NVMe IP Core for PCIe switch

October 12, 2020 Product Specification Rev1.2

Design Gateway Co.,Ltd

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
Instantiation Templates	VHDL
Reference Designs & Application Notes	Vivado Project, See Reference Design Manual
Additional Items	Demo on KCU105, ZCU106, VCU118
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- NVMe host controller for access one NVMe SSD by direct connection or up to four NVMe SSDs by connection through one PCIe switch
- No need for CPU and external memory
- Include 256 Kbyte RAM to be data buffer
- Simple user interface by dgIF typeS to access one device at a time
- Support six commands, i.e. Identify, Shutdown, Write, Read, SMART, and Flush
- 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)
  - LBA unit: 512 bytes or 4096 bytes
- User clock frequency must be more than or equal to PCIe clock (250 MHz for PCIe Gen3)
- Operating with Integraged Block for PCI Express from Xilinx by using 4-lane PCIe Gen3 (128-bit bus interface)
- Available reference design: KCU105, ZCU106 and VCU118 with AB18-PCIeX16/AB16-PCIeXOVR adaptor board
- Customized service for following features
  - Additional NVMe commands
  - RAM size or RAM type (URAM) modification
  - Support more than four NVMe SSDs

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	363	5360	3492	1001	-	66	-	-	Vivado2017.4
Zynq-Ultrascale+	XCZU7EV-FFVC1156-2E	400	5360	3516	1029	-	66	-	-	Vivado2017.4
Virtex-Ultrascale+	XCVU9P-FLGA2104-2L	400	5360	3514	944	-	66	-	-	Vivado2017.4

Applications

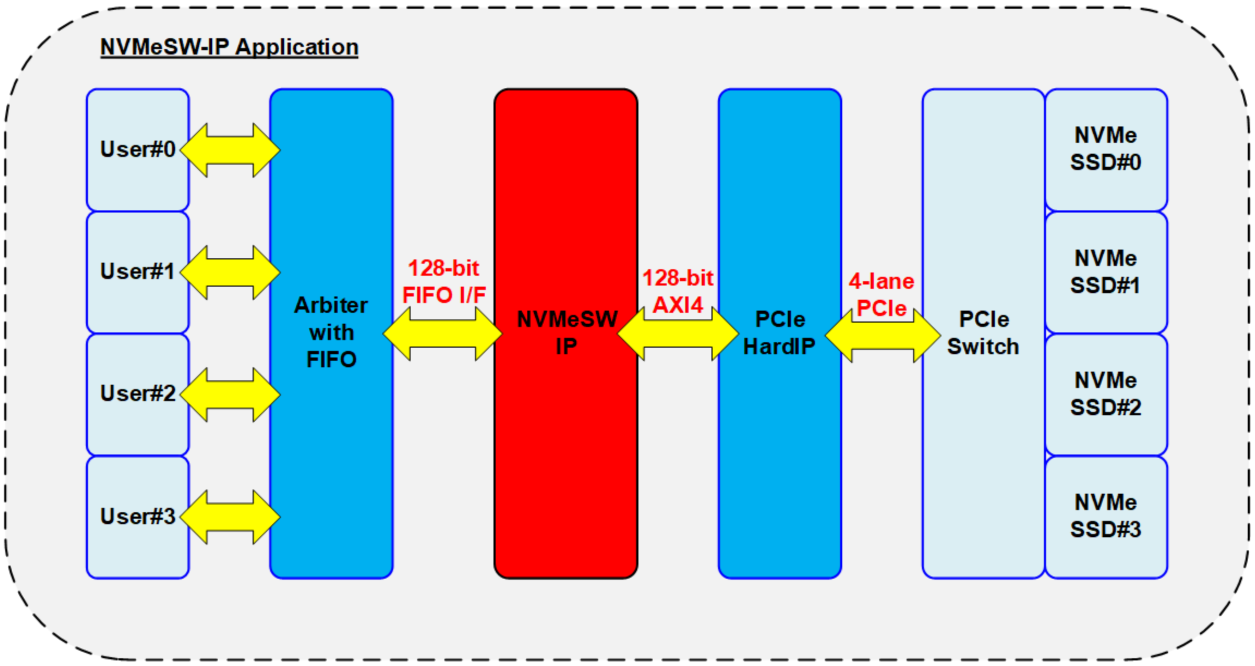

Figure 1: NVMeSW IP Application

NVMeSW IP Core integrated with Integrated Block for PCI Express (PCIe hard IP) from Xilinx is ideal to access NVMe SSD without CPU and external memory such as DDR. NVMeSW IP can access more than one SSD through PCIe switch, as shown in Figure 1. By designing Arbiter including the FIFO for bufferring the data, multiple user logics can be accessed the different NVMe SSDs through one set of NVMeSW IP, PCIe hard IP and PCIe switch. As a result, each SSD can store the different data stream from the different data source at very high performance. For more specific applications, we also provides alternative IP as follows.

Random Access NVMe IP Core– To access NVMe SSD with multiple commands, individual address for each. Recommended for the application which requires the access in non-contiguous area, stream data recording without specified length or application that one SSD can be accessed by multiple users.

https://dgway.com/raNVMe-IP_X_E.html

NVMe IP Core with PCIe Gen 3 Soft IP – When the selected FPGA does not have enough PCIe hard IP for the application.

NVMe IP Core with PCIe Gen 4 Soft IP – To access with PCIe Gen4 NVMe SSD.

https://dgway.com/NVMeG4-IP_X_E.html

General Description

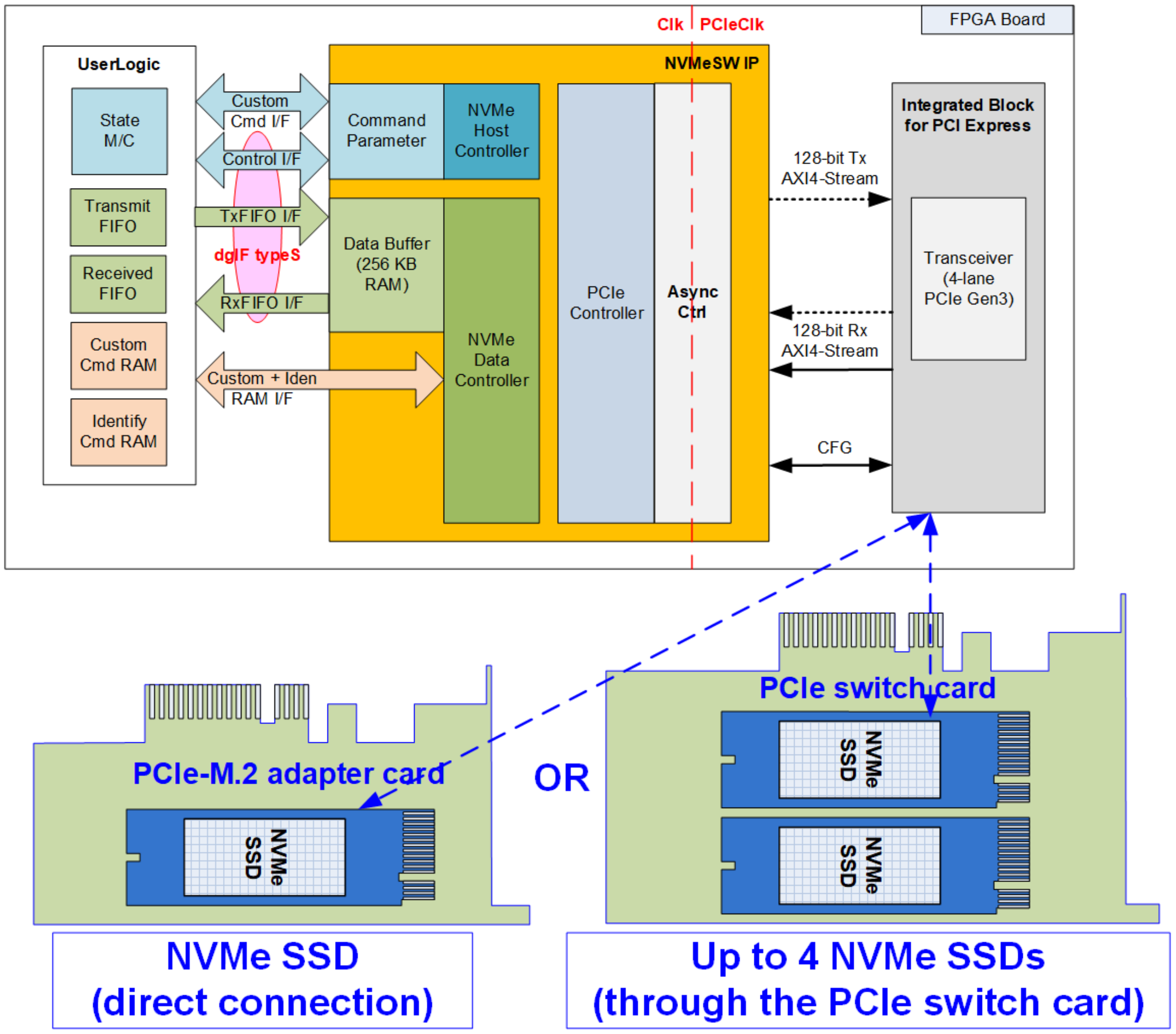

Figure 2: NVMeSW IP Block Diagram

NVMeSW IP implements as host controller to access NVMe SSD following NVM express standard. Physical interface of NVMe SSD is PCIe. The lower layer hardware is implemented by using Integrated Block for PCI Express from Xilinx.

NVMeSW IP supports six NVMe commands, i.e. Identify, Shutdown, Write, Read, SMART and Flush command by using two user interface groups. First is Control interface for transferring command and the parameters. Another is Data interface for transferring data when the command must have the data transferring. The Control interface and Data interface for Write/Read command use dgIF typeS format. Control interface of dgIF typeS has start address and transfer length with asserting the request signal while Data interface of dgIF typeS is the FIFO interface.

Although up to four NVMe SSDs could be connected to NVMeSW IP, only one SSD is operated at a time. NVMeSW IP has the signal for selecting active device, called UserDevSel. The device selection is the input parameter for all commands.

Note: When connecting through PCIe switch, the FPGA supports only one-level PCIe switch. More than one PCIe switch card connected as multiple-level are not supported. All SSDs are connected to the same PCIe switch.

