IPCTEK

TCP-UDP-IP STACK 10G RTL EXAMPLE DESIGN

Example design user guide

UG001

Version 1.0

December 04, 2021

Table of Contents

Introduction

The purpose of this example design is to demonstrate the performance of the IPCTEK's TCP/UDP/IP stack 10G FPGA IP core. In the example design, a fully functional Ten-Gigabit TCP/UDP/IP stack is implemented on Xilinx's evaluation boards. Main characteristics of the IP core are given in the table below.

Parameter	Value	Unit
MAC MTU	9000	Bytes
Number of entries in the routing table	64	
Number of TCP server (or client)	1	
TCP TX buffer size	32	KBytes
TCP RX buffer size	32	KBytes
TCP out-of-order Handling Enable	"TRUE"	
Out-of-order reordering buffer size	32	KBytes
Number of UDP TX	1	
Number of UDP RX	1	

Table 1 - IP core's parameters

Example design source code download

In order to compile the example design we need to download the Vivado Hdl firmware, the SDK bare-metal application and the Qt test bench source code. All the design source code and an evaluation netlist can be requested by sending an e-mail to contact@ipctek.net

- **Vivado Hdl firmware**: this repository (**ip_stack_10g/**) contains necessary source code and scripts to generate the FPGA firmware Vivado project.
- **SDK bare-metal application project**: the example design is running on a Microblaze-based subsystem. The source code is in **drivers/** folder.
- **Qt-based test bench (optional)**: an UDP transceiver and a TCP server/client test bench which can be run on a PC host to interact with the FPGA firmware. The source code is in **Qt/** folder.

Firmware generation

In this section we detail the step-by-step process in order to generate the FPGA firmware used in the example design.

Firmware synoptic

The FPGA firmware synoptic is illustrated in the figure below. In the design, we implement a Ten Gigabit TCP/UDP/IP stack which will be running on Xilinx's evaluation boards. We use the Xilinx's 10G/25G Ethernet Subsystem IP serving as the Ethernet MAC along with the PCS/PMA stack.

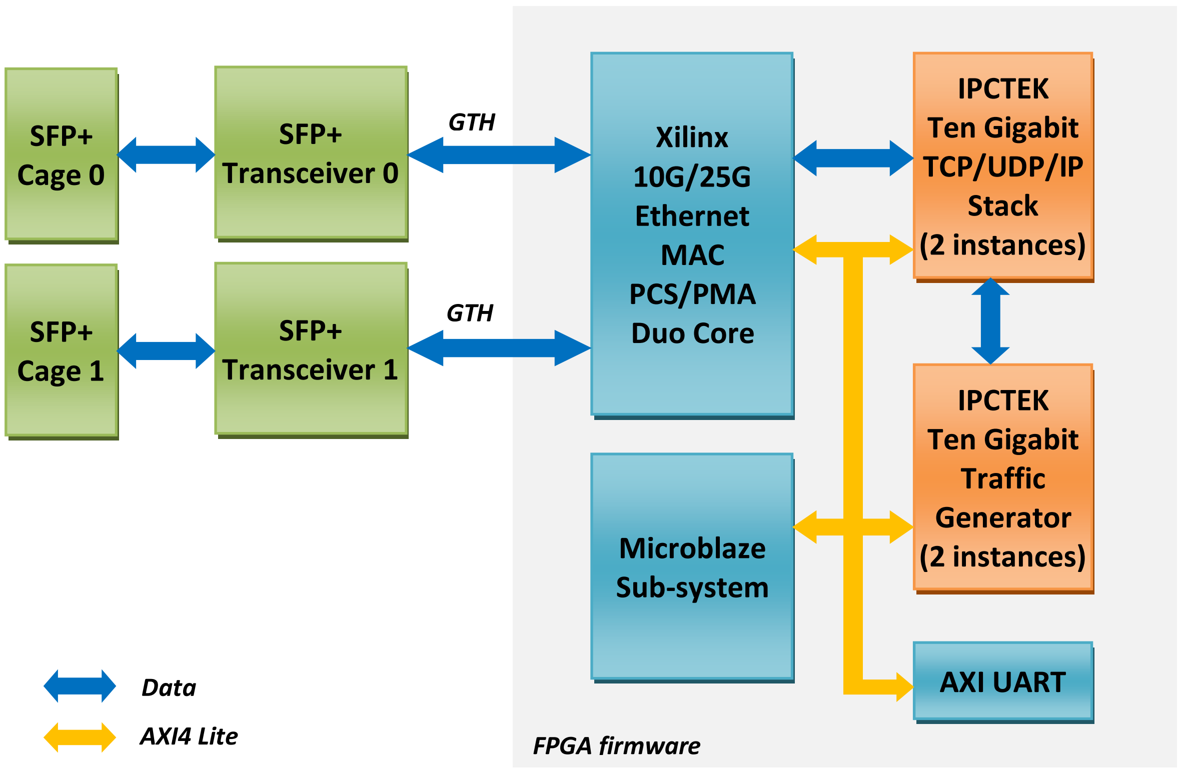

SFP+ Cage 0
SFP+ Transceiver 0
GTH
SFP+ Cage 1
SFP+ Transceiver 1
GTH
Xilinx 10G/25G Ethernet MAC PCS/PMA Duo Core
IPCTEK Ten Gigabit TCP/UDP/IP Stack (2 instances)
IPCTEK Ten Gigabit Traffic Generator (2 instances)
Microblaze Sub-system
AXI UART
Data
AXI4 Lite
FPGA firmware

Figure 1 - FPGA firmware synoptic

The FPGA firmware is composed of following principal components:

- **10G Ethernet MAC**: A Xilinx's 10G/25G Ethernet Subsystem is used. The IP is configured with two cores in order to interface with two different SFP+ transceivers.
- **IPCTEK's Ten-Gigabit TCP/UDP/IP stack**: One TCP engine (can be switched between a server and a client), one UDP transmitter and one UDP receiver are instantiated in the IP core. The MAC MTU is equal to 9000 bytes. The IP core is embedded in an AXI4-Lite wrapper for an easy integration within a Microblaze sub-system. Two

instances of the IP are implemented in order to have two different TCP/UDP/IP stacks.

- **IPCTEK's Ten-Gigabit traffic generator IP core**: this IP core is used to send/receive UDP and TCP traffic to/from the TCP/UDP/IP stack. The traffic type can be configured to be either a static greeting message or a PRBS sequence. The IP core also includes a PRBS sequence verification module which can be used to verify the data integrity of a TCP stream or UDP packets. An AXI4-Lite interface is used to configure the IP. There are two IP instances implemented in the design in order to interface with the two TCP/UDP/IP stacks. The source code of this IP core is delivered as is. The principal purpose of this IP core is to demonstrate to users how to interface with the Ten-Gigabit TCP/UDP/IP Stack.
- **A Microblaze sub-system** to configure the whole system via an AXI4-Lite interface. An UART interface is also implemented in order to interact with the user.
- The two SFP+ ports must be connected together (or to a switch) in case a loopback test is required.

