Design Gateway Co.,Ltd
E-mail: ip-sales@design-gateway.com
URL: design-gateway.com

Core Facts

Provided with Core	
Documentation	Reference Design Manual Demo Instruction Manual
Design File Formats	Encrypted File
Instantiation Templates	VHDL
Reference Designs & Application Notes	Quartus Project, See Reference Design Manual
Additional Items	Demo on Agilex7 I-Series development kit
Support	
Support provided by Design Gateway Co., Ltd.	

Features

- Directly access NVMe Gen5 SSD without CPU and external memory
- Two data buffer modes: High speed (1 MB RAM) or Small memory (256 KB RAM)
- Simple user interface by dgIF typeS
- Support seven commands: Identify, Shutdown, Write, Read, SMART, Secure Erase, and Flush
- Supported NVMe devices
  - 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: A least half of the PCIe clock frequency (250 MHz for Gen5)
- PCIe Gen5 Hard IP: 256-bit interface using R-Tile Avalon-ST Intel FPGA for PCIe
- Available reference design
  - Agilex7 I-Series development board with AB19-M2PCI adapter board
- Customized service for additional NVMe commands such as TRIM and Sanitize

Table 1: Example Implementation Statistics

Family	Example Device	Buf Mode	Fmax (MHz)	Logic utilization (ALMs)	Registers	Pin	Block Memory bit	Design Tools
Agilex7 I-Series	AGIB027R29A1E2VR3	1 MB	375	10,820	22,668	-	8,701,696	Quartus 23.1
		256 KB	375	9,922	19,586	-	2,410,240	

Notes: Actual logic resource dependent on percentage of unrelated logic

Applications

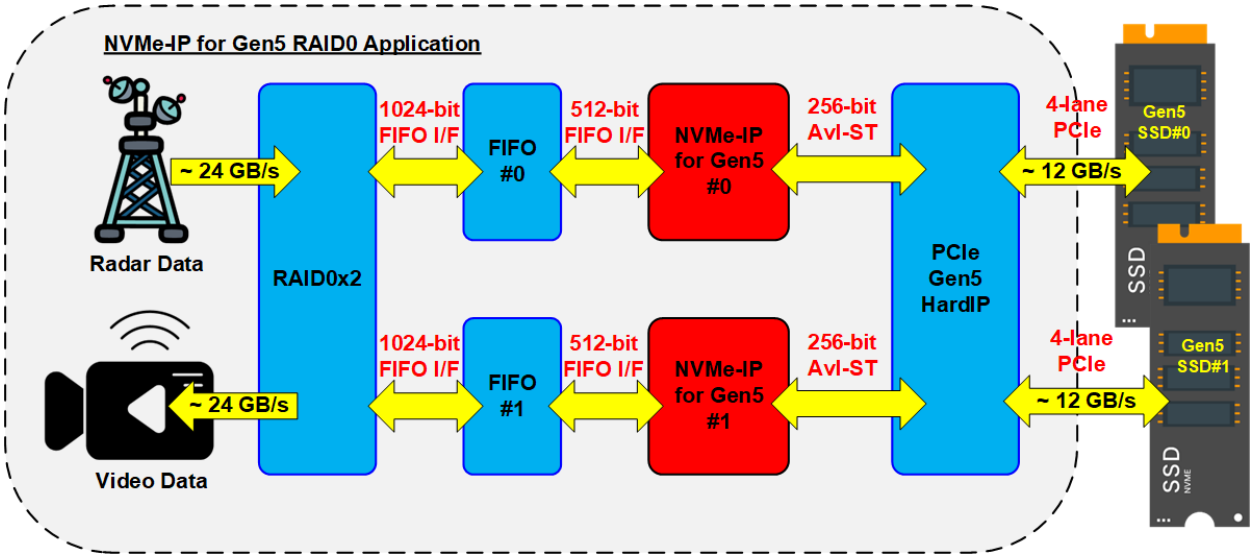

Figure 1: NVMe IP for Gen5 Application

The NVMe IP Core for Gen5 integrated with PCIe Gen5 hard IP (R-Tile Avalon-ST Intel FPGA for PCIe) provides an ideal solution for accessing NVMe Gen5 SSD without a CPU or external memory, including DDR. The NVMe IP core includes a Data buffer implemented by memory block to store transferred data between user logic and NVMe Gen5 SSD. Each PCIe hard IP supports 16-lane of PCIe Gen5 speed and it can be configured to be four 4-lane PCIe Gen5 interfaces, allowing up to four NVMe Gen5 SSDs and NVMe-IPs to be connected to each PCIe hard IP.

For example, Figure 1 demonstrates a 2-channel RAID0 system that uses two NVMe IPs to increase transfer speed up to two times. Based on the current reports of NVMe Gen5 SSDs on the market, the best SSD achieves 12 GB/s write speed and 13 GB/s read speed. By using two SSDs, the RAID0x2 user interface can increase the performance to 24 GB/s, which can accommodate high-speed data from multiple radar systems or high-resolution video data streams.

We also offer alternative IP cores for specific applications such as Multiple users, Random access, and PCIe switch.

Multiple User NVMe IP Core – Enables multiple users to access an NVMe SSD for high-performance write and read operation simultaneously.
https://dgway.com/muNVMe-IP_A_E.html
Random Access by Multiple User NVMe IP Core – Enables two users to write and read to the same NVMe SSD simultaneously, providing high random access performance for applications with non-contiguous storage requirements.
https://dgway.com/rmNVMe-IP_A_E.html
NVMe IP Core for PCIe Switch – Access multiple NVMe SSDs via PCIe switch to extend storage capacity. Enables high-speed write and read access to shared storage.
https://dgway.com/NVMe-IP_A_E.html

General Description

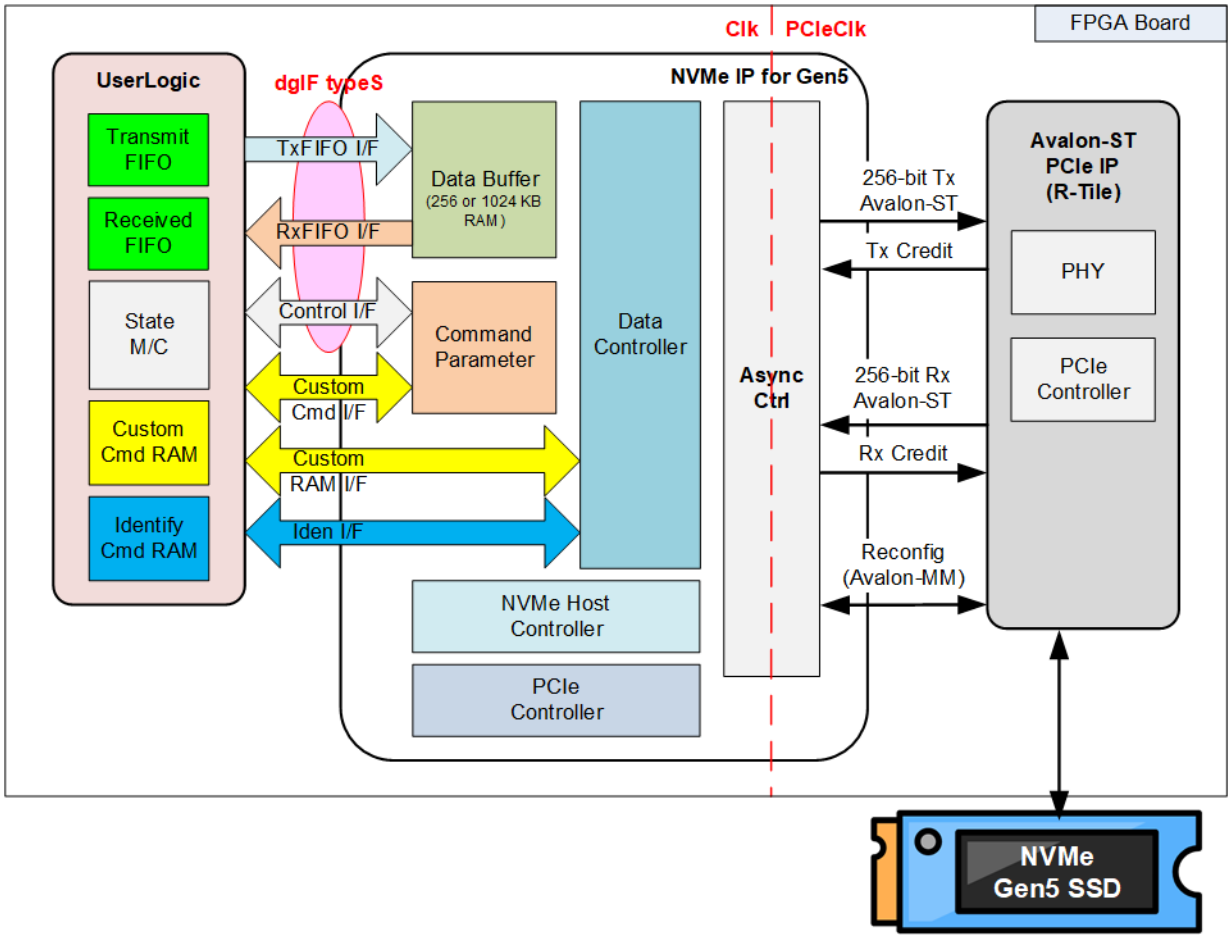

Figure 2: NVMe IP for Gen5 block diagram

The NVMe IP for Gen5 is a complete host controller solution that enables access to an NVMe SSD using the NVM express standard. The physical interface for the NVMe SSDs is PCIe, and the lower layer hardware is implemented using PCIe Gen5 hard IP (R-Tile Avalon-ST Intel FGPA IP for PCIe).

The NVMe IP core implements seven NVMe commands, including Identify, Shutdown, Write, Read, SMART, Secure Erase, and Flush command, and utilizes two user interface groups to transfer commands and data. The Control interface is used for transferring commands and their parameters, while the Data interface is used for transferring data when required by the command. For Write/Read commands, the Control interface and Data interface uses dgIF typeS, which is our standard interface for the storage. The Control interface of dgIF typeS includes start address, transfer length, and request signals, and the Data interface uses the standard FIFO interface.

SMART, Secure Erase, and Flush command are Custom commands that use the Ctm Cmd I/F for control path and Ctm RAM I/F for data path. Meanwhile, the Identify command uses its own data interface – Iden I/F, and the same Control interface as Write or Read command, as shown in Figure 2.

If abnormal conditions are detected during initialization or certain command operation, the NVMe IP may assert an error signal. The error status can be read from the IP for more details. Once the error cause is resolved, both the NVMe IP and SSD must be reset.

