



PCIe Scatter-Gather Direct Memory Access (SGDMA) Core User Guide

UG-PCIeSGDMA-v1.0
November 2025
www.efinixinc.com



Contents

Introduction.....	4
Features.....	4
Device Support.....	5
Resource Utilization and Performance.....	5
Release Notes.....	5
Functional Description.....	6
Ports.....	7
Register List.....	16
cfg_identifier (0x0000).....	18
reg_user_max_payload (0x0004).....	18
reg_user_max_read_req (0x0008).....	19
reg_write_timeout (0x000C).....	19
msi_enable and msix_enable (0x0018).....	19
max_payload (0x001C).....	20
max_read_req (0x0020).....	20
clk_pcie_w (0x0024).....	20
dsc_identifier (0x1000).....	20
dsc_crd_en (0x1004).....	21
dsc_stop (0x100C).....	21
dsc_restart (0x1018).....	21
irq_identifier (0x2000).....	21
usr_irq.....	22
dma_irq (0x2008).....	22
user_irq_i (0x200C).....	22
dma_irq_j (0x2010).....	23
user_irq_lut (0x2014).....	23
user_irq_lut (0x2018).....	23
user_irq_lut (0x201C).....	24
user_irq_lut (0x2020).....	24
dma_irq_lut (0x2024).....	24
dma_irq_lut (0x2028).....	25
user_en (0x202C).....	25
dma_en (0x2038).....	25
htc_identifier (0x3000).....	26
htc_dsc_adj (0x3004).....	26
htc_dsc_crd (0x3008).....	26
htc_dsc_addr_l (0x300C).....	26
htc_dsc_addr_h (0x3010).....	26
htc_dma_wb_addr_l (0x3014).....	27
htc_dma_wb_addr_h (0x3018).....	27
htc_dma_ctl (0x301C).....	27
htc_dma_status (0x3028).....	28
htc_dma_dsc_compl_cnt (0x3030).....	28
htc_dma_intr_mask (0x3038).....	29
cth_identifier (0x4000).....	29
cth_dsc_adj (0x4004).....	29
cth_dsc_crd (0x4008).....	30
cth_dsc_addr_l (0x400C).....	30
cth_dsc_addr_h (0x4010).....	30
cth_dma_wb_addr_l (0x4014).....	30
cth_dma_wb_addr_h (0x4018).....	30
cth_dma_ctl (0x401C).....	31

cth_dma_status (0x4028).....	32
cth_dma_dsc_compl_cnt (0x4030).....	33
cth_dma_intr_mask (0x4038).....	33
MSI-X/MSI Interrupt Vector Table (0x8000).....	34
Descriptors.....	35
AXI4-MM Bypass Master.....	36
AXI4-Stream Data Interface.....	36
APB3 Configuration Interface.....	37
Customizing the PCIe SGDMA.....	38
Software Driver.....	40
PCIe SGDMA Example Design.....	40
PCIe SGDMA Testbench.....	40
Revision History.....	40

Introduction

This user guide describes how to implement high-performance direct memory access (DMA) using the Efinix® PCIe Scatter-Gather Direct Memory Access (SGDMA) IP, a configurable IP core designed specifically for PCIe endpoint applications. You use this SGDMA IP with the Titanium or Topaz PCIe Controller for high-speed data transfers between the FPGA and host.

Features

PCIe Scatter-Gather Direct Memory Access supports:

- Data transfer using the standard AXI4-MM protocol
- Data transfer in AXI4-Stream mode
- Master APB3 interface for host-to-card register configuration
- Slave APB3 interface for user logic-to-IP register configuration
- Legacy and MSI-X interrupts

The following table provides for a more indepth overview of the features supported by the PCIe SGDMA IP core.

Table 1: Supported Features

Feature	Efinix PCIe SGDMA IP
Data width	✓ 256 bits
Host-to-Card	Up to four channels
Card-to-Host	Up to four channels
Descriptor Engine Pause Control	✓ Supported
AXI-MM Bypass Interface	✓ Supported
Descriptor	✓ Up to 256 MB data length per descriptor
Write back/polling	Supported
Legacy INTX	✓ Single channel (INTA only)
User INTR	✓ INTA ✓ MSIX ✓ MSI
MSIX	✓ Supported
MSI	✓ Supported
MPS, MRRS	✓ MPS = 128, 256, 512 ✓ MRRS = 128, 256, 512, 1024, 2048, 4096 Measured in bytes
Clock frequency	✓ AXI clock = {125 to 250} MHz ✓ APB clock = {20 to 200} MHz Must use same CLK source as PCIe EP
APB interface	Supported

Feature	Efinix PCIe SGDMA IP
AXI4-MM interface	Supported
AXI4-Stream interface	Supported

Device Support

Refer to the [Titanium Selector Guide](#) and [Topaz Selector Guide](#) to get the updated list of devices that support transceivers.

Resource Utilization and Performance

These resource and performance values are based on specific supported FPGAs. These values serve as a guideline only and may vary depending on device resource utilization, design congestion, and user design modifications.

Table 2: Titanium Resource Utilization and Performance

FPGA Model	Configuration		Timing Model	Logic Element (XLR)	Memory Blocks	DSP Blocks	f _{MAX} (MHz)	Efinity® Version
	HTC & CTH Channel	User Interface						
Ti375N1156	1	AXIMM	C4	24,745	47	0	264	2025.2
	4	AXIMM		89,418	52	0	270	
	1	AXIST		45,527	66	0	272	
	4	AXIST		109,675	108	0	257	

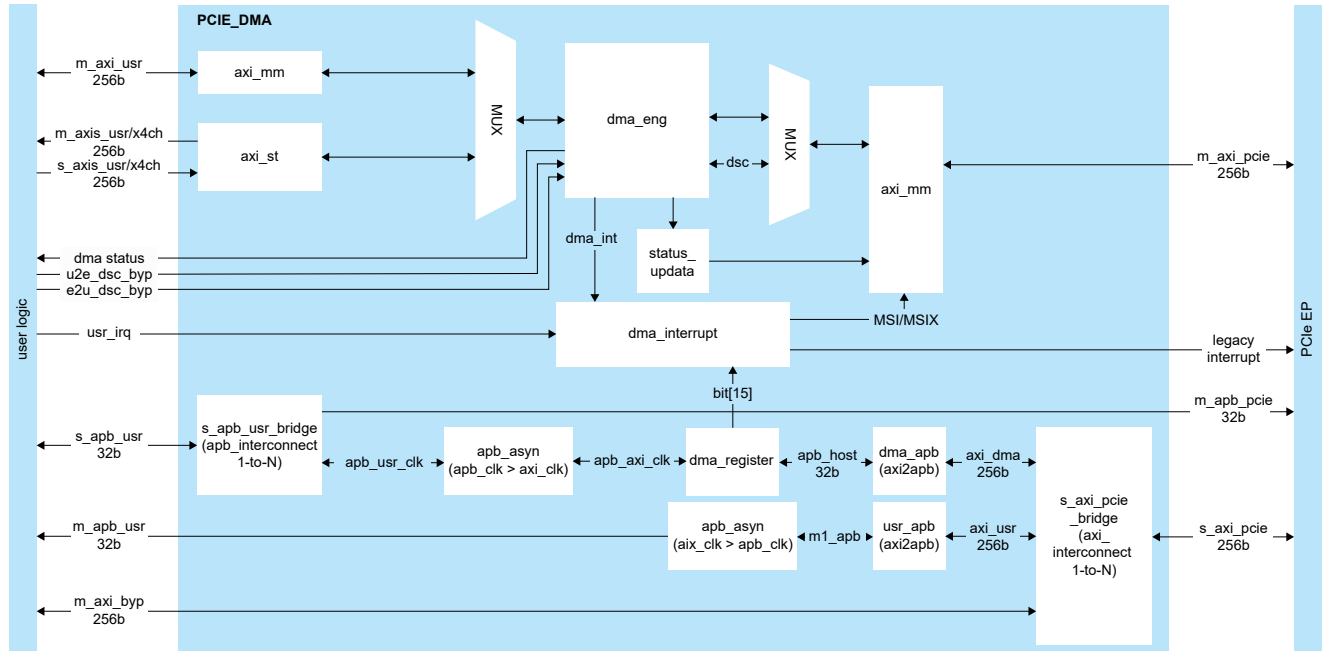
Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes are available on the [Efinity Downloads](#) page under each Efinity® software release version.

Functional Description

This section outlines some of the capabilities of the PCIe SGDMA IP core.