SMART and Flush command require the specific interface, called Custom command interface, which consists of Ctm I/F for control path and Ctm RAM I/F for data path. Furthermore, Identify command has its own data interface, named Iden RAM I/F, as shown in Figure 2.

During initialization or running some commands, error signal may be asserted by NVMeSW IP if some abnormal conditions are found. The IP includes the error status to check the more details of error condition. To recover error status, NVMeSW IP and all SSDs must be reset.

There is one limitation about clock frequency of user logic. Transmit packet to PCIe hard IP must be sent continuously until end of packet. Therefore, data must be valid every clock between start of packet and end of packet. To support this feature, user logic clock frequency must be more than or equal to PCIe clock frequency (250 MHz for PCIe Gen3) to have the bandwidth of transmit logic higher than or equal to PCIe hard IP bandwidth.

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

Functional Description

Figure 3 shows the operation flow of NVMeSW IP after IP reset is de-asserted.

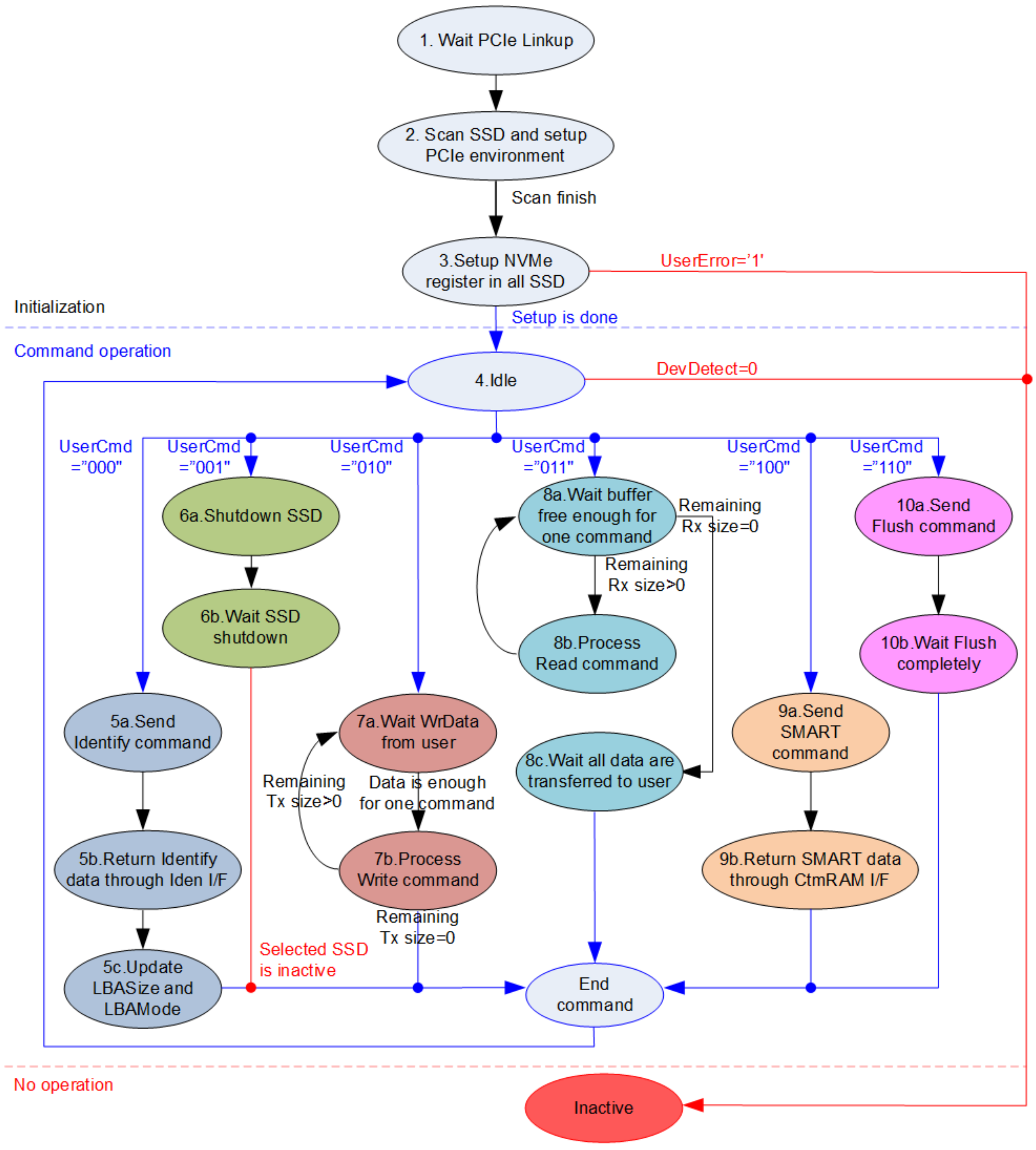

Figure 3: NVMeSW IP Operation Flow

The operation of NVMeSW IP is described as follows.

1) IP waits until PCIe is ready by monitoring Linkup status from PCIe IP core.
2) IP reads PCIe configuration to scan device connection type (direct connection or connection through PCIe switch) and scan the number of SSDs in the connection. After that, the IP sets up PCIe environment of all SSDs (up to 4 SSDs).
3) IP sets up NVMe parameters to one SSD at a time. When multiple SSDs are connected, the setup process is run many times to setup all SSDs. After finishing the setup process, IP enters to the Idle state to wait for a new command request from the user. If some SSDs are detected during setup process, the IP changes to the Inactive state.
4) In the Idle state, IP is ready to receive command from user. To start the new command operation, user sets UserCmd with the input parameters and sets UserDevSel to select the active SSD.
5) The 1st command from user must be Identify command (UserCmd=”000”) to update LBASize (disk capacity) and LBAMode (LBA unit=512 bytes or 4 Kbytes). When many SSDs are connected through PCIe switch, Identify command must be sent many times to get the information of all SSDs.
6) The last command before power down the system must be Shutdown command (UserCmd=”001”). This command is recommended to guarantee SSD powered down in a good sequence. Without Shutdown command, Write data in SSD cannot be guaranteed. After finishing Shutdown command, the selected SSD changes to inactive state. Next, NVMeSW IP changes to the Idle state to receive new command the ramaining active SSDs. The inactive SSD could not be accessed until the system is reset.

   <u>Note</u>: After all SSDs are inactive by Shutdown command, the state changes from the Idle state to the Inactive state. NVMeSW IP does not receive any command in Inactive state until resetting the IP.
7) For Write command (UserCmd=”010”), the maximum data size of one command is 128 Kbytes. If total length from user is more than 128 Kbytes, repeat step 7a) – 7b) for many times.
   a) The IP waits until Write data, sent by user, is enough for one command (transfer size of one command in NVMeSW IP is 128 Kbytes, except the last loop which could be less than 128 Kbytes).
   b) The IP sends Write command to SSD and then waits until the status is returned from SSD. The IP changes to the Idle state when total data is completely transferred. Otherwise, the IP goes back to step 7a) to send the next Write command.
8) Similar to Write command, Read command (UserCmd=”011”) must be sent to the SSD many times when total length from user is more than 128 Kbytes. As a result, step 8a) – 8b) are repeated many times.
   a) The IP waits until the free space of data buffer in NVMeSW IP is enough for one command. If remaining transfer size is equal to zero, the IP skips to step c).
   b) The IP sends Read command to SSD. After that, goes back to step a) to check the remaining transfer size and the free space of data buffer.
   c) IP waits until all data are completely transferred from data buffer to user logic and then changes to the Idle state. Therefore, data buffer is empty after finishing Read command.
9) For SMART command (UserCmd=”100”), 512-byte data is returned after finishing the operation.
   a) IP sends Get Log Page command to read SMART/Health information from SSD.
   b) 512-byte data is returned from the SSD. The IP forwards the data through Custom command RAM interface (CtmRamAddr=0x000 – 0x01F).
10) For Flush command (UserCmd=”110”), there is no data transferring during the operation.
    a) IP sends Flush command to the SSD.
    b) IP waits until SSD returns status to complete the operation.

To design NVMe host controller, NVMeSW IP implements two protocols, i.e. NVMe protocol for interfacing with user and PCIe protocol for interfacing with PCIe hard IP. The details of the hardware inside NVMeSW IP are described as follows.

NVMe

NVMeSW IP supports six commands, i.e. Identify, Write, Read, SMART, Flush and Shutdown command which can be divided to two command types, i.e. Admin command and NVM command. NVMeSW IP supports three Admin commands, i.e. Identify, Shutdown and SMART command and three NVM commands, i.e. Write, Read and Flush command. After finishing the command, the status returned from the SSD is latched to AdmCompStatus signal when running Admin command or IOCompStatus signal when running NVM command.

The parameters of Write or Read command are set by Control interface of dgIF typeS while the parameters of SMART or Flush command are set by CtmSubm of Ctm interface. Data interface for Write or Read command is transferred by FIFO interface, a part of dgIF typeS, which is finally connected with 256-Kbyte buffer inside the IP. The data interface of other commands has its own interface, i.e. Identify RAM for Identify command and Custom RAM for SMART command.

The details of each submodule are described as follows.

- **NVMe Host Controller**

  NVMe host controller is the main controller in NVMeSW IP. The operation is split into two phases. First is the initialization phase which is once run after the system is boot up for setting NVMe register inside the SSD. After finishing the initialization phase, the next phase is the command operation phase. The order of transmitted and received packet is controlled by NVMe host controller.

  To operate the command, the parameters of each command are prepared in Command Parameter and then forwarded to AsyncCtrl. After finishing the command, the status packet returned from SSD is monitored by NVMe host controller to check if no error is found. When the command needs to transfer data such as Write, Read, SMART and Identify command, NVMe host controller must also handle with NVMe data controller.

- **Command Parameter**

  This module is designed to prepare command packet for SSD and decode status packet returned from SSD.

  Typically, the command consists of 16 Dwords (1 Dword = 32-bit). When running Identify, Shutdown, Write or Read command, all 16 Dwords are created by Command parameter, following the user inputs on dgIF typeS. When running SMART and Flush command, all 16 Dwords are directly loaded via CtmSubmDW0-CtmSubmDW15 of Ctm interface.

