NVMe IP Core

November 23, 2018 Product Specification Rev3.1

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, Alaric board
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- Implement application layer to access NVMe 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 six commands, i.e. Identify, Shutdown, Write, Read, SMART, and Flush
- Additional command support as optional
- 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)
  - LBA unit: 512 byte or 4096 byte
- Reference design
  - Arria V GX Starter board, Arria10 GX development board, and Alaric board with AB16-PCIeXOVR adapter board
  - 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	1890	3780	-	2,162,688	QuartusII 16.0
Stratix V GX	5SGXMA7N2F45C2	300	1869	3826	-	2,162,688	QuartusII 16.0
Arria10 SX	10AS066N3F40E2SGE2	300	1862	3869	-	2,162,688	QuartusII 16.0

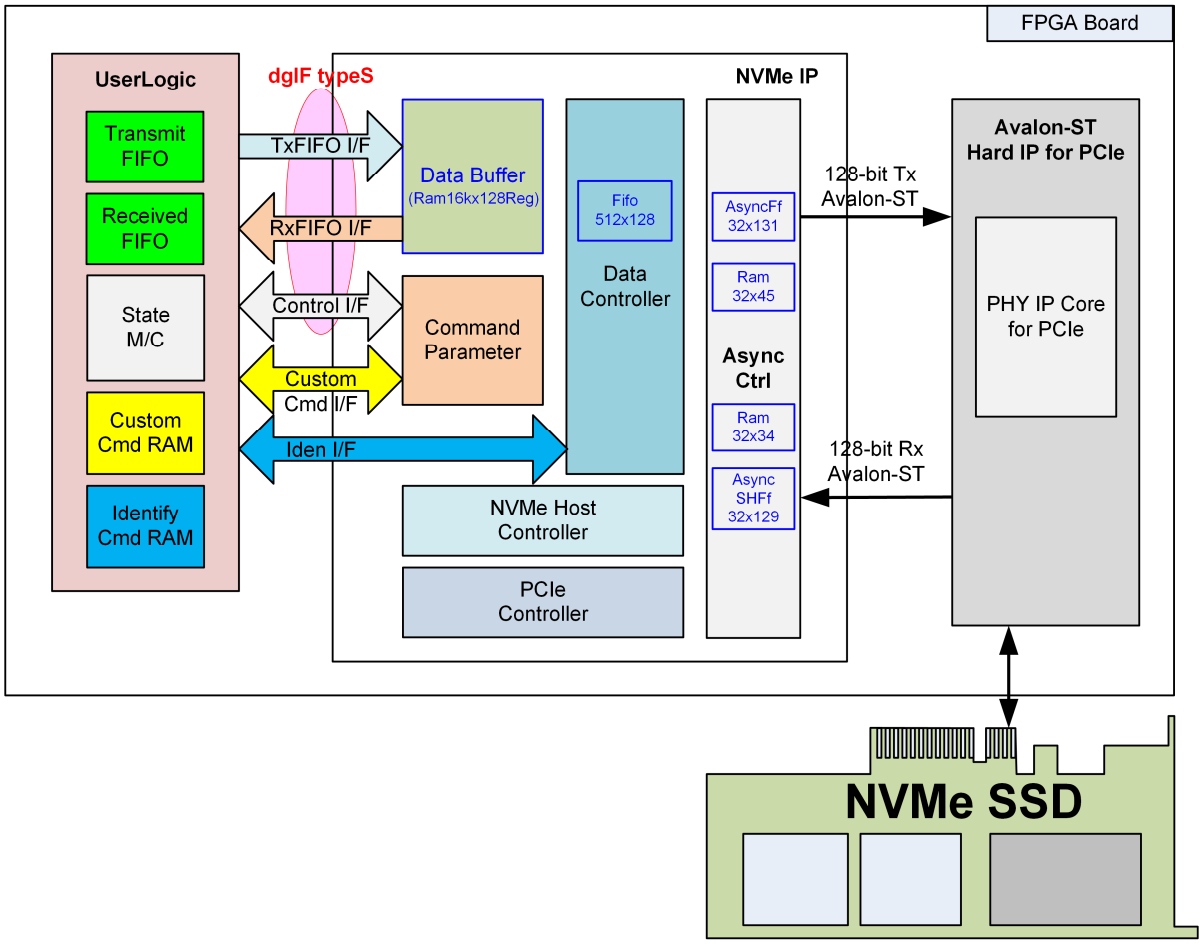

Figure 1: NVMe IP Block Diagram

Applications

NVMe IP Core integrated with Avalon-ST PCIe Hard IP from Intel is ideal to access NVMe SSD without CPU or DDR. 256 Kbyte buffer implemented by Block Memory is included in NVMe IP Core to be data buffer 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 package.

General Description

NVMe IP implements as host controller to access NVMe 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 six NVMe commands, i.e. Identify, Shutdown, Write, Read, SMART, and Flush command.

NVMe IP includes many small buffers implementing as RAM and FIFO, as shown in blue color of Figure 1. 256 Kbyte RAM (Ram16kx128Reg) is also included to be data buffer between User Logic and NVMe SSD when operating Write or Read command. By using big data buffer size, the system achieves good performance 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.

To support SMART and Flush command, Custom command interface is additional designed to receive command parameters. Also, SMART data is tranferred through Custom command RAM interface.

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 to ‘1’ with the error status if IP detects the abnormal condition during packet transferring.

The reference design on FPGA development boards are available to evaluate before purchasing.