Figure 1: SGDMA Block Diagram



The SGDMA IP enables data transfer between the host memory and FPGA user logic via the PCIe interface. It supports two transfer modes:

- Host-to-Card (HTC)
- Card-to-Host (CTH)

Both modes support bidirectional data transfer between the FPGA and the host. Using a chained descriptor queue mechanism, the SGDMA engine efficiently and flexibly handles data transfer tasks.

The operational flow for SGDMA is as follows:

1. **Descriptor Configuration:** The driver on the host side configures software descriptors and builds the transfer queue.
2. **Data Transfer:** The SGDMA engine actively reads the descriptor commands from the queue and performs the data transfer operations.
3. **Full-Duplex Processing:** The SGDMA engine supports multitasking with parallel processing. The HTC and CTH engines operate independently to handle data transfers from host to FPGA and from FPGA to host, respectively.

On the user logic side, the SGDMA provides either an AXI4 memory-mapped (AXI4-MM) master interface or an AXI4-Stream interface for integration with user logic. On the PCIe controller side, it connects via an AXI4-MM master interface. The SGDMA performs data transfers based on descriptors, which contain information such as source address, destination address, data length, and the address of the next descriptor in a chain. The SGDMA fetches and parses descriptors from host memory, and upon completing the transfer, it notifies the host of the transfer status via interrupt.

In addition, this IP integrates an SGDMA bypass mode that allows the host to access device memory directly for read/write operations without going through the SGDMA engine. This is suitable for low-latency applications involving small data volumes. Working with the SGDMA transfer engine, it provides a flexible data access solution for the system.

Ports

Table 3: Interface Bus

Signal Name	Direction	Description
Global Clock and Reset		
axi_clk	Input	Increase clock frequency to 250 MHz.
apb_clk	Input	Clock for APB master and slave interfaces, generated by logic. Clock frequency <200 MHz.
dma_rstn	Input	SGDMA IP asynchronous reset signal, set to 0 for reset.
apb_rstn	Input	Asynchronous active-low reset for the APB master and slave interfaces.
m_axi4_pcie Bus		
m_axi_pcie_awready	Input	Write address interface ready signal.
m_axi_pcie_awvalid	Output	Write address interface signal: 1: Valid. 0: Invalid.
m_axi_pcie_awaddr[63:0]	Output	Address signal line, transmits address information, gives the first address of a burst transaction.
m_axi_pcie_awid[7:0]	Output	Transaction identifier, used in out-of-order transactions, identifies the write address group; default is 0.
m_axi_pcie_awlen[7:0]	Output	AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.
m_axi_pcie_awsz[2:0]	Output	Number of bytes per transfer, related to WDATA width. For example, if WDATA is 32-bit, AWSIZE = $\log_2(32/8) = 2$.
m_axi_pcie_wready	Input	Write data interface ready signal.
m_axi_pcie_wvalid	Output	Write data interface signal: 1: Valid. 0: Invalid.
m_axi_pcie_wdata[255:0]	Output	Data signal line, transmits data information.
m_axi_pcie_wdata_par[63:0]	Output	Reserved.
m_axi_pcie_wstrb[63:0]	Output	Data bus valid byte control. For a 32-bit bus, WSTRB = 4'b0010 means data in WDATA[15:8] is valid. If all 32 bits of WDATA are valid, WSTRB should be 4'b1111.
m_axi_pcie_wlast	Output	Asserted high to indicate the last data in a burst transaction.
m_axi_pcie_bready	Output	Write response interface ready signal.
m_axi_pcie_bvalid	Input	Write response interface valid signal: 1: Valid. 0: Invalid.

Signal Name	Direction	Description
m_axi_pcie_bresp[1:0]	Input	Write response status: 0: OKEY (normal access successful). 1: EXOKEY. 2: SLVERR (slave error). 3: DECERR (decode error, e.g., no slave address).
m_axi_pcie_bresp_par	Input	Write response status parity.
m_axi_pcie_bid[7:0]	Input	Transaction identifier.
m_axi_pcie_bid_par	Input	Transaction identifier parity.
m_axi_pcie_arready	Input	Read address interface ready signal.
m_axi_pcie_arvalid	Output	Read address interface valid signal: 1: Valid. 0: Invalid.
m_axi_pcie_araddr[63:0]	Output	Address signal line, transmits address information.
m_axi_pcie_arid[7:0]	Output	Read address ID.
m_axi_pcie_arlen[7:0]	Output	Burst read length, AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.
m_axi_pcie_arsize[2:0]	Output	Number of bytes per transfer, related to RDATA width. For example, if RDATA is 32-bit, ARSIZE = $\log_2(32/8) = 2$.
m_axi_pcie_rready	Output	Read data ready signal.
m_axi_pcie_rvalid	Input	Read data valid signal: 1: Valid. 0: Invalid.
m_axi_pcie_rid[7:0]	Input	Transaction identifier.
m_axi_pcie_rid_par	Input	Transaction identifier parity.
m_axi_pcie_rdata[255:0]	Input	Data signal line, transmits data information.
m_axi_pcie_rdata_par[63:0]	Input	Reserved.
m_axi_pcie_rresp[1:0]	Input	Read response, indicates the status of the read transfer.
m_axi_pcie_rresp_par	Input	Read response parity.
m_axi_pcie_rlast	Input	Asserted high when the last valid data in a read burst transfer is transmitted.
s_axi4_pcie Bus		
s_axi_pcie_awready	Output	Write data interface ready signal.
s_axi_pcie_awvalid	Input	Write data valid signal: 1: Valid. 0: Invalid.
s_axi_pcie_awaddr[63:0]	Input	Address signal line, transmits address information, gives the first address of a burst transaction.
s_axi_pcie_awid[7:0]	Input	Transaction identifier, used in out-of-order transactions, identifies the write address group.
s_axi_pcie_awlen[7:0]	Input	AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.

Signal Name	Direction	Description
s_axi_pcie_awsiz[2:0]	Input	Number of bytes per transfer, related to WDATA width. For example, if WDATA is 32-bit, AWSIZE = $\log_2(32/8) = 2$.
s_axi_pcie_wready	Output	Write address interface ready signal.
s_axi_pcie_wvalid	Input	Write address interface valid signal: 1: Valid. 0: Invalid.
s_axi_pcie_wdata[255:0]	Input	Data signal line, transmits data information.
s_axi_pcie_wstrb[31:0]	Input	Data bus valid byte control. For example, for a 32-bit bus, WSTRB = 4'b0010 means data in WDATA[15:8] is valid. If all 32 bits of WDATA are valid, WSTRB should be 4'b1111.
s_axi_pcie_wlast	Input	Asserted high to indicate the last data in a burst transaction.
s_axi_pcie_bready	Input	Write response interface ready signal.
s_axi_pcie_bvalid	Output	Write response interface valid signal: 1: Valid. 0: Invalid.
s_axi_pcie_bresp[1:0]	Output	Write response status: 0: OKEY (normal access successful). 1: EXOKEY. 2: SLVERR (slave error). 3: DECERR (decode error, e.g., no slave address).
s_axi_pcie_bresp_par	Output	Response parity bit.
s_axi_pcie_bid[7:0]	Output	Transaction identifier.
s_axi_pcie_bid_par	Output	Transaction identifier parity.
s_axi_pcie_arready	Output	Read address interface ready signal.
s_axi_pcie_arvalid	Input	Read address interface valid signal: 1: Valid. 0: Invalid.
s_axi_pcie_araddr[63:0]	Input	Address signal line, transmits address information.
s_axi_pcie_arid[7:0]	Input	Transaction identifier.
s_axi_pcie_arlen[7:0]	Input	Burst read length, AXI4 extended burst length supports INCR burst types for 1 to 256 transfers. Burst transfers have the following rules: <ul style="list-style-type: none"> • Wrapping burst length must be 2, 4, 8, or 16. • Bursts cannot cross 4KB boundaries. • Early termination of burst transfers is not supported.
s_axi_pcie_arsize[2:0]	Input	Number of bytes per transfer, related to RDATA width. For example, if RDATA is 32-bit, ARSIZE = $\log_2(32/8) = 2$.
s_axi_pcie_rready	Input	Read data ready signal.
s_axi_pcie_rvalid	Output	Read data interface valid signal: 1: Valid. 0: Invalid.
s_axi_pcie_rid[7:0]	Output	Transaction identifier.