- **Data Buffer**

  256-Kbyte simple dual port RAM is implemented by BlockRAM to be data buffer. The buffer stores data transferred between UserLogic and SSD during Write and Read command operation.

- **NVMe Data Controller**

  This module is operated when the command must transfer the data, i.e. Identify, SMART, Write and Read command. There are three data interfaces for transferring with the SSD, i.e. FIFO interface with 256-Kbyte buffer when running Write or Read command, Custom command RAM interface when running SMART command or Identify RAM interface when running Identify command.

  When running Write or Read command, the address of the data buffer is controlled by NVMe data controller.

PCIe

The PCIe standard is the outstanding lower layer protocol for very high speed application. The NVMe standard is the protocol which is run over PCIe protocol. In the initialization process, NVMe layer is setup after finishing PCIe layer setup. Two modules are designed to support PCIe protocol, i.e. PCIe controller and AsyncCtrl. More details of each module are described as follows.

- **PCIe Controller**

  During initialization process, PCIe controller sets up PCIe environment of all SSDs via CFG interface. After that, PCIe packet is created or decoded via 128-bit Tx/Rx AXI4-Stream. NVMeSW IP must scan the connected device and detect the device type. If no NVMe SSD is detected in the chain, the error flag is asserted.

  The command and data packet from NVMe module are converted to be PCIe packet by PCIe controller. On the other hand, the received PCIe packet is decoded and converted to be NVMe packet for NVMe module by this module.

- **AsyncCtrl**

  AsyncCtrl includes asynchronous registers and buffers to support signal and data clock-crossing. Most logics in NVMeSW IP run on user clock domain while PCIe hard IP runs on PCIe clock domain. AXI4-stream interface of PCIe hard IP must transfer data of each packet continuously, so the user bandwidth must be greater than or equal to PCIe bandwidth by running at higher or the same clock frequency of PCIe clock.

User Logic

This module could be designed by using small state machine to send the commands with assigning the parameters for each command. For Write/Read command, command parameters are address and transfer size. Also, Device selector is the parameter for every command to select the active SSD. Data interface for Write/Read command can directly connect to FIFO while data output from SMART and Identify command can directly connect to simple dual port RAM. RAM size depends on the data size transferring in each command, but data width of all commands is 128-bit. Data size of Identify command is 8-Kbyte while data size of SMART command is 512-byte.

Integrated Block for PCI Express

The integrated block for PCIe is the hard IP integrated in some Xilinx FPGAs. NVMeSW IP uses one PCIe hard IP which can connect to one PCIe switch with up to four NVMe SSDs. More connections can be customized. By using multiple IPs, more SSDs can be connected to the system. More details of PCIe hard IP are described in following document.

PG156: UltraScale Devices Gen3 Integrated Block for PCI Express

PG213: UltraScale+ Devices Integrated Block for PCI Express

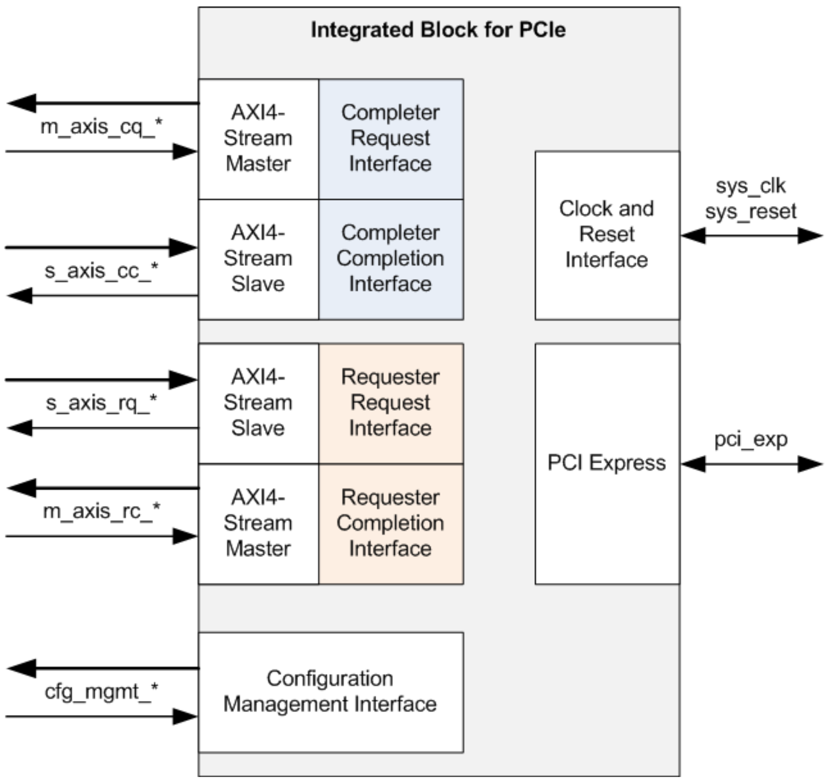
Integrated Block for PCIe
m_axis_cq_*
AXI4-Stream Master
Completer Request Interface
s_axis_cc_*
AXI4-Stream Slave
Completer Completion Interface
s_axis_rq_*
AXI4-Stream Slave
Requester Request Interface
m_axis_rc_*
AXI4-Stream Master
Requester Completion Interface
cfg_mgmt_*
Configuration Management Interface
Clock and Reset Interface
sys_clk
sys_reset
PCI Express
pci_exp

Figure 4: Integrated Block for PCI Express (UltraScale+)

Core I/O Signals

Descriptions of all signal I/Os are provided in Table 2 and Table 3.

Table 2: User logic I/O Signals (Synchronous to Clk signal)

Signal	Dir	Description
Control I/F of dgIF typeS		
RstB	In	Synchronous reset signal. Active low. De-assert to '1' when Clk signal is stable.
Clk	In	System clock for running NVMeSW IP. The frequency must be more than or equal to PCIeClk which is output from PCIe hard IP (250 MHz for PCIe Gen3).
DevDetect[3:0]	Out	Detected SSD status. Each bit shows one SSD status. Bit[0], [1], [2] and [3] are SSD#0, #1, #2 and #3 respectively. '0': Not found, '1': NVMe SSD detected. DevDetect signal is valid after the IP completes initialization process (UserBusy='0'). When one SSD is connected to the FPGA directly, DevDetect is equal to "0001". Four bits are all asserted to '1' when four SSDs are connected. After user sends Shutdown command, DevDetect of the selected SSD is de-asserted to '0'.
UserDevSel[1:0]	In	Select the active SSD for running the command. Set "00", "01", "10" or "11" for selecting SSD#0, #1, #2 or #3 respectively. Before selecting the device, please check DevDetect signal that the selected SSD is still active.
UserCmd[2:0]	In	User Command. Valid when UserReq='1'. ("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. Valid when UserReq='1'. 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 to write/read SSD 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. Valid when UserReq='1'.
UserReq	In	Asserts to '1' to send the new command request and de-asserts to '0' after IP starts the operation by asserting UserBusy to '1'. This signal can be asserted when the IP is Idle (UserBusy='0'). Command parameters (UserDevSel, UserCmd, UserAddr, UserLen and CtmSubmDW0-DW15) must be valid and stable during UserReq='1'. UserDevSel, UserAddr and UserLen are inputs for Write/Read command while UserDevSel and CtmSubmDW0-DW15 are inputs for SMART/Flush command.
UserBusy	Out	Asserted to '1' when IP is busy. New request must not be sent (UserReq to '1') when IP is busy.
LBASize0[47:0] - LBASize3[47:0]	Out	Total capacity of SSD#0-#3 in 512-byte unit. Default value is 0. LBASize0, 1, 2 and 3 are the size of SSD#0, #1, #2 and #3 respectively. LBASize of each SSD is valid after finishing Identify command.
LBAMode[3:0]	Out	LBA unit size of SSD ('0': 512 bytes, '1': 4 Kbytes). Default value is 0. Bit[0], [1], [2] and [3] are referred to SSD#0, #1, #2 and #3 respectively.- This value is valid after finishing Identify command.
UserError	Out	Error flag. Assert when UserErrorType is not equal to 0. The flag can be cleared by asserting RstB to '0'.
UserErrorDev[1:0]	Out	Show the active device which is in operating. So, UserErrorDev can be applied to check the error device number.

Signal	Dir	Description
Control I/F of dgIF typeS		
UserErrorType[31:0]	Out	Error status. [0] – Error when no SSD is detected during initialization process. *Note: This flag is not asserted after all SSDs finish Shutdown command.* [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 PCIe hard IP detects Error correction code (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 bytes or 4 Kbytes) [31:17] – Reserved *Note: Timeout period of bit[2]/[4] is set from TimeOutSet input.*
Data I/F of dgIF typeS		
UserFifoWrCnt[15:0]	In	Write data counter of Receive FIFO. Used to check full status. If total FIFO size is less than 16 bits, please fill '1' to upper bit.
UserFifoWrEn	Out	Asserted to '1' to write data to Receive FIFO during running Read command.
UserFifoWrData[127:0]	Out	Write data bus of Receive FIFO. Valid when UserFifoWrEn='1'.
UserFifoRdCnt[15:0]	In	Read data counter of Transmit FIFO. Used to check data size stored in FIFO. If FIFO size signal is less than 16 bits, please fill '0' to upper bit.
UserFifoEmpty	In	The signal is unused for this IP.
UserFifoRdEn	Out	Asserted to '1' to read data from Transmit FIFO during running Write command.
UserFifoRdData[127:0]	In	Read data returned from Transmit FIFO. Valid in the next clock after UserFifoRdEn is asserted to '1'.

