raNVMe-IP data stream reference design manual

Rev1.1 21-Aug-23

raNVMe-IP data stream reference design manual

Rev1.1 21-Aug-23

1 Overview

raNVMe-IP is the NVMe host controller IP for writing and reading the data with NVMe SSD to support the application which requires the random data access at high-speed rate. Transfer size of Write/Read command is fixed to 4 Kbyte and 32 Write or Read commands can be sent at the same time. As shown in the left side of Figure 1-1, the main control signals are 48-bit address and the valid pulse of the address.

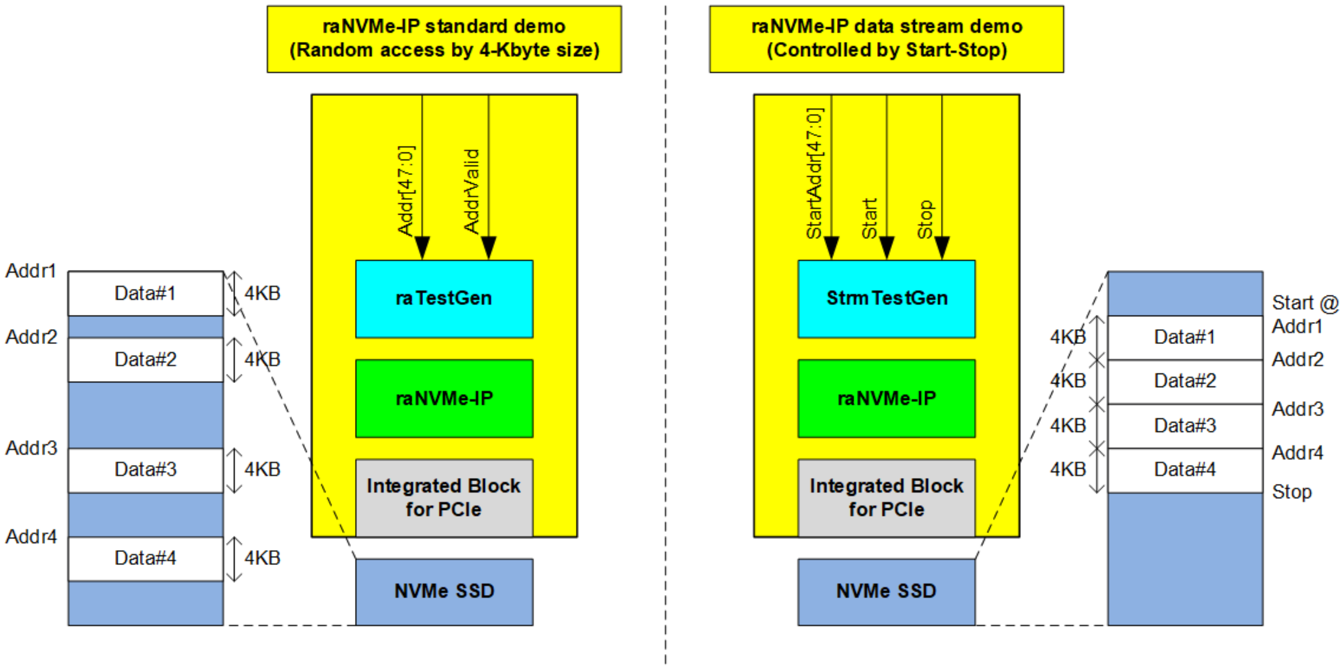

Figure 1-1 Standard demo and data stream demo comparison

Though raNVMe-IP is designed to support random access, the IP can transfer the data to NVMe SSD by using sequential access with the high performance as shown in the right side of Figure 1-1. There are some applications which needs to record data stream such as video data from the camera to NVMe SSD, but the receiver does not know the total data size. The control signals of the system are start flag to start data recording with the start address and stop flag to stop data recording.

raNVMe-IP can be designed to be Start/Stop control system by splitting the data stream input to 4KB unit size and then stores to NVMe SSD as sequential access. If the last data is not aligned to 4 KB size, the dummy data is filled and then sent the last command to complete the stop operation. Using sequential access can achieve the high performance to write and read the data with NVMe SSD.

2 Hardware overview

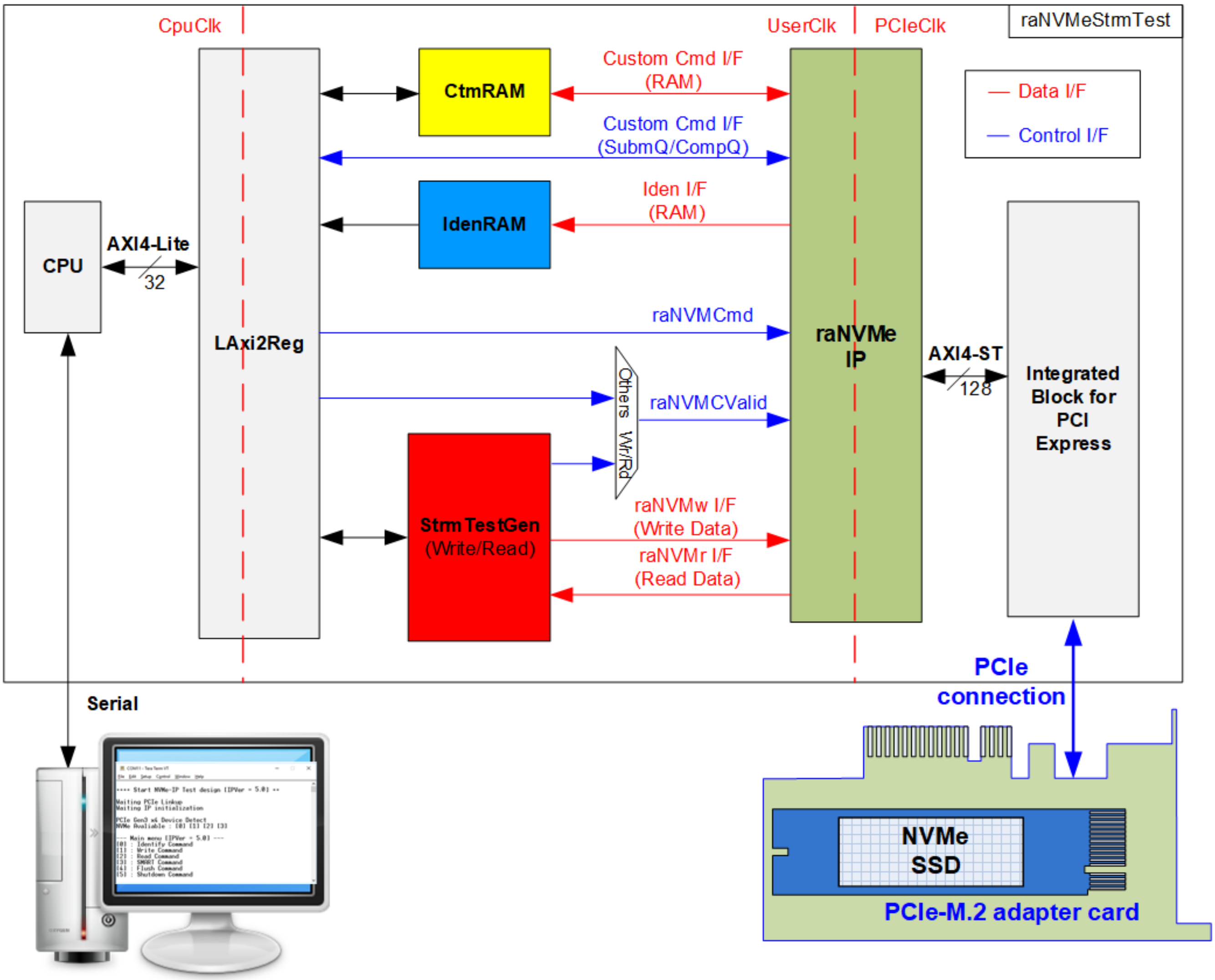

Figure 2-1 raNVMe-IP data stream demo hardware

In the standard raNVMe-IP demo, the test module (raTestGen) which is designed to send the command request signal (raNVMCValid) for Write or Read command (Multiple mode) and transfer the data with raNVMe-IP. In raNVMe-IP data stream, this module is replaced by StrmTestGen module. While other modules in NVMe function (CtmRAM, IdenRAM, raNVMe-IP and PCIe block) and CPU system (CPU and LAxi2Reg) are similar.

According to raNVMe-IP specification, the command in Single mode (Identify, Shutdown, Flush or SMART) can operate one command at a time. These single-mode commands are controlled by CPU firmware and the data is transferred to CtmRAM (SMART command) and IdenRAM (Identify command) for decoded by CPU firmware.

StrmTestGen module sends the 4KB address and transfer the data as sequential access. In Write command, when 4KB write data is ready and raNVMe-IP command queue is not full, the new Write command with the next 4KB address is sent. In Read command, when raNVMe-IP command queue is not full, the new Read command for reading the next 4KB data is sent. The new command is requested to raNVMe-IP after the user asserts start flag and it is run until the user asserts stop flag.

