exFAT IP for NVMe reference design manual

Rev1.5 14-Mar-22

1 Introduction

In the hardware system, data stream can be stored to the disk by using raw data or file system. Using raw data, the data is allocated in the disk by physical address. When many data types are stored in one disk, user needs to assign the different address for each data group. Without standard, each system defines its own data structure to arrange data groups. It is not practical for the Host to support every data structure for reading data from some systems.

As a result, file system is created to manage data in the disk by setting up the table to be an index of data. The data is separated into many groups. Each group is called a “file”. For system flexibility, one file has some information to represent itself such as file name, file type, file size, and physical address of file data. File information helps user to know file structure and free space in the disk.

exFAT is standard file system which is common support in many platforms. Comparing to FAT32 file system, exFAT file system improves many features. First, exFAT supports more than 4 GB file size and supports more than 2 TB disk capacity. Next, Name hash of file name is implemented to improve search function. Besides, checksum is applied in system data area to increase data reliability.

Generally, file system is implemented as standard library run on CPU. To write/read file by using CPU software, it has overhead time to access file header for reading file allocation before writing or reading file data. So, write/read performance when running file system by using CPU software is reduced, comparing to using raw data format which does not have the header.

exFAT-IP is the hardware which designs file system data structure following exFAT standard. exFAT-IP reduces the overhead time to access file header. As a result, write/read performance when using exFAT-IP is almost equal to using raw data format which is run by DG NVMe(G3)-IP.

The hardware design of exFAT-IP for NVMe demo is different from the hardware design of NVMe(G3)-IP design (raw data), as shown in Figure 1-1.

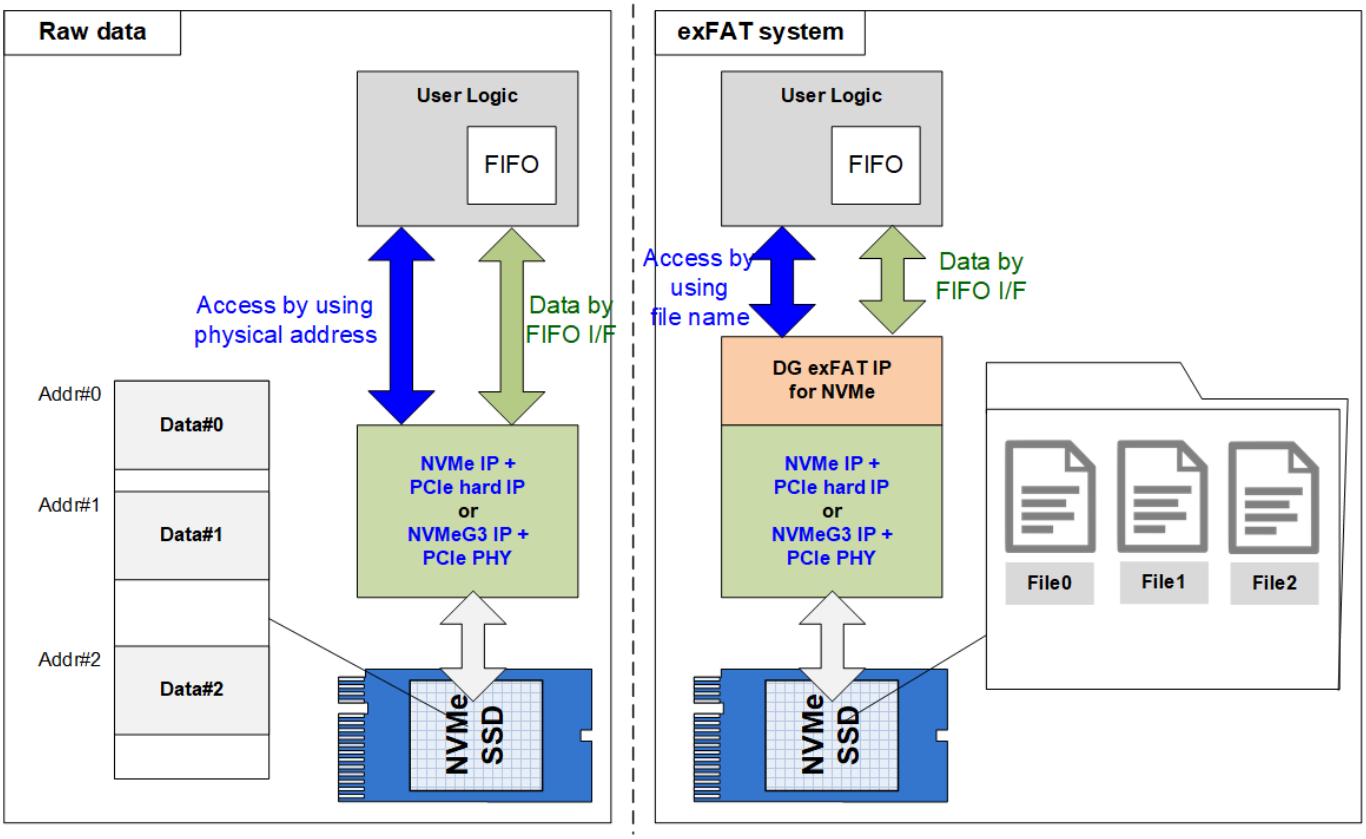

Figure 1-1 Hardware system for raw data and file system

Comparing to raw data system in the left side of Figure 1-1, exFAT system includes exFAT-IP for NVMe to connect between User Logic and NVMe(G3)-IP. The parameter of user interface changes from physical parameters (address and length) to be file parameters (file name and the number of files). However, the data interface of raw data and exFAT system are similar by using general FIFO interface. More details of exFAT-IP for NVMe reference design are described in the next topic.

2 Hardware overview

The reference design of exFAT-IP for NVMe is modified from NVMe-IP or NVMeG3-IP reference design by including exFAT-IP. Also, the control interface is updated from physical parameters to be file parameters. The updated part is displayed as the blue color in Figure 2-1.

More details of NVMe-IP reference design are described in following document.
https://dgway.com/products/IP/NVMe-IP/dg_nvmeip_refdesign_en.pdf
https://dgway.com/products/IP/NVMe-IP/dg_nvmeip_instruction_en.pdf

More details of NVMeG3-IP reference design are described in following document.
https://dgway.com/products/IP/NVMe-IP/dg_nvmeg3ip_refdesign_xilinx_en.pdf

Updated lists from NVMe-IP and NVMeG3-IP reference design are as follows.

1) File parameters for control interface of exFAT-IP are received from TestGen module.
2) The registers inside LAxi2Reg are updated following the control signals of the test system.
3) CPU firmware is updated to receive file parameters from the user and then converts to be the control signals of the hardware through AXI4-Lite bus. The example parameters from user are file name, file size, the number of files, created date, and created time.

