UPPSALA
UNIVERSITET

IT 16046
Examensarbete 30 hp
Juni 2016

The Timing and Fast Control Demonstrator

Jiheng Chen
Vasileios Filos

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

The Timing and Fast Control Demonstrator

Jiheng Chen and Vasileios Filos

In this thesis, the feasibility of an FPGA to host a system for generating and distributing clocks as well as distributing synchronous and asynchronous commands is tested. This system will be an imitation of the SHiP (Search for Hidden Particle) DAQ (Data Acquisition) system led by CERN. The practical implementation is to mainly apply Altera Cyclone V GT development board attached with SFP+ daughter card to achieve accurate timing and high speed performance.
Experiments include three loopback test implementations. Loopback test is the simplest technique to assess a channel's integration. The first one is the Ethernet loopback test. An Ethernet card daughter board is inserted to the HSMC port of the Cyclone V GT development board. After that, a SFP card is applied alternatively on the same port to do the similar loopback test but at a much higher speed via optical fibers. And finally, a more advanced XAUI to SFP+ card daughter board will be used to replace the previous SFP card in order to get a further speed improvement at around 10Gbps. The last part is being implemented to check whether the system can distribute clock and data even on higher transfer rates. An alternative, more appropriate, DE4 FPGA development board is also used for the last experiment part apart from Cyclone V.
The system is implemented by Altera Cyclone V GT board, Altera DE4 board, Terasic Ethernet-HSMC board, Terasic SFP-HSMC card and Dual XAUI to SFP+ HSMC card. The design is built and programmed by the Quartus II 13.1 and Nios II Software Build Tool. Some embedded tools of Quartus for test and verification are used including Transceiver Toolkit and SignalTap II Logic Analyzer.

Handledare: Leif Gustafsson
Ämnesgranskare: Pawel Marciniewski
Examinator: Arnold Neville Pears
IT 16046
Tryckt av: Reprocentralen ITC

Acknowledgments

We would like to express our deepest appreciation to our supervisor Leif Gustafsson and our reviewer Pawel Marciniewski for their continuous guidance and persistent help during this thesis until the very end of it.

Also, many thanks go to our friends and colleagues for their useful feedback and help, during our thesis.

Last but not least, we would like to thank our families for their support and trust during our whole studying career. Without them, we would not have reached this point.

Contents

Chapter 1 1

1.1 Introduction 1

1.2 Motivation 1

1.3 Objectives 2

Chapter 2 Background 3

2.1 OSI model 4

2.1.1 Physical Layer 4

2.1.2 Data Link Layer 5

2.1.3 Network Layer 5

2.2 Ethernet Protocol 5

2.3 Optical Fibers 7

2.4 GBT 9

2.4.1 GBT-FPGA Core 10

2.4.2 GBT-FPGA Block Diagram 11

2.5 MicroPOD 13

2.6 Quantum Dots 13

Chapter 3 Hardware and Software 9

3.1 Hardware 9

3.1.1 Cyclone V GT 9

3.1.1.1 Overview 9

3.1.1.2 FPGA 10

3.1.1.3 Clocking 10

3.1.1.4 Gigabit Ethernet PHY 11

3.1.1.5 HSMC 11

3.1.1.6 Transceivers PLLs 12

3.1.2 DE4 Development Board 13

3.1.2.1 Overview 13

3.1.2.2 FPGA 14

3.1.2.3 Clocking 14

3.1.2.4 Gigabit Ethernet PHY 14

3.1.2.5 HSMC 14

3.1.2.6 Transceiver PLLs 15

3.1.3 Device Comparison 16

3.1.4 Ethernet HSMC Card and Marvel 88E1111 Controller 17

3.1.5 SFP-HSMC Card 18
3.1.6 DUAL XAUI to SFP+ HSMC Board 20
3.2 Software 22
Chapter 4 Transceivers Datapath 23
4.1 Standard Transceiver Datapath 23
4.1.1 Transmitter (TX) 23
4.1.1.1 Phase Compensation FIFO 23
4.1.1.2 Byte Serializer 24
4.1.1.3 8b/10b Encoder 24
4.1.1.4 Bit Serializer (SerDes) 28
4.1.2 Receiver (RX) 28
4.1.2.1 Receiver CDR 28
4.1.2.2 Bit Deserializer (SerDes) 28
4.1.2.3 Word Aligner 29
4.1.2.4 Deskew FIFO 29
4.1.2.5 Rate Matcher 30
4.1.2.6 8b/10b Decoder 30
4.1.2.7 Byte Deserializer 30
4.1.2.8 Byte Ordering 30
4.1.2.9 Phase Compensation FIFO 31
4.2 10G Transceiver Datapath 32
4.2.1 Transmitter (TX) 32
4.2.1.1 TX FIFO 32
4.2.1.2 Frame Generator 32
4.2.1.3 64B/66B Encoder 32
4.2.1.4 CRC-32 Generator 33
4.2.1.5 Scrambler 34
4.2.1.6 Disparity Generator 34
4.2.1.7 Transmitter Gearbox 35
4.2.2 Receiver (RX) 35
4.2.2.1 Receiver Gearbox 35
4.2.2.2 Block Synchronizer 35
4.2.2.3 Disparity Checker 35
4.2.2.4 Descrambler 36
4.2.2.5 Frame Synchronizer 36
4.2.2.6 BER Monitor 36

4.2.2.7 64B/66B Decoder 36
4.2.2.8 CRC-32 Checker 36
4.2.2.9 RX FIFO 36
Chapter 5 Implementation and Results 37
5.1 Ethernet Loopback Test 37
5.1.1 Ethernet Frame 37
5.1.2 RGMII 37
5.1.3 Triple Speed Ethernet (TSE) IP Core 39
5.1.4 Implementation 40
5.1.4.1 QSYS-Nios II subsystem 40
5.1.4.2 IP Generation and Customization 42
5.1.4.3 VHDL code 44
5.1.4.4 C code 45
5.1.5 Results 46
5.2 SFP Loopback Test 47
5.2.1 SFP Modules 47
5.2.2 IP cores 48
5.2.2.1 Custom PHY IP 49
5.2.3 Implementation 51
5.2.3.1 Transceiver Toolkit 51
5.2.3.2 Raw Data Loopback Test 55
5.2.4 Results 56
5.3 XAUI Loopback Test 58
5.3.1 10 Gigabit Media-Independent Interface 58
5.3.2 10 Gigabit Attachment Unit Interface 59
5.3.2.1 Signal levels 60
5.3.2.2 Amplitude and swing 60
5.3.2.3 Functional Specifications 61
5.3.3 XAUI PHY IP Core 63
5.3.3.1 Block Diagram 63
5.3.3.2 Transceiver Datapath 65
5.3.4 Implementation 67
5.3.4.1 IP Generation and Customization 69
5.3.5 Results 72
5.4 GBT Loopback Test 74
5.4.1 GBT-FPGA IP 74

5.4.2 Implementation 76
5.4.3 Results 76
Chapter 6 Conclusion and Future work 79
6.1 Conclusion 79
6.2 Future work 81
Split of the work 82
References 83

List of Figures

Figure 1: Simplified Data Network 3
Figure 2: OSI Model 4
Figure 3: Gigabit Ethernet PHY Division 6
Figure 4: Snell's law 7
Figure 5: Light wave propagation inside optical fibers [3] 8
Figure 6: GBT Link Architecture 9
Figure 7: GBT-Frame encoding frame 10
Figure 8: 8b10b encoding frame 10
Figure 9: Wide-Bus encoding frame 11
Figure 10: GBT Bank simplified block diagram 11
Figure 11: GBT Link simplified block diagram 11
Figure 12: PCIe40 board 12
Figure 13: MicroPOD attached to FPGA chip 13
Figure 14: MicroPOD relative size 13
Figure 15: Quantum Dot 8
Figure 16: Overview of the Cyclone V GT FPGA Development Board Features 9
Figure 17: Cyclone V GT FPGA Development Board Block Diagram 10
Figure 18: RGMII Interface between FPGA (MAC) and Marvell 88E1111 PHY 11
Figure 19: HSMC signal and bank diagram 11
Figure 20: The schematics of the Bank 1 of a HSMC port 12
Figure 21: Simplified layout of the fPLLs in the transceiver channels 12
Figure 22: Overview of the DE4 Development Board Features 13
Figure 23: DE4 FPGA Development Board Block Diagram 13
Figure 24: HSMC signal and bank diagram 15
Figure 25: The schematics of the Bank 1 of a HSMC port a 15
Figure 26: The schematics of the Bank 1 of a HSMC port b 15
Figure 27: Simplified layout of the ATX PLLs in the transceiver channels 16
Figure 28: Ethernet-HSMC Daughterboard 17
Figure 29: The block diagram of the HSMC-NET card 17
Figure 30: Marvell 88E1111 Device used in Copper Application 18
Figure 31: The block diagram of the EEPROM and HSMC connector 18
Figure 32: SFP-HSMC Daughterboard 18
Figure 33: SFP HSMC Card Block Diagram 19
Figure 34: DUAL XAUI Daughterboard 20
Figure 35: Block diagram of the Dual XAUI to SFP+ HSMC board 21
Figure 36: BCM8727 Signal channel block diagram 22
Figure 37: Transmitter Channel PCS and PMA 23
Figure 38: TX Phase Compensation FIFO Instance 23
Figure 39: Byte Serializer Block Diagram 24
Figure 40: Effect of AC coupled channel on the DC component of a signal 25
Figure 41: Bit ordering 25
Figure 42: Bit Serializer Block Diagram 28
Figure 43: Receiver Channel PCS and PMA 28
Figure 44: Bit Deserializer Block Diagram 29
Figure 45: Byte Ordering Diagram 30

Figure 46: RX Phase Compensation FIFO Instance 31
Figure 47: 10G Transmitter Channel Datapath 32
Figure 48: Transmitter FIFO 32
Figure 49: The structure of a 16-bit LFSR 33
Figure 50: Hardware implementation of CRC-16 34
Figure 51: 10G Receiver Channel Datapath 35
Figure 52: Ethernet Frame 37
Figure 53: MII Interface 38
Figure 54: GMII Interface 38
Figure 55: RGMII Signal Interface Diagram 39
Figure 56: Triple Speed Ethernet IP Block Diagram 39
Figure 57: Loopback System Block Diagram 40
Figure 58: Triple Speed Ethernet Core Configurations 1 42
Figure 59: Triple Speed Ethernet Core Configurations 2 42
Figure 60: Triple Speed Ethernet Core MAC options 43
Figure 61: Triple Speed Ethernet Core FIFO options 43
Figure 62: Triple Speed Ethernet Core Timestamp options 44
Figure 63: Triple Speed Ethernet Core PCS/Transceiver options 44
Figure 64: Nios II EDS embedded console instance 46
Figure 65: Pinouts on the PCB 47
Figure 66: Block Diagram of Custom PHY IP core 49
Figure 67: Snippet of Custom PHY Settings 1 51
Figure 68: Snippet of Custom PHY Settings 2 51
Figure 69: Qsys SFP Loopback System Block Diagram 52
Figure 70: Basic page for Transceiver Link 53
Figure 71: Test the link communication at 5000Mbps data rate 53
Figure 72: Link communication after plugging out the optical fiber 54
Figure 73: Link communication after injecting error 54
Figure 74: Raw data SFP Loopback System Block Diagram 55
Figure 75: Custom PHY Settings 1 56
Figure 76: Custom PHY Settings 2 56
Figure 77: SignalTap of 16-bit interface at 3,5Gb/s data rate 57
Figure 78: SignalTap of 32-bit interface at 5Gb/s data rate 57
Figure 79: XGMII 58
Figure 80: XGMII data transfer 58
Figure 81: XGMII to XAUI at the XGXS 59
Figure 82: XAUI and XGXS relationship to the ISO/IEC Open Systems Interconnection (OSI) reference model and the IEEE 802.3 CSMA/CD LAN model 60
Figure 83: Output voltage limits and definitions [Li<P> and Li<N> are the positive and negative sides of the differential signal pair for lane i (i = 0, 1, 2, 3)] 60
Figure 84: XAUI PHY IP Core 63
Figure 85: Soft XAUI vs. Hard XAUI 64
Figure 86: XAUI Interface signals 64
Figure 87: XAUI PHY data transfer 65
Figure 88: XAUI PHY Datapath 65
Figure 89: Top level block diagram 67
Figure 90: Data Interface 68
Figure 91: Byte 0 Start of Frame Transmission Example 68

Figure 92: Byte 5 Start of Frame Transmission Example 69
Figure 93: Formed message 69
Figure 94: Cyclone V GT XAUI IP general options 70
Figure 95: Cyclone V GT XAUI IP analog options 70
Figure 96: Cyclone V GT XAUI IP advanced options 70
Figure 97: Stratix IV GX XAUI IP general options 71
Figure 98: Stratix IV GX XAUI IP analog options 71
Figure 99: Stratix IV GX XAUI IP advanced options 72
Figure 100: Transmission instance 1 72
Figure 101: Transmission instance 2 72
Figure 102: Unplugged fiber cable instance 73
Figure 103: Synchronization process instance 73
Figure 104: GBT Tx simplified block diagram 74
Figure 105: GBT Rx simplified block diagram 74
Figure 106: GBT Tx/Rx detailed block diagram 75
Figure 107: GBT loopback test block diagram 76
Figure 108: GBT transmitted data instance 77
Figure 109: GBT received data instance 77

List of Tables

Table 1: Ethernet frame packet 5
Table 2: Gigabit Ethernet Varieties 7
Table 3: Standard GBT vs. Latency Optimized GBT 10
Table 4: Cyclone V GT Resources Distribution 10
Table 5: Cyclone V GT On-Board Oscillators 11
Table 6: Transmitter PLL Capability and Availability 12
Table 7: DE4 Resources Distribution 14
Table 8: DE4 On-Board Oscillators 14
Table 9: 8B/10B Mapping Tables 27
Table 10: Generator Polynomials of some CRC codes 34
Table 11: Receiver Characteristics 61
Table 12: XAUI special symbols 61

Abbreviations and Symbols

Abbreviations

AC: Alternative Current

ALMs: Adaptive Logic Modules

API: Application Programming Interface

ASIC: Application-Specific Integrated Circuit

ATX PLL: Auxiliary PLL

BER: Bit Error Rate

CDR: Clock Data Recovery

CMOS: Complementary Metal-Oxide Semiconductor

CMU PLL: Clock Multiplier PLL

CRC: Cyclic Redundancy Code

DAQ: Data Acquisition

DC: Direct Current

DDIO: Double Data rate I/O

DDR: Double Data Rate

DMA: Direct Memory Access

DSP: Digital Signal Processor

EDS: Embedded Design Suite

EEPROM: Electrically Erasable Programmable Read-Only Memory

EMI: Electromagnetic Interference

FEC: Forward Error Correction

FIFO: First In First Out

FPGA: Field Programmable Gate Array

fPLL: fractional PLL

GbE: Gigabit Ethernet

GBIC: Gigabit Interface Converter

GMII: Gigabit Media Independent Interface

HSMC: High Speed Mezzanine Card

IDE: Integrated Development Environment

I/O: Input Output

I2C: Inter-Integrated Circuit

IEEE: Institute of Electrical and Electronics Engineers

IP: Intellectual Property

IPG: Inter-Packet Gap

JTAG: Joint Test Action Group

LEs: Logic Elements

LED: Light-Emitting Diode

LFRS: Linear Feedback Shift Register

LPM: Library of Parameterized Modules

LSB: Least Significant Bit

LVDS: Low-Voltage Differential Signaling

MAC: Media Access Control

MDC: Management Data Clock

MDIO: Management Data I/O

MGT: Multi-Gigabit Transceiver

MII: Media Independent Interface

MSB: Most Significant Bit

OSI: Open System Interconnection

PCB: Printed Circuit Board

PCI: Peripheral Component Interconnect

PCS: Physical Coding Sublayer

PHY: Physical Layer

PLL: Phase Lock Loop

PMA: Physical Medium Attachment

PMD: Physical Medium Dependent

PPM: Parts Per Million

RD: Running Disparity

RGMII: Reduced Gigabit Media Independent Interface

RJ-45: Registered Jack-45

RX: Receiver

SC: Slow Control

SDR: Single Data Rate

SerDeS: Serializer/Deserializer

SFI: SerDes Framer Interface

SFP: Small Form-factor Pluggable

SGDMA: Scatter Gather DMA

SGMII: Serial Media Independent Interface

SMA: SubMiniature version A

SOP: Start Of Package

SPI: Serial Peripheral Interface

TCL: Tool Command Language

TFC: Timing and Fast Control

TSE: Triple Speed Ethernet

TTC: Timing and Trigger Control

TX: Transmitter

UART: Universal Asynchronous Receiver/Transmitter

VHDL: VHSIC Hardware Description Language

VHSIC: Very High Speed Integrated Circuit

XAUI: 10 gigabit Attachment Unit Interface

XCVR: Transceiver

XGMII: 10 Gigabit Media Independent Interface

XGXS: XGMII eXtender Sublayer

XOR: Exclusive OR

Symbols

k: Kilo

dB: decibel

dB/km: decibel per kilometer

G: Giga

Gb/s: Gigabit per second

GBd: Giga Baud

Gbps: Giga bit per second

Hz: Hertz

kb: Kilobit

kHz: Kilohertz

km: Kilometer

m: meter

Mb: Megabit

Mb/s: Megabit per second

Mbit/s: Megabit per second

Mbps: Megabit per second

MHz: Megahertz

ps: picoseconds

V: Volts

Chapter 1

1.1 Introduction

Millions of experiments take place every data by pioneer researchers and engineers. Billions of devices gather, process and forward experimental (or not) data. Terabyte of data is being transmitted to a centralized unit to be processed or even stored. Every day the demands of networks, which host all those devices and convey all that huge amount of data, grow in an almost exponential rate. New challenges pop up making scientist work on more difficult and important problems.

More and more data needed to be transferred in data networks. Faster transfer rate needs emerge. The number of connected devices and its requirements about synchronization grow. The environmental conditions of the experiments narrow the options. For all those reasons-obstacles above, high end designs and systems are used to cover all the demands. Specific protocols are used to transfer data in high rates and special devices are applied as nodes to the network. Most of the data is time-dependent, thus timing need to be extremely accurately distributed or computed in the network.

This is the point where the trade-off arises over the cost and the appropriateness of those systems. Besides the practical issues, the cost of those systems is another problem. Hence, there are scientists, who try to minimize the cost and simultaneously keep the reliability and the performance of the designs at highest level.

1.2 Motivation

The TFC system that currently works for CERN's front end equipment is built by a PCIe40 board [34]. But for assessment and testing in laboratories or simplified applications in other places, the PCIe40 board is expensive and comparatively wasteful on its hardware resources and capabilities. Hence, it is more than challenging trying to replace the PCIe40 board with a low-end or a mid-end FPGA to achieve the requirements of its performance as a demonstrator.

Nowadays, programmable logic devices like FPGAs are more and more widely used in various areas as in ASIC prototyping, computer vision, digital processing etc. In theory, an FPGA can solve any computable problem. This fact has been proven by the truth that FPGA can either be applied to implement a soft microprocessor or comes with an integrated on-chip hard microprocessor.

Besides, optical fibers have already become an essential medium for long-distance wired network systems. It is common that some large companies, organizations or populous cities use stable and safe networks made of optical fibers to exchange huge amount of data at

extremely high speed. Optical fibers enable data to be transmitted in the form of light through repeated reflections which accelerates the growth of network speed.

As Embedded Systems students, the authors hold strong interest and excitement to combine these two popular techniques together as a good opportunity to enrich corresponding theoretic knowledge and practice implementing abilities. For the cooperation of Altera FPGA devices and optical fibers, Altera supplies some useful IPs as handful tools to liberate designers from basic VHDL coding but focus more on the configurations and settings of those IPs in order to get best result.

In summary, this thesis can be a good stepping-stone to have a deeper research on high speed transmission techniques with the aid of FPGAs in the future as well as an acceptable starting point for the followers who are also interested in this field to read, refer and improve.

1.3 Objectives

The ultimate goal of this project is to produce the tools and put them together in order to create a Timing and Fast Control Demonstrator. The Demonstrator unit consists of two FPGA boards connected to each other through high speed Gigabit Fiber Optical cables communicating over a GBT interface protocol [35]. An alternative would be implementing a loopback test using the same protocol in a single device. Yet, before pointing to the final target, the thesis is distributed in smaller goals:

1. First goal is to get familiar with our main device Cyclone V GT board, by checking Reference Manual and the User's Guide. That part also embeds a simple device testing by making clear the programming procedure for our board.
2. Next objective is connecting a simple Ethernet HSMC board to the main board and implementing a loopback test. Despite of the seeming simplicity of this part, its importance is really high, because of the HSMC port usage for the first time.
3. Third goal, after succeeding with the Ethernet board, is to replace it with an SFP HSMC board and implementing a standard loopback test. The plan is to generate a bunch of data to be sent through the Fiber Optical cables pointing to a comparator implemented at the same design, in order to check the integrity and validity of the sent information.
4. Fourth part consists of an XAUI to SFP+ HSMC board usage. The same design as before is going to be used to test the functionality of this faster card. Loopback testing still stays simple, by sending unframed data from one point to the other comparing the sent and received data. Since the only seeming difference to the previous step is the new HSMC card, the challenging background part is the synchronization requirements, which become more demanding on higher speeds.
5. Last implementation part is the most ambitious part and could be considered as an extra future step. In this case testing data will be formed and sent using the GBT interface protocol, from one device to another Cyclone V GT or at the same device using loopback test again. The critical and important point of this test is to achieve communication under those circumstances between two devices/ transceiver modules running individual system clocks.

Chapter 2 Background

The first network of connected systems that exchange data and communicate each other was established more than half a century ago. From this point and on systems that connect modules, which are capable of gathering information, process data and distribute it through a wired or wireless medium, keep developing and evolving. The variety of every network's demands has led to the establishment of many different data system networks. Depending on the size or the importance of data, or depending on the distance between the connected nodes, different network protocols and interfaces have been introduced. Three of the major and widely used network systems are: Data Acquisition Systems, Timing Control Systems and Slow Control Systems. All network systems serve different purposes and follow exact specifications; however, the same fundamental principle is applied in any transceiver of those systems. Even if it is time-critical systems, (where the latency has to be accurate), or data-critical systems, (where data has to be conveyed successfully as many times as possible), or rate-critical systems (where the data transfer rate has to remain steady) the low level parts remains similar.

Figure 1: Simplified Data Network

Nowadays, wired networks have achieved significantly faster data rates and they behave more stable than over wireless networks in long-distance transmission. This is the reason why a wired network is preferred in large scale experiments, where data needs to be transferred or in industries. On the other hand, wireless systems are mainly used for public-related usage or small and infrequently information transfer.

2.1 OSI model

The nodes of a network usually are not all the same or they do not share the exact same architecture. In order to achieve a successful communication over every possible node, there is a common model. This is the Open System Interconnection model (OSI), which determines and standardizes the communication over data systems regardless their structure. It targets interoperability over standard protocols. It is divided in seven layers, starting from the lower level of Physical layer up to the higher and more abstract of Application layer.

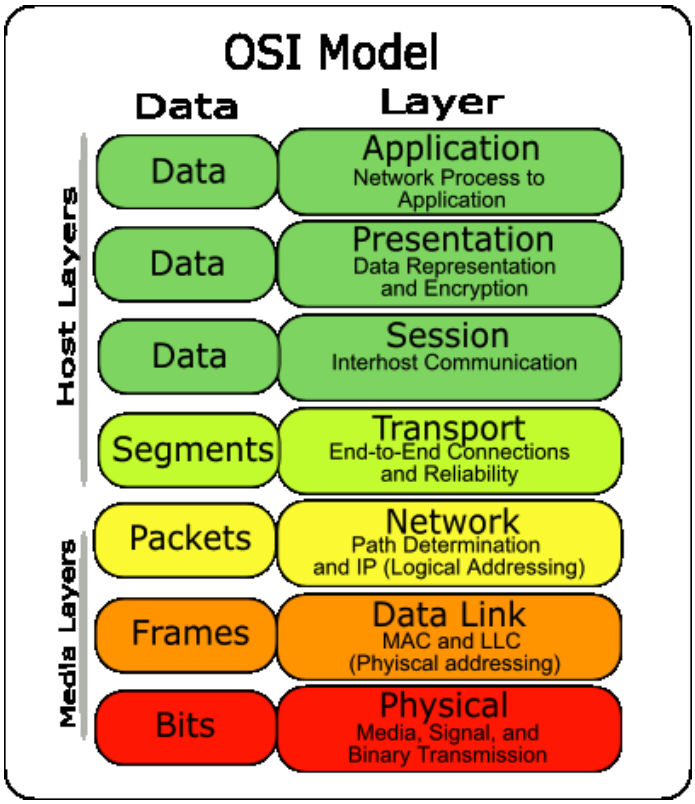

Figure 2: OSI Model

The purpose of our thesis is to test how feasible are two FPGAs for sending and receiving data at high speed data rates. The main solution to challenge is a successful loopback test, which simulates the transmission and reception of data to and from other nodes. In order to give the appropriate background information for the implementation part [ch. 5], it is needed to understand deeply the first three layers only.

2.1.1 Physical Layer

The physical Layer is the part, which is just before the physical transmission medium. It defines both electrical and physical specifications of the connection. It sets the relation between the medium and the device like layout pins, line impedance, signal timing and frequency. The data being handled by the Physical Layer is mostly raw and unstructured data, since it is directly connected to the medium. Finally, it is responsible for determining either half or duplex mode for data transmission.

2.1.2 Data Link Layer

The Data Link Layer is working as a complement of the Physical Layer. Errors that may have happened in the previous layer are being detected and corrected (if possible). The procedure to start and terminate a connection of two devices following a given protocol is part of this layer. According to IEEE 802 specifications the Data Link Layer can be considered as two sublayers:

- Media Access Control: Defining and controlling the way devices gain access to use the medium.
- Logical Link Control: Encapsulating and Synchronizing frames according to the protocols and error checking as well.

2.1.3 Network Layer

The third layer constitutes the first and simplest "data-translator" of a data network. It is the part which processes the message in order to find out if it is targeting the specific node or someone else. So, it decodes the destination part of a transmitted message and checks if it has reached its destination or if it is needed to be retransmitted. That part was merely used in the first presented loopback implementation.

2.2 Ethernet Protocol

Ethernet protocol is one of the oldest (1983), most important and most commonly used networking technologies for Local Area Networks. Starting with almost 3 Mbps data rate, it is expected to reach 10^5 times faster value at 400 Gbps by late 2017. As it will be described later in the implementations part, Ethernet is the protocol that is mainly chosen in our designs. Data streams in Ethernet communications is a concatenation of an addressing part, a pure information part and error checking of data like table 1. As per the OSI model, Ethernet provides services up to and including the Data Link Layer.

	Size
Preamble	7 octets
Start of frame delimiter	1 octet
MAC destination	6 octets
MAC source	6 octets
802.1Q tag (optional)	(4 octets)
Ethernet (Ethernet II) or length (IEEE 802.3)	2 octets
Payload	46 – 1500 octets
Frame check sequence (32-bit CRC)	4 octets
Interpacket gap	12 octets

Table 1: Ethernet frame packet

In table 1, a typical structure of an Ethernet frame packet is shown [35]. The preamble consists of a seven-byte long pattern of alternating 1 and 0 bits which allows devices on the network to easily synchronize their receiver clocks. The SFD (Start of Frame Delimiter) is the

one-byte value that marks the end of the preamble. The following destination and source MAC addresses, the Ethernet type (or length) field and an IEEE 802.1Q tag compose the frame header. The Ethernet type (or length) field is two-byte long that can be used for two purposes. One purpose is to make the field indicate the size of the payload in octets. Another one is to mean the type of the Ethernet. The IEEE 802.1Q tag, if present, indicates virtual LAN membership and IEEE 802.1p priority. The payload's size ranges from 42 octets to 1500 octets. The frame check sequence is a four-byte cyclic redundancy check which allows detection of corrupted data within the entire frame on the receiver side. Interpacket gap is idle time between packets.

As Ethernet protocol is evolving reaching higher transfer speed new protocol variations are introduced having suitable specifications according to the purpose of every variation. One common part that most of the Gigabit Ethernet versions share is the division of PHY in tree sublayers. Those parts are the Physical Coding Sublayer (PCS), the Physical Medium Attachment Sublayer (PMA) and the Physical Medium Dependent Sublayer (PMD).

The last two sublayers are closely related to the chosen transmission medium as it is clearly understandable by their names. Several mediums can be used to convey the data from one node to another using the same data rate. In this case, the PCS part remains the same in every implementation, while the PMA and the PMD have to get conformed to each medium. Basically, PMA and PMD define the details of transmitted/received bits on a physical medium, like bit timing, signal encoding, medium interaction.

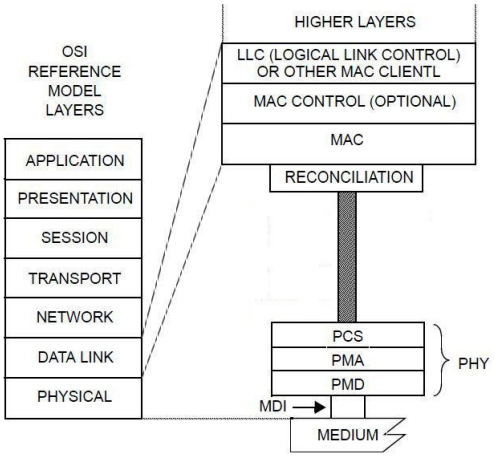

Figure 3: Gigabit Ethernet PHY Division

Since the purpose of our thesis is to design successfully loopback test systems over increasing data rates, the PCS configuration is a considerable step in our design. The Coding sublayer is the point where common procedures like synchronization and alignment, rate difference compensation, data coding/encoding, data scrambling/descrambling, etc. take place. All the used procedures and implemented mechanisms in our designs will be extensively described in part [ch. 5].

Eventually the processed data, which is "filtered" over optimal algorithms and encoding or scrambled in a special way, will reach its destination over the defined medium. Engineers have also devoted much time in evolving the corresponding mediums apart from network protocols.

As it is expected Ethernet protocol has developed different protocol versions for its Gigabit variation over optical fibers, twisted pair cables and shielded balanced copper cables.

Name	Medium	Specified Distance
1000BASE-CX	Shielded balanced copper cable	25 m
1000BASE-KX	Copper backplane	1 m
1000BASE-SX	Multi-mode fiber	220 m...550 m
1000BASE-LX	Multi-mode fiber	550 m
1000BASE-LX	Single-mode fiber	5 km
1000BASE-LX10	Single-mode fiber	10 km
1000BASE-EX	Single-mode fiber	~ 40 km
1000BASE-ZX	Single-mode fiber	~ 70 km
1000BASE-BX10	Single-mode fiber	10 km
1000BASE-T	Twisted-pair cabling	100 m
1000BASE-TX	Twisted-pair cabling	100 m

Table 2: Gigabit Ethernet Varieties

Taking into consideration the Medium and the Distance columns of table 2, fiber optics are the only medium capable of establishing a widely spread network where its nodes are separated by a distance over 1000 m. Contemporary wired networks prefer optical fibers against any type of cables and wires as medium.

2.3 Optical Fibers

Optical Fibers enable data to be transmitted in the form of light through repeated reflections. The idea of optical data transmission was demonstrated already in 1840s by Daniel Colladon and Jacques Babinet. During 1970s, optical fibers could be implemented with attenuation less than 20 dB/km [1].

Nowadays, optical fibers are able to achieve higher bandwidths and lower attenuation than copper cables [2]. One of the few disadvantages of this technique is the comparatively high price. However, if optical fibers are applied for long-distance communication, they will still be competitive financially since much fewer signal repeaters are needed.

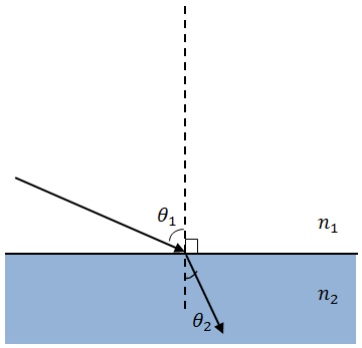

Figure 4: Snell's law

Every sort of material has a characteristic refractive index $n = \frac{c}{v}$, where v is the velocity of light in this specific material and c is the velocity of light in vacuum. Since light spreads

fastest in vacuum, n is greater than 1. According to Snell's law, the equation $\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}$ governs the angles of incidence θ_1 and refraction θ_2 of a light wave when it hits a boundary between two materials with different refractive indices n_1 and n_2. Thus, if the incidence angle is larger than a critical angle $\theta_c = \cos^{-1}(\frac{n_2}{n_1})$, where light enters a material with refractive index n_2 from another material with the one of n_1, and $n_2 > n_1$, the light wave will propagate without loss. This phenomenon is called total internal reflection. To make use of this characteristic, modern optical fibers consist of the cores with high refractive index, surrounded by lower refractive index claddings. The claddings can also keep the internal cores in a pure lightless environment as well as increase the cables' strength and lifetime. The illustration of light propagation inside an optical fiber cable is listed in figure 5.

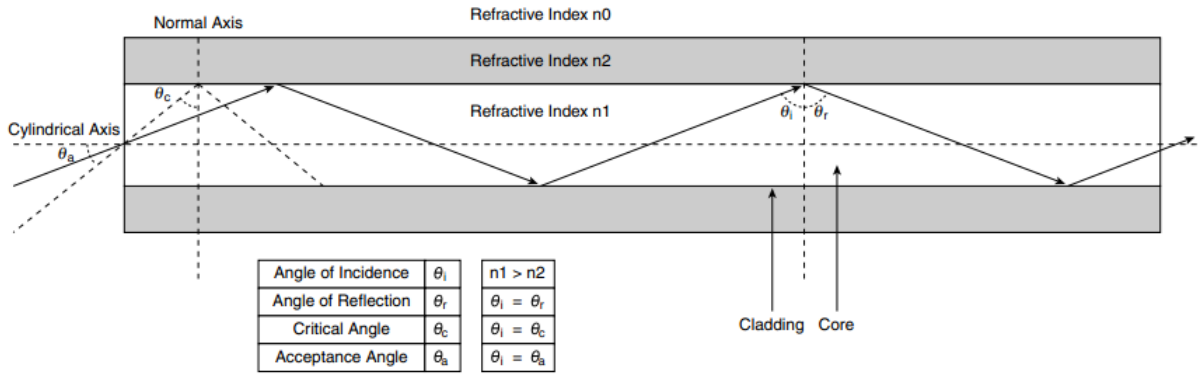

Figure 5: Light wave propagation inside optical fibers [3]

Light must enter the core from the air at a specific angle less than an entity known as the acceptance angle $\theta_a = \sin^{-1}(\frac{n_1}{n_0} \cdot \sin(\theta_c))$. In this formula, n_0 is the refractive index of air which equals to 1. This angle is measured from the cylindrical axis of the core. Lately, fiber optics have more and more application systems, where they are used to establish data connections.

So, fibers are preferred over metal wires because signals manage to travel along them with lesser amounts of loss. Another advantage of fiber optics is their tolerance to electromagnetic interference, a quite problematic characteristic of metal wires [4]. Additionally, any losses that come of joining them are minimum compared to extension joints made of any other type of cables. However, it is not that simple procedure; it requires perfect alignment and careful cleaving of the connected fiber cores. For that reason, there are special mechanisms, which are used to achieve demanding fiber optics connections. Light propagation, as it was described above, helps optical fiber to become optimal medium for long distances since its propagation comes with slight attenuation compared to electrical cables. Another characteristic of optical fibers is that there is no cross-talk phenomenon in different cables and of course no pick up of environmental noise. Data security and integrity are two aspects ensured by optical links. Wiretapping without detection is very difficult for optical connections in comparison to copper. It is also worth pointing out that simple fiber optics are not affected in high voltage environments, since they do not conduct electricity as metal wires do.

Summarizing the advantages of fiber optics over copper wiring will make clearer the reason why they are used in more and more applications and why we implemented our speed-demanding test using optical links.

- Higher Bandwidth
- Low attenuation loss over long distances
- Electromagnetic interference immune
- Electrical insulator
- Security of information conveyed through fibers.

Local exchange carriers (LECs) apply optical fibers to carry plain old telephone services (POTS) between central switches at local levels and some neighborhoods or individual' homes [3]. International companies need stable, reliable and secure systems to transfer high amount of data and essential financial information from servers to desktop terminals all over the world. Cable television corporations also use optical fibers for the delivery of digital video and music services. Transportation system components, like intelligent traffic lights and unmanned tollbooths are also driven by optical fiber telemetry systems. In researching areas, optical fiber techniques are also important tools [5]. In most modern telemedicine devices, optical fiber systems are used to transmit digital diagnostic images. Some remote-control medical assistance robots also need optical fibers as communication media to be controlled by experts who are not present during the operation.

2.4 GBT

The GBT project is a research project to implement a high speed bidirectional link to be used in a high-energy physics experiment. One of the design's specialties is that the actual device has to be radiation tolerant because of the experiment's environment. Another specialty is that the data/information transferred through this link has to follow a specific interface also called GBT.

The GBT chip has already been implemented for experimental purposes at CERN [36]. However, the main chips that host the design and implement the protocol are high-end, quite expensive FPGAs boards. That is the reason why the research about the GBT continues, trying to find a better value for money solution. The importance and the utility of those attempts are high due to the fact that experiments like the one running at CERN may be part of experiments in other research centers or labs. Providing scientist with an affordable solution, which offers equal characteristics and performance levels, is quite important.

The GBT link operates at 4.8 Gbps and it is divided in three parts; a part for Timing and Trigger Control (TTC), one for Data Acquisition (DAQ) and the last for Slow Control (SC) information. Practically, the parts are not divided, yet they are merged in the same single optical link.

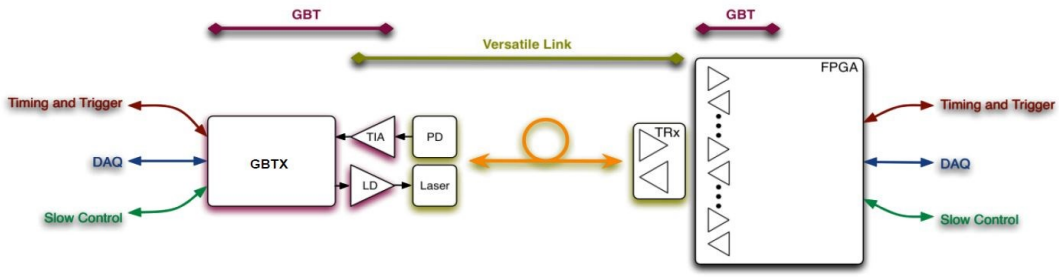

Figure 6: GBT Link Architecture

2.4.1 GBT-FPGA Core

CERN provides the so called GBT-FPGA core, which is a VHDL based design, including custom code parts as well as custom IPs. By now there are only two different link types supported.

- **Standard**, targeting non time critical systems, like DAQ.
- **Latency Optimized**, ensuring low, fixed and deterministic latency of clock and data, targeting time critical systems like TTC.

	Standard	**Latency Optimized**
Latency	Non Fixed, Higher, Non Deterministic	Fixed, Low, Deterministic
Logic Resources Utilization	Low	Low
Clocking Resources Utilization	Low	**High**
Clock Domain Crossing	Do not care	**Critical**
Implementation	Simple	Complex

Table 3: Standard GBT vs. Latency Optimized GBT

Another parameter offered is the encoding type, which defines the general purpose of the link.

- **GBT-Frame**

This frame structure follows Reed-Solomon method [37], capable of correcting bursts of bit errors. This scheme can be used for DAQ, TTC and SC systems.

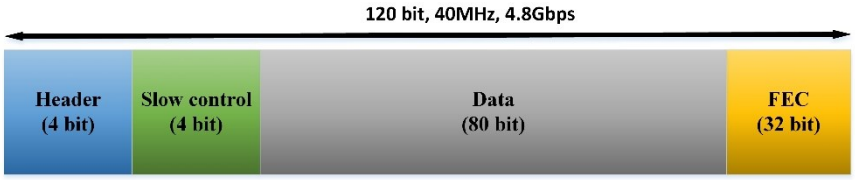

Figure 7: GBT-Frame encoding frame

- **8b10b**

The 8b10b has no error correction mechanism, so it provides the user with 8 extra bits to be used. In this case, this encoding type is useful only in DAQ and SC systems.

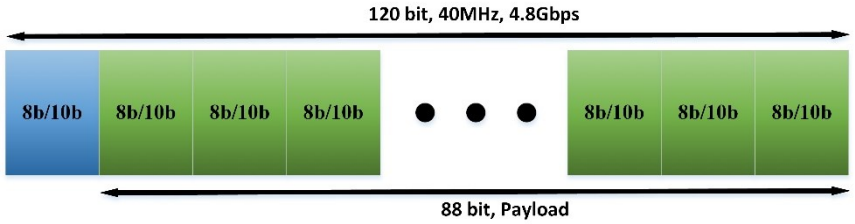

Figure 8: 8b10b encoding frame

- **Wide-Bus**

The last encoding type adopts the GBT-Frame format. However, the FEC part is also used for user data. As a result, it can be used only for DAQ and SC systems.

The top level of the GBT-FPGA core is called GBT Bank, since it is parameterized and able to embed more than one GBT-Links,

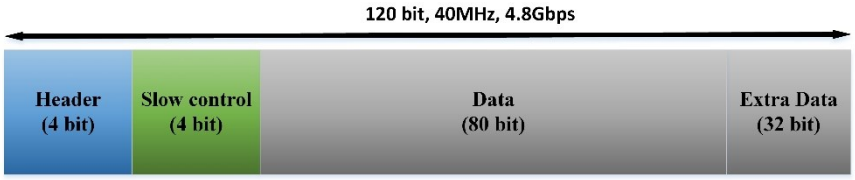

Figure 9: Wide-Bus encoding frame

2.4.2 GBT-FPGA Block Diagram

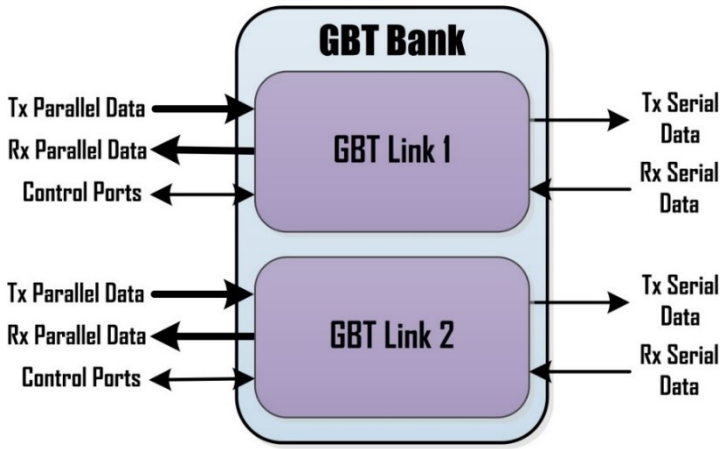

Figure 10: GBT Bank simplified block diagram

As it is depicted in figure 10 above on the simplified version of a GBT Bank, the I/O of the modules look extremely similar to a common high speed serial transceiver for FPGAs, which is described in part [ch. 4] later on. Setting the corresponding parameter, the number of links included in a GBT Bank varies from 1 to 3.

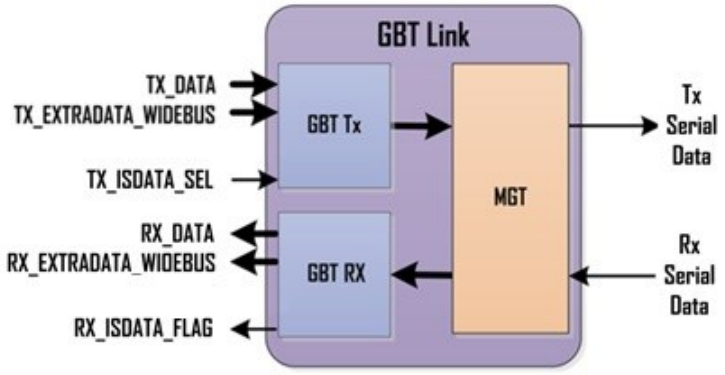

Figure 11: GBT Link simplified block diagram

Analyzing the simple version of a link, the main three components are clearly proposed. Indeed, there is a parallel transmitter (GBT Tx) and a parallel receiver (GBT Rx). Both of them exchange data with the Multi-Gigabit Transceiver (MGT) unit. The latter unit is responsible for serializing and deserializng the data, while the previous two scramble and encode the transmitted data and align, decode and, descramble the received data respectively.

More details and deeper analysis on the datapath of the GBT Link are presented on its implementation part [ch. 5.4].

An important part which needs careful treatment and control before any long-term design is the dependency of the clock phase with temperature. Clock drift caused by the rise of the temperature needs to be taken into consideration for any experiment's specifications.

Finally, it needs to be mentioned that the GBT-FPGA core is available/verified for a small number of devices apart from the PCIe40 board [34], which is originally used at CERN.

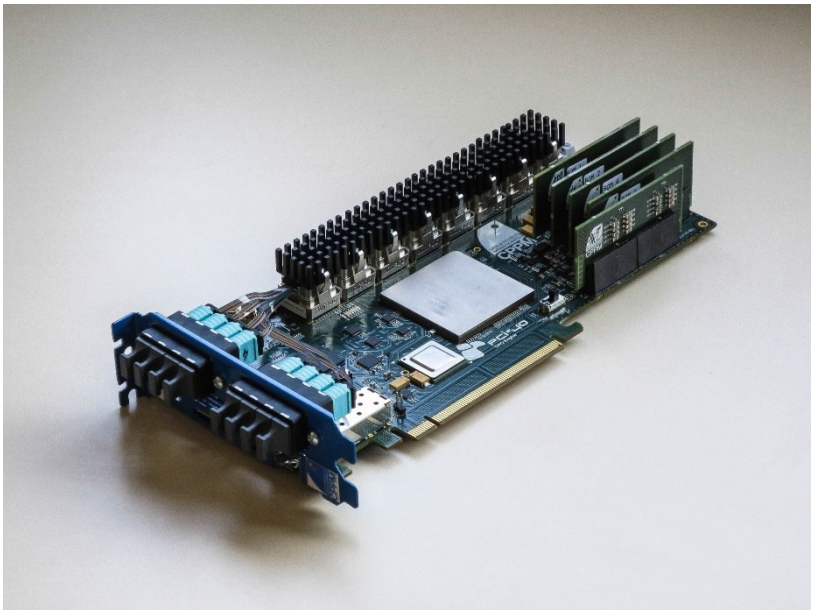

Figure 12: PCIe40 board

The PCIe40 board is designed and manufactured for special requirements at CERN. However, its price (~150000 €) is extremely high for custom experiments in smaller laboratories. That is why, GBT-FPGA core is officially supported by other devices too. Depending on the FPGA vendor and the used IDE the core settings and options differ. The more power and specialized the device the better. However, what follows the previous sentence is the better the device the more expensive. So, CERN has recruited a team of scientists to implement the design and to verified the GBT-FPGA core in more and more devices.

2.5 MicroPOD

Bandwidth-intensive applications have led to another evolutionary step towards the significant increase of interconnect bandwidth and the simultaneous system's complexity and price reduction. This has happened by trying to shorten as much as possible the trace copper wiring trace in high speed systems. Actually, the wiring path has been totally removed in between the optical link connector and the chip.

Figure 13: MicroPOD attached to FPGA chip

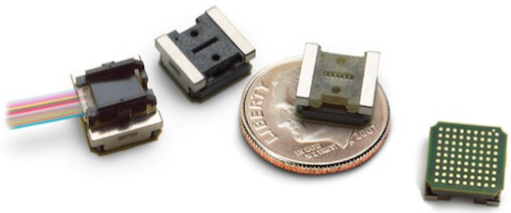

Figure 14: MicroPOD relative size

For the interaction between FPGA and optical fibers, Altera, together with Avago, have developed the world's first demonstration of the company's Optical FPGA technology, MicroPOD [6]. By integrating programmable devices and optical transceivers within a single package, MicroPOD is capable to break through the limitations of copper-based and conventional optical solutions on power, port density and cost. Reducing to a fraction of an inch the electrical signal path from the I/O pad of the chip to the input of the optical transceiver is quite important. It helps to lower signal degradation and jitter effects and at the same time improve signal integrity by reducing data errors caused by parasitic elements in the signal path.

This creative technique will be valuable to meet the rapidly growing high demands of next-generation video, cloud computing, 3D gaming applications, computer storage and communication infrastructure.

2.6 Quantum Dots

After implementing appropriate protocols, designing high-performance and low-power devices, selecting the most efficient medium there is nothing much left to improve in order to build high speed optical link systems. However, researchers have shown a way to provide light to the optical fiber by a much faster light source using quantum dots. Eventually, this can boost up the transfer rate.

Absorbing energy is the fundamental principle way that all light sources follow to produce light. As an example, they absorb energy by an electric current, emitting it back as light. Of, course light is not the only way the energy is emitted out. Unfortunately, a part is being transformed to heat, which is considered as a loss. It is therefore critical to emit light as fast as possible, in order to minimize the losses and the required initial energy. Examining the phenomenon deeper, an electron can be excited by the absorbed energy, jumping on from his initial position, by shining a light. This "movement" creates an "empty space", which has to be covered again. It depends on the interaction between light and matter how fast the electron will decay back emitting the light. Usually, the interaction is not that strong slowing down the whole procedure, which means slow and energy inefficient light sources.

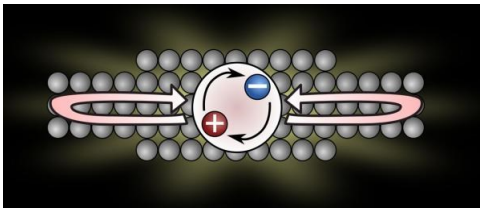

Figure 15: Quantum Dot

A quantum dot has both positive particles and negative particles that are missing electrons. Meaning that there are enough "empty spaces", which needed to be filled up. The attraction between the electrons and those spaces create a new quantum state. The main characteristic of this state is the interaction between light and matter, which creates an efficient and fast lighting source. Of course, this is not as simple as it is described. Still there are some bottlenecks, which have to be overcome. Experiments have shown that the atoms may come so close together bumping to each other or being far enough to cancel any quantum dot speed up.

Chapter 3 Hardware and Software

3.1 Hardware

3.1.1 Cyclone V GT

3.1.1.1 Overview

The Cyclone V GT FPGA development board is a powerful platform for development and implementation designs based on Cyclone V GT 5CGTFD9E5F35C7N device. Various memory interfaces and peripherals are available on the board to support development in different areas [7]. The high-speed mezzanine card (HSMC) connectors, which can be installed with a wide range of HSMC components from Altera and its partners, offer a wide variety of design expansion solutions.

Figure 16: Overview of the Cyclone V GT FPGA Development Board Features

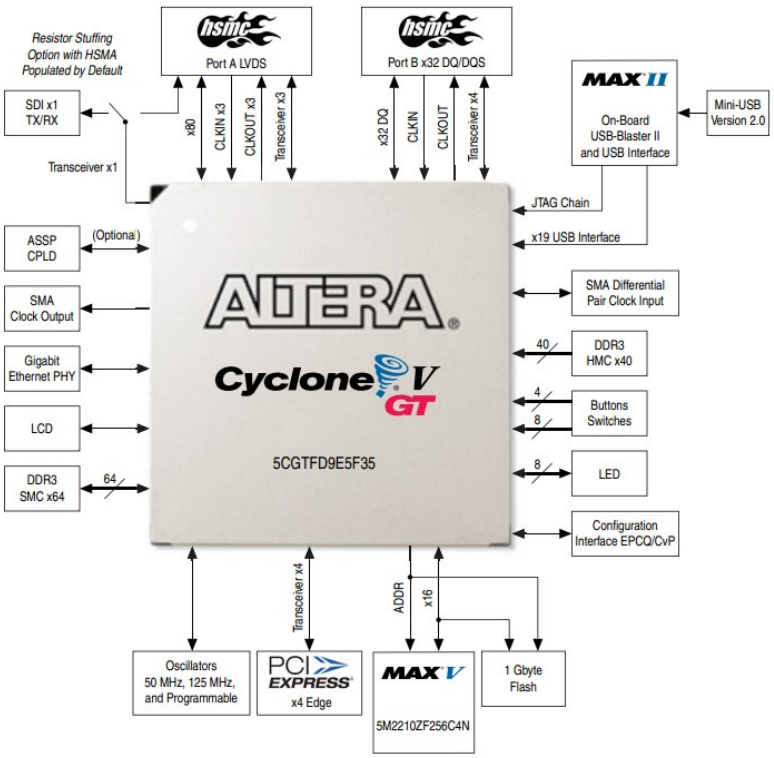

Figure 17: Cyclone V GT FPGA Development Board Block Diagram

Figure 17 shows the components and interfaces on the board. Among them, the most useful and critical ones for this thesis are the two HSMC connectors, static and programmable oscillators, gigabit Ethernet PHY and its Transceivers.

3.1.1.2 FPGA

The Cyclone V GT FPGA development board is embedded with a Cyclone V GT 5CGTFD9E5F35C7N device in a 1152-pin package. The hardware resources also include totally 560 user I/Os and 12 transceiver channels.

Resource	5CGTFD9E5F35C7N
LEs (Logic elements)	301K
ALMs (Adaptive Logic Modules)	91200
Registers	454240
Total Memory	13917Kb
18-bit × 18-bit Multiplier	684
PLLs	8
Transceivers (6 Gbps)	12

Table 4: Cyclone V GT Resources Distribution

3.1.1.3 Clocking

Multiple oscillators are available on the Cyclone V GT FPGA development board. The standard oscillators have frequencies of 50 MHz and 125MHz. The programmable oscillators have a frequency range from 10–810 MHz and can be adjusted by the application provided by Altera.

Oscillator Type	Frequency	I/O Standard
Standard_50	50.000 MHz	1.5-V CMOS
Standard_125_P	125.000 MHz	LVDS
Standard_125_N		
Programmable1_P	100.000 MHz (Programmable between 10-810 MHz)	LVDS
Programmable1_N		
Programmable2_P	148.500 MHz (Programmable between 10-810 MHz)	LVDS
Programmable2_N		
Standard_Eth_PHY	25.000 MHz	2.5-V CMOS

Table 5: Cyclone V GT On-Board Oscillators

3.1.1.4 Gigabit Ethernet PHY

The development board provides 10/100/1000 base-T Ethernet with an on-board Marvell 88E1111 PHY and Altera Triple-Speed Ethernet MegaCore MAC function. The PHY-to-MAC interface only supports a RGMII interface [8].

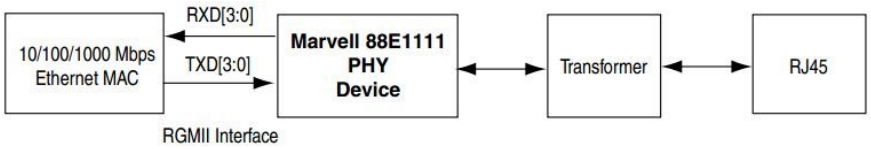

Figure 18: RGMII Interface between FPGA (MAC) and Marvell 88E1111 PHY

3.1.1.5 HSMC

The HSMC connectors are supplied for serving the single-ended signaling on most pins or multi-gigahertz differential signaling on some specific ones. They are optimized for current and emerging High-speed Serial Interconnect standards like PCI Express, Gigabit Ethernet, SPI4.2 and others. The I/O connections of HSMC pins can be programmed onto the development board depending on the requirements of a design.

Each HSMC interface can provide four channels of 5.0 Gbps transceivers. The four channels are located at the Bank 1 of each HSMC ports as we can see in figure 19.

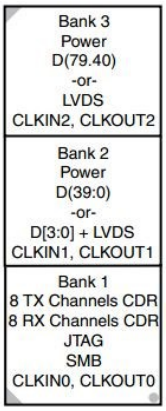

Figure 19: HSMC signal and bank diagram

More specifically, as shown in figure 20, pins 17 to 32 are able to be used as serial high speed transceiver connection links.

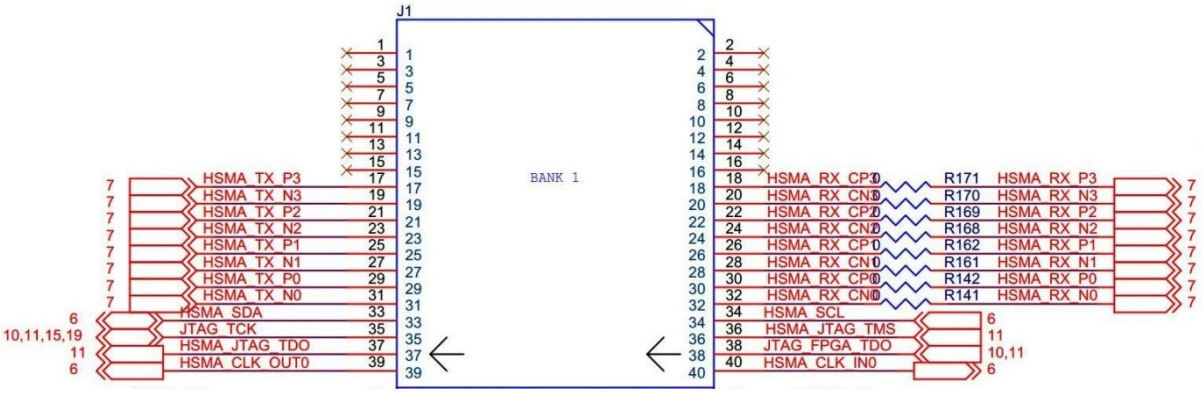

Figure 20: The schematics of the Bank 1 of a HSMC port

Both HSMC ports follow the same signal and bank diagram, using the exact same pins as transceiver connectors. This is why the implementation does not need to be modified whichever port is used.

3.1.1.6 Transceivers PLLs

In Cyclone V GX/GT/SX/ST devices, there are two transceiver PLL sources: CMU PLL (Clock Multiplier PLL) and fPLL (Fractional PLL).

Transmitter PLL	Serial Data Range	Availability
CMU PLL	0.611 Gbps to 6.144 Gbps	Every channel when not used as receiver CDR
fPLL	0.611 Gbps to 3.125 Gbps	Two per transceiver bank

Table 6: Transmitter PLL Capability and Availability

For the fPLLs, they have several key advantages over CMU PLLs. First, they allow larger reference frequency values. Then, they have much smaller step-size or higher resolution. An fPLL allows step sizes on the order of tens of Hz, while a CMU PLL may result in tens of kHz. Moreover, the fPLLs locks faster compared to a similar CMU solution.

In the Cyclone V GT 5CGTFD9E5F35C7N device, the fPLL located adjacent to the transceiver banks are available for clocking the transmitters for serial data rates up to 3.125 Gbps.

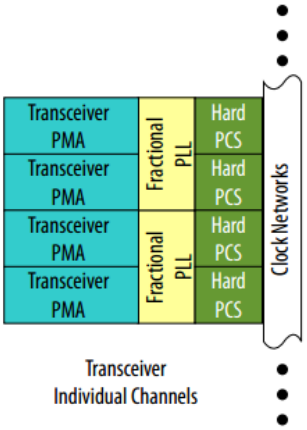

Figure 21: Simplified layout of the fPLLs in the transceiver channels

3.1.2 DE4 Development Board

3.1.2.1 Overview

The DE4 Development board is a high end board powered by the Altera's Stratix® IV GX chip. This device is ideal for projects implementation related to high speed serial connectivity, memory interface and generally high performance designs. Its key features are: the improved jitter performance, the low power consumption and the high bandwidth it offers. The board offers a wide variety of peripherals and connectors, which were quite useful for the implementation of this thesis.

Figure 22: Overview of the DE4 Development Board Features

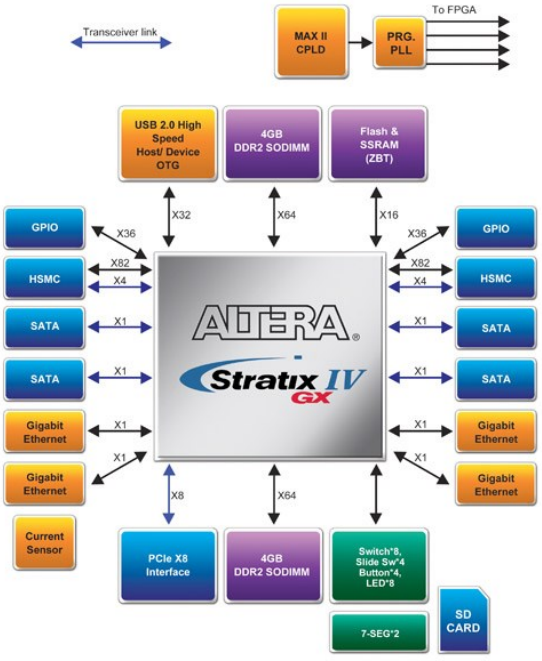

Figure 23: DE4 FPGA Development Board Block Diagram

As it is depicted in figure 23 above the development board is equipped with four Ethernet Controllers and two HSMC connectors as well and some programmable oscillators.

3.1.2.2 FPGA

The resources' distribution of the embedded chip is summarized in table 6 below.

Resource	EP4SGX230KF40
LEs (Logic elements)	228K
ALMs (Adaptive Logic Modules)	91200
Registers	182400
Total Memory	17133Kb
18-bit × 18-bit Multiplier	1288
PLLs	8
Transceivers (8.5 Gbps / 6.5Gbps)	24/12

Table 7: DE4 Resources Distribution

3.1.2.3 Clocking

The DE4 board comes with a mixture of standard and programmable oscillators available to produce a wide range of clock frequencies [9].

Oscillator Type	Frequency	I/O Standard
Standard_50	50 MHz	1.8/2.5/3.0-V
Standard_100	100 MHz	1.8-V
Programmable1_P	62.5 / 75 / 100 /125 / 150 / 156.25 / 187.5 / 200 / 250 / 312.5 / 625 MHz	2.5-V or LVDS
Programmable1_N		
Programmable2_P	62.5 / 75 / 100 /125 / 150 / 156.25 / 187.5 / 200 / 250 / 312.5 / 625 MHz	LVDS
Programmable2_N		
Programmable3_P	62.5 / 75 / 100 /125 / 150 / 156.25 / 187.5 / 200 / 250 / 312.5 / 625 MHz	LVDS
Programmable3_N		
Standard_Eth_PHY	25.000 MHz	2.5-V CMOS

Table 8: DE4 On-Board Oscillators

3.1.2.4 Gigabit Ethernet PHY

The development board carries four Marvell Integrated 10/100/1000 Gigabit Ethernet Controllers. Those controllers work as an Ethernet PHY, which interfaces a default SGMII MAC and they are being clocked by a dedicated 25 MHz clock oscillator.

3.1.2.5 HSMC

The DE4 contains two HSMC [10] ports, which can host enough external devices, since they do not follow any specific interconnection interface. They offer multiple I/O connections that can be programmed as required according to the connected device. The connectors offer optimal interfaces for high speed multi-gigahertz differential signaling as well as single-ended signaling for lower speed connections.

Some of the standard interfaces that can easily be implemented through the HSMC connectors are the Gigabit Ethernet, the PCI Express and the latest version of SPI.

Both HSMC ports are divided in three banks which contain signal connections for different purpose.

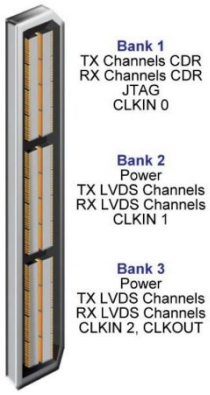

Figure 24: HSMC signal and bank diagram

The first partition contains all the connections which were mainly related to the most considerable part of this thesis. Although both ports share the same bank division and functionality, there is a slight but important difference. Checking the schematic files of the DE4 we realized that one of the ports support the double number of serial link transceiver channels since all its Bank 1 pins are connected to the chip.

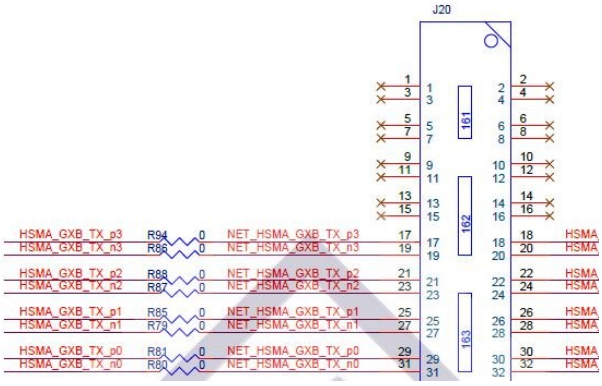

Figure 25: The schematics of the Bank 1 of a HSMC port a

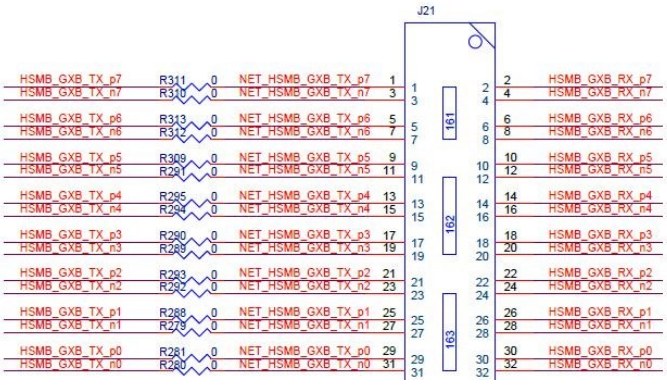

Figure 26: The schematics of the Bank 1 of a HSMC port b

The schematic of figure 26 belongs to HMSC port B, which allows multiple and selective Transceiver usage through its fully wired Bank 1.

3.1.2.6 Transceiver PLLs

The DE4 development board offers the option of two PLL types. One option is the common Clock Multiplier (CMU) PLL and the other is a special low-jitter, high-frequency Auxiliary Transmit (ATX) PLL. Every transceiver block has its dedicated PLL for every channel, while an ATX PLL is available per two transceiver blocks.

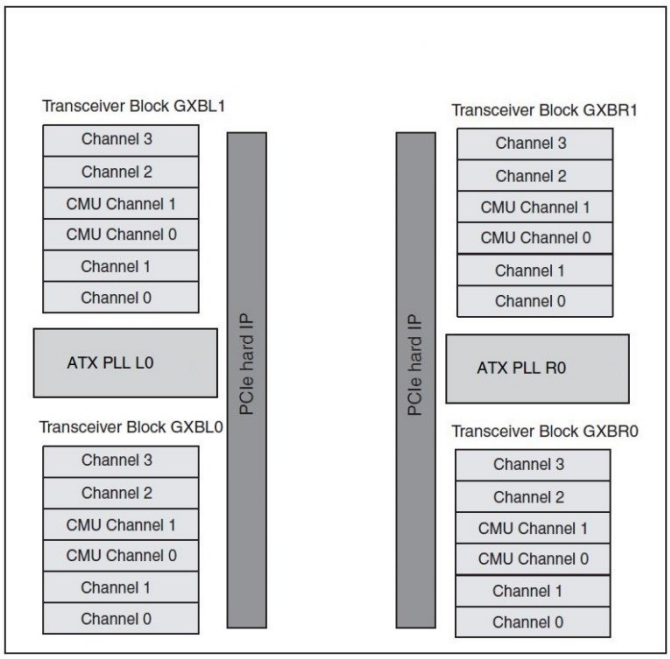

Figure 27: Simplified layout of the ATX PLLs in the transceiver channels

Another advantage -except of its characteristics- of an ATX PLL is that it can be utilized over transceiver blocks as a shared PLL bonding the channels without using one of the transceiver's channel as the CMU does.

3.1.3 Device Comparison

As the major part of our thesis is the implementation of high speed serial links structures in order to check the feasibility and the integrity of our devises, our comparison is related to parts that are important in our designs.

First of all, it is clear that DE4 is equipped with more and faster transceivers compared to Cyclone V GT, which is an exceptional chip of the Cyclone V device family. Additionally, the DE4 development board has one HMCS connector, which is fully wired by all its pins to the FPGA chip, meaning that it can host and fully exploit any HCMS external board. Taking a closer look we note that the extra connected pins belong to the group of high speed serial transceivers pins. As it will be presented later, the Cyclone V GT board is restricting one of our designs.

Finally, another transceiver related issue, which will be addressed in the implementation part, is that Cyclone V board is not giving the option to use one of its fPLLs in order to clock a transceiver. As far as the DE4 board is concerned, it allows using the ATX PLL option and it is also recommended by the corresponding user guide.

3.1.4 Ethernet HSMC Card and Marvel 88E1111 Controller

Figure 28: Ethernet-HSMC Daughterboard

On the Ethernet HSMC card, there are two RJ45 Ethernet transceivers, two Marvell 88E1111 Ethernet controllers, two 25 MHz oscillators, two voltage regulators, one I2C EEPROM, twelve LED indicators and an HSMC connector on the back side [11]. Figure 29 shows the block diagram of the Ethernet HSMC card.

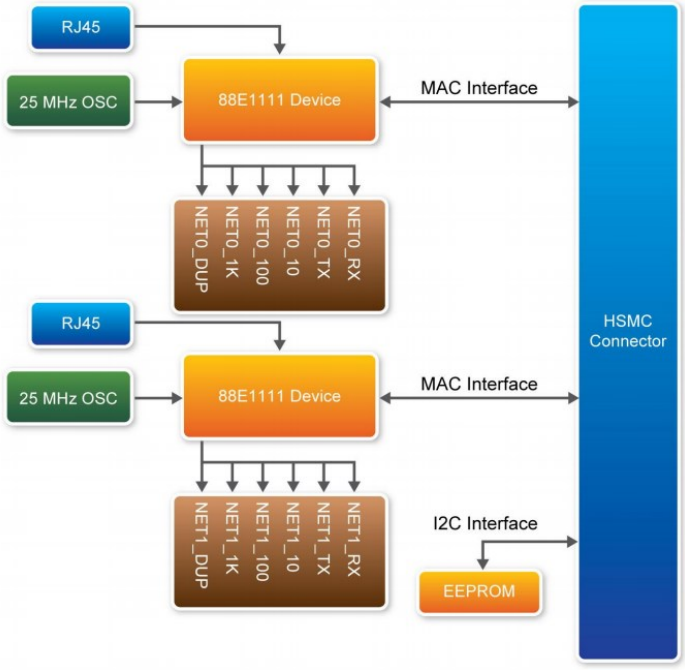

Figure 29: The block diagram of the HSMC-NET card

The card is dynamically configurable to support 10 Mbps, 100 Mbps (Fast Ethernet) or 1000 Mbps (Gigabit Ethernet) operation. Its dual-port integrated transceiver supports GMII/MII/RGMII/TBI MAC interfaces for direct connection to a MAC port. The two Marvell 88E1111 Ethernet Controllers are physical layer devices for Ethernet 1000BASE-T, 100BASE-TX, and 10BASE-T applications. The 88E1111 devices incorporate an optional 1.25 GHz SERDES (Serializer/Deserializer) [12]. The serial interface may be connected directly to a fiber-optic transceiver for 1000BASE-T/1000BASE-X media conversion applications. Additionally, the 88E1111 device may be used to implement 1000BASE-T Gigabit Interface Converter (GBIC) or Small Form-factor Pluggable (SFP) modules. Figure 30 is the illustration of how 88E1111 is applied in a Copper Application.

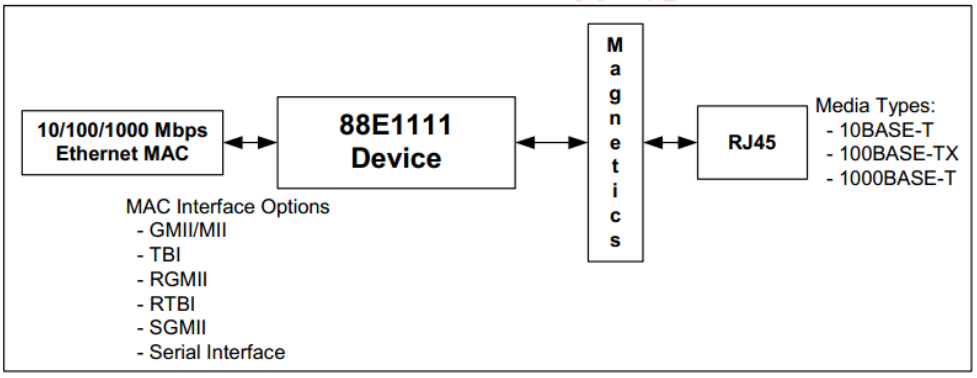

Figure 30: Marvell 88E1111 Device used in Copper Application

The data flow direction, working mode and transmission speed is presented by the LED indicators immediately which enable users to observe the performance and analyze the problems during the development. An EEPROM is also provided that is configured by the I2C interface [13]. The size of the EEPROM is 2 Kbit which can store MAC information or user's data. The default I2C slave address is '0xA0'. The pin description between the HSMC connector and EEPROM is shown below in figure 31.

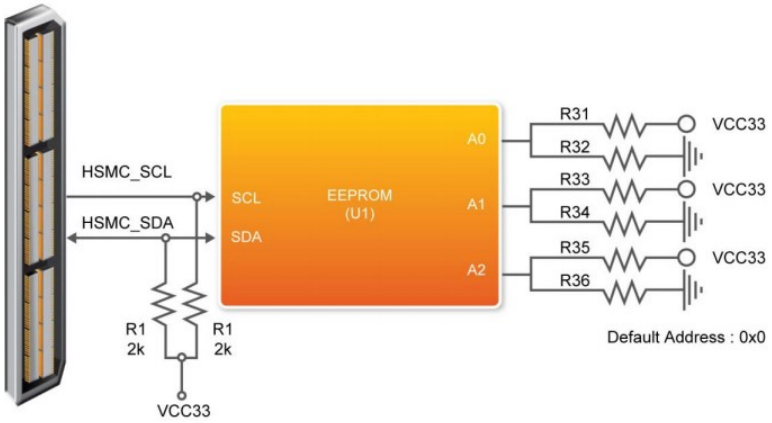

Figure 31: The block diagram of the EEPROM and HSMC connector

3.1.5 SFP-HSMC Card

Figure 32: SFP-HSMC Daughterboard

This board took its name by the Small Form-Factor Pluggable [14] Transceiver modules it can host and the High Speed Mezzanine Connection [10] to the FPGA board. It is a hardware evaluation platform, related to the Intellectual Properties of Altera for every device, which are

going to be analyzed in more details later in a dedicated part. The main purpose of this daughter card is the implementation of high speed serial links by plugging the appropriate SFP connectors to its ports.

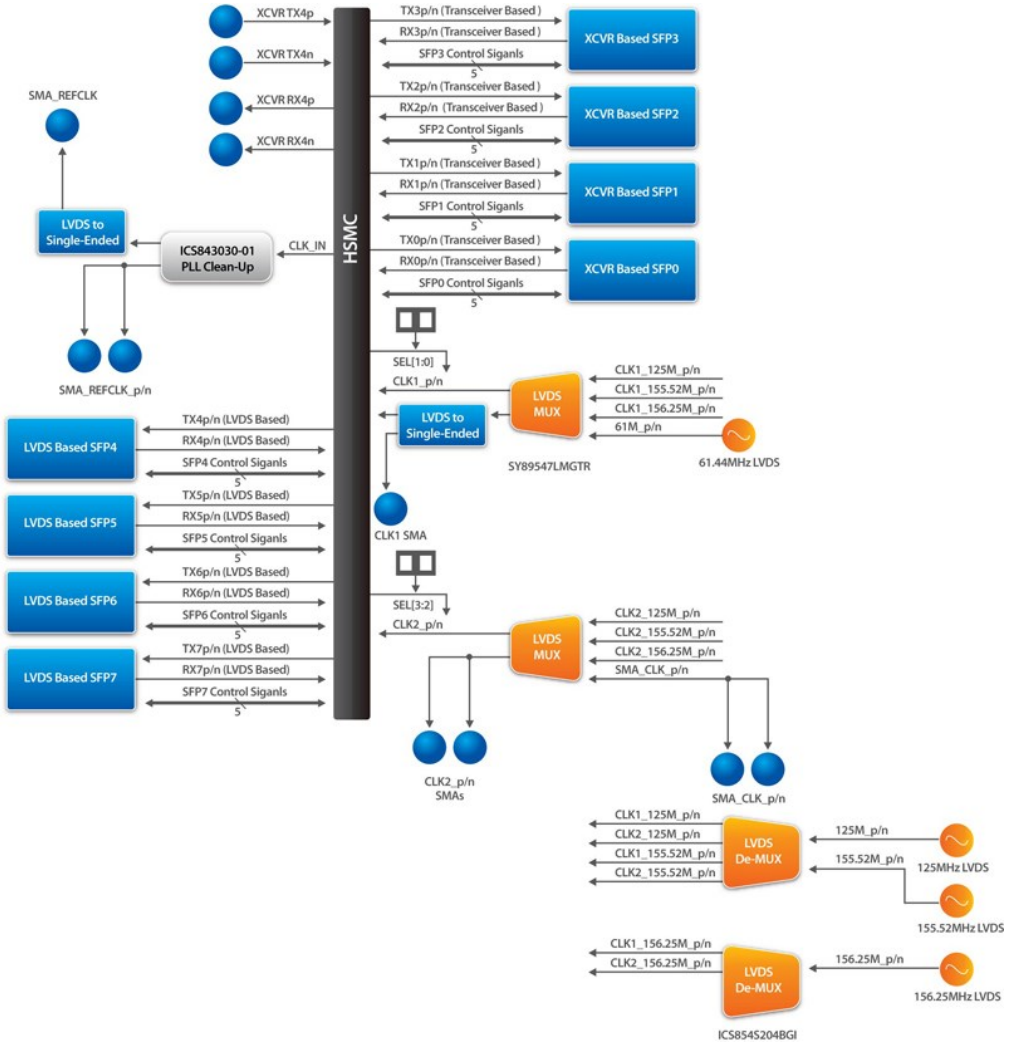

Figure 33: SFP HSMC Card Block Diagram

As it is clear by the block diagram the board embeds some clock oscillators and some SMA connectors to export those clocks as well as some other SMAs as data transceivers. There is no processing unit for the incoming or outgoing serial data on the Daughter Card. Therefore, the whole data processing part takes place in the FPGA side of the design. For our purpose we used one of the XCVR Based ports to transfer data to/from the board and check its clock-data integrity. As soon as the used IP provide us with high-end transceiver design, there was no need using any SFP-HSMC Card clock, since individual FPGAs CDR modules were provided by the PHY part. In a high level perspective, we used TX (p/n) and RX (p/n) pins of a transceiver, for our differential signal transmission, and its respective control signals to enable and monitor the behavior of the specific high speed channel.

3.1.6 DUAL XAUI to SFP+ HSMC Board

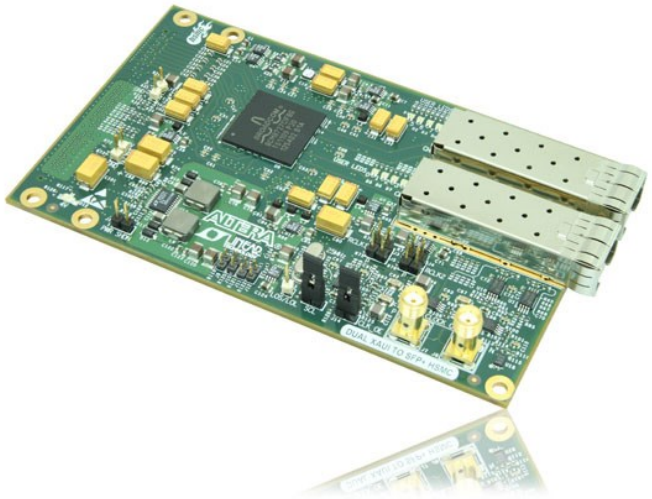

Figure 34: DUAL XAUI Daughterboard

This daughterboard is intended to be used for design and implementation of 10G Ethernet systems, which are based on Transceivers implemented in host FPGAs. Of course, it is prerequisite that those devices support XAUI interface. The card embeds 2 full duplex 10G SFP+ channels with a XAUI backend interface. So, at the FPGA side a hardware driver is needed to be implemented.

The important features describing the board and its functionality are:

- Two independent XAUI interfaces from the HSMC to the BCM8727
- Two independent SFI interfaces from the BCM8727 to SFP+ cages
- MDIO interfaces
- I2C EEPROM for HSMC identification and user data
- Si5334C clock generator
- 156.25MHz reference available on SMA connectors and through the HSMC connector
- 4 user bi-color LEDS for each channel (8 total bi-color LEDs)

Taking a closer look of the board's block diagram at the figure below, we can easily distinguish the main parts that compose the XAUI expansion card.

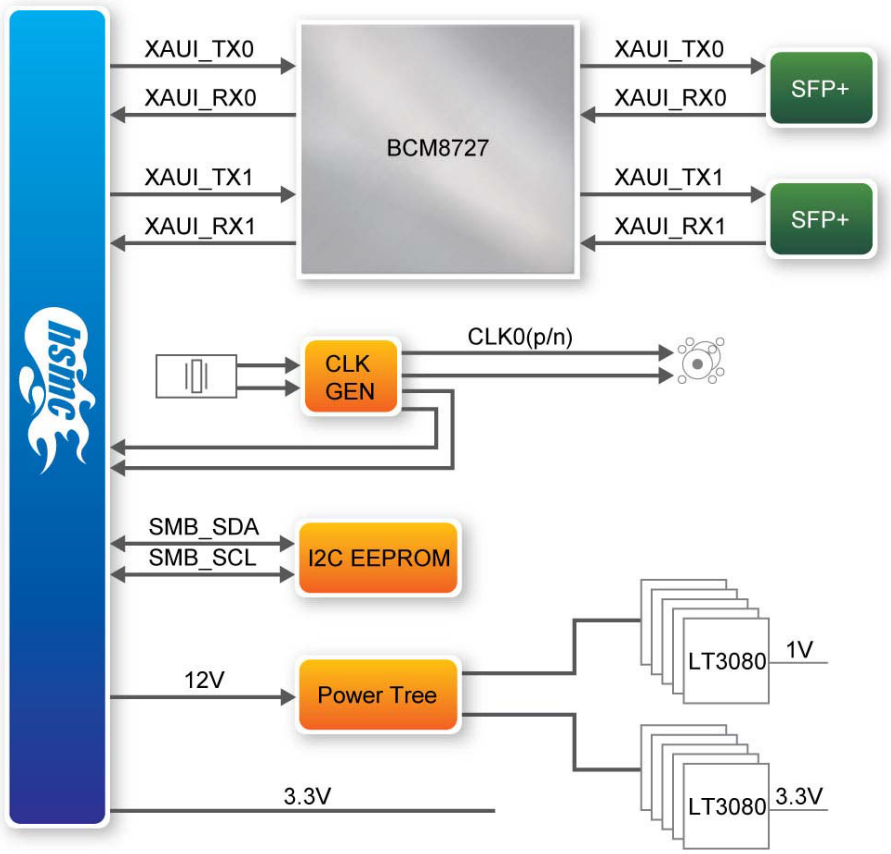

Figure 35: Block diagram of the Dual XAUI to SFP+ HSMC board

- Clocks
  An oscillator capable of generating clean low jitter 156.25 MHz is placed on the board. This clock can supply not only the daughterboard itself but also the host board.

- Memory Devices
  Most SFP+ optical modules contain status and configuration registers, which are accessible through an I2C interface available on the HSMC connection pins of the DUAL XAUI board.

- Power
  The power is supplied to the board from the 12V and 3.3V supply of the host board available on the HSMC connector.

- Featured Device: BCM8727
  The BCM8727 is a dual-channel 10 GbE transceiver [ch. 4.2] that can be used as multi-rate PHY interfacing both for SFP and SFP+ modules. It is fully compliant to the IEEE 802.3 standard and it is developed using a Digital Signal Processor (DSP). Finally, an on-chip microcontroller implements the control algorithm for the DSP core.

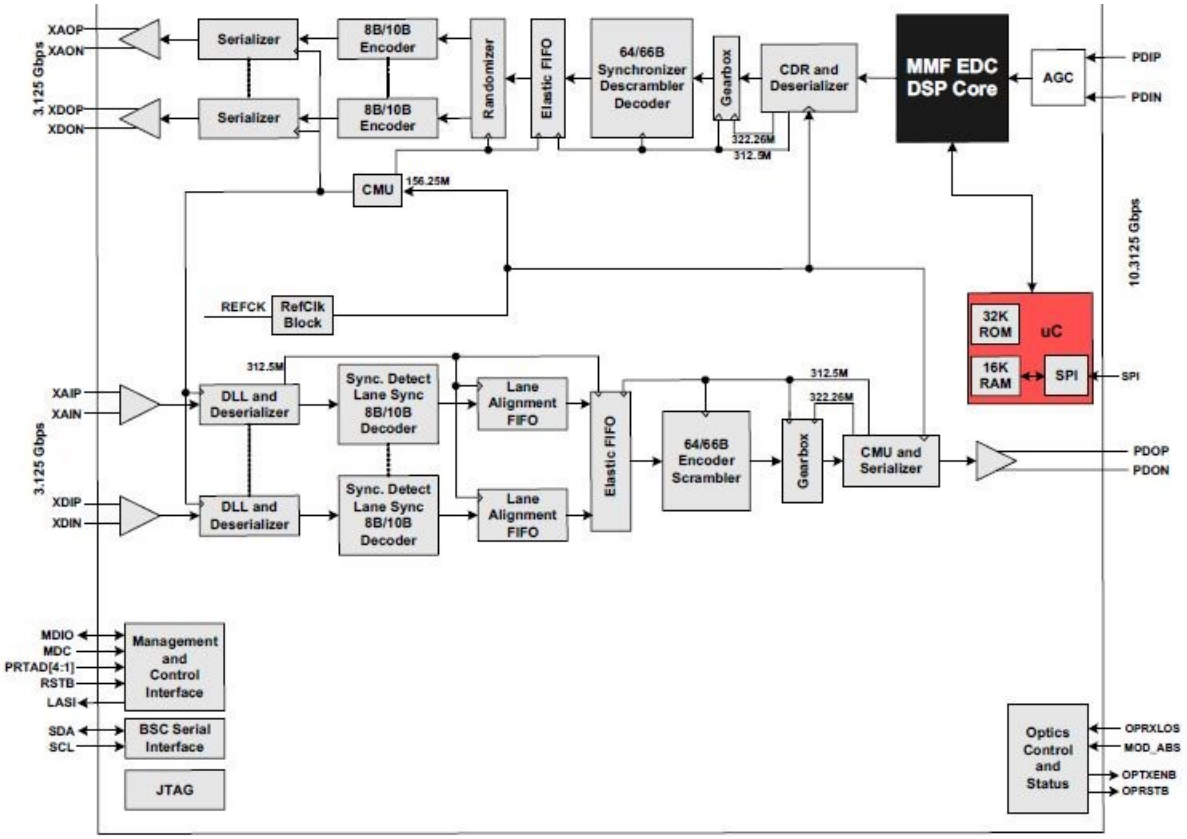

Figure 36: BCM8727 Signal channel block diagram

3.2 Software

The IDE we used in order to build our designs and program the devices with the corresponding generated programming file is the Quartus II, version 13.1 by Altera. Additionally, we used some embedded tools of Quartus for checking and verification of our designs. These are: The Transceiver Toolkit [15] and the SignalTap II Logic Analyzer [15]. Alongside with Quartus, the Nios II Software Build Tools for Eclipse [16], version 13.1 was used to program any soft processor instance, which was created.

Chapter 4 Transceivers Datapath

4.1 Standard Transceiver Datapath

Fast Ethernet connection up to 1 Gb/s is being replaced in more and more systems by high speed serial links, which reach faster data rates. FPGAs are capable to serve and handle these links due to special transceiver hard logic existing in contemporary chips. The deployed transceivers provide specific datapath to the data stream to follow. Since high speed FPGA's transceivers are mainly used in this thesis we will analyze them by explaining in more details the most important aspects.

4.1.1 Transmitter (TX)

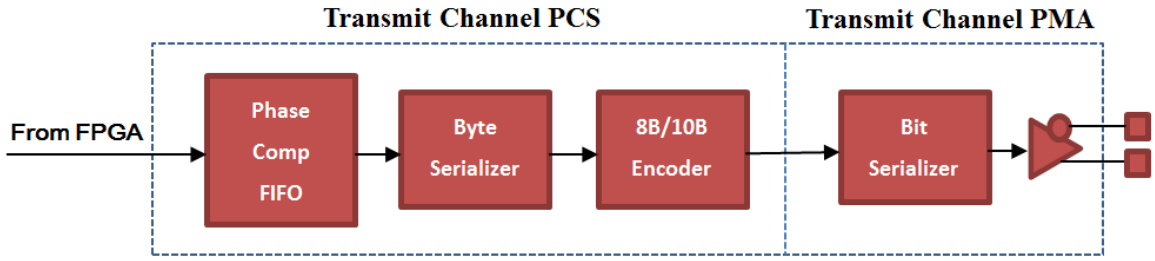

Figure 37: Transmitter Channel PCS and PMA

4.1.1.1 Phase Compensation FIFO

The Phase Compensation FIFO is an interface between the transceiver and FPGA logic. It compensates for differences of phase between transmitter clock and FPGA core clock. Because of the high running speed with specific jitter tolerances, the transmitters use a local PLL. This Local PLL can generate the transmitter's necessary clock, which follows the jitter tolerance design requirements. Core clock is not capable of providing the required clocks with desired jitter tolerances.

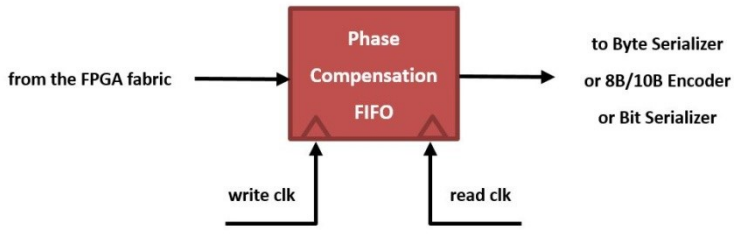

Figure 38: TX Phase Compensation FIFO Instance

With the Phase Compensation FIFO, the parallel data can be synchronized into the transmitter's own clock domain. The FIFO can process data in different widths from 8 to 40 bits wide, based on which one is supported by the device family.

4.1.1.2 Byte Serializer

The Byte Serializer mainly aims to decrease the data rate of the interface between the transmitter and the FPGA logic but the line rate should be kept the same meanwhile. By dividing the input datapath by two, the transceiver channel can run at higher data rates and also keep the FPGA fabric frequency within the maximum limit.

Based on the device types, the Byte Serializer can serialize data in one or two modes: single width mode that uses an 8 or 10-bit wide data path and double width mode that uses a 16 or 20-bit wide data path. Selecting, for example, single width mode, the FPGA's interface will be 16 or 20 bits and the Byte Serializer converts it to 8 or 10 bits. On the output of the Byte Serializer, the least significant byte (LSB) is transmitted first. In figure 39, if desirable line rate is 3.125 Gb/s, the FPGA transmitter interface needs to run at 312.5 MHz which is not allowed according to the interface frequency maximum limit. If the user doubles the FPGA transmitter interface and enables the byte serializer, the interface needs to run at half the frequency as before or at 156.25 MHz.

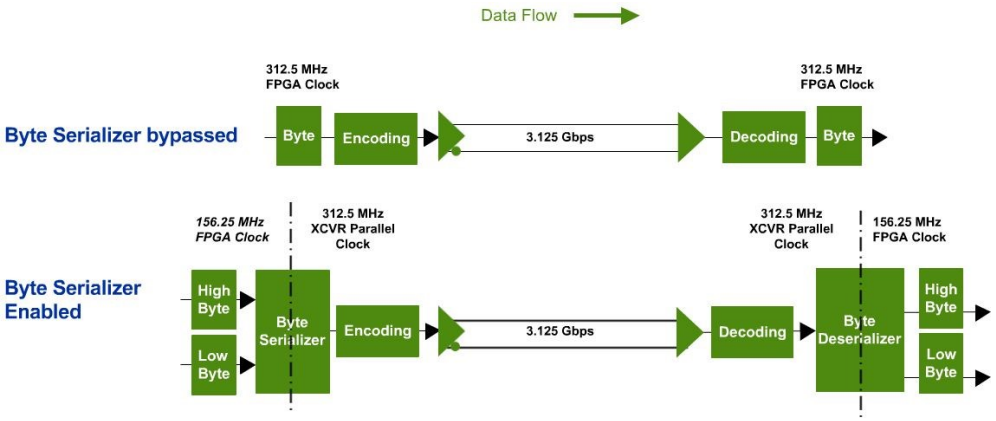

Figure 39: Byte Serializer Block Diagram

Increasing the transfer data rate over 1 Gbps, the data is being conveyed in serial way. This means that data and reference clock for the receiver, in order to sample and deserialize the stream, have to travel together. Whichever medium is used to transfer the data, in the end it will end up as electrical signal, which has to strictly follow some characteristics, so as to help the corresponding design rebuild the sampling clock and receive the transmitted information successfully. Therefore, the next part is quite important and it will be analyzed extensively.

4.1.1.3 8b/10b Encoder

Through every electronic circuit, electric energy is being transferred in its wires because of capacitive coupling. In some cases, it may not cause issues, yet in other cases, like baseband transmission of digital data, it can have destructive effects. In a channel with high-pass characteristics, **A**lternative **C**urrent (**AC**) coupled electrical connections might lose their **D**irect **C**urrent (**DC**) information of the signal, due to attenuation.

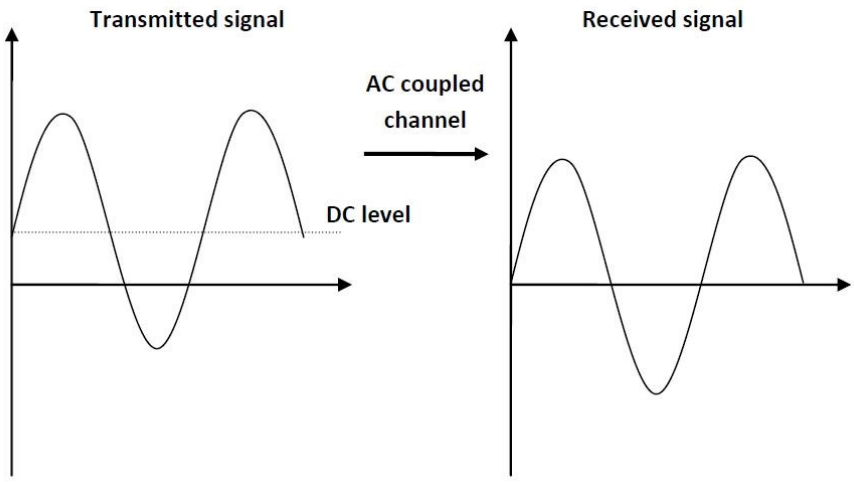

Figure 40: Effect of AC coupled channel on the DC component of a signal

The 8b/10b encoding was first introduced by Al Widmer and Peter Franaszek in [17] and its main purpose is to maintain the DC-balance in the transmitted data. No matter how redundant the data is, the total number of 1s and 0s transmitted only differs by at most 2 at all times. Also, except for a comma control signal used for synchronization, there are never more than five 0s or 1s transmitted in a row. This ensures proper clock recovery at the receiver end.

The 8 bits of data, denoted HGFEDCBA, are divided into two groups: The lower 5 bits (EDCBA), and the upper 3 bits (HGF). An 8-bit data string is denoted Dx.y, where x is the decimal value of EDCBA, and y is the decimal value of HGF. Control symbols are denoted Kx.y. There are $2^{10} = 1024$ possible symbols, but only $2^8 = 256$ possible data strings, so that only the symbols with less than 5 same consecutive bits are used. Also, some of the 8-bit data symbols can be encoded into two different 10-bit symbols, one of them having two more 1s than 0s and vice versa.

The x portion of the data is encoded into a 6-bit entity (abcdei), and the y portion is encoded into a 4-bit entity (fghj). The bits are then transmitted from least to most significant, as depicted in the following figure.

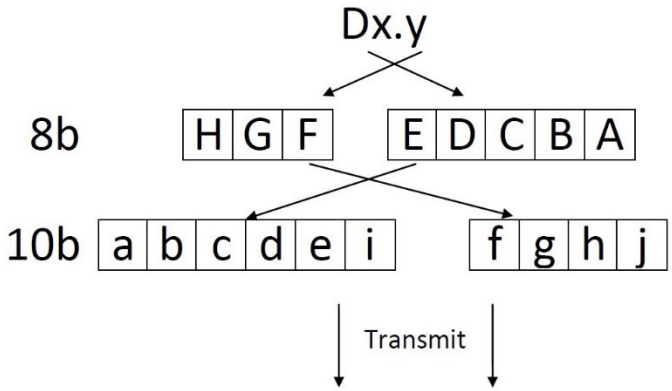

Figure 41: Bit ordering

The encoder needs to keep track of the disparity in the number of 1s and 0s sent. A value, called the Running Disparity (RD), holds this information. By convention, the RD starts at a value of -1. When the transmitted symbol has no disparity, the RD is left unchanged. In the

case where the code word to be transmitted is one that has two possible encodings, a disparity will occur. If the RD is -1, then the code word with two more 1s is chosen and the next RD value is +1. If the RD is +1, then the code word with two more 0s is chosen and the next RD is set to be -1. Therefore, this RD information has to be propagated and used as an input into the next encoding step. Note that the decoders do not require this information, since the backward mapping is unique. Tables below provide the mappings for encoding x and y and for common control symbols.

Input		**Output**	
		RD = -1	**RD = +1**
Notation	**EDCBA**	**abcdei**	
D.00.x	00000	100111	011000
D.01.x	00001	011101	100010
D.02.x	00010	101101	010010
D.03.x	00011	110001	
D.04.x	00100	110101	001010
D.05.x	00101	101001	
D.06.x	00110	011001	
D.07.x	00111	111000	000111
D.08.x	01000	111001	000110
D.09.x	01001	100101	
D.10.x	01010	010101	
D.11.x	01011	110100	
D.12.x	01100	001101	
D.13.x	01101	101100	
D.14.x	01110	011100	
D.15.x	01111	010111	101000
D.16.x	10000	011011	100100
D.17.x	10001	100011	
D.18.x	10010	010011	
D.19.x	10011	110010	
D.20.x	10100	001011	
D.21.x	10101	101010	
D.22.x	10110	011010	
D.23.x	10111	111010	000101
D.24.x	11000	110011	001100
D.25.x	11001	100110	
D.26.x	11010	010110	
D.27.x	11011	110110	001001
D.28.x	11100	001110	
D.29.x	11101	101110	010001
D.30.x	11110	011110	100001
D.31.x	11111	101011	010100

Input		Output	
		RD = -1	RD = +1
Notation	**HGF**	**fghj**	
D.x.0	000	1011	0100
D.x.1	001	1001	
D.x.2	010	0101	
D.x.3	011	1100	0011
D.x.4	100	1101	0010
D.x.5	101	1010	
D.x.6	110	0110	
D.x.P7	111	1110	0001
D.x.A7	111	0111	1000

Input		Output	
		RD = -1	RD = +1
Notation	**HGFEDCBA**	**abcdeifghj**	
K.28.0	00011100	0011110100	1100001011
K.28.1	00111100	0011111001	1100000110
K.28.2	01011100	0011110101	1100001010
K.28.3	01111100	0011110011	1100001100
K.28.4	10011100	0011110010	1100001101
K.28.5	10111100	0011111010	1100000101
K.28.6	11011100	0011110110	1100001001
K.28.7	11111100	0011111000	1100000111
K.23.7	11110111	1110101000	0001010111
K.27.7	11111011	1101101000	0010010111
K.29.7	11111101	1011101000	0100010111
K.30.7	11111110	0111101000	1000010111

Table 9: 8B/10B Mapping Tables

For D.x.7, the preceding 6-bit code determines which one of the primary (D.x.P7) or the alternate (D.x.A7) code is to be sent. This choice prevents any sequence of 5 consecutive 1s or 0s (which is reserved for comma control symbols). Control codes K28.1, K28.5 and K28.7 are comma symbols used for determining the alignment of the 8b/10b encoded received signal.

4.1.1.4 Bit Serializer (SerDes)

The Bit Serializer is the last module before actual transmission and is related to the Byte Serializer since both follow the same functional principle. The Bit Serializer converts the parallel data, whether scrambled or encoded, to serial data. To do this, the Bit Serializer requires two clocks, a low-speed for the parallel side and a high-speed clock for the serial side. These clocks are generated by dedicated clock resources located in or near the transceivers. The data from the Bit Serializer is transmitted with the LSB transmitted first.

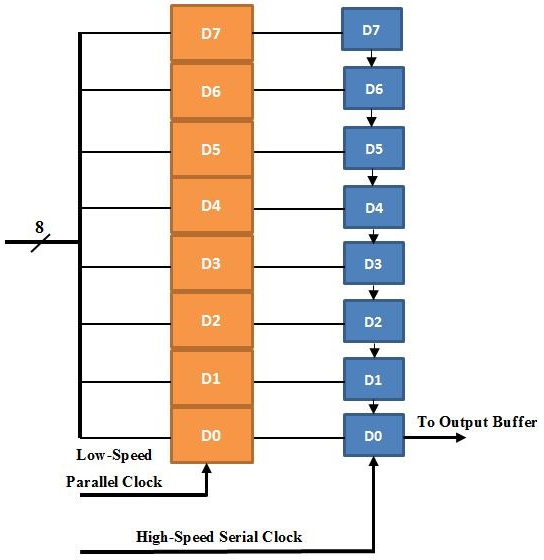

Figure 42: Bit Serializer Block Diagram

4.1.2 Receiver (RX)

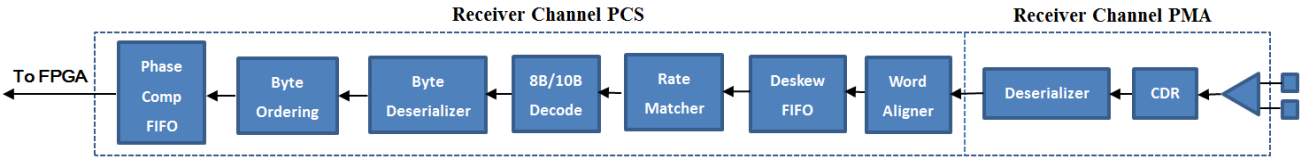

Figure 43: Receiver Channel PCS and PMA

4.1.2.1 Receiver CDR

The clock and data recovery unit (CDR) can extract the clock from the serial input data. After that, other blocks in the receiver can also use the clock signals to sample data. The CDR should first be trained so as to get the correct frequency by an input clock source. As the ones used for the transmitter path, this input clock source can be an input I/O pin or the output of PLLs. When the training is up, the CDR is able to use the transitions in the data signals to track the incoming data stream.

4.1.2.2 Bit Deserializer (SerDes)

The last block in the PMA of the Receiver is a Bit Deserializer, performing as it is expected just the reverse procedure of the Bit Serializer unit. It is responsible for converting the serial data stream into parallel data, no matter how they are encoded or scrambled data. The LSB is

received first. In short, the idea of the Bit Deserializer is to split the data into wider data at half the speed. All transceiver devices support single width mode, yet there are some FPGAs supporting also double width mode.

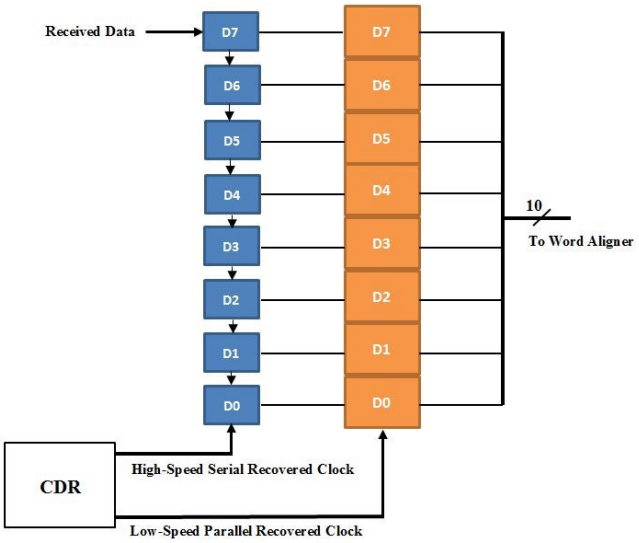

Figure 44: Bit Deserializer Block Diagram

4.1.2.3 Word Aligner

Parallel data at the input of the receiver PCS loses the word boundary of the upstream transmitter from the serial-to-parallel conversion in the deserializer. The Word Aligner applies an alignment pattern to locate byte or word boundaries in the incoming data. Once the alignment pattern has been found, the Word Aligner can then shift the data to align itself and subsequent data to that boundary.

The Word Aligner is composed of 4 blocks. The first one is the pattern detector. It can search for a predefined alignment pattern by checking the incoming data within the current word boundary and set a flag when the alignment pattern is detected. The second one is the aligner. It locates the alignment pattern and realigns the word boundary based on it. When realignment is ready, it sets a flag. The third one is the manual bit slip which makes possible the manual shift of the word boundary one bit a time to get alignment. And the last one is the run length checker. It is used for searching a user or protocol defined number of consecutive 0s or 1s in the incoming data stream and sets a flag when it occurs.

4.1.2.4 Deskew FIFO

Deskew FIFOs are available in some transceivers' receiver path. They can also be called as channel aligner. The Deskew FIFO exists in multi-lane channels like XAUI. It aligns all channels' clocks to channel 0's one. To do that, each of the channels needs to transmit special align characters simultaneously. The Deskew FIFO makes sure the align symbol is in the same columns across all the channels, otherwise, the Deskew FIFO will try to align all channels. Once aligned, if misaligned special characters are received, the Deskew FIFO treats the channels as out of alignment.

4.1.2.5 Rate Matcher

The Rate Matcher tries to compensate for slight small clock frequency differences between the upstream transmitter and the local receiver clocks. In links where the transmitter and the receiver are clocked by independent reference clock sources, frequency differences -count in ppm- can affect the data. The Rate Matcher is implemented as small FIFO (20 words deep), which compensates differences up to ±300 ppm. According to a predefined 20-bit long pattern, consisting of 10-bit control pattern and a skip pattern, the module handles the clock frequency differences by looking for the control pattern followed by the skip pattern. Once the detection is successful, the FIFO performs the proper functions to avoid under/overflow.

- Skip pattern insertion, when the local clock frequency is greater than the upstream transmitter reference clock frequency.
- Skip pattern deletion, when the local clock frequency is less than the upstream transmitter reference clock frequency.

Skip pattern has to be chosen in a way to preserve neutral disparity of the data stream.

4.1.2.6 8b/10b Decoder

On receiver's part the decoder simply maps its incoming 10-bit data to the correct 8-bit long portion using exactly the same mapping table as the transmitter for encoding technique.

4.1.2.7 Byte Deserializer

Like the byte serializer, the byte deserializer widens the FPGA parallel interface to reduce the FPGA interface clock rate.

4.1.2.8 Byte Ordering

The Byte Ordering block can record bytes after byte deserialization by detecting a predefined byte ordering pattern. This block is unable to be used with rate matching as the Rate Matcher is deleting bytes, the ones Byte Ordering block is trying to arrange. Instead, the Byte Ordering block uses a user-defined pad symbol to align the bytes into the parallel pattern with which they were originally sent. If the predefined Byte Ordering pattern found is not in the LSB position, Byte Ordering should insert the predefined pad pattern to the byte deserialized data. Byte Ordering can work on both single and double-width mode following the same technique, as other modules do, to divide the input by two.

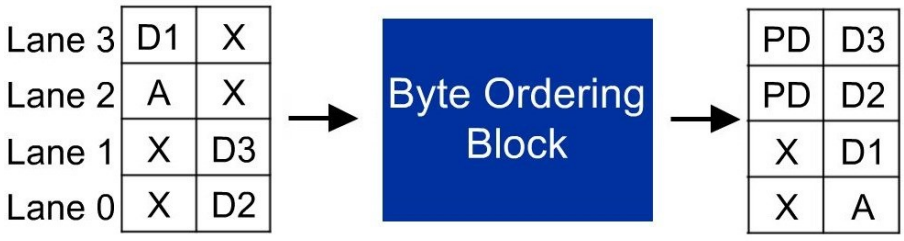

Figure 45: Byte Ordering Diagram

4.1.2.9 Phase Compensation FIFO

As the transmitter, the receiver also contains a Phase Compensation FIFO. It enables users to synchronize the incoming data to a specific clock domain as well as compensate for some phase differences. The receiver path should be driven by a local PLL. There is an essential requirement that the clock that drives the write side of the FIFO must have no PPM difference with the read side clock. This needs to be done by making the clocks come from the same PLL or be driven by the same clock source.

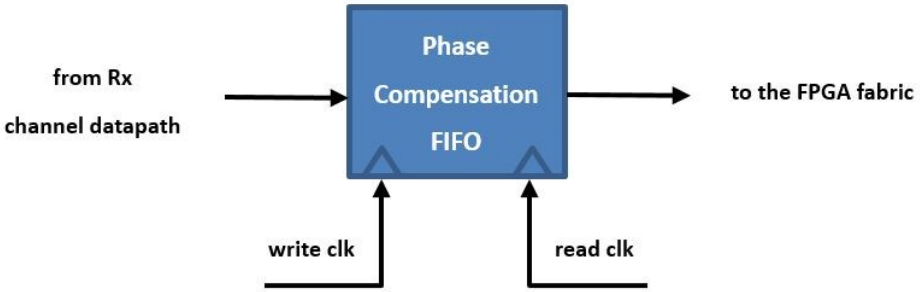

Figure 46: RX Phase Compensation FIFO Instance

The clock input of the read side can be imported by the transmitter PLL clock output or the recovered clock. After passing the phase compensation FIFO, the data will be further analyzed and processed by a media access control or any additional PCS.

4.2 10G Transceiver Datapath

Nowadays, FPGAs have reached data transfer speeds over 40Gbps and more in some cases, but the 10 Gb/s transfer rate is widely used by a variety of systems and especially in data acquisition systems. So, implementing a successful loopback test in 10 Gbps is part of this thesis and the reason why we present "10G Transceivers" section.

The differences with the standard version of the Transceivers, introduced just above, are not so essential, since the datapath module change but the main technique and functionality do not change a lot. The PMA parts on both transmitter and receiver remain exactly the same so they will not be analyzed again.

4.2.1 Transmitter (TX)

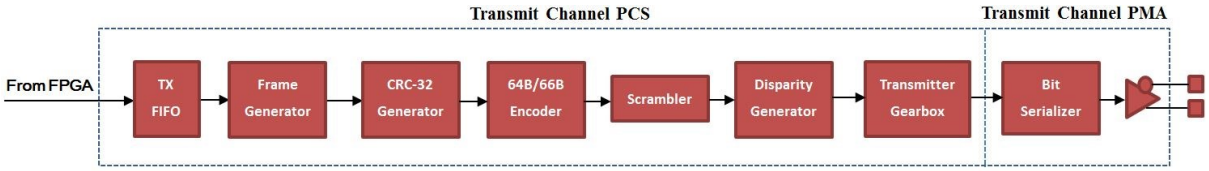

Figure 47: 10G Transmitter Channel Datapath

4.2.1.1 TX FIFO

Like the phase compensation FIFO, the Transmitter FIFO rearranges data and the corresponding control bits into transmitter clock domain for eliminating phase offsets. The width of this interface is decided by the target protocol.

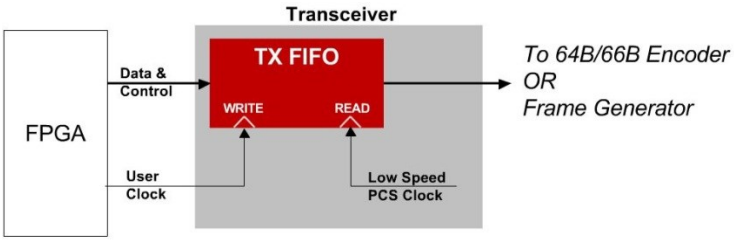

Figure 48: Transmitter FIFO

4.2.1.2 Frame Generator

The Frame Generator is used only in Interlaken configurations, as described in [18]. The Frame Generator block receives the data stream from the Transmitter FIFO and forms the protocol's corresponding frame. It encapsulates the payload and burst/idle control words from the FPGA fabric with the framing layer's control words, such as the synchronization word, scrambler state word, skip word, and diagnostic word.

4.2.1.3 64B/66B Encoder

The 64B/66B Encoder is used in 10G BASE-R configuration, as described in IEEE 802.3-2008 Clause 49. Although the 8B/10B protocol is very popular, it converts every byte into 10 bits. So 20% of the bandwidth is stolen. For data rates higher than 5Gb/s, encoding methods like 64B/66B scheme will be used. This encoding method transforms every 8 bytes (64 bits)

of data into 66 bits for transmission across the link, so less overhead will occur. 64B/66B still allows DC balancing and disparity so that the embedded clock can be recovered. The output is usually passed to a scrambler to make sure the data being sent have enough transitions for clock recovery.

4.2.1.4 CRC-32 Generator

One of the methods to detect errors is redundant encoding which spreads the information across more bits than the original data. More redundant bits can increase the probability of detecting transmission errors [19].

Although these redundant bits are efficient and handy, undetectable error bits may still appear. Those common errors vary due to the storage medium and transmission but undetectable errors can be caused by short bursts of changed bits or occasional isolated changed bits. In order to decrease the undetectable errors as many as possible, the data can be distributed so that it will be less possible that transmission errors lead to a valid encoding of data.

Cyclic Redundancy Code (CRC) is a popular kind of redundant encoding. CRC checkers are capable of detecting the differences between the original data and the transmitted data. They are applied by data transmission applications widely. For example, IBM's Synchronous Data Link Control and other protocols use CRC-16. For larger transmissions like Ethernet and token ring local area network protocols, a 32-bit CRC is used [20].

CRC is effective for two main reasons. First, it provides reliable protection against common errors, such as burst errors where consecutive bits in a data stream are corrupted during transmission. Second, the original data is the first part of the transmission, which makes systems that use CRC checkers easy to detect and implement [19].

The theory of a CRC generator is related to the usages of Linear Feedback Shift Registers (LFSRs). An LFSR is a kind of shift register whose input bit is a linear function of its previous state. To build a CRC generator, the mostly used linear function of single bits is exclusive-or (XOR). As a result, an LFSR drives the input bits by the XOR of some bits of the overall shift register value.

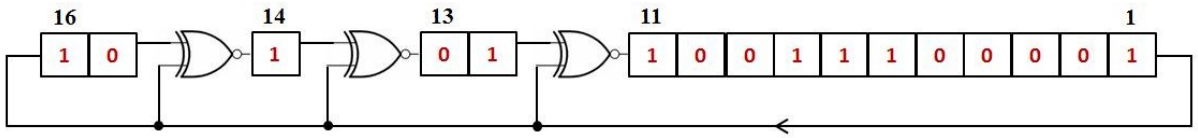

Figure 49: The structure of a 16-bit LFSR

Figure 49 indicates a 16-bit LFSR whose feedback polynomial is $x^{16} + x^{14} + x^{13} + x^{11} + 1$. The register number in bold is corresponding to its primitive polynomial and are counted in reverse to the direction of shifting. The register also cycles through the maximal number of 65535 states excluding the all-zeroes states.

A typical hardware implementation by LFSRs of CRC-16 is shown as the figure 50 [21]:

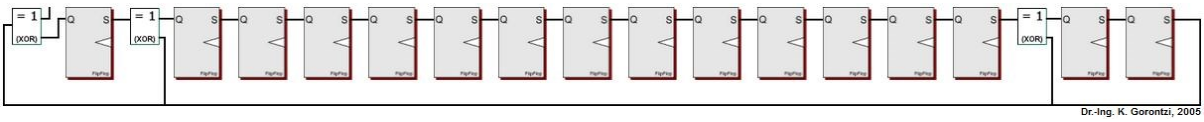

Figure 50: Hardware implementation of CRC-16

The input bits are shifted into the most left XOR gate. The MSB of each byte is shifted in first. Each flip-flop register represents a single CRC output bit. The leftmost flip-flop register is the MSB of the CRC. This implementation does not need to augment the serial input message with zeros and the flip-flop registers are cleared to zeros at the beginning of each calculation.

A CRC is called an n-bit CRC when its check value is n-bits. For a given n, different CRCs have their corresponding polynomials [22].

Common Name	*n*	Generator	
		Polynomial	Hex
CRC-12	12	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$	80F
CRC-16	16	$x^{16} + x^{15} + x^2 + 1$	8005
CRC-32	32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$	04C11DB7

Table 10: Generator Polynomials of some CRC codes

Table 9 presents the generator polynomials used by some common CRC standards. The "Hex" column shows the hexadecimal representation of the generator polynomial (the most significant bit is omitted, as it is always 1).

In this thesis, CRC32 is applied in the data packets for the Ethernet loopback Test and the XAUI part as well.

4.2.1.5 Scrambler

The Scrambler works to decrease the effects from electromagnetic interference between channels by scrambling long sequences of 1s and 0s as well as the repetitious patterns. This encoding method is applied for the 10G protocol and do not prevent a string of up to 64 same bits from being sent in a row. Thus, the scrambler can work using a polynomial to the data words to scramble the bit patterns. The 10G protocol determines the method of scrambling and the polynomial used for scrambling.

4.2.1.6 Disparity Generator

The Disparity Generator is used only in Interlaken configurations. Like the Frame Generator, it conforms the protocol specification and provides a DC balanced data output. It inverts the running disparity of the incoming data to stay within ±96-bit boundary. To ensure this running disparity requirement, the disparity generator inverts bits [63:0] and sets bit [66] to indicate the inversion.

4.2.1.7 Transmitter Gearbox

In the 10G configuration, the parallel input of the PMA can be 40 or 64 bits wide. Considering the PCS can be 66 (10G BASE-R) or 67-bit (Interlaken) words, each word is supposed to be reformatted to the width of the PMA. In this case, the Transmitter Gearbox is needed.

The clocking of the gearbox has to match the one of PCS in order to support the target line rate. In other words, if the data is removed from the gearbox in 40-bit parallel words and is being placed into the gearbox in 66 or 67-bit words, it is essential to ensure that the clocks of the two sides of the gearbox should be matched to neither overflow nor starve the gearbox. The Transmitter Gearbox also needs to maintain the perceived target line rate to the link.

The transmitter gearbox reverses the parallel word, too. As a result, the default behavior of sending the LSB will be modified to send the MSB first.

4.2.2 Receiver (RX)

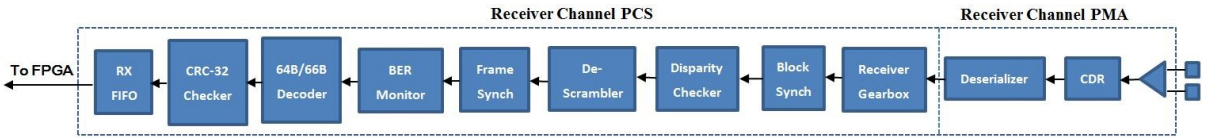

Figure 51: 10G Receiver Channel Datapath

4.2.2.1 Receiver Gearbox

As the Transmitter Gearbox, the Receiver Gearbox needs to adapt the bit width of the PMA to the PCS. Thus, the 40 or 64-bit output of the PMA can be adjusted to fit the 66 or 67 input to the PCS. The PCS receives the LSB first, but the Receiver Gearbox is also capable to reverse bits in order to make the MSB be received first.

4.2.2.2 Block Synchronizer

The Block Synchronizer determines the block boundary following 10GBASE-R or Interlaken protocol. By bit-slipping the incoming data stream, it tries to detect a valid synchronization header. After the detection of the predefined number of synchronization headers (as required by the protocol specification) the appropriate status signal is asserted.

4.2.2.3 Disparity Checker

The Disparity Checker is only used in Interlaken configurations. Basically, it gets its name by its functionality of checking the transmitter's Disparity's Generator. After word synchronization is achieved, the Disparity Checker monitors the status of the 67^{th} bit of the incoming word and determines whether or not to invert bits [63:0] of the received word.

4.2.2.4 Descrambler

The Descrambler uses the same polynomial as the Scrambler module on transmission part. It brings the data back to its original form. The used polynomial is always determined by the applied protocol.

4.2.2.5 Frame Synchronizer

The Frame Synchronizer block in receiver's side checks incoming data to confirm followed protocol's specification. It achieves lock by looking for four synchronization words in consecutive data blocks. After synchronization, the Frame Synchronizer monitors the scrambler word in the data block. In case of three consecutive mismatches, it loses the lock de-asserting its status signal and starts the synchronization process again.

4.2.2.6 BER Monitor

The BER Monitor block conforms to the 10GBASE-R protocol calculating the wrong received bits' ratio, asserting a signal whenever the bit error rate is over a protocol defined threshold.

4.2.2.7 64B/66B Decoder

The 64B/66B Decoder converts the 66 bits encoded data back into its original 8-byte wide state with one control flag per byte. It also monitors the bit error flag from the BER monitor and if asserted sends fault codes into the receiver FIFO and into the FPGA core.

4.2.2.8 CRC-32 Checker

When a CRC checker is used to verify a data frame, the frame is processed as a large binary number that will be divided by a generator number [19]. A remainder is produced by the division and then transmitted together with the data. For the receiver, the data is divided by the same generator number and the remainder will be compared to the one attached to the end of the data frame. If the two remainders are not identical, then an error or even more errors occurred during the data transmission.

4.2.2.9 RX FIFO

The 10G PCS Receiver FIFO can work in different modes which can be enabled based on the protocol implementation or the functional requirements. In clock compensation mode, the FIFO works as the Rate Matcher block in the standard PCS in which it inserts or removes ordered sets to compensate in the range of ±100 PPM between the link's endpoints. This mode is widely used in 10GBASE-R Ethernet. In receiver phase compensation mode, the FIFO takes the incoming data and retimes it into the core clock domain of the FPGA to account for any slight phase offsets between the two domains. This is very analogous to the phase compensation FIFO in the standard PCS. In generic mode, the Receiver FIFO is more like a simple FIFO but supplies the FPGA logic with flags so that the control logic is able to manage the data flow and monitor the FIFO. The generic mode is applied in the Interlaken protocol.

Chapter 5 Implementation and Results

5.1 Ethernet Loopback Test

5.1.1 Ethernet Frame

Data can not be sent as raw data through an Ethernet cable, it has to be framed before reaching the receiver. In that case, we encapsulate our data in the simplest frame we could use.

Figure 52: Ethernet Frame

First we start with 32 bits of Preamble or alignment bits, which are simply 32 bits of 0's. Then it combines the Destination Address followed by the Source Address. In the Destination Address part, a special Address for broadcasting was used in order to be able to transmit the message to any PHY. By setting all the 128 bits of address as 1's, it is not that necessary to know the exact destination MAC address. Yet, in our case, the Source Address used in the package has to be the same as the one, which will be initialized in the Triple Speed Ethernet (TSE) IP [23]. Additionally, Size of Message and Data are placed before the Cyclic Redundancy Check (CRC) part. The last part was explained previously in the report [ch. 4.2.1.4]. It is the module which "connects" transmitter and receiver, by checking the incoming data for transmission errors, requesting retransmission or throwing received packages away. This is the frame we used to implement our test, and it will become clearer later on the source code part, where we can see that the user can type up to 45 characters and still achieve successful communication. This thing proves that, if both edges follow the same interface and the Cyclic Redundancy Check (CRC) [ch. 4.2.1.4] method then communication between any devices can be achieved.

5.1.2 RGMII

The media-independent interface (MII) was original defined as a standard interface used to connect a Fast Ethernet media access control (MAC) block to a PHY chip [8]. The MII bus is standardized by IEEE 802.3u and has been extended to a large number of variants. The MII interface is shown in the figure 53.

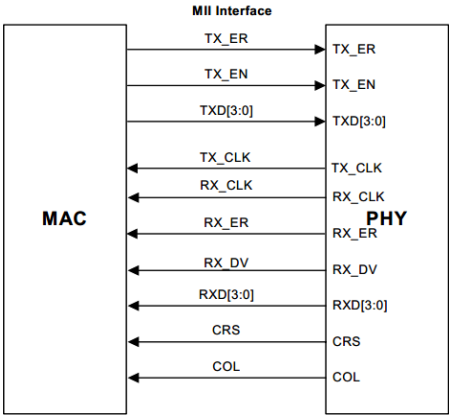

Figure 53: MII Interface

In MII, the simplest variant, a free-running clock is generated by the PHY according to the desired data rate (25MHz for 100Mbit/s, 2.5MHz for 10Mbit/s), which is used as a transmission clock. This clock is also used to drive synchronously on its rising edge all the remaining control signals. This arrangement allows the MAC to operate independently on the link speed. By checking the signal naming in figure 53 we can easily understand or speculate their role. Details and specifications are given in [8].

One of the MII derivations is Gigabit media-independent Interface (GMII), which defines higher speeds up to 1000 Mbit/s, implemented using a data interface clocked at 125 MHz with separate 8-bit data paths for receive and transmit, and is backward compatible with the MII specification. GMII contains all the status and clock signals MII owns besides that its data paths are twice as many. Thus, GMII is able to operate on lower speeds of 10 or 100 Mbit/s as the MII specification could do. The only major difference is the GTX clock provided by the MAC side this time for Gigabit speed transceivers implementations. The GMII interface is shown in figure 54; details are provided in [8].

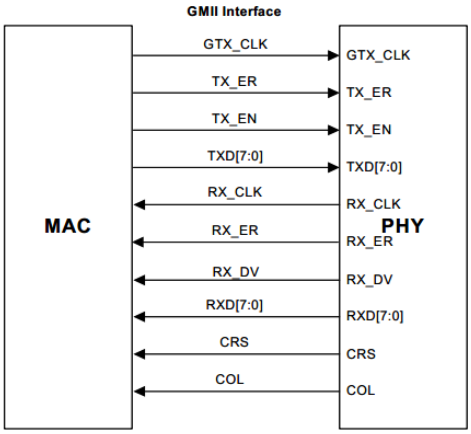

Figure 54: GMII Interface

In order to save the hardware resources and simplify the design, Reduced Gigabit Media Independent Interface (RGMII) is created which only uses half the data pins as used in the GMII interface. This reduction is achieved by clocking data on both the rising and falling edges of the clock in 1000 Mbit/s operation, and by eliminating non-essential signals (carrier-sense and collision-indication). The RGMII signal diagram is shown in figure 55.

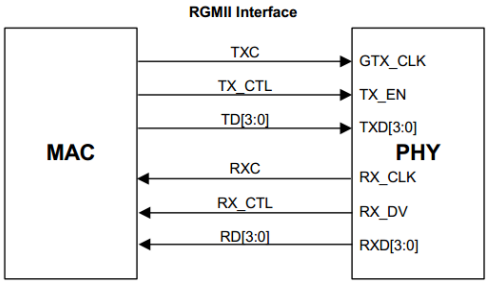

Figure 55: RGMII Signal Interface Diagram

The RGMII only contains: RX_CTL, RXC, RD[3:0], TX_CTL, TXC and TD[3:0]. Unlike GMII, the transmit clock signal is always provided by the MAC on the TXC line, rather than being provided by the PHY for 10/100 Mbit/s operation and by the MAC at 1000 Mbit/s. Source-synchronous clocking is used: the clock signal that is output (by either the PHY or the MAC) is synchronous with the data signals. This requires the PCB to be designed to add a 1.5–2 ns delay to the clock signal to make the setup and hold times on the sink. RGMII v2.0 specifies an optional internal delay, obviating the need for the PCB designer to add delay.

In our Ethernet loopback test, we selected to implement RGMII. It is based on the pin resources on the Development board (Cyclone V GT) and Ethernet HSMC card. The Ethernet controller 88E1111 on the GT board has limited quantity of pins connected to the FPGA i.e., only four data input pins and four data output pins are available. The two 88E1111 controllers on the Ethernet HSMC card has been configured by setting the CONFIG[6..0] bits. Thus, their default working mode is RGMII. On the other hand, RGMII fits our performance requirements. It supports 1000 Mbit/s duplex transmission and its working modes can be modified via the MDIO (Management Data IO) and MDC (Management Data Clocks) signals connected to the 88E1111 controllers.

5.1.3 Triple Speed Ethernet (TSE) IP Core

As it becomes clear by its naming the TSE IP offers a fully multispeed Ethernet integrated solution. It provides MAC and PCS for fast Ethernet applications at 10/100 Mb/s and also Gigabit Ethernet applications of 1000 Mb/s. It is a complete solution supporting all the functionalities like full-duplex mode, transparent and full Ethernet frame termination and generation and efficient power management.

The Triple Speed Ethernet IP function consists of the following blocks: MAC, PCS, and PMA. All blocks are optional and configurable at synthesis time. The MAC and PCS modules are accessible through Avalon Memory-mapped [24] interface by writing or reading their registers [23]. Additionally, status and control signals are provided.

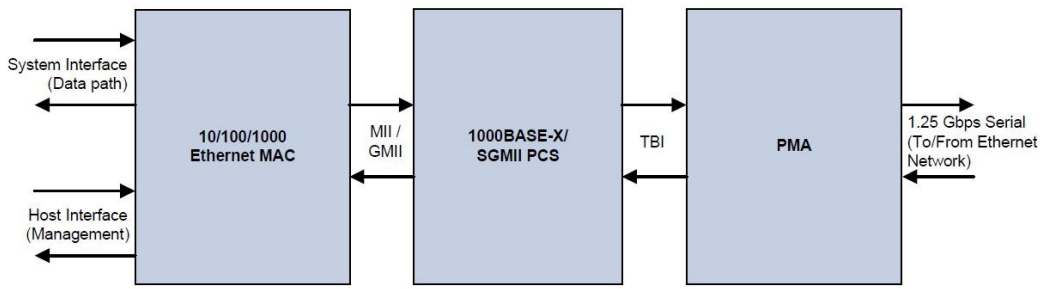

Figure 56: Triple Speed Ethernet IP Block Diagram

By checking the block diagram in figure 56, the simple structure of TSE IP becomes more obvious, which gives the option of MAC and PCS interface selection as will be described in the customization part later. While the PMA plays the role either of the bridge between the PCS and the Marvell device (RJ-45 systems) or the gigabit serial data provider (optical links).

5.1.4 Implementation

5.1.4.1 QSYS-Nios II subsystem

Unfortunately, there are no clear instructions how to send / receive data to the Ethernet controller directly. For that reason, we redirected our focus to Triple Speed Ethernet (TSE) IP by Altera, which seemed to be suitable for our desired implementation. After checking the documentation about the TSE IP core [23], we found out that it will be much easier integrating TSE IP in a system with a soft processor since there is available an API for the specific IP and a tutorial [25] on how to successfully integrate it to a system. Basically a clear block diagram of the implemented system is like this:

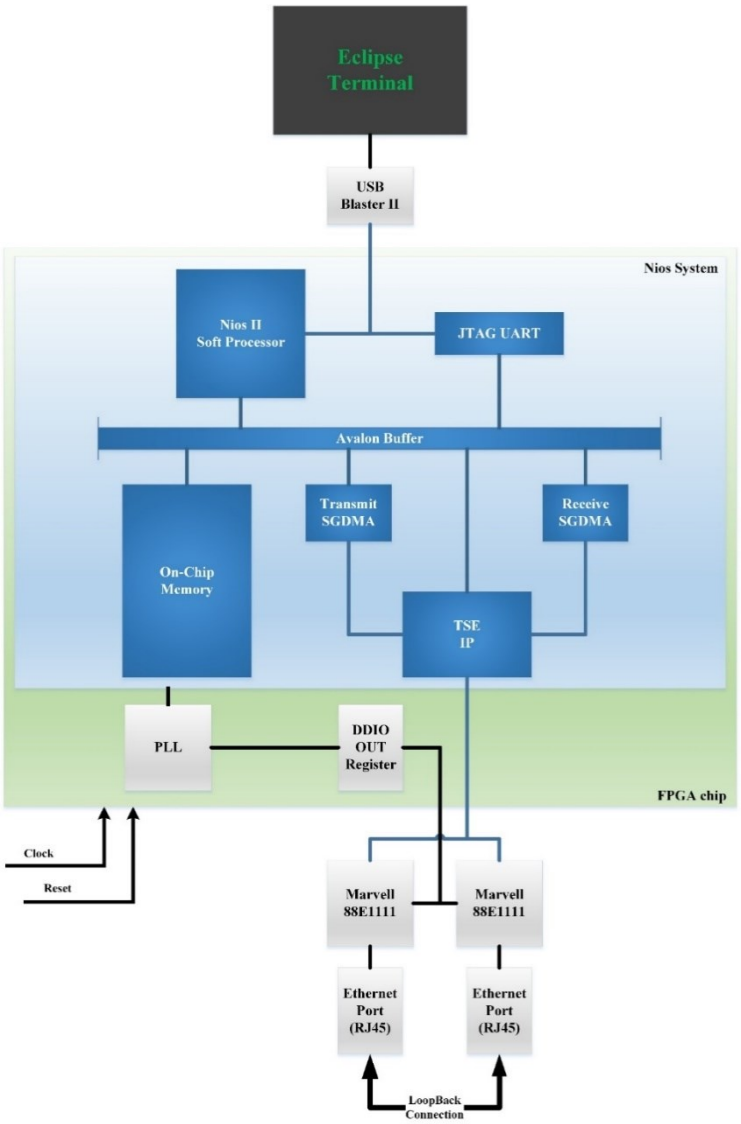

Figure 57: Loopback System Block Diagram

The Nios II subsystem communicates externally through the JTAG UART interface, sending/receiving information to/from the software terminal. Additionally, it takes a clock signal generated by a Phase-Locked Loop (PLL) and an active-low reset as inputs. The soft processor is being used to run application programs, handling the data sent to or received from the TSE IP and of course take care of the generated interrupts. Next important part is the on-chip memory, which is the place where program code for Nios II processor is stored, as well as any data correlated to the Triple Speed Ethernet MegaCore. The reason for building this system is the Ethernet IP core we used in order to implement the MAC sublayer and partially the PHY layer when it is needed, depending on the interface we are using. Physical Chip, Nios II soft processor and Triple Speed Ethernet IP are the basic components of our loopback system and they should be the fundamental parts of any Ethernet subsystem in an FPGA. Last part is the Scatter Gather DMA (SGDMA) controllers, which are responsible for connecting the on-chip memory and the TSE for data transfer. Those controllers establish and handle connections between streaming interfaces like our Ethernet IP and memory-mapped interfaces like our memory. These two controllers are necessary to store the conveyed data in the on-chip memory. We use two SGDMA controller instances, one for transmitted data and another one for received data. All parts described previously are connected to a common Avalon Buffer [24], which supports streaming interfaces for unidirectional flow of data and also serves memory-mapped interfaces for master-slave connections as an address-based read/write interface.

Except the processor subsystem, one PLL and one Double Data Output Register were used. The PLL generate clock signals of specific different frequencies in order to connect them to the corresponding devices, since they do not share the same clock. On the other hand, DDIO OUT Register transmits data out on both clock edges, providing a very accurate relationship between data and clock for the PHY chip.

The purpose of the design is to implement a loopback test between the Cyclone V GT FPGA and the Ethernet Daughterboard, initializing the RJ-45 port of the FPGA as transceiver and the daughterboard's port as a "mirror", which loops back any received message.

5.1.4.2 IP Generation and Customization

The TSE IP comes with a simple generation procedure, where the available options and the customizable parts can be followed and filled in easily.

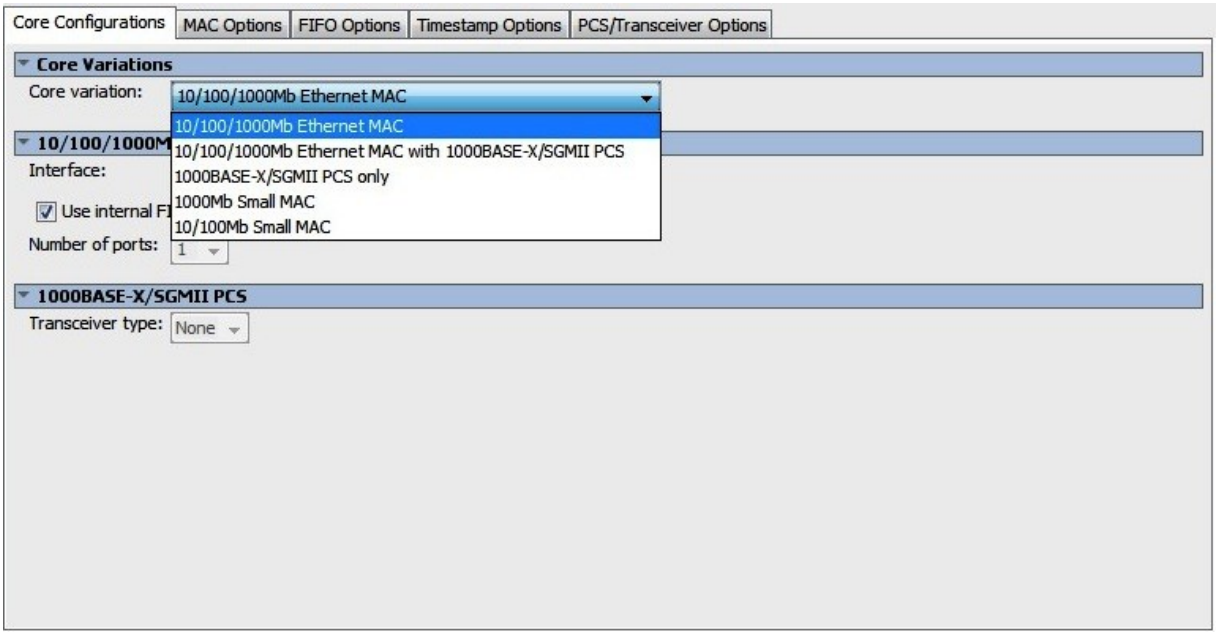

Figure 58: Triple Speed Ethernet Core Configurations 1

Like it was mentioned on the IP introductions all parts are optional and configurable according to the chosen variation. In our design the highlighted variation was used with the simple Ethernet MAC. In continuation the MAC interface has to be determined.

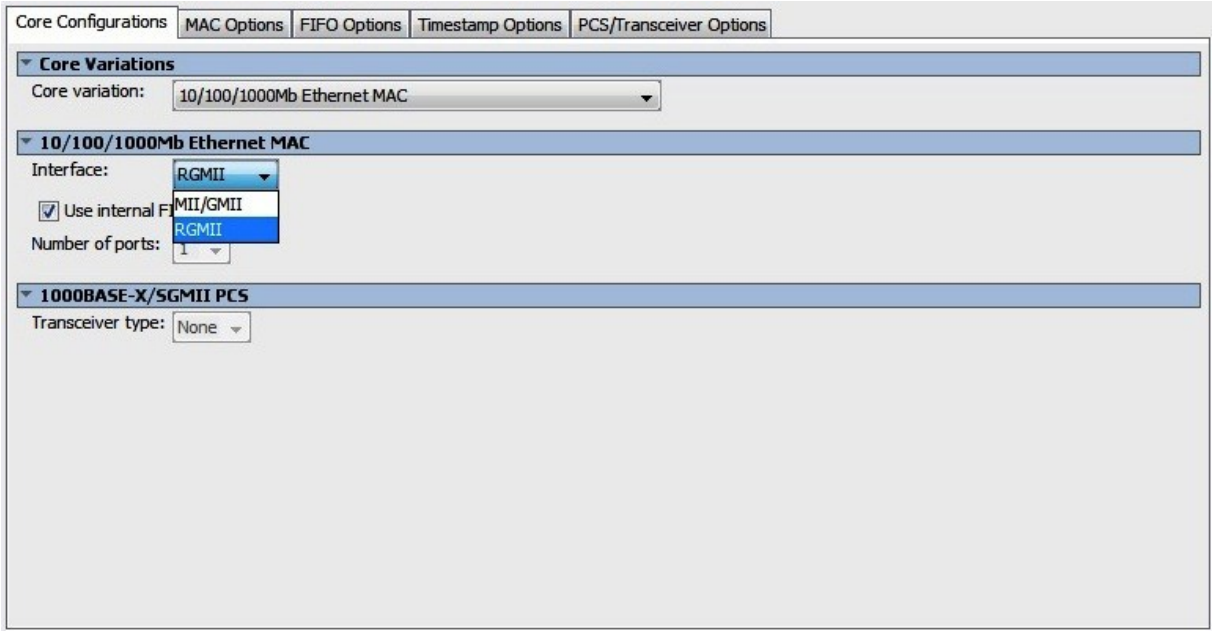

Figure 59: Triple Speed Ethernet Core Configurations 2

Due to our plan to use Gigabit Ethernet the RGMII option was selected as an interface, followed by an internal FIFO for saving any transmitted frame before transmission and also placing the received messages.

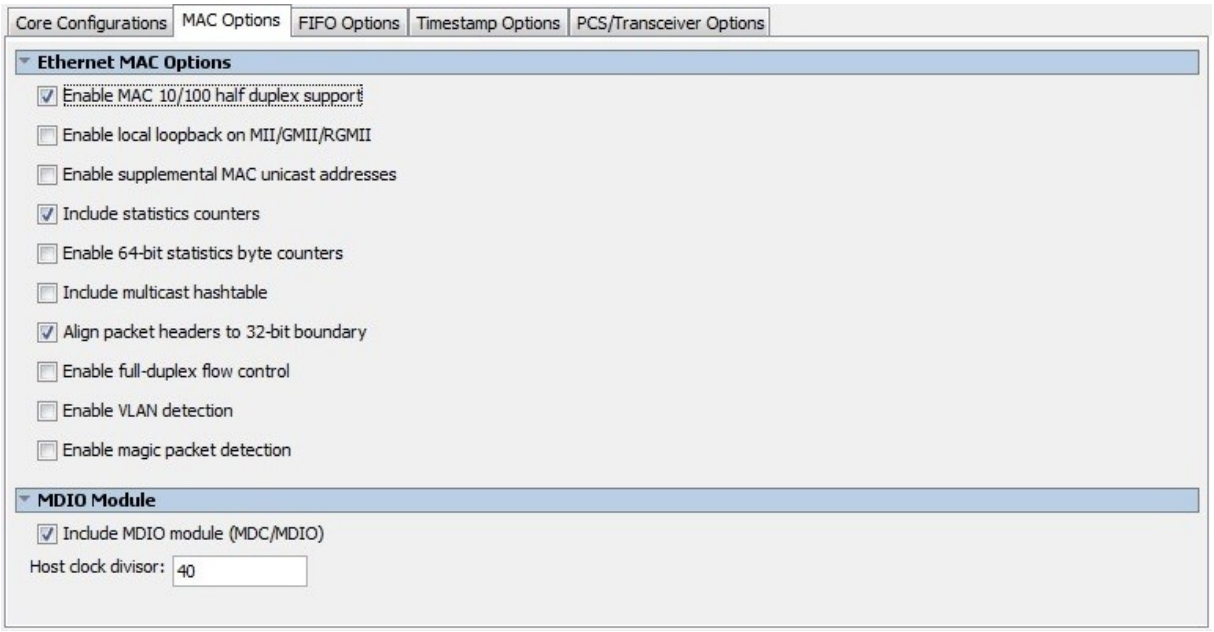

Figure 60: Triple Speed Ethernet Core MAC options

The next tab on the IP generation is the MAC options, where the standard pre-defined options were set. The only extra and important chosen option is the packet headers alignment. This helps to align all packets headers to 32-bit boundaries, reducing software overhead for processing and re-aligning data buffers. This option is used alongside with the internal FIFO, which was enabled on the previous step.

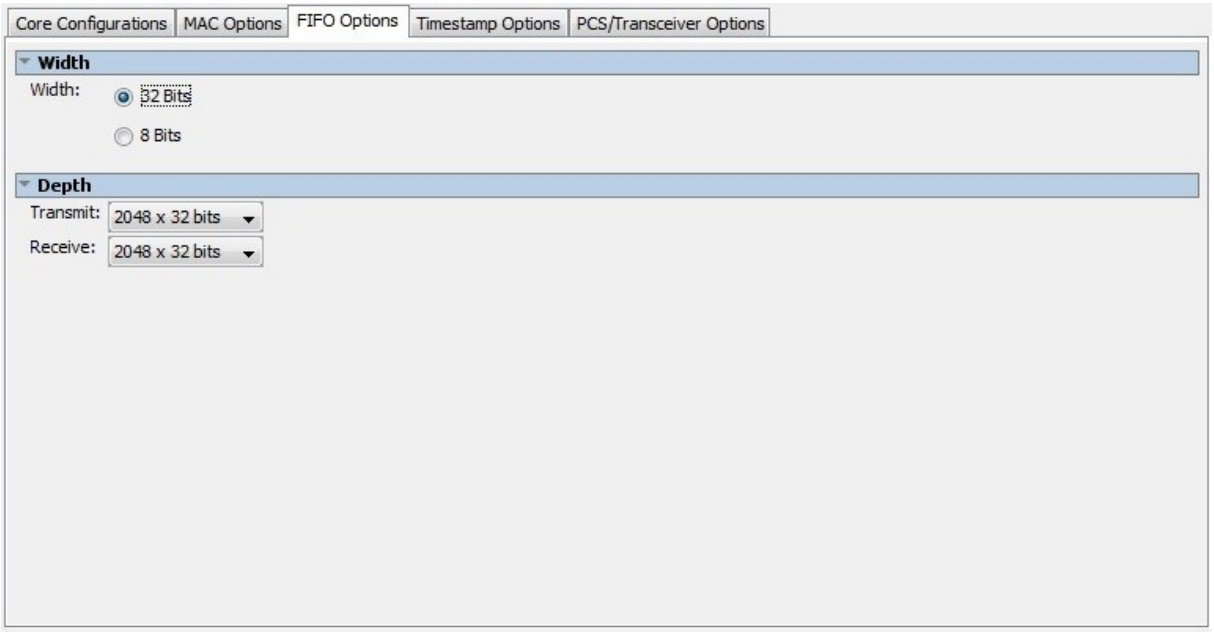

Figure 61: Triple Speed Ethernet Core FIFO options

In the final step the internal FIFO was configured according to the previous selections of 32-bit boundary alignment and according to a reasonable depth of 2 Mb. As it will be shown on the next two figures the other generation IP tabs are not available for customization, because of our prior selections.

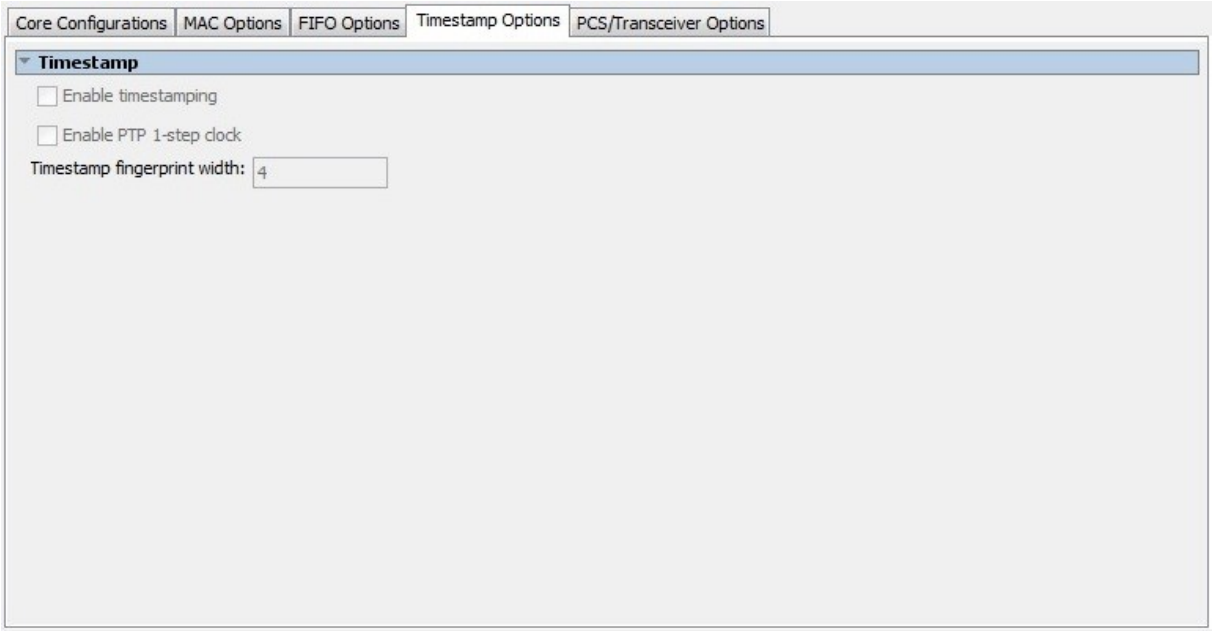

Figure 62: Triple Speed Ethernet Core Timestamp options

Figure 63: Triple Speed Ethernet Core PCS/Transceiver options

5.1.4.3 VHDL code

After building the Nios II subsystem using Qsys tool we had to integrate it to the higher level project. For this reason, a loopback project was created by Altera Quartus 13.1, where our soft processor and the additional IPs were port mapped. As it is expected the top level has only one user interactive input as hard reset and a clock input provided by the development kit on-board oscillators. If we take a closer look at the Top Level block diagram above, we can

clearly see the three port maps for the Nios II subsystem, the PLL and the DDIO Register. The subsystem uses 100MHz working clock, while the Top Level is working at 50MHz. In advance, the PLL provides the system with a clock signal of 2.5MHz for MDC usage and with another one at 125 MHz for transmission reference clock.

5.1.4.4 C code

Now, as far as the software part concerns, in general we had to implement an infinite loop to send data to the loopback port and receive them again after the loop. Prior to the infinite "while" loop, the frame initiation and some configurations had to be done.

```
1.  // Create a transmit frame
2.  unsigned char tx_frame[1024] = {
3.      0x00,0x00,                              // for 32-bit alignment
4.      0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,      // destination address (broadcast)
5.      0x01,0x60,0x6E,0x11,0x02,0x0F,      // source address of transmitter
6.      0x00,0x2E,                          // length or type of the payload data
7.      '\0'                                // payload data (ended with termination
8.                                          // character)
9.  };
```

The frame was implemented as an array of characters following the simple Ethernet format as it was presented in the part [ch. 5.1.1]. Destination and Source Address are predefined like the message length, but all can be dynamically changed and redefined as is obvious.

```
10.     // Initialize the MAC address of transmitter
11.     *(tse + 3) = 0x116E6001;
12.     *(tse + 4) = 0x00000F02;
13.
14.     // Specify the addresses of the PHY devices to be accessed through MDIO interface
15.     *(tse + 0x0F) = 0x00;           // The PHY address of Ethernet device 1
16.     *(tse + 0x10) = 0x12;           // The PHY address of Ethernet device 2
17.
18.     // Write to register 20 of the PHY chip for Ethernet device 1 to set up line loopback
19.     *(tse + 0x94) = 0x4000;
20.
21.
    // Write to register 16 of the PHY chip for Ethernet device 2 to enable automatic crossover
22.     // for all modes
23.     *(tse + 0xB0) = *(tse + 0xB0) | 0x0060;
24.
25.     // Write to register 20 of the PHY chip for Ethernet device 2 to set up delay for
26.     // input/output clk
27.     *(tse + 0xB4) = *(tse + 0xB4) | 0x0082;
28.
29.     // Software reset the second PHY chip and wait
30.     *(tse + 0xA0) = *(tse + 0xA0) | 0x8000;
31.     while ( *(tse + 0xA0) & 0x8000 );
32.
33.     // Enable read and write transfers, gigabit Ethernet operation, and CRC forwarding
34.     *(tse + 2) = *(tse + 2) | 0x0000004B;
```

The configuration part is also simple and straight-forward. The TSE IP serves the interconnection between the system and the Ethernet controller. So, setting up carefully its registers, this is translated to the corresponding configuration of the Marvell chip.

5.1.5 Results

By building and compiling the code using "Nios II EDS" we were able to run it through the embedded Eclipse terminal and observe that whatever we send was bounced back by the loopback port through our transceiver. The initial message is printed on the screen, just above the received message so as we can check and confirm that the procedure runs as expected.

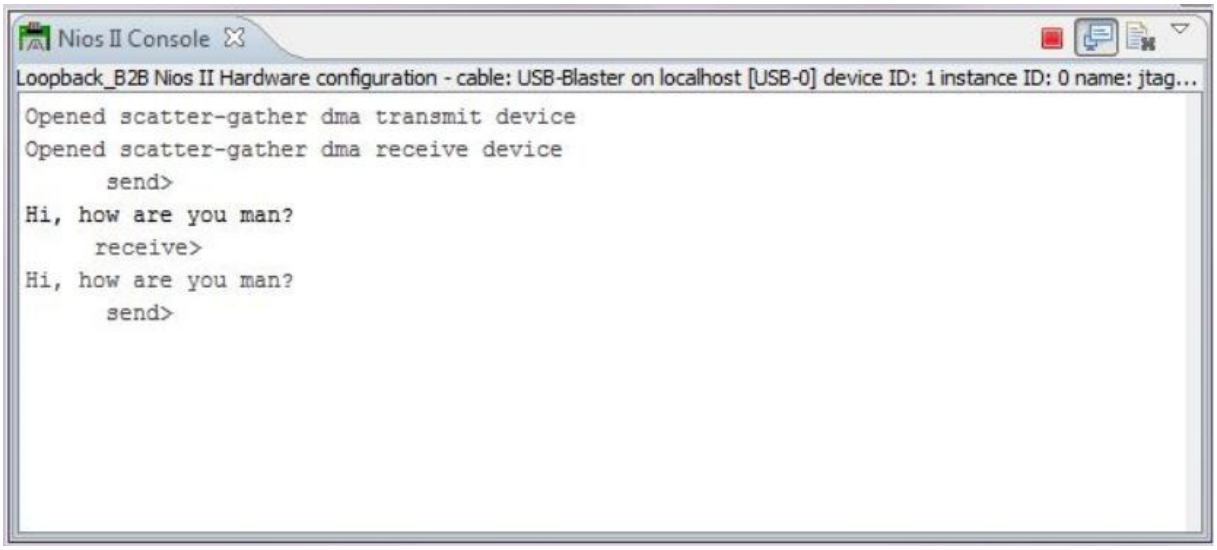

Figure 64: Nios II EDS embedded console instance

Additionally, by doing slight changes on the port-mapping part and on the PHY device addresses of the C code we manage to invert the whole process. In that case the transceiver port was the on-board Ethernet port, while the HSMC-Net card port was the loopback one.

Finally, we tried to connect two independent devices by sending messages to each other causing an interrupt to the transceiver in order to receive and print out the sent data. By removing any connection of the loopback side in the port-map as well as the corresponding C code, we successfully managed to exchange data between the Cyclone V GT Development Board and the Altera DE4 Development Board, using two host computers to observe the messages.

5.2 SFP Loopback Test

5.2.1 SFP Modules

Targeting the increase of the data rate of 1Gb/s, achieved by the simple Ethernet HSMC Daughter Card shown in the prior part, we applied the special Daughter Card for SFP modules. As it was mentioned previously [ch. 3.1.5], this hardware board consists of special ports for SFP modules as well as SFP+ modules. The maximum data rate of the SFP standard is 5Gb/s, so -first- we tried slightly to increase the previous data rate by using a Finisair module [26] up to 2Gb/s transmission speed. In continuation, a faster module was used to push SFP Daughter Card to its limits of 5Gb/s. Provided by Avago [27], a Small Form-Factor Pluggable Plus (SFP+) [14] module running up to 10Gb/s was used successfully to implement a loopback test of 5Gb/s data rate. Both SFP and SFP+ modules do not embed internally any data processing module, but an E2PROM accessible over a 2-wire serial interface for saving customization settings of the device. An SFP transceiver connects the host system via a 20-pin connector.

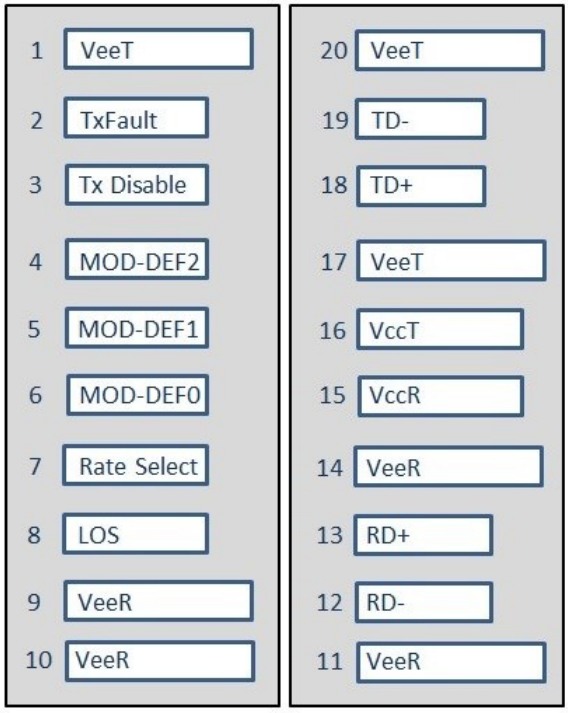

Figure 65: Pinouts on the PCB

The definitions of the SFP module pins are listed below:

Pin	Name	Function
1	VeeT	Transmitter ground
2	TxFault	Transmitter fault indication
3	TxDisable	Optical output disabled when high
4	MOD-DEF(2)	Data for serial ID interface
5	MOD-DEF(1)	Clock for serial ID interface
6	MOD-DEF(0)	Grounded by the module to indicate module presence
7	RateSelect	Low selects reduced bandwidth
8	LOS	When high, indicates received optical power below worst-case receiver sensitivity
9	VeeR	Receiver ground
10	VeeR	Receiver ground
11	VeeR	Receiver ground
12	RD-	Inverted received data
13	RD+	Received data
14	VeeR	Receiver ground
15	VccR	Receiver power (3.3 V, max. 300 mA)
16	VccT	Transmitter power (3.3 V, max. 300 mA)
17	VeeT	Transmitter ground
18	TD+	Transmit data
19	TD-	Inverted transmit data
20	VeeT	Transmitter ground

5.2.2 IP cores

Altera provides three types of IP transceivers PHY implementations suitable for some device families. Those cores are not available for every device because they require specific hard electronic circuits as we will see below. The types supported for Cyclone V device family are:

- Protocol-specific PHY
- Non-protocol-specific PHY
- Native transceiver PHY

All transceiver PHYs above are customizable, yet the lesser options come with the protocol-specific implementations, while the native transceivers implementations provide low level access to hardware having the most configurable part. In our case we will take the middle path implementing a fiber optics channel loopback test by using a non-protocol specific PHY.

5.2.2.1 Custom PHY IP

The Custom PHY IP core was used for the purpose of our implementation

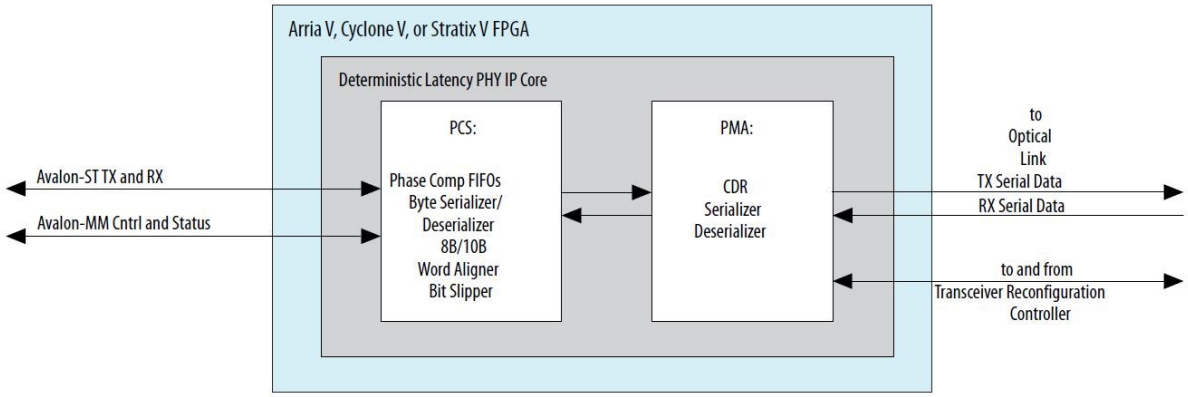

Figure 66: Block Diagram of Custom PHY IP core

Checking the block diagram above we can clearly ascertain that custom IP as all transceiver IPs consists of a **P**hysical **C**oding **S**ublayer (**PCS**), a **P**hysical **M**edium **A**ttachment Sublayer (**PMA**) and of course an **Avalon MM** interface may be needed for special register access.

The **PCS** implements part of the physical layer specification for networking protocols. Depending upon the protocol that you choose, the PCS may include many different functions. Some of the most commonly included functions are: 8B/10B, 64B/66B encoding and decoding, rate matching and clock compensation, scrambling and descrambling, word alignment, phase compensation, error monitoring, and gearbox, depending on the desired interface and data rate.

The **PMA** receives and transmits differential serial data on the device external pins. The transmitter (TX) channel supports programmable pre-emphasis and programmable output differential voltage (VOD). It converts parallel input data streams to serial data. The receiver (RX) channel supports offset cancellation to correct for process variation and programmable equalization. It converts serial data to parallel data for processing in the PCS. The PMA also includes a clock data recovery (CDR) module with separate CDR logic for each RX channel.

An **Avalon-MM** PHY Management module can be used to read and write the control and status registers in the PCS and PMA for the protocol-specific transceiver PHY.

Altera provides dynamic reconfiguration of its transceiver IPs, which is why in every design a Reconfiguration Controller IP is needed even it is not planned to dynamically change the settings of the design. Another IP, which may be embedded to the transceiver or standalone is the Reset Controller IP. The Reset controller is responsible for reliable initialization of both TX and RX channels and can also be modified to meet the design requirements.

Achieving loopback test success by transmitting raw data will ensure that any protocol specified communication is able to be set up. For that reason, we used the non-protocol IP "Custom PHY". Despite the fact that it is less customizable than the other non-protocol IPs like Native PHY; it has enough customizable options to take care of before adding it to any design.

The Custom PHY IP provides the following customizable options:

- ***Operation Mode*** as either Duplex or Transmitter or Receiver
- ***Number of Lanes*** in each direction
- ***Bonding mode*** providing each line with same clock (reducing clock skew) or not by separating clock sources for each channel

Moving on with more practical parts:

- ***Transceiver Interface Width*** specifies the total serialization factor, from an input or output pin to the MAC-layer logic.
- ***PCS-PMA Interface*** width is also important; since it depends on the FPGA fabric transceiver interface width and whether 8B/10B encoding is enabled.

Furthermore, device dependent options are available for:

- ***Data Rate*** determining the desired transfer speed of the link
- ***Reference Frequency*** for the transceiver's embedded PLL.

Those were the general customize options. Next part is the PCS options, which are mainly related to ***synchronization and alignment methods and patterns***.

- ***Word Aligner*** options are available in order to set the channel ***synchronization pattern*** and the ***alignment mode***. The user can choose between three alignment modes. In ***manual alignment*** mode, the user can asserted/de-asserted the enabling signal manually in order to initiate the alignment process. In ***bit slipping*** mode, the word boundary is shifted by 1 bit for every rising edge of the corresponding bitslip signal. Each bit slip removes the earliest received bit from the received data. Finally, the ***automatic synchronization*** mode, controlled by a programmable state machine, allows the user to determine the number of consecutive patterns in order to achieve synchronization, as well as the number of bad received data words before losing synchronization.
- ***Rate Match FIFO*** options part, which compensates for fixing small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing skip symbols or ordered-sets.
- ***8b/10b*** part is responsible for turning on/off the encoder and its control signals.
- ***Byte Order Parameters***, finally, have to be set, in order to let the IP identify the first byte of a packet by determining whether the programmed start-of-packet (SOP) pattern is present. After successful detection it inserts enough pad characters in the data stream to force the SOP to the lowest order byte lane. The last customization tab, related to the reconfiguration process of the IP, was left as it is by default, as long as there is no intention for any reconfigurable design.

A Custom PHY IP is used in all setups [28], [29]. Depending on the needed data exchange interface that is needed to be followed, the PHYs can be connected to the corresponding **M**edia **A**ccess **C**ontrol (**MAC**) IP provided by Altera -if exists- or to the custom design for the desired data frame and encapsulation. In order to check the data transfer integrity of the fiber optical channel, apart from the software Transceiver toolkit, we also used Custom PHY IP standalone to send raw data in a loopback implementation, confirming everything by the use of SignalTap.

5.2.3 Implementation

5.2.3.1 Transceiver Toolkit

Like the Ethernet loopback test, Qsys subsystem was needed to build the loop network systems so as to make the development more efficient, clear and scalable. The system is based on the reference of "Transceiver Toolkit Examples for Stratix® V GX, Arria V GX/GT, Cyclone V GX/GT and Stratix IV GX/GT Devices" on the Altera official website [30]. Because that is the sample, which only works with the loopback connector chip [31] attached to the HSMC port on the Cyclone V GT board, SFP HSMC Card design demands to modify the settings of several Qsys components and add the assignments to the control signals of SFP modules in the top level port mapping. This prototype design, which uses the Transceiver Toolkit [15] was implemented in order to easily check and confirm hardware feasibility and integrity. It also worked as a skeleton design later in the main loopback implementation conveying raw Data through optical fibers.

For the SFP loopback test, a Custom PHY was selected as the transceiver PHY. The Custom PHY will run together with a Transceiver Reconfiguration Controller for working properly after resetting.

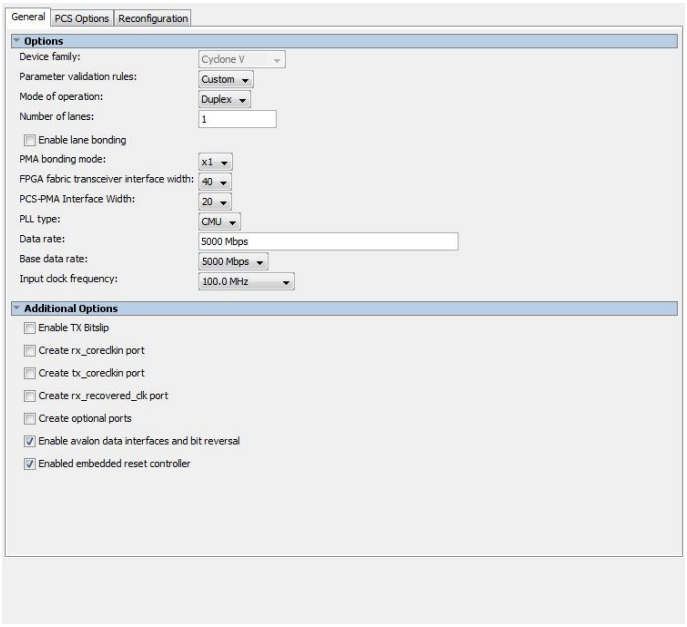

Figure 67: Snippet of Custom PHY Settings 1

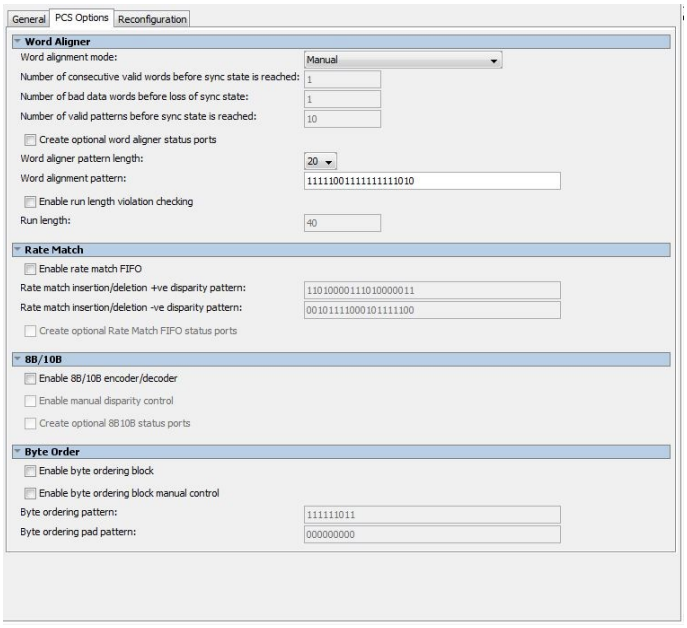

Figure 68: Snippet of Custom PHY Settings 2

On the transmitter side, an *Avalon Data Pattern Generator* was connected to generate pseudo random data and sent through the Custom PHY. On the other hand, an *Avalon Data Pattern*

Checker is placed after the receiver part; in order to check the outgoing data from the Custom PHY whether it is identical to the sent data. Both Data Pattern Generator and Checker are synchronized by an *Avalon-ST Timing Adapter*. The TX side Timing Adapter processes the data from Data Pattern Generator and sends them to the Custom PHY according to the appropriate Tx Clock generated by the Custom PHY. The RX side Timing Adapter processes the data from Custom PHY and sends them to the Data Pattern Checker synchronized by the generated Rx Clock. The data transfer rates used in that experiment were 2 Gb/s and 5Gb/s, due to the fact that it is recommended to ensure that data transfer/acquisition designs work well on both high and low speeds.

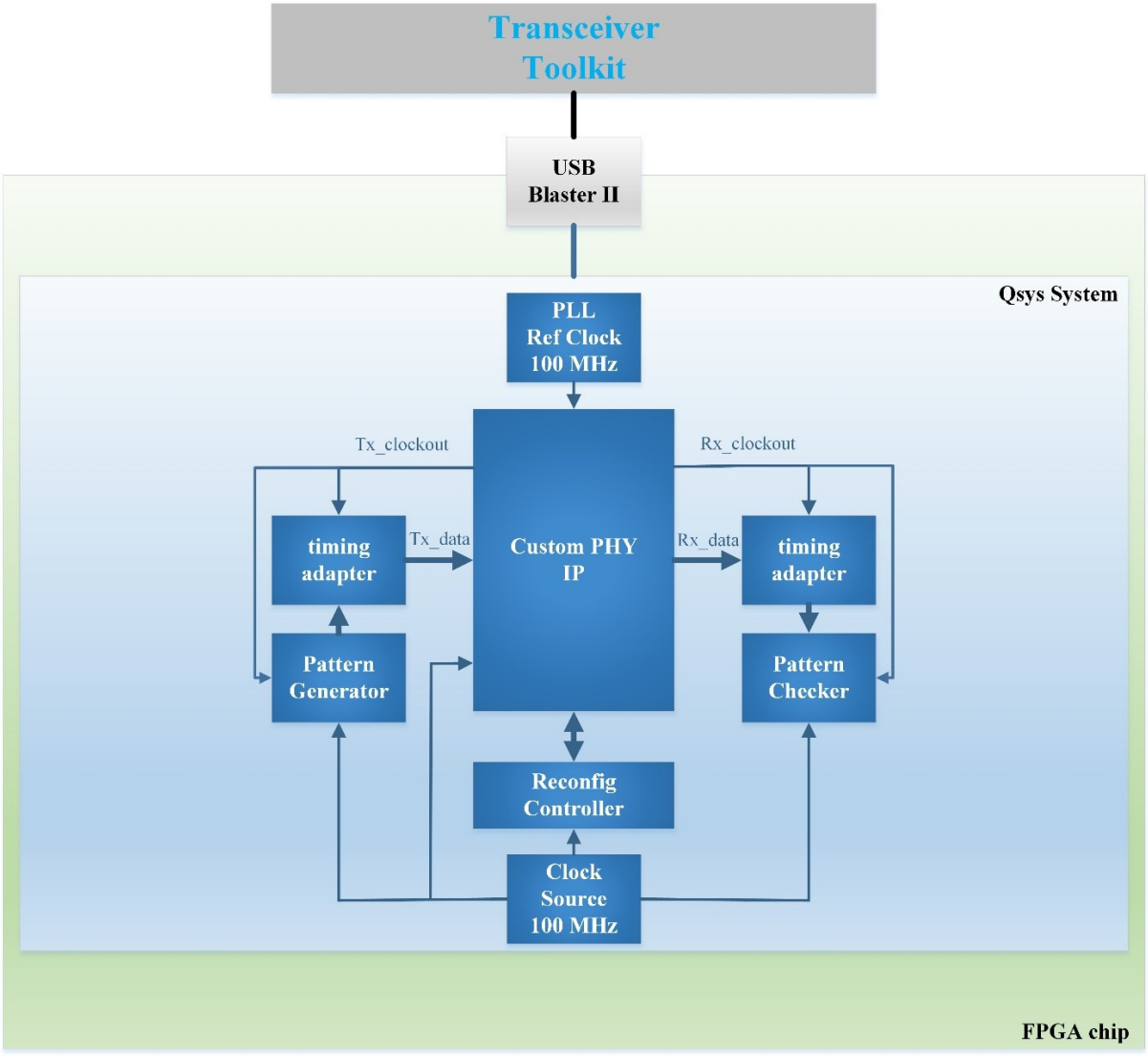

Figure 69: Qsys SFP Loopback System Block Diagram

Once the loopback test design is implemented, the Altera Transceiver Toolkit should be applied to monitor, analyze and configure the communication.

Now, the transmitter and receiver channels are all in correct state. Users can change the default settings of the "**Test pattern**" and "**Generator / checker mode**" in order to test the quality of the loopback link. In receiver channel part, under the "**Checker**" title, the "**Number of bits tested**", "**Number of error bits**" and "**Bit error rate**" values are useful for assessing the performance of the communication.

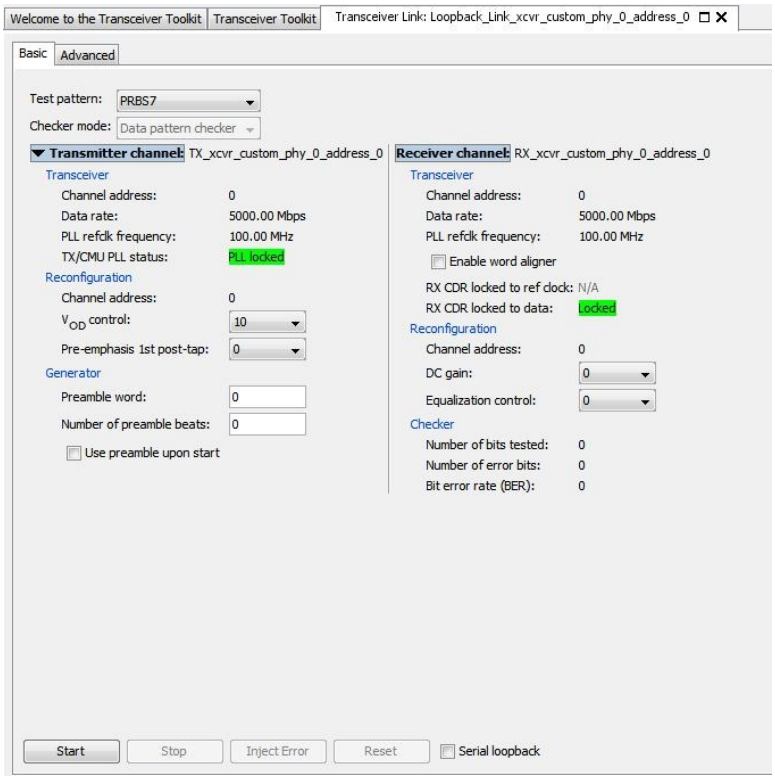

Figure 70: Basic page for Transceiver Link

During the test, the loopback communication was kept for 30 minutes. The link turned to be stable and reliable because the "**Number of error bits**" and "**Bit Error Rate**" are both 0. To see how the link reacts to disconnection, the cable was plugged out some times.

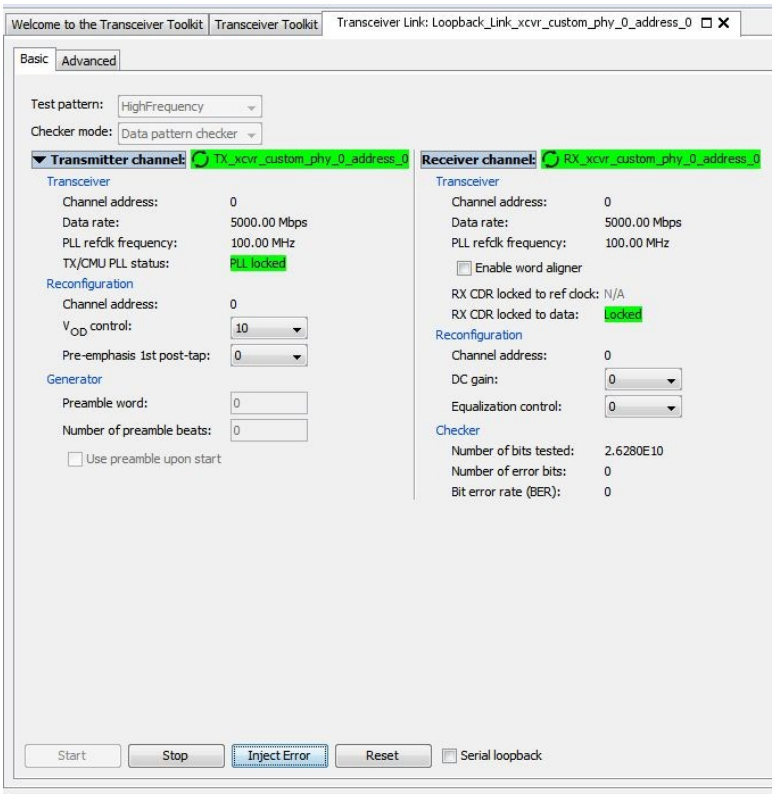

Figure 71: Test the link communication at 5000Mbps data rate

The accumulated "**Number of error bits**" and the accumulated "**Bit Error Rate**" increased to 6.4255E5 and 2.9401E-6 respectively, after the optical fiber has been plugged out for some seconds. Besides, "**Inject Error**" button was pressed sometimes for observing the similar results caused by the errors. Clicking the "**Inject Error**" one time generates one error bit in the communication link.

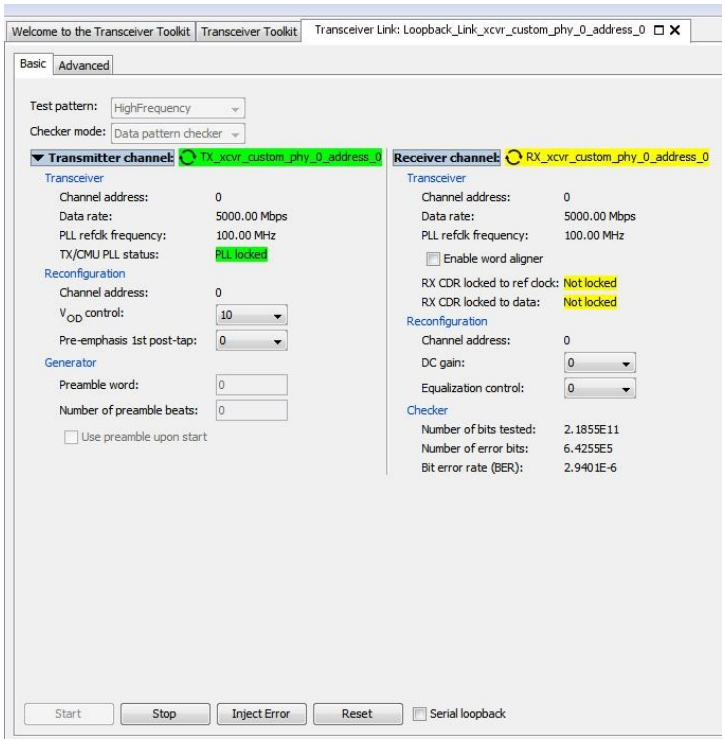

Figure 72: Link communication after plugging out the optical fiber

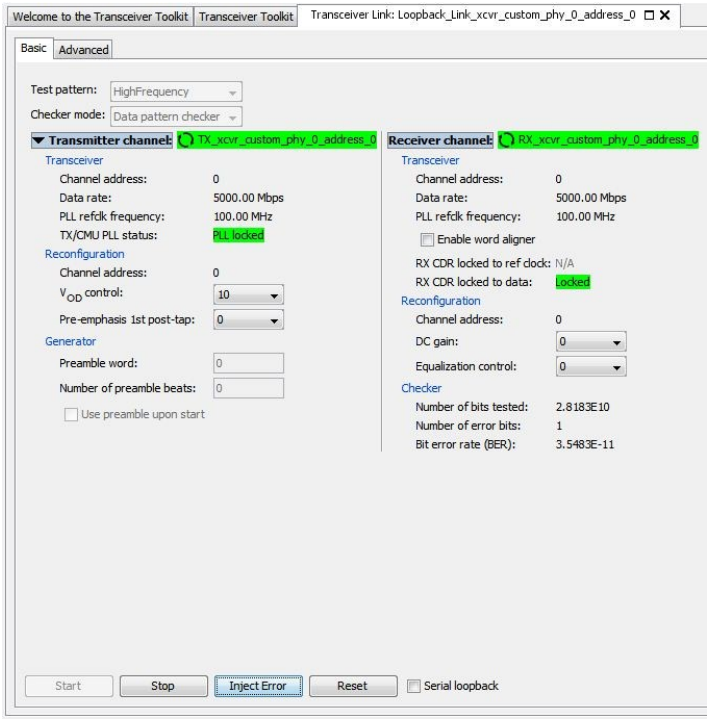

Figure 73: Link communication after injecting error

5.2.3.2 Raw Data Loopback Test

To make the system more practical the *Data Pattern Generator*, *Data Pattern Checker* and *Timing Adapters* were removed. Since the correct functionality of the Daughter Card, the IP and the optical fiber cables was confirmed, a non-protocol raw data transfer had to take place in order to prove that the design can encapsulate any data format/interface that may be needed. The previous *Generator* and *Checker* units were replaced. A combination of *lpm_mux*, *lpm_constant* and *lpm_counter* took the place of the data generator and the SignalTap II Logic Analyzer replaced successfully the data checker. Additionally, the tested data interface was not only -of course- the simplest version of 8 bit at first, but also 2 and 4-words (16 and 32 bits) incoming parallel data also did successfully pass the loopback test in 3.5 Gb/s and 5 Gb/s data rate respectively.

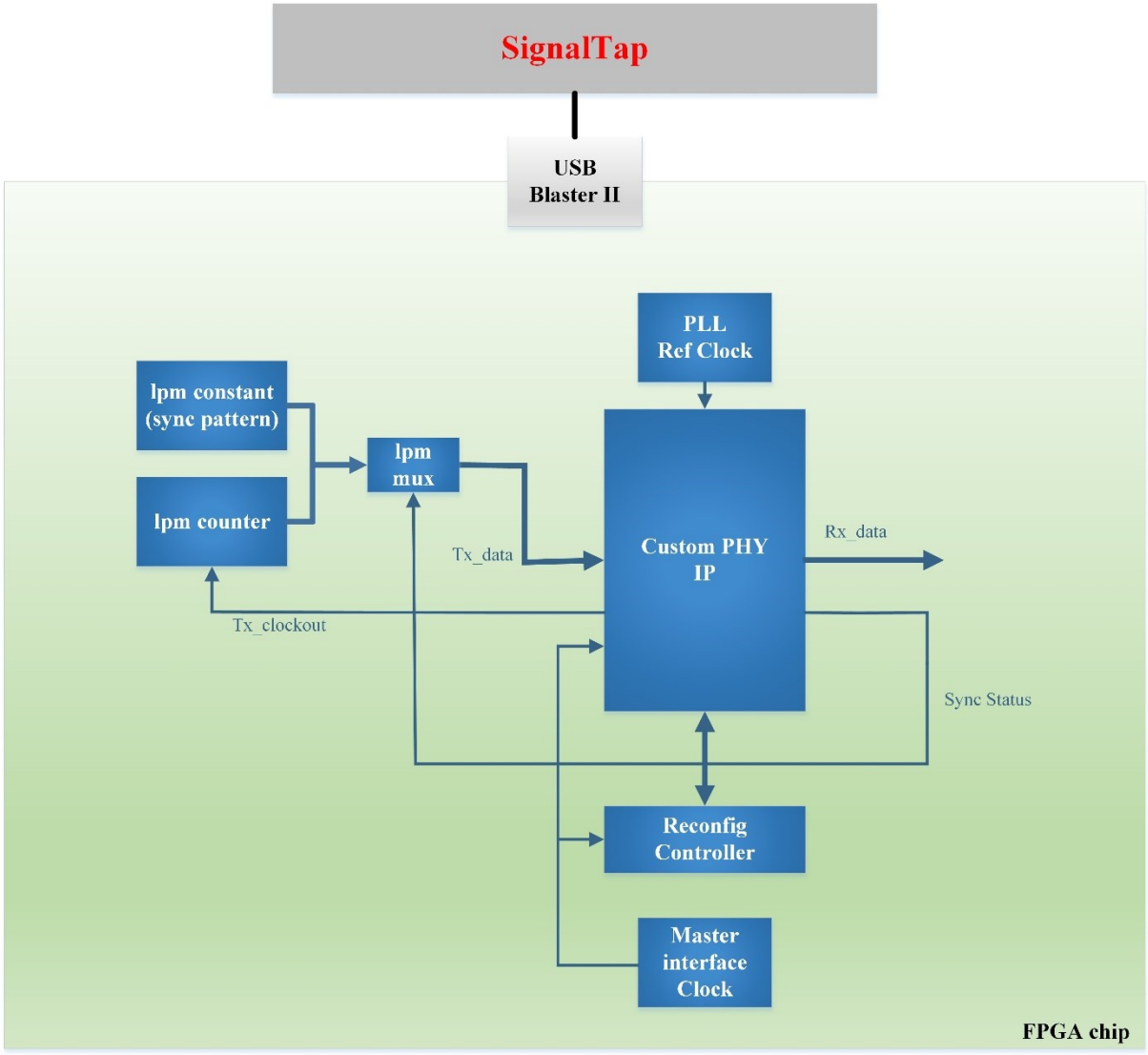

Figure 74: Raw data SFP Loopback System Block Diagram

As it was mentioned previously, in high speed serial links data travels without a clock. In that case, it is needed special treatment for sending raw data. The first important difference is the enabling of 8b/10b encoder/decoder for transmitted signal DC balance. That means, critical and cautious configuration for alignment and ordering of data. Especially, in case of parallel

data which enter the transceiver IP design having bigger width than the PCS-PMA interface. For that reason, as it is clear on the instance below, special comma character has to be set as training character for the channel in order to achieve synchronization or recover it in case of loss. Inserting and deleting patterns accordingly were set for sake of skipping occasion, in order to compensate small clock differences between the upstream transmitter and the local receiver clocks. In our case the special order pattern is the alignment comma character for the sake of channel's alignment and ordering by a neutral character at the initialization of the channel, allowing the normal transmission of any data character. In simpler words, a pattern of an ordered set Dx.y/Kz.w is being sent for synchronization and alignment having the control character as a dedicated pattern. Finally, byte ordering settings were customized for placing the special pattern always in the LSB part of received data as an extra measurement of alignment-ordering mechanism, whenever again the width of transmitted data interface is bigger than PCS-PMA interface (See figure 76). After successful channel initialization any kind of Dx.y data can follow either in 16-bit mode interface or 32-bit (See figure 75).

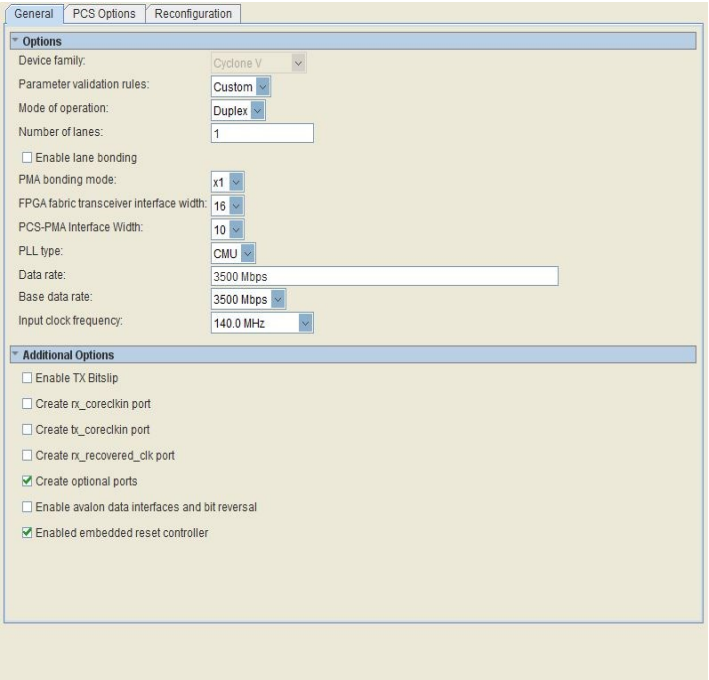

Figure 75: Custom PHY Settings 1

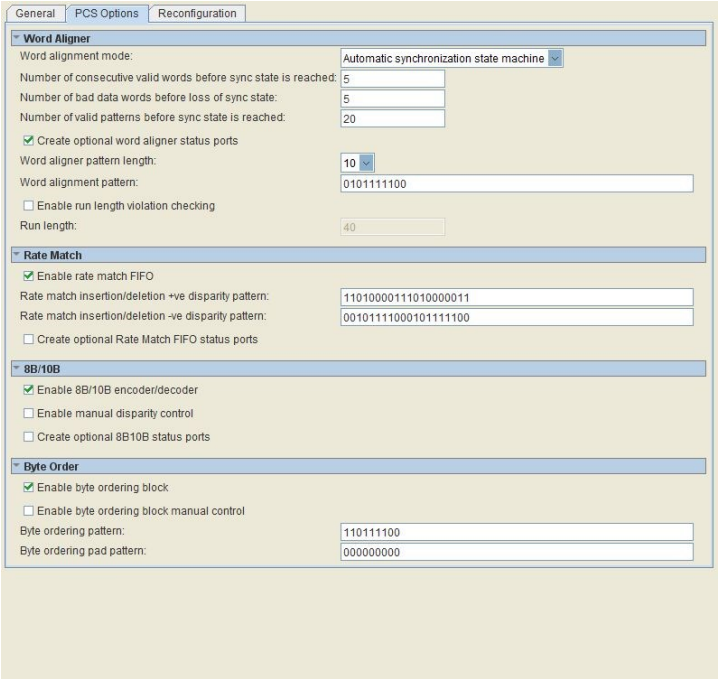

Figure 76: Custom PHY Settings 2

By using the setting above and the respective synchronization status export pin of the Transceiver IP as the select in the transmitted data multiplexer, it is possible to plug out the fiber optic cable anytime, creating errors and synchronization loss, yet recovering link successfully by plug in again. This is happening, due to the fact that the mux module sends the synchronization sequence when synchronization status de-asserts, until channel is again capable for data transmission.

5.2.4 Results

The Raw Data System Block Diagram (see figure 74) worked with different data width at 1.875 Gb/s, 3.5 Gb/s and 5 Gb/s. The *constant* block holds the training pattern each time used for synchronization reason. It is simply a comma symbol or a data symbol followed by a comma, in case of 32-bit interface and 5Gb/s rate used as searched pattern within a bit-stream; so as this pattern is found the channel asserts the synced status pin and performs byte ordering.

Once the synchronization is done, the *mux* will shift the output content from *constant* to *counter*, which increments its value by 1 at the frequency of *tx_clk*. After programming the device and modifying the frequency of the oscillators on the development board, loopback test behavior can be verified by clicking the "**Autorun Analysis**" in SignalTap (See figures 77 and 78).

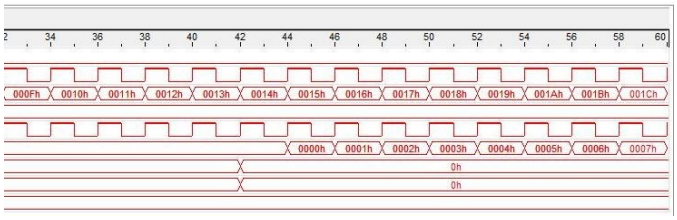

Figure 77: SignalTap of 16-bit interface at 3,5Gb/s data rate

Figure 78: SignalTap of 32-bit interface at 5Gb/s data rate

SignalTap instances above verify the correct implementation of complex and fast 3.5 Gb/s and 5Gb/s interfaced by checking the similarity of transmitted and received data. Although SignalTap is better than Testbench mechanism to observe your design in real time it is not that consistent in high speeds followed by high data rate.

Through this experiment Cyclone V GT development board were used to do the loopback test via SFP modules and optical fibers successfully. There are two important positive outcomes by this part. First is the introduction and usage of the Transceiver Toolkit, combined in the Quartus II software, in order to check channel integrity and general device behavior before implementing your own design and interface. Secondly and most important is the understanding of high speed serial links and the fundamental rules for data transmission and clock recovery over fiber optics and differential signaling in general.

5.3 XAUI Loopback Test

5.3.1 10 Gigabit Media-Independent Interface

Previously, in this thesis report, MII and RGMII interfaces were presented [ch. 5.1.2]. For this part, it is needed to introduce an offspring of the former interfaces. It took its name by the Latin representation of number 10(ten) 'X' followed by the Gigabit Media-Independent Interface (XGMII).

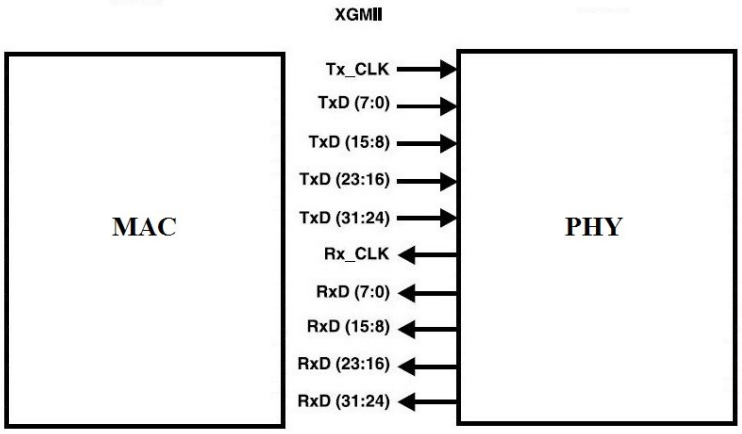

Figure 79: XGMII

Figure 79 depicts the high level interconnection between a MAC and a PHY following XGMII. The XGMII supports full duplex operation at a rate of 10Gb/s between the MAC and the PHY. Transmitting and receiving directions are independent of each other, containing 32-bit data path each, as well as clock and control signals. Totally the interface is 72 bits wide. As it is clearly understandable by the picture below data is being transferred in four lanes. It is clocked by a specific clock of 156.25 MHz using both edges for transmission as it is defined in Clause 46 of the IEEE 802.3-2008.

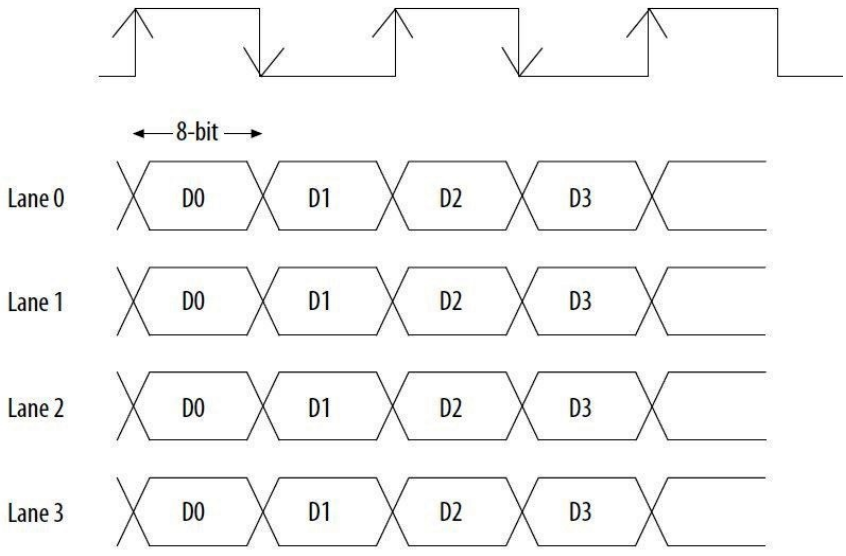

Figure 80: XGMII data transfer

While using XGMII, which provides a 10Gb/s payload, a real challenge arises in routing the bus longer than the recommended short distance of 7 cm. The separate transmission of clock and data coupled with the timing requirement to latch data on both rising and falling edge of the clock makes it difficult and challenging. For this reason, XGMII can not be considered as a practical interface for chip-to-chip, board-to-board, chip-to-optical module applications, since distances may exceed recommended distance. Consequently, the XGMII bus puts many limitations on the number of ports that may be implemented on a data transfer system.

5.3.2 10 Gigabit Attachment Unit Interface

Following the XGMII naming pattern 10 Gigabit Attachment Unit Interface abbreviation converts number 10(ten) to its Latin symbol in order to form XAUI. This high speed attachment unit interface was developed to use the features of XGMII, which were introduced above. It is also a full duplex interface that uses 4 self-clocked serial differential links in each direction achieving 10Gb/s data throughput. Each link operates at 3.125 Gb/s in order to serve both data transmission and the overhead associated with 8b/10b encoding/decoding. The self-clocked nature of the links helps eliminating any skew concerns between clock and data, and also extends the recommended distance of XGMII up to approximately 50 cm. The conversion of XGMII signals to XAUI takes place in the so called XGMII Extender Sublayer, shortly called as XGXS.

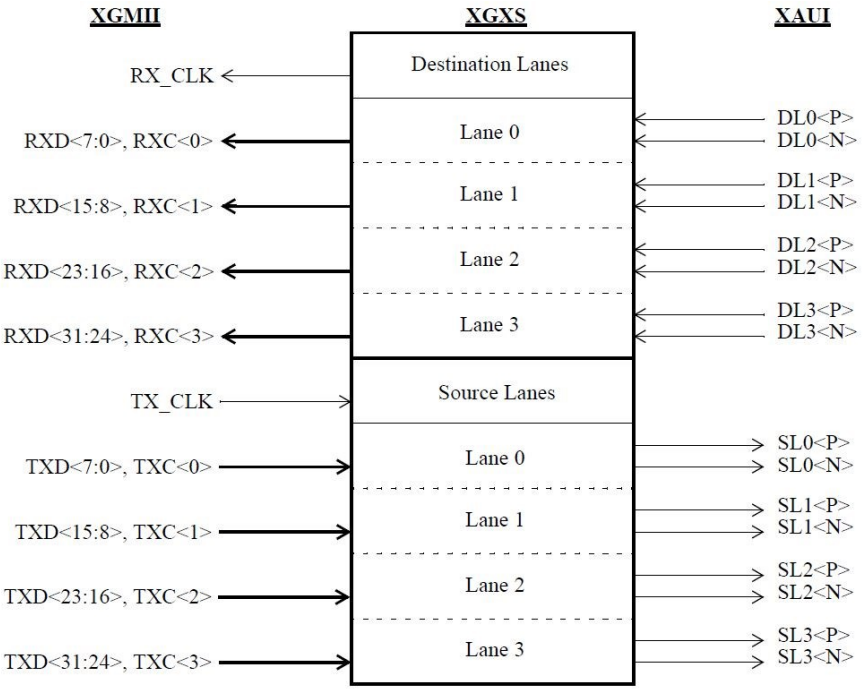

Figure 81: XGMII to XAUI at the XGXS

As seen in figure 81 each data stream is transmitted across a single differential pair running at 3.125 Gb/s. At the receiver side the clock is recovered from the incoming data stream, it is decoded and then mapped back to the 32-bit XGMII format. This procedure results in the reduction of the 72-pin interface to 8 differential pairs of 16 pins.

The 10 Gigabit Ethernet link, defined in the IEEE802.3ae-2002 specification, has a specific physical layer implementation provided by XAUI. The XAUI PHY uses the XGMII to connect to the IEEE802.3 MAC and Reconciliation Sublayer (RS). According to the IEEE

specification, XAUI PHY link is required to support a 10 Gbps data rate at the XGMII interface and 4 lanes at 3.125 Gbps at the Physical Medium Dependent interface.

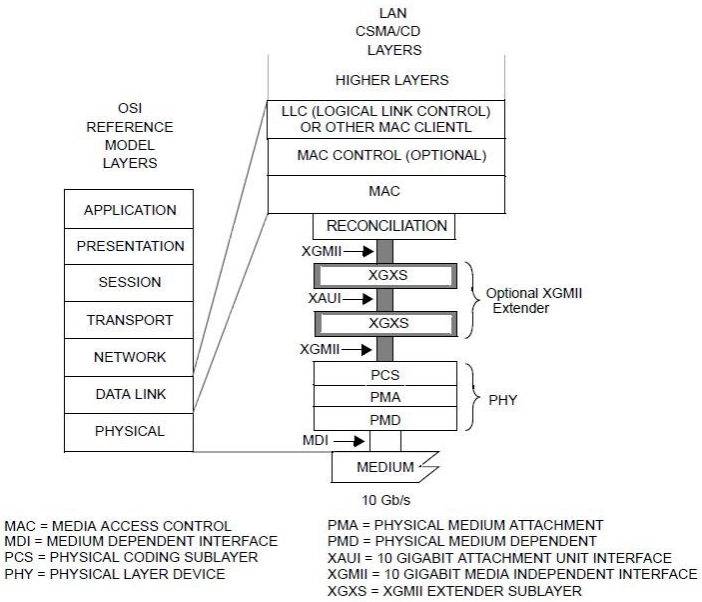

Figure 82: XAUI and XGXS relationship to the ISO/IEC Open Systems Interconnection (OSI) reference model and the IEEE 802.3 CSMA/CD LAN model

5.3.2.1 Signal levels

The XAUI is characterized as a low swing AC coupled differential interface. Interoperability between component operating from different supply voltages is being allowed by the AC coupling. Additionally, improved electromagnetic interference (EMI) and noise immunity is provided by low swing differential signals like XAUI signals.

