UDP10G IP reference design manual

Rev1.3   23-May-19

1 Introduction

Comparing to TCP, UDP provides a procedure to send messages with a minimum of protocol mechanism, but the data cannot guarantee to arrive the destination because of no handshaking dialogues. Similar to TCP, UDP provides checksums for data integrity, and port numbers for addressing the different functions at the source and the destination of the datagram.

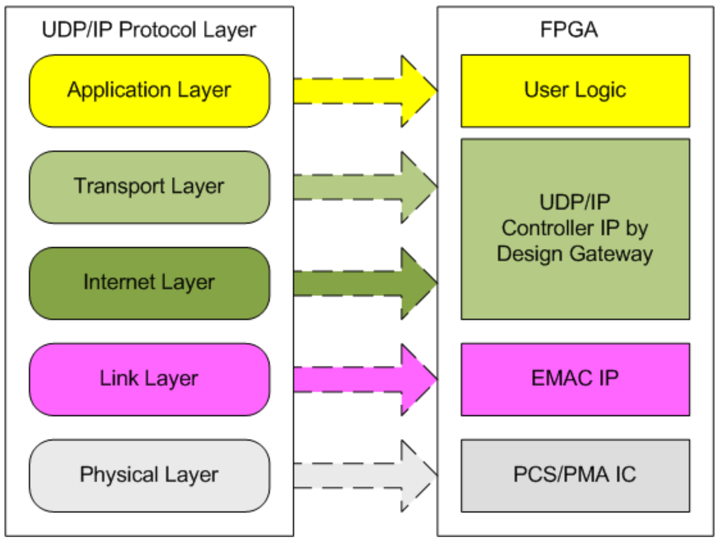


Figure 1-1 UDP/IP Protocol Layer

UDP10G IP implements Transport and Internet layer of UDP/IP Protocol. For transmit side, UDP10G IP prepares UDP data from user logic and add UDP/IP header to generate Ethernet packet format before sending out to EMAC. For receive side, UDP10G IP extracts UDP data from Ethernet packet. UDP/IP header is verified to check packet valid. If packet is valid, UDP data will be extracted and stored in buffer for user logic reading.

The lower layer protocols are implemented by EMAC IP and PCS/PMA IP from Intel FPGA.

This reference design provides evaluation system which includes simple user logic to send and receive data by using UDP10G IP. This system demonstrates on Intel FPGA development board to operate with Test application on PC for transferring high speed data on network. More details are described as follows.

In the demo, CPU firmware is bare-metal. User can input the parameters through NiosII command shell and select transfer directions to start test operation. The test application on PC is “udpdatatest.exe”.

2 Hardware description

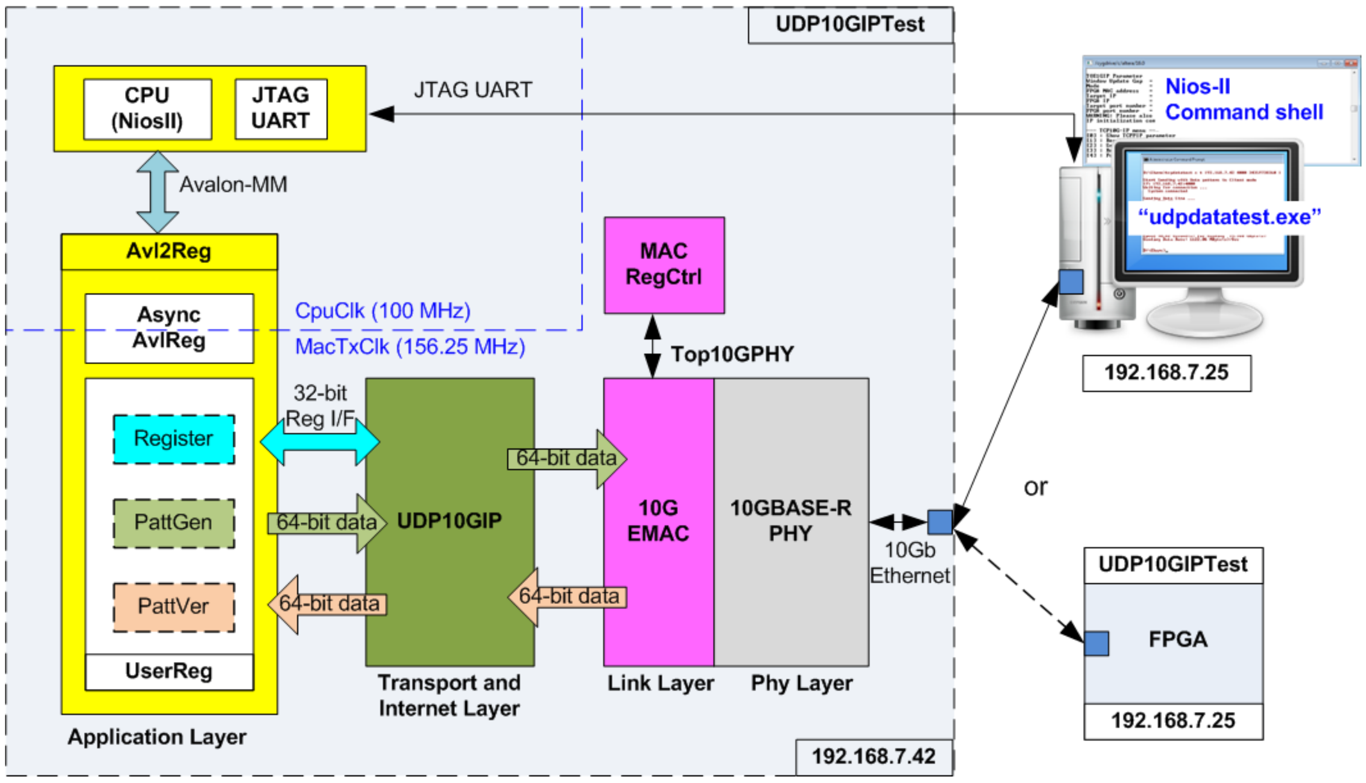


Figure 2-1 Hardware Architecture reference design

In test environment, two devices are used for 10Gb Ethernet transferring. First device is FPGA board and another device is FPGA board or PC. So, two test environments can be setup as shown in Figure 2-1.

When test system uses the PC to transfer data with FPGA, "udpdatatest" must be applied to send and receive data with FPGA. To test half-duplex mode (send or receive data only), one test application is called on the PC. To test full-duplex mode (send and receive data at the same time), two test applications must be called by using different port number and different transfer direction (one is receiving mode and another is sending mode).

In FPGA logic, UDP10G IP connects to 10G Ethernet MAC and 10G Ethernet PHY to complete all UDP/IP layer implementation. User interface of UDP10G IP connects to UserReg within Avl2Reg for both data interface and register interface. Register files of UserReg are split into two regions, i.e. UDP10G IP region and internal test logic region (PattGen and PattVer). Register files of UserReg are controlled by the firmware operating on the CPU (NiosII) through Avalon-MM interface. User can control the operation of UDP10G IP, PattGen, and PattVer by modifying the firmware on the CPU.

Otherwise, UDP10GIPTest includes MACRegCtrl which is designed to setup Ethernet MAC register such as disable and enable flag to send and receive data. 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 Ethernet PHY. So, AsyncAvlReg is designed to support asynchronous transfer between CpuClk and MacTxClk. More details of each module inside the UDP10GIPTest are described as follows.

2.1 10G Ethernet MAC and 10GBASE-R PHY

Link layer and physical layer in the reference design are implemented by IntelFPGA IP core. For Arria10 FPGA, Low latency 10G Ethernet MAC is applied to connect with UDP10G IP and 10GBASE-R PHY. More details are described in following link.
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-10gbps-ethernet-mac.html

10GBASE-R PHY is designed to connect with 10G SFP+ module. Another side is connected to 10G Ethernet MAC through XGMII interface running at 156.25 MHz. More details are described in following link.
https://www.altera.com/products/intellectual-property/ip/interface-protocols/m-alt-10gbase-r-pcs.html

2.2 MACRegCtrl

This module is designed to configure parameter to 10 Gb Ethernet MAC and monitor EMAC status through Avalon MM bus. The logic is simply designed by using state machine. The logic runs only one time after the system powers up for initializing Ethernet MAC.

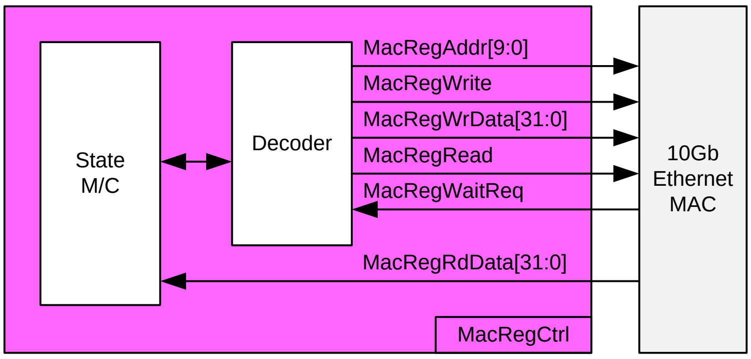


Figure 2-2 MACRegCtrl Block Diagram

The sequence of initialization sequence is below.
1) Disable transmit and receive path of EMAC
2) Wait until transmit and receive path are idle
3) Set receive module to remove CRC and padding
4) Disable pause frame transmission
5) Enable transmit and receive path of EMAC

2.3 UDP10G IP

UDP10G IP implements UDP/IP stack and offload engine. User interface consists of control signals and data signals. Control and status signals are accessed through register interface. Data signals are accessed through FIFO interface. More details are described in datasheet.
http://www.dgway.com/products/IP/UDP10G IP/dg_udp10gip_data_sheet_intel_en.pdf

2.4 CPU and Peripherals

32 bit Avalon-MM is applied to be the bus interface for the CPU accessing the peripherals such as Timer and JTAG UART. To control and monitor the test logic of UDP10G IP, the test logic is connected to CPU as a peripheral on 32 bit Avalon-MM bus. CPU assigns the different base address and the address range for each peripheral.

