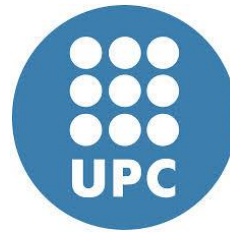# SASTRA UNIVERSITY

UNIVERSITAT POLITÈCNICA DE CATALUNYA

B. Tech. in Electronics & Communication Engineering

Bachelor Thesis

BIT ERROR RATE TEST FOR OPTICAL COMMUNICATION LINK USING PRBS GENERATED BY AN FPGA – HARDWARE IMPLEMENTATION

UPC Thesis Director : **Prof. José Antonio Lazaro Villa**
UPC Co- Director : **Senior Eng. José Antonio Altabas Navarro**
SASTRA Mentor : **Prof. M.Sridharan**

SADHVI VARAGUR KARTHIKEYAN

FEB - JUN 2014

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONA**TECH**

SASTRA UNIVERSITY

(A University u/s 3 of UGC Act, 1956)

Thanjavur – 613 401, Tamil Nadu, India

School of Electrical & Electronics Engineering

BONAFIDE CERTIFICATE

Certified that the project work entitled **"BIT ERROR RATE TEST FOR OPTICAL COMMUNICATION LINK USING PRBS GENERATED BY AN FPGA – HARDWARE IMPLEMENTATION"** is the work done by **SADHVI VARAGUR KARTHIKEYAN (Reg. No. : 114004170)** under my guidance in partial fulfilment for the award of the degree of Bachelor of Technology in **Electronics & Communication Engineering** during the period February – June 2014.

Prof. M. Sridharan   **Prof. José Antonio Lazaro Villa**
Internal Guide / Exchange Coordinator   **External Guide**

Senior Eng. José Antonio Altabas Navarro

Supervisor

Dr. K. Thenmozhi   **Dr. B. Viswanathan**
Associate Dean/ECE   **Dean SEEE**

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
UPC BARCELONATECH

Prof. **José Antonio Lazaro Villa**
Department of Signal theory and Communication,
ETSETB, Universitat Politecnica de Catalunya,
Barcelona, Spain

CERTIFICATE

This is to certify that the project work titled **"BIT ERROR RATE TEST FOR OPTICAL COMMUNICATION LINK USING PRBS GENERATED BY AN FPGA – HARDWARE IMPLEMENTATION"** submitted to SASTRA University by **Ms**. **SADHVI VARAGUR KARTHIKEYAN (Reg. No.: 114004170)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** is the original and independent work carried out under my guidance at Department of Signal theory and Communication, ETSETB, Universitat Politecnica de Catalunya, Barcelona, Spain, during the period February to June 2013. The contents of this thesis done by her, in full, or in parts have not been submitted to any institute or University for the award of any degree or diploma.

Place: **Prof. José Antonio Lazaro Villa**

Date:

                                                                      Official Seal

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
UPC BARCELONATECH

DECLARATION

I, hereby declare that the project work titled **"BIT ERROR RATE TEST FOR OPTICAL COMMUNICATION LINK USING PRBS GENERATED BY AN FPGA – HARDWARE IMPLEMENTATION"** submitted to SASTRA University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering was carried out at Department of Signal theory and Communication, ETSETB, Universitat Politecnica de Catalunya, Barcelona, Spain under the guidance of **Prof. José Antonio Lazaro Villa** (External Guide), ETSETB, Universitat Politecnica de Catalunya, Barcelona, Spain, **Prof. M. Sridharan** (Internal Guide), School of Electrical & Electronics Engineering, SASTRA University, India and **José Antonio Altabas Navarro** (Supervisor), ETSETB, Universitat Politecnica de Catalunya, Barcelona, Spain during the academic year 2013-2014. This was a record of my own work and to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted by any other University or Institute of higher learning, except where due acknowledgments have made in the text.

Place:

Date:

(SADHVI VARAGUR KARTHIKEYAN)

ACKNOWLEDGEMENTS

I, in the first place would like to thank **Prof. R. Sethuraman**, Vice-Chancellor, and **Dr. G. Bhalachandran**, Registrar, SASTRA University for giving me the opportunity to work in such a prestigious Institute of International status. My sincere gratitude's to Dr**. S. Vaidhyasubramaniam**, Dean-Planning and Development and **Dr. S. Swaminathan**, Dean-Sponsored Research for their support and encouragement.

I thank **Dr. B. Viswanathan**, Dean-SEEE, Associate Deans and Faculty members of SEEE, SASTRA University, for their encouragement and all the faculty members who have contributed in shaping me as an engineer.

Thanks to **Prof. José Antonio Lazaro Villa**, my project guide at UPC who incited and enlightened me all through my project experience. A very hearty thanks to my supervisor **José Antonio Altabas Navarro**, ETSETB, Universitat Politecnica de Catalunya, Barcelona, Spain. I must also mention the support provided by the International Office of ETSETB, UPC, Barcelona.

I am immensely grateful to my parents **Mr V.S. Karthikeyan** and **Ms. Usha Karthikeyan** for believing in me and my brother for his motivation.

I thank SASTRA University for giving me the opportunity to carry out my project work at ETSETB, UPC Barcelona, Spain, through the Semester Abroad Programme and also for providing me Desh-Videsh Scholarship. I am extremely grateful to my Internal guide **Prof. M. Sridharan,** Department of Electronics & Communication Engineering, SASTRA University for all the support for my project work / administrative procedures and for his constant initiative to maintain and strengthen the link between SASTRA and ETSETB, UPC Barcelona, Spain. I also convey my thanks to **Mr. Raja Subramanian**, SAP Coordinator, Department of Training, Placements and International Relations, SASTRA University.

I would finally like to thank Altera Forum and its members for providing me a platform to learn and interact with so many other engineers.

INDEX

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

ABSTRACT

Field Programmable Gate Arrays (FPGAs) have impacted the world of communication in a way that has not only simplified complex tasks but has also made the dream of achieving fast data throughput possible. The interfacing of FPGA with optical communication system, involves the measurement of link performance by testing and properly characterizing the communication system. Bit Error Rate Testing (BERT) is an important test to study the link performance of any data transmission system. In this report, the design of a Pseudo Random Bit Sequence (PRBS) generator & BER Tester using Altera's Cyclone III FPGA Starter Kit along with Terasic's THDB_ADA daughter board is done. A PRBS generator was designed and interfaced with a single mode optical fiber to study the nature and effect of channel imperfections resulting in bit errors when information is transmitted through the fiber.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
UPC BARCELONATECH

THESIS OBJECTIVE

The idea is to design and realize a Pseudorandom Bit Sequence Generator (PRBS) and Bit Error Rate Tester (BERT) on the Cyclone III Starter Kit along with the high speed AD/DA daughter card. The flow chart below summarizes the stages through which this design has been carried out.
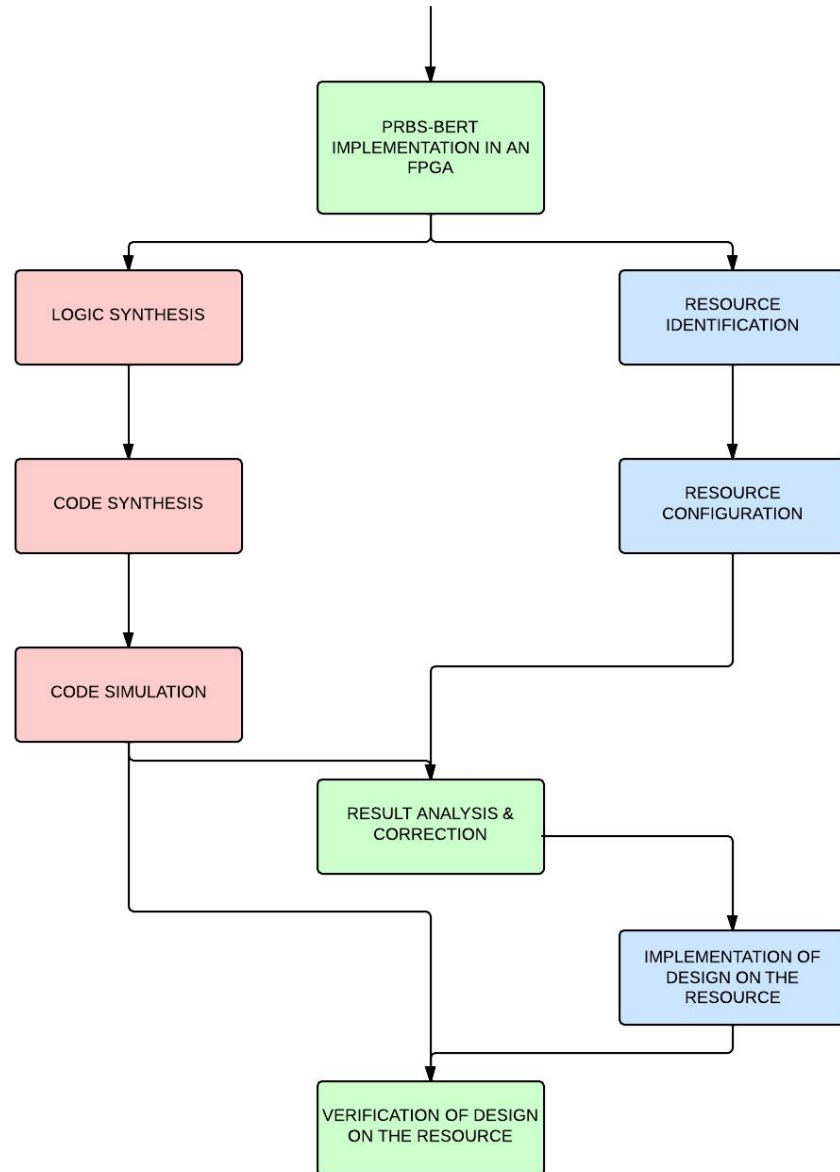
Pink, blue: Individual task, Green: Common task

This involves the following-

- Problem Statement Analysis
- Resource Identification
- Resource configuration
- Result Analysis and correction
- Implementation of design on the resource
- Verification of design on the resource

Problem Statement Analysis:

The given problem statement Pseudorandom Bit Sequence Generator (PRBS)-Bit Error Rate Tester (BERT) is analyzed to define a prototype logic.

Resource Identification:

The resource is the Cyclone III Starter Kit and the ADA_THDB daughter board. Resource identification involves finding out whether the hardware suits the design requirements such as optimal Operating speed for the design to run, ease of implementation, I/O supports, memory and all this at an economical price.

Resource Configuration:

This involves identifying the interconnection from the FPGA to the Daughter card and defining the FPGA with a code to work with the Daughter card. This code is the "configuration code" or the "Top-level entity" that loads on to the FPGA upon power-up. This way we ensure that the FPGA interconnection is safe for working with the daughter card.

Result Analysis and correction:

The results obtained from the code simulation stage are then analyzed for logical correctness and changed accordingly.

Implementation of design on the resource:

Once the logic is verified for its correctness, the design can then be downloaded on to the as part of the "Top-level entity" file and check for the working.

Verification of design on the resource:

Using the On-chip debugger tools available in the resource tool namely "Quaruts II" can probe the signals on the FPGA to see if it is working right. And then finally verify it in the Complete setup.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

1. INTRODUCTION TO OPTICAL COMMUNICATION AND ROLE OF FPGA AS AN INTERFACE

1.1. INTRODUCTION

Life, without communication, is like a car without wheels. It is only by expressing one's thoughts and ideas with many others who have similar and new thoughts, can the dreams be realized. It is only by realizing such dreams, innovation is encouraged. Communication has not only been a key factor for various technological advancements, but has also played an important role in maintaining strong bonds within friends, family members, who live in different corners of the world. The need to transmit information at a faster rate than that is already in use is the challenge for the communication engineers. As there is so much to know so quickly, today's world demands for more advancements in the field of communication, especially in terms of speed at which information is exchanged, quality of received information and the cost effectiveness.

One of the means of information exchange is by the use of light to transfer information from one point to another through a channel. This type of communication, called Optical communication, has become very popular in the recent days especially because of its massive data carrying capacity and very low post-installation maintenance cost.

In communications, digital signal processing plays a major role in linking the analog world with the digital world. In order to perform high-order filtering which are essential for processing digital signal, corresponding to the real world signals, high speed processors are needed. The traditional Digital Signal Processors which are designed to perform specialized DSP application are out-weighed by Field Programmable Gate Arrays (FPGAs) when it comes to efficiency in terms power consumption, parallel processing and speed of DSP operations [1]. For communication systems which involve high speed DSP operation, the use of FPGA is the best choice as it can implement complex logic very fast.