5.3.2.2 Amplitude and swing

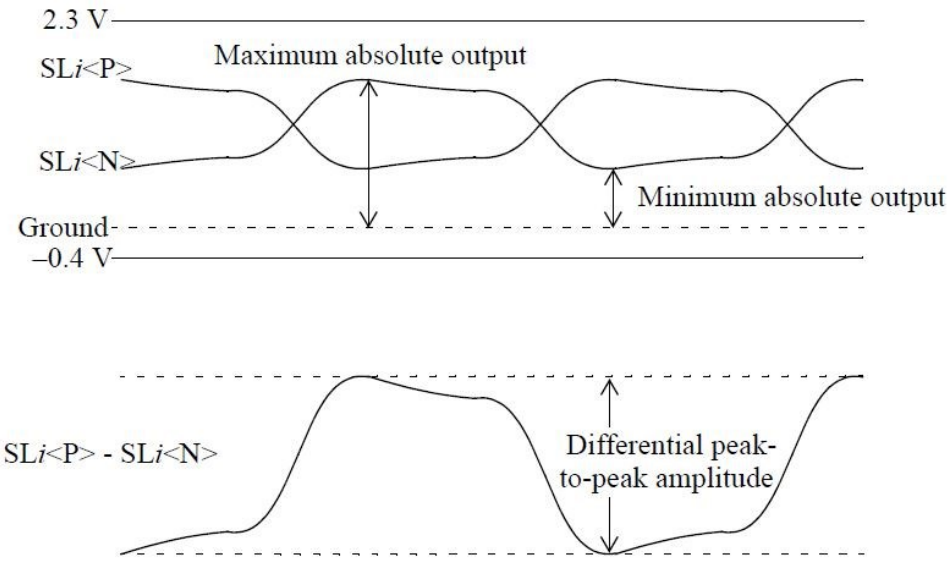

Figure 83: Output voltage limits and definitions [Li<P> and Li<N> are the positive and negative sides of the differential signal pair for lane i (i = 0, 1, 2, 3)]

The Differential output amplitude upper limit is 1600mVp-p taking into consideration any transmit equalization. Since the destination side is AC coupled, no DC-referenced logic levels are defined. Output voltage absolute value level shall be between -0.4V and 2.3V with respect to ground. Very important for the transferred signal and channel integrity is the operating BER limit of the receiver as well as the channel peak-to-peak total jitter amplitude tolerance. The acceptable values are 10^{12} and 0.65 UI respectively.

Parameter	Value	Units
Baud rate	3.125	GBd
tolerance	±100	ppm
Unit interval (UI) nominal	320	Ps
Receiver coupling	AC	
Return loss		
Differential	10	dB
Common-mode	6	dB
Jitter amplitude tolerance	0.65	UI_{p-p}

Table 11: Receiver Characteristics

5.3.2.3 Functional Specifications

The XAUI uses the common robust 8B/10B transmission encoding/decoding technique as the majority of the interfaces. Converting 8-bit to 10-bit data, there is the ability to create extra code groups to form the so called ordered sets, which may not correspond to any possible 8-bit data stream. Some of the extra code groups/ordered sets, as they are described in part 48.2.2 in Clause 48 of the IEEE 802.3-2008, are used for control signaling; such as start of frame, end of frame, channel idle, link configurations and so on. Special Control characters are used during the Inter-Packet Gap (IPG) time and during idle periods to continuously succeeding word and lane alignment for the XAUI interface. A single bit indicator is needed to be attached for every 8-bit long data or control symbol. This is for distinguish data streams (control bit = 0) from control streams (control bit = 1).

Control bit	Data(in hex)	PCS code-group	Description
0	00 through FF	Dx.y	Normal Data Transmission
1	07	K28.0/K28.3/K28.5	Idle
1	FB	K27.7	Start Transmission
1	FD	K29.7	Terminate Transmission
1	FE	K30.7	Error

Table 12: XAUI special symbols

Essential procedures of frame synchronization and lane alignment are two-step processes. Code group synchronization is achieved on each lane upon reception of three ordered sets for the lane. Special and common pattern called “comma” initiates the XAUI receiver procedure for frame alignment upon the incoming stream. Each lane has to achieve proper alignment so as to consider the channel aligned. However, each serial transmission lane operates independently and can often come out of alignment with respect to one another. That is why another special control character is used for lane alignment attainment. Despite the existence of those characters, it doesn’t mean that timing and ordering constraints should not be taken into consideration about the exact time spot and order of transmitted control groups/order sets.

As far as the clock differences that often exist, XAUI handles them by monitoring the difference between incoming and outgoing data rates, each XAUI connection can add or delete specific control words in the IPG, to balance the data rate at each connection without effecting lane disparity. All those functions are not related and dependent on any upper or lower layer, labeling XAUI as a self-managed interface.

The Optional XGMII Extender as it is called in figure 82 has some main characteristics and can be summarized in a list of major concepts, which is worth to be mentioned because of their help in understanding the functionality of XGXS and XAUI.

Characteristics:

- Simple signal mapping to the XGMII
- Independent transceiver data paths
- Differential signaling with low voltage swing
- Self-timed interface provides jitter control to the PCS.
- Four lanes transferring the XGMII 32bit data and control
- Utilization of 8B/10B coding
- Shared technology and functionality with other 10Gb/s interfaces and Ethernet blocks respectively

Major Concepts:

- Inserting the optional extender between RS and the PHY, it successfully extends the physical reach of XGMII and efficiently reduces the interface pin count.
- The XGMII is organized into four lanes each of them conveying a data octet or control character on both edges of the respective clock. XGXS source converts bytes on an XGMII lane into a self-clocked, serial, 8B/10B encoded data stream. Then each XGMII lane is transmitted across the XAUI path.
- The XGXS source converts XGMII data and control characters into an 8B/10B code sequence. The XGXS destination recovers clock and data from each XAUI lane and deskews the four XAUI lanes into the single-clock XGMII.
- The XGXS receiver adds to or deletes special non-data characters as needed for clock rate disparity compensation prior to converting the stream back into XGMII based data.

5.3.3 XAUI PHY IP Core

5.3.3.1 Block Diagram

The Altera XAUI PHY IP Core implements the IEEE 802.3 Clause 48 specification, which was shortly introduced and described before [ch. 5.3.2]. The high level block diagram of the IP seems familiar because it has enough common parts with the used PHY on the previous implementation of the SFP daughterboard.

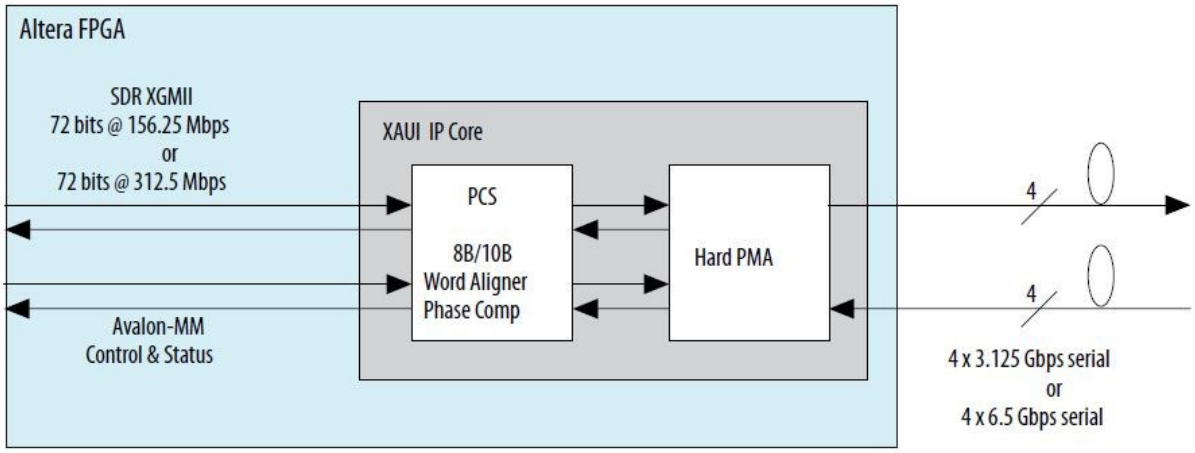

Figure 84: XAUI PHY IP Core

Explaining a little bit more of the block diagram above, it becomes clearer that the XAUI Transceiver datapath consists of PCS and PMA parts like the majority of transceivers and more specifically its PMA part is implemented using hard logic saving FPGA's resources. In high speed transceiver systems, where accuracy and timing constraints play the most important role, it is more preferable all parts being deterministically implemented in hard logic by the usage of special PLLs for clock and data recovery.

For that reason, we decided to use another FPGA -besides the initial, which was used in the previous implementations- capable of implementing everything in hard electronics. Additionally, the new device offers the option to use either Clock Multiplier (CMU) or Auxiliary Transmit (ATX) PLL type. The first and common variation has a larger frequency range and provides clocks to all the transmitter channels within the transceiver block. On the other hand, the ATX option offers very low-jitter high-frequency clocks and is designed to operate in a narrow frequency range. The ATX PLL distributes high-speed clocks through the clock divider, which also produces low-speed parallel clocks. The ATX PLL allows you to clock and bond all of the transceiver channels with a single PLL. Another advantage of the ATX PLL is that it does not use a transceiver channel, while the CMU PLL does. The implication of losing one channel per bonding is to end up with less available active channels for data transmission than the initial addressed channel number.

Although the dedicated block diagrams do not seem to differ much; the importance of stable and low jitter PLL integration in high speed link designs becomes more noticeable in systems with more than one transceiver.

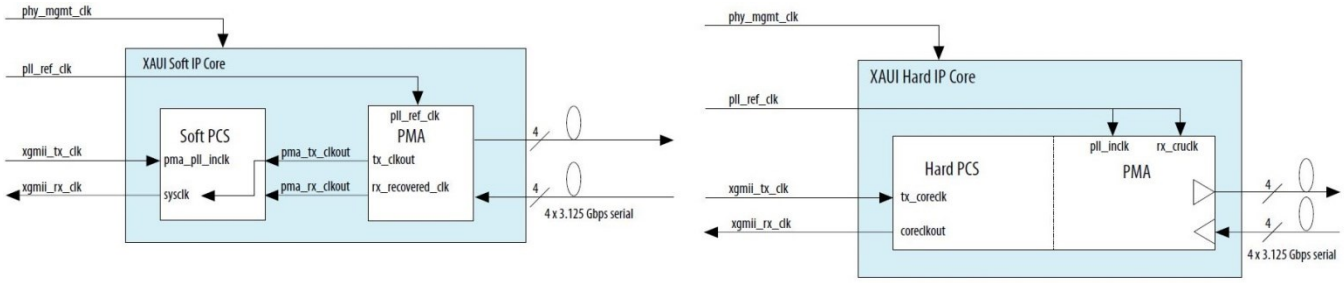

Figure 85: Soft XAUI vs. Hard XAUI

As it will be described in the implementation part [ch. 5.3], either in Cyclone V GT FPGA (soft PCS) or in DE4 Development Board FPGA (hard PCS) 10 Gbps, loopback test implementation was successfully developed. Slight differences appear on the configuration of the XAUI IP in a Stratix IV FPGA, where the customizable options are extended.

Summing up the main signals going in and out of the IP, we can say that totally 72 parallel bits travel through the transceiver on each side. To be more specific 64 bits correspond to data stream and 8 bits are control signals. Considering that everything is equally divided to 4 lanes, each one of the lanes conveys 18-bit (16-bit data, 2-bit control) wide streams. Apart from the parallel form of data streams, the XAUI PHY provides the analogous serial data interface, which is connected to the transceivers side of the FPGA directly. Another important part is the reference clock input port, where it should be connected to a low jitter clock source of exactly 156.25 MHz frequency. Additionally, as it will be described in the implementation part, the PHY has some configurable options which can be set during its generations. However, the user can change some of the settings after the generation procedure, by an Avalon Management Interface [24], which provides access to some registers of the created IP. Finally, as every other transceiver IP, it outputs some status and control signals, to verify its correct functionality or for connecting them to related designs.

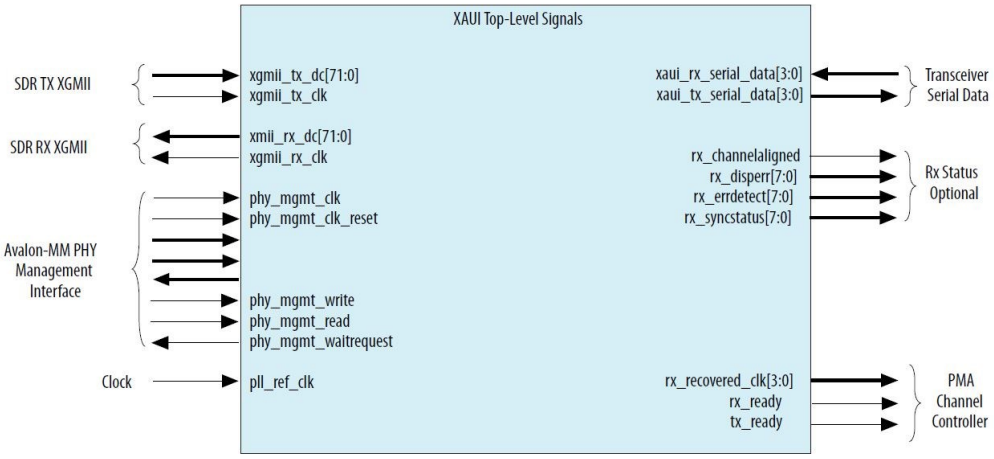

Figure 86: XAUI Interface signals

5.3.3.2 Transceiver Datapath

According to the definition of XGMII in Clause 46 of the IEEE 802.3-2008 specification each of the four XAUI lanes is required to transfer 8-bit of data and 1-bit wide control code at both edges (DDR) of a 156.25 MHz interface clock. XAUI PHY IP does not support the data transfer illustrated in figure 80, yet it allows the transferring of 16-bit data and 2-bit control code on each of the four XAUI lanes, only at the positive edge (SDR) of the interface clock.

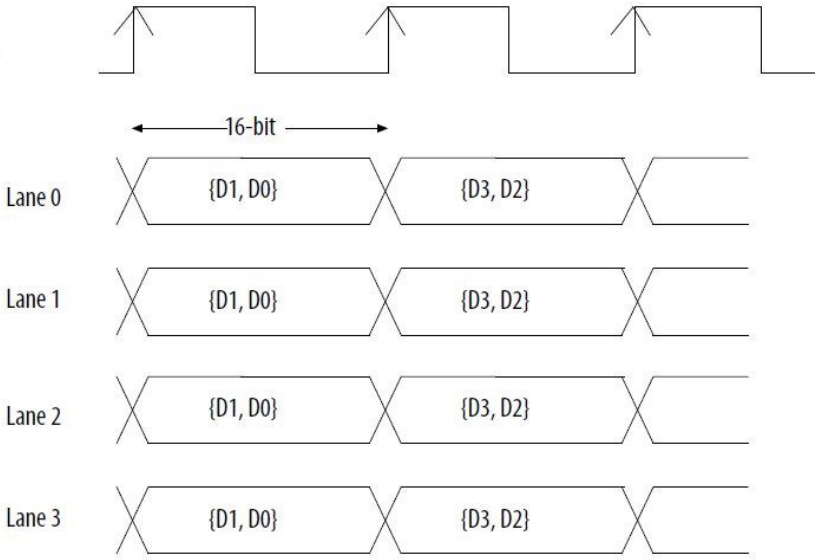

Figure 87: XAUI PHY data transfer

In this case we can consider every lane having a unique, but at the same time identical to every other lane datapath conveying 16 bits of data.

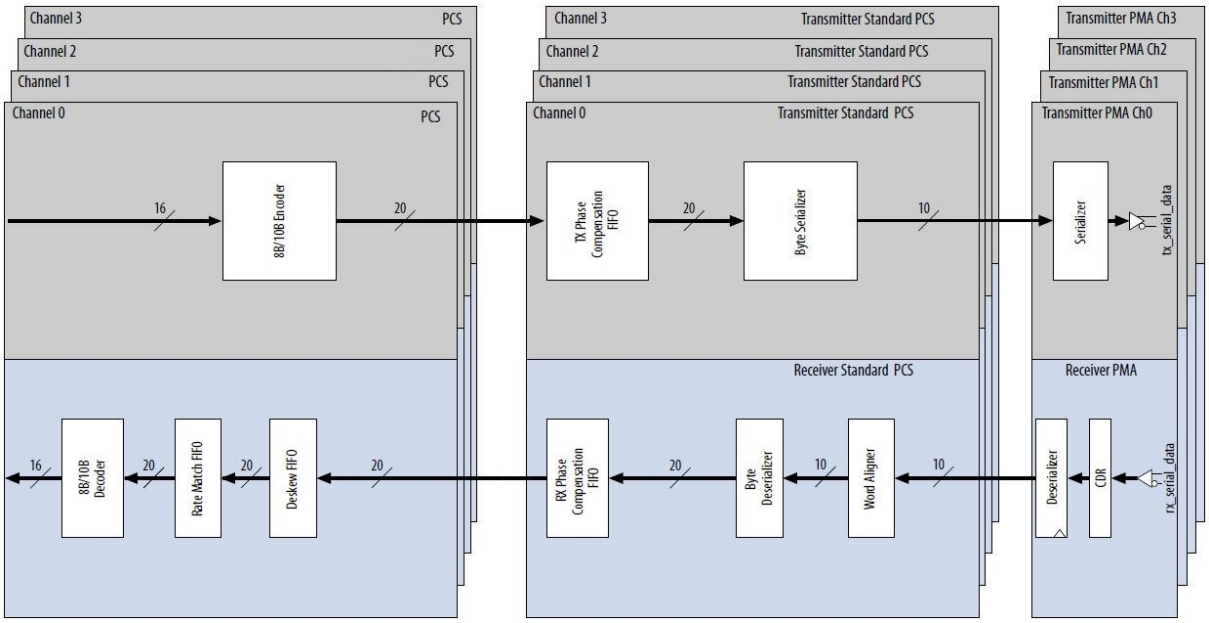

Figure 88: XAUI PHY Datapath

Combining the IEEE 802.3-2008 specification followed by XAUI PHY IP Core and the section [ch. 4.2], where a 10 Gigabit transceiver datapath was described and analyzed, it is easy to understand the structure of PHY's PCS and PMA parts as well as their purpose.

8B/10B Encoding Decoding

Each of the 4 lanes supports an independent encoder/decoder for limiting the maximum number of consecutive 1s and 0s in the serial link stream to five, thereby ensuring DC balance and appropriate transmissions for the CDR to maintain a lock in the incoming data.

Synchronization

By implementing the synchronization diagram, defined in IEEE 802.3-2008 specification Clause 48, receiver Word Aligner block synchronizes to a valid word boundary.

Deskew

Likewise, the synchronization part, Deskew FIFO follows the corresponding diagram defined in IEEE 802.3-2008 specification Clause 48. Deskew process starts only after every line is synchronized into a valid boundary.

Clock Compensation

The clock compensation operation hosted by the Rate Match FIFO adds or removes special skip character columns from the stream to compensate the clock difference between the remote transmitter and the local receiver.

5.3.4 Implementation

The third loopback test is closely connected with the previous one, because the purpose is the same. The only and essential difference is that the data rate is 10 Gb/s. The higher speed you need to achieve in serial links the more accurate and flawless protocol is needed. Following a "secure" and "sure" path we chose to implement the XAUI interface in order to reach double digit transfer speed. Initially, the Cyclone V GT device passed successfully the test of XAUI frontend driver for the DUAL XAUI daughterboard. Later the design was moved to the more appropriate Stratix IV GX DE4 FPGA. Likewise, the implementation was successful as we will see in the figures below. The different part was only the IP generation by the number of customizable options provided in the more suitable DE4 device and the recommended clock source for the register access of the XAUI IP (50 MHz-DE4/125 MHz Cyclone V GT).

For that reason, the top level's block diagram is common on both devices.

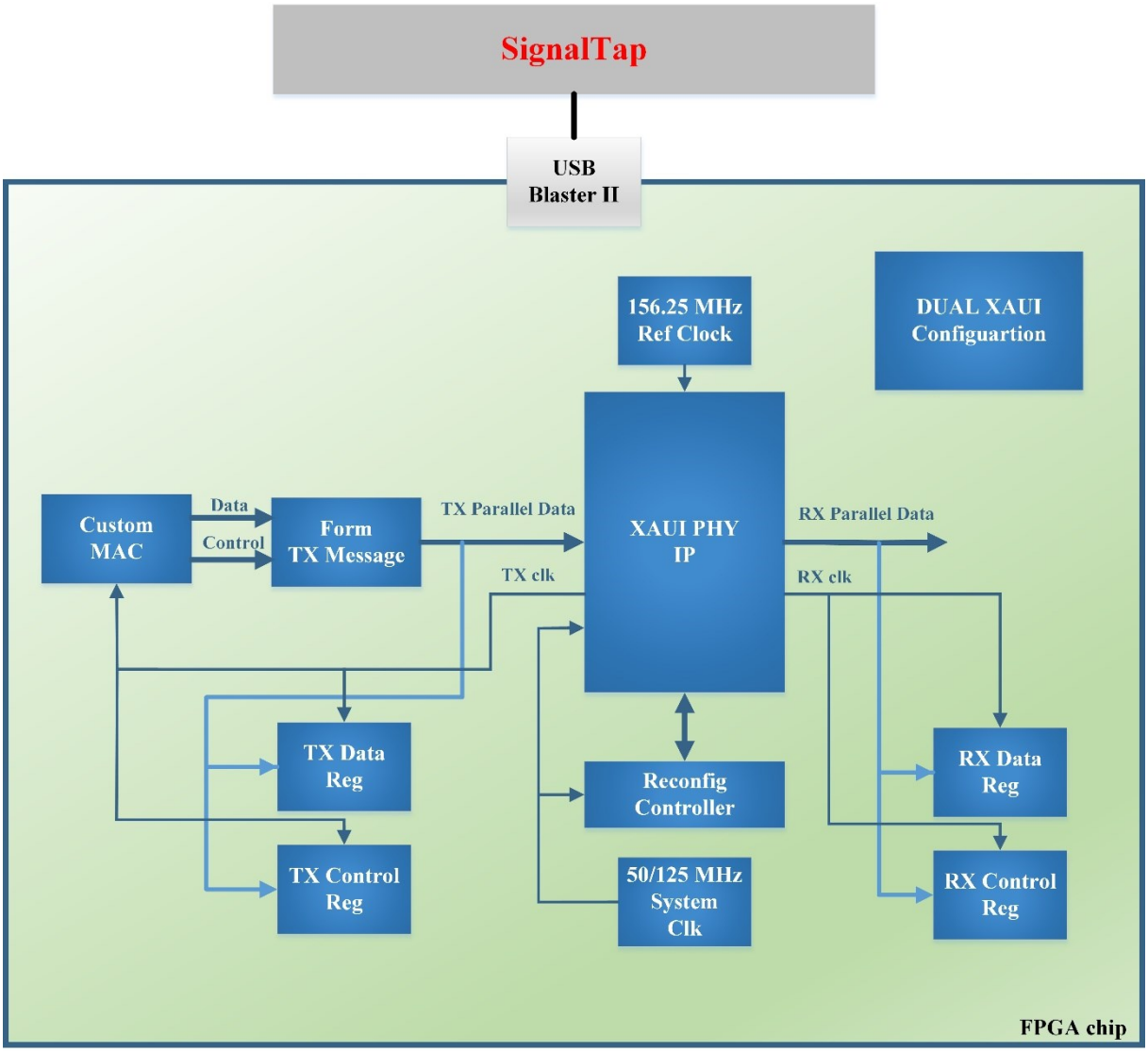

Figure 89: Top level block diagram

The general rule followed in all our implementations is to generalize the problem we face and try to solve it by generating a solution, which proves that any specific problem can be solved.

So again we created our own MAC, following XAUI PHY IP Core specifications only, which sends raw data to check systems functionality.

The custom PHY triggered by the optical fiber loopback connection presence starts a loop sending 30% consecutive raw data, and 70% of idle special character "0x07" according to table 12. On the reset state or before any trigger it sends continuously "0x07". This special character is the XAUI character that is being translated into the corresponding ordered sets or symbols in XGMII interface for synchronization and alignment as it is more thoroughly presented in IEEE 802.3-2008. That is why the transmission procedure loops from data to special character, assisting the channel to remain synchronized and its lanes aligned. Additionally, this is the simplest way to achieve successful channel recovery after global reset or optical fiber reconnection. On the other hand, the transmitted raw data is an incremented 8-bit counter concatenated 8 times (to form 64-bit wide data) in every transmission. By this way we were able to observe in SignalTap the loopback test time-period. The MAC produces two different outputs, one 64-bit wide for data stream and one 8-bit long for control signals. This is happening because it is more convenient to set the correct value to control signals when your stream contains a mixture of data and control characters. The 64-bit long stream is divided in 8 octets, assigning each one bit of control signal to its corresponding octet like in figure 90.

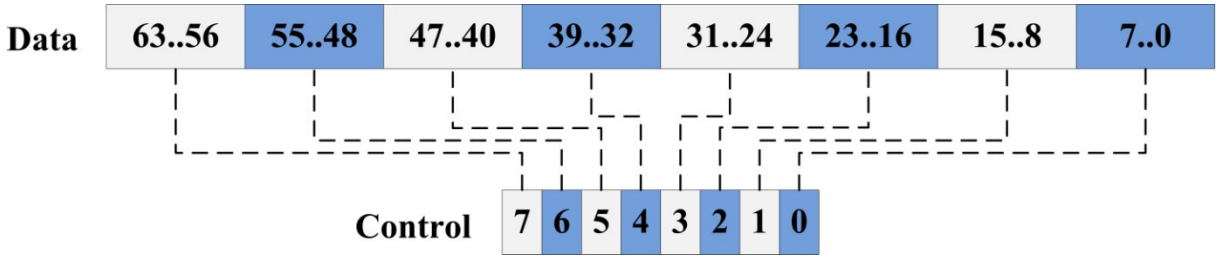

Figure 90: Data Interface

The Custom MAC has only one odd part, caused by the ability of XAUI IP to support DDR XAUI, which double the data rate using both reference clock edges. Because of this variant, the special start control character "0xFB" has to be aligned to either byte 0 or byte 5, as it is illustrated bellow. As far as the ending symbol "0xFD" is concerned, its position is not affected.

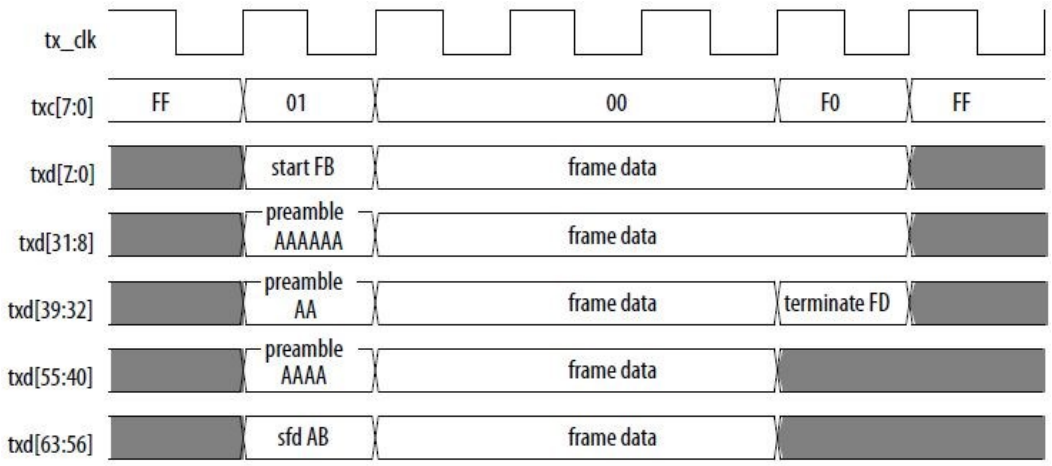

Figure 91: Byte 0 Start of Frame Transmission Example

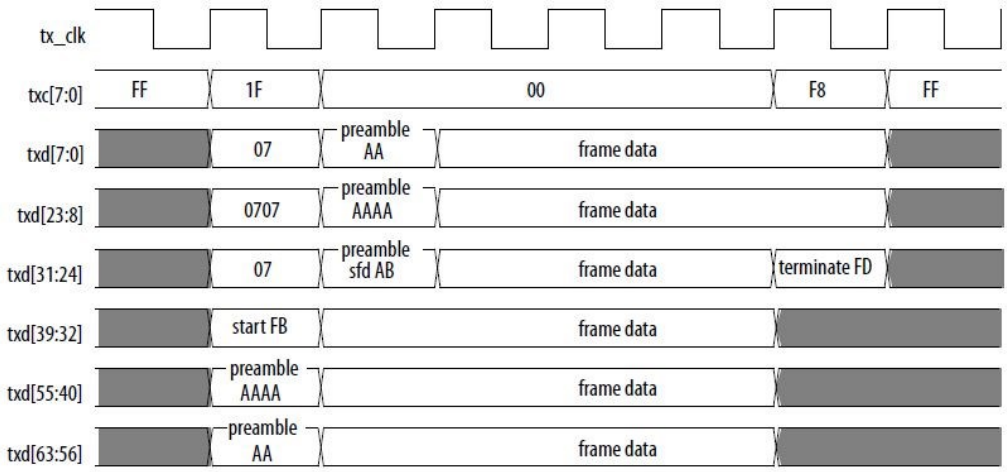

Figure 92: Byte 5 Start of Frame Transmission Example

Just after the MAC there is an asynchronous unit that concatenates the data and the control bits to form the transmitted message in the demanded way by XAUI specifications. A 72-bit long stream is being generated by placing each control bit to the left of its corresponding octet (figure 93). This longer stream is ready to be fed at the parallel data input of the PHY IP core.

7	63..56	6	55..48	5	47..40	4	39..32	3	31..24	2	23..16	1	15..8	0	7..0

Figure 93: Formed message

The 72-bit wide parallel data coming out of and the equally wide going in are separated to data and control. They are saved to 4 registers for 1 clock cycle till the next update input and output of the transceiver, making their observation in SignalTap easier.

Last part of our design is another asynchronous module for the DUAL XAUI daughterboard setup and initiation, according to the provided Test Designs [32] of its user manual.

5.3.4.1 IP Generation and Customization

Even in this case the procedure was very straightforward and simple to follow, IP generation is always important when the IP is integrated in a design. The first attempt was made with the Cyclone V GT FPGA, where the majority of the options where pre-defined and not allowed to change. As it is clear on the three snippets below there is nothing to edit or set on the general options tab. For the Cyclone FPGA everything is pre-set; PCS part will be implemented using soft logic and the used PLL type will be CMU, while the data rate logically does not give extra option than 3.125 Mbps.

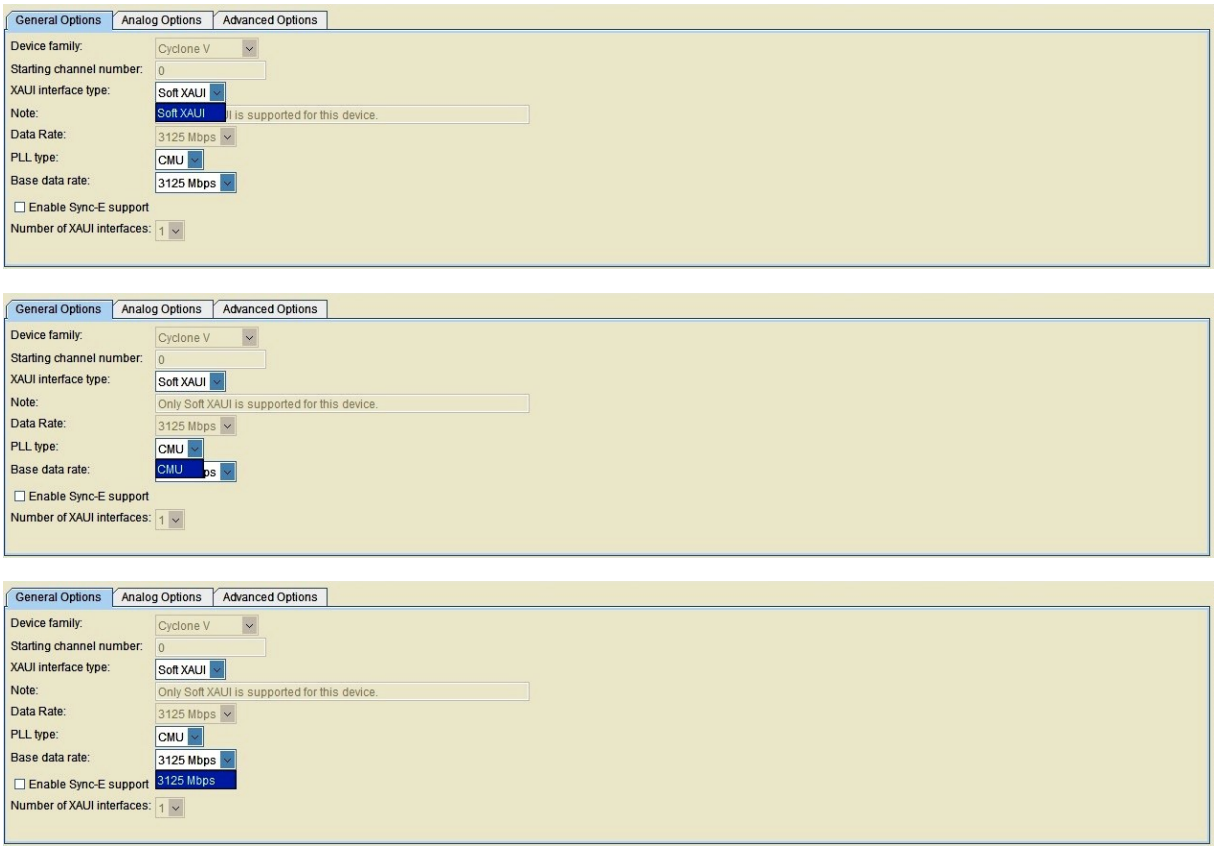

Figure 94: Cyclone V GT XAUI IP general options

The next Tab of Analog Options is totally blank for this device family as it is shown on figure 95.

