

TOE10G-IP Multisession Reference design manual

Rev1.0 18-Nov-16

1. Introduction

It is recommended to read “dg_toe10gip_refdesign_altera_en.pdf” document which is standard demo of TOE10G-IP firstly. It will help the user to understand the basic operation of TOE10G-IP. Multi-session demo is designed by implementing eight TOE10G-IPs to support data transfer by using eight sessions at the same time. UserCtrl module is same design as standard demo, but EMAC multiplexer is additional designed to share EMAC-IP to all TOE10G-IPs. More details are described as follows.

2. Environment

This reference design is based on the following environment as shown in Figure 1.

- Arria10 SoC development board
- QuartusII Programmer 15.1 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 “tcpdatatest.exe”, provided by Design Gateway

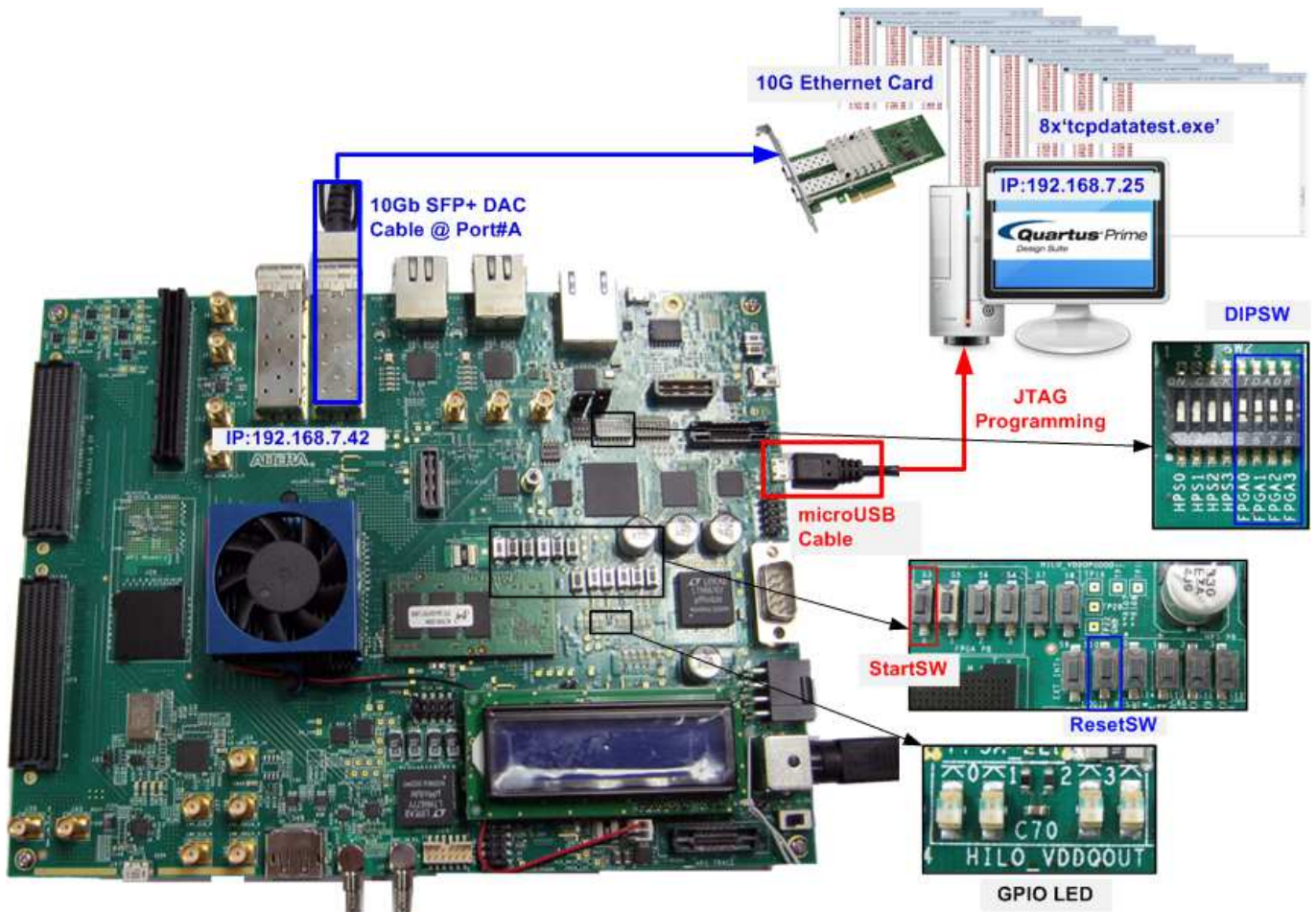


Figure 1 TOE10GIP Multi-session Demo on FPGA board

3. Hardware description