Signal Name	Direction	Description
s_axi_pcie_rid_par	Output	Transaction identifier parity.
s_axi_pcie_rdata[255:0]	Output	Data signal line, transmits data information.
s_axi_pcie_rresp[1:0]	Output	Read response, indicates the status of the read transfer.
s_axi_pcie_rresp_par	Output	Read response parity.
s_axi_pcie_rlast	Output	Asserted high when the last valid data in a read burst transfer is transmitted.
m_axi4_user Bus		
m_axi_usr_awready	Input	Write address interface ready signal.
m_axi_usr_awvalid	Output	Write address interface valid signal: 1: Valid. 0: Invalid.
m_axi_usr_awaddr[63:0]	Output	Address signal line, transmits address information, gives the first address of a burst transaction.
m_axi_usr_awid[7:0]	Output	Transaction identifier, used in out-of-order transactions, identifies the write address group.
m_axi_usr_awlen[7:0]	Output	AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.
m_axi_usr_awsz[2:0]	Output	Number of bytes per transfer, related to WDATA width.
m_axi_usr_wready	Input	Write data interface ready signal.
m_axi_usr_wvalid	Output	Write data valid signal: 1: Valid. 0: Invalid.
m_axi_usr_wdata [AXI_DATA_WIDTH - 1:0]	Output	Data signal line, transmits data information.
m_axi_usr_wstrb [AXI_DATA_WIDTH/8-1:0]	Output	Data bus valid byte control. For example, for a 32-bit bus, WSTRB = 4'b0010 means data in WDATA[15:8] is valid. If all 32 bits of WDATA are valid, WSTRB should be 4'b1111.
m_axi_usr_wlast	Output	Asserted high when the last valid data in a write burst transfer is transmitted.
m_axi_usr_bready	Output	Write response interface ready signal.
m_axi_usr_bvalid	Input	Write response interface valid signal: 1: Valid. 0: Invalid.
m_axi_usr_bresp[1:0]	Input	Write response status: 0: OKEY (normal access successful). 1: EXOKEY. 2: SLVERR (slave error). 3: DECERR (decode error, e.g., no slave address).
m_axi_usr_bid[7:0]	Input	Transaction identifier.
m_axi_usr_arready	Input	Read address interface ready signal.

Signal Name	Direction	Description
m_axi_usr_arvalid	Output	Read address interface valid signal: 1: Valid. 0: Invalid.
m_axi_usr_araddr[63:0]	Output	Address signal line, transmits address information.
m_axi_usr_arid[7:0]	Output	Transaction identifier.
m_axi_usr_arlen[7:0]	Output	Burst read length, AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.
m_axi_usr_arsize[2:0]	Output	Number of bytes per transfer in a read burst.
m_axi_usr_rready	Output	Read data interface ready signal.
m_axi_usr_rvalid	Input	Read data valid signal: 1: Valid. 0: Invalid.
m_axi_usr_rid[7:0]	Input	Transaction identifier.
m_axi_usr_rdata [AXI_DATA_WIDTH -1:0]	Input	Data signal line, transmits data information.
m_axi_usr_rresp[1:0]	Input	Read response, indicates the status of the read transfer.
m_axi_usr_rlast	Input	Asserted high when the last valid data in a read burst transfer is transmitted.
m_apb_usr Bus		
m_apb_usr_pready	Input	Ready signal used by the slave to extend the APB data transfer cycle. Active high.
m_apb_usr_psel	Output	Transfer selection signal from master to slave. Active high.
m_apb_usr_pwrite	Output	Read/write indication signal. High for write, low for read.
m_apb_usr_penable	Output	Enable signal indicating the second and subsequent cycles of an APB transfer. Active high.
m_apb_usr_paddr[31:0]	Output	Address bus, up to 32 bits.
m_apb_usr_pwdata[31:0]	Output	Write data bus, up to 32 bits.
m_apb_usr_prdata[31:0]	Input	Read data bus, up to 32 bits.
m_apb_usr_pslverror	Input	Transfer failure indication signal returned by the slave. Optional. Active high.
AXI4-Stream Bus		
CTH		
s_axis_usr_tready	Output	Asserted high to indicate the SGDMA side is ready to receive data. When both tready and tvalid are high, data is transferred. When tvalid is high but tready is low, the user side must hold the valid data until tready goes high.
s_axis_usr_tvalid	Input	Asserted high by the user side to indicate that tdata is valid.
s_axis_usr_tlast	Input	Asserted high by the logic side to indicate the last data in a burst transfer.
s_axis_usr_tdata [AXI_DATA_WIDTH -1:0]	Input	Transfer data bus. Bit width is defined by the AXI_DATA_WIDTH parameter.

Signal Name	Direction	Description
s_axis_usr_tkeep [AXI_DATA_WIDTH/8-1:0]	Input	Byte-valid indicator for the last data word in a packet. Each bit corresponds to one byte: 1: Valid. 0: Invalid.
HTC		
m_axis_usr_tready	Input	Asserted high to indicate the user side is ready to receive data. When both tready and tvalid are high, data is transferred. When tvalid is high but tready is low, the SGDMA side must hold the valid data until tready goes high.
m_axis_usr_tvalid	Output	Asserted high by the SGDMA to indicate that tdata is valid.
m_axis_usr_tlast	Output	Asserted high to indicate the last data in a burst transfer.
m_axis_usr_tdata [AXI_DATA_WIDTH - 1:0]	Output	Transfer data bus. Bit width is defined by the AXI_DATA_WIDTH parameter.
m_axis_usr_tkeep [AXI_DATA_WIDTH/8 - 1:0]	Output	Byte-valid indicator for the last data word in a packet. Each bit corresponds to one byte: 1: Valid. 0: Invalid.
m_apb_pcie Bus		
m_apb_pcie_pready	Input	Ready signal used by the slave to extend the APB data transfer cycle. Active high.
m_apb_pcie_psel	Output	Transfer selection signal from master to slave. Active high.
m_apb_pcie_pwrite	Output	Read/write indication signal. High for write, low for read.
m_apb_pcie_penable	Output	Enable signal indicating the second and subsequent cycles of an APB transfer. Active high.
m_apb_pcie_paddr[23:0]	Output	Address bus, up to 24 bits.
m_apb_pcie_pwdata[31:0]	Output	Write data bus, up to 32 bits.
m_apb_pcie_pwdata_par[3:0]	Output	End-to-end parity bits for m_apb_pcie_pwdata.
m_apb_pcie_pstrb[3:0]	Output	Write strobe signals for sparse data transfer on the write data bus.
m_apb_pcie_pstrb_par	Output	End-to-end parity bits for m_apb_pcie_pstrb.
m_apb_pcie_prdata[31:0]	Input	Read data bus, up to 32 bits.
m_apb_pcie_prdata_par[3:0]	Input	End-to-end parity bits for m_apb_pcie_prdata.
m_apb_pcie_pslverror	Input	Transfer failure indication signal returned by the slave. Optional. Active high.
s_apb_usr Bus		
s_apb_usr_pready	Output	Ready signal used by the slave to extend the APB data transfer cycle. Active high.
s_apb_usr_psel	Input	Transfer selection signal from master to slave. Active high.
s_apb_usr_pwrite	Input	Read/write indication signal. High for write, low for read.
s_apb_usr_penable	Input	Enable signal indicating the second and subsequent cycles of an APB transfer. Active high.
s_apb_usr_paddr[31:0]	Input	Address bus, up to 32 bits.

Signal Name	Direction	Description
s_apb_usr_pwdata[31:0]	Input	Write data bus, up to 32 bits.
s_apb_usr_prdata[31:0]	Output	Read data bus, up to 32 bits.
s_apb_usr_pslverror	Output	Transfer failure indication signal returned by the slave. Optional. Active high.
m_byp_axi4 Bus		
m_axi_byp_awready	Input	Write address interface ready signal.
m_axi_byp_awvalid	Output	Write address interface valid signal: 1: Valid. 0: Invalid.
m_axi_byp_awaddr [AXI_ADDR_WIDTH - 1:0]	Output	Address signal line, transmits address information, gives the first address of a burst transaction.
m_axi_byp_awid [AXI_ID_WIDTH - 1:0]	Output	Transaction identifier, used in out-of-order transactions, identifies the write address group.
m_axi_byp_awlen[7:0]	Output	AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.
m_axi_byp_awsz[2:0]	Output	Number of bytes per transfer, related to WDATA width. For example, if WDATA is 32-bit, AWSIZE = $\log_2(32/8) = 2$.
m_axi_byp_wready	Input	Write data interface ready signal.
m_axi_byp_wvalid	Output	Write data valid signal: 1: Valid. 0: Invalid.
m_axi_byp_wdata [AXI_DATA_WIDTH - 1:0]	Output	Data signal line, transmits data information.
m_axi_byp_wstrb [AXI_DATA_WIDTH/8 - 1:0]	Output	Data bus valid byte control. For a 32-bit bus, WSTRB = 4'b0010 means data in WDATA[15:8] is valid. If all 32 bits of WDATA are valid, WSTRB should be 4'b1111.
m_axi_byp_wlast	Output	Asserted high to indicate the last data in a burst transaction.
m_axi_byp_bready	Output	Write response interface ready signal.
m_axi_byp_bvalid	Input	Write response interface valid signal: 1: Valid. 0: Invalid.
m_axi_byp_bresp[1:0]	Input	Write response status: 0: OKEY (normal access successful). 1: EXOKEY. 2: SLVERR (slave error). 3: DECERR (decode error, e.g., no slave address).
m_axi_byp_bid [AXI_ID_WIDTH - 1:0]	Input	Transaction identifier.
m_axi_byp_arready	Input	Read address interface ready signal.
m_axi_byp_arvalid	Output	Read address interface valid signal: 1: Valid. 0: Invalid.