Firmware generation steps

Library generation

Before generating the example project, different necessary IPs need to be compiled.

--- Tip ---
In Windows, open the cmd console then use subst command to create a virtual Disk pointing to the Vivado Hdl root folder in order to avoid Windows's maximum path length limitation error. For example, to create a virtual disk named T, tap the following in the cmd console.

```
subst T: <path_to_the_root_folder>
```

Open Vivado, cd into the /library folder, use the Vivado tcl console

```
cd T:/library
```

Run the script buildLib.tcl to compile the library, use the Vivado tcl console

```
source ./buildLib.tcl
```

When the library compilation is finished, we proceed to generate the example project.

Project generation

cd into the project folder /projects/tcp_udp_10g_over_kcu105, use the Vivado tcl console

```
cd T:/projects/tcp_udp_10g_over_kcu105
```

Run the script system_project.tcl to generate the project, use the Vivado tcl console

```
source ./system_project.tcl
```

After the script is finished loading the board design, run the synthesis then the implementation and generate the bitstream. These processes take about one hour to finish depending on the host PC.

Export the bitstream then launch the SDK. We proceed to create an SDK application project.

SDK application project

Create an application project based on the hardware that we have just exported from the Vivado project.

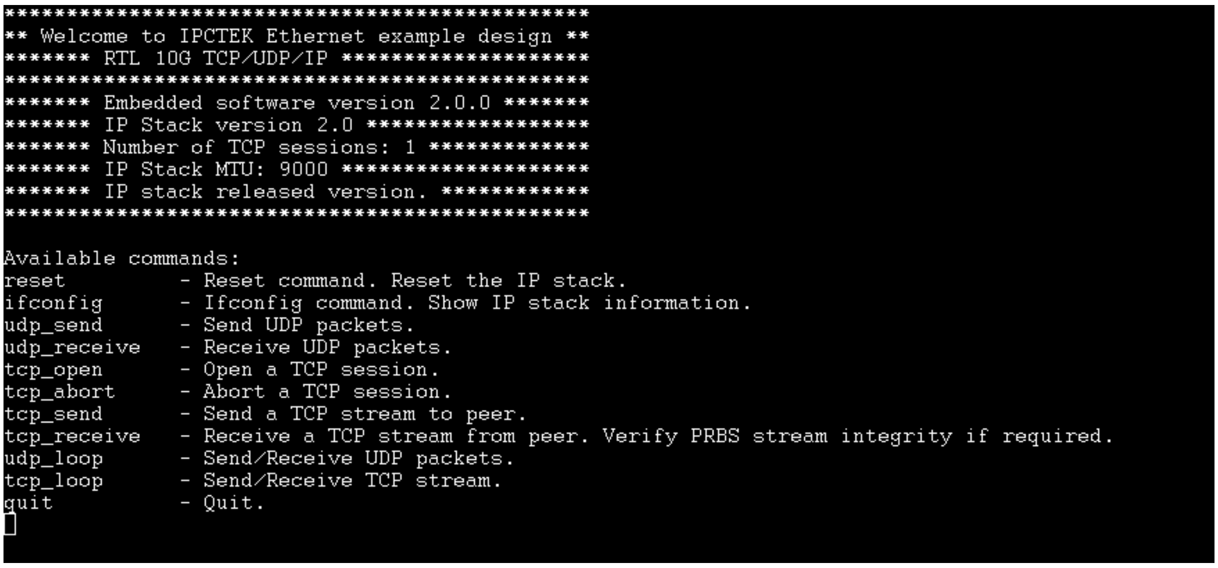
The SDK application source code for the example design is located at /drivers/ip_stack_10g/tcp_udp_10g_over_kcu105/src folder. Import this folder into your SDK project.

--- Note---
Reconfigure the Linker Script to increase the stack size and the heap size to 8 KB for a stable application.

Program the board, plug the USB/UART cable into the KCU105 board, open a console application (e.g. Tera Term) then run the application. The UART baud rate is equal to 115200 Hz. Remember to check the “local echo” option in Tera Term in order to see the user’s input command.

Firmware test menu

Users should find this UART console screen after launching the SDK application.



```
**********************************************
** Welcome to IPCTEK Ethernet example design **
******* RTL 10G TCP/UDP/IP *******************
**********************************************
******* Embedded software version 2.0.0 *******
******* IP Stack version 2.0 ******************
******* Number of TCP sessions: 1 *************
******* IP Stack MTU: 9000 ********************
******* IP stack released version. ***********
**********************************************

Available commands:
reset           - Reset command. Reset the IP stack.
ifconfig        - Ifconfig command. Show IP stack information.
udp_send        - Send UDP packets.
udp_receive     - Receive UDP packets.
tcp_open        - Open a TCP session.
tcp_abort       - Abort a TCP session.
tcp_send        - Send a TCP stream to peer.
tcp_receive     - Receive a TCP stream from peer. Verify PRBS stream integrity if required.
udp_loop        - Send/Receive UDP packets.
tcp_loop        - Send/Receive TCP stream.
quit            - Quit.
```

Figure 2 - Firmware UART console

Available commands for test are:

- **reset** command: use this command to reset the TCP/UCP/IP stacks and the Traffic Generator IP cores.
- **ifconfig** command: use this command to show statistical information concerning the TCP/UCP/IP stacks.
- **udp_send** command: use this command to send UDP packets from the KCU105 board to a destination. In this test the SFP+ port 0 is used.
- **udp_receive** command: use this command to tell the Traffic generator IP to filter and receive UDP packets. If the predefined PRBS sequence is about to be received, the PRBS verification function can be enabled to verify the data integrity. In this test the SFP+ port 0 is used.
- **udp_loop** command: use this command to send UDP packets from one SFP+ port to the other port on the same KCU105 board. A PRBS sequence is sent from one TCP/UDP/IP stack, passing though the SFP+ transceivers and is received by the other TCP/UDP/IP stack. A PRBS verification module is also activated in order to verify the data integrity of the UDP packets. In this test both SFP+ port 0 and port 1 are used.
- **tcp_open** command: use this command to open a TCP session. The server/client mode can be chosen during the configuration. Be sure to have a TCP peer running somewhere to interact with the FPGA's TCP server/client (The Qt-based IP test bench delivered along with the example design can be used for this purpose). In this test the SFP+ port 0 is used.
- **tcp_abort** command: use this command to abort a TCP session. In this test the SFP+ port 0 is used.
- **tcp_send** command: after the connection was established, use this command to send a data stream to the TCP peer. In this test the SFP+ port 0 is used.
- **tcp_receive** command: after the connection was established, use this command to prepare for receiving a data stream coming from the peer. If the predefined PRBS sequence is to be received, the user can activate the PRBS verification module in order to verify the data integrity. Theoretically, as a TCP connection is guaranteed to be error-free, the data integrity check should always pass. If the Qt-based TCP server/client given by the example design is used, this data integrity check function is already implemented. This allows to test the speed performance and to validate the correct functioning of the IP core. In this test the SFP+ port 0 is used.
- **tcp_loop** command: use this command to test the TCP loop configuration. One IP stack is configured as a TCP server, the other is configured as a TCP client. In this test the connection between the TCP server and the TCP client will be established and a PRBS sequence will be sent from the client to the server. At the server side, the PRBS verification component is also activated to verify the data integrity of the TCP data

stream. Do not forget to physically connect the two SFP+ ports in your evaluation board during this test. In this test both SFP+ port 0 and port 1 are used.

- **quit** command: use this command to quit the application.

--- Tip ---
When udp_loop or tcp_loop commands are used, loopback the SFP+ transceivers in your evaluation board, or connect them to an Ethernet switch.

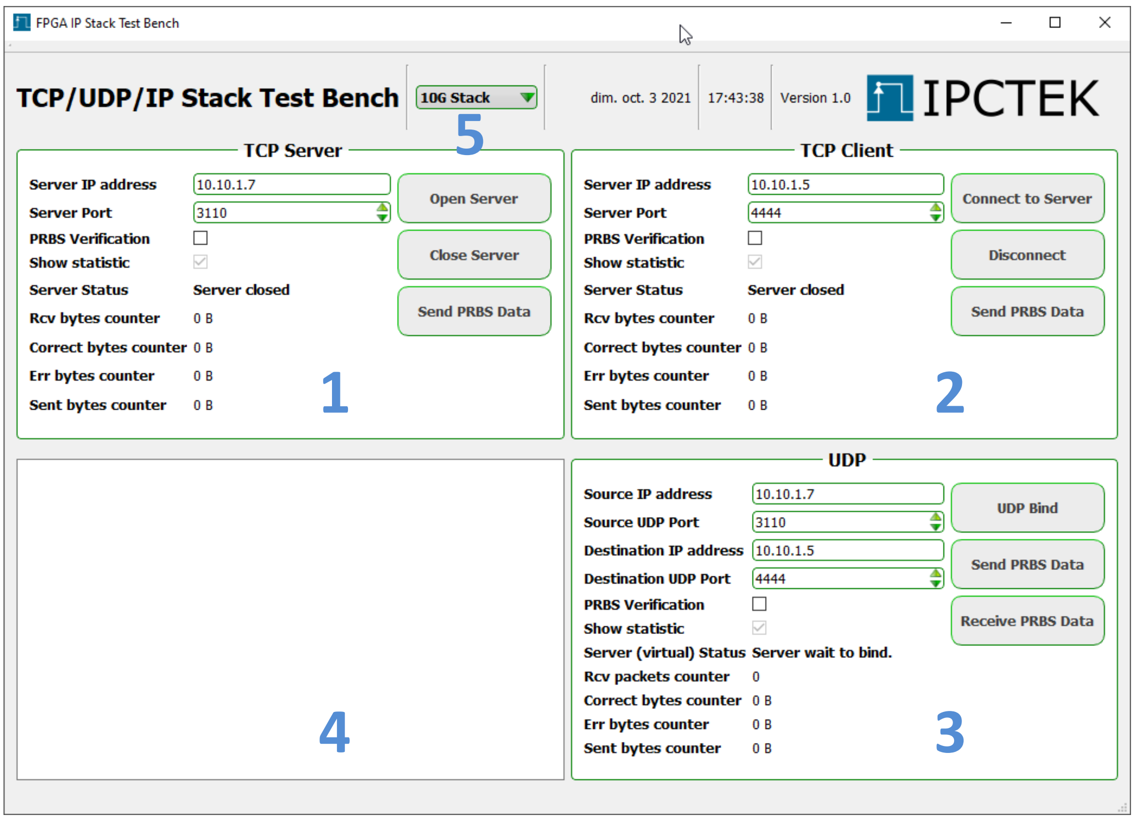
Qt test bench interface

The Qt test bench is written with Qt Creator version 4.13.2 and Qt version 5.12.2. In the test bench we implement a TCP Server, a TCP client and an UDP transmit/receive engine in order to interact with the IP stack on the FPGA.

In the test bench we use Windows socket for TCP server/client and UDP transmitter. However, we use npcap SDK to implement the UDP Receiver engine because of the poor performance of the native socket.

--- *Important Note*--
The npcapsdk version 1.05 is included in the test bench source code. In order to use the library, user must install the npcap on the host PC. By the time this document is written, the nmap version 7.92 has been installed. The nmap installer also handles the npcap installation.

The test bench interface is shown in the figure below.



FPGA IP Stack Test Bench

TCP/UDP/IP Stack Test Bench | 10G Stack | dim. oct. 3 2021 | 17:43:38 | Version 1.0

5

TCP Server

Server IP address	10.10.1.7
Server Port	3110
PRBS Verification	☐
Show statistic	✓
Server Status	Server closed
Rcv bytes counter	0 B
Correct bytes counter	0 B
Err bytes counter	0 B
Sent bytes counter	0 B

Open Server | Close Server | Send PRBS Data

1

TCP Client

Server IP address	10.10.1.5
Server Port	4444
PRBS Verification	☐
Show statistic	✓
Server Status	Server closed
Rcv bytes counter	0 B
Correct bytes counter	0 B
Err bytes counter	0 B
Sent bytes counter	0 B

Connect to Server | Disconnect | Send PRBS Data

2

UDP

Source IP address	10.10.1.7
Source UDP Port	3110
Destination IP address	10.10.1.5
Destination UDP Port	4444
PRBS Verification	☐
Show statistic	✓
Server (virtual) Status	Server wait to bind.
Rcv packets counter	0
Correct bytes counter	0 B
Err bytes counter	0 B
Sent bytes counter	0 B

UDP Bind | Send PRBS Data | Receive PRBS Data

3

4

Figure 3 - Qt test bench interface

1. TCP server interface, used to test the FPGA TCP client mode.
2. TCP client interface, used to test the FPGA TCP server mode.
3. UDP transceiver interface, used to test the FPGA UDP Tx and Rx engines.

4. Test bench log, used to display useful messages during the test.
5. Select the speed. Select **10G Stack** for this example design. This is used to correctly configure the PRBS engine that matches with the FPGA Traffic Generator IP.

Test examples

In this test examples, the SFP+ **port 0** of the KCU105 board is directly connected to an Intel(R) 82599 10 Gigabit Network Connection PCIe card. The Qt test bench is running on a Windows 10 host PC.

Configure the host PC Ethernet interface to static mode with the following information

- IP address: 10.10.1.7
- Netmask: 255.255.255.0
- Gateway: 10.10.1.1

TCP client test

In this test we use the Qt TCP server interface and the FPGA TCP/IP stack is configured to client mode.

Open the server by clicking on the ***Open Server*** button.

Wait until "**Server is listening**." text is displayed on the Server Status.

--- Note---
The Windows firewall may ask for permission for the program to have access to the Ethernet interface. In this case click Yes to give the permission.

The TCP server is now listening for a connection request. We proceed to prepare for a TCP client on the FPGA.

Use the **tcp_open** command to create a TCP session.

When asked for the TCP server/client mode, input 0 to the console to select the client mode.