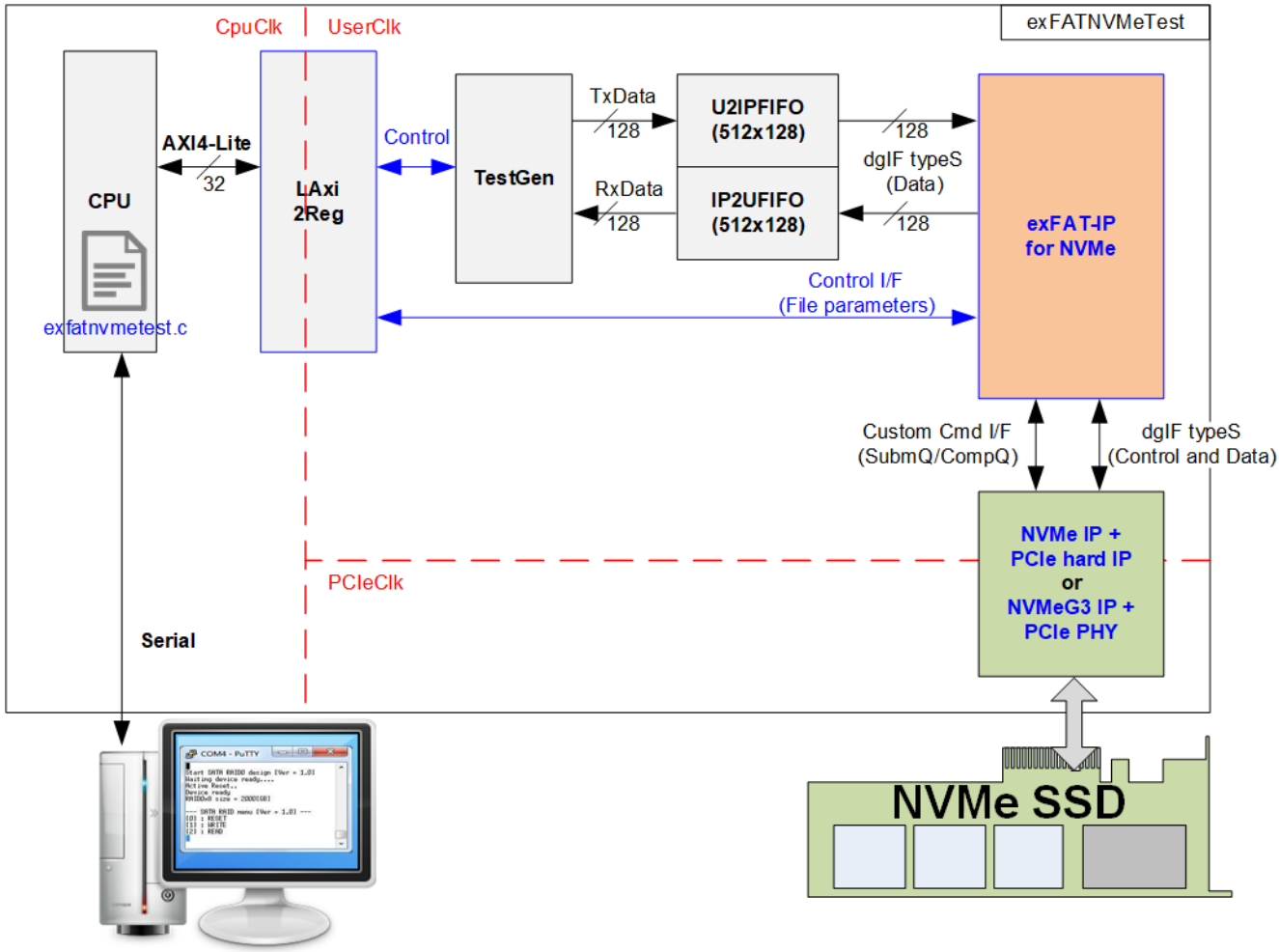

Figure 2-1 exFAT-IP for NVMe demo system

Four commands are supported by exFAT-IP, i.e., Format, Write file, Read file, and Shutdown. The transfer performance is displayed on Serial console as a test result after finishing Write file or Read file command. After finishing Write file operation, the user can plug-in the NVMe SSD to the other hosts which support exFAT system such as PC for reading and verifying test data file in the NVMe SSD.

Three clock domains are applied in the system, i.e., CpuClk, UserClk, and PCIeClk.

1) CpuClk is the clock domain of CPU and its peripherals. This clock must be stable clock which can be different from the other hardware interface.
2) For NVMe-IP, PCIeClk is the clock output from PCIe hard IP to synchronous with data stream of 128-bit AXI4 stream bus. PCIeClk is equal to 250 MHz when using 4-lane PCIe Gen3 while it is equal to 125 MHz when using 4-lane PCIe Gen2.

   For NVMeG3-IP, PCIeClk is the clock output from PCIe PHY to synchronous with 128-bit PIPE interface. PCIe PHY is equal to 250 MHz for 4-lane PCIe Gen3.
3) UserClk is the example user clock domain which can be different from other clock domains for being the main clock domain of the user interface of exFAT-IP, NVMe(G3)-IP, FIFO, and TestGen. According to NVMe-IP and NVMeG3-IP datasheet, clock frequency of UserClk must be more than or equal to PCIeClk. In this reference design, UserClk is equal to 275/280 MHz for PCIe Gen3 or 200 MHz for PCIe Gen2.

More details of the hardware in exFAT-IP for NVMe demo design are described as follows.

2.1 TestGen

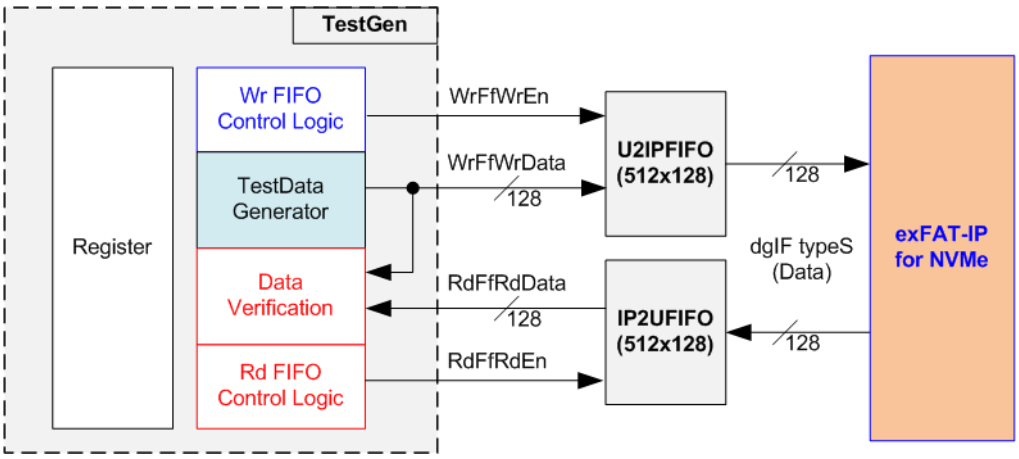

Figure 2-2 TestGen interface

TestGen module is the test logic to send test data to exFAT-IP through U2IPFIFO when operating Write file command. Also, the test data is fed to be the expected value to verify the received data from exFAT-IP through IP2UFIFO when operating Read file command. Control logic asserts Write enable or Read enable to '1' when the FIFOs are ready. Data bandwidth of TestGen is matched to exFAT-IP by running at the same clock and using the same data bus size. Therefore, exFAT-IP transfers data with U2IPFIFO and IP2UFIFO without waiting FIFO ready. As a result, the test logic shows the best performance to write and read data with the device through exFAT-IP.