To ensure continuous packet transmission until the end of the packet on the user interface of PCIe hard IP, the user logic clock frequency must be equal to or greater than half of the PCIe clock frequency. This requires that data is valid every clock cycle between the start and the end of the frame. Using double data bus size with this user logic clock frequency limitation can guarantee that the bandwidth on the user interface is equal to or greater than PCIe hard IP bandwidth. For PCIe Gen5 speed, which configures the PCIe clock frequency to 500 MHz, the user logic clock must be at least 250MHz.

Overall, the NVMe IP for Gen5 provides a comprehensive solution for accessing NVMe SSDs. The IP core comes with reference designs on FPGA evaluation boards, allowing users to evaluate the product's capabilities before making a purchase.

Functional Description

The NVMe IP operation is divided into three phases, including IP initialization, Operating command, and Inactive status, as shown in Figure 3. Upon de-asserting the IP reset, the initialization phase begins, and the user should execute the Identify command to check the device status and capacity. During the Operating command phase, the user can perform write and read operations, execute Custom commands such as SMART, Secure Erase, and Flush. Finally, before shutting down the system, it is recommended to execute the Shutdown command to ensure safe operation.

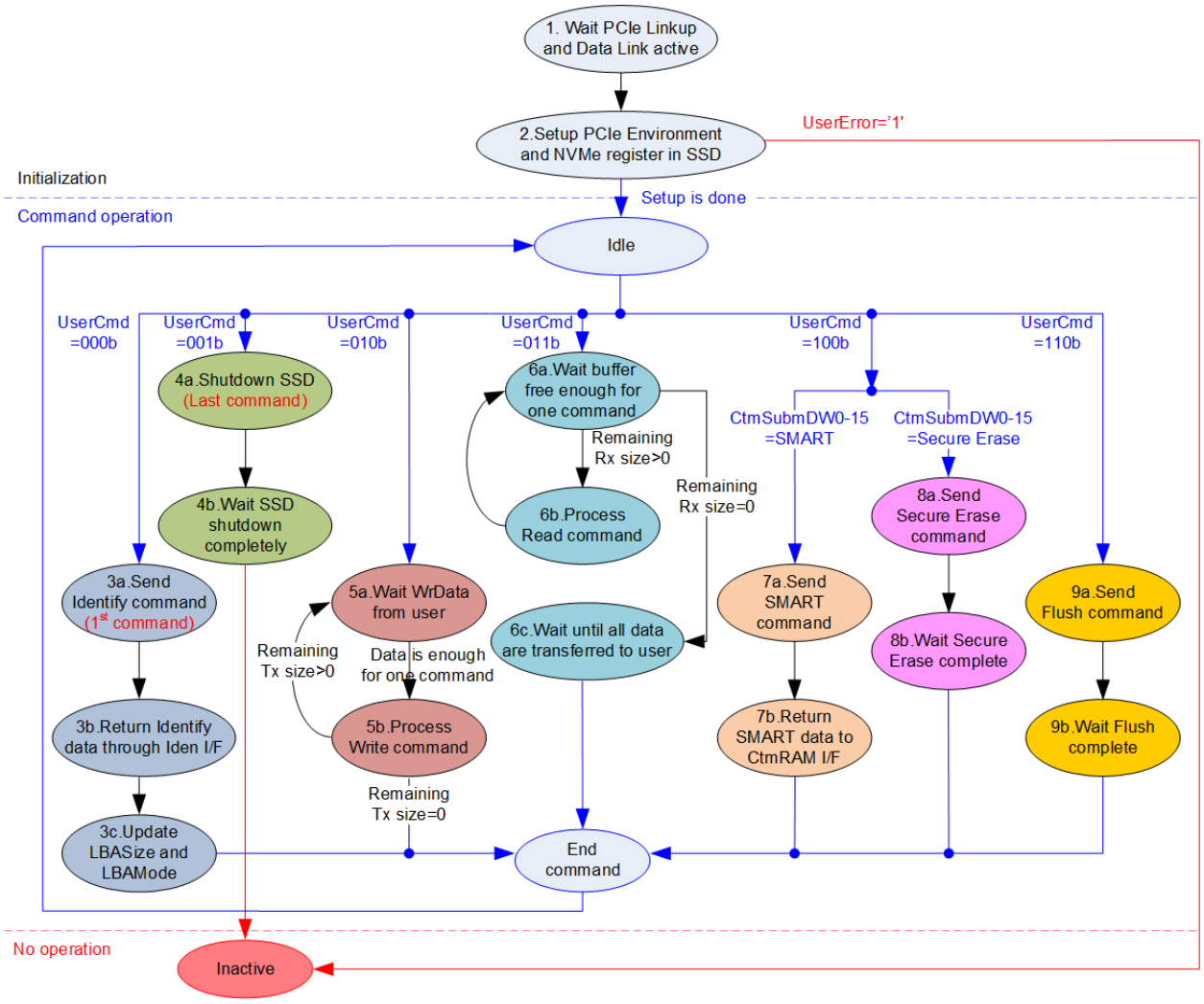

Figure 3: NVMe IP operation flow

The operation flow of the NVMe IP can be described in the following steps.

1) The IP waits for PCIe to be ready by monitoring the Linkup and Data Link active status from the PCIe IP core.
2) The IP begins the initialization process by initializing flow control, configurating PCIe and NVMe registers. After the initialization process, the IP enters to the Idle state, waiting for a new command request from the user. If any errors are detected during the initialization process, the IP enters the Inactive state, with UserError asserted to 1b.
3) The first command from the user must be the Identify command (UserCmd=000b), which updates the LBASize (disk capacity) and LBAMode (LBA unit=512 byte or 4 Kbyte).
4) The last command before powering down the system must be the Shutdown command (UserCmd=001b). This command is recommended to guarantee the SSD is powered down in a good sequence. Without the Shutdown command, the write data in the SSD cannot be guaranteed. After finishing the Shutdown command, the NVMe IP and SSD change to the Inactive state. The new command cannot be operated until the IP is reset.
5) For the Write command (UserCmd=010b), the maximum data size of each command is 128 Kbyte. If the total length from the user is more than 128 Kbytes, the IP repeats the following steps, 5a) – 5b), automatically until all the data is completely transferred.
   a) The IP waits until the write data, sent by the user, is enough for one command (transfer size of one command in NVMe IP is 128 Kbytes, except the last loop which could be less than 128 Kbytes).
   b) The IP sends the Write command to the SSD and then waits until the status is returned from the SSD. The IP returns to the Idle state when all the data is completely transferred. Otherwise, the IP goes back to step 5a) to send the next Write command.
6) Similar to the Write command, when executing the Read command (UserCmd=011b) which has a transfer size more than 128 Kbytes, the IP must repeat the following steps, 6a) – 6b), many times.
   a) If the remaining transfer size is equal to zero, the IP skips to step 6c). Otherwise, the IP waits until there is enough free space in the Data buffer of the NVMe IP for one command (128 Kbytes or remaining transfer size for the last loop).
   b) The IP sends the Read command to the SSD and then returns to step 6a).
   c) The IP waits until all the data is completely transferred from the Data buffer to the user logic and then returns to the Idle state. Therefore, the Data buffer is empty after finishing the Read command.
7) For the SMART command (UserCmd=100b and CtmSubmDW0-15=SMART), 512-byte data is returned after finishing the operation.
   a) The IP sends the Get Log Page command to read SMART/Health information from the SSD.
   b) The 512-byte data is returned from the SSD, and the IP forwards the data through the Custom command RAM interface (CtmRamAddr=0x000 – 0x01F).
8) For the Secure Erase command (UserCmd=100b and CtmSubmDW0-15=Secure Erase), there is no data transferring during the operation.
   a) The IP sends the Secure Erase command to the SSD.
   b) The IP waits until the SSD returns the status to complete the operation.
9) For Flush command (UserCmd=110b), there is also no data transferring during the operation.
   a) The IP sends Flush command to the SSD.
   b) The IP waits until the SSD returns the status to complete the operation.

The design of an NVMe host controller consists of two protocols: NVMe and PCIe. The NVMe protocol is used to interface with the user, while the PCIe protocol is used to interface with PCIe hard IP. Figure 2 shows the hardware inside the NVMe IP which is split into two groups, NVMe and PCIe.

NVMe

The NVMe group supports seven commands, which are split into two types - Admin commands and NVM commands. Admin commands include Identify, Shutdown, SMART, and Secure Erase, while NVM commands include Write, Read, and Flush. After executing a command, the status returned from the SSD is latched to AdmCompStatus (status returned from Admin commands) or IOCompStatus (status returned from NVM commands), depending on the command type.

The parameters of Write or Read command are set by the Control interface of dgIF typeS, while the parameters of SMART, Secure Erase, or Flush command are set by CtmSubmDW0-15 of Ctm Cmd interface. The Data interface for Write or Read command is transferred by the FIFO interface, a part of dgIF typeS. The data for Write and Read commands are stored in the Data buffer inside the IP. The Data interface of other commands has its own interface - Identify I/F for Identify command and Custom RAM I/F for SMART command.

- **NVMe Host Controller**

  The NVMe host controller is the core controller in the NVMe IP. The controller has two phases for operating: initialization phase and command operation phase. The initialization phase is run once after the system is booted up to configure the NVMe register inside the SSD. Once the initialization phase is complete, it enters the command operation phase. In this phase, the controller controls the order of transmitted packet and received packet for each command. To start operating each command, the parameters of the command are latched to Command Parameter for creating the packet. After that, the packet is forwarded to AsyncCtrl to convert NVMe packet to PCIe packet. After each command operation is done, the status packet is returned from the SSD. The controller decodes the status value and checks whether the operation is complete or error. If the command needs to transfer data, such as Write or Read commands, the controller must handle the order of data packets created and decoded by Data controller.

- **Command Parameter**

  The Command Parameter module creates the command packet sent to the SSD, and also decode the status packet returned from the SSD. The input and output of this module are controlled by the NVMe host controller. Typically, a command consists of 16 Dwords (1 Dword = 32-bit). When executing Identify, Shutdown, Write, and Read commands, all 16 Dwords are created by Command parameters that are initialized by the user inputs on dgIF typeS. When executing SMART, Secure Erase, and Flush commands, all 16 Dwords are directly loaded via CtmSubmDW0-CtmSubmDW15 of Ctm Cmd interface.

- **Data Buffer**

  Two data buffer modes are supported: High speed mode, which uses 1 Mbyte RAM, and Small memory mode, which uses 256 Kbyte RAM. The RAM is implemented using memory blocks. The buffer stores data for transferring with the SSD while operating Write and Read commands.

- **Data Controller**

  The Data Controller module is used when the command must transfer data such as Identify, SMART, Write, and Read commands. There are three data interfaces for transferring with the SSD – the FIFO interface with the Data buffer when executing Write or Read command, the Custom RAM interface when executing SMART command, and the Identify interface when executing Identify command. The data packet is created and decoded by this module. Similar to the Command Parameter module, the input and output signals of the Data controller module are controlled by the NVMe host controller.

