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
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- Implement application layer to access NVMe PCIe SSD without CPU usage
- Simple user control I/F and FIFO interface for data port
- Direct connect to Avalon-MM Hard IP for PCI Express from Altera by using 128-bit bus interface
- 256 Kbyte (MODE=1) or 512 Kbyte (MODE=2) data buffer connecting through 128-bit Avalon bus interface (Bigger buffer size by using DDR is supported as optional)
- Support three commands, i.e. IDENTIFY DEVICE, 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): 0 (no limitation) or at least 5 (128 Kbyte)
- Reference design with AB16-PCIeXOVR adapter board available on Arria V GX Starter board and without adapter on Arria10 SoC development board.

Table 1: Example Implementation Statistics (MODE=2)

Family	Example Device	Fmax (MHz)	Logic utilization (ALMs)	Registers	Pin	Block Memory bit[1]	Design Tools
ArriaV GX	5AGXFB3H4F35C4	125	686	1192	-	4,194,304	QuartusII 15.1
Arria10 SX	10AS066N3F40E2SGE2	250	737	1444	-	4,194,304	QuartusII 16.0

Notes:
1) Block Memory bit usage is the resource to implement 512 Kbyte buffer to connect to NVMe IP core in Mode 2

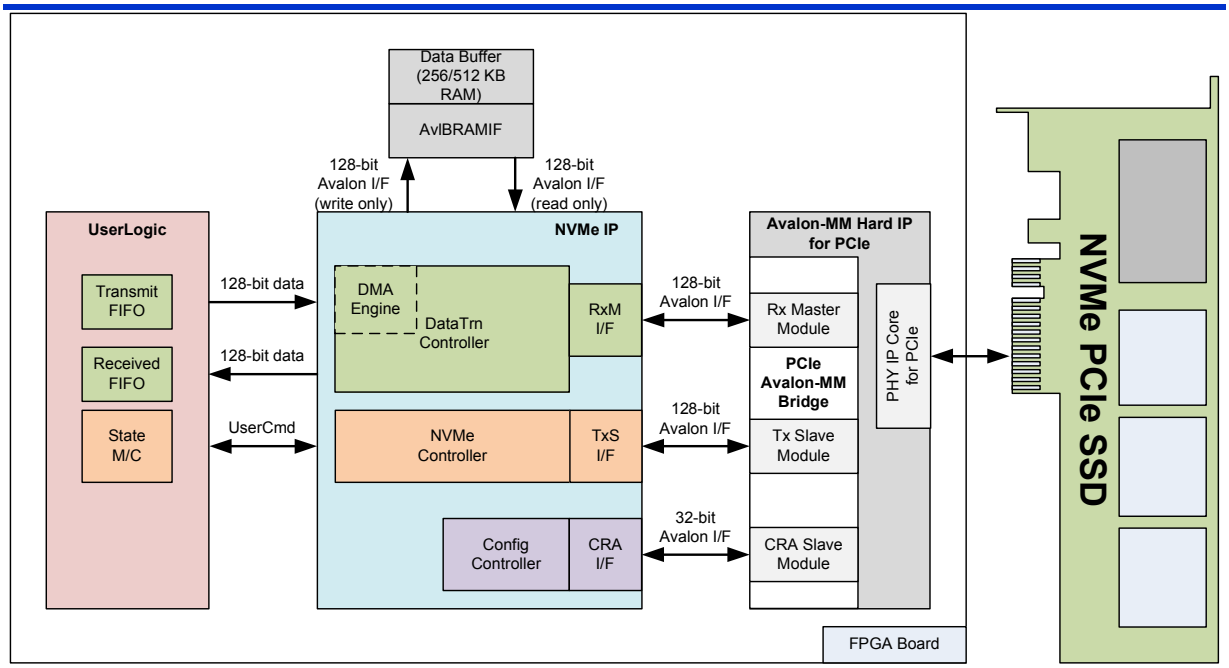

Figure 1: NVMe IP Block Diagram

Applications

NVMe IP Core operating with Avalon MM PCIe Hard IP from Altera is ideal to access NVMe PCIe SSD without CPU or DDR. 256 or 512 Kbyte buffer implemented by Block Memory is used to store data 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 also designed by M.2 storage which uses PCIe protocol standard.

General Description

NVMe IP implements host controller to access NVMe PCIe SSD following NVM express standard. Physical interface of NVMe SSD is PCIe, so Avalon MM PCIe Hard IP from Altera is applied for PCIe protocol. NVMe IP designs the logic to access PCIe and NVMe controller register to support three NVMe commands, i.e. Identify, Write, and Read command. Typically, NVMe SSD includes more than one controller, so data request returned from SSD may not be in sequential format. Big data buffer implemented by Block Memory is required to support random data request from SSD. Two NVMe IPs are provided in release stuff to run with two buffer sizes. First is IP in MODE 1 which is economic mode for using with 256 Kbyte buffer. Another is IP in MODE 2 which is high performance mode for using with 512 Kbyte buffer. The user interface is simple designed by dgIF typeS which requires only command, start address, and transfer length for command interface and general FIFO for data interface. Since there is no asynchronous logic in the IP, clock domain of the IP must use the same clock with output from Avalon MM PCIe Hard IP. Error signal will be asserted with the error status if IP detects the abnormal condition during packet transferring.

The reference design on Altera Development board are available to download and evaluate before IP purchasing

Functional Description

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

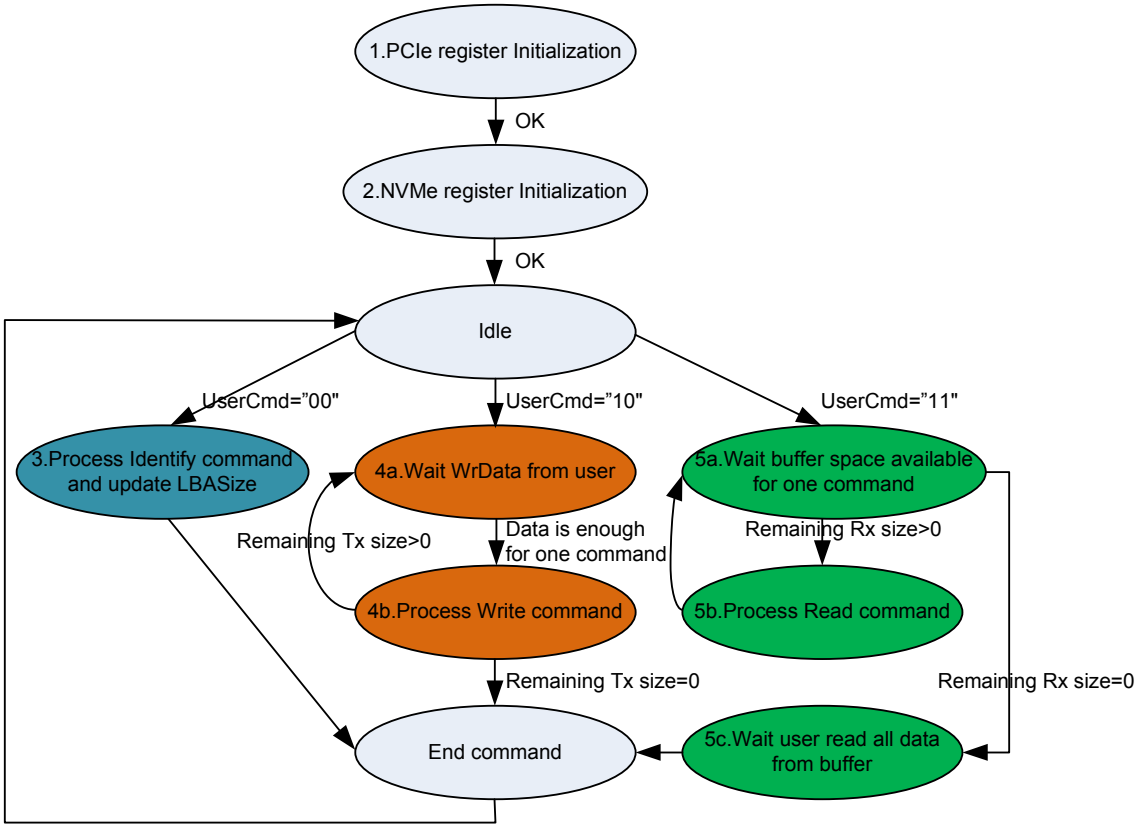

Figure 2: NVMe-IP Operation Flow

1) IP sets Avalon MM PCIe Hard IP register and PCIe configuration space to setup PCIe environment for NVMe operation.
2) IP sets parameter and environment to NVMe controller in SSD. 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 to show disk capacity for user.
4) In case of write command, IP will wait until write data from user in the data buffer enough for transfer size in one command (Maximum transfer size of one command in NVMe IP is 128 Kbyte). Then, IP sends write command to NVMe SSD. IP will go back to Idle state after receiving status from SSD that data of all commands have been transferred completely.
5) In case of read command, IP will monitor space area in data buffer that is enough for transfer size in one command, and then sends read command to NVMe SSD. IP will go back to Idle state after sending all commands and user reads all data from the data buffer.

From above sequence, NVMe IP Core is split into three groups following the interface of Avalon MM PCIe Hard IP, i.e. configuration, NVMe, and data interface. The details of each group are follows.

Configuration

After system power-on, PCIe root complex system needs to write and read configuration data with PCIe device following PCIe standard. The sequence to program configuration data to the device is designed within this block.

- **Config Controller**

  This module controls the initialization sequence for PCIe device such as checking PCIe device class, setting BAR address, and enable master mode. Most sequences need to generate configuration write/read packet to Avalon-MM PCIe Hard IP. To send one packet, it needs to access many registers within Avalon-MM PCIe Hard IP. The sequence to generate configuration packet is desigend in CRA I/F.

- **CRA I/F**

  This module is used for creating TLP packets such as configuration write/read through Avalon-MM PCIe Hard IP. It constructs the packet from address and data input value from Config Controller for write packet, and it decodes the read data from the read packet to send to Config Controller.

NVMe

This block is the main controller of the IP. After PCIe initialization process completes, this block will set parameter to SSD following NVMe standard. To control write/read operation, it prepares write/read command packet and then sets doorbell register to send new command request to the device. Also, completion packet will be monitored and then sets doorbell register to flush completion packet from the device. The memory address to store command/status/data are defined to command packet and register by this block.

- **NVMe Controller**

  This block has two functions, i.e. NVMe initialization and User command processing. After system boot-up, NVMe register will be initialized by this module. This process will run only one time. After that, it will wait new command from user. To process command, this module will decode user inputs and then prepare command parameter such as Command ID, Opcode, and Data pointer. If total length is more than 128 Kbyte size, the controller will generate more than one command to SSD. Before setting doorbell register to start new command request, data buffer status must be monitored that available data is at least 128 Kbyte in case of 128-Kbyte write command or monitored that space data is more than 128 Kbyte in case of 128-Kbyte read command. Status within completion packet returned from the device will be monitored to check error status. Busy signal output to user will be cleared after the last status packet is returned from SSD to confirm that all data have been transferred completely for write command or after user reads all data from the data buffer for read command.

- **TxS I/F**

  NVMe register is mapped to BAR0/BAR1 of PCIe device which can be accessed through 128-bit Avalon bus standard. Only 32-bit size with single access is used to access NVMe register through TxS port of Avalon-MM PCIe Hard IP. Similar to CRA I/F, this module is designed in master mode.

Data

This block is designed to decode the address request from Avalon-MM PCIe Hard IP and select data source/destination of each request. User data from FIFO does not transfer to Avalon-MM PCIe Hard IP directly, but transferring through data buffer. DMA Engine is designed to transfer data between data buffer and UserFIFO through 128-bit Avalon bus.

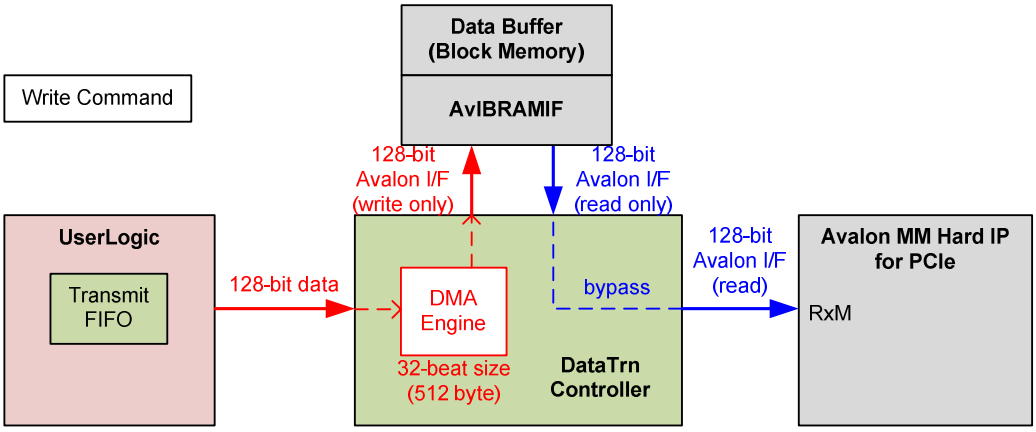

Figure 3: Data Flow when Write command

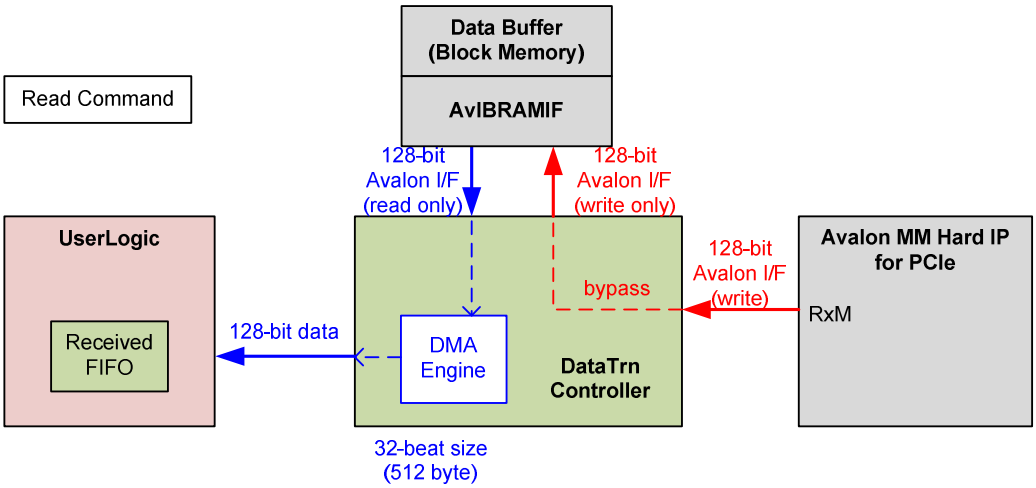

Figure 4: Data flow when Read command

- **DataTrn Controller**

  As shown in Figure 3 and Figure 4, Data buffer must support two 128-bit Avalon-MM masters, one is write only port and another is read only port. Both connections will connect to internal DMA engine or RxM of Avalon-MM PCIe Hard IP depending on command direction. For internal DMA engine, burst size of each request is fixed to be 32-beat or 512-byte. To increase performance, read request supports pipelined transfer which can send many read commands without waiting data returned from Data buffer. So, the adapter logic between Block Memory and NVMe-IP needs to support pipeline transfer for high performance.

For control/status data, data type will be decoded from address in the request and then the packet will be processed. Data type from RxM write request can be Identify data, Admin completion data, or IO completion data. Status value within each completion data will be monitored to confirm that no error is found for each transfer. Command ID value within IO completion data is used to confirm which command has been completed.

For RxM read request, data type can be PRP List, Admin submission, or IO submission which are sent from the logic inside NVMe controller module.

- **RxM I/F**

  This is the slave side of 128-bit Avalon bus. When access data area, it will bypass RxM interface to data buffer interface. When access non-data area, Avalon interface will be converted to RAM interface for easily processing.

User Logic

Simple logic to send command, address, and size can be designed. Data can transfer by FIFO interface.

