**Design Gateway Co.,Ltd**

54 BB Building 14th Fl., Room No.1402 Sukhumvit 21 Rd. (Asoke), Klongtoey-Nua, Wattana, Bangkok 10110
Phone: 66(0)2-261-2277
Fax: 66(0)2-261-2290
E-mail: ip-sales@design-gateway.com
URL: www.design-gateway.com

## Core Facts

| Provided with Core | |
|---|---|
| Documentation | Reference Design Manual<br>Demo Instruction Manual |
| Design File Formats | Encrypted hdl File |
| Instantiation Templates | VHDL |
| Reference Designs & Application Notes | QuartusII Project,<br>See Reference Design Manual |
| Additional Items | Demo on<br>Arria10 GX development kit |
| **Support** | |
| Support Provided by Design Gateway Co., Ltd. | |

## Features

- Implement application layer, transaction layer, data link layer, and some parts of physical layer to access NVMe SSD without CPU usage
- Operating with Transceiver Native PHY Intel FPGA IP, configured to 4-lane PCIe Gen3 (128-bit bus interface)
- Include 256 Kbyte RAM to be data buffer
- Simple user interface by dgIF typeS
- Supports six commands, i.e. Identify, Shutdown, Write, Read, SMART, and Flush (Supporting the additonal commands can be customized)
- 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 byte or 4096 byte
- User clock frequency must be more than or equal to PHY clock (250MHz for Gen3)
- Available reference design
  - Arria10 GX development board with AB18-PCIeX16/AB16-PCIeXOVR adapter board

**Table 1: Example Implementation Statistics**

| Family | Example Device | Fmax (MHz) | Logic utilization (ALMs) | Registers | Pin | Block Memory | Design Tools |
|---|---|---|---|---|---|---|---|
| Arria10 GX | 10AX115S2F45I1SG | 300 | 8560 | 10984 | - | 140 M20Ks | QuartusII 18.0 |