PCIe

The PCIe protocol is the outstanding low-layer protocol for the high-speed application, and the NVMe protocol runs over it. Therefore, the NVMe layer can be operated after the PCIe layer completes the initialization. Two modules are designed to support the PCIe protocol - PCIe controller and AsyncCtrl.

- **PCIe Controller**

  In initialization process, the PCIe controller initializes flow control through Tx/Rx Credit port and sets up the PCIe environment of the SSD via the Reconfig port, which is the Avalon-MM interface. After that, the PCIe packet is created and decoded through the 256-bit Tx/Rx Avalon-Stream. The data flow in the Tx/Rx Avalon-Stream is controlled through Credit Control via the Tx/Rx Credit port. The PCIe controller converts the command packet and data packet from the NVMe module into a PCIe packet, and vice versa.

- **AsyncCtrl**

  AsyncCtrl includes asynchronous registers and asynchronous buffers with an asymmetric data width to facilitate clock domain crossing and different data width. The data width on the user interface is twice the data width on the PCIe interface. Therefore, the user clock frequency can be equal to or more than half value of the PCIe clock frequency to balance the data bandwidth. Most logics in the NVMe IP run on the user clock domain, while the PCIe hard IP runs on the PCIe clock domain.

User Logic

The user logic could be designed using a small state machine that sends commands and parameters for each command. For instance, simple registers are used to define the parameters for the Write or Read command such as the address and transfer size. Two FIFOs are connected to transfer data for the Write and Read commands individually. When operating the SMART and Identify commands, each data output interface connects to a simple dual port RAM with byte enable. The FIFO and RAM have a data width of 512-bit while the memory depth can be set to different values. The Identify command has a data size of 8 Kbytes, while the SMART command has a data size of 512 bytes.

PCIe Hard IP (R-Tile Avalon-ST Intel FPGA for PCIe)

The NVMe IP requires to connect with the PCIe hard IP (R-Tile Avalon-ST Intel FPGA for PCIe). The PCIe hard IP consists of five interfaces: Avalon-ST RX Port (rx_st0/1_*) for receiving packets, Rx Credit Port (rx_st_*) for Rx flow control, Avalon-ST Tx Port (tx_st#_*) for transmitting packets, Tx Credit Port (tx_st_*) for Tx flow control, and Hard IP Reconfiguration (hip_reconfig_*) for PCIe configuration.

The PCIe hard IP implements the Transaction layer, Data Link layer, and Physical layer of the PCIe protocol. The number of SSDs that can connect to one FPGA device is limited by the number of PCIe hard IP blocks. Each PCIe hard IP supports up to 16-lane PCIe Gen5 interface, which can be configured to four 4-lane PCIe Gen5 interfaces. Therefore, each PCIe hard IP can connect up with four NVMe IPs and four NVMe Gen5 SSDs. For more details about the PCIe hard IP, please refer to the R-Tile Avalon-ST Intel FPGA for PCIe document available on the Intel website.

https://www.intel.com/content/www/us/en/docs/programmable/683501/

Core I/O Signals

Descriptions of Core parameters and the I/O signals are provided in Table 2 - Table 5.

Table 2: Core Parameters

Name	Value	Description
BufMode	0 or 1	Data buffer mode. 1-High speed mode by using 1 MB buffer, 0-Small memory mode by using 256 KB buffer

Table 3: User logic I/O Signals (Synchronous to Clk signal)

Signal	Dir	Description
Control I/F of dgIF typeS		
RstB	In	Synchronous reset. Active low. It should be de-asserted to 1b when the Clk signal is stable.
Clk	In	User clock to run the NVMe IP. The frequency of this clock must be equal to or greater than half of the PCIeClk frequency, which is the clock output from the PCIe hard IP. For example, when using a 500 MHz PCIe hard IP for Gen5 speed, the Clk frequency must be equal to or greater than 250MHz.
UserCmd[2:0]	In	User Command. Valid when UserReq=1b. The possible values are 000b: Identify, 001b: Shutdown, 010b: Write SSD, 011b: Read SSD, 100b: SMART/Secure Erase, 110b: Flush, 101b/111b: Reserved
UserAddr[47:0]	In	The start address to write/read from the SSD in 512-byte units. Valid when UserReq=1b. If the LBA unit = 4 Kbyte, then UserAddr[2:0] must always be set to 000b to align with 4 Kbyte unit. If the LBA unit = 512 byte, it is recommended to set UserAddr[2:0]=000b to align with 4 Kbyte size (SSD page size). The write/read performance of most SSDs is reduced when the start address is not aligned with page size.
UserLen[47:0]	In	The total transfer size to write/read from the SSD in 512-byte units. Valid from 1 to (LBASize-UserAddr). If the LBA unit = 4 Kbyte, then UserLen[2:0] must always be set to 000b to align with the 4 Kbyte unit. Valid when UserReq=1b.
UserReq	In	Assert to 1b to send a new command request and de-assert to 0b after the IP starts the operation by asserting UserBusy to 1b. This signal can only be asserted when the IP is Idle (UserBusy=0b). Command parameters (UserCmd, UserAddr, UserLen, and CtmSubmDW0-DW15) must be valid and stable when UserReq=1b. UserAddr and UserLen are inputs for the Write/Read command, while CtmSubmDW0-DW15 are inputs for SMART/Secure Erase/Flush command.
UserBusy	Out	Asserted to 1b when the IP is busy. A new request must not be sent (UserReq to 1b) when the IP is busy.
LBASize[47:0]	Out	The total capacity of the SSD in 512-byte units. Default value is 0. This value is valid after finishing the Identify command.
LBAMode	Out	The LBA unit size. 0b: 512byte, 1b: 4 Kbyte. Default value is 0. This value is valid after finishing the Identify command.
UserError	Out	Error flag. Asserted to 1b when the UserErrorType is not equal to 0. The flag is de-asserted to 0b by asserting RstB to 0b.

Signal	Dir	Description
Control I/F of dgIF typeS		
UserErrorType[31:0]	Out	Error status. [0] – Error when the PCIe class code is incorrect. [1] – Error from Controller capabilities (CAP) register, which can be caused from various reasons. - Memory Page Size Minimum (MPSMIN) is not equal to 0. - NVM command set flag (bit 37 of CAP register) is not set to 1. - Doorbell Stride (DSTRD) is not equal to 0. - Maximum Queue Entries Supported (MQES) is less than 15. More details of each register can be found in the NVMeCAPReg signal. [2] – Error when the Admin completion entry is not received until timeout. [3] – Error when the status register in the Admin completion entry is not 0 or when the phase tag/command ID is invalid. More details can be found in the AdmCompStatus signal. [4] – Error when the IO completion entry is not received until timeout. [5] – Error when the status register in the IO completion entry is not 0 or when the phase tag is invalid. More details can be found in the IOCompStatus signal. [6] – Error from an unsupported LBA unit, which is not equal to 512 bytes or 4 Kbytes. [7] – Reserved [8] – Error when receiving TLP packet with an incorrect size. [9] – Reserved Bit[15:10] are mapped to Uncorrectable Error Status Register [10] – Mapped to Unsupported Request Error Status (bit[20]). [11] – Mapped to Completer Abort Status (bit[15]). [12] – Mapped to Unexpected Completion Status (bit[16]). [13] – Mapped to Completion Timeout Status (bit[14]). [14] – Mapped to Poisoned TLP Received Status (bit[12]). [15] – Mapped to ECRC Error Status (bit[19]). [23:16] - Reserved Bit[30:24] are also mapped to Uncorrectable Error Status Register [24] – Mapped to Data Link Protocol Error Status (bit[4]). [25] – Mapped to Surprise Down Error Status (bit[5]). [26] – Mapped to Receiver Overflow Status (bit[17]). [27] – Mapped to Flow Control Protocol Error Status (bit[13]). [28] – Mapped to Uncorrectable Internal Error Status (bit[22]). [29] – Mapped to Malformed TLP Status (bit[18]). [30] – Mapped to ACS Violation Status (bit[21]). [31] – Reserved *Note: Timeout period of bit[2]/[4] is set from TimeOutSet input.*

Signal	Dir	Description
Data I/F of dgIF typeS		
UserFifoWrCnt[15:0]	In	Write data counter of the Receive FIFO. Used to check the full status of the FIFO. If the FIFO is full, the returned data transmission from the Read command may be paused. If the size of FIFO data count is less than 16 bits, the remaining upper bits must be filled with 1b.
UserFifoWrEn	Out	Asserted to 1b to write data to the Receive FIFO when executing the Read command.
UserFifoWrData[511:0]	Out	Write data bus of the Receive FIFO. Valid when UserFifoWrEn=1b.
UserFifoRdCnt[15:0]	In	Read data counter of the Transmit FIFO. Used to check the data size stored in the FIFO. The transmitted data packet for the Write command may be paused when the counter shows an empty status. If the FIFO data count is less than 16 bits, the remaining upper bis must be filled with 0b.
UserFifoEmpty	In	Unused for this IP.
UserFifoRdEn	Out	Asserted to 1b to read data from the Transmit FIFO when executing the Write command.
UserFifoRdData[511:0]	In	Read data returned from the Transmit FIFO. Valid in the next clock after UserFifoRdEn is asserted to 1b.
NVMe IP Interface		
IPVesion[31:0]	Out	IP version number
TestPin[127:0]	Out	Reserved to be the IP Test point.
TimeOutSet[31:0]	In	Timeout value to wait for completion from SSD. The time unit is equal to 1/(Clk frequency). When TimeOutSet is equal to 0, Timeout function is disabled.
AdmCompStatus[15:0]	Out	Status output from Admin Completion Entry [0] – Set to 1b when the 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 1b when the 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 1b. [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
Identify Interface		
IdenWrEn	Out	Asserted to 1b for sending data output from the Identify command.
IdenWrDWEn[15:0]	Out	Dword (32-bit) enable of IdenWrData. Valid when IdenWrEn=1b. 1b: This Dword data is valid, 0b: This Dword data is not available. Bit[0], [1], …, [15] correspond to IdenWrData[31:0], [63:32], …, [511:480], respectively.
IdenWrAddr[6:0]	Out	Index of IdenWrData in 512-bit unit. Valid when IdenWrEn=1b. 0x00-0x3F: 4Kbyte Identify controller data, 0x40-0x7F: 4Kbyte Identify namespace data.
IdenWrData[511:0]	Out	4Kbyte Identify controller data or Identify namespace data. Valid when IdenWrEn=1b.

