4-Ch RAID0 NVMe-IP for Gen5 reference design manual

Rev1.0 2-Oct-23

1 Introduction 2
2 Hardware overview 3
2.1 TestGen 5
2.2 NVMeRAID0x4IP 9
2.2.1 NVMe-IP for Gen5 10
2.2.2 PCIe Hard IP (R-Tile Avalon-ST Intel Hard IP for PCIe) 10
2.2.3 Two-port RAM 10
2.2.4 FIFO 11
2.2.5 RAID0x4 11
2.3 CPU and Peripherals 17
2.3.1 AsyncAvlReg 18
2.3.2 UserReg 20
3 CPU Firmware 23
3.1 Test firmware (nvmeraid0g5test.c) 23
3.1.1 Identify command 23
3.1.2 Write/Read command 24
3.1.3 SMART Command, ... 24
3.1.4 Flush Command 25
3.1.5 Secure Erase Command 25
3.1.6 Shutdown Command 25
3.2 Function list in Test firmware 26
4 Example Test Result 29
5 Revision History 30

1 Introduction

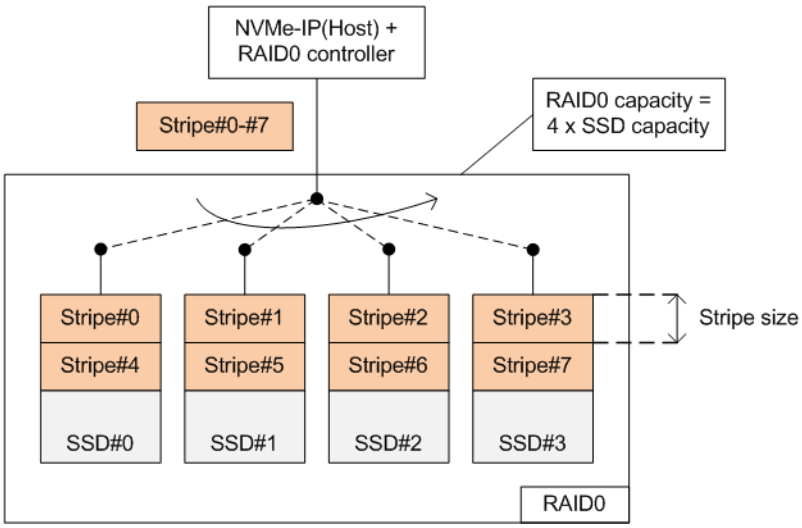

Figure 1-1 RAID0 by 4 SSDs data format

The RAID0 system utilizes multiple storage device to expand total storage capacity and enhance write/read performance. Assuming the total number of devices connecting in the RAID0 system is represented by N, the overall storage capacity of RAID0 becomes N times the capacity of a single device. Similarly, the write and read performance of RAID0 is nearly N times that of a single device.

Figure 1-1 illustrates the data format of RAID0. The data stream from the host side is divided into stripe unit for transferring data with each SSD at a time. The stripe size refers to the amount of data transferred with an SSD before switching to others. In this RAID0 reference design, the stripe size is set to **4Kbytes**.

For this demo, a system with four SSDs is employed. It is recommended to use the same SSD model for all channels to ensure compatibility and achieve optimal performance. As a result, the total capacity becomes four times that of a single SSD, while the write/read performance is nearly four times that of a single SSD.

In this demo, FIFO buffer implemented by Block Memory is utilized, which has smaller size compared to using DDR. Therefore, if the SSD suspends data transmission for an extended duration during the Write process, it results in the buffer becoming full, necessitating a pause in data transfer. The performance tested in the demo represents the average speed, rather than the sustained rate. User have the flexibility to modify the RAID0 reference design by increasing the numbers of SSD to attain improved performance and larger capacity. Furthermore, user can integrate DDR as a data buffer in the system to support high-speed transfers at a sustained rate.

Before running the reference design, it is recommended to read NVMe-IP for Gen5 datasheet and the standard demo which is a single-channel demo, via the following links.
https://dgway.com/products/IP/NVMe-IP/dg_nvme_datasheet_g5_intel/
https://dgway.com/products/IP/NVMe-IP/dg_nvmeip_refdesign_g5_intel/

2 Hardware overview

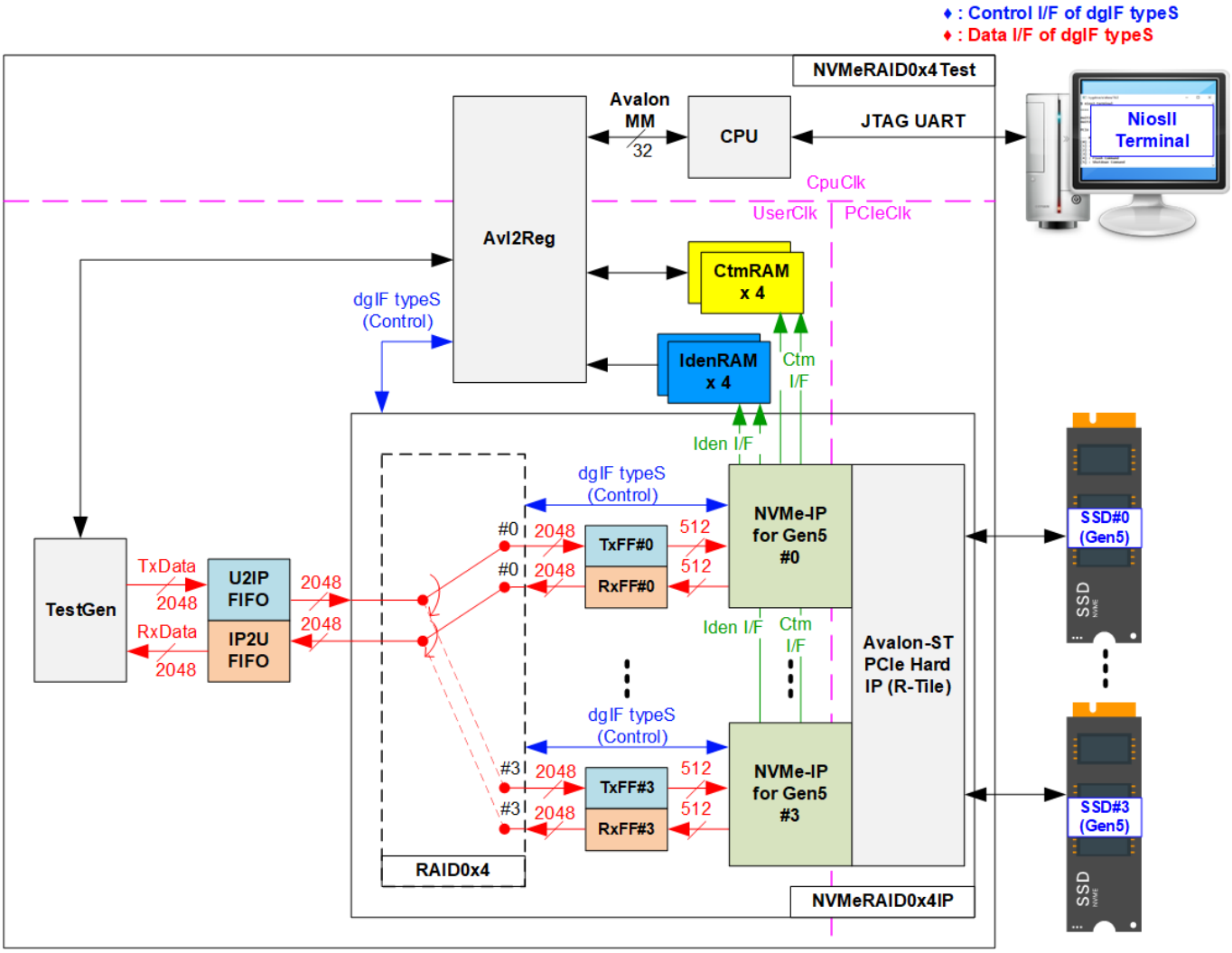

Figure 2-1 4-Channel RAID0 NVMe-IP for Gen5 hardware in refdesign

This test system consists of three hardware modules: TestGen, NVMeRAID0x4IP, and the CPU system consisting of CPU and Avl2Reg.

The TestGen module connects to the user interface of NVMeRAID0x4IP, which is specifically designed to be compatible with the NVMe-IP but quadrupling the data bus size. TestGen serves as an example of user logic for generating test data stream of Write command and verifying test data stream of Read command. The write and read data streams are stored at two FIFOs (U2IPFIFO and IP2UFIFO). TestGen always writes or reads data when the FIFO is ready, allowing for optimal transfer performance evaluation of the NVMeRAID0x4 system.

The NVMeRAID0x4IP module consists of RAID0x4, four NVMe-IPs, and a PCIe hard IP (R-Tile) enabling direct access to four NVMe Gen5 SSDs without PCIe switch. The PCIe hard IP is configured as 4x4-lane PCIe Gen5, capable of connecting up to 4 NVMe Gen5 SSDs. The CPU inputs command requests and parameters for each command though Avl2Reg module. RAID0x4 manages the command requests of four NVMe-IPs and controls the data transfer for Write and Read commands. The data interface of Custom and Identify commands is connected to RAMs, accessible by the CPU.

The CPU integrates a JTAG UART for communication with the user. The user can select the command type, configure command parameters, and monitor test progress through the console. The CPU firmware is designed to facilitate the execution of multiple test cases, ensuring comprehensive verification of IP functionality.

Figure 2-1 illustrates three clock domains: CpuClk, UserClk, and PCIeClk. CpuClk is the clock domain for the CPU and its peripherals, requiring a stable clock. It can be independent clock from other hardware. UserClk is the main clock domain for the user logic (TestGen) and NVMeRAID0x4IP. According to the NVMe-IP for Gen5 datasheet, the frequency of UserClk must be equal to or greater than half of the PCIeClk frequency. In the reference design, UserClk frequency is set to 280 MHz. PCIeClk is generated by PCIe hard IP, synchronized with the 256-bit Avalon stream. The frequency of PCIeClk is 500 MHz for 4-lane PCIe Gen5. Using a lower frequency of PCIeClk is possible, but it will limit the maximum bandwidth on the user interface of PCIe hard IP to be lower than PCIe Gen5 performance. Further hardware details are described below.

2.1 TestGen

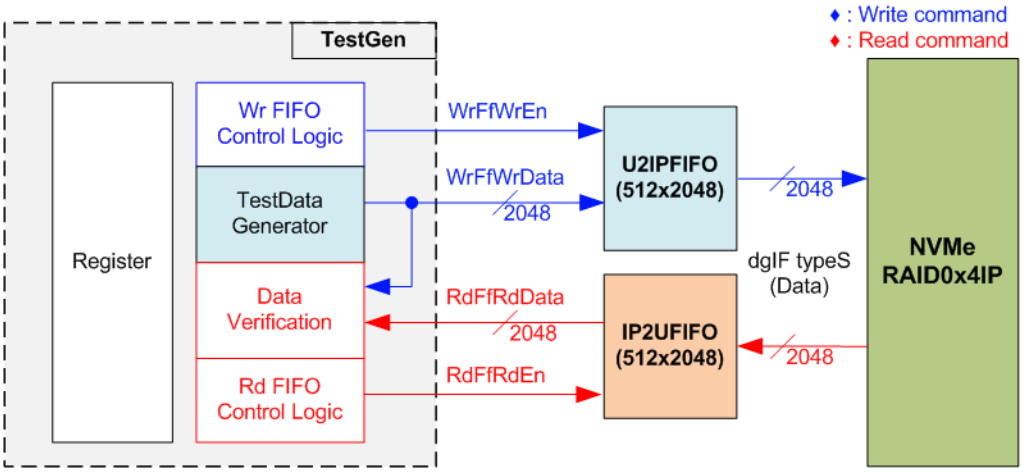

Figure 2-2 TestGen interface

The TestGen module manages the data interface of NVMeRAID0x4IP, enabling data transfer for both Write and Read commands. In case of a Write command, TestGen sends 2048-bit test data to NVMeRAID0x4IP via U2IPFIFO. In contrast, for a Read command, the test data is received from IP2UFIFO for comparison with the expected value, ensuring data accuracy. Data bandwidth of TestGen is configured to match that of NVMeRAID0x4IP by operating at the same clock frequency and data bus size.

The control logic within TestGen guarantees that the Write or Read enable signal is always asserted to 1b when the FIFO is ready to write or read data, respectively. This ensures that both U2IPFIFO and IP2UFIFO are always ready to transfer data with NVMeRAID0x4IP without delay, maximizing performance when writing and reading data with four SSDs through NVMeRAID0x4IP.

To provide a flexible test environment, user can adjust test parameters via the console such as total transfer size, transfer direction, and test pattern selector. These test parameters are stored in the Register block. The detailed hardware logic of TestGen is illustrated in Figure 2-3.

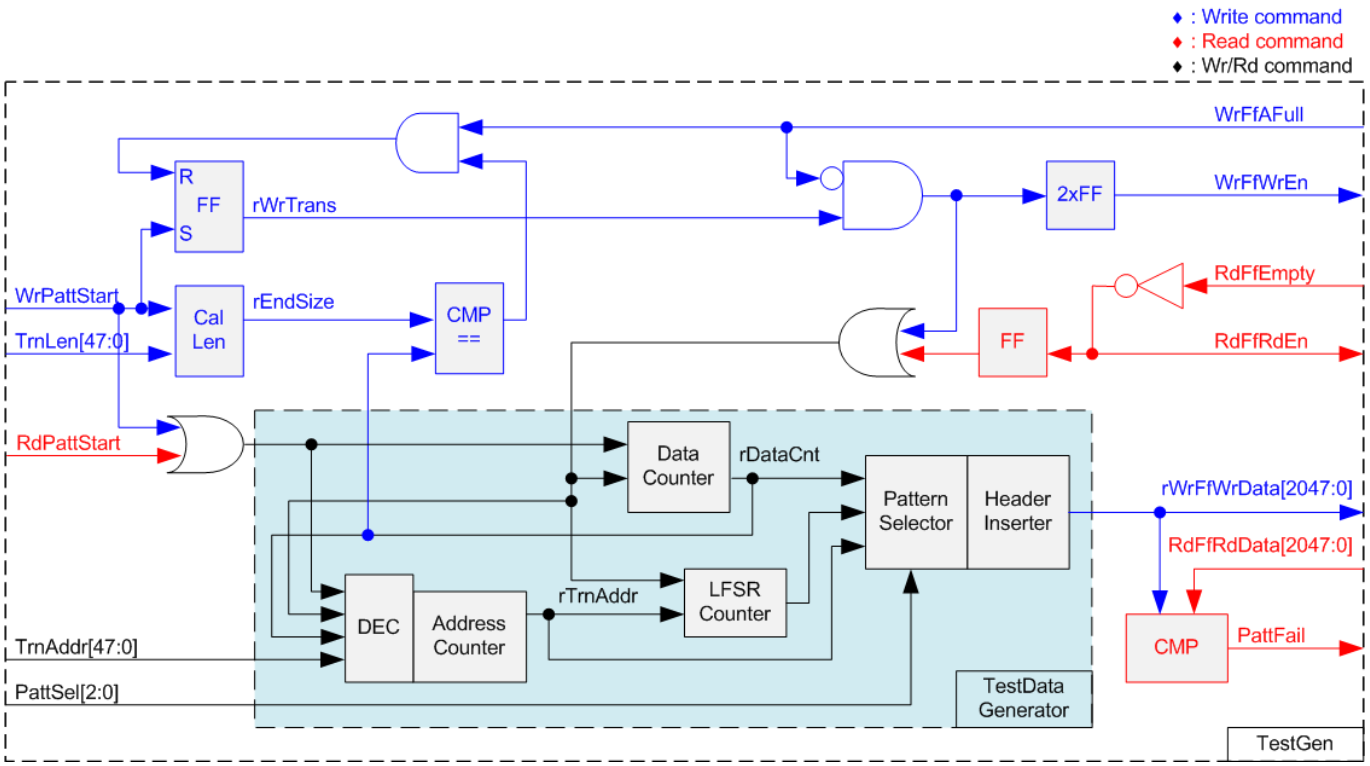

Figure 2-3 TestGen hardware

Figure 2-3 provides an overview of the TestGen module. The right side of the figure shows the utilization of flow control signals within the FIFO, including WrFfAFull and RdFfEmpty signals. During a write operation, if the FIFO almost reaches its capacity (indicated by WrFfAFull=1b), WrFfWrEn is set to 0b to pause data transmission to the FIFO. Similarly, during a read operation, if there is data available in the FIFO (denoted by RdFfEmpty=0b), the logic retrieves data from the FIFO for comparison by setting RdFfRdEn to 1b.

The left side of Figure 2-3 shows the logic designed to count the transfer size. Once the total data count (rDataCnt) matches the user-defined end size (rEndSize), the Write enable or Read enable of the FIFO is set to 0b to halt data transfer. The lower side of Figure 2-3 provides the details to generate test data for writing to the FIFO or verifying data from the FIFO. There are five available test patterns: all-zero, all-one, 32-bit incremental data, 32-bit decremental data, and LFSR, selected by Pattern Selector. In case of all-zero or all-one pattern, each data bit is set to zero or one, respectively. The remaining patterns involve splitting the data into two parts to create unique test data within each 4Kbytes data block, as shown in Figure 2-4.

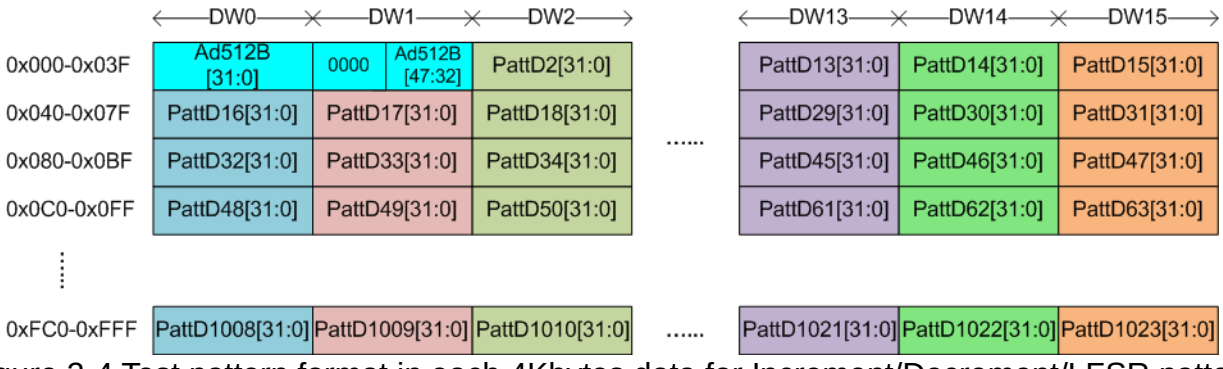

Figure 2-4 Test pattern format in each 4Kbytes data for Increment/Decrement/LFSR pattern

Within every 4Kbytes block, the first two double words (Dword#0 and Dword#1) form a 64-bit header, while the remaining words (Dword#2 – Dword#1023) are the actual test data. The header is generated using the address counter block, which employs a 4Kbyte address (rTrnAddr) to create a 512byte address (Ad512B). The initial value of the address counter is configured by the user (TrnAddr). Its value is incremented to the next 4Kbyte address value upon the completion of each 4Kbytes data transfer. The value of the remaining Dwords (DW#2 – DW#1023) depends on the pattern selector, which may be 32-bit incremental data, 32-bit decremental data, or LFSR. The 32-bit incremental data is generated using Data counter, while the decremental data can be obtained by connecting NOT logic to incremental data. The LFSR pattern is generated using the LFSR counter with the polynomial equation x^31 + x^21 + x + 1.

By employing a 4-step lookahead logic design, the system can generate four consecutive 32-bit LFSR data sets, resulting in a total of 128 bits of test data, within a single clock cycle. However, the RAID0 module requires 2048 bits of test data as input for each clock cycle, necessitating the incorporation of sixteen instances of the logic responsible for generating 128-bit test data. Each of these sixteen sets is individually configured with a unique initial value. These distinct sixteen initial values are derived from a combination of two signals: the 32 bits of 512bytes unit address (LBAAd) and its inverted counterpart (LBAAdB). The 32-bit initial values for these sixteen sets (PattD0, PattD1, …, PattD31) are listed in Table 2-1.

Table 2-1 Initial value of 16 data sets when using LFSR pattern

Sets of Data	Initial value
PattD0 [31:0]	LBAAd [31:0]
PattD1 [31:4]	LBAAd [31:4]
PattD1 [3:0]	LBAAdB [3:0]
PattD2 [31:8]	LBAAd [31:8]
PattD2 [7:0]	LBAAdB [7:0]
PattD3 [31:12]	LBAAd [31:12]
PattD3 [11:0]	LBAAdB [11:0]
PattD4 [31:16]	LBAAd [31:16]
PattD4 [15:0]	LBAAdB [15:0]
PattD5 [31:20]	LBAAd [31:20]
PattD5 [19:0]	LBAAdB [19:0]
PattD6 [31:24]	LBAAd [31:24]
PattD6 [23:0]	LBAAdB [23:0]
PattD7 [31:28]	LBAAd [31:28]
PattD7 [27:0]	LBAAdB [27:0]

Sets of Data	Initial value
PattD8 [31:0]	LBAAdB [31:0]
PattD9 [31:4]	LBAAdB [31:4]
PattD9 [3:0]	LBAAd [3:0]
PattD10 [31:8]	LBAAdB [31:8]
PattD10 [7:0]	LBAAd [7:0]
PattD11 [31:12]	LBAAdB [31:12]
PattD11 [11:0]	LBAAd [11:0]
PattD12 [31:16]	LBAAdB [31:16]
PattD12 [15:0]	LBAAd [15:0]
PattD13 [31:20]	LBAAdB [31:20]
PattD13 [19:0]	LBAAd [19:0]
PattD14 [31:24]	LBAAdB [31:24]
PattD14 [23:0]	LBAAd [23:0]
PattD15 [31:28]	LBAAdB [31:28]
PattD15 [27:0]	LBAAd [27:0]

As illustrated in Figure 2-5, each 32-bit initial value serves as the basis for generating the next three data using a look-ahead technique within a single clock cycle, as represented by the same color. This process facilitates the creation of the required 2048-bit test data, which can be either written to the FIFO as write data or compared with the data read from the FIFO to perform verification. In cases where the data verification process encounters an error, the Fail flag is set to 1b. For a more comprehensive understanding of the data generation process for the FIFO, it is recommended referring to Figure 2-6, which provides detailed step-by-step instructions.

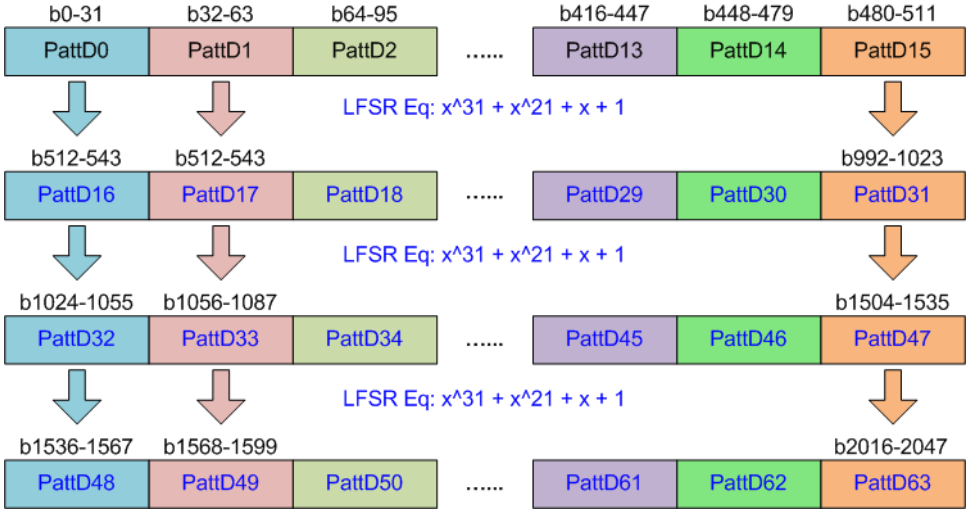

Figure 2-5 2048 bits of LFSR data generating in a single clock cycle

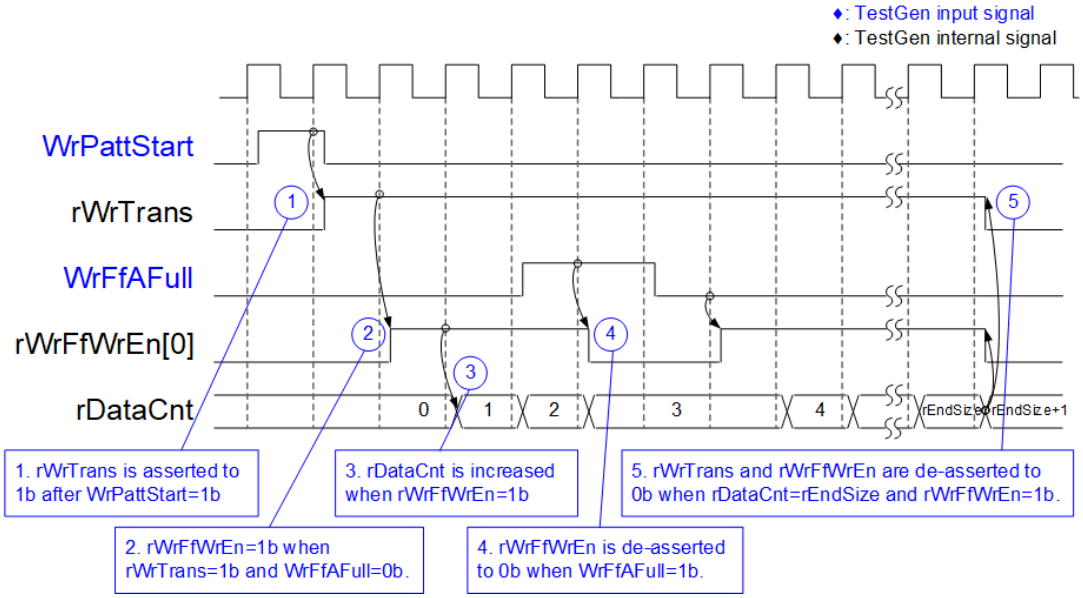

Figure 2-6 Timing diagram of Write operation in TestGen

1) The write operation is initiated by setting WrPattStart signal to 1b for one clock cycle, which is followed by the assertion of rWrTrans to enable the control logic for generating write enable to FIFO.
2) If two conditions are met (rWrTrans is asserted to 1b during the write operation and the FIFO is not full, indicated by WrFfAFull=0b), the write enable (rWrFfWrEn) to FIFO is asserted to 1b.
3) The write enable is fed back to the counter to count the total amount of data in the write operation.
4) If FIFO is almost full (WrFfAFull=1b), the write process is paused by de-asserting rWrFfWrEn to 0b.
5) The write operation is finished when the total data count (rDataCnt) is equal to the set value (rEndSize). At this point, both rWrTrans and rWrFfWrEn are de-asserted to 0b.

Unlike the write enable, the read enable signal is not stopped by total data count and not started by start flag. For read transfer, the read enable of FIFO is controlled by the empty flag of FIFO. When the read enable is asserted to 1b, the data counter and the address counter are increased for counting the total amount of data and generating the header of expected value, respectively.

2.2 NVMeRAID0x4IP

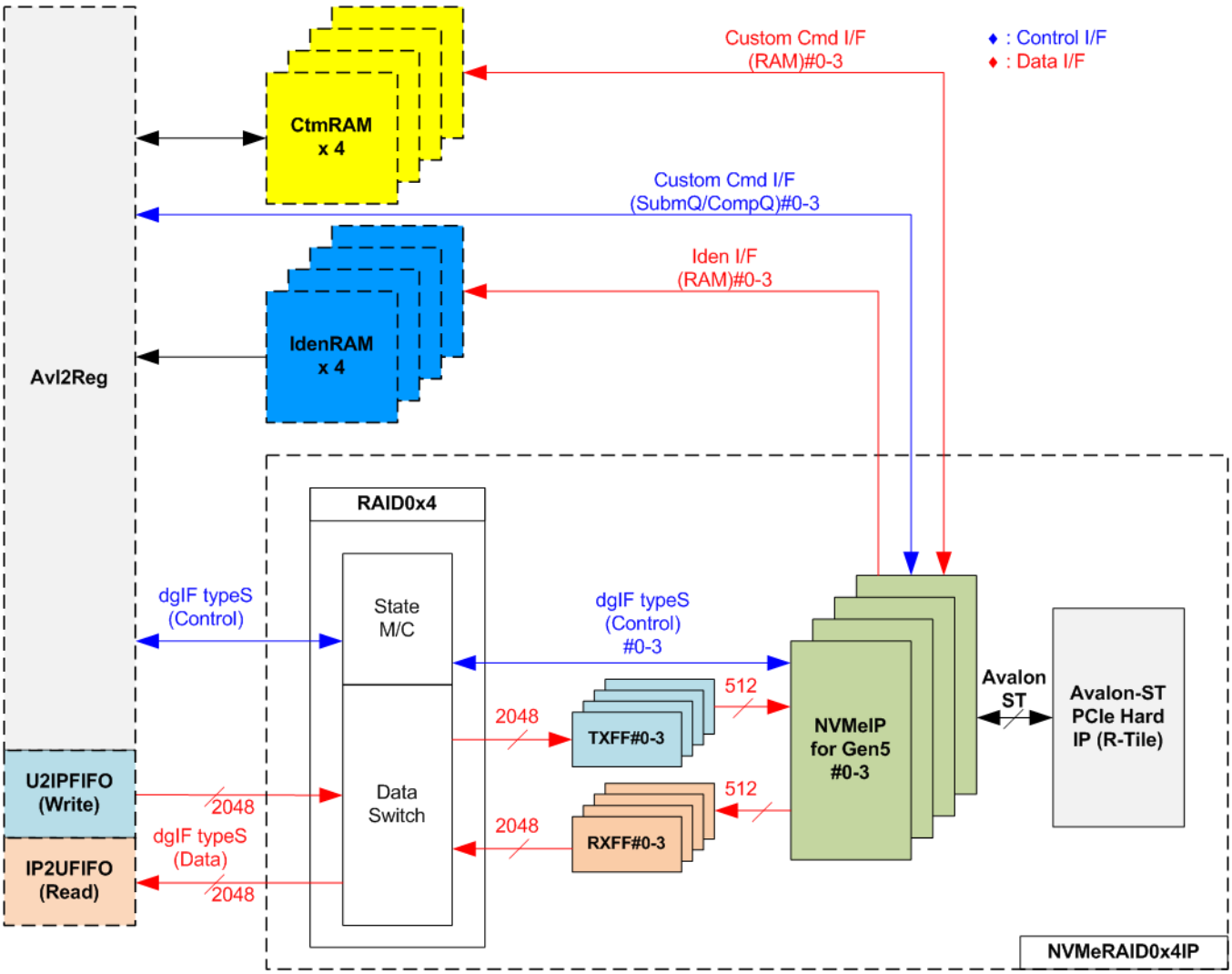

Figure 2-7 NVMeRAID0x4IP hardware

In the reference design, the NVMeRAID0x4IP's user interface consists of a control interface and a data interface. The control interface receives commands and parameters from either the Custom command interface or dgIF typeS, depending on the type of command. For instance, Custom command interface is used when operating SMART command, Flush command, or Secure Erase command.

On the other hand, the data interface of NVMeRAID0x4IP has four different interfaces, including Custom command RAM interface, Identify interface, FIFO input interface (dgIF typeS), and FIFO output interface (dgIF typeS). While the Custom command RAM interface is a bi-directional interface, the other interfaces are one directional interface. In the reference design, the Custom command RAM interface is used for one-directional data transfer when NVMeRAID0x4IP sends SMART data to Avl2Reg.

2.2.1 NVMe-IP for Gen5

The NVMe-IP module implements the NVMe protocol on the host side, enabling direct access to an NVMe SSD without the need for a PCIe switch connection. It supports seven commands, i.e., Write, Read, Identify, Shutdown, SMART, Flush, and Secure Erase. The NVMe-IP module can be connected to the PCIe Hard IP (R-Tile) within the Intel FPGA device directly. For a more comprehensive understanding of the NVMe-IP module, you can refer to the detailed information provided in the datasheet, available at following link.
https://dgway.com/products/IP/NVMe-IP/dg_nvme_datasheet_g5_intel/

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

This block represents hard IP integrated into Intel FPGA devices and is responsible for implementing the Physical, Data Link, and Transaction Layers of the PCIe protocol. Further details about this component can be found in the official Intel FPA documentation, accessible at following link.

R-Tile Avalon-ST Intel FPGA for PCIe
https://www.intel.com/content/www/us/en/docs/programmable/683501/

2.2.3 Two-port RAM

Two of two-Port RAMs, CtmRAM and IdenRAM, are utilized to store data returned from the Identify and SMART commands, respectively. IdenRAM is simple dual-port RAM with one read port and one write port, featuring an 8 Kbytes data capacity to store the 8 Kbyte data output from the Identify command.

The data bus sizes for NVMe-IP and Avl2Reg differ. NVMe-IP utilizes a 512-bit bus size, whereas Avl2Reg employs a 32-bit bus size. Consequently, IdenRAM operates as an asymmetric RAM, with different bus sizes for its Write and Read interfaces. Additionally, NVMe-IP incorporates a double-word enable feature, enabling it to write only 32-bit data in specific scenarios.

The RAM settings in the IP catalog of Quartus facilitate write byte enable, resulting in each bit of the double-word enable being expanded to a 4-bit write byte enable, as shown in Figure 2-8.

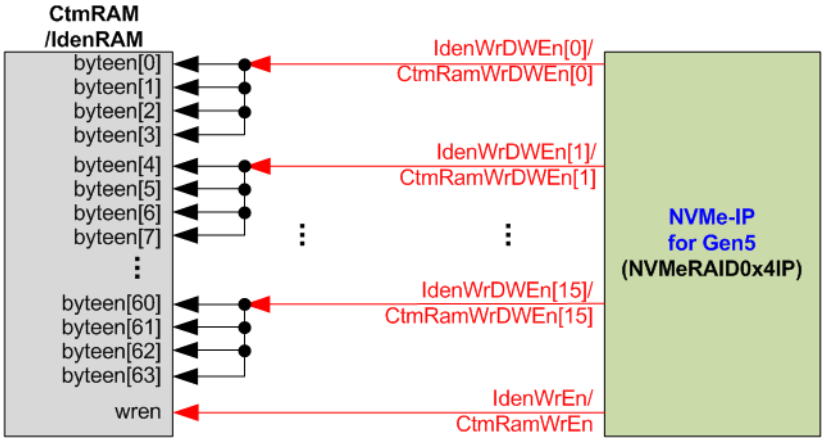

Figure 2-8 Byte enable conversion logic

The 16 bits of the WrDWEn are used to drive the byte write enable of IdenRAM as follows: bits[0], [1], …, [15] of WrDWEn are fed to bits[3:0], [7:4], …, [63:60] of IdenRAM byte write enable.

On the other hand, CtmRAM is implemented as a two-Port RAM with two read ports and two write ports, and with byte write enable. The connection from the double-word enable of NVMe-IP to byte enable of CtmRAM is similar to that of IdenRAM. The two-Port RAM is utilized to support additional features when the customized Custom command requires data input. For supporting SMART command, a simple dual-port RAM is sufficient, even though the data size returned from the SMART command is 512 bytes. However, CtmRAM is implemented with an 8Kbyte RAM for the customized Custom command.

2.2.4 FIFO

Within NVMeRAID0x4IP, the data interface of each NVMe-IP is connected to two FIFOs: TxFF and RxFF. These FIFOs are applied for two purposes: buffering the data and converting the data bus size between the RAID0x4 interface, which has a size of 2048 bits, and the NVMe-IP interface, which has a size of 512 bits. Both TxFF and RxFF are asymmetric FIFOs with a capacity of 128 Kbytes.

2.2.5 RAID0x4

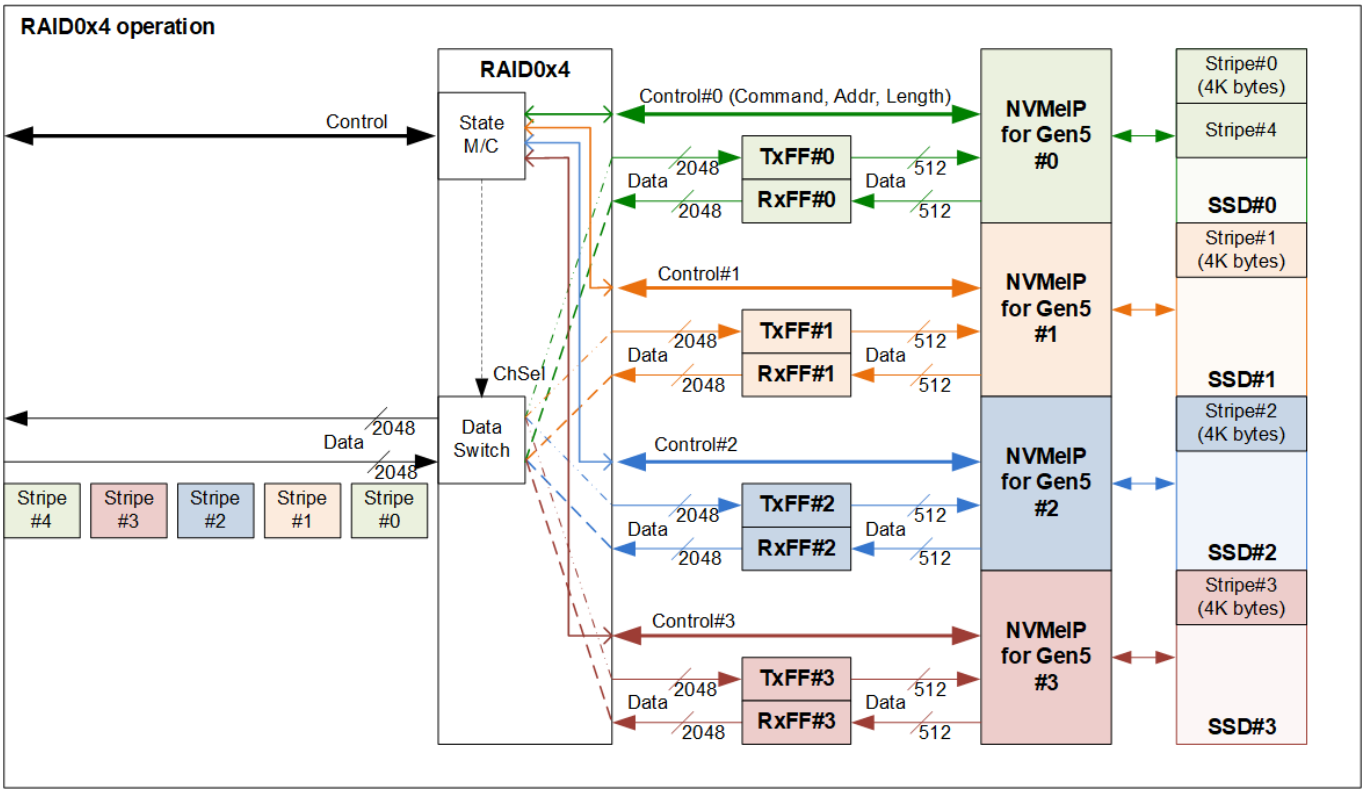

Figure 2-9 RAID0x4 operation

RAID0x4 includes a state machine responsible for managing the control interface and a data switch for managing the data interface. When a new command request is received from Avl2Reg, the state machine decodes the command type and determines the corresponding parameters of each NVMe-IP. For instance, when executing the Write and Read commands, the state machine determines the address and transfer length required of each NVMe-IP. Then, the state machine generates a command request to the NVMe-IP.

Figure 2-9 provides a comprehensive overview of the data flow required to operate the RAID0x4 function. The input data stream from the user (TestGen) is partitioned into segments of the stripe size (measured in 4Kbyte units). RAID0 determines the first active NVMe-IP and transfers the first 4Kbyte data to it.

For instance, in the scenario depicted in Figure 2-9, a Write command is executed, and the first active NVMe-IP is determined to be Ch#0. Consequently, Stripe#0 is stored in SSD#0 through TxFF#0 and NVMe-IP#0. Subsequently, the active channel switches to the next one (Ch#1) where the subsequent stripe (Stripe#1) is stored in SSD#1 through TxFF#1 and NVMe-IP#1. This process continues, with the active channel transitioning in a cyclical sequence: Ch#0 -> Ch#1 -> Ch#2 -> Ch#3 -> Ch#0, with each switch occurring after every 4Kbyte data transfer, until all the data has been successfully transferred.

To determine the first active channel, it is based on the LSB of the start address, measured in 4Kbyte units. Specifically, when the LSB is 00b, the first active channel is Ch#0; when it is 01b, it corresponds to Ch#1. Similarly, a LSB of 10b designates Ch#2 as the first active channel, and 11b designates Ch#3.

The data switch operation involves the use of pipeline registers, which introduce an overhead time when transitioning the active channel during each 4Kbyte data transfer. In the reference design, this overhead time corresponds to 1 clock cycle out of every 16 clock cycles, making up 6.25% of the overall operation time, as 16 clock cycles are required for a 4Kbyte data transfer.

To address this overhead, the clock frequency for RAID0x4 must be set 6.25% higher than the clock frequency required for the NVMe module. For instance, to operate with PCIe Gen5 at 250 MHz, the clock frequency for RAID0x4 would need to be adjusted to 265.625 MHz (106.25% of 250 MHz). In the reference design, a clock frequency of 280 MHz is employed to optimize Write/Read performance.

The user interface of RAID0x4 is designed to be compatible with the user interface of NVMe-IP, utilizing dgIF typeS. Table 2-2 shows the detailed information of the user interface.

Table 2-2 Signal description of RAID0x4 (only User interface)

Signal	Dir	Description
Control I/F of dgIF typeS		
RstB	In	Synchronous reset signal. Active low. De-asserted to 1b when the Clk signal is stable
Clk	In	User clock source to both RAID0x4 and NVMe-IP. The frequency of this clock must be equal to or greater than 265.625 MHz for PCIe Gen5 clock frequency of 250 MHz.
UserCmd[2:0]	In	User Command. Valid when UserReq=1b. The possible values are 000b: Identify, 001b: Shutdown, 010b: Write SSDs, 011b: Read SSDs, 100b: SMART/Secure Erase, 110b: Flush, 101b/111b: Reserved
UserAddr[47:0]	In	Start address to write/read from the RAID0x4 is specified in 512byte units. However, since RAID0x4 is designed with a 4KB stripe size, the bits[2:0] are ignored and automatically set to 000b internally. It is valid when UserReq=1b.
UserLen[47:0]	In	The Total transfer size to write/read from the RAID0x4 is specified in 512byte units. However, since RAID0x4 is designed with a 4KB stripe size, the bits[2:0] are ignored and automatically set to 000b internally. It is valid from 8 to (LBASize-UserAddr).
UserReq	In	Asserts to 1b to send a new command request and de-asserts to 0b after the RAID0x4 initiates the operation by asserting UserBusy to 1b. This signal can only be asserted when the RAID0x4 is Idle (UserBusy=0b). Command parameter (UserCmd, UserAddr, UserLen, and CtmSubmDW0-DW15) must be valid and stable during UserReq being set to 1b. UserAddr and UserLen are inputs for the Write/Read command while CtmSubmDW0-DW15 are inputs for SMART/Secure Erase/Flush command.
UserBusy	Out	Asserted to 1b when the RAID0x4 is busy. A new request must not be sent (UserReq to 1b) while the RAID0x4 is busy.
LBASize[47:0]	Out	The total capacity returned by the RAID0x4 in 512-byte units. Default value is 0, and it becomes valid upon the completion of the Identify command. This value is set to be four times the LBASize output from NVMe-IP#0.
UserError	Out	Error flag. Asserted to 1b when the UserErrorType is not equal to 0. The flag is de-asserted to 0b by asserting RstB to 0b.
UserErrorType[0-3][31:0]	Out	Error status, directly mapped from UserErrorType in each NVMe-IP. [0]-IP#0, [1]-IP#1, [2]-IP#2, and [3]-IP#3.
Data I/F of dgIF typeS		
UserFifoWrCnt[15:0]	In	Write data counter of the Receive FIFO to monitor the FIFO's full status. When the FIFO is full, it pauses the transmission of returned data from the Read command. In cases where the size of the FIFO data count is less than 16 bits, the remaining upper bits must be filled with the value 1b.
UserFifoWrEn	Out	Asserted to 1b to write data to the Receive FIFO while executing the Read command.
UserFifoWrData[2047:0]	Out	Write data bus of the Receive FIFO. Valid when UserFifoWrEn=1b.
UserFifoRdCnt[15:0]	In	Read data counter of the Transmit FIFO to indicate the data size stored in the FIFO. When the FIFO is empty, data transmission is paused. If the FIFO data count is less than 16 bits, the remaining upper bits are padded with 0b.
UserFifoEmpty	In	Unused for this IP.
UserFifoRdEn	Out	Asserted to 1b to read data from the Transmit FIFO while executing the Write command.
UserFifoRdData[2047:0]	In	Read data returned from the Transmit FIFO. Valid in the next clock after UserFifoRdEn is asserted to 1b.

Timing diagram of RAID0x4 module when running Write command is shown as follows.

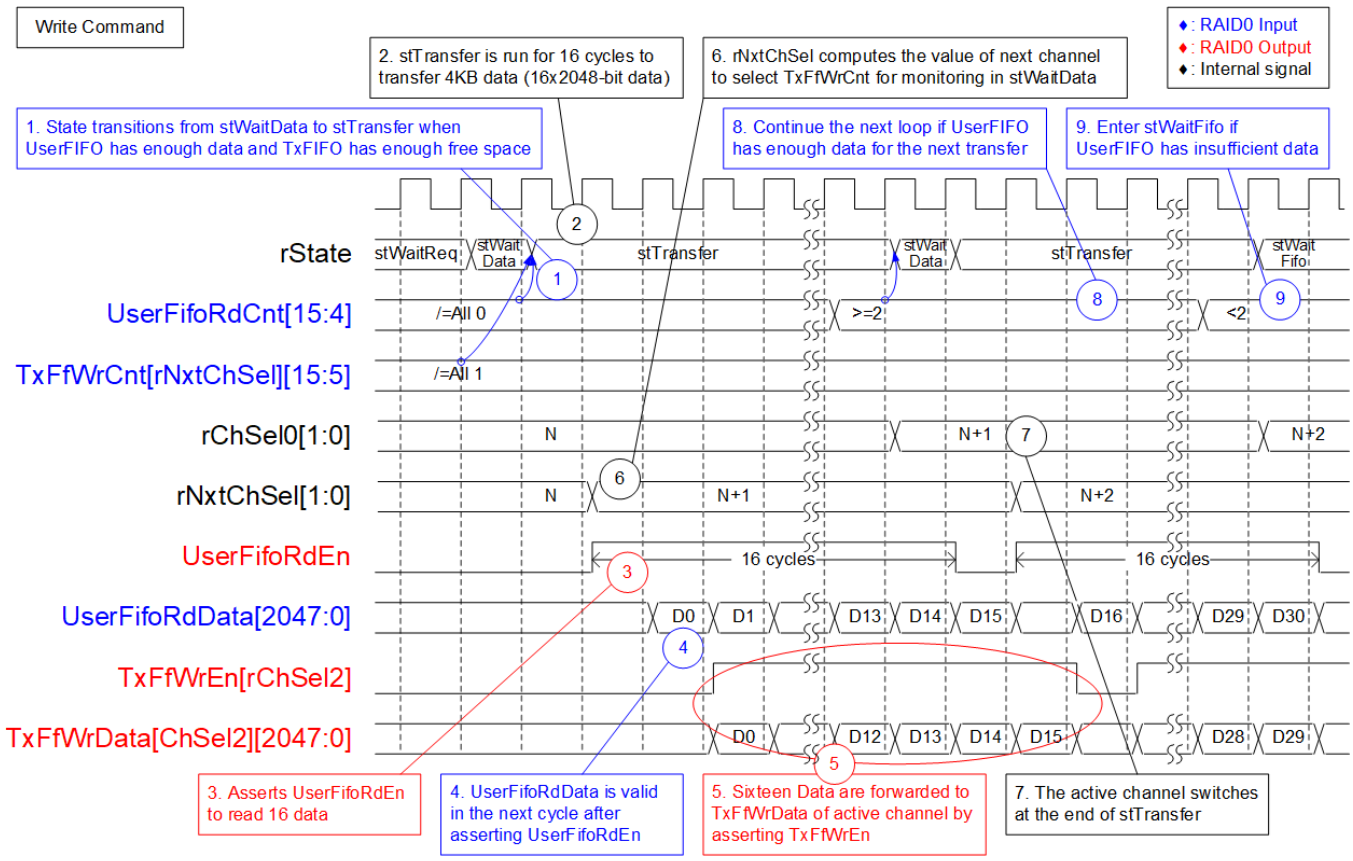

Figure 2-10 RAID0x4 timing diagram during executing Write command

When initiating a Write command to RAID0x4, data is transmitted from UserFIFO (U2IPFIFO) to TxFIFO[0]-[3]. Only one TxFIFO operates at a time to transfer 4Kbyte data. Following the RAID0 behavior, the active channel switches to the next channel upon completing the 4Kbyte data transfer. The procedure of executing the Write command is as follows.

1) The key state of the RAID0x4 module is stWaitData. Firstly, it checks the remaining transfer size. If the remaining transfer size is 0, the operation is finished. Otherwise, it monitors UserFifoRdCnt and TxFfWrCnt to ensure that at least 4Kbyte data is stored in U2IPFIFO and TxFIFO has at least 8Kbyte free space. If the FIFOs are ready, the write operation begins.
   Note:
   i) *TxFfWrCnt of four channels are fed to multiplexer to select the active channel. This introduces a one-clock latency compared to UserFifoRdCnt.*
   ii) *rNxtChSel controls TxFfWrCnt and indicates the next active channel for running in stTransfer. After starting the first transfer loop, rNxtChSel is incremented to monitor the free space in the FIFO of the next active channel.*
2) The state machine enters stTransfer to start forwarding write data from user logic to TxFIFO. This state remains active for 16 clock cycles.
3) During the stTransfer state, UserFifoRdEn is asserted to read 4Kbyte data from UserFIFO.

4) The read data (UserFifoRdData) becomes valid in the next cycle after UserFifoRdEn is asserted.
5) The data is forwarded to the TxFIFO of the active channel, selected by rChSel2, which is two-clock latency signal of rChSel0.
   Note: rChSel0 indicates the active channel for data transfer in the stTransfer state.
6) In stTransfer state, rNxtChSel is computed to determine the next active channel based on rChSel0. This result is then employed to read the TxFfWrCnt value from the selected channel in the stWaitData state.
7) After finishing 4Kbyte data transfer, the active channel for data transfer (rChSel0) is incremented.
8) To minimize the overhead time for initiating the next transfer, UserFifoRdCnt is monitored in advance during the stTransfer state, before the current data transfer is completed. If the read counter indicates at least two sets of 4Kbyte stored in the FIFO, the new transfer can commence in the next cycle by transitioning back to the stWaitData state and returning to step 1. Otherwise, the next state is stWaitFifo, as further described in step 9.
9) stWaitFIFO is designed to wait until the current data transfer is completed and UserFifoRdCnt becomes valid for monitoring. After waiting for three clock cycles, the state enters stWaitData to continue the next transfer or complete the operation.

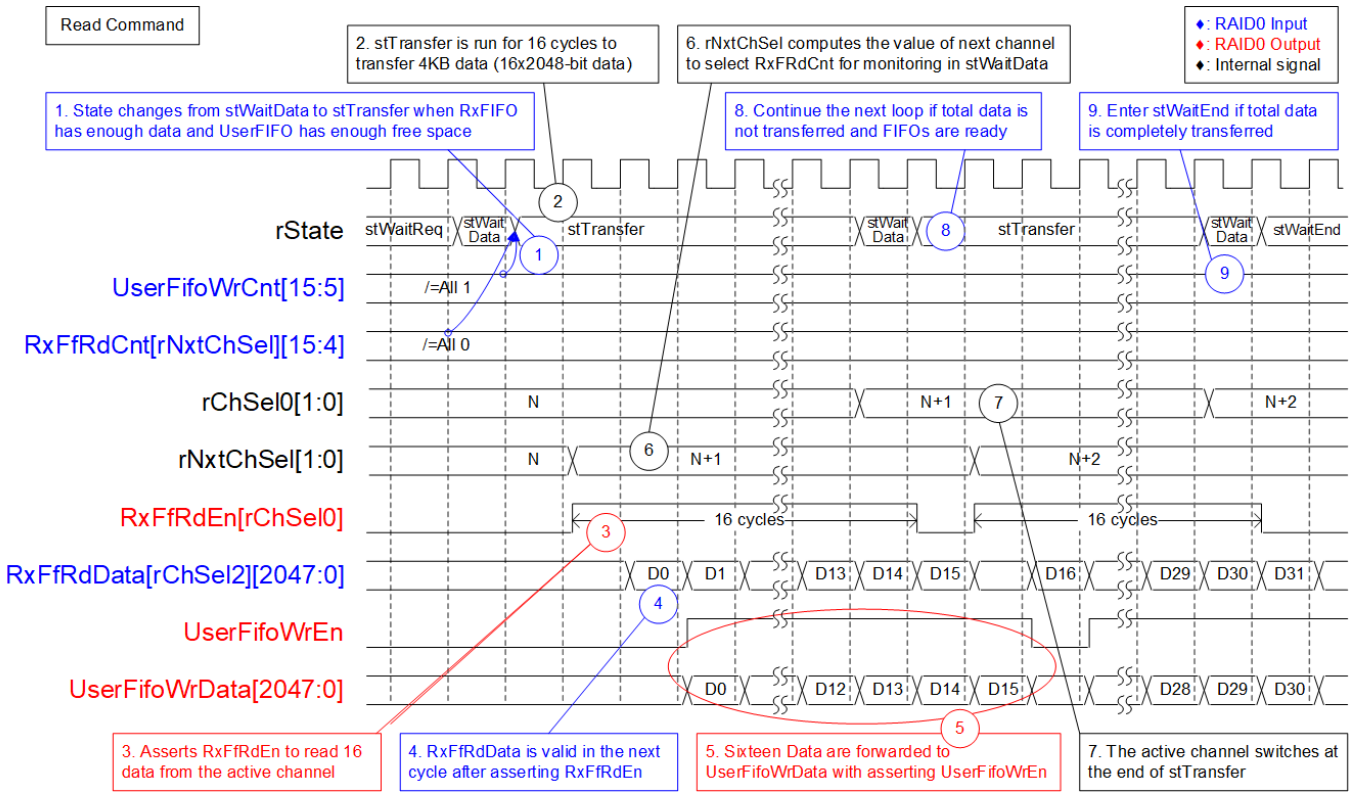

Figure 2-11 RAID0x4 timing diagram in Read command

When initiating a Read command for RAID0x4, data is forwarded from RxFIFO[0]-[3] to IP2UFIFO. Similar to the Write command, only one RxFIFO is active at a time to transfer 4Kbyte data. Following the RAID0 behavior, the active channel switches to the next channel after completing the transfer of 4Kbyte data. The operational sequence outlines as follows.

1) The state “stWaitData” is responsible for checking the remaining transfer length and two FIFO status signals: RxFfRdCnt and UserFifoWrCnt. It needs to ensure that at least 4Kbyte data is stored in RxFIFO and IP2UFIFO has at least 8Kbyte free space. If the FIFOs are ready and the remaining transfer size is not 0, the read operation begins.
   Note:
   i) RxFfRdCnt of four channels are fed to multiplexer to select the active channel. This introduces a one-clock latency compared to UserFifoWrCnt.
   ii) rNxtChSel controls RxFfRdCnt and indicates the next active channel for running in stTransfer.

2) The state machine enters stTransfer to start forwarding the read data from RxFIFO to user logic.
3) RxFfRdEn of the active channel, selected by rChSel0, is asserted to 1b for 16 clock cycles to read 4Kbyte data from RxFIFO.
4) Following the standard FIFO behavior, the Read data (RxFfRdData) becomes valid in the next cycle after asserting the Read enable (RxFfRdEn).
5) The data of the active channel, selected by rChSel2, is forwarded to the UserFIFO.
   Note: rChSel2 is ChSel0 signal with a two-clock latency.
6) In stTransfer state, rNxtChSel computes the next active channel based on rChSel0. This calucated value guides the selection of the active channel for reading RxFfRdCnt in advance during in stWaitData state.
7) After finishing 4Kbyte data transfer, the active channel for data transfer (rChSel0) is incremented.
8) If there is remaining transfer length, the state returns to stTransfer to start a new data transfer, similar to step 2. Otherwise, the next state is stWaitEnd, as described in step 9.
9) If all data has been completely transferred, the state enters stWaitEnd to wait until all devices finish their operation by de-asserting the Busy signal to 0b.

2.3 CPU and Peripherals

The CPU system uses a 32-bit Avalon-MM bus as the interface to access peripherals such as the Timer and JTAG UART. The system also integrates an additional peripheral to access NVMeRAID0x4IP test logic by assigning a unique base address and address range. To support CPU read and write operations, the hardware logic must comply with the Avalon-MM bus standard. Avl2Reg module, as shown in Figure 2-12, is designed to connect the CPU system via the Avalon-MM interface, in compliance with the standard.

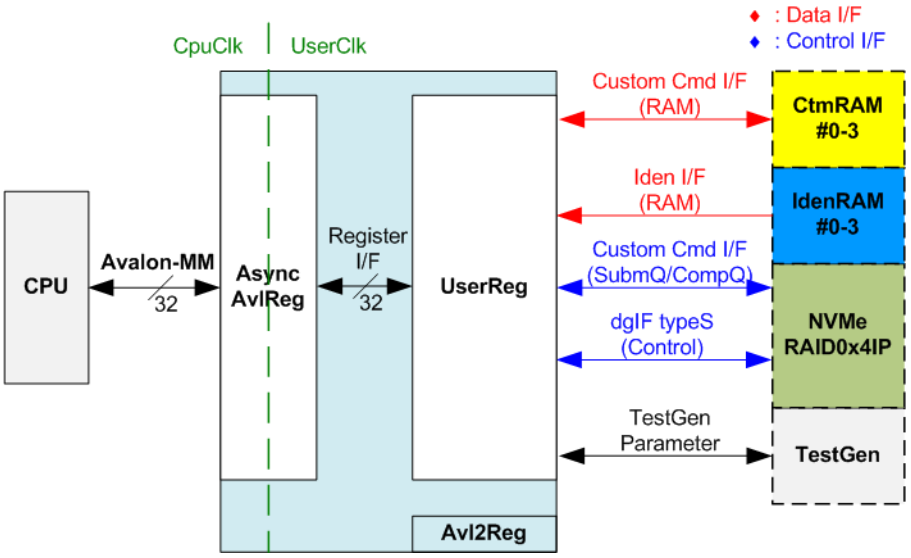

Figure 2-12 CPU and peripherals hardware

Avl2Reg consists of AsyncAvlReg and UserReg. AsyncAvlReg converts Avalon-MM signals into a simple Register interface with a 32-bit data bus size, similar to the Avalon-MM data bus size. It also includes asynchronous logic to handle clock domain crossing between the CpuClk and UserClk domains.

UserReg includes the register file of the parameters and the status signals of other modules in the test system, including the CtmRAMs, IdenRAMs, NVMeRAID0x4IP, and TestGen. More details of AsyncAvlReg and UserReg are explained below.

2.3.1 AsyncAvlReg

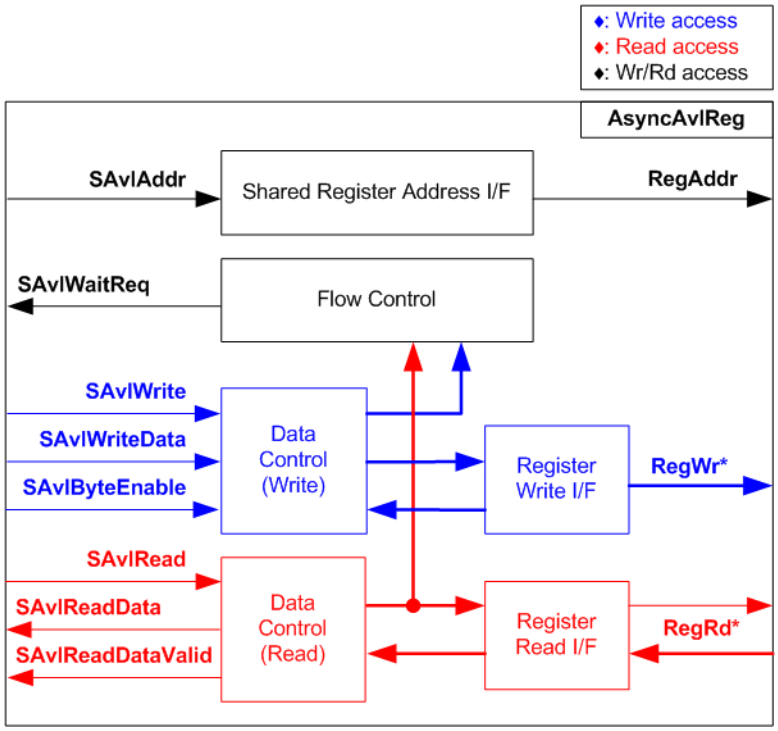

Figure 2-13 AsyncAxiReg Interface

The Avalon-MM bus interface signal can be grouped into three categories: Write channel (blue), Read channel (red), and Shared control channel (black). More details about the Avalon-MM interface specification can be found in the following document.
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

According to Avalon-MM specification, only one command (write or read) can be executed at a time. AsyncAvlReg's logic is divided into three groups: Write control logic, Read control logic, and Flow control logic. The flow control logic asserts SAvlWaitReq to hold the next request from the Avalon-MM interface if the current request has not finished. Write control and Write data I/F of the Avalon-MM bus are latched and transferred to the Write register interface with clock domain crossing registers. Similarly, Read control I/F are latched and transferred to be Read register interface. Afterward, the data returned from Register Read I/F is transferred to Avalon-MM bus with using clock domain crossing registers. The Address I/F of Avalon-MM is also latched and transferred to the Address register interface.

The Register interface is compatible with the single-port RAM interface for write transaction. However, the read transaction of the Register interface is slightly modified from RAM interface by adding RdReq and RdValid signals to control the read latency time. Since the address of the Register interface is shared for write and read transactions, the user cannot write and read the register simultaneously. The timing diagram of the Register interface is shown in Figure 2-14.

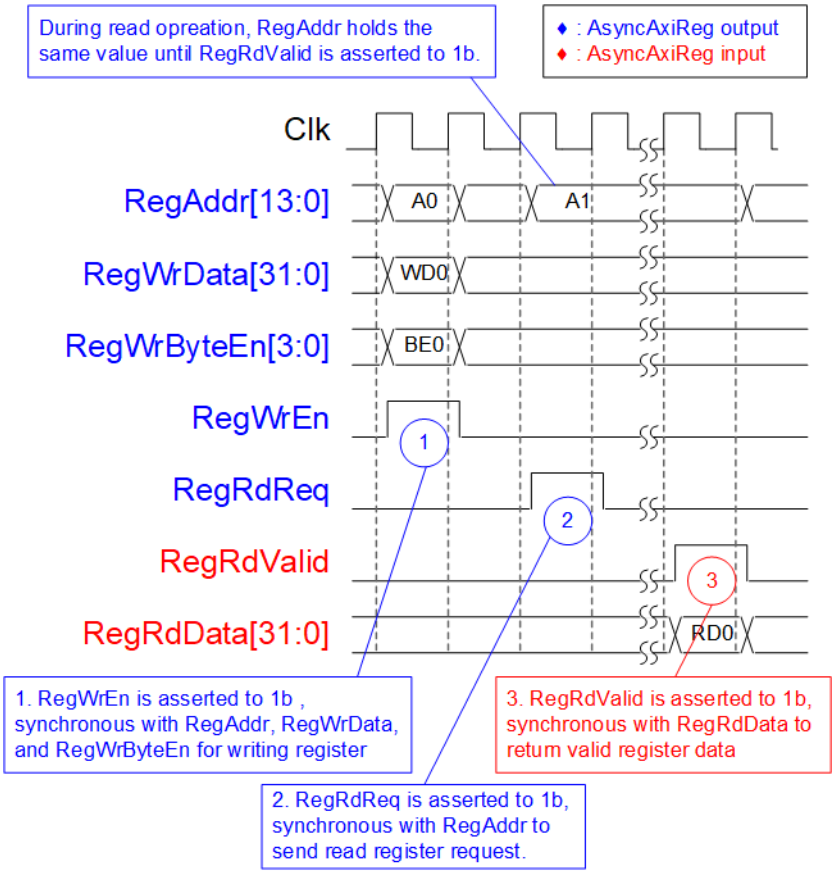

Figure 2-14 Register interface timing diagram

1) Timing diagram to write register is similar to that of a single-port RAM. The RegWrEn signal is set to 1b, along with a valid RegAddr (Register address in 32-bit units), RegWrData (write data for the register), and RegWrByteEn (write byte enable). The byte enable consists of four bits that indicate the validity of the byte data. For example, bit[0], [1], [2], and [3] are set to 1b when RegWrData[7:0], [15:8], [23:16], and [31:24] are valid, respectively.
2) To read register, AsyncAvlReg sets the RegRdReq signal to 1b with a valid value for RegAddr. The 32-bit data is returned after the read request is received. The slave detects the RegRdReq signal being set to start the read transaction. In the read operation, the address value (RegAddr) remains unchanged until RegRdValid is set to 1b. The address can then be used to select the returned data using multiple layers of multiplexers.
3) The slave returns the read data on RegRdData bus by setting the RegRdValid signal to 1b. After that, AsyncAvlReg forwards the read value to the SAvlRead interface.

2.3.2 UserReg

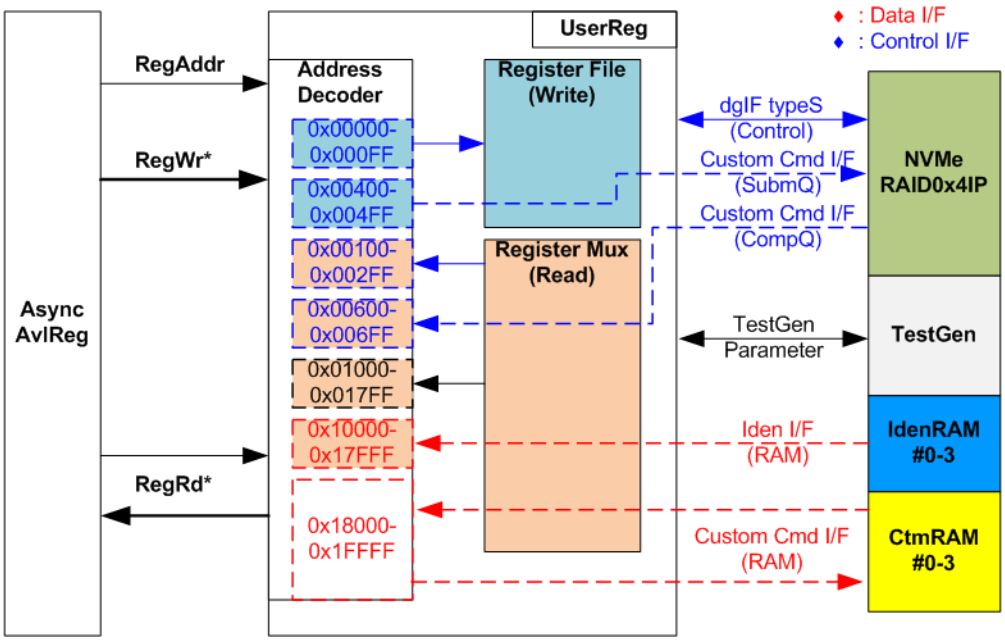

Figure 2-15 UserReg Interface

The UserReg module consists of an Address decoder, a Register File, and a Register Mux. The Address decoder decodes the address requested by AsyncAvlReg and selects the active register for either write or read transactions. The assigned address range in UserReg is divided into six areas, as shown in Figure 2-15.

1) 0x00000 – 0x000FF: mapped to set the command with the parameters of TestGen and NVMeRAID0x4IP. This area is write-access only.
2) 0x00400 – 0x004FF: mapped to set the parameters for the Custom command interface of NVMeRAID0x4IP. This area is write-access only.
3) 0x00100 – 0x002FF: mapped to read the status signals of NVMeRAID0x4IP. This area is read-access only.
4) 0x00600 – 0x006FF: mapped to read the status of Custom command interface (NVMeRAID0x4IP). This area is read-access only.
5) 0x01000 – 0x017FF: mapped to read the status signals of TestGen. This area is read-access only.
6) 0x10000 – 0x17FFF: mapped to read data from IdenRAM. This area is read-access only.
7) 0x18000 – 0x1FFFF: mapped to write or read data using Custom command RAM interface. This area allows both write-access and read access. However, the demo shows only read-access by running the SMART command.

The Address decoder decodes the upper bits of RegAddr to select the active hardware (NVMeRAID0x4IP, TestGen, IdenRAM, or CtmRAM). The Register File within UserReg has a 32-bit bus size, so the write byte enable (RegWrByteEn) is not used in the test system, and the CPU uses a 32-bit pointer to set the hardware register.

To read the register, multi-level multiplexers (mux) select the data to return to the CPU by using the address. The lower bits of RegAddr are fed to the submodule to select the active data within each submodule. While the upper bits are used in UserReg to select the submodule to read the returned data. The latency time of read data equals three clock cycles, so RegRdValid is created by RegRdReq, with three D Flip-flops asserted. More details of the address mapping within the UserReg module are shown in Table 2-3.

Table 2-3 Register Map

Address Wr/Rd	Register Name (Label in “nvmeraid0g5test.c”)	Description
0x00000 – 0x000FF: Control signals of NVMeRAID0x4IP and TestGen (Write access only)		
BA+0x00000	User Address (Low) Reg (USRADRL_INTREG)	[31:0]: Input to be bit[31:0] of start address in 512-byte unit (UserAddr[31:0] of dgIF typeS for NVMeRAID0x4IP)
BA+0x00004	User Address (High) Reg (USRADRH_INTREG)	[15:0]: Input to be bit[47:32] of start address in 512-byte unit (UserAddr[47:32] of dgIF typeS for NVMeRAID0x4IP)
BA+0x00008	User Length (Low) Reg (USRLENL_INTREG)	[31:0]: Input to be bit[31:0] of transfer length in 512-byte unit (UserLen[31:0] of dgIF typeS for NVMeRAID0x4IP)
BA+0x0000C	User Length (High) Reg (USRLENH_INTREG)	[15:0]: Input to be bit[47:32] of transfer length in 512-byte unit (UserLen[47:32] of dgIF typeS for NVMeRAID0x4IP)
BA+0x00010	User Command Reg (USRCMD_INTREG)	[2:0]: Input to be user command (UserCmd of dgIF typeS for NVMeRAID0x4IP) 000b: Identify, 001b: Shutdown, 010b: Write RAID, 011b: Read RAID, 100b: SMART/Secure Erase, 110b: Flush, 101b/111b: Reserved. When this register is written, the command request is sent to NVMeRAID0x4IP to start the operation.
BA+0x00014	Test Pattern Reg (PATTSEL_INTREG)	[2:0]: Select test pattern 000b-Increment, 001b-Decrement, 010b-All 0, 011b-All 1, 100b-LFSR
BA+0x00020	NVMe Timeout Reg (NVMTIMEOUT_INTREG)	[31:0]: Input to be timeout value of all NVMe-IPs (TimeOutSet[31:0] of NVMe-IP)
0x00100 – 0x002FF: Status signals of NVMeRAID0x4IP (Read access only)		
BA+0x0100	User Status Reg (USRSTS_INTREG)	[0]: UserBusy of dgIF typeS for NVMeRAID0x4IP (0b: Idle, 1b: Busy) [1]: UserError of dgIF typeS for NVMeRAID0x4IP (0b: Normal, 1b: Error) [2]: Data verification fail in TestGen (0b: Normal, 1b: Error)
BA+0x0104	Total device size (Low) Reg (LBASIZEL_INTREG)	[31:0]: Mapped to LBASize[31:0] of NVMeRAID0x4IP
BA+0x0108	Total device size (High) Reg (LBASIZEH_INTREG)	[15:0]: Mapped to LBASize[47:32] of NVMeRAID0x4IP
(BA+0x0120)- (BA+0x012F)	User Error Type CH#0-#3 Reg (USRERRTYPECH0-3_INTREG)	[31:0]: Mapped to UserErrorType of NVMe-IP 0x0120: NVMe-IP#0, 0x0124: NVMe-IP#1, 0x0128: NVMe-IP#2, 0x012C: NVMe-IP#3
(BA+0x0140)- (BA+0x014F)	PCIe Status CH#0-#3 Reg (PCIESTSCH0-3_INTREG)	[0]: PCIe linkup status from PCIe hard IP (0b: No linkup, 1b: linkup) [3:2]: Two lower bits to show PCIe link speed. *Note: Two upper bits are [17:16].* (0000b: Not linkup, 0001b: PCIe Gen1, 0010b: PCIe Gen2, 0011b: PCIe Gen3, 0111b: PCIe Gen4, 1111b: PCIe Gen5) [6:4]: PCIe link width status from PCIe hard IP (001b: 1-lane, 010b: 2-lane, 100b: 4-lane) [13:8]: Current LTSSM State of PCIe hard IP. Please see more details of LTSSM value in Avalon-ST PCIe Hard IP datasheet [17:16]: Two upper bit to show PCIe link speed. *Note: Two lower bits are [3:2].* 0x0140: NVMe-IP#0, 0x0144: NVMe-IP#1, 0x0148: NVMe-IP#2, 0x014C: NVMe-IP#3
(BA+0x0160)- (BA+0x016F)	Completion Status CH#0-#3 Reg (COMPSTSCH0-3_INTREG)	[15:0]: Status from Admin completion (AdmCompStatus[15:0] of NVMe-IP) [31:16]: Status from I/O completion (IOCompStatus[15:0] of NVMe-IP) 0x0160: NVMe-IP#0, 0x0164: NVMe-IP#1, 0x0168: NVMe-IP#2, 0x016C: NVMe-IP#3
(BA+0x0180)- (BA+0x018F)	NVMe CAP CH#0-#3 Reg (NVMCAPCH0-3_INTREG)	[31:0]: Mapped to NVMeCAPReg[31:0] of NVMe-IP 0x0180: NVMe-IP#0, 0x0184: NVMe-IP#1, 0x0188: NVMe-IP#2, 0x018C: NVMe-IP#3

Address Wr/Rd	Register Name (Label in "nvmeraid0g5test.c")	Description
0x00100 – 0x002FF: Status signals of NVMeRAID0x4IP (Read access only)		
(BA+0x00200)- (BA+0x0020F)	NVMe IP Test pin CH#0 Reg (NVMTESTPIN0-3CH0_INTREG)	Mapped to TestPin[127:0] of NVMe-IP #0 0x0200: Bit[31:0], 0x204: Bit[63:32], 0x0208: Bit[95:64], 0x20C: Bit[127:96]
(BA+0x00210)- (BA+0x0023F)	NVMe IP Test pin CH#1–#3 Reg	0x0210 – 0x021F: TestPin[127:0] of NVMe-IP#1 0x0220 – 0x022F: TestPin[127:0] of NVMe-IP#2 0x0230 – 0x023F: TestPin[127:0] of NVMe-IP#3
0x00400 – 0x005FF: Custom command of NVMeRAID0x4IP		
(BA+0x00400)- (BA+0x004FF) Wr	Custom Submission Queue CH#0–#3 Reg (CTMSUBMQ0-3_STRUCT)	[31:0]: Submission queue entry of SMART, Flush, or Secure Erase command. Input to be CtmSubmDW0-DW15 of NVMe-IP#0-#3. 0x400: DW0, 0x404: DW1, …, 0x43C: DW15 of NVMe-IP#0 0x440: DW0, 0x444: DW1, …, 0x47C: DW15 of NVMe-IP#1 0x480: DW0, 0x484: DW1, …, 0x4BC: DW15 of NVMe-IP#2 0x4C0: DW0, 0x4C4: DW1, …, 0x4FC: DW15 of NVMe-IP#3
(BA+0x00600)- (BA+0x0063F) Rd	Custom Completion Queue CH#0–#3 Reg (CTMCOMPQ0-3_STRUCT)	[31:0]: CtmCompDW0-DW3 output from NVMe-IP#0-#1 successively. 0x600: DW0, 0x604: DW1, …, 0x60C: DW3 of NVMe-IP#0 0x610: DW0, 0x614: DW1, …, 0x61C: DW3 of NVMe-IP#1 0x620: DW0, 0x624: DW1, …, 0x62C: DW3 of NVMe-IP#2 0x630: DW0, 0x634: DW1, …, 0x63C: DW3 of NVMe-IP#3
BA+0x00800 Rd	IP Version Reg (IPVERSION_INTREG)	[31:0]: Mapped to IPVersion[31:0] of NVMe-IP
0x01000 – 0x017FF: Status signals of TestGen (Read access only)		
BA+0x01000	Data Failure Address(Low) Reg (RDFAILNOL_INTREG)	[31:0]: Bit[31:0] of the byte address of the 1st failure when executing a Read command
BA+0x01004	Data Failure Address(High) Reg (RDFAILNOH_INTREG)	[24:0]: Bit[56:32] of the byte address of the 1st failure when executing a Read command
BA+0x01008	Current test byte (Low) Reg (CURTESTSIZEL_INTREG)	[31:0]: Bit[31:0] of the current test data size in TestGen module
BA+0x0100C	Current test byte (High) Reg (CURTESTSIZEH_INTREG)	[24:0]: Bit[56:32] of the current test data size in TestGen module
BA+0x01200	Expected value Word0 Reg (EXPPATW0_INTREG)	2048-bit of the expected data at the 1st failure data in Read command 0x1200: Bit[31:0], 0x1204: Bit[63:32], …, 0x12FC: Bit[2047:2016]
(BA+0x01204)- (BA+0x012FC)	Expected value Word1-63 Reg	
BA+0x01400	Read value Word0 Reg (RDPATW0_INTREG)	2048-bit of the read data at the 1st failure data in Read command 0x1400: Bit[31:0], 0x1404: Bit[63:32], …, 0x14FC: Bit[2047:2016]
(BA+0x01404)- (BA+0x014FC)	Read value Word1-63 Reg	
0x10000 – 0x1FFFF: Identify RAM and Custom RAM		
(BA+0x10000) – (BA+0x17FFF) Rd	Identify Controller Data CH#0 Identify Namespace Data CH#0 IDENCTRL/NAME0_CHARREG	0x10000-0x10FFF: 4Kbyte Identify controller data of NVMe-IP#0 0x11000-0x11FFF: 4Kbyte Identify namespace data of NVMe-IP#0
(BA+0x12000) – (BA+0x17FFF)	Identify Controller and Namespace Data CH#1-3	0x12000-0x13FFF: NVMe-IP#1, 0x14000-0x15FFF: NVMe-IP#2, 0x16000-0x17FFF: NVMe-IP#3
(BA+0x18000) – (BA+0x1FFFF) Wr/Rd	Custom command RAM CH#0-#3 (CTMRAM0-3_CHARREG)	8K byte CtmRAM interface of NVMe-IP#0-#3 for storing 512-byte data output from SMART command. 0x18000-0x19FFF: NVMe-IP#0, 0x1A000-0x1BFFF: NVMe-IP#1, 0x1C000-0x1DFFF: NVMe-IP#2, 0x1E000-0x1FFFF: NVMe-IP#3

3 CPU Firmware

3.1 Test firmware (nvmeraid0g5test.c)

The CPU follows these steps upon system startup to complete the initialization process.

1) Initializes JTAG UART and Timer parameters.
2) Waits until the PCIe connection becomes active (PCIESTSCH0-3_INTREG[0]= 1b).
3) Waits until NVMeRAID0x4IP completes its own initialization process (USRSTS_INTREG[0]=0b). The error message is displayed and the process stops when some errors are found.
4) CPU displays PCIe link status (the number of PCIe lanes and the PCIe speed) of each channel by reading PCIESTSCH0-3_INTREG[17:2].
5) CPU displays the main menu. There are seven menus for running seven commands of RAID0 module, i.e., Identify, Write, Read, SMART, Flush, Secure Erase, and Shutdown.

More details of the operation flow in each command are described as follows.

3.1.1 Identify command

The sequence for the firmware when the Identify command is selected by user is as follows.

1) Set USRCMD_INTREG[2:0]=000b to send the Identify command request to NVMeRAID0x4IP. The busy flag (USRSTS_INTREG[0]) will then change from 0b to 1b.
2) The CPU waits until the operation is completed or an error is detected by monitoring USRSTS_INTREG[1:0].
   - If bit[0] is de-asserted to 0b after the operation is finished, the data of Identify command returned by NVMe-IP will be stored in IdenRAM.
   - If bit[1] is asserted to 1b, indicating an error, the error message will be displayed on the console with details decoded from USRERRTYPECH0-3_INTREG[31:0]. The process will then stop.
3) Once the busy flag (USRSTS_INTREG[0]) is de-asserted to 0b, the CPU will display decoded information from IdenRAM (IDENCTRL0-3_CHARREG). This information includes the model name of SSD#0-SSD#3 and the device details obtained from the output signals of NVMeRAID0x4, such as the device capacity (LBASIZEL/H_INTREG), and support for Secure Erase command.

3.1.2 Write/Read command

The sequence for the firmware when the Write/Read command is selected is as follows.

1) Receive start address, transfer length, and test pattern from JTAG UART. If any inputs are invalid, the operation will be cancelled.
2) After obtaining all the inputs, set them to USRADRL/H_INTREG, USRLENL/H_INTREG, and PATTSEL_INTREG.
3) To execute the Write or Read command, set USRCMD_INTREG[2:0]= 010b or 011b, respectively. This sends the command request to the NVMeRAID0x4IP. Once the command is issued, the busy flag of NVMeRAID0x4IP (USRSTS_INTREG[0]) will change from 0b to 1b.
4) The CPU waits until the operation is completed or an error (excluding verification error) is detected by monitoring USRSTS_INTREG[2:0].
   - If bit[0] is de-asserted to 0b after the operation is finished, the CPU will then proceed to the next step.
   - If bit[1] is asserted to 1b, indicating an error, the error message will be displayed on the console with details decoded from USRERRTYPECH0-3_INTREG[31:0]. The process will then stop.
   - If bit[2] is asserted to 1b, indicating a data verification fails, the verification error message will be displayed on the console, but the CPU will continue to run until the operation is completed or the user cancels the operation by pressing any key.

   While the command is running, the current transfer size, read from CURTESTSIZEL/H_INTREG, will be displayed every second.

5) After the busy flag (USRSTS_INTREG[0]) is de-asserted to 0b, CPU will calculate and display the test result on the console including the total time usage, total transfer size, and transfer speed.

3.1.3 SMART Command,

The sequence of the firmware when the SMART command is selected is as follows.

1) The 16-Dword of Submission Queue entry (CTMSUBMQ0-3_STRUCT) is set to the SMART command value.
2) Set USRCMD_INTREG[2:0]=100b to send the SMART command request to NVMeRAID0x4IP. The busy flag (USRSTS_INTREG[0]) will then change from 0b to 1b.
3) The CPU waits until the operation is completed or an error is detected by monitoring USRSTS_INTREG[1:0].
   - If bit[0] is de-asserted to 0b after the operation is finished, the data of SMART command returned by NVMe IP will be stored in CtmRAM.
   - If bit[1] is asserted to 1b, indicating an error, the error message will be displayed on the console with details decoded from USRERRTYPECH0-3_INTREG[31:0]. The process will then stop.

4) After the busy flag (USRSTS_INTREG[0]) is de-asserted to 0b, the CPU will display information decoded from CtmRAM (CTMRAM0-3_CHARREG), i.e., Remaining Life, Percentage Used, Temperature, Total Data Read, Total Data Written, Power On Cycles, Power On Hours, and Number of Unsafe Shutdown.

   For more information on the SMART log, refer to the NVM Express Specification. https://nvmexpress.org/resources/specifications/

3.1.4 Flush Command

The sequence of the firmware when the Flush command is selected is as follows.

1) The 16-Dword of Submission Queue entry (CTMSUBMQ0-3_STRUCT) is set to the Flush command value.
2) Set USRCMD_INTREG[2:0]=110b to send Flush command request to NVMeRAID0x4IP. The busy flag of NVMeRAID0x4IP (USRSTS_INTREG[0]) will then change from 0b to 1b
3) The CPU waits until the operation is completed or an error is detected by monitoring USRSTS_INTREG[1:0].
   - If bit[0] is de-asserted to 0b after the operation is finished, the CPU will then return to the main menu.
   - If bit[1] is asserted to 1b, indicating an error, the error message will be displayed on the console with details decoded from USRERRTYPECH0-3_INTREG[31:0]. The process will then stop.

3.1.5 Secure Erase Command

The sequence of the firmware when the Secure Erase command is selected is as follows.

1) The 16-Dword of Submission Queue entry (CTMSUBMQ0-3_STRUCT) is set to the Secure Erase command value.
2) Set NVMTIMEOUT_INTREG to 0 to disable timer to prevent the timeout error.
3) Set USRCMD_INTREG[2:0]=100b to send Secure Erase command request to NVMeRAID0x4IP. The busy flag of NVMeRAID0x4IP (USRSTS_INTREG[0]) will then change from 0b to 1b.
4) The CPU waits until the operation is completed or an error is detected by monitoring USRSTS_INTREG[1:0].
   - If bit[0] is de-asserted to 0b after the operation is finished, the CPU will then proceed to the next step.
   - If bit[1] is asserted to 1b, indicating an error, the error message will be displayed on the console with the details decoded from USRERRTYPECH0-3_INTREG[31:0]. The process will then stop.

5) After completing the command, the timer is re-enabled to generate timeout error in NVMe-IP by setting NVMTIMEOUT_INTREG to the default value.

3.1.6 Shutdown Command

The sequence of the firmware when the Shutdown command is selected is as follows.

1) Set USRCMD_INTREG[2:0]=001b to send the Shutdown command request to NVMeRAID0x4IP. The busy flag of NVMeRAID0x4IP (USRSTS_INTREG[0]) will then change from 0b to 1b.
2) The CPU waits until the operation is completed or an error is detected by monitoring USRSTS_INTREG[1:0].
   - If bit[0] is de-asserted to 0b after the operation is finished, the CPU will then proceed to the next step.
   - If bit[1] is asserted to 1b, indicating an error, the error message will be displayed on the console with details decoded from USRERRTYPECH0-3_INTREG[31:0]. The process will then stop.

3) After Shutdown command completes, all four SSDs and all NVMe-IPs will become inactive and the CPU will be unable to receive any new commands from the user. To continue testing, the user must power off and power on the system.

3.2 Function list in Test firmware

int exec_ctm(unsigned int user_cmd)	
Parameters	user_cmd: 4-SMART/Secure Erase command, 6-Flush command
Return value	0: No error, -1: Some errors are found in NVMeRAID0x4IP
Description	Execute SMART command as outlined in section 3.1.3 (SMART Command,), execute Flush command as outlined in section 3.1.4 (Flush Command), or execute Secure Erase command as outlined in section 3.1.5 (Secure Erase Command).

unsigned long long get_cursize(void)	
Parameters	None
Return value	Read value of CURTESTSIZEH/L_INTREG
Description	Read the value of CURTESTSIZEH/L_INTREG and return it as the result of the function.

int get_param(userin_struct* userin)	
Parameters	userin: Three inputs from user, i.e., start address, total 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 iden_dev(void)	
Parameters	None
Return value	None
Description	Execute Identify command as outlined in section 3.1.1 (Identify command).

int setctm_erase(void)	
Parameters	None
Return value	0: No error, -1: Some errors are found in NVMeRAID0x4IP
Description	Set Secure Erase command to CTMSUBMQ0-3_STRUCT and call exec_ctm function to execute Secure Erase command.

int setctm_flush(void)	
Parameters	None
Return value	0: No error, -1: Some errors are found in NVMeRAID0x4IP
Description	Set Flush command to CTMSUBMQ0-3_STRUCT and call exec_ctm function to execute Flush command.

int setctm_smart(void)	
Parameters	None
Return value	0: No error, -1: Some errors are found in NVMeRAID0x4IP
Description	Set SMART command to CTMSUBMQ0-3_STRUCT and call exec_ctm function to execute SMART command. Finally, decode and display SMART information on the console.

void show_error(void)	
Parameters	None
Return value	None
Description	Read USRERRTYPECH0-3_INTREG, decode the error flag, and display the corresponding error message. Also, call show_pciestat function to check the hardware's debug signals.

void show_pciestat(unsigned int channel)	
Parameters	channel: Select channel to display PCIe status on the console
Return value	None
Description	Read PCIESTSCH<channel>_INTREG until the read value from two read times is stable. After that, display the read value on the console. Also, debug signal is read by (NVMTESTPIN0-3CH0_INTREG + <channel>*channel offset).

void show_result(void)	
Parameters	None
Return value	None
Description	Print total size by calling get_cursize and show_size functions. After that, compute total time usage from global parameters (timer_val and timer_upper_val) and display the result 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 or Gbyte unit. The displayed format allows for representing data sizes with up to 6 digits and 3 decimal places. This means that the maximum value that can be shown is 999,999.999 GB.

void show_smart_hex16byte(volatile unsigned char *char_ptr)	
Parameters	*char_ptr: Pointer of 16-byte SMART data
Return value	None
Description	Display 16-byte SMART data as hexadecimal unit.

void show_smart_int8byte(volatile unsigned char *char_ptr)	
Parameters	*char_ptr: Pointer of 8-byte SMART data
Return value	None
Description	When the input value is less than 4 billion (32-bit), the 8-byte SMART data is displayed in decimal units. If the input value exceeds this limit, an overflow message is displayed.

void show_smart_size8byte(volatile unsigned char *char_ptr)	
Parameters	*char_ptr: Pointer of 8-byte SMART data
Return value	None
Description	Display 8-byte SMART data as GB or TB unit. When the input value is more than limit (500 PB), the overflow message is displayed instead.

void show_vererr(void)	
Parameters	None
Return value	None
Description	Read RDFAILNOL/H_INTREG (error byte address), EXPPATW0_INTREG (expected value), and RDPATW0_INTREG (read value) to display verification error details on the console.

void shutdown_dev(void)	
Parameters	None
Return value	None
Description	Execute Shutdown command as outlined in section 3.1.6 (Shutdown Command)

int wrrd_dev(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	Execute Write command or Read command as outlined in section 3.1.2 (Write/Read command). Call 'show_result' function to compute and display transfer performance in Write/Read command.

4 Example Test Result

Figure 4-1 illustrates the result of executing the RAID0x4 demo, using four SSDs: the 2TB CFD Gaming PG5NFZ SSD, the 2TB AORUS Gen5 10000 SSD and the two 2TB Crucial T700. Since all four SSDs utilize the same controller, their characteristics are expected to be similar. The result was measured using Write and Read commands, utilizing an LFSR test data pattern and a transfer size of 512 GB.

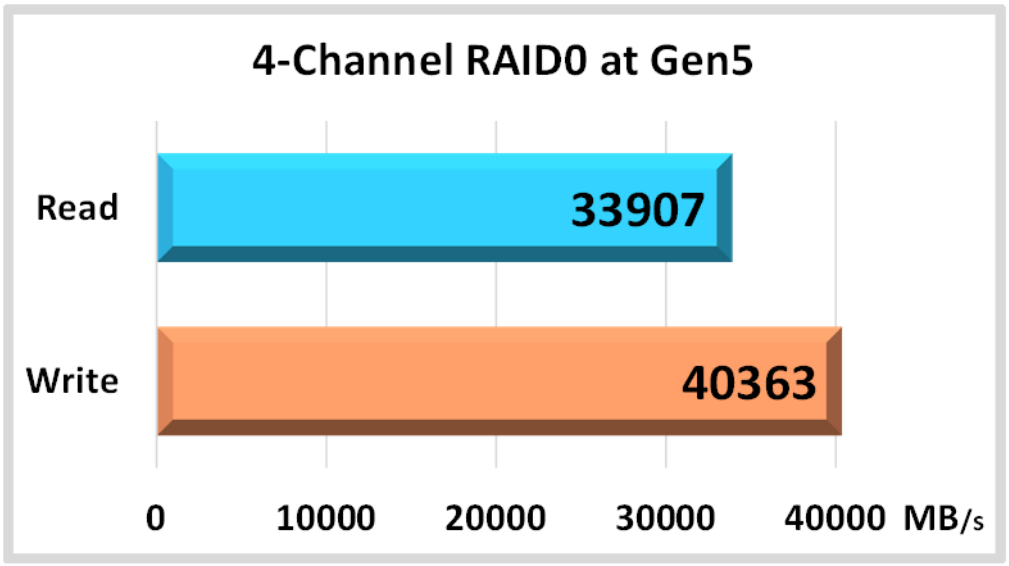

Figure 4-1 Performance of 4-ch RAID0 demo by using four Gen5 SSDs

Utilizing the Agilex 7 I-Series board with PCIe Gen5, the system demonstrates remarkable performance levels. The write speed reaches approximately 40,000 Mbyte/sec, while the read speed achieves around 34,000 Mbyte/sec.

5 Revision History

Revision	Date	Description
1.0	2-Oct-23	Initial version release