NVMe IP Core

June 8, 2017 Product Specification Rev2.1

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 hdl File
Instantiation Templates	VHDL
Reference Designs & Application Notes	QuartusII Project, See Reference Design Manual
Additional Items	Demo on ArriaV GX starter kit, Arria10 SoC development kit, Arria10 GX development kit
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
- User clock frequency must be more than or equal to PCIe clock (125 MHz for PCIe Gen2, 250 MHz for PCIe Gen3)
- Direct connect to Avalon-ST Hard IP for PCI Express from Intel by using 128-bit bus interface
- Include 256 Kbyte RAM to be data buffer
- Support three commands, i.e. IDENTIFY, WRITE, and READ.
- Support 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 Arria V GX Starter board, Arria10 GX development board and without adapter on Arria10 SoC development board.

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	Logic utilization (ALMs)	Registers	Pin	Block Memory bit[1]	Design Tools
ArriaV GX	5AGXFB3H4F35C4	212	1175	2133	-	2,162,688	QuartusII 16.0
Arria10 SX	10AS066N3F40E2SGE2	280	1144	2120	-	2,162,688	QuartusII 16.0

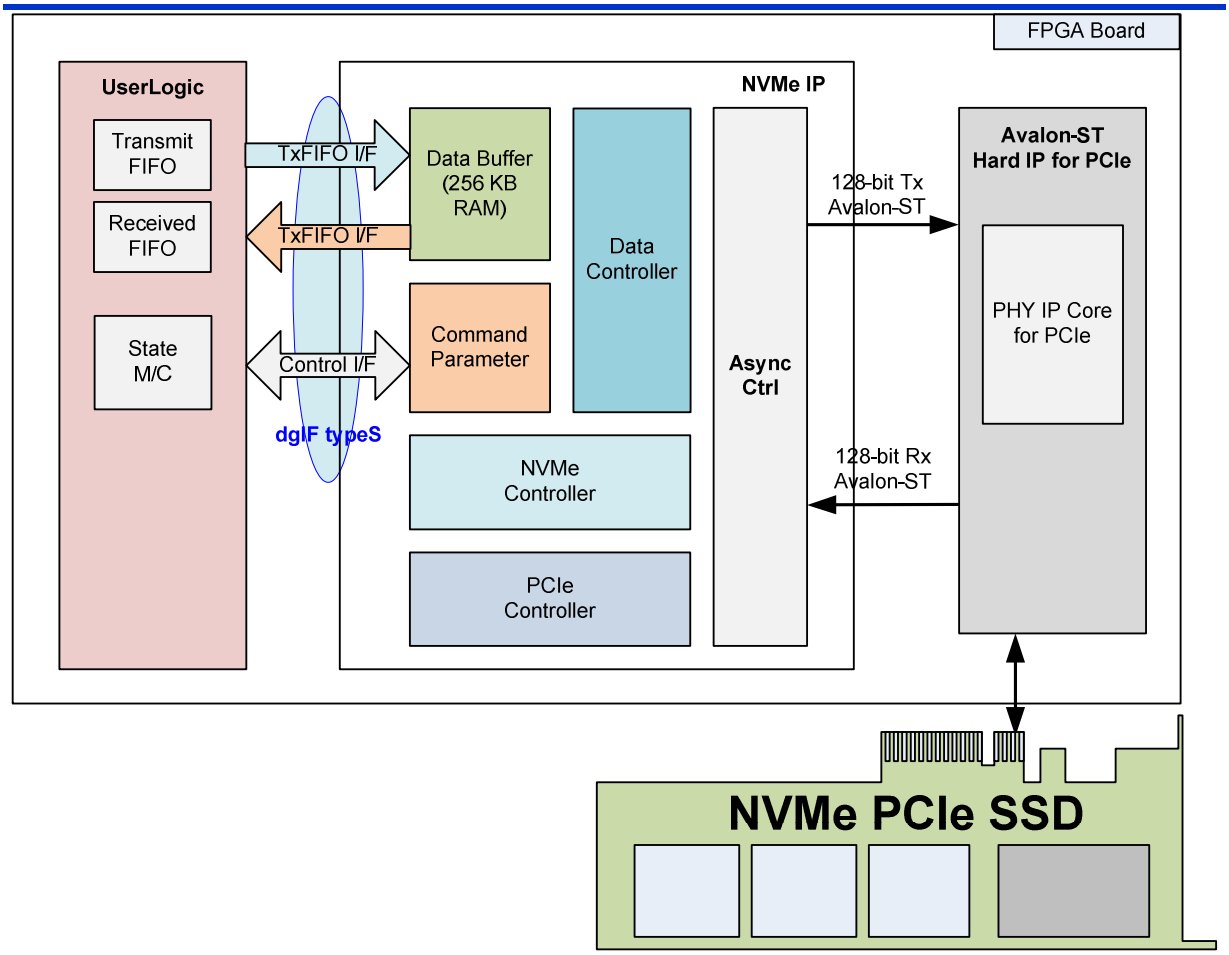

Figure 1: NVMe IP Block Diagram

Applications

NVMe IP Core integrated with Avalon-ST PCIe Hard IP from Intel is ideal to access NVMe PCIe SSD without CPU or DDR. 256 Kbyte buffer implemented by Block Memory is included in NVMe IP Core to store data which is transferred between user logic and PCIe SSD. It is recommended to use in 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 NVM express standard. Physical interface of NVMe SSD is PCIe, so the hardware of lower layer is implemented by using Avalon-ST PCIe Hard IP from Intel. NVMe IP supports three NVMe commands, i.e. Identify, Write, and Read command. NVMe protocol supports multiple commands, so NVMe IP needs to include big data buffer (256 Kbyte RAM) to send or receive data of multiple write/read commands at the same time. Using multiple commands can achieve the best write/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. From PCIe Hard IP limitation, clock frequency of user logic must be more than or equal to PCIe clock frequency (250 MHz for PCIe Gen3, 125 MHz for PCIe Gen2). Error signal will be asserted with the error status if IP detects the abnormal condition during packet transferring.

The reference design on Intel Development boards are available to evaluate before purchasing

