NVMe-IP reference design manual

Rev2.1 27-Jul-17

1. NVMe

NVM Express (NVMe) defines the interface for the host controller to access solid state drives (SSD) by PCI Express. NVM Express optimizes the process to issue command and completion by using only 2 register writes for command issue/completion cycle. Also, NVMe supports parallel operation by supporting up to 64K commands within single queue. So, performance for both sequential and random access is improved.

In PCIe SSD market, two standards are found, i.e. AHCI and NVMe. AHCI is the older standard to provide the interface for SATA hard disk drives while NVMe is optimized for non volatile memory like SSD. The comparison between both AHCI and NVMe protocol in more details can be found from "A Comparison of NVMe and AHCI" document.
https://sata-io.org/system/files/member-downloads/NVMe%20and%20AHCI_%20_long_.pdf

The example of NVMe storage devices is shown in http://www.nvmexpress.org/products/.

Generally, user needs to install NVMe driver to access NVMe SSD as shown in Figure 1. Physical connector of NVMe SSD is PCIe type such as M.2 connector. NVMe-IP implements NVMe driver and the task running on CPU by pure-hardware logic. So, CPU is not required to access NVMe SSD when using NVMe-IP in FPGA board.

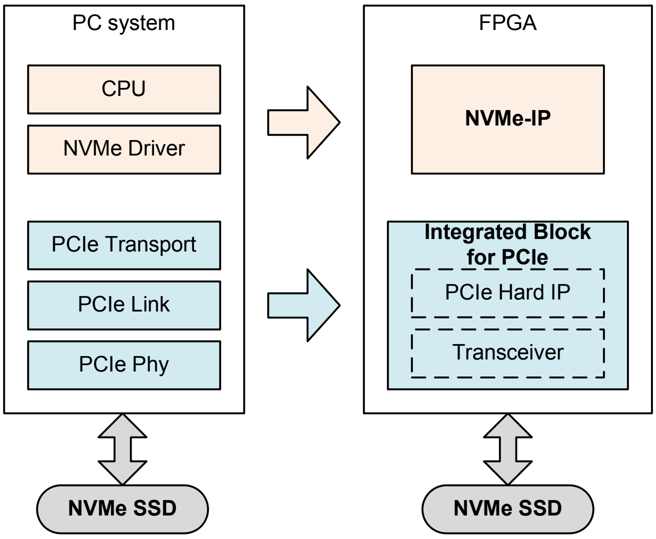
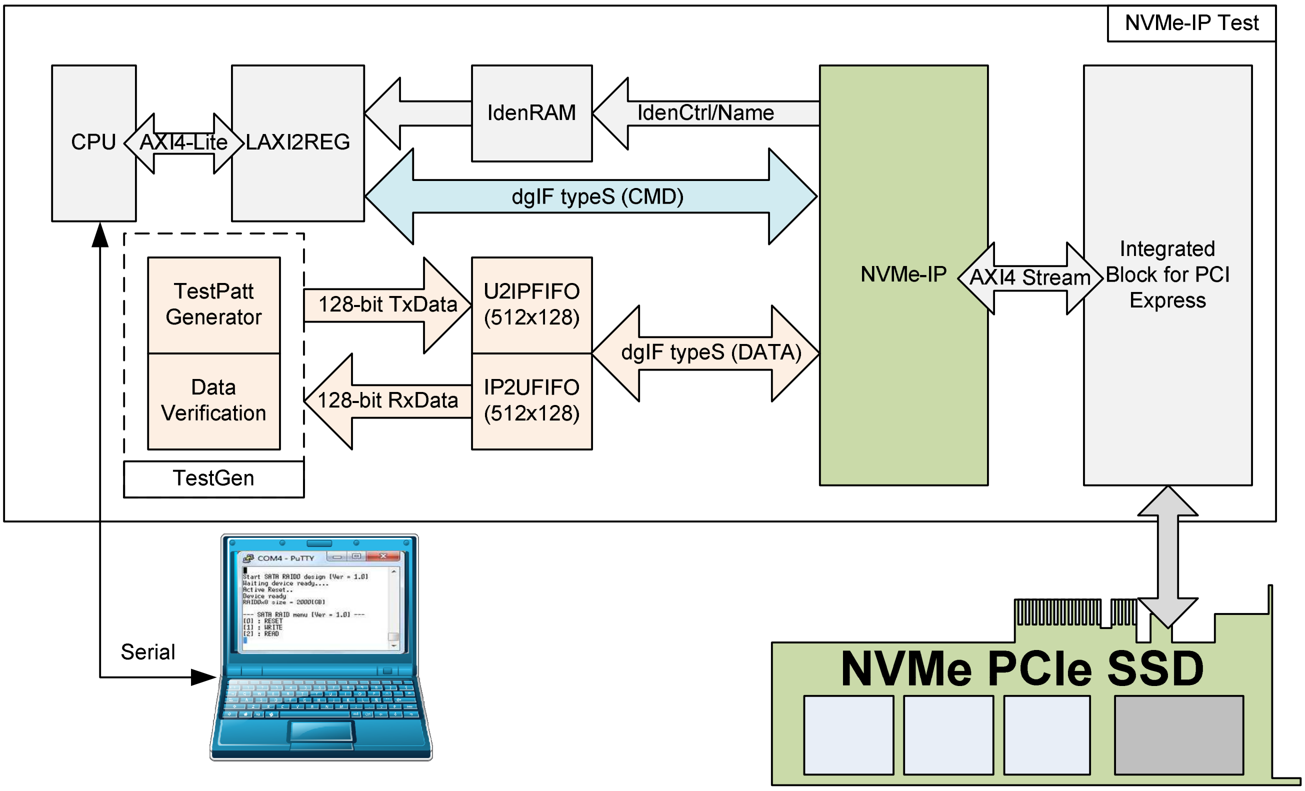