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 Avalon-MM bus standard for writing and reading the register. Avl2Reg module is designed to connect the CPU system as shown in Figure 2-3.

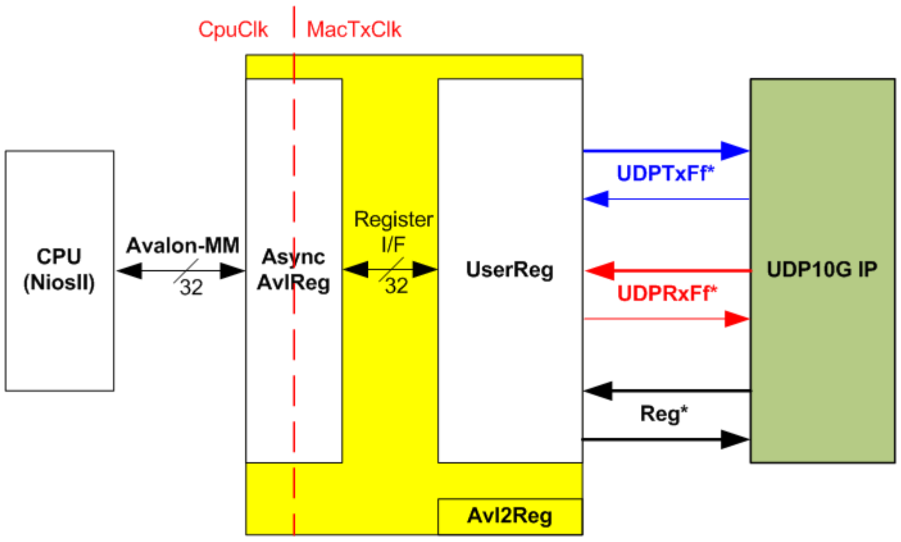


Figure 2-3 Avl2Reg block diagram

Avl2Reg consists of AsyncAvlReg and UserReg. AsyncAvlReg is designed to convert the Avalon-MM signals to be the simple register interface which has 32 bit data bus size (same as Avalon-MM data bus size). Otherwise, AsyncAvlReg 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. Also, both data interface and control interface of UDP10G IP are connected to UserReg. More details of AsyncAvlReg and UserReg are described as follows.

2.4.1 AsyncAvlReg

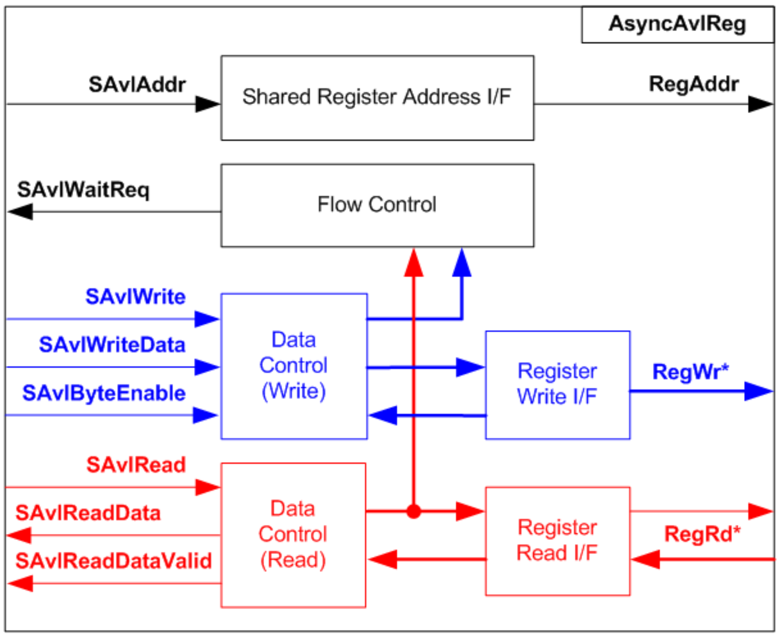


Figure 2-4 AsyncAvlReg Interface

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

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

The simple register interface is designed to be compatible to general RAM interface for write transaction. The read transaction of the register interface is little modified from RAM interface by adding RdReq signal. The address of register interface is shared for write and read transaction. So, user cannot write and read the register at the same time. The timing diagram of the register interface is shown in Figure 2-5