CPU and LAxi2Reg are designed to interface with user via Serial interface. The user can set command with the parameters on Serial console. Also, the current status of the test hardware can be displayed on Serial console for monitoring the test progress and test result.

There are three clock domains displayed in Figure 2-1, i.e., CpuClk, UserClk and PCIeClk. CpuClk is the clock domain of CPU and its peripherals. This clock must be stable clock which is independent from the other hardware interface. UserClk is the example user clock domain which may be independent clock for running the user interface of raNVMe-IP. According to raNVMe-IP datasheet, clock frequency of UserClk must be more than or equal to PCIeClk. So, this reference design uses 275/280 MHz. PCIeClk is the clock output from PCIe hard IP to synchronous with data stream of 128-bit AXI4 stream bus. When the PCIe hard IP is set to 4-lane PCIe Gen3, PCIeClk frequency is equal to 250 MHz.

More details of the hardware are described as follows.

2.1 StrmTestGen

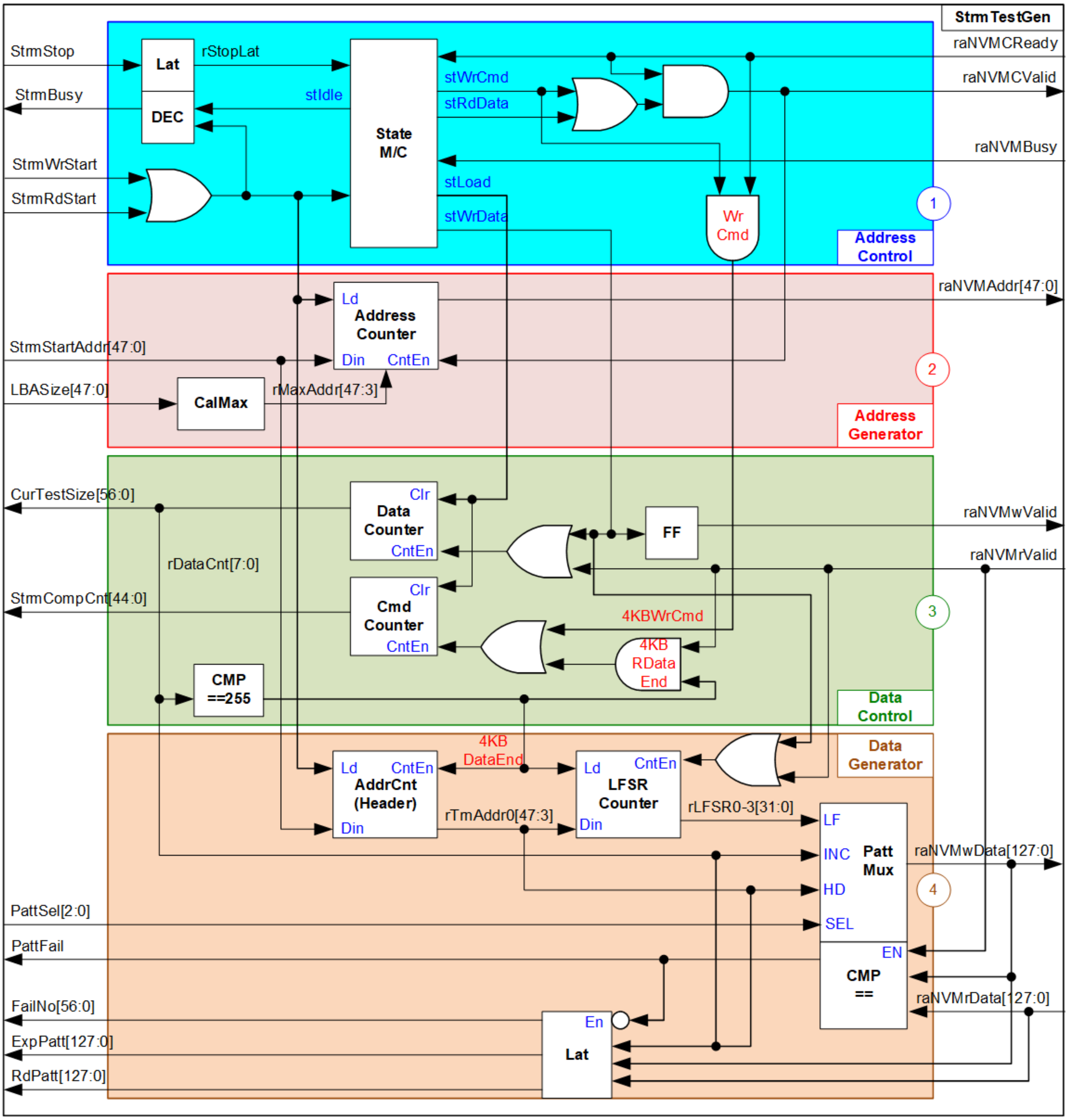

Figure 2-2 StrmTestGen Interface

StrmTestGen is the module to generate command request (raNVMCValid) and the address (raNVMAddr) when user asserts start flag for Write or Read command. This command request is continuously generated until user asserts stop flag. Also, test pattern for sending or verifying in Write or Read command with the data flow control signal are created in StrmTestGen. As shown in Figure 2-2, StrmTestGen logic design can be divided into four groups, i.e., Address Control, Address Generator, Data Control and Data Generator.

When running Write command, 4KB data or 256 cycles of 128-bit data is created by StrmTestGen before asserting the Write command request to raNVMe-IP. While the Read command request is sent to raNVMe-IP and then 4KB data is returned to StrmTestGen. So, the sequence of the address and the data of Write and Read command is inverted.

Address Control

The command request for sending the address in Write or Read command (raNVMCValid) is controlled by the state machine which consists of seven states, described as follows.

(1) stIdle: This state is designed to wait the start flag of Write or Read command from the user. It changes to stLoad when the start flag is asserted.
(2) stLoad: This state is applied to clear the signals such as data counter which holds the result of the previous test. Also, it is applied to load the start address from the user for the first Write or Read command. After that, the state changes to stWrWait for Write command or stRdData for Read command.
(3) stWrWait: This state is designed to wait until raNVMe-IP is ready to receive the data of the next Write command. After that, the state changes to stWrData to start transferring 4KB data. Otherwise, if the user asserts stop flag, the state changes to stStop to finish the Write operation.
(4) stWrData: This state is stayed for 256 cycles to send 4KB Write data to raNVMe-IP. Next, it changes to stWrCmd to send the Write command request.
(5) stWrCmd: This state is designed to assert the write command request (raNVMCValid) and it changes to stWrWait after raNVMe-IP accepts the request by asserting raNVMCReady to ‘1’.
(6) stRdData: This state is designed to wait stop flag asserted from the user when running Read command.
(7) stStop: This state is designed to wait until all commands within raNVMe-IP are completely operated by monitoring raNVMBusy signal. After raNVMBusy is de-asserted to ‘0’, the state goes to stIdle.

As shown in Block (1), raNVMCValid is asserted when running in stWrWait and stRdData when operating Write and Read command respectively. It is de-asserted to ‘0’ if raNVMe-IP is not ready to receive the new request, raNVMCReady=’0’. StrmBusy is designed for the user checking if StrmTestGen completes the operation. StrmBusy is de-asserted to ‘0’ when state machine is in stIdle state.

Address Generator

The address sent to raNVMe-IP, raNVMAddr, is valid when raNVMCValid is asserted to ‘1’. In data stream reference design, the address is designed to be up-counter to store the data as sequential format. The start value can be set from the user via StrmStartAddr. When running the command for long time until the address is equal to the end address of the SSD, rMaxAddr, the address will be reset to 0 for storing the next data at the beginning address of the SSD. The maximum address is calculated from the total device capacity returned from raNVMe-IP.

Data Control

The data interface of raNVMe-IP uses valid signal to be control signal. The write data, raNVMwData, is valid when raNVMwValid is asserted to ‘1’. Similarly, the read data, raNVMrData, is valid when raNVMrValid is asserted to ‘1’.

When operating Write command, raNVMwValid is asserted to ‘1’ for 256 cycles when state is equal to stWrData. While raNVMrValid is asserted by raNVMe-IP for 256 cycles after raNVMe-IP receives the Read command request. The valid signal of Write data and Read data are fed to be the counter enable to show the current data size, CurTestSize, to the user. Moreover, there is StrmCompCnt which is designed to show total count of complete command. The command counter is increased when Write command is asserted or 4KB data is received.

Data Generator

Test data is created to be Write data, raNVMwData, when running Write command or expected data for comparing with received data when running Read command. The data for one Write/Read command is 4 KB which consists of 64-bit header data and the test pattern, selected by PattSel.

	DW0	DW1		DW2	DW3
0x000-0x007	Ad512B [31:0]	0000	Ad512B [47:32]	PattD2[31:0]	PattD3[31:0]
0x008-0x010	PattD4[31:0]	PattD5[31:0]		PattD6[31:0]	PattD7[31:0]
⋮					
0xFF8-0xFFF	PattD1020[31:0]	PattD1021[31:0]		PattD1022[31:0]	PattD1023[31:0]

Figure 2-3 Test pattern format of 4096-byte data for Increment/Decrement/LFSR pattern

As shown in Figure 2-3, 4KB data consists of 64-bit header in DW#0 and DW#1 and the test data in DW#2 – DW#1023. 64-bit header uses the value of a physical address of SSD which stores the data. AddrCnt is designed to be up-counter, similar to Address Counter, but the value is increased after finishing transferring each 4KB data. While the remaining data is the test pattern which may be 32-bit incremental data, 32-bit decremental data or 32-bit LFSR counter. The 32-bit incremental data is designed by combining current address, rTrnAddr0, with the lower bit of data counter, rDataCnt[7:0]. The decremental data is designed by using NOT logic to the incremental data. The equation of 32-bit LFSR data is x^31 + x^21 + x + 1. Four 32-bit LFSR data must be generated within one clock to create 128-bit data. Therefore, the logic uses look-ahead style to generate four LFSR data in the same clock.

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 has 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.

When running Read command, PattFail is asserted to ‘1’ if the received data, raNVMrData, is not equal to the expected data. Also, the signals to show information of the first failure data, i.e., failure data postion (FailNo), expected data (ExpPatt) and received data (RdPatt) are latched to Flip Flop for user reading.

Figure 2-4 shows timing diagram of StrmTestGen when starting Write command.

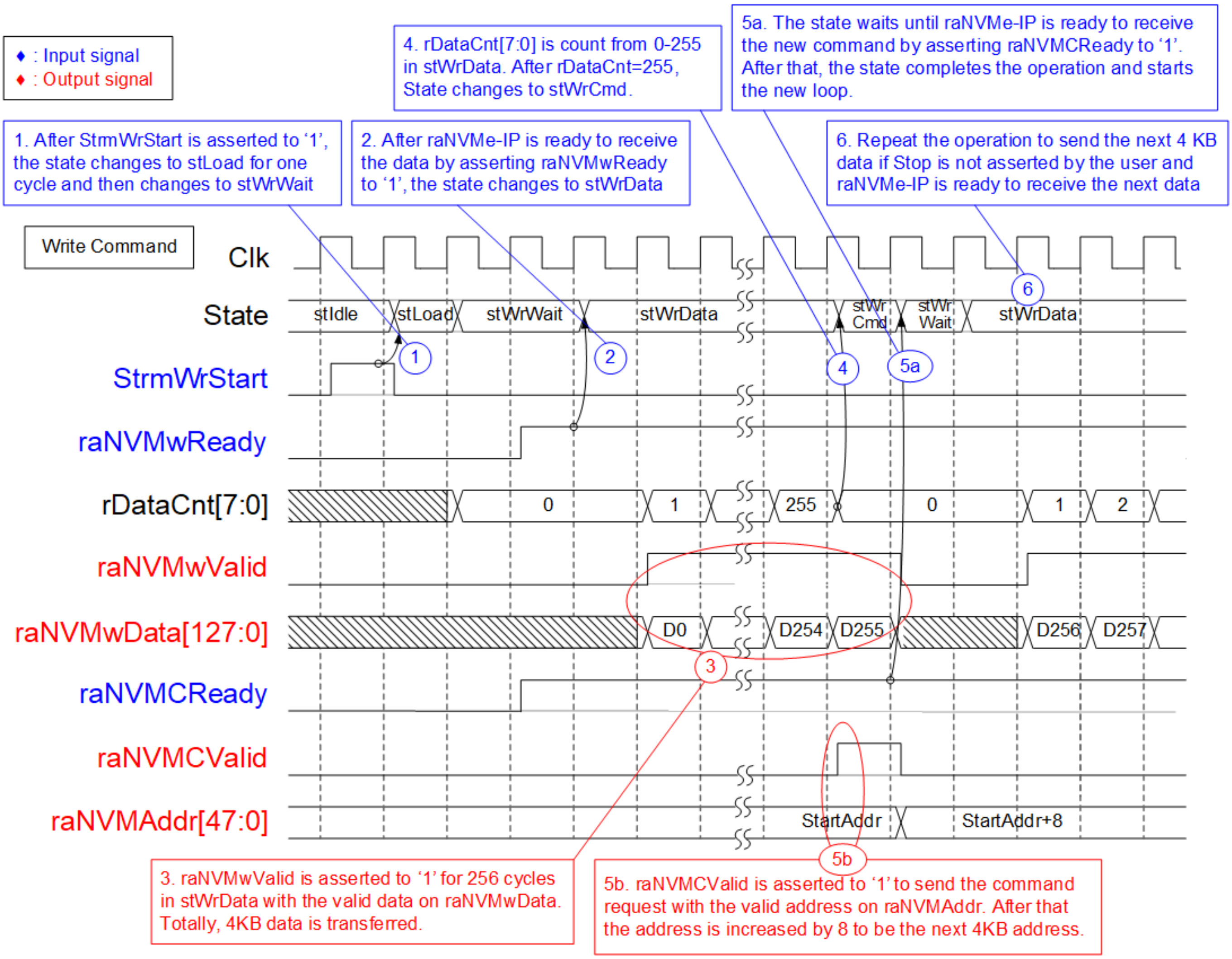

Figure 2-4 Timing diagram of StrmTestGen when running Write command

(1) When the user asserts StrmWrStart to start Write command, the state changes to StLoad. StLoad is run for one cycle only for initializing the internal signals for Write operation. After that, the state changes to stWrWait.

(2) In stWrWait, it is designed to check if raNVMe-IP is ready for receiving the write data. If raNVMwReady is asserted to '1', the state changes to the next state, stWrData.

(3) In stWrData, it is designed to run for 256 cycles to generate 4KB data to raNVMe-IP by asserting raNVMwValid to '1' with the valid data on raNVMwData. Also, rDataCnt is increased every cycle in this state.

(4) When rDataCnt[7:0] is equal to 255, the state changes to stWrCmd.

(5) In stWrCmd, it confirms raNVMe-IP is ready to receive new command. If raNVMCReady is asserted to '1', raNVMCValid is asserted with the valid address on raNVMAddr. The address is increased by 8 which is the next 4KB address in the next clock. Also, the state changes to stWrWait for the next loop run.

Note: In raNVMe-IP data stream demo, 4KB data is sent before sending one Write command. If 4KB data can be sent to raNVMe-IP completely, raNVMe-IP must be ready to receive one Write command. Therefore, stWrCmd is always run for one cycle in this demo.

(6) If the user does not asserts StrmStop and raNVMe-IP is ready to receive new data, the state changes to stWrData for transferring the data of the next command.

StrmStop is asserted to '1' for one cycle, so rStopLat is designed to be latch register of StrmStop which is asserted to '1' until the stop operation is started. The state detects rStopLat asserted when running in stWrWait and then changes to stStop, as shown in Figure 2-5.

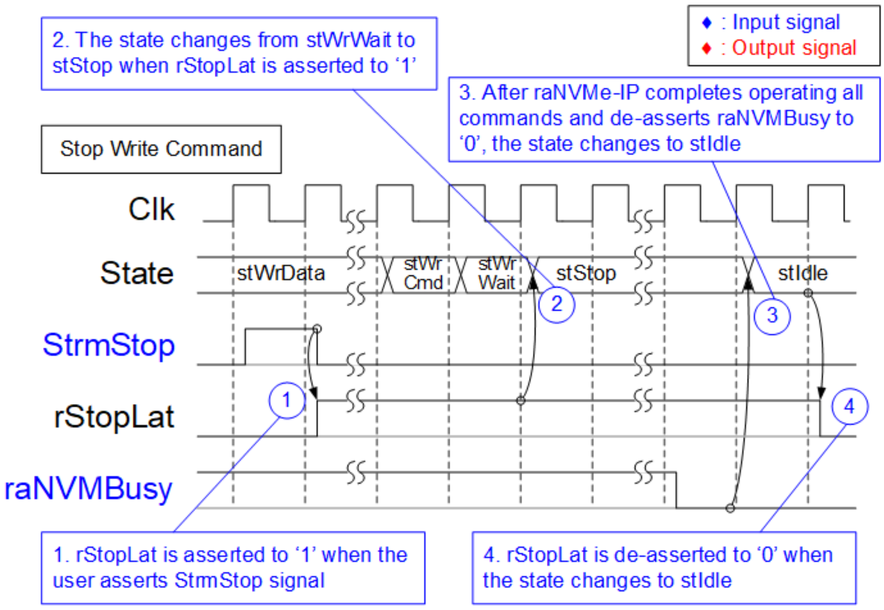

Figure 2-5 Timing diagram of StrmTestGen when stopping Write command

In stStop, StrmTestGen does not send the additional data or the command request. It waits until raNVMe-IP completes all commands in the queue. raNVMBusy is de-asserted to '0' by raNVMe-IP after finishing all Write commands. Finally, the state changes to stIdle to wait the new start flag from the user.

Figure 2-6 shows timing diagram of StrmTestGen when starting Read command.

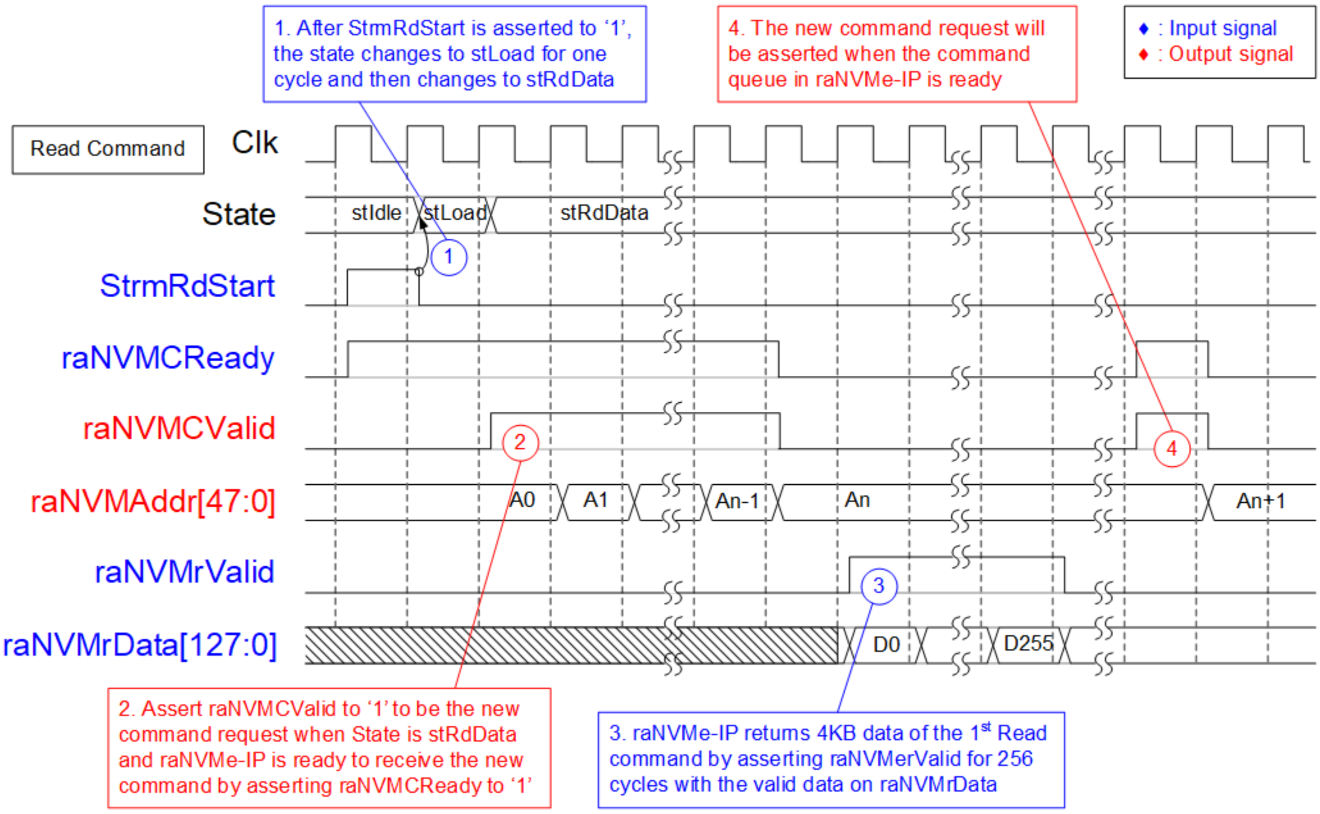

Figure 2-6 Timing diagram of StrmTestGen when running Read command

(1) Similar to Write command, when the user asserts StrmRdStart to start Read command, the state changes to StLoad. StLoad is run for one cycle only for initializing the internal signals for Read operation. After that, the state changes to stRdData.
(2) In stRdData, raNVMCValid is asserted to '1' when raNVMe-IP is ready to receive the new command by asserting raNVMCReady to '1'. Multiple read commands are sent if raNVMCReady is asserted to '1' for several clocks. It will stop sending the command when the command queue in raNVMe-IP is full. The address of the command is increased by 8 which is the next 4 KB address for reading the data as sequential access.
(3) 4KB data of each Read command is returned from raNVMe-IP. The data path is transferred independently with the command request.
(4) If the command queue in raNVMe-IP is not full (raNVMCReady='1'), the new command request is sent to raNMVe-IP.

Similar to stop operation of Write command, rStopLat which is the latch register of StrmStop is read in stRdData. The state changes to stStop if the stop flag is detected, as shown in Figure 2-7.

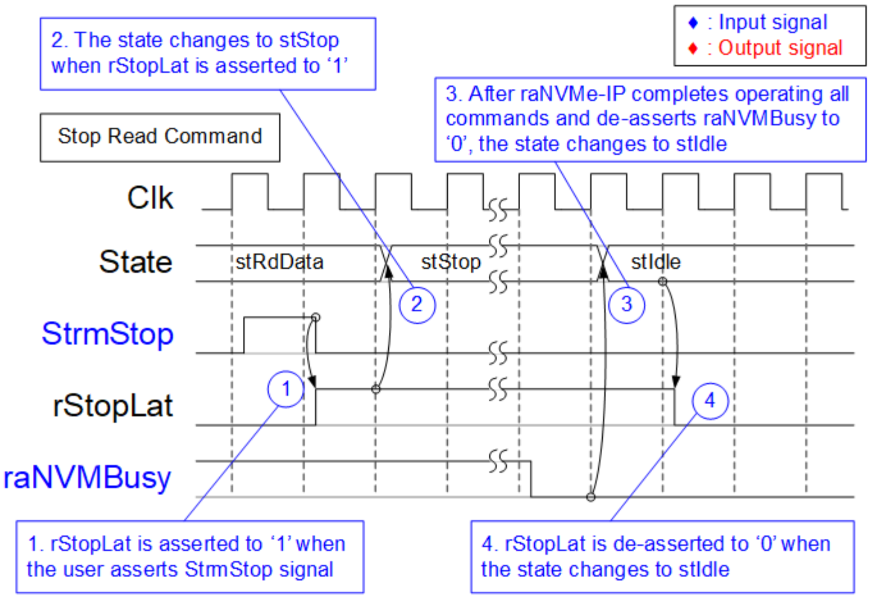

Figure 2-7 Timing diagram of StrmTestGen when stopping Read command

In stStop, there is no additional request sent to raNVMe-IP. It needs to wait until raNVMe-IP returns 4KB data of all commands which is sent in stRdData. After finishing transferring all data, raNVMBusy is de-asserted to '0'. Finally, the state changes to stIdle to wait the new start flag from the user.

2.2 NVMe

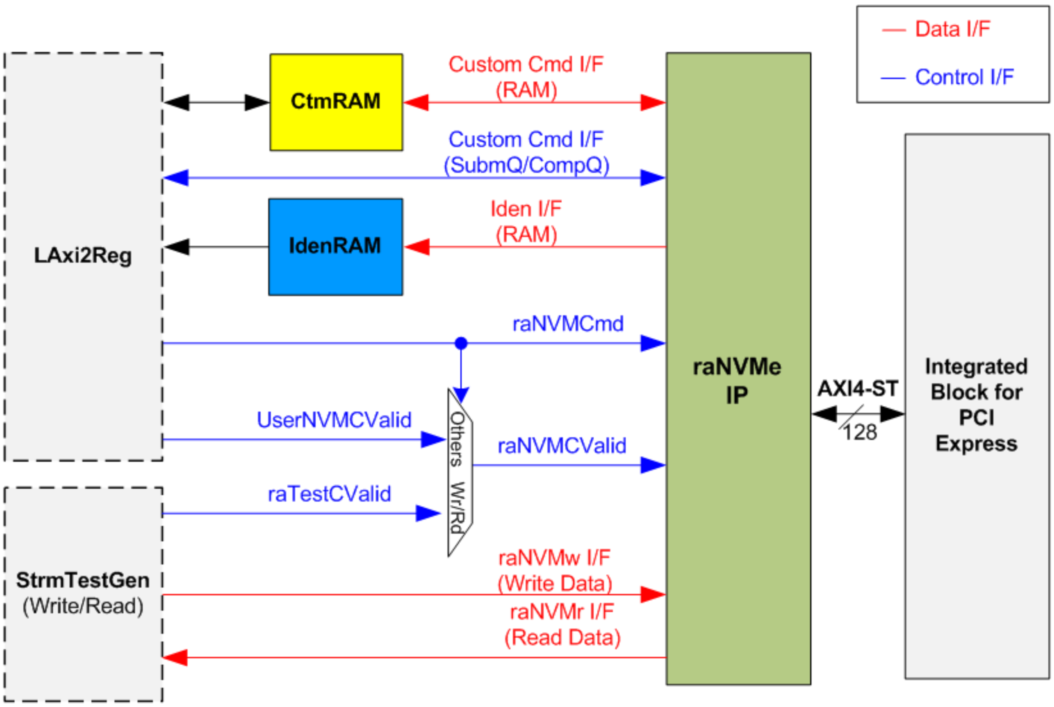

Figure 2-8 NVMe hardware

Figure 2-8 shows the example to interface raNVMe-IP in the reference design. The user interface of raNVMe-IP consists of control interface and data interface. The control interface receives command and the parameters from the user while data interface transfers the data when the command needs data transferring.