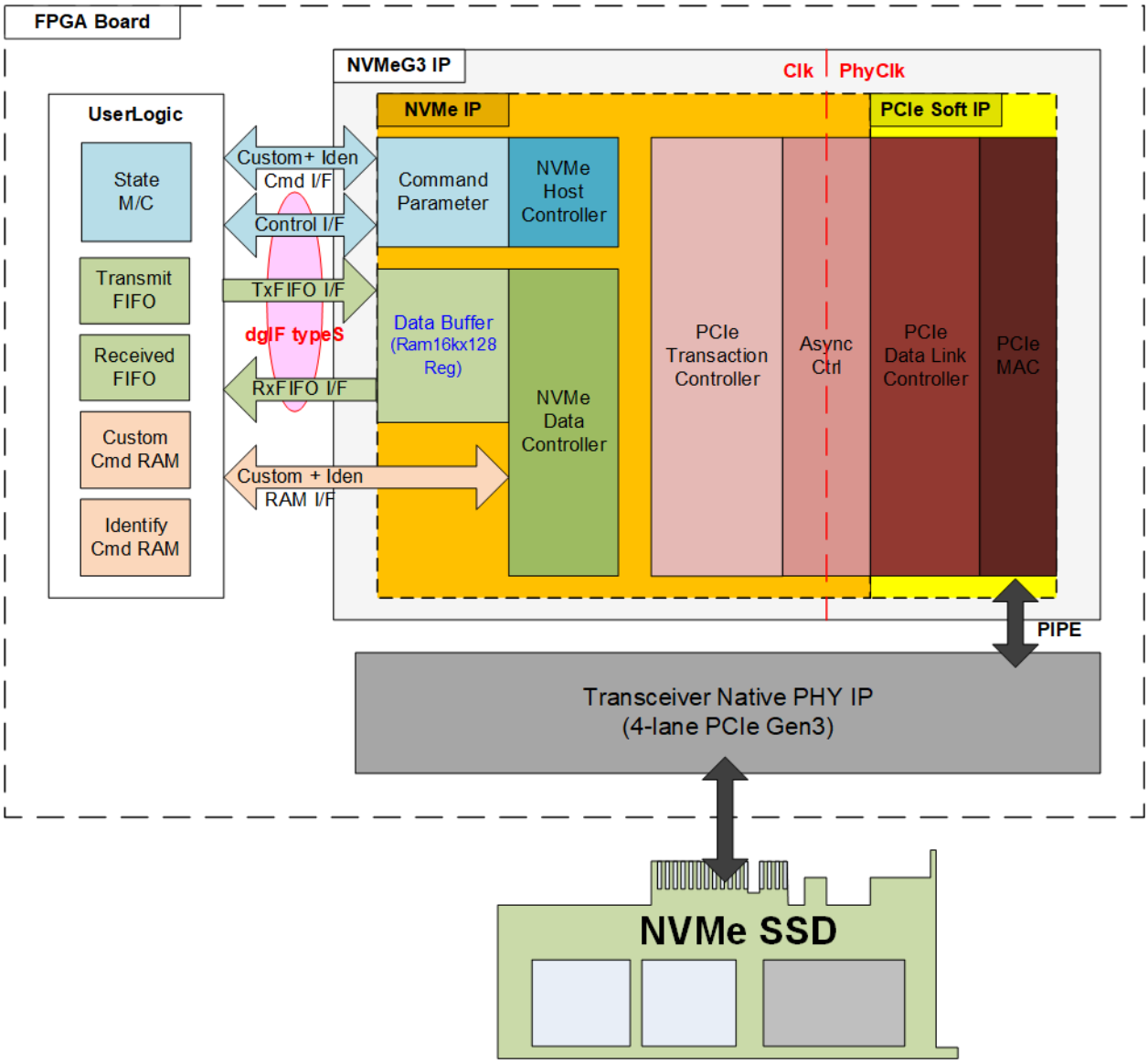


**Figure 1: NVMeG3 IP Block Diagram**

## Applications

NVMe IP Core with PCIe Gen3 Soft IP (NVMeG3 IP) is ideal to access NVMe SSD without PCIe Hard IP, CPU, and external memory. NVMeG3 IP includes PCIe Gen3 Soft IP and 256 Kbyte memory. This solution is strongly recommended for the application which requires very large storage and ultra high speed performance or the application that requires RAID0 system which having a number of SSD more than a number of PCIe hard IP in the FPGA.

When the number of SSD using in the customer application is not more than the number of PCIe hard IP in the FPGA, it is recommended to use DG NVMe IP Core for smaller FPGA resource utilization. Also, small size system is achieved by using M.2 SSD package.

## General Description

Design Gateway develops NVMeG3 IP to run as NVMe host controller for accessing NVMe SSD, like standard DG NVMe IP. The user interface and features in NVMeG3 IP is identical to NVMe IP. The additional feature of NVMeG3 IP is the built-in PCIe soft IP implementing Data link layer and some parts of Physical layer of PCIe protocol by pure logic.

Similar to NVMe IP, NVMeG3 IP supports six NVMe commands, i.e. Identify, Shutdown, Write, Read, SMART, and Flush command. 256 Kbyte Embedded RAM is integrated in the NVMeG3 IP to be data buffer. The system does not need CPU and external memory. Clock frequency of the user logic must be more than or equal to PHY clock (250 MHz). More details of the standard NVMe IP are described in NVMe IP datasheet which can be downloaded from our website.

https://dgway.com/products/IP/NVMe-IP/dg_nvme_ip_data_sheet_en.pdf

**Table 2: DG NVMe-IP comparison**

| Feature | NVMe IP | NVMeG3 IP |
|---|---|---|
| PCIe Interface | 128-bit Avalon ST | 128-bit PIPE |
| Intel IP required | Avalon-ST PCIe Hard IP | **Transceiver PHY IP** |
| PCIe Hard IP | Necessary | **Not use** |
| PCIe Speed | **1-4 Lane with Gen3 or lower speed** | Support only 4-lane PCIe Gen3 |
| User Interface | dgIF typeS | dgIF typeS |
| FPGA resource | **Smaller** | Larger |
| Maximum SSD | Depend on the number of PCIe Hard IPs | **Depend on the number of transceivers** |
| SSD Performance | Up to 3300 MB/s* | Up to 3300 MB/s* |

*Note: The performance is achieved by testing with 500 GB Samsung 970 PRO SSD

As shown in Table 2, the main advantage of NVMeG3 IP is built-in PCIe soft IP feature. Therefore, the maximum number of SSD is not limited by the number of PCIe hard IP but limited by the number of transceivers and the resource utilization. However, the disadvantage of the NVMeG3 IP is the resource utilzation which is larger than NVMe IP for implementing additional PCIe soft IP. Also, NVMeG3 IP supports only 4-lane PCIe Gen3 SSD.

Therefore, if the customer system requires vast stroage or rapid speed which RAID0 needs to implemented, NVMeG3 IP is recommended. However, if the number is PCIe hard IP is enough for the system, we recommend the standard NVMe IP which uses less resource.

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

## Functional Description

Figure 2 shows operation flow of NVMeG3 IP after IP reset is released.

