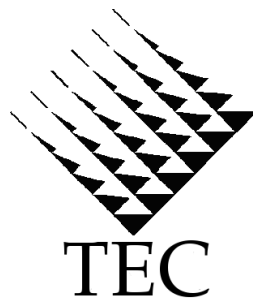


Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica



**Experimental measurements for Free Space Optical Links
characterization**

**Informe de Proyecto de Graduación para optar por el título
de Ingeniero en Electrónica con el grado académico de
Licenciatura**


Brian White Hernández

Turin, Italia, Noviembre, 2007

INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE INGENIERIA ELECTRONICA
PROYECTO DE GRADUACIÓN
TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal



Ing. Gabriela Ortiz León

Profesor lector



Ing. William Marín Moreno

Profesor lector



Ing. Anibal Coto Cortes

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 23 de Enero del 2008

Authenticity Declaration

Hereby I declare that this graduation project is made by myself using literature related to the subject and applying my own ideas and knowledge to its proper fulfillment.

If I'll need to cite information from literature or the internet I'll proceed by properly indicate the sources using bibliographical quotes.

By consequence I assume complete responsibility for all the information present in this graduation project and the corresponding documentation.

Torino, February 26 of 2007



Brian White Hernández
Id: 1-1129-0802

Abstract

The project described consists on the development of a compact size Bit Error Rate Tester(BERT), for the Free Space Optical(FSO) communications system that is being developed on the photonics laboratory of Politecnico di Torino(Photonlab), located in Turin, Italy.

The BERT was based on a Xilinx Virtex II-PROTM, this FPGA was chosen because of the high speed serializer interface, that allowed the implementation of a 2.5Gb/s pattern generator and error detector.

The software was programmed in Microsoft© Visual C#, to implement the interface with the user, also it's capable of creating a .log file with the BER test results in order to perform a post processing analysis of the system's BER evolution in time.

Keywords:

- RocketIO.
- BERT.
- Virtex II-PRO.
- PRBS.
- Pattern generator.
- Visual C#.
- Free space optical communications.

Dedicatoria

A mis padres Charles y Carmen que siempre han estado para brindarme su apoyo y amor; y que a lo largo de toda mi vida de estudiante me han permitido tener la fuerza para continuar.

A mis hermanos:

Priscilla, por el gran ejemplo de lucha y valor que siempre la ha caracterizado. Por que gracias a ello encuentre el valor para afrontar este desafío.

Christian y Angela, por darme el impulso que necesite para tomar la decisión de partir y vivir esta experiencia extraordinaria.

Finalmente a mis sobrinos Manfred, Nathalie y Kirsten, por que a travez de ellos veo la ilusión para el futuro.

Les dedico mi tesis de graduación con todo el cariño. Muchas gracias por todo.

Acknowledgments

I want to thank all of the staff at Photonlab for the opportunity to develop this thesis as part of their Project ASI. In particular, I want to express my gratitude to the following people:

- Professor Valter Ferrero.
- Dr. Stefano Camatel.
- Eng. Ramon Mata.

Also I want to thank Miss Carla Ghiani and her family, Maria, Patrizia, Federica, Giorgio, Atilio, Mattia and Angela Morbello. For letting me be a part of their family.

Finally, I want to thank God for all the gifts, for giving me an amazing family and extraordinary friends.

Contents

| | |
|---|-----------|
| Chapter 1: Preface | 15 |
| 1.1 Existing problem and solution's relevance | 15 |
| 1.1.1 Project surroundings | 15 |
| 1.1.2 General aspects | 18 |
| 1.1.3 Problem synthesis | 19 |
| 1.2 Solution selected | 19 |
| Chapter 2: Goal and objectives | 20 |
| 2.1 Goal | 20 |
| 2.2 General objective | 20 |
| 2.3 Specific objectives: | 20 |
| Chapter 3: Theoretical background | 21 |
| 3.1 System description | 21 |
| 3.2 Existing literature | 22 |
| 3.3 Electronics and physical principles description | 23 |
| 3.3.1 Pseudorandomness | 23 |
| 3.3.2 Pseudorandom binary sequence | 23 |

| | | |
|---|---|-----------|
| 3.3.3 | PRBS generation | 24 |
| 3.3.4 | High speed PRBS generation | 25 |
| 3.3.5 | Confidence level | 27 |
| Chapter 4: Methodological procedures | | 29 |
| 4.1 | Problem definition | 29 |
| 4.2 | Compilation and analysis of information | 29 |
| 4.3 | Solution Synthesis | 29 |
| 4.4 | Solution Implementation | 30 |
| Chapter 5: Solution description | | 33 |
| 5.1 | Solution analysis and final selection | 33 |
| 5.2 | Hardware description | 35 |
| 5.2.1 | First level diagram | 35 |
| 5.2.2 | Second level diagram | 36 |
| 5.2.3 | Third level diagram | 48 |
| 5.2.4 | Fourth level diagram: | 53 |
| 5.3 | Software description | 59 |
| Chapter 6: Results analysis | | 62 |
| 6.1 | Results | 62 |
| 6.1.1 | First test results | 62 |

| | | |
|---|---|-----------|
| 6.1.2 | Second test results | 65 |
| 6.1.3 | Third test results(FPGA Loopback): | 67 |
| 6.1.4 | Fourth test results(Connected with BERT Anritsu commercial equipment): | 73 |
| 6.2 | Analysis | 78 |
| 6.2.1 | Design phase | 78 |
| 6.2.2 | Test phase | 78 |
| Chapter 7: Conclusions and recommendations | | 81 |
| 7.1 | Conclusions | 81 |
| 7.2 | Recommendations | 81 |
| Bibliography | | 82 |
| Appendix | | 83 |
| A.1 | Glossary, abbreviations and symbols | 83 |
| A.2 | Measuring protocols | 84 |
| A.2.1 | First test: | 84 |
| A.2.2 | Second test: | 84 |
| A.2.3 | Third test: | 85 |
| A.2.4 | Fourth test: | 85 |
| A.3 | Maximum gating time calculus | 86 |
| A.4 | Test duration to guarantee a confidence level higher than 90% | 86 |

A.5 Serial port thread 88

List of Figures

| | | |
|------|--|----|
| 1.1 | Possible FSO applications. | 16 |
| 3.1 | Basic FSO system diagram. | 21 |
| 3.2 | Autocorrelation comparison between PRBS sequence and regular bit sequence. | 22 |
| 3.3 | PRBS generator. | 24 |
| 3.4 | Multirate PRBS generator. | 26 |
| 5.1 | BER tester first level diagram. | 36 |
| 5.2 | Serial communication second level diagram. | 37 |
| 5.3 | Data package sent to the PC. | 38 |
| 5.4 | Control logic block diagram. | 39 |
| 5.5 | Programmable frequency block diagram. | 42 |
| 5.6 | RocketIO TM transceiver block diagram. | 43 |
| 5.7 | RocketIO TM transmission pattern 8B/10B bypassed. | 44 |
| 5.8 | RocketIO TM transmission pattern 8B/10B bypassed. | 44 |
| 5.9 | Pattern generator's second level block diagram. | 45 |
| 5.10 | Error detector's second level block diagram. | 47 |
| 5.11 | Clock generator's third level block diagram. | 48 |

| | | |
|------|---|----|
| 5.12 | Serial communications hardware third level block diagram. | 49 |
| 5.13 | Error detector control machine's state diagram. | 50 |
| 5.14 | Pattern generator control machine's state diagram. | 51 |
| 5.15 | Pattern generator hardware third level block diagram. | 52 |
| 5.16 | Clock generator's hardware. | 53 |
| 5.17 | RX control machine's state diagram. | 54 |
| 5.18 | Normal implementation of the PRBS $2^7 - 1$ | 55 |
| 5.19 | Final implementation of the PRBS $2^7 - 1$ | 55 |
| 5.20 | Bit counter hardware. | 56 |
| 5.21 | Error counter hardware. | 57 |
| 5.22 | Bit to bit adder hardware. | 58 |
| 5.23 | Graphic interface main window. | 60 |
| 5.24 | Changes to the error panel when error threshold gating is selected. | 61 |
| 5.25 | Results visualization window. | 61 |
| 6.1 | Eye diagram for the RocketIO working at 625Mb/s. | 63 |
| 6.2 | Eye diagram for the RocketIO working at 1Gb/s. | 63 |
| 6.3 | Eye diagram for the RocketIO working at 2.5Gb/s. | 64 |
| 6.4 | Eye diagram for the RocketIO working at 3.125Gb/s. | 64 |
| 6.5 | Eye diagram for the improved signal working at 625Mb/s. | 65 |
| 6.6 | Eye diagram for the improved signal working at 1Gb/s. | 66 |

| | | |
|------|--|----|
| 6.7 | Eye diagram for the improved signal working at 2.5Gb/s. | 66 |
| 6.8 | PRBS2 ⁷ – 1 loopback test results. | 67 |
| 6.9 | PRBS2 ¹⁵ – 1 loopback test results with 1e-9 error insertion ratio. | 68 |
| 6.10 | PRBS2 ²⁹ – 1 loopback test results. | 69 |
| 6.11 | PRBS2 ²⁹ – 1 BER vs Time. | 69 |
| 6.12 | PRBS2 ²⁹ – 1 loopback test results, sharing clock generator. | 70 |
| 6.13 | PRBS2 ²⁹ – 1 BER vs Time, sharing clock generator. | 70 |
| 6.14 | PRBS2 ²⁹ – 1 loopback test results, changing RocketIO channel. | 71 |
| 6.15 | PRBS2 ²⁹ – 1 BER vs Time, changing RocketIO channel. | 71 |
| 6.16 | PRBS2 ³¹ – 1 loopback test results, when using external CDR. | 72 |
| 6.17 | PRBS2 ³¹ – 1 BER vs Time, when using external CDR. | 72 |
| 6.18 | Error free interval measured using Anritsu’s error detector. | 73 |
| 6.19 | BER measured using Anritsu’s error detector. | 74 |
| 6.20 | BER measured using the FPGA’s error detector. | 74 |
| 6.21 | Bitrate measured using Anritsu’s error detector. | 75 |
| 6.22 | Number of errors measured using Anritsu’s error detector. | 76 |
| 6.23 | BER measured using Anritsu’s error detector. | 76 |
| 6.24 | BER measured using the FPGA’s error detector. | 77 |
| A.1 | First test setup. | 84 |
| A.2 | Second test setup. | 84 |

A.3 Third test setup. 85

A.4 Fourth test setup. 85

List of Tables

| | | |
|-----|--|----|
| 3.1 | PRBS polynomials implemented on XAPP661 | 25 |
| 5.1 | Comparison between Xilinx and Avnet development boards | 34 |
| 5.2 | Control logic pinout description | 41 |
| 5.3 | Register address and data width | 59 |
| 6.1 | First test results | 62 |
| 6.2 | Second test results | 65 |

Chapter 1

Preface

1.1 Existing problem and solution's relevance

1.1.1 Project surroundings

The Photonlab is an experimental facility of Politecnico di Torino and Istituto Superiore Mario Boella, located in Torino, Italy. It's a 250 square meter laboratory that has an experimentation area for optical systems and networking, plus a class 1000 clean room, electronics lab, a machine shop and a dark room. This lab is focused on research and development of new technologies in the photonics field.

One of the current research projects of Photonlab consists in studying the possibility of implementing Optical Payload for Free Space Optical communications systems (FSO) in different link scenarios, this means to be able to establish an optical link between two points without the use of any waveguide (such as optic fiber). This project is assigned to a research group lead by of Professor Valter Ferrero from Politecnico di Torino.

The research project is focused on the application of FSO for the following links:

- ISL (Intersatellite Link) LEO (Low Earth Orbit) to LEO.
- Link LEO to Ground.
- Link Stratospheric Aircraft to Ground.
- Link Stratospheric Aircraft to LEO.

Low Earth Orbit refers to the orbit located between 200km and 2000km above the Earth's surface, while the stratospheric aircraft would be flying in the stratosphere which is defined between 10km and 50km above the Earth's surface. On figure 1.1 it's possible to see a representation of the possible links.

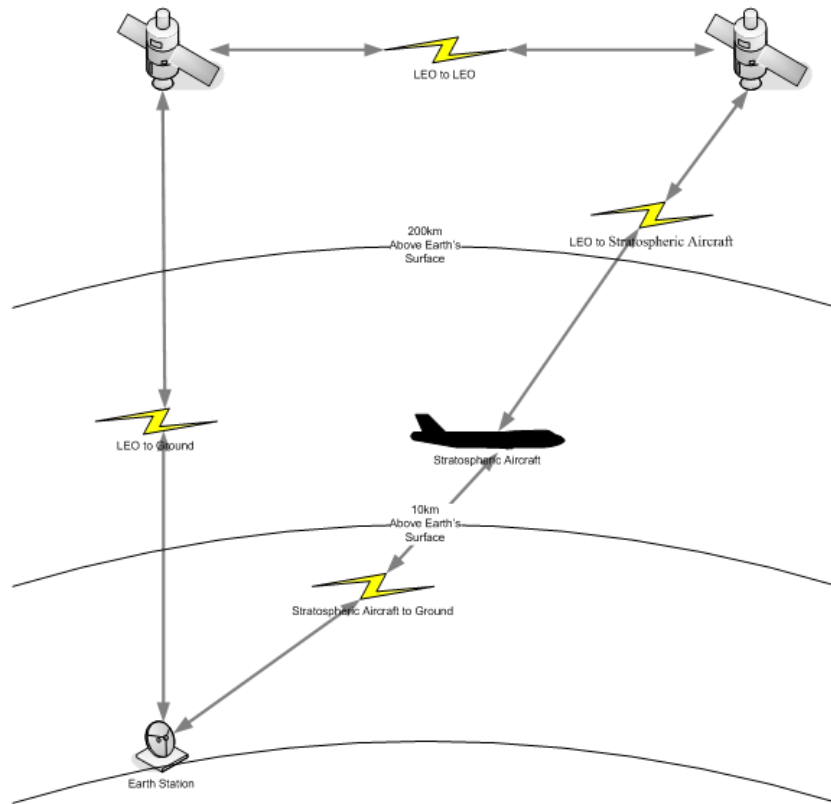


Figure 1.1: Possible FSO applications.

The FSO system is very similar to the radiofrequency communications system (RF) regarding system architecture and physical parameters, the difference is that while the FSO use a beam of light to establish the link the RF system use electromagnetic waves. The propagation characteristics of these two means define the application field and mission scenario of the two communication systems. In FSO systems is not possible to perform a broadcast, its main application is point to point communication.

The relative small beam aperture and the high gain of the antennas (implemented with a telescope) in the FSO, causes that less power is required in the transmitter to obtain the same receiver power (compared to RF systems). However the telescopes often require the implementation of an aiming system which increases the complexity of the FSO.

A comparison between RF communication system and an equivalent FSO system, both thought for free space applications, reveals the following advantages of laser communication over radiofrequency:

- Smaller antenna size (in the FSO case, the telescope is used as an antenna and requires less space than its counterpart in RF systems).
- Lower weight.
- Lower power consumption.
- The small Field Of View (FOV) makes the link immune to interception.
- Larger bandwidth (and consequently higher bit rates) compared to the RF systems.
- Immunity to electromagnetic interference.

The major disadvantage is the technological complexity since the RF communications are proven and consolidated systems while the FSO is a complete new application.

The Bit Error Rate (also known as BER or Bit Error Ratio) is the relation between the number of bits incorrectly received and the total number of bits received, however it's impossible to check the BER continuously on normal operating conditions, because the information transmitted would need to be on the receiver for comparison purposes, plus it'll not allow to use of the channel for transmitting information. That's why it's necessary to make a test with a fixed duration and to extrapolate the BER measurement relying on the probability. The duration of this test will depend on the link transmission speed because faster bit rates will allow collecting the data for the extrapolation more quickly.

1.1.2 General aspects

The main problems when establishing an optical link are the acquisition, tracking, and atmospheric turbulence effects such as scintillation that have a strong impact in the communication link. In fact, when light travels through a turbulent atmosphere the optical field becomes blurred or distorted. The investigation is in the phase of implementing devices that can compensate those effects and eventually allow establishing a reliable communication link. This raises one important question which is how to measure the reliability of the link, the answer is the BER, because it allows to know how much of the information transmitted is decoded wrongfully, and if this amount of information is below certain limit the reliability of the link would be assured.

There are commercial BER testers available in the market that could be used for this purpose; however the main limitations when using this type of equipment are the size and weight. A typical BERT consists of a pattern generator which measures 424.5mm width, 190.5mm height and 500mm depth, and the error detector that measures 424.5mm width, 221.5mm height and 580mm depth. These two units have a combined weight of 30kg. This, of course, are rough figures, the dimensions and weight could differ from models and manufacturers however it's important to notice that BERT's aren't portable units.

Besides on future stages of the project would be necessary to install the BERT on the airplane, where the space available for this equipment will be very reduced and a commercial one wouldn't fit. This means that one important task is to determine if it's possible to implement a compact BERT (bit error rate tester) that would fulfill the application requirements.

One of the considerations to determine the viability of the BERT implementation is that the bit rate of the link will vary from 700Mb/s up to 2.5Gb/s approximately, this implies that any device used for the calculation of the BER needs to be able to process the information fast enough. Also in order to develop a reliable test it's necessary to implement Pseudo Random Bit Sequence or PRBS, to stress the optical link in a proper way (the PRBS is the best way to emulate real information traffic on a link).

The PRBS is random because the value of a particular element is independent of the values of the other elements, similar to a real random sequence; however it's called pseudo because it's a deterministic function that repeats the same pattern after N elements (the PRBS are explained with more detail on Chapter 3). If the receiver and the transmitter have the same seed (initial value) and use the same pseudo random pattern algorithm they'll create the identical sequence which is a very important aspect for the development of this project.

All the previous aspects have to be taken into account to solve the main problem which is to know if the FSO link is reliable enough to transfer meaningful information.

1.1.3 Problem synthesis

Determine the reliability of a free space optical link between two points.

1.2 Solution selected

The solution selected consist in the development of a compact BER tester, based on an FPGA, that fulfills the project requirements that will be explained in the following chapters.

This FPGA will generate the bit patterns to stress the optical link and check the number of errors received. Obviously as the communication is bidirectional a single FPGA must have a pattern generator and error detector that must be independent from each other.

The BERT will be connected to the system designed by the research group to validate the reliability of the link.

Chapter 2

Goal and objectives

2.1 Goal

Determine the viability of implementing free space optical communication in different links scenarios.

2.2 General objective

Be able to measure the B.E.R. of the FSO communication system.

Indicator: Results obtained from BER test in different atmospheric conditions.

2.3 Specific objectives:

- Accomplish the coupling between the BERT designed and a commercial one.
Indicator: Its possible to accomplish an error free test between the FPGA and the commercial BERT?
- Accomplish a BER test at a bitrate ranging from 700Mb/s to 2.5Gb/s with a confidence level higher than 90%.
Indicator: theoretical confidence level of the test based on its duration.
- Control the BER test parameters from a computer.
Indicator: It's possible to control the BERT parameter using a computer program?

Chapter 3

Theoretical background

3.1 System description

The free space optical communications system consist on two electronics systems that work independently, one for signal processing and the other for telescope alignment.

The signal processing system converts the electrical signal into an optical signal and feeds it to a telescope, on the receiver side the optical signal passes through another telescope, and is processed to recover the transmitted data, and regenerate the electrical signal.

The aiming system is in charge of aligning the telescopes and keep them aligned during the transmission. A basic diagram of the system is showed on figure 3.1.

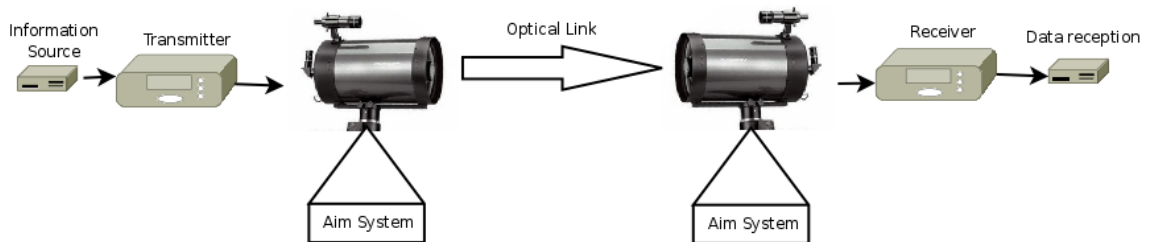


Figure 3.1: Basic FSO system diagram.

