FTP Server Demo reference design

Rev1.0 7-Feb-20

1 Introduction

File Transfer Protocol (FTP) is a standard network protocol used for file transmission between a client and server on a network. For the reliable and efficient transmission, TCP/IP is used as lower layer.

Reference documents

1. File Transfer Protocol: http://tools.ietf.org/html/rfc959
2. File Transfer Protocol: http://www.tcpipguide.com/free/t_FileTransferProtocolFTP.htm
3. FTP Sequence: www.eventhelix.com/realtimemantra/networking/FTP.pdf
4. List of FTP commands: http://en.wikipedia.org/wiki/List_of_FTP_commands
5. List of FTP server return codes: http://en.wikipedia.org/wiki/List_of_FTP_server_return_codes

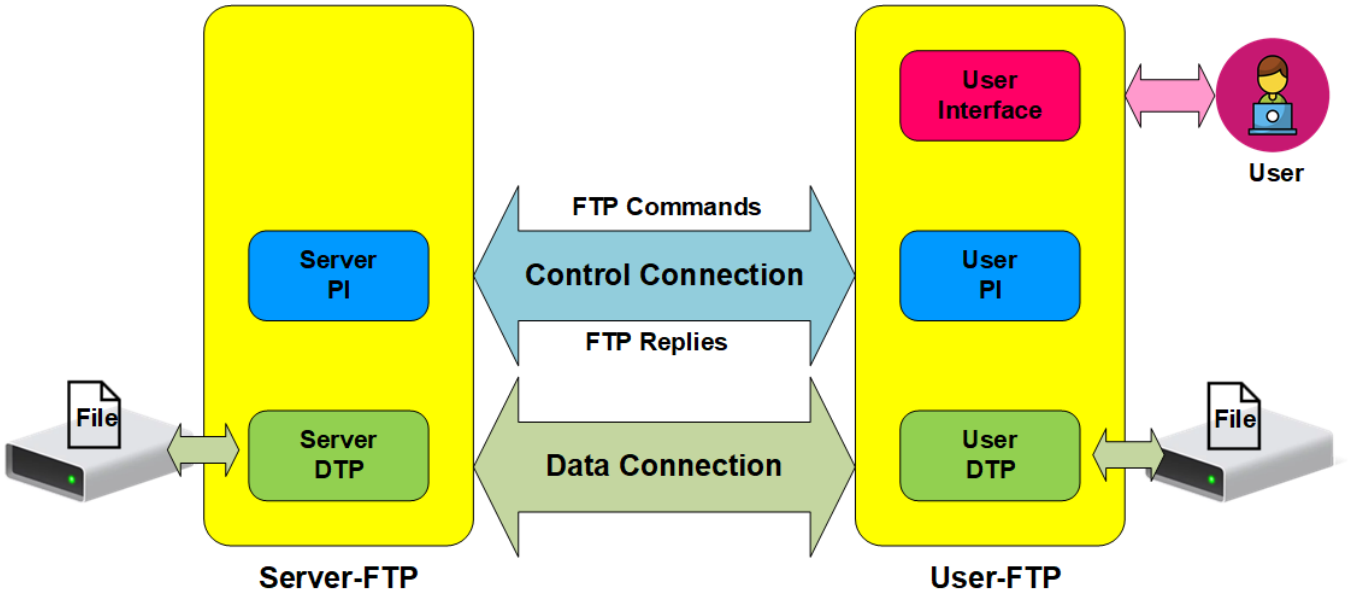

Figure 1-1 Model of FTP Use

In the model described in Figure 1-1, there are two hosts with their own data storage. One is server with shared data in the network and another is client for general users who can authenticate themselves with sign-in protocol, normally in form of username and password. For this protocol, two connections are required, i.e. control connection for FTP command and FTP replies transferring between client and server and data connection for data transmission between two hosts.

Control connection is taken responsibility by the server-protocol interpreter (Server-PI) and the user-protocol interpreter (User-PI). The server-PI listens on its own port number (Port 21 is used since it is a well-known control port) until the connection is established. After that, server-PI can receive FTP commands from User-PI, send standard FTP replies over the control connection in response of the command, and manage the server data transfer process (Server-DTP). For client side, User-PI is responsible to forward the commands received from the user interface to Server-PI, receive FTP replies, and manage the user data transfer process (User-DTP).

Data connection is taken responsibility by the server data transfer process (Server-DTP) and the user data transfer process (User-DTP) for sending or receiving data. The data connection establishment can be done by Server-DTP (active mode) or User-DTP (passive mode). Passive mode is commonly used to resolve the firewall problem on FTP client. The data connection is established to transfer files between the server and the client and terminated when finishing file transferring. Both Server-DTP and User-DTP interact with the local file system to read or write files.

User Interface provides the interface for a person requiring to obtain file transfer service through the FTP software to issue commands and check the responses.

1.1 FTP Connection Establishment and User Authentication

After user initiates the connection on control port, the user authentication is the next process to allow only authorized user to access FTP server. The details of the connection establishment and login process are described as follows.

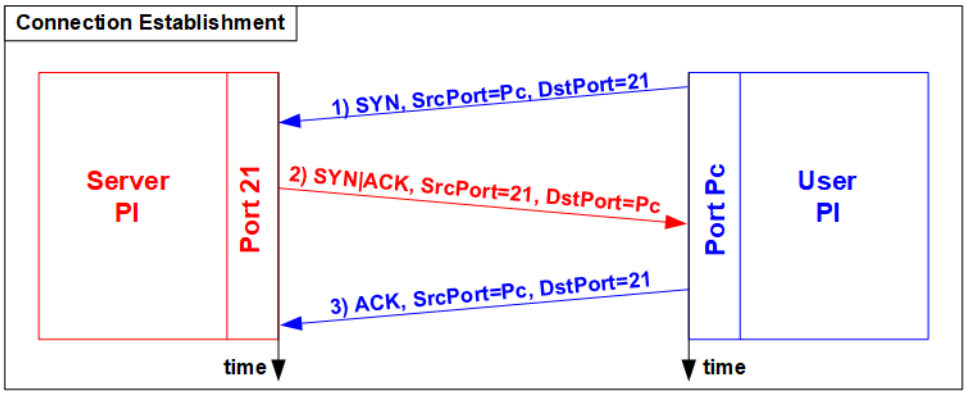

Figure 1-2 Connection Establishment on Control port

According to Transmission Control Protocol (TCP), 3-Way Handshake process is the way to establish the connection, as shown in Figure 1-2. FTP client starts sending TCP packet with SYN flag to server on Port 21 as the request to initiate the connection. After this packet is detected by FTP Server, the server returns the TCP packet with SYN and ACK flag to accept the connection. Next, client sends acknowledgement as ACK flag to complete this process. After that, server and client can communicate on the control port by using FTP commands and replies.

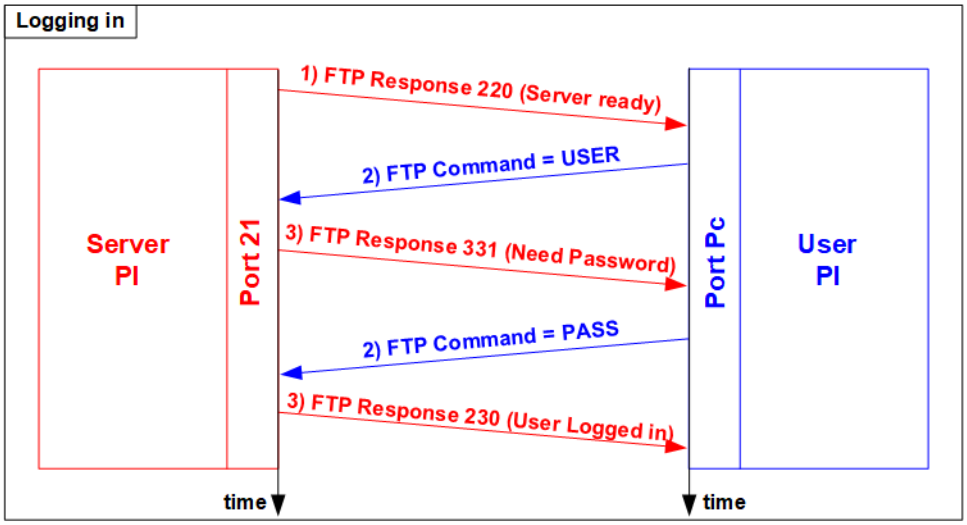

Figure 1-3 User Authentication

As shown in Figure 1-3, after the connection was established, the authorized users have to identify themselves with allowed username and password. For user authentication, client sends two FTP commands which are USER command with username and PASS with password respectively. Finally, the server replies the client with FTP response 230 to allow opening the new session.

1.2 FTP Data Connection Management by Passive Mode

This reference design implements the data connection establishment by Client (Passive mode) which is generally used. So, only the step for running as Passive mode is described in this topic.

Some FTP commands e.g. LIST (List subdirectories or files), STOR (Store files), and RETR (Retrieve files) need data or file transferring through the data connection. Before transferring data in previously mentioned commands, PASV command is firstly sent from client to specify the parameters for the data connection, i.e. IP address and port number of Server-DTP.

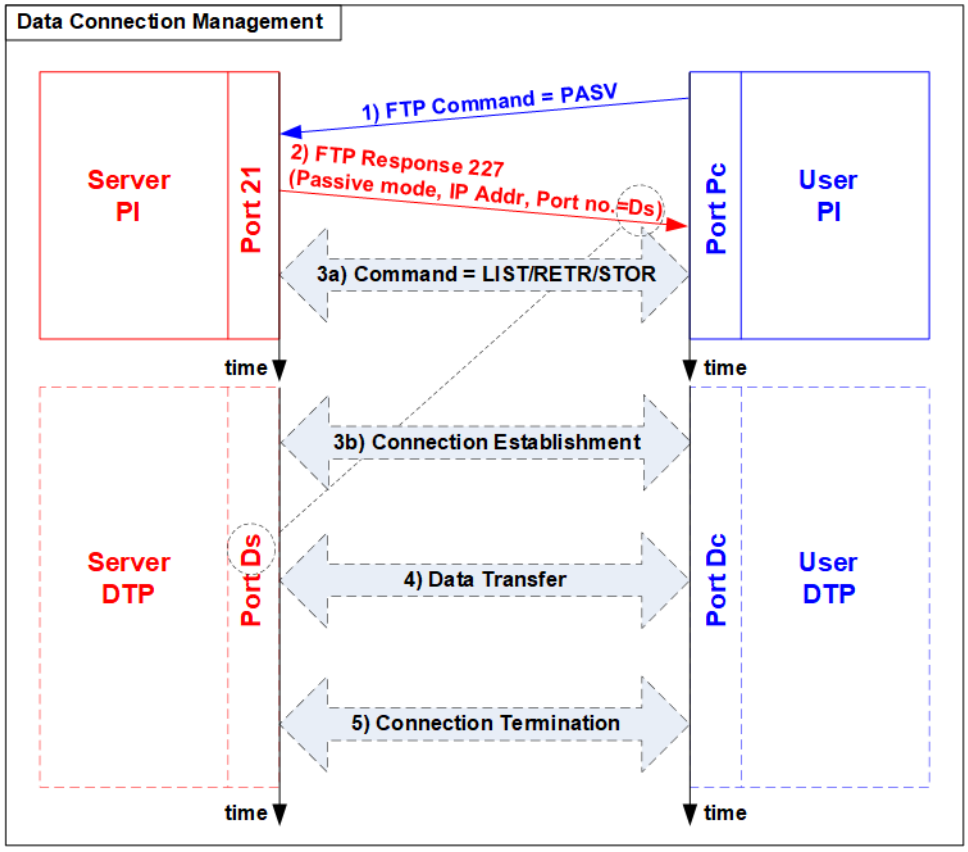

Figure 1-4 Data Connection Management in Passive Mode

After receiving PASV command, the server returns FTP response 227 to inform IP address and port number of Server to Client. Assume that the data port number of the server is equal to Ds. Server-PI instructs Server-DTP to listen on Port Ds and wait for data connection establishment. Next, User-PI sends FTP command which requires data transfer to Server-PI. At the same time, User-DTP establishes the data connection to transfer data with Server-DTP. Step 3a) for transferring FTP command and step 3b) for the data connection establishment can be swapped, depending on FTP client behavior. When the data transfer is finished, the data connection is terminated by the side transmitting data.

1.3 LIST Command

The LIST command is issued to transfer information about files in the specified directory, stored on the server's side, through an established data connection. PASV command is sent firstly to initialize the data connection. After that, User-PI sends LIST command to Server-PI while User-DTP establishes data connection. Step 2a) for transferring FTP command and step 2b) for the data connection establishment is possibly swapped.