Signal	Dir	Description
NVMeSW IP Interface		
IPVesion[31:0]	Out	IP version number
TestPin[63:0]	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 set to 0, Timeout function is disabled.
PCIeLinkup	In	Asserts to '1' when LTSSM state of PCIe hard IP is in L0 State.
AdmCompStatus[15:0]	Out	Status output from Admin Completion Entry [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	Status output from IO Completion Entry [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	The parameter value of the NVMe capability register when UserErrorType[1] is asserted to '1'. [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	Asserted to '1' for sending data output from Identify command.
IdenWrDWEn[3:0]	Out	Dword (32 bit) enable of IdenWrData. Valid when IdenWrEn='1'. '1': this dword data is valid, '0': this dword data is not available. Bit[0], [1], [2] and [3] corresponds to IdenWrData[31:0], [63:32], [95:64] and [127:96] respectively.
IdenWrAddr[8:0]	Out	Index of IdenWrData in 128-bit unit. Valid when IdenWrEn='1'. 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. Valid when IdenWrEn='1'.
Custom interface		
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	Asserted to '1' for sending data output from custom command such as SMART command.
CtmRamWrDWEn[3:0]	Out	Dword (32-bit) enable of CtmRamWrData. Valid when CtmRamWrEn='1'. '1': this dword data is valid, '0': this dword data is not available. Bit[0], [1], [2] and [3] corresponds to CtmRamWrData[31:0], [63:32], [95:64] and [127:96] respectively.
CtmRamAddr[8:0]	Out	Index of CtmRamWrData when SMART data is received. Valid when CtmRamWrEn='1'. (Optional) Index to request data input through CtmRamRdData for customized custom commands.
CtmRamWrData[127:0]	Out	512-byte data output from SMART command. Valid when CtmRamWrEn='1'.
CtmRamRdData[127:0]	In	(Optional) Data input for customized custom commands.

Table 3: Physical I/O Signals for PCIe Gen3 Hard IP (Synchronous to PCIeClk)

Signal	Dir	Description
PCIe Gen3 hard IP		
PCIeRstB	In	Synchronous reset signal. Active low. De-asserts to ‘1’ when PCIe hard IP is not in reset state.
PCIeClk	In	Clock output from PCIe hard IP (250 MHz for PCIe Gen3).
Configuration Management Interface		
PCIeCfgDone	In	Read/Write operation complete. Asserted for 1 cycle when the operation completes.
PCIeCfgRdEn	Out	Read enable. Asserted to ‘1’ for a read operation.
PCIeCfgWrEn	Out	Write enable. Asserted to ‘1’ for a write operation.
PCIeCfgWrData[31:0]	Out	Write data which is used to configure the Configuration and Management registers.
PCIeCfgByteEn[3:0]	Out	Byte enable for write data, where bit[0] corresponds to PCIeCfgWrData[7:0], and so on.
PCIeCfgAddr[18:0]	Out	Read/Write Address.
Requester Request Interface		
PCIeMtTxData[127:0]	Out	Requester request data bus.
PCIeMtTxKeep[3:0]	Out	Bit[i] indicates that Dword[i] of PCIeMtTxData contains valid data.
PCIeMtTxLast	Out	Asserted this signal in the last cycle of a TLP to indicate the end of the packet.
PCIeMtTxReady[3:0]	In	Asserts to accept data. Data is transferred when both PCIeMtTxValid and PCIeMtTxReady are asserted in the same cycle.
PCIeMtTxUser[59:0]	Out	Requester request user data. Valid when PCIeMtTxValid is high.
PCIeMtTxValid	Out	Asserted to drive valid data on PCIeMtTxData bus. NVMeSW IP keeps the valid signal asserted during the transfer of a packet.
Completer Request Interface		
PCIeMtRxData[127:0]	In	Receive data from PCIe hard IP.
PCIeMtRxKeep[3:0]	In	Bit[i] indicates that Dword[i] of PCIeMtRxData contains valid data.
PCIeMtRxLast	In	Asserts this signal in the last beat of a packet to indicate the end of the packet.
PCIeMtRxReady	Out	Indicates that NVMeSW IP is ready to accept data.
PCIeMtRxUser[74:0]	In	Sideband information for the TLP being transferred. Valid when PCIeMtRxValid is high.
PCIeMtRxValid	In	Asserts when PCIe hard IP drives valid data on PCIeMtRxData bus. PCIe hard IP keeps the valid signal asserted during the transfer of packet.
Completer Completion Interface		
PCIeSlTxData[127:0]	Out	Completion data from NVMeSW IP.
PCIeSlTxKeep[3:0]	Out	Bit[i] indicates that Dword[i] of PCIeSlTxData contains valid data.
PCIeSlTxLast	Out	Asserted this signal in the last cycle of a packet to indicate the end of the packet.
PCIeSlTxReady[3:0]	In	Indicates that PCIe hard IP is ready to accept data.
PCIeSlTxUser[32:0]	Out	Sideband information for the TLP being transferred. Valid when PCIeSlTxValid is high.
PCIeSlTxValid	Out	Asserted to drive valid data on PCIeSlTxData bus. NVMeSW IP keeps the valid signal asserted during the transfer of a packet.
Requester Completion Interface		
PCIeSlRxData[127:0]	In	Receive data from PCIe hard IP.
PCIeSlRxKeep[3:0]	In	Bit[i] indicates that Dword[i] of PCIeSlRxData contains valid data.
PCIeSlRxLast	In	Asserts this signal in the last beat of a packet to indicate the end of the packet.
PCIeSlRxReady	Out	Indicates that NVMeSW IP is ready to accept data.
PCIeSlRxUser[84:0]	In	Sideband information for the TLP being transferred. Valid when PCIeSlRxValid is high.
PCIeSlRxValid	In	Asserts when PCIe hard IP drives valid data on PCIeSlRxData bus. PCIe hard IP keeps the valid signal asserted during the transfer of packet.

Timing Diagram

Initialization

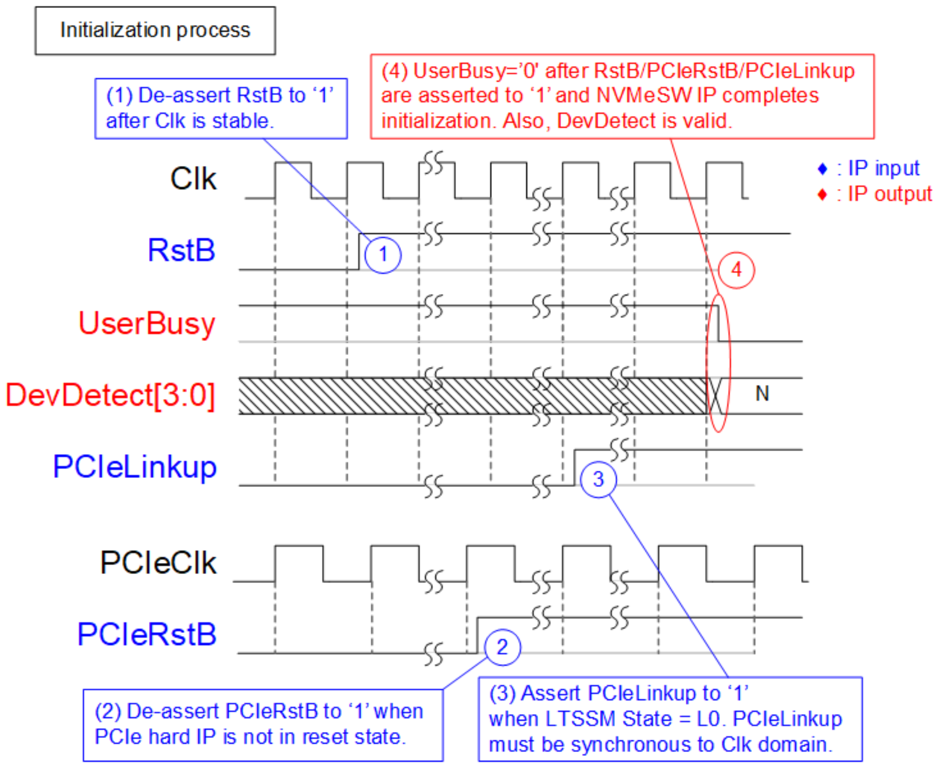
Initialization process
(1) De-assert RstB to '1' after Clk is stable.
(4) UserBusy='0' after RstB/PCIeRstB/PCIeLinkup are asserted to '1' and NVMeSW IP completes initialization. Also, DevDetect is valid.
♦ : IP input
♦ : IP output
Clk
RstB
UserBusy
DevDetect[3:0]
N
PCIeLinkup
PCIeClk
PCIeRstB
1
2
3
4
(2) De-assert PCIeRstB to '1' when PCIe hard IP is not in reset state.
(3) Assert PCIeLinkup to '1' when LTSSM State = L0. PCIeLinkup must be synchronous to Clk domain.

Figure 5: Timing diagram during initialization process

The sequence of the initialization process is as follows.

1) Wait until Clk is stable and then de-asserts RstB to '1' to start IP operation.
2) PCIe hard IP de-asserts PCIeRstB to '1' after finishing PCIe reset sequence. PCIe hard IP is ready to interface with the application layer.
3) Assert PCIeLinkup to '1' after LTSSM state of PCIe hard IP is L0 state. Though LTSSM state is run on PCIeClk, PCIeLinkup must be generated on Clk domain. Asynchronous register must be applied to generate PCIeLinkup. After that, NVMeSW IP starts initialization process.
4) When finishing initialization, NVMeSW IP de-asserts UserBusy to '0'. DevDetect[3:0] signal is valid to show the on-line status of four SSDs. If no device is found, DevDetect="0000" and error flag is asserted to '1'. UserBusy is not de-asserted to '0' when no device is detected.

After finishing above sequence, NVMeSW IP is ready to receive the command from user.

Control interface of dgIF typeS

dgIF typeS signals are split into two groups. First group is control interface for sending command with the parameters and monitoring the status. Second group is data interface for transferring data stream in both directions.