While RAM size of TOE10GIP in the standard demo is set to maximum value to achieve the best performance, RAM size in this demo is set to use minimum resource instead to support maximum numbers of TCP session. The size of the buffer inside TOE10GIP is equal to 4 Kbyte. In the demo, user can select data transfer direction of each session independently through DIPSW. By setting different port number for each TOE10GIP, all sessions can transfer data with PC at the same time.

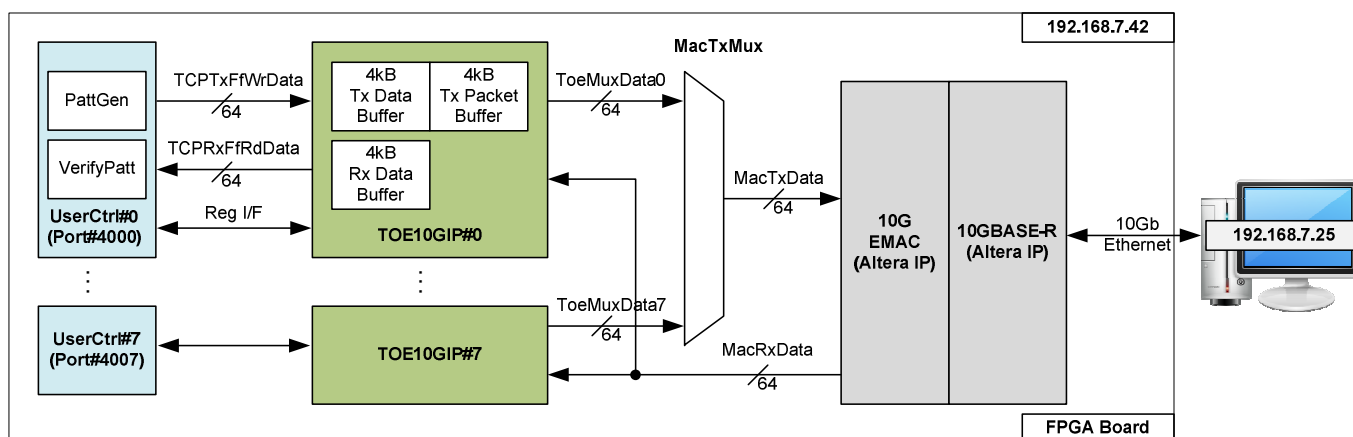


Figure 2 Hardware Architecture in reference design

- 10GBASE-R IP

This is implemented by using Arria10 Transceiver Native PHY IP and setting configuration rules = 10GBASE-R. Please see more details from following website.

http://www.altera.com/literature/hb/arria-10/ug_arria10_xcvr_phy.pdf

- 10GEMAC

This is 10Gb Ethernet MAC IP core provided by Altera. User can read more details from following website. http://www.altera.com/literature/ug/ug_32b_10g_ethernet_mac.pdf

- MacTxMux

This module is the arbiter to select data from one of eight TOE10GIPs to send to 10GEMAC. The active channel will be changed when end of current packet and other channels assert data valid signal to send the new packet.