1.2. OPTICAL FIBER COMMUNICATION

The type of communication wherein information is transmitted from the transmitter to the receiver by modulating light that forms an electromagnetic carrier wave, in accordance to the message signal, is called optical communication. These optical pulses can be transmitted wirelessly or through wireline. Wireless transmission results in heavy loss of information due to various environmental effects which directly affect the unprotected signal. To protect the optical signal carrying important information, the optical pulses are passed through a fiber, essentially made of glass, which can transmit optical signal by total internal reflection with significantly less transmission loss. As signal is transmitted over long distances, losses do occur, for which Erbium-Doped Fiber Amplifiers (EDFA) can be used for signal regeneration.

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

1.2.1. FIBER OPTIC TRANSMISSION SYSTEMS

A complete setup of a fiber-optic transmission system consists of three basic elements: An optical transmitter, fiber optic channel and an optical receiver.

An *optical transmitter* gets electrical signal as input which is fed to an electro-optic modulator that modulates the signal by varying the intensity of light. The light source can be a laser or LED. The laser light is divided into two paths, by a beam splitter. One of the two paths has a phase modulator. The two paths are then recombined. By altering the electric field on the path which was phase modulated and by determining if the beams at the output interfere destructively or constructively, the intensity or amplitude of the exiting light can be controlled. This is done by a device called Mach-Zehnder modulator (MZM).

The *fiber-optic channel*, then, acts as a medium through which the optical pulses are transmitted. The main parts of an optic fiber are: the *core*, which is the center of the fiber through which light is transmitted, the *cladding* is the region that surrounds the core in a glass and has a refractive index less than that of the core, hence allowing the light to be confined within the fiber core and an outer coating, usually made of plastic, that provides strength and protects the optical fiber from external damage.

The *optical receiver* converts the transmitted optical pulse stream back to its corresponding electrical signal with the help of a photodetector. An electrical filter is usually used after the photodetector in order to enhance the signal quality.

1.2.2. NEED FOR OPTIC FIBER

The use of optic fibers that guide light waves for communication dates back to the 1970's. The fact that photon travels faster than electrons and, in addition can be guided similar to that of electrons, led to the use of light for achieving higher data rates at around Gb/s. This optical communication has evolved over three generations and is now available at over Tb/s data rates. These data rates have been contributed by various modern modulation techniques like the Wavelength Division Multiplexing (WDM). Also, hardware tools like the Field Programmable Gate Arrays that provide parallel, embedded Digital Signal Processing solutions facilitate an additional ease with achieving such high rates.

The data-handling capability of Optical fibers exceeds that of copper wire with respect to many factors like, very high bandwidths, immunity to electromagnetic interference, etc. Optical fibers, usually glass fibers, have the property of conducting a beam of light, containing within the fiber core by a phenomenon called total internal reflection and are free from problems like capacitance, inductance and other interference that usually affect electrical signals.

1.3. FIELD PROGRAMMABLE GATE ARRAYs (FPGA)

Field Programmable Gate Arrays are basically semiconductor innovation that can be programmed to adapt any functionality, be it simple or complex. This flexibility in them can be realized with the help of programmable logic called "Logic Elements" (LE). The FPGA gates are programmed using Hardware Description Languages (HDL) like Very High Speed Integrated Circuit Hardware Description Language (VHDL) or Verilog.

There are three leading FPGA manufacturers: Altera, Xilinx and Lattice. In this thesis, Altera's Cyclone III FPGA Starter Kit which has Cyclone III EP3C25F324 FPGA is used as the transmitter and receiver part of the optical communication system. The Cyclone III Starter Kit is an economical, easy to use embedded solution widely adapted for low cost DSP applications.

1.3.1. NEED TO USE FPGA

The FPGAs are reprogrammable silicon chips which are extremely flexible because of the fact that they can be fully reconfigured when compiled with a new circuit configuration. The main advantages of using an FPGA are its capability to process multiple functions in parallel, without having to perform the operations sequentially and to be used in communication interfaces because of its capability to provide sufficient processing power. A portion of the FPGA chip is allocated with a task to be performed and each task can run independent of the other logic blocks. This would make sure that, when new processes are added to the existing ones, the application that was already running is not affected. These have made possible in realizing various real-time and high-speed applications.

FPGAs, because of the advantage of hardware parallelism, tend to break the usual method of sequential execution and far exceed the computational power of Digital Signal Processors (DSPs) by being able to process more data per clock cycle. They have witnessed an increased demand in the communication field, especially for dedicated communication interfaces. It is therefore critical to properly test and characterize such interfaces before using them for the actual application.

1.3.2. RELEVANCE OF CYCLONE III FPGA KIT FOR THIS APPLICATION

The Cyclone III Starter Kit provides an easy and economical solution towards studying FPGA interfacing with Optical Communication. It does so with the help of on-board HSMC connector that helps interface the board with a THDB_ADA daughter board. This daughter board further enhances the DSP benefits of the Cyclone III Starter Kit with the help of dual-port, high speed Digital to Analog (D/A) and Analog to Digital (A/D) converters.

One of the important measures to study the performance of any digital communication system is Bit Error Rate Testing (BERT). In this work, Cyclone III FPGA acts as both the transmitter and the receiver to study the Bit error characteristic through a single mode fiber optic cable.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

2. PSEUDO RANDOM BIT SEQUENCE GENERATOR (PRBS) & BIT ERROR RATE TESTER (BERT)

PRBS-BERT is a common method of testing the performance of any link. This involves realization of transceiver on an FPGA, where, the PRBS forms the transmitter while the BER forms the receiver. Bit Error Rate in an optical link is the measure of number of bits that have been distorted due to various reasons like noise interference, losses due to transmission of signal along the optic fiber, chromatic dispersion, non-linear effects, etc., [2] when a stream of data bits are transmitted through an optical fiber from the transmitter to the receiver. It is the ratio between the total numbers of bits that got corrupted when passed through the channel and the total number of bits that were transmitted from the transmitter.

BER = (Total number of corrupted bits) / (Total number of bits transmitted from the transmitter)

2.1. OVERALL SETUP

The setup consists of Cyclone III EP3C25F324 FPGA in the Cyclone III Starter Kit which acts as a Pseudo Random Bit Sequence (PRBS) generator and as a Bit Error Rate Tester (BERT). A High Speed Mezzanine Connector (HSMC) connects the Cyclone III board to a daughter card which has DAC and ADC ports. The digital signal from the FPGA, which works at 50 MHz clock frequency, is passed through a 14 bit resolution DAC. The analog output from the DAC is fed to an electrical amplifier before the signal is modulated using a Mach-Zehnder Modulator (MZM). The MZM is an electro-optic modulator that modulates the electrical signal from the amplifier and thus produces intensity modulated optical signal.
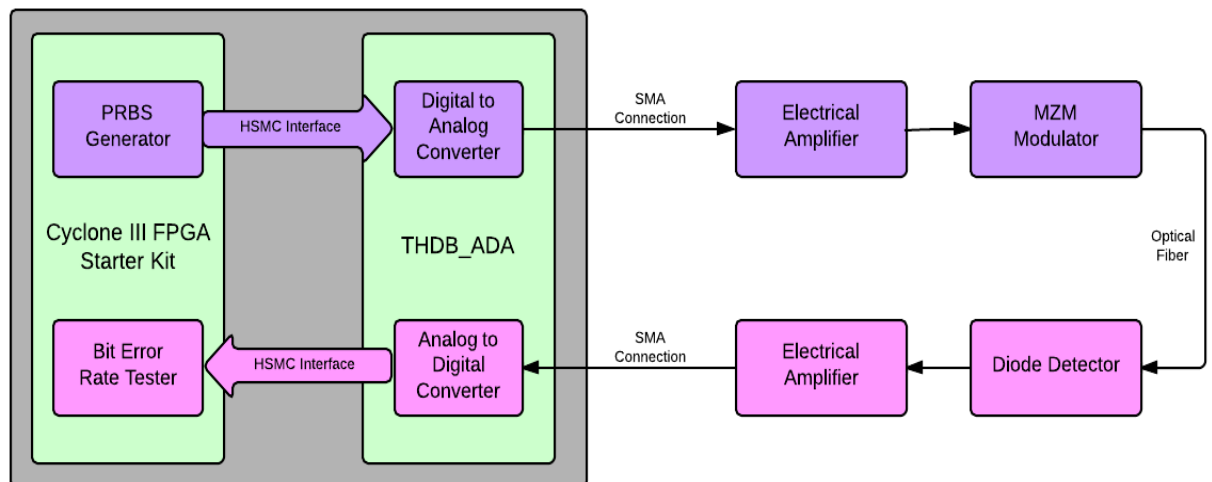
Figure 1: Basic optical link set-up

The optical signal is then passed through the optic fiber and received by a diode detector at the receiver. This diode detector converts the optical signal back to electrical signal which has to be amplified. The attenuated electrical signal is amplified using an electrical amplifier and is given as input to a 14 bit resolution ADC which is embedded on the daughter board.

The ADC converts the analog input to its corresponding digital binary value. This digital value is fed to the same FPGA which functions as a Bit Error Rate Tester and processes the received data to compute for bit errors, if any.

2.2. PRBS GENERATOR

In this thesis, PRBS generator has been implemented using a 4 bit Linear Feedback Shift Register (LFSR). An LFSR is a shift register in which, a linear function of the output of one state is fed back as the input to the next state. The most widely used linear function is XOR operation. Since the operation is deterministic, the sequence of values that would result by continuously performing such operations can be determined by knowing the initial state of the shift register. The initial value of the shift register is called the seed and the bit positions that affect the next state are called taps. The seed value can be anything other than all zeroes value [3].

There are N registers in an LFSR circuit of length N and the sequence of states through which the LFSR transitions through is determined by the feedback taps. The feedback taps are usually modulo-2 polynomial. An N bit LFSR cycles through a Maximum Length Sequence (MLS) and produces all possible ((2^N)-1) states before it starts to repeat itself. For example, a 4 bit LFSR repeats its states after 15 states. The PRBS signal is a single binary output from the LFSR. This binary output is pseudorandom as the LFSR sequence is deterministic, but at the same time has the correlation properties of a random signal [3].

PRBS is semi-random in the sense that it appears random within the length of the sequence, but repeats the full sequence beyond that, at regular intervals. To a random observer, the sequence appears to be totally random however the user who knows the seed value and all the properties of the sequence can predict the values.

2.3. PRBS THROUGH OPTICAL LINK

A PRBS can be used for studying the optical channel characteristics like signal integrity and receiver sensitivity by observing changes in the PRBS bits after it is received at the receiver end of the optical link. This could be done by locally generating PRBS sequence at the transmitter and the receiver blocks of the optical communication system. In order to synchronize the two locally generated pseudorandom bits, the transmitter would first send initialization bits to the receiver [4]. Given that the first few initialization bits are received correctly, the transmitter and the receiver PRBS generators are synchronized. At the receiver, comparison between the received PRBS bits and the locally generated PRBS bits begins, in order to determine the bit errors due to the communication channel.

3. IMPLEMENTATION: SOFTWARE AND HARDWARE

This section illustrates the tools required for any design implementation on the FPGA and the logic used to implement BERT. Initially, this involves creating a basic design file that defines all the pin assignments and constraints, especially for the HSMC pins needed for all future designs. This file in general is referred to as the Top-Level entity. Any further implementation will be instantiated as a component within this Top-Level entity. This would ensure that the hardware is always working under the "safe-mode", thus avoiding any damage due to logic level conflicts while interfacing with the daughter board.

Software and Hardware Requirements

1) Cyclone III FPGA Starter Kit

2) THDB_ADA Daughter Board

3) Quartus II 11.1

4) ModelSim Edition 6.5b

3.1. HARDWARE SPECIFICATIONS

3.1.1. CYCLONE III FPGA STARTER KIT

The Cyclone III Starter Kit is powered by a 12V adapter. An external USB cable enables connection from PC to the Board. The Cyclone III Board when powered up loads a configuration file from its Flash. This file moves the FPGA into the "user mode" which ensures that the I/O pins can flexibly be assigned to perform the function required by removing them from weak pull-up. This configuration file can be .sof or a .pof file. The .sof (SRAM object file) configuration is lost when power is turned off while the .pof (Programming object file) runs the configuration file from the flash device present on board even during power off. Configuring also involves creating a top-level design that includes all the pin assignments and constraints needed for all of future designs.

Configuration files-

Scripts: It contains the Synopsys Design Constraint file (SDC) specifying the timing Constraints, synth.tcl (that sources the required files and compiles them into a folder "qwork").

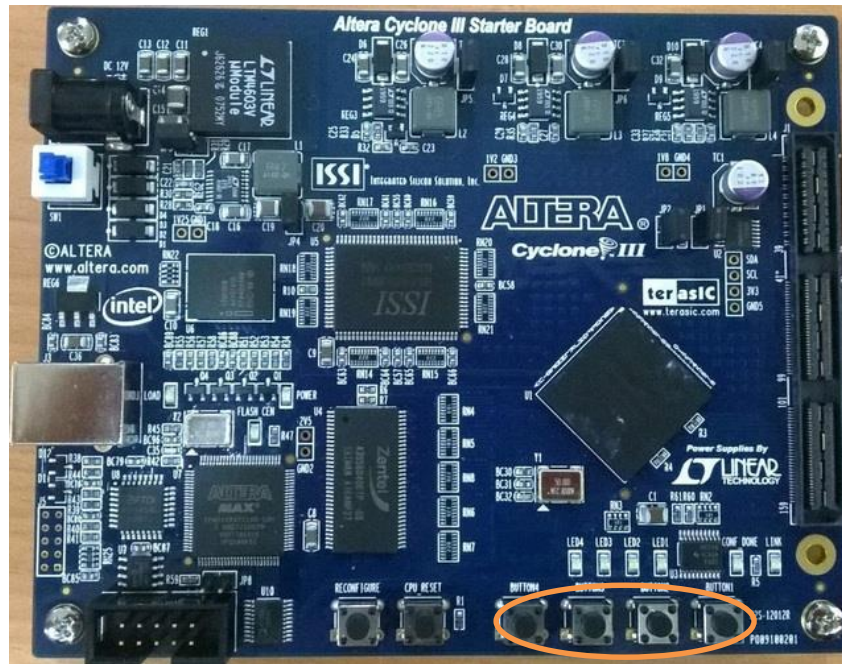**Source:** This file contains the top-level design.

FPGA Board-

Figure 2: Cyclone III FPGA Starter Kit

3.1.2. THDB_ ADA DAUGHTER BOARD

The Terasic Analog-to-Digital and Digital-to-Analog (ADA) board is interfaced to the Cyclone III FPGA Starter Kit via the HSMC (High Speed Mezzanine Card). Before connecting the two boards, we must ensure logical compatibility between the signal levels and pins. For this purpose a spreadsheet was first created inferring all the above details of the two boards. This spreadsheet can be used for all future designs to ensure correct pin assignments. The compatibility here refers to verifying that the power rails of the two boards match and the pins are at the same logic-level. The THDB_ADA board comes with a utility that facilitates instant verification of the board. This utility is a software waveform generator that can generate different waveforms like a square, sine, triangular wave, etc. up to a maximum frequency of 5MHz [5]. It can also be used as an arbitrary waveform generator, capable of generating any random waveform that we draw on the system screen.

In order to view the PRBS output on the oscilloscope, the design involves routing the PRBS bit signal to the DAC channels via the HSMC port. Here the DAC has a 14-bit resolution. And so a PRBS output bit of '0' will assign the corresponding analog minimum value, which the electrical amplifier in the setup can support, to the 14 DAC channels. Similarly a PRBS bit of '1' will assign the corresponding analog maximum value, which the electrical amplifier can support, to the 14 DAC channels. Thus the DAC output swings between the required (depending on the maximum and minimum voltage value supported by the amplifier used in the setup) maximum and minimum

value. This signal is then modulated using an MZM to produce the required optical output to be passed via the fiber.
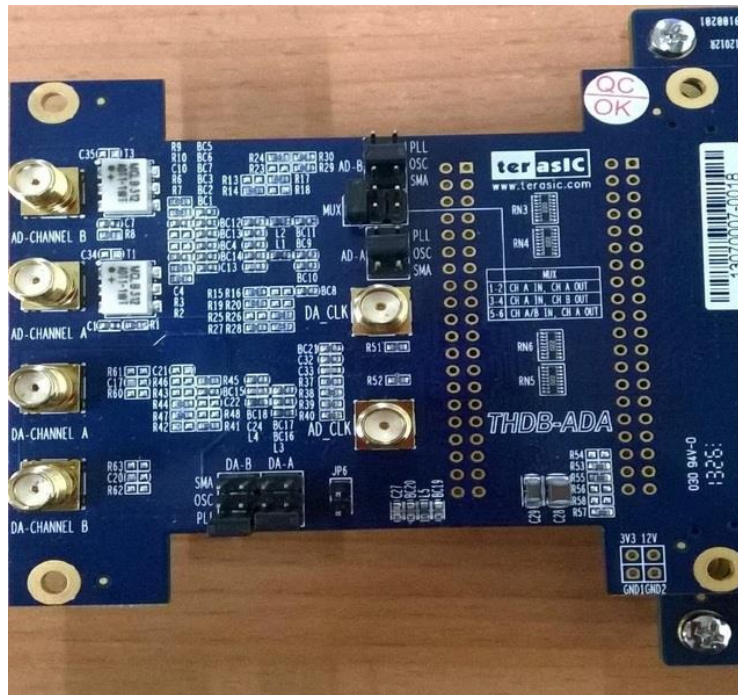
Figure 3: THDB_ADA board

3.1.3. BUTTON DEBOUNCING

The Cyclone III FPGA Starter Kit has 4 user push buttons as shown in Figure 2 (marked). When a push-button is used as a source of input, problems like rebound or bouncing of the mechanical switch can occur. The bouncing of the button might sometimes be considered as an input and can produce erroneous results. This problem can be avoided by using two flip flops to check if the button press is producing a stable value for a minimum time, say 10 milliseconds (for 50MHz clock). The button's logic level is continuously clocked by the first flip flop which in turn is connected to the second flip flop and a counter. The counter is incremented until it reaches its maximum value of 10 milliseconds. Once the counter reaches its maximum count value, the second flip flop is triggered to output the logic level of the button. This logic has been implemented through software as a VHDL code to ensure that only a stable value is considered as button press.

3.2. SOFTWARE SPECIFICATIONS

The two main Tools that automate design using the Cyclone III Kit are:

1) ModelSim Edition 6.5b

2) Quartus II 11.1

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

3.2.1. MODELSIM

All VHDL codes are simulated using this tool. It enables instant modeling of Real-time designs and verification of the logic. This is done with the help of a Test bench.

A *Test bench* is a "stimulus code" that verifies the functionality of the design. It consists of all the initializations to be done at the beginning, such that the design gets triggered. A test bench code is called through a set of ModelSim commands. It requires creating the following set of files:

1. Source Code (source)

2. Test bench code (simulation only)

3. Script (consists of .tcl and .do files that automates the simulation)

3.2.2. QUARTUS II

It is the software support for the board provided by Altera. It enables the user to verify their design through a series of steps. For this thesis, Quartus II 11.1 sp1 Web Edition was used. A Quartus II project includes all the necessary design files and software source files necessary for the implementation of a design. This tool verifies the user design at each of the following stages.

Design entry stage-

This involves creating a design by means of a VHDL code which could be done using the *Text editor*. The design file could also be a Verilog file or a Tcl Script.

The Tool Command Language (TCL) is a scripting language that allows making device assignments, define design constraints, compile the design, perform timing analysis, etc. It is basically a comprehensive detail of the current set top level entity. The VHDL designs can contain built-in MegaWizard components called the Altera Megafunctions.

Altera Megafunctions: The Quartus II software provides a MegaWizard Plug-In Manager in the Tools menu. This tool enables using the existing built-in designs to suit user requirement by just varying the parameters. They are then instantiated as components in the user code. These reconfigurable functions help in saving a lot of design time.

Assignment editor: It is the next part of *Design entry* which involves mapping the pins present in the design to the FPGA pins.

The Pin Planner is a tool that depicts the ports mentioned in the design, their direction, I/O bank, location (i.e. the FPGA pin to which they are assigned), reference voltage and their I/O standard and current strength. I/O pins can be defined to use numerous standards e.g., the DDR

memory has to use 2.5V SSTL logic, but the LEDs could use anything. However, since the board has a 2.5V power supply, the LEDs also use 2.5V, but they use the logic standard "2.5 V", which just means 2.5V CMOS logic levels. Hence a brief overview of the hardware pin assignments is obtained.
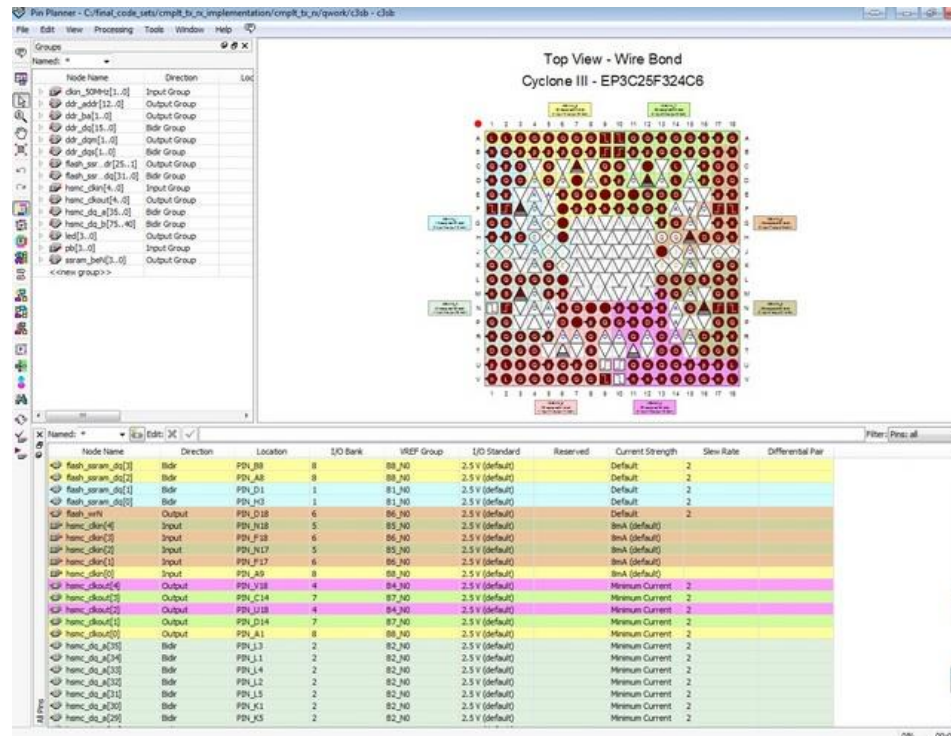
Figure 4: Pin Planner

Analysis and synthesis stage-

As the name indicates it synthesizes the design after verifying the syntax and checking for the logical correctness of the design. The analysis and synthesis stage concludes on the amount of flip-flops, logic elements required, and uses specific algorithms that make sure in using minimum number of these elements. The design can then be viewed using the RTL viewer that gives a graphical representation.

Fitter-

It uses the information from Analysis and Synthesis and verifies the pin assignments. Verifies that the logic- levels match and also that the timing requirements are satisfied. The results of the fitter analysis can be viewed with the help of the Chip Planner.

Chip planner can be used to locate the logic cells that are used by the pin planner. The pins assigned can be easily differentiated from the unassigned once by their color. Mainly this tool helps to check the amount of congestion that occurs for a given design and edit it accordingly.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

TimeQuest timing analysis-

This involves analysis of number of clock signals in the design, the path between these clocks, Setup and Hold time and Clock to output delay. This ensures that the design successfully establishes the path flow between all the modules without any timing clash. For instance, for DAC interface, the timing diagram can be got from the datasheet which will have to be translated into timing constraints. These timing constraints can then be verified by the timing report generated by Quartus. The timing report basically tells if the input signal to the DAC meets its specified setup/hold time. The constraint file thus translated is called a Synopsis Design File (SDC).

Assembler (generate programming files) stage-

It generates the programming file that can be downloaded onto the hardware once the project is successfully compiled and synthesized. The programming file can be of two types: A .sof (SRAM object file) or a .pof (programmable object file). These are explained in detail under the FPGA configuration section.

Debugging-

SignalTap II Logic Analyzer is a very useful debugging tool provided by Quartus II which can be used to track the signals assigned to the hardware pins. It provides an oscilloscopic view of the signals present in the design. This is done by routing the signals via the JTAG port.