There are two types of the command: Single command and Multiple command. The command (raNVMCmd) is set by CPU firmware via LAxi2Reg, but the command request (raNVMCValid) is controlled by two sources. When the command is Single mode, the command request is created by CPU firmware (UserNVMCValid). When the command is Multiple mode, the command request is created by StrmTestGen (raTestCValid). SMART command and Flush command are the custom command which needs to set additional parameters via Custom Cmd I/F. In the test design, these parameters are set by CPU firmware via LAxi2Reg module.

There are four commands which has data transferring. Each command transfers data via its own interface.

- Custom Cmd I/F (RAM) sends SMART data to CtmRAM when running SMART command.
- Iden I/F (RAM) sends Identify data to IdenRAM when running Identify command.
- raNVMw I/F sends Write data from StrmTestGen when running Write command.
- raNVMr I/F sends Read data from raNVMe-IP when running Read command.

Though each command uses the different interface for transferring the data, every data interface has the same data bus size, 128-bit data.

2.2.1 raNVMe-IP

The raNVMe-IP implements NVMe protocol of the host side to access one NVMe SSD. Up to 32 Write or Read commands with random addressing can be sent to raNVMe-IP. Six commands are supported in the IP, i.e., Write, Read, Identify, Shutdown, SMART and Flush. raNVMe-IP can connect with the PCIe hard IP directly. More details of raNVMe-IP are described in datasheet.

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

2.2.2 Integrated Block for PCIe

This block is the hard IP in Xilinx device which implements Physical, Data Link and Transaction Layers of PCIe specification. More details are described in Xilinx document.

PG156: UltraScale Devices Gen3 Integrated Block for PCI Express

PG213: UltraScale+ Devices Integrated Block for PCI Express

2.2.3 Dual port RAM

Two dual port RAMs, CtmRAM and IdenRAM, store the returned data from Identify command and SMART command respectively.

IdenRAM has 8 Kbyte size to store 8 Kbyte data, output from Identify command. raNVMe-IP and LAxi2Reg have the different data bus size, 128-bit on raNVMe-IP but 32-bit on LAxi2Reg, so IdenRAM has the different bus size for connecting with two modules. Also, raNVMe-IP has double word enable to write only 32-bit data in some cases. The RAM setting on Xilinx IP tool supports the write byte enable, so the small logic to convert double word enable to be write byte enable is designed as shown in Figure 2-9.

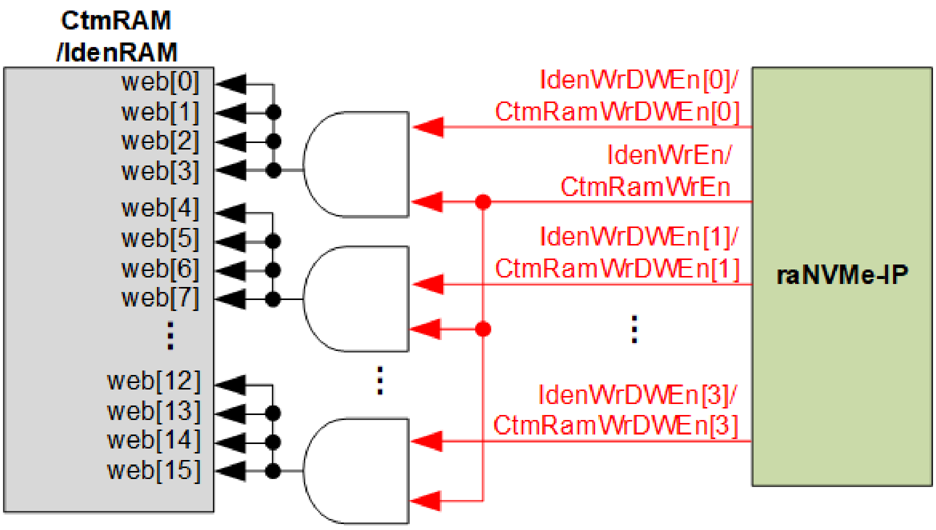

Figure 2-9 Byte write enable conversion logic

Bit[0] of WrDWEn with WrEn signal are the inputs to AND logic. The output of AND logic is fed to bit[3:0] of IdenRAM byte write enable. Bit[1], [2] and [3] of WrDWEn are applied to be bit[7:4], [11:8] and [15:12] of IdenRAM write byte enable respectively.

Comparing with IdenRAM, CtmRAM is implemented by true dual-port RAM with byte write enable. The small logic to convert double word enable of custom interface to be byte write enable must be used, similar to IdenRAM. True dual-port RAM is used to support the additional features when the customized custom command needs the data input. To support SMART command, using simple dual port RAM is enough. The data size returned from SMART command is 512 bytes.

2.3 CPU and Peripherals

32-bit AXI4-Lite bus is applied to be the bus interface for CPU accessing the peripherals such as Timer and UART. The test system of raNVMe-IP is connected with CPU as a peripheral on 32-bit AXI4-Lite 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. The base address and the range for accessing the test logic are defined in the CPU system. So, the hardware logic must be designed to support AXI4-Lite bus standard for CPU writing and reading. LAxi2Reg module is designed to connect with the CPU system as shown in Figure 2-10.

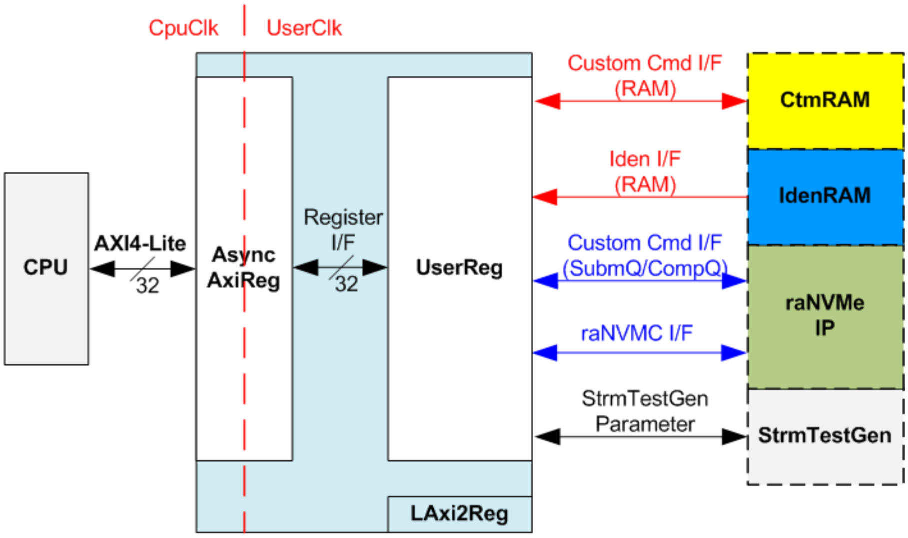

Figure 2-10 CPU and peripherals hardware

LAxi2Reg consists of AsyncAxiReg and UserReg. AsyncAxiReg is designed to convert the AXI4-Lite signals to be the simple register interface which has 32-bit data bus size, similar to AXI4-Lite data bus size. Additionally, AsyncAxiReg includes asynchronous logic to support clock crossing between CpuClk and UserClk domain.

UserReg includes the register file of the parameters and the status signals of other modules in the Test system, i.e., CtmRAM, IdenRAM, raNVMe-IP and StrmTestGen. More details of AsyncAxiReg and UserReg are described as follows.

