TOE1G-IP Two-Port Demo Reference Design Manual

Rev1.3 2-Sep-16

1 Introduction

Two-port demo implements more than one TCP connection by using TOE1G-IP and CPU firmware. In general system, at least two TCP ports are required, one for data transfer (fast connection) and another for control signal (slow connection). In the demo, fast connection is implemented by TOE1G-IP and slow connection is implemented by CPU firmware and simple logic.

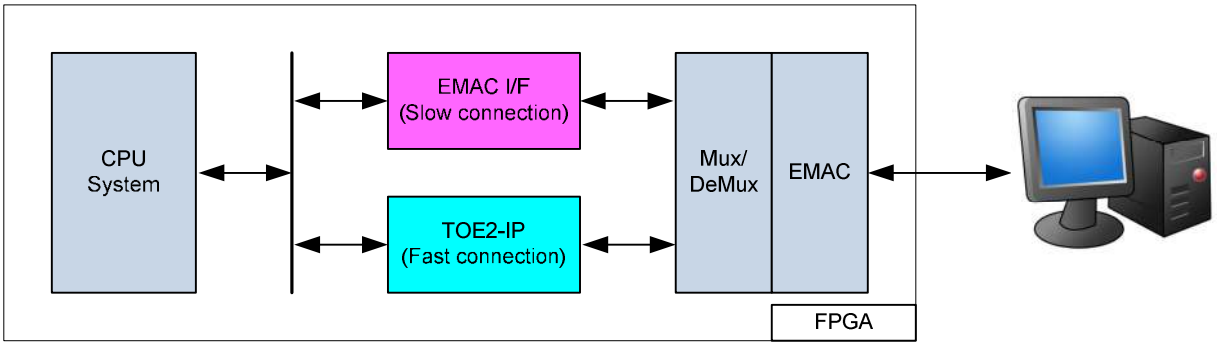

Figure 1 Two-port Hardware Structure

User can select test operation through Serial console. There are three test modes in the demo, i.e. receive data through fast connection, send data through fast connection, and receive/send data through slow connection. For fast connection test, special test application provided by Design Gateway is used to run on TestPC. For slow connection test, ping command is used to test. Ping command uses Echo reply and request type of ICMP protocol to send the request and wait response.

Otherwise, FTP server demo is also developed by using same hardware design, but modifying only CPU firmware. So, by using two-port hardware system, user can develop many applications by modifying only the firmware. More details about hardware are described in the next topic.

2 Hardware Structure

As shown in Figure 2, the hardware of two port demo is split into two blocks, i.e. AXITOE1G for fast connection and AXIEMAC for slow connection. The hardware can separate the packet from EMAC by using different port number. Port number of each connection is programmed by CPU firmware through register access.

