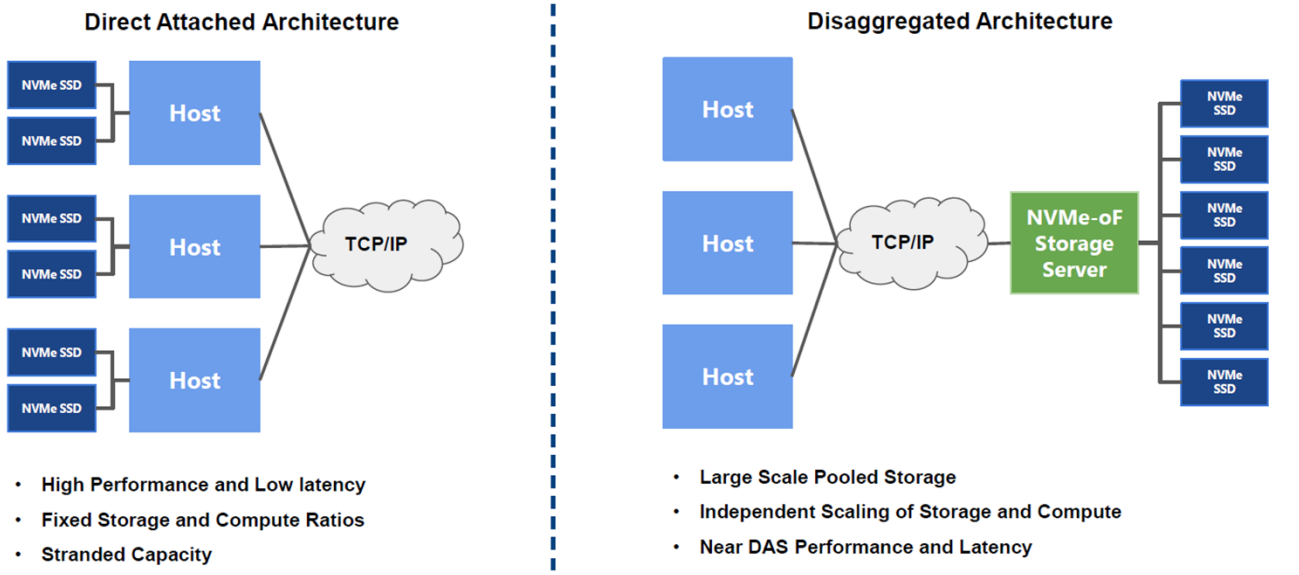
NVMeTCP-IP for 10G reference design manual

Rev1.0 17-Aug-23

1 Overview



Source: https://nvmexpress.org/wp-content/uploads/March-2019-NVMe-TCP-What-You-Need-to-Know-About-the-Specification.pdf

Figure 1-1 NVMe/TCP usage

Figure 1-1 shows the comparison between the traditional NVMe SSD by direct attached with the host via PCIe interface (NVMe over PCIe) and the new topology of NVMe storage that the host accesses across a network, called NVMe over Fabrics (NVMe-oF). While NVMe over PCIe achieves the good performance with the low latency, NVMe-oF supports the pooled and the scalable storage among many hosts. Therefore, the data in NVMe-oF storage is handled by many hosts which can be installed in the different location with the storage server.

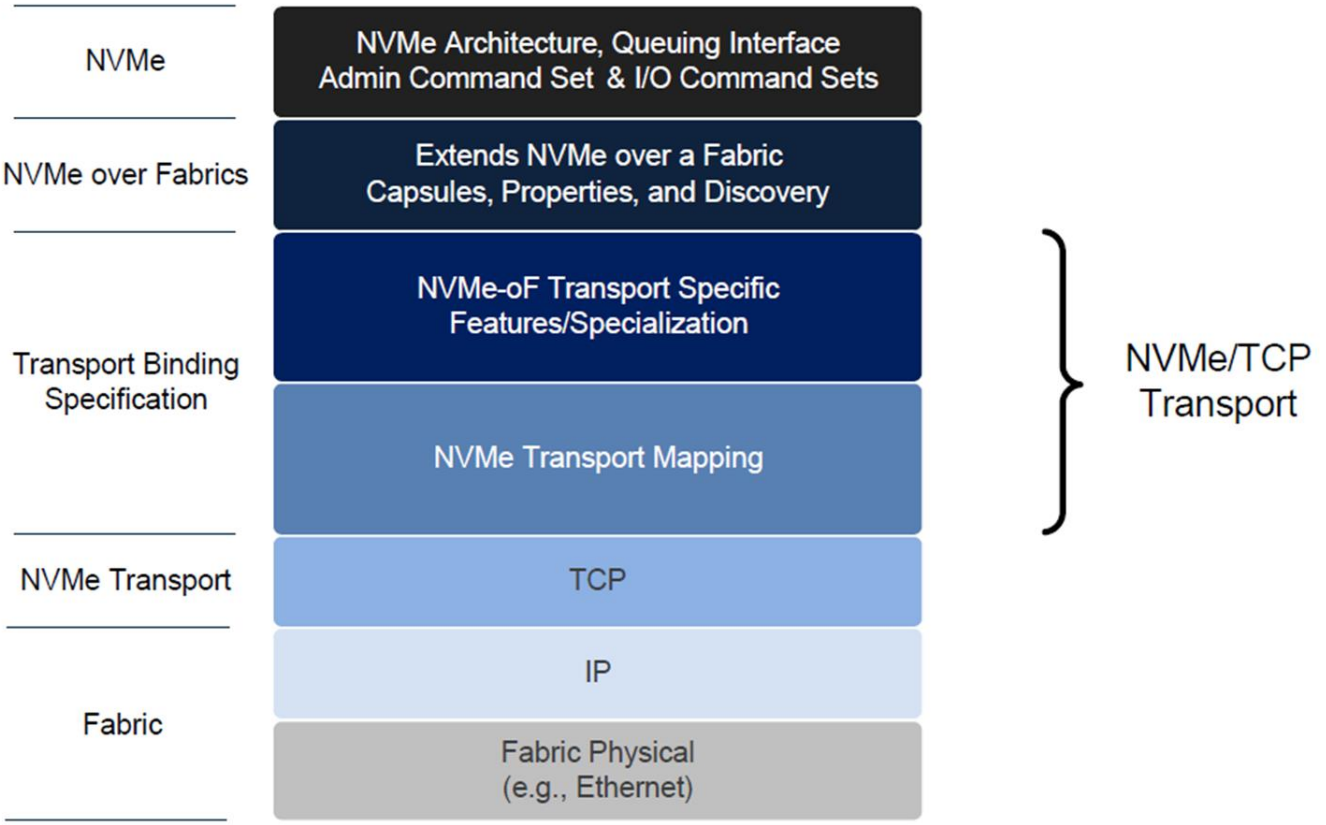


Figure 1-2 NVMe/TCP protocol layer

Figure 1-2 shows the details of protocol layer for NVMe/TCP. Similar to NVMe over PCIe, NVMe protocol is applied for the top layer which is the user application to handle the command and the status of the storage. Therefore, the specification of the command is referred to NVMe base specification which can be downloaded by following link.
https://nvmexpress.org/developers/nvme-specification/

To remote communication between the host and the storage server across the network, NVMe-oF is applied. NVMe-oF standard describes the use of capsules for commands, responses, and data transfers. Also, it includes the method for the host to establish a connection with NVM subsystem (SSD connected with the server in Figure 1-1) and the discovery mechanism for the host to determine which NVM subsystem accessed. More details of NVMe-oF specification can be downloaded by following link.
https://nvmexpress.org/developers/nvme-of-specification/

Table 1: Characteristics of NVMe over Fabric Networks

	FC-NVMe	NVMe/RoCE	NVMe/TCP
Bandwidth	16-32GFC	10-100GbE	10-100GbE
IOPS	~1.2M	~1.5M	~1.5M
Network Type	Fibre Channel SAN	Lossless Ethernet	Ethernet Network
Scalability	Yes, dedicated fabric	Limited – 1 to 2 hops	Yes
Adapter Latency	Low	Very Low	High
OS Support	Linux, VMware, Windows	Linux, VMware	Linux, VMware
Security	Yes, dedicated FC network	No	No

Source: https://www.marvell.com/content/dam/marvell/en/public-collateral/fibre-channel/marvell-nvme-over-fabrics-technology-brief.pdf

Figure 1-3 NVMe-oF comparison

To implement NVMe-oF, there are three well-known standards, i.e., FC, RoCE (RDMA over Converged Ethernet), and TCP/IP. Even though FC and RoCE can outstanding achieve low-latency access, they require the specific hardware for the network system which greatly expands the overall system cost. Whereas, NVMe over TCP (NVMe/TCP) can be implemented by using ubiquitous TCP/IP protocol. However, in term of latency time for storage accessing, NVMe/TCP does not overcome both FC and RoCE protocol. More details about the NVMe transport specification can be downloaded by following link.
https://nvmexpress.org/developers/nvme-transport-specifications/

TCP/IP provides a reliable data stream transferring between a sender and receiver. Also, it supports bi-directional transferring by using the same connection. NVMe/TCP uses TCP/IP to transfer NVMe/TCP Protocol Data Units (PDUs). More details about TCP/IP protocol are described in following link.
https://tools.ietf.org/html/rfc793

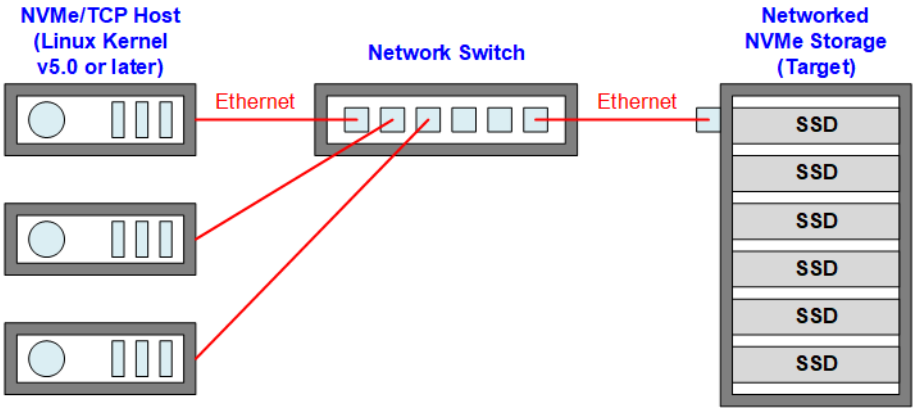


Figure 1-4 General system of NVMe/TCP

Figure 1-4 shows the general system which implements NVMe/TCP. NVMe/TCP host is the device that sends the request for accessing the storage while NVMe/TCP target is the device that has the storage device. The host can be implemented by using the server or PC installed LinuxOS (Kernel version 5.0 or later). However, limited resources and overhead processes via OS cause some disadvantages, especially performance bottleneck. Thus, hardware logic in FPGA steps to take the role of NVMe/TCP device to improve the system performance.

