IPCTEK

# TCP-UDP-IP STACK 1G
# RTL EXAMPLE DESIGN

*Example design user guide*

**UG003**

**Version 1.0**

**December 04, 2021**

IPCTEK

# Table of Contents

# Introduction

The purpose of this example design is to demonstrate the performance of the TCP/UDP/IP stack 1G FPGA IP core. In the example design, a fully functional One Gigabit TCP/UDP/IP stack is implemented in the Xilinx's KCU105 evaluation board. Principle characteristics of the FPGA IP core are given in the table below.

| Parameter | Value | Unit |
|---|---|---|
| MTU | 1500 | 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 | 16 | KBytes |
| TCP out-of-order Handling Enable | "TRUE" | |
| Out-of-order reordering buffer size | 16 | 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_1g/**) 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

This section details 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 Gigabit TCP/UDP/IP stack which will be running on the Xilinx's KCU105 evaluation board. We use the Xilinx's Tri Mode Ethernet Mac IP to interface with the Marvell's PHY on the board via an SGMII link.

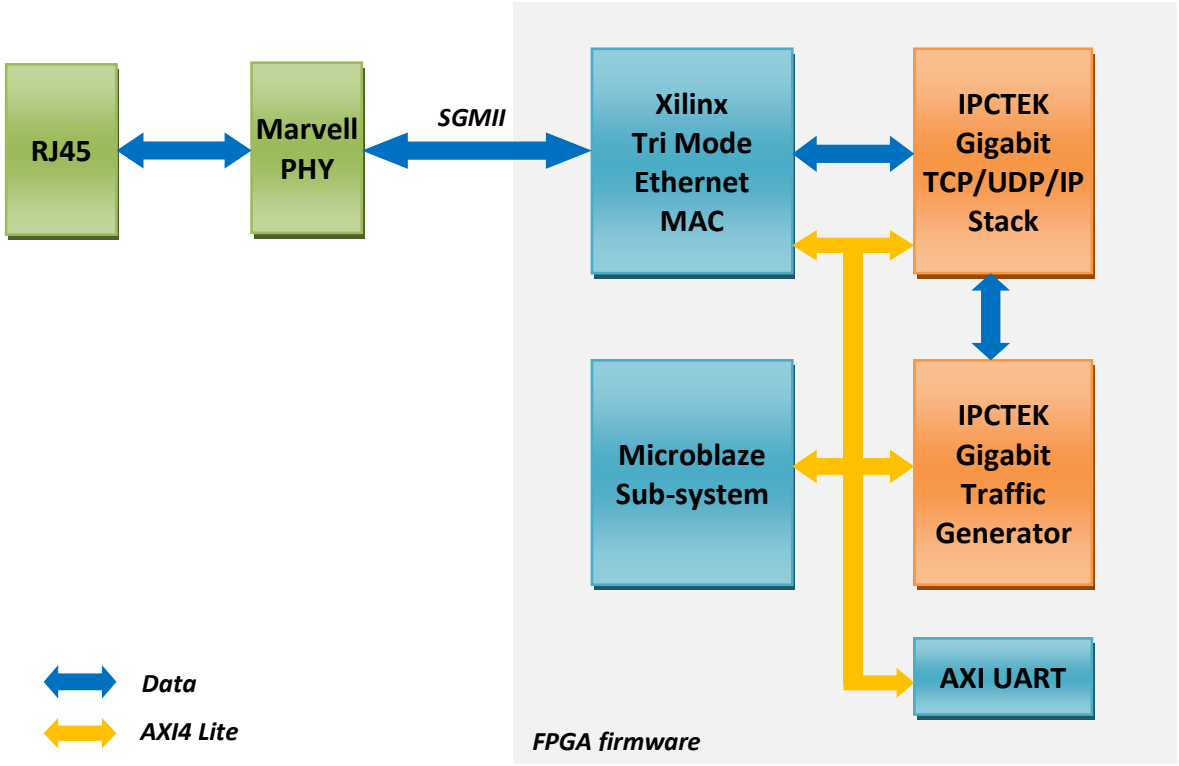


*Figure 1 - FPGA firmware synoptic*

The FPGA firmware has the following principle components:

- **Ethernet MAC:** The Xilinx's Tri Mode Ethernet MAC is used.
- **IPCTEK's Gigabit TCP/UDP/IP stack**: one TCP engine (can be switched between a server and a client), 1 UDP transmitter and 1 UDP receiver are instantiated in the IP core. The MAC MTU is equal to 1500 bytes. The IP core is embedded in an AXI4 Lite wrapper for easy integration within a Microblaze sub-system.
- **IPCTEK's Gigabit traffic generator IP core**: this FPGA 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 implemented for an easy integration. The source code of this IP core is delivered as is. The user is free to use and modify the code. The main purpose of this IP core is to demonstrate to users how to interface with the Gigabit TCP/UDP/IP Stack.

- **A Microblaze sub-system** to control the whole system via an AXI4 Lite interface. An UART interface is also implemented in order to receive user's commands.

# Firmware generation steps

## Library generation

Before generating the example project, we need to compile necessary libraries.

*--- Tip ---*

*In Windows, open the cmd console then use subst command to create a virtual Disk pointing to the example design 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_over_kcu105

```
cd T:/projects/tcp_udp_over_kcu105
```

Run the script system_project.tcl to create the project

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

After the script is finished loading the board design, run synthesis then 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_1g/tcp_udp_over_kcu105/src folder. Import this folder into your SDK project.

*--- Tip ----------*

*Reconfigure the Linker Script to increase the stack size and the heap size to 4 KB for a stable application.*