Input the server IP address which is 10.10.1.7 by default.

Input the server port which is 3110 by default.

Input a number for the client port, e.g. 4444.

Upon success, "**Connected**." text should be displayed on the Server status label.

Client to Server transfer test

In this example we will transfer 8.96 GB of PRBS data to the server. This corresponds to 1 million of segments of size 8960 bytes. While receiving the data stream, the server also verifies the data integrity of the stream.

Check the ***PRBS Verification*** check box to enable the verification option.

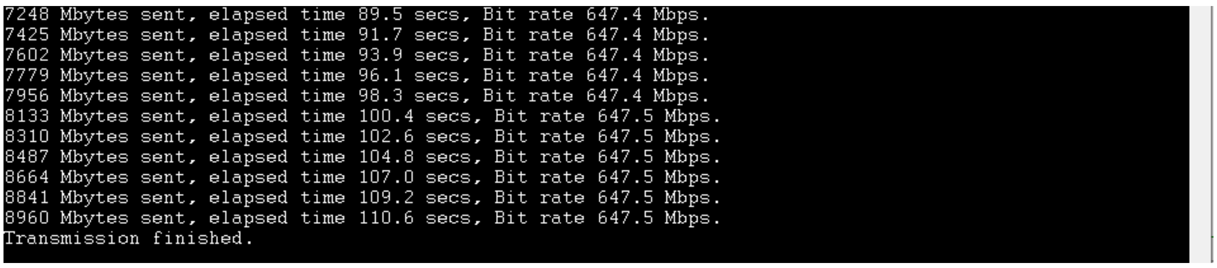
In the FPGA UART console, use the **tcp_send** command to send data to the server.

When asked for the data mode, input 1 to select the PRBS data.

When asked for the packet length (or also the segment length), input 8960. It is noted that this is the maximum segment length value, given that the MAC MTU is equal to 9000 bytes.

When asked for the number of packets (also segments), input 1000000.

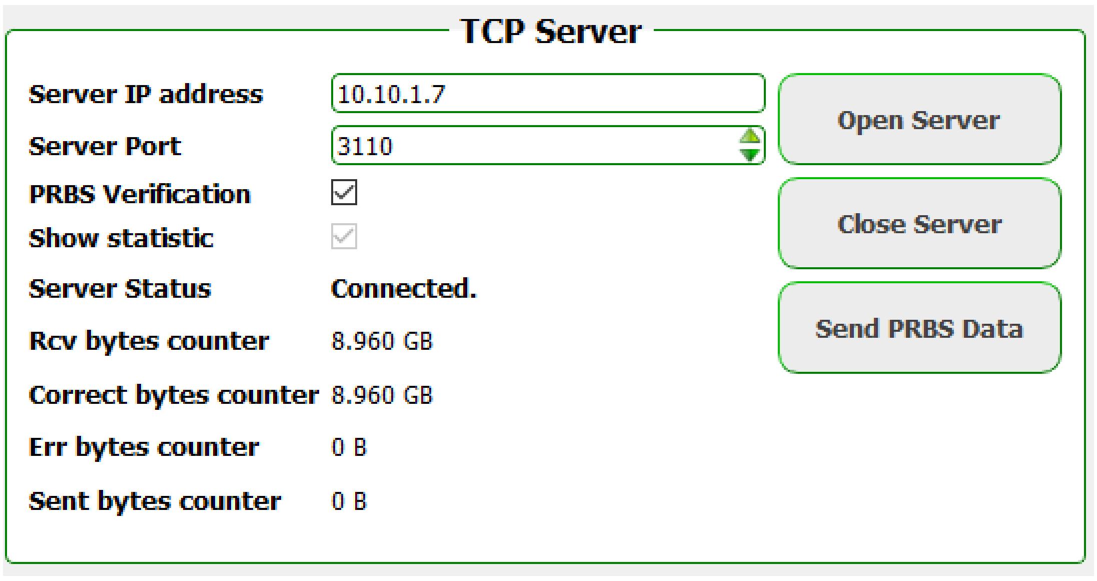
The screenshots of the UART console and the test bench interface when the transmission is finished are shown in figures below.



```
7248 Mbytes sent, elapsed time 89.5 secs, Bit rate 647.4 Mbps.
7425 Mbytes sent, elapsed time 91.7 secs, Bit rate 647.4 Mbps.
7602 Mbytes sent, elapsed time 93.9 secs, Bit rate 647.4 Mbps.
7779 Mbytes sent, elapsed time 96.1 secs, Bit rate 647.4 Mbps.
7956 Mbytes sent, elapsed time 98.3 secs, Bit rate 647.4 Mbps.
8133 Mbytes sent, elapsed time 100.4 secs, Bit rate 647.5 Mbps.
8310 Mbytes sent, elapsed time 102.6 secs, Bit rate 647.5 Mbps.
8487 Mbytes sent, elapsed time 104.8 secs, Bit rate 647.5 Mbps.
8664 Mbytes sent, elapsed time 107.0 secs, Bit rate 647.5 Mbps.
8841 Mbytes sent, elapsed time 109.2 secs, Bit rate 647.5 Mbps.
8960 Mbytes sent, elapsed time 110.6 secs, Bit rate 647.5 Mbps.
Transmission finished.
```

Figure 4 - FPGA TCP Client. Client to server transfer, UART screen

--- Important *Note*--
Make sure to configure your network card to support jumbo frame (or extended mode with the packet length larger than 9000 bytes).


TCP Server

Server IP address	10.10.1.7
Server Port	3110
PRBS Verification	✓
Show statistic	✓
Server Status	Connected.
Rcv bytes counter	8.960 GB
Correct bytes counter	8.960 GB
Err bytes counter	0 B
Sent bytes counter	0 B

Figure 5 - FPGA TCP Client. Client to server transfer, test bench screen

--- Note---
*After each transmission using the **tcp_send** command, the PRBS generator in the FPGA is resetted. Before restarting another transmission, the PRBS generator in the test bench should also be resetted to the initial value. Toggling the **PRBS Verification** checkbox to reset the PRBS engine.*
--

Server to client transfer test

In this example the Qt test bench interface server will transfer 8.96 GB of PRBS data to the FPGA client. This corresponds to 1 million of segments of size 8960 bytes. While receiving the data stream, the client also verifies the data integrity of the stream.

In the FPGA UART console, use the **tcp_receive** command to prepare for receiving the data from the server.

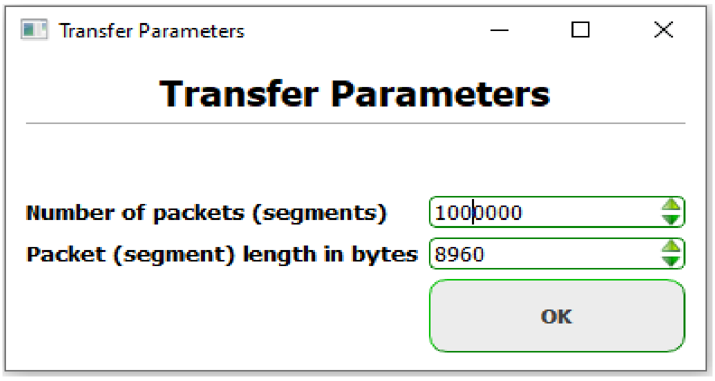
When asked for the verification option, input 1 to enable the PRBS verification function.