Functional Description

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

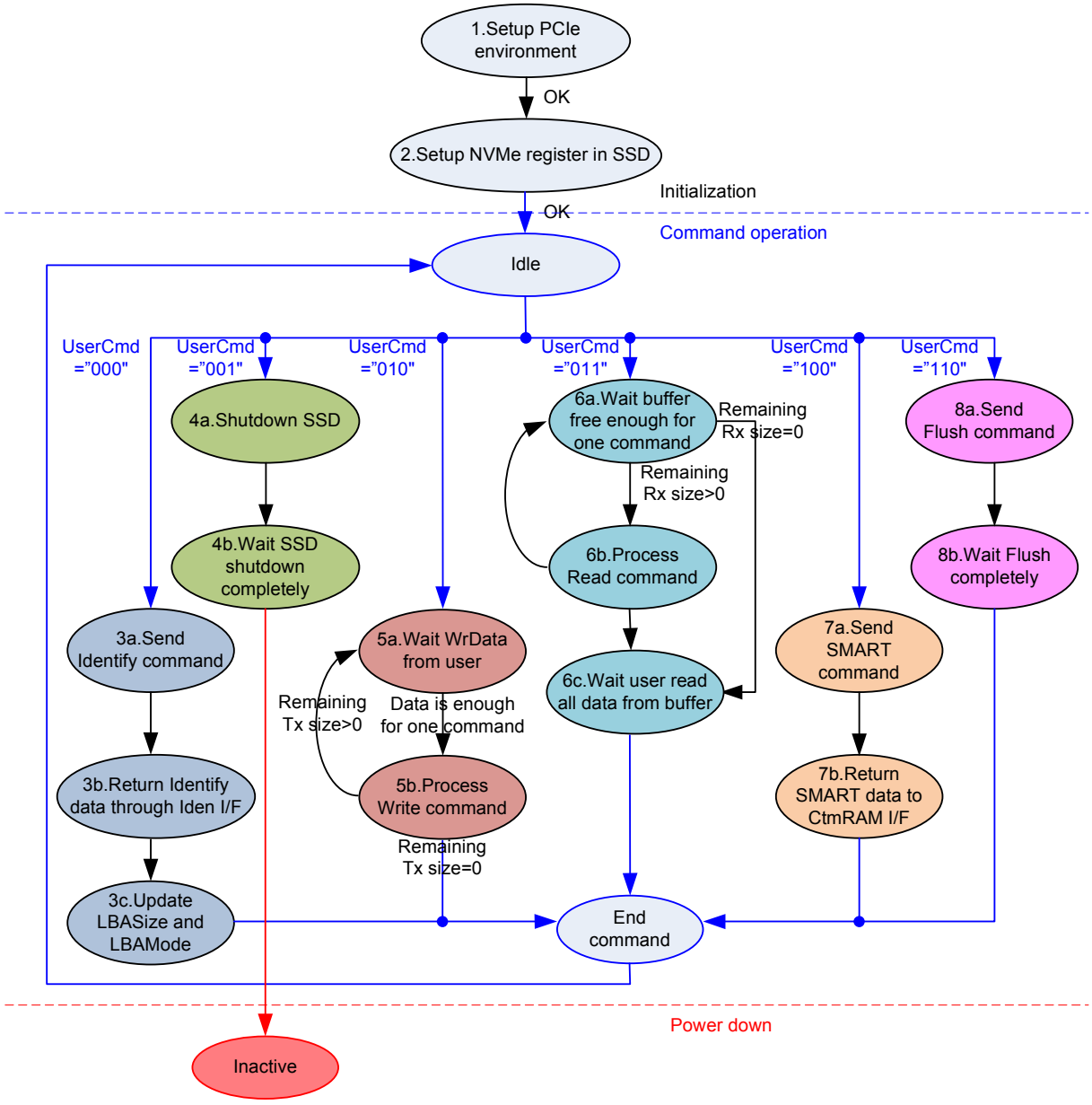

Figure 2: NVMe IP Operation Flow

The operation of NVMe IP is described as follows.

1) IP sets up PCIe environment for NVMe operation.
2) IP 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 (UserCmd=”000”) to update LBASize (disk capacity) and LBAMode (LBA unit=512 byte or 4 Kbyte).
4) The last command before power down the system must be Shutdown command (UserCmd=”001”). This command is used to shutdown SSD in good sequence. Without Shutdown command, Write data in SSD cannot be guaranteed. After running Shutdown command, NVMe IP and SSD are Inactive state.
5) In case of Write command (UserCmd=”010”),
   - 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.
6) In case of Read command (UserCmd=”011”),
   - 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.
7) In case of SMART command (UserCmd=”100”),
   - IP sends Get Log Page command to read SMART/Health information from SSD.
   - 512-byte data is returned from SSD and forwarded to user through Custom Command RAM interface (CtmRamAddr=0x000 – 0x01F).
8) In case of Flush command (UserCmd=”110”),
   - IP sends Flush command to SSD.
   - IP waits until SSD returns status to complete the operation.

From above sequence, 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 is designed to check PCIe device class, to set BAR address, and to enable master mode. The essential parameters to set up PCIe environment are mapped to configuration space area of SSD which can be accessed through 128-bit Tx/Rx Avalon-ST bus. So, PCIe controller includes the logic to set up configuration space through 128-bit Tx/Rx Avalon-ST bus to during initialization process.

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/Shutdown/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.

For SMART/Flush command, Submission queue entry is prepared by user through Custom command interface. NVMe block still monitors error status within Completion queue entry like other commands. For more information, Completion queue entry of SMART/Flush command is also mapped to be output signal through Custom command interface. Data output from SMART command is transferred to user through Custom command RAM 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, Identify interface for Identify command, and Custom command RAM interface for SMART 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, Shutdown, Write, and Read command, Command parameter module creates the value of all 16 dwords of Submission queue entry following Command interface of dgIF typeS. When running SMART and Flush command, 16 dwords of Submission queue entry are loaded from user input through Custom command interface.

  Completion queue entry size is 4 dwords. The value of Completion queue entry is monitored by Command parameter for all commands. For SMART and Flush command, 4 dwords of Completion queue entry are latched to be output of Custom command interface.

- **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, SMART 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.

- **AsyncCtrl**

  NVMe IP is designed to run user logic in another clock domain which is not PCIe clock. The limitation is that User clock frequency must be higher than or equal to PCIe clock because transmit packet to Avalon-ST Hard IP for PCIe must be always ready between start of frame and end of frame. Using low frequency, data packet in each frame could not be sent to Avalon-ST Hard IP for PCIe continuously.

Buffer

NVMe IP uses many buffers which are created by IP catalog inside Quartus tools. As shown in Figure 1, there are six buffers in NVMe IP which are described in more details as follows. The recommended setting of each buffer are shown in NVMe IP reference design.

- **Ram16kx128Reg (Data Buffer)**

  This buffer is used to store user data for Write/Read command. It is two-port RAMs which has one read port and one write port. RAM size is 16kx128-bit and implemented by Block Memory with byte enable option. Read output port includes register, so read data latency is equal to two clock cycles.

- **Fifo512x128**
  This buffer is used within data controller to store data returning to SSD. Write and Read port use same clock domain (synchronous FIFO). Reset signal in FIFO is asynchronous type. Data counter, full, and empty signal outpus are required. FIFO size is 512x128-bit and implemented by Block Memory.

- **AsyncFf32x131**
  This buffer is asynchronous FIFO (Write and Read port run in different clock domain). FIFO size is 32x131-bit. Similar to Fifo512x128, data counter, full, amd empty status are required. Reset signal is asynchronous reset type. To reduce resource, these FIFOs are implemented by using MLAB instead of Block Memory.

