NVMe IP with PCIe Gen4 Soft IP reference design manual

Rev1.0 29-Jan-20

1 NVMe

NVM Express (NVMe) defines the interface for the host controller to access solid state drive (SSD) by PCI Express. NVM Express optimizes the process to issue command and completion by using only two registers (Command issue and Command completion). Besides, NVMe supports parallel operation by supporting up to 64K commands within single queue. 64K command entries improve transfer performance for both sequential and random access.

In PCIe SSD market, two standards are used, i.e. AHCI and NVMe. AHCI is the older standard to provide the interface for SATA hard disk drive while NVMe is optimized for non-volatile memory (SSD). The comparison between AHCI and NVMe protocol in more details is described in "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 device is shown in http://www.nvmexpress.org/products/

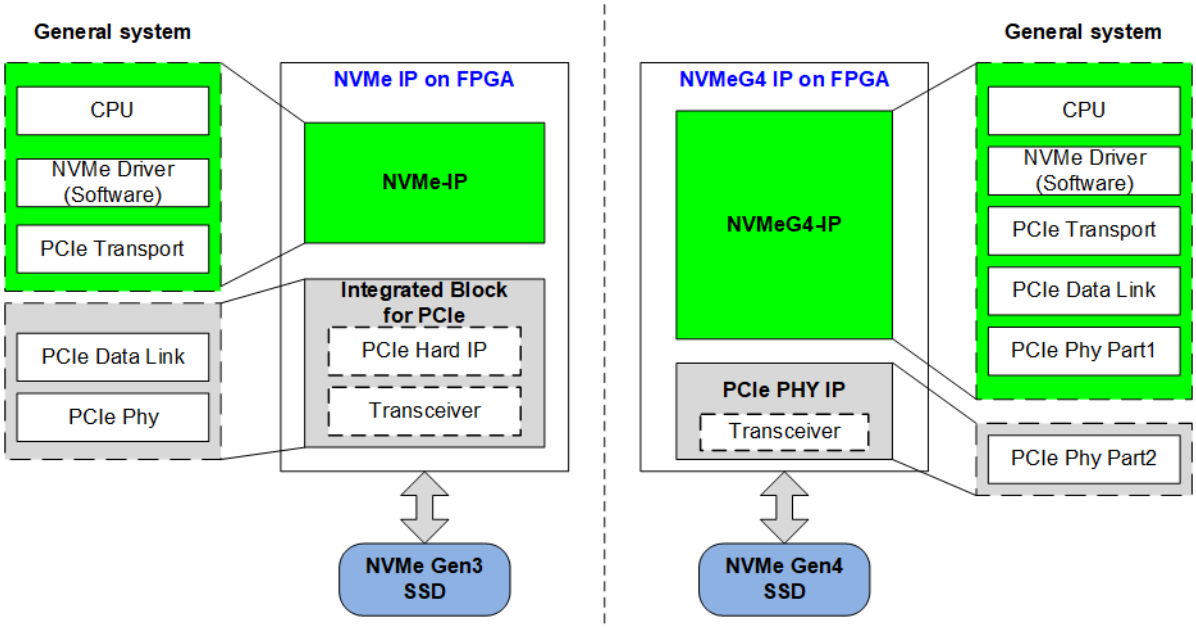

Figure 1-1 NVMe implementation

DG has the solution by using NVMe IP with Gen3 PCIe hard IP, as shown in the left side of Figure 1-1. Without using CPU and external main memory, the NVMe host controller can be implemented within FPGA with less FPGA resource and achieving ultra-speed write and read performance. However, there are some limitations for the NVMe IP with Gen3 PCIe hard IP.

First, PCIe hard IP in Ultrascale and Ultrascale+ device does not support PCIe Gen4 speed which is higher than Gen3. Second, PCIe hard IP is available in some FPGA models, so some models cannot connect with PCIe SSD by using above solution. Finally, the number of PCIe hard IP in one FPGA is limited which is effect to the maximum number of SSDs connecting to one FPGA.