*----------*

Program the board, plug the USB/UART cable, open a console application (e.g. Tera Term) then run the application. The UART baud rate is 115200 Hz. Remember to check the "local echo" option in Tera Term in order to see 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 TCP/UDP/IP 1G stack  **************
*********************************************
******* Embedded software version 2.0.0 *******
******* IP Stack version 2.0 ******************
******* Number of TCP sessions: 1 *************
******* IP Stack MTU: 1500 ********************
******* 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.
quit          - Quit.
```

***Figure 2 - Firmware UART console***

Available commands for test are:

- **reset** command: use this command to reset the TCP/UCP/IP stack and the Traffic Generator IP cores.
- **ifconfig** command: use this command to show statistical information concerning the TCP/UCP/IP stack.
- **udp_send** command: use this command to send UDP packets from the KCU105 board to a destination.

- **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.
- **tcp_open** command: use this command to open a TCP session. The server/client mode can be chosen during configuration. Be sure to have a TCP peer instantiated 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).
- **tcp_abort** command: use this command to abort a TCP session.
- **tcp_send** command: after the connection was established, use this command to send a data stream to the TCP peer.
- **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, users can activate the PRBS verification option 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 included. This allows to test the speed performance and to validate the correct functioning of the IP core.
- **quit** command: use this command to quit the application.

# 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 a poor performance of the native socket.

*--- Important Note----------*
*The npcap sdk 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.

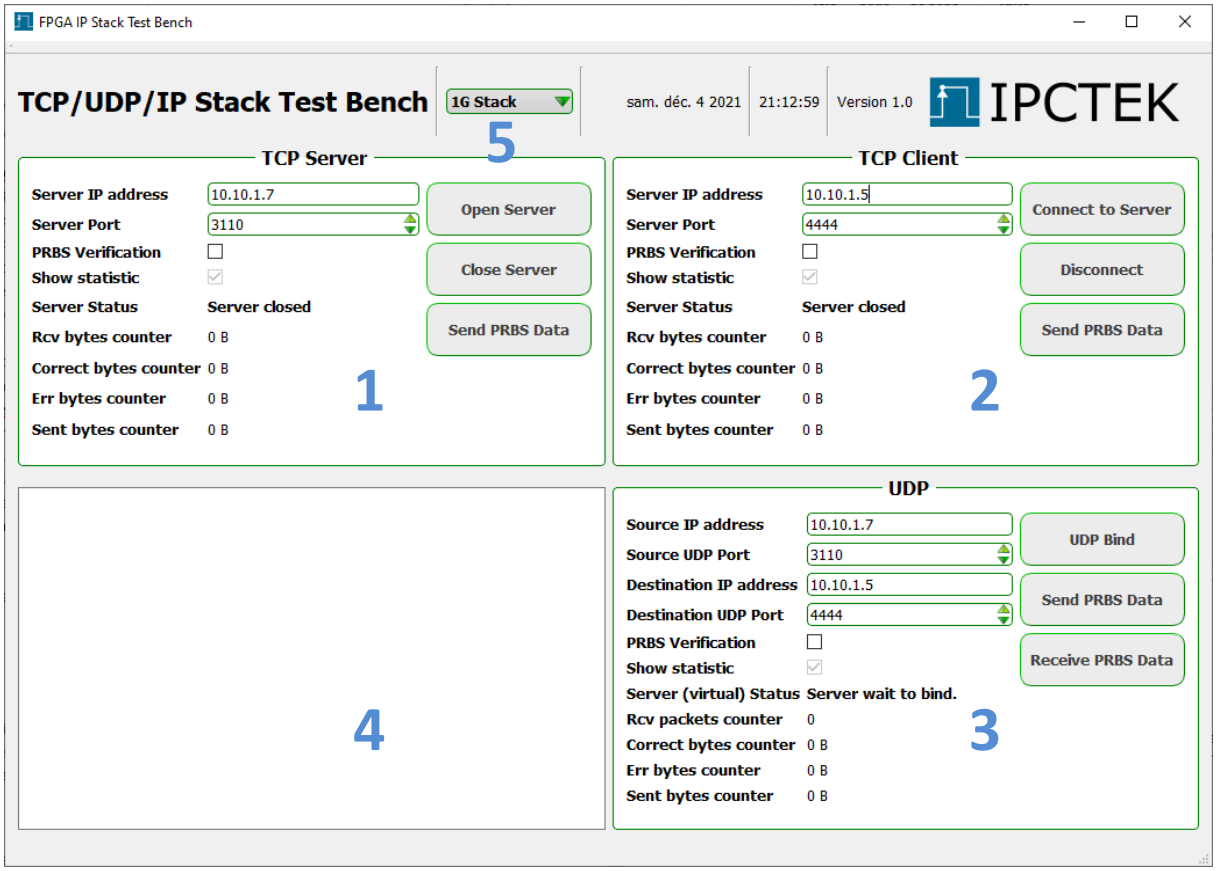


***Figure 3 - Qt test bench interface***

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, used to test the FPGA UDP Tx and Rx engines.
4. Test bench log, used to display useful message during the test.
5. Select the speed. Be sure to select **1G Stack** for this example design. This is used to correctly configure the PRBS engine that matches with the FPGA Traffic Generator IP.

# Test examples

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 the TCP client on the FPGA.

Use **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 1.4 GB of PRBS data to the server. This corresponds to 1 million of segments of size 1400 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 (also the segment length), input 1400.

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.

```
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.
quit           - Quit.
Please input data mode (0 for IPCTEK greeting message, 1 for PRBS sequence)
Please input packet length (i.e. 1400)
Please input number of packets to be sent (i.e. 1'000'000)
378 MB sent, elapsed time 3 secs, Bit rate 945 Mbps.
756 MB sent, elapsed time 6 secs, Bit rate 944 Mbps.
1135 MB sent, elapsed time 9 secs, Bit rate 945 Mbps.
1400 MB sent, elapsed time 11 secs, Bit rate 944 Mbps.
Transmission finished.
```

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

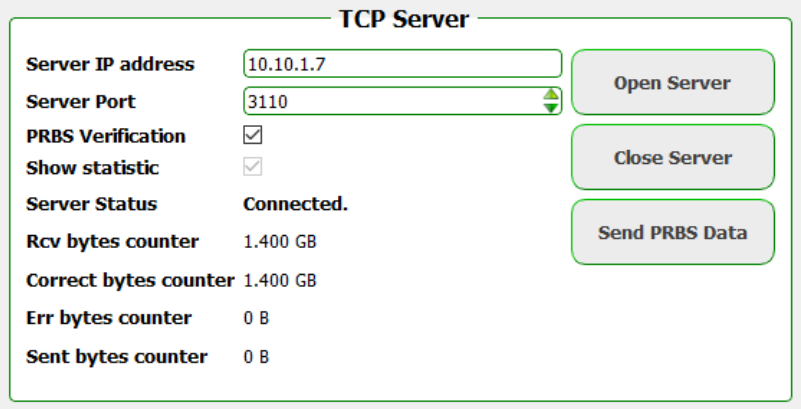


***Figure 5 - FPGA TCP Client. Client to server transfer, test bench screen***

*--- 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 1.4 GB of PRBS data to the FPGA client. This corresponds to 1 million of segments of size 1400 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 1400.

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.

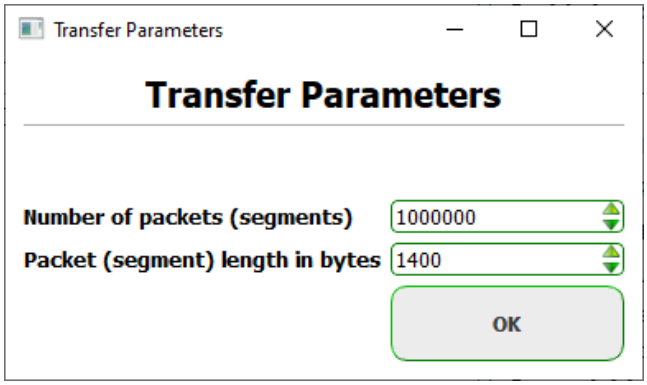


***Figure 6 - FPGA TCP Client. Server to client transfer parameters***

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

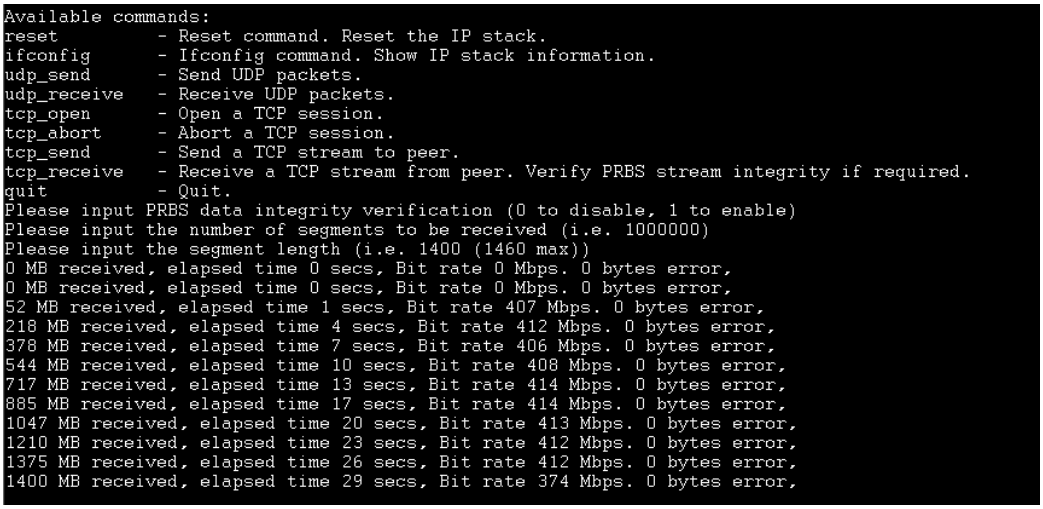


***Figure 7 - FPGA TCP Client. Server to client transfer, UART screen***

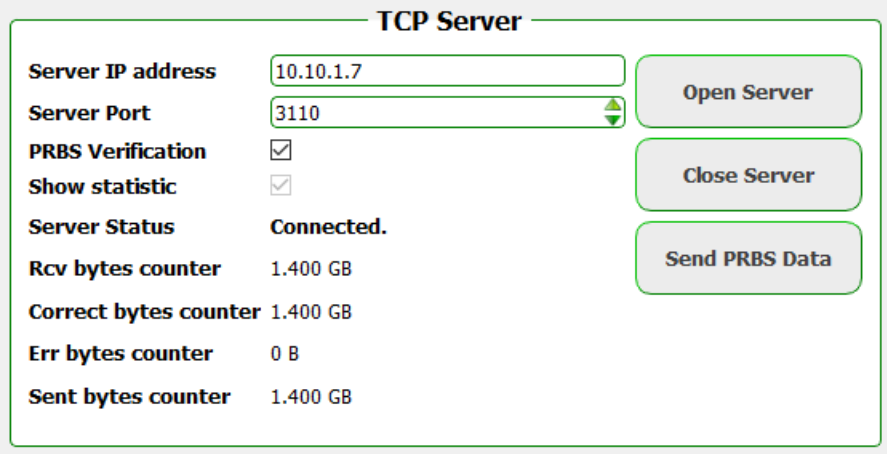


***Figure 8 - FPGA TCP Client. Server to client transfer, test bench screen***

# TCP server test

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

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

When asked for the 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 1400 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 (also the segment length), input 1400.

When asked for the number of packets (also 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 figures below.

```
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.
quit           - Quit.
Please input data mode (0 for IPCTEK greeting message, 1 for PRBS sequence)
Please input packet length (i.e. 1400)
Please input number of packets to be sent (i.e. 1'000'000)
373 MB sent, elapsed time 3 secs, Bit rate 933 Mbps.
752 MB sent, elapsed time 6 secs, Bit rate 940 Mbps.
1131 MB sent, elapsed time 9 secs, Bit rate 942 Mbps.
1400 MB sent, elapsed time 11 secs, Bit rate 943 Mbps.
Transmission finished.
```

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

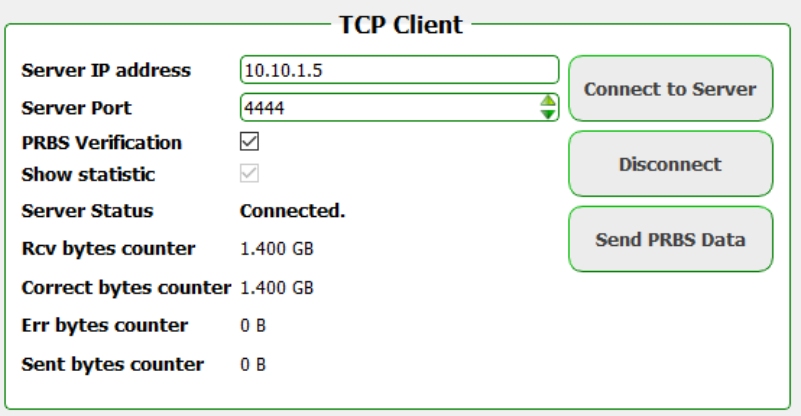


***Figure 10 - FPGA TCP Server. Server to client transfer, test bench screen***

*--- Remark----------*

*The transmission performance is slightly different from that of FPGA TCP client mode. This is mainly due to the host PC and the windows socket performance. Architecturally the FPGA TCP server and client have the same Tx and Rx engines.*

*----------*

# Client to server transfer test

In this example test we will use the Qt test bench interface to send 1.4 GB of PRBS data to the FPGA TCP server. The PRBS Verification engine in the FPGA will be enabled to verify the data integrity.

Use the **tcp_receive** command to prepare for a reception of 1.4 GB.

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

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

When asked for the packet length (also segment length), input 1400.

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 finishes are shown in figures below.