- **AsyncSHFf32x129**
  Similar to AsyncFf32x131, this buffer is asynchronous FIFO and implemented by using MLAB. FIFO size is 32x129-bit. The different setting is Rdreq option which uses show-ahead mode instead of normal mode.

- **Ram32x45/Ram32x34**
  These buffers are two-port RAMs which has one read port and one write port. RAM size is 32x45-bit/32x34-bit. Read input ports include register, so read data latency is equal to one clock cycle. To optimize resource, these buffers are implemented by using MLAB instead of Block Memory.

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. Data output from SMART and Identify command are connected to simple dual port RAM with byte enable. Data size returning from Identify command is 8 Kbyte while data size from SMART command is 512 byte. Bus width of all data interface are 128-bit.

Avalon-ST PCIe Hard IP

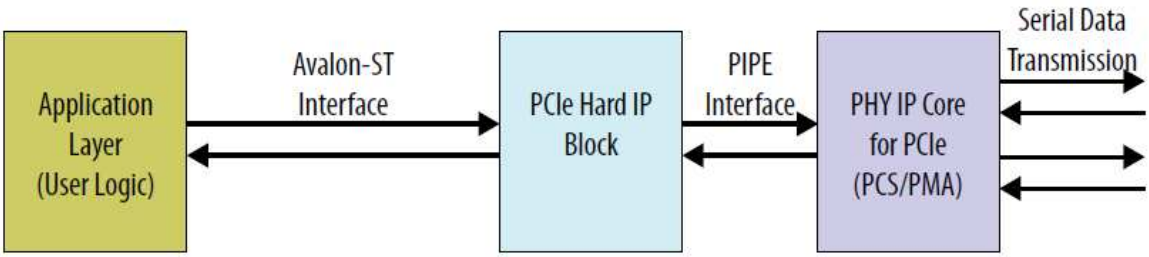

Figure 3: Architecture of Avalon-ST PCIe Hard IP

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 number of SSDs connecting to one FPGA device is limited by the numbers of PCIe Hard IP Block. More details of Avalon-ST PCIe Hard IP are described in following document.

ArriaV Avalon-ST Interface for PCIe Solutions User Guide
https://www.altera.com/en_US/pdfs/literature/ug/ug_a5_pcie_avst.pdf
Stratix V Avalon-ST Interface for PCIe Solutions User Guide
https://www.altera.com/en_US/pdfs/literature/ug/ug_s5_pcie_avst.pdf
Intel Arria10 Avalon-ST Interface for PCIe Solutions User Guide
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
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 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[2:0]	In	User Command ("000": Identify, "001": Shutdown, "010": Write SSD, "011": Read SSD, "100": SMART, "110": Flush, "101"/"111": Reserved)
UserAddr[47:0]	In	Start address to write/read SSD in 512-byte unit. In case LBA unit = 4 Kbyte, UserAddr[2:0] must be always set to "000" to align 4 Kbyte unit. In case LBA unit = 512 byte, it is recommended to set UserAddr[2:0]="000" to align 4 Kbyte size (SSD page size). Write/Read performance of most SSDs is reduced when start address is not aligned to page size.
UserLen[47:0]	In	Total transfer size in the request in 512-byte unit. Valid from 1 to (LBASize-UserAddr). In case LBA unit = 4 Kbyte, UserLen[2:0] must be always set to "000" to align 4 Kbyte unit.
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, CtmSubmDW0-DW15) must be valid and stable during UserReq='1'. UserAddr and UserLen are inputs for Write/Read command while CtmSubmDW0-DW15 are inputs for SMART/Flush command.
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.
LBAMode	Out	LBA unit size ('0': 512byte, '1': 4 Kbyte). Default value is 0. This value is valid after user sends Identify command.
UserError	Out	Error flag. Assert to '1' when UserErrorType is not equal to 0. The flag is de-asserted to '0' by asserting RstB to '0'.
UserErrorType[31:0]	Out	Error status. [0] – Error when PCIe class code is not correct. [1] – Error from CAP (Controller capabilities) register which may be caused from - MPSMIN (Memory Page Size Minimum) is not equal to 0. - NVM command set flag (bit 37 of CAP register) is not set to 1. - DSTRD (Doorbell Stride) is not 0. - MQES (Maximum Queue Entries Supported) is more than or equal to 7. More details of each register can be checked from NVMeCAPReg signal. [2] – Error when Admin completion entry is not received until timeout. [3] – Error when status register in Admin completion entry is not 0 or phase tag/command ID is invalid. Please see more details from AdmCompStatus signal. [4] – Error when IO completion entry is not received until timeout. [5] – Error when status register in IO completion entry is not 0 or phase tag is invalid. Please see more details from IOCompStatus signal.

Signal	Dir	Description
Control I/F of dgIF typeS (Synchronous to Clk)		
UserErrorType[31:0]	Out	[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. [15:10] – Reserved [16]- Error from unsupport LBA unit (LBA unit is not equal to 512 byte or 4 Kbyte) [31:17] – 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 to check 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 to 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 size stored 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. This signal is not used.
UserFifoRdEn	Out	Assert to '1' to read data from 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)		
IPVesion[31:0]	Out	IP version number
TestPin[31:0]	Out	Reserved to be IP Test point.
TimeOutSet[31:0]	In	Timeout value to wait completion from SSD. Time unit is equal to 1/(Clk frequency). If TimeOutSet is equal to 0, Timeout will be disabled.
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

Signal	Dir	Description
IdenWrEn	Out	Assert to '1' when IdenWrAddr and IdenWrData are valid on the bus.
IdenWrDWEn[3:0]	Out	Data enable in Dword (32-bit) unit of IdenWrData. Valid at the same clock as IdenWrEn='1'. Bit[0] is enable of IdenWrData[31:0]. Bit[1] is enable of IdenWrData[63:32]. Bit[2] is enable of IdenWrData[95:64]. Bit[3] is enable of IdenWrData[127:96].
IdenWrAddr[8:0]	Out	Index of IdenWrData in 128-bit unit. Synchronous to IdenCtrlWrEn. 0x000-0x0FF is 4Kbyte Identify controller data, 0x100-0x1FF is 4Kbyte Identify namespace data.
IdenWrData[127:0]	Out	4Kbyte Identify controller data or Identify namespace data. Synchronous to IdenWrEn.
NVMe IP Custom interface (Synchronous to Clk)		
CtmSubmDW0[31:0] – CtmSubmDW15[31:0]	In	16 Dwords of Submission queue entry for SMART/Flush command. DW0: Command Dword0, DW1: Command Dword1, …, and DW15: Command Dword15. These inputs must be valid and stable when UserReq='1' and UserCmd="100" (SMART) or "110" (Flush).
CtmCompDW0[31:0] – CtmCompDW3[31:0]	Out	4 Dwords of Completion queue entry output from SMART/Flush command. DW0: Completion Dword0, DW1: Completion Dword1, …, and DW3: Completion Dword3
CtmRamWrEn	Out	Assert to '1' when CtmRamAddr and CtmRamWrData are valid on the bus (found in SMART command).
CtmRamWrDWEn[3:0]	Out	Data enable of Dword (32-bit) unit of CtmRamWrData. Valid at the same clock as CtmRamWrEn='1'. Bit[0] is enable of CtmRamWrData[31:0]. Bit[1] is enable of CtmRamWrData [63:32]. Bit[2] is enable of CtmRamWrData [95:64]. Bit[3] is enable of CtmRamWrData [127:96].
CtmRamAddr[8:0]	Out	Index of CtmRamWrData when command returns SMART data. (Optional) Index to request data input through CtmRamRdData to support other custom commands.
CtmRamWrData[127:0]	Out	512-byte data output from SMART command.
CtmRamRdData[127:0]	In	(Optional) Data input for other custom commands.
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	Assert to '1' to indicate that PCIeRxData is valid. Must be de-assert to '0' within 2 clocks after PCIeRxReady is de-asserted to '0'.
PCIeRxEOP	In	Assert to '1' to indicate that this is the last cycle of the TLP when PCIeRxValid is asserted to '1'.
PCIeRxReady	Out	Assert to '1' to indicate that NVMe IP is ready to accept data.
PCIeRxError	In	Assert to '1' to indicate 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	Assert to '1' to indicate that PCIeTxData is valid when PCIeTxReady is also asserted to '1'.
PCIeTxSOP	Out	Assert to '1' to indicate first cycle of a TLP when asserted together with PCIeTxValid.
PCIeTxEOP	Out	Assert to '1' to indicate 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	Assert to '1' to indicate that Avalon-ST PCIe Hard IP is ready to accept data.
PCIeTxError	Out	Assert to '1' to indicate 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 after Clk is stable.
2) NVMe IP waits until both PCIeRstB and PCIeLinkup are asserted to '1'. PCIeLinkup is asserted to '1' when PCIe Hard IP completes power sequence (LTSSM State is in L0 state). PCIeRstB is de-asserted to '1' after PCIe Hard IP is not in reset state and ready to interface with application layer. PCIeRstB is generated in PCIeClk domain.
3) NVMe IP starts initialization process.
4) 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.

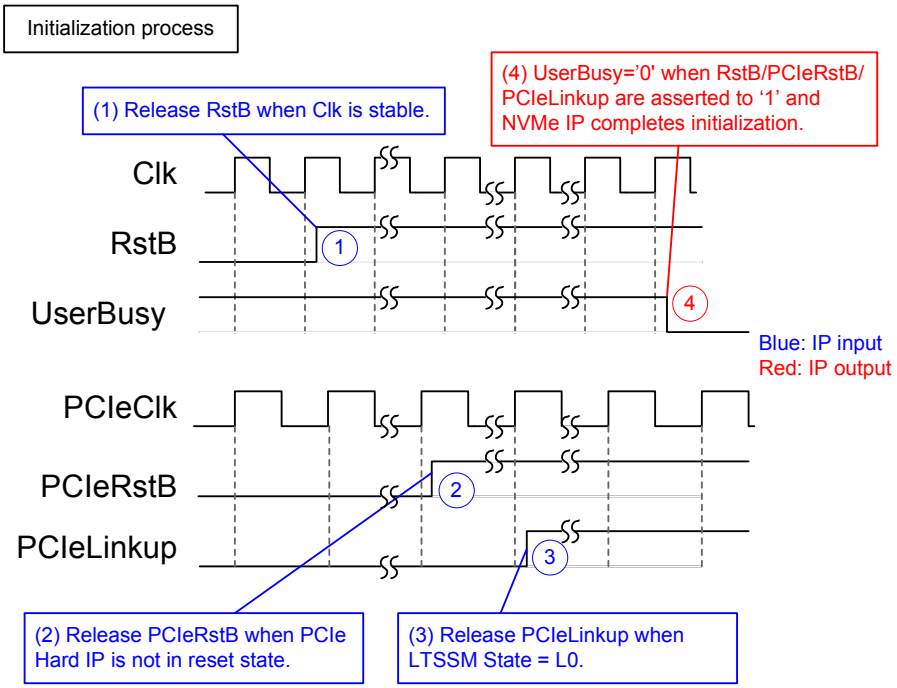

Figure 4: UserBusy after system boot-up

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 equal to '0' to confirm that IP is Idle. Command parameters (UserCmd, UserAddr, UserLen, and CtmSubmDW0-DW15) 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: 1) UserAddr and UserLen value are not used in Identify, Shutdown, SMART, and Flush command.
2) CtmSubmDW0 – DW15 are not used in Identify, Shutdown, Write, and Read command.

For data interface, Transmit FIFO is read for Write command, while Received FIFO is written for Read command. Timing diagram of FIFO interface is shown in Figure 6 and Figure 7.

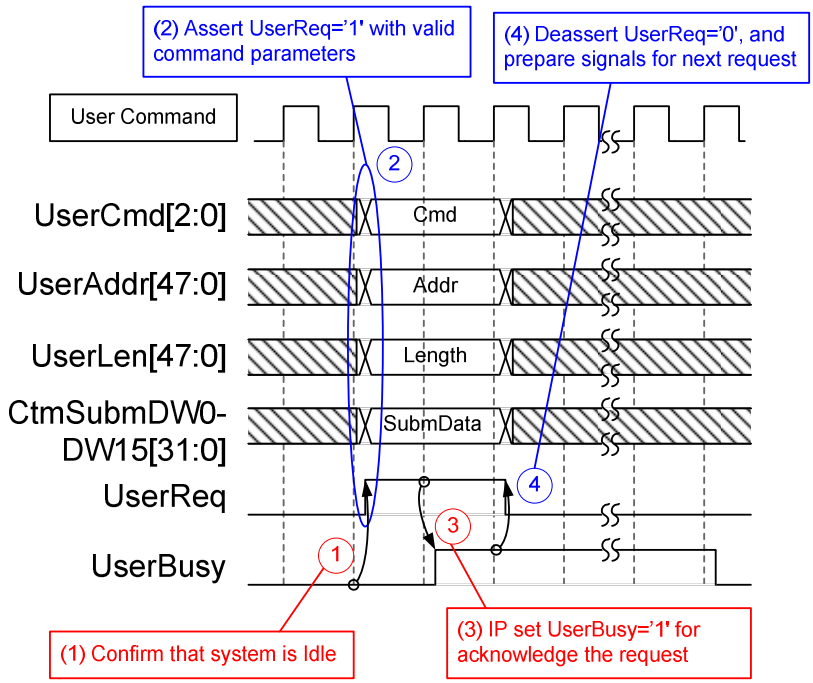

Figure 5: Command Interface of dgIF typeS Timing diagram

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 or more than 512 bytes. After that, UserFifoRdEn is asserted to '1' for 32 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 after UserFifoRdEn is asserted to '1'.

