Design Gateway Co.,Ltd
E-mail: ip-sales@design-gateway.com
URL: design-gateway.com

Core Facts

Provided with Core	
Documentation	Reference Design Manual Demo Instruction Manual
Design File Formats	Encrypted Netlist
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on VCK190, Alveo U50
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- Access one NVMe Gen4 SSD without CPU and external memory
- Include 256-Kbyte RAM to be data buffer
- Simple user interface by dgIF typeS
- Support six commands, i.e., Identify, Shutdown, Write, Read, SMART and Flush
- Supported NVMe device
  - Base Class Code:01h (mass storage), Sub Class Code:08h (Non-volatile), Programming Interface:02h (NVMHCI)
  - MPSMIN (Memory Page Size Minimum): 0 (4Kbyte)
  - MDTS (Maximum Data Transfer Size): At least 5 (128 Kbyte) or 0 (no limitation)
  - LBA unit: 512 bytes or 4096 bytes
- User clock frequency: More than or equal to PCIe clock frequency (250MHz for Gen4)
- PCIe Hard IP: Integrated Block for PCI Express from Xilinx (256-bit interface of 4-lane Gen4)
- One NVMe IP connects to one NVMe SSD directly
- Available reference design: 1-ch demo and 4-ch RAID0 demo
  - VCK190 board with AB18-PCIeX16 adapter board
  - Alveo U50 board with customized AB18-PCIeX16 adapter board
- Customized service for following features
  - Additional NVMe commands
  - RAM size or RAM type modification

Table 1: Example Implementation Statistics (Versal)

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	Slice[1]	IOB	BRAMTile[1]	URAM	Design Tools
Versal AI Core	XCVC1902-VSVA2197-2MP-E-S	375	6270	3848	1050	-	4	8	Vivado2022.1

Table 2: Example Implementation Statistics (UltraScale+)

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB[1]	IOB	BRAMTile[1]	URAM	Design Tools
Alveo-U50	XCU50-FSVH2104-2-E	375	6525	3805	1093	-	4	8	Vivado2021.1

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

Applications

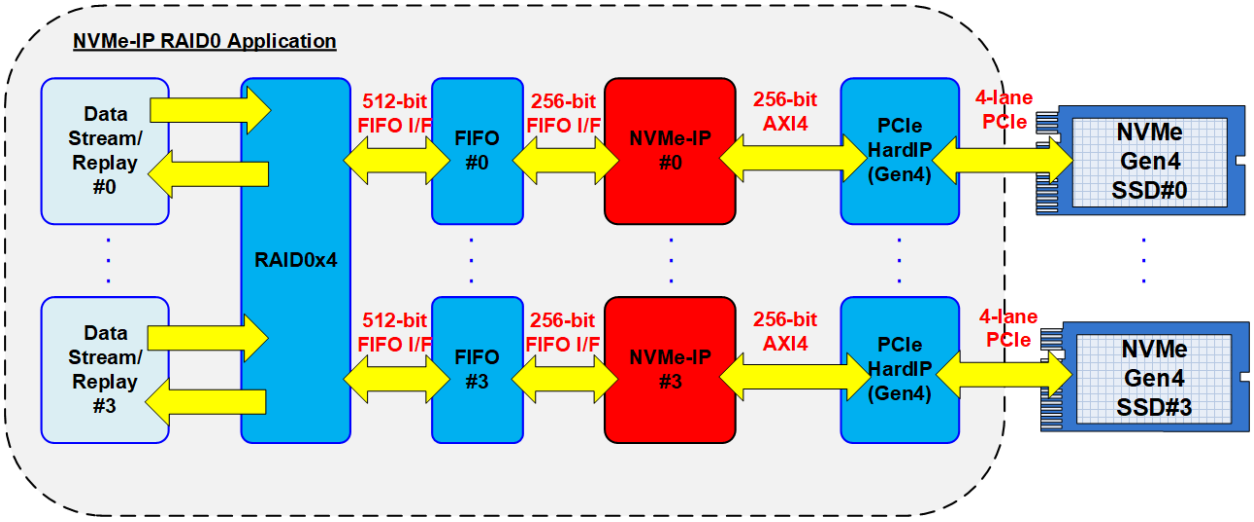

Figure 1: NVMe IP Application

NVMe IP Core integrated with Integrated Block for PCI Express (PCIe hard IP) from Xilinx is ideal to access NVMe Gen4 SSD without CPU and external memory such as DDR. 256 Kbyte buffer implemented by UltraRAM is included in NVMe IP Core to be data buffer between user logic and NVMe Gen4 SSD. One PCIe hard IP is connected to one NVMe-IP and one NVMe Gen4 SSD by using 4-lane PCIe interface. Therefore, 4-ch RAID0 system can be designed to increase transfer speed up to four times of one SSD performance, as shown in Figure 1. Besides, the total storage capacity of RAID0 system is increased to four times. One Gen4 SSD write/read performance can achieve 6000 Mbyte/sec, so using four Gen4 SSDs can transfer up to 24 Gbyte/sec.

We also provide alternative IP cores for more specific applications such as Multiple users, Random access, PCIe switch, and Soft IP core.

Multiple User NVMe IP Core– To access one NVMe SSD by multiple users. It is the solution for the application that needs to write and read the SSD with high-performance at the same time.

https://dgway.com/muNVMe-IP_X_E.html

Random Access NVMe IP Core– To access NVMe SSD with multiple commands, individual address for each. Recommended for the application which requires the access in non-contiguous area.

https://dgway.com/raNVMe-IP_X_E.html

NVMe IP Core for PCIe Switch– For access one or multiple NVMe SSDs via PCIe switch.

https://dgway.com/NVMe-IP_X_E.html

NVMe IP Core with PCIe Gen 3/Gen4 Soft IP – When the selected FPGA does not have PCIe hard IP to access the SSD.

https://dgway.com/NVMeG4-IP_X_E.html

General Description

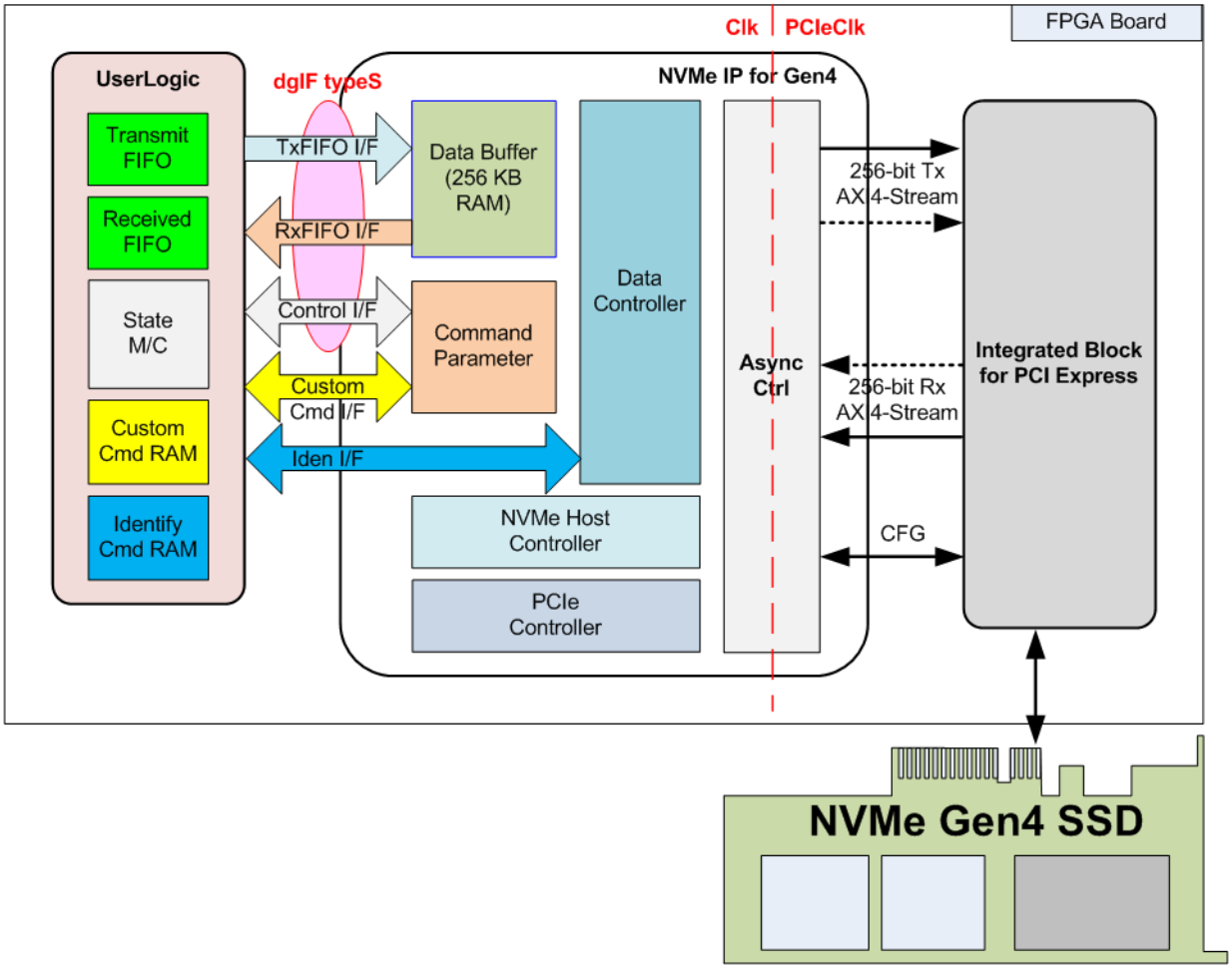

Figure 2: NVMe IP Block Diagram

NVMe IP implements the host controller to access NVMe SSD following NVM express standard. Physical interface of NVMe SSD is PCIe. The lower layer hardware is implemented by Integrated Block for PCI Express (PCIe hard IP) from Xilinx.