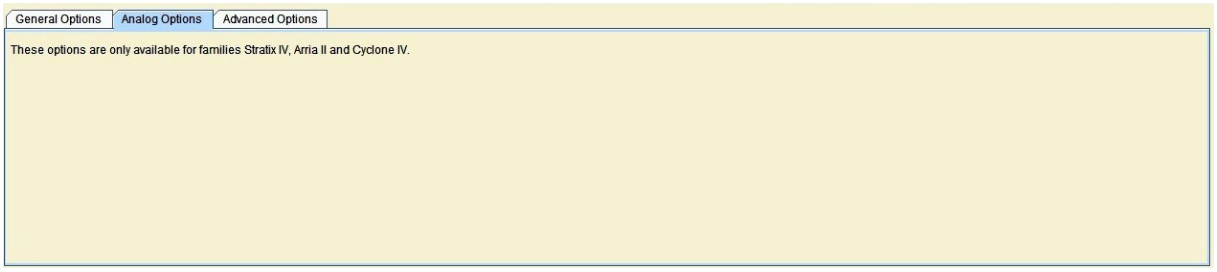

Figure 95: Cyclone V GT XAUI IP analog options

The last tab of Advanced Options enables or disables some control and status signals of the IP.

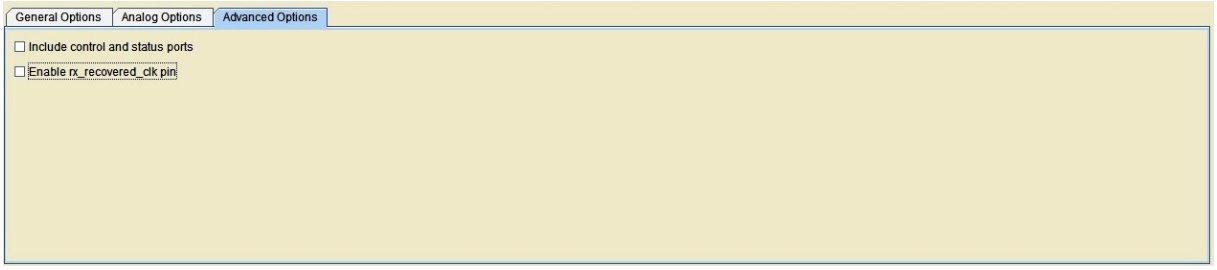

Figure 96: Cyclone V GT XAUI IP advanced options

All the snippets above indicate how straightforward but “restricted” the XAUI IP generation is in the Cyclone GT FPGA.

Following the same procedure for the DE4 board, which embeds a Stratix IV GX chip more options appear.

Figure 97: Stratix IV GX XAUI IP general options

Both drop down menus have more options offering not only hard PCS implementation, where resources are saved for user logic, but also a double speed DDR XAUI interface, where data rate reaches the value of 20Gbps. Another important difference is an ATX PLL option, which offers the suitable PLL solution for high speed serial links, providing low-jitter high-frequency clocks.

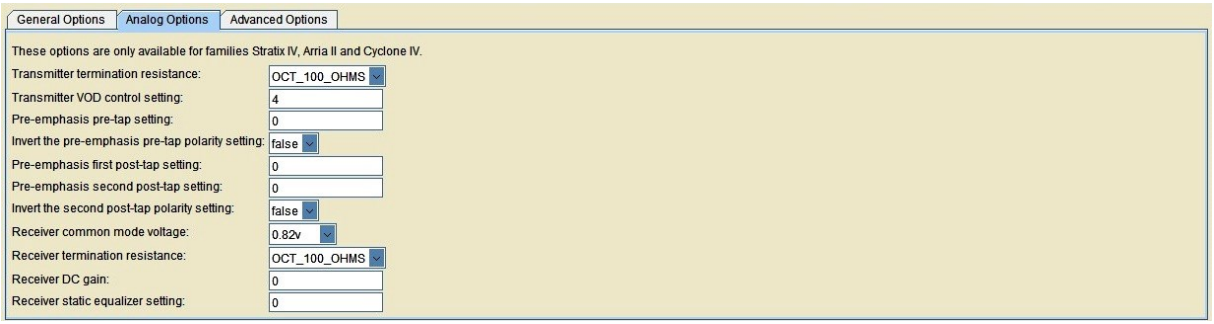

Figure 98: Stratix IV GX XAUI IP analog options

Figure 98 illustrates the default analog options of the IP, which were left as they are during the implementation. However, values like transceiver’s termination resistance, equalization DC gain level and pre-emphasis settings can be determined.

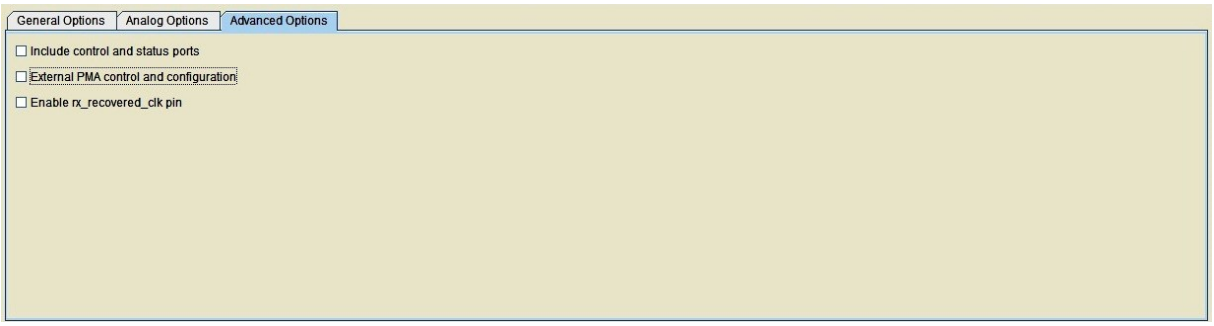

Figure 99: Stratix IV GX XAUI IP advanced options

The final tab, compared with the previous FPGA's advanced options tab offers access to the PMA configuration part of the IP, due to the fact that Stratix IV does not support direct reconfiguration of the XAUI by a specific IP as Cyclone V, but still allows it by exporting configuration inputs and control outputs.

After finishing the design of our system we proceeded by the pin planning discovering another disadvantage of Cyclone V GT device against DUAL XAUI HSMC daughterboard. The mounted board embeds a specific clean, low jitter clock oscillator as it was mentioned in its description [33] able to pass it through HSMC or SMA connectors. Unfortunately, using the native HSMC pins our design could not pass the place and root phase, due to error caused by clock-in pins location. In order to solve this SMA cables were used, providing to the system the required clock. Luckily, in DE4 we provided our system with 156.25 MHz clock directly imported by the HSMC port where the 10 Gigabit card was plugged in.

5.3.5 Results

Finally, SignalTap tool verified the accurate and correct functionality of our implementation in both devices.

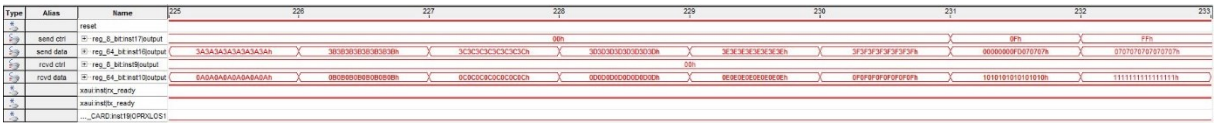

Figure 100: Transmission instance 1

The previous figure depicts transmitted and received incremented data according to our custom MAC. It is impossible to present in the same image the transmitted stream and its corresponding received 64-bit long information because as is highlighted in next figure there is a big "imagewise" latency in-between.

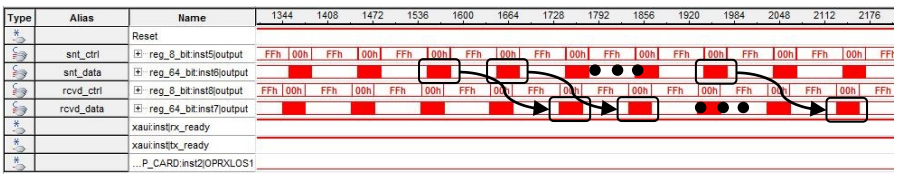

Figure 101: Transmission instance 2

In the final step of checking design's correct functionality, we broke the connection on loopback test by unplugging the optical fibers. Immediately the custom MAC's trigger signal went high indicating connection loss.

Figure 102: Unplugged fiber cable instance

After plugging the cable again, the synchronization process started, since the MAC initiated again its transmission loop, described previously. The transmitted stream containing consecutive instances of special character “0x07” restored the link. In the figure 103, a snippet of the restoration procedure is illustrated.

Figure 103: Synchronization process instance

5.4 GBT Loopback Test

5.4.1 GBT-FPGA IP

As it was described in the introduction about GBT [ch. 2.4], each device can implement a different GBT Bank instance. The point is that every GBT Bank will have at least one GBT Link, which is the common part of every design and it is being repeated. According to the design purpose and the FPGA resources, multiple instances of a GBT links can be organized in GBT Banks. So the GBT-FPGA IP includes a single transceiver link, which follows the appropriate architecture in order to fulfill GBT protocol specifications and requirements.

A simple block diagram of the GBT Tx part is presented in the figure 104 below.

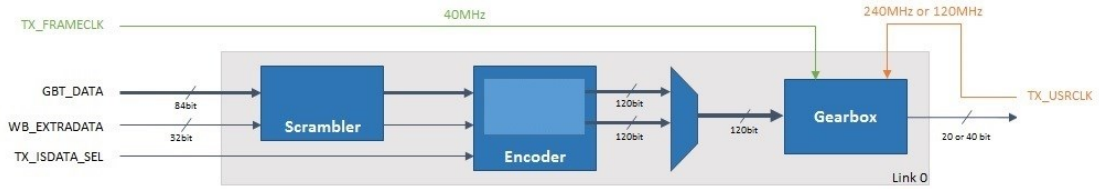

Figure 104: GBT Tx simplified block diagram

Analyzing the diagram, it is obvious enough that 120 bit enter the Scrambler module. In continuation, according to the encoding selection the data is being encoded and driven to the Gearbox before reaching the MGT part to get serialized.

The corresponding receiver is shown below.

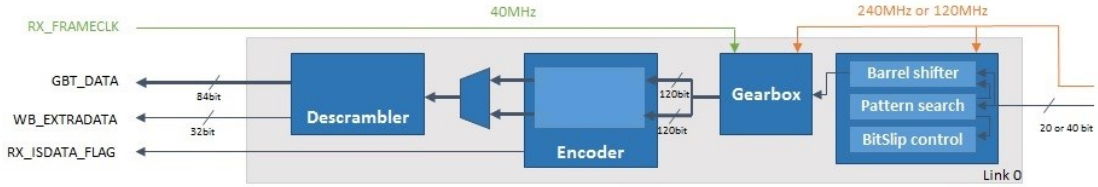

Figure 105: GBT Rx simplified block diagram

After the deserialization process, which also takes place at the MGT SerDes, the data is checked for an alignment pattern. Until the pattern is found, data are shifted accordingly. Then data follows the reverse process of the transmitter part. The Gearbox module is followed by the decoder, which not only decodes the data, but also outputs a signal informing about the used frame. Finally, the Descrambler part recovers the data to its initial value using the same polynomial as the Scrambler at the transmitter part.

Checking in more detail in the GBT FPGA IP block diagram, we can see clock instances and its values as well as all the timing related devices that are used in the design.

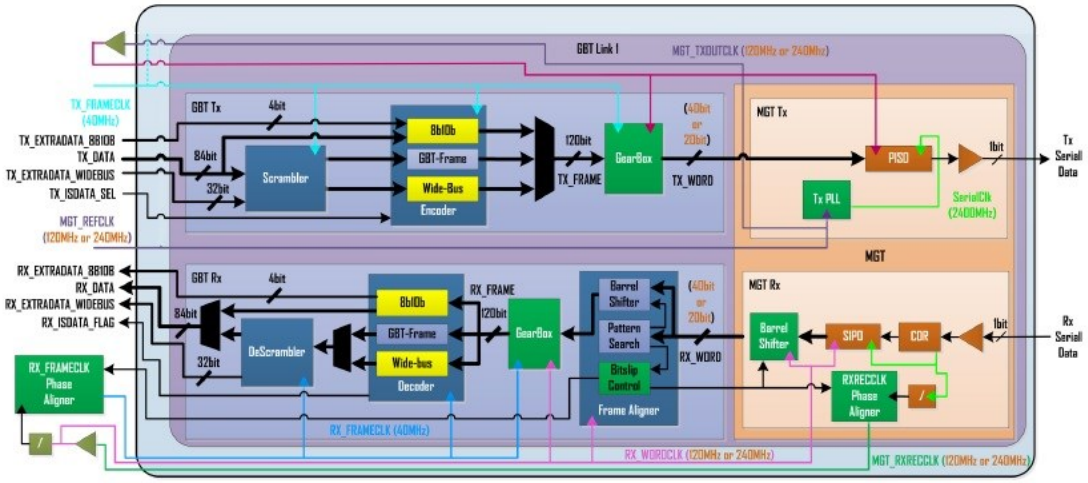

Figure 106: GBT Tx/Rx detailed block diagram

More specifically, data on the left side of the Gearbox unit either in receiver or in transmitter side are driven by a 40 MHz clock. While the "divided" information in the right side of the Gearbox, as expected, runs in 3 or 6 times higher frequency depending on the PMA interface (40 bit or 20 bit respectively). Of course, the “serial clock” used in SerDes part is much higher.

GBT-Frame encoding is able to convey timing and trigger data in a system, where timing constraints have to be accurate. A latency optimized version offers deterministic data and clock behavior becoming ideal for TTC systems. Yet, this version is quite challenging in implementing that is why mostly high-end FPGAs support it. There are some challenging parts that need special treatment.

- **Clock Domain Crossing**

The number of used clock domains has to be minimized as much as possible, because clock domain crossing is one of the main sources of latency uncertainty.

- **Non Deterministic Latency**

Even if the clock domains have been minimized, not all can be merged. This is a result of some components that require two clock domains, like the gearbox or the SerDes. The tricky part with those devices is to follow an asynchronous FIFO implementation method. Although this method may be simple, reliable and efficient, it lacks determinism and needs undesired clock cycles. The solution to this issue comes with a registered-based implementation of those structures. This can ensure low deterministic latency, yet it has a trade-off calibration after the system implementation in order to secure data integrity.

- **Clock Phase Relationship**

Different clocks generated by a single clock source do not always have fixed phase relationship, so as to achieve clock and data determinism. This is happening because the clock multiplication, division and recovery is not based on special components like a clock synthesizer, but it is based on common PLLs and firmware monitors and controllers for phase determinism. That is a differentiation between the devices which fully support GBT over the others which partially support it.

5.4.2 Implementation

The GBT-FPGA Core provided by CERN embeds different versions of the GBT-FPGA IP according to the supported devices. The remaining device-independent parts are located in a common folder. Cyclone V GT is one of the supported devices. In this case, when the corresponding project is open by the IDE, a tcl file needs to be run in order to attach all the design parts into the project. Cyclone V GT supports only a single GBT Bank, including a single link. Those parameters have to be set before compiling the design.

The loopback test is implemented using the SFP-HSMC daughterboard, which requires some extra input and output signals to be added in the design for controlling it and enabling its SFP corresponding cage. Additionally, the options and variations of GBT described in [ch. 2.4] come with specific signals in the Top level. Those signals need initiation or extraction in switches and buttons, providing the user with the option to change the version or encoding of the GBT Link.

The design also includes a data generator and a data checker, which are also customizable producing a static output or the concatenation of some counters. Finally, the required reference clock of 120 MHz was provided to the design by one of the device's programmable oscillators in order to achieve data rate of 4.8 Gb/s, as it is mentioned in the GBT specifications.

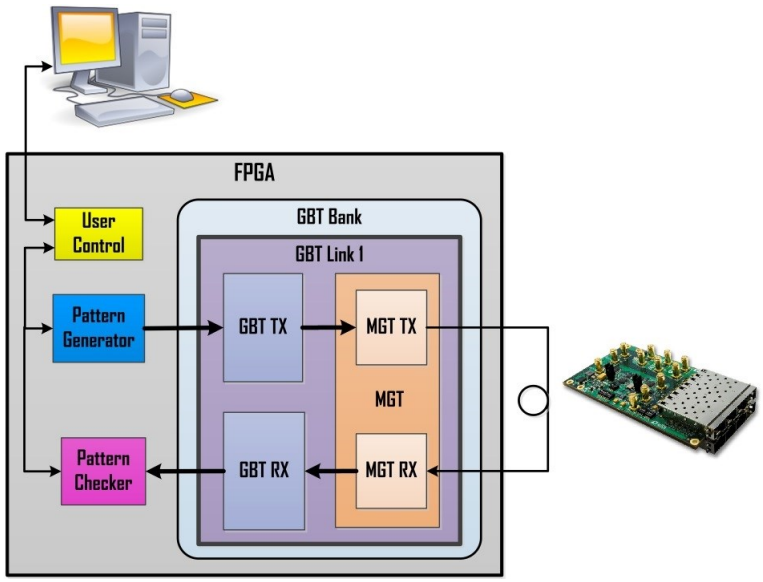

Figure 107: GBT loopback test block diagram

5.4.3 Results

In our loopback test we used the Standard version of the GBT-FPGA Core, since Cyclone V GT is discouraged for Latency Optimized mode because of the device's difficulty to achieve timing closure. For data generation we used the internal data generator, due to the option of using different counters' concatenation as output, which is really similar to our previous designs, where a single counter was used. The Wide-Bus fame is set as the used frame for our transmission. Finally, as in the prior loopback tests, SignalTap was selected as a verification tool.

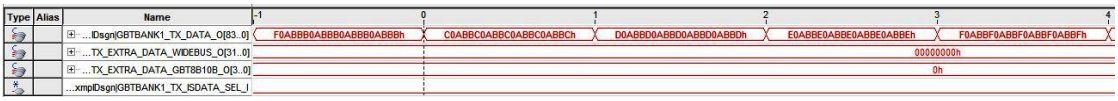

Figure 108: GBT transmitted data instance

Figure 108 above, we see the transmitted values. The LSB of the data increments by one in every instance. This is the desired behavior, which is also expected on the receiver side.

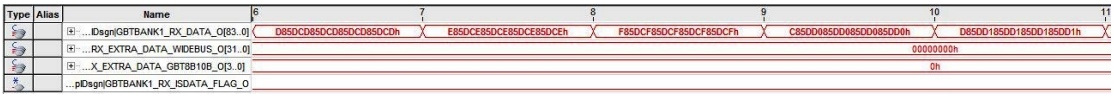

Figure 109: GBT received data instance

Observing the LSB of the received data, we can confirm the correct behavior and functionality of the implemented GBT-Link in standard encoding mode. The data generator can easily be replaced by a custom logic module driven at 40 MHz, which may drive as an example the experimental data of a DAQ system or may even convey simple instructions in order to control non-time critical devices in a SC system.

Chapter 6 Conclusion and Future work

6.1 Conclusion

A new low-end and affordable FPGA development board and some peripheral daughterboards were introduced to us for this thesis. All the extension-daughterboards were used to run tests for checking the FPGA's ability to host a "Controller" unit in a high data rate network. The most common and simple way to test the connectivity and performance of the devices' combinations is through loopback tests.

Summing up all the implementations' results, we could claim that a general loopback test was successfully implemented over different media, transfer rates and protocols. The first part of the Ethernet loopback test was an introduction for us into the gigabit serial links and the method of sending data through a specific format using a protocol. Additionally, it was the first time we got into our hands the Cyclone V GT FPGA board and the daughterboards. During the background reading, we found that it would be more convenient to build a system with a soft processor in order to succeed in our goal. So, we also had the chance to understand and apply a soft-processor-based design. This was recommended by the Aletra's IP, which we used in our design. IPs were also something new for us, on how to generate them, customize them and embed them to any design. Making a partial conclusion about the Ethernet part, we can comment that it was a difficult and time demanding part, because of the amount of information and obstacles we had to process and overcome respectively. However, loopback test with 1 Gbps data rate was successfully implemented.

In continuation, we dealt with fiber optics and high speed serial transceivers. It was something totally different and new for us because we had no previous experience or idea. Both of us we were not aware of how data is being serialized and deserialized. We also got to know the reason why data is being encoded properly before reaching the medium. Another part which still remains new and challenging for us is the datapath of high speed serial transceivers and the time dependencies between its components. For that reason, we needed the guidance of our supervisor and our reviewer to proceed with this step and even find the IP group, which fitted to our purpose. This part "design-wise" was quite simpler than the Ethernet part due to the absence of a soft processor or any complex structure. It consists of a counter as data generator and an IP, which plays the role of the PHY. Although, it seems simpler than the prior test it took the same amount of time to get the desired results, because of the IP's configuration. It had to be accurate in setting the timing and data aligning options. Fortunately, the outcome of our effort was positive for 16-bit and 32-bit raw data, which was looped back at 3.5 Gb/s and 5 Gb/s respectively.

After the second part's successful implementation we were boosted up with confidence, significant background knowledge and understanding at the transceiver datapath. In the third part we aimed to increase data transfer rate. To achieve our goal, we used the XAUI protocol

for 10 Gb/s, which made our design simpler. Applying the special XAUI IP and the standard counter as data generator, we managed to reach 10Gbps transfer rate in a loopback test. The duration of this part was quite smaller compared to each of the previous two. The existence of the XAUI IP, which worked as hardware driver to the XAUI daughterboard, combined with the better understanding of the aspect helped completing this thesis's main part on time.

By exploring the Altera's IP set for high speed serial transceivers, we experienced the inability of Cyclone V GT FPGA to support any IP customization related to clock and data determinism. Low-end devices, like Cyclone V GT FPGA, may face difficulties in achieving timing closure for latency-optimized version of GBT. Due to this fact we were provided with a Stratix IV DE4 development board. The latter board not only embeds more transceivers of a higher maximum data rate, but also uses different transceivers' timing components architecture. The existence of a different type of PLLs and their placement adjacent of the transceivers' blocks gives more options in the corresponding IPs' configurations. The DE4 board also hosted successfully the XAUI loopback test.

Having finished the main objective of our thesis to examine and evaluate the Cyclone V GT board over different transfer rates and protocols, we still had a little time to cope with an extra goal of this thesis. This goal was the implementation of the GBT protocol in the affordable, low-end Cyclone V GT board. As long as the GBT-Core was officially verified by the GBT researchers' team for our device, the only objective was to understand the datapath and its implementation in order to adapt it to our needs. Studying and analyzing the GBT project in order to eventually succeed in a loopback test at 4.8 Gb/s, we came to two different types of conclusions. Initially, the loopback test was successfully implemented. Conceiving the purpose, the frame reasoning and the protocol structure was a step of completion in the background knowledge we got in this thesis. In the perspective of GBT's datapath and the transceivers architecture of Cyclone V GT, the reason of the FPGA's inability to implement a low latency mode of the protocol is the lack of a high-level phase aligner in receiver's PMA part. So, the last thesis objective can be divided in three parts. They include background reading about the GBT interface, implementation and conclusions verification by cross comparison of already read information.

The last two implementations, about XAUI protocol and GBT protocol, were the most clear to implement and test. The XAUI is considered as an extension of its previous part about SFP loopback. The SFP loopback took us enough time to realize its concept and implement it, but it gave us knowledge to accelerate XAUI's part. Additionally, GBT-FPGA Design describes a custom interface, which uses common components of the transceiver datapath. The use of common modules and our previous experience made the understanding part easier and simpler. As far as the implementation part of GBT-FPGA Core is concerned, we had only to edit/add some parameters in order to fit the core in our system, since the main design part is already implemented by a CERN group of developers.

6.2 Future work

Taking into consideration the need of a TFC unit in a local laboratory experiment, equipped with a DAQ system and/or a SC system, we can propose some future extensions of this thesis.

- **Protocol establishment**

A custom protocol may be created, studied and evaluated. A custom data frame with specific part for error detection or error correction, would be quite useful.

- **DE4 board tests completion**

Unfortunately, the Stratix IV board was not extensively tested and evaluated. All the options of already used IPs can be tested using DE4 as a first step. Furthermore, different serial high speed IPs that support different protocols and mainly time-dependent protocols can be used to implement corresponding loopback tests. This will help significantly to assess the DE4 chip's transceivers determinism and latency.

- **Custom PCB design**

As long as the TFC will be applied in a custom experiment, a custom board can be designed to further minimize the cost. After finding an appropriate FPGA chip for the experiments specifications, then it can be implemented in a custom board, which will embed only the required components, peripheral I/O pins and connectors. This might allow developers to design a mid-end custom board at the same price as a low-end development board, since it will not include unnecessary components and parts.

- **Higher data rate GBT imitation**

Choosing the Cyclone V GT FPGA as the main chip of a possible TFC and GBT as the desired protocol limits our options in implementing the Standard version of GBT. In this case, the data rate will be 4.8 Gbps and the latency of the link will be non-deterministic. The data and clock integrity have been already ensured in the Cyclone V GT board for high speed links up to 10 Gb/s by the XAUI loopback test. So, if the TFC is going to be embedded in a timing/latency non-critical system, then a protocol which merges and frames the data like the GBT-Frame Standard version can be established in higher transfer rates than 4.8 Gbps.

Split of the work

- **Ethernet part:**
  - Theory: Vasileios Filos
  - Implementation: Jiheng Chen
  - Test / Debug: Vasileios Filos
- **SFP part:**
  - Theory: Jiheng Chen
  - Implementation: Vasileios Filos
  - Test / Debug: Jiheng Chen
- **XAUI: part**
  - Theory: Vasileios Filos
  - Implementation: Jiheng Chen
  - Test / Debug: Vasileios Filos
- **XAUI: part**
  - Theory: Jiheng Chen
  - Implementation: Vasileios Filos
  - Test / Debug: Jiheng Chen

References

[1] R.D. Mauer, *Glass fibers for optical communications*, Proceedings of the IEEE, vol. 61, no. 4, pp. 452-462, April 1973

[2] Altera Corporation, *Overcome Copper Limits with Optical Interfaces*, Accessed April 20, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01161-optical-fpga.pdf

[3] V. Alwayn, *Optical Network Design and Implementation*, San Jose: Cisco Press, 2004, Accessed April 23, 2016, http://ptgmedia.pearsoncmg.com/images/1587051052/samplechapter/1587051052content.pdf

[4] J. M. Senior, M. Y. Jamro, *Optical fiber communications: principles and practice*, Pearson Education, 2009. Accessed May 21, 2016

[5] S. Minami, J. Hoffmann, N. Kurz, W. Ott, *Design and Implementation of a Data Transfer Protocol Via Optical Fiber*, IEEE Transactions on Nuclear Science, vol. 58, no. 1, pp. 1816-1819, 2011

[6] Altera Corporation, *Altera Shows World's First Optical FPGA Technology Demonstration*, Accessed April 22, 2016, https://www.altera.com/about/news_room/releases/_2012/products/nr-optical-fpga-demo.html

[7] Terasic Technologies Incorporated, *Cyclone V GT Development Kit Resources*, Accessed April 16, 2016, http://www.terasic.com.tw/cgi-bin/page/member.pl?Language=English&Return=http%3A%2F%2Fwww.terasic.com.tw%2Fcgi-bin%2Fpage%2Farchive.pl%3FLanguage%3DEnglish%26CategoryNo%3D167%26No%3D843%26PartNo%3D3

[8] Wikipedia, The Free Encyclopedia, s.v. *Media-independent interface*, Accessed March 3, 2016, https://en.wikipedia.org/wiki/Media-independent_interface

[9] Terasic Technologies Incorporated, *Altera DE4 Development and Education Board Resources*, Accessed May 10, 2016, http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=138&No=501&PartNo=4

[10] Altera Corporation, *High Speed Mezzanine Card - Specification*, Accessed April 16, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ds/hsmc_spec.pdf

[11] Terasic Technologies Incorporated, *Terasic HSMC-NET Daughter Board - User Manual*, Accessed March 3, 2016, http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=355&FID=886a996265218e6f26e23bfd4013bd3c

[12] Wikipedia, The Free Encyclopedia, s.v. *SerDes*, Accessed May 23, 2016, https://en.wikipedia.org/wiki/SerDes

[13] Wikipedia, The Free Encyclopedia, s.v. *I^2C*, Accessed April 16, 2016, https://en.wikipedia.org/wiki/I%C2%B2C

[14] Wikipedia, The Free Encyclopedia, s.v. *Small form-factor pluggable transceiver*, Accessed April 16, 2016, https://en.wikipedia.org/wiki/Small_form-factor_pluggable_transceiver

[15] Altera Corporation, *Quartus Prime Standard Edition Handbook Volume 3: Verification*, Accessed April 20, 2016, https://www.altera.com/en_US/pdfs/literature/hb/qts/qts-qps-5v3.pdf

[16] Altera Corporation, *Nios II Hardware Development Tutorial*, Accessed March 5, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/tt/tt_nios2_hardware_tutorial.pdf

[17] A.X. Widmer, P.A. Franaszek, *A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code*, IBM Journal of Research and Development, vol. 27, no. 5, pp. 440-451, 1983

[18] Cortina Systems Inc., Cisco Systems Inc., *Interlaken Protocol Definition, rev 1.2,* October 7, 2008, Accessed May 26, 2016, http://www.interlakenalliance.com/Interlaken_Protocol_Definition_v1.2.pdf

[19] Altera Corporation, *Implementing CRCCs in Altera Devices*, Accessed May 28, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an049_01.pdf

[20] Search Networking, *Cyclic redundancy checking*, Accessed May 26, 2016, http://searchnetworking.techtarget.com/definition/cyclic-redundancy-checking

[21] GHS Infotronic, *Online CRC Calculation*, Accessed May 26, 2016, https://www.ghsi.de/CRC/index.php?Polynom=11000000000000101&Message=E100CAFE

[22] S. Henry, S. Warren Jr, *Hacker's Delight, 2nd Edition*, ch.14, pp 319-330, Accessed May 28, 2016, http://www.hackersdelight.org/crc.pdf

[23] Altera Corporation, *Triple-Speed Ethernet MegaCore Function - User Guide*, Accessed March 3, 2016 https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_ethernet.pdf

[24] Altera Corporation, *Avalon Interface Specifications*, Accessed March 3, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec.pdf

[25] Altera Corporation, *Using Triple-Speed Ethernet on DE2-115 Boards*, Accessed March 3, 2016, ftp://ftp.altera.com/up/pub/Altera_Material/13.1/Tutorials/DE2-115/using_triple_speed_ethernet.pdf

[26] Finisar Corporation, *2.125 Gb/s RoHS Compliant Short-Wavelength SFP Transceiver*, Accessed April 20, 2016, http://pdf1.alldatasheet.com/datasheet-pdf/view/277496/FINISAR/FTLF8519P2BTL.html

[27] Avago Technologies, *AFBR-709DMZ 10Gb/1Gb Ethernet, 850nm SFP+ Transceiver - Data Sheet*, Accessed April 16, 2016, http://docs.avagotech.com/docs/AV02-3828EN

[28] Altera Corporation, *Altera Transceiver PHY IP Core User Guide*, Accessed April 20, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/xcvr_user_guide.pdf

[29] Altera Corporation, *Cyclone V Device Handbook Volume 2: Transceivers*, Accessed April 20, 2016, https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-v/cv_5v3.pdf

[30] Altera Corporation, *Transceiver Toolkit Examples for Stratix® V GX, Arria V GX/GT, Cyclone V GX/GT and Stratix IV GX/GT Devices*, Accessed April 16, 2016, https://www.altera.com/support/support-resources/design-examples/design-software/on-chip-debugging.html

[31] Terasic Technologies Incorporated, *HSMC Debug & Loopback Connector Package*, Accessed April 16, 2016, http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=78&No=495

[32] Altera Wiki, *10-Gbps Ethernet MAC and XAUI PHY Interoperability Hardware Demonstration Reference Design*, Accessed May 5, 2016, http://alterawiki.com/uploads/e/e2/AN638_10GMAC_XAUI_HSMC_SIV_GX_ACDS-12.0sp2.qar

[33] Terasic Technologies Incorporated, *DUAL-XAUI Board - User Manual*, Accessed May 2, 2016, http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=597&FID=ef795de1ac586e31317766f74e7ffedd

[34] J. Mitra, S.A. Khan, M. B. Marin, J.-P. Cachemiche, E. David, F. Hachon, F. Rethore, T. Kiss, S. Baron, A. Kluge, T.K. Nayak, *GBT link testing and performance measurement on PCIe40 and AMC40 custom design FPGA boards*, Lisbon, Portugal, 2016, Accessed June 8, 2016, http://iopscience.iop.org/article/10.1088/1748-0221/11/03/C03039/pdf

[35] Wikipedia, The Free Encyclopedia, s.v. *Ethernet frame*, Accessed May 29, 2016, https://en.wikipedia.org/wiki/Ethernet_frame

[36] CERN org., *The GBT Project*, Accessed May 29, 2016, https://espace.cern.ch/GBT-Project/default.aspx

[37] Wikipedia, The Free Encyclopedia, s.v. *Reed–Solomon error correction*, Accessed June 2, 2016, https://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction