2-ch RAID0 Design (NVMe-IP) reference design manual

Rev1.1 5-Jun-18

1 Introduction

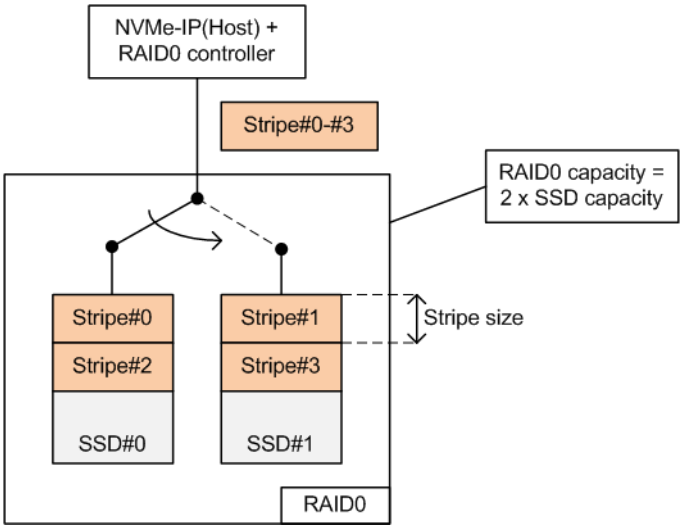

Figure 1-1 RAID0 by 2 SSDs data format

RAID0 system uses multiple storages to extend total storage capacity and increase write/read performance. Assumed that total number of device is N, total storage capacity is equal to N multiply by amount of storage. Write and read speed of RAID0 are almost equal to N multiply by speed of one SSD.

Data format of RAID0 is shown in Figure 1-1. Data stream of the host side are split into a small stripe and transfer to one SSD at a time. Stripe size is the data size to store in one SSD before switching to other SSDs.

In the reference design, two SSDs are applied to run RAID0 system. Stripe size is equal to 512 bytes (one sector unit). Two SSDs connecting in the system should use same model to match characteristic and get the best performance and capacity. By using RAID0 design, the total capacity is equal to two times of one SSD and the performance for both write and read are almost two times of one SSD performance. In our test system, Write speed of RAID0 by NVMe-IP is about 4200 MB/s and Read speed is about 6200 MB/s. (Performance of one NVMe-IP demo by using one SSD are 2100 MB/s for Write command and 3200 MB/s for Read command).

The demo does not include DDR and uses only FIFO implemented by BlockRAM to be the buffer. Test performance in the demo is average speed. The system is suitable for ultra-speed data recording application which has flow control to pause data transferring when SSD is not ready to transfer data. User can modify RAID0 reference design to increase the numbers of NVMe SSD to achieve the better performance and bigger disk capacity. Also, user can add DDR to be data buffer in the system when high performance as sustain rate is required.

Before running the reference design, it is recommended to study NVMe-IP datasheet and single channel demo firstly from following link.
http://www.dgway.com/products/IP/NVMe-IP/dg_nvme_ip_data_sheet_en.pdf
http://www.dgway.com/products/IP/NVMe-IP/dg_nvmeip_refdesign_en.pdf
http://www.dgway.com/products/IP/NVMe-IP/dg_nvmeip_instruction_en.pdf

2 Hardware overview

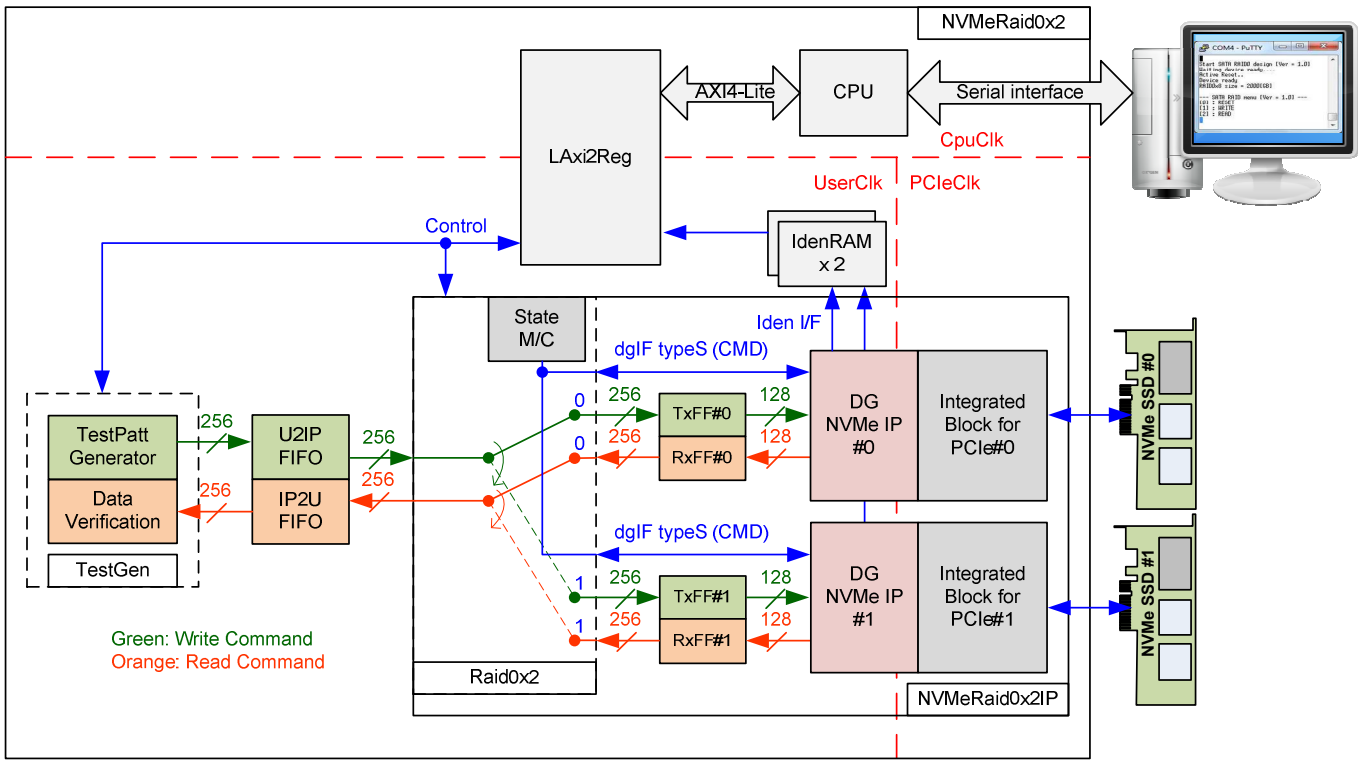

Figure 2-1 RAID0x2 demo system by using NVMe-IP

The hardware system is split into three groups following the interface i.e.