Figure 1 NVMe protocol layer

2. Overview



```
Start SATA RAID0 design [Ver = 1.0]
Waiting device ready....
Active Reset..
Device ready
RAID0x8 size = 2000[GB]

--- SATA RAID menu [Ver = 1.0] ---
[0] : RESET
[1] : WRITE
[2] : READ
```

Figure 2 NVMe-IP Demo System

The reference design integrates NVMe-IP with simple logic to write and read data with NVMe PCIe SSD at high-speed rate. CPU is additional designed for user interface through Serial console.

For simple test, user inputs the parameters such as start address, transfer size, and command from keyboards. After that, the logic processes all inputs and converts to be NVMe-IP input. After the operation is completed, CPU will check time usage and calculate write/read performance of the SSD. To interface with CPU bus, LAXI2REG module is used to decode the address and data from AXI4 bus and converts to command interface of dgIF typeS. Data interface of dgIF typeS is connected to external FIFO and transferred to data buffer within NVMe-IP. TestGen module includes test pattern generator to generate Test data. The test data is transferred to SSD in Write Test, and used to be expected value for data verification in Read Test. IdenCtrl/IdenName data are transferred to IdenRAM, and CPU decodes SSD model name by using Identify data.

NVMe-IP clock frequency in the reference for PCIe Gen3 is 275 MHz, while the frequency for PCIe Gen2 is 200 MHz. The clock frequency of NVMe-IP is more than or equal to clock output from Integrated Block for PCI Express (125 MHz for PCIe Gen2, 250 MHz for PCIe Gen3 interface).

User can download NVMe-IP datasheet and send request to evaluate the IP from our website, http://www.dgway.com/NVMe-IP_X_E.html.
The real transfer performance in the demo depends on NVMe PCIe SSD characteristic.

