The Pennsylvania State University

The Graduate School

College of Engineering

A MODIFIED DUOBINARY COMMUNICATION SYSTEM FOR RAPID PROTOTYPING USING FPGAs AND ASICs

A Thesis in

Electrical Engineering

by

Ashraf Ibrahim Umar

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

May 2014

The thesis of Ashraf Ibrahim Umar was reviewed and approved* by the following:

Aldo W. Morales

Professor of Electrical Engineering

Thesis Co-Adviser

Sedig S. Agili

Associate Professor, Department Head of Electrical Engineering

Program Coordinator, Master of Science in Electrical Engineering

Thesis Co-Adviser

Jeremy Blum

Associate Professor of Computer Science

*Signatures are on file in the Graduate School

ABSTRACT

High speed communication channels, including backplanes, always have distorting effects on signals being transmitted through them. This is mainly a result of the frequency dependent nature of such channels. In order to address this issue two common techniques exist: either carefully selecting the materials used in the backplane design or modifying the signal to suit characteristics of the communication channel/backplane by employing different line coding schemes and equalization. The most common line coding method is NRZ; however, as speed further increases, duobinary and PAM-4 are also promising techniques being investigated.

Most of the past research in duobinary and PAM-4 was concentrated on simulations of the performance of coding and equalization techniques to compensate for the channel distortion. This proposed work focuses on rapid prototyping, using FPGAs and/or ASIC, of NRZ and duobinary coding and channel equalization. NRZ and duobinary coding are chosen because they are generally less complex than PAM-4, which makes them a good choice for higher data rates.

A typical duobinary transceiver system comprises of an encoder at the transmitter and the corresponding decoder at the receiver. The complete encoderconsists of a duobinary pre-coder, which in turn includes a unit delay and an XOR gate to prevent error propagation, and a delay and add filter that converts the two level NRZ signal into a three level duobinary signal. The duobinary signal is then transmitted to the communication channel. At the receiver side, the duobinary decoder is implemented using a signal splitter, two comparators, and an XNOR gate. The duobinary signal generated is a three level signal which current commercial FPGAs are not capable of handling. In order to solve this problem, a simple new architecture of a duobinary system to be used with the commercial, off-the-shelf FPGAs is proposed.

The standard duobinary system architecture is modified by placing the duobinary encoder after the transmit equalization, before the channel, while the duobinary decoder is placed immediately after the channel, before receiver equalization is performed. This scheme offers the advantage of allowing us to use the FPGA equalizers in the NRZ coding without having to modify them to support the three-level duobinary signal. Hence, this modified architecture takes advantage of the well-developed digital signal processing blocks in commercial FPGAs while allowing faster development times. The duobinary encoder and decoder can be built in an ASIC and interfaced

with the FPGA. Simulation of this architecture is performed in Simulink, and results obtained show that hardware implementation of such architecture is feasible as the transmitted data is reliably recovered at the receiver.

To accomplish this research, two software tools were used, MATLAB Simulink and Altera's Quartus. The FPGA board used was a Stratix IV GT SI Development board with the EP4SG210040I1 chip. Simulink was used for the NRZ simulation and Quartus for hardware implementation. Two real-world channels were used: a 29 in Megtron-6 Caltrace board and a 32 inch backplane both provided by FCI electronics. Eye diagram scopes in Simulink are used to view the simulation results. The transceivers of the Stratix IV GT SI board were run at 5.65 Gbps and 11.3 Gbps using both channels to verify proper operation and also to demonstrate the equalization features within the transceivers. NRZ measurements were taken with the DSA 8200 Tektronix Time Domain Reflectometer. A correlation between the simulated and measured NRZ data is made and the results show a high degree of correlation. The BER for NRZ and duobinary were also computed for both channels the results were comparable; however, the duobinary uses half of the bandwidth.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADS	Advanced Design System
ASIC	Application Specific Integrated Circuit
BER	Bit Error Rate
CAD	Computer Aided Design
CDR	Clock and Data Recovery
CLB	Configurable Logic Block
CMU	Clock Multiplier Unit
CPLD	Complex Programmable Logic Device
DC	Direct Current
DFE	Decision Feedback Equalizer
DUT	Device Under Test
FFT	Fast Fourier Transform
FIFO	First Input First Output
FIR	Finite Impulse Response
FBGA	Fine-Line Ball Grid Array
FPGA	Field Programmable Gate Array

FEXT	Far-End Crosstalk
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronic Engineers
IC	Integrated Circuit
ISI	Inter-Symbol Interference
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LMS	Least Mean Square
NEXT	Near-End Crosstalk
NRZ	Non-Return to Zero
PAL	Programmable Array Logic
PAM	Pulse Amplitude Modulation
PLA	Programmable Logic Array
PRBS	Pseudo-Random Bit Sequence
PROM	Programmable Read Only Memory
SMA	SubMiniature version A
SOF	Static-RAM Object File

UCF	Universal Constraint File
USB	Universal Serial Bus
VHDL	Very High Speed Integrated Circuit Hardware Description Language

ACKNOWLEDGEMENTS

I extend sincere appreciation to my parents and siblings for the support received both morally and otherwise.

My gratitude also goes to my supervisors Drs. Aldo Morales and Sedig Agili for introducing me into the exciting field of Signal Integrity and all the support they rendered during the course of this project. I also want to thank them for letting me work at the Center for Signal Integrity Lab to improve on my practical knowledge. I also want to thank Dr. Jeremy Blum for accepting to serve as a member of my thesis committee.

Special thanks to the team at FCI for offering me an internship position and also providing me with the FPGA board and channels used in this thesis.

My profound gratitude goes to faculty, staff and colleagues that made my stay here worthwhile.

Chapter 1

Introduction

1.1 Research Motivation

The need for higher data capacity among electronic devices today implies the requirement for faster transmission rates between them. There is so much data today that needs to be transmitted from one point to another, be it from servers or multi service switches to hosts or even among portable devices in our homes and offices. These include data from computers to printers, fax machines, and scanners. This puts a great burden on current backplanes and connectors to support the required bandwidth necessary for such high speed data transmissions. Consequently, different techniques are investigated for improving the integrity of the signal being transmitted to allow for faster transmission speeds. Typically, two approaches exist: the passive and active [1]. The passive approach involves the use of high quality microwave substrate materials and new connector technologies. In other words, it is concerned with improving the properties of the channel itself by investing in new board designs and improved via hole technologies to mention a few. This approach, however, tends to be expensive as much research effort and time is needed in the study and selection of such materials. Furthermore, replacing the backplane will result in a lot of downtime [1, 2, 3]. The active approach, on the other hand, is concerned with the signal transmitted through the channel. It involves a form of processing on the signal to give it certain properties which aid in overcoming the poor response of the channel. The latter method is arguably easier, faster and cheaper to

implement than the former [1]. When working on implementing new and improved systems, it is always recommended to utilize fast and easy methods. This will help to make design changes that will arise much quicker and hence helps reduce the time to market the final device. Current Computer Aided Design (CAD) tools make design and prototyping systems much quicker and less cumbersome. It is also possible to use more than one CAD tool in a design. This thesis aims to explore this property by using Simulink and Quartus to prototype a duobinary transceiver system for high speed backplane applications and also propose architecture for integration with Field Programmable Gate Array (FPGA) transceivers with an ASIC.

1.2 The Need for Simulation in Simulink

Work on duobinary simulation has been carried out in [1, 2, 4, 5, 6]. In [2], a duobinary transceiver system for multi-gigabit backplane applications for data rates from 10 to 40 Gbps was simulated using the Advanced Design Software (ADS) software. However, it had no direct link to hardware. As this thesis is geared more towards hardware implementation, Simulink is chosen as the tool for simulation because Hardware Description Language (HDL) code for programming the hardware, the FPGA in this case, can easily be generated and exported to Quartus from where it can be used to directly program the board or instantiated as part of a larger design. This reduces the time required to learn HDL coding and hence allows us to focus more on the implementation aspect of the design.

1.3 The role of the FPGA

Duobinary signaling was first proposed by Lender [3]. A method to encode a two level binary data into a three level duobinary signal was proposed. The encoder was simply an AND gate logic and a flip flop. The AND gate complements the input to the flip flop which then sends the data out on the channel. At the receiving end, a rectifier and slicer are used to recover the transmitted binary signals from the duobinary signal. The duobinary system as mentioned had little difference with the NRZ in terms of components required for coding and encoding of data. The spectral characteristics of both straight binary and duobinary techniques were also compared, and it was found that the duobinary system had half the bandwidth of the straight binary, and hence, data could be transmitted -at least theoretically- at twice the data rate of the straight binary. The duobinary system is mainly used in optical communications because of the wide bandwidths involved.

With multigigabit transmissions now possible in the electrical domain and the low bandwidth of backplanes, simple line coding methods are needed. As a result of its simplicity and low bandwidth requirements, duobinary signaling is making a comeback in the field of high speed electrical backplane applications. Studies are being carried out in the field and various ways of implementing the encoder and decoder are springing up.

In [4], Sinsky et. al. ran a simulation of high speed electrical backplane transmission using duobinary signaling. A pseudo random generator of 10 Gbps was used to generate the test signal input. The duobinary encoder was implemented as a simple two tap Finite

Impulse Response (FIR) pre-emphasis filter cascaded with the channel to generate the required duobinary signal. The decoder was implemented using a splitter which splits the incoming duobinary signal and feeds it to two comparators whose outputs are then fed to an XOR gate for final decoding. Tyco Quadroute traces with lengths of 6, 20, and 34 inches were used for the experiments. Eye diagrams for these trace lengths were viewed and the Bit Error Rate (BER) also measured. BERs of less than 10^{-13} were achieved. The eye diagrams and BER comparisons for duobinary were also made with those of Non-Return to Zero NRZ and Pulse Amplitude Modulation PAM-4.

A. Adamiecki et. al. in [5] demonstrate for the first time 25 Gbps electrical duobinary transmission over FR-4 backplanes. A 25 Gbps Non-Return to Zero (NRZ) Pseudo-Random Bit Sequence (PRBS) data source is used, and an FIR filter implements the duobinary encoding while a BERT is used at the receiver to study the BERs for 14 and 24 inch channel lengths. The eye diagrams at the input and output of the backplane were also viewed.

Yamaguchi et. al. [6] also studied duobinary signaling at speeds of 12 Gbps with two times oversampled edge equalization. The work compared transfer functions of the duobinary, PAM-2 and PAM-4 signaling.

In [7], a 10 Gbps duobinary signaling was implemented. Backplanes used for the experiment were the 20 and 34 inch Quadroute and XAUI. A two tap FIR filter was used for pre-emphasis, and PRBS 23 and 31 patterns were used. In both cases, bit error rates of less than 10^{-13} were observed with zero errors in a 20 minute measurement period.

An Altera white paper, [8] states the possibility of using 28 nm FPGAs for backplane applications. Such FPGAs have built-in transmit and receive equalization capabilities. The white paper begins by stating the common mechanisms for signal loss in backplanes and then going on to discuss in detail the various transmit and receive equalizers present in the 28 nm FPGAs. An example application also shows an eye diagram with and without equalization. A similar white paper, [9] discusses the various serial transceiver protocols that could be implemented in FPGAs. The 28 nm transceiver architecture is reviewed; the various clocking methods for the transceivers are also discussed. An important parameter, the power efficiency of the transceivers is compared at various data rates. The authors also discuss the equalization schemes that could be implemented with the FPGAs. The jitter and BER were also discussed. The information discussed in [8] and [9] present FPGAs as suitable devices for prototyping and subsequently implementing duobinary transceivers for high speed backplane applications.

All the research mentioned above did not use an FPGA. This research aims to make rapid prototyping of a duobinary transceiver using FPGAs and multi-vendor CAD tools like Simulink from MATLAB and Quartus from Altera. The idea is to reduce writing codes as much as possible as this will make the design and troubleshooting as well as time to market devices quicker. The duobinary scheme is expected to be incorporated into the FPGA transceiver architecture as the FPGA has built-in transceivers with pattern generators and checkers as well as equalization capabilities. These make setting up and studying communication links easier and quicker. Also, as stated in Section 1.2, Hardware Description Language (HDL) code for programming the FPGA can be

generated directly from Simulink. This makes verification of the design in hardware faster. Interestingly, duobinary implementation with FPGAs has not been attempted before; previous work concentrates on equalization with the FPGA while this work aims to incorporate line coding using FPGAs. Duobinary is chosen because it has half the bandwidth of NRZ line coding and hence offers a potential for doubling the present data rates with less channel equalization requirements. Throughout this research, two real-world channels were used: a 29 in Megtron-6 Caltrace board and a 32 inch backplane both provided by FCI electronics. Eye diagram scopes in Simulink are used to view the simulation results. The transceivers of the Stratix IV GT SI board were run at 5.65 Gbps and 11.3 Gbps using both channels to verify proper operation and also to demonstrate the equalization features within the transceivers. Measurements were taken with the DSA 8200 Tektronix Time Domain Reflectometer. A correlation between the simulated and measured NRZ data is made and the results show a high degree of correlation. The BER was also computed for both channels for both channels and the results were similar.

1.4 Project Outline

This thesis comprises five chapters. Chapter 1 contains a general introduction about the work and the tools used. Chapter 2 has the theoretical background about FPGAs, issues in high speed signal transmission and duobinary signaling. Chapter 3 deals with the NRZ simulation and demonstration of the operation of the transceivers, with real world channels, in the FPGA. Chapter 4 discusses a duobinary architecture using ASICs for integration with the FPGA transceivers and the simulation of such architecture while

Chapter 5 concludes the work by stating the results obtained, limitations encountered in the course of the work, as well as the recommendations for further study.

Chapter 2

Theoretical Background

2.1 Field Programmable Gate Array

Since the invention of transistors for amplification and switching, a lot has been going on in the field of electronics. Notable is the issue of miniaturization. Transistors used to be bulky. However, with its abundance and the improvement in technology, silicon has been utilized to produce Integrated Circuits (ICs) which have significantly scaled down the size of present day electronic devices. In addition, the development of CAD tools has allowed for easier and faster ways to implement complex designs comprising many transistors.

Designs implemented in the ICs were normally Application Specific, where the operation of the system is defined during manufacture. Hence, they are called Application Specific Integrated Circuits (ASICs). However, this architecture does not allow for flexibility as the design is hardwired from the factory. There is also the issue of long time to market and increased cost of production as any mistakes made during production normally renders that piece of hardware useless.

Programmable Logic Devices (PLDs) were introduced as a means of achieving flexibility so that users have the ability to program the device after manufacture. One of such PLDs is the Programmable Logic Array (PLA). This consists of a matrix of programmable AND planes followed by a matrix of programmable OR planes. Basic logic functions are

then implemented by connecting the gates together usually after reducing the design to the Sum of Products form. Another similar PLD architecture called the Programmable Array Logic (PAL) also exists which consists of a matrix of programmable AND plane followed by a fixed OR plane. Designs are also implemented by programming the AND planes which allow connections between the AND gates and the OR gates. The fixed architecture of the OR planes in the PAL allows them to run faster than the PLAs because the fixed connections switch faster than the programmable connections. The PLAs on the other hand are more flexible than the PALs because of the ability to program both the AND and OR planes [10]. Figure 2-1 shows a PAL and PLA architecture.

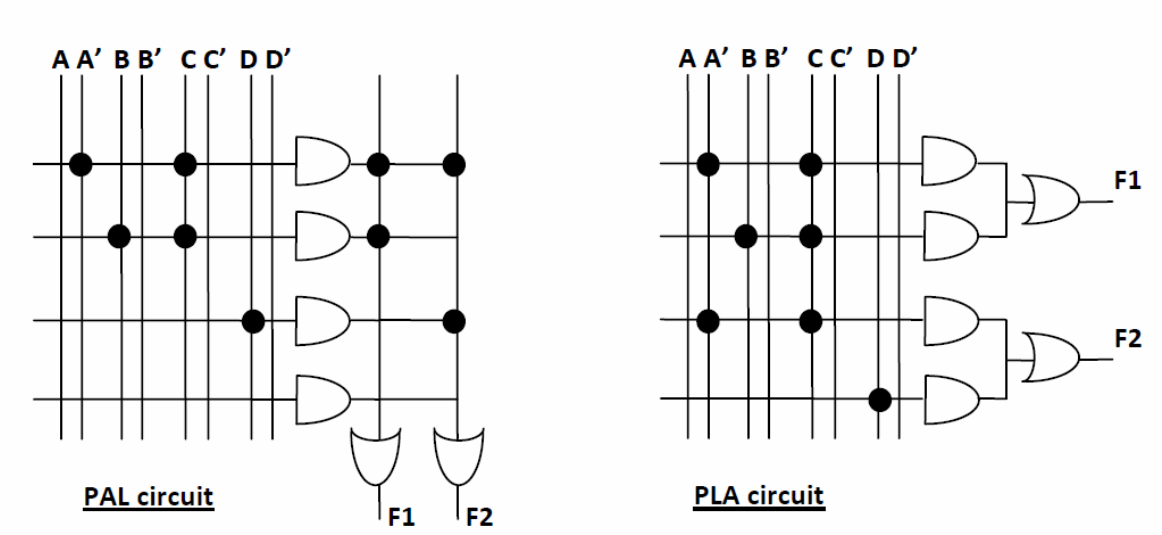


Figure 2-1. PAL and PLA architecture [10].

Complex Programmable Logic Devices (CPLDs) have an architecture based on the PAL. They consist of multiple PAL logic blocks that are connected with a programmable switch matrix. CPLDs can be reprogrammed several times and can include storage devices and feedback lines.

A Field Programmable Array (FPGA) is simply a Programmable Logic Device, (PLD) that comprises a collection of Configurable Logic Blocks, (CLB) which could be logic gates, memory devices or almost any other element arranged in an array with interspersed switches [11]. This is different from the architecture of the devices discussed earlier which comprised of a combination of AND and OR gates. From the architecture of the FPGA, at least two things should be obvious. One is that of flexibility. Since the FPGA is programmed to connect various switches, it means that it can be used to achieve almost any design by rearranging the connection between the switches. The second is the fact that the logic blocks can be programmed independently of each other, and this implies that multiple designs can be implemented at the same time on a single FPGA to achieve parallel computing. The FPGA is usually programmed by connecting the CLBs together with the aid of the switches between them to achieve virtually any desired function. The CLBs are seen in white with the switches that enable connections between these blocks in grey. Note that the I/O blocks are also incorporated into the FPGA that allow communication with the outside world. Figure 2-1 shows the schematic of a typical FPGA.

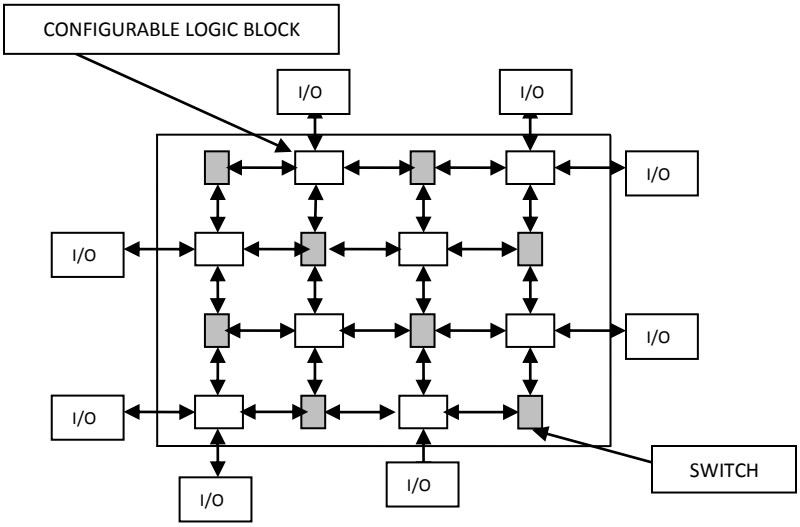


Figure 2-2. Schematic of an FPGA [10].

The CLB in an FPGA normally consists of a Look Up Table (LUT), a memory element which is usually a flip flop and a multiplexer that selects which of the registered or unregistered outputs of the LUT are to be used as the output of the CLB. Figure 2-3 shows the structure of a CLB in an FPGA.

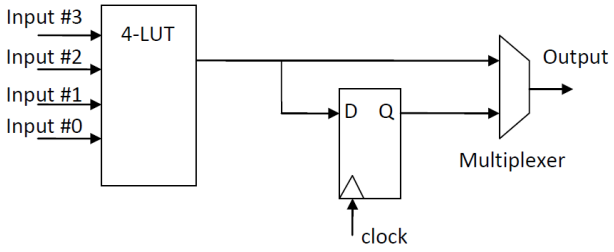


Figure 2-3. Structure of a CLB in an FPGA [11].

FPGA Programming is normally done using HDLs of which the most common are the Very High Speed Integrated Circuit Hardware Description Language (VHDL) and Verilog. VHDL was initially developed by the United States Department of Defense for the modeling and simulation of electronic devices. In 1987, however, it became an IEEE standard 1076 and found its way out of the military circles. Other standards have since followed. They include the 1076 of 1993 and the 1076 of 2001 [11].

Verilog on the other hand was developed by Philip Moorby of Gateway Design Automation. In 1989, Cadence Design Systems acquired Gateway and put Verilog in the public domain and it eventually became an IEEE standard in 1995 with another standard following in 2001 [11]. VHDL and Verilog can both be used for system design, simulation and synthesis and are also supported by most EDA vendors. VHDL, however, supports system level modeling and has a more comprehensive simulation than Verilog. In terms of use, Verilog is normally easier to learn as it is weakly typed as compared to VHDL. A key advantage of VHDL over Verilog, however, is the concept of packages. These are simply procedures and functions that can be used by any design unit. This does not exist in Verilog. Verilog is similar to C while VHDL has its roots in Ada. Most times, it is a matter of personal convenience that determines which of the languages to use. Verilog is used in this thesis because component instantiation in a hierarchical design is easier to implement than it is in VHDL.

2.2 Quartus II

A very important component of this work is the Quartus II software. It is within this program the transceiver is defined, compilation is done and device programming is achieved. Quartus II is a complete Integrated Development Environment IDE from Altera that allows for system design and testing. Design entry in VHDL, Verilog and schematic can be done, compilation of generated code, analysis and synthesis, placing and routing, programming file generation as well as device programming can all be achieved with the Quartus II. Simulation is normally performed with Modelsim. The Quartus II also has other tools like the Signal Tap logic analyzer for probing FPGA pins, the transceiver toolkit for transceiver testing, and the Qsys for defining and implementing systems that include processors among others. A 64 bit Quartus II Version 12.0 environment is shown in Figure 2-4.

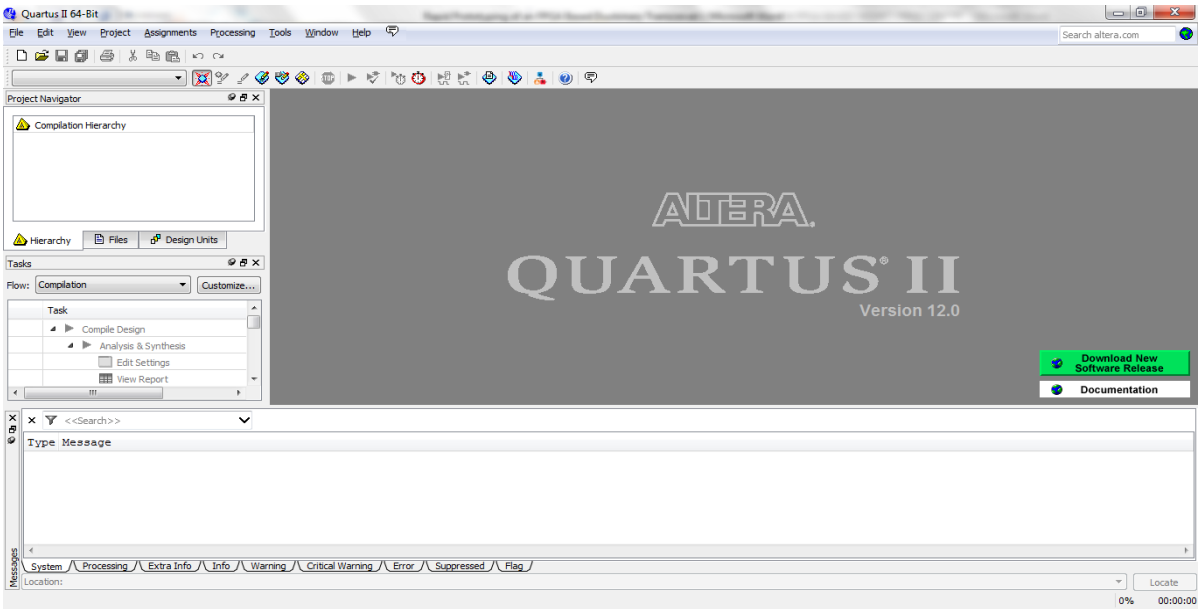


Figure 2-4. Quartus environment

Project Compilation in Quartus deals with checking the syntax of the code to verify its correctness. Other issues include steps like analysis and synthesis, fitting and programming file generation. It consists of some basic functions which include: Analysis and Synthesis which is a process that brings together all files within the design to create a single file that describes the operation of the complete system; Fitting (Place and Route) - As the FPGA comprises many logic blocks, the fitter determines which of these blocks are to be used in implementing the design (fitting) and also how these blocks should be connected (routing). This process is basically an optimization process and typically takes a longer time in FPGAs with more logic elements; Programming File Generation - Here, the actual bit file used in programming the FPGA is generated. When this step completes, it produces a static ram object file (sof) to be used in device programming and this is usually done with the aid of the device programmer; Pin Placement - Up to this stage, we have not stated which pins serve what function on the FPGA. This is done using the pin planner in Quartus. One way to run it is to click on Pin Planner from the Assignments menu in Quartus. The I/Os in the design are then assigned physical pins on the device. After this step, it is sometimes necessary to re-compile the design to update it before device programming is done. An easier and faster way though is to run Analysis and Elaboration then performing pin assignment before running the complete compilation.

2.3 Stratix IV GT SI Board

The Stratix IV GT Signal Integrity (SI) board is a high-speed Altera device widely used for signal processing and SI applications. It has built-in integrated transceivers capable of running at speeds of up to 11.3 Gbps. Transmit and Receive equalization can also be

performed with the board as it has built-in equalizers with programmable taps. A couple of user LEDs, push buttons, switches and seven segment displays also exist on the board to allow for the implementation of basic digital logic designs. The board comes with the EP4S100G2F40I1 FPGA chip that has the Fine-Line Ball Grid Array (FBGA) architecture with 1517 pins.

Two common ways of programming the chip are the embedded Universal Serial Bus (USB) blaster and the MAX II Fast Passive Parallel (FPP) method. The embedded USB-blaster method allows for direct programming of the chip by using a USB cable to connect the board and computer directly. The Quartus programmer in Joint Test Action Group (JTAG) mode is then used to download the programming file to the FPGA chip. The FPP method on the other hand, allows for automatic loading of the configuration to the FPGA on reset or power-up. This method is helpful because the FPGA is volatile, and any configuration downloaded to it is lost on reset or power-up. However, with the FPP configuration, the programming file is stored in flash memory and automatically loaded into the FPGA chip whenever the board is reset or powered up. Figure 2-5 shows the Stratix IV GT SI board.

Figure 2-5. The Stratix IV GT SI board.

2.4 Stratix IV GT Transceivers

Up until now, our discussion has been on logic devices and implementation. An interesting feature of FPGAs is that they can also be used for high-speed data transmission. The FPGA chip on the Stratix IV GT SI board is equipped with 36 high speed transceivers. These transceivers are placed in two sides of the device with three blocks on each of the two sides. Each transceiver bank consists of four data channels and two Clock Multiplier Unit (CMU) channels for providing reference clocks to the transceiver channels. The Auxiliary Transmit Phase Locked Loops (ATX PLLs) also exist within the transceiver block, and can be used to generate the required reference clock required for transceiver operation. Figure 2-6 shows the architecture of the

transceivers on the Stratix IV GT SI board.

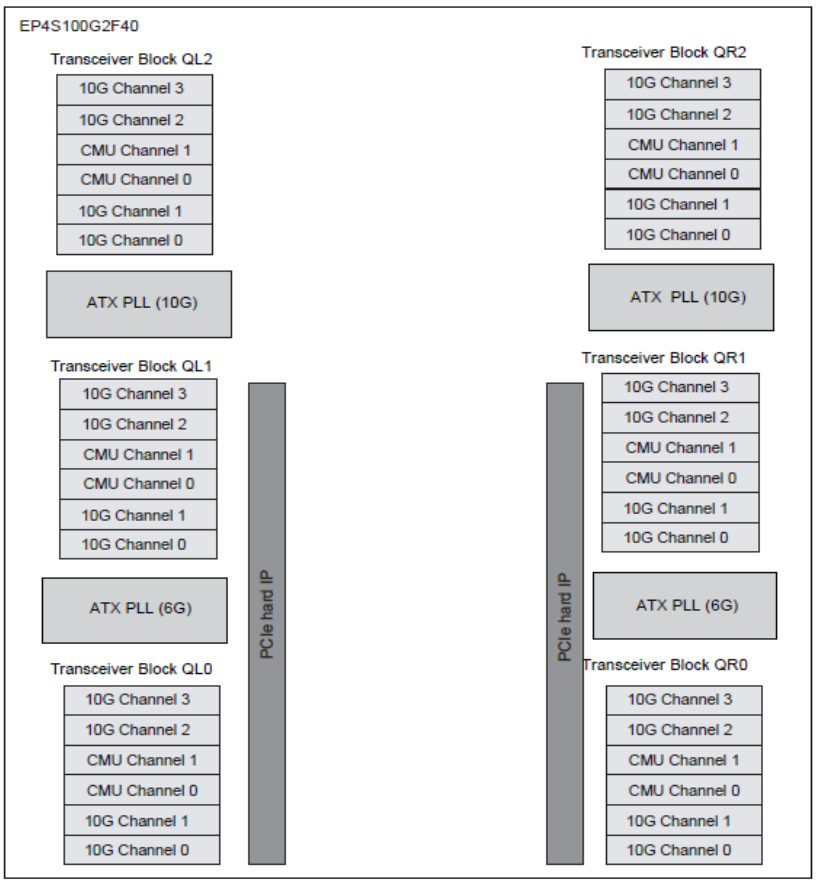


Figure 2-6. Transceiver architecture in the Stratix IV GT SI board

The transceiver channels allow for transmit (pre-emphasis), receive equalization such as Decision Feedback Equalization (DFE), and dynamic reconfiguration. Dynamic Reconfiguration refers to the ability to change the properties of the transceiver such as the equalizer tap settings and the differential output voltage levels as needed without having to first power down the device. Furthermore, oscillators exist for providing the required triggers needed to view eye diagrams on external scopes. These triggers are sent out through dedicated SMA ports available on the board. This makes the board a complete

tool for the analysis and study of high-speed transmission links. The transceivers on the Stratix IV GT board can run at speeds of up to 11.3 Gbps and can support multiple serial protocols. In addition, custom protocols at different speeds can also be implemented on the board. Table 2-1 shows the serial protocols supported by the board.

Table 2-1. Serial Protocols supported by the Stratix IV GT board [12]

Protocol	Description
PCIe	Gen 1 at 2.5 Gbps and Gen 2 at 5.0 Gbps
XAUI	3.125 Gbps up to HiGig at 3.75 Gbps
GIGE	1.25 Gbps
Serial RapidIO	2.5 Gbps and 3.125 Gbps
SONET/SDH	OC-48 and OC-96
(OIF) CEI PHY Interface	4.976 Gbps to 6.375 Gbps
Serial Digital Interface (SDI)	3G-SDI at 2.97Gbps and 2.967 Gbps

Figure 2-7, shows one transceiver channel of the Stratix IV GT SI board

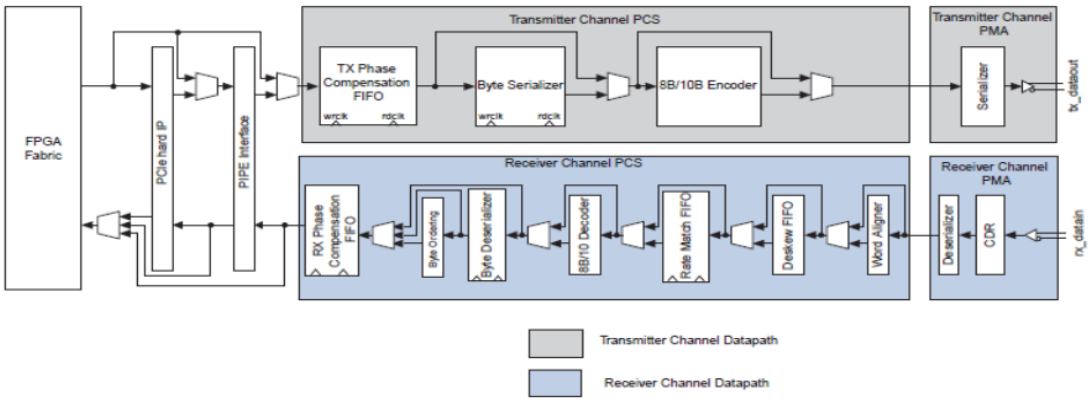


Figure 2-7. One channel of the Stratix IV GT transceiver

The transceiver has two main components: the Physical Coding Sublayer (PCS), and the Physical Medium Attachment (PMA). These components exist both in the transmitter

and receiver sections of a transceiver channel. The next few sections briefly outline each of the blocks, starting from the transmitter channel PCS.

2.4.1Transmit Phase Compensation FIFO

The transmit phase compensation FIFO serves as an interface between the fabric of the FPGA and the PCS of the transmitter. It also compensates for any phase difference that might arise between the FPGA fabric interface clock and the low-speed parallel clock generated by the clock dividers.

2.4.2 Byte Serializer

There is normally a maximum frequency at which the FPGA is able to run. Sometimes when the data rate of the transceivers is high, this maximum frequency is exceeded. To address this issue, the byte serializer allows the FPGA to run at half the frequency and double the bit width. For example, clocking a 20-bit parallel data source at 100 MHz at the input of the byte serializer, we can have two 10-bit parallel data sources clocked at 200 MHz at the serializer output.

2.4.3 8B/10B Encoder

The 8B/10B encoder allows for the conversion of an 8-bit data word into a 10-bit word by encoding the lower 5 bits into 6 bits, and the upper 3 bits into 4 bits. This ensures a running disparity between ones and zeroes in the data stream which allows for Direct Current (DC) balance and easier clock recovery.

2.4.4 Serializer

The serializer converts the incoming parallel data from the PCS of the transmitter into serial data, and sends it to the transmit buffer, allowing for pre-emphasis and modification of the differential output voltage before the data is transmitted serially on the channel. Output termination which could be off or on-chip is also performed at the transmit buffer. To perform off-chip termination, resistors have to be soldered on the board while on-chip termination is enabled from the Assignment Editor tool within the Quartus II software and no external resistors are required.

The next few sections review the blocks in the receiver section of the transceiver, starting with the receiver input buffer.

2.5.1 Receiver Input Buffer

The receiver input buffer receives the serial data from the channel and forwards it to the Clock and Data Recovery (CDR) unit. Just like the transmit buffer, a number of functions can be achieved, among which are equalization, DC gain and On-Chip Terminations (OCT). The equalizers in the transceivers of the Stratix IV GT board support 16 equalization settings that provide up to 16 dB boost of the attenuated high frequencies. DC gain is also performed here, whereby the signal is boosted across its frequency spectrum. Gain values range from 0 dB to 12 dB in increments of 3 dB.

The receiver input buffer also allows for adaptive equalization which is necessary for different data rates and varying channel conditions. It is normally challenging to determine the optimum equalizer tap settings for these varying channel conditions. There

are three modes of operation for the Adaptive Equalizer (AEQ): continuous, powerdown and one-time. The first two modes are not supported in the Stratix IV GT device. The one-time mode determines the optimum tap settings for the equalizer and then locks these setting preventing further changes even if channel conditions vary.

The PMA channel of the receiver also contains what is known as the EyeQ block. When enabled, the EyeQ hardware allows the CDR to sample the incoming data at 32 different positions within a Unit Interval (UI) of a data eye. At the center of the eye, which corresponds to the optimum sampling point, the BER is zero. Moving away from the center toward any of the edges increases the BER, and with these values, the eye width can be indirectly measured.

2.5.2 Clock and Data Recovery Unit

This unit is mainly concerned with recovering the clock from the incoming serial data stream. The CDR can operate in either of two modes: the Lock to Reference (LTR), or Lock to Data (LTD). In the LTR mode, the CDR tracks the input reference clock which is normally sourced from the dedicated reference clocks (REFCLK) of the transceiver. On the other hand, the CDR tracks the incoming serial data in the LTD mode.

2.5.3 Deserializer

The deserializer performs the opposite function of the serializer, converting the incoming serial data back to parallel, and feeding it to the receiver PCS. Clocks utilized include the high-speed recovered serial clock and the low-speed recovered parallel clock.

2.5.4 Word Aligner

The process of data conversion from parallel to serial (and vice versa) tends to result in misalignment of the data stream. This is addressed by the word aligner which operates in one of two modes: manual alignment, and bit-slip. In the manual mode, an input port is used to trigger the word aligner to look for a specified word pattern. When alignment is lost, the trigger has to be asserted again for the word aligner to function. The bit-slip mode is achieved by slipping a bit into the incoming data stream on every rising edge of a bit-slip input signal until the word alignment pattern is found. Within the word aligner is a Programmable Run Length Violation Detector which is used to determine if the number of consecutive ones or zeros in a data stream has exceeded a specified maximum.

2.5.5 Deskew FIFO

Imperfections in the physical transmission medium can result in skew between lanes of transmitted data. The deskew FIFO takes care of this by aligning the data in the lanes. This feature is only supported in XAUI mode.

2.5.6 Rate Match FIFO

The rate match FIFO is used to compensate for differences between the transmitter and receiver clocks. It inserts SKP symbols when the receiver reference clock frequency is greater than that of the transmitter reference clock frequency, and does the opposite when the transmitter reference clock frequency is greater than the receive clock.

2.5.7 8B/10B Decoder

The 8B/10B decoder performs the reverse operation of the 8B/10B encoder by converting a received 10-bit data stream back to an 8-bit data stream with a 1-bit control identifier.

2.5.8 Byte Deserializer

Just as in the transmitter side, the byte deserializer is needed to resolve issues with maximum FPGA clock speed. In high-speed systems, the clock speed of the receiver PCS might be greater than the maximum supported clock speed of the FPGA. In such cases, the byte deserializer is used to halve the clock rate by deserializing the data stream before forwarding it to the FPGA core for error checking.

2.5.9 Receiver Phase Compensation FIFO

This block ensures reliable data transfer between the receiver PCS and the FPGA fabric. It also compensates for the phase difference between the receiver PCS clock and the FPGA fabric clock. The transceivers are extremely flexible as it is possible to include/remove certain blocks in the PCS of both the transmitter and receiver. In fact, the whole PCS can be bypassed to implement what is known as the PMA-only channel.

2.6 Mechanisms for Signal Degradation in Backplanes

The next few sections briefly discuss degradation issues involved in high-speed digital data transmission. Whenever a signal is transmitted through a channel, be it wireline or wireless, it suffers from some sort of degradation as a result of the channel characteristics, or some other unwanted external signal in the form of noise. As a result,

what leaves the transmitter is not always the same as what reaches the receiver. Some of the mechanisms for signal degradation are outlined in the following subsections.

2.6.1 Cross Talk

Crosstalk is simply the coupling of energy from one transmission line to another. This could be as a result of inductive or capacitive coupling. Inductive coupling occurs when changing electric current flows through a conductor. This gives rise to a magnetic field that induces another current in the adjacent transmission line. Capacitive coupling on the other hand occurs due to the capacitor formed when two conductors are separated by a distance. Crosstalk could be Near End Crosstalk (FEXT) or Far End Crosstalk (FEXT). NEXT is crosstalk measured at the same side of the conductor while FEXT is crosstalk taken at opposite ends of the conductor.

2.6.2 Return Loss

Return loss is a ratio of the signal reflected from a Device Under Test (DUT) to the signal launched into the DUT. This occurs as a result of discontinuities in the link or impedance mismatches. Return loss is measured in decibels and because the logarithm of a number less than one is negative, the return loss is usually a negative number.

2.6.3 Reflection

Similar to return loss, reflection occurs as a result of impedance mismatch along the channel. As a result, not all the energy transmitted from the transmitter reaches the receiver and this makes detecting the signal difficult resulting in errors. Also, if multiple

discontinuities exist, multiple reflections will occur leading to an even worse channel performance.

2.6.4 Skin Effect

At high frequencies, current tends to flow on the surface of a conductor rather than its whole cross-section. This reduces the effective cross sectional area for conduction thereby increasing resistance and causing attenuation.

2.6.5 Dielectric Loss

The dielectric constant is a property of Printed Circuit Boards which affects the impedance of a transmission line. It is normally determined by comparing its effect on a conductor pair to that of a conductive pair in vacuum. Materials with lower dielectric constants offer less degradation to signals passing through them meaning they can support transmission over longer distances than those materials with high dielectric constants [13].

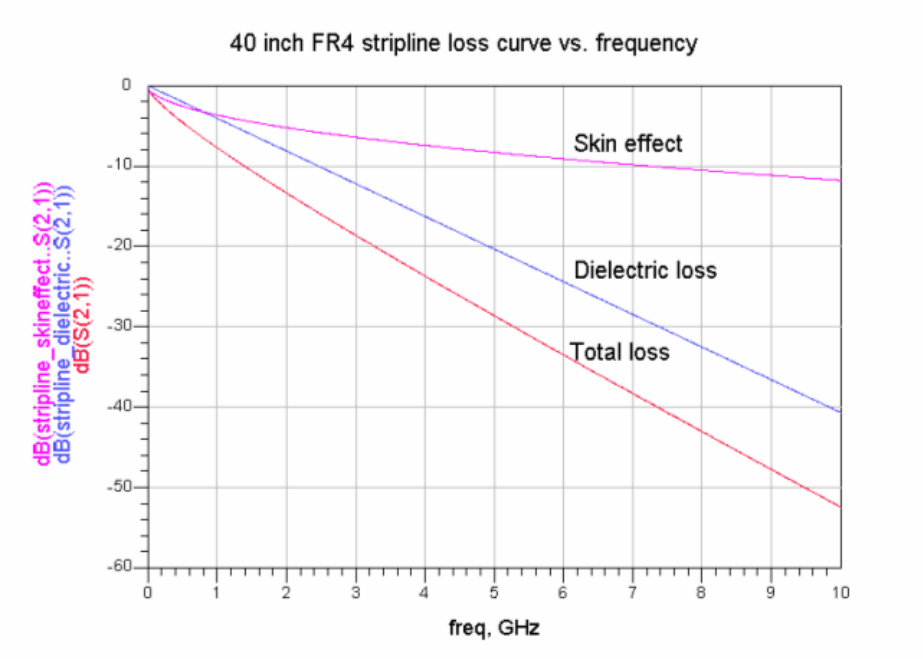


Figure 2-8. Loss curve versus frequency for a typical backplane. [20]

Table 2-2. Some materials with their dielectric constants [20].

Material	Dielectric Constraint
FR-4	3.9 – 4.7
GETEK	3.5 – 4.3
Polyimide	4.0 - 4.5
Speedboard N	3.0
Rogers (RO3003)	3.0

At high frequencies, the dielectric loss is more dominant than skin effect as it varies in proportion with frequency, while skin effect varies with the square root of the frequency. This can be seen in Figure 2-8 [20].

2.7 Effects of Signal Degradation

The mechanisms for signal degradation mentioned above lead to the following detrimental effects on the signal being transmitted:

2.7.1 Attenuation

This is simply the decrease in the energy of the signal being transmitted making signal detection at the receiver end especially difficult when there is a significant amount of noise in the channel. Attenuation can be caused by skin effect and reflection.

2.7.2 Inter-Symbol Interference

Due to the loss mechanisms discussed above, backplane frequency responses are usually low- pass in nature. The attenuation of the high frequency components in the frequency domain tends to spread out the transmitted signal in time, thereby leading to Intersymbol Interference ISI. Two methods generally exist to mitigate this detrimental effect. The first is to design bandlimited pulses (i.e Nyquist pulses) by pulse-shaping. The ideal Nyquist pulse in the time domain is the sinc pulse which is non-causal in the time domain and therefore not realizable. Approximations to this, such as the raised cosine and root raised cosine pulse, are used instead. A second approach used to tackle the effects of ISI is that of equalization which can loosely be defined as the process of flattening out the frequency response of the channel by incorporating certain filters in cascade with the channel.

2.8 Duobinary Signaling

Most of the mechanisms for signal degradation mentioned above are frequency dependent. This implies the need for a solution to the frequency dependent data loss present in the channel. One of such solutions is the use of multilevel signaling schemes like duobinary signaling. Duobinary signaling is a three level partial response signaling that is generated by delaying and adding two subsequent bits. Duobinary coding should not be confused with other line coding techniques that utilize three levels like ternary and pseudo-ternary. These techniques use independent bit levels where transitioning from one level to any other is possible, as opposed to the duobinary system where certain transitions are not allowed. Ternary and pseudo-ternary codes both occupy the same bandwidth as the NRZ signaling [3]. Figure 2-9 shows a comparison between the bandwidth of NRZ and duobinary signaling techniques.

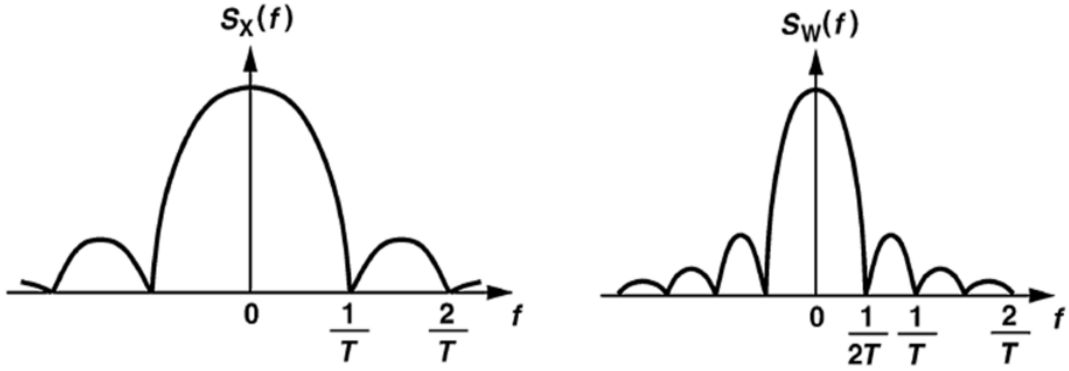


Figure 2-9. Bandwidth Comparison between NRZ and Duobinary signaling [14].

Although duobinary coding leads to ISI, it is done in a controlled manner such that no ISI is present at the sampling instances. As a result, it is now possible to transmit at speeds faster than the Nyquist rate [22].

The duobinary bandwidth is half that of NRZ. This means performing equalization over a narrower frequency range, thereby making equalizers less complex. In addition, nulls that occur in the transfer function of the backplane do so towards the higher end of the frequency spectrum, so that having a reduced bandwidth helps mitigate this null effect. Duobinary has built-in error checking ability because certain transitions are not allowed [19, 24]. For the same reason, duobinary has better immunity to crosstalk/reflection than PAM-4, and the effects of crosstalk/reflection are proportional to maximum transitions in a signal [3]. A duobinary signal can be generated by using delay and add logic, backplanes that create the proper ISI or the use of filters and backplanes among other methods [4]. Duobinary has an advantage over NRZ in terms of bandwidth and over PAM-4 in terms of complexity and power [1]. It requires two decision threshold levels as opposed three for PAM-4 thereby making receiver design less complex and provides only a 2.1 dB SNR penalty over NRZ, while that of PAM-4 over NRZ is around 7 dB. So even though duobinary has the same bandwidth as PAM-4, it has a 5 dB SNR advantage. In addition, duobinary has just two decision thresholds as opposed to three for PAM-4 [1]. So for all the reasons mentioned above, duobinary has been chosen as the proposed solution to the high frequency problem in this research.

2.9 Equalizers

As discussed earlier, the channels tend to attenuate the high-frequency components of the signal being transmitted. A natural line of thought to address this would be to increase the signal power. However, this does not solve the problem as the attenuation is frequency-dependent. Moreover, noise is not considered in this case, as any increase in signal power leads to a corresponding increase in noise power. Most importantly, however, will be the overall increase in the power requirements of the system, which is not desirable. Equalizers are electronic circuits that are used to aid in flattening out the frequency response of backplanes by amplifying or boosting only the high- frequency contents of signal that suffer from attenuation. The equalizers normally have an inverted response of the channel, so that when cascaded with a channel, a flat response is produced. When considered in the time domain, equalizers are used to prevent ISI, since they practically restore the high frequency components of the signal which aids in better detection at the receiver. Equalization, which can be performed at both the transmitter and receiver, basically fall into two categories as discussed in the following subsections.

2.9.1 Linear Equalizer

The linear equalizers include the zero-forcing equalizers and adaptive types. Linear equalizers simply invert the backplane response which is usually a problem whenever nulls exist in the frequency response, as an inverted null is undefined. This leads to the need for higher order networks for equalization. Furthermore, the zero forcing equalizer does not account for the noise in the system, and hence could worsen the system's noise

performance. As boards are of different lengths, materials, and data rates of operation, it is unlikely that a single set of equalizer taps will be adequate for efficient equalization; these taps need to be adjusted for varying conditions, thereby requiring the need for equalizers whose taps are changed based on channel conditions and other factors.

Adaptive equalizers adjust their coefficients in response to a channel whose properties vary with time. An identified stream of data known as the training sequence is transmitted to the receiver. The transmitted and received sequences are then compared, and based on the error at the receiver, the coefficients are adjusted. This adjustment is normally done based on the principle of optimization by minimizing a metric, usually the Mean Square Error (MSE), Least Mean Square (LMS), and Recursive Least Square (RLS), etc. After coefficient computation, the equalizer taps are adjusted and data transmission continues. Although, adaptive equalization provides better equalization than the zero forcing equalizers, the price paid is increased complexity and power requirements. Moreover, because the adaptive equalizers use training sequences for tap adjustment, a significant amount of overhead is incurred.