As it was said before, this system is in the development phase and it's necessary to execute a series of tests to evaluate the proper function of it.

The test methodology consists on transmitting a known data sequence and verify it on the receiver side. However it's very important that this data emulates random traffic(exhibits good autocorrelation¹) to properly test the system's dynamic, this is accomplish using a Pseudo Random Bit Sequence (PRBS) generator.

A PRBS is a bit sequence that presents randomness properties, however this signal repeats itself periodically after a finite number of bits(depending on the PRBS length), which means that it's a deterministic function.

Figure 3.2 ² presents a graphical representation of the autocorrelation for a PRBS sequence(a) and a not PRBS sequence(b).

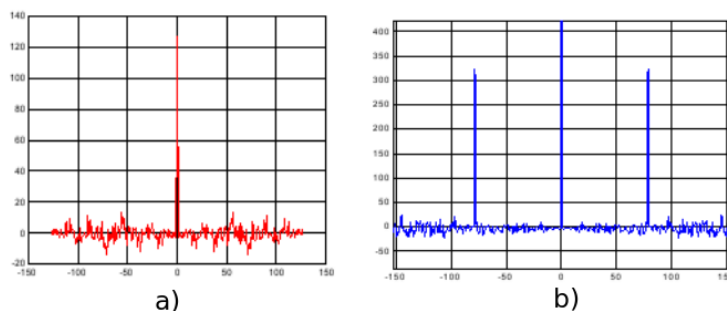


Figure 3.2: Autocorrelation comparison between PRBS sequence and regular bit sequence.

3.2 Existing literature

There is plenty of literature that talks about the implementation of the PRBS sequences and BER tester. Actually the BER testers are very important in the telecommunications industry, as several manufacturers(Agilent, Tektroniks, Hewlett Packard, Ixia, among others) create their own implementation, it was necessary to define an standard for the test pattern. This standard is described by the International Telecommunication Union(ITU), specifically on document ITU-T recommendation[5], and ITU-T recommendation O.150 - Corrigendum[6].

¹Autocorrelation measures how well a discrete signal can differentiate to a time shift variant of itself.

²Taken from [1], *40 Gbit/s electrical time domain pattern generator and BER measurements*

XilinxTM corporation on application note XAPP661[10] describes the use of a Virtex-II Pro to implement a simple BER tester. Also the application note XAPP211[9], explains the implementation of the PRBS generators using the shift register macro.

3.3 Electronics and physical principles description

3.3.1 Pseudorandomness

Pseudorandomness refers to a process that exhibits statistical randomness despite of being generated by a deterministic causal process.

Randomness refers to a process that cannot be predicted. In statistics this term is used to infer the lack of correlation between the members of the process.

To this date there isn't any algorithm to produce true random sequences because if there is any possibility to predict the outcome, the process is not random. A pseudorandom process by the other hand has the advantage that can be used several times producing exactly the same number sequence while the numbers sequence are not correlated to each other. These characteristics make the pseudorandom process an useful tool for testing the communications equipment, as it will emulate realistic data transmission.

3.3.2 Pseudorandom binary sequence

A PseudoRandom Binary Sequence(PRBS) refers to a binary sequence composed by N number of elements that are independent of the values of any of the other elements, this means that the autocorrelation($C(v)$) function given by equation 3.1.

$$C(v) = \sum_{n=0}^N a_n \cdot a_{n+v} \quad (3.1)$$

Where N is the number of bits(referred from now on as PRBS length). Has only two possible values:

$$C(v) = \begin{cases} m, & \text{if } v \equiv 0 \\ mc, & \text{otherwise} \end{cases}$$

where m is the number of ones in the sequence and c is the PRBS duty cycle given by equation 3.2.

$$c = \frac{m - 1}{N - 1} \quad (3.2)$$

3.3.3 PRBS generation

The PRBS generation is usually implemented using a series of shift registers with the appropriated feedback, according to the length of the sequence. This feedback is implemented using the XOR function of some of the values stored on the shift registers. Figure 3.3³ shows the implementation of the PRBS generator, where n is the number of shift registers needed to implement a PRBS of length equal to $2^n - 1$.

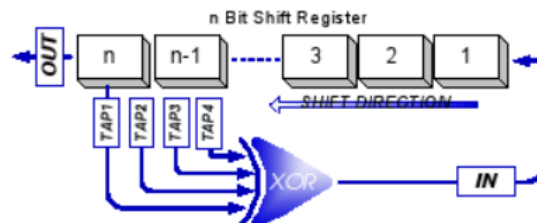


Figure 3.3: PRBS generator.

The feedback function is represented as a polynomial $x^i + x^j + \dots + 1$ where the exponential values represent the taps position on the shift register, table 3.1 shows the polynomials that are used in Xilinx XAPP661[10].

³Taken from: [4], *Pattern Creator/Converter Software User Manual*

Table 3.1: PRBS polynomials implemented on XAPP661

| PRBS length | Polynomial |
|--------------|-----------------------|
| $2^7 - 1$ | $x^7 + x^6 + 1$ |
| $2^9 - 1$ | $x^9 + x^5 + 1$ |
| $2^{10} - 1$ | $x^{10} + x^7 + 1$ |
| $2^{11} - 1$ | $x^{11} + x^9 + 1$ |
| $2^{15} - 1$ | $x^{15} + x^{14} + 1$ |
| $2^{20} - 1$ | $x^{20} + x^3 + 1$ |
| $2^{23} - 1$ | $x^{23} + x^{18} + 1$ |
| $2^{29} - 1$ | $x^{29} + x^{27} + 1$ |
| $2^{31} - 1$ | $x^{31} + x^{28} + 1$ |

This implementation, despite of its practicality and simplicity, becomes inefficient when working at high bit rate. This because the propagation delay of the XOR gate, defines the maximum operation frequency of the PRBS generator.

3.3.4 High speed PRBS generation

With the problem of the propagation delay in mind a new technique was developed based on the original PRBS generator. The main idea is that a PRBS can be generated by using several PRBS that work at a lower frequency. On electronic terms this means that several PRBS generators are multiplexed as showed on figure 3.4. It's very important that the single PRBS blocks have the appropriate delay between each other, to ensure that the multiplexer's output is a PRBS.

The delays(expressed as clock cycles) are defined using equation 3.3, where div is the number of PRBS generators employed, M is the PRBS length and i refers the block for which the delay is being calculated. Since there isn't any reference point it's possible to assume that $delay_0$ is equal to 0 clock cycles.

$$delay_i = delay_0 + \frac{i \cdot (M + 1)}{div}, \quad i = 1, \dots, (div - 1) \quad (3.3)$$

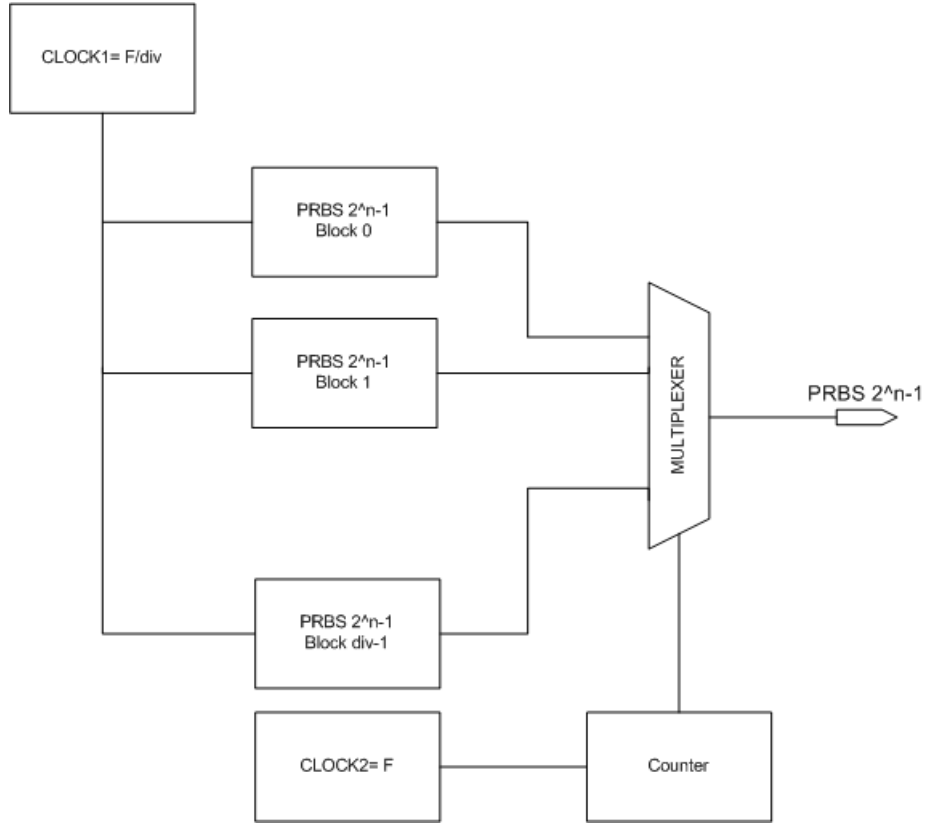


Figure 3.4: Multirate PRBS generator.

Since the pattern generators work on discrete time, the delay obtained by this equation must correspond to positive natural numbers, this means that the value of $(M+1)$ must be divisible by the number subrate PRBS blocks used.

For example to generate a PRBS $2^7 - 1$ at 4Gb/s using 5 pattern generator blocks that work at 800Mb/s each, the delays obtained with equation 3.3 are:

$$\begin{aligned}
 delay_0 &= 0 \\
 delay_1 &= \frac{128}{5} = 25.6 \\
 delay_2 &= \frac{2 \cdot 128}{5} = 51.2 \\
 delay_3 &= \frac{3 \cdot 128}{5} = 76.8 \\
 delay_4 &= \frac{4 \cdot 128}{5} = 102.4
 \end{aligned}$$

This is impossible to implement since there is no way to create a 25.6 clock cycles delay. On the other hand if an architecture of 4 pattern generators working at 1Gb/s each is chosen, the delays obtained for each block applying equation 3.3 are:

$$\begin{aligned}
 delay_0 &= 0 \\
 delay_1 &= \frac{128}{4} = 32 \\
 delay_2 &= \frac{2 \cdot 128}{4} = 64 \\
 delay_3 &= \frac{3 \cdot 128}{4} = 96
 \end{aligned}$$

3.3.5 Confidence level

The confidence level is defined as the probability, based on a set of measurements, that the probability of an event is better than some specified level[3]. For this application the confidence level(CL), given by equation 3.4, is interpreted as the probability that the BER(probability of errors $P(\epsilon)$) results is below an specified limit(γ).

$$CL = P[P(\epsilon) < \gamma] \quad (3.4)$$

However when designing a BER tester what really matters is the duration of the test in order to obtain a desired confidence level. Equation 3.5 can be used to calculate the number of bits(n) that need to be transmitted, to ensure a certain confidence level, when measuring a BER(p). N is the number of errors obtained during the test.

$$n = -\frac{\ln(1 - CL)}{p} + \frac{\sum_{k=0}^N \frac{(n \cdot p)^k}{k!}}{p} \quad (3.5)$$

From the data obtained with equation 3.5 the duration of the test can be determined by applying equation 3.6.

$$TT = \frac{n}{\textit{bitrate}} \quad (3.6)$$

Chapter 4

Methodological procedures

4.1 Problem definition

Basically to define the problem a reunion with the research team took place, in this reunion it was explained how the FSO communications system works, and defined the future applications for it, along with all the requirements for the test equipment. Also in that meeting it was defined that this thesis must be focused on developing a BER tester(BERT) to be used on the field test of the FSO system.

4.2 Compilation and analysis of information

Most of the information compiled by this project was obtained from internet, and because it was defined that the BERT is based on an FPGA. The search was focused on the application notes of FPGA manufacturers such as Xilinx, Altera, Lattice, etc.

The method to evaluate the accuracy of the information collected was to compare it with the standards defined by ITU, and to consult a researcher that has experience with this particular subject(PRBS, and BERT implementations).

4.3 Solution Synthesis

As it was explained before the research team defined the requirements of the BER test equipment that is going to be used to test the FSO communications system.

The requirements specified by the research team were:

- The size of the BERT must not exceed 40cm X 30 cm X 10cm.
- The BERT must be based on an FPGA.
- Use VHDL to design the BER tester.
- Serial speed transmission from 700 Mbps up to 2.5 Gbps.
- Non volatile program memory (the FPGA doesnt need to be programmed with a computer every time that its powered).
- The unit can be upgraded to add more functions at a later time.
- Has a data port compatible with P.C. standard RS-232 (serial port), to control the BERT and acquire data for analysis purposes.

This implied a research of the FPGA technologies in the field of high speed data serialization, it's characteristics, cost comparisson between manufacturers and finally availability of the hardware. This last aspect is very important because determines the viability of creating a solution within the available time.

4.4 Solution Implementation

The methodology for the solutions implementation consist basically on the following steps:

1. Research for a development board: This activity involves the search on internet of a development board based on an FPGA, which means analyzing several manufacturers proposal to determine if one of their products fulfills the project requirements, defined on section 4.1. In this part its very important to understand how the resources of the development board work in order to determine any possible complication when implementing the BERT.

2. Chosing an FPGA board: Once the information of the manufacturers is compiled the selection of the board could be made, its necessary to make a complete presentation of all the advantages and disadvantages of the selected board so the research team could make the final decision.
3. Development board acquisition: Once the selection of the FPGA is done it's necessary to make the order to one of the distributors.
4. Programming PRBS Algorithm: Creating a pattern generator that implements several PRBS algorithms using VHDL.
5. Simulation of PRBS Algorithm: Performing a behavioral simulation and time simulation of the pattern generator using Modelsim® Xilinx edition 6.0d.
6. Programming the simple BERT: This part involves the implementation of the BERT on the FPGA, the tester implemented will only perform a test with fixed parameters (no selectable PRBS or bit rate), and at a low bit rate (approximately 625 Mbps).
7. Simulation of simple BERT: Performing a behavioral simulation and time simulation of the simple BERT using Modelsim® Xilinx edition 6.0d.
8. Testing pattern generator on the FPGA: Once the development board arrives, the code of the pattern generator will be tested to see if the pattern generated by the real hardware works as expected from simulation. Also an important aspect of this test is to determine if the critical operation frequency of the pattern generator is above the maximum frequency needed by the BERT working at 3 Gbps.
9. Testing the simple BERT: In this test the idea is to check the proper function of the BERT, it will be made by applying an external loopback, the error will be inserted in the transmitter by inverting one or several bits. Once the test at 625 Mbps is successfully the operation frequency will be raised, changing the VHDL code, in order to determine the highest transmission speed or bit rate.
10. Creating the software for the P.C.: Programming an application on the P.C. using a high level programming language such as Microsoft® visual C++, visual C# or CodeGear® Delphi; in order to change the parameters of the test and to visualize the result of the test.

11. Implementing selectable bit rates on the BERT: Once the hardware of the BERT works properly its time to implement additional logic to change the operational frequency to be able of modifying the transmission bit rate without changing the VHDL code
12. Testing the function of the BERT: Basically consist on repeating the previous test but this time to change the transmission bit rate will only be necessary to change a dip switch (only for testing purposes, later itll be modifiable via software) on the board.
13. Implementing selectable parameters from the PC: This is the final phase of implementation the BERT. In this point is necessary to create all the software and hardware that allows modifying the test parameters from the personal computer.
14. Testing the final implementation of the BERT: Test the proper function of the BERT, first on the same loopback test and then in real life application and comparing the results with a normal BERT.

Chapter 5

Solution description

5.1 Solution analysis and final selection

As it was said in the previous chapter the solution consist on developing a BERT tester based on a FPGA. Consequently the first step consisted on selecting a development board that fit better all the project requirements.

The solutions were basically:

- BERT implementation using Xilinx ML-323 development board.
- BERT implementation using Avnet's Virtex-II Pro FF1152 Development Board.
- BERT implementation using Altera's Transceiver SI Development Kit, Stratix II GX Edition.

Altera's FPGA was ineligible, despite that fulfilled almost all requirements because the research team had previously work with FPGA's from Altera and experience some problems with the chip's reliability and technical support.

A comparison of the features from the two remaining boards is shown in table 5.1.

Table 5.1: Comparison between Xilinx and Avnet development boards

| Features | ML323 | Virtex II pro FF1152 | Virtex II pro FF1152 |
|-----------------------------------|---------------------------|--|--|
| Manufacturer | Xilinx | Avnet using Xilinx FPGA's | |
| FPGA Chip | XC2VP50 - 6FF1152 | XC2VP20 - 6FF1152 | XC2VP50 - 6FF1152 |
| Slices | 23616 | 9280 | 23616 |
| Logic Cells | 53136 | 20880 | 53136 |
| Block RAM (kb) | 4176 | 1584 | 4176 |
| 18X18 Multipliers | 232 | 88 | 232 |
| Rocket IO | 16 (64 SMA connectors) | 8, 6 to SMA connectors and 2 For Gigabit Ethernet | 8, 6 to SMA connectors and 2 For Gigabit Ethernet, 8 are unconnected |
| Ports | RS232 | RS232(2), Ethernet 10-100, Gigabit Ethernet, P160 expansion board. | |
| Display | Not available | 2X16 LCD | |
| Push Button | 4 | 4 | |
| DIP switches | 20 | 8 | |
| LED's | 20 | 4 | |
| Programming Methods | JTAG and Compact Flash | JTAG, onboard Flash, Compact Flash (using system ACE module) | |
| Permanent Data Storage | Not available | On Compact Flash (using system ACE Module) | |
| Clock Synthesizer | FPGA Embedded DCM's | FPGA Embedded DCM's, plus 3 external synthesizers programmable via software or Dip switches. | |
| SDRAM | Not available | 2 modules of 32 MB each, with 16bit data BUS. | |
| Price (U.S. Dollars) | \$4,495.00 | \$1,595.00 | \$3,195.00 |

The comparison between the two options showed significant advantages of the Avnet board. One of the most important aspects that led to the acquisition of the Avnet board was the programmable clock synthesizers because they allowed more flexibility regarding the possible transmission bit rates.

Another aspect is that both boards are sold by the same distributor and the delivery time of the Avnet's board was shorter.

There were two possible choices from AVNET, however, there isn't significant advantage of the XC2VP50 over the XC2VP20, because the development board for both FPGA chips is the same which means that all the extra pins of the XC2VP50 are unconnected, limiting the advantage only to its internal resources (slices, multipliers, block RAM, etc).

Finally the XC2VP20 board was selected because it fulfilled all the project requirements at a lower cost.

5.2 Hardware description

5.2.1 First level diagram

The BERT is composed of eight main blocks, which are:

- Serial communication: this block is in charge of the communication between the PC and the BERT. Its function is to program all the test parameters in the control logic and then transmit the results to the PC.
- Control logic: This block controls all the test parameters of the BERT (PRBS length, synchronization threshold, test time, transmission bit rate, etc). It's also capable of establishing bidirectional communication with modules that require a more complex control.
- Programmable frequency (TX/RX): these are two identical but independent blocks, one programs the clock synthesizer for the transmission and the other one varies the clock synthesizer of the receiver's reference clock.
- Pattern generator: Generates the transmission test sequence.
- Transmitter: Serializes the data from the pattern generator and transmit it through the channel.
- Receiver: Takes the data from the channel, deserializes it and recovers the clock.

- Error detector: Synchronizes the internal sequence generator with the received sequence, and compares it in order to determine the number of errors received, also counts the total number of bits received.

Figure 5.1 shows a schematic of these blocks.

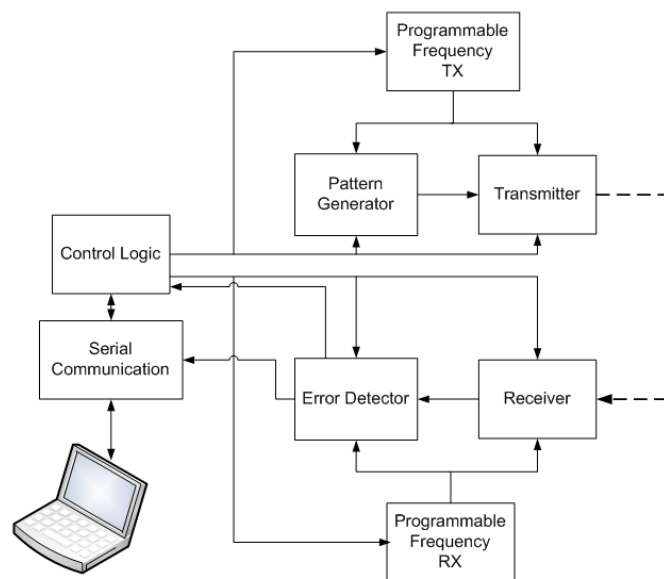


Figure 5.1: BER tester first level diagram.

5.2.2 Second level diagram

Serial communication:

The serial communications block is constituted basically by three blocks shown on figure 5.2, the *clock generator block*, *data transmission hardware* and the *serial communication hardware*.

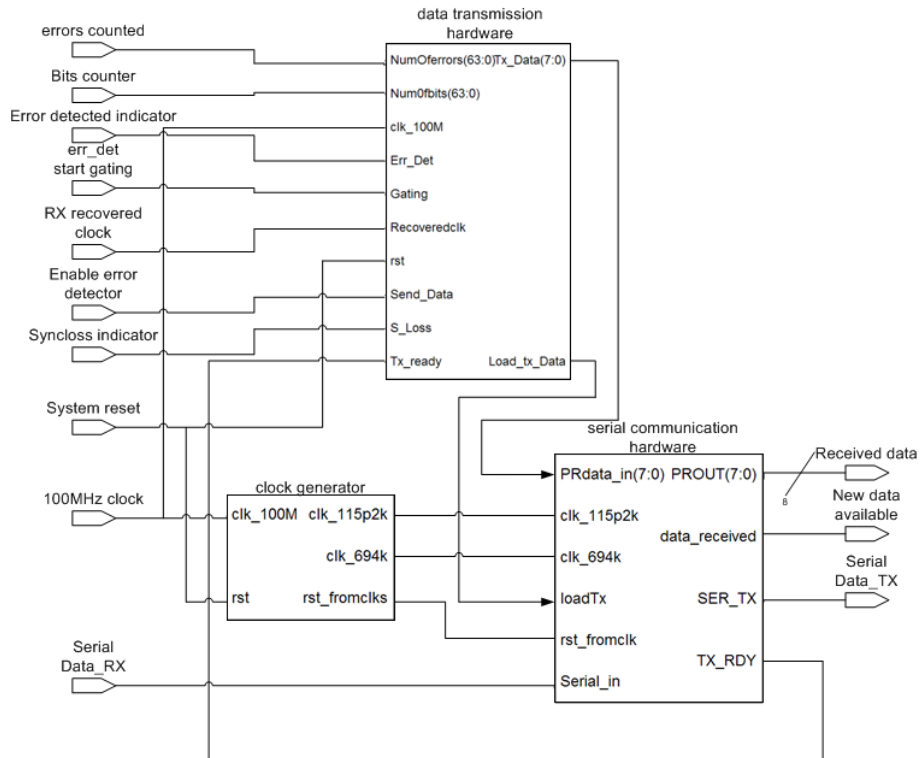


Figure 5.2: Serial communication second level diagram.

The clock generator takes the input clock at 100MHz and generates two clocks one for transmission and the other for reception. The reception clock is 6 times faster than the transmission clock in order to synchronize with the data stream. The ports of this module are:

Inputs

- `clk_100M`: 100MHz clock input.
- System reset (active low): asynchronous reset for the system.

Outputs

- `clk_115p2k`: transmission clock at 115.2 kHz.
- `clk_694k`: reception clock at 694 kHz
- `rst_fromclks` (active low): reset output, active until the output clocks are stable.

The *serial communication hardware* has the function of serializing the data, for transmission and deserializes the data received, from the PC, this block has the following ports:

Inputs

- Transmit Data: 8 bit data to be serialized and send to the PC.
- Start Transmission (active low): orders the block to load the value present on the Transmit Data port and start the transmission.
- Serial Data_RX: data reception, connected directly to the transmission line of the PC's serial port.

Outputs

- Received Data: data received from the PC once it's deserialized.
- New Data Available (active low): indicates that there is a new byte to be read on the Received Data port, is active for one period of the 694 kHz clock.
- Serial Data_TX: data transmission, connected directly to the reception line of the PC's serial port.
- Ready to transmit (active high): indicates that the serial port is not busy transmitting data and it can receive a start transmission request.

When the error detector is enabled the *data transmission hardware* manages the serial port through the Start transmission pin, the data pattern is shown in figure 5.3. Where errors refer to the error counter, bits is the number of received bits, and flags refer to gating status, syncloss indicator and errors indicator. After each data package is sent the system waits for 3ms before starting a new transmission.

| | | | | | | | | | | | | | | | | | |
|------------|----------------|----------------|----------------|----------------|----------------|----------------|---------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|-------------|------------|--------|
| Id - Flags | Errors (63:56) | Errors (55:48) | Errors (47:40) | Errors (39:32) | Errors (31:24) | Errors (23:16) | Errors (15:8) | Errors (7:0) | Bits (63:56) | Bits (55:48) | Bits (47:40) | Bits (39:32) | Bits (31:24) | Bits (23:16) | Bits (15:8) | Bits (7:0) | AA hex |
|------------|----------------|----------------|----------------|----------------|----------------|----------------|---------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|-------------|------------|--------|

Figure 5.3: Data package sent to the PC.

Control Logic block:

The control logic block is composed mainly from the *control regs* module which stores the values of all BERT's modifiable parameters, the *finish gating* module which defines when the gating must end, the *pattern generator control* and the *error detector control* a diagram of this block is shown on figure 5.4.

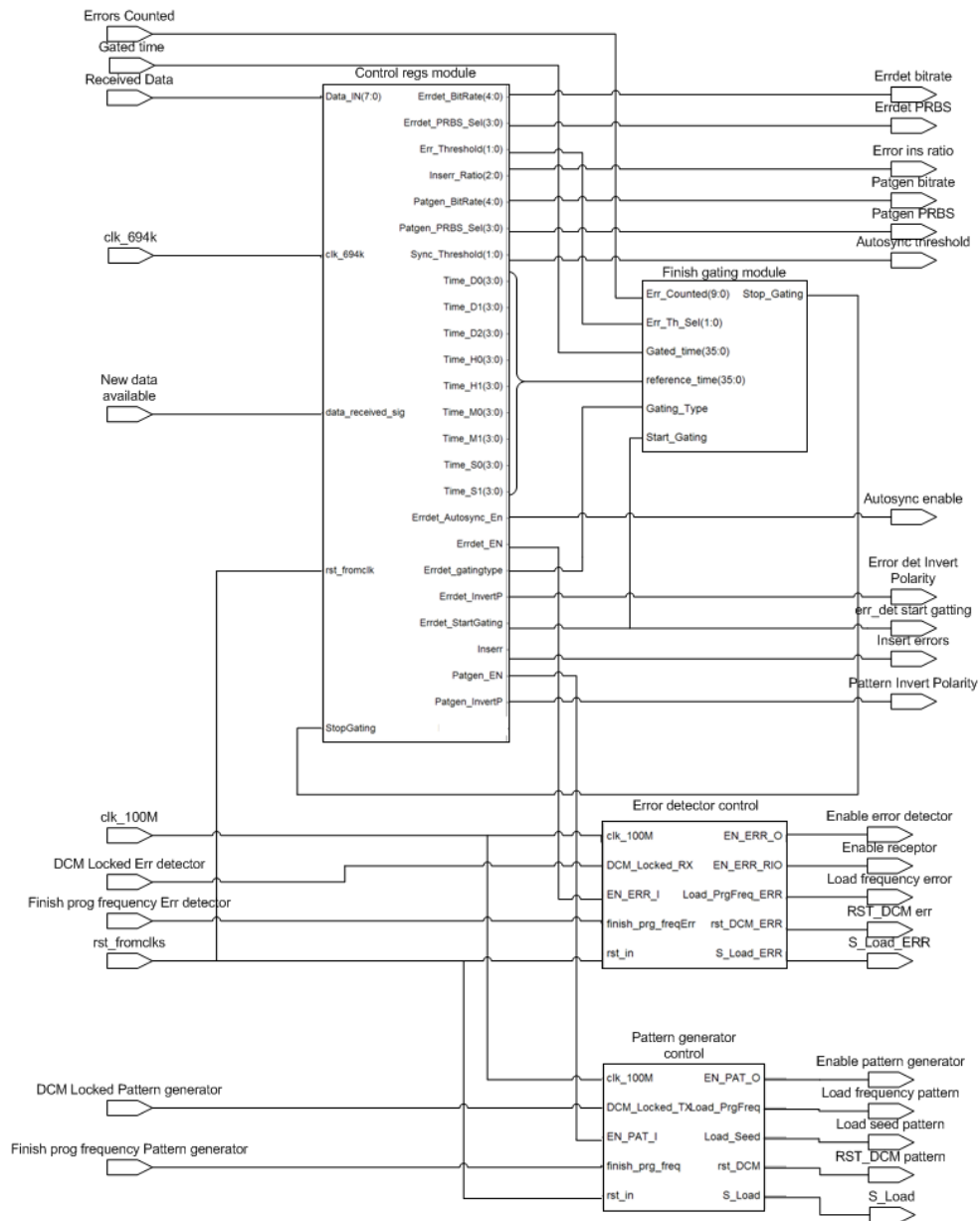


Figure 5.4: Control logic block diagram.

The *Control regs* module is basically a register bank that is loaded according with the data received from the serial port, the first data received is interpreted as the register address and the second byte is the register value to be written. The only value that can be modified by hardware is the start gating register that can be set to 1 by asserting the stop gating signal.

The *finish gating* module is a comparison module, it has two possible operation modes, compare number of errors or compare gating time. The stop values are given by the control regs module, while the number of errors and gating time are given by the error detector block. If the value received from the error detector is greater than or equal to the stop value it asserts the stop signal.

The *error detector control* and the *pattern generator control* are two state machines that manage the operation of the error detector and pattern generator blocks. Table 5.2 shows a description of the module's input and output pins.

Table 5.2: Control logic pinout description

| Signal Name | Direction | Function |
|---|-----------|--|
| clk_100M | input | 100 MHz clock. |
| clk_694k | input | 694kHz clock from serial port clocks generator module. |
| rst_fromclks | input | Reset from serial port clocks generator module. |
| DCM locked error detector | input | Indicates when the error detector reference frequency is stable. |
| Finish prog frequency error detector | input | Indicates when the clock synthesizer of the error detector has been programmed. |
| DCM locked pattern generator | input | Indicates when the pattern generator frequency is stable. |
| Finish prog frequency pattern generator | input | Indicates when the clock synthesizer of the pattern generator has been programmed. |
| New data available | input | Indicates that new data has been received from the serial port. |
| Received data | input | Data byte received from the serial port. |
| Gated time | input | Gating time counter. |
| Errors counted | input | Number of errors counted |
| Errdet bitrate | output | error detector bitrate selection. |
| Errdet PRBS | output | error detector selected PRBS. |
| Autosync threshold | output | Autosynchronization threshold value. |
| Autosync enable | output | Autosynchronization enable. |
| Errdet Invert polarity | output | Invert receiver polarity. |
| err_det start gating | output | Start gating process. |
| enable error detector | output | Enables the error detector |
| enable receptor | output | enable receiver |
| Load frequency error | output | Load frequency enable. |
| RST_DCM err | output | Reset error detector DCM. |
| S_Load_ERR | output | Serial load frequency enable. |
| Error ins ratio | output | Pattern generator error insert ratio selection. |
| Patgen bitrate | output | Pattern generator bitrate selection. |
| Patgen PRBS | output | Pattern generator selected PRBS. |
| Insert errors | output | Enable error insertion. |
| Pattern invert polarity | output | Inverts pattern generator polarity. |
| Enable pattern generator | output | Enable pattern generator. |
| Load frequency pattern | output | Load frequency enable, pattern generator. |
| Load seed pattern | output | Load pattern generator seed. |
| RST_DCM pattern | output | Reset pattern generator DCM. |
| S_Load | output | Serial load frequency enable. |

Programmable frequency (TX/RX):

This module is composed from two modules, one ROM that store the values needed to program the external clock synthesizer¹, and a shift register used to transfer these values to the synthesizer. Once that 14 shift cycles have occurred, the signal *finish* becomes active indicating that the programming cycle has ended. Figure 5.5 shows a schematic of these modules.

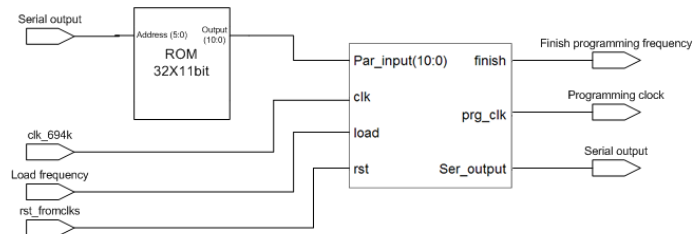


Figure 5.5: Programmable frequency block diagram.

The *programming clock* is the clock signal used to load the values on the clock synthesizer, it's only enabled when a programming cycle is taking place, otherwise is a fixed 0. This clock signal has the same frequency as the input clock (clk_694k), but has been delayed 180 degrees in order to allow the propagation of the serial output through the line.

Transmitter/Receiver:

These two blocks are implemented using Xilinx RocketIOTM, figure 5.6 ² shows the block diagram of the transceiver.

One RocketIO is used for the transmitter, and other is used for the receiver, as it can be seen on the transceiver block diagram the RocketIOTM have a lot of functions that are bypassed, in order to use it just as a high speed serializer. The output was programmed at 800mVpp on single ended configuration and 1600mVpp on differential.

¹See [2]

²Taken from: [8], *RocketIOTM Transceiver User Guide*

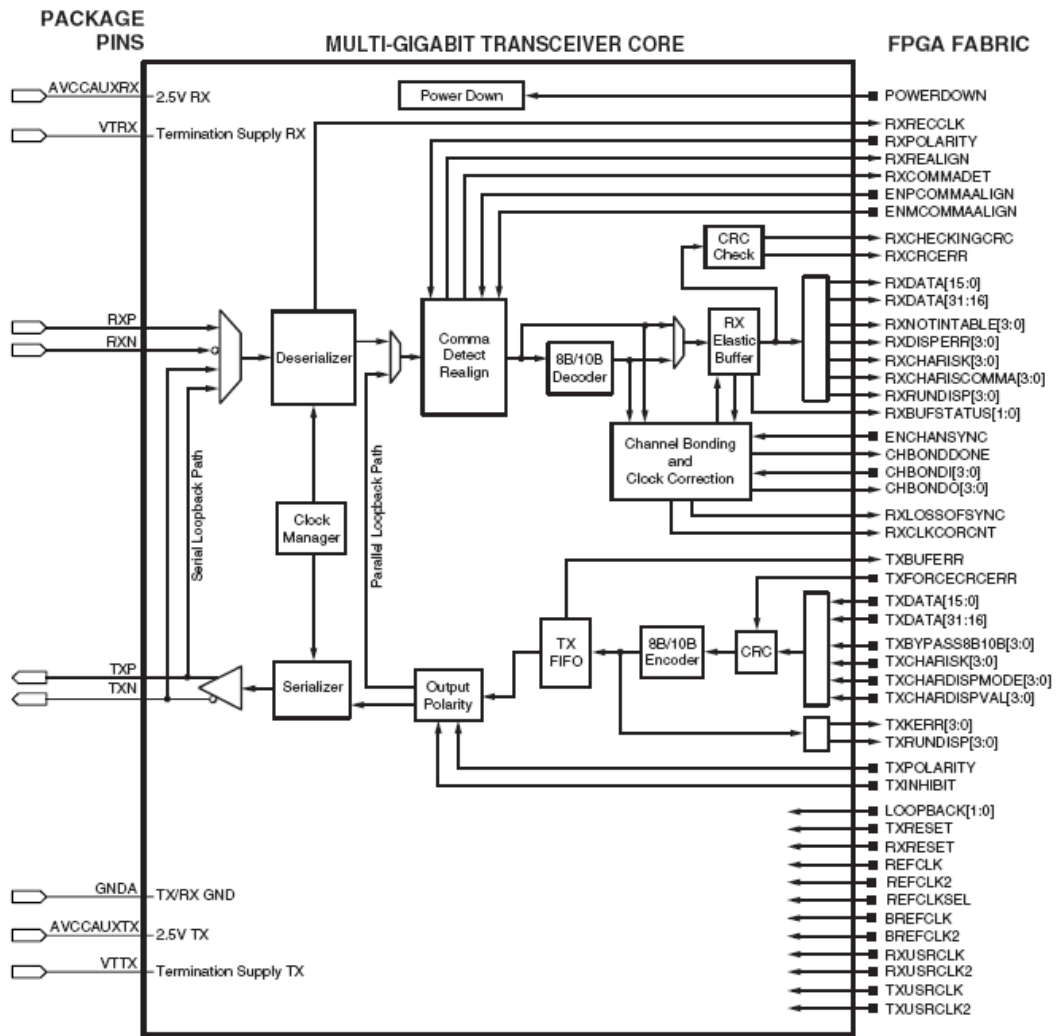


Figure 5.6: RocketIO™ transceiver block diagram.

In the transmission the CRC and the 8B/10B encoder are bypassed, also a two byte data width is selected, and the transmission clock is selected to full rate, this means that the transmission bitrate is twenty times the TXUSRCLK, and the data that is fed to the RocketIO™ has to be 20 bit width. Figure 5.7 shows the data pattern composed by the signals of the FPGA fabric pins, where 0 is the first bit transmitted.

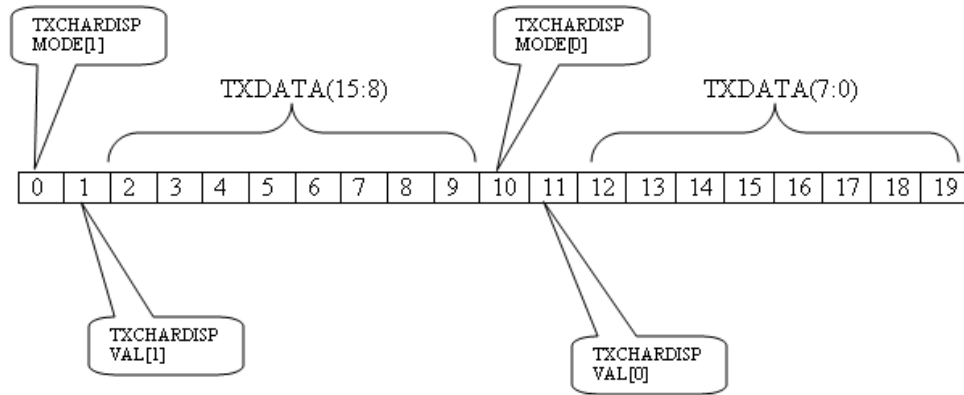


Figure 5.7: RocketIO™ transmission pattern 8B/10B bypassed.

On the receiver side had the same considerations, in this case, the comma detect and realign, 8B/10B decoder, CRC check and channel bonding and clock correction blocks where bypassed, the recovered clock is 1/20 of the data rate. Figure 5.8 shows the data pattern composed by the signals of the FPGA fabric pins, where 0 is the first bit received.

