INTERNET ACCESS OVER VISIBLE LIGHT COMUNICATIONS

By

Gehad Mohie El-Din Abd ElHafez

Reem Ahmed Mohamed

Rwan Nabil Mohamed

Salma Mohamed Attia

Samar Hassan Mohamed

Under supervision of

Dr. Hassan Mostafa

Dr. Tawfik Ismail

A Graduation Project Report Submitted to
the Faculty of Engineering at Cairo University
in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science
in Electronics and Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

July 2016

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

VHDL	VHSIC Hardware Description Language
PC	Personal computer
DC	Direct current
BJT	Bipolar Junction Transistor
LDR	light-dependent resistor
IBERT	Integrated bit error rate test
FCS	Frame check sequence
Wi-Fi	wireless fidelity
UART	Universal asynchronous receiver/transmitter
USB	Universal Serial Bus
EMI	Electromagnetic Interference
ADSL	Asymmetric digital subscriber line
SMA	Sub Miniature version A
JTAG	Joint Test Action Group
PCB	printed circuit board
GTP	gigabit transceiver protocol
VLAN	virtual local area network
CPRI	Common Public Radio Interface
SPI	Serial Peripheral Interface
PCI	Peripheral Component Interconnect
IP	intellectual property
TEMAC	Tri Mode Ethernet MAC
MII	Media independent interface
PHY	Physical
MMCM	Mixed-Mode Clock Manager
GMII	Gigabit Media Independent Interface
RGMII	Reduced Gigabit Media Independent Interface
IOB	Input Output buffer

DDR	Double data rate
FIFO	first in first out
AVB	Audio video bridging
MAC	Media access control
CLK	Clock
Ucf	User constraint file
DIP	Dual In-line Package
IO	input output
DRP	dynamic reconfiguration ports
PRBS	pseudo random bit sequence
PMA	Physical Medium Attachment
PLL	Phase-Locked Loop
PCS	Physical Coding Sublayer
LiFi	Light Fidelity
LED	Light Emitting Diode
FPGA	Field Programmable Gate Array
Op-Amp	Operational Amplifier
MGT	Multiple gigabit transceivers
VLC	Visible Light Communication
RF	Radio Frequency

Abstract

Looking into the crowded RF spectrum and the increasing demand for wireless communications services, it is obvious that the wireless communications field needs to expand which does not only require new technologies utilizing the same RF spectrum but also requires using a new band of frequencies so the Visible light band can be a good alternative. A typical application scenario might be to use the LEDs for high speed wireless transmission in a full network, which is known as LiFi.

Our final outcome is a complete hardwired channel using Spartan-6 Xilinx FPGA providing large speed internet access using more than one serial communication protocol that will be used to cover a medium sized room replacing WiFi and wired ADSL through Visible Light Communication.

.

ACKNOWLEDGEMENTS

Thanks to God The Almighty for giving us the patience to deliver this report with our best efforts.

Special thanks to Dr. Hassan and Dr. Tawfik for introducing the project idea, providing us with the needed hardware and giving feedbacks.

Our sincere thanks also go to Eng.Sameh , Eng. Khaled and, Eng. Karim Ismail for helping us through the FPGA, and to last year's team members that were working on the first phase of this project

Chapter 1: Introduction

1.1 Visible Light Communication

Nowadays trend is to have an environment friendly communications technologies in all aspects, so the VLC technology hit was greatly invading the communications markets and scientific theories five years ago .Having an unexploited source like the visible light spectrum in our daily life which is a license free and nearly has no health regulation on the transmitted power is a great motive to think of using it for communication purposes. The main idea of using VLC in communications is having a medium that uses visible light between 400 and 800 terahertz (THz) that works just like switching a torch on and off according to a certain pattern relaying a secret message in binary code.

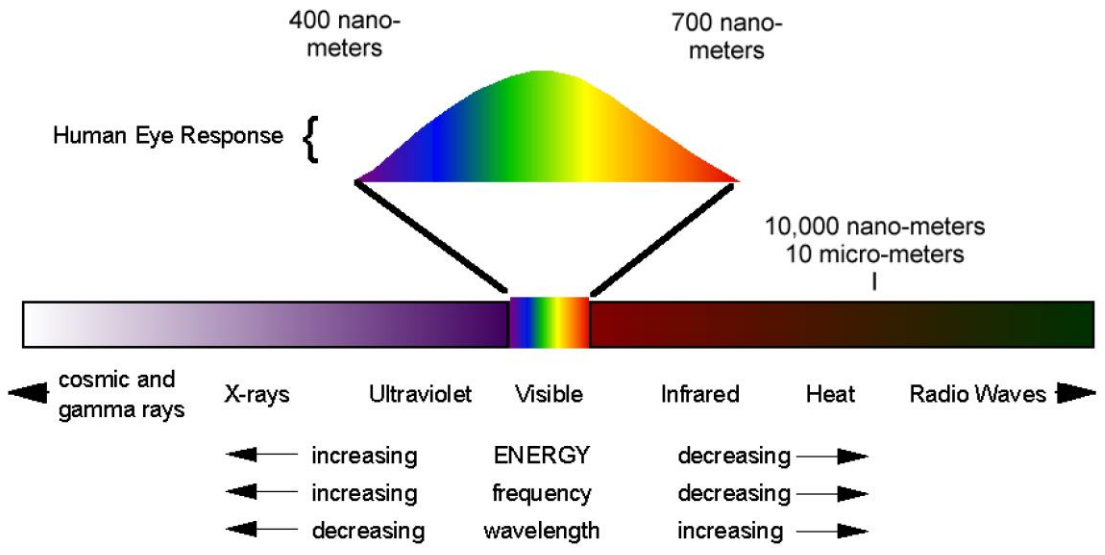


Figure 1-1: Electromagnetic spectrum

The wide bandwidth in VLC -as it works in the THz range of frequencies, the high transmission data rates reaching to Gigabits, the increased level of security for a reliable transmission of data, and the low power consumption are the main benefits that we get from the LiFi technology if it is compared with Wi-Fi.

1.2 Historical Background

The term LiFi was first coined by Professor Harold Hass- the Director of the LiFi Research and Development Center at the University of Edinburgh. LiFi is high speed wireless communications through LEDs. As LEDs become a more common source for room lighting, they're opening a new pathway for linking mobile devices to the Internet, with the potential for wider bandwidth and quicker response time than Wi-Fi, At least that's what researchers such as Harald Haas, are hoping. [1]

VLC history dates back to the 1880s, including any use of the visible light portion of the electromagnetic spectrum to transmit information. In April 2014, the Russian company Stins Coman announced the development of a Li-Fi wireless local network called BeamCaster. Their current module transfers data at 1.25 gigabytes per second but they foresee boosting speeds up to 5 Gbps in the near future. In 2014 a new record was established by Sisoft -a Mexican company- that was able to transfer data at speeds of up to 10 Gbps across a light spectrum` emitted by LED lamps. [2]

1.3 Applications

1.3.1 Security

In contrast to radio waves, the light does not pass through the walls. Therefore, with minimal precautions to avoid leakage from windows or any gaps, security is fundamentally enhanced as compared with Wi-Fi.

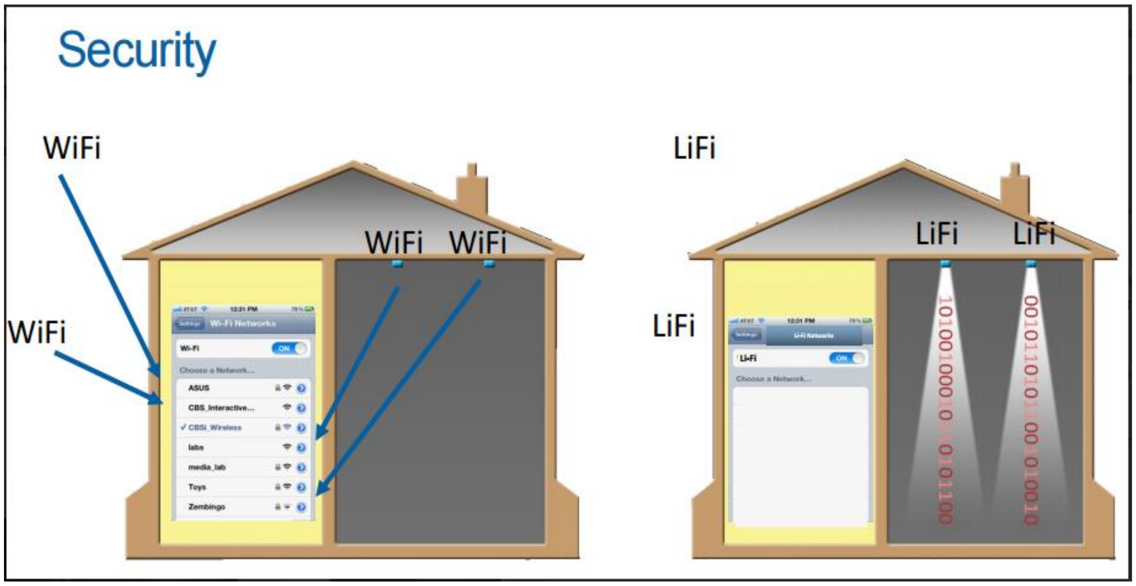


Figure 1-2: Comparison between security in LiFi and WiFi

1.3.2 EMI Sensitive Environments

On aircraft, Li-Fi enabled lighting will allow high data rate connectivity for each passenger. It will allow connectivity at all times, without creating electromagnetic interference (EMI) with sensitive radio equipment on the flight deck. The reduction in cabling requirement also means a lighter aircraft. In explosion hazard environments, the use of electrical equipment, including mobile phones, is generally greatly restricted. The use of Li-Fi to pass data will simplify the configuration of data networks in such environments, and can enable new systems to enhance security in these environments.

Hospitals are a specific case of an environment where both EMI sensitivity and security of data are issues. Li-Fi can enable the better deployment of secure networked medical instruments, patient records, etc.

1.3.3 Cellular Communication

In external urban environments, the use of Li-Fi enabled street lamps would provide a network of internet access points. In cellular communication, the distance between radio base stations has come down to about 200-500 meters. So, instead of deploying new radio base stations in our cities, street lamps could provide both, illumination during night, and high speed data communication 24/7. Surprisingly, even when the lights are off as perceived by the eye, full data communication rates are still possible. There is also an additional cost benefit as installing new radio base stations usually comes with large cost – for installation and site lease**.** Apple looks set to include a li-fi capability in future versions of the iPhone, meaning it can access high-speed data using lighting. The iPhone's operating system now openly references li-fi capability in its programming code. Apple already holds a patent on using its camera to capture data as well as images, so the company is well placed to exploit the new technology. [3]

1.3.4 Augmented Reality

Exhibits in museums and galleries are illuminated with specific lighting. Li-Fi enabled lighting can provide localized information within that light. This means that a visitor's camera or mobile phone can be used to download further information regarding the object being viewed from the light that illuminates the exhibit.

1.3.5 Local Advertising

By using shop display lighting as a Li-Fi broadcast channel, it is possible to transmit advertising information on the goods being viewed, as well as say special offers and coupons. This will allow the merging of the high street and online shopping experience, and provide novel retail business models to emerge. Catalogue information, discount coupons, and advertising videos could all be provided to shoppers.

1.3.6 Underwater Communication

Radio waves are quickly absorbed in water, preventing underwater radio communications, but light can penetrate for large distances. Therefore, Li-Fi can enable communication from diver to diver, diver to mini-sub, diver to drilling rig, etc.

Figure 1-3: Visible light communication underwater

1.4 Motivation

This project is considered complementary for another graduation project that was proposed last year at our department under the title "Visible Light Communication System Over FPGA". They were finally able to send Data from the FPGA to the laptop and the opposite using USB-to-UART module of the Spartan-6 FPGA through the VLC interface. On the PC they used a program called **TeraTerminal** , a PC ports emulator through which they were able to access the USB port to send frames of certain format with a start and end bits used for error checking. Through this setting they reached a final speed of few Kbps at a distance around 1m. Below is their block diagram using only one FPGA for both the uplink and downlink

In the second phase of the project the target is to create a hardware system for internet access over VLC with speed ranging around tens of Mbps to cover a medium sized room

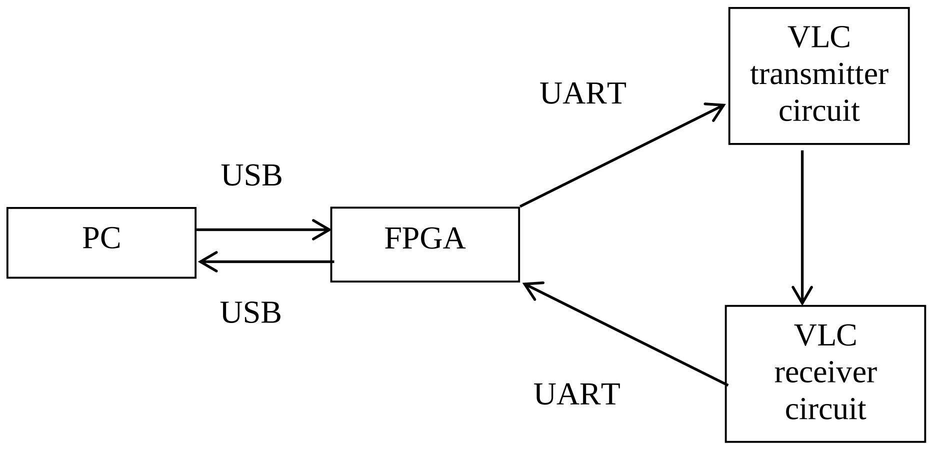


Figure 1-4: Phase one block diagram

Our basic motive is that we believe VLC is very promising in Egypt specially using it for local internet access that suffers from many issues in Egypt right now and has an increasing demand for the speeds. Having a look over the VLC market revenue in different regions that started using it, VLC/ LiFi market size is expected to grow with Compound Annual Growth Rate (CAGR) of 87.313% from 2014 to 2020. The VLC/Li-Fi technology is still in the research and development phase. However, the commercialization of the technology will start taking place from 2015. Till 2016, the market will enter into almost all the major application areas of VLC technology. [4]

Table 1-1: LiFi market revenue in different cities

Region	**2012**	**2014**	**2016**	**2018**	**CAGR% (2013-2018)**
North America	29.58	203.91	824.5	1,940.42	83.9
Europe	24.89	180.8	743.68	1,769.48	85.4
APAC	33.09	233.35	905.43	1,992.12	79.5
ROW	8.52	60.49	217.55	436.01	73.9
Total	96.08	678.54	2,691.08	6,138.02	82.00

Beside the previous reasons, our personal motive to choose the idea for our graduation project is its learning outcome. Since the idea is rich in the research content, implies a technology that in new to our study and has high demand in the market and involve a hardware optimization and permit exposure to many software platforms.

1.5 Chapters Summary

In chapter 2 we will illustrate the description of our system showing its block diagram and our flow of work. In chapter 3 we provide a detailed illustration for our software approach, the platforms we used and the hardware coding and how we dealt with the kit and the already made IP cores for the communications standards. In chapter 3 we show our hardware approach through clear images of the circuits and simulations using analog circuit simulators. Finally, In chapter 5 we provide a simple method for monitoring the system channel through another VHDL code. The codes will be provided in an external CD attached with the project report.

Chapter 2: **System Description**

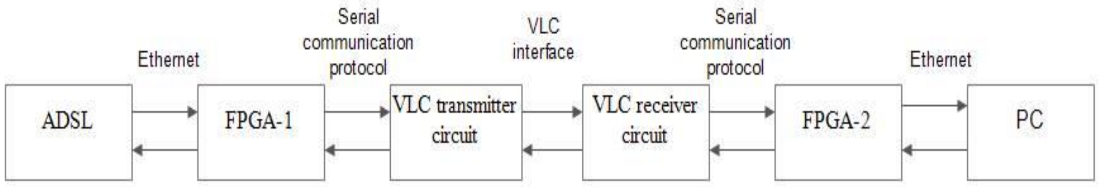


Figure 2-1: System block diagram

The block diagram illustrates the targeted final outcome of our system, wireless internet access will be provided for the user (PC) in a bidirectional channel for both uplink and downlink. The first FPGA functionality is to process on the coming Ethernet frames to be able to send it on the wireless visible light medium through a certain communication protocol that is compatible with the desired high speeds that will hopefully be in the range of tens of Mbps. The transmitter circuit is simply the circuit that has the blinking LEDs, alternatively the receiver circuit is holds the photo sensors. The second FPGA functionality is to do the reverse function of the first one, it processes on the coming electrical signal to eventually convert it back to the original Ethernet frames. For the bidirectional channel both circuits should be replaced by a transceiver that contains both the LEDs and the photodiodes, and both FPGA should have both functions.

2.1 Flow of Building up the System

Our approach to build the system will be hierarchical we will start by the software approach till we have the two FPGA working correctly using SMA connectors instead of the VLC interface .Then we will proceed in connecting the transmitter and receiver circuits for the downlink channel . After the system is stable we will go for the uplink channel, enhancing the power and distance. Finally, there will be an addition in the form of a software user interface to monitor the system or accessing the internet using it.

A detailed flow chart for the system build up is illustrated below.

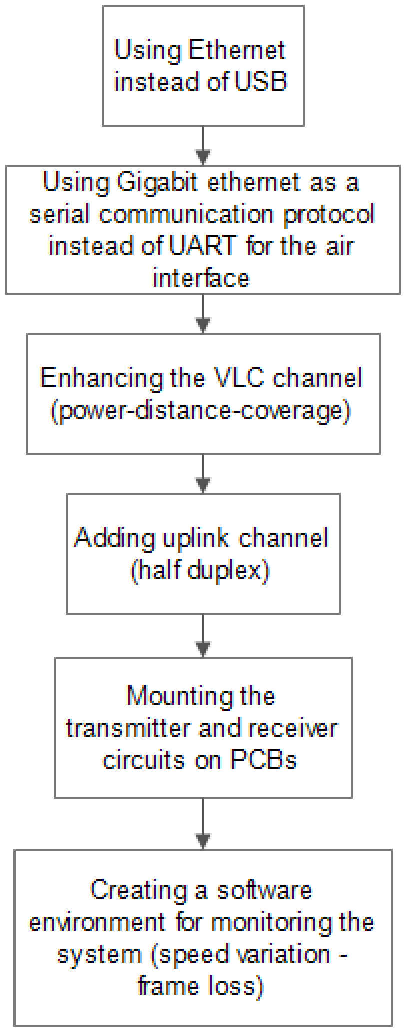

Using Ethernet instead of USB
Using Gigabit ethernet as a serial communication protocol instead of UART for the air interface
Enhancing the VLC channel (power-distance-coverage)
Adding uplink channel (half duplex)
Mounting the transmitter and receiver circuits on PCBs
Creating a software environment for monitoring the system (speed variation - frame loss)

Figure 2-2: Flow chart for project work

Chapter 3: **Software Approach**

3.1 ISE Software Platform

The SP605 evaluation kit includes entitlement to a seat that permits the ISE Design Suite: Logic Edition to be used with a Spartan-6 XC6SLX45T FPGA.

By using ISE we have the availability of burning IP core which is a reusable unit of logic, cell, or chip layout design that is the intellectual property of one party. IP cores may be licensed to another party or can be owned and used by a single party alone. The IP cores can be used as building blocks within FPGA logic designs. Included with the ISE some tools that we used in our work

- **ISim**: which provides a complete, full-featured HDL simulator integrated within ISE. HDL simulation now can be an even more fundamental step within the design flow with the tight integration of the ISim within your design environment.
- **ChipScope Pro system**: ChipScope is an embedded, software based logic analyzer for hardware debugging. By inserting an “integrated controller core” (icon) and an “integrated logic analyzer” (ila) into your design and connecting them properly, you can monitor any or all of the signals in your design. ChipScope provides you with a convenient software based interface for controlling the “integrated logic analyzer,” including setting the triggering options and viewing the waveforms. [5]

The two blocks requiring VHDL coding are the Ethernet and the serial communication protocol at both the transmitting and receiving sides. The procedure followed through the software approach was:

- Generating the cores and writing the code of each protocol separately after understanding the core specifications and functions of its input and output signals.
- Software Simulation
- Hardware Test
- Writing the top level code

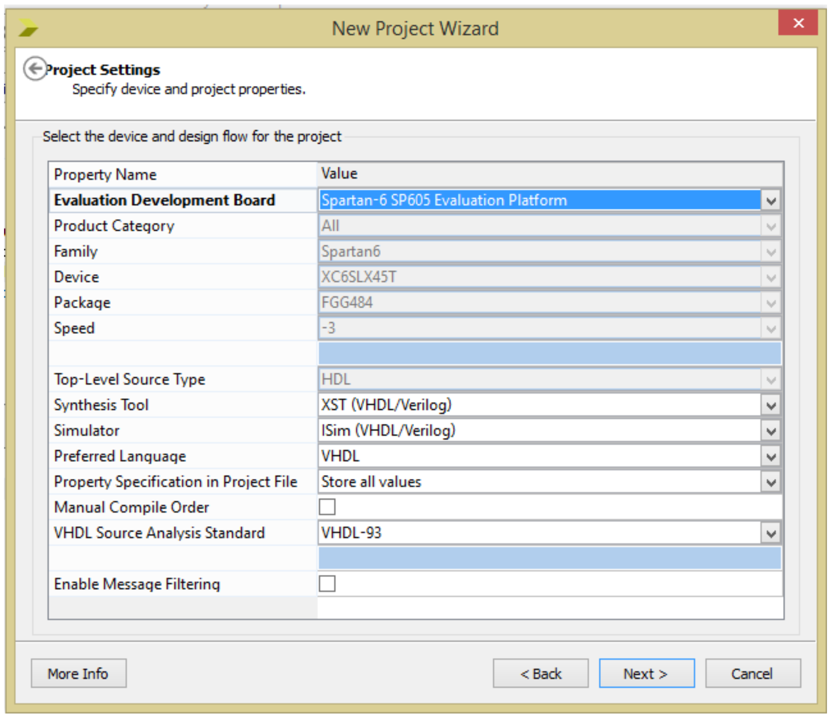
We started first by burning simple VHDL codes as AND gate with switches and LEDs to get familiar with the steps of ISE platform and the SP605 Evaluation kit.

3.2 The Ethernet IP Core

3.2.1 Generating the Core

- Start a New Project wizard where you choose the Xilinx board used and the hardware description language of the generated core whether VHDL or Verilog and the simulation tool Isim.

-


New Project Wizard

Project Settings
Specify device and project properties.

Select the device and design flow for the project

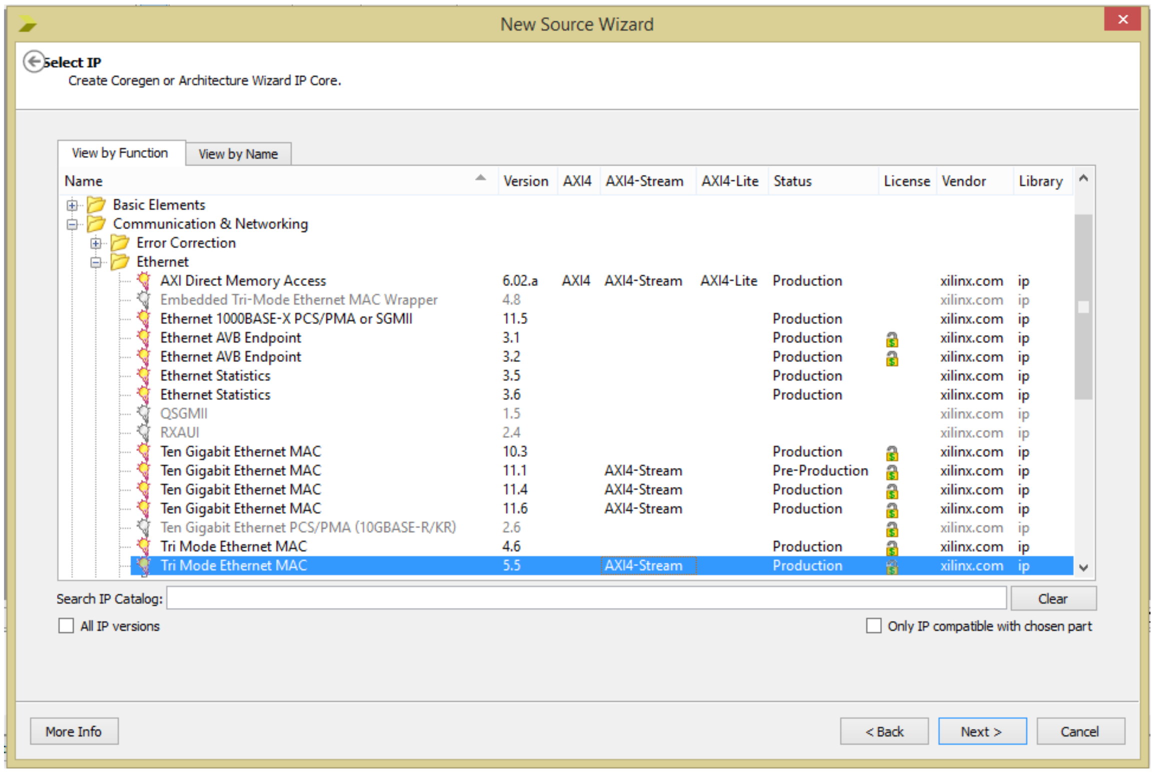
Property Name	Value
Evaluation Development Board	Spartan-6 SP605 Evaluation Platform
Product Category	All
Family	Spartan6
Device	XC6SLX45T
Package	FGG484
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Figure 3-1: New project wizard window

- Select the IP core.

We used Tri Mode Ethernet MAC "TEMAC" version 5.5 which is the AXI version of this core

The AXI Interconnect IP connects one or more AXI memory-mapped Master devices to one or more memory-mapped Slave devices. The AXI interfaces conform to the AMBA® AXI version 4 specifications from ARM®, including the AXI4-Lite control register interface subset. The Interconnect IP is intended for memory-mapped transfers only. [6]


New Source Wizard

Select IP
Create Coregen or Architecture Wizard IP Core.

View by Function | View by Name

Name	Version	AXI4	AXI4-Stream	AXI4-Lite	Status	License	Vendor	Library
Basic Elements								
Communication & Networking								
Error Correction								
Ethernet								
AXI Direct Memory Access	6.02.a	AXI4	AXI4-Stream	AXI4-Lite	Production		xilinx.com	ip
Embedded Tri-Mode Ethernet MAC Wrapper	4.8						xilinx.com	ip
Ethernet 1000BASE-X PCS/PMA or SGMII	11.5				Production		xilinx.com	ip
Ethernet AVB Endpoint	3.1				Production	🔒	xilinx.com	ip
Ethernet AVB Endpoint	3.2				Production	🔒	xilinx.com	ip
Ethernet Statistics	3.5				Production		xilinx.com	ip
Ethernet Statistics	3.6				Production		xilinx.com	ip
QSGMII	1.5						xilinx.com	ip
RXAUI	2.4						xilinx.com	ip
Ten Gigabit Ethernet MAC	10.3				Production	🔒	xilinx.com	ip
Ten Gigabit Ethernet MAC	11.1		AXI4-Stream		Pre-Production	🔒	xilinx.com	ip
Ten Gigabit Ethernet MAC	11.4		AXI4-Stream		Production	🔒	xilinx.com	ip
Ten Gigabit Ethernet MAC	11.6		AXI4-Stream		Production	🔒	xilinx.com	ip
Ten Gigabit Ethernet PCS/PMA (10GBASE-R/KR)	2.6					🔒	xilinx.com	ip
Tri Mode Ethernet MAC	4.6				Production	🔒	xilinx.com	ip
Tri Mode Ethernet MAC	5.5		AXI4-Stream		Production	🔒	xilinx.com	ip

Search IP Catalog: Clear

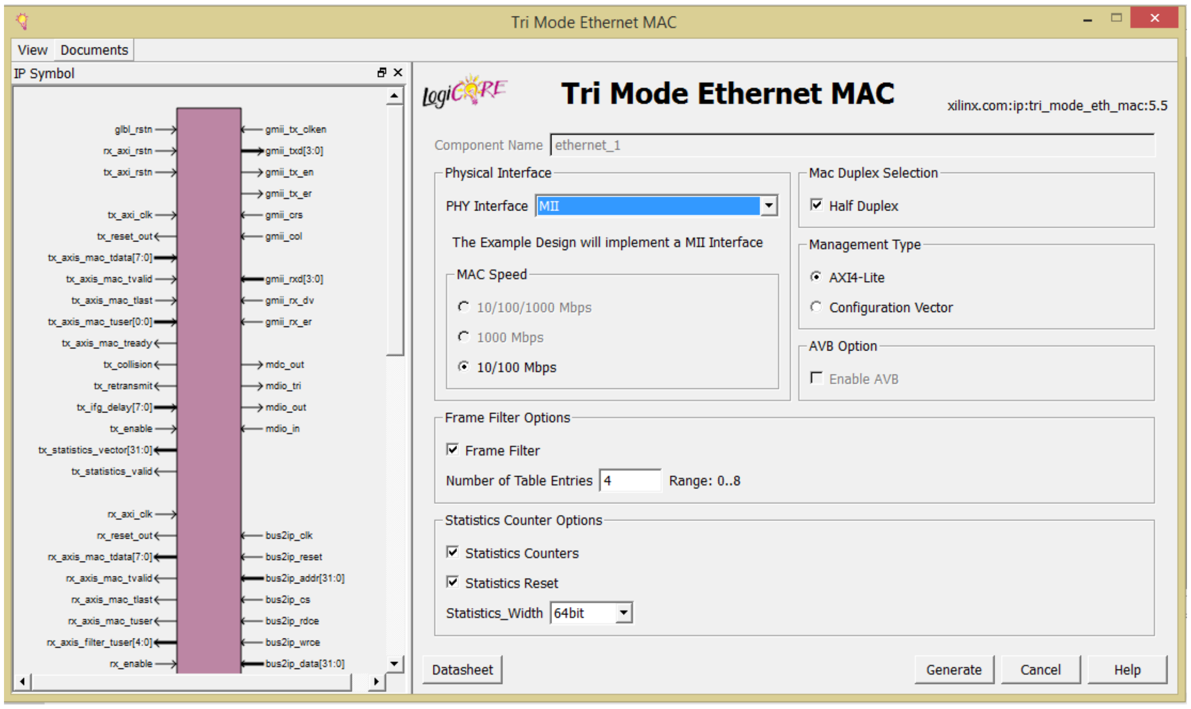
All IP versions Only IP compatible with chosen part

More Info < Back Next > Cancel

Figure 3-2: TEMAC new source wizard window

- Specify the desired core specifications.

Choose the PHY layer to be MII which is the standard interface used to connect a fast Ethernet standardized by **IEEE 802.3u,** it uses a clock of 25 MHZ frequency and 4 data lines to achieve (100Mbit/s).



Tri Mode Ethernet MAC

xilinx.com:ip:tri_mode_eth_mac:5.5

Component Name: ethernet_1

Physical Interface

PHY Interface: MII

The Example Design will implement a MII Interface

MAC Speed: ( ) 10/100/1000 Mbps ( ) 1000 Mbps [✓ 10/100 Mbps]

Mac Duplex Selection: [✓ Half Duplex]

Management Type: [✓ AXI4-Lite] ( ) Configuration Vector

AVB Option: ☐ Enable AVB

Frame Filter Options: [✓ Frame Filter]

Number of Table Entries: 4 Range: 0..8

Statistics Counter Options: [✓ Statistics Counters] [✓ Statistics Reset]

Statistics_Width: 64bit

Figure 3-3: TEMAC wizard window

3.2.2 Using Example Design

Attached with the generated core a complete example design that has certain features and utilizes the TEMAC core as a HDL block. It has several options for example it can operate as a pattern generator, it has a pattern checker and the received data can be looped back. This allows the functionality of the core to be demonstrated either using a simulation package or in hardware, if placed on a suitable board, also the codes of the example design are available as an open source unlike the core so they can be modified and customized to be compatible with the desired performance, and this is specifically what we went through. The example design has the following features.

- An instance of the TEMAC solution

- Clock management logic, including MMCM and Global Clock Buffer instances.
- MII, GMII or RGMII interface logic, including IOB and DDR registers instances.
- Statistics vector decode logic.
- AXI4-Lite to IPIF interface logic.
- User Transmit and Receive FIFOs with AXI4-Stream interfaces.
- User basic pattern generator module with a frame generator and frame checker plus loopback logic.
- User AVB pattern generator module providing a second frame generator and frame checker for designs including the AVB Endpoint.
- A simple state machine to bring up the PHY (if any) and MAC ready for frame transfer. [7]

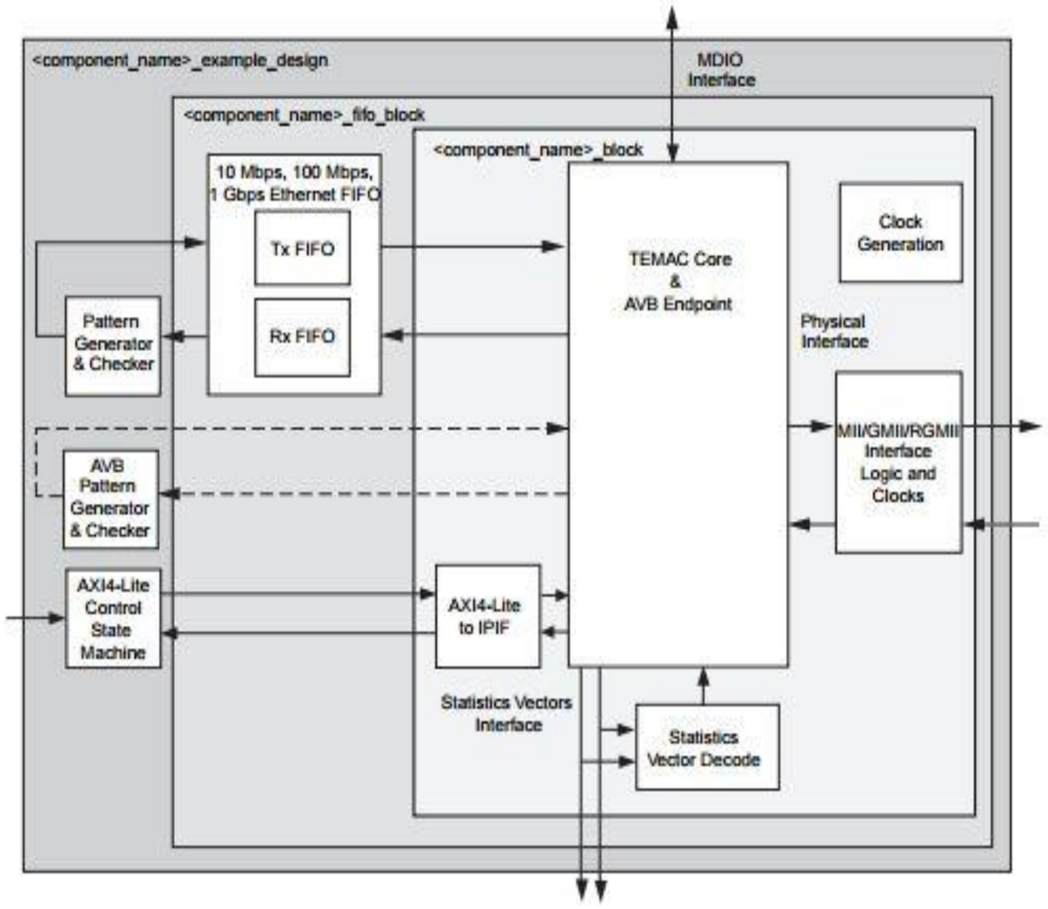

<component_name>_example_design
MDIO Interface
<component_name>_fifo_block
<component_name>_block
10 Mbps, 100 Mbps, 1 Gbps Ethernet FIFO
Tx FIFO
Rx FIFO
TEMAC Core & AVB Endpoint
Clock Generation
Physical Interface
Pattern Generator & Checker
AVB Pattern Generator & Checker
MII/GMII/RGMII Interface Logic and Clocks
AXI4-Lite Control State Machine
AXI4-Lite to IPIF
Statistics Vectors Interface
Statistics Vector Decode

Figure 3-4: Illustrates the top-level design for the TEMAC solution example design

- After the core is generated, start including the files of the example design in your project from "Add source". Include all the .vhd and .ucf files including the test bench and the all of the cores available in the ChipScope Pro system.

- Burn the example design on the FPGA by clicking on "Generate Programming File" then "Configure Target Device” after connecting the USB JTAG. The JTAG chain can be used to program the FPGA and access the FPGA for hardware and software debugging.

- We were able to observe the packets on the Wireshark when sending from the FPGA to the PC by enabling the Pattern generator "Turn the third DIP switch ".

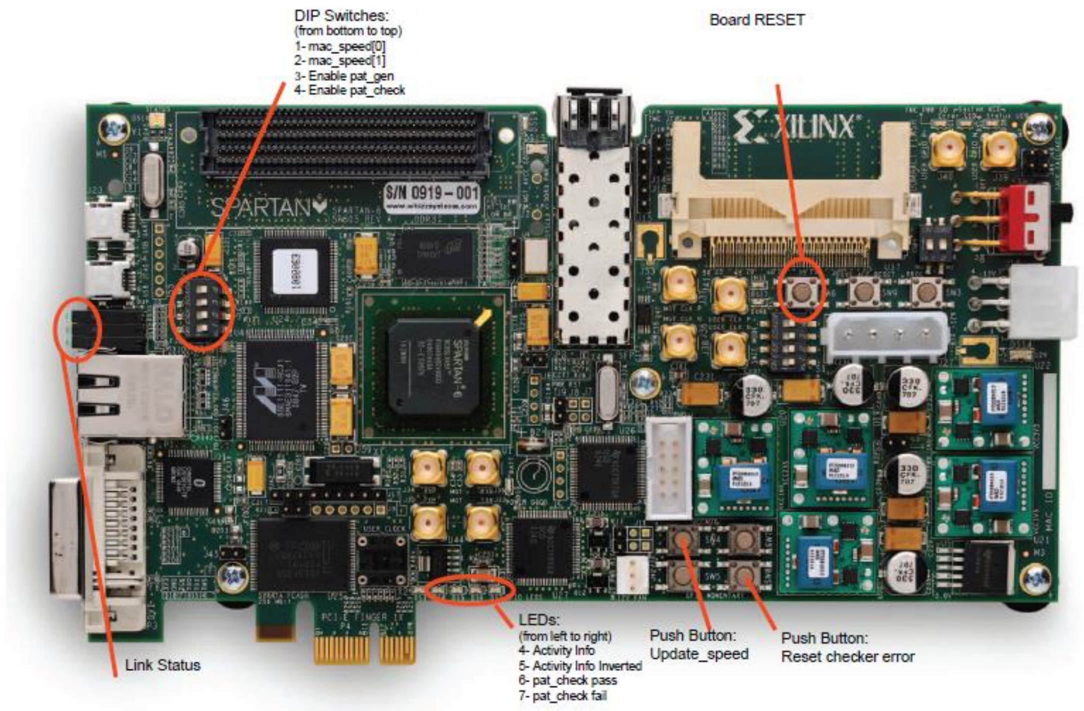


Figure 3-5: Spartan-6 evaluation kit

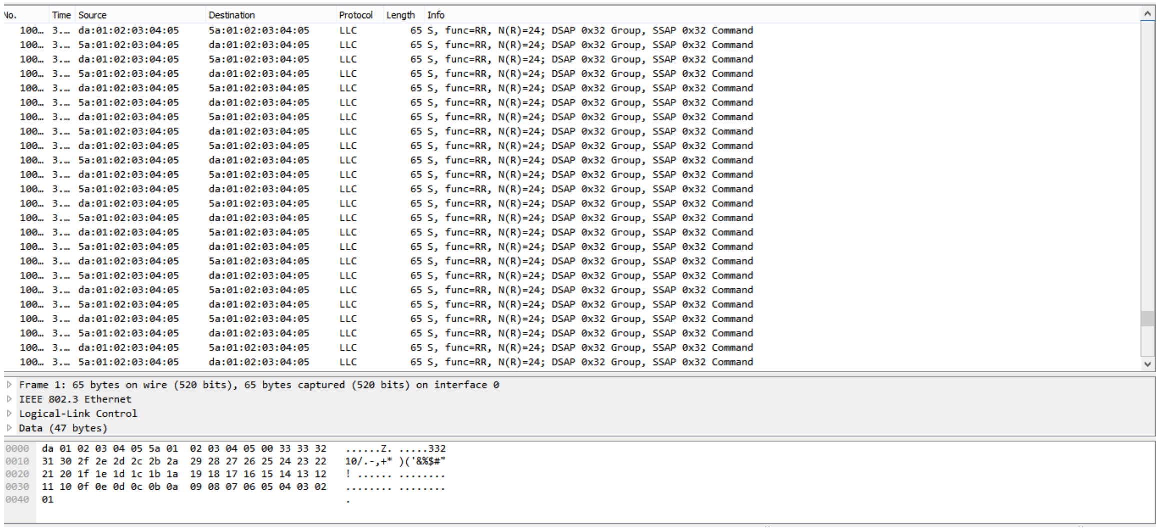
- By disabling the pattern generator, we managed to send the packets from the PC to FPGA using. **SoftPerfect Network Protocol Analyzer** which is a free professional tool for analyzing, debugging, maintaining and monitoring local networks and Internet connections. It captures the data passing through the dial-up connection or Ethernet network card, analyses this data and then represents it in a readable form.

- We sent those packets to the address swap block and the packets were looped back after reversing the source and destination address

3.2.3 Hardware Testing

- Burn the example design on the FPGA "Generate Programming File" then "Configure Target Device" after connecting the USB JTAG. The JTAG chain can be used to program the FPGA and access the FPGA for hardware and software debug.

- We were able to observe the packets on the Wireshark when sending from the FPGA to the PC by enabling the Pattern generator "Turn the third DIP switch ".

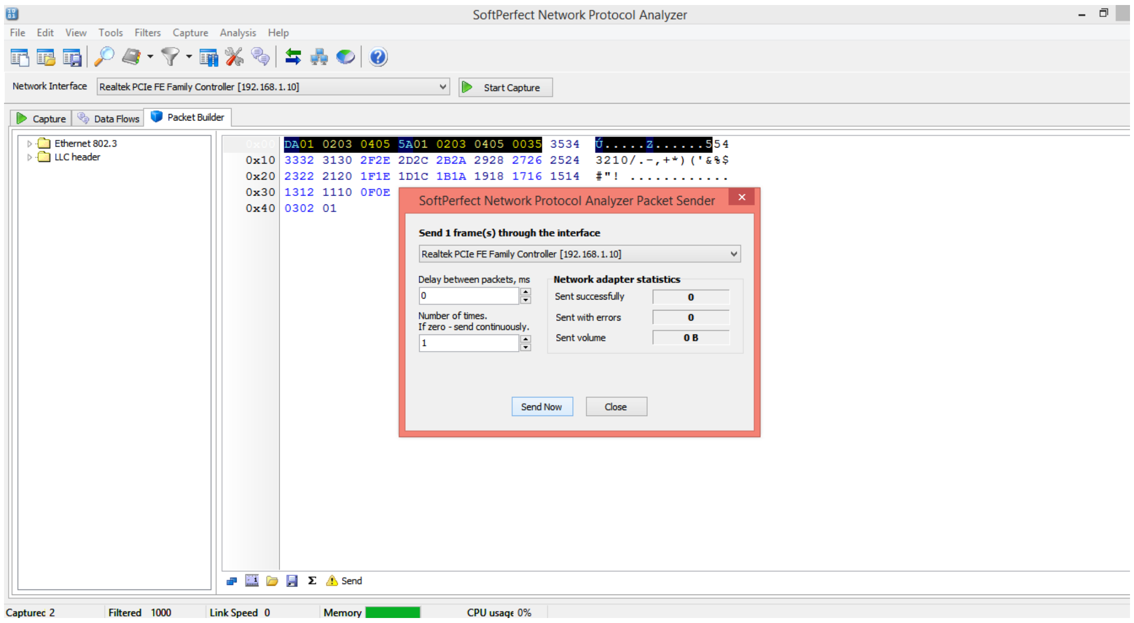


```
No.    Time Source              Destination          Protocol  Length Info
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  da:01:02:03:04:05   5a:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command
 100…  3…  5a:01:02:03:04:05   da:01:02:03:04:05    LLC           65 S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Command

▷ Frame 1: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
▷ IEEE 802.3 Ethernet
▷ Logical-Link Control
▷ Data (47 bytes)

0000  da 01 02 03 04 05 5a 01  02 03 04 05 00 33 33 32   ......Z. .....332
0010  31 30 2f 2e 2d 2c 2b 2a  29 28 27 26 25 24 23 22   10/.-,+* )('&%$#"
0020  21 20 1f 1e 1d 1c 1b 1a  19 18 17 16 15 14 13 12   ! ...... ........
0030  11 10 0f 0e 0d 0c 0b 0a  09 08 07 06 05 04 03 02   ........ ........
0040  01                                                 .
```

Figure 3-6: Frames arriving from FPGA to PC on Wireshark

- Disabling the pattern generator, we managed to send the packets from the PC to FPGA using. SoftPerfect Network Protocol Analyzer which is is a free professional tool for analyzing, debugging, maintaining and monitoring local networks and Internet connections. It captures the data passing through the dial-up connection or Ethernet network card, analyses this data and then represents it in a readable form.

- Finally we generated some Ethernet frames **SoftPerfect Network Protocol Analyzer** and forwarded them to the FPGA which has the address swap block activated and the frames were looped back after reversing the source and destination address.



Network Interface: Realtek PCIe FE Family Controller [192.168.1.10]

```
0x00  DA01 0203 0405 5A01 0203 0405 0035 3534   Ú.....Z.......554
0x10  3332 3130 2F2E 2D2C 2B2A 2928 2726 2524   3210/.-,+*)('&%$
0x20  2322 2120 1F1E 1D1C 1B1A 1918 1716 1514   #"! ............
0x30  1312 1110 0F0E
0x40  0302 01
```

Send 1 frame(s) through the interface

Realtek PCIe FE Family Controller [192.168.1.10]

Delay between packets, ms: 0

Number of times. If zero - send continuously.: 1

Network adapter statistics

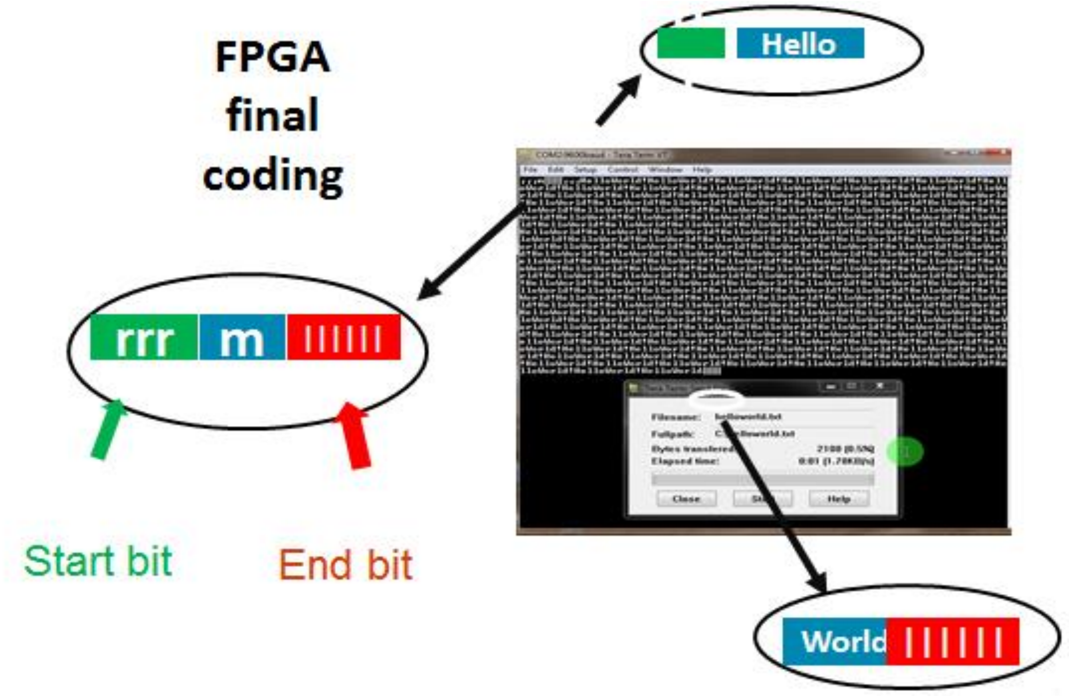
Sent successfully	0
Sent with errors	0
Sent volume	0 B

Figure 3-7: Ethernet frames generation using SoftPerfect Network Protocol Analyzer

3.3 The Serial Communication Protocol

Serial signaling is the preferred choice in all markets such as telecom, datacom, computing, and storage, it supports very high multi-gigabit data rates, avoids clock/data skew by using embedded clock, power consumption. It is the only IO technology that meets today's high-speed requirements as it is faster than parallel signaling. Parallel I/O data rates are inherently limited due to unavoidable skew between clock lines and multiple data lines.

As aforementioned the initial prototype of last year's team used UART as the serial communication protocol for the VLC transmitter and receiver interface with the FPGA. The transmitter takes bytes of the data and sends each bit sequentially with its own clock it also adds start, stop and parity check bits for error checking. At the receiver they had another UART code which re-assembles the bits into complete bytes with its own clock after extracting the added start, stop and parity bits.



Filename: helloworld.txt
Fullpath: C:\...helloworld.txt
Bytes transfered: 2100 (0.5%)
Elapsed time: 0:01 (1.78KB/s)

Figure 3-8: First phase output on teraterminal

If using 8-bit data, 1 start, 1 stop, and no parity bits, the effective throughput is: 115200 * 8 / 10 = 92160 bits/sec.

UART is limited by the max Baud Rate. The main problem with this protocol is its speed which is not compatible with the internet access over VLC that one of its main target is enhancing the speeds. Also we already used the 10 Mbps Ethernet so if we used UART after Integrating the two protocols one with high speed and the other with low speed, we will be restricted to the lower one . [8]

The second alternative we thought of was SPI. The SPI bus is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems. Data link setup with a Master / Slave interface and can support up to 1 mega baud or 10Mbps of speed.

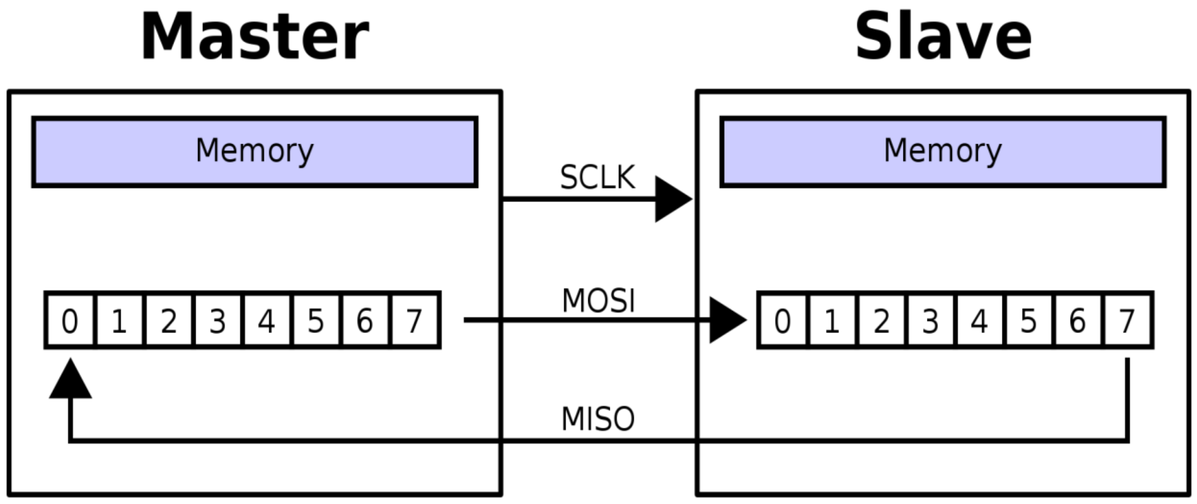


Figure 3-9: SPI protocol

The speeds of SPI could have been enough for our system, but the problem was that its IP core is not supported by Spartan-6 license. So we started looking for an open source code online that we can overwrite some of its lines to be compatible with our system. However, the codes required many details for the clocking synchronization and modifications for the hardware of the transmitter and receiver circuits, that made it better to look for an alternative for the protocol. Moreover, using another protocol that supports higher data rates even if we would not use them in the initial prototype would be good for future enhancements.

Finally we decided to use the Gigabit transceivers of the Spartan-6 that uses SMA connectors instead of the ordinary copper wires and can support data rates up to 1.25 GHz. The transceivers' wizard has the following features.

• Creates customized HDL wrappers to configure Spartan-6 FPGA GTP transceivers

• Predefined templates automate transceiver configuration for industry standard protocols such as: CPRI™, Display Port, Gigabit Ethernet, High-Definition Serial Digital Interface (HD-SDI), Open Base Station Architecture Initiative (OBSAI), PCI EXPRESS ® (PCIe ®) generation I, Serial RapidIO, 10 Gb Attachment Unit Interface (XAUI), Aurora 8B/10B, Serial Advanced Technology Attachment (SATA) 1.5 Gb/s, and SATA 3 Gb/s.

• Custom protocols can be specified using the **Start from Scratch** option in the GUI.

• Automatically configures transceiver analog settings of the Spartan-6 FPGA GTP transceivers

• Supports 8B/10B encoding/decoding.

• Includes an example design with a companion test bench as well as implementation and simulation scripts. [9]

The main reason behind the high data rates supported by the GTP transceivers is the usage of the SMA connectors. They are semi-precision coaxial RF connectors developed in the 1960s as a minimal connector interface for coaxial cable with a screw type coupling mechanism. The connector has a 50 Ω impedance. SMA is designed for use from DC to 18 GHz, but is most commonly encountered with WiFi antenna systems and USB Software Defined Radio dongles. [10]

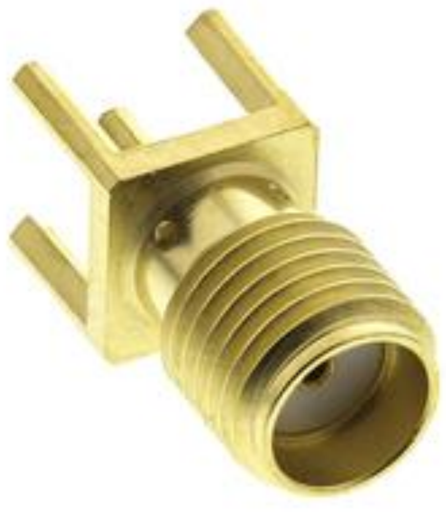


Figure 3-10: SMA socket

3.3.1 Gigabit Ethernet

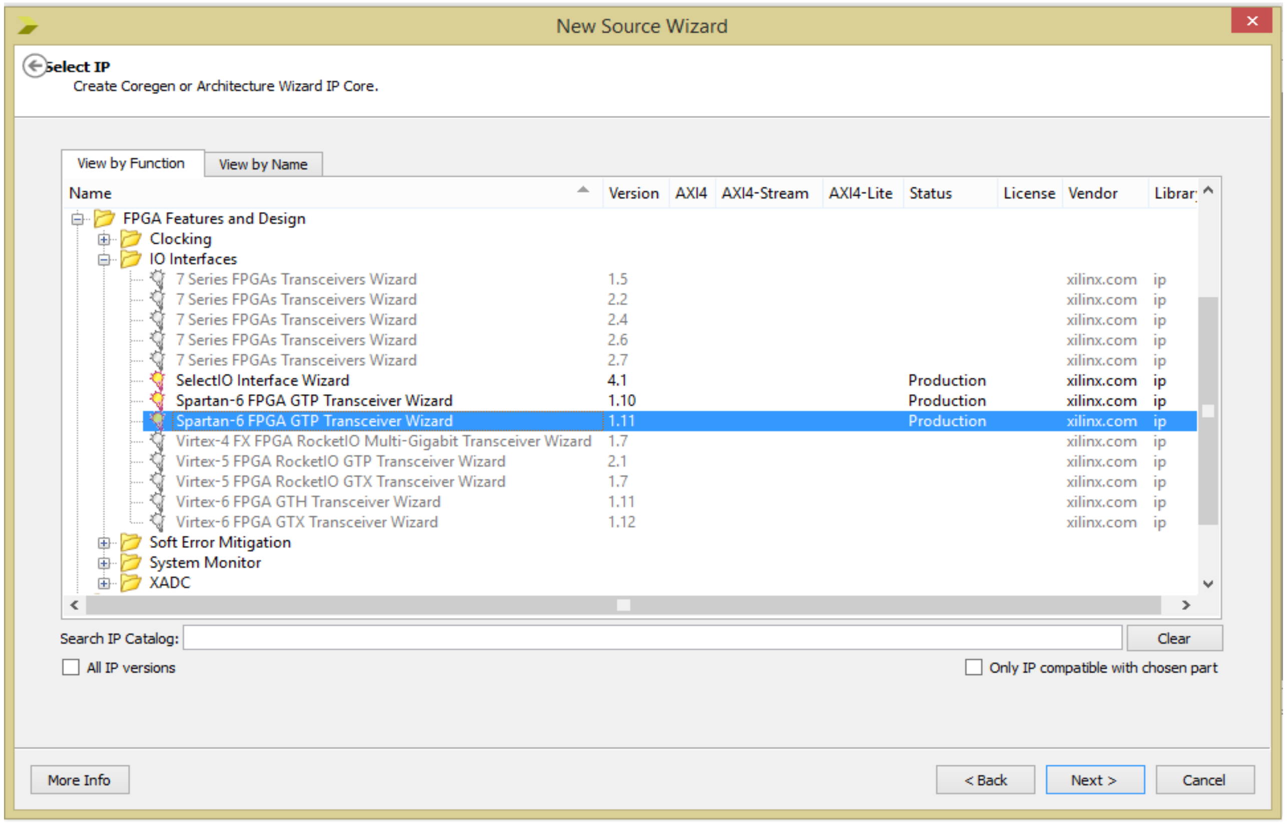
The Gigabit transceivers support many protocols as mentioned but we chose to work with Gigabit Ethernet defined by the IEEE 802.3-2008 standard. Gigabit Ethernet builds on top of the Ethernet protocol, but increases speed tenfold over Fast Ethernet to 1000 Mbps, or 1 gigabit per second (Gbps). This protocol, which was standardized in June 1998, promises to be a dominant player in high-speed local area network backbones and server connectivity.

It has been decided that Gigabit Ethernet will look identical to Ethernet from the data link layer upward but several changes has been made to the physical interface of Ethernet. Leveraging these two technologies means that the standard can take advantage of the existing high-speed physical interface technology of Fiber Channel while maintaining the IEEE 802.3 Ethernet frame format, backward compatibility for installed media, and use of full- or half-duplex carrier sense multiple access collision detect (CSMA/CD). This scenario helps minimize the technology complexity, resulting in a stable technology that can be quickly developed.

3.3.1.1 Generating the Core

- Follow the same steps that were illustrated in generating the Ethernet IP core in choosing the kit.

- Choose the GTP V1.11 IP core which has the previously mentioned features.



New Source Wizard

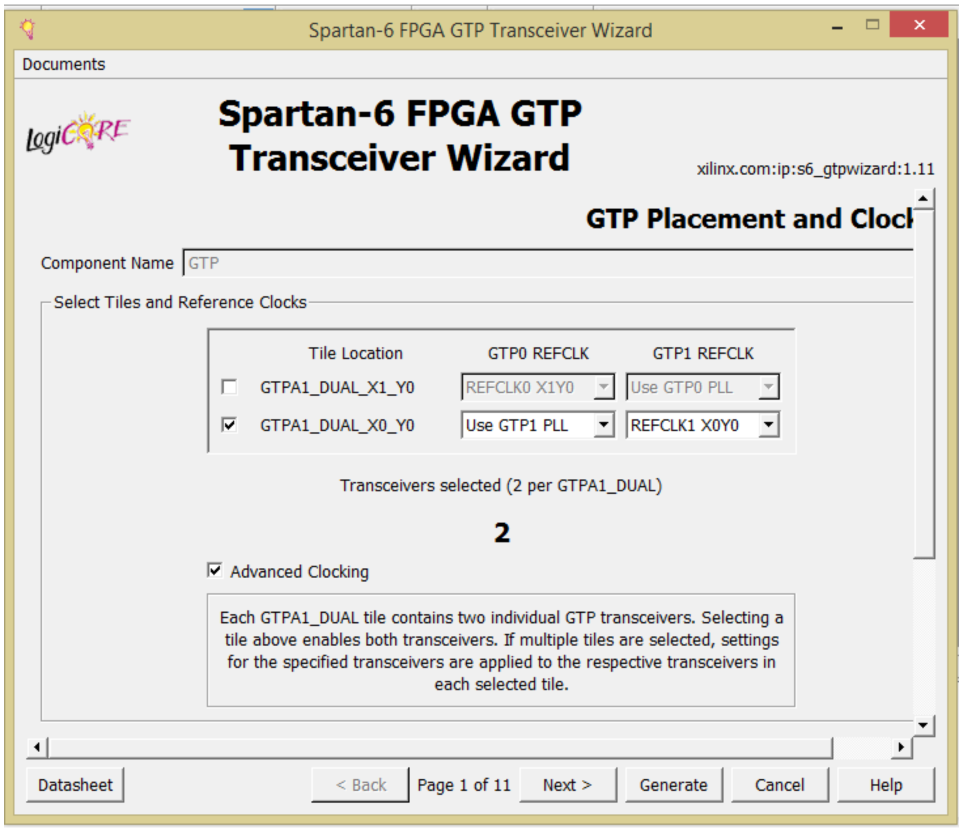
Select IP
Create Coregen or Architecture Wizard IP Core.

Name	Version	AXI4	AXI4-Stream	AXI4-Lite	Status	License	Vendor	Librar
FPGA Features and Design								
Clocking								
IO Interfaces								
7 Series FPGAs Transceivers Wizard	1.5						xilinx.com	ip
7 Series FPGAs Transceivers Wizard	2.2						xilinx.com	ip
7 Series FPGAs Transceivers Wizard	2.4						xilinx.com	ip
7 Series FPGAs Transceivers Wizard	2.6						xilinx.com	ip
7 Series FPGAs Transceivers Wizard	2.7						xilinx.com	ip
SelectIO Interface Wizard	4.1				Production		xilinx.com	ip
Spartan-6 FPGA GTP Transceiver Wizard	1.10				Production		xilinx.com	ip
Spartan-6 FPGA GTP Transceiver Wizard	1.11				Production		xilinx.com	ip
Virtex-4 FX FPGA RocketIO Multi-Gigabit Transceiver Wizard	1.7						xilinx.com	ip
Virtex-5 FPGA RocketIO GTP Transceiver Wizard	2.1						xilinx.com	ip
Virtex-5 FPGA RocketIO GTX Transceiver Wizard	1.7						xilinx.com	ip
Virtex-6 FPGA GTH Transceiver Wizard	1.11						xilinx.com	ip
Virtex-6 FPGA GTX Transceiver Wizard	1.12						xilinx.com	ip
Soft Error Mitigation								
System Monitor								
XADC								

Search IP Catalog:

Figure 3-11: GTP new source wizard window

- Specify the core specifications


Spartan-6 FPGA GTP Transceiver Wizard

Documents

Spartan-6 FPGA GTP Transceiver Wizard

xilinx.com:ip:s6_gtpwizard:1.11

GTP Placement and Clock

Component Name: GTP

Select Tiles and Reference Clocks

	Tile Location	GTP0 REFCLK	GTP1 REFCLK
☐	GTPA1_DUAL_X1_Y0	REFCLK0 X1Y0	Use GTP0 PLL
☑	GTPA1_DUAL_X0_Y0	Use GTP1 PLL	REFCLK1 X0Y0

Transceivers selected (2 per GTPA1_DUAL)

2

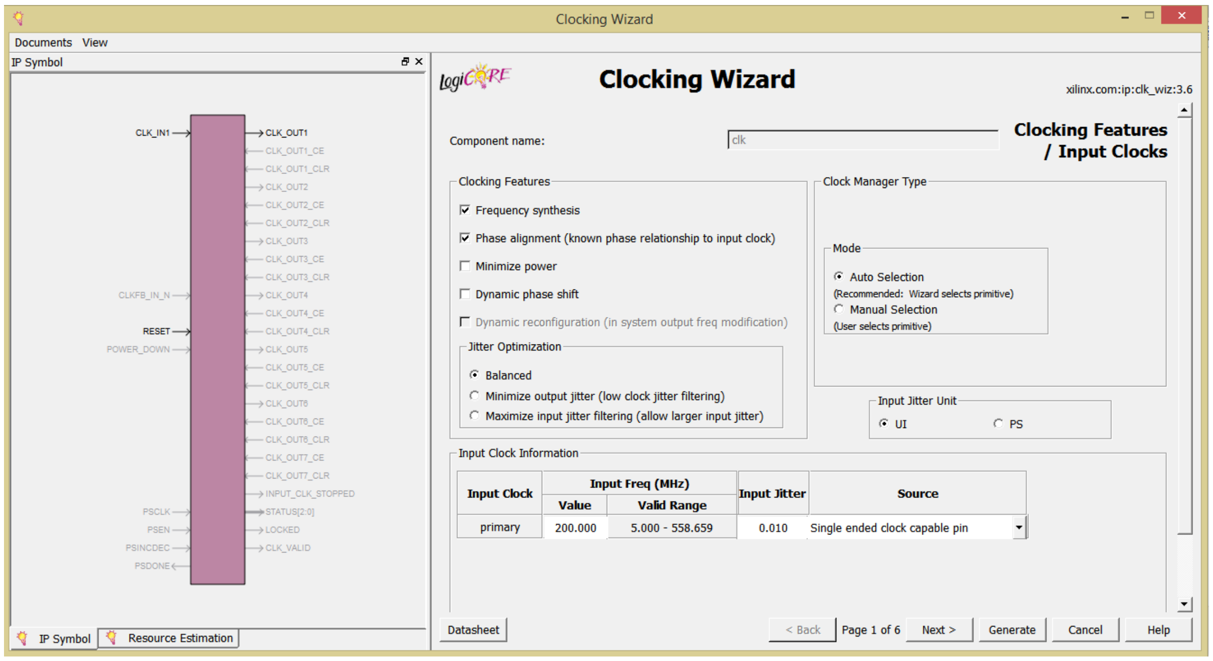
☑ Advanced Clocking

Each GTPA1_DUAL tile contains two individual GTP transceivers. Selecting a tile above enables both transceivers. If multiple tiles are selected, settings for the specified transceivers are applied to the respective transceivers in each selected tile.

Figure 3-12: GTP transceivers wizard window

We chose this clocking scheme which allows us to use the 200MHz oscillator of the FPGA to generate the 125 MHz using another IP core, clock wizard.

- Generate the Clock wizard IP core and add it the same project. It is included under FPGA Features and Design, Clocking, **Clocking Wizard V3.6**.
- Configure its input clock to be 200MHz differential capable pin and.
- Configure its output clock to be 125 MHZ which is required in the GTP core.
- The clock wizard instance will be defined as a component in the top level of the GTP codes at both the receiver and transmitter **gtp_ip_top.**



Component name: clk

Clocking Features

- [✓] Frequency synthesis
- [✓] Phase alignment (known phase relationship to input clock)
- [ ] Minimize power
- [ ] Dynamic phase shift
- [ ] Dynamic reconfiguration (in system output freq modification)

Jitter Optimization

- [✓ Balanced]
- Minimize output jitter (low clock jitter filtering)
- Maximize input jitter filtering (allow larger input jitter)

Clock Manager Type

Mode

- [✓ Auto Selection] (Recommended: Wizard selects primitive)
- Manual Selection (User selects primitive)

Input Jitter Unit: [✓ UI] PS

Input Clock Information

Input Clock	Input Freq (MHz)		Input Jitter	Source
	Value	Valid Range		
primary	200.000	5.000 - 558.659	0.010	Single ended clock capable pin

Figure 3-13: Clocking wizard window

3.3.1.2 Using Example Design

The PCI EXPRESS example consists of the following components:

- A demonstration test bench to drive the example design in simulation
- An example design providing clock signals and connecting an instance of the PCI EXPRESS wrapper with modules to drive and monitor the wrapper in hardware, including optional ChipScope™ Pro tool support.
- Scripts to synthesize and simulate the example design

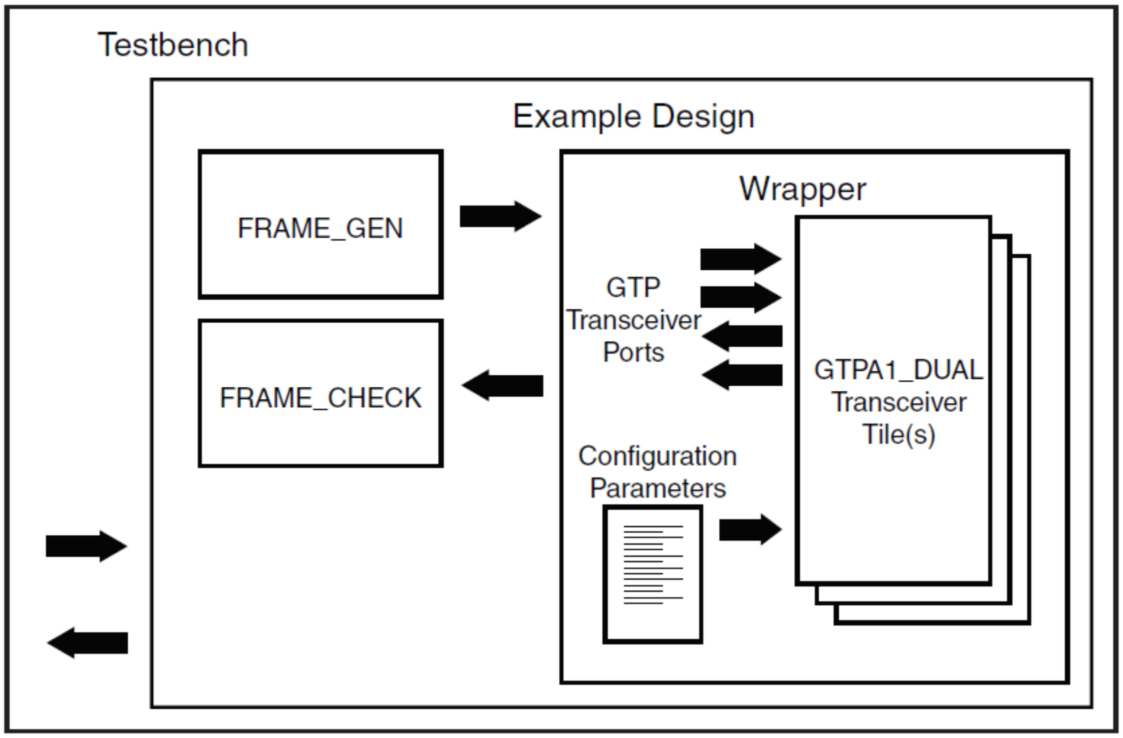


Figure 3-14: GTP example design block diagram

The GTP IP core is generated by default as a transceiver, having the functionality of both the transmitter and receiver, so in the first phase of the project while implementing the downlink channel only we need the code at the first FPGA to have the transmitter functionality so we had to remove the code lines and connections of the receiver pins, and the same procedure applies for the receiver code.

At the transmitter side a counter with automatic reset is generated to be received at the receiver side.

The counter frames was received successfully at the receiver and viewed using the chipscope, the chipscope has multiple cores that will be included in the GTP core, which assist with on-chip debugging: integrated logic analyzer (ILA), integrated bus analyzer

(IBA), and virtual input/output (VIO) low-profile software cores. These cores allow you to view internal signals and nodes in your FPGA. We only used ILA and ICON.

- ICON

The Integrated Controller (ICON) core provides the communication between the embedded ILA, IBA, and VIO cores and the computer running the ChipScope Pro Analyzer software. ICON generates one vector signal CONTROL with length equal to the number of the cores included in the main code.

- ILA

The ILA core is a customizable logic analyzer core that can be used to monitor the internal signals in your design. Because the ILA core is synchronous to the design being monitored, all design clock constraints applied to your design are also applied to the components inside the ILA core. ILA simply generates three signals, CONTROL vector INOUT signal that will be connected to the CONTROL signal of ICON, CLK signal that is connected to the clock of the main code, and TRIGGER vector signal which will be connected the signals to be viewed on the Chipscope waveform analyzer.

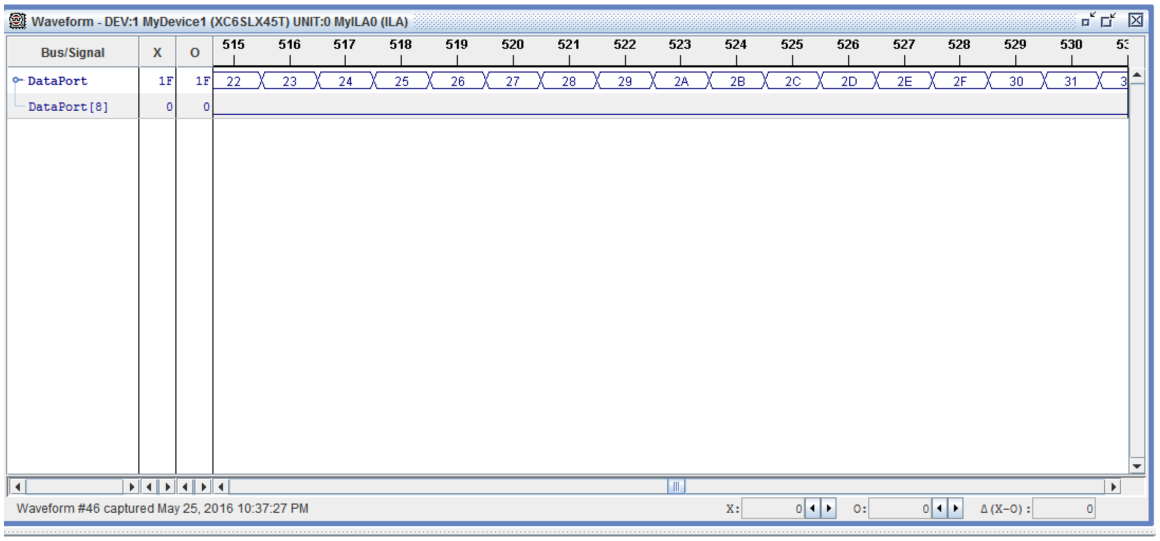

Waveform - DEV:1 MyDevice1 (XC6SLX45T) UNIT:0 MyILA0 (ILA)
Bus/Signal
X
O
515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
DataPort 1F 1F 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31
DataPort[8] 0 0
Waveform #46 captured May 25, 2016 10:37:27 PM
X: 0
O: 0
Δ (X-O) : 0

Figure 3-15: GTP test with counter on Chipscope

3.4 Top Level Design

In a simple description at this step we are having two cores or blocks the TEMAC and GTP, and it is required to get the Ethernet frames from the TEMAC core to the GTP core at the transmitter and the opposite at the receiver. The standard format of the Ethernet frame is shown below as defined by IEEE Std 802.3. The signals that carries the frame are **tx_axis_av_tdata[7:0]** for the input frames and **rx_axis_av_tdata[7:0]** for the output frames, note that output or input here refers to the Ethernet core not the other interfacing node which is known as client.

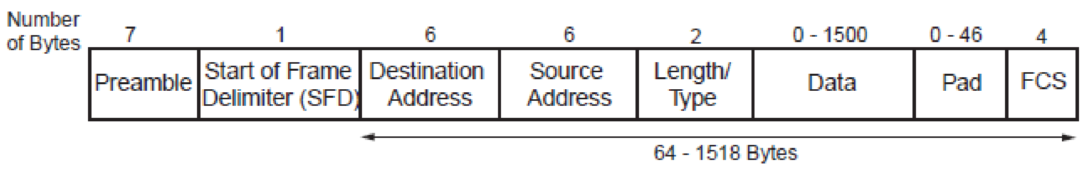


Figure 3-16: Standard Ethernet frame

- Preamble: used for synchronization and contains seven bytes with the pattern 0x55 this field is always stripped from the incoming frame, before the data is passed to the client
- Start of Frame Delimiter: marks the start of the frame and must contain the pattern 0xD5. for transmission on the physical interface, this field is automatically inserted by the Ethernet MAC. For reception, this field is always stripped from the incoming frame before the data is passed to the client.
- Destination Address: MAC address of the destination and is always retained in the receive packet data
- Source Address: MAC address of the source and is always retained in the receive packet data.
- Length/Type: When used as a length field, the value in this field represents the number of bytes in the following data field .If it contains a value of 0x8100 indicates that the frame is a VLAN frame, and a value of 0x8808 indicates a PAUSE MAC control frame which we disabled at this stage of our work.
- Data: The required Payload, The data field can vary from 0 to 1,500 bytes in length for a normal frame. The Ethernet MAC can handle jumbo frames of any length.
- Pad: vary from 0 to 46 bytes in length. This field is used to ensure that the frame length is at least 64 bytes in length (the preamble and SFD fields are not considered part of the frame for this calculation), which is required for successful CSMA/CD operation, are used in the frame check sequence calculation but are not included in the length field value.

- FCS: The value of the FCS field is calculated over the destination address, source address, length/type, data, and pad fields using a 32-bit Cyclic Redundancy Check (CRC) as defined in IEEE Std 802.3-2008 para. 3.2.8:

G(x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x1 + x0

The Data field is what we actually want to transmit or receive between the two cores and between both kits. Usually any core has some signal associated with the data transmission for flow control some of them are ignored but some signals were connected for proper operation and they were useful in the final implementation. As for the GTP core it mainly needed three signals beside the data which are CLK, RESET and Valid_Data which can be used as a flow control signal to discard the invalid data. The Ethernet core has around 30 signals and some of them are even vectors, but we will illustrate only the ones that were critical in our flow.

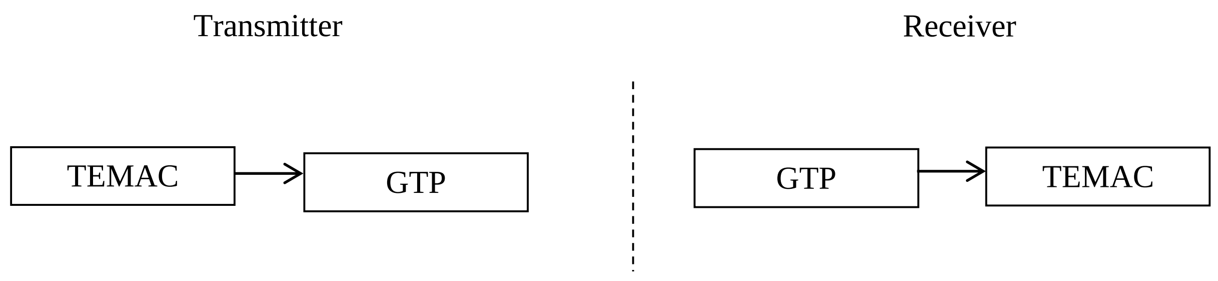


Figure 3-17: Downlink cores connections

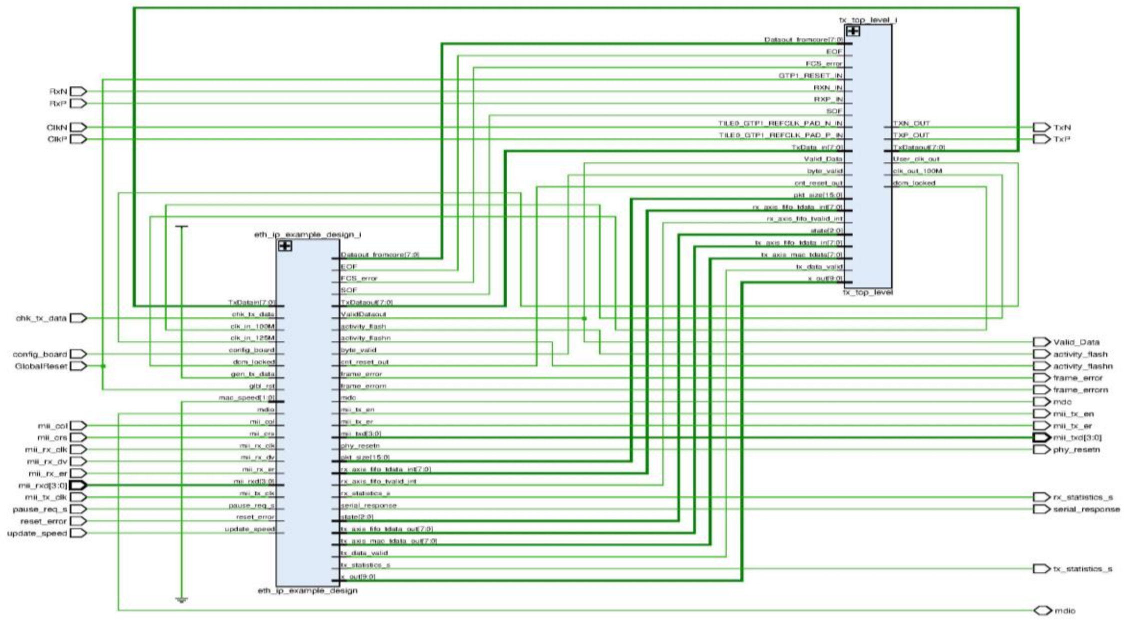


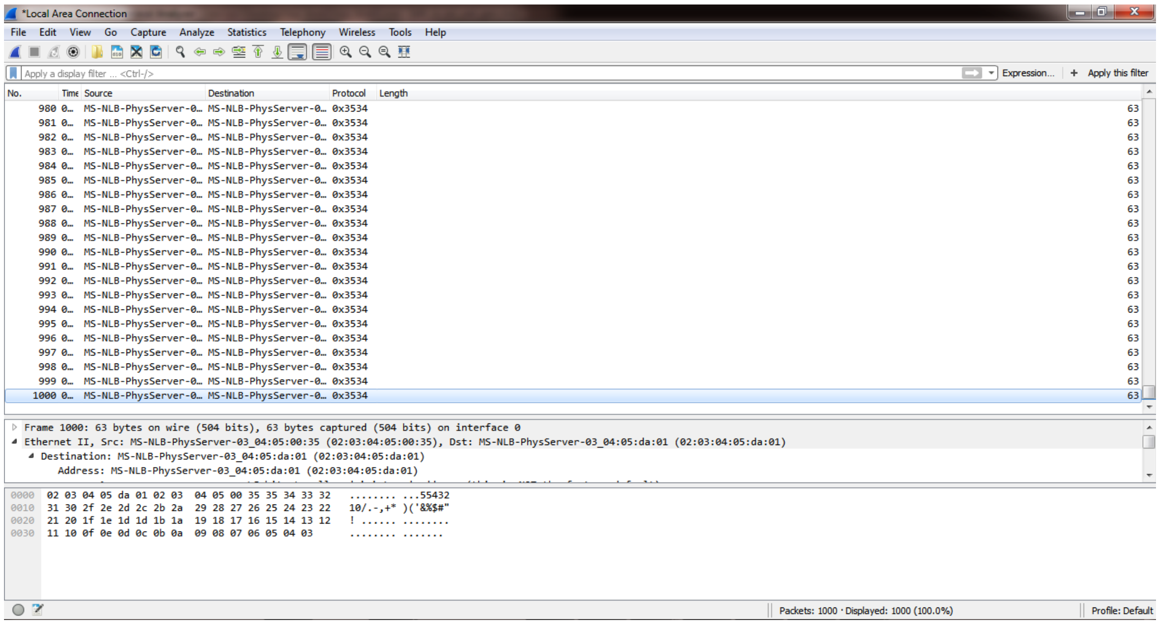
Figure 3-18: Top Level RTL Schematic

Firstly, whole system must use the same clock for synchronization, so CLK signal was input to both cores also the RESET of both cores were connected. At the transmitter side the **rx_axis_av_tdata[7:0]** signal of the TEMAC was connected to **Tx_DATA** of the GTP and the data was delivered successfully using frames generated by **SoftPerfect Network Protocol Analyzer** without too much complications unlike the receiver side.

The main problem aroused at the receiver side was how to detect the start of frame generated by **SoftPerfect Network Protocol Analyzer** while the TEMAC core directly strips both the preamble and the SFD. Before this point the frames were coming to TEMAC from the frame generator so the whole bit stream was a sequence of frames and with lengths that are known to the core so it was easy for it to detect the start of frame unlike the case of randomly generated frames which will be the generic case of the final system.

The example design interprets the frame based on a Finite State Machine, so we worked on understanding the FSM and trying to modify over it till the core interprets the frames generated by **SoftPerfect Network Protocol Analyzer.** We used the code of the frame generator to insert the random frames in order to make use of the signals generated by the frame generator instead of writing a code from scratch so that we can guarantee its compatibility.

First step in solving this issue is that we made the destination address fixed , since after stripping the preamble and the SFD the destination address field becomes the first field in the received frame and check on its first byte thus the core can step into the state of HEADER in the FSM .To move from one state to another this takes one clock cycle which means that two cycles will be lost one for checking on the first byte of the fixed destination address and the second one will be lost to move to the next state. The length of frame was also made fixed to make the transition from the state of data to the next state after a fixed number of counts.



```
No.    Time Source                    Destination               Protocol  Length
   980 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   981 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   982 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   983 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   984 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   985 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   986 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   987 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   988 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   989 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   990 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   991 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   992 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   993 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   994 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   995 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   996 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   997 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   998 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
   999 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63
  1000 0…   MS-NLB-PhysServer-0…      MS-NLB-PhysServer-0…      0x3534        63

▷ Frame 1000: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface 0
◢ Ethernet II, Src: MS-NLB-PhysServer-03_04:05:00:35 (02:03:04:05:00:35), Dst: MS-NLB-PhysServer-03_04:05:da:01 (02:03:04:05:da:01)
  ◢ Destination: MS-NLB-PhysServer-03_04:05:da:01 (02:03:04:05:da:01)
      Address: MS-NLB-PhysServer-03_04:05:da:01 (02:03:04:05:da:01)

0000  02 03 04 05 da 01 02 03  04 05 00 35 35 34 33 32   ........ ...55432
0010  31 30 2f 2e 2d 2c 2b 2a  29 28 27 26 25 24 23 22   10/.-,+* )('&%$#"
0020  21 20 1f 1e 1d 1d 1b 1a  19 18 17 16 15 14 13 12   ! ...... ........
0030  11 10 0f 0e 0d 0c 0b 0a  09 08 07 06 05 04 03      ........ .......

Packets: 1000 · Displayed: 1000 (100.0%)        Profile: Default
```

Figure 3-19: Receiving frames without the first two bytes

Consequently we have three problems the fixed destination address, the two bytes that are missed from the frame and the fixed frame length. To solve the first problem, we needed to detect the start of frame two bytes earlier, checking on the block diagram and internal signals of the TEMAC core we found that it contains a packet **FIFO** that holds the frames before it reaches the core by some cycles, This signal can be exploited for the purpose of early detection.

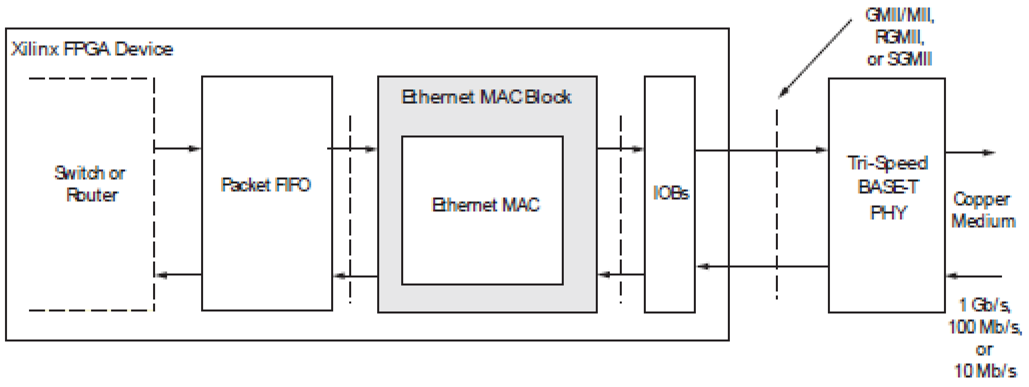


Figure 3-20: TEMAC connection in applications

Rx_statistics [27:0] is another vector signal that gives 28 bits according to some statistics in the coming frame. Bit 22 is called **BYTE_VALID** asserted if a MAC frame byte (destination address to FCS inclusive) is in the process of being received. This is valid on every clock cycle.

Keeping the frame length fixed and observing the number of cycles between the **BYTE_VALID** positive edge and **FIFO** signal carrying the frame they are found constant. So we created a **COUNTER** signal that resets after the constant number of counts so that the core interprets it is the start of the frame at the counter reset.

Allowing the frame length to be variable **BYTE_VALID** keeps high for variable duration depending on the frame length consequently the number of counts will not be constant. But the duration between negative edge of **BYTE_VALID** and **FIFO** signal carrying the frame is constant regardless of the frame length. To detect the negative edge, using a CLK'event gave an error in the code synthesis so we made another FSM depending on the negative edge of **BYTE_VALID**.

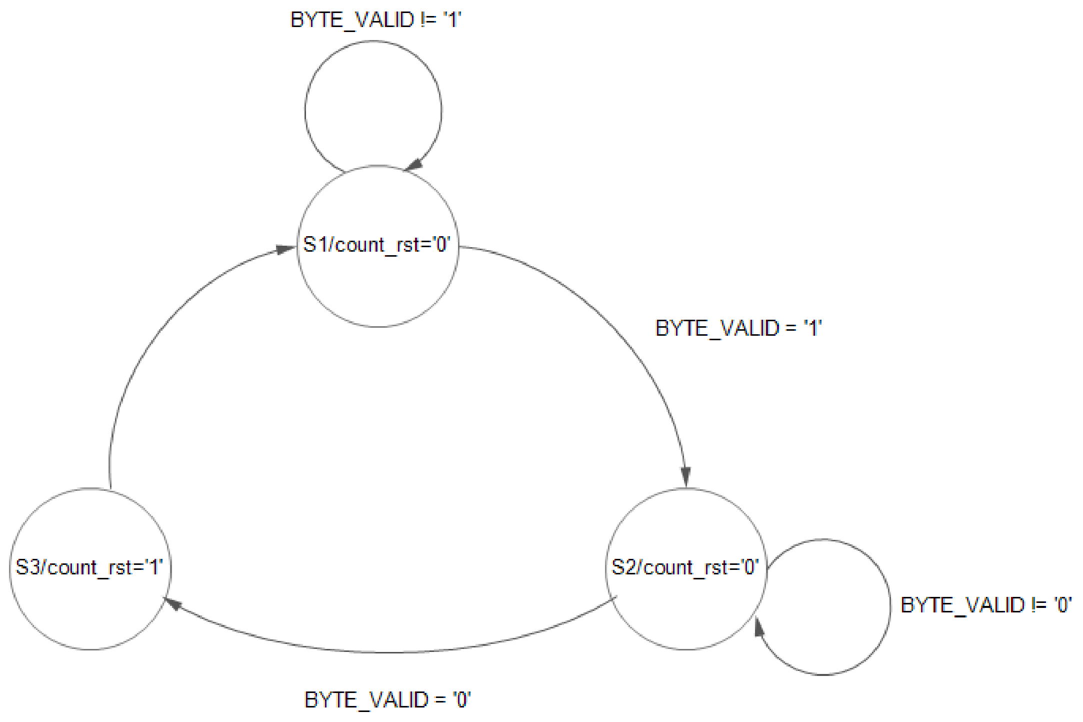
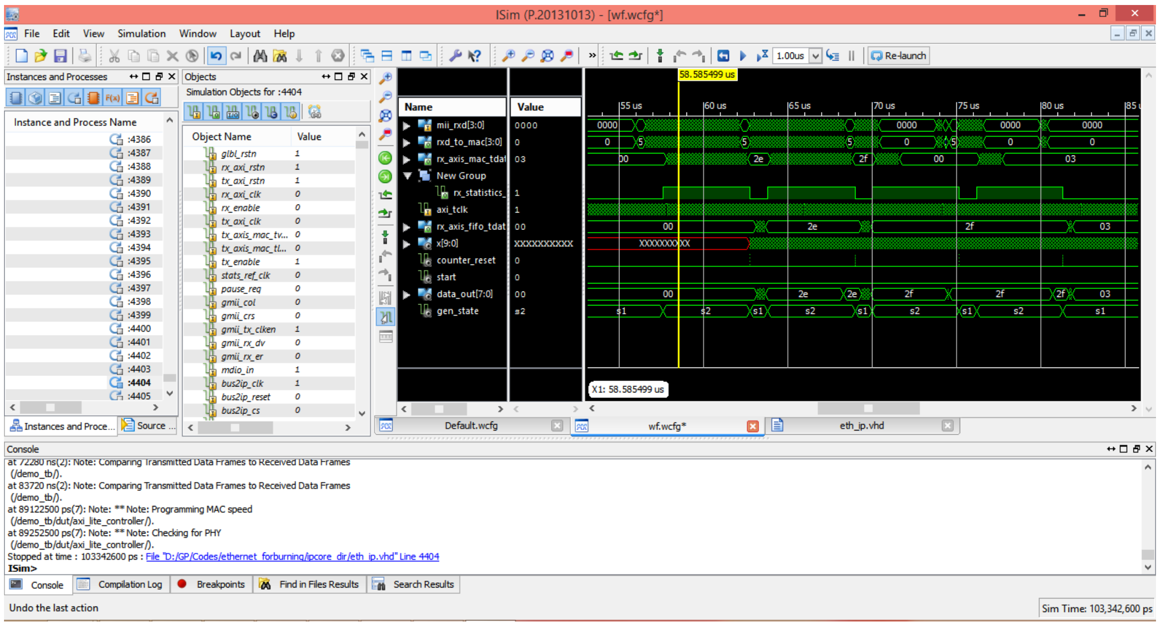


Figure 3-21: FSM for negative edge of BYTE_VALID dependency



```
at 72280 ns(2): Note: Comparing Transmitted Data Frames to Received Data Frames
 (/demo_tb/).
at 83720 ns(2): Note: Comparing Transmitted Data Frames to Received Data Frames
 (/demo_tb/).
at 89122500 ps(7): Note: ** Note: Programming MAC speed
 (/demo_tb/dut/axi_lite_controller/).
at 89252500 ps(7): Note: ** Note: Checking for PHY
 (/demo_tb/dut/axi_lite_controller/).
Stopped at time : 103342600 ps : File "D:/GP/Codes/ethernet_forburning/ipcore_dir/eth_ip.vhd" Line 4404
ISim>
```

Figure 3-22: Counter reset on negative edge of BYTE_VALID

Last observation was for the frames with error, it is noticed that **BYTE_VALID** is high whenever it detects an Ethernet frame even if it did not pass the Frame Check Sequence test although **FIFO** will not pass it to the core. Recalling the statistics vector, Bit 2 is called **FCS_ERROR** which is asserted if the previous frame received was correctly aligned but had an incorrect FCS value or the MAC detected error codes during frame reception.

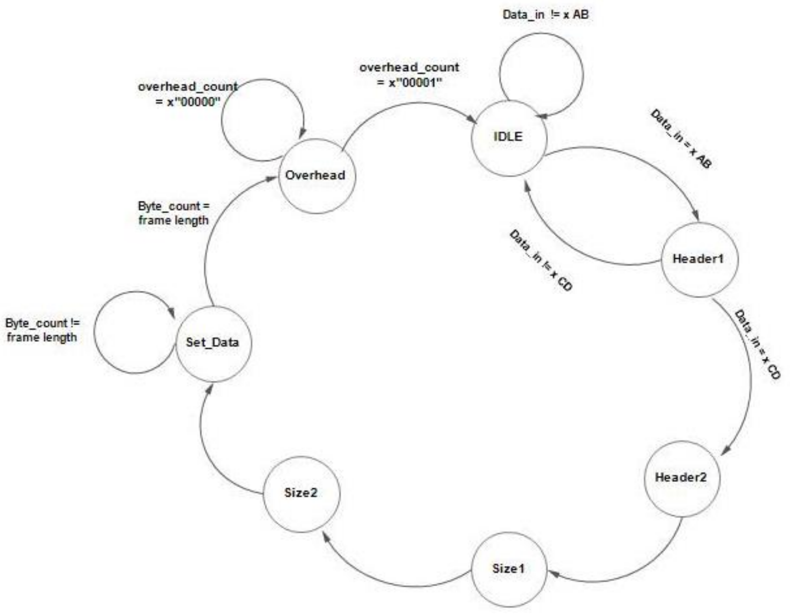
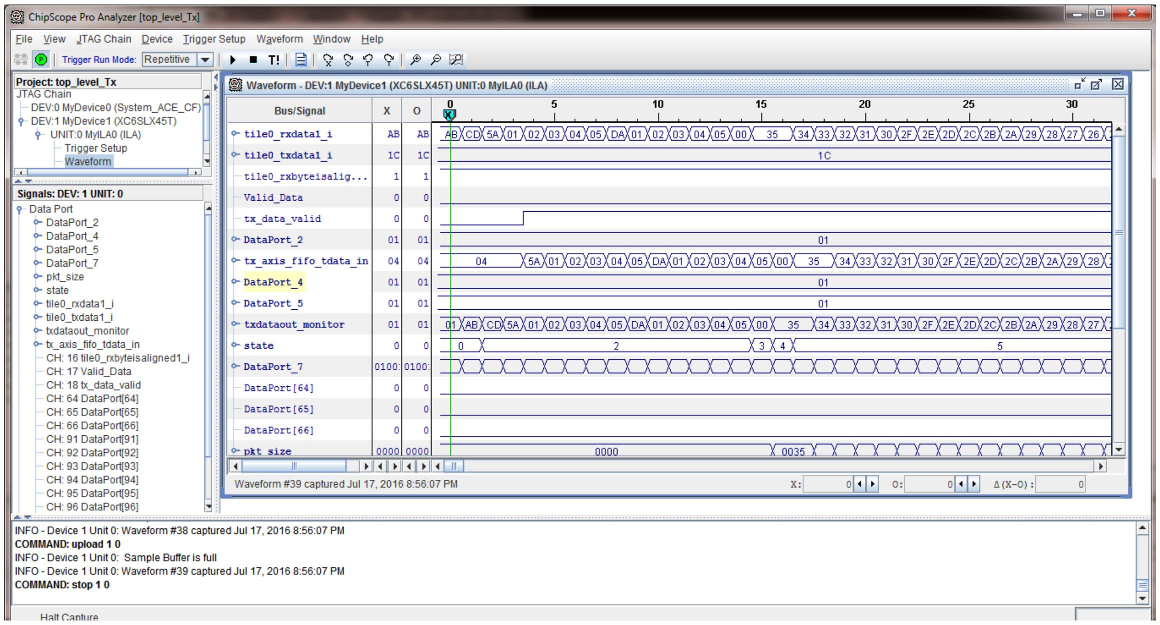


Figure 3-23: FSM for Ethernet frames reception

We used the chipscope to check the states are working properly.


ChipScope Pro Analyzer [top_level_Tx]
File View JTAG Chain Device Trigger Setup Waveform Window Help
Trigger Run Mode: Repetitive
Project: top_level_Tx
JTAG Chain
DEV:0 MyDevice0 (System_ACE_CF)
DEV:1 MyDevice1 (XC6SLX45T)
UNIT:0 MyILA0 (ILA)
Trigger Setup
Waveform
Signals: DEV:1 UNIT:0
Waveform - DEV:1 MyDevice1 (XC6SLX45T) UNIT:0 MyILA0 (ILA)
Bus/Signal
X
O
Halt Capture

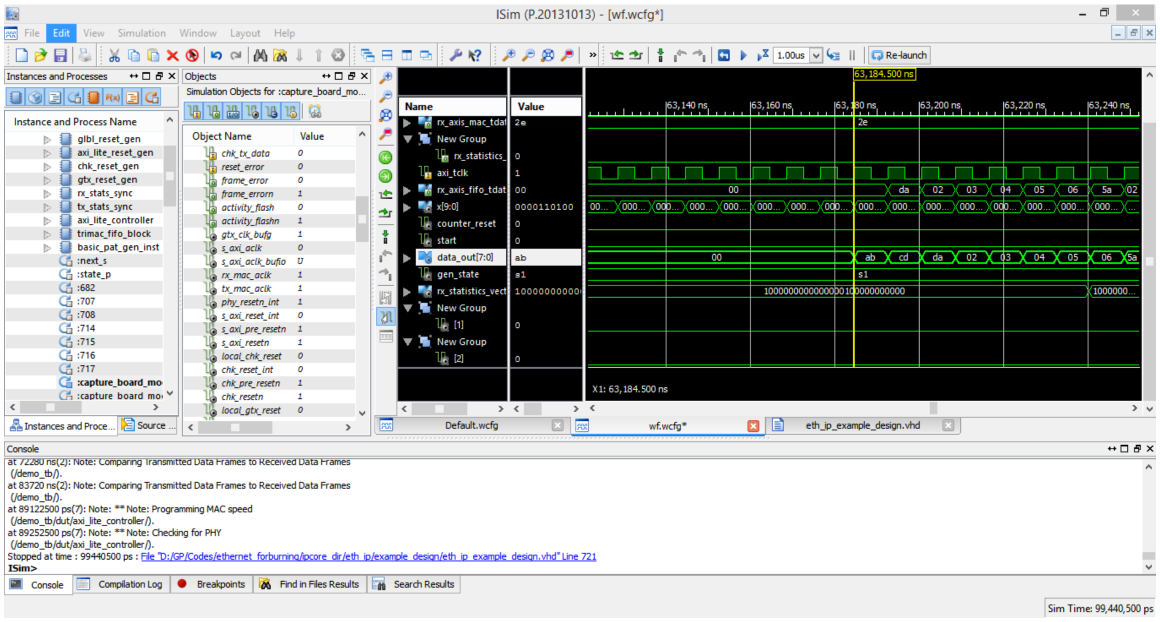
```
Bus/Signal              X     O
tile0_rxdata1_i         AB    AB
tile0_txdata1_i         1C    1C
tile0_rxbyteisalig...   1     1
Valid_Data              0     0
tx_data_valid           0     0
DataPort_2              01    01
tx_axis_fifo_tdata_in   04    04
DataPort_4              01    01
DataPort_5              01    01
txdataout_monitor       01    01
state                   0     0
DataPort_7              0100  0100
DataPort[64]            0     0
DataPort[65]            0     0
DataPort[66]            0     0
pkt_size                0000  0000
```

Waveform #39 captured Jul 17, 2016 8:56:07 PM

```
INFO - Device 1 Unit 0: Waveform #38 captured Jul 17, 2016 8:56:07 PM
COMMAND: upload 1 0
INFO - Device 1 Unit 0:  Sample Buffer is full
INFO - Device 1 Unit 0: Waveform #39 captured Jul 17, 2016 8:56:07 PM
COMMAND: stop 1 0
```

Figure 3-24: States on Chipscope

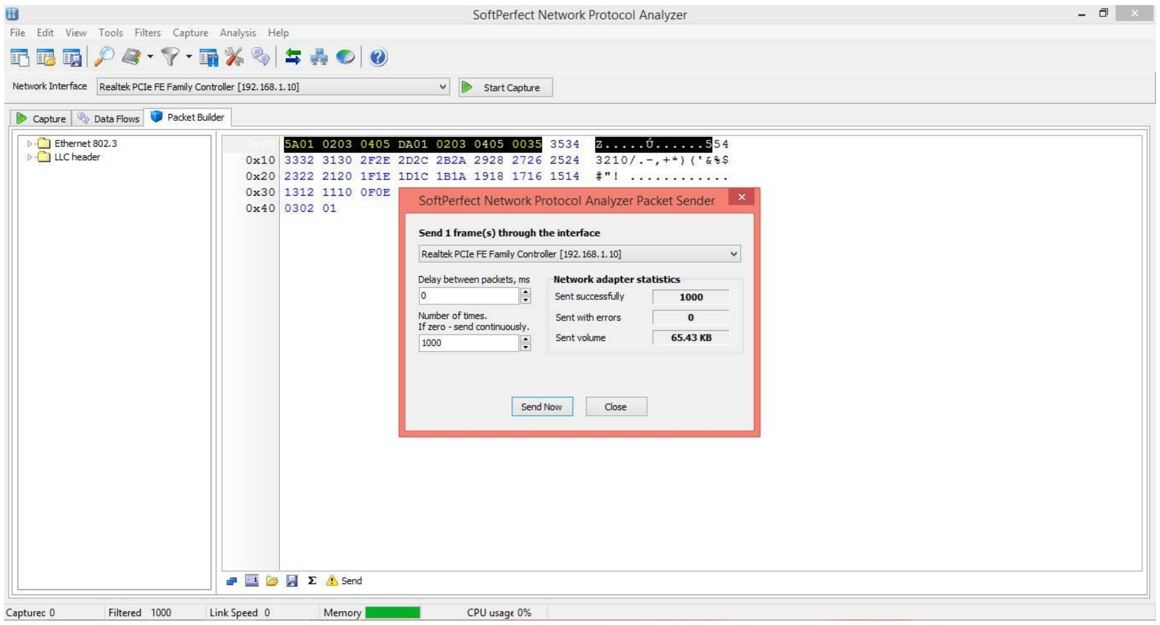
And we checked the header is being sent properly in the simulation.



```
at 72280 ns(2): Note: Comparing Transmitted Data Frames to Received Data Frames
 (/demo_tb/).
at 83720 ns(2): Note: Comparing Transmitted Data Frames to Received Data Frames
 (/demo_tb/).
at 89122500 ps(7): Note: ** Note: Programming MAC speed
 (/demo_tb/dut/axi_lite_controller/).
at 89252500 ps(7): Note: ** Note: Checking for PHY
 (/demo_tb/dut/axi_lite_controller/).
Stopped at time : 99440500 ps : File "D:/GP/Codes/ethernet_forburning/ipcore_dir/eth_ip/example_design/eth_ip_example_design.vhd" Line 721
ISim>
```

Figure 3-25: Adding header after counter reset

Finally we sent 1000 frames and received them successfully with the first two bytes.


SoftPerfect Network Protocol Analyzer
File Edit View Tools Filters Capture Analysis Help
Start Capture
Capture
Data Flows
Packet Builder
Ethernet 802.3
LLC header
Send
Send Now
Close

Network Interface: Realtek PCIe FE Family Controller [192.168.1.10]

```
0x00 5A01 0203 0405 DA01 0203 0405 0035 3534  Z.....Ú......554
0x10 3332 3130 2F2E 2D2C 2B2A 2928 2726 2524  3210/.-,+*)('&%$
0x20 2322 2120 1F1E 1D1C 1B1A 1918 1716 1514  #"! ............
0x30 1312 1110 0F0E
0x40 0302 01
```

SoftPerfect Network Protocol Analyzer Packet Sender

Send 1 frame(s) through the interface

Realtek PCIe FE Family Controller [192.168.1.10]

Delay between packets, ms: 0

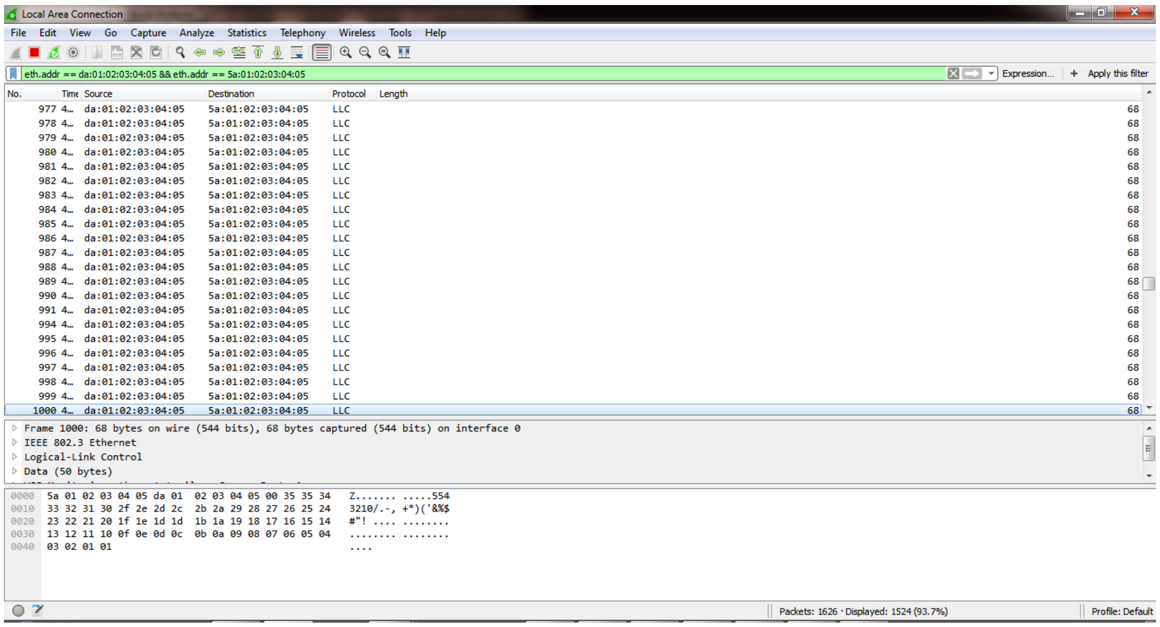
Number of times. If zero - send continuously.: 1000

Network adapter statistics

Sent successfully	1000
Sent with errors	0
Sent volume	65.43 KB

Capturec 0 | Filtered 1000 | Link Speed 0 | Memory | CPU usage 0%

Figure 3-26: Sending 1000 frames


Local Area Connection
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
Apply this filter

eth.addr == da:01:02:03:04:05 && eth.addr == 5a:01:02:03:04:05

```
No.   Time Source             Destination        Protocol Length
 977 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 978 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 979 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 980 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 981 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 982 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 983 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 984 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 985 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 986 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 987 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 988 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 989 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 990 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 991 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 994 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 995 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 996 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 997 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 998 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
 999 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68
1000 4…  da:01:02:03:04:05  5a:01:02:03:04:05  LLC       68

Frame 1000: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
IEEE 802.3 Ethernet
Logical-Link Control
Data (50 bytes)

0000  5a 01 02 03 04 05 da 01  02 03 04 05 00 35 35 34   Z....... .....554
0010  33 32 31 30 2f 2e 2d 2c  2b 2a 29 28 27 26 25 24   3210/.-, +*)('&%$
0020  23 22 21 20 1f 1e 1d 1d  1b 1a 19 18 17 16 15 14   #"! .... ........
0030  13 12 11 10 0f 0e 0d 0c  0b 0a 09 08 07 06 05 04   ........ ........
0040  03 02 01 01                                        ....
```

Packets: 1626 · Displayed: 1524 (93.7%) | Profile: Default

Figure 3-27: Receiving the 1000 frames after resolving the two bytes problem

3.5 UART

During the last period of the project we tried working on another path so that we can present the hardware in a working state. Besides, the UART can have acceptable data rates even if it cannot reach the rates of GTP which would be compatible to use in places that already have poor infrastructure and cannot support the very high data rates.

First we wrote the UART transmitter code that receives input data (byte by byte) in Data register with clock (4.8 MHZ) and retransmit it (bit by bit) through shift register in order to have parallel to serial transmission but with a clock that is 8 times faster than that used to receive the data (38.4 MHZ), i.e the transmitter receives the data with clock (4.8 MHZ) and transmits the bits with clock (38.4 MHZ).

The receiver code makes the opposite direction that receives serially (bit by bit) and goes with the following sequence:

- Start of the frame is detected.
- The following 8 bits are received in a shift register to form “receiving byte”.
- When the end of frame is detected the “received byte” is delivered to the output bus of the receiver.

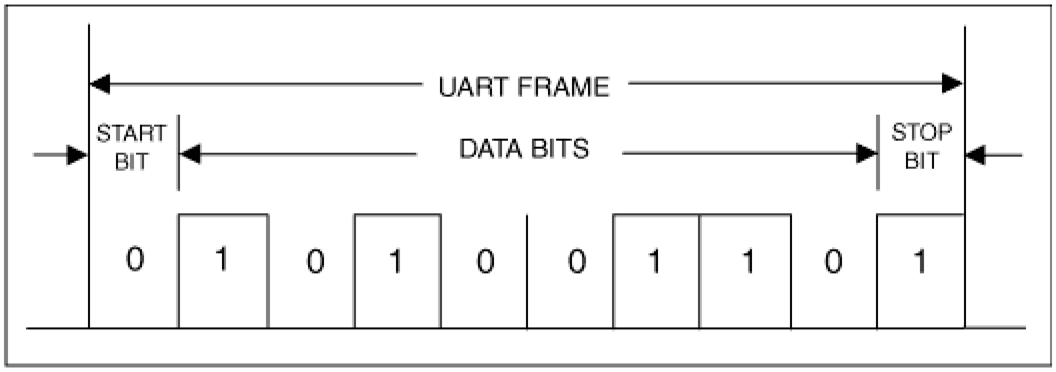


Figure 3-28: UART complete frame

So, the transmitter alternates between two states (IDLE, Sending) through two signals (Data_ready and count). Data_ready is set to 1 if the transmitter receives new byte to be sent while the count signal is responsible for counting the transmitted bits so when it reaches 8 this indicates that the whole byte is sent so it returns to the idle state again.

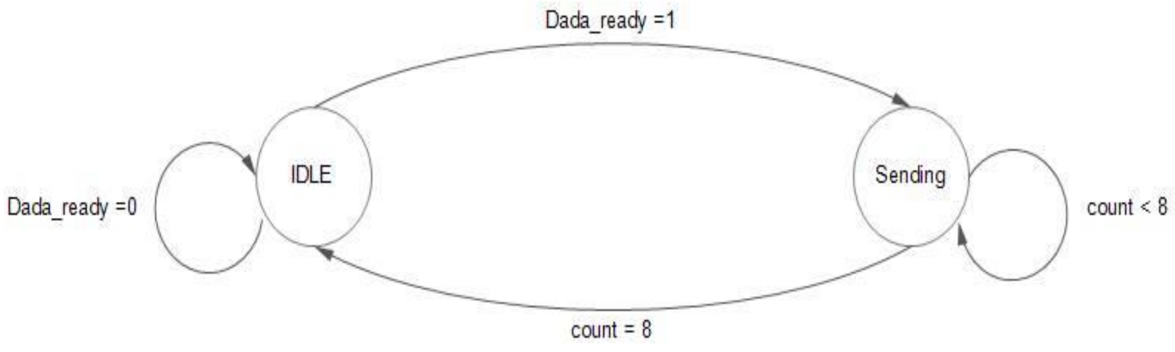


Figure 3-29: Transmitter state diagram

Also the receiver has only two alternating state (IDLE and Receiving) and they are determined according to (start_determined and end_determined) signals. The first indicates the start bit –start of the frame - so it moves the receiver to the receiving mode while the later indicates the end bit –end of frame- that returns the receiver back to the idle state.

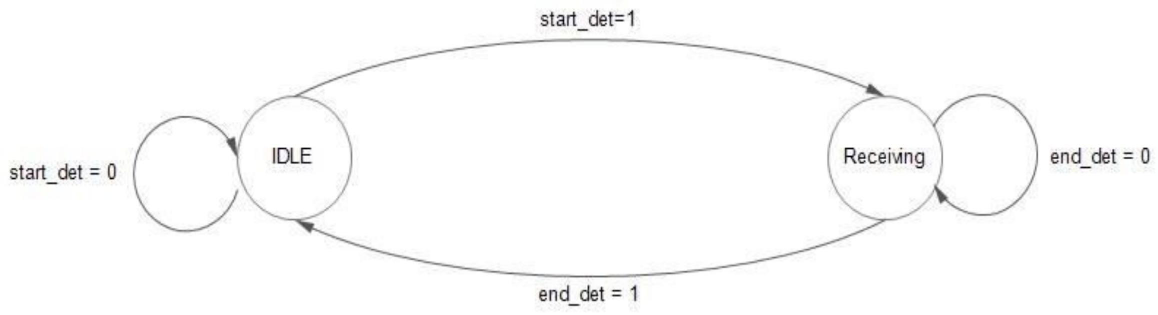
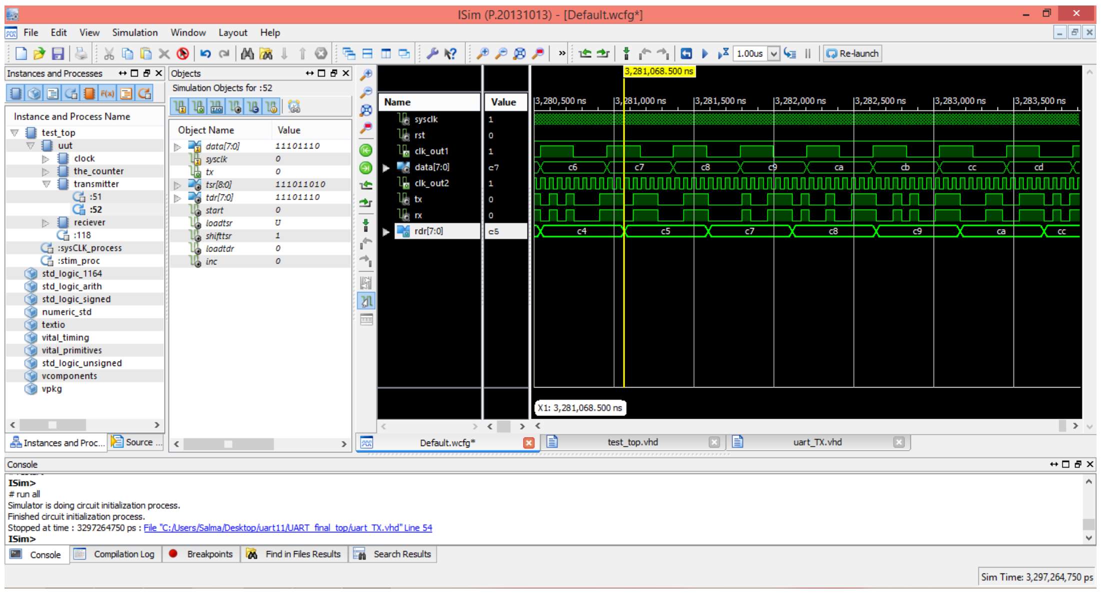


Figure 3-30: Receiver state diagram

In order to test the UART separately we designed the top level module that has the transmitter and the receiver as components, also it has a counter that delivers the data successively to the transmitter and the following result was brought from the software test.



Simulation Objects for :52

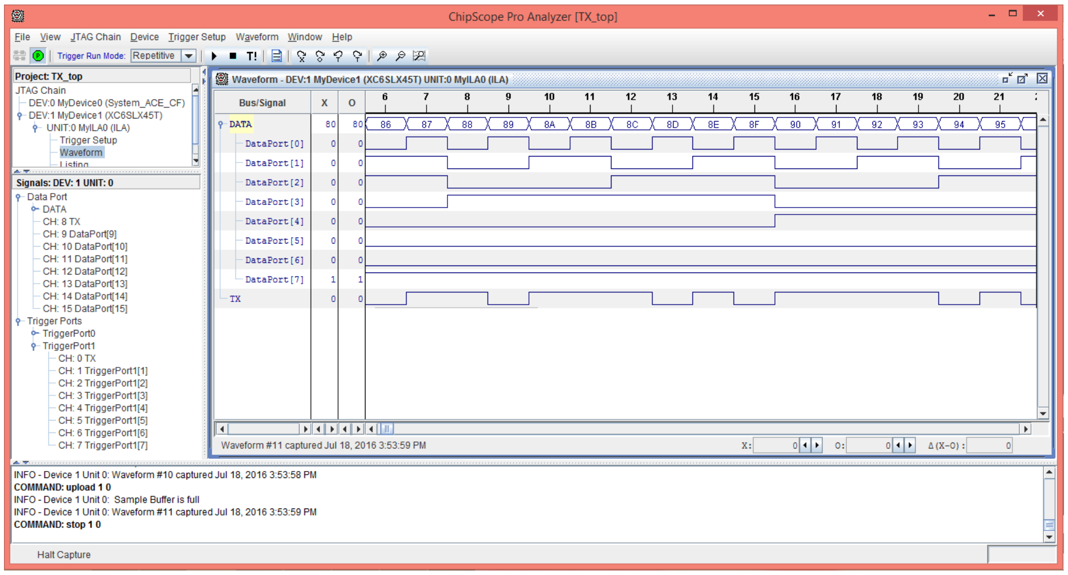
Object Name	Value
data[7:0]	11101110
sysclk	0
tx	0
tsr[8:0]	111011010
tdr[7:0]	11101110
start	0
loadtsr	U
shifttsr	1
loadtdr	0
inc	0

Name	Value
sysclk	1
rst	0
clk_out1	1
data[7:0]	c7
clk_out2	1
tx	0
rx	0
rdr[7:0]	c5

```
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Stopped at time : 3297264750 ps : File "C:/Users/Salma/Desktop/uart11/UART_final_top/uart_TX.vhd" Line 54
ISim>
```

Figure 3-31: software simulation of UART

After connecting the two FPGAs back to back with jumpers between general purposes pins, we have tested the performance through Chipscope and the following result was got.



```
INFO - Device 1 Unit 0: Waveform #10 captured Jul 18, 2016 3:53:58 PM
COMMAND: upload 1 0
INFO - Device 1 Unit 0:  Sample Buffer is full
INFO - Device 1 Unit 0: Waveform #11 captured Jul 18, 2016 3:53:59 PM
COMMAND: stop 1 0
```

Figure 3-32: transmitter circuit hardware test.

Finally we integrated the UART code with the Ethernet code to form the final code as an alternative path that will use common jumpers rather than SMA connectors that supports very high frequencies. Ethernet and UART codes interfaced through some signals that is shown in the following block diagram

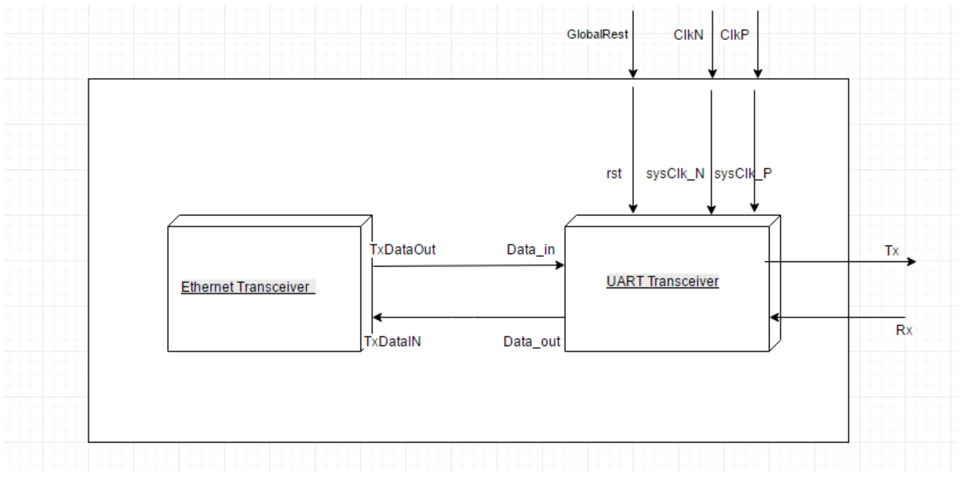
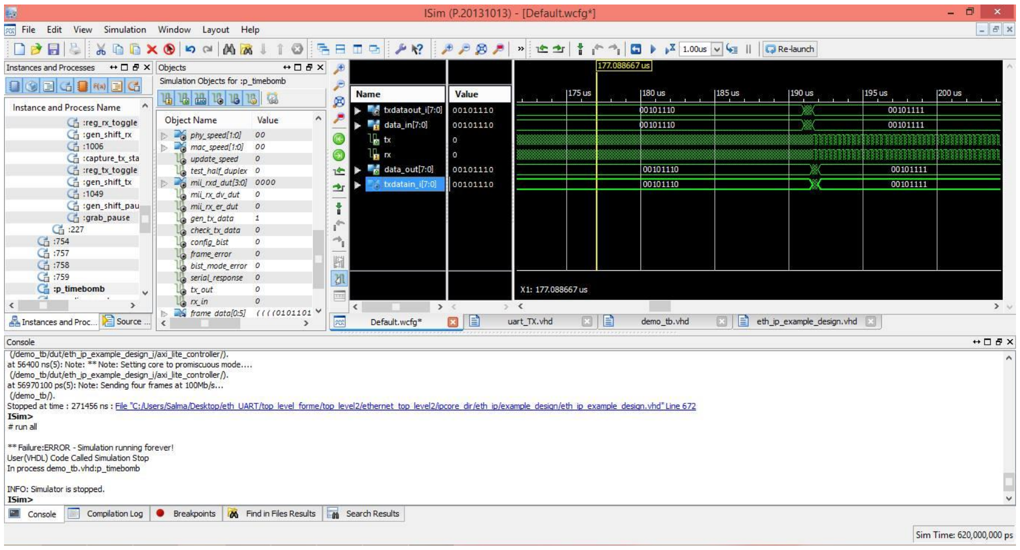


Figure 3-33: Block diagram of the Ethernet and UART

We had tested the integrated code by the software simulator (ISIM) to guarantee that the packets are delivered successfully.

The following result shows the delivery of the packets in order.



Object Name	Value
phy_speed[1:0]	00
mac_speed[1:0]	00
update_speed	0
test_half_duplex	0
mii_rxd_dut[3:0]	0000
mii_rx_dv_dut	0
mii_rx_er_dut	0
gen_tx_data	1
check_tx_data	0
config_bist	0
frame_error	0
bist_mode_error	0
serial_response	0
tx_out	0
rx_in	0
frame data[0:5]	((((0101101

Name	Value
txdataout_i[7:0]	00101110
data_in[7:0]	00101110
tx	0
rx	0
data_out[7:0]	00101110
txdatain_i[7:0]	00101110

```
 (/demo_tb/dut/eth_ip_example_design_i/axi_lite_controller/).
at 56400 ns(5): Note: ** Note: Setting core to promiscuous mode....
 (/demo_tb/dut/eth_ip_example_design_i/axi_lite_controller/).
at 56970100 ps(5): Note: Sending four frames at 100Mb/s...
 (/demo_tb/).
Stopped at time : 271456 ns : File "C:/Users/Salma/Desktop/eth_UART/top_level_forme/top_level2/ethernet_top_level2/ipcore_dir/eth_ip/example_design/eth_ip_example_design.vhd" Line 672
ISim>
# run all

** Failure:ERROR - Simulation running forever!
User(VHDL) Code Called Simulation Stop
In process demo_tb.vhd:p_timebomb

INFO: Simulator is stopped.
ISim>
```

Figure 3-34: Software Simulation of Ethernet and UART

Chapter 4: Hardware Approach

Recalling the block diagram of our system the two blocks that require external hardware are the transmitter and receiver circuit. We already had the hardware on last year's team circuits implemented by lumped components on breadboards, both circuits are designed using well known analog stages consisting of Operational Amplifiers that were simulated and tested inside the lab using NI instruments kit that has a function generator and an oscilloscope.

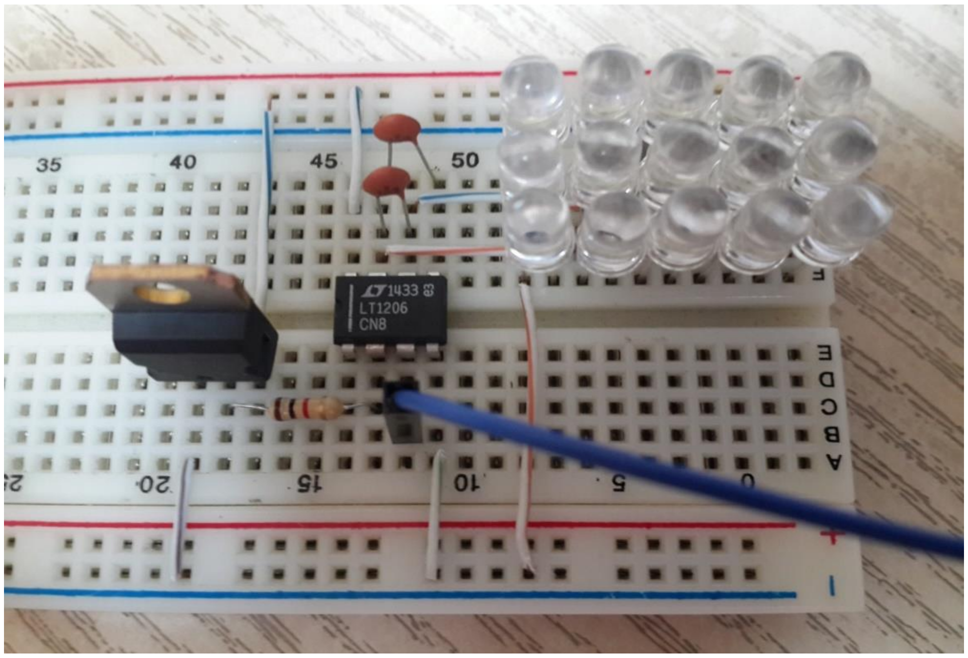


Figure 4-1: First phase transmitter circuit

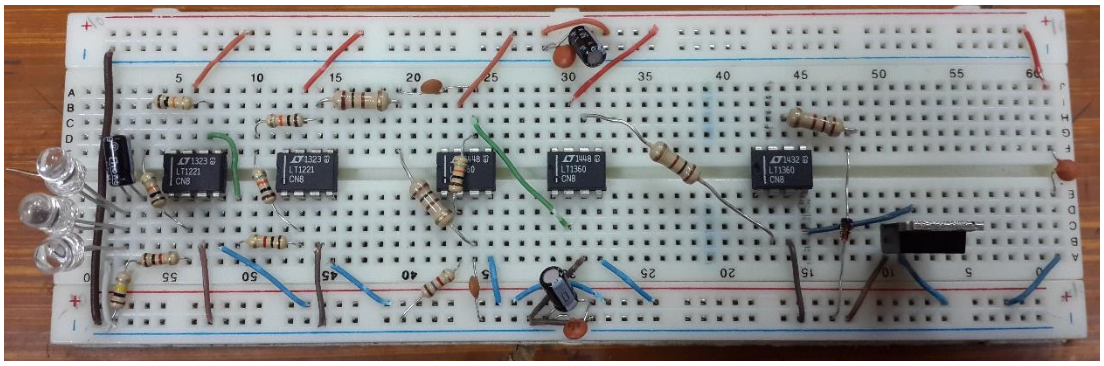


Figure 4-2: First phase receiver circuit

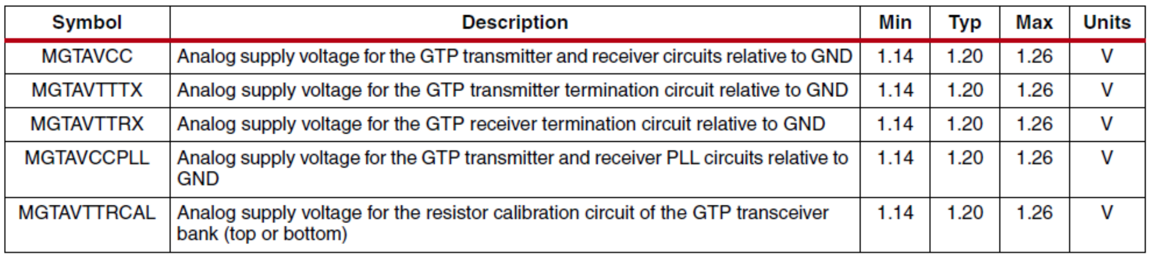
4.1 Transmitter Circuit

The transmitter circuit simply gets a random input of ones and zeros output from the Spartan-6 kit that are specified by 3.3v for the high level and -0.7v for the low level according to the USB-to-UART module. This voltage is required to lit up an array of parallel LEDs that will typically require higher current level that cannot be drawn directly from the kit so it has a single stage Op-Amp to amplify the current. We used the same design initially.

We begin looking for already made driver with LEDs but most of them were mounted on a kit together with a microcontroller. While drivers containing SMA sockets were mostly implemented with wireless antennas for short range communications like ZigBee and Bluetooth.

The first modification was to use the SMA ports and connector instead of the normal jumpers. We had a problem that the SMA ports are not found on the usual electronic analog circuit simulators that we used like Proteus and Multisim. They were only found on CST (Computer Simulation Technology simulators that offers accurate, efficient computational solutions for electromagnetic design) and analysis, on the other hand does not contain the available lumped components as they are used mainly to simulate high frequency components as transmission lines and wave guides, so we searched for other open source simulators that can be used to simulate both but found none. Finally through some search we concluded that the SMA can be simulated as 50 ohm resistor in series with the input line which represents its equivalent impedance.

The Second problem that came along is the voltage levels specified by the GTP transceivers differ from the levels of the UART module it has a maximum input voltage of 1.32v. and minimum of -0.5, by measuring the actual operating voltages using a normal voltmeter with its probes placed on the pins of the SMA connector after burning a code sending only ones then another sending only zeros , they were found even less than the stated voltages around 0.8v for the ones and 0v for the zeros.



Symbol	Description	Min	Typ	Max	Units
MGTAVCC	Analog supply voltage for the GTP transmitter and receiver circuits relative to GND	1.14	1.20	1.26	V
MGTAVTTTX	Analog supply voltage for the GTP transmitter termination circuit relative to GND	1.14	1.20	1.26	V
MGTAVTTRX	Analog supply voltage for the GTP receiver termination circuit relative to GND	1.14	1.20	1.26	V
MGTAVCCPLL	Analog supply voltage for the GTP transmitter and receiver PLL circuits relative to GND	1.14	1.20	1.26	V
MGTAVTTRCAL	Analog supply voltage for the resistor calibration circuit of the GTP transceiver bank (top or bottom)	1.14	1.20	1.26	V

Figure 4-3: GTP transceivers voltage levels

Firstly we started by setting the old hardware in the same way to ensure it is functioning well, unfortunately it was unstable but we succeeded in receiving some characters, the main obvious source for the instability is the breadboards and the connections using copper wires the effect of parasitic capacitance which cannot be neglected at high frequencies.

The instability of the breadboard and the high noise affecting the performance that will typically increase when the frequencies increase, as well as the inability to mount SMA ports on the breadboard are the main reasons for implementing the design on a PCB. We implemented the designs directly after ensuring that the old hardware is functioning and that the simulations are giving the required output for the 16 parallel LEDs.

Unfortunately that was before discovering the second problem of the voltage levels since it is not stated in the hardware user guide of Spartan-6 but when the PCB was not functioning upon connecting the SMA with random data we started measuring it and searching for another documents and found the information about the voltage levels output and input to the kit from any pin in DC and switching characteristics. The steps followed in this part were:

- Simulating the design on CADsoft eagle for printing then on Proteus.

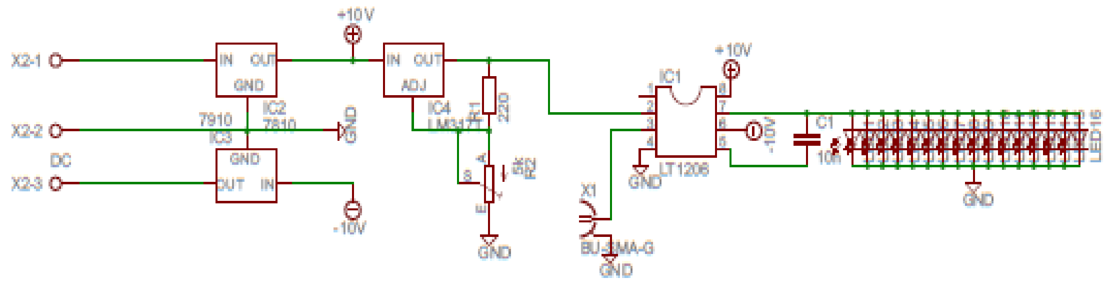


Figure 4-4: Transmitter circuit design on CADsoft

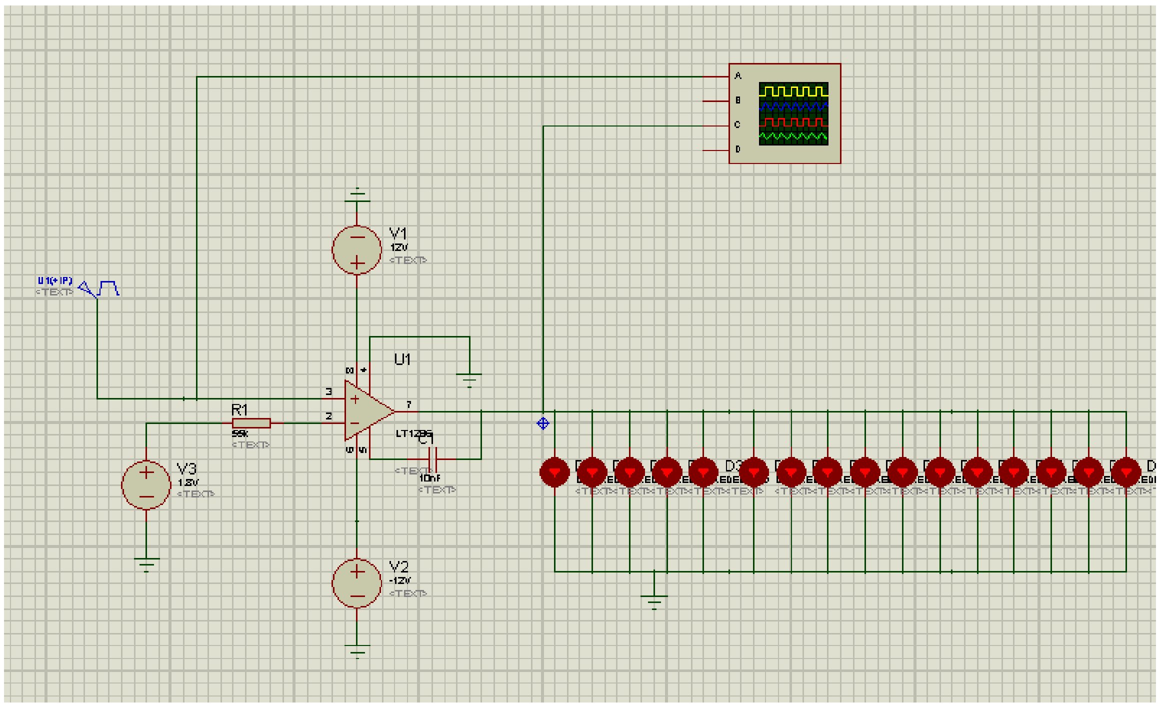


Figure 4-5: Transmitter circuit simulation on Proteus

The yellow represents the input 3.3-0 v and the pink represents the output voltage of the Op-Amp.

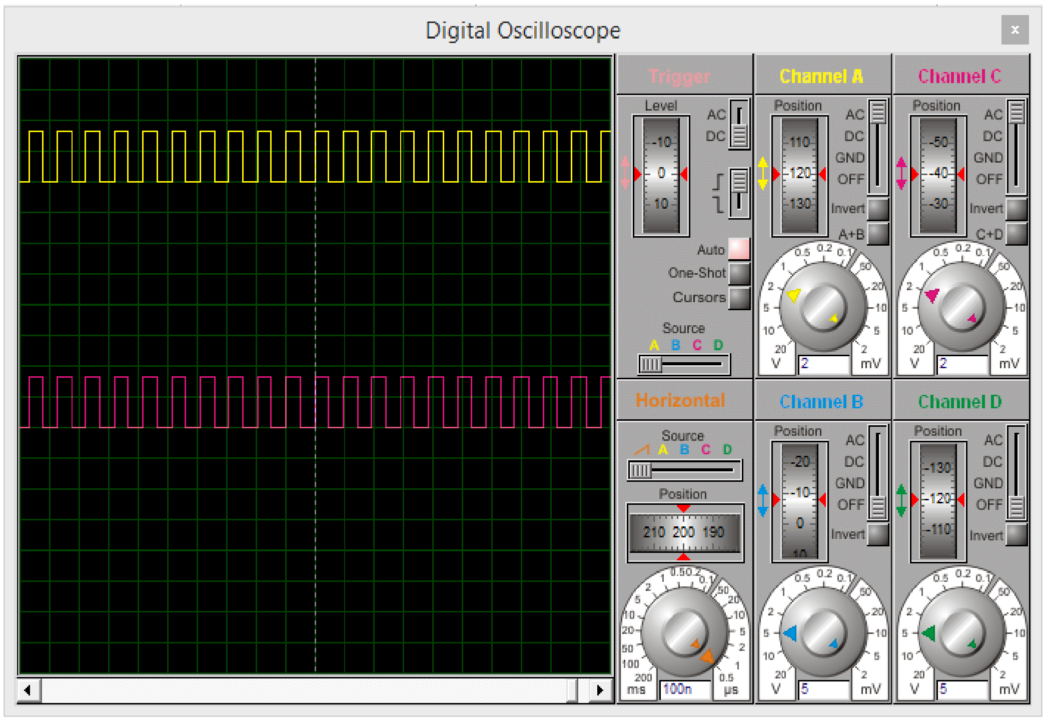


Figure 4-6: Oscilloscope output in Proteus

- Creating the PCB layout.

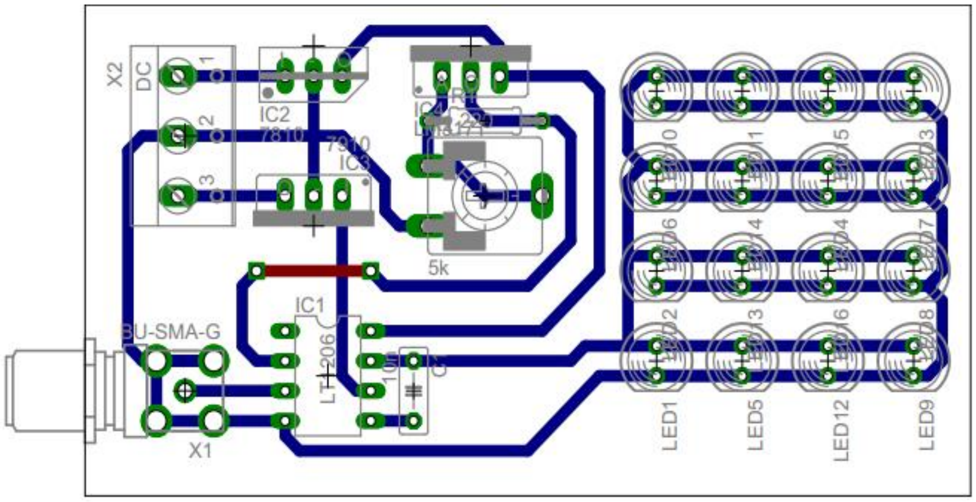


Figure 4-7: Transmitter circuit layout for print

- Implementing the PCB that was done outside the college at one of the electronics stores.

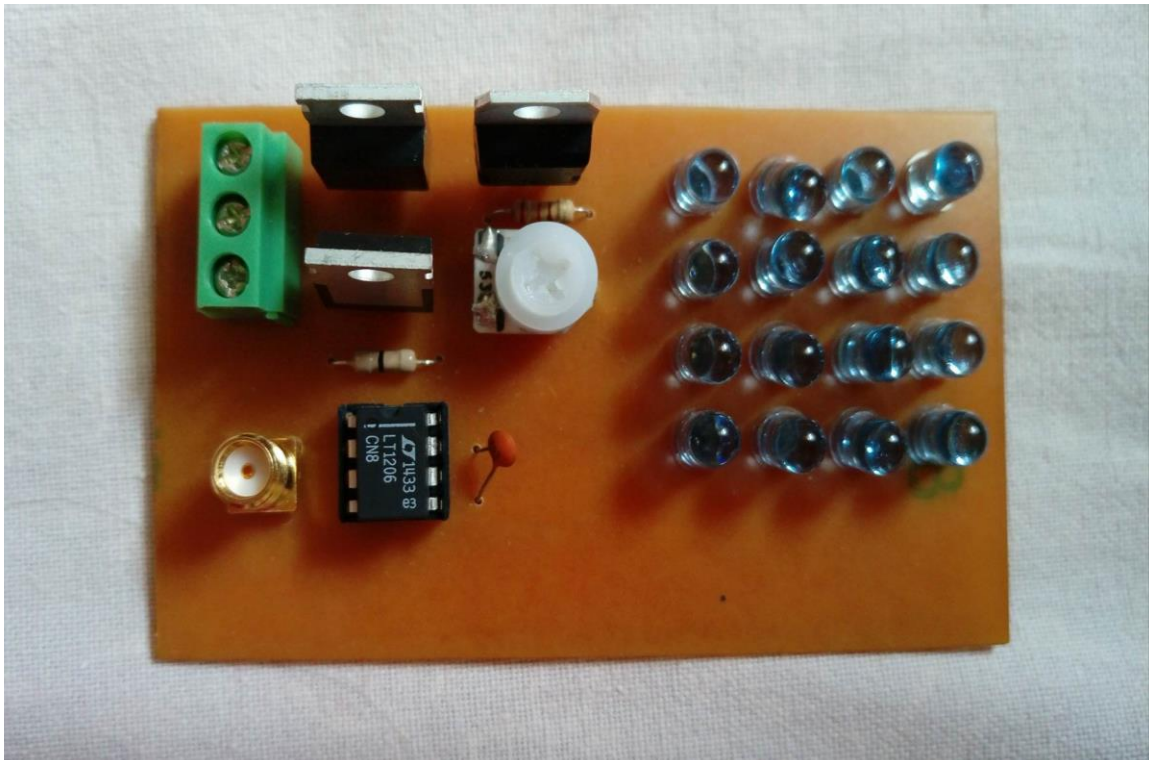


Figure 4-8: Transmitter PCB

The second step is trying to solve the two problems, adding the 50 ohm resistor in the simulations and lessening the value of the input signal to 0.8v instead of 3.3v and the regulator output which represents the comparator threshold to 0.4v the LEDs were not blinking and the voltage output from pin 7 gets lower till it reached 0.2 for the 0.8 high input, where the LEDs has a minimum voltage around 2.2v for high brightness and might accept a lower voltage in the range of 1.7v.

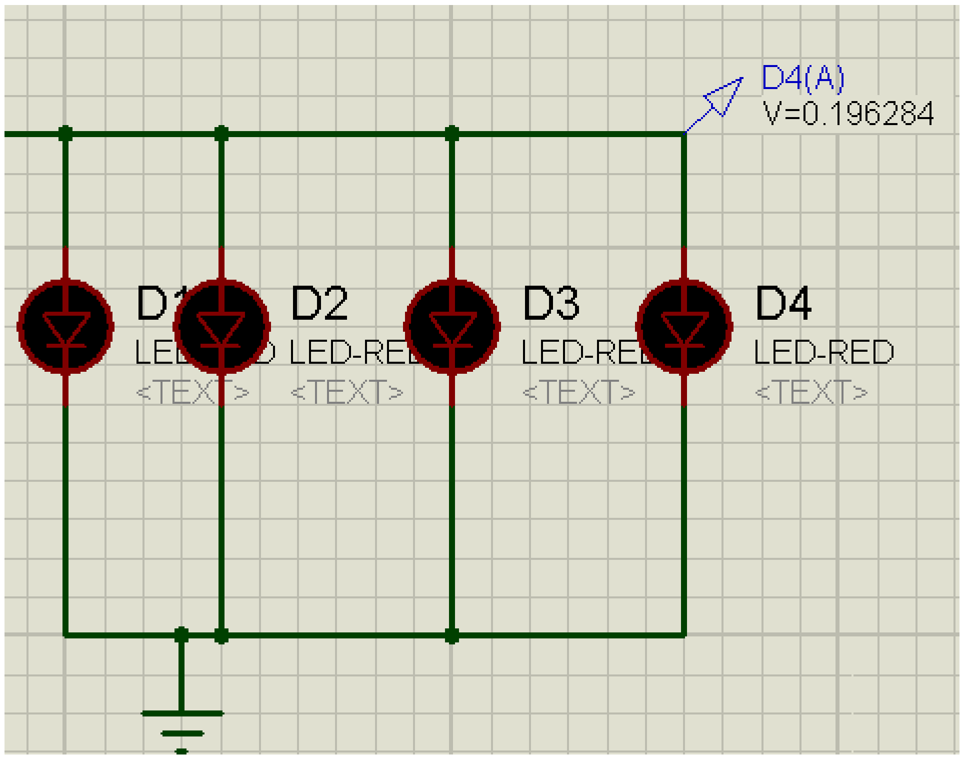


Figure 4-9: LEDs not blinking for 0.8v input

We tried using relay which is an electrically operated switch it is used where it is necessary to control a circuit by a separate low-power signal to control the old design transmitter with the small input voltage we have from the kit but this path has a serious problem which is the relay can’t operate at high frequency so we used common emitter circuit. The common emitter configuration has the highest power gain combined with medium voltage and current gain.

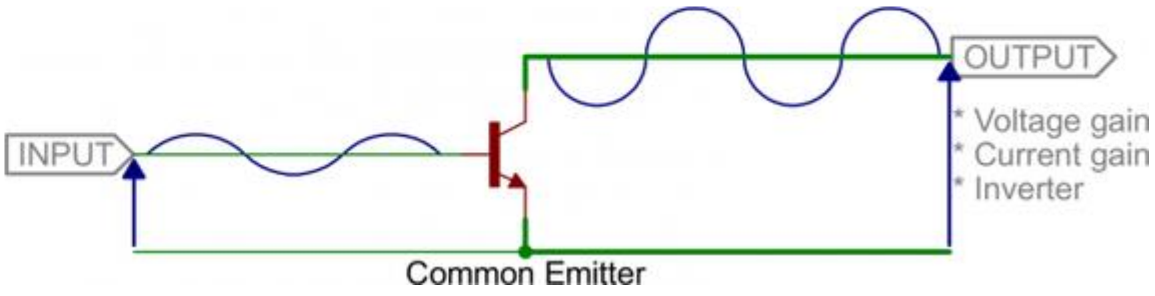


Figure 4-10: Common emitter theory of operation

The ability of this configuration in the above figure to increase input signal power by 20dB (100 times) and more is widely used as signal amplifiers in communications. Weak signals can restore their power passing through common-emitter amplifier.

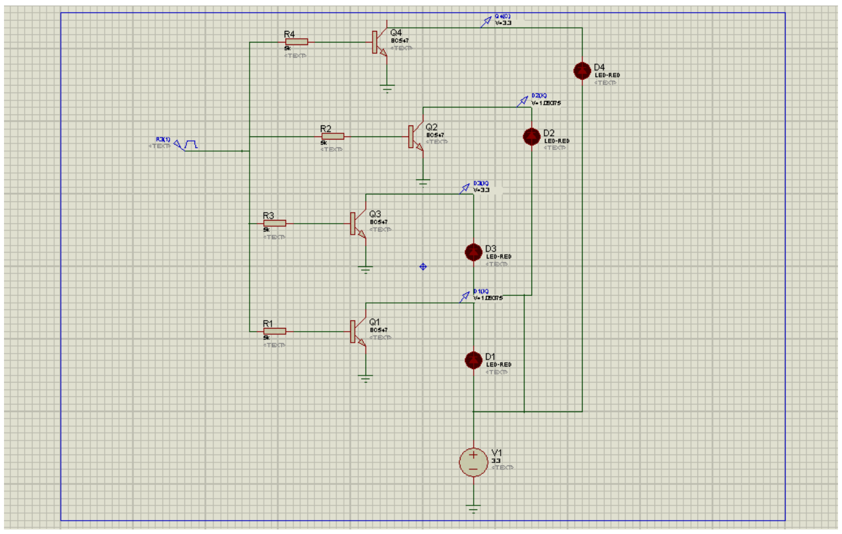


Figure 4-11: Transmitter circuit using common emitter amplifiers

As in fig 4-11 input is introduced to a resistance of 5K Ohm then to the base of the BJT , emitter is connected to ground and the collector output is connected to negative terminal of LED while positive terminal is connected to 3.3 v, when we have high signal 1 ‘0.8’

LEDs are lit on..

4.2 The Receiver Circuit

The receiver circuit on the other hand has photodiode that normally give an analog signal varying with the sensed light intensity, available photodiodes differ in their specifications regarding the slew rate and the spectral sensitivity .In the old design that used photodiode SFH203 with a five stage that are required finally to deliver a digital signal with 3.3v high voltage and -0.7v for the low voltage to the FPGA according to the blinking LEDs variations with the data sent.

The first and second stages were just amplifiers providing gain of 52 dB, the third stage was low pass filter implemented with Sallen-key design to filter the low frequency variations that is obtained from the surrounding ambient light and has to be ignored. The fourth stage has a comparator that will produce two values only thus transforming the analog signal into digital and the last stage consists of a level shifter to shift the volt down to suit the FPGA as 3.3 volt as input one and a -0.7 volt as an input zero.

The same steps followed with the transmitter circuit were followed with the receiver circuit.

- Simulating the design on CADsoft eagle for printing then on Proteus, the photodiode usually not found in simulators as Proteus instead we had to use an equivalent model for it as shown in the image below with standard calculations to get the values for the model parameters.

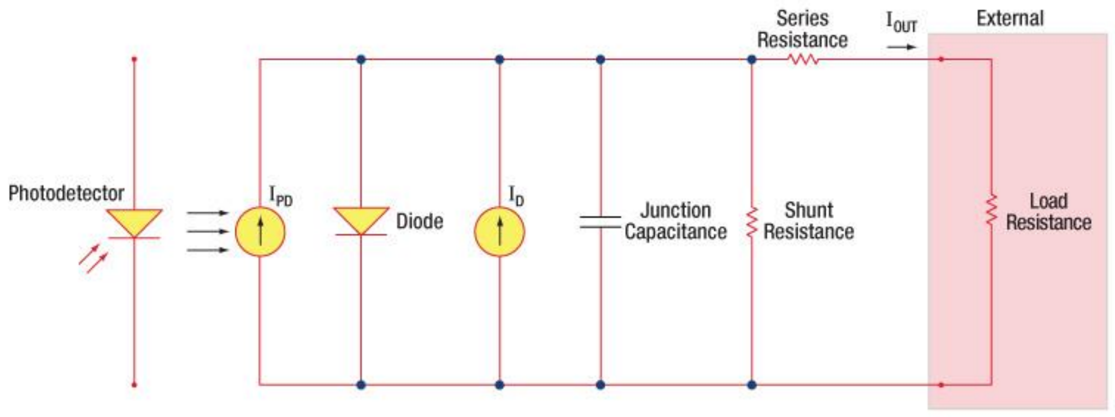


Figure 4-10: Photodiode equivalent model

Another solution that we did is to use the LDR with torch but it did not produce the output waveform as expected so we depended on hardware testing using a voltmeter with the already made circuits on the breadboards.

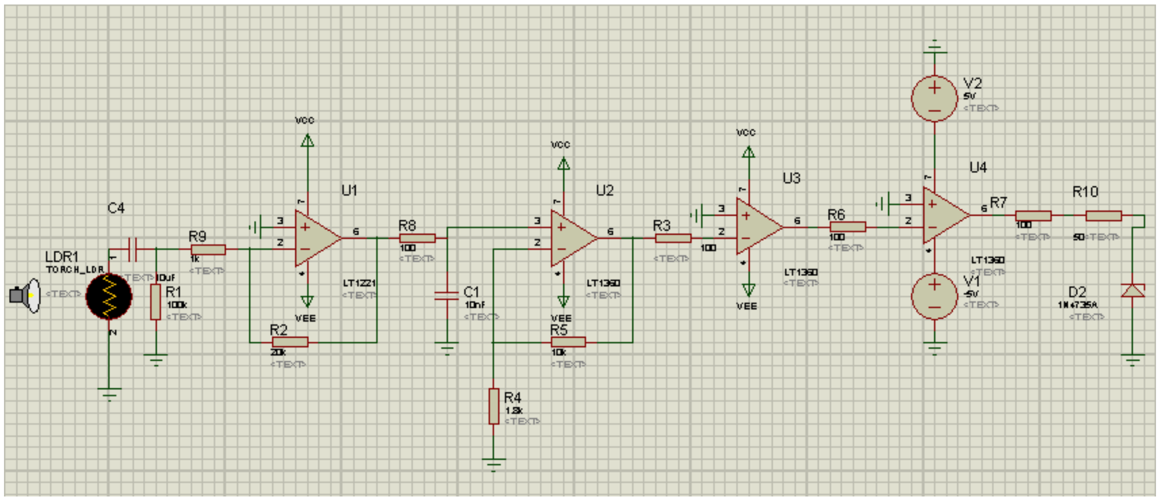


Figure 4-11: Receiver circuit simulation on Proteus using LDR

- Creating the PCB layout.

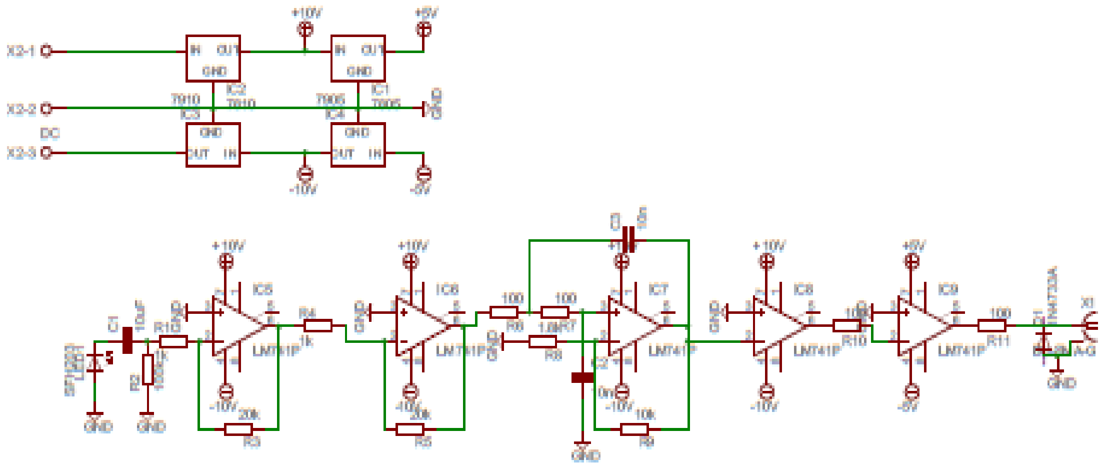


Figure 4-12: Receiver circuit design on CADsoft

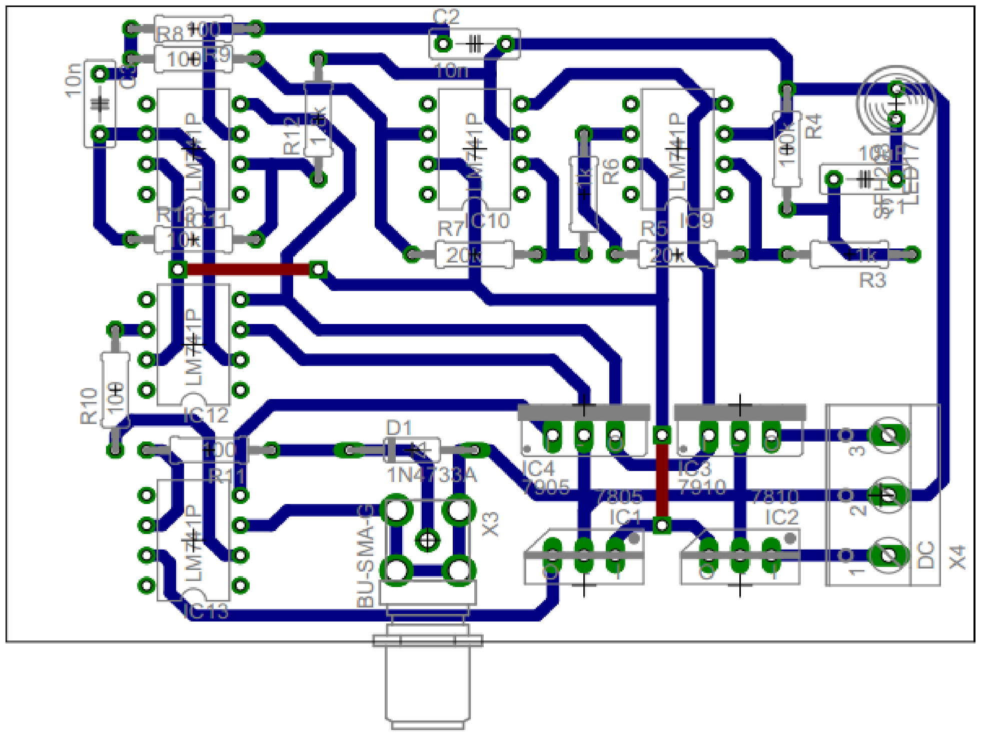


Figure 4-13: Receiver circuit layout for print

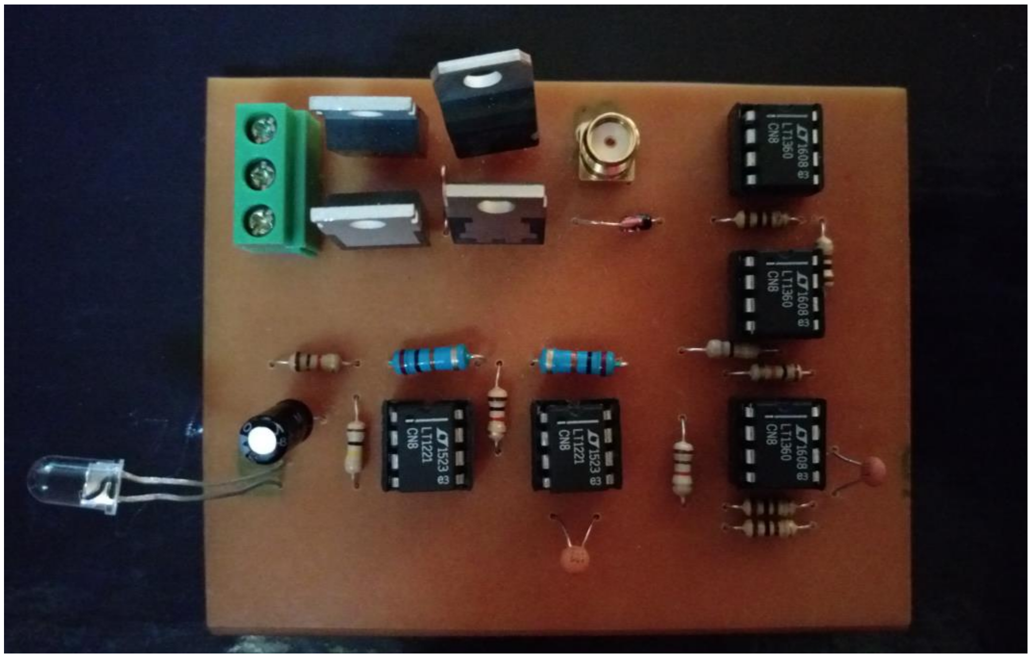


Figure 4-14: Receiver PCB

Chapter 5: **Monitoring System**

5.1 Design characterization

5.1.1 IBERT CORE

The Xilinx ChipScope™ Pro IBERT core for Spartan-6 GTP transceivers is self-contained and can be used to evaluate and monitor Spartan-6 GTP transceivers. The design includes pattern checkers implemented in FPGA logic. The IBERT core has all the logic to control, monitor, and modify transceiver features and perform bit error ratio tests in more integrated way.

5.1.2 IBERT COMPONENTS

- BERT Logic Part

The BERT logic shows the actual transceiver component, and have the pattern generators and checkers.In this part we have different available patterns such as PRBS (pseudo random bit sequence) patterns and comma disclosures.

- Dynamic Reconfiguration Port (DRP) Logic

Each transceiver has a Dynamic Reconfiguration Port (DRP) on it, so that transceiver properties can be modified in system. All properties and DRP addresses can be read and can be written in IBERT core.

5.1.3 IBERT DESIGN FLOW

Because the IBERT is a individualistic design, the design flow is very simple. When using the ChipScope IBERT Core Generator to generate IBERT core designs we get two important files the first one is the design directory and the other is the bit file. The design flow for generating IBERT core designs for Virtex-7, Kintex-7, Virtex-6, and Spartan-6 devices are very identical except the Xilinx CORE Generator tool is used. The main variance is that the design directory and device information is excluded in the Xilinx CORE Generator project. [11]

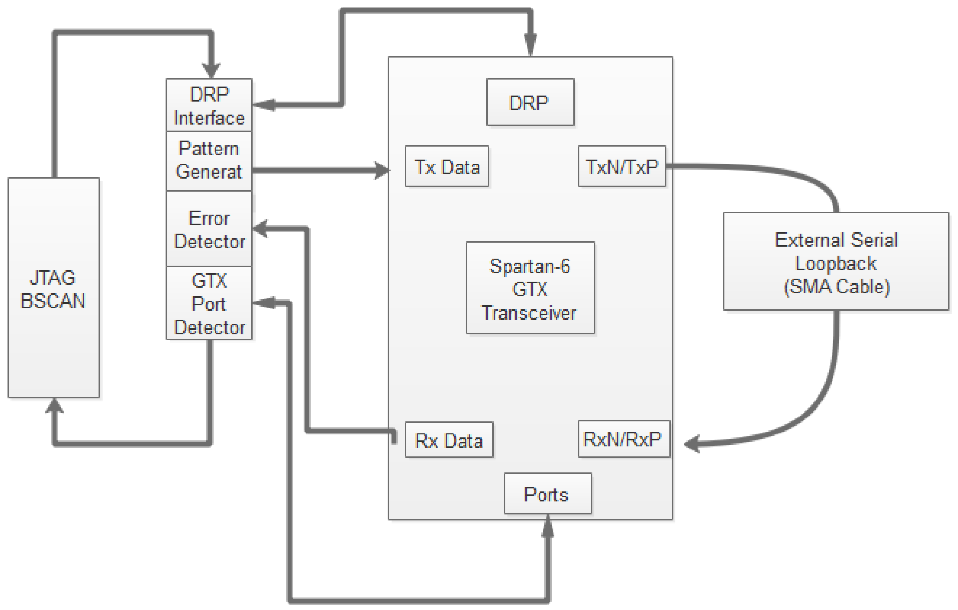


Figure 5-1: Design Flow of IBERT Core

5.1.4 IBERT FEATURES

- This feature supplies a communication path between the ChipScope Pro Analyzer software and the IBERT core itself.
- Has a user-selectable number of Spartan-6 GTP transceivers.
- Each transceiver can be specified for the desired line rate, reference clock rate and data path width.

The IBERT core supports a wider based Physical Medium Attachment (PMA) testing and demonstrated scheme for Spartan-6 GTP Transceiver. Data pattern generators and checkers are enclosed in each serial transceiver desired, giving a variance in PRBS and clock patterns to be sent over the channels. As well, the configuration and tuning of the serial transceivers can be accessed through logic that are connected to the DRP port of the serial transceiver, to change aspect settings, as well as registers that control the values on the ports. At run time testing the ChipScope Analyzer tool communicates to the IBERT core through JTAG using the Xilinx cables. [12]

5.1.5 SERIAL TRANSCEIVER FEATURES

IBERT is designed for PMA assessment and demonstration. All the major PMA features of the serial transceiver are supported and administrable in IBERT, including:

- Transmit (TX) pre-emphasis and post-emphasis
- TX differential swing
- Receive (RX) equalization
- Phase-Locked Loop (PLL) Divider settings

5.1.6 GENERATING THE CORE

IBERT has the feature to plot eye diagram for measuring Additive noise in the received signal , Peak distortion due to interruptions in the signal path ,Timing synchronization & jitter effects and Intersymbol interference. Eye diagram is an oscilloscope views in which a digital signal from a receiver is replicated sampled and applied to the vertical input, while the data rate is used to trigger the horizontal sweep.

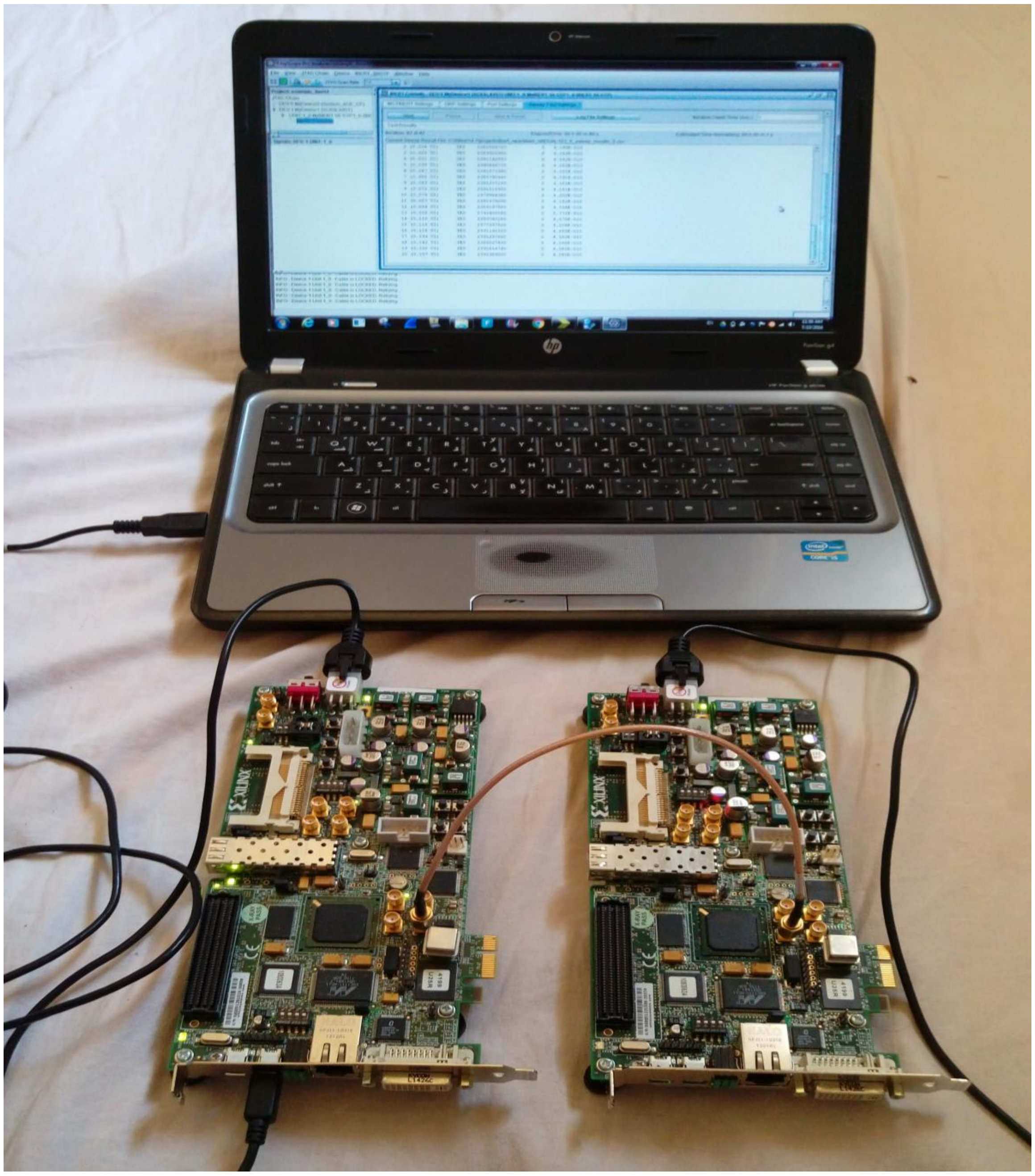
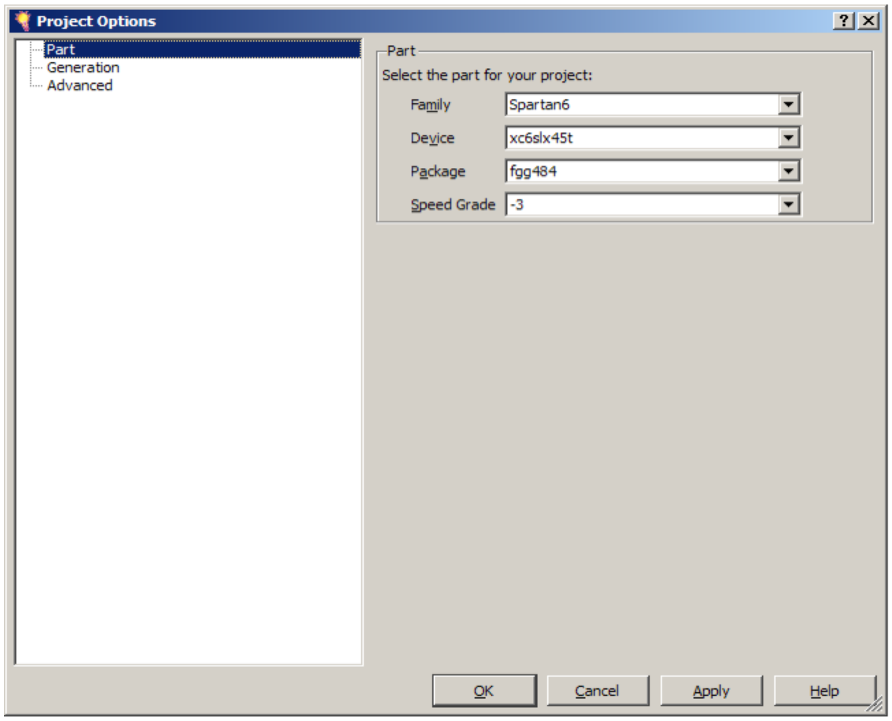


Figure 5-2: Hardware Connection Setting

It is named to be eye diagram for various types of coding, the pattern looks like a sequence of eyes between a pair of rails. It is an experimental tool for the testing of the combined effects of channel noise and intersymbol interference on the performance of a baseband pulse-transmission system. [13]

5.2 Design Creation, Customization and Generating of IBERT core

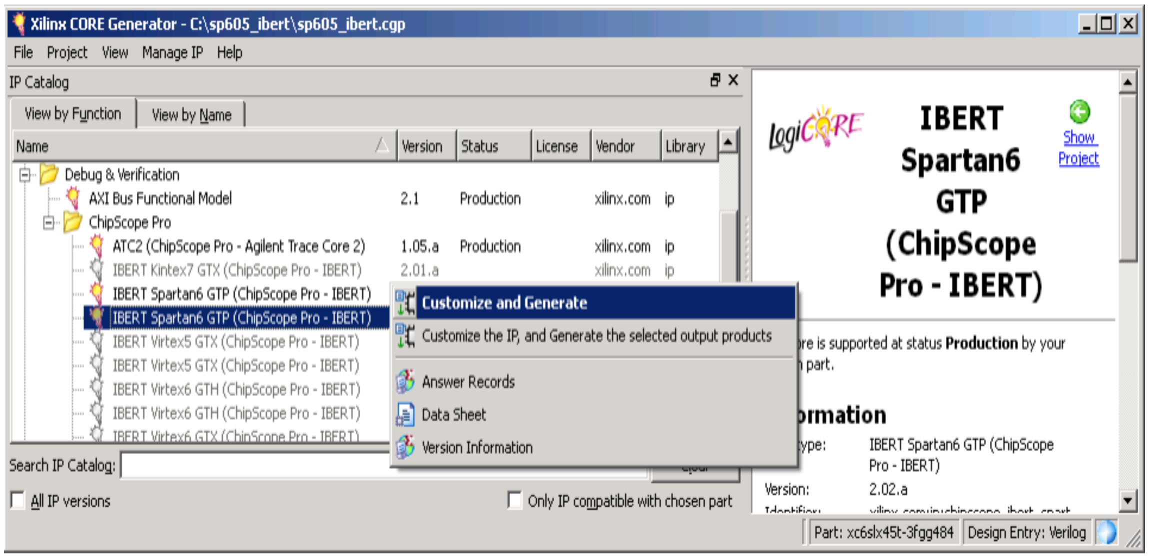
- The initial steps are already discussed in details in chapter 3 but here we choose in the "project options" those options as clear in figure below.



Part
Select the part for your project:
Family Spartan6
Device xc6slx45t
Package fgg484
Speed Grade -3

Figure 5-3: New project options part

- Create IBERT Design


Xilinx CORE Generator - C:\sp605_ibert\sp605_ibert.cgp

Name	Version	Status	License	Vendor	Library
Debug & Verification					
AXI Bus Functional Model	2.1	Production		xilinx.com	ip
ChipScope Pro					
ATC2 (ChipScope Pro - Agilent Trace Core 2)	1.05.a	Production		xilinx.com	ip
IBERT Kintex7 GTX (ChipScope Pro - IBERT)	2.01.a			xilinx.com	ip
IBERT Spartan6 GTP (ChipScope Pro - IBERT)					
IBERT Spartan6 GTP (ChipScope Pro - IBERT)					
IBERT Virtex5 GTX (ChipScope Pro - IBERT)					
IBERT Virtex5 GTX (ChipScope Pro - IBERT)					
IBERT Virtex6 GTH (ChipScope Pro - IBERT)					
IBERT Virtex6 GTH (ChipScope Pro - IBERT)					
IBERT Virtex6 GTX (ChipScope Pro - IBERT)					

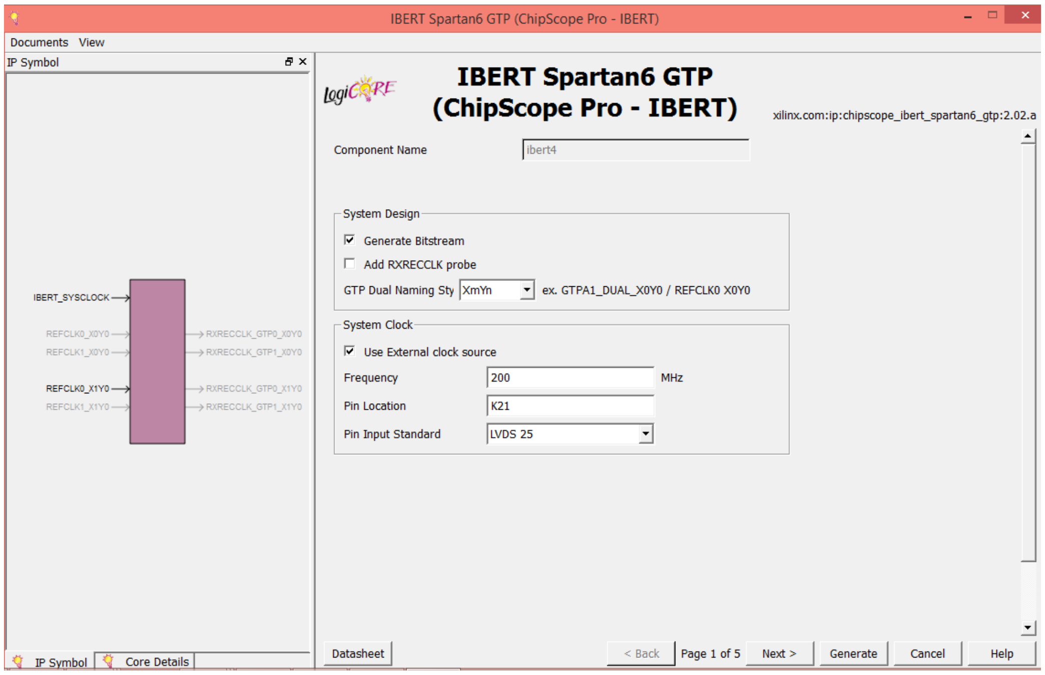
re is supported at status **Production** by your part.

ype: IBERT Spartan6 GTP (ChipScope Pro - IBERT)
Version: 2.02.a

Part: xc6slx45t-3fgg484
Design Entry: Verilog

Figure 5-4: Customizing and generating IBERT Core

- Select the following settings


IBERT Spartan6 GTP (ChipScope Pro - IBERT)

xilinx.com:ip:chipscope_ibert_spartan6_gtp:2.02.a

Component Name: ibert4

System Design
- [✓] Generate Bitstream
- [ ] Add RXRECCLK probe
- GTP Dual Naming Sty: XmYn — ex. GTPA1_DUAL_X0Y0 / REFCLK0 X0Y0

System Clock
- [✓] Use External clock source
- Frequency: 200 MHz
- Pin Location: K21
- Pin Input Standard: LVDS 25

Figure 5-5: Generating IBERT Core page 1 window

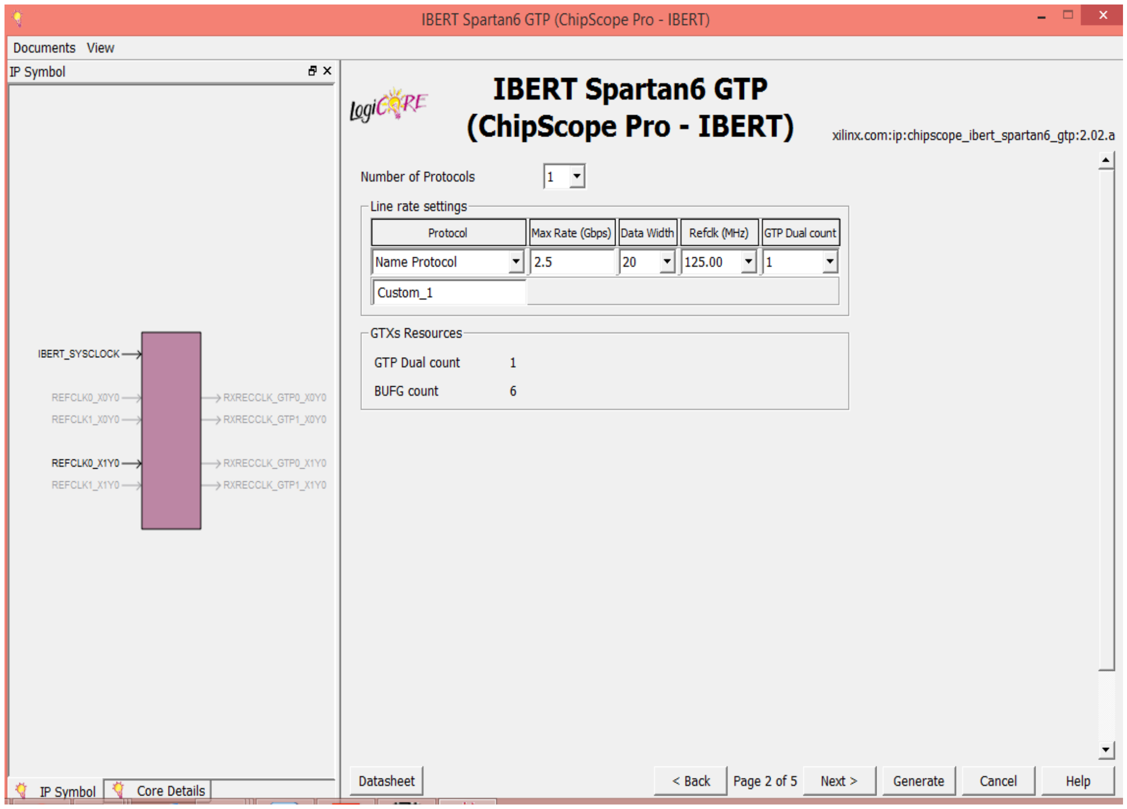
- Make the following settings:

Set the number of Protocols: 1

Set the line rate to Max Rate: 2.5 Gbps

Set the Refclk frequency to: 125 MHz

GTP Dual count: 1



IBERT Spartan6 GTP (ChipScope Pro - IBERT)

xilinx.com:ip:chipscope_ibert_spartan6_gtp:2.02.a

Number of Protocols: 1

Line rate settings

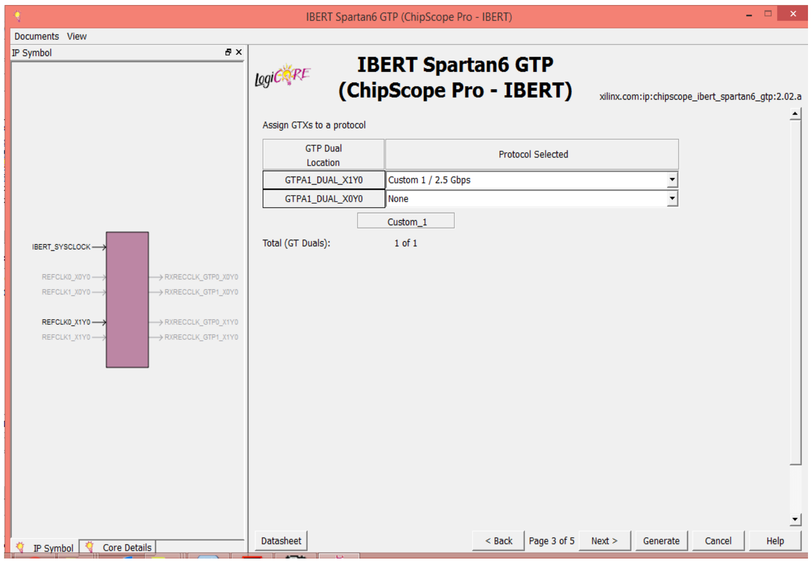
Protocol	Max Rate (Gbps)	Data Width	Refclk (MHz)	GTP Dual count
Name Protocol	2.5	20	125.00	1
Custom_1				

GTXs Resources

GTP Dual count	1
BUFG count	6

Figure 5-6: Generating IBERT Core page 2 window

Set the Protocol for both GTP Dual Locations: Custom 1 / 2.5 Gbps



IBERT Spartan6 GTP (ChipScope Pro - IBERT)

xilinx.com:ip:chipscope_ibert_spartan6_gtp:2.02.a

Assign GTXs to a protocol

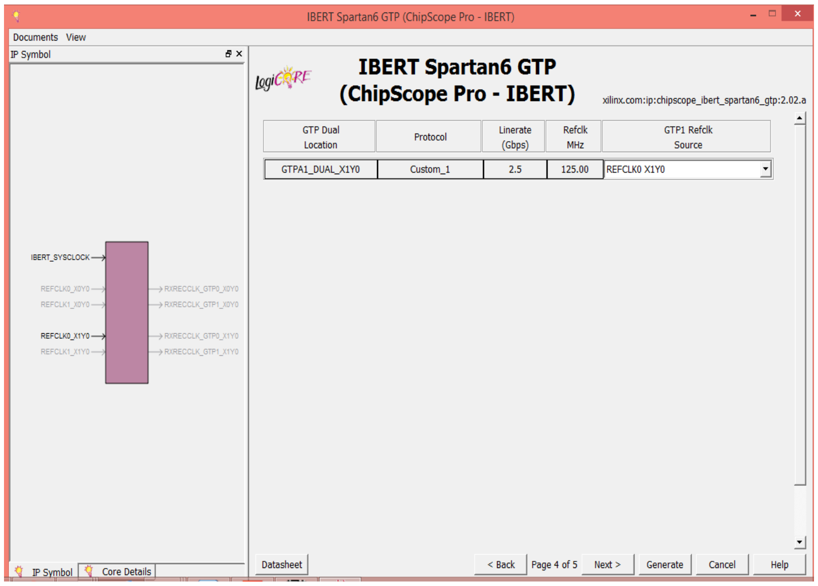
GTP Dual Location	Protocol Selected
GTPA1_DUAL_X1Y0	Custom 1 / 2.5 Gbps
GTPA1_DUAL_X0Y0	None

Custom_1

Total (GT Duals): 1 of 1

Figure 5-7: Generating IBERT Core page 3 window

- Set GTP Duals Refclk Sources to:REFCLK0 X1Y0



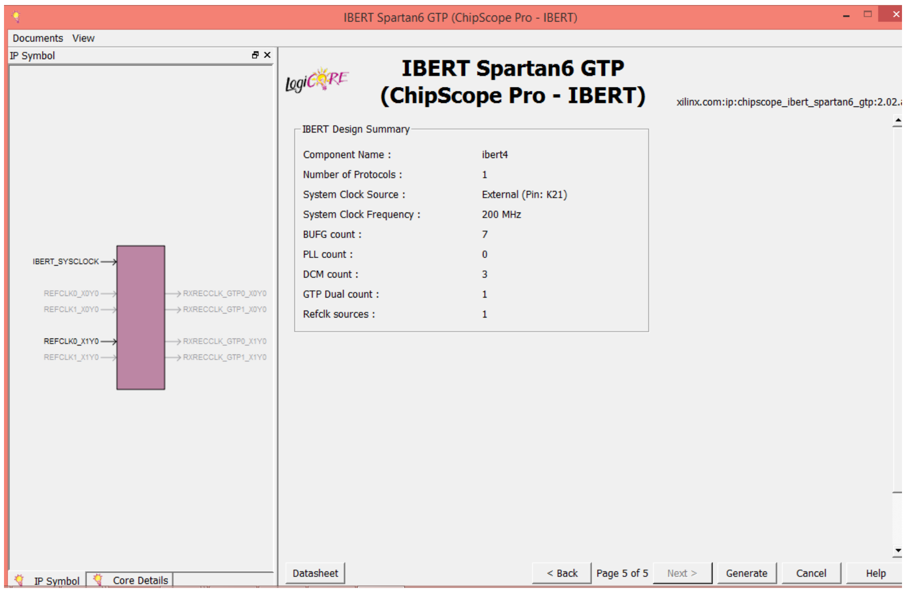
IBERT Spartan6 GTP (ChipScope Pro - IBERT)

xilinx.com:ip:chipscope_ibert_spartan6_gtp:2.02.a

GTP Dual Location	Protocol	Linerate (Gbps)	Refclk MHz	GTP1 Refclk Source
GTPA1_DUAL_X1Y0	Custom_1	2.5	125.00	REFCLK0 X1Y0

Figure 5-8: Generating IBERT Core page 4 window

- Click Generate



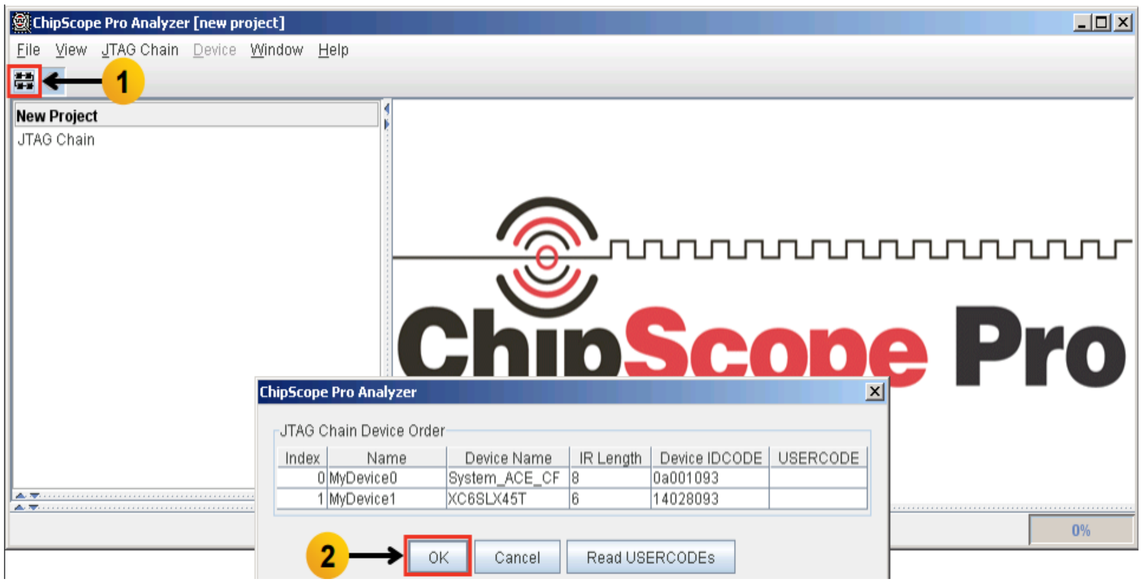
IBERT Design Summary

Component Name :	ibert4
Number of Protocols :	1
System Clock Source :	External (Pin: K21)
System Clock Frequency :	200 MHz
BUFG count :	7
PLL count :	0
DCM count :	3
GTP Dual count :	1
Refclk sources :	1

Figure 5-9: Generating IBERT Core page 5 window

5.3 Dealing with the Design using ChipScope pro Analyzer

- Open ChipScope Pro and click on the Open Cable Button

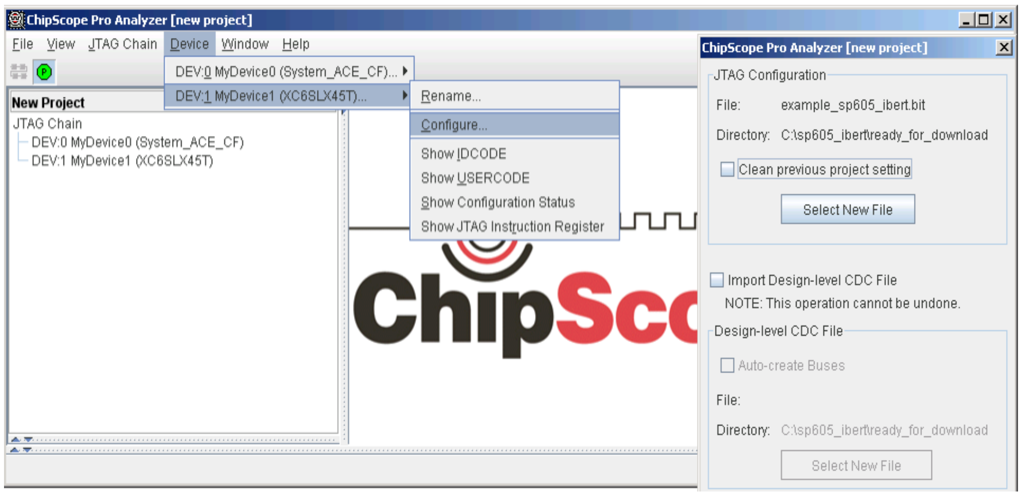


Index	Name	Device Name	IR Length	Device IDCODE	USERCODE
0	MyDevice0	System_ACE_CF	8	0a001093	
1	MyDevice1	XC6SLX45T	6	14028093	

Figure 5-10: IBERT testing on ChipScope Pro page 1 Window

- Select Device > DEV:1 MyDevice1 (XC6SLX45T) >Configure

Select<design path>ready_for_download\ example_sp605_ibert.bit

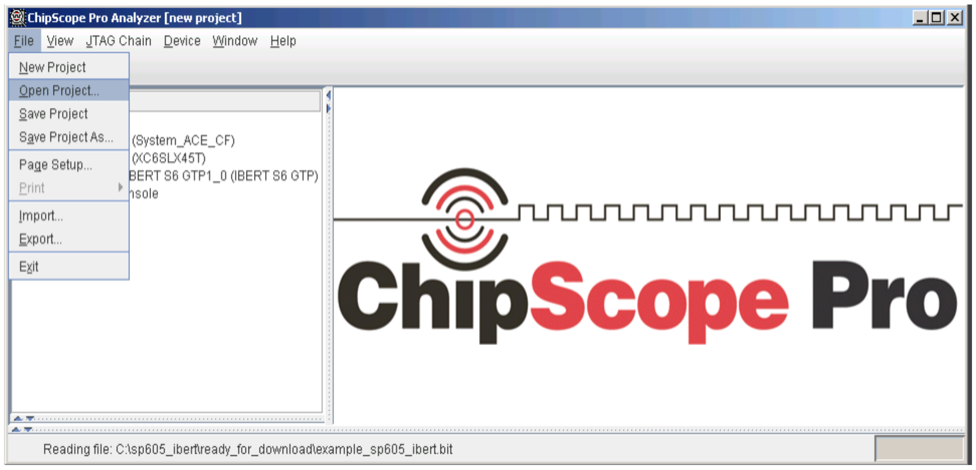

ChipScope Pro Analyzer [new project]
File View JTAG Chain Device Window Help
DEV:0 MyDevice0 (System_ACE_CF)...
DEV:1 MyDevice1 (XC6SLX45T)...
Rename...
Configure...
Show IDCODE
Show USERCODE
Show Configuration Status
Show JTAG Instruction Register
New Project
JTAG Chain
DEV:0 MyDevice0 (System_ACE_CF)
DEV:1 MyDevice1 (XC6SLX45T)
ChipScope Pro Analyzer [new project]
JTAG Configuration
Clean previous project setting
Select New File
Import Design-level CDC File
Design-level CDC File
Auto-create Buses
Select New File

File: example_sp605_ibert.bit
Directory: C:\sp605_ibert\ready_for_download
NOTE: This operation cannot be undone.
File:
Directory: C:\sp605_ibert\ready_for_download

Figure 5-11: IBERT testing on ChipScope Pro page 2 Window

-Select File > Open Project.

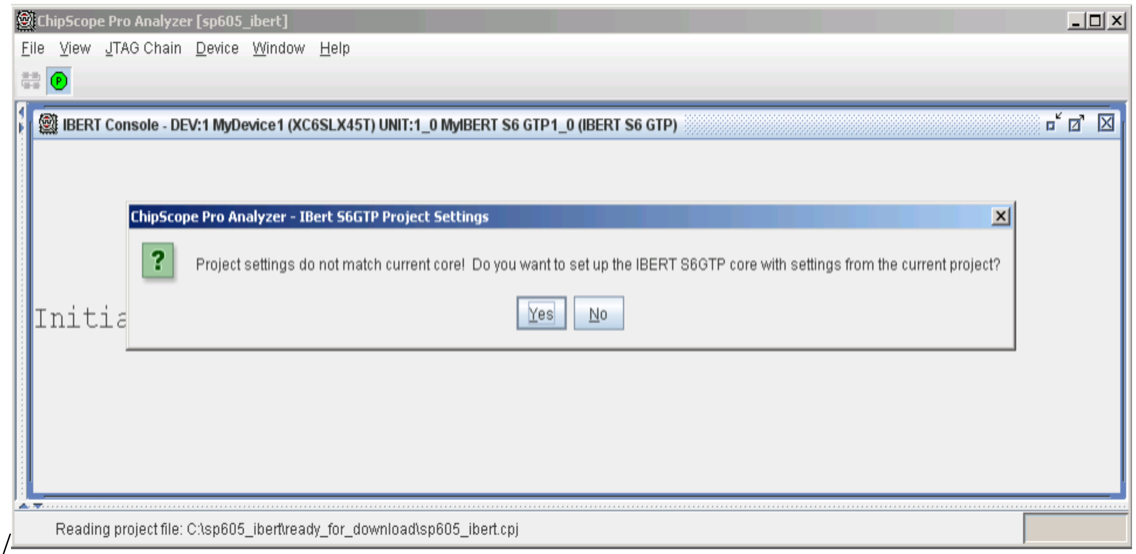
- Select \ready_for_download\sp605_ibert.cpj



Reading file: C:\sp605_ibert\ready_for_download\example_sp605_ibert.bit

Figure 5-12: IBERT testing on ChipScope Pro page 3 Window

- Click Yes on this Dialog


ChipScope Pro Analyzer [sp605_ibert]
File View JTAG Chain Device Window Help
IBERT Console - DEV:1 MyDevice1 (XC6SLX45T) UNIT:1_0 MyIBERT S6 GTP1_0 (IBERT S6 GTP)
ChipScope Pro Analyzer - IBert S6GTP Project Settings
Initia
Yes
No

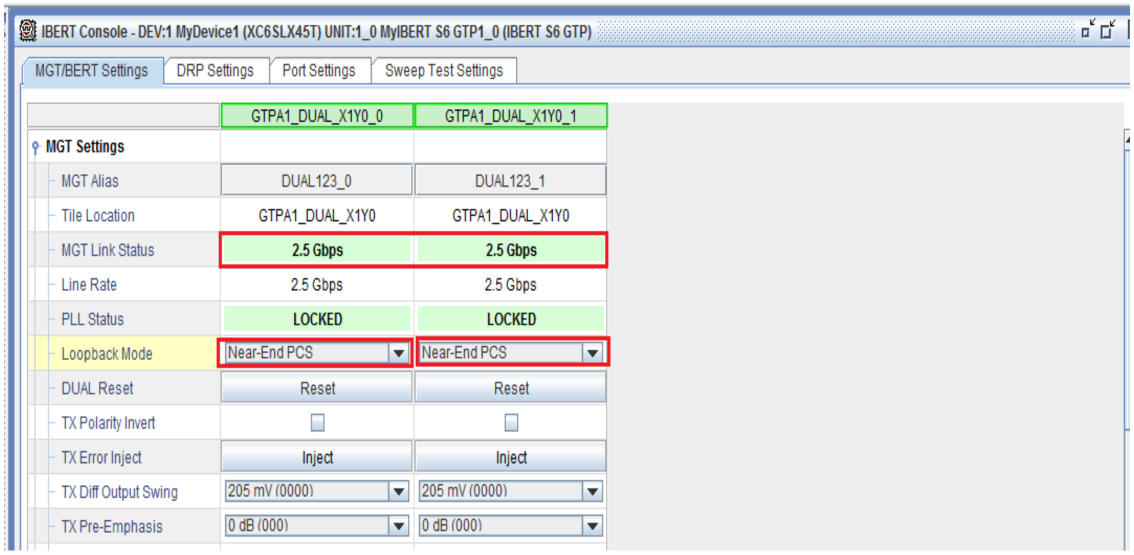
Project settings do not match current core! Do you want to set up the IBERT S6GTP core with settings from the current project?

Reading project file: C:\sp605_ibert\ready_for_download\sp605_ibert.cpj

/

Figure 5-13: IBERT testing on ChipScope Pro page 4 Window

-The line rate is 2.5 Gbps for all four GTPs Near-End PMA is selected for the PCIe and FMC GTPs and TX Diff Output Swing = 205 mV (0000)

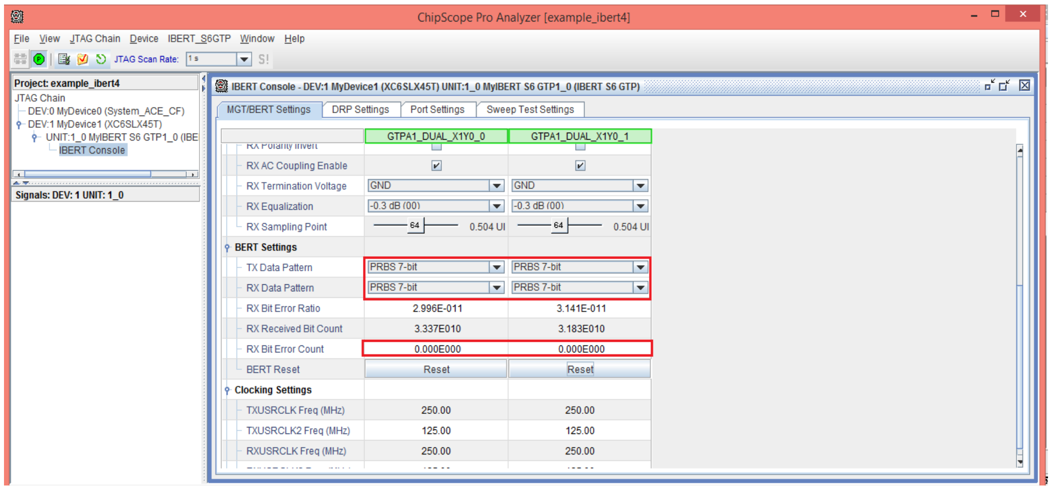

IBERT Console - DEV:1 MyDevice1 (XC6SLX45T) UNIT:1_0 MyIBERT S6 GTP1_0 (IBERT S6 GTP)
MGT/BERT Settings
DRP Settings
Port Settings
Sweep Test Settings

	GTPA1_DUAL_X1Y0_0	GTPA1_DUAL_X1Y0_1
MGT Settings		
MGT Alias	DUAL123_0	DUAL123_1
Tile Location	GTPA1_DUAL_X1Y0	GTPA1_DUAL_X1Y0
MGT Link Status	2.5 Gbps	2.5 Gbps
Line Rate	2.5 Gbps	2.5 Gbps
PLL Status	LOCKED	LOCKED
Loopback Mode	Near-End PCS	Near-End PCS
DUAL Reset	Reset	Reset
TX Polarity Invert	☐	☐
TX Error Inject	Inject	Inject
TX Diff Output Swing	205 mV (0000)	205 mV (0000)
TX Pre-Emphasis	0 dB (000)	0 dB (000)

Figure 5-14: IBERT Console for MGT/BERT Settings page 1 Window

- TX/RX Data Patterns are set to PRBS 7-bit then Click BERT Reset buttons.

After pressing the BERT Reset buttons   the RX Bit Error Count becomes as shown.


ChipScope Pro Analyzer [example_ibert4]
File View JTAG Chain Device IBERT_S6GTP Window Help
JTAG Scan Rate: 1 s
Project: example_ibert4
JTAG Chain
DEV:0 MyDevice0 (System_ACE_CF)
DEV:1 MyDevice1 (XC6SLX45T)
UNIT:1_0 MyIBERT S6 GTP1_0 (IBE
IBERT Console
Signals: DEV: 1 UNIT: 1_0
IBERT Console - DEV:1 MyDevice1 (XC6SLX45T) UNIT:1_0 MyIBERT S6 GTP1_0 (IBERT S6 GTP)
MGT/BERT Settings
DRP Settings
Port Settings
Sweep Test Settings

	GTPA1_DUAL_X1Y0_0	GTPA1_DUAL_X1Y0_1
RX Polarity Invert	☐	☐
RX AC Coupling Enable	✓	✓
RX Termination Voltage	GND	GND
RX Equalization	-0.3 dB (00)	-0.3 dB (00)
RX Sampling Point	64 0.504 UI	64 0.504 UI
BERT Settings		
TX Data Pattern	PRBS 7-bit	PRBS 7-bit
RX Data Pattern	PRBS 7-bit	PRBS 7-bit
RX Bit Error Ratio	2.996E-011	3.141E-011
RX Received Bit Count	3.337E010	3.183E010
RX Bit Error Count	0.000E000	0.000E000
BERT Reset	Reset	Reset
Clocking Settings		
TXUSRCLK Freq (MHz)	250.00	250.00
TXUSRCLK2 Freq (MHz)	125.00	125.00
RXUSRCLK Freq (MHz)	250.00	250.00

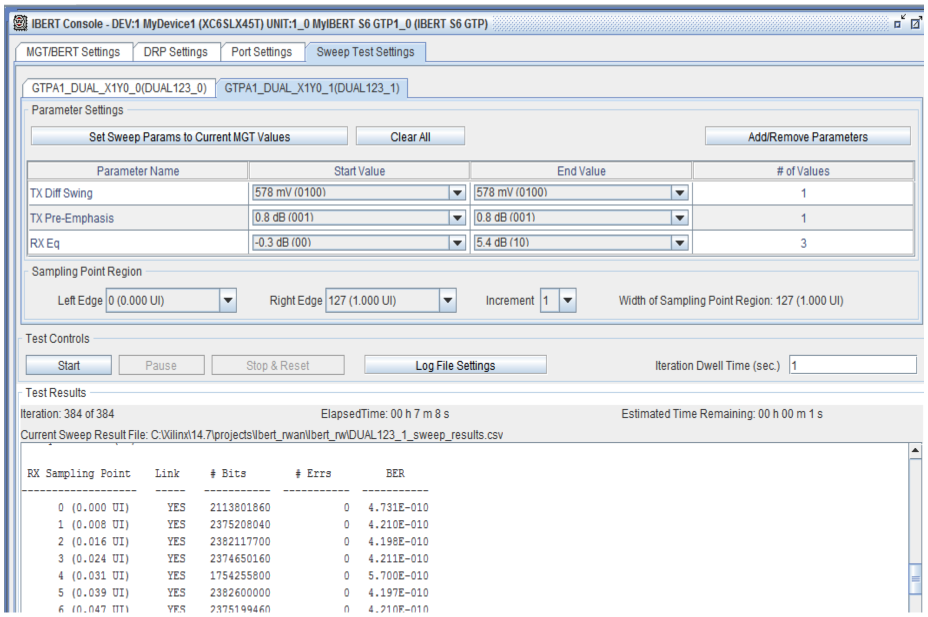
Figure 5-15: IBERT Console for MGT/BERT Settings page 2 Window

5.4 Performing Sweep Test Analysis

Sweep test is a tool in IBERT that can set up a channel test that sweeps through a variety of transceiver settings.The sweeping test is available only if you set the transceivers to one of the near-end or external loopack .In our settings for the previous step in dealing with ChipScope pro we applied near-end loopback.

The most important part in dealing with sweeping test that we become able to get several analysis aspects in detecting the performance of wireless channel using GTP transceivers which is compatible with our system.

In this part, we can get a log file which implies all the tested features mainly the bit error rate through a specified number of iterations inserted in the options of the sweep test panel settings .In the sweep test plot ,you can plot an eyediagram with a 2D scan to detect the troubleshooting and noise additives of the channel at the receiver side.


IBERT Console - DEV:1 MyDevice1 (XC6SLX45T) UNIT:1_0 MyIBERT S6 GTP1_0 (IBERT S6 GTP)

MGT/BERT Settings | DRP Settings | Port Settings | Sweep Test Settings

GTPA1_DUAL_X1Y0_0(DUAL123_0) | GTPA1_DUAL_X1Y0_1(DUAL123_1)

Parameter Settings

Set Sweep Params to Current MGT Values | Clear All | Add/Remove Parameters

Parameter Name	Start Value	End Value	# of Values
TX Diff Swing	578 mV (0100)	578 mV (0100)	1
TX Pre-Emphasis	0.8 dB (001)	0.8 dB (001)	1
RX Eq	-0.3 dB (00)	5.4 dB (10)	3

Sampling Point Region

Left Edge 0 (0.000 UI) Right Edge 127 (1.000 UI) Increment 1 Width of Sampling Point Region: 127 (1.000 UI)

Test Controls

Start | Pause | Stop & Reset | Log File Settings | Iteration Dwell Time (sec.) 1

Test Results

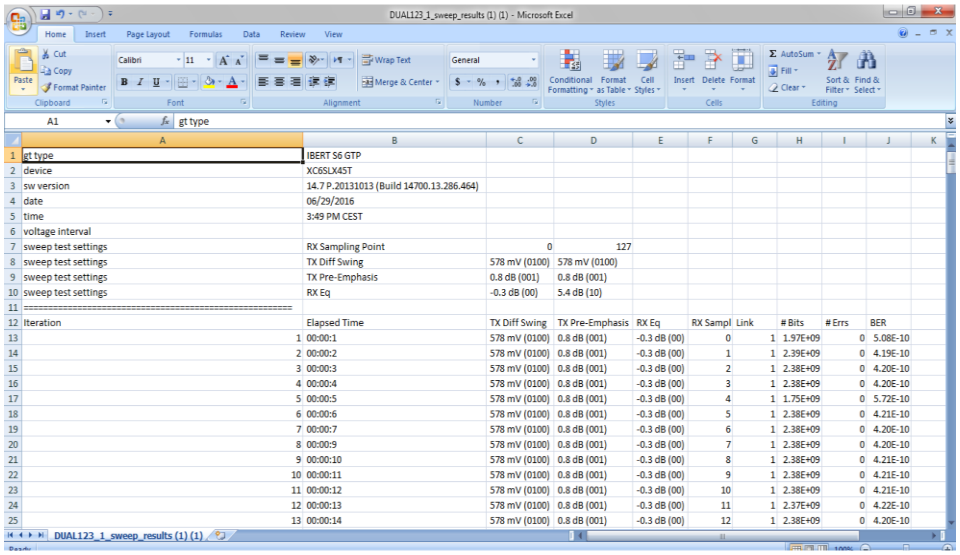
Iteration: 384 of 384 ElapsedTime: 00 h 7 m 8 s Estimated Time Remaining: 00 h 00 m 1 s

Current Sweep Result File: C:\Xilinx\14.7\projects\Ibert_rwan\Ibert_rw\DUAL123_1_sweep_results.csv

```
 RX Sampling Point   Link     # Bits       # Errs        BER
-------------------  -----  -----------  -----------  -----------
      0 (0.000 UI)    YES    2113801860            0   4.731E-010
      1 (0.008 UI)    YES    2375208040            0   4.210E-010
      2 (0.016 UI)    YES    2382117700            0   4.198E-010
      3 (0.024 UI)    YES    2374650160            0   4.211E-010
      4 (0.031 UI)    YES    1754255800            0   5.700E-010
      5 (0.039 UI)    YES    2382600000            0   4.197E-010
      6 (0.047 UI)    YES    2375199460            0   4.210E-010
```

Figure 5-16: IBERT Console for Sweep test Settings Window

Extracting the Log file from the "Log file setting" tab


DUAL123_1_sweep_results (1) (1) - Microsoft Excel

A1 gt type

	A	B	C	D	E	F	G	H	I	J
1	gt type	IBERT S6 GTP								
2	device	XC6SLX45T								
3	sw version	14.7 P.20131013 (Build 14700.13.286.464)								
4	date	06/29/2016								
5	time	3:49 PM CEST								
6	voltage interval									
7	sweep test settings	RX Sampling Point	0	127						
8	sweep test settings	TX Diff Swing	578 mV (0100)	578 mV (0100)						
9	sweep test settings	TX Pre-Emphasis	0.8 dB (001)	0.8 dB (001)						
10	sweep test settings	RX Eq	-0.3 dB (00)	5.4 dB (10)						
11	===									
12	Iteration	Elapsed Time	TX Diff Swing	TX Pre-Emphasis	RX Eq	RX Sampl	Link	# Bits	# Errs	BER
13	1	00:00:1	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	0	1	1.97E+09	0	5.08E-10
14	2	00:00:2	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	1	1	2.39E+09	0	4.19E-10
15	3	00:00:3	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	2	1	2.38E+09	0	4.20E-10
16	4	00:00:4	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	3	1	2.38E+09	0	4.20E-10
17	5	00:00:5	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	4	1	1.75E+09	0	5.72E-10
18	6	00:00:6	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	5	1	2.38E+09	0	4.21E-10
19	7	00:00:7	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	6	1	2.38E+09	0	4.20E-10
20	8	00:00:9	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	7	1	2.38E+09	0	4.20E-10
21	9	00:00:10	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	8	1	2.38E+09	0	4.21E-10
22	10	00:00:11	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	9	1	2.38E+09	0	4.21E-10
23	11	00:00:12	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	10	1	2.38E+09	0	4.21E-10
24	12	00:00:13	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	11	1	2.37E+09	0	4.22E-10
25	13	00:00:14	578 mV (0100)	0.8 dB (001)	-0.3 dB (00)	12	1	2.38E+09	0	4.20E-10

DUAL123_1_sweep_results (1) (1)

Figure 5-17: Log file Extracted from Sweep test panel

Chapter 6: **Conclusion and Future Work**

We can summarize our results and improvements as follows:

- Building the PHY and MAC layers of the system software in an optimized VHDL coding making use of the IP cores provided by Xilinix for Spartan-6 to make a top level module for a VLC transmitter and receiver.
- Providing hardware design and implementation for the transmitter and receiver circuits working with UART serial communications protocol that can support relatively acceptable speeds and only requires connecting the Ethernet TEMAC with UART instead of GTP which we tried implementing during the last duration of the project in order to show the hardware in a working state. In addition to the main hardware of the PCBs implemented with the SMA sockets that require small modification to work with the high frequencies and low voltage levels that we moved through more than one approach to solve it
- An application layer in the form of a monitoring system ready to output some channel parameters after setting it.

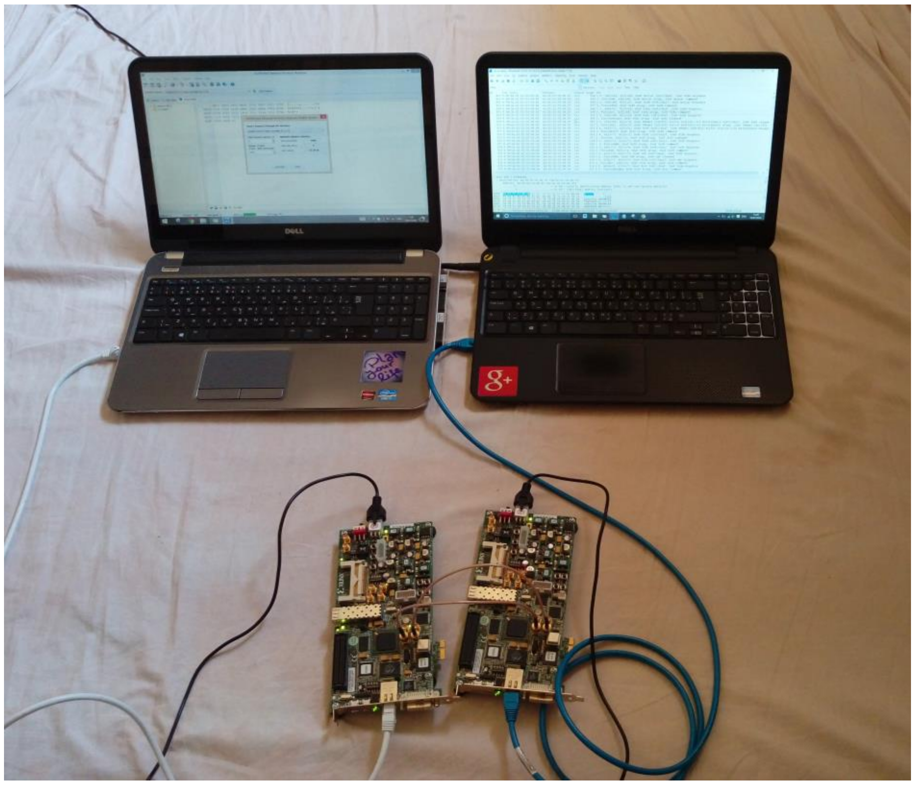


Figure 6-1: Final System Setting

To sum up, providing high data rate Internet access, VLC channel enhancement and creating a software application to monitor the system performance are the main features of the project's second phase.

Internet access over visible light communication has been a trend in research and development. And we believe this prototype and our work flow will become an important milestone for future enhancements that could be:

- Integrating the prototype in a more practical handheld hardware.
- Creating a multiuser environment (using RGB and a multiple access technique).
- Including mobile phone users, which is an important deliverable as Internet is commonly used by mobile devices more than PCs due to the great and fast development in smart phones and tablets which made Apple consider it in its future versions of Iphone. But can

be implemented on a simpler scale we searched in this track to add it to our system and found out that there are light sensors that can be used with relatively high frequencies rather than the camera that can detect only low frequencies. Such sensors can be connected using the headphone jack as shown in the image below. This requires a quite good knowledge of Android or IOS system to modify on the default Internet access technique using either Wifi or Mobile data.

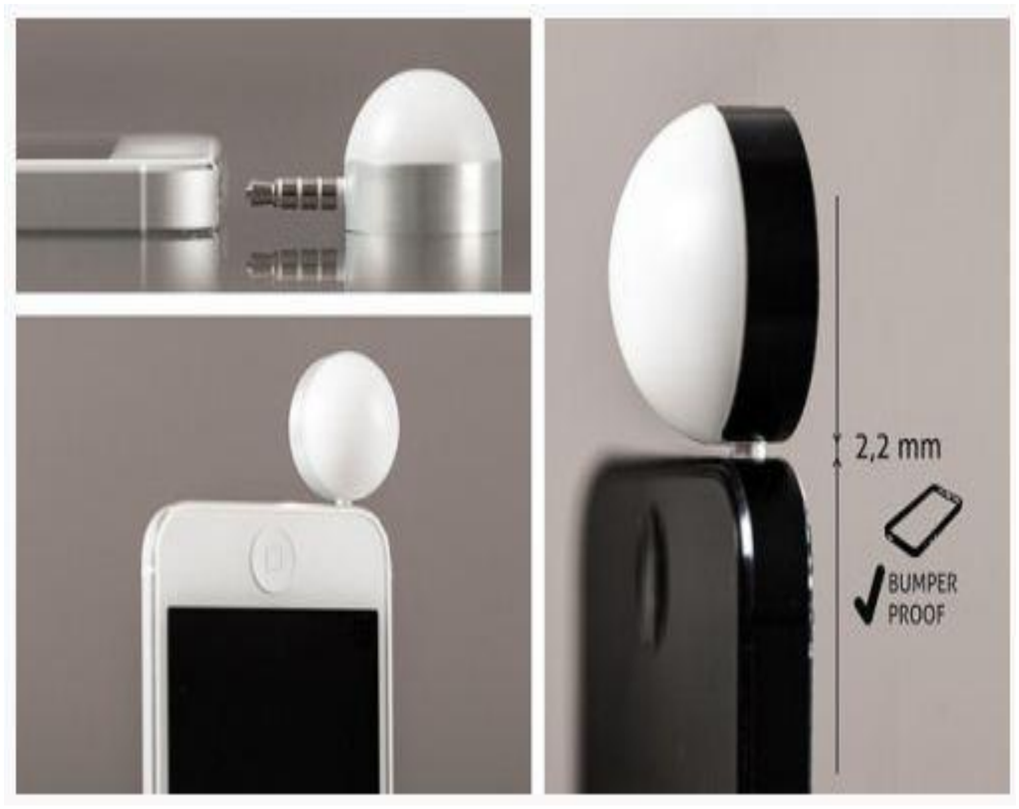


Figure 6-2: Visible light sensor through headphone jack

References

[1] http://spectrum.ieee.org/telecom/internet/lifi-gets-ready-to-compete-with-wifi.

[2] https://en.wikipedia.org/wiki/Li-Fi#History.

[3] http://luxreview.com/article/2016/02/breaking-apple-set-to-add-lifi-capability-to-iphone.

[4] http://www.lifi-lab.com/lifi/lifi-vlc-analisi-2014-2020.html.

[5] http://www-inst.eecs.berkeley.edu/~cs150/sp10/Collections/ChipScope.pdf.

[6] http://www.xilinx.com/products/intellectual-property/axi_interconnect.html.

[7] http://www.xilinx.com/support/documentation/boards_and_kits/ug525.pdf.

[8] www.eeherald.com/section/design-guide/esmod12.html.

[9] http://www.xilinx.com/support/documentation/ip_documentation/s6_gtpwizard/v1_11/ug546_s6_gtpwizard.pdf.

[10] https://en.wikipedia.org/wiki/SMA_connector.

[11] http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/chipscope_pro_sw_cores_ug029.pdf.

[12] http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug811_ChipScopeUsingIBERTwithAnalyzer.pdf.

[13] http://www.xilinx.com/support/documentation/ip_documentation/chipscope_ibert_spartan6_gtp.pdf.