2.3.1 AsyncAxiReg

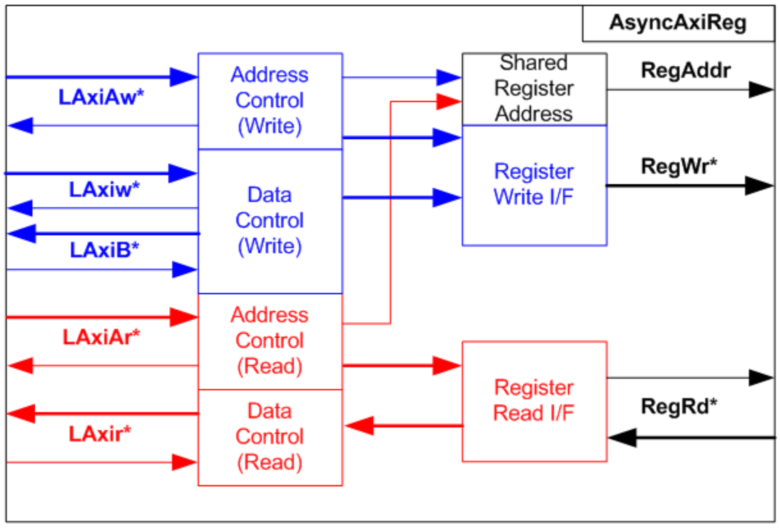

Figure 2-11 AsyncAxiReg Interface

The signal on AXI4-Lite bus interface can be split into five groups, i.e., LAxiAw* (Write address channel), LAxiw* (Write data channel), LAxiB* (Write response channel), LAxiAr* (Read address channel) and LAxir* (Read data channel). More details to build custom logic for AXI4-Lite bus is described in following document.
https://forums.xilinx.com/xlnx/attachments/xlnx/NewUser/34911/1/designing_a_custom_axi_slave_rev1.pdf

According to AXI4-Lite standard, the write channel and the read channel are operated independently. Also, the control and data interface of each channel are run separately. So, the logic inside AsyncAxiReg to interface with AXI4-Lite bus is split into four groups, i.e., Write control logic, Write data logic, Read control logic and Read data logic as shown in the left side of Figure 2-11. Write control I/F and Write data I/F of AXI4-Lite bus are latched and transferred to be Write register interface with clock-crossing registers. In the same way, Read control I/F and Read data I/F of AXI4-Lite bus are latched and transferred to be Read register interface with clock-crossing registers. In Register interface, RegAddr is shared signal for write and read access, so it loads the value from LAxiAw for write access or LAxiAr for read access.

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.

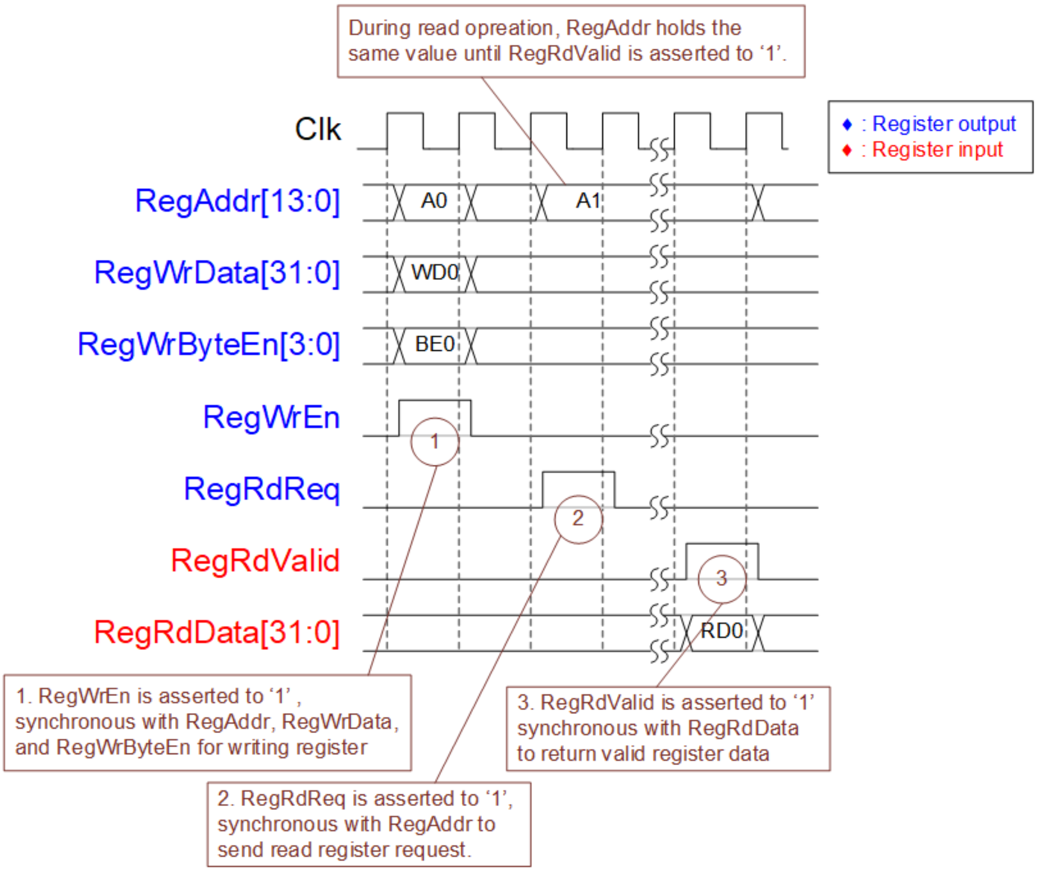

Figure 2-12 Register interface timing diagram

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, AsyncAxiReg asserts RegRdReq to '1' with the valid value of RegAddr. 32-bit data must be returned after receiving the read request. The slave must monitor RegRdReq signal to start the read transaction. During read operation, the address value (RegAddr) does not change the value until RegRdValid is asserted to '1'. So, 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, AsyncAxiReg forwards the read value to LAxir* interface.

2.3.2 UserReg

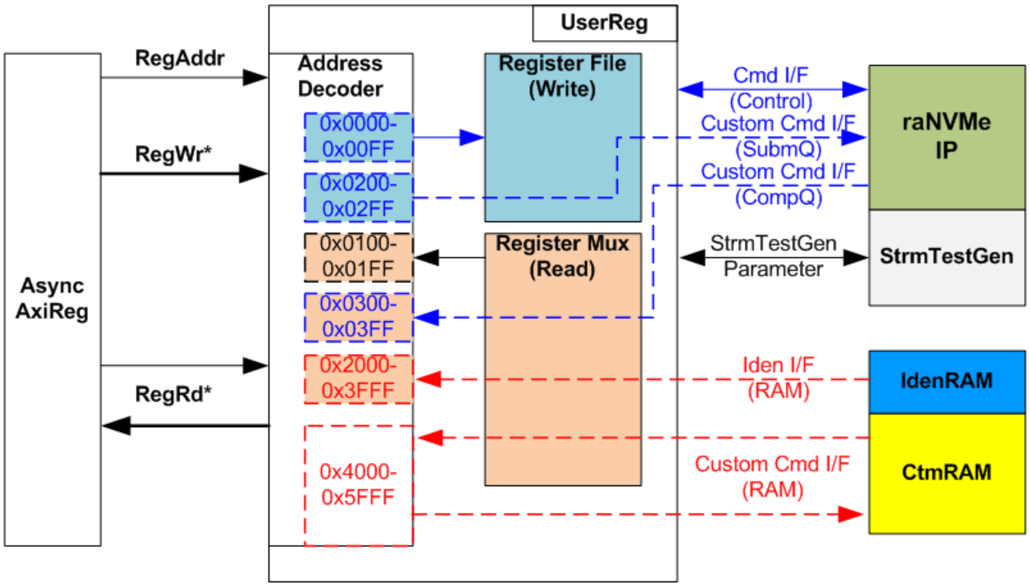

Figure 2-13 UserReg Interface

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

1) 0x0000 – 0x00FF: mapped to set the command with the parameters of raNVMe-IP and StrmTestGen. This area is write access only.
2) 0x0200 – 0x02FF: mapped to set the parameters for custom command interface of raNVMe-IP. This area is write access only.
3) 0x0100 – 0x01FF: mapped to read the status signals of raNVMe-IP and StrmTestGen. This area is read access only.
4) 0x0300 – 0x03FF: mapped to read the status of custom command interface (raNVMe-IP). This area is read access only.
5) 0x2000 – 0x3FFF: mapped to read data from IdenRAM. This area is read access only.
6) 0x4000 – 0x5FFF: mapped to write or read data with custom command RAM interface. This area supports write and read access. The demo shows only read access for running SMART command.

Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The register file inside UserReg is 32-bit bus size, so write byte enable (RegWrByteEn) is not used. To write hardware registers, the CPU must use 32-bit pointer to place 32-bit valid value on the write data bus.

