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
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
- 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 available on KCU105 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	3923	2561	772	-	61	-	-	Vivado2017.4

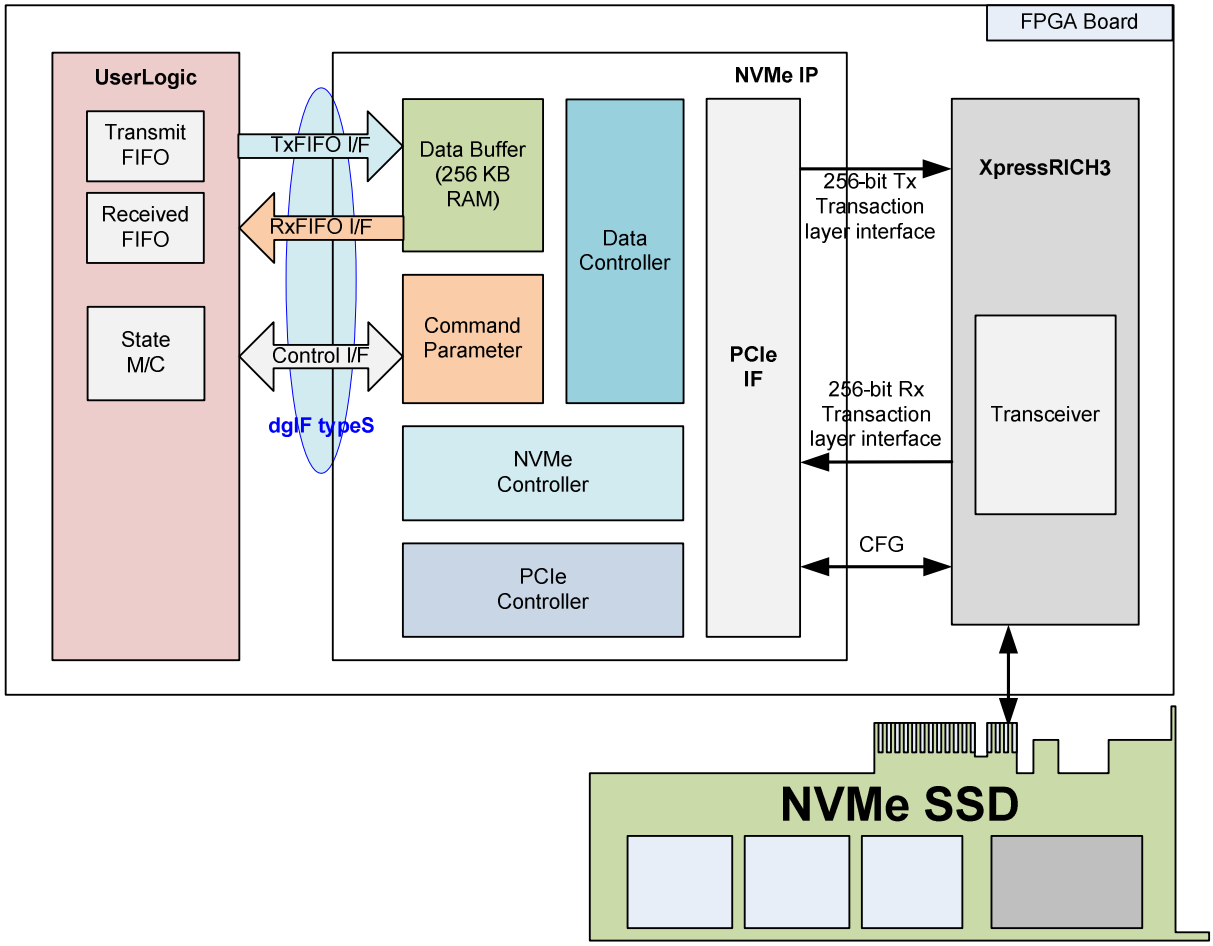

Figure 1: NVMe IP Block Diagram

Applications

NVMe IP Core integrated with PCI Express XpressRICH3 and PHY from PLDA is an ideal to access NVMe PCIe 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 PCIe SSD. This solution is recommended for the application which requires high capacity storage at very high-speed performance. Small size system can be designed by using M.2 PCIe storage.