Signal Name	Direction	Description
m_axi_byp_araddr [AXI_ADDR_WIDTH -1:0]	Output	Address signal line, transmits address information.
m_axi_byp_arid [AXI_ID_WIDTH -1:0]	Output	Transaction identifier.
m_axi_byp_arsize[7:0]	Output	Burst read length, AXI4 extended burst length supports INCR burst types for 1 to 256 transfers.
m_axi_byp_rready	Output	Read data ready signal.
m_axi_byp_rvalid	Input	Read data interface valid signal: 1: Valid. 0: Invalid.
m_axi_byp_rid [AXI_ID_WIDTH -1:0]	Input	Transaction identifier.
m_axi_byp_rdata [AXI_DATA_WIDTH -1:0]	Input	Data signal line, transmits data information.
m_axi_byp_rresp[1:0]	Input	Read response, indicates the status of the read transfer.
m_axi_byp_rlast	Input	Asserted high when the last valid data in a read burst transfer is transmitted.
SGDMA Engine Status Indicators and Interrupt Signals		
Status Indicators		
cfg_max_payload_size[2:0]	Input	MPS (Max Payload Size) indicator from PCIe IP. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. Others: Reserved.
cfg_max_read_req_size[2:0]	Input	MRRS (Max Read Request Size) indicator from PCIe IP. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes. Others: Reserved.
HTC_STS[8*CHN_NUM-1:0]	Output	Working status of the HTC engine. Bit[7:4]: Reserved Bit[3]: Run Bit[2]: Interrupt Bit[1]: DSC done Bit[0]: Busy

Signal Name	Direction	Description
CTH_STS[8*CHN_NUM-1:0]	Output	Working status of the CTH engine. Bit[7:4]: Reserved Bit[3]: Run Bit[2]: Interrupt Bit[1]: DSC done Bit[0]: Busy
Interrupt Signals		
cfg_msi_enable	Input	MSI interrupt enable signal. 0: Disabled. Other values: Enabled.
cfg_msix_enable	Input	MSI-X interrupt enable signal. 0: Disabled. Other values: Enabled.
legacy_irq[3:0]	Output	Legacy interrupt request signal. Held high until acknowledged and cleared. Bit[0]: INTA interrupt request Bit[1]: INTB interrupt request Bit[2]: INTC interrupt request Bit[3]: INTD interrupt request
legacy_irq_ack	Input	Legacy interrupt acknowledge signal.
usr_irq [NUM_USR_IRQ-1:0]	Input	User interrupt request signals. Held high until acknowledged and cleared.
usr_irq_ack [NUM_USR_IRQ-1:0]	Output	User interrupt acknowledge signal. This only means that an interrupt request has been received. If the configuration is wrong, the interrupt request may fail to be sent.

Register List

Table 4: Register List

Base Address	Channel Offset	Dword Offset	Name	R/W	Description
0x00000000		0x00	cfg_identifier	RO	Configuration identifier
		0x04	reg_user_max_payload	RO	User-defined maximum payload register
		0x08	reg_user_max_read_req	RO	User-defined maximum read request register
		0x0C	reg_write_timeout	RO	Write timeout register
		0x18	msi_enable&msix_enable	RO	MSI/MSI-X interrupt enable
		0x1C	max_payload	RO	Maximum payload register
		0x20	max_read_req	RO	Maximum read request register
		0x24	clk_pciew	RO	PCIe width: (0, 1, 2, 3) represent (64, 128, 256, 512)
0x00001000		0x00	dsc_identifier	RO	Descriptor identifier
		0x04	dsc_crd_en	RW	Enable signal for HTC/CTH channel credit
		0x0C	dsc_stop		Descriptor fetch stop signal for the corresponding channel
		0x18	dsc_restart	RW	Reset signal for HTC/CTH channel credit
0x00002000		0x00	irq_identifier	RO	Interrupt identifier
		0x04	user_irq	RO	User interrupt request indicator
		0x08	dma_irq	RO	SGDMA interrupt request indicator
		0x0C	user_irq_i	RO	User interrupt pending indicator
		0x10	dma_irq_i	RO	SGDMA interrupt pending indicator
		0x14	user_irq_lut	RW	User interrupt vector lookup
		0x18	user_irq_lut	RW	User interrupt vector lookup
		0x1C	user_irq_lut	RW	User interrupt vector lookup
		0x20	user_irq_lut	RW	User interrupt vector lookup
		0x24	dma_irq_lut	RW	SGDMA interrupt vector lookup
		0x28	dma_irq_lut	RW	SGDMA interrupt vector lookup
		0x2C	user_en	RW	User interrupt enable register
		0x38	dma_en	RW	SGDMA interrupt enable register

Base Address	Channel Offset	Dword Offset	Name	R/W	Description
0x00003000	0x000 / 0x100 / 0x200 / 0x300	0x00	htc_identifier	RO	HTC identifier
		0x04	htc_dsc_adj	RW	Number of adjacent descriptors for HTC
		0x08	htc_dsc_crd	RW	HTC channel credit value
		0x0C	htc_dsc_addr_l	RW	HTC descriptor lower base address
		0x10	htc_dsc_addr_h	RW	HTC descriptor upper base address
		0x14	htc_dma_wb_addr_l	RW	HTC descriptor writeback lower address
		0x18	htc_dma_wb_addr_h	RW	HTC descriptor writeback upper address
		0x1C	htc_dma_ctl	RW	HTC engine control register
		0x28	htc_dma_status	RO	HTC engine status register
		0x30	htc_dma_dsc_compl_cnt	RO	HTC engine SGDMA completed descriptor count register
		0x38	htc_dma_intr_mask	RW	HTC engine SGDMA interrupt enable register
0x00004000	0x000 / 0x100 / 0x200 / 0x300	0x00	cth_identifier	RO	CTH identifier
		0x04	cth_dsc_adj	RW	Number of adjacent descriptors for CTH
		0x08	cth_dsc_crd	RW	CTH channel credit value
		0x0C	cth_dsc_addr_l	RW	CTH descriptor lower base address
		0x10	cth_dsc_addr_h	RW	CTH descriptor upper base address
		0x14	cth_dma_wb_addr_l	RW	CTH descriptor writeback lower address
		0x18	cth_dma_wb_addr_h	RW	CTH descriptor writeback upper address
		0x1C	cth_dma_ctl	RW	CTH engine control register
		0x28	cth_dma_status	RW	CTH engine status register
		0x30	cth_dma_dsc_compl_cnt	RW	CTH engine SGDMA completed descriptor count register
		0x38	cth_dma_intr_mask	RW	CTH engine SGDMA interrupt enable register
0x00008000		0x00	Interrupt vector table	RW	MSI and MSIX interrupt vector table
0x80000000					Bypass DMA register space. Use this base address space with 24-bit APB address to access the transceiver hard block APB register.

*cfg_identifier (0x0000)***Table 5: *cfg_identifier (0x0000)***

Bit(s)	Default	Access	Description
31:24	8'h0	RO	Unused
23:0	24'h100003	RO	Configuration register group identifier

*reg_user_max_payload (0x0004)***Table 6: *reg_user_max_payload (0x0004)***

Bit(s)	Default	Access	Description
6:4	3'h5	RW	Programmed maximum payload size issued to the user application. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes.
3	-	-	Reserved
2:0	3'h5	RO	Actual maximum payload size issued to the user application. This value may be lower than user_prg_payload due to IP configuration or data path width. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes.

*reg_user_max_read_req (0x0008)***Table 7: reg_user_max_read_req (0x0008)**

Bit(s)	Default	Access	Description
6:4	3'h5	RW	Maximum read request size issued to the user application. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes.
3	-	-	Reserved
2:0	3'h5	RO	Actual maximum read request size issued to the user application. This value may be lower than user_max_read due to PCIe configuration or data path width. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes.