There is no de-multiplexer for Rx interface of EMAC. Rx interface of EMAC will connect to all TOE10GIPs directly because TOE10G-IP has the logic to filter the received packet. Only packet which has matched header will be processed.

- TOE10G-IP

RAM size within the IP in the reference design is set to be 4 Kbyte. From our test environment, transfer speed when running one channel by using 4Kbyte RAM is about 200-250 Mbyte/sec. This performance is about 1/4 of maximum speed which is achieved when using 64 Kbyte RAM. So, when running eight sessions at the same time by using 4 Kbyte RAM, total bandwidth will be almost equal to the maximum performance.

- UserCtrl

HDL code of this module is same as the design of standard demo. More details of the module are described in

http://www.dgway.com/products/IP/TOE10G-IP/dg_toe10gip_refdesign_altera_en.pdf

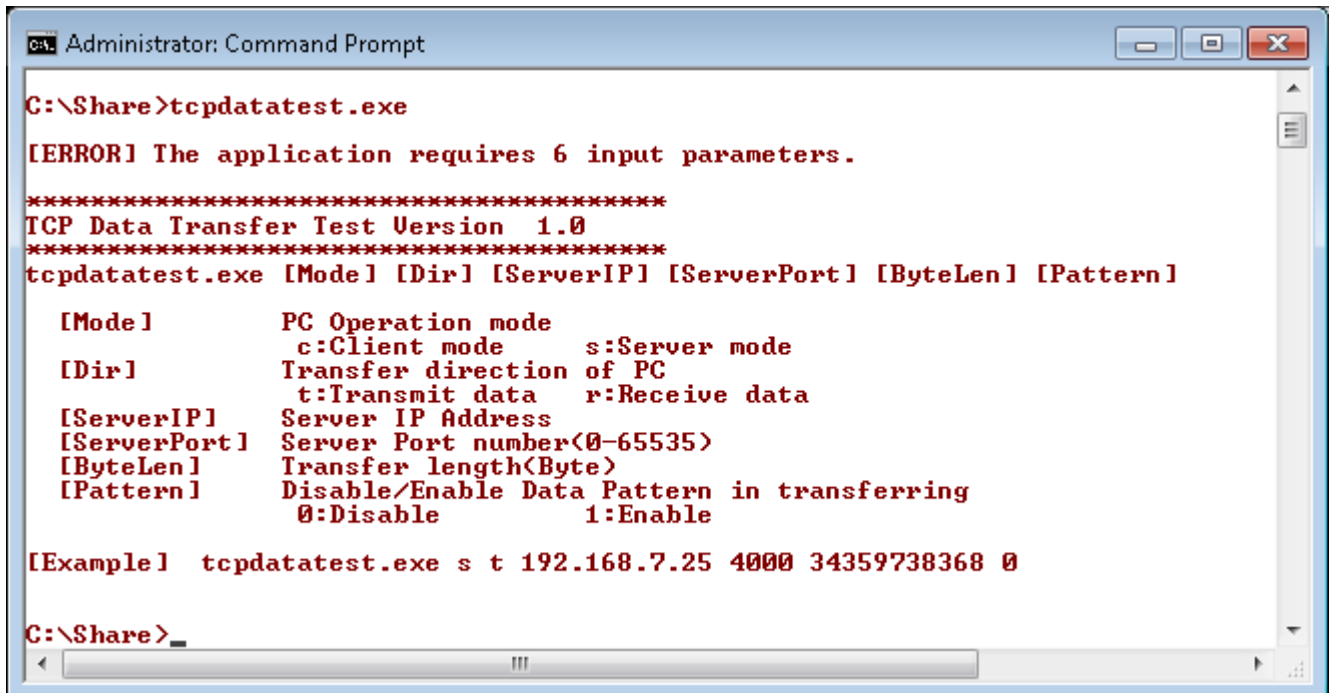
Comparing to standard demo, multisession demo will fix running mode to send data only non-jumbo frame mode and to receive data with enable data verification. DIPSW are used for selecting transfer direction only.