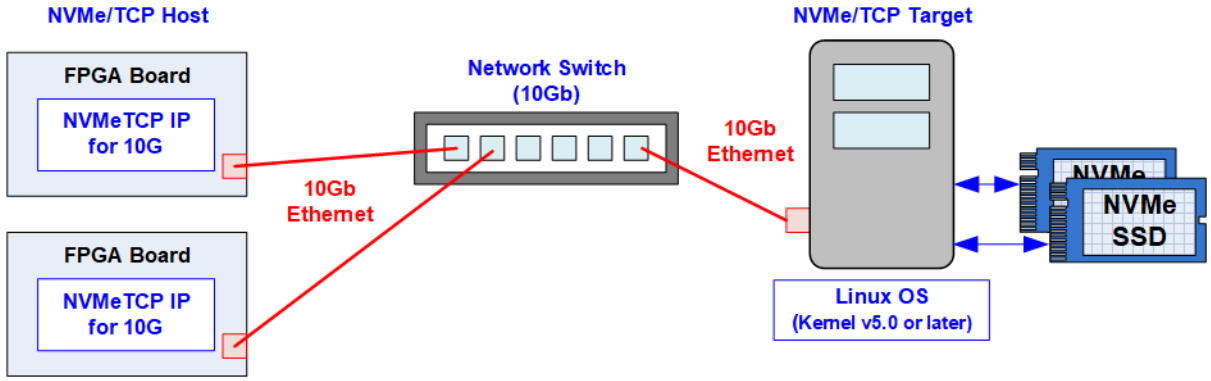


Figure 1-5 NVMe/TCP system by using NVMeTCP10G-IP

Figure 1-5 shows the system that applies NVMeTCP10G-IP to be the host controller for accessing NVMe SSD. One host has one connection for accessing one SSD. To change the SSD, the host needs to terminate the connection of the active SSD and then create the new connection for the new SSD. Without CPU and DDR, NVMeTCP10G-IP in FPGA is designed as standalone hardware in the system which can achieve higher performance to access the storage over general host system, implemented by PC.

2 Hardware overview

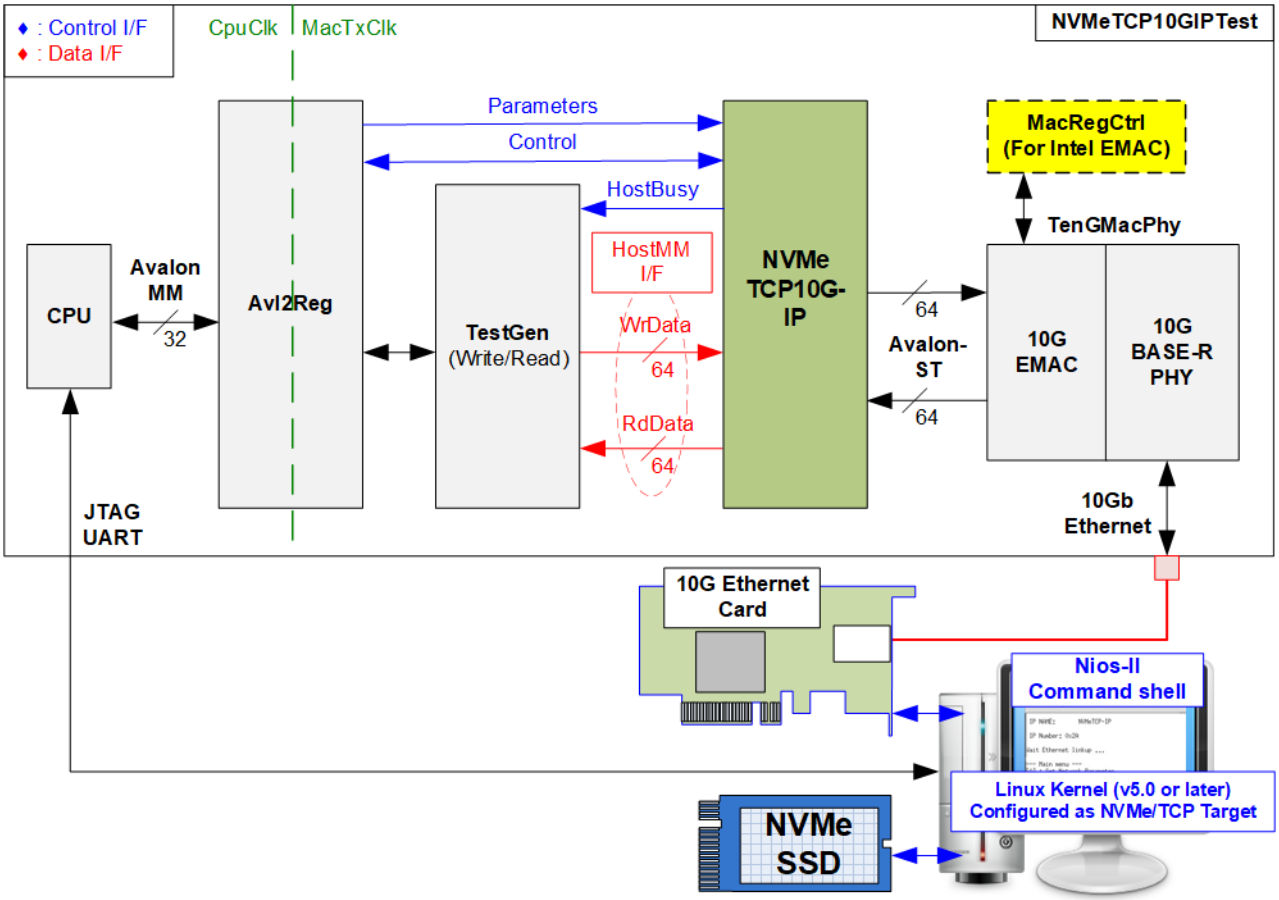


Figure 2-1 NVMeTCP10G-IP demo hardware

Following the function of each module, all hardware modules inside the test system are divided to three parts, i.e., NVMe/TCP function (NVMeTCP10G-IP and TenGMacPhy), test function (TestGen), and CPU system (CPU and Avl2Reg).

To connect with 10G Ethernet interface, NVMeTCP10G-IP connects to 10G Ethernet MAC and 10G Ethernet PHY (10GBASE-R PHY) as shown in Figure 2-1. User interface of NVMeTCP10G-IP consists of Parameters, Control, and HostMM interface. These three interfaces connect with 2 components. The first component is test logic (TestGen). TestGen generates test data stream directly to the IP and verifies received data stream output from the IP through HostMM interface. HostBusy, a signal of Control interface, is sent from the IP to TestGen to confirm command completion. The second component is CPU and peripherals (CPU and Avl2Reg) which connect to NVMeTCP10G-IP via Parameters interface and Control interface. CPU and Avl2Reg are designed to interface with user via JTAG UART. On the console, user can set command and the test parameters to TestGen and the IP. Also, the current status of the test hardware is monitored by user on the console through Control interface. The CPU firmware must be implemented to control the flow for operating each command.

Besides, an additional logic, MacRegCtrl, is designed in the reference design when the system uses Intel FPGA 10G EMAC-IP. This module is applied to configure control register of Intel 10G EMAC-IP through Avalon-MM interface. This module is not necessary when using DG 10G EMAC-IP.

Two clock domains are applied in the test design, i.e., CpuClk which is the independent clock for running the CPU system and MacTxClk which is the clock output from 10G Ethernet PHY. Therefore, AsyncAvlReg is designed to support asynchronous signals between CpuClk and MacTxClk. More details of each module inside the NVMeTCP10GIPTest are described as follows.

2.1 NVMe/TCP

2.1.1 10GBASE-R PHY

Intel 10GBASE-R PHY is applied to connect with 10G SFP+ module. User interface of PHY is connected to 10G Ethernet MAC through 64-bit XGMII interface at 156.25 MHz. More details are described in following link.
https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/10g-base-r-pcs.html

2.1.2 10G EMAC

10G EMAC connects between NVMeTCP10G-IP and 10GBASE-R PHY. The interface with NVMeTCP10G -IP is 64-bit Avalon stream while the interface with 10GBASE-R PHY is 64-bit XGMII at 156.25 MHz. Both DG 10G EMAC-IP and Intel FPGA 10G EMAC-IP can be used in the reference design. More details of 10G EMAC IP are described in the following link.

DG 10G EMAC-IP
https://dgway.com/products/IP/10GEMAC-IP/dg_tengemacip_data_sheet_intel_en.pdf

Intel FPGA 10G EMAC IP
https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/low-latency-10gbps-ethernet-mac.html

When using Intel FPGA 10G EMAC-IP, MacRegCtrl must be included to configure 10G EMAC parameters through 32-bit Avalon-MM bus interface. MacRegCtrl is designed as Avalon-MM master for writing and reading register within 10G EMAC IP. The details of the hardware inside MacRegCtrl are shown in Figure 2-2.

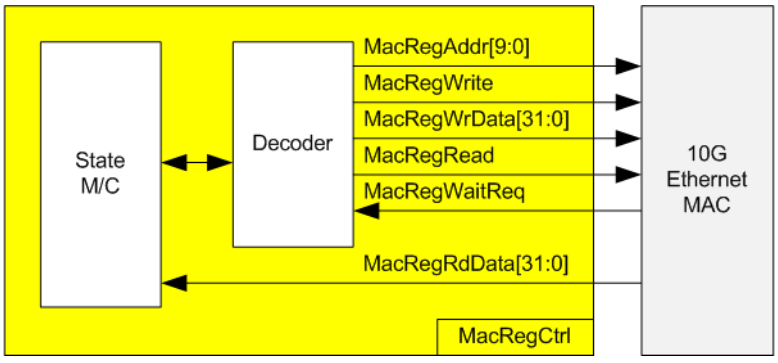


Figure 2-2 MacRegCtrl block diagram

State machine is designed to write/read the register as following sequence.

1) Disable transmit and receive path of EMAC.
2) Wait until transmit and receive path are idle.
3) Set receive module to remove CRC and padding.
4) Disable pause frame transmission.
5) Enable transmit and receive path of EMAC.

