NVMe IP Core for PLDA PCIe

February 13, 2019 Product Specification Rev1.3

Design Gateway Co.,Ltd
54 BB Building 14th Fl., Room No.1402 Sukhumvit
21 Rd. (Asoke), Klongtoey-Nua, Wattana,
Bangkok 10110
Phone: 66(0)2-261-2277
Fax: 66(0)2-261-2290
E-mail: ip-sales@design-gateway.com
URL: www.design-gateway.com

Core Facts

Provided with Core	
Documentation	Reference Design Manual Demo Instruction Manual
Design File Formats	Encrypted Netlist
Constraints Files	Constraint file example in reference design project
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KCU105, ZCU102
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- Implement application layer to access NVMe PCIe SSD without CPU usage
- Simple user interface by dgIF typeS
- Support to connect NVMe SSD directly or connect NVMe SSD through one PCIe switch
- Co-operate with PCIe XpressRICH3 from PLDA
- Include 256 Kbyte RAM to be data buffer
- Support three commands, i.e. IDENTIFY, WRITE, and READ
- 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)
- Reference design with AB16-PCIeXOVR adapter board for KCU105 board or AB17-M2FMC adapter board for ZCU102 board

Table 1: Example Implementation Statistics for Ultrascale/Ultrascale+ device (PCIe Gen3)

Family	Example Device	Fmax (MHz)	CLB Regs	CLB LUTs	CLB	IOB	BRAMTile[1]	PLL	GTX	Design Tools
Kintex-Ultrascale	XCKU040FFVA1156-2E	384	4717	3018	820	-	61	-	-	Vivado2017.4
Zynq-Ultrascale+	XCZU9EG-FFVB1156-2	384	4717	2999	785	-	61	-	-	Vivado2017.4

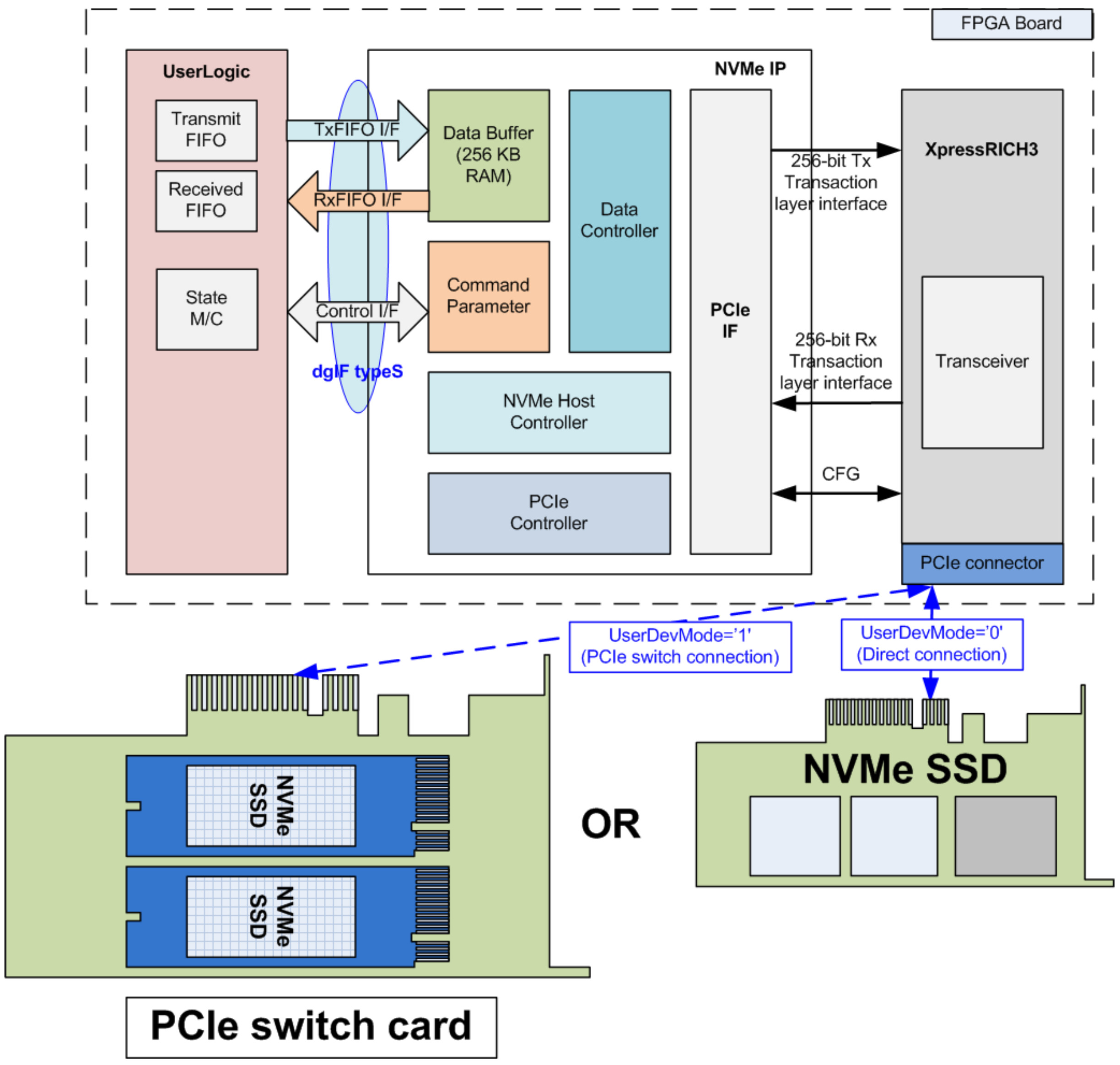

Figure 1: NVMe IP Block Diagram

Applications

NVMe IP Core integrated with PCI Express XpressRICH3 and PHY from PLDA is an ideal to access NVMe SSD without CPU and DDR. However, NVMe-IP needs 256 Kbyte buffer implemented by BlockRAM to store data which is transfered between user logic and NVMe SSD. This solution is recommended for the application which requires high capacity storage at ultra high-speed performance. Small size system can be designed by using M.2 SSD storage.

General Description

NVMe IP implements as host controller to access NVMe SSD following NVMe standard. Physical interface of NVMe SSD is PCIe, so the hardware of lower layer is implemented by XpressRICH3 and PHY from PLDA.

NVMe IP supports to connect to NVMe SSD directly or connect SSD through one PCIe switch. Only one SSD on PCIe switch card could be active, selected by UserDevNo signal. To change active device on PCIe switch card, the system needs to restart by power off/on system.

NVMe IP supports three NVMe commands, i.e. Identify, Write, and Read command. 256 Kbyte RAM is included in NVMe IP to be data buffer between user logic and NVMe SSD when operating Write and Read command. By using big buffer size, the system achieves good performace to write and read SSD.

The user interface of NVMe IP is simply designed by using dgIF typeS. dgIF typeS consists of two interfaces, i.e. command interface and data interface. Input parameters of Write and Read command (received through dgIF typeS) are start address and transfer length. Data stream of Write and Read command are transferred through TxFIFO and RxFIFO interface of dgIF typeS.

Clock frequency of user logic must be more than or equal to PCIe clock frequency (125 MHz for PCIe Gen3). Error signal will be asserted with the error status if IP detects the abnormal condition during packet transferring.

The reference design on Xilinx evaluation boards are available to evaluate before purchasing.

Functional Description

Figure 2 shows operation sequence of NVMe IP after IP reset is released.

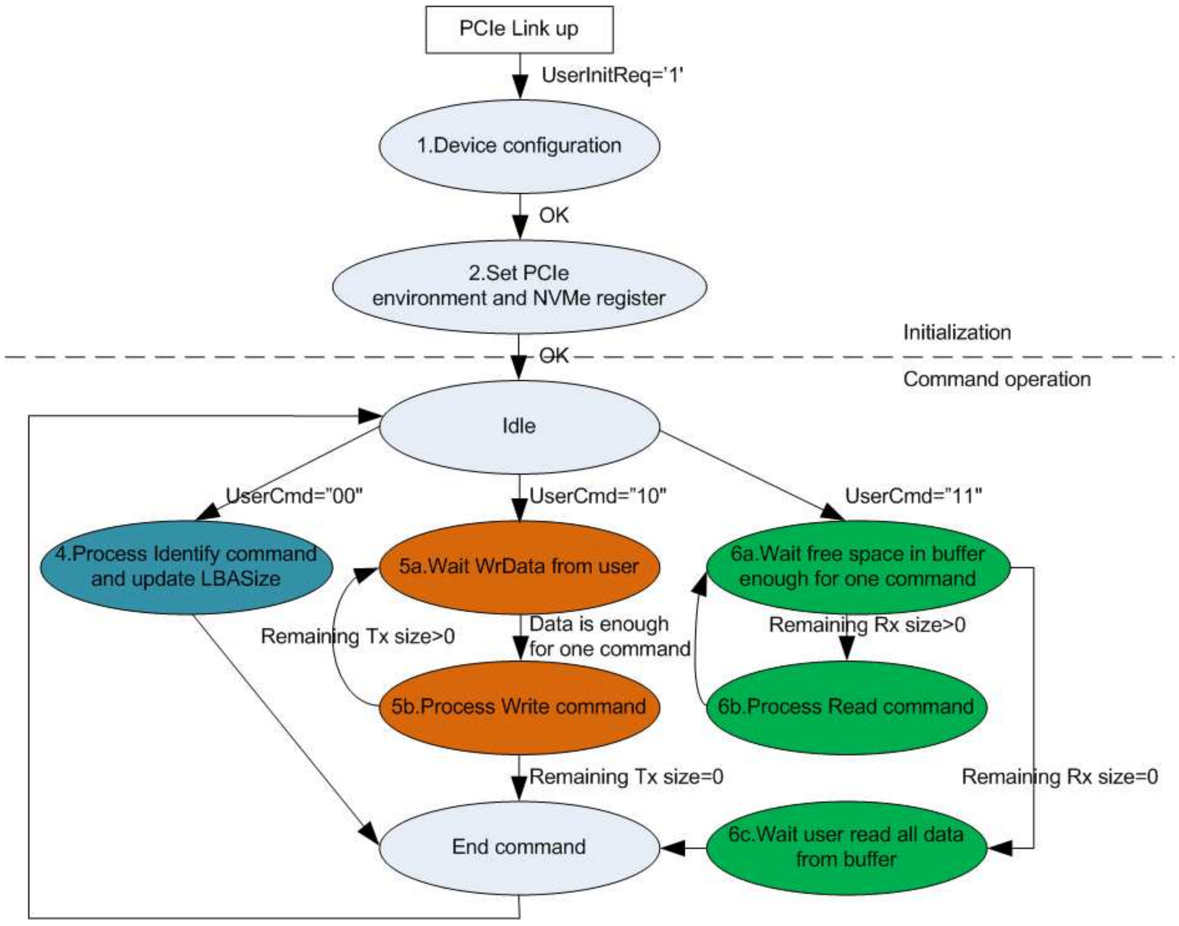

Figure 2: NVMe IP Operation Flow

1) After PCIe link up is detected, IP waits user to set device mode and asserts UserInitReq='1'. After that, IP sets PCIe configuration following device mode.
2) IP sets up PCIe environment for NVMe operation and sets up NVMe parameters in SSD. After complete initialization process, IP is in idle state to wait new command request from user.
3) The 1st command from user must be Identify command to update LBASize signal (disk capacity).
4) In case of Write command,
   - IP waits until user sends write data to NVMe IP is much enough for one command (Transfer size of one command in NVMe IP is 128 Kbyte or less).
   - IP sends Write command to NVMe SSD.
   - IP waits status from SSD to confirm that all data in one command have been received completely. Remaining transfer size is decreased by the transmit size.
   - If remaining transfer size is not zero, IP will continue to check write data size from user in NVMe IP. If data size is much enough, the next command will be sent.
   - If remaining transfer size is zero, IP will go back to Idle state.

5) In case of Read command,
    - IP checks free space of data buffer in NVMe IP.
    - If free space is much enough for one command, IP will send Read command to SSD.
    - IP waits until received size is equal to transfer size in one command. After that, the data is ready for user to read. Remaining transfer size is decreased by the received size.
    - If remaining transfer size is not zero, IP will check free space size for sending next command.
    - If remaining transfer size is zero, IP will go back to Idle state.

From above sequences, NVMe IP consists of three controllers, i.e. PCIe controller, NVMe host controller, and Data controller. After system power-on, PCIe controller sets up PCIe environment for connecting with SSD. Next, NVMe host controller writes and reads NVMe register of SSD following NVMe specification to complete initialization phase.

NVMe host controller controls the packet sequence of each user command following NVMe standard. The input parameters from user are received to create Command packet. Data controller is responsible to create and decode all packets transferring with SSD.

More details of each module in NVMe IP are described as follows.

PCIe

NVMe protocol uses PCIe standard to be physical interface and lower layer protocol, so the intialization sequence and lower layer communication are designed by PCIe Controller.

- **PCIe Controller**

  This module includes state machine to check PCIe device class, to set BAR address, and to enable master mode. The essential parameters to setup PCIe environment are mapped to configuration space area of SSD which can be acccessed through 256 bit transaction layer interface.

  XpressRICH3 needs to setup configuration space through CFG interface. Therefore, PCIe controller includes the logic to intialize configuration space in XpressRICH3.

NVMe

Following NVMe standard, there are two command types, i.e. Admin command and NVM command. Admin command is controlled through Admin submission queues and Admin completion queues. NVM command is controlled through I/O submission queues and I/O completion queues.

In case of Identify/Write/Read command, Submission queue entry has been prepared and status value within Completion queue entry has been monitored by NVMe block to check error status. 256 Kbyte data buffer is used to store data transferring between user logic and SSD when operating Write/Read command. Data output from Identify command is transferred through Identify interface.

The details of each module within NVMe block are described as follows.

- **NVMe Host Controller**

  The sequence of NVMe IP (shown in Figure 2) is controlled by NVMe host controller. The first step is sending Submission queue entry to SSD to start new command operation. If the command requires data transferring, NVMe host controller needs to prepare and send data pointer for using in the command. Data pointer for each command is defined to different interface, i.e. data buffer for Write/Read command and Identify interface for Identify command. The command is completed after Completion queue entry is received from SSD. Busy flag of user interface is de-asserted to '0' when the command is completed.

  Otherwise, the logic of NVMe host controller needs to create multiple Write/Read command when total transfer length from user is more than 128 Kbyte size (Maximum transfer size of one NVM command is 128 Kbyte).

  Busy flag of most command is de-asserted to '0' after SSD returns no error status in Completion queue entry to NVMe IP, except Read command. Busy flag of Read command is de-asserted to '0' after user reads all data from NVMe IP (Data buffer is empty).

- **Command Parameter**

  This module is designed to prepare Submission queue entry for all commands and decode status value of Completion queue entry.

  Submission queue entry size is 16 dwords (1 dword=32-bit). When running Identify, Write, and Read command, Command parameter module creates the value of all 16 dwords of Submission queue entry following Command interface of dgIF typeS.

  Completion queue entry size is 4 dwords. The value of Completion queue entry is monitored by Command parameter for all commands.

- **Data Controller**

  This module is operated when the command requires data transferring such as Write and Read command. Data packet request is created by SSD after receiving new command. Data controller decodes the address in SSD request packet to check data type such as Write/Read data, Identify command data, and pointer list. Each data type is mapped to different address. When SSD sends read data request, the logic inside Data controller selects returned data following the address in the request.

- **PCIe IF**

  This module includes FIFO to control data flow between NVMe IP and XpressRICH3. It includes RAM to be data buffer when another side is not ready to receive the packet.

- **Data Buffer**

  256 Kbyte simple dual port RAM is implemented by BlockRAM. This RAM is used to be data buffer transferring between UserLogic and SSD in Write and Read command.

User Logic

To set user parameters such as command, address, and transfer size, simple logic with small state machine could be designed in User Logic. Data stream for Write/Read command are connected to FIFO.

XpressRICH3 for Xilinx

XpressRICH3 is provided by PLDA. This IP is PCIe soft IP core. More details are described in following website.
https://www.plda.com/node/403

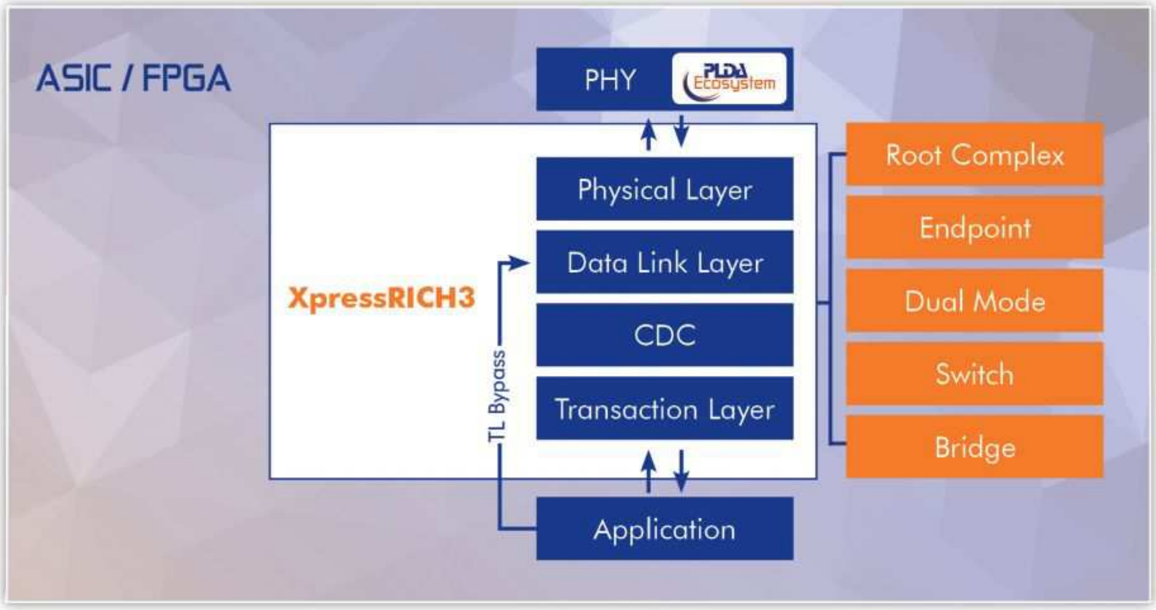

Figure 3: XpressRICH3 CORE

Core I/O Signals

Descriptions of all signal I/O are provided in Table 2.

Table 2: Core I/O Signals

Signal	Dir	Description
Control I/F of dgIF typeS (Synchronous to Clk)		
RstB	In	Synchronous reset signal. Active low. Release to ‘1’ when Clk signal is stable.
Clk	In	System clock for running NVMe IP. The recommend frequency of Clk should be more than or equal to 125MHz for PCIe Gen3 (75MHz for PCIe Gen2).
UserDevMode	In	“0”: direct connection (connect SSD directly), “1’: PCIe switch connection (connect SSD through PCIe switch). This signal must not change while IP is in initialization process.
UserDevNo[4:0]	In	Input to indicate target device number. This signal must not change while IP is in initialization process.
UserInitReq	In	Assert to ‘1’ for one clock cycle to start device configuration. This signal is asserted to ‘1’ only one time after UserInitReady is asserted to ‘1’.
UserInitReady	Out	Assert to ‘1’ when IP is ready to start device configuration.
UserCmd[1:0]	In	User Command (“00”: Identify command, “10”: Write SSD, ”11”: Read SSD).
UserAddr[47:0]	In	Start address to write/read SSD in 512-byte unit *Note: From SSD characteristic, it is recommended to set bit[2:0]=”000” to align 4 Kbyte size (SSD page size). Write/Read performance in most SSD are reduced when start addrss is not aligned to page size.*
UserLen[47:0]	In	Total transfer size in the request in 512-byte unit. Valid from 1 to (LBASize-UserAddr).
UserReq	In	Assert to ‘1’ to request the new command. Can be asserted when the IP is Idle (UserBusy=’0’). Command parameter (UserCmd, UserAddr, UserLen) must be valid and stable during UserReq=’1’.
UserBusy	Out	Assert to ‘1’ when IP is busy. UserReq will not be allowed to ‘1’ if IP is still busy status.
LBASize[47:0]	Out	Total capacity of SSD in 512-byte unit. Default value is 0. This value is valid after user sends Identify command.
UserError	Out	Error flag. Assert when UserErrorType is not equal to 0. The flag can be cleared by asserting RstB signal.
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 returned 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 returned 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. [6] – Error when Completion TLP packet size is not correct. [7] – Error when XpressRICH3 detects ECC in the TLP. [8] – Error from Unsupported Request (UR) flag in Completion TLP packet.

Signal	Dir	Description
Control I/F of dgIF typeS (Synchronous to Clk)		
UserErrorType[31:0]	Out	[9] – Error from Completer Abort (CA) flag in Completion TLP packet. [10] – Error when Length[1:0] in Memory Write Request TLP packet is not equal to 0 (not aligned to 128-bit unit). [11] - Error when Address[3:2] in Memory Write or Memory Read Request TLP packet is not equal to 0 (not aligned to 128-bit unit). [12] – Error when XpressRICH3 detects the TLP payload size does not match with information in the header. [13] – Error when XpressRICH3 detects an uncorrectable error. [31:14] - Reserved Note: Timeout period of bit[2]/[4] is set from TimeOutSet input.
Data I/F of dgIF typeS (Synchronous to Clk)		
UserFifoWrCnt[15:0]	In	Write data counter of Received FIFO. Used for checking full status. If total FIFO size is less than 16-bit, please fill '1' to upper bit.
UserFifoWrEn	Out	Assert to '1' to write data valid of Received FIFO.
UserFifoWrData[255:0]	Out	Write data bus of Received FIFO. Synchronous to UserFifoWrEn.
UserFifoRdCnt[15:0]	In	Read data counter of Transmit FIFO. Used for checking total numbers of data stored in FIFO. If FIFO size signal is less than 16-bit, please fill '0' to upper bit.
UserFifoEmpty	In	FIFO empty flag of Transmit FIFO to check avaiable data status. This signal is not used.
UserFifoRdEn	Out	Assert to '1' to read data from Transmit FIFO.
UserFifoRdData[255:0]	In	Read data returned from Transmit FIFO. Valid in the next clock cycle after UserFifoRdEn is asserted.
NVMe IP Interface		
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).
PCIeLinkup	In	Asserted to '1' when LTSSM state of XpressRICH3 is in L0 State.
AdmCompStatus[15:0]	Out	[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	[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	Some parts of NVMe capability register output from SSD. [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
IdenWrEn	Out	Assert to '1' when IdenWrAddr and IdenWrData are valid on the bus.
IdenWrAddr[7:0]	Out	Index of IdenWrData in 256-bit unit. Synchronous to IdenCtrlWrEn. When transferring 4Kbyte Identify Controller data, IdenWrAddr is equal to 0x00 – 0x7F. 0x00 is 1st Identify Controller data (byte0 – byte31) and 0x7F is 127th Identfy Controller data (byte4064 - byte4095). When transferring 4Kbyte Identify Namespace data, IdenWrAddr is equal to 0x80 – 0xFF. 0x80 is 1st Identify Namespace data (byte0 – byte31) and 0xFF is 127th Identfy Namespace data (byte4064 - byte4095).
IdenWrData[255:0]	Out	4Kbyte Identify controller data or Identify Namespace data from Identify command. Synchronous to IdenWrEn.

Signal	Dir	Description
XpressRICH3 from PLDA		
Configuration Interface		
PCIeCfgAddr[11:0]	Out	Read/Write Address.
PCIeCfgWrite	Out	Command ('1': Write operation, '0': Read operation).
PCIeCfgEn	Out	Configuration access enable. Asserted to request write or read operation
PCIeCfgReady	In	Read/Write operation complete. Asserted for 1 cycle when operation completes.
PCIeCfgWrData[31:0]	Out	Write data to set the configuration registers.
PCIeCfgStrb [3:0]	Out	Byte enable for write data (bit[0] corresponds to PCIeCfgWrData[7:0], and so on).
PCIeCfgRdData[31:0]	in	Read value from the configuration register. This signal is valid during read access operation when PCIeCfgReady is asserted.
PCIe Transmit Interface		
PCIeTxData[255:0]	Out	Transmit data to XpressRICH3.
PCIeTxValid[7:0]	Out	Transmit data valid in DWORD unit. Indicates that which DWORDs in PCIeTxData are valid. Bit[7] indicates that PCIeTxData[255:224] is valid, while bit[0] indicates that PCIeTxData[31:0] is valid. ('0': Not valid, '1': Valid).
PCIeTxSOP	Out	Transmit Start-of-Packet.
PCIeTxEOP	Out	Transmit End-of-Packet.
PCIeTxWait	In	Indicates that PCI Express XpressRICH3 is not ready to accept data.The simultaneous assertion of PCIeTxValid and de-assertion of PCIeTxWait marks the successful transfer of one data beat on PCIeTxData
PCIeTxErr	Out	Reserved.
PCIeTxCred	In	Reserved.
PCIe Receive Interface		
PCIeRxData[255:0]	In	Receive data from XpressRICH3. Valid when PCIeRxValid is asserted.
PCIeRxValid[7:0]	In	Receive data strobe. Determine which data bytes are valid on PCIeRxData. Bit[7] indicates that PCIeRxData[255:224] is valid, while bit[0] indicates that PCIeRxData[31:0] is valid. ('0': Not valid, '1': Valid).
PCIeRxSOP	In	Receive Start-of-Packet.
PCIeRxEOP	In	Receive End-of-Packet.
PCIeRxWait	Out	De-assertd indicates that NVMe IP is ready to accept data on PCIeRxData. The simultaneous assertion of PCIeRxValid and de-assertion of PCIeRxWait marks the successful transfer of one data beat on PCIeRxData.
PCIeRxError[2:0]	In	Bit[0]: Receive ECRC error. Bit[1]: Invalid TLP or TLP Payload size is not correct. Bit[2]: Receive Buffer Uncorrectable Read Error.

Timing Diagram

Initialization

User could select device connection mode to be direct connection or PCIe switch connection by setting UserDevMode value. The sequence of the initialization process is shown in Figure 4 - Figure 5.

1) RstB is released to '1' after Clk is stable.
2) NVMe IP waits until PCIeRstB and PCIeLinkup are asserted to '1' to confirm that XpressRICH3 is in ready status. After PCIeLinkup is asserted to '1', XpressRICH3 is ready to exchange data packets.
3) User waits until UserInitReady is asserted to '1' to confirm that NVMe-IP is ready to initialization.
4) User sets UserDevMode ('0': Direct connection, '1': PCIe switch connection), UserDevNo (for PCIe switch connection only), and UserInitReq='1' to start initialization sequence. UserInitReq is asserted to '1' for one clock cycle. After that, NVMe IP starts initialization process.
5) UserBusy is de-asserted to '0' after NVMe IP completes initialization process.

After complete above sequence, NVMe IP is ready to receive the command from user.

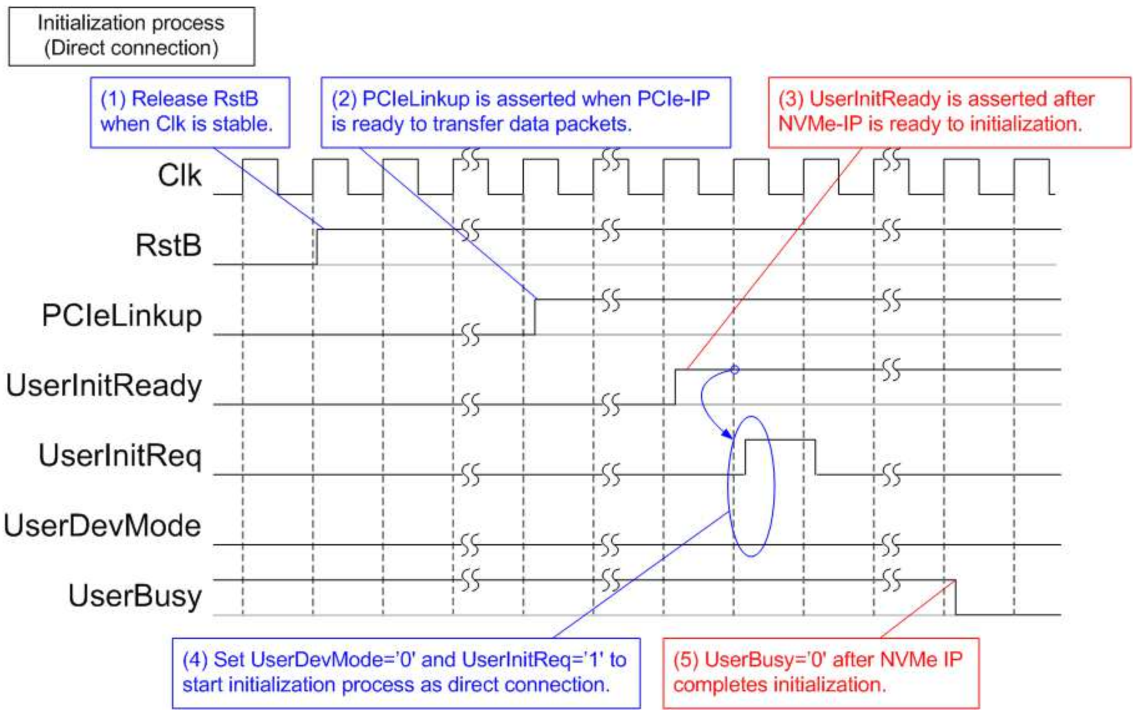
Initialization process (Direct connection)
(1) Release RstB when Clk is stable.
(2) PCIeLinkup is asserted when PCIe-IP is ready to transfer data packets.
(3) UserInitReady is asserted after NVMe-IP is ready to initialization.
Clk
RstB
PCIeLinkup
UserInitReady
UserInitReq
UserDevMode
UserBusy
(4) Set UserDevMode='0' and UserInitReq='1' to start initialization process as direct connection.
(5) UserBusy='0' after NVMe IP completes initialization.

Figure 4: UserBusy after system boot-up in direct connection mode

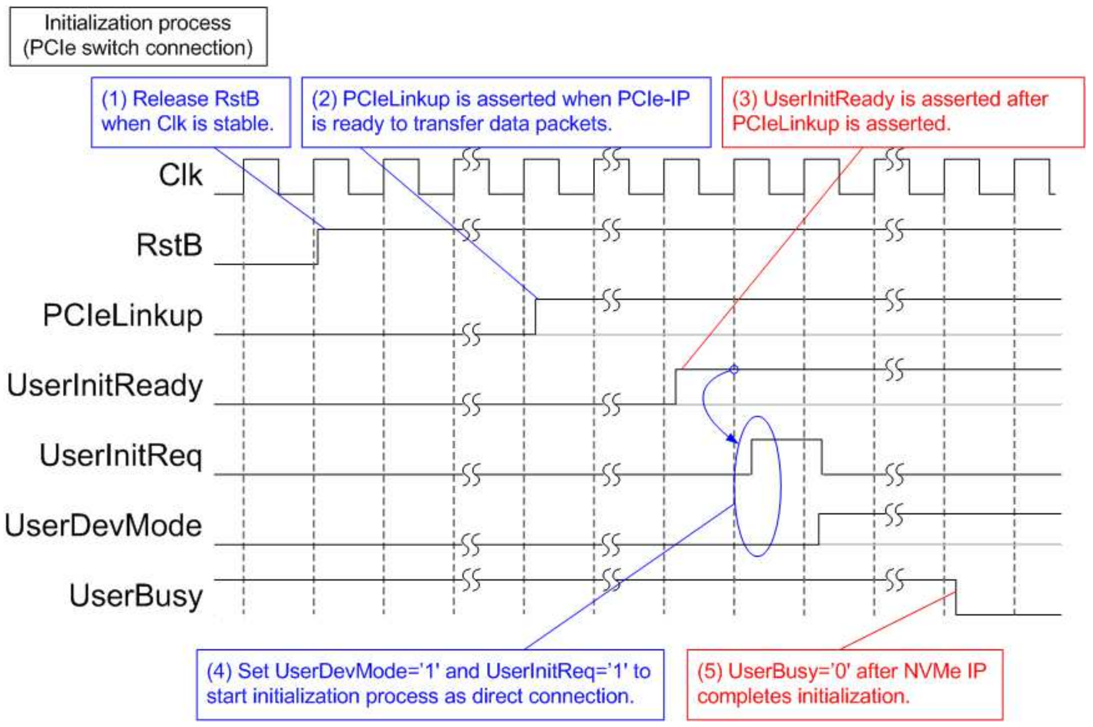
Initialization process (PCIe switch connection)
(1) Release RstB when Clk is stable.
(2) PCIeLinkup is asserted when PCIe-IP is ready to transfer data packets.
(3) UserInitReady is asserted after PCIeLinkup is asserted.
Clk
RstB
PCIeLinkup
UserInitReady
UserInitReq
UserDevMode
UserBusy
(4) Set UserDevMode='1' and UserInitReq='1' to start initialization process as direct connection.
(5) UserBusy='0' after NVMe IP completes initialization.

Figure 5: UserBusy after system boot-up in PCIe switch connection mode

dgIF typeS

dgIF typeS signal is split into two interfaces, i.e. command interface and data interface. Figure 6 shows timing diagram of command interface of dgIF typeS. Before sending new command to the IP, UserBusy must be monitored to confirm that IP is Idle. Command parameters (UserCmd, UserAddr, and UserLen) must be valid and stable during asserting UserReq='1'. After receiving new request from user, UserBusy changes status from '0' to '1'. Next, UserReq could be de-asserted to '0' and user logic can prepare the next command parameter on the bus.

Note: UserAddr and UserLen value are ignored in Identify command.

For data interface, Transmit FIFO is read by NVMe IP for Write command, while Received FIFO is written by NVMe IP for Read command. Timing diagram of data interface is shown in Figure 7 and Figure 8.

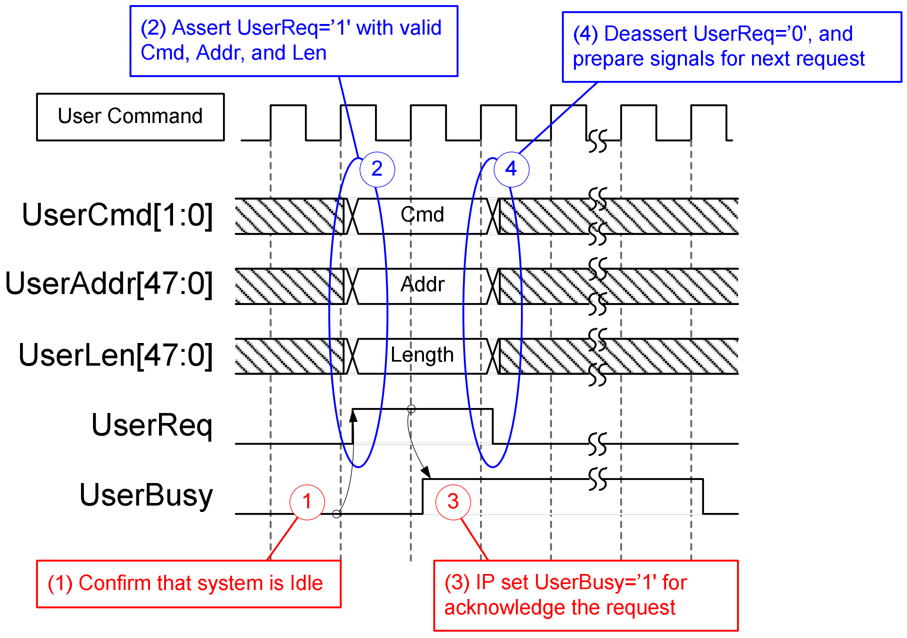
(2) Assert UserReq='1' with valid Cmd, Addr, and Len
(4) Deassert UserReq='0', and prepare signals for next request
User Command
2
4
UserCmd[1:0]
Cmd
UserAddr[47:0]
Addr
UserLen[47:0]
Length
UserReq
UserBusy
1
3
(1) Confirm that system is Idle
(3) IP set UserBusy='1' for acknowledge the request

Figure 6: Command Interface of dgIF typeS Timing diagram

For Write command, data from Transmit FIFO is stored to data buffer in NVMe IP. DMA Engine in NVMe IP monitors UserFifoRdCnt signal until it indicates that data in Transmit FIFO is equal to or more than 512 bytes. After that, UserFifoRdEn is asserted to '1' for 16 clock cycles to read 512-byte data, as shown in Figure 7. Similar to general FIFO timing diagram, UserFifoRdData is valid in the next clock cycle after UserFifoRdEn is asserted to '1';.

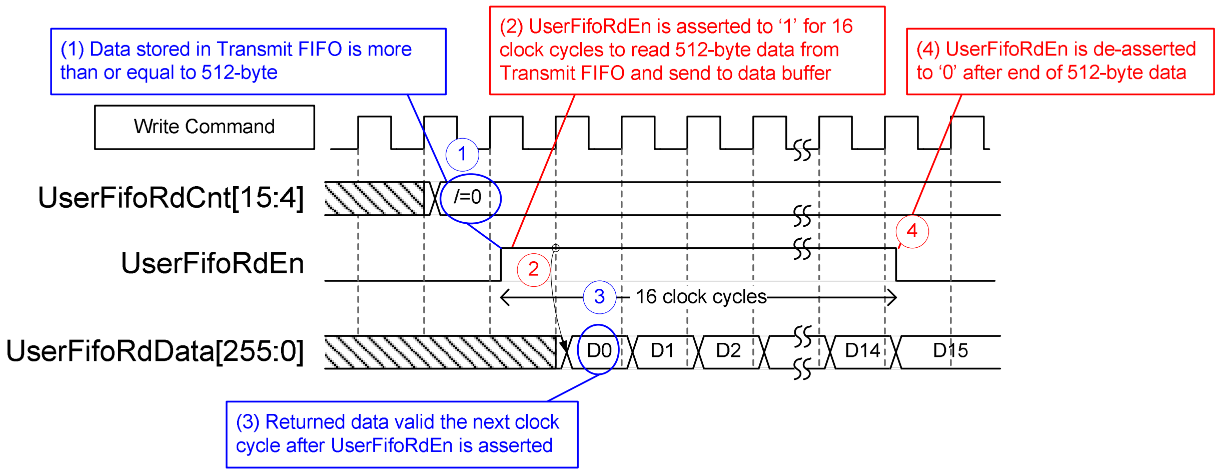
(1) Data stored in Transmit FIFO is more than or equal to 512-byte
(2) UserFifoRdEn is asserted to '1' for 16 clock cycles to read 512-byte data from Transmit FIFO and send to data buffer
(4) UserFifoRdEn is de-asserted to '0' after end of 512-byte data
Write Command
UserFifoRdCnt[15:4]
/=0
UserFifoRdEn
1
2
3
4
16 clock cycles
UserFifoRdData[255:0]
D0
D1
D2
D14
D15
(3) Returned data valid the next clock cycle after UserFifoRdEn is asserted

Figure 7: Transmit FIFO Interface for Write command

For Read command, UserFifoWrEn is asserted to '1' with the valid value of UserFifoWrData to send Received data from data buffer. Similar to Write command, UserFifoWrCnt is monitored to check free space of Received FIFO is more enough (more than or equal 1024-byte) before transferring 512-byte data to FIFO.

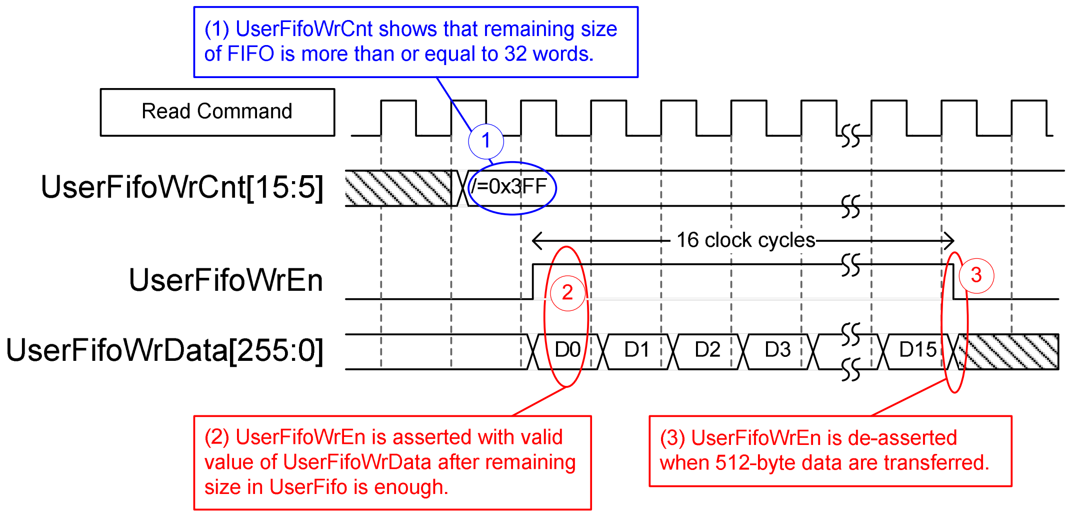
(1) UserFifoWrCnt shows that remaining size of FIFO is more than or equal to 32 words.
Read Command
UserFifoWrCnt[15:5]
/=0x3FF
1
16 clock cycles
UserFifoWrEn
2
3
UserFifoWrData[255:0]
D0
D1
D2
D3
D15
(2) UserFifoWrEn is asserted with valid value of UserFifoWrData after remaining size in UserFifo is enough.
(3) UserFifoWrEn is de-asserted when 512-byte data are transferred.

Figure 8: Received FIFO Interface for Read command

IdenCtrl/IdenName

Before sending Write or Read command to IP, user should send Identify command firstly to update LBASize output. LBASize value is used in User Logic to confirm that the sum of address and length in Write/Read command is not out-of-range.

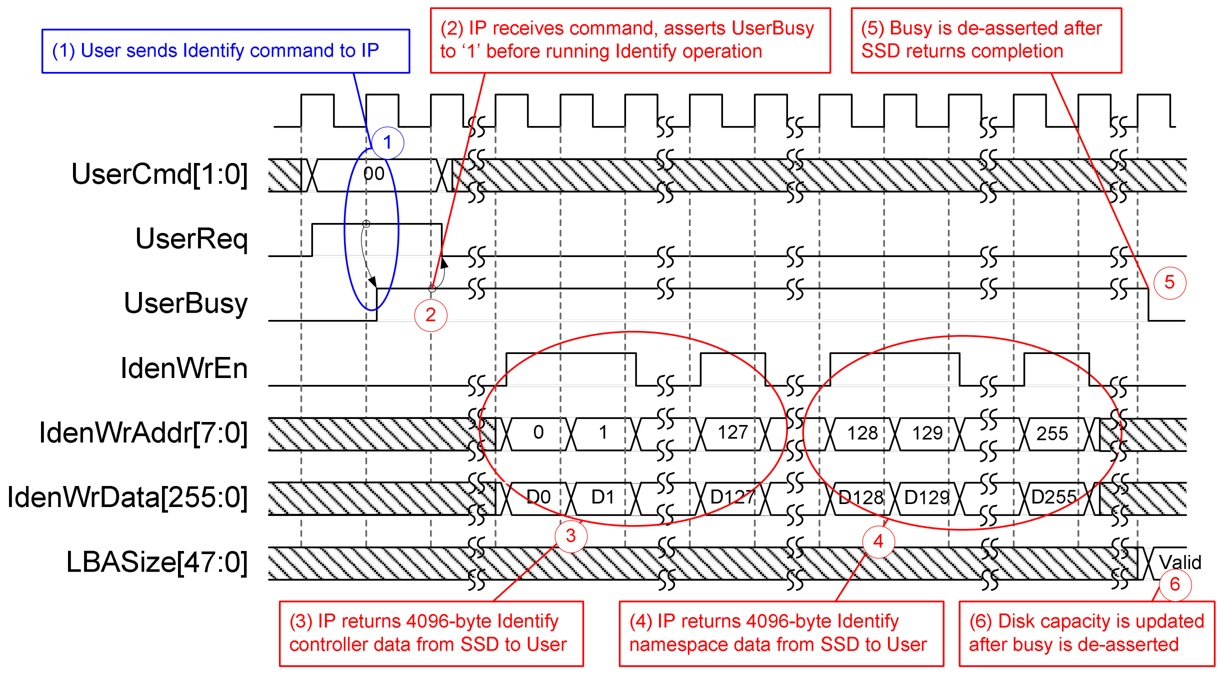
(1) User sends Identify command to IP
(2) IP receives command, asserts UserBusy to '1' before running Identify operation
(5) Busy is de-asserted after SSD returns completion
UserCmd[1:0]
00
UserReq
UserBusy
IdenWrEn
IdenWrAddr[7:0]
0
1
127
128
129
255
IdenWrData[255:0]
D0
D1
D127
D128
D129
D255
LBASize[47:0]
Valid
(3) IP returns 4096-byte Identify controller data from SSD to User
(4) IP returns 4096-byte Identify namespace data from SSD to User
(6) Disk capacity is updated after busy is de-asserted

Figure 9: LBASize update after Identify command

As shown in Figure 9, UserCmd and UserReq are set when UserBusy='0'. UserAddr and UserLen input are not required for Identify command. After Identify command is sent, 4096-byte Identify Controller data and 4096-byte Identify Namespace data are received. IdenWrAddr is equal to 0 – 127 during sending Identify Controller data and IdenWrAddr is equal to 128 - 255 during sending Identify Namespace data. IdenWrData shows 16-byte of Identify Controller data or Identify Namespace data in each clock cycle, synchronous to IdenWrAddr and IdenWrEn signal. Finally, LBASize is updated after UserBusy is de-asserted to '0' from Identify command.

Error

During normal operation, UserError and all bits of UserErrorType signal are always 0. UserError is generated by OR condition of each-bit of UserErrorType. If some bits of UserErrorType are asserted to ‘1’, UserError will be asserted to ‘1’. UserError is latched to ‘1’ by asserting RstB to ‘0’.

If AdmCompStatus or IOCompStatus value has error condition, UserErrorType bit[3]/[5] will be set. User can see more details of the error by reading AdmCompStatus and IOCompStatus value.

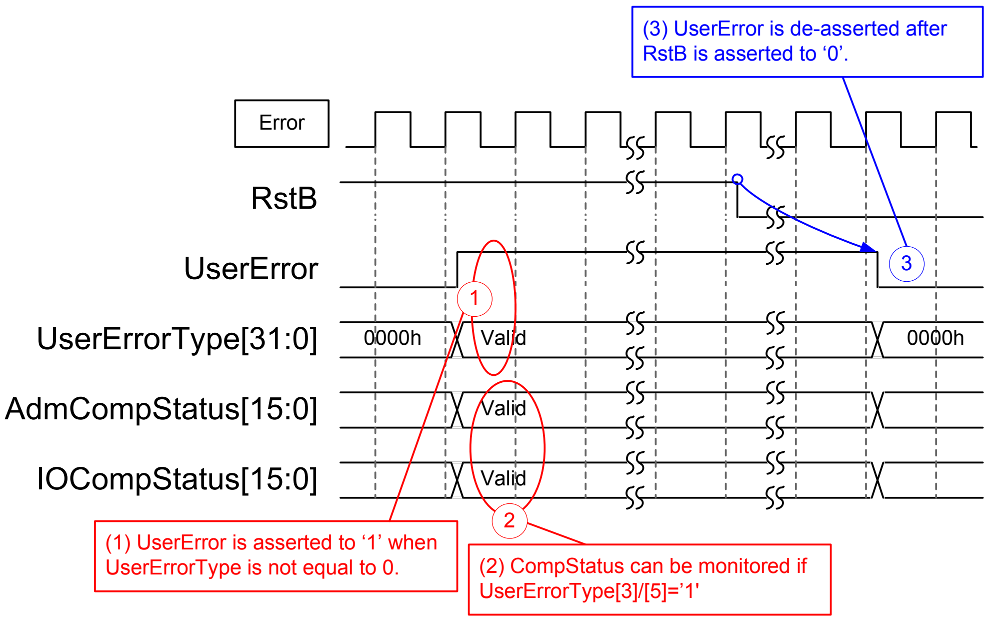
(3) UserError is de-asserted after RstB is asserted to ‘0’.
Error
RstB
UserError
UserErrorType[31:0]
AdmCompStatus[15:0]
IOCompStatus[15:0]
0000h
Valid
0000h
Valid
Valid
1
2
3
(1) UserError is asserted to ‘1’ when UserErrorType is not equal to 0.
(2) CompStatus can be monitored if UserErrorType[3]/[5]=’1'

Figure 10: Error flag Timing diagram

Verification Methods

The NVMe IP Core functionality was verified by simulation and also proved on real board design by using KCU105/ZCU106 board.

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 Gatway 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
1.0	Feb-7-2018	Initial Release
1.1	Oct-9-2018	Support ZCU102
1.2	Feb-2-2019	Support PCIe switch connection
1.3	Feb-13-2019	Correct Figure 4-5