DG NVMeG4 IP is the latest solution provided by Design Gateway that implements NVMe host IP without using PCIe hard IP to support PCIe Gen 4 speed. The IP includes the lower layer of PCIe protocol, i.e. Data Link Layer and some parts of Physical Layer by using the logic. This feature is known as PCIe soft IP. By integrating PCIe Soft IP and NVMe IP, NVMeG4 IP connecting with Xilinx PCIe PHY IP is the recent solution for implementing NVMe host in many FPGAs which has the transceivers for running at PCIe Gen4 speed. User interface of NVMeG4 IP are similar to NVMe IP, except the data width which are double from NVMe IP.

2 Hardware overview

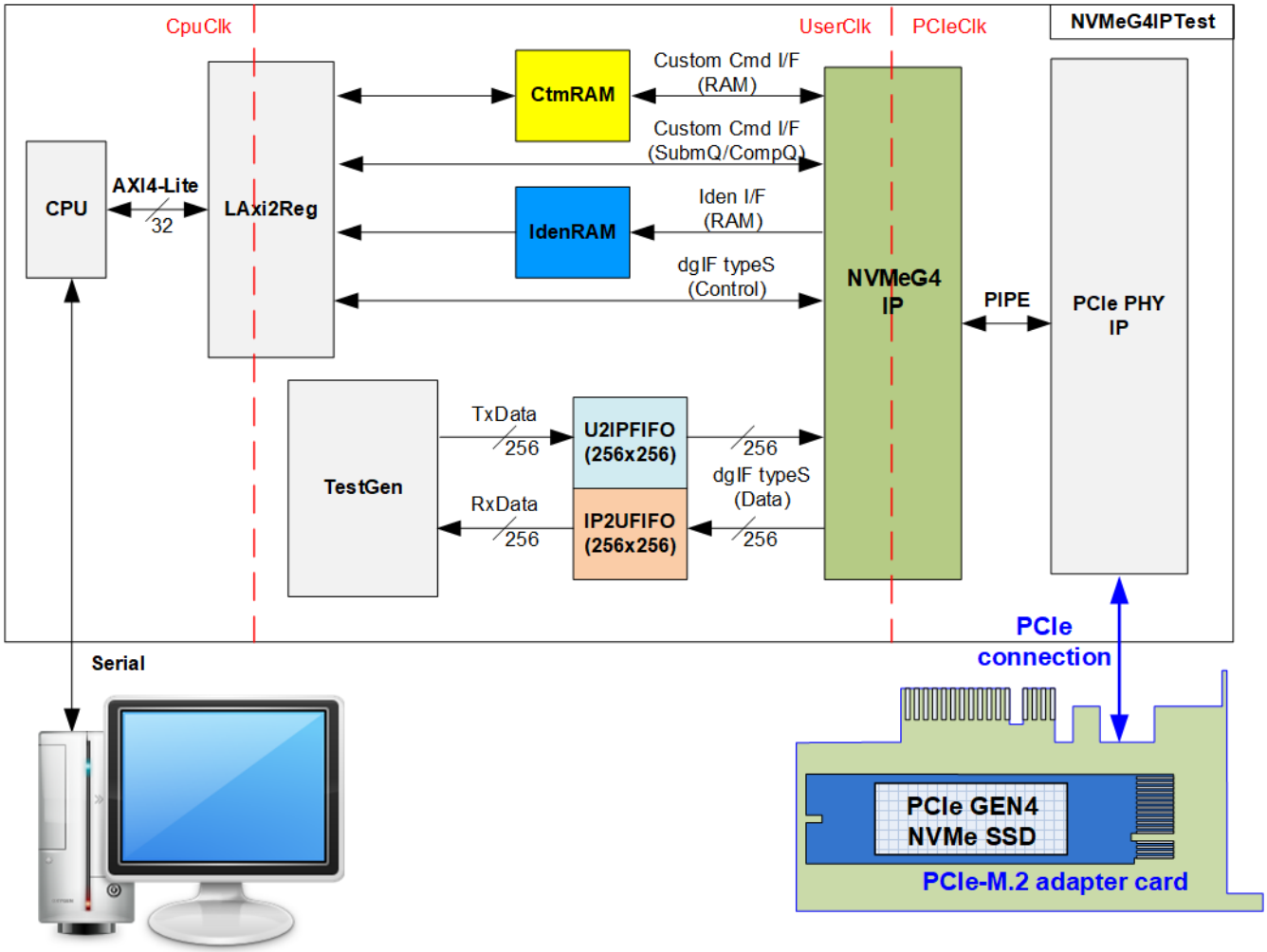

Figure 2-1 NVMeG4 IP demo hardware

Since user interface of NVMeG4 IP and NVMe IP are mostly similar, the modules for connecting user interface at control path such as CPU system of NVMeG4IPTest are designed by using the same modules in NVMe IP reference design. The modification point is in TestGen module which increasing data width from 128-bit to 256-bit. Also, Xilinx PCIe PHY IP is applied to connect to NVMeG4 IP instead of PCI hard IP, as shown in Figure 2-1.

This document describes only the modification point to run NVMeG4 IP reference design, based on NVMe IP reference design. The details of the user interface are described in NVMe IP reference design document which are provided on our website.
https://dgway.com/products/IP/NVMe-IP/dg_nvmeip_refdesign_en.pdf

NVMe IP reference design document shows the details of the logic design with timing diagram of TestGen, LAxi2Reg, RAM, and FIFO. All modules have same behavior as the design in NVMeG4IPTest module. However, there is the additional test pin of MAC layer which is output from NVMeG4 IP, named MACTestPin. This signal is necessary for system debugging when the problem is found from PCIe initialization process. So, UserReg within LAxi2Reg is slightly modified to read this signal by CPU. The signal is mapped to BA+0x0124 and BA+0x0128 as shown in Table 2-1.

Table 2-1 Register Map

Address Rd/Wr	Register Name (Label in the “nvmeg4iptest.c”)	Description
0x0000 – 0x00FF: Control signals of NVMeG4 IP and TestGen (Write access only)		