Avalon-MM PCIe Hard IP

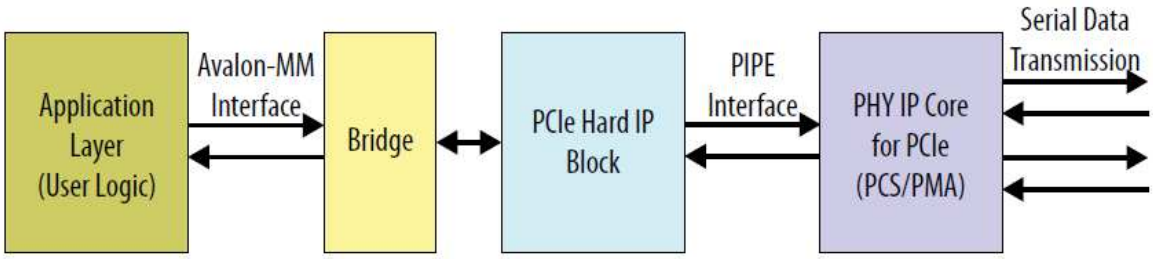

Figure 5: Architechture of Avalon-MM PCIe Hard IP

The Hard IP for PCIe IP core using the Avalon-MM interface removes some of the complexities of the PCIe protocol. For example, it handles all of the Transaction Layer Protocol (TLP) encoding and decoding. The IP includes PCIe Hard IP Block, so maximum numbers of NVMe IP for connectiong many SSDs in one FPGA device will be limited by the numbers of PCIe Hard IP Block. More details are described in “ArriaV Avalon-MM Interface for PCIe Solutions” or “Arria10 Avalon-MM Interface for PCIe Solutions”.
https://www.altera.com/en_US/pdfs/literature/ug/ug_a10_pcie_avmm.pdf
https://www.altera.com/en_US/pdfs/literature/ug/ug_a5_pcie_avmm.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
User Interface		
RstB	In	Reset signal. Active low. Release this signal when Clk signal input is stable.
Clk	In	Clock output from Avalon-MM PCIe Hard IP to synchronous with Avalon bus interface. 125 MHz for PCIe Gen2 and 250 MHz for PCIe Gen3.
dgIF typeS		
UserCmd[1:0]	In	User Command. “00”: Identify command, “10”: Write PCIe SSD, ”11”: Read PCIe SSD.
UserAddr[47:0]	In	Start address of PCIe SSD to write/read in sector unit (512 byte).
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 device 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. [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. [31:6] - Reserved Note: Timeout period of bit[2]/[4] is set from TimeOutSet input.
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 data available size in FIFO. If total 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 after UserFifoRdEn asserted about one clock period.

Signal	Dir	Description
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 Clk domain (8 ns for Gen2 or 4 ns for Gen3).
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	PCI linkup status output from Avalon-MM PCIe Hard IP.
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.
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.
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.
CRA I/F		
CraChipSel	Out	Assert to select CRA port.
CraAddress[13:0]	Out	Write/Read byte address. Bit[1:0] is always equal to 00b for 32-bit access.
CraByteEnable[3:0]	Out	Byte enable.
CraRead	Out	Read request. Indicate that valid read address is available.
CraReadData[31:0]	In	Read data.
CraWrite	Out	Write request. Inidicate that valid write address, write data, and byte enable are available.
CraWriteData[31:0]	Out	Write data.
CraWtRequest	In	Wait Request to hold off more requests.
TxS I/F		
TxSChipselect	Out	Assert to select TX Slave port.
TxSByteEnable[15:0]	Out	Byte enable for write data.
TxSReadData[127:0]	In	Read data return from PCIe Hard IP.
TxSWriteData[127:0]	Out	Write data to TX Slave port.
TxSRead	Out	Issue read request.
TxSWrite	Out	Issue write request.
TxSBurstCount[5:0]	Out	Indicate the amount of data requested. Always set to 000001b for single access.
TxSReadDataValid	In	Indicate that valid read data is available.
TxSWaitRequest	In	Hold off read or write data.
TxSAddress[28:0]	Out	Write/Read address.

Signal	Dir	Description
RxM I/F		
RxMAddress[31:0]	In	Write/Read address to access.
RxMBurstCount[5:0]	In	Gives the exact number of the burst count for write/read request.
RxMByteEnable[15:0]	In	Byte enable for write data.
RxMWrite	In	Write request sent from PCIe Hard IP.
RxMRead	In	Read request sent from PCIe Hard IP
RxMWaitRequest	Out	Assert to hold data transfer.
RxMWriteData[127:0]	In	Write data.
RxMReadData[127:0]	Out	Read data.
RxMReadDataValid	Out	Indicate that read data is valid.
Avalon-MM Master I/F for Data buffer		
BufAvwAddress[31:0]	Out	Write address to access.
BufAvwBurstCount[5:0]	Out	Gives the exact number of the burst count for write request.
BufAvwByteEnable[15:0]	Out	Byte enable for write data.
BufAvwWrite	Out	Write request sent to data buffer.
BufAvwWaitRequest	In	Assert to hold data write transfer.
BufAvwWriteData[127:0]	Out	Write data.
BufAvrAddress[31:0]	Out	Read address to access.
BufAvrBurstCount[5:0]	Out	Gives the exact number of the burst count for read request.
BufAvrByteEnable[15:0]	Out	Byte enable for read data. Always set to 0xFFFF.
BufAvrRead	Out	Read request sent to data buffer.
BufAvrWaitRequest	In	Assert to hold read data transfer.
BufAvrReadData[127:0]	In	Read data.
BufAvrReadDataValid	In	Indicate that read data is valid.

Timing Diagram

Initialization

Clk and RstB input signal of the IP are generated from Avalon-MM PCIe Hard IP. After RstB is released, the IP will initialize PCIe configuration register and set NVMe register of PCIe SSD. UserBusy is de-asserted to '0' after both initialization sequences run completely.

