UDP40G-IP reference design

Rev2.00 14-Mar-24

1 Introduction

In comparison to the TCP protocol, UDP minimizes protocol mechanisms during data transmission. It lacks a handshake and data recovery process to ensure the receiver accepts all data accurately. However, similar to TCP, UDP provides checksums for data integrity and port numbers for addressing different functions at the source and destination in networks.

Figure 1-1 illustrates the UDP/IP protocol layer, comprising four layers: Application, Transport, Internet, and Network Access. The Network Access layer is further divided into two sublayers, Link and Physical, to link them with the hardware implementation using an FPGA.

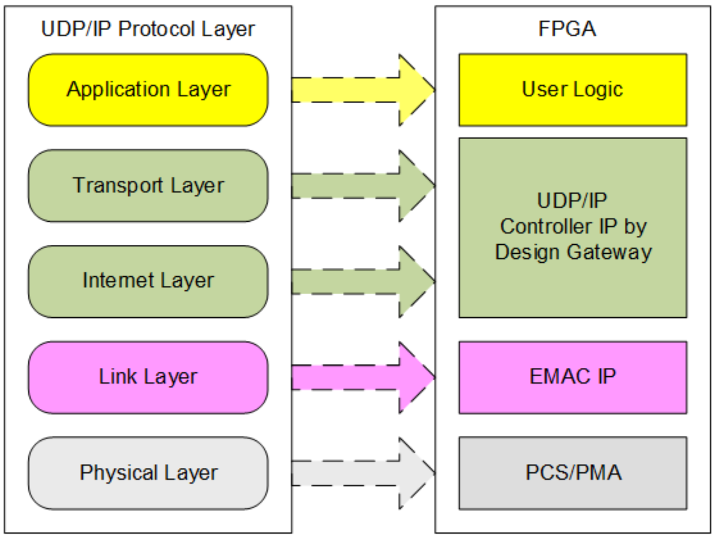


Figure 1-1 UDP/IP Protocol Layer

Typically, the Link and Physical layers are implemented using Xilinx IP cores, which provide the IP suite with Ethernet MAC, PCS, and PMA functionalities. For 40G Ethernet, this is accomplished using the “40G/50G Ethernet Subsystem”.

The UDP40G-IP implements the Transport and Internet layer of UDP/IP protocol using full hardwire logic, without the need for a CPU or DDR. This allows the user logic to be designed for processing the UDP payload data at the user interface of UDP40G-IP. The UDP40G-IP is responsible for building an Ethernet packet that encapsulates the UDP payload data from the user and transmitting it to the Ethernet MAC (EMAC). If the user data size exceeds the maximum size of Ethernet packet, the UDP40G-IP will partition the data into multiple packets for transmission. To form a complete Ethernet packet, the UDP40G-IP must process and append the UDP/IP header before transmission. On the other hand, when the UDP40G-IP receives an Ethernet packet from the EMAC, it extracts and verifies the packet. Valid packets result in the UDP40G-IP extracting the UDP payload data forwarding it to the user logic. However, invalid packets are rejected.

This reference design comprises simple user logic, UDP40G-IP, and 40G Ethernet Subsystem designed for data transfer using the UDP/IP protocol. Data transfer can occur with two targets, PC or other FPGA integrating UDP40G-IP. Design Gateway provides a test application on the PC, named “udpdatatest”, facilitating the data transfer using a single UDP session. This application allows for the configuration of data direction (send or receive).

To provide flexibility during testing, users can adjust test parameters and control the operation of the UDP40G-IP demo via UART, integrated with a CPU system. Users can monitor current status and configure test parameters through the console. The CPU firmware is developed using a simple bare-metal OS. Further details of the demo are outlined below.

2 Hardware design

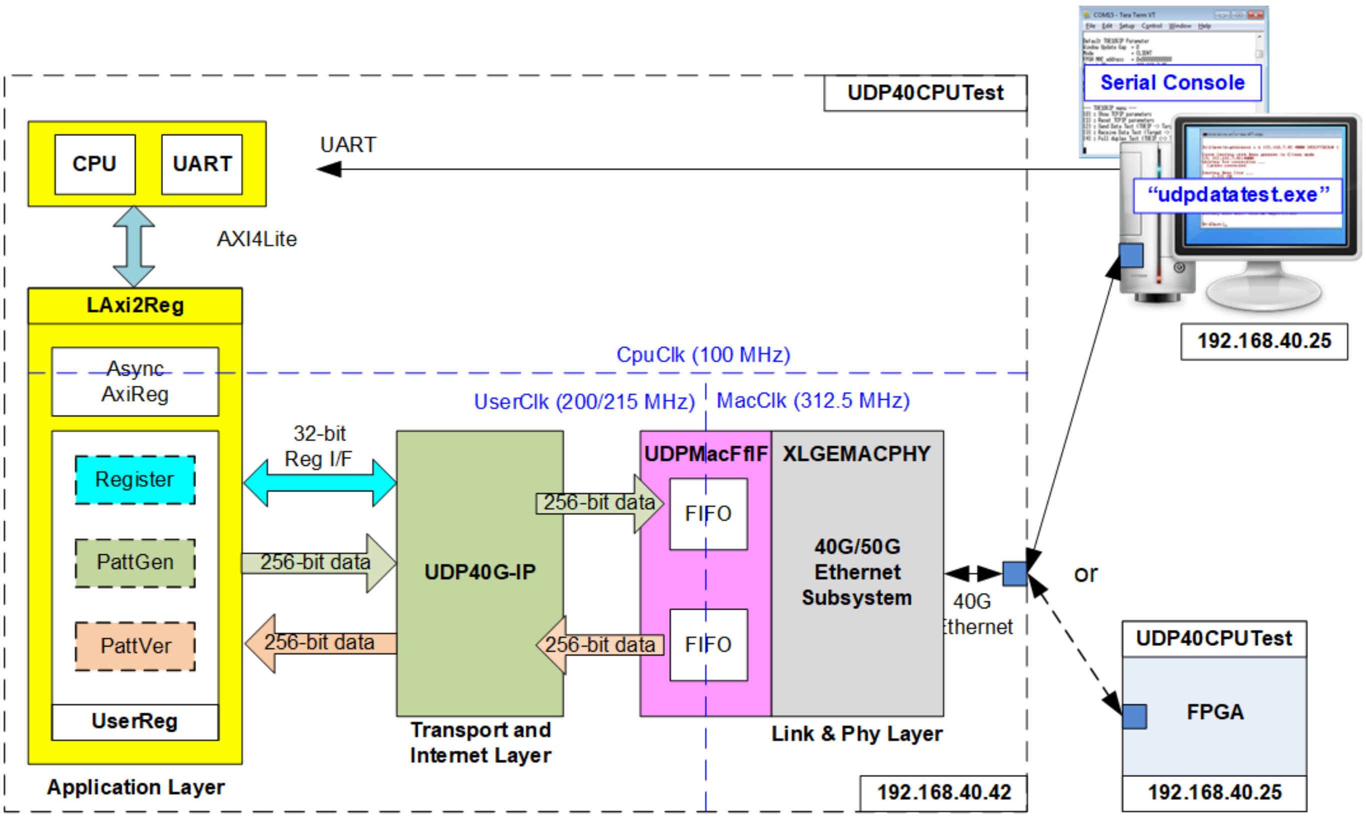


Figure 2-1 Demo block diagram

In the test environment, two devices are used to transfer data over a 40G Ethernet connection. The first device is an FPGA initialized in Client mode while the second device, which can be a PC or another FPGA, is initialized in Server mode. When employing two FPGAs, additional initialization options are available: Client <-> Fixed-MAC or Fixed-MAC <-> Fixed-MAC. An application called “udpdatatest” facilitates data transfer using the UDP/IP protocol between the PC and FPGA.

Within the FPGA system, the UDP40G-IP interface with the 40G/50G Ethernet Subsystem to handle all UDP/IP players. To establish the connection between the UDP40G-IP and the 40G/50G Ethernet Subsystem, an adapter logic named UDPMacFfIF is employed. This adapter converts the 256-bit MACFIFO (FWFT) interface of the UDP40G-IP to the 256-bit AXI4-ST interface of the 40G/50G Ethernet Subsystem. Additionally, this adapter facilitates data stream transfer across clock domains, bridging between the UserClk and MacClk domains.

The user interface of the UDP40G-IP is connected to UserReg within LAxi2Reg, comprising a Register file for interfacing with the Register interface, PattGen for transmitting test data via the Tx FIFO interface, and PattVer for verifying test data via the Rx FIFO interface. The Register file of UserReg is controlled by CPU firmware through the AXI4-Lite bus.

The test design incorporates three distinct clock domains: CpuClk for the CPU system, MacClk for interfacing with the 40G/50G Ethernet Subsystem, and UserClk for the user logic of the UDP40G-IP. As a result, AsyncAxiReg is designed to support asynchronous signal transmission between CpuClk and UserClk. Additional details about each module within the UDP40CPUTest are provided below.

Note:
1) UserClk can be adjusted to utilize the same clock as CpuClk to optimize clock resource.
2) The recommended frequency for the clock input of UDP40G-IP is 200 MHz or higher.

