UDP-IP reference design manual

Rev1.0 6-Jan-16

1. Introduction

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

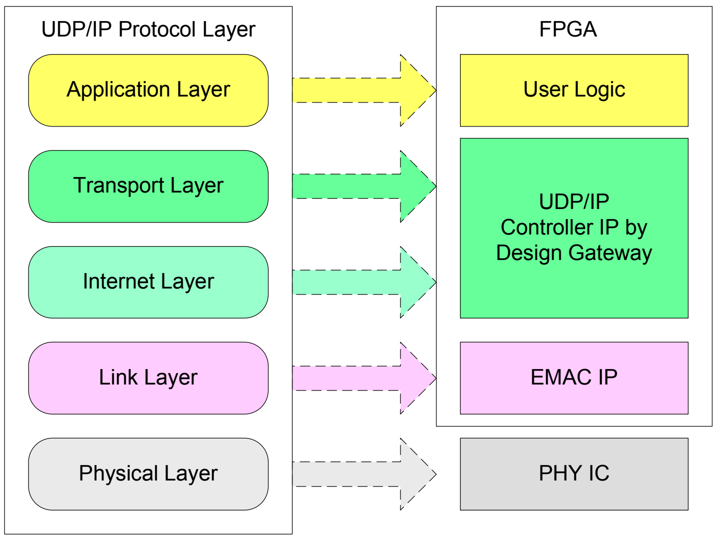

Figure 1 UDP/IP Protocol Layer

UDP-IP implements Transport and Internet layer of UDP/IP Protocol. For transmit side, UDP-IP will prepare UDP data from user logic and add UDP/IP header to generate Ethernet packet format before sending out to EMAC. For received side, UDP-IP will extract UDP data from Ethernet packet. UDP/IP header will be 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 from Xilinx and external PHY chip.

This reference design provides evaluation system which includes simple user logic to send and receive data with UDP-IP. This system demonstrates on AC701 Development board to operate with Test application on PC for transferring high speed data on network. More details are described as follows.

2. Environment

This reference design is based on the following environment as shown in Figure 2.

- AC701 Platform
- iMPACT 14.4 or later
- Ethernet cable (Cat5e or Cat6)
- PC with Gigabit Ethernet
- USB Micro-B cable for FPGA configuration
- Test Application, i.e. “send_udp_client” and “recv_udp_client”, provided by Design Gateway