NVMe IP supports six NVMe commands, i.e., Identify, Shutdown, Write, Read, SMART, and Flush command by using two user interface groups. First is Control interface for transferring command and the parameters. Another is Data interface for transferring data when the command must have the data transferring. Control interface and Data interface for Write/Read command use dgIF typeS format. Control interface of dgIF typeS consists of start address and transfer length with asserting the request signal while Data interface of dgIF typeS is the FIFO interface.

SMART and Flush command are Custom command which use Ctm I/F for control path and Ctm RAM I/F for data path. Identify command uses the same Control interface as Write or Read command, but uses its own data interface - Iden I/F, as shown in Figure 2.

While running initialization process or operating some commands, error signal may be asserted by NVMe IP if some abnormal conditions are found. The IP includes the error status to check more details of error condition. To recover error status, NVMe IP and SSD must be reset.

There is one limitation about clock frequency of user logic. Transmit packet to PCIe hard IP must be sent continuously until end of packet. Therefore, data must be valid every clock between start of packet and end of packet. To support this feature, user logic clock frequency must be more than or equal to PCIe clock frequency (250 MHz) to have the bandwidth of transmit logic higher than or equal to PCIe hard IP bandwidth.

The reference designs on FPGA evaluation boards are available for evaluation before purchasing.

Functional Description

Figure 3 shows the operation flow of NVMe IP after IP reset is de-asserted. There are three phases, i.e., IP initialization, Operating command, and Inactive status.

After finishing IP initialization, the first command that user runs must be Identify command to check device status and capacity. After finishing all operations, the last command before shutdown system should be Shutdown command for safety operation.

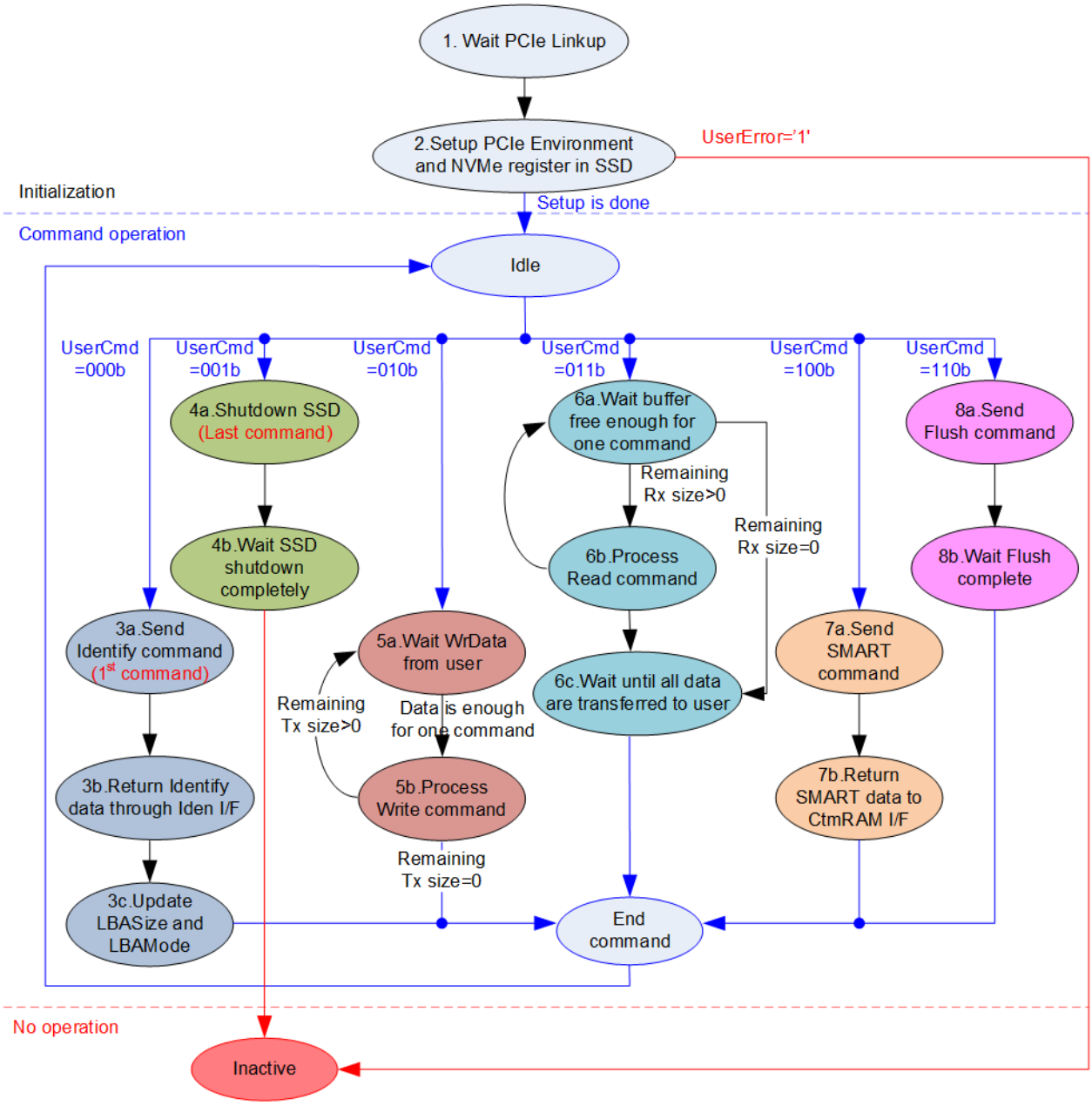

Figure 3: NVMe IP Operation Flow

The operation flow of NVMe IP is described as follows.

1) IP waits until PCIe is ready by monitoring Linkup status from PCIe IP core.
2) IP begins the initialization process by configurating PCIe and NVMe registers. After that, the IP enters to the Idle state to wait for a new command request from user. If some errors are detected during initialization process, the IP goes to the Inactive state with asserting UserError to '1'.
3) The first command from user must be Identify command (UserCmd=000b) to update LBASize (disk capacity) and LBAMode (LBA unit=512 byte or 4 Kbyte).
4) The last command before power down the system must be Shutdown command (UserCmd=001b). This command is recommended to guarantee SSD powered down in a good sequence. Without Shutdown command, Write data in SSD cannot be guaranteed. After finishing Shutdown command, NVMe IP and SSD change to the Inactive state. The new command cannot be opreated until the IP is reset.
5) For Write command (UserCmd=010b), the maximum data size of one command is 128 Kbyte. If total length from user is more than 128 Kbyte, the IP repeats step 5a) – 5b) automatically until total data are completely transferred.
    a) The IP waits until Write data, sent by user, is enough for one command (transfer size of one command in NVMe IP is 128 Kbyte, except the last loop which could be less than 128 Kbyte).
    b) The IP sends Write command to SSD and then waits until the status is returned from SSD. The IP returns to the Idle state when total data are completely transferred. Otherwise, the IP goes back to step 5a) to send the next Write command.
6) Similar to Write command, when running Read command (UserCmd=011b) which has the transfer size more than 128 Kbyte, the IP must repeat step 6a) – 6b) many times.
    a) If remaining transfer size is equal to zero, the IP skips to step 6c). Otherwise, the IP waits until free space of data buffer in NVMe IP is enough for one command (128 Kbyte or remaining transfer size for the last loop).
    b) The IP sends Read command to SSD and then returns to step 6a).
    c) IP waits until all data are completely transferred from data buffer to user logic and then returns to the Idle state. Therefore, data buffer is empty after finishing Read command.
7) For SMART command (UserCmd=100b), 512-byte data is returned after finishing the operation.
    a) IP sends Get Log Page command to read SMART/Health information from the SSD.
    b) 512-byte data is returned from the SSD. The IP forwards the data through Custom command RAM interface (CtmRamAddr=0x000 – 0x01F).
8) For Flush command (UserCmd=110b), there is no data transferring while operating.
    a) IP sends Flush command to the SSD.
    b) IP waits until SSD returns status to complete the operation.

To design NVMe host controller, NVMe IP implements two protocols, i.e., NVMe protocol for interface with user and PCIe protocol for interface with PCIe hard IP. Therefore, the hardware inside NVMe IP can be split into two groups, NVMe and PCIe. More details of each module are described as follows.

NVMe

Six commands that NVMe supports can be divided to two command types - Admin command and NVM command. Admin command consists of three commands - Identify, Shutdown, and SMART command while NVM command consists of three commands - Write, Read, and Flush command. After finishing operating the command, the status returned from the SSD is latched to AdmCompStatus (status returned from Admin command) or IOCompStatus (status returned from NVM command), depending on the command types.

The parameters of Write or Read command are set by Control interface of dgIF typeS while the parameters of SMART or Flush command are set by CtmSubmDW0-15 of Ctm interface. Data interface for Write or Read command is transferred by FIFO interface, a part of dgIF typeS. The data of Write command and Read command are stored to 256 Kbyte buffer inside the IP. The data interface of other commands has its own interface - Identify RAM for Identify command and Custom RAM for SMART command.

The details of each submodule are described as follows.

