exFAT IP for NVMe reference design manual

Rev1.0 22-Jan-19

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 through physical address. When many data types are stored in one disk, user needs to assign different address for each data group. Without standard, each system defines different data structure to arrange data groups. It is the problem for the Host to read data from different system which uses different data structure.

As a result, many file system standards are created to manage data in the disk by setting up the table to be an index for data written to the disk. 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 one standard file system which is common in many platforms. Comparing to FAT32 file system, exFAT file system is designed to improve many features. exFAT supports more than 4 GB file size and supports more than 2 TB device capacity. Name hash of file name is implemented to improve search function. Also, checksum is applied in system data area to increase data reliability.

Generally, file system is implemented as standard library running on CPU. To write/read file by using CPU software, it has overhead time to access file header to read data 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 and reduces overhead time to access file header during write/read file. As a result, write/read performance when using exFAT IP is almost same as raw data format which is run by DG NVMe IP. From the reference design, the performance of Write file command is about 2144 MB/s while the performance of Read file command is about 3251 MB/s.

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

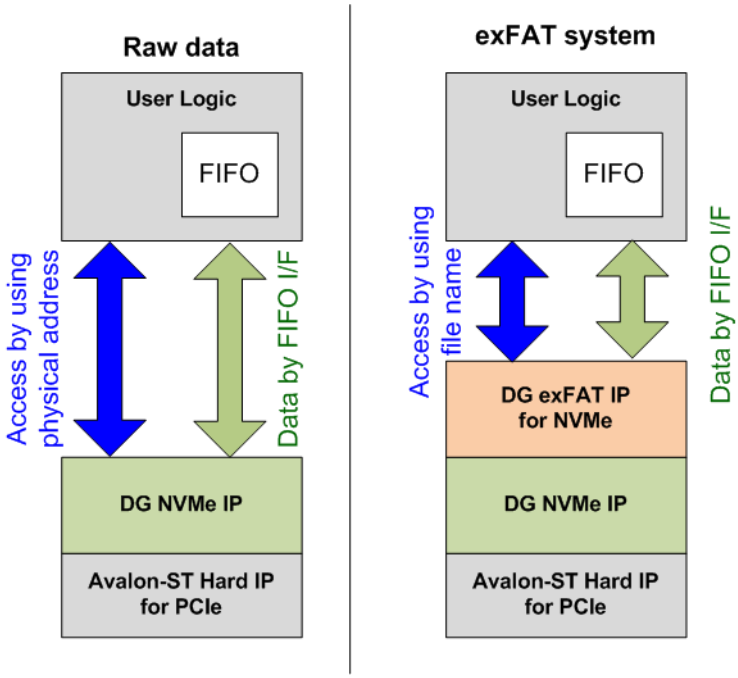

Figure 1-1 Hardware system comparison between raw data and file system

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 DG NVMe IP. The parameter of user interface changes from physical parameters (address and length) to be file parameters (file name and total 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 DG NVMe IP reference design by including exFAT IP and updating control path to be file parameters instead of physical parameters, as shown in blue color of Figure 2-1.

