration proceeds in parallel with each Local Sector Manager (LSM) configuring its own sector. Consequently, FPGA registers and core logic are not released from reset at exactly the same time, as has always been the case in previous families.

In order to keep application logic held in the reset state until the entire FPGA fabric is in user mode, Stratix 10 and Agilex 7 devices require you to include the Stratix 10 Reset Release IP.

Refer to the Multi Channel DMA for PCI Express IP design example to see how the Reset Release IP is connected with the Multi Channel DMA for PCI Express IP component.

Related Information

[AN 891: Using the Reset Release Intel FPGA IP](#)

8. Software Programming Model

The Multi Channel DMA IP for PCI Express Linux software consists of the following components:

- Test Applications
- User space library for custom driver (libmqdma)
- DPDK Poll mode based driver
- PCIe end point driver (ifc_uio)
- Kernel mode network device driver (ifc_mcdma_net)

The software files are created in the Multi Channel DMA IP for PCI Express design example project folder when you generate an Multi Channel DMA IP for PCI Express design example from the IP Parameter Editor as shown below. The software configuration is specific to the example design generated by Quartus Prime.

Multi Channel DMA Intel FPGA IP for PCI Express design example project folder has multiple software directories depending on the Hard IP mode selected (1x16, 2x8 or 4x4) for Quartus Prime Pro Edition 23.4 version and onwards. Each software folder is specific to each port:

- p0_software folder is generated only for 1x16 Hard IP modes.
- p1_software folder is generated only for 2x8 Hard IP modes.
- p2_software and p3_software folders are generated only for 4x4 Hard IP modes.

Note: You must use the corresponding software folder with each IP port.

Figure 45. Software Folder Structure

Name

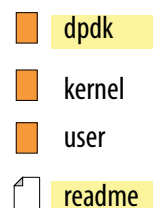


Table 99. MCDMA IP Software Driver Differentiation

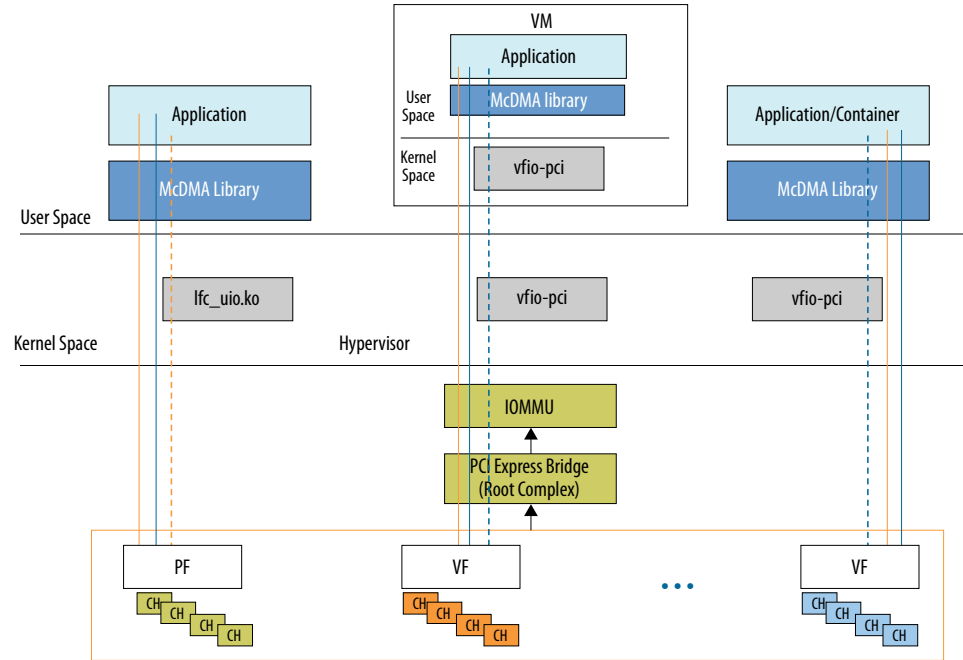
Driver	Description	Use Case / Application
Custom	<ul style="list-style-type: none"> Customized user space MCDMA library, which can be installed in the form of a static library file and corresponding test utility. Supports accessing the device by using VFIO and UIO kernel frameworks. Sample performance-based application developed to demonstrate the performance and usage. 	<p>If you have your own user space platform to use this driver with custom APIs.</p> <p>API information is shared in this User Guide</p> <p>Example: Any user space application which needs DMA features.</p>
DPDK	<ul style="list-style-type: none"> Poll mode MCDMA driver by using DPDK infrastructure and example test application are developed. DPDK Patches are provided to support MSI-X and address some error cases Supports both UIO and VFIO kernel frameworks, which can be enabled at DPDK build time. 	<p>If you use DPDK as your platform, you can integrate this PMD with your DPDK framework to perform DMA.</p> <p>Example: DPDK based NFV applications</p>
Netdev	<ul style="list-style-type: none"> MCDMA network driver exposes the device as ethernet device. (<code>ifconfig</code> displays the device as ethernet device). DMA operations can be initiated in kernel mode and all the TCP/IP applications can be used. Use kernel base framework for DMA memory management. 	<p>All TCP/IP applications can use this driver. <code>iperf</code>, <code>netperf</code>, <code>scapy</code> use this driver.</p>

8.1. Multi Channel DMA IP Custom Driver

8.1.1. Architecture

The figure below shows the software architecture block diagram of MCDMA custom driver.

Figure 46. Block Level Software Architecture



In the above block diagram, dotted lines represent memory mapped I/O interface. The other two lines represent read and write operations triggered by the device.

The Multi Channel DMA IP for PCI Express supports the following kernel based modules to expose the device to user space.

- vfio-pci
- UIO

These drivers do not perform any device management and indicate to the Operating System (OS) that the devices are being used by user space such that the OS does not perform any action (e.g. scanning the device etc.) on these devices.

vfio-pci

This is the secure kernel module, provided by kernel distribution. This module allows you to program the I/O Memory Management Unit (IOMMU). IOMMU is the hardware which helps to ensure memory safety in user space drivers. In case, if you are using Single Root I/O Virtualization (SR-IOV), you can load vfio-pci and bind the device.

- This module enables IOMMU programming and Function level reset (FLR)
- To expose device Base Address Registers (BAR) to user space, vfio-pci enables ioctl
- Supports MSI-X (Message Signal Interrupts extensions) interrupts
- Kernel versions ≥ 5.7 , supports the enablement of virtual functions by using sysfs interface.

If you are using kernel versions below 5.7, you have the following alternatives:

- Use `ifc_uio`, which supports to enable VFs.
- Apply the patch on kernel to enable virtual functions by using `sysfs`. It needs kernel rebuild.

`ifc_uio`

This is the alternative driver to `vfio-pci`, which doesn't use IOMMU.

By using PCIe, `sysfs`, interrupt framework utilities this module reads allows the user space to access the device.

Like `vfio-pci`, this module can also be used from guest VM through the hypervisor. This driver allows the enablement/disablement of virtual functions. Once a virtual function is created, by default it binds to `ifc_uio`. Based on the requirement, you may unbind and bind to another driver.

Following are the functionalities supported by using this module:

- Allows enablement/disablement of virtual functions by using `sysfs` interface.
- Probes and exports channel BARs to `libmqdma`
- Supports Interrupt notification/clearing

`libmqdma`

This is a user-space library used by the application to access the PCIe device.

- This library has the APIs to access the MCDMA IP design and you can develop your application using this API.
- It features calls for allocation, release and reset of the channels
- `libmqdma` supports accessing the devices bound by two user space drivers `UIO` (`uio`) or Virtual Function I/O (VFIO) (`vfio-pci`).

You can tune these options from the make file.

In case of `UIO`, `ifc_uio` driver reads the BAR register info by using `sysfs` and register MSI-X info by using `eventfds`.

In case of `VFIO`, user space uses `IOCTL` command to read BAR registers, MSIX information and programming of IOMMU table.

Typically, when an application is running in a virtualized environment, you bind the device to `vfio-pci` module and `libmqdma` can access the device by using `ioctl`. Currently, the support of `UIO` and `VFIO` can be switched from `common.mk` file. `UIO` is enabled by default.

Sample application

This application uses the APIs from `libmqdma` and takes the following command line arguments as the input.

- Total message sizes/ time duration
- Packet size per descriptor
- Write/Read
- Completion reporting method
- Number of channels

It runs multiple threads for accessing the DMA channel. It also has performance measuring capabilities. Based on the number threads you are using and number of channels you are processing, queues are scheduled on threads.

8.1.2. libmqdma library details

libmqdma library has the user space framework which enables the DMA operation with PCIe device, is responsible for following actions:

- Device management
- Channel management
- Descriptor Memory Management
- Interrupts management

The libmqdma framework is installed on the host as a dynamic link library and exports the APIs to the application. Applications running in user space are responsible to use MCDMA IP by using those APIs.

8.1.2.1. Channel Initialization

When **libmqdma** is handing over the available channel to the application, it performs the following functions:

1. Reset the channel
 - The libmqdma sets the reset register of the channel.
 - Polls the register back till the reset happens.

This process resets the queue logic and sets all the channel parameters to default.

2. Initialize the channel
 - Allocates the required number of descriptors in the host.
 - Update the starting address of the descriptors to the registers.
 - Update the number of descriptors.

Based on these parameters, hardware performs queue management.

3. Enable the channel

8.1.2.2. Descriptor Memory Management

At the time of channel initialization, the device allocates the descriptor and data memory.

Descriptor memory

Maximum length of data in descriptor is 1 MB. Link specifies whether the next descriptor is in any other page.

AVST H2D/D2H descriptor

- source address
- Destination address
- Data length
- Start of file (SOF)

- End of file (EOF)
- Descriptor index
- Link

Application need to pass these values to the hardware through the libmqdma.

Data Memory

The user space data page can be much bigger than the normal TLB entry page size of 4 KB. libmqdma library implements the allocator to organize the memory.

Following are the hardware registers which the software updates as part of the channel enumeration.

- **Q_START_ADDR_L, Q_START_ADDR_H**: Contains the physical address of the start of the descriptor array.
- **Q_SIZE**: Logarithmic value of number of descriptors
- **Q_CONS_HEAD_ADDR_L, Q_CONS_HEAD_ADDR_H**: Physical address of the head index of the ring, where FPGA sync the value of the head.

8.1.2.3. Descriptor Completion Status Update

There are two modes for selecting the descriptor completion status, MSI-X and Writeback mode. The default mode is writeback. This can be changed in the following C header file.

```
pX_software/user/common/include/ifc_libmqdma.h

/* Set default descriptor completion */
#define IFC_CONFIG_QDMA_COMPL_PROC <Set with following method >
```

Descriptor Completion Status Update Method

1. **Writeback mode: (CONFIG_QDMA_QUEUE_WB)**: In this approach, MCDMA IP updates the completed descriptor index in the host memory. libmqdma goes for local read and not for PCIe read.
2. **MSI-X interrupt mode: (CONFIG_QDMA_QUEUE_MSIX)**: In this approach, when the transaction completed, MCDMA IP sends the interrupt to the Host and updates the completed descriptor index in the host memory. libmqdma reads the completion status up on receiving the interrupt.
3. **Register Mode (CONFIG_QDMA_QUEUE_REG)**: In this approach, driver knows the completion status by polling the completion head register. As register read is costly from host perspective, performance of smaller payloads would be less in this approach.

8.1.3. Application

At the time of starting the application, by using the APIs provided by driver, it reads the MCDMA capabilities, creates application context, open BAR registers, initializes the PCI Express functions. At the time of termination, it clears the application context and stops all the channels.

Multi-threading with Multiple Channels

Based on the input parameters, the application starts multiple threads with posix thread APIs, associate the queue to the thread and submit DMA transactions one at a time independently. As part of this, driver updates the tail register of that channel. Update tid ID update, hardware picks up the channel and start DMA operation.

Each thread performs the following tasks.

1. Get the device context based on BDF (Bus Device Function)
2. Acquire the available channel
3. Get DMA capable memory
4. Start DMA operation
5. Release the channel

As multiple threads can try to grab and release the channel at a time, userspace driver (libmqdma) handles synchronization problems while performing channel management.

Scheduling Threads

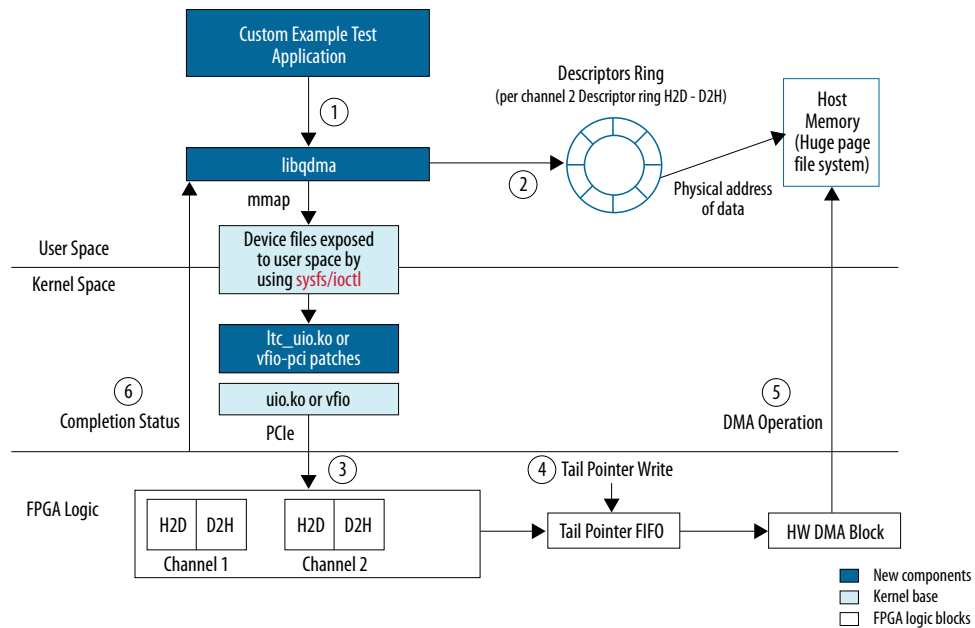
As POSIX libraries are being used for thread management, Linux scheduler takes care of scheduling the threads, there is no custom scheduler which takes care of scheduling the threads.

perfq_app schedules multiple queues on single threads for DMA operations.

1. Reads number of channels from user (-c <num>)
2. Reads number of threads from user (-a <num>)
3. Calculate number of queues DMA need to perform from one thread
4. After every TID update, perfq_app swaps out a queue and swaps in other queue to perform DMA operation.

8.1.4. Software Flow

Figure 47. Multi Channel DMA IP for PCI Express Software Operation Flow



Step 1

- Application creates the thread based on the required port of a channel
- After spawning the thread, the thread tries to acquire the available channel's port. In case if all channels ports are busy thread waits in poll mode
- In the context of thread, libqdma allocates descriptors buffer memory in the host
- libqdma initializes following registers in QCSR associates with queue, which includes Starting address of descriptors, queue size, write back address for Consumed Head, payload size in D2H descriptors and then enables the channels

QCSR registers:

- Q_RESET (offset 8'h48)
- Q_TAIL_POINTER (offset 8'h14) Set 0
- Q_START_ADDR_L (Offset 8'h08)
- Q_START_ADDR_H (Offset 8'h0C)
- Q_SIZE (Offset 8'h10)
- Q_CONSUMED_HEAD_ADDR_L (Offset 8'h20)
- Q_CONSUMED_HEAD_ADDR_H (Offset 8'h24)
- Q_BATCH_DELAY (Offset 8'h28)
- Set q_en, q_wb/intr_en bits Q_CTRL (Offset 8'h00)
- (Q_PYLD_COUNT) (Offset 8'h44)

GCSR register:

- WB_INTR_DELAY (Offset 8'h08)

Step 2

- Threads continuously try to send/receive the data and library keeps checking if channel is busy or descriptor ring is full
- If channel is not busy and descriptor ring is not full it goes to step 3. If channel is busy or descriptors ring is full thread retries to initiate the transfer again

Descriptor ring full is identified by checking the Consumed Head and Tail pointer registers.

Step 3

Thread requests for new descriptor to submit the request and updates the required field i.e. descriptor index, SOF, EOF, Payload, MSI-X enable and writeback enable.

Step 4

After initializing descriptor ring buffer, the libqdma writes number of descriptor updates into tail register of QCSR region. On every descriptor update the tail pointer is increased by 1.

QCSR tail pointer register: Q_TAIL_POINTER (Offset 8'h14)

Step 5

- Once the tail pointer write happens, Multi Channel DMA IP for PCI Express fetches descriptors from host memory starting from the programmed Q_START_ADDR_L/H address
- Multi Channel DMA IP for PCI Express parses the descriptor content to find the sources, destination addresses and length of the data from descriptor and starts DMA operation

Step 6

Once descriptor processing is completed, IP notifies the completion status based on following methods, which can be enabled in each descriptor.

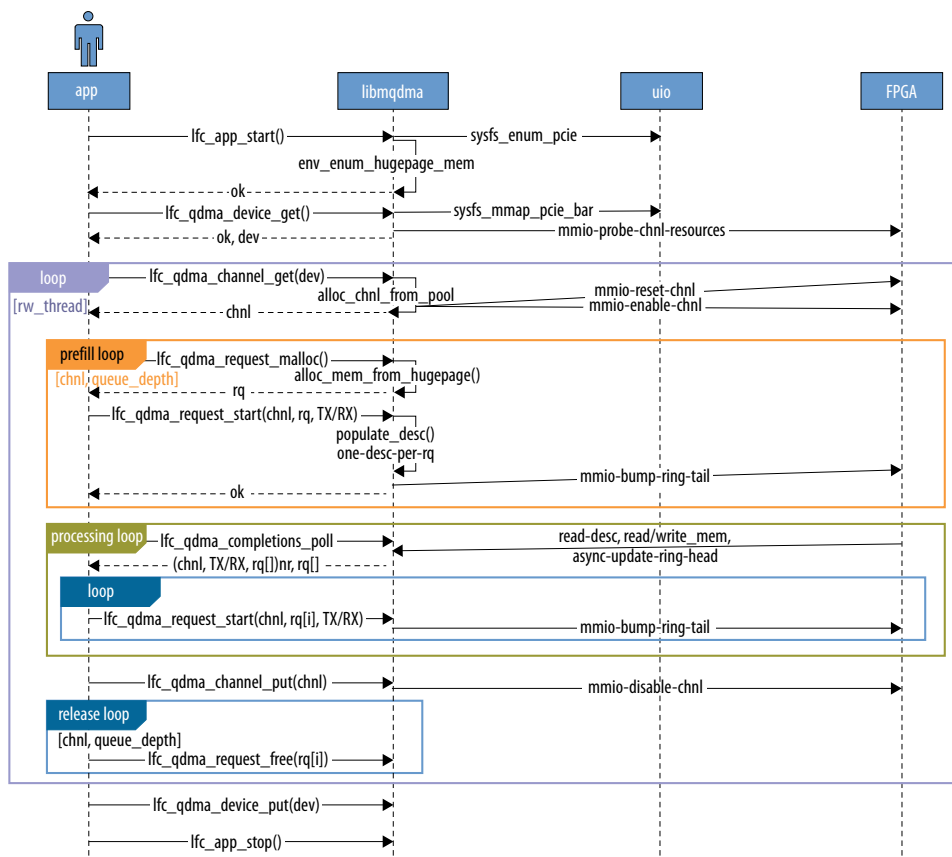
- Either based on MSI-X Interrupt : Multi Channel DMA IP for PCI Express sends MSI-X interrupt to host if enabled in Q_CTRL.
- Writeback: Multi Channel DMA for PCI Express IP updates Q_CONSUMED_HEAD_ADDR_L/H, if writeback is enabled in Q_CTRL.

8.1.5. API Flow

8.1.5.1. Single Descriptor Load and Submit

The API flow below shows loading one descriptor in the descriptor ring buffer and then submit DMA transfer by updating the tail pointer register by increment of 1.