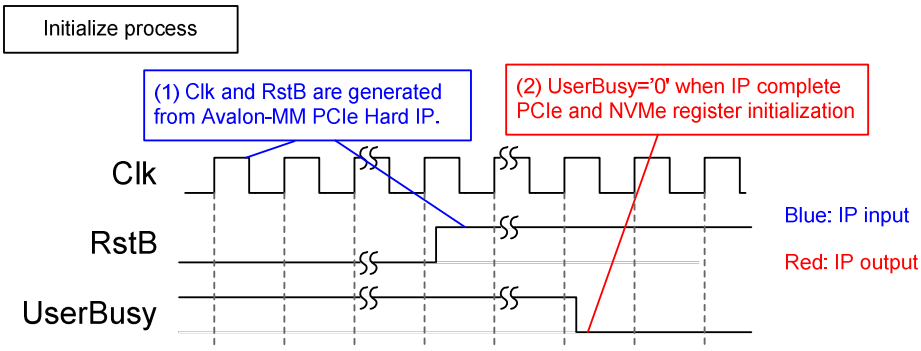

Figure 6: RstB and UserBusy after system boot-up

dgIF typeS

dgIF typeS signal can be split into two interfaces, i.e. command interface and data interface. Figure 7 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 start the command operation. So, UserReq can be cleared and user logic can prepare the next command to the command bus.

Note: UserAddr and UserLen value may be ignored in some commands such as Identify command.

For data interface, transmit FIFO will be read for Write command while received FIFO will be written for Read command. Timing diagram of data interface is shown in Figure 8 and Figure 9.

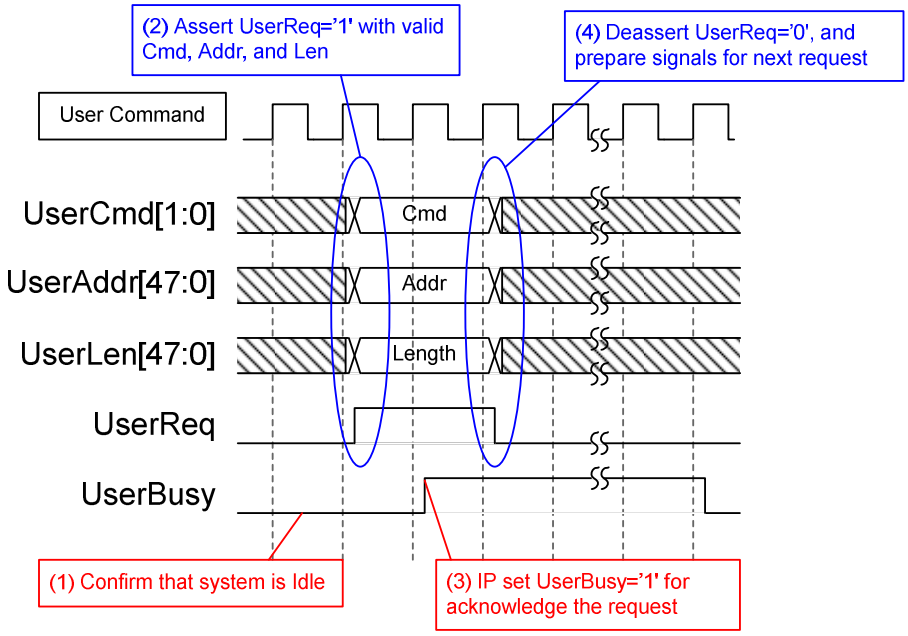

Figure 7: User Command Interface Timing diagram

For write command, data from Transmit FIFO will be forwarded to data buffer through 128-bit Avalon-MM bus interface. Burst size to data buffer is fixed to 32-beat or 512-byte. UserFifoRdCnt will be monitored to confirm that at least 512-byte is available in Transmit FIFO before starting each burst transfer. As shown in Figure 8, UserFifoRdData must be valid after UserFifoRdEn is asserted for 1 clock period and hold the same value until next read enable is received.