Register file in the TestGen receives test parameters from user, i.e., file size, file name, the number of files, the command, verification enable, and test pattern selector. The internal logic includes the counter to control total transfer size of test data. The details of hardware logic of TestGen are shown in Figure 2-3.

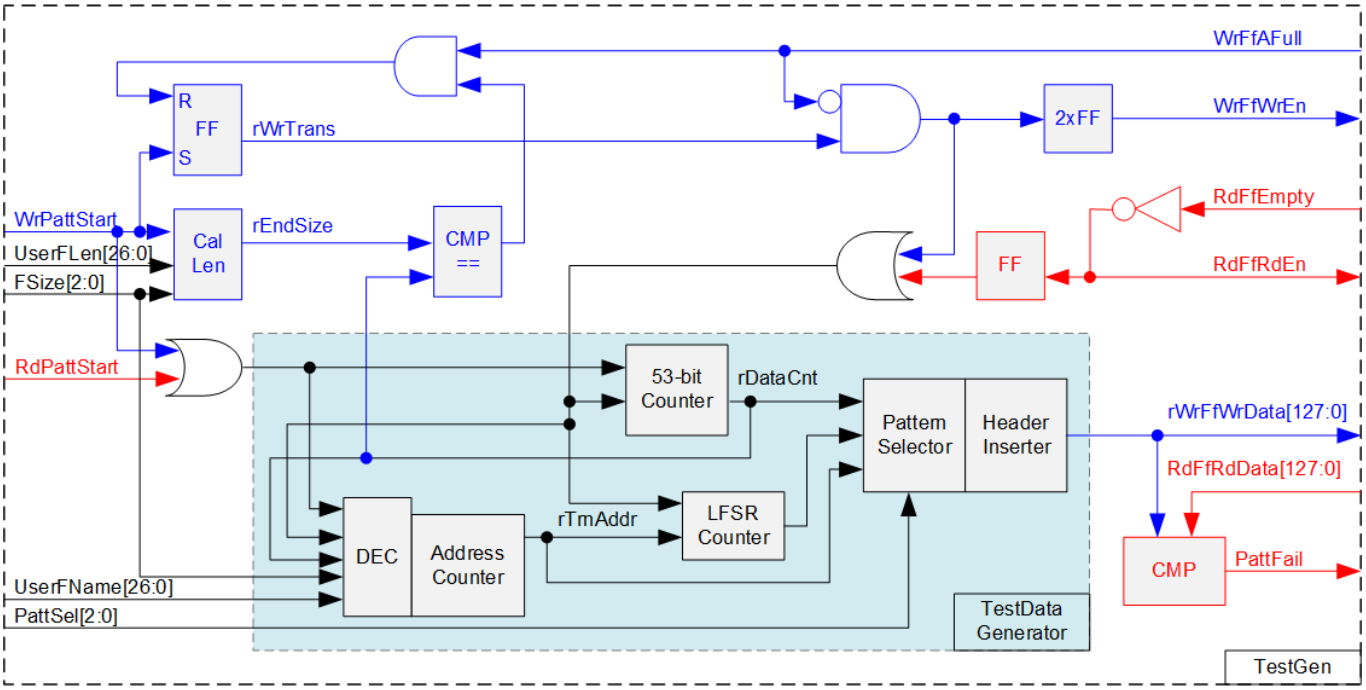

Figure 2-3 TestGen hardware

As shown in the right side of Figure 2-3, flow control signals of FIFO are WrFfAFull and RdFfEmpty. When FIFO is almost full during write operation (WrFfAFull='1'), WrFfWrEn is de-asserted to '0' to pause data sending to FIFO. For read operation, when FIFO has data (RdFfEmpty='0'), the logic reads data from FIFO to compare with the expected data by asserting RdFfRdEn to '1'.

The logic in the left side of Figure 2-3 is designed to count transfer size. When total data count is equal to the end size (calculated by UserFLen x File size decoded from FSize), write enable or read enable of FIFO is de-asserted to '0'. So, the total data count to write FIFO or read FIFO is controlled by user.

The lower side of Figure 2-3 shows the details to generate test data for writing to FIFO or verifying with data from FIFO. There are five patterns to generate, i.e., all zero, all one, 32-bit incremental data, 32-bit decremental data, and LFSR counter, selected by Pattern Selector. When creating all zero or all one pattern, every bit of data is fixed zero or one respectively. While other patterns are designed by separating the data as two parts to create unique test data in every 512-byte data.

As shown in Figure 2-4, 512-byte data consists of 64-bit header in Dword#0 and Dword#1 and the test data in remaining words of 512-byte data (Dword#2 – Dword#127).

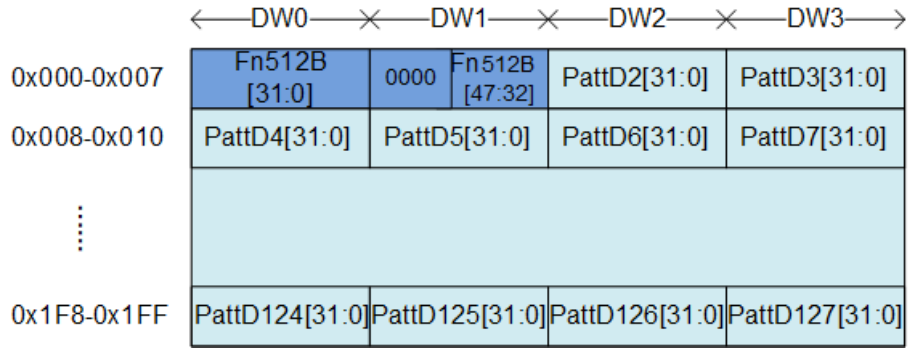

Figure 2-4 Test pattern format in each 512-byte data for Increment/Decrement/LFSR pattern

64-bit header is created by calculating the unique value for each 512-byte data, controlled by Address counter inside TestData Generator. The initial value of the address counter is calculated by UserFName x File size, decoded from FSize. After that, the address counter is increased when finishing transferring 512-byte data.

Test data in other bits (DW#2 – DW#127) can be selected as three patterns, i.e., 32-bit incremental data, 32-bit decremental data, and LFSR counter. The 32-bit incremental data is designed by using 53-bit counter while the decrement data can be designed by connecting NOT logic to increment data. The LFSR pattern is designed by using LFSR counter. The equation of LFSR is x^31 + x^21 + x + 1. Data bus size of TestGen is 128-bit, so four 32-bit LFSR data must be generated within one clock. The logic to design LFSR must use look-ahead style to generate four LFSR data in the same clock.

Test data is fed to be write data to the FIFO or the expected data comparing with the read data from FIFO. Fail flag is asserted to ‘1’ when data verification is failed. The example of timing diagram to write data to FIFO is shown as follows.