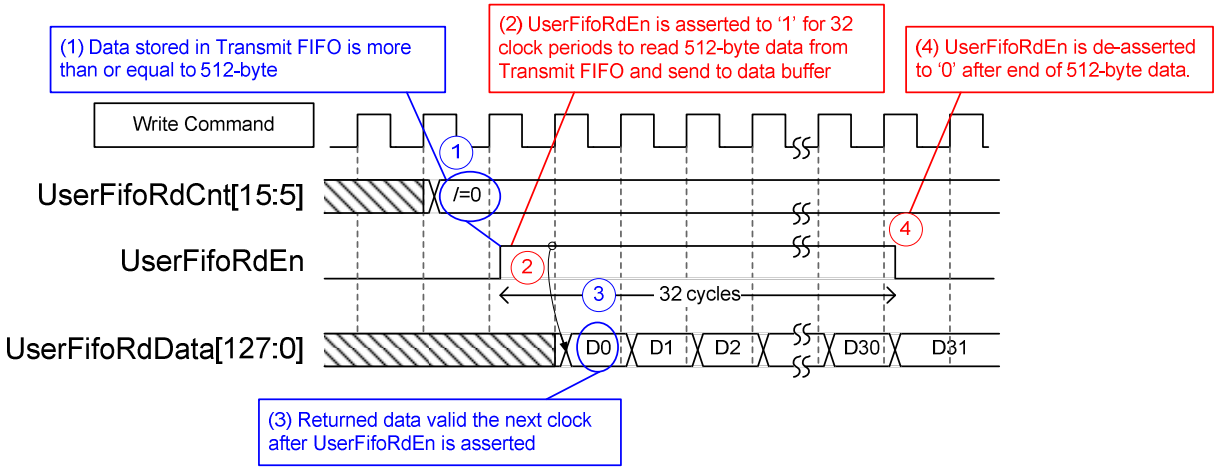

Figure 6: Transmit FIFO Interface for Write command

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 much enough (more than or equal to 1024-byte) before transferring 512-byte data to FIFO.

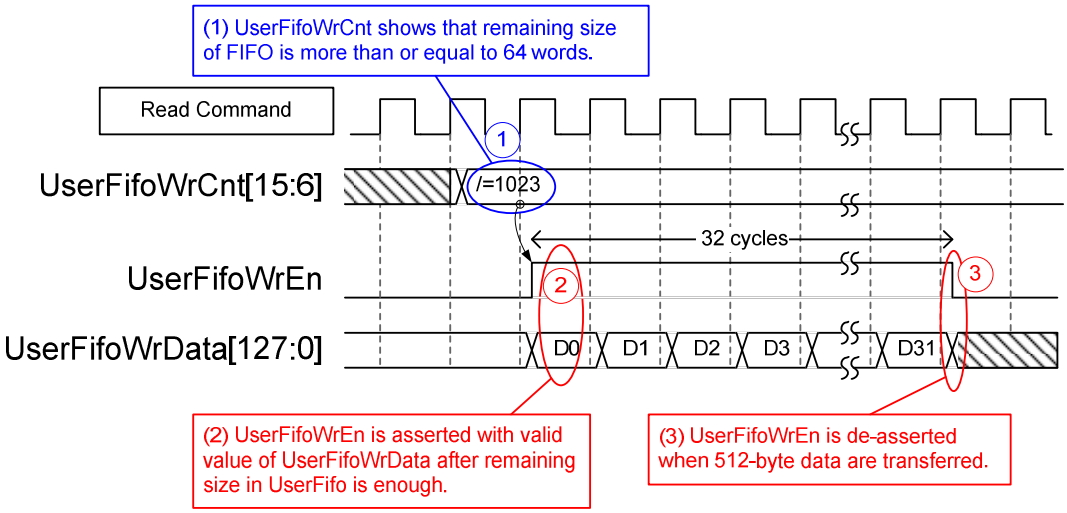

Figure 7: Received FIFO Interface for Read command

IdenCtrl/IdenName

Before sending Write or Read command to IP, user should send Identify command firstly to update LBASize and LBAMode 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. If LBAMode='1' (LBA unit = 4 Kbyte), the address and length input for Write/Read command must be aligned to 4Kbyte.