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 data. Next the address decoder uses the upper bit to select the read data from each area for returning to CPU. Totally, the latency of read data is equal to two clock cycles, so 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	Register Name (Label in the "ranvmestrmtest.c")	Description
0x0000 – 0x00FF: Control signals of raNVMe-IP and StrmTestGen (Write access only)		
BA+0x0000	User Address (Low) Reg (USRADRL_REG)	[31:0]: Input to be start address as 512-byte unit (UserAddr[31:0] of raNVMe-IP for Write or Read command)
BA+0x0004	User Address (High) Reg (USRADRH_REG)	[15:0]: Input to be start address as 512-byte unit (UserAddr[47:32] of raNVMe-IP for Write or Read command)
BA+0x0008	User Stop Reg (USRSTOP_REG)	[0]: Assert to '1' to stop Write/Read command. This flag is auto-cleared by the hardware.
BA+0x0010	User Command Reg (USRCMD_REG)	[2:0]: Input to be user command (UserCmd of raNVMe-IP) "000": Identify, "001": Shutdown, "010": Write SSD, "011": Read SSD, "100": SMART, "110": Flush, "101"/"111": Reserved When this register is written with Single command (not Write/Read), the new command request (raNVMCValid) is asserted to raNVMe-IP. Otherwise, start flag for Write or Read command is asserted to StrmTestGen. After that, the command request (raNVMCValid) for Multiple command is asserted by StrmTestGen.
BA+0x0014	Test Pattern Reg (PATTSEL_REG)	[2:0]: Select test pattern. "000"-Increment, "001"-Decrement, "010"-All 0, "011"-All 1, "100"-LFSR. [3]: Verification enable. '0' -No verification, '1'-Enable verification.
BA+0x0020	NVMe Timeout Reg (NVMTIMEOUT_REG)	[31:0]: Timeout value of raNVMe-IP (TimeOutSet[31:0] of raNVMe-IP)
0x0100 – 0x01FF: Status signals of raNVMe-IP and StrmTestGen (Read access only)		
BA+0x0100	User Status Reg (USRSTS_REG)	[0]: Mapped to raNVMBusy of raNVMe-IP. '0': IP is Idle, '1': IP is busy. [1]: Mapped to raNVMError of raNVMe-IP. '0': No error, '1': Error is found. [2]: Data verification fail. '0': Normal, '1': Error. [3]: Busy flag of StemTestGen. '0'-Idle, '1'- Write/Read command is operating.
BA+0x0104	Total disk size (Low) Reg (LBASIZEL_REG)	[31:0]: Mapped to LBASize[31:0] of raNVMe-IP
BA+0x0108	Total disk size (High) Reg (LBASIZEH_REG)	[15:0]: Mapped to LBASize[47:32] of raNVMe-IP
BA+0x010C	User Error Type Reg (USRERRTYPE_REG)	[31:0]: Mapped to UserErrorType[31:0] of raNVMe-IP to show error status
BA+0x0110	PCIe Status Reg (PCISTS_REG)	[0]: PCIe linkup status from PCIe hard IP ('0': No linkup, '1': linkup) [3:2]: PCIe link speed from PCIe hard IP ("00": Not linkup, "01": PCIe Gen1, "10": PCIe Gen2, "11": PCIe Gen3) [7:4]: PCIe link width status from PCIe hard IP ("0001": 1-lane, "0010": 2-lane, "0100": 4-lane, "1000": 8-lane) [13:8]: Current LTSSM State of PCIe hard IP. Please see more details of LTSSM value in Integrated Block for PCIe datasheet
BA+0x0114	NVMe CAP Reg (NVMCAP_REG)	[31:0]: Mapped to NVMeCAPReg[31:0] of raNVMe-IP
BA+0x0118	Admin Completion Status Reg (ADMCOMPSTS_REG)	[15:0]: Mapped to AdmCompStatus[15:0] of raNVMe-IP to show status of Admin completion
BA+0x011C	IO Completion Status Reg (IOCOMPSTS_REG)	[31:0]: Mapped to IOCompStatus[15:0] of raNVMe-IP to show status of I/O completion.
BA+0x0120	NVMe IP Test pin Reg (NVMTESTPIN_REG)	[31:0]: Mapped to TestPin[31:0] of raNVMe-IP

Address Rd/Wr	Register Name (Label in the "ranvmestrmtest.c")	Description
0x0100 – 0x01FF: Status signals of raNVMe-IP and StrmTestGen (Read access only)		
BA+0x0130	Expected value Word0 Reg (EXPPATW0_REG)	[31:0]: Bit[31:0] of the expected data at the 1st failure data in Read command
BA+0x0134	Expected value Word1 Reg (EXPPATW1_REG)	[31:0]: Bit[63:32] of the expected data at the 1st failure data in Read command
BA+0x0138	Expected value Word2 Reg (EXPPATW2_REG)	[31:0]: Bit[95:64] of the expected data at the 1st failure data in Read command
BA+0x013C	Expected value Word3 Reg (EXPPATW3_REG)	[31:0]: Bit[127:96] of the expected data at the 1st failure data in Read command
BA+0x0140	Read value Word0 Reg (RDPATW0_REG)	[31:0]: Bit[31:0] of the read data at the 1st failure data in Read command
BA+0x0144	Read value Word1 Reg (RDPATW1_REG)	[31:0]: Bit[63:32] of the read data at the 1st failure data in Read command
BA+0x0148	Read value Word2 Reg (RDPATW2_REG)	[31:0]: Bit[95:64] of the read data at the 1st failure data in Read command
BA+0x014C	Read value Word3 Reg (RDPATW3_REG)	[31:0]: Bit[127:96] of the read data at the 1st failure data in Read command
BA+0x0150	Data Failure Address(Low) Reg (RDFAILNOL_REG)	[31:0]: Bit[31:0] of the byte address of the 1st failure data in Read command
BA+0x0154	Data Failure Address(High) Reg (RDFAILNOH_REG)	[24:0]: Bit[56:32] of the byte address of the 1st failure data in Read command
BA+0x0158	Completed Count (Low) Reg (CMDCMPCNTL_REG)	[31:0]: Bit[31:0] of the completed command count in StrmTestGen (StrmCompCnt of StrmTestGen)
BA+0x015C	Completed Count (High) Reg (CMDCMPCNTH_REG)	[12:0]: Bit[44:32] of the completed command count in StrmTestGen (StrmCompCnt of StrmTestGen)
Other interfaces (Custom command of raNVMe-IP, IdenRAM and Custom RAM)		
BA+0x0200 – BA+0x023F Wr	Custom Submission Queue Reg (CTMSUBMQ_REG)	[31:0]: Submission queue entry of SMART and Flush command. Input to be CtmSubmDW0-DW15 of raNVMe-IP. 0x200: DW0, 0x204: DW1, …, 0x23C: DW15
BA+0x0300 – BA+0x030F Rd	Custom Completion Queue Reg (CTMCOMPQ_REG)	[31:0]: CtmCompDW0-DW3 output from raNVMe-IP. 0x300: DW0, 0x304: DW1, …, 0x30C: DW3
BA+0x0800 Rd	IP Version Reg (IPVERSION_REG)	[31:0]: Mapped to IPVersion[31:0] of raNVMe-IP
BA+0x2000 – BA+0x2FFF Rd	Identify Controller Data (IDENCTRL_REG)	4Kbyte Identify Controller Data Structure
BA+0x3000 – BA+0x3FFF Rd	Identify Namespace Data (IDENNAME_REG)	4Kbyte Identify Namespace Data Structure
BA+0x4000 – BA+0x5FFF Wr/Rd	Custom command Ram (CTMRAM_REG)	Connect to 8K byte CtmRAM interface. Used to store 512-byte data output from SMART Command.

3 CPU Firmware

3.1 Test firmware (ranvmestrmtest.c)

After system boot-up, CPU starts the initialization sequence as follows.

1) CPU initializes UART and Timer parameters.
2) CPU waits until PCIe connection links up (PCISTS_REG[0]='1').
3) CPU waits until raNVMe-IP completes initialization process (USRSTS_REG[0]='0'). If some errors are found, the process stops with displaying the error message.
4) CPU displays PCIe link status (the number of PCIe lanes and the PCIe speed) by reading PCISTS_REG[7:2].
5) CPU displays the main menu. There are six menus for running six commands of raNVMe-IP, i.e., Identify, Write, Read, SMART, Flush and Shutdown.

More details for operating each command in CPU firmware are described as follows.

3.1.1 Identify Command

The sequence of the firmware when user selects Identify command is below.

1) Set USRCMD_REG="000". Next, Test logic generates command and asserts command request to raNVMe-IP. After that, raNVMe-IP busy flag (USRSTS_REG[0]) changes from '0' to '1'.
2) CPU waits until the operation is completed or some errors are found by monitoring USRSTS_REG[1:0].

   Bit[0] is de-asserted to '0' after finishing operating the command. After the command is completed, the data from Identify command of raNVMe-IP is stored in IdenRAM.
   Bit[1] is asserted to '1' when some errors are detected. The error message is displayed on the console to show the error details, decoded from USRERRTYPE_REG[31:0]. Finally, the process is stopped.

3) After raNVMe-IP busy flag (USRSTS_REG[0]) is de-asserted to '0', CPU displays the information decoded from IdenRAM (IDENCTRL_REG) such as SSD model name and the information from raNVMe-IP output such as SSD capacity (LBASIZEH/L_REG).

3.1.2 Write/Read Command by Start/Stop

The sequence of the firmware when user selects Write/Read command is below.

1) Receive start address and test pattern from Serial console. If some inputs are invalid, the operation is cancelled.
   <u>Note</u>: Start address must be aligned to 8.
2) Set test pattern to PATTSEL_REG and set start address to USRADRL/H_REG[31:0].
3) Set USRCMD_REG[2:0]="010" for Write command or "011" for Read command.
4) CPU reads error status by reading USRSTS_REG[2:1]. Display the error message when some bits are asserted to '1'.
   Bit[1] is asserted when IP error is detected. The process is hanged up when this error is found.
   Bit[2] is asserted when running Read command and data failure is found. The verification error message is displayed.
5) In every second, the current transfer size, calculated from CMDCMPCNTL/H_REG x 4KB (data size of one command), is displayed on the console.
6) Repeat step 5 and 6 until user enters 'x' to stop Write/Read operation. After that, CPU sets USRSTOP_REG[0]='1'.
7) CPU waits until StrmTestGen finishes the operation by reading busy flag of StrmTestGen (USRSTS_REG[3]='0'). Display the error message if some errors are found.
8) Display the test result on the console, i.e., total time usage, total transfer size and transfer speed in MB/s and IOPS unit.

3.1.3 SMART Command

The sequence of the firmware when user selects SMART command is below.

1) Set 16-Dword of Submission queue entry (CTMSUBMQ_REG) to be SMART command value.
2) Set USRCMD_REG[2:0]=”100”. Next, Test logic generates command and asserts the request to raNVMe-IP. After that, raNVMe-IP busy flag (USRSTS_REG[0]) changes from ‘0’ to ‘1’.
3) CPU waits until the operation is completed or some errors are found by monitoring USRSTS_REG[1:0].

   Bit[0] is de-asserted to ‘0’ after finishing operating the command. If the command is completed, the data from SMART command of raNVMe-IP is stored in CtmRAM.
   Bit[1] is asserted when some errors are detected. The error message is displayed on the console to show the error details, decoded from USRERRTYPE_REG[31:0]. Finally, the process is stopped.

4) After raNVMe-IP busy flag (USRSTS_REG[0]) is de-asserted to ‘0’, CPU displays some information decoded from CtmRAM (CTMRAM_REG) such as Temperature, Total Data Read, Total Data Written, Power On Cycles, Power On Hours and Number of Unsafe Shutdown.

   More details of SMART log are described in NVM Express Specification.
   https://nvmexpress.org/resources/specifications/

3.1.4 Flush Command

The sequence of the firmware when user selects Flush command is below.

1) Set 16-Dword of Submission queue entry (CTMSUBMQ_REG) to be Flush command value.
2) Set USRCMD_REG[2:0]=”110”. Next, Test logic generates command and asserts the request to raNVMe-IP. After that, raNVMe-IP busy flag (USRSTS_REG[0]) changes from ‘0’ to ‘1’.
3) CPU waits until the operation is completed or some errors are found by monitoring USRSTS_REG[1:0].

   Bit[0] is de-asserted to ‘0’ after finishing operating the command. If the command is completed, the CPU goes back to the main menu.
   Bit[1] is asserted when some errors are detected. The error message is displayed on the console to show the error details, decoded from USRERRTYPE_REG[31:0]. Finally, the process is stopped.

3.1.5 Shutdown Command

The sequence of the firmware when user selects Shutdown command is below.

1) Set USRCMD_REG[2:0]="001". Next, Test logic generates command and asserts the request to raNVMe-IP. After that, raNVMe-IP busy flag (USRSTS_REG[0]) changes from '0' to '1'.
2) CPU waits until the operation is completed or some errors are found by monitoring USRSTS_REG[1:0].

   Bit[0] is de-asserted to '0' after finishing operating the command. After that, the CPU goes to the next step.
   Bit[1] is asserted when some errors are detected. The error message is displayed on the console to show the error details, decoded from USRERRTYPE_REG[31:0]. Finally, the process is stopped.

3) After Shutdown command, the SSD and raNVMe-IP change to inactive status. So, the CPU cannot receive the new command from user and the user must power off the test system.

3.2 Function list in Test firmware

int exec_ctm(unsigned int user_cmd)	
Parameters	user_cmd: 4-SMART command, 6-Flush command
Return value	0: No error, -1: Some errors are found in the raNVMe-IP
Description	Run SMART command or Flush command, following in topic 3.1.3 (SMART Command) and 3.1.4 (Flush Command).

int flush_ctmnvm(void)	
Parameters	None
Return value	0: No error, -1: Some errors are found in the raNVMe-IP
Description	Set Flush command to CTMSUBMQ_REG and call exec_ctm function to operate Flush command.

int get_param(userin_struct* userin)	
Parameters	userin: Two inputs from user, i.e., start address 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 iden_dev(void)	
Parameters	None
Return value	None
Description	Run Identify command, following in topic 3.1.1 (Identify Command).

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

void show_pciestat(void)	
Parameters	None
Return value	None
Description	Read PCISTS_REG until the read value from two read times is stable. After that, display the read value on the console.

void show_result(void)	
Parameters	None
Return value	None
Description	Print total size by reading CMDCMPCNT_REG and then calling 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 on MB/s unit and IOPS 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_smart_hex(unsigned char *char_ptr16B)	
Parameters	*char_ptr16B
Return value	None
Description	Display SMART data as hexadecimal unit.

void show_smart_raw(unsigned char *char_ptr16B)	
Parameters	*char_ptr16B
Return value	None
Description	Display SMART data as decimal unit when the input value is less than 4 MB. Otherwise, display overflow message.

void show_smart_unit(unsigned char *char_ptr16B)	
Parameters	*char_ptr16B
Return value	None
Description	Display SMART data as GB or TB unit. When the input value is more than a limit (500 PB), the overflow message is displayed instead.

void show_vererr(void)	
Parameters	None
Return value	None
Description	Read RDFAILNOL/H_REG (error byte address), EXPPATW0-W3_REG (expected value) and RDPATW0-W3_REG (read value) to display verification error details on the console.

void shutdown_dev(void)	
Parameters	None
Return value	None
Description	Run Shutdown command, following in topic 3.1.5 (Shutdown Command)

int smart_ctmadm(void)	
Parameters	None
Return value	0: No error, -1: Some errors are found in the raNVMe-IP
Description	Set SMART command to CTMSUBMQ_REG and call exec_ctm function to operate SMART command. Finally, decode and display SMART information on the console.

int wrrd_auto(unsigned int user_cmd)	
Parameters	user_cmd: 2-Write command, 3-Read command
Return value	0: No error, -1: Receive invalid input or some errors are found.
Description	Run Write command or Read command by Start/Stop, following in topic 3.1.2 (Write/Read Command)

4 Example Test Result

The example test results when running demo system by using 280 GB Intel Optane 900P and KCU105 board (PCIe Gen3) in Write and Read access are shown in Figure 4-1.

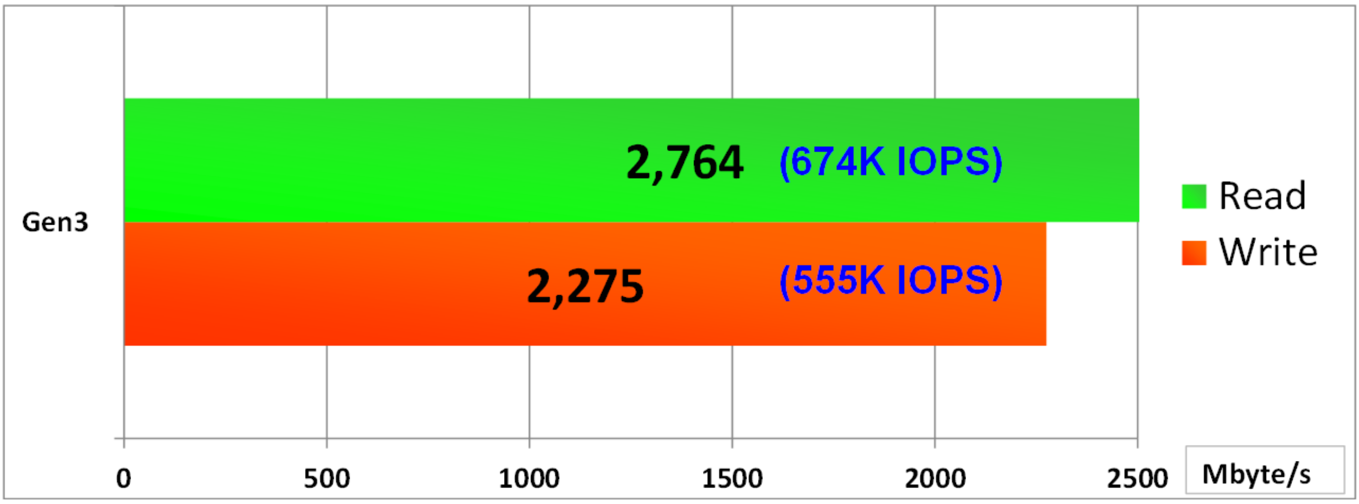

Figure 4-1 Test Performance of raNVMe-IP data stream demo

Write and Read performance depends on the SSD characteristic. Some SSDs show the better Write performance but Read performance is less. Read performance in some SSDs can improve by increasing memory size inside raNVMe-IP. The memory size modification inside raNVMe-IP can be customized.

5 Revision History

Revision	Date	Description
1.0	27-Nov-20	Initial Release
1.1	12-Jan-21	Update Test result