General Description

NVMe IP implements as host controller to access NVMe PCIe 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 three NVMe commands, i.e. Identify, Write, and Read command.

NVMe protocol operates simultenous commands, so NVMe IP needs to include big data buffer to send or receive data for multiple write or multiple read commands at the same time. As a result, using multiple commands can achieve the best write and read performance of SSD.

The user interface of NVMe IP is simply designed by dgIF typeS which consists of two interfaces, i.e. command interface and data interface. The inputs of command interface are write/read command, start address, and transfer length. The data interface is designed by using general FIFO standard. The clock frequency of user logic must be higher 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.

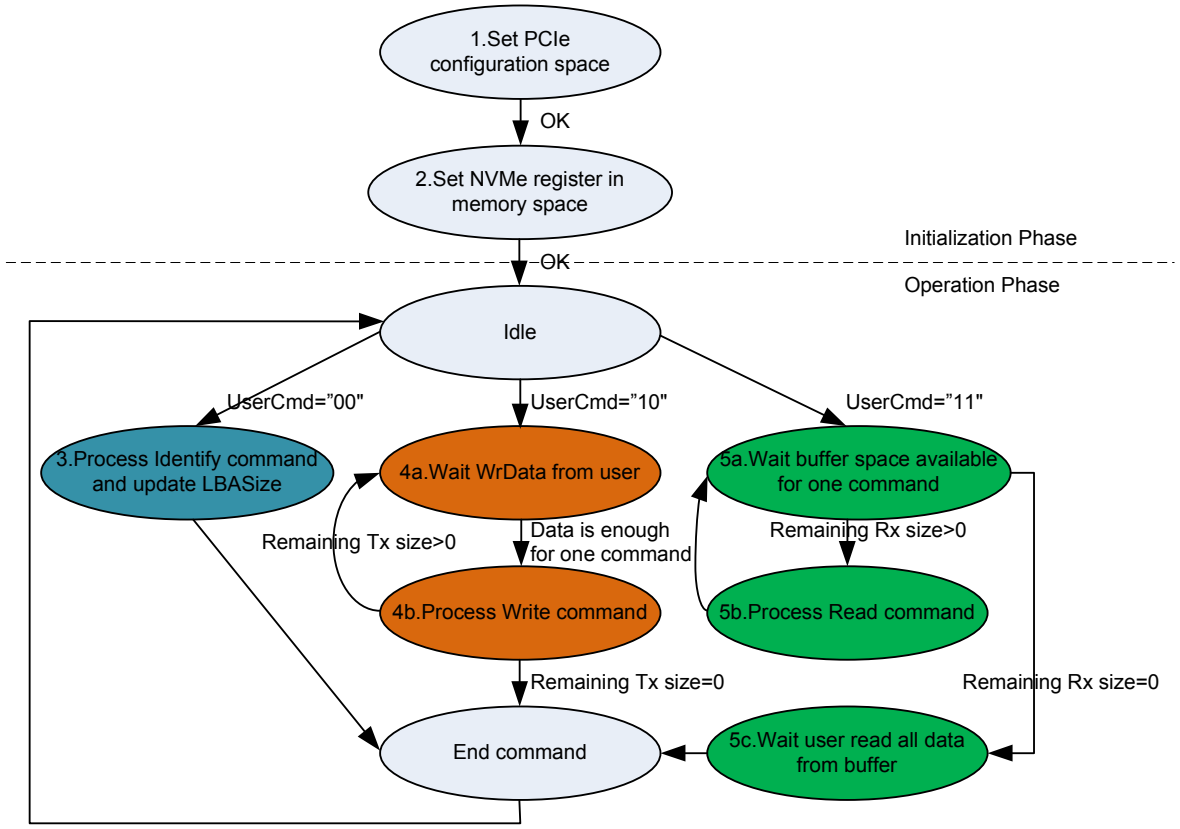

Figure 2: NVMe IP Operation Flow

1) IP sets PCIe configuration space to setup PCIe environment for NVMe operation.
2) IP sets NVMe parameters by access memory space to setup SSD's environment. After complete initialization process, IP is in idle state to wait new command 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 write data from user in the data buffer is enough for transferring in one command (Transfer size of one command in NVMe IP is 128 Kbyte except the last command. Transfer size of the last command is equal to the remaining size which is not aligned to 128Kbyte when total transfer size from user is not aligned to 128Kbyte).
   - IP sends write command to NVMe SSD.
   - IP waits status from SSD to confirm that all data in the command have been transferred completely.
   - If remaining transfer size is not zero, IP will continue to check the numbers of write data in the data buffer for sending the next command.
   - If remaining transfer size is zero, IP will go back to Idle state.