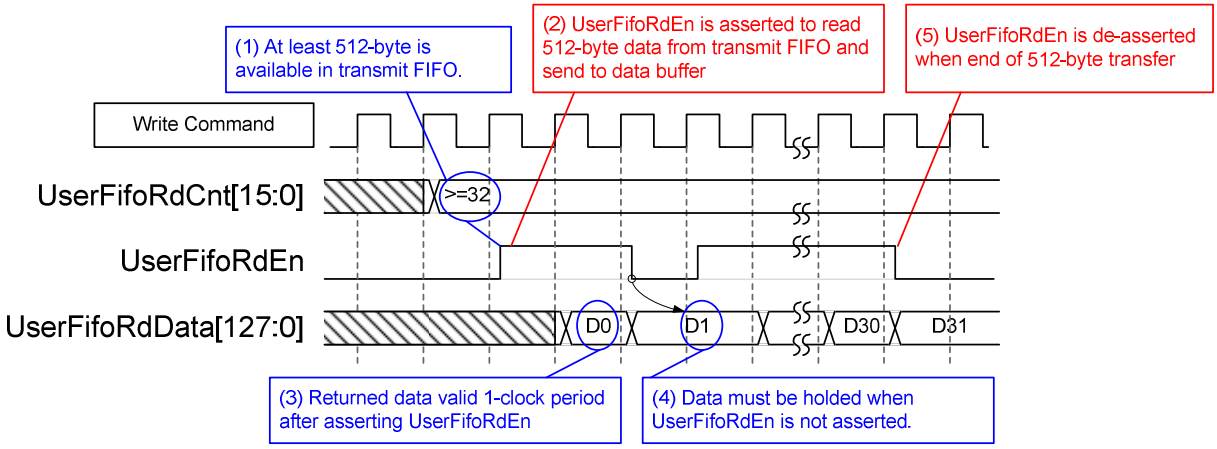

Figure 8: Transmit FIFO Interface for Write command

For read command, UserFifoWrEn will be 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 at least 1024-byte space area is available in Received FIFO before transferring 512-byte data to FIFO.

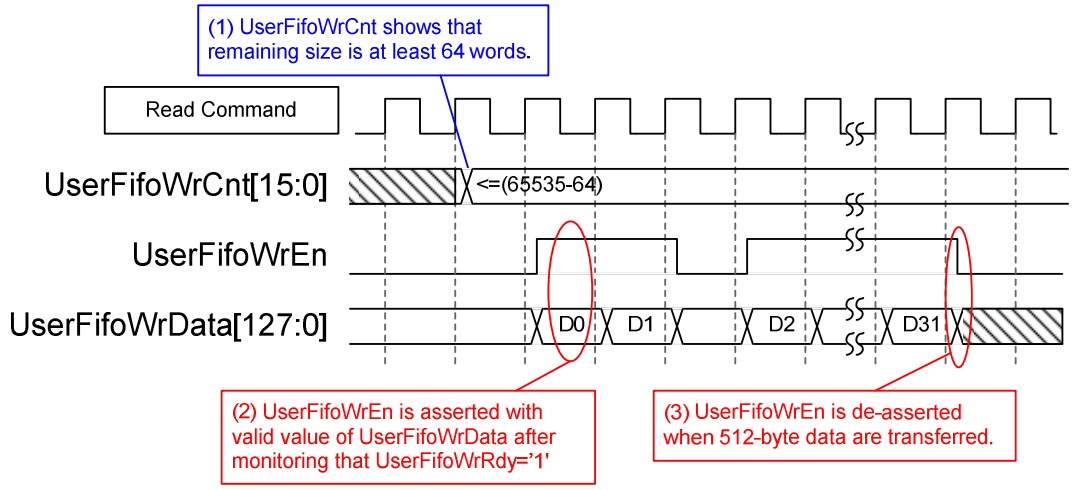

Figure 9: Received FIFO Interface for Read command

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.