2.1 40G/50G Ethernet Subsystem

The 40G/50G Ethernet Subsystem comprises the MAC, PCS, and PMA layers, facilitating communication with the target device. This module is provided by Xilinx with the following configurations.

• Data Path Interface	: 256-bit Regular AXI4-Stream
• BASE R/KR Standard	: BASE-R standard
• Control and Statistics Interface	: Control and Status Vectors

Further information about the 40G/50G Ethernet Subsystem can be accessed via the following link.
https://www.xilinx.com/products/intellectual-property/ef-di-50gemac.html

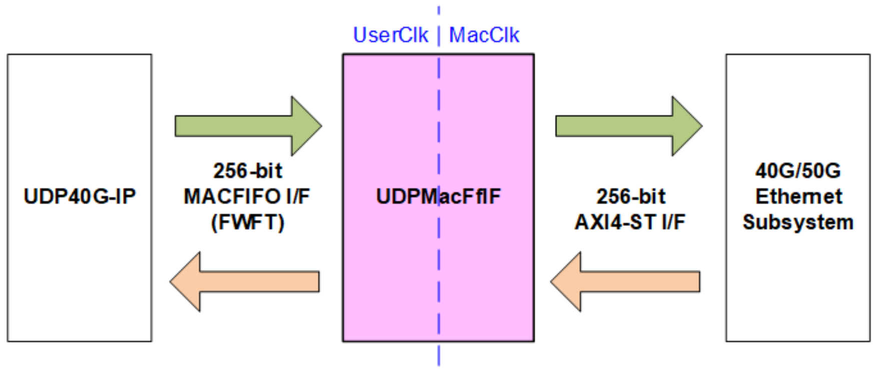


Figure 2-2 An adapter logic for connecting UDP40G-IP to 40G/50G Ethernet Subsystem

The data path interface of the 40G/50G Ethernet Subsystem does not include FIFO logic, resulting in different user clock domains for the Transmit and Receive interfaces. To address this, an adapter logic named UDPMacFfIF has been developed. This logic incorporates two asynchronous FIFOs for clock domain crossing between UserClk and MacClk. Additionally, small logics are integrated to convert the interface from 256-bit FWFT FIFO I/F to 256-bit AXI4-ST I/F. Further details of UDPMacFfIF are provided below.

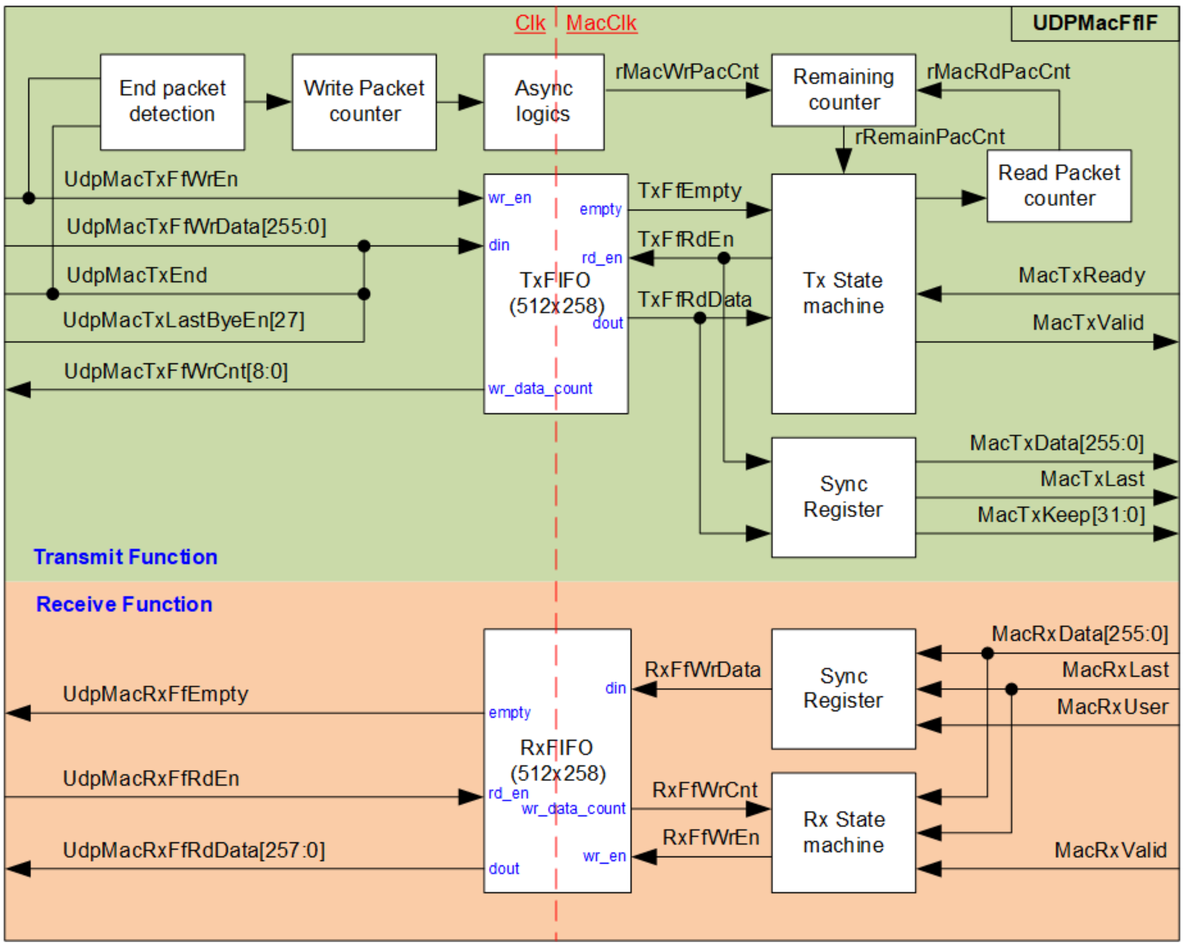


Figure 2-3 UDPMacFfIF block diagram

The TxFIFO and RxFIFO are configured as asynchronous types capable of storing 258 bits of data with a depth size of 512. Employing a 512x258-bit FIFO is sufficient for storing a maximum Ethernet packet size of 9014 bytes. Both FIFOs are set to be FWFT type, enabling valid read data to be available simultaneously with the assertion of the read enable signal.

According to the UDP40G-IP datasheet, transmit packets output from the UDP40G-IP necessitate continuous transfer from the first data of packet to the last. This differs from the typical characteristic of the Tx user interface of Ethernet MAC, which generally transfers one data every two clock cycles. The utilization of FIFOs addresses this difference in the Transmit interface characteristics. Conversely, continuous data transfer is not required for the receive interface of both UDP40G-IP and Ethernet MAC.

Figure 2-3 illustrates separate logics for packet transmission and reception, enabling simultaneously processing and transfer of packets in both directions. Further details regarding the logics for packet transmission and reception are outlined below.

<u>Transmit Function</u>

The transmit packets output from the UDP40G-IP are stored into the TxFIFO. The write enable, write data, and write counter of the TxFIFO are directly connected to the UDP40G-IP. To optimize data size, 258 bits of data are utilized to store 256 bits of data, the last flag of the packet, and bit27 of the byte enable. According to the UDP40G-IP, the byte enable can be equal to two values: 0x0FFF_FFFF and 0x0000_03FF, which can be distinguished using only bit27.

During the writing of data stream to the TxFIFO, the 'End packet detection' monitors the last flag of the packet (UdpMacTxEnd). Upon detecting the last data of the packet, the 'Write Packet counter' increments its value to indicate the number of packets stored in the TxFIFO. This updated counter value is then transferred from the Clk domain to the MacClk domain via 'Async logics', initiating the packet transmission from the TxFIFO to the Ethernet MAC.

The Read interface of the TxFIFO is controlled by Tx state machine. The number of write packets to the TxFIFO (rMacWrPacCnt) and the number of read packets from the TxFIFO (rMacRdPacCnt) are used to compute the available packets in the TxFIFO (rRemainPacCnt). If at least one packet is available, the Tx State machine begins reading the packet from the TxFIFO to transmit it to the Ethernet MAC. After completing the packet transmission to the Ethernet MAC, the 'Read Packet counter' increases its value to update the number of read packets.

The operation of the Read interface of the TxFIFO is illustrated using a timing diagram in Figure 2-4.

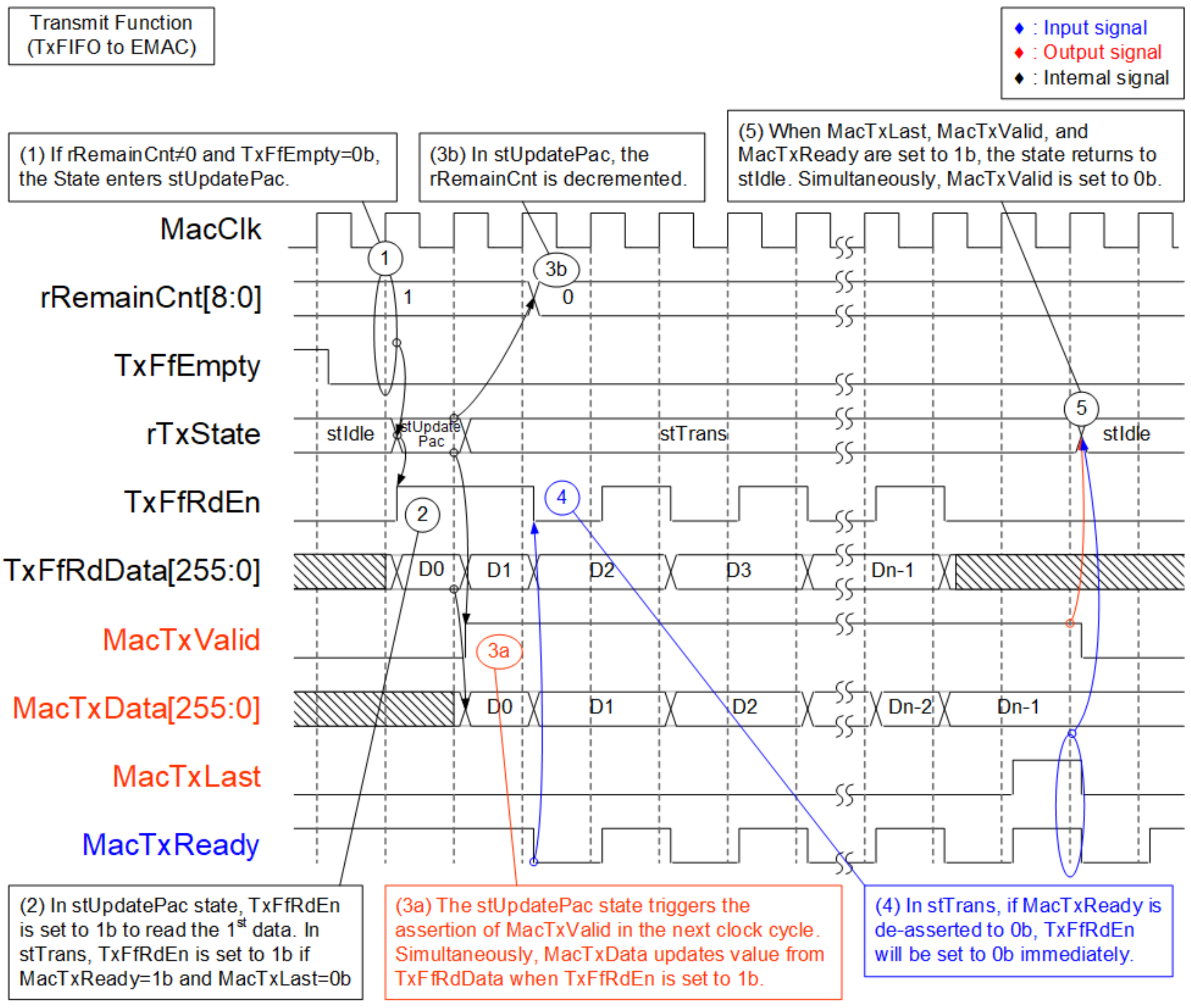

Transmit Function (TxFIFO to EMAC)
♦ : Input signal
♦ : Output signal
♦ : Internal signal
(1) If rRemainCnt≠0 and TxFfEmpty=0b, the State enters stUpdatePac.
(3b) In stUpdatePac, the rRemainCnt is decremented.
(5) When MacTxLast, MacTxValid, and MacTxReady are set to 1b, the state returns to stIdle. Simultaneously, MacTxValid is set to 0b.
MacClk
rRemainCnt[8:0]
TxFfEmpty
rTxState
TxFfRdEn
TxFfRdData[255:0]
MacTxValid
MacTxData[255:0]
MacTxLast
MacTxReady
1
3b
0
stIdle
stUpdatePac
stTrans
stIdle
5
4
2
D0
D1
D2
D3
Dn-1
3a
D0
D1
D2
Dn-2
Dn-1
(2) In stUpdatePac state, TxFfRdEn is set to 1b to read the 1st data. In stTrans, TxFfRdEn is set to 1b if MacTxReady=1b and MacTxLast=0b
(3a) The stUpdatePac state triggers the assertion of MacTxValid in the next clock cycle. Simultaneously, MacTxData updates value from TxFfRdData when TxFfRdEn is set to 1b.
(4) In stTrans, if MacTxReady is de-asserted to 0b, TxFfRdEn will be set to 0b immediately.

Figure 2-4 Read interface of the TxFIFO timing diagram

1) When at least one packet is available in the TxFIFO (rRemainCnt≠0 and TxFfEmpty is set to 0b), the state transitions to stUpdatePac to initiate the forwarding of the packet from the TxFIFO to the Ethernet MAC.
2) The state remains in stUpdatePac for a single clock cycle. During this state, TxFfRdEn is set to 1b to read the first data of the packet (D0) from the TxFIFO.
3) Subsequently, the state enters stTrans to continue transferring the remaining data of the packet. MacTxValid is set to 1b to initiate the packet transfer to the Ethernet MAC, remaining asserted until the last data of the packet (Dn-1) is completely transmitted. At the same time, MacTxData loads the first data of the packet from TxFfRdData. MacTxData updates its value only when TxFfRdEn is set to 1b, indicating the validity of TxFfRdData. Additionally, in the stUpdatePac state, the 'Read Packet counter' is incremented, resulting to the rRemainCnt being decremented.
4) During packet transmission in the stTrans state, if MacTxReady is set to 0b, TxFfRdEn is set to 0b to pause packet transmission and maintain the value of MacTxData.
5) Once the last data of the packet is completely transmitted, indicated by MacTxValid, MacTxLast, and MacTxReady being set to 1b, the state returns to stIdle to proceed with the transmission of the subsequent packet.

Receive Function

When the new packet is received from the Ethernet MAC, the Rx state machine ensures that the remaining space of RxFIFO is sufficient for storing maximum Ethernet packet size, 9014 bytes, by reading the RxFIFO data counter, RxFfWrCnt. The packet is stored to RxFIFO if there is enough free space. Otherwise, the Rx state machine discards the packet and not store it to the RxFIFO.

The Read interface of RxFIFO can be directly connected to the UDP40G-IP, so this section describes only the Write interface of RxFIFO for both accepted and discarded packet conditions.

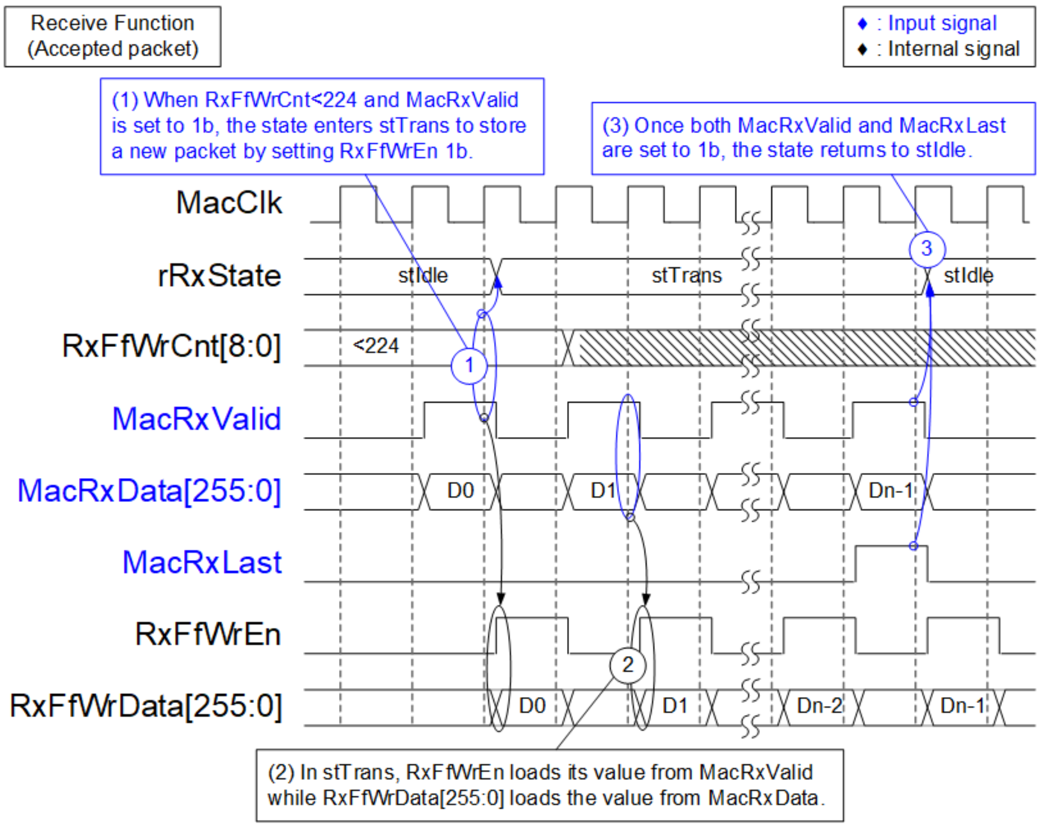

Receive Function (Accepted packet)
♦ : Input signal
♦ : Internal signal
(1) When RxFfWrCnt<224 and MacRxValid is set to 1b, the state enters stTrans to store a new packet by setting RxFfWrEn 1b.
(3) Once both MacRxValid and MacRxLast are set to 1b, the state returns to stIdle.
MacClk
rRxState
stIdle
stTrans
stIdle
RxFfWrCnt[8:0]
<224
MacRxValid
MacRxData[255:0]
D0
D1
Dn-1
MacRxLast
RxFfWrEn
RxFfWrData[255:0]
D0
D1
Dn-2
Dn-1
1
2
3
(2) In stTrans, RxFfWrEn loads its value from MacRxValid while RxFfWrData[255:0] loads the value from MacRxData.

Figure 2-5 Write interface of the RxFIFO timing diagram when the packet is accepted

1) Before initiating packet reception from the Ethernet MAC, the state verifies that there is sufficient free space in the RxFIFO to store a 9KB packet. This is confirmed by ensuring that the value of RxFfWrCnt is less than 224. If this condition is met, the state transitions to stTrans, and the packet is stored in the RxFIFO by setting RxFfWrEn to 1b. The value of RxFfWrData[255:0] are loaded from 256 bits of MacRxData.
2) During packet transmission in the stTrans state, RxFfWrEn is determined by MacRxValid, while the 258 bits of RxFfWrData are loaded from t 256 bits of MacRxData, MacRxLast, and MacRxUser.
3) Upon receiving the last data of the packet, indicated by both MacRxValid and MacRxLast being asserted to 1b, the state returns to stIdle to process the subsequent packet.

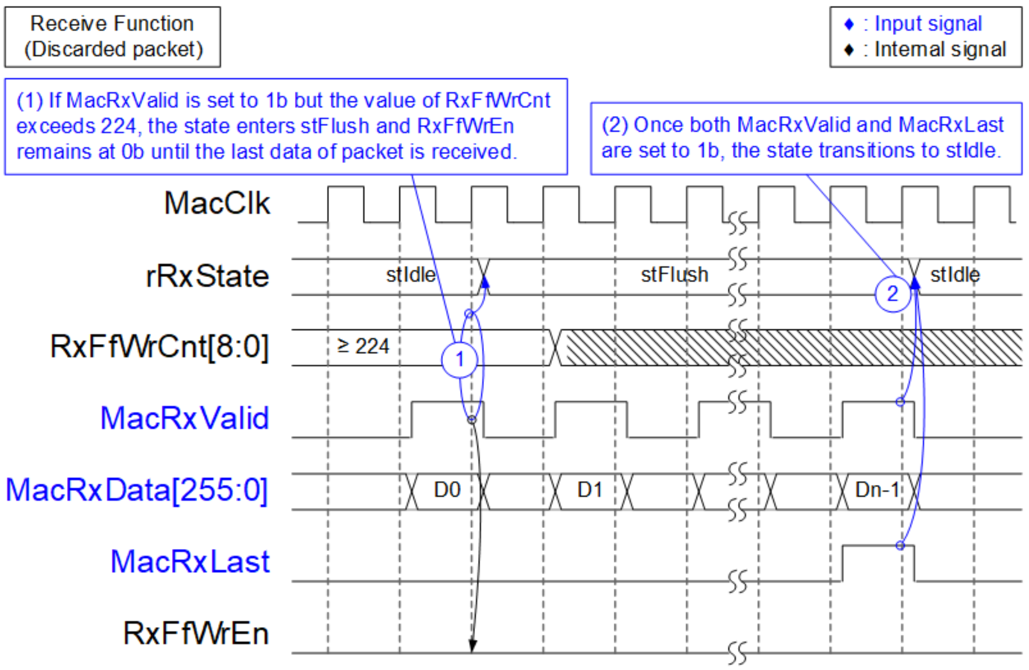

Receive Function
(Discarded packet)
♦ : Input signal
♦ : Internal signal
(1) If MacRxValid is set to 1b but the value of RxFfWrCnt exceeds 224, the state enters stFlush and RxFfWrEn remains at 0b until the last data of packet is received.
(2) Once both MacRxValid and MacRxLast are set to 1b, the state transitions to stIdle.
MacClk
rRxState
stIdle
stFlush
stIdle
RxFfWrCnt[8:0]
≥ 224
MacRxValid
MacRxData[255:0]
D0
D1
Dn-1
MacRxLast
RxFfWrEn
1
2

Figure 2-6 Write interface of the RxFIFO timing diagram when the packet is discarded

1) Upon receiving a new packet, indicated by MacRxValid being set to 1b, if the remaining space in the RxFIFO is insufficient to store a packet with the maximum packet size, the state transitions to stFlush. In this state, RxFfWrEn remained at 0b until the last data of the packet is received.
2) After the completion of packet reception, the state returns to stIdle to process the next packet.

2.2 UDP40G-IP

UDP40G-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/UDP40G-IP/dg_udp40gip_data_sheet_xilinx_en/

2.3 CPU and Peripherals

The CPU system uses a 32-bit AXI4-Lite bus as the interface to access peripherals such as the Timer and UART. The system also integrates an additional peripheral to access the test logic by assigning a unique base address and address range. To support CPU read and write operations, the hardware logic must comply with the AXI4-Lite bus standard. LAxi2Reg module, as shown in Figure 2-7, is designed to connect the CPU system via the AXI4-Lite interface, in compliance with the standard.

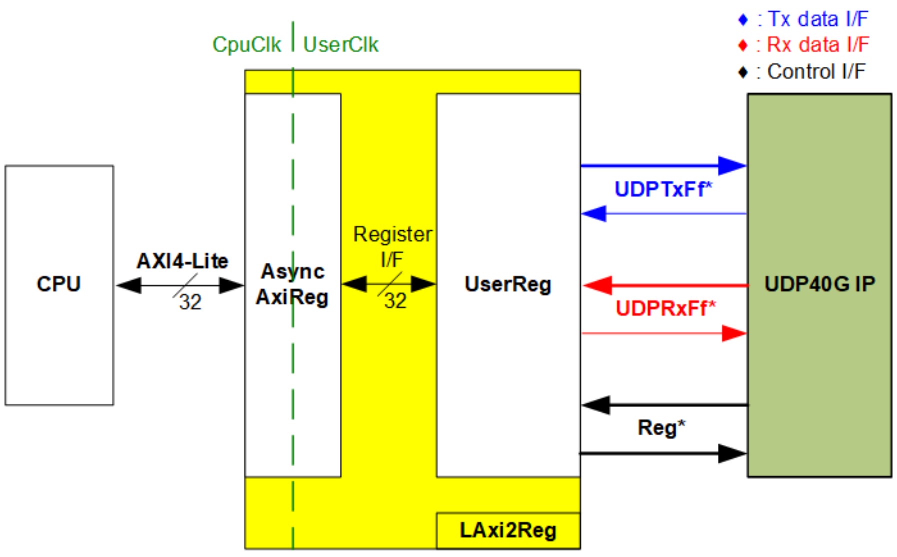


Figure 2-7 LAxi2Reg block diagram

LAxi2Reg consists of AsyncAxiReg and UserReg. AsyncAxiReg converts AXI4-Lite signals into a simple Register interface with a 32-bit data bus size, similar to the AXI4-Lite data bus size. It also includes asynchronous logic to handle clock domain crossing between the CpuClk and UserClk domains.

UserReg includes the Register file of the parameters and the status signals of the test logics. Both the data interface and control interface of UDP40G-IP are connected to UserReg. Further details of AsyncAxiReg and UserReg are described below.

2.3.1 AsyncAxiReg

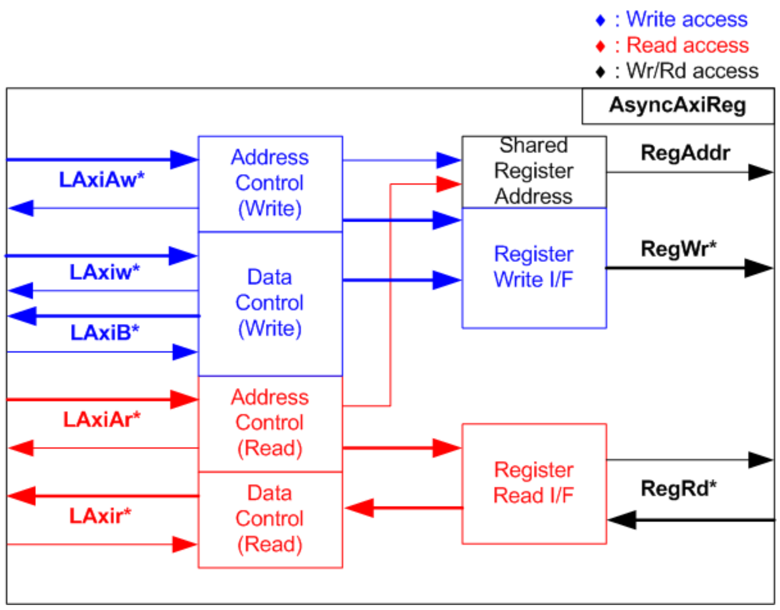


Figure 2-8 AsyncAxiReg Interface

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 channel operate independently for both control and data interfaces. Therefore, the logic in the AsyncAxiReg module to interface 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 2-8. 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 2-9.

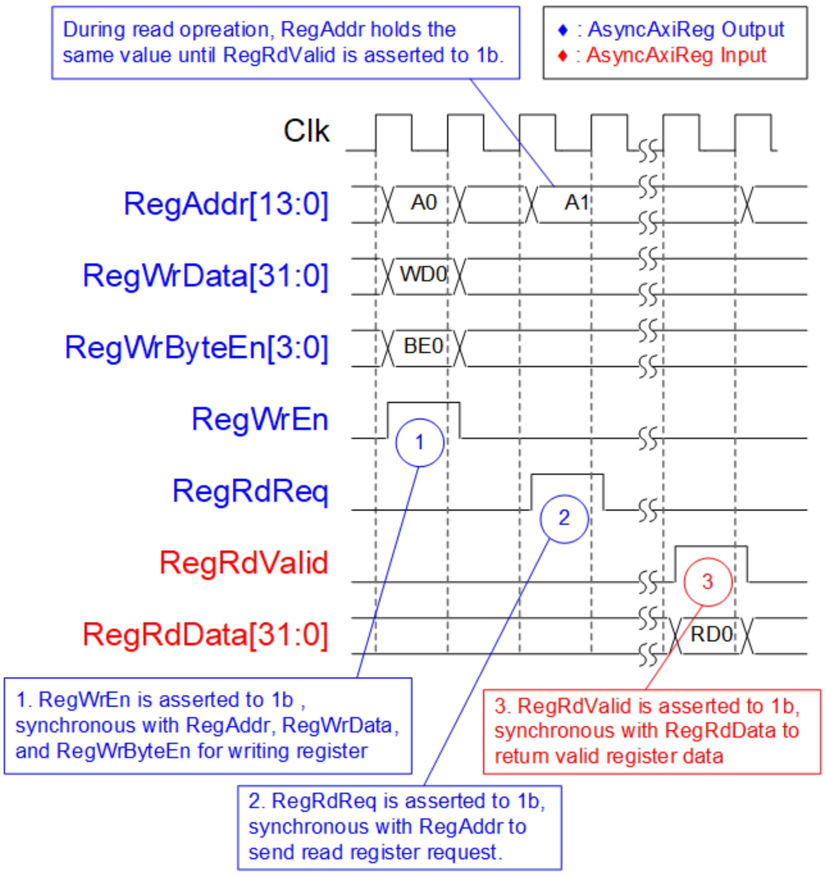

During read opreation, RegAddr holds the same value until RegRdValid is asserted to 1b.
♦ : AsyncAxiReg Output
♦ : AsyncAxiReg Input
Clk
RegAddr[13:0]
A0
A1
RegWrData[31:0]
WD0
RegWrByteEn[3:0]
BE0
RegWrEn
RegRdReq
RegRdValid
RegRdData[31:0]
RD0
1
2
3
1. RegWrEn is asserted to 1b , synchronous with RegAddr, RegWrData, and RegWrByteEn for writing register
3. RegRdValid is asserted to 1b, synchronous with RegRdData to return valid register data
2. RegRdReq is asserted to 1b, synchronous with RegAddr to send read register request.

Figure 2-9 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.

2.3.2 UserReg

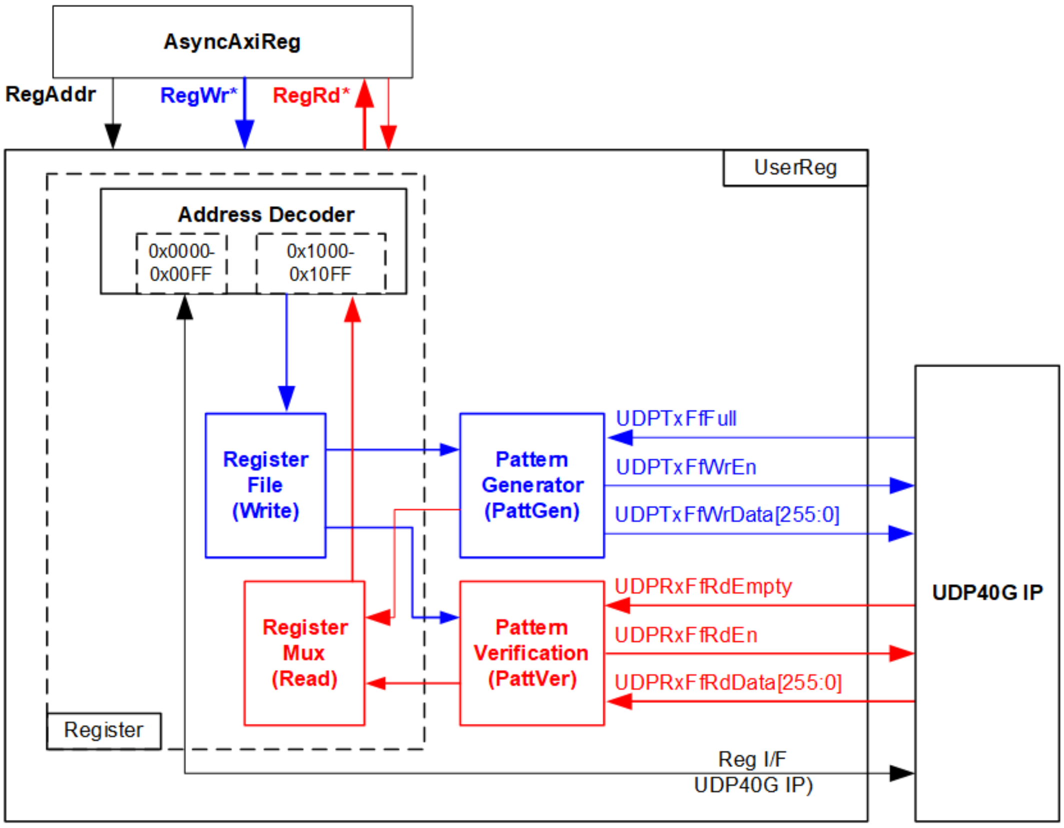


Figure 2-10 UserReg block diagram

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 256-bit test data to UDP40G-IP following FIFO interface standard, while the PattVer block is designed to read and verify 256-bit data from UDP40G-IP following FIFO interface standard.

Register Block

The address range, mapped to UserReg, is split into two areas: UDP40G-IP registers (0x0000-0x00FF) and UserReg registers (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 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 "udp40gtest.c")	Description
BA+0x0000 – BA+0x00FF: UDP40G-IP Register Area More details of each register are described in UDP40G-IP datasheet.		
BA+0x0000	UDP_RST_INTREG	Mapped to RST register within UDP40G-IP
BA+0x0004	UDP_CMD_INTREG	Mapped to CMD register within UDP40G-IP
BA+0x0008	UDP_SML_INTREG	Mapped to SML register within UDP40G-IP
BA+0x000C	UDP_SMH_INTREG	Mapped to SMH register within UDP40G-IP
BA+0x0010	UDP_DIP_INTREG	Mapped to DIP register within UDP40G-IP
BA+0x0014	UDP_SIP_INTREG	Mapped to SIP register within UDP40G-IP
BA+0x0018	UDP_DPN_INTREG	Mapped to DPN register within UDP40G-IP
BA+0x001C	UDP_SPN_INTREG	Mapped to SPN register within UDP40G-IP
BA+0x0020	UDP_TDL_INTREG	Mapped to TDL register within UDP40G-IP
BA+0x0024	UDP_TMO_INTREG	Mapped to TMO register within UDP40G-IP
BA+0x0028	UDP_PKL_INTREG	Mapped to PKL register within UDP40G-IP
BA+0x0034	UDP_SRV_INTREG	Mapped to TDH register within UDP40G-IP
BA+0x0038	UDP_RST_INTREG	Mapped to SRV register within UDP40G-IP
BA+0x003C	UDP_VER_INTREG	Mapped to VER register within UDP40G-IP
BA+0x0040	UDP_DML_INTREG	Mapped to DML register within UDP40G-IP
BA+0x0044	UDP_DMH_INTREG	Mapped to DMH register within UDP40G-IP
BA+0x1000 – BA+0x10FF: UserReg control/status		
BA+0x1000 Wr/Rd	Total transmit length (Low) (USER_TXLENL_INTREG)	Wr [31:0] – 32 lower bits of 43-bit total transmit length in 256-bit unit. Valid from 1-0x7FF_FFFF_FFFF. Rd [31:0] – 32 lower bits of 43-bit current transmit length in 256-bit unit. The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x1004 Wr/Rd	Total transmit length (High) (USER_TXLENH_INTREG)	Wr [10:0] – 11 upper bits of 43-bit total transmit length in 256-bit unit. Rd [10:0] – 11 upper bits of 43-bit current transmit length in 256-bit unit.
BA+0x1008 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+0x100C 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] – Latch 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 from Ethernet MAC (0b: Link down, 1b: Link up)
BA+0x1010 Rd	FIFO status (USER_FFSTS_INTREG)	Rd [4:0]: Mapped to UDPRxFfLastRdCnt signal of UDP40G-IP [15:5]: Mapped to UDPRxFfRdCnt signal of UDP40G-IP [24]: Mapped to UDPTxFfFull signal of UDP40G-IP
BA+0x1014 Rd	Total receive length (low) (USER_RXLENL_INTREG)	Rd [31:0] – 32 lower bits of 43-bit current received size in 256-bit unit. The value is cleared to 0 when USER_CMD_REG is written by user.
BA+0x1018 Rd	Total receive length (high) (USER_RXLENH_INTREG)	Rd [10:0] – 11 upper bits of 43-bit current received size in 256-bit unit.
BA+0x1040 Wr/Rd	Error from buff overflow (USER_ERRINT_INTREG)	Wr [0] – Set 1b to clear read value of USER_ERRINT_INTREG[0] to 0b. Rd [0] – Latched value of the interrupt output from UDPMacFfIF, asserted when the RxFIFO is overflow and the received packet is discarded. [31:16] – The number of received packets which are discarded.