5) In case of read command,
   - IP monitors free space in data buffer whether it is enough for receiving data of one command or not.
   - IP sends read command to NVMe SSD.
   - IP waits status from SSD to confirm that all data in the command have been transferred completely.
   - If remaining transfer size is not zero, IP will check free space 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 Controller, and Data Controller.

After system power-on, PCIe Controller setups PCIe environment for connecting with SSD. Next, NVMe Controller initializes NVMe register in SSD following NVMe specification to complete initialization phase.

For operating phase, it starts when user sends command to NVMe IP through dgIF typeS interface. NVMe controller decodes the command from user and loads the parameters to store in Command parameter. The sequence of each NVMe command is controlled by NVMe controller. For Data Controller, it is designed to handle Command packet, Status packet, and Data packet.

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 in SSD. To write/read configuration space, the packets are transferred through 256 bit transaction layer interface.

  XpressRICH3 needs to setup configuration space through CFG interface. Therefore, PCIe controller with state machine inside is created to control intialization sequence of configuration space in XpressRICH3.

NVMe

Following NVMe standard, the NVMe host communicates with the NVMe device by using four queue types, i.e. Admin Submission for the NVMe host sending Admin command, Admin Completion for the SSD returning ACK, I/O Submission for the host sending I/O command, and I/O Completion for the SSD returning ACK. To send new command to SSD, the NVMe host prepares command to Submission queue, and updates Submission queue tail pointer to doorbell register. After SSD completes to process command, it writes completion status to Completion queue. The NVMe host updates Completion queue head pointer to doorbell register after completes to process completion. The sequence of each command operation is designed in NVMe Controller, while data packet is processed by Data Controller. Data packet has two types, i.e. Raw data which is stored in Data buffer and Control data and Status data which are stored in Command parameter.

- **NVMe Controller**

  When user sends new commands to NVMe IP, NVMe controller processes command, address, and length from user logic. After that, it creates Submission Queue to store Command parameter and updates tail pointer of Submission Queue to doorbell register. For Write/Read command, if total transfer length is more than 128 Kbyte size, NVMe controller will generate multiple commands. The transfer length of each command is equal to 128 Kbyte size, except the last packet. The size of the last packet is equal to the remaining transfer size which is equal or less than 128 Kbyte.

  For Write command, tail doorbell of I/O Submission queue is updated to send new command after all raw data from user logic for the new command are stored in Data buffer. For Read command, tail doorbell of I/O Submission queue is updated after free space size of data buffer is more than or equal to 128 Kbyte. The status value within Completion queue is extracted by Data controller and stored in Command Parameter. NVMe Controller monitors the status to confirm that the command is completed without the error.

  For Write command, busy signal output to user is cleared after SSD returns the status to I/O Completion Queue without the error. For Read command, busy signal is cleared after user reads all data from the data buffer.

- **Data Buffer**
  256 Kbyte simple dual port RAM is implemented by BlockRAM. This RAM is used to store raw data transferring from UserLogic to SSD for Write command or transferring from SSD to UserLogic for Read command.

- **Command Parameter**
  This block is used to store command packet (Admin and I/O submission queue) and status packet (Admin and I/O completion queue). Command packet is prepared by NVMe Controller, but read by Data Controller. Status packet is sent from Data Controller, but read by NVMe Controller.

- **Data Controller**

  Two data types are processed in Data Controller. Raw data in data buffer and Control data in Command parameter are mapped into different memory space. Data controller selects data source or destination by decoding the address in the request which is sent by SSD. To build the packet sending through XpressRICH3, data from data buffer or command parameter is combined with TLP header which is extracted from the received request packet sent by XpressRICH3. So, the logic includes small memory to store the header of TLP packet. Data bus size of data controller is 256-bit which is the bus size of PCI Express XpressRICH3.