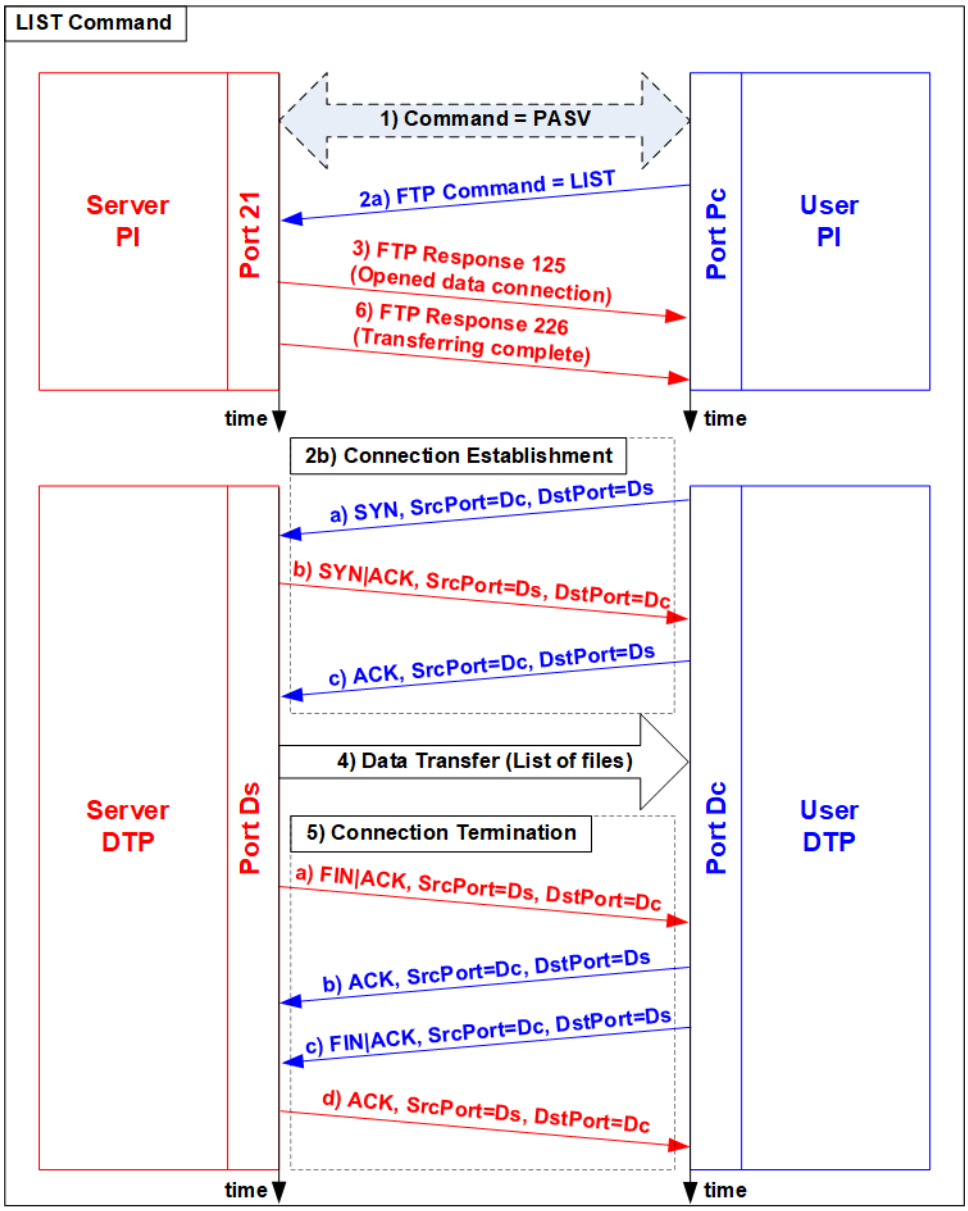

Figure 1-5 LIST Command Operation

For the server side, after the data connection was established, Server-PI returns response 125 for LIST command and Server-DTP returns information of files or the list of files to client. When the data transfer is finished, Server-DTP terminates the data connection. Finally, Server-PI sends response 226 to complete LIST command operation.

1.4 STOR Command

The STOR command is issued by client when the user requires to upload a copy of a file to store on the server's storage. The client provides the file name for uploading with the STOR command. If the file already exists on the server, it is replaced by the uploaded file. Otherwise, the uploaded file is created. Next, the server decodes file name to be a parameter to store a file. After that, data transmission begins, similar to the operation of LIST command. Data transfer direction is opposite from LIST and RETR command by sending from User-DTP (client) to Server-DTP (server). User-DTP terminates the data connection after finishing all data transmission.

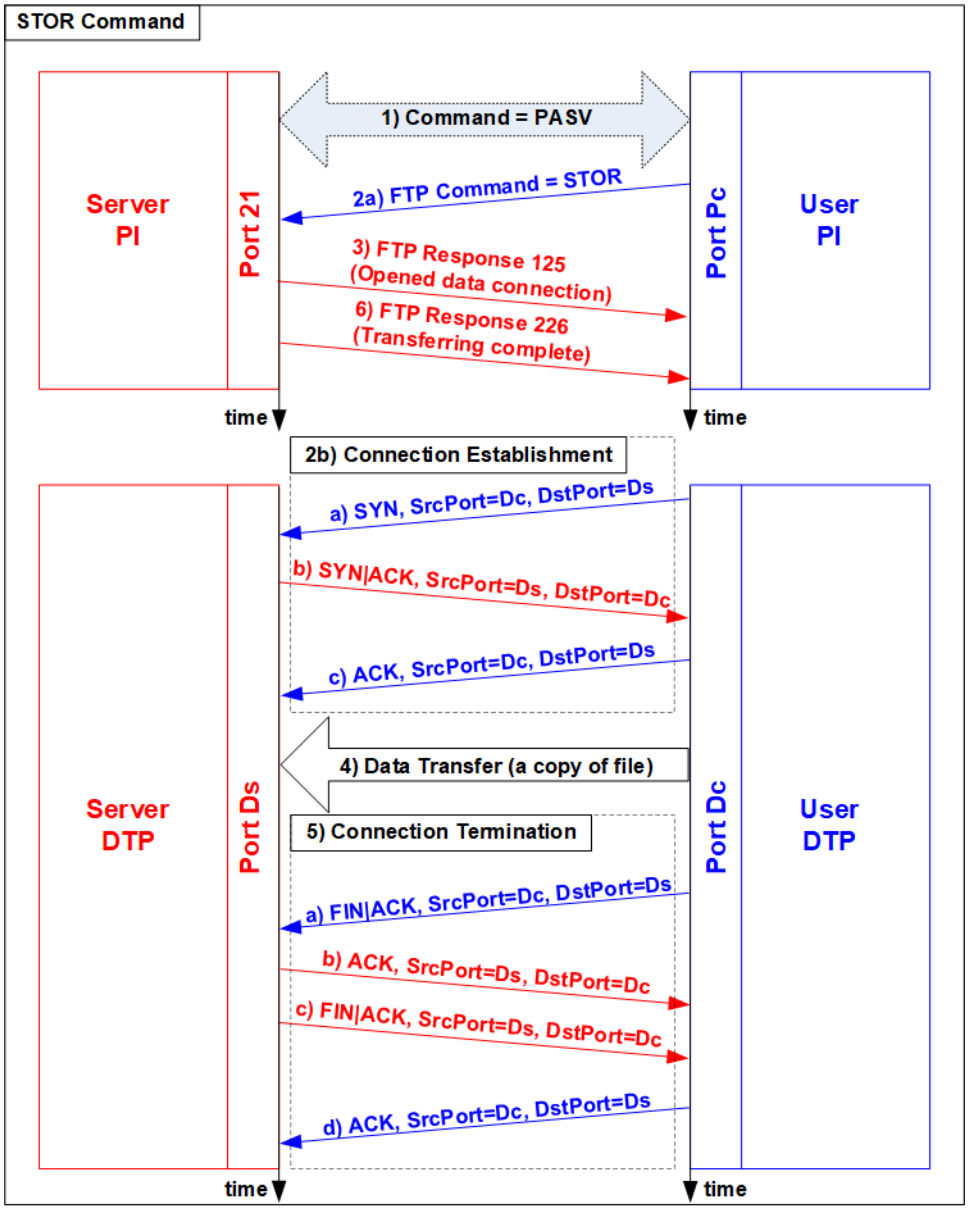

Figure 1-6 STOR Command Operation

1.5 RETR Command

The RETR command is sent from the client when user requires to download a copy of a file on the server. The client provides the file name along with the RETR command. On the server side, after the file name is decoded, the data of the requested file is returned. The step of data sequence is similar to LIST command. The data connection is released by Server-DTP after the data transfer is completed.