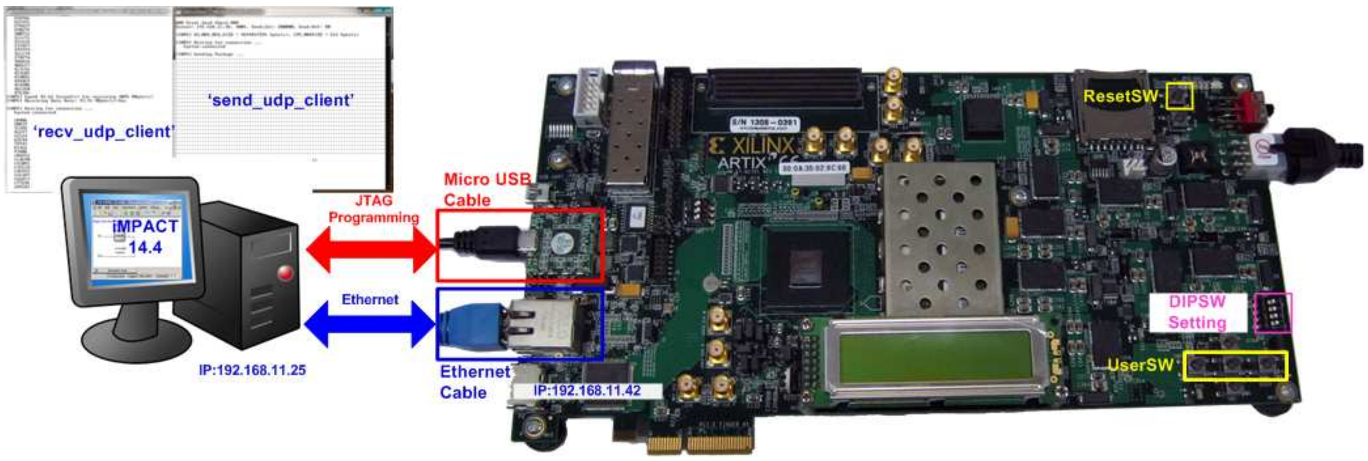

Figure 2 UDPIP Demo on Development board

3. Hardware description

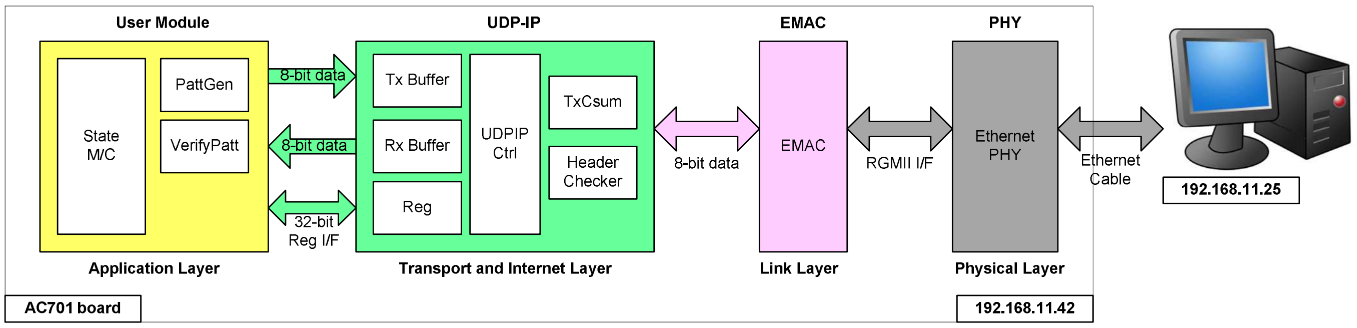

Figure 3 Hardware Architecture in AC701 reference design

As shown in Figure 3, hardware architecture can be divided into 4 modules to support each UDP/IP layer protocol. UDP-IP operates with EMAC and external PHY to implement all four lower layers of UDP/IP Protocol. The reference design can transfer data in both directions by running with test application on PC. For FPGA to PC direction, UDP data in reference design is generated by test pattern generator inside User module. Test data will be verified by test application on PC (recv_udp_client.exe). For PC to FPGA direction, data generated by test application on PC (send_udp_client.exe) will be verified by User module. State machine is designed to set and monitor UDP-IP command and status.

- External PHY

Physical layer is implemented by external PHY chip, interfaced by RGMII on AC701.

- EMAC

Link layer is implemented by EMAC-IP (Tri Mode Ethernet MAC with 1000 Mbps speed), provided by Xilinx. The setting of EMAC-IP is followed.

  - PHY Interface is RGMII
  - MAC speed is fixed to 1000 Mbps
  - Configuration is Ver
  - MDIO I/F is disable

Data interface of EMAC can connect with UDP-IP directly.

- UDP-IP

More details are described in “dg_udpip_data_sheet_xilinx_en.pdf” document.

- User Module

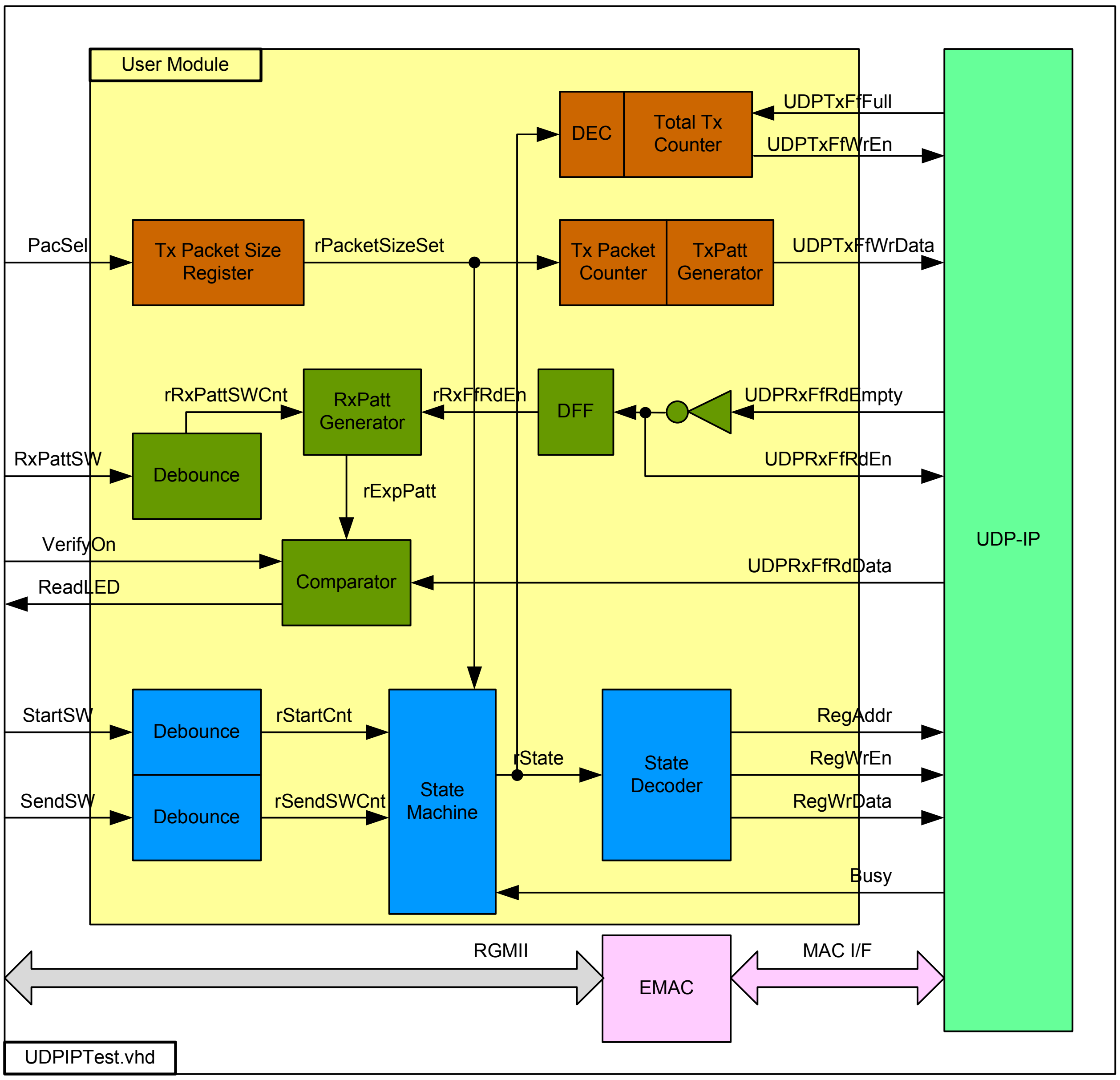

Figure 4 User Module block diagram

User Module can split into three parts, i.e. TxFIFO interface, RxFIFO interface, and Control interface. For transmit operation, TxPatt Generator generates 32-bit increment test data to be output for TxFIFO. Pattern is increased every end of each Tx packet, so Tx Packet Counter is designed to count the number of data in each Tx packet. Two packet sizes can be selected from PacSel DIPSW, i.e. 1472 for non-jumbo frame size, or 8972 for jumbo frame size. Test pattern will be generated during transmit operation until equal to total transfer size. State machine is decoded to confirm that user requests to run new transmit test. Total Tx Counter is used to count total numbers of test data, which is fixed to 0xFFFF_FFFF (4GB) in the reference design.

32-bit increment data is also generated by RxPatt Generator to verify data output from RxFIFO interface of UDP-IP. The logic to read out data from FIFO is simply designed by monitoring Empty flag. ReadLED is blink when read data is not equal to expected data from RxPatt Generator and data verification is enabled.

Control interface is designed by using State Machine. Register address and write value signals are decoded from State Machine to program parameter for initialization process and set packet size/transfer size/command register for transmit operation. Transmit operation will start after SendSW is pressed by user. State Machine diagram is shown in Figure 5. Busy flag output from IP is used to monitor status that IP operation has been completed.

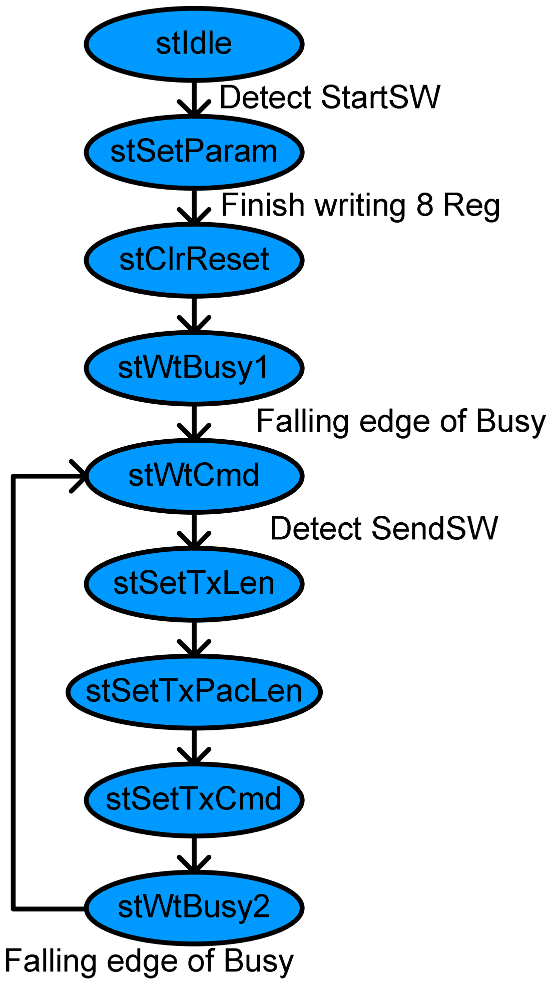

Figure 5 State Machine Diagram within User Module

State machine changes to stSetParam state after user press StartSW button. In stSetParam state, following parameters will be set to UDP-IP register.

- Source MAC address (SML/SMH Reg)	= 00:01:02:03:04:05
- Source IP address (SIP Reg)	= 192.168.11.42
- Source Port number (SPN Reg)	= 4000
- Destination IP address (DIP Reg)	= 192.168.11.25
- Destination Port number for Tx(DPN Reg[15:0])	= 60000
- Destination Port number for Rx(DPN Reg[31:16])	= 60001

Next State, stClrReset, is applied to release Reset signal (RST Reg=0) within UDP-IP and then IP starts parameter initialization. State machine waits initialization complete by monitoring falling edge of Busy signal. Then, state will stay in stWtCmd which is Idle state to wait command from user.

For data transmit operation, state machine is designed to control to send 4GB test data from FPGA to PC. After user press SendSW button, state machine will go to stSetTxLen for setting total transfer size (TDL Reg), stSetTxPacLen for setting packet size (PKL Reg), and stSetTxCmd for setting command register (CMD Reg). Then, Busy signal is monitored to wait transfer complete in stWtBusy2 state. After all data are transferred completely, it will go back to stWtCmd for waiting next command.

For received mode, IP can transfer UDP data from PC to user logic without setting any register by State machine. So, user can test data transfer in both directions at the same time.

4. Test Software description

Two test applications are applied within this demo, i.e. “recv_udp_client” and “send_udp_client”. To run both applications at the same time, port number at PC side will be set by different value. 60000 is used for FPGA to PC direction, while 60001 is used for PC to FPGA direction.

- recv_udp_client

This test application runs to test sending operation of UDP-IP, so data sending to PC will be verified by test application. This application requires five input parameters from user, which is fixed value from UserModule HDL code. Since parameter input of test application must be matched with the value inside hardware, user needs to modify HDL code to change any input parameter value.

- Dst_Addr: IP address of FPGA. Set to “192.168.11.42” for this demo.
- Dst_Port: FPGA Port number. Set to “4000” for this demo.
- Src_Port: PC Port number. Set to “60000” for this demo.
- Recv_Len: Packet size in byte unit. Set to “1472” for non-Jumbo frame mode, or “8972” for Jumbo frame mode. If setting with wrong value, verified error message will be displayed on Test application and operation will be stopped.
- Total_Len: Total transfer size in byte unit. Set to “4294967295” for this demo.

The operation sequence of the application is follows.

(1) Get parameters from user.

(2) Create socket and then set properties of received buffer.

(3) Set IP address and Port number from user parameter and then connect.

(4) Loop to verify data until total received data is equal to set value or no more received data within 0.5 sec. Verification pattern is 32-bit increment starting from 0. Pattern is increased after end of each packet size (1472 or 8972 byte). So, all data in one packet will be similar. Two messages can be printed out from verification process, i.e.
   - “Drop Expect” when 1st data of packet is not equal to expect value. This is warning message and application still continue to verify data.
   - “Error Expect” when data within the packet is not equal to 1st data. This case is error condition, so application will stop operation.

   During running, application will print total received size on the console every second.

(5) Socket is closed by PC. Application will show performance with total number of received data and total drop packet to be test result. Also, “Timeout” message will be displayed if the loop is exit from 0.5 sec timeout, not total data are received.

- send_udp_client

This test application sends data out to UDP-IP to test receiving operation. Similar to recv_udp_client, this application requires five input parameters from user, which is fixed value from UserModule HDL code.

- Dst_Addr: IP address of FPGA. Set to “192.168.11.42” for this demo.
- Dst_Port: FPGA Port number. Set to “4000” for this demo.
- Src_Port: PC Port number. Set to “60001” for this demo. Different port number from recv_udp_client is used to support running two test applications at the same time.
- Packet Count: Total number of 8kByte packet to send out to FPGA. Total byte size is equal to this value x 8096 byte. Valid range is 1-524287.
- Verification On/Off: Set ‘0’ to transfer dummy data, or ‘1’ to transfer 32-bit increment data out. This setting value is effect to output performance from PC. In some PC, performance in dummy data mode is better than increment data mode.

The operation sequence of the application is follows.

(1) Get parameters from user.
(2) Create socket and then set properties of transmit buffer.
(3) Set IP address and Port number from user parameter and then connect.
(4) Fill test pattern with dummy (all ‘0’) or increment pattern to buffer and then send data out. Each packet size is fixed to 8096 byte unit. By using this packet size, PC which can support jumbo-frame will generate UDP packet without fragmentation.
Note: *IP does not support to receive IP packet which has fragment.*
(5) Close socket and print out performance with total number of transferred data as test result.

5. Revision History

Revision	Date	Description
1.0	6-Jan-16	Initial Release