2.1.3 NVMeTCP-IP for 10G

The NVMeTCP10G-IP implements NVMe over TCP which additionally includes TCP/IP stack and offload engine of the host controller to access one target SSD via 10G Ethernet. The NVMeTCP10G-IP supports Write and Read commands. The user needs to connect the IP (host) to the target before writing/reading. After finishing, user disconnects the target from the host. User interface has three signal groups, i.e., Parameters, Control, and Memory Map (HostMM) Interface. Please see more details of NVMeTCP10G-IP on our website.
https://dgway.com/products/IP/NVMeTCP-IP/dg_nvmetcp10g_ip_data_sheet_intel.pdf

2.2 TestGen

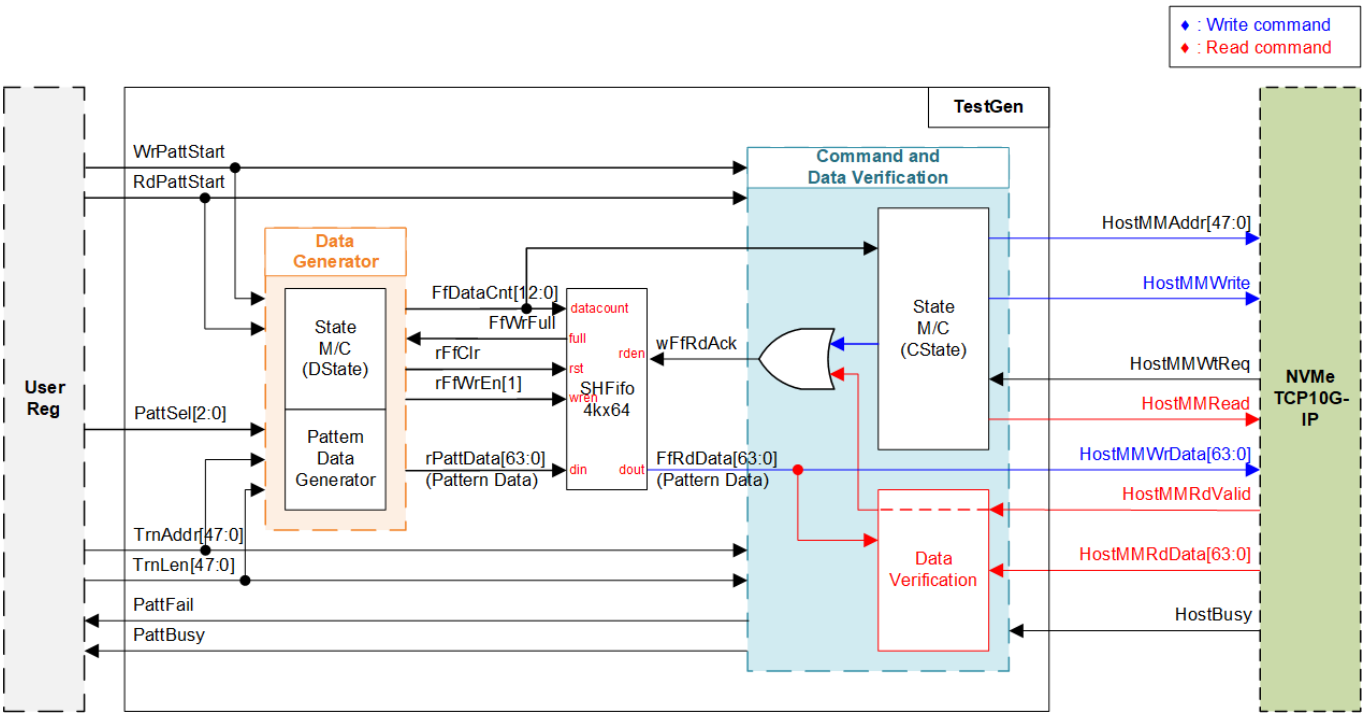


Figure 2-3 TestGen Interface

TestGen generates Command (HostMMWrite and HostMMRead) and the address (HostMMAddr) to NVMeTCP10G-IP when user asserts Write or Read Request (WrPattStart or RdPattStart). Also, 4-Kbyte test pattern for sending or verifying data in a Write or Read Command is created in TestGen. HostMMWtReq is asserted by NVMeTCP10G-IP when the IP is not ready to receive more commands or more data from TestGen. Therefore, SHFifo is included to store test data generated by Data Generator when NVMeTCP10G-IP is not ready for receiving more commands or data.

The logic inside TestGen is divided into two groups to handle write interface and read interface of FIFO. Data Generator generates test data to store to FIFO while Command and Data Verification sends the command request and transfers data from FIFO to NVMeTCP10G-IP. There is a state machine for controlling the operation in each group, DState for Data Generator and CState for Command and Data Verification.

FIFO is Show-ahead type to store Test data from Data Generator. When running Write Command, Command and Verification reads prepared test data out from the FIFO to be Write Data (HostMMWrData) in HostMM interface. When running Read Command, Command and Verification waits Read Data (HostMMRdData) in HostMM interface returned from NVMeTCP10G-IP. After that, Command and Verification reads prepared test data out from the FIFO for verifying correctness of the Read Data (HostMMRdData).

Data Generator

Data Generator inside the TestGen consist of two logic groups, i.e., State machine to control FIFO write enable (DState), and Pattern Data Generator to generate FIFO write data (rPattData). When user starts the test by asserting Write or Read Request (WrPattStart or RdPattStart), test pattern data is generated in Pattern Data Generator. The prepared test pattern data is written to FIFO by State machine which controls the flow of FIFO writing. According to NVMeTCP10G-IP specification, data size in a Write or Read Command is fixed to 4 Kbytes. Thus, Data Generator writes data in burst mode by 4-Kbyte size to FIFO in each round, controlled by Data State Machine (DState) described as follows.

(1) stDIdle: This state is designed to wait Write or Read Request (WrPattStart or RdPattStart) from user. When user starts the test by asserting Write or Read Request (WrPattStart or RdPattStart) along with test parameters i.e., total transfer size (TrnLen: 512-byte unit), start transfer address (TrnAddr: 512-byte unit), and test pattern (PattSel), the logic in Data Generator loads the first value from test parameters. Next, continue the next state.
(2) stDRst: This state asserts FIFO reset signal (rFfClr) to clear the FIFO before beginning the test. The state is run for one clock cycle.
(3) stDWtRst: This state is designed to wait until FIFO is ready to be written. Also, FIFO reset signal (rFfClr) is de-asserted. After FIFO full signal (FfWrFull) is de-asserted, state continues to the next step.
(4) stDWtFf: This state is designed to wait until free size of FIFO is enough by checking FIFO Data Counter (FfDataCnt). Continue the next state if the FIFO space is enough for 8 Kbytes (2 burst sizes).
(5) stDHd: This state is one clock cycle to prepare 64-bit header of each 4- Kbyte data.
(6) stDPatt: In this state, 4-Kbyte pattern data is written to the FIFO by asserting FIFO Write Enable (rFfWrEn) to '1' for 512 clock cycles. Also, Pattern Data Generator uses look-ahead style to prepare next 64-bit pattern data for the next clock writing. When the 511th pattern data is prepared, state exit choices are considered as follows.
    a. If the current FIFO writing is the last 4 Kbytes of the testing (indicated via rPattLenCnt), state continues to stDIdle (1) to end the FIFO writing.
    b. If the current FIFO writing is not the last 4 Kbytes but FIFO space is less than 8 Kbytes (2 burst sizes), state continues to stDWtFf (4) to finish current 4 Kbytes writing. It needs to wait more FIFO space for the next 4-Kbyte writing. This state transition triggers the logics to write the last 64-bit data to FIFO and de-asserts FIFO Write Enable (rFfWrEn) to '0'.
    c. If the current FIFO writing is not the last 4 Kbytes and FIFO space is enough for 8 Kbytes (2 burst sizes), state continues to stDHd (5). This state transition allows the logics to write the next 4-Kbyte data to FIFO continuously after finishing transferring current 4 Kbyte data. Thus, FIFO Write Enable (rFfWrEn) is still set to '1' for another 512-clock cycle.

From this test logic design, FIFO size is 32 Kbytes (support 8 burst sizes) and FIFO writing is paused when FIFO space is less than 8 Kbytes (2 burst sizes). Thus, pattern data is usually prepared and stored in the FIFO via this Data State Machine of Data Generator. After Command and Verification starts reading the FIFO, 64-bit prepared pattern data can be read out from the FIFO for at least 512 clock cycles. Timing diagram when Data Generator writes pattern data to the FIFO in burst mode is shown in Figure 2-4.

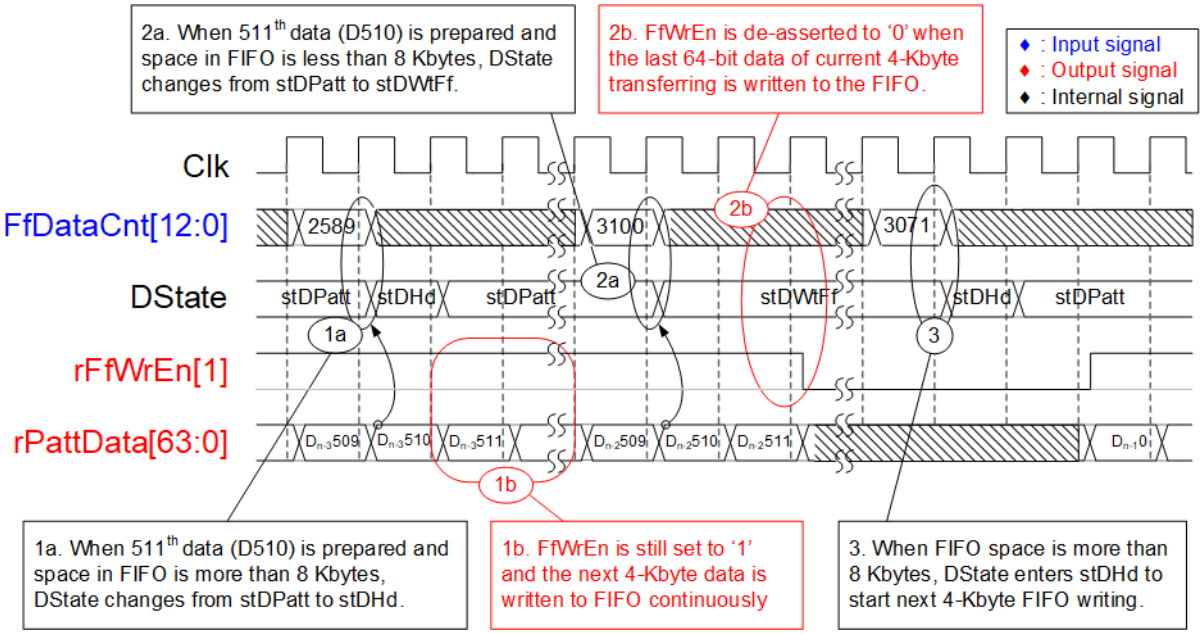


Figure 2-4 Timing diagram of Data Generator when writing pattern data to FIFO

1) When Data Generator finishes preparing the 511th 64-bit pattern data of the current 4-Kbyte transferring, FIFO space is checked. If the space is more than 8 Kbytes (FfDataCnt<3072), DState continues to stDHd to write next 4-Kbyte pattern continuously (rFfWrEn[1]='1').
2) When Data Generator finishes preparing the 511th 64-bit pattern data of the current 4-Kbyte transferring and FIFO space is less than 8 Kbytes (FfDataCnt>=3072), DState continues to stDWtFf to pause the FIFO writing (rFfWrEn[1]='0')
3) FIFO writing is continued when FIFO space is enough for 8 Kbytes (2 burst sizes).
   Note: Although FIFO size is 32 Kbytes (support 8 burst sizes), free space threshold for stopping FIFO writing is 8-Kbyte space to prevent FIFO from being full.

Command and Data Verification

Command and Data Verification inside the TestGen consist of two logic groups, i.e., State machine (CState) and Data Verification. When user starts the test by asserting Write or Read Request (WrPattStart or RdPattStart), State machine sends Write Command (HostMMWrite) or Read Command (HostMMRead) to NVMeTCP10G-IP. State Machine (CState) consists of six states, described as follows.

(1) stCIdle: This state is designed to wait Write or Read Request (WrPattStart or RdPattStart) from user. When user starts the test along with test parameters, i.e., total transfer size (TrnLen: 512-byte unit), start transfer address (TrnAddr: 512-byte unit), and test pattern (PattSel), the logics load the first value from test parameters. Moreover, PattBusy is asserted to '1' to indicate that the test is running. If user starts the reading test by asserting Read Request (RdPattStart), state continues to stCRdTrn (2). On the other hand, if user starts the writing test by asserting Write Request (WrPattStart), state continues to stCWrWt (3).

(2) stCRdTrn: In this state, Read Command (HostMMRead) is sent to NVMeTCP10G-IP in HostMM interface along with the start read address (HostMMAddr) of each 4-Kbyte transferring. After sending the last Read Command, state continues to stCWtEnd (6) to wait until all read data is returned and IP finishes the operation.

(3) stCWrWt: This state is designed to wait until data is enough in FIFO via checking FIFO Data Counter (FfDataCnt). If the data in FIFO is more than or equal to 4 Kbytes (1 burst size), state continues to the next step.

(4) stCWrTrn: This state is designed to read 4-Kbyte FIFO data and send them out as Write Data (HostMMWrData) to NVMeTCP10G-IP along with the Write Command (HostMMWrite). When write command and write data are received (HostMMWrite='1' and HostMMWtReq='0'), FIFO Read Enable (wFfRdAck) is asserted to read the next data from FIFO. The FIFO Read Data (FfRdData) is bypassed to be HostMMWrData (as shown Figure 2-3). At the first clock cycle of this state, the first 64-bit Write Data (HostMMWrData), the address (HostMMAddr), and Write Command (HostMMWrite) are sent to the IP. After the 511th of 64-bit Write Data (HostMMWrData) is sent to the IP, state continues to the next step.

(5) stCWrChkFf: This state is designed to send the last 64-bit Write Data (HostMMWrData) of the current 4-Kbyte transferring. Moreover, state machine checks data amount in FIFO via checking FIFO Data Counter (FfDataCnt) for next 4-Kbyte bursting decision. After the last 64-bit Write Data (HostMMWrData) of the current 4-Kbyte transferring is sent, state exit choices are considered as follows.

   a. If the current 4-Kbyte Write Data is not the last 4 Kbytes of the Write Command and data amount in FIFO is not enough for the next round, state continues to stCWrWt (3) to wait until Data Generator fills more data to the FIFO. This state transition triggers the logics to de-assert Write Command (HostMMWrite) to '0'.
   b. If the current 4-Kbyte Write Data is not the last 4 Kbytes of the Write Command and data amount in FIFO is enough for the next round, state continues to stCWrTrn (4). This state transition allows the logic to start sending the next 4 Kbytes data continuously after finishing transferring current 4-Kbyte data without de-asserting Write Command (HostMMWrite) to '0'.
   c. If the current 4-Kbyte Write Data is the last 4 Kbytes of the Write Command (indicated via rMMLenCnt), state continues to stCWtEnd (6). Also, Write Command (HostMMWrite) is de-asserted to '0' to end the writing test.

   *Note: There is latency time to update FIFO Data Counter after asserting FIFO Read enable. Thus, the threshold value to check 1-burst-size data being available in FIFO in stCWrChkFf must be more than 4 Kbytes to compensate latency time of FIFO Data Counter. For simple design, the threshold value in this state is equal to 4352 bytes (*FfDataCnt=544*).*

(6) stCWtEnd: This state is designed to wait until the writing or reading test is done. TestGen waits until NVMeTCP10G-IP finishes its internal processes by monitoring the IP busy flag (HostBusy) with Flip Flop. When HostBusy with Flip Flop is '0' in this state, PattBusy is de-asserted to '0' to notify user that the test is done. State returns back to stCIdle (1).

Data Verification works only in reading test and its working process is not controlled by Command State Machine (CState). After user starts the reading test by asserting Read Request (RdPattStart), Data Verification waits until Read Data (HostMMRdData) is returned from NVMeTCP10G-IP. Data Verification uses validation signal of Read Data (HostMMRdValid) to be FIFO Read Enable (wFfRdAck) for reading pattern data out from the FIFO. Each 64-bit Read Data (HostMMRdData) and pattern data (FfRdData) are compared to verify the Read Data (HostMMRdData) correctness. If a 64-bit Read Data (HostMMRdData) does not match the pattern, Data Verification asserts PattFail to '1' to notify user about the verification failure.

The timing diagram examples of Command and Data Verification when running Write Command and Read Command are shown as follows.

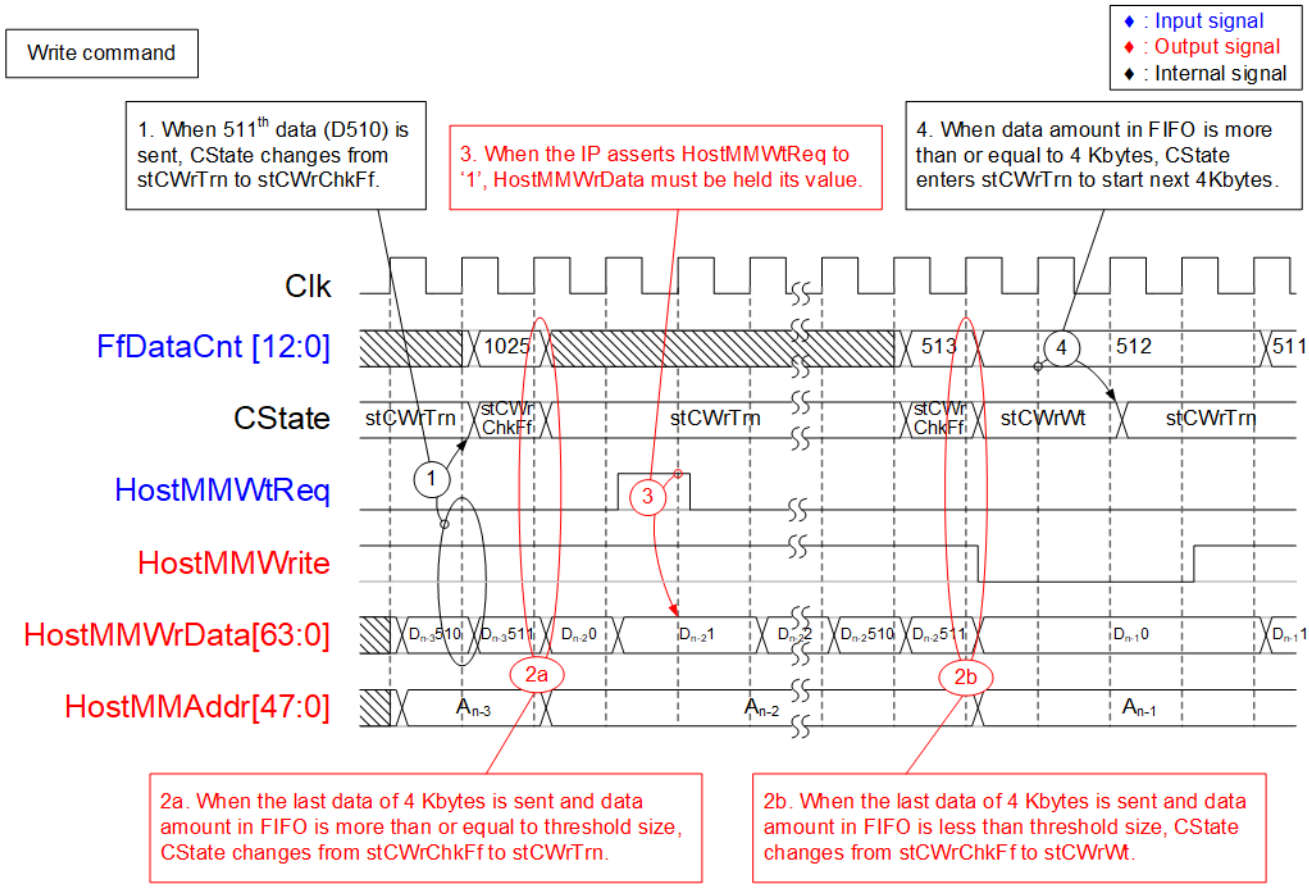


Figure 2-5 Timing diagram of Command S/M when sending each 4-Kbyte Write Command

1) When the 511th 64-bit Write Data (HostMMWrData) of the current 4 Kbytes is sent to the IP, CState continues to stCWrChkFf for next 4-Kbyte bursting decision.
2) In stCWrChkFf, the last 64-bit Write data (HostMMWrData) of each 4-Kbyte Write Command is sent to the IP and data amount in FIFO is checked.
    a. If data amount in FIFO is enough (4352 bytes: FfDataCnt>544), CState enters stCWrTrn to start next 4-Kbyte Write Command continuously. Also, Write Command (HostMMWrite) is not de-asserted to '0'.
    b. If data amount in FIFO is too less (less than or equal to 4352 bytes: FfDataCnt<=544), CState enters stCWrWt to wait the FIFO Data Counter (FfDataCnt) updating. Write Command (HostMMWrite) is de-asserted to '0' to finish current 4 Kbytes.
3) The validation of each 64-bit Write Data (HostMMWrData) is considered from HostMMWtReq, output of NVMeTCP10G-IP. At the point that HostMMWtReq is '0' and the Write Command (HostMMWrite) is '1', the 64-bit Write Data (HostMMWrData) is considered as successfully sent to the IP. On the other hand, when the IP pauses the Write Command by asserting HostMMWtReq to '1', the value of HostMMWrData must be held.
4) While FIFO data amount is checked in stCWrChkFf and at least 4-Kbyte (4096 bytes) data is ready in FIFO, CState continues to stCWrTrn to start next 4-Kbyte Write Command.

PattBusy is the output signal of TestGen for the user monitoring complete status after sending the Write or Read request.

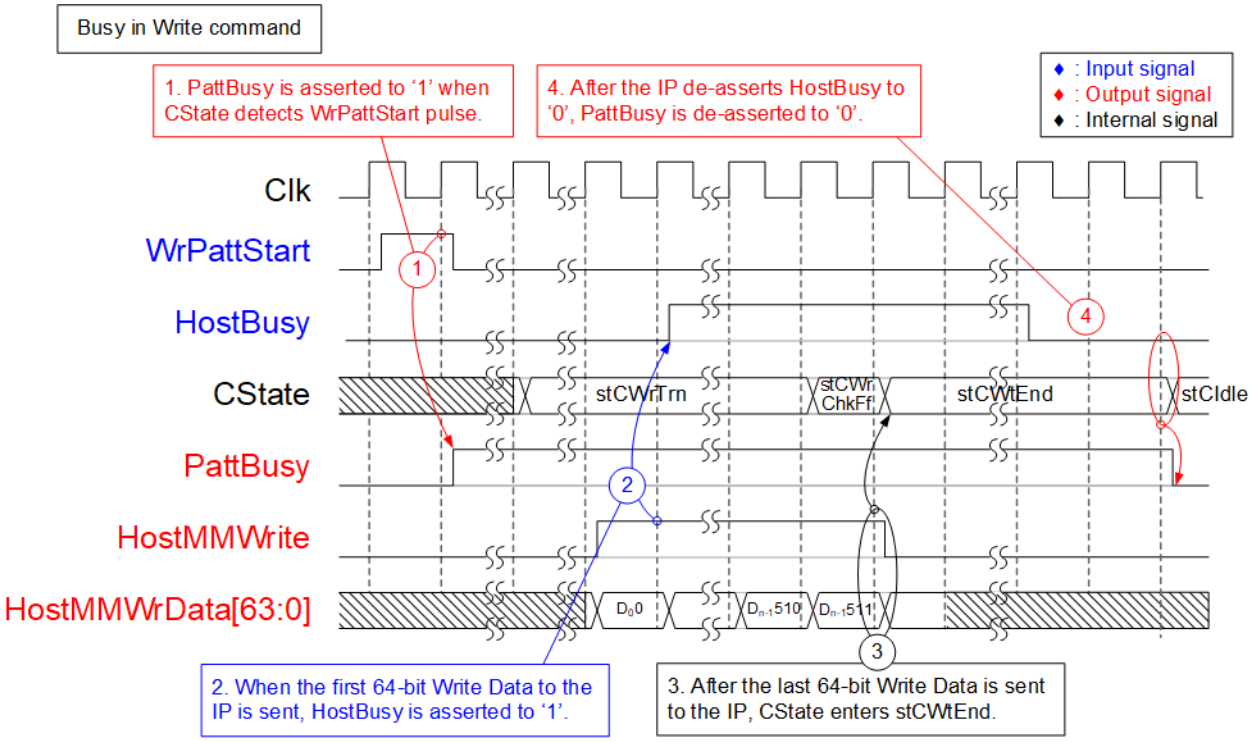


Figure 2-6 Timing diagram of PattBusy when running Write Command

1) When user starts the writing test by asserting Write Request (WrPattStart), test busy flag (PattBusy) is also asserted to '1' to notify user that the test is running.
2) When TestGen sends the first 64-bit Write Data (HostMMWrite) to the IP, IP busy flag (HostBusy) is asserted to '1' referred from NVMeTCP10G-IP specification.
3) After TestGen finishes sending the last Write data to the IP, CState enters stCWtEnd to wait internal writing processes of NVMeTCP10G-IP.
4) When the IP finishes its internal processes, HostBusy is de-asserted to '0'. After that, PattBusy is de-asserted to '0' to notify user that the test is done. Also, CState returns back to stCIdle.