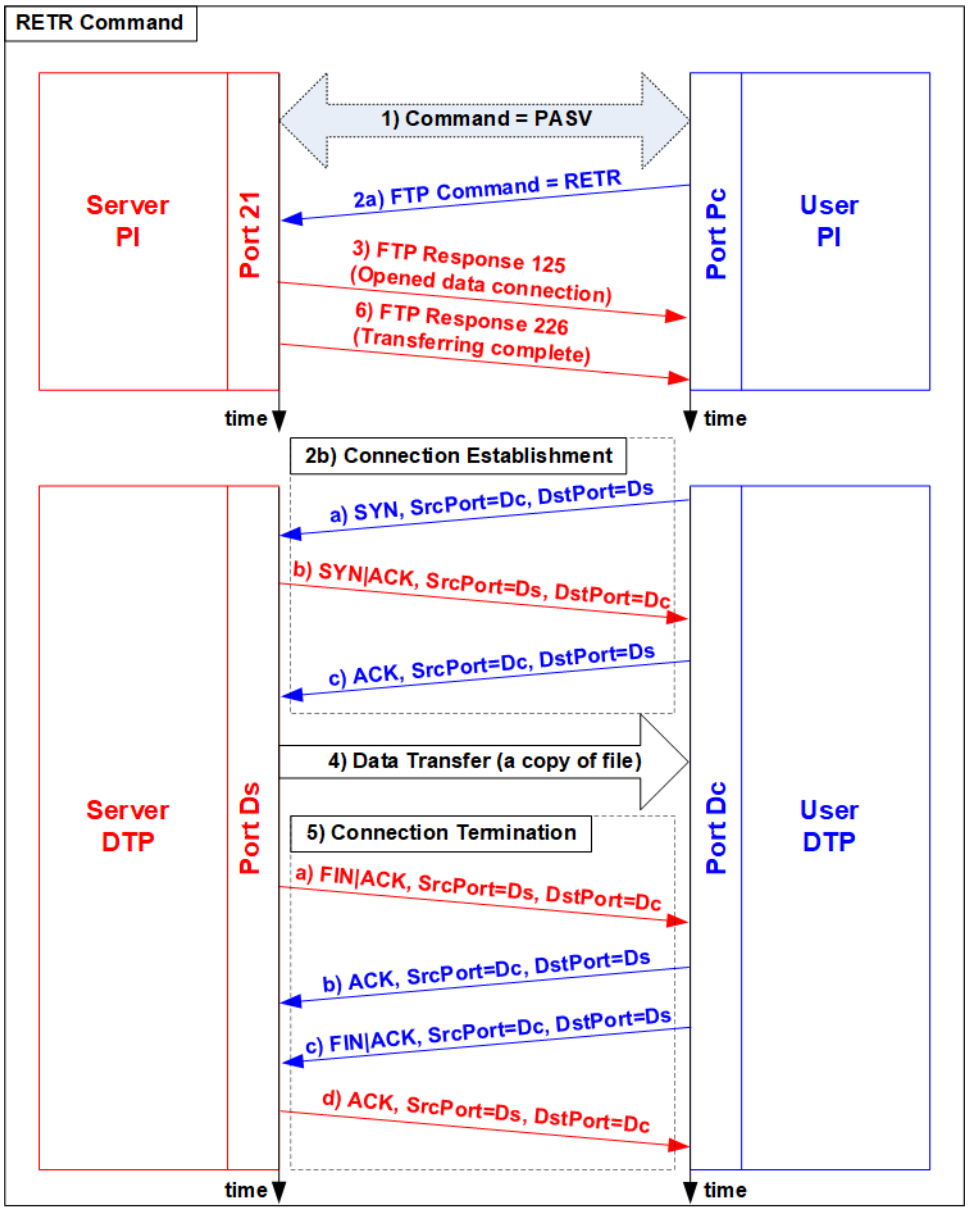

Figure 1-7 RETR Command Operation

2 Hardware Structure

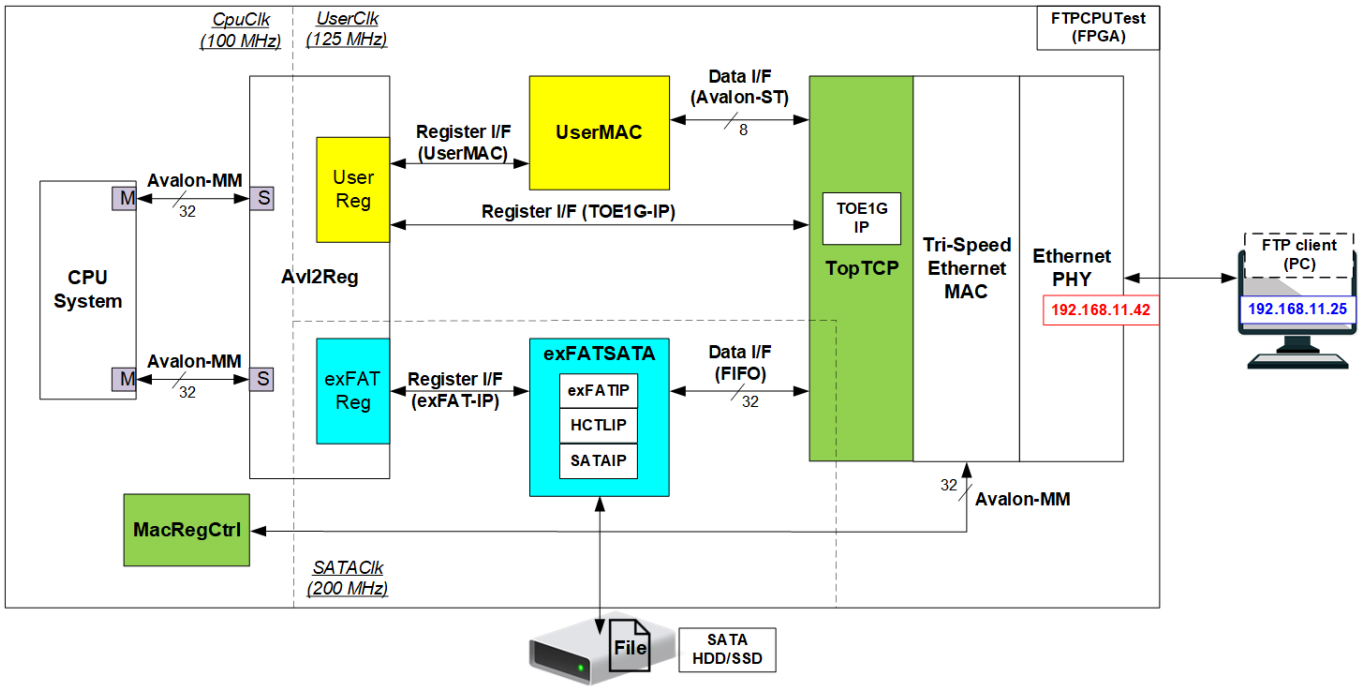

Figure 2-1 Demo Block Diagram

As described in the introduction, at least two connections are necessary for running FTP protocol, i.e. control connection for transferring FTP command and response and data connection for the command which needs to transfer data such as LIST, STOR, and RETR command.

The control connection is handled by CPU system. All data for control connection are created and monitored by CPU system through UserReg and UserMAC. When FTP client sends the new command, UserMAC forwards the packet from Tri-speed Ethernet MAC (TEMAC) to CPU system for processing. After that, CPU returns FTP response to UserMAC which is forwarded to TEMAC. CPU system accesses the hardware through Avalon-MM bus which UserReg and exFATReg are mapped to different address.

The data connection is handled by TOE1G-IP which is implemented inside TopTCP. The data source of TOE1G-IP has two sources. First is file information, created by CPU system, when operating LIST command. Second is data in the requested file, read from SATA device by exFATSATA module, when operation RETR command. The received data in the data connection is directly connected to exFATSATA module for operating STOR command.

By using TOE1G-IP with exFAT for SATA system consisting of exFAT-IP, HCTL-IP, and SATA-IP, data transferring in FTP demo can achieve the best performance on 1G Ethernet speed. Though exFAT for SATA system performance is equal to SATA device speed (about 530 – 560 MB/s), the performance in the demo is limited by 1 Gb Ethernet speed which maximum speed is about 120 – 130 MB/s.

More details of the hardware inside the demo are described in this topic.

2.1 Ethernet PHY

Ethernet PHY is implemented by external PHY chip. The interface of external PHY chip must be matched to Ethernet MAC. PHY Interface could be selected to be GMII, RGMII, or SGMII. When running on Cyclone10GX, the interface is SGMII. Ethernet speed is fixed to 1 Gbps mode.

2.2 Tri-Speed Ethernet MAC

Link layer and PCS/PMA are implemented by Triple-speed Ethernet MAC, provided by Intel FPGA. EMAC has two user interfaces, i.e. Avalon stream for transferring data and Avalon-MM for configuration. In reference design, Avalon stream of EMAC is connected to TOE1G-IP while Avalon-MM interface is connected to MACRegCtrl module. The details about the register for EMAC configuration are described in “Configuration Register Space” topic within “Triple-Speed Ethernet MegaCore Function User Guide” document, provided by Intel. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_ethernet.pdf

2.3 MacRegCtrl

This module is designed to configure parameter of Triple Ethernet MAC and monitor EMAC status through Avalon-MM bus. The logic is simply designed by using state machine. This module runs once after system power up to initialize Ethernet MAC.

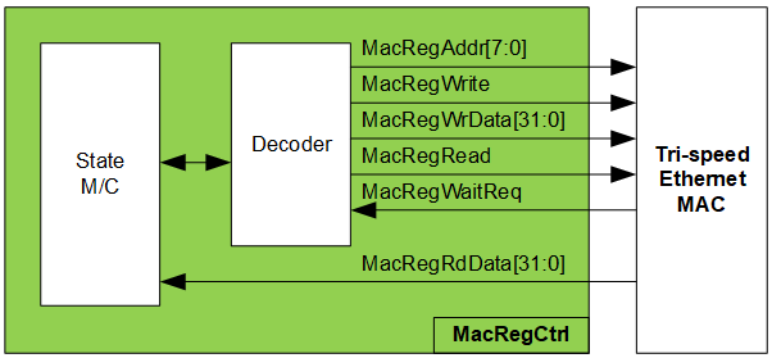

Figure 2-2 MACRegCtrl block diagram

The step to initialize EMAC is as follows.

1) Disable transmit and receive path of EMAC
2) Set software reset
3) Set frame length to support jumbo frame
4) Set Tx IFG length
5) Enable transmit and receive path of EMAC

For SGMII mode, the configuration step completes in step 5. Step 6 is necessary for RGMII mode to enable Receive/Transmit timing control function through MDIO.

6) Enable RGMII timing control

2.4 TopTCP

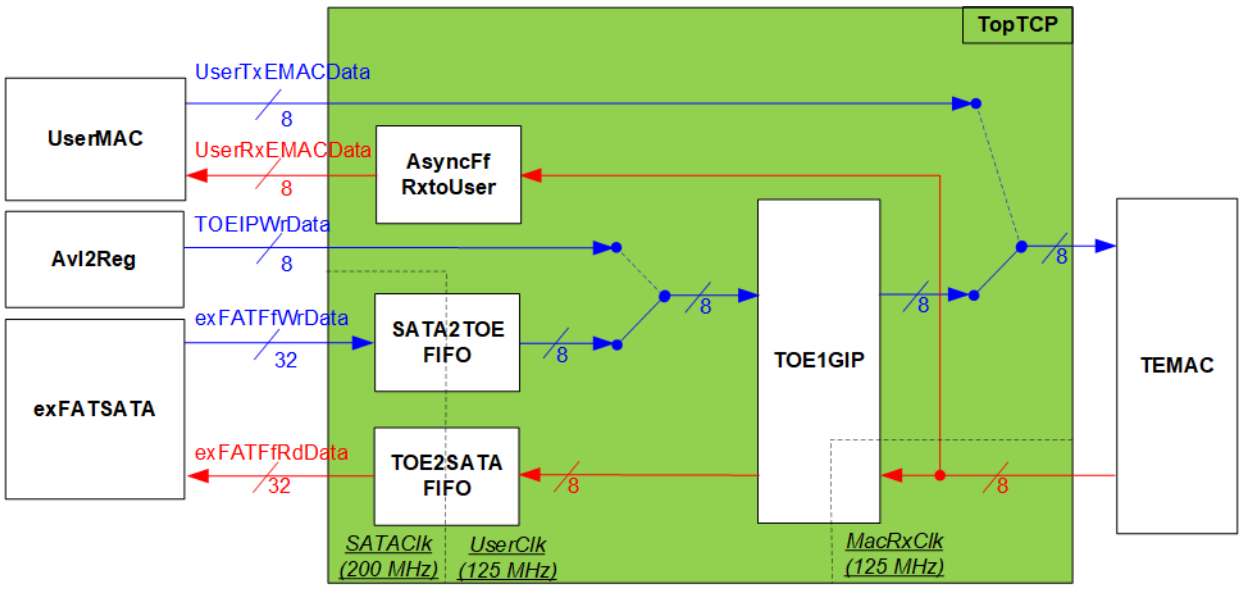

Figure 2-3 Data path of TopTCP block diagram

TopTCP is the module to interface with Tri-speed Ethernet MAC (TEMAC) for transferring data. To operation FTP operation, two ports are implemented on the hardware. First is control connection and another is data connection. The control connection is responsible to decode FTP command and return FTP response. The data size of control connection is small, so this port is implemented by using UserMAC controlled by CPU which is low-speed channel. The data connection is applied to transfer data of each file in RETR and STOR command and file list in LIST command. Since data size of data connection in RETR and STOR command is big, this port is implemented by using TOE1G-IP to achieve high-speed performance. The data source of TOE1G-IP is selected between Avl2Reg and exFATSATA. File list in LIST command is created by CPU through Avl2Reg while data in the file when running RETR command is read from SATA device by exFATSATA.

After finishing the connection establishment, the port number of UserMAC is the port for control connection while the port number of TOE1G-IP is the port for data connection. Though all received packets from TEMAC are forwarded to both UserMAC and TOE1G-IP, the packet filtering inside UserMAC and TOE1G-IP has never operated the same received packet by checking different port number.

There are three clock domains inside TopTCP, i.e. SATAClk which is applied to interface with exFATSATA, UserClk which is the main clock of TopTCP, and MacRxClk which is the clock synchronous with Rx data stream from TEMAC. Three asynchronous FIFOs are designed in TopTCP to support clock-crossing and different data bus size between write interface and read interface. First, AsyncFfRxtoUser stores the received packet in control connection of TEMAC which is run under MacRxClk and forwards to UserMAC which is run under UserClk. Next, SATA2TOEFIFO stores the read data returned from SATA device which is 32-bit size and runs under SATAClk and forwards to TOE1G-IP which uses 8-bit data and runs under UserClk. Last FIFO is TOE2SATAFIFO which is similar to SATA2TOEFIFO, but the data direction is reversed.

2.4.1 AsyncFfRxClktoUserClk

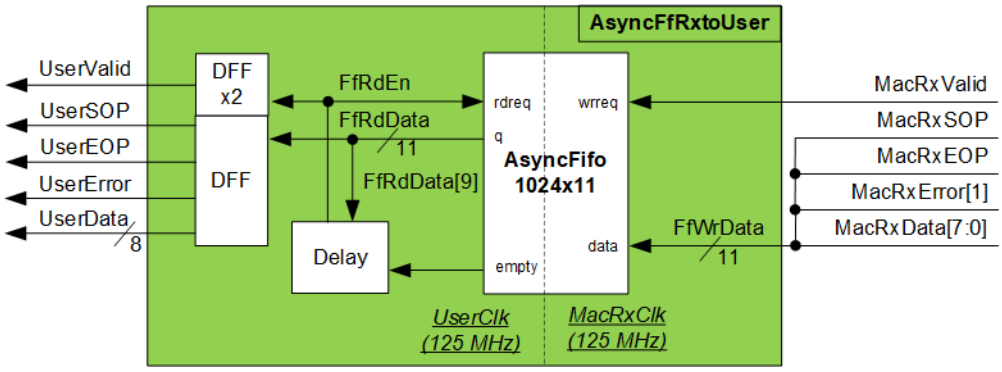

Figure 2-4 AsyncFfRxClktoUserClk module

Received data stream from EMAC is synchronous with MacRxClk. This module is designed to generate received data stream of EMAC to be synchronous with UserClk for decoded by UserMAC. Delay logic is designed to hold starting time to read data stream from AsyncFifo1024x11. The 1st data is read from AsyncFifo1024x11 after FfEmpty is de-asserted to '0' for 4 clock cycles. After that, the data is constantly read until AsyncFifo1024x11 is empty.