Functional Description

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

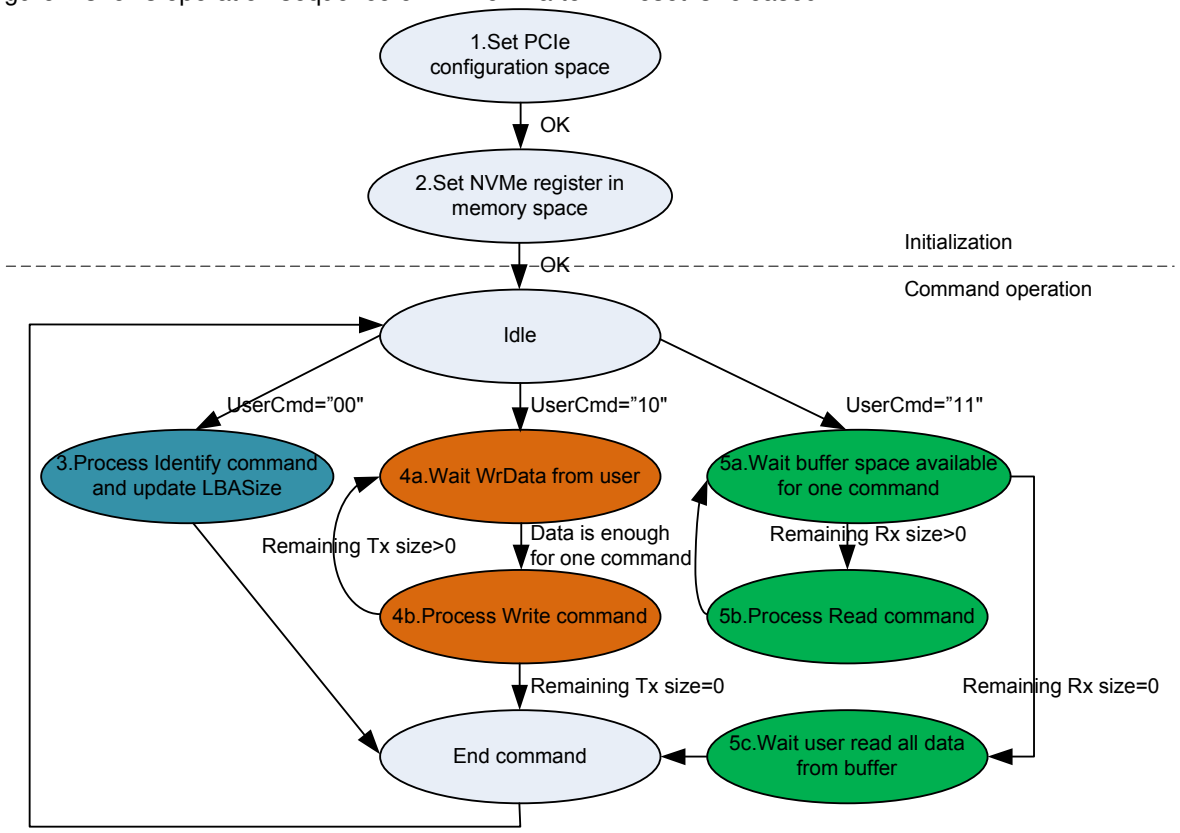

Figure 2: NVMe IP Operation Flow

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 much enough for transferring in one command (Maximum transfer size of one command in NVMe IP is 128 Kbyte).
   - 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 will monitor free space in data buffer that is enough for receiving data of one command
   - 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 sequence, NVMe IP consists of three controllers, i.e. PCIe Controller, NVMe Controller, and Data Controller. After system power-on, PCIe Controller will setup PCIe environment for connecting with SSD. Next, NVMe Controller initializes NVMe register within SSD following NVMe specification to complete initialization phase.

For command 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 parameter to store in Command parameter. The sequence of each NVMe command is controlled by NVMe controller. Command packet, Status packet, and Data packet are processed by Data controller.

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 within SSD. To write/read configuration space, the packets are transferred through 128-bit Tx/Rx Avalon-ST bus.

  Avalon-ST Hard IP for PCIe also needs to setup configuration space through 128-bit Tx/Rx Avalon-ST bus firstly. PCIe controller also includes state machine to control intialization sequence of configuration space within Avalon-ST Hard IP for PCIe.