3. CPU

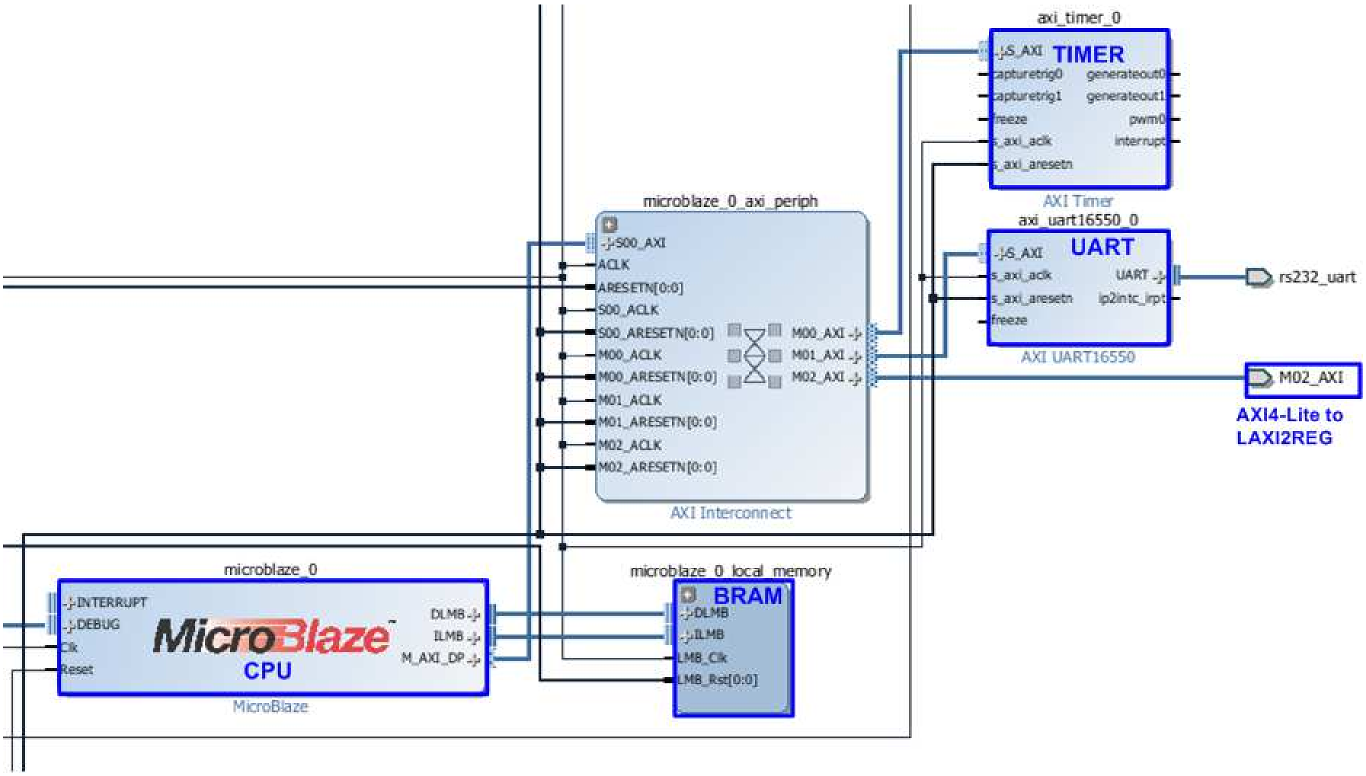


Figure 3 CPU system in reference design

In reference design, CPU peripherals consist of UART for user interface, Timer for performance measurement, and BRAM for CPU firmware. AXI-interconnect is used for MicroBlaze to interface with its peripherals. MicroBlaze interface for controlling/monitoring test system is 32-bit AXI4-Lite bus. More details about memory map of this AXI4-Lite are follows.

Table 1 Register Map

Address Rd/Wr	Register Name (Label in the "nvmeiptest.c")	Description
BA+0x00 Wr	User Address (Low) Reg (USRADRL_REG)	[31:0]: Input to be start sector address (UserAddr[31:0] of dgIF TypeS)
BA+0x04 Wr	User Address (High) Reg (USRADRH_REG)	[15:0]: Input to be start sector address (UserAddr[47:32] of dgIF TypeS)
BA+0x08 Wr	User Length (Low) Reg (USRLENL_REG)	[31:0]: Input to be transfer length in sector unit (UserLen[31:0] of dgIF TypeS)
BA+0x0C Wr	User Length (High) Reg (USRLENH_REG)	[15:0]: Input to be transfer length in sector unit (UserLen[47:32] of dgIF TypeS)
BA+0x10 Wr	User Command Reg (USRCMD_REG)	[1:0]: Input to be user command (UserCmd of dgIF TypeS) "00"-Identify device, "10"-Write SSD, "11"-Read SSD When this register is written, the design will generate command request to NVMe-IP to start new command operation.
BA+0x14 Wr	Test Pattern Reg (PATTSEL_REG)	[2:0]: Test pattern select "000"-Increment, "001"-Decrement, "010"-All 0, "011"-All 1, "100"-LFSR