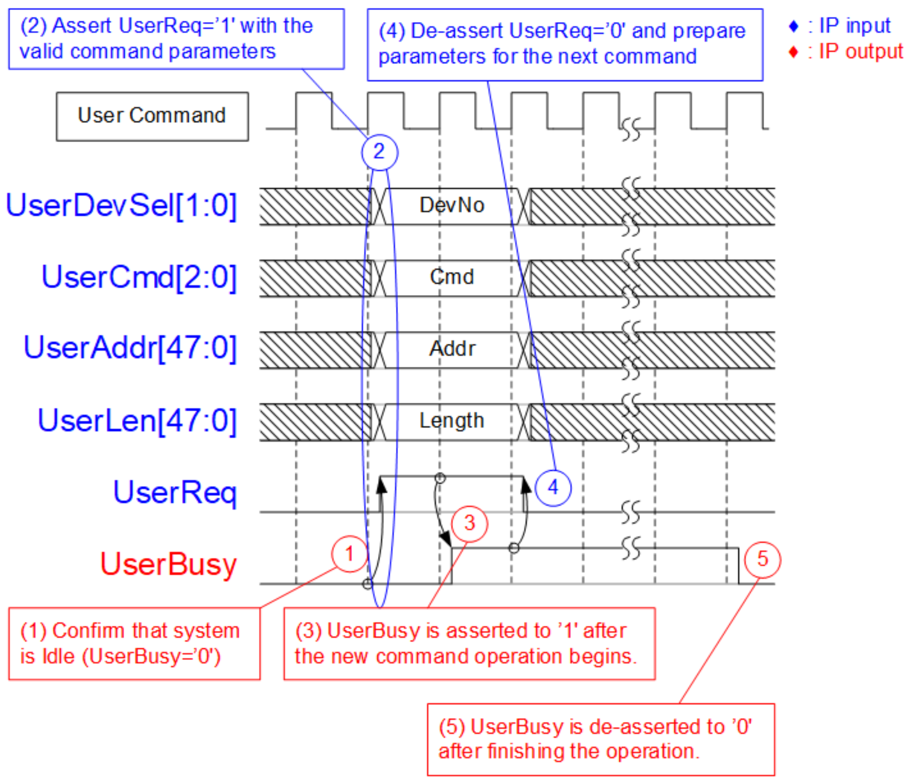
(2) Assert UserReq='1' with the valid command parameters
(4) De-assert UserReq='0' and prepare parameters for the next command
♦ : IP input
♦ : IP output
User Command
UserDevSel[1:0]
UserCmd[2:0]
UserAddr[47:0]
UserLen[47:0]
UserReq
UserBusy
DevNo
Cmd
Addr
Length
1
2
3
4
5
(1) Confirm that system is Idle (UserBusy='0')
(3) UserBusy is asserted to '1' after the new command operation begins.
(5) UserBusy is de-asserted to '0' after finishing the operation.

Figure 6: Contol interface of dgIF typeS timing diagram

1) Before sending new command to the IP, UserBusy must be equal to '0' to confirm that IP is the Idle state.
2) Command and the parameters such as UserDevSel, UserCmd, UserAddr and UserLen must be valid when asserting UserReq to '1' for sending the new command request.
3) IP asserts UserBusy to '1' after starting the new command operation.
4) After UserBusy is asserted to '1', UserReq is de-asserted to '0' to finish the current request. New parameters for the next command could be prepared on the bus. UserReq for the new command must not be asserted to '1' until the current command operation is finished.
5) UserBusy is de-asserted to '0' after the command operation is completed. New command request could be sent by asserting UserReq to '1'.

<u>Note</u>: The number of parameters using in each command is different.

- *Write and Read command: Use UserDevSel, UserCmd, UserAddr and UserLen.*
- *SMART and Flush command: Use UserDevSel, UserCmd and CtmSubmDW0-DW15.*
- *Identify and Shudown command: Use UserDevSel and UserCmd.*

Data interface of dgIF typeS

Data interface of dgIF typeS is applied for transferring data stream when operating Write command or Read command. The interface is compatible to general FIFO interface. 16-bit FIFO read data counter (UserFifoRdCnt) shows total data stored in the FIFO before transferring as a burst. The burst size is 512 bytes or 32 cycles of 128-bit data.

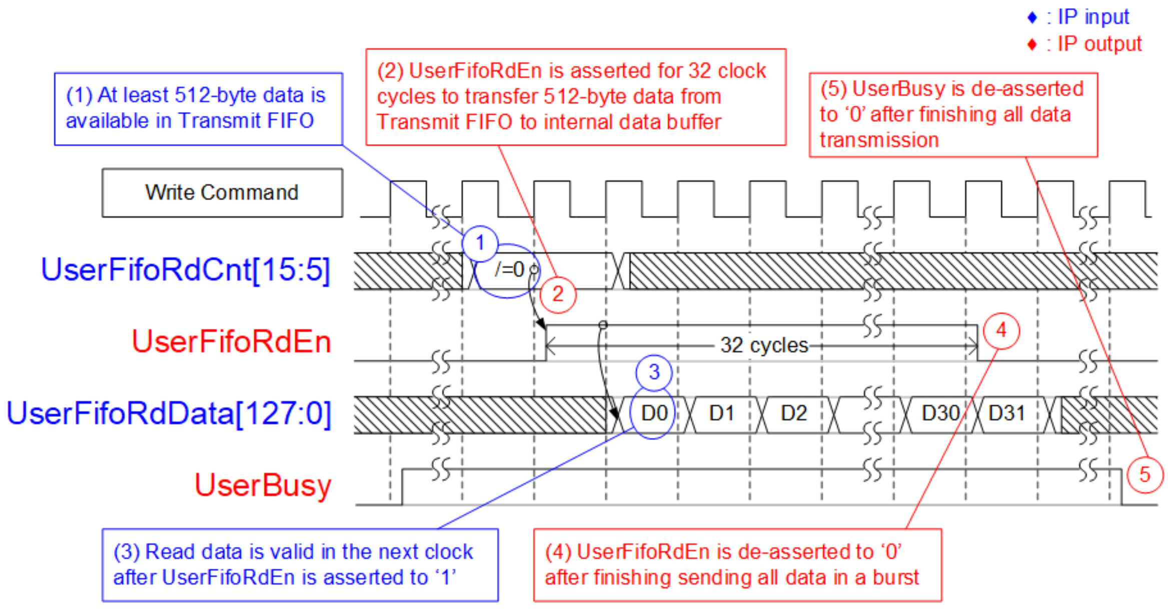
♦ : IP input
♦ : IP output
(1) At least 512-byte data is available in Transmit FIFO
(2) UserFifoRdEn is asserted for 32 clock cycles to transfer 512-byte data from Transmit FIFO to internal data buffer
(5) UserBusy is de-asserted to '0' after finishing all data transmission
Write Command
UserFifoRdCnt[15:5]
/=0
UserFifoRdEn
32 cycles
UserFifoRdData[127:0]
D0
D1
D2
D30
D31
UserBusy
1
2
3
4
5
(3) Read data is valid in the next clock after UserFifoRdEn is asserted to '1'
(4) UserFifoRdEn is de-asserted to '0' after finishing sending all data in a burst

Figure 7: Transmit FIFO Interface for Write command

In Write command, data is read from Transmit FIFO until total data are transferred completely. The details to transfer data are described as follows.

1) Before starting a new burst transfer, UserFifoRdCnt[15:5] is monitored. The IP waits until at least 512-byte data is available in Transmit FIFO (UserFifoRdCnt[15:5] is not equal to 0).
2) The IP asserts UserFifoRdEn to '1' for 32 clock cycles to read 512-byte data from Transmit FIFO.
3) UserFifoRdData is valid in the next clock cycle after asserting UserFifoRdEn to '1'. 32 data are continuously transferred.
4) UserFifoRdEn is de-asserted to '0' after reading the 32th data. Repeat step 1) – 4) to transfer the next 512-byte until total data size is equal to the transfer size in the command.
5) After total data is completely transferred, UserBusy is de-asserted to '0'.

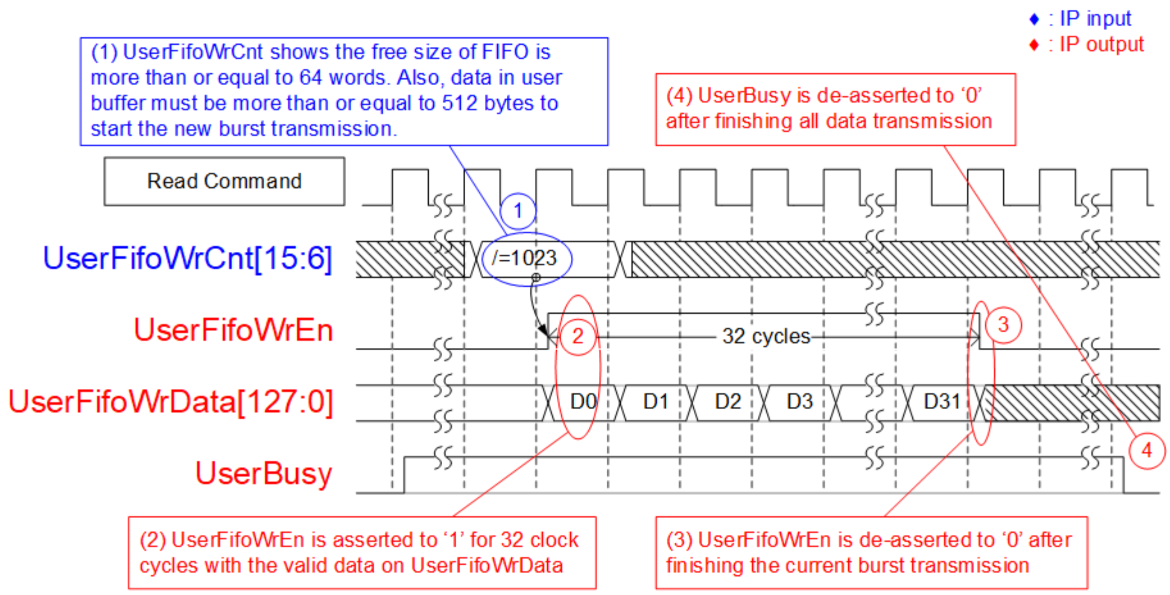
♦ : IP input
♦ : IP output
(1) UserFifoWrCnt shows the free size of FIFO is more than or equal to 64 words. Also, data in user buffer must be more than or equal to 512 bytes to start the new burst transmission.
(4) UserBusy is de-asserted to '0' after finishing all data transmission
Read Command
UserFifoWrCnt[15:6]
/=1023
UserFifoWrEn
32 cycles
UserFifoWrData[127:0]
D0
D1
D2
D3
D31
UserBusy
1
2
3
4
(2) UserFifoWrEn is asserted to '1' for 32 clock cycles with the valid data on UserFifoWrData
(3) UserFifoWrEn is de-asserted to '0' after finishing the current burst transmission

Figure 8: Receive FIFO Interface for Read command

In Read command, data is transferred from SSD to Receive FIFO until total data are completely transferred. The details to transfer data are as follows.