*reg_write_timeout (0x000C)***Table 8: reg_write_timeout (0x000C)**

Bit(s)	Default	Access	Description
4:0	5'h0	RW	Write flush timeout. Applies to the AXI4-Stream CTH channel. This register specifies the number of clock cycles the channel waits for data before flushing received write data from PCIe. This operation closes the descriptor and triggers a writeback. A value of 0 disables the timeout. Timeout period = 2 ^{value} clock cycles.

*msi_enable and msix_enable (0x0018)***Table 9: msi_enable & msix_enable (0x0018)**

Bit(s)	Default	Access	Description
1	-	RO	MSI interrupt enable.
0	-	RO	MSI-X interrupt enable.

max_payload (0x001C)

Table 10: *max_payload (0x001C)*

Bit(s)	Default	Access	Description
2:0	-	RO	Maximum write payload size. This value is the minimum of the PCIe IP Maximum Payload Size (MPS) and the SGDMA/Bridge Subsystem for PCIe parameter. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes.

max_read_req (0x0020)

Table 11: *max_read_req (0x0020)*

Bit(s)	Default	Access	Description
2:0	-	RO	Maximum read request size. This value is the minimum of the PCIe IP MPS and the SGDMA/Bridge Subsystem for PCIe parameter. 3'b000: 128 bytes. 3'b001: 256 bytes. 3'b010: 512 bytes. 3'b011: 1024 bytes. 3'b100: 2048 bytes. 3'b101: 4096 bytes.

clk_pciew (0x0024)

Table 12: *clk_pciew (0x0024)*

Bit(s)	Default	Access	Description
2:0	-	RO	PCIe AXI4-Stream interface width. 0: 64-bit 1: 128-bit 2: 256-bit 3: 512-bit

dsc_identifier (0x1000)

Table 13: *dsc_identifier (0x1000)*

Bit(s)	Default	Access	Description
31:24	8'h00	RO	Unused
23:0	24'h100100	RO	Descriptor register group identifier

*dsc_crd_en (0x1004)***Table 14: dsc_crd_en (0x1004)**

Bit(s)	Default	Access	Description
31:20	-	-	Reserved
19:16	4'h0	RW	HTC engine descriptor credit enable. One bit per HTC channel; active high. NOT SUPPORTED IN THIS VERSION.
15:4	-	-	Reserved
3:0	4'h0	RW	CTH engine descriptor credit enable. One bit per CTH channel; active high. NOT SUPPORTED IN THIS VERSION.

*dsc_stop (0x100C)***Table 15: dsc_stop (0x100C)**

Bit(s)	Default	Access	Description
31:20	-	-	Reserved
19:16	4'h0	RW	Pause control bits for the descriptor engine to stop fetching descriptors from CTH engines. One bit per CTH channel; setting a bit high pauses fetching, setting it low resumes operation.
15:4	-	-	Reserved
3:0	1'h0	RW	Pause control bits for the descriptor engine to stop fetching descriptors from HTC engines. One bit per HTC channel; setting a bit high pauses fetching, setting it low resumes operation.

*dsc_restart (0x1018)***Table 16: dsc_restart (0x1018)**

Bit(s)	Default	Access	Description
31:20	-	-	Reserved
19:16	4'h0	RW	Active high reset signal for HTC descriptor engine to request for channel credit. One bit per HTC channel.
15:4	-	-	Reserved
3:0	4'h0	RW	Active high reset signal for CTH descriptor engine to request for channel credit. One bit per CTH channel.

*irq_identifier (0x2000)***Table 17: irq_identifier (0x2000)**

Bit(s)	Default	Access	Description
31:24	8'h0	RO	Unused
23:0	24'h100002	RO	Interrupt request register group identifier

*usr_irq***Table 18: *usr_irq***

Bit(s)	Default	Access	Description
[USER_INTERRUPT_NUM:0]	'h0	RO	User interrupt request. This register reflects the interrupt status only when both the interrupt source and the corresponding enable mask register are asserted.

*dma_irq (0x2008)***Table 19: *dma_irq (0x2008)***

Bit(s)	Default	Access	Description
[31:NUM_ENGINE]	–	–	Reserved
[NUM_ENGINE-1:0]	'h0	RO	SGDMA interrupt request status for HTC and CTH engines. NUM_ENGINE represents the total number of HTC and CTH engines. Each bit corresponds to the interrupt request of one engine. A bit is asserted only when both the interrupt source and its corresponding interrupt enable bit are active. The mapping of each bit to the respective engine is illustrated in the reference diagram. HTC engines are mapped starting from bit [0], followed by CTH engines.

*user_irq_i (0x200C)***Table 20: *user_irq_i (0x200C)***

Bit(s)	Default	Access	Description
[USER_INTERRUPT_NUM:0]	'h0	RO	User interrupt pending. This register indicates the presence of a pending event. The pending event can be cleared by removing the event cause condition on the source component.

*dma_irq_i (0x2010)***Table 21: dma_irq_i (0x2010)**

Bit(s)	Default	Access	Description
[31:NUM_ENGINE]	-	-	Reserved
[NUM_ENGINE-1:0]	'h0	RO	<p>Interrupt pending status for each engine. Each bit corresponds to a read or write engine. This register indicates that a pending interrupt condition exists.</p> <p>The pending status can be cleared by removing the cause of the interrupt condition at the source component.</p> <p>The bits for HTC (Host to Card) engines always start from bit 0.</p> <p>The bits for CTH (Card to Host) engines follow the last HTC bit.</p>

*user_irq_lut (0x2014)***Table 22: user_irq_lut (0x2014)**

Bit(s)	Default	Access	Description
28:24	5'h0	RW	<p>Vector 3</p> <p>Vector ID used when user IRQ usr_irq[3] triggers an interrupt.</p>
20:16	5'h0	RW	<p>Vector 2</p> <p>Vector ID used when user IRQ usr_irq[2] triggers an interrupt.</p>
12:8	5'h0	RW	<p>Vector 1</p> <p>Vector ID used when user IRQ usr_irq[1] triggers an interrupt.</p>
4:0	5'h0	RW	<p>Vector 0</p> <p>Vector ID used when user IRQ usr_irq[0] triggers an interrupt.</p>

*user_irq_lut (0x2018)***Table 23: user_irq_lut (0x2018)**

Bit(s)	Default	Access	Description
28:24	5'h0	RW	<p>Vector 7</p> <p>Vector ID used when user IRQ usr_irq[7] triggers an interrupt.</p>
20:16	5'h0	RW	<p>Vector 6</p> <p>Vector ID used when user IRQ usr_irq[6] triggers an interrupt.</p>
12:8	5'h0	RW	<p>Vector 5</p> <p>Vector ID used when user IRQ usr_irq[5] triggers an interrupt.</p>
4:0	5'h0	RW	<p>Vector 4</p> <p>Vector ID used when user IRQ usr_irq[4] triggers an interrupt.</p>

*user_irq_lut (0x201C)***Table 24: user_irq_lut (0x201C)**

Bit(s)	Default	Access	Description
28:24	5'h0	RW	Vector 11 Vector ID used when user IRQ usr_irq[11] triggers an interrupt.
20:16	5'h0	RW	Vector 10 Vector ID used when user IRQ usr_irq[10] triggers an interrupt.
12:8	5'h0	RW	Vector 9 Vector ID used when user IRQ usr_irq[9] triggers an interrupt.
4:0	5'h0	RW	Vector 8 Vector ID used when user IRQ usr_irq[8] triggers an interrupt.

*user_irq_lut (0x2020)***Table 25: user_irq_lut (0x2020)**

Bit(s)	Default	Access	Description
28:24	5'h0	RW	Vector 15 Vector ID used when user IRQ usr_irq[15] triggers an interrupt.
20:16	5'h0	RW	Vector 14 Vector ID used when user IRQ usr_irq[14] triggers an interrupt.
12:8	5'h0	RW	Vector 13 Vector ID used when user IRQ usr_irq[13] triggers an interrupt.
4:0	5'h0	RW	Vector 12 Vector ID used when user IRQ usr_irq[12] triggers an interrupt.

*dma_irq_lut (0x2024)***Table 26: dma_irq_lut (0x2024)**

Bit(s)	Default	Access	Description
28:24	5'h0	RW	Vector 3 Vector ID used when SGDMA channel 3 triggers an interrupt.
20:16	5'h0	RW	Vector 2 Vector ID used when SGDMA channel 2 triggers an interrupt.
12:8	5'h0	RW	Vector 1 Vector ID used when SGDMA channel 1 triggers an interrupt.
4:0	5'h0	RW	Vector 0 Vector ID used when SGDMA channel 0 triggers an interrupt.