3.2.3. LOGIC IMPLEMENTED

PRBS-DAC Signaling -

The diagram below describes the signaling between the 1-bit PRBS generator (FPGA) and the DAC available on the daughter board. The 1 bit PRBS is given as input to all the 14-channels of the DAC. So if the 1 bit PRBS value is a '1', 14 bit equivalent of the maximum voltage value supported by the amplifier in the setup, is given to the 14 channels of the DAC. Similarly for '0', the minimum voltage value supported by the amplifier in the setup is given as input to the 14 channels of the DAC. The DAC is operated in dual mode and its outputs are updated at every rising edge of the clock.
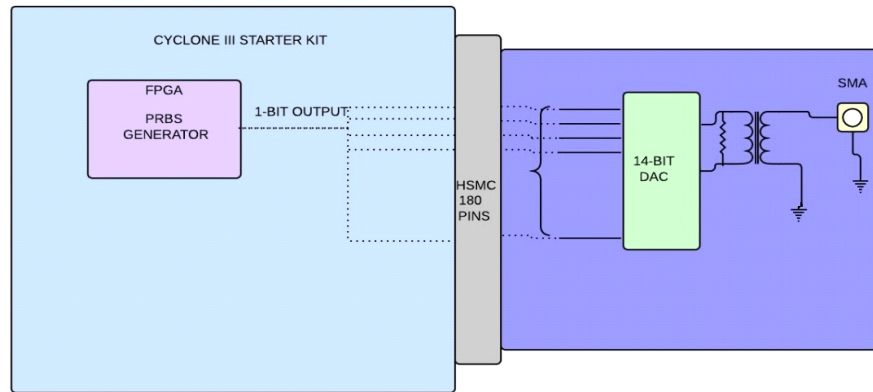
Figure 5: *PRBS from an FPGA to DAC Signaling*

The logic for transmission of PRBS signal from the transmitter and the Bit error rate testing at the receiver has been implemented in two ways.

Method 1-

In this method, the transmitter LFSR is loaded with a seed value as soon as the power is switched on and the transmitter starts to generate PRBS signal. Fibonacci LFSR has been chosen to implement the pattern generator and the VHDL implementation for the PRBS generation is shown below:

```
>>        if (load = '1') then
              q <= seed;
          elsif (enable = '1') then
              q <= (q(1) xor q(0)) & q(3 downto 1);
          end if;
```
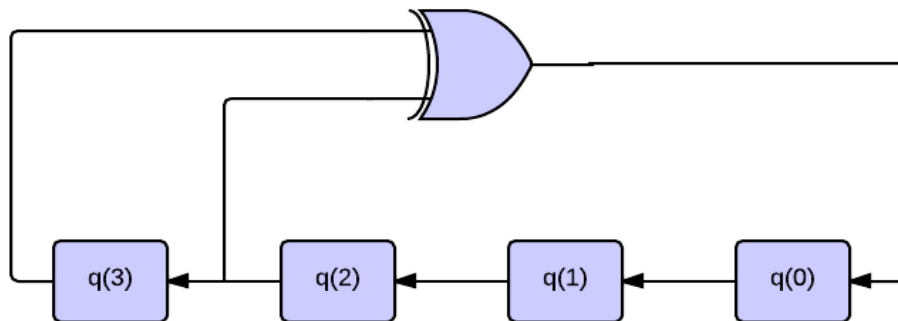

Figure 6: 4 Bit LFSR Signal diagram

The MSB of the transmitter data register is sent as PRBS signal and is received by a receiver shift register. Only after 4 clock cycles, the receiver shift register is fully loaded with the transmitter data register's 0th state value as shown in the VHDL code below:

```
>>            if (rstN = '0') then

              rx_reg <= (others => '0');

              elsif rising_edge (clk) then

              rx_reg <= rx_reg (2 downto 0) & tx_data (3);
```

During every clock cycle, a function of the receiver shift register's content is loaded into the receiver seed register. From the 5th clock cycle onwards, the transmitter data register's state and the receiver seed register's state are the same. From the Table 1, it can be seen that the 4th state of the transmitter data register will be a function of the 0th state and hence, the same function is used to load the receiver seed register from the receiver shift register in order to have the receiver seed register to be in the same state as the transmitter data register. When the seed is, say, f = 1100. The 1,1,0,0 are transmitted one by one and received by the rx_reg. By the end of 4 clock cycles, the rx_reg will have the seed value but the txr_data will have this value:

(f(3) xor f(2)) & (f(2) xor f(1)) & (f(1) xor f(0)) & (f(3) xor f(2) xor f(0)) = 0100

The rxr_seed should be in the same state as that of the txr_data, as the rxr_seed is loaded into the rxr_data in the next clk and the rxr_prbs is compared with the delayed txt_prbs. So, the rxr_seed should be:

```
>>   rx_seed   <=   (rx_reg (3) xor rx_reg (2)) &

                    (rx_reg (2) xor rx_reg (1)) &

                    (rx_reg (1) xor rx_reg (0)) &

                    (rx_reg (3) xor rx_reg (2) xor rx_reg (0));
```

So, rx_seed= 0100. In this way, once the transmitter LFSR is enabled, the rx_seed is made sure to be same as that of the tx_data, so that, once the receiver LFSR is loaded with the seed, the delayed txr_prbs and the rx_prbs bits are in synchronization. For loading the receiver seed value, the receiver LFSR is loaded with the receiver seed register's content, once a push button is pressed on the Cyclone III board. Since the transmitter data register and the receiver seed register are in the same state and the receiver LFSR is loaded from the receiver seed register, the transmitter data

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
UPC BARCELONATECH

register will be one clock ahead of the receiver data register. So, the MSB of the receiver LFSR and the delayed version of the MSB of the transmitter LFSR, that was transmitted, are compared to check for bit errors.

Comparison is done by performing XOR operation between the delayed transmitter PRBS signal and the receiver PRBS signal. Mismatch in the comparison, increments a counter which shows the number of errors detected.

```
>>        prbs_mismatch <= tx_prbs_dly xor rx_prbs;

>>        if (prbs_mismatch = '1') then

            count <= count + 1;

        end if;
```

Table 1: PRBS4 shift-register contents (taps [1, 2], X3 +X + 1, 11001b, Fibonacci form, XOR feedback).

SAMPLE INDEX	f3	f2	f1	f0
0	3	2	1	0
1	2	1	0	3 xor 2
2	1	0	3 xor 2	2 xor 1
3	0	3 xor 2	2 xor 1	1 xor 0
4	3 xor 2	2 xor 1	1 xor 0	0 xor 3 xor 2
5	2 xor 1	1 xor 0	0 xor 3 xor 2	3 xor 1
6	1 xor 0	0 xor 3 xor 2	3 xor 1	2 xor 0
7	0 xor 3 xor 2	3 xor 1	2 xor 0	1 xor 3 xor 2
8	3 xor 1	2 xor 0	1 xor 3 xor 2	0 xor 2 xor 1
9	2 xor 0	1 xor 3 xor 2	0 xor 2 xor 1	3 xor 1 xor 2 xor 0

10	1 xor 3 xor 2	0 xor 2 xor 1	3 xor 1 xor 2 xor 0	0 xor 1 xor 3
11	0 xor 2 xor 1	3 xor 1 xor 2 xor 0	0 xor 1 xor 3	0 xor 3
12	3 xor 1 xor 2 xor 0	0 xor 1 xor 3	0 xor 3	3
13	0 xor 1 xor 3	0 xor 3	3	2
14	0 xor 3	3	2	1

Table 2: PRBS4 sequence for taps [1, 2], X3 +X + 1, 11001b.

SAMPLE INDEX	LFSR STATE	LFSR STATE	PRBS OUTPUT
	BINARY	HEX	
0	1111	F	1
1	1110	E	1
2	1100	C	1
3	1000	8	1
4	0001	1	0
5	0010	2	0
6	0100	4	0
7	1001	9	1
8	0011	3	0
9	0110	6	0
10	1101	D	1
11	1010	A	1
12	0101	5	0
13	1011	B	1
14	0111	7	0

Clock Alignment

The Cyclone III FPGA acts as both the transmitter and the receiver and hence both are clocked using the same source. But, once the digital signal goes outside the FPGA and is received at the receiver, the receiver's clock phase will not be the same as that of the transmitter's clock phase. Hence, a separate receiver clock with adjustable phase, with the help of PLL Megafunction provided by Quartus II software, is used so that it can be aligned with the incoming data at the receiver.
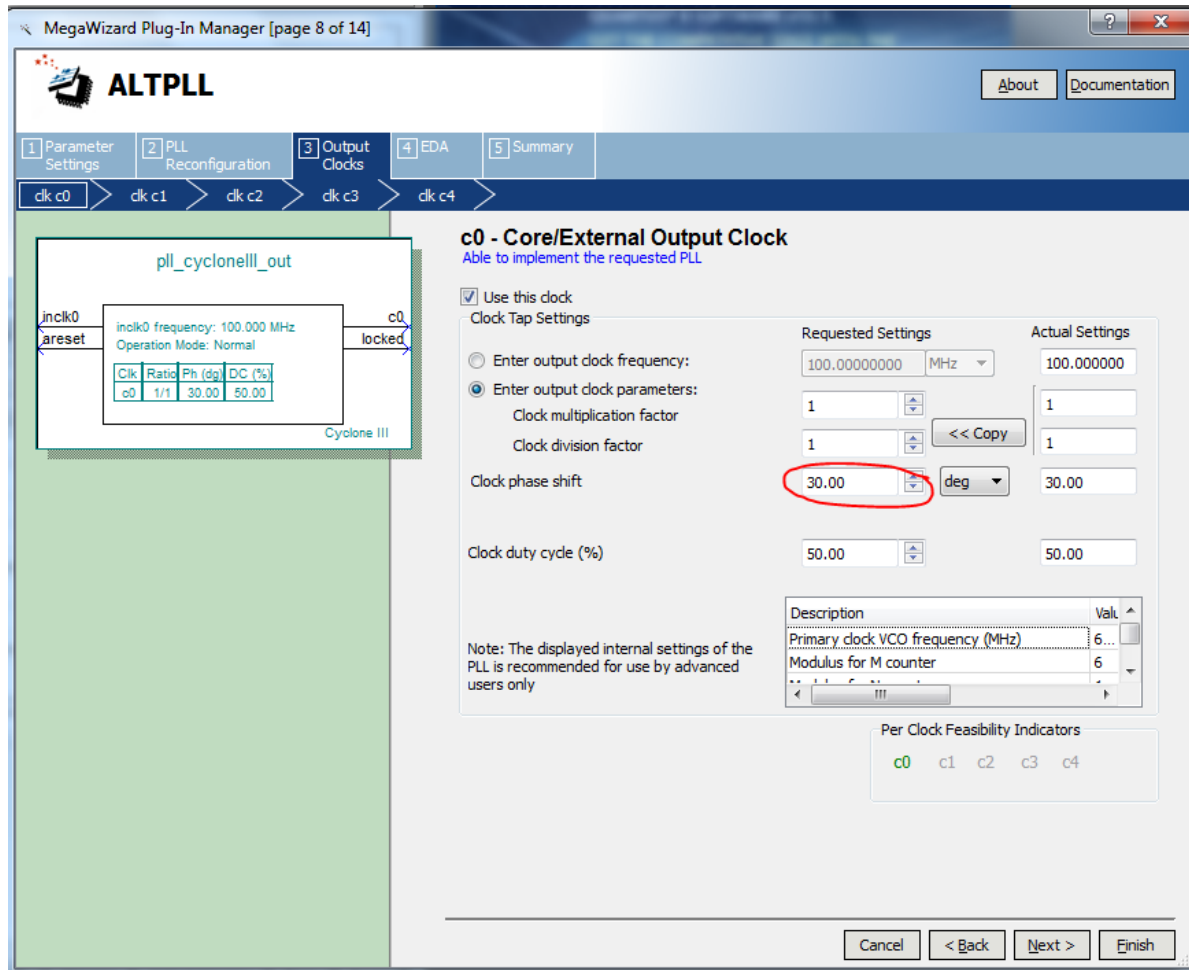
Figure 7: The PLL Megafunction in Quartus.

The reference input signal to the PLL is the same as the transmitter's clock (50MHz).The output of the PLL has the same frequency as that of the transmitter but whose phase can be adjusted. This helps in ensuring that the delayed data at the receiver is sampled exactly at the center of the data bit. The transmitter sends data such that the rising and falling edges of the clock are at the middle of the data. But, once the data is sent outside the FPGA, there will be delay due to the

various modules, and hence at the receiver, the rising and falling edges of the receiver's clock might not be at the center of the received data if the transmitter and the receiver's clock source and phase are same.