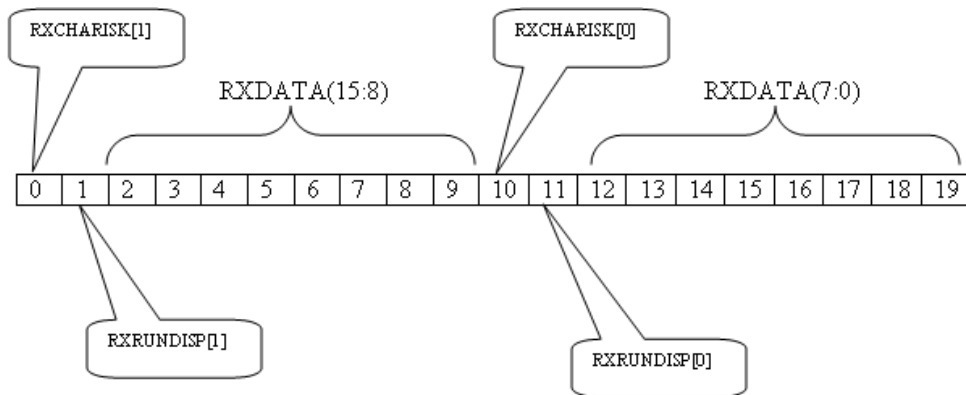


Figure 5.8: RocketIO™ transmission pattern 8B/10B bypassed.

Pattern Generator:

The *pattern generator*, shown in figure 5.9 is constituted by four blocks which are:

- *DCM_TX*: digital clock manager core from Xilinx³.
- *Pattern generator hardware*.
- *Error insertion block*: its basically a counter that enables the insert error (in-serr) signal every N clock cycles to accomplish a fixed error ratio. Where N can be set by the signal Error insertion ratio and can assume the following values: 500000000, 50000000, 5000000, 500000, 50000, 5000 and 500.
- *One bit inverter*: this block inverts the most significant bit of the pattern when the err input is asserted.

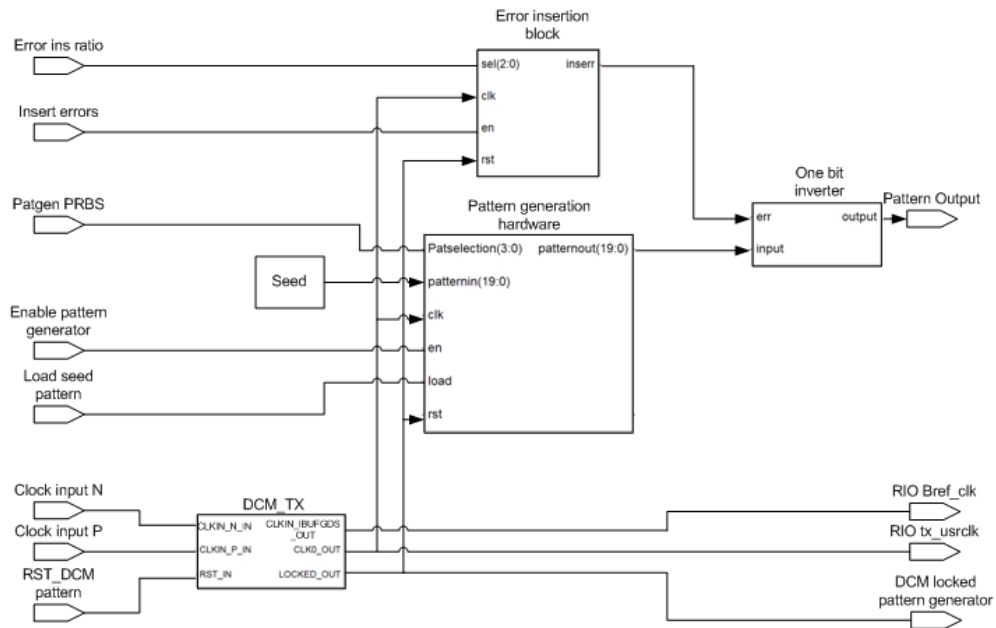


Figure 5.9: Pattern generator's second level block diagram.

³See [7]

Error detector:

The error detector shown in figure 5.10 is constituted by the following blocks:

- *DCM_RX*: digital clock manager core from Xilinx⁴.
- *Pattern generator hardware*.
- *Time counter*: used to measure the gating time.
- *Clock divider to 1Hz*: generates a 1Hz clock for the time counter and the Error and Syncloss indicator.
- *error and bit counter*: counts the total number of bits and errors received during a gating cycle.
- *error counter sync*: counts the number of errors for every 249994 bits received in order to calculate the BER, if the number of errors is greater than the Autosync threshold selected, the syncloss becomes active and syncloss indicator and the error indicator become active for 1 second. If the number of errors is less than the threshold selected, but there are errors in the pattern, the error indicator becomes active for one second. Otherwise no indicator is active. This counter is only in reset state if the error detector is disabled.

⁴See [7]

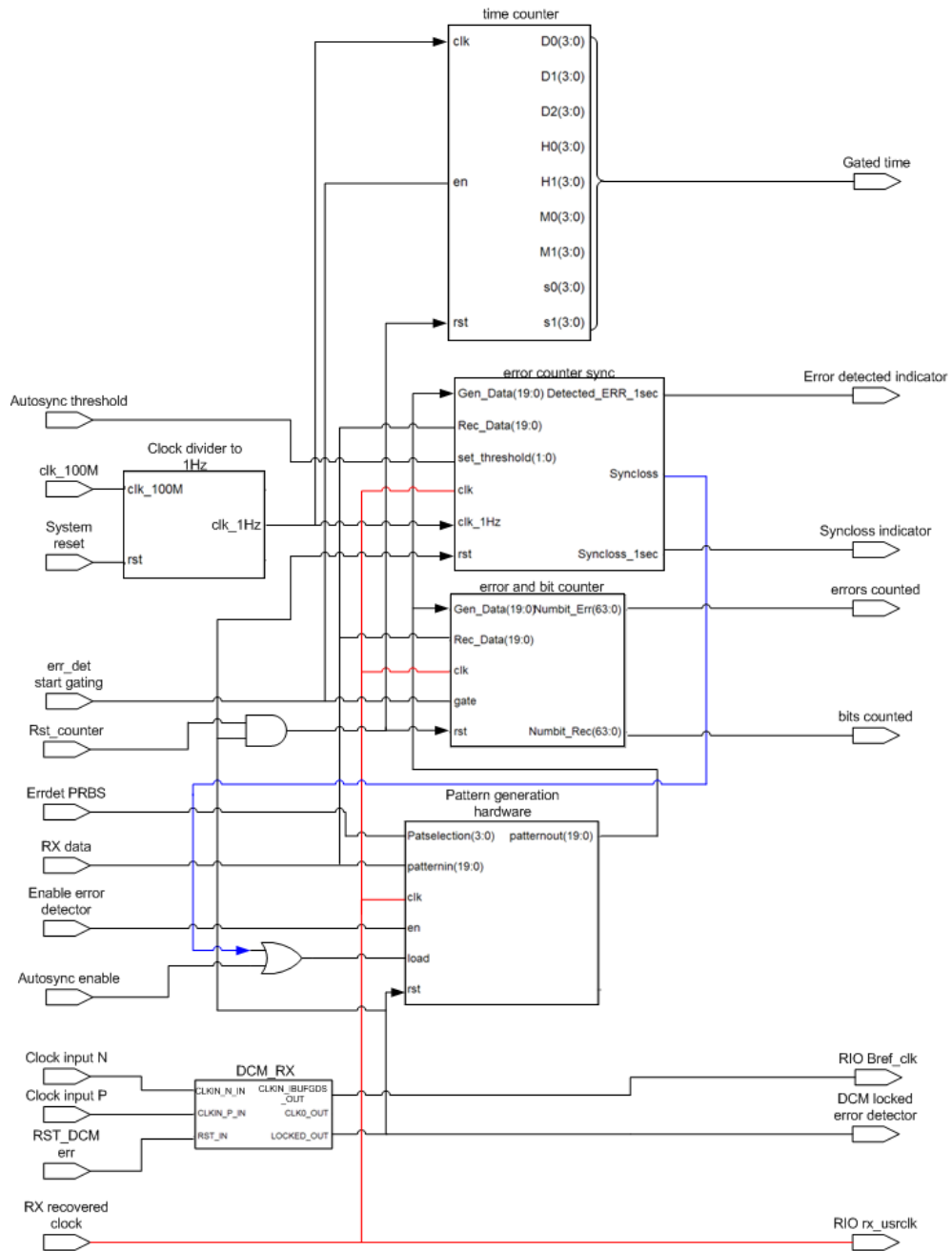


Figure 5.10: Error detector's second level block diagram.

5.2.3 Third level diagram

Clock generator:

The *clock generator* block, shown on figure 5.11, consists basically on two clock dividers, called *115.2 kHz clock generator* and *694 kHz clock generator*, each of this blocks generates it's respective clocks and reset signal, the *reset_fromclks* signal (active low) becomes inactive only when the reset signal of both blocks is inactive.

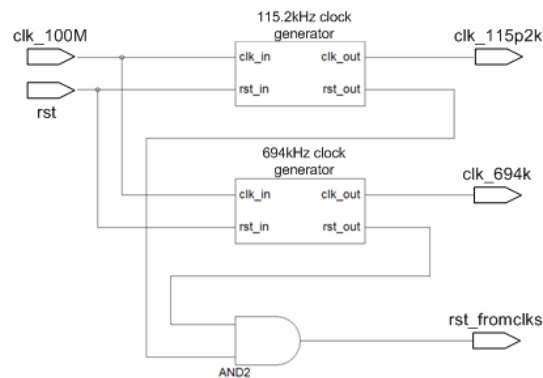


Figure 5.11: Clock generator's third level block diagram.

Serial communication hardware:

The *serial communication hardware*, shown in figure 5.12, is constituted by a *parallel in-serial out shift register* which works at 115.2 kHz, this shift register also generates a flag to indicate when 11 bits (start bit, 8 data bits and 2 stop bits) have been transmitted and it's ready to transmit a new byte.

The *RX control machine* is used to control the loading of both the *serial in parallel out shift register* and the *8 bit register*. This hardware works at a higher frequency in order to accomplish the synchronization with the *Serial.in* data stream. The load signal of the 8 bit register is delayed one clock cycle to generate the new data available signal.

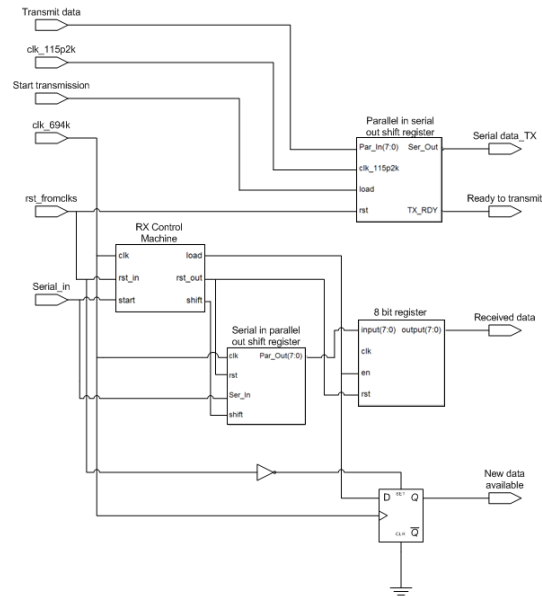


Figure 5.12: Serial communications hardware third level block diagram.

Error detector control:

The *error detector control* is a state machine that controls the activation of the error detector, it must follow the proper sequence in order to properly initialize the error detector otherwise it wont work. The state diagram of this machine is shown in figure 5.13.

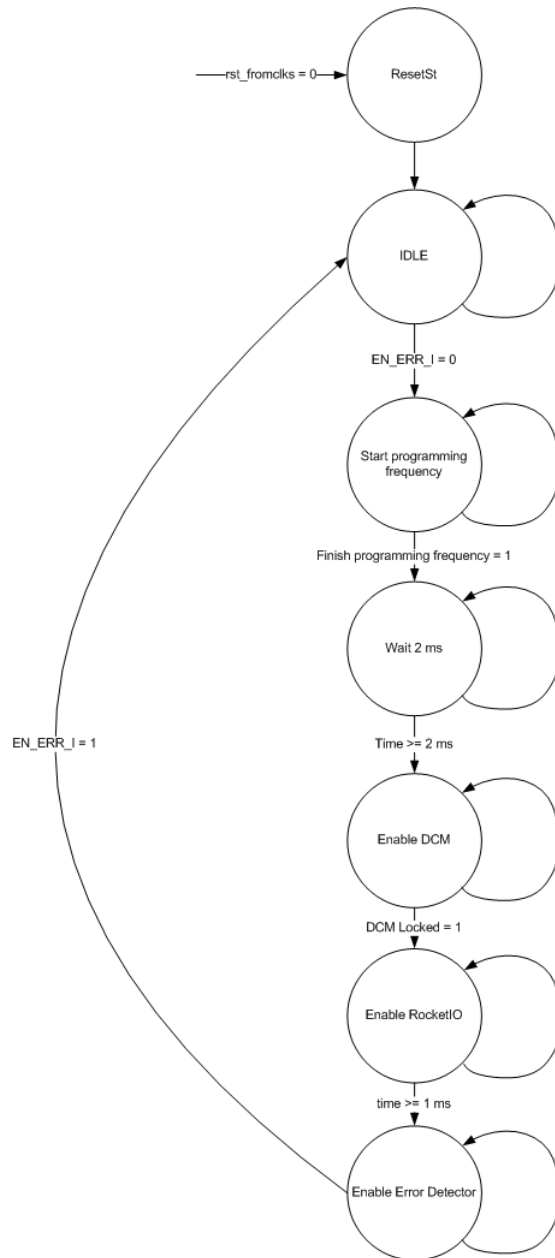


Figure 5.13: Error detector control machine's state diagram.

Pattern generator control:

As the error detector the pattern generator must follow a sequence in order to be properly initialized, this is again implemented by a state machine called *pattern generator control*, the state diagram is shown in figure 5.14.

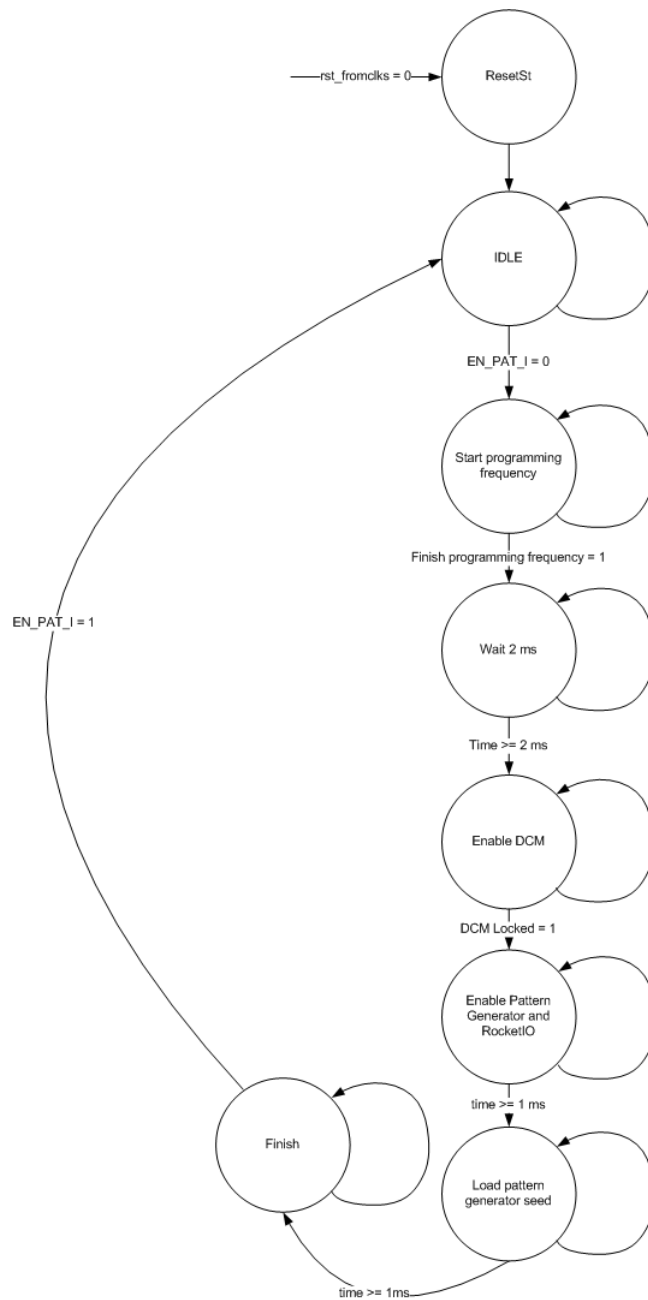


Figure 5.14: Pattern generator control machine's state diagram.

Pattern generation Hardware:

The *pattern generation hardware*, shown in figure 5.15, used on the error detector and pattern generator is the same; the only difference is that all the patterns that aren't PRBS where eliminated on the error detector reducing the multiplexer size. The twenty bit register at the pattern generation hardware output is only used to reduce the logic path, to accomplish the timing requirements.

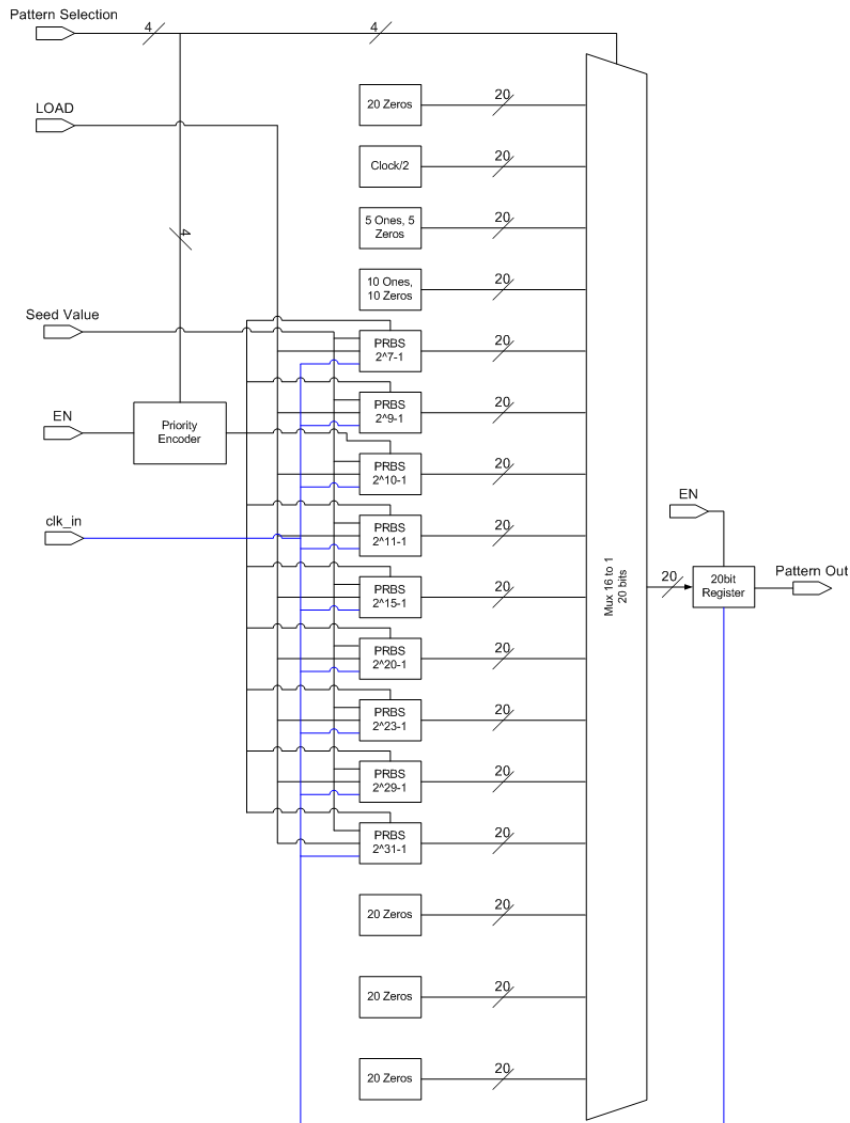


Figure 5.15: Pattern generator hardware third level block diagram.

5.2.4 Fourth level diagram:

115.2 kHz clock generator and 694 kHz clock generator:

These two blocks are constituted by one binary *Counter*, one *Comparator*, and two D-type Flip Flops, the only difference is the number of bits of the counter and the value from the constant (Value), the circuit is shown in figure 5.16.