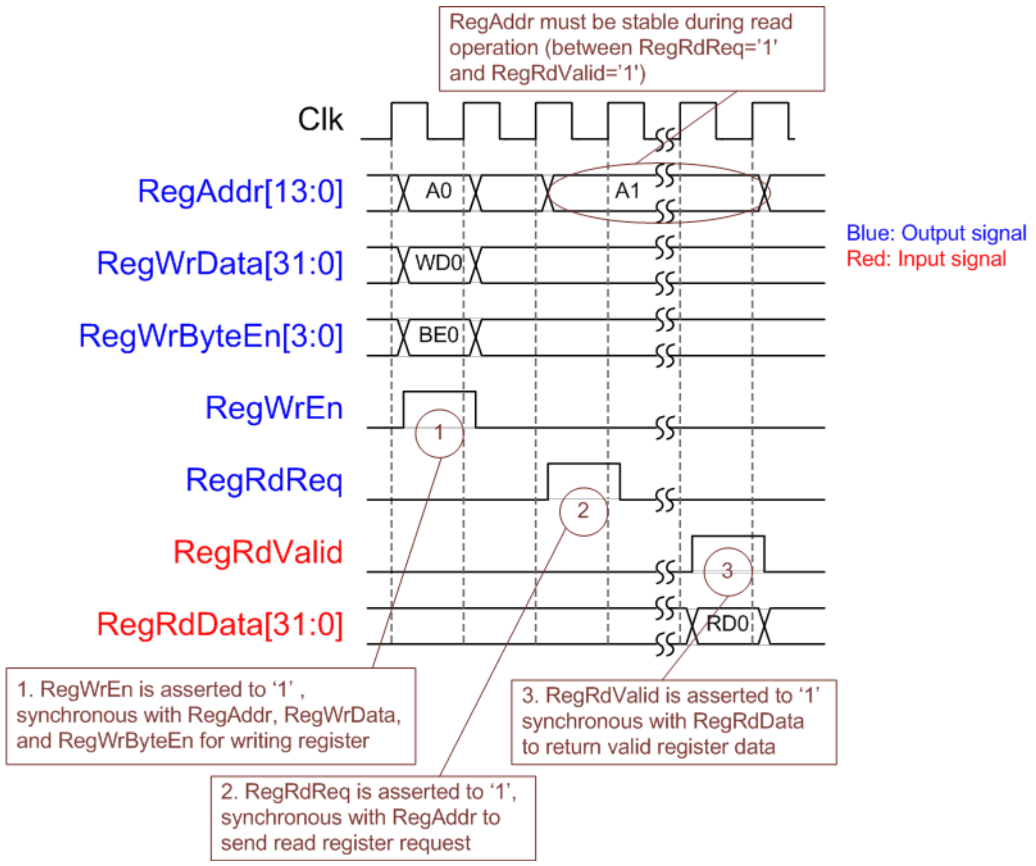

RegAddr must be stable during read operation (between RegRdReq='1' and RegRdValid='1')
Clk
RegAddr[13:0]
A0
A1
RegWrData[31:0]
WD0
RegWrByteEn[3:0]
BE0
RegWrEn
RegRdReq
RegRdValid
RegRdData[31:0]
RD0
Blue: Output signal
Red: Input signal
1
2
3
1. RegWrEn is asserted to '1' , synchronous with RegAddr, RegWrData, and RegWrByteEn for writing register
2. RegRdReq is asserted to '1', synchronous with RegAddr to send read register request
3. RegRdValid is asserted to '1' synchronous with RegRdData to return valid register data

Figure 2-5 Register interface timing diagram

1) To write register, the timing diagram is same as general 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 bit to be the byte data valid, i.e. bit[0] for RegWrData[7:0], bit[1] for RegWrData[15:8], and so on.
2) To read register, AsyncAvlReg asserts RegRdReq to '1' with the valid value of RegAddr. 32 bit data must be returned after receiving the read request. The slave must monitor RegRdReq signal to start the read transaction.
3) The read data is returned on RegRdData bus by the slave with asserting RegRdValid to '1'. After that, AsyncAvlReg forwards the read value to SAvlRead interface.