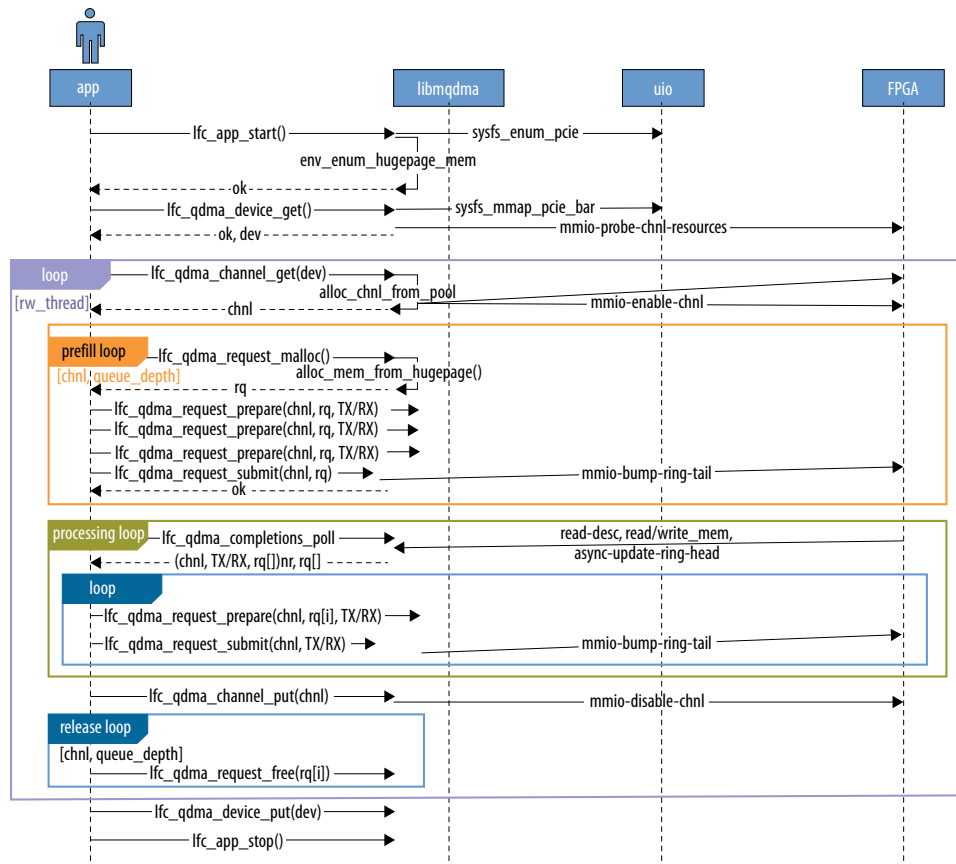
Figure 48. Single Descriptor Load and Submit



8.1.5.2. Multiple Descriptor Load and Submit

The API flow below shows loading the descriptors in bunch in the descriptor ring buffer and then submit for DMA transfer by updating the tail pointer register with total loaded descriptors.

Figure 49. Multiple Descriptor Load and Submit



8.1.6. libmqdma Library API List

This section describes the list of APIs which are exposed to the application.

8.1.6.1. ifc_api_start

Table 100. ifc_api_start

API	API Description	Input Parameters	Return Values
void ifc_app_start(void)	This function is called at the time of application initialization. Probe and prepare the application for DMA transactions. <ul style="list-style-type: none"> maps the enabled device memory to user space memory allocation from huge page file system hugepagefs allows user space to get continuous and unswappable memory, which you can use for DMA operations.	void	0 on success negative otherwise

API	API Description	Input Parameters	Return Values
	Set default huge page size as 1 GB at boot time and allocate the required memory from huge pages.		

8.1.6.2. ifc_mcdma_port_by_name

Table 101. ifc_api_start

API	API Description	Input Parameters	Return Values
int ifc_mcdma_port_by_name(const char* bdf)	This function returns the port number to corresponding BDF	BDF	0 on success negative otherwise

8.1.6.3. ifc_qdma_device_get

Table 102. ifc_qdma_device_get

API	API Description	Input Parameters	Return Values
int ifc_qdma_device_get(int port, struct ifc_qdma_device **qdev)	Based on the port number, API returns corresponding device context to the application. Application must maintain the device context and use it for further operations. When the application is done with I/O, it releases the context by using ifc_qdma_device_put API.	port - port number of the device, which is returned by ifc_mcdma_port_by_name API qdev - Address of the pointer to device context	Updates device context and returns 0 on success negative otherwise

8.1.6.4. ifc_num_channels_get

Table 103. ifc_num_channels_get

API	API Description	Input Parameters	Return Values
int ifc_num_channels_get(struct ifc_qdma_device *qdev);	This API returns the total number of channels supported by QDMA device	qdev - Pointer to device context	Number of channels supported

8.1.6.5. ifc_qdma_channel_get

Table 104. ifc_qdma_channel_get

API	API Description	Input Parameters	Return Values
int ifc_qdma_channel_get(struct ifc_qdma_device *qdev, struct ifc_qdma_channel **chnl, int chno)	Before submitting the DMA transactions, application is responsible to acquire the channel and pass the context on further interactions with framework. This API performs following:	qdev: QDMA device chnl: Pointer to update channel context chno: Channel no if user wants specific channel. -1 if no specific	0 : on success and populates channel context -1 : No channel is ready to be used. Channel context is returned as NULL. -2 : Requested channel is already allocated. But valid channel context is returned. Application may use this channel context.

API	API Description	Input Parameters	Return Values
	<ul style="list-style-type: none"> Get the next available channel Initialize the descriptors and data memory for both TX and RX queues Enable the channel <p>The last parameter in this API is the channel number. In case if you pass this parameter as -1, it returns available free channel. Otherwise, it allocates the available free channel.</p>		

8.1.6.6. ifc_qdma_acquire_channels

ifc_qdma_acquire_channels

Table 105.

API	API Description	Input Parameters	Return Values
int ifc_qdma_acquire_channels(struct ifc_qdma_device *qdev, int num)	This API acquires n number of channels from hardware. Once the channels acquired, user must call ifc_qdma_channel_get() to initialize the channels and use for DMA.	qdev: QDMA device num: Number of channels requested	Number of channels acquired successfully. negative otherwise

8.1.6.7. ifc_qdma_release_all_channels

Table 106. ifc_qdma_release_all_channels

API	API Description	Input Parameters	Return Values
int ifc_qdma_release_all_channels(struct ifc_qdma_device *qdev)	This API releases all the channels acquired by the device. User must make sure to stop the traffic on all the channels, before calling this function. Perfq_app calls this API at the exit of application.	qdev: QDMA device	0 on success Negative otherwise

8.1.6.8. ifc_qdma_device_put

Table 107. ifc_qdma_device_put

API	API Description	Input Parameters	Return Values
void ifc_qdma_device_put(struct ifc_qdma_device *qdev)	This API performs the unmapping of the device memory and releases the allocated resources.	qdev: QDMA device	0 on success negative otherwise

8.1.6.9. ifc_qdma_channel_put

Table 108. ifc_qdma_channel_put

API	API Description	Input Parameters	Return Values
<pre>void ifc_qdma_channel_put(struct ifc_qdma_channel *qchnl)</pre>	<p>Once the DMA transactions are completed, the application must call this API to release the acquired channel so that another process or another thread will acquire again.</p> <p>libmqdma disables this channel so that hardware does not look for DMA transactions.</p>	qchnl: channel context	0 on success, negative otherwise

8.1.6.10. ifc_qdma_completion_poll

Table 109. ifc_qdma_completion_poll

API	API Description	Input Parameters	Return Values
<pre>int ifc_qdma_completion_poll(struct ifc_qdma_channel *qchnl, int direction, void *pkt, int quota)</pre>	<p>Check if any previously queued and pending request got completed. If completed, the number of completed transactions are returned to called so that the application submits those transactions.</p>	<p>qchnl: channel context</p> <p>dir: DMA direction, one of IFC_QDMA_DIRECTION_*</p> <p>pkts: address where completed requests to be copied</p> <p>quota: maximum number of requests to search</p>	0 on success, negative otherwise

8.1.6.11. ifc_qdma_request_start

Table 110. ifc_qdma_request_start

API	API Description	Input Parameters	Return Values
<pre>int ifc_qdma_request_start(struct ifc_qdma_channel *qchnl, int dir, struct ifc_qdma_request *r);</pre>	<p>Depending on data direction, application selects TX/RX queue, populates the descriptors based on the passed request object and submits the DMA transactions.</p> <p>This is not blocking request. You may need to poll for the completion status.</p>	<p>qchnl: channel context received on ifc_qchannel_get()</p> <p>dir: DMA direction, one of IFC_QDMA_DIRECTION_*</p> <p>r: request struct that needs to be processed</p>	0 on success, negative otherwise

8.1.6.12. ifc_qdma_request_prepare

Table 111. ifc_qdma_request_prepare

API	API Description	Input Parameters	Return Values
<pre>int ifc_qdma_request_prepare(struct ifc_qdma_channel *qchnl, int dir, struct ifc_qdma_request *r);</pre>	<p>Depending on the direction, application selects the queue and prepares the descriptor but not submit the transactions. Application must use</p>	<p>qchnl: channel context received on ifc_qchannel_get()</p> <p>dir: DMA direction, one of IFC_QDMA_DIRECTION_*</p> <p>r: request struct that needs to be processed</p>	Returns the number of transactions completed. negative otherwise

API	API Description	Input Parameters	Return Values
	ifc_qdma_request_submit API to submit the transactions to DMA engine.		

8.1.6.13. ifc_qdma_descq_queue_batch_load

Table 112. ifc_qdma_descq_queue_batch_load

API	API Description	Input Parameters	Return Values
int ifc_qdma_descq_queue_batch_load(struct ifc_qdma_channel *qchnl, void *req_buf, int dir, int n)	Depending on the direction, application selects the queue and prepares n number of descriptors but does not submit the transactions. Application must use ifc_qdma_request_submit API to submit the transactions to DMA engine.	qchnl: channel context received on ifc_qchannel_get() dir: DMA direction, one of IFC_QDMA_DIRECTION_* r: request struct that needs to be processed.	Returns the number of transactions completed. negative otherwise

8.1.6.14. ifc_qdma_request_submit

Table 113. ifc_qdma_request_submit

API	API Description	Input Parameters	Return Values
int ifc_qdma_request_submit(struct ifc_qdma_channel *qchnl, int dir);	Submits all prepared and pending DMA transactions to MCDMA engine. Before calling this API, Application may need to call ifc_qdma_request_prepare to prepare the transactions. If you want to give priority to one channel and submit more number of transactions at a time from one channel, application may need to call multiple times and then submit APIs at a time to submit all the transactions.	qchnl: channel context received on ifc_qchannel_get() dir: DMA direction, one of IFC_QDMA_DIRECTION_*	0 on success negative otherwise

8.1.6.15. ifc_qdma_pio_read32

Table 114. ifc_qdma_pio_read32

API	API Description	Input Parameters	Return Values
uint32_t ifc_qdma_pio_read32(struct ifc_qdma_device *qdev, uint64_t addr)	Read the value from BAR2 address This API is used for PIO testing, dumping statistics, and pattern generation.	qdev: QDMA device addr: address to read	0 on success negative otherwise

8.1.6.16. ifc_qdma_pio_write32

Table 115. ifc_qdma_pio_write32

API	API Description	Input Parameters	Return Values
void ifc_qdma_pio_write32(struct ifc_qdma_device *qdev, uint64_t addr, uint32_t val)	Writes the value to BAR2 address	qdev: QDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

8.1.6.17. ifc_qdma_pio_read64

Table 116.

API	API Description	Input Parameters	Return Values
uint64_t ifc_qdma_pio_read64(struct ifc_qdma_device *qdev, uint64_t addr);	Read the value from BAR2 address This API would be used for PIO testing, dumping statistics, pattern generation etc.	qdev: QDMA device addr: address to read	0 on success negative otherwise

8.1.6.18. ifc_qdma_pio_write64

Table 117.

API	API Description	Input Parameters	Return Values
void ifc_qdma_pio_write64(struct ifc_qdma_device *qdev, uint64_t addr, uint64_t val)	Writes 64 bit value to BAR2 address	qdev: QDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

8.1.6.19. ifc_qdma_pio_read128

Table 118.

API	API Description	Input Parameters	Return Values
uint128_t ifc_qdma_pio_read128(struct ifc_qdma_device *qdev, uint64_t addr);	Read the value from BAR2 address This API would be used for PIO testing, dumping statistics, pattern generation etc.	qdev: QDMA device addr: address to read	0 on success negative otherwise

8.1.6.20. ifc_qdma_pio_write128

Table 119.

API	API Description	Input Parameters	Return Values
void ifc_qdma_pio_write128(struct ifc_qdma_device *qdev, __uint128_t addr, uint64_t val)	Writes 64 bit value to BAR2 address	qdev: QDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

8.1.6.21. ifc_qdma_pio_read256

Table 120.

API	API Description	Input Parameters	Return Values
int ifc_qdma_pio_read128(struct ifc_qdma_device *qdev, uint64_t addr, void *vall);	Read 128 bit value from BAR2 address This API would be used for PIO testing, dumping statistics, pattern generation etc.	qdev: QDMA device addr: address to read	0 on success negative otherwise

8.1.6.22. ifc_qdma_pio_write256

Table 121.

API	API Description	Input Parameters	Return Values
void ifc_qdma_pio_write128(struct ifc_qdma_device *qdev, __uint128_t addr, uint64_t *val)	Writes 128 bit value to BAR2 address	qdev: QDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

8.1.6.23. ifc_request_malloc

Table 122. ifc_request_malloc

API	API Description	Input Parameters	Return Values
struct ifc_qdma_request *ifc_request_malloc(si ze_t len)	libmqdma allocates the buffer for I/O request. The returned buffer is DMA-able and allocated from huge pages	len - size of data buffer for I/O request	0 on success Negative otherwise

8.1.6.24. ifc_request_free

Table 123. ifc_request_free

API	API Description	Input Parameters	Return Values
void ifc_request_free(void *req);	Release the passed buffer and add in free pool	req - start address of allocation buffer	0 : on success and populates channel context -1 : channel not available -2 : Requested for specific channel. But already occupied

8.1.6.25. ifc_app_stop

Table 124. ifc_app_stop

API	API Description	Input Parameters	Return Values
void ifc_app_stop(void)	ifc_app_stop unmaps the mapped resources and allocated memory	void	void

8.1.6.26. ifc_qdma_poll_init

Table 125. ifc_qdma_poll_init

API	API Description	Input Parameters	Return Values
int ifc_qdma_poll_init(struct ifc_qdma_device *qdev)	This resets the poll eventfds Application, need to pass this fd_set to poll in case if MSI-X interrupts enabled.	qdev: QDMA device	0 on success Negative otherwise

8.1.6.27. ifc_qdma_poll_add

Table 126. ifc_qdma_poll_add

API	API Description	Input Parameters	Return Values
int ifc_qdma_poll_add(struct ifc_qdma_device *qdev, ifc_qdma_channel *chnl, int dir)	Append event fds to the poll list	qdev: QDMA device chnl: channel context dir: direction, which needs to poll	0 on success Negative otherwise

8.1.6.28. ifc_qdma_poll_wait

Table 127. ifc_qdma_poll_wait

API	API Description	Input Parameters	Return Values
int ifc_qdma_poll_wait(struct ifc_qdma_device *qdev, ifc_qdma_channel **chnl, int *dir)	Monitor for interrupts for all added queues. In case if any interrupt comes, it will return. Timeout: 1 msec	qdev: QDMA device qchan: address of channel context dir: address of direction parameters	0 on success and updates channel context and direction Negative otherwise

8.1.6.29. ifc_mcdma_port_by_name

Table 128. ifc_api_start

API	API Description	Input Parameters	Return Values
int ifc_mcdma_port_by_name(const char* bdf)	This function return the port number to corresponding BDF	Input parameters: BDF	negative otherwise Return values 0 on success

8.1.7. Request Structures

To request a DMA operation, the application needs to use a common structure called **ifc_qdma_request**.

It contains the following fields.

```
struct ifc_qdma_request {
    void *buf;           /* src/dst buffer */
    uint32_t len;        /* number of bytes */
    uint64_t pyld_cnt;
    uint32_t flags;      /* SOF and EOF */
}
```

```
uint64_t metadata;
void *ctx;          /* libqdma contextst, NOT for application */
};
```

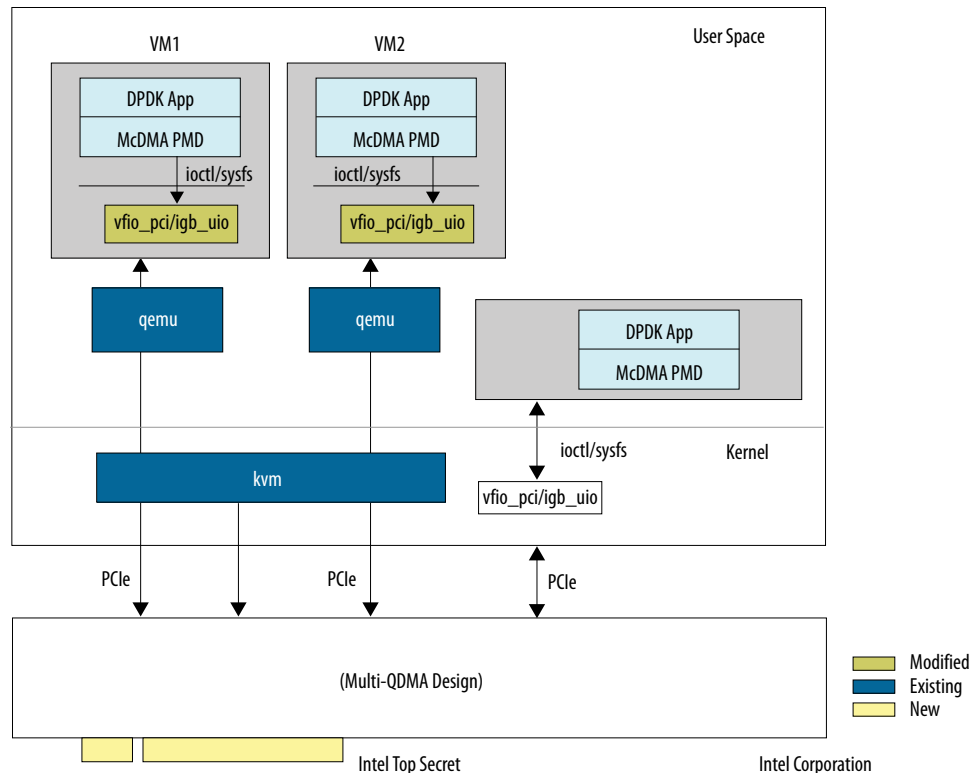
1. **buf:** - DMA buffer. In the case of H2D, this buffer data is moved to FPGA. In the case of D2H, FPGA copies the content to this buffer.
2. **len:** Length of the data in this descriptor
3. **pyld_cnt:** D2H: Length of the valid date, in case if descriptor contains EOF. H2D: This field not used
4. **flags:** This the mask which contains the flags which describe the content. Currently, these flags are being used to notify the SOF and EOF of data.
5. **metadata:** In case of H2D, you need to update the metadata in this field. In case of D2H driver, updates back the metadata

Note: For the Single Port AVST Design, the sof and eof should be on the same descriptor or SOF can be at the start and EOF at the end descriptor of a single TID update.

8.2. Multi Channel DMA IP DPDK Poll-Mode based Driver

8.2.1. Architecture

Figure 50. MCDMA Driver Architecture



igb_uio

This is the PCIe end point kernel module provided by DPDK framework and on top of this, there are some paths added to support MSI-X and SRIOV features. By using PCIe, **sysfs** interrupt framework utilities, this module reads allows the user space to access the device.

Following are the functionalities supported by using this module:

- Probes and exports channel BARs to User space
- Supports Interrupt notification/clearing
- Enable SRIOV functionality and set virtual functions

igb_uio kernel module does not support IOMMU programming.

vfio-pci

vfio-pci is the base kernel module, allows you to access the device and allows IOMMU programming by using **ioctl** interface. If you want to enable the VFs by using **vfio-pci**, you may need to use the kernel version >5.7

MCDMA PMD

This is a poll mode driver which implements the APIs to perform channel management, device management and also DMA on both H2D and D2H directions. part This module exposes the device as **ethdev** interface.

Example Test Application

aUsing DPDK Environment Abstraction Layer (EAL) utilities to perform the memory management and device management.

In this application you are using to continuously sending/receiving data traffic from/to device, use the following command line arguments as the input.

- Total message sizes/ time duration
- Packet size per descriptor
- Write/Read
- Completion reporting method
- Number of channels

The test application runs multiple threads for accessing the DMA channels. It also has performance measurement capability. Based on the number threads being used and number of channels being processed, queues are scheduled on threads.

testpmd

The testpmd application can also be used to test the DPDK in a packet forwarding mode.

The following command line arguments are used to initiate data transfer from Host to device or device to Host:

- Forwarding mode
- Number of CPU cores
- TX and RX channels per port
- Number of packets per burst
- Number of descriptors in the RX and TX ring
- Maximum packet length

Note: testpmd support is provided only in CentOS and not provided in Ubuntu.

8.2.2. MCDMA Poll Mode Driver

8.2.2.1. Completion Status Update

There are two modes for selecting the descriptor completion status, MSI-X mode & Writeback mode. The default mode is Writeback. This can be changed, if desired, in the following C header file.

```
drivers/net/mcdma/rte_pmd_mcdma.h

/* Set default descriptor completion */
#define IFC_CONFIG_MCDMA_COMPL_PROC <Set with following method >
```

Writeback mode: (CONFIG_MCDMA_QUEUE_WB)

In this approach, MCDMA IP updates the completed descriptor index in the host memory. MCDMA PMD goes for local read and not for PCIe read.

MSI-X interrupt mode: (CONFIG_MCDMA_QUEUE_MSIX)

In this approach, when the transaction is completed, the MCDMA IP sends the interrupt to the Host and updates the completed descriptor index in the host memory. MCDMA PMD reads the completion status up on receiving the interrupt.

Register Mode (CONFIG_QDMA_QUEUE_REG)

In this approach, driver knows the completion status by polling the completion head register. As register read is costly from host perspective, performance of smaller payloads would be less in this approach.

8.2.2.2. Metadata Support

Metadata is the 8 byte private data, which the Host wants to send to the device in H2D direction and the device wants to send to Host in D2H direction. In case of AVMM interface, both **srcaddr** and **dstaddr** fields are used. In case of AVST interface, **dstaddr** in H2D, **srcaddr** in D2H are used to store private meta data. As both addresses are used in AVMM, metadata support is not available if AVMM is enabled.

