TOE2-IP reference design on SP605 manual

Rev1.0 11-Dec-12

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.

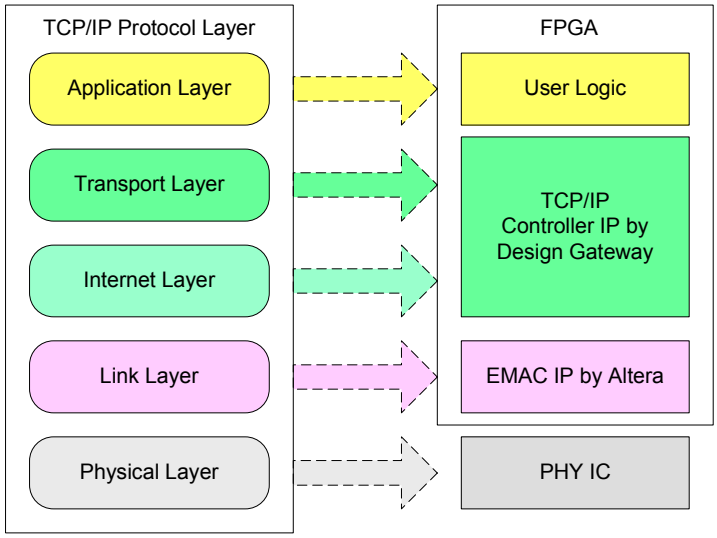

Figure 1 TCP/IP Protocol Layer

TOE2-IP implements Transport and Internet layer of TCP/IP Protocol. For transmit side, TOE2-IP will arrange TCP data from user logic to packet format with TCP and IP header generation and then send out to EMAC. TOE2-IP will retransmit data if acknowledge returned is not correct or not arrive in time. For receive side, TOE2-IP will extract TCP data and header from IP packet and then store only TCP data in buffer for user logic reading. If receiving packet is not in a sequence, TOE2-IP will send acknowledge for request the lost packet.

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 TOE2-IP. This system demonstrates on SP605 Development board to operate with Test application on PC for transferring high speed data on network. More details are described as follows.

2. Hardware description

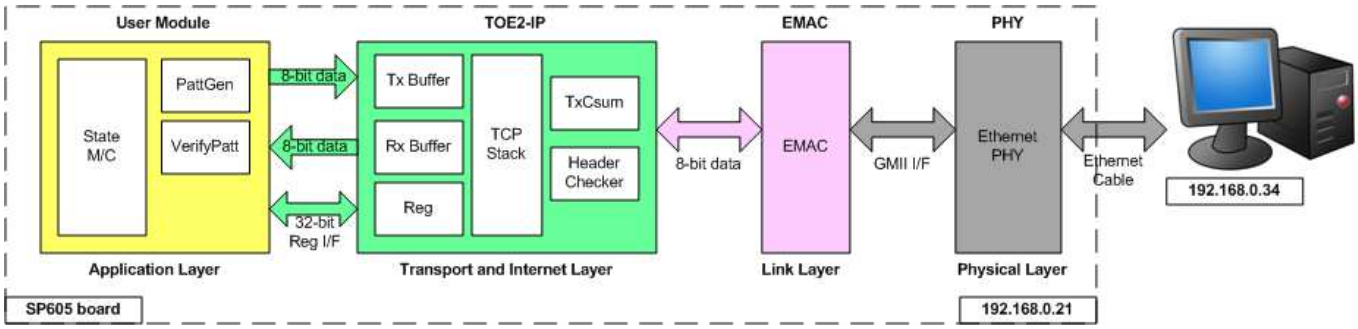

Figure 2 Hardware Architecture in reference design

As shown in Figure 2, hardware architecture can be divided into 4 modules to support each TCP/IP layer protocol. TOE2-IP operates with EMAC and external PHY to implement all four lower layers of TCP/IP Protocol. User prepares modules to transfer TCP data and write/read control signal with TOE2-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 receive side based on SP605 development board.

- External PHY

Physical layer is implemented by external PHY chip, 88E1111 Device available on SP605 development board. The interface is GMII Interface.

- EMAC

Link layer is implemented by EMAC-IP (Tri Mode Ethernet MAC with 1000 Mbps speed), provided by Xilinx. PHY Interface is GMII and MAC speed is fixed to 1000 Mbps. EMAC interface can connect with TOE2-IP directly.

- TOE2-IP

8-bit data stream from User Module is buffered within Tx Buffer. TCP and IP header will be mixed with TCP data from Tx Buffer within TOE2-IP and then send out to EMAC as 8-bit interface. On the other side, 8-bit data stream from EMAC is extracted to be TCP/IP header and TCP data by TOE2-IP. Header data will be checked and all data in packet will be rejected if header is not correct. After header is confirmed, TCP data will be stored to Rx Buffer for User Module reading out.

Signal	Dir	Clk	Description
Common Interface Signal			
RstB	In		Reset IP core. Active Low.
Clk	In		125 MHz fixed clock frequency input for user interface and MAC transmit interface.
User Interface			
RegAddr[3:0]	In	Clk	Register address bus
RegWrData[31:0]	In	Clk	Register Write data bus
RegWrEn	In	Clk	Register Write enable pulse. Assert with valid value of RegAddr and RegWrData signals.
RegRdData[31:0]	Out	Clk	Register Read data bus. Available after asserting RegAddr with 1 Clk period latency
ConnOn	Out	Clk	Connection Status (‘1’ : connection is opened, ‘0’: connection is closed)
TimerInt	Out	Clk	Timer interrupt. Assert to high for 1 Clk period when time out is detected. User can read TMO[6:0] register to check interrupt status.
Tx Data Buffer Interface			
TOETxFfFlush	Out	Clk	Transmit buffer within IP is reset. Assert to high only 1 Clk period when connection is closed or reset.
TOETxFfFull	Out	Clk	Transmit buffer full flag. User needs to stop writing data within 4 clock period after this flag is asserted to high.
TOETxFfWrEn	In	Clk	Transmit buffer write enable. Assert to write data to Transmit buffer.
TOETxFfWrData[7:0]	In	Clk	Transmit buffer write data bus. Synchronous with TOETxFfWrEn.
Rx Data Buffer Interface			
TOERxFfFlush	Out	Clk	Received buffer within IP is reset. Assert to high only 1 Clk period when connection is opened.
TOERxFfRdCnt[15:0]	Out	Clk	Received buffer data counter to show total number of received data in buffer.
TOERxFfRdEmpty	Out	Clk	Received buffer empty flag. User needs to stop reading data immediately.
TOERxFfRdEn	In	Clk	Received buffer read enable. Assert to read data from Received buffer.
TOERxFfRdData[7:0]	Out	Clk	Received buffer read data bus. Valid after TOERFfRdEn assert with 1 Clk period latency.
MAC Interface			
RxClk	In		Received clock from MAC.
MacRxGood	In	RxClk	Received packet is good status. Generated from MAC.
MaxRxBad	In	RxClk	Received packet is bad status. Generated from MAC.
MacRxData[7:0]	In	RxClk	Received data bus from MAC
MacRxDataValid	In	RxClk	Received data valid signal from MAC. Synchronous with MacRxData.
MaxTxData[7:0]	Out	Clk	Transmit data bus to MAC
MaxTxDataValid	Out	Clk	Transmit data valid to MAC
MaxTxDataAck	In	Clk	Transmit data acknowledge from MAC
MacHostOpCode[1:0]	Out	Clk	Host operation code to MAC
MacHostAddr[9:0]	Out	Clk	Host address bus to MAC
MacHostWrData[31:0]	Out	Clk	Host write data bus to MAC

Table 1 TOE2-IP Interface Signal Description

RegAddr [3:0]	Reg Name	Dir	Bit	Description
0000b	RST	Wr	[0]	Reset IP. '0': Release reset, '1': Reset. Default value is '1'. After user setting all parameters, set '0' to this register to release reset and start system parameter intialization. Reset needs to be set again if user will change value of SML, SMH, DIP, SIP, DPN, or SPN register.
0001b	CMD	Wr	[1:0]	User Command in active mode. "00": Send data, "10": Open connection (active), "11": Close connection (active), "01": Undefined. Before setting this register to start any active command, user needs to check system busy flag by reading bit[0] of this register to confirm that there is no operation running. Active command will auto-start after this register is set by user.
		Rd	[0]	System busy flag. '0': Idle, '1': System operating.
			[3:1]	Current operation. "000": Send data, "001": Idle, "010": Active open connection, "011": Active close connection, "100": Receive data, "101": Undefined, "110": Passive open connection, "111": Passive close connection.
0010b	SML	Wr	[31:0]	Define 32-bit lower MAC address (bit [31:0]) for this IP. User needs to set this register before clearing RST register.
0011b	SMH	Wr	[15:0]	Define 16-bit upper MAC address (bit [47:32]) for this IP. User needs to set this register before clearing RST register.
0100b	DIP	Wr	[31:0]	Define 32-bit target IP address. User needs to set this register before clearing RST register.
0101b	SIP	Wr	[31:0]	Define 32-bit IP address for this IP. User needs to set this register before clearing RST register.
0110b	DPN	Wr	[15:0]	Define 16-bit target port number. User needs to set this register before clearing RST register if user wants to use active open connection. Target port number will be auto defined from passive open connection packet which parameters in header are matched with setting value.
0111b	SPN	Wr	[15:0]	Define 16-bit port number for this IP. User needs to set this register before clearing RST register.
1000b	TDL	Wr	[31:0]	Total Tx data length transfer in byte unit. Valid from 1-0xFFFFFFFF. User needs to set this register before setting CMD register = "00". Value in this register will be latched. If user will send data again with same length, this register doesn't need to set again.
		Rd	[31:0]	Remain data transfer length in byte unit which still not transmit.
1001b	TMO	Wr	[31:0]	Define timeout value for waiting Rx packet during running any command. This register is used by 125 MHz counter, so timer unit is about 8 ns. This value must be more than 0x6000.
		Rd		[0]-Timeout from not receiving ARP reply packet [1]-Timeout from not receiving SYN and ACK flag during active open operation [2]-Timeout from not receiving ACK flag during passive open operation [3]-Timeout from not receiving FIN and ACK flag during active close operation [4]-Timeout from not receiving ACK flag during passive close operation [5]-Timeout from not receiving ACK flag during data transmit operation [6]-Timeout from Rx packet lost, Rx data FIFO full, or wrong sequence number [23]-Rx packet ignored because of Rx data buffer full [27]-Rx packet lost detected [30]-RST flag is detected in Rx packet
1010b	PKL	Wr	[15:0]	Data length of Tx packet in byte unit. Valid from 1-16000. Default value is 1460 byte (Maximum size for non-jumbo frame).

Table 2 Register Map in TOE2-IP Description

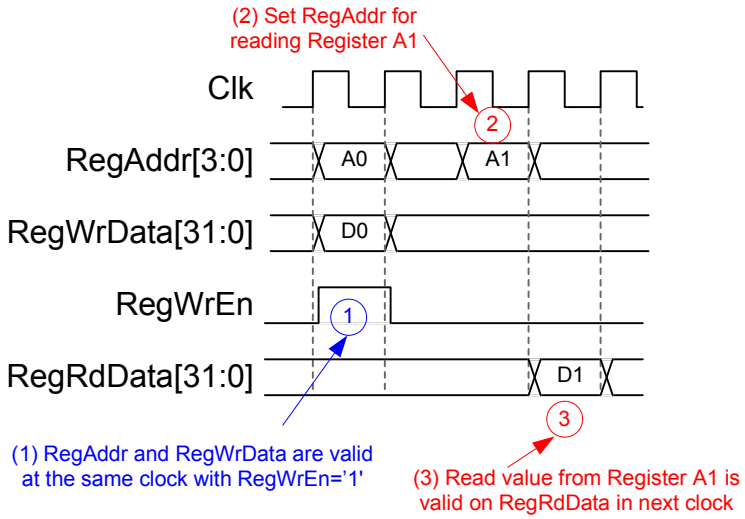

Figure 3 Waveform of Register interface

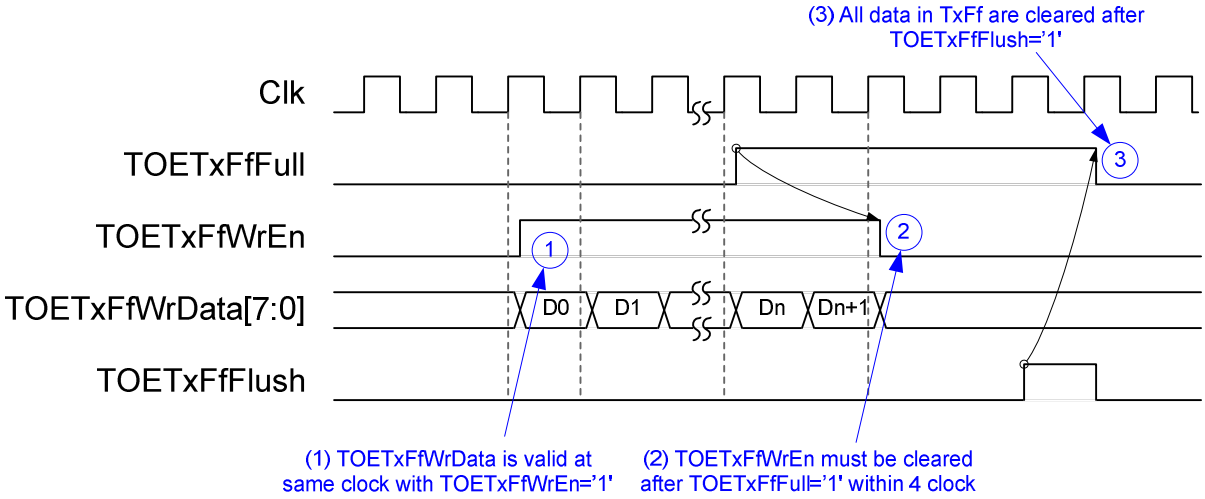

Figure 4 Waveform of data transmit transaction

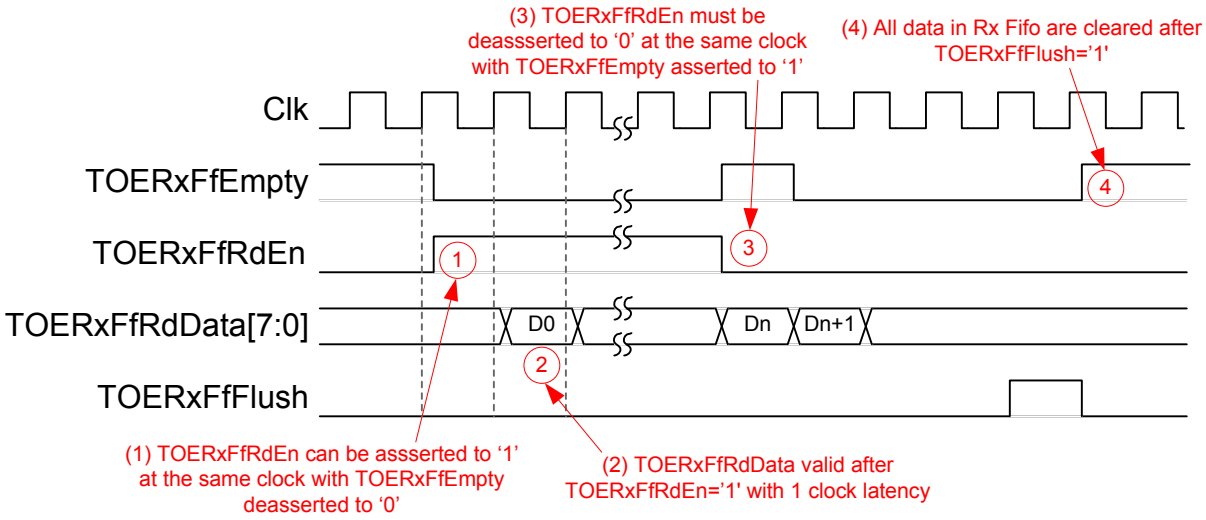

Figure 5 Waveform of receive transaction by Empty flag

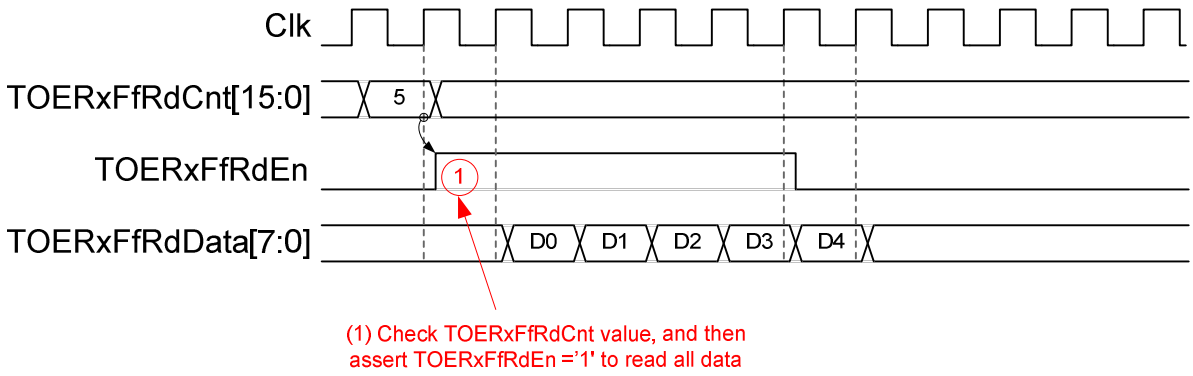

Figure 6 Waveform of receive transaction by Read Counter

- User Module

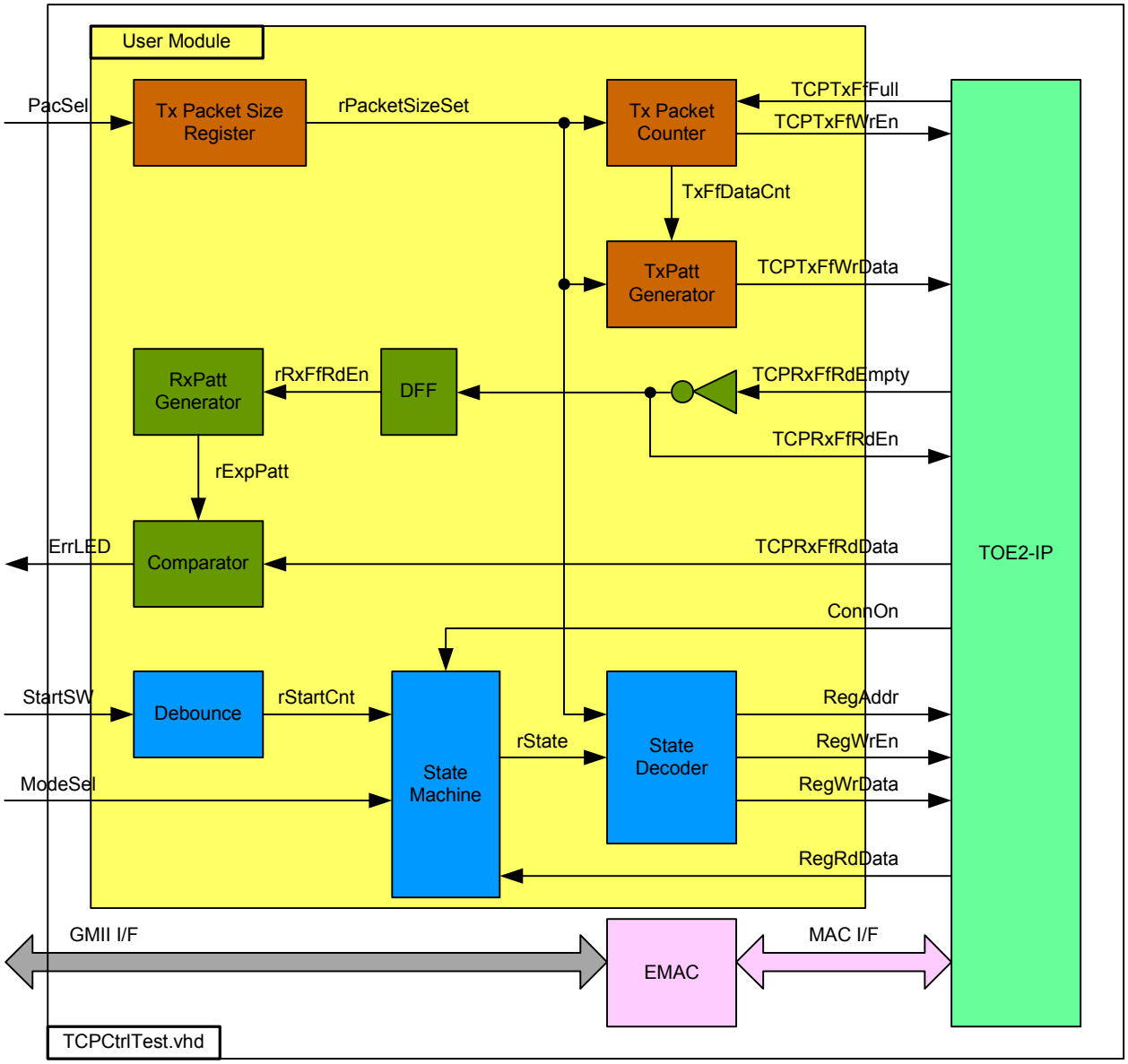

Figure 7 User Module and Test System block diagram

User Module can split into three parts, i.e. TxFIFO interface, RxFIFO interface, and Register Control interface. 32-bit increment test data is increased by 1 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. 1460 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 RxFIFO interface of TOE2-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 pushed by user. State Machine diagram is displayed as shown in Figure 8.

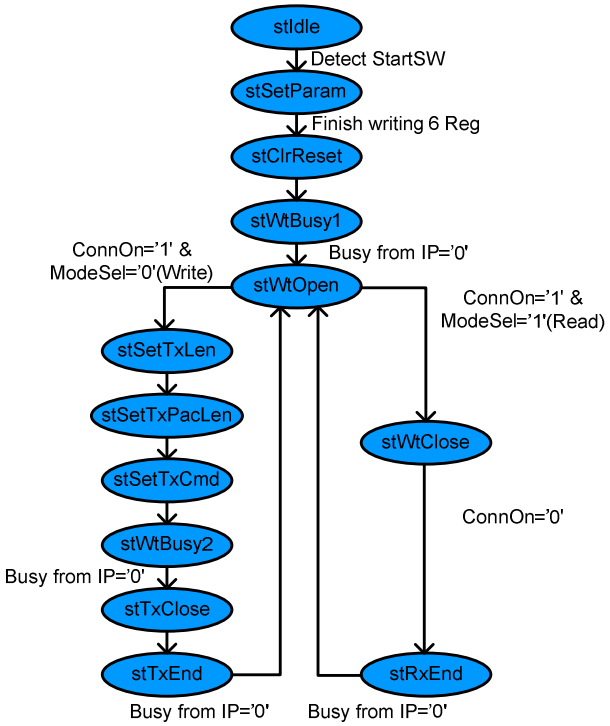

Figure 8 State Machine Diagram within User Module

State machine changes to stSetParam state after user press StartSW button. In stSetParam state, it will set parameters to register within TOE2-IP, i.e.

- Source MAC address = 00:01:02:03:04:05
- Source IP address = 192.168.0.21
- Source Port number = 4000
- Destination IP address = 192.168.0.34

Next State, stClrReset, is applied to release Reset signal within TOE2-IP and then starting parameter initialization. State machine waits initialization complete by monitoring Busy flag 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 TOE2-IP will change to '1' and then state machine will change to stSetTxLen for transmit mode or to stWtClose for receive mode.

On transmit mode, more three states, i.e. stSetTxLen, stSetTxPacLen, and stSetTxCmd are designed for setting total transfer size, packet size, and data sending command to register within TOE2-IP. Then, Busy signal is monitored to wait transfer complete in stWtBusy2 state. After all data are transferred completely, State sends command to TOE2-IP for sending packet to close connection in stTxClose state. After connection is closed and Busy = '0', state will run in stWtOpen for waiting next transfer.

On receive 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 TOE2-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.

3. Test Software description

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

- recv_tcp_client

This test application runs to test sending operation of TOE2-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 “192.168.0.21”. 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. 1460 for non-Jumbo frame mode, and 8960 for Jumbo frame mode. If setting with wrong value, verify 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 receive 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 print 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 after no any more data transfer received. 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

This test application sends data out to TOE2-IP to test receiving operation. This application requires four input parameters from user, i.e.

- FPGA IP address: This demo sets IP address to “192.168.0.21”. 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 16kByte unit. Total transfer size is equal to this value x 16kByte.
- 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. In some PC, performance in dummy data mode is better than increment data mode.

The operation sequence of the application is follows.

(1) Get three parameters from user.
(2) Create socket and then set properties of send 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.

4. Revision History

Revision	Date	Description
1.0	11-Dec-12	Initial Release