UDP10G-IP reference design manual

Rev1.3 21-Aug-20

1 Introduction

Comparing to TCP protocol, UDP protocol provides a procedure to send data with a minimum of protocol mechanism, but the data cannot guarantee to be accepted by the destination because of no handshaking dialogues providing in UDP mechanism. Similar to TCP protocol, UDP protocol provides checksums for data integrity and port numbers for addressing different functions at the source and the destination in the communication networks.

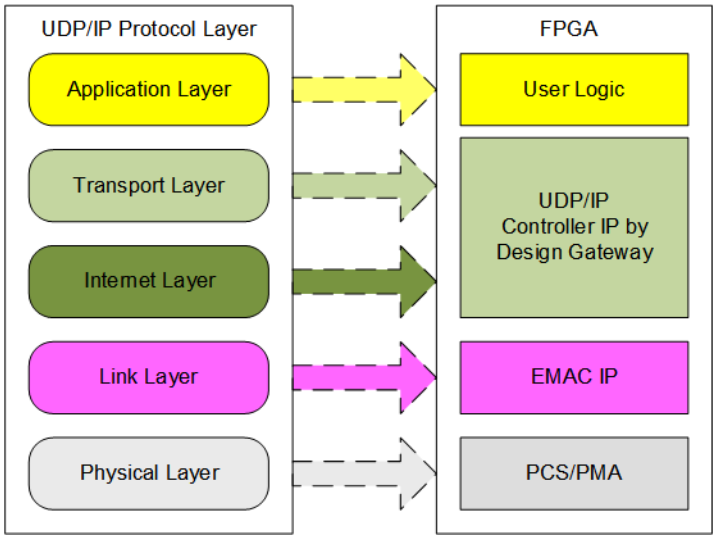

Figure 1-1 UDP/IP Protocol Layer

UDP10G-IP implements Transport and Internet layer of UDP/IP Protocol for building Ethernet packet from the user data which is UDP data to EMAC. UDP10G-IP prepared UDP data from the user logic and then inserts UDP/IP header. On the other hand, the received Ethernet packet from EMAC is extracted by UDP10G-IP. The header of the packet is verified and UDP data is forwarded to the user logic when the packet is valid. Otherwise, the packet is rejected.

The lower layer protocols are implemented by EMAC-IP and PCS/PMA-IP. 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 provides the evaluation system which includes simple user logic to transfer data by using UDP10G-IP. The target device for transferring data with FPGA in test environment can be Test PC or another UDP10G-IP on FPGA board. When running with PC, the test applications for sending or verifying UDP data via Ethernet connection at high-speed rate must be run, i.e. “udpdatatest”.

To allow the user controlling the test parameters and the operation of UDP10G-IP demo via UART, the CPU system is included. It is easy for the user to set the test parameters and monitor the current status on UART console. The firmware on CPU is built by using bare-metal OS. More details of the demo are described as follows.

2 Hardware overview

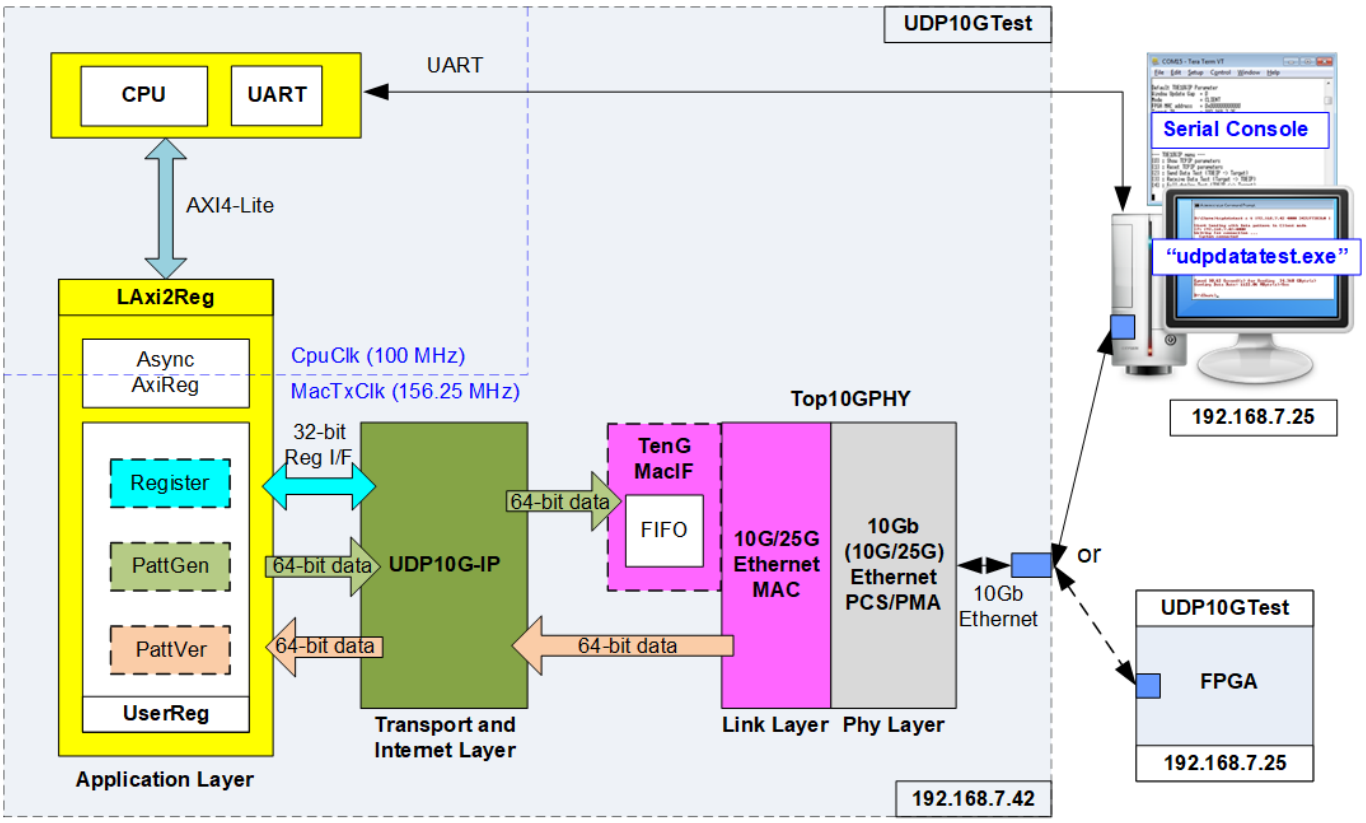

Figure 2-1 Demo Block Diagram

In test environment, two devices are used for 10Gb Ethernet transferring. First device runs in Client mode and another device runs in Server mode. In the demo, the client device is UDP10G-IP on FPGA board while the server device can be UDP10G-IP on FPGA board or PC, as shown in **Error! Reference source not found.**. When using PC, the test applications (udpdatatest) must be run on PC to transfer data with UDP10G IP within FPGA.

In FPGA system, UDP10G-IP connects to 10G/25G Ethernet MAC and 10G/25G Ethernet PCS/PMA to implement all UDP/IP layers. User interface of UDP10G-IP connects to UserReg within AsyncAxiReg which consists of Register file for interfacing with Register interface, PattGen for sending test data via Tx FIFO interface and VerifyPatt for verifying test data via Rx FIFO interface. Register files of UserReg are controlled by CPU firmware through AXI4-Lite bus.