- **NVMe Host Controller**

  NVMe host controller is the core controller in NVMe IP. Similar to the operation flow, the controller has two phases for operating. First is the initialization phase which is once run after the system is boot up for setting NVMe register inside the SSD. After finishing the initialization phase, the next phase is operating the command. The controller controls the order of transmitted packet and received packet for each command.

  To start operating each command, the parameters of the command are latched to Command Parameter for creating the packet. After that, the packet is forwarded to AsyncCtrl to convert NVMe packet to PCIe packet. After each command operation is done, the status packet is returned from SSD. The controller decodes the status value and check if the operation is complete or error. If the command needs to transfer the data such as Write command and Read command, the controller must handle the order of data packet that is created and decoded by Data controller.

- **Command Parameter**

  This module creates Command packet sent to SSD. Also, the status packet returned from SSD is decoded by this module. However, the input/output of this module are controlled by the NVMe host controller. Typically, the command consists of 16 Dwords (1 Dword = 32-bit). When running Identify, Shutdown, Write, and Read command, all 16 Dwords are created by Command parameters that are loaded from the user inputs on dgIF typeS. When running SMART and Flush command, all 16 Dwords are directly loaded via CtmSubmDW0-CtmSubmDW15 of Ctm interface.

- **Data Buffer**
  256-Kbyte simple dual port RAM is implemented by URAM to be data buffer. The buffer stores data that is stored in SSD while operating Write command and Read command.

- **Data Controller**

  This module is operated when the command must transfer the data, i.e., Identify, SMART, Write, and Read command. There are three data interfaces for transferring with the SSD - FIFO interface with 256-Kbyte buffer when running Write or Read command, Custom command RAM interface when running SMART command, and Identify interface when running Identify command. The data packet is created and decoded by this module. Similar to Command Parameter module, Data controller input and output signals are controlled by the NVMe host controller.

PCIe

The PCIe standard is the outstanding low-layer protocol for very high-speed application. The NVMe standard is the protocol which is run over PCIe protocol. In the initialization process, NVMe layer is setup after finishing PCIe layer setup. Two modules are designed to support PCIe protocol - PCIe controller and AsyncCtrl. More details of each module are described as follows.

- **PCIe Controller**

  In initialization process, PCIe controller sets up PCIe environment of SSD via CFG interface. After that, PCIe packet is created or decoded via 256-bit Tx/Rx AXI4-Stream. The command packet and the data packet from NVMe module are converted to be PCIe packet by PCIe controller and vice versa.

- **AsyncCtrl**

  AsyncCtrl includes asynchronous registers and asynchronous buffers to support clock domain crossing. Most logics in NVMe IP run on user clock domain while PCIe hard IP runs on PCIe clock domain. To transfer a packet to PCIe hard IP continuously, the user interface bandwidth must be greater than or equal to PCIe interface bandwidth. Thus, the user clock frequency must be higher than or equal to PCIe clock frequency.

User Logic

This module could be designed by using small state machine to send the commands and the parameters for each command. For example, the address and transfer size which are the parameters for Write or Read command are designed by using simple registers. While FIFO is connected for transferring data in Write and Read command. The data output of SMART command and Identify command interface connects to simple dual port RAM with byte enable. The data width of FIFO and RAM are 256-bit while the memory depth can be set by different value. Data size of Identify command is 8 Kbytes while data size of SMART command is 512 bytes.

Integrated Block for PCI Express

Some UltraScale+ and Versal devices have Integrated Block for PCI Express (PCIe hard IP), called PCIe4C and PL PCIE4 respectively, to operate at PCIe Gen4 protocol. To connect with NVMe IP, PCIe4C or PL PCIE4 is configured to be 4-lane PCIe Gen4 by using 256-bit data interface. One NVMe IP connects to one PCIe hard IP for controlling one NVMe Gen4 SSD. Therefore, the maximum number of SSDs connecting to one FPGA device is limited by the number of PCIe hard IPs in FPGA device.

The IP wizard on Xilinx tool to generate PCIe hard IP when selecting UltraScale+ and Versal device are different. On UltraScale+ device, the user uses one IP wizard to generate the PCIe hard IP integrating with the transceiver. On Versal device, the user needs to call two IP wizards to generate PCIe hard IP and the Transceiver individually.

More details of PCIe hard IP are described in following document.

PG213: UltraScale+ Devices Integrated Block for PCI Express
https://www.xilinx.com/products/intellectual-property/pcie4-ultrascale-plus.html#documentation
PG343: Versal ACAP Integrated Block for PCI Express
https://www.xilinx.com/products/intellectual-property/pcie-versal.html#documentation

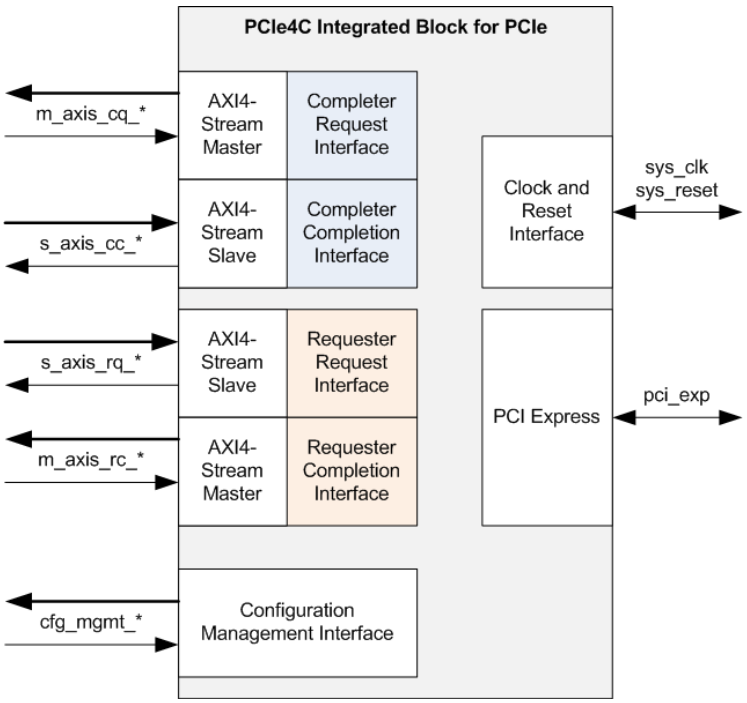

Figure 4: PCIe4C Integrated Block for PCI Express

Core I/O Signals

Descriptions of all signal I/O are provided in Table 3 - Table 4.

Table 3: User logic I/O Signals (Synchronous to Clk signal)

Signal	Dir	Description
Control I/F of dgIF typeS		
RstB	In	Synchronous reset signal. Active low. De-assert to '1' when Clk signal is stable.
Clk	In	System clock for running NVMe IP. The frequency must be more than or equal to PCIeClk which is output from PCIe hard IP (250 MHz for PCIe Gen4).
UserCmd[2:0]	In	User Command. Valid when UserReq='1'. (000b: Identify, 001b: Shutdown, 010b: Write SSD, 011b: Read SSD, 100b: SMART, 110b: Flush, 101b/111b: Reserved)
UserAddr[47:0]	In	Start address to write/read SSD in 512-byte unit. Valid when UserReq='1'. In case LBA unit = 4 Kbyte, UserAddr[2:0] must be always set to 000b to align 4-Kbyte unit. In case LBA unit = 512 byte, it is recommended to set UserAddr[2:0]=000b to align 4-Kbyte size (SSD page size). Write/Read performance of most SSDs is reduced when start address is not aligned to page size.
UserLen[47:0]	In	Total transfer size to write/read SSD in 512-byte unit. Valid from 1 to (LBASize-UserAddr). In case LBA unit = 4 Kbyte, UserLen[2:0] must be always set to 000b to align 4-Kbyte unit. Valid when UserReq='1'.
UserReq	In	Assert to '1' to send the new command request and de-assert to '0' after IP starts the operation by asserting UserBusy to '1'. This signal can be asserted when the IP is Idle (UserBusy='0'). Command parameters (UserCmd, UserAddr, UserLen, and CtmSubmDW0-DW15) must be valid and stable when UserReq='1'. UserAddr and UserLen are inputs for Write/Read command while CtmSubmDW0-DW15 are inputs for SMART/Flush command.
UserBusy	Out	Asserted to '1' when IP is busy. New request must not be sent (UserReq to '1') when IP is busy.
LBASize[47:0]	Out	Total capacity of SSD in 512-byte unit. Default value is 0. This value is valid after finishing Identify command.
LBAMode	Out	LBA unit size of SSD ('0': 512 bytes, '1': 4 Kbytes). Default value is 0. This value is valid after finishing Identify command.
UserError	Out	Error flag. Asserted to '1' when UserErrorType is not equal to 0. The flag is cleared to '0' by asserting RstB to '0'.
UserErrorType[31:0]	Out	Error status. [0] – Error when PCIe class code is not correct. [1] – Error from CAP (Controller capabilities) register which may be caused from - MPSMIN (Memory Page Size Minimum) is not equal to 0. - NVM command set flag (bit 37 of CAP register) is not set to 1. - DSTRD (Doorbell Stride) is not 0. - MQES (Maximum Queue Entries Supported) is more than or equal to 7. More details of each register can be checked from NVMeCAPReg signal. [2] – Error when Admin completion entry is not received until timeout. [3] – Error when status register in Admin completion entry is not 0 or phase tag/command ID is invalid. Please see more details from AdmCompStatus signal. [4] – Error when IO completion entry is not received until timeout. [5] – Error when status register in IO completion entry is not 0 or phase tag is invalid. Please see more details from IOCompStatus signal.

Signal	Dir	Description
Control I/F of dgIF typeS		
UserErrorType[31:0]	Out	[6] – Error when Completion TLP packet size is not correct. [7] – Error when PCIe hard IP detects Error correction code (ECC) error from the internal buffer. [8] – Error from Unsupported Request (UR) flag in Completion TLP packet. [9] – Error from Completer Abort (CA) flag in Completion TLP packet. [15:10] – Reserved [16] - Error from unsupported LBA unit (LBA unit is not equal to 512 bytes or 4 Kbytes) [31:17] – Reserved *Note: Timeout period of bit[2]/[4] is set from TimeOutSet input.*
Data I/F of dgIF typeS		
UserFifoWrCnt[15:0]	In	Write data counter of Receive FIFO. Used to check full status. When full status is detected, the returned data transmission from Read command may be paused. If FIFO data counter signal is less than 16 bits, please fill '1' to upper bit.
UserFifoWrEn	Out	Asserted to '1' to write data to Receive FIFO when running Read command.
UserFifoWrData[255:0]	Out	Write data bus of Receive FIFO. Valid when UserFifoWrEn='1'.
UserFifoRdCnt[15:0]	In	Read data counter of Transmit FIFO. Used to check data size stored in FIFO. The transmitted data packet for Write command may be paused when the counter shows empty status. If FIFO data counter signal is less than 16 bits, please fill '0' to upper bit.
UserFifoEmpty	In	The signal is unused for this IP.
UserFifoRdEn	Out	Asserted to '1' to read data from Transmit FIFO when running Write command.
UserFifoRdData[255:0]	In	Read data returned from Transmit FIFO. Valid in the next clock after UserFifoRdEn is asserted to '1'.
NVMe IP Interface		
IPVesion[31:0]	Out	IP version number
TestPin[31:0]	Out	Reserved to be IP Test point.
TimeOutSet[31:0]	In	Timeout value to wait completion from SSD. Time unit is equal to 1/(Clk frequency). When TimeOutSet is set to 0, Timeout function is disabled.
PCIeLinkup	In	Asserted to '1' when LTSSM state of PCIe hard IP is in L0 State.
AdmCompStatus[15:0]	Out	Status output from Admin Completion Entry [0] – Set to '1' when Phase tag or Command ID in Admin Completion Entry is invalid. [15:1] – Status field value of Admin Completion Entry
IOCompStatus[15:0]	Out	Status output from IO Completion Entry [0] – Set to '1' when Phase tag in IO Completion Entry is invalid. [15:1] – Status field value of IO Completion Entry
NVMeCAPReg[31:0]	Out	The parameter value of the NVMe capability register when UserErrorType[1] is asserted to '1'. [15:0] – MQES (Maximum Queue Entries Supported) [19:16] – DSTRD (Doorbell Stride) [20] – NVM command set flag [24:21] – MPSMIN (Memory Page Size Minimum) [31:25] – Undefined

Signal	Dir	Description
Identify Interface		
IdenWrEn	Out	Asserted to '1' for sending data output from Identify command.
IdenWrDWEn[7:0]	Out	Dword (32-bit) enable of IdenWrData. Valid when IdenWrEn='1'. '1': this dword data is valid, '0': this dword data is not available. Bit[0], [1], …, [7] corresponds to IdenWrData[31:0], [63:32], …, [255:224] respectively
IdenWrAddr[7:0]	Out	Index of IdenWrData in 256-bit unit. Valid when IdenWrEn='1'. 0x00-0x7F is 4Kbyte Identify controller data, 0x80-0xFF is 4Kbyte Identify namespace data.
IdenWrData[255:0]	Out	4Kbyte Identify controller data or Identify namespace data. Valid when IdenWrEn='1'.
Custom interface		
CtmSubmDW0[31:0] – CtmSubmDW15[31:0]	In	16 Dwords of Submission queue entry for SMART/Flush command. DW0: Command Dword0, DW1: Command Dword1, …, and DW15: Command Dword15. These inputs must be valid and stable when UserReq='1' and UserCmd=100b (SMART) or 110b (Flush).
CtmCompDW0[31:0] – CtmCompDW3[31:0]	Out	4 Dwords of Completion queue entry, output from SMART/Flush command. DW0: Completion Dword0, DW1: Completion Dword1, …, and DW3: Completion Dword3
CtmRamWrEn	Out	Asserted to '1' for sending data output from custom command such as SMART command.
CtmRamWrDWEn[7:0]	Out	Dword (32 bit) enable of CtmRamWrData. Valid when CtmRamWrEn='1'. '1': This dword data is valid, '0': This dword data is not available. Bit[0], [1], …, [7] corresponds to CtmRamWrData[31:0], [63:32], …, [255:224] respectively.
CtmRamAddr[7:0]	Out	Index of CtmRamWrData when SMART data is received. Valid when CtmRamWrEn='1'. (Optional) Index to request data input through CtmRamRdData for customized custom commands.
CtmRamWrData[255:0]	Out	512-byte data output from SMART command. Valid when CtmRamWrEn='1'.
CtmRamRdData[255:0]	In	(Optional) Data input for customized custom commands.

Table 4: Physical I/O Signals for PCIe4C/PL PCIE4 (Synchronous to PCIeClk)

Signal	Dir	Description
PCIe hard IP		
PCIeRstB	In	Synchronous reset signal active low. De-assert to '1' when PCIe hard IP is not in reset state.
PCIeClk	In	Clock output from PCIe hard IP (250 MHz for PCIe Gen4).
Configuration Management Interface		
PCIeCfgDone	In	Read/Write operation complete. Assert for 1 cycle when operation completes.
PCIeCfgRdEn	Out	Read enable. Asserted to '1' for a read operation.
PCIeCfgWrEn	Out	Write enable. Asserted to '1' for a write operation.
PCIeCfgWrData[31:0]	Out	Write data which is used to configure the Configuration and Management registers.
PCIeCfgByteEn[3:0]	Out	Byte enable for write data, where bit[0], [1], [2], and [3] corresponds to PCIeCfgWrData[7:0], [15:8], [23:16], and [31:24] respectively.
PCIeCfgAddr[9:0]	Out	Read/Write Address.
Requester Request Interface		
PCIeMtTxData[255:0]	Out	Requester request data bus.
PCIeMtTxKeep[7:0]	Out	Bit [i] indicates that Dword [i] of PCIeMtTxData contains valid data.
PCIeMtTxLast	Out	Asserted this signal in the last cycle of a TLP to indicate the end of the packet.
PCIeMtTxReady[3:0]	In	Assert to accept data. Data is transferred when both PCIeMtTxValid and PCIeMtTxReady are asserted in the same cycle.
PCIeMtTxUser[61:0]	Out	Requester request user data. Valid when PCIeMtTxValid is high.
PCIeMtTxValid	Out	Asserted to drive valid data on PCIeMtTxData bus. NVMe IP keeps the valid signal asserted during the transfer of a packet.
Completer Request Interface		
PCIeMtRxData[255:0]	In	Receive data from PCIe hard IP.
PCIeMtRxKeep[7:0]	In	Bit [i] indicates that Dword [i] of PCIeMtRxData contains valid data.
PCIeMtRxLast	In	Assert this signal in the last beat of a packet to indicate the end of the packet.
PCIeMtRxReady	Out	Indicate that NVMe IP is ready to accept data.
PCIeMtRxUser[74:0]	In	Sideband information for the TLP being transferred. Valid when PCIeMtRxValid is high.
PCIeMtRxValid	In	Assert when PCIe hard IP drives valid data on PCIeMtRxData bus. PCIe hard IP keeps the valid signal asserted during the transfer of packet.
Completer Completion Interface		
PCIeSlTxData[255:0]	Out	Completion data from NVMe IP.
PCIeSlTxKeep[7:0]	Out	Bit [i] indicates that Dword [i] of PCIeSlTxData contains valid data.
PCIeSlTxLast	Out	Assert this signal in the last cycle of a packet to indicate the end of the packet.
PCIeSlTxReady[3:0]	In	Indicate that PCIe hard IP is ready to accept data.
PCIeSlTxUser[32:0]	Out	Sideband information for the TLP being transferred. Valid when PCIeSlTxValid is high.
PCIeSlTxValid	Out	Assert to drive valid data on PCIeSlTxData bus. NVMe IP keeps the valid signal asserted during the transfer of a packet.
Requester Completion Interface		
PCIeSlRxData[255:0]	In	Receive data from PCIe hard IP.
PCIeSlRxKeep[7:0]	In	Bit [i] indicates that Dword [i] of PCIeSlRxData contains valid data.
PCIeSlRxLast	In	Assert this signal in the last beat of a packet to indicate the end of the packet.
PCIeSlRxReady	Out	Indicate that NVMe IP is ready to accept data.
PCIeSlRxUser[87:0]	In	Sideband information for the TLP being transferred. Valid when PCIeSlRxValid is high.
PCIeSlRxValid	In	Assert when PCIe hard IP drives valid data on PCIeSlRxData bus. PCIe hard IP keeps the valid signal asserted during the transfer of packet.

Timing Diagram

Initialization

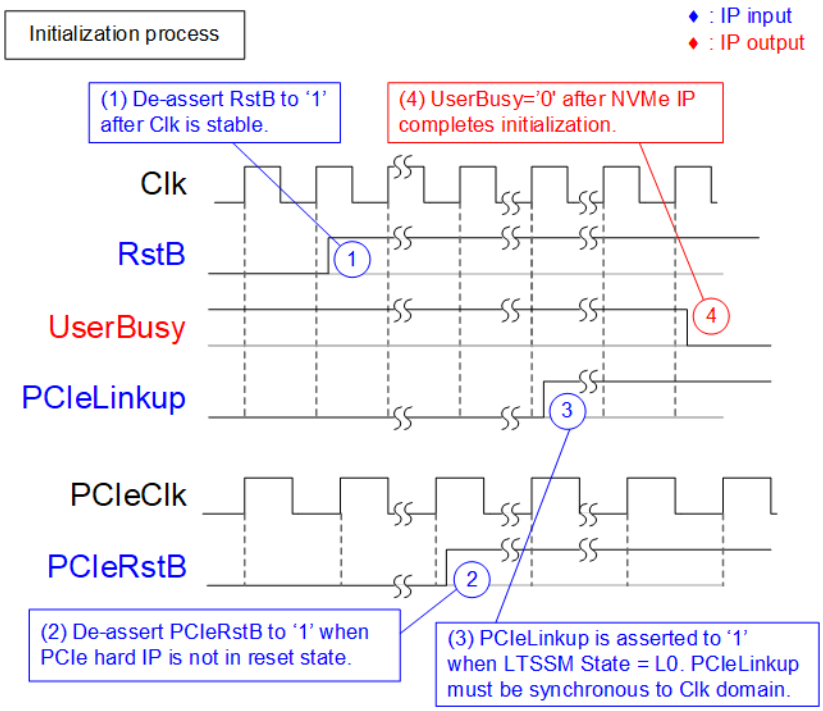

Figure 5: Timing diagram during initialization process

The step of the initialization process is described as follows.

1) Wait until Clk is stable and then de-assert RstB to '1' to start IP initialization.
2) PCIe hard IP de-asserts PCIeRstB to '1' after PCIe reset sequence is done. PCIe hard IP is ready to transfer data with the application layer.
3) Assert PCIeLinkup to '1' after LTSSM state of PCIe hard IP is L0 state. Though LTSSM state is run on PCIeClk, PCIeLinkup must be generated on Clk domain. Asynchronous register must be applied. After that, NVMe IP starts initialization process.
4) UserBusy is de-asserted to '0' after NVMe IP completes initialization process.

After finishing all above steps, NVMe IP is ready to receive the command from user.

Control interface of dgIF typeS

dgIF typeS signals are split into two groups. First group is control interface for sending command with the parameters and monitoring the status. Second group is data interface for transferring data stream in both directions. Figure 6 shows Control interface of dgIF typeS.

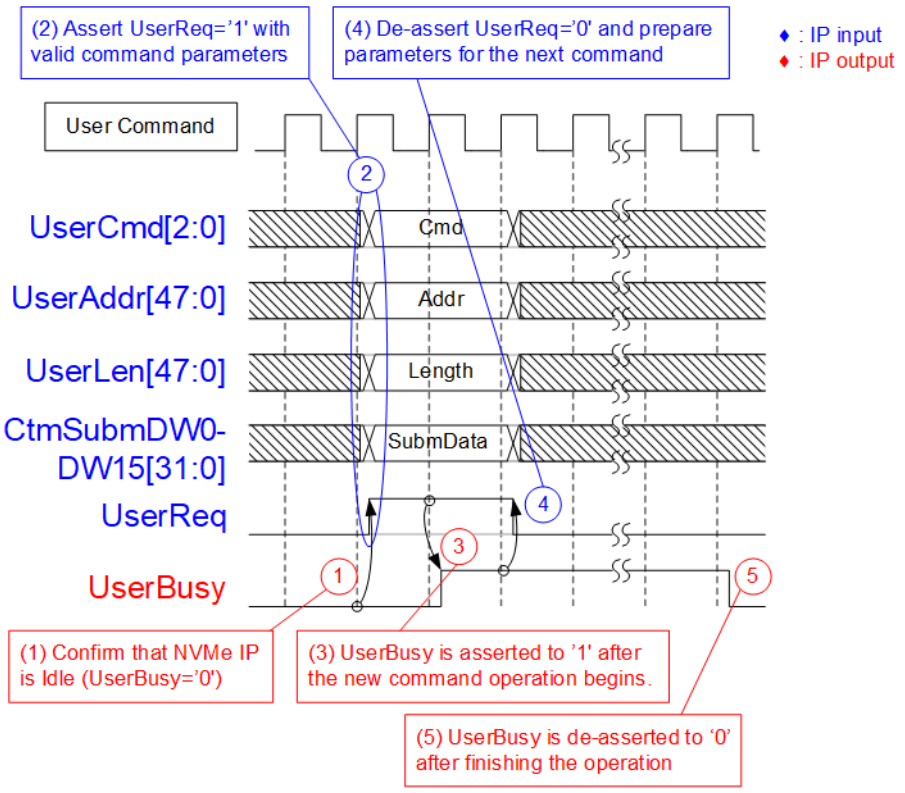

Figure 6: Control Interface of dgIF typeS timing diagram

1) Before sending new command request to the IP, UserBusy must be equal to '0' to confirm that IP is Idle.
2) Command and the parameters such as UserCmd, UserAddr, and UserLen must be valid when asserting UserReq to '1' for sending the new command request.
3) IP asserts UserBusy to '1' after starting the new command operation.
4) After UserBusy is asserted to '1', UserReq is de-asserted to '0' to finish the current request. New parameters for the next command could be prepared on the bus. UserReq for the new command must not be asserted to '1' until the current command operation is finished.
5) UserBusy is de-asserted to '0' after the command operation is completed. New command request can be asserted.

Note: The number of parameters using in each command are different, described more details as follows.

- *Write and Read command: UserCmd, UserAddr, and UserLen.*
- *SMART and Flush command: UserCmd and CtmSubmDW0-DW15.*
- *Identify and Shudown command: UserCmd.*

Data interface of dgIF typeS

Data interface of dgIF typeS is applied for transferring data stream when operating Write command or Read command. The interface is compatible to general FIFO interface. Figure 7 shows the data interface of dgIF typeS when transferring Write data to the IP in Write command. 16-bit FIFO read data counter (UserFifoRdCnt) shows total amount of data stored. If the amount of data is enough, 512-byte data (16x256-bit) is transferred.

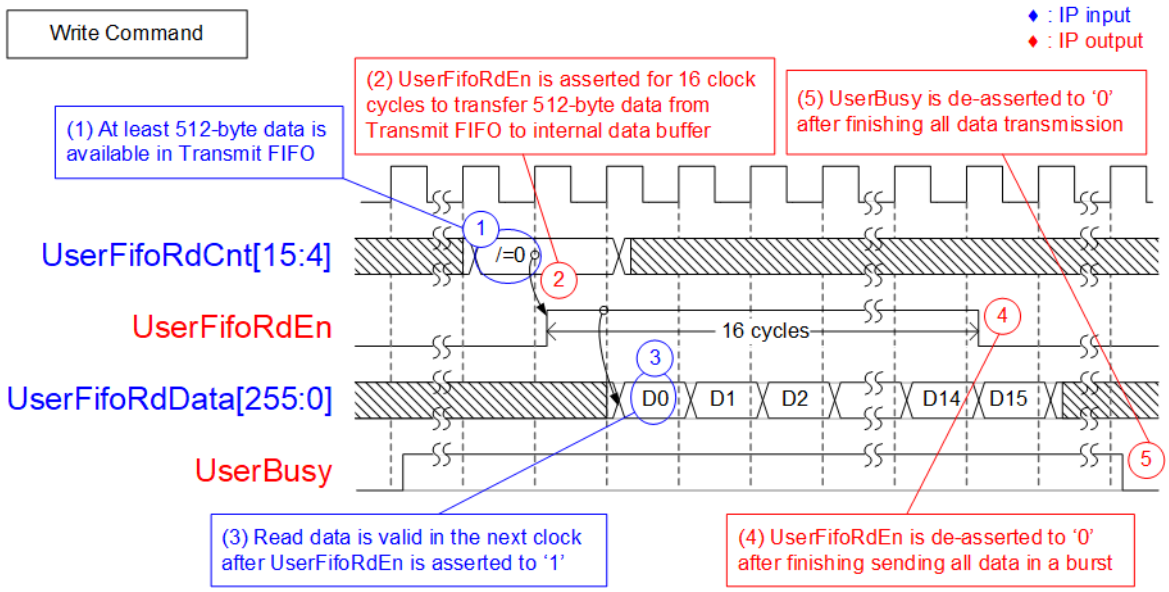

Figure 7: Transmit FIFO Interface for Write command

In Write command, data is read from Transmit FIFO until total data are transferred completely. The details to transfer data are described as follows.

1) Before starting a new burst transfer, UserFifoRdCnt[15:4] is monitored. The IP waits until at least 512-byte data is available in Transmit FIFO (UserFifoRdCnt[15:4] is not equal to 0).
2) The IP asserts UserFifoRdEn to '1' for 16 clock cycles to read 512-byte data from Transmit FIFO.
3) UserFifoRdData is valid in the next clock cycle after asserting UserFifoRdEn to '1'. 16 data are continuously transferred.
4) UserFifoRdEn is de-asserted to '0' after reading the 16th data (D15). Repeat step 1) – 4) to transfer the next 512-byte until total data size is equal to the transfer length, set by the user.
5) After total data is completely transferred, UserBusy is de-asserted to '0'.

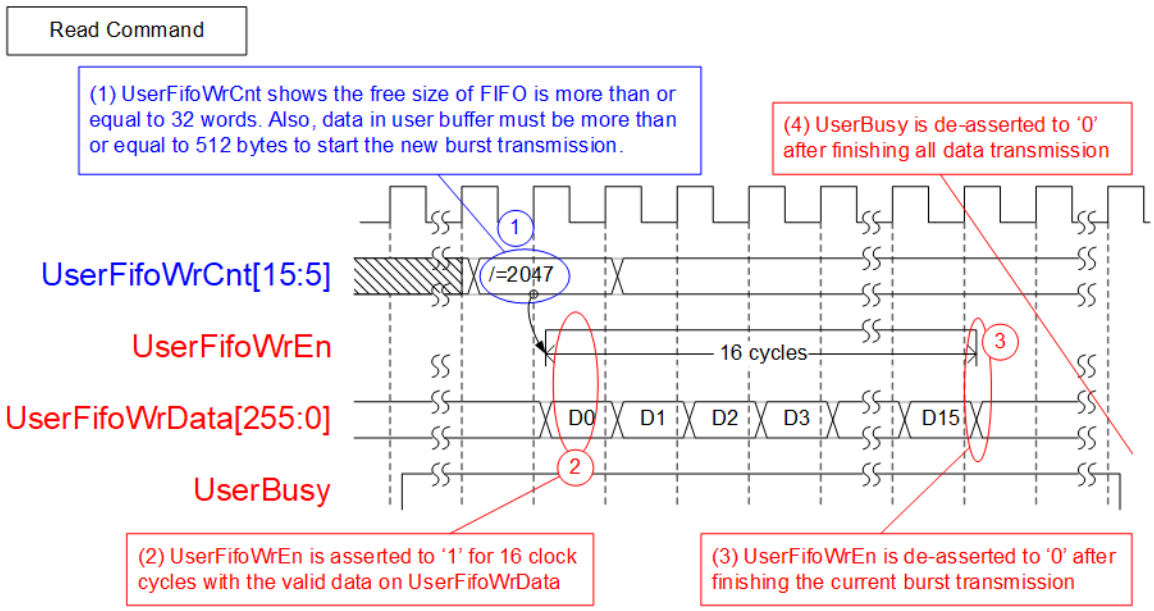

Figure 8: Receive FIFO Interface for Read command

In Read command, data is transferred from SSD to Receive FIFO until total data are completely transferred. The details to transfer a burst of data are described as follows.

1) Before starting the new burst transmission, UserFifoWrCnt[15:5] is monitored. The IP waits until the free space of Receive FIFO is enough (UserFifoWrCnt[15:5] is not equal to all 1 or 2047). After received data from the SSD is more than or equal to 512 bytes, the new burst transmission begins.
2) The IP asserts UserFifoWrEn to '1' for 16 clock cycles to transfer 512-byte data from the data buffer to user logic.
3) After finishing transferring 512-byte data, UserFifoWrEn is de-asserted to '0'. Repeat step 1) – 3) to transfer the next 512-byte data until total data size is equal to the transfer length, set by the user.
4) After total data is completely transferred, UserBusy is de-asserted to '0'.

IdenCtrl/IdenName

It is recommened to send Identify command to the IP as the first command after system boots up. This command updates the necessary information of SSD, i.e., total capacity (LBASize) and LBA unit size (LBAMode). The SSD information is applied to be the limitation of the input parameters when operating Write and Read command, described as follows.

1) The sum of the address (UserAddr) and transfer length (UserLen), inputs of Write and Read command, must not be more than total capacity (LBASize) of the SSD.
2) If LBAMode of the SSD is equal to '1' (LBA unit size is 4 Kbytes), the three lower bit (bit[2:0]) of UserAddr and UserLen must be always equal to '0' to align 4-Kbyte unit.

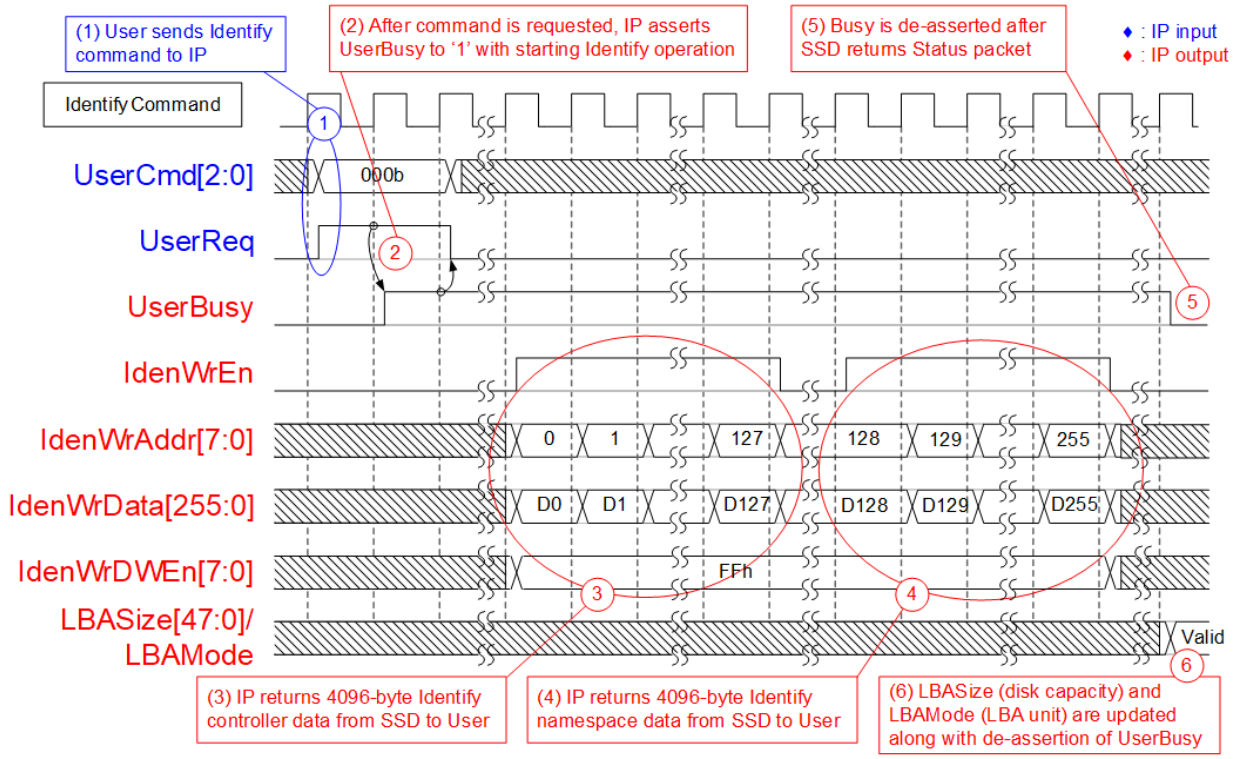

Figure 9: Identify command timing diagram

The details when running Identify command are shown as follows.

1) Send Identify command to the IP (UserCmd=000b and UserReq='1').
2) The IP asserts UserBusy to '1' after running Identify command.
3) 4096-byte Identify controller data is returned to user. IdenWrAddr is equal to 0-127 with asserting IdenWrEn. Also, IdenWrData and IdenWrDWEn are valid at the same clock as IdenWrEn='1'.
4) 4096-byte Identify namespace data is returned. IdenWrAddr is equal to 128-255. IdenWrAddr[7] can be applied to check data type which is Identify controller data or Identify namespace data.
5) UserBusy is de-asserted to '0' after finishing the Identify command.
6) LBASize and LBAMode of the SSD are simultaneously updated.

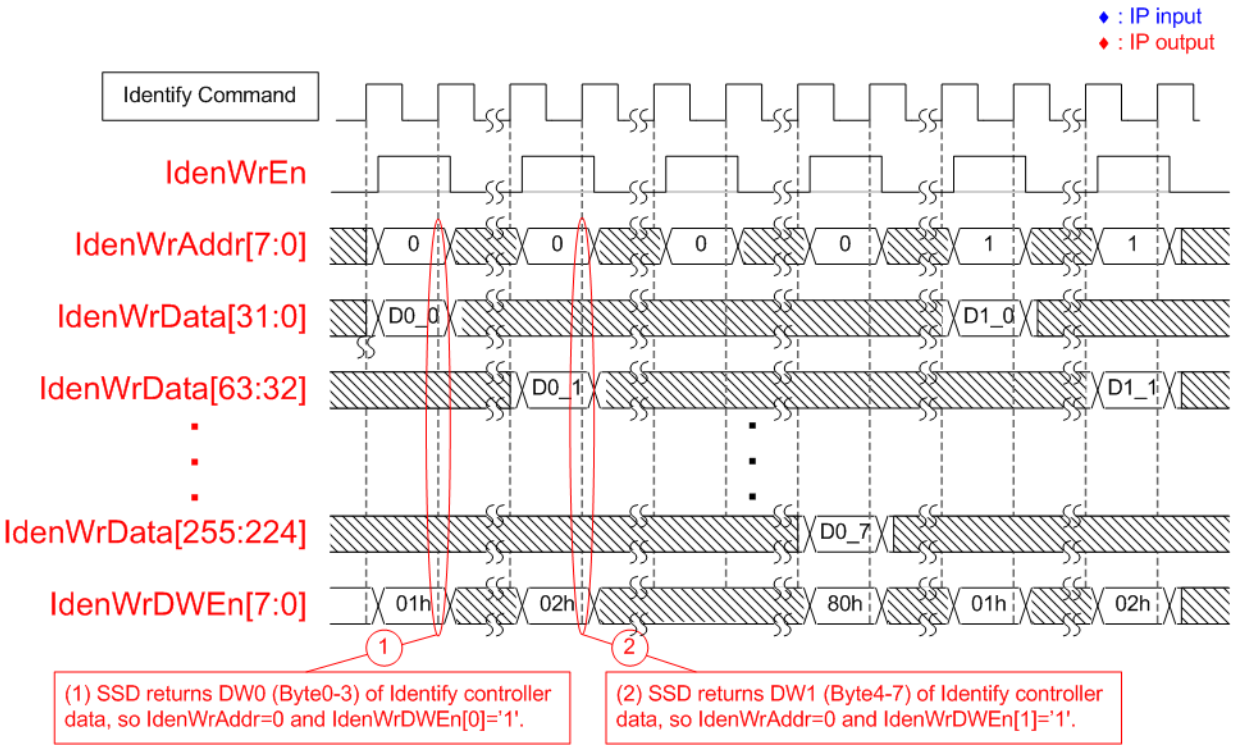

Figure 10: IdenWrDWEn timing diagram

IdenWrDWEn is 8-bit signal to be valid signal of 32-bit data. Some SSDs do not return 4-Kbyte Identify controller data and Identify namespace data continuously, but return only one dword (32-bit) at a time. Therefore, one bit of IdenWrDWEn is asserted to '1' in the write cycle to write 32-bit data, as shown in Figure 10. IdenWrDWEn[0], [1], …, [7] corresponds to IdenWrData[31:0], [63:32], …, [255:224], respectively.

Shutdown

Shutdown command is recommended to send as the last command before the system is powered down. When Shutdown command is issued, SSD flushes the data from the internal cache to flash memory. After Shutdown command is done, NVMe IP and SSD are inactive until the system is powered down. If the SSD is powered down without Shutdown command, the total count of unsafe shutdowns (returned data of SMART command) is increased.

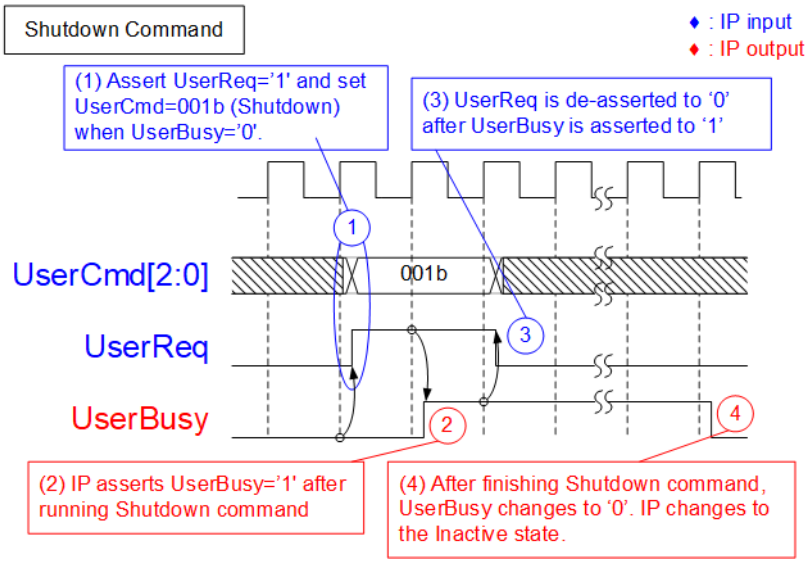

Figure 11: Shutdown command timing diagram

The details when running Shutdown command are shown as follows.

1) Before sending the command request, the IP must be in the Idle state (UserBusy='0'). To send Shutdown command, user asserts UserReq to '1' with UserCmd=001b.
2) UserBusy is asserted to '1' after NVMe IP runs Shutdown command.
3) UserReq is de-asserted to '0' to clear the current request after UserBusy is asserted to '1'.
4) UserBusy is de-asserted to '0' when the SSD is completely shut down. After that, the IP does not receive any command requested from user.

SMART

SMART command is the command to check the SSD health. After sending SMART command, 512-byte health information is returned from the SSD. SMART command loads the parameters from CtmSubmDW0-DW15 signals on Custom command interface. User sets 16-dword data as constant value for SMART command before asserting UserReq. After that, the SMART data is returned via CtmRAM port as shown in Figure 12.

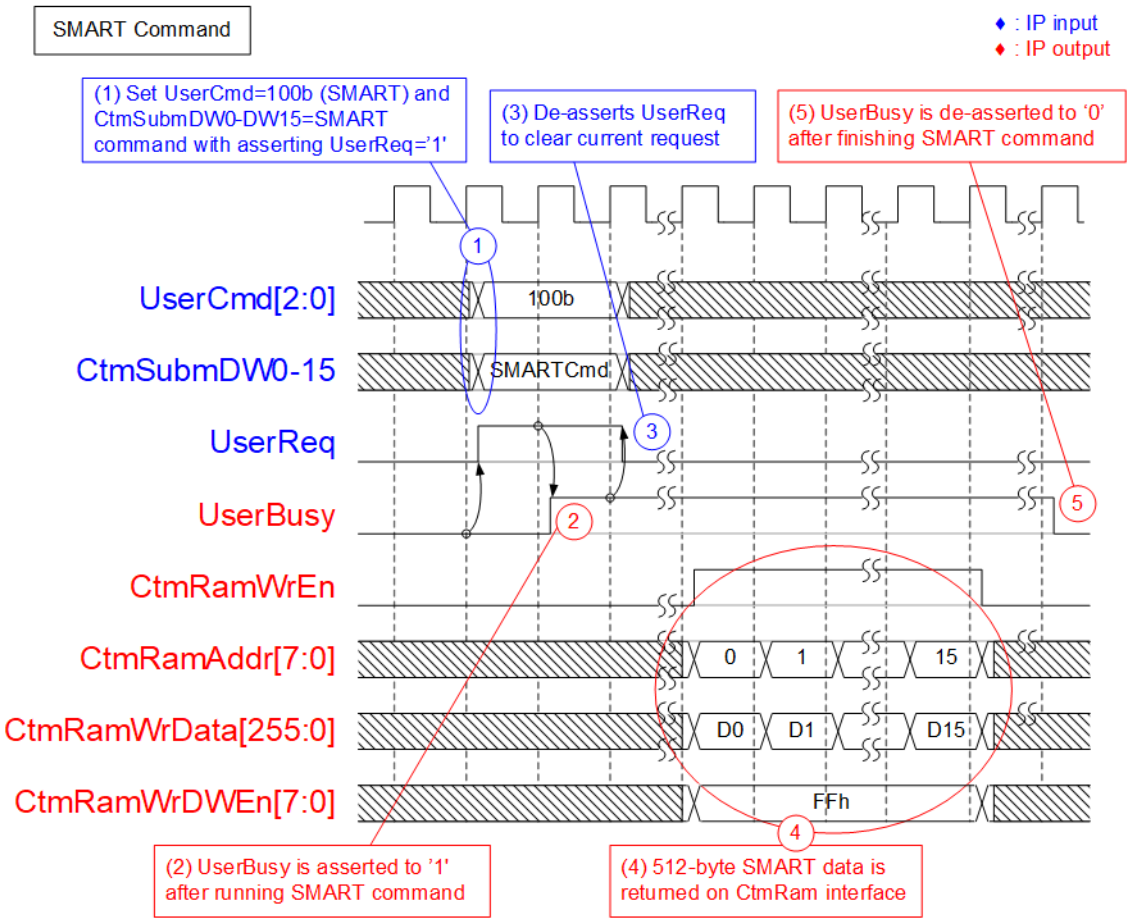

Figure 12: SMART command timing diagram

The details when running SMART command are shown as follows.

1) Before sending the command request, the IP must be in the Idle state (UserBusy='0').
   All input parameters must be stable when UserReq is asserted to '1' for sending the request.
   CtmSubmDW0-DW15 is set as constant value by following value for SMART command.
   CtmSubmDW0 = 0x0000_0002
   CtmSubmDW1 = 0xFFFF_FFFF
   CtmSubmDW2 – CtmSubmDW5 = 0x0000_0000
   CtmSubmDW6 = 0x2000_0000
   CtmSubmDW7 – CtmSubmDW9 = 0x0000_0000
   CtmSubmDW10 = 0x007F_0002
   CtmSubmDW11 – CtmSubmDW15 = 0x0000_0000
2) Assert UserBusy to '1' after NVMe IP runs SMART command.
3) UserReq is de-asserted to '0' to clear the current request. Next, user logic can change the input parameters for the next command request.
4) 512-byte SMART data is returned on CtmRamWrData signal with asserting CtmRamWrEn to '1'. CtmRamAddr is equal to 0-15 to be data index of 512-byte data. When CtmRamAddr=0, byte0-31 of SMART data is valid on CtmRamWrData. CtmRamWrDWEn is dword enable for each 32-bit CtmRamWrData. If CtmRamWrDWEn=FFh, all 256-bit CtmRamWrData are valid.
5) UserBusy is de-asserted to '0' when finishing SMART command.