Pattern Generator

The logic diagram and timing diagram of Pattern Generator (PattGen) are illustrated in Figure 2-11 and Figure 2-12, respectively.

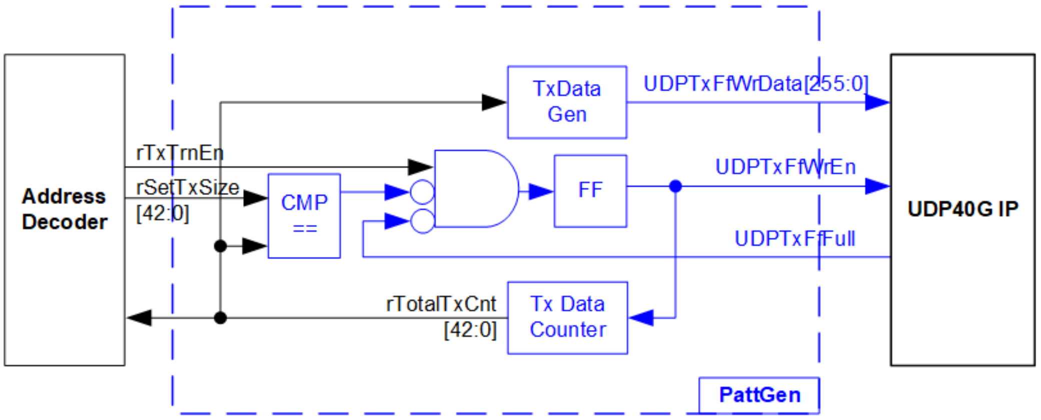


Figure 2-11 PattGen block

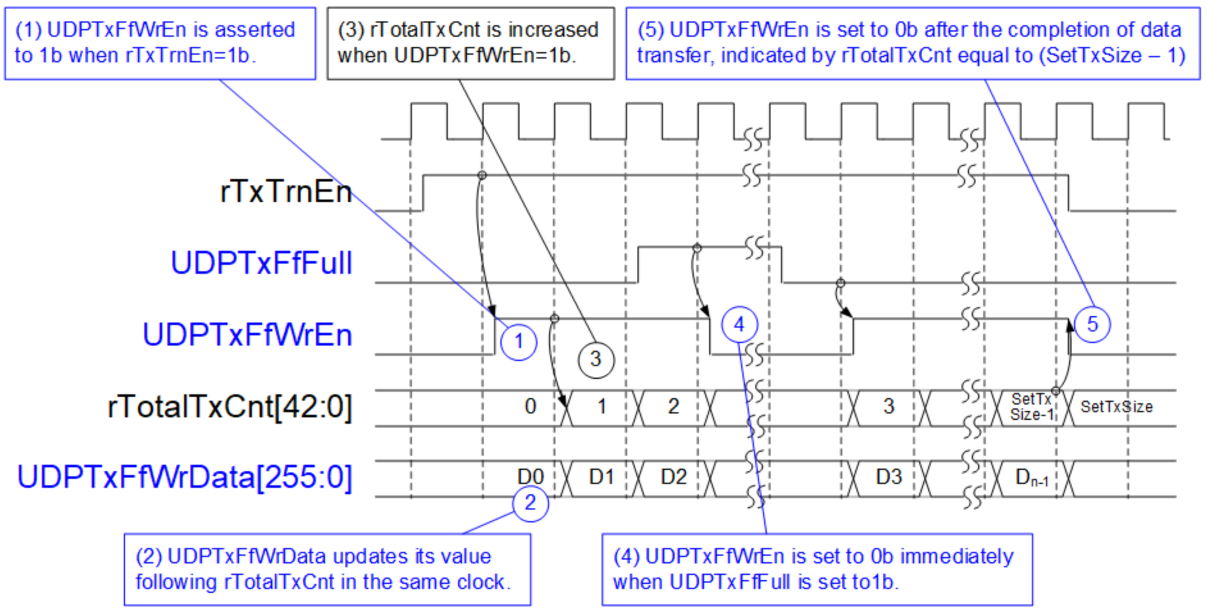

(1) UDPTxFfWrEn is asserted to 1b when rTxTrnEn=1b.
(3) rTotalTxCnt is increased when UDPTxFfWrEn=1b.
(5) UDPTxFfWrEn is set to 0b after the completion of data transfer, indicated by rTotalTxCnt equal to (SetTxSize – 1)
rTxTrnEn
UDPTxFfFull
UDPTxFfWrEn
rTotalTxCnt[42:0]
UDPTxFfWrData[255:0]
0
1
2
3
SetTx Size-1
SetTxSize
D0
D1
D2
D3
Dn-1
(2) UDPTxFfWrData updates its value following rTotalTxCnt in the same clock.
(4) UDPTxFfWrEn is set to 0b immediately when UDPTxFfFull is set to1b.

Figure 2-12 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 UDP40G-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 2-13 and Figure 2-14, respectively. The verification feature is executed when the verification flag (rVerifyEn) is enabled.

