exFAT IP for SATA reference design manual

Rev1.0 19-Nov-18

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 disk from different system which is designed different data structure.

As a result, file system is 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 standard file system which is common support 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. So, write/read performance when running file system by using CPU software is reduced, comparing to using raw data format.

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. So, write/read performance when using exFAT IP is almost same as raw data format which is run by SATA HCTL IP only. From the demo design, the performance of Write file command is about 520 MB/s while the performance of Read file command is about 560 MB/s.

The hardware design of exFAT IP for SATA demo is different from the hardware design of SATA HCTL IP design, as shown in Figure 1-1.

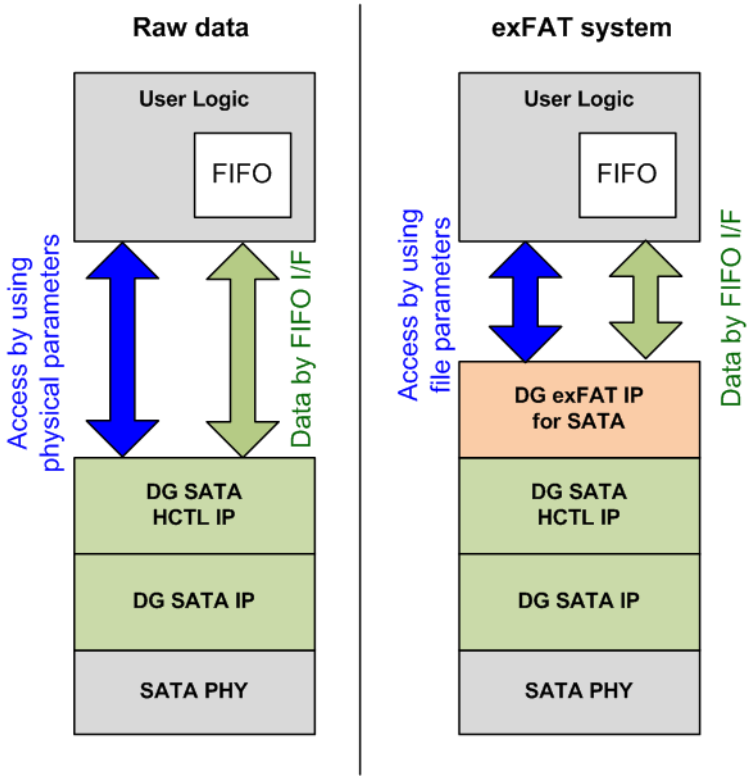

Figure 1-1 Hardware system for raw data and file system

Comparing to raw data system in the left side of Figure 1-1, exFAT system includes exFAT IP for SATA to connect between User Logic and SATA HCTL 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 SATA reference design are described in the next topic.

2 Hardware overview

The reference design of DG exFAT IP for SATA is modified from SATA HCTL IP reference design by including DG exFAT IP and updating control path to be file parameters instead of physical parameters, as shown in blue color of Figure 2-1.

Please see more details of SATA HCTL IP reference design from following document.
https://dgway.com/products/IP/SATA-IP/dg_satactlip_refdesign_en.pdf
https://dgway.com/products/IP/SATA-IP/dg_satahctltip_instruction_en.pdf