*dma_irq_lut (0x2028)***Table 27: dma_irq_lut (0x2028)**

Bit(s)	Default	Access	Description
28:24	5'h0	RW	Vector 7 Vector ID used when SGDMA channel 7 triggers an interrupt.
20:16	5'h0	RW	Vector 6 Vector ID used when SGDMA channel 6 triggers an interrupt.
12:8	5'h0	RW	Vector 5 Vector ID used when SGDMA channel 5 triggers an interrupt.
4:0	5'h0	RW	Vector 4 Vector ID used when SGDMA channel 4 triggers an interrupt.

*user_en (0x202C)***Table 28: user_en (0x202C)**

Bit(s)	Default	Access	Description
[USER_INTERRUPT_NUM:0]	'h0	RW	user_int_enmask User interrupt enable mask 0: Interrupt generation is blocked when the user interrupt source is asserted. 1: An interrupt is generated on the rising edge of the user interrupt source. If both the "Enable Mask" and the source are set, a user interrupt is generated.

*dma_en (0x2038)***Table 29: dma_en (0x2038)**

Bit(s)	Default	Access	Description
[DMA_INTERRUPT_NUM:0]	-	RW	channel_int_enmask SGDMA interrupt enable mask. Each read/write engine is mapped to one bit. 0: Interrupt generation is blocked when the interrupt source is asserted. HTC bits always start from bit 0. CTH bits are placed after the last HTC bit, depending on the HTC_CH_NUM parameter. 1: An interrupt is generated on the rising edge of the interrupt source. If both the enmask bit and the source are asserted, an interrupt is generated.

*htc_identifier (0x3000)***Table 30: *htc_identifier (0x3000)***

Bit(s)	Default	Access	Description
31:16	16'h3100	RO	HTC channel register group identifier
15:13	3'h0	RO	Unused
12	-	RO	DMA engine user interface type. 1'b1: AXI-Stream Interface 1'b0: AXI-MM Interface
11:8	-	RO	HTC Channel ID
7:0	8'h00	RO	Unused

*htc_dsc_adj (0x3004)***Table 31: *htc_dsc_adj (0x3004)***

Bit(s)	Default	Access	Description
[31:6]	-	-	Reserved, fixed to 0.
[5:0]	6'h0	RW	Number of descriptors following the starting descriptor in the first descriptor block of the HTC engine.

*htc_dsc_crd (0x3008)***Table 32: *htc_dsc_crd (0x3008)***

Bit(s)	Default	Access	Description
[31:10]	-	-	Reserved, fixed to 0.
[9:0]	10'h0	RW	Descriptor credit value to be added for the HTC engine. This register is valid only when enabled per channel in the Descriptor Credit Mode register.

*htc_dsc_addr_l (0x300C)***Table 33: *htc_dsc_addr_l (0x300C)***

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Lower 32 bits of HTC descriptor base address

*htc_dsc_addr_h (0x3010)***Table 34: *htc_dsc_addr_h (0x3010)***

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Upper 32 bits of HTC descriptor base address

*htc_dma_wb_addr_l (0x3014)**Table 35: htc_dma_wb_addr_l (0x3014)*

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Lower 32 bits (4-byte aligned) of the host memory address for completed HTC descriptors writeback.

*htc_dma_wb_addr_h (0x3018)**Table 36: htc_dma_wb_addr_h (0x3018)*

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Upper 32 bits (4-byte aligned) of the host memory address for completed HTC descriptors writeback.

*htc_dma_ctl (0x301C)**Table 37: htc_dma_ctl (0x301C)*

Bit(s)	Default	Access	Description
31:28	-	-	Reserved
27:23	-	-	Reserved
22:18	-	-	Reserved
17:13	-	-	Reserved
12	-	-	Reserved
11	1'b0	RW	In AXI4-Stream mode, disables HTC write-back and enables default write-back when set to 1.
10	0x0	RW	Polling mode enable. When set to 1, the HTC engine automatically writes back the completion count to a predefined memory address after processing descriptors with the Completed flag, for host polling.
9:7	-	-	Reserved
6	1'b0	RW	Enables logging of descriptor completion flags in the Status register. If the corresponding interrupt enable bit is asserted, an interrupt is triggered simultaneously.
5	1'b0	RW	Enables logging of descriptor stop flags in the Status register. If the corresponding interrupt enable bit is asserted, an interrupt is triggered simultaneously.
4	-	-	Reserved
3	-	-	Reserved
2	-	-	Reserved
1	-	-	Reserved
0	1'b0	RW	Run bit for the HTC engine. When set to 1, the engine starts data transfer. Clearing this bit stops the engine after completing the current descriptor if busy.

*htc_dma_status (0x3028)***Table 38: *htc_dma_status (0x3028)***

Bit(s)	Default	Access	Description
31:24	-	-	Reserved
23:19	-	-	Reserved
18:14	-	-	Reserved
13:9	-	-	Reserved
8:7	-	-	Reserved
6	1'b0	RW	Descriptor Completed flag status field. This field is cleared when the run bit [0] of the control register transitions from 0 to 1. When the corresponding logging enable bit in the control register is set to 1, this bit is asserted high by the HTC engine after completing the data transfer of a descriptor with the Completed flag.
5	1'b0	RW	Descriptor Stop flag status field. This field is cleared when the run bit [0] of the control register transitions from 0 to 1. When the corresponding logging enable bit in the control register is set to 1, this bit is asserted high by the HTC engine after completing the data transfer of a descriptor with the Stop flag.
4	-	-	Reserved
3	-	-	Reserved
2	-	-	Reserved
1	-	-	Reserved
0	1'b0	RO	HTC engine Busy status. This bit is set to 1 when the HTC engine is performing data transfer, and cleared to 0 when the engine is idle.

*htc_dma_dsc_compl_cnt (0x3030)***Table 39: *htc_dma_dsc_compl_cnt (0x3030)***

Bit(s)	Default	Access	Description
[31:0]	32'h0	RO	Indicates the number of descriptors that have completed data transfer. This register is cleared when the run bit [0] of the control register transitions from 0 to 1.

htc_dma_intr_mask (0x3038)

Table 40: *htc_dma_intr_mask (0x3038)*

Bit(s)	Default	Access	Description
31:24	-	-	Reserved
23:19	-	-	Reserved
18:14	-	-	Reserved
13:9	-	-	Reserved
8:7	-	-	Reserved
6	1'b0	RW	Interrupt enable bit corresponding to the status field for descriptors with the Completed flag. When set to 1, an interrupt is generated simultaneously as the status is logged in the Status register.
5	1'b0	RW	Interrupt enable bit corresponding to the status field for descriptors with the Stop flag. When set to 1, an interrupt is generated simultaneously as the status is logged in the Status register.
4	-	-	Reserved
3	-	-	Reserved
2	-	-	Reserved
1	-	-	Reserved
0	-	-	Reserved

cth_identifier (0x4000)

Table 41: *cth_identifier (0x4000)*

Bit(s)	Default	Access	Description
31:16	16'h4100	RO	CTH channel register group identifier
15:13	3'h0	RO	Unused
12	-	RO	DMA engine user interface type. 1'b1: AXI-Stream Interface 1'b0: AXI-MM Interface
11:8	-	RO	CTH Channel ID
7:0	8'h00	RO	Unused

cth_dsc_adj (0x4004)

Table 42: *cth_dsc_adj (0x4004)*

Bit(s)	Default	Access	Description
[31:6]	-	-	Reserved, fixed to 0.
[5:0]	6'h0	RW	Number of descriptors following the initial descriptor in the first descriptor block of the CTH engine in host memory.

*cth_dsc_crd (0x4008)**Table 43: cth_dsc_crd (0x4008)*

Bit(s)	Default	Access	Description
[31:10]	-	-	Reserved, fixed to 0.
[9:0]	10'h0	RW	Descriptor credit value to be added for the CTH engine. This register is valid only when enabled per channel in the Descriptor Credit Mode register.

*cth_dsc_addr_l (0x400C)**Table 44: cth_dsc_addr_l (0x400C)*

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Lower 32 bits of CTH descriptor base address

*cth_dsc_addr_h (0x4010)**Table 45: cth_dsc_addr_h (0x4010)*

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Upper 32 bits of CTH descriptor base address

*cth_dma_wb_addr_l (0x4014)**Table 46: cth_dma_wb_addr_l (0x4014)*

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Lower 32 bits (4-byte aligned) of the host memory address for completed CTH descriptors writeback.

*cth_dma_wb_addr_h (0x4018)**Table 47: cth_dma_wb_addr_h (0x4018)*

Bit(s)	Default	Access	Description
31:0	32'h00000000	RW	Upper 32 bits (4-byte aligned) of the host memory address for completed CTH descriptors writeback.

*cth_dma_ctl (0x401C)*Table 48: *cth_dma_ctl (0x401C)*

Bit(s)	Default	Access	Description
31:28	-	-	Reserved
27:23	-	-	Reserved
22:18	-	-	Reserved
17:13	-	-	Reserved
12	-	-	Reserved
11	1'b0	RW	When the CTH engine's user-side interface is set to AXI4-Stream mode, setting this bit to 1 disables CTH writeback information and enables the default writeback behavior.
10	0x0	RW	Writeback enable bit for polling mode. When set to 1, after the CTH engine completes data transfer for descriptors with the Completed flag, the writeback engine writes the number of completed descriptors to a preconfigured memory address for the host to check transfer completion.
9:7	-	-	Reserved
6	1'b0	RW	When set to 1, enables logging of descriptors with the Completed flag in the Status register. If the corresponding interrupt enable bit is 1, an interrupt is generated.
5	1'b0	RW	When set to 1, enables logging of descriptors with the Stop flag in the Status register. If the corresponding interrupt enable bit is 1, an interrupt is generated.
4	-	-	Reserved
3	-	-	Reserved
2	-	-	Reserved
1	-	-	Reserved
0	1'b0	RW	CTH engine run enable bit. When set to 1, the CTH engine starts data transfer. When cleared to 0, the transfer is stopped. If the engine is busy when cleared, it finishes the current descriptor before stopping.

*cth_dma_status (0x4028)***Table 49: *cth_dma_status (0x4028)***

Bit(s)	Default	Access	Description
31:24	-	-	Reserved
23:19	-	-	Reserved
18:14	-	-	Reserved
13:9	-	-	Reserved
8:7	-	-	Reserved
6	1'b0	RW	Status field for descriptors with the Completed flag. This field is cleared when bit[0] (Run) of the control register transitions from 0 to 1. When the corresponding status logging in the control register is enabled, this bit is asserted high after the CTH engine completes data transfer for descriptors with the Completed flag.
5	1'b0	RW	Status field for descriptors with the Stop flag. Same behavior as bit[6], but for descriptors with the Stop flag.
4	-	-	Reserved
3	-	-	Reserved
2	-	-	Reserved
1	-	-	Reserved
0	1'b0	RO	CTH engine Busy status. This bit is set to 1 when the CTH engine is performing data transfer, and cleared to 0 when the engine is idle.

*cth_dma_dsc_compl_cnt (0x4030)***Table 50: *cth_dma_dsc_compl_cnt (0x4030)***

Bit(s)	Default	Access	Description
[31:0]	32'h0	RO	Indicates the number of descriptors that have completed data transfer. This register is cleared when the run bit [0] of the control register transitions from 0 to 1.

*cth_dma_intr_mask (0x4038)***Table 51: *cth_dma_intr_mask (0x4038)***

Bit(s)	Default	Access	Description
31:24	-	-	Reserved
23:19	-	-	Reserved
18:14	-	-	Reserved
13:9	-	-	Reserved
8:7	-	-	Reserved
6	1'b0	RW	Interrupt enable bit corresponding to the status field for descriptors with the Completed flag. When set to 1, an interrupt is generated simultaneously as the status is logged in the Status register.
5	1'b0	RW	Interrupt enable bit corresponding to the status field for descriptors with the Stop flag. When set to 1, an interrupt is generated simultaneously as the status is logged in the Status register.
4	-	-	Reserved
3	-	-	Reserved
2	-	-	Reserved
1	-	-	Reserved
0	-	-	Reserved

MSI-X/MSI Interrupt Vector Table (0x8000)

The SGDMA IP supports MSI-X and MSI interrupts. The interrupt vector table supports up to 32 interrupt vectors. The table starts at address 0x8000, and the Pending Bit Array (PBA) is located at address 0x8FE0. The MSI interrupt address is the `pcie_dma0_control` bar address of the DMA. Details are as follows.

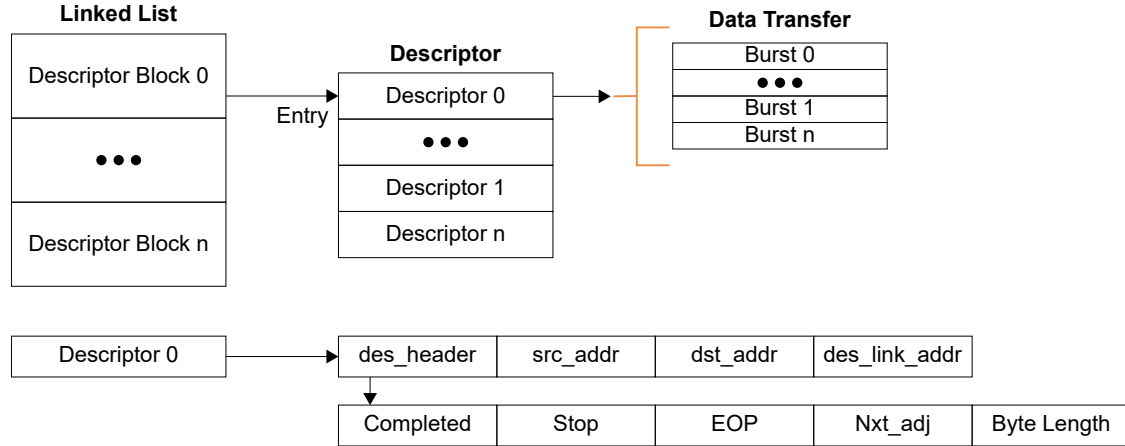
Table 52: Interrupt Vector Table

Byte Offset	Bits	Default	Access	Description
0x00	[31:0]	32'h0	RW	Lower 32 bits of the address for MSI-X interrupt vector 0.
0x04	[31:0]	32'h0	RW	Upper 32 bits of the address for MSI-X interrupt vector 0.
0x08	[31:0]	32'h0	RW	Data field for MSI-X interrupt vector 0.
0x0C	[31:0]	32'hFFFFFFFF	RW	Control field for MSI-X interrupt vector 0. <ul style="list-style-type: none"> • Bits [31:1]: Reserved. • Bit [0]: Mask bit for the MSI-X interrupt vector. When set to 1, the interrupt message is suppressed; when cleared to 0, the interrupt message is allowed.
...	-	-	-	-
0x1F0	[31:0]	32'h0	RW	Lower 32 bits of the address for MSI-X interrupt vector 31.
0x1F4	[31:0]	32'h0	RW	Upper 32 bits of the address for MSI-X interrupt vector 31.
0x1F8	[31:0]	32'h0	RW	Data field for MSI-X interrupt vector 31.
0x1FC	[31:0]	32'hFFFFFFFF	RW	Control field for MSI-X interrupt vector 31. <ul style="list-style-type: none"> • Bits [31:1]: Reserved. • Bit [0]: Mask bit for the MSI-X interrupt vector. When set to 1, the interrupt message is suppressed; when cleared to 0, the interrupt message is allowed.
0xFE0	[31:0]	32'h0	RW	Pending Bit Array (PBA). Each bit corresponds to an interrupt vector.
0xF00	[31:0]	32'h0	RW	Lower 32 bits of the address for MSI interrupt.
0xF04	[31:0]	32'h0	RW	Upper 32 bits of the address for MSI interrupt.
0xF08	[31:0]	32'h0	RW	Data field for MSI interrupt.

Descriptors

A descriptor is the basic control unit for SGDMA transfer, used to define the attributes and control information for a single data transfer. Its structure is as follows:

Figure 2: Descriptor Structure



- **Descriptor:** 32 bytes, defines the source address, destination address, and length for a single transfer. The descriptor must be 32-byte aligned in memory.
- **Descriptor Block:** Composed of multiple consecutive descriptors. The descriptors within the same block are stored consecutively in memory and cannot span a 4 KB memory boundary.
- **Descriptor List:** Composed of multiple descriptor blocks defining a complete SGDMA transfer task. The descriptor blocks are linked together using a linked list, and they do not need to be stored consecutively in memory.