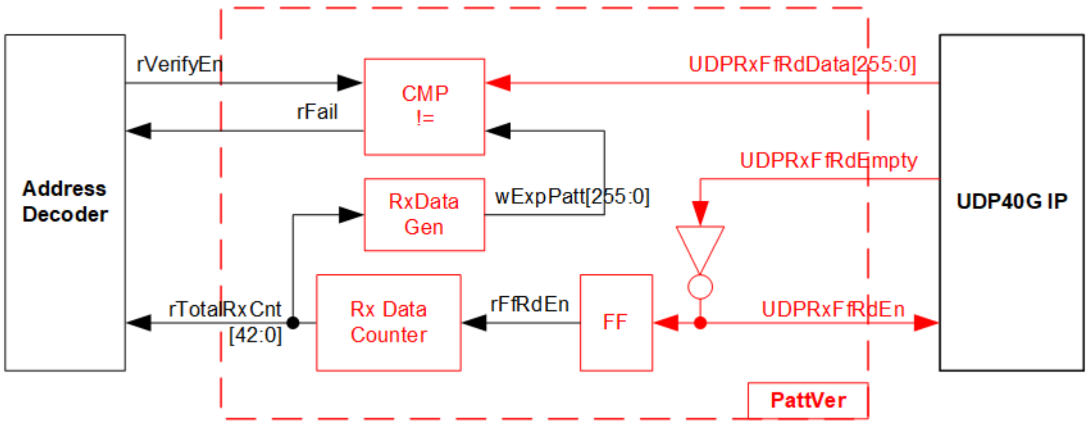


Figure 2-13 PattVer block

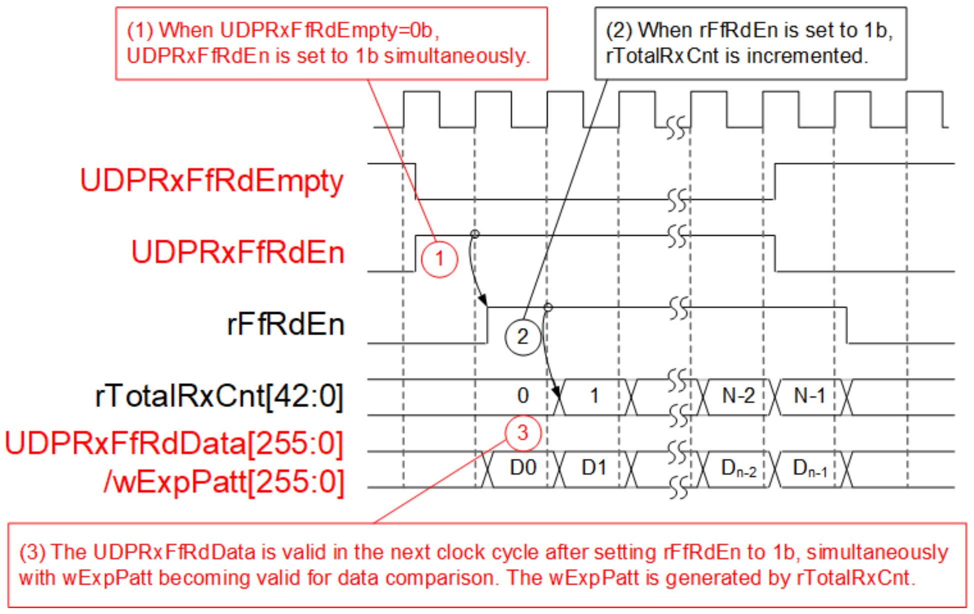
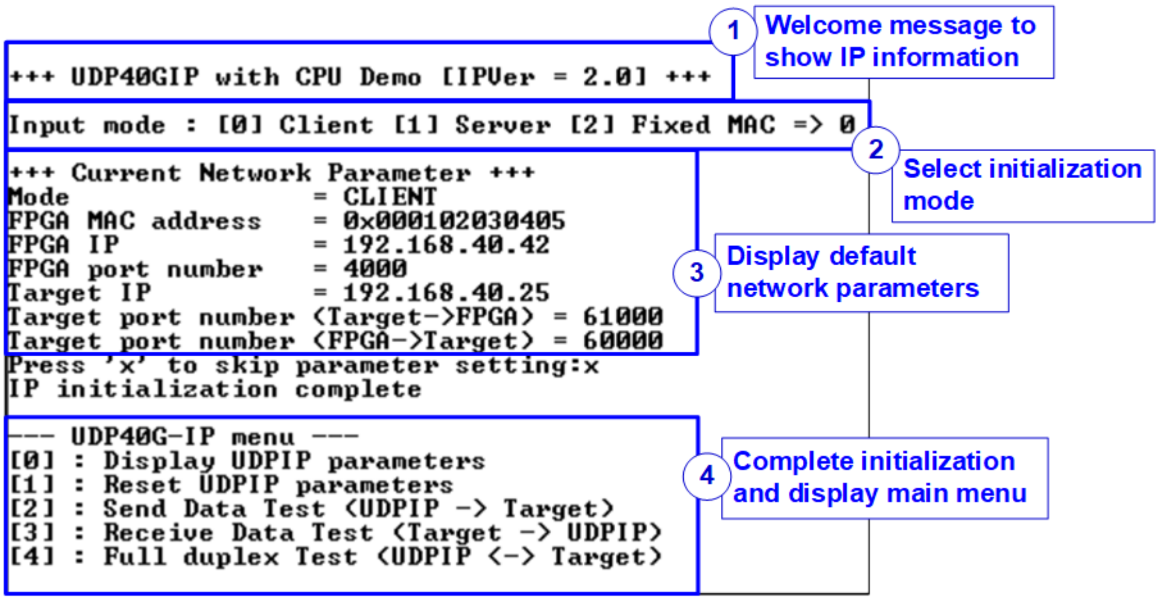

(1) When UDPRxFfRdEmpty=0b, UDPRxFfRdEn is set to 1b simultaneously.
(2) When rFfRdEn is set to 1b, rTotalRxCnt is incremented.
UDPRxFfRdEmpty
UDPRxFfRdEn
rFfRdEn
rTotalRxCnt[42:0]
UDPRxFfRdData[255:0] /wExpPatt[255:0]
1
2
3
0
1
N-2
N-1
D0
D1
Dn-2
Dn-1
(3) The UDPRxFfRdData is valid in the next clock cycle after setting rFfRdEn to 1b, simultaneously with wExpPatt becoming valid for data comparison. The wExpPatt is generated by rTotalRxCnt.

Figure 2-14 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 Sequence (FPGA)

The reference design employs a bare-metal OS for the CPU firmware operating, which facilitates hardware handling. When executing the test system, the first step involves hardware initialization, as outlined below.



```
+++ UDP40GIP with CPU Demo [IPVer = 2.0] +++
Input mode : [0] Client [1] Server [2] Fixed MAC => 0
+++ Current Network Parameter +++
Mode                      = CLIENT
FPGA MAC address          = 0x000102030405
FPGA IP                   = 192.168.40.42
FPGA port number          = 4000
Target IP                 = 192.168.40.25
Target port number (Target->FPGA) = 61000
Target port number (FPGA->Target) = 60000
Press 'x' to skip parameter setting:x
IP initialization complete
--- UDP40G-IP menu ---
[0] : Display UDPIP parameters
[1] : Reset UDPIP parameters
[2] : Send Data Test (UDPIP -> Target)
[3] : Receive Data Test (Target -> UDPIP)
[4] : Full duplex Test (UDPIP <-> Target)
```

1 Welcome message to show IP information
2 Select initialization mode
3 Display default network parameters
4 Complete initialization and display main menu

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

Figure 3-1 illustrates the four-step process for hardware initialization.

1) Upon FPGA boot-up, the firmware polls the Ethernet link status until it is established, as indicated by USER_RST_INTREG[16] being set to 1b. Upen successful link establishment, the console displays the information of the IP.
2) The console then displays an option menu to set the initialization mode of UDP40G-IP, including Client, Server, and Fixed-MAC mode. The selection of the initialization mode is guided by the following considerations.

   i) Three initialization modes determine how the MAC address of the target device is obtained: from a received ARP reply packet in Client mode, from a received ARP request packet in Server mode, or from a user-defined value in Fixed-MAC mode.
   ii) When the target device is PC, selecting the Client mode simplifies the setup.
   iii) In FPGA – FPGA test scenarios, initialization mode can be set to Client <-> Server, Fixed-MAC <-> Fixed MAC, and Client <-> Fixed-MAC.

3) Upon confirming the initialization mode selection, the console displays default values for network parameters corresponding to the chosen mode. These parameters include initialization mode, FPGA MAC address, FPGA IP address, FPGA port number, Target IP address, and Target port number. There are two default parameter sets: Server parameter set (used for Server mode only) and Client parameter set (used for both Client and Fixed-MAC mode). For Fixed-MAC mode, an additional parameter, Target MAC address, is displayed. Users have the option to proceed with initialization using default parameters or to customize certain parameters before starting the initialization process. Further instructions to update parameter are provided in the Reset parameters menu (section 3.2).
4) The completion of the initialization process is indicated by UDP_CMD_INTERG[0] being de-asserted to 0b. At this point, the console shows "IP initialization complete" and presents the test options of the main menu, comprising five choices. Each main menu option is elaborate upon in subsequent sections.

3.1 Display parameters

This menu option displays the current value of all UDP40G-IP parameters. The following steps outline the process for displaying parameters.

1) Retrieve the initialization mode.
2) Read all network parameters from each variable in the firmware according to the initialization mode. These parameters include the FPGA MAC address, FPGA IP address, FPGA port number, Target MAC address (only displayed in Fixed-MAC mode), Target IP address, and Target port number.
   Note: The FPGA parameters refer to the host parameters defined for the UDP40G-IP, while the Target parameters refer to the parameters assigned for a PC or another FPGA.
3) Print out each variable.

3.2 Reset parameters

This menu option allows user to modify certain UDP40G-IP parameters, such as the IP address and port number for the FPGA. Upon updating specific parameters, the IP is reset and re-initialized with the new values. The CPU then waits until the initialization is completed. The following steps outline the procedure for resetting parameters.

1) Display all parameters on the console, similar to described in section 3.1 (Display parameters).
2) If the user chooses to use the default value, proceed to the next step. Otherwise, display the menu for setting all parameters.
   i) Obtain the initialization mode from the user. If the initialization mode is changed, display the latest parameter set for the new mode on the console.
   ii) Receive the remaining parameters from the user and validate all inputs. If any input is invalid, the corresponding parameter is not updated.
3) Force reset of the UDP40G-IP by setting UDP_RST_INTREG[0] to 1b.
4) Set all parameters to the UDP40G-IP registers, such as UDP_SML_INTREG and UDP_DIP_INTREG.
5) De-assert the UDP40G-IP reset by setting UDP_RST_INTREG[0] to 0b to initiate the initialization process.
6) Reset the PattGen and PattVer logics by setting USER_RST_INTREG[0] to 1b.
7) Monitor the UDP40G-IP busy flag (UDP_CMD_INTREG[0]) until the initialization process is completed (the busy flag is de-asserted to 0b).

3.3 Send data test

This menu option allows the user to execute the Send data test. Users can set the parameters such as the total transmit length. Upon validating all inputs, the data is transferred using 32-bit incremental test data. The operation is considered completed when all data has been transferred. The following steps outline the procedure for sending data.

1) Receive the transfer size and packet size from the user and verify the validity of all inputs. If any input is found to be invalid, the operation is cancelled.
2) Configure the UserReg registers as follows: set the transfer size (USER_TXLEN_INTREG), set the Reset flag to clear the initial value of the test pattern (USER_RST_INTREG[0]=1b), and set the Command register to start the data pattern generator (USER_CMD_INTREG=0). The test pattern generator in UserReg then starts generating test data for UDP40G-IP.
3) Display the recommended parameters of the test application on the PC by reading the current parameters in the system. Wait for the user to press any key to start the IP sending operation.
4) Set the parameters in UDP40G-IP to begin the operation. Set the packet size to UDP_PKL_INTREG and the total size to UDP_TDL/H_INTREG. Finally, set UDP_CMD_INTREG to 1b to start sending data via IP.
5) Monitor UDP40G-IP to determine when the operation is completed by reading the busy flag of the IP (UDP_CMD_INTREG[0]) until it is set to 0b. While monitoring the busy flag, the CPU reads the current transfer size from the user logic (USER_TXLENL/H_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 option allows the user to execute the Receive data test. Users can set parameters such as the total receive length. Upon validating all inputs, the data is generated using 32-bit incremental test data for verification with the received data from the target (PC or FPGA) when the data verification is enabled. The following steps outline the procedure for receiving data.