1) Before starting the new burst transmission, UserFifoWrCnt[15:6] is monitored. The IP waits until the free space of Receive FIFO is enough (UserFifoWrCnt[15:6] is not equal to all 1 or 1023). After received data from the SSD is more than or equal to 512 bytes, the new burst transmission begins.
2) The IP asserts UserFifoWrEn to '1' for 32 clock cycles to transfer 512-byte data from the data buffer to user logic.
3) After finishing transferring 512-byte data, UserFifoWrEn is de-asserted to '0'. Repeat step 1) – 3) to transfer the next 512-byte data until total data size is equal to the transfer size in the command.
4) After total data is completely transferred, UserBusy is de-asserted to '0'.

IdenCtrl/IdenName

It is recommened to send Identify command to the IP as the first command after system boots up. This command updates the necessary information of all SSDs, i.e. total capacity (LBASize0-LBASize3) and LBA unit size (LBAMode[3:0]). The SSD information is applied to be the limitation of the input parameters when operating Write or Read command, described as follows.

1) The sum of the address (UserAddr) and transfer length (UserLen) of Write and Read command must not be more than total capacity (LBASize) of the active SSD.
2) If LBAMode of the active SSD is equal to '1' (LBA unit size is 4 Kbytes), the three lower bit (bit[2:0]) of UserAddr and UserLen must be always equal to '0' to align 4-Kbyte unit.

When multiple SSDs are detected in the system, user logic must send Identify command many times with switching UserDevSel value for operating with all SSDs. After finishing the command, LBASize and LBAMode of all SSDs are updated (not equal to 0). The example to send Identify command to SSD#0 is shown in Figure 9.

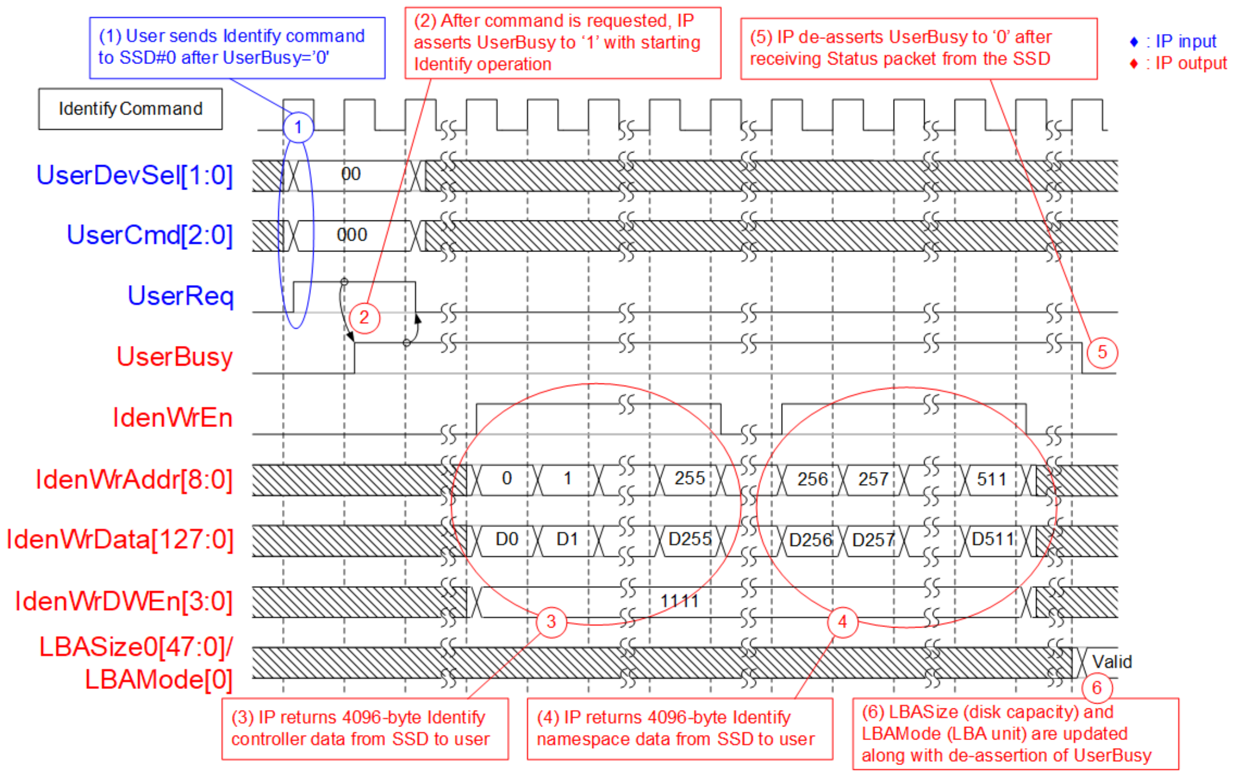
(1) User sends Identify command to SSD#0 after UserBusy='0'
(2) After command is requested, IP asserts UserBusy to '1' with starting Identify operation
(5) IP de-asserts UserBusy to '0' after receiving Status packet from the SSD
♦ : IP input
♦ : IP output
Identify Command
UserDevSel[1:0]
00
UserCmd[2:0]
000
UserReq
UserBusy
IdenWrEn
IdenWrAddr[8:0]
0
1
255
256
257
511
IdenWrData[127:0]
D0
D1
D255
D256
D257
D511
IdenWrDWEn[3:0]
1111
LBASize0[47:0]/ LBAMode[0]
Valid
1
2
3
4
5
6
(3) IP returns 4096-byte Identify controller data from SSD to user
(4) IP returns 4096-byte Identify namespace data from SSD to user
(6) LBASize (disk capacity) and LBAMode (LBA unit) are updated along with de-assertion of UserBusy

Figure 9: Identify command timing diagram

The details when running Identify command are shown as follows.

1) Set the active SSD (UserDevSel) and send Identify command (UserCmd="000" and UserReq='1').
2) The IP asserts UserBusy to '1' after running Identify command.
3) 4096-byte Identify controller data is returned to user. IdenWrAddr is equal to 0-255 with asserting IdenWrEn. Also, IdenWrData and IdenWrDWEn are valid at the same clock as IdenWrEn='1'.
4) 4096-byte Identify namespace data is returned. IdenWrAddr is equal to 256-511. IdenWrAddr[8] can be applied to check data type which is Identify controller data or Identify namespace data.
5) UserBusy is de-asserted to '0' after finishing the Identify command.
6) LBASize and LBAMode of the active SSD are simultaneously updated. Change UserDevSel and resend Identify command to the new SSD if some SSDs are not identified.

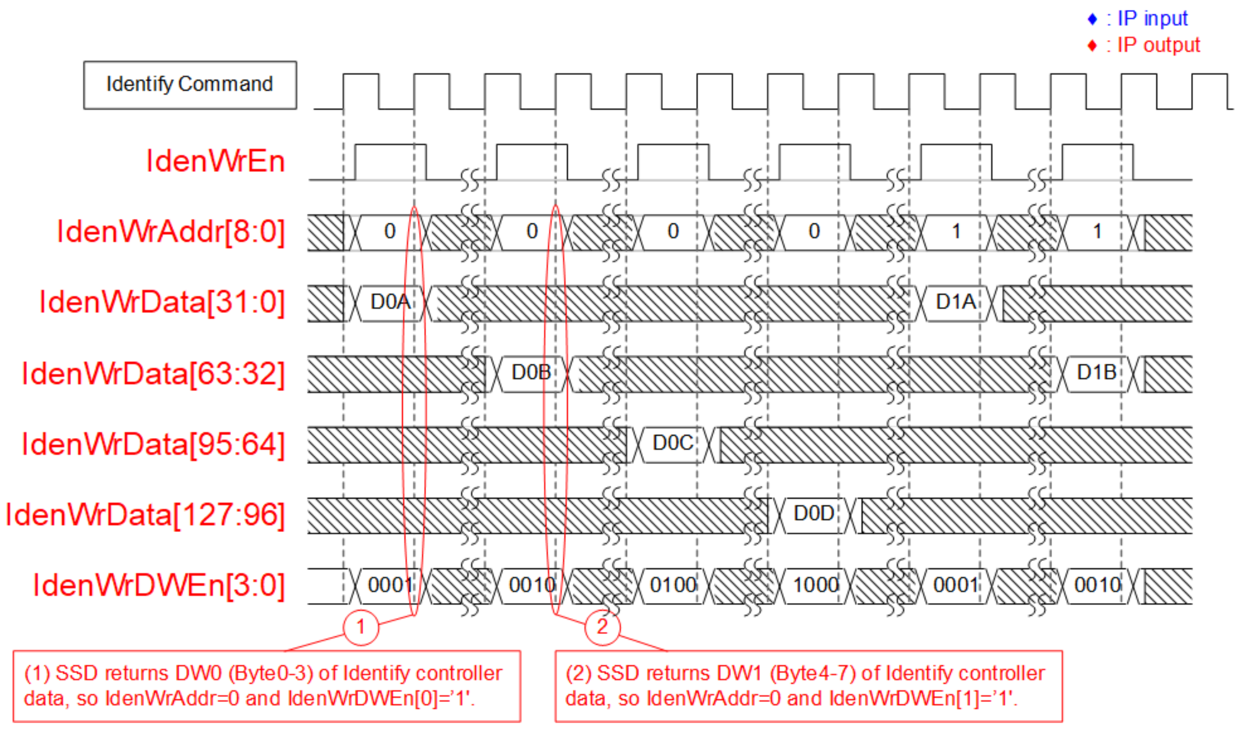

Figure 10: IdenWrDWEn timing diagram

IdenWrDWEn is 4-bit signal to be valid signal of 32-bit data. Some SSDs do not return 4-Kbyte Identify controller data and Identify namespace data continuously, but it returns only one dword (32-bit) at a time. Therefore, one bit of IdenWrDWEn is asserted to ‘1’ in the write cycle to write 32-bit data, as shown in Figure 10. IdenWrDWEn[0], [1], [2] and [3] corresponds to IdenWrData[31:0], [63:32], [95:64] and [127:96] respectively.