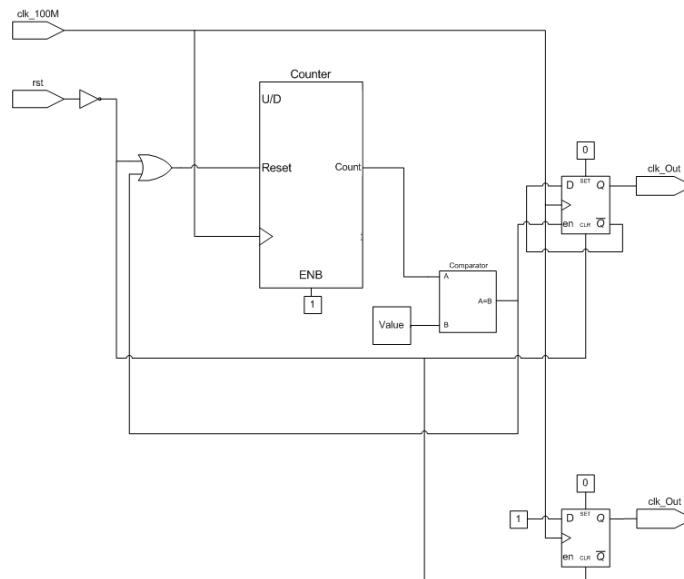


Figure 5.16: Clock generator's hardware.

RX Control Machine:

This control machine executes the synchronization with the data received from the serial port, the state diagram is shown in figure 5.17.

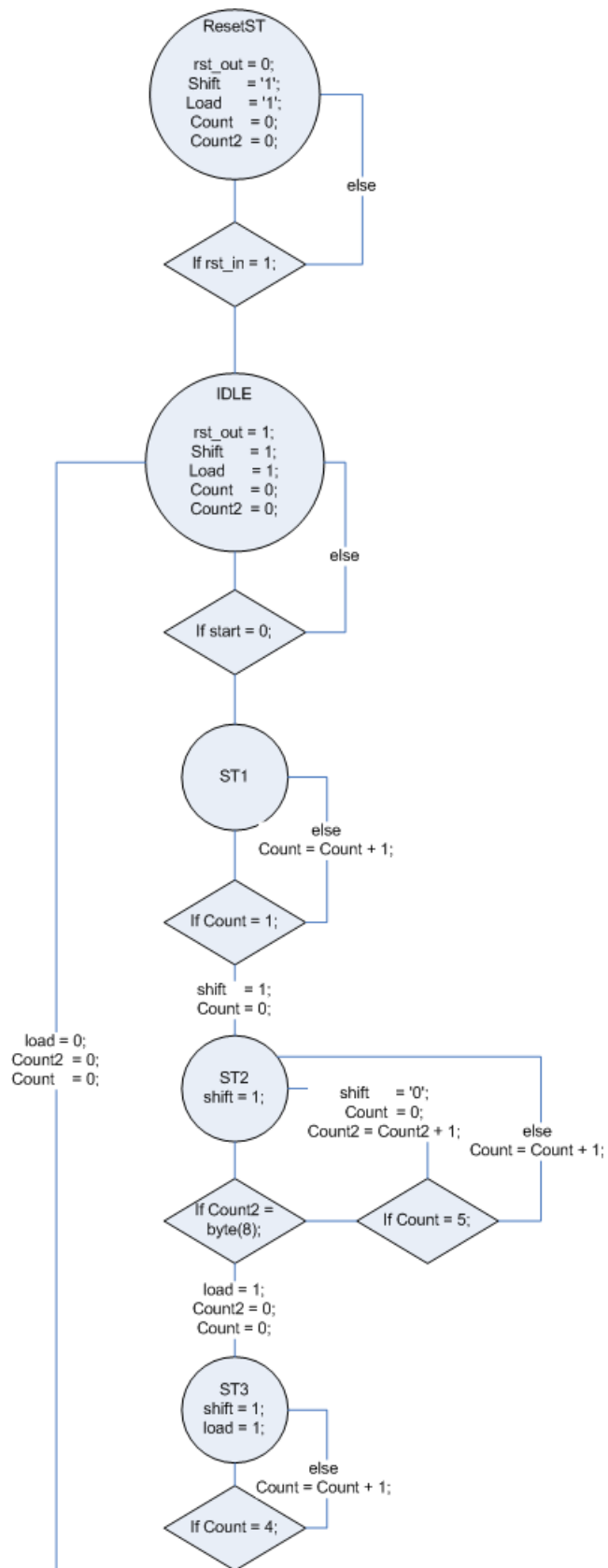


Figure 5.17: RX control machine's state diagram.

Pattern generation hardware:

The RocketIO input defined that the pattern has to be generated in 20 bits for every clock cycle, but no PRBS length can be divided by 20 this means that the technique explained in chapter 3 cannot be used.

Instead taking the original PRBS generator showed in figure 5.18, and replicating the feedback to generate all 20 bits in one clock cycle the circuit shown in figure 5.19 is obtained. Where if implemented with a 20 bit register the actual state would be the register output and the next state is all the values at the register inputs.

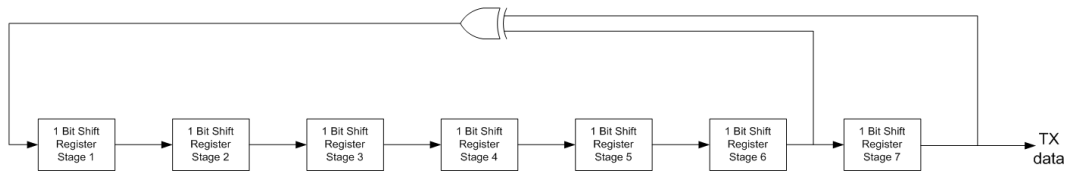


Figure 5.18: Normal implementation of the PRBS $2^7 - 1$.

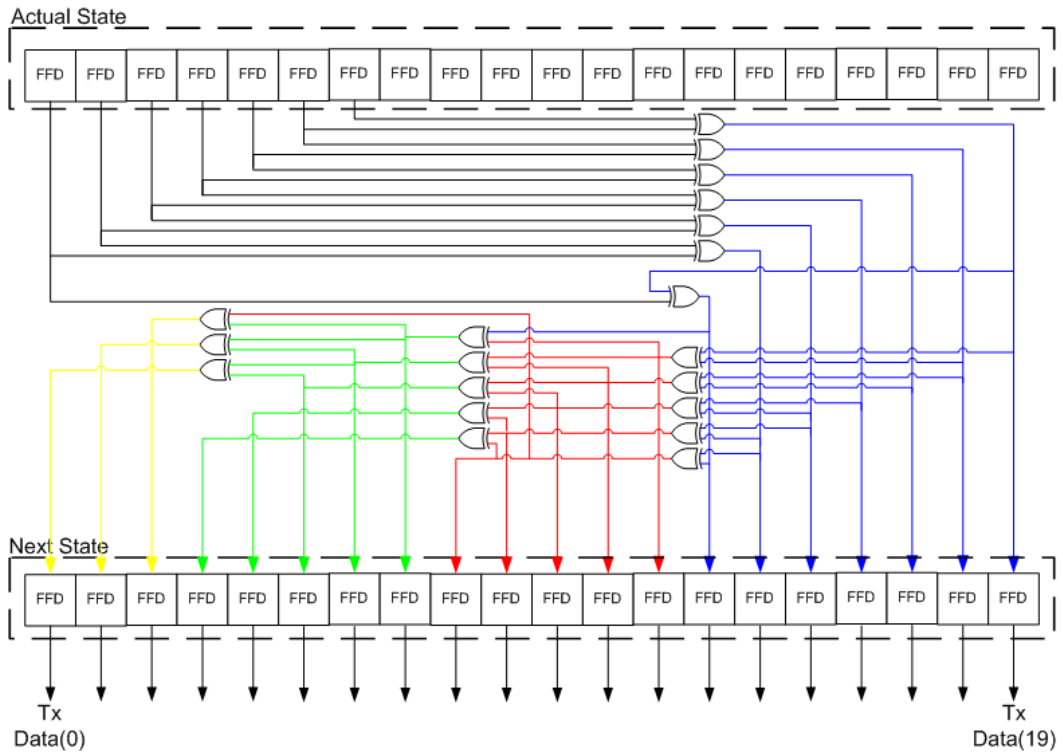


Figure 5.19: Final implementation of the PRBS $2^7 - 1$.

It's important to mention that this 20 bit register has another input that when asserted allows it to load directly the value present at the seed input without passing through the next state logic.

This implementation technique is the same used for all pattern generators, with the difference that for PRBS which feedback requires a separation of more than 20 bits (PRBS2a23 PRBS2a29 PRBS2a31) an extra register must be used to store the previous value.

Bit counter:

The count is incremented by 20 every clock cycle, the flip flops are used in order to compensate the delay of the pipeline registers used on the error counter. This structure is shown in figure 5.20. The maximum gating time defined by the register length is approximately 187 years, these calculations are shown in appendix A.3.

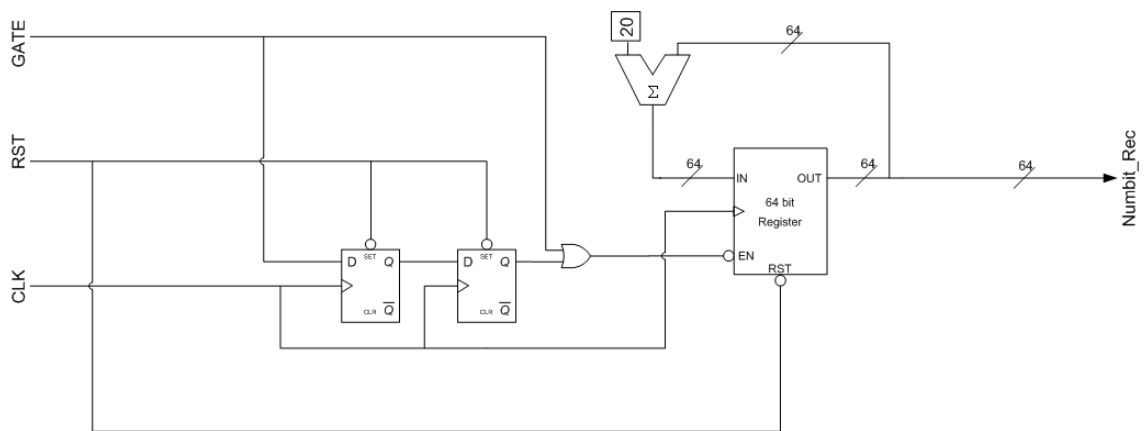


Figure 5.20: Bit counter hardware.

Error counter:

It's composed by a 20 bit comparator, which compare the received sequence and the local sequence bit to bit, after this comparison all the erroneous bits are added to obtain the total number of errors in the received package, finally this number of errors is added to the total gating of errors accumulated.

Figure 5.21 shows the diagram for the error counter, the reset value of the 64 bit register is 0, and must be reset prior the gating process starts, the 20bit register and 5 bit register are pipeline registers, used only to divide the combinatorial logic path, and achieve the timing specifications of 8ns ($125\text{MHz} = 2.5\text{Gb/s}$).

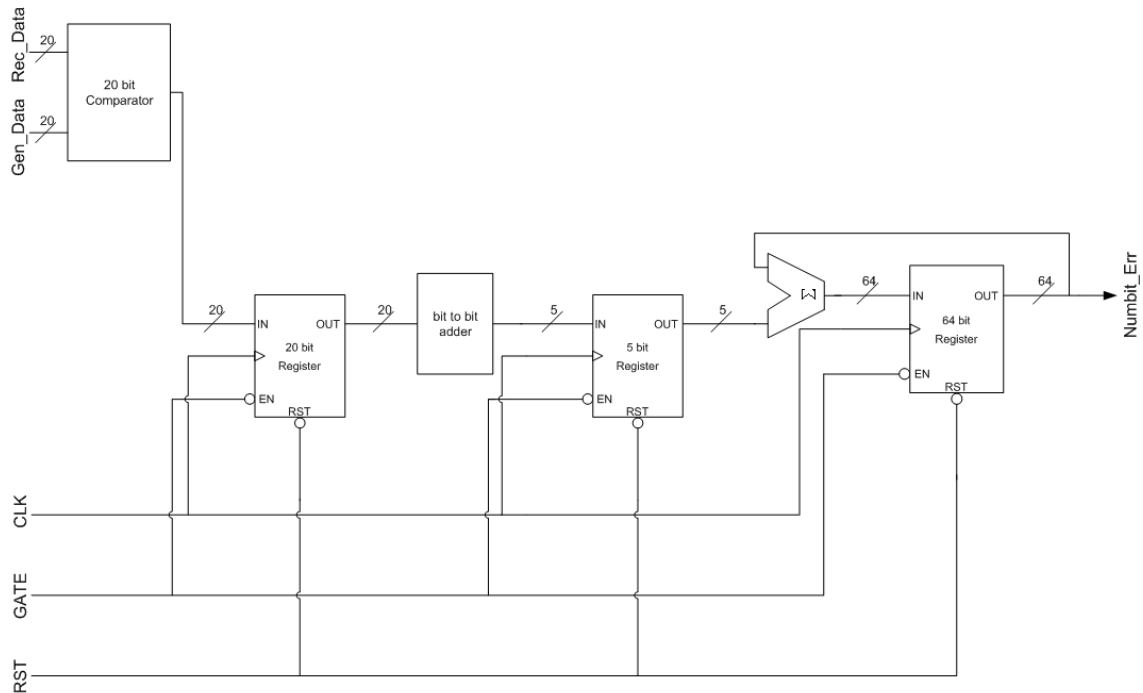


Figure 5.21: Error counter hardware.

The implementation of the bit to bit adder was made using 4 ROM of 32 positions each; the contents of these memories correspond to the total number of the ones present at the inputs. To obtain the total number of errors, the data present at the output of the ROM is added. This kind of structure was chosen because it allowed the addition of pipeline registers, if the delays of the logic path wouldnt allow the accomplishment of the timing constraints (156.25MHz). Figure 5.22 shows the hardware implementation of the bit to bit adder, it's important to notice that the logic path is balanced so any of the signals would have similar propagation delays.

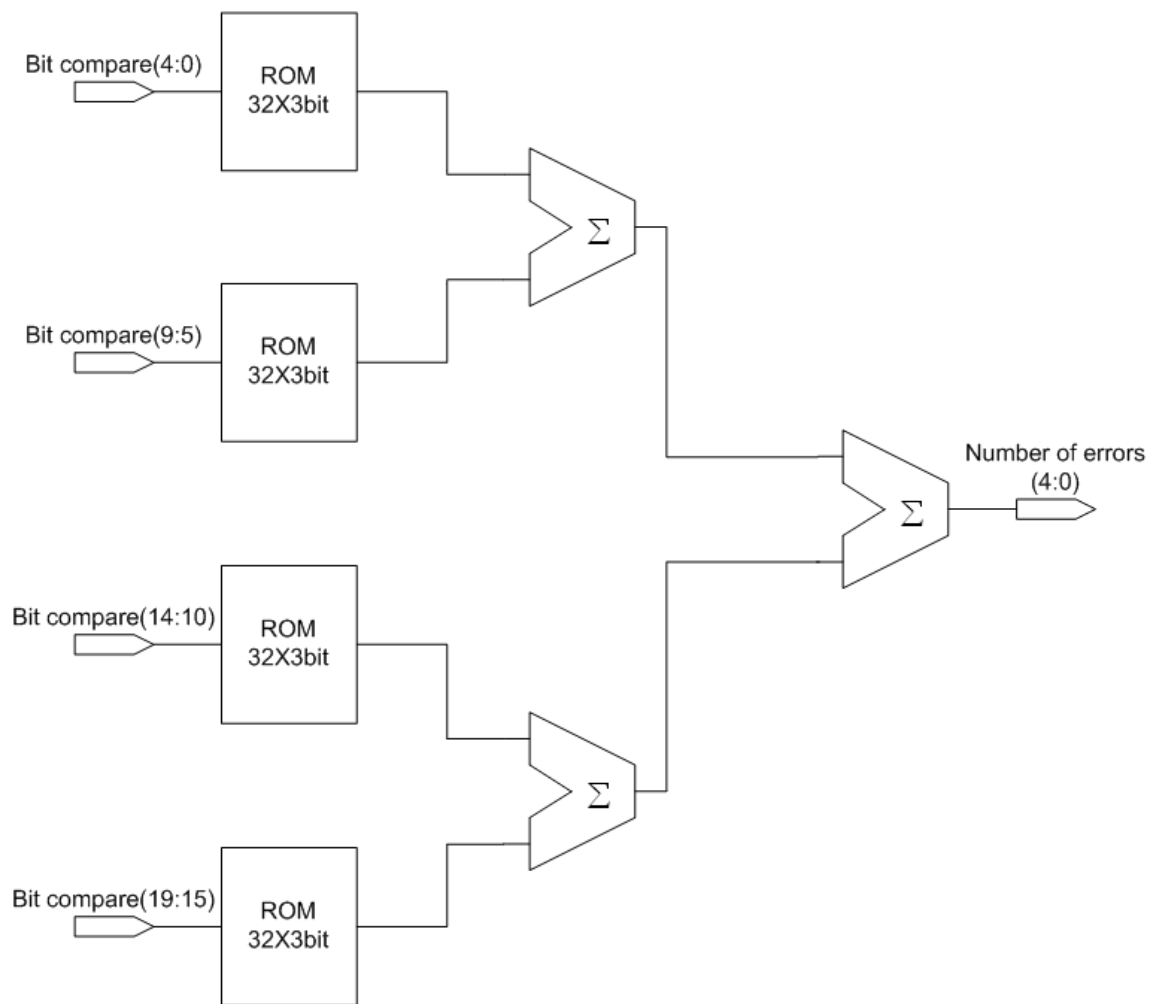


Figure 5.22: Bit to bit adder hardware.

5.3 Software description

A windows application was programmed in order to control the operation of the BERT. Basically this application program the values of the BERT control registers, to do this first the address of the register is transmitted followed by the desired value. Table 5.3 shows the address and the data width of the control register.

Table 5.3: Register address and data width

| BERT | Address (hexadecimal) | Parameter | bits | Description |
|--------------------------|-----------------------|-----------------|------|-------------------------------------|
| Pattern generator | 60 | PatgenEN | 1 | Pattern generator enable |
| | 61 | TXPRBS_sel | 4 | Pattern selection |
| | 62 | TX_BR | 5 | Bitrate selection |
| | 63 | TXinvertP | 1 | Invert transmission polarity |
| | 64 | Insert_err | 1 | Insert errors |
| | 65 | err_ratio | 3 | Error insertion ratio |
| Error detector | 66 | ERRdet_EN | 1 | Error detector enable |
| | 67 | RXPRBS_sel | 4 | Pattern selection |
| | 68 | RX_BR | 5 | Bitrate selection |
| | 69 | RXinvertP | 1 | Invert reception polarity |
| | 6A | Autosync_EN | 1 | Enable autosynchronization |
| | 6B | Start_gating | 1 | Error detector start gating |
| | 6C | Gating_Type | 1 | Gating type selection |
| | 6D | TimeD2 | 4 | Gating time days(0 to 9) |
| | 6E | TimeD1 | 4 | Gating time days(0 to 9) |
| | 6F | TimeD0 | 4 | Gating time days(0 to 9) |
| | 70 | TimeH1 | 4 | Gating time hours (0 to 2) |
| | 71 | TimeH0 | 4 | Gating time hours (0 to 9) |
| | 72 | TimeM1 | 4 | Gating time minutes (0 to 5) |
| | 73 | TimeM0 | 4 | Gating time minutes (0 to 9) |
| | 74 | TimeS1 | 4 | Gating time seconds (0 to 5) |
| | 75 | TimeS0 | 4 | Gating time seconds (0 to 9) |
| | 76 | Error_Threshold | 2 | Gating error threshold selection |
| | 77 | Sync_threshold | 2 | Synchronization threshold selection |

The other operation that this application performs is the the serial port reading and information processing, in order to visualize the results, and finally write the .log file. The thread programmed to handle the serial port reading is included in appendix A.5.

The graphic interface of the program is showed on figure 5.23, there is one panel for the pattern generator settings and other for the error detector, when error threshold gating is selected, the error detector panel changes as shown on figure 5.24.

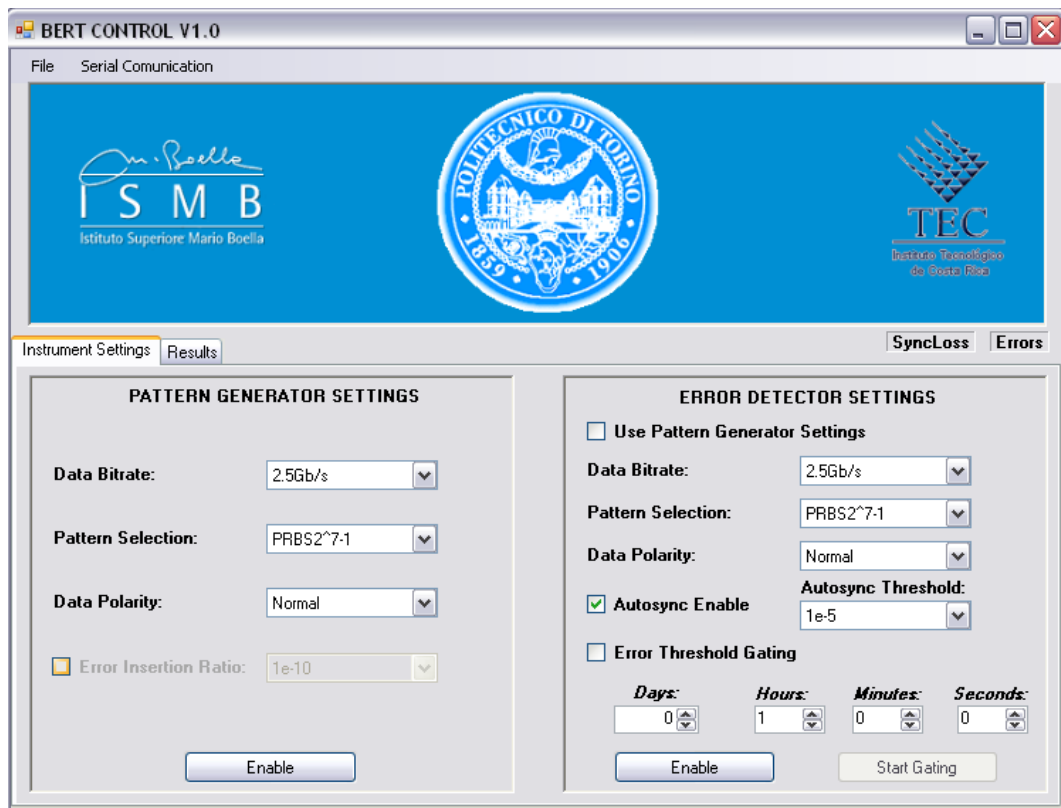


Figure 5.23: Graphic interface main window.

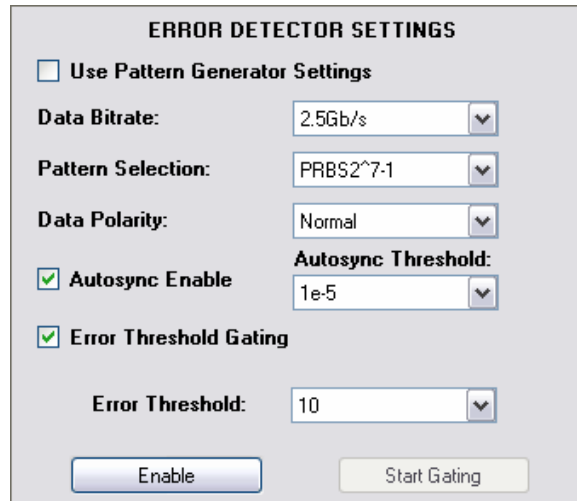


Figure 5.24: Changes to the error panel when error threshold gating is selected.

Figure 5.25 shows the results visualization window, this window display the total number of bits received, total number of errors, gating time, selected PRBS and BER.

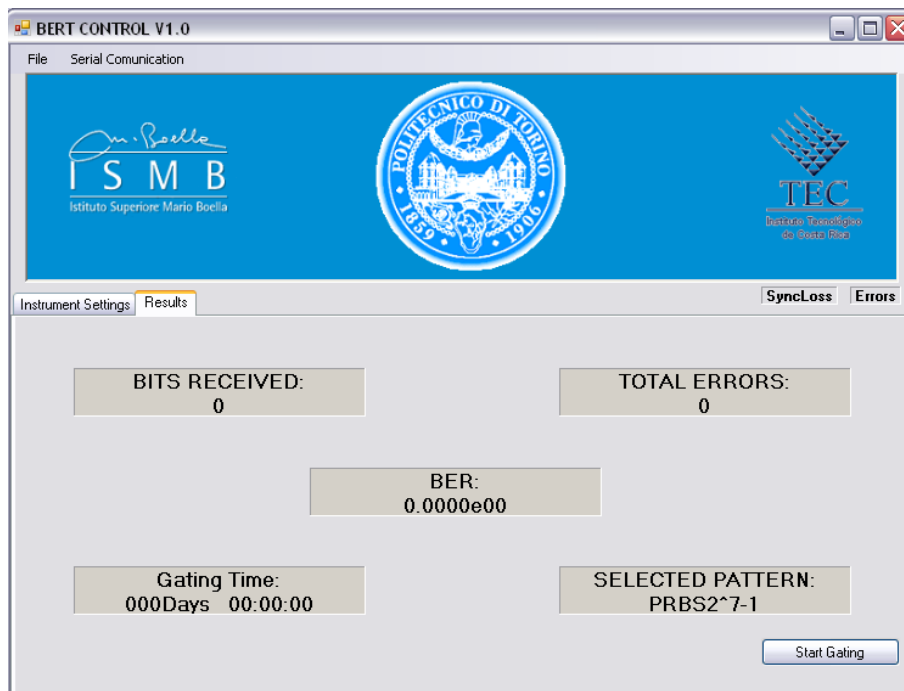


Figure 5.25: Results visualization window.

Chapter 6

Results analysis

6.1 Results

6.1.1 First test results

The first test was performed at several bitrates to study the performance of the RocketIO™. The results from this test are summarized on table 6.1.

Table 6.1: First test results

| Bitrate (Mb/s) | Eye Amplitud(mV) | Effective Bitrate(Mb/s) | Jitter(ps) |
|----------------|------------------|-------------------------|------------|
| 625 | 1183 | 615 | 18.9 |
| 1000 | 1156 | 987 | 15.6 |
| 2500 | 952 | 2530 | 8.14 |
| 3125 | 868 | 3110 | 9.4 |

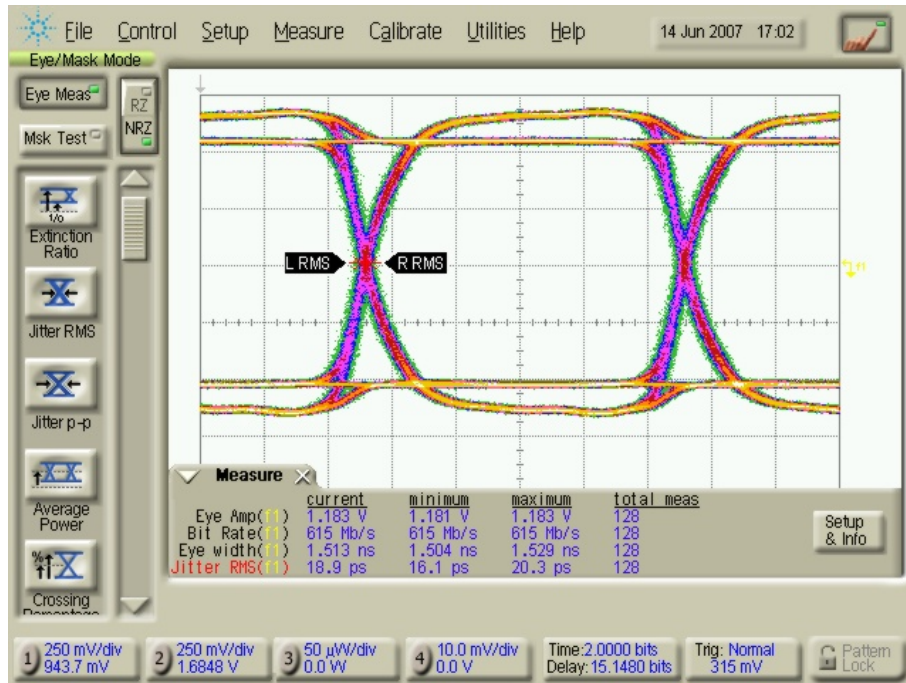


Figure 6.1: Eye diagram for the RocketIO working at 625Mb/s.

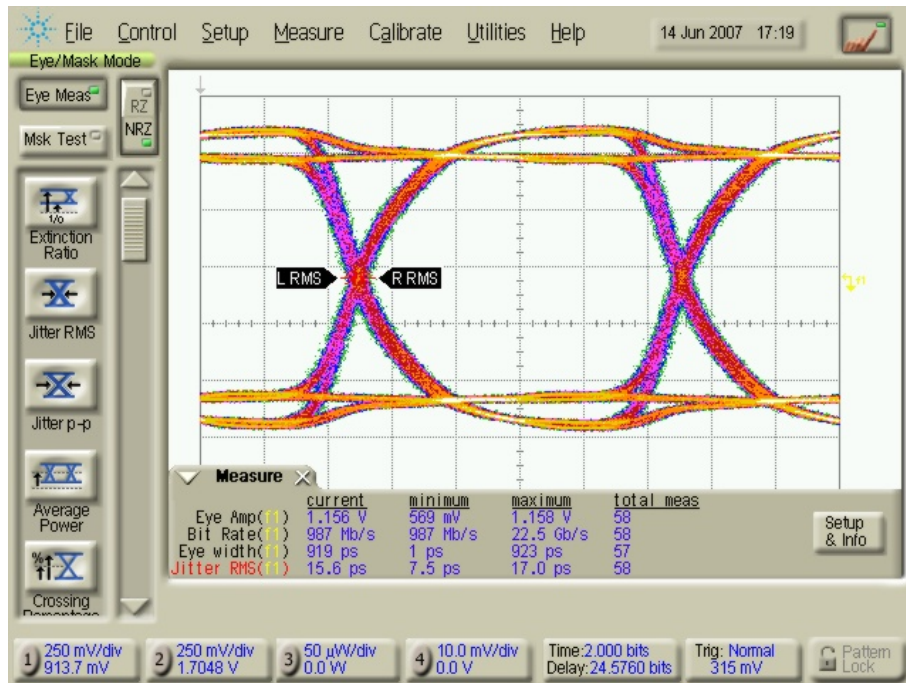


Figure 6.2: Eye diagram for the RocketIO working at 1Gb/s.

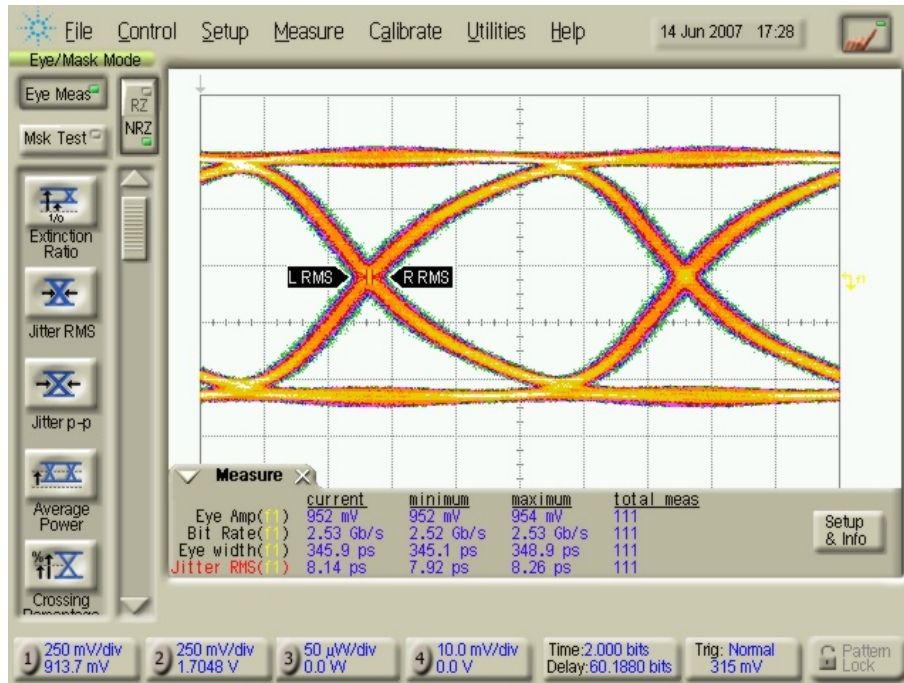


Figure 6.3: Eye diagram for the RocketIO working at 2.5Gb/s.

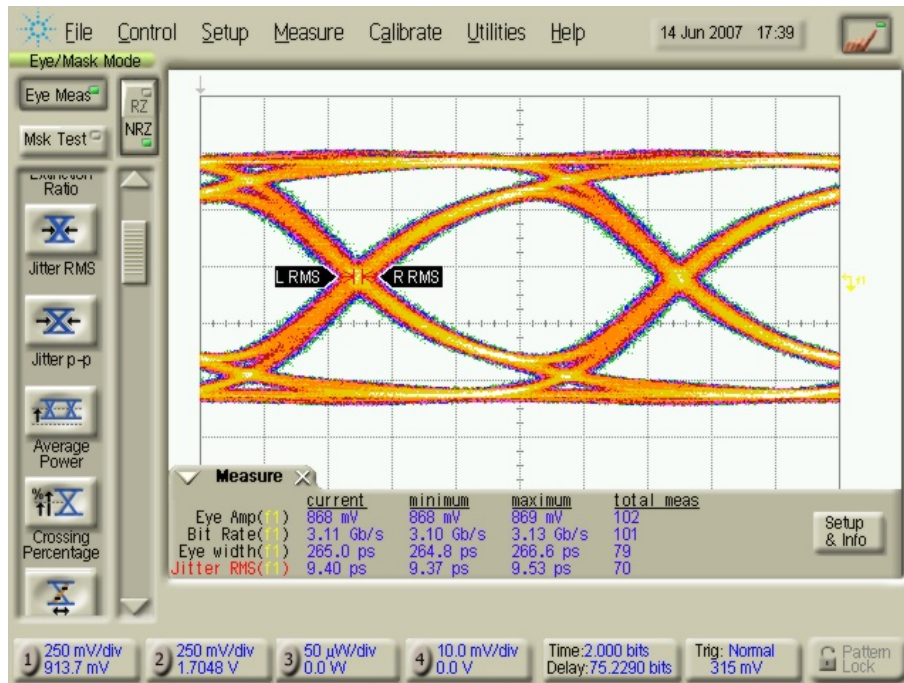


Figure 6.4: Eye diagram for the RocketIO working at 3.125Gb/s.

6.1.2 Second test results

When the clock and data recovery evaluation board from analog devices was used, an improvement on the eye diagram is accomplished. The results obtained are shown in table 6.2.

Table 6.2: Second test results

| Bitrate (Mb/s) | Eye Amplitud(mV) | Effective Bitrate(Mb/s) | Jitter(ps) |
|----------------|------------------|-------------------------|------------|
| 625 | 787 | 622 | 4.5 |
| 1000 | 762 | 1060 | 5.5 |
| 2500 | 764 | 2500 | 7.15 |
| 3125 | N/A | N/A | N/A |

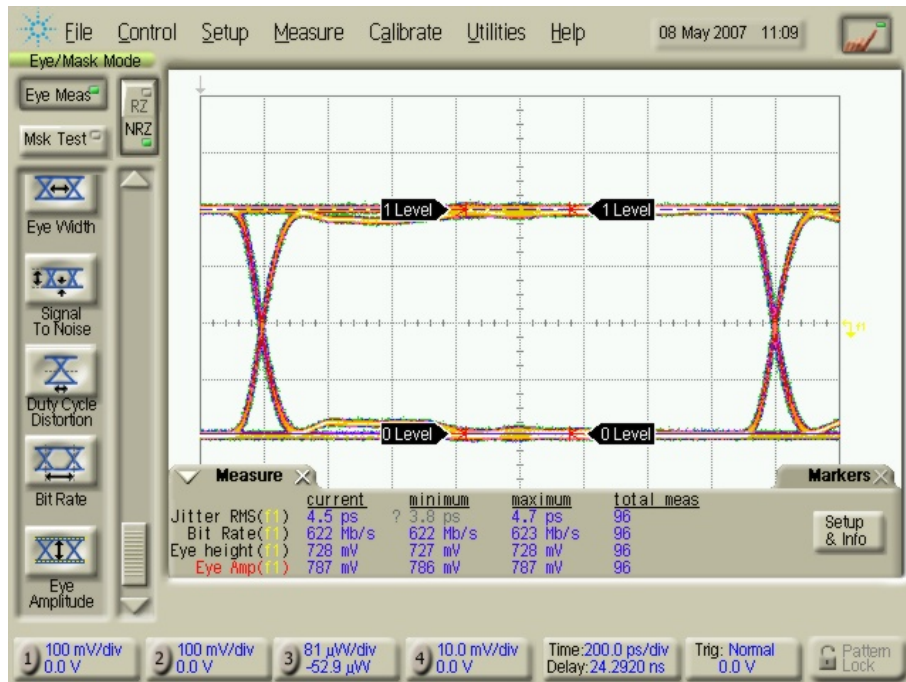


Figure 6.5: Eye diagram for the improved signal working at 625Mb/s.

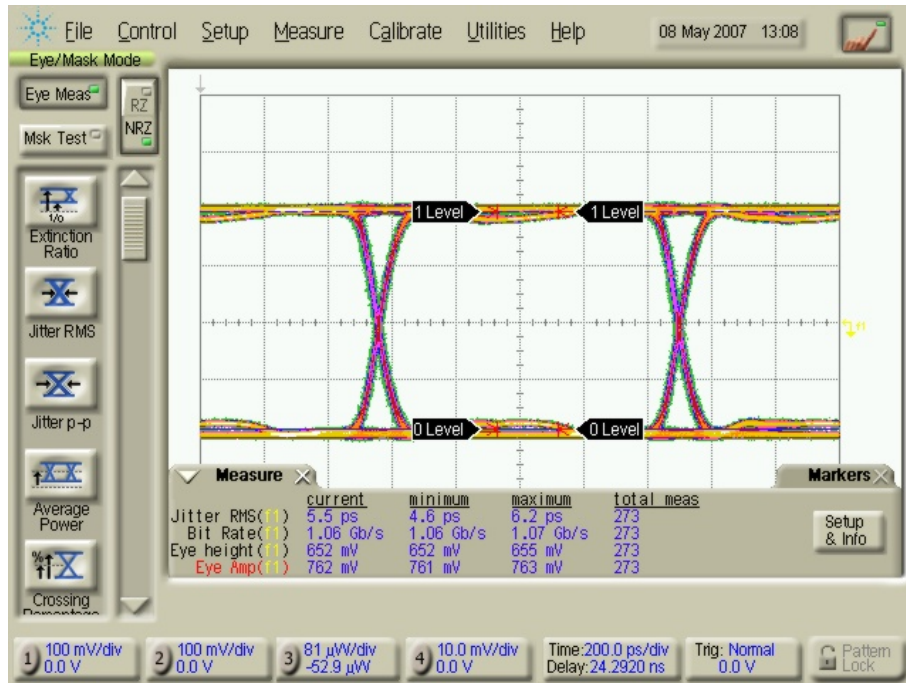


Figure 6.6: Eye diagram for the improved signal working at 1Gb/s.