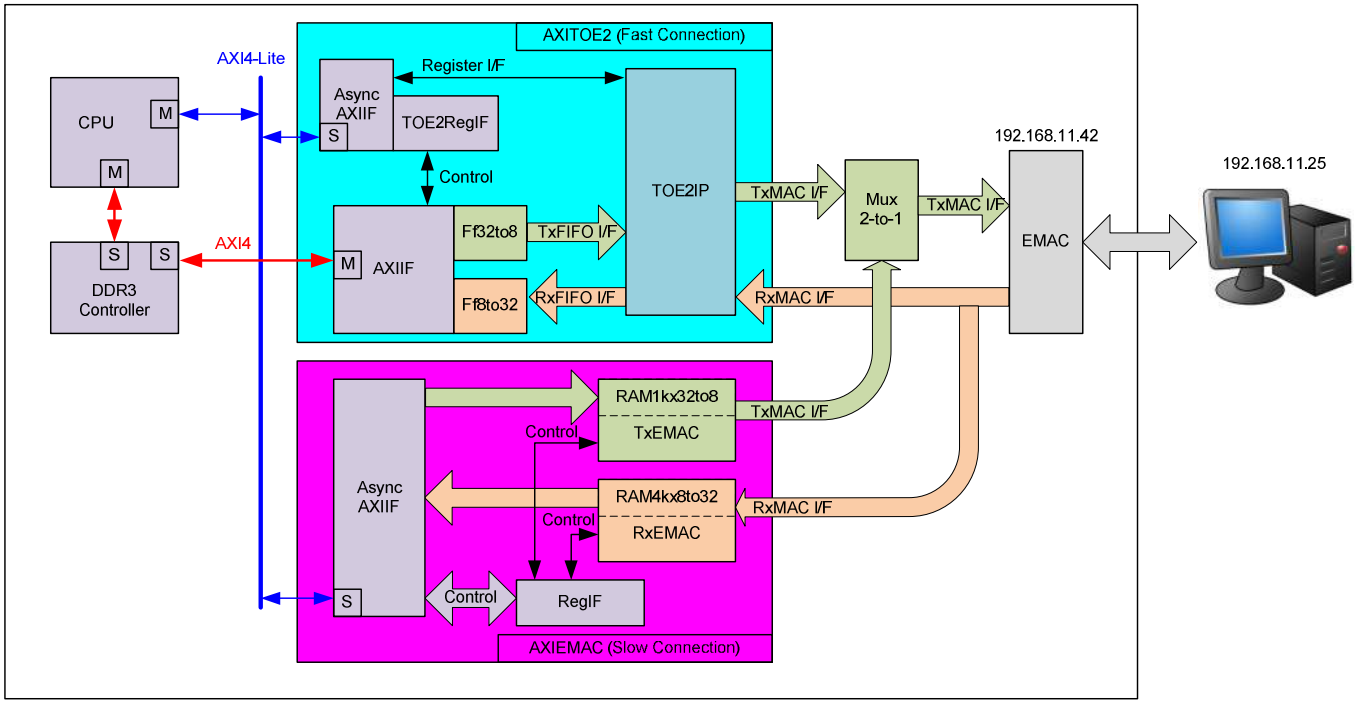

Figure 2 Two-port Demo Hardware Block Diagram

AsyncAXIIF module is used to convert AXI4-Lite interface in AXI clock domain to be register interface in another clock domain.

For fast connection, data is burst transferred with DDR3 memory directly through AXI4 bus interface which is suitable for high bandwidth transfer. DMA engine is designed within AXIIF module to burst transfer from/to DDR3 to/from Ff32to8, depending on transfer direction. AsyncAXIIF is mapped to two register areas, i.e. TOE1GIP and TOE1GRegIF, so CPU can access control/status signal of TOE1GIP and AXIIF through AXI4-Lite bus.

For slow connection, AsyncAXIIF module is mapped to three register areas, i.e. TxEMAC, RxEMAC, and RegIF. Hardware is designed to bypass Ethernet packet between EMAC and CPU by using RAM to be data buffer. For transmit side, TxEMAC is designed to dump data from RAM to EMAC. For received side, it includes packet header filtering which can be programmed by CPU. So, only the packet which has valid header will be stored to the RAM for CPU processing. Since the packet of slow connection is Ethernet packet without adding/removing TCP/IP header, CPU firmware needs to implement TCP/IP stack for constructing and decoding TCP packet. Slow connection does not require high bandwidth, so AXI4-Lite bus is used to access control register and RAM. RegI/F stores all control/status signals for this connection.

2.1 AXITOE1G

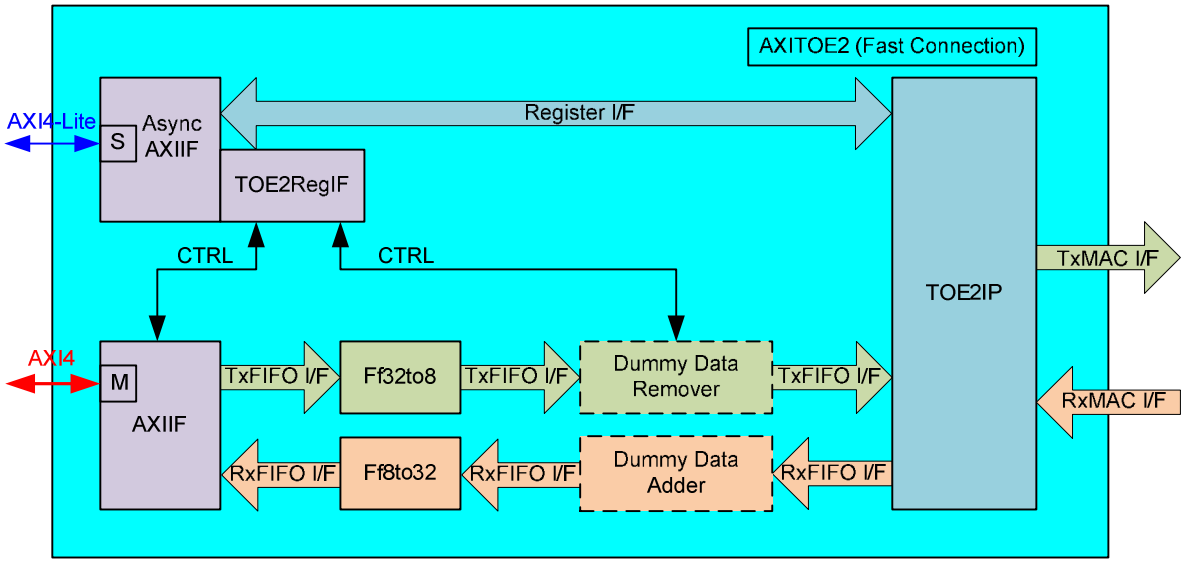

Figure 3 AXITOE1G Block Diagram

AXITOE1G module includes TOE1G-IP to handle TCP/IP packet for fast connection. For high-performance and simple design, DMA engine within AXIIF fixes burst transfer to be 512-byte by using 128-beatx32-bit. Since total data length setting from CPU may not align 512-byte, the logic to remove dummy data for transmit side and to add dummy data for received side are designed between FIFO and TOE1G-IP module. Ff32to8 and Ff8to32 are designed to be data buffer and also used for converting data bus size between 32-bit of AXI4 bus and 8-bit of TOE1G-IP interface. To support different clock domain between AXI4 and TOE1G-IP, both FIFOs are asynchronous type. The details of sub-module inside AXITOE1G are described as follows.

2.1.1 TOE1G-IP

The IP includes simple TCP/IP stack, and TCP offload engine for both TX and RX side to calculate checksum of TCP packet. More details about TOE1G-IP have been described in TOE1G-IP datasheet.

2.1.2 AXIIF

DMA engine within AXIIF is designed by using state machine. Since AXIr interface (AXI4 -> TxFIFO) and AXIw interface (RxFIFO->AXI4) can run independently, two state machines are designed to control each interface independently.

For transmit side (AXIr interface), after receiving start signal from CPU, state machine will check remain space in Ff32to8 which must be more than 512-byte. If FIFO space area is enough, 512-byte read request will be generated to AXI4 bus. After that, AXI4 bus returns 512-byte data to store to Ff32to8. State machine repeats to check FIFO status step until waiting end of each 512-byte data transfer step. Repeat step will be stopped when complete total size transfer.

For received side (AXIw interface), after receiving start signal from CPU, state machine will wait until at least 512-byte data is available in Ff8to32. Next, state machine generates the write request to AXI4 bus. After AXI4 returns request acknowledgement, 512-byte data will be burst transferred from Ff8to32 to AXI4. Similar to AXIr, state machine runs in the loop from FIFO status checking to data sending until complete total request size.

2.1.3 AsyncAXIIF

This module is used to convert AXI4-Lite I/F to be register interface which both run in different clock domain. Memory map of register interface is mapped to TOE1G-IP and other control/status signals within AXITOE1G module. More details are shown in Table 1.

2.1.4 TOE1GRegIF

This module is used to decode address of control/status signals for DMA engine and data adder/remover. Start signal and transfer length for both directions are generated in both clock domains (AXI clock for AXIIF operation, and TOE1GIP clock for data remover/adder logic).