The incoming data at the receiver should satisfy the setup and hold time of the receiver register [6]. Setup time is the minimum time the data should be stable before it is read into a flip flop. Hold time is the minimum time the data should be stable after it is read into a flip flop. Violation of the setup and hold time will lead to the register going into metastable state and could incorrectly output the opposite logic value and erroneously be counted as bit error. So, in order to make sure that there are no logic level transitions during the setup and hold time, after the transmitted data is delayed due to the channel, the phase of the receiver clock is adjusted such that the data is properly captured at the receiver. Since there are multiple phase-shift settings, JTAG-to-Avalon-MM bridge can be used to dynamically change the phase of the receiver clock [7].
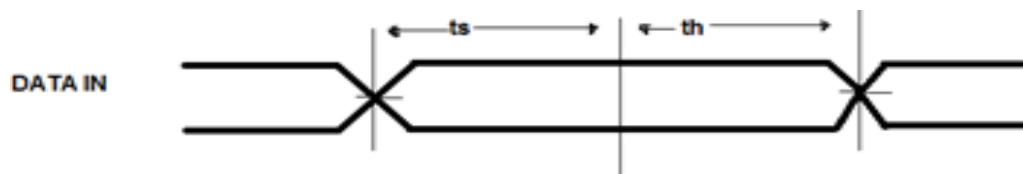
Figure 8: Data- Setup and Hold time

The above method concentrates on realizing transmitter and receiver synchronization using different seeds. However, in a real-time system where noise and other kinds of distortions are random and probable, the intricacies increase. Hence, the probability of the seed getting corrupted becomes yet another possible reason, in the myriad of possibilities, for high BER at the receiver. So Method 2, where the seeds are hard-coded in the FPGA and loaded independently into the transmitter and receiver was proposed. This reduces the number of trials and errors by reducing the possibility of seed corruption. Also, here the PRBS system on the FPGA is then said to be more reliable and stable.

Method 2-

The transmitter sends 4 bits (initialization bits-can be any number of bits), which after fully received by the receiver, is compared with the same 4 bits that is already stored at the receiver. Once the comparison shows no error, it is indicated by a glowing an LED and the transmitter sends PRBS bits which are initially not used by the receiver. A push button is used to load the seed value into the Transmitter LFSR and the Receiver LFSR independently for PRBS generation. Comparison starts thereafter. The received PRBS value is compared with the PRBS value generated by the receiver LFSR by performing XOR operation between the received PRBS signal

and the PRBS signal locally generated at the receiver. Count is incremented each time a bit error is encountered. If the initial few bits are not the same as the bits stored at the receiver, then the bits are re-transmitted and the same procedure is to be followed.

```
>>          if count_1 > 0 or count_1 = 0 then

            rx_reg   <=  rx_reg(2 downto 0) & prbs_1;

            count_1  <=  count_1 - 1;

        elsif rx_reg  /=  initial_cmpr  then

            led3_pattern_error    <= '1';

            led4_no_pattern_error <= '0';

        else

            led3_pattern_error    <= '0';

            led4_no_pattern_error <= '1';
```

When all the 4 bits are received without error, one LED glows (led3_pattern_error from the code). So, when the load button is pressed, the transmitter and receiver LFSR's are loaded with the same seed value. Bit Error counting begins thereafter by incrementing count value for each bit error. When there is an error in the received initialization bits, another LED glows (led4_no_pattern_error) and the initialization bits are re-transmitted and the procedure continues for 3 to 4 trials. If the initialization bits are received incorrectly for more than 4 times, there is a problem with the channel which has to be investigated.

For the above two methods, the 14 bit output from the ADC should be processed to produce a single bit, as the received PRBS bit. In the Method 1, this was be done by using the sign bit. The sign bit is 1 for positive voltage and 0 for negative voltage as the DAC is operated in differential mode. In Method 2, the 14 bit output from the ADC is compared with a reference value (14 bit value). A value greater than the reference value would output '1' and, a value lesser than the reference value would output '0'. The reference value can be fixed with the help of initialization bits that are sent from the transmitter, before BERT begins.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH
UPC

4. EXPERIMENTAL RESULTS

4.1.   PRBS

The Cyclone III FPGA which acts as a PRBS generator produces digital signal which is given as input to a 14 bit resolution Digital to Analog converter (DAC). The one bit PRBS signal from the transmitter LFSR which is fed as input to all the 14 channels of the DAC can be tracked with the help of a SignalTap II Logic Analyzer tool provided by Quartus.
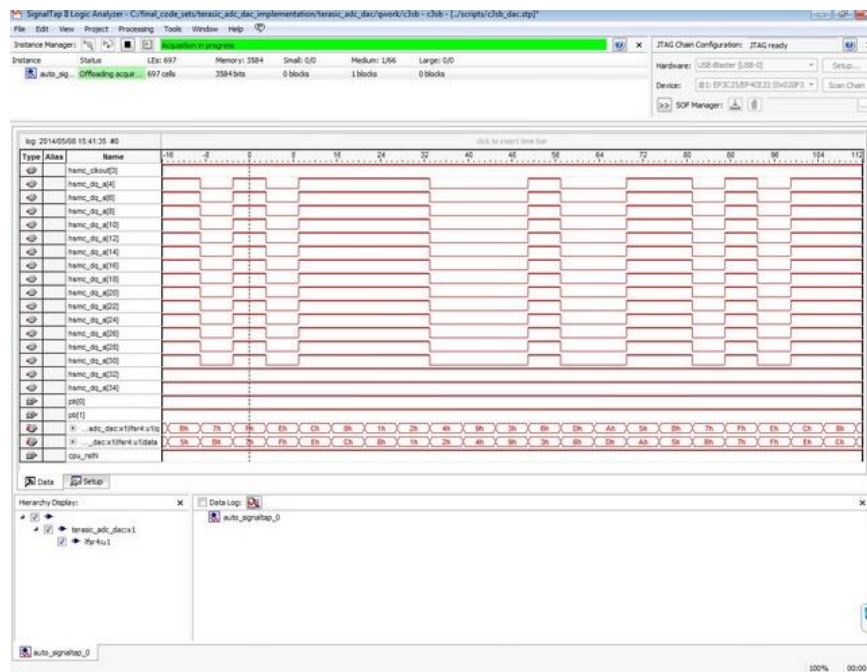
Figure 9: HSMC signal probing for PRBS-DAC interface - SignalTap II Logic Analyzer

The analog output from the DAC is viewed on an oscilloscope. Theoretically, the waveform should resemble a perfect digital waveform with sharp rising and falling edges. The reason for the waveform, as seen on the oscilloscope (Figure 10), to have curves could be because of the connector used.
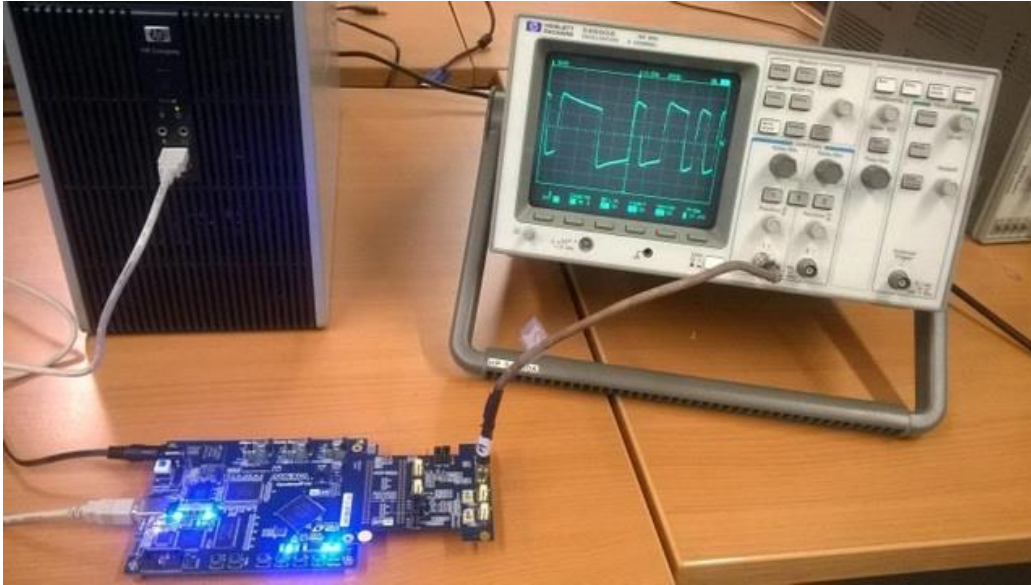
Figure 10: PRBS signal on an oscilloscope.

4.2. PRBS THROUGH THE OPTICAL LINK
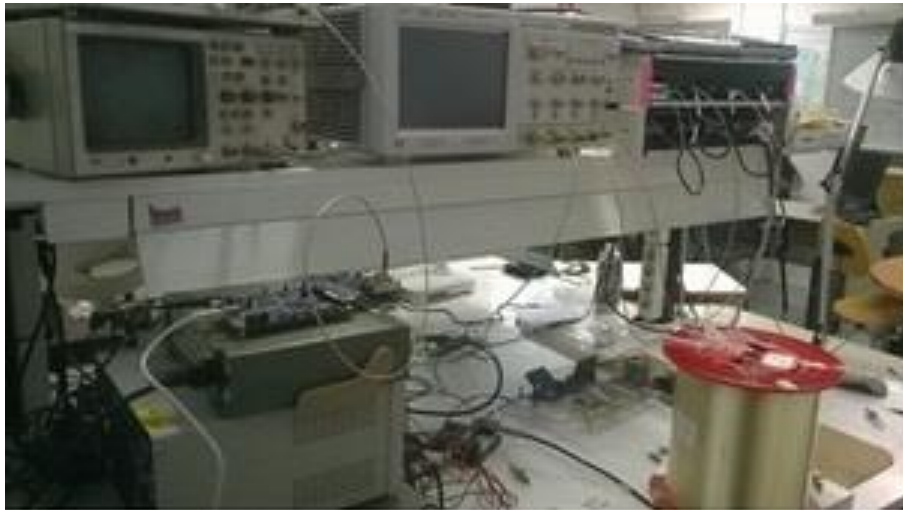
Figure 11: FPGA Interfaced with the Optical Setup

The Cyclone III FPGA board along with the THDB_ADA daughter board is connected to a Mach Zehnder Modulator to convert the analog electrical signal from the DAC, to its equivalent optical signal to be able to be transmitted through an optical fiber. The optical output signal from the MZM was viewed on the oscilloscope which produced a signal as shown in Figure 12. The optical signal after passing through a 25 kilometers long optical fiber when viewed using an oscilloscope looks as shown in Figure 13. This signal is then to be passed through a diode detector,

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

to convert the optical signal back to electrical signal and is to be given as input to the ADC present on the THDB_ADA board.

The FPGA which acts as the transmitter generates PRBS signal. The digital data is given as input to a 14 input DAC in the THDB_ADA board. The output from the DAC is an electrical signal which when captured on an oscilloscope appears as shown in Figure 12. The signal on the top is the electrical signal. This, when passed through a Mach-Zehnder Modulator, produces an equivalent optical signal which when captured on an oscilloscope is as shown in the Figure. The second signal is the optical signal. There is very little delay as the signal reaches till the Mach-Zehnder Modulator which can be seen as a shift in the second signal with respect to the first.
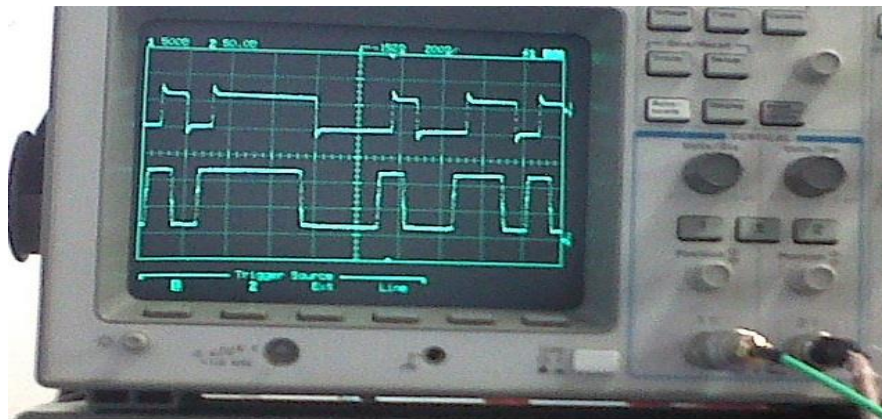
Figure 12: PRBS as an electrical signal and optical signal before passing through a 25kms long optical fiber.

The optical signal is then passed through a 25 kilometers long optical fiber. The optical fiber can allow even very low frequency (in the order of MHz) signal to be transmitted through it. The 50MHz PRBS signal after passing through the optical fiber is captured on an oscilloscope as shown in Figure 13.
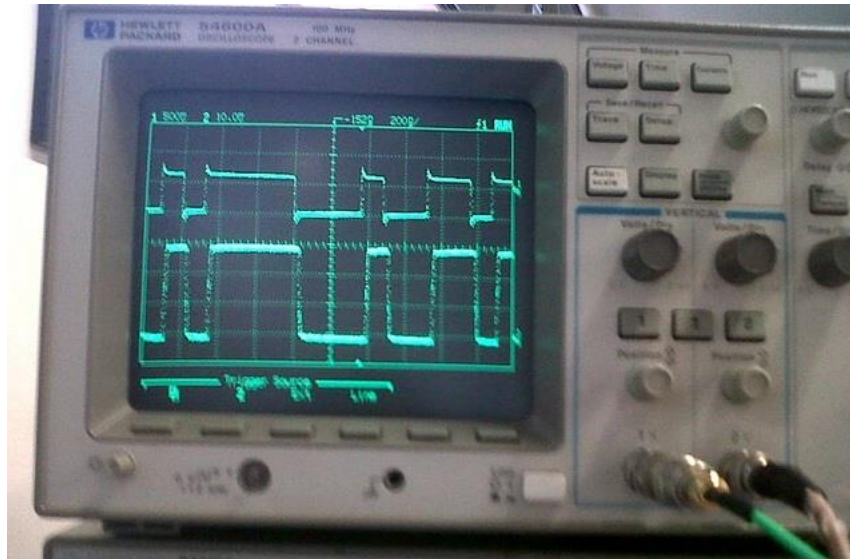
Figure 13: PRBS as an electrical signal and optical signal after passing through a 25kms long optical fiber.

Due to the length of the fiber and the various components in the setup, there is a considerable delay in the signal at the receiver end of the optical fiber.

Delay Time (T) = Distance travelled / (Speed of light/Refractive index of optic fiber)

T= 25000 / (3.8 * 10^8/1.46) = 96.0526 microseconds

4.3. TRANSMITTER_ RECEIVER – FPGA

The Signal from the transmitter is first directly fed to the receiver within the FPGA, without interfacing it with the optical link.

4.3.1. METHOD 1

In this method, the transmitter data register produces pseudo random bits and continuously updates the receiver seed register with the same state as that of the transmitter data register. After load button is pressed, the receiver data register is loaded with the contents of the receiver seed register. The transmitter data register is one clock cycle ahead of the receiver data register. Bit error is done by comparing the delayed received bit (MSB of transmitter data register) and the MSB of the receiver data register. The ModelSim simulation (Figure 14) shows zero count for bit errors after the button press, as there is no source of channel distortion.

Figure 14: Method 1 – ModelSim Waveform

Hardware testing of the code involves pressing a button to load the receiver LFSR seed value into the receiver data register. Before loading the receiver data register with the seed value, the receiver data register value and the transmitter data register are not the same and hence error count is non-zero. Comparison is made between the tx_prbs_dly and the rx_prbs_out signals. A non-zero error count value turns on LED 4 while a zero error count turns on LED 3.

Figure 15: Before loading receiver LFSR- SignalTap II Logic Analyzer

Figure 16: Before loading receiver LFSR- error produced- LED 4(circled in orange) is ON

After loading the receiver data register with the seed value, the tx_prbs_dly and the rx_prbs_out signals have the same value, as seen in simulation, as there can be no possible error since there is no channel to distort the signal. Hence the error count is zero which turns on LED 3.
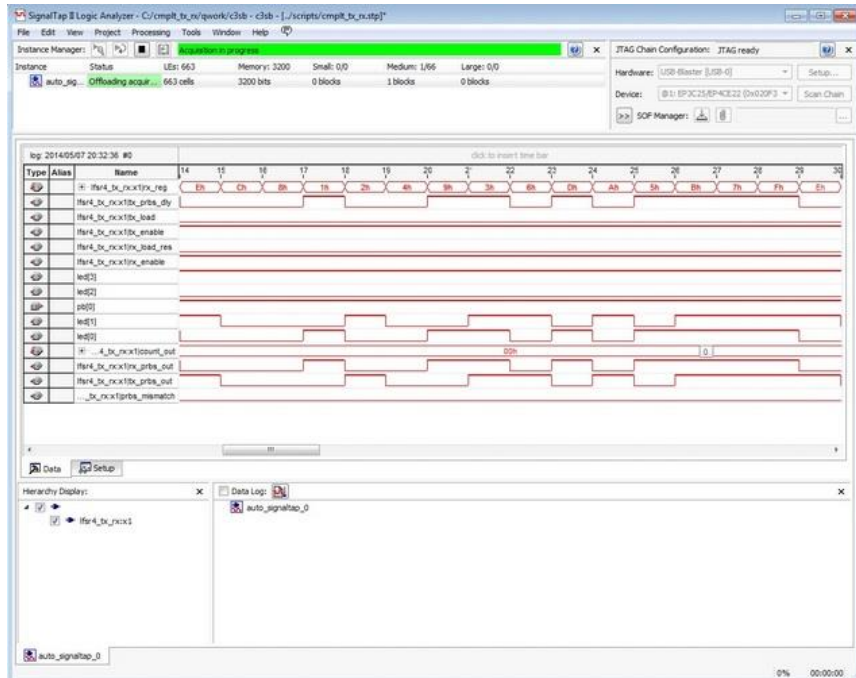
Figure 17: After loading receiver LFSR- SignalTap II Logic Analyzer
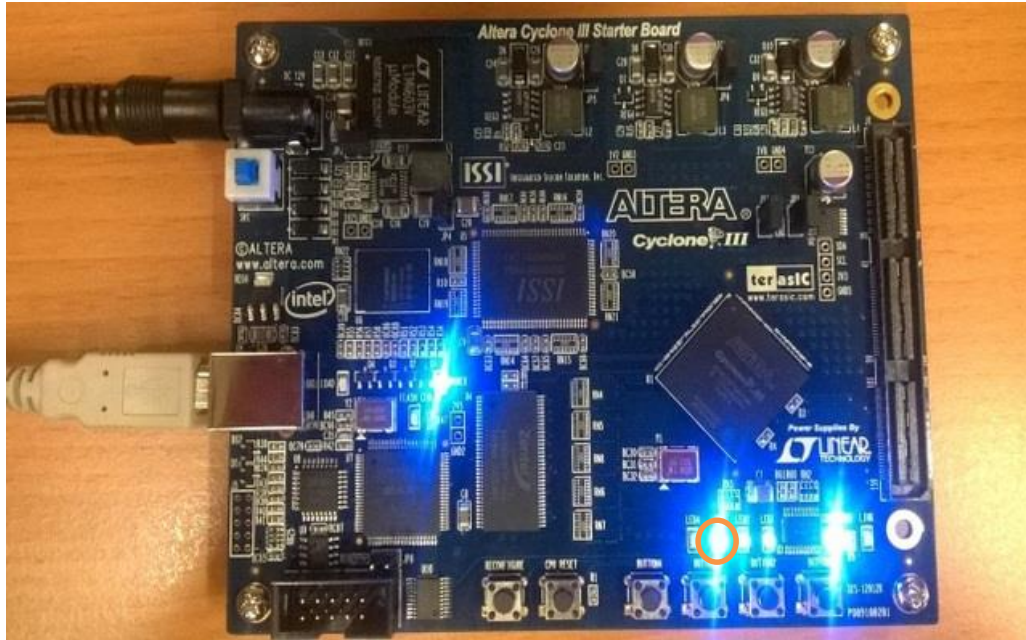
Figure 18: After loading receiver LFSR- No error - LED 3(circled in orange) is ON

The transmitted PRBS signal was assigned to one of the HSMC pins connected to the FPGA and received via another HSMC pin by externally wiring the two HSMC pins. The signals at the two pins were tracked using a Signal Tap II Logic Analyzer. Bit Error through this digital link was found to be zero, as a small piece of wire can produce almost zero error.

4.3.2. METHOD 2

In this method, 4 bits are sent as initialization bits to the receiver. Once the initialization bits are received correctly, a button is used to load the transmitter and receiver LFSR's seed value independently. Bit error counting begins thereafter.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH
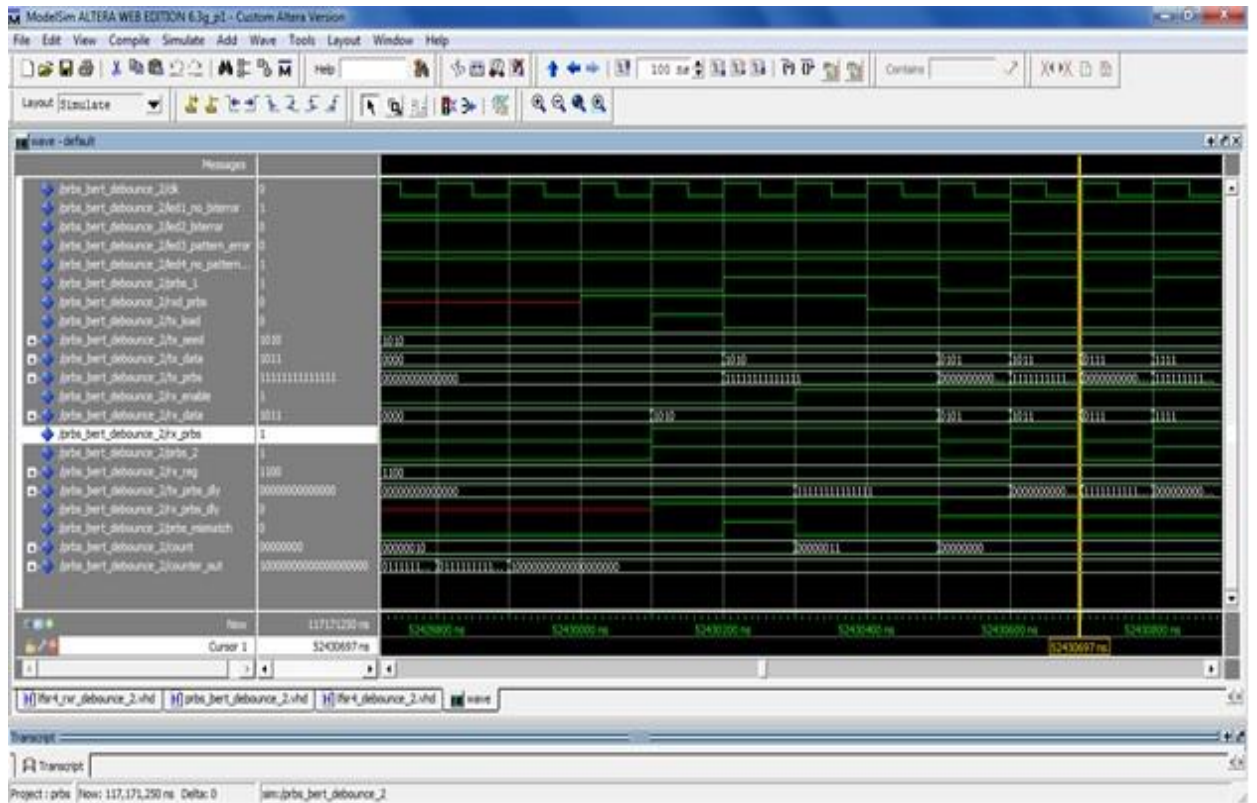
Figure 19: Method 2 - ModelSim Waveform

4.4. DAC-ADC INTERFACE

Figure 20: DAC-ADC LINK
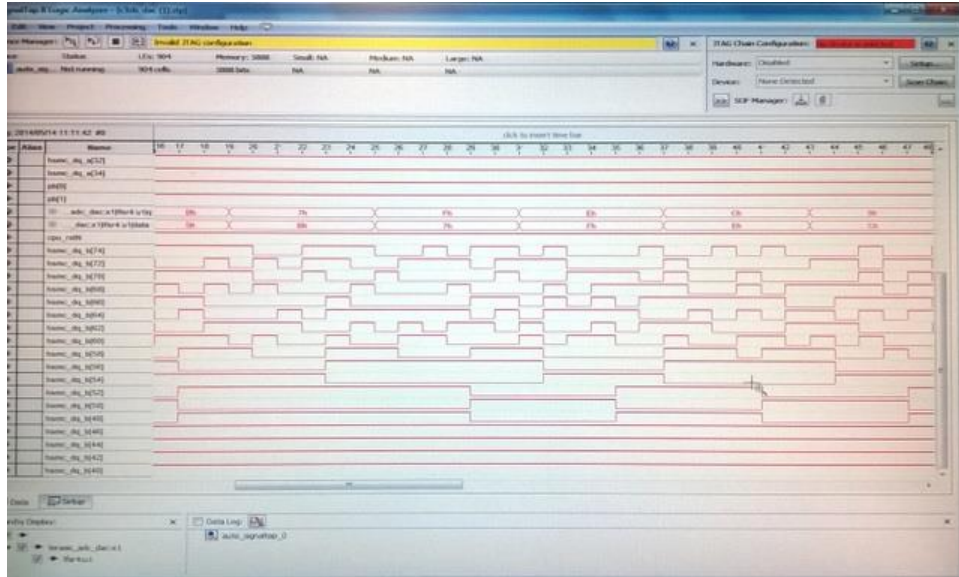
Figure 21: SignalTap II Logic Analyzer- ADC output for a PRBS signal

The DAC when directly connected to the ADC using an SMA connector, produced 14 bit output for the PRBS signal, as shown in Figure 21. These 14 bits are converted into either a '1' or a '0' by comparing the 14 bits with a reference value and checking if value is greater than or lesser than the reference value.

5. CONCLUSION AND FUTURE WORK

The hardware testing involved sending the PRBS signal through the DAC and passing it through a 25 km long fiber. The signal at the receiver end of the optical fiber was viewed on an oscilloscope and compared with the PRBS signal generated at the transmitter. There was considerable delay between the two signals, as the signal had to pass through a 25kms long fiber.

The overall Bit Error Rate Test for a digital link has been implemented and tested using Cyclone III FPGA Starter Kit, to show zero bit error as the channel does not introduce any kind of distortion. The DAC-ADC interface was verified by directly connecting the DAC and ADC using a Sub Miniature version A (SMA) connector and the ADC output was tracked using Signal Tap II Logic Analyzer.

The BERT test using Method 1 has been fully implemented and tested at the simulation level. The zero Bit error rate through a digital link, without the optical fiber, has been observed using the Cyclone III FPGA Starter Kit. In the method 1, the receiver LFSR's seed is loaded only after receiving it from the transmitter. The bit errors produced could be due to incorrect receiver clock phase, the channel, corrupted seed, etc. Once the bit error rate test starts, the receiver phase is changed. If the bit error reduces for a particular phase setting, then that phase setting with reduced BER is the correct receiver phase. The low BER at that point could be because of the channel. If the bit error does not reduce for any clock phase setting, then the high BER could be because of bad channel or because of corrupted seed. This can be analyzed with the help of an oscilloscope or spectrum analyzer by analyzing the initial few bits (seed). Using an oscilloscope, the traces of the received signal can be captured and recorded. This can be compared with the transmitted signal and it can be analyzed to see if the seed was corrupted.

Since in Method 1, there are possibilities of the getting corrupted before it is loaded into the receiver LFSR, Method 2 was proposed. In Method 2, the transmitter LFSR seed and the receiver LFSR seed are loaded independently. So, the Bit error that occurs may be due to the channel or receiver clock phase. Once the receiver clock phase is properly adjusted to receive the data, the Bit error that occurs may be mainly due to the channel. From a spectrum analyzer the noise and distortion characteristics can be studied.

It can be seen that each of the two methods pose a drawback. Method 1 one is time-consuming, as it does not guarantee that the transmitted seed will be right and hence extra steps have to be carried out to rule out all possibilities before narrowing down to the right cause of bit-error rate. In Method 2, the seed value cannot be changed each time. It's all a matter of the kind of trade-off that is acceptable for a particular application.

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

5.1. FUTURE WORK

Data transmission from one computer to another can be done using Forward Error Correction (FEC), wherein the data bits are transmitted along with additional bits called redundant bits. These redundant bits are used to correct the errors in the data (if any), using FEC coding designed for that particular application. Each of the computers could be connected to an FPGA which transmits and receives the data. Since FEC can be better implemented in MATLAB, due to the mathematical computations involved, HDL Coder can be used to convert the MATLAB code to its corresponding VHDL code.

The MATLAB and Simulink models for OFDM and Nyquist modulation schemes for optical communication can be implemented in FPGAs with the help of HDL Coder provided by MATLAB. The robustness of the two modulation techniques can then be studied and compared by performing a BERT in Real-Time, using an FPGA.

5.2. CHALLENGES FACED

Hardware implementation needs to ensure that everything is perfect in the software code too. While implementing the application using Cyclone III FPGA Starter Kit (main board), a few problems were encountered. After testing the FPGA with basic programs, the THDB_ADA daughter board was interfaced with the main board with the help of an on board High Speed Mezzanine Connector (HSMC). Precautions had to be taken to ensure the proper working of the FPGA before a daughter board is interfaced with the main board.

While checking the working of Transmitter and Receiver through a digital link without passing it through DAC-ADC, a problem was encountered. As few of the FPGA pins are connected to the HSMC pins, internally, wiring two bi-directional HSMC pins externally using a wire, can establish transmitter – receiver link. Such wiring could cause shorting of two pins when the point of contact is slightly deviated. The HSMC pins could be accessed externally by connecting a General Purpose Input Output (GPIO) onto the daughter board.

Precautions to be taken

- When the FPGA powers on, it loads a configuration from Flash. As there is no information as to how the configuration file uses the HSMC connector, there could be a possibility that it could be driving all the signals to ground or VCC. When the ADA board is plugged in, the ADC signals that were HIGH would create a driver conflict with the FPGA. If that was the case, then when the board is turned on, it might damage the FPGA and possibly the AD/DA board.

- A few of the port entities used in the configuration file for the Cyclone III Starter board drive some of the ADA signals on the HSMC connector. This could cause a bus conflict that could possibly damage the board. The HSMC connector interface must be defined as tri-state until it is decided on what is to be plugged in, otherwise there could be driver

conflict that could damage the board. Therefore, a known FPGA configuration is to be downloaded before plugging in the ADA board

- The ADA board should not be plugged in or removed from the Cyclone III board, while it is powered on. Accidental shorting of the pins on the HSMC connector by placing it on uneven, conductive surface could also cause a problem possibly damaging the board.

- The HSMC pins should be accessed only with the help of a GPIO breakout board. This would ensure that there is no accidental shorting of any of the HSMC pins and there is no connection with the power supply pins.

5.3. OTHER APPLICATIONS

1. When two different FPGAs are independently used as transceivers, each of the two FPGAs can be connected to a computer. The computer, with the help of specific software, can process the incoming packets and can check for the correctness of the data with the help of Cyclic Redundancy Check (CRC). The data can be music, if error is detected, the computer can either correct the data or send a message to the other computer to re-transmit the data. Here the co-ordination between the two FPGAs can be achieved using the Internet Protocol (IP) and Media Access Control (MAC) Ids of the two computers.

2. Data transfer testing for any kind of data like music, written document, etc. between the two ends of the system can be done in two ways-

    - The file is stored at the Transmitter and Receiver. The data is transmitted from one end which is received at the other end and compared with the data that is already stored at the receiver.

    - Cyclic redundancy check- an error detecting code, can be used at the receiver to check if the received packets are correctly received.

REFERENCES

1. Yannis Benlachtar, Philip M. Watts, Rachid Bouziane, Peter Milder, Robert Koutsoyannis, James C. Hoe, Markus Püschel, Madeleine Glick, and Robert I, "Real-Time Digital Signal Processing for the Generation of Optical Orthogonal Frequency Division Multiplexed Signals".

2. Agrawal: Govind P. Agrawal, "Fiber-optic communication systems", 4th Edition, Editorial: Wiley series in microwave and optical engineering, ISBN 978-0-470-50511-3.

3. D.W.Hawkins. Linear feedback shift-registers (LFSR) and Pseudo-random binary sequences (PRBS), California Institute of Technology.

4. Marius Strobl, Thomas Waas; Marcel Moolenaar; Angelika Schingale, Norbert Balbierer, "Bit Error Rate Testing Serial Communication Equipment using Pseudo-Random Bit Sequences".

5. Terasic, "THDB_ADA User Guide".

6. Sanad Bushnaq,Toru Nakura, Makoto Ikeda Kunihiro Asada, "All Digital Baseband 50 Mbps Data Recovery Using 5x Oversampling With 0.9 Data Unit Interval Clock Jitter Tolerance".

7. D. W. Hawkins, "Altera JTAG-to-Avalon-MM Tutorial", Version 1.0.

8. R. N. Mutagi, "Pseudo noise sequences for engineers", Electronics & Communications Engineering Journal.

9. Altera, "Cyclone III FPGA Starter Board Schematic".

10. Altera, "Cyclone III FPGA Starter Board Reference Manual".

11. Altera, "Cyclone III FPGA Starter Board User Guide".

12. Altera, "High Speed Mezzanine Card (HSMC) Standard"

13. Koji IGARASHI, Yojiro MORI, Kazuhiro KATOH, and Kazuro KIKUCHI "Bit-error Rate Performance of Nyquist Wavelength-Division Multiplexed Quadrature Phase-Shift Keying Optical Signals".

14. Analog Devices: AD9767: 14-Bit, 125 MSPS Dual TxDAC+ Digital-to-Analog Converter.

15. Analog Devices: AD9248: 14-Bit, 20 MSPS/40 MSPS/65 MSPS Dual A/D Converter.

APPENDIX

LIST OF ABBREVIATIONS

PRBS	Pseudo Random Bit Sequence
BER	Bit Error Rate
FPGA	Field Programmable Gate Array
MZM	Mach-Zehnder Modulator
HSMC	High Speed Mezzanine Connector
LFSR	Linear Feedback Shift Register
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
JTAG	Joint Test Action Group
TCL	Tool Command Language
VHDL	Very high speed integrated circuit (VHSIC) Hardware Description Language
DSP	Digital Signal Processing
SMA	Sub Miniature version A

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Cyclone III FPGA Starter Kit & THDB_ADA Daughter Board – Pin Mapping

Cyclone 3 Starter Board								AD/DA HSMC Daughter Board		
HSMC Connector (J1)	HSMC Net Name	VHDL Port name	Signal Direction	FPGA Pin Number	Pin Group/Function	VCCIO Bank		Non-Standard HSMC Connector (J	Schematic Net Name	I/O Direction
2										
4										
6										
8										
10										
12										
14										
16										
18										
20										
22										
24										
26										
28										
30										
32										
33	HSMC_SDA	hsmc_sda	bidirectional	E1				131	ID_I2CDAT	
34	HSMC_SCL	hsmc_scl	output	F3				132	ID_I2CSCL	
35	HSMC_TCK		not clr					129		
36	HSMC_TMS		not clr					130		
37	HSMC_TDO		not clr					123		
38	HSMC_TDI		not clr					124		
39	HSMC_CLKOUT0	hsmc_clkout[0]	input	A1	IO,	bank 8		121	POWER_ON	input to dac nd adc
40	HSMC_CLKIN0	hsmc_clkin[0]	output	A9	CLK10,	bank 8		122		
41	HSMC_D0	hsmc_dq_a(0)	bidirectional	H6	I/O	bank 1		119		
42	HSMC_D1	hsmc_dq_a(1)	bidirectional	D3	IO	bank 1		120		
43	HSMC_D2	hsmc_dq_a(2)	bidirectional	M5	IO	bank 2		117		
44	HSMC_D3	hsmc_dq_a(3)	bidirectional	L6	IO	bank 2		118		
45	3.3V							115	3.3V	
46	12V							116	12V	
47	HSMC_D4	hsmc_dq_a(4)	bidirectional	T1	IO RDN1	bank 2		113	DA_DA0	d/a bit frm host
48	HSMC_D5	hsmc_dq_a(5)	bidirectional	M3	IO	bank 2		114	DA_DB0	d/a bit frm host
49	HSMC_D6	hsmc_dq_a(6)	bidirectional	N7	IO DIFFIO_B4n	bank 3		111	DA_DA1	d/a bit frm host
50	HSMC_D7	hsmc_dq_a(7)	bidirectional	T2	IO RUP1	bank 2		112	DA_DB1	d/a bit bost
51	3.3V							109	3.3V	
52	12V							110	12V	
53	HSMC_D8	hsmc_dq_a(8)	bidirectional	N8	IO	bank 3		107	DA_DA2	d/a bit frm host
54	HSMC_D9	hsmc_dq_a(9)	bidirectional	H15	IO	bank 6		108	DA_DB2	d/a bit frm host
55	HSMC_D10	hsmc_dq_a(10)	bidirectional	J13	IO	bank 6		105	DA_DA3	
56	HSMC_D11	hsmc_dq_a(11)	bidirectional	H16	IO	bank 6		106	DA_DB3	d/a bit frm host
57	3.3V							103	3.3V	
58	12V							104	12V	
59	HSMC_D12	hsmc_dq_a(12)	bidirectional	N10	IO DIFFIO_B21p	bank 6		101	DA_DA4	d/a bit frm host

60	HSMC_D13	hsmc_dq_a(13)	bidirectional	N16	IO DIFFIO_R13n	bank 5	102	DA_DB4	d/a bit frm host
61	HSMC_D14	hsmc_dq_a(14)	bidirectional	N11	IO DIFFIO_B21n	bank 4	99	DA_DA5	d/a bit frm host
62	HSMC_D15	hsmc_dq_a(15)	bidirectional	N15	I/O	bank 5	100	DA_DB5	d/a bit frm host
63	3.3V						97	3.3V	
64	12V						98	12V	
65	HSMC_D16	hsmc_dq_a(16)	bidirectional	K17	IO DIFFIO_R7n	bank 5	95	DA_DA6	d/a bit frm host
66	HSMC_D17	hsmc_dq_a(17)	bidirectional	R16	IO RDN3	bank 5	96	DA_DB6	d/a bit frm host
67	HSMC_D18	hsmc_dq_a(18)	bidirectional	P11	IO DIFFIO_B17n	bank 4	93	DA_DA7	d/a bit frm host
68	HSMC_D19	hsmc_dq_a(19)	bidirectional	T16	IO RUP3	bank 5	94	DA_DB7	d/a bit frm host
69	3.3V						91	3.3V	
70	12V						92	12V	
71	HSMC_TX_p4	hsmc_dq_a(20)	bidirectional	B2	IO DIFFIO_L1p	bank 1	89	DA_DA8	d/a bit frm host
72	HSMC_RX_p4	hsmc_dq_a(21)	bidirectional	C2	IO DIFFIO_L3p	bank 1	90	DA_DB8	d/a bit frm host
73	HSMC_TX_n4	hsmc_dq_a(22)	bidirectional	B1	IO DIFFIO_L1n	bank 1	87	DA_DA9	d/a bit frm host
74	HSMC_RX_n4	hsmc_dq_a(23)	bidirectional	C1	IO DIFFIO_L3n DATA1, ASDO	bank 1	88	DA_DB9	d/a bit frm host
75	3.3V						85	3.3V	
76	12V						86	12V	
77	HSMC_TX_p5	hsmc_dq_a(24)	bidirectional	G2	IO DIFFIO_L5p	bank 1	83	DA_DA10	d/a bit frm host
78	HSMC_RX_p5	hsmc_dq_a(25)	bidirectional	H2	IO DIFFIO_L6p	bank 1	84	DA_DB10	d/a bit frm host
79	HSMC_TX_n5	hsmc_dq_a(26)	bidirectional	G1	IO DIFFIO_L5n	bank 1	81	DA_DA11	d/a bit frm host
80	HSMC_RX_n5	hsmc_dq_a(27)	bidirectional	H1	IO DIFFIO_L6n	bank 1	82	DA_DB11	d/a bit frm host
81	3.3V						79	3.3V	
82	12V						80	12V	
83	HSMC_TX_p6	hsmc_dq_a(28)	bidirectional	K2	IO DIFFIO_L7p	bank 2	77	DA_DA12	d/a bit frm host
84	HSMC_RX_p6	hsmc_dq_a(29)	bidirectional	K5	IO DIFFIO_L8p	bank 2	78	DA_DB12	d/a bit frm host
85	HSMC_TX_n6	hsmc_dq_a(30)	bidirectional	K1	IO DIFFIO_L7n	bank 2	75	DA_DA13	d/a bit frm host
86	HSMC_RX_n6	hsmc_dq_a(31)	bidirectional	L5	IO DIFFIO_L8n	bank 2	76	DA_DB13	d/a bit frm host
87	3.3V						73	3.3V	
88	12V						74	12V	

89	HSMC_TX_p7	hsmc_dq_a(32)	bidirectional	L2	IO DIFFIO_L3p	bank 2		71	DA_WRTA	input to dac
90	HSMC_RX_p7	hsmc_dq_a(33)	bidirectional	L4	IO DIFFIO_L10p	bank 2		72	DA_WRTB	input to dac
91	HSMC_TX_n7	hsmc_dq_a(34)	bidirectional	L1	IO DIFFIO_L9n	bank 2		69	DA_MODE	input to dac
92	HSMC_RX_n7	hsmc_dq_a(35)	bidirectional	L3	IO DIFFIO_L10n	bank 2		70		
93	3.3V							67	3.3V	
94	12V							68	12V	
95	HSMC_CLKOUT_p1	hsmc_clkout[1]	input to hsmc	D14	IO DIFFIO_T24n	bank 7		65	PLL_OUT_DAC1	input to dac
96	HSMC_CLKIN_p1	hsmc_clkin[1]	output from hsmc	F17	CLK4 DIFFCLK_2p	bank 6		66	OSC_SMA_ADC4	input to dac
97	HSMC_CLKOUT_n1	hsmc_clkout[3]	input to hsmc	C14	IO DIFFIO_T24p	bank 7		63	PLL_OUT_DAC0	input to dac
98	HSMC_CLKIN_n1	hsmc_clkin[3])	output from hsmc	F18	CLK5 DIFFCLK_2n	bank 6		64	SMA_DAC4	input to dac
99	3.3V							61	3.3V	
100	12V							62	12V	
101	HSMC_TX_p8	hsmc_dq_b(40)	bidirectional	M2	IO DIFFIO_L11p	bank 2		59		
102	HSMC_RX_p8	hsmc_dq_b(41)	bidirectional	P2	IO DIFFIO_L12p	bank 2		60		
103	HSMC_TX_n8	hsmc_dq_b(42)	bidirectional	M1	IO DIFFIO_L11n	bank 2		57		
104	HSMC_RX_n8	hsmc_dq_b(43)	bidirectional	P1	IO DIFFIO_L12n	bank 2		58		
105	3.3V							55	3.3V	
106	12V							56	12V	
107	HSMC_TX_p9	hsmc_dq_b(44)	bidirectional	R2	IO DIFFIO_L13p	bank 2		53		
108	HSMC_RX_p9	hsmc_dq_b(45)	bidirectional	T3	IO DIFFIO_L14p	bank 2		54	ADC_OEA	a/d ouptut enable frm host
109	HSMC_TX_n9	hsmc_dq_b(46)	bidirectional	R1	IO DIFFIO_L13n	bank 2		51		
110	HSMC_RX_n9	hsmc_dq_b(47)	bidirectional	R3	IO DIFFIO_L14n	bank 2		52	ADC_OEB	a/d output enable frm host
111	3.3V							49	3.3V	
112	12V							50	12V	
113	HSMC_TX_p10	hsmc_dq_b(48)	bidirectional	E17	IO DIFFIO_R4p	bank 6		47	AD_DB13	a/d bit to host
114	HSMC_RX_p10	hsmc_dq_b(49)	bidirectional	G17	IO DIFFIO_R5p CRC_ERROR	bank 6		48	AD_DA13	a/d bit to host
115	HSMC_TX_n10	hsmc_dq_b(50)	bidirectional	E18	IO DIFFIO_R4n Nceo	bank 6		45	AD_DB12	a/d bit to host

116	HSMC_RX_n10	hsmc_dq_b(51)	bidirectional	G18	IO DIFFIO_R5n INIT_DONE	bank 6	46	AD_DA12	a/d bit to host
117	3.3V						43	3.3V	
118	12V						44	12V	
119	HSMC_TX_p11	hsmc_dq_b(52)	bidirectional	H17	IO DIFFIO_R6p	bank 6	41	AD_DB11	a/d bit to host
120	HSMC_RX_p11	hsmc_dq_b(53)	bidirectional	K18	IO DIFFIO_R8p	bank 5	42	AD_DA11	a/d bit to host
121	HSMC_TX_n11	hsmc_dq_b(54)	bidirectional	H18	IO DIFFIO_R6n	bank 6	39	AD_DB10	a/d bit to host
122	HSMC_RX_n11	hsmc_dq_b(55)	bidirectional	L18	IO DIFFIO_R8n	bank 5	40	AD_DA10	a/d bit to host
123	3.3V						37	3.3V	
124	12V						38	12V	
125	HSMC_TX_p12	hsmc_dq_b(56)	bidirectional	L17	IO DIFFIO_R9p DEV_CLRn	bank 5	35	AD_DB9	a/d bit to host
126	HSMC_RX_p12	hsmc_dq_b(57)	bidirectional	L16	IO DIFFIO_R10p	bank 5	36	AD_DA9	a/d bit to host
127	HSMC_TX_n12	hsmc_dq_b(58)	bidirectional	M18	IO DIFFIO_R9n DEV_OE	bank 5	33	AD_DB8	a/d bit to host
128	HSMC_RX_n12	hsmc_dq_b(59)	bidirectional	M17	IO DIFFIO_R10n	bank 5	34	AD_DA8	a/d bit to host
129	3.3V						31	3.3V	
130	12V						32	12V	
131	HSMC_TX_p13	hsmc_dq_b(60)	bidirectional	L14	IO DIFFIO_R11p	bank 5	29	AD_DB7	a/d bit to host
132	HSMC_RX_p13	hsmc_dq_b(61)	bidirectional	L13	IO DIFFIO_R12p	bank 5	30	AD_DA7	a/d bit to host
133	HSMC_TX_n13	hsmc_dq_b(62)	bidirectional	L15	IO DIFFIO_R11n	bank 5	27	AD_DB6	a/d bit to host
134	HSMC_RX_n13	hsmc_dq_b(63)	bidirectional	M14	IO DIFFIO_R12n	bank 5	28	AD_DA6	a/d bit to host
135	3.3V						25	3.3V	
136	12V						26	12V	
137	HSMC_TX_p14	hsmc_dq_b(64)	bidirectional	P17	IO DIFFIO_R13p	bank 5	23	AD_DB5	a/d bit to host
138	HSMC_RX_p14	hsmc_dq_b(65)	bidirectional	R17	IO DIFFIO_R14p	bank 5	24	AD_DA5	a/d bit to host
139	HSMC_TX_n14	hsmc_dq_b(66)	bidirectional	P18	IO DIFFIO_R13n	bank 5	21	AD_DB4	a/d bit to host
140	HSMC_RX_n14	hsmc_dq_b(67)	bidirectional	R18	IO DIFFIO_R14n	bank 5	22	AD_DA4	a/d bit to host
141	3.3V						19	3.3V	
142	12V						20	12V	
143	HSMC_TX_p15	hsmc_dq_b(68)	bidirectional	R5	IO DIFFIO_L15p	bank 2	17	AD_DB3	a/d bit to host

144	HSMC_RX_p15	hsmc_dq_b(69)	bidirectional	M6	IO DIFFIO_B2p	bank 3		18	AD_DA3	a/d bit to host
145	HSMC_TX_n15	hsmc_dq_b(70)	bidirectional	R4	IO DIFFIO_L15n	bank 2		15	AD_DB2	a/d bit to host
146	HSMC_RX_n15	hsmc_dq_b(71)	bidirectional	N6	IO DIFFIO_B2n	bank 3		16	AD_DA2	a/d bit to host
147	3.3V							13	3.3V	
148	12V							14	12V	
149	HSMC_TX_p16	hsmc_dq_b(72)	bidirectional	T17	IO DIFFIO_R15p	bank 5		11	AD_DB1	a/d bit to host
150	HSMC_RX_p16	hsmc_dq_b(73)	bidirectional	M13	IO DIFFIO_B25p	bank 4		12	AD_DA1	a/d bit to host
151	HSMC_TX_n16	hsmc_dq_b(74)	bidirectional	T18	IO DIFFIO_R15n	bank 5		9	AD_DB0	a/d bit to host
152	HSMC_RX_n16	hsmc_dq_b(75)	bidirectional	N13	IO DIFFIO_B25n	bank 4		10	AD_DA0	a/d bit to host
153	3.3V							7	3.3V	
154	12V							8	12V	
155	HSMC_CLKOUT_p2	hsmc_clkout[2]	input	U18	IO PLL4_CLKOUTp	bank 4		5	PLL_OUT_ADC1	output frm adc
156	HSMC_CLKIN_p2	hsmc_clkin[2]	output	N17	CLK6 DIFFCLK_3p	bank 5		6	AD_OTRB	output from adc ic
157	HSMC_CLKOUT_n2	hsmc_clkout[4]	input	V18	IO PLL4_CLKOUTn	bank 4		3	PLL_OUT_ADC0	output frm adc
158	HSMC_CLKIN_n2	hsmc_clkin[4]	output	N18	CLK7 DIFFCLK_3n	bank 5		4	AD_OTRA	output from adc ic
159	3.3V							1	3.3V	
160	HSMC_PSTN_N							2	GND	
161	GND							192	GND	
162	GND							191	GND	
163	GND							190	GND	
164	GND							189	GND	
165	GND							188	GND	
166	GND							187	GND	
167	GND							186	GND	
168	GND							185	GND	
169	GND							184	GND	
170	GND							183	GND	
171	GND							182	GND	
172	GND							181	GND	

No Matching Pins on Standard HSMC	
125	TDO_TDI
126	TDO_TDI