Shutdown

Shutdown command is recommended to send as the last command before the system is powered down. When Shutdown command is issued, SSD flushes the data from the internal cache to flash memory. After the command is issued, NVMeSW IP and SSD are not available until the system is powered down. If the SSD is powered down without Shutdown command, the total count of unsafe shutdowns, read by SMART command, is increased.

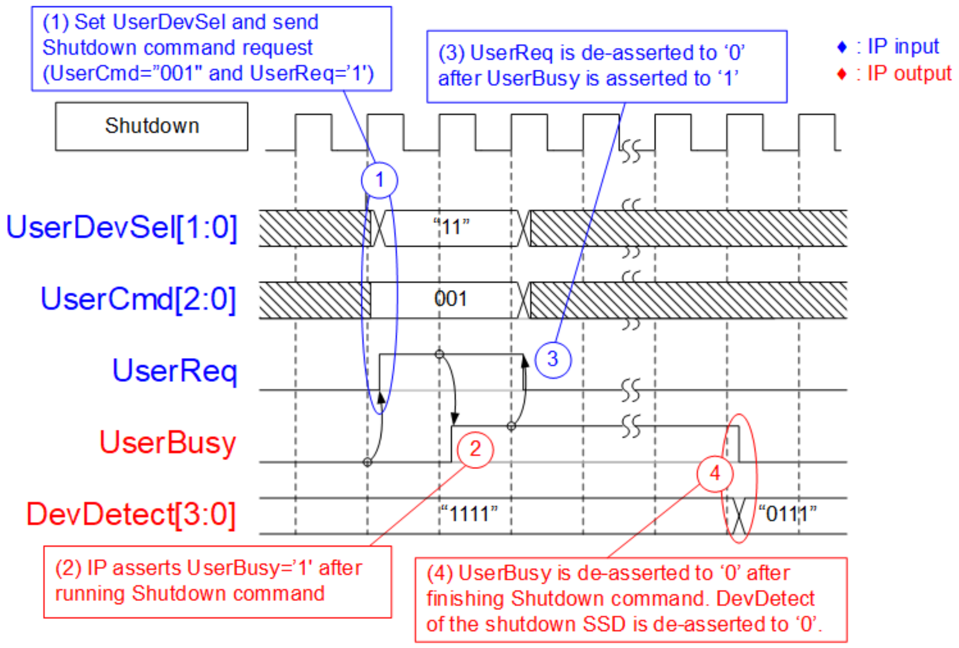
(1) Set UserDevSel and send Shutdown command request (UserCmd="001" and UserReq='1')
(3) UserReq is de-asserted to '0' after UserBusy is asserted to '1'
♦ : IP input
♦ : IP output
Shutdown
UserDevSel[1:0]
"11"
UserCmd[2:0]
001
UserReq
UserBusy
DevDetect[3:0]
"1111"
"0111"
1
2
3
4
(2) IP asserts UserBusy='1' after running Shutdown command
(4) UserBusy is de-asserted to '0' after finishing Shutdown command. DevDetect of the shutdown SSD is de-asserted to '0'.

Figure 11: Shutdown command timing diagram

The details when running Shutdown command are shown as follows.

1) Before sending the command request, the IP must be in the Idle state (UserBusy='0'). To send Shutdown command, UserDevSel is set to select the SSD and UserCmd is set to "001". UserReq is asserted to '1' with the stable value on UserDevSel and UserCmd.
2) UserBusy is asserted to '1' after NVMeSW IP runs Shutdown command.
3) UserReq is de-asserted to '0' to clear the current request. Next, user logic can change the value of UserDevSel and UserCmd for the next command request.
4) UserBusy is de-asserted to '0' when the selected SSD is completely shut down. DevDetect of selected SSD is de-asserted to '0'. The example of shutdown command is shown in Figure 11. DevDetect[0] changes to '0' after SSD#0 (UserDevSel="00") completes Shutdown operation. After that, user cannot send the command request to SSD#0.

SMART

SMART command is applied to check the SSD health. After sending SMART command, 512-byte health information is returned from the SSD. SMART command loads the parameters from CtmSubmDW0-DW15 signals on Custom command interface. User sets 16-dword data as constant value for SMART command. After that, the SMART data is returned via CtmRAM port as shown in Figure 12.

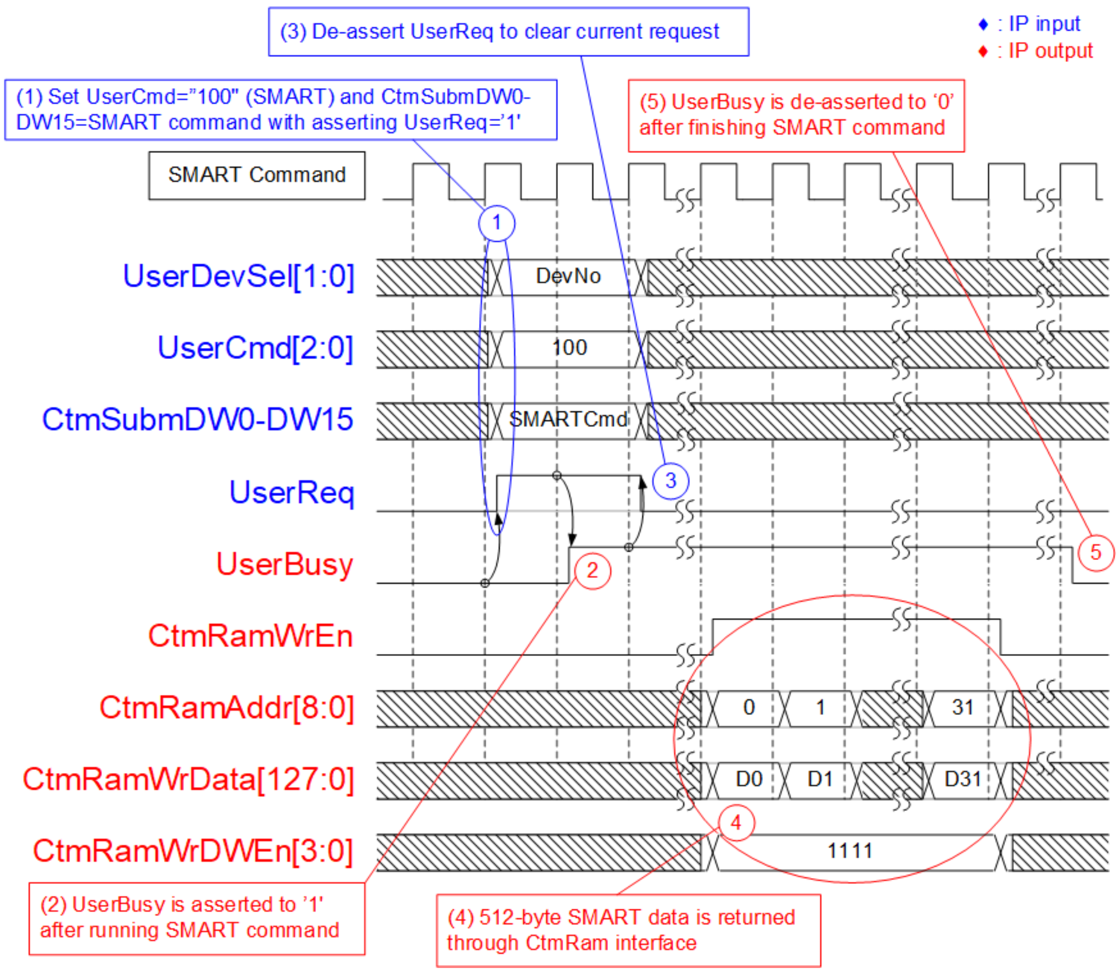
(3) De-assert UserReq to clear current request
♦ : IP input
♦ : IP output
(1) Set UserCmd="100" (SMART) and CtmSubmDW0-DW15=SMART command with asserting UserReq='1'
(5) UserBusy is de-asserted to '0' after finishing SMART command
SMART Command
UserDevSel[1:0]
DevNo
UserCmd[2:0]
100
CtmSubmDW0-DW15
SMARTCmd
UserReq
UserBusy
CtmRamWrEn
CtmRamAddr[8:0]
0
1
31
CtmRamWrData[127:0]
D0
D1
D31
CtmRamWrDWEn[3:0]
1111
(2) UserBusy is asserted to '1' after running SMART command
(4) 512-byte SMART data is returned through CtmRam interface

Figure 12: Example to send SMART command

The details when running SMART command are shown as follows.

1) Before sending the command request, the IP must be in the Idle state (UserBusy='0').
   All input parameters must be stable when UserReq is asserted to '1' for sending the request.
   CtmSubmDW0-DW15 is set as constant value by following value for SMART command.

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

2) Assert UserBusy to '1' after NVMeSW IP runs SMART command.
3) UserReq is de-asserted to '0' to clear the current request. Next, user logic can change the input parameters for the next command request.
4) 512-byte SMART data is returned on CtmRamWrData signal with asserting CtmRamWrEn to '1'. CtmRamAddr is equal to 0-31 to be data index of 512-byte data. When CtmRamAddr=0, byte0-15 of SMART data is valid on CtmRamWrData. CtmRamWrDWEn is dword enable for each 32-bit CtmRamWrData. If CtmRamWrDWEn="1111", all 128-bit CtmRamWrData are valid.
5) UserBusy is de-asserted to '0' when finishing SMART command.