Table 129. Field Definition

Field Name in Descriptor	Descriptor (Avalon-ST)
srcaddr	H2D - source address of data in host
<i>continued...</i>	

Field Name in Descriptor	Descriptor (Avalon-ST)
	D2H - 64-bit metadata
dstaddr	H2D - 64-bit metadata D2H - destination address of data in host

8.2.2.3. User MSI-X

Each DMA queue is allocated with 2 interrupts:

- For reporting Descriptor completion status.
- For User MSI-X: If some event or error happens, user logic generates the User MSI-X.

Application is responsible to poll for a set of registered interrupt addresses and if User MSI-X is triggered, the corresponding registered interrupt callback gets called. Currently, this callback address is being sent in private pointers of queue config registers.

For D2H queue

```
int
rte_eth_rx_queue_setup(uint16_t port_id, uint16_t rx_queue_id,
                        uint16_t nb_rx_desc, unsigned int
socket_id,
                        const struct rte_eth_rxconf
*rx_conf,
                        struct rte_mempool *mp)
struct rte_eth_rxconf {
    struct rte_eth_thresh rx_thresh; /**< RX ring threshold registers. */
    uint16_t rx_free_thresh; /**< Drives the freeing of RX descriptors. */
    uint64_t offloads;
    ...
    uint64_t reserved_64s[2]; /**< Reserved for future fields */
    void *reserved_ptrs[2]; reserved_ptrs[0] should be populated with
user MSIX callback
};
```

For H2D queue

```
Int
rte_eth_tx_queue_setup(uint16_t port_id, uint16_t tx_queue_id,
                        uint16_t nb_tx_desc, unsigned int
socket_id,
                        const struct rte_eth_txconf
*tx_conf)
struct rte_eth_txconf {
    struct rte_eth_thresh tx_thresh; /**< TX ring threshold registers. */
    uint16_t tx_rs_thresh; /**< Drives the setting of RS bit on TXDs. */
    uint16_t tx_free_thresh;
    ...
    uint64_t reserved_64s[2]; /**< Reserved for future fields */
    void *reserved_ptrs[2]; reserved_ptrs[0] should be populated with
user MSIX callback
};
```

8.2.3. DPDK based application

MCDMA PMD exposes the device as an Ethernet device. By using environment abstraction layer utilities, application initializes the device, memory and interrupts based on the capability. By using **ethdev** APIs, this application performs DMA in H2D and D2H directions.

Multi-threading with Multiple Channels

Based on the input parameters, application starts multiple threads and submits DMA transactions one at a time in run to completion model. As multiple threads try to grab and release the channel at a time, MCDMA PMD handles synchronization problems while performing channel management.

Scheduling queues from threads

DPDK thread management libraries are used for thread creation and initialization. As more number of queues must be supported from single thread, test application schedules multiple queues on single threads for DMA operations.

- BDF (Eg: "-b 86:00.0")
- Reads number of threads from user (-a <num>)
- Calculate number of queues DMA need to perform. (-c <num>)
- Calculate number of queues per thread and assigns multiple threads to single thread.
- After every transaction, testapp swaps out a queue and swaps in other queue to perform DMA operation.
- Batch size (-x <num>)

Software Process Kernel Context Switching

User space driver performs DMA operation. Hence, kernel context switch handling is not needed in any scenario.

8.2.4. Request Structures

To request a DMA operation, the application uses `rte_mbuf` structure which contains packet information such as Physical address of segment buffer, length of the data in this descriptor, Number of segments, 8-byte userdata which is used to notify SOF and EOF of data.

Note:

For the Avalon-ST Design, the application should pass the sof, eof and metadata to driver in the private structure added in `rte_mbuf` structure. sof and eof flags should be updated based on file size. For example, if `file_size = 127`, 0th descriptor should contains SOF flag should be set and 126th descriptor should contains EOF file should be set.

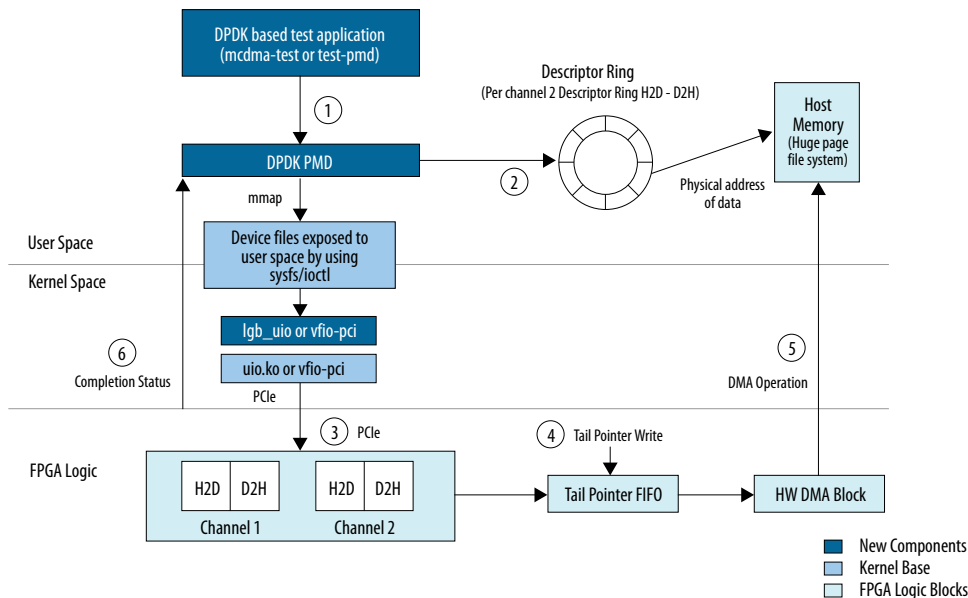
Metadata is private information and can be used for any purpose.

Structure of Private Data

```
struct private_data {
    uint64_t flags; /* SOF, EOF */
    uint64_t metadata; /* Private meta data */
};
```

8.2.5. Software Flow

Figure 51. DPDK Poll-Mode Driver Software Flow



Step 1

- Based on the specified number of queues, application sets up Tx and Rx queues.
- MCDMA Poll mode driver (PMD) takes care of memory management and reserves a portion of physical memory with specified alignment and boundary.
- PMD initializes following registers in QCSR associated with the queue, which includes Starting address of descriptors, queue size, write back address for Consumed Head, payload size in D2H descriptors and then enables the channels.

- QCSR registers:
 - Q_RESET (offset 8'h48)
 - Q_TAIL_POINTER (offset 8'h14) Set 0
 - Q_START_ADDR_L (Offset 8'h08)
 - Q_START_ADDR_H (Offset 8'h0C)
 - Q_SIZE (Offset 8'h10)
 - Q_CONSUMED_HEAD_ADDR_L (Offset 8'h20)
 - Q_CONSUMED_HEAD_ADDR_H (Offset 8'h24)
 - Q_BATCH_DELAY (Offset 8'h28)
 - Set q_en, q_wb/intr_en bits, Q_CTRL (Offset 8'h00)
 - (Q_PYLD_COUNT) (Offset 8'h44)
- Once all the queues are configured it then starts the device.
- Q Application creates the thread based on the number of queues specified.

Step 2

Thread requests for new descriptor to submit the request and updates the required field i.e., descriptor index, SOF, EOF, Payload, MSI-X enable and writeback enable.

Step 3

After initializing descriptor ring buffer, the MCDMA PMD writes number of descriptor updates into tail register of QCSR region. On every descriptor update the tail pointer is increased by 1. QCSR tail pointer register: Q_TAIL_POINTER (Offset 8'h14)

Step 4

- Once the tail pointer write happens, MCDMA IP fetches descriptors from host memory starting from the programmed Q_START_ADDR_L/H address.
- MCDMA IP parses the descriptor content to find the sources, destination addresses and length of the data from descriptor and starts DMA operation.

Step 5

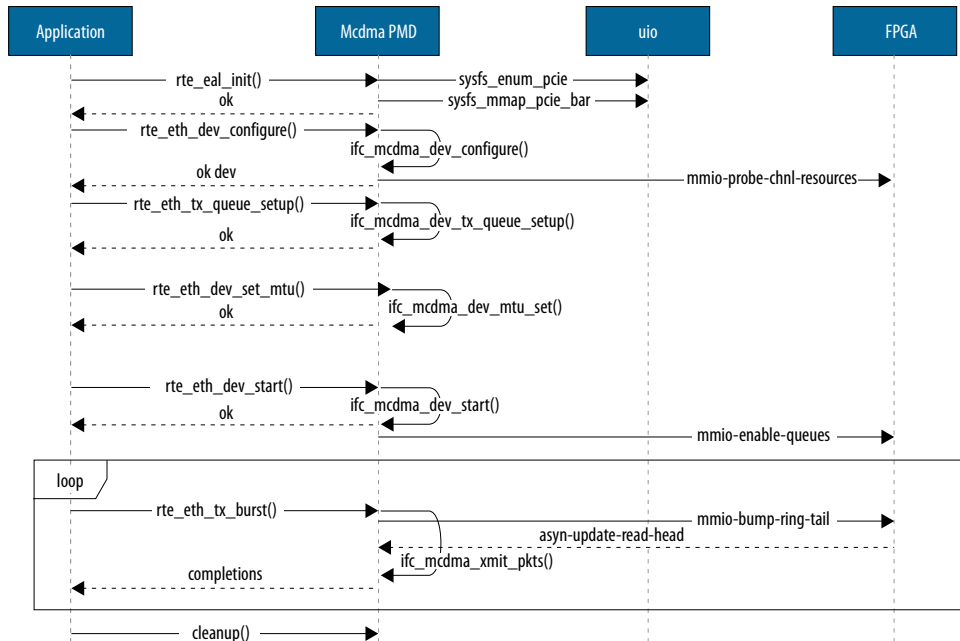
Once descriptor processing is completed, IP notifies the completion status based on following methods, which can be enabled in each descriptor.

- Either based on MSI-X Interrupt: MCDMA IP sends MSI-X interrupt to host if enabled in Q_CTRL.
- Writeback: MCDMA IP updates Q_CONSUMED_HEAD_ADDR_L/H, if writeback is enabled in Q_CTRL.

8.2.6. API Flow

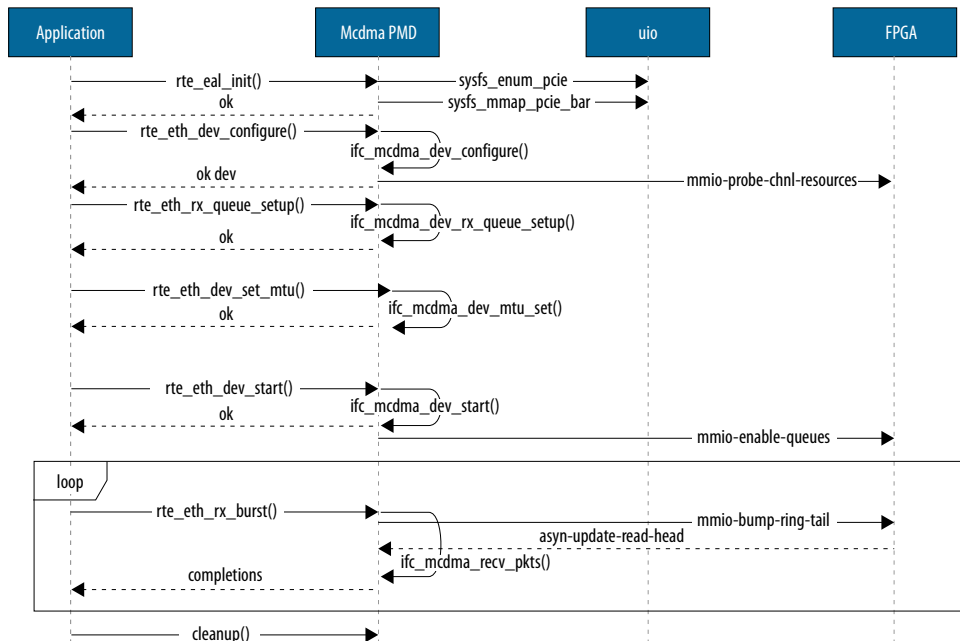
The flow between the Host software components and hardware components is depicted in below sequence diagram for Host to Device data transfer.

Figure 52. Host to Device Sequence



The flow between the Host software components and hardware components is depicted in below sequence diagram for Device to Host data transfer.

Figure 53. Device to Host Sequence



8.2.7. API List

Table 130. rte_eth_dev_configure

API	API Description	Input Parameters	Return Values
int rte_eth_dev_configure(uint16_t port_id, uint16_t nb_rx_q, uint16_t nb_tx_q, const struct rte_eth_conf *dev_conf)	This API configures an Ethernet device. This function must be invoked first before any other function in the Ethernet API.	port ID: device ID nb_tx_queues : Number of TX Queues num_rx_queues: Number of Rx Queues eth_config : input configuration	0 Success, device configured <0 Error code returned by the driver configuration function

Table 131. rte_eth_tx_queue_setup

API	API Description	Input Parameters	Return Values
Int rte_eth_tx_queue_setup(uint16_t port_id, uint16_t tx_queue_id, uint16_t nb_tx_desc, unsigned int socket_id, const struct rte_eth_rxconf *tx_conf)	This API allocates and set up a receive queue for DMA device.	port ID: Port ID of device tx_queue_id: Queue ID nb_tx_desc: Number of Tx descriptors to allocate for the transmit ring socket_id: socket identifier tx_conf: TX configuration context	0 on success negative otherwise

Table 132. rte_eth_rx_queue_setup

API	API Description	Input Parameters	Return Values
int rte_eth_rx_queue_setup(uint16_t port_id, uint16_t rx_queue_id, uint16_t nb_rx_desc, unsigned int socket_id, const struct rte_eth_rxconf *rx_conf, struct rte_mempool *mp)	This API allocates and set up a receive queue for DMA device.	port ID: Port ID of device rx_queue_id: Queue ID nb_rx_desc: Number of Rx descriptors to allocate for the receive ring socket_id: socket identifier rx_conf: RX configuration context mp: Pointer to the memory pool from which is used to allocate memory buffers for each descriptor of the receive ring	0 on success negative otherwise

Table 133. rte_eth_dev_set_mtu

API	API Description	Input Parameters	Return Values
int rte_eth_dev_set_mtu(uint16_t port_id, uint16_t mtu)	This API sets the payload value for the processing.	port ID: Port ID of device mtu: MTU to be applied	0 on success negative otherwise

Table 134. rte_eth_dev_start

API	API Description	Input Parameters	Return Values
int rte_eth_dev_start(uint16_t port_id)	This API starts the Ethernet device by initializing descriptors, QCSR Rx and Tx context.	port ID: Port ID of device	0 on success negative otherwise

Table 135. rte_eth_dev_stop

API	API Description	Input Parameters	Return Values
void rte_eth_dev_stop(uint16_t port_id)	This API stops the Ethernet device.	port ID: Port ID of device	void

Table 136. rte_eth_dev_close

API	API Description	Input Parameters	Return Values
Void rte_eth_dev_close(uint16_t port_id)	This API closes the Ethernet device.	port ID: Port ID of device	void

Table 137. rte_eth_tx_burst

API	API Description	Input Parameters	Return Values
static inline uint16_t rte_eth_tx_burst (uint16_t port_id, uint16_t queue_id, struct rte_mbuf **tx_pkts, const uint16_t nb_pkts)	This API is used to transmit burst of packets.	port ID: Port ID of device queue_id: Queue ID Queue ID tx_pkts: Array of pointers to *rte_mbuf* structures nb_pkts: Maximum number of packets to retrieve	The number of output packets actually stored in transmit descriptors.

Table 138. rte_eth_rx_burst

API	API Description	Input Parameters	Return Values
static inline uint16_t rte_eth_rx_burst (uint16_t port_id, uint16_t queue_id, struct rte_mbuf **rx_pkts, const uint16_t nb_pkts)	This API is used to receives burst of packets.	port ID: Port ID of device queue_id: Queue ID rx_pkts: Array of pointers to *rte_mbuf* structures nb_pkts: Maximum number of packets to retrieve	The number of packets actually retrieved.

Table 139. ifc_mcdma_pio_read64

API	API Description	Input Parameters	Return Values
Uint64_t ifc_mcdma_pio_read64(stru ct ifc_mcdma_device *qdev, uint64_t addr); - New	Read 64b value from BAR2 address This API is used for PIO testing, dumping statistics, pattern generation etc.	qdev: MCDMA device addr: address to read	0 on success negative otherwise

Table 140. ifc_mcdma_pio_write64

API	API Description	Input Parameters	Return Values
void ifc_mcdma_pio_write64(stru ct ifc_mcdma_device *qdev, uint64_t addr, uint64_t val) - New	Writes 64 bit value to BAR2 address	qdev: MCDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

Table 141. ifc_mcdma_pio_read128

API	API Description	Input Parameters	Return Values
int ifc_mcdma_pio_read128(uin t16_t portid, uint64_t offset, uint64_t *buf, int bar_num) – New	Read 128 bit value from specified address of specified BAR This API is used for PIO testing etc.	qdev: MCDMA device addr: address to read	0 on success negative otherwise

Table 142. ifc_mcdma_pio_write128

API	API Description	Input Parameters	Return Values
int ifc_mcdma_pio_write128(uin t16_t portid, uint64_t offset, uint64_t *val, int bar_num) – New	Writes 128 bit value to specified address of specified BAR. This API is used for PIO testing etc.	qdev: MCDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

Table 143. ifc_mcdma_pio_read256

API	API Description	Input Parameters	Return Values
int ifc_mcdma_pio_read256(uin t16_t portid, uint64_t offset, uint64_t *buf, int bar_num) – New	Read 256 bit value from specified address of specified BAR This API is used for PIO testing etc.	qdev: MCDMA device addr: address to read	0 on success negative otherwise

Table 144. ifc_mcdma_pio_write256

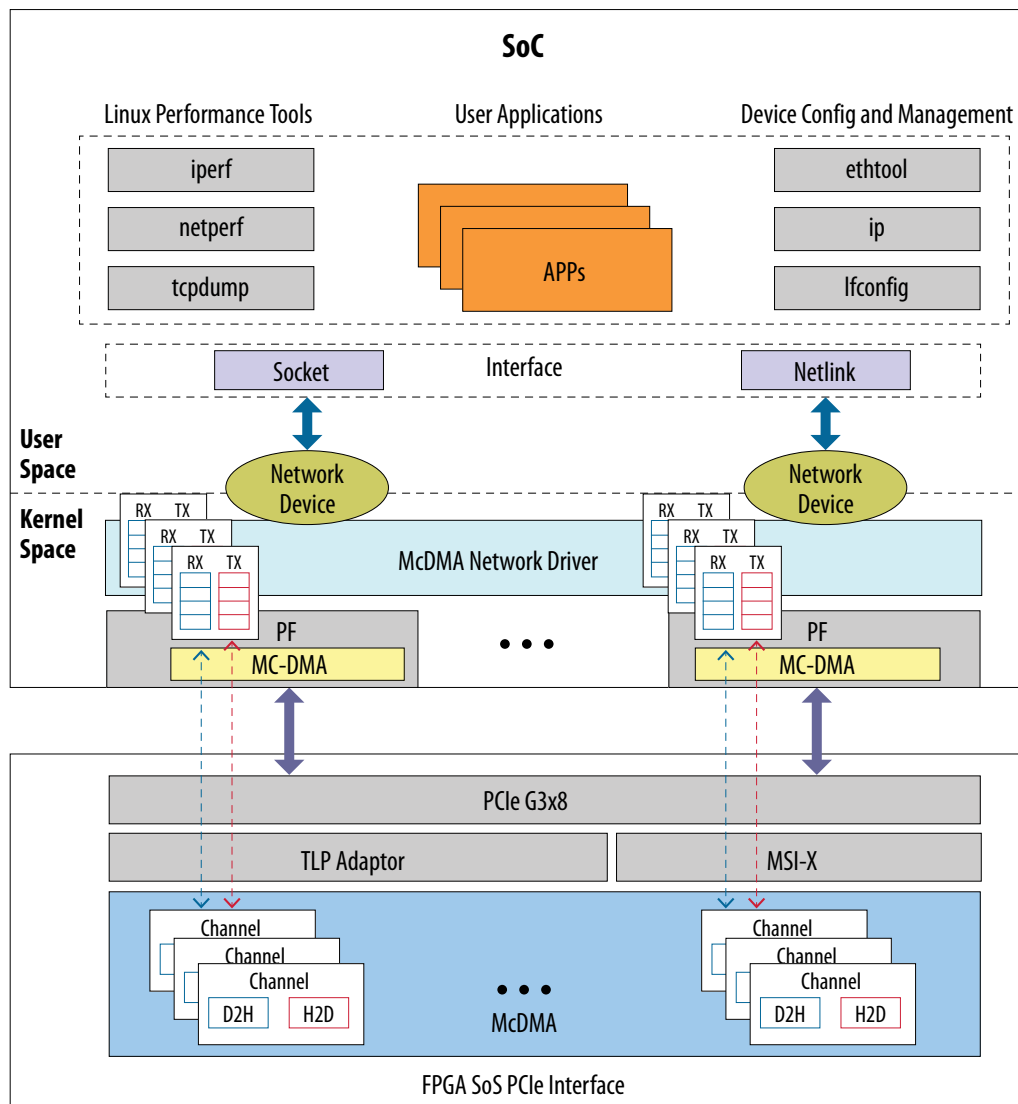
API	API Description	Input Parameters	Return Values
int ifc_mcdma_pio_write512(uin t16_t portid, uint64_t offset, uint64_t *val, int bar_num) – New	Writes 128 bit value to specified address of specified BAR This API is used for PIO testing etc.	qdev: MCDMA device addr: address to write val: value to write	0 on success and populates channel context negative otherwise

8.3. Multi Channel DMA IP Kernel Mode Network Device Driver

Note: The Kernel mode network device driver is a preliminary release. Production version may be available in future release.