Signal	Dir	Description
Custom Interface (Command and RAM)		
CtmSubmDW0[31:0] – CtmSubmDW15[31:0]	In	16 Dwords of Submission queue entry for SMART/Secure Erase/Flush command. DW0: Command Dword0, DW1: Command Dword1, …, and DW15: Command Dword15. These inputs must be valid and stable when UserReq=1b and UserCmd=100b (SMART/Secure Erase) or 110b (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 1b for sending data output from Custom command such as SMART command.
CtmRamWrDWEn[15:0]	Out	Dword (32-bit) enable of CtmRamWrData. Valid when CtmRamWrEn=1b. 1b: This dword data is valid, 0b: This dword data is not available. Bit[0], [1], …, [15] corresponds to CtmRamWrData[31:0], [63:32], …, [511:480], respectively.
CtmRamAddr[6:0]	Out	Index of CtmRamWrData when SMART data is received. Valid when CtmRamWrEn=1b. (Optional) Index to request data input through CtmRamRdData for customized Custom commands.
CtmRamWrData[511:0]	Out	512-byte data output from SMART command. Valid when CtmRamWrEn=1b.
CtmRamRdData[511:0]	In	(Optional) Data input for customized Custom commands.

Table 4: Physical I/O Signals for PCIe Hard IP (Synchronous to PCIeClk signal)

Signal	Dir	Description
PCIe System signal		
PCIeRstB	In	Synchronous reset signal. Active low. De-assert to 1b when PCIe hard IP is not in reset state.
PCIeClk	In	Clock output from PCIe hard IP. It is recommended to configure as 500 MHz to achieve the best performance. Using lower clock speed reduces the PCIe interface bandwidth which may show the lower performance.
DlUp	In	Assert to 1b when Data Link layer of PCIe hard IP is ready for data transfer.
PCIe Hard IP Rx Interface		
PCIeRxReady	Out	Asserted to 1b to indicate that the NVMe IP is ready to accept data. This signal is always set to 1b. The Rx flow control is handled through the Rx Credit Interface.
PCIeRxHdValidL	In	Assert to 1b to indicate that PCIeRxHdL is valid.
PCIeRxDValidL	In	Assert to 1b to indicate that PCIeRxDataL is valid.
PCIeRxSOPL	In	Assert to 1b to indicate that this is the first cycle of the TLP starting in the lower segment. Valid when PCIeRxHdValidL or PCIeRxDValidL is asserted to 1b.
PCIeRxEOPL	In	Assert to 1b to indicate that this is the last cycle of the TLP ending in the lower segment. Valid when PCIeRxHdValidL or PCIeRxDValidL is asserted to 1b.
PCIeRxHdL[127:0]	In	Received header data in the lower segment. Valid when PCIeRxHdValidL and PCIeRxSOPL are asserted to 1b.
PCIeRxDataL[127:0]	In	Received data in the lower segment. Valid when PCIeRxDValidL is asserted to 1b.
PCIeRxEmptyL[1:0]	In	Specific the number of empty Dwords of PCIeRxDataL when the TLP ends in the lower segment. Valid when PCIeRxDValidL and PCIeRxEOPL are asserted to 1b.
The signals in the upper segment have the same description as the signals in the lower segment. The upper segment signals include PCIeRxHdValidH, PCIeRxDValidH, PCIeRxSOPH, PCIeRxEOPH, PCIeRxHdH[127:0], PCIeRxDataH[127:0], and PCIeRxEmptyH[1:0].		

Signal	Dir	Description
PCIe Hard IP Tx Interface		
PCIeTxReady	In	Assert to 1b to indicate that the PCIe hard IP is ready to receive data. If it is de-asserted to 0b, PCIeTxHdValidL/H and PCIeTxDValidL/H must be de-asserted to 0b within 16 clock cycles. When PCIeTxReady is re-asserted to 1b, PCIeTxDValidL/H must be re-asserted to 1b within 1 clock cycle to continue packet transmission.
PCIeTxHdValidL	Out	Asserted to 1b to indicate that PCIeTxHdL is valid. When PCIeTxReady is de-asserted, this signal must be de-asserted within 16 clock cycles.
PCIeTxDValidL	Out	Asserted to 1b to indicate that PCIeTxDataL is valid. When PCIeTxReady is de-asserted, this signal must be de-asserted within 16 clock cycles. When PCIeTxReady is re-asserted, this signal must be re-asserted within 1 clock cycle to continue packet transmission.
PCIeTxSOPL	Out	Asserted to 1b to indicate the first cycle of a TLP in the lower segment. Valid when PCIeTxHdValidL or PCIeTxDValidL is asserted to 1b.
PCIeTxEOPL	Out	Assert to 1b to indicate the last cycle of a TLP ending in the lower segment. Valid when PCIeTxHdValidL or PCIeTxDValidL is asserted to 1b.
PCIeTxHdL[127:0]	Out	Transmitted header data in the lower segment. Valid when PCIeTxHdValidL and PCIeTxSOPL are asserted to 1b.
PCIeTxDataL[127:0]	Out	Transmitted data in the lower segment. Valid when PCIeTxDValidL is asserted to 1b.
The signals in the upper segment have the same description as the signals in the lower segment. The upper segment signals include PCIeTxHdValidH, PCIeTxDValidH, PCIeTxSOPH, PCIeTxEOPH, PCIeTxHdH[127:0], and PCIeTxDataH[127:0].		
PCIe Hard IP Rx Credit Interface		
RxHCdInit[2:0]	Out	Asserted to 1b to request and indicate initialization phase of each Rx header credit type. De-assertion from 1b to 0b indicates the completion of the header credit initialization. [0] – Posted Header [1] – Non-Posted Header [2] – Completion Header
RxHCdAck[2:0]	In	The response of RxHCdInit. Assert to 1b to indicate that PCIe hard IP is ready for initialization of the Rx header credit. Bit[0], [1], and [2] correspond to RxHCdInit [0], [1], and [2], respectively.
RxHCdCntValid[2:0]	Out	Valid of RxHCdCnt to release credit. Bit[0], [1], and [2] correspond to RxHCdCnt[1:0], [3:2], and [5:4], respectively.
RxHCdCnt[5:0]	Out	Number of released header credit in the Rx side. [1:0] – Posted Header, valid when RxHCdCntValid[0] is asserted to 1b. [3:2] – Non-Posted Header, valid when RxHCdCntValid[1] is asserted to 1b. [5:4] – Completion Header, valid when RxHCdCntValid[2] is asserted to 1b.
RxDCdInit[2:0]	Out	Asserted to 1b to request and indicate initialization phase of each Rx data credit type. De-assertion from 1b to 0b indicates the completion of the data credit initialization. [0] – Posted Data [1] – Non-Posted Data [2] – Completion Data
RxDCdAck[2:0]	In	The response of RxDCdInit. Assert to 1b to indicate that the PCIe hard IP is ready for initialization of the Rx data credit. Bit[0], [1], and [2] correspond to RxDCdInit [0], [1], and [2], respectively.

Signal	Dir	Description
PCIe Hard IP Rx Credit Interface		
RxDCdCntValid[2:0]	Out	Valid of RxDCdCnt to release credit. Bit[0], [1], and [2] correspond to RxDCdCnt[3:0], [7:4], and [11:8], respectively.
RxDCdCnt[11:0]	Out	Number of released data credit in the Rx side. [3:0] – Posted Data, valid when RxDCdCntValid[0] is asserted to 1b. [7:4] – Non-Posted Data, valid when RxDCdCntValid[1] is asserted to 1b. [11:8] – Completion Data, valid when RxDCdCntValid[2] is asserted to 1b.
PCIe Hard IP Tx Credit Interface		
TxHCdInit[2:0]	In	Assert to 1b to request and indicate initialization phase of each Tx header credit type. De-assertion from 1b to 0b indicates the completion of the header credit initialization. [0] – Posted Header [1] – Non-Posted Header [2] – Completion Header
TxHCdAck[2:0]	Out	The response of TxHCdInit. Asserted to 1b to indicate that NVMe IP is ready for initialization of the Tx header credit. Bit[0], [1], and [2] correspond to TxHCdInit [0], [1], and [2], respectively.
TxHCdCntValid[2:0]	In	Valid of TxHCdCnt to release credit. Bit[0], [1], and [2] correspond to TxHCdCnt[1:0], [3:2], and [5:4], respectively.
TxHCdCnt[5:0]	In	Number of released header credit in the Tx side. [1:0] – Posted Header, valid when TxHCdCntValid[0] is asserted to 1b. [3:2] – Non-Posted Header, valid when TxHCdCntValid[1] is asserted to 1b. [5:4] – Completion Header, valid when TxHCdCntValid[2] is asserted to 1b.
TxDCdInit[2:0]	In	Assert to 1b to request and indicate initialization phase of each Tx data credit type. De-assertion from 1b to 0b indicates the completion of the data credit initialization. [0] – Posted Data [1] – Non-Posted Data [2] – Completion Data
TxDCdAck[2:0]	Out	The response of TxDCdInit. Asserted to 1b to indicate that NVMe IP is ready for initialization of the Tx data credit. Bit[0], [1], and [2] correspond to TxDCdInit [0], [1], and [2], respectively.
TxDCdCntValid[2:0]	In	Valid of TxDCdCnt to release credit. Bit[0], [1], and [2] correspond to TxDCdCnt[3:0], [7:4], and [11:8], respectively.
TxDCdCnt[11:0]	In	Number of released data credit in the Tx side. [3:0] – Posted Data, valid when TxDCdCntValid[0] is asserted to 1b. [7:4] – Non-Posted Data, valid when TxDCdCntValid[1] is asserted to 1b. [11:8] – Completion Data, valid when TxDCdCntValid[2] is asserted to 1b.

Table 5: PCIe Hard IP Reconfiguration and Link Status Signals (Synchronous to CfgClk signal)

Signal	Dir	Description
System and Link status signal		
CfgRstB	In	Synchronous reset signal. Active low. De-assert this signal to 1b when the PCIe hard IP is not in reset state.
CfgClk	In	Clock output from the PCIe hard IP. This clock value can be configured via divide-by-2 or divide-by-4 from PCIeClk.
PCIeLinkup	In	Asserted to 1b when LTSSM state of the PCIe hard IP is in L0 State.
LinkSpeed[2:0]	Out	Negotiated PCIe link speed that is read out from the PCIe Hard IP Reconfiguration Space. 000b: Undefined, 001b: Gen1, 010b: Gen2, 011b: Gen3, 100b: Gen4, 101b: Gen5, 110b/111b: Reserved. This register is valid after the NVMe IP has finished initialization (UserBusy=0b).
LinkWidth[1:0]	Out	Negotiated PCIe link width that is read out from the PCIe Hard IP Reconfiguration Space. 00b: Undefined, 01b: x1 lane, 10b: x2 lane, 11b: x4 lane. This register is valid after the NVMe IP has finished initialization (UserBusy=0b).
Reconfiguration Interface		
CfgAddr[31:0]	Out	Reconfiguration address for writing or reading.
CfgWr	Out	Asserted to 1b to send a Write request.
CfgWrData[7:0]	Out	Write data. Valid when CfgWrite=1b.
CfgRd	Out	Asserted to 1b to send a Read request
CfgRdData[7:0]	In	Read data. Valid when CfgRdValid=1b.
CfgRdValid	In	Assert to 1b when CfgRdData is valid.
CfgWaitRequest	In	Assert to 1b to indicate that the IP core is not ready to respond to a request.