BA+0x0000	User Address (Low) Reg (USRADRL_REG)	[31:0]: Input to be start address as 512-byte unit (UserAddr[31:0] of dgIF typeS)
BA+0x0004	User Address (High) Reg (USRADRH_REG)	[15:0]: Input to be start address as 512-byte unit (UserAddr[47:32] of dgIF typeS)
BA+0x0008	User Length (Low) Reg (USRLENL_REG)	[31:0]: Input to be transfer length as 512-byte unit (UserLen[31:0] of dgIF typeS)
BA+0x000C	User Length (High) Reg (USRLENH_REG)	[15:0]: Input to be transfer length as 512-byte unit (UserLen[47:32] of dgIF typeS)
BA+0x0010	User Command Reg (USRCMD_REG)	[2:0]: Input to be user command (UserCmd of dgIF typeS for NVMeG4 IP) “000”: Identify, “001”: Shutdown, “010”: Write SSD, “011”: Read SSD, “100”: SMART, “110”: Flush, “101”/”111”: Reserved When this register is written, the command request is sent to NVMeG4 IP to start the operation.
BA+0x0014	Test Pattern Reg (PATTSEL_REG)	[2:0]: Select test pattern “000”-Increment, “001”-Decrement, “010”-All 0, “011”-All 1, “100”-LFSR
BA+0x0020	NVMe Timeout Reg (NVMTIMEOUT_REG)	[31:0]: Timeout value of NVMeG4 IP (TimeOutSet[31:0] of NVMeG4 IP)
0x0100 – 0x01FF: Status signals of NVMeG4 IP and TestGen (Read access only)		
BA+0x0100	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)
BA+0x0104	Total disk size (Low) Reg (LBASIZEL_REG)	[31:0]: LBASize0[31:0] output from NVMeG4 IP
BA+0x0108	Total disk size (High) Reg (LBASIZEH_REG)	[15:0]: LBASize0[47:32] output from NVMeG4 IP [31]: LBAMode output from NVMeG4 IP
BA+0x010C	User Error Type Reg (USRERRTYPE_REG)	[31:0]: User error status (UserErrorType[31:0] of dgIF typeS)
BA+0x0110	PCIe Status Reg (PCISTS_REG)	[7:0]: Unused for NVMeG4 IP [15:8]: MACStatus output from NVMeG4 IP
BA+0x0114	Completion Status Reg (COMPSTS_REG)	[15:0]: Status from Admin completion (AdmCompStatus[15:0] of NVMeG4 IP) [31:16]: Status from I/O completion (IOCompStatus[15:0] of NVMeG4 IP)
BA+0x0118	NVMe CAP Reg (NVMCAP_REG)	[31:0]: NVMeCAPReg[31:0] output from NVMeG4 IP
BA+0x0120	NVMe Test pin Reg (NVMTESTPIN_REG)	[31:0]: TestPin[31:0] output from NVMeG4 IP
BA+0x0124	MAC Test pin (Low) Reg (MACTESTPINL_REG)	[31:0]: MACTestPin[31:0] output from NVMeG4 IP
BA+0x0128	MAC Test pin (High) Reg (MACTESTPINH_REG)	[31:0]: MACTestPin[63:0] output from NVMeG4 IP