When asked for the data mode, input 1 to select the PRBS data.

When asked for the number of segments expected to receive, input 1000000.

When asked for the segment length (in number of bytes), input 8960.

In the Qt test bench interface, click on the button ***Send PRBS Data*** to begin sending data to the client. Configure the corresponding numbers of segments and the segment length as shown in the figure below.



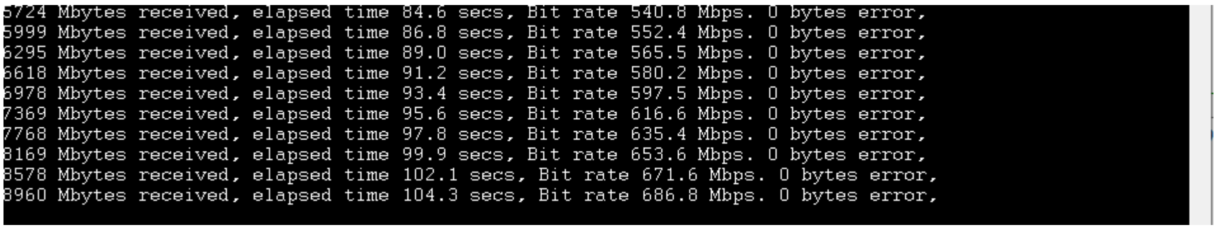
Transfer Parameters

Transfer Parameters

Number of packets (segments)	1000000
Packet (segment) length in bytes	8960

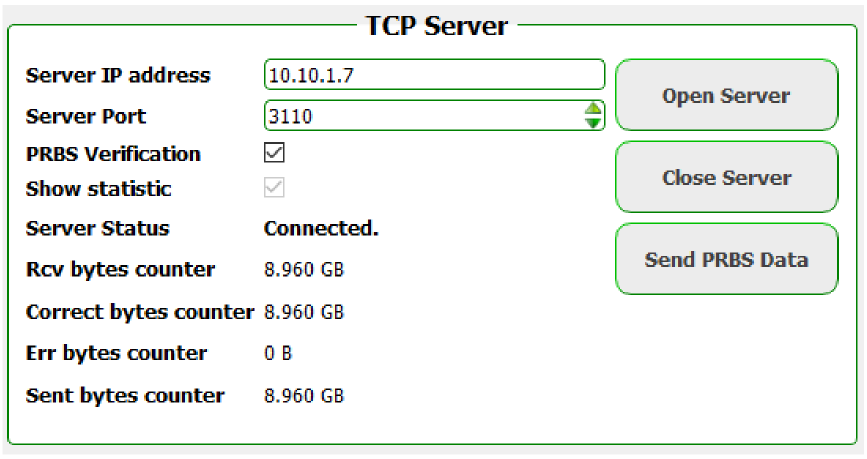
Figure 6 - FPGA TCP Client. Server to client transfer parameters

The screenshots of the UART console and the Qt test bench interface when the transmission is finished are shown in figures below.



```
5724 Mbytes received, elapsed time 84.6 secs, Bit rate 540.8 Mbps. 0 bytes error,
5999 Mbytes received, elapsed time 86.8 secs, Bit rate 552.4 Mbps. 0 bytes error,
6295 Mbytes received, elapsed time 89.0 secs, Bit rate 565.5 Mbps. 0 bytes error,
6618 Mbytes received, elapsed time 91.2 secs, Bit rate 580.2 Mbps. 0 bytes error,
6978 Mbytes received, elapsed time 93.4 secs, Bit rate 597.5 Mbps. 0 bytes error,
7369 Mbytes received, elapsed time 95.6 secs, Bit rate 616.6 Mbps. 0 bytes error,
7768 Mbytes received, elapsed time 97.8 secs, Bit rate 635.4 Mbps. 0 bytes error,
8169 Mbytes received, elapsed time 99.9 secs, Bit rate 653.6 Mbps. 0 bytes error,
8578 Mbytes received, elapsed time 102.1 secs, Bit rate 671.6 Mbps. 0 bytes error,
8960 Mbytes received, elapsed time 104.3 secs, Bit rate 686.8 Mbps. 0 bytes error,
```

Figure 7 - FPGA TCP Client. Server to client transfer, UART screen



TCP Server

Server IP address	10.10.1.7
Server Port	3110
PRBS Verification	☑
Show statistic	☑
Server Status	Connected.
Rcv bytes counter	8.960 GB
Correct bytes counter	8.960 GB
Err bytes counter	0 B
Sent bytes counter	8.960 GB

Figure 8 - FPGA TCP Client. Server to client transfer, test bench screen

TCP server test

Use either the **tcp_abort** command or the ***Close Server*** button to close the actual TCP session.

Use the **tcp_open** command to open the FPGA server.

When asked for server/client mode, input 1 to configure the stack to the server mode.

When asked for the server port, input 4444 for example.

The FPGA TCP server is listening to a connection request. Use the Qt test bench TCP Client interface to connect to the server.

Click on the ***Connect to Server*** button.

Wait until "***Connected.***" text is displayed on the Server Status label.

Server to client transfer test

In this example test we will transfer 1000000 segments of size 8960 bytes from the FPGA server to the Qt test bench client.

Check the ***PRBS Verification*** checkbox in order to enable the data integrity check.

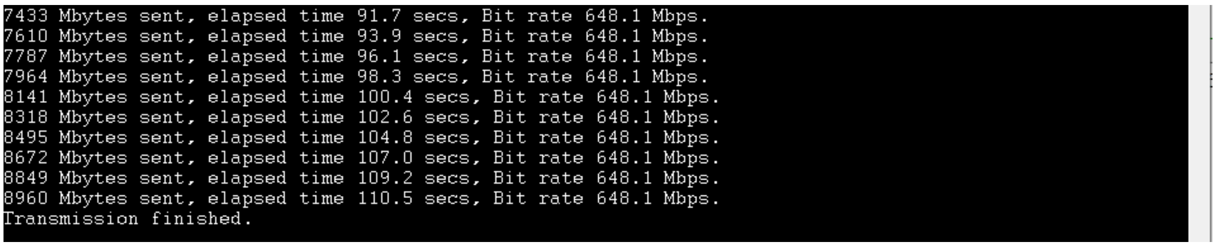
Use the **tcp_send** command to send a data stream to the Qt test bench.

When asked for the data mode, input 1 to configure the PRBS data.

When asked for the packet length (or the segment length), input 8960.