2.4.2 UserReg

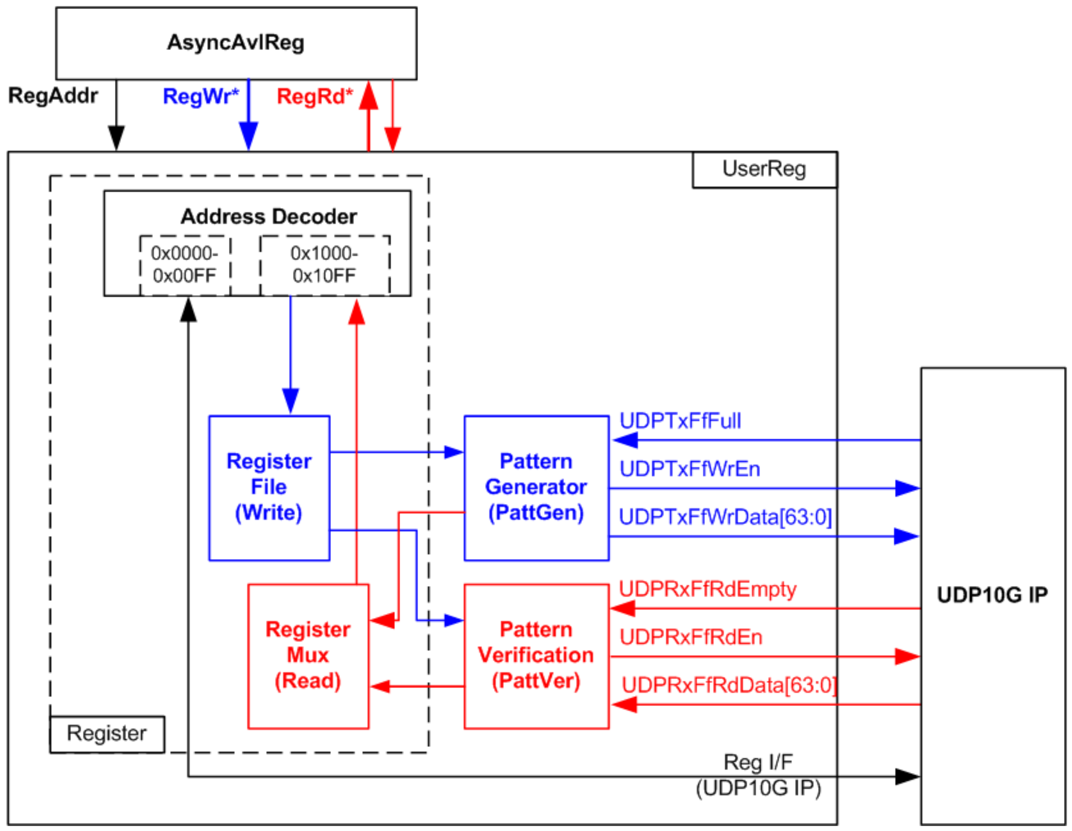


Figure 2-6 UserReg block diagram

The logic inside UserReg could be split into three parts, i.e. Register, Pattern generator (PattGen), and Pattern verification (PattVer). Register block decodes the address requested from AsyncAvlReg 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 set the parameters in the hardware, the CPU must use 32 bit pointer to force 32 bit valid value of the write data.

To read register, one multiplexer is designed to select the read data within each address area. The lower bit of RegAddr is applied in the Register Mux. 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 RegRdValid with asserting one D Flip-flop. More details of the address mapping within UserReg module is 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	UDP10_RST_REG	Mapped to RST register within UDP10G IP
BA+0x0004	UDP10_CMD_REG	Mapped to CMD register within UDP10G IP
BA+0x0008	UDP10_SML_REG	Mapped to SML register within UDP10G IP
BA+0x000C	UDP10_SMH_REG	Mapped to SMH register within UDP10G IP
BA+0x0010	UDP10_DIP_REG	Mapped to DIP register within UDP10G IP
BA+0x0014	UDP10_SIP_REG	Mapped to SIP register within UDP10G IP
BA+0x0018	UDP10_DPN_REG	Mapped to DPN register within UDP10G IP
BA+0x001C	UDP10_SPN_REG	Mapped to SPN register within UDP10G IP
BA+0x0020	UDP10_TDL_REG	Mapped to TDL register within UDP10G IP
BA+0x0024	UDP10_TMO_REG	Mapped to TMO register within UDP10G IP
BA+0x0028	UDP10_PKL_REG	Mapped to PKL register within UDP10G IP
BA+0x0038	UDP10_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.