Timing Diagram

Initialization

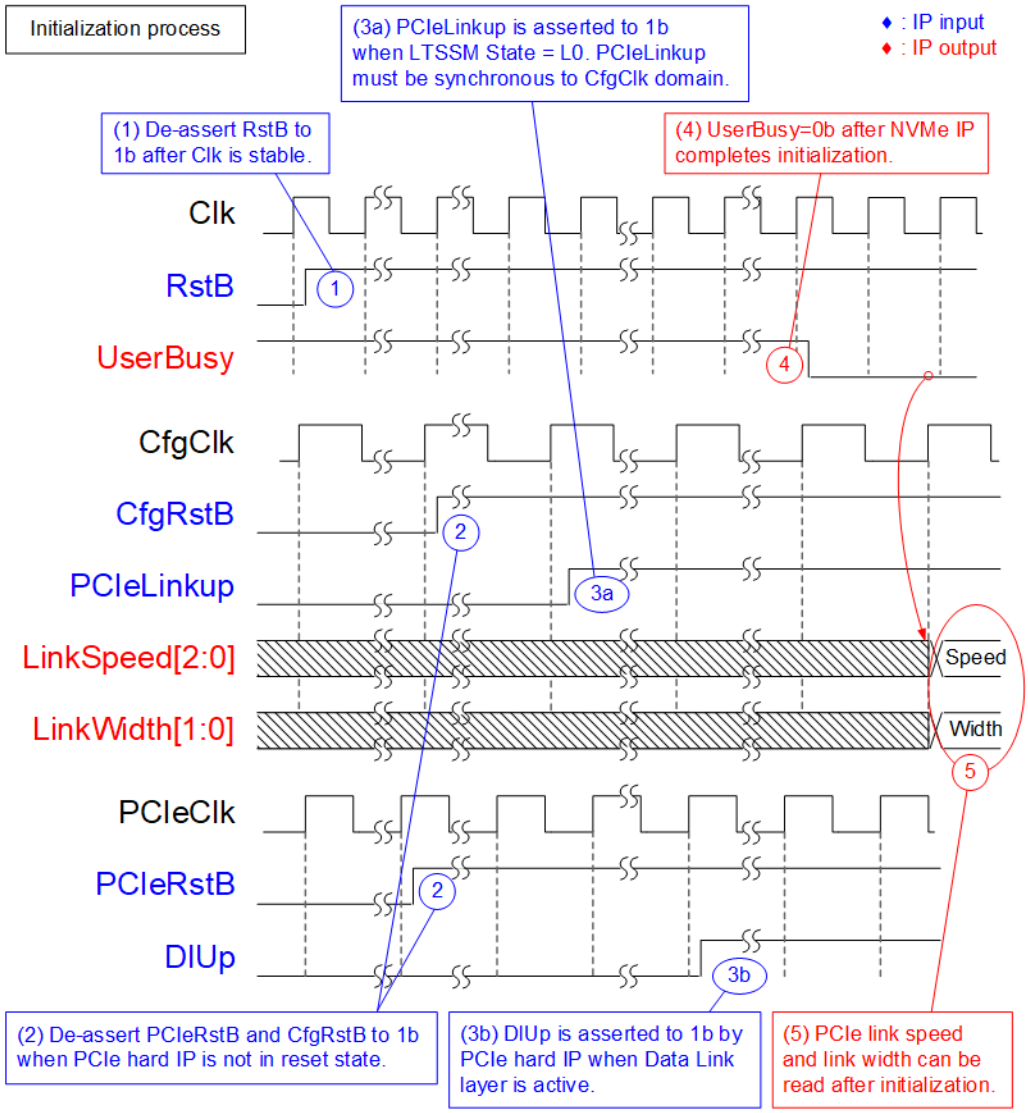

Figure 4: Timing diagram during initialization process

The initialization process of the NVMe IP follows the steps below as shown in timing diagram.

1) Wait until Clk is stable and then set RstB to 1b.
2) After completing the PCIe reset sequence, the PCIe hard IP de-asserts PCIeRstB and CfgRstB to 1b, indicating that it is ready to transfer data with the application layer.
3) The PCIe hard IP initiates the PCIe initialization by following steps.
    a. To set PCIeLinkup to 1b when the LTSSM state is L0 state, an asynchronous register must be used because PCIeLinkup, which requires to run on CfgClk, is decoded from the LTSSM signal generated on PCIeClk.
    b. The PCIe hard IP then asserts DlUp to 1b when the Data Link layer becomes active.

    After this, the NVMe IP initiates its own initialization process.
4) Upon completion of the NVMe IP initialization process, the NVMe IP de-asserts UserBusy to 0b.
5) Once the initialization is complete, the user can read the PCIe link speed and link width from the NVMe IP. However, these registers are synchronous with CfgClk and must be converted to the user logic clock domain using asynchronous registers.

After completing all of the above steps, the NVMe IP is ready to receive commands from the user.

Control interface of dgIF typeS

The dgIF typeS signals can be split into two groups: the Control interface for sending commands and monitoring status, and the Data interface for transferring data streams in both directions.

Figure 5 shows an example of how to send a new command to the IP via the Control interface of dgIF typeS.

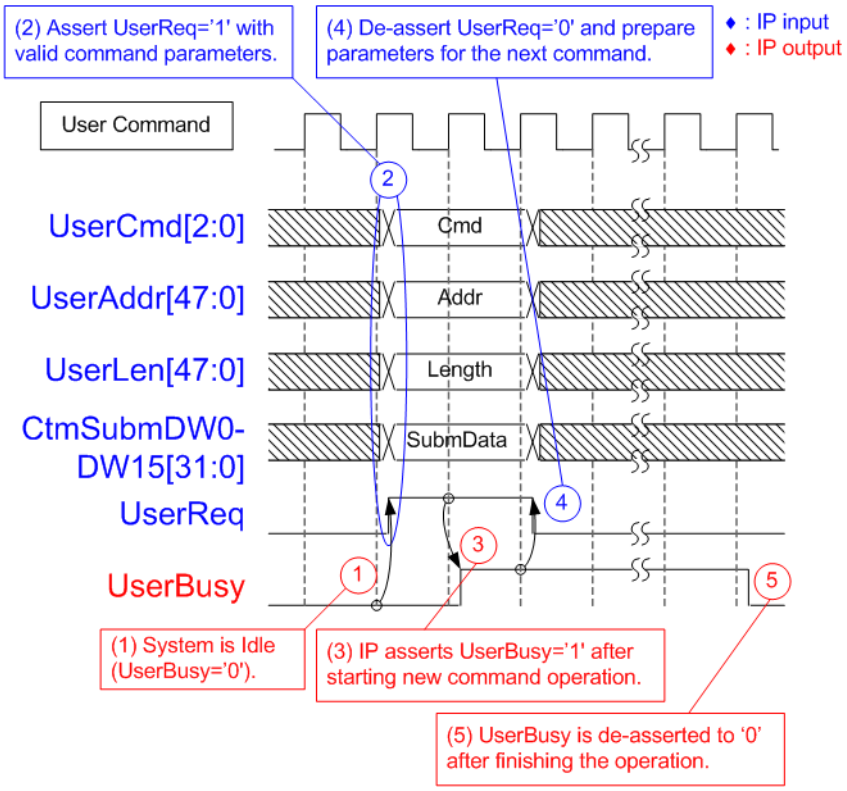

Figure 5: Control interface of dgIF typeS timing diagram

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

Note: The number of parameters used in each command is different, and this is described in more details below.

- *Write and Read command: UserCmd, UserAddr, and UserLen*
- *SMART, Secure Erase, and Flush command: UserCmd and CtmSubmDW0-DW15*
- *Identify and Shutdown command: UserCmd*

Data interface of dgIF typeS

The Data interface of dgIF typeS is used for transferring data stream when operating Write or Read command, and it is compatible with a general FIFO interface. Figure 6 shows the data interface of dgIF typeS when transferring Write data to the IP in the Write command.

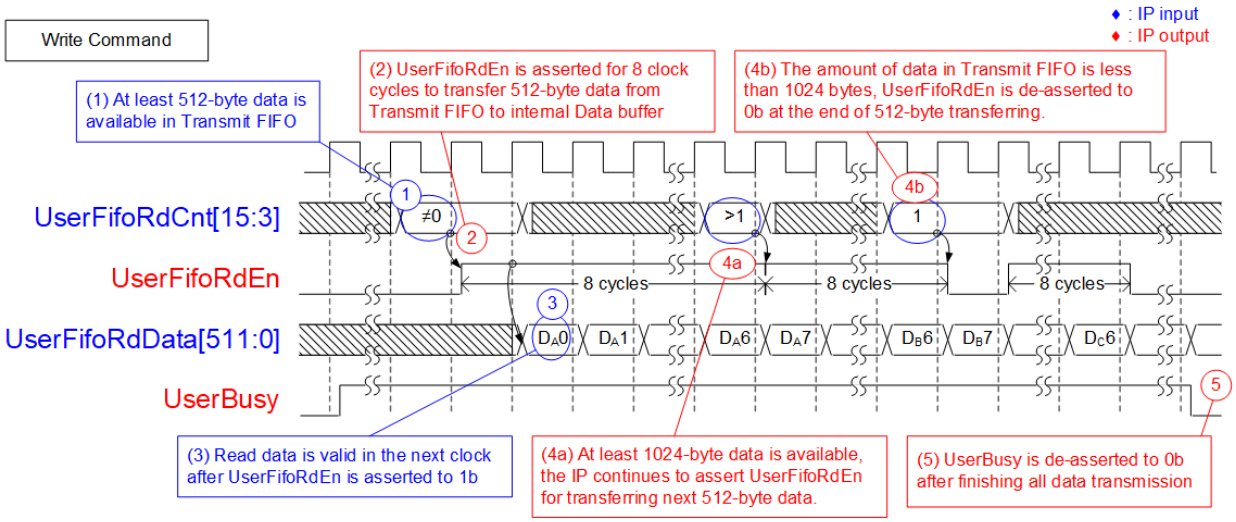

Figure 6: Transmit FIFO interface for Write command

The 16-bit FIFO read data counter (UserFifoRdCnt) shows the total amount of data stored, and if the amount of data is sufficient, 512-byte data (8x512-bit) is transferred.

In the Write command, data is read from the Transmit FIFO until the total data is transferred completely, and the process to transfer data is described as follows.