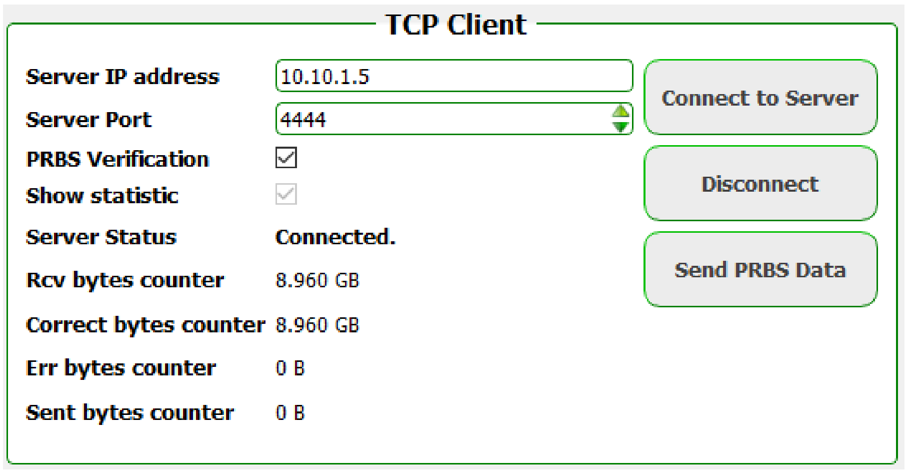
When asked for the number of packets (or segments) to be sent, input 1000000.

The screenshots of the UART console and the test bench interface when the transmission is finished are shown in the figures below.



```
7433 Mbytes sent, elapsed time 91.7 secs, Bit rate 648.1 Mbps.
7610 Mbytes sent, elapsed time 93.9 secs, Bit rate 648.1 Mbps.
7787 Mbytes sent, elapsed time 96.1 secs, Bit rate 648.1 Mbps.
7964 Mbytes sent, elapsed time 98.3 secs, Bit rate 648.1 Mbps.
8141 Mbytes sent, elapsed time 100.4 secs, Bit rate 648.1 Mbps.
8318 Mbytes sent, elapsed time 102.6 secs, Bit rate 648.1 Mbps.
8495 Mbytes sent, elapsed time 104.8 secs, Bit rate 648.1 Mbps.
8672 Mbytes sent, elapsed time 107.0 secs, Bit rate 648.1 Mbps.
8849 Mbytes sent, elapsed time 109.2 secs, Bit rate 648.1 Mbps.
8960 Mbytes sent, elapsed time 110.5 secs, Bit rate 648.1 Mbps.
Transmission finished.
```

Figure 9 - FPGA TCP Server. Server to client transfer, UART screen



TCP Client

Server IP address	10.10.1.5
Server Port	4444
PRBS Verification	☑
Show statistic	☑
Server Status	Connected.
Rcv bytes counter	8.960 GB
Correct bytes counter	8.960 GB
Err bytes counter	0 B
Sent bytes counter	0 B

Figure 10 - FPGA TCP Server. Server to client transfer, test bench screen

--- Remark--
The transmission performance is highly dependent on the PC host's PCIe network card performance. We will see a near line-rate speed performance with loopback tests.
--

Client to server transfer test

In this example test we will use the Qt test bench interface to send 8.96 GB of PRBS data to the FPGA TCP server. The PRBS Verification engine in the FPGA is also enabled to verify the data integrity.

Use the **tcp_receive** command to prepare for a reception of 8.96 GB.

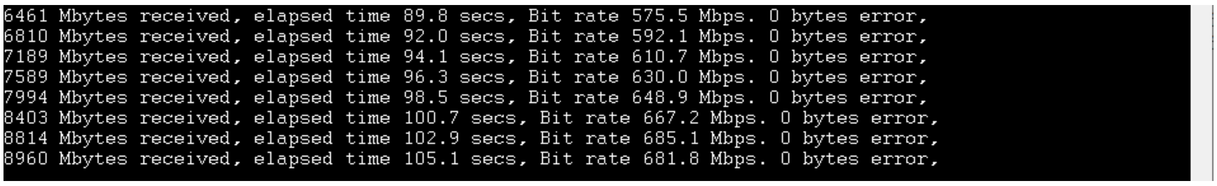
When asked for the PRBS verification option, input 1 to enable this functionality.

When asked for the expected number of packets (or segments), input 1000000.

When asked for the packet length (or segment length), input 8960.

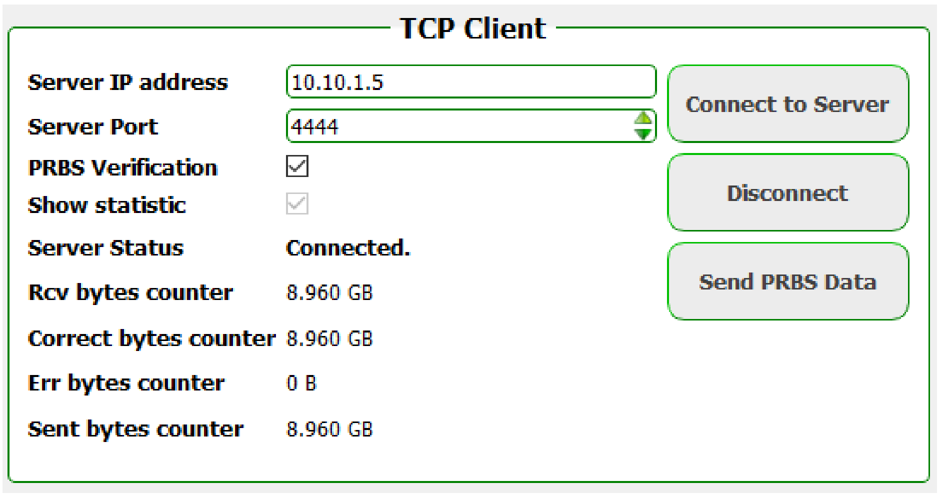
Click on the button ***Send PRBS Data*** (on the TCP Client group) to send data to the FPGA server.

The screenshots of the UART console and the test bench interface when the transmission is finished are shown in the figures below.



```
6461 Mbytes received, elapsed time 89.8 secs, Bit rate 575.5 Mbps. 0 bytes error,
6810 Mbytes received, elapsed time 92.0 secs, Bit rate 592.1 Mbps. 0 bytes error,
7189 Mbytes received, elapsed time 94.1 secs, Bit rate 610.7 Mbps. 0 bytes error,
7589 Mbytes received, elapsed time 96.3 secs, Bit rate 630.0 Mbps. 0 bytes error,
7994 Mbytes received, elapsed time 98.5 secs, Bit rate 648.9 Mbps. 0 bytes error,
8403 Mbytes received, elapsed time 100.7 secs, Bit rate 667.2 Mbps. 0 bytes error,
8814 Mbytes received, elapsed time 102.9 secs, Bit rate 685.1 Mbps. 0 bytes error,
8960 Mbytes received, elapsed time 105.1 secs, Bit rate 681.8 Mbps. 0 bytes error,
```

Figure 11 - FPGA TCP Server. Client to server transfer, UART screen



TCP Client

Server IP address: 10.10.1.5
Server Port: 4444
PRBS Verification: ☑
Show statistic: ☑
Server Status: Connected.
Rcv bytes counter: 8.960 GB
Correct bytes counter: 8.960 GB
Err bytes counter: 0 B
Sent bytes counter: 8.960 GB

Figure 12 - FPGA TCP Server. Client to server transfer, test bench screen

UDP transmitter test

In this example test we use the FPGA UDP Tx engine to send 100000 packets of size 8960 bytes to the Qt test bench UDP receiver. The PRBS verification option is also enabled to verify the integrity of the received data.