1) TestGen: The example of user logic to write and read data in this reference design is TestGen module. TestGen module generates test data to U2IPFIFO at the highest speed with flow control in Write command. For Read command, TestGen reads and verifies test data from IP2UFIFO at the highest speed with flow control. TestGen uses 256-bit data bus and runs in UserClk domain which is equal to 275 MHz, so maximum bandwidth of TestGen is 8800 MB/s which is more than maximum performance of two NVMe SSDs.
2) RAID: Two NVMe-IPs are applied to control two NVMe SSDs. Raid0x2 arranges data stored in two SSDs as RAID0 format. Raid0x2 decodes the current address to select one SSD to transfer data with TestGen. Data bus size of RAID0 is 256-bit which is two times of NVMe-IP to match data bandwidth of two SSDs. Two FIFO sets (TxFF and RxFF) are connected between Raid0x2 and NVMe-IP to convert data bus size.
3) CPU: Test operation in the demo is controlled by user through Serial console. CPU firmware is designed to receive test parameters and the command from user. CPU sets parameters to the hardware through AXI4-Lite bus. LAxi2Reg has the register sets of test parameters which are mapped to different address of CPU. LAxi2Reg decodes the address of AXI4-Lite bus to select the active parameter. For write access, Write data from AXI4-Lite bus is set to the selected parameter following the address. For read access, Read data from selected parameter is returned to AXI4-Lite bus. Read access is applied for CPU monitoring and displaying the hardware status to the user through Serial console.

   More details of the hardware are described as follows.

2.1 TestGen

This module is designed to generate Test pattern to WrFf in Write command or reads data from RdFf to verify in Read command at the fastest speed to check system performance. The details of hardware inside TestGen are shown in Figure 2-2.

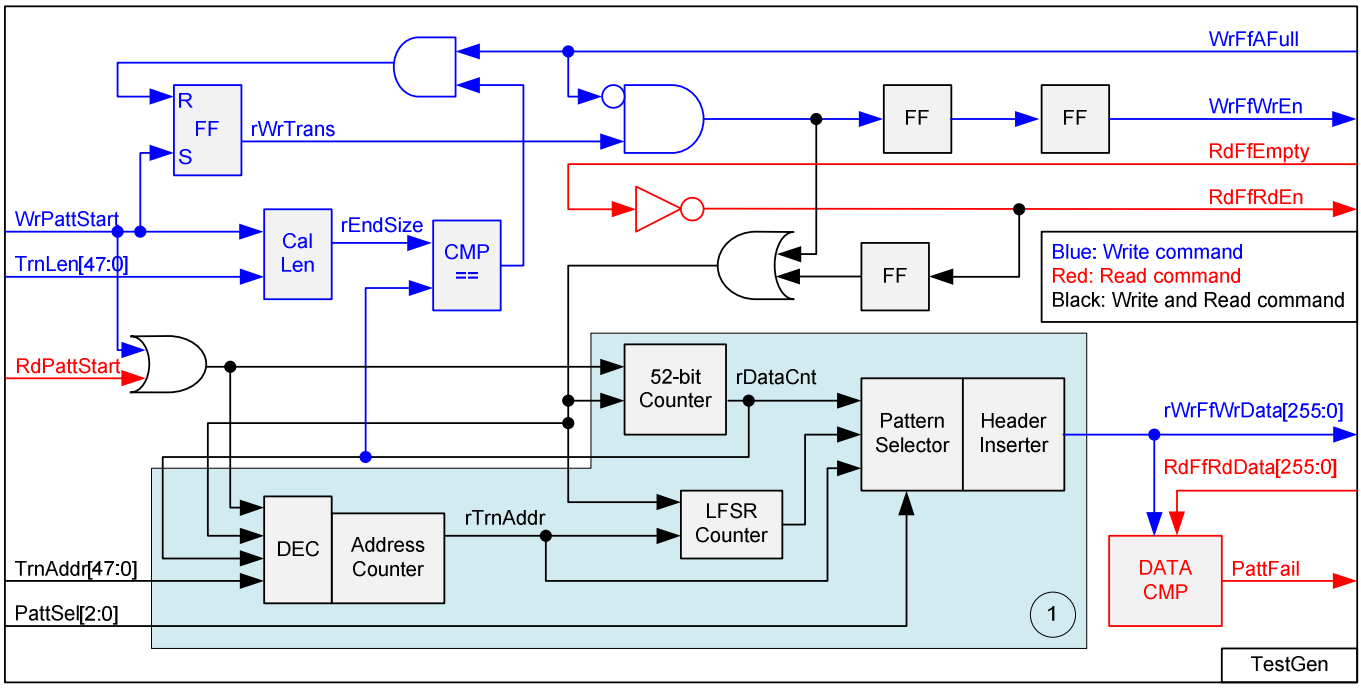

Figure 2-2 TestGen hardware

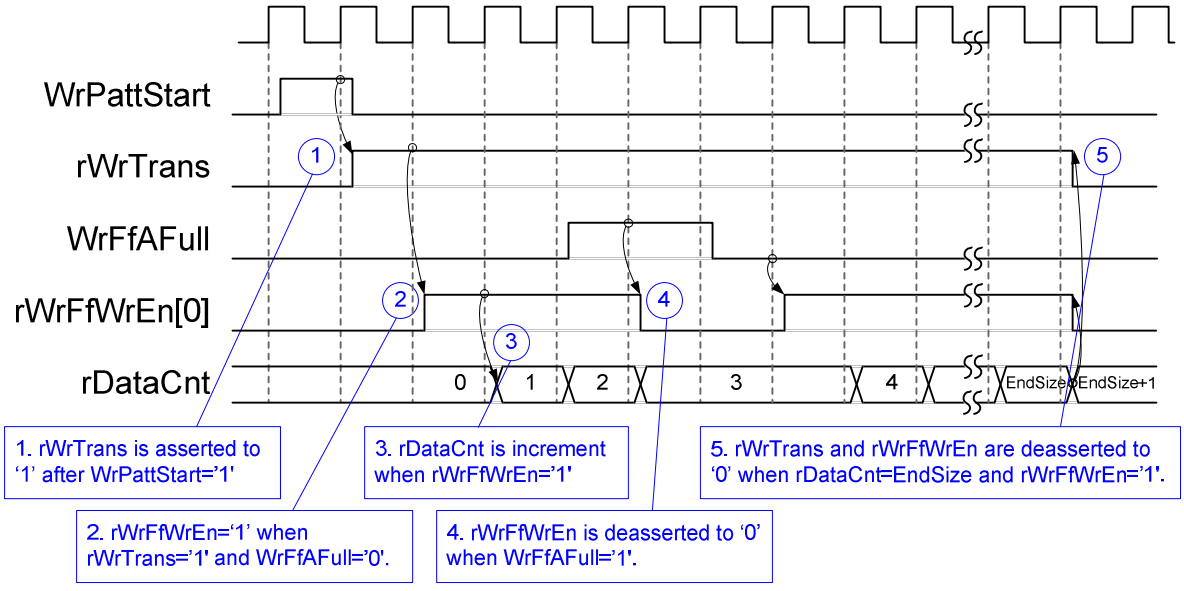

Figure 2-3 Timing diagram of Write operation in TestGen

To start Write operation, rWrTrans is asserted to '1' when WrPattStart from LAxi2Reg is asserted to '1'. When rWrTrans='1' and WrFfAFull='0', rWrFfWrEn[0] is asserted to '1' to send test data to WrFf. If WrFfAFull='1', rWrFfWrEn[0] will be de-asserted to '0' to pause data transferring. rDataCnt is data counter to check total transfer size, increased by rWrFfWrEn[0]. When total data are transferred completely (rDataCnt=EndSize), rWrTrans and rWrFfWrEn[0] are de-asserted to '0' to stop data transferring.

