MICROELECTRONIC IMPLEMENTATION OF DICODE PPM SYSTEM EMPLOYING RS CODES

BASMAN MONTHER AL-NEDAWE

A thesis submitted to the University of Huddersfield

in partial fulfilment of the requirements for

the degree of Doctor of Philosophy

The University of Huddersfield

December 2014

Abstract

Optical fibre systems have played a key role in making possible the extraordinary growth in world-wide communications that has occurred in the last 25 years, and are vital in enabling the proliferating use of the Internet. Its high bandwidth capabilities, low attenuation characteristics, low cost, and immunity from the many disturbances that can afflict electrical wires and wireless communication links make it ideal for gigabit transmission and a major building block in the telecommunication infrastructure.

A number of different techniques are used for the transmission of digital information between the transmitter and receiver sides in optical fibre system. One type of coding scheme is Pulse Position Modulation (PPM) in which the location of one pulse during 2^M time slots is used to convey digital information from M bits. Although all the studies refer to advantages of PPM, it comes at a cost of large bandwidth and a complicated implementation. Therefore, variant PPM schemes have been proposed to transmit the data such as: Multiple Pulse Position Modulation (MPPM), Differential Pulse Position Modulation (DPPM), Pulse Interval Modulation (PIM), Digital Pulse Interval Modulation (DPIM), Dual Header Pulse Interval Modulation (DH-PIM), Dicode Pulse Position Modulation (DiPPM).

The DiPPM scheme has been considered as a solution for the bandwidth consumption issue that other existing PPM formats suffer from. This is because it has a line rate that is twice that of the original data rate. DiPPM can be efficiently implemented as it employs two slots to transmit one bit of pulse code modulation (PCM). A PCM conversion from logic zero to logic one provides a pulse in slot RESET (R) and from one to zero provides a pulse in slot SET (S). No pulse is transmitted if the PCM data is unvarying. Like other PPM schemes, DiPPM suffers from three types of pulse detection errors wrong slot, false alarm, and erasure.

The aim of this work was to build an error correction system, Reed Solomon (RS) code, which would overcome or reduce the error sources in the DiPPM system. An original mathematical program was developed using the Mathcad software to find the optimum RS parameters which can improve the DiPPM system error performance, number of photons and transmission efficiency. The results showed that the DiPPM system employing RS code offered an improvement over uncoded DiPPM of 5.12 dB, when RS operating at the optimum code rate of approximately ¾ and a codeword length of 2^5 symbols.

Moreover, the error performance of the uncoded DiPPM is compared with the DiPPM system employing maximum likelihood sequence detector (MLSD), and RS code in terms

of number of photons per pulse, transmission efficiency, and bandwidth expansion. The DiPPM with RS code offers superior performance compared to the uncoded DiPPM and DiPPM using MLSD, requiring only $4.5x10^3$ photons per pulse when operating at a bandwidth equal to or above 0.9 times the original data rate.

Further investigation took place on the DiPPM system employing RS code. A Matlab program and very high speed circuit Hardware Description language (VHDL) were developed to simulate the designed communication system. Simulation results were considered and agreed with the previous DiPPM theory. For the first time, this thesis presents the practical implementation for the DiPPM system employing RS code using Field Programmable Gate Array (FPGA).

Acknowledgements

Praise belongs to God, the Lord of Mercy and the Giver of Mercy, for his blessing that made this work possible and completed.

I would like to take the opportunity to express my sincere gratitude to Dr Martin M J N Sibley, my supervisor, for his continuous guidance and support throughout the research work and particularly during the writing up stage. His useful critiques and valuable suggestions have always been very much appreciated.

I would also like to thank Dr Peter J Mather, my second supervisor for his assistance with a software program for the research work.

My special thanks and appreciation must go to the Iraqi Ministry of Higher Education and Scientific Research, for awarding me the fee-waiver scholarship to purse this PhD. It would have been difficult for me to pursue a research degree without this great opportunity.

A very special thanks to my family for all their constant support and encouragement that made my research easier.

I can hardly overlook the co-operation, timely help and moral support extended by my friends, colleagues, and university staff. Acknowledgements are inherently endless and incomplete, and I request indulgence from many friendly and helpful people whom I could not name here, due to paucity of space.

Contents

List of Figures

List of Tables

List of abbreviations

Symbol	Definition
B	*Bit rate*
b	*The number of photons*
BCH	*Bose, Chaudhuri, and Hocquenghem codes*
c(X)	*The codeword polynomial*
CIRC	*Cross-Interleaved Reed Solomon Code*
DH-PIM	*dual header pulse interval modulation*
DiPPM	*Dicode Pulse Position Modulation*
d_{min}	*Code minimum distance*
DPIM	*digital pulse interval modulation*
DPPM	*Differential Pulse Position Modulation*
DVB	*Digital Video Broadcasting*
FEC	*Forward Error Correction*
FPGA	*Field Programmable Gate Array*
g(X)	*generator polynomial*
k	*Message symbol number*
k_B	*Bandwidth factor*
m	*Number of bits per symbol*
m(X)	*Message polynomial*
MLSD	*Maximum Likelihood Sequence Detection*
MPPM	*Multiple Pulse Position Modulation*
n_{DiPPM}	*DiPPM maximum number of consecutive like symbols*
n_{RS}	*RS Codeword symbol number*
$N_o(t)^2$	*The mean square noise of the receiver*
OOK	*On Off keying*
p(X)	*parity polynomial*
PCM	*Pulse Code Modulation*
PDD	*Proportional-Derivative-Delay*
P_e	*Equivalent PCM probability of error*
P_E	*The probability of Reed Solomon error*
P_{eb}	*The average binary error probability*
P_{er}	*The probability of an erasure error at DiPPM*
PER	*Packet Error Rate*
P_{es}	*The probability for wrong slot error*

P_f	*The probability of a false alarm error.*
PIM	*pulse interval modulation*
POF	*Plastic Optical Fibre*
PPM	*Pulse Position Modulation*
PSD	*Power Spectral Density*
r	*Code rate*
R	*DiPPM Reset pulse*
RS	*Reed Solomon Codes*
S	*DiPPM Set pulse*
t	*Number of error symbols that RS can correct*
T_b	*The PCM bit-time*
v	*The threshold variable*
v_d	*The threshold crossing voltage*
VHDL	*Very High Speed Integrated Circuits Hardware Description Language*
v_{pk}	*The peak voltage of the signal*
ρ	*The transmission efficiency*
ω_B	*Filter bandwidth*

Introduction

1.1. Optical Communications

Optical communication represents the transfer of information in the form of light signals. There are transmitters that transfer the desired communication message and this communication often takes place through optical fibres. The major elements necessary to allow this communication include a modulator and demodulator, a transmitter and receiver for transmitting and receiving the communicated message, and a channel for the transfer of the data (Janssen, 2014). The information or signal that is passed through optical communications can take place in both analog and digital formats with the signal being converted into a form which is compatible with the system. Often an A/D converter is used for the transmission of the signal whereas such a block is not required as the data already exists in the digital form (Bagad & Dhotre, 2009).

1.1.1. Lasers

The term LASER is an acronym for Light Amplification by Stimulated Emission of Radiation. The light that is emitted by the laser is coherent. The laser has a spatial coherence and this property of a laser allows the laser beam to remain narrow over long distances (Svelto, 2010). These special properties of lasers make them ideal for a number of situations like optical disk driver, barcode scanner, printer, optical communication, etc. The principle on which a laser works is explained by quantum physics. According to quantum physics an atom stays in discrete energy levels (Al-Azzawi, 2006). When the atom is excited it moves to state of higher energy level. If an incident photon is sent to this atom in the excited state, then the atom falls to the lower energy level and emits a second photon that is exactly similar to the first photon (Agrawal, 2010). In designing the laser this principle is utilized and a large number of excited atoms are placed within two mirrors. When the first photon is sent to the excited atom a chain reaction begins and a large number of identical photons are created which are then released as a single coherent beam of laser (Thyagarajan & Ghatak, 2010).

1.1.2. Optical fibre Transmitter

The main part of the optical transmitter is the light wave source. This source must have minimum characteristics requirements in order to be valid as a light source. According to Sibley (Sibley, 1995), the light source should operate with a wide range of temperature for a long time in order to be reliable. He also stated that the source should operate

within one window of a wavelength which is suitable with the fibre and that the output spectrum must be narrow in order to reduce the material dispersion of the fibre link. Moreover, the light wave source can couple large amount of power by reducing the emitting area of the device. All these parameters are essential considerations in selecting a light wave source in fibre optics communication (Sibley, 1995).

The principal light sources used for fibre optic communications applications are heterojunction semiconductor laser diodes (also referred to as injection laser diodes or ILDs) and light-emitting diodes (LEDs). A heterojunction consists of two adjoining semiconductor materials with different band-gap energies. These devices are suitable for fibre transmission systems because they have adequate output power for a wide range of applications, their optical power output can be directly modulated by varying the input current to the device, they have a high efficiency, and their dimensional characteristics are compatible with those of the optical fibre. Comprehensive treatments of the major aspects of LEDs and laser diodes are presented in various books and journals (Sibley, 1995; Keiser, 2000).

The selection of LEDs and ILDs depends on the application itself, where each one has major difference than the other. Laser output light is coherent where the light is produced in an optical resonant cavity and has spatial and temporal coherence; all of which makes the output light from laser source is very directional and highly mono-coherent. On the other hand, LEDs are incoherent sources where there is no cavity for wavelength selectivity (Keiser, 2010).

Source selection must have certain factors in respect to the fibre, these factors should be seriously considered as follows;

- Geometry and attenuation as function in wavelength.
- Group delay distortion that limits the bandwidth.
- Spectrum width, radiation pattern, and modulation capability.

Sibley (Sibley, 1995) identified the differences between semiconductor laser diodes (SLDs) and light emitting diodes (LEDs) in several ways: SLDs emit light by stimulated emission, whereas LEDs emit light spontaneously; the application of a constant current is required by a laser diode to preserve stimulated emission; the output is more directional; and the response time is faster.

1.1.3. Optical fibre Receiver

The optical receiver consists of a photodetector, an amplifier, and signal processing circuitry. Therefore, the basic structure of an optical receiver consists of: a photodiode, a low-noise pre-amplifier, the front-end, feeds further amplification stages, the post-amplifier, before filtering. The following figure shows the basic structure of an optical receiver (Sibley, 1995; Keiser, 2010):

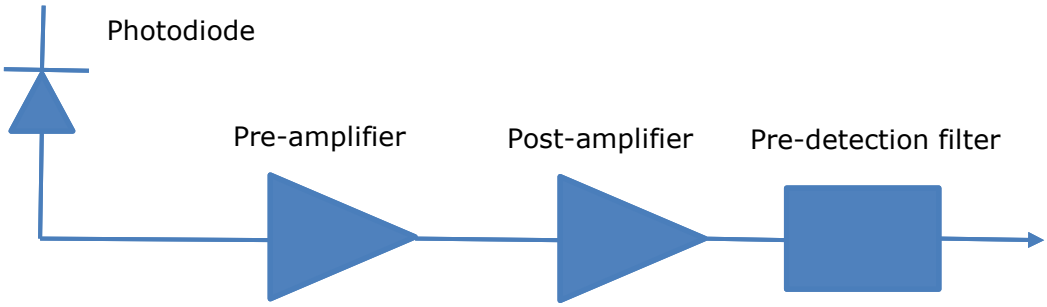


Figure 1.1 The basic structure of an optical receiver (Sibley, 1995)

The main part of the optical receiver is the photo detector which acts as a demodulator converting the optical signal into an electrical signal. This easily happens when the photo detector has the ability to detect and absorb the light photons. There are minimum performance requirements which the photo detector should have in order to perform this job (Sibley, 1995).

The photo detector should have a high sensitivity at the operating wave length and a high fidelity that allows it to have a linear characteristic with the optical signal in the analogue systems. As well, high quantum efficiency is a necessary parameter to produce the maximum electric signal from the input optical power that enables the receiver to have larger electric response to the input signal. In addition, as the optical bandwidth is increasing, the receiver should have short response time to obtain this bandwidth which expected to achieve terahertz in the future. The size of the photo detector should be small in order to couple with the fibre. The noise should be as low as possible in the photo detector and all circuitry should have low noise. However, all of these requirements should be satisfied while the receiver is being designed, the cost matter and reliability issues should be considered (Senior, 2008).

There are many types of photo detector, such as PIN photodiode and the avalanche photodiode (APD), they are different in term of operation and also in materials, but the selection of these types must be decided according to these performance requirements

of the application. Besides, the full receiver structure depends mainly on the application (Sibley, 1995).

1.1.4. Optical fibre

Optical fibres are actually used as a channel that is used to transmit optical signals from the transmitter to the receiver stations. In 1964, the use of glass in optical fibre communication for communicating over long distances was first stated by Charles K. Kao and it has come into use in the present generation in the form of fibre optic communication. Optical fibre is considered as a cylindrical dielectric waveguide which is used to transmit light along the axis of the waveguide through a process that is known as total internal reflection. The different types of favour that exists are single mode fibre and multimode fibre (Crisp, 2005).

Single mode fibres are fabricated to support one mode field to propagate where the core diameter plays this role. The single mode fibre has the advantage that the model dispersion, which occurs as a result between the delayed modes is avoided. Thus the single mode fibre is employed in telecommunication due to greater bandwidth and lowest losses (Senior, 2008). Single mode fibre is divided into step index and graded index fibres. In the modern communication system there are three classes of single mode fibre that are used. In includes NDSF or non-dispersion shifted fibre, non-zero dispersion shifted fibre and dispersion shifted fibre (Downing, 2004). Multimodal fibres are those which allows for the transmission of a number of modes of light at the same time. There are two types of multimodal fibre that are step-index and graded index multimodal fibres (Hecht, 2004).

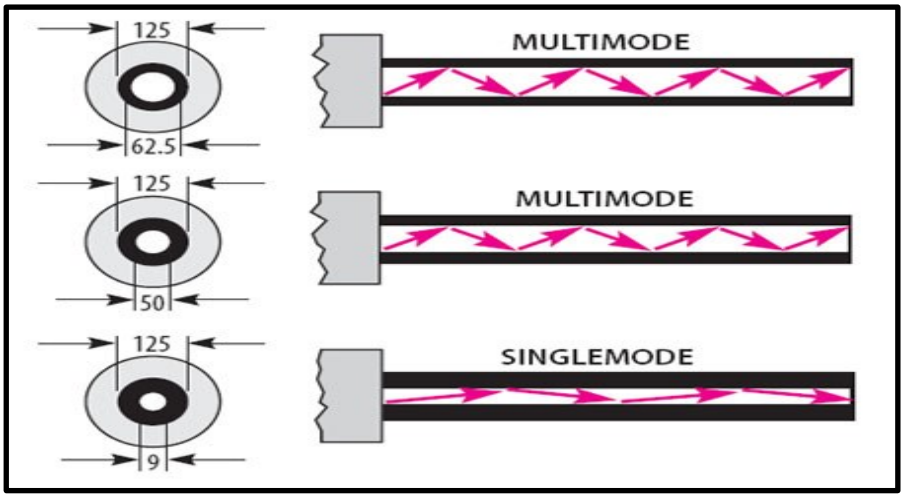


Figure 1.2 types of optical fibre (Sibley, 1995)

1.1.5. Principle of dispersion

In an optical fibre it is found that a short optical pulse actually widens after it propagates in an optical fibre. This is a phenomenon that is caused by dispersion. Two types of dispersion that exists are explained below.

Intermodal dispersion: This is caused by the differences in the group velocities that exist in between the various modes that propagate within an optical fibre (Anderson, Johnson & Bell, 2004). The intermodal dispersion is actually a phenomenon which leads to the fact that the group velocity of light that is propagating in a multimode fibre not only depends on optical frequency but also on the propagation mode that is involved.

Intramodal dispersion- This is in turn caused by the fact that the refractive index of fibre core is dependent on the wavelength of light. This leads to a velocity difference is created in between the spectral components of a source of light.

1.1.6. Wave division multiplexing principle

The term wavelength division multiplexing (WDM) refers to the technology, which is used to multiplex number of optical carrier single on a single piece of optical fibre. It is done through the use of different colours of laser light in order to carry different signals. A multiplexer is used at the transmitter end to join the signals together and a demultiplexer is placed at the receiver in order to split the signals (Sivalingam & Subramaniam, 2006). There are two types of wavelength division multiplexing. They are coarse wavelength division multiplexing and dense wavelength division multiplexing. DWDM refers to the fact that it helps in transmitting more channels which are closely spaced. On the other hand, in case of CWDM the number of channels is less.

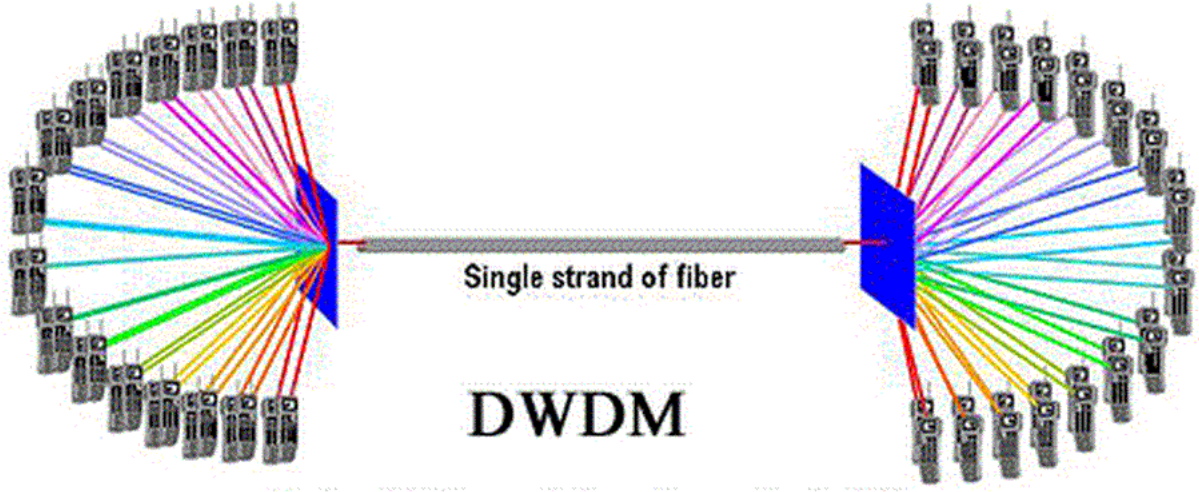


Figure 1.3 40 virtual high speed channels per physical fibre (Infocellar, 2015)

1.1.7. Coding Schemes

Particular coding schemes are used in optical communications such as PCM, PPM, Digital PPM, Multiple PPM, PIM, Dicode PPM.

PCM – PCM or pulse-code modulation is a method based on digital encoding. The baseband signal in the system is quantised and sampled and there is a series of bits that encodes the sample. In optical communications a light is turned on and off using binary signals. This is a good choice for optical communications, particularly when a laser diode has to be utilized owing to presence of inherent nonlinearity in the source of the optics (Bandyopadhyay, 2014).

PPM – PPM or pulse position modulation is used for optical communications where the code scheme involves a bit time divided into two slots. Data can either be 1 or 0. If it is 1, the pulse gets transmitted in the first bit time slot. If the data is 0, then the pulse is generated in the second slot of the bit time. Expansion of PPM is also possible and proves to be efficient for optical communications (Alexander, 1997).

Digital PPM – PPM can also be used in the digital transmission of information. It is a variation of the PPM coding, which can transmit data irrespective of the presence of time. If delays occur in the transmission, they take reference from falling edge of the pulse that was earlier transmitted (Lazaridis, 2011).

Multiple PPM and Dicode PPM – Dicode PPM and Multiple PPM are considered as the most bandwidth efficient codes for PPM transmission. These codes offer a good sensitivity without the large increase in bandwidth given by digital PPM (Nikolaidis, 2008).

PIM – PIM or pulse interval modulation forms a part of a synchronous PTM technique, the coding schemes of which have a fixed structure for their symbol. PTM or pulse time modulation represents a transmission scheme where modulation of the carrier results in production of several pulses whose position and width in a frame vary according to the modulation. (Wilson and Ghassemlooy, 1993). Thus, there are variations in the length of the symbol and can be measured from the content or information that is carried by the symbol. Therefore, it enables effective transmission of the signals in optical communications (Herceg, Svedek & Matic, 2010).

1.2. Error Correction

Digital electronic data-storage systems are widely used in the recent times. Unfortunately, errors can take place and so error correction methods are essential. A

most widely developed mechanism includes error detection and correction coding theory. Detection and recovery are essential for digital systems, hence the mechanisms being developed. The types of codes are varied and depend on the capabilities of corrections required, the efficiency of the codes, and the level of complexity associated with the coding and decoding methods (Fleetwood & Schrimpf, 2004).

The first class of linear block codes that were developed for error correction purposes included the Hamming codes. Digital communications and data-storage systems have widely made use of these Hamming codes owing to the availability of variations. A single bit correction and double bit detection were provided by Hamming codes. While error correction methods are available, it has to be noted that the processes of coding are significantly complex in nature (Fleetwood & Schrimpf, 2004).

Reed Solomon error correction codes are used for the purpose of correcting errors in digital systems. The Reed Solomon codes are widely used to correct the errors that occur in optical recording systems (McDaniel & Victora, 1995) and in digital TV transmissions. Reed Solomon Codes represent BCH codes, their length, requiring larger sizes of the field. However, the codes have significant importance, in error correction. These codes are used by compact disc players effectively for error correction needs (Betten et al, 2006).

Based on blocks, the Reed Solomon codes are used in several applications that include digital communications and data storages. The following are some of the most effective applications of Reed Solomon codes (Riley & Richardson, 1998):

- Devices for storing information – these are generally devices such as tapes, compact disks, DVDs, barcodes, and other such devices
- Communication links or devices that are mobile or wireless such as mobile phones, microwave links, and so on
- Devices and networks involved in satellite data transmissions

1.3. Design Automation

Design Automation (DA) is the method by which steps in the design of integrated circuits can be automated. VHDL represents very high speed integrated circuit describing a hardware description language – HDL. The purpose of creating VHDL was to make hardware designs portable for use and VHDL could be used as a stimulator and developed for this purpose. VHDL is a tool for design-automation. The digital systems can take advantage of this tool at their different levels of generalisation and substantiation of functionality. Test data that is generated can also be verified using the

design automation tool. Based on such verification, the hardware can be installed in the systems (Kaur, 2011).

Another effective design automation tool is the FPGA or field programmable gate array. Embedded processors are offered by manufacturers of FPGA which can be customized for interfacing with the logic fabric of FPGA. The two types of FPGA embedded processors are a soft embedded processor and a hard embedded processor. The soft processors have features of configuration including caches and registered sizes of files, blocks of RAM/ROM, and necessary instructions for customization. Availability of these processors is in the form of description language and the hard processors are embedded on the FPGA in a dedicated form of hardware (Alpert, Mehta & Sapatnekar, 2008).

1.4. Aims and Objectives

Dicode pulse position modulation (DiPPM) systems suffer from three types of pulse detection errors: wrong slots, false alarm and erasure. The main aim of this work was to design error correction system, Reed Solomon codes (RS), which will overcome or reduce the error sources in the DiPPM system. Investigations undertaken and intermediate objectives fulfilled during PhD research:

- To find the Reed Solomon (RS) optimum parameters, that give lower number of photons and higher transmission efficiency, by developing a system Mathcad program. This involves computing the minimum number of photons and the transmission efficiency to determine the effect of adding a RS system to the DiPPM system. The two detection methods, slope detection and central decision detection, will be used to check if the RS optimum parameters are going to be compatible.
- To verify whether employing RS code with the DiPPM system will improve the system performance, with the RS working at optimum parameters. This will be done by comparing the uncoded DiPPM system, the DiPPM employing maximum likelihood sequence detection (MLSD) system, and the DiPPM employing RS system in terms of the number of photons and transmission efficiency.
- To confirm whether the RS system will repair the DiPPM system errors that occur during the transmission process. This will be ascertained by developing a Matlab program to simulate the system error performance.
- To design a schematic for the DiPPM system working with RS code. Each part of the system will be described using the VHDL. The optimum RS parameters will be used to design the system. The simulation result of each part of the system will be compared to the theoretical to agree that the system working correctly.

- To design a test bench environment for the schematic system. The erasure only, error only, and erasure and error will be applied on the designed system. This will be achieved by using Modelsim_Altera software to prove whether the system has the ability to detect and correct the mentioned types of error and the system limitation.
- To implement the designed system on FPGA, and connect the system (transmitter and receiver) via an optical fibre system. This will be realised by synthesising the parts of the system on an FPGA and building an optical transmitter and receiver. The practical output results will be compared to the simulation results to validate the system.

1.5. Thesis Layout

The rest of this thesis is organised into eight chapters as follows:

In chapter two, a literature review is presented that considers coding schemes in an optical fibre system in terms of characteristics, modulation, and transmission. Previous researches in DiPPM will be considered to illustrate system implementation and performance analysis. A PPM documentation for RS decoding is clarified that involved fundamentals of RS codes, encoding and decoding. Finally, a review on FPGA is given regarding applications and VHDL.

Chapter three, considers explanation of the DiPPM system theory. The errors affecting DiPPM, wrong slot errors, erasure errors, and false alarm errors, are described in this chapter. The DiPPM coder and decoder previous practical implementation are also shown.

Chapter four, investigates employing the RS code with the dicode pulse position modulation (DiPPM) to find the precise characteristics of the RS code that minimizes the errors in the DiPPM. The non-coded dicode pulse position modulation which applies MLSD and the RS coding paradigms are compared with regards to the number of photons which are contained in each pulse and the transmission efficiency. Moreover, the coded DiPPM system is compared with the coded PCM system.

In chapter five, Matlab software has been used to simulate the DiPPM with the RS code system. The simulation was developed through three versions. Although there was a Matlab simulation of the DiPPM system, a new version of DiPPM (coder & decoder)

simulation has been presented in this chapter. In the second version, the RS system has been employed with the DiPPM system in order to overcome the errors that affect the system. In the third version, the noise is added to channel to generate the errors by varying signal to noise ratio. In the third version, a PCM binary sequence was replaced by a picture's data to analyse the transmission performance of the system.

In chapter six, a very high speed integrated circuit (VHSIC) hardware description language (HDL) source code for the DiPPM system employing (31,23) RS error correcting code system is given. A schematic and a full block description of the system are shown. Modelsim_Altera version (6.5b) software is used to simulate each part of the system.

In chapter seven, the results of three test bench environments, erasure only, error only, and erasure and error, are presented. A Modelsim_Altera version (6.5b) software is used to simulate the system. The system has shown that it has the ability to detect and correct erasure and error symbols when they are within its limitation.

Chapter eight, presents a practical implementation of the designed system using Altera Quartus II software, and Cyclone III Field Programmable Gate Array (FPGA) based DSP development board. The design for the optical system transceiver is demonstrated as well.

Chapter nine, concludes the work presented in this thesis, highlighting the original contribution of the designed system and suggesting further work to be done.

1.6. Original Work Contributions

The author has

- Developed a Mathcad program for the DiPPM pulse detection by using the slope detection method, in order to compute the number of photons for the detected pulse after added RS codes. The number of photons was computed for different normalised bandwidth.
- Developed a Mathcad program for the DiPPM pulse detection by using the central detection method, in order to compute the number of photons for the detected signal in each pulse after added RS codes. The number of photons was computed for different normalise bandwidth.
- Calculated the minimum number of photons for each pulse of the uncoded DiPPM system and a DiPPM system employing RS code for a different normalised bandwidth at a different code rate and code length. A mathematical formula of the DiPPM system was derived to calculate the system transmission efficiency for

the uncoded and coded system. The simulation results have shown that the use of RS code can greatly increase the transmission efficiency of DiPPM by reducing the number of photons. The DiPPM system employing RS code offers a 5.12 dB improvement over the uncoded system when RS code operates at the optimum code rate of (3/4).

- Developed a Matlab program to simulate the DiPPM system with the RS code system. The simulation was developed through three versions. First version was the DiPPM (coder & decoder) system simulation; the second version was the DiPPM system employing RS code in order to prevent the errors that affect the system; the third version was data from a picture to analyse the transmission performance of the system.
- Developed a very high speed integrated circuit (VHSIC) hardware description language (HDL) source code for the DiPPM system employing (31,23) RS error correcting code system. A schematic and a full block description of the system are given. Modelsim_Altera version (6.5b) software is used to simulate each part of the system.
- Tested three test bench environments, erasure only, error only, and erasure and error, on the designed system. A Modelsim_Altera version (6.5b) software was used to simulate the system. The system has shown that it has the ability to detect and correct erasure and error symbols when they not overcome its limitation.
- Established a practical implementation of the designed system by using Altera Quartus II software, and Cyclone III Field Programmable Gate Array (FPGA) based DSP development board. The implementation of the optical system transceiver is done as well.

Literature Review

2.1. Introduction

This section of the thesis is an extensive literature review concerned with the use of coding schemes, Reed Solomon codes and dicode pulse position modulation (PPM).

The topics that will be reviewed in this chapter include**,**

- Coding schemes in optical fibre.
  - Characteristics, modulation, transmission.
- Dicode pulse position modulation.
  - Implementation, performance analysis.
- PPM employing Reed Solomon code.
  - Reed Solomon codes, encoding and decoding.
- Field programmable gate array.
  - VHDL, applications.

2.2. Coding Schemes in Optical Fibre

Transmission of data can occur either through analog transmission, digital transmission, and digital baseband, line coding, transmission. In the case of analog and digital transmission of data, carrier transmission is used, whereas with digital baseband transmission, a digital bit stream over a baseband channel is transfered. This implies that logic signals are sent for low and high levels of light. For low light level the logic signal is 0 and for high level of light the signal is 1. Sometimes a certain density of the transmission is obtained through data coding that can be applied in the process (Goff, 2002). A good balance of 0s and 1s is offered by line coding schemes as discussed by Senior, and Jamro (2009). FEC refers to Forward Error Correction the method of which is used for correcting errors in the system of communication and forms an essential strategy developed within the line code. There are several line codes and schemes that can be developed in order to accomplish FEC in optical fibre systems of communication (Senior & Jamro, 2009).

A large number of studies have investigated PPM and its development for use in optical fibre communication system. During the past 40 years much more information has become available on the digital PPM scheme (Gagliardi, & Ling, 1986; Davidson, & Sun, 1989; Sibley, 1987; Elmirghani, Cryan, & Clayton, 1992a, 1992b, 1992c). Analysis of digital PPM over optical fibre channel was first carried out in 1980 (Elmirghani, Cryan, &

Clayton, 1992a). The performance of the optical fibre digital system using direct detection and coherent detection PIN-FET optical receivers for Gaussian received pulse shape was reported by Dolinar, Garret (Karp, Gagliaridi, 1969; Gol'dsteyn, & Frezinskiy, 1978; Dolinar, 1983). They demonstrated that the PIN-FET PPM receiver gives a sensitivity of 10 to 12 dB more than that of pulse code modulation (PCM).

In 1988, digital PPM over optical fibre was first demonstrated by Calvert (Calvert, Sibley, & Unwin, 1988), by means of a theoretical model based on a modified Garrett analysis. His results showed that the sensitivity of a digital PPM receiver outperformed an equivalent PCM system by *4.2 dB* when the fibre bandwidth is several times that required by PCM. Although all the studies refer to the advantages of PPM, it comes at a cost of large bandwidth and a complicated implementation. Therefore, many PPM variant schemes have been derived or modified to transmit the data such as:

- Multiple Pulse Position Modulation (MPPM) (Lee, & Schroeder, 1977).
- Differential Pulse Position Modulation (DPPM) (Shirokov, & Bukhinnik, 1984).
- Pulse interval modulation (PIM) (Gol'dsteyn, & Frezinskiy, 1978).
- Digital pulse interval modulation (DPIM) (Ghassemlooy, & Hayes, 2000).
- Dual header pulse interval modulation (DH-PIM) (Aldibbiat et al, 2002).
- Dicode Pulse Position Modulation (Sibley, 2003).

These schemes were proposed to reduce the bandwidth expansion inherent in digital PPM (Sibley, 2012).

2.2.1. Characteristics of Line Coding

Line coding is generally necessary for data passing through a communication channel and has certain characteristics of its own. Some of the particular aspects of line coding are signal level versus data level, pulse rate versus bit rate, DC components, and self-synchronization (Forouzan & Fegan, 2003).

Signal level versus data level – As far as the digital signal is concerned, the number of values is limited. Few of these values can be used for the purpose of data representation. The number of levels that are allowed through a specific transmit signal are referred to as signal level, whereas the levels that perform the representation of data are referred to as data levels (Forouzan & Fegan, 2003).

Pulse rate versus bit rate – The number of pulses transmitted per second is referred as the pulse rate. On the other hand, the number of bits per second represents the bit rate

for the transmission of the signal. In cases where a signal pulse defines only a single bit, the pulse rate and bit rate are considered to be the same (Forouzan & Fegan, 2003).

DC Components – In case the transmission signal has to pass through a communication channel that does not allow DC components, signal must be modified so that the DC content is zero. Thus, there can be two different line coding schemes depending on the channel for the DC components one with the DC component, and another without the DC component (Forouzan & Fegan, 2003).

Self-synchronization – A digital signal which is self-synchronizing in nature includes information related to timing for the data that is being transmitted through the communication system. It is possible to achieve this if the signals have transitions that can create alerts for the receipt of the signals in the beginning, middle, and end of the transmission (Forouzan & Fegan, 2003).

Some of the other characteristics of line coding that are essentially associated with optical fibres and its role in communications systems, also considering the difference between each line code, include power spectral density (PSD), regularity on the transitions of the signals, immunity of noise, and the capability to detect errors. In many cases, researchers prefer experimenting with applications of small or no DC content (Guimaeres, 2002, 2010).

The energy that is carried by the line codes varies from one code to another. Hence, depending on the information carried and communicated by the line codes, the energy of the codes may be determined. Also, depending on the energy carried by the line codes, the noise immunity of the codes also varies and the more energy a line code can carry would lead to more immunity of noise for that particular code. Synchronization of the system is obtained through the regularity of the transitions of the signals. If the noise immunity can be made high, then the chances of errors in the bits reduce significantly, enhancing the system of communications. Level codes and transition codes are the two major types of binary line codes. While the level codes are responsible for carrying information depending on the level of the voltage; the transition codes are responsible for carrying information that are in the form of changes in the levels. Also, line codes may be bipolar or unipolar depending on the use of bipolar or unipolar pulses (Guimaeres, 2002, 2010).

2.2.2. Rate Adaptive Modulation and Coding for Optical Fibre Transmission Systems

As Nam (2006) discussed, wireless communications channels are generally random or erratic nature. Thus, it often becomes difficult to rely completely on the system or obtain high rate of data transmission. Hence, reliability of the system needs to be accomplished by some means. Adaptive modulation refers to a way through which the fading effects of the wireless channels can be suitably accommodated where, depending on the quality of the channel of communication, a variable constellation is used for the purpose of transmission. Adaptive modulation is also therefore referred to as near-capacity technique for achievement of effective signal transmission (Nam, 2006).

Gho, and Kahn (2012) have proposed a scheme of rate adaptive transmission depending on a variable rate correction method involving FEC codes (FEC-forward error correction) along with constellations of different sizes at a fixed rate of symbol, being capable of quantifying the variation of the rates of bit with distances as can be achieved. Such a system could be evaluated with the use of a single channel transmission based on inline distribution return. Researchers have focused on achieving extensions of the distances of transmissions where the rate adaptive modulation case has been considered. There are different constellations for different orders of the modulation. If a symbol rate is given as R_s, an RS-RS rate of code given by $r_C=k/n$, repetition rate of code given by $r_R=1/f_R$. The repetition factor f_R denotes the number of repetitions of each bit from the output of the inner RS encoder, and ranges from 1 to 4. The rate of line code given by r_L, and order of modulation given by M, then the data rate given by R_b can be calculated by the following equation:

$$R_b = 2r_L r_C r_R R_s \log_2 M \tag{2.1}$$

considering an assumption of the polarization multiplexing transmission (Gho and Kahn, 2012).

The working of adaptive modulation systems depends on adaption of modulation parameters, which responds to the path of propagation associated with the characteristics and traffic control of the path. The modulations can be either slow adaptive modulation or fast adaptive modulation, depends on the scheme of the modulation. A general outline of the adaptive modulation systems can be presented, figure 2.1, as given by Sasaoka (2000):

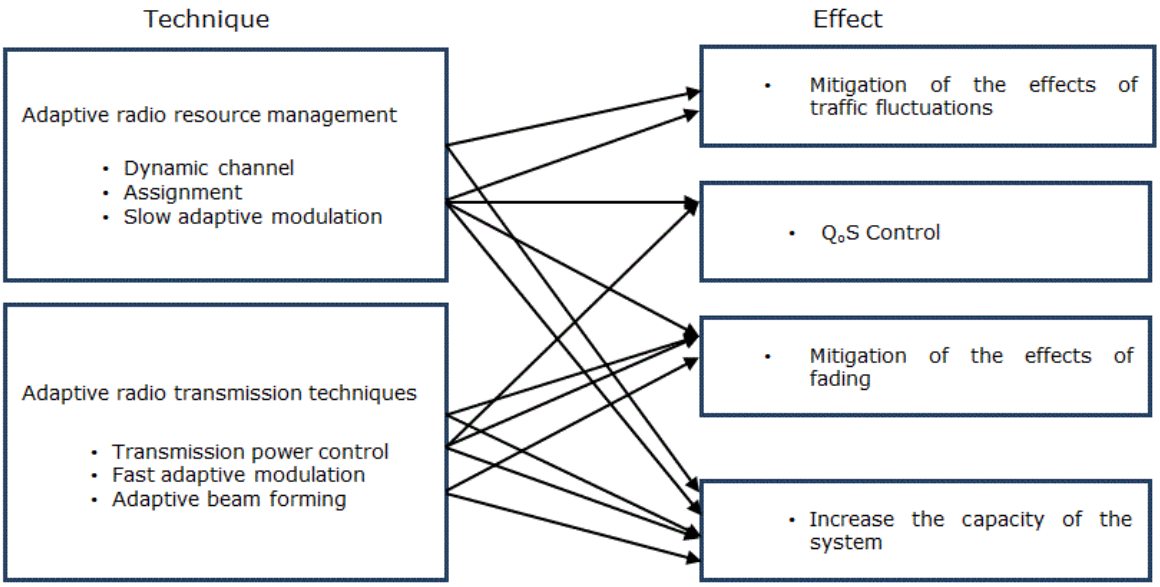


Figure 2.1 Over of Adaptive Modulation System (Sasaoka, 2000)

The determination of the parameters for the channel of transmission is dependent on the rate at which the channel varies in its transmission. The estimation error is also affected in this way. There are times when both slow and fast fading components appear in the channel system, and the adaptive transmission might get adapted to both the stages. Generally the techniques of adaptive modulation are based on the variations in any of these factors – data rate, power, coding, probability of error, and combining these techniques (Goldsmith, 2005). In general the process involves general of a call request from a terminal point, which is responded by the slow adaptive modulation system considering the conditions of the traffic. On the other hand, the role of the fast adaptive modulation system is such that it monitors the direct conditions of the channel of propagation that takes place in between the terminal and the base point (Sasaoka, 2000).

As Blogh and Hanzo (2002) discussed in their studies, a reliable method of rate adaptive modulation for optical fibre transmission systems was given by experiments conducted by Otsuki et al. According to their finding, the parameters, modulation level, could be embedded in the mid-amble of the frames of transmission with the use of Walsh codes. With this, maximum likelihood detection could be possible and the Walsh sequences could be decoded. The modulation mode that was required could also be estimated with use of another technique that was suggested by Hanzo and Torrance, where symbols of unequal protection of errors could be represented by modulation control symbols (Blogh and Hanzo, 2002).

Gho, Klak and Kahn (2011) discussed that, with the use of the FEC scheme, serially concatenated RS codes can be employed based on decoding of hard decision and varying the rate of the codes by shortening and puncturing them. Variations of the rate can be further achieved through a method of inner repetition associated with the soft combination of the codes. Different coding schemes have different performances and performance gaps based on which the modulation of the transmission systems can be used.

2.2.3. Pulse Position Modulation

Pulse position modulation represents a modulation of the position of the pulse relative to its unmodulated position. There are several advantages of digital pulse modulation as mentioned by Ko (2011). The major advantages include its performance, ruggedness, reliability, security, efficiency, and integration of the system. The problems of noise degradation of the channels and the distortions of signals are reduced with this demodulation process (Ko, 2011).

In PMM, *m* bits of binary data which can be represented by means of a solitary pulse. There is a trade-off between the enhanced bandwidth consumption and the final line rate. The identical amount of data must be transported in the same time frame. Consequently, in the consideration of a PCM bit interval where T_b is the bit time, the frame time is represented by mT_b, in which there are 2^m PPM time slots. This relationship infers that the PPM rate is $2^m / m$ faster than the PCM in order to sustain the identical traffic flow of data. (Shalaby, 1999; Sibley & Massarella, 1993; Sibely, 1994; Sibley, 2003; Sibley, 2004; Zwillinger, 1988).

With use of digital communication system, the tolerance of effects of noises from the communication channels and signal distortions increases. It is a highly reliable system as it can exploit strong and powerful coding schemes for control of errors. Encryption algorithms can be effectively used in order to make the system more secure. Moreover, digital communication system is more efficient than analog communication system (Ko, 2011). As Liu (2002) explained, in pulse position modulation the bandwidth is exchanged for the signal to noise ratio.

The problems that are mostly faced in communications are in relation to communicating the message that has been sent from one point to another. All messages that are communicated are different and hence have different meanings that need to be effectively communicated in order to keep the meanings intact. For this purpose an effective system of communication is essential and engages a source of information, the

transmitter, the signals and the receiver. Based on the elements and the processes of communications, the systems can be discrete, continuous, or mixed (Liu, 2002).

Pulse position modulation has been presented by Liu through the following diagram figure 2.2 (2002). In the words of Liu (2002), Pulse position modulation is a "method of encoding information in a signal by varying the position of pulses. The unmodulated signal consists of a continuous train of pulses of constant frequency, duration, and amplitude. During modulation the pulse positions are changed to reflect the information being encoded".

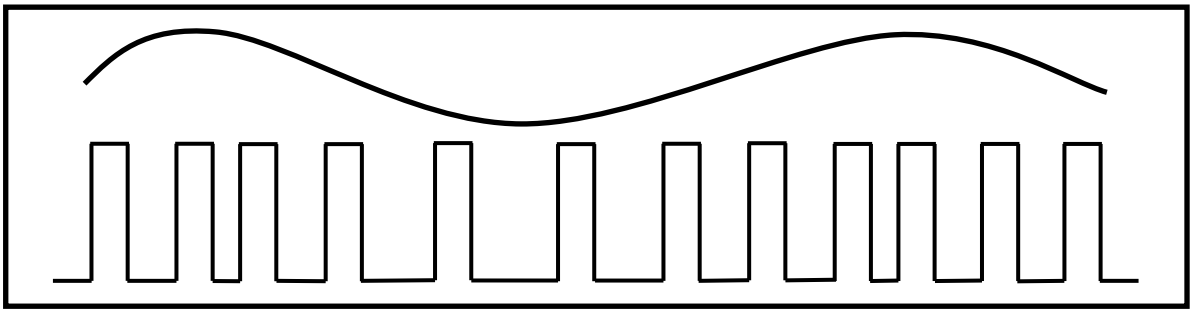


Figure 2.2 Pulse Position Modulation (Liu, 2002)

An interesting feature of pulse position modulation is that it can provide a low average power. In case of using digital pulse position modulation, the appropriate channeling and use of line coding is essential. There are different techniques of coding that are available for this purpose. These include the convolutional codes, the turbo codes, and the codes of Reed Solomon. Pulse position modulation proves to be an effective replacement to on-off keying (OOK). However, while there are advantages of the pulse position modulation, there are certain disadvantages as well that needs to be realized while using this technique of communication system, as explained by Xu, Khalighi and Bourennane (2009).

2.3. DiPPM Coding Scheme for Optical Fibre Communications

As Sibley (2003) discussed, there are various schemes of pulse position modulation that have been developed for the purpose of using with communication links based on optical fibre. Significantly better sensitivity than pulse code modulation is obtained with the use of pulse position modulation.

An original coding scheme explained by Sibley consisted of combining dicode and digital PPM to form dicode PPM. As the digital PPM is considered, it consists of 4 bits of PCM being converted into digital PPM. The original word of the PCM is in charge of controlling the position of the pulse. The level of coding and the index of modulation are the two

key variables in signalling of the PPM. The final rate of data, which is in most cases very high, has been given by the following equation (Sibley, 2003):

$$f_{DPPM} = \left(2^m / xm\right) B \tag{2.2}$$

where, m represents the level of coding, B represents the bit rate of the PCM, and *x* represents the index of modulation of the transmission. The index modulation, *x*, is again given by the following equation (Sibley, 2003):

$$x = 2^m / \left(2^m + g\right) \tag{2.3}$$

Here, *g* represents the number of guard slots present in the frame (Sibley, 2003).

There has been significant interest among the researchers to study and analyze data transmission based on PPM method and optical fibres, allowing transmission of data to be more effective. As line codes play a significant role in this regard, the selection of the novel line codes for the purpose of transmission has been found to be dependent on specific features of the communication channel as need to be passed or opposed (Osterberg, 2003). The characteristics of bandwidth for the different types of line codes that eventually present the novel coding for the optical fibre transmission system have been presented by researchers as given in the following table 2.1 (Osterberg, 2003):

Table 2.1 Bandwidth characteristics for different types of line codes (Osterberg, 2003)

Line Codes	Transmission Bandwidth	Transmission Efficiency
RZ	±2B	¼ bit/s/Hz
NRZ	±B	½ bit/s/Hz
Duobinary	±(1/2)B	1 bit/s/Hz
Single sideband	±(1/2)B	1 bit/s/Hz
M-ary ASK (M=2N)	±B/N	Log_2N

Alis and Faiman (2004) also presented a novel coding, which is dependent on data and makes use of the phase encoding scheme for the purpose of optical fibre communications and transmission systems. This particular scheme focused on the redistribution of energy among 1-bits that enabled the reduction of the optical power being leaked into the 0-bit time slots. This has been found to be useful and can be applied for a wide range of input powers in cases of transmissions of signals involving return-to-zero-features. The coding scheme being dependent on data and being introduced in the system has been obtained to increase the Q factor (Qualitity factor) by approximately 4 dB. This has been supported by the tremendous efforts of the part of

the researchers over the years that has eventually led to develop of the novel coding for the optical fibre transmission systems (Alic and Faiman, 2004).

2.3.1. Maximum Likelihood Sequence Detector

In the early years, dicode pulse position modulation was developed as a means to develop greater advantages over the schemes of the standard digital pulse position modulation. In the development of the dicode pulse position modulation, the original method of slope detection could be used for its investigation of performance. However, the slope detector can be replaced by central decision detection (CDD) (Charitopoulos, Sibley, & Mather, 2011).

Considering PPM, it is well known that there are three different types of errors of detection: wrong-slots; erasure; and false-alarm. The occurrence of wrong slots arises in cases where a pulse appears in the previous slot within the same frame or in the slot in the same frame in the following section due to the presence of noise. Errors of erasure occur when noise causes a loss of the pulse. False alarm errors occur when noise causes a threshold crossing when there is no pulse (Charitopoulos, Sibley, & Mather, 2011).

The advantages of dicode pulse position modulation have been obtained over the standard pulse position modulation for communications through optical channels. The construction of MLSD in DiPPM can also be obtained in VHDL, which shall be discussed in the further sections of the literature review. Researchers focused on construction of the MLSD in DiPPM including all the components of the pulse position modulation, such as coder, timing extraction, and decoder, and VHDL could be used for the development of the structure followed by its implementation on Altera FPGA (Charitopoulos, Sibley, & Mather, 2011).

For the MLSD to work for the detection of errors in the system of communication, when constructed in the DiPPM, it is essential that the information that is received through the channel of communication is stored. This storage of the information enables the system to determine whether there is an error in the communication and accordingly it can then be removed from the system. As researchers have conducted experiments of DiPPM through MLSD based on optical channels of communication, the key components focused on for the studies included DiPPM coder, the transmitter/receiver of the information passed through the channel, timing extraction, DiPPM MLSD decoder, and DiPPM decoder (Charitopoulos, Sibley, & Mather, 2011). Figure 2.3 shows the image of the MLSD output waveform identical with the optical decoder input, in which case no error appears.

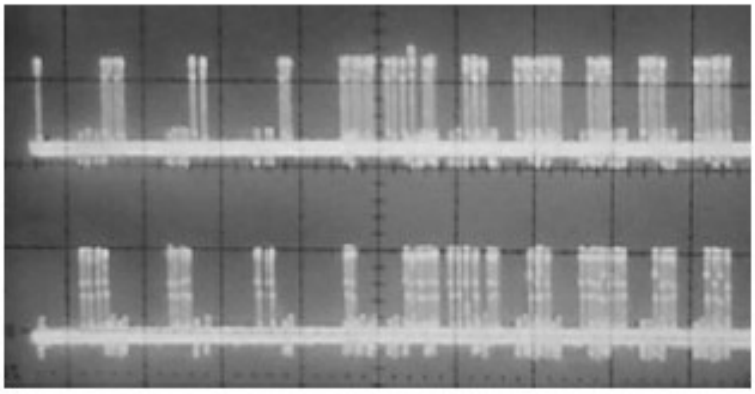


Figure 2.3 Above image representing decoder input, below image representing MLSD results (Charitopoulos, Sibley, & Mather, 2011)

Thus, it obtained by the researchers (Charitopoulos, Sibley, & Mather, 2011) that MLSD could be used for error detection in DiPPM, the flowchart for which has also been provided explaining the process in detail. While the errors in the optical communication can be detected with implementation of the MLSD, the correcting measures can also be applied for the correction of the errors that are detected (Charitopoulos, Sibley, & Mather, 2011). The flowchart of the process, figure 2.4, has been given by the researchers who experimented with the error detection and correction measures using MLSD. With the help of this flowchart the MLSD can be implemented in DiPPM and used for error detection and correction in order to obtain effective communications through optical channels.

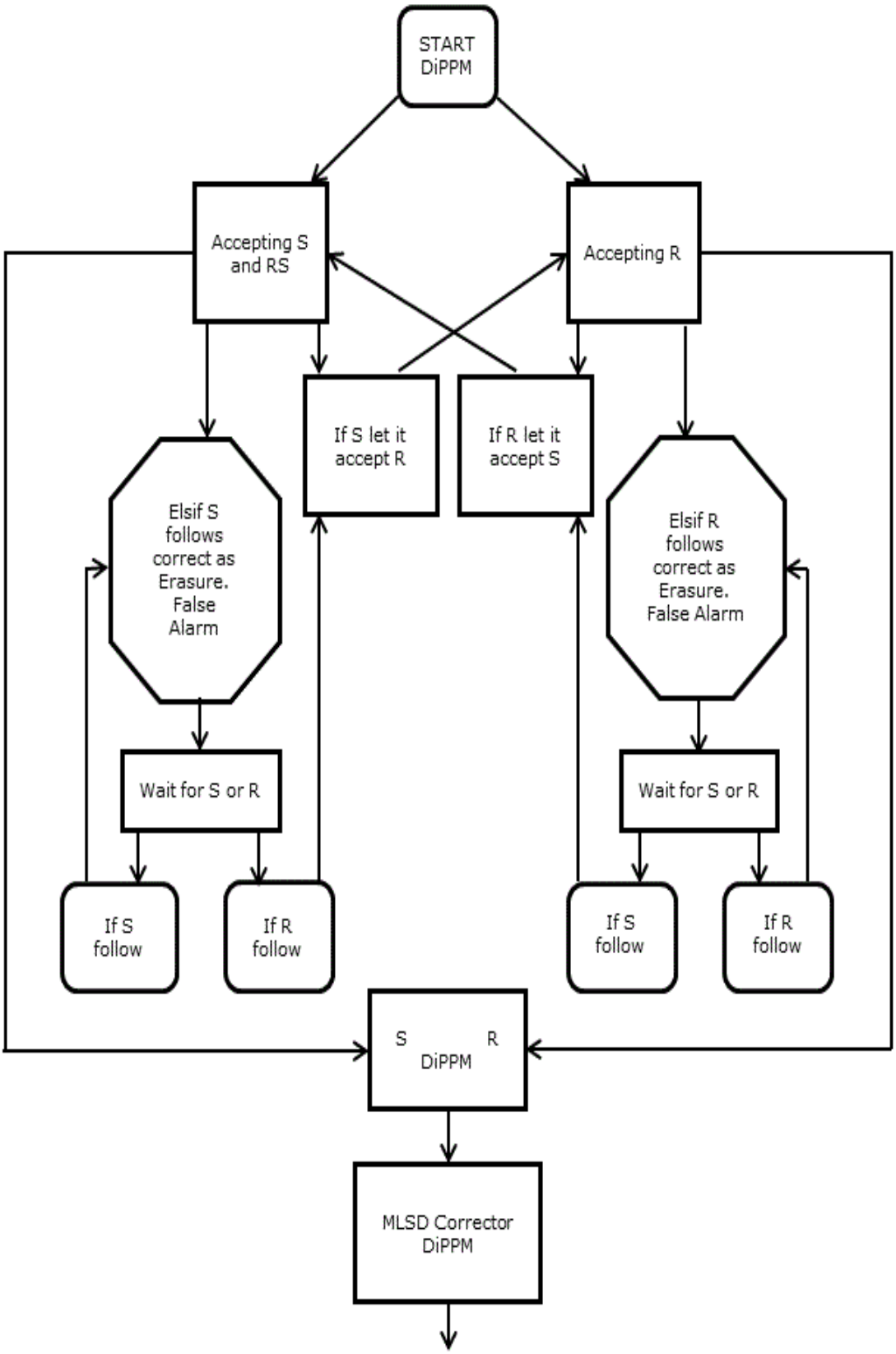


Figure 2.4 MLSD Flowchart with MLSD Corrector (Charitopoulos, Sibley, & Mather, 2010)

2.3.2. Implementation

As presented in the flowchart, the system of the incorporating the MLSD in the DiPPM occurs through enabling the detector to study the signals obtained when the message is being passed. As Zhu and Kahn (2003) discussed in their studies, free space turbulence is a factor that largely affects the effective working of the system. Thus, while the process is being implemented, researchers have to consider measures to overcome such impacts in order to mitigate the effects of the external factors. With the use of the MLSD, the temporal correlation of fading which is induced by turbulence can be exploited, which, when implemented is expected to go one better than the symbol to symbol ML detector (Zhu, & Kahn, 2003).

Considering the transmission of n number of bits, the implementation of the MLSD in the DiPPM can be understood in the way the likelihood ratio of all the sequences are computed by the MLSD. The possible bit sequences can be represented as $s = [s_1\ s_2 \ldots. s_n]$ for the 2^n possible sequences of the transmitted bits. The signal sequence that is received may be represented by $r = [r_1\ r_2 \ldots . r_n]$. Considering this transmission and receiving of the signals for optical channel communications, the complexity of the MLSD can be obtained to be proportional to $n.2^n$. This is so in this case it is necessary to compute n-dimensional essential for all the sequences of the signal being transmitted (Zhu, & Kahn, 2003).

In order to reduce the problems of complexities associated with the implementation of the MLSD, MLSD algorithm is derived and implemented in the process. MLSD solutions have been obtained from multiple input multiple output (MIMO) optical wireless (OW) systems. It has been obtained by researchers that MLSD can be implemented with MIMO proving to be highly effective as MIMO encounters significant complexities owing to which it is not capable of being used for practical reasons. The most effective algorithm that has been proposed for MLSD effectiveness is the EM (Expectation-Maximization) algorithm (Chatzidiamantis et al, 2009).

The EM algorithm is a widely used algorithm used for the implementation of the MLSD in DiPPM for ensuring that the complexities are reduced and the detection of errors and their corrections can occur effectively. The algorithm is particularly of use because in most cases there is certain necessary information related to the communications that are missing and hence make it difficult for the estimation of the state of the channel of communication. Thus the EM algorithm has been obtained to a low complexity solution for the implementation of MLSD (Chatzidiamantis et al, 2009).

Associated with this, researchers have also investigated the performance of the symbol by symbol ML detector, PSAM, and MLSD with respect to the MIMO optical wireless systems. Error detection techniques that are based on the use of MLSD can be engaged when direct channel state information (CSI) is not accessible at the point of the receiver of the communication message. In this case, an assumption is made that states that the receiver end has an understanding of the marginal joint allotment of the variations of concentrations, but not of their immediate condition (Chatzidiamantis et al, 2009).

The implementation of the MLSD in DiPPM can be well understood through the following table 2.2 that represent the detection of the PPM sequence by the MLSD in which a *R* pulse has been erased. From this table, as Sibley obtained, it could be determined that for any pulse for it to get corrected, the optimum location is somewhere in between the two symbols irrespective of where the source of error might exist (Sibley, 2005).

Table 2.2 MLSD detection of a dicode PPM sequence in which an R pulse has been erased (Sibley, 2005).

Invalid sequence	S	N	N	N	N	N	N	N	S	Binary representation								
	S	R	N	N	N	N	N	N	S	1	0	0	0	0	0	0	0	1
	S	N	R	N	N	N	N	N	S	1	1	0	0	0	0	0	0	1
	S	N	N	R	N	N	N	N	S	1	1	1	0	0	0	0	0	1
	S	N	N	N	R	N	N	N	S	1	1	1	1	0	0	0	0	1
	S	N	N	N	N	R	N	N	S	1	1	1	1	1	0	0	0	1
	S	N	N	N	N	N	R	N	S	1	1	1	1	1	1	0	0	1
	S	N	N	N	N	N	N	R	S	1	1	1	1	1	1	1	0	1
Average										7/7	6/7	5/7	4/7	3/7	2/7	1/7	0/7	7/7
MLSD output	S	N	N	N	R	N	N	N	S	1	1	1	1	0	0	0	0	1
Original word	S	N	R	N	N	N	N	N	S	1	1	0	0	0	0	0	0	1
Error bits (2 off)												1	1					

2.3.3. Performance Analysis

There are three major types of errors related to pulse position modulation systems: wrong-slot; false alarm; and erasure. These errors are needed to be detected before they can damage the communication message that is being transmitted through the signals. Sibley (2005) presented an MLSD algorithm for this purpose that makes use of sequences of the natural pulses in dicode pulse position modulation and intends to

completely remove the detected errors or minimize their effects on the communication system as much as possible.

It has been obtained that the application of the MLSD algorithm can be done in optical channels that are non-directed, indoor, free space. This is primarily so because the errors of the signal sequence can be obtained from ISI or intersymbol interference, which is initiated by these optical channels of communication. When the MLSD scheme is used for error detection, the transmission of data makes use of only certain words or sequences of the signals. The word that is detected is matched with all available words and sequences to determine if it is an error or not. This function takes place at the receiver. In the case where the word that is received is corrupted by ISI or the noise existing in the channel of communication, then the MLSD decoder determines the word which most likely matches with the sequences being checked, and makes use of that word for the generation of the data that has been decoded (Sibley, 2005).

The performance of the MLSD in communication error detection in optical channels is determined by the way the algorithm can effectively detect the errors in the system. The probability of the errors can be determined as follows, thereby helping in the analysis of the performance of the MLSD in DiPPM (Sibley, 2005):

If the signal sequence of a general dicode PPM is considered, say SxNRyNS, the average PCM probability of error that can be calculated occurring owing to an event of error would be:

$$P_{e_{x,y}} = \sum_{y}^{n-1}\left(\sum_{x}^{n-1}\left(\left(\frac{1}{2}\right)^{x+2}\left(\frac{1}{2}\right)^{y+2} P_e Error_{x,y}\right)+\left(\frac{1}{2}\right)^{n+1}\left(\frac{1}{2}\right)^{y+2} P_e Error_{n,y}\right)+ \sum_{x}^{n-1}\left(\left(\frac{1}{2}\right)^{x+2}\left(\frac{1}{2}\right)^{n+1} P_e Error_{x,n}\right)+\left(\frac{1}{2}\right)^{n+1}\left(\frac{1}{2}\right)^{n+1} P_e Error_{n,n} \quad (2.4)$$

Where P_e represents the probability of error of a certain detection of the pulse (erasure or false-alarm) and $Error_{x,y}$ represents the number of PCM errors ensuing from the error of the detection of the pulse (Sibley, 2005).

The analysis of the performance of the MLSD can also be achieved using a simulation model. According to this model, an optical receiver is used having a limited bandwidth, with its output having a white noise spectrum. A classic matched filter could also be engaged for the purpose of predetection has been included in the system. Sibley also considered the graded index POF for the transmission of the dicode PPM. The block diagram of the receiver system has been given by researcher as shown in figure 2.5, and

is applicable for the performance analysis of the MLSD in DiPPM (Sibley, 2005). Table 2.3 provides a comparison of error probabilities at specific normalised link bandwidths for dicode PPM operating with and without MLSD.

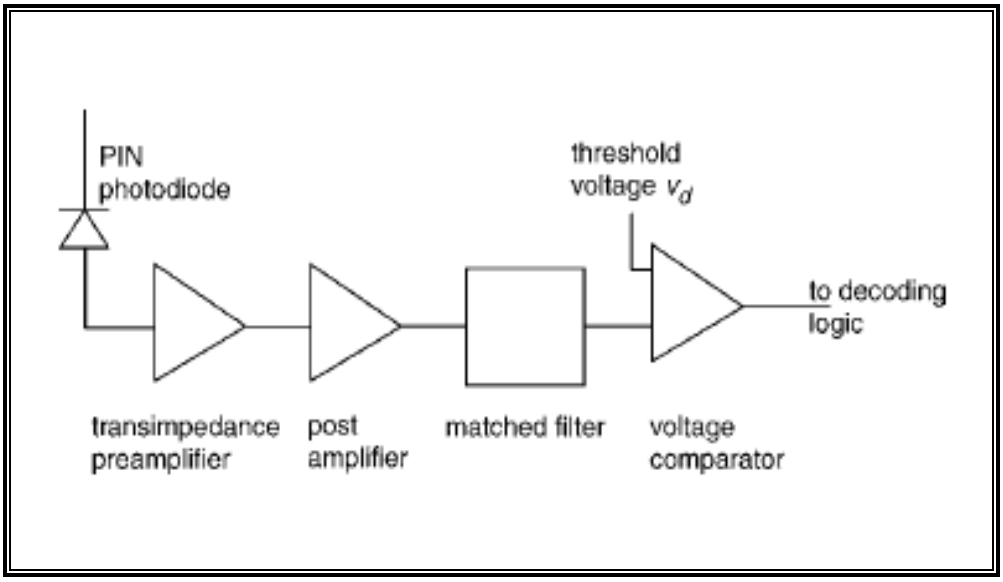


Figure 2.5 Block diagram of receiver system (Sibley, 2005)

Table 2.3 Comparison of error probabilities at specific normalised link bandwidths for dicode PPM operating with and without MLSD (Sibley, 2005).

Normalized link bandwidth		100	10	1	0.46	0.29
Photons per pulse (×10^3)	MLSD Non-MLSD	2.11 2.14	4.59 4.66	14.27 95.75	40.22 658.38	408.16 -
Threshold parameter, v	MLSD Non-MLSD	0.54 0.54	0.53 0.52	0.50 0.93	0.60 0.98	0.97 -
Error probabilities ×10^-10						
Eraser R-N Eraser S-N	MLSD Non-MLSD MLSD Non-MLSD	1.85 2.68 3.57 2.68	1.73 2.65 3.34 2.65	1.97 4.42 3.16 2.95	1.15 0.22 5.35 0.17	0.37 - 6.21 -
False Alarm N-R False Alarm N-S	MLSD Non-MLSD MLSD Non-MLSD	2.45 1.86 2.13 2.78	2.64 1.88 2.28 2.82	2.82 2.64 2.07 0	1.70 0.94 1.81 0.03	1.66 - 1.77 -
Wrong-slot R-S	MLSD Non-MLSD	0 0	0 0	0 0	0 8.64	0 -

The DiPPM employing MLSD model needs $4x10^4$ photons in each of the pulses. This aspect can be contrasted to $66x10^4$ photons for the DiPPM in the model which does not possess MLSD. This infers an enhancement in sensitivity of *12.2* decibels.Thus the maximum likelihood sequence detection can be used in dicode pulse position modulation based on the signal sequences occurring naturally, thereby allowing detection and correction of system errors that occur during the transmission of the signals of communication through the optical channels (Sibley, 2005).

The ARQ algorithm (automated repeat request) has also been recommended in order to diminish the sources of error. Wang et al (2007) had performed this experiment and obtained that the PER, packet error rate, performance of DiPPM can be improved with the use of ARQ, automatic repeat request, during the time of the detection of the error. In this experiment, the structures of modulation of the DiPPM and the requirement of the optical power have been presented with the PER. It has been shown that the PER for DiPPM is lower than the cases of OOK and 4-PPM, figure 2.6. The PER performance for the DiPPM could be enhanced by 19 dB, at average received signal power -54dbm, by application of the ARQ compare with OOK. Also, DiPPM reflected higher efficiency of power in comparison with other PPM. However, in comparison with higher order PPM, the power efficiency is lower for DiPPM with the higher packet error rate, but with lower requirement of the bandwidths (Wang et al, 2007).

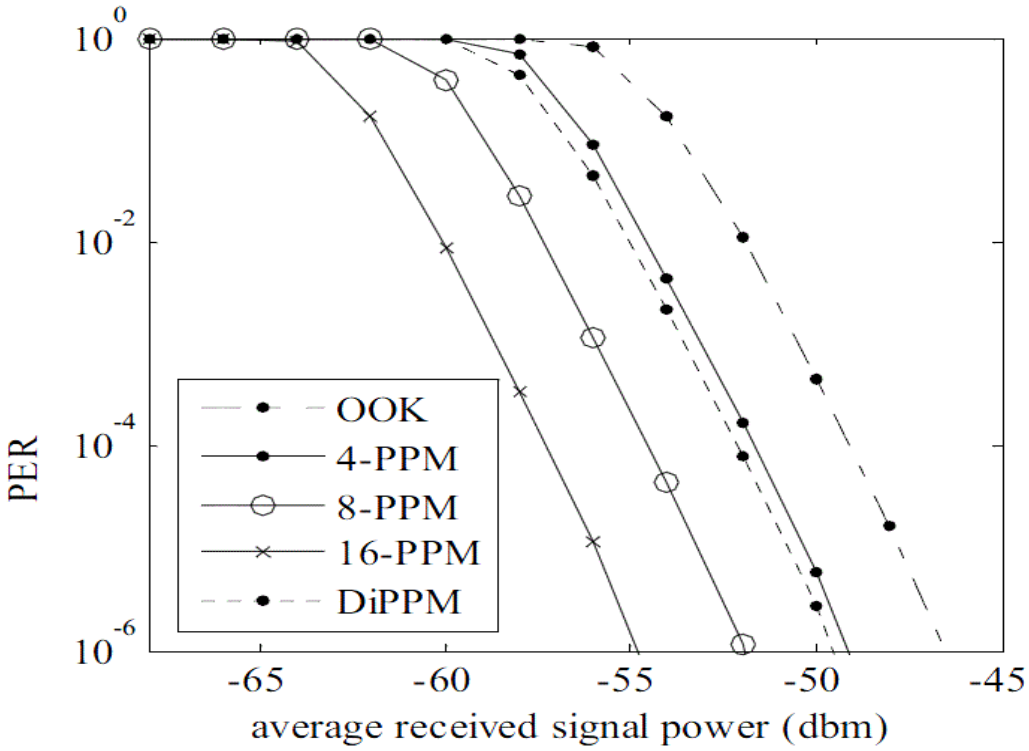


Figure 2.6 Packet error rate comparisons of OOK, 4-PPM, 8-PPM, 16-PPM and DiPPM (Wang et al, 2007)

2.3.4. Suboptimal Filtering in Zero Guard DiPPM

Optimal filters that are used for the DiPPM are comprised of a noise whitening matched filter along with proportional-derivative-delay (PDD) network. It has been found that the PDD network can be removed from the system and such removal would only cause a slight loss of sensitivity of the system. However, the process of implementation of the rest of the structure consisting of the matched filter without the PDD network has been found to be highly complex in nature. In order to reduce such complexities, one of the alternative methods that has been obtained as effective is the use of suboptimal filtering along with third order Butterworth filter in a zero-guard interval dicode PPM system that has the capability to operate over a dispersive optical channel (Sibley, 2004).

In early researches, the use of a classical matched filter was considered for the purpose of predetection filter. Such filters were advantageous in their own ways but implementation was difficult due to certain factors. Firstly they required a noise whitening filter before the signal could be matched with the filter. The main problem in this was that the noise whitening filter could not be realized owing to its varying frequency and characteristics associated with the noise of the preamplifier. Another problem associated with this was that the filter section that was to be matched was matched to the shape of the pulse that was received, however this was dependent on the optical channel. Hence, it necessitated construction of filters for every link individually (Sibley, 2004).

The suboptimal filtering in zero guard makes use of a simulation system based on a receive system, the process of which can be realized from the following diagram:

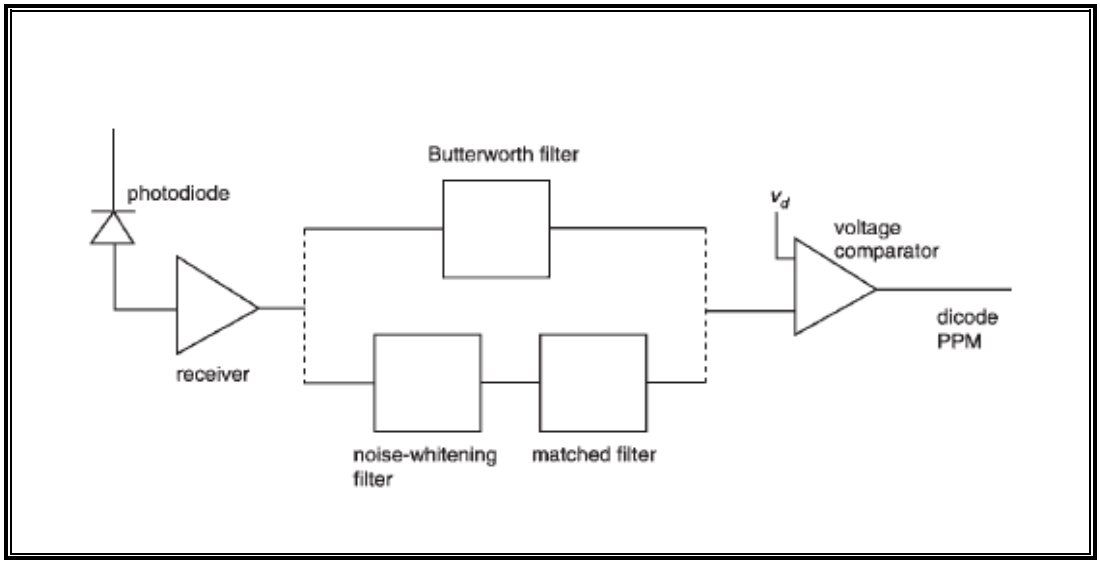


Figure 2.7 Block diagram of dicode PPM receiver used in simulations (Sibley, 2004)

The Butterworth filter has the frequency response:

$$H_f(w) = \frac{1}{(jw)^3 + 2(jw)^2 w_B + 2jww_B^2 + w_B^3} \quad (2.5)$$

Where ω_B is the *-3dB* bandwidth of the filter and is given by

$$w_B = \frac{\sqrt{2ln2}}{\alpha k_B} \quad (2.6)$$

Here k_B represents the bandwidth factor (Sibley, 2004).

With the use of suboptimal filtering in zero guard, based on Butterworth filter, researchers obtained the inter symbol interference effects of the system. It could be obtained that less photons are required by the Butterworth filter per pulse in comparison to other filters involving noise whitening attributes. Also, the Butterworth filter is capable of functioning in cases where the channel bandwidths are lower. The bandwidths of the filter and the receiver system are not dependent on the bandwidth of the channel, thereby enabling a simple design of the link to be obtained (Sibley, 2004).

2.4. PPM Employing Reed Solomon Codes

The performance of optical communication link can be improved by adding an error correction code. Considering this factor, researchers have found that Reed Solomon block codes are of significant benefits as against the frames of the PPM for the purpose of detection of error to the highest extent possible. Reed Solomon codes are capable of correcting the errors of the symbols as well as erasures that occur over the symbols. In this case, the focus is considered on demodulation of the laser fields of the PPM, which in turn leads to generation of input symbols for the decoder of the Reed Solomon codes (Divsalar et al, 1982).

It depends largely on the method chosen for the process of demodulation that allows conversion of the received laser fields into digital formats that the probability of errors is able to be detected either occurring in the form of erasures or symbol errors. This in turn leads to defining the symbols of the communication channels being transmitted. There are several demodulating schemes that are available. The effects of the different schemes on the performance of the Reed Solomon coding and decoding. Computations have been performed by researchers for several models of the optical receiver to obtain the different possible results (Divsalar et al, 1982).

The determination of the probabilities of error in the communication channels is also associated with the length of the Reed Solomon codes as used in the system. It has been shown that the simple threshold detection of the pulses is capable of degrading performance, which can degrade even faster with the rate of the increase of noise in the system. This happens mainly because of the fact that for the Reed Solomon decoder to be used, too many erasures are generated in the process. Researchers have thus suggested a decision scheme, delta-max demodulation, which can overcome the existing problems and improve upon the threshold detection of the system with the generation of the erasures being newly defined (Divsalar et al, 1982).

In cases where no Reed Solomon codes are used, the MLSD of the DiPPM is in need for greatest count assortment for each individual frame of the PPM with the choice of the codes and frames being random in nature (Divsalar et al, 1982). In order to achieve near-capacity performance of the optical communication channels, it is essential, as obtained by early researchers to ensure proper modulation and coding of the optical signal. Optical PPM is chosen as it is considered as an efficient method for the purpose of modulation (Hemmati, 2006).

With the selection of the modulation format, it is also essential to select the suitable channel coding. Researchers are continuously focused on developing upon the steps and measures of modulation such that the channel coding for the optical channels can be developed for effective communication purposes. The use of Reed Solomon Codes has been considered as efficient by the early researchers considering their ability to naturally map to the 2^m – ary alphabet representing the symbol of the PPM (Hemmati, 2006).

2.4.1. Reed Solomon Codes

The development of the Reed Solomon Codes can be dated back to 1960 when Irving Reed and Gus Solomon researched and reported their findings describing a set of codes that could correct errors in a new way and were named as the Reed Solomon codes. These codes have significant usage in the optical channel systems of communication for the purpose of correction errors, supported by the MLSD that performs in the detection of the errors in the system. The power and utility of these codes are extremely high. In the present times the codes are used widely in several applications for the benefits that they provide, particularly in wireless communications systems (Sklar, n.d).

Some of the basic features of the Reed Solomon Codes include (Sklar, n.d):

- These codes are nonbinary cyclic in nature and are formed of symbols involving sequences of *m*-bit, where *m* represents any positive integer whose value is greater than *2*.
- Considering the *m*-bit symbols for the Reed Solomon codes, there are associated symbols for the system such as *n* and *k* that can be represented as

$$0 < k < 2^m + 2 \tag{2.7}$$

In this case, the '*k*' represents the number of symbols of the data that are decoded in the process of data transmission with the use of Reed Solomon codes. 'n_{RS}' represents the total number of symbols of the codes that are present in the block which is embedded with the codes.

- With the use of Reed Solomon codes, it is possible to achieve the code that is the largest possible with minimum distance (d_{min}) being covered for any code that is in linear position from the encoder input and output system.
- Reed Solomon Codes are capable of correcting communications channel errors of *t* or fewer combinations, which can be represented as follows:

$$t = \left[\left(d_{min} - 1 \right) / 2 \right] = \left[\left(n - k \right) / 2 \right] \tag{2.8}$$

where in this case [*x*] represents the largest integer that should not exceed the value of *x*.

- Reed Solomon codes are particularly useful for the correction of burst error. This is primarily because these codes are highly effective for memory based channels of communication. The use of the codes is also effective in cases where the input symbols for the channels are large.
- The Reed Solomon symbol error probability $P_{E,}$ in the context of the channel symbolic error possibility can be demonstrated as the following relationship (Sklar, n.d):

$$P_E \approx \frac{1}{2^m - 1} \sum_{j=t+1}^{2^m - 1} j \binom{2^m - 1}{j} p^j \left(1 - p\right)^{2^m - 1 - j} \tag{2.9}$$

- The symbol error probability is associated with the binary error probability P_{eb} by the following formula

$$P_{eb} = \frac{2^{M-1}}{2^M - 1} P_E \tag{2.10}$$

A typical system of the Reed Solomon codes can be represented through the following diagram figure 2.8:

Figure 2.8 System of Reed Solomon Code (Riley & Richardson, 1998)

Table 2.4 presents the Reed Solomon codes as researchers obtained through experiments for symbol mapping of *2-6* PPM. Similarly, other PPM modes can also be mapped with the Reed Solomon codes, as researchers obtained them. The pattern of coding of the Reed Solomon codes constitutes a plurality of target elements that eventually form the target grid associated with the channel of communication (Lapstun, 2009).

Table 2.4 2-6PPM to Reed-Solomon symbol mapping (Lapstun, 2009)

2-6 PPM symbol value (p5 – p0)	Corresponding Reed Solomon Symbol Value (base 15)
000011	0
000101	1
000110	2
001001	3
001010	4
001100	5
010001	6
010010	7
010100	8
011000	9
100001	a
100010	b
100100	c
101000	d
110000	e

2.4.2. Reed Solomon Encoding and Decoding

Reed Solomon encoding can most effectively be expressed by the following equation that expresses the codes based on the most conventional parameters such as *n,k,t* and positive integers represented by *m* the value of which has to be greater than *2* (Sklar, n.d).

The equation of the Reed Solomon codes for encoding is (Sklar, n.d):

$$(n,k)=\left(2^m-1,2^m-1-2t\right) \tag{2.11}$$

where $n-k\ =2t$ represents the number of parity symbols, and *t* represents the symbol-error correcting capability of the Reed Solomon code.

The generating polynomial for a Reed Solomon code is represented by the following equation (Sklar, n.d):

$$g(X)=g_0+g_1X+g_2X^2+\ldots+g_{2t-1}X_{2t-1}+X^{2t} \tag{2.12}$$

As Irving Reed and Gus Solomon have obtained the codes of Reed Solomon are represented through the codes of Bose, Chaudhuri, and Hocquenghem (BCH) codes. The encoding of the Reed Solomon Codes can also be performed systematically. Since these codes are cyclic in nature, hence the systematic approach has proved to be equivalent to the procedure of the binary encoding. In this case, a message polynomial, *m(X)* can be considered to be shifting into the stages of codeward register, the stages being represented by *k*, followed by appending of a parity polynomial, *p(X)*. This is generally placed in the stages that are on the left most and are designated by *n-k* positions. Thus in order to shift the message polynomial, manipulation can be done in it through multiplication of the *m(X)* by X^{n-k}. It can then be divided by *g(X)*, which is the generator polynomial, and hence can be represented through the following equation (Sklar, n.d):

$$X^{n-k}m(X)=q(X)g(X)+p(X) \tag{2.13}$$

where $q(X)$ and $p(X)$ are quotient and remainder polynomials, respectively.

The decoding of the Reed Solomon codes can be understood from the early researchers' views and analysis as well. It is generally assumed that during the transmission of the communication signal, the codewords involved are corrupted as a result of errors in the system.

The process of systematic encoding of the Reed Solomon codes based on *(n-k)* shifter of stage and register has been explained by researchers through the following diagram figure 2.9.

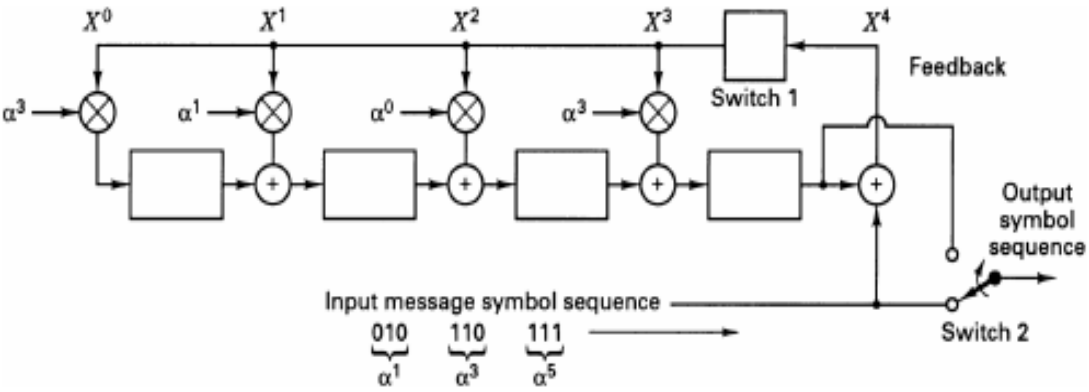


Figure 2.9 Systematic Encoding with an (n - k)–Stage Shift Register (Sklar, n.d)

Pavert (2011) discussed that with addition of extra bits to the communication data, the capacity of binary channel gets enhanced. As a result the quality of the digital data gets improved. Channel encoding represents this addition of the extra data bits while there could be cases where the errors are not randomly distributed on the sites. Reed Solomon codes are also available as the Cross-Interleaved Reed Solomon Code (CIRC) where the errors that occur during the initiation of the system, are spread over larger frames enabling enhanced process of decoding.

The process of decoding of the codes depends on linear equation systems being solved simultaneously as the data transmission takes place. One algorithm that can be used for this purpose is the Berlekamp-Massey algorithm. This algorithm enables solutions to linear equations therefore allowing decoding to occur of the Reed Solomon codes (Pavert, 2011). Block diagram figure 2.10 of the CIRC has been given by researchers as follows (Pavert, 2011):

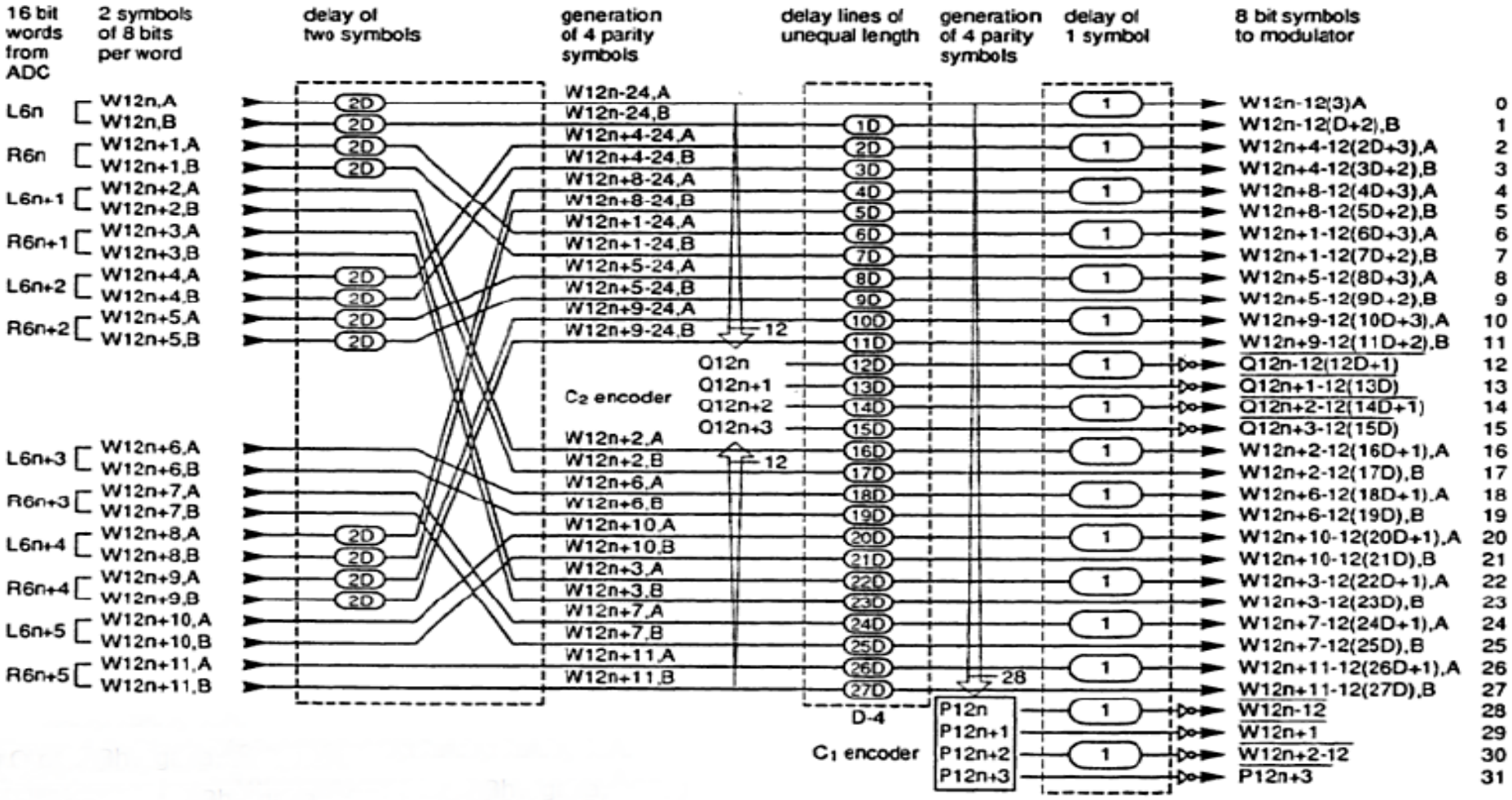


Figure 2.10 Block diagram of CIRC (Pavert, 2011)

2.4.3. Reed Solomon Codes Applications

Reed Solomon codes are the codes that are mainly used for correcting errors, and are based on blocks. This section studies and presents the applications of Reed Solomon codes, which are mainly found in digital communications and storage of data (Riley & Richardson, 1998).

The applications of Reed Solomon codes are mainly obtained in data storage and digital communications. The following major applications can be listed for Reed Solomon codes:-

Data Storage – In case of both CDs, and DVDs, it is possible to apply codes for correction of errors and measure the raw errors before correcting them. The application is in holographic data storage or optical storage and the two major schemes that are applicable include CIRC for CDs and a Reed-Solomon Product Code (RS-PC) for DVDs. Each bit of data is considered for a firm decision by both these codes to understand if the bit is 1 or 0. Following this, the correction scheme of the codes can fix the errors in the data storage devices (Curtis et al, 2010).

The application is most effective in cases where error occurs in bursts. Reed Solomon codes are capable of correcting up to 2 byte errors per 32 byte block. Up to 4000 bits of error bursts can be corrected by CIRC as a result of the features and applications of the codes (Prosch & Daskalaki, 2011).

Data Transmission – Reed Solomon codes can be used in several applications for the purpose of transmission of data. Data can be transmitted from the receiver to the transmitter. The applications include transmission systems for mobile data, and for highly reliable military systems of communications (Wicker & Bhargava, 1999). There are specialized forms of RS codes enabling data transmission, such as Caucy-RS and Vandermonde-RS where the code performing the task is an RS(n,k) code (Kythe & Kythe, 2012).

DVB-T Transmission – Digital Video Broadcasting or DVB-T comprises of a bandwidth of 8MHz. For purpose of transmission, it needs correction of its errors, and Reed Solomon codes can be effectively applied in this case. It is used as an outer code (204,188, T=8). The error control takes place by combining the Reed Solomon codes with inner convolutional codes (Lamba et al, 2005).

Space Transmission – In several planetary exploration events set by NASA and ESA, the use of Reed Solomon Codes has already been done. With combined use of

convolutional and Reed Solomon codes, it has been found to be possible to gain coding at high levels. The Reed Solomon codes can be used to correct errors. The *Voyager* expeditions represent the most popular case of applications of the Reed Solomon codes, led to other planets such as Uranus and Neptune. These codes could be used for transmission of images from these further planets and hence communicated to earth (Wicker & Bhargava, 1999; Houghton, 2001).

2.4.4. Implementation and Performance of Reed Solomon Codes

Sankaran (2000) experimented and reported implementation of Reed Solomon decoder on TMS320C64xE DSP family. The codes of Reed Solomon have been accepted over for several applications for the purpose of error control in ADSL networks, digital cellular phones and high-definition television systems. As these codes are extremely robust in correcting errors, their popularity has also largely increased increasing their usage in data communications systems, particularly in optical channels and DiPPM. Implementations of the Reed Solomon codes lead to offering the designer of the communication system with such flexibility that is unique in nature. Such an arrangement and the high flexibility of the system allow achieving a trade-off between the bandwidth of the data. As a result, variation in the errors takes place allowing correction capability to increase for particular communication channels (Sankaran, 2000).

One of the algorithms that allow effective implementation of the Reed Solomon code decoding is the Peteren-Gorenstein-Zierler (PGZ) algorithm. With use of the TMS320C64xE DSP family, there are digital signal processes that enable exploitation of the data level along with the level of instructions presenting several units of the ALU that can work in combination in order to obtain performance of high level. There are four basic steps of the Peteren-Gorenstein-Zierler (PGZ) algorithm that need to be performed for decoding of the Reed Solomon codes. These steps are (Sankaran, 2000):

- Syndrome Computation.
- Berlekamp Massey Algorithm for solving the error locator polynomial.
- Chien Search Algorithm for solving for the roots of the error locator polynomial.
- Forney algorithm for computing the error magnitudes.

When the algorithms are considered, the focus is on maximizing the number of Galois field multiplies that can be used. Generally there are four cycles of latency in the Galois field thereby necessitating certain loops to be unrolled such that the latency of the field can be taken complete advantage of. Of all the steps the computation of the syndrome is the first major step in the process of implementation where the number of cases may

vary depending on the data bits and accordingly the syndrome is computed (Sankaran, 2000).

The performance of the decoder has also been studied by Sankaran and the results obtained have been provided for a detailed understanding for the current research. The performance results for the case of *t=8* is provided table 2.5 below as obtained from the research:

Table 2.5 Performance of the Decoder for t = 8 (204,188,8) Code (Sankaran, 2000).

Name of Module	C Code	Assembly Optimizer	Hand Optimized
Syndrome Accumulate	480 cycles	470 cycles	470 cycles
Chien Search	1110 cycles	326 cycles	318 cycles
Berlekamp-Massey	340 cycles	263 cycles	246 cycles
Forney	180 cycles	154 cycles	150 cycles
Driver Function	80 cycles	80 cycles	80 cycles
Reed Solomon Decoder	2180 cycles	1293 cycles	1268 cycles

In cases where there are no errors in the system, the computation of the syndromes would come to zero value. In that case, the decoder can be preceded for decoding the next block and this does not require the decoder to exit from the other algorithms associated in the process. For the case of *t=8*, as obtained from the experiment conducted by Sankaran, the performance can be further optimized that would enable obtaining a decoder under 1000 cycles of data bits. For this purpose it is essential to know the code sizes that have also been provided by Sankaran as follows in table 2.6 (Sankaran, 2000).

Table 2.6 Code Size for (204,188,8) Decoder (Sankaran, 2000)

Name of Module	C Code	Assembly Optimizer	Hand Optimized
Syndrome Accumulate	1084 bytes	1100 bytes	1128 bytes
Chien Search	792 bytes	920 bytes	872 bytes
Berlekamp-Massey	460 bytes	628 bytes	296 bytes
Forney	696 bytes	1036 bytes	792 bytes
Total Code Size	3032 bytes	3684 bytes	3088 bytes

Besides these, the optimizations can be further modified and used for the purpose of implementation of the decoder and measuring its performance in data communication channels using the Reed Solomon Codes.

2.5. Field Programmable Gate Array (FPGA)

The use of field programmable gate array (FPGA) is another popular measure for optical channels communications. This is focused on the implementation of optical fibre on the interface between two computers. The FPGA can be placed between the two computers and the link of the optical channel for communication. Kadric (2011) experimented with the implementation of the FPGA considering the use of two nodes comprising high level of bandwidth that could be obtained in between them. FPGA has been used in the form of a chip using which the channels of the optical fibre could be connected to the interface of the communication medium. Use of transceivers allows control over the channels. In cases where the communication environments where there is a scarcity of the CPU cycles, use of the FPGA has been found to be lower the load that otherwise prevails on the CPU (Kadric, 2011).

Some of the common tasks of the FPGA thus include correction of the error, encryption, and compression of the communication channels as and when needed. Different systems of the FPGA can be developed depending on its need and use in the system. In cases where the PCIe card is used, it enables a master control over the front end cards that can be scaled according to usage. With the help of the PCIe cards, a computer system can be linked with the FPGA board which in turn is associated with links from the fibre optic channels passing over to the front end cards. Researchers have obtained rates of stability in this system at 1.6Gbit/s. With such high rates of stability it is possible to develop and enhance the performance of the communication channels. The use of FPGA in communication links has made it applicable in wide range of applications related to communications channels (Kadric, 2011).

The system as explained in this experiment has also been given in the form of a diagram for better understanding of the association of the FPGA with communication links as follows in figure 2.11.

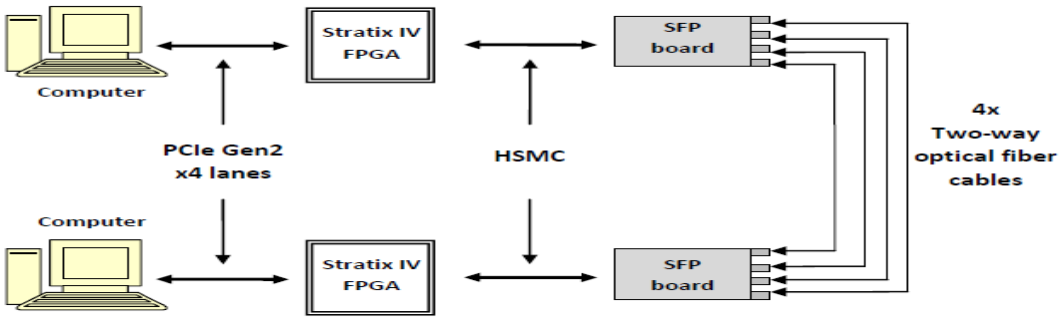


Figure 2.11 The hardware setup for linkage and performance of the FPGA with communication channels (Kadric, 2011)

Watts et al (2006) discussed the design of "a digitally programmable optical transmitter for creating advanced signal formats and predistorted signals using FPGA technology". The system that this experiment could obtain involves a tool which is flexible and can be used for experimental processes in DSP for the purpose of communications through optical channels. As the study reflects, the use of DSP or digital signal processing is mostly used in cases of the wireless systems. However the use of the same in optical channels has been found to be limited owing to the high rates of bits that it presents. Over the years, researchers have developed CMOS digital technology thereby increasing the interest of the researchers in DSP for communications through optical channels (Watts et al, 2006).

An example of this is the forward error correction that functions by detection of the bits that passes through the system. On the other hand, the other techniques that are available are in need for ADC/DAC with higher power of resolutions, which also includes the maximum likelihood sequence detection process for estimation, the reception of the diversity of the polarization, the transmission of the single sideband, and predistortion of the optical signal (Watts et al, 2006).

The FPGA has been found to be beneficial as it can be reprogrammed depending on use in the optical channels. It is in the present trend of the use of the FPGA that FPGAs are being produced by interfaces of high speeds (Watts et al, 2006). The design of the transmitter based on the use of the FPGA can be understood from the following two diagrams, figures 2.12 & 2.13, as given by researchers based upon their experiments.

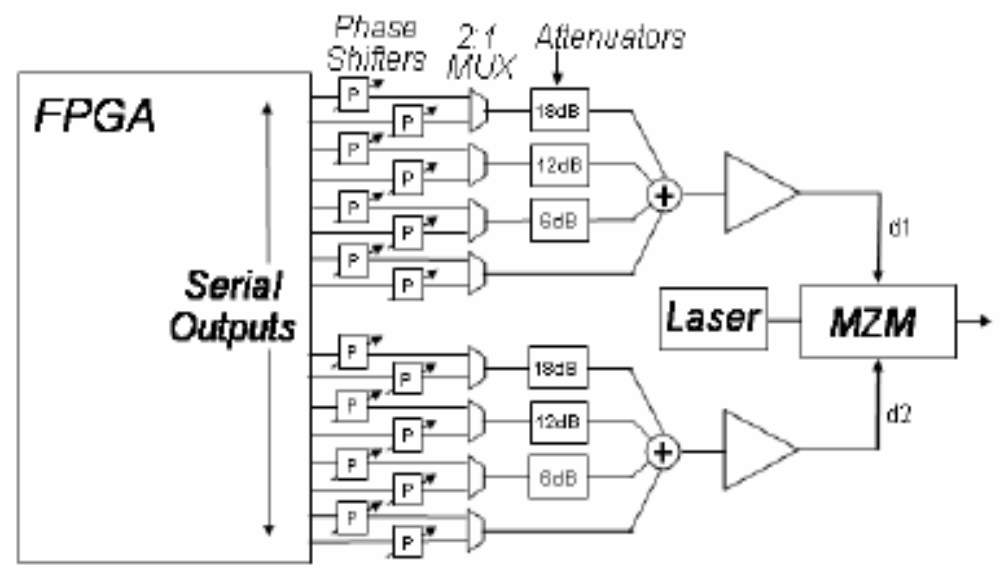


Figure 2.12 Microwave and optical components of the digitally programmable optical transmitter (Watts et al, 2006.)

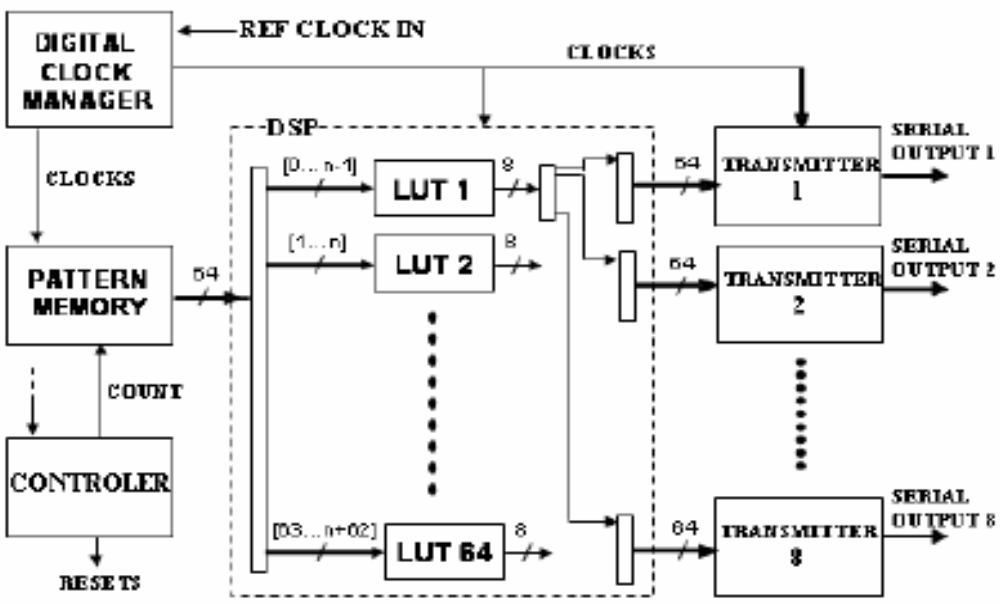


Figure 2.13 The design of the FPGA (Watts et al, 2006)

The control of the electric field amplitude in the above designed set up could be obtained by the use of the Cartesian (triple) Mach-Zehnder modulator (MZM). For each of the signals that are driven from the modulator, the serial outputs of the FPGA are multiplexed. The DeBrujn sequence that needs to be transmitted through the system can be stored as a pattern of memory in the FPGA. The processing of the signal in the system can be implemented with the use of look-up tables thereby enabling implementation of arbitrary and non linear responses (Watts et al, 2006). Simulation parameters are also available that can be used for the experiments and are given as in table 2.7.

Table 2.7 Simulation Parameters (Watts et al, 2006)

Simulation Type	Semi-analytic, Gaussian noise approx	2:1 MUX output pulses	Raised Cosine, 20 ps rise/fall time
Signal format	OOK-NRZ	Transmitted bit sequence	Sequence of DeBrujn
MZM Bandwidth	18GHz	Transmission fibre	17 ps/nm.km, linear, lossless
Performance criteria	Required OSNR (0.1nm)	Rx electrical filter	7 GHz, 4th order Bessel

2.5.1. VHDL and Applications

VHDL typically represents a hardware language that is extremely versatile in nature and is powerful and useful in electronic systems that can be modelled with the help of this language. Being widely available for use and its effectiveness for describing the electronic systems, the language has become highly popular. It enables transfer of the information of the system (Introduction to the VHDL Language, n.d.). The use of the language has also been obtained by Charitopoulos in the design of dicode pulse position modulation coder and decoder (Charitopoulos, 2009).

Experiments by Charitopoulos (2009) explained the development of both DiPPM coder and decoder in VHDL, which is also the focus of the current research. The use of the VHDL has been useful in the way it allowed programming of the timing extraction. It was done with the use of the digital, analogue and mathematical equations with the program being developed in VHDL-AMS. Also, construction of the DiPPM MLSD is possible through the use of VHDL. The results of the theoretical understanding could be obtained from the simulations, proving the experiments to be successful (Charitopoulos, 2009).

With the use of the VHDL, a complete system of the DiPPM could be developed. A source code is used for the development of the FPGA involving high speed integrated circuits. The specific functions of the devices are programmed and determined by the VHDL (Very High Speed Integrated Circuits Hardware Description Language). FPGA is associated with the programming for the development of the coder and decoder of the DiPPM to ensure that the level of noise in the external medium is lower also, with focus on reducing any internal delays that could take place from the earlier implementations of the systems (Charitopoulos, 2009). The VHDL program runs the coder simulation of the DiPPM as follows:

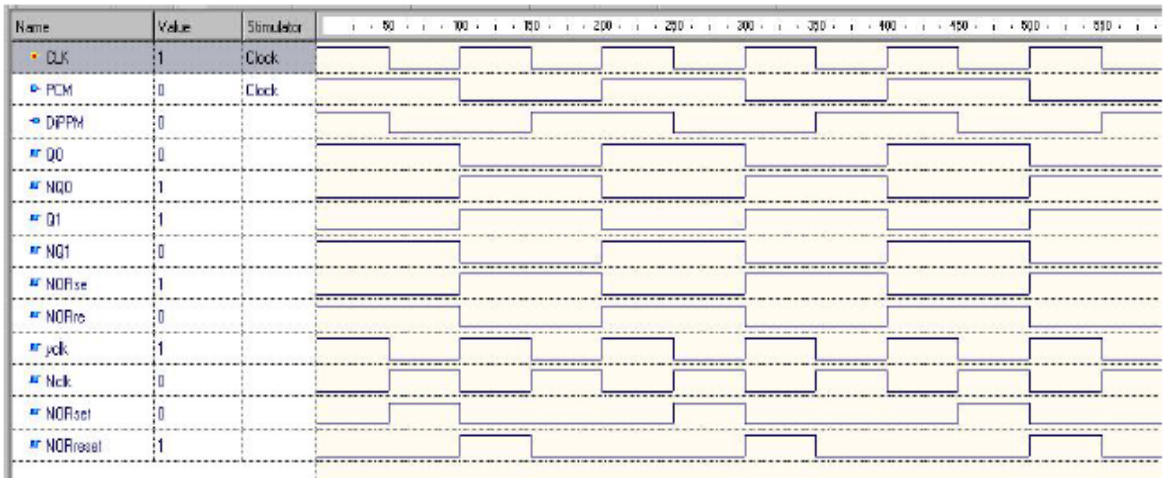


Figure 2.14 The DiPPM VHDL coder (Charitopoulos, 2009)

Figure 2.14 presents the process of the software of the DiPPM VHDL coder. The waveforms as could be obtained by the researcher match with the theoretical backgrounds. The coding of the PCM waveform has been found to be correct in relation to the format of the DiPPM. The DiPPM coder-decoder as programmed by VHDL based on the formation of the FPGA has been given by the researcher as in figure 2.15 (Charitopoulos, 2009).

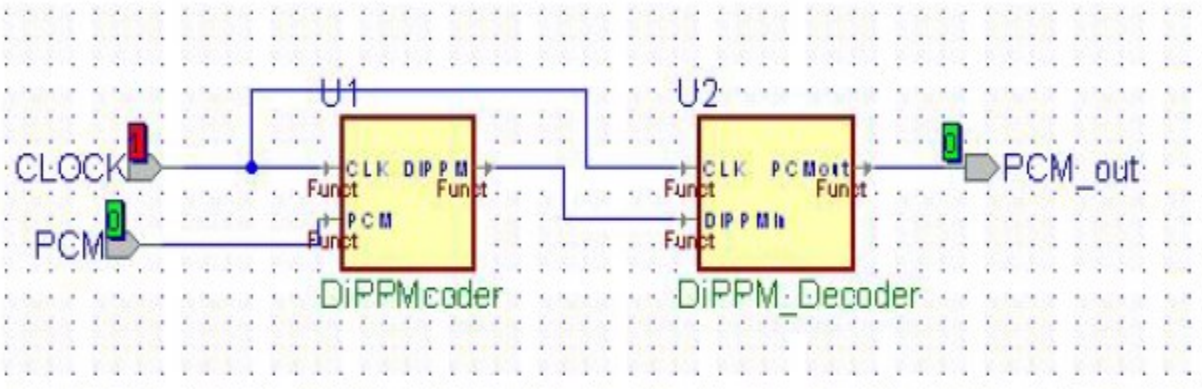


Figure 2.15 DiPPM upgraded versions of coder decoder (Charitopoulos, 2009)

For the purpose of simulation of the coder, the upgraded version of the DiPPM was needed to be used in order to generate the input signal of the DiPPM decoder. An FPGA was used for the real time measurements of the DiPPM coder and decoder. It could be obtained from researchers that for the development of the DiPPM coder and decoder in VHDL, thereby enhancing their applications in communication channels, the VHDL program could be used with use of software from the ALTERA Company. Programming can be obtained from the Quartus software as well. The DiPPM process for coder and decoder based on VHDL in the Quartus software has been given through the following representation figure 2.16 (Charitopoulos, 2009).

In the words of Charitopoulos (2009), "in line 3, *nclk* is set as invert clock (not clk) and is used in line 4 with the input DiPPM sequence. Positive pulse has to be produced when the DiPPM SET pulses appear and the *nclk* is equal to '1'. Thus, positive PCM is achieved with a half clock delay. While a RESET pulse appears the positive PCM false until a SET DiPPM pulse appears again".

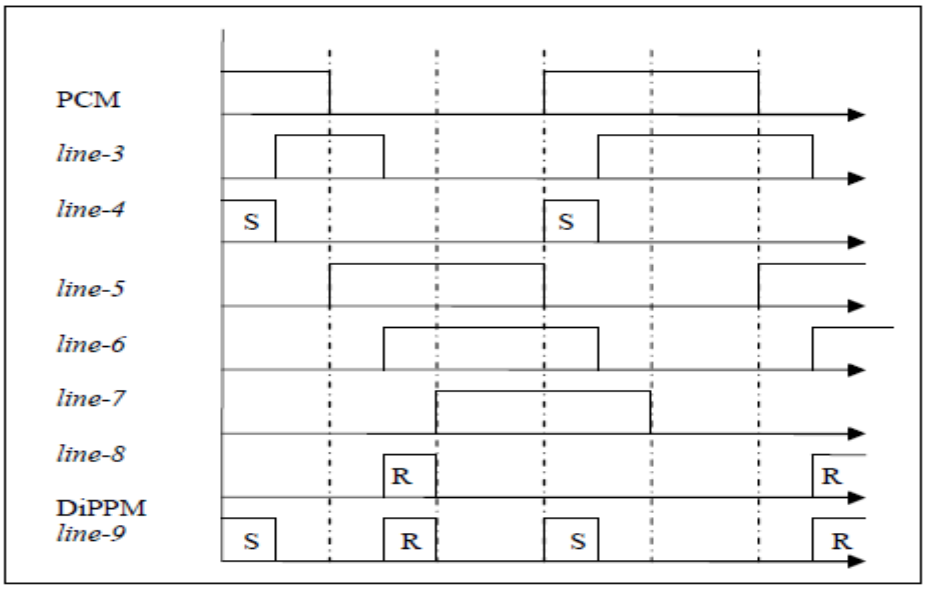


Figure 2.16 VHDL: DiPPM coder process in Quartus (Charitopoulos, 2009)

In order to complete the system of the DiPPM, the process being represented by coder-timing extraction- decoder, timing extraction was required to be simulated by the use of the software based upon the format of the DiPPM. However, there is a need for the use of the VCO in the program of the timing extraction as could be obtained from the researchers experiments. In order to achieve this, different languages and software had to be developed having commonness with the VHDL language (Charitopoulos, 2009).

Thus, for the timing extraction programming of the coder decoder development of DiPPM, the development of VHDL-AMS language has been obtained and made successful for applications in optical communication channel. With the help of the VHDL-AMS language, digital and analogue signals can be accepted. Signals can be transformed from digital to analogue and from analogue to digital as per need of the communications data. PLL circuit can be obtained with the use of the VHDL-AMS language. Following this, the timing extraction of the DiPPM can be obtained through methods of simulation (Charitopoulos, 2009).

The five major elements of the timing extraction process of DiPPM coder-decoder include the buffer, the phase detector, the loop filter, the VCO, and the digitaliser. When all these elements are used in combination, the process naturally becomes more complex. The complete timing extraction of the DiPPM as obtained has been represented in figure 2.17.

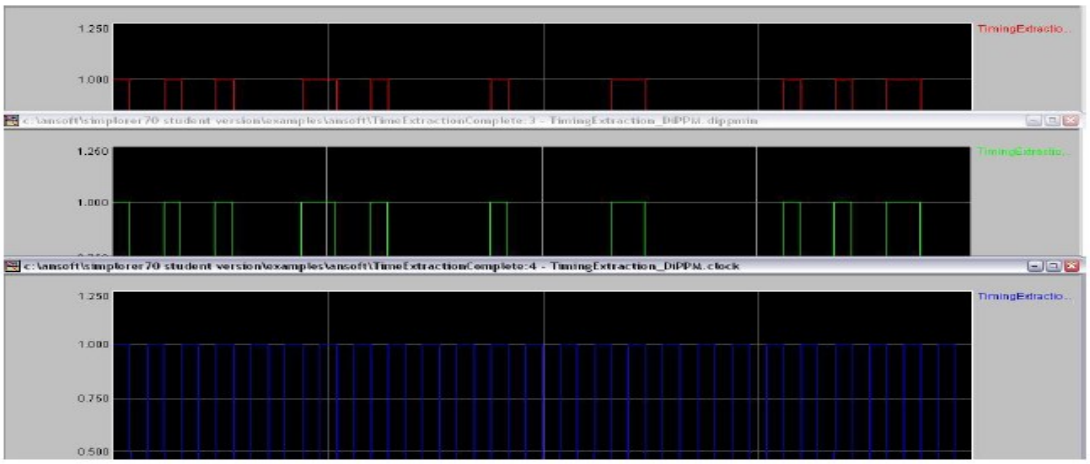


Figure 2.17 DiPPM in (top trace), DiPPM out (middle trace), clock recovered (bottom trace) (Charitopoulos, 2009)

2.6. Summary

The overview of the existing literature can be said to be highly beneficial in understanding the concepts, implementation and performance of DiPPM, its coder and decoder, based on programming, which will assist in the performance of the current research. Thus, it could be realized from the experiments, views and opinions of the early researchers that DiPPM has significant advantages over the standard system of communication systems. Thus, investigation of the DiPPM scheme through the optical channels holds significant importance in the world of electronic and optical communications. Development of the DiPPM coder and decoder in the VHDL language has also been significantly realized from the review of the literature.

The VHDL development is essential for the effective development of the coder and decoder based on which the timing extraction is also delivered, which in a combined way would lead to the effective implementation and investigation of performance of the Dicode PPM over dispersive optical channels. Another essential point which is part of the current research, and has been successfully reviewed from the literature as well, is the error correction method. The types of errors could be studied and the measures by which these may be corrected, could also be learnt well from the literature review. These understandings would now be utilized for the research and its findings.

DICODE PULSE POSITION MODULATION

3.1. Introduction

The availability of lasers which emit narrow pulses with high peak power means that pulse position modulation (PPM) is an attractive modulation scheme. PPM represents a method of intensity modulation**,** in which information to be communicated is located in the position of the optical pulse within a time frame divided into a certain number of segments (Band, 2006).

Thus, this technique involves the process of optical communication to occur in specific time slots, with the optical pulses being transmitted in these time slots. Two major advantages of PPM include its high intensity for the optical pulses, and low average power. These two factors are essential for the purpose of wireless communications and hence prove to be advantageous for the overall system. However, one problem with PPM is that the receiver is highly complex in nature and it needs to be synchronized effectively with the time slots and the frame (Band, 2006).

Several alternatives have been developed by researchers that reflect smaller expansion of bandwidth. Of these, multiple PPM and dicode PPM have been obtained to offer the lowest expansion of bandwidths. With DiPPM, only a single pulse is transmitted during availability of transitions between different levels of communication. Errors are also associated with the PPM techniques, which if not corrected, may result in corruption of the system (Ghosna & Sibley, 2010; Sibley, 2012).

Researchers have focused on determining the advantages of dicode pulse position modulation over digital pulse position modulation. It has been shown that the receiver of the DiPPM technique can be simplified by using central decision detection instead of slope. This is associated with achievement of the corresponding act of sensitivity in the fibres representing higher bandwidths and significantly superior performance at the lower bandwidths (Charitopoulos & Sibley, 2009).

Owing to the above reasons, the implementation of the DiPPM technique is found to be easy. This is more because with the technique of DiPPM, two slots are used for the process of transmission allowing transmission of one bit of PCM. Also, improved sensitivity is obtained with DiPPM and the slot rate is two times higher than the rate which PCM has in its original form. The coding of the data takes two steps as follows (Charitopoulos & Sibley, 2009):

- A PCM transition from zero to one with production of a pulse in slot S.

- A one to zero transition with production of a pulse in slot R.

There is no transmission of pulses in case when the data of the PCM is constant at 1 or 0. The functioning of the system is found to be advantageous for optical communications (Charitopoulos & Sibley, 2009).

3.2. Dicode Pulse Position Modulation: Understanding of the Theory

As explained in the introduction, the dicode pulse position modulation is mainly used owing to its advantages over the normal pulse position modulation. The simplification of the system makes it more convenient for use in optical communications. The processing of the data coding through the dicode pulse position modulation can be represented by the following figure 3.1:

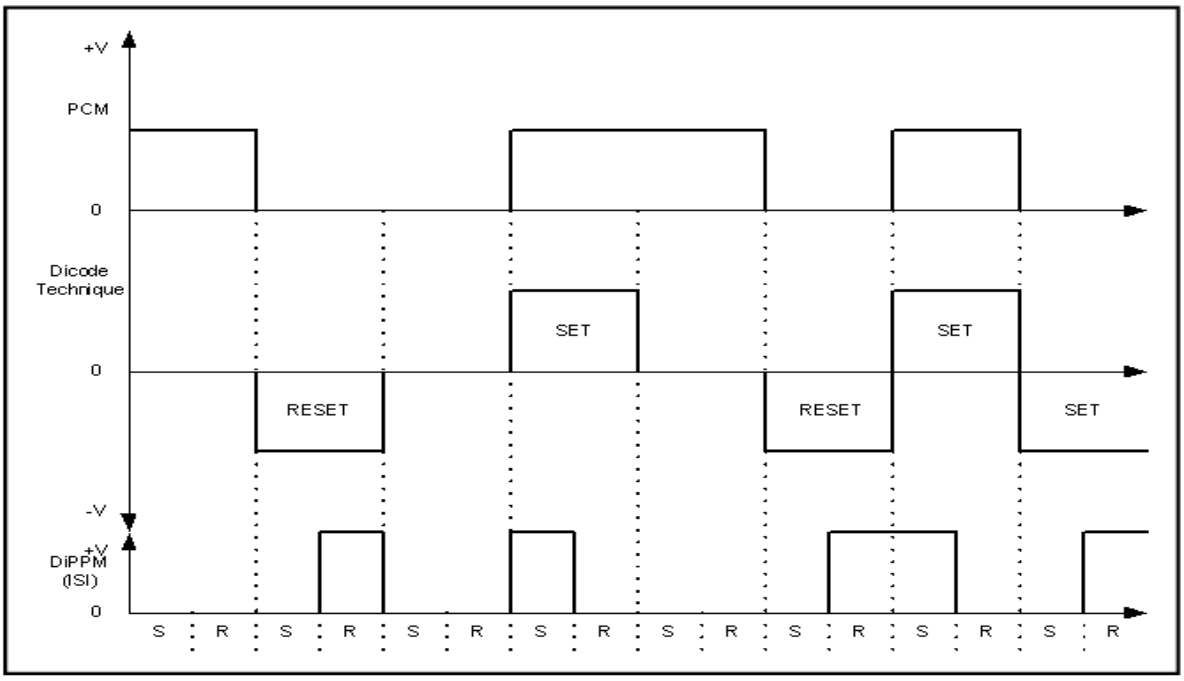


Figure 3.1 PCM data (top trace), Dicode technique (middle trace), and dicode PPM (bottom trace)

The dicode technique is mostly used in channels that are involved in magnetic recordings where in general there is limitation of bandwidths. The signalling format of this technique is such that no signal transmission occurs in this case when the data remains constant. Instead, transitions of data are sent. The formation of the dicode pulse position modulation takes place through combination of the original scheme of coding with a dicode. Considering the figure 3.1 above, in a dicode signalling format, the transitions of the data from logic zero to logic one are represented through code +V. On the other hand, -V represents the code for transition of data from logic one to logic zero. In case

when there is no change in the PCM signal, the signal transmitted through the system is the zero signal (Sibley, 2003).

Two pulses can be obtained from the transmission of the signal. These include a positive pulse and a negative pulse. When the data is set to logic one, it is reflected through the positive pulse. At this time, the pulse is SET. When the data is reset to logic zero, the negative pulse is obtained as pulse RESET. These signals of the SET and RESET pulses can be transformed into two pulse positions within the frame of a data. Hence the transition of the PCM from logic zero to one results in production of a pulse in slot S (representing SET pulse) and transition of logic one to zero results in production of the pulse in slot R (representing RESET pulse). Depending on the number of slots used in the system, the line rate can be determined. For instance, when four slots are used, the line rate is four times from that of the original PCM (Sibley, 2003).

The requirement of bandwidth in case of dicode PPM is much less than the digital PPM. Hence the use of dicode PPM is suitable in dense wavelength division multiplexing (DWDM) systems. The technique makes use of four symbol alphabets for representation of the PCM and dicode PPM. These are as shown in table 3.1 (Sibley, 2003).

Table 3.1 Dicode PPM Technique (Sibley, 2003)

PCM	Dicode PPM	Symbol
00	No pulse	N
01	SET	S
10	RESET	R
11	No pulse	N

The probabilities of the symbols *R, N, S* are ¼, $(1/2)^x$, and ½ respectively. After the transmission of an R pulse takes place, there are only two sequences of the PCM which are possible – 00 or 01. This is the reason for which the probability of the signal S is ½. With line coding in the original PCM, and limiting the run of the like symbols to n, the maximum run of the dicode PPM that would be achieved is *R, nN, S*. In this situation, the probability of the *S* symbol is one. This is so because in this case, the presence of the *S* symbol is certain at the end of the run which involves *n* number of *N* symbols (Sibley, 2003)

One act which is common to the digital PPM is that in the optimum filter for the dicode PPM receiver; there is a filter which is noise-whitened matched, along with a PDD network. The data for transmission is sliced by a voltage comparator and a flip-flop is used for application of the pulses, which is programmed depending on the rules of the decoding (Sibley, 2003). The working of the receiver can be understood from figure 3.2.

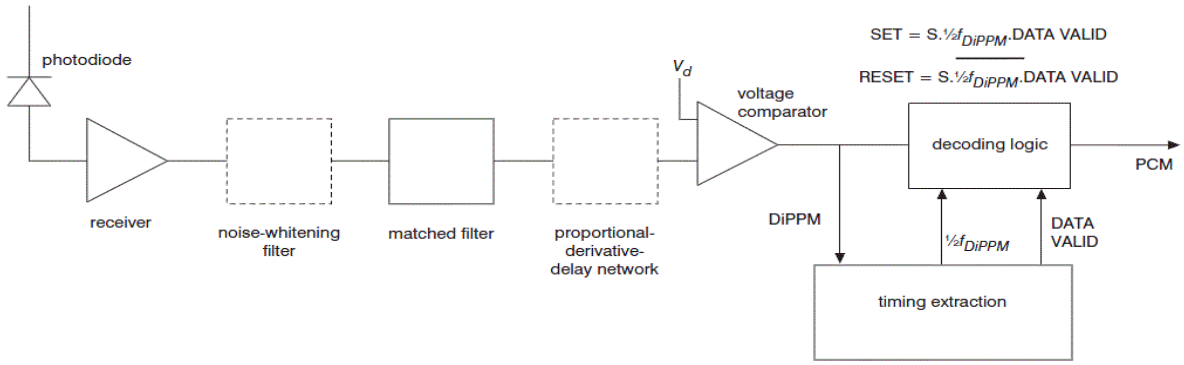


Figure 3.2 Schematic representation of the Dicode PPM Receiver (Sibley, 2003)

Only the active slots of the data to be transmitted are examined, and the process of decoding halts when the frame receives a valid pulse of the data. The synchronization of the frame can be maintained by extracting the slot clock from the data followed by generation of different phases of the signal of data. This can be presented through figure 3.3 (Sibley, 2003)

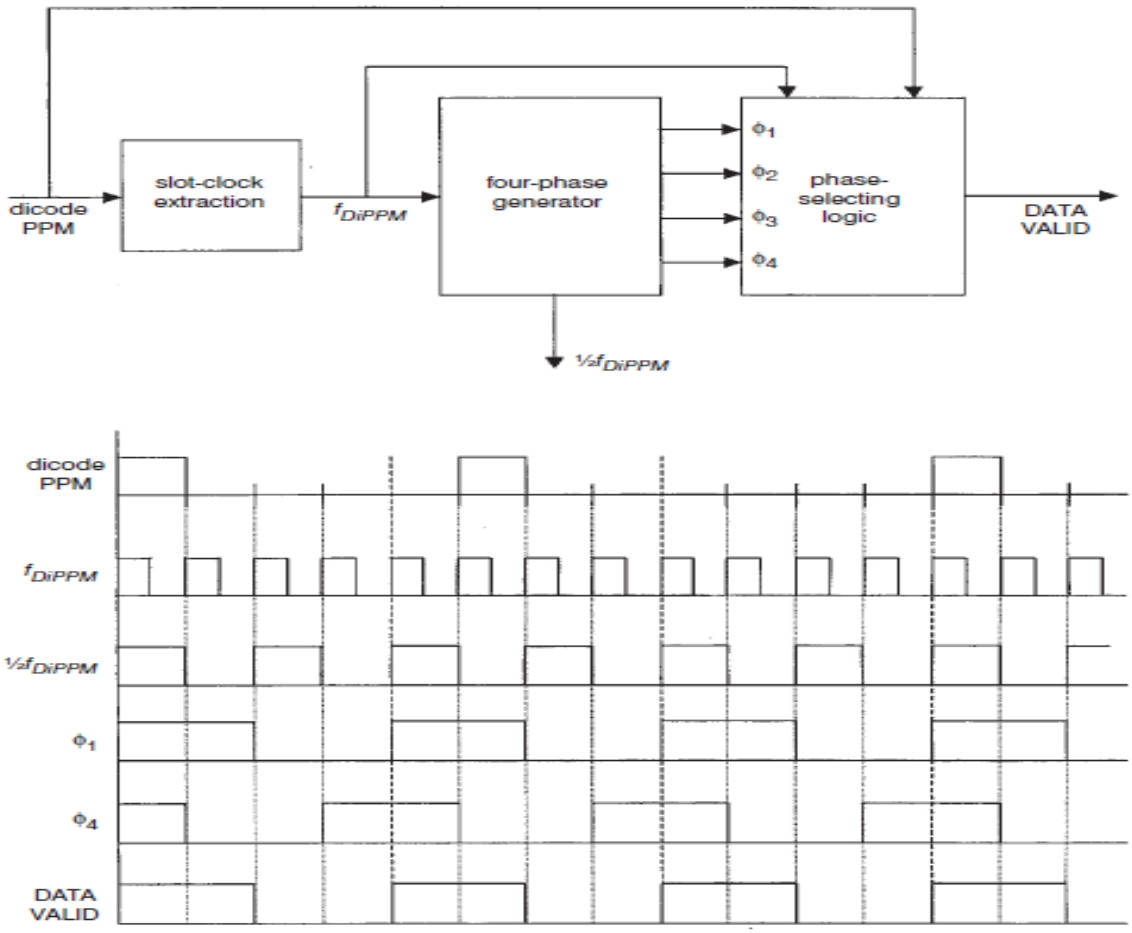


Figure 3.3 Block diagram of proposed timing extraction system and related timing diagram (Sibley, 2003)

3.2.1. DiPPM system optical power

The required optical power for the digital PPM system, P_{DPPM}, can be found by using equation

$$P_{DPPM} = bhv\frac{B}{m} \tag{3.1}$$

Where *b* is the minimum number of photons, *m* is the number of PCM coded bits and *B* is the original data rate $(1/T_b)$.

For the DiPPM system, the required optical power, P_{DiPPM}, is given by (Sibley, 2003)

$$P_{DiPPM} = bhv\frac{n+1}{8n}B \tag{3.2}$$

Where *n* is the maximum number of consecutive like symbols for DiPPM.

Al-Suleimani et al (2008), argued that in case $n\rightarrow\infty$ the probability of having a pulse in the frame tend to *1/2* and not *1/8*. Hence, a new power equation was derived

$$P_{DiPPM} = bhv\frac{2^n}{2^{n+1}-1}B \tag{3.3}$$

By using equation (3.3), the researchers found that the optimal PPM scheme outperforms the DiPPM by between 3 and 5dB, depending on the normalised fibre bandwidth, figure 3.4 (Al-Suleimani, Phillips & Woolfson, 2008). However, Sibley claims that digital PPM has a single pulse in a frame of slots - empty apart from the active slot. Hence the average signal is going to be *1/slots = 1/2*^{m}. In DiPPM there are four codewords each equiprobable with a probability of *1/4*. There is a one pulse in a frame of two slots. Therefore the average is *b/2*1/4 + b/2*1/4 + 0*1/4 + 0*1/4*. This gives an average of b/4 (Sibley, 2003).

Al-Suleimani et al (2008), also investigated the performance of DiPPM and compared with the PPM an OOK NRZ, in term of sensitivity as a function of DiPPM run length at two different bandwidths 622Mbit/s, 2.5Gbit/s, depending on equation (3.3). The researchers argued that the PPM significantly outperforms DiPPM, however the PPM coding level should not overcome 4 at *fn*=3 and 7 at *fn*=10. Moreover, the DiPPM becomes more sensitive with increasing n, as shown in figure 3.5. This results was dependent on their new derived sensitivity equation (Al-Suleimani, Phillips & Woolfson, 2008). Sibley (2003), showed that there is an improvement in sensitity of 0.2dB if the DiPPM line coding *n* increases form 5 to 10.

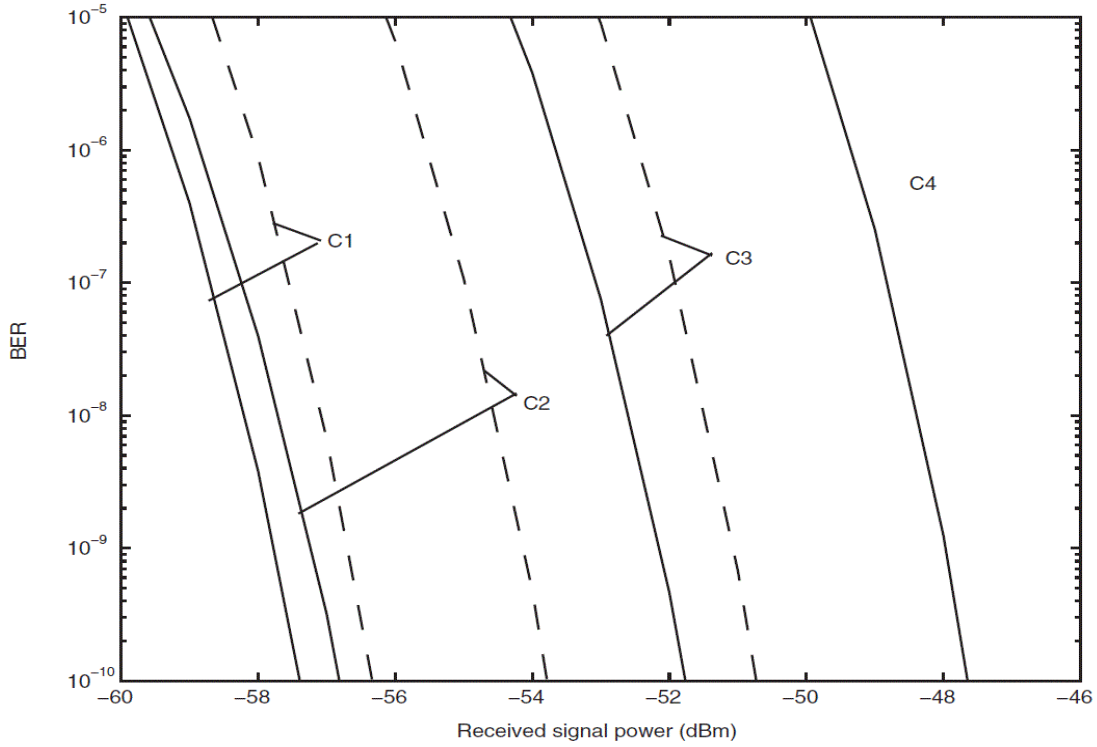


Figure 3.4 BER against received signal power at 622 Mbit/s. C1, DiPPM—using Equation (3.2) (for fn=3 at n=10 and fn=10 at n=10); C2, PPM (for fn=3 at M=4 and fn=10 at M=7); C3, DiPPM—using Equation (3.3) (for fn=3 at n=10 and fn=10 at n=10); C4, OOK NRZ; ------fn=3; _____ fn=10 (Al-Suleimani, Phillips & Woolfson, 2008).

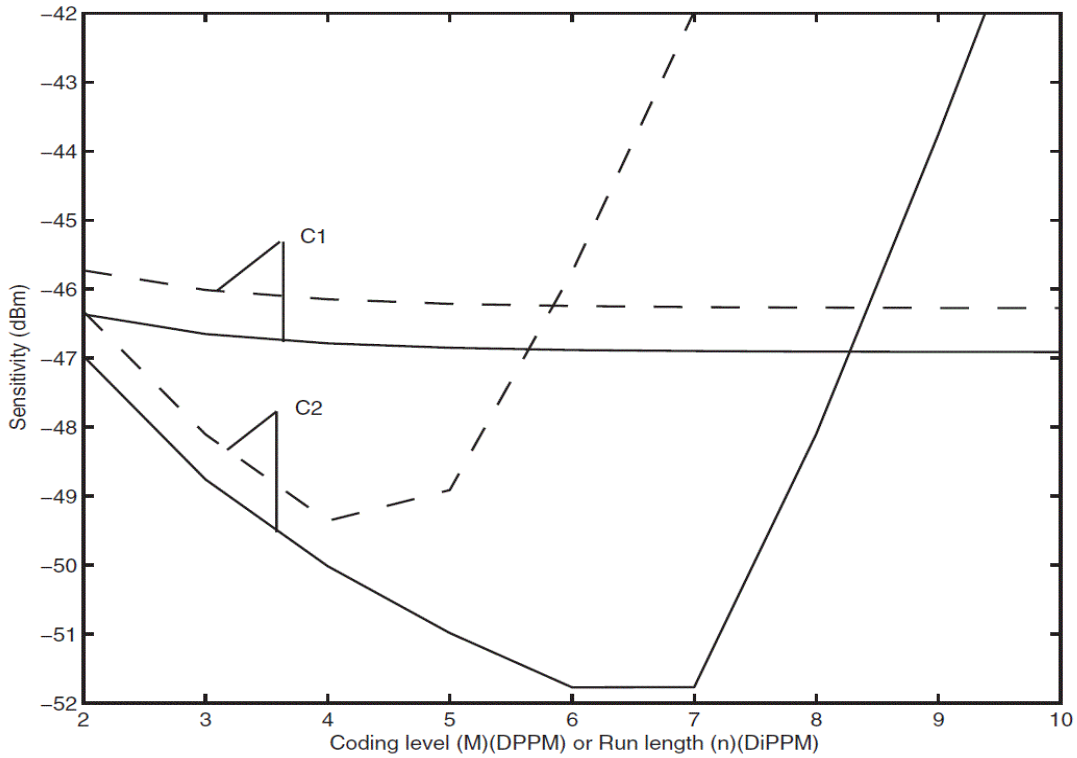


Figure 3.5 Sensitivity as a function of coding level (PPM) or run length (DiPPM) at B=2.5 Gbit/s. C1, DiPPM; C2, PPM; ----- fn=3; _____ fn=10 (Al-Suleimani, Phillips & Woolfson, 2008).

3.3. Errors Affecting DiPPM

There are three types of errors that affect the system: wrong-slot errors, erasure errors, and false alarm errors. With the use of the DiPPM technique in optical communications, the probabilities of the errors need to be determined (Sibley, 2003).