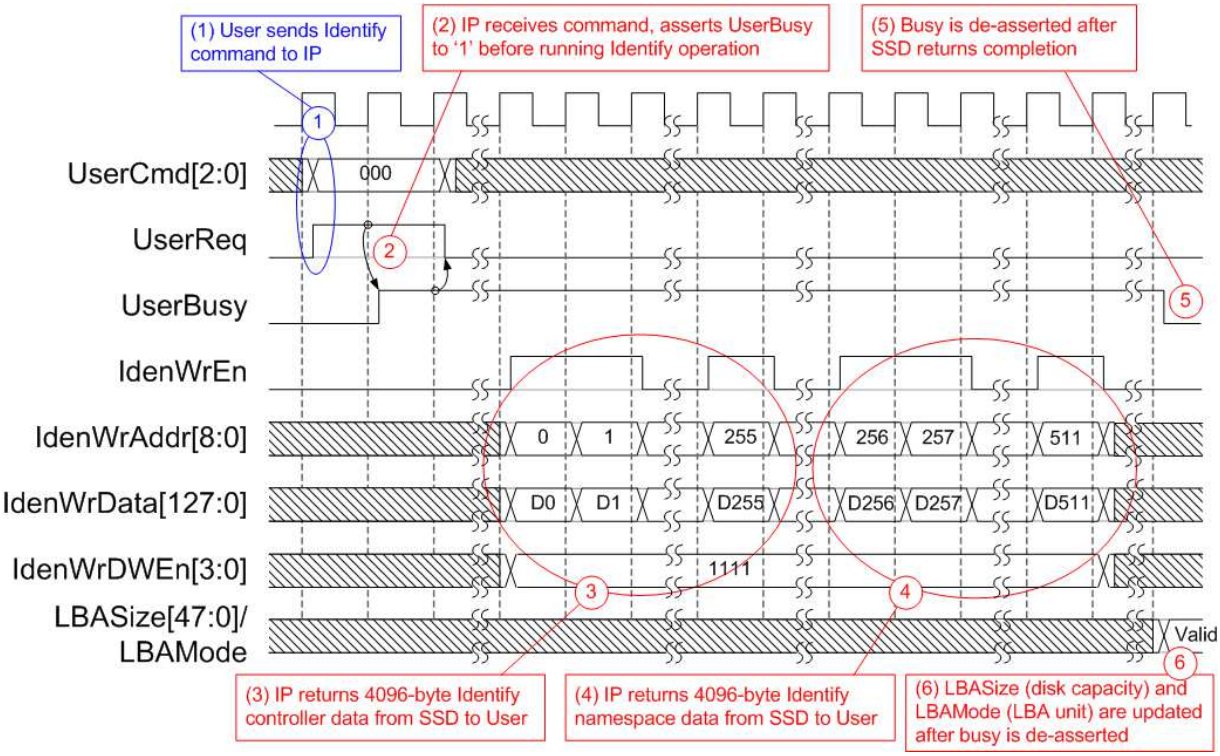

Figure 8: LBASize/LBAMode updated after Identify command

Figure 8 shows timing diagram of Identify command. UserCmd and UserReq are sent out when UserBusy='0'. UserAddr and UserLen input are not used for Identify command. After Identify command is sent to SSD, 4096-byte Identify Controller data and 4096-byte Identify Namespace data are returned. IdenWrAddr is equal to 0 – 255 during sending Identify Controller data and equal to 256 – 511 during sending Identify Namespace data. 8-byte Identify Controller data or Identify Namespace data is valid on IdenWrData bus for each clock cycle, synchronous to IdenWrAddr and IdenWrEn signal. Generally, IdenWrDWEn is equal to "1111" which means all 128-bit of IdenWrData are valid. Finally, LBASize and LBAMode are updated when UserBusy is de-asserted to '0' from Identify command.

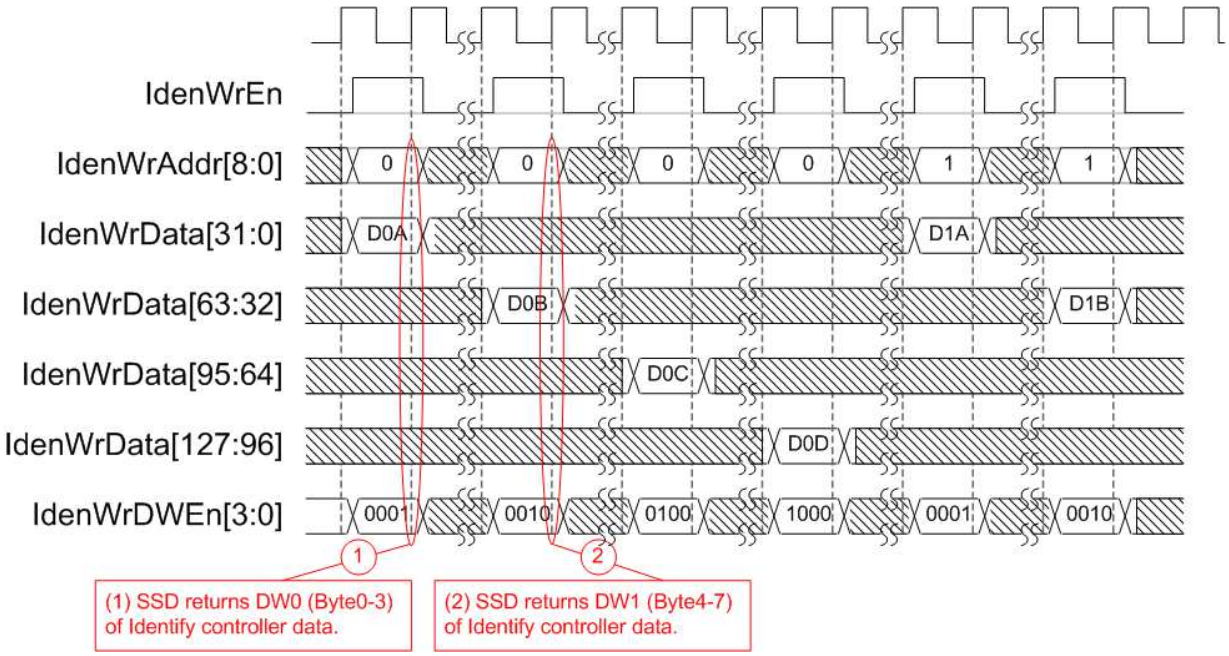

Figure 9 IdenWrDWEn timing diagram

Some SSDs may return Identify controller data or Identify namespace data in 32-bit data format instead of 128-bit data format. So, IdenWrDWEn must be used to check the valid Dword of IdenWrData signal. As shown in Figure 9, bit[0] of IdenWrDWEn is referred to bit[31:0] of IdenWrData, bit[1] is referred to bit[63:32] of IdenWrData, bit[2] is referred to bit[95:64] of IdenWrData, and bit[3] is referred to bit[127:96] of IdenWrData. Similar to IdenWrData, IdenWrDWEn is valid when IdenWrEn='1'.

Shutdown

Before system is power down, it is recommended to send Shutdown command to NVMe IP. After issue this command, SSD shutdowns in good sequence. During shutdown operation, Cache in SSD is flushed to the memory and all write data sending to SSD could be guaranteed. After issue this command, NVMe IP and SSD cannot be used until system is power down.

To send Shutdown command, UserCmd is set to “001”. UserAddr, UserLen, and CtmSubmDW0-DW15 are not used in this command.

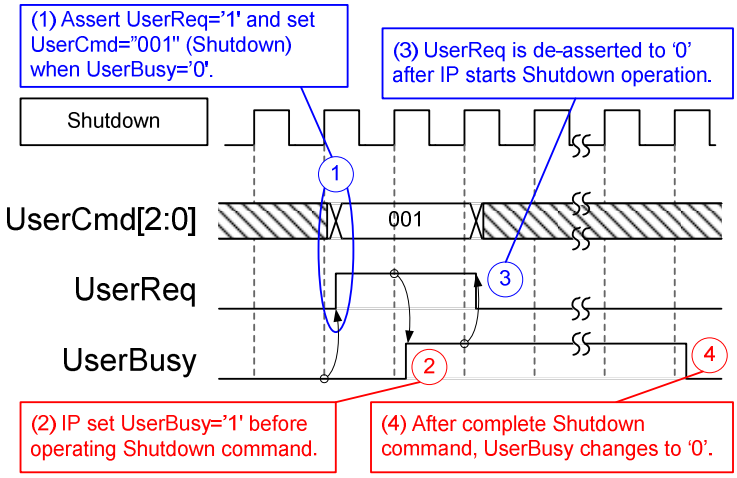

Figure 10: Example to send Shutdown command

SMART

To check SSD health, SMART command is applied to get 512-byte information through Custom command interface. To run SMART command, user must set UserCmd=”100”, set 16-Dword data to CtmSubmDW0-DW15, and assert UserReq to ‘1’ at the same clock cycle. UserAddr and UserLen input are not used in SMART command. After that, SSD returns 512-byte SMART data through CtmRam (Custom command RAM) interface. 128-bit SMART data is valid on CtmRamWrData bus at the same clock cycle as CtmRamWrEn=’1’. CtmRamAddr is used to be data index of 32 x 128-bit SMART data.