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, SSD will write completion status to Completion queue. The NVMe host will update 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 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 in 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 will be cleared after SSD returns the status to I/O Completion Queue without the error. For Read command, busy signal will be cleared after user reads all data from the data buffer.

- **Data Buffer**
  256 Kbyte dual port RAM is implemented by Block Memory. 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 to PCIe Hard IP, data from data buffer or command parameter is combined with TLP header which is extracted from the received request packet sent by PCIe Hard IP. So, the logic includes small memory to store the header of TLP packet. Data bus size of data controller is 128-bit which is the bus size of PCIe Hard IP.

- **AsyncCtrl**

  NVMe IP is designed to support user clock domain which is different from PCIe clock, but user clock frequency must be higher than or equal to PCIe clock. AsyncCtrl includes small asynchronous FIFO which is implemented by MLAB to support clock domain crossing.

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.

Avalon-ST PCIe Hard IP

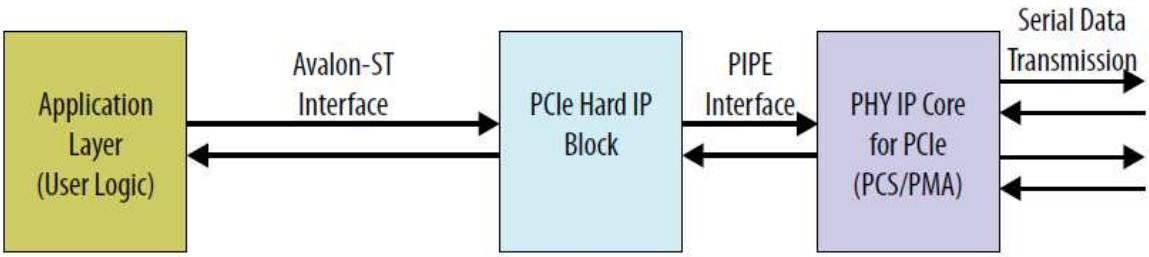

Figure 3: Architecture of Avalon-ST PCIe Hard IP

PCIe Hard IP Block embeds the PCIe protocol stack into the Intel FPGA. The hard IP block includes the transceiver modules, physical layer, data link layer, and transaction layer. The maximum numbers of SSD in one FPGA device is limited by the numbers of PCIe Hard IP Block. More details of Avalon-ST PCIe Hard IP are described in “ArriaV Avalon-ST Interface for PCIe Solutions User Guide” or “Intel Arria10 Avalon-ST Interface for PCIe Solutions User Guide”.
https://www.altera.com/en_US/pdfs/literature/ug/ug_a5_pcie_avst.pdf
https://www.altera.com/en_US/pdfs/literature/ug/ug_a10_pcie_avst.pdf