Moreover, the additional logics are designed in the reference design. TenGMacIF is designed to be the data buffer between UDP10G-IP and Xilinx 10G EMAC-IP when Xilinx 10G EMAC-IP is not ready to receive the packet during one packet transmission.

Two clock domains are applied in the test design, i.e. CpuClk which is the independent clock for running the CPU system and MacTxClk which is the clock output from 10G/25G Ethernet PCS/PMA. So, AsyncAxiReg is designed to support asynchronous signals between CpuClk and MacTxClk. More details of each module inside the UDP10GCPUTest are described as follows.

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

10/25Gb PCS/PMA implements physical layer of 10G Ethernet. It connects to the external hardware which is 10G BASE-R SFP+ module. The user interface is 64-bit XGMII interface running at 156.25 MHz which is designed to connect with 10G Ethernet MAC. This IP core is provided by Xilinx without the charge. More details of the core are described in the following link.

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

10G/25G EMAC connects between UDP10G-IP and 10G PCS/PMA module. The interface with UDP10G-IP is 64-bit AXI4 stream while the interface with 10G PCS/PMA module is 64-bit XGMII interface at 156.25 MHz.

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

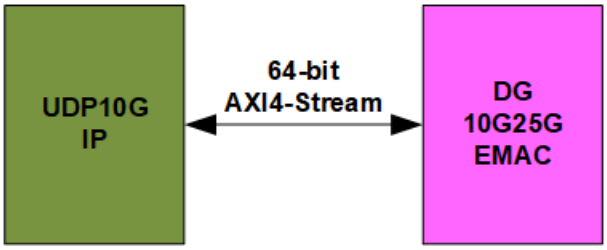

Figure 2-2 UDP10G-IP connecting with DG 10G25GEMAC IP

When connecting with Xilinx 10G/25G EMAC-IP, the small buffer must be connected between UDP10G-IP and Xilinx 10G/25G EMAC-IP. So, TenGMacIF module is designed to be the interface module. More details of TenGMacIF are described in the next topic. The details of Xilinx EMAC are described in the following link.
https://www.xilinx.com/products/intellectual-property/do-di-10gemac.html
https://www.xilinx.com/products/intellectual-property/ef-di-25gemac.html

2.3 TenGMacIF (only when using Xilinx EMAC IP)

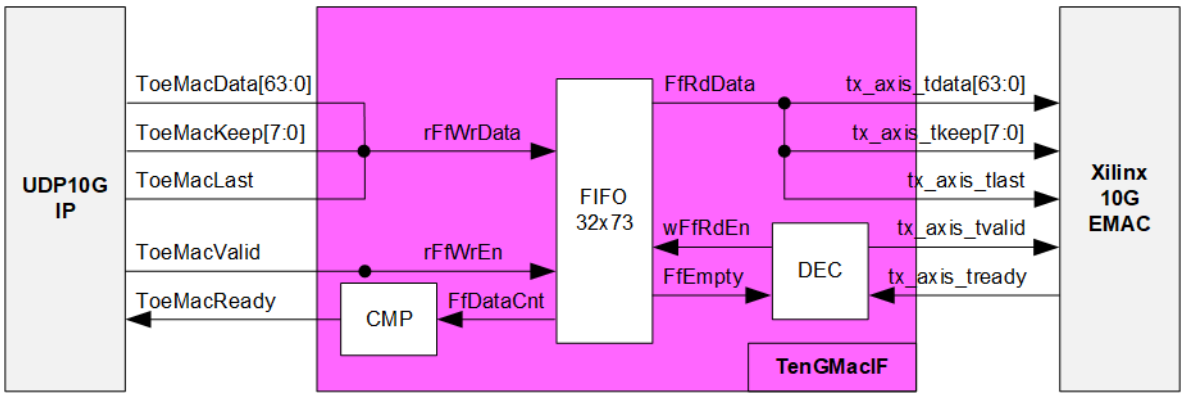

Figure 2-3 TenGMacIF Block Diagram

Tx interface timing diagram of Xilinx 10G/25G EMAC and UDP10G-IP are different. UDP10G-IP needs to send data of one packet continuously, but Xilinx EMAC does not support this feature. Xilinx EMAC may de-assert ready signal to pause data receiving before end of the packet. TenGMacIF is designed to store transmitted data from UDP10G-IP when Xilinx 10G/25G EMAC is not ready to receive new data. According to real-board testing, tready output from Xilinx 10G EMAC is not de-asserted for long time per packet, so small FIFO which has 32-data depth is included to store the data during pausing time.

The operation of TenGMacIF is split into two sides, following the interface, i.e. Writing FIFO and Reading FIFO. Timing diagram for writing FIFO from UDP10G-IP is shown in Figure 2-4 while timing diagram for reading FIFO and forwarding to Xilinx 10G EMAC is shown in Figure 2-5.

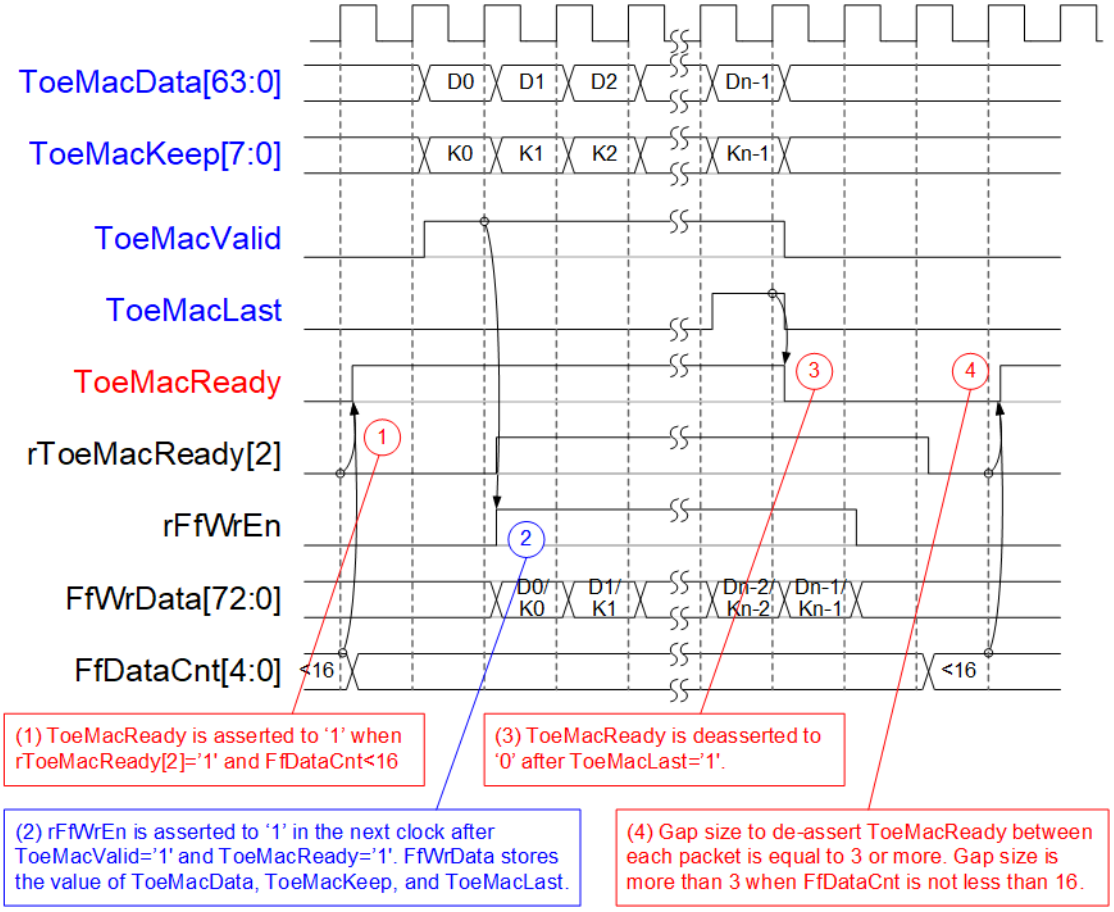

Figure 2-4 Write FIFO timing diagram of TenGMacIF

(1) Before asserting ToeMacReady to '1' for receiving the new data, two conditions must be met.
   a) The logic checks if free space in FIFO is more than 16 by monitoring when FfDataCnt is less than 16.
   Note: By real-board monitoring, it is found that Xilinx 10G/25G EMAC-IP de-asserts tx_axis_tready to '0' less than 16 cycles for each packet.
   b) The previous packet is completely transferred more than two clock cycles by checking rToeMacReady[2]='0'. rToeMacReady[2] is created by ToeMacReady with two clock cycle latency time. To pause the packet transmission, ToeMacReady is de-asserted to '0' to wait until EMAC and TenGMacIF is ready.

(2) During packet transmission, ToeMacValid from UDP10G-IP and ToeMacReady are asserted to '1' continuously until end of packet, so the packet is transferred without pausing time. In the next clock after ToeMacReady and ToeMacValid are asserted to '1', rFfWrEn is asserted to '1' to store received data. FIFO data size is 73-bit for storing 64-bit data (ToeMacData), 8-bit of byte enable (ToeMacKeep) and last flag (ToeMacLast).

(3) When the last data is received (ToeMacLast='1'), ToeMacReady is de-asserted to '0' for 3 clock cycles to be the gap size for each packet transmission.

(4) ToeMacReady is de-asserted to '0' more than 3 clock cycles if FfDataCnt is not less than 16.

Timing diagram to forward data from FIFO to Xilinx 10G/25G EMAC is shown in Figure 2-5.

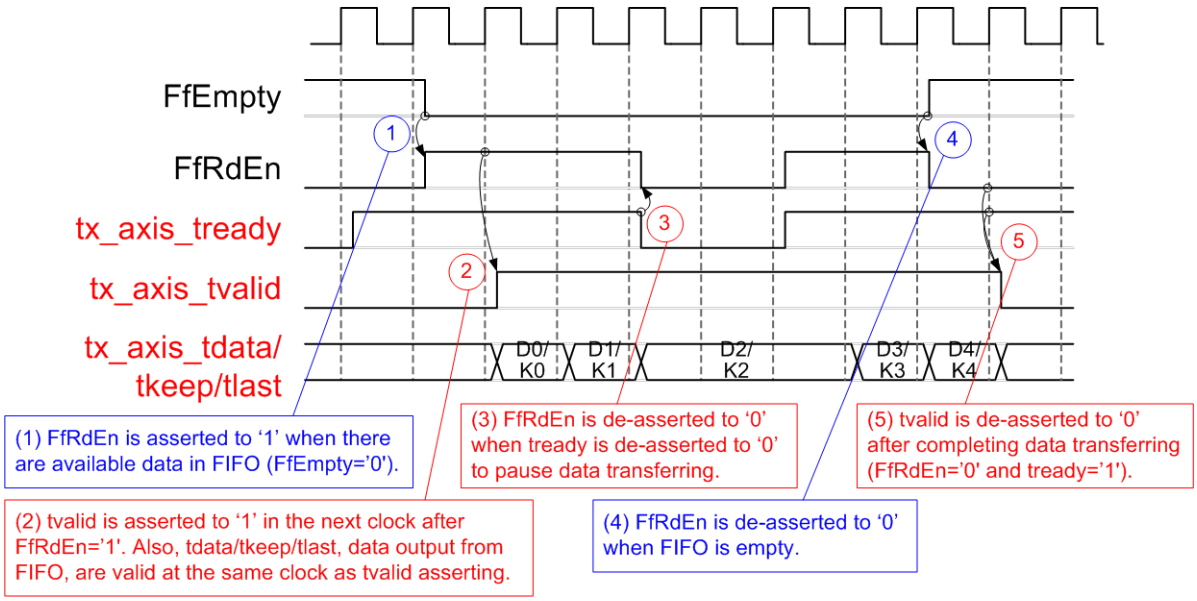

Figure 2-5 Read FIFO timing diagram of TenGMacIF

(1) When the new packet is received from UDP10G-IP, FIFO empty (FfEmpty) is de-asserted to '0'. Since the packet from UDP10G-IP is transferred continuously, FIFO is not empty until a packet is completely transferred. The logic asserts read enable of FIFO (FfRdEn) to '1' for reading data from FIFO and forwarding data to Ethernet MAC.
(2) After reading data from FIFO, tx_axis_tvalid is asserted to '1' with the valid data for transferring to EMAC in the next cycle.
(3) When EMAC is not ready to receive data (tx_axis_tready='0'), read enable must be de-asserted to '0' to pause data transmission.
(4) After the packet is completely transferred, the FIFO shows empty status (FfEmpty='1'). At the same time, FfRdEn is de-asserted to '0' to stop data transmission.
(5) Finally, tx_axis_tvalid is de-asserted to '0' to finish the packet transmission.

2.4 UDP10G-IP

UDP10G-IP implements UDP/IP stack and offload engine. User interface has two signal groups, i.e. control signals and data signals. Register interface is applied to set control registers and monitor status signals. Data signals are accessed by using FIFO interface. More details are described in datasheet.
https://dgway.com/products/IP/UDP10G-IP/dg_udp10gip_data_sheet_xilinx_en.pdf

2.5 CPU and Peripherals

32-bit AXI4-Lite is applied to be the bus interface for the CPU accessing the peripherals such as Timer and 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 AXI4-Lite bus. CPU assigns the different base address and the address range to each peripheral for accessing one peripheral at a time.

In the reference design, the CPU system is built with one additional peripheral to access the test logic. The base address and the range for accessing the test logic are defined in the CPU system. So, the hardware logic must be designed to support AXI4-Lite bus standard for supporting CPU writing and reading. LAxi2Reg module is designed to connect the CPU system as shown in Figure 2-6.

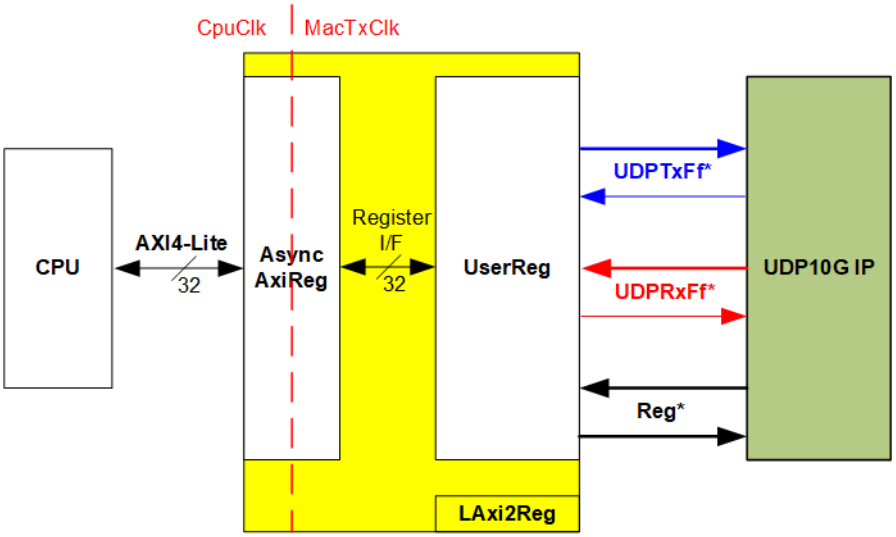

Figure 2-6 LAxi2Reg block diagram

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

UserReg includes the register file of the parameters and the status signals of test logics, including UDP10G-IP. Both data interface and control interface of UDP10G-IP are connected to UserReg. More details of AsyncAxiReg and UserReg are described as follows.

2.5.1 AsyncAxiReg

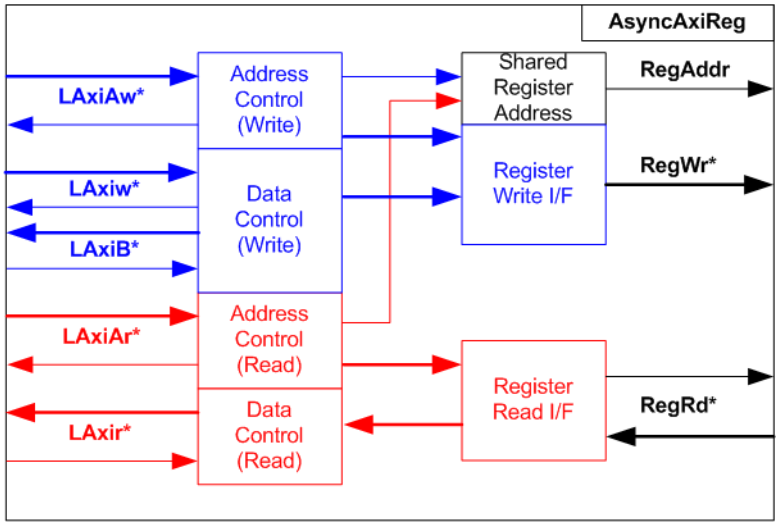

Figure 2-7 AsyncAxiReg Interface

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

According to AXI4-Lite standard, the write channel and the read channel are operated independently. Also, the control and data interface of each channel are run separately. So, the logic inside AsyncAxiReg to interface with AXI4-Lite bus is split into four groups, i.e. Write control logic, Write data logic, Read control logic and Read data logic as shown in the left side of Figure 2-7. Write control I/F and Write data I/F of AXI4-Lite bus are latched and transferred to be Write register interface with clock-crossing registers. In the same way, Read control I/F and Read data I/F of AXI4-Lite bus are latched and transferred to be Read register interface with clock-crossing registers. In register interface, RegAddr is shared signal for write and read access, so it loads the value from LAxiAw for write access or LAxiAr for read access.

The simple register interface is compatible with single-port RAM interface for write transaction. The read transaction of the register interface is slightly modified from RAM interface by adding RdReq and RdValid signals for controlling read latency time. The address of register interface is shared for write and read transaction, so user cannot write and read the register at the same time. The timing diagram of the register interface is shown in Figure 2-8.

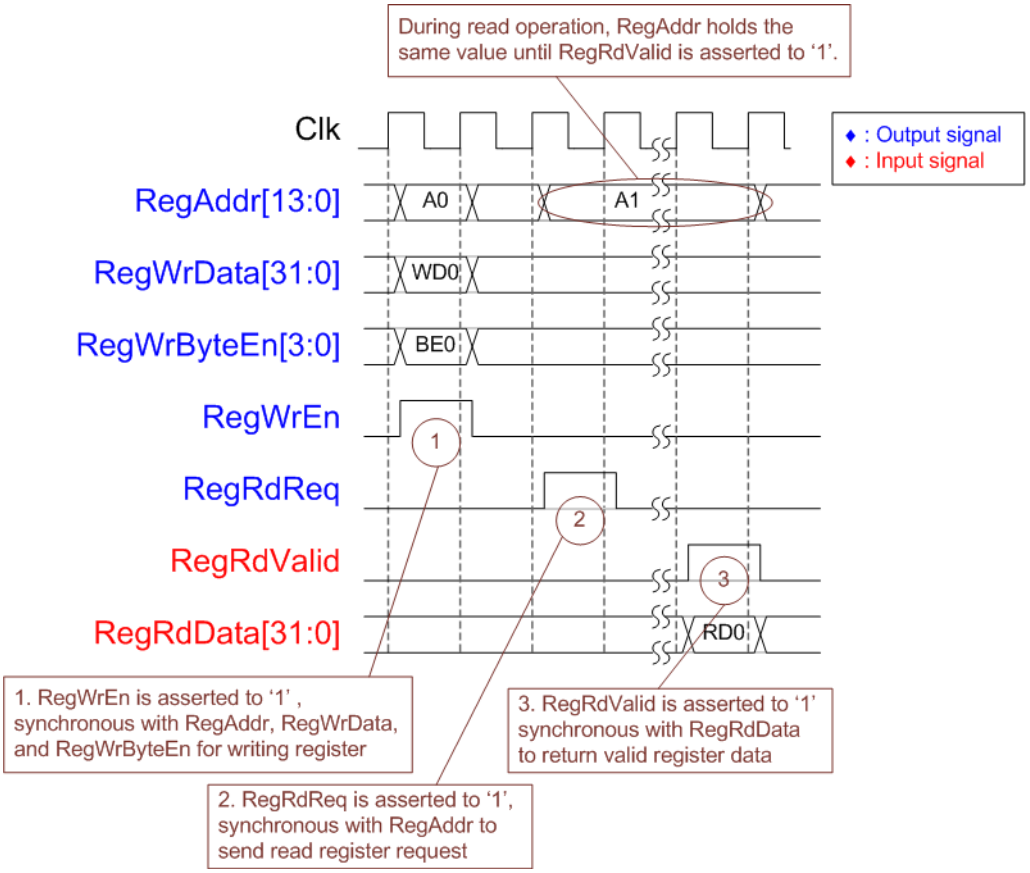

Figure 2-8 Register interface timing diagram

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

2.5.2 UserReg

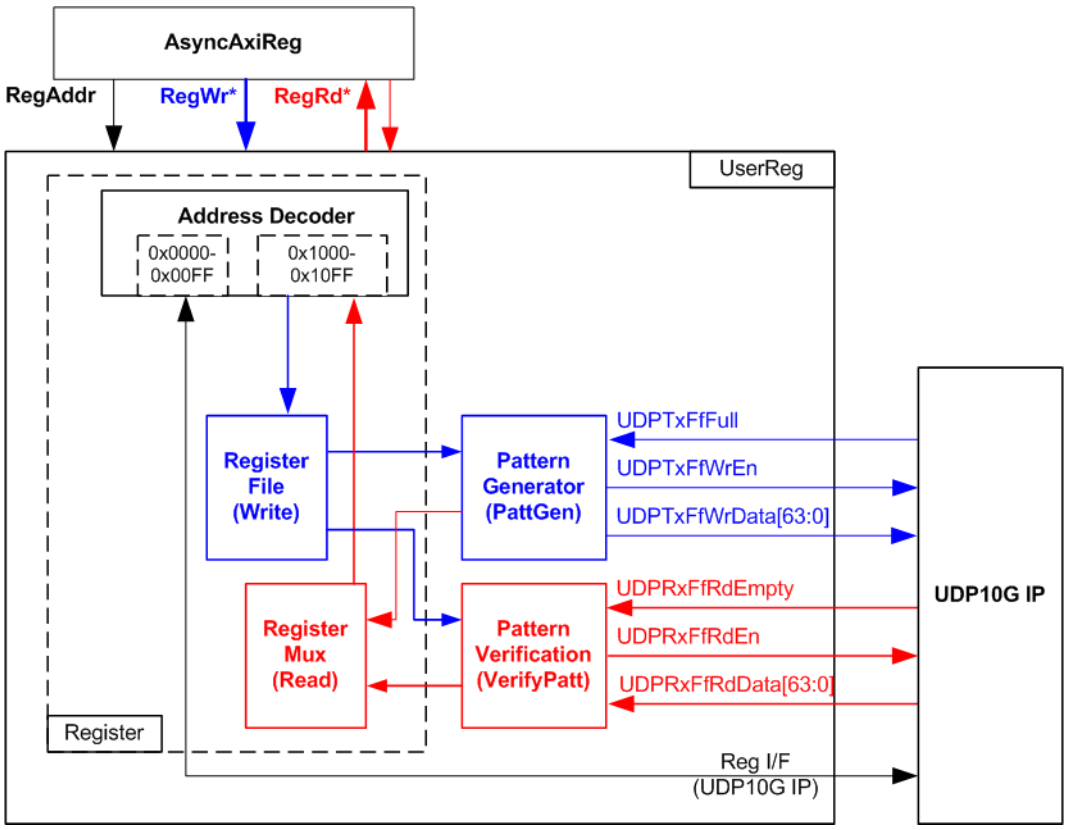

Figure 2-9 UserReg block diagram

The logic inside UserReg has three operations, i.e. Register, Pattern generator (PattGen), and Pattern verification (VerifyPatt). Register block decodes the address requested from AsyncAxiReg and then selects the active register for write or read transaction. Pattern generator block is designed to send 64-bit test data to UDP10G-IP following FIFO interface standard. Pattern verification block is designed to read and verify 64-bit data from UDP10G-IP following FIFO interface standard. More details of each block are described as follows.

Register Block

The address range to map to UserReg is split into two areas, i.e. UDP10G-IP register (0x0000-0x00FF) and UserReg register (0x1000-0x10FF).

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

To read register, one multiplexer is designed to select the read data within each address area. The lower bit of RegAddr is applied in each Register area to select the data. Next, the address decoder uses the upper bit to select the read data from each area for returning to CPU. Totally, the latency of read data is equal to one clock cycle, so RegRdValid is created by RegRdReq with asserting one D Flip-flop. More details of the address mapping within 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_REG	Mapped to RST register within UDP10G-IP
BA+0x0004	UDP_CMD_REG	Mapped to CMD register within UDP10G-IP
BA+0x0008	UDP_SML_REG	Mapped to SML register within UDP10G-IP
BA+0x000C	UDP_SMH_REG	Mapped to SMH register within UDP10G-IP
BA+0x0010	UDP_DIP_REG	Mapped to DIP register within UDP10G-IP
BA+0x0014	UDP_SIP_REG	Mapped to SIP register within UDP10G-IP
BA+0x0018	UDP_DPN_REG	Mapped to DPN register within UDP10G-IP
BA+0x001C	UDP_SPN_REG	Mapped to SPN register within UDP10G-IP
BA+0x0020	UDP_TDL_REG	Mapped to TDL register within UDP10G-IP
BA+0x0024	UDP_TMO_REG	Mapped to TMO register within UDP10G-IP
BA+0x0028	UDP_PKL_REG	Mapped to PKL register within UDP10G-IP
BA+0x0038	UDP_SRV_REG	Mapped to SRV register within UDP10G-IP
BA+0x1000 – BA+0x10FF: UserReg control/status		
BA+0x1000 Wr/Rd	Total transmit length (USER_TXLEN_REG)	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_REG is written by user.
BA+0x1004 Wr/Rd	User Command (USER_CMD_REG)	Wr [0] – Start Transmitting. Set ‘0’ to start transmitting. [1] – Data Verification enable (‘0’: Disable data verification, ‘1’: Enable data verification) Rd [0] – Tx Busy. (‘0’: Idle, ‘1’: Tx module is busy) [1] – Data verification error (‘0’: Normal, ‘1’: Error) This bit is auto-cleared when user starts new operation or reset.
BA+0x1008 Wr/Rd	User Reset (USER_RST_REG)	Wr [0] – Reset signal. Set ‘1’ to reset the logic. This bit is auto-cleared to ‘0’. [8] – Set ‘1’ to clear TimerInt latch value Rd [8] – Latch value of TimerInt output from IP (‘0’: Normal, ‘1’: TimerInt=’1’ is detected) This flag can be cleared by system reset condition or setting USER_RST_REG[8]=’1’. [16] – Ethernet Linkup status (‘0’: Link down, ‘1’: Link up)
BA+0x100C Rd	FIFO status (USER_FFSTS_REG)	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_REG)	Rd [31:0] – Current received size in QWord unit (64-bit). The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x1080 Rd	EMAC IP version (EMAC_VER_REG)	Rd[31:0] – Mapped to IPVersion output from DG 10G25GEMAC-IP when the system integrates DG 10G25GEMAC-IP.

Pattern Generator

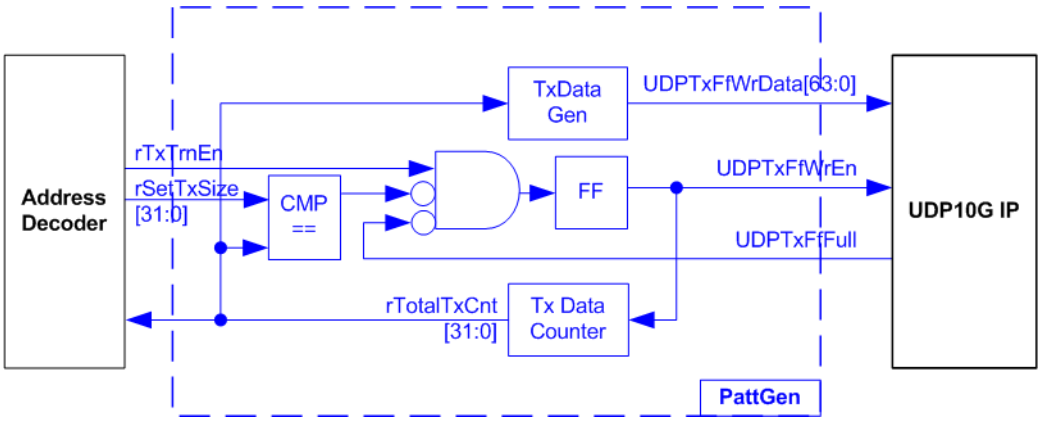

Figure 2-10 PattGen block

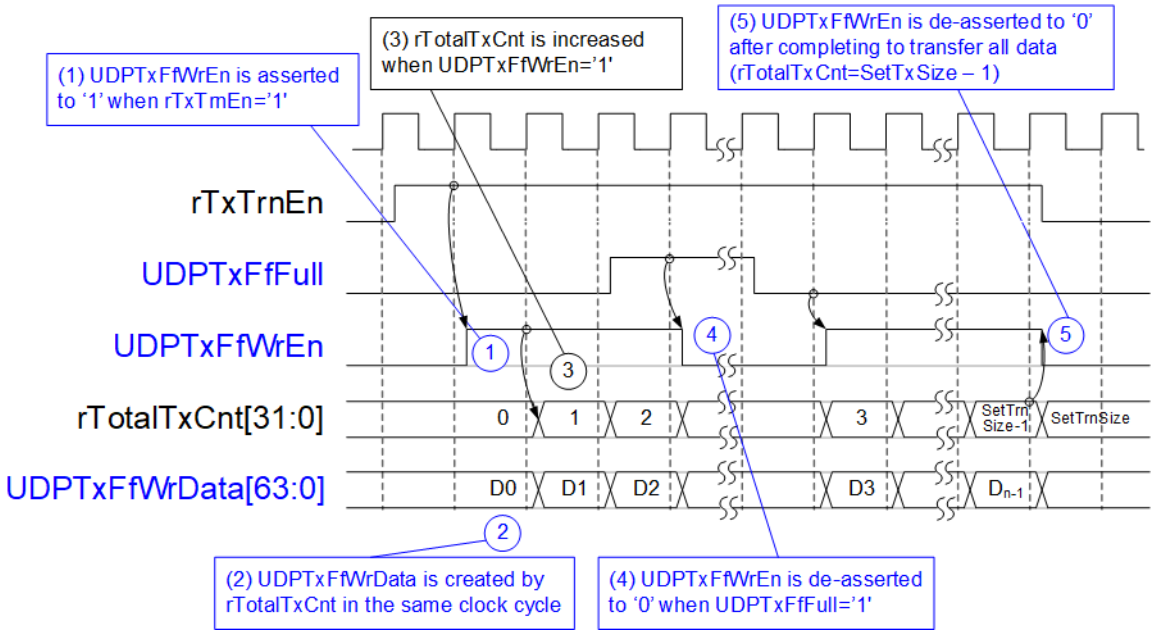

Figure 2-11 PattGen Timing diagram

PattGen generates test data to UDP10G-IP. rTxTrnEn is asserted to '1' when USER_CMD_REG[0] is set to '0'. When rTxTrnEn is '1', UDPTxFfWrEn is controlled by UDPTxFfFull. UDPTxFfWrEn is de-asserted to '0' when UDPTxFfFull is '1'. rTotalTxCnt is the data counter to check total data sent to UDP10G-IP. rTotalTxCnt is also used to generate 32-bit incremental data to UDPTxFfWrData signal. rTxTrnEn is de-asserted to '0' after finishing transferring total data, set by rSetTxSize.

Pattern Verification

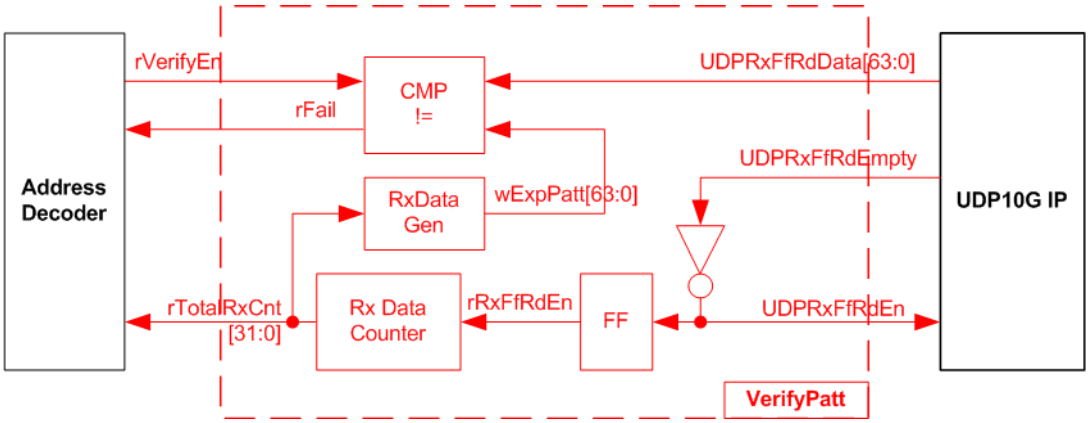

Figure 2-12 PattVer block

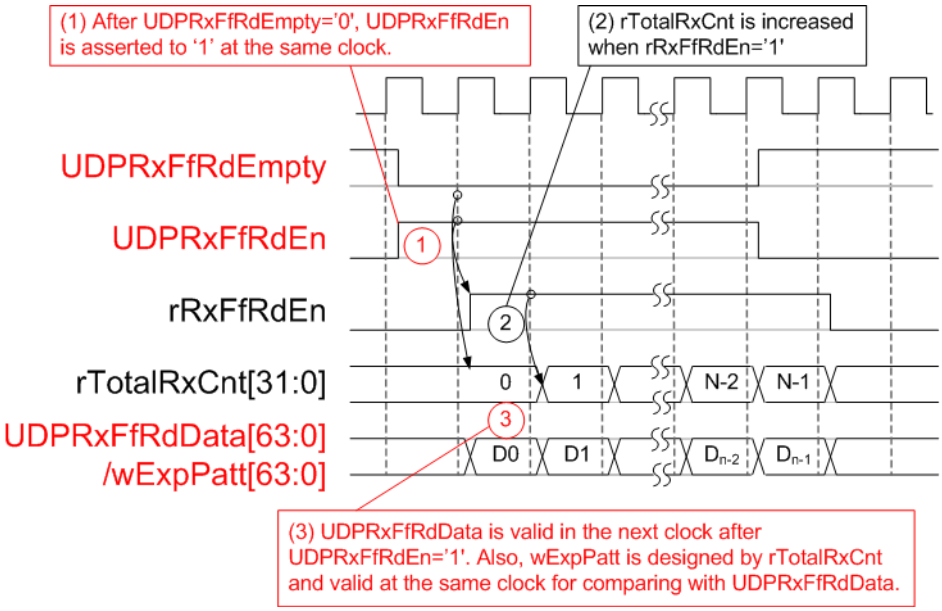

Figure 2-13 PattVer Timing diagram

PattVer is designed to read test data from UDP10G-IP with or without data verification, depending on rVerifyEn flag. When rVerifyEn is set to '1', data comparison is enabled to compare read data (UDPRxFfRdData) with the expected pattern (wExpPatt). If data verification is failed, rFail is asserted to '1'. UDPRxFfRdEn is designed by using NOT logic of UDPRxFfRdEmpty. UDPRxFfRdData is valid for data comparison in the next clock. rRxFfRdEn which is one clock latency of UDPRxFfRdEn is applied to be counter enable of rTotalRxCnt which is designed to count total transfer size and used to generate wExpPatt.

3 CPU Firmware on FPGA

After FPGA boot-up, 10G Ethernet link up status (USER_RST_REG[16]) is polling. The CPU waits until link up is found. Next, welcome message is displayed and user selects the initialization mode of UDP10G-IP to be Client or Server.

To initialize as client mode, UDP10G-IP sends ARP request to get the MAC address from the destination device. For server mode, UDP10G-IP waits ARP request to decode MAC address and returns ARP reply to complete initialization process.

If test environment uses two FPGA boards, the operation mode on two UDP10G-IPs must be different (one is client and another is server). To run with PC, it is recommended to set FPGA as client mode. When PC receives ARP request, PC always returns ARP reply. It is not simple to force PC sending ARP request to FPGA.

The software has two default parameters for each operation mode. After receiving the mode from the user, the default parameters in the selected mode are displayed on the console. The user can select to complete the initialization by using default parameters or updated parameters. The example when the system is initialized in Client mode by using default parameters is shown in Figure 3-1.

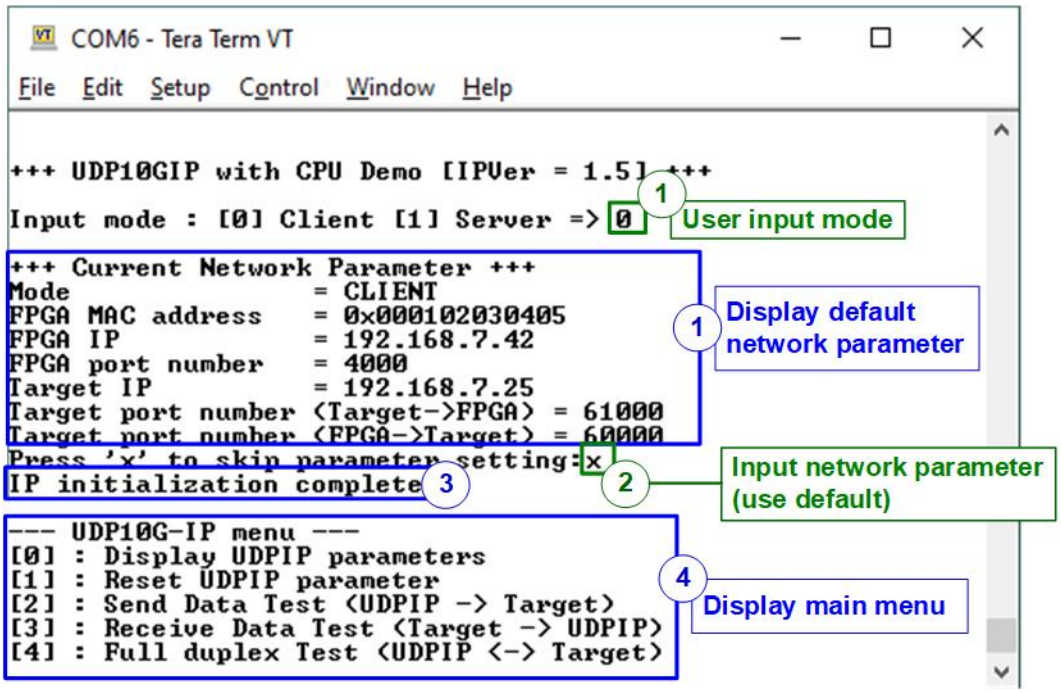

Figure 3-1 System initialization in Client mode by using default parameters

There are four steps to complete initialization process as follows.

1) CPU receives the initialization mode and then displays default parameters in that mode on the console.
2) User inputs 'x' to complete initialization process by using default parameters. Other keys are set for changing some parameters. More details for changing some parameters are described in Reset IP menu (topic 3.2).
3) CPU waits until UDP10G-IP completes initialization sequence (UDP_CMD_REG[0]='0').
4) Main menu is displayed. There are five test operations for user selection. More details of each menu are described as follows.

3.1 Show parameters

This menu is used to show current parameters of UDP10G-IP, i.e. the initialization mode, source MAC address, destination IP address, source IP address, destination port and source port. The sequence of display parameters is as follows.

1) Read all network parameters from each variable in firmware.
2) Print out each variable.

3.2 Reset IP

This menu is used to change UDP10G-IP parameters such as IP address and source port number. After setting updated parameter to UDP10G-IP register, the CPU resets the IP to re-initialize by using new parameters. Finally, the CPU monitors busy flag to wait until the initialization is completed. The sequence to reset the IP is as follows.

1) Display current parameter value on the console.
2) Receive initialization mode from user and confirm that input is valid. If initialization mode is changed, the latest parameter set of new mode will be displayed on the console.
3) Receive remaining input parameters from user and check input whether it is valid or not. If the input is invalid, the parameter will not be updated.
4) Force reset to UDP10G-IP by setting UDP_RST_REG[0]='1'.
5) Set all parameters to UDP10G-IP register such as UDP_SML_REG and UDP_DIP_REG.
6) De-assert UDP10G-IP reset by setting UDP_RST_REG[0]='0'.
7) Clear PattGen and PattVer logic by sending reset to user logic (USER_RST_REG[0]='1').
8) Monitor UDP10G-IP busy flag (UDP_CMD_REG[0]) until the initialization process is finished (busy flag is de-asserted to '0').

3.3 Send data test

User needs to input two parameters, i.e. total transmit length and packet size. The operation is cancelled if some inputs are invalid. During the test, 32-bit incremental data is generated from the logic and sent to the target (PC or FPGA). The received data is verified by the target (PC or FPGA). After all data are transferred completely, the operation is exit. More details of this menu are described as follows.

1) Receive transfer size and packet size from user and verify that all inputs are valid.
2) Set UserReg registers, i.e. transfer size (USER_TXLEN_REG), reset flag to clear initial value of test pattern (USER_RST_REG[0]='1') and command register to start data pattern generator (USER_CMD_REG=0). After that, test pattern generator in UserReg starts to generate test data to UDP10G-IP.
3) Display recommended parameters of test application on PC by reading current parameters in the system. Wait until user press any key to start IP sending operation.
4) Set parameters to UDP10G-IP to start operation. Packet size is set to UDP_PKL_REG and total size is set to UDP_TDL_REG. Finally, UDP_CMD_REG is set to 1 to start IP sending data.
5) Wait until UDP10G-IP completes operation by monitoring IP busy flag (UDP_CMD_REG[0] ='0'). During monitoring busy flag, CPU reads current transfer size from user logic (USER_TXLEN_REG) and displays on the console every second.
6) Calculate performance and show test result on the console.

3.4 Receive data test

In receive data test, user sets total received size and data verification mode (enable or disable). The operation is cancelled if some inputs are invalid. During the test, 32-bit incremental data is generated to verify the received data from PC/FPGA when data verification mode is enabled. The sequence of this test is as follows.

1) Receive total transfer size and data verification mode from user and verify that all inputs are valid.
2) Set UserReg registers, i.e. reset flag to clear the initial value of test pattern (USER_RST_REG[0]='1') and data verification mode (USER_CMD_REG[1]='0' or '1').
3) Display recommended parameter of test application running on PC by reading current parameters in the system.
4) Wait until UDP10G-IP receives the first packet by monitoring current received size (USER_RXLEN_REG) not equal to '0'. Start timer after receiving the first packet.
5) Wait until received size (USER_RXLEN_REG) does not change more than 100 msec or total data are received. During the operation, CPU reads current received size from user logic (USER_RXLEN_REG) and displays the results on the console every second.
6) Stop timer. Check interrupt from timeout (USER_RST_REG[8]) and data verification flag (USER_CMD_REG[1]) register when verification mode is applied. If some errors are found, the error message will be displayed.
7) Calculate performance and show test result on the console.

3.5 Full duplex test

This menu is designed to run full duplex test by transferring data between FPGA and PC/FPGA in both directions at the same time. Three inputs are received from user, i.e. total size for both directions, packet size for FPGA sending logic and data verification mode for FPGA receiving logic.

To run full duplex test by using PC, user runs “udpdatatest” application on two consoles. First application is used to receive data with FPGA and another application is used to send data with FPGA. The port using in two applications must be different. In case of two FPGAs test environment, one port is applied to set to each FPGA. The sequence of this test is as follows.

1) Receive total data size, packet size and data verification mode from the user and verify that all inputs are valid.
2) Set UserReg registers, i.e. transfer size (USER_TXLEN_REG), reset flag to clear the initial value of test pattern (USER_RST_REG[0]=’1’) and command register to start data pattern generator with data verification mode (USER_CMD_REG=1 or 3).
3) Display recommended parameter of test application running on PC by reading current parameters in the system.
4) Set UDP10G-IP registers, i.e. packet size (UDP_PKL_REG), total transfer size (UDP_TDL_REG) and write command (UDP_CMD_REG=1). IP starts sending data operation after UDP_CMD_REG is set to 1. For receiving data, IP is always ready to receive data without additional setting.
5) Wait until finishing the operation for both sending and receiving direction.
   a. For sending direction, wait until busy flag of UDP10G-IP (UDP_CMD_REG[0])=’0’.
   b. For receiving direction, wait until total receive size is equal to set value or total receive size does not change for 100 msec (timeout condition)

   During running the test, CPU reads current transfer size of both directions from user logic (USER_TXLEN_REG and USER_RXLEN_REG) and displays on the console every second.
6) Check interrupt from timeout (USER_RST_REG[8]) and data verification flag (USER_CMD_REG[1]) register when verification mode is applied. If some errors are found, error message will be displayed.
7) Calculate performance and show 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	Set network parameters to UDP10G-IP register from global parameters. After reset is de-asserted, it waits until UDP10G-IP busy flag is de-asserted to '0'.

int input_param(void)	
Parameters	None
Return value	0: Valid input, -1: Invalid input
Description	Receive network parameters from user, i.e. Mode, FPGA MAC address, FPGA IP address, FPGA port number, Target IP address and Target port number for both transfer directions. 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.

void show_cursize(void)	
Parameters	None
Return value	None
Description	Read USER_TXLENL/H_REG and USER_RXLENL/H_REG, and then display the current transmitted and received size in Byte, KByte, or MByte unit

void show_interrupt(void)	
Parameters	None
Return value	None
Description	Read interrupt status from UDP_TMO_REG 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 current value of the network parameters set to UDP10G-IP such as IP address, MAC address and port number.

void show_result(void)	
Parameters	None
Return value	None
Description	Read USER_TXLENL/H_REG and USER_RXLENL/H_REG to display total transmit size and total received 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.

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.

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.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read USER_RST_REG[16] and wait until ethernet is linked up

4 Test Software (PC)

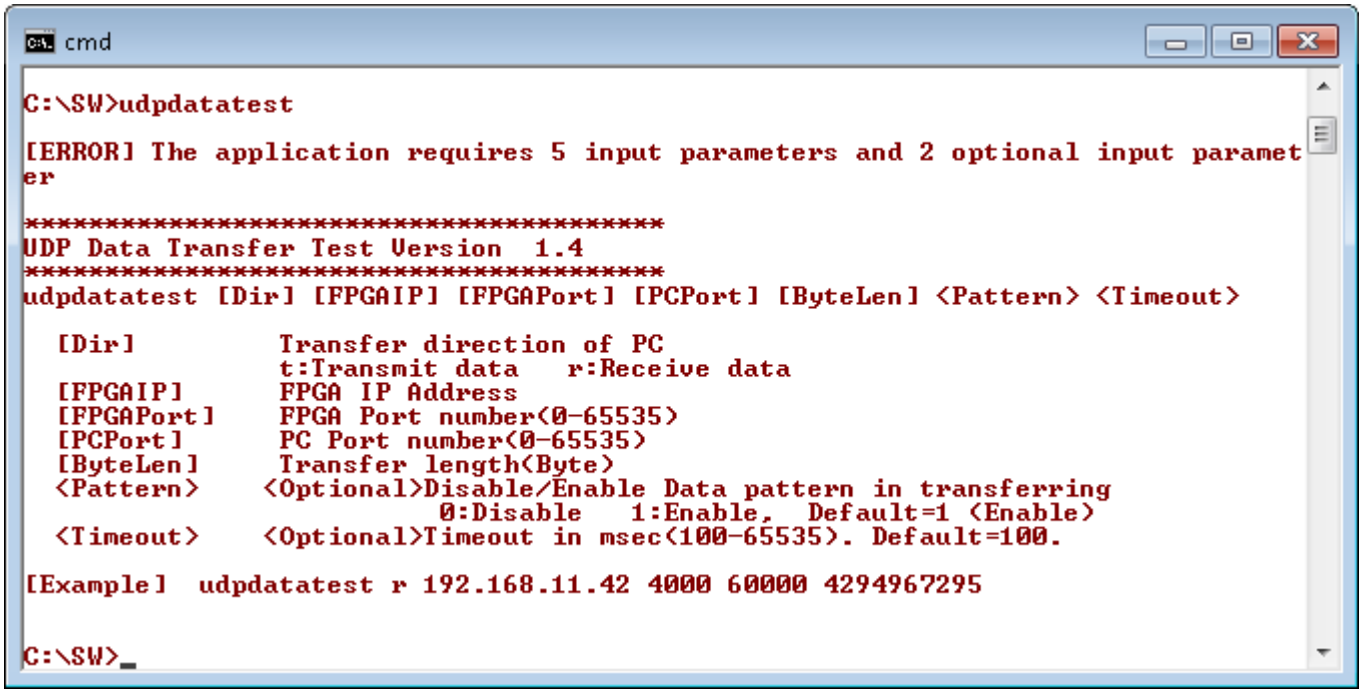

Figure 4-1 udpdatatest application parameter

"udpdatatest" is an application on PC for sending or receiving UDP data. There are five parameters and two optional parameters. The parameter input should be matched to parameter set on FPGA. More details of each parameter input are as follows.

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.
6) Pattern (optional): 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.
7) Timeout (optional): Timeout for receiving data in msec unit.
   Default value when user does not input this parameter is 100.
   100 ms is recommended value for running with UDP10G-IP.

Receive data mode

The step when running the test application in receive mode is described as follows.

1) Get parameters from user and verify that the input is valid.
2) Create the socket and then set properties of received buffer.
3) Set IP address and port number from user parameter and then connect.
4) Repeat to read data until total data is equal to set value or no more data is received with timeout asserting. During reading data, the application prints total received data on the console every second.
   a) When Pattern=1, the read data is verified by 32-bit incremental pattern which is increased every 4-byte received data.
   b) When Pattern=0, the read data is not verified.
5) When the read loop is finished by timeout condition, “Timeout” message is displayed with total lost size and total received size.
6) After finishing the operation, the application displays performance as a test result.

Transmit data mode

The step when running the test application in transmit mode is described as follows.

1) Follow step (1)-(3) in Receive data mode.
2) Send data to the send buffer for sending out. During sending data, the application prints total sent data on the console every second.
   a) When Pattern=1, the send buffer is filled by 32-bit incremental pattern.
   b) When Pattern=0, the send buffer is not filled. Dummy data is applied in the test.
3) After finishing sending all data, the application displays performance with total data size as a test result.

5 Revision History

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