Address Rd/Wr	Register Name (Label in the “nvmeipg4test.c”)	Description
0x0100 – 0x01FF: Status signals of NVMeG4 IP and TestGen (Read access only)		
BA+0x0130 – BA+0x014C	Expected value Word0-7 Reg (EXPPATW0-W7_REG)	256-bit of the expected data at the 1st failure data in Read command 0x0130: Bit[31:0], 0x0134[31:0]: Bit[63:32], …, 0x014C[31:0]: Bit[255:224]
BA+0x0150 – BA+0x016C	Read value Word0-7 Reg (RDPATW0-W7_REG)	256-bit of the read data at the 1st failure data in Read command 0x0150: Bit[31:0], 0x0154[31:0]: Bit[63:32], …, 0x016C[31:0]: Bit[255:224]
BA+0x0170	Data Failure Address(Low) Reg (RDFAILNOL_REG)	[31:0]: Bit[31:0] of the byte address of the 1st failure data in Read command
BA+0x0174	Data Failure Address(High) Reg (RDFAILNOH_REG)	[24:0]: Bit[56:32] of the byte address of the 1st failure data in Read command
BA+0x0178	Current test byte (Low) Reg (CURTESTSIZEL_REG)	[31:0]: Bit[31:0] of the current test data size in TestGen module
BA+0x017C	Current test byte (High) Reg (CURTESTSIZEH_REG)	[24:0]: Bit[56:32] of the current test data size of TestGen module
Other interfaces (Custom command of NVMeG4 IP, IdenRAM, and Custom RAM)		
BA+0x0200 – BA+0x023F Wr	Custom Submission Queue Reg (CTMSUBMQ_REG)	[31:0]: Submission queue entry of SMART and Flush command. Input to be CtmSubmDW0-DW15 of NVMeG4 IP. 0x200: DW0, 0x204: DW1, …, 0x23C: DW15
BA+0x0300 – BA+0x030F Rd	Custom Completion Queue Reg (CTMCOMPQ_REG)	[31:0]: Submission queue entry of SMART and Flush command. Input to be CtmSubmDW0-DW15 of NVMeG4 IP. 0x200: DW0, 0x204: DW1, …, 0x23C: DW15
BA+0x0800 Rd	IP Version Reg (IPVERSION_REG)	[31:0]: CtmCompDW0-DW3 output from NVMeG4 IP. 0x300: DW0, 0x304: DW1, …, 0x30C: DW3
BA+0x2000 – BA+0x2FFF Rd	Identify Controller Data (IDENCTRL_REG)	[31:0]: IP version number (IPVersion[31:0] of NVMeG4 IP)
BA+0x3000 – BA+0x3FFF Rd	Identify Namespace Data (IDENNAME_REG)	4Kbyte Identify Controller Data Structure
BA+0x4000 – BA+0x5FFF Wr/Rd	Custom command Ram (CTMRAM_REG)	4Kbyte Identify Namespace Data Structure

3 CPU Firmware

CPU Firmware on NVMeG4 IP reference design is slightly modified from NVMe IP reference design in PCIe initialization sequence. The step to check PCIe link up signal is removed in NVMeG4 IP. After reset sequence is finished, the IP begins the initialization sequence. CPU runs the following step to initialize the system.