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 (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 frequency of Clk must be more than or equal to PCIeClk which is output from Avalon-ST PCIe Hard IP (125 MHz for PCIe Gen2, 250 MHz for PCIe Gen3).
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). 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 will be updated after user sets Identify command.
UserError	Out	Error flag. Assert when UserErrorType is not equal to 0. The flag is 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 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. [6] – Error when Completion TLP packet size is not correct. [7] – Error when Avalon-ST PCIe Hard IP detects 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. [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). [31:12] - Reserved Note: Timeout period of bit[2]/[4] is set from TimeOutSet input.

Signal	Dir	Description
dgIF typeS (Synchronous to Clk)		
UserFifoWrCnt[15:0]	In	Write data counter of Received FIFO. Used to check 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[127:0]	Out	Write data bus of Received FIFO. Synchronous to UserFifoWrEn.
UserFifoRdCnt[15:0]	In	Read data counter of Transmit FIFO. Used to check the numbers of data in FIFO. If FIFO size is less than 16-bit, please fill '0' to upper bit.
UserFifoEmpty	In	FIFO empty flag of Transmit FIFO to check data available status.
UserFifoRdEn	Out	Read valid of Transmit FIFO.
UserFifoRdData[127:0]	In	Read data returned from Transmit FIFO. Valid in the next clock after UserFifoRdEn is asserted.
NVMe IP Interface (Synchronous to Clk)		
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).
LinkSpeed[1:0]	Out	PCIe speed. "00": No linkup, "01": Gen1 (2.5 Gbps), "10": Gen2 (5.0 Gbps), "11": Gen3 (8.0 Gbps).
PCIeLinkup	In	Asserted to '1' when LTSSM state of PCIe Hard IP 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
IdenCtrlWrEn	Out	Valid signal of IdenCtrlWrData and IdenCtrlWrAddr. Asserted when Identify controller data is transferred.
IdenCtrlWrAddr[7:0]	Out	Index of IdenCtrlWrData in 128-bit unit. Synchronous to IdenCtrlWrEn.
IdenCtrlWrData[127:0]	Out	4Kbyte Identify controller data from Identify command. Synchronous to IdenCtrlWrEn.
IdenNameWrEn	Out	Valid signal of IdenNameWrData and IdenNameWrAddr. Asserted when Identify Namespace is transferred.
IdenNameWrAddr[7:0]	Out	Index of IdenNameWrData in 128-bit unit. Synchronous to IdenNameWrEn.
IdenNameWrData[127:0]	Out	4Kbyte Identify Namespace data from Identify command. Synchronous to IdenNameWrEn.

Signal	Dir	Description
Avalon-ST PCIe Hard IP I/F (Synchronous to PCIeClk)		
PCIeRstB	In	Synchronous reset signal. Active low. Release to '1' when PCIe Hard IP is not in reset state.
PCIeClk	In	Clock output from Avalon-ST PCIe Hard IP within NVMe IP (125 MHz for PCIe Gen2 and 250 MHz for PCIe Gen3).
PCIeRxValid	In	Indicates that PCIeRxData is valid. Deasserts within 2 clocks of PCIeRxReady desassertion.
PCIeRxEOP	In	Indicates that this is the last cycle of the TLP when PCIeRxValid is asserted.
PCIeRxReady	Out	Indicates that NVMe IP is ready to accept data.
PCIeRxError	In	Indicates that there is an uncorrectable error correction coding (ECC) in the internal RX buffer of Avalon-ST PCIe Hard IP.
PCIeRxData[127:0]	In	Receive data bus.
PCIeTxValid	Out	Indicates that PCIeTxData is valid when PCIeTxReady is also asserted
PCIeTxSOP	Out	Indicates first cycle of a TLP when asserted together with PCIeTxValid.
PCIeTxEOP	Out	Indicates last cycle of a TLP when asserted together with PCIeTxValid.
PCIeTxEmpty	Out	Indicates whether the upper qword at the end of a packet (PCIeTxEOP is asserted) contains data. '0': PCIeTxData[127:0] is valid, '1': PCIeTxData[63:0] is valid.
PCIeTxReady	In	Indicates that Avalon-ST PCIe Hard IP is ready to accept data.
PCIeTxError	Out	Indicates an error on transmitted TLP. This signal is always set to '0'.
PCIeTxData[127:0]	Out	Data for transmission.