Click on the button ***UDP Bind*** to bind the UDP socket to the corresponding address and ports. The "***Binding success***." text should appear on the **Server (virtual) status** label.

Enable the PRBS verification option by checking the ***PRBS Verification*** checkbox.

Click on the ***Receive PRBS data*** button to prepare for receiving data. Refer to the system log for useful information. At this stage, the npcap is called to sniff for 100,000 UDP packets whose destination port is equal to 3110.

Use the **udp_send** command to send data to the Qt test bench UDP receiver.

When asked for the destination IP address, input 10.10.1.7.

When asked for the UDP source port, input for example 4444.

When asked for the UDP destination port, the user must input 3110 to match that of the npcap packet filter.

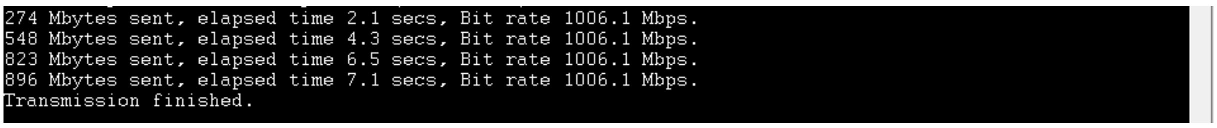
When asked for the data mode, input 1 for PRBS data.

When asked for the packet length, input 8960.

When asked for the number of packets, input 100000.

When asked for the Interframe Gap value, input 10000 for example.

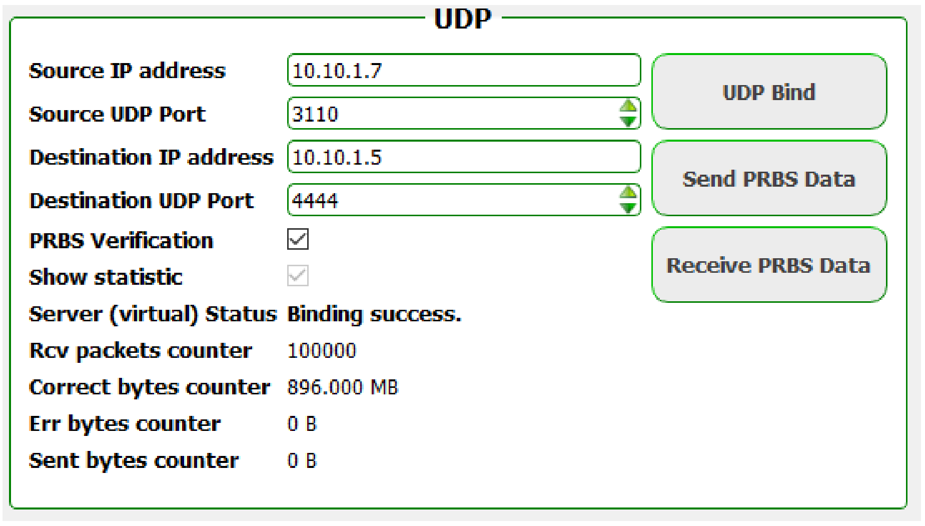
The screenshots of the UART console and the Qt test bench interface when the transmission is finished are shown in figures below.



```
274 Mbytes sent, elapsed time 2.1 secs, Bit rate 1006.1 Mbps.
548 Mbytes sent, elapsed time 4.3 secs, Bit rate 1006.1 Mbps.
823 Mbytes sent, elapsed time 6.5 secs, Bit rate 1006.1 Mbps.
896 Mbytes sent, elapsed time 7.1 secs, Bit rate 1006.1 Mbps.
Transmission finished.
```

Figure 13 - UDP Tx test, UART screen

IPCTEK



Source IP address 10.10.1.7
Source UDP Port 3110
Destination IP address 10.10.1.5
Destination UDP Port 4444
PRBS Verification ☑
Show statistic ☑
Server (virtual) Status Binding success.
Rcv packets counter 100000
Correct bytes counter 896.000 MB
Err bytes counter 0 B
Sent bytes counter 0 B

Figure 14 - UDP Tx test, test bench screen

--- Remark--
The interframe gap value is the pause time in between two consecutive UDP packets. This allows modulating the UDP transmission throughput. The minimum value is equal to 1 corresponding to the maximum throughput of the FPGA UDP transmitter. Users must regulate this value in order to be able to capture all the packets at the host PC. If a small value of interframe gap is configured it is possible that UDP packets will be dropped and a large number of errors should be detected by the PRBS verification module.
*When the Qt test bench drops UDP packets, use the **udp_send** command to send another burst of UDP packets so that the Qt UDP receiver is finished receiving 100,000 packets. Then toggle the PRBS Verification checkbox to reset the PRBS engine and restart all over again with a larger interframe gap value.*
--

UDP receiver test

In this example test we use the test bench interface to send 1000000 UDP packets of size 8960 bytes to the FPGA UDP receiver. The PRBS verification module in the FPGA is also enabled in order to verify the data integrity.

Use the **udp_receive** command to prepare for receiving UDP packets.

When asked for the source IP address, input 10.10.1.7, which is the IP address of the Qt test bench UDP virtual server.

When asked for the destination IP address, input 10.10.1.5, which is the address of the FPGA TCP/UDP/IP stack.

When asked for the UDP source port, input 3110.

When asked for the UDP destination port, input 4444.

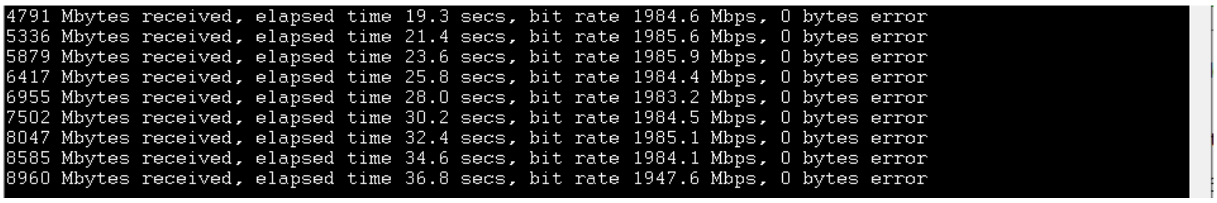
When asked for the PRBS verification option, input 1 to enable the module.

When asked for the packet length, input 8960.

When asked for the number of packets, input 1000000.