```
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.
quit           - Quit.
Please input PRBS data integrity verification (0 to disable, 1 to enable)
Please input the number of segments to be received (i.e. 1000000)
Please input the segment length (i.e. 1400 (1460 max))
0 MB received, elapsed time 0 secs, Bit rate 0 Mbps. 0 bytes error,
0 MB received, elapsed time 0 secs, Bit rate 0 Mbps. 0 bytes error,
2 MB received, elapsed time 0 secs, Bit rate 352 Mbps. 0 bytes error,
211 MB received, elapsed time 3 secs, Bit rate 518 Mbps. 0 bytes error,
412 MB received, elapsed time 6 secs, Bit rate 510 Mbps. 0 bytes error,
624 MB received, elapsed time 9 secs, Bit rate 516 Mbps. 0 bytes error,
825 MB received, elapsed time 12 secs, Bit rate 512 Mbps. 0 bytes error,
1012 MB received, elapsed time 16 secs, Bit rate 503 Mbps. 0 bytes error,
1205 MB received, elapsed time 19 secs, Bit rate 500 Mbps. 0 bytes error,
1400 MB received, elapsed time 22 secs, Bit rate 497 Mbps. 0 bytes error,
```

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

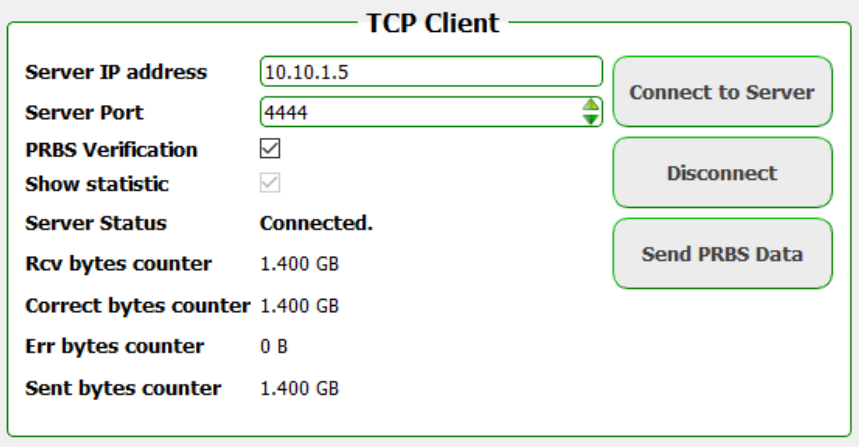


***Figure 12 - FPGA TCP Server. Client to server transfer, test bench screen***

# UDP transmitter test

In this example test we use the FPGA UDP Tx engine to send 100000 packets of size 1400 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 ***UDP Bind*** button 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. See the system log for useful information. At this stage, the npcap is called to sniff for 100000 UDP packets whose destination port is equal to 3110.

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

When ask 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, 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 1400.

When asked for the number of packets, input 100000.

When asked for the interframe gap value, input 5000 for example.

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