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

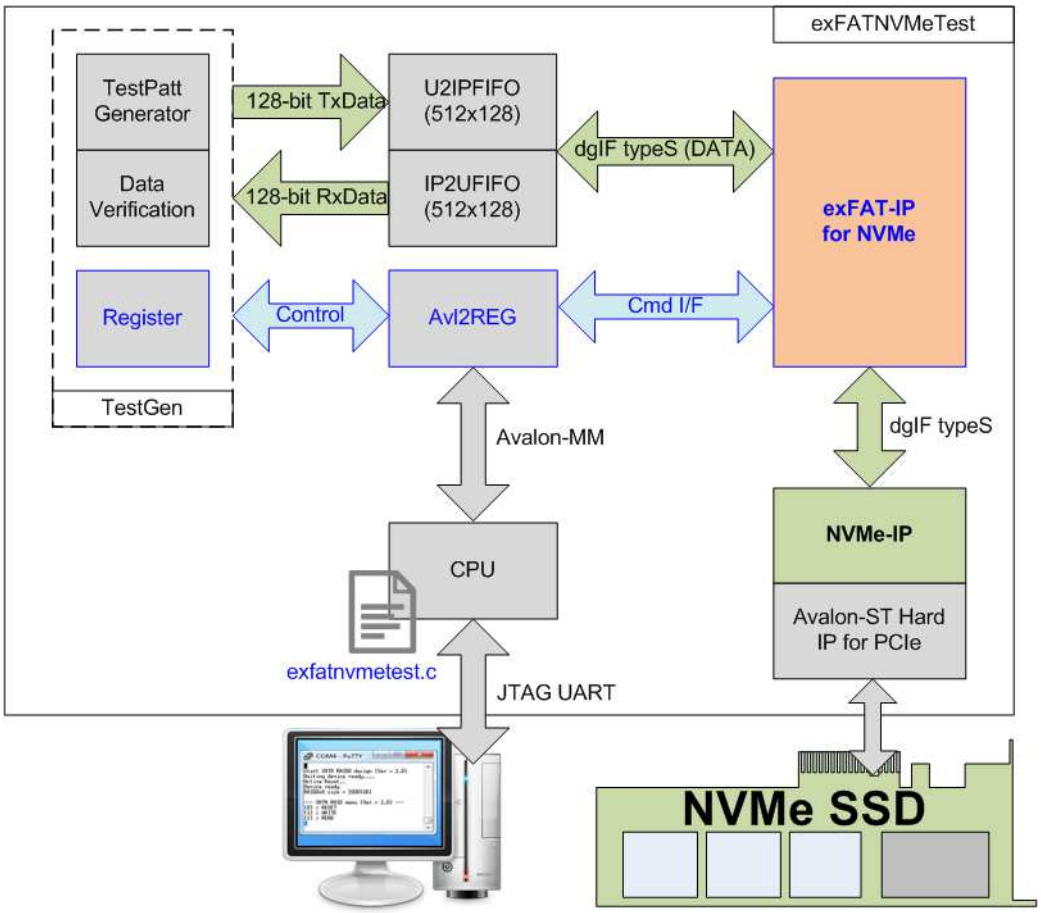

Figure 2-1 exFAT IP for NVMe demo system

File parameters input to exFAT IP through control interface are received from TestGen module. The register inside TestGen module is set by CPU through Avalon-MM bus. CPU firmware monitors JTAG UART to get file parameters from user. The examples of file parameter are file name, file length, file size, total file, created date, and created time.

Otherwise, user selects the command to run on exFAT IP, i.e. Format, Write file, Read file, and Shutdown. As a test result, transfer performance is displayed on NiosII command shell after complete to write file or read file operation. File in NVMe SSD could be read by other Hosts which support exFAT system such as PC.

Similar to NVMe IP, data path in the design are controlled by test pattern generator and data verification module. More details of the hardware in exFAT IP for NVMe demo design are described as follows.

2.1 TestGen

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

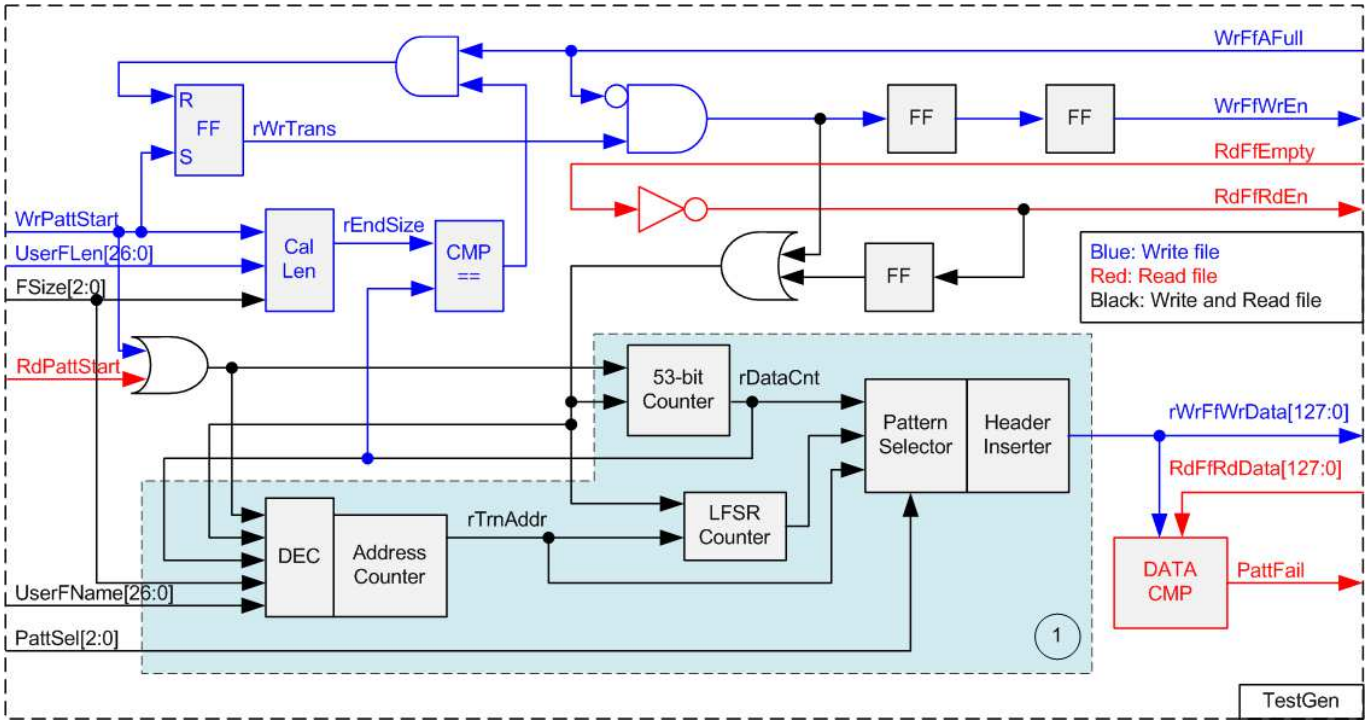

Figure 2-2 TestGen hardware

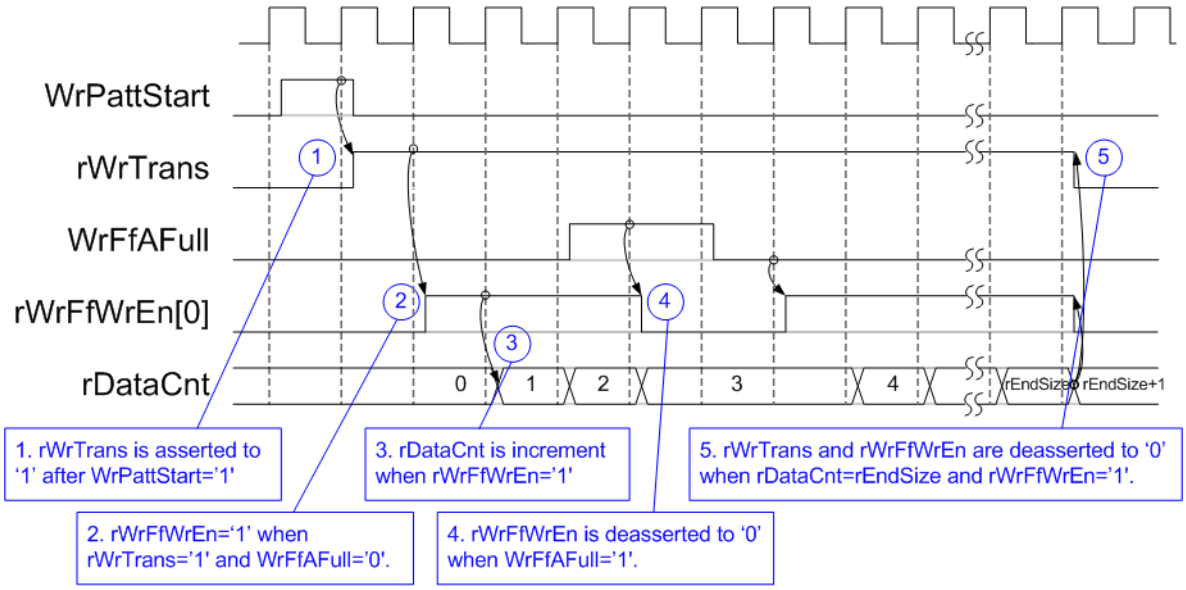

Figure 2-3 Timing diagram of Write operation in TestGen

As shown in Figure 2-3, WrPattStart is asserted to '1' when Write file command is requested. After that, rWrTrans is asserted to '1' until end of operation. During Write file command operation, rWrFfWrEn[0] is controlled by WrFfAFull signal. rWrFfWrEn is asserted to '1' with the valid data on rWrFfWrData to generate test data when WrFfAFull='0'. Otherwise, rWrFfWrEn[0] is de-asserted to '0' to pause data transferring.

rDataCnt is increased by rWrFfWrEn[0] to check total transfer size. After total data are transferred completely (rDataCnt=rEndSize), rWrTrans and rWrFfWrEn[0] are de-asserted to '0'. Total transfer size in 512-byte unit is calculated by UserFLen x Fsize (File size).

Refer to Figure 2-2, RdFfRdEn signal is simple designed by connecting NOT logic to RdFfEmpty. Similar to Write file operation, rDataCnt is increased by RdFfRdEn to count total transfer size. rDataCnt is also used to generate test pattern for verifying the received data (RdFfRdData).

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 every 512-byte data, test pattern is designed as shown in Figure 2-4.