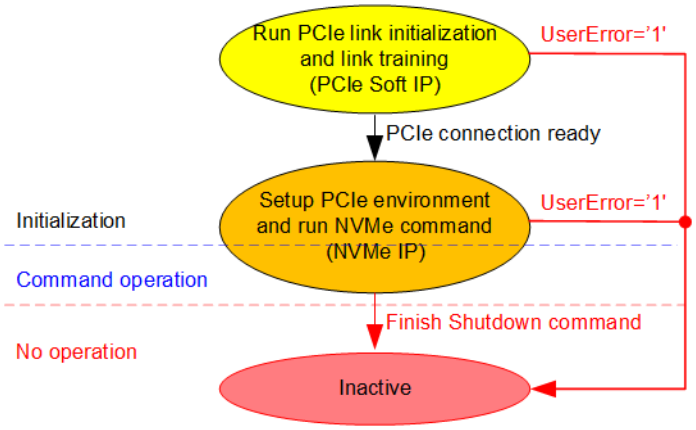


**Figure 2: NVMeG3 IP Operation Flow**

As shown in Figure 2, NVMeG3 IP operations are mostly similar to NVMe IP. The additional part is only the first step which runs PCIe link initialization and training. This process is controlled by the Physical layer to configure and initialize link and port. The process consists of several steps as follows.

1) Detects device connection by monitoring electrical idle signal.
2) Polling until Bit/Symbol of every lane is locked.
3) Sets the number of lanes to 4 lanes.
4) Sets PCIe speed to Gen3.
5) Adjusts the equalizer parameters.
6) Sets flow control parameters.

After finishing this step, the signal quality is in the good status and ready to transfer PCIe packet.

The next step is the same process as NVMe IP. First, PCIe registers and NVMe registers are configured to finish NVMe IP initialization process. After that, the IP is ready to receive the command from user. There are six operations for six commands. When finishing Shutdown command or detecting some errors in initialization process, the IP goes to Inactive status which must be recovered by reset process. More details of NVMe IP operation flow are described in NVMe IP datasheet.

As shown in Figure 1, NVMeG3 IP consists of two modules, i.e. NVMe IP and PCIe Soft IP.

The user interface of NVMe IP has three groups. First one is dgIF typeS which is the standard storage interface of DG IP core. Data interface is 128-bit FIFO interface while control interface is simple interface by assigning address, length, and command number. Second is the control interface for custom command and Identify command. Third is the data interface of custom command and Identify command which is 128-bit RAM interface.

The user parameters are stored in Command parameter and forwarded to NVMe Host controller which is the main controller to create the command packet and decode the status packet. Data buffer is implemented by 256 KB Embedded RAM to store the data from user logic. NVMe data controller is designed to create and decode data packet following NVMe protocol.

Furthermore, the NVMe IP implements PCIe transaction controller to create and decode PCIe TLP packet. AsyncCtrl is clock-crossing module to transfer packet between Clk domain (user clock) and PhyClk domain which is output from Transceiver PHY IP and fixed at 250 MHz for PCIe Gen3 speed.

The details of PCIe Soft IP which is implemented in NVMeG3 IP are described as follows.

- **PCIe Data Link Controller**

  PCIe Data Link Controller implements Data Link Layer of PCIe protocol. The function of the Data Link Layer is to ensure reliable delivery of TLPs which is the packet format transferring between PCIe Transation Controller and PCIe Data Link Controller. LCRC (Link Cyclic Redundancy Code) is added to each TLP for error checking at the receiver. Besides, the Sequence Number is appended to check the packet order. As a result, the receiver can sort the packet to be the same order as the sender . Also, the receiver can detect the missing TLPs.

  After the receiver verifies LCRC and the Sequence Number, Ack DLLPs (Data Link Layer Packets) are generated to confirm good reception of TLPs. Nak DLLPs are created to indicate a transmission error. When Nak is received, the transmitter re-sends the TLPs to solve the problem.

  To re-send the TLPs, Replay buffer is included within Data Link Layer. This module is implemented by 2 Kbyte RAM. Moreover, another 2 Kbyte RAM is included in the receiver to be the data buffer between Data Link Controller and Transaction Controller.

- **PCIe Media Access Controller (PCIe MAC)**

  PCIe MAC is designed to interface with Transceiver Native PHY IP (Transceiver PHY IP) by PIPE. There are two purposes of this module. First is to run Link initialization and training process. Second is to control data packet following PCIe physical specification.

  For Link initialization and training, some processes are implemented within Transceiver PHY IP such as CDR for Bit lock and Block lock for Gen3 speed. LTSSM (Link Training and Status State Machine), implemented in PCIe MAC, is responsible to control Link width, Lane reversal, Polarity inversion, and Link data rate (Gen3 speed). As Gen3 operates at 8.0 GT/s which is more sensitive than Gen1 and Gen2, the additional features must be implemented in PCIe Gen3 MAC, i.e. DC balance and equalization.

  After finishing the initialization and training, data packet is transferred. To transmit the packet, PCIe MAC includes multiplexer for selecting the data types, byte striping for arranging data format in each lane, and data scrambler for reducing the noise. On the other hand, the receiver consists of the logic to run data de-scrambler, byte un-striping, and data filtering.

## User Logic

User logic for running NVMeG3 IP is similar to user logic for running NVMe IP. To transfer data, the interfaces are dgIF typeS (Write and Read command) and two RAMs interface (Identify and custom commands such as SMART command).

## Transceiver Native PHY IP

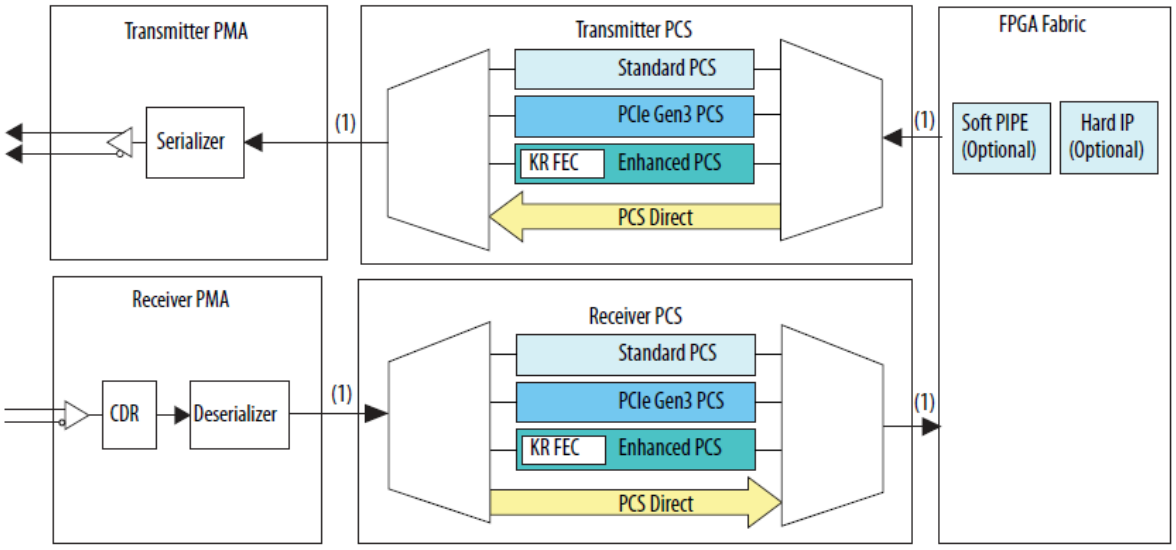


**Figure 3: Architecture of Transceiver Native PHY IP in Intel FPGA**

Intel FPGA provides Transceiver Native PHY IP which is a part of physical layer by implementing PCS and PMA by using PHY Interface for PCI Express (PIPE). To operate with NVMeG3 IP, Transceiver PHY IP must be configured as Gen3 PIPE interface with Lane width to x4 and Link speed to 8.0 GT/s. More details of PHY IP are described in Transceiver user guide.

For example, “Intel® Arria® 10 Transceiver PHY User Guide” can be downloaded from following link.

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/ug_arria10_xcvr_phy.pdf

## Core I/O Signals

Descriptions of all signal I/Os are provided in Table 3 and Table 4.

**Table 3: 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. Release to '1' when Clk signal is stable. |
| Clk | In | System clock for running NVMeG3 IP.<br>*Note: The Clk frequency must be more than or equal to PhyClk (250 MHz for PCIe Gen3).* |
| UserCmd[2:0] | In | User Command ("000": Identify, "001": Shutdown, "010": Write SSD, "011": Read SSD, "100": SMART, "110": Flush, "101"/"111": Reserved) |
| UserAddr[47:0] | In | Start address to write/read SSD in 512-byte unit.<br>In case LBA unit = 4 Kbyte, UserAddr[2:0] must be always set to "000" to align 4 Kbyte unit.<br>In case LBA unit = 512 byte, it is recommended to set UserAddr[2:0]="000" to align 4 Kbyte size (SSD page size). Write/Read performance of most SSDs is reduced when start address is not aligned to page size. |
| UserLen[47:0] | In | Total transfer size in the request in 512-byte unit. Valid from 1 to (LBASize-UserAddr).<br>In case LBA unit = 4 Kbyte, UserLen[2:0] must be always set to "000" to align 4 Kbyte unit. |
| UserReq | In | Assert to '1' to send the new command request. This signal can be asserted when the IP is Idle (UserBusy='0'). Command parameters (UserCmd, UserAddr, UserLen, CtmSubmDW0-DW15) must be valid and stable during UserReq='1'.<br>UserAddr and UserLen are inputs for Write/Read command while CtmSubmDW0-DW15 are inputs for SMART/Flush command. |
| UserBusy | Out | Assert to '1' when IP is busy. New request must not be sent (UserReq to '1') when IP is still busy. |
| LBASize[47:0] | Out | Total capacity of SSD in 512-byte unit. Default value is 0.<br>This value is valid after user sends Identify command. |
| LBAMode | Out | LBA unit size of SSD ('0': 512byte, '1': 4 Kbyte). Default value is 0.<br>This value is valid after user sends Identify command. |
| UserError | Out | Error flag. Assert to '1' when UserErrorType is not equal to 0.<br>The flag is de-asserted to '0' by asserting RstB to '0'. |
| UserErrorType[31:0] | Out | Error status.<br>[0] – Error when PCIe class code is not correct.<br>[1] – Error from CAP (Controller capabilities) register which may be caused from<br>- MPSMIN (Memory Page Size Minimum) is not equal to 0.<br>- NVM command set flag (bit 37 of CAP register) is not set to 1.<br>- DSTRD (Doorbell Stride) is not 0.<br>- MQES (Maximum Queue Entries Supported) is more than or equal to 7.<br>More details of each register can be checked from NVMeCAPReg signal.<br>[2] – Error when Admin completion entry is not received until timeout.<br>[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.<br>[4] – Error when IO completion entry is not received until timeout.<br>[5] – Error when status register in IO completion entry is not 0 or phase tag is invalid. Please see more details from IOCompStatus signal.<br>[6] – Error when Completion TLP packet size is not correct.<br>[7] – Reserved |

| Signal | Dir | Description |
|---|---|---|
| **Control I/F of dgIF typeS** | | |
| UserErrorType[31:0] | Out | [8] – Error from Unsupported Request (UR) flag in Completion TLP packet.<br>[9] – Error from Completer Abort (CA) flag in Completion TLP packet.<br>[10] – Error when Rx Buffer in Data Link controller is overflow.<br>[15:11] – Reserved<br>[16]- Error from unsupport LBA unit (LBA unit is not equal to 512 byte or 4 Kbyte)<br>[31:17] – Reserved<br>*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 Received FIFO. Used to check full status.<br>If total FIFO size is less than 16-bit, please fill '1' to upper bit. |
| UserFifoWrEn | Out | Assert to '1' to write data to Received FIFO during running Read command. |
| UserFifoWrData[127:0] | Out | Write data bus of Received FIFO. Valid when UserFifoWrEn='1'. |
| UserFifoRdCnt[15:0] | In | Read data counter of Transmit FIFO. Used to check data size stored in FIFO.<br>If FIFO size is less than 16-bit, please fill '0' to upper bit. |
| UserFifoEmpty | In | The signal is unused for this IP. |
| UserFifoRdEn | Out | Assert 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'. |
| **NVMeG3 IP Interface** | | |
| IPVesion[31:0] | Out | IP version number |
| TestPin[31:0] | Out | Reserved to be IP Test point. |
| TimeOutSet[31:0] | In | Timeout value to wait completion from SSD. Time unit is equal to 1/(Clk frequency).<br>When TimeOutSet is set to 0, Timeout function is disabled. |
| AdmCompStatus[15:0] | Out | Status output from Admission Completion Entry<br>[0] – Set to '1' when Phase tag or Command ID in Admin Completion Entry is invalid.<br>[15:1] – Status field value of Admin Completion Entry |
| IOCompStatus[15:0] | Out | Status output from IO Completion Entry<br>[0] – Set to '1' when Phase tag in IO Completion Entry is invalid.<br>[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'.<br>[15:0] – MQES (Maximum Queue Entries Supported)<br>[19:16] – DSTRD (Doorbell Stride)<br>[20] – NVM command set flag<br>[24:21] – MPSMIN (Memory Page Size Minimum)<br>[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'.<br>'1': this dword data is valid, '0': this dword data is not available.<br>Bit[0]: IdenWrData[31:0], Bit[1]: IdenWrData[63:32], and so on. |
| IdenWrAddr[8:0] | Out | Index of IdenWrData in 128-bit unit. Valid when IdenWrEn='1'.<br>0x000-0x0FF is 4Kbyte Identify controller data.<br>0x100-0x1FF is 4Kbyte Identify namespace data. |
| IdenWrData[127:0] | Out | 4Kbyte Identify controller data or Identify namespace data. Valid when IdenWrEn='1'. |

| Signal | Dir | Description |
|---|---|---|
| **Custom command interface** | | |
| CtmSubmDW0[31:0] – CtmSubmDW15[31:0] | In | 16 Dwords of Submission queue entry for SMART/Flush command.<br>DW0: Command Dword0, DW1: Command Dword1, …, and DW15: Command Dword15.<br>These inputs must be valid and stable during 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.<br>DW0: Completion Dword0, DW1: Completion Dword1, …, and DW3: Completion Dword3 |
| CtmRamWrEn | Out | Assert 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'.<br>'1': this dword data is valid, '0': this dword data is not available.<br>Bit[0]: CtmRamWrData[31:0], Bit[1]: CtmRamWrData[63:32], and so on. |
| CtmRamAddr[8:0] | Out | Index of CtmRamWrData when SMART data is received. Valid when CtmRamWrEn='1'.<br>(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 4: Physical I/O Signals (Synchronous to PhyClk)**

| Signal | Dir | Description |
|---|---|---|
| **PHY Clock and Reset** | | |
| PhyRstB | In | Synchronous reset signal. Active low.<br>Release to '1' when Transceiver PHY IP is not in reset state. |
| PhyClk | In | Clock output from Transceiver PHY IP (250 MHz when operating at PCIe Gen3 speed). |
| **Other PHY Interface** | | |
| MACTestPin[63:0] | Out | Test point of PCIe MAC. |
| MACStatus[7:0] | Out | Status output from PCIe MAC. |
| **PIPE Data Interface** | | |
| PhyTxData[127:0] | Out | Tx parallel data output to Transceiver PHY IP. |
| PhyTxDataK[15:0] | Out | Indicator of control data to indicate whether PCIeTxData is control or data. |
| PhyTxDataValid[3:0] | Out | Asserted to '1' when the valid data is on PhyTxData. |
| PhyTxStartBlock[3:0] | Out | Asserted to '1' at the first clock of 128b block to indicate start of block.<br>Valid when PhyTxDataValid='1'. |
| PhyTxSyncHeader[7:0] | Out | Indicates whether data block is ordered set or data stream.<br>Valid at the first clock of 128b block together with PhyTxStartBlock. |
| PhyRxData[127:0] | In | Rx parallel data input from Transceiver PHY IP. |
| PhyRxDataK[15:0] | In | Indicator of control data to indicate whether PhyRxData is control or data. |
| PhyRxDataValid[3:0] | In | Asserted to '1' when the valid data is on PhyRxData. |
| PhyRxStartBlock[3:0] | In | Asserted to '1' at the first clock of 128b block to indicate start of block.<br>Valid when PhyRxDataValid='1'. |
| PhyRxSyncHeader[7:0] | In | Indicates whether data block is ordered set or data stream.<br>Valid at the first clock of 128b block together with PhyRxStartBlock. |
| **PIPE Control and Status Signal** | | |
| PhyTxDetectRx | Out | Requests PHY to begin a receiver detection operation. |
| PhyTxElecIdle[3:0] | Out | Forces Tx to enter electrical idle state. |
| PhyTxCompliance[3:0] | Out | Asserted to '1' for running negative disparity. |
| PhyRxPolarity[3:0] | Out | Requests PHY to perform polarity inversion on the received data. |
| PhyPowerdown[1:0] | Out | Requests PHY to change the power state. |
| PhyRate[1:0] | Out | Requests PHY to change link rate. |
| PhyRxValid[3:0] | In | Indicates symbol lock and valid data when logic high. |
| PhyPhyStatus[3:0] | In | Used to communicate completion of several PIPE operations. |
| PhyRxElecIdle[3:0] | In | Indicates Rx electrical idle detected. |
| PhyRxStatus[11:0] | In | Rx status and error codes. |
| **PIPE Driver and Equalization Signal** | | |
| PhyTxMargin[2:0] | Out | Selects Tx voltage levels. This signal is fixed to 000b. |
| PhyTxSwing | Out | Conrols Tx voltage swing level. This signal is fixed to 0b. |
| PhyTxDeEmph[71:0] | Out | Selects Tx de-emphasis during equalization. |
| PhyRxPresetHnt[11:0] | Out | Signal used to trigger CTLE adaptation during equlization. |

## Timing Diagram

### Initialization

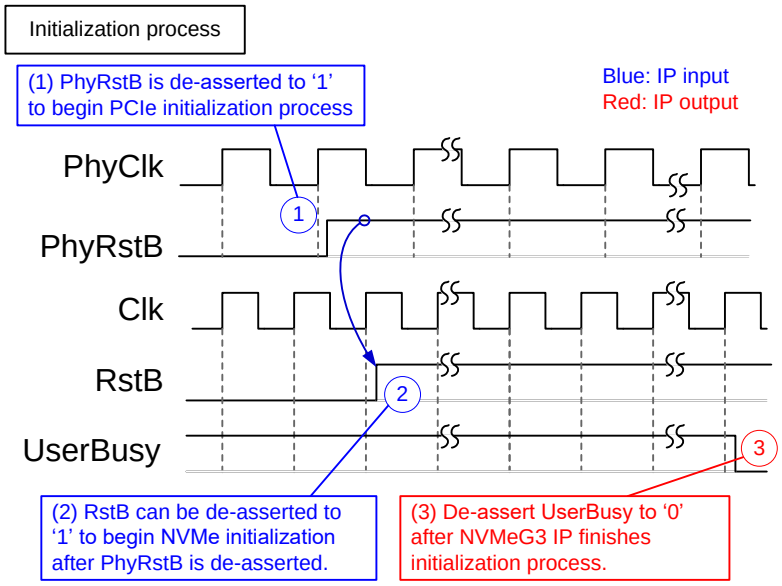


**Figure 4: Timing diagram of the reset sequence**

The sequences of the initialization process are as follows.

1) Wait until Transceiver Native PHY IP finishing reset sequence which can be monitored from tx_ready and rx_ready signals, output from Transceiver Native PHY IP. After that, user de-asserts PCIeRstB to '1' to begin the link training process by NVMeG3.
2) De-assert RstB to '1' after PhyRstB is de-asserted. NVMe logic within NVMeG3 IP starts the operation.
3) After IP finishes initialization processes such as link training, flow control initialization, and PCIe register and NVMe register configuration, UserBusy is de-asserted to '0'.

After complete above sequences, NVMeG3-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.

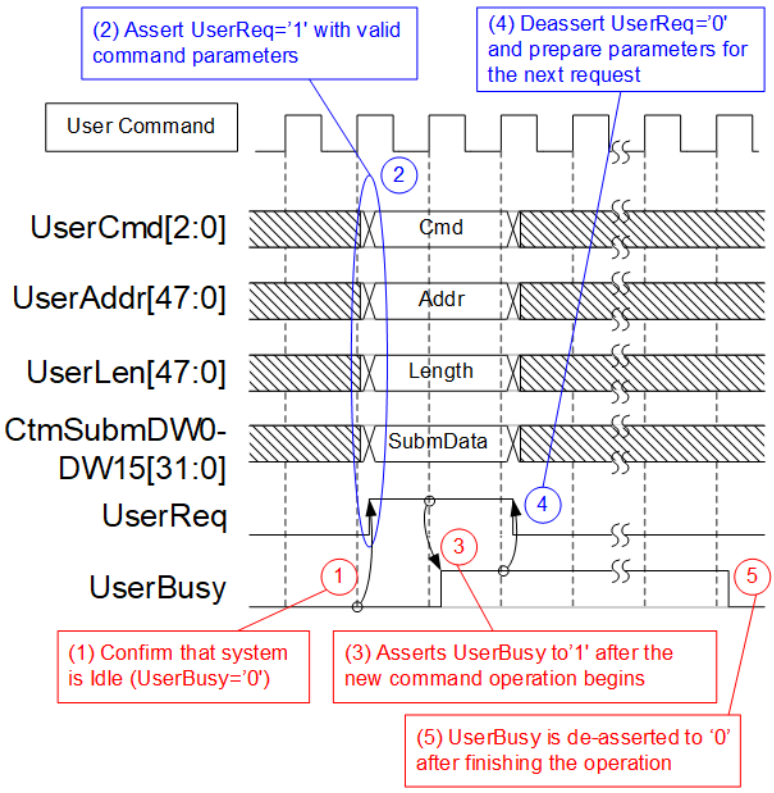


**Figure 5: Control Interface of dgIF typeS Timing diagram**

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 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'.

Note: The number of parameters using in each command is different.