Timing Diagram

Initialization

The sequence of the initialization process is as follows.
1) RstB is released by user when Clk is stable.
2) NVMe IP waits until both PCIeRstB and PCIeLinkup are asserted to '1' to confirm that PCIe Hard IP is in ready status.
3) PCIeRstB is deasserted to '1' when PCIe Hard IP is not in reset state and ready to interface with application layer. PCIeRstB is generated in PCIeClk domain.
4) PCIeLinkup is asserted when LTSSM State within PCIe Hard IP is in L0.
5) NVMe IP starts initialization process.
6) UserBusy is deasserted to '0' after NVMe IP completes initialization process.
After complete above sequence, NVMe IP will be ready to receive the command from user.

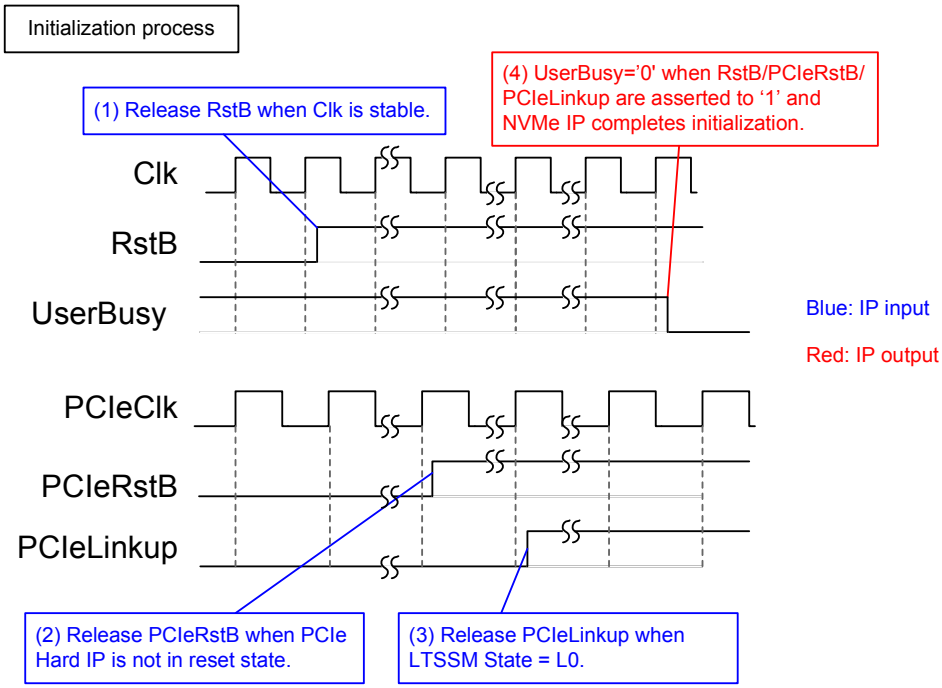

Figure 4: UserBusy after system boot-up

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 always monitored to confirm that IP is Idle. UserCmd, UserAddr, and UserLen must be valid and latched during asserting UserReq='1'. UserBusy will change status from '0' to '1' after the IP starts 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 for Write command, while Received FIFO is written for Read command. Timing diagram of data interface is shown in Figure 6 and Figure 7.