2.4.2 TOE1G-IP

TOE1G-IP implements TCP/IP stack and offload engine. Control and status signals for user interface are accessed through register interface. Data interface is accessed through FIFO interface. More details are described in datasheet.
https://dgway.com/products/IP/TOE1G-IP/dg_toe1gip_data_sheet_intel_en.pdf

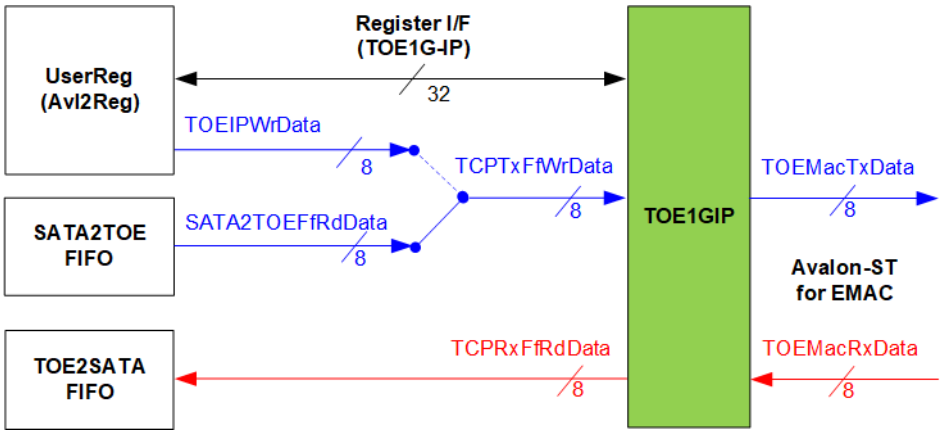

Figure 2-5 TOE1G-IP interface in the demo

In this demo, TOE1G-IP is applied for transferring data of data connection at high-speed rate. The network parameters of TOE1G-IP are set by CPU through Avl2Reg for data connection of FTP command. Also, total transmit size of TOE1G-IP is calculated and set by CPU. The CPU decodes the data size of the requested file for returning in RETR command. To operate STOR command, received data of TOE1G-IP is stored to TOE2SATAFIFO when RxFIFO within TOE1G-IP is not empty and TOE2SATA FIFO is not full.

2.4.3 SATA2TOEFIFO

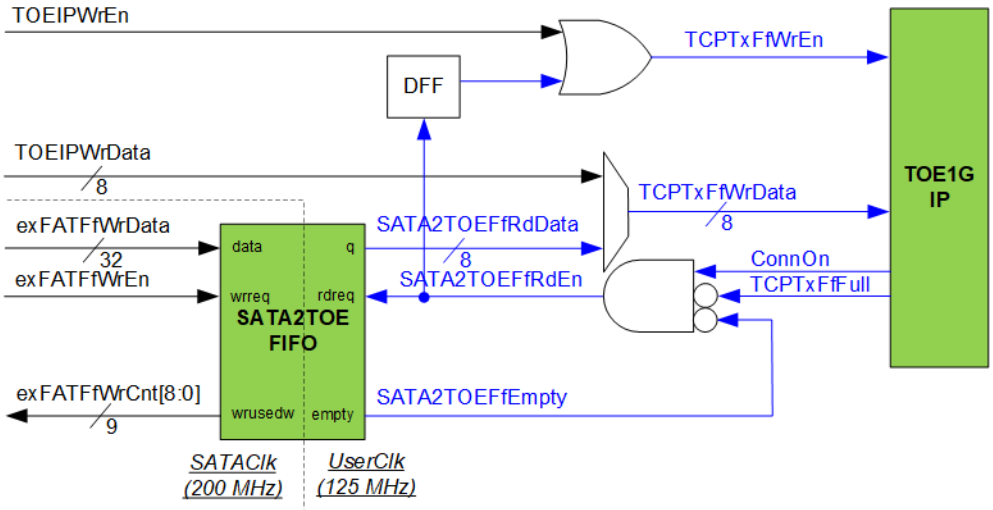

Figure 2-6 Transmitted data path of TOE1G-IP

To transmit data to client through TOE1G-IP, there are two data sources, i.e. the list of file name stored in SATA SSD written by CPU firmware and the data of the file returned from SATA device through exFATSATA module.

As shown in Figure 2-6, to generate request for reading the data of the file from SATA device and forwarding to client, the hardware must ensure that SATA2TOE FIFO has remaining data (SATA2TOEFfEmpty='0'), Tx FIFO of TOE1G-IP is not full (TCPTxFfFull='0'), and data connection establishment is completed (ConnOn='1'). After that, the data transmits to TOE1G-IP by asserting TCPTxFfWrEn to '1'. File data from exFAT-IP (SATA2TOEFfRdData) is forwarded to TOE1G-IP when TCPTxFfWrEn is asserted to '1'. Otherwise, the data written by CPU (TOEIPWrData) is transferred with write enable signal (TOEIPWrEn).

The FIFO size is 4 Kbyte using different bus size and different clock domain for write interface and read interface. exFATFfWrCnt is monitored by exFATSATA module to check the free space size in the FIFO. When the free space is much enough and data of SATA device is ready, 512-byte data is sent to SATA2TOEFIFO.

2.4.4 TOE2SATAFIFO

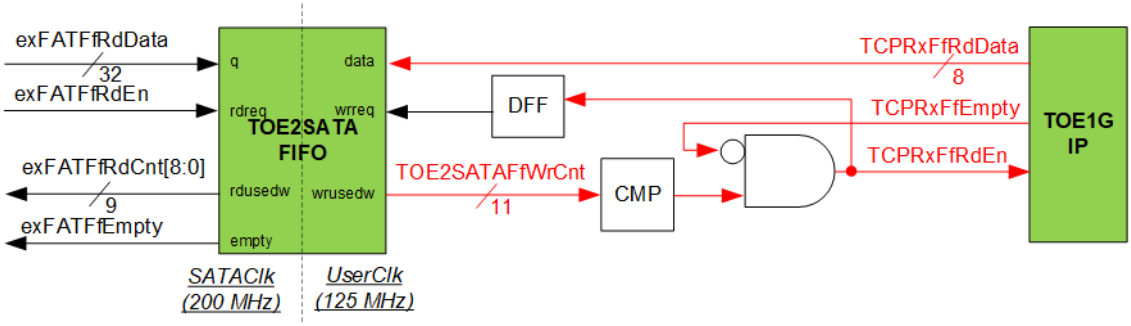

Figure 2-7 Received data path of TOE1G-IP

The data read from client through TOE1G-IP is stored to TOE2SATA FIFO when operating STOR command. Data of the file received from FTP client is stored to SATA device through exFATSATA module. exFATFfRdCnt and exFATFfEmpty are monitored to check the available data in FIFO before forwarding to SATA device.

Comparing to SATA2TOEFIFO, the operation of TOE2SATAFIFO is conversely transferred. FIFO size is 4 Kbyte setting different bus size and different clock domain for write interface and read interface.

2.5 UserMAC

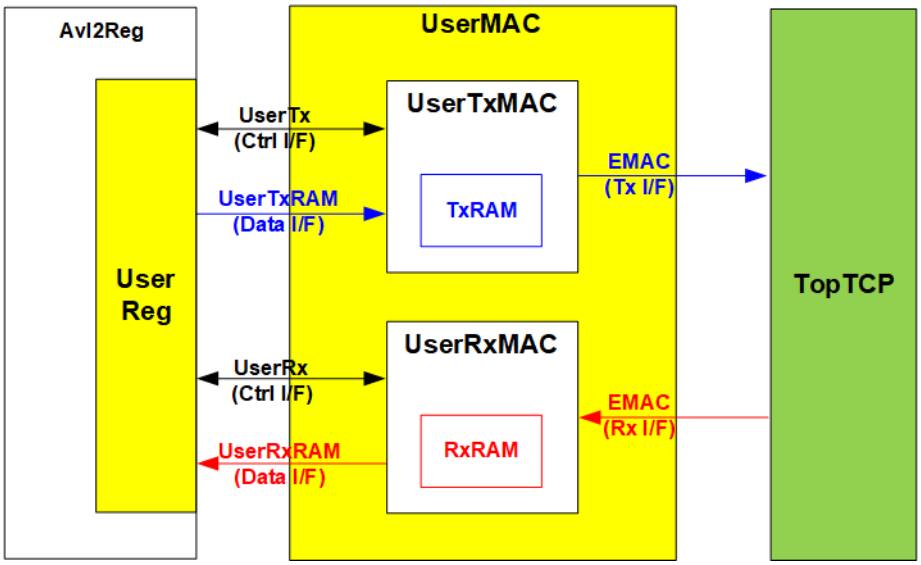

Figure 2-8 UserMAC Block Diagram

UserMAC is responsible to transfer TCP/IP packet for FTP control connection. The step of FTP control connection is designed as CPU firmware to access UserMAC registers through UserReg module. Data interface of UserMAC with UserReg is 32-bit RAM standard while the data interface with EMAC is 8-bit Avalon-ST standard.

UserMAC consists of two modules i.e. UserTxMAC and UserRxMAC. UserTxMAC includes TxRAM which stores the data written by CPU to EMAC. UserRxMAC includes RxRAM which stores the received packet from EMAC which has been proved by the packet filtering logic for CPU processing. More details of UserTxMAC and UserRxMAC are described as follows.

2.5.1 UserTxMAC

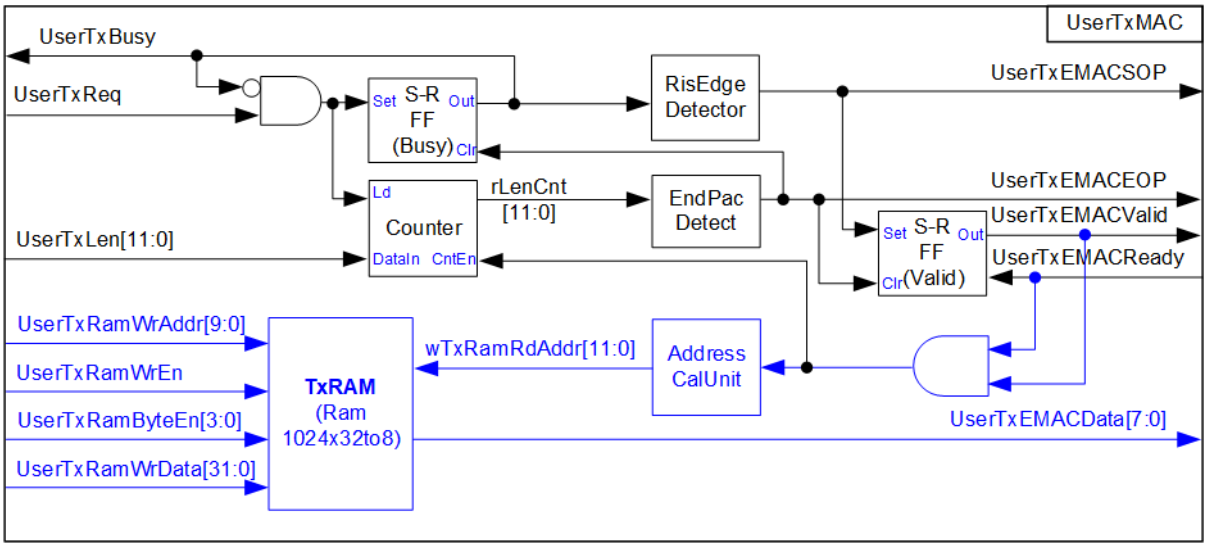

Figure 2-9 UserTxMAC Logic Diagram

The step to transfer the packet from TxRAM to EMAC is as follows.

(1) CPU checks that UserTxBusy='0' to confirm UserTxMAC is in Idle state.
(2) CPU writes transmit packet to TxRAM, starting from the 1st address (TxRamWrAddr=0).
(3) CPU sets UserTxLen=Packet size and asserts UserTxReq to '1' to begin data transmission.
(4) UserTxBusy is asserted to '1' during transmitting data. At the same time, Counter loads total transfer size and count down to control packet size. AddressCalUnit resets the read address (wTxRamRdAddr) to 0 for reading the 1st data of the packet to send to UserTxEMACData signal.
(5) UserTxEMACSOP is asserted to '1' to send the 1st data with asserting UserTxEMACValid to '1'. At the same clock, UserTxEMACData is valid on the bus.
(6) After UserTxEMACReady is asserted to '1' to accept the 1st data of the packet, the next data is read out from TxRAM and sent to EMAC continuously until end of the packet.
(7) UserTxEMACEOP is asserted to '1' after the remaining packet size (rLenCnt) is equal to 2.
(8) UserTxEMACBusy is de-asserted to '0' when the end of packet is transmitted to EMAC.

2.5.2 UserRxMAC

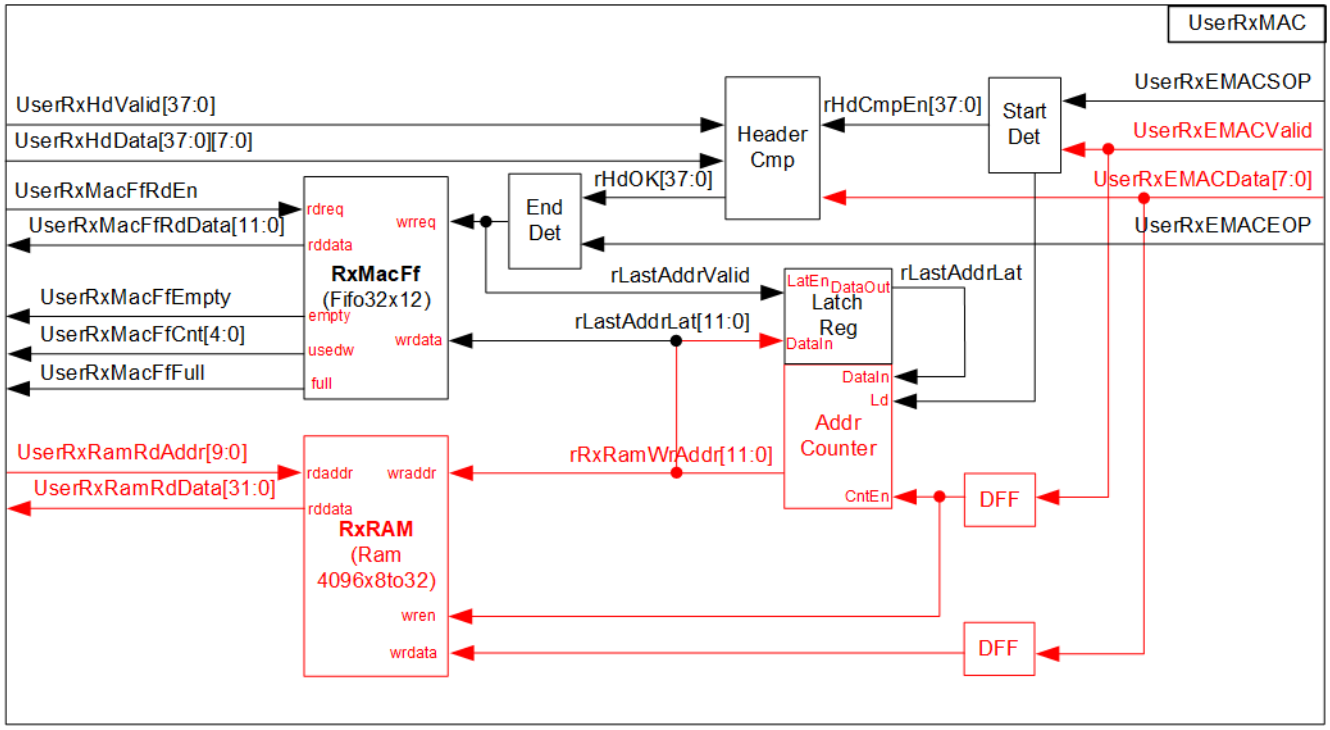

Figure 2-10 UserRxMAC Logic Diagram

The step when the new packet is received from EMAC is as follows.

(1) a) When start-of-packet (UserRxEMACSOP) is detected, the enable signal to compare 38-byte header of each packet (rHdCmpEn) is asserted to '1' for 38 clock cycles.

b) At the same time, the address counter to generate write address of RxRAM loads the latest position of valid packet from rLastAddrLat. After that, the write address is increased to store the next data to the next address of RxRAM.

c) The received packet is stored to RxRAM by using UserRxEMACValid as write enable of RxRAM and UserRxEMACData as write data.

(2) Byte0 – 37 of received data (UserRxEMACData) are fed to header comparator module (HeaderCmp) to compare with the expected value (UserRxHdData), set by CPU firmware. UserRxHdValid is enable/disable flag to compare each byte of the header. Bit0 of UserRxHdValid is byte0 enable, bit1 is byte1 enable, and so on.

(3) When end-of-packet (UserRxEMACEOP) is detected, the header result flag (rHdOK) are read. The valid signal for updating the last write address (UserRxLastAddrValid) is asserted to '1' with the valid of rLastAddrLat when all headers are valid (rHdOK='1'). Otherwise, the valid signal is not asserted to '1' and rLastAddrLat still be the same value. So, RxMacFf stores only the last address of valid packet.

(4) CPU reads FIFO count (UserRxMacFfCnt) to check that new packet is received. When FIFO count is not equal to 0, CPU asserts read enable (UserRxMacFfRdEn) to read the last valid byte address (UserRxMacFfRdData). After that, CPU reads and decodes the received packet until the read address (UserRxRamRdAddr) is equal to the last address. CPU continues the next packet processing if FIFO count still not be equal to 0. In the hardware, UserRxRamRdAddr is the address for 32-bit data while rRxRamWrAddr is the address for 8-bit data.

The 38-byte header of received packet for control connection of FTP application is shown as Figure 2-11.

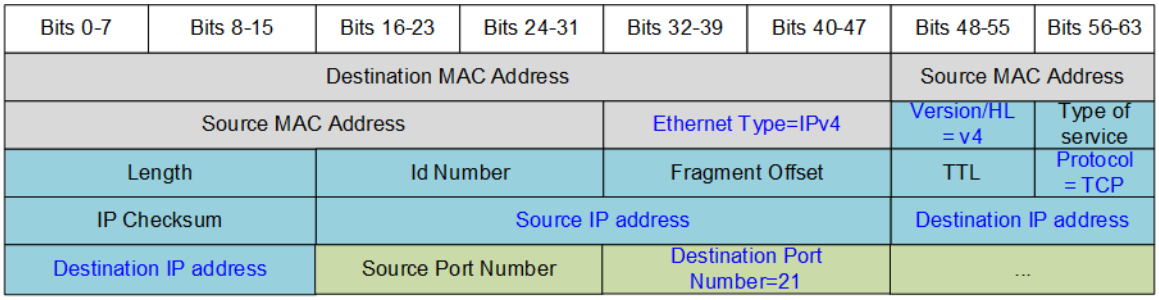

Figure 2-11 TCP/IP Packet header for FTP application

HeaderCmp is the packet filtering module which is controlled by CPU firmware. 38-byte header is verified by the set value from CPU. In the demo, CPU firmware assigns the enable flag and the verification value to compare six parameters as follows.

1) Ethernet Type (2 bytes) = 0x0800 (IPv4)
2) IP version (1 byte) = 0x45 (Version 4)
3) Protocol (1 byte) = 0x06 (TCP Protocol)
4) Source IP Address (4 bytes) = IP address of FTP client
5) Destination IP Address (4 bytes) = IP address of FPGA
6) Destination Port Number (2 bytes) = 0x0015 (Port 21)

When the header of the packet is matched to the above parameters, the received packet is stored to RxRAM for CPU processing.

2.6 exFATSATA

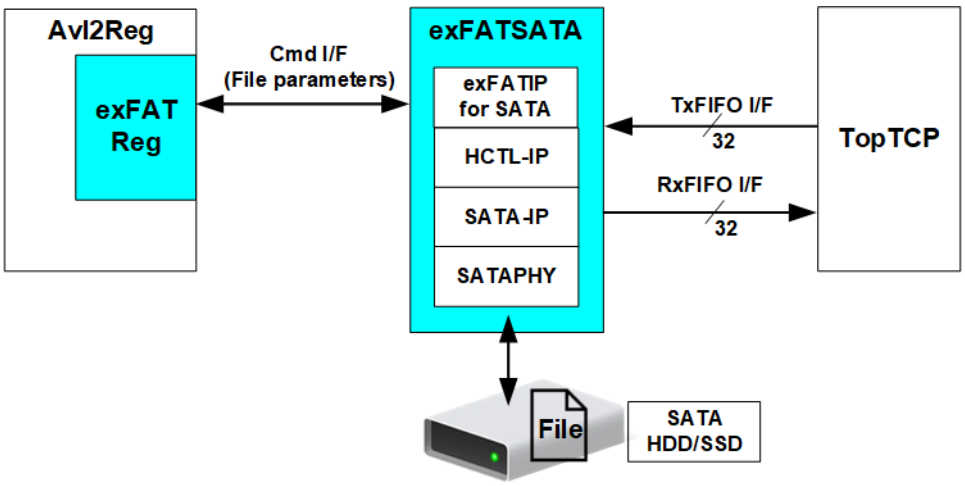

Figure 2-12 UserMAC Block Diagram

exFATSATA is exFAT IP core for SATA system which consists of four modules, i.e. exFAT-IP, HCTL-IP, SATA-IP and SATA PHY module. More details of exFAT-IP are described in the data sheet.
https://dgway.com/products/IP/SATA-IP/Altera/dg_exfatip_sata_data_sheet_intel_en.pdf

In FTP demo system, the command interface for setting file parameters is conrolled by CPU through exFAT Reg inside Avl2Reg module. The data interface using FIFO standard is connected to TopTCP to transfer data with TOE1G-IP.

2.7 CPU and Peripherals

32-bit Avalon-MM is applied to be the bus interface for the CPU accessing the peripherals such as Timer and JTAG UART. To control and monitor the test system, the control and status signals are connected to register for CPU access as a peripheral through 32-bit Avalon-MM bus. CPU assigns the different base address and the address range to each peripheral for accessing the internal registers of the peripheral.

In the reference design, the CPU system is built with two additional peripherals to access the two hardware systems. First is applied to control Ethernet system which is run under UserClk. Second is applied to control exFAT system which is run under SATAClk. The base address and the range for accessing the hardware are defined in the CPU system, so the hardware logic must be designed to support Avalon-MM bus standard for writing and reading the register. Avl2Reg module is designed to connect the CPU system as shown in Figure 2-13.

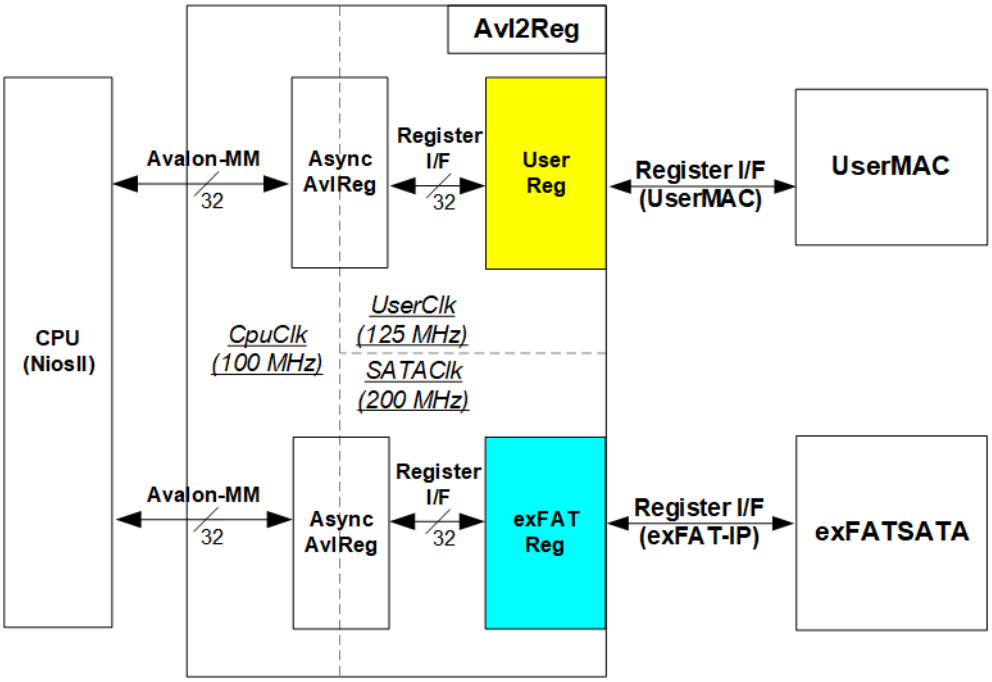

Figure 2-13 Avl2Reg Block diagram

In Figure 2-13, Avl2Reg consists of three modules, i.e. AsyncAvlReg, UserReg, and exFATReg. The main responsibility of AsyncAvlReg is to convert 32-bit Avalon interface in CPU clock domain (100 MHz) to register Interface in User clock domain (125 MHz) and SATA clock domain (200 MHz). CPU firmware communicates with both UserReg module and exFATReg module. Two AsyncAvlReg modules are included in the system for supporting two clock domains. UserReg and exFATReg include the register file of the parameters and the status signals for UserMAC and exFATSATA respectively. More details of each module are described as follows.

2.7.1 AsyncAvlReg

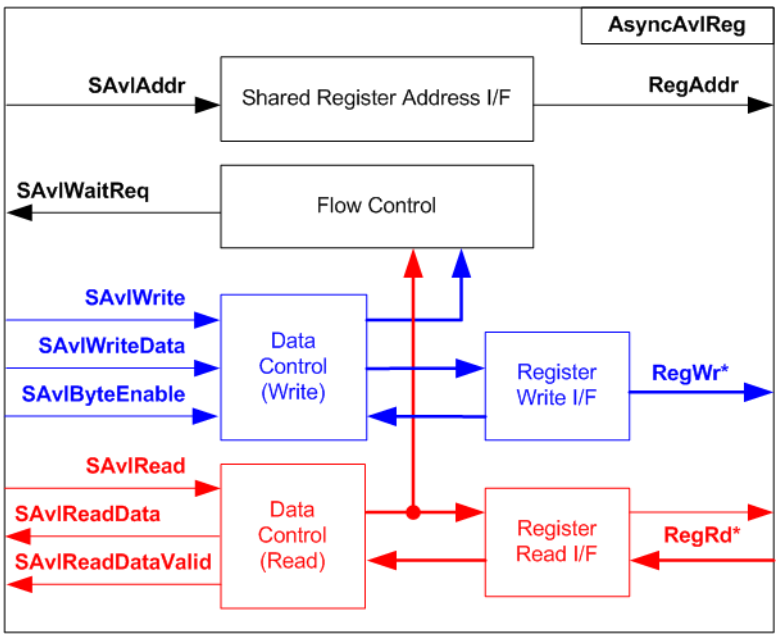

Figure 2-14 AsyncAvlReg Interface

The signal on Avalon-MM bus interface can be split into three groups, i.e. Write channel (blue color), Read channel (red color), and Shared control channel (black color). More details of Avalon-MM interface specification are described in following document.
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

According to Avalon-MM specification, only one command (write or read) can be operated at a time. The logics inside AsyncAvlReg are split into three groups, i.e. Write control logic, Read control logic, and Flow control logic. Flow control logic to control SAvlWaitReq is applied to hold the next request from Avalon-MM interface while the current request is operating. Write control I/F and Write data I/F of Avalon-MM bus are latched and transferred as Write register. On the other hand, Read control I/F and Read data I/F of Avalon-MM bus are latched and transferred as Read register. Address I/F of Avalon-MM is latched and transferred to Address register interface as well.

The simple register interface is designed to be compatible to general RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RegRdReq and RegRdValid signal. 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 2-15.

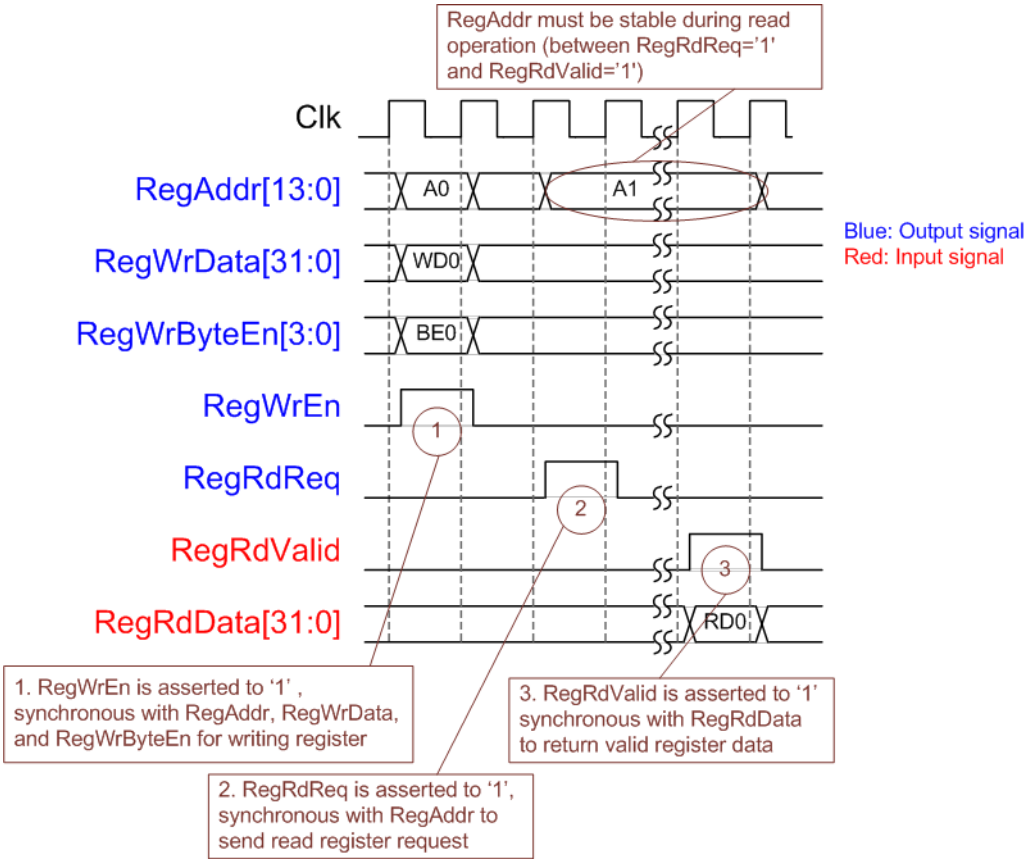

Figure 2-15 Register interface timing diagram

1) To write register, the timing diagram is the same as general RAM interface. RegWrEn is asserted to '1' with the valid 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, i.e. bit[0] for RegWrData[7:0], bit[1] for RegWrData[15:8], and so on.
2) To read register, AsyncAvlReg 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.
3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAvlReg forwards the read value to SAvlRead interface.

2.7.2 UserReg

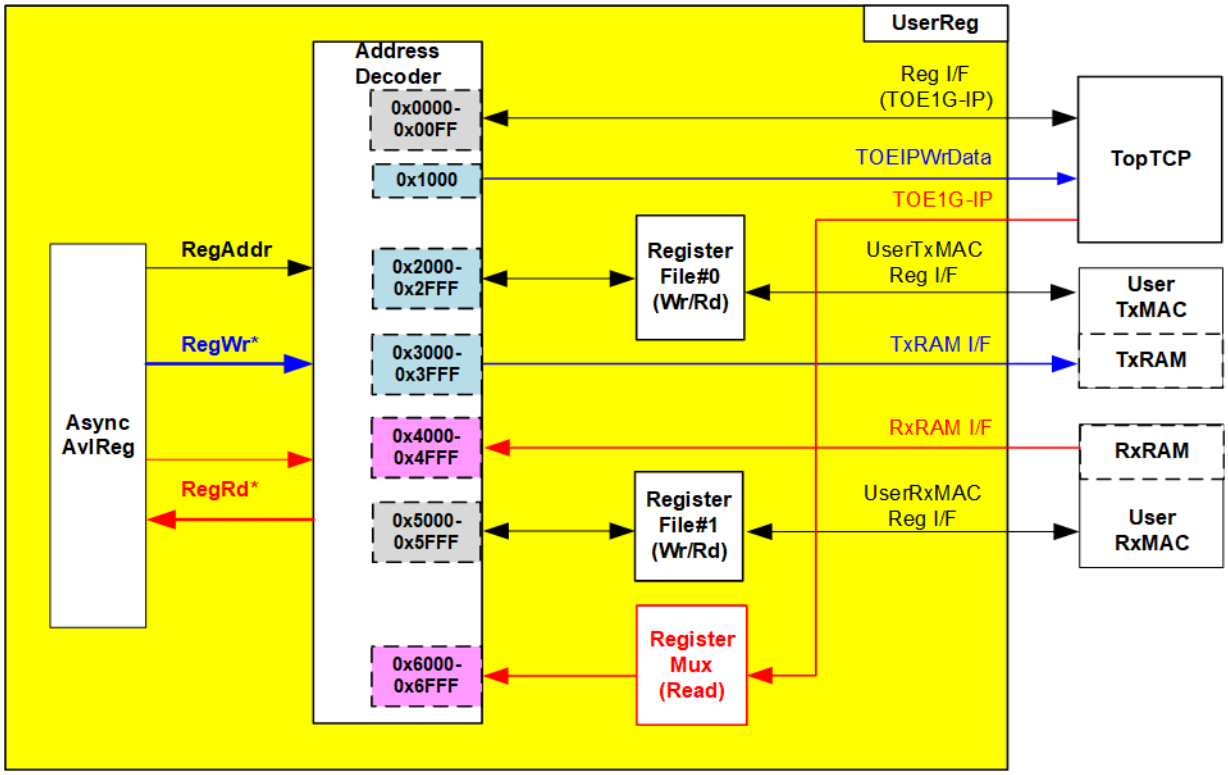

Figure 2-16 UserReg Block diagram

UserReg includes the registers for the module processing Ethernet packet such as TOE1G-IP inside TopTCP, UserTxMAC, and UserRxMAC. The address range is split into seven areas.

1) 0x0000 – 0x00FF: mapped to TOE1G-IP register for setting and monitoring network parameters and control signals.
2) 0x1000: mapped to write TCP data which is the list of file name sent to TOE1G-IP for LIST command.
3) 0x2000 – 0x2FFF: mapped to the control/status signals of UserTxMAC for transmitting packet.
4) 0x3000 – 0x3FFF: mapped to write TxRAM which stores Tx packet of FTP control connection. This area is write-access only.
5) 0x4000 – 0x4FFF: mapped to read RxRAM which stores Rx packet of FTP control connection. This area is read-access only.
6) 0x5000 – 0x5FFF: mapped to the control/status signals of UserRxMAC for receiving packet.
7) 0x6000 – 0x6FFF: mapped to read the status signal, output from TOE1G-IP. This area is read-access only.

Address decoder decodes the upper bit of RegAddr for selecting the active area. To read register, three-step multiplexers are designed to select the read data for returning to CPU. The latency of read data is equal to three clock cycles, so RegRdValid is created by RegRdReq with asserting three D Flip-flops.

More details of the address mapping within UserReg module is shown in Table 2-1.

Table 2-1 Memory Map of UserReg

Address Wr/Rd	Register Name Label in “ftpdemo.c”	Description
BA0+0x0000 - BA0+0x00FF: TOE1G-IP Register Area More details of each register are described in TOE1G-IP datasheet		
BA0+0x0000	TOE_RST_REG	Mapped to RST register within TOE1G-IP
BA0+0x0004	TOE_CMD_REG	Mapped to CMD register within TOE1G-IP
BA0+0x0008	TOE_SML_REG	Mapped to SML register within TOE1G-IP
BA0+0x000C	TOE_SMH_REG	Mapped to SMH register within TOE1G-IP
BA0+0x0010	TOE_DIP_REG	Mapped to DIP register within TOE1G-IP
BA0+0x0014	TOE_SIP_REG	Mapped to SIP register within TOE1G-IP
BA0+0x0018	TOE_DPN_REG	Mapped to DPN register within TOE1G-IP
BA0+0x001C	TOE_SPN_REG	Mapped to SPN register within TOE1G-IP
BA0+0x0020	TOE_TDL_REG	Mapped to TDL register within TOE1G-IP
BA0+0x0024	TOE_TMO_REG	Mapped to TMO register within TOE1G-IP
BA0+0x0028	TOE_PKL_REG	Mapped to PKL register within TOE1G-IP
BA0+0x002C	TOE_PSH_REG	Mapped to PSH register within TOE1G-IP
BA0+0x0030	TOE_WIN_REG	Mapped to WIN register within TOE1G-IP
BA0+0x0034	TOE_ETL_REG	Mapped to ETL register within TOE1G-IP
BA0+0x0038	TOE_SRV_REG	Mapped to SRV register within TOE1G-IP
BA0+0x1000 - BA0+0x3FFF: Tx interface for EMAC		
BA0+0x1000 Wr	Write data to TOE1G-IP (TOE_TXDATA_REG)	Wr [7:0] - Transmitted data written by CPU for sending through TOE1G-IP
BA0+0x2000 Wr/Rd	Total Transmit Length (TXEMAC_LEN_REG)	Wr [11:0] - Total transmitted size in byte unit. Valid from 1-0xFFF. UserTxMAC starts transmitting packet after this register is written. Rd [0] – Busy flag of UserTxMAC. ‘0’-Idle, ‘1’-Busy.
BA0+0x3000 Wr	TxRAM in UserTxMAC (TXRAM_BASE_ADDR)	Transmitted data written by CPU for sending through UserTxMAC
BA0+0x4000 - BA0+0x5FFF: UserRxMAC Area		
BA0+0x4000 – BA0+0x4FFF Rd	RxRAM in UserRxMAC (RXRAM_BASE_ADDR)	Received data stored in RxRAM within UserRxMAC
BA0+0x5000 – BA0+0x5024 Wr	UserRxMAC Header Data (RXEMAC_HDVAL)	Wr: 38-byte to set the packet filtering inside UserRxMAC for comparing byte#0 – byte#37 of received packet when header byte enable is asserted to ‘1’. If header byte enable is de-asserted to ‘0’, the filtering bypassed that byte. 0x5000[7:0]-byte#0, [15:8]-byte#1, [23:16]-byte#2, [31:24]-byte#3 0x5004[7:0]-byte#4, [15:8]-byte#5, [23:16]-byte#6, [31:24]-byte#7 … 0x5020[7:0]-byte#32, [15:8]-byte#33, [23:16]-byte#34, [31:24]-byte#35 0x5024[7:0]-byte#36, [15:8]-byte#37
BA0+0x5028 – BA0+0x502C Wr	UserRxMAC Header Byte Enable (RXEMAC_HDEN)	Wr: 38-bit to compare 38-byte header of received packet in UserRxMAC 0x50028[0] – Compare enable of byte#0, [1]-byte#1, …, [31]-byte#31 0x5002C[0] - byte#32, [1]-byte#33, .., [5]-byte#37
BA0+0x5040 Rd	RxEMAC Last Address (RXEMAC_LASTADDR)	Rd [11:0] – Read last address from RxMacFf. Read enable is asserted to ‘1’ for one clock cycle to read one data from RxMacFf. So, please check that RXEMAC_FFCNT is not equal to 0 before reading this register.
BA0+0x5044 Rd	FIFO Counter of RxMacFf (RXEMAC_FFCNT)	Rd [4:0] – FIFO counter to show total data in RxMacFf.
BA0+0x6000 - BA0+0x6FFF: TOE1G-IP Status Area		
BA0+0x6000 Rd	TOE1G Status Register (TOEIP_STS_REG)	Rd [0] - Mapped to Busy signal of TOE1G-IP ('0': Idle, '1': Busy) [1] - Mapped to ConnOn signal of TOE1G-IP [2] - Mapped to TCPTxFfFull signal of TOE1G-IP
BA0+0x6004 Wr/Rd	Interrupt Status of Ethernet interface (ETHER_INT_STS)	Wr [0] - Set ‘1’ to clear TimerInt latch value [8] – Set ‘1’ to clear interrupt when RxMacFf is full Rd [0] – ‘1’: Detect TimerInt from TOE1G-IP to be asserted to ‘1’, ‘0’: Normal [8] – ‘1’: Detect Full flag of RxMacFf to be asserted to ‘1’, ‘0’: Not full

2.7.3 exFATReg

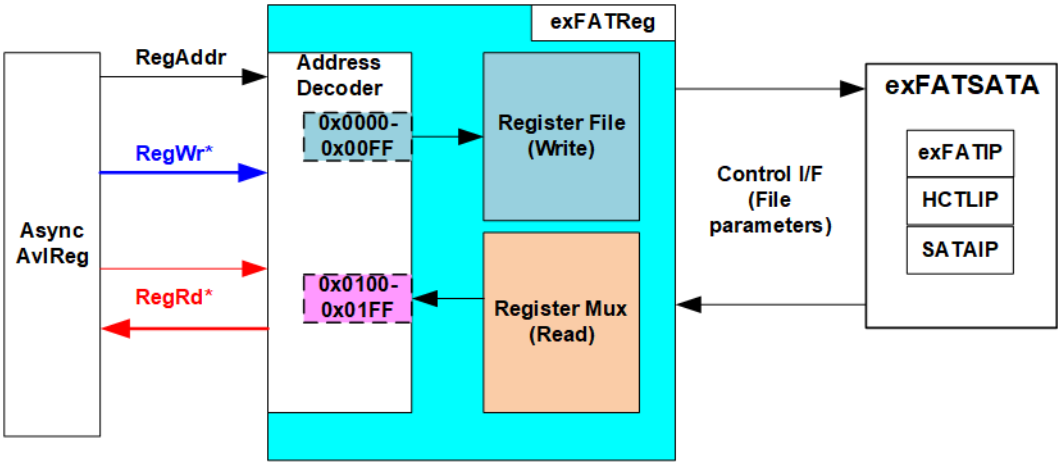

Figure 2-17 exFATReg Block diagram

The address range to map to exFATReg is split into two areas, as shown in Figure 2-17.

1) 0x0000 – 0x00FF: mapped to set file parameters of exFATSATA module. This area is write-access only.
2) 0x0100 – 0x01FF: mapped to read the status of exFATSATA module. This area is read-access only.

Similar to UserReg, Address decoder decodes the upper bit of RegAddr for selecting the active hardware. The latency of read data is equal to two clock cycles, so RegRdValid is created by RegRdReq with asserting two D Flip-flops.

More details of the address mapping within exFATReg module is shown in Table 2-2.

Table 2-2 Memory Map of exFATReg

Address Wr/Rd	Register Name Label in “ftpdemo.c”	Description
BA1+0x0000 – BA1+0x00FF: Control signals of exFAT-IP (Write access only)		
BA1+0x0000	User File Name Reg (USRFNAME_REG)	[26:0] - Input to be UserFName of exFAT-IP for SATA
BA1+0x0004	User File Length Reg (USRFLEN_REG)	[26:0] - Input to be UserFLen of exFAT-IP for SATA
BA1+0x0008	User File Size Reg (USRFSIZE_REG)	[2:0] - Input to be FSize of exFAT-IP for SATA. Set File size to exFAT-IP when running Format command.
BA1+0x000C	Date and 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
BA1+0x0010	exFAT Command Reg (EXFATCMD_REG)	[1:0] - Input to be UserCmd of exFAT-IP for SATA When this register is written, UserReq of exFAT-IP for SATA is asserted to ’1’ to start new command operation.
BA1+0x0100 – BA1+0x01FF: Status signals of exFAT-IP (Read access only)		
BA1+0x0100	exFAT Status Reg (FATSTS_REG)	[0] - Mapped to UserBusy of exFAT-IP for SATA [1] - Mapped to UserError of exFAT-IP for SATA
BA1+0x0104	Total file capacity Reg (TOTALFCAP_REG)	[26:0] - Mapped to TotalFCap[26:0] of exFAT-IP for SATA
BA1+0x0108	User Error Type Reg (FATERRTYPE_REG)	[31:0] - Mapped to UserErrorType[31:0] of exFAT-IP for SATA
BA1+0x010C	exFAT IP Test pin (Low) Reg (FATTESTPINL_REG)	[31:0] - Mapped to TestPin[31:0] of exFAT-IP for SATA
BA1+0x0110	exFAT IP Test pin (High) Reg (FATTESTPINH_REG)	[31:0] - Mapped to TestPin[63:32] of exFAT-IP for SATA
BA1+0x0114	Directory capacity Reg (DIRCAP_REG)	[19:0] - Mapped to DirCap[19:0] of exFAT-IP for SATA
BA1+0x0118	File Size in the disk Reg (DFSIZE_REG)	[2:0] - Mapped to DiskFsize of exFAT-IP for SATA. This register shows the current file size used in the device, read by exFAT-IP for SATA.
BA1+0x011C	Total file in the disk Reg (DFNUM_REG)	[26:0] - Mapped to DiskFnum of exFAT-IP for SATA
BA1+0x0120	Disk Capacity (Low) Reg (DCAPL_REG)	[31:0]: Mapped to LBASize(bit[31:0]) of HCTL-IP to check total capacity of the SATA device.
BA1+0x0124	Disk Capacity (High) reg (DCAPH_REG)	[15:0]: Mapped to LBASize(bit[47:32]) of HCTL-IP to check total capacity of the SATA device.
BA1+0x0128	Current transfer size (Low) Reg (FATCURTRNSIZEL_REG)	[31:0]: Bit[31:0] of the current transfer byte size in exFATSATA module
BA1+0x012C	Current transfer size (High) Reg (FATCURTRNSIZEH_REG)	[24:0]: Bit[56:32] of the current transfer byte size in exFATSATA module
BA1+0x0800	exFAT IP Version Reg (exFATIPVER_REG)	[31:0]: IP version number, mapped to IPVersion [31:0] of exFAT-IP for SATA

3 CPU Firmware

CPU firmware implements two functions for running ftp server demo. First, CPU handles TCP/IP packet of FTP control connection through UserReg and UserMAC module which transmits and receives TCP/IP packet of control connection. Second, CPU controls and monitors the registers of exFATSATA and TOE1G-IP for handling TCP/IP packet of FTP data connection.

To run FTP server demo, CPU firmware operation is shown in Figure 3-1.

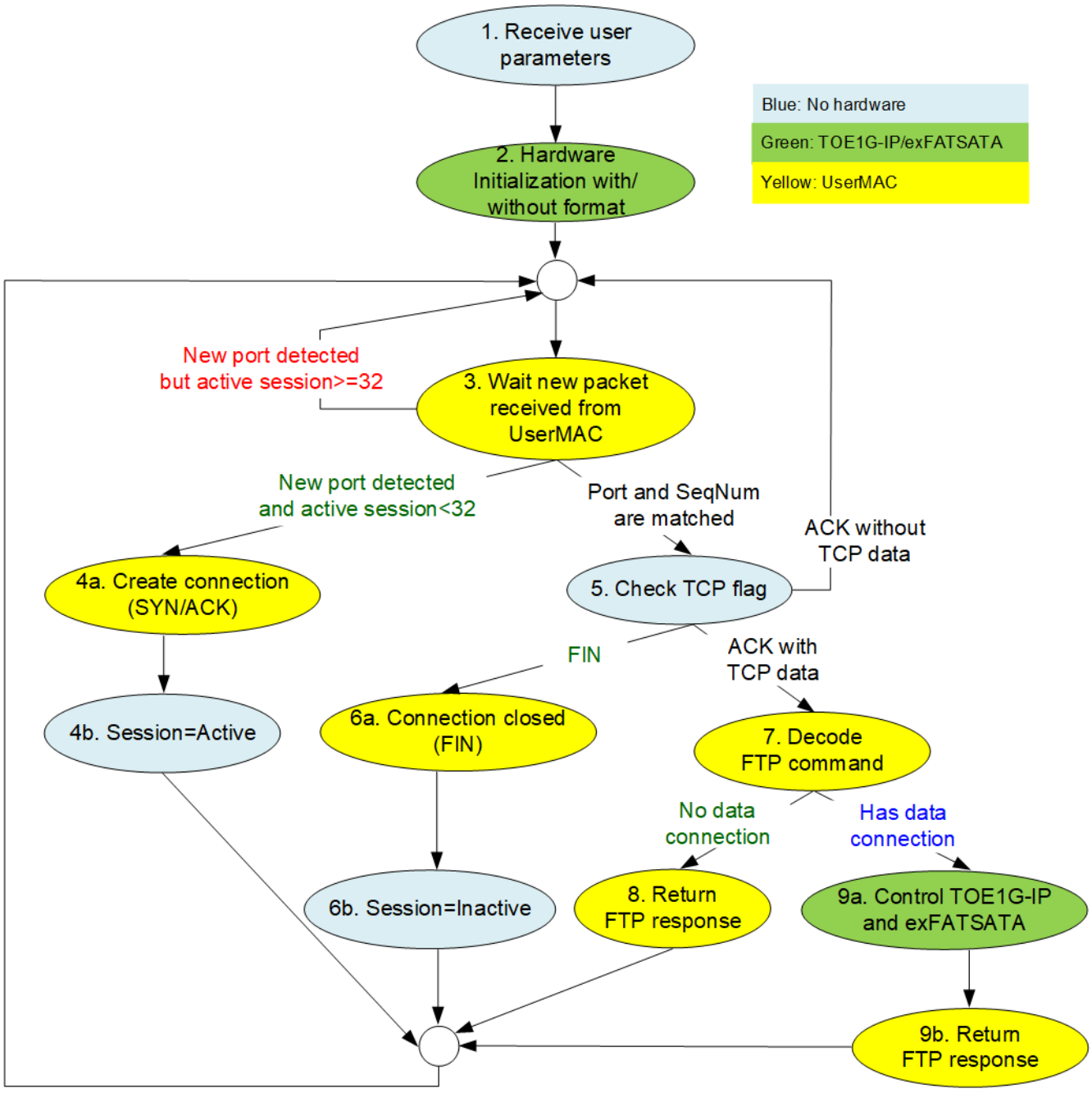

Figure 3-1 Firmware operation overview

1) Receive network parameters for TOE1G-IP and UserRxMAC such as IP address, MAC address, and port number and file parameters for exFATSATA such as created date and time from the user. After that, CPU validates the value.
2) Set the parameters to TOE1G-IP and UserRxMAC. After that, CPU waits until the hardware finishes the initialization phase by monitoring busy flag of TOE1G-IP (TOE_CMD_REG[0]) and exFAT-IP (FATSTS_REG). Next, ask users to format SATA device or skip format process. To run Format command, the user must set file size for transferring in FTP demo.
3) CPU waits until the new packet is received from UserRxMAC. When the new packet is detected, CPU decodes TCP flag to select the next step.
   - When the flag is SYN and total session now is less than 32, CPU goes to step 4.
   - If total session is equal to 32, CPU stays in this step.
   - Otherwise, CPU goes to step 5.
4) a. Create the new connection by returning SYN/ACK packet. After that, the valid ACK without TCP data must be received to complete three-way handshake.
   b. Change the session status from Inactive to Active.
5) CPU finds the active session which the port number is matched and confirms that sequence number is correct value.
   - When the flag is FIN, CPU goes to step 6.
   - When the flag is ACK and TCP data length is not equal to 0, CPU goes to step 7.
   - Otherwise, CPU has no operation and goes back to step 3.
6) a. Transmit FIN/ACK packet to close the connection. After that, the valid ACK without TCP data must be received to complete three-way handshake.
   b. Change the session status from Active to Inactive.
7) CPU decodes FTP command. There are two command types, i.e. command without using data connection and command with using data connection.
   - CPU goes to step 8 for processing command without using data connection.
   - Otherwise, CPU goes to step 9.
8) CPU creates FTP response for each FTP command to TxRAM and then sets the parameters for UserTxMAC transmitting FTP response.
9) a. CPU accesses TOE1G-IP and exFATSATA registers to transfer packet of FTP data connection.
   b. CPU creates FTP response to TxRAM and then sets the parameters for UserTxMAC transmitting FTP response.

More details of CPU firmware in some steps to operate FTP commands are described in the next topic.

3.1 FTP command without data connection

The step when the new FTP command is received and CPU returns response on control connection is as follows.

1) Wait until new received packet is stored in RxRAM by monitoring that data counter of RxMacFIFO (RXEMAC_FFCNT) is not equal to 0.
2) Read data from RxMacFIFO by reading RXEMAC_LASTADDR. After that, CPU copies data from RxRAM to Temp buffer on firmware, starting from the latest position to RXEMAC_LASTADDR value.
   Note: Temp buffer on firmware is defined by BUFFSIZE parameter which is equal to 100. So, the demo without modification can support up to 100-byte packet size which is enough for processing FTP control connection.
3) Update the latest position in the firmware to be the new value (RXEMAC_LASTADDR).
4) Find the active port which the client port value is equal to the source port in the received packet. Continue to the next step when the port is matched and the sequence number in the received packet is equal to the expected value.
   Note: Next step describes the details when TCP flag is ACK without SYN and FIN flag.
   - *When TCP flag in the received packet is SYN, the next process is to create the new session following three-way handshake.*
   - *When TCP flag in the received packet is FIN, the next process is to destroy the session following three-way handshake.*
5) Continue the next step when TCP data length in the received packet is not equal to 0. TCP data length is calculated by (IP length – IP header length – TCP header length).
6) Create FTP response packet to transmitted temp buffer depending on FTP command. Since there are a lot of standard FTP commands, this demo implements some mandatory commands, related to FTP client application, FileZilla. Lists of implemented command without data connection are described in Table 3-1.
7) Call function to prepare the header and calculate checksum for transmitting FTP response.
8) Copy the packet from transmitted temp buffer to TxRAM and set total length to TXEMAC_LEN_REG for starting packet transmission.
9) Wait until the packet transmission is finished by comparing that busy flag of UserTxMAC (TXEMAC_LEN_REG) is equal to 0.
10) Go back to step 1) for processing the next FTP command.

Table 3-1 FTP Response for FTP command without data connection

FTP command	Description	Implemented FTP Response
USER	Authentication username	331 User is correct. Password is required
PASS	Authentication password	230 Logged in
		530 Login is incorrect
PWD	Print working directory	257 "PATHNAME" is the current directory
TYPE	Set the transfer mode	200 Type set to I
PASV	Enter passive mode	227 Enter Passive mode (h1,h2,h3,h4,p1,p2) h1-h4: IP address, p1-p2: Port number
CWD	Change working directory	250 Requested file action Okay
DELE	Delete file	202 No implemented
QUIT	Log out session	221 Bye.

When the command is not in the list such as AUTH, PORT, SYST, and FEAT, FTP response 500 (syntax error) is returned. PASV command is the command sent by client before sending FTP command with data connection, i.e. LIST, RETR, and STOR. More details of FTP command with data connection are described in the next topic.

Note: *DELE command is not implemented in the reference design, so user cannot delete any files from the server's storage, SATA device.*

3.2 FTP command with data connection

Table 3-2 FTP Response for FTP command with data connection

FTP command	Description	Implemented FTP Response
LIST	Return File list in the current directory	125 Open data connection 226 Transferring complete
STOR	Accept the data and store the data as a file at the server site	
RETR	Retrieve a copy of the file	

In the demo, three FTP commands using data connection are implemented as shown in Table 3-2. The details of each FTP command are described as follows.

3.2.1 LIST

This command is applied to return file list in the current directory. List of files is created by CPU firmware, but the packet must be returned in FTP data connection which is controlled by TOE1G-IP. So, CPU writes file list to TOE1G-IP directly when running LIST command. The step to run LIST command is described as follows.

1) Read the number of files in the device from DFNUM_REG.
2) Wait until data connection establishment completes by monitoring ConnOn status (TOEIP_STS_REG[1]='1').
3) Call function to return FTP response 125 to TxRAM and start packet transmission.
4) Set TOE1G-IP parameters to prepare sending file information from CPU, i.e. transfer packet size, total data length, and command for sending data.
5) Prepare file information and write to TOE_TXDATA_REG for sending packet through TOE1G-IP.
6) Wait until finishing data transmission by monitoring busy flag of TOE1G-IP (TOE_CMD_REG[0]='0').
7) Write command register of TOE1G-IP to close connection (TOE_CMD_REG=3).
8) Wait until TOE1G-IP completes releasing connection by monitoring busy flag of TOE1G-IP (TOE_CMD_REG[0]='0').
9) Call function to return FTP response 226 to TxRAM and start packet transmission.

3.2.2 STOR

1) Decode file name received from client which is ASCII code to be unsigned integer.
2) Set exFAT-IP parameters to store data received from FTP client to SATA device, i.e. file name (USRFNAME_REG) and command (EXFATCMD_REG=Write file). The number of files is fixed to 1 to transfer one file data at a time.
3) Wait until data connection establishment completes by monitoring ConnOn status (TOEIP_STS_REG[1]='1').
4) Call function to return FTP response 125 to TxRAM and start packet transmission.
5) Wait until finishing data transmission by monitoring busy flag of exFATSATA (FATSTS_REG='0').
6) Wait until data connection is destroyed by monitoring ConnOn status (TOEIP_STS_REG[1]='0').
7) Call function to return FTP response 226 to TxRAM and start packet transmission.

3.2.3 RETR

1) Decode file name received from client which is ASCII code to be unsigned integer.
2) Set exFAT-IP parameters to return data from SATA device to FTP client, i.e. file name (USRFNAME_REG) and command (EXFATCMD_REG=Read file). The number of files is fixed to 1 to transfer one file data at a time.
3) Call function to return FTP response 125 to TxRAM and start packet transmission.
4) Set TOE1G-IP parameter to send data from SATA device, i.e. transfer packet size, total data length, and command for sending data.
5) Wait until finishing data transmission by monitoring busy flag of TOE1G-IP (TOE_CMD_REG[0]='0'). The maximum file size of SATA device is 512 GB while the maximum transfer size of TOE1G-IP is 4 GB – 1. When file size is more than or equal to 4 GB, step 4) and step 5) must be repeated in many times until total size of TOE1G-IP is equal to total size of exFATSATA.
6) Follow step 7) – step 9) of LIST command to close the connection and return FTP response.

3.3 Function List in Test Firmware

The function in the firmware is split into three groups, i.e. exFAT-IP function, TOE1G-IP function, and FTP function. The description of each function is shown as follows.

3.3.1 Function for exFAT-IP

Function in this topic is applied to control exFAT-IP.

void change_ftime(void)	
Parameters	None
Return value	None
Description	Print current create 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 are updated to DATETIME_REG and global parameter (DateTime).

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

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), and total file in the disk (DFnum).

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

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_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_faterror(void)	
Parameters	None
Return value	None
Description	Read FATERRTYPE_REG and decode the value. Print error type when the flag is found. The example of error type is timeout error, SATA-IP error, and unsupported disk capacity.

void update_dfsize(void)	
Parameters	None
Return value	None
Description	Read total file (TOTALFCAP_REG) and current file size in the disk (DFSIZE_REG) 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).

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

3.3.2 Function for TOE1G-IP

Function in this topic is applied to control TOE1G-IP.

void exec_port(unsigned int port_ctl, unsigned int mode_active)	
Parameters	port_ctl: 1-Open port, 0-Close port mode_active: 1-Active open/close, 0-Passive open/close
Return value	None
Description	For active mode, write TOE_CMD_REG to open or close connection. After that, call read_conon function to monitor connection status until it changes from ON to OFF or OFF to ON, depending on port_ctl mode.

void init_toe_param(void)	
Parameters	None
Return value	None
Description	1) Set network parameters to TOE1G-IP register from global parameters. After reset is de-asserted, it waits until TOE1G-IP busy flag is de-asserted to '0'. 2) Set the header value and header enable for packet filtering in UserRxMAC. 3) Set default value to FTP response parameter.

int input_param(void)	
Parameters	None
Return value	0: Valid input, -1: Invalid input
Description	Receive network parameters from user, i.e. mode, window threshold, FPGA MAC address, FPGA IP address, FPGA port number, Target IP address, and Target port number. When the input is valid, the parameters are updated. Otherwise, the value does not change. After receiving all parameters, the current value of each parameter is displayed.

unsigned int read_conon(void)	
Parameters	None
Return value	0: Connection is OFF, 1: Connection is ON.
Description	Read value from USER_CMD_CONNON register and return only bit2 value to show connection status.

void show_param(void)	
Parameters	None
Return value	None
Description	Print the current value of the network parameters setting to TOE1G-IP such as IP address, MAC address, and port number.

3.3.3 Function for ftp operation

unsigned int cal_checksum(unsigned int byte_len, unsigned char *buf)	
Parameters	bytel_len: number of bytes required to calculate checksum buf: the packet to calculate checksum
Return value	Checksum value
Description	Calculate 16-bit checksum of the data in the buffer

unsigned int decode_fname(void)	
Parameters	None
Return value	The value output from file name
Description	Convert filename from received packet which is ASCII code to the value

void read_rxpacket(void)	
Parameters	None
Return value	None
Description	Wait new received packet from UserRxMAC. When new packet is received, the packet is copied to received temp buffer for processing.

void send_list(void)	
Parameters	None
Return value	None
Description	Return file list with the information such as modified date and time for LIST command as described in topic 3.2.1.

void send_resp(unsigned int flag, unsigned int ftp_response)	
Parameters	flag: TCP flag in the received packet ftp_response: FTP response message
Return value	None
Description	Send FTP response by UserTxMAC

void set_tx_csum(void)	
Parameters	None
Return value	None
Description	Assign checksum to Tx header

void set_tx_header(unsigned int flag, unsigned int data_len)	
Parameters	flag: TCP Flag for transmitted packet data_len: Total transmitted length in byte unit
Return value	None
Description	Set packet header of the transmitted packet including Ethernet header, IP header, and TCP header

void take_ftp_resp(void)	
Parameters	None
Return value	None
Description	Operate FTP command in Table 3-1 and Table 3-2 by creating FTP response and control data connection as described in topic 3.2.

4 Revision History

Revision	Date	Description
1.0	7-Feb-20	Initial version release