1) CPU initializes UART and Timer parameters.
2) CPU waits until IP completes PCIe and NVMe initialization process by monitoring IP busy flag (USRSTS_REG[0]=’0’). When some errors are found, the process stops and displays the error message.
3) CPU displays the main menu. There are six menus for running six commands, i.e. Identify, Write, Read, SMART, Flush, and Shutdown.

The details of CPU firmware to operate all commands are similar to NVMe IP reference design. Please see more details from NVMe IP reference design document.

4 Example Test Result

The example test result when running demo system by using 1 TB Aorus NVMe Gen4 SSD sequentially transferring 32 GB data is shown in Figure 4-1.

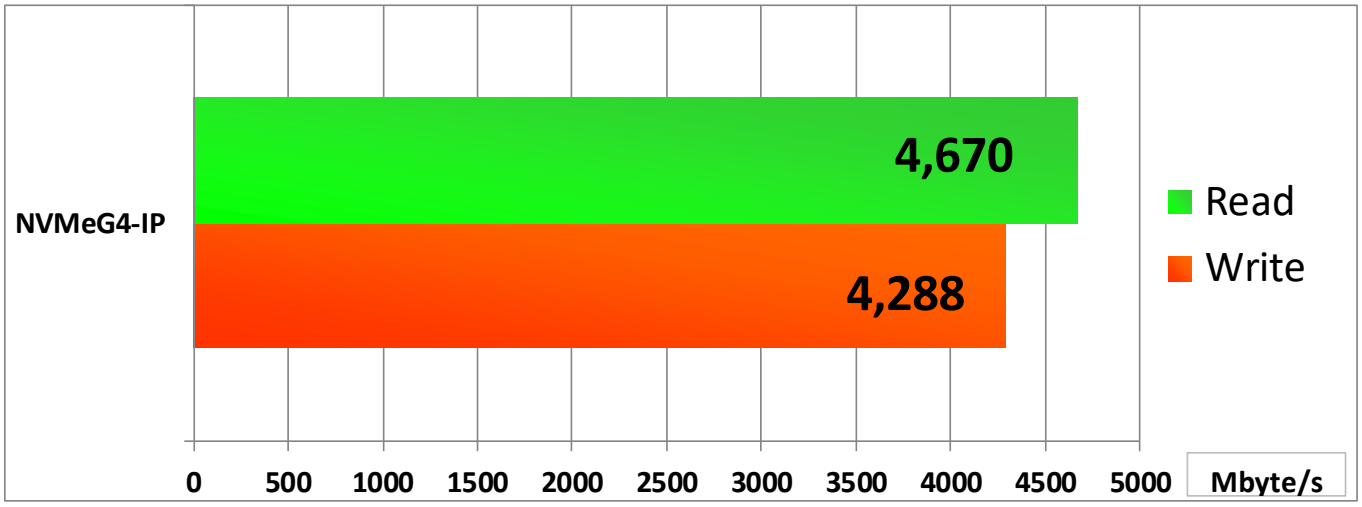

Figure 4-1 Test Performance of NVMeG4 IP demo by using Aorus NVMe Gen4 SSD

By running NVMeG4-IP on VCU118 board, write performance is about 4300 Mbyte/sec and read performance is about 4700* Mbyte/sec.

Note: The result of the read performance is measured by customized NVMeG4-IP core with extended buffer size to achieve the best performance matched with the SSD characteristic.

5 Revision History

Revision	Date	Description
1.0	29-Jan-20	Initial Release