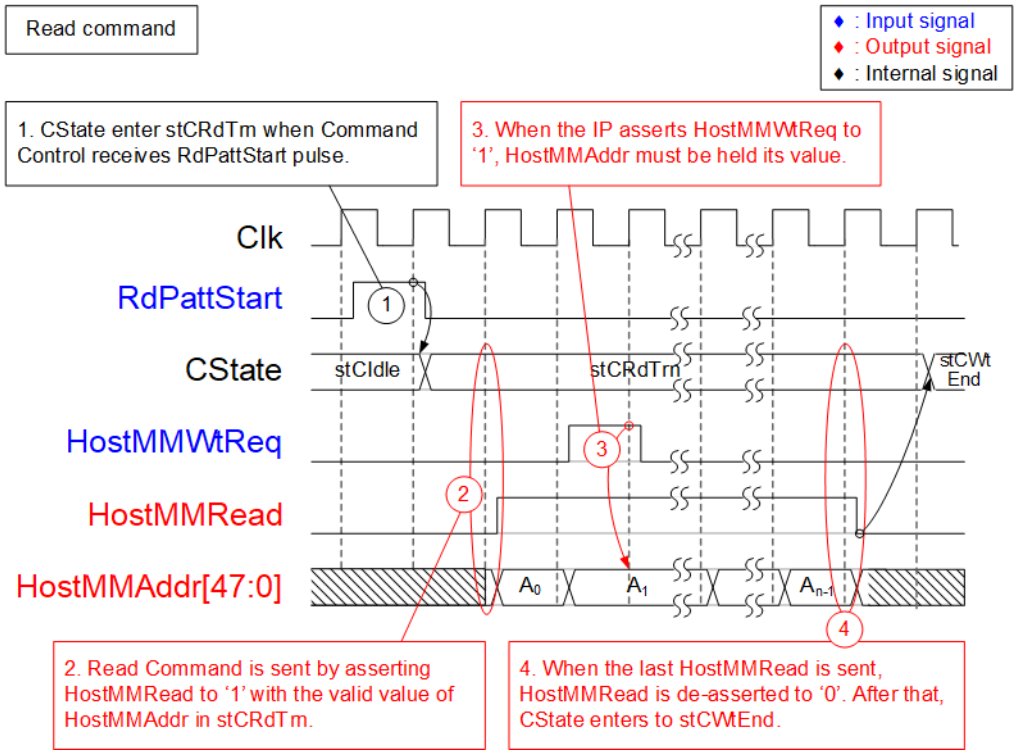


Figure 2-7 Timing diagram of Command S/M when sending each Read Command

1) When user starts the reading test by asserting Read Request (RdPattStart), CState changes to stCRdTrn to start sending Read Command (HostMMRead).
2) At the first clock cycle of stCRdTrn, Read Command (HostMMRead) is asserted to '1' with the address (HostMMAddr).
3) The validation of each Read Command (HostMMRead) is considered from HostMMWtReq, output of NVMeTCP10G-IP. At the point that HostMMWtReq is '0', the Read Command (HostMMRead) with the address (HostMMAddr) is considered as successfully sent to the IP. On the other hand, when the IP pauses the Read Command by asserting HostMMWtReq to '1', the value of HostMMAddr must be held. When a Read Command (HostMMRead) is successfully sent to the IP, the next Command is sent continuously without waiting 4-Kbyte Read Data (HostMMRdData) returning from the IP.
4) After Read Command (HostMMRead) has been asserted, the value is held until the last Command is sent. When the last Read Command (HostMMRead) is sent to the IP, Read Command (HostMMRead) is de-asserted to '0'. After that, CState changes to stCWtEnd to wait until IP finishes Read command and returns all Read data.

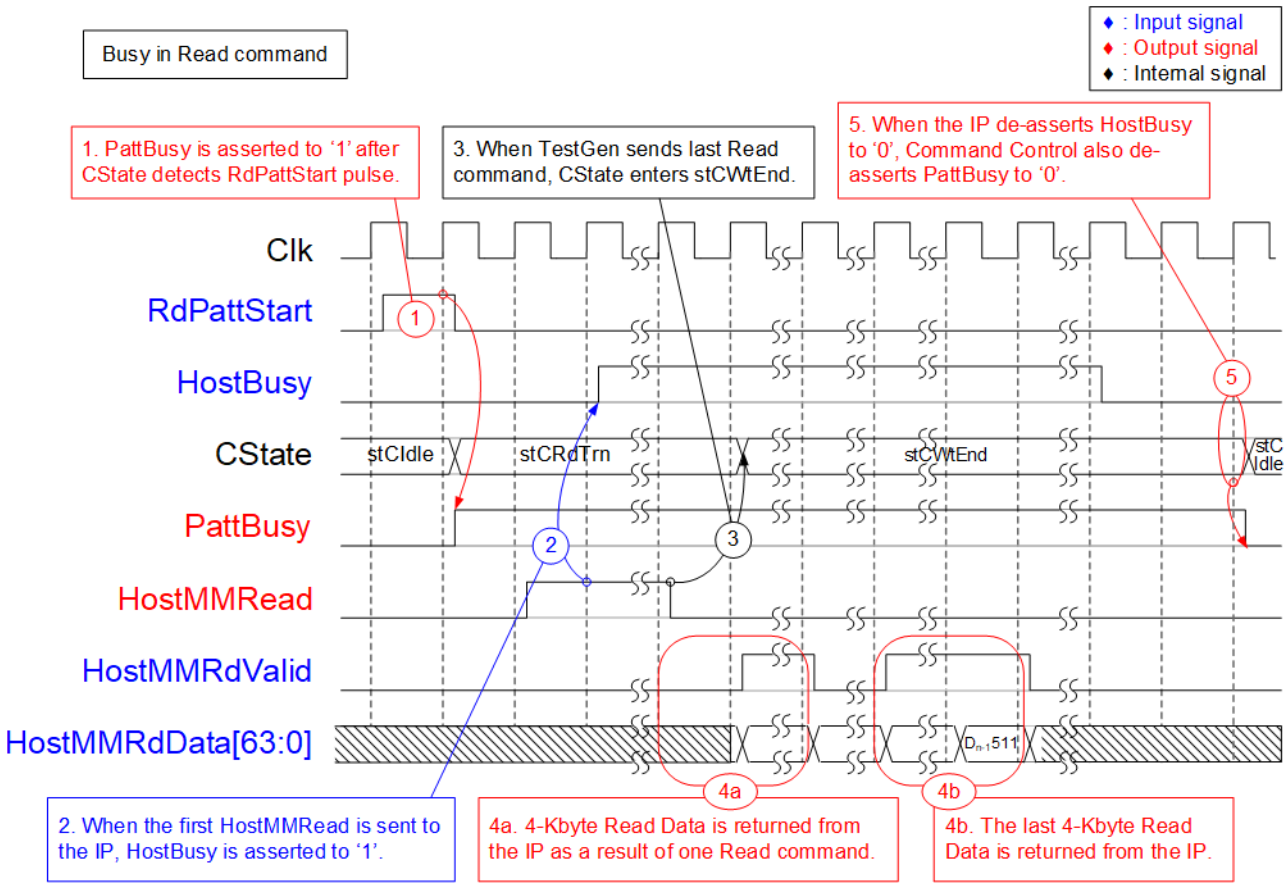


Figure 2-8 Timing diagram of PattBusy when running Read Command

1) When user starts the reading test by asserting Read Request (RdPattStart), test busy flag (PattBusy) is asserted to '1 to notify user that the test is running.
2) When the first Read Command (HostMMRead) is sent to the IP, IP busy flag is asserted to '1' referred from NVMeTCP10G-IP specification.
3) After the last Read command is sent, the state enters to stCWtEnd.
4) The command (HostMMRead) and the read data (HostMMRdData) are transferred independently. When the IP is ready, 4-Kbyte Read data of the first command is returned continuously (512 clock cycles) without de-asserting the Read Data Valid (HostMMRdDataValid). The order of each 4-Kbyte data returned back from the IP is the same order as Read Command which TestGen is sent. After TestGen receives the last Read Data (HostMMRdData) from the IP, NVMeTCP10G-IP has de-asserted HostBusy to '0' (referred from NVMeTCP10G-IP specification).
5) PattBusy is de-asserted to '0' to notify user that the test is done. Also, CState returns back to stCIdle.

Test Data

The test data is generated and stored to FIFO by Data Generator and read out from FIFO by Command and Data Verification to be Write Data (HostMMWrData) or expected data to verify with Read Data (HostMMRdData). The test data of one Command is 4-Kbyte size which consists of 64-bit header data and the test pattern, selected by PattSel.

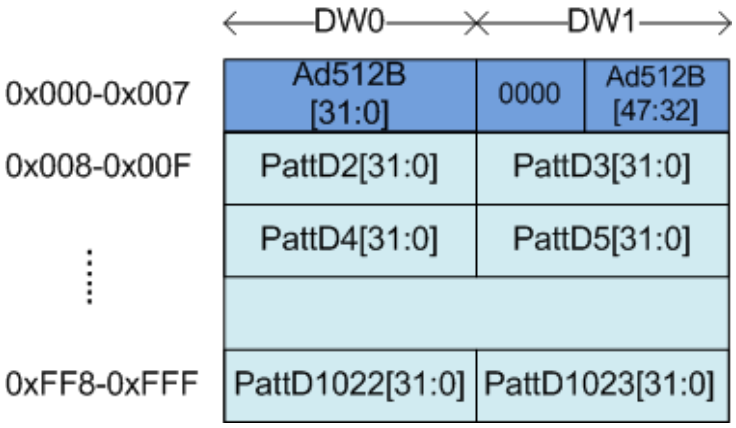


Figure 2-9 Test pattern format of 4096-byte data for Increment/Decrement/LFSR pattern

As shown in Figure 2-9, 4-Kbyte data consists of 64-bit header in DW#0 and DW#1 created by using 48-bit address value (512-byte unit) of a physical address of the target SSD. Remaining data (DW#2 – DW#1023) is the test pattern which can be selected by three formats: 32-bit incremental data, 32-bit decremental data, and 32-bit LFSR counter. 32-bit incremental data is designed by using the up-counter. The decremental data can be designed by connecting NOT logic to incremental data. The equation of 32-bit LFSR data is x^31 + x^21 + x + 1. Two 32-bit LFSR data must be generated in the same clock to create 64-bit data, so the LFSR counter logic uses look-ahead style to generate two LFSR data in one clock cycle.