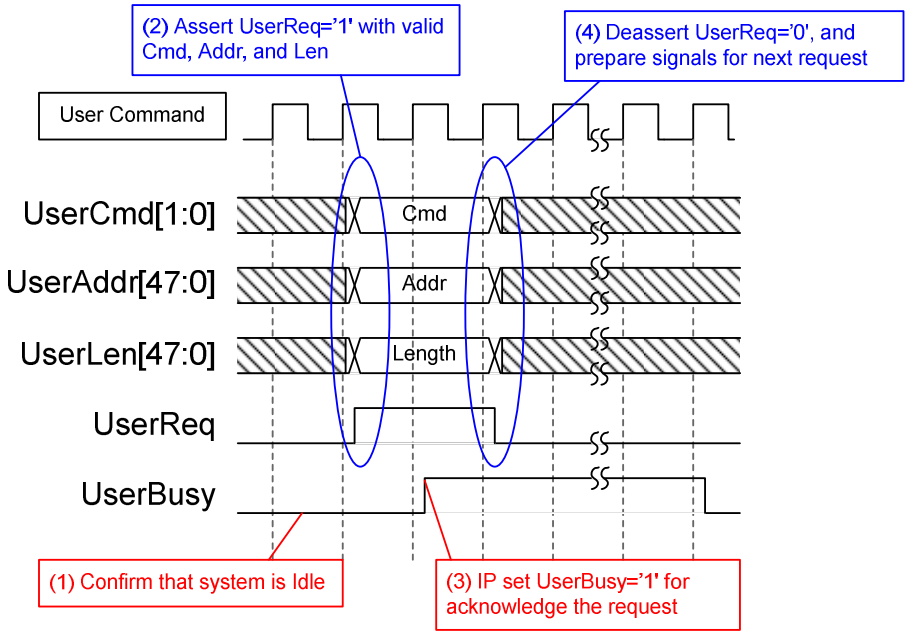

Figure 5: Command Interface of dgIF typeS Timing diagram

For Write command, 128-bit data from Transmit FIFO is stored to data buffer within NVMe IP. DMA Engine in NVMe IP monitors UserFifoRdCnt signal until it indicates that data in Transmit FIFO is equal to more than 512 bytes. After that, UserFifoRdEn is asserted for 32 clocks to read 512-byte data, as shown in Figure 6. Similar to general FIFO timing diagram, UserFifoRdData is valid in the next clock after UserFifoRdEn is asserted.

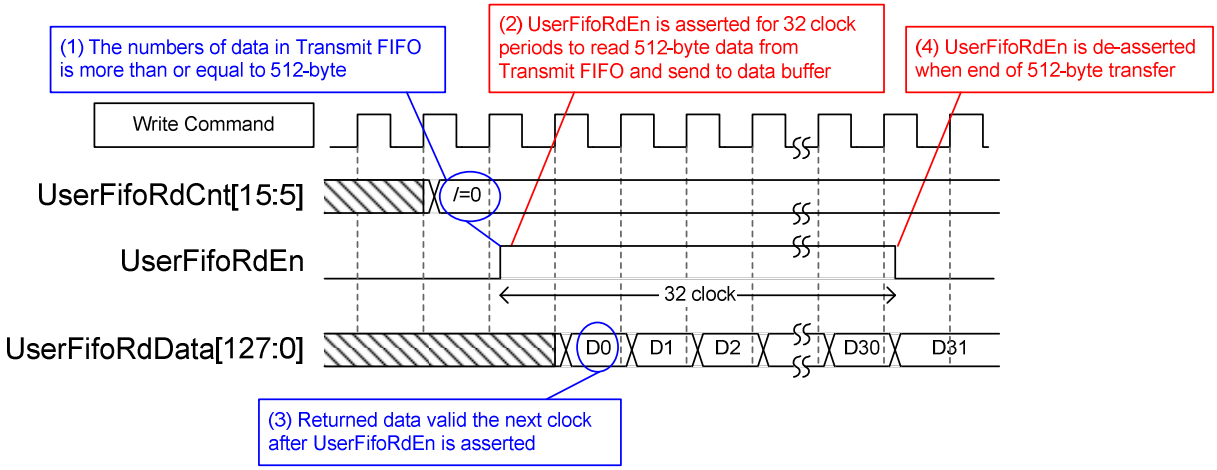

Figure 6: Transmit FIFO Interface for Write command

For Read command, UserFifoWrEn is asserted with the valid value of UserFifoWrData to store Receive 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.