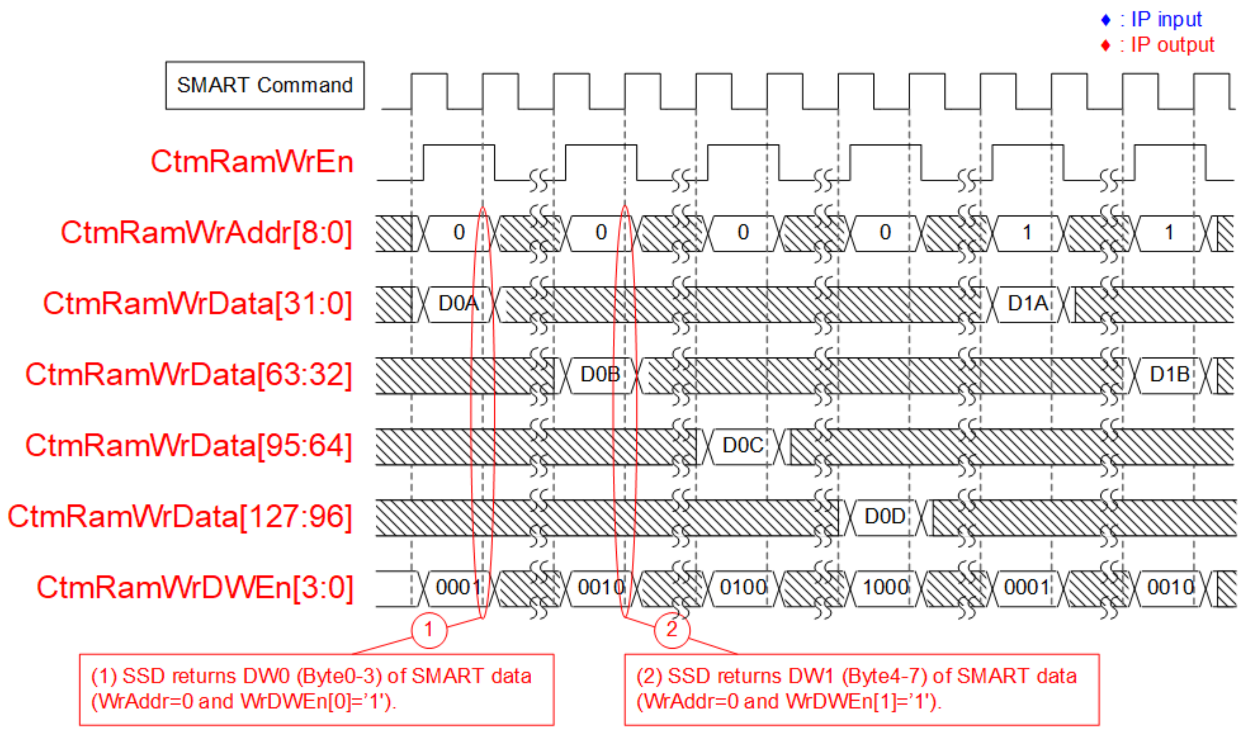

Figure 13: CtmRamWrDWEn timing diagram

Similar to Identify command, some SSDs do not return 512-byte data continuously but returns only one dword (32-bit) at a time. Therefore, one bit of CtmRamWrDWEn is asserted to '1' in the write cycle to be the valid signal of 32-bit CtmRamWrData. CtmRamWrDWEn[0], [1], [2] and [3] corresponds to CtmRamWrData[31:0], [63:32], [95:64] and [127:96] respectively.

Flush

Most SSDs accelerate write performance by storing write data to cache before flushing to the flash memory by the SSD controller. If power is down unexpectedly, the data in the cache may be lost and not stored to the flash memory. Flush command is the command to force the SSD controller to flush data from the cache. After sending Flush command, all data in previous Write command can be guaranteed.

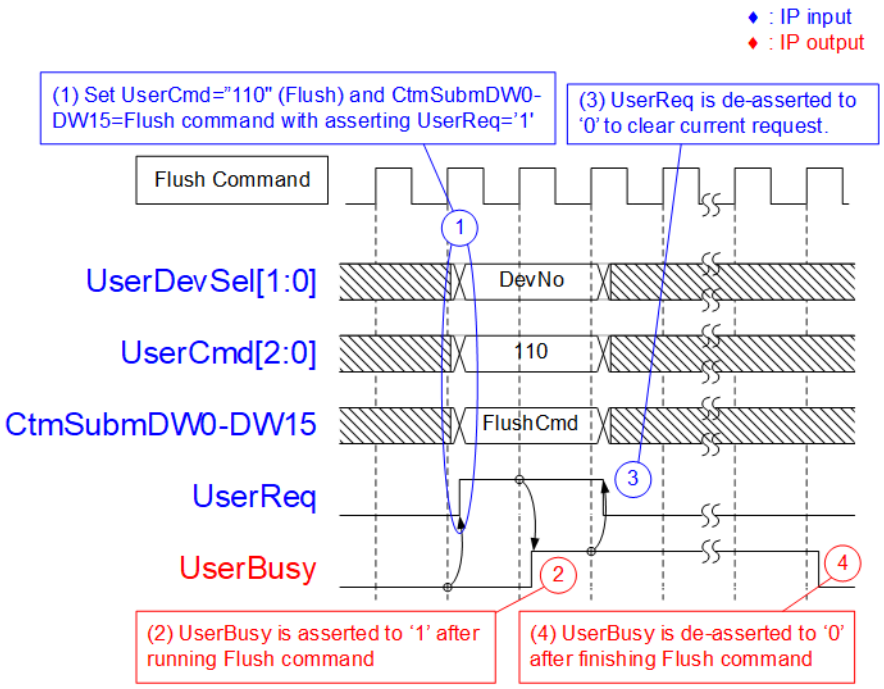
♦ : IP input
♦ : IP output
(1) Set UserCmd="110" (Flush) and CtmSubmDW0-DW15=Flush command with asserting UserReq='1'
(3) UserReq is de-asserted to '0' to clear current request.
Flush Command
UserDevSel[1:0]
DevNo
UserCmd[2:0]
110
CtmSubmDW0-DW15
FlushCmd
UserReq
UserBusy
1
2
3
4
(2) UserBusy is asserted to '1' after running Flush command
(4) UserBusy is de-asserted to '0' after finishing Flush command

Figure 14: Flush command timing diagram

The details for running Flush command are shown as follows.

1) Before sending the command request, the IP must be in the Idle state (UserBusy='0'). All input parameters must be stable when UserReq is asserted to '1' for sending the request. CtmSubmDW0-DW15 is set as constant value by following value for Flush command.

CtmSubmDW0	= 0x0000_0000
CtmSubmDW1	= 0x0000_0001
CtmSubmDW2 – CtmSubmDW15	= 0x0000_0000

2) UserBusy is asserted to '1' after NVMeSW IP runs Flush command.
3) UserReq is de-asserted to '0' to clear the current request. Next, user logic can change the input parameters for the next command request.
4) UserBusy is de-asserted to '0' when finishing Flush command.

Error

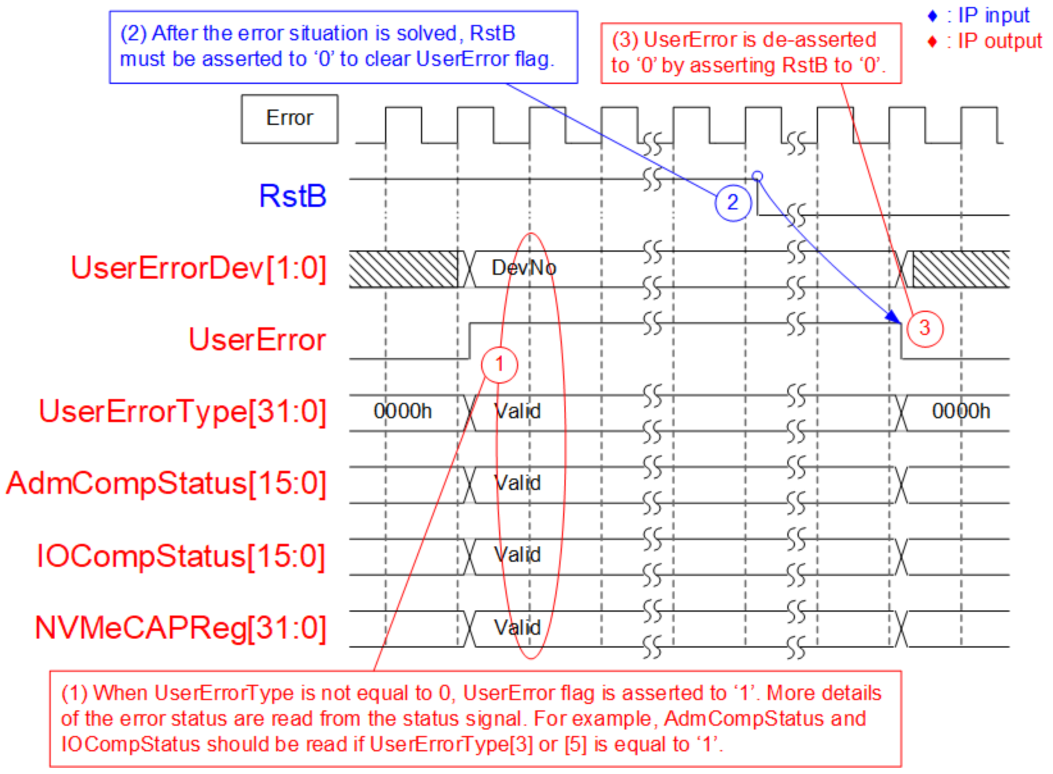
(2) After the error situation is solved, RstB must be asserted to '0' to clear UserError flag.
(3) UserError is de-asserted to '0' by asserting RstB to '0'.
♦ : IP input
♦ : IP output
Error
RstB
UserErrorDev[1:0]
DevNo
UserError
UserErrorType[31:0]
0000h
Valid
0000h
AdmCompStatus[15:0]
Valid
IOCompStatus[15:0]
Valid
NVMeCAPReg[31:0]
Valid
1
2
3
(1) When UserErrorType is not equal to 0, UserError flag is asserted to '1'. More details of the error status are read from the status signal. For example, AdmCompStatus and IOCompStatus should be read if UserErrorType[3] or [5] is equal to '1'.

Figure 15: Error flag timing diagram

When the error is found during running initialization process or operating some commands, UserError flag is asserted to '1'. UserErrorType is read to check the error type. NVMeCAPReg, AdmCompStatus and IOCompStatus are valid for monitoring error details after UserError is asserted to '1'.

When the error is found during running initialization process, it is recommended to read NVMeCAPReg to check capability of NVMe SSD. When the error is found during operating the command, it is recommended to read AdmCompStatus or IOCompStatus, depending on the command type, which is decoded from the received packet.

UserError flag is cleared by RstB signal only. After the failure is solved, RstB is asserted to '0' to clear the error flag.

Verification Methods

The NVMeSW IP Core functionality was verified by simulation and also proved on real board design by using KCU105/ZCU106/VCU118 evaluation 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 Gateway 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	26-Apr-2019	Initial Release
1.1	13-Feb-2020	Support AB18
1.2	12-Oct-2020	Update company info