Pattern Generator

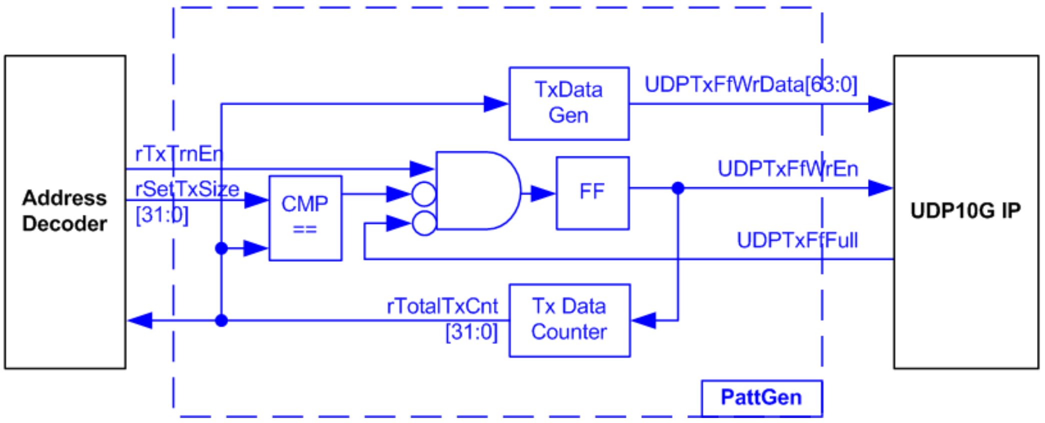


Figure 2-7 PattGen block

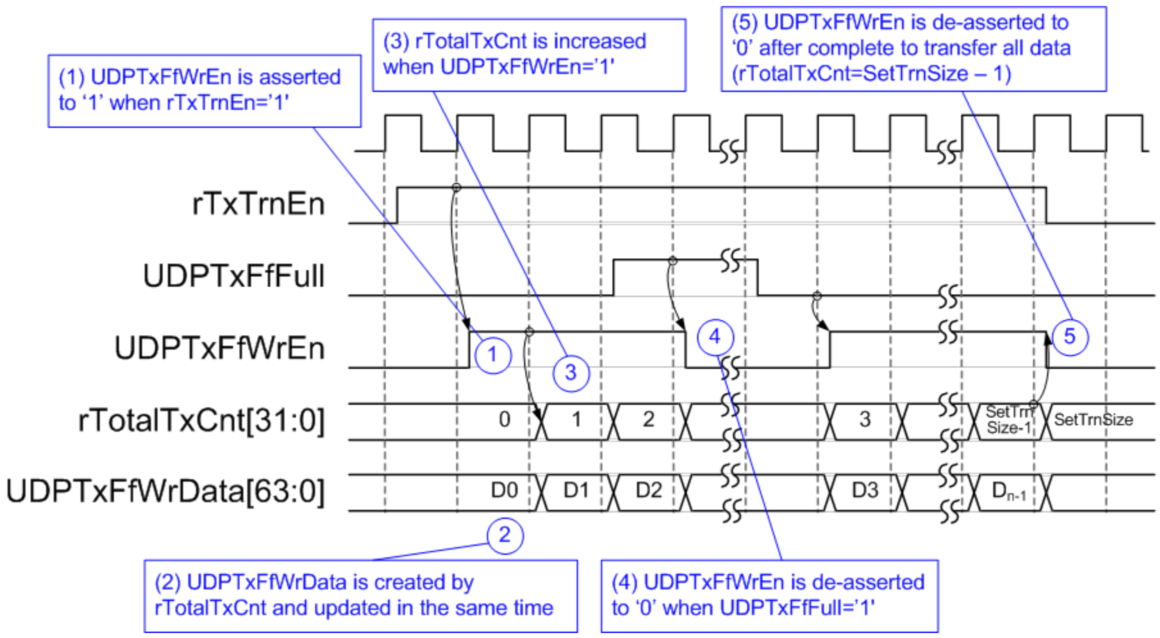

(1) UDPTxFfWrEn is asserted to '1' when rTxTrnEn='1'
(3) rTotalTxCnt is increased when UDPTxFfWrEn='1'
(5) UDPTxFfWrEn is de-asserted to '0' after complete to transfer all data (rTotalTxCnt=SetTrnSize – 1)
rTxTrnEn
UDPTxFfFull
UDPTxFfWrEn
rTotalTxCnt[31:0]
UDPTxFfWrData[63:0]
0
1
2
3
SetTrnSize-1
SetTrnSize
D0
D1
D2
D3
Dn-1
(2) UDPTxFfWrData is created by rTotalTxCnt and updated in the same time
(4) UDPTxFfWrEn is de-asserted to '0' when UDPTxFfFull='1'

Figure 2-8 PattGen Timing diagram

PattGen is designed to generate test data to UDP10G IP. rTxTrnEn is asserted to ‘1’ when USER_CMD_REG[0] is set to ‘1’. 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 sending to UDP10G IP. rTotalTxCnt is also used to generate 32 bit increment data to UDPTxFfWrData signal. rTxTrnEn is de-asserted to ‘0’ when finishing to transfer total data (total data is set by rSetTxSize).

Pattern Verification

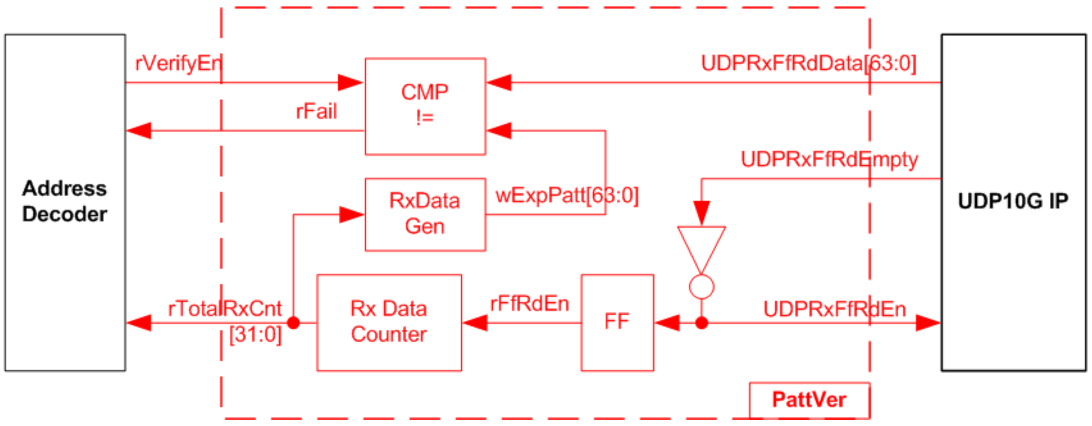


Figure 2-9 PattVer block

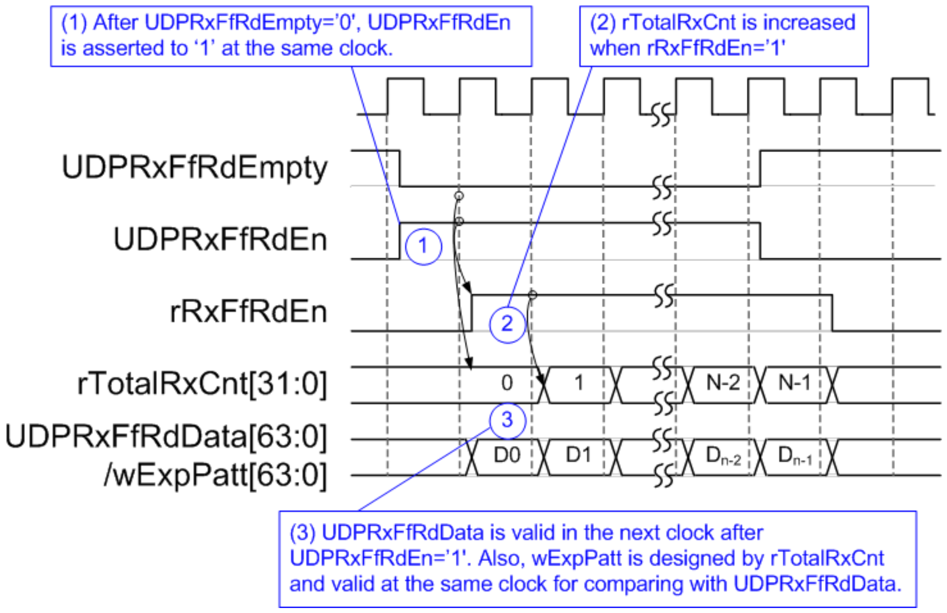

(1) After UDPRxFfRdEmpty='0', UDPRxFfRdEn is asserted to '1' at the same clock.
(2) rTotalRxCnt is increased when rRxFfRdEn='1'
UDPRxFfRdEmpty
UDPRxFfRdEn
rRxFfRdEn
rTotalRxCnt[31:0]
UDPRxFfRdData[63:0] /wExpPatt[63:0]
0
1
N-2
N-1
D0
D1
Dn-2
Dn-1
(3) UDPRxFfRdData is valid in the next clock after UDPRxFfRdEn='1'. Also, wExpPatt is designed by rTotalRxCnt and valid at the same clock for comparing with UDPRxFfRdData.

Figure 2-10 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) to the expected pattern (wExpPatt). If data verification is failed, rFail will be 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 to count total transfer size. rTotalRxCnt is used to generate wExpPatt.

3 CPU Firmware Sequence

After FPGA boot-up, user must select the operation mode on FPGA. Two modes could be selected, i.e. client or server. The operation mode is the set value for UDP10_SRV_REG register.

(a) In client mode, FPGA sends ARP request to get the MAC address from the target during initialization sequence.

(b) In server mode, FPGA waits ARP request from the target and returns ARP reply during initialization sequence.

When test environment uses two FPGAs, the 1st FPGA must run as server mode and another FPGA must run as client mode. User needs to start server FPGA initialization before client FPGA initialization.

When test environment uses FPGA and Test PC, FPGA could be set to initialize as client mode. Server mode is not recommended because it is not easy to force PC to generate ARP request to FPGA in initialization phase.

In the firmware, there are two sets of default parameters following the mode, i.e. client parameters and server parameters. The initialization sequence after system boot-up is as follows.

1) CPU receives the operation mode from user and displays default parameters on the console.
2) User inputs 'x' to complete initialization sequence by using default parameters or inputs other keys to change some parameters. In case of changing parameters, the operation sequence is same as Reset IP which is described in topic 3.2.
3) CPU waits until UDP10G IP completes initialization sequence (UDP10_CMD_REG[0]='0').
4) Main menu is displayed with five operations. More details of each operation are described as follows.

3.1 Show parameters

This menu is used to show current parameters of UDP10G IP such as operation mode, FPGA MAC address, FPGA IP address, and FPGA port number. The sequence to display parameters is as follows.

1) Read 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, source port number. After setting UDP10G IP register, CPU resets the IP to re-initialize by using new parameters. CPU monitors busy flag to wait until the initialization is completed. The sequence of reset sequence is shown as follows.

1) Display current parameter value to 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 UDP10_RST_REG[0]='1'.
5) Set all parameters to UDP10G IP register such as UDP10_SML_REG, UDP10_DIP_REG.
6) De-assert UDP10G IP reset by setting UDP10_RST_REG[0]='0'.
7) Reset UserReg by setting USER_RST_REG[0]='1'.
8) Monitor UDP10G IP busy flag (UDP10_CMD_REG[0]). Wait until busy flag is de-asserted to '0' to confirm that the initialization sequence is completed.

3.3 Send data test

User needs to input two parameters, i.e. total transmit length and packet size. The operation will be cancelled if the input is invalid. 32 bit increment 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), 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 parameter of test application running 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 UDP10_PKL_REG and total size is set to UDP10_TDL_REG. Finally, UDP10_CMD_REG is set to 1 to start IP sending data.
5) Wait until UDP10G IP completes operation by monitoring IP busy flag (UDP10_CMD_REG[0]='0'). During waiting, 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 will be cancelled if the input is invalid. During the test, 32 bit increment 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 initial value of test pattern (USER_RST_REG) 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 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 waiting, CPU reads current received size from user logic (USER_RXLEN_REG) and displays 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, error message will be displayed.
7) Calculate performance and show test result on the console.

3.5 Full duplex test

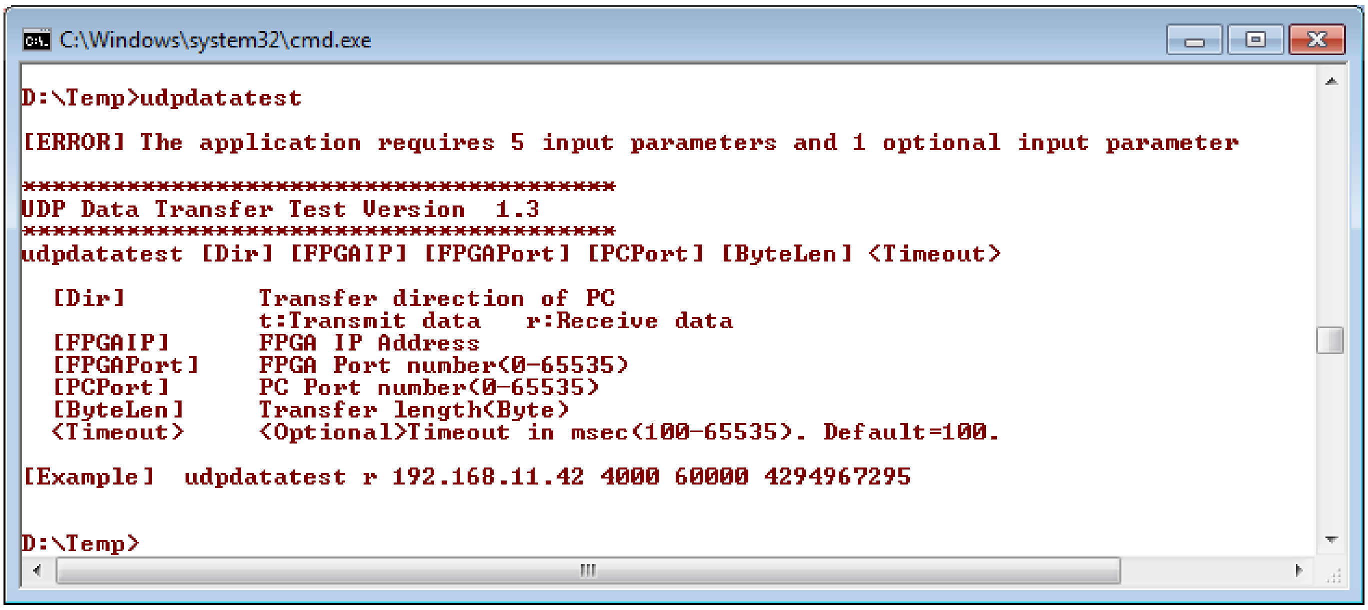
This menu is designed to run full duplex test by transferring data between FPGA and PC/FPGA in both directions at the same time. Three inputs are received from user, i.e. total size for both directions, packet size for FPGA sending logic, and data verification mode for FPGA receiving logic.

To run full duplex test by using PC, user opens “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 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), 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 (UDP10_PKL_REG), total transfer size (UDP10_TDL_REG), and write command (UDP10_CMD_REG=1). IP starts sending data operation after UDP10_CMD_REG is set to 1. For receiving data, IP is always ready to receive data without additional setting.
5) Wait until the operation is completed for both sending and receiving direction.
   a. For sending direction, wait until busy flag of UDP10G IP (UDP10_CMD_REG[0])=’0’.
   b. For receiving direction, wait until total received size is equal to set value or total received size does not change for 100 msec (timeout condition)

   During waiting, 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.

4 Test Software description


C:\Windows\system32\cmd.exe

```
D:\Temp>udpdatatest

[ERROR] The application requires 5 input parameters and 1 optional input parameter

****************************************
UDP Data Transfer Test Version  1.3
****************************************
udpdatatest [Dir] [FPGAIP] [FPGAPort] [PCPort] [ByteLen] <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)
    <Timeout>      <Optional>Timeout in msec(100-65535). Default=100.

[Example]  udpdatatest r 192.168.11.42 4000 60000 4294967295


D:\Temp>
```

Figure 4-1 udpdatatest application parameter

"udpdatatest" is an application on PC for sending or receiving UDP data. There are five parameters and one optional parameter. The parameter input should be matched to parameter setting 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) Timeout : Timeout for receiving data in msec unit. It is recommended to use 100 for running with UDP10G IP. Default value in software when user does not input this parameter is 100.

Receive data mode

Following is the sequence when test application runs in receive mode.

(1) Get parameters from user.
(2) Create socket and then set properties of receive buffer.
(3) Set IP address and port number from user parameter and then connect.
(4) Loop to verify data until total data is equal to set value or no more data is received until timeout. Verification pattern is 32 bit increment starting from 0, so test pattern is increased after receiving 4 bytes. During running, application prints total received size on the console every second.
(5) In case of timeout condition, “Timeout” message is displayed with total lost size and total receive size when end of operation.

Transmit data mode

Following is the sequence when test application runs in transmit mode.

(1) – (3) are same as Receive data mode
(4) Generate 32 bit increment pattern to buffer and then send data out. The packet size is fixed to 1472 byte.
(5) After completing to send all data, the application displays performance with total data size as 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	22-Mar-18	Add Avl2Reg details
1.3	23-May-19	Support FPGA <-> FPGA connection