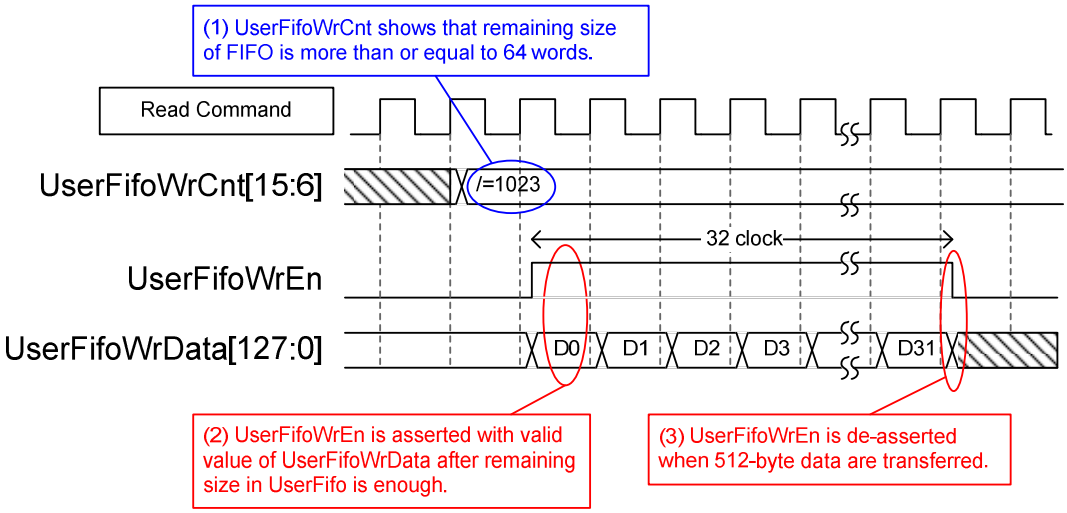

Figure 7: Received FIFO Interface for Read command

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.

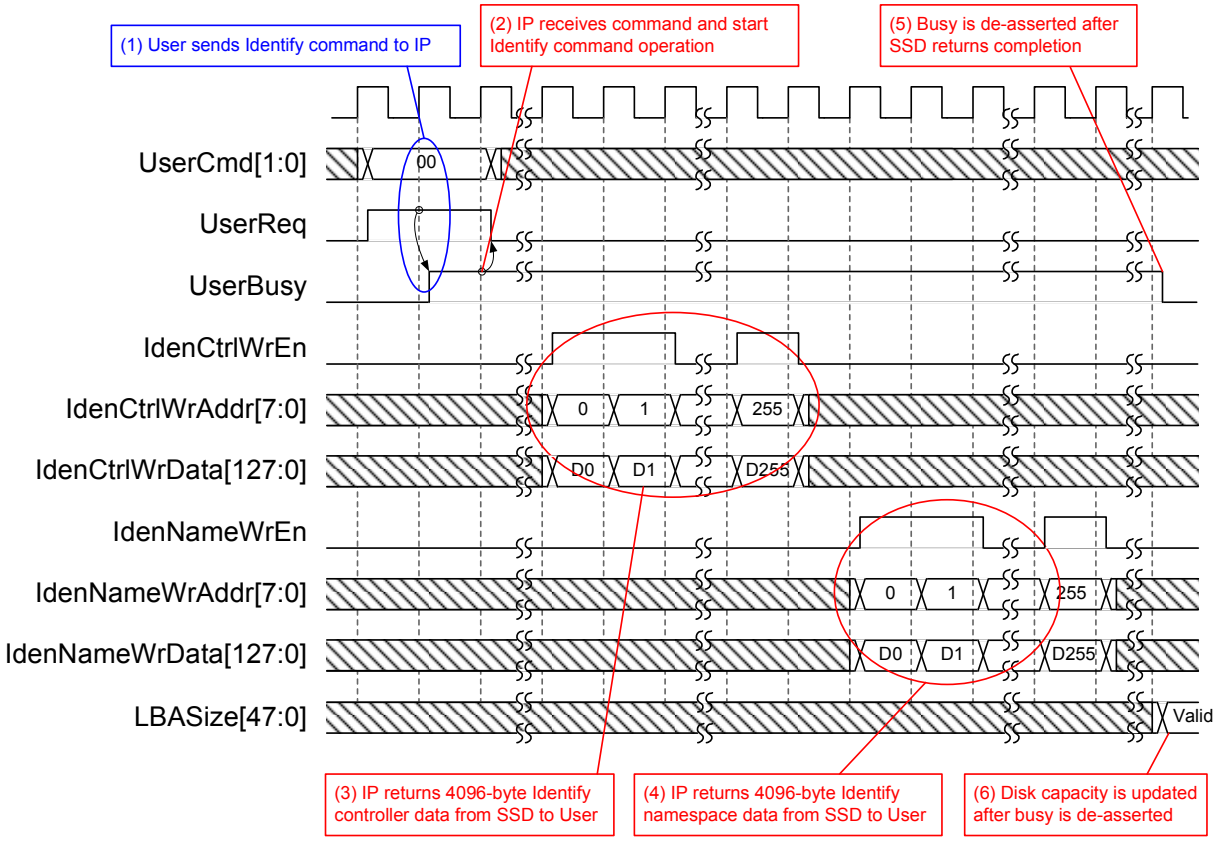

Figure 8: LBASize update after Identify command

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 will be received. Both Identify Controoler data and Identify Namespace data are not transferred continously. 4096-byte data is split in many burst transfers depending on SSD characteristic. Finally, LBASize is updated when UserBusy is de-asserted 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 is set to '1', UserError will be asserted and latched until RstB is asserted to '0', as shown in Figure 9.

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.

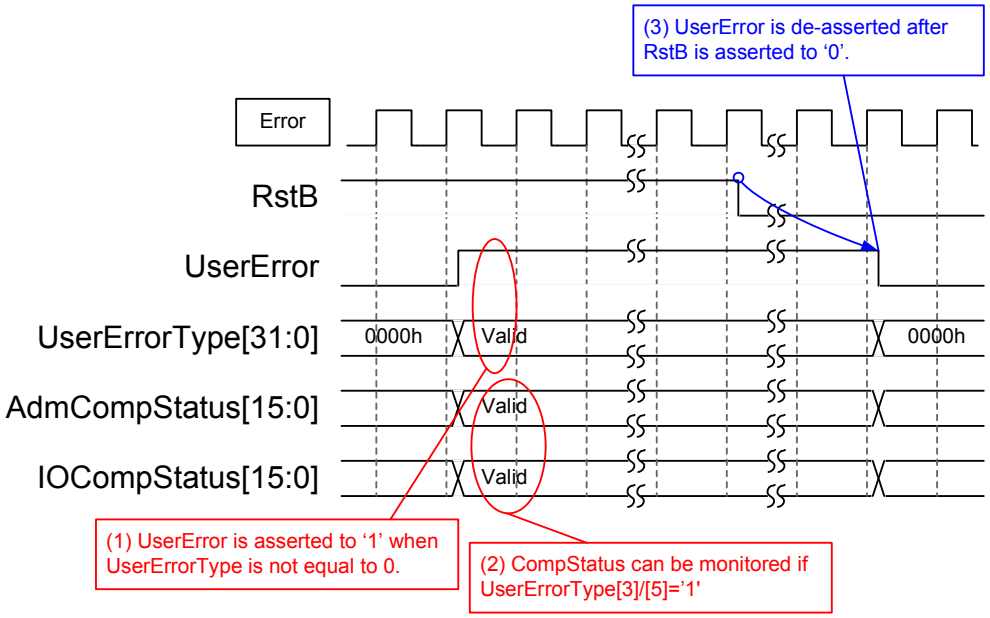

Figure 9: Error flag Timing diagram

Verification Methods

The NVMe IP Core functionality was verified by simulation and also proved on real board design by using ArriaV GX Starter board/Arria10 SoC Development board/Arria10 GX Development board.

Recommended Design Experience

Experience design engineers with a knowledge of QuartusII 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	Aug-9-2016	New release
1.1	Dec-15-2016	Modify buffer to be Block Memory and modify user interface to dgIF typeS
2.0	May-4-2017	Built-in 256/512 Kbyte buffer and connect to Avalon-ST PCIe Hard IP
2.1	Jun-8-2017	Support only 256 Kbyte buffer