For Read operation, RdFfRdEn signal is designed by using NOT logic to RdFfEmpty. rDataCn is increased when RdFfRdEn is asserted to '1'.

Block no.1 in lower side of Figure 2-2 shows the logic for generating test pattern in TestGen module. To create unique test data for each sector, test pattern in this demo is designed as shown in Figure 2-4.

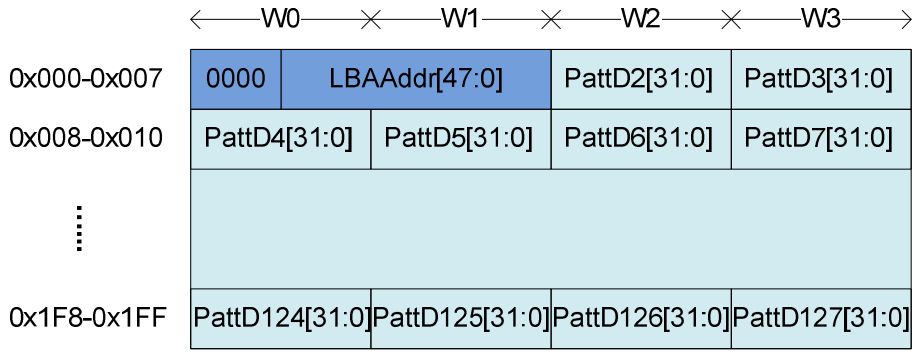

Figure 2-4 Test pattern format in each sector

Test pattern consists of two parts, i.e. 64-bit header in word#0 and word#1 of each sector and test data in word#2 – word#127. 64-bit header is created by using LBA address value of the data (LBA address is the address in sector unit). As shown in Figure 2-2, TrnAddr is used to be initial value of rTrnAddr. rTrnAddr is applied to be 64-bit header of each sector and increased after completing transfer one sector data. rDataCnt and write/read enable signal are monitored to check end of sector transferring.

TestGen supports to generate five patterns, i.e. 32-bit increment, 32-bit decrement, all 0, all 1, and 32-bit LFSR. 32-bit increment is generated by using lower-bit of rTrnAddr and rDataCnt. Decrement pattern is designed by using NOT logic with increment data. To create 32-bit LFSR counter, 256-bit data is designed by using two sets of 32-bit LFSR pattern as shown in Figure 2-5.

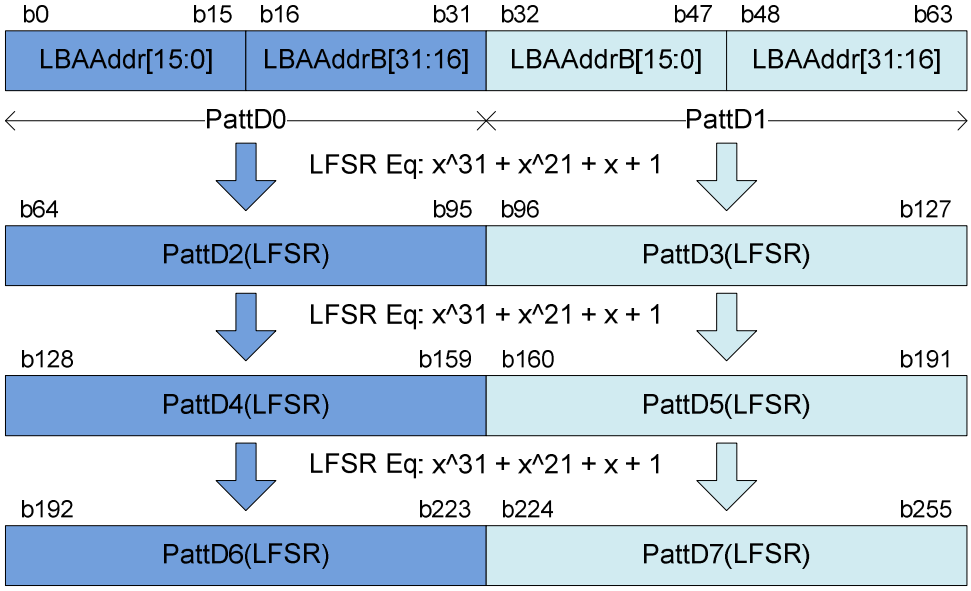

Figure 2-5 LFSR pattern in TestGen

Start value of LFSR is designed by using combination signal of 32-bit LBA address and NOT logic of LBA address (LBAAddrB means NOT logic of LBAAddr signal). PattD0/D2/D4/D6 are the sequence of LFSR counter set#0 which using LBAAddr[15:0] and NOT LBAAddr[31:16] to be initial value. PattD1/D3/D5/D7 are the sequence of LFSR counter set#1 which using NOT LBAAddr[15:0] and LBAAddr[31:16] to be initial value. 256-bit data is generated within one clock, so PattD2-D7 must design the logic to calculate 32-bit LFSR pattern as look-ahead style.

3-bit PattSel signal are used to select one of five test patterns. Header Inserter logic inserts 64-bit header to be the 1st data of each sector. After that, test data from pattern counter is selected to be rWrFfWrData. In Read command, rWrFfWrData is used to be expected value to compare with read data from FIFO (RdFfRdData). PattFail is asserted to '1' when data verification is failed.

2.2 RAID

User interface of RAID is designed by using dgIF typeS format. CMD interface is connected to LAxi2Reg to receive the parameter from user through Serial console. Data bus size of RAID block is 256-bit which is connected with U2IPFIFO and IP2UFIFO. Another side of RAID is connected to two NVMe SSDs. The system includes two IdenRAMs for receiving Iden I/F of NVMe-IP#0 - #1. The hardware details of RAID are shown in Figure 2-6.

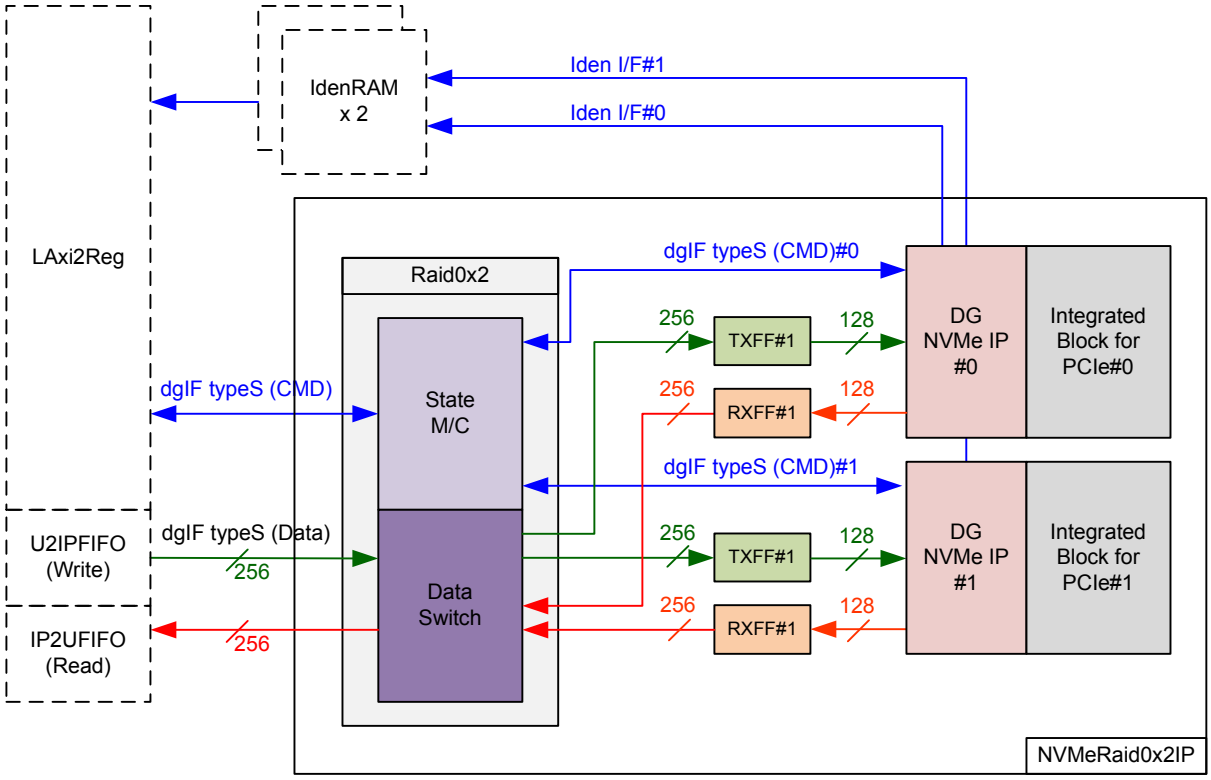

Figure 2-6 RAID hardware

The hardware inside RAID consists of three modules, i.e. Raid0x2, two FIFO sets, and two NVMe-IPs.

2.2.1 NVMe-IP

NVMe-IP implements NVMe protocol of Host side to access NVMe SSD. User interface is simple designed by using dgIF typeS format. NVMe-IP is designed to connect with Integrated Block for PCIe which is Hard IP in Xilinx device. More details of NVMe-IP are described in datasheet.
http://www.dgway.com/products/IP/NVMe-IP/dg_nvme_ip_data_sheet_en.pdf

2.2.2 FIFO

Two FIFOs are applied for one NVMe-IP. TxFIFO is designed to convert 256-bit data which is Raid0x2 bus size to 128-bit data which is NVMe-IP bus size. FIFO size is 16 kByte, so it can store data up to 31 sectors. RxFIFO size is same as TxFIFO, but it converts data from 128-bit to 256-bit.

2.2.3 Raid0x2

Raid0x2 consists of state machine for controlling command interface and Data Switch for controlling data interface.

After receiving new command request from LAxi2Reg, state machine calculates the address and transfer length of each NVMe-IP by decoding user parameter inputs. Next, state machine generates command request with the valid address and length to all NVMe-IPs.

State machine is responsible to control Data Switch for selecting the active NVMe-IP channel. State machine calculates the current sector address by loading the initial value from user inputs. The address is updated (increased by 1) at the end of one sector data transferring with U2IPFIFO/IP2UFIFO. The lower bit of current sector is used to select NVMe channel (‘0’ for NVMe-IP#0 and ‘1’ for NVMe-IP#1).

There are many pipeline registers to switch data bus of two NVMe channels. The overhead time from pipeline register and control signal is about 6 clock cycles for transferring 512-byte data (16x256-bit). The overhead time to transfer 512-byte data is 27% (6 / (16+6) = 27%). To compensate the overhead time, clock frequency of Raid0x2 must be high enough. From calculation, at least 317.5 MHz should be used (317.5 = 1.27 x 250 MHz, 250 MHz is PCIe Clock frequency for Gen3 speed). 317.5 MHz is too high to meet timing constraint in some FPGA devices. In the reference design, 275 MHz is the maximum frequency which can implement without timing constraint problem. So, performance of RAID0 operation may less than two times of one SSD. From test result, read performance of RAID0 = 6200 MB/s which is almost equal to two times of one channel (one channel shows read performance = 3200 MB/s). But write performance of RAID0 is equal to two times of one channel.

The user interface of Raid0x2 is shown in Table 2-1. Command and data interface are designed as dgIF typeS format. Please see more details of dgIF types format from NVMe-IP datasheet.

Table 2-1 Signal description of Raid0x2 (only User interface)

Signal	Dir	Description
User Interface		
RstB	In	Reset signal. Active low. Please use same reset signal as NVMe-IP.
Clk	In	Use same clock as Clk signal of NVMe-IP. The frequency must be more than or equal to PCIeClk (Clock output of Integrated Block for PCIe: 125 MHz for Gen2, 250 MHz for Gen3).
dgIF typeS		
UserCmd[1:0]	In	User Command. “00”: Identify command, “10”: Write PCIe SSD, ”11”: Read PCIe SSD.
UserAddr[47:0]	In	Start address to write/read SSD in sector unit (512 byte). **From SSD characteristic, it is recommended to set bit[3:0]=”0000” to align 8 Kbyte size (2xSSD page size)**. Write/Read performance in most SSD are reduced when start addrss is not aligned to page size (4 Kbyte).
UserLen[47:0]	In	Total transfer size in the request in sector unit (512 byte). Valid from 1 to (LBASize-UserAddr).
UserReq	In	Request the new command. Asserted to ‘1’ with valid value on UserCmd/UserAddr/UserLen signals. Can be asserted to ‘1’ when the IP is Idle (UserBusy=’0’).
UserBusy	Out	IP Busy status. ‘0’-IP is idle, ‘1’-IP is busy. New request will not be allowed if this signal is asserted to ‘1’.
LBASize[47:0]	Out	Total capacity of SSD in sector unit (512 byte). Default value is 0. This value is equal to two times of LBASize value output from IP#0.
UserError	Out	Error flag. Assert to ‘1’ when UserErrorType is not equal to 0. The flag can be reset only by de-asserting RstB to ‘0’.
UserErrorType[0-1][31:0]	Out	Error status, directly mapped from UserErrorType in each NVMe-IP. [0]-IP#0, [1]-IP#1
UserFifoWrCnt[15:0]	In	Write data counter of User received FIFO. Used to check FIFO space size. If FIFO size signal is less than 16-bit, please fill ‘1’ to upper bit. UserFifoWrEn is asserted to ‘1’ for 16 clock cycles to send 512-byte data when UserFifoWrCnt[15:5] is not equal to all 1.
UserFifoWrEn	Out	Write data valid of User received FIFO. Assert to ‘1’ to write data.
UserFifoWrData[255:0]	Out	Write data bus of User received FIFO. Synchronous to UserFifoWrEn.
UserFifoRdCnt[15:0]	In	Read data counter of User transmit FIFO. Used to check total data in FIFO. If FIFO size signal is less than 16-bit, please fill ‘0’ to upper bit. UserFifoRdEn can be asserted to ‘1’ for 16 clock cycles to read 512-byte data when UserFifoRdCnt[15:4] is not equal to 0.
UserFifoEmpty	In	FIFO empty flag of User transmit FIFO. This signal is unused in the design.
UserFifoRdEn	Out	Read valid of User transmit FIFO. Assert to ‘1’ to read data.
UserFifoRdData[255:0]	In	Read data returned from User transmit FIFO. Valid the next clock after UserFifoRdEn is asserted to ‘1’.