- **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.

User Logic

Simple logic with small state machine to send command, address, and size can be designed for command interface of dgIF typeS. Data stream is designed to transfer by using FIFO interface.

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

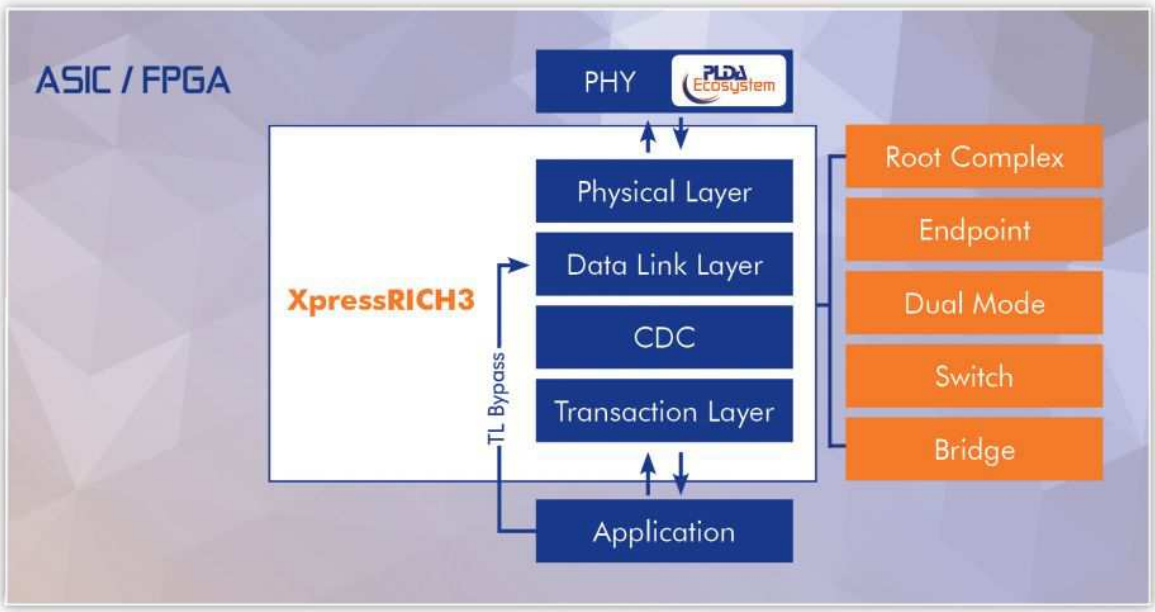

Figure 3: XpressRICH3 CORE

Core I/O Signals

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

Table 2: Core I/O Signals

Signal	Dir	Description
dgIF typeS		
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).
UserCmd[1:0]	In	User Command. "00": Identify command, "10": Write PCIe SSD, "11": Read PCIe SSD.
UserAddr[47:0]	In	Start address to write/read SSD in sector unit (512 byte). *Note: From SSD characteristic, it is recommended to set bit[2:0]="000" to align 4 Kbyte size which is SSD page size. Write/Read performance in most SSD are reduced when start addrss is not aligned to 4 Kbyte unit.*
UserLen[47:0]	In	Total transfer size in the request in sector unit (512 byte). Valid from 1 to (LBASize-UserAddr).
UserReq	In	Request the new command. Can be asserted only when the IP is Idle (UserBusy='0'). Asserted with valid value on UserCmd/UserAddr/UserLen signals.
UserBusy	Out	IP Busy status. New request will not be allowed if this signal is asserted to '1'.
LBASize[47:0]	Out	Total capacity of PCIe SSD in sector unit (512 byte). Default value is 0. This value is updated after user sets Identify command.
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. [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.

Signal	Dir	Description
dgIF typeS		
UserError	Out	Error flag. Assert when UserErrorType is not equal to 0. The flag can be cleared by asserting RstB signal.
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	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.
UserFifoRdEn	Out	Read valid of 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	Valid signal of IdenWrData and IdenWrAddr. Asserted when Identify controller data or Identify Namespace data is transferred.
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 IdenCtrlWrEn.

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

The sequence of the initialization process is shown as follows.
1) RstB is released to '1' when Clk is stable.
2) NVMe IP waits until PCIeRstB and PCIeLinkup are asserted to '1' to confirm that XpressRICH3 is in ready status.
3) PCIeLinkup is asserted when XpressRICH3 is ready to exchange data packets.
4) NVMe IP starts initialization process.
5) UserBusy is deasserted to '0' after NVMe IP completes initialization process.
After complete above sequence, NVMe IP is ready to receive the command from user.

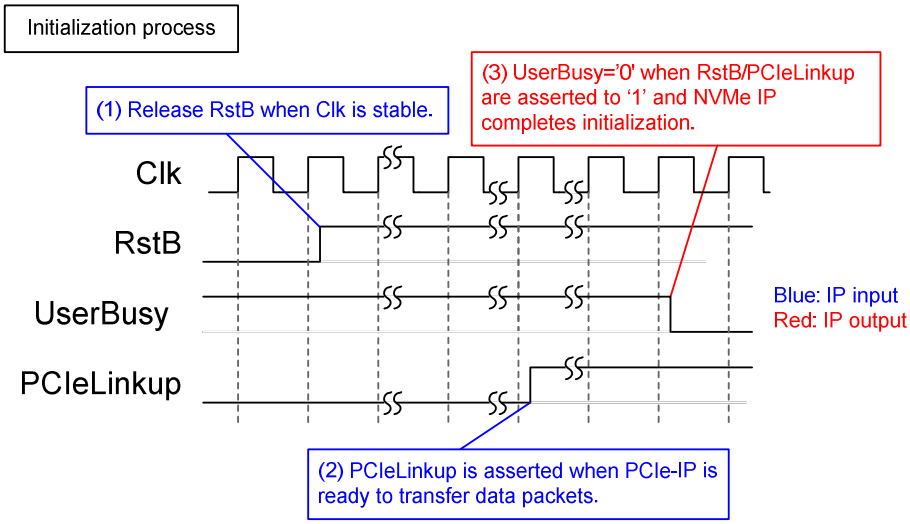

Figure 4: UserBusy after system boot-up

dgIF typeS

dgIF typeS signal is split into two interfaces, i.e. command interface and data interface. Figure 5 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. UserCmd, UserAddr, and UserLen must be valid and latched during asserting UserReq='1'. UserBusy changes status from '0' to '1' after start the command operation. Finally, UserReq is de-asserted to '0' and user logic can prepare the next command to the command 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 6 and Figure 7.

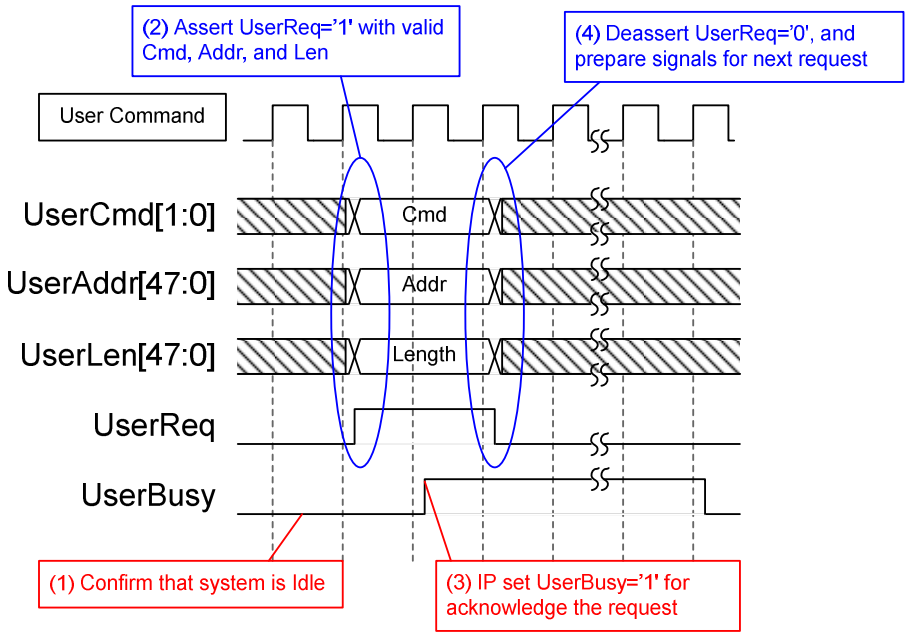

Figure 5: Command Interface of dgIF typeS Timing diagram

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 for 16 clock cycles to read 512-byte data, as shown in Figure 6. Similar to general FIFO timing diagram, UserFifoRdData is valid in the next clock cycle after UserFifoRdEn is asserted.