Address Rd/Wr	Register Name (Label in the "nvmeiptest.c")	Description
BA+0x100 Rd	User Status Reg (USRSTS_REG)	[0]: UserBusy of dgIF TypeS ('0': Idle, '1': Busy) [1]: UserError of dgIF TypeS ('0': Normal, '1': Error) [2]: Data verification fail ('0': Normal, '1': Error) [4:3]: PCIe speed from IP ("00": No linkup, "01": PCIe Gen1, "10": PCIe Gen2, "11": PCIe Gen3)
BA+0x104 Rd	Total disk size (Low) Reg (LBASIZEL_REG)	[31:0]: Total capacity of SSD in sector unit (LBASize[31:0] of dgIF TypeS)
BA+0x108 Rd	Total disk size (High) Reg (LBASIZEH_REG)	[15:0]: Total capacity of SSD in sector unit (LBASize[47:32] of dgIF TypeS)
BA+0x10C Rd	User Error Type Reg (USRERRTYPE_REG)	[31:0]: User error status (UserErrorType[31:0] of dgIF TypeS)
BA+0x114 Rd	Completion Status Reg (COMPSTS_REG)	[15:0]: Status from Admin completion (AdmCompStatus[15:0] from NVMe-IP) [31:16]: Status from IO completion (IOCompStatus[15:0] from NVMe-IP)
BA+0x118 Rd	NVMe CAP Reg (NVMCAP_REG)	[31:0]: NVMeCAPReg[31:0] output from NVMe-IP
BA+0x11C Rd	NVMe IP Test pin Reg (NVMTESTPIN_REG)	[31:0]: TestPin[31:0] output from NVMe-IP
BA+0x120 Rd	Data Failure Address (Low) Reg (RDFAILNOL_REG)	[31:0]: Latch value of failure address[31:0] in byte unit from read command
BA+0x124 Rd	Data Failure Address (High) Reg (RDFAILNOH_REG)	[24:0]: Latch value of failure address [56:32] in byte unit from read command
BA+0x130 Rd	Expected value Word0 Reg (EXPPATW0_REG)	[31:0]: Latch value of expected data [31:0] from read command
BA+0x134 Rd	Expected value Word1 Reg (EXPPATW1_REG)	[31:0]: Latch value of expected data [63:32] from read command
BA+0x138 Rd	Expected value Word2 Reg (EXPPATW2_REG)	[31:0]: Latch value of expected data [95:64] from read command
BA+0x13C Rd	Expected value Word3 Reg (EXPPATW3_REG)	[31:0]: Latch value of expected data [127:96] from read command
BA+0x140 Rd	Read value Word0 Reg (RDPATW0_REG)	[31:0]: Latch value of read data [31:0] from read command
BA+0x144 Rd	Read value Word1 Reg (RDPATW1_REG)	[31:0]: Latch value of read data [63:32] from read command
BA+0x148 Rd	Read value Word2 Reg (RDPATW2_REG)	[31:0]: Latch value of read data [95:64] from read command
BA+0x14C Rd	Read value Word3 Reg (RDPATW3_REG)	[31:0]: Latch value of read data [127:96] from read command
BA+0x150 Rd	Current test byte (Low) Reg (CURTESTSIZEL_REG)	[31:0]: Current test data size of TestGen module in byte unit (bit[31:0])
BA+0x154 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 (IDENCTRL_REG)	4Kbyte Identify Controller Data Structure
BA+0x3000 – 0x3FFF	Identify Namespace Data (IDENNAME_REG)	4Kbyte Identify Namespace Data Structure

After initialization is completed, CPU firmware in the demo will be in idle state to wait user command input from Serial console. The user command is Identify device, write, or read command. The sequence of each command is follows.

For Identify device command,

1) Set USRCMD_REG=”00”. Next, Test logic generates command and request to NVMe-IP. After that, Busy flag (USRSTS_REG[0]) changes from ‘0’ to ‘1’.
2) CPU waits until command is completed or the error is found by monitoring USRSTS_REG value. Bit[0] is cleared to ‘0’ when command is completed. Bit[1] is asserted to ‘1’ when some errors is detected. If the error is detected, error message will be displayed.
3) To be test result, SSD model name decoded from IDENCTRL_REG and SSD capacity read from LBASIZEL/H_REG are displayed to the command shell.

For Write/Read command,

1) Receive start address, transfer length, and test pattern value from user through Serial console. If some input is invalid, the operation will be cancelled.
2) Get all inputs and set the value to USRADRL/H_REG, USRLENL/H_REG, and USRCMD_REG (USRCMD_REG=”10” for write transfer, and “11” for read transfer).
3) Similar to step 2) in Identify command. But USRSTS_REG[2] is also monitored for read command to confirm that all read data are correct.
4) During running command, current transfer size is displayed every second. Finally, test performance is displayed on the console shell when command is completed.