In addition, the user can select test pattern to be all zero or all one data to show the best performance of some SSDs which have data compression algorithm in SSD controller. When the pattern is all zero or all one, there is no 64-bit header inserted to 4 KB data.

2.3 CPU and Peripherals

32-bit Avalon-MM bus is applied to be the bus interface for CPU accessing the peripherals such as Timer and JTAG UART. The test system of NVMeTCP10G-IP is connected with CPU as a peripheral on 32-bit Avalon-MM bus for CPU controlling and monitoring. CPU assigns the different base address and the address range to each peripheral for accessing one peripheral at a time.

In the reference design, the CPU system is built with one additional peripheral to access the test logic. So, the hardware logic must be designed to support Avalon-MM bus standard for CPU writing and reading. Avl2Reg module is designed to connect with the CPU system as shown in Figure 2-10.

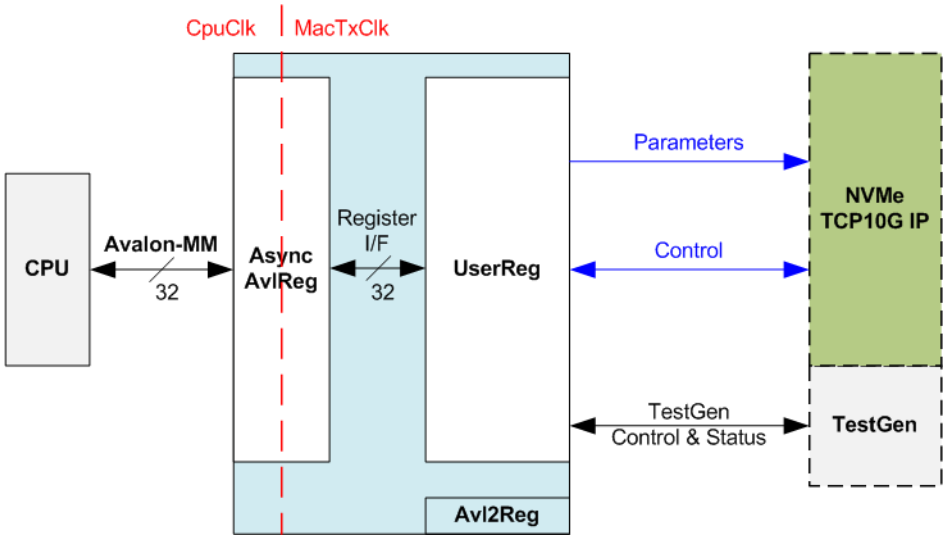


Figure 2-10 CPU and peripherals hardware

Avl2Reg consists of AsyncAvlReg and UserReg. AsyncAvlReg is designed to convert the Avalon-MM signals to be the simple register interface which has 32-bit data bus size, similar to Avalon-MM data bus size. Additionally, AsyncAvlReg includes asynchronous logic to support clock domain crossing between CpuClk and MacTxClk domain.

UserReg includes the register file of the parameters and the status signals of other modules in the test system, i.e., NVMeTCP10G-IP and TestGen. More details of AsyncAvlReg and UserReg are described as follows.

2.3.1 AsyncAvlReg

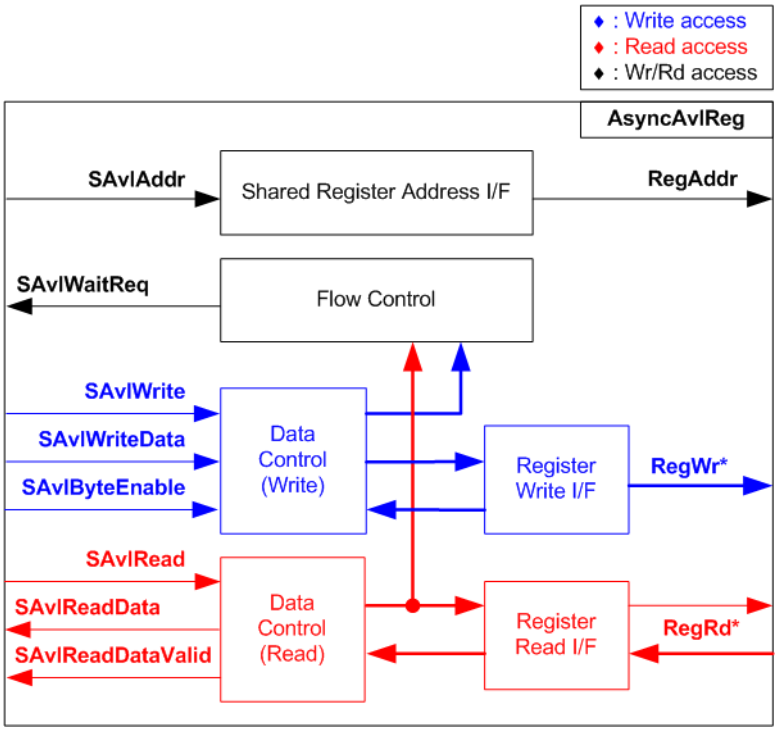


Figure 2-11 AsyncAvlReg Interface

The signal on Avalon-MM bus interface can be split into three groups, i.e., Write channel (blue color), Read channel (red color), and Shared control channel (black color). More details of Avalon-MM interface specification are described in following document.
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

According to Avalon-MM specification, one command (write or read) can be operated at a time. The logics inside AsyncAvlReg are split into three groups, i.e., Write control logic, Read control logic, and Flow control logic. Flow control logic controls SAvlWaitReq to hold the next request from Avalon-MM interface if the current request does not finish. Write control and Write data I/F of Avalon-MM bus are latched and transferred to be Write register interface with clock domain crossing registers. Similarly, Read control I/F are latched and transferred to be Read register interface. After that, the returned data from Register Read I/F is transferred to Avalon-MM bus by using clock domain crossing registers. Address I/F of Avalon-MM is latched and transferred to Address register interface as well.

The simple register interface is compatible with single-port RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq and RdValid signals for controlling read latency time. The address of register interface is shared for write and read transaction, so user cannot write and read the register at the same time. The timing diagram of the register interface is shown in Figure 2-12.

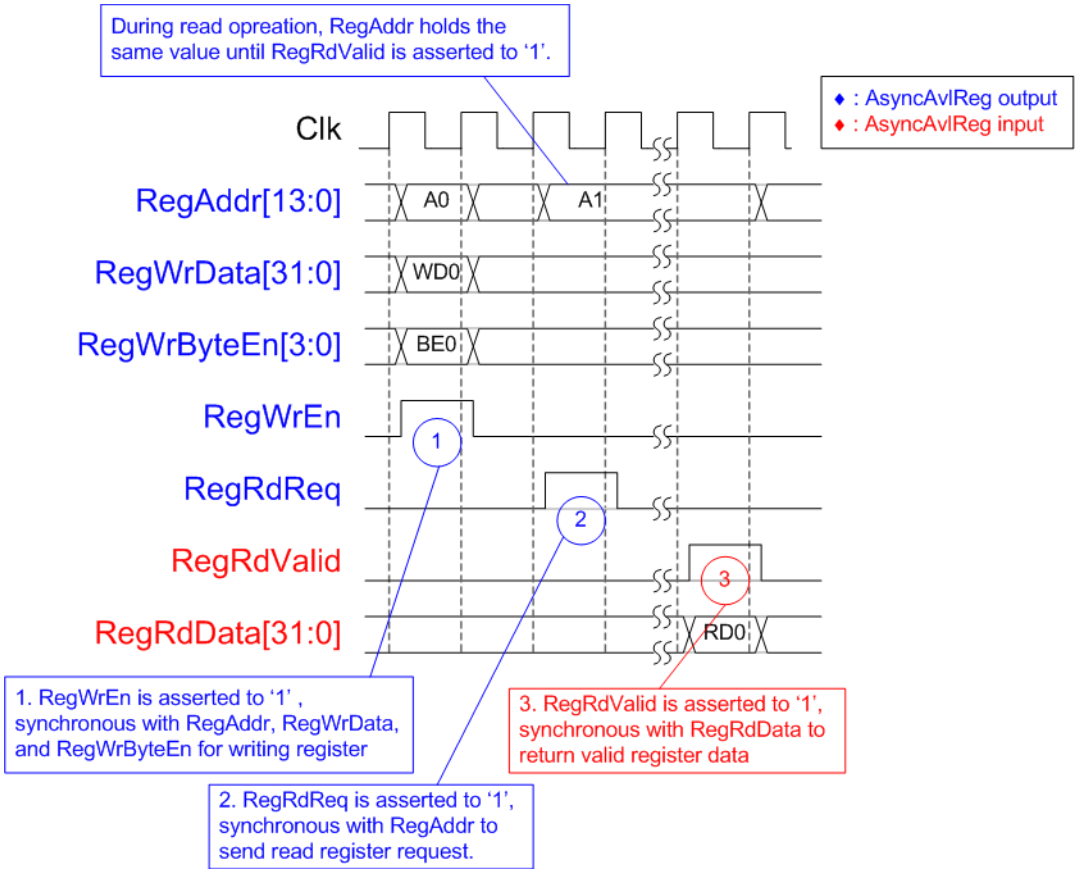


Figure 2-12 Register interface timing diagram

1) To write register, the timing diagram is similar to single-port RAM interface. RegWrEn is asserted to '1' with the valid signal of RegAddr (Register address in 32-bit unit), RegWrData (write data of the register), and RegWrByteEn (the write byte enable). Byte enable has four bits to be the byte data valid. Bit[0], [1], [2], and [3] are equal to '1' when RegWrData[7:0], [15:8], [23:16], and [31:24] are valid respectively
2) To read register, AsyncAvlReg asserts RegRdReq to '1' with the valid value of RegAddr. 32-bit data is returned after receiving the read request. The slave detects RegRdReq asserted to start the read transaction. During read operation, the address value (RegAddr) does not change until RegRdValid is asserted to '1'. Therefore, the address can be used for selecting the returned data by using multiple layers of multiplexer.
3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAvlReg forwards the read value to SAvlRead interface.

