

TOE10G-IP reference design manual

Rev1.0 18-Sep-14

1. Introduction

TCP/IP is the core protocols of the Internet Protocol Suite for networking application. TCP/IP model has four layers, i.e. Application Layer, Transport Layer, Internet Layer, and Network Access Layer. In Figure 1, five layers are displayed for simple matching with hardware implementation by FPGA. Network Access Layer is split into Link and Physical Layer.

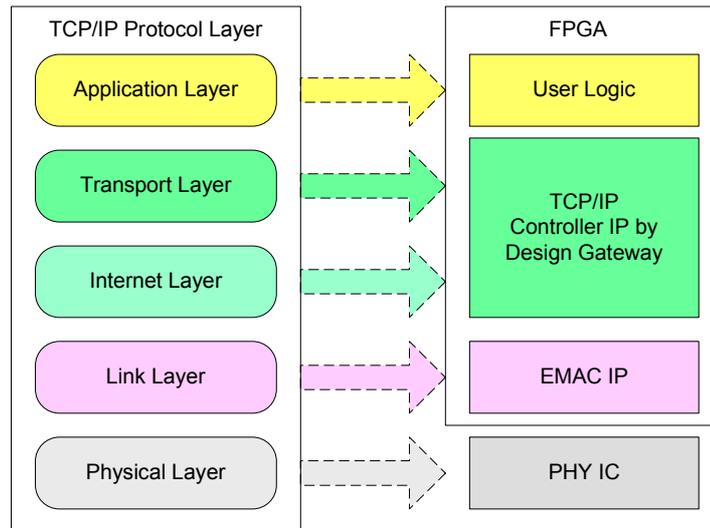


Figure 1 TCP/IP Protocol Layer

TOE10G-IP implements Transport and Internet layer of TCP/IP Protocol. For transmit side, TOE10G-IP will split TCP data from user logic into packet format with TCP and IP header generation and then send out to EMAC. For received side, TOE10G-IP will extract TCP data and header from IP packet and then store only TCP data in buffer for user logic reading.

By using TCP, data reliability is guaranteed by monitoring acknowledge packet. For transmit side, TOE10G-IP will retransmit data if acknowledge packet is lost. For received side, if TOE10G-IP detects that some received data is lost, it will send duplicate acknowledge packet to request the lost packet.

The lower layer protocols are implemented by EMAC-IP and PHY-IP from Xilinx.

This reference design provides evaluation system which includes simple user logic to send and receive data with TOE10G-IP. This system demonstrates on KC705/VC707 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.

- KC705/VC707 Platform
- iMPACT 14.4 or later
- 10-Gigabit SFP+ DAC cable or 2x10-Gigabit SFP+ Transceiver with optical cable
- PC with 10Gigabit Ethernet support or 10Gigabit Ethernet card
- USB Micro-B cable for FPGA configuration
- Test Application, i.e. “send_tcp_client_10G” and “rcv_tcp_client_10G”, provided by Design Gateway

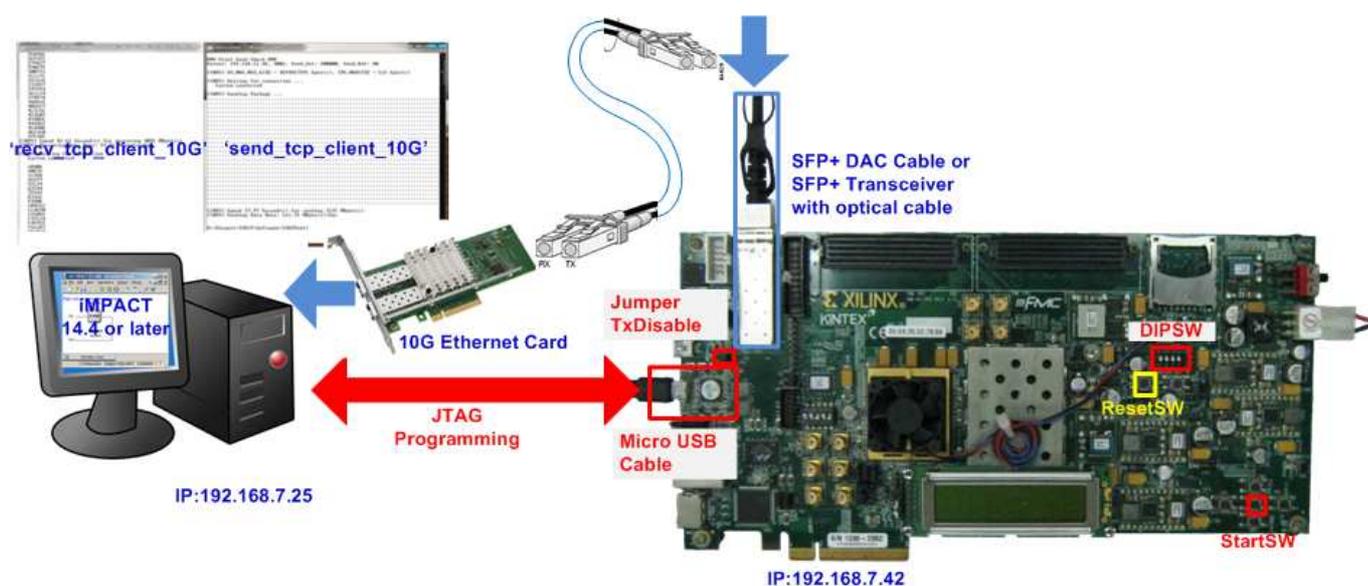


Figure 2 TOE10GIP Demo on Development board

3. Hardware description

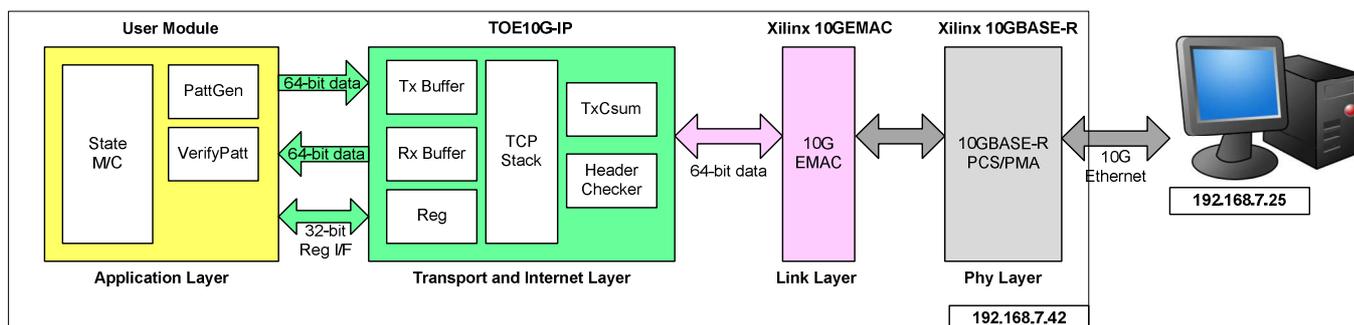


Figure 3 Hardware Architecture in reference design

As shown in Figure 3, hardware architecture can be divided into 4 modules to support each TCP/IP layer protocol. TOE10G-IP operates with 10GEMAC and PCS/PMA to implement all four lower layers of TCP/IP Protocol. User prepares modules to transfer TCP data and write/read control signal with TOE10G-IP for data transferring with test application on PC. This reference design includes the example of User Module to generate Test pattern for transmit and verify Test pattern for received side based on KC705/VC707 development board.

- 10GEMAC and PHY

Both Link layer and physical layer of 10G Ethernet are implemented by using Xilinx IP core. 10GEMAC-IP is implemented for Link layer while 10GBASE-R PCS/PMA is implemented for Physical layer. The connection of 10GEMAC can connect to TOE10G-IP directly.

- TOE10G-IP

More details about both modules are described in “dg_toe10gip_data_sheet_xilinx_en.pdf” document.

● User Module

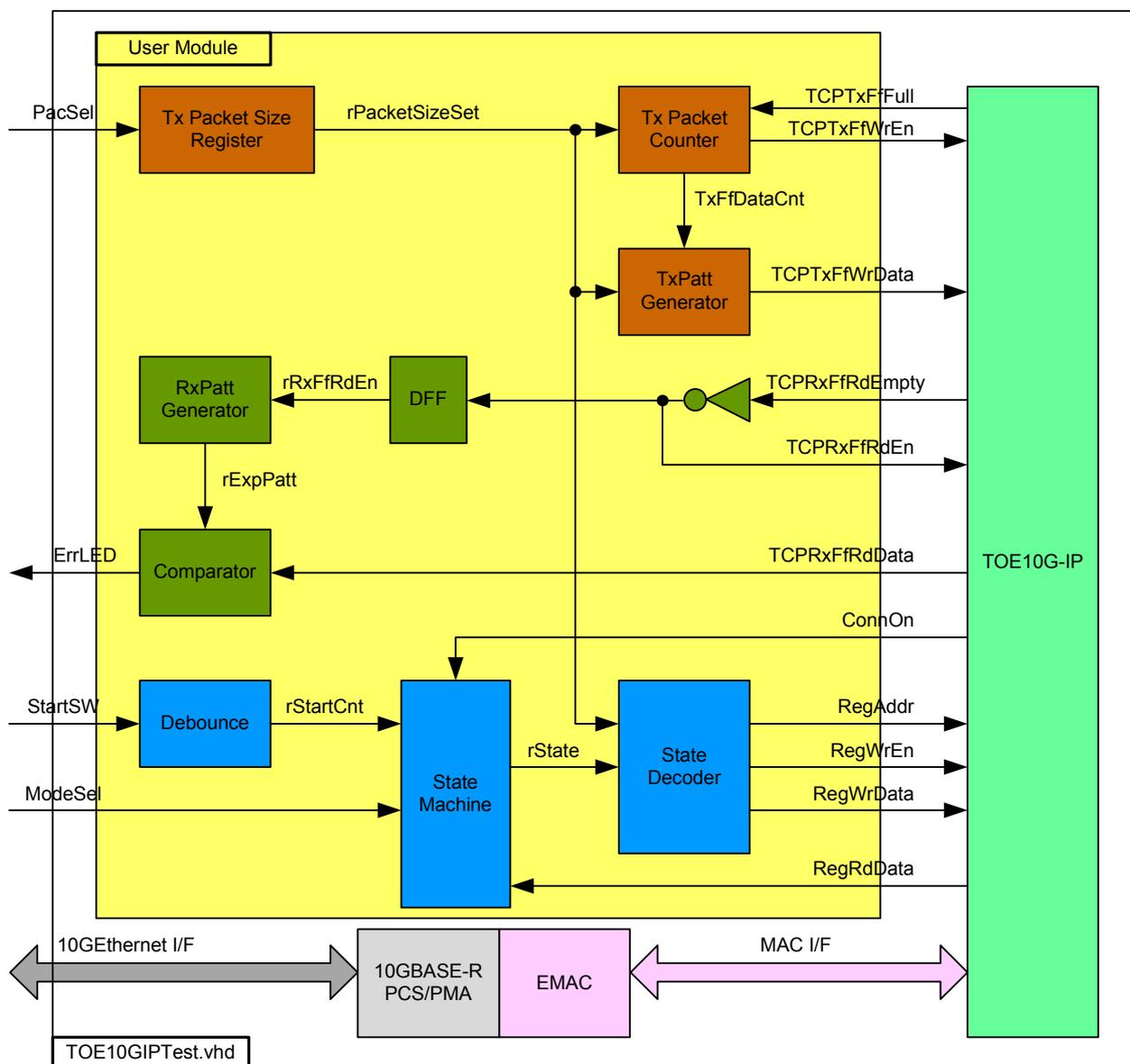


Figure 4 User Module and Test System block diagram

User Module can split into three parts, i.e. Tx FIFO interface, Rx FIFO interface, and Register Control interface. 32-bit increment test data is increased by one every end of each Tx packet for Transmit side. Tx Packet counter is designed to count the number of data in each Tx packet which can be set by external PacSel DIPSW. There are two supported sizes in this demo, i.e. 1456 bytes for non-Jumbo frame, and 8960 bytes for Jumbo frame.

32-bit increment data is also generated by RxPatt Generator to verify data output from Rx FIFO interface of TOE10G-IP. Simple logic to read out data from FIFO by monitoring Empty flag is designed. ErrLED is Blink when data verification is error.

Register Control interface is designed by using State Machine. Register address and write value signals are decoded from State Machine. Transfer data direction is selected by ModeSel DIPSW, and data starts transferring when StartSW is pressed by user. State Machine diagram is displayed as shown in Figure 5.

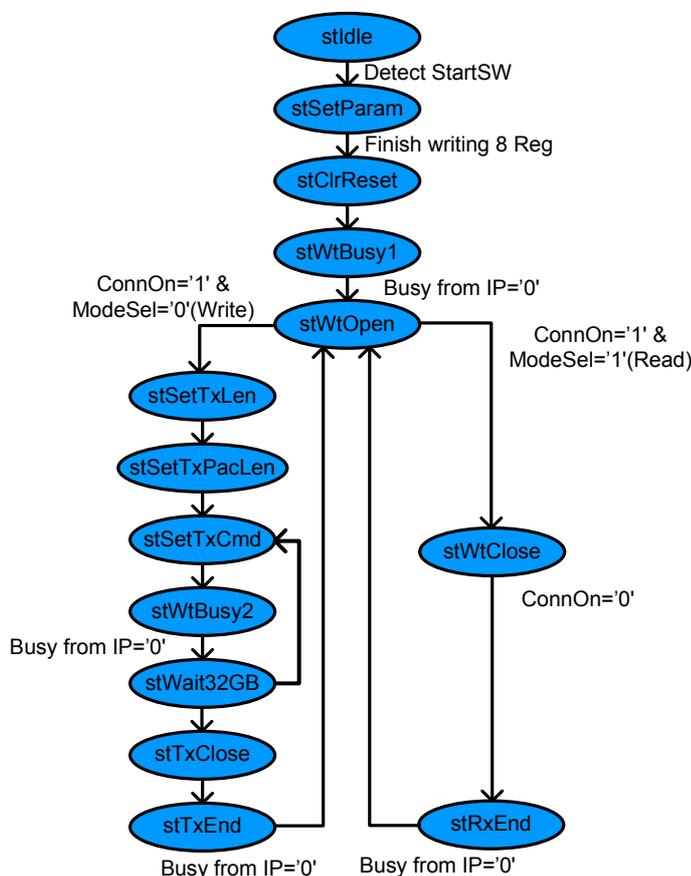


Figure 5 State Machine Diagram within User Module

State machine changes to stSetParam state after user press StartSW button. In stSetParam state, it will set parameters to register within TOE10G-IP, i.e.

- Source MAC address = 00:01:02:03:04:05
- Source IP address = 192.168.7.42
- Source Port number = 4000
- Destination IP address = 192.168.7.25

Next State, stClrReset, is applied to release Reset signal (RST Reg=0) within TOE10G-IP and then starting parameter initialization. State machine waits initialization complete by monitoring Busy flag (bit0 of CMD Reg) through Register interface. Then, state will pause in stWtOpen until test application on PC starts running.

On this demo, FPGA runs as Server mode and Test application on PC runs as Client mode, so the connection is opened by Test application on PC. ConnOn output from TOE10G-IP will change to '1' and then state machine will change to stSetTxLen for transmit mode or to stWtClose for received mode.

On transmit mode, more three states, i.e. stSetTxLen, stSetTxPacLen, and stSetTxCmd are designed for setting total transfer size (TDL Reg), packet size (PKL Reg), and data sending command to register (CMD Reg=0) within TOE10G-IP. Then, Busy signal is monitored to wait transfer 4GB complete in stWtBusy2 state. If data is not equal to 32 GB, State will change to send command again. After all data are transferred completely, State sends command to TOE10G-IP for sending packet to close connection (CMD Reg=0x3) in stTxClose state. After connection is closed and Busy = '0', state will run in stWtOpen for waiting next transfer.

On received mode, state will stay in stWtClose to wait data transfer from PC complete and follow with connection closed command from PC. So, ConnOn value from TOE10G-IP will change from '1' to '0' after connection has already closed. Similar to transmit mode, state will go back to stWtOpen for waiting next transfer.

In conclusion, the demo uses passive open mode for both transmit and received transfer, but uses both modes to close connection. Active close connection is operated for data transmit mode while passive close connection is operated for data received mode.

4. Test Software description

Two test applications are applied within this demo, i.e. “recv_tcp_client_10G” and “send_tcp_client_10G”. Both application runs as client mode.

● recv_tcp_client_10G

This test application runs to test sending operation of TOE10G-IP, so data sending to PC will be verified by test application. This application requires three input parameters from user, i.e.

- FPGA IP address: This demo sets IP address to “192.168.7.42”. User can modify HDL code of User module to change this value.
- FPGA Port number: This demo sets Port number to “4000”. User can modify HDL code of User module to change this value.
- Packet size: This demo can set as two values, i.e. 1456 for non-Jumbo frame mode, and 8960 for Jumbo frame mode. If setting with wrong value, verified error message will be displayed on Test application and operation will be stopped.

The operation sequence of the application is follows.

- (1) Get three 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 run for receiving data and verify data. Data format will be 32-bit increment value which will increase every end of packet. Thus, all data in same packet will be similar. Two errors can be printed out from verification process, i.e. “Drop Expect” printed out when 1st data of packet is not expect value, and “Error Expect” printed out when data within each packet is not expect value. Total number of packet is printed out on console every second.
- (5) Socket is closed by FPGA side and then Performance with total number of transferred data will be printed out as test result.
- (6) Go back to step (3) in loop to continue test operation until operation cancel.

● send_tcp_client_10G

This test application sends data out to TOE10G-IP to test received operation. This application requires four input parameters from user, i.e.

- FPGA IP address: This demo sets IP address to “192.168.7.42”. User can modify HDL code of User module to change this value.
- FPGA Port number: This demo sets Port number to “4000”. User can modify HDL code of User module to change this value.
- Packet Count: This value is set transfer size in 60kByte unit. Total transfer size is equal to this value x 60kByte. Valid range is 1-559241.
- Verification On/Off: Select ‘0’ to transfer dummy data and ‘1’ to transfer 32-bit increment data out. This setting value is effect to output performance from PC. Typically, performance in dummy data mode is better than increment data mode.

The operation sequence of the application is follows.

- (1) Get four 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. Transfer size is set from user.
- (5) Close socket and print out performance with total number of transferred data as test result.



5. Revision History

Revision	Date	Description
1.0	18-Aug-14	Initial Release

Copyright: 2014 Design Gateway Co,Ltd.