Table 53: Descriptor Format Table

Offset	Field	Description
0x0	des_header_h	[4:0]: Upper 5 bits of the data transfer length. [31:5]: Reserved, default value is 0.
0x04	des_header_l	[0]: Completed bit, set to 1 to indicate that an interrupt should be generated after the data transfer is complete. [1]: Stop bit, set to 1 to indicate that the SGDMA engine should stop requesting new descriptors. [2]: EOP bit, end-of-packet flag for AXI4-Stream interface. [8:3]: nxt_adj, represents the remaining number of descriptors in the current block. [31:9]: Lower 23 bits of the data transfer length, combined with des_header_h [4:0] to form a 28-bit data length, in bytes.
0x08	src_addr_h	Upper 32 bits of the source address for data transfer.
0x0C	src_addr_l	Lower 32 bits of the source address for data transfer.
0x10	dst_addr_h	Upper 32 bits of the destination address for data transfer.
0x14	dst_addr_l	Lower 32 bits of the destination address for data transfer.
0x18	des_link_address_h	Upper 32 bits of the next descriptor address.
0x1C	des_link_address_l	Lower 32 bits of the next descriptor address.

The operational flow of the descriptor is as follows:

1. **List Creation:** The host driver creates the descriptor list and stores it in the host memory.
2. **Register Configuration:** The host driver configures the SGDMA registers through the BAR space and initializes the following parameters:
 - Start address of the current descriptor list (HTC: 0x300C and 0x3010; CTH: 0x400C and 0x4010).
 - Number of descriptors in the first descriptor block of the current descriptor list (HTC: 0x3004; CTH: 0x4004).
3. **Transfer Start:** The SGDMA engine fetches descriptors from the starting address of the current descriptor list and uses the `des_link_address` field of the descriptor to obtain the address of the next descriptor.
4. **Descriptor Block Processing:**
 - Descriptors within the same descriptor block have consecutive addresses and the next descriptor address can be obtained by offsetting 32 bytes from the current descriptor address.
 - When the `nxt_adj` value of a descriptor is 0, it indicates that the current descriptor is the last descriptor in the descriptor block. The next descriptor block address is obtained via the `des_link_address` field.
5. **List Termination:** When the `stop` bit of a descriptor is set to 1, it indicates that the current descriptor list has ended. The SGDMA engine will stop the transfer task after completing this descriptor.

AXI4-MM Bypass Master

The AXI4-MM bypass master interface bypasses the SGDMA module, providing the host with direct access to user logic memory. The PCIe BAR4/5 address space is mapped to this interface and any transaction layer packet (TLP) targeting BAR4/5 is forwarded directly to the user logic. Additionally, the TLP address must meet the 8-byte alignment requirement.

AXI4-Stream Data Interface

In addition to the basic AXI4-MM interface, the SGDMA IP also provides a streaming AXI4-Stream user logic interface. This interface uses a non-addressed transfer mechanism, enabling high throughput and low-latency transmission of large data streams. It is particularly suitable for continuous data transfer scenarios, such as video streams, network packet transmission, and sensor data collection.

While the CTH and HTC can be configured with different user interfaces, the two interfaces cannot be used simultaneously.

The SGDMA provides an AXI4-Stream interface for streaming transfer mode, enabling the transfer of SGDMA data between the host and the device. The HTC AXI4-Stream interface is used to transfer HTC SGDMA data to external user logic, while the CTH AXI4-Stream interface handles the transfer of CTH SGDMA data to the host. The user-side interface of the SGDMA channel supports the AXI4-Stream bus.

When data is transferred from the host to the user side, the source address is obtained from the host, but the destination address in the descriptor is not used. Stream-based data can be divided into multiple descriptors for transfer, with the end of each data packet indicated by the EOP bit in the descriptor. When transferring a descriptor containing the EOP bit, the AXI4-Stream bus sets the `tlast` signal high on the last data beat.

When selecting AXI4-Stream port mode, the IP allocates a dedicated AXI4-Stream port on the user logic side for streaming SGDMA transfers. The packet boundary is identified

by the combined EOP flag in the descriptor and the t_{last} signal of the AXI4-Stream interface, ensuring accurate data packet segmentation. Note that channel interleaving is not supported in this mode. All port switching operations must be strictly aligned with the packet boundary, meaning that port switching is only allowed when EOP/ t_{last} is asserted.

The t_{keep} bits must be set to all 1s for every data transfer in a packet, except for the last packet. On the last transfer—when t_{last} is asserted—you may use a t_{keep} value with fewer than all 1s to indicate that the final data cycle is not fully occupying the datapath width. In such cases, the asserted t_{keep} bits must be packed toward the least significant bits, representing contiguous valid data. Note that asserting t_{last} with all zeros in t_{keep} is invalid and may result in unpredictable DMA behavior.

For CTH transfers, when the CTH engine is enabled and detects a valid descriptor, it receives the data stream from the user side through the AXI4-Stream bus. The engine transfers data byte-by-byte according to the address parameters in the descriptor until it encounters the end-of-packet (EOP) flag or completes the data transfer as specified in the current descriptor. Once the transfer is complete, the CTH engine writes the completion status information to the pre-configured write-back address on the host side. This information includes the execution result of the current descriptor and the actual transferred byte length. It should be noted that this write-back information is completely isolated from the polling-mode status registers, and follows an incremental, descriptor-based state feedback mechanism.

The write-back mechanisms of HTC and CTH are fundamentally different. HTC uses polling registers to obtain global status, while CTH uses descriptor chains to provide fine-grained task completion notifications.

APB3 Configuration Interface

The IP provides register configuration through two sets of APB3 interfaces:

- M_APB3
- S_APB3

The M_APB3 interface provides user-side register access. The host initiates read and write requests to this interface through PCIe, which is mapped to the BAR0/1 space.

The S_APB3 interface is dedicated to managing internal SGDMA registers and is restricted to internal operations within the user logic. It does not provide access to PCIe registers and does not initiate requests to the host.

Customizing the PCIe SGDMA

The core has parameters so you can customize its function. You set the parameters in the General and BAR Space Configuration tabs of the core's IP Configuration window.

Table 54: IP Parameter Configuration (General Tab)

Parameter	Options	Description
AXI Data Width	256	AXI data width on the user side
SGDMA User Interface	AXI4-MM, AXI4-Stream	SGDMA user interface option: Default: AXI4-MM
APB3 Slave Interface	Enable, Disable	APB3 slave interface: Default: Enable
APB3 Master Interface	Enable, Disable	APB3 master interface: Default: Enable
SGDMA Bypass Interface	Enable, Disable	BYPASS interface: Default: Enable
Number of User IRQs	1-16	Number of user interrupt requests: Default: 4
MSI-X Vector Count	1-32	Enable vector, maximum number of 32. Typically, Linux uses only one vector for MSI. This option can be disabled with $>(\text{CTH_CH_NUM} + \text{HTC_CH_NUM} + \text{USR_INT_NUM})$. Default: 32
Number of CTH Channels	1-4	Number of SGDMA write channels Default: 1
Number of HTC Channels	1-4	Number of SGDMA Read channels Default: 1

Table 55: BAR Space Configuration Parameters

Parameter	Options	Description
BAR0 Space Base Address	64'h0-64'hFFFFFFFFFFFFFFF	BAR0 space base address, used by the host driver to configure SGDMA registers; corresponds to the AXI inbound BAR0 register configuration. Default: 64'h0
BAR0 Space Size	128 B, 256 B, 512 B, 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, 64 GB, 128 GB, 256 GB	BAR0 space size corresponds to the hard IP BAR0 space configuration. Default: 512 KB
BAR4 Space Base Address	64'h0-64'hFFFFFFFFFFFFFFF	BAR4 space base address, used by the host driver via the SGDMA APB master interface to configure user logic; corresponds to the AXI inbound BAR4 register configuration. Default: 64'ha0000
BAR4 Space Size	128 B, 256 B, 512 B, 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, 64 GB, 128 GB, 256 GB	BAR4 space size corresponds to the hard IP BAR4 space configuration. Default: 4 KB
BAR2 Space Base Address	64'h0-64'hFFFFFFFFFFFFFFF	BAR2 space base address, used as the AXI address space for SGDMA bypass (bypassing the SGDMA engine for traffic); corresponds to the AXI inbound BAR2 register configuration. Default: 64'h90000
BAR2 Space Size	128 B, 256 B, 512 B, 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, 64 GB, 128 GB, 256 GB	BAR2 space size corresponds to the hard IP BAR2 space configuration. Default: 4 KB

Software Driver



Note: Software driver can be download from the Efinix support site. Contact your nearest AE for support.

PCIe SGDMA Example Design



Note: Example design can be downloaded from the Efinix support site. Contact your nearest AE for support.

PCIe SGDMA Testbench



Learn more: Contact your nearest AE for support.

Revision History

Table 56: Revision History

Date	Version	Description
November 2025	1.0	Initial release.