2.9.2 Decision Feedback Equalizer

Previous equalizers discussed do not consider noise, and therefore have a poor performance in the presence of noise. A DFE equalizes a signal without corresponding noise amplification. The DFE, which is non-linear, performs equalization by feeding back a weighted sum of previously detected samples to remove their ISI contribution on the incoming sample. A major drawback to the DFE, however, is the tendency for error

propagation, because the DFE assumes the past decisions are correct, and this is not always the case. The DFE basically consists of two filters: the feedforward, and the feedback. Figure 2-10 shows the block diagram of the DFE.

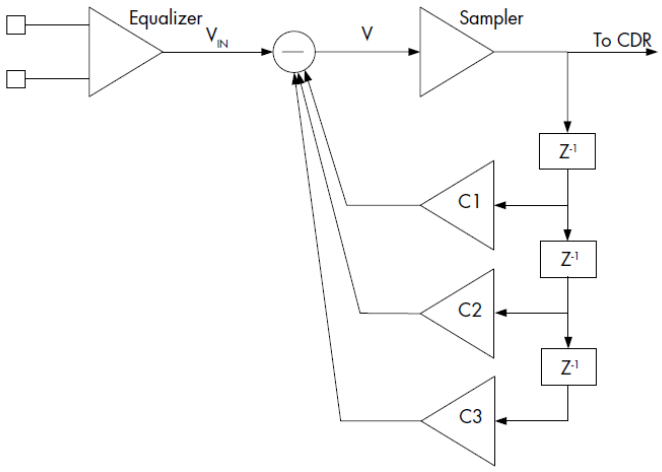


Figure 2-10. DFE Block Diagram

2.10 Eye Diagrams

Eye diagrams (also called eye patterns) are used to evaluate the performance of a digital communication link. They are generated by overlaying several successive symbol intervals of the transmitted signal. The eye height gives information about the noise margin of the system, while the eye width gives information about jitter in the system. The eye diagram can also be used to determine the best time to sample the signal which is normally in the middle of the eye. Figure 2-11 shows a typical eye diagram.

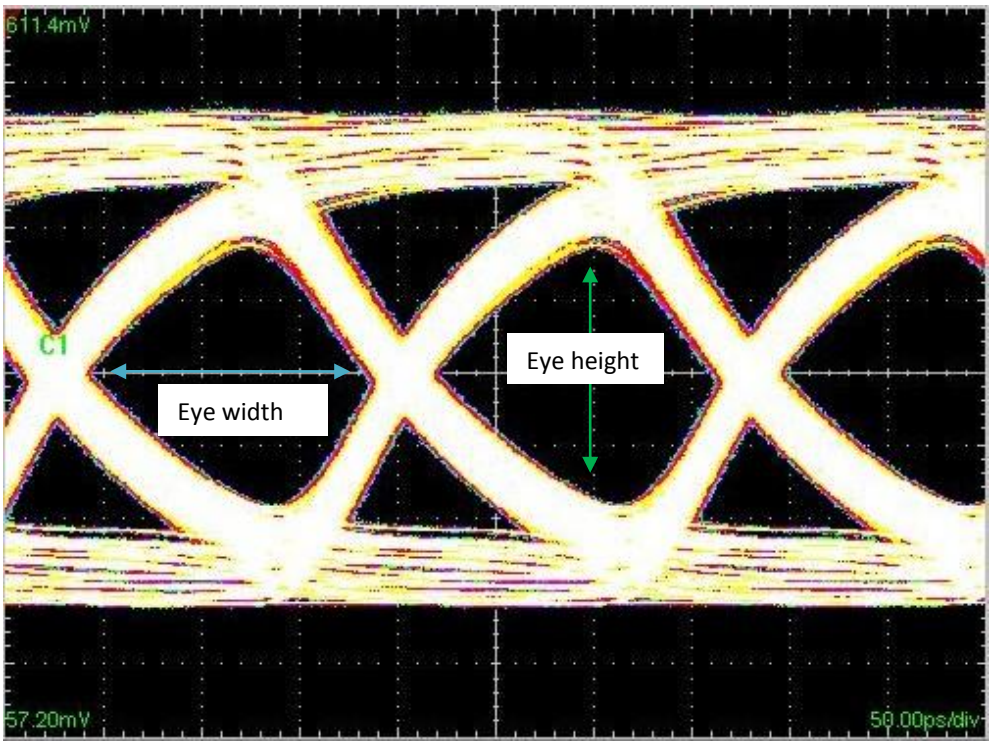


Figure 2-11. A typical eye diagram

Chapter 3

NRZ Simulation and Transceiver Demonstration

3.1 NRZ Simulation

This chapter discusses a simulation in Simulink of a high speed NRZ transceiver system running at 11.3 Gbps and then a replication of the same performance in hardware. This is to ensure that the transceivers in the FPGA are configured correctly as the same method is employed for the duobinary system prototyped in subsequent chapters. The channels used for the simulation and measurement are 29-inch Megtron 6 calibration trace and an FCI backplane with a total length of 32-inches. The transfer characteristic (insertion loss) of the boards was taken with the Vector Network Analyzer (VNA) 5227 from Agilent and the characterization frequency ranged from 10 MHz to 50 GHz with 5000 sampling points and a frequency spacing of 1 kHz. The channel responses are as shown in Figure 3-1.

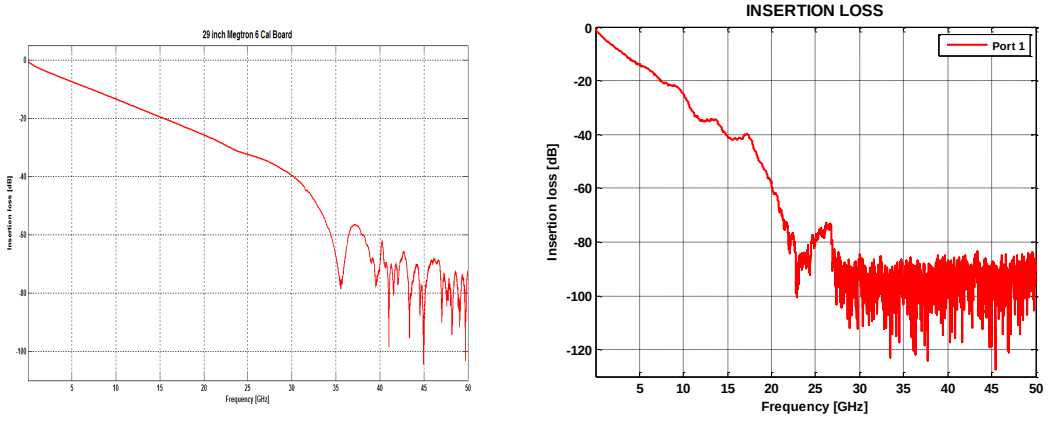


Figure 3-1. (a) Megtron-6 caltrace board (b) FCI backplane response

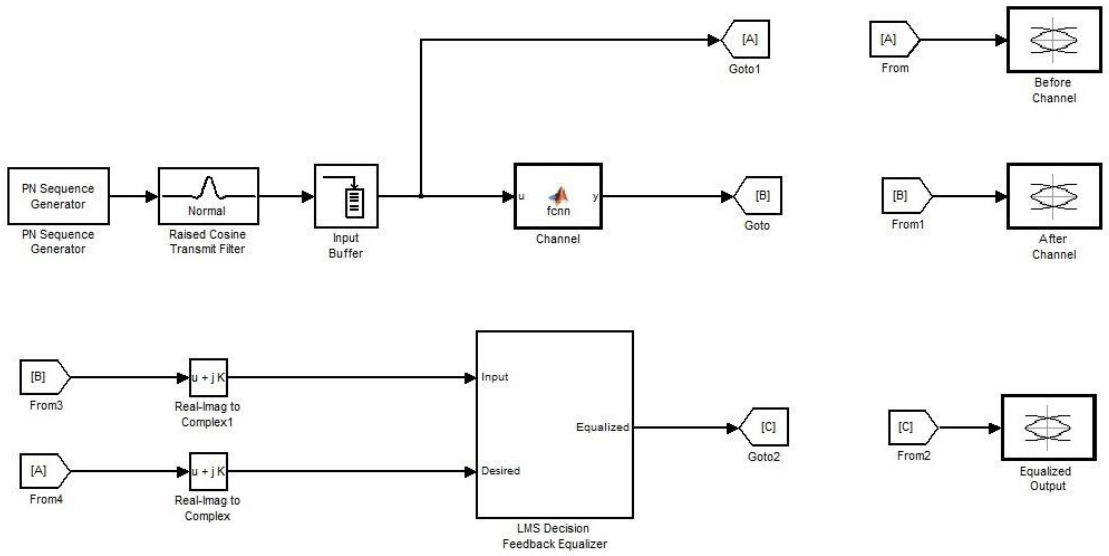


Figure 3-2. NRZ link design in Simulink

The complete system is shown in Figure 3-2. Starting from the block at the left is the NRZ source. This is a PN sequence generator that generates a PRBS 7 pattern of ones and zeros at 11.3 Gbps. The next block is the raised cosine transmit filter which is used for pulse shaping of the incoming random data stream. The frequency domain equation for the pulse shaping filter is shown in equation 3.1 while its bandwidth equation is shown in equation 3.2 [16].

$$\begin{cases} H(f)|_{\beta=1} = \begin{cases} \frac{T}{2}[1+\cos(\pi f T], & |f| \le \frac{1}{T} \\ 0, & otherwise \end{cases} \end{cases} \tag{3.1}$$

$$B_T = \frac{(1+r)R_b}{2} \tag{3.2}$$

Where B_T is the spectral bandwidth

r is the rolloff factor and varies between 0 and 1

R_b is the bit rate

T is the symbol rate

It is required to reduce the bandwidth of the signal to satisfy Nyquist criterion which states that to transmit data at R bits per second (bps), it is required to have at least R/2 Hz of bandwidth. It is a well-known fact that bandwidth is a limited or scarce resource and needs to be conserved, hence the need for the pulse shaping filter. The roll-off factor value for the filter used is 0.6. Figure 3-3 shows the settings window for the raised cosine filter.

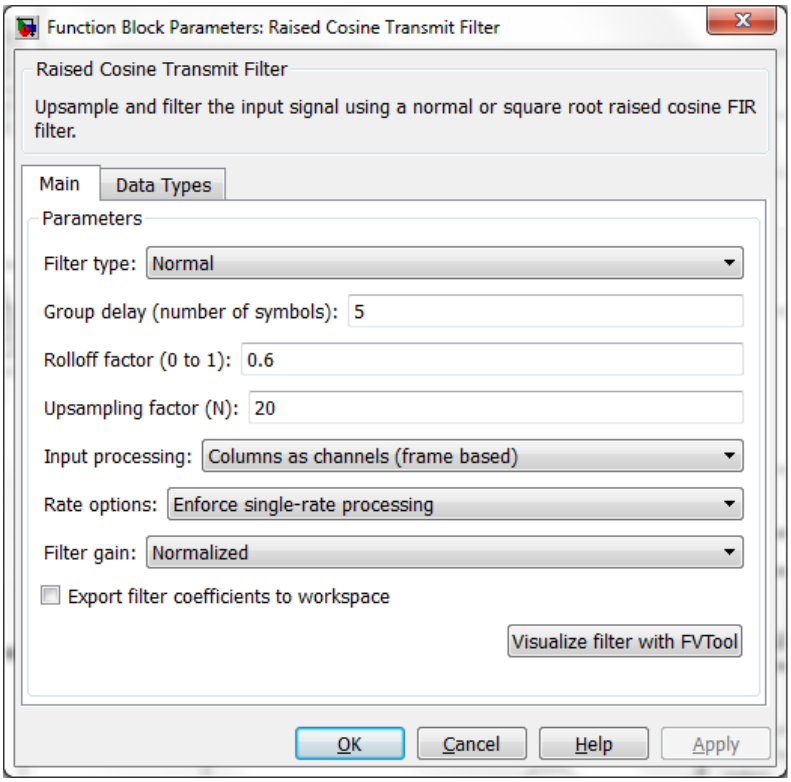


Figure 3-3. Settings window for the raised cosine filter

Next, after the raised cosine filter is the input buffer. This block serves to convert the incoming sample-based signal to a frame-based signal with 5000 samples per frame [21]. Within the MATLAB function block is an m-file that converts the incoming data into the frequency domain using the Fast Fourier Transform (FFT). The result is then multiplied by the insertion loss parameter of the channel used for the simulation after which the result is converted back to the time domain for onward transmission to the receiver. The operation simulates the data passing through the channel. At the output of the MATLAB function block is the output buffer. This performs the reverse operation of the input buffer block. It converts the parallel data back to serial. The final block is the DFE. This

block is used to perform receive equalization and compensate for signal distortion as it propagates through the channel. The DFE uses the Least Mean Square (LMS) algorithm to compute its taps. A two-tap DFE is used in the simulation, and Figure 3-4 shows a window with the DFE settings.

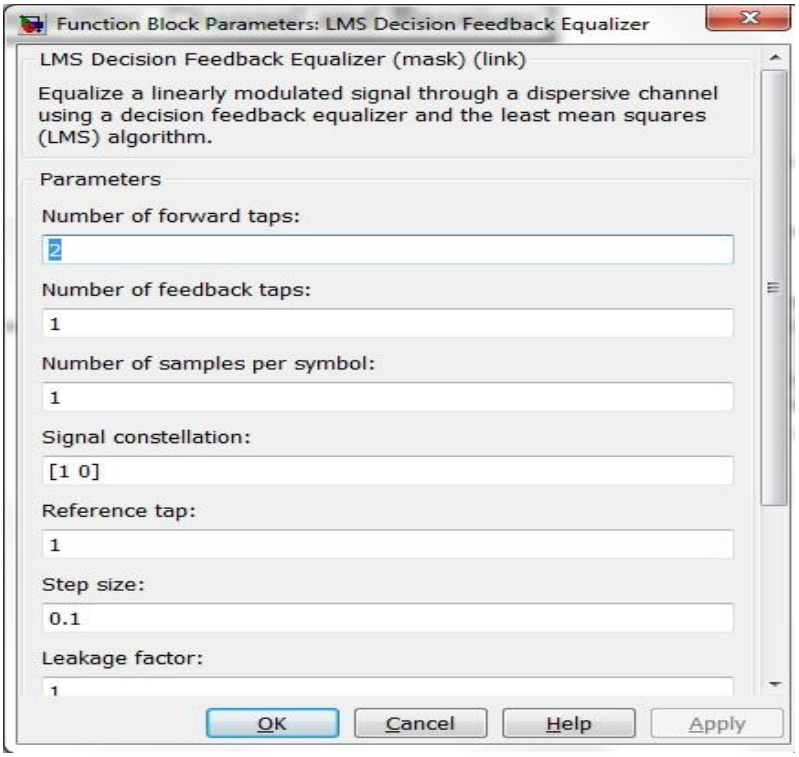
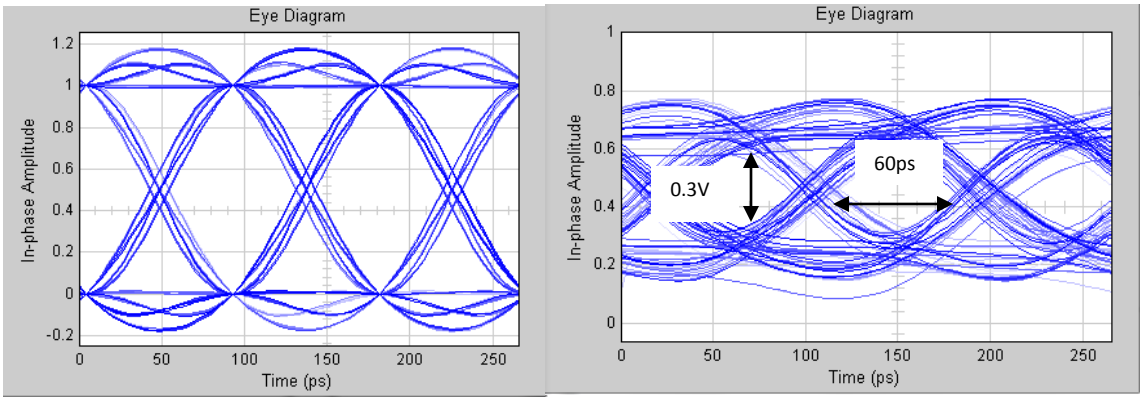


Figure 3-4. DFE settings

The other blocks in the design are eye scopes that are utilized for viewing the eye diagrams. The results are observed in the figures that follow.



(a) (b)

Figure 3-5.(a) Transmitted eye diagram before passing through megtron-6 channel, (b) Eye diagram after passing through megtron-6 channel at 11.3 Gbps before equalization.

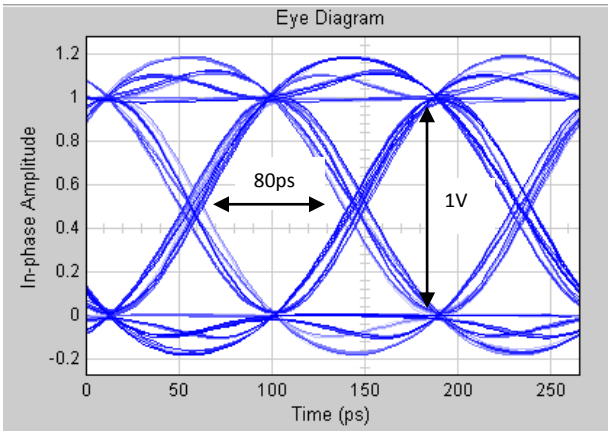


Figure 3-6. Received eye diagram after DFE at 11.3 Gbps

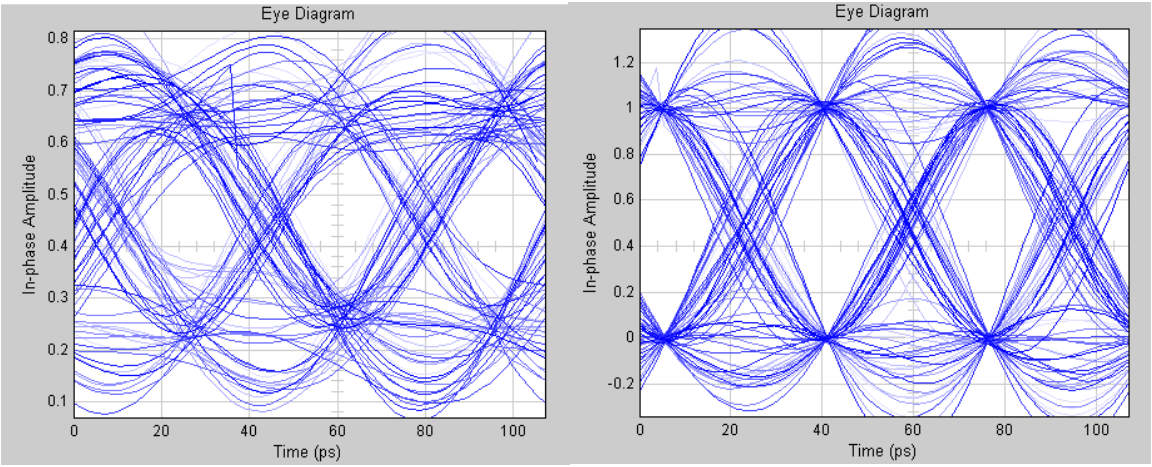


Figure 3-7. (a) Eye diagram after passing through megtron-6 channel at 28 Gbps (b) Equalized eye

After the data passes through the megtron-6 channel, there is significant reduction in the size of the eye as observed by comparing Figure 3-5 (a) and Figure 3-6. After equalization is performed, the eye is open again, this time looking similar to the transmitted data before passing through the channel. This is observed by comparing Figure 3-5 (a) and Figure 3-5 (b). Results obtained indicate the simulation is running correctly.

Table 3-1. NRZ simulation results for 11.3 Gbps and 28 Gbps for the megtron-6 channel

Data Rate (Gbps)	11.3		28	
	Before Eq.	After Eq.	Before Eq.	After Eq.
Eye Height (V)	0.3	1	0.3	1
Eye Width (ps)	60	80	20	30

Table 3-1 shows simulation results for data rates of 11.3 Gbps and 28 Gbps. The eye heights remain almost the same while the eye widths significantly decrease. Measurement at 28 Gbps is not performed because the maximum data rate supported by the FPGA used is 11.3 Gbps.

3.2 Transceiver Demonstration

The next few sections explain how to use the MegaWizard and Qsys to parameterize the transceivers in the FPGA, and then an example reference design [15] is used to operate the transceivers at 11.3 Gbps using a PRBS 7 data pattern. The implemented architecture is shown in Figure 3-8. The pattern is transmitted through the channel and the eye diagram viewed using the Digital Serial Analyzer (DSA) 8200 sampling scope from Tektronix. The eye height, eye width, RMS and peak jitter are observed. Trigger for the scope is supplied by a dedicated SMA pin on the FPGA. The transceiver toolkit is also run to demonstrate dynamic reconfiguration of some parameters such as pre-emphasis

and DC equalization gain of the transceivers. The protocol demonstrated is the basic double width mode with a parallel data size of 40 bits. Quartus II software is used for compilation and programming of the FPGA.

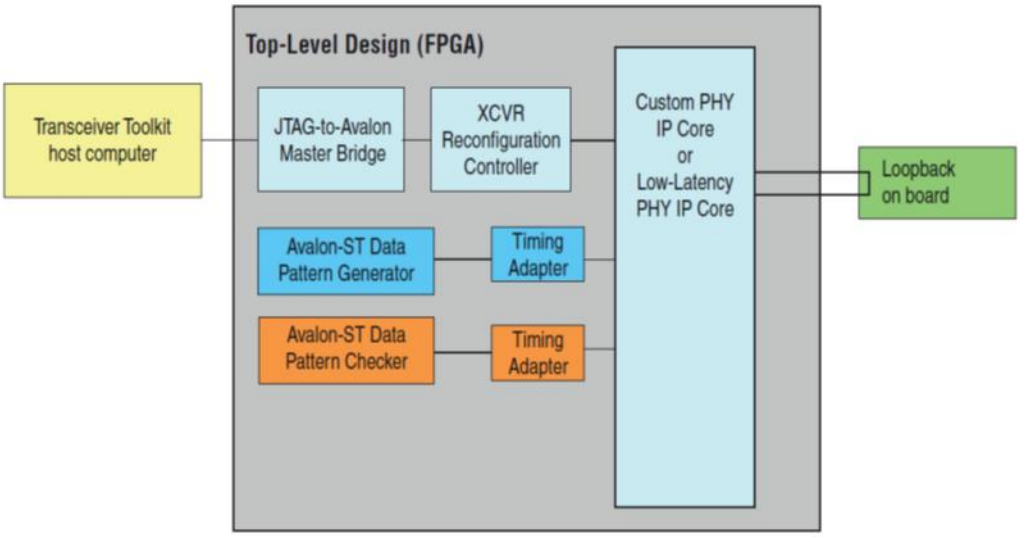


Figure 3-8. Design architecture demonstrated with the transceiver toolkit

The transceiver architecture is shown in Figure 3-8. It comprises of a JTAG to Avalon master bridge which allows for manipulating the reconfiguration controller and by extension the equalizers in the FPGA. There is also a data pattern generator and checker as well as timing adapters to take care of any timing delays within the system.

The next step is to initiate the transceiver. The on-board loopback feature is turned off and the data pattern is transmitted directly to a scope for viewing the eye diagram.

The first step is to start the MegaWizard Plug-In Manager from the Tools menu of the Quartus software to create a new custom megafunction variation. The device to be used

and output file language are then specified in the dialog box that appears next. The transceiver Megawizard function, ALTGX, is also chosen from the tab on the left. This is as seen in Figure 3-9.

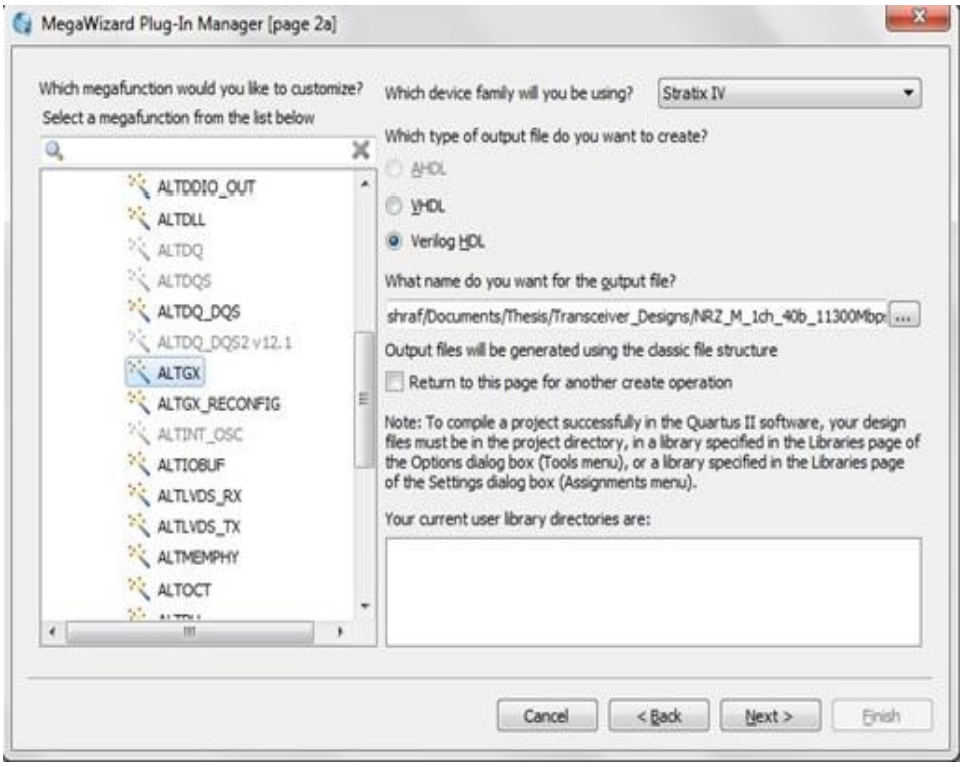


Figure 3-9. Transceiver definition window

The window that appears next allows for specifying what device to use, the intended data rate and protocol, among other settings. The EP4S100G2F40I1 chip on the Stratix IV GT board used is the fastest with a speed grade of 1 [17]. The protocol implemented is basic double width mode with no sub-protocol. This means there is no PCS functionality incorporated in the design [12]. The sub-protocols allow for lane bonding which enables multiple transceiver channels to be connected together but this is not used in this design. The transceivers can be configured for transmitter-only operation, receiver-only operation

and then, the receiver and transmitter configuration. This design uses the receiver and transmitter configuration with a single channel. The transceiver is run in double width mode with a channel width of 40 bits. This is also set in the window. The data rate is set to 11.3 Gbps, for comparison with simulations, with a clock frequency of 706.25 MHz. The same clock frequency is used for 5.65 Gbps. A summary of these settings is shown in Figure 3-10.

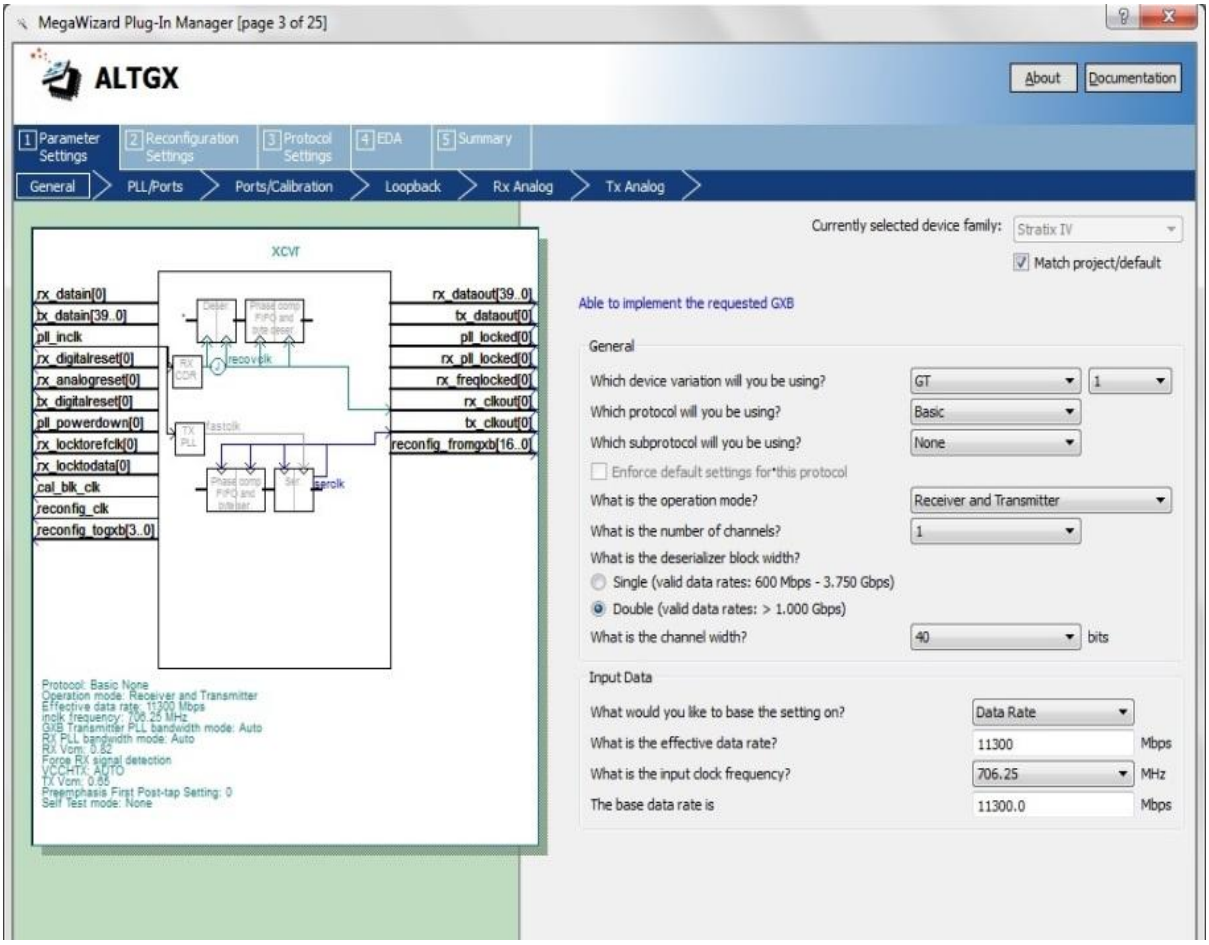


Figure 3-10. Data Rate settings for transceiver

The next step in the settings window is used to specify whether to train the CDR from the PLL input clock. This block is checked to avoid supplying an external clock. Optional ports for controlling and monitoring the transceiver operation are also specified at this stage.

The next window is a continuation of the optional ports. They are retained with the default settings. Moving on to the next page, a termination resistance of 100 Ohms is specified with a receiver common mode voltage of 0.82 V. The DC gain setting is set to 1 which corresponds to a gain of 3 dB [18] across the whole frequency spectrum of the signal (note that in this case a DC gain setting of 1 is doubling the gain). These settings are as illustrated in Figure 3-11.

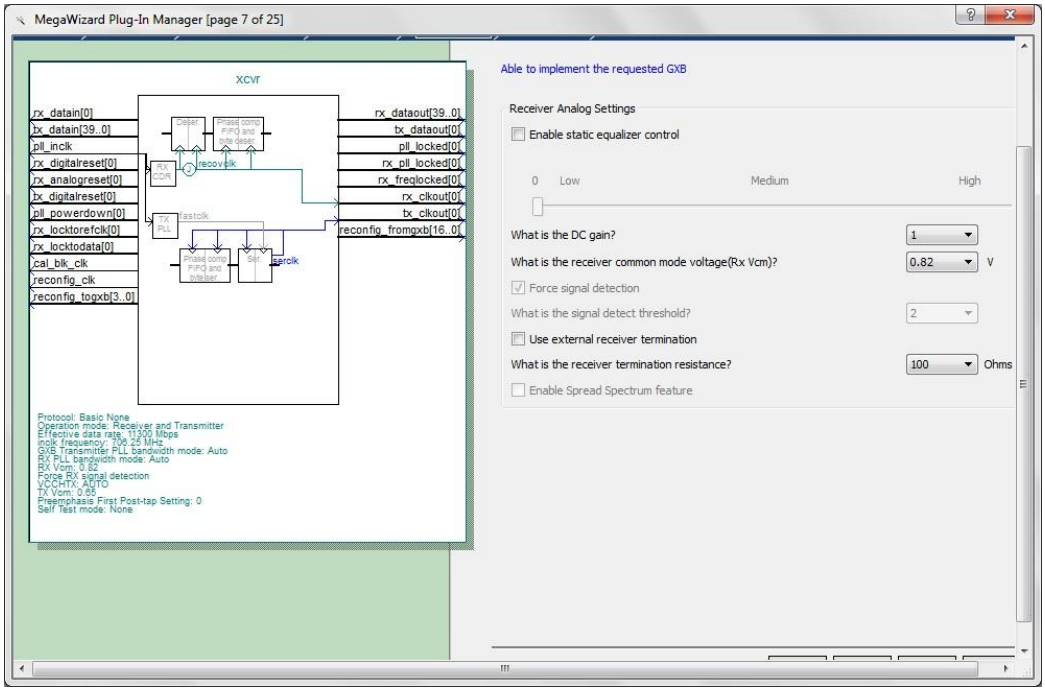


Figure 3-11. DC gain setting

The transmitter analog settings and the dynamic reconfiguration settings remain unchanged, and so are the next few windows. The finish button is clicked to generate the transceiver instance just created. The final window is shown in Figure 3-12.

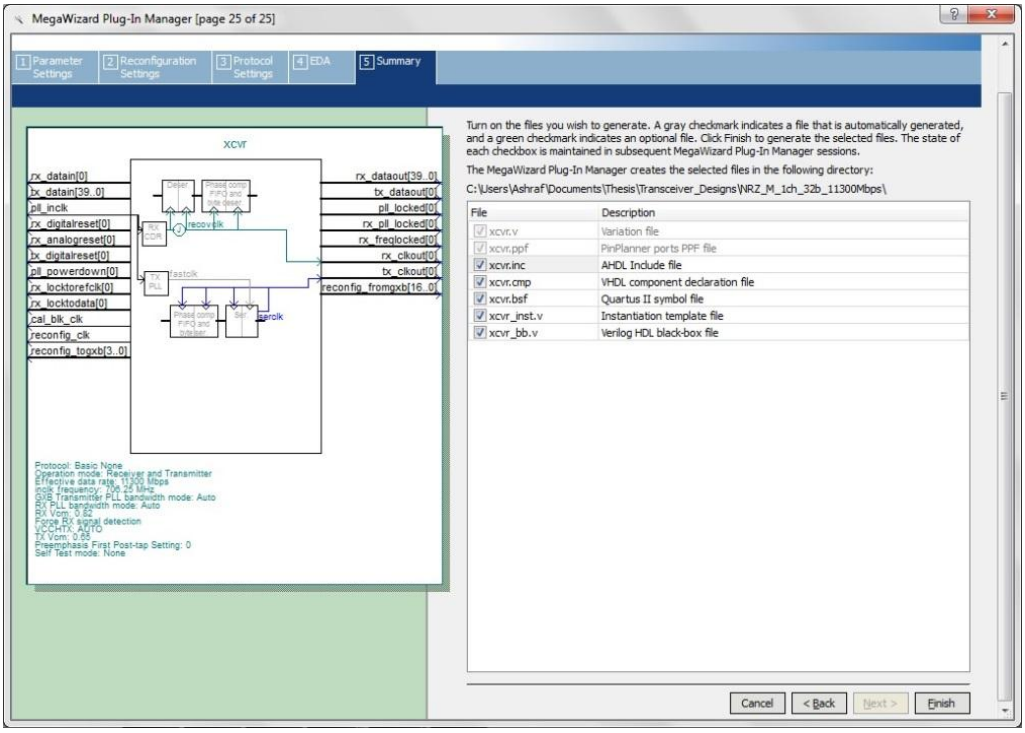


Figure 3-12. Final page for transceiver settings

Now that the code for the transceiver has been generated, Qsys is started and the transceiver is incorporated into the complete architecture of the system. The architecture consists of a data pattern generator, a data pattern checker, the designed transceiver and clocks that enable proper functioning of the complete design. Just like the MegaWizard manager, Qsys is also started from the tools menu of the Quartus software. When the Qsys window opens, New Component is clicked to define the earlier created transceiver

instance. The interfaces with no signals are removed and then the remaining interfaces are renamed. This is reflected in the Signals tab as shown in Figure 3-13.

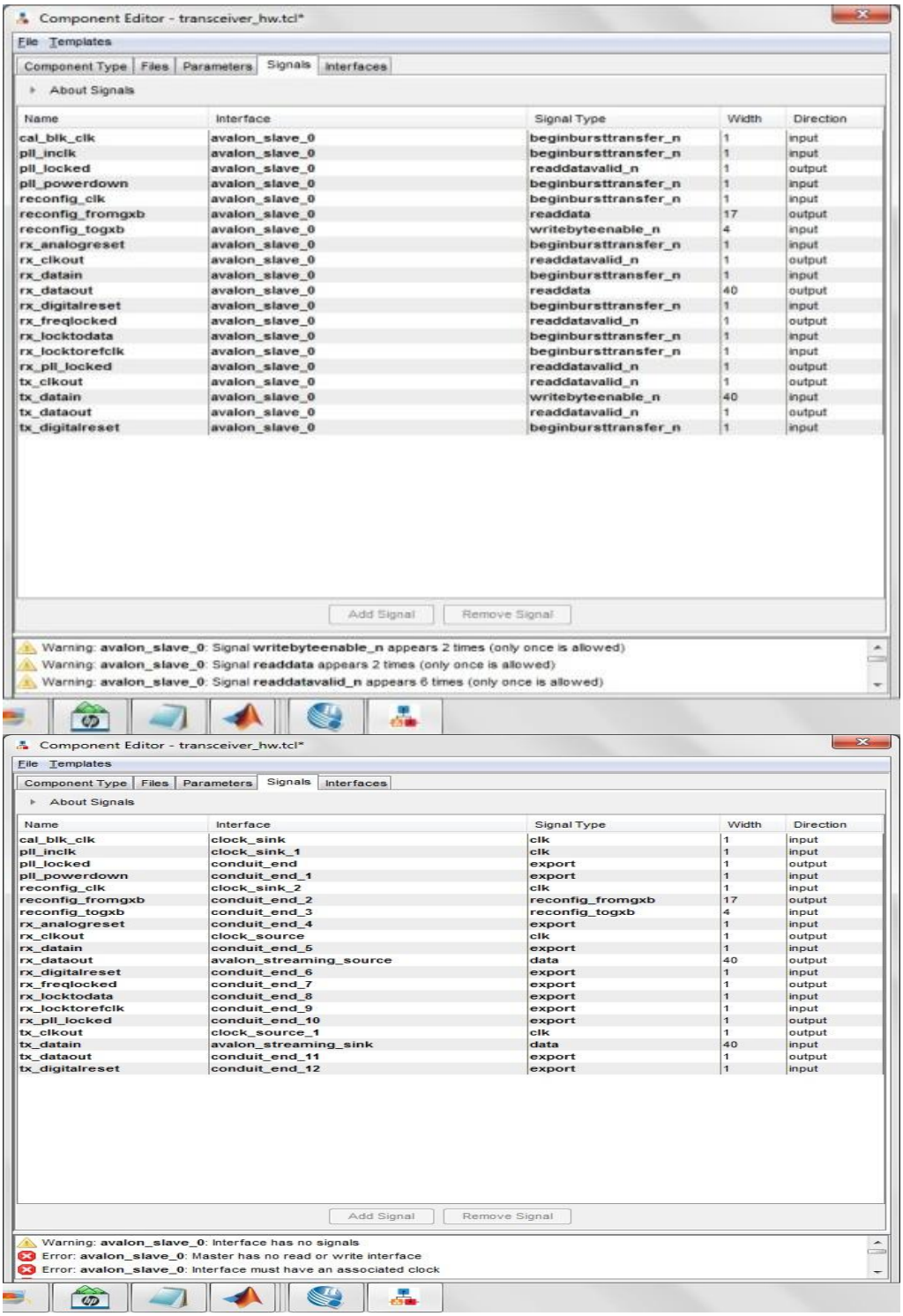


Figure 3-13. Signals tab in component editor

The component is then saved. Other components of the design such as the PLL reference clock source, the reconfiguration clock, the data pattern generator and checker, as well as timing adaptors and JTAG-to-Avalon Master Bridge are added. The final design with the associated connections is shown in Figure 3-14.

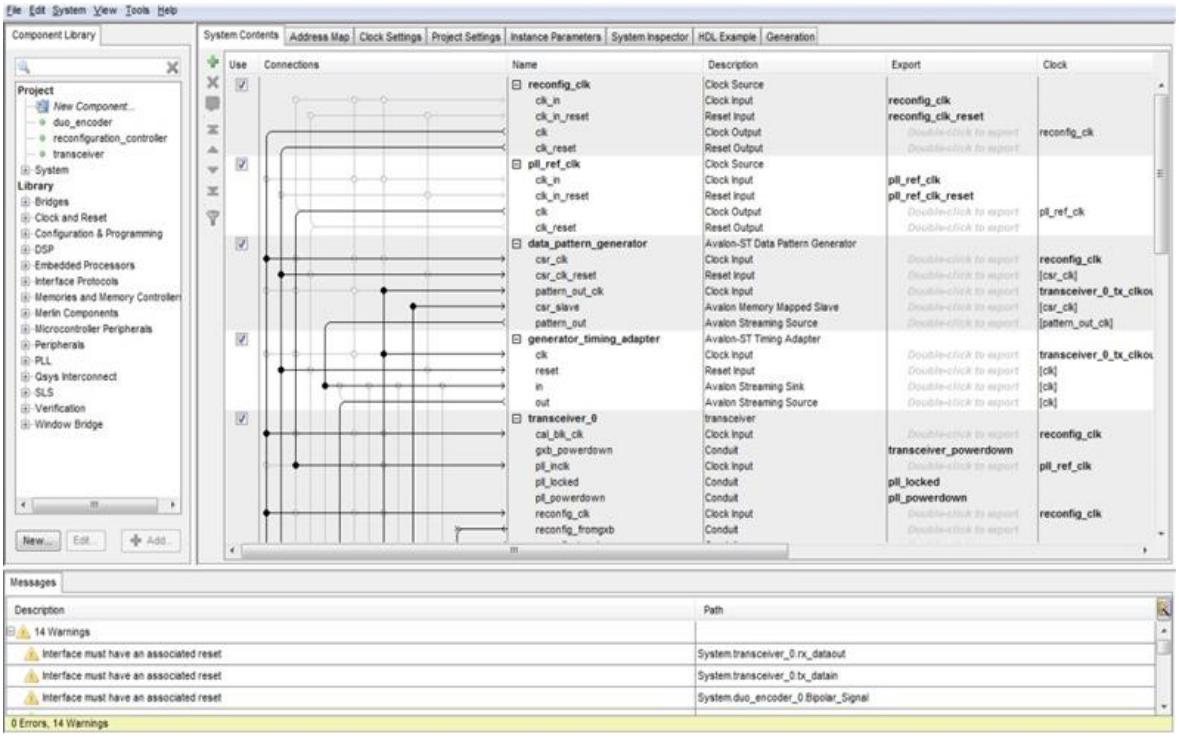


Figure 3-14. Complete NRZ system architecture implemented in Qsys

From the generation tab, the complete system is generated and saved in the job folder. Qsys is closed and a top level Verilog file is written to instantiate the whole system, after which compilation, programming and testing can be performed. After making all necessary settings and modifications in Qsys, the system is generated and instantiated in the top level Verilog file in Quartus. Compilation and device programming is done with the aid of the Quartus software. Figures 3-13 shows a successfully compiled system.

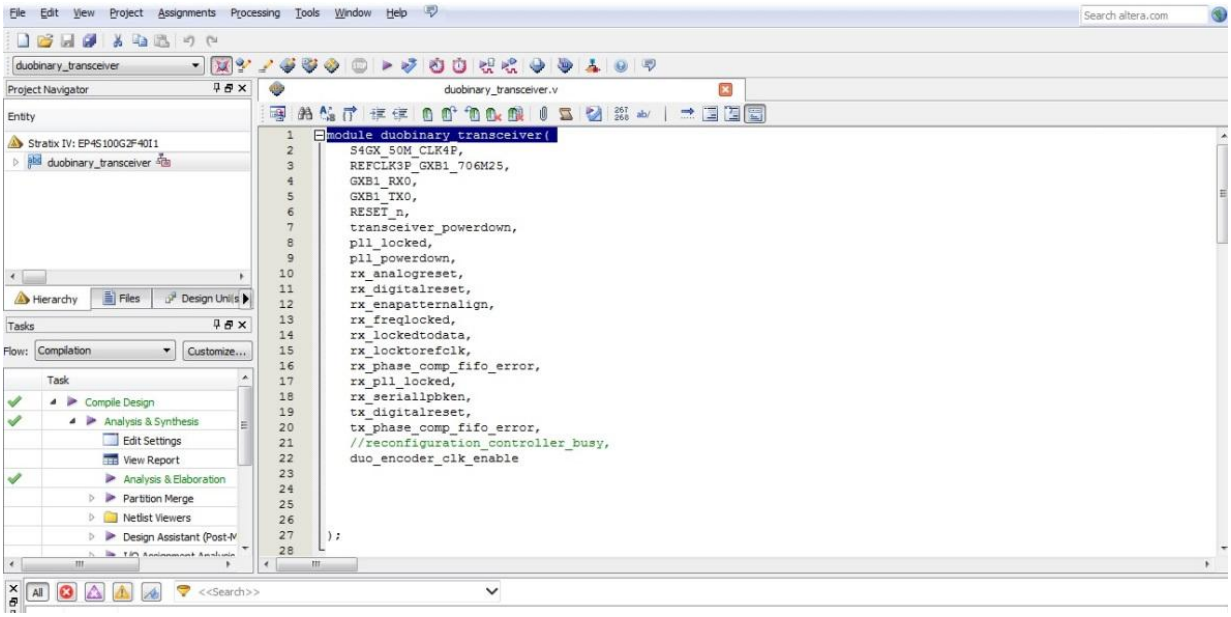


Figure 3-15. Compiled system

3.3 System Testing

To test the transceivers, the reference design example is compiled to generate a programming file that is downloaded to the FPGA. After programming the FPGA, correct operation of the link can be verified with the help of the transceiver toolkit. This can be started from the Tools menu on the Quartus software. Figure 3-16 shows a typical transceiver toolkit window and Figure 3-17 shows the hardware setup for the measurement.

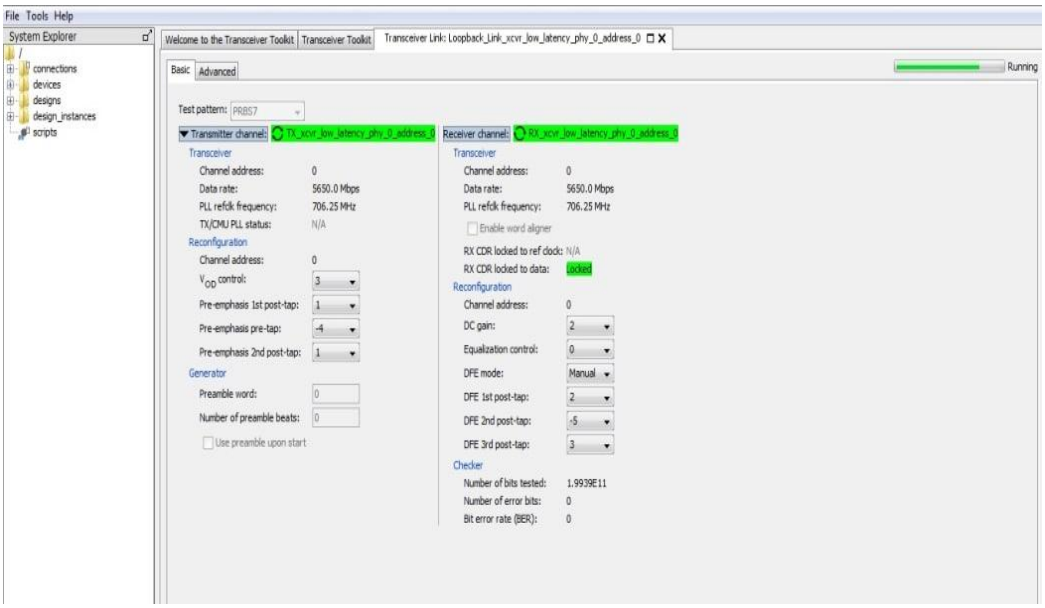


Figure 3-16. Transceiver toolkit Window

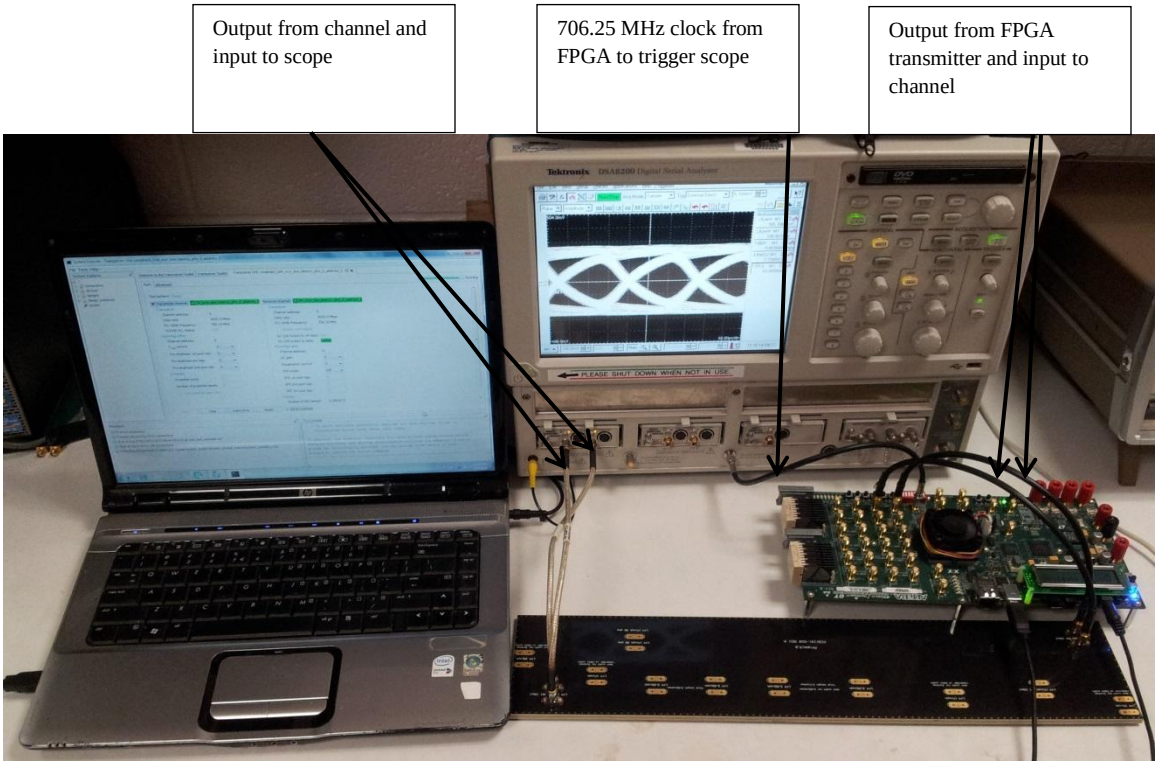


Figure 3-17. Hardware setup

The USB blaster cable is used to connect the FPGA and the computer for programming, and to send dynamic reconfiguration commands from the computer to the FPGA transceivers. SMA cables from the output of the transceivers are connected to the channel at one end. At the other end of the connector, another set of cables are used to connect to the scope. The two channels are tested at data rates of 5.65 and 11.3 Gbps and the obtained eye diagrams are observed using the same scale to show the effect of equalization. Eye diagrams results for the Megtron 6 caltrace at 5.65 and 11.3 Gbps are shown in Figure 3-18 and Figure 3-19, respectively.

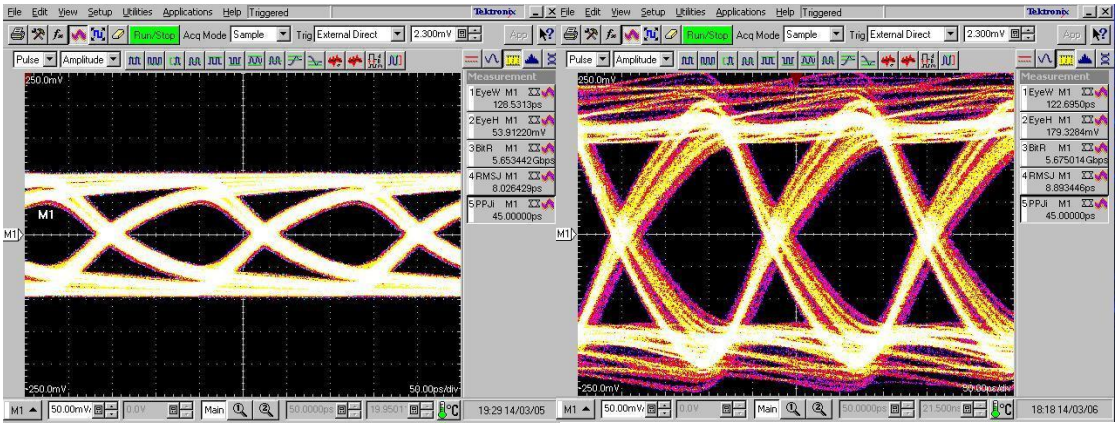


Figure 3-18. Results for Megtron 6 caltrace before and after equalization at 5.65 Gbps

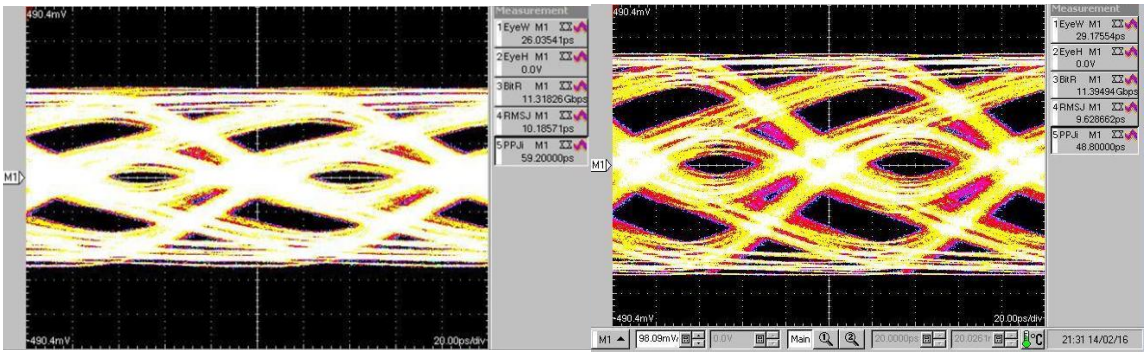


Figure 3-19. Results for Megtron 6 caltrace before and after equalization at 11.3 Gbps

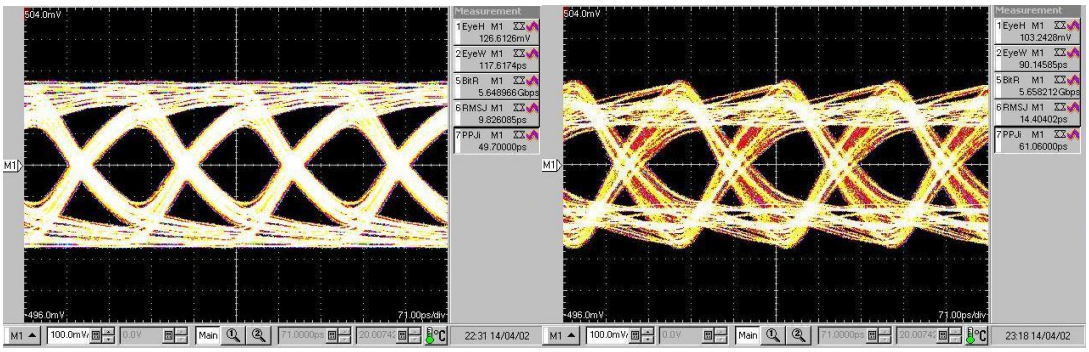


Figure 3-20. Results for FCI backplane before and after equalization at 5.65 Gbps

Results for FCI backplane before and after equalization at 5.65 are shown in Figure 3-20. Comparison between the simulated data in Simulink and the actual measurement for the Megtron 6 board at 5.65 and 11.3 Gbps is shown in Figure 3-21 and 3-22, respectively. At 5.65 Gbps, the eye height and width for simulation are 0.3V and 130 ps, respectively while for measurement they are 0.06 V and 128 ps. As the data rate is increased to 11.3 Gbps, the eye height and eye width for simulation are 0.3V and 60 ps respectively and 0V and 29 ps for measurement. The simulation is slightly better because of the absence of the effects of the cables used for measurement and possible noise within the system. Table 3-2 summarizes these results.

Table 3-2. NRZ simulation and measurement comparison results

Data Rate (Gbps)	5.65		11.3	
	Simulation	Measurement	Simulation	Measurement
Eye Height (V)	0.3	0.06	0.3	0.25
Eye Width (ps)	130	128	60	29

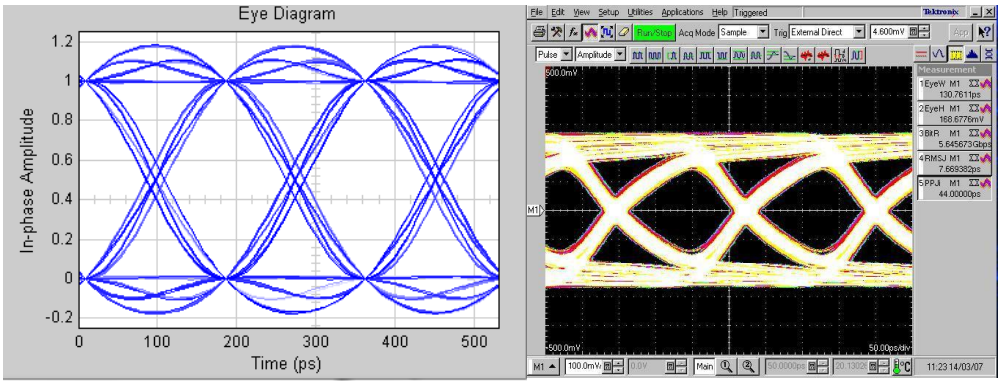


Figure 3-21. NRZ Simulation and measurement eye diagrams at 5.65 Gbps

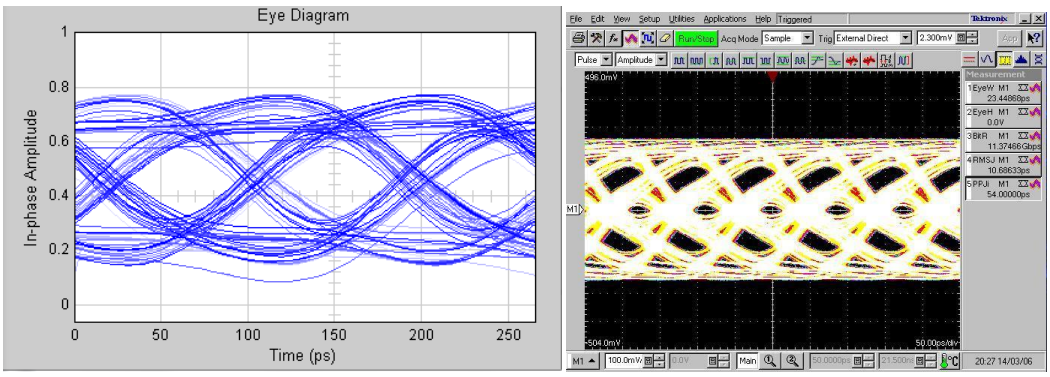


Figure 3-22. NRZ Simulation and measurement eye diagrams at 11.3 Gbps

Table 3-3 illustrates the effect of using the equalizers in the FPGA at data rates of 5.65 Gbps and 11.3 Gbps. It is seen in the table that summarizes the result for the measurements taken on the Megtron 6 channel.

Table 3-3. Summary of NRZ measurement results for the Megtron 6 channel

Data Rate (Gbps)	5.65				11.3			
	Before Eq		After Eq		Before Eq		After Eq	
	Sim.	Meas.	Sim.	Meas.	Sim.	Meas.	Sim.	Meas.
Eye Height (mV)	60	53.9	200	179.32	30	20	200	25
Eye Width (ps)	130	128.53	135	122.69	60	26	80	29

In Table 3-3, it is seen that at a lower data rate of 5.65 Gbps, before equalization, the eye height and width are 53.9 mV and 128.5 ps respectively. However, after equalization is performed, the eye height significantly increases to 179.3 mV while the eye width decreases to 122.7 ps. As the data rate is increased, however, both eye height and eye width degrade; at 11.3 Gbps, the eye height and width are 25 mV and 26 ps respectively before equalization. After equalization, there is no improvement in the eye height and only a slight improvement in the eye width to 29 ps. This is because no receive equalization is performed. Transmit equalization can be implemented for both 5.65 Gbps and 11.3 Gbps while receive equalization can be only performed at 5.65 Gbps as the

Stratix IV GT FPGA is designed to run receive equalization at a maximum data rate of 6.5 Gbps [26].

Chapter 4

Duobinary Simulation and Proposed ASIC architecture

4.1 Duobinary Precoder

The duobinary signal is generated by the addition of two subsequent bits and hence, when decoding the signal from duobinary back to binary there is the tendency for a decoding error in a single bit to propagate through the system. The duobinary precoder circuit is used to prevent such error propagation in the system. This works by modifying the data in such a way that when the data is to be decoded, the error in a given bit does not depend on the previous bits. As a result, not only do we achieve better error performance, decoding the data also becomes much easier. The duobinary precoder consists of a unit delay and an XOR gate as given in equation (4.1)

$$y[n] = x[n]\ XOR\ x[n-1] \tag{4.1}$$

Figure 4-1. shows the structure of a duobinary precoder.

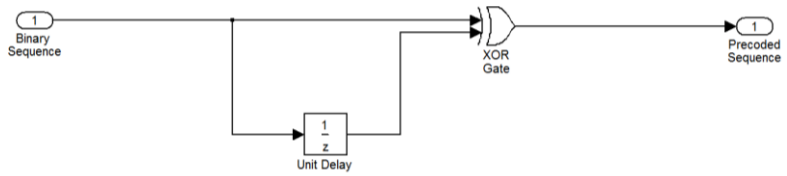


Figure 4-1. Duobinary Precoder

Tables 4-1 and 4-2 use a simple example to illustrate the importance of the duobinary precoder. In Table 4-1, the data stream is converted from bipolar straight to duobinary, in

which case there is no precoding, and a single error can be seen propagating from the fourth down to the seventh bit position.

In Table 4-2 however, where precoding is implemented, the same error remains at bit position four and doesn't propagate any further.

Table 4-1. Duobinary Decoding without a precoder.

Binary	0	1	1	0	1	0	1	1	1	0	0	0	
Bipolar	-1	-1	1	1	-1	1	-1	1	1	1	-1	-1	-1
Duobinary		-2	0	2	2	0	0	0	2	2	0	-2	-2
Decoded		0	1	1	1	0	1	0	1	1	0	0	0

Table 4-2. Duobinary Decoding with a precoder.

Binary		0	1	1	0	1	0	1	1	1	0	0	0
Precoded	0	0	1	0	0	1	1	0	1	0	0	0	0
Bipolar	-1	-1	1	-1	-1	1	1	-1	1	-1	-1	-1	-1
Duobinary		-2	0	0	0	0	2	0	0	0	-2	-2	-2
Decoded		0	1	1	1	1	0	1	1	1	0	0	0

4.2 Duobinary Encoder

The duobinary decoder converts the two-level NRZ signal to a three-level duobinary signal. This can be achieved in a number of ways [7]. In this work, however, the delay and add filter approach is chosen for its simplicity. The duobinary encoding process is as shown in equation (4.2).

$$y[n] = x[n] + x[n-1] \tag{4.2}$$

A block diagram representing the duobinary encoder is as shown in Figure 4-2.

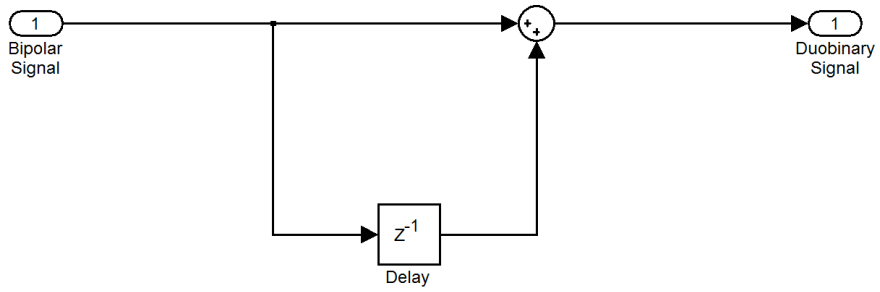


Figure 4-2. Duobinary Encoder.

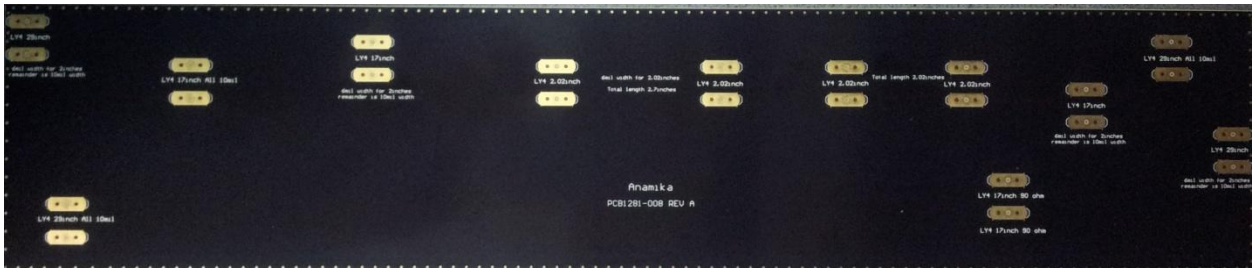
4.3 Pre-Emphasis Filter

As previously discussed in Chapter 2, backplane channels usually have low pass characteristics leading to the attenuation of the high frequency components of the signal being transmitted. As a result, sometimes before signal transmission, the high frequency components of the signal are boosted. This is done with a filter called a pre-emphasis filter. The pre-emphasis filter used for simulation in this work is a simple two-tap FIR

filter whose taps were determined using the Filter Design and Analysis (FDA) tool in Simulink.

4.4 Channels

Two channels are used in this research. The first is a 29-inch Megtron 6 Backplane channel which has been characterized up to a frequency of 50 GHz and the second is an FCI backplane with a total length of 32 inches. The response of these channels is saved in a touchstone (snp) format and MATLAB is used to extract the data. Figure 4-3 and Figure 4-4 shows these channels and their corresponding responses.



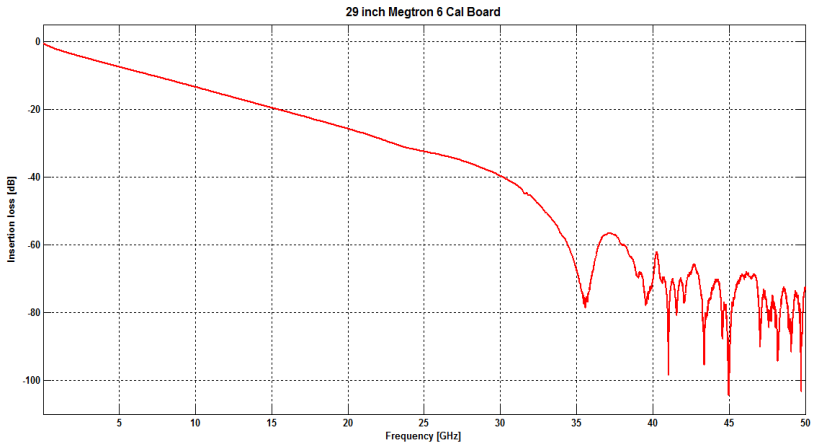
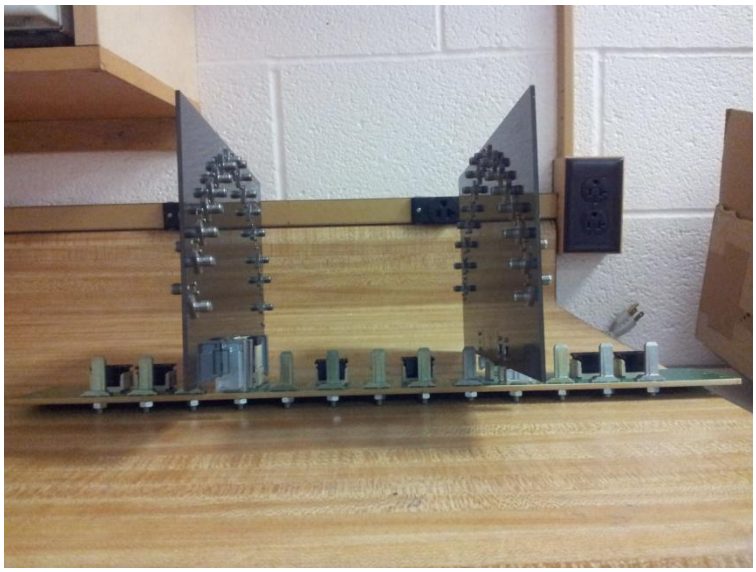


Figure 4-3. (a)Megtron 6 Cal Trace board and (b) channel response



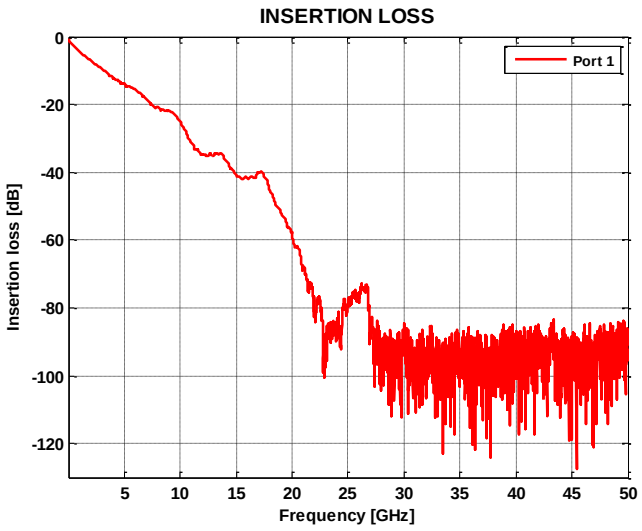


Figure 4-4. (a). Backplane Channel and (b) channel response

4.5 Receive Equalizer

Sometimes in addition to the transmit equalization, a receive equalizer is required at the receiver end. A DFE utilizing the LMS algorithm is used in the simulation of the design. It has two feedforward taps and one feedback tap with a step size of 0.1.

4.6 Duobinary Decoder

At the receiver, it is necessary to recover the original binary signal from the duobinary signal. The duobinary decoder is used for that purpose and it comprises two comparators, two threshold voltage levels and an XNOR gate [1]. The duobinary decoder is shown in

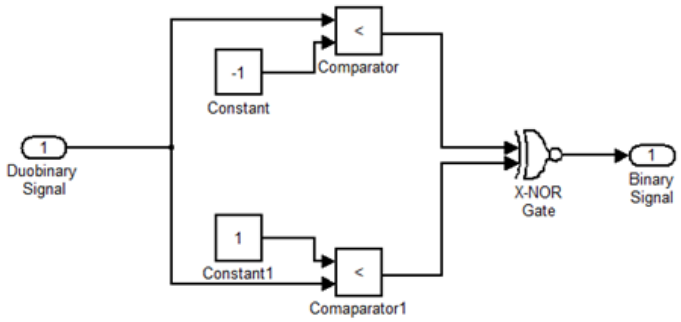


Figure 4-5. Duobinary Decoder

The high speed duobinary signal is fed into both comparators and compared with the given thresholds. The resulting outputs are then passed on to the XNOR gate. When the duobinary input is greater than both reference levels at the comparators, the output is a 1. The output is a 0 when the duobinary input is greater than the reference at “Comparator" and less than that at "Comparator 1". Finally, the output is a 1 when the duobinary input is less than the reference levels at both comparator inputs. Table 4-3 summarizes these results in a truth table.

Table 4-3. Truth table showing result of duobinary decoding

	Comparator	Comparator 1	Output
V>V2	0	0	1
V1<V<V2	0	1	0
V<V1	1	1	1

4.7 Model Simulation

This section first discusses the model simulation without the channel and equalizers. The next simulation is done with the channel and transmit filter only and the final simulation is done for the complete system including the transmitter, channel and receiver.

4.7.1 Without Channel

Figure 4-6, illustrates the complete system model without the channel and equalizers present. This is simulated to verify the correct operation of the duobinary encoder/ decoder system without the channel included. It comprises of an 11.3 Gbps Bernoulli binary generator followed by a duobinary precoder. Next, it is the unipolar to bipolar converter and the duobinary decoder. At the receiving end is the duobinary decoder which basically consists of comparators and an XNOR gate.

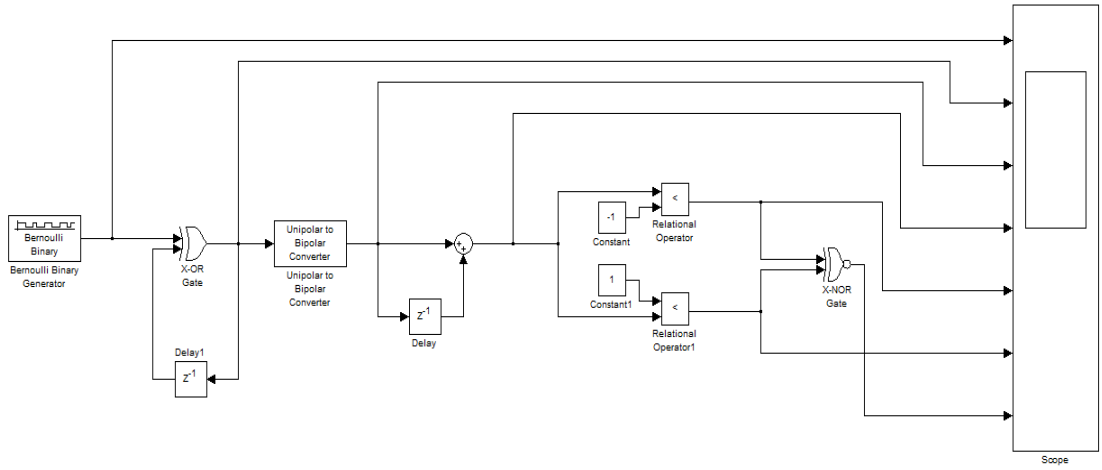


Figure 4-6. System model without channel.

Figure 4-7 shows the simulation results. The signal in the first row is the 11.3 Gbps NRZ signal generated by the Bernoulli Random Number Generator, while the second indicates the delayed NRZ signal. Moving on to the third row is the precoded binary sequence followed by the bipolar signal in the fourth row. In the fifth row is the three-level duobinary signal. In the last row is the recovered binary signal which is the same as the initially transmitted NRZ sequence in the first row. This indicates proper decoding of the transmitted signal, and; hence correct operation of the model.

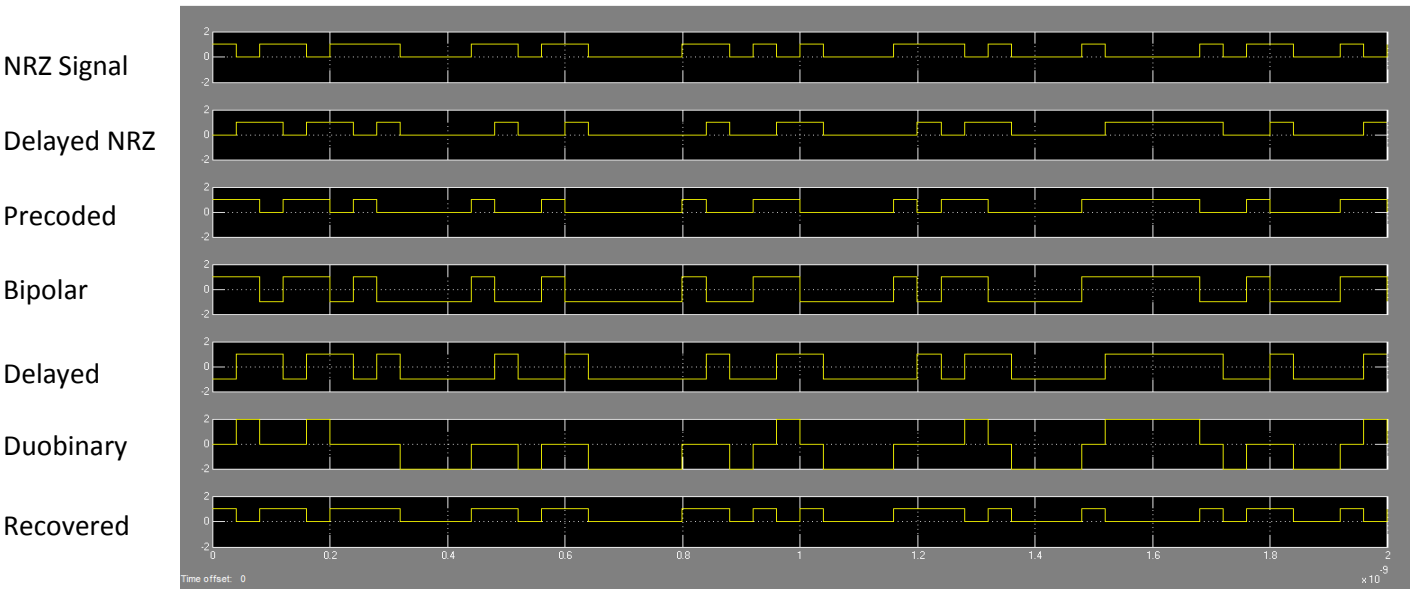


Figure 4-7. Simulation results without channel.

4.7.2 With Channel

Here, all other components of the design such as the channel and the equalizers are included to complete the model. A simulation is then performed and this time, eye diagrams instead of time scopes are used to view the results. Just as in Section 4.7.1, the binary source, precoder, and duobinary precoder are present. However, in addition to that are the channel, and equalizers as well as a couple of other supporting blocks. The complete system with the channel and equalizers is as depicted in Figure 4-8.

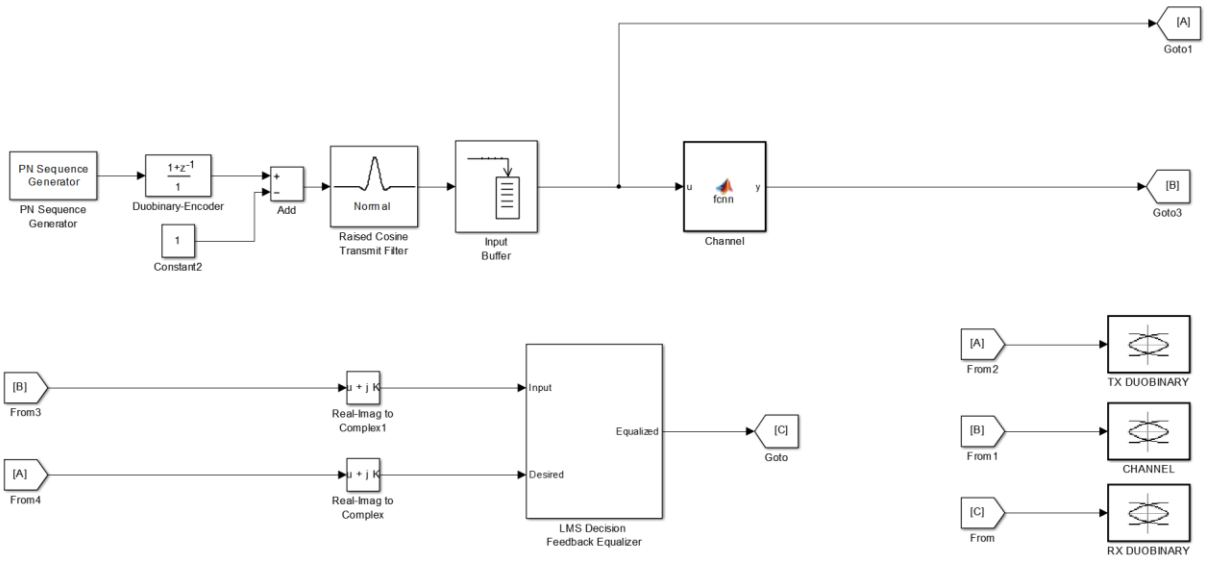
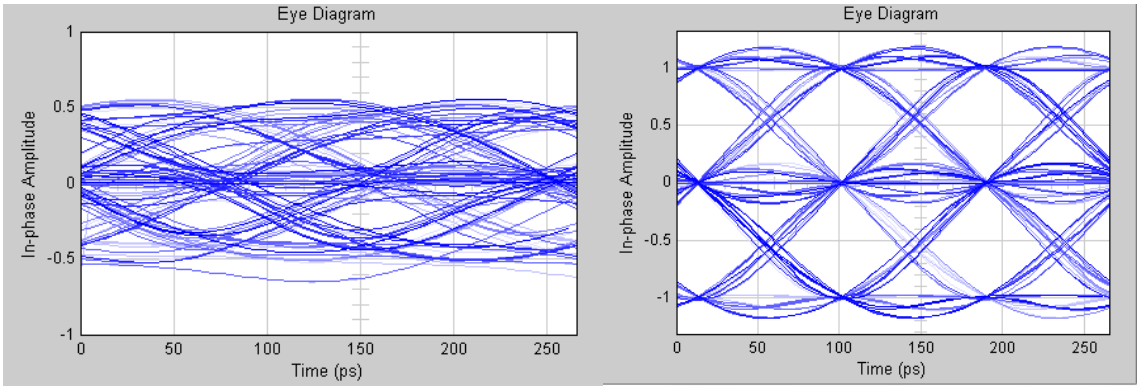


Figure 4-8. Complete duobinary transmitter model including channel.

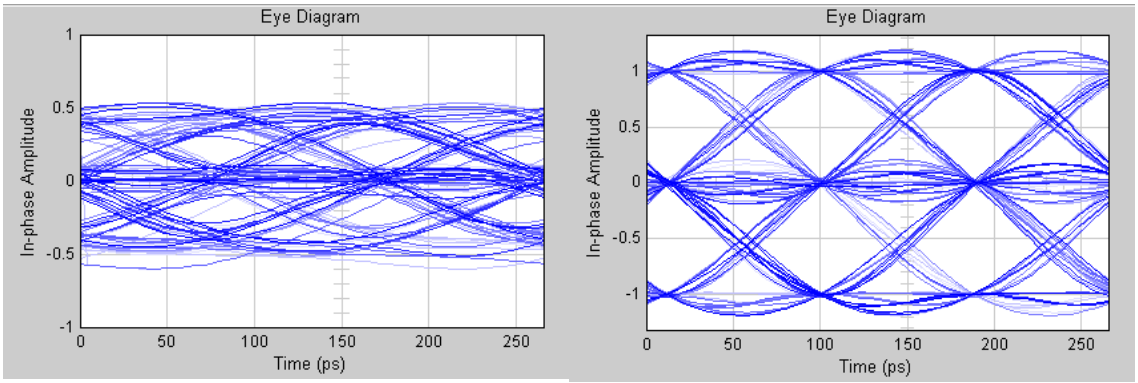
The data source used for this simulation is a PN sequence generator that generates a PRBS 7 pattern at the desired data rate. The pattern is then passed to the duobinary decoder where it is converted from a two –level NRZ to a three-level duobinary signal. The raised cosine filter with an excess bandwidth of 0.6 and an upsampling factor of 20 is used to shape the duobinary pulse before transmission to the channel. The input buffer converts sample based signal to a frame based signal at a rate of 5000 samples a frame. This is because the number of samples in the S-parameters of the channel is 5000. The MATLAB function block simulates the given channel. Within this block, the Fourier Transform of the samples is computed, after which it is multiplied by the channel frequency response. The result is then converted back to the time domain by taking the Inverse Fast Fourier Transform (IFFT). Equalization is next performed using the receive equalizer. Comparing the resulting eye diagrams before and after the equalizer, shows a

remarkable improvement. After passing through the channel, the duobinary eye is severely attenuated but after equalization, the three level duobinary signal comes out clean. These results are as shown in Figure 4-9 for the Megtron 6 caltrace board and Figure 4-10 for the FCI backplane.



(a) Before Equalization (b) After Equalization

Figure 4-9. Eye diagrams for transmitter simulation with Megtron 6 board at 11.3 Gbps



(a) Before Equalization (b) After Equalization

Figure 4-10. Eye diagrams for transmitter simulation with FCI backplane at 11.3 Gbps

4.8 Proposed ASIC Implementation

Because the FPGA is inherently a binary device, implementing a three-level signal is not feasible at the moment. The author tried to have access to the internal fabric of the transceivers from Altera but it was not possible. Therefore, an ASIC implementation of the duobinary encoder/decoder section is proposed with the FPGA still serving as the device for data generation, checking and channel equalization. The ASIC would be placed on the FPGA board such that when the NRZ data is transmitted from the FPGA chip and it passes from the transmitter PCS and PMA, it goes to the duobinary encoder on the board before getting transmitted on the channel. At the receiver, the duobinary signal is converted back to NRZ before CDR takes place. A block diagram for the complete proposed architecture is shown in Figure 4-11.

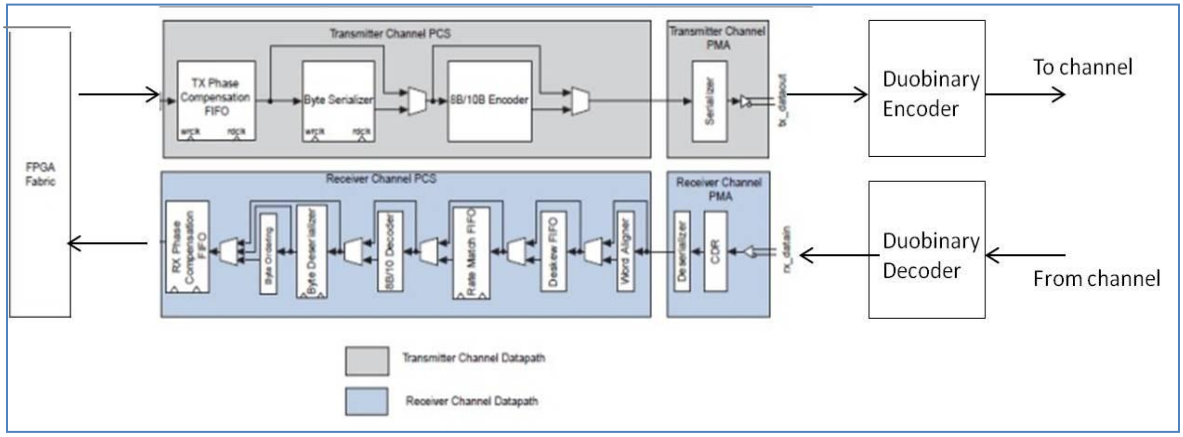


Figure 4-11. Proposed ASIC implementation of FPGA transceiver

The above architecture offers several advantages. The primary one being that the whole structure of the FPGA transceivers is not distorted. All NRZ operations can still be carried out as usual. Secondly, other line coding methods or operations can also be easily integrated into the transceivers. Subsequent sections demonstrate how HDL code (not synthesizable at the moment for reasons discussed at the beginning of this sub-section) for the system is generated and also the proposed architecture in Simulink. Also, simulation of the architecture in Simulink is shown.

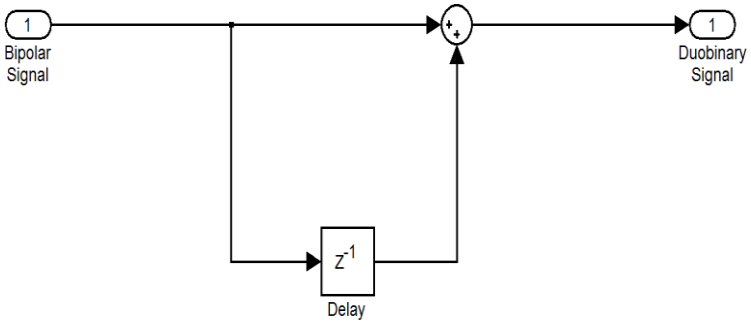


Figure 4-12. Duobinary encoder architecture for ASIC implementation

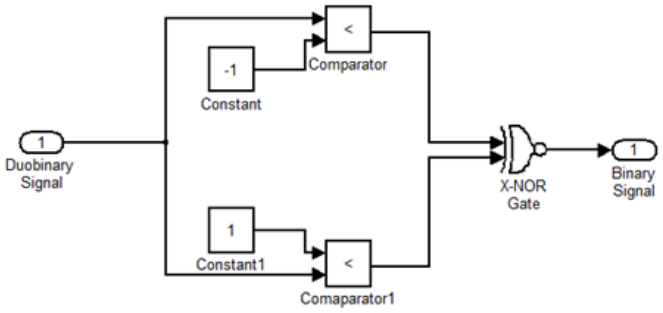


Figure 4-13. Duobinary decoder architecture for ASIC implementation

4.9 HDL Code generation for duobinary encoder/decoder

The HDL coder from within Simulink is used to generate the HDL code required to implement the duobinary encoder/decoder. It is started from the Simulation tab of the menu bar in simulink. The settings window is shown in Figure 4-14.

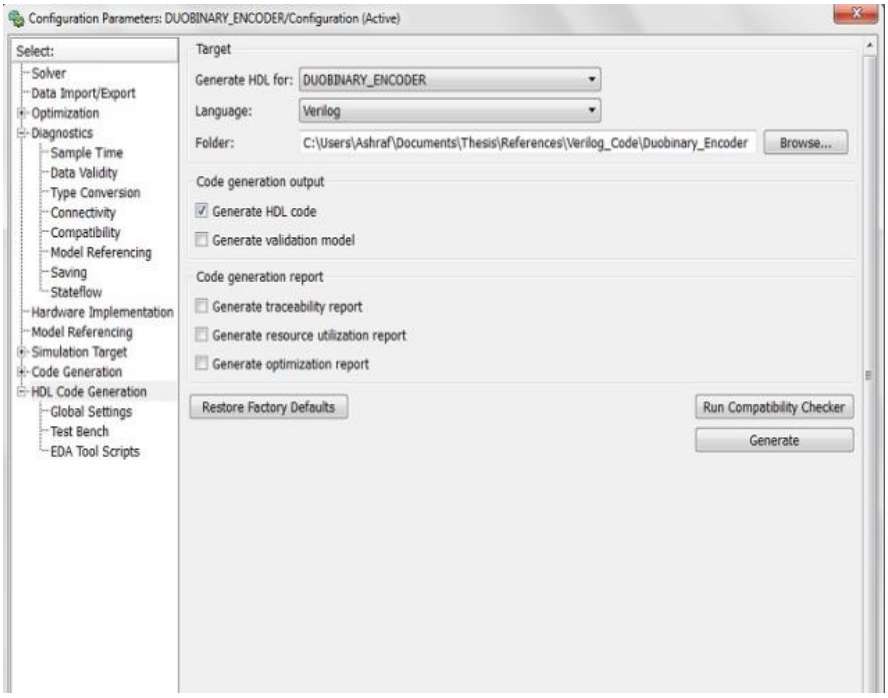


Figure 4-14. Settings window for generating HDL code for duobinary encoder

In Figure 4-15, we see a successful HDL code generation for the duobinary encoder while Figure 4-16 shows the architecture connection in Quartus. The duobinary encoder/decoder blocks were created in Quartus from the code generated by Simulink.

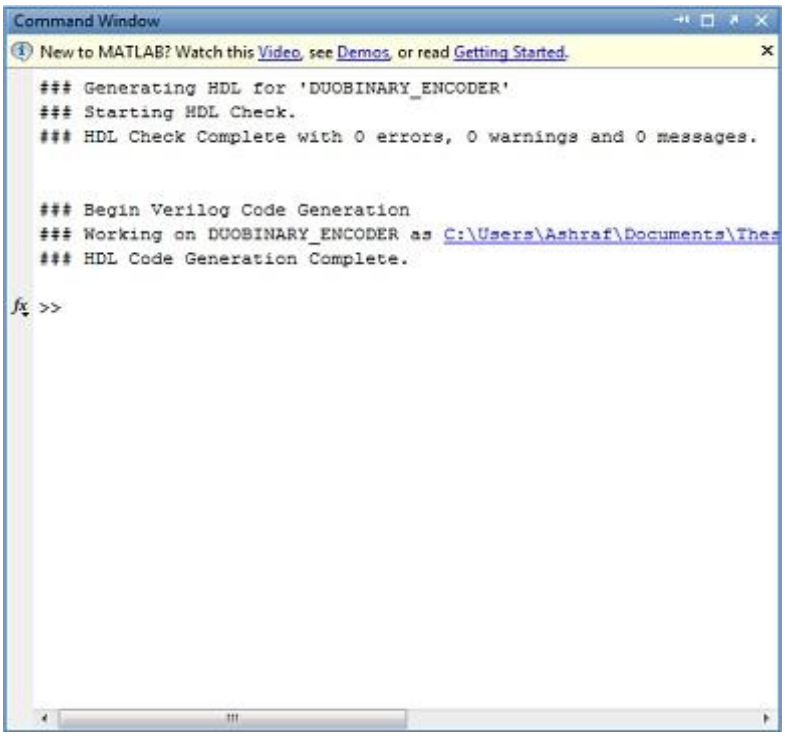


Figure 4-15. Successful HDL code generation

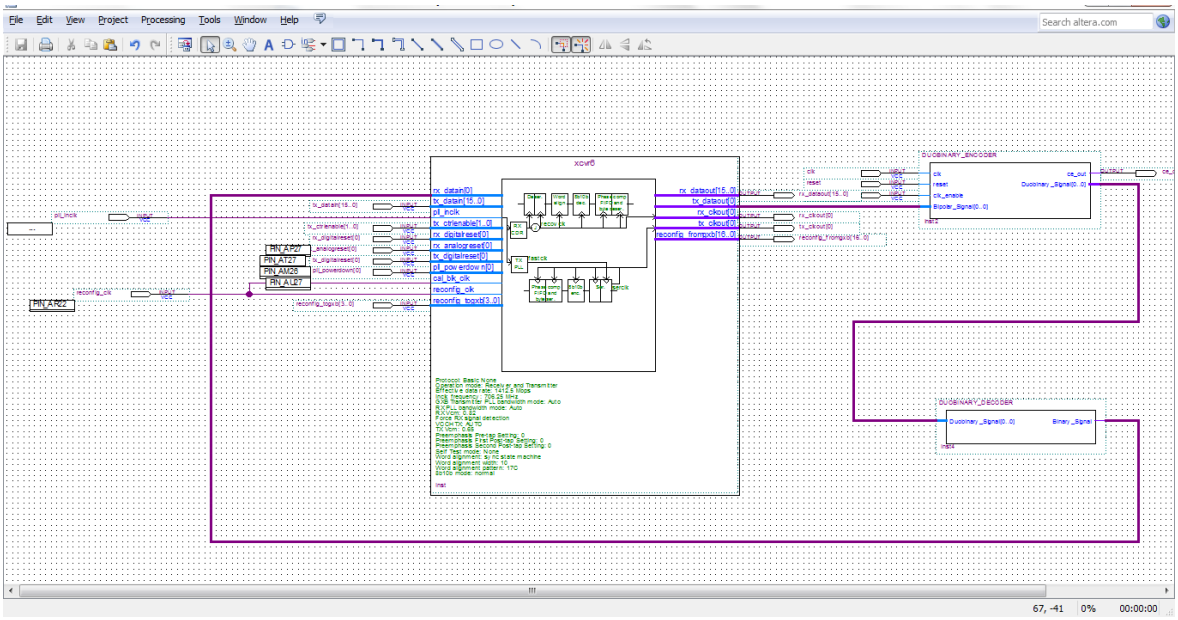


Figure 4-16. Proposed Architecture in Quartus

4.10 Simulation of Proposed Architecture

This section has the simulation of the complete system from end to end including the transmitter section, the channel and the receiver. The model is shown in Fig 4.17

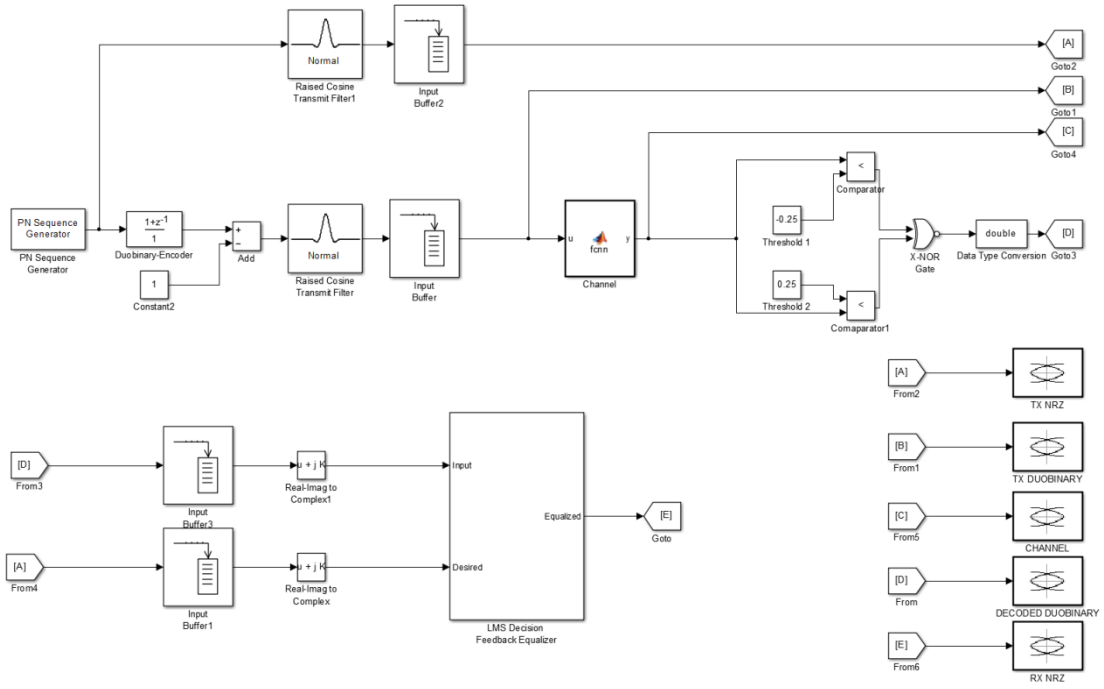
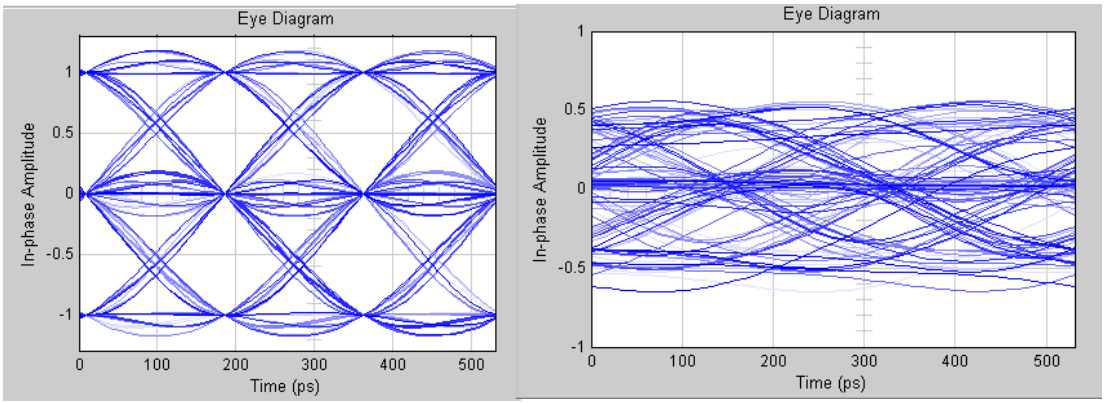


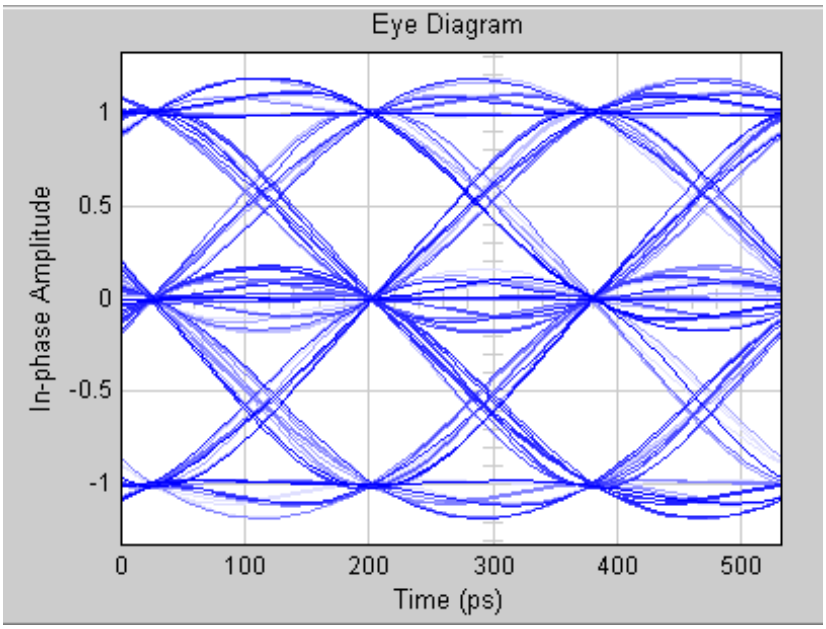
Figure 4-17. Proposed end to end architecture model.

The performance of the duobinary system depends on the threshold values of the decoder. One approach would be to pick these values at simulation time and then use them value for hardware implementation. The simulation results shown in Figure 4-18(b) and Figure 4-19(b) indicate that the duobinary signal is attenuated to about half its magnitude. This means the signal level has dropped to between -0.5V and 0.5V. Sampling is normally done in the middle of the eye which corresponds to 0.25V and -0.25V in this case. For that reason, the threshold levels of the duobinary decoder are set to -0.25V and 0.25V,

respectively. Lower values of -0.1V and 0.1V can also be used. If however, the channel attenuation is not known, the optimum threshold values might have to be determined by an adaptive algorithm. The system is simulated and the resulting eye diagrams into and out of both channels are compared. It is observed that the eye diagrams for the FCI backplane suffer more attenuation because of the longer trace length and also the effect of the connectors. Notwithstanding the attenuation; however, the duobinary decoder successfully decodes the eye back to binary. Eye diagrams at the transmitter closely resemble those at the receiver for both channels. Figure 4-18 and Figure 4-19 show these results.

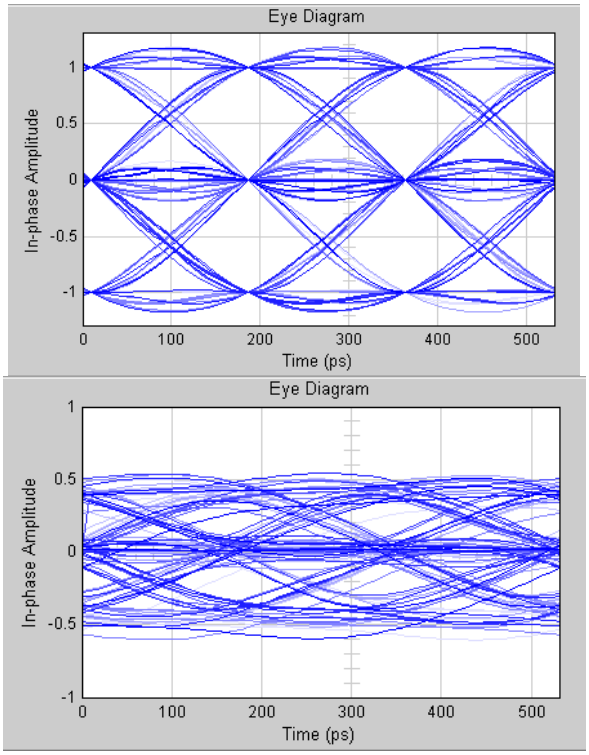


(a)Before Channel (b) After channel

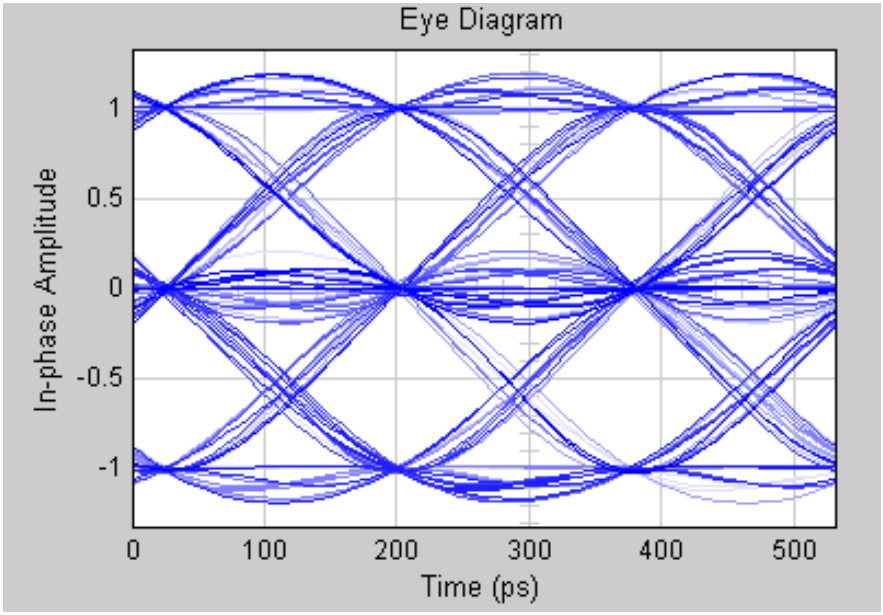


(c)After equalization

Figure 4-18. Duobinary results for the Megtron 6 board at 5.65 Gbps



(a)Before Channel (b) After channel



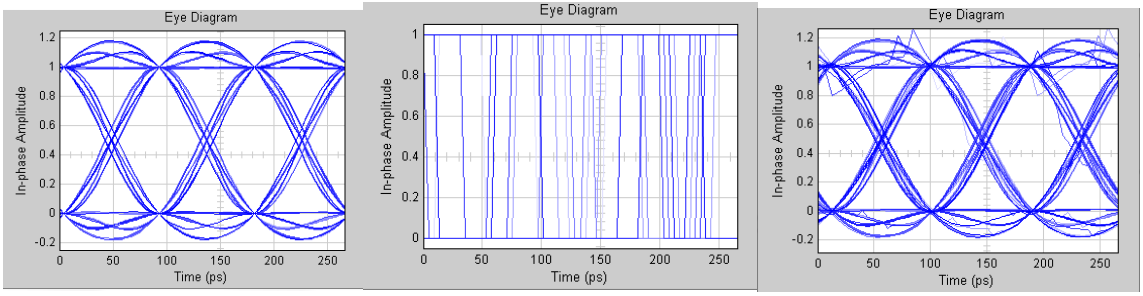
(c)After equalization

Figure 4-19. Duobinary results for the FCI backplane at 5.65 Gbps

Notice that there is improvement of the eye opening. Details will be explained in section 4.12.

4.11 Duobinary decoding with channel

Figure 4-17 shows a complete duobinary system which includes the encoder, channel and decoder. This section views the original NRZ eye diagram transmitted to the duobinary encoder and through the channel and then compares it to the eye diagram at the receiver without and with receive equalization. Figure 4-20 illustrates these results.



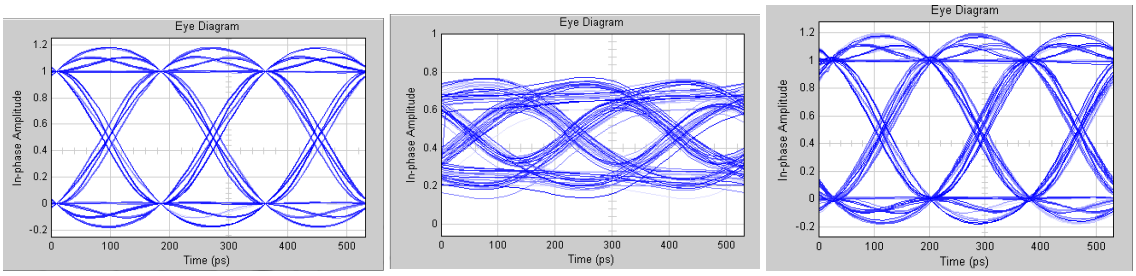
(a) Transmitted NRZ (b) Decoded NRZ (c) Received NRZ

Figure 4-20. Significance of receive equalization

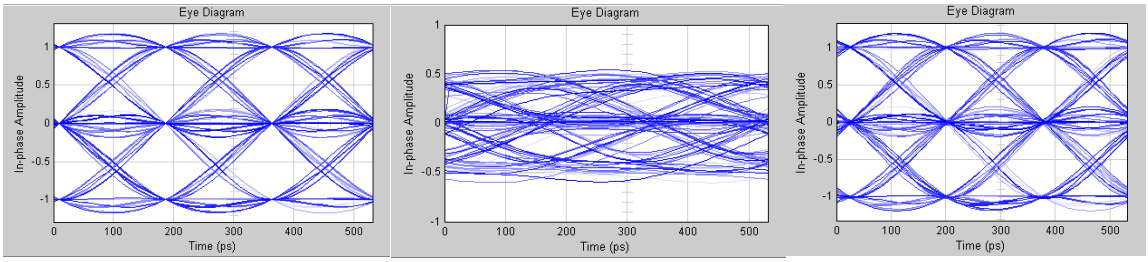
Figure 4-20 (a) shows the eye pattern transmitted to the channel while Figure 4-20 (b) shows the eye pattern for the decoded duobinary signal before it is equalized by the DFE. It is seen that even though it is a two level NRZ signal, it is not neat and can lead to errors in signal detection. Figure 4-20 (c) shows a decoded and equalized signal. This is a clear signal. It is therefore important to have the equalizer to clean up the signal after it must have been corrupted by the channel impairments and noise.

4.12 Comparison of NRZ and Duobinary Simulation Results

This section compares the eye diagram results for both NRZ and duobinary signaling in terms of the eye diagrams obtained. No settings were changed in moving from NRZ to duobinary except for the inclusion of the duobinary encoder and duobinary decoder blocks. Results for both channels are viewed with data rates of 5.65 Gbps and 11.3 Gbps.

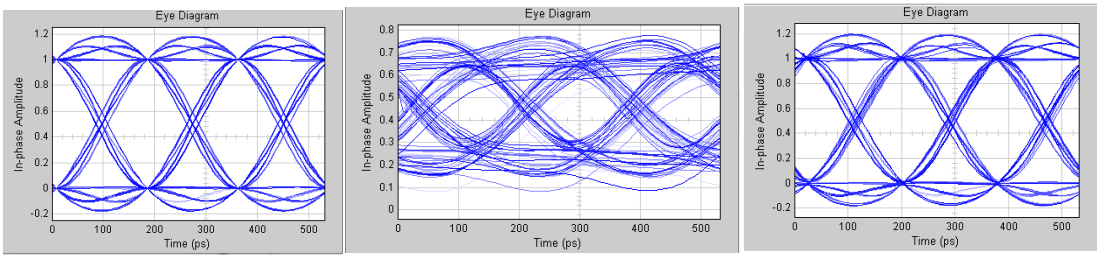


(a) Transmitted NRZ (b) NRZ through channel (c) Received NRZ

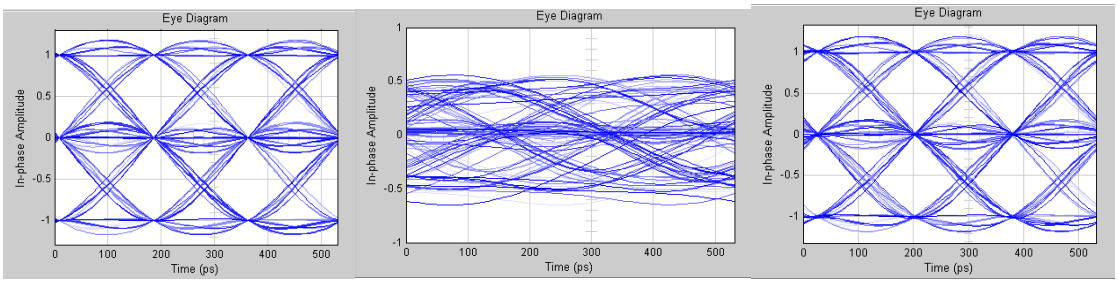


(a) Transmitted duobinary (b) Duobinary through channel (c) Received duobinary

Figure 4-21. NRZ and duobinary comparison, FCI backplane at 5.65 Gbps

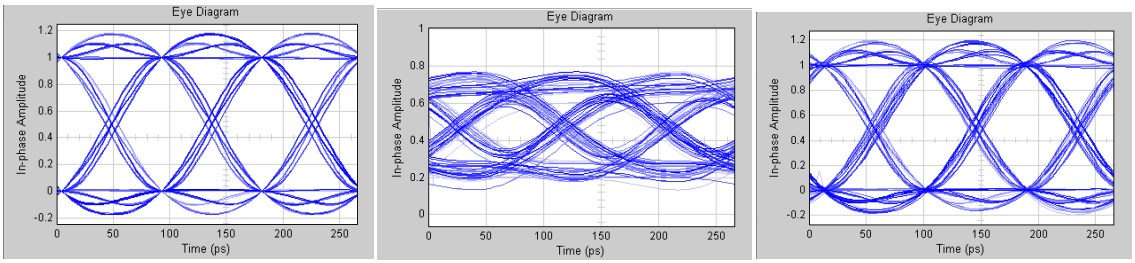


(a) Transmitted NRZ (b) NRZ through channel (c) Received NRZ

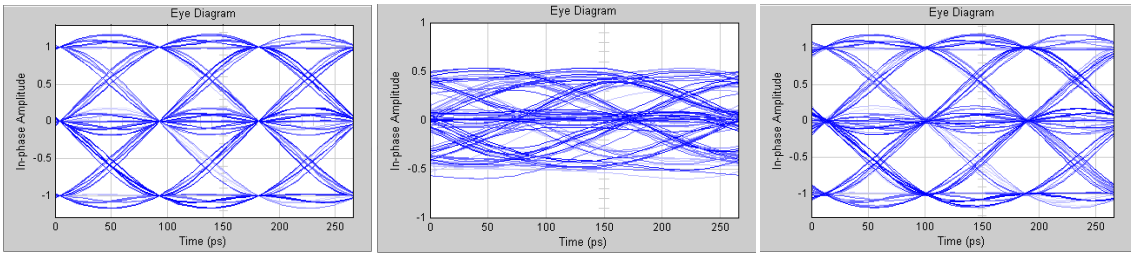


(a) Transmitted duobinary (b) Duobinary through channel (c) Received duobinary

Figure 4-22. NRZ and duobinary comparison, megtron-6 board at 5.65 Gbps

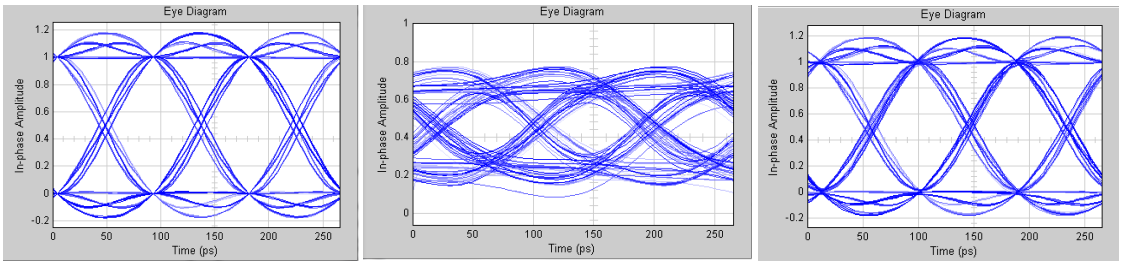


(a) Transmitted NRZ (b) NRZ through channel (c) Received NRZ

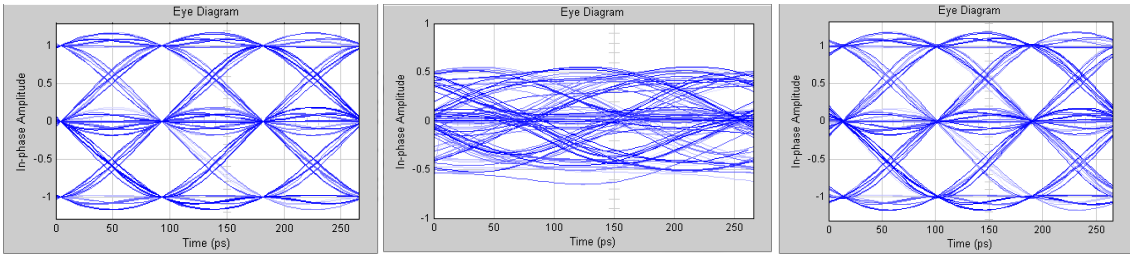


(a) Transmitted duobinary (b) Duobinary through channel (c) Received duobinary

Figure 4-23. NRZ and duobinary comparison, FCI backplane at 11.3 Gbps.



(a) Transmitted NRZ (b) NRZ through channel (c) Received NRZ



(a) Transmitted duobinary (b) Duobinary through channel (c) Received duobinary

Figure 4-24. NRZ and duobinary comparison, megtron-6 board at 11.3 Gbps

Table 4-4. Comparison simulation results at 5.65 Gbps

	Megtron 6 board		FCI backplane	
	NRZ	Duobinary	NRZ	Duobinary
Eye height (V)	0.3	0.45	0.3	0.4
Eye width (ps)	130	140	130	130

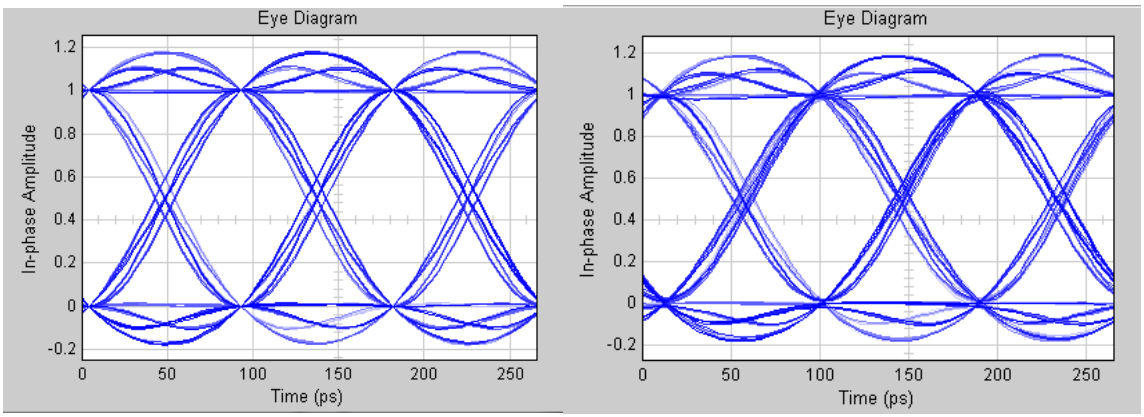
Table 4-5. Comparison simulation results at 11.3 Gbps

	Megtron 6 board		FCI backplane	
	NRZ	Duobinary	NRZ	Duobinary
Eye height (V)	0.3	0.3	0.25	0.4
Eye width (ps)	60	65	60	65

Table 4.4 and 4.5 summarize the results in Figures 4-21 to 4-24. A careful observation indicates that in general, the duobinary signal has a better eye height and eye width than the NRZ for both channels and data rates and as such a more suitable choice when moving up to higher data transmission rates. The duobinary scheme has a 30% improvement over the NRZ scheme.

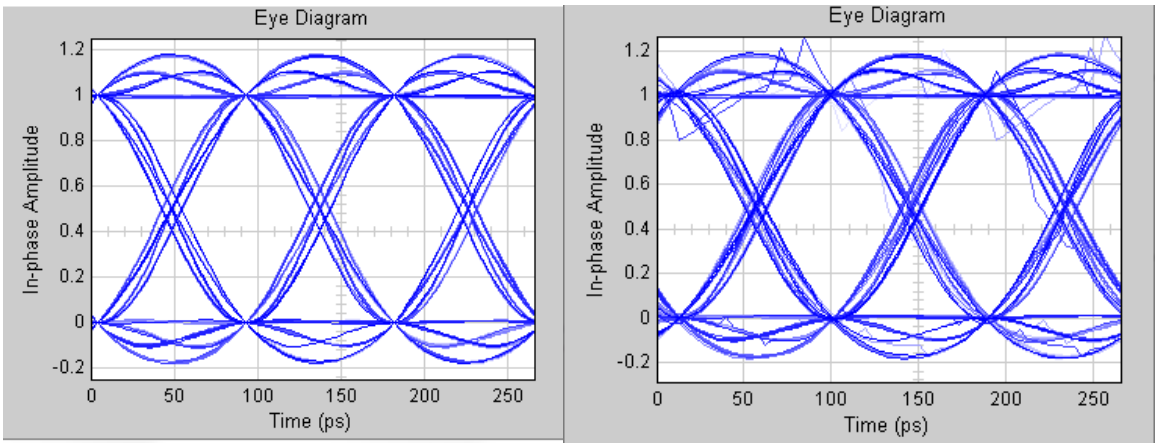
4.13 End-to-End NRZ and duobinary comparison

This section compares the recovered eye diagrams for both the NRZ and duobinary systems. The eye diagrams for both systems are identical except for some little noise in the duobinary system. This implies correct operation of the duobinary encoder and decoder subsystems. Figure 4-25 shows the transmitted and received NRZ eye in the NRZ scheme while Figure 4-26 shows the transmitted and recovered NRZ eye transmitted in the duobinary system.



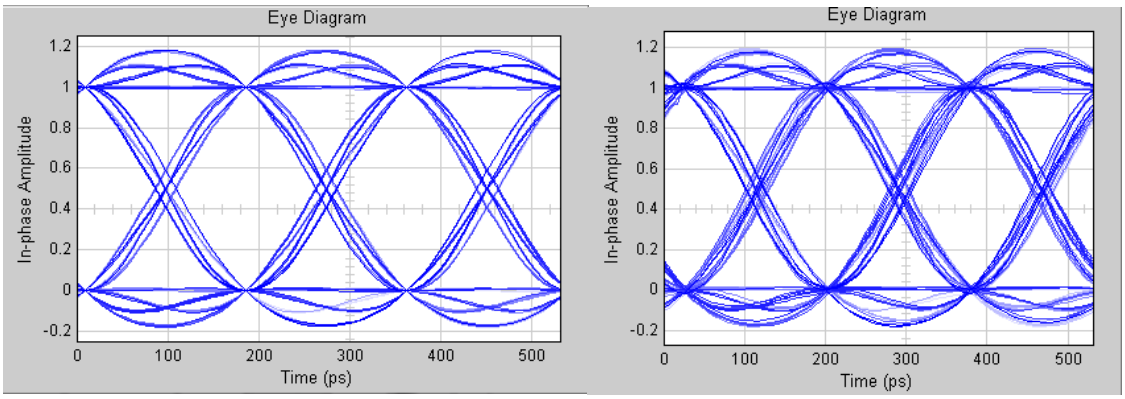
(a) Transmitted eye (b) Recovered eye

Figure 4-25. NRZ system for 11.3 Gbps for backplane



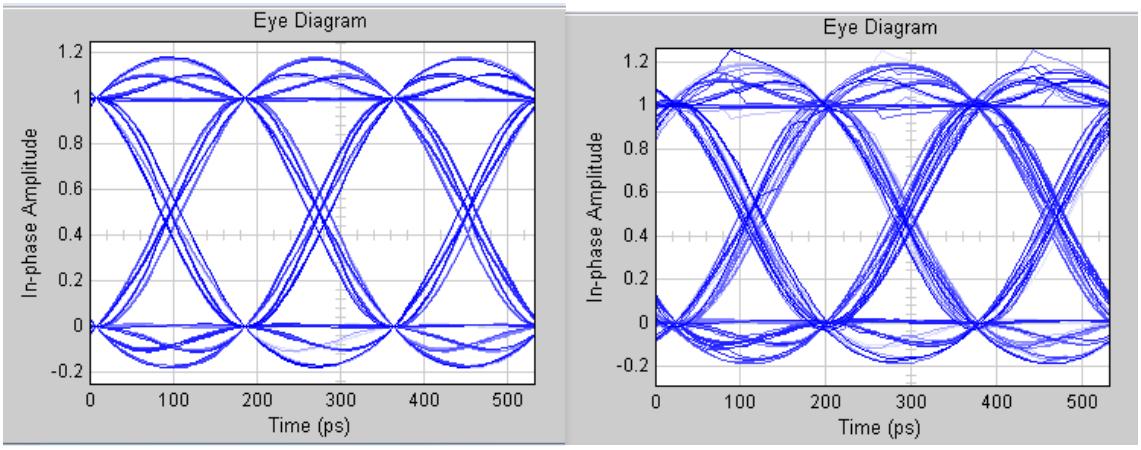
(a) Transmitted eye (b) Recovered eye

Figure 4-26. Duobinary system for 11.3 Gbps for backplane



(a) Transmitted eye (b) Recovered eye

Figure 4-27. NRZ system at 5.65 Gbps for backplane



(a) Transmitted eye (b) Recovered eye

Figure 4-28. Duobinary system at 5.65 Gbps for backplane

4.14 Bit Error Rate Results

In this section, the BER for NRZ and duobinary signaling of the architecture running at 5.65 Gbps for both channels are obtained and compared using Simulink. The model is the same as in previous sections except for the addition of the "Align Signals" block which is included to compensate for delay the signal encounters as it propagates through the channel and the "Error Calculation" block which computes the BER. "Rounding Function" blocks are also included within the model to reduce errors in the system because Simulink only compares exact values, if they are off even one decimal place, an error results. The simulink model for the NRZ BER calculation is as shown in Figure 4-20 while the model for the duobinary BER calculation is shown in Figure 4-21.

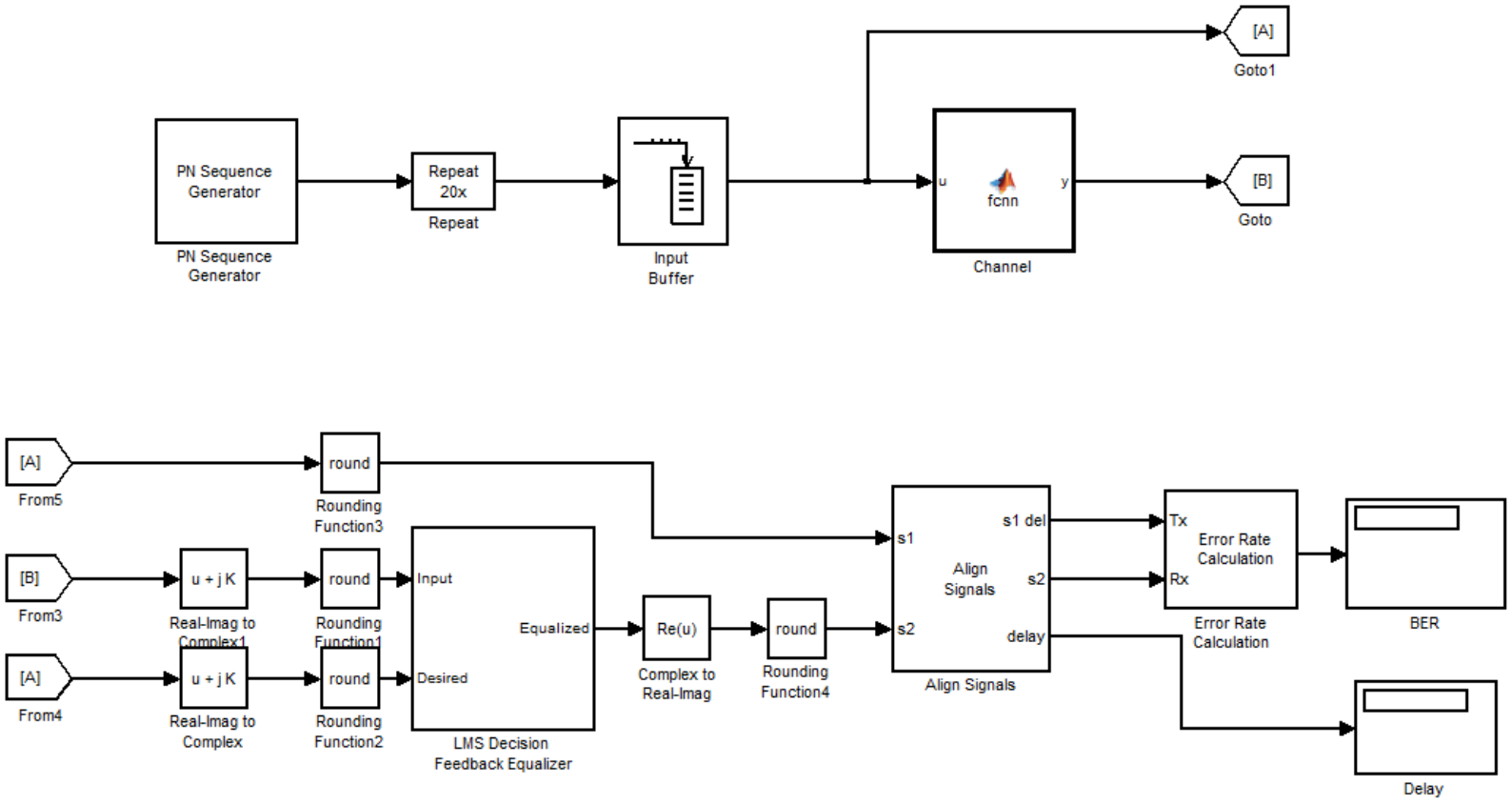


Figure 4-29. Simulink model for NRZ BER calculation

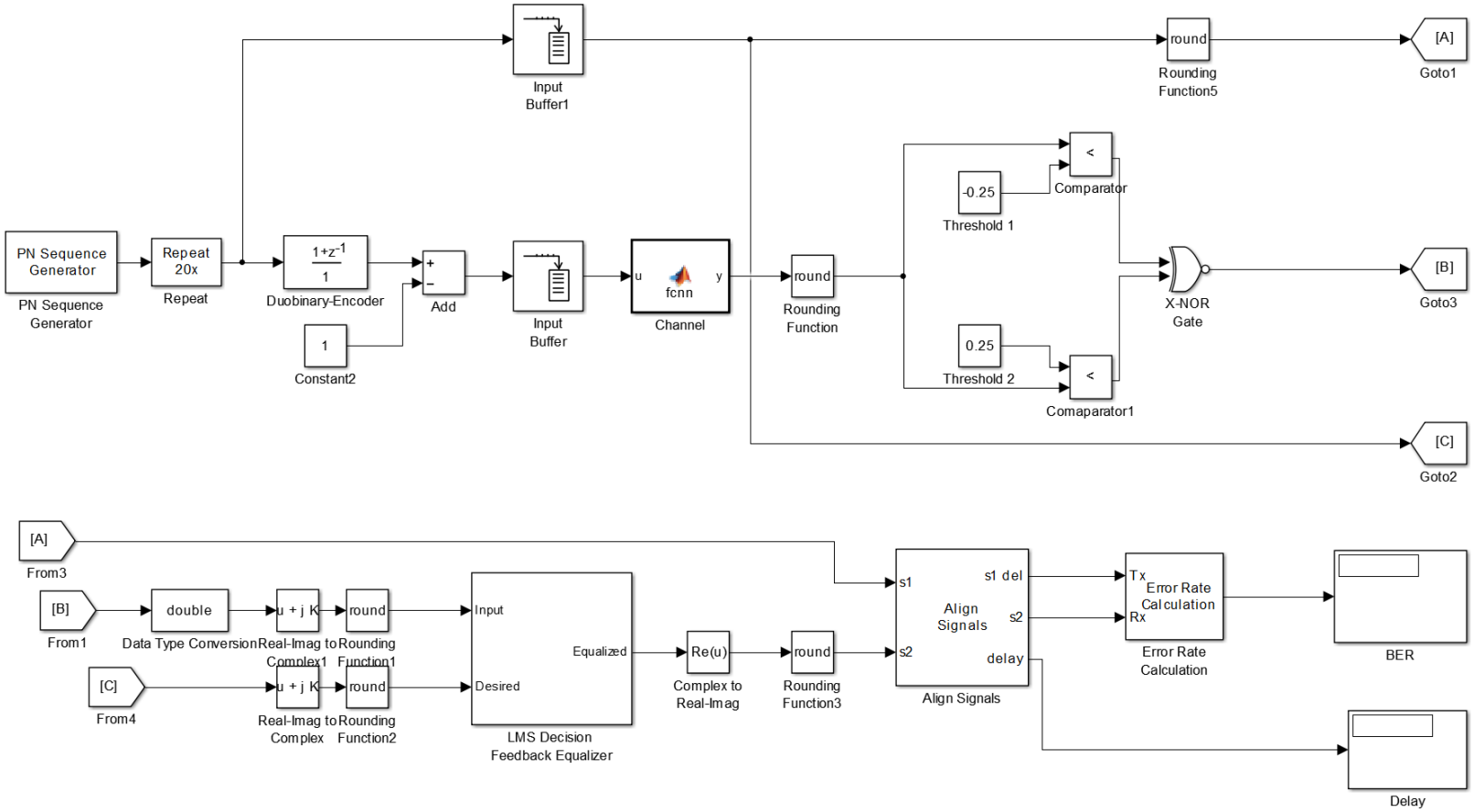


Figure 4-30. Simulink model for Duobinary BER calculation

The BER for the megtron-6 caltrace board at a data rate of 5.65 Gbps is 4.082 $x10^{-6}$ when NRZ signaling is used and 3.675 $x10^{-6}$ for duobinary. The BER of the backplane using NRZ, however, is 5.351 $x10^{-6}$ while it is 4.72 $x10^{-6}$ for duobinary signaling. The BER for the backplane is worse than the BER of the megtron-6 caltrace because of the connectors and daughter cards present on the backplane. In both cases, it is seen that the duobinary BER is better than that of the NRZ. The BERs are relatively high and this can be attributed to truncations, padding and real to complex operations that take place during FFT and IFFT operations performed within the MATLAB function block. In addition, care must be taken with the LMS block since the convergence rate contributes to error rate at the beginning of the simulation while the filter taps are adjusting. The BER improves as the simulation progresses, and therefore, a better BER is expected if the simulation time is increased. Table 4-6 shows these results BER results at 5.65 Gbps while Table 4-7 shows BER results at 11.3 Gbps.

Table 4-6. BER results at 5.65 Gbps

	Megtron 6 board		FCI backplane	
Signaling	NRZ	Duobinary	NRZ	Duobinary
BER($x10^{-6}$)	4.082	3.675	5.351	4.72

Table 4-7. BER results at 11.3 Gbps

	Megtron 6 board		FCI backplane	
Signaling	NRZ	Duobinary	NRZ	Duobinary
BER($x10^{-6}$)	4.4	3.139	6.854	3.902

Chapter 5

Conclusions and Recommendations for Future Studies

5.1 Conclusions

This thesis demonstrates the ability to model and run a complete high speed transceiver system on an FPGA board using the FPGAs using the NRZ line coding scheme. Eye diagram results were used to compare the NRZ simulation and measurement results at data rates of 5.65 Gbps and 11.3 Gbps using two different channels. The transceivers in the FPGA were also run to demonstrate the equalization features of the FPGA for both channels and data rates. Simulation and measurement results matched indicating a correct simulation. A proposed duobinary ASIC scheme for integration with the FPGA transceivers was designed and simulated in Simulink. Two different channels were used at data rates of 5.65 and 11.3 Gbps. The transmitted duobinary eye was correctly recovered at the receiver and the eye heights and widths were better than those for the system. The BER for both systems was also compared and it was found out that the duobinary system is better than the NRZ hence it is a good choice for high data rates. FPGAs were chosen for this research work because it is easy to make design changes and also because of the FPGA's built-in features such as equalizer and transceivers. Through a feature called dynamic reconfiguration, both transmit and receive equalization can be performed on the transceiver links as the device is running.

5.2 Recommendations for Further Study

It is expected that the work done in this thesis will allow for quick and easy implementation of the proposed architecture as major issues such as data generation, equalization etc have been addressed. The next step will be implementing this architecture for duobinary as well as for PAM-4. If carried out, it will go a long way in simplifying high speed analysis of digital transmission systems as all needed components are integrated on a single board. Boards that support higher speeds and more equalization features such as the Stratix V GT FPGAs should be used for future work. Higher speeds of up to 40 Gbps can also be studied and explored by operating the transceivers in bonded mode. Comparative studies can also be carried out between hardware implementation of NRZ, duobinary and PAM-4 signaling schemes. More accurate BER methods for simulation could also be explored as the FFT, IFFT, and LMS operations performed in the simulation cause rounding errors and worsen the BER performance.

Appendix

/* Matlab script for extracting s-parameters from measurement*/

```
a = read(rfdata.data,'LY4 29inch All 10mil.s4p');

[SP,freq] = extract(a,'S_Parameters',50);

thru = SP(1,2,:);

thru = squeeze(thru);
```

/* Matlab script for characterizing channel behavior*/

```
function y = fcnn(u)

y=ones(5000,1);
eml.extrinsic('importdata')
eml.extrinsic('fft')
eml.extrinsic('ifft')
eml.extrinsic('times')


s12=importdata('thru.mat');

input_freq_domain=fft(u,5000);
u_freq=input_freq_domain;

channel_sim=times(u_freq,s12);
output=ifft(channel_sim);

y = real(output);
```

/* Generated Verilog code for duobinary encoder*/

```
// -------------------------------------------------------------
//
// File Name:
C:\Users\Ashraf\Documents\Thesis\References\Verilog_Code\Duobinary_Enco
der\DUOBINARY_ENCODER.v
// Created: 2014-02-12 09:11:13
//
// Generated by MATLAB 7.14 and HDL Coder 3.0
//
```

```
//
// -- -----------------------------------------------------------
// -- Rate and Clocking Details
// -- -----------------------------------------------------------
// Model base rate: 4e+07
// Target subsystem base rate: 4e+07
//
//
// Clock Enable  Sample Time
// -- -----------------------------------------------------------
// ce_out        4e+07
// -- -----------------------------------------------------------
//
//
// Output Signal                 Clock Enable  Sample Time
// -- -----------------------------------------------------------
// Duobinary_Signal              ce_out        4e+07
// -- -----------------------------------------------------------
//
// -------------------------------------------------------------


// -------------------------------------------------------------
//
// Module: DUOBINARY_ENCODER
// Source Path: DUOBINARY_ENCODER
// Hierarchy Level: 0
//
// -------------------------------------------------------------

`timescale 1 ns / 1 ns

module DUOBINARY_ENCODER
          (
           clk,
           reset,
           clk_enable,
           Bipolar_Signal,
           ce_out,
              Sum_out1,
           Duobinary_Signal
          );


  input   clk;
  input   reset;
  input   clk_enable;
  input   [0:0] Bipolar_Signal;  // double
  output  ce_out;
  output  [0:0] Duobinary_Signal;  // double
  output  Sum_out1;

  wire enb;
```

```
  wire Bipolar_Signal_double;  // double
  wire alphaDelay_out1;  // double
  wire Sum_out1;  // double


  //always @* Bipolar_Signal_double = $bitstoreal(Bipolar_Signal);

  assign enb = clk_enable;

 // always @(posedge clk or posedge reset)
  //  begin : alphaDelay_process
  //    if (reset == 1'b1) begin
   //     alphaDelay_out1 <= 0.0;
   //   end
   //   else begin
    //    if (enb) begin
     //     alphaDelay_out1 <= Bipolar_Signal_double;
     //    end
   ////   end
   // end



  assign Sum_out1 = Bipolar_Signal_double + alphaDelay_out1;



  //assign Duobinary_Signal = $realtobits(Sum_out1);

  assign ce_out = clk_enable;

endmodule  // DUOBINARY_ENCODER
```

/* Generated Verilog code for duobinary decoder*/

```
// -------------------------------------------------------------
//
// File Name:
C:\Users\Ashraf\Documents\Thesis\References\Verilog_Code\Duobinary_Deco
der\DUOBINARY_DECODER.v
// Created: 2013-09-29 13:04:47
//
// Generated by MATLAB 7.12 and Simulink HDL Coder 2.1
//
//
// -- -------------------------------------------------------------
// -- Rate and Clocking Details
// -- -------------------------------------------------------------
// Model base rate: 4e-011
// Target subsystem base rate: 4e-011
```

```
//
// -------------------------------------------------------------


// -------------------------------------------------------------
//
// Module: DUOBINARY_DECODER
// Source Path: DUOBINARY_DECODER
// Hierarchy Level: 0
//
// -------------------------------------------------------------

`timescale 1 ns / 1 ns

module DUOBINARY_DECODER
          (
           Duobinary_Signal,
           Binary_Signal
          );


  input   [0:0] Duobinary_Signal;  // double
  output  Binary_Signal;


  real Duobinary_Signal_double;  // double
  real Constant_out1;  // double
  wire Comparator_relop1;
  real Constant1_out1;  // double
  wire Comaparator1_relop1;
  wire Comparator_out1;


  always @* Duobinary_Signal_double = $bitstoreal(Duobinary_Signal);

  initial Constant_out1 = -1.0;



  assign Comparator_relop1 = (Duobinary_Signal_double < Constant_out1 ?
1'b1 :
              1'b0);


  initial Constant1_out1 = 1.0;


  assign Comaparator1_relop1 = (Constant1_out1 <
Duobinary_Signal_double ? 1'b1 :
              1'b0);
```

```
  assign Comparator_out1 =  ~ (Comparator_relop1 ^
Comaparator1_relop1);



  assign Binary_Signal = Comparator_out1;

endmodule  // DUOBINARY_DECODER
```

/* Top level Verilog Code for Transceiver */

```
module gx_link_test_example
(
    S4GX_50M_CLK4P,
    REFCLK3P_GXB1_706M25,
    GXB1_RX0,
    GXB1_TX0,
);

    input  S4GX_50M_CLK4P;
    input  REFCLK3P_GXB1_706M25;
    input  GXB1_RX0;
    output GXB1_TX0;

    wire clk_50Mhz;
    wire pll_inclk;

    assign clk_50Mhz = S4GX_50M_CLK4P;
    assign pll_inclk = REFCLK3P_GXB1_706M25;



    link_test_sopc_sys_10g link_test_sopc_sys_10g_inst (
        .clk_clk_in_clk                                 (clk_50Mhz),
//          clk_clk_in.clk
        .clk_clk_in_reset_reset_n                       (system_reset),
//    clk_clk_in_reset.reset_n
        .pllclk_clk_in_clk                              (pll_inclk),
//       pllclk_clk_in.clk
        .pllclk_clk_in_reset_reset_n                    (system_reset),
// pllclk_clk_in_reset.reset_n
        .xcvr_low_latency_phy_0_tx_serial_data_export   (GXB1_TX0),
// xcvr_tx_serial_data.export
        .xcvr_low_latency_phy_0_rx_serial_data_export   (GXB1_RX0)  //
xcvr_rx_serial_data.export
    );
```

```
Endmodule
```

/* Synopsis Design Constraint file for clocking */

```
create_clock -period 50MHz [get_ports {S4GX_50M_CLK4P}]
create_clock -period 706.25MHz [get_ports {REFCLK3P_GXB1_706M25}]
derive_pll_clocks
derive_clock_uncertainty

set_input_delay -clock {altera_reserved_tck} 20 [get_ports
altera_reserved_tdi]
set_input_delay -clock {altera_reserved_tck} 20 [get_ports
altera_reserved_tms]
set_input_delay -clock {altera_reserved_tck} 20 [get_ports
altera_reserved_ntrst]
set_output_delay -clock {altera_reserved_tck} 20 [get_ports
altera_reserved_tdo]

set_clock_groups -asynchronous \
-group {S4GX_50M_CLK4P} \
-group {REFCLK3P_GXB1_706M25} \
-group [get_clocks {*|receive_pcs0|clkout }]\
-group [get_clocks {*|transmit_pcs0|clkout }]\
-group [get_clocks {*|receive_pma0|deserclock* }]\
-group [get_clocks {*|transmit_pma0|refclk0in* }]
```

References

[1] J.H. Sinsky, M. Duelk, A. Adamiecki, "High-Speed Electrical Backplane Transmission using Duobinary Signaling", *IEEE Transactions on Microwave Theory and Techniques*, Vol. 53, No. 1, pp. 152-160, January, 2005.

[2] Lakshmi P. Baskaran, "Duobinary Signaling Systems for High-Speed Data Transmission from 10 – 40 Gbps across Legacy Backplane Channels", Masters Paper, Pennsylvania State University, Harrisburg, September, 2006.

[3] A. Lender, "The Duobinary Technique for High-Speed Data Transmission", *IEEE Transactions on Communications and Electronics*, vol 82, pp. 214-218, May, 1963.

[4] J.H. Sinsky, A. Adamiecki, M. Duelk, H. Walter, H.J. Goetz, M. Mandich, "10 Gbps Duobinary Signaling over Electrical Backplanes, Experimental Results and Discussion", July, 2004.

[5] A. Adamiecki, J.H. Sinsky, M. Duelk, "25 Gbps Electrical Duobinary Transmission over FR-4 Backplanes", *Electronic Letters*, Vol. 41, No. 14, July, 2005.

[6] K. Yamaguchi et. al., "12 Gbps Duobinary Signaling with X2 Oversampled edge Equalization", *IEEE International Solid State Circuits Conference*, Session 3, pp. 70-71, February, 2005.

[7] J. D'Ambrosia, J. Khoury, M. Barazande-Pour, G. Koziuk, "802.3AP Backplane Ethernet", Lucent Technologies, 2004.

[8] "Extending Transceiver Leadership at 28 nm", White Paper, Altera Corporation, October, 2012.

[9] "Backplane Applications with 28 nm FPGAs", White Paper, Altera Corporation, December, 2012.

[10] Edin Kadrid, "An FPGA Implementation for a High-Speed Optical Link with a PCIe Interface", Master's Thesis, McGill University, Montreal, Canada, October, 2011.

[11] M.B. Muazu, I.A. Umar, "Design, Simulation and Development of a Field Programmable Gate Array (FPGA) Based Monitoring System", *International Conference on Artificial Intelligence*, Vol II, pp. 741-777, Las Vegas, USA, July, 2010.

[12] Stratix IV device handbook, "Transceiver Architecture in Stratix IV devices", January, 2014.

[13] "Basic Principles of Signal Integrity", White Paper, Altera Corporation, December, 2007.

[14] Marcel Welport, "Duobinary Coding Scheme and Real-Time Implementation", Research Report, SI Lab, Penn State Harrisburg, Fall, 2011.

[15]http://www.altera.com/support/examples/download/transceivertoolkit_examples_12_1.zip

[16] B.P. Lathi, "Modern Digital and Analog Communication Systems", 3rd Edition, Oxford University Press, 1998.

[17] Stratix IV Device Handbook, Volume 1, "Overview for the Stratix IV device family", Altera Corporation, September, 2012.

[18] Stratix IV Device Handbook, Volume 4, "DC and switching characteristics for Stratix IV Devices", Altera corporation, March, 2014.

[19] S. Pasupathy, "Correlative Coding: A Bandwidth-Efficient Signaling Scheme", *IEEE Communications Society Magazine*, pp. 4-11, July, 1977.

[20] "Using Pre-Emphasis and Equalization with Stratix GX", White Paper, Altera Corporation, September, 2003.

[21] D. Silage, "Digital Communication Systems Using Matlab and Simulink", Bookstand Publishing, 2009.

[22] J.G. Proakis, M. Salehi, "Digital Communications", Fifth Edition, McGraw Hill, November, 2007

[23] J.H. Sinsky, A. Adamiecki, M. Duelk, “10-Gb/s Electrical Backplane Transmission using Duobinary Signaling”, *IEEE MTT-S Digest*, pp. 109-112, 2004.

[24] M. Barazande-Pour, J. Houry, “Duobinary Transmission over ATCA Backplanes”, *IEEE 802.3ap Backplane Ethernet Task Force Plenary Meeting*, San Antonio, Texas, November, 2004.

[25] W. Kaiser, T. Wuth, M. Wichers, W. Rosenkranz, “A Simple System Upgrade from Binary to Duobinary”, *Proceedings National Fiber Optics Engineers Conference*, pp. 1043-1050, 2001.

[26] https://mysupport.altera.com/Training/registeredClassesViewDetail.html?regNo=1-719460961&courseId=1-7KEEC9