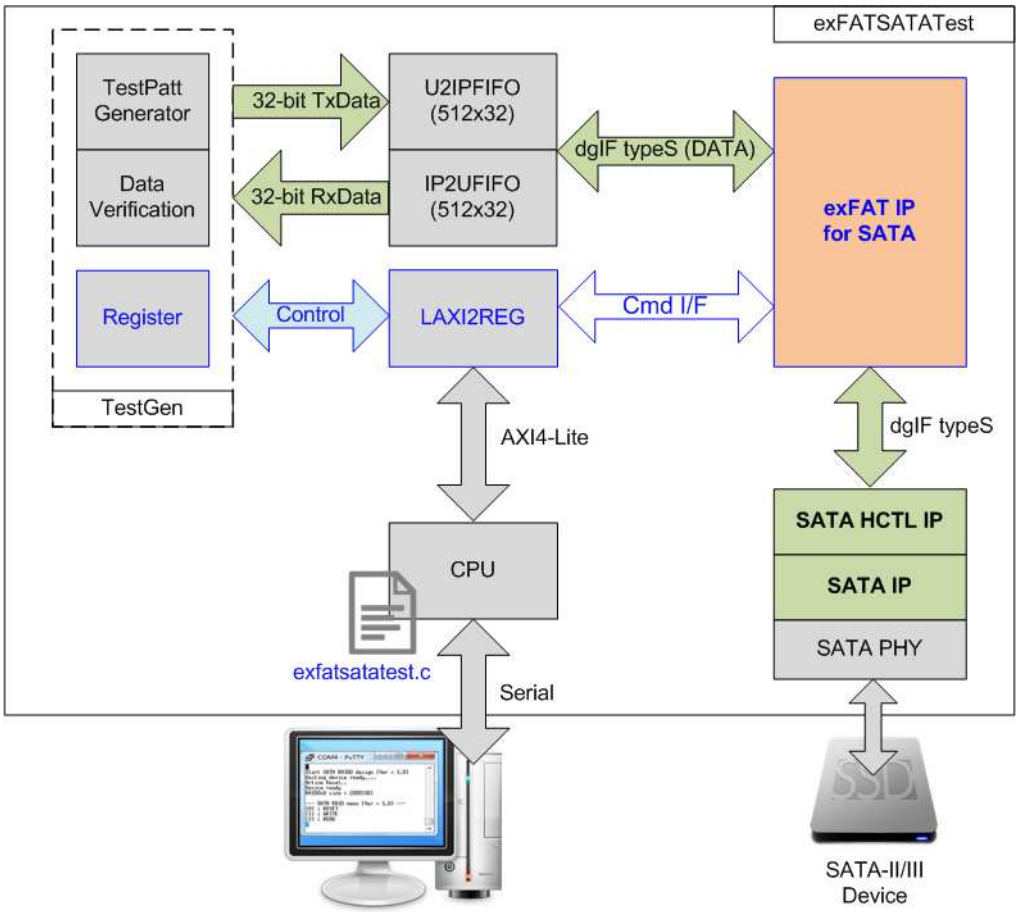

Figure 2-1 exFAT IP for SATA demo system

File parameters to assign to control interface of exFAT IP are received from TestGen module. The register inside TestGen module is programmed by CPU through AXI4-Lite bus. User inputs file parameters through Serial port such as file name, file length, file size, total file, created data, and created time.

Otherwise, user selects the command to run on exFAT IP, i.e. Format, Write file, and Read file. As a test result, transfer performance is displayed on Serial console after complete to write file or read file operation. File data in SATA device could be plugged to other Hosts which support exFAT system such as PC to read test file data.

Similar to HCTL IP, data path in the design is designed by using test generator and verification module. More details of the hardware in exFAT IP for SATA 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.

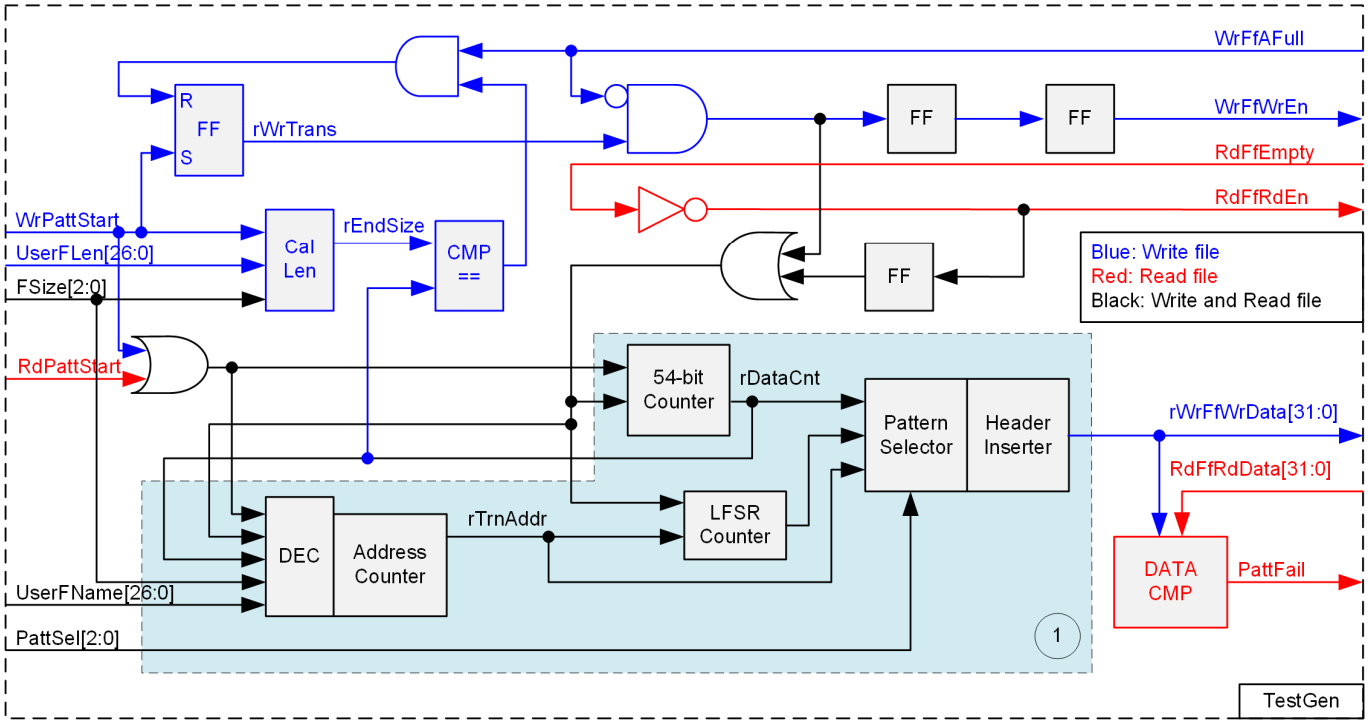

Figure 2-2 TestGen hardware

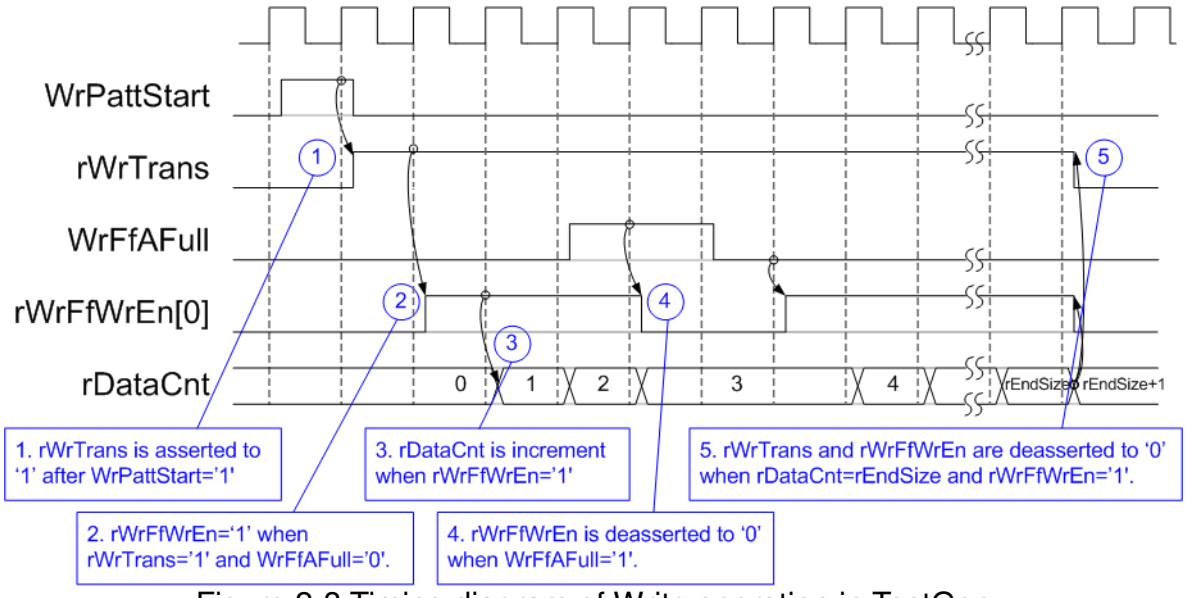

Figure 2-3 Timing diagram of Write operation in TestGen

WrPattStart is asserted to '1' when Write file command is request. 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 sector unit is calculated by UserFLen x File size (File size is decoded from FSize signal).

For Read file operation, RdFfRdEn signal is simple designed by connecting NOT logic to RdFfEmpty. Similar to Write file operation, rDataCnt is increased by RdFfRdEn to check 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 each sector, test pattern is designed as shown in Figure 2-4.

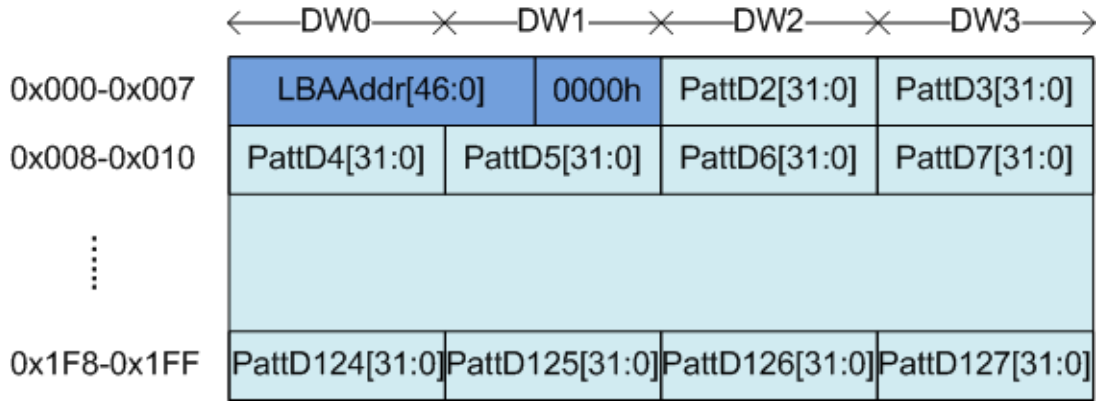

Figure 2-4 Test pattern format in each sector

Test pattern consists of two parts, i.e. 64-bit header in Dword#0 - #1 of each sector and test data in Dword#2 – #127. 64-bit header is created by using LBA address value of the data (LBA address is physical address in sector unit). As shown in Figure 2-2, UserFName and FSize are calculated to be initial value of rTrnAddr (Initial value = UserFName x File size). rTrnAddr is applied to be the header at DW0 of each sector and increased after completing one sector data 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 data is generated by using rTrnAddr and rDataCnt. Decrement pattern is designed by using NOT logic to increment data.

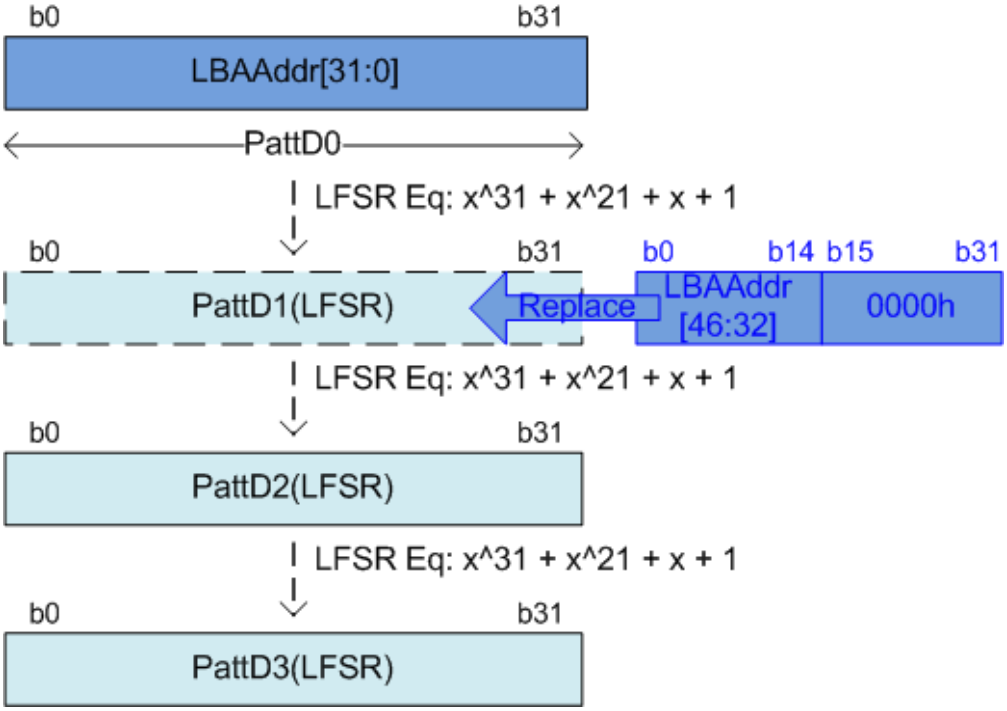

Figure 2-5 LFSR pattern in TestGen

Start value of LFSR pattern for each 512-byte data is same as other patterns by loading 32-bit header to be initial value. PattD1 is replaced by remaining 32-bit header data. After that, LFSR pattern is generated by using equation = x^31 + x^21 + x + 1.

3-bit PattSel is used to select one of five test patterns. Header Inserter 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

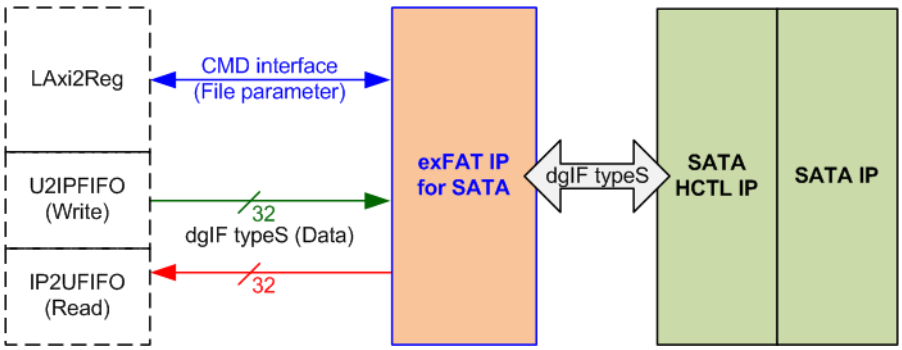

Figure 2-6 exFAT hardware

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 LAxi2Reg to receive file parameters from user through Serial console. Data bus size is 32-bit and connects with U2IPFIFO and IP2UFIFO. Another side of exFAT IP is connected to DG SATA HCTL IP.

2.2.1 exFAT IP for SATA

exFAT IP implements the logic to handle data in SATA device following exFAT file system. exFAT IP must be integrated with DG SATA HCTL IP. Data bus size is 32-bit. More details of exFAT IP for SATA are described in datasheet.
https://dgway.com/products/IP/SATA-IP/dg_exfatip_sata_data_sheet_en.pdf

2.2.2 SATA HCTL IP

DG SATA HCTL IP implements application layer of SATA protocol to create/decode SATA FIS interface. It must be integrated with DG SATA IP to access SATA Device. More details of DG SATA HCTL IP are described in datasheet.
https://dgway.com/products/IP/SATA-IP/dg_sata_hctl_ip_data_sheet_en.pdf

2.2.3 SATA IP

SATA IP implements some parts of transport layer and link layer of SATA protocol. It must be integrated with SATA PHY which includes Xilinx Transceiver. More details of SATA-IP are described in datasheet.
https://www.dgway.com/products/IP/SATA-IP/dg_sata_ip_data_sheet_7series_en.pdf

2.3 CPU and Peripherals

The hardware is connected to CPU through AXI4-Lite 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 LAxi2Reg.

LAxi2Reg connects to many hardwares in the system such as TestGen, exFAT IP. The control and status signals of each module could be accessed by CPU through LAxi2Reg. As shown in Figure 2-7, there are two clock domains applied in this block, i.e. CpuClk which is clock domain of AXI4-Lite bus of CPU and UserClk which is clock domain for TestGen and SATA HCTL IP.

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

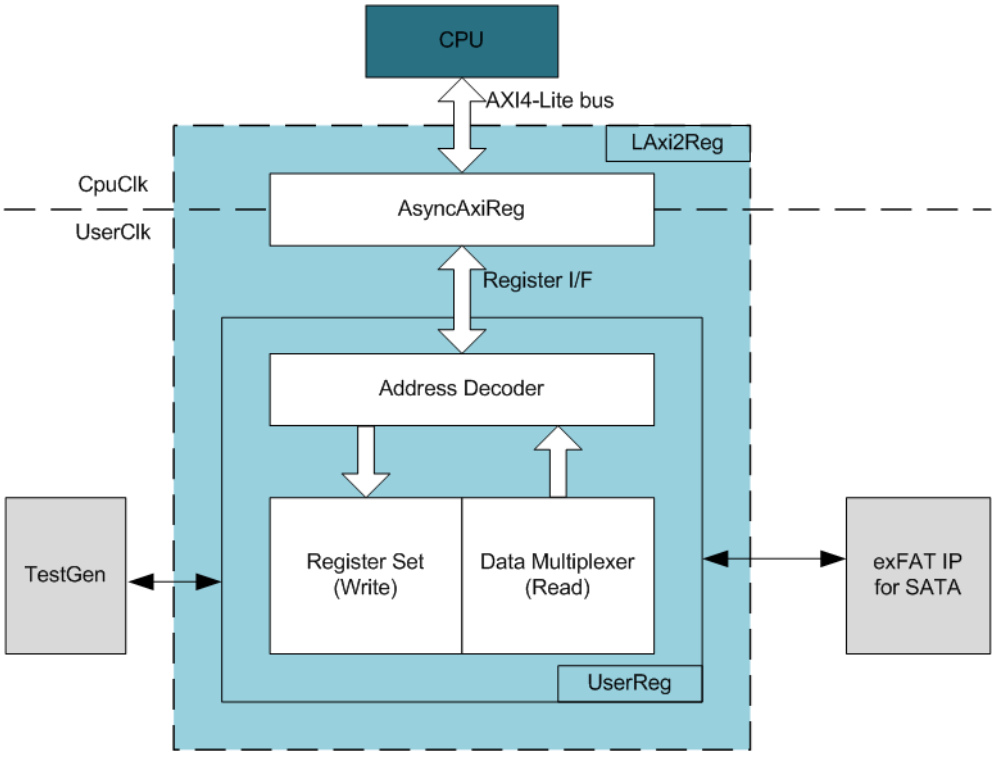

Figure 2-7 CPU and peripherals hardware

2.3.1 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 2-8.

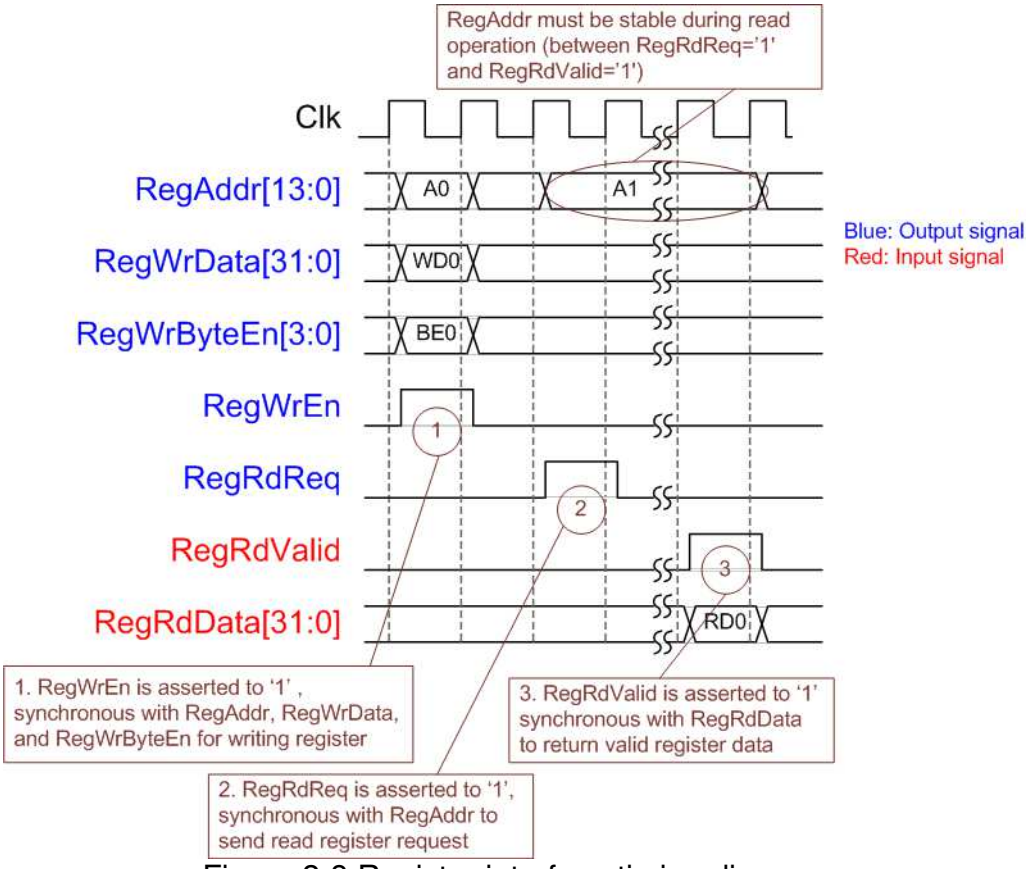

Figure 2-8 Register interface timing diagram

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, AsyncAxiReg asserts RegRdReq='1' with the valid value of RegAddr (the register address in 32-bit unit). 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 is shown in Figure 2-7. After RegWrEn or RegRdReq is asserted to '1' by AsyncAxiReg to request write or read register access, RegAddr is read by Address decoder to select the active register. For write access, RegWrData signal is loaded to be the new value for 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 cycles latency, measured by 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 “exfatsatatest.c”)	Description
BA+0x000 Wr	User File Name Reg (USERFNAME_REG)	[26:0]: Input to be UserFName of exFAT IP for SATA
BA+0x004 Wr	User File Length Reg (USERFLEN_REG)	[26:0]: Input to be UserFLen of exFAT IP for SATA
BA+0x008 Wr	File Size Reg (FSIZE_REG)	[2:0]: Input to be FSize of exFAT IP for SATA
BA+0x00C Wr	File Time Reg (DATETIME_REG)	[4:0]: Input to be FTimeS of exFAT IP for SATA [10:5]: Input to be FTimeM of exFAT IP for SATA [15:11]: Input to be FTimeH of exFAT IP for SATA [20:16]: Input to be FDateD of exFAT IP for SATA [24:21]: Input to be FDateM of exFAT IP for SATA [31:25]: Input to be FDateY of exFAT IP for SATA
BA+0x010 Wr	User Command Reg (USERCMD_REG)	[1:0]: Input to be UserCmd of exFAT IP for SATA When this register is written, the design asserts UserReq=’1’ (command request) to exFAT IP for SATA to start new command operation.
BA+0x014 Wr	Pattern Select 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]: Mapped to UserBusy of exFAT IP for SATA [1]: Mapped to UserError of exFAT IP for SATA [2]: Data verification fail (‘0’: Normal, ‘1’: Error)
BA+0x104 Rd	Total file capacity Reg (TOTALFCAP_REG)	[26:0]: Mapped to TotalFCap of exFAT IP for SATA
BA+0x108 Rd	User Error Type Reg (USRERRTYPE_REG)	[31:0]: Mapped to UserErrorType of exFAT IP for SATA
BA+0x10C Rd	exFAT IP Test pin (Low) Reg (TESTPINL_REG)	[31:0]: Mapped to TestPin[31:0] of exFAT IP for SATA
BA+0x110 Rd	exFAT IP Test pin (High) Reg (TESTPINH_REG)	[31:0]: Mapped to TestPin[63:32] of exFAT IP for SATA
BA+0x114 Rd	Directory capacity Reg (DIRCAP_REG)	[21:0] Mapped to DirCap[21:0] of exFAT IP for SATA
BA+0x200 Rd	Expected value Reg (EXPPATW_REG)	[31:0]: Expected data [31:0] of failure address.
BA+0x204 Rd	Read value Reg (RDPATW_REG)	[31:0]: Read data [31:0] of failure address.
BA+0x208 Rd	Failure Byte Address Reg (FAILADDRL_REG)	[31:0]: Failure data address in the file in byte unit (bit[31:0])
BA+0x20C Rd	Failure Byte Address Reg (FAILADDRH_REG)	[6:0]: Failure data address in the file in byte unit (bit[38:32])
BA+0x210 Rd	Failure File Name Reg (FAILFNAME_REG)	[26:0]: Failure file name
BA+0x214 Rd	Current test byte (Low) Reg (CURTESTSIZEL_REG)	[31:0]: Current test data size of TestGen module in byte unit (bit[31:0])
BA+0x218 Rd	Current test byte (High) Reg (CURTESTSIZEH_REG)	[23:0]: Current test data size of TestGen module in byte unit (bit[55:32])
BA+0x800 Rd	IP Version Reg (IPVERSION_REG)	[31:0]: IP version number, mapped to IPVersion [31:0] of exFAT IP for SATA

3 CPU Firmware

After system boot-up, CPU initializes its peripherals such as 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 the console. 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 will be updated.

CPU firmware supports four menus. Menu 1-3 is designed to run three commands following USRCMD_REG value, i.e. "00" for Format command, "10" for Write file command, and "11" for Read file command. Menu 4 is designed to change file size in SATA device. More details of the sequence in each operation 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 the console. If the command is completed, maximum file in the disk (TOTALFCAP_REG) will be displayed on the console.

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 Serial console. If some inputs are invalid, the operation will be cancelled. In case of Write file operation, user selects to 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.
2) Get all inputs and set the value to USERFNAME_REG, USERFLEN_REG, PATTSEL_REG, and USRCMD_REG (USRCMD_REG="10" for Write file and "11" for Read file).
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 Serial console when command is completed.

3.3 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 Serial console. 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 256 GB Samsung 860 Pro is shown in Figure 4-1.

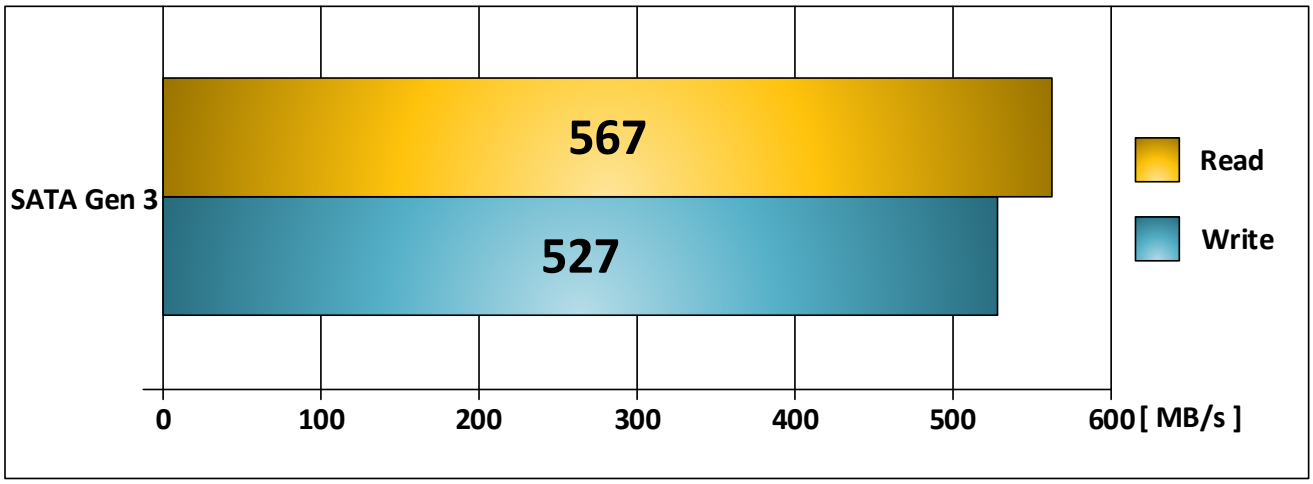

Figure 4-1 Test Performance of exFAT IP demo for SATA by using Samsung 860 Pro SSD

By using SATA Gen3 on ZC706 board, write performance is about 527 Mbyte/sec and read performance is about 567 Mbyte/sec.

5 Revision History

Revision	Date	Description
1.0	19-Nov-18	Initial release