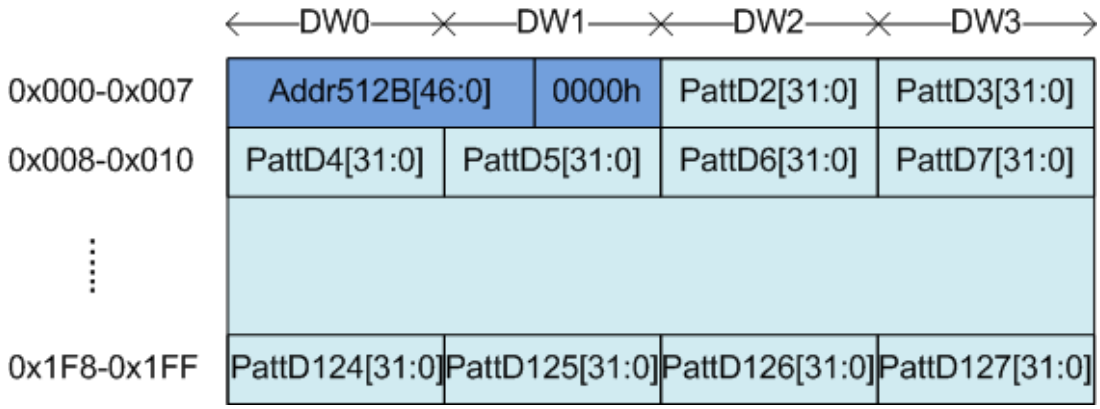

Figure 2-4 Test pattern format in every 512-byte data

Every 512-byte test pattern data consists of two parts, i.e. 64-bit header in Dword#0 - #1 and test data in Dword#2 – #127. 64-bit header is created by using address value in 512-byte unit (Addr512B or rTrnAddr in Figure 2-2). The initial value of Addr512B is calculated by UserFName x File size (512-byte unit). After complete to generate 512-byte data, rTrnAddr is increased by one. So, the header of the next 512-byte is not same as current 512-byte data.

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 data is generated by using rTrnAddr and rDataCnt. Decrement pattern is designed by using NOT logic wired to increment data. To create 32-bit LFSR counter, the 1st DW (Addr512B[31:0]) is used to be initial value for generating test pattern in each 512 byte. The equation of LFSR is x^31 + x^21 + x + 1.

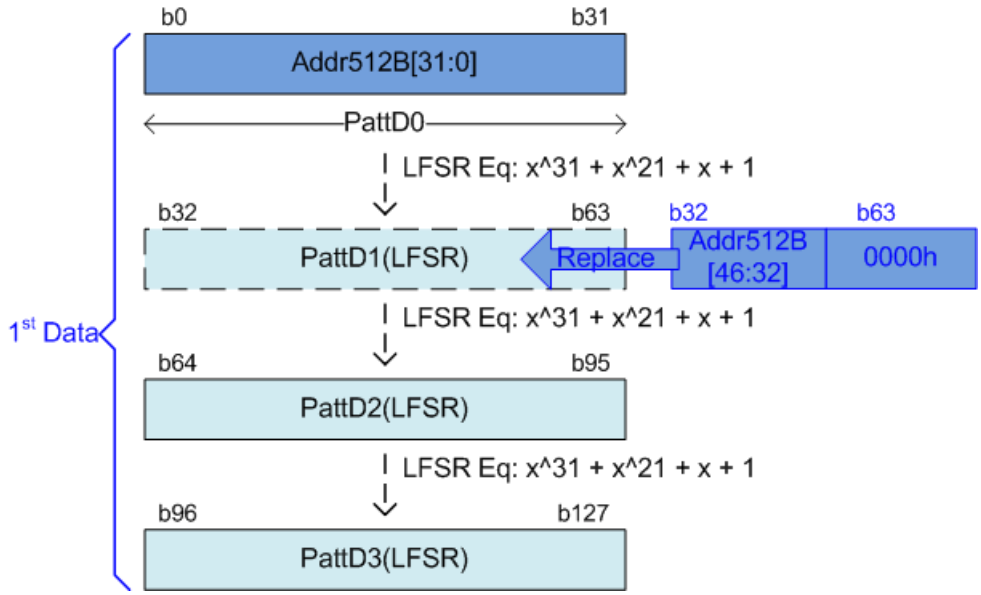

Figure 2-5 LFSR pattern in TestGen

As shown in Figure 2-5, 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 PattD0 – PattD3 (128-byte data) in the same clock.

As shown in Figure 2-2, 3-bit PattSel is used to select one of five test patterns. Header Inserter logic inserts 64-bit header to be the 1st and 2nd data of each 512-byte data. After that, test data from pattern counter is transferred to be rWrFfWrData. In Read file 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 exFAT

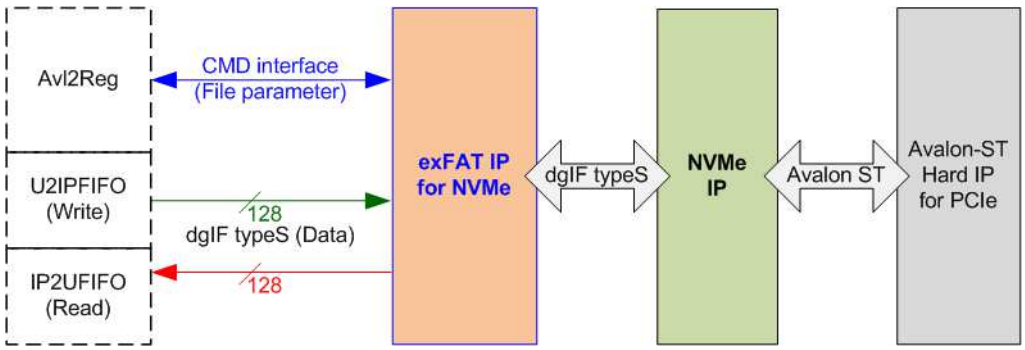

Figure 2-6 exFAT hardware

As shown in Figure 2-6, user interface of exFAT IP is split into two groups, i.e CMD interface and Data interface. CMD interface is connected to Avl2Reg to receive file parameters from user through JTAG UART. Data interface is 128-bit bus size and connects to U2IPFIFO and IP2UFIFO for different transfer direction. Another side of exFAT IP is connected to NVMe IP.

2.2.1 exFAT IP for NVMe

exFAT IP implements the logic to handle data in NVMe SSD following exFAT file system. exFAT IP must be integrated with NVMe IP. 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_intel_en.pdf

2.2.2 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 Avalon-ST Hard IP for PCIe (Hard IP in Intel FPGA device). More details of NVMe IP are described in datasheet.
https://dgway.com/products/IP/NVMe-IP/dg_nvmeip_datasheet_intel_en.pdf

2.2.3 Avalon-ST PCIe Hard IP

This block is hard IP in Intel FPGA device which implements Physical, Data Link, and Transaction Layers of PCIe specification. More details are described in Intel FPGA document.

ArriaV Avalon-ST Interface for PCIe Solutions User Guide
https://www.altera.com/en_US/pdfs/literature/ug/ug_a5_pcie_avst.pdf
Stratix V Avalon-ST Interface for PCIe Solutions User Guide
https://www.altera.com/en_US/pdfs/literature/ug/ug_s5_pcie_avst.pdf
Intel Arria10 Avalon-ST Interface for PCIe Solutions User Guide
https://www.altera.com/en_US/pdfs/literature/ug/ug_a10_pcie_avst.pdf

2.3 CPU and Peripherals

The hardware is connected to CPU through Avalon-MM bus, similar to other CPU peripherals. The hardware registers are mapped to CPU memory address, as shown in Table 2-1. The control and status registers for CPU access are designed in Avl2Reg.

Avl2Reg connects to many hardwares in the system such as TestGen, exFAT IP to generate control signals and monitor status signals by CPU. As shown in Figure 2-7, there are two clock domains applied in this block, i.e. CpuClk which is clock domain of Avalon-MM bus interface with CPU and UserClk which is clock domain for TestGen and NVMe IP.

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

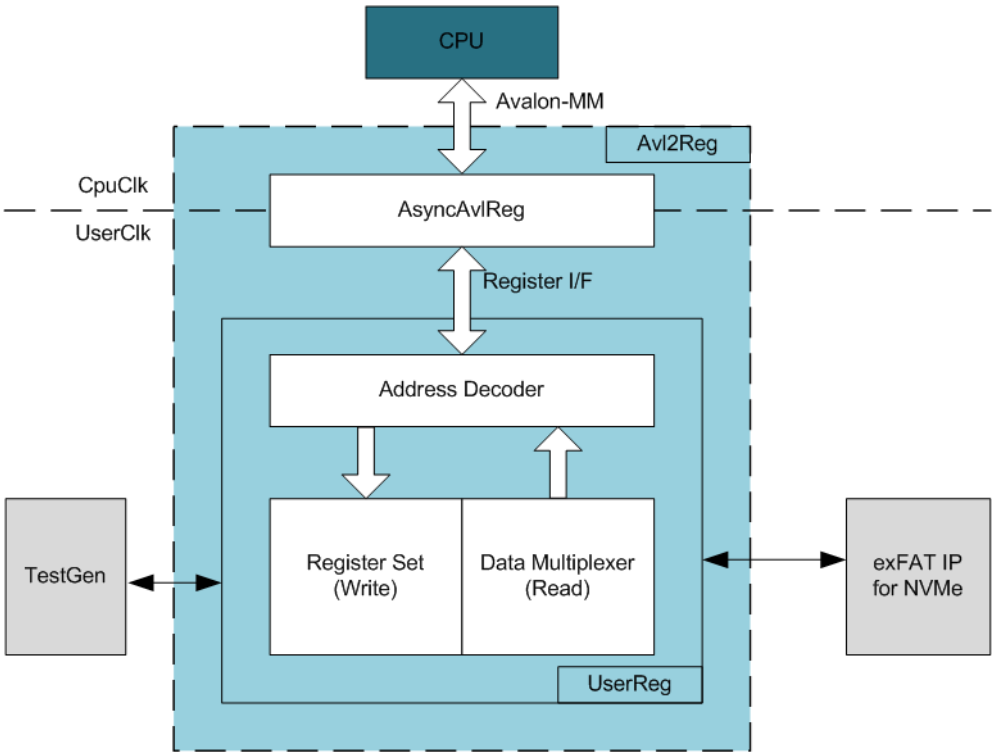

Figure 2-7 CPU and peripherals hardware

2.3.1 AsyncAvlReg

This module is designed to convert the signal interface of Avalon-MM 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 2-8.

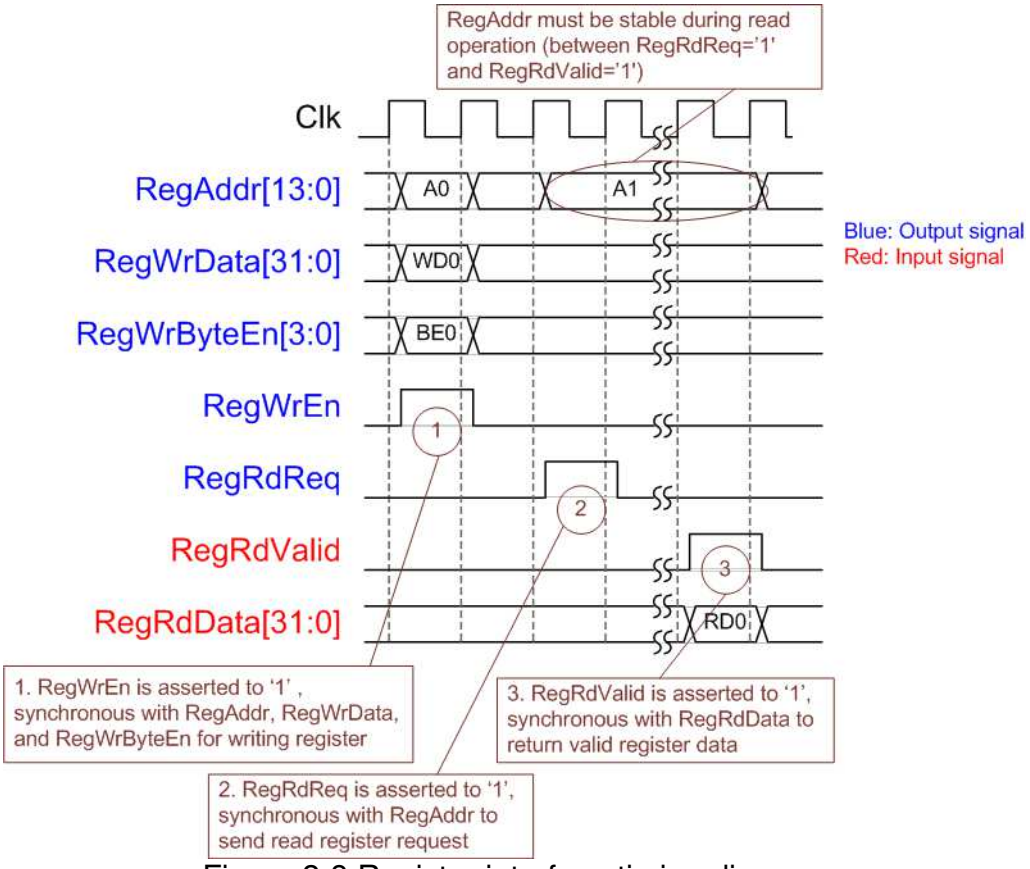

Figure 2-8 Register interface timing diagram

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, AsyncAvlReg asserts RegRdReq='1' with the valid value of RegAddr. After that, the module waits until RegRdValid is asserted to '1' to get the read data through RegRdData signal. During read access, RegAddr holds the same value until RegRdValid is asserted to '1'.

2.3.2 UserReg

The details of UserReg module are displayed in Figure 2-7. After RegWrEn or RegRdReq is asserted to '1' by AsyncAvlReg to request write or read register access, RegAddr is loaded to Address decoder to select the active register. For write access, RegWrData signal is loaded to replace value to the requested 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, there are many status signals for CPU access such as TestGen, exFAT IP. So, data multiplexer with pipeline register is designed to select the read data to return to CPU following RegAddr. RegRdValid is designed by using two D Flip-flops, input by RegRdReq signal. So, the read access has two-clock cycle latency (the delay time from RegRdReq to RegRdValid).

Memory map of control and status signals inside UserReg module is shown in Table 2-1.

Table 2-1 Register Map

Address Rd/Wr	Register Name (Label in the “exfatnvmetest.c”)	Description
BA+0x0000 Wr	User File Name Reg (USERFNAME_REG)	[26:0]: Input to be UserFName of exFAT IP for NVMe
BA+0x0004 Wr	User File Length Reg (USERFLEN_REG)	[26:0]: Input to be UserFLen of exFAT IP for NVMe
BA+0x0008 Wr	File Size Reg (FSIZE_REG)	[2:0]: Input to be FSize of exFAT IP for NVMe
BA+0x000C Wr	File Time Reg (DATETIME_REG)	[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+0x0010 Wr	User Command Reg (USERCMD_REG)	[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 new command operation.
BA+0x0014 Wr	Pattern Select Reg (PATTSEL_REG)	[2:0]: Test pattern select “000”-Increment, “001”-Decrement, “010”-All 0, “011”-All 1, “100”-LFSR
BA+0x0100 Rd	User Status Reg (USRSTS_REG)	[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+0x0104 Rd	Total file capacity Reg (TOTALFCAP_REG)	[26:0]: Mapped to TotalFCap[26:0] of exFAT IP for NVMe
BA+0x0108 Rd	User Error Type Reg (USRERRTYPE_REG)	[31:0]: Mapped to UserErrorType[31:0] of exFAT IP for NVMe
BA+0x010C Rd	exFAT IP Test pin (Low) Reg (FATTESTPINL_REG)	[31:0]: Mapped to TestPin[31:0] of exFAT IP for NVMe
BA+0x0110 Rd	exFAT IP Test pin (High) Reg (FATTESTPINH_REG)	[31:0]: Mapped to TestPin[63:32] of exFAT IP for NVMe
BA+0x0114 Rd	Directory capacity Reg (DIRCAP_REG)	[21:0] Mapped to DirCap[21:0] of exFAT IP for NVMe
BA+0x0118 Rd	Completion Status Reg (COMPSTS_REG)	[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)
BA+0x011C Rd	NVMe CAP Reg (NVMCAP_REG)	[31:0]: Mapped to NVMeCAPReg[31:0], output from NVMe IP
BA+0x0120 Rd	NVMe Test pin (Low) Reg (NVMTESTPIN_REG)	[31:0]: Mapped to TestPin[31:0], output from NVMe IP
BA+0x0200 Rd	Expected value Word0 Reg (EXPPATW0_REG)	[31:0]: Expected data [31:0] of failure address
BA+0x0204 Rd	Expected value Word1 Reg (EXPPATW1_REG)	[31:0]: Expected data [63:32] of failure address
BA+0x0208 Rd	Expected value Word2 Reg (EXPPATW2_REG)	[31:0]: Expected data [95:64] of failure address
BA+0x020C Rd	Expected value Word3 Reg (EXPPATW3_REG)	[31:0]: Expected data [127:96] of failure address

Address Rd/Wr	Register Name (Label in the “exfatnvmetest.c”)	Description
BA+0x0210 Rd	Read value Word0 Reg (RDPATW0_REG)	[31:0]: Read data [31:0] of failure address
BA+0x0214 Rd	Read value Word1 Reg (RDPATW1_REG)	[31:0]: Read data [63:32] of failure address
BA+0x0218 Rd	Read value Word2 Reg (RDPATW2_REG)	[31:0]: Read data [95:64] of failure address
BA+0x021C Rd	Read value Word3 Reg (RDPATW3_REG)	[31:0]: Read data [127:96] of failure address
BA+0x0220 Rd	Failure Byte Address (Low) Reg (FAILADDRL_REG)	[31:0]: Failure byte address in the file (bit[31:0])
BA+0x0224 Rd	Failure Byte Address (High) Reg (FAILADDRH_REG)	[6:0]: Failure byte address in the file (bit[38:32])
BA+0x0228 Rd	Failure File Name Reg (FAILFNAME_REG)	[26:0]: Failure file name
BA+0x022C Rd	Current test byte (Low) Reg (CURTESTSIZEL_REG)	[31:0]: Current test data size of TestGen module in byte unit (bit[31:0])
BA+0x0230 Rd	Current test byte (High) Reg (CURTESTSIZEH_REG)	[24:0]: Current test data size of TestGen module in byte unit (bit[56:32])
BA+0x0800 Rd	IP Version Reg (IPVERSION_REG)	[31:0]: IP version number, mapped to IPVersion [31:0] of exFAT IP for NVMe

3 CPU Firmware

After system boot-up, CPU initializes its peripherals such as JTAG UART and Timer. Next, CPU waits until exFAT IP completes initialization process (USRSTS_REG[0]='0'). After that, maximum file to store in the disk (reading from TOTALFCAP_REG) is displayed on NiosII command shell. This value is calculated by using default file size (FSIZE_REG="000" or 32 MB). File size could be changed by using "Change file size" menu. If file size is changed, maximum file to store in the disk will be updated.

CPU firmware supports five menus. Four menus are designed to test four commands of USRCMD_REG, i.e. "00" for Format command, "10" for Write file command, "11" for Read file command, and "01" for Shutdown command. The fifth menu is designed to change file size input to exFAT IP. More details of the sequence in each menu are described as follows.

3.1 Format

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

1) Set created date and created time of directory to DATETIME_REG or skip this setting by using default value.
2) Set USERCMD_REG="00" to format the disk. exFAT IP busy flag (USRSTS_REG[0]) changes from '0' to '1' after operating Format command.
3) 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 NiosII command shell. If the command is completed, maximum file in the disk (TOTALFCAP_REG) will be displayed on NiosII command shell.

3.2 Write file/Read file command

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

1) Receive file name, total files, and test pattern through JTAG UART. If some inputs are invalid, the operation will be cancelled. In case of Write file operation, user could set created date and created time of file by setting DATETIME_REG or use current value.
   Note: File name input of Write file operation must continue from the latest write file. CPU firmware is designed to check that the value of Filename is continuous when running Write file command.
2) Get all inputs and set the value to USERFNAME_REG, USERFLEN_REG, PATTSEL_REG, and USRCMD_REG (USRCMD_REG="10" for Write file command and "11" for Read file 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] 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 NiosII command shell when command is completed.

3.3 Shutdown Command

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

1) Set USRCMD_REG=”01” to send Shutdown SSD. exFAT IP busy flag (USRSTS_REG[0]) changes from ‘0’ to ‘1’ to operate Shutdown command.
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.
3) If the command is completed, SSD will change to inactive status. Also, CPU does not receive new command from user. User should power off system after completing this command.

3.4 Change file size

The sequence of the firmware when user selects Change file size is below.

1) Receive new file size value from user through JTAG UART. If the input is valid and UserBusy=’0’, the new value will be set to FSIZE_REG.
2) CPU displays maximum file in the disk by reading TOTALFCAP_REG. Warning message is displayed to recommend user to format the disk after changing file size.

4 Example Test Result

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

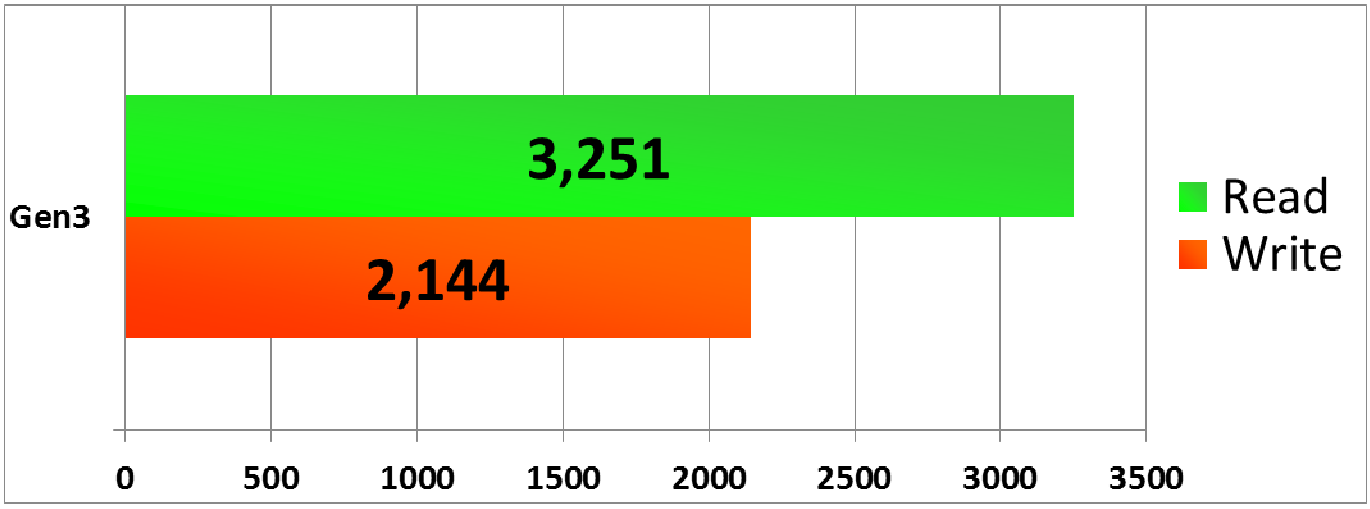

Figure 4-1 Test Performance of exFAT IP demo for NVMe by using Samsung 960 Pro SSD

By using PCIe Gen3 on Arria10 GX development kit, write performance is about 2144 Mbyte/sec and read performance is about 3251 Mbyte/sec.

5 Revision History

Revision	Date	Description
1.0	22-Jan-19	Initial release