3.3.1. Wrong-slot Errors

These errors take place in cases when a pulse is caused to arrive early or late due to noise on the slope of a detected pulse being large enough to cause a false trigger. In order to reduce this error, it is essential to detect the pulse in the center of the time slot the width of which represented as T_s. Hence, the generation of the errors occurs with the movement of the edge takes place by $|T_s/2|$. The probability of the error is represented as P_{es}, which appears in the preceding slot. It has been given by (Sibley, 2003):

$$P_{es} = 0.5 erfc\left(Q_{es} / \sqrt{2}\right) \tag{3.4}$$

Here, Q_s is given by (Sibley, 2003):

$$Q_{es} = \left[T_s slope\left(t_d\right)\right] / \left[2\sqrt{n_o^2}\right] \tag{3.5}$$

Where, n_o^2 represents the mean square of the noise of the receiver, and *slope* (t_d) represents the slope of the pulse that has been received at the instant of the threshold crossing, marked at t_d (Sibley, 2003).

In the case dicode PPM, there are four possible errors that can take place as a result of a wrong-slot event. Depending on the position of the pulse in the slot, the edge can appear in the preceding slot or in the following. In case it appears in the preceding slot, detection error will not be obtained and no recognition of the false threshold crossing would be obtained by the decoder. Hence such a detection error results in error in the PCM and the bits that follow from this step until the R pulse is received (Sibley, 2003). The transmission and receiving of sequences in case of a wrong slot event have been given in the following table 3.2 (Sibley, 2003):

Table 3.2 Transmitted and received sequences with a wrong-slot error (Sibley, 2003)

Transmitted	S	xN	R
Received	R	xN	R
Probability	¼ Ps	(1/2)^x	1/2

The above transmission and receiving of signals is achieved when the number of *N* signals is *x* (Sibley, 2003). The theory of MLSD or maximum likelihood of sequence

detection has been associated with the correction of these errors and is tried to apply MLSD for the detection of the errors. This represents detection of all the types of errors that DiPPM is suffering from (Charitopoulos, Sibley & Mather, 2010).

As the wrong slot error is concerned, a pulse placed in the slot *R* may appear in the preceding slot *S* of the frame or in the slot *S* which is following in the frame representing the next frame. The first case has been given in the following table 3.3 (Sibley, 2005):

Table 3.3 Wrong-slot pulse error and method of detection for MLSD of dicode PPM (Sibley, 2005).

Pulse Error	Invalid Sequence	Detection Method
S ⟵ R	S*x* N *SR* *y* N S	Double pulse in frame
R ⟶ S	S*x* N *N S* (*y*-1) N S	Three consecutive S symbols
R ⟵ S	R (*y*-1) N *R Sx* N R	Corrected to R *y*N S *x*N R
S ⟶ R	R *y* N *R x* N R	Three consecutive R symbols

Errors related to wrong slot events are associated with pulses that are highly dispersed. Thus, it is essential to remove them absolutely from the operation of the low bandwidth. Considering this scenario, the *R (y-1) NRS* sequence is said to occur owing to the wrong slot event and irrespective of the source of the error, *RyNS* is used for the correction of the error. However, when the maximum number of like symbols gets exceeded, then it is considered as an exception. This case represents a case where erasure has taken place in the source of the error, which is another form of error which makes the functioning of the DiPPM difficult and hence needs to be corrected (Sibley, 2005).

3.3.2. Erasure Errors

Erasure errors occur in cases when the noise on the pulse is large and capable of reducing the voltage of the peak signal such that it falls below the threshold level. The probability of the error, which can be represented by P_{er}, can be given by (Sibley, 2003):

$$P_{er} = 0.5\,erfc\left(\frac{Q_{er}}{\sqrt{2}}\right) \tag{3.6}$$

Here,

$$Q_{er} = \left(v_{pk} - v_d\right) / \sqrt{n_o^2} \tag{3.7}$$

Where, v_{pk} represents the voltage of the peak signal as obtained at the receiver's output, and v_d represents the voltage of the threshold crossing (Sibley, 2003).

In the case of a dicode PPM technique, the same number of PCM errors is generated by erasure of SET or RESET pulse (Sibley, 2003). As the erasure errors occur in the system the *R* and *S* pulses of the data transmission get erased and they get converted into *N* symbols. Thus consecutive like symbols are possible to be generated those are in the form of *SxNNyNS* or *RxNNyNR*. The use of MLSD is done in this case in order to try to correct the codes that are not valid for the transmission of the data. This correction can be presented as given in the table 2.2 (Sibley, 2005).

Consecutive *S* symbols are left as a result of the erasure of the *R* pulse. These *S* symbols are separated by *N* symbols resulting in different positions for the *R* pulse that has been erased owing to the error. All possible sequences of the PCM are detected and from these the most likely code for the missing pulse is tried to be determined, and the most likely sequence is determined by averaging all the bits of the sequence (Sibley, 2005). As owing to the error it is not possible to obtain the exact position of the original *R* pulse, hence the MLSD is used for insertion of an *R* pulse in the next slot of the data sequence which is given by *x/2* or *y/2* (Charitopoulos, Sibley & Mather, 2010).

3.3.3. False Alarm Errors

False alarm errors are caused owing to the formation of noise in the data transmission that results in a threshold crossing event in any slot that has remained unoccupied. The probability of this error, P_t, has been given by (Sibley, 2003):

$$P_t = 0.5\,erfc\left(\frac{Q_t}{\sqrt{2}}\right) \tag{3.8}$$

Where,

$$Q_t = v_d \,/\, \sqrt{n_o^2} \tag{3.9}$$

The number of samples that are uncorrelated depending upon each time slot can be determined as T_s / τ_R where τ_R represents the time during which the function of the autocorrelation in the filter of the receiver becomes very small. The probability of the false alarm error, P_f , is given by (Sibley, 2003):

$$P_f = \frac{T_s}{\tau_R} 0.5 erfc\left(\frac{Q_t}{\sqrt{2}}\right) \tag{3.10}$$

In dicode PPM, in order for PCM errors to occur, it is essential for the false alarm error to be opposite type to that of the symbol with which the sequence had been initiated. For instance, in case where the false alarm error occurs in the following *R* slot, caused due to the pulse in the *S* slot, then the decoder would stop when it receives the pulse, and hence the detection of PCM errors would not take place. However, the generation of the error would take place when false alarm takes place in the *N* strings of the following slot of the sequence. In such a case, the error and its severity largely depend on the location of the occurrence of the false alarm, which has been given in the following table 3.4 (Sibley, 2003):

Table 3.4 Transmitted and received sequences with a false-alarm error (Sibley, 2003)

Transmitted	S	N	N	N	N	N	R
Received	S	N	N	R	N	N	R

The occurrence of the false alarm error can be better understood from this. In case when the amplitude of the noise of the sequence is higher than the level of the threshold, then the occurrence of false *S* or *R* pulses is possible if a slot is empty. As a result of the occurrence of false *R* pulse, it is possible that the *S6NR* might get converted into *SNR4NR*. This has been presented in the following table 3.5 (Sibley, 2005).

Table 3.5 MLSD detection of a DiPPM sequence in which a false R symbol has been detected (Sibley, 2005)

Invalid sequence	S	N	R	N	N	N	N	R	Binary representation							
	S	N	R	S	N	N	N	R	1	1	0	1	1	1	1	0
	S	N	R	N	S	N	N	R	1	1	0	0	1	1	1	0
	S	N	R	N	N	S	N	R	1	1	0	0	0	1	1	0
	S	N	R	N	N	N	S	R	1	1	0	0	0	0	1	0
Average									4/4	4/4	0/4	¼	2/4	¾	4/4	0/4
MLSD output	S	N	R	N	N/S	S	N	R	1	1	0	0	0/1	1	1	0
Original word	S	N	N	N	N	N	N	R	1	1	1	1	1	1	1	0
Error bits (2.5 off)											1	1	1/0			

Table 3.5 represents the operation of the MLSD in case of the occurrence of the false alarm error. The false R pulse is considered as the valid bit and the role of the MLSD here is to insert a correct *S* pulse similar to the case of the erasure errors. However as the PCM provides with an average of 2/4, hence a suitable output is not possible for the MLSD to provide with, since it is probable that the logic could be either 0 or 1 depending

on the rounding up or rounding down of the result. In such a situation, the design of the decoder can be so obtained such that one of the two decisions can be considered to be true for the case. Thus as far as the MLSD is concerned, it is capable of introducing errors in the system as well as correcting them and hence is used in the error detection and correction mechanism in dicode PPM technique (Sibley, 2005).

Table 2.3 presents a comparison of error probabilities at specific normalised link bandwidths for dicode PPM operating with and without MLSD for better understanding of the error formation and detection with implementation of the MLSD in DiPPM.

3.3.4. DiPPM Error Probabilities

DiPPM uses a four symbol alphabet; a typical sequence would be *S, xN, R* with symbol probabilities of *1/2, (1/2)^x* and *1/4*. The *S* signal has a probability of *1/2* because there are only two possible PCM sequences which are (00 or 01) after an R pulse has been transmitted. If the original PCM is line coded in order that the run of like aspect of the symbols is limited to *n*, the maximum DiPPM run would be *R, nN and S*. In this sequence, the *S* has a probability of one due to its presence being guaranteed at the end of a run of n lots of *N* symbols (Cryan, & Sibley, 2006).

In DiPPM, the shape of the *S* and *R* pulses will depend on the transmitted pattern. The new pulse shapes must be found using a general DiPPM sequence *S, xN, R, yN* and *S*. The general form of the total DiPPM binary error probability $\{P_{eb}\}$ can be computed from the summation of the equivalent PCM probability of errors for DiPPM error sources that consider a complete sequence for all *x* and y (Sibley, 2003; Cryan, & Sibley, 2006):

$$P_{etotal} = P_{es} + P_{er} + P_{efR} + P_{efN} \tag{3.11}$$

Where $\{P_{etotal}\}$ is the equivalent PCM error probability due to wrong-slot errors, which is equal to:

$$P_{es} = 3\left[\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^2 P_s(x+1) + \left(\frac{1}{2}\right)^{n+2} P_s(n+1)\right] \tag{3.12}$$

The PCM error probability for erasures $\{P_{er}\}$ is:

$$P_{er} = 2\left[\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^2 P_r(x+1) + \left(\frac{1}{2}\right)^{n+2} P_r(n+1)\right] \tag{3.13}$$

False alarm error may occur between *S* and *R* pulses where there is no ISI. Hence, the number of PCM decoding errors will depend on the symbol´*s* position, *k*, where the false alarm error appears within the run of N-symbols. Thus, the equivalent PCM error probability (Sibley, 2003; Cryan, & Sibley, 2006):

$$P_{efN} = \sum_{x=1}^{n-1}\left(\frac{1}{2}\right)^{x+3}\sum_{k=1}^{x}(x+1-k)P_f + \left(\frac{1}{2}\right)^{n+2}\sum_{k=1}^{n}(n+1-k)P_f + \sum_{x=2}^{n-1}\left(\frac{1}{2}\right)^{x+3}\sum_{k=2}^{x}(x+1-k)P_f + \left(\frac{1}{2}\right)^{n+2}\sum_{k=2}^{n}(n+1-k)P_f \tag{3.14}$$

A numerical evaluation, which had been conducted recently reviewed the production of DiPPM by applying a slope detection method in a distributed surrounding (Cryan & Sibley, 2006). In this research, thought is delegated to a distinct discovery perspective and novel outcomes are introduced. These perspectives demonstrate that the ISI (Inter Symbol Interference) can be eradicated by the application of the central decision detection method, which are applied in conjunction with a raise cosine filter. The pragmatic implementations of this perspective are evaluated and have the outcome of similar sensitivity tolerance at the elevated fiber bandwidths of the third-order Butterworth filter and the pre-amplification system (Shalaby, 1999; Sibley & Massarella, 1993; Sibely, 1993; Sibley, 2003; Sibley, 2004; Zwillinger, 2004).

In the Cryan and Sibley study it is demonstrated that establishing the equalizer and the preamplifier to sixty percent of the dicode slot rate yields tolerance sensibilities which are within *0.2 dB* of the optimal dicode pulse position monitor raised cosine selection and that this symbolic enables the receiver to function in a broader scope of fiber bandwidths with minimal decomposition in the tolerance sensibilities (Cryan & Sibley, 2006).

An alternate detection technique for dicode PPM is suggested, which is founded on the application of central decision discovery and raised cosine filtering. This strategy infers that the dicode PPM pulse can distributed into adjacent time shifts in the absence degrading the performance. This is attributed to a pulse being sampled at the centre of the slot. The application of raised cosine filtering guarantees that the voltage due to adjacent pulses is established at zero at the decision instance, consequently eliminating ISI. As a result the likelihood of wrong-slot errors is minimal, and so P_{etotal} converts to:

$$P_{etotal} = \left[1 - \frac{1}{2^{n+1}}\right]P_r + \left[\frac{3}{2} + \frac{2n+5}{2^{n+2}}\right]P_{fN} \qquad for\ n > 1 \tag{3.15}$$

3.4. Coder and Decoder Circuits for the DiPPM

The coder of DiPPM forms an element of the technique of the DiPPM that enables formatting of PCM sequences into such sequences that constitute the symbols of the DiPPM alphabets. The DiPPM coder can be completed with the use of logic components that include flip-flops, and five NOR gates. This can be presented through the following figure 3.6 (Charitopoulos, 2009):

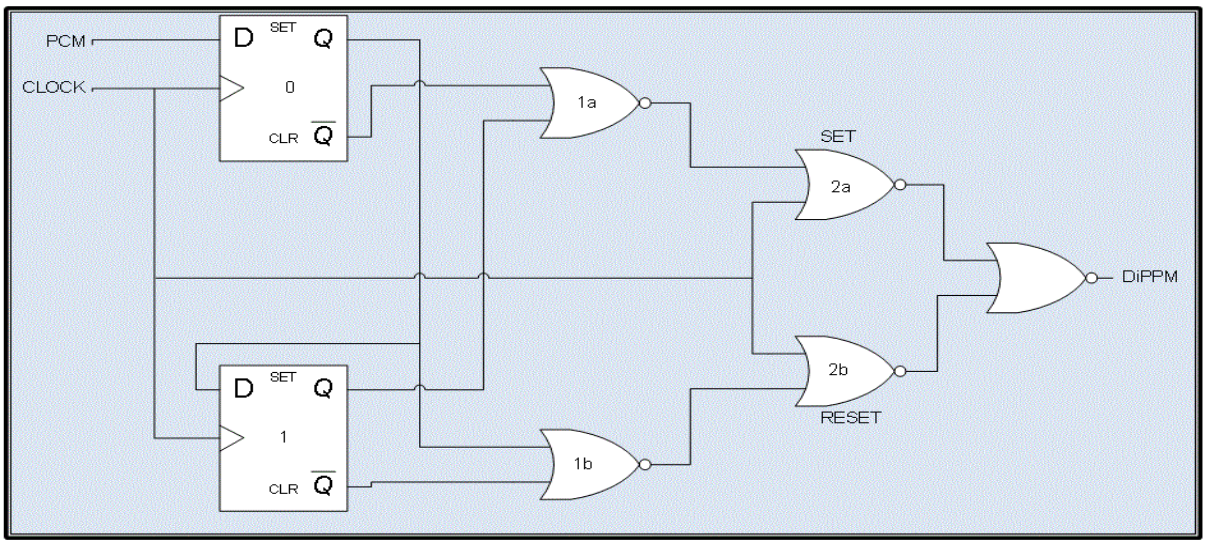


Figure 3.6 DiPPM coder circuit (Charitopoulos, 2009)

A two-bit store is formed by Flip-Flop 0 and Flip-Flop 1, and the resulting outputs can be used for the generation of *SET* and *RESET* sequences. The *SET* sequence is produced by passing the $\overline{Q0}$ and *Q1* through the NOR gate 1(a), and the *RESET* sequence is generated by passing the pair $\overline{Q0}$ and *Q1* through the NOR gate 1(b). In order to retime the *SET* and *RESET* sequences of the DiPPM system, the use of the CLK and NOT CLK would be necessary to obtain. The 2a and 2b NOR gates result in the production of the *SET* and *RESET* sequences of the DiPPM. The final DiPPM sequence is formed by the combination of the sequences by the NOR gate (Charitopoulos, 2009).

Measurement of DiPPM coder can be done through the Power Spectral Density (PSD). The diagram figure 3.7 represents the PSD of the deterministic DiPPM signal (Charitopoulos, 2009). Also, the DiPPM coder waveforms can be understood from the following figure 3.8:

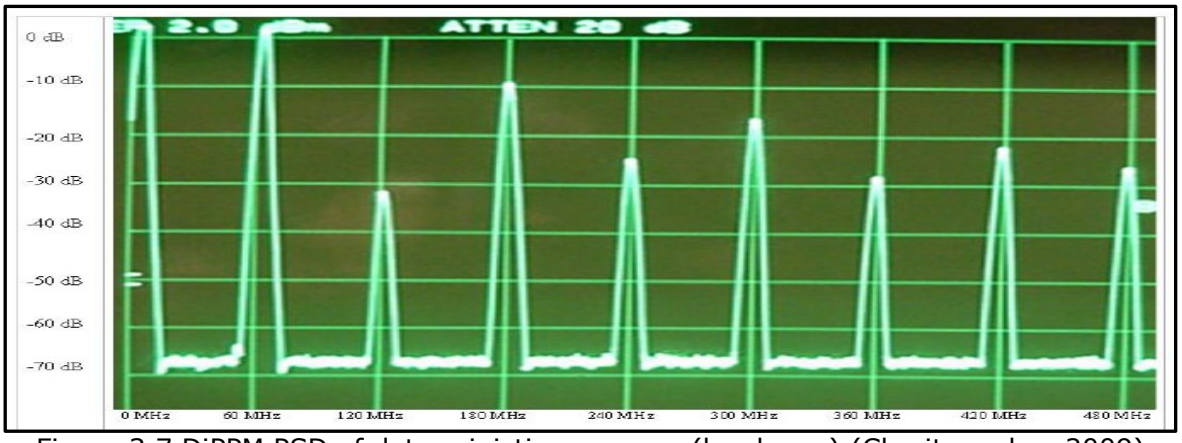


Figure 3.7 DiPPM PSD of deterministic sequence (hardware) (Charitopoulos, 2009)

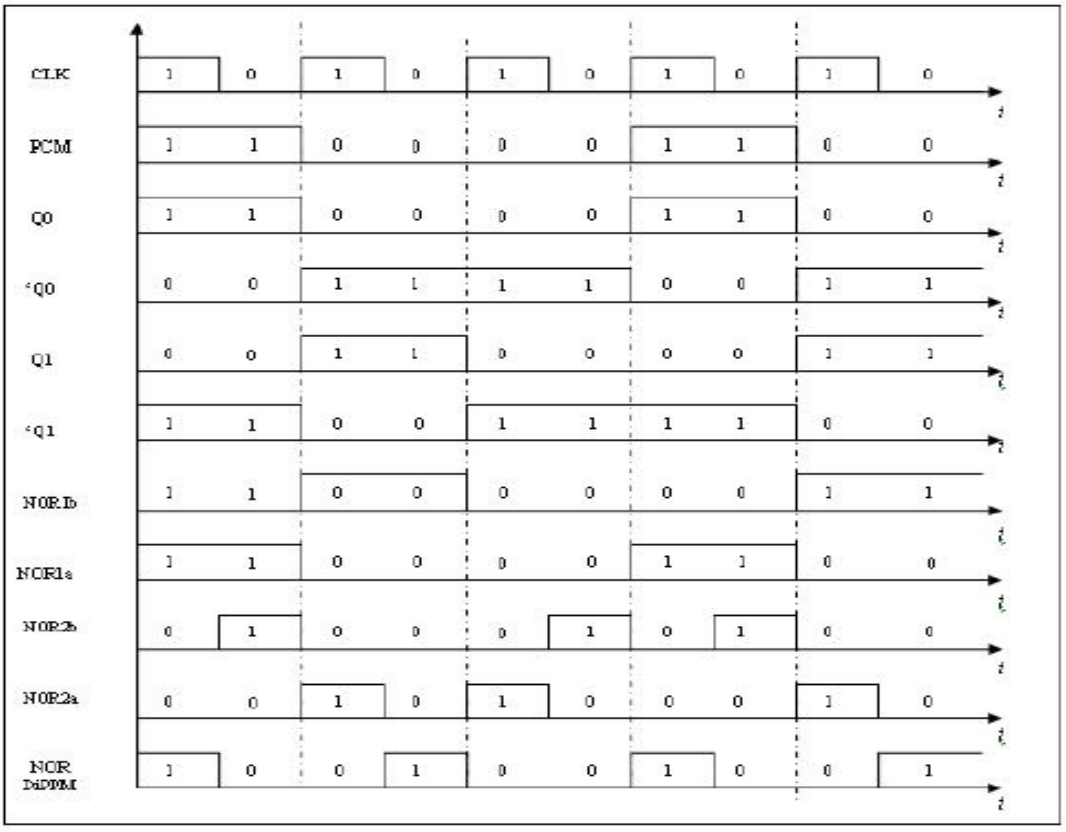


Figure 3.8 DiPPM coder's waveforms (Charitopoulos, 2009)

The use of the DiPPM decoder is in conversion of the signal of the DiPPM that has been coded into its original PCM format. The elements of a DiPPM decoder include a NOR/OR gate, three NOR gates, a D type Flip-Flop and a Direct Set/Clear component. The DiPPM decoder has been presented in the following diagram figure 3.9 (Charitopoulos, 2009):

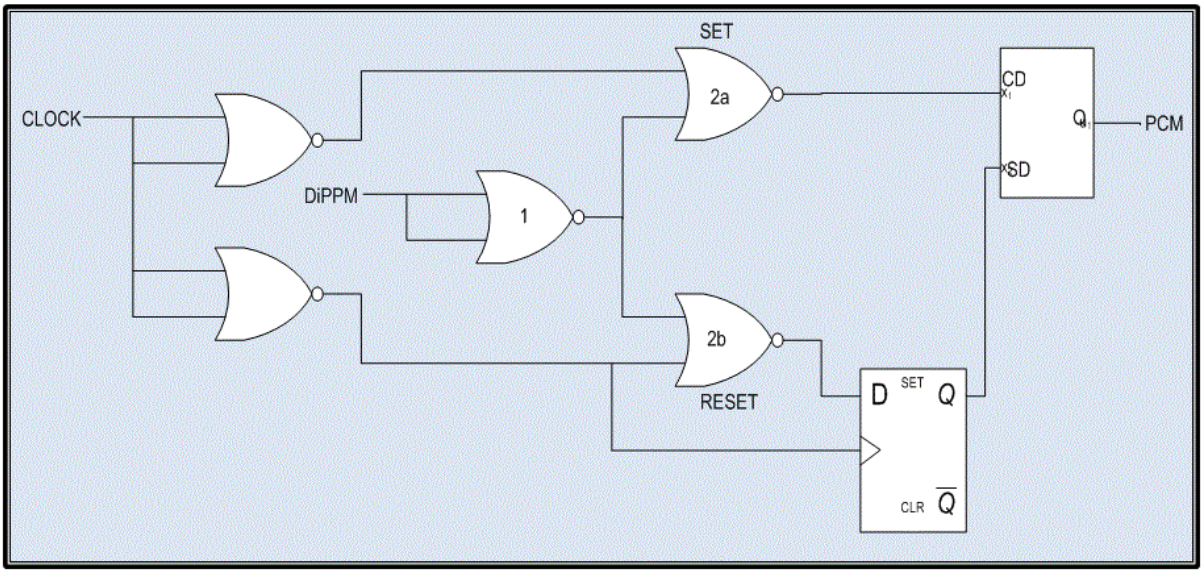


Figure 3.9 DiPPM decoder circuit (Charitopoulos, 2009)

A double OR gate can be used for buffering of the clock signal. The output of the buffer which is synchronized is made to pass through the coaxial wires, which are of same length one of which passes to the coder and the other to the decoder. This results in synchronization of both the clock signals. The clock and NOT clock signals are used in the case of decoder as well and the generation of these signals occurs through passage of the clock signal by a NOR/OR gate. The NOR gate is capable of inverting the input signal of the DiPPM to the decoder before it is gated to the clock and NOT clock signals enabling the production of the *SET* and *RESET* sequences which are independent to the system. The PCM signal is produced by the *SET/CLEAR* component, with the amplitude being very high when the sequence SET is one and zero in case when the *RESET* sequence is 1 (Charitopoulos, 2009).

The DiPPM decoder's waveforms can be understood from the following diagram figure 3.10 (Charitopoulos, 2009):

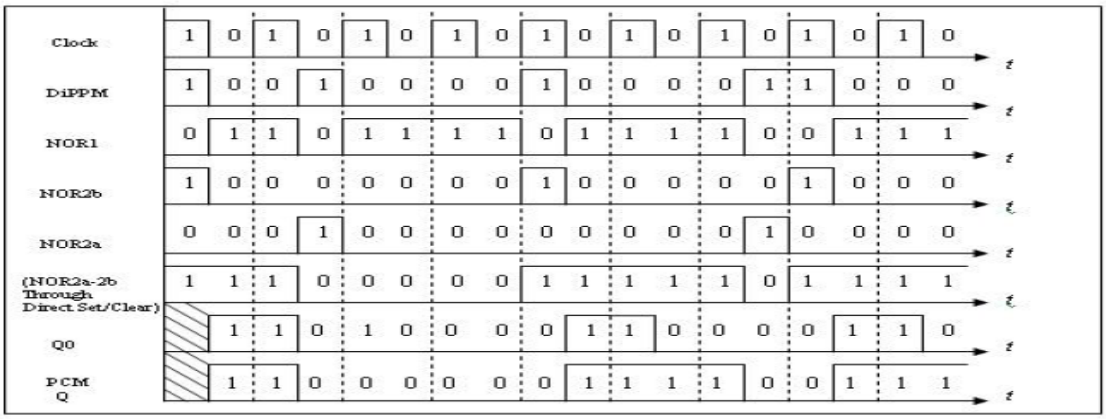


Figure 3.10 DiPPM decoder's waveforms (Charitopoulos, 2009)

Owing to the passage of the *SET* signal through components other than the *RESET* signal, there exists a synchronization fault. Hence an addition of a delay takes place in the D Flip-Flop component at the sequence of the *SET* pulse. The measurement of the decoder is also done similarly like the coder measurement when the input signals result in the deterministic outcome of the system. The above measures represented the construction of the coder and decoder circuits of the DiPPM. However, researchers are still investigating the correctness of the circuits, the errors and their correction towards effective implementation of the DiPPM system in optical communications (Charitopoulos, 2009).

FINDING OPTIMUM PARAMETERS FOR REED SOLOMON CODE WORKING WITH DICODE PULSE POSITION MODULATION SYSTEM

4.1. INTRODUCTION

This chapter will review the application of the RS code with regards to DiPPM. DiPPM has been presented as another paradigm in comparison to the digital pulse position modulation as it demonstrates similar receiver sensitivity while functioning at substantially decreased line rates. The precise application of the RS code minimizes the errors in coding schemes (Shalaby, 1999; Sibley & Massarella, 1993; Sibley, 1993; Sibley, 2003; Sibley, 2004; Zwillinger, 1988; McEliece, 1979, 1981).

The non-coded DiPPM which applies MLSD and the RS coding paradigms will be compared with regards to the quantity of photons which are required to be contained in each pulse and the effectiveness of the transmission. The object of this chapter is to find the optimum parameters, which achieve higher transmission efficiency and lower number of photons, for the RS code working with DiPPM.

4.2. FORWARD ERROR CORRECTION SYSTEM MODEL

The slope detection and central detection methods are used to find the optimum parameters of a RS code with a DiPPM system. First of all a model of that system should be developed, to start with simulation. A system model attempts to simulate some characteristics of a system. The model matches up the forward error correction (FEC) communication scheme, which is dependent on a RS error-control code, and shown in fig 4.1. The performance of each block of the model is described in Mathcad software, Appendix (1).

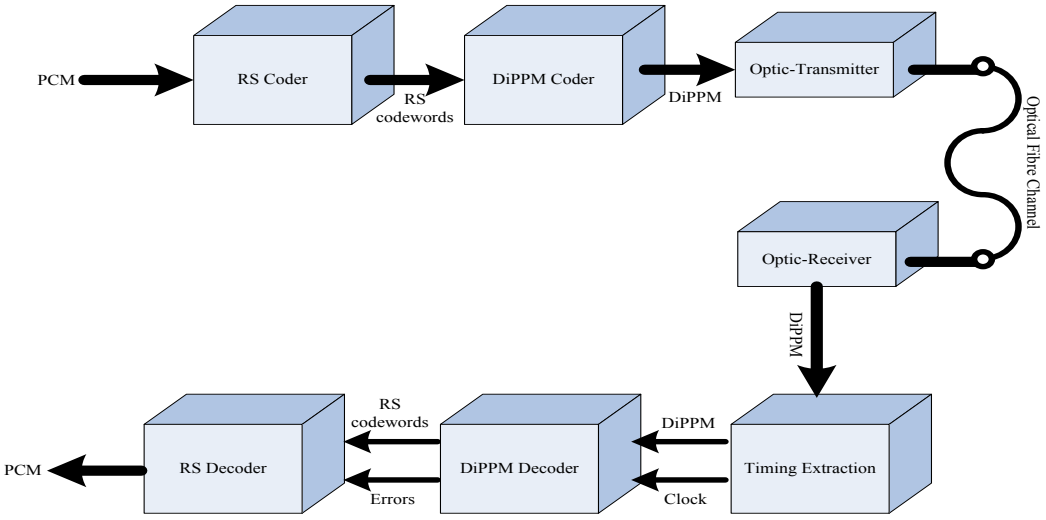


Figure 4.1 Block diagram of forward error correction

4.2.1. Slope Detection Approach

A fixed bandwidth, 1 GHz, PIN-bipolar (PIN-BJT) transimpedance optical receiver was considered in the DiPPM receiver with noise spectral density of 24 × 10^{-24} A^2/Hz. An operating wavelength of 1.55 μm and a photodiode quantum efficiency of 100% were taken and simulations were carried out using an original NRZ OOK data rate of 1 Gbit/s with line coding that resulted in n=10. Gaussian shape received pulses were assumed corresponding to a link bandwidth of 1.8 GHz. A threshold variable v was defined as:

$$v = \frac{v_d}{v_{pk}} \tag{4.1}$$

Where $\{v_d\}$ is the threshold crossing voltage and $\{v_{pk}\}$ is the peak voltage of the signal. The total equivalent PCM error probability is obtained initially by adding together the individual probabilities of DiPPM, which should be the same as for a PCM system considering 1 error in 10^9 pulses. Then equations (2.9) and (2.10) are used to compute the probability of error for the coded system with different code rate and code length. The decision time t_d, can be determined and the number of photons per bit b, can be found.

The transmission efficiency {ρ} for the uncoded and coded DiPPM, can be written as in equations (4.2) and (4.3) respectively. Equation (4.2) shows the transmission efficiency for uncoded DiPPM, where b is the number of photons while equation (4.3) shows the transmission efficiency for DiPPM when a RS code is applied. It can be seen from equation (4.3) that applying an RS code reduces the transmission efficiency of the system by the code rate r. However, at the optimum code rate, the application of a RS

code reduces the number of photons to achieve an overall improvement in transmission efficiency.

$$\rho = \frac{\ln 2}{b} \qquad (\frac{nats}{photon}) \tag{4.2}$$

$$\rho = r\frac{\ln 2}{b} \qquad (\frac{nats}{photon}) \tag{4.3}$$

The bandwidth expansion $\left(BWE\right)$ for coded DiPPM can calculate using equation (4.4).

$$BWE_{coded\ DiPPM} = \frac{n}{k} \times BW_{DiPPM} \tag{4.4}$$

4.2.2. Central Detection Approach

The model is based on the paradigm suggested by Sibley (2005). A block diagram of the receiver system is shown in fig. 4.2. The simulation used an optical receiver with a limited bandwidth ω_c, and a white-noise spectrum at its output. A classical matched filter has been used as the pre-detection filter due to the aspect of the receiver having a white-noise spectrum. Transmission of dicode PPM through graded-index POF had been considered and the signal presented to the threshold detector was (Sibley, 2005).

$$v_o(t) = b\eta qR_T \tfrac{\omega_c}{2} \exp(\alpha^2\omega^2) \times \exp(-\omega_c t) erfc[\alpha\omega_c - (\tfrac{t}{2\alpha})] \tag{4.5}$$

where b is the number of photons per pulse, η is the quantum efficiency of the detector, q is the electronic charge, R_T is the mid-band transimpedance of the receiver, and α is the variance of the received Gaussian pulse. This is linked to the fibre bandwidth by (Sibley, 2005).

$$\alpha = \frac{0.1874T_b}{f_n} \tag{4.6}$$

where T_b is the PCM bit-time and f_n is the fibre bandwidth normalised to the PCM data rate. The noise appearing on this signal is given by (Sibley, 2005).

$$\left\langle n_o^2 \right\rangle = S_o \tfrac{\omega_c}{2} R_T^2 \exp(\alpha^2\omega_c^2) erfc(\alpha\omega_c) \tag{4.7}$$

where S_o is the double-sided, equivalent input-noise current spectral density of the preamplifier. A PIN photodiode was used so that its shot noise could be neglected. The

time, at which the autocorrelation function of the noise at the output of the filter becomes small, has been taken to be α, thus $\tau_R=\alpha$. The threshold level, ν, was used as a system variable defined by equation (4.1), where ν_{pk} is the peak voltage of an isolated pulse. For a given fibre bandwidth, the pulse shape and noise can be determined, and the optimum value of ν that produces the lowest number of photons per pulse, b, can be found for a specified PCM error rate (1 in 10^9 in the simulations). A 1 Gbit/s PCM data-rate system, operating at a wave-length of 650 nm and a photodiode quantum efficiency of 100%, was considered. The preamplifier had a bandwidth of 10 GHz and white noise of 50 x 10^{-24} A^2/Hz when referred to the input. These parameters were obtained from a commercial device. Line-coded PCM data was used so that $n_{DiPPM}=10$. Simulations were conducted on DiPPM systems operating with and without RS code.

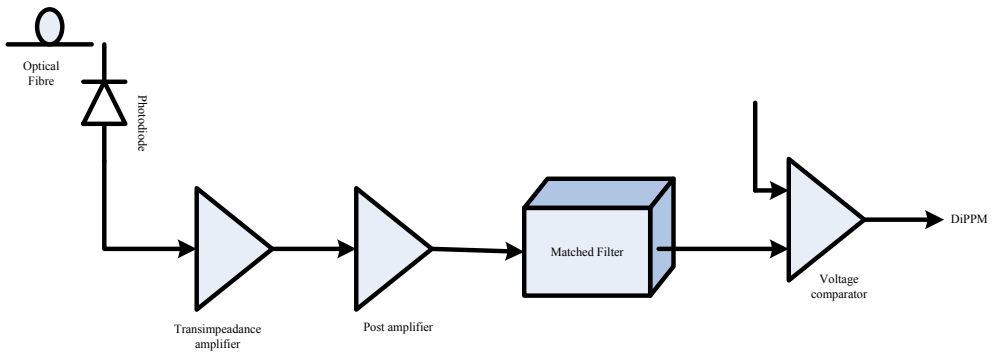


Figure 4.2 Block diagram of DiPPM system receiver

4.2.3. DiPPM Employing RS vs PCM Employing RS

To analyse the PCM employing RS system performance, a 1 GHz bandwidth PIN-BJT receiver having a noise current spectral density of 16x10-24 A2/Hz (double sided) was used. The optical channel used in the simulation had a Gaussian impulse response, as obtained from the graded-index plastic optical fibre (GI-POF). An operating wavelength of 650 nm was taken, corresponding to the first transmission window in POF, and the photo-diode quantum efficiency was considered to be 100%. An algorithm was used to calculate the number of photons per pulse (b) needed to give an error rate of 1 error in 10^9 pulses of the uncoded data.

Let the received pulse shape, HP (t), has the following property:

$$\int_{-\infty}^{\infty} h_p(t) = 1 \tag{4.8}$$

The impulse response of the channel (GI-POF) can be approximated to a Gaussian and thus

$$h_p(t) = \frac{1}{\sqrt{2\pi\alpha^2}} exp\left(-\frac{t^2}{2\alpha^2}\right) \tag{4.9}$$

The pulse variance, α, is linked to the fibre bandwidth by equation (4.6). The error probability can determine from the following equation:

$$P_e = \frac{1}{2} erfc\left(Q/\sqrt{2}\right)^2 \tag{4.10}$$

where

$$Q = \frac{v_{max} + v_{min}}{2\alpha} \tag{4.11}$$

Where, $v_{max} and\ v_{min}$ represent the received signal levels at the output of the detection filter. Simulations were conducted on PCM systems operating with RS code.

4.3. Results

4.3.1. Finding Optimum RS System Parameters

Figure 4.3 shows the PCM code symbolised by the DiPPM signal using different normalised bandwidth. In DiPPM, the shape of the S and R pulses, figure 4.4, will depend on the transmitted pattern. The new pulse shapes must be found using a general DiPPM sequence SxNRyNS.

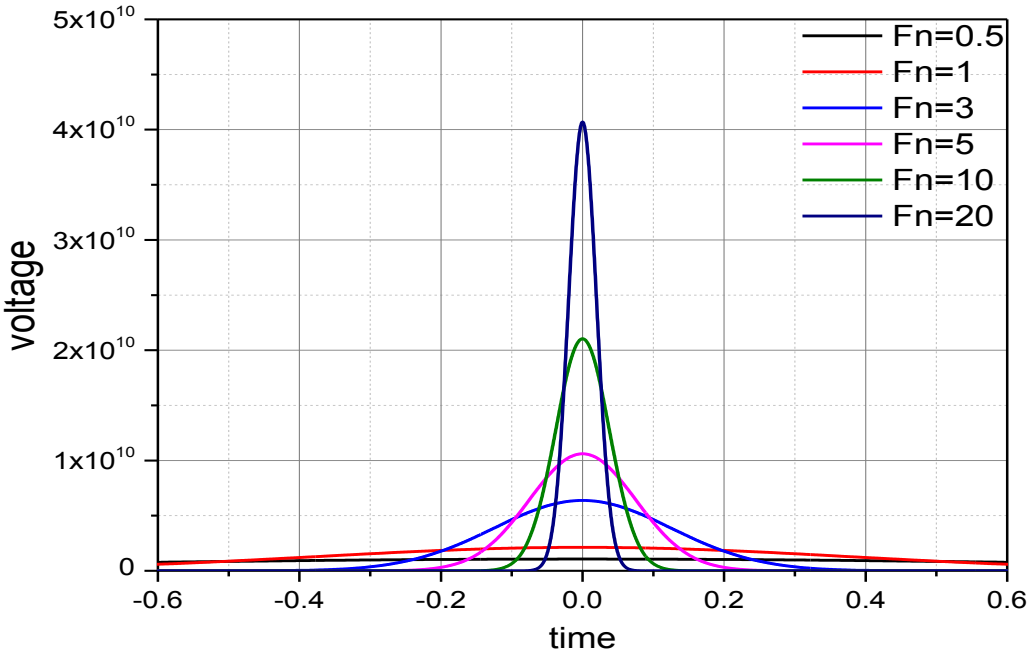


Figure 4.3 The received DiPPM signal

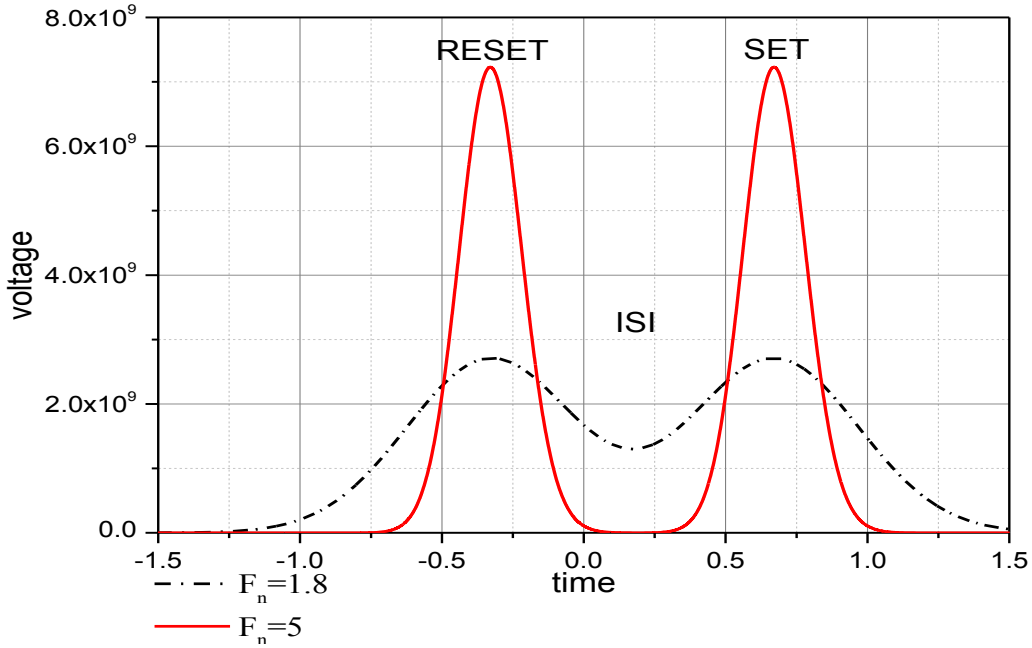


Figure 4.4 DiPPM SET & RESET pulses at two normalised bandwidths

Figure 4.5 and 4.6 show the number of photons in the DiPPM coded system when it works at different code rates using slope, and central detection methods. The data presented in these figures use code word length 2^m, where m=3,4,5,6,7 to compute the number of photons. It should be noticed that the number of photons increases with the increasing in the RS code rates. This is because of the number of data symbols is directly proportional to the RS code rates. Moreover, the results show that the number of photons is directly proportional to the normalised bandwidth when the slope detection method is used. This is because of the slope detection method depends on the received signal shape. Figure 4.7 shows the clear superiority of central detection over slope detection for number of photons.

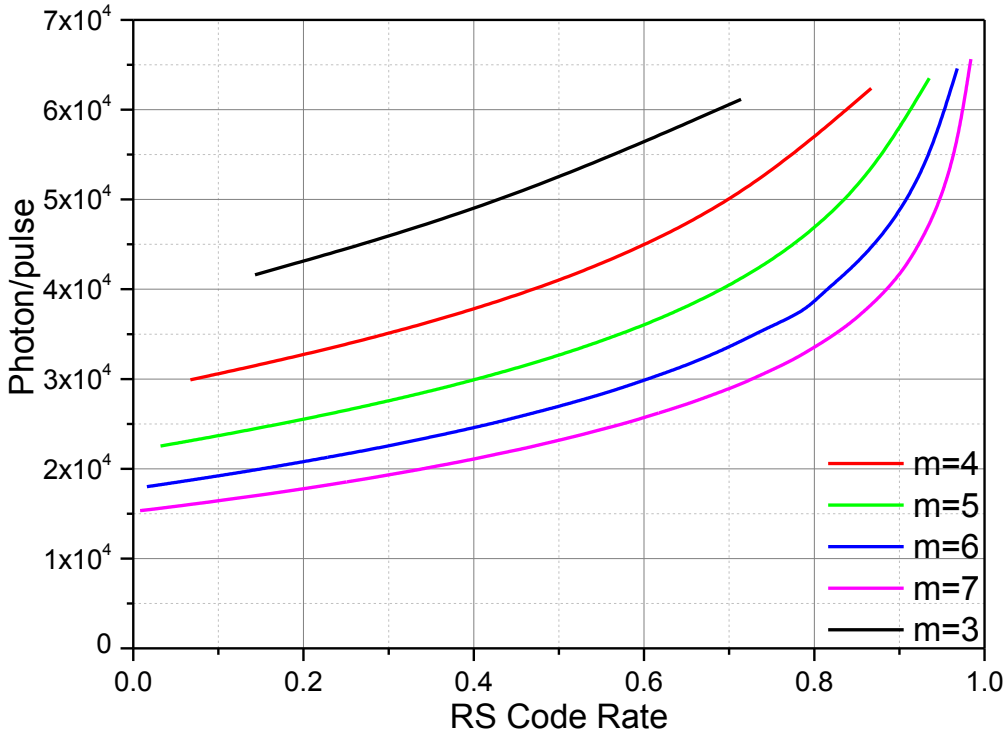


Figure 4.5 Number of photons for the coded DiPPM system function of RS code rate at different RS codeword length using the slope detection method (fn=1.8)