The operation of data interface in Raid0x2 module is shown by using timing diagram as follows.

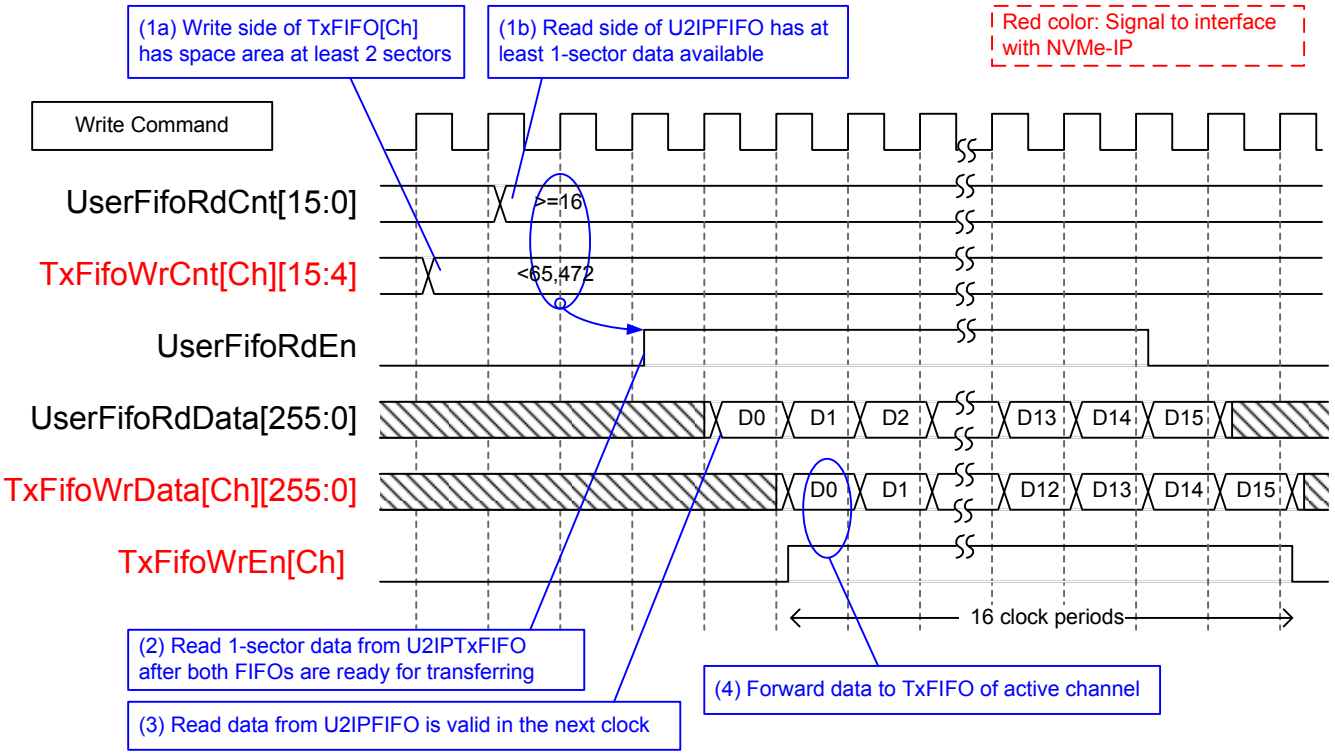

Figure 2-7 Raid0x2 data interface timing diagram in Write command

When user sends Write command to RAID0, data is forwarded from U2IPFIFO to TxFIFO[0]-[1]. One TxFIFO is active to transfer one sector data. After that, the active NVMe channel is switched to the next channel for transferring the next sector, following RAID0 behavior. Before forwarding one sector data, UserFifoRdCnt and TxFifoWrCnt of active channel are monitored to confirm that at least 1 sector data is stored in U2IPFIFO and TxFIFO of active channel has at least 2-sector free space. After FIFO is ready, UserFifoRdEn is asserted to '1' for 16 clock cycles to transfer 512-byte data continuously.

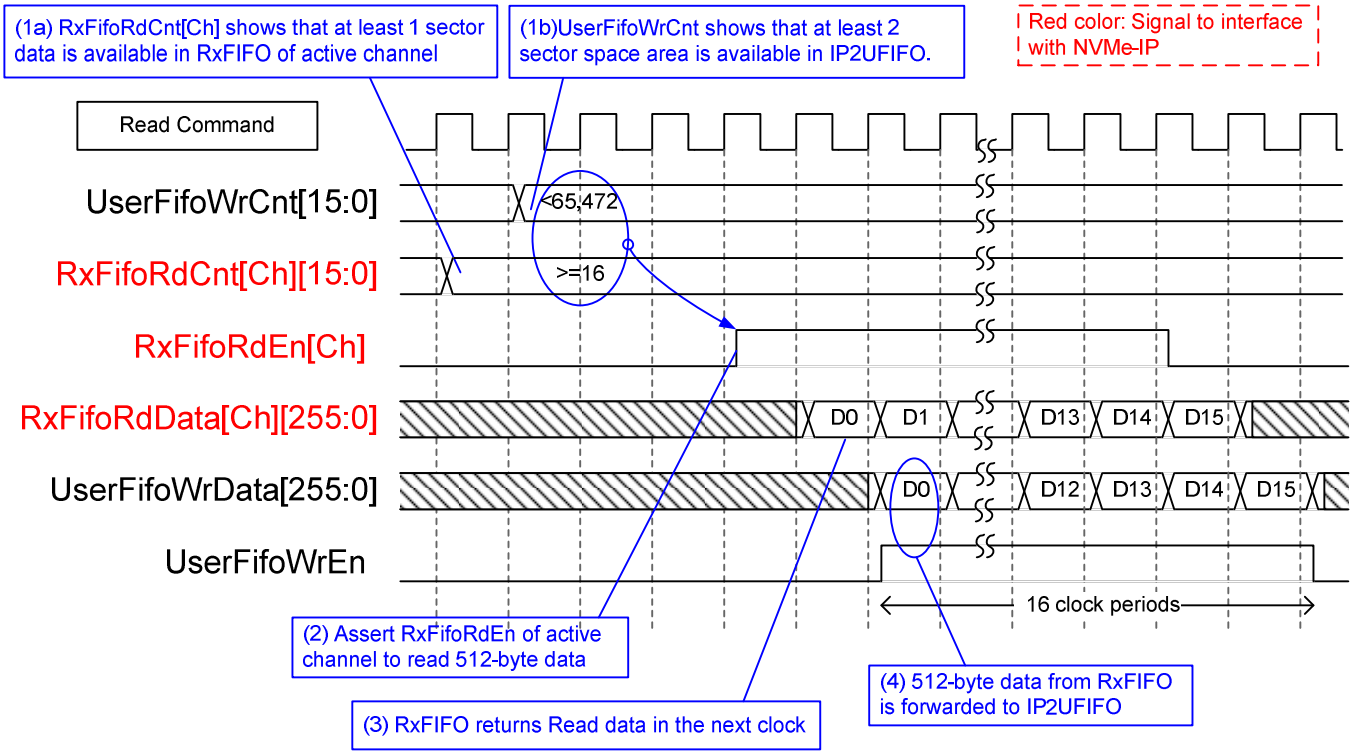

Figure 2-8 Raid0x2 data interface timing diagram in Read command

When user sends Read command to RAID0, data is forwarded from RxFIFO[0]-[1] to IP2UFIFO, as shown in Figure 2-8. Similar to Write command, one RxFIFO is active to transfer 512-byte. The active channel is switched to the next channel to transfer next 512-byte data. Before forwarding data, UserFifoWrCnt and RxFifoRdCnt of active channel are monitored to confirm that at least 1 sector data is available in RxFIFO and IP2UFIFO has at least 2-sector free space. UserFifoWrEn is asserted to '1' for 16 clock cycles to transfer 512-byte data continuously.

2.3 CPU and Peripherals

The hardware is connected to CPU through AXI4-Lite bus, similar to other CPU peripherals. Memory map of hardware registers to CPU is shown in Table 2-2. LAxi2Reg is the module to interface with CPU following memory map.

LAxi2Reg connects to many hardwares in the system such as TestGen, Raid0x2, and IdenRAM to get the control and status signals of each module. As shown in Figure 2-9, there are two clock domains applied in this block, i.e. CpuClk (CPU Clock and AXI4-Lite bus) and UserClk (User clock domain for TestGen and RAID0 block).

AsyncAxiReg includes asynchronous circuit between CpuClk and UserClk. More details of each hardware are described as follows.

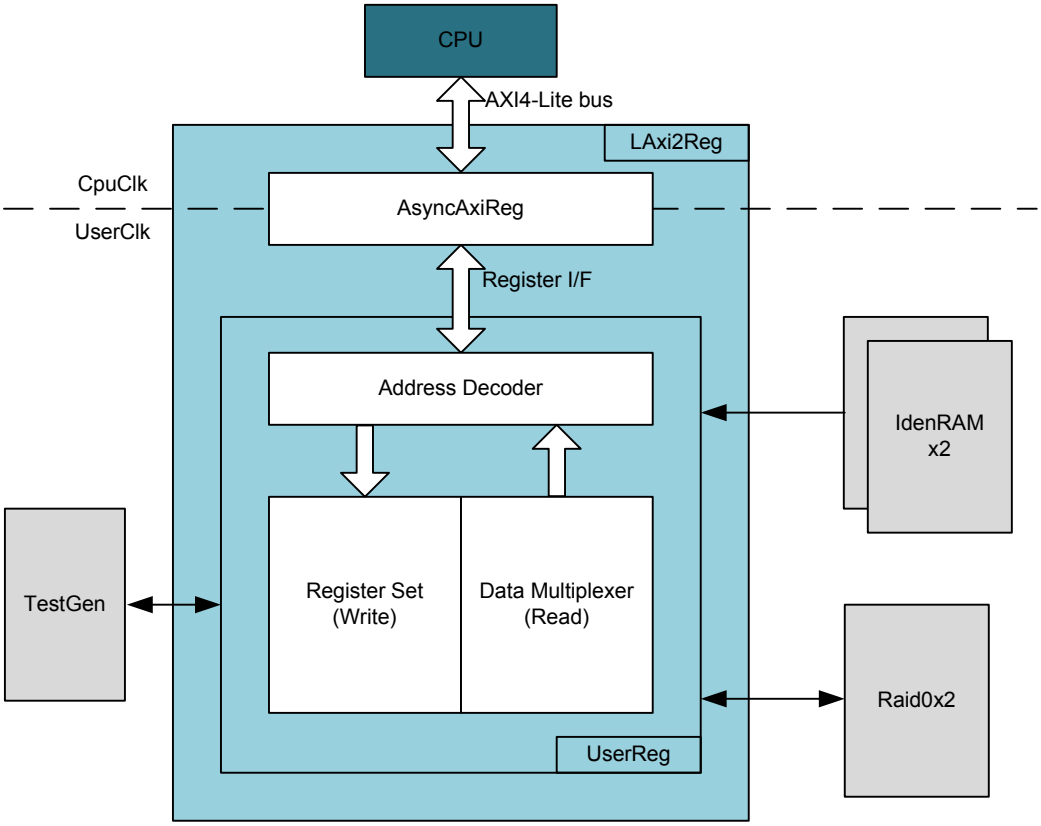

Figure 2-9 CPU and peripherals hardware

2.3.1 AsyncAxiReg

This module is designed to convert the signal interface of AXI4-Lite to be register interface. Also, it supports to convert clock domain from CpuClk to be UserClk domain. Timing diagram of register interface is shown in Figure 2-10.

To write register, timing diagram is same as 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 byte enable of this access: bit[0] is write enable for RegWrData[7:0], bit[1] is used for RegWrData[15:8], …, and bit[3] is used for RegWrData[31:24]).

To read register, AsyncAxiReg asserts RegRdReq to '1' with the valid value of RegAddr (the register address in 32-bit unit). After that, the read data is valid on RegRdData bus with asserting RegRdValid to '1'.

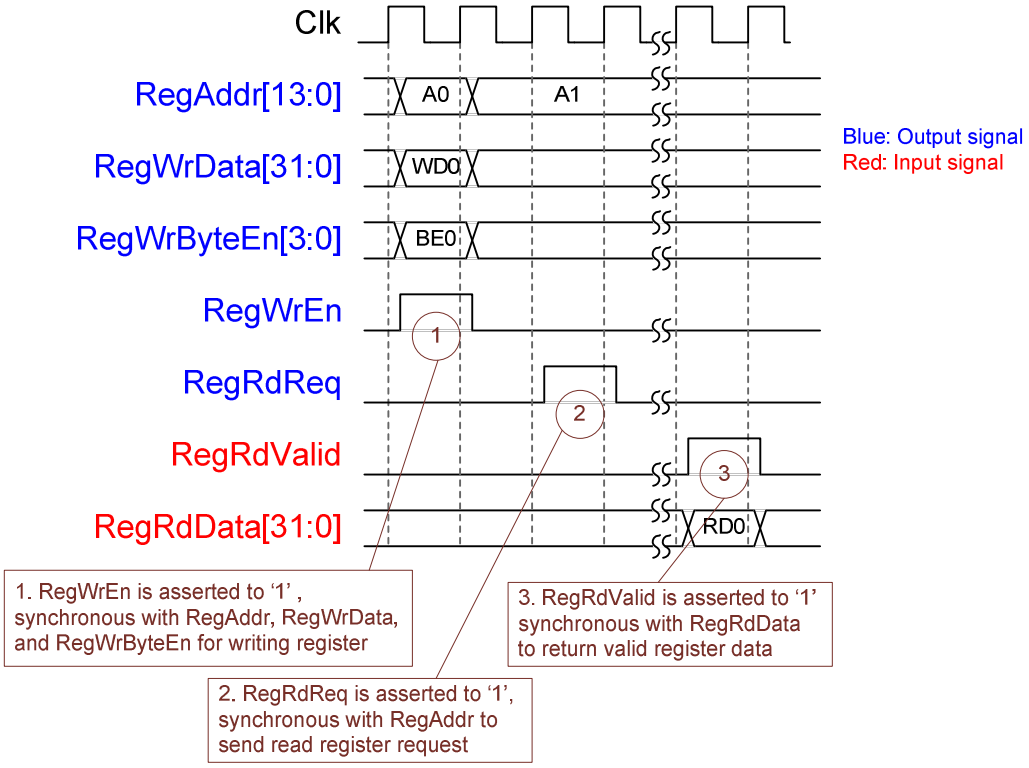

Figure 2-10 Register interface timing diagram

2.3.2 UserReg

As shown in Figure 2-9, after RegWrEn or RegRdReq is asserted to '1' to request write or read register, RegAddr is loaded to Address decoder to select the active register. For write register, RegWrData signal is loaded to be the new value of active register. In this module, RegWrByteEn is not used, so CPU firmware needs to access the hardware register by using 32-bit pointer only.

For read request, CPU monitors status signals of many modules such as TestGen, Raid0x2, and IdenRAM. To avoid timing constraint problem, many status signals are selected by using multiplexer with two-stage pipeline registers. So, RegRdValid is asserted to '1' after RegRdReq is asserted for two clock cycles. Two latency clock cycles is designed by adding two D Flip-flops to generate RegRdValid from RegRdReq.

Memory map of control and status signals inside UserReg module is shown in Table 2-2.

Table 2-2 Register Map

Address Rd/Wr	Register Name (Label in “nvmeipraid0x2test.c”)	Description
BA+0x000 Wr	User Address (Low) Reg (USRADRL_REG)	[31:0]: Input to be start sector address (UserAddr[31:0] of dgIF typeS for RAID0)
BA+0x004 Wr	User Address (High) Reg (USRADRH_REG)	[15:0]: Input to be start sector address (UserAddr[47:32] of dgIF typeS for RAID0)
BA+0x008 Wr	User Length (Low) Reg (USRLENL_REG)	[31:0]: Input to be transfer length in sector unit (UserLen[31:0] of dgIF typeS for RAID0)
BA+0x00C Wr	User Length (High) Reg (USRLENH_REG)	[15:0]: Input to be transfer length in sector unit (UserLen[47:32] of dgIF typeS for RAID0)
BA+0x010 Wr	User Command Reg (USRCMD_REG)	[1:0]: Input to be user command (UserCmd of dgIF typeS for RAID0) “00”-Identify, “10”-Write SSD, “11”-Read SSD. When this register is written, the design generates command request to RAID0 to start new command operation.
BA+0x014 Wr	Test Pattern Reg (PATTSEL_REG)	[2:0]: Test pattern select “000”-Increment, “001”-Decrement, “010”-All 0, “011”-All 1, “100”-LFSR
BA+0x100 Rd	User Status Reg (USRSTS_REG)	[0]: UserBusy of dgIF typeS for RAID0 (‘0’: Idle, ‘1’: Busy) [1]: UserError of dgIF typeS for RAID0 (‘0’: Normal, ‘1’: Error) [2]: Data verification fail (‘0’: Normal, ‘1’: Error)
BA+0x104 Rd	Total device size (Low) Reg (LBASIZEL_REG)	[31:0]: Total capacity of RAID0 in sector unit (LBASize[31:0] of dgIF typeS for RAID0)
BA+0x108 Rd	Total device size (High) Reg (LBASIZEH_REG)	[15:0]: Total capacity of RAID0 in sector unit (LBASize[47:32] of dgIF typeS for RAID0)
BA+0x110 Rd	PCIe Status CH#0 Reg (PCIESTS0_REG)	Status signal from Integrated Block for PCIe#0 [0]: PCIe linkup status (‘0’: No linkup, ’1’: linkup) [3:2]: PCIe link speed (“00”: No linkup, “01”: PCIe Gen1, “10”: PCIe Gen2, “11”: PCIe Gen3) [7:4]: PCIe link width status (“0001”: 1 lane, “0010”: 2 lane, ”0100”: 4 lane, ”1000”: 8 lane) [13:8]: Current LTSSM State. Please see more details of LTSSM value in Integrated Block for PCIe datasheet
BA+0x114 Rd	PCIe Status CH#1 Reg (PCIESTS1_REG)	Same as PCIESTS0_REG, but signals are fed from Integrated Block for PCIe#1
BA+0x180 Rd	User Error Type CH#0 Reg (USRERRTYPE0_REG)	[31:0]: Mapped to UserErrorType of NVMe-IP#0
BA+0x184 Rd	User Error Type CH#1 Reg (USRERRTYPE1_REG)	[31:0]: Mapped to UserErrorType of NVMe-IP#1
BA+0x190 Rd	Completion Status CH#0 Reg (COMPSTS0_REG)	[15:0]: Mapped to AdmCompStatus[15:0] of NVMe-IP#0 [31:16]: Mapped to IOCompStatus[15:0] of NVMe-IP#0
BA+0x194 Rd	Completion Status CH#1 Reg (COMPSTS1_REG)	Same as COMPSTS0_REG, but signals are fed from NVMe-IP#1
BA+0x1A0 Rd	NVMe CAP CH#0 Reg (NVMCAP0_REG)	[31:0]: Mapped to NVMeCAPReg[31:0] output of NVMe-IP#0
BA+0x1A4 Rd	NVMe CAP CH#1 Reg (NVMCAP1_REG)	[31:0]: Mapped to NVMeCAPReg[31:0] output of NVMe-IP#1
BA+0x1B0 Rd	Test pin of NVMe-IP#0 Reg (NVMTESTPIN0_REG)	[31:0]: Mapped to TestPin output of NVMe-IP#0
BA+0x1B4 Rd	Test pin of NVMe-IP#1 Reg (NVMTESTPIN1_REG)	[31:0]: Mapped to TestPin output of NVMe-IP#1

Address Rd/Wr	Register Name (Label in “nvmeipraid0x2test.c”)	Description
BA+0x200 Rd	Expected value Word0 Reg (EXPPATW0_REG)	[31:0]: Latch value of expected data [31:0] in TestGen when running Read command
BA+0x204 Rd	Expected value Word1 Reg (EXPPATW1_REG)	[31:0]: Latch value of expected data [63:32] in TestGen when running Read command
BA+0x208 Rd	Expected value Word2 Reg (EXPPATW2_REG)	[31:0]: Latch value of expected data [95:64] in TestGen when running Read command
BA+0x20C Rd	Expected value Word3 Reg (EXPPATW3_REG)	[31:0]: Latch value of expected data [127:96] in TestGen when running Read command
BA+0x210 Rd	Expected value Word4 Reg (EXPPATW4_REG)	[31:0]: Latch value of expected data [159:128] in TestGen when running Read command
BA+0x214 Rd	Expected value Word5 Reg (EXPPATW5_REG)	[31:0]: Latch value of expected data [191:160] in TestGen when running Read command
BA+0x218 Rd	Expected value Word6 Reg (EXPPATW6_REG)	[31:0]: Latch value of expected data [223:192] in TestGen when running Read command
BA+0x21C Rd	Expected value Word7 Reg (EXPPATW7_REG)	[31:0]: Latch value of expected data [255:224] in TestGen when running Read command
BA+0x240 Rd	Read value Word0 Reg (RDPATW0_REG)	[31:0]: Latch value of read data [31:0] in TestGen when running Read command
BA+0x244 Rd	Read value Word1 Reg (RDPATW1_REG)	[31:0]: Latch value of read data [63:32] in TestGen when running Read command
BA+0x248 Rd	Read value Word2 Reg (RDPATW2_REG)	[31:0]: Latch value of read data [95:64] in TestGen when running Read command
BA+0x24C Rd	Read value Word3 Reg (RDPATW3_REG)	[31:0]: Latch value of read data [127:96] in TestGen when running Read command
BA+0x250 Rd	Read value Word4 Reg (RDPATW4_REG)	[31:0]: Latch value of read data [159:128] in TestGen when running Read command
BA+0x254 Rd	Read value Word5 Reg (RDPATW5_REG)	[31:0]: Latch value of read data [191:160] in TestGen when running Read command
BA+0x258 Rd	Read value Word6 Reg (RDPATW6_REG)	[31:0]: Latch value of read data [223:192] in TestGen when running Read command
BA+0x25C Rd	Read value Word7 Reg (RDPATW7_REG)	[31:0]: Latch value of read data [255:224] in TestGen when running Read command
BA+0x280 Rd	Data Failure Address (Low) Reg (RDFAILNOL_REG)	[31:0]: Latch value of failure address [31:0] in byte unit when running Read command
BA+0x284 Rd	Data Failure Address (High) Reg (RDFAILNOH_REG)	[24:0]: Latch value of failure address [56:32] in byte unit when running Read command
BA+0x288 Rd	Current test byte (Low) Reg (CURTESTSIZEL_REG)	[31:0]: Current test data size of TestGen module in byte unit (bit[31:0])
BA+0x28C Rd	Current test byte (High) Reg (CURTESTSIZEH_REG)	[24:0]: Current test data size of TestGen module in byte unit (bit[56:32])
BA+0x2000 – 0x2FFF	Identify Controller Data (IDENCTRL0_REG)	4Kbyte Identify Controller Data Structure from NVMe CH#0
BA+0x3000 – 0x3FFF	Identify Namespace Data (IDENNAME0_REG)	4Kbyte Identify Namespace Data Structure NVMe CH#0
BA+0x4000 – 0x4FFF	Identify Controller Data (IDENCTRL1_REG)	4Kbyte Identify Controller Data Structure from NVMe CH#1
BA+0x5000 – 0x5FFF	Identify Namespace Data (IDENNAME1_REG)	4Kbyte Identify Namespace Data Structure NVMe CH#1

3 CPU Firmware

After system boot-up, CPU initializes its peripherals such as UART and Timer. Next, CPU waits until PCIe connection of two SSDs is link up (PCISTS0_REG[0]='1' and PCISTS1_REG[0]='1') and RAID0 is not busy (USRSTS_REG[0]='0'). Finally, Main menu is displayed.

CPU firmware supports three commands following USRCMD_REG value, i.e. "00" for Identify command, "10" for Write command, and "11" for Read command. More details of the sequence in each command are described as follows.

3.1 Identify command

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

1) Set USRCMD_REG="00". Next, RAID0 sends Identify command to all NVMe-IPs. RAID0 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 value. Bit[0] is de-asserted to '0' when command is completed. Bit[1] is asserted to '1' when some errors are detected. In case of error condition, there is error message displayed on the console. If the command is completed, the data from Identify command of all NVMe-IPs will be stored in IdenRAMs.
3) CPU reads Identify data from IdenRAMs which are mapped to IDENCTRL0/1_REG address. Then, CPU displays the information such as SSD model name on Serial console. Also, RAID0 device capacity (LBASIZEL/H_REG) is displayed in GB unit on the console.

3.2 Write/Read command

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

1) Receive start address, transfer length, test pattern through Serial console. If some inputs are invalid, the operation will be cancelled.
2) Get all inputs and set the value to USRADRL/H_REG, USRLENL/H_REG, PATTSEL_REG, and USRCMD_REG (USRCMD_REG="10" for Write command, and "11" for Read command).
3) CPU waits until the operation is completed or some errors (except verification error) are found by monitoring USRSTS_REG[2:0].
   If USRSTS_REG[2] (verification error) is asserted to '1', verification error message will be displayed. After that, CPU still runs until end of operation or user inputs any key to cancel operation.
4) During running command, current transfer size reading from CURTESTSIZE_REG is displayed every second. Finally, test performance is displayed on Serial console when command is completed.

4 Example Test Result

The example test result when running RAID0 demo system by using two 512 GB Samsung 960 Pro SSDs is shown in Figure 4-1.

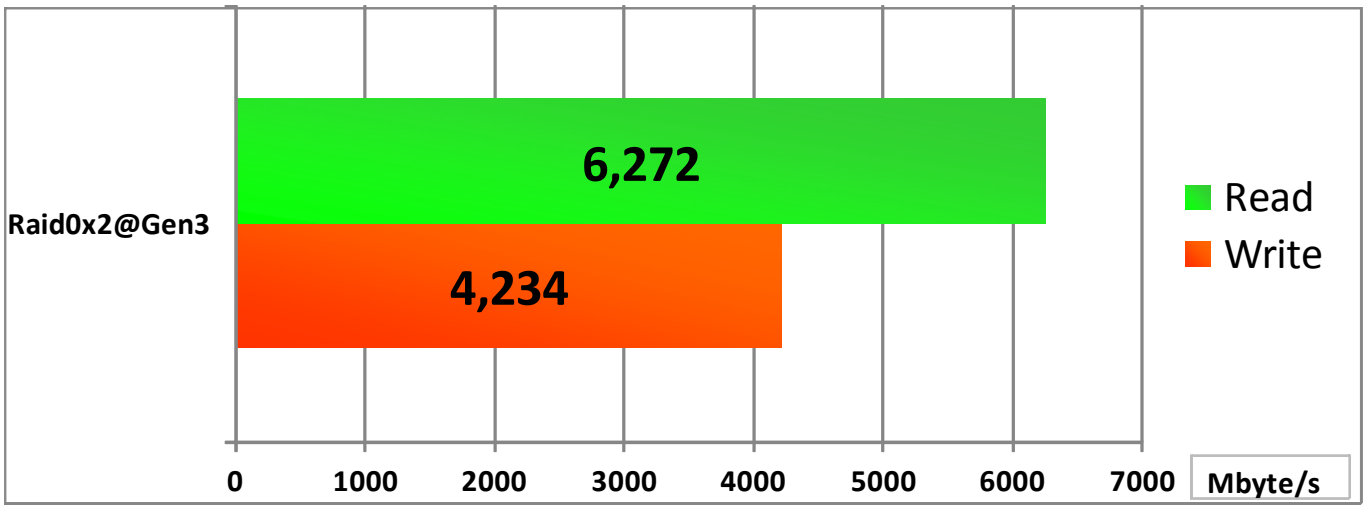

Figure 4-1 Performance of 2-ch RAID0 demo by using Samsung 960 Pro SSD

When running 2-ch RAID0 with 2 SSDs@Gen3 speed, write performance is about 4200 Mbyte/sec and read performance is about 6200 Mbyte/sec.

5 Revision History

Revision	Date	Description
1.0	6-Oct-17	Initial version release
1.1	5-Jun-18	Update register map and description