```
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.
quit           - Quit.
Please input destination ip address (i.e. 10.10.1.7)
Please input UDP source port (i.e. 4444)
Please input UDP destination port (i.e. 3110)
Please input the data mode (0 for IPCTEK greeting message, 1 for PRBS sequence)
Please input the packet length (i.e. 1400 (max 1472)
Please input the number of packets to be sent (i.e. 1000000)
Please input the Interframe Gap value (i.e. 10000, unit 8 ns)
87 MB sent, elapsed time 3 secs, Bit rate 218 Mbps.
140 MB sent, elapsed time 5 secs, Bit rate 218 Mbps.
Transmission finished.
```

***Figure 13 - UDP Tx test, UART screen***

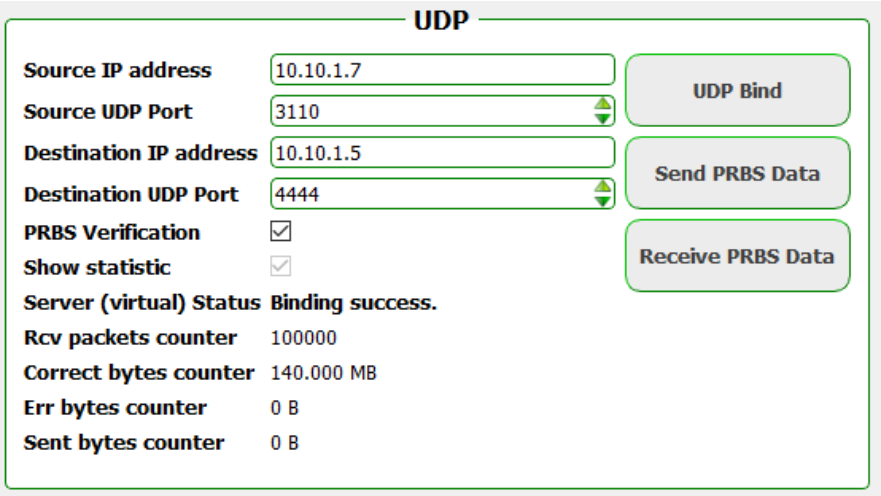


***Figure 14 - UDP Tx test, test bench screen***

*--- 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 error will 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 Qt test bench interface to send 1000000 UDP packets of size 1400 bytes to the FPGA UDP receiver. The PRBS verification module in the FPGA is also enable 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 1400.

When asked for the number of packets, input 1000000.

On the Qt test bench interface, click on the button ***Send PRBS Data*** 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 figures below.

```
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.
quit           - Quit.
Please input filtered source ip address (i.e. 10.10.1.7)
Please input filtered destination ip address (i.e. 10.10.1.5)
Please input filterd UDP source port (i.e. 3110)
Please input filtered UDP destination port (i.e. 4444)
Please input PRBS data integrity verification (0 to disable, 1 to enable)
Please input the number of packets to be received (i.e. 1000000)
Please input the packet length (i.e. 1400 (1472 max))
0 MB received, elapsed time 0 secs, bit rate 0 Mbps, 0 bytes error
0 MB received, elapsed time 0 secs, bit rate 0 Mbps, 0 bytes error
267 MB received, elapsed time 2 secs, bit rate 810 Mbps, 0 bytes error
594 MB received, elapsed time 5 secs, bit rate 813 Mbps, 0 bytes error
922 MB received, elapsed time 9 secs, bit rate 815 Mbps, 0 bytes error
1250 MB received, elapsed time 12 secs, bit rate 816 Mbps, 0 bytes error
1400 MB received, elapsed time 15 secs, bit rate 724 Mbps, 0 bytes error
```

***Figure 15 - UDP Rx test, UART screen***

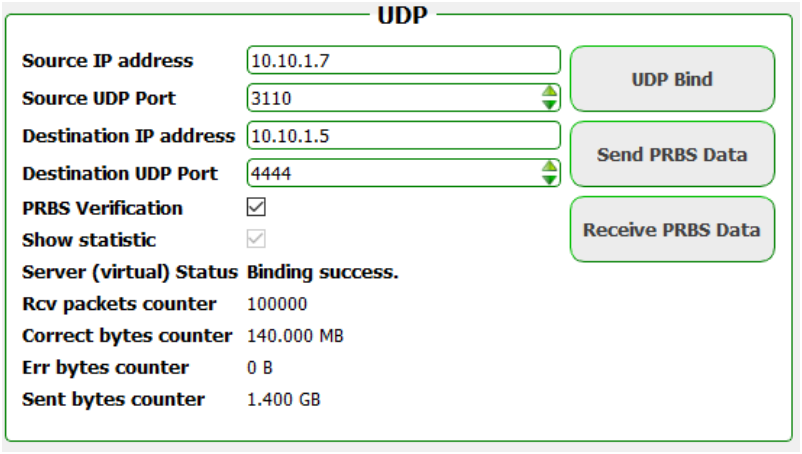


***Figure 16 - UDP Rx test, test bench screen***

*--- 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.*

*----------*

End Of Document.