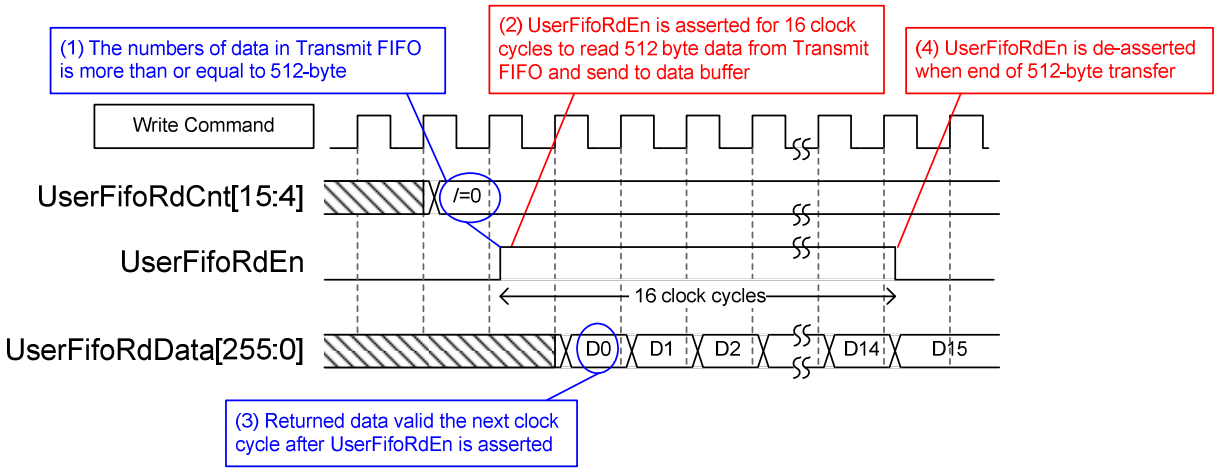

Figure 6: Transmit FIFO Interface for Write command

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

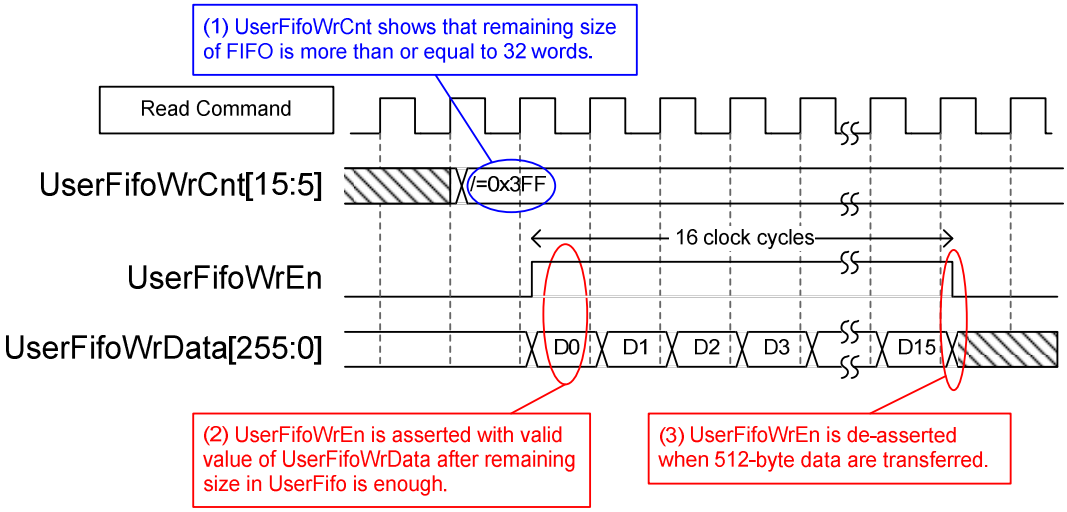

Figure 7: Received FIFO Interface for Read command

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.

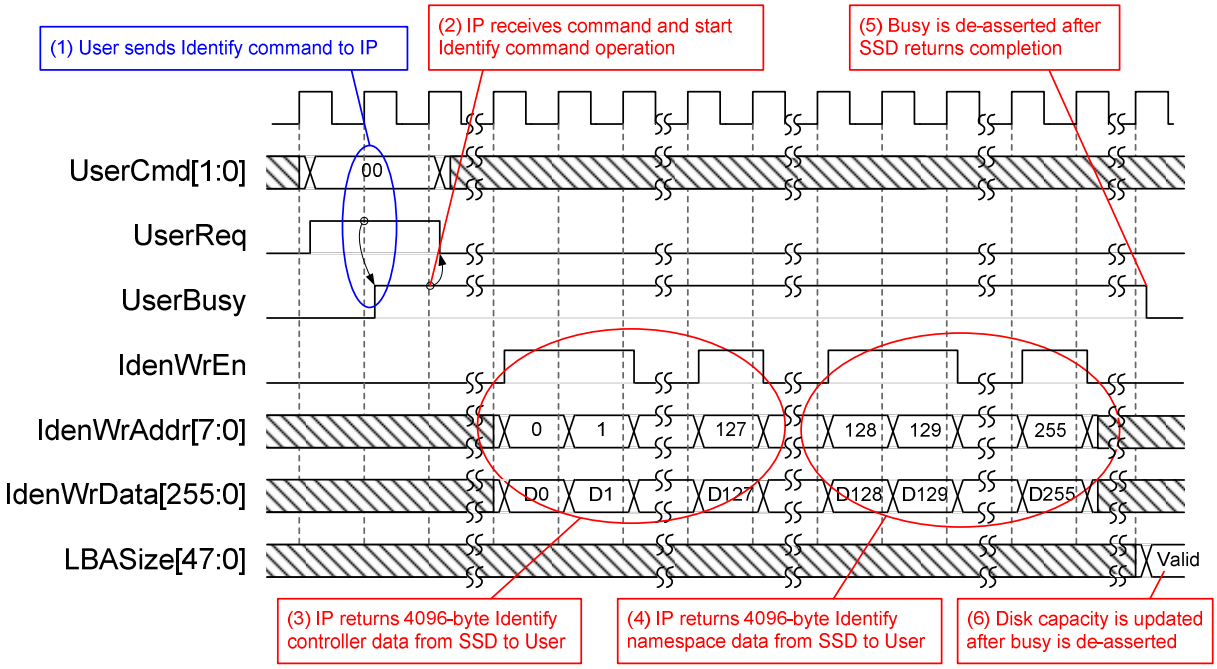

Figure 8: LBASize update after Identify command

As shown in Figure 8, 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 equal to '1', UserError will be asserted to '1'. In error condition, UserError is latched to '1' until IP is reset (RstB='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.

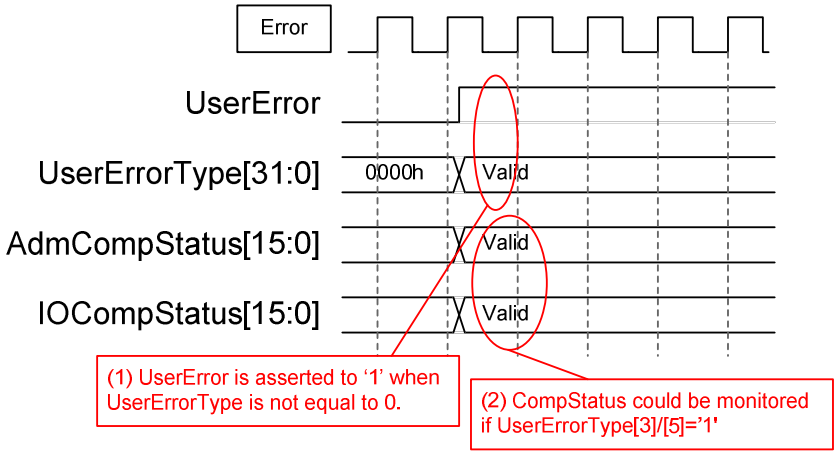

Figure 9: Error flag Timing diagram

Verification Methods

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