Address Rd/Wr	Register Name Label in "ftp_demo.c/twoport.c"	Description
BA+0x80000 – BA+0x8002B: TOE1G-IP Register Area		
BA+0x80000 Wr	RST Reg of TOE1G-IP (TOE1G_RST_REG)	Please see more details in "Table 2: Register map Definition" of TOE1G-IP datasheet. Address of the demo is byte unit while register address in TOE1G-IP datasheet is 32-bit unit.
BA+0x80004 Wr/Rd	CMD Reg of TOE1G-IP (TOE1G_CMD_REG)	
BA+0x80008 Wr	SML Reg of TOE1G-IP (TOE1G_SML_REG)	
BA+0x8000C Wr	SMH Reg of TOE1G-IP (TOE1G_SMH_REG)	
BA+0x80010 Wr	DIP Reg of TOE1G-IP (TOE1G_DIP_REG)	
BA+0x80014 Wr	SIP Reg of TOE1G-IP (TOE1G_SIP_REG)	
BA+0x80018 Wr	DPN Reg of TOE1G-IP (TOE1G_DPN_REG)	
BA+0x8001C Wr	SPN Reg of TOE1G-IP (TOE1G_SPN_REG)	
BA+0x80020 Wr/Rd	TDL Reg of TOE1G-IP (TOE1G_TDL_REG)	
BA+0x80024 Wr/Rd	TMO Reg of TOE1G-IP (TOE1G_TMO_REG)	
BA+0x80028 Wr	PKL Reg of TOE1G-IP (TOE1G_PKL_REG)	

Address Rd/Wr	Register Name Label in “ftp_demo.c/twoport.c”	Description
BA+0x80100 – BA+0x8011B: Other control/status of AXITOE1G		
BA+0x80100 Wr/Rd	Start Transmit DDR Address (TX_DDR_ADDR)	Wr [31:0] – Start DDR Address for transmit from DDR to TOE1G-IP (Must be aligned to 32-bit or bit[1:0] must be equal to “00”). Rd [31:0] – Current DDR Address for transmit from DDR to TOE1G-IP
BA+0x80104 Wr	Transmit Length (TX_DDR_LEN)	Wr [31:0] – Total transmit data size from DDR to TOE1G-IP in byte unit
BA+0x80108 Wr/Rd	Start Receive DDR Address (RX_DDR_ADDR)	Wr [31:0] – Start DDR Address for receive from TOE1G-IP to DDR (Must be aligned to 32-bit or bit[1:0] must be equal to “00”). Rd [31:0] – Current DDR Address for receive from TOE1G-IP to DDR
BA+0x8010C Wr	Receive Length (RX_DDR_LEN)	Wr [31:0] – Total receive data size from TOE1G-IP to DDR in byte unit
BA+0x80110 Wr/Rd	AXITOE1G Control (AXITOE1G_CTRL)	Wr [0] – Start transmit data from DDR to TOE1G-IP. Auto clear. [1] – Start to receive data from TOE1G-IP to DDR. Auto clear. Rd [0] – Busy flag from DDR -> TOE1GIP process [1] – Busy flag from TOE1GIP -> DDR process
BA+0x80114 Rd	ConnOn of TOE1G-IP CONNON_REG	Rd [0] – Mapped to ConnOn signal output from TOE1G-IP to show port status
BA+0x80118 Rd	FIFO Read Count of TOE1G-IP FFRDCNT_REG	Rd [15:0] – Mapped to TCPRxFfRdCnt signal output from TOE1G-IP to show total number of received data in byte unit

Table 1 Register Map in TOE1GRegIF

Note: BA is base address of AXI4-Lite interface which is different value on each FPGA platform.

2.2 AXIEMAC

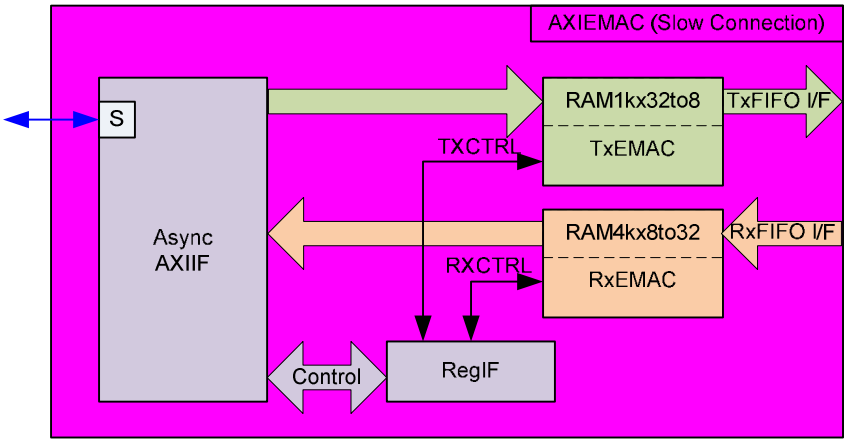

Figure 4 AXIEMAC Block Diagram

The slow connection is designed to transfer control/status information which is small packet and does not sent more frequently. The memory map of AXIEMAC is split into three areas, i.e. RAM of TxEMAC (TxRAM), RAM of RxEMAC (RxRAM), and general register as shown in Table 2. TxRAM and RxRAM store Ethernet packet which is transferred between CPU and EMAC.

Address Rd/Wr	Register Name (Label in the "ftp_demo.c")	Description
BA+0x00000- BA+0x00FFF Wr	RAM in TxEMAC (TXRAM)	RAM to store transmit Ethernet data of control connection
BA+0x10000- BA+0x10FFF Rd	RAM in RxEMAC (RXRAM)	RAM to store received Ethernet data of control connection
BA+0x20000 Wr	Destination MAC address [31:0] (DSTMACL_REG)	Lower 32-bit MAC address value for FPGA
BA+0x20004 Wr	Destination MAC address [47:32] (DSTMACH_REG)	Upper 16-bit MAC address value for FPGA
BA+0x20008 Wr	Destination IP address (DSTIP_REG)	32-bit IP address value for FPGA
BA+0x20010 Wr	Rx Pattern enable (RXPATTEN_REG)	Enable to compare header of received packet ('1': enable, '0': disable) [0] – Enable to compare byte23 with RXPATT23_REG[7:0] [1] – Enable to compare byte36 with RXPATT32_REG[7:0] [2] – Enable to compare byte37 with RXPATT32_REG[15:8]
BA+0x20014 Wr	Rx Pattern Byte 23 (RXPATT23_REG)	[7:0] Pattern to compare with byte23 of received packet
BA+0x20018 Wr	Rx Pattern Byte 34 (RXPATT36_REG)	[15:0] Pattern to compare with byte36 – byte 37 of received packet
BA+0x20100 Wr	Start TXRAM address (TXADDR_REG)	Start address of TXRAM to transmit Ethernet data
BA+0x20104 Wr/Rd	Transmit length (TXLEN_REG)	Wr: Transfer length of transmit Ethernet data in byte unit. Valid form 2-4095. Rd: Remain transmit length in byte unit. Polling this register=0 for monitoring that data can transmit completely.
BA+0x20200 Rd	Received RXRAM address (RXADDR_REG)	Current end address of received packet. Compare this value with previous value to calculate total received data size in byte unit.

Table 2 Memory Map of RegIF within AXIEMAC module

Note: BA is base address of AXI4-Lite interface which is different value on each FPGA platform.

2.2.1 TxEMAC

Similar to Ff32to8 inside AXITOE1G, TxRAM (RAM1kx32to8) is used to be data buffer and convert 32-bit data from AXI4-Lite to 8-bit data. Small data transmitter is designed to read data from RAM and arrange into the packet for sending to EMAC. The value of packet size and start RAM address for 1st data are programmed by CPU through RegIF module. Also, CPU can monitor remaining transfer size to check transfer progress through register access.

RAM1kx32to8 is simple dual-port RAM. Write interface is 32-bit bus size and mapped to address = BA – (BA+0xFFF) on CPU system. Read interface is 8-bit bus size and connected to internal logic within TxEMAC module.

2.2.2 RxEMAC

Similar to Ff8to32 inside AXITOE1G, RxRAM (RAM4kx8to32) is used to be data buffer and convert 8-bit data from EMAC I/F to 32-bit data for CPU. The logic includes header packet filtering feature to filter only valid packet for CPU. The header pattern for filtering function consists of the constant value inside the logic and the value which can be programmed by CPU.

The constant value inside the logic is the IP version and IP header length. IP version must be IPv4 and IP header length must be 20 bytes. Programmable value from CPU is Destination MAC address, Destination IP address, Protocol value, and Destination port number. Protocol value and destination port number can be selected to enable/disable pattern verification while other values are designed to enable mode only.

To support ping command, the filter header is programmed to be following value.

- Destination MAC address = broadcast ID or set value from CPU
- Destination IP address of = set value from CPU
- Protocol = ICMP (RXPATT23_REG=0x01, RXPATTEN_REG[0]=’1’)
- Destination port number = Disable (RXPATTEN_REG[2:1]=”00”)

From above setting, only ICMP packet will be stored to RxRAM.

RAM4kx8to32 is simple dual-port RAM. Write interface is 8-bit bus size and connected to EMAC I/F. Read interface is 32-bit bus size and mapped to address = (BA+0x10000) – (BA+0x10FFF) on CPU system.

Write address of RxRAM is auto-increment by internal logic to store received packet continuously, so CPU must remember current read address of the last packet to be start read address of next valid packet. By comparing the write address with read address of RAM, CPU can calculate the size of received packet.

2.2.3 RegIF

This module is designed to decode address and data for controlling TxEMAC and RxEMAC operation and monitoring the current status.

3 CPU Firmware

Three test operations are designed in CPU Firmware, i.e.
(1) Transmit data test through fast connection
(2) Receive data test through fast connection
(3) Ping command test through slow connection

User can select the operation through Serial console. So, the sequence of CPU operation for two-port demos is follows.
(1) Initialize network parameters such as MAC address, IP address, and Port number to TOE1G-IP and AXIEMAC module.
(2) Wait until TOE1G-IP initialization complete.
(3) Receive all user inputs to start the test operation.
(4) Set the parameter from user to the hardware and start operation.
(5) Since FPGA board runs in Server mode, the port will be opened by test application on TestPC after running the application.
(6) Run the test until complete.

For fast connection, the upper 512 MB area is used to be data buffer for transmit/receive test. So, if transfer length value from user is more than 512 MB, data verification feature will not be available. Maximum transfer size of fast connection test is 4 GByte.

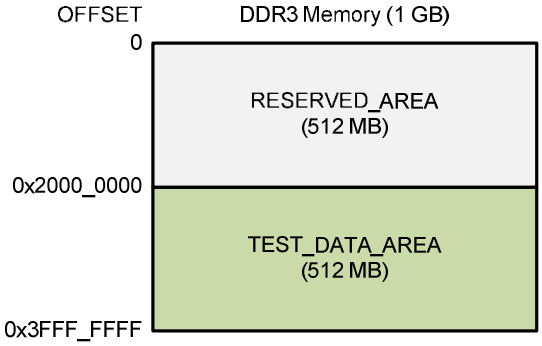

Figure 5 DDR3 Memory Map

3.1 Ping Command Test

Ping command can be used to test for both transmit and received side on slow connection. IP Datagram of ICMP protocol for ping command is shown in Figure 6.