2.3.2 UserReg

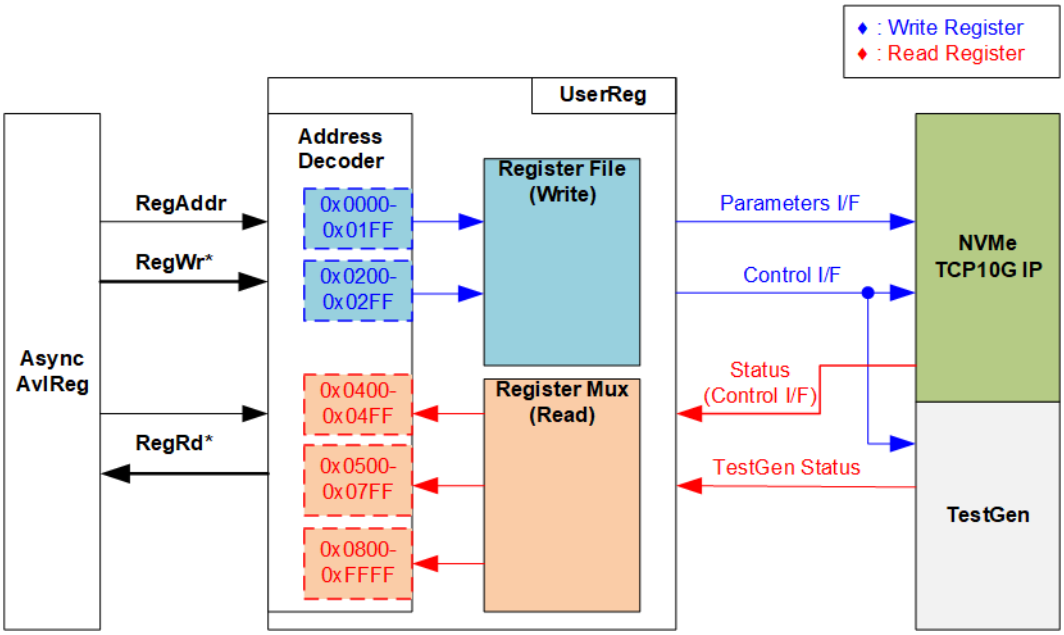


Figure 2-13 UserReg Interface

The address range to map to UserReg is split into five areas, as shown in Figure 2-13.

1) 0x0000 – 0x01FF: mapped to set the parameters of NVMeTCP10G-IP. This area is write access only.
2) 0x0200 – 0x02FF: mapped to set the control signals of NVMeTCP10G-IP and TestGen. This area is write access only.
3) 0x0400 – 0x04FF: mapped to read the status signals of NVMeTCP10G-IP. This area is read access only.
4) 0x0500 – 0x07FF: mapped to read the status signals of TestGen. This area is read access only.
5) 0x0800 – 0xFFFF: mapped to read IP version of NVMeTCP10G-IP. This area is read access only.

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The register file inside UserReg is 32-bit bus size. Therefore, write byte enable (RegWrByteEn) is not applied in the test system and the CPU uses 32-bit pointer to set the hardware register.

To read register, two-step multiplexer is designed to select the read data within each address area. The lower bit of RegAddr is applied in each Register area to select the active data. Next, the address decoder uses the upper bit to select the read data from active area and return to CPU. Totally, the latency of read data is equal to two clock cycles. Therefore, RegRdValid is created by RegRdReq with asserting two D Flip-flops. More details of the address mapping within UserReg module are shown in Table 2-1.

Table 2-1 Register Map

Address Rd/Wr	Register Name (Label in “nvmetcpiptest.c”)	Description
0x0000 – 0x01FF: Parameters of NVMeTCP10G-IP (Write access only)		
BA+0x0000	Host MAC Address (Low) Reg (TCP_HML_INTREG)	[31:0]: Input to be host MAC address. (HostMAC[31:0] of NVMeTCP10G-IP)
BA+0x0004	Host MAC Address (High) Reg (TCP_HMH_INTREG)	[15:0]: Input to be host MAC address. (HostMAC[47:32] of NVMeTCP10G-IP)
BA+0x0008	Host IP Address Reg (TCP_HIP_INTREG)	[31:0]: Input to be host IP address. (HostIPAddr[31:0] of NVMeTCP10G-IP)
BA+0x000C	Host Port Number Reg (TCP_HPN_INTREG)	[15:0]: Input to be host admin port number (HostAdmPort[15:0] of NVMeTCP10G-IP) [31:16]: Input to be host I/O port number (HostIOPort[15:0] of NVMeTCP10G-IP)
BA+0x0010	Target IP Address Reg (TCP_TIP_INTREG)	[31:0]: Input to be target IP address. (TrgIPAddr[31:0] of NVMeTCP10G-IP)
BA+0x0020	TCP Timeout Reg (TCP_TMO_INTREG)	[31:0]: Input to be TCP timeout value (TCPTimeOutSet[31:0] of NVMeTCP10G-IP)
BA+0x0024	NVMe Timeout Reg (NVM_TMO_INTREG)	[31:0]: Input to be NVMe timeout value (NVMeTimeOutSet[31:0] of NVMeTCP10G-IP)
BA+0x0100 - BA+0x010F	Host NQN Word 0-3 Reg (HSTNQNW0-W3_INTREG)	128-bit input to be NVMe Qualifed Name (NQN) of the host (HostNQN [127:0] of NVMeTCP10G-IP) 0x0100: Bit[31:0], 0x0104: Bit[63:32], …, 0x010C:Bit[127:96]
BA+0x0180 - BA+0x018F	Target NQN Word 0-3 Reg (TRGNQNW0-W3_INTREG)	128-bit input to be NVMe Qualifed Name (NQN) of the target (TrgNQN[127:0] of NVMeTCP10G-IP) 0x0180: Bit[31:0], 0x0184: Bit[63:32], …, 0x018C:Bit[127:96]
0x00200 – 0x02FF: Control signals of NVMeTCP10G-IP and TestGen (Write access only)		
BA+0x0200	Connection Enable Reg (CONNEN_INTREG)	[0]: Input to enable the connection with the target (HostConnEn of NVMeTCP10G-IP)
BA+0x0210	User Command Reg (USERCMD_INTREG)	[0]: Input to be command request for writing/reading target SSD ‘0’: Write SSD , ‘1’: Read SSD When this register is written, Write or Read command is generated from TestGen to NVMeTCP10G-IP in HostMM interface.
BA+0x0214	Test Pattern Reg (PATTSEL_INTREG)	[2:0]: Select test pattern. “000”-Increment, “001”-Decrement, “010”-All 0, “011”-All 1, “100”-LFSR.
BA+0x0220	Host MM Address (Low) Reg (HMM_ADRL_INTREG)	[31:0]: Input to be start address (512-byte unit) of SSD Writing/Reading in HostMM interface.
BA+0x0224	Host MM Address (High) Reg (HMM_ADRH_INTREG)	[15:0]: Input to be start address (512-byte unit) of SSD Writing/Reading in HostMM interface.
BA+0x0228	Host MM Length (Low) Reg (HMM_LENL_INTREG)	[31:0]: Input to be transfer length (512-byte unit) of SSD Writing/Reading in HostMM interface.
BA+0x022C	Host MM Length (High) Reg (HMM_LENH_INTREG)	[15:0]: Input to be transfer length (512-byte unit) of SSD Writing/Reading in HostMM interface.

Address Rd/Wr	Register Name (Label in “nvmetcpiptest.c”)	Description
0x0400 – 0x04FF: Status signals of NVMeTCP10G-IP (Read access only)		
BA+0x0400 Rd	Host Status Reg (HSTS_INTREG)	[0]: Mapped to linkup of 10G EMAC IP. ‘0’: Link is down, ‘1’: Link is up. [1]: Mapped to HostConnStatus of NVMeTCP10G-IP. ‘0’: NVMe/TCP connection off, ‘1’: NVMe/TCP connection on. [2]: Mapped to HostBusy of NVMeTCP10G-IP. ‘0’: IP is Idle, ‘1’: IP is busy. [3]: Mapped to HostError of NVMeTCP10G-IP. ‘0’: No error, ‘1’: Error is found. [4]: Mapped to PattBusy of TestGen. ‘0’: Test is Idle, ‘1’: Test is running. [5]: Mapped to PattFail of TestGen which is fail flag of Read data verification from TestGen. ‘0’: No error, ‘1’: Verification Error is found.
BA+0x0410 Rd	Total disk size (Low) Reg (LBASIZEL_INTREG)	[31:0]: Mapped to TrgLBASize[31:0] of NVMeTCP10G-IP.
BA+0x0414 Rd	Total disk size (High) Reg (LBASIZEH_INTREG)	[15:0]: Mapped to TrgLBASize[47:32] of NVMeTCP10G-IP.
BA+0x0420 Rd	Capability (Low) Status Reg (CAPSTSL_INTREG)	[31:0]: Mapped to TrgCAPStatus[31:0] of NVMeTCP10G-IP.
BA+0x0424 Rd	Capability (High) Status Reg (CAPSTSL_INTREG)	[31:0]: Mapped to TrgCAPStatus[47:32] of NVMeTCP10G-IP.
BA+0x0430 Rd	Host Error Type Reg (HERRTYPE_INTREG)	[31:0]: Mapped to HostErrorType[31:0] of NVMeTCP10G-IP to show error status.
BA+0x0440 Rd	NVMe Completion Status Reg (NVMCOMPSTS_INTREG)	[15:0]: Mapped to TrgAdmStatus[15:0] of NVMeTCP10G-IP. [31:16]: Mapped to TrgIOStatus[15:0] of NVMeTCP10G-IP.
BA+0x0450 - BA+0x045F Rd	Test pin Word 0-3 Reg (NVMTESTPINW0-W3_INTREG)	128-bit signal mapped to TestPin[127:0] of NVMeTCP10G-IP. 0x0450: Bit[31:0], 0x0454: Bit[63:32], …, 0x045C:Bit[127:96]
0x0500 – 0x07FF: Status signals of TestGen (Read access only)		
BA+0x0500 Rd	Expected Pattern Word0 Reg (EXPPAT0_INTREG)	[31:0]: Mapped to bit[31:0] of the expected data at the 1st failure data in Read data verification of TestGen.
BA+0x0504 Rd	Expected Pattern Word1 Reg (EXPPAT1_INTREG)	[31:0]: Mapped to bit[63:32] of the expected data at the 1st failure data in Read data verification of TestGen.
BA+0x0540 Rd	Read Pattern Word0 Reg (RDPAT0_INTREG)	[31:0]: Mapped to bit[31:0] of the read data at the 1st failure data in Read data verification of TestGen.
BA+0x0544 Rd	Read Pattern Word1 Reg (RDPAT1_INTREG)	[31:0]: Mapped to bit[63:32] of the read data at the 1st failure data in Read command of TestGen.
BA+0x0580 Rd	Failure Address (Low) Reg (RDFAILNOL_INTREG)	[31:0]: Mapped to bit[31:0] of the byte address of the 1st failure data in Read data verification of TestGen.
BA+0x0584 Rd	Failure Address (High) Reg (RDFAILNOH_INTREG)	[24:0]: Mapped to bit [56:32] of the byte address of the 1st failure data in Read data verification of TestGen.
BA+0x0590 Rd	Current Test Byte (Low) Reg (CURSIZEL_INTREG)	[31:0]: Mapped to bit[31:0] of the current test data size in TestGen to show the byte amount of successful test data (without error).
BA+0x0594 Rd	Current Test Byte (High) Reg (CURSIZEH_INTREG)	[24:0]: Mapped to bit[56:32] of the current test data size in TestGen to show the byte amount of successful test data (without error).
0x0800 – 0xFFFF: Other interfaces		
BA+0x0800 Rd	IP Version REG (IPVERSION_INTREG)	[31:0]: Mapped to IPVersion[31:0] of NVMeTCP10G-IP
BA+0x0804 Rd	EMAC Version REG (EMACVERSION_INTREG)	[31:0]: Mapped to IPVersion[31:0] of DG 10GEMAC-IP

3 CPU Firmware

3.1 Test firmware (nvmetcpiptest.c)

After system boot-up, CPU initializes JTAG UART and Timer parameters. Next, 10G Ethernet link up status (HSTS_INTREG[0]) is polling. The CPU waits until the ethernet link is established. After that, the main menu is displayed on the console. There are five test operations for user selection, i.e., Set network parameter, Connect, Write command, Read command, and Disconnect. User must select the proper menu following the IP sequential processes as below.