The value of CtmSubmDW0-DW15 for SMART command is fixed as follows.

CtmSubmDW0	= 0x0000_0002
CtmSubmDW1	= 0xFFFF_FFFF
CtmSubmDW2 – CtmSubmDW5	= 0x0000_0000
CtmSubmDW6	= 0x2000_0000
CtmSubmDW7 – CtmSubmDW9	= 0x0000_0000
CtmSubmDW10	= 0x007F_0002
CtmSubmDW11 – CtmSubmDW15	= 0x0000_0000

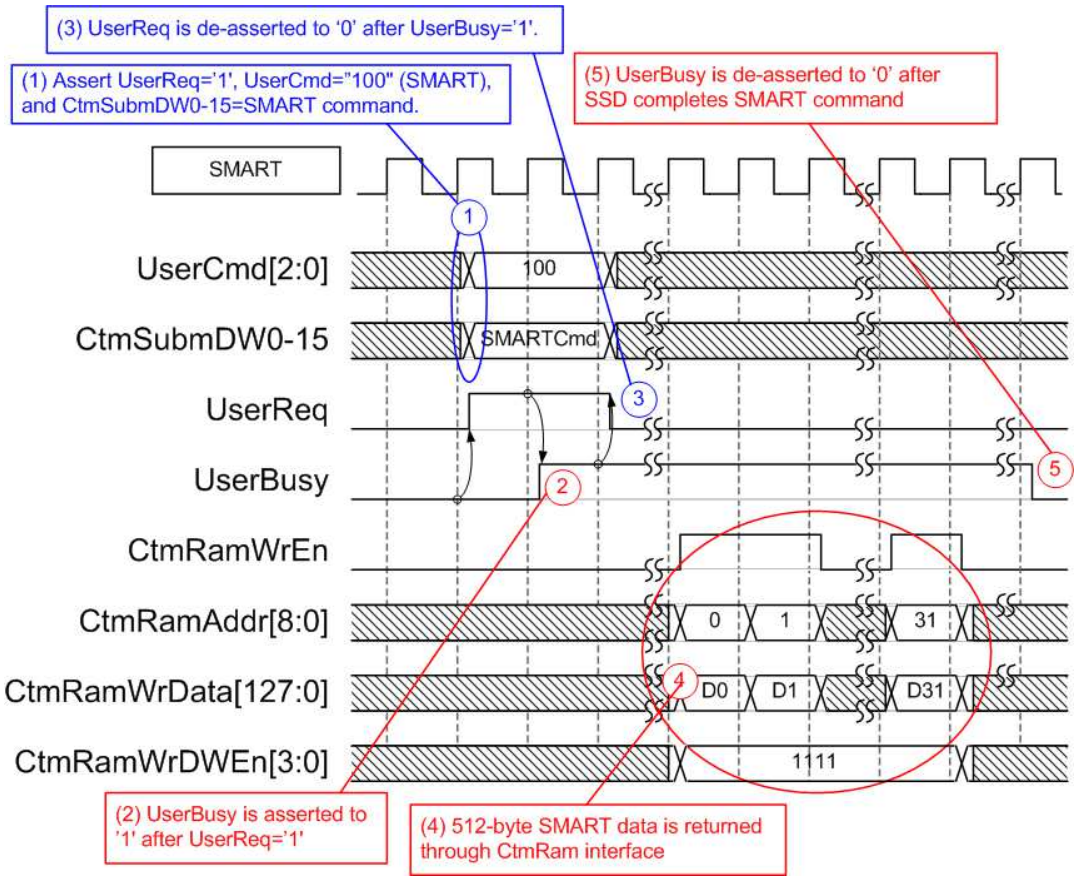

Figure 11: Example to send SMART command

As shown in Figure 11, SMART data is sent through CtmRam interface. CtmRamWrEn may not be asserted to ‘1’ continuously. CtmRamAddr is the index of CtmRamWrData. When CtmRamAddr=0 with asserting CtmRamWrEn to ’1’, byte0 – byte15 of SMART data is valid on CtmRamWrData signal. The last data (byte496 – byte511) of SMART data is valid on CtmRamWrData when CtmRamAddr=31 with asserting CtmRamWrEn to ’1’. Generally, CtmRamWrDWEn is equal to “1111” which means all 128-bit of CtmRamWrData are valid.

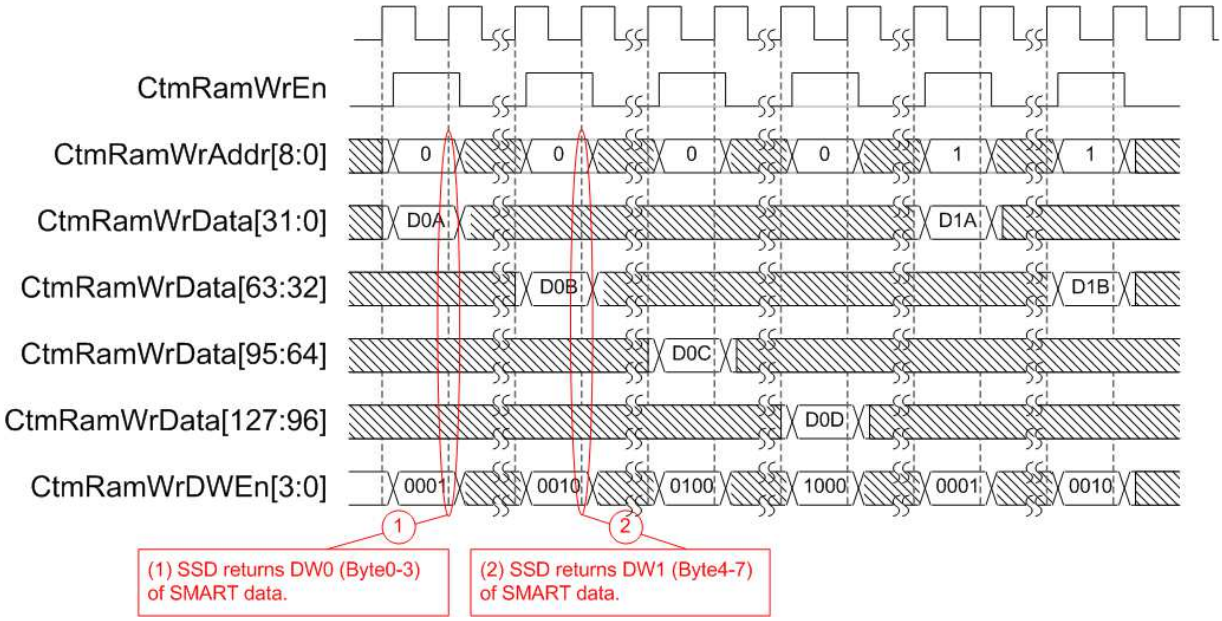

Figure 12 CtmRamWrDWEn timing diagram

Similar to Identify command, some SSDs may return SMART data in 32-bit data format instead of 128-bit data format. So, CtmRamWrDWEn must be used to check the valid Dword of CtmRamWrData signal. As shown in Figure 12, bit[0] of CtmRamWrDWEn is referred to bit[31:0] of CtmRamWrData, bit[1] is referred to bit[63:32] of CtmRamWrData, bit[2] is referred to bit[95:64] of CtmRamWrData, and bit[3] is referred to bit[127:96] of CtmRamWrData. Similar to CtmRamWrData, CtmRamWrDWEn is valid when CtmRamWrEn=’1’.

Flush

Generally, when user writes data to SSD, the data is stored to internal cache of SSD firstly. If SSD still not flush data from cache to memory and system is power down immediately, Write data in cache will be lost. To guarantee that data is not lost, there are two solutions.

First solution is sending Shutdown command before power down the system. This command is required before shutdown the system. Second solution is sending Flush command to SSD. Using this command, data in cache is flushed to the memory. Flush command could be used when system is Idle to back up data from cache to the memory. After using Flush command, SSD still be active status.

Similar to SMART command, Flush command must be sent through CtmSubmDW0-DW15 by using following value.

CtmSubmDW0	= 0x0000_0000
CtmSubmDW1	= 0x0000_0001
CtmSubmDW2 – CtmSubmSW15	= 0x0000_0000

UserAddr and UserLen inputs are not used in Flush command.

As shown in Figure 13, after sending Flush command, user logic waits until UserBusy changes from ‘1’ to ‘0’ to confirm that write data in cache has already flushed.

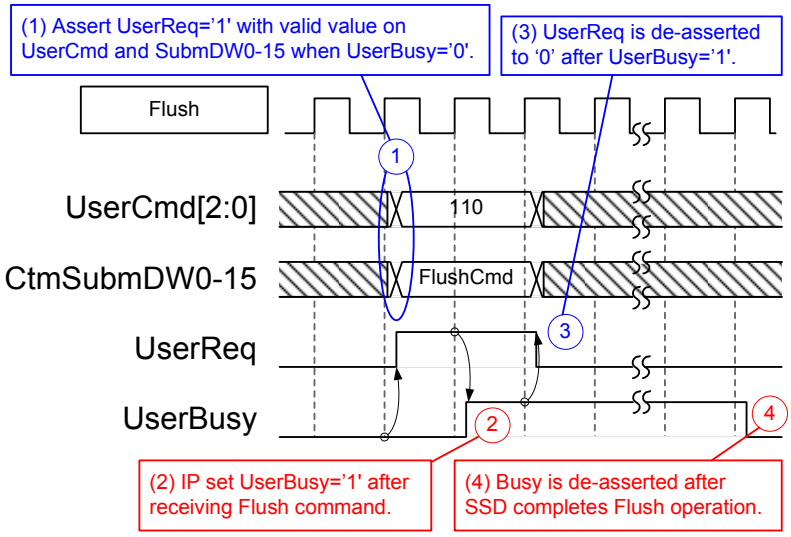

Figure 13: Example to send Flush command

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 bit of UserErrorType is set to '1', UserError will be asserted to '1'. UserError could be cleared by asserting RstB to be '0' only, as shown in Figure 14.

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.

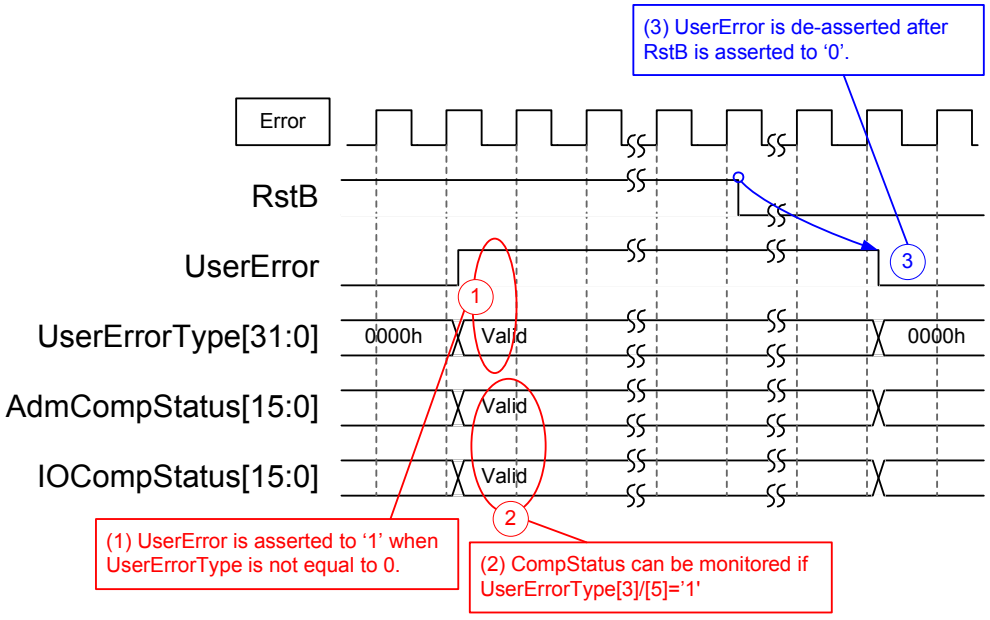

Figure 14: Error flag Timing diagram

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/Alaric 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
3.0	July-17-2018	Support Shutdown, SMART, and Flush command
3.1	Nov-23-2018	Support Memory Write Request which is not aligned to 128-bit unit