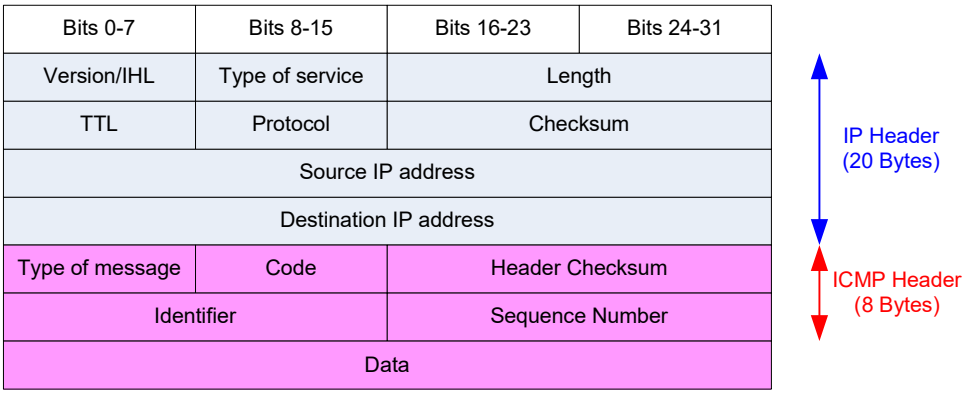

Figure 6 IP Datagram for ICMP

For ping command test, TestPC sends Echo request (Type=8) to FPGA, and FPGA generates Echo reply (Type=0) back to TestPC. All data of Echo reply are copied from Echo request. More information about ping command can be checked from below website.
http://en.wikipedia.org/wiki/Ping_(networking_utility)

The sequence of firmware is follows.

1) Polling RXADDR_REG to monitor that received data available in RXRAM.
2) Dump received packet to received temp buffer on firmware. Only 256-byte size is reserved by defining “TMPBUF_SIZE” parameter value.
3) Calculate IP and ICMP checksum to check that packet is valid. Return error when checksum is not correct.
4) Prepare Echo reply packet in TxRAM. Data is copied from the received packet, but IP and ICMP checksum must be re-calculated.
5) Set register to start data sending by AXIEMAC module which is slow connection.

3.2 Receive Data Test

In this test, test data will be transferred from TestPC by using “send_tcp_client” application to FPGA through fast connection. 512 MB area of DDR3 is used to be data buffer, so data verification function of the firmware will be disabled when total transfer length is more than 512 MB area. The sequence of firmware is follows.

1) Wait the connection on by monitoring CONNON_REG. CONNON_REG will be set after TestPC runs “send_tcp_client” application.
2) Wait until data is available in received FIFO of TOE1G-IP.
3) Calculate DDR3 start address and transfer size, and set the result to AXITOE1G register with start transfer flag. Then, data will be dump from TOE1G-IP to DDR3.
4) Wait until complete data transfer.
5) Step 2) – 4) runs in the loop until the connection is closed by TestPC.
6) If total data length is not more than 512 MB, the menu to start data verification will be displayed.

3.3 Transmit Data Test

In this test, test data will be transferred from FPGA to TestPC. “recv_tcp_client_single” application is used to run on TestPC to receive/verify data from FPGA. Data can be verified on TestPC only when transfer length is not more than 512 MB, same as Receive Data Test. The sequence of firmware is follows.

1) Fill test data to DDR3.
2) Set transfer size and packet length which are user inputs to TOE1G-IP register. Then, set register to start TOE1G-IP operation.
3) If total transfer size is more than 512 MB, the transfer size of AXIIF will be split into more than one loop to avoid DDR3 address overlap. Transfer size and DDR3 start address will be re-calculated for each loop.
4) Wait until complete all data transfer, and then set TOE1G-IP register to close connection.
5) Wait until the connection is closed.

4 Necessary consideration

For simple design, fast connection and slow connection test cannot operate at the same time. But user can modify the firmware to support running fast and slow connection at the same time. Please see the example design which run fast and slow connection together from FTP Server demo.

5 Revision History

Revision	Date	Description
1.0	9-Jan-15	Initial version release
1.1	28-Oct-15	Correct Table1 description
1.2	29-Dec-15	Add asynchronous clock support
1.3	2-Sep-16	IP core product renamed from TOE2-IP to TOE1G-IP