All parameters except port number are similar to standard demo. To support multi-session, the port number of each IP will be different. First port value is 4000 and increased by 1 for the next port (IP#0 uses port#4000, IP#1 uses port#4001, ..., and IP#7 uses port#4007).

Since RAM size of the IP is only 4Kbyte, WIN register of TOE10GIP is set to 1Kbyte size to allow the IP sending the windows update packet after user logic completes to verify every 1Kbyte data. Windows update packet is used to show the available size of received buffer inside the IP. So, test application can continue to send the new packet to the IP when available size is enough for new packet.

The transfer direction of UserCtrl is setting from DIPSW. Four bits of DIPSW are used in the demo to select data direction of eight channels. So, one bit will be used to set transfer direction for two sessions.

4. Test Software description



```

Administrator: Command Prompt

C:\Share>tcpdatatest.exe

[ERROR] The application requires 6 input parameters.

*****
TCP Data Transfer Test Version 1.0
*****
tcpdatatest.exe [Mode] [Dir] [ServerIP] [ServerPort] [ByteLen] [Pattern]

[Mode]      PC Operation mode
             c:Client mode    s:Server mode
[Dir]       Transfer direction of PC
             t:Transmit data  r:Receive data
[ServerIP]  Server IP Address
[ServerPort] Server Port number(0-65535)
[ByteLen]   Transfer length(Byte)
[Pattern]   Disable/Enable Data Pattern in transferring
             0:Disable       1:Enable

[Example] tcpdatatest.exe s t 192.168.7.25 4000 34359738368 0

C:\Share>

```

Figure 3 Test application usage

“tcpdatatest” is designed to run on PC for sending/receiving TCP data through Ethernet for both Server and Client mode. User can input parameter to select transfer direction and the mode. Six parameters are required, i.e.

- 1) Mode: c – when PC runs in Client mode and FPGA runs in Server mode
- 2) Dir: t – when PC sends data to FPGA
r – when PC receives data from FPGA
- 3) ServerIP: IP address of FPGA when PC runs in client mode. Valid value for hardware in the reference design is 192.168.7.42.
- 4) ServerPort: Port number of FPGA when PC runs in client mode. Valid value for hardware in the reference design is 4000 – 4007.
- 5) ByteLen: Total transfer size in byte unit. This input is used in transmit mode only and is ignored in receive mode. In receive mode, application will not know received data size and end operation when connection is destroyed from sender.
- 6) Pattern:
 - 0 – Generate dummy data in transmit mode/Disable data verification in receive mode.
 - 1 – Generate increment data in transmit mode/Enable data verification in receive mode.

4.1 Transmit data mode

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

- 1) Allocate 1 MB memory to be send buffer.
- 2) Create socket and set properties of send buffer.
- 3) Create new connection
 - a) For Client mode, create new connection to server by using IP address and port number from user.
- 4) Generate increment test pattern to send buffer when test pattern is enabled. Skip this step if dummy pattern is selected.
- 5) Send data out and decrease remaining transfer size.
- 6) Print total transfer size every second.
- 7) Run step 4) – 6) in the loop until remaining transfer size is 0.
- 8) Close socket and print total size and performance.

4.2 Receive data mode

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

- 1) Allocate 1 MB memory to be received buffer.
- 2) Create socket and set properties of received buffer.
- 3) Same step as step3) in Transmit data mode.
- 4) Read data from received buffer and increase total received data size.
- 5) If verification is enabled, data will be verified with increment pattern and error message will be printed out when data is not correct. Skip this step if data verification is disabled.
- 6) Print total transfer size every second.
- 7) Run step 4) – 6) in the loop until connection status is closed.
- 8) Close socket and print total size and performance.



5. Revision History

Revision	Date	Description
1.0	18-Nov-16	Initial Release

Copyright: 2016 Design Gateway Co,Ltd.