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

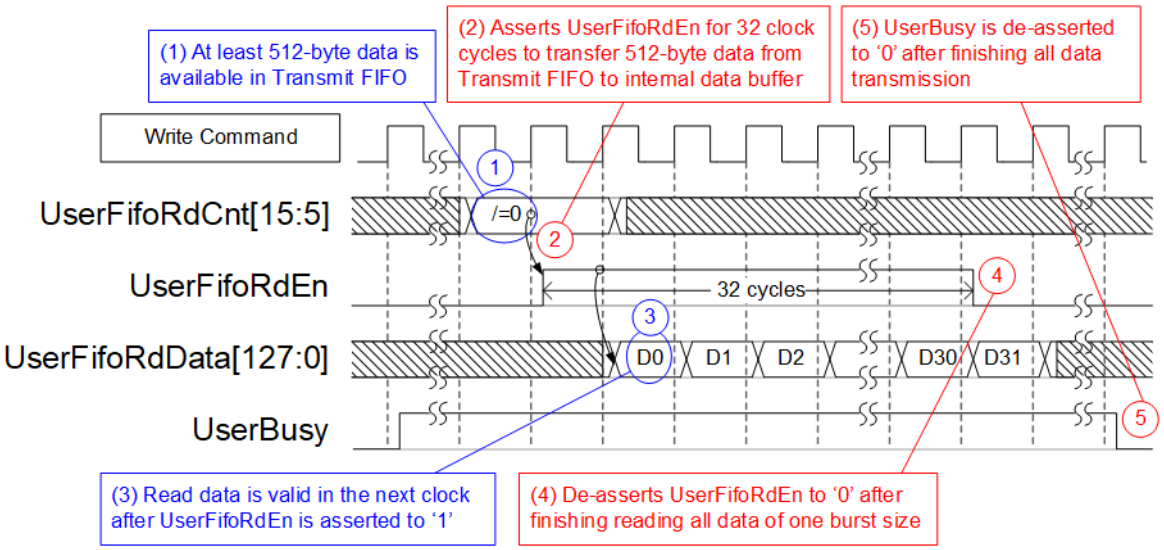


**Figure 6: Transmit FIFO Interface for Write command**

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) 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 transferred continuously.
4) UserFifoRdEn is deasserted to '0' after reading the 32$^{th}$ data. When total data is less than the transferred size in the command, the IP goes back to step 1) for transferring the next 512-byte data.
5) After total data have already transferred, UserBusy is de-asserted to '0'.

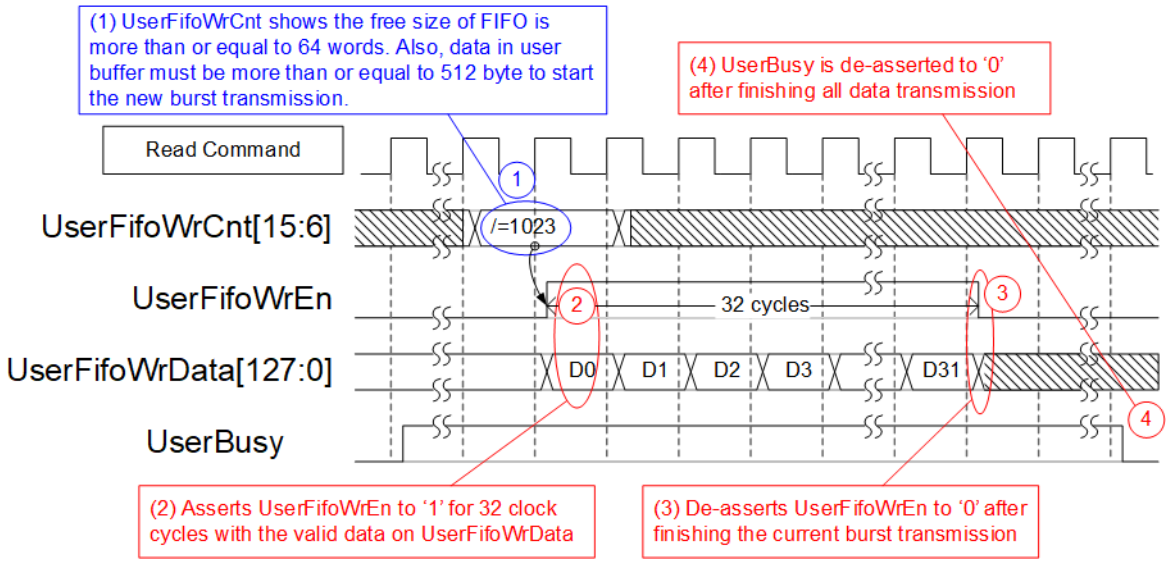


**Figure 7: Received FIFO Interface for Read command**

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

1) Before starting the new burst transmission, UserFifoWrCnt[15:5] is monitored. The IP waits until the free space of Received FIFO is much 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 byte, the new burst transmission begins.
2) Asserts UserFifoWrEn to '1' for 32 clock cycles to transfer 512-byte data from the internal buffer to user.
3) After transferring 512-byte data, UserFifoWrEn is de-asserted to '0'. When total data is less than the transferred size in the command, the IP goes back to step 1) for transferring the next 512-byte data.
4) After total data have already transferred, UserBusy is de-asserted to '0'.

The timing diagrams of user interface when running other commands are similar to the timing diagram described in NVMe IP datasheet. Please see more details from NVMe IP datasheet which can be downloaded from our website.

## Verification Methods

The NVMeG3 IP Core functionality was verified by simulation and also proved on real board design by using Arria10 GX Development board.

## Recommended Design Experience

Experience design engineers with a knowledge of QuartusII Tools should easily integrate this IP into their design.

## Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design Gatway Co., Ltd. For pricing and additional information about this product using the contact information on the front page of this datasheet.

## Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | Apr-27-2020 | New release |