1) User sets network parameter as the first action.
2) User selects Connect as the second action to connect with target SSD.
3) After Connect operation is done, user can write or read the target SSD by Write command or Read command.
4) When user wants to disconnect with the target SSD (disconnect NVMeTCP10G-IP from the target), user selects Disconnect.
5) After Disconnect operation is done, user can go back to set network parameter again (1) or connect with the target again by Connect (2).
   Note: When user connects host with the same target again (2), setting network parameter (1) can be omitted.

In main menu, only proper menu is displayed. If user selects a menu which is unavailable on the console, no action will occur.

3.1.1 Set network parameter

This menu is used to set the network parameters for NVMeTCP10G-IP initialization, i.e., host MAC address, host IP address, host port number, and target IP address. Also, NVMe Qualified Name (NQN) of the host and the target are set in this menu. The sequence of parameter setting is as follows.

1) Display current parameter values on the console. If the parameters have never been set, the default values (assigned in the testing firmware) are shown.
2) Ask user to skip (confirm the current parameter values) or set the desired values.
   (a) Press ‘x’ on keyboard to skip.
   (b) Press other keys to start parameter value setting.
3) When pressed key is not ‘x’, user is asked to input the desired parameter values, i.e., target NQN, host MAC address, host IP address, host port numbers (Admin and I/O), and target IP address respectively. If the input is invalid, the parameter is set by its latest value.
4) After parameter values are set (or ‘x’ key is pressed), CPU assigns them to parameter registers (HSTNQNW0-3_INTREG, TRGNQNW0-3_INTREG, TCP_HML_INTREG, TCP_HMH_INTREG, TCP_HIP_INTREG, TCP_HPN_INTREG, TCP_TIP_INTREG, TCP_TMO_INTREG, and NVM_TMO_INTREG).
   Note: Timeout value of TCP, Timeout value of NVMe, and host NQN are always set as default values by the firmware.
   - *TCP timeout value is 1 sec.*
   - *NVMe timeout value is 4 sec.*
   - *NQN of the host value is “dgnvmehtest”*

3.1.2 Connect

This menu is used create the connection between NVMeTCP25G-IP (the host) with the target system before writing or reading the target SSD. The sequence of Connect is as follows.

1) Set connection enable (CONNEN_INTREG) to '1'.
2) CPU waits until the connection is created completely via monitoring host connection status flag (HSTS_INTREG[1]='1'). If some errors are found (HSTS_INTREG[3]='1'), the process stops with displaying the error message.
3) After that, the target SSD capacity (LBASIZEL/H_INTREG) in GB unit is displayed.

3.1.3 Write/ Read command

These menus are used to test NVMeTCP10G-IP by writing or reading the target SSD. The sequence of the commands is as follows.

1) Receive start address, transfer length, and test pattern from user. If some inputs are invalid, the operation is cancelled.
   Note: A Write or Read command data size is fixed to 4 Kbytes. So, start address and transfer length which are 512-byte unit must be aligned to 8.
2) Get all inputs and set to HMM_ADRL/H_INTREG, HMM_LENL/H_INTREG, and PATTSEL_INTREG.
3) Set USRCMD_INTREG[0] according to command type ('0' for Write command, '1' for Read command). After that, Test logic (TestGen) generates the command in HostMM I/F to NVMeTCP10G-IP. Test busy flag (HSTS_INTREG[4]) changes from '0' to '1' and NVMeTCP10G-IP starts operating the command.
4) CPU waits until the operation is completed or some errors (except verification error) are found by monitoring HSTS_INTREG[5:3].

   Bit[3] is asserted to'1' when error is detected. After that, error message is displayed on the console to show the error details. Finally, the process is hanged up.
   Bit[4] is de-asserted to '0' when the command operation in the test system is done which means both NVMeTCP10G-IP and TestGen return back to be idle.
   Bit[5] is asserted to '1' when data verification is failed. Then, the verification error message is displayed. CPU is still running until the operation is done or user inputs any key to cancel operation.

   During running command, current amount of transferred data (read from CURSIZEL/H_INTREG) is displayed every second.

5) After test busy flag (HSTS_INTREG[4]) is de-asserted to '0', CPU displays the test result on the console, i.e., total time usage, total transfer size, and transfer speed.

3.1.4 Disconnect

This menu is used to terminate TCP/IP and NVMe/TCP connections between the host and the target which are built by Connect command. The sequence of Disconnect is as follows.

1) Reset connection enable (CONNEN_INTREG) to '0'.
2) CPU waits until the disconnecting process is completed via monitoring host connection status flag (HSTS_INTREG[1]='0'). If some errors are found (HSTS_INTREG[3]='1'), the process stops with displaying the error message.
3) After disconnect is done, user can re-connect the host with the same target again via Connect command. If the target is changed, "Set Network Parameter" must be selected to set the new parameters for the new target.

3.2 Function list in Test firmware

int get_param(userin_struct* userin)	
Parameters	userin: Three inputs from user, i.e., start address, total transfer length in 512-byte unit, and test pattern
Return value	0: Valid input, -1: Invalid input
Description	Receive the input parameters from the user and verify the value. When the input is invalid, the function returns -1. Otherwise, all inputs are updated to userin parameter.

void get_string(unsigned char* out_val)	
Parameters	out_val: Character array which stores 16 bytes of NQN
Return value	None
Description	Receive 16 characters of NQN from user. When user presses “Enter”, the input NQN is stored in output character array. If the input is less than 16 characters, the upper characters are filled as ”00h” (Null).

void proc_connect(void)	
Parameters	None
Return value	None
Description	Establish the connection with the target SSD, following topic 3.1.2 (Connect)

void proc_disconnect(void)	
Parameters	None
Return value	None
Description	Terminate the connection with the target SSD, following topic 3.1.4 (Disconnect)

int set_netparam(void)	
Parameters	None
Return value	0: No error, -1: Out of range input
Description	Set network parameters, following topic 3.1.1 (Set network parameter).

void set_param(void)	
Parameters	None
Return value	None
Description	Set parameters from global parameters to the registers.

void show_error(void)	
Parameters	None
Return value	None
Description	Read HERRTYPE_INTREG, decode the error flag, and display error message following the error flag.

void show_param(void)	
Parameters	None
Return value	None
Description	Display the current value of the parameters, read from global parameters such as NQN, IP address, MAC address, and port number.

void show_result(void)	
Parameters	None
Return value	None
Description	Print total size by calling get_cursize and show_size function. After that, calculate total time usage from global parameters (timer_val and timer_upper_val) and display in usec, msec, or sec unit. Finally, transfer performance is calculated and displayed in MB/s unit.

void show_size(unsigned long long size_input)	
Parameters	size_input: transfer size to display on the console
Return value	None
Description	Calculate and display the input value in MByte, GByte or TByte unit

void show_testpin(void)	
Parameters	None
Return value	None
Description	Read NVMTESTPINW0-W3_INTREG to display IP test pin on the console for debugging.

void show_vererr(void)	
Parameters	None
Return value	None
Description	Read RDFAILNOL/H_INTREG (error byte address), EXPPAT0-1_INTREG (expected value), and RDPAT0-1_INTREG (read value) to display verification error details on the console.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read HSTS_INTREG[0] and wait until Ethernet link is established.

int wrrd_dev(unsigned int user_cmd)	
Parameters	user_cmd: 0-Write command, 1-Read command,
Return value	0: No error, -1: Receive invalid input or some errors are found.
Description	Run Write or Read command, following topic 3.1.3 (Write/Read command)

4 Example Test Result

The demo uses NVMeTCP10G-IP on Arria 10 GX board as the host. A 512 GB Samsung 960 Pro is plugged in target PC installing LinuxOS with kernel version 5.4.0-81. The example test result is shown in Figure 4-1.

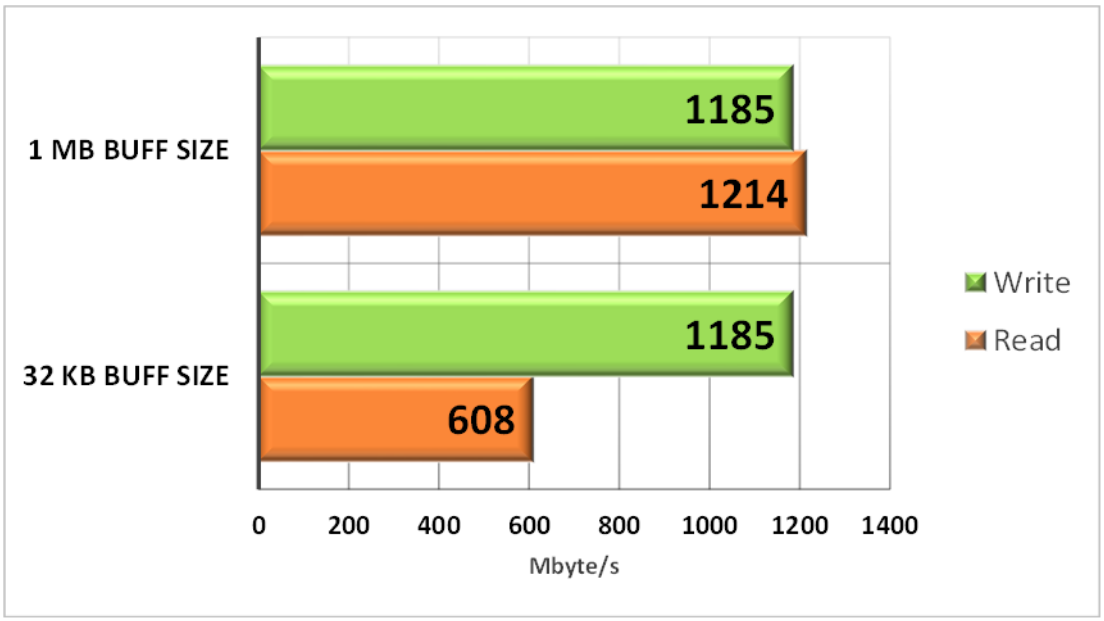


Figure 4-1 Test Performance of NVMeTCP10G-IP demo

Write performance is about 1100 - 1200 Mbyte/sec for every buffer size. While read performance depends on read buffer size referred from NVMeTCP10G-IP specification. The maximum read performance is about 1200 Mbyte/sec with 1-Mbyte read buffer. The minimum read performance is about 600 Mbyte/sec with 32-Kbyte read buffer. The performance results may be unstable affected from resources of the target side. The results shown in Figure 4-1 are the best performance values in our test environment.

5 Revision History

Revision	Date	Description
1.0	25-Mar-22	Initial Release