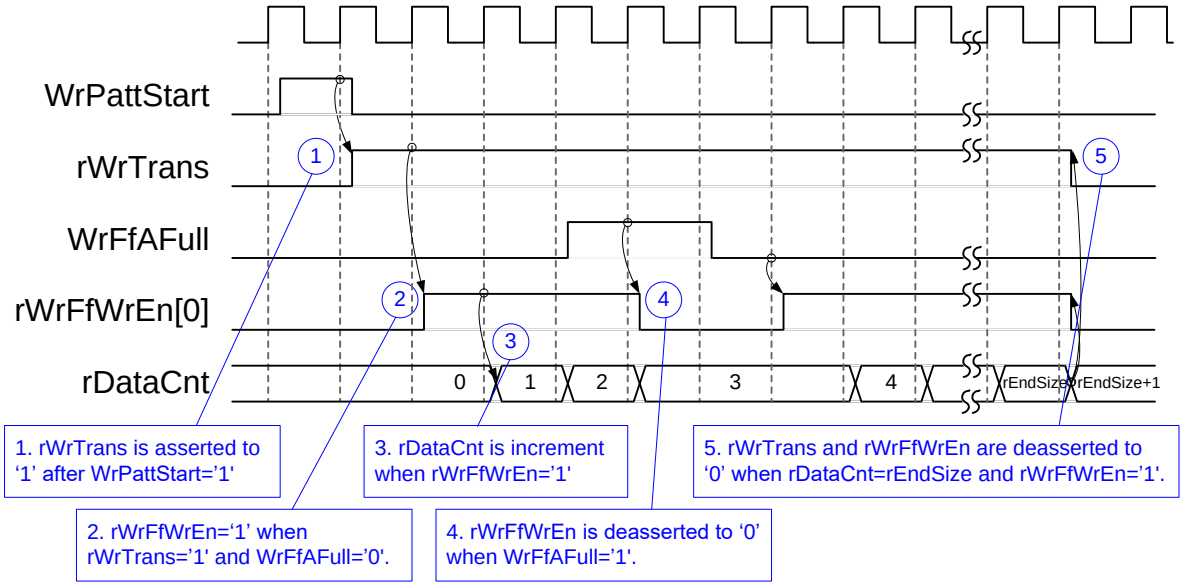

Figure 2-5 Timing diagram of Write operation in TestGen

1) WrPattStart is asserted to ‘1’ for one clock cycle when user sets the register to start Write file operation. In the next clock, rWrTrans is asserted to ‘1’ to enable the control logic for generating write enable to FIFO.
2) Write enable to FIFO (rWrFfWrEn) is asserted to ‘1’ when two conditions are met. First, rWrTrans must be asserted to ‘1’ during running the write operation being active. Second, the FIFO must not be full by monitoring WrFfAFull=’0’.
3) The write enable is fed back to be counter enable to count total data in the write operation.
4) If FIFO is almost full (WrFfAFull=’1’), the write process is paused by de-asserting rWrFfWrEn to ‘0’.
5) When total data count is equal to the set value, rWrTrans is de-asserted to ‘0’. At the same time, rWrFfWrEn is also de-asserted to ‘0’ to stop data generating.

For read timing diagram, read enable of FIFO is controlled by empty flag of FIFO. Comparing to write enable, the read enable signal is not stopped by total count and not started by start flag. When the read enable is asserted to ‘1’, the data counter and the address counter are also increased for counting total data and generating the header of expect value.

2.2 exFAT

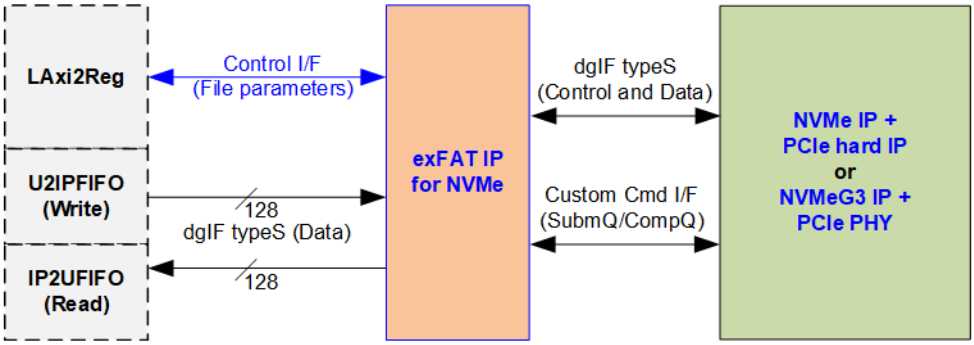

Figure 2-6 exFAT hardware

As shown in Figure 2-6, the user interface of exFAT-IP is split into two groups, i.e., the control interface and the data interface. The control interface is connected to LAxi2Reg to receive file parameters from user through Serial console. The data interface is 128-bit data bus and connects with U2IPFIFO and IP2UFIFO. Another side of exFAT-IP is connected to NVMe-IP or NVMeG3-IP.

2.2.1 exFAT-IP for NVMe

exFAT-IP implements the logic to handle data in NVMe device following exFAT file system. Data bus size is 128-bit. More details of exFAT-IP for NVMe are described in datasheet.
https://dgway.com/products/IP/NVMe-IP/dg_exfatip_nvme_data_sheet_en.pdf

exFAT-IP for NVMe has two interfaces for connecting with NVMe(G3)-IP, i.e., dgIF typeS (both control and data interface) and Custom command interface (Submission Queue and Completion Queue). RAM interface for Identify command and custom command of NVMe(G3)-IP is unconnected in the reference design. If user needs to know the information from Identify command, IdenRAM can be connected, similar to NVMe(G3)-IP standard reference design. Custom RAM for SMART command is not supported in the standard exFAT-IP design. The additional feature can be designed as customized support.

2.2.2 NVMe(G3)-IP

NVMe(G3)-IP implements NVMe protocol of the host side to directly access one NVMe device without PCIe switch connection. User interface is simply designed by using dgIF typeS format. Both NVMe-IP and NVMeG3-IP have the same NVMe features. The difference between these two IPs is in low layer boundary of PCIe.

Low-level interface of 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.
https://dgway.com/products/IP/NVMe-IP/dg_nvme_ip_data_sheet_en.pdf

Low-level interface of NVMeG3-IP is designed to connect with PCIe PHY which is Xilinx IP Core. Therefore, it is the solution for the device that does not have PCIe hard IP. More details of NVMeG3-IP are described in datasheet.
https://dgway.com/products/IP/NVMe-IP/dg_nvmeg3_ip_data_sheet_xilinx_en.pdf

2.2.3 PCIe

a) Integrated Block for PCIe

This block is hard IP in Xilinx device which implements Physical, Data Link, and Transaction Layers of PCIe specification. More details are described in Xilinx document.
PG054: 7 Series FPGAs Integrated Block for PCI Express
PG023: Virtex-7 FPGA Gen3 Integrated Block for PCI Express
PG156: UltraScale Devices Gen3 Integrated Block for PCI Express
PG213: UltraScale+ Devices Integrated Block for PCI Express

b) PCIe PHY IP

This module is Xilinx IP Core which implements Physical Layer of PCIe specification. The user interface is PHY Interface for PCI Express (PIPE). To operate with NVMeG3-IP, PCIe PHY uses Lane width to x4 and Link speed to 8.0 GT/s. More details of PCIe PHY IP are described in “PG239: PCI Express PHY” document.
https://www.xilinx.com/support/documentation/ip_documentation/pcie_phy/v1_0/pg239-pcie-phy.pdf

2.3 CPU and Peripherals

32-bit AXI4-Lite bus is applied to be the bus interface for CPU accessing the peripherals such as Timer and UART. To control and monitor the test logic of exFAT-IP, the test logic is connected to CPU as a peripheral on 32-bit AXI4-Lite bus. CPU assigns the different base address and the address range for each peripheral.