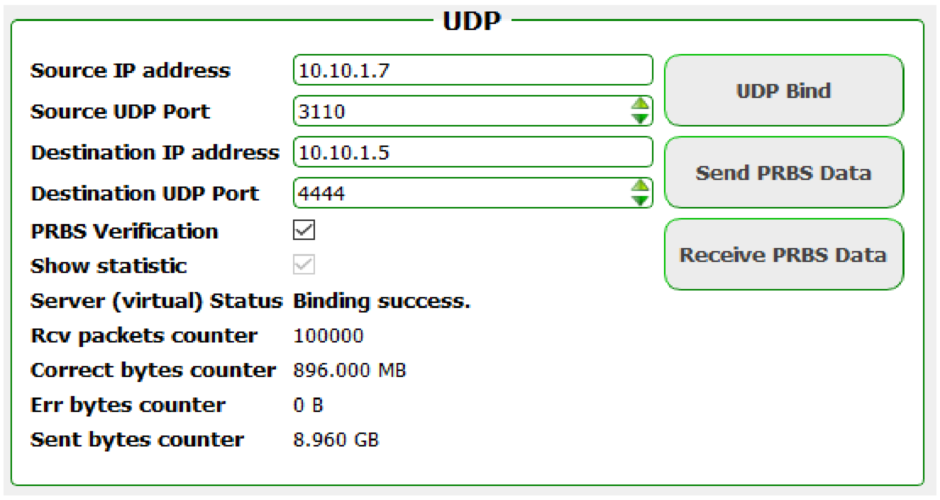
On the Qt test bench interface, click on the ***Send PRBS Data*** button to begin sending. Configure the corresponding number of packets and the packet length then click on the ***OK*** button.

The screenshots of the UART console and the Qt test bench interface when the transmission is finished are shown in the figures below.



```
4791 Mbytes received, elapsed time 19.3 secs, bit rate 1984.6 Mbps, 0 bytes error
5336 Mbytes received, elapsed time 21.4 secs, bit rate 1985.6 Mbps, 0 bytes error
5879 Mbytes received, elapsed time 23.6 secs, bit rate 1985.9 Mbps, 0 bytes error
6417 Mbytes received, elapsed time 25.8 secs, bit rate 1984.4 Mbps, 0 bytes error
6955 Mbytes received, elapsed time 28.0 secs, bit rate 1983.2 Mbps, 0 bytes error
7502 Mbytes received, elapsed time 30.2 secs, bit rate 1984.5 Mbps, 0 bytes error
8047 Mbytes received, elapsed time 32.4 secs, bit rate 1985.1 Mbps, 0 bytes error
8585 Mbytes received, elapsed time 34.6 secs, bit rate 1984.1 Mbps, 0 bytes error
8960 Mbytes received, elapsed time 36.8 secs, bit rate 1947.6 Mbps, 0 bytes error
```

Figure 15 - UDP Rx test, UART screen



UDP

Source IP address	10.10.1.7
Source UDP Port	3110
Destination IP address	10.10.1.5
Destination UDP Port	4444
PRBS Verification	✓
Show statistic	✓
Server (virtual) Status	Binding success.
Rcv packets counter	100000
Correct bytes counter	896.000 MB
Err bytes counter	0 B
Sent bytes counter	8.960 GB

Figure 16 - UDP Rx test, test bench screen

--- Remark---
While sending UDP packets, it is possible that one or several UDP packets get lost. If that is the case, when the transmission is finished, the FPGA UDP Rx continues waiting for the last packets. In this case, send another burst of packets to the FPGA so that it quits the waiting loop and restart all over again.

TCP loop test

In this test the two SFP+ modules on the evaluation board should be connected together. In this test one Ten-Gigabit TCP/UDP/IP instance serves as a TCP server, the other instance serves as a TCP client.

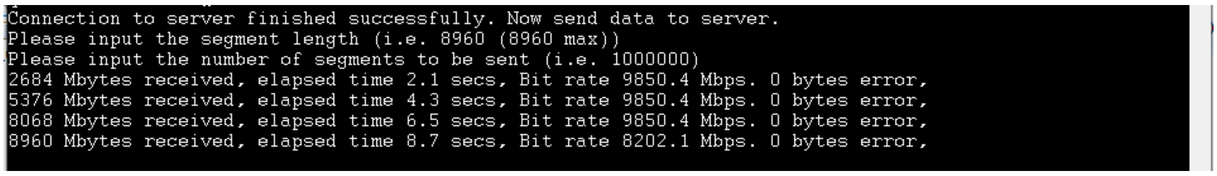
After the connection is established, we send 1000000 segments of size 8960 bytes from the client to the server.

Use the **tcp_loop** command to begin the test.

When asked for the segment length, input 8960.

When asked for the number of segments to be sent, input 1000000.

A screenshot of the UART console when the transmission is finished is shown in figure below.



```
Connection to server finished successfully. Now send data to server.
Please input the segment length (i.e. 8960 (8960 max))
Please input the number of segments to be sent (i.e. 1000000)
2684 Mbytes received, elapsed time 2.1 secs, Bit rate 9850.4 Mbps. 0 bytes error,
5376 Mbytes received, elapsed time 4.3 secs, Bit rate 9850.4 Mbps. 0 bytes error,
8068 Mbytes received, elapsed time 6.5 secs, Bit rate 9850.4 Mbps. 0 bytes error,
8960 Mbytes received, elapsed time 8.7 secs, Bit rate 8202.1 Mbps. 0 bytes error,
```

Figure 17 - FPGA TCP Loop. Client to server transfer, UART screen

--- Note---
The design does not support the hot-plug functionality. After connecting the two SFP+ ports together, the firmware needs to be restarted.
--

UDP loop test

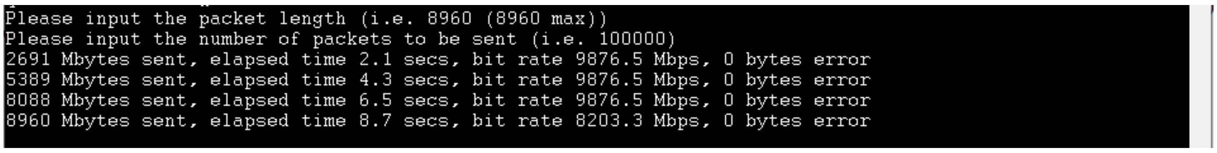
In this test we use one Ten-Gigabit TCP/UDP/IP instance to send 1000000 packets of size 8960 bytes to the other instance.

Use the **udp_loop** command to begin the test.

When asked for the packet length, input 8960.

When asked for the number of packets to be sent, input 1000000.

The screenshot of the UART console is shown in the following figure.



```
Please input the packet length (i.e. 8960 (8960 max))
Please input the number of packets to be sent (i.e. 100000)
2691 Mbytes sent, elapsed time 2.1 secs, bit rate 9876.5 Mbps, 0 bytes error
5389 Mbytes sent, elapsed time 4.3 secs, bit rate 9876.5 Mbps, 0 bytes error
8088 Mbytes sent, elapsed time 6.5 secs, bit rate 9876.5 Mbps, 0 bytes error
8960 Mbytes sent, elapsed time 8.7 secs, bit rate 8203.3 Mbps, 0 bytes error
```

Figure 18 - UDP Loop test, UART screen

End Of Document.