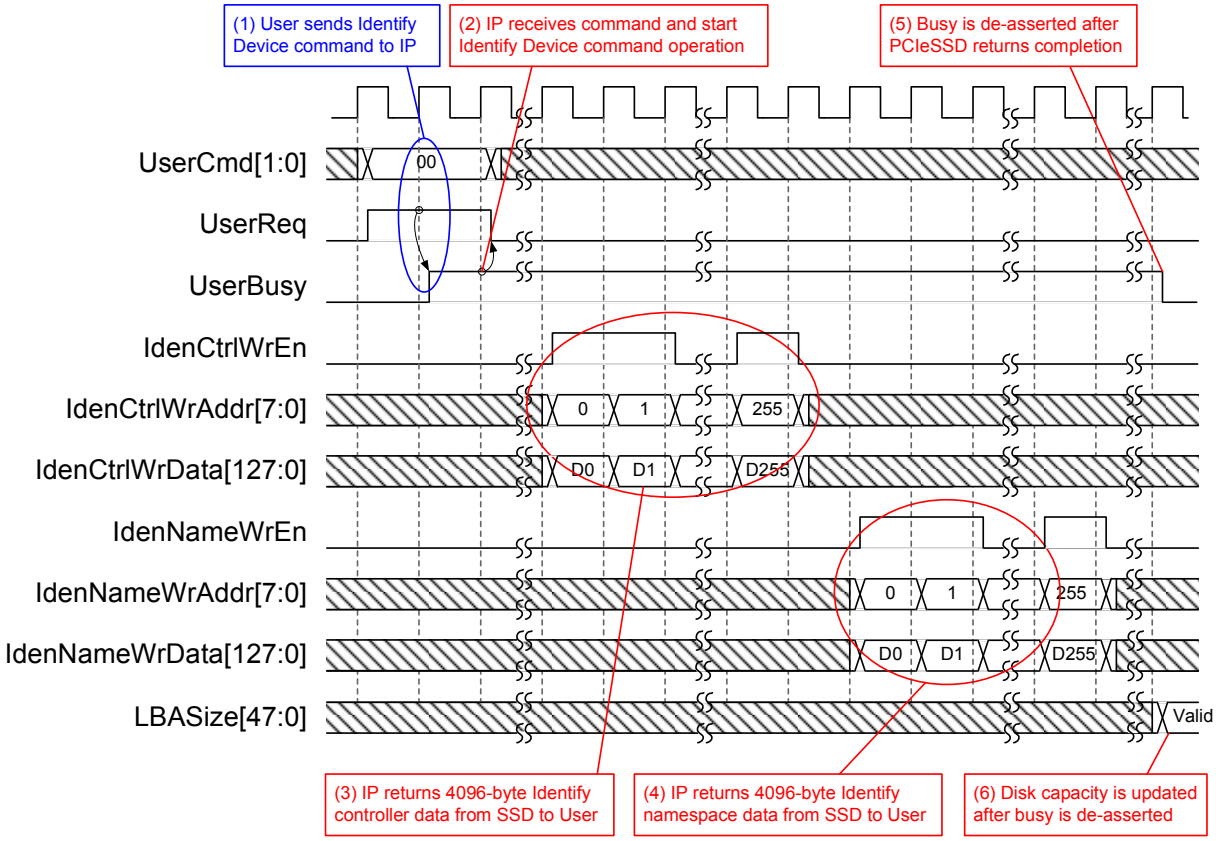

Figure 10: LBASize update after Identify Device command

As shown in Figure 10, UserCmd and UserReq are set when UserBusy='0'. UserAddr and UserLen input are not required for Identify command. After sending request for Identify command, 4096-byte Identify Controller data and 4096-byte Identify Namespace data will be sent out. Both Identify data will not be sent out continously. Data will be split in many burst transfers depending on SSD characteristic. Finally, LBASize will be updated when UserBusy is de-asserted from Identify command.

Error

During normal operation, UserError and all bits of UserErrorType signal will be always 0. UserError is generated by OR condition of each-bit of UserErrorType. If any bit of UserErrorType is set to ‘1’, UserError will be asserted and latched until RstB is asserted to ‘0’, as shown in Figure 11.

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.

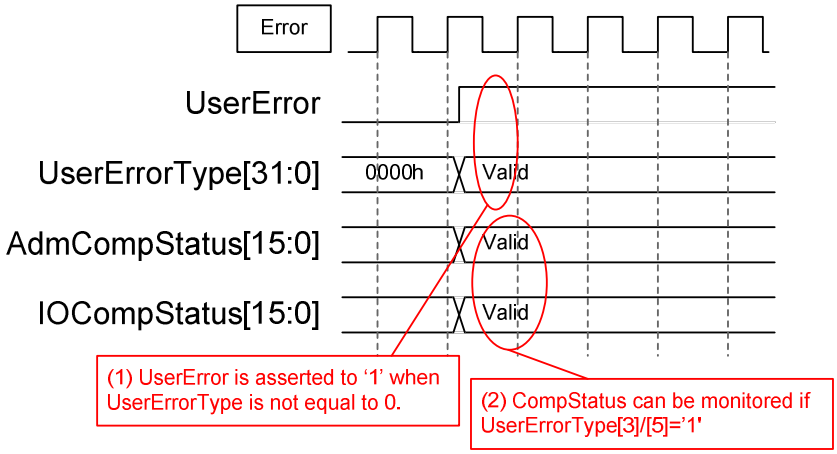

Figure 11: Error flag Timing diagram

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.

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