In the reference design, the CPU system is built with one additional peripheral to access the test logic. Therefore, the hardware logic must be designed to support AXI4-Lite bus standard for writing and reading the register. LAxi2Reg module is designed to connect the CPU system as shown in Figure 2-7.

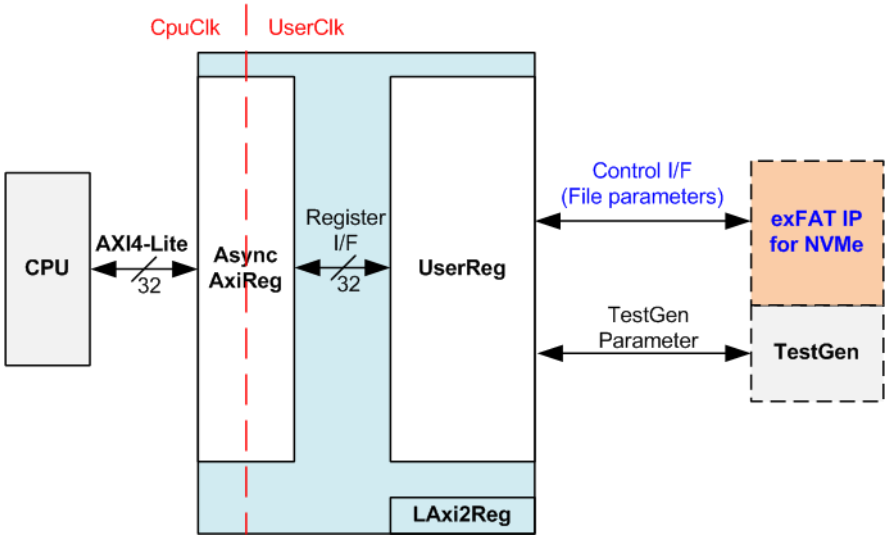

Figure 2-7 CPU and peripherals hardware

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

UserReg includes the register file of the parameters and the status signals to control the other modules, i.e., exFAT-IP and TestGen. More details of AsyncAxiReg and UserReg are described as follows.

2.3.1 AsyncAxiReg

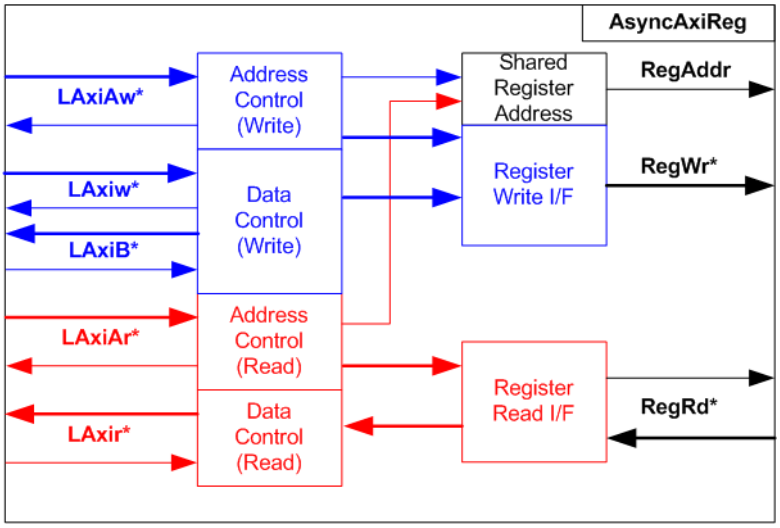

Figure 2-8 AsyncAxiReg Interface

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

According to AXI4-Lite standard, the write channel and the read channel are operated independently. Also, the control and data interface of each channel are run separately. So, the logic inside AsyncAxiReg to interface with AXI4-lite bus is split into four groups, i.e., Write control logic, Write data logic, Read control logic, and Read data logic as shown in the left side of Figure 2-8. Write control I/F and Write data I/F of AXI4-Lite bus are latched and transferred to be Write register interface with clock domain crossing registers. Similarly, Read control I/F of AXI4-Lite bus are latched and transferred to be Read register interface. While the returned data from Register Read I/F is transferred to AXI4-Lite bus by using clock domain crossing registers. In Register interface, RegAddr is shared signal for write and read access. Therefore, it loads the value from LAxiAw for write access or LAxiAr for read access.

The simple register interface is compatible with single-port RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq and RdValid signals for controlling read latency time. The address of register interface is shared for write and read transaction, so user cannot write and read the register at the same time. The timing diagram of the register interface is shown in Figure 2-9.

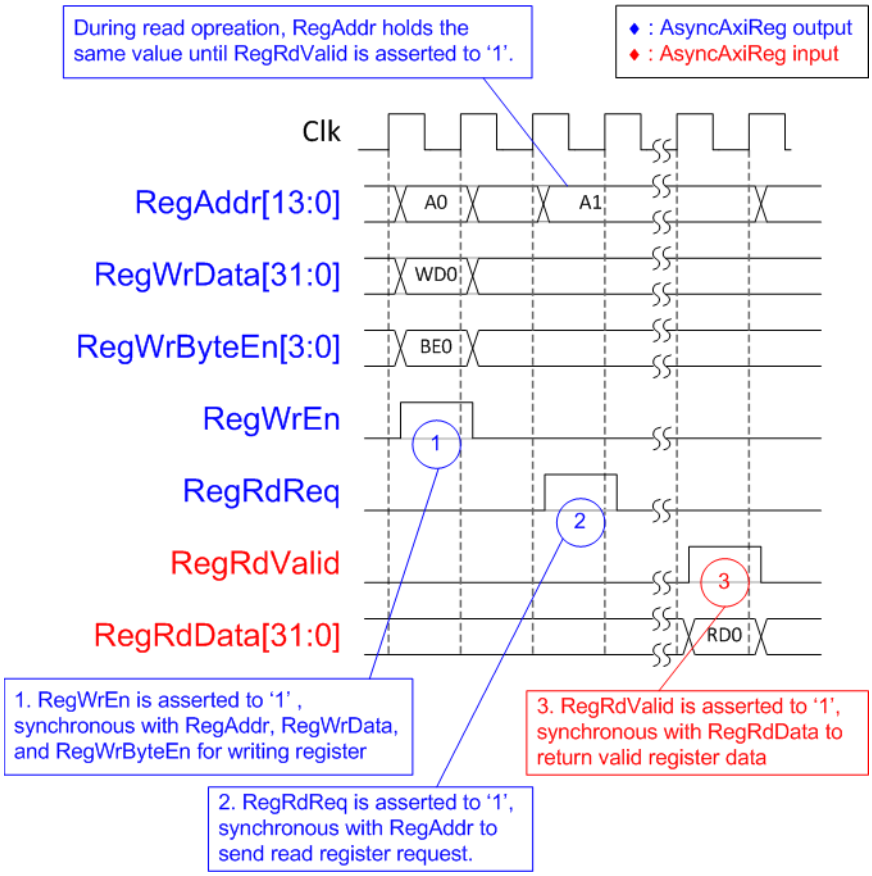

Figure 2-9 Register interface timing diagram

1) To write register, the timing diagram is similar to single-port RAM interface. RegWrEn is asserted to '1' with the valid signal of RegAddr (Register address in 32-bit unit), RegWrData (write data of the register), and RegWrByteEn (the write byte enable). Byte enable has four bits to be the byte data valid. Bit[0], [1], [2], and [3] are equal to '1' when RegWrData[7:0], [15:8], [23:16], and [31:24] are valid respectively.
2) To read register, AsyncAxiReg asserts RegRdReq to '1' with the valid value of RegAddr. 32-bit data must be returned after receiving the read request. The slave must monitor RegRdReq signal to start the read transaction. During read operation, the address value (RegAddr) does not change the value until RegRdValid is asserted to '1'. Therefore, the address can be used for selecting the returned data by using multiple layers of multiplexer.
3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAxiReg forwards the read value to LAxir* interface.

2.3.2 UserReg

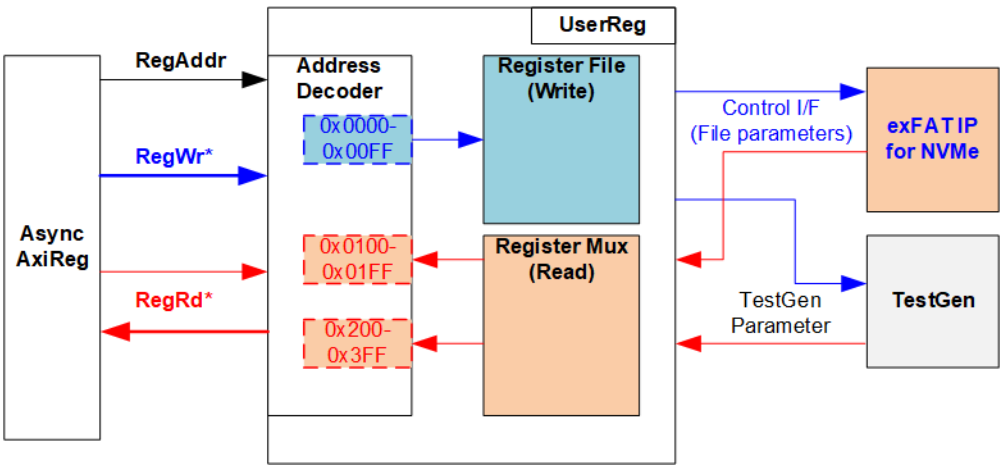

Figure 2-10 UserReg Interface

The address range to map to UserReg is split into three areas, as shown in Figure 2-10.

1) 0x0000 – 0x00FF: mapped to set the test parameters of exFAT-IP and TestGen. This area is write access only.
2) 0x0100 – 0x01FF: mapped to read the status of exFAT-IP. This area is read access only.
3) 0x0200 – 0x02FF: mapped to read the status of TestGen. This area is read access only.

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

To read register, two-step multiplexer is designed to select the read data within each address area. The lower bit of RegAddr is applied in each Register area to select the data. Next, the address decoder uses the upper bit to select the read data from each area for returning to CPU. Totally, the latency of read data is equal to two clock cycles. Therefore, RegRdValid is created by RegRdReq with asserting two D Flip-flops. More details of the address mapping within UserReg module are shown in Table 2-1.

Table 2-1 Register Map

Address Rd/Wr	Register Name (Label in the “exfatnvmetest.c”)	Description
0x0000 – 0x00FF: Control signals of exFAT-IP and TestGen (Write access only)		
BA+0x000	User File Name Reg (USRFNAME_INTREG)	[26:0]: Input to be UserFName of exFAT-IP for NVMe
BA+0x004	User File Length Reg (USRFLEN_INTREG)	[26:0]: Input to be UserFLen of exFAT-IP for NVMe
BA+0x008	File Size Reg (USRFSIZE_INTREG)	[2:0]: Input to be FSize of exFAT-IP for NVMe. Comparing to DFSIZE_INTREG, this input is used to set File size to exFAT-IP for NVMe during running Format command.
BA+0x00C	Created Date and Time Reg (DATETIME_INTREG)	[4:0]: Input to be FTimeS of exFAT-IP for NVMe [10:5]: Input to be FTimeM of exFAT-IP for NVMe [15:11]: Input to be FTimeH of exFAT-IP for NVMe [20:16]: Input to be FDateD of exFAT-IP for NVMe [24:21]: Input to be FDateM of exFAT-IP for NVMe [31:25]: Input to be FDateY of exFAT-IP for NVMe
BA+0x010	User Command Reg (USRCMD_INTREG)	[1:0]: Input to be UserCmd of exFAT-IP for NVMe When this register is written, the design asserts UserReq=’1’ (command request) to exFAT-IP for NVMe to start the operation.
BA+0x014	Pattern Select Reg (PATTSEL_INTREG)	[2:0]: Select test pattern “000”-Increment, “001”-Decrement, “010”-All 0, “011”-All 1, “100”-LFSR
0x0100 – 0x01FF: Status signals of exFAT-IP (Read access only)		
BA+0x100	User Status Reg (USRSTS_INTREG)	[0]: Mapped to UserBusy of exFAT-IP for NVMe [1]: Mapped to UserError of exFAT-IP for NVMe [2]: Data verification fail (‘0’: Normal, ‘1’: Error)
BA+0x104	Total file capacity Reg (TOTALFCAP_INTREG)	[26:0]: Mapped to TotalFCap[26:0] of exFAT-IP for NVMe
BA+0x108	User Error Type Reg (USRERRTYPE_INTREG)	[31:0]: Mapped to UserErrorType[31:0] of exFAT-IP for NVMe
BA+0x10C	exFAT IP Test pin (Low) Reg (TESTPINL_INTREG)	[31:0]: Mapped to TestPin[31:0] of exFAT-IP for NVMe
BA+0x110	exFAT IP Test pin (High) Reg (TESTPINH_INTREG)	[31:0]: Mapped to TestPin[63:32] of exFAT-IP for NVMe
BA+0x114	Directory capacity Reg (DIRCAP_INTREG)	[19:0]: Mapped to DirCap[19:0] of exFAT-IP for NVMe
BA+0x118	File Size in the disk Reg (DFSIZE_INTREG)	[2:0]: Mapped to DiskFsize of exFAT-IP for NVMe. Comparing to USRFSIZE_INTREG, this is the current file size which exFAT-IP reads from the device.
BA+0x11C	Total file in the disk Reg (DFNUM_INTREG)	[26:0]: Mapped to DiskFnum of exFAT-IP for NVMe
BA+0x120	Disk Capacity (Low) Reg (DFSIZEL_INTREG)	[31:0]: Mapped to LBASize(bit[31:0]) of NVMe(G3)-IP to check total capacity of the disk.
BA+0x124	Disk Capacity (High) reg (DFSIZEH_INTREG)	[15:0]: Mapped to LBASize(bit[47:32]) of NVMe(G3)-IP to check total capacity of the disk.
BA+0x128	Completion Status Reg (COMPSTS_INTREG)	Completion Status from NVMe(G3)-IP [15:0]: Admin completion Status (AdmCompStatus[15:0]) [31:16]: I/O completion Status (IOCompStatus[15:0])
BA+0x12C	NVMe CAP Reg (NVMCAP_INTREG)	[31:0]: NVMeCAPReg[31:0] output from NVMe(G3)-IP
BA+0x130	NVMe Test pin Reg (NVMTESTPIN_INTREG)	[31:0]: Mapped to TestPin[31:0] NVMe(G3)-IP

Address Rd/Wr	Register Name (Label in the “exfatnvmetest.c”)	Description
0x0100 – 0x01FF: Status signals of exFAT-IP (Read access only)		
BA+0x134	MAC Test pin (Low) Reg (MACTESTPINL_INTREG)	[31:0]: Mapped to MACTestPin[31:0] of NVMeG3-IP This register.is not applied for NVMe-IP.
BA+0x138	MAC Test pin (High) Reg (MACTESTPINH_INTREG)	[31:0]: Mapped to MACTestPin[63:32] of NVMeG3-IP This register.is not applied for NVMe-IP.
0x0200 – 0x02FF: Status signals of TestGen (Read access only)		
BA+0x200	Expected value Word0 Reg (EXPPATW0_INTREG)	[31:0]: Bit[31:0] of the expected data at the 1st failure data in Read file command
BA+0x204	Expected value Word1 Reg (EXPPATW1_INTREG)	[31:0]: Bit[63:32] of the expected data at the 1st failure data in Read file command
BA+0x208	Expected value Word2 Reg (EXPPATW2_INTREG)	[31:0]: Bit[95:64] of the expected data at the 1st failure in Read file command
BA+0x20C	Expected value Word3 Reg (EXPPATW3_INTREG)	[31:0]: Bit[127:96] of the expected data at the 1st failure in Read file command
BA+0x210	Read value Word0 Reg (RDPATW0_INTREG)	[31:0]: Bit[31:0] of the read data at the 1st failure data in Read file command
BA+0x214	Read value Word1 Reg (RDPATW1_INTREG)	[31:0]: Bit[63:32] of the read data at the 1st failure data in Read file command
BA+0x218	Read value Word2 Reg (RDPATW2_INTREG)	[31:0]: Bit[95:64] of the read data at the 1st failure data in Read file command
BA+0x21C	Read value Word3 Reg (RDPATW3_INTREG)	[31:0]: Bit[127:96] of the read data at the 1st failure data in Read file command
BA+0x220	Failure Byte Address (Low) Reg (FAILADDRL_INTREG)	[31:0]: Bit[31:0] of the byte address in the file at the 1st failure data in Read file command
BA+0x224	Failure Byte Address (High) Reg (FAILADDRH_INTREG)	[6:0]: Bit[38:32] of the byte address in the file at the 1st failure data in Read file command
BA+0x228	Failure File Name Reg (FAILFNAME_INTREG)	[26:0]: Filename of the 1st failure data
BA+0x22C	Current test byte (Low) Reg (CURTESTSIZEL_INTREG)	[31:0]: Bit[31:0] of the current test data size in TestGen module
BA+0x230	Current test byte (High) Reg (CURTESTSIZEH_INTREG)	[24:0]: Bit[56:32] of the current test data size in TestGen module
Other interfaces		
BA+0x800 Rd	exFAT-IP Version Reg (EXFATNVMVER_INTREG)	[31:0]: exFAT-IP version number, mapped to IPVersion [31:0] of exFAT-IP
BA+0x804 Rd	NVMe-IP Version Reg (NVMEVER_INTREG)	[31:0]: NVMe-IP or NVMeG3-IP version number, mapped to IPVersion [31:0] of NVMe-IP or NVMeG3-IP

3 CPU Firmware

3.1 Test firmware (exfatnvmetest.c)

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

1) CPU initializes its peripherals such as UART and Timer.
2) CPU waits until exFAT-IP completes initialization process (USRSTS_INTREG[0]='0').
3) CPU reads the disk information.
4) Receive the input from user to run format the disk or not.

   In case of start up without format,
   i) CPU loads the default parameter to set to exFAT-IP.
   ii) The current system information is displayed on the console. Three parameters which are outputs of exFAT-IP are read by CPU to display on the console, i.e., current file size (DFSIZE_INTREG), total file in the disk (DFNUM_INTREG), and maximum file to store in the disk (TOTALFCAP_INTREG).

   In case of start up with format, the next step is running Format command menu.

5) After all parameters are completely set, main menu is displayed on the console. There are four commands for the test, i.e., Format command (USRCMD_INTREG =”00”), Write file command (USRCMD_INTREG =”10”), Read file command (USRCMD_INTREG =”11”), and Shutdown command (USRCMD_INTREG =”01”).

More details of the operation sequence for each command are described as follows.

3.1.1 Format

The sequence of the firmware when user selects Format menu is below.

1) Ask user to set created date and created time of directory or use default value. Then, set the value to DATETIME_INTREG.
2) Read disk capacity from DCAPH/L_INTREG and calculate supported file size for displaying on the console.
3) Ask user to set file size and then set the value to USRFSIZE_INTREG.
4) Set USRCMD_INTREG =”00” to run Format command. After that, exFAT-IP changes to busy status (USRSTS_INTREG [0] changes from ‘0’ to ‘1’).
5) CPU waits until the operation is completed or some errors are found by monitoring USRSTS_INTREG[1:0].

   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, error message is displayed on the console.

6) After the command is completed, the disk information is displayed on the console, i.e., maximum amount of files in the disk (TOTALFCAP_INTREG), maximum amount of files per directory (DIRCAP_INTREG), current file size (DFSIZE_INTREG), and total amount of files in the disk (DFNUM_INTREG).

3.1.2 Write file/Read file command

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

1) Skip to the next step for Read file command. For Write file command, ask user to set created date and created time for the new created file or use the latest value. Then, set the value to DATETIME_INTREG.
2) In case of Write command,
   i) Ask user to use default Start file No. which continues from previous Write file command or change to other value.
   ii) When the input is valid, the new value is set to USRFNAME_INTREG.

   In case of Read command,
   i) Receive Start file No. from user.
   ii) When the input is valid, the new value is set to USRFNAME_INTREG.

3) Receive total files and test pattern through Serial console. When some inputs are invalid, the operation is cancelled.
4) Set the inputs to USRFLEN_INTREG and PATTSEL_INTREG.
5) Send Write file or Read file command by setting USRCMD_INTREG (“10” for Write file command or “11” for Read file command).
6) CPU waits until the operation is completed or some errors (except verification error) are found by monitoring USRSTS_INTREG[2:0].

   Bit[0] is de-asserted to ‘0’ when command is completed.
   Bit[1] is asserted when error is detected. After that, error message is displayed on the console to show the error details and the process is cancelled.
   Bit[2] is asserted when data verification is failed. Then, the verification error message is displayed. CPU is still running until the operation is done or user inputs any key to cancel operation.

   During running command, current transfer size reading from CURTESTSIZE_INTREG is displayed every second.

7) After busy flag (USRSTS_INTREG[0]) is de-asserted to ‘0’, CPU displays the test result on the console, i.e., total time usage, total transfer size, and transfer speed.

3.1.3 Shutdown Command

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

1) Set USRCMD_INTREG =”01” to run Shutdown command. After that, exFAT-IP changes to busy status (USRSTS_INTREG[0] changes from ‘0’ to ‘1’).
2) CPU waits until the operation is completed or some errors are found by monitoring USRSTS_INTREG[1:0].

   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, error message is displayed on the console.

3) After the command is completed, the device changes to inactive status and the CPU cannot receive new command from the user. The user must power off the test system after completing this command.

3.2 Function list in Test firmware

void change_ftime(void)	
Parameters	None
Return value	None
Description	Print current created time and date by calling show_ftime function. After that, ask user to change the value. If input is valid, the created time and date is updated to DATETIME_INTREG and global parameter (DateTime).

int format_fat(void)	
Parameters	None
Return value	0: User cancels command or command is finished. -1: Receive invalid input or error is found.
Description	Run Format command, following in topic 3.1.1

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

int get_param(userin_struct* userin, unsigned int user_cmd)	
Parameters	userin: Three inputs from user, i.e., start file number, a number of files, and test pattern user_cmd: 2-Write file command and 3-Read file command
Return value	0: Valid input, -1: Invalid input
Description	Read user_cmd and then calculate the input parameter range to display on the console. After receiving user input, the value is verified. If input is invalid, the function returns -1. Otherwise, all inputs are updated to userin parameter.

void show_dir(userin_struct* userin, unsigned int user_cmd)	
Parameters	User input, i.e., file name and number of file and User command
Return value	None
Description	Print file name and a directory of the 1st file and the last file which are currently written or read. Print file name and a directory of the last file in the device.

void show_diskinfo(void)	
Parameters	None
Return value	None
Description	Print the current disk information from global parameters, i.e., file size (DFnumB), maximum file in the disk (TotalFCap), maximum file per directory (DirCap), and total file in the disk (DFnum).

void show_error(void)	
Parameters	None
Return value	None
Description	Read USRERRTYPE_INTREG and decode the value. Print error type when the flag is found such as timeout error, NVMe(G3)-IP error, unsupported disk capacity, and unsupported LBA size.

void show_ftime(void)	
Parameters	None
Return value	None
Description	Print current created date and time from global parameter (DateTime)

void show_result(void)	
Parameters	None
Return value	None
Description	Print total size by calling get_cursize and show_size function. After that, calculate total time usage from global parameters (timer_val and timer_upper_val) and display in usec, msec, or sec unit. Finally, transfer performance is calculated and displayed on MB/s unit.

void show_size(unsigned long long size_input)	
Parameters	Size in byte unit
Return value	None
Description	Print input value in MB, GB, or TB unit

void show_testpin(void)	
Parameters	None
Return value	None
Description	Read TESTPINL/H_INTREG, NVMTESTPIN_INTREG to display exFAT-IP and NVMe(G3)-IP test pin on the console for debugging. For NVMeG3-IP, MACTESTPINL/H_INTREG are also read and displayed.

void show_vererr(void)	
Parameters	None
Return value	None
Description	Print information from hardware register to show verification error details, i.e., the 1st error file name (FAILFNAME_INTREG), the 1st error address (FAILADDRL/H_INTREG), expected value (EXPPATW0-3_INTREG), and error read value (RDPATW0-3_INTREG).

int shutdown_dev (void)	
Parameters	None
Return value	0: Shutdown command is finished. -1: User cancels command or error is found.
Description	Run Shutdown command, following in topic 3.1.3

void update_dfnum(void)	
Parameters	None
Return value	None
Description	Read total file in the disk from DFNUM_INTREG, and then update read value to global parameter (DFnum).

void update_dfsize(void)	
Parameters	None
Return value	None
Description	Read total file (TOTALFCAP_INTREG) and current file size in the disk (DFSIZE_INTREG) from hardware. File size is decoded and converted to be byte unit. Finally, total file and file size are updated to global parameters (TotalFCap and DFsizeB).

int wrrd_file(unsigned int user_cmd)	
Parameters	Command from user (2: Write file command, 3: Read file command)
Return value	0: Operation is successful. -1: Receive invalid input or error is found.
Description	Run Write file or Read file command, following in topic 3.1.2

4 Example Test Result

The example test result when running demo system by using 512 GB Samsung 970 Pro is shown in Figure 4-1.

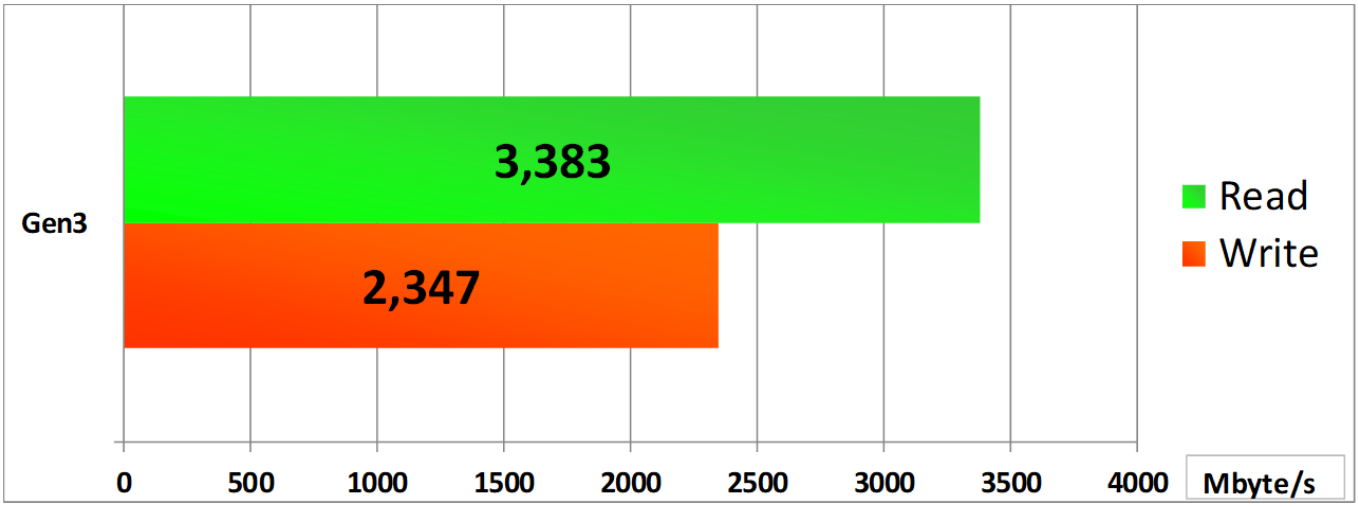

Figure 4-1 Test Performance of exFAT-IP demo for NVMe by using Samsung 970 Pro SSD

By using PCIe Gen3 on KCU105 board, write performance is about 2300 Mbyte/sec and read performance is about 3300 Mbyte/sec.

5 Revision History

Revision	Date	Description
1.0	15-Jan-19	Initial release
1.1	22-Mar-19	- Add DiskFsize, DiskFnum signal - Add function list
1.2	14-May-19	Update Dircap size and the design details
1.3	19-Feb-20	Update TestGen for all zero and all one pattern
1.4	23-Jul-20	Update CPU firmware
1.5	14-Mar-22	Add NVMeG3 IP to reference design