8.3.1. Architecture

Figure 54. MCDMA IP Kernel Mode Network Device Driver Architecture



Network driver expose the device as ethernet interface. Following are the different components involved in this architecture.

ifc_mcdma_netdev: This driver is responsible for device management, channel management, interrupt management and enables the transmission between the network based applications and hardware.

ethtool, ip, ifconfig are the utilities which are a part of the kernel tree and are used to configure and manage the device.

iperf, netperf, iperf3 are opensource applications, that typically are used to verify the performance of network based applications.

8.3.2. Driver Information

ifc_mcdma_netdev kernel module scans the devices to identify MCDMA device based on vendor ID and device ID, enables the bus mastership and maps the BAR area of corresponding devices by using existing Linux PCI framework. The driver creates the Ethernet Interface and registers the device to the Linux network framework. Kernel Driver currently supports four Physical Functions simultaneously. Each PF supports 1 channel up to 512 channels as per the device configuration. If device supports SRIOV, netdev allows to enable the virtual functions and run DMA from VF devices.

8.3.2.1. Device Management

```
static const struct net_device_ops ifc_mcdma_netdev_ops = {
    .ndo_open          = ifc_mcdma_netdev_open,
    .ndo_stop          = ifc_mcdma_netdev_close,
    .ndo_start_xmit     = ifc_mcdma_netdev_xmit,
#ifdef IFC_SELECT_QUEUE_ALGO
    .ndo_select_queue   = ifc_mcdma_select_queue,
#endif
#ifdef RHEL_RELEASE_CODE
    .ndo_change_mtu_rh74 = ifc_mcdma_change_mtu,
#else
    .ndo_change_mtu      = ifc_mcdma_change_mtu,
#endif
    .ndo_validate_addr  = eth_validate_addr,
    .ndo_set_mac_address = eth_mac_addr,
    .ndo_get_stats       = ifc_mcdma_netdev_get_stats,
    .ndo_do_ioctl        = ifc_mcdma_netdev_ioctl,
}
```

ifc_mcdma_netdev driver supports the ethtool and ifconfig and ip utilities to configure and manage the device.

IP Reset

IP reset is performed through the **sysfs** attribute file mcdma_ipreset.

Following command performs the IP reset:

```
echo 1 > /sys/bus/pci/devices/<bdf>/mcdma_ipreset
```

ifconfig support

By using ifconfig, the driver supports bring-down and bring-up of the device. To support these operations, the driver overrides ndo_open and ndo_stop operations of the device.

Bring-down of the device

When you bring down the device by using ifconfig command, the kernel changes the state of the device to DOWN state and executes the registered call back. As a result of the callback functionality, the driver stops the TX queue, disables the interrupts and releases the acquired channels and all the internal resources allocated to the device.

Command for bring-down of the device example:

```
ifconfig ifc_mcdma0 down
```

Bring-up of the device

When you bring up the device by using `ifconfig` command, the kernel changes the state of the device to "UP" state and executes the registered call back. As a result of the callback functionality, the driver starts the TX queue, acquires and enables channels and corresponding interrupts.

Command for bring-up of the device example:

```
ifconfig ifc_mcdma0 up
```

8.3.2.2. Channel Management

Each network device associated with one physical or virtual function and can support up to 512 channels. As part of network device (`ifc_mcdma<i>`) bringup, all channels of that device are initialized and enabled. Whenever a packet arrives from application, one of the Tx queues are selected to transfer it.

Strategies for Queue Selection

1. **Linux default queue selection:** In this case, queue is selected based on logic provided by Linux multi-queue support feature.
2. **XPS (Transmit Packet Steering):** This technique is part of the Linux kernel & provides a mechanism to map multiple cores to a Tx queue of the device. For all packets coming from any of these cores, the mapped Tx queue will be used for transfer. For more information, refer to [XPS: Transmit Packet Steering](#).
3. **MCDMA custom queue selection:** This technique provides a mechanism to map multiple queues to each core. For each core, a separate list is managed to keep track of every flow of transfer coming to the core. This is done using 4 tuple hash over IP and TCP addresses of each packet and the Tx the queue allocated for that flow.

Upon receipt of a Tx packet from the upper layers, a lookup on this table is done using the hash of that packet. If a match is found, the corresponding queue is used for transfer. Otherwise, a new queue is allocated to this new flow.

8.3.2.3. Descriptor Memory Management

As a part of channel initialization, the driver allocates the memory for descriptors and associates to the channel. Driver uses `dma_alloc_coherent` API of Linux DMA framework to allocate non-swappable and physically contiguous memory.

- By default, currently 1 page is enabled. 1 page contains 128 descriptors. By using `ethtool`, `netdev` supports changing the queue size.
- Each queue (H2D & D2H) of each channel gets its descriptor memory.
- After allocation of this memory, the hardware is notified of it by a write of the starting address to the QCSR region.

8.3.2.4. Completions Management

The kernel module and the hardware supports MSI-X interrupt mechanism as descriptor process completion indication. At queue initialization, the device enables the interrupts in the kernel by using the interrupts framework.

8.3.2.5. ethtool support

Following ethtool operations are supported by netdev kernel mode driver. Implementation added for channel management and ring management.

```
const struct ethtool_ops ifc_mcdma_ethtool_ops = {
    .get_msglevel      = ifc_mcdma_get_msglevel,
    .set_msglevel      = ifc_mcdma_set_msglevel,
    .get_drvinfo       = ifc_mcdma_get_drvinfo,
    .get_ts_info       = ethtool_op_get_ts_info,
    .get_link          = ethtool_op_get_link,
    .get_channels       = ifc_mcdma_get_channels,
    .set_channels       = ifc_mcdma_set_channels,
    .get_ringparam      = ifc_mcdma_get_ringparam,
    .set_ringparam      = ifc_mcdma_set_ringparam,
};
```

8.3.2.6. debugfs support

Debugfs capabilities added to network driver to support debuggability. In `/sys/kernel/debug`, separate directories for each interface are created and channel specific statistics are added.

Debugfs directories get created, when the device and channel specific files are created while initializing the channels.

8.3.2.7. SRIOV Support

VF devices creation

Netdev driver supports enabling the interrupts and perform the DMA from virtual functions. Netdev driver registers SRIOV callback and reuses the sysfs infrastructure created by the kernel's infrastructure.

```
static struct pci_driver ifc_mcdma_netdev_pci_driver = {
    .name = "ifc_mcdma_netdev",
    .sriov_configure = ifc_mcdma_sriov_configure,
};
```

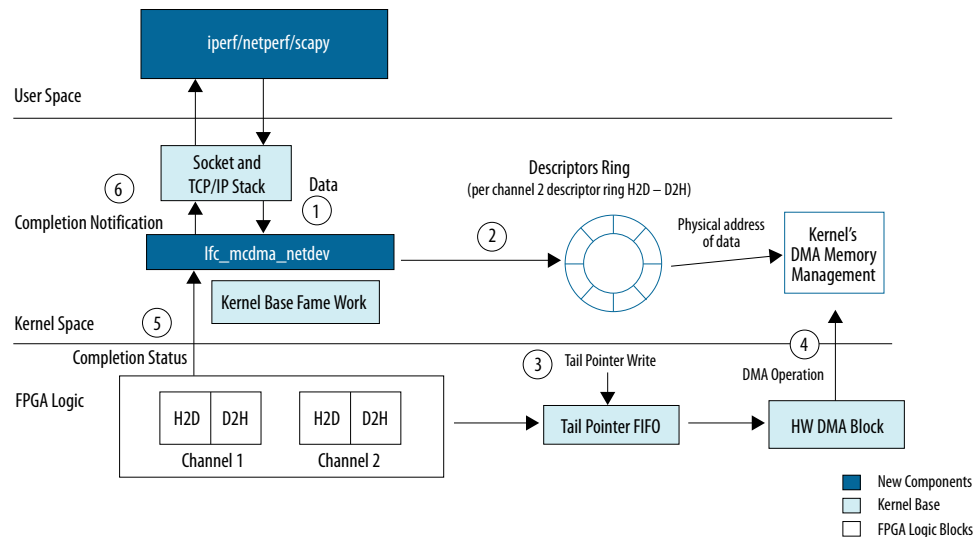
IOMMU Support

If host is supporting IOMMU and is enabled from boot parameters, the Netdev driver maps the device memory to IOMMU by using API called `dma_map_single` and configure the I/O Virtual address in descriptor. This enables protection to the host from the attacks attempted by malicious or unsecured devices. If IOMMU is disabled, netdev configures the physical address provided by MMU in host.

8.3.3. Software Flow

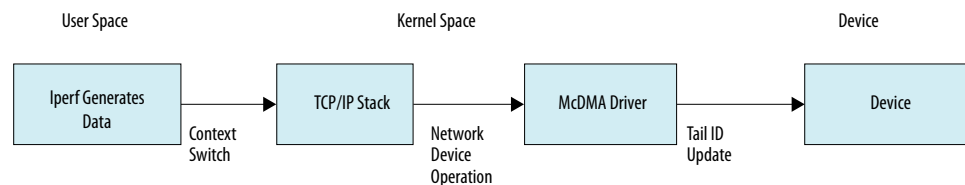
The software flow when the user space application tries to access the DMA in hardware is shown in the figure below

Figure 55. MCDMA IP Kernel Mode Network Device Driver : Software Flow



8.3.3.1. Host to Device Flow

Figure 56. H2D Flow



When user space application attempts to send one packet to network device

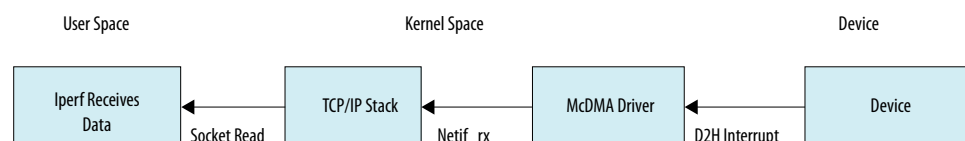
1. Application generates the data and the data can be copied to the kernel.
2. TCP/IP Stack creates the skb and calls the transmit handler registered by `ndo_start_xmit` callback overridden by MCDMA network driver.
3. Driver retrieves the physical address or I/O Virtual address,, loads the descriptor and submits the DMA transactions.

When descriptor processing is completed

1. Hardware completes the transaction and notifies the host via an interrupt.
2. MCDMA driver receives the completion and frees the allocated skb.

8.3.3.2. Device to Host Flow

Figure 57. D2H Flow



When user space application attempts to send one packet to network device

1. At queue initialization, the driver prefills all the descriptors.
2. When a completion is received from the user, the driver determines the packet's protocol ID by using `eth_type_trans` call
3. Driver submits the corresponding data packet to the TCP/IP Stack.
4. Refills the ring to accommodate the incoming packets.

9. Registers

The Multi Channel 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)

These Multi Channel DMA registers are mapped to BAR0 of a function.

Note: GCSR is only for PF0.

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

Following table shows 4 MB aperture space mapped for PF0 in PCIe config space through BAR0.

Table 145. Multi Channel 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 2048 D2H and 2048 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

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

Table 146. 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

continued...

Address Space Name	Size	DMA Channel	Size	Description
		DMA Channel 1	256 B	QCSR for DMA channel 1
	
		DMA Channel N	256 B	QCSR for DMA channel 2

9.1. Queue Control (QCSR)

QCSR space contains queue control and status information. This register space of 1 MB can support up to 2048 H2D and 2048 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 H2D/D2H queues. The base address for H2D and D2H are different, but registers (H2D and D2H) has the same address offsets.

Table 147. 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-bit of 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-bit of 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 max 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 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.
continued...			

Register Name	Address Offset	Access Type	Description
Q_CONSUMED_HEAD_ADDR_L	8'h20	R/W	Lower 32-bit 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-bit 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 delayvalue in this register to maximize fetching efficiency.
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. Max 1 MB, with 20'h0 indicating 1 MB. The value set in this register must be the same as used by Host SW to populate the PYLD_CNT field of descriptors for the respective channel. Applicable only for D2H AVST 1 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 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-bit of address.

Table 148. Q_CTRL (Offset 8'h00)

Bit [31:0]	Name	R/W	Default	Description
[31:10]	rsvd			Reserved
[9]	q_intr_en	R/W	0	If set, upon completion generate a MSI-X interrupt.
continued...				

Bit [31:0]	Name	R/W	Default	Description
[8]	q_wb_en	R/W	0	If set, upon completion, do a write back.
[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 149. Q_START_ADDR_L (Offset 8'h08)

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

Table 150. Q_START_ADDR_H (Offset 8'h0C)

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

Table 151. Q_SIZE (Offset 8'h10)

Bit [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 max 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}).
<i>continued...</i>				

Bit [31:0]	Name	R/W	Default	Description
				<i>Note:</i> When Q_SIZE is set to 0x10, the maximum valid value for Q_TAIL_POINTER should be 0xFFFF, as it is defined as a 16-bit parameter. Any value beyond this, such as 0x10000, is considered invalid.

Table 152. Q_TAIL_POINTER (Offset 8'h14)

Bit [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 upto this position of the buffer. <i>Note:</i> Writing 0x0 to Q_TAIL_POINTER is illegal.

Table 153. Q_HEAD_POINTER (Offset 8'h18)

Bit [31:0]	Name	R/W	Default	Description
[31:25]	rsvd			Reserved
[24]	q_err_during_desc_fetch	R	0	Indicates a received UR response OR a Completion time out for a Descriptor Read request (for both H2D and D2H).
[23:16]	rsvd			Reserved
[15:0]	q_hd_ptr	R/W	0	After 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 pointer is not equal.

Table 154. Q_COMPLETED_POINTER (Offset 8'h1C)

Bit [31:0]	Name	R/W	Default	Description
[31:25]	rsvd			Reserved
[24]	q_err_during_data_fetch	R	0	Indicates a received UR response OR a Completion time out for a Descriptor Read request (for H2D only). This bit is reserved for D2H.
[23:16]	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, that is all data for that descriptor and previous descriptors have arrived at the intended destinations. Software can poll this register to find out the status of the DMA for a specific queue.

Table 155. Q_CONSUMED_HEAD_ADDR_L (Offset 8'h20)

Bit [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 156. Q_CONSUMED_HEAD_ADDR_H (Offset 8'h24)

Bit [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 157. Q_BATCH_DELAY (Offset 8'h28)

Bit [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
continued...				

Bit [31:0]	Name	R/W	Default	Description
				between fetches for descriptors. Each unit is 2ns.

Table 158. Q_DATA_DRP_ERR_CTR (Offset 8'h40)

Note: The register Q_DATA_DRP_ERR_CTR is defined for D2H only and reserved for H2D.

Bit [31:0]	Name	R/W	Default	Description
[31:21]	rsvd	-	-	-
[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	-	-	-
[18]	rsvd	-	-	-
[17]	q_data_err_mm	R/W	0	Data error status for a channel in AVMM mode
[16]	q_data_err_st	R/W	0	Data error status for a channel in AVST mode
[15:0]	q_data_drp_err_cnt	R/W	0	<p>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 1s and needs software to clear.</p> <p>Data drop example cases:</p> <ul style="list-style-type: none"> • Packets received for a channel that is not enabled • Packets received for a channel that is enabled, but does not have TID update <p><i>Note:</i> 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.</p>

Table 159. Q_PYLD_CNT register (offset 8'h44)

Bit [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. Max 1 MB, with 20'h0 indicating 1 MB. This value has to be same as set in the descriptors payload count field. Applicable only for D2H AVST 1 port mode. Unused in all other modes

Table 160. Q_RESET (Offset 8'h48)

Bit [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 value of 1'b0 when reset has been completed by hardware. Hardware clears this bit after completing the reset of a queue.

9.2. MSI-X Memory Space

The MSI-X Table and PBA memory is mapped to the second MB space of the Register address space. Allocated memory space can support up to 2048 MSI-X interrupts for a function. Actual amount of memory depends on the Multi Channel DMA IP for PCI Express 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. To support 2048 interrupts, MSI-X Table requires 32 KB of space per function. But it is mapped to a 512 KB of space.

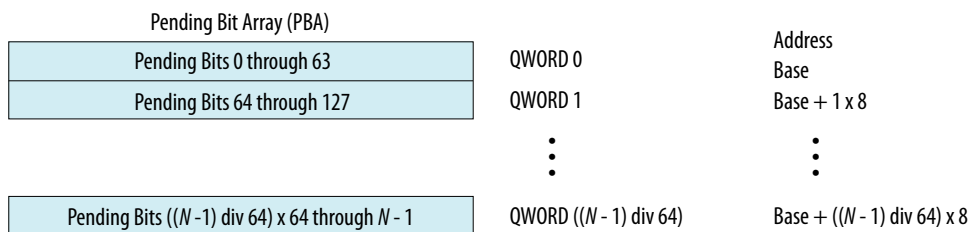
Figure 58. MSI-X Table Structure

DWORD 3	DWORD 2	DWORD 1	DWORD 0		Host Byte Addresses
Vector Control	Message Data	Message Upper Address	Message Address	Entry 0	Base
Vector Control	Message Data	Message Upper Address	Message Address	Entry 1	Base + 1 x 16
Vector Control	Message Data	Message Upper Address	Message Address	Entry 2	Base + 2 x 16
⋮	⋮	⋮	⋮	⋮	⋮
Vector Control	Message Data	Message Upper Address	Message Address	Entry (N - 1)	Base + (N - 1) x 16

MSI-X PBA

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

Figure 59. 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

9.3. 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 161. Control Register

Register Name	Address Offset	Access Type	Description
CTRL	8'h00	R/W	Reserved
RESERVED	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	Multi Channel DMA IP for PCI Express version number
SW_RESET	9'h120	RW	Write this register to issue Multi Channel DMA IP reset without disturbing PCI Express link. This resets all queues and erase all the context. Can be issued only from PF0.

Table 162. CTRL (Offset 8'h0)

Bit [31:0]	Name	R/W	Default	Description
[31:0]	rsvd			Reserved

Table 163. WB_INTR_DELAY (Offset 8'h08)

Bit [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 164. VER_NUM (Offset 9'h070)

Bit [31:0]	Name	R/W	Default	Description
[31:24]	rsvd			RESERVED
[23:16]	MAJOR_VER	RO	0	Major version number of Multi Channel DMA IP for PCI Express
[15:8]	UPDATE_VER	RO	0	Update version number of Multi Channel DMA IP for PCI Express
[7:0]	PATCH_VER	RO	0	Patch version number of Multi Channel DMA IP for PCI Express

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

Table 165. SW_RESET (Offset 9'h120)

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



10. Troubleshooting/Debugging

10.1. Debug Toolkit

10.1.1. Overview

The Debug Toolkit is a System Console-based tool that provides real-time control, monitoring and debugging of the PCIe links at the Physical Layer.

The Debug Toolkit allows you to:

- View protocol and link status of the PCIe links.
- View PLL and per-channel status of the PCIe links.
- View the channel analog settings.
- View the receiver eye and measure the eye height and width for each channel.
- Indicate the presence of a re-timer connected between the link partners.

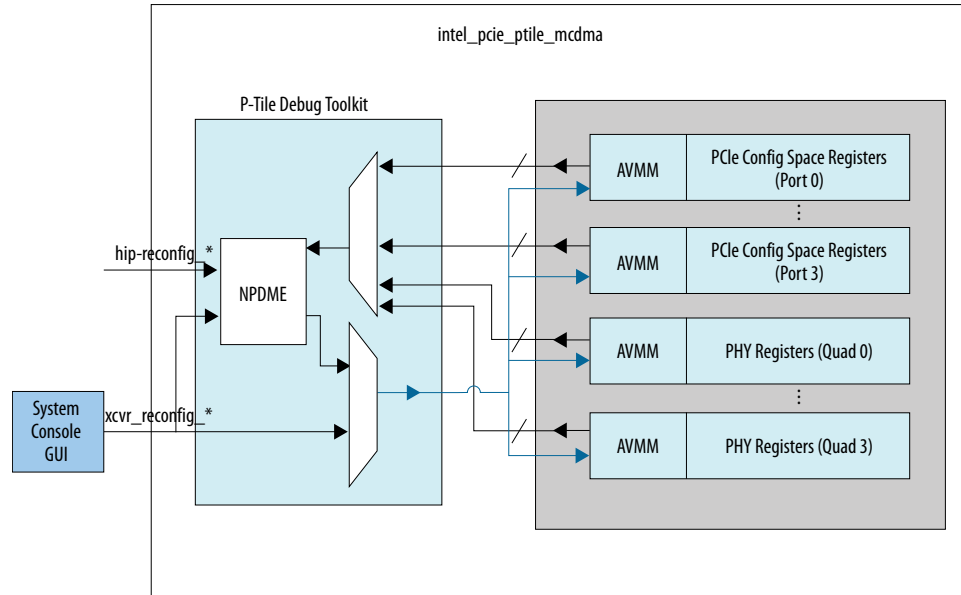
Note: Unless or otherwise noted, the features described in this chapter apply to P-Tile, F-Tile and R-Tile versions of MCDMA IP.

Note: The current version of Quartus Prime supports enabling the Debug Toolkit for Endpoint mode only and with the Linux and Windows operating systems only.

Note: Debug Toolkit is not verified in Root Port mode for MCDMA IP

The following figure provides an overview of the Debug Toolkit in the Multi Channel DMA IP for PCI Express.

Figure 60. Overview of the Debug Toolkit



When you enable the Debug Toolkit, the `intel_pcie_ptile_mcdma` or `intel_pcie_ftile_mcdma` or `intel_pcie_rtile_mcdma` module of the generated IP includes the Debug Toolkit modules and related logic as shown in the figure above.

Drive the Debug Toolkit from a System Console. The System Console connects to the Debug Toolkit via an Native PHY Debug Master Endpoint (NPDME). Make this connection via an Intel FPGA Download Cable.

The PHY reconfiguration interface clock (`xcvr_reconfig_clk`) is used to clock the following interfaces:

- The NPDME module
- PHY reconfiguration interface (`xcvr_reconfig`)
- Hard IP reconfiguration interface (`hip_reconfig`)

Provide a clock source (50 MHz - 125 MHz, 100 MHz recommended clock frequency) to drive the `xcvr_reconfig_clk` clock. Use the output of the Reset Release Intel FPGA IP to drive the `ninit_done`, which provides the reset signal to the NPDME module.

Note: When you enable the Debug Toolkit, the Hard IP Reconfiguration interface is enabled by default.

When you run a dynamically-generated design example on the Intel Development Kit, make sure that clock and reset signals are connected to their respective sources and appropriate pin assignments are made. Here is a sample `.qsf` assignments for the Debug Toolkit.

For P-Tile using Stratix 10 DX FPGA Development Kit:

- `set_location_assignment PIN_C23 -to xcvr_reconfig_clk_clk`

For F-Tile, these statements below are needed in .qsf for the Debug Toolkit launch

```
set_location_assignment PIN_CK18 -to xcvr_reconfig_clk_clk  
set_instance_assignment -name IO_STANDARD "TRUE DIFFERENTIAL SIGNALING" -to  
xcvr_reconfig_clk_clk -entity pcie_ed
```

For R-Tile:

```
set_location_assignment PIN_AN61 -to xcvr_reconfig_clk_clk  
set_instance_assignment -name IO_STANDARD "TRUE DIFFERENTIAL SIGNALING" -to  
xcvr_reconfig_clk_clk -entity pcie_ed
```

10.1.2. Enabling the Debug Toolkit

To enable the Debug Toolkit in your design, enable the option **Enable Debug Toolkit** in the **Top Level Settings** tab of the Multi Channel DMA IP for PCI Express.

A multiplexer is implemented to allow dynamic switching between the user AVMM reconfiguration interface and the System Console-based Debug Toolkit, when the Debug Toolkit is enabled. This allows you to switch between the user logic driving the reconfiguration interface and the Debug Toolkit, as both access the same set of registers within the Hard IP.

The user AVMM reconfiguration interface has default access (the default is when `toolkit_mode = 0`). Upon launching the Debug Toolkit (DTK) from System Console, `toolkit_mode` is automatically set to 1 for DTK access. While the Debug Toolkit is open in System Console, user logic is not able to drive the signals on the user AVMM interface as the multiplexer is set to `toolkit_mode = 1`. Upon exiting (closing) the Debug Toolkit window in System Console, `toolkit_mode` is automatically set to 0 for user access.

The Debug Toolkit can be launched successfully only if pending read/write transactions on the reconfiguration interface are completed (indicated by the deassertion of the `reconfig_waitrequest` signal).

Note: Upon being launched from System Console, the Debug Toolkit first checks if any of the `waitrequest` signals from the Hard IP are asserted (i.e. if there is an ongoing request from the user). The System Console message window shows an error message to let the user know there is an ongoing request and the Debug Toolkit cannot be launched.

10.1.3. Launching the Debug Toolkit

Use the design example you compiled by following the *Quick Start Guide* to familiarize yourself with the Debug Toolkit. Follow the steps in the *Generating the Design Example* and *Compiling the Design Example* to generate the SRAM Object File, (.sof) for this design example.

To use the Debug Toolkit, download the .sof to the Intel Development Kit. Then, open the System Console and load the design to the System Console as well. Loading the .sof to the System Console allows the System Console to communicate with the design using NPDME. NPDME is a JTAG-based Avalon-MM master. It drives Avalon-MM slave interfaces in the PCIe design. When using NPDME, the Quartus Prime software inserts the debug interconnect fabric to connect with JTAG.

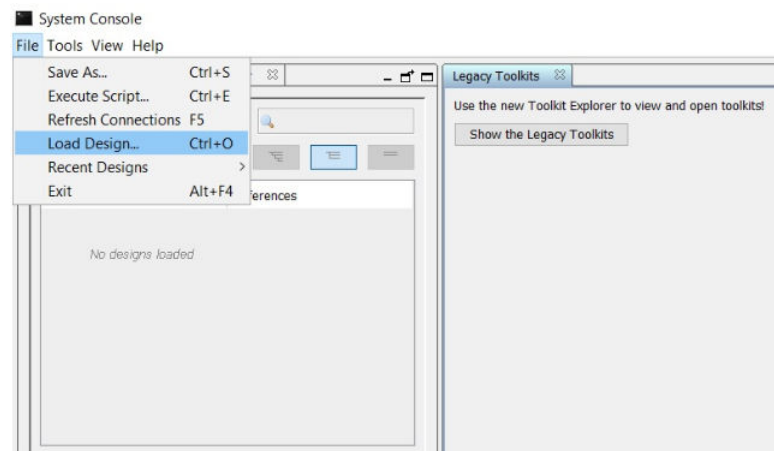
Here are the steps to complete these tasks:

1. Use the Quartus Prime Programmer to download the .sof to the Intel FPGA Development Kit.

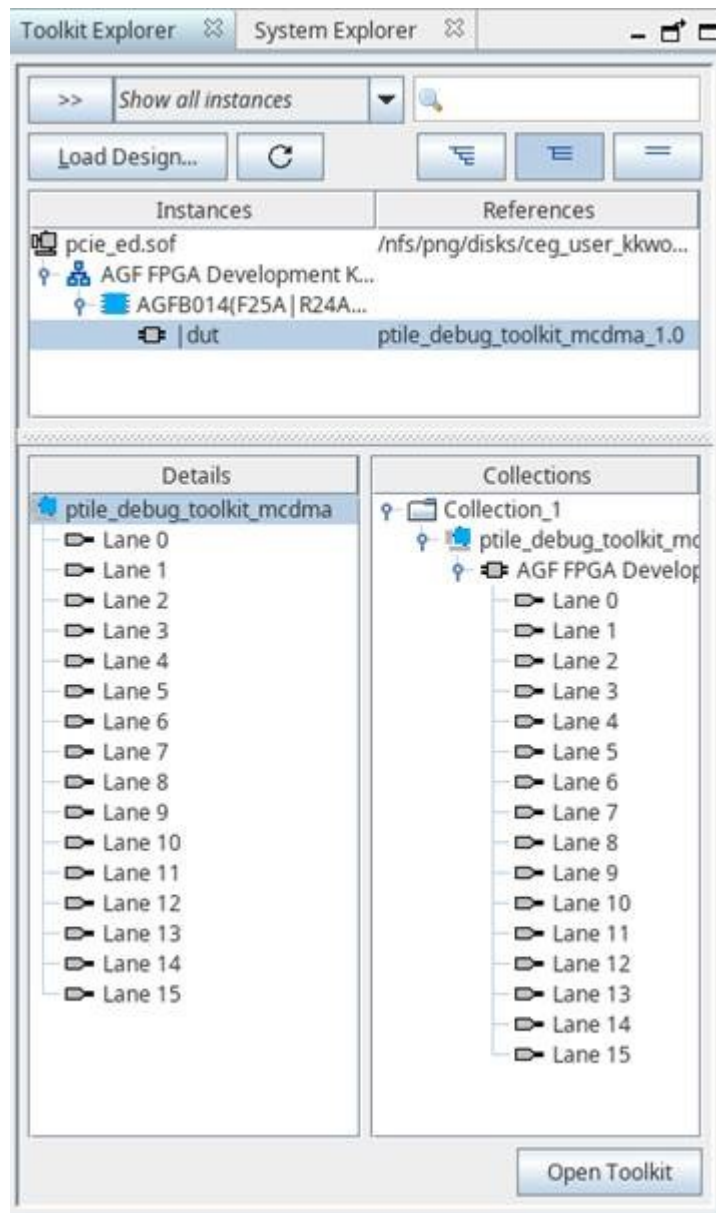
Note: To ensure correct operation, use a full installation of the Quartus Prime Pro Edition Software and Devices of the same version of the Quartus Prime Programmer and Quartus Prime Pro Edition software that you used to generate the .sof.

Note: A standalone install of the Quartus Prime Pro Edition Programmer and Tools will not work.

2. To load the design into System Console:
 - a. Launch the Quartus Prime Pro Edition software.
 - b. Start System Console by choosing **Tools**, then **System Debugging Tools**, then **System Console**.
 - c. On the System Console File menu, select **Load design** and browse to the .sof file.



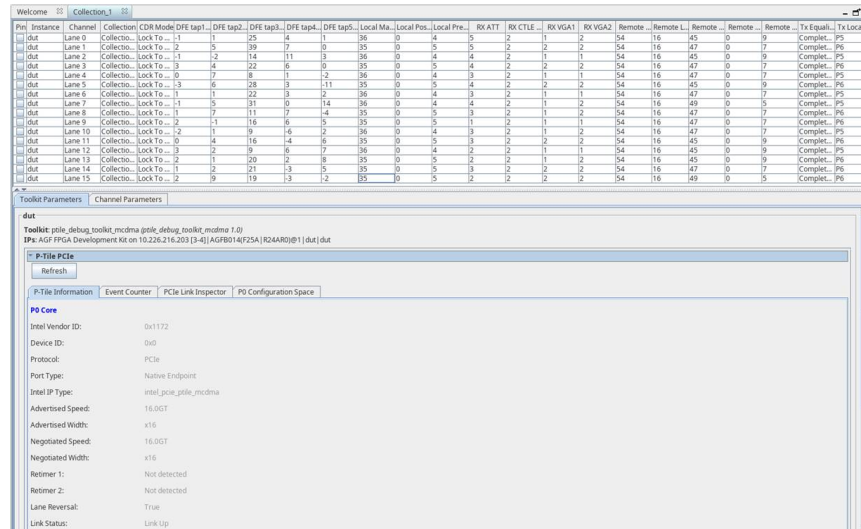
- d. Select the .sof and click **OK**. The .sof loads to the System Console.
3. The System Console Toolkit Explorer window will list all the DUTs in the design that have the Debug Toolkit enabled.
 - a. Select the DUT with the Debug Toolkit you want to view. This will open the Debug Toolkit instance of that DUT in the **Details** window. An example of P-Tile Debug Toolkit is shown in the following figure.



- b. Click on the **ptile_debug_toolkit_mcdma** to open that instance of the Toolkit. Once the Debug Toolkit is initialized and loaded, you will see the following message in the **Messages** window: **"Initializing P-Tile debug toolkit – done"**.



c. A new window **Main view** will open with a view of all the channels in that instance.



10.1.4. Using the Debug Toolkit

The following sections describe the different tabs and features available in the Debug Toolkit.

10.1.4.1. Main View

The main view tab lists a summary of the transmitter and receiver settings per channel for the given instance of the PCIe IP.

Collection 1																						
Pin	Instance	Channel	CDR Mode	DFE tap1...	DFE tap2...	DFE tap3...	DFE tap4...	DFE tap5...	Local Ma...	Local Pos...	Local Pre...	Rx ATT	Rx CTLE	Rx VGA1	Rx VGA2	Remote	Remote L...	Remote	Remote	Remote	Tx Equal...	Tx Local
dut	Lane 0	Collectio...	Lock To ...	-1	1	25	4	1	36	0	4	5	2	1	2	54	16	45	0	9	Comple...	P5
dut	Lane 1	Collectio...	Lock To ...	-2	5	19	7	0	35	0	5	5	2	2	2	54	16	47	0	7	Comple...	P6
dut	Lane 2	Collectio...	Lock To ...	-1	-2	14	11	3	36	0	4	4	2	1	1	54	16	45	0	9	Comple...	P5
dut	Lane 3	Collectio...	Lock To ...	-3	4	22	6	0	35	0	5	4	2	2	2	54	16	47	0	7	Comple...	P6
dut	Lane 4	Collectio...	Lock To ...	-0	7	8	1	-2	36	0	4	3	2	1	1	54	16	47	0	7	Comple...	P5
dut	Lane 5	Collectio...	Lock To ...	-3	6	28	3	-11	35	0	5	4	2	2	2	54	16	45	0	9	Comple...	P6
dut	Lane 6	Collectio...	Lock To ...	-1	1	22	3	2	36	0	4	3	2	1	1	54	16	47	0	7	Comple...	P5
dut	Lane 7	Collectio...	Lock To ...	-1	7	11	7	-4	35	0	5	3	2	1	2	54	16	49	0	5	Comple...	P5
dut	Lane 8	Collectio...	Lock To ...	-2	-1	16	6	5	35	0	5	1	2	1	1	54	16	47	0	7	Comple...	P6
dut	Lane 9	Collectio...	Lock To ...	-2	1	9	-4	2	36	0	4	3	2	1	2	54	16	47	0	7	Comple...	P5
dut	Lane 10	Collectio...	Lock To ...	-0	4	16	-4	6	35	0	5	3	2	2	2	54	16	45	0	9	Comple...	P6
dut	Lane 11	Collectio...	Lock To ...	-0	4	16	-4	6	35	0	5	3	2	1	1	54	16	45	0	9	Comple...	P5
dut	Lane 12	Collectio...	Lock To ...	-3	2	9	6	7	36	0	4	2	2	1	1	54	16	45	0	9	Comple...	P5
dut	Lane 13	Collectio...	Lock To ...	-2	1	20	2	8	35	0	5	2	2	1	2	54	16	45	0	9	Comple...	P6
dut	Lane 14	Collectio...	Lock To ...	-1	2	21	-3	5	35	0	5	3	2	2	2	54	16	47	0	7	Comple...	P6
dut	Lane 15	Collectio...	Lock To ...	-2	9	19	-3	-2	35	0	5	2	2	2	2	54	16	49	0	5	Comple...	P6

The following table shows the channel mapping when using subdivided ports.

Table 166. Channel Mapping for Subdivided Ports

Toolkit Channel	x16 Mode	x8 Mode	x4 Mode	2x8 Mode
Lane 0	Lane 0	Lane 0	Lane 0	Lane 0
Lane 1	Lane 1	Lane 1	Lane 1	Lane 1
Lane 2	Lane 2	Lane 2	Lane 2	Lane 2
Lane 3	Lane 3	Lane 3	Lane 3	Lane 3
Lane 4	Lane 4	Lane 4	N/A	Lane 4
Lane 5	Lane 5	Lane 5	N/A	Lane 5
Lane 6	Lane 6	Lane 6	N/A	Lane 6
Lane 7	Lane 7	Lane 7	N/A	Lane 7
Lane 8	Lane 8	N/A	N/A	Lane 0
Lane 9	Lane 9	N/A	N/A	Lane 1
Lane 10	Lane 10	N/A	N/A	Lane 2
Lane 11	Lane 11	N/A	N/A	Lane 3
Lane 12	Lane 12	N/A	N/A	Lane 4
Lane 13	Lane 13	N/A	N/A	Lane 5
Lane 14	Lane 14	N/A	N/A	Lane 6
Lane 15	Lane 15	N/A	N/A	Lane 7

10.1.4.2. Toolkit Parameters

The Toolkit parameters window has the following sub-tabs.

Tile Information

This lists a summary of the PCIe IP parameter settings in the PCIe IP Parameter Editor when the IP was generated, as read by the Debug Toolkit when initialized. If you have port subdivision enabled in your design (for example, x8x8), then this tab will populate the information for each core (P0 core, P1 core, etc.).

All the information is read-only.

Use the **Refresh** button to read the settings.

Table 167. Available Parameter Settings

Parameter	Values	Descriptions
Intel Vendor ID	1172	Indicates the Vendor ID as set in the IP Parameter Editor.
Device ID	0	This is a unique identifier for the device that is assigned by the vendor.
Protocol	PCIe	Indicates the Protocol.
Port Type	Root Port, Endpoint ⁽¹⁾	Indicates the Hard IP Port type.

continued...

Parameter	Values	Descriptions
Intel IP Type	intel_pcie_mcdma	Indicates the IP type used.
Advertised speed	8.0GT, 16.0GT	Indicates the advertised speed as configured in the IP Parameter Editor.
Advertised width	x16, x8, x4	Indicates the advertised width as configured in the IP Parameter Editor.
Negotiated speed	2.5GT, 5.0GT, 8.0GT, 16.0GT	Indicates the negotiated speed during link training.
Negotiated width	x16, x8, x4, x2, x1	Indicates the negotiated link width during link training.
Link status	Link up, link down	Indicates if the link (DL) is up or not.
LTSSM State	Refer to the P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide	Indicates the current state of the link.
Lane Reversal	True, False	Indicates if lane reversal happens on the link.
Retimer 1	Detected, not detected	Indicates if a retimer was detected between the Root Port and the Endpoint.
Retimer 2	Detected, not detected	Indicates if a retimer was detected between the Root Port and the Endpoint.
Tx TLP Sequence Number	Hexadecimal value	Indicates the next transmit sequence number for the transmit TLP.
Tx Ack Sequence Timeout	Hexadecimal value	Indicates the ACK sequence number which is updated by receiving ACK/NAK DLLP.
Replay Timer Timeout	Green, Red	Green: no timeout Red: timeout
Malformed TLP Status	Green, Red	Green: no malformed TLP Red: malformed TLP detected
First Malformed TLP Error Pointer	<ul style="list-style-type: none"> AtomicOp address alignment AtomicOp operand AtomicOp byte enable TLP length mismatch Max payload size Message TLP without TC0 Invalid TC Unexpected route bit in Message TLP Unexpected CRS status in Completion TLP Byte enable Memory address 4KB boundary TLP prefix rules Translation request rules 	

continued...

⁽¹⁾ The current version of Quartus Prime supports enabling the Debug Toolkit for Endpoint mode only, and for the Linux and Windows operating systems only.

Parameter	Values	Descriptions
	<ul style="list-style-type: none"> Invalid TLP type Completion rules Application 	
PIPE PhyStatus (For F-Tile debug toolkit only)	0,1	Indicates the PMA and PCS are in reset mode. <ul style="list-style-type: none"> 0: PMA and PCS are out of reset 1: PMA and PCS are in reset

Figure 61. Example of P-Tile Parameter Settings

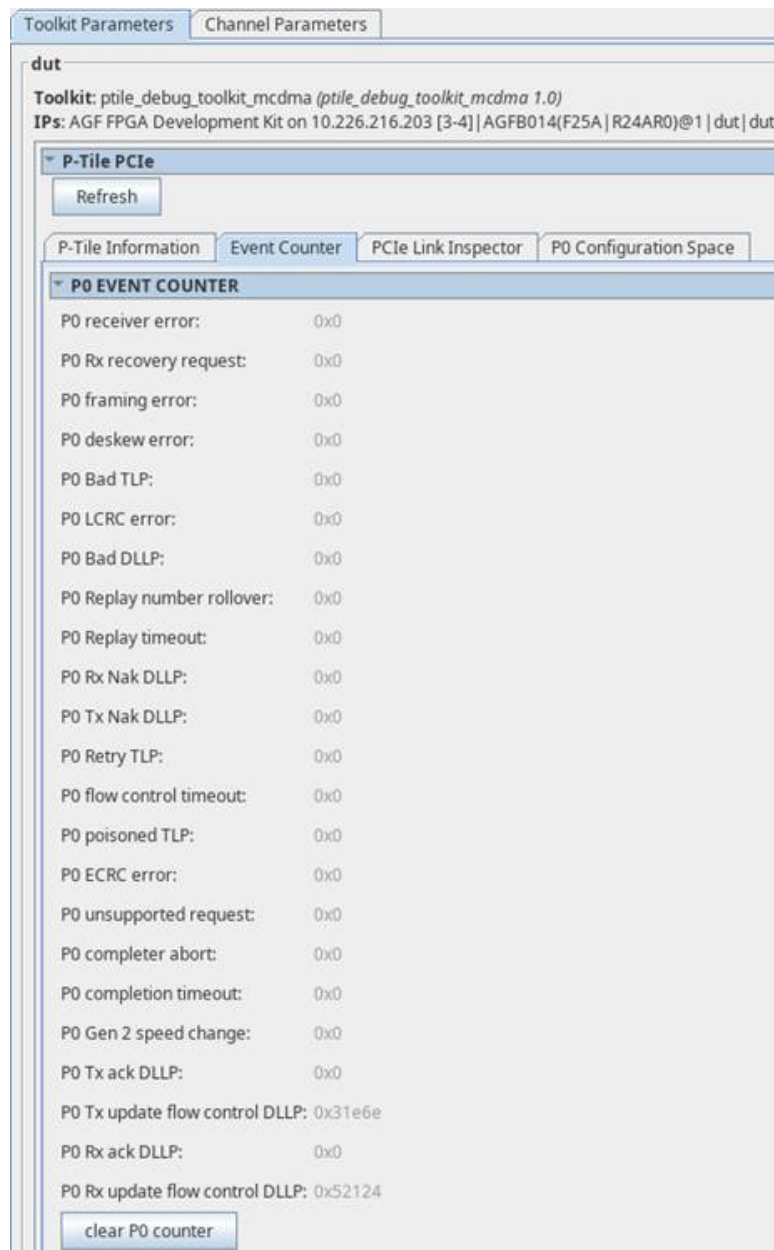
The screenshot displays the 'P-Tile PCIe' configuration window. It includes a 'Refresh' button and four tabs: 'P-Tile Information', 'Event Counter', 'PCIe Link Inspector', and 'P0 Configuration Space'. The 'P-Tile Information' tab is active, showing a list of parameters for the 'P0 Core'.

Parameter	Value
Intel Vendor ID:	0x1172
Device ID:	0x0
Protocol:	PCIe
Port Type:	Native Endpoint
Intel IP Type:	intel_pcie_ptile_mcdma
Advertised Speed:	16.0GT
Advertised Width:	x16
Negotiated Speed:	16.0GT
Negotiated Width:	x16
Retimer 1:	Not detected
Retimer 2:	Not detected
Lane Reversal:	True
Link Status:	Link Up
LTSSM State:	L0
Tx TLP Sequence Number:	0x3dc
Tx Ack Sequence Number:	0x3db
Replay Timer Timeout:	●
Malformed TLP Status:	●
First Malformed TLP Error Pointer:	N/A

Event Counter

This tab allows you to read the error events like the number of receiver errors, framing errors, etc. for each port. You can use the **Clear P0 counter/Clear P1 counter** to reset the error counter.

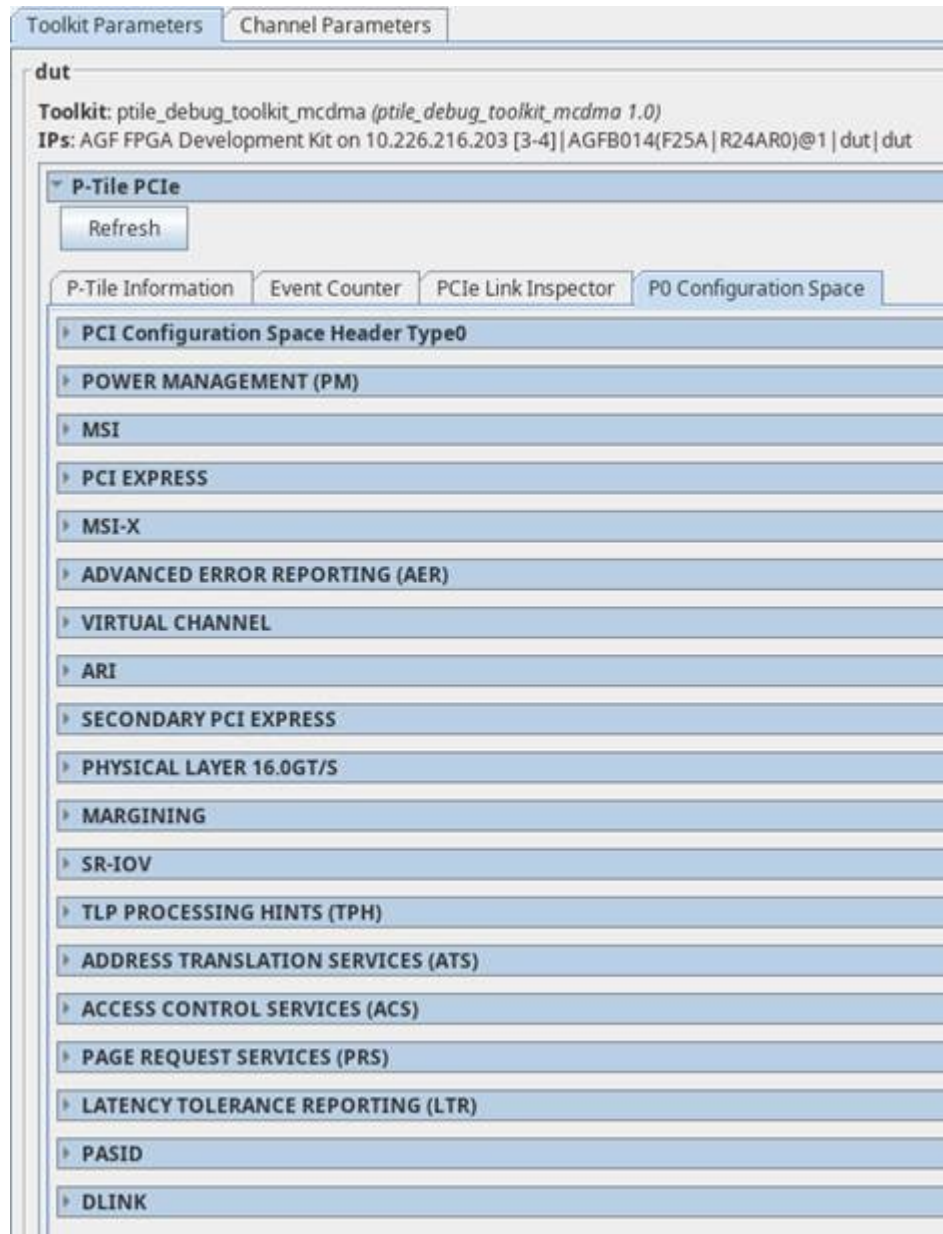
Figure 62. Example of P-Tile Event Counter Tab



P0/P1 Configuration Space

This tab allows you to read the configuration space registers for that port. You will see a separate tab with the configuration space for each port.

Figure 63. Example of P-Tile PCIe Configuration Settings



10.1.4.3. Channel Parameters

The channel parameters window allows you to read the transmitter and receiver settings for a given channel. It has the following 3 sub-windows. Use the **Lane Refresh** button to read the status of the General PHY, TX Path, and RX Path sub-windows for each channel.

Note: To refresh channel parameters for more than one lanes simultaneously, select the lanes under the **Collection** tab, right click and select **Refresh**.

General PHY

This tab shows the reset status of the PHY. In the F-Tile debug toolkit, the PHY Reset is not available. The reset status is indicated by PIPE PhyStatus under Tile Information tab.

Figure 64. Channel Parameters Tab

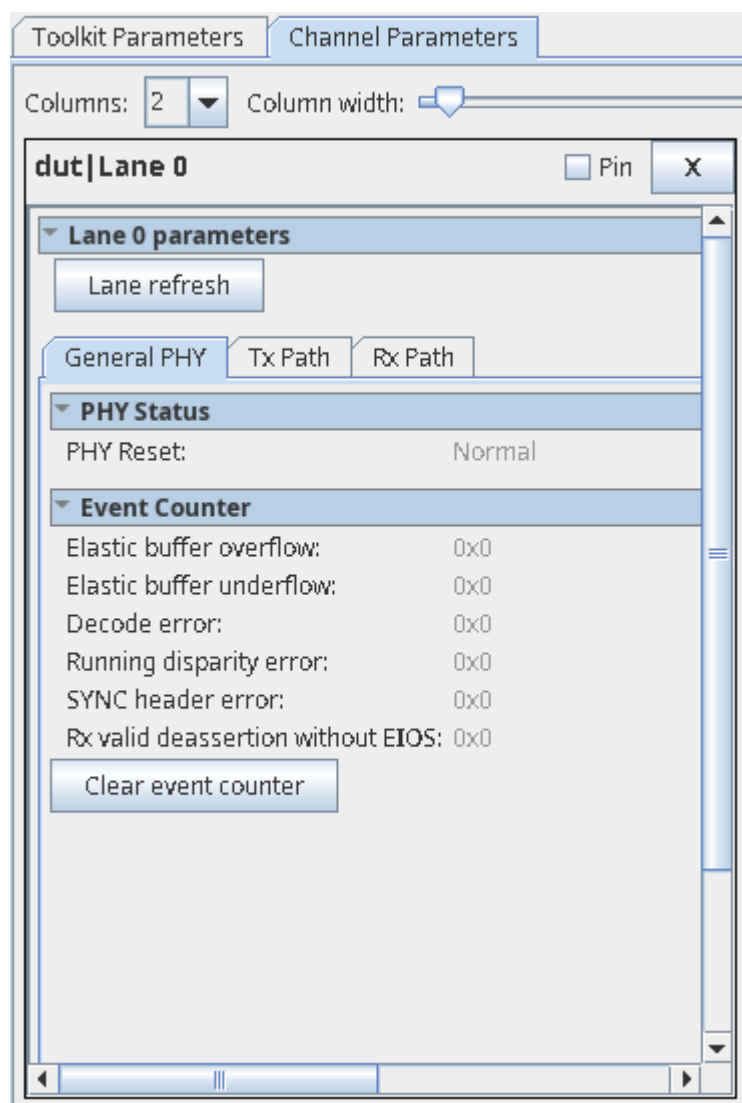


Table 168. General PHY Settings

	Parameters	Values	Descriptions
PHY Status	PHY reset (For P-Tile debug toolkit only)	Normal, Reset	Indicates the PHY is in reset mode. Normal: PHY is out of reset.
continued...			

	Parameters	Values	Descriptions
			Reset: PHY is in reset.
Event Counter ⁽²⁾ to clear the counter values.	Elastic buffer overflow	Hex value	Indicates elastic buffer overflow errors.
	Elastic buffer underflow	Hex value	Indicates elastic buffer underflow errors.
	Decode error	Hex value	Indicates decode errors.
	Running disparity error	Hex value	Indicates running disparity errors.
	SYNC header error	Hex value	Indicates SYNC header errors.
	RX valid deassertion without EIEOS	Hex value	Indicates errors when RX valid deassertion occurs without EIEOS.

TX Path

This tab allows you to monitor the transmitter settings for the channel selected.

Table 169. Transmitter Settings

	Parameters	Values	Descriptions
TX Status	TX Reset (For P-Tile debug toolkit only)	Normal, Reset	Indicates if TX (TX datapath, TX settings) is in reset or normal operating mode. Normal: TX is in normal operating mode. Reset: TX is in reset.
	TX Electrical Idle	True, False	Indicates if TX is in electrical idle. True: indicates TX is in electrical idle. False: indicates TX is out of electrical idle.
TX PLL	TX PLL lock	Green, Red	Indicates if TX PLL is locked. This is dependent on the PLL selected as indicated by TX PLL select. In P-Tile, there is one set of PLLs per Quad. The TX path of each channel reads out the PLL status corresponding to that Quad. In F-tile, there is one PLL per channel. The TX path of the channel reads out the PLL status corresponding to that channel.

continued...

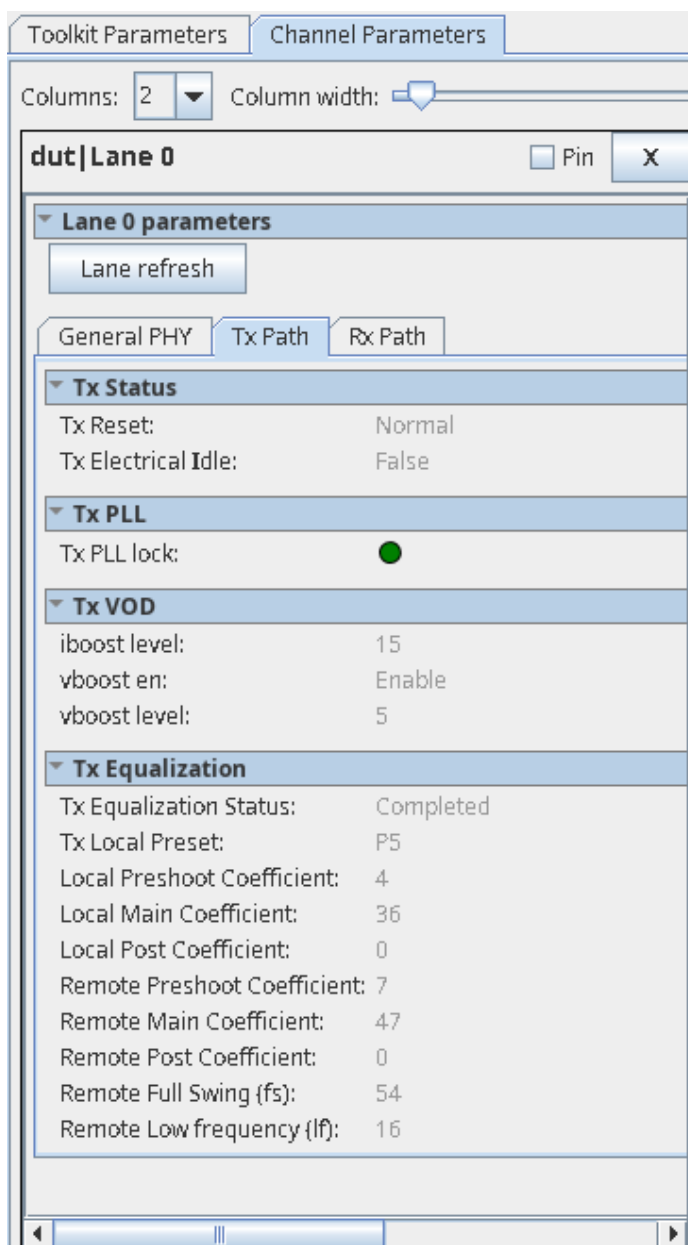
⁽²⁾ Use **Clear event counter**

	Parameters	Values	Descriptions
			<ul style="list-style-type: none"> TX path for Ch0 to 3: Status of PLLs in Quad0 TX path for Ch4 to 7: Status of PLLs in Quad1 TX path for Ch8 to 11: Status of PLLs in Quad2 TX path for Ch12 to 15: Status of PLLs in Quad3 Green: TX PLL is locked. Red: TX PLL is not locked.
TX VOD (For P-Tile debug toolkit only)	Iboost level	Gen3: 15 Gen4: 15	Indicates the transmitter current boost level when the TX amplitude boost mode is enabled.
	Vboost en	Gen3: Enable Gen4: Enable	Indicates if the TX swing boost level is enabled. Enable: TX swing boost is enabled. Disable: TX swing boost is disabled.
	Vboost level	Gen3: 5 Gen4: 5	Indicates the TX Vboost level.
TX Equalization	TX Equalization Status	Not attempted, Completed, Unsuccessful	Indicates transmitter equalization status. The TX local and remote parameters are valid only when the value of Equalization status is returned as completed, indicating equalization has completed successfully.
	TX Local Preset	P0 to P10	Indicates the transmitter driver preset value as requested by the link partner during the Equalization phase of link training. If the preset is not one of these values, then no value is shown.
	Local Pre-shoot coefficient	Depends on the coefficient requested by the link partner.	Indicates the transmitter driver output pre-emphasis (pre-cursor coefficient value).
	Local main coefficient	Depends on the coefficient requested by the link partner.	Indicates the transmitter driver output pre-emphasis (main cursor coefficient value).
	Local post coefficient	Depends on the coefficient requested by the link partner.	Indicates the transmitter driver output pre-emphasis (post-cursor coefficient value).
	Remote Pre-shoot coefficient (†)	Depends on the transmitter driver output of the link partner.	Indicates link partner's transmitter driver's output pre-cursor coefficient value, as received by P-Tile or F-Tile during the Equalization phase of link training. When P-Tile or F-Tile is configured in Endpoint mode, this value
continued...			

	Parameters	Values	Descriptions
			corresponds to the coefficient received during Phase 2 of Equalization.
	Remote main coefficient (†)	Depends on the transmitter driver output of the link partner.	Indicates link partner's transmitter driver's output main cursor coefficient value, as received by P-Tile or F-Tile during the Equalization phase of link training. When P-Tile or F-Tile is configured in Endpoint mode, this value corresponds to the coefficient received during Phase 2 of Equalization.
	Remote post coefficient (†)	Depends on the transmitter driver output of the link partner.	Indicates the link partner's transmitter driver's output post-cursor coefficient value, as received by P-Tile or F-Tile during the Equalization phase of link training. When P-Tile or F-Tile is configured in Endpoint mode, this value corresponds to the coefficient received during Phase 2 of Equalization.
	Remote full swing (fs) (†)	Depends on the device capability of the link partner.	Indicates the full swing value used by the link partner during the Equalization phase of link training.
	Remote low frequency (lf) (†)	Depends on the device capability of the link partner.	Indicates the low frequency value used by the link partner during the Equalization phase of link training.

Note: (†) Refer to the following sections of the *PCI Express Base Specification Revision 4.0*: 4.2.3 Link Equalization Procedure for 8.0 GT/s and Higher Data Rates and 8.3.3 Tx Voltage Parameters.

Figure 65. Example of Transmitter Settings



RX Path

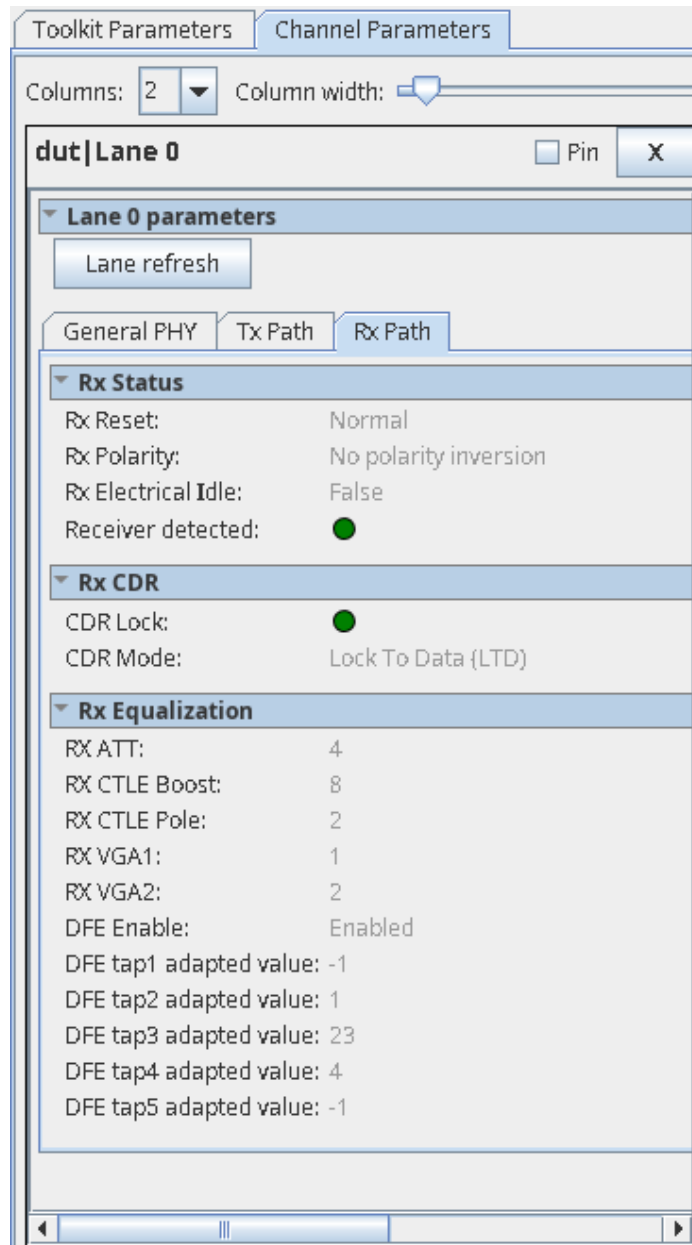
This tab allows you to monitor and control the receiver settings for the channel selected.

Table 170. Receiver Settings

	Parameters	Values	Descriptions
RX Status	RX Reset (For P-Tile debug toolkit only)	Normal, Reset	Indicates if RX (RX datapath, RX settings) is in reset or normal operating mode. Normal: RX is in normal operating mode. Reset: RX is in reset.
	RX Polarity	No polarity inversion, Polarity inversion	Indicates RX polarity inversion for the selected lane. No polarity inversion: no polarity inversion on RX. Polarity inversion: polarity inversion on RX.
	RX Electrical Idle	True, False	Indicates if RX is in electrical idle or not. True: RX is in electrical idle. False: RX is out of electrical idle.
	Receiver Detected	Green, Grey	Green: Far end receiver is detected. Grey: Far end receiver is not detected.
RX CDR	CDR Lock	Green, Red	Indicates the CDR lock state. Green: CDR is locked. Red: CDR is not locked.
	CDR Mode	Locked to Reference (LTR), Locked to Data (LTD)	Indicates the CDR lock mode. LTR: CDR is locked to reference clock. LTD: CDR is locked to data.
RX Equalization	RX ATT (For P-Tile debug toolkit only)	Gen3: 0 Gen4: 0	Indicates the RX equalization attenuation level.
	RX CTLE Boost (For P-Tile debug toolkit only)	Gen3: 12 Gen4: 16	Indicates the RX CTLE boost value.
	RX CTLE Pole (For P-Tile debug toolkit only)	Gen3: 2 Gen4: 2	Indicates the RX CTLE pole value.
	RX VGA1 (For P-Tile debug toolkit only)	Gen3: 5 Gen4: 5	Indicates the RX AFE first stage VGA gain value.
	RX VGA2 (For P-Tile debug toolkit only)	Gen3: 5 Gen4: 5	Indicates the RX AFE second stage VGA gain value.
	DFE Enable (For P-Tile debug toolkit only)	Enable, Disable	Indicates DFE adaptation is enabled for taps 1 - 5. Enable: DFE adaptation is enabled for taps 1 - 5. Disable: DFE adaptation is disabled for taps 1 - 5.
	DFE Tap1 adapted value (For P-Tile debug toolkit only)	<-128 to 127>	Indicates the adapted value of DFE tap 1. This is a signed input (two's complement encoded).
continued...			

	Parameters	Values	Descriptions
	DFE Tap2 adapted value (For P-Tile debug toolkit only)	<-32 to 31>	Indicates the adapted value of DFE tap 2. This is a signed input (two's complement encoded).
	DFE Tap3 adapted value (For P-Tile debug toolkit only)	<-32 to 31>	Indicates the adapted value of DFE tap 3. This is a signed input (two's complement encoded).
	DFE Tap4 adapted value (For P-Tile debug toolkit only)	<-32 to 31>	Indicates the adapted value of DFE tap 4. This is a signed input (two's complement encoded).
	DFE Tap5 adapted value (For P-Tile debug toolkit only)	<-32 to 31>	Indicates the adapted value of DFE tap 5. This is a signed input (two's complement encoded).
	RX VGA Gain (For F-Tile debug toolkit only)	<0 to 127>	Indicates the RX AFE VGA gain value.
	RX High Freq Boost (For F-Tile debug toolkit only)	<0 to 63>	Indicates the RX AFE high frequency boost value.
	DFE Data Tap1 (For F-Tile debug toolkit only)	<0 to 63>	Indicates the adapted value of DFE tap 1.

Figure 66. Example of Receiver Settings



10.1.4.4. Eye Viewer

The Debug Toolkit supports the Eye Viewer tool that allows you to measure on-die eye margin.

The Eye Viewer tool:

- Provides a pictorial representation of the eye for each channel, both in the subdivided (e.g., x8x8) and non-subdivided (e.g., x16) configurations. This feature is available in P-Tile debug toolkit only.
- Provides information on the total eye height, total eye width and eye measurement information from the center of the eye to the four corners (left, right, top, bottom).
- Uses fixed step sizes in the horizontal and vertical directions.
- For P-Tile debug toolkit, Performs the eye measurement at the following bit error rates (BER):
 - 8.0 GT/s (Gen3) @ e-8, 100% confidence level
 - 16.0 GT/s (Gen4) @ e-9, 90% confidence level
 - 8.0 GT/s (Gen3) and 16.0 GT/s (Gen4) @ e-12, 95% confidence level
- For F-tile debug toolkit, performs the eye measurement at BER = 1e-12, 95% confidence level.

Note: The Eye Viewer feature of the Debug Toolkit does not support independent error sampler for performing eye margining. The eye margining is performed on the actual data path. As a result, the eye margining may produce uncorrectable errors in the data stream and cause the LTSSM to go to the Recovery state. You may mask out all errors (example AER registers) while performing the eye margining and reset all error counters, error registers etc. after margining is completed.

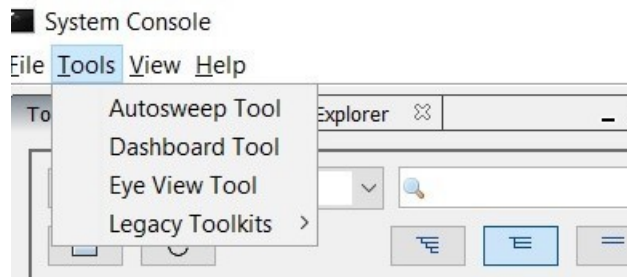
10.1.4.4.1. Running Eye Viewer in the P-Tile Debug Toolkit

Steps to run Eye Viewer in the P-Tile Debug Toolkit

1. In the System Console **Tools** menu option, click on **Eye View Tool**.

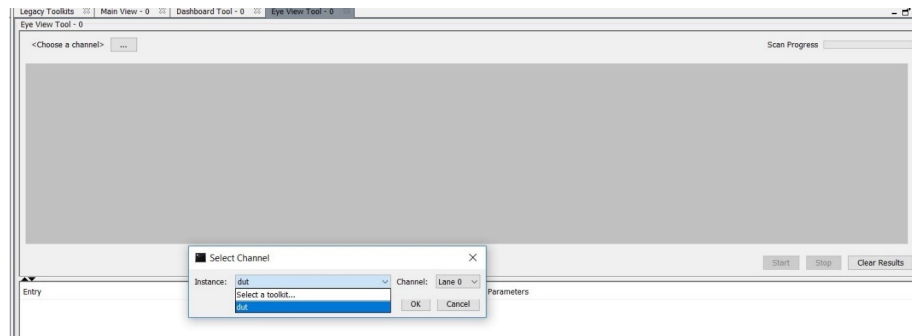
Note: The per-lane information under the **Eye Viewer** tab corresponds to the physical lanes.

Figure 67. Opening the Eye Viewer



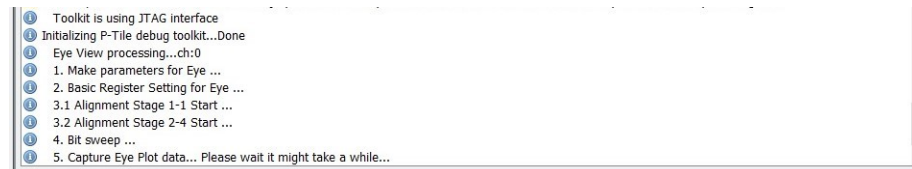
2. This will open a new tab **Eye View Tool** next to the **Main View** tab. Choose the instance and channel for which you want to run the eye view tests.

Figure 68. Opening the Instance and Channel



- For P-Tile debug toolkit, set the **Eye Max BER**. Two options are available: 1e-9 or 1e-12.
- Click **Start** to begin the eye measurement for the selected channel.
- The messages window displays information messages to indicate the eye view tool's progress.

Figure 69. Eye View Tool Messages



- Once the eye measurement is complete, the eye height, eye width and eye diagram are displayed.

Figure 70. Sample Eye Plot [for BER = 1e-9 in P-Tile debug toolkit]

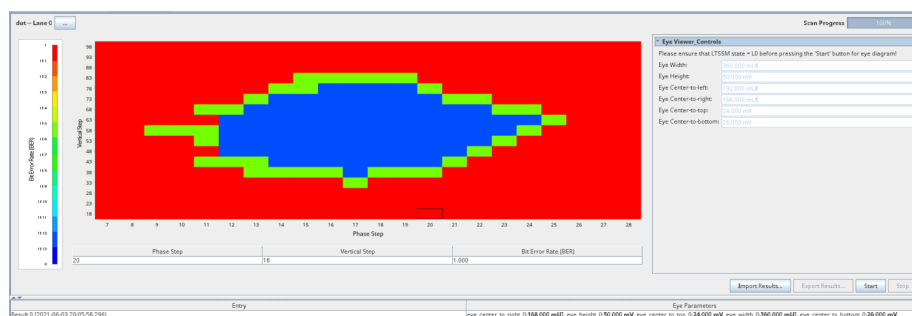
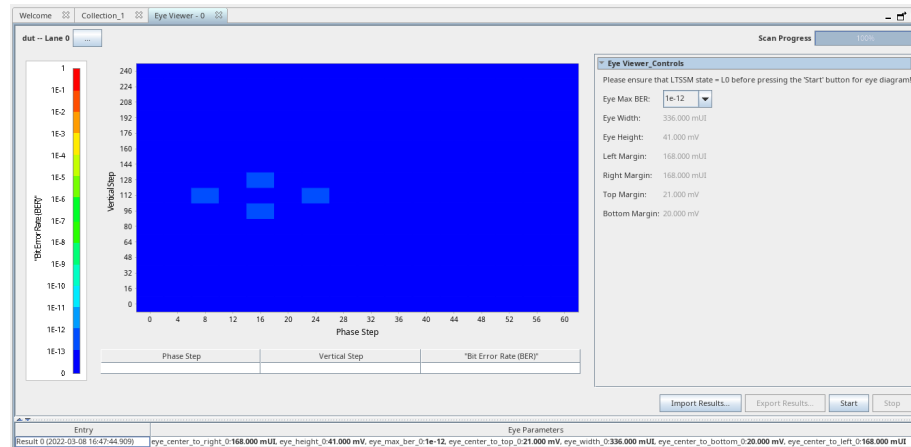


Figure 71. Sample Eye Plot [for BER = 1e-12 in P-Tile debug toolkit]



Note: Full eye plot is not drawn for BER = 1e-12.

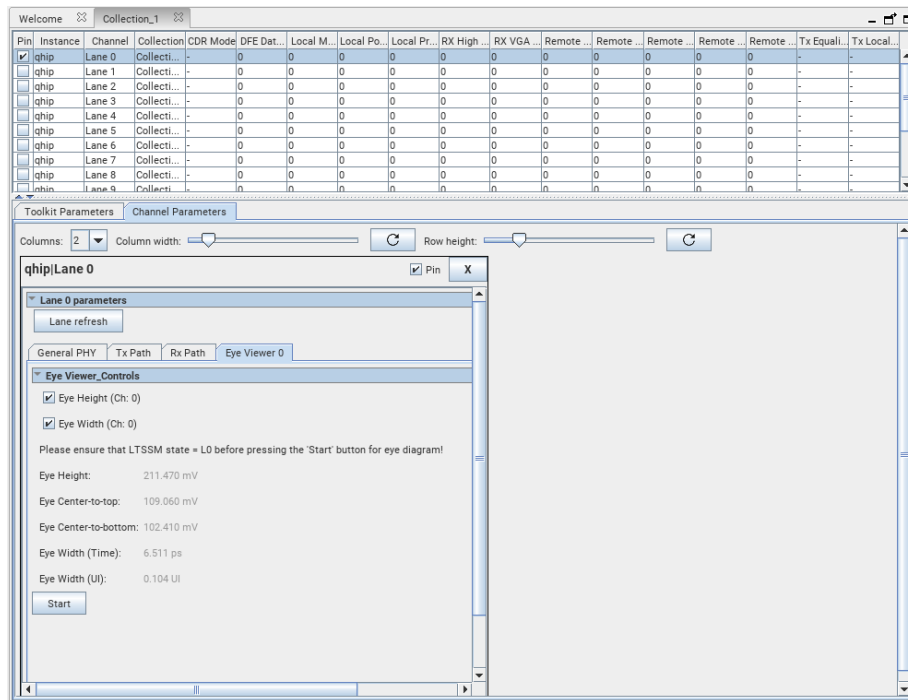
Note: For P-Tile debug toolkit eye margin mask, refer to the [P-Tile Avalon Streaming Intel FPGA IP for PCI Express User Guide](#)

10.1.4.4.2. Running Eye Viewer in the F-Tile Debug Toolkit

Steps to run Eye Viewer in the F-Tile debug toolkit:

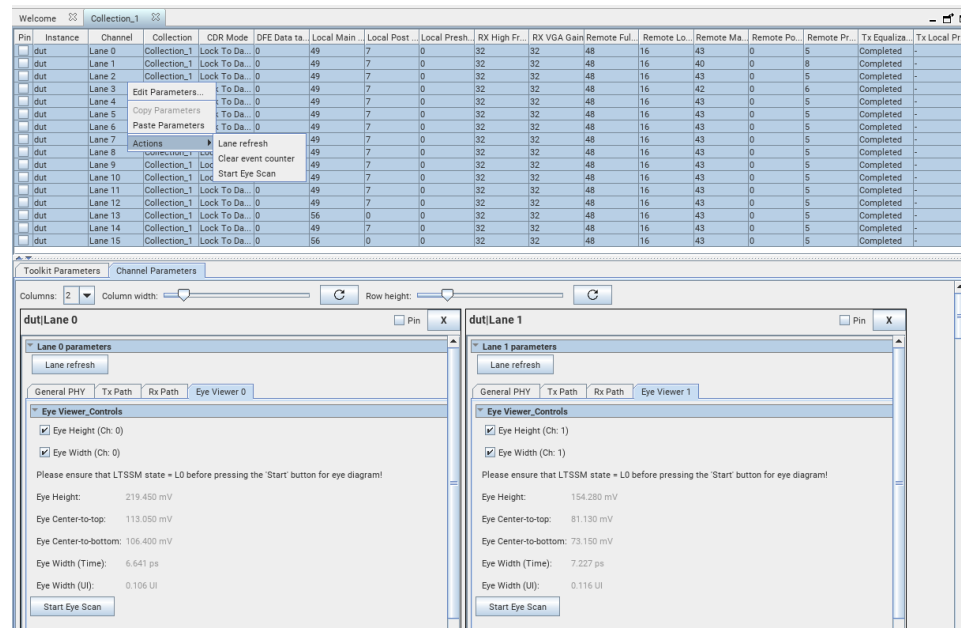
1. To run Eye Viewer for a lane, select the lane from the Collection table.
2. Select the **Eye Viewer** tab in the channel parameter window of the lane.
3. Select **Eye Height**, **Eye Width** or both options.
4. Click **Start Eye Scan** to begin the eye measurement for the selected lane.
5. The messages window displays information messages to indicate the eye view tool's progress.
6. Once the eye measurement completes, the eye height and eye width results are displayed.

Figure 72. Eye Viewer Results



To reduce the repetitive steps to run eye viewer for more than one lanes, select the lanes from the Collection table, right click, select **Actions**, and select **Start Eye Scan**. The eye viewer runs for the selected lanes sequentially.

Figure 73. Enable Eye Viewer for multiple lanes

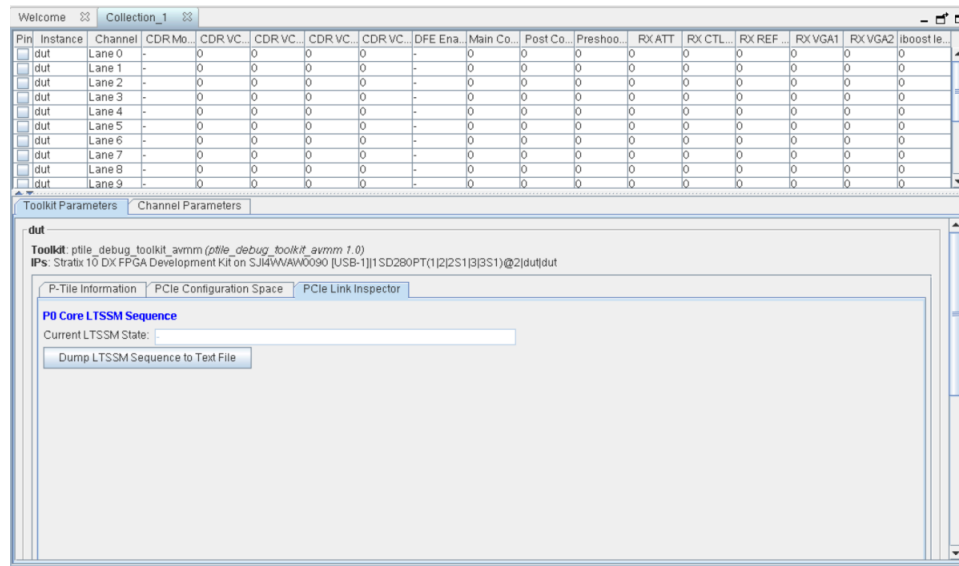


10.1.4.5. Link Inspector

The Link Inspector is found under the **PCIe Link Inspector** tab after opening the Debug Toolkit.

The Link Inspector is enabled by default when the **Enable Debug Toolkit** is enabled. It tracks up to 1024 state transitions with the capability to dump them into a file.

Figure 74. View of the Link Inspector



When the **Dump LTSSM Sequence to Text File** button is initially clicked, a text file (ltssm_sequence_dump_p*.txt) with the LTSSM information is created in the location from where the System Console window is opened. Depending on the PCIe topology, there can be up to four text files. Subsequent LTSSM sequence dumps will append to the respective files.

Note: If you open System Console in a directory that is not writable, the text file will not be generated. To avoid this issue, open System Console from the Command Prompt window (on a Windows system) or change the directory's permission settings to writable.

Figure 75. Example LTSSM Sequence Dump (Beginning)

	LTSSM State	Timer (ns)	LTSSM
0x11	L0	10433620	0
0x0d	RCVRY_LOCK	183	1
0x0f	RCVRY_RCVRCFG	157	2
0x10	RCVRY_IDLE	103	3
0x11	L0	18776291	4
0x0d	RCVRY_LOCK	177	5
0x0f	RCVRY_RCVRCFG	157	6
0x10	RCVRY_IDLE	111	7
0x11	L0	9909106	8
0x0d	RCVRY_LOCK	186	9
0x0f	RCVRY_RCVRCFG	157	10
0x10	RCVRY_IDLE	100	11
0x11	L0	9909117	12
0x0d	RCVRY_LOCK	183	13
0x0f	RCVRY_RCVRCFG	157	14
0x10	RCVRY_IDLE	103	15
0x11	L0	113946626	16
0x0d	RCVRY_LOCK	180	17
0x0f	RCVRY_RCVRCFG	154	18
0x10	RCVRY_IDLE	111	19
0x11	L0	51004154	20
0x0d	RCVRY_LOCK	177	21
0x0f	RCVRY_RCVRCFG	157	22
0x10	RCVRY_IDLE	103	23
0x11	L0	8600454	24
0x0d	RCVRY_LOCK	451	25
0x0f	RCVRY_RCVRCFG	131	26
0x10	RCVRY_IDLE	129	27
0x11	L0	65654751	28
0x0d	RCVRY_LOCK	177	29
0x0f	RCVRY_RCVRCFG	157	30
0x10	RCVRY_IDLE	103	31
0x11	L0	9909114	32
0x0d	RCVRY_LOCK	186	33
0x0f	RCVRY_RCVRCFG	154	34
0x10	RCVRY_IDLE	103	35
0x11	L0	9593126	36
0x0d	RCVRY_LOCK	177	37
0x0f	RCVRY_RCVRCFG	157	38
0x10	RCVRY_IDLE	103	39
0x11	L0	1249051	40

Figure 76. Example LTSSM Sequence Dump (End)

0x0d	RCVRY_LOCK	180	985
0x0f	RCVRY_RCVRCFG	154	986
0x10	RCVRY_IDLE	103	987
0x11	L0	67652631	988
0x0d	RCVRY_LOCK	183	989
0x0f	RCVRY_RCVRCFG	157	990
0x10	RCVRY_IDLE	109	991
0x11	L0	41453594	992
0x0d	RCVRY_LOCK	183	993
0x0f	RCVRY_RCVRCFG	157	994
0x10	RCVRY_IDLE	103	995
0x11	L0	67832460	996
0x0d	RCVRY_LOCK	183	997
0x0f	RCVRY_RCVRCFG	157	998
0x10	RCVRY_IDLE	103	999
0x11	L0	12705806	1000
0x0d	RCVRY_LOCK	183	1001
0x0f	RCVRY_RCVRCFG	157	1002
0x10	RCVRY_IDLE	111	1003
0x11	L0	9909106	1004
0x0d	RCVRY_LOCK	186	1005
0x0f	RCVRY_RCVRCFG	154	1006
0x10	RCVRY_IDLE	103	1007
0x11	L0	6952631	1008
0x0d	RCVRY_LOCK	177	1009
0x0f	RCVRY_RCVRCFG	157	1010
0x10	RCVRY_IDLE	103	1011
0x11	L0	9909114	1012
0x0d	RCVRY_LOCK	186	1013
0x0f	RCVRY_RCVRCFG	154	1014
0x10	RCVRY_IDLE	103	1015
0x11	L0	9909117	1016
0x0d	RCVRY_LOCK	183	1017
0x0f	RCVRY_RCVRCFG	157	1018
0x10	RCVRY_IDLE	103	1019
0x11	L0	93771306	1020
0x0d	RCVRY_LOCK	180	1021
0x0f	RCVRY_RCVRCFG	157	1022
0x11	L0	-	Current

The current LTSSM state is: L0

Each LTSSM monitor has a FIFO storing the time values and captured LTSSM states. When you choose to dump out the LTSSM states, reads are dependent on the FIFO elements and will empty out the FIFO.

The Link Inspector only writes to its FIFO if there is a state transition. In cases where the link is stable in L0, there will be no write and hence no text file will be dumped.

When you want to dump the LTSSM sequence, a single read of the FIFO status of the respective core is performed. Depending on the empty status and how many entries are in the FIFO, successive reads are executed.



11. Multi Channel DMA FPGA IP for PCI Express User Guide Archives

For the latest and previous versions of this document, refer to the [Multi Channel DMA FPGA IP for PCI Express User Guide](#). If an IP or software version is not listed, the user guide for the previous IP or software version applies.

12. Revision History for the Multi Channel DMA FPGA IP for PCI Express User Guide

Date	Quartus Prime Version	IP Version	Changes
2025.08.04	25.1.1	24.2.2 [H-Tile] 8.3.2 [P-Tile] 9.3.2 [F-Tile] 5.3.2 [R-Tile]	<ul style="list-style-type: none"> Updated the IP name for Altera rebranding. Updated the IP version numbers in <i>Release Information</i>. Updated the screenshots in the <i>Parameters (P-Tile) (F-Tile) (R-Tile)</i> chapter. Updated the <i>Legacy Interrupt Interface</i> description.
2025.01.27	24.3.1	24.2.0 [H-Tile] 8.3.0 [P-Tile] 9.3.0 [F-Tile] 5.3.0 [R-Tile]	<ul style="list-style-type: none"> Updated the maximum number of DMA channels in <i>Endpoint Mode</i>. Updated the tables in <i>Resource Utilization</i>. Updated the IP and Quartus versions in <i>Release Information</i>. Added a Note in <i>Legacy Interrupt Interface</i>.
2024.07.30	24.2	24.1.0 [H-Tile] 8.1.0 [P-Tile] 9.1.0 [F-Tile] 5.1.0 [R-Tile]	<ul style="list-style-type: none"> <i>Terms and Acronyms</i>: PCIe Gen1/2/3/4/5 terms added <i>Known Issues</i>: 24.2 known issues added <i>Device Family Support</i>: Agilex 9 support added <i>Release Information</i>: Release Information table updated <i>Avalon-MM Write (H2D) and Read (D2H) Master</i>: Information updated <i>Avalon-MM Read Master (D2H)</i>: Information updated <i>Legacy Interrupt Interface</i>: New section added <i>MCDMA Settings</i>: Information updated
2024.01.29	23.4	23.1.0 [H-Tile] 7.1.0 [P-Tile] 8.0.0 [F-Tile] 4.1.0 [R-Tile]	<ul style="list-style-type: none"> <i>Endpoint Mode</i>: Information updated <i>Root Port Mode</i>: Information updated <i>Release Information</i>: Release Information table updated [Updated 2024.02.27] <i>Endpoint MSI Support Through BAS</i>: Note added <i>Data Mover Only</i>: Device to Host (D2H) Data Mover bullet point updated <i>H2D Descriptor Format (h2ddm_desc)</i>: New row added in the table <i>RSVD [255:229]</i> <i>D2H Descriptor Format (d2hdm_desc)</i>: New row added in the table <i>RSVD [255:225]</i> <i>D2H Descriptor Format (d2hdm_desc)</i>: New table added <i>Writeback and MSI-X Format</i> <i>Precision Time Management (PTM) Interface</i>: New section added <i>PCIe0 Configuration, Debug and Extension Options</i>: Table updated for Default Values for F-Tile <i>PCIe0 PCI Express / PCI Capabilities</i>: GUI screenshot updated [Updated 2024.02.27] <i>PCIe0 PTM</i>: R-Tile support information added
continued...			

Date	Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> • <i>PCIe0 ACS for Physical Functions</i>: Information updated • <i>PCIe0 ACS for Virtual Functions</i>: Feature support information added • <i>Receiver Detection (F-Tile MCDMA IP Only)</i>: New section added [Updated 2024.02.27] • <i>MCDMA Settings</i>: <ul style="list-style-type: none"> — R-Tile support information updated for parameters <i>Enable User-FLR</i> and <i>Enable MSI Capability</i> — Data Mover Only information updated for parameter <i>User Mode</i> — <i>Export pld_warm_rst_rdy</i> and <i>link_req_rst_n</i> interface to top level: Parameter support removed • <i>Architecture</i>: <i>ifc_uio</i> replaced with <i>igb_uio</i> in the <i>MCDMA Architecture</i> block diagram • <i>API Flow: Device to Host Sequence</i> figure updated
2023.10.06	23.3	23.0.0 [H-Tile] 7.0.0 [P-Tile] 7.0.0 [F-Tile] 4.0.0 [R-Tile]	<ul style="list-style-type: none"> • <i>Known Issues</i>: List updated with new bullet points from (7) to (11) added for the Quartus Prime 23.3 release • <i>Release Information</i>: Table updated for Quartus Prime 23.3 release. • <i>User MSI-X</i>: Note added about R-Tile support • <i>Endpoint MSI Support through BAS</i>: Note added about H-Tile support • <i>Hard IP Reconfiguration Interface</i>: Note added about H-Tile support • <i>MCDMA Settings</i>: New row added to table <i>Enable address byte aligned transfer</i> • <i>MSI-X</i>: Note updated and Table added • <i>Top Level Settings</i>: New row added to table <i>Enable PIPE Mode Simulation</i> • <i>PCIeo MSI-X</i>: Note updated • <i>PCIe0 PRS</i>: New row added to the table <i>PF0 Page Request Services Outstanding Capacity</i> • <i>PCIe0 TPM</i>: New section added • <i>Analog Parameters (F-Tile MCDMA IP Only)</i>: New section added • <i>Software Programming Model</i>: Software folder information added • <i>Overview</i>: Debug Toolkit support information note updated
2023.07.14	23.2	22.3.0 [H-Tile] 6.0.0 [P-Tile] 6.0.0 [F-Tile] 3.0.0 [R-Tile]	<p>General updates:</p> <ul style="list-style-type: none"> • Added support for Agilex 9. • <i>Known Issues</i>: Bullet points (5), (6), (7), (8), (9) added • <i>Recommended Speed Grades</i>: Gen4 Stratix 10 DX values updated • <i>Release Information</i>: Information updated for 23.2 • <i>Port List (P-Tile) (F-Tile) (R-Tile)</i>: Figure updated • <i>Avalon-MM Write (H2D) and Read (D2H) Master</i>: Burst Count and Data Width updated

continued...

Date	Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> • <i>Top Level Settings:</i> <ul style="list-style-type: none"> — <i>Hard IP mode (P-Tile):</i> Parameter updated — <i>Hard IP mode (F-Tile):</i> New parameter added — <i>Hard IP mode (R-Tile):</i> Parameter updated — <i>Port Mode:</i> Parameter updated • <i>PCIe0 Configuration, Debug and Extension Options:</i> <i>Disable Equalization Phase 2 and Phase3</i> new parameter added • <i>PCIe0 Configuration, Debug and Extension Options:</i> R-Tile & F-Tile addresses updated • <i>PCIe0 ACS for Physical Functions:</i> Note added • R-Tile specific updates in the following sections: <ul style="list-style-type: none"> • <i>Endpoint Mode</i> • <i>Root Port Mode</i> • <i>Endpoint MSI Support through BAS</i> • <i>Hard IP Reconfiguration Interface</i> • <i>Config TL Interface</i> • <i>Data Mover Only</i> • <i>Interface Overview</i> • <i>Avalon-MM Write Master (H2D)</i> • <i>Avalon-MM Read Master (D2H)</i> • <i>Avalon-ST Source (H2D)</i> • <i>Avalon-ST Sink (D2H)</i> • <i>Bursting Avalon-MM Master (BAM) Interface</i> • <i>Bursting Avalon-MM Slave (BAS) Interface</i> • <i>H2D Data Mover Interface</i> • <i>MCDMA Settings</i>
2023.04.11	23.1	22.2.0 [H-Tile] 5.1.0 [P-Tile] 5.1.0 [F-Tile] 2.0.0 [R-Tile]	<ul style="list-style-type: none"> • Updated product family name to "Intel Agilex® 7". • <i>Known Issues:</i> Removed issues fixed in Quartus Prime 23.1 release. Bullet point (7) newly added in Quartus Prime 23.1 release. • <i>Release Information:</i> Release Information table updated • <i>Bursting Avalon-MM Slave (BAS) Interface:</i> <i>bas_address_i</i> parameter information updated • <i>Top-Level Settings:</i> <i>Enable Independent Perst</i> parameter updated • <i>Running Eye Viewer in the F-Tile Debug Toolkit:</i> New section added
2023.02.11	22.4	22.1.0 [H-Tile] 5.0.0 [P-Tile] 5.0.0 [F-Tile] 1.0.0 [R-Tile]	<ul style="list-style-type: none"> • Initial Release and selected feature support for MCDMA R-Tile IP • <i>Multi Channel DMA IP Kernel Mode Character Device Driver</i> is no longer supported from the Quartus Prime 22.4 release onwards. All related information has been removed. • <i>Known Issues:</i> Bullet points (3) to (7) added for Quartus Prime 22.4 release • <i>Endpoint Mode:</i> R-Tile information added • <i>Root Port Mode:</i> F/R/P/H-Tile specific information added • <i>Recommended Speed Grades:</i> R-Tile information added. Gen5 information added. • <i>Resource Utilization:</i> BAM_BAS_MCDMA user mode information added • <i>Resource Utilization:</i> <i>Agilex R-Tile PCIe x8 [Avalon-MM Interface]</i> table added
continued...			

Date	Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> • <i>Release Information</i>: IP version information updated. R-Tile version added. • <i>Functional Description</i>: BAM+BAS+MCDMA user mode information added • <i>Endpoint MSI Support through BAS</i>: Note added at the end of the section • <i>Configuration Intercept Interface (EP Only)</i>: R-Tile information added • <i>Interface Overview</i>: R-Tile information added • <i>Port List (P-Tile) (F-Tile) (R-Tile)</i>: Port List updated • <i>Clocks</i>: app_slow_clk and pcie_systempll_clk clock signals added • <i>F-Tile System PLL Reference Clock Requirements</i>: New section added • <i>Resets</i>: R-Tile reset signals added • <i>Resets</i>: i_gpio_perst#_n signal information added • <i>Interface clock domain for R-Tile added for following</i>: <ul style="list-style-type: none"> — Avalon-MM PIO Master — Avalon-MM Write Master (H2D) — Avalon-MM Read Master (D2H) — User Event MSI-X Interface — User Functional Level Reset (FLR) Interface • <i>Bursting Avalon-MM Master (BAM) Interface</i>: BAM Signals table updated • <i>Bursting Avalon-MM Slave (BAS) Interface</i>: BAS Signals table updated • <i>Hard IP Reconfiguration Interface</i>: Table updated with new signals usr_hip_reconfig_clk_o, usr_hip_reconfig_rst_n_o • <i>Hard IP Reconfiguration Interface</i>: Reconfig Address added for R-Tile • <i>Configuration Intercept Interface (EP Only)</i>: R-Tile information added • <i>Hard IP Status Interface</i>: R-Tile information added • <i>Hard IP Status Interface</i>: p0_ltssm_st_hipfifo_ovrflw_o new signal added • <i>MCDMA Settings</i>: BAM+BAS+MCDMA added for User Mode parameter • <i>Top-Level Settings</i>: R-Tile information added. P-Tile and F-Tile specific information added. • <i>Base Address Register</i>: BAM+BAS+MCDMA user mode added • <i>PCIe0 Configuration, Debug and Extension Options</i>: R-Tile information added • <i>PCIe0 Link</i>: New row added to the table <i>Enable Modified TS</i> • <i>PCIe0 MSI-X</i>: New row added <i>VF Table size</i> to table <i>PCIe0 PF MSI-X</i> • <i>MCDMA Settings</i>: BAM+BAS+MCDMA added for User Mode parameter. R-Tile specific information added. • <i>Example Designs</i>: BAM+BAS+MCDMA User Mode information added. R-Tile specific information added. • <i>Registers</i>: Note added about Read/Write access to CSR address space
continued...			

Date	Quartus Prime Version	IP Version	Changes
2022.11.01	22.3	22.0.0 [H-Tile] 4.0.0 [P-Tile] 4.0.0 [F-Tile]	<ul style="list-style-type: none"> • <i>Known Issues</i>: Past issues that no longer present in the current release have been removed • <i>Endpoint Mode</i>: Feature list updated • <i>Root Port Mode</i>: Feature list updated • <i>Release Information</i>: IP Version and Quartus Prime Version updated • <i>Functional Description</i>: IP Block Diagram updated • <i>Multi Channel DMA</i>: Note description updated to specify alignment in D2H direction • <i>D2H Descriptor Fetch</i>: Information added about Tail pointer updates • <i>Support for Unaligned (or byte-aligned) Data Transfer</i>: New section added • <i>MSI Interrupt</i>: New section and sub-sections added • <i>29 bit AVMM address format</i>: New sub-section added under Config Slave (CS) • <i>14 bit AVMM address format</i>: New sub-section added under Config Slave (CS) • <i>Configuration Access Mechanism</i>: New sub-section added under Config Slave (CS) • <i>Root Port Address Translation Table Enablement</i>: New section added • <i>Port List (P-Tile) (F-Tile)</i>: Figure updated with new MSI Interface added • <i>MSI Interface</i>: New section added • <i>Top-Level Settings</i> <ul style="list-style-type: none"> — <i>Hard IP mode</i> parameter information updated — <i>Number of PCIe</i> parameter information updated — <i>PLD Clock Frequency</i> parameter information updated — <i>Enable Independent Perst</i> parameter information added • <i>PCIe0 Configuration, Debug and Extension Options</i> <ul style="list-style-type: none"> — Default value of parameter <i>Gen 3 Requested equalization far-end TX preset vector</i> for F-Tile updated — Default Value of parameter <i>Gen 4 Requested equalization far-end TX preset vector</i> for F-Tile updated • <i>MCDMA Settings</i>: New parameters added <ul style="list-style-type: none"> — <i>Enable address byte aligned transfer</i> — <i>Enable MSI Capability</i> — <i>Enable MSI 64-bit addressing</i> — <i>Number of MSI Messages Requested</i> — <i>Enable MSI Extended Data Capability</i> — <i>Export pld_warm_rst_rdy and link_req_rst_n interface to top level</i> • <i>MCDMA Settings</i>: New table added <i>Root Port MCDMA Settings ATT Parameters</i> • <i>MCDMA Settings</i>: GUI screenshot updated

continued...

Date	Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> • <i>PCIe1 Settings</i>: New section added • <i>Example Designs</i> <ul style="list-style-type: none"> — Information updated for <i>Current development kit</i> parameter — GUI screenshot added <i>Example Design Settings [for x16 mode]</i> — GUI screenshot added <i>Example Design Settings [for 2x8 mode]</i> • <i>Queue Control (QCSR)</i> <ul style="list-style-type: none"> — Table <i>Queue Control Registers</i>: Register Name <i>Q_DEBUG_STATUS_3</i> information removed — Table <i>Queue Control Registers</i>: Register Name <i>Q_DATA_DRP_ERR_CTR</i> information added — New register table <i>Q_DATA_DRP_ERR_CTR (Offset 8'h40)</i> added • Troubleshooting/Debugging Chapter updates <ul style="list-style-type: none"> • Unless or otherwise noted, the debug toolkit features described in the <i>Troubleshooting/Debugging</i> Chapter apply to both P-Tile and F-Tile. • <i>Enabling the Debug Toolkit</i>: Description of this operation expanded • <i>Launching the Debug Toolkit</i>: New note added for about the standalone install of Quartus Prime Pro Edition • <i>Main View: Channel Mapping for Bifurcated Ports</i> table updated • <i>Toolkit Parameters</i> <ul style="list-style-type: none"> — In table <i>Available Parameter Settings</i>, rows <i>Advertised speed</i>, <i>Advertised width</i>, <i>Negotiated speed</i>, <i>Negotiated width</i> have been updated — In table <i>Available Parameter Settings</i>, new row <i>PIPE PhyStatus (For F-Tile debug toolkit only)</i> has been added • <i>Channel Parameters</i>: Tag (For P-Tile debug toolkit only) has been added wherever required • <i>Eye Viewer</i>: F-Tile support information added • <i>Link Inspector</i>: New section added
2022.08.19	22.2	21.5.0 [H-Tile] 3.1.0 [P-Tile] 3.0.0 [F-Tile]	<ul style="list-style-type: none"> • <i>Release Information</i>: Table updated • <i>Functional Description</i>: Block Diagram updated • <i>MCDMA Settings</i>: <i>Number of ports</i> parameter info updated • <i>Top-Level Settings</i>: <i>Enable Ptile Debug Toolkit (P-Tile)</i> and <i>Enable Debug Toolkit (F-Tile)</i> parameters row added • <i>Multifunction and SR-IOV System Settings Parameters [Endpoint Mode: Number of DMA channels allocated to PF0]</i> parameter info updated • <i>Example Designs</i>: <i>Currently Selected Example Design</i> parameter info updated for H-Tile • <i>Example Designs</i>: <i>Current development kit</i> parameter info updated for P-Tile and F-Tile • <i>Eye Viewer</i>: Note added about eye-margining <p>4 Port AVST Mode has been deprecated from this release. All related 4 Port Mode information has been removed from the following sections:</p> <ul style="list-style-type: none"> • <i>Endpoint Mode</i> • <i>Avalon-ST Source (H2D) and Sink (D2H)</i> • <i>Packet (File) Boundary</i>

continued...

Date	Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> Bursting Avalon-MM Slave (BAS) Avalon-ST Source (H2D) Avalon-ST Sink (D2H) Port List (H-Tile) Port List (P-Tile and F-Tile)
2022.04.20	22.1	21.4.0 [H-Tile] 3.0.0 [P-Tile] 2.0.0 [F-Tile]	<ul style="list-style-type: none"> Known Issues [New section added] Release Information [IP Version updated] Recommended Speed Grades [Table updated] Resource Utilization [All tables updated] Descriptors [Software Descriptor Format Table Rows Updated: SRC_ADDR [63:0], DEST_ADDR [127:64], PYLD_CNT [147:128]] Avalon-MM PIO Master [Description updated] Avalon-ST 1-Port Mode [Note added] Bursting Avalon-MM Master [Description updated] H2D Descriptor Format (h2ddm_desc) [Table Rows Updated: RSVD [200:200], RSVD [216:201]] H2D Descriptor Completion Packet Format (h2ddm_desc_cmpl) [Table Rows Updated: En_partial_cmpl_data [82:82], Completion data] D2H Data Mover [Table Row updated: d2hdm_desc] D2H Descriptor Format (d2hdm_desc) [Table Rows updated: SRC_ADDR [63:0], MM_mode [176:176], App_specific_bits [179:177], DESC_IDX1 [195:180], RSVD [196:196], RSVD [212:197]] Application Specific Bits [Table updated] Avalon-MM PIO Master [Note added] Avalon-MM PIO Master [Table updated. Row: rx_pio_address_o[n:0]] MCDMA Settings [H-Tile. New GUI Screenshot added] [D2H Prefetch channels Row in table updated] Base Address Register [P-Tile and F-Tile] [Note added] MCDMA Settings [P-Tile and F-Tile] [D2H Prefetch channels Row in table updated] Example Designs [Currently Selected Example Design Row in Table updated] Software Flow [QCSR Registers list updated in Step 1] Queue Control (QCSR) [Queue Control Registers Table Rows Updated: Q_PYLD_CNT, Q_RESET] Control Register (GCSR) [Note Added]
2022.01.14	21.4	21.3.0 [H-Tile] 2.2.0 [P-Tile] 1.1.0 [F-Tile]	<ul style="list-style-type: none"> Data Mover Only user mode option added to Endpoint Mode Resource Utilization tables updated IP Version updated in Release Information Data Mover Only user mode option added in Chapter Functional Description Data Mover Interface and Hard IP Status Interface added to Chapter Interface Interview Port List (P-Tile and F-Tile) figure updated with data mover mode interfaces
continued...			

Date	Quartus Prime Version	IP Version	Changes
			<ul style="list-style-type: none"> • <i>PCIe0 Configuration, Debug and Extension Options</i> section updated in <i>Parameters (P-Tile and F-Tile)</i> Chapter • <i>Enable 10-bit tag support</i> GUI feature added to <i>MCDMA Settings</i> in <i>Parameters (P-Tile and F-Tile)</i> Chapter • Data Mover Only user mode option added to <i>Example Designs</i> table in <i>Parameters (P-Tile and F-Tile)</i> Chapter • <i>Device Management</i> and <i>Channel Management</i> sections updated for Network Device Driver • <i>ethtool support</i> and <i>debugfs support</i> added to Network Device Driver • <i>SRIOV Support</i> information added for Network Device Driver
2021.10.29	21.3	21.2.0 [H-Tile] 2.1.0 [P-Tile] 1.0.0 [F-Tile]	<ul style="list-style-type: none"> • <i>Recommended Speed Grades</i> table updated with F-Tile support information • <i>Resource Utilization</i> tables updated • <i>Release Information</i> updated • <i>Valid user modes and required functional blocks</i> table updated • Address format information added to <i>Config Slave</i> • <i>Multi Channel DMA IP for PCI Express Port List (P-Tile and F-Tile)</i> figure updated with F-Tile information • <i>Config TL Interface</i> signal table updated • F-Tile support information added to <i>Configuration Intercept Interface (EP Only)</i> • F-Tile support information added to <i>Parameters (P-Tile and F-Tile)</i> Chapter • <i>MCDMA IP Software Driver Differentiation</i> table added • Network Device Driver information added in <i>Multi Channel DMA IP Kernel Mode Network Device Driver</i> • Debug Toolkit information added
2021.08.16	21.2	21.1.0 [H-Tile] 2.0.0 [P-Tile]	<ul style="list-style-type: none"> • Fixed H-Tile IP revision number • Added 500 MHz support for P-Tile MCDMA IP • Added P-Tile single port Avalon-ST DMA up to 256 channels • Added MCDMA IP DPDK Poll-Mode based Driver • Added MCDMA IP Kernel Mode (No SRIOV) Driver
2021.05.28	21.1	2.0.0 [H-Tile] 1.0.0 [P-Tile]	<ul style="list-style-type: none"> • PCIe Gen4 (P-Tile) Support • Support for x8 link width • MCDMA 1 port AVST interface • BAM, BAS, BAM+BAS, BAM+MCDMA modes • SR-IOV support • Root Port support (IP only) • Config Slave interface for RP
2020.07.20	20.2	20.0.0 [H-Tile]	Initial Release