NVMe Target Controller v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG329 (v1.0) June 3, 2020

Table of Contents

Introduction

The Xilinx® NVMe™ Target Controller IP allows for the implementation of a Non-Volatile Memory Express (NVMe) device inside an FPGA. The IP works in tandem with the Xilinx QDMA Subsystem for PCI™ Express and exposes an NVMe 1.3 spec compliant device view to the host. The IP manages the following functions:

1. Exposes and emulates the NVMe controller registers as defined in the NVMe 1.3 specification.
2. Manages the Submission Queue (SQ)/Completion Queue (CQ) doorbells from the host.
3. Arbitrates across available SQs and programs the Queue Direct Memory Access (QDMA) to fetch the required submission queue entries (SQEs).
4. Parses the SQEs and programs the QDMA to fetch the physical region page (PRP) list if applicable.
5. Programs the QDMA for data transfer between host and FPGA (for reads or writes) based on instruction from the application/user logic.
6. Programs the QDMA for completion queue entry (CQE) transfer to the host based on instruction from the application/user logic.

The following sections provide a top-level description of the IP along with its interfaces and programing information. The terms “flash drives”, “NVMe drives” and “SSDs” are used interchangeably in the document and signify an NVMe based flash device used for storage.

***Note*:** The information provided in this document is preliminary and is subject to change.

Features

The NVMe Target Controller IP provides the following features on the host side and application/user logic side interface.

Features on the host side include:

- Configurable number of host side SQ/CQs per controller (maximum of 64)
- Configurable depth of SQ/CQs
- Support for the PRP

- Command parsing for errors
- MSI-x interrupt generation handling

Features on the application side include:

- Offloads the application/user logic from complete QDMA programing
- Admin queues are mapped to software while the I/O queues are mapped to the hardware user logic interface
- Memory-mapped AXI4 interface for software to post “instructions” to NVMe TC
- AXI4-Stream interface for hardware application/module to post “instruction” to NVMe TC

IP Facts

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™[2]
Supported User Interfaces	AXI4-Lite, AXI4-Stream, AXI4
Resources	Performance and Resource Use web page
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Constraints File (XDC)
Simulation Model	Not Provided
Supported S/W Driver[3]	Linux
Tested Design Flows[4]	
Design Entry	Vivado® Design Suite Vivado IPI
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Design Suite
Support	
Release Notes and Known Issues	Master Answer Record: 73241
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Provided by Xilinx at the Xilinx Support web page	

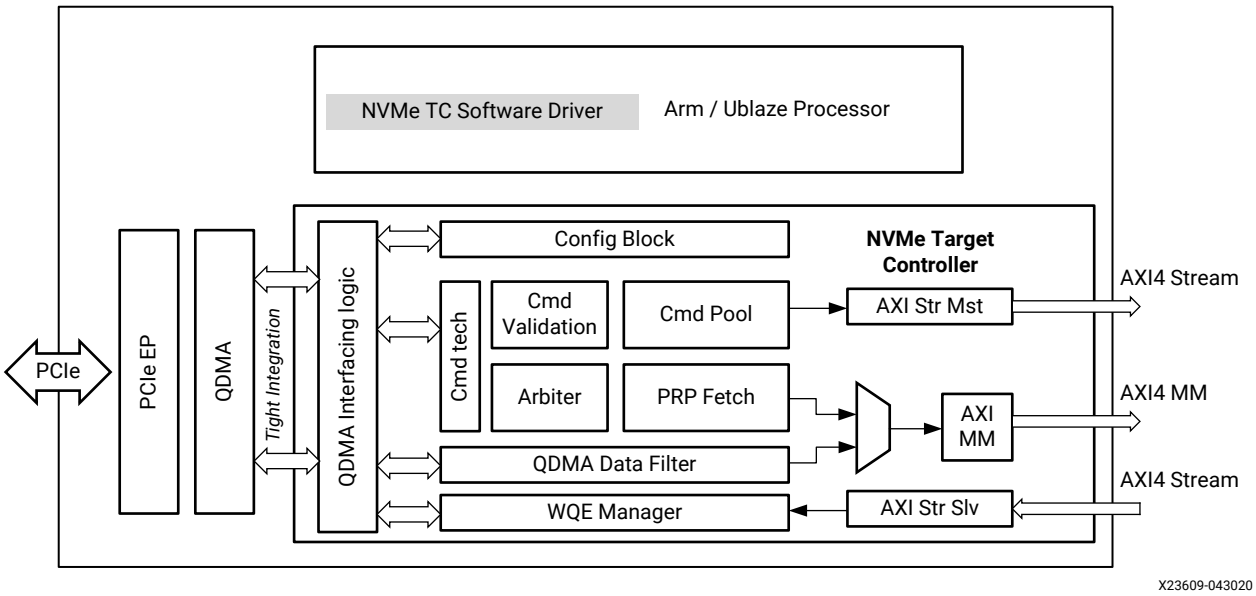
Notes:

1. For a complete list of supported devices, see the Vivado® IP catalog.
2. QDMA only supports UltraScale+ devices.
3. Linux: Linux OS and driver support information is available from the https://www.xilinx.com/member/nvmetc.html page.
4. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The NVMe™ Target Controller core interfaces with QDMA on the host facing side and with the hardware application, processor, and DDR (or any memory region) on the FPGA facing side. The NVMe TC IP maps the admin queues of all PCIe® functions to the software interface and I/O queues of all functions to the hardware interface. The IP manages all control paths and programs the QDMA for SQE fetches (and PRP fetches, if required), CQE writes, and all data transfers. The figure below shows the top-level diagram of the NVMe TC IP and the various interfaces it exposes.

Figure 1: **NVMe Target Controller Block Diagram**



Control and Status Register Module

This module implements the controller registers described in NVM Express 1.3 specification for each VF/PF function. Please refer to section 3.1 of the NVMe 1.3 specification for details on the controller registers.

The QDMA IP allows for one or more PCIe BAR of any function to be mapped to the AXI4-Lite interface of the IP. This interface is mapped to the `host_s_axi_lite` (slave) interface of the NVMe TC IP. The QDMA configuration is done at the time of IP generation. In order for a seamless integration between QDMA and NVMe TC, all VF and PF BAR0 and BAR1 need to be mapped to the AXI4-Lite bridge interface of the QDMA. Function ID, bar id (bar hit), VF group, and VF group offset is available from the QDMA as a part of ARUSER and AWUSER of the AXI4-Lite interface and is used by the NVMe TC to identify the source of the memory access.

The Control and Status Register (CSR) modules also implements the NVMe TC specific configuration registers.

Host Submission Queue Arbitration Module

This module arbitrates among all available Host Submission Queues (HSQs) and computes the number of commands to be fetched in a single request. The arbitration module takes information from the CSR and the command validation module to know if it can initiate further fetches. Each I/O command that requires an additional PRP fetch is allocated a unique ID (UID) by the HSQ arbiter. The total number of free UID represents the total number of I/O commands (that need additional PRP fetch) under processing or outstanding at the NVMe TC IP. In case the maximum number of outstanding commands (as defined by the parameter C_NUM_CMD_INDX) is exhausted, no new commands are fetched until one or more commands are completed by the hardware application.

Command Validation/Decode Module

This module validates the incoming command (SQE) from the arbitration module and does the following checks:

- PRP1/2 offset error check
- Invalid field in command

Once the command is validated validation, it is pushed to the local command buffer pool. If the command requires additional PRP entries, the required information (PRP list pointer) is pushed to PRP fetch manager module for fetching the remaining PRP data pointers. Once the first set of PRPs are fetched, the SQE command is modified and pushed on the `cmd_m_axis` towards the hardware application. Each command is appended with the UID. In case any error is encountered, the same is reported to the hardware/software application. The application may then prepare an error completion (CQE) and post it to the host using the NVMe TC IP. The format of the entry provided to the hardware/software application is shown in the following table.

Table 1: **Validated Command Entry Format**

Bitwidth	Field	Size (Bytes)	Description
[511:0]	SQE	64	Standard NVMe Command
[527:512]	UID	2	NVMe TC level unique ID, used to identify PRP buffer. ***Note***: 16'hFFFF is used for no PRP list case.
[543:528]	SQID	2	Submission Queue ID
[559:544]	FNID	2	Physical/Virtual Function ID
[575:560]	ERR_CODE	2	NVMe Command Validation Error Status

The error code decoding is provided in the following table.

Table 2: **ERR_CODE Decoding**

ERR_CODE	Description
16'h0002	Invalid Field in Command: Asserted when host submits a command that exceeds MDTS transfer size or in case of an invalid Namespace ID in the command.
16'h0006	Internal Error: Error while fetching Submission Queue entry from QDMA/Host
16'h0013	PRP Offset Invalid: The Offset field for a PRP entry is invalid. This may occur when there is a PRP entry with a non-zero offset after the first entry.

PRP Fetch Manager Module

This module manages the fetching of additional PRP entries from the host by programming the descriptor on the QDMA H2C bypass interface. The NVMe TC IP supports PRPs and fetches all PRPs required for complete data transfer specified in the NVMe command. These PRPs are then pushed on the `ddr_m_axi` interface to the PRP memory location as provided by the parameter C_SGL_PRP_BUF_BA. PRP memory location of each UID holds PRPs and 128-bit PRP fetch information.

The NVMe TC fetches the maximum possible PRPs (or 2*(C_NUM_PRPS_PER_INDX – 1) PRPs) in initial fetch and provides pending PRP fetch requirement in 128-bit PRP fetch information , as shown in the following table, to the hardware application layer. Additional PRP request and response is done on `sgl_prp_req_axis` and `sgl_prp_fill_m_axis` interfaces respectively. The application must trigger additional PRP fetch by providing PRP fetch information on the `sgl_prp_req_axis` interface. The NVMe TC fetches additional PRPs and HW application is intimated on sgl_prp_fill_m_axis. NVMe TC implements two buffers in order to store PRPs. Let's call them ping and pong buffers. First set of PRPs are filled in the ping buffer and NVMe command is delivered to the application. On receiving the trigger for the remaining

PRPs, the NVMe TC fetches and stores additional PRPs in the pong buffer. TC switches between the ping and pong buffers alternatively for every PRP fetch request from the application. The application can hide additional PRP fetch latency by triggering PRP fetch before processing the current PRP set. Last PRP information is embedded in the PRP fetch information provided along with each PRP set. The application should stop trigger for additional fetch once last the PRP information is decoded in the PRP fetch information. The PRP fetch manager module detects invalid PRP offset error and sets appropriate error status along with command information to the hardware application.

Table 3: **PRP Header Information**

Bitwidth	Field	Size (Bits)	Description
[31:0]	Data_size_rem	32	Remaining data size to be fetched
[92:32]	Host_addr	61	Host address [63:3] for additional PRP fetch.
[94]	Sgl_or_prp	1	0 – PRPs are used for this command.
[106:95]	prp_index	12	PRP Buffer Index
[107]	ping_or_pong	1	0 – Ping Buffer 1 – Pong Buffer
[117:108]	No_of_desc_rem	9	Number of PRP descriptors remaining in the current PRP list.
[123:118]	Error_Code	6	PRP error code
[127:124]	Valid_entries	4	Number of valid entries in current PRP Buffer Index. Valid vValuesareis 0 to 2*(C_NUM_PRPS_PER_INDX)-1

Notes:

1. In the current release, only the PRP buffer addressing is supported by the design.
2. C_NUM_PRPS_PER_INDX is automatically derived from a Vivado build based on user configured MDTS values.

The response for the next set of PRP fetch is provided on the `sgl_prp_fill_m_axis` interface in the format given in the following table.

Table 4: **PRP Fill Descriptor information**

Bitwidth	Field	Size (Bits)	Description
[11:0]	prp_index	12	PRP Buffer Index
[12]	Ping_or_pong	1	0 – Ping Buffer 1 – Pong Buffer
[15:13]	Reserved	3	Reserved

Work Queue Manager

This module manages the data movement between card to host and host to card and is also responsible for posting completions. This module receives the work instructions/requests also referred to as work queue entries (WQE) from the software application/hardware module and programs the QDMA for the required data transfer. The hardware application pushes these WQEs through the `wqe_s_axis` interface. Internally, the IP maintains two FIFOs for incoming WQEs. All WQEs that transfer data from the host to FPGA (H2C) are pushed to H2C FIFO while all WQEs that transfer data from FPGA to the host (C2H) are pushed to the C2H FIFO. Two FIFO full signals (`h2c_wqe_fifo_full` and `c2h_wqe_fifo_full`) are also provided to the hardware applications. The application is expected to not push WQEs for the respective data transfers if the corresponding fifo_full bits are set. The NVMe TC IP drops any packets targeted to FIFOs that are full and sets the appropriate status register bits in DBG_WQE_MGR register. The WQEs are temporarily stored in these FIFOs until they are processed and pushed to the relevant QDMA queues. The same structure of WQEs is also applicable to the software interface.

The structure of the WQE is given in the following table:

Table 5: **Work Queue Entry Format**

Bitwidth	Field	Size (Bytes)	Comment
[15:0]	WRID	2	Work Request ID. Unique Identifier for every WQE
[31:16]	SQID	2	Submission Queue ID
[47:32]	FNID	2	Function ID=0
[55:48]	OPCODE	1	8'h00 = C2H DMA Descriptor (Push only when c2h_wqe_fifo_full is 0) 8'h02 = NVMe H2C/C2H Completion[1] 8'h03 = NVMe H2C Completion[1] (Optional) 8'h04 = H2C DMA Descriptor (Push Only when h2c_wqe_fifo_full is 0) ***Note***: All other OPCODEs are reserved.
[63:56]	RESERVED	1	Reserved
[127:64]	LADDR/CQE_LSB	8	Local Buffer Address for DMA (or) CQE {DW1, DW0} for NVMe Completion ***Note***: Local Buffer Address must be 4 KB aligned.
[191:128]	HADDR/CQE_MSB	8	Host Buffer Address for DMA (or) CQE {DW3, DW2} for NVMe Completion
[207:192]	LENGTH	2	Data transfer length for DMA's Reserved if (OPCODE = 8'h02 or 8'h03)
[223:208]	UID	2	UID of the command that is getting completed. Only valid if the (OPCODE = 8'h02 or 8'h03)

Table 5: **Work Queue Entry Format** *(cont'd)*

Bitwidth	Field	Size (Bytes)	Comment
[255:224]	Reserved	4	Reserved

Notes:
1. Push only when c2h_wqe_fifo_full is 0.

***Note*:** The phase bit and HSQ head pointer information in the CQE is replaced by the NVMe TC IP.

If the work queue entry is posted to send the NVMe completion (CQE) information to the host, the completion data (CQE) can be inserted in the work request itself. Also, in this case the UID information in the work queue entry is used by the NVMe TC IP to free up the resources related to the corresponding command.

Work Queue Completion

For every work request that is submitted to the NVMe TC IP, a completion is provided by the IP to the corresponding hardware or software application. This can be used by the application to free up resources linked to the particular work request. The format of the work queue completion is given in the following table. The same completion is provided on the software as well as the hardware interface.

Table 6: **Work Request Completion Format**

Bitwidth	Field	Size (Bits)	Description
[15:0]	WRID	16	Work Request ID. Unique Identifier for every WQE
[31:16]	SQID	16	Submission Queue ID
[47:32]	FNID	16	Physical/Virtual Function ID
[59:48]	CREDIT	12	SW WQE Request FIFO available credits (to be ignored by the hardware application)
[63:60]	STATUS	4	Work completion status: 4'h0 – Success 4'h1 - DMA Error at PCIe/QDMA 4'h2 - DMA Error at Local DDR/Memory 4'h3 – Unsupported Opcode Error 4'h4 to 4'hF – Reserved

The credit information in the work request completion is only valid for the software interface.

Software Interface

The NVMe TC IP maps the admin queues of all functions to the software interface. Software can interface with the IP using the `sw_s_axi` interface. When a new admin queue command for any function is received, the admin queue command is written to the software queue defined by the SW_Q_ATTRIBUTE register using the `ddr_m_axi` interface. An interrupt is also raised if the INTR_EN [16] is enabled and the INTR_STS [16] is set. The software can then read the admin queue entries from the queue based on the write pointer information provided in the SW_Q_ATTRIBUTE [111: 96] register. Once those entries are processed, the software can update the read pointer information in the same register to inform the NVMe TC IP about the entries that have been freed.

Based on the processing of each admin command, the software can transfer some data or completion. This can be done by pushing WQEs (work instructions) to the NVMe TC through the `sw_s_axi` interface. The format of the WQEs is the same as defined in Table 5: Work Queue Entry Format. The NVMe TC IP provides a FIFO-like interface to the software to post WQEs and read back completions. The depth of this FIFO is defined by the parameter C_DEPTH_SW_WQE_FIFO. This also forms the initial "credit" to the software to post WQEs. Every time a WQ entry is posted by the software, the credit is decremented by 1. The completions returned by the NVMe TC provide incremental "credits" back to the software.

Table 7: **SW AXI4 (sw_s_axi) Slave Memory Map**

Address Offset	Field	Comment
0x000	WQE entry (32B each)	Any writes to this location is regarded as a WQE push into the internal software WQE FIFO irrespective of the actual location it is written to.
0x100	WQE completion entry (8B each)	Any reads from this location is regarded as a pop from the internal software WQE completion FIFO. Any writes to this location is ignored.

For every work queue entry that is completed by the NVMe TC IP, a corresponding completion is provided to the software. The IP also asserts an interrupt to inform the software of new completions to be read by the software if the INTR_EN [17] is set. The WQ_CMPL_ATTRIBUTE [15:0] register can be read to know the number of Work Queue Request Completions.

Hardware Interface

The NVMe TC IP maps the I/O queues of all functions to the hardware interface. Any hardware application/user logic can interface with the IP using the `cmd_m_axis`, `wqe_s_axis`, `sgl_prp_req_axis`, `sgl_prp_fill_m_axis`, and `wqe_cmpl_m_axis` interfaces. Any new SQE available on any I/O queue is pushed by the NVMe TC IP through the `cmd_m_axis` interface. Also, any work request completions to be provided to the hardware application is provided through `wqe_cmpl_m_axis` interface. The `wqe_s_axis` interface is used by the hardware application to post work requests to the NVMe TC IP. The `sgl_prp_req_axis` interface is used by the hardware application to request for fetching of additional PRP and fetch done information is intimated on `sgl_prp_fill_m_axis` to hardware application.

Related Information
NVMe I/O and Admin Commands Data Flow

Applications

The NVMe Target Controller core enables a broad range of storage solutions such as:

- Storage aggregation and sharing
- Enables FPGA based acceleration platform, like data compression, LZ4
- Enables NVMe based computational storage

Unsupported Features

The following features are not supported in the NVMe Target Controller IP.

- Controller memory buffer (CMB).
- Boot partition.
- Queue depths not in the power of two.
- Non-contiguous SQ/CQs.
- Vendor specific arbitration.
- Weighted round robin (WRR) arbitration.
- Variable metadata format per namespace.
- NVM subsystem reset feature.

- Index/data pair registers and access.
- Mixture of commands with different metadata options.
- Advanced optional features, like power management, reservations, HMB, RPMB, device self-test, telemetry, sanitize, boot partitions, etc.
- Optional admin commands.
- Optional I/O commands.
- Error checks other than specified in this specification are not supported.

Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. To generate a full license, visit the product licensing web page. Evaluation licenses and hardware timeout licenses might be available for this core. Contact your local Xilinx sales representative for information about pricing and availability.

***Note*:** To verify that you need a license, check the License column of the IP Catalog. Included means that a license is included with the Vivado® Design Suite; Purchase means that you have to purchase a license to use the core.

For more information about this core, visit the NVMe Target Controller product web page.

Information about other Xilinx® LogiCORE™ IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

License Checkers

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

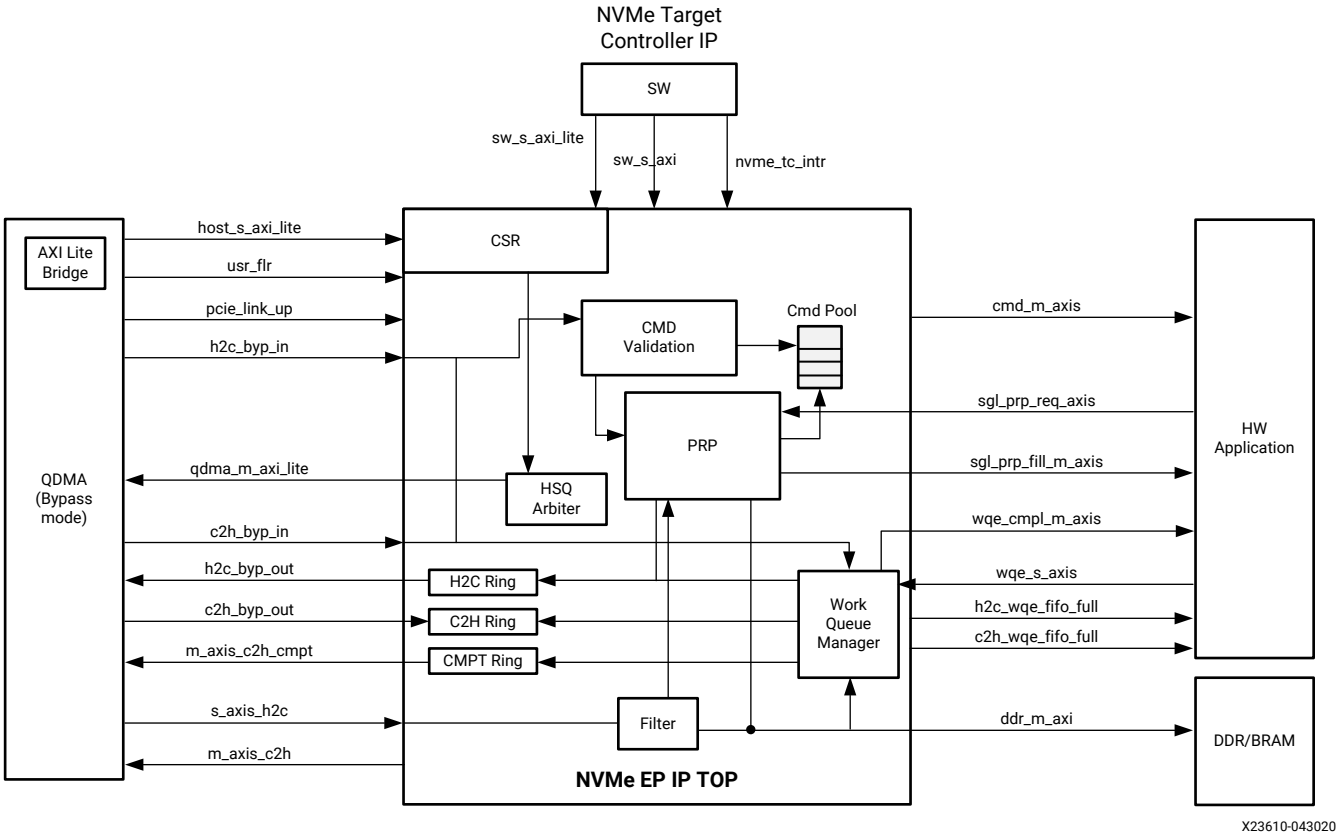
- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT! *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Product Specification

The functional block diagram of the core is shown in the following figure.

Figure 3: **Core Block Diagram**



Standards

This core adheres to the following standard(s):

- NVM_Express_Revision_1.3

Performance and Resource Use

For full details about performance and resource use, visit the Performance and Resource Use web page.

Port Descriptions

NVMe Target Controller Interfaces

The interfaces of NVMe Target Controller IP are given in the following table along with the connectivity expected.

Table 8: **NVMe Target Controller Interfaces**

Interface Name	Interface Type	Interfacing block	Interface Data Width (bits)	Clock Domain	Purpose
c2h_wqe_fifo_full	Active-High	HW	1	qdma_clk	TC application should not generate C2H work queue entries when this full signal is asserted.
h2c_wqe_fifo_full	Active-High	HW	1	qdma_clk	TC application should not generate H2C work queue entries when this full signal is asserted.
nvme_tc_intr	Active-High	SW	1	qdma_clk	NVMe TC IP events notification to software.
pcie_link_up	Active-High	QDMA	1	qdma_clk	PCIe link-up event
usr_flr	Native Interface	QDMA	8	qdma_clk	Function level reset information from QDMA.
sgl_prp_fill_m_axis	AXI-ST(M)	HW	16	qdma_clk	Fill notification of PRP to hardware application.
host_s_axi_lite	AXI4-L (S)	HOST	32	qdma_clk	To access NVMe 1.3 Spec registers via QDMA AXI4-Lite Bridge IF.
qdma_m_axi_lite	AXI4-L (M)	QDMA	32	qdma_clk	To access QDMA registers from NVMe TC IP.
sw_s_axi_lite	AXI4-L (S)	SW	32	lite_clk	NVMe TC IP Control and Status Registers access.
wqe_cmpl_m_axis	AXI-ST (M)	HW	64	qdma_clk	To push Work Request completions.
m_axis_c2h_cmpt	AXI4-ST (M)	QDMA	128	qdma_clk	To Write Completion Queue Entry (CQE) to QDMA CMPT Ring.
sgl_prp_req_axis	AXI4-ST (S)	HW	128	qdma_clk	Request from the hardware application to fetch the next set of PRPs for a command.

Table 8: **NVMe Target Controller Interfaces** *(cont'd)*

Interface Name	Interface Type	Interfacing block	Interface Data Width (bits)	Clock Domain	Purpose
sw_s_axi	AXI4-MM (S)	SW	128	qdma_clk	Software interface to push Admin Queue related WQEs (Work Queue Entries or instruction) to NVMe TC.
c2h_byp_in	Req/Rdy (S)	QDMA	256	qdma_clk	To get C2H (card-to-host) data descriptors completion notification.
c2h_byp_out	Req/Rdy (M)	QDMA	256	qdma_clk	To program C2H (card-to-host) DMA descriptors.
h2c_byp_in	Req/Rdy (S)	QDMA	256	qdma_clk	• Incoming to read SQE entries. • Incoming to get DMA descriptor completion notifications.
h2c_byp_out	Req/Rdy (M)	QDMA	256	qdma_clk	To program descriptor to fetch PRP entries and H2C (host-to-card) DMA descriptors.
wqe_s_axis	AXI4-ST (S)	HW	256	qdma_clk	Hardware application interface to push WQEs to NVMe TC IP.
cmd_m_axis	AXI4-ST (M)	HW	512	qdma_clk	To push SQEs.
ddr_m_axi	AXI4-MM (M)	BRAM/DDR	512	qdma_clk	This interface is used for the following transfers: • To write admin commands. • To write I/O commands PRP. • To read/write H2C/C2H data payload.
m_axis_c2h	AXI4-ST (M)	QDMA	512	qdma_clk	To write C2H data payload to QDMA.
s_axis_h2c	AXI4-ST (S)	QDMA	512	qdma_clk	This interface is used for the following transfers • To receive PRPs from QDMA. • To receive H2C data payload.

Notes:

1. `qdma_clk` is fed from QDMA clock.
2. `lite_clk` is the slower clock.

Interface Ports Associated with QDMA

The following interfaces/ports are associated with QDMA. All ports are in the `core_clk` domain.

AXI4-Stream H2C Port

Table 9: **AXI4-Stream H2C Port Descriptions**

Port Name	I/O	Description
s_axis_h2c_tdata[C_M_AXI_DATA_WIDTH -1:0]	I	Data input for H2C AXI4-Stream
s_axis_h2c_qid[10:0]	I	Queue ID
s_axis_h2c_port_id[2:0]	I	Port ID
s_axis_h2c_err	I	If set, indicates the packet has an error. This error could be from PCIe, or the QDMA might have encountered a double bit error.
s_axis_h2c_mdata[31:0]	I	Metadata
s_axis_h2c_mty[5:0]	I	The number of bytes that are invalid on the last beat of the transaction.
s_axis_h2c_zero_byte	I	When set, it indicates that the current beat is empty beat (zero bytes are being transferred)
s_axis_h2c_dpar [C_M_AXI_DATA_WIDTH/8-1:0]	I	Old parity calculated bit-per-byte over `s_axis_h2c_data`
s_axis_h2c_tvalid	I	Valid
s_axis_h2c_tlast	I	Indicates last cycle of the packet transfer
s_axis_h2c_tready	O	Ready

AXI4-Stream C2H Port

Table 10: **AXI4-Stream C2H Port Descriptions**

Port Name	I/O	Description
m_axis_c2h_tdata[C_M_AXI_DATA_WIDTH-1:0]	O	Data output for C2H AXI4-Stream
m_axis_c2h_mty[5:0]	O	The number of bytes that are invalid on the last beat of the transaction
m_axis_c2h_dpar [C_M_AXI_DATA_WIDTH/8-1:0]	O	Old parity calculated bit-per-byte
m_axis_c2h_tvalid	O	Valid
m_axis_c2h_tlast	O	Indicates last cycle of the packet transfer
m_axis_c2h_tready	I	Ready
m_axis_c2h_ctrl_qid[10:0]	O	Queue ID
m_axis_c2h_ctrl_len[15:0]	O	Length of the packet
m_axis_c2h_ctrl_has_cmpt	O	1'b1: The data packet has a completion 1'b0: The data packet doesn't have a completion
m_axis_c2h_ctrl_marker	O	Marker message used for making sure pipeline is completely flushed
m_axis_c2h_ctrl_port_id	O	Port ID

AXI4-Stream C2H Completion Port

Table 11: **AXI4-Stream C2H Completion Port Descriptions**

Port Name	I/O	Description
m_axis_c2h_cmpt_tdata[511:0]	O	Completion data from the user application. This contains information that is written to the completion ring in the host
m_axis_c2h_cmpt_dpar [C_M_AXI_DATA_WIDTH/8-1:0]	O	Odd parity computed as bit per 32b. `m_axis_c2h_cmpt_dpar[0]` is parity over `m_axis_c2h_cmpt_tdata[31:0]`. `m_axis_c2h_cmpt_dpar[1]` is parity over `m_axis_c2h_cmpt_tdata[63:31]` and so on.
m_axis_c2h_cmpt_tvalid	O	Valid
m_axis_c2h_cmpt_tready	I	Ready
m_axis_c2h_cmpt_ctrl_qid[10:0]	O	Queue ID
m_axis_c2h_cmpt_size[1:0]	O	00: 8B completion 01: 16B completion 10: 32B completion 11: 64B completion
m_axis_c2h_cmpt_ctrl_cmpt_type[1:0]	O	2'b00: NO_PLD_NO_WAIT. The CMPT packet does not have a corresponding payload packet, and it does not need to wait. 2'b01: NO_PLD_BUT_WAIT. The CMPT packet does not have a corresponding payload packet; however, it still needs to wait for the payload packet to be sent before sending the CMPT packet. 2'b10: RSVD 2'b11: HAS_PLD. The CMPT packet has a corresponding payload packe, and it needs to wait for the payload packet to be sent before sending the CMPT packet.
m_axis_c2h_cmpt_ctrl_marker	O	Marker message to ensure that the pipeline is completely flushed.
m_axis_c2h_cmpt_ctrl_user_trig	O	You can trigger the interrupt and the status descriptor write, if they are enabled.
m_axis_c2h_cmpt_ctrl_wait_pld_pkt_id[15:0]	O	The data payload packet ID that the CMPT packet needs to wait for before it can be sent.
m_axis_c2h_cmpt_ctrl_col_idx[2:0]	O	Color index that defines whether you not you want the color bit in the CMPT packet and the bit location of the color bit, if present.
m_axis_c2h_cmpt_ctrl_err_idx[2:0]	O	Error index that defines whether or not you want the error bit in the CMPT packet and the bit location of the error bit, if present.
m_axis_c2h_ctrl_port_id	O	Port ID

H2C-Streaming Bypass Input Port

Table 12: **H2C-Streaming Bypass Input Port Descriptions**

Port Name	I/O	Description
i_h2c_byp_in_dsc[255:0]	I	The H2C descriptor fetched from the host.
i_h2c_byp_in_dsc_sz[1:0]	I	Descriptor size. This field indicates the amount of valid descriptor information. 0:8B 1:16B 2:32B 3:64B
i_h2c_byp_in_vld	I	Valid
o_h2c_byp_in_rdy	O	Ready
i_h2c_byp_in_qid[10:0]	I	Queue ID
i_h2c_byp_in_port_id[2:0]	I	Port ID
i_h2c_byp_in_mrkr_rsp	I	Indicates completion status in response to mrkr requests of h2c_byp_out
i_h2c_byp_in_st_mm	I	Indicates whether this is a streaming data descriptor or memory-mapped descriptor 0: Streaming (only mode supported) 1: memory-mapped
i_h2c_byp_in_error	I	Indicates that an error was encountered in descriptor fetch or execution of a previous descriptor.
i_h2c_byp_in_func[7:0]		PCIe Function ID
i_h2c_byp_in_cidx[15:0]		Consumer Index. The ring index of the descriptor is fetched.

C2H-Streaming Bypass Input Port

Table 13: **C2H-Streaming Bypass Input Port Descriptions**

Port Name	I/O	Description
i_c2h_byp_in_dsc[255:0]	I	The C2H descriptor fetched from the host.
i_c2h_byp_in_dsc_sz[1:0]	I	Descriptor size. This field indicates the amount of valid descriptor information. 0:8B 1:16B 2:32B 3:64B
i_c2h_byp_in_vld	I	Valid
o_c2h_byp_in_rdy	O	Ready
i_c2h_byp_in_qid[10:0]	I	Queue ID
i_c2h_byp_in_port_id[2:0]	I	Port ID

Table 13: **C2H-Streaming Bypass Input Port Descriptions** *(cont'd)*

Port Name	I/O	Description
i_c2h_byp_in_mrkr_rsp	I	Indicates completion status in response to mrkr requests of h2c_byp_out
i_c2h_byp_in_st_mm	I	Indicates whether this is a streaming data descriptor or memory mapped descriptor. 0: Streaming (only mode supported) 1: memory mapped
i_c2h_byp_in_error	I	Indicates that an error was encountered in descriptor fetch or execution of a previous descriptor.
i_c2h_byp_in_func[7:0]		PCIe function ID
i_c2h_byp_in_cidx[15:0]		Consumer Index. The ring index of the descriptor is fetched.

C2H-Streaming Simple Bypass Output Port

Table 14: **C2H-Streaming Simple Bypass Output Port Descriptions**

Port Name	I/O	Description
o_c2h_byp_out_sim_addr[63:0]	O	The C2H descriptor fetched from the host. 64-bit address where the QDMA will DMA the data to.
o_c2h_byp_out_sim_vld	O	Valid
i_c2h_byp_out_sim_rdy	I	Ready
o_c2h_byp_out_sim_qid[10:0]	O	Queue ID
o_c2h_byp_out_sim_port_id[2:0]	O	Port ID
o_c2h_byp_out_sim_error	O	Indicates that an error was encountered in the descriptor fetch or execution of a previous descriptor.
o_c2h_byp_out_sim_func[7:0]	O	PCIe function ID
o_c2h_byp_out_sim_at[1:0]	O	Address type. 2'b00: The address in the request is untranslated. Only address type supported by the IP.

H2C-Streaming Simple Bypass Output Port

Table 15: **H2C-Streaming Simple Bypass Output Port Descriptions**

Port Name	I/O	Description
o_h2c_byp_out_addr[63:0]	O	64-bit starting address of DMA transfer
o_h2c_byp_out_vld	O	Valid
i_h2c_byp_out_rdy	I	Ready
o_h2c_byp_out_qid[10:0]	O	Queue ID
o_h2c_byp_out_port_id[2:0]	O	Port ID
o_h2c_byp_out_error	O	Indicates that an error was encountered in descriptor fetch or execution of a previous descriptor.

Table 15: **H2C-Streaming Simple Bypass Output Port Descriptions** *(cont'd)*

Port Name	I/O	Description
o_h2c_byp_out_func[7:0]	O	PCIe function ID
o_h2c_byp_out_at[1:0]	O	Address Type 'b00: The address in the request is untranslated. Only address type supported by the IP.
o_h2c_byp_out_len[15:0]	O	The number of bytes to transfer.
o_h2c_byp_out_eop	O	Indicates end of packet. Set for the last descriptor. Reset for the rest of the descriptors
o_h2c_byp_out_sop	O	Indicates start of packet. Set for the first descriptor. Reset for the rest of the descriptors.
o_h2c_byp_out_sdi	O	H2C Bypass Out (which is QDMA H2C Bypass In) Status Descriptor/Interrupt. If set, the user application signals the QDMA to send the status descriptor to the host and to generate an interrupt to host when the QDMA has fetched the last byte of the data associated with this descriptor. The QDMA honors the request to generate an interrupt only if interrupts have been enabled in the H2C SW context for this QID and armed by the driver. This can only be set for an EOP descriptor.
o_h2c_byp_out_mrkr_req	O	H2C Bypass In Marker Request. When set, the descriptor passes through the H2C Engine pipeline. Once complete, it produces a marker response on the H2C Streaming Bypass-Out interface. This can only be set for an EOP descriptor.
o_h2c_byp_out_no_dma	O	H2C Bypass In no DMA (Constant 1'b0)
o_h2c_byp_out_cidx[15:0]	O	The CIDX used for the status descriptor update and/or interrupt (aggregation mode).

QDMA Master AXI Lite Port

Table 16: **QDMA Master AXI4-Lite Port Descriptions**

Port Name	Description
qdma_m_axi_lite_*	AXI4-Lite master ports

QDMA/Host Slave AXI Lite Port

Table 17: **QDMA/Host Slave AXI Lite Port Descriptions**

Port Name	Description
host_s_axi_lite_*	AXI4-Lite slave ports

QDMA FLR Port

***Note*:** FLR is not supported in this release.

Table 18: **QDMA FLR Port Descriptions**

Port Name	I/O	Description
i_flr_set	I	Set Asserted for one cycle indicating that the FLR status of the function indicated on `i_flr_fid[[7:0]` is active.
i_flr_clr	I	Clear Asserted for one cycle indicating that the FLR status of the function indicated on `i_flr_fid[7:0]` is completed.
i_flr_fid[7:0]	I	Function: The function number of the FLR status change.
o_flr_done_fid[7:0]	O	Done Function: The function for which FLR has been completed by user logic.
o_flr_done_vld	O	Done Valid Assert for one cycle to signal that FLR for the function on `o_flr_done_fid[7:0]` has been completed.

Interface Ports Associated with Application

Work Queue Entry (WQE) AXI4-Stream Interface Port

Table 19: **Work Queue Entry (WQE) AXI4-Stream Interface Port Descriptions**

Port Name	Description
hwip_wqe_s_axis_*	AXI4-Stream slave ports
hwip_wqe_s_axis_tuser	User-defined port

Command AXI4-Stream Interface Port

Table 20: **Command AXI4-Stream Interface Port Descriptions**

Port Name	Description
cmd_m_axis_*	AXI4-Stream master ports
cmd_m_axis_tuser[67:0]	User-defined port

Work Queue Entry (WQE) Completion AXI4-Stream Interface Port

Table 21: **Work Queue Entry (WQE) Completion AXI4-Stream Interface Port Descriptions**

Port Name	Description
hwip_wqe_cmpl_*	AXI4-Stream master ports
hwip_wqe_cmpl_user	User-defined port

PRG Fetch Request AXI4-Stream Interface Port

Table 22: **PRG Fetch Request AXI4-Stream Interface Port Descriptions**

Port Name	Description
hwapp_sgl_prp_req_*	AXI4-Stream slave ports
hwapp_sgl_prp_req_tuser	User-defined port

PRG Fetch Request Response AXI4-Stream Interface Port

Table 23: **PRG Fetch Request Response AXI4-Stream Interface Port Descriptions**

Port Name	Description
hwapp_sgl_prp_req_respl_*	AXI4-Stream Master Ports
hwapp_sgl_prp_req_resp_user	User defined port

Global Ports

Table 24: **NVMe Global Port Descriptions**

Port Name	I/O	Description
core_clk	I	Core clock
core_rstn	I	Active-Low reset for core_clk
nvme_tc_intr	O	Interrupt
sw_s_axi_lite_aclk	I	AXI4-Lite clock
sw_s_axi_lite_aresetn	I	Active-Low reset for sw_s_axi_lite_aclk
pcie_link_up	I	Active-High identifies that the PCI Express core is linked up with a host device
pcie_phy_ready	I	Phy ready out status
soft_reset_n	O	Soft reset (active-Low). Use this port to assert reset and reset the DMA logic in QDMA. This only resets the DMA logic.

Software Interface Ports

The following ports/interfaces are associated with the processor for the driver/firmware. All ports are in the `sw_s_axi_lite_aclk` clock domain.

AXI4-Lite Slave Interface Ports

Table 25: **AXI4-Lite Slave Interface Ports Description**

Port Name	Description
sw_s_axi_lite_*	AXI4-Lite slave interface ports

AXI4 Slave Interface Ports

Table 26: **AXI4 Slave Interface Ports Description**

Port Name	Description
sw_s_axi_*	AXI4 slave interface ports

DDR AXI4 Master Interface Ports

Table 27: **DDR AXI4 Master Interface Ports**

Port Name	Description
ddr_m_axi_*	AXI4 master interface ports

Register Space

All NVMe TC IP registers are accessible via AXI4-Lite interface.

Any bits not specified in the following register tables are considered reserved and return 0 upon being read. The power on reset values of control registers are 0, unless it is specified in the definition.

Only address offsets are listed in the following table; the base address is configured by the AXI interconnect at system level.

Depending on the number of functions supported (as defined by the parameter C_MAX_FUNC), the register space of the NVMe TC IP would vary. However, the NVMe TC IP control and status registers are common across the functions and do not change based on parameter.

Note: The current release supports only one function, C_MAX_FUNC =0.

AXI4-Lite Slave (sw_s_axi_lite) Memory Map

The table below shows the memory map of the AXI4-Lite slave interface based on number of PCIe® functions supported.

Table 28: **AXI4-Lite slave (sw_s_axi_lite) memory map**

Start Address Offset	Size (KB)	Description	Comments
Function 0 Registers Space			
18'h0_0000	8	NVMe TC IP Control and Status Registers	Internal to TC IP

Table 28: **AXI4-Lite slave (sw_s_axi_lite) memory map** *(cont'd)*

Start Address Offset	Size (KB)	Description	Comments
18'h0_2000	8	Function 0 NVMe Registers (excluding doorbells)	Exposed to the host using Function0.BAR0 (excluding doorbells)
18'h0_4000	8	Function 0 Bridge Registers	QDMA Bridge Registers
18'h0_6000	40	Reserved	Reserved
18'h1_0000	32	Function0 HSQ attributes	Internal to TC IP
18'h1_8000	32	Function0 HCQ attributes	Internal to TC IP
18'h2_0000	128	Function0 QDMA Registers	QDMA Specific Registers
Function 1 Registers Space			
18'h0_0000 + 256KB	8	Reserved	Reserved
18'h0_2000 + 256KB	8	Function 1 NVMe Registers (excluding doorbells)	Exposed to the host using Function1.BAR0 (excluding doorbells)
18'h0_4000 + 256KB	8	Function 1 Bridge Registers	QDMA Bridge Registers
18'h0_6000 + 256KB	40	Reserved	Reserved
18'h1_0000 + 256KB	32	Function1 HSQ attributes	Internal to TC IP
18'h1_8000 + 256KB	32	Function1 HCQ attributes	Internal to TC IP
18'h2_0000 + 258KB	128	Function1 QDMA Registers	QDMA Specific Registers
Function N Registers Space			
18'h0_0000 + (256KB * N)	8	Reserved	Reserved
18'h0_2000 + (256KB * N)	8	Function N NVMe Registers (excluding doorbells)	Exposed to the host using FunctionN.BAR0 (excluding doorbells)
18'h0_4000 + (256KB * N)	8	Function N Bridge Registers	QDMA Bridge Registers
18'h0_6000 + (256KB * N)	40	Reserved	Reserved
18'h1_0000 + (256KB * N)	32	Function N HSQ attributes	Internal to TC IP
18'h1_8000 + (256KB * N)	32	Function N HCQ attributes	Internal to TC IP
18'h2_0000 + (256KB * N)	128	Function N QDMA Registers	QDMA Specific Registers

Notes:

1. N = 0. Only one function is supported in this release.

Register Map

Table 29: **Register Map**

Address Offset	Register Name	Access	Details
0x0000	NVMe TC Interrupt status register (INTR_STS)	RO	[0] Controller enabled. This bit is set when the CC.EN bit transitions from 0 to 1. Clear CTRLR_EN_STS to clear this bit.
		RO	[1] Controller reset. This bit is set when the CC.EN bit transitions from 1 to 0. Clear CTRLR_RST_STS to clear this bit.
		RO	[2] Controller normal shutdown. This bit is set when the CC.SHN field is set to 01. Clear CTRLR_NSHN_STS to clear this bit.
		RO	[3] Controller abrupt shutdown. This bit is set when the CC.SHN field is set to 10. Clear CTRLR_ASHN_STS to clear this bit.
		RW1C	[4] QDMA WRB/CMPT Ring ID is invalidated. Check the CMPT_Q_INVLD register for more details.
		RW1C	[5] Reserved
		RW1C	[6] QDMA C2H/H2C Ring ID is invalidated. Check the QDMA_Q_INVLD register for more details
		RW1C	[7] This bit is set when the host software attempts to write an invalid doorbell value to any SQ. Check the LOG_HOST_QUEUE register to know the Host SQ ID and Function ID.
		RW1C	[8] This bit is set when the host software attempts to write an invalid doorbell value to any CQ. Check the LOG_HOST_QUEUE register to know the Host CQ ID and Function ID.
		RW1C	[9] This bit is set when the host software attempts to write the doorbell of an SQ which was not created. Check the LOG_HOST_QUEUE register to know the Host SQ ID and Function ID.
		RW1C	[10] This bit is set when the host software attempts to write the doorbell of a CQ which was not created. Check the LOG_HOST_QUEUE register to know the Host CQ ID and Function ID.
		RW1C	[11] This bit is set when fused command first part is received from the host. Check the LOG_HOST_QUEUE register to know the Host SQ ID and Function ID.
		RO	[12] Function Level Reset (PCI reset). Clear CTRLR_FLR_STS to clear this bit.
		RW1C	[13] PCIe Reset/PCIe Link Down Event.
		RW1CO	[15:14] PCIe Link Up EventReserved.
		RW	[15] TC Fatal errors event. Check the TC_ERR_STS register for more information.
		RW1C	[16] New Admin command available in the software queue. Check SW_Q_ATTRIBUTE [95:80] to know the number of available admin commands.
		RW1C	[17] New work request completion available for the software to read. Check WQ_CMPL_ATTRIBUTE [15:0] to know the number of work queue request completions.
		RW1C	[18] Admin Command Write Failed Error. This error occurs when the AXI Bresp error is seen by TC when pushing Admin Command on AXI Interface. Check ERR_ADMIN_CMD register for details.
		RW1C	[19] Host SQ safe deletion interrupt. Check HOST_SQ_DELETE register for details.
		RW1C	[20] PCIe Phy Ready event.

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x0004	QDMA Reset Control Status register (QDMA_CS)	RW	[0] – qdma_usr_reset_n • Reset the QDMA logic through the soft_reset_n port. This port needs to be held in reset for a minimum of 100 clock cycles (axi_aclk cycles) • 0: reset • 1: clear reset
		RO	[8] - pcie_link_up
		RO	[9] – pcie_phy_ready
		RW1C	[16] – axi_resetn, core reset interrupt status register, non-maskable
0x0008	NVMe TC Interrupt enable register (INTR_EN)	RW	Bitwise interrupt enable bit for INTR_STS
0x000C	NVMe TC Configuration (TC_CFG)	RW	• [15:0] Maximum Host I/O SQ/CQ Depth. Applicable for all functions. Default value is C_CAP_MAX_HOST_Q_DEPTH • [18:16] Arbitration Burst. Applicable for all functions. Default value is C_ARB_BURST • [23:19] Reserved. • [31:24] Timeout. Applicable for all Functions. Default value is C_CAP_TIMEOUT
0x0010	NVMe TC IP Control Register (TC_CTRL)	RW	• [0] Setting this bit halts the NVMe TC • [1] Setting this bit to 1 stops fetching of the command from all host SQs, including ASQ • [2] Setting this bit to 1 stops the Work Queue Arbitration between software and hardware and processing of WQEs • [3] Setting this bit to 1 initializes the UID index FIFO. • [723:3] Reserved • [15:8] Maximum outstanding command fetch requests at QDMA. Value zero means unlimited • [16] NVMe registers write access enable, except AQA, ASQ, ACQ registers, across all functions. • [17] NVMe registers write access enable, including AQA, ASQ, ACQ registers, across all functions. • [23:18] Reserved
0x0014	Function Level Interrupt Clear (FUNC_INTR_CLR)	RW	• [7:0] Function ID • [15:8] Reserved • [19:16] Interrupt Clear Opcode ◦ 4’h0 - FLR Interrupt clear • [31:20] Reserved
0x0018	Host SQ/CQ ID logging (LOG_HOST_QUEUE)	RO	• [15:0] Host Submission Queue ID is logged here when INTR_STS [7] or INTR_STS [9] • [31:16] Host Submission Controller ID is logged here when INTR_STS [7] or INTR_STS [9] bits are set. • [47:32] Host Completion Queue ID is logged here when INTR_STS [8] or INTR_STS [10] • [63:48] Host Completion Controller ID is logged here when INTR_STS [8] or INTR_STS [10] bits are set.

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x0020	NVMe TC IP Status Register (TC_STS)	RO	• [0] NVMe TC IP functionality halted. This bit is set in response to TC_CTRL [0] bit set • [1] NVMe TC IP stopped HSQ arbitration and fetching. This bit is set in response to TC_CTRL [1] bit set • [2] NVMe TC IP stopped WQE arbitration and processing. This bit is set in response to TC_CTRL [2] bit set. • [3] UID index FIFO Initialization is done. This bit is set in response to TC_CTRL [3] bit set. • [31:3] Reserved
0x0024	Work Queue Completion Attributes (WQ_CMPL_ATTRIBUTE)	RO	• [15:0] Number of available completions • [31:16] Reserved
0x0028	Legacy Interrupt Enable (LEGACY_INTR_EN)	RW	[0] Enable Legacy Interrupt
0x002C	Reserved	RO	Reserved
0x0030	Software Queue Attribute (SW_Q_ATTRIBUTE)	RW	[63:0] Software Queue Base Address
		RW	[79:64] Software Queue Size. Zero based value
		RO	[95:80] Number of available admin commands
		RO	[111: 96] Software Admin Queue Write/tail Pointer
		RW	[127: 112] Softwaare Admin Queue Read/head pointer
0x0040	QDMA_Q_INVLD	RO	• [10:0] QDMA QID which is deleted/aborted • [13:11] Port ID • [14] Error • [15] Queue Invalidated • [16] Queue Enabled • [17] Bypass Mode • [18] 0 – H2C, 1 – C2H • [31:21] Credits
0x0044	CMPT_Q_INVLD	RO	• [10:0] axis_c2h_status_qid • [16] axis_c2h_status_last • [18] axis_c2h_status_cmp
0x0048	ERR_ADMIN_CMD	RO	• [15:0] CMDID of the errored Admin Command • [23:16] Function ID
0x004C	HOST_SQ_DELETE	RO	• [15:0] HSQ ID to be deleted • [23:16] Function ID
0x0050	DATA_BUF_BA_0	RW	Local Data Buffer LSB 32-bit address
0x0054	DATA_BUF_BA_1	RW	Local Data Buffer MSB 32-bit address
0x0058	PRP_BUF_BA_0	RW	PRP Buffer LSB 32-bit Address
0x005C	PRP_BUF_BA_1	RW	PRP Buffer LSB 32-bit Address

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x0060	Debug Control (DBG_CTRL)	RW	• [1] dbg_cmd_err_inj_en • [2] dbg_cqe_sts_zero_en • [8] dbg_sw_wqcmpl_fifo_overwrite • [9] dbg_cmd_val_check_dis • [10] dbg_parity_type • [11] dbg_cmd_pool_rd_en • [19:16] dbg_reg_indx • [31:20] dbg_wqe_fifo_addr
0x0064	NVMe TC Fatal Errors (TC_ERR_STS)	RO	• [0] c2h_wqe_fifo_overrun • [1] h2c_wqe_fifo_overrun • [2] hw_unexpected_sqid • [3] sw_unexpected_sqid • [4] h2c_wqcmpl_fifo_overrun • [5] h2c_zero_len_dma • [7:6] Reserved • [8] c2h_axi_req_fifo_overrun • [9] c2h_axi_rsp_fifo_overrun • [10] c2h_zero_len_dma • [11] Reserved • [12] Unexpected CMPT • [13] Unsupported WQE opcode • [15:14] Reserved • [16] SW Q overrun • [17] SW Credits Underrun • [18] SW wq cmpl read on empty by SW • [19] h2c stream context fifo underrun • [20] axi_wr_error • [21] axi_rd_error • [22] axi wr req fifo overrun • [23] axi rd req fifo overrun • [24] axi wr req fifo underrun • [25] axi rd req fifo underrun • [27:26] Reserved • [28] sw_q_hdbl_invld • [29] sw_q_hdbl_out_of_range • [63:30] Reserved
0x0070	Debug Window Counter (DBG_WIN_CNT)	RW	• [0] Window Counter Enable • [1] Clear all debug counters • [31:4] Window Counter Time
0x0074	Module Reset Enable (MRE)	RW1C	[0] NVMe TC Module Reset Enable. This bit is auto set on PCIe link down event

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x0078	Module Reset Done (MRD)	RO	[0] NVMe TC module reset done
0x0100	Each Controller Enable status register (CTRLR_EN_STS)	RW1C	[255:0] Each bit position when set represents that the corresponding controller is enabled.
0x0120	Each Controller Reset status register (CTRLR_RST_STS)	RW1C	[255:0] Each bit position when set represents that the corresponding controller is in reset.
0x0140	Each Controller Normal Shutdown status register (CTRLR_NSHN_STS)	RW1C	[255:0] Each bit position when set represents that the corresponding controller is in normal shutdown
0x0160	Each Controller Abrupt Shutdown status register (CTRLR_ASHN_STS)	RW1C	[255:0] Each bit position when set represents that the corresponding controller is in abrupt shutdown
0x01A0	Each Controller FLR Status (CTRLR_FLR_STS)	RO	[255:0] Each bit position when set represents that the corresponding controller received FLR. Program FUNC_INTR_C2R to clear this bit
0x01D0	TC_CFG_QUEUE_NUM	RW	• [10:0] Host SQs per functions, including ASQ. Default/Maximum value is C_MAX_HSQ_PER_FUNC • [26:16] Host CQs per functions, ncluding ACQ. Default/ maximum value is C_MAX_HSQ_PER_FUNC
0x01D4	TC_CFG_MDTS	RW	[7:0] Maximum data transfer size (MDTS).
0x01D8	TC_CFG_SW_WQE_CR	RW	[7:0] Software credits for work queue entries. Default/Maximum value is 32.
0x01DC	TC_CFG_NUM_FUNC	RW	[7:0] Number of functions/controllers. Default/Maximum value is C_MAX_FUNC.
0x01E0	Reserved	RO	Reserved
0x01E4	TC_CFG_SGL_SUPPORT	RW	[0] SGL Support ***Note:*** SGL is not supported in this release.
0x01E8	TC_CFG_NSZE_LO	RW	[31:0] Lower 32 bits of Namespace size. This is used in identify response of TC standalone scenario.
0x01EC	TC_CFG_NSZE_UP	RW	[31:0]: Upper 32 bits of Namespace size. This is used in identify response of TC standalone scenario.
0x0200	DBG_SGL_PRP	RO	[127:0] PRP debug and status information
0x0210	DBG_CMD_FETCH	RO	• [3:0] cmd_fetch_cs • [5:4] hsq_arb_cs • [6] sw_q_full_early • [7] h2c_byp_in_vld • [11:8] cmd_val_cs • [12] Internal Arbitration FIFO full • [13] Internal Arbitration FIFO empty • [14] No free UID indexes

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x0214	DBG_CMD_VAL	RO	Reserved
0x0218	DBG_PCIE_LINK_STS	RO	• [0] pcie_link_up • [1] pcie_phy_ready
0x021C	DBG_CMD_CNT	RO	• [15:0] Number of admin commands pushed to the software queue. • [31:16] Number of I/O commands pushed to the hardware IP module.
0x0220	DBG_WQE_FIFO	RO	• [0] sgl_prp_wqe_fifo_empty • [1] sgl_prp_wqe_fifo_afull • [2] sgl_prp_wqe_fifo_full • [3] Reserved • [4] sw_c2h_wqe_fifo_empty • [5] sw_c2h_wqe_fifo_afull • [6] sw_c2h_wqe_fifo_full • [7] Reserved • [8] hw_c2h_wqe_fifo_empty • [9] hw_c2h_wqe_fifo_afull • [10] hw_c2h_wqe_fifo_full • [11] Reserved • [12] hw_h2c_wqe_fifo_empty • [13] hw_h2c_wqe_fifo_afull • [14] hw_h2c_wqe_fifo_full • [15] Reserved • [16] sw_cmpt_wqe_fifo_empty • [17] sw_cmpt_wqe_fifo_afull • [18] sw_cmpt_wqe_fifo_full • [19] Reserved • [20] sw_cmpt_wqe_fifo_empty • [21] sw_cmpt_wqe_fifo_afull • [22] sw_cmpt_wqe_fifo_full • [23] Reserved • [24] hw_cmpt_wqe_fifo_empty • [25] hw_cmpt_wqe_fifo_afull • [26] hw_cmpt_wqe_fifo_full • [31:27] Reserved
0x0224	DBG_WQE_H2C	RO	• [3:0] wqe_mgr_cs • [4] h2c_data_wr_req_fifo_full • [5] h2c_cmpl_fifo_full • [6] h2c_data_wr_req_fifo_empty • [7] h2c_cmpl_fifo_empty • [15] h2c_data_buf_not_avail

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x0228	DBG_WQE_C2H	RO	• [3:0] data_rd_cs • [7:4] post_wqe_cs • [16] axi_rsp_empty • [17] axi_rsp_afull • [18] axi_rsp_full • [19] post_wqe_empty • [20] post_wqe_afull • [21] post_wqe_full
0x022C	DBG_WQE_CMPT	RO	[3:0] cmpt_wqe_cs
0x0230	DBG_CMPT_IN_CNT	RO	• [15:0] Number of software command completions received by the TC. • [31:16] Number of I/O command completions received by the TC.
0x0234	DBG_CMPT_OUT_CNT	RO	• [15:0] Number of software command completions pushed to the QDMA. • [31:16] Number of I/O command completions pushed to the QDMA.
0x0238	DBG_CMPT_CMPL_CNT	RO	• [15:0] Number of software command completion responses to software. • [31:16] Number of I/O command completion responses to the hardware IP.
0x023C	DBG_SW_DMA_CNT	RO	• [15:0] Number of software C2H DMA work request counts. • [31:16] Number of PRP DMA work request counts.
0x0240	DBG_HW_DMA_CNT	RO	• [15:0] Number of hardware IP C2H DMA work request count. • [31:16] Number of hardware IP H2C DMA work request count.
0x0244	DBG_DMA_POP_CNT	RO	• [15:0] Number of C2H DMA work request pops. • [31:16] Number of H2C DMA work request pops.
0x0248	DBG_DMA_PUSH_CNT	RO	• [15:0] Number of C2H DMA pushed to QDMA. • [31:16] Number of H2C DMA pushed to QDMA.
0x024C	DBG_DMA_RSP_CNT	RO	• [15:0] Number of C2H DMA completed. • [31:16] Number of H2C DMA completed.
0x0260 – 0x029C	DBG_TC_TOP	RO	[511:0] NVMe TC IP debug information.

Table 29: **Register Map** *(cont'd)*

Address Offset	Register Name	Access	Details
0x10000 + (0x40000*n) + (0x10*m)[1, 2]	Host SQ Attributes (HSQ_ATTRIBUTE)	RW	• [10:0] Host Submission Queue ID to QDMA QP ID Mapping • [15:112] Reserved • [23:16] Reserved • [24] Host Submission Queue Enable/Disable • [25] Host Submission Queue Arbitration disable • [26] Reserved • [27] SQ deletion pending bit • [31:28] Reserved • [47:32] Outstanding Commands for this SQ • [63:48] Last PIDX value updated to QDMA QID • [79:64] Host Submission Queue Size[3], Zero based value • [95:80] Host CQ ID mapped to this SQ
		RO	• [111:96] Host Submission Queue Tail pointer • [127:112] Host Submission Queue Head pointer
0x18000 + (0x40000*n) +(0x10*m)[1, 2]	Host CQ Attributes (HCQ_ATTRIBUTE)	RW	• [10:0] Host Completion Queue ID to QDMA QP ID Mapping • [15:112] Reserved • [23:16] Reserved • [24] Host Completion Queue Enable/Disable • [25] Host Completion Queue Interrupt Enable • [63:26] Reserved • [79: 64] Host Completion queue size[3], Zero based value. • [80] Host Completion Queue Phase bit.
		RO	• [111: 96] Host Completion Queue Tail Pointer • [127: 112] Host Completion Queue Head Pointer

Notes:

1. n: 0. Only one function/controller is supported in this release.
2. m: (0-64) SQ/CQs per a given function.
3. Size for admin SQ/CQ is written as part of the AQA register write by the host.

Base Specification Registers

The following table describes the controller registers as defined by the NVMe specification and implemented by the NVMe TC IP.

Table 30: **Base Specification Registers (TC SW View)**

Address Offset	Register Name	Access	Details
0x2000 +(0x40000*n)[1]	Controller capabilities (CAP)	RW	• [15:0] Maximum queue entries supported. • [16] Continuous queues required. • [18:17] Arbitration mechanism supported • [31:24] Timeout • [35:32] Doorbell stride • [36] NVM subsystem reset supported • [44:37] Command sets supported. • [45] Boot partition support • [51:48] Memory page size minimum • [55:52] Memory page size maximum
0x2000 +(0x40000*n) + 0x8[1]	Version (VS)	RO	• [7:0] Tertiary version number • [15:8] Minor version number. • [31:16] Major version number.
0x2000 +(0x40000*n) + 0xC[1]	Interrupt mask set (INTMS)	RO	[31:0] Interrupt vector mask set
0x2000 +(0x40000*n) + 0x10[1]	Interrupt mask clear (INTMC)	RO	[31:0] Interrupt vector mask clear
0x2000 +(0x40000*n) + 0x14[1]	Controller configuration (CC)	RO	• [0] Enable • [6:4] I/O command set selected • [10:7] Memory page size • [13:11] Arbitration mechanism selected • [15:14] Shutdown notification • [19:16] I/O submission queue entry size • [23:20] I/O completion queue entry size
0x2000 +(0x40000*n) + 0x1C[1]	Controller status (CSTS)	RW	• [0] Ready • [1] Controller fatal status • [3:2] Shutdown status • [4] NVM Subsystem Reset Occurred • [5] Processing paused
0x2000 +(0x40000*n) + 0x24[1]	Admin queue attributes (AQA)	RO	• [11:0] Admin submission queue size • [27:16] Admin completion queue size
0x2000 +(0x40000*n) + 0x28[1]	Admin submission queue base address (ASQ)	RO	[63:12] Admin submission queue base
0x2000 +(0x40000*n) + 0x30[1]	Admin completion queue base address (ACQ)	RO	[63:12] Admin completion queue base
0x10000 + (0x40000*n) + (0x10*m) + 0xC[1, 2]	Submission Queue tail doorbell (STDBL)	RO	[15:0] Submission Queue Tail
0x18000 + (0x40000*n) +(0x10*m) +0xC[1, 2]	Completion Queue head doorbell (CQHDBL)	RO	[15:0] Completion Queue Head

Notes:

1. n:0. Only one function/controller is supported in this release.
2. m: (0-64) SQ/CQs per a given function.

XILINX

Chapter 4

Designing with the Core

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

General Design Guidelines

Use the Example Design

Each instance of the NVMe Target Controller core created by the Vivado design tool is delivered with an example design that can be implemented in a device and then simulated. This design can be used as a starting point for your own design or can be used to sanity-check your application in the event of difficulty. See the Example Design content for information about using and customizing the example designs for the core.

Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Recognize Timing Critical Signals

The constraints provided with the example design identify the critical signals and timing constraints that should be applied.

Make Only Allowed Modifications

You should not modify the core. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the core can only be made by selecting the options in the customization IP dialog box when the core is generated.

Clocking

Table 31: **Clocks**

Clock	Description
core_clk	IP primary clock that is usually connected to qdma/axi_clk.
sw_s_axi_lite_aclk	AXI4-Lite clock. All the register accesses work on this clock.

Resets

The NVMe TC IP requires two active-Low resets that are synchronized to `core_clk` and `sw_s_axi_lite_aclk` clock domains respectively.

Design Flow Steps

This section describes customizing and generating the core, constraining the core, and the simulation, synthesis, and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

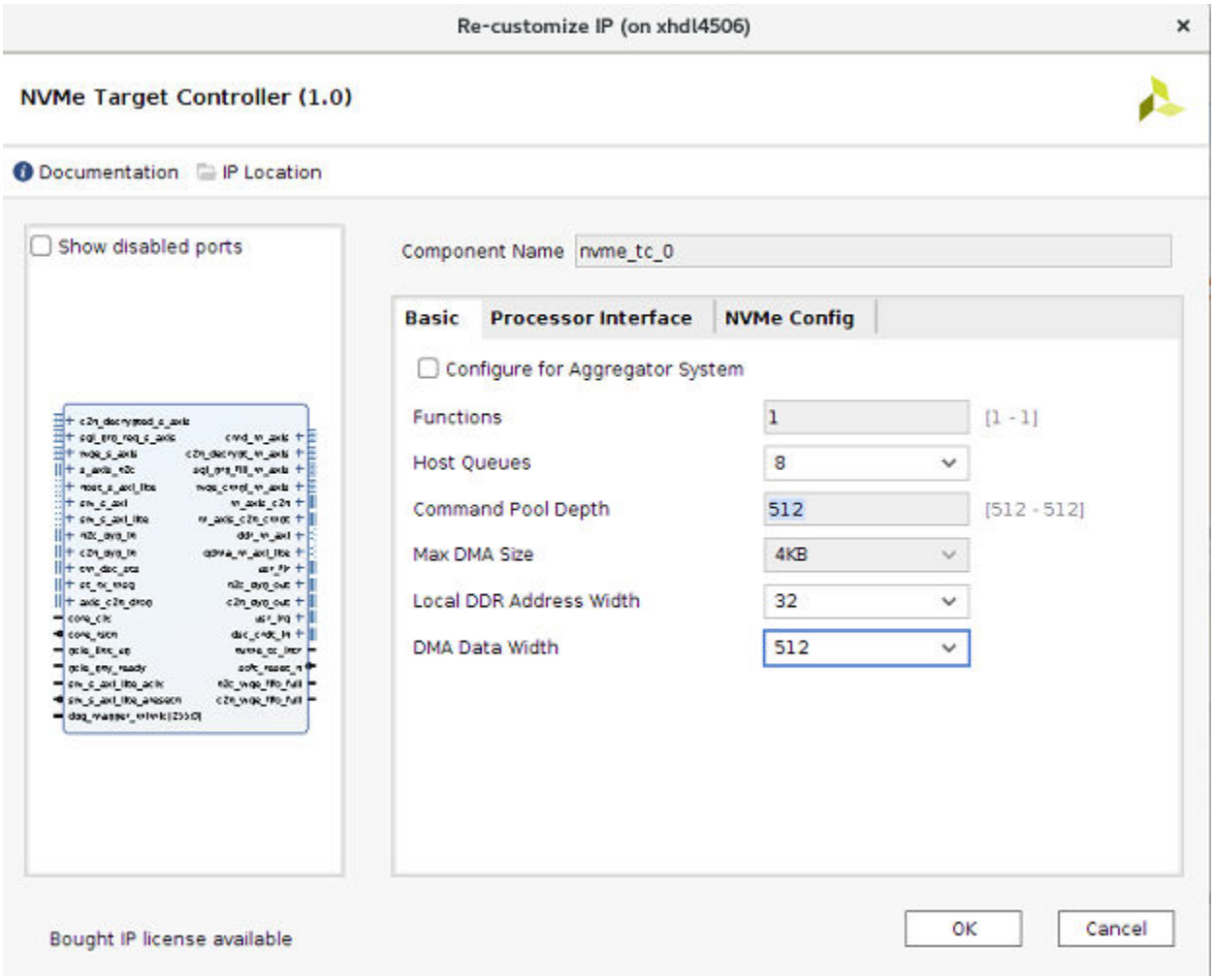
Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Configuration Tabs

Basic Tab

The Basic tab is shown in the following figure:

Figure 5: **Basic Tab**



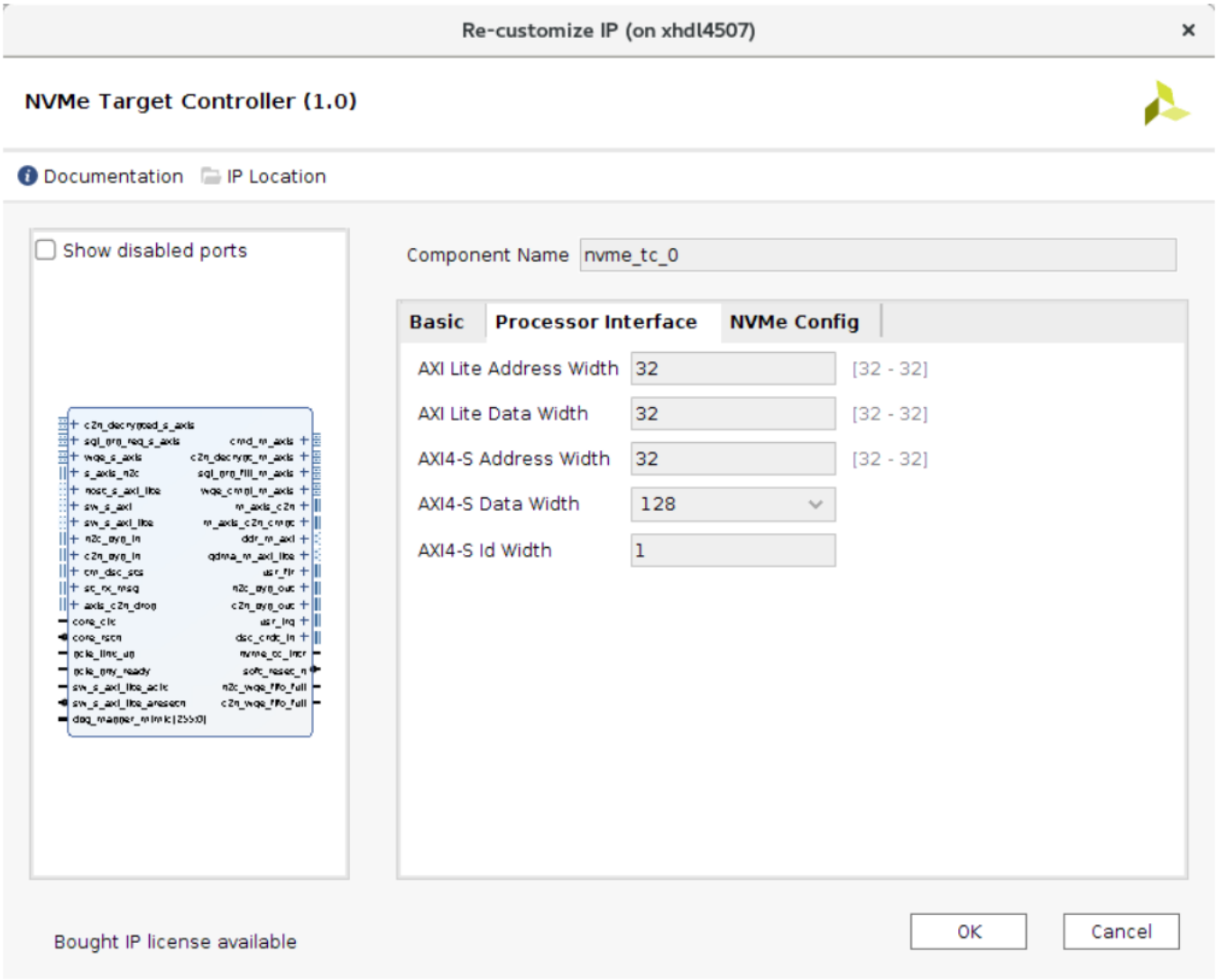
- **Configure for Aggregator System:** Allows you to configure the IP to be used in the aggregator solution.
- **Functions:** Number of functions/NVMe endpoint controllers.
- **Command Pool Depth:** Number of commands to pool for the hardware application.
- **Maximum DMA Size:** 4 KB tuned for QDMA requirement
- **Local DDR Address Width:** NVMe TC subsystem local address bus width

- **DMA Data Width:** This should match with the QDMA lane width and AXI data width.

Processor Interface Tab

The Processor Interface Tab is shown in the following figure:

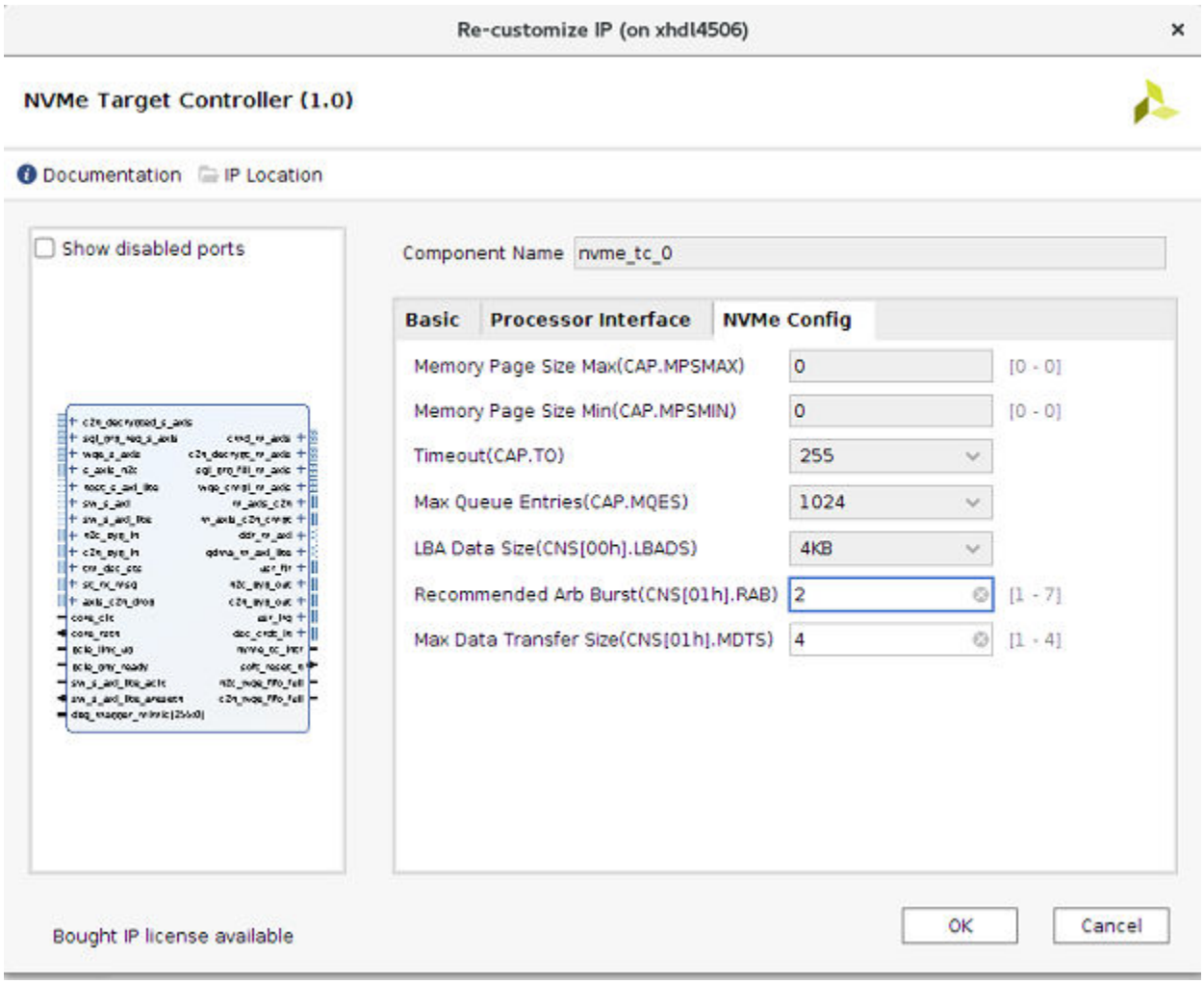
Figure 6: **Processor Interface Tab**



NVMe Config Tab

The NVMe Config tab shows NVME CAP and CNS fields configuration. It is shown in the following figure.

Figure 7: **NVMe Config Tab**



User Parameters

The following table shows the relationship between the fields in the Vivado® IDE and the user parameters (which can be viewed in the Tcl Console).

Table 32: **User Parameters**

Vivado IDE Parameter/ Value[1]	User Parameter	Valid Values	Default Value
Configure for Aggregator System	C_AGGREGATOR_SYS	TRUE/FALSE	FALSE
Number of Functions/ Controllers	C_NUM_FUNC	1	1
Host Queues	C_NUM_HSQ	2, 4, 8, 16, 32, 64	8
Command Pool Depth	C_NUM_CMD_INDX	512	512

Table 32: **User Parameters** *(cont'd)*

Vivado IDE Parameter/ Value[1]	User Parameter	Valid Values	Default Value
LBA DATA SIZE (Bytes)	C_LBA_DATA_SIZE	4096	4096
Maximum DMA Transfer Size (Bytes)	C_MAX_DMA_SIZE	4096	4096
Maximum Data Transfer Size (MDTS) (units of minimum page size)	C_MDTS	1-4	4
Local DDR Address Width	C_M_AXI_ADDR_WIDTH	32, 64	32
DMA Data Width	C_M_AXI_DATA_WIDTH	128, 256, 512	512
Recommended Arb Burst	C_ARB_BURST	1-7	2

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

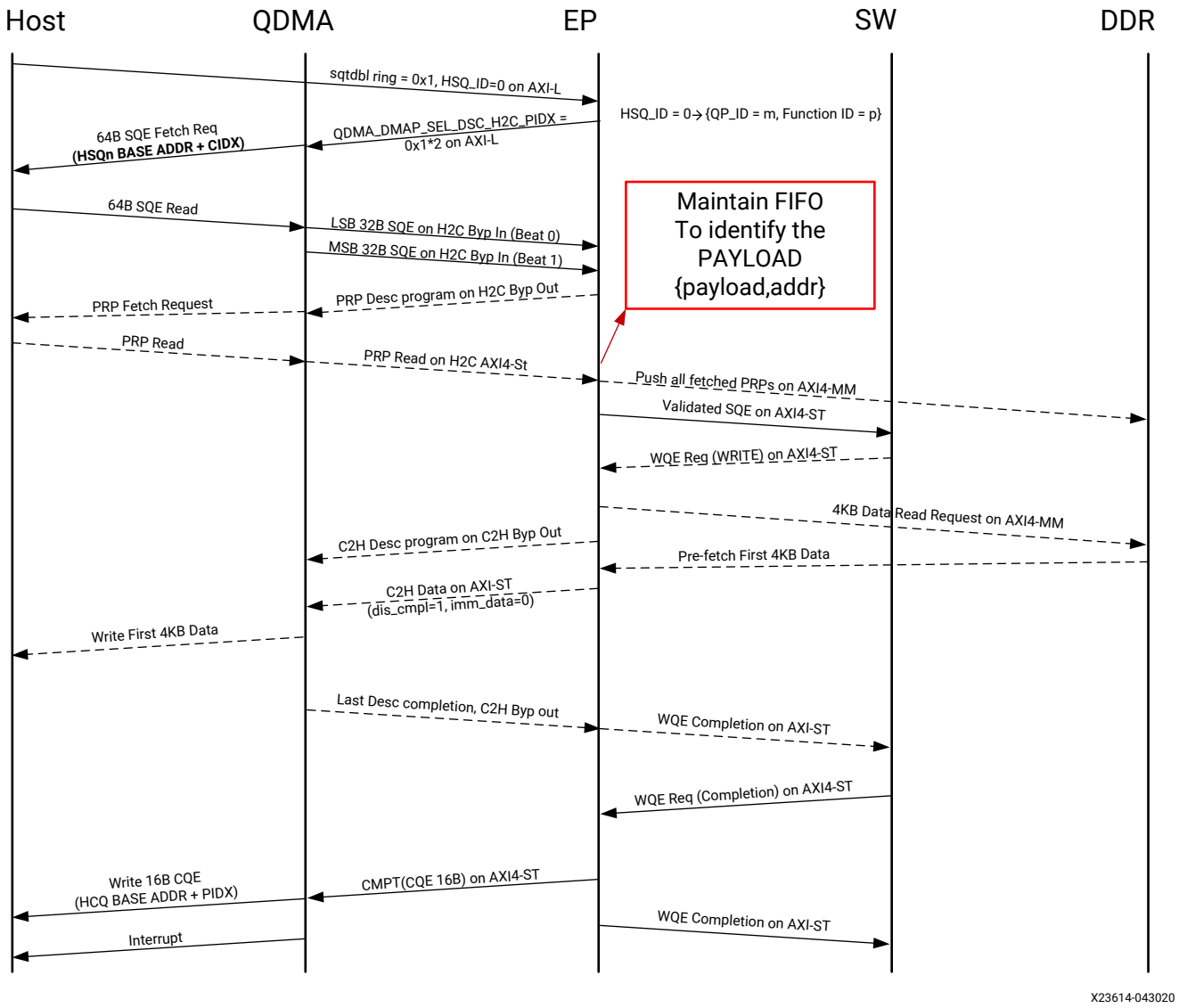
For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

NVMe I/O and Admin Commands Data Flow

NVMe Admin Command Flow with C2H Data Transfer

The following figure shows the NVMe admin command flow with C2H data transfer.

Figure 8: **Admin Command Flow with C2H Data Transfer**



The process is described in the following steps:

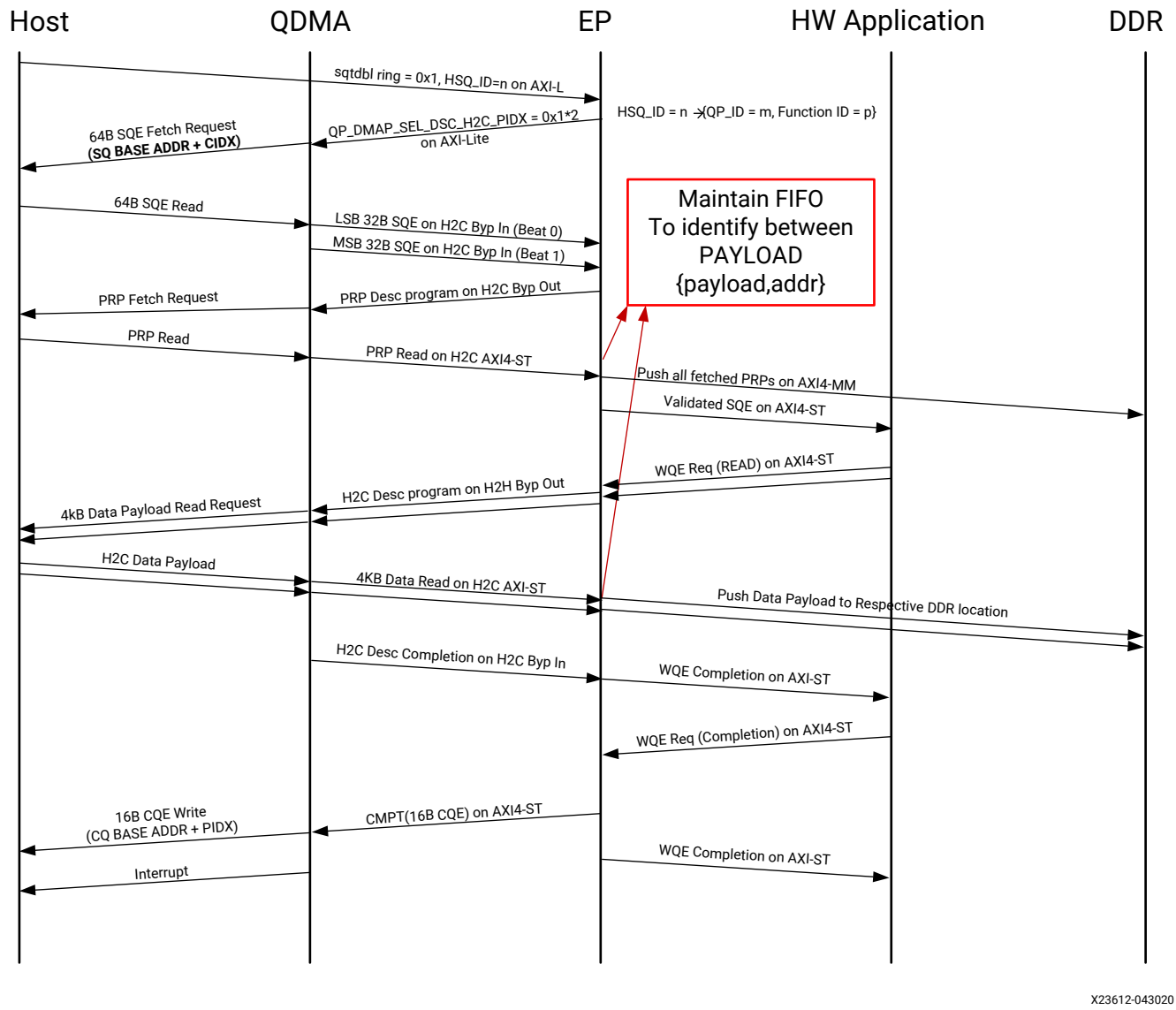
- The host writes the SQ Doorbell through the `host_s_axi_lite` interface.
- The NVMe TC IP issues an SQE Fetch request to read 64B SQE from the host.
- Fetch further PRP lists, if required.
- Validate the SQE and push to firmware.
- Firmware issues the WQE Request for WRITE.
- The NVMe TC IP programs the C2H Descriptor Bypass Out interface, reads data from DDR, and pushes to the C2H Data AXIS Interface.
- The NVMe TC IP pushes the WQE Completion to firmware.

- The firmware issues a WQE Completion Request.
- The NVMe TC IP sends 16B Completion Queue Entry (CQE) to the host and interrupt host.

NVMe I/O Write Command Flow (H2C Data Transfer)

The following figure shows the NVMe I/O write command flow (H2C data transfer).

Figure 9: **NVMe WRITE Data Flow**



The process is described in the following steps:

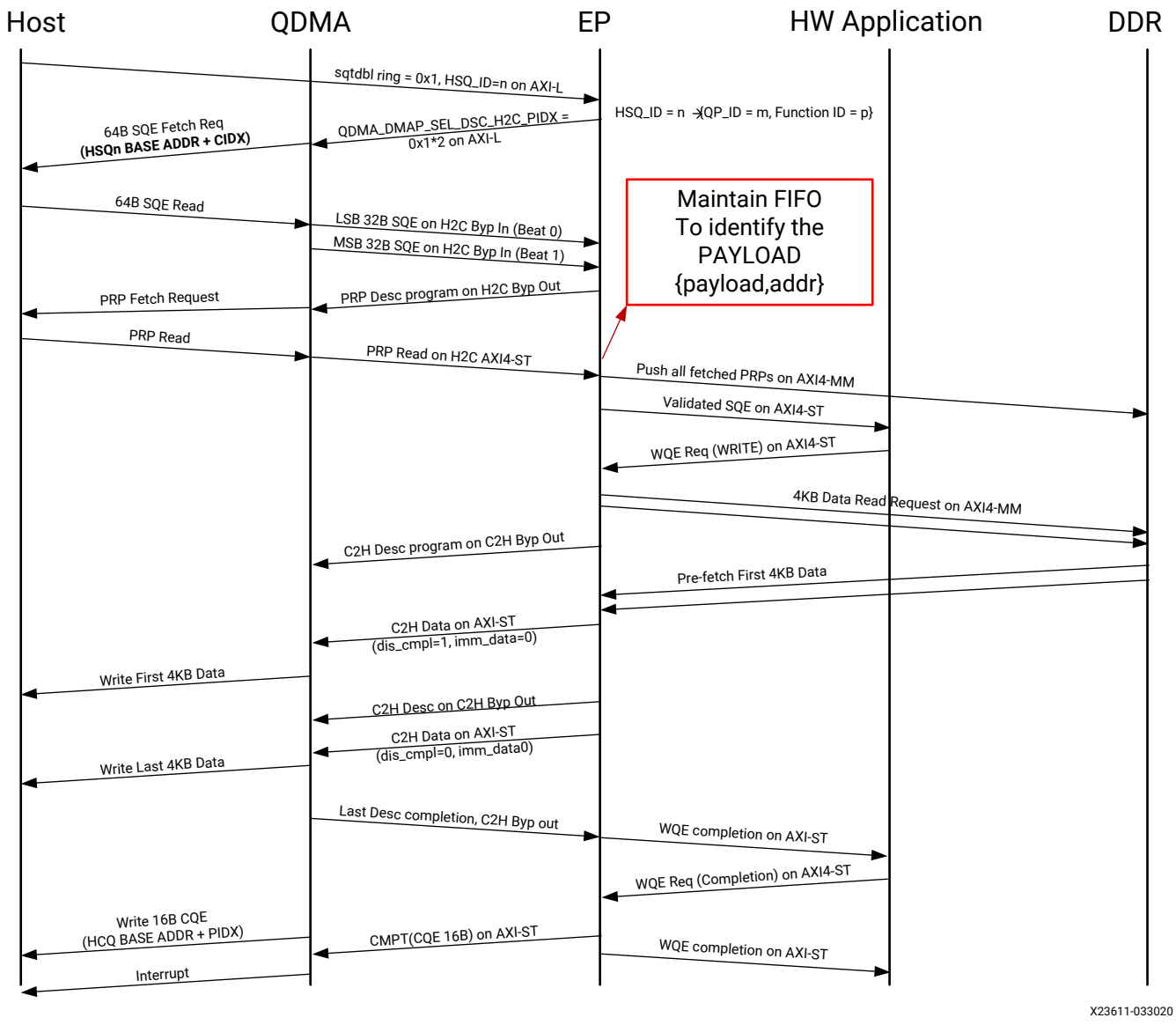
- The host writes the SQ Doorbell through the `host_s_axi_lite` interface.
- The NVMe TC IP issues an SQE Fetch request to read 64B SQE from the host.

- Fetch further PRP lists, if required.
- Validate the SQE and push to the hardware application.
- The hardware application issues a WQE Request for READ.
- The NVMe TC IP programs the H2C Descriptor Bypass Out interface, reads data from DDR, and pushes it to the H2C Data AXIS Interface.
- The NVMe TC IP pushes the WQE Completion to the hardware application.
- The hardware application issues the WQE Completion Request.
- The NVMe TC IP sends 16B Completion Queue Entry (CQE) to the host and interrupt host.

NVMe I/O Read Command Flow (C2H Data Transfer)

The following figure shows the NVMe I/O read command flow (C2H data transfer).

Figure 10: **NVMe Read Data Flow**



The process is described in the following steps:

- The host writes the SQ Doorbell through the `host_s_axi_lite` interface.
- The NVMe TC IP issues a SQE Fetch request to read 64B SQE from the host.
- Fetch further PRP lists, if required.
- Validate the SQE and push to the hardware application.
- The hardware application issues the WQE Request for WRITE.
- The NVMe TC IP programs the C2H Descriptor Bypass Out interface, reads data from DDR, and pushes it to the C2H Data AXIS Interface.

- The NVMe TC IP pushes the WQE Completion to the hardware application.
- The hardware application issues the WQE Completion Request.
- The NVMe TC IP sends 16B Completion Queue Entry (CQE) to the host and interrupt host.

Constraining the Core

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This IP needs two clocks:

- A software clock `sw_s_axi_lite_aclk` with a frequency of 100 MHz.
- A host interface clock `core_clk` which is the output of the QDMA.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900).

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

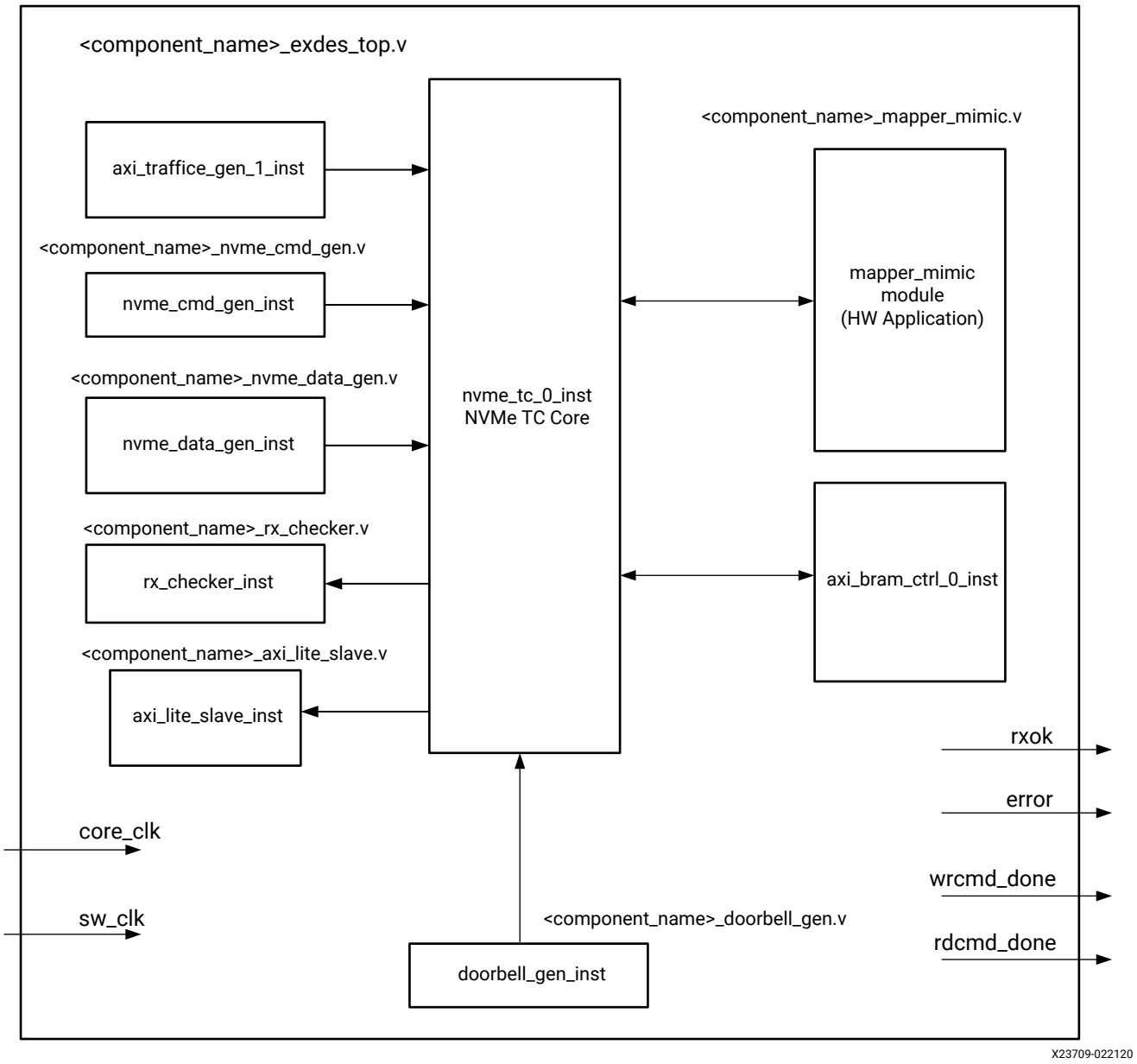
Example Design

This chapter provides information about the example design, including a description of the files and the directory structure generated by the Xilinx® Vivado® Design Suite, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

The example design includes the following modules

- **Driver:** This uses AXI Traffic Generator to initiate the NVMe™ TC IP
- **Command Generator:** NVMe command generator
- **Data Generator:** Payload generator for H2C interface
- **RX Checker:** Receives and checks payload from C2H interface
- **axi_lite_slave:** Emulates QDMA registers
- **Doorbell Generator:** Emulates QDMA axi_master interface that updates NVMe doorbell registers
- **HW Application:** Mapper Mimic module is sample HW application that interacts with NVMe TC IP to emulate NVMe Storage (A simple BRAM at backend)

Figure 11: **Core Example Design**



Test Bench

Test bench provides the input clocks to the example design and checks for status once the `write` and `read` commands are executed.

If the test passes during simulation, the following message is displayed:

```
Test Completed Successfully
```

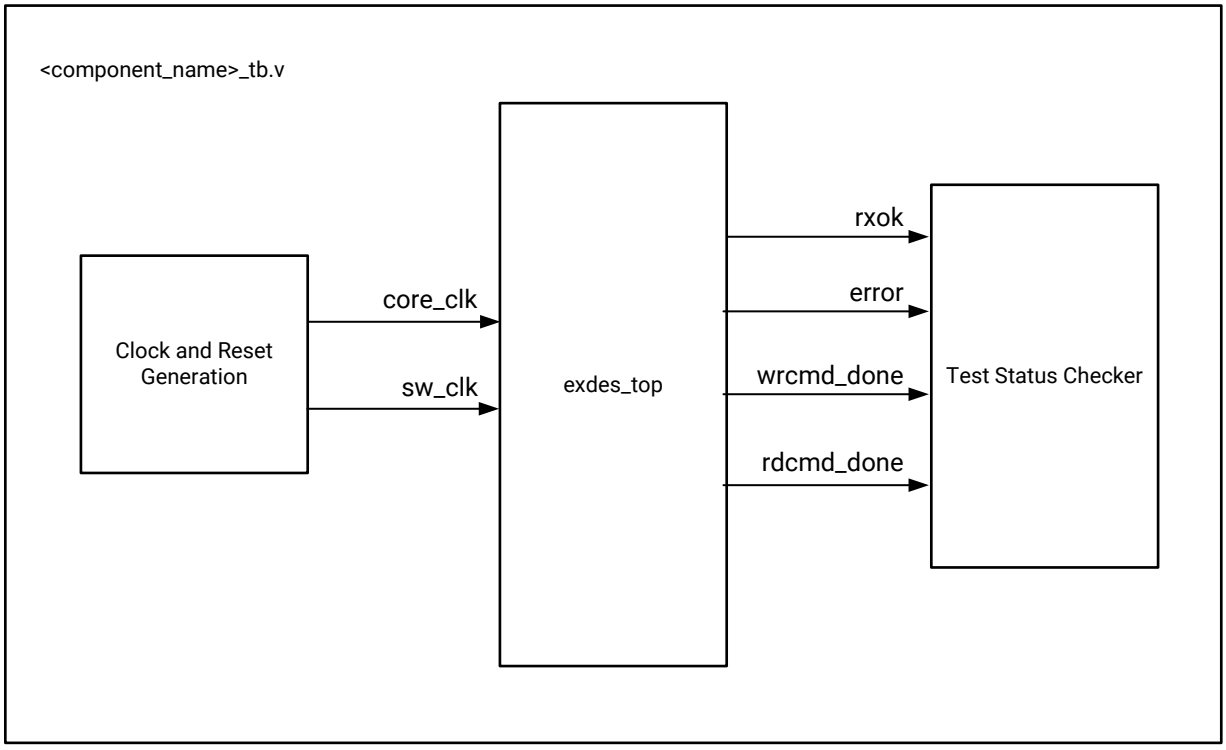
If the test fails, then the following message is displayed:

```
ERROR: Test Failed
```

In case of timeout, the following message is displayed:

```
ERROR:Test did not complete (timed-out)
```

Figure 12: **Test Bench**



Appendix A

Verification, Compliance, and Interoperability

Compliance Testing

Xilinx® NVMe TC core Version 1.00 passed NVME LECROY Conformance Tests.

Interoperability

The NVMe Target Controller IP core is successfully tested for compliance and interoperability at the University of New Hampshire Interoperability (UNH) Lab. Please contact Xilinx marketing for further details.

Xilinx NVMe Target Controller IP based computational storage array is listed in the NVMe Integrator's List .

Appendix B

Upgrading

This appendix is not applicable for the first release of the core.

Appendix C

Debugging

This appendix includes details about resources available on the Xilinx® Support website and debugging tools.

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT! *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Finding Help on Xilinx.com

To help in the design and debug process when using the core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator. Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the Core/Subsystem

AR 73241.

Technical Support

Xilinx provides technical support on the Xilinx Community Forums for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To ask questions, navigate to the Xilinx Community Forums.

Debug Tools

There are many tools available to address NVMe Target Controller design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx® devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado® debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.
- If your outputs go to 0, check your licensing.

XILINX

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

Locating Design Files

1. Download the reference design files from the NVMe Target Controller lounge. Registration is required to access the lounge.
2. Extract the ZIP file contents into any write-accessible location.

References

These documents provide supplemental material useful with this guide:

1. *Vivado Design Suite User Guide: Designing with IP* (UG896)
2. *QDMA Subsystem for PCI Express Product Guide* (PG302)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
06/03/2020 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright