UDP10G-IP reference design manual

Rev1.4 9-Mar-23

1 Introduction

Compared to the TCP protocol, the UDP protocol minimizes protocol mechanisms when sending data. There is no handshake and no data recovery process to confirm that the receiver accepts all data correctly. However, like TCP, UDP provides checksums for data integrity and port numbers for addressing different functions at the source and destination in networks.

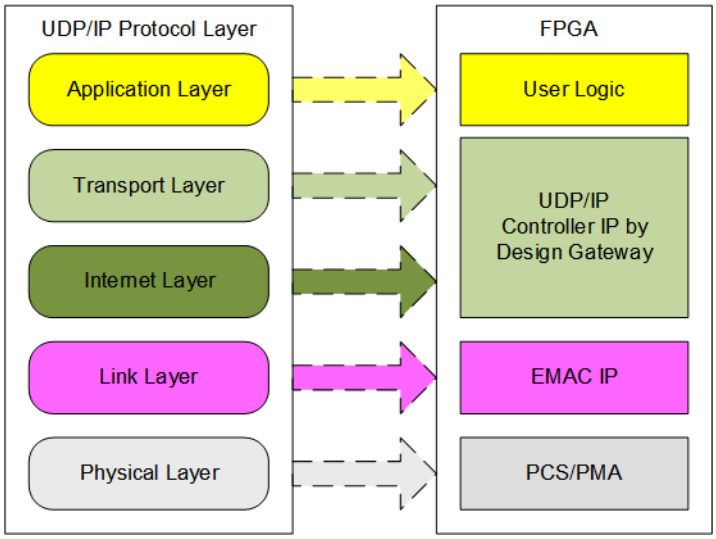

Figure 1-1 UDP/IP Protocol Layer

UDP10G-IP implements the Transport and Internet layers of the UDP/IP Protocol to build Ethernet packets from user data (UDP payload data) to EMAC. If the UDP payload data size is larger than a packet size, UDP10G-IP splits the user data into smaller size to fit in one packet. After that, the payload data is appended by UDP/IP header. Conversely, when the EMAC receives the Ethernet packet, UDP10G-IP extracts the data from the packet and verifies the header. If the header is valid, UDP payload data is forwarded to the user logic. Otherwise, the packet is rejected.

The lower layer protocols are implemented by EMAC-IP and PCS/PMA-IP, where PCS/PMA-IP is provided by Xilinx FPGA while EMAC-IP can be implemented by DG 10G25GEMAC-IP or Xilinx EMAC-IP.

The reference design includes a simple user logic to transfer data using UDP10G-IP. UDP10G-IP transfers data with a PC or another UDP10G-IP on another FPGA board. To transfer data with a PC, the test application, udpdatatest, is called on the PC to send and verify UDP payload data via Ethernet connection at a high-speed rate. One application is called for transferring data in one direction, while two test applications are called for full-duplex test to send and receive data simultaneously.

To allow the user to control the test parameters and the operation of the UDP10G-IP demo via UART, the CPU system is included. The user can easily set the test parameters and monitor the current status on the UART console. The firmware on CPU is built using bare-metal OS. More details of the demo are described below.

2 Hardware overview

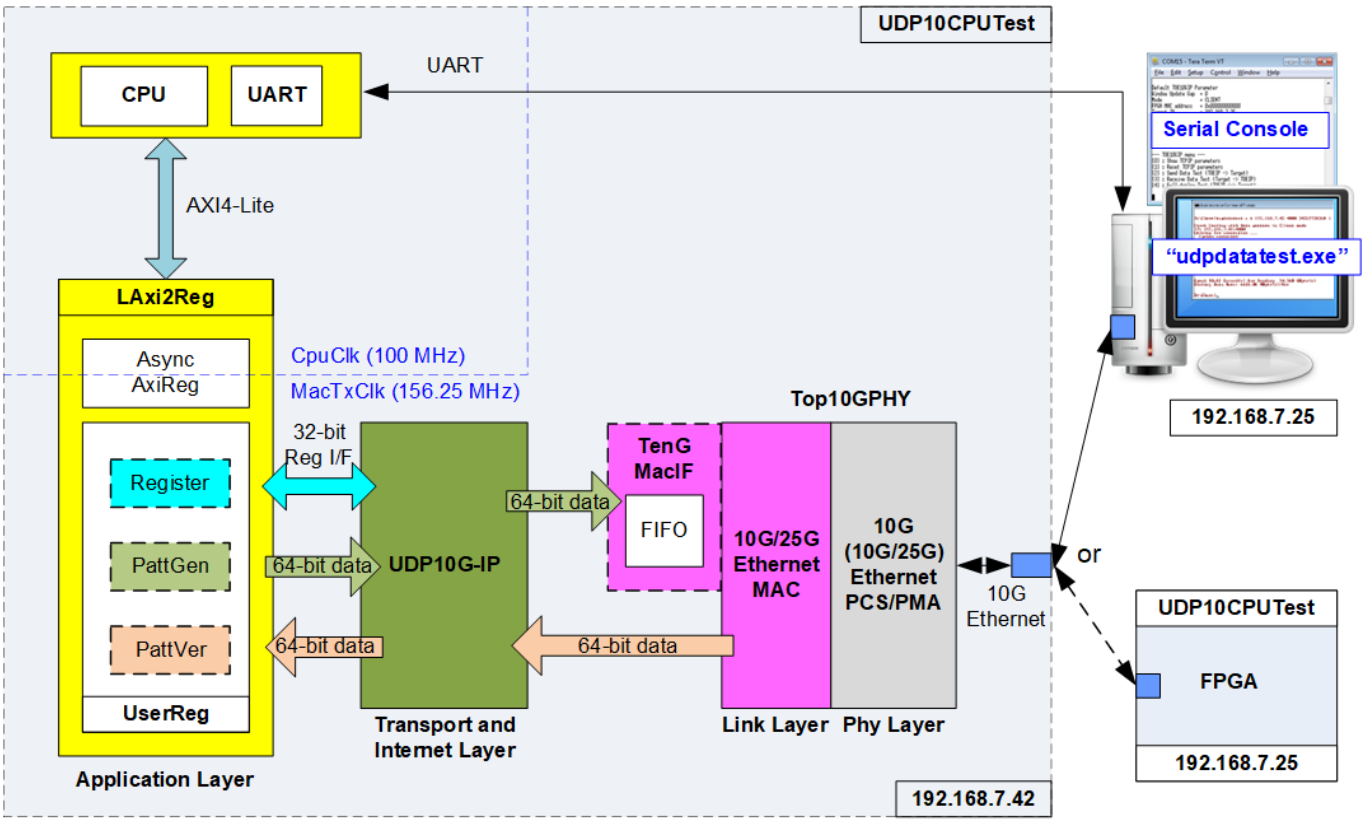

Figure 2-1 Demo Block Diagram

During testing, two devices are used to transfer data over a 10G Ethernet connection. The first device operates in Client mode and the second device operates in Server mode. In the demonstration, the Client device is the UDP10G-IP on an FPGA board, while the Server device can be either the UDP10G-IP on an FPGA board or a PC, as shown in Figure 2-1. If a PC is used, the test application "udpdatatest" must be executed on the PC to transfer data with the UDP10G-IP within the FPGA.

In the FPGA system, the UDP10G-IP is connected to the 10G/25G Ethernet MAC and 10G/25G Ethernet PCS/PMA to implement all the UDP/IP layers. The user interface of the UDP10G-IP is connected to UserReg within AsyncAxiReg, which includes a Register file for interfacing with the Register interface, PattGen for sending test data via Tx FIFO interface, and PattVer for verifying test data via Rx FIFO interface. The Register file of UserReg is controlled by CPU firmware through the AXI4-Lite bus.

Additionally, TenGMacIF serves as a data buffer between the UDP10G-IP and the Xilinx 10G EMAC-IP when the Xilinx 10G EMAC-IP is not ready to receive the packet during a packet transmission. If the DG 10G25GEMAC-IP is used, the UDP10G-IP can connect to EMAC directly without adding TenGMacIF.

The test design uses two clock domains, CpuClk, which is the independent clock for running the CPU system, and MacTxClk, which is the clock output from 10G/25G Ethernet PCS/PMA. Therefore, AsyncAxiReg is designed to support asynchronous signals between CpuClk and MacTxClk. More information about each module inside the UDP10CPUTest is described below.

2.1 10G/25G Ethernet PCS/PMA (10G BASE-R)

The physical layer of 10G Ethernet is implemented by 10G/25G PCS/PMA, which connects to an external 10G BASE-R SFP+ module. The user interface is 64-bit XGMII interface running at 156.25 MHz and is designed to connect with 10G Ethernet MAC. This IP core is provided by Xilinx free of charge. Further details about this core can be found at the following links.

10G (10G/25G) Ethernet PCS/PMA (BASE-R)
https://www.xilinx.com/products/intellectual-property/10gbase-r.html
https://www.xilinx.com/products/intellectual-property/ef-di-25gemac.html

2.2 10G/25G Ethernet MAC

The 10G/25G EMAC implements Ethernet MAC layer of 10G Ethernet and acts as an interface between UDP10G-IP and the 10G PCS/PMA module. The interface of UDP10G-IP's EMAC interface is a 64-bit AXI4 stream, while the user interface of 10G PCS/PMA module is a 64-bit XGMII interface running at 156.25 MHz.

When using DG 10G25GEMAC-IP, UDP10G-IP can directly connect to EMAC-IP, as shown in Figure 2-2. More details about DG 10G25GEMAC-IP can be found in the following link.
https://dgway.com/products/IP/10GEMAC-IP/dg_10g25gemacip_data_sheet_xilinx_en.pdf

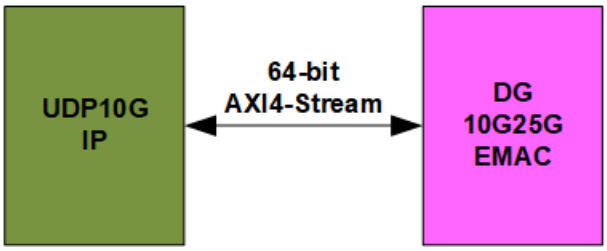

Figure 2-2 UDP10G-IP connecting with DG 10G25GEMAC IP

When connecting with Xilinx 10G/25G EMAC-IP, a small buffer must be used to interface between UDP10G-IP and Xilinx 10G/25G EMAC-IP. The TenGMacIF module acts as this interface. More information about the TenGMacIF can be found in the next topic. Further details about Xilinx EMAC can be found at the following links.

Xilinx 10G EMAC-IP
https://www.xilinx.com/products/intellectual-property/do-di-10gemac.html
Xilinx Ethernet 10G/25G PCS/PMA
https://www.xilinx.com/products/intellectual-property/ef-di-25gemac.html

2.3 TenGMacIF (only when using Xilinx EMAC IP)

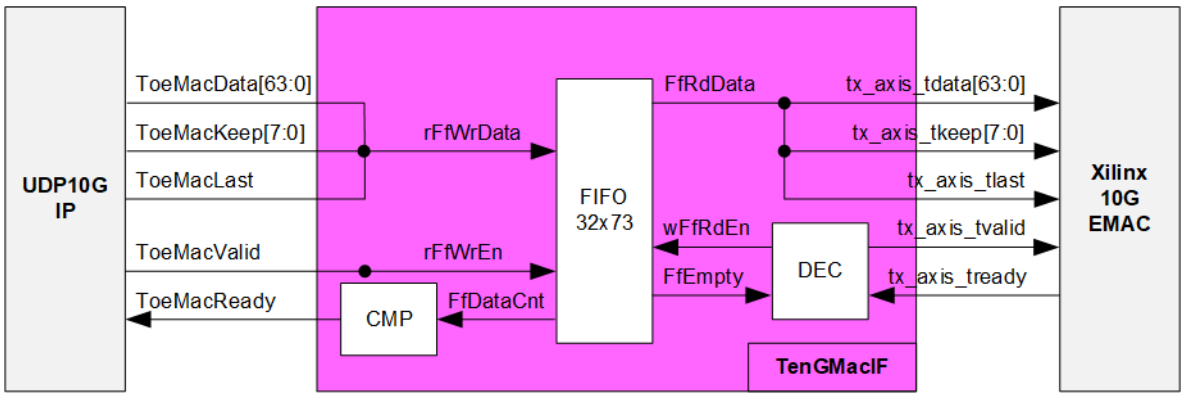

Figure 2-3 TenGMacIF Block Diagram

The user interface of the Xilinx 10G/25G EMAC, which is 64-bit AXI4 stream, is not compliant with the EMAC interface of the UDP10G-IP for transmitting a packet. The Xilinx IP may pause data transmission by de-asserting tx_axis_tready to 1b while transferring a packet, but UDP10G-IP requires continuous data transfer for each packet. Therefore, TenGMacIF is designed to buffer the data when the Xilinx IP pauses data transmission. Using the real-board test environment, the ready signal of Xilinx IP is de-asserted for a few clock cycles. Therefore, only a small FIFO with 32-data depth is applied in TenGMacIF to store the transmitted data from UDP10G-IP while EMAC pausing transmission.

The logic of TenGMacIF is split into two groups: Writing FIFO and Readinf FIFO. The timing diagrams for each group are displayed in Figure 2-4 and Figure 2-5, respectively.

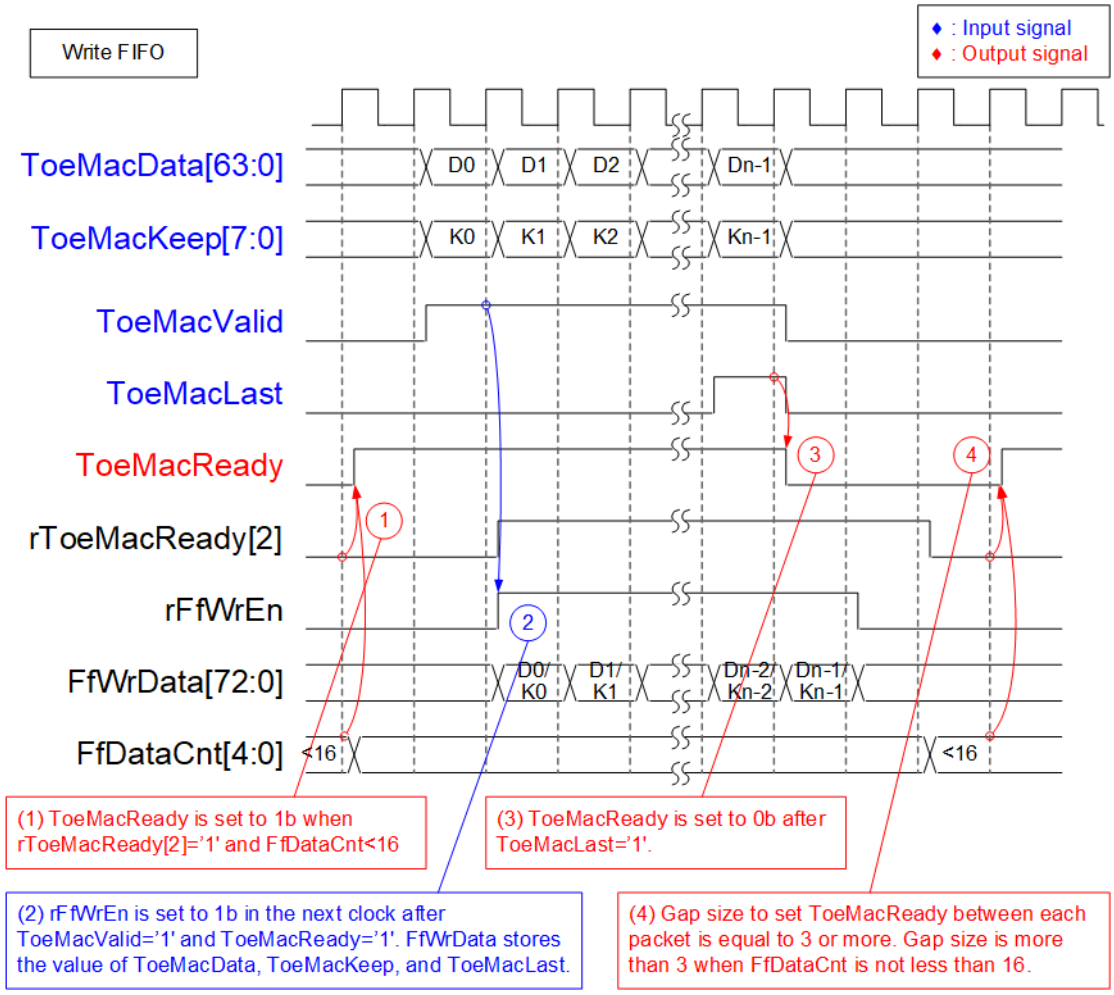

Figure 2-4 Timing diagram for transferring data from UDP10G-IP to FIFO

1) Before asserting ToeMacReady to 1b for receiving a new packet from user, two conditions must be satisfied. Firstly, The FIFO must have sufficient free space to store the next packet. A simple monitoring logic can be implemented by checking the upper bit of FfDataCnt to ensure that the amount of data in the FIFO is below 16. Secondly, rToeMacReady[2] must be set to 0b to indicate that the previous packet has been fully transferred for at least two clock cycles.
   Note: It is found that the Xilinx 10G/25G EMAC-IP does not de-assert tx_axis_tready to 0b for long time. It is enough to support to pause data transmission for 16 clock cycles per packet.
2) The user initiates packet transmission by setting ToeMacValid to 1b. The input signals from user (ToeMacData, ToeMacKeep, and ToeMacLast) are valid and stored in FIFO once ToeMacValid and ToeMacReady are both set to 1b. Subsequently, the inputs are written to the FIFO by setting rFfWrEn to 1b. The 73-bit write data to FIFO comprises of 64-bit data (ToeMacData), 8-bit empty byte (ToeMacKeep), and end flag (ToeMacLast).
3) After receiving the final data of a packet (ToeMacLast=1b and ToeMacValid=1b), ToeMacReady is de-asserted to 0b to pause data transmission for 3 clock cycles to create the gap size of each packet transmission.
4) ToeMacReady can be re-asserted to 1b if the FfDataCnt < 16.

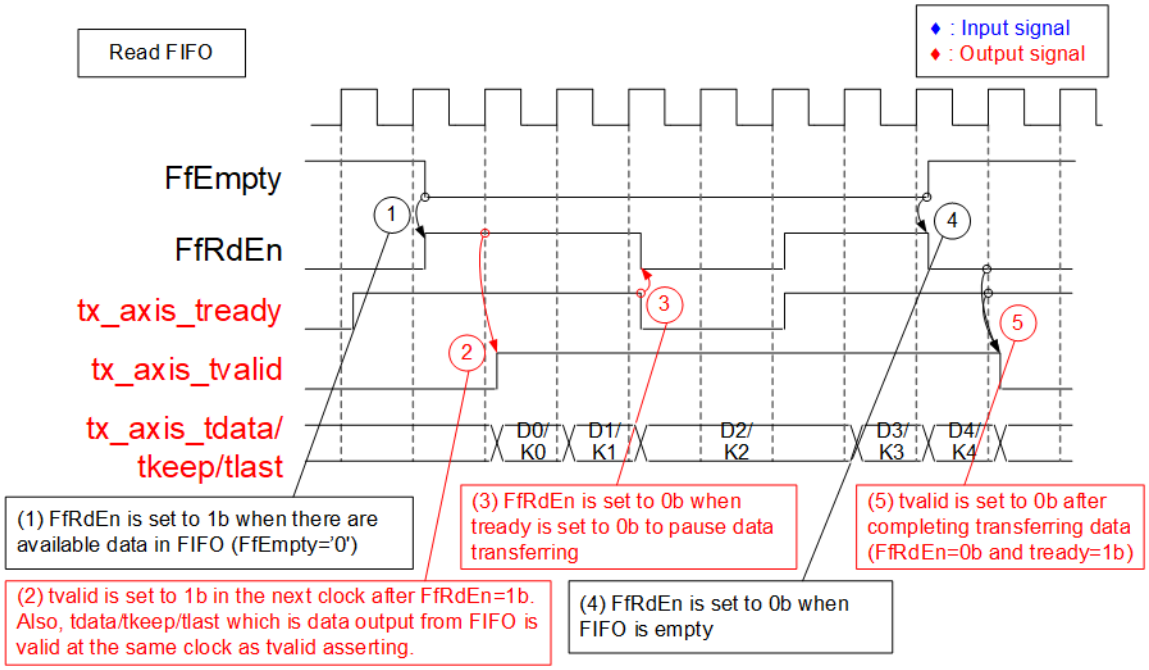

Figure 2-5 Timing diagram for transferring data from FIFO to EMAC

1) Data transmission of a new packet begins once the FIFO has stored some data (FfEmpty=0b). UDP10G-IP always sends data of each packet continuously, so the data in FIFO is ready for reading until the end of the packet. The read enable of FIFO (FfRdEn) is set to 1b to read all data from the FIFO and forward it to the Ethernet MAC.
2) In the next clock cycle after the data is read, the data becomes valid for transferring to EMAC. To transmit the data to EMAC, tx_axis_tvalid is set to 1b.
3) If EMAC is not ready to receive data (tx_axis_tready=0b), FfRdEn must be de-asserted to 0b to pause data transmission.
4) Once a packet has been completely transferred, the FIFO becomes empty (FfEmpty=1b) and FfRdEn is then de-asserted to 0b to stop data transmission.
5) Finally, to finish the packet transmission, tx_axis_tvalid is de-asserted to 0b.

2.4 UDP10G-IP

UDP10G-IP implements UDP/IP stack and fully offload engine without requiring the CPU and the external memory. User interface has two signal groups - control signals and data signals. Control and status signals use Single-port RAM interface for write/read register access. Data signals use FIFO interface for transferring data stream in both directions. More information can be found from the datasheet.
https://dgway.com/products/IP/UDP10G-IP/dg_udp10gip_data_sheet_xilinx_en.pdf

2.5 CPU and Peripherals

The 32-bit AXI4-Lite is used for CPU access to peripherals such as Timer and UART in the test system. Control and status signals are connected to be registers for CPU access as a peripheral through the 32-bit AXI4-Lite bus. The CPU assigns a different base address and address range to each peripheral, allowing access to one peripheral at a time.

In the reference design, the test hardware is connected to the CPU system as a peripheral with a specified base address and range. Therefore, the LAxi2Reg module that interfaces with the CPU must support the AXI4-Lite bus standard for CPU writing and reading, as shown in Figure 2-6.

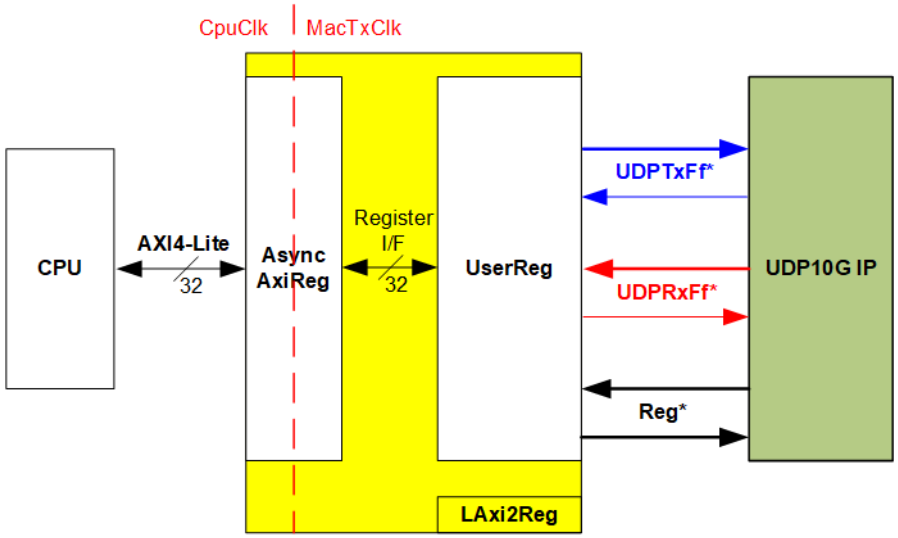

Figure 2-6 LAxi2Reg block diagram

The LAxi2Reg module includes two parts: AsyncAxiReg and UserReg. AsyncAxiReg is designed to convert the AXI4-Lite signals to a simple Register interface with a 32-bit data bus size, similar to AXI4-Lite data bus size. In addition, AsyncAxiReg includes asynchronous logic to support clock domain crossing between the CpuClk domain and MacTxClk domain.

UserReg includes the Register file for the parameters and the status signals of the test logics. Both the data interface and control interface of UDP10G-IP are connected to UserReg. Further details of AsyncAxiReg and UserReg are described below.

2.5.1 AsyncAxiReg

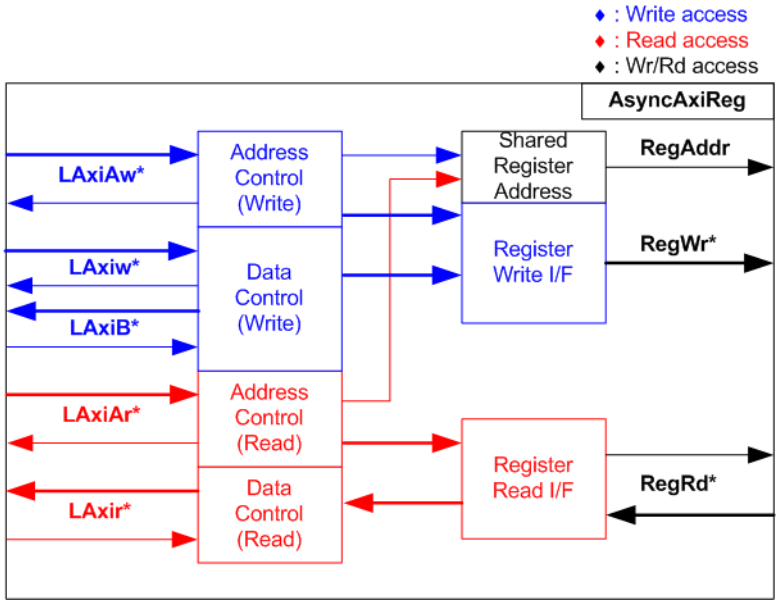

Figure 2-7 AsyncAxiReg Interface

The AXI4-Lite bus interface signals are categorized into five groups: LAxiAw* (Write address channel), LAxiw* (Write data channel), LAxiB* (Write response channel), LAxiAr* (Read address channel), and LAxir* (Read data channel). More information on creating custom logic for the AXI4-Lite bus can be found in the following document.
https://github.com/Architech-Silica/Designing-a-Custom-AXI-Slave-Peripheral/blob/master/designing_a_custom_axi_slave_rev1.pdf

According to the AXI4-Lite standard, the write channel and read channels operate independently for both control and data interfaces. Therefore, the logic in the AsyncAxiReg module that interfaces with the AXI4-Lite bus is divided into four groups: Write control logic, Write data logic, Read control logic, and Read data logic, as shown on the left side of Figure 2-7. The Write control I/F and Write data I/F of the AXI4-Lite bus are latched and transferred to become the Write register interface with clock domain crossing registers. Similarly, the Read control I/F of the AXI4-Lite bus is latched and transferred to the Read register interface, while Read data is returned from the Register interface to the AXI4-Lite bus via clock domain crossing registers. In the Register interface, RegAddr is a shared signal for write and read access, loading the value from LAxiAw for write access or LAxiAr for read access.

The Register interface is compatible with a single-port RAM interface for write transaction. However, the read transaction of the Register interface has been slightly modified from the RAM interface by adding the RdReq and RdValid signals to control read latency time. Since the address of the Register interface is shared for both write and read transactions, the user cannot write and read the register simultaneously. The timing diagram of the Register interface is shown in Figure 2-8.

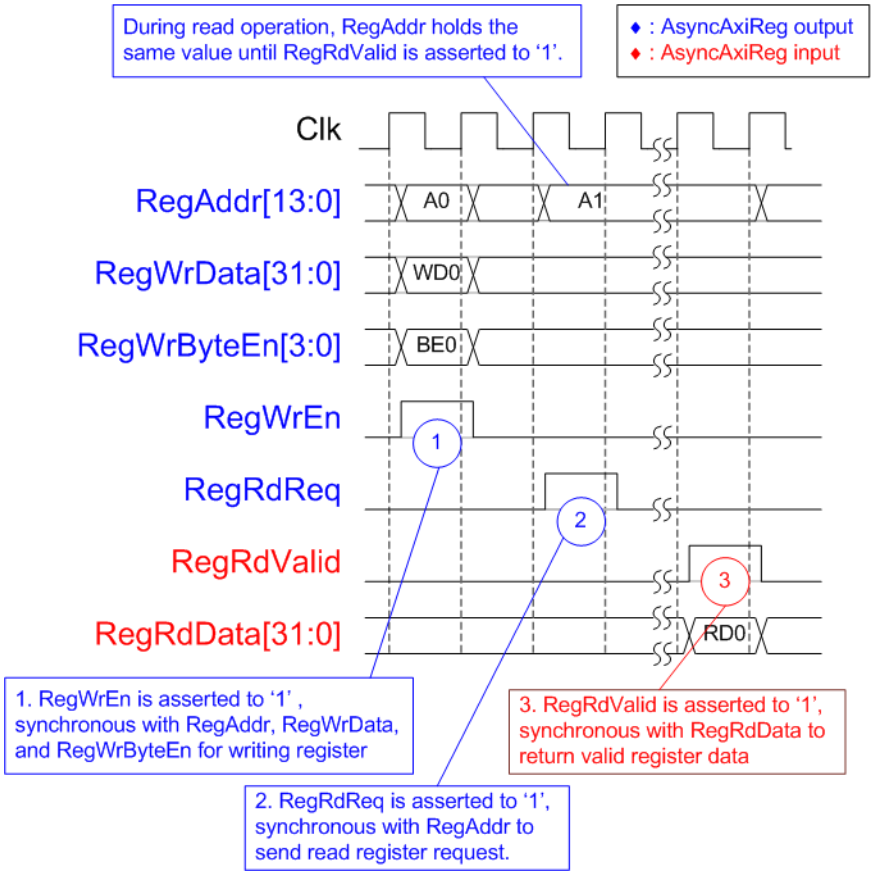

Figure 2-8 Register interface timing diagram

1) Timing diagram to write register is similar to that of a single-port RAM. The RegWrEn signal is set to 1b, along with a valid RegAddr (Register address in 32-bit units), RegWrData (write data for the register), and RegWrByteEn (write byte enable). The byte enable consists of four bits that indicate the validity of the byte data. For example, bit[0], [1], [2], and [3] are set to 1b when RegWrData[7:0], [15:8], [23:16], and [31:24] are valid, respectively.
2) To read register, AsyncAxiReg sets the RegRdReq signal to 1b with a valid value for RegAddr. The 32-bit data is returned after the read request is received. The slave detects the RegRdReq signal being set to start the read transaction. In the read operation, the address value (RegAddr) remains unchanged until RegRdValid is set to 1b. The address can then be used to select the returned data using multiple layers of multiplexers.
3) The slave returns the read data on RegRdData bus by setting the RegRdValid signal to 1b. After that, AsyncAxiReg forwards the read value to the LAxir* interface.

2.5.2 UserReg

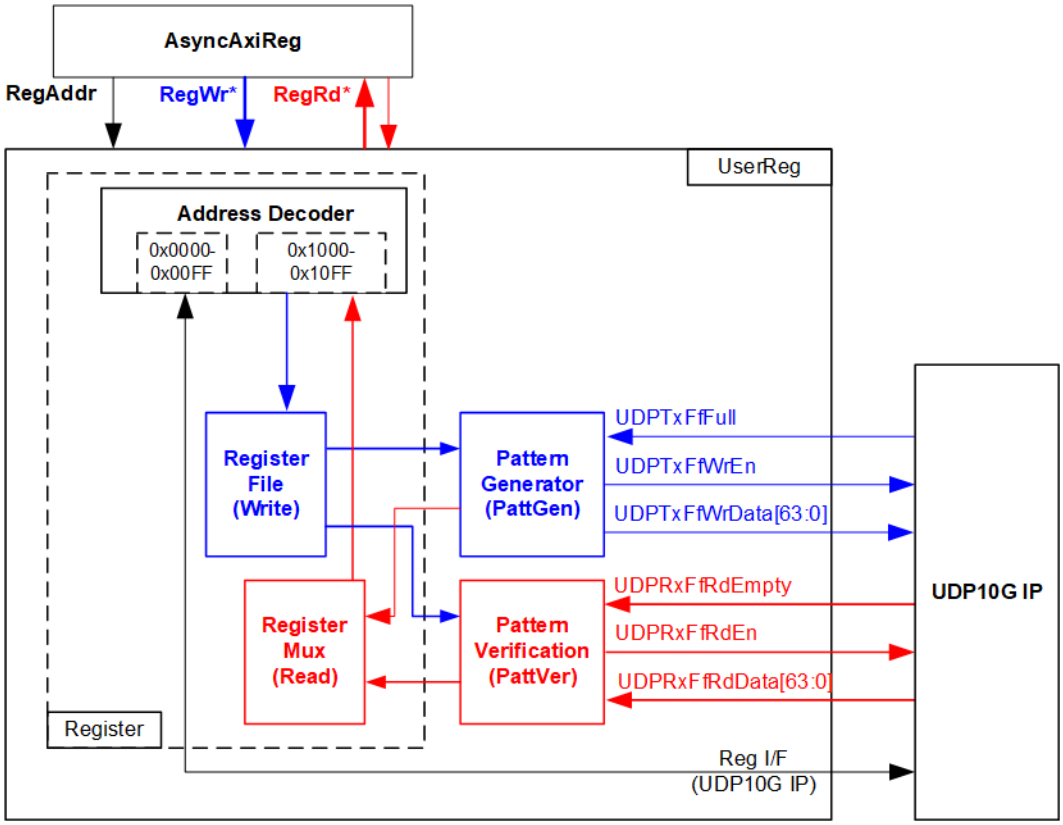

Figure 2-9 UserReg block diagram

The UserReg module includes three functions: Register, Pattern generator (PattGen), and Pattern verification (PattVer). The Register block decodes the requested address from AsyncAxiReg and selects the active register for a write or read transaction. The PattGen block is designed to send 64-bit test data to UDP10G-IP following FIFO interface standard, while the PattVer block is designed to read and verify 64-bit data from UDP10G-IP following FIFO interface standard.

Register Block

The address range is split into two areas: UDP10G-IP register (0x0000-0x00FF) and UserReg register (0x1000-0x10FF). The Address decoder decodes the upper bits of RegAddr to select the active hardware. Since the Register file inside UserReg is 32-bit bus size, Write byte enable (RegWrByteEn) is not used. To write hardware registers, the CPU must use a 32-bit pointer to place a 32-bit valid value on the write data bus.

For reading a register, a multiplexer selects the data to return to CPU by using the address. The lower bits of RegAddr are applied to select the active data within each Register area. While the upper bits are used to select the returned data from each Register area. The total latency time of read data is equal to one clock cycle, and RegRdValid is created by RegRdReq by asserting a D Flip-flop. More details of the address mapping within the UserReg module are shown in Table 2-1.

Table 2-1 Register map Definition

Address Wr/Rd	Register Name (Label in the “udp10gtest.c”)	Description
BA+0x0000 – BA+0x00FF: UDP10G-IP Register Area More details of each register are described in Table2 of UDP10G-IP datasheet.		
BA+0x0000	UDP_RST_INTREG	Mapped to RST register within UDP10G-IP
BA+0x0004	UDP_CMD_INTREG	Mapped to CMD register within UDP10G-IP
BA+0x0008	UDP_SML_INTREG	Mapped to SML register within UDP10G-IP
BA+0x000C	UDP_SMH_INTREG	Mapped to SMH register within UDP10G-IP
BA+0x0010	UDP_DIP_INTREG	Mapped to DIP register within UDP10G-IP
BA+0x0014	UDP_SIP_INTREG	Mapped to SIP register within UDP10G-IP
BA+0x0018	UDP_DPN_INTREG	Mapped to DPN register within UDP10G-IP
BA+0x001C	UDP_SPN_INTREG	Mapped to SPN register within UDP10G-IP
BA+0x0020	UDP_TDL_INTREG	Mapped to TDL register within UDP10G-IP
BA+0x0024	UDP_TMO_INTREG	Mapped to TMO register within UDP10G-IP
BA+0x0028	UDP_PKL_INTREG	Mapped to PKL register within UDP10G-IP
BA+0x0038	UDP_SRV_INTREG	Mapped to SRV register within UDP10G-IP
BA+0x003C	UDP_VER_INTREG	Mapped to VER register within UDP10G-IP
BA+0x1000 – BA+0x10FF: UserReg control/status		
BA+0x1000 Wr/Rd	Total transmit length (USER_TXLEN_INTREG)	Wr [31:0] – Total transmit size in QWord unit (64-bit). Valid from 1-0xFFFFFFFF. Rd [31:0] – Current transmit size in QWord unit (64-bit). The value is cleared to 0 when USER_CMD_INTREG is written by user.
BA+0x1004 Wr/Rd	User Command (USER_CMD_INTREG)	Wr [0] – Start Transmitting. Set 0b to start transmitting. [1] – Data Verification enable (0b: Disable data verification, 1b: Enable data verification) Rd [0] – PattGen Busy. (0b: Idle, 1b: PattGen is busy) [1] – Data verification error (0b: Normal, 1b: Error) This bit is auto-cleared when user starts new operation or reset.
BA+0x1008 Wr/Rd	User Reset (USER_RST_INTREG)	Wr [0] – Reset signal. Set 1b to reset the logic. This bit is auto-cleared to 0b. [8] – Set 1b to clear read value of USER_RST_INTREG[8] to 0b Rd [8] – Latched value of IntOut, the output from IP (0b: Normal, 1b: IntOut has been asserted) This flag can be cleared by system reset condition or setting USER_RST_INTREG[8]=1b. [16] – Ethernet Linkup status (0b: Link down, 1b: Link up)
BA+0x100C Rd	FIFO status (USER_FFSTS_INTREG)	Rd [2:0]: Mapped to UDPRxFfLastRdCnt signal of UDP10G-IP [15:3]: Mapped to UDPRxFfRdCnt signal of UDP10G-IP [24]: Mapped to UDPTxFfFull signal of UDP10G-IP
BA+0x1010 Rd	Total receive length (USER_RXLEN_INTREG)	Rd [31:0] – Current receive size in QWord unit (64-bit). The value is cleared to 0 when USER_CMD_INTREG is written by user.
BA+0x1080 Rd	EMAC IP version (EMAC_VER_INTREG)	Rd[31:0] – Mapped to IPVersion output from DG 10G25GEMAC-IP when the system integrates DG 10G25GEMAC-IP.

Pattern Generator

The logic diagram and timing diagram of Pattern Generator (PattGen) are illustrated in Figure 2-10 and Figure 2-11, respectively.

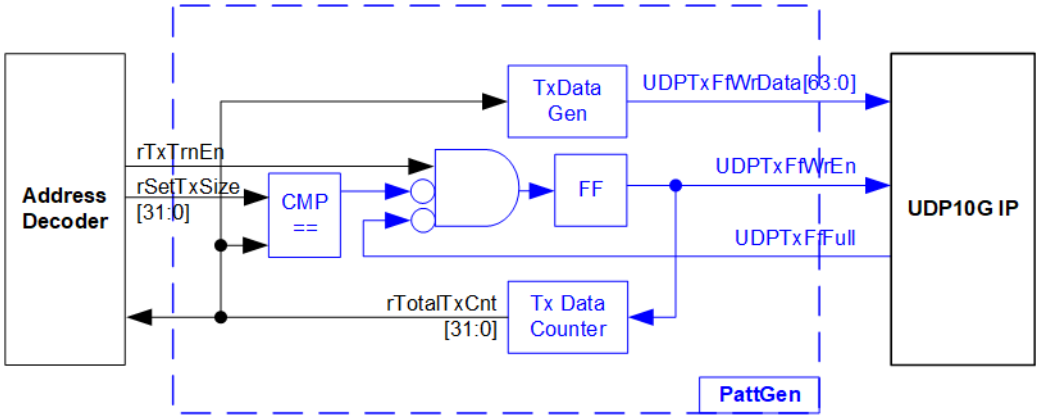

Figure 2-10 PattGen block

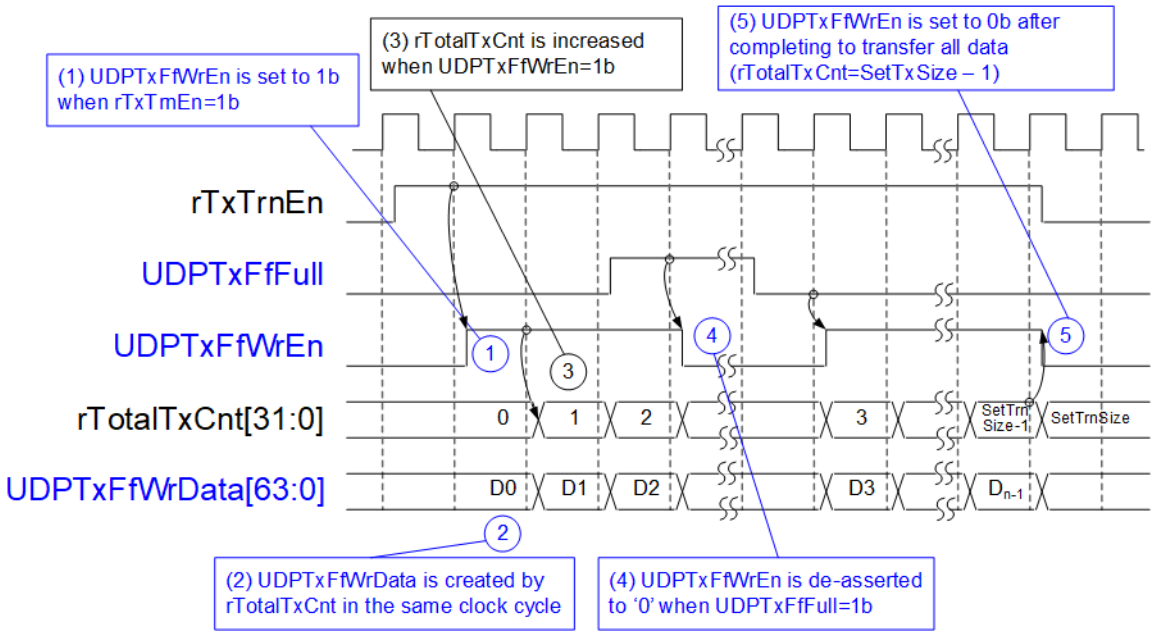

Figure 2-11 PattGen Timing diagram

When USER_CMD_INTREG[0] is set to 0b, PattGen initiates the operation of generating test data by setting rTxTrnEn to 1b. While rTxTrnEn remains set to 1b, UDPTxFfWrEn is controlled by UDPTxFfFull. If UDPTxFfFull is 1b, UDPTxFfWrEn is de-asserted to 0b. The data counter, rTotalTxCnt, checks the total amount of data sent to UDP10G-IP. The lower bits of rTotalTxCnt generate 32-bit incremental data for the UDPTxFfWrData signal. Once all data has been transferred, equal to rSetTxSize, rTxTrnEn is de-asserted to 0b.

Pattern Verification

The logic diagram and timing diagram of Pattern Verification (PattVer) are illustrated in Figure 2-12 and Figure 2-13, respectively. The verification feature is executed when the verification flag (rVerifyEn) is enabled.

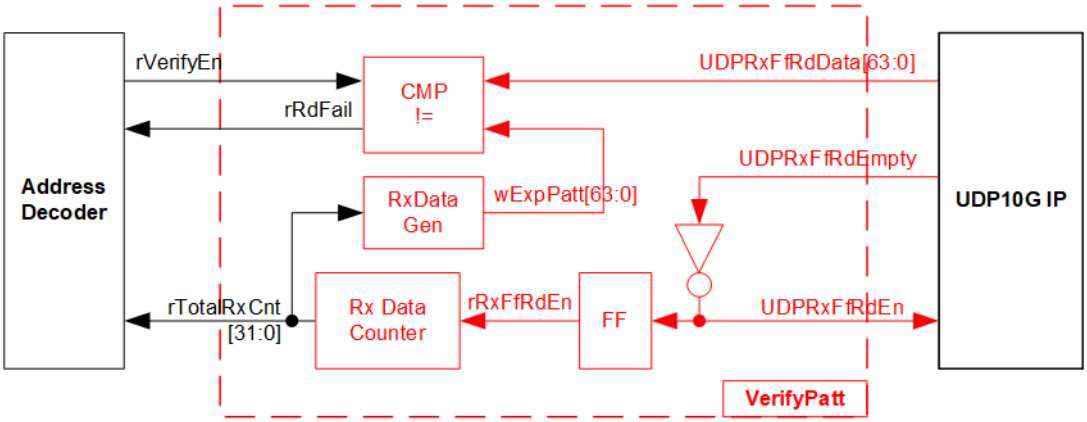

Figure 2-12 PattVer block

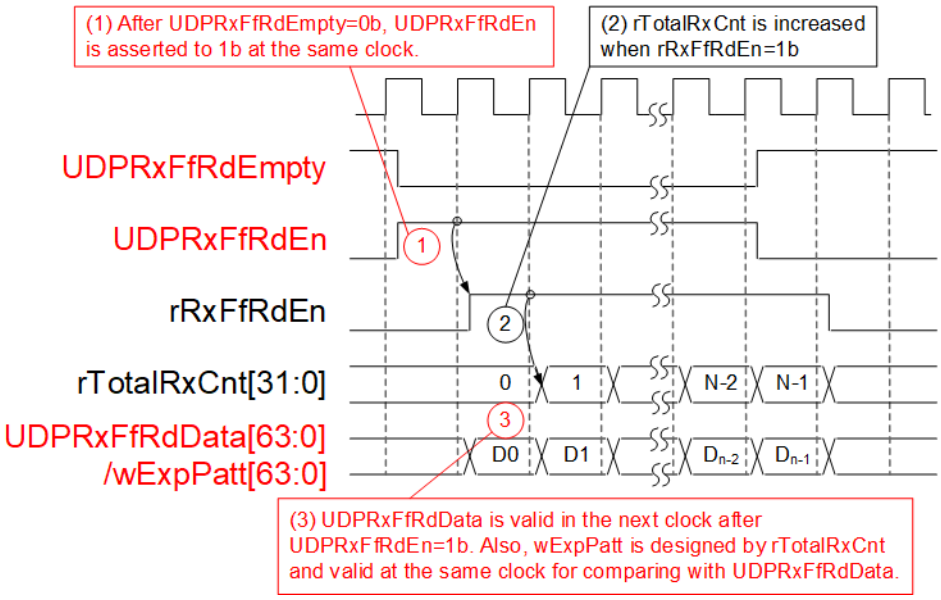

Figure 2-13 PattVer Timing diagram

When rVerifyEn is set to 1b, the verification logic is processed. It compares the received data (UDPRxFfRdData) with the expected data (wExpPatt). If comparison fails, rRdFail is asserted to 1b. The UDPRxFfRdEn signal is created by applying NOT logic to UDPRxFfRdEmpty. The data for comparison, UDPRxFfRdData, becomes valid in the next clock cycle. To count the total size of received data, rTotalRxCnt is enabled by rRxFfRdEn, which is delayed by one clock cycle from UDPRxFfRdEn. The lower bits of rTotalRxCnt are applied to generate wExpPatt for comparison with UDPRxFfRdData. Therefore, UDPRxFfRdData and wExpPatt are valid in the same clock cycle and can be compared using rRxFfRdEn signal.

3 CPU Firmware on FPGA

The reference design uses a bare-metal OS for the CPU firmware operating, which facilitates hardware handling. When executing the test system, the first step is to initialize the hardware, described in more details below.

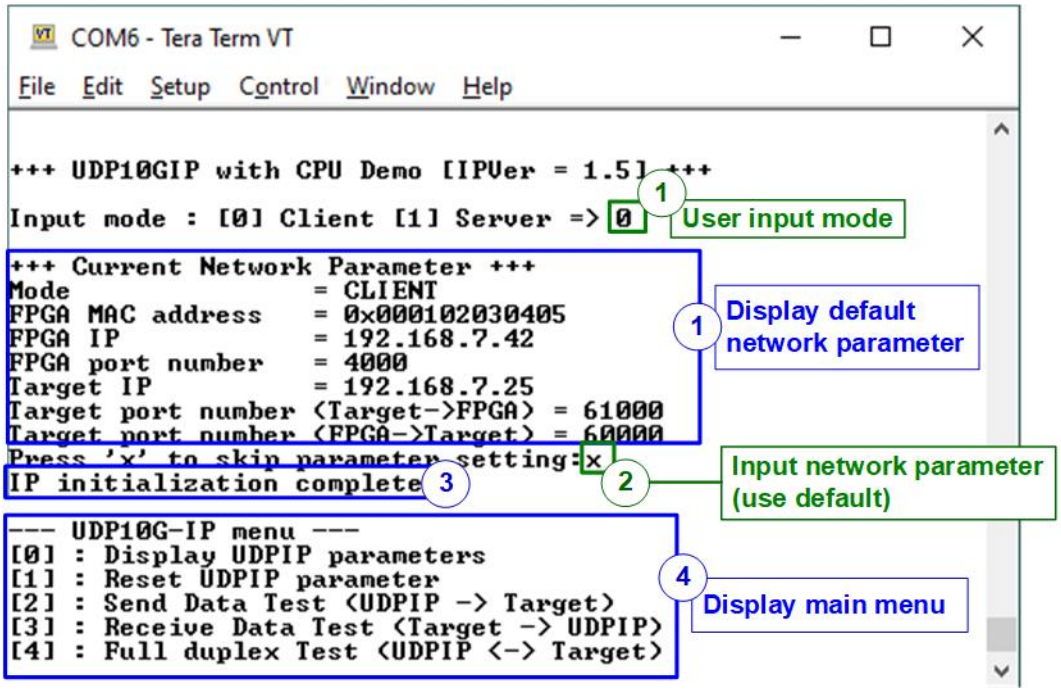

Figure 3-1 System initialization in Client mode by using default parameters

Figure 3-1 illustrates the four-step process for hardware initialization, which is described below.

1) Upon FPGA boot-up, the firmware polls the status of the 10G Ethernet link (USER_RST_INTREG[16]). The CPU waits until the link is up, and then displays a welcome message to show IP information.
2) The menu to select the initialization mode of UDP10G-IP is displayed, allowing the user to choose between the Client or Server mode.

   Note:
   a) *When running in Client mode, UDP10G-IP sends an ARP request to obtain the MAC address of the target device from the ARP reply. When running in Server mode, UDP10G-IP waits until an ARP request is received to decode the MAC address and return an ARP reply.*
   b) *When running the test environment with one FPGA board and Test PC, it is recommended to set the FPGA to run as Client mode.*

3) The CPU displays the default values of the network parameters, including the initialization mode, FPGA MAC address, FPGA IP address, FPGA port number, Target IP address, and Target port number. The firmware has two default parameter sets for the operation mode: Server parameter set and Client parameter set. The user can select to complete the initialization process using the default parameters or by updating some parameters. The details of how to change the parameter are provided in Reset parameters menu (topic 3.2).

4) The CPU waits until the IP completes the initialization process by checking if the busy status (UDP_CMD_INTREG[0]) is equal to 0b. After that, “IP initialization complete” is displayed with the main menu. There are five test operations in the main menu, and more details of each menu are described below.

3.1 Display parameters

This menu displays the current values of all UDP10G-IP parameters. The following steps are executed to display parameters.

1) Read the initialization mode.
2) Read all network parameters from each variable in the firmware following the initialization mode, i.e., source (FPGA) MAC address, source (FPGA) IP address, source (FPGA) port number, Target IP address, and Target port number.
   Note: The source parameters are the FPGA parameters set to UDP10G-IP, while the Target parameters are the parameters of a PC or another FPGA.
3) Print out each variable.

3.2 Reset IP

This menu is used to change some UDP10G-IP parameters, such as IP address and source port number. After setting the updated values to UDP10G-IP, the CPU resets the IP to re-initialize the process using new parameters. Finally, the CPU waits until the initialization is completed. The following steps are executed to reset the parameters.

1) Display all parameters on the console, similar to topic 3.1.
2) If the user uses the default value, skip to the next step. Otherwise, display the menu to set all parameters.
   i) Receive the initialization mode from the user. If the initialization mode is changed, display the latest parameter set of new mode on the console.
   ii) Receive the remaining parameters from the user and verify all inputs. If the input is invalid, the parameter is not updated.
3) Force reset to UDP10G-IP by setting UDP_RST_INTREG[0]=1b.
4) Set all parameters to UDP10G-IP register, such as UDP_SML_INTREG and UDP_DIP_INTREG.
5) De-assert UDP10G-IP reset by setting UDP_RST_INTREG[0]=0b to initiate the initialization process.
6) Clear PattGen and PattVer logic by sending a reset to user logic (USER_RST_INTREG[0]=1b).
7) Monitor the UDP10G-IP busy flag (UDP_CMD_INTREG[0]) until the initialization process is finished (busy flag is de-asserted to 0b).

3.3 Send data test

This menu allows the user to execute Send data test. The user can set the parameters such as total transmit length. If all inputs are valid, the data is transferred by sending 32-bit incremental test data. The operation is completed when all data is transferred.

The following are the steps to send data.

1) Receive the transfer size and packet size from the user and verify that all inputs are valid. If any input is invalid, the operation is cancelled.
2) Set the UserReg registers - the transfer size (USER_TXLEN_INTREG), the Reset flag to clear the initial value of test pattern (USER_RST_INTREG[0]=1b), and the Command register to start the data pattern generator (USER_CMD_INTREG=0). The test pattern generator in UserReg starts to generate test data to UDP10G-IP.
3) Display the recommended parameters of the test application on PC by reading the current parameters in the system. Wait until the user presses any key to start the IP sending operation.
4) Set parameters to UDP10G-IP to start the operation. The packet size is set to UDP_PKL_INTREG, and the total size is set to UDP_TDL_INTREG. Finally, UDP_CMD_INTREG is set to 1b to start IP sending data.
5) Wait for UDP10G-IP to complete the operation by monitoring the busy flag of the IP (UDP_CMD_INTREG[0]=0b). While monitoring the busy flag, the CPU reads the current transfer size from the user logic (USER_TXLEN_INTREG) and displays it on the console every second.
6) Once the operation is completed, the CPU calculates the performance and displays the test result on the console.

3.4 Receive data test

This menu allows the user to execute Receive data test. The user can set the parameters such as total receive length. If all inputs are valid, a 32-bit incremental test data is created for verification with the received data from PC/FPGA when the data verification is enabled.

The following are the steps to receive data.

1) Receive the total transfer size and data verification mode from user and verify that all inputs are valid. The operation is cancelled if some inputs are invalid.
2) Set the UserReg registers, i.e., the Reset flag to clear the initial value of the test pattern (USER_RST_INTREG[0]=1b) and data verification mode (USER_CMD_INTREG[1]= 0b/1b to enable/disable).
3) Display recommended parameter (similar to Step 3 of Send data test).
4) Wait until the total number of received data (USER_RXLEN_INTREG) is equal to the set value (complete condition), or the number of received data is not updated for 100 msec (timeout condition). During receiving data, the CPU displays the current number of received data on the console every second.
5) Stop the timer. Check the interrupt from the timeout (USER_RST_INTREG[8]) and data verification flag (USER_CMD_INTREG[1]) registers when the verification mode is applied. If some errors are found, the error message will be displayed.
6) Calculate performance and show the test result on the console.

3.5 Full duplex test

This menu enables full duplex testing by simultaneously transferring data between the FPGA and another device (PC/FPGA) in both directions. User-defined parameters, such as the total transfer length, are received to initiate the test. If all inputs are valid, the data transfer begins and completes when the data is completely transferred in both directions.
Note: When testing with a PC, the transfer size on the test application (udpdatatest) must match the transfer size set on the FPGA. Two "udpdatatest" are executed, one for sending data and another for receiving data using different port number. When testing with two FPGAs, the port number for sending and receiving data must be the same.

The steps to execute a full duplex test are as follows.

1) Receive the total data size (using the same size for both transfer directions), packet size, and data verification mode (enabled or disabled) from the user and verify that all inputs are valid. The operation is cancelled if some inputs are invalid.
2) Set UserReg registers including transfer size (USER_TXLEN_INTREG), the Reset flag to clear the initial value of the test pattern (USER_RST_INTREG[0]=1b), and the Command register to start data pattern generator with data verification mode (USER_CMD_INTREG=0 or 2).
3) Display the recommended parameters for the test application running on the PC by reading the current parameters in the system.
4) Set UDP10G-IP registers, including packet size (UDP_PKL_INTREG), total transfer size (UDP_TDL_INTREG), and Send command (UDP_CMD_INTREG=1). The IP begins sending data once the UDP_CMD_INTREG is set to 1b. For receiving data, the IP is always ready to receive data without any additional setting.
5) The CPU controls data flow of both directions simultaneously, with two tasks running during the test, as follows.
   a) To send data, the CPU reads the busy flag (UDP_CMD_INTREG[0]) and waits until it is de-asserted to 0b. The busy flag is de-asserted to 0b when the Send command is finished.
   b) To receive data, the CPU reads the total number of received data. The read process finishes when the total number of received data is equal to the set value (no data lost). If the total number of received data does not change for 100 msec (timeout), the read process is also finished.

   If the data is not completely transferred, the current number of transmitted data size (USER_TXLEN_INTREG) and received data size (USER_RXLEN_INTREG) are read and displayed on the console every second.

6) Stop the timer and check the data verification status (USER_CMD_INTREG[1]). If a verification error is found, an error message is displayed.
7) Calculate performance and display the test result on the console.

3.6 Function list in User application

This topic describes the function list to run UDP10G-IP operation.

void init_param(void)	
Parameters	None
Return value	None
Description	Reset parameters following the description in topic 3.2. In the function, show_param and input_param function are called to display parameters and get parameters from user.

int input_param(void)	
Parameters	None
Return value	0: Valid input, -1: Invalid input
Description	Receive network parameters from user, i.e., the initialization mode, FPGA MAC address, FPGA IP address, FPGA port number, Target IP address, and Target port numbers. If all inputs are valid, the parameters are updated. Otherwise, the value does not change. After receiving all parameters, calling show_param function to display parameters.

void show_cursize(void)	
Parameters	None
Return value	None
Description	Read USER_TXLEN_INTREG and USER_RXLEN_INTREG, and then display the current transmitted and received data sizes in Byte, KByte, or MByte unit

void show_interrupt(void)	
Parameters	None
Return value	None
Description	Read interrupt status from UDP_TMO_INTREG and decode interrupt type to display the details of interrupt on the console.

void show_param(void)	
Parameters	None
Return value	None
Description	Display the parameters following the description in topic 3.1

void show_result(void)	
Parameters	None
Return value	None
Description	Read USER_TXLEN_INTREG and USER_RXLEN_INTREG to display total transmitted data size and total received data size. Read the global parameters (timer_val and timer_upper_val) and calculate total time usage to display in usec, msec, or sec unit. Finally, transfer performance is calculated and displayed on MB/s unit.

int udp_recv_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Run Receive data test following description in topic 3.4. It calls show_interrupt, show_cursize, and show_result function.

int udp_send_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Run Send data test following description in topic 3.3. It calls show_cursize and show_result function.

int udp_txrx_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Run Full duplex test following described in topic 3.5. It calls show_interrupt, show_cursize, and show_result function.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read USER_RST_INTREG[16] and wait until the Ethernet connection is linked up

4 Test Software (PC)

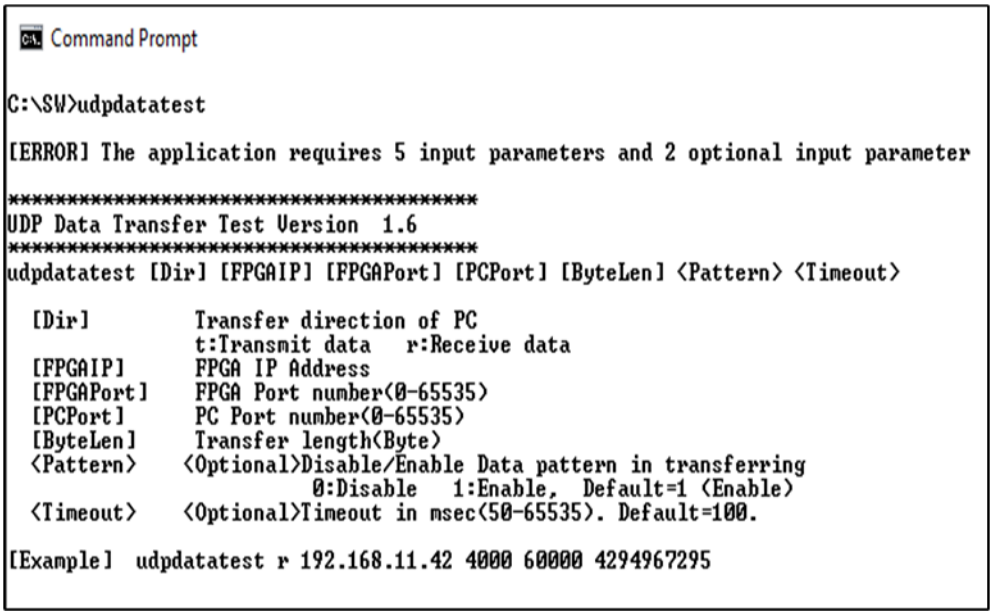

Figure 4-1 udpdatatest application parameter

The “udpdatatest” application is executed to send or receive UDP data on a PC. It requires five mandatory parameters and two optional parameters. It is important to ensure that the parameter inputs match the parameters set on the FPGA. The details of each parameter input are as follows.

Mandatory parameters

1) Dir: : t – when PC sends data to FPGA
   : r – when PC receives data from FPGA
2) FPGAIP : IP address setting on FPGA (Default value in is 192.168.7.42)
3) FPGAPort : Port number of FPGA (Default value in FPGA is 4000)
4) PCPort : PC port number for sending or receiving data
   (Default is 60001 for PC to FPGA and 60000 for FPGA to PC)
5) ByteLen : Transfer length for sending or receiving in byte unit. This value must be aligned to 8 from UDP10G-IP limitation.

Optional parameters

1) Pattern : Default value when user does not input this parameter is 1.
   0 – Generate dummy data in transmit mode or disable data verification in receive mode.
   1 – Generate incremental data in transmit mode or enable data verification in receive mode.
2) Timeout : Timeout for receiving data in msec unit.
   Default value when user does not input this parameter is 100.
   The 100 ms is recommended value for running with UDP10G-IP.

Transmit data mode

The steps for running the test application in transmit mode are as follows.

1) Get the parameters from the user and verify that all inputs are valid.
2) Create a socket and configure the received buffer properties.
3) Set the IP address and port number based on the user parameter, and then establish a connection.
4) Populate the Send buffer with data for transmission. While the data is being sent, the application prints the total amount of data sent every second to the console.
   a) If Pattern=1, the Send buffer is filled with a 32-bit incremental pattern.
   b) If Pattern=0, the Send buffer is not filled and dummy data is used for the test.
5) After all data has been sent, the application displays the test results, including the total size of transmitted data and the performance.

Receive data mode

The steps when running the test application in receive mode are as follows.

1) Follow step (1)-(3) from the Transmit data mode.
2) Continuously read data until the total number of received data equals the set value. If there is no new data received before the timeout, the operation is cancelled. During the data reception, the application prints the total amount of received data every second.
   a) If Pattern=1, the received data is verified using a 32-bit incremental pattern that increases every four bytes of received data.
   b) If Pattern=0, the received data is not verified.
3) If the read loop finishes due to a timeout, the application displays a “Timeout” message with the total number of lost data and received data. The total time used is also reduced by timeout value.
4) After the operation is complete, the application displays the test results, including the performance and the total amount of received data.

5 Revision History

Revision	Date	Description
1.4	9-Mar-23	Update udpdatatest features
1.3	21-Aug-20	Add 10G25GEMAC IP
1.2	8-Mar-19	Support FPGA<->FPGA connection
1.1	17-Nov-17	Correct Receive data test sequence
1.0	14-Sep-17	Initial Release