1) Before starting a new burst transfer, the IP waits until at least 512-byte data is available in the Transmit FIFO by monitoring UserFifoRdCnt[15:3] that must be not equal to 0.
2) The IP asserts UserFifoRdEn to 1b for 8 clock cycles to read 512-byte data from the Transmit FIFO.
3) UserFifoRdData is valid in the next clock cycle after asserting UserFifoRdEn to 1b, and 8 data are continuously transferred.
4) Each 512-byte data is transferred until the total data size is equal to the transfer length set by the user. When there is more data to be transferred, the IP decides to continue the next 512-byte transferring after reading the last data (D_x7; where x is the index of 512-byte data set) of the current transferring. The transfer continuity is considered from UserFifoRdCnt[15:3].
    a. If UserFifoRdCnt[15:3] shows that at least 1024-byte data is available in the Transmit FIFO, the IP continues to assert UserFifoRdEn to 1b for the next 8 clock cycles (repeat step [2] – [4]).
    b. If UserFifoRdCnt[15:3] shows that data in the Transmit FIFO is less than 1024 bytes, the IP de-asserts UserFifoRdEn to 0b to end the current 512-byte transferring. After that, step [1] – [4] are repeated.
5) After the total data is completely transferred, UserBusy is de-asserted to 0b.

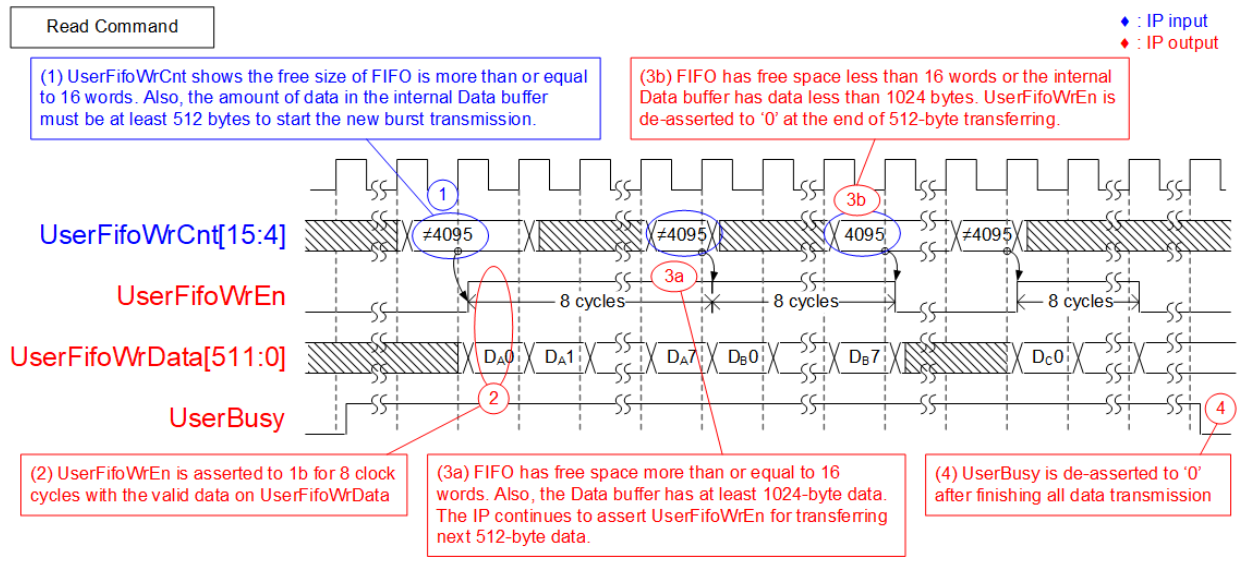

Figure 7: Receive FIFO interface for Read command

When executing the Read command, the data is transferred from the SSD to the Receive FIFO until the entire data is transferred. The steps for transferring a burst of data are below.

1) Before starting a new burst transmission, the UserFifoWrCnt[15:4] is checked to ensure that there is enough free space in the Receive FIFO, which is indicated by the condition UserFifoWrCnt[15:4] ≠ all 1 or 4095. The IP waits until there is enough free space available, and the received data from the SSD is at least 512 bytes. Once these conditions are met, the new burst transmission begins.
2) The IP asserts UserFifoWrEn to 1b for 8 clock cycles to transfer 512-byte data from the Data buffer to the user logic.
3) Each 512-byte data is transferred until the total data size is equal to the transfer length set by the user. If there is more data to be transferred, the IP decides to continue the next 512-byte transferring after writing the last data (D_x7; where x is the index of 512-byte data set) of the current transfer. The continuity of the transfer is determined based on the free space available in the Receive FIFO, which is monitored using UserFifoWrCnt[15:4].
    a. If there is enough free space (more than or equal to 16 words or 1024 bytes) in the Receive FIFO and the received data from the SSD stored in the Data buffer is also at least 1024 bytes, the IP continue to assert UserFifoWrEn to 1b for the next 8 clock cycles by repeating steps [2] – [3].
    b. If there is not enough free space in the Receive FIFO (less than 1024 bytes) or the received data from the SSD is less than 1024 bytes, the IP de-asserts UserFifoWrEn to 0b to end the current 512-byte transfer. After that, steps [1] – [3] are repeated.
4) After the total data is completely transferred, UserBusy is de-asserted to 0b.

IdenCtrl/IdenName

To ensure proper operation of the system, it is recommended to send the Identify command to the IP as the first command after the system boots up. This command updates important information about the SSD, such as its total capacity (LBASize) and LBA unit size (LBAMode), which are necessary for Write and Read commands to operate correctly. The following rules apply to the input parameters of these commands.

1) The sum of the address (UserAddr) and transfer length (UserLen), inputs of Write and Read command, must not exceed the total capacity (LBASize) of the SSD.
2) If LBAMode is 1b (LBA unit size is 4 Kbyte), the three lower bit (bit[2:0]) of UserAddr and UserLen must be set to 0b to align with the 4 Kbyte unit.

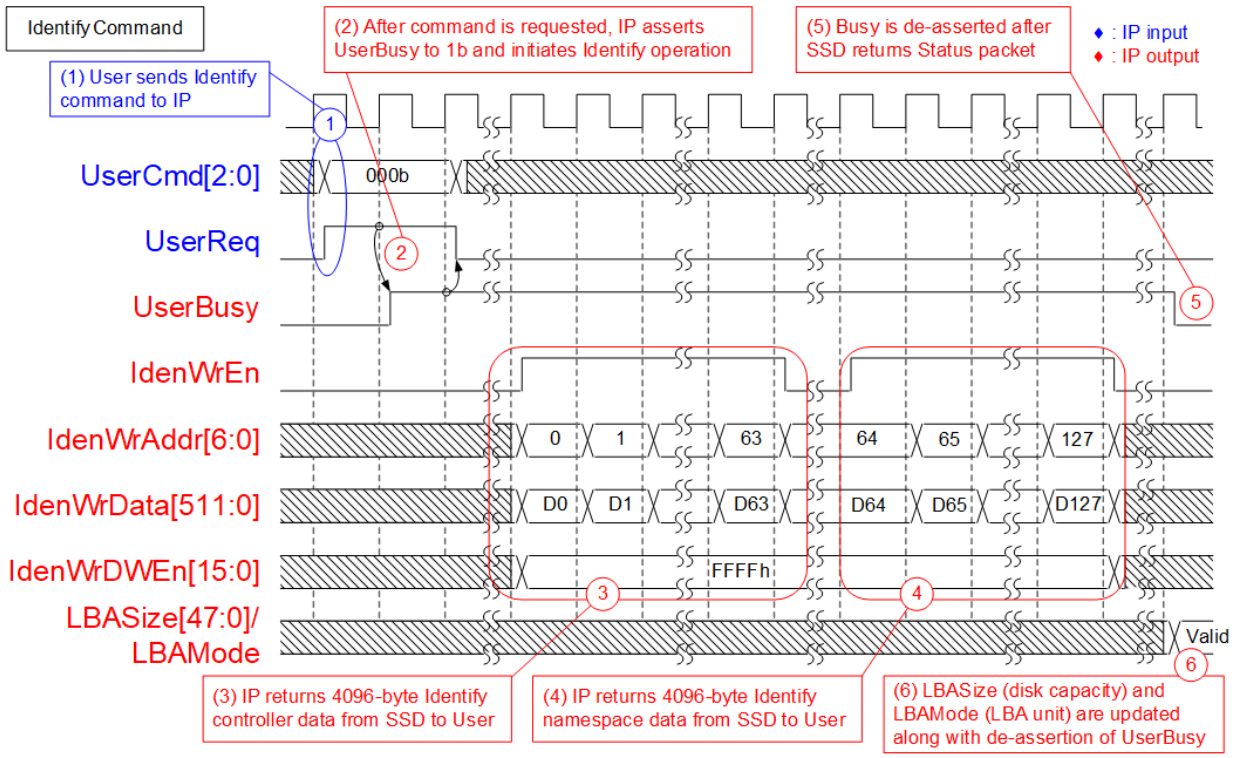

Figure 8: Identify command timing diagram

When executing the Identify command, the following steps are taken.

1) Send the Identify command to the IP (UserCmd=000b and UserReq=1b).
2) The IP asserts UserBusy to 1b after receiving the Identify command.
3) The IP returns 4096-byte Identify controller data to the user with IdenWrAddr equal to 0-63 and asserts IdenWrEn. IdenWrData and IdenWrDWEn are valid at the same clock as IdenWrEn=1b.
4) The IP returns 4096-byte Identify namespace data to the user with IdenWrAddr equal to 64-127. IdenWrAddr[6] can be used to determine the data type as Identify controller data or Identify namespace data.
5) UserBusy is de-asserted to 0b after finishing the Identify command.
6) The LBASize and LBAMode of the SSD are simultaneously updated with the values obtained from the Identify command.

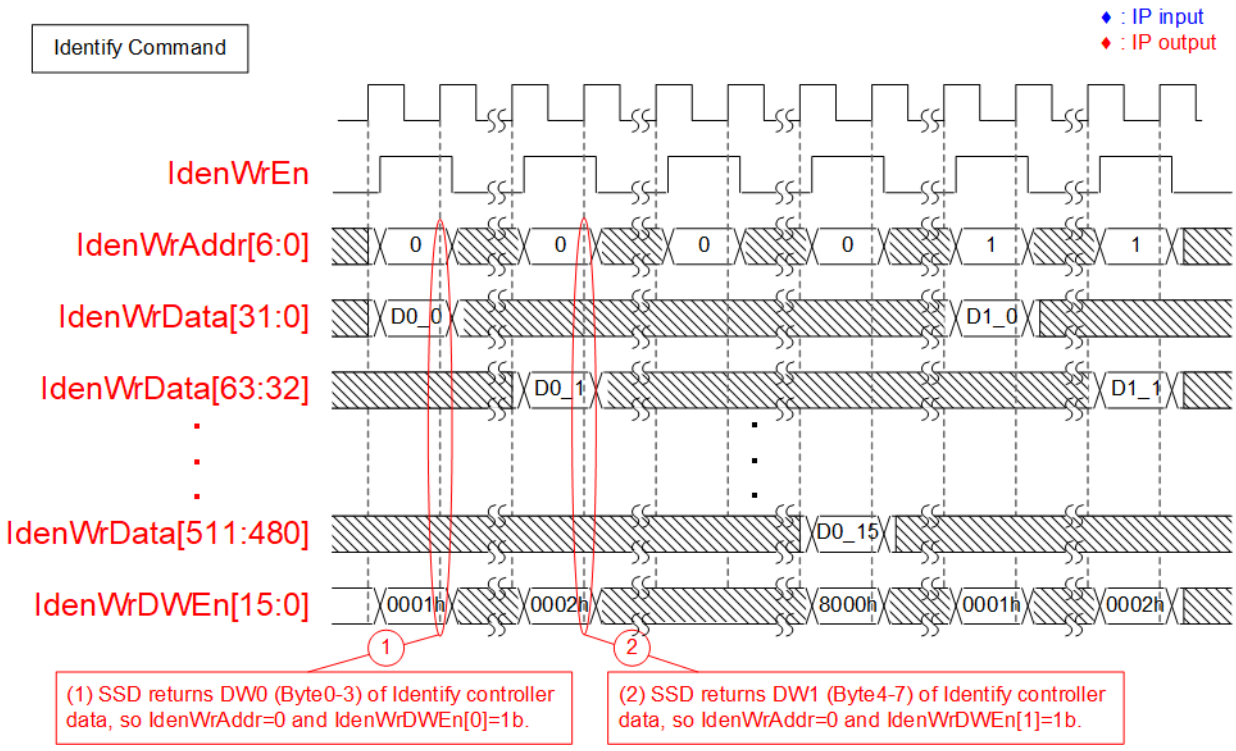

Figure 9: IdenWrDWEn timing diagram

The signal IdenWrDWEn is a 16-bit signal used to validate a 32-bit data signal. Some SSDs return the 4-Kbyte Identify controller data and Identify namespace data one word (32-bit) at a time instead of continuously. To forward 32-bit data, one bit of IdenWrDWEn is asserted to 1b in the write cycle, as illustrated in Figure 9. Each bit of IdenWrDWEn (IdenWrDWEn[0], [1], …, [15]) corresponds to each 32-bit data of IdenWrData (IdenWrData[31:0], [63:32], …, [511:480]).

Shutdown

The Shutdown command is a command that should be sent as the last command before the system is powered down. The SSD ensures that the data from its internal cache is written to the flash memory before the shutdown process finishes. After the shutdown operation is complete, the NVMe IP and the SSD become inactive status. If the SSD is powered down without executing the Shutdown command, the total count of unsafe shutdowns is increased, as returned data from the SMART command.

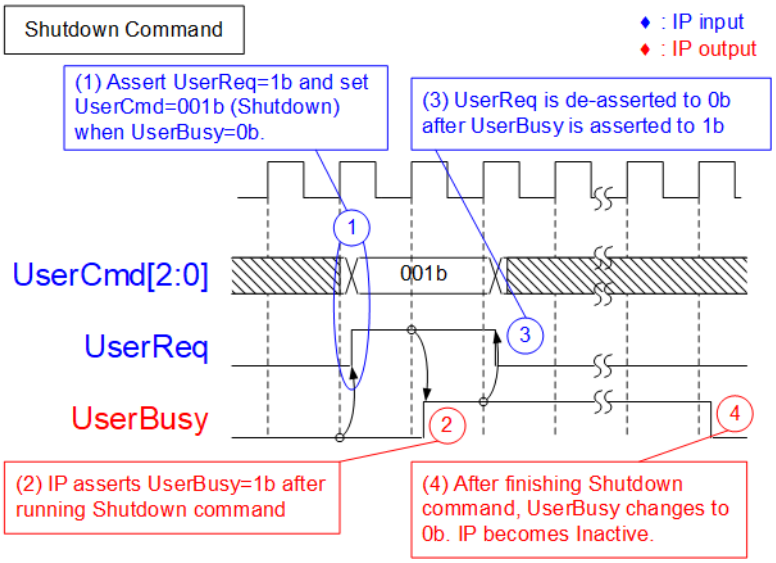

Figure 10: Shutdown command timing diagram

The process for executing the Shutdown command is described below.

1) The IP must be in Idle state (UserBusy=0b) before sending the Shutdown command. The user must set UserReq=1b and UserCmd=001b to send the Shutdown command request.
2) After the NVMe IP runs the Shutdown command, UserBusy is asserted to 1b.
3) To clear the current request, UserReq is de-asserted to 0b after UserBusy is asserted to 1b.
4) UserBusy is de-asserted to 0b when the SSD is completely shut down. After the shutdown process is completed, the IP will not receive any command requested from the user.

SMART

The SMART command is the command to check the health of the SSD. When this command is sent, the SSD returns 512-byte health information. The SMART command parameters are loaded from the CtmSubmDW0-DW15 signals on the Custom command interface. The user must set the 16-dword data which is a constant value before asserting UserReq. Once the SMART data is returned, it can be accessed via the CtmRAM port, as shown in Figure 11.

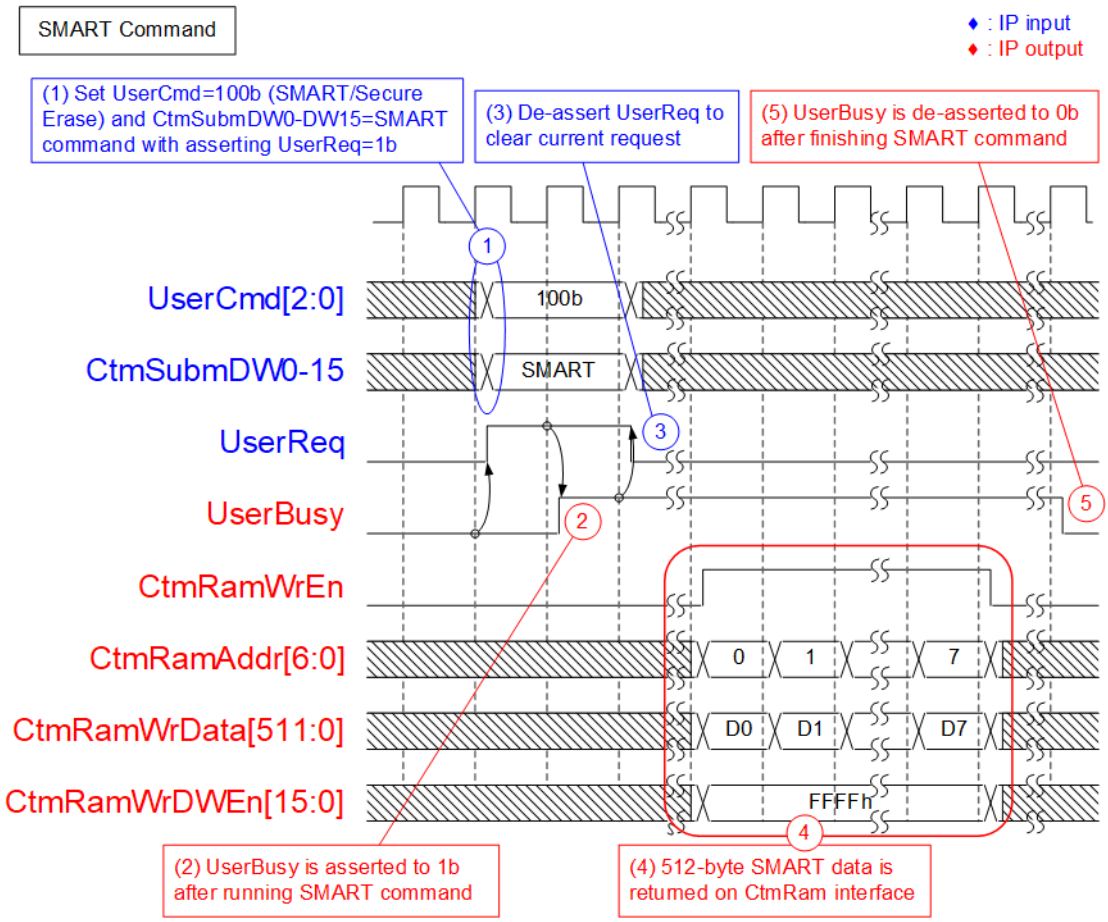

Figure 11: SMART command timing diagram

Below are the details of how to run the SMART command.

1) The NVMe IP must be Idle (UserBusy=0b) before sending the command request. All input parameters must be stable when UserReq is asserted to 1b for sending the request. The CtmSubmDW0-DW15 is set as a constant value for the SMART command by following values.
   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) UserBusy is asserted to 1b after the NVMe IP executes the SMART command.
3) UserReq is de-asserted to 0b 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 1b. CtmRamAddr is equal to 0-7 to be data index of 512-byte data. When CtmRamAddr=0, byte0-63 of SMART data is valid on CtmRamWrData. CtmRamWrDWEn is Dword enable for each 32-bit CtmRamWrData. If CtmRamWrDWEn=FFFFh, all 512 bits of CtmRamWrData are valid.
5) UserBusy is de-asserted to 0b when finishing the SMART command.

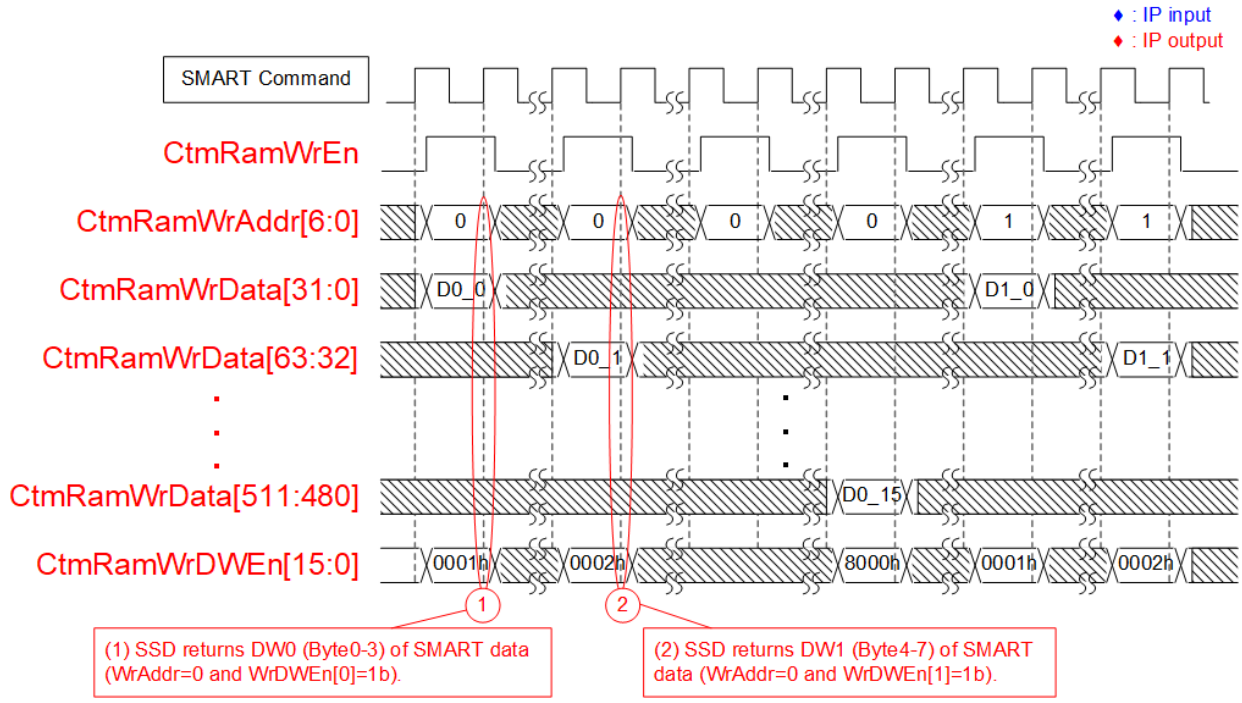

Figure 12: CtmRamWrDWEn timing diagram

Similar to Identify command, some SSDs returns only one Dword (32-bit) of data at a time instead of 512-byte data continuously. In such cases, one bit of CtmRamWrDWEn is asserted to 1b in the write cycle to be the valid signal of 32-bit CtmRamWrData. Each bit of CtmRamWrDWEn (bit[0], [1], …, [15]) corresponds to each 32 bits of CtmRamWrData (bit[31:0], [63:32], …, [511:480]).

Secure Erase

Secure Erase is a command that erases all user data in the SSD. After the Secure Erase command is executed, the contents of the user data are indeterminate.

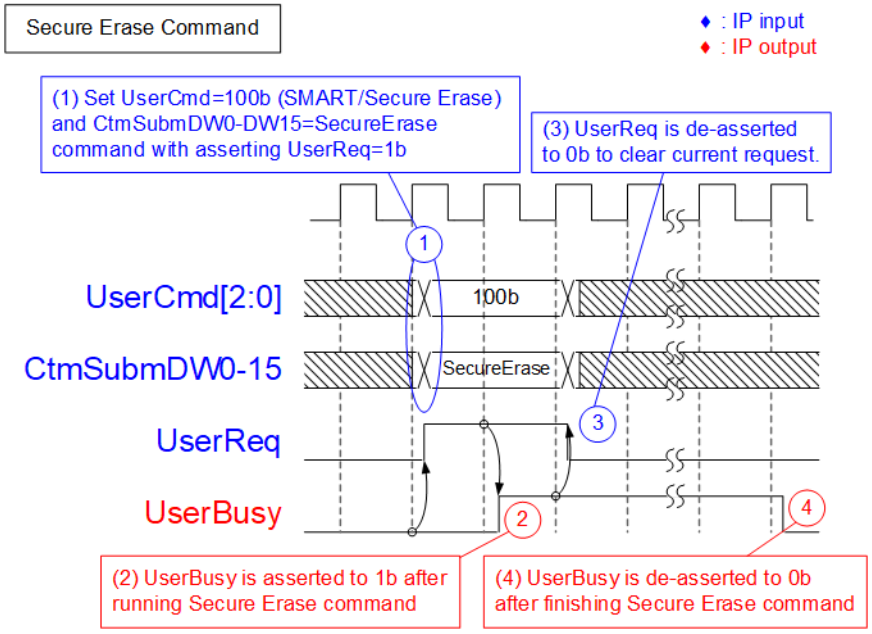

Figure 13: Secure Erase command timing diagram

Below are the details of how to run the Secure Erase command.

1) The IP must be in an Idle state (UserBusy=0b) before sending the command request. All input parameters must be stable when UserReq is asserted to 1b to send the request. The CtmSubmDW0-DW15 is set as a constant value for the Secure Erase command by following values.
   CtmSubmDW0 = 0x0000_0080
   CtmSubmDW1 = 0x0000_0001
   CtmSubmDW2 – CtmSubmDW9 = 0x0000_0000
   CtmSubmDW10 = 0x0000_0200
   CtmSubmDW11 – CtmSubmDW15 = 0x0000_0000
2) After the NVMe IP executes the Secure Erase command, UserBusy is asserted to 1b.
3) UserReq is then de-asserted to 0b to clear the current request, the user logic can change the input parameters for the next command request.
4) UserBusy is de-asserted to 0b when the Secure Erase command is completed.

Note: Some SSDs may experience a decrease in performance after long data transfer. For such SSDs the Secure Erase command can help restore their performance.

Flush

The SSDs typically enhance write performance by caching write data before writing it to the flash memory. However, unexpected power loss can result data loss as cached data may not be stored in flash memory. To avoid data loss, the Flush command cam be used to force the SSD controller to write cached data to the flash memory.

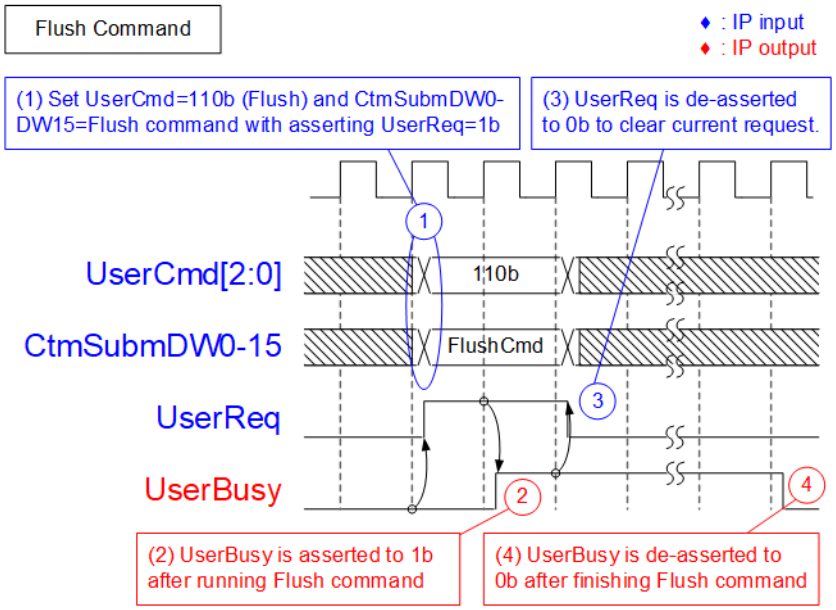

Figure 14: Flush command timing diagram

To execute the Flush command, the following details should be mentioned.

1) The IP must be Idle (UserBusy=0b) before sending the command request, and all input parameters must be stable when UserReq is asserted to 1b for sending the request. The CtmSubmDW0-DW15 is set as a constant value with the following values for Flush command.
   CtmSubmDW0 = 0x0000_0000
   CtmSubmDW1 = 0x0000_0001
   CtmSubmDW2 – CtmSubmDW15 = 0x0000_0000
2) UserBusy is asserted to 1b after the NVMe IP executes the Flush command.
3) UserReq is de-asserted to 0b to clear the current request, and the user logic can change the input parameters for the next command request.
4) UserBusy is de-asserted to 0b when the Flush command is completed.

Using the Flush command ensures that all data from the previous Write command is guaranteed to be stored in flash memory, thus preventing data loss in the event of unexpected power loss.

Error

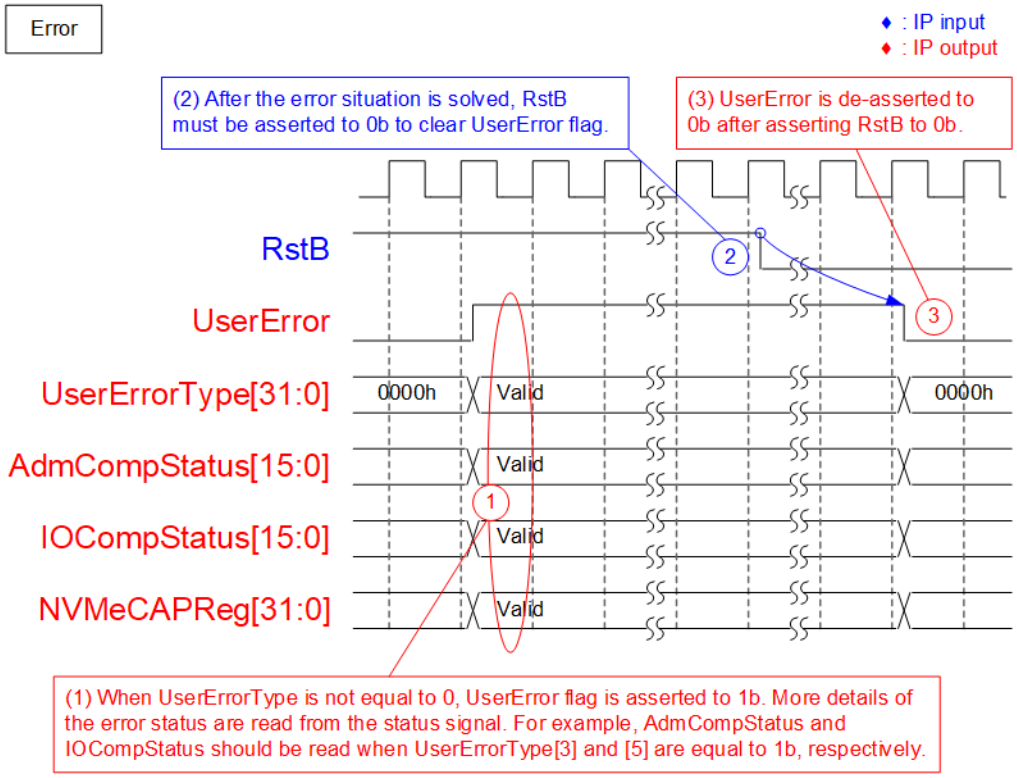

Figure 15: Error flag timing diagram

If an error occurs during the initialization process or when running some commands, the UserError flag is set to 1b. To check the type of error, the UserErrorType should be read. The NVMeCAPReg, AdmCompStatus, and IOCompStatus signals can be used to monitor the error details after UserError is set to 1b.

If an error occurs during the initialization process, it is recommended to read the NVMeCAPReg signal to check the capabilities of the NVMe SSD. If an error occurs while operating a command, it is recommended to read the AdmCompStatus and IOCompStatus signals.

- If bit[3] of UserErrorType is asserted, the AdmCompStatus signal should be checked for more details.
- If bit[5] of UserErrorType is asserted, the IOCompStatus signal should be checked for more details.

The UserError flag is cleared only by the RstB signal. After the failure is resolved, RstB is asserted to 0b to clear the error flag.

Verification Methods

The NVMe IP Core for Gen5 functionality was verified by simulation and also proved on real board design by using Intel Agilex 7 FPGA I-Series development kit.

Recommended Design Experience

Experience design engineers with a knowledge of Quartus 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	2-May-23	New release