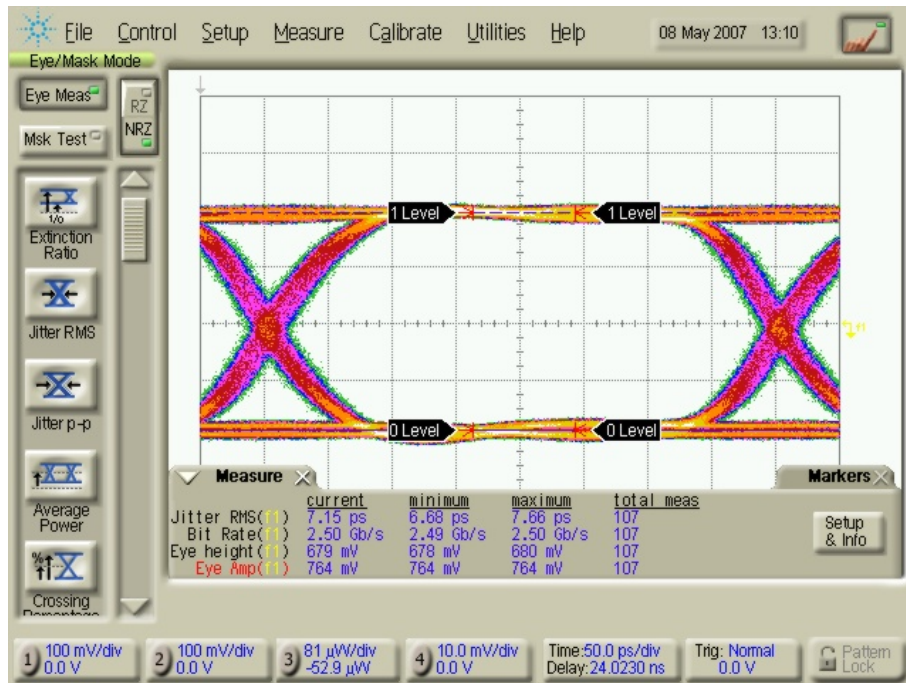


Figure 6.7: Eye diagram for the improved signal working at 2.5Gb/s.

6.1.3 Third test results(FPGA Loopback):

The first FPGA loopback test was performed at several bitrates and test times, however just a few representative test are shown.

Test A: PRBS $2^7 - 1$ at 2.5Gb/s for 5 hours and error insertion disabled, figure 6.8 shows the results obtained.

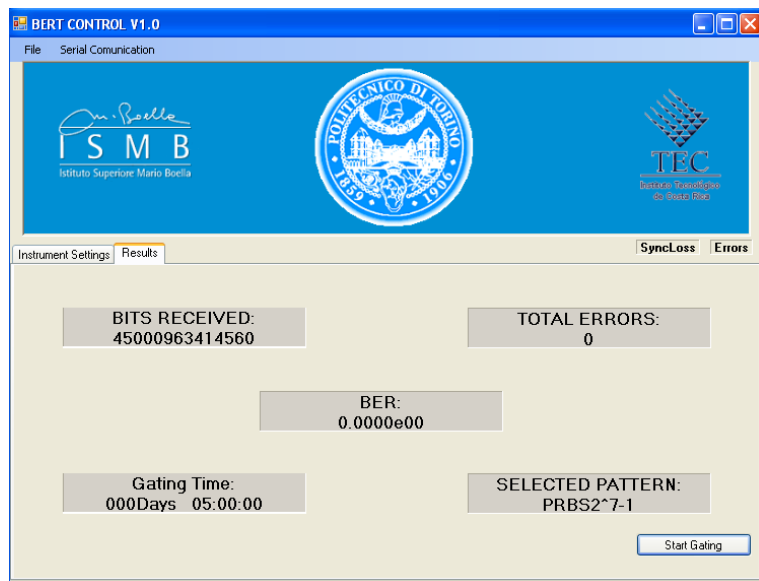


Figure 6.8: PRBS $2^7 - 1$ loopback test results.

Test B: PRBS2¹⁵ – 1 at 2.5Gb/s for 35 minutes error insertion ratio 1e-9. The results obtained are shown in figure 6.9.

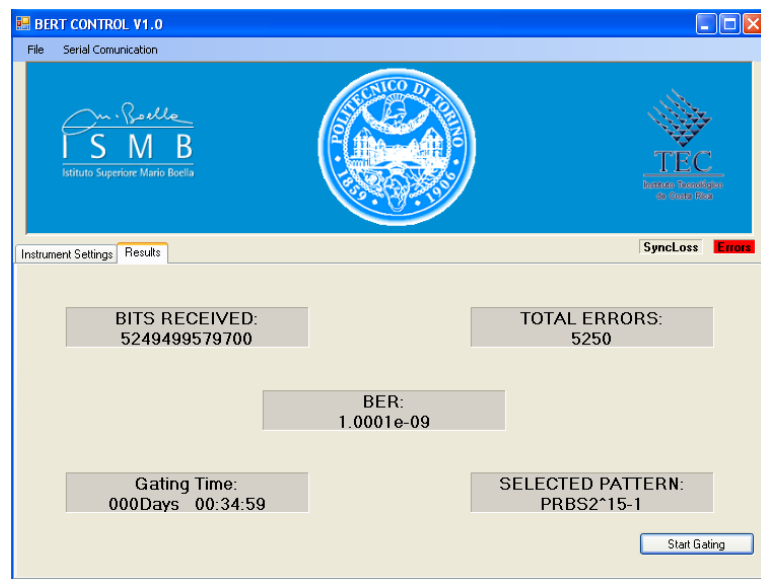


Figure 6.9: PRBS2¹⁵ – 1 loopback test results with 1e-9 error insertion ratio.

Test C: PRBS2²⁹ – 1 at 2.5Gb/s for 1 hour and error insertion ratio disabled. Figure 6.10 shows the results obtained, and figure 6.11 shows the graphic of the BER evolution vs time(created from the .log file using Matlab). As it can be seen this test put on evidence that the RocketIO™ channel wasn't error free, which lead to the following tests in order to determine the error source.

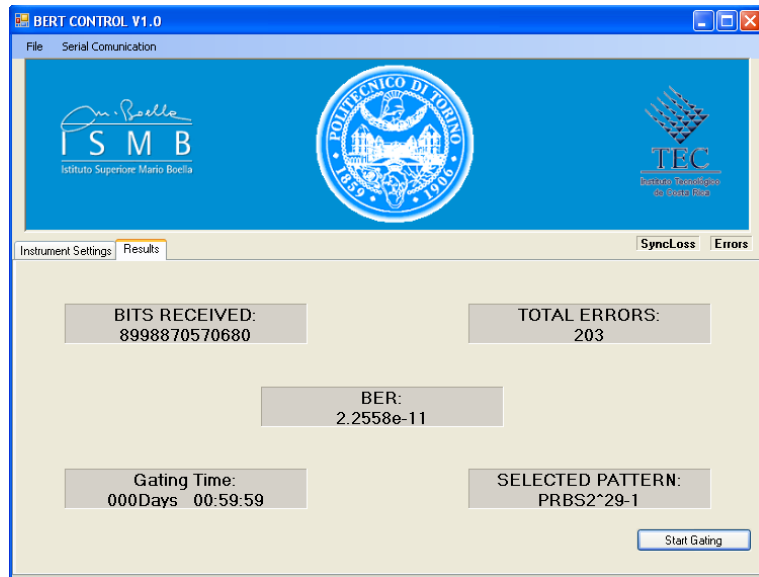


Figure 6.10: PRBS2²⁹ – 1 loopback test results.

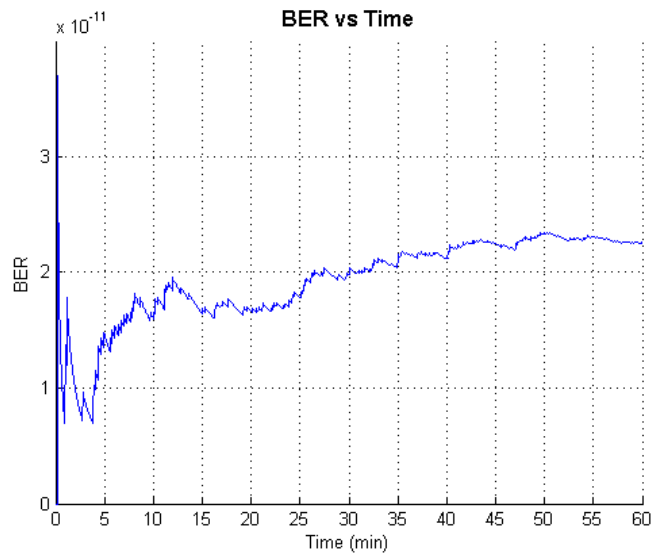


Figure 6.11: PRBS2²⁹ – 1 BER vs Time.

Test D: PRBS $2^{29} - 1$ at 2.5Gb/s for 1 hour and error insertion ratio disabled: This test was performed using the same reference clock for the transmitter and the receiver in order to study the behavior when the CDR of the RocketIO wasn't forced. Figure 6.12 shows the results obtained, and figure 6.13 shows the graphic of the BER evolution vs time(created from the .log file using Matlab).

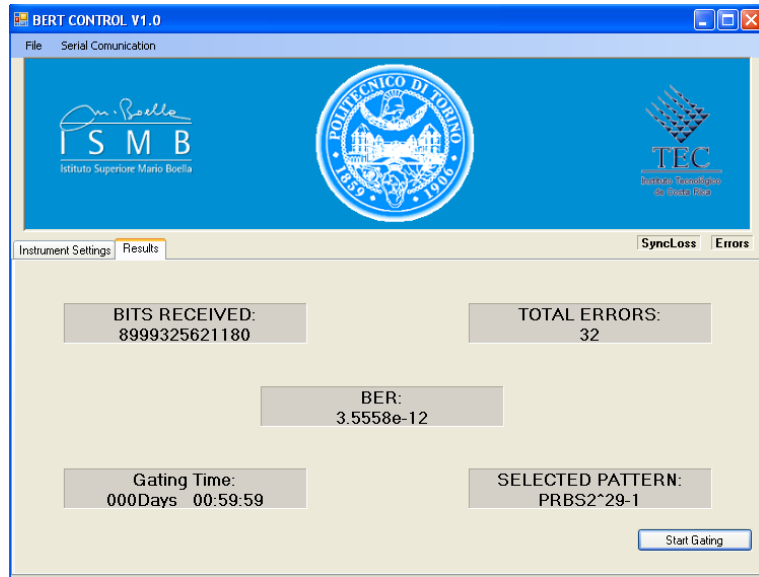


Figure 6.12: PRBS $2^{29} - 1$ loopback test results, sharing clock generator.

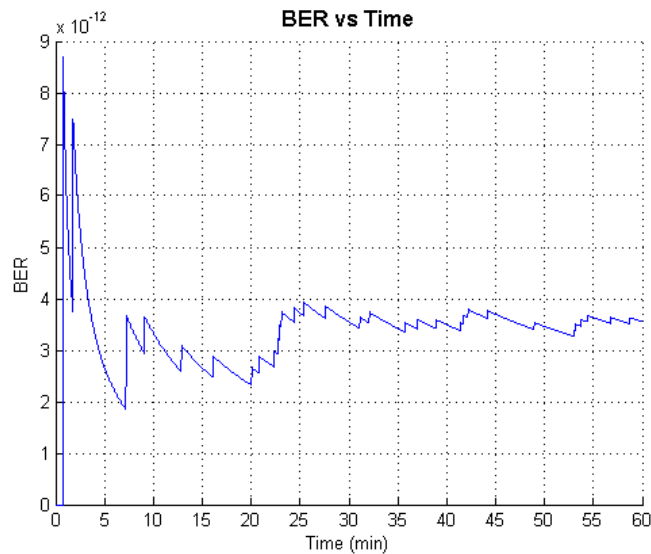


Figure 6.13: PRBS $2^{29} - 1$ BER vs Time, sharing clock generator.

Test E: PRBS2²⁹ – 1 at 2.5Gb/s for 1 hour and error insertion ratio disabled: This test was performed using independent reference clock for the transmitter and the receiver, but another RocketIO was assigned for the receiver in order to study the possible effect of capacitance on the development board. Figure 6.14 shows the results obtained, and figure 6.15 shows the graphic of the BER evolution vs time(created from the .log file using Matlab).

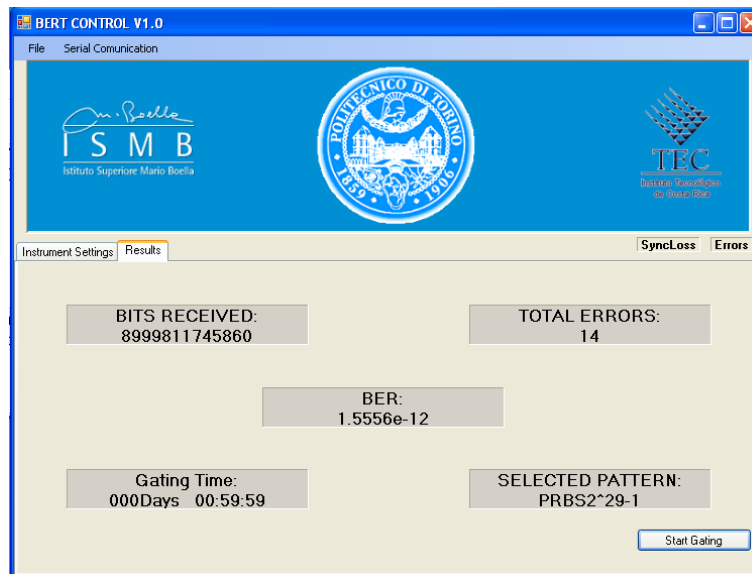


Figure 6.14: PRBS2²⁹ – 1 loopback test results, changing RocketIO channel.

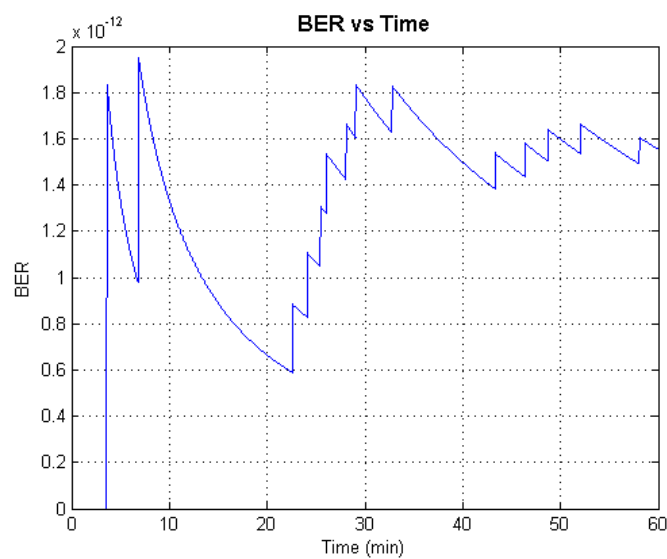


Figure 6.15: PRBS2²⁹ – 1 BER vs Time, changing RocketIO channel.

Test F: PRBS $2^{31} - 1$ at 2.5Gb/s for 5 hour using analog devices CDR to improve the signal, error insertion ratio disabled: Figure 6.16 shows the results obtained, and figure 6.17 shows the graphic of the BER evolution vs time(created from the .log file using Matlab).

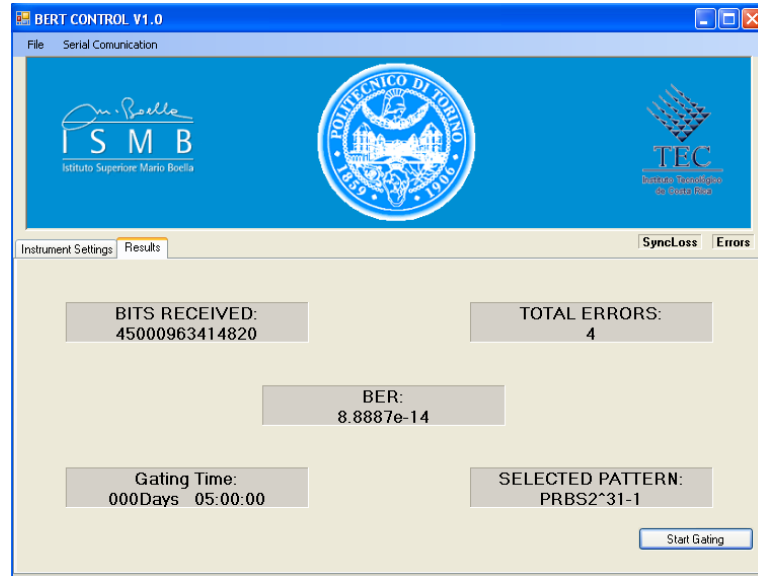


Figure 6.16: PRBS $2^{31} - 1$ loopback test results, when using external CDR.

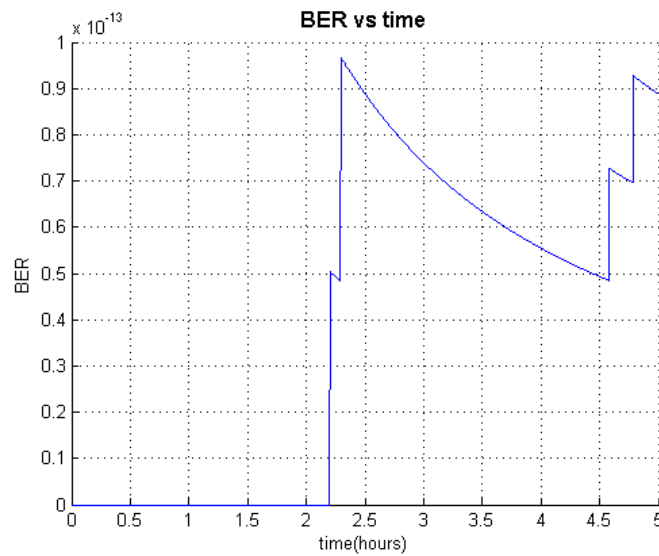


Figure 6.17: PRBS $2^{31} - 1$ BER vs Time, when using external CDR.

6.1.4 Fourth test results(Connected with BERT Anritsu commercial equipment):

The BER tester Anritsu was used to validate the function of the FPGA and the proper generation of the PRBS. All the PRBS implemented where tested, however only the PRBS $2^{31} - 1$ is shown since it presents the worst case scenario.

Test A: The PRBS $2^{31} - 1$ at 2.5Gb/s was selected in both, the FPGA's pattern generator and Anritsu's pattern generator, the error insertion ratio was disabled, and the gating was programed for 1 hour measure. Figure 6.18 shows the error free interval, figure 6.19 the BER measured, and finally figure 6.20 shows the result on the FPGA's error detector.



Figure 6.18: Error free interval measured using Anritsu's error detector.

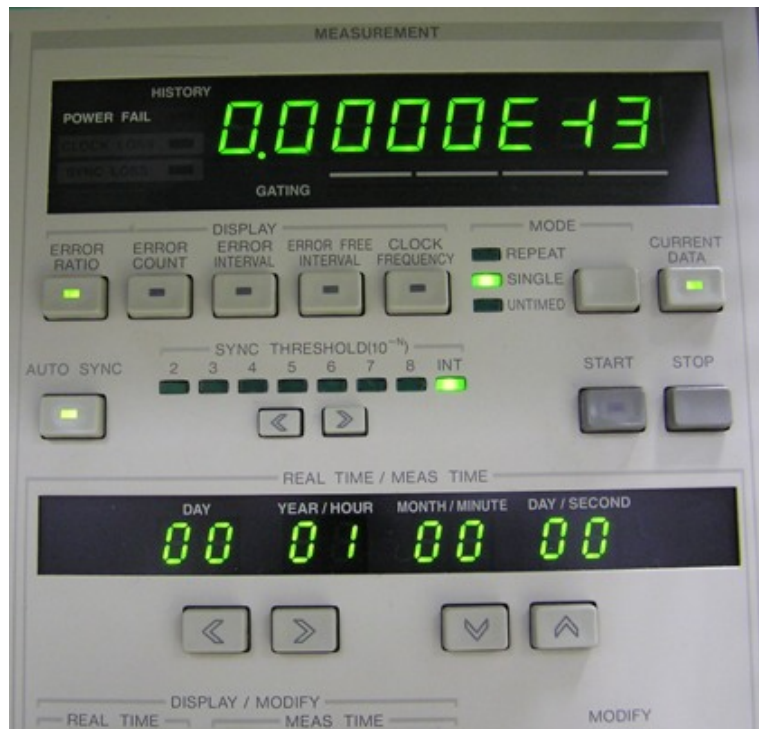


Figure 6.19: BER measured using Anritsu's error detector.

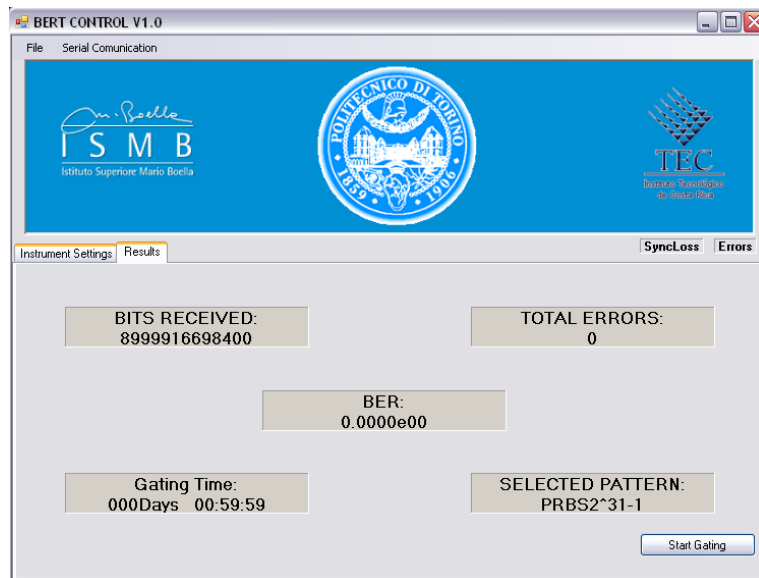


Figure 6.20: BER measured using the FPGA's error detector.

Test B: This test was performed to check the independence of the pattern generator and error detector, since each channel(pattern generator and error detector) is working at a different bitrate and different pattern. Also the accuracy of the measure and the error insertion ratio.

The FPGA's pattern generator settings where: PRBS $2^{15} - 1$, bitrate=2.25Gb/s and error insertion ratio of $1e-7$.

On Anritsu's pattern generator the settings where: PRBS $2^{31} - 1$, bitrate=2.5Gb/s and error insertion ratio of $1e-8$.

Figures 6.21, 6.22 and 6.23 show the results obtained with Anritsu's error detector and figure 6.24 shows the result obtained with the FPGA's error detector.



Figure 6.21: Bitrate measured using Anritsu's error detector.



Figure 6.22: Number of errors measured using Anritsu's error detector.



Figure 6.23: BER measured using Anritsu's error detector.

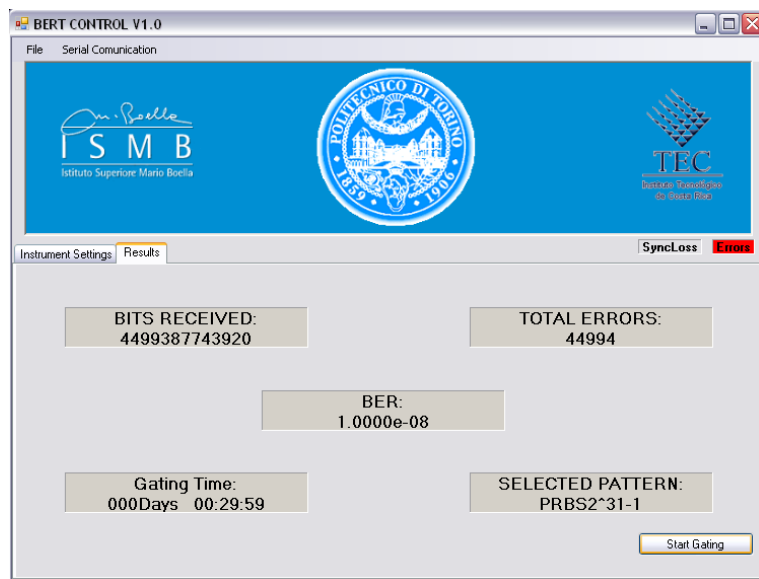


Figure 6.24: BER measured using the FPGA's error detector.

6.2 Analysis

6.2.1 Design phase

In the design phase the main problem was the maximum frequency that the hardware must support. To accomplish the timing constraints it was necessary to modify the hardware in order to add the pipeline registers.

The most difficult task in this thesis was related to the design of circuits that could work in a pipeline structure, due to their characteristics. The difficulty was to find the point where the hardware could be segmented so it wouldn't interfere with the operation flow, and that the process wouldn't take longer than 6.4ns. In order to achieve this some of the hardware had to be modified or optimized.

An example of these modifications was the error counter. In the bit to bit adder it was necessary to design a more complex structure, and despite it seemed that it was going to have a high propagation delay, this structure was faster than a chain of adders and was able to meet the timing constraint without the addition pipeline registers. The reason for this is that the ROMs are implemented using look up tables inside the FPGA which have a shorter access time than the logic path delay of the adder.

Another consideration was to insert the same number of pipeline registers on the error counter and bit counter in order to keep both branches balanced, so the precision of the instrument wouldn't be compromised.

6.2.2 Test phase

The first test was performed to analyze the behavior of the RocketIO™, the eye showed that the transceiver has a good amplitude, however, when the transmission bitrate is forced to the limit, the eye becomes deformed and because of this the probability that the system connected to the pattern generator(error detector or the FSO system) fails to recognize the logic level is considerably higher.

On the second test the clock and data recovery board from Analog Devices was used in order to improve the signal from the RocketIO™. It's important to notice that the eye in fact was improved, the eye amplitude isn't higher but the eye is more open which means that the transition time was reduced, approximately to 1/4 of the RocketIO time. However since the maximum operation frequency of this CDR is 2.8Gb/s the test at 3.125Gb/s was not possible this only affects the maximum bitrate that can be reached but since it's still over the 2.5Gb/s limit doesn't affect the system operation.

The third test was the back to back loop of the FPGA. Test A showed that for a PRBS²⁷ - 1 the BERT is error free; test B was performed to check the error insertion ratio and BERT's accuracy.

Test C on the other hand revealed problems with the long PRBS, there was a large number of errors in this test indicating that there was some problem with the RocketIO, since the problem appeared only with the long sequences (large number of consecutive ones), the first guess was that the problem could be related to the internal CDR of the RocketIO. Test D was performed to see the behavior of the RocketIO when the transmitter's and receiver's reference frequency was the same, thus reducing the stress of the internal CDR, but even with this change there were too many errors, so the problem with the CDR was discarded.

Since the errors were related to the length of the PRBS, the other possible reason was a pattern dependency problem due to the development's board capacitance. Test D was performed to determine the effect of the board's parasitic capacitance, the hardware was restored to the test A configuration with the difference that another RocketIO channel was chosen. This test showed a error reduction of 14 times. Which meant that the board has a design problem, since the board cannot be modified, the solution consist on the addition of external components to compensate the capacitance effects.

Finally test E was done using the CDR from analog devices in order to improve the signal, as it can be seen on figure 6.17 the number of errors was considerably reduced but didn't accomplish an error free test with the long PRBS. However it can be assured that if the BER measured is 1000 times bigger than the channel BER, the accuracy will be of $\pm 0.009 \cdot 10^{-11}$, and considering that the limit established to guarantee the reliability of a communications system is $1 \cdot 10^{-9}$ and the fact that the FOV system expected BER is on the order of $1 \cdot 10^{-12}$; the BERT can be used for the FSO system tests.

The fourth test demonstrated that the PRBS generated with the FPGA is compatible with commercial test equipment, and that the pattern generated was error free. Also the error detector of the FPGA showed an error free test. On test B showed the independence between the pattern generator and error detector, also the accuracy of the pattern generator's error insertion and the error detector BER measure.

Chapter 7

Conclusions and recommendations

7.1 Conclusions

- The frequency constraints of the Xilinx ISE software are an useful tool to accomplish the hardware optimization and error debugging.
- The system is capable of measuring the BER of any electrical communications system, including the FSO.
- The coupling between the BERT tester designed with the FPGA and the commercial one was successful, the synchronization work as expected and the accuracy of the measure was good.
- The internal registers of the BERT are capable of handling 187.18 years of data, so it's possible to perform a measure long enough to guarantee the confidence level higher of 90%.
- The RocketIOTM is not able to compensate the capacitance effects with long sequences of ones when bypassing the clock correction and alignment hardware, so it's necessary to add external electronic components to compensate this effects.

7.2 Recommendations

- A higher performance could be obtained by upgrading the communication's interface with the FPGA.

Bibliography

- [1] Valter Ferrero. 40 gbit/s electrical time domain pattern generator and ber measurements.
- [2] Integrated Circuit Systems INC. *700MHZ, Crystal oscillator-to-diferential LVDS frequency synthesizer ICS8442*, July 8 2004. Rev.C.
- [3] MAXIM Integrated Products. APPLICATION NOTE 1095: Statistical confidence levels for estimating BER probability. HFTA-05.0, Oct 26 2000.
- [4] MAXIM Integrated Products. *Pattern Creator/Converter Software User Manual*, August 2004. Rev 0.
- [5] International Telecommunication Union(ITU). ITU-T O.150: General requirements for instrumentation for performance measurements on digital transmission equipment., May 1995.
- [6] International Telecommunication Union(ITU). ITU-T O.150 Corrigendum1: General requirements for instrumentation for performance measurements on digital transmission equipment. Corrigendum 1, May 2002.
- [7] XilinxTM. *Libraries guide ISE 8.1i DCM*.
- [8] XilinxTM. *RocketIOTM Transceiver User Guide*, December 9 2004. UG024(v2.5).
- [9] XilinxTM. Xapp211(v1.2): PN generators using the SRL macro, Application note: Virtex series, Virtex-II series, and Spartan-II family, June 14 2004.
- [10] XilinxTM. Xapp661(v2.0.2): RocketIO transceiver Bit-Error Rate Tester, Application note: Virtex-II Pro family, May 24 2004.

Appendix

A.1 Glossary, abbreviations and symbols

FPGA: field programmable gate array.

FSO: free space optical communications.

BER: bit error rate.

BERT: bit error rate tester.

PRBS: pseudorandom bit sequence.

PC: personal computer.

ISL: intersatellite link.

LEO: low earth orbit.

RF: radiofrequency).

FOV: field of view.

VHDL: very high speed integrated circuit hardware description language.

ITU: international telecommunications union.

TX: transmitter.

RX: receiver.

CRC: cyclic redundancy check.

DCM: digital clock manager.

A.2 Measuring protocols

A.2.1 First test:

The first test consisted on connecting the FPGA with a 10GHz oscilloscope, that has special communication measuring capabilities, and allows the visualization of the eye diagram. The Measuring setup is showed in figure A.1.



Figure A.1 First test setup.

A.2.2 Second test:

The second test consisted on connecting transmission RocketIO™ with the input of the CDR evaluation board from analog devices, and the output of this board to the 10GHz oscilloscope. Figure A.2 shows an schematic of the second test setup.

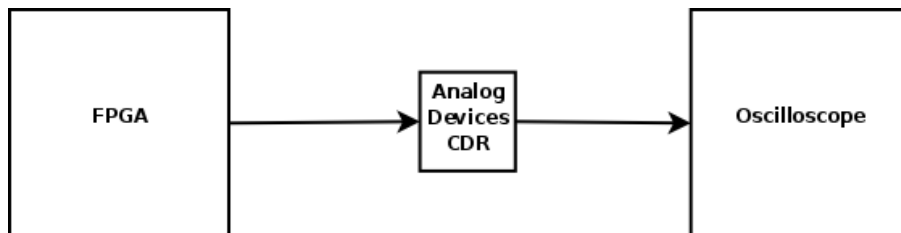


Figure A.2 Second test setup.

A.2.3 Third test:

Figure A.3 shows the setup used for the third test, back to back connection of the pattern generator and error detector.

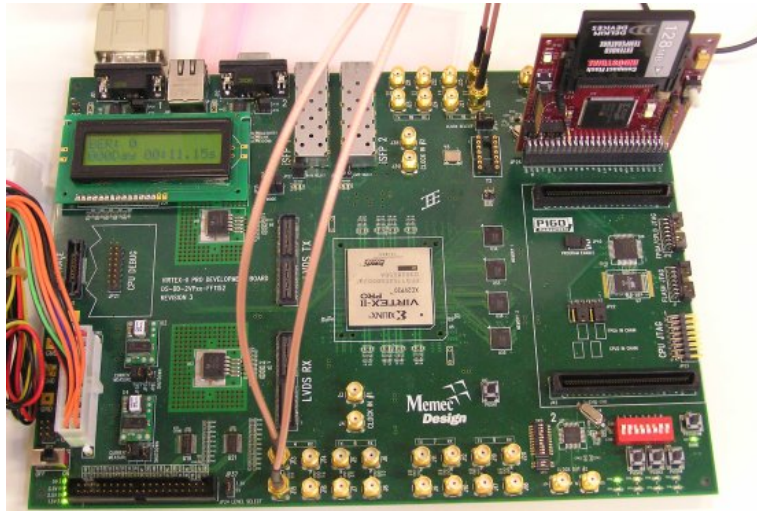


Figure A.3 Third test setup.

A.2.4 Fourth test:

Figure A.4 shows fourths test setup used, which was a cross connection between the FPGA BERT and the commercial one. As the commercial BERT's error detector needed an external clock source, the external CDR had to be used.

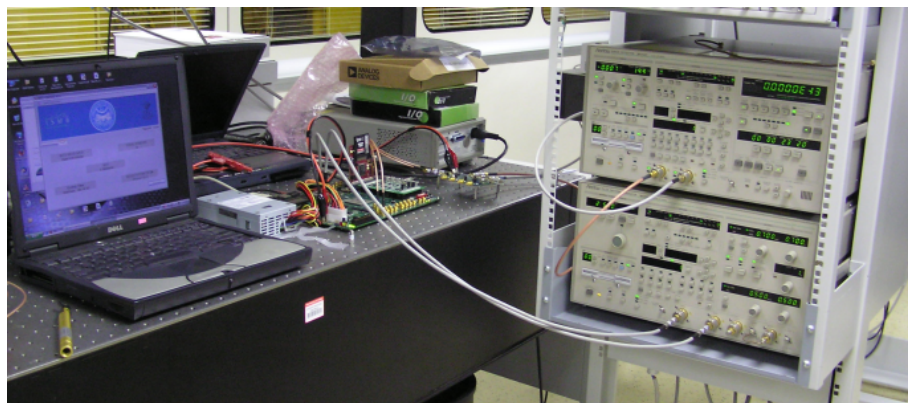


Figure A.4 Fourth test setup.

A.3 Maximum gating time calculus

To calculate the maximum gating time it's necessary to evaluate the worst case scenario, that is defined by the following conditions:

- Maximum transmission bitrate(3.125Gb/s, that is the maximum bitrate despite it cannot be reached due to board parasitic capacitance).
- Maximum bits that can be counted($2^{64} - 1$).

Given these conditions the maximum gating time($Gtime_{max}$) is given by equation A.1.

$$\begin{aligned}
 Gtime_{max} &= \frac{Maxbits}{maxbitrate} \quad (A.1) \\
 Gtime_{max} &= \frac{2^{64} - 1}{3.125Gb/s} \\
 Gtime_{max} &\approx 5.903 \cdot 10^9 s \\
 Gtime_{max} &\approx 187.18 years
 \end{aligned}$$

A.4 Test duration to guarantee a confidence level higher than 90%

To guarantee a confidence level higher than 90% it's necessary to transmit a certain number of bits calculated with equation 3.5, for this calculus is necessary to assume the worst case scenario which is the BER of $1 \cdot 10^{-9}$ and 5 errors received, with a confidence level of 95%.

$$n = -\frac{\ln(1 - 0.95)}{1 \cdot 10^{-9}} + \frac{\sum_{k=0}^5 \frac{(n \cdot 1 \cdot 10^{-9})^k}{k!}}{1 \cdot 10^{-9}}$$

$$\begin{aligned}n &\approx 10.513 \cdot 10^9 \\TT &= \frac{10.513 \cdot 10^9}{700 \text{ Mb/s}} \\TT &\approx 15.019 \text{ s}\end{aligned}$$

A.5 Serial port thread

```

private void ThreadTask()
{
while (true) //infinite loop
{
if (serialComP.IsOpen) //executes the reading process only if the serial
                        //port is open.
{
    if (serialComP.BytesToRead > 17) //Read the serial port only if the data
                                    //package constituted by 18 bytes is complete.
    {
        bool REC_ERR;
        byte[] buffer = new byte[17];

        int FLAGS = serialComP.ReadByte();//Reads the first byte received

        switch (FLAGS) //checks if the first byte is correct
                        //and process it's information
        {
            case 128:
                GatingStatus = true;
                SL = false;
                ERR = false;
                REC_ERR = false;
                break;

            case 129:
                GatingStatus = true;
                SL = false;
                ERR = true;
                REC_ERR = false;
                break;

            case 130:
                GatingStatus = true;
                SL = true;
                ERR = true;
                REC_ERR = false;
                break;

            case 131:
                GatingStatus = true;
                SL = true;
                ERR = true;
                REC_ERR = false;
                break;

            case 132:
                GatingStatus = false;
                SL = false;
                ERR = false;
                REC_ERR = false;
                break;

            case 133:
                GatingStatus = false;
                SL = false;
                ERR = true;
                REC_ERR = false;
                break;

            case 134:
                GatingStatus = false;
                SL = true;
                ERR = true;
                REC_ERR = false;
                break;

            case 135:
                GatingStatus = false;
                SL = true;
                ERR = true;
        }
    }
}
}
}

```

```

        REC_ERR = false;
        break;

default:
    SL = false;
    ERR = false;
    REC_ERR = true;
break;
}

if (REC_ERR == false)//reads buffer and process the data only if the
                        //first byte received is correct
{
    serialComp.Read(buffer,0,17);//Read the rest of package received

    int FLAGS2=Convert.ToInt32(buffer[16]);

    if (FLAGS2 == 170)//Checks last byte to determine if the data
                        //package was received properly
    {
        REC_ERR = false;
    }
    else
    {
        REC_ERR = true;
    }

    if (REC_ERR == false)//process data only if the package
                        //was received properly
    {
        uint NE8 = Convert.ToUInt32(buffer[0]);
        uint NE7 = Convert.ToUInt32(buffer[1]);
        uint NE6 = Convert.ToUInt32(buffer[2]);
        uint NE5 = Convert.ToUInt32(buffer[3]);
        uint NE4 = Convert.ToUInt32(buffer[4]);
        uint NE3 = Convert.ToUInt32(buffer[5]);
        uint NE2 = Convert.ToUInt32(buffer[6]);
        uint NE1 = Convert.ToUInt32(buffer[7]);
        uint RB8 = Convert.ToUInt32(buffer[8]);
        uint RB7 = Convert.ToUInt32(buffer[9]);
        uint RB6 = Convert.ToUInt32(buffer[10]);
        uint RB5 = Convert.ToUInt32(buffer[11]);
        uint RB4 = Convert.ToUInt32(buffer[12]);
        uint RB3 = Convert.ToUInt32(buffer[13]);
        uint RB2 = Convert.ToUInt32(buffer[14]);
        uint RB1 = Convert.ToUInt32(buffer[15]);

        //Calculates number of errors
        NET = (double)(NE1 + (256 * NE2) + (65536 * NE3) + (16777216 * NE4)
            + (4294967296 * NE5) + (1099511627776 * NE6)
            + (281474976710656 * NE7) + (72057594037927936 * NE8));

        //Calculates total number of received bytes
        RBT = (double)(RB1 + (256 * RB2) + (65536 * RB3) + (16777216 * RB4)
            + (4294967296 * RB5) + (1099511627776 * RB6)
            + (281474976710656 * RB7) + (72057594037927936 * RB8));

        //Calculates the BER
        BER = (float)(NET / RBT);

        if (GatingStatus)//if a gating is taking place, the results are written
                        //to the log file
        {
            tw.Write(Convert.ToString(NET));
            tw.Write(",");
            tw.WriteLine(Convert.ToString(RBT));
        }
    }
}
else
{

```

```
if (GatingStatus)//if a gating is taking place, but the results aren't
    //received properly a reception error message is written
    //to the log file
{
    tw.WriteLine("Reception Error");
}
SL = false;
ERR = false;
```

```
}
```

```
}
```

```
}  
}  
}
```