4. LAXI2REG

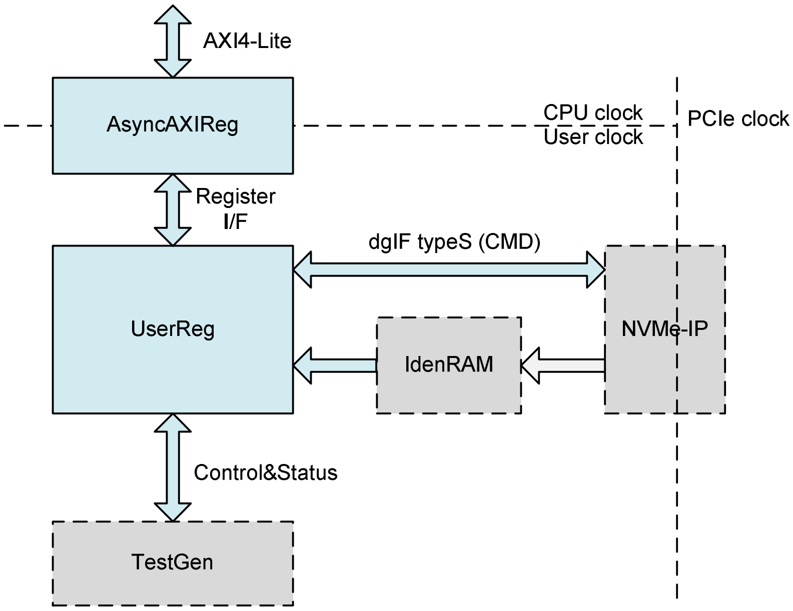


Figure 4 LAXI2REG interface

This module consists of two submodules, i.e. AsyncAXIReg and UserReg. AsyncAXIReg is designed to convert AXI4-Lite bus to be register interface and to convert clock domain from CPU clock to user clock system. UserReg module includes the logic to decode Write/Read address to select the register for current access. The address is decoded following Table 1. Transfer parameters such as transfer direction, size, and address from user are converted to be command interface of dgIF typeS for NVMe-IP and converted to be control signal for TestGen module. During transferring, CPU reads the register to check NVMe-IP status, TestGen result, or Identify device data.

5. TestGen

In this module, there are two operations. For Write command, it generates test data to WrFf port, while it verifies received data from RdFf port with expected value for Read command. The details of logic design inside this module are displayed in Figure 5.

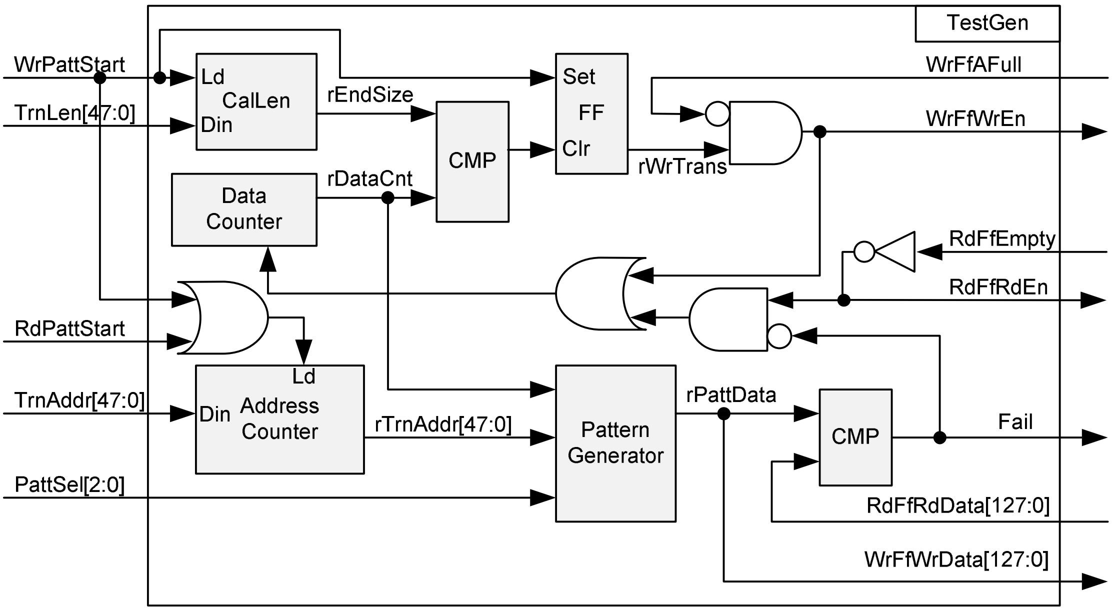


Figure 5 Logic design in TestGen

To start write transfer, WrPattStart is asserted to '1' with valid signal on TrnLen and TrnAddr. TrnLen is used to calculate the end position of write transfer. rWrTrans which is designed to be write enable signal for WrFf port is controlled by WrPattStart and end position signal (rEndSize). WrFfAFull is used to be flow control for write transfer. If this signal is asserted to '1', WrFfWrEn will be de-asserted to '0' to pause data generating.

There are two counters inside this module, i.e. Data counter which is used to count total transfer size to monitor end transfer timing. Another is current address counter in sector unit (512-byte). The address counter loads the start value from TrnAddr signal. The address counter is increment when end of each 512-byte transfer. Pattern Generator reads the current address from rTrnAddr to be 64-bit header value of each sector. Also, this value is used to be start test pattern for 32-bit increment, 32-bit decrement, and 32-bit LFSR counter. rPattData is applied to be WrFfWrData for write transfer and applied to be expect value for read command.

For read transfer, RdPattStart is used to be load signal for Address counter only. RdFfRdEn is controlled by RdFfEmpty. It is simple design of RdFfRdEn by using not logic of RdFfEmpty signal. Fail flag will be asserted to '1' if RdFfRdData from RdFf port is not equal to test pattern.

6. Example Test Result

The example test result when running demo system by using 512 GB Samsung 960 Pro is shown in Figure 6.

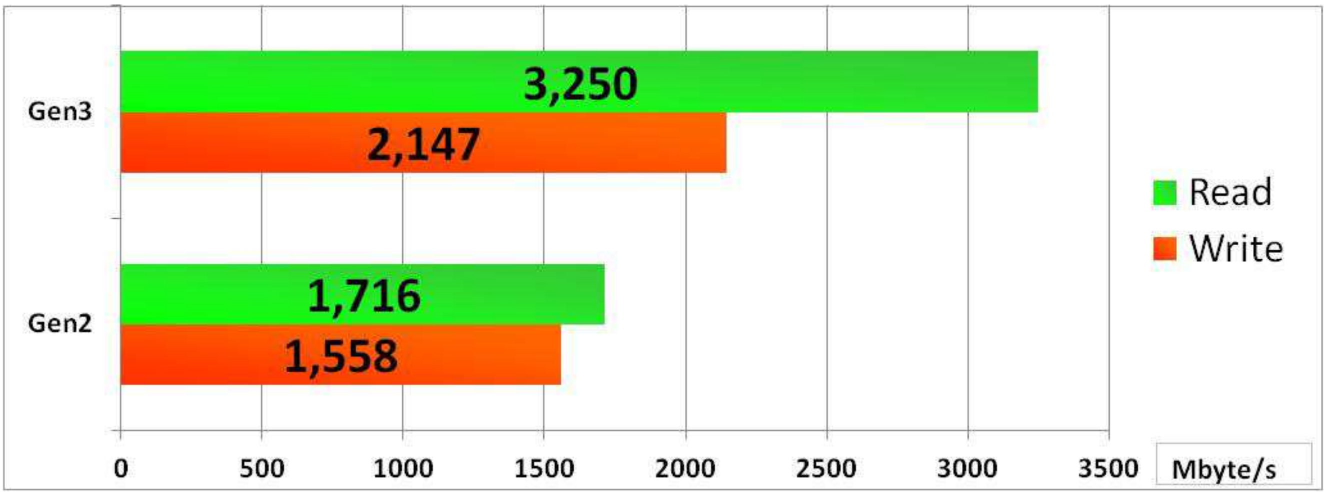


Figure 6 Test Performance of NVMe IP demo by using Samsung 960 Pro SSD

By using PCIe Gen3 on KCU105 board, write performance is about 2100 Mbyte/sec and read performance is about 3200 Mbyte/sec. Performance by using PCIe Gen2 on VC707 board is slower than Gen3. Write and read performance on Gen2 are about 1500-1700 Mbyte/sec.

7. Revision History

Revision	Date	Description
1.0	1-Jun-16	Initial Release
1.1	5-Sep-16	Add CURTESTSIZE register
1.2	6-Dec-16	Change buffer from DDR to BRAM
2.0	12-Jun-17	New NVMe-IP version
2.1	27-Jul-17	Add LFSR pattern