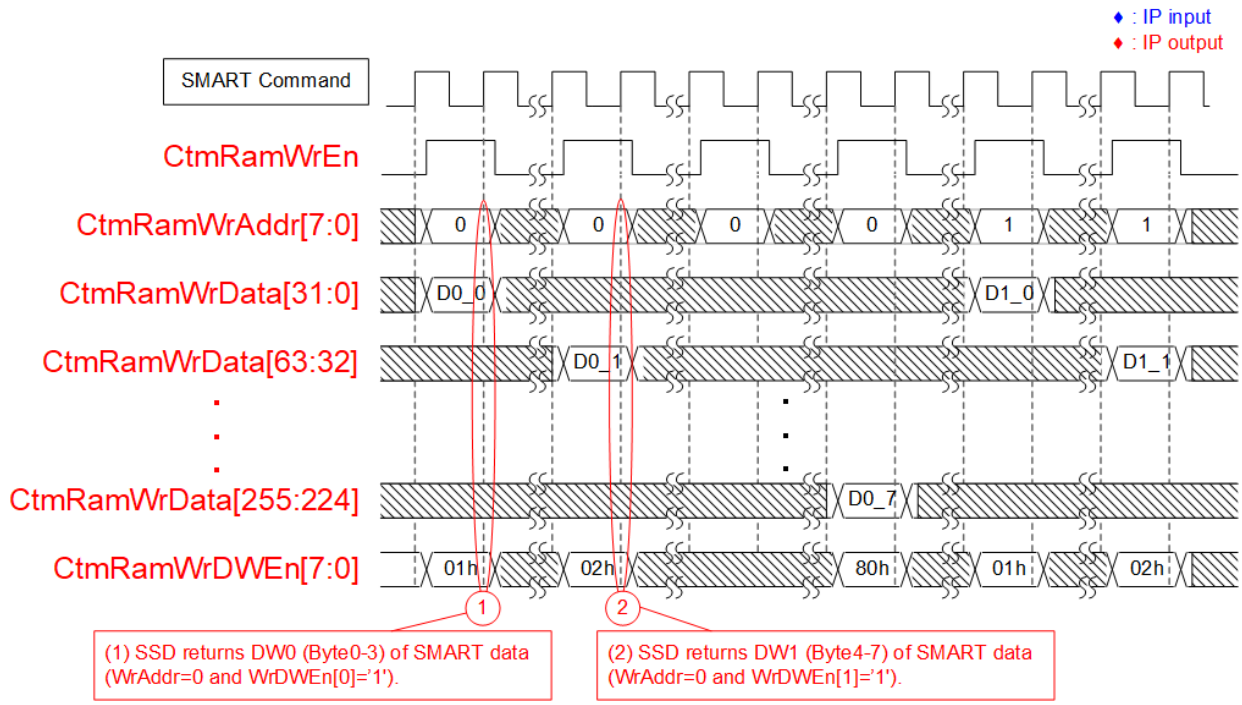

Figure 13: CtmRamWrDWEn timing diagram

Similar to Identify command, some SSDs do not return 512-byte data continuously but return only one dword (32-bit) at a time. Therefore, one bit of CtmRamWrDWEn is asserted to '1' in the write cycle to be the valid signal of 32-bit CtmRamWrData. CtmRamWrDWEn[0], [1], …, [7] corresponds to CtmRamWrData[31:0], [63:32], …, [255:224], respectively.

Flush

Most SSDs accelerate write performance by storing write data to cache before flushing to the flash memory by the SSD controller. If power is down unexpectedly, the data in the cache may be lost and not stored to the flash memory. Flush command is the command to force the SSD controller to flush data from the cache. After sending Flush command, all data in previous Write command can be guaranteed.

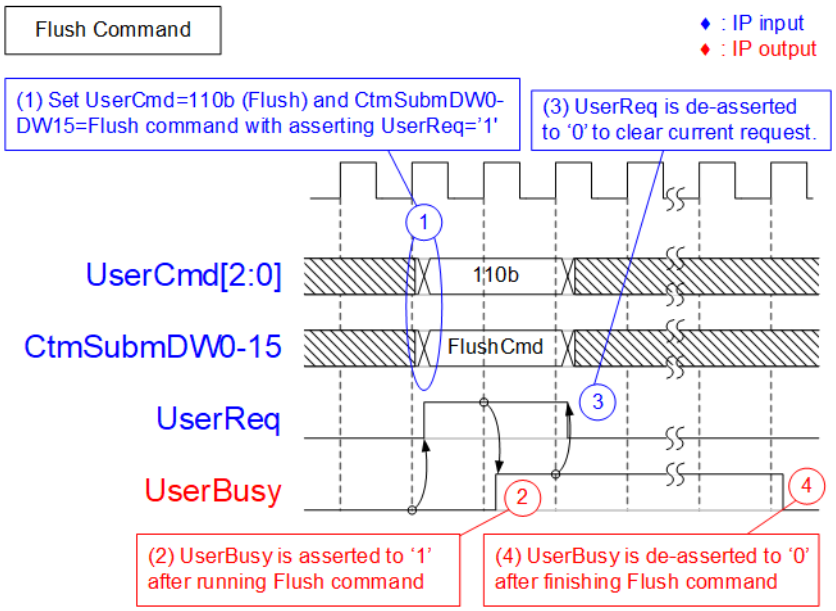

Figure 14: Flush command timing diagram

The details for running Flush command are shown as follows.

1) Before sending the command request, the IP must be in the Idle state (UserBusy='0').
   All input parameters must be stable when UserReq is asserted to '1' for sending the request.
   CtmSubmDW0-DW15 is set as constant value by following value for Flush command.
   CtmSubmDW0 = 0x0000_0000
   CtmSubmDW1 = 0x0000_0001
   CtmSubmDW2 – CtmSubmDW15 = 0x0000_0000
2) UserBusy is asserted to '1' after NVMe IP runs Flush command.
3) UserReq is de-asserted to '0' to clear the current request. Next, user logic can change the input parameters for the next command request.
4) UserBusy is de-asserted to '0' when Flush command is done.

Error

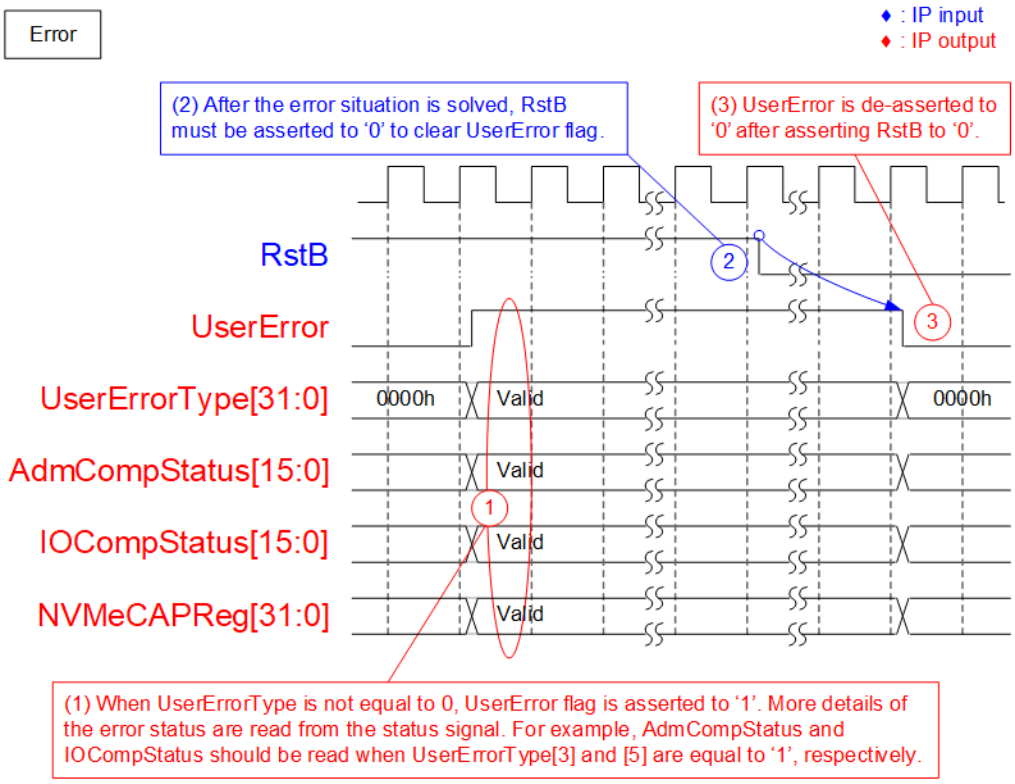

Figure 15: Error flag timing diagram

When the error is found while running initialization process or operating some commands, UserError flag is asserted to '1'. UserErrorType is read to check the error type. NVMeCAPReg, AdmCompStatus, and IOCompStatus are valid for monitoring error details after UserError is asserted to '1'.

When the error is found while running initialization process, it is recommended to read NVMeCAPReg to check capability of NVMe SSD. When the error is found while operating the command, it is recommended to read AdmCompStatus or IOCompStatus.

- When bit[3] of UserErrorType is asserted, read AdmCompStatus to check more details.
- When bit[5] of UserErrorType is asserted, read IPCompStatus to check more details.

UserError flag is cleared by RstB signal only. After the failure is solved, RstB is asserted to '0' to clear the error flag.

Verification Methods

The NVMe IP Core functionality was verified by simulation and also proved on real board design by using VCK190 evaluation board and Alveo-U50 Accelerator Card.

Recommended Design Experience

Experience design engineers with a knowledge of Vivado Tools should easily integrate this IP into their design.

Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design Gateway Co., Ltd. for pricing and additional information about this product using the contact information on the front page of this datasheet.

Revision History

Revision	Date	Description
1.1	21-Jul-2022	Support VCK190 board
1.0	13-Sep-2021	Initial Release