1) Receive the total transfer size and data verification mode from the user and verify that all inputs are valid. The operation is cancelled if some inputs are invalid.
2) Set the UserReg registers, including 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 parameters (similar to Step 3 of the Send data test).
4) Wait until the total number of received data (USER_RXLENL/H_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 timeout interrupt assertion by reading USER_RST_INTREG[8], data verification fail flag by reading USER_CMD_INTREG[1] (if the verification mode is enabled), and error interrupt assertion by reading USER_ERRINT_INTREG. If any errors are found, the error message will be displayed.
6) Calculate performance and display the test result on the console.

3.5 Full duplex test

This menu option 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. The operation is considered completed once all data in both directions is completely transferred. The following steps outline the procedure for transferring data in both directions.

Note: When testing with a PC, ensure that the transfer size on the test application (udpdatatest) matches the transfer size set on the FPGA. Two instances of "udpdatatest" are executed, one for sending data and another for receiving data using different port number. When testing with two FPGAs, ensure that the port number for sending and receiving data are the same.

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 the validity of all inputs. The operation is cancelled if some inputs are invalid.
2) Set UserReg registers including the transfer size (USER_TXLENL/H_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 the 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. The CPU proceeds to the next step under one of two conditions.
   a) The user inputs any key to initiate the operation.
   b) The UDP40G-IP is initialized by the Server mode and some data is received.
4) Set UDP40G-IP registers, including the packet size (UDP_PKL_INTREG), total transfer size (UDP_TDL/H_INTREG), and the 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 the data flow of both directions simultaneously, with two tasks running during the test.
   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_TXLENL/H_INTREG) and received data size (USER_RXLENL/H_INTREG) are read and displayed on the console every second.

6) Stop the timer. Check timeout interrupt assertion by reading USER_RST_INTREG[8], data verification fail flag by reading USER_CMD_INTREG[1] (if the verification mode is enabled), and error interrupt assertion by reading USER_ERRINT_INTREG. If any errors are found, the error message will be 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 UDP40G-IP operation.

void check_ethlink(unsigned int* status)	
Parameters	status: Returned value to indicate the ethernet status. 0: Ethernet link down, 1: Ethernet link up.
Return value	None
Description	Read Ethernet link status from USER_RST_REG[16], and return the result to the 'status' parameter.

void init_param(void)	
Parameters	None
Return value	None
Description	Reset parameters as described in section 3.2. It calls the 'show_param' and 'input_param' functions to display parameters and get parameters from user.

int input_param(void)	
Parameters	None
Return value	0: Valid input, -1: Invalid input
Description	Obtain network parameters from user, including the initialization mode, FPGA MAC address, FPGA IP address, FPGA port number, Target MAC address (when operating in Fixed-MAC mode), Target IP address, and Target port numbers. If all inputs are valid, update the parameters; otherwise, retain the previous values. Once all parameters are received, call the 'show_param function to display them.

void show_cursize(void)	
Parameters	None
Return value	None
Description	Retrieve the current number of transmitted and received data by reading USER_TXLENL/H_INTREG and USER_RXLENL/H_INTREG, representing data in byte, KB, or MB.

void show_error_interrupt(void)	
Parameters	None
Return value	None
Description	Read the discarded packet count from USER_ERRINT_INTREG[31:16] and display the read value. Then, clear the error flag by setting USER_ERRINT_INTREG[0] to 1b.

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

void show_param(void)	
Parameters	None
Return value	None
Description	Display the parameters as described in section 3.1.

void show_result(void)	
Parameters	None
Return value	None
Description	Retrieve the values of USER_TXLENL/H_INTREG and USER_ RXLENL/H_REG to display the total transmitted data size and total received data size. Additionally, read the global parameters (timer_val and timer_upper_val) and calculate the total time usage to display it in usec, msec, or sec units. Finally, calculate and display the transfer performance in MB/s unit.

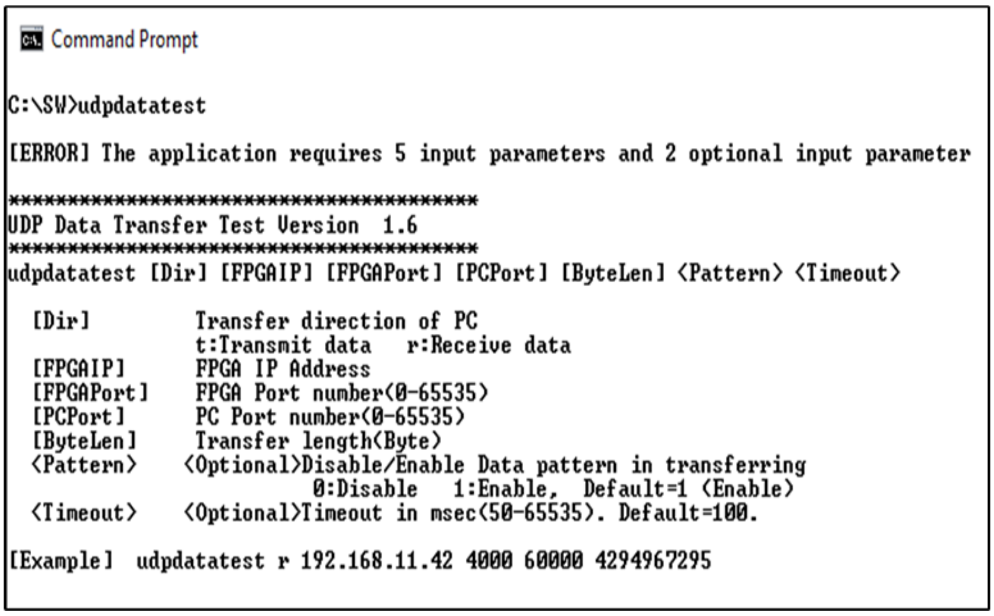
int udp_recv_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Execute the Receive data test as described in section 3.4. This involves calling the functions 'show_interrupt, 'show_error_interrupt', 'show_cursize', and 'show_result'.

int udp_send_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Execute the Send data test as described in section 3.3. This involves calling the functions 'show_cursize' and 'show_result'.

int udp_txrx_test(void)	
Parameters	None
Return value	0: The operation is successful -1: Receive invalid input or error is found
Description	Execute the Full duplex test as described in section 3.5. This involves calling the functions 'show_interrupt, 'show_error_interrupt', 'show_cursize', and 'show_result'.

void wait_ethlink(void)	
Parameters	None
Return value	None
Description	Read USER_RST_REG[16] and wait until the Ethernet connection is successfully established.

4 Test Software on PC



```
C:\SW>udpdatatest

[ERROR] The application requires 5 input parameters and 2 optional input parameter

****************************************
UDP Data Transfer Test Version  1.6
****************************************
udpdatatest [Dir] [FPGAIP] [FPGAPort] [PCPort] [ByteLen] <Pattern> <Timeout>

  [Dir]         Transfer direction of PC
                t:Transmit data   r:Receive data
  [FPGAIP]      FPGA IP Address
  [FPGAPort]    FPGA Port number(0-65535)
  [PCPort]      PC Port number(0-65535)
  [ByteLen]     Transfer length(Byte)
  <Pattern>    <Optional>Disable/Enable Data pattern in transferring
                          0:Disable   1:Enable,  Default=1 (Enable)
  <Timeout>    <Optional>Timeout in msec(50-65535). Default=100.

[Example]  udpdatatest r 192.168.11.42 4000 60000 4294967295
```

Figure 4-1 "udpdatatest" application usage

The "udpdatatest" application is utilized for sending or receiving UDP data on a PC. It requires five mandatory parameters and two optional parameters. Ensure that the parameter inputs match those set on the FPGA. The details of each parameter input are described as follows.

Mandatory parameters

1) Dir	: t –PC sends data to FPGA r –PC receives data from FPGA
2) FPGAIP	: IP address setting on the FPGA (Default value is 192.168.40.42)
3) FPGAPort	: Port number of the FPGA (Default value in FPGA is 4000)
4) PCPort	: Port number of the PC for sending or receiving data (Default is 61000 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 32 due to UDP40G-IP limitation.

Optional parameters

1) Pattern	: Default value when the 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 the user does not input this parameter is 100. The 100 ms is recommended value for running with UDP40G-IP.

Transmit data mode

The procedure for running the test application in Transmit mode is outlined below.

1) Obtain the parameters from the user and verify their validity.
2) Create a socket and configure the properties of the received buffer.
3) Set the IP address and port number based on the user's parameter, and establish a connection.
4) Populate the Send buffer with data for transmission. During transmission, the application prints the total amount of data sent to the console every second.
    a) If Pattern=1, fill the Send buffer with a 32-bit incremental pattern.
    b) If Pattern=0, the send buffer remains unfilled, and dummy data is used for the test.
5) Once all data has been sent, the application displays the test results, including the total size of transmitted data and performance.

Receive data mode

The procedure for running the test application in Receive mode is outlined below.

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

5 Revision History

Revision	Date	Description
2.00	14-Mar-24	Update register map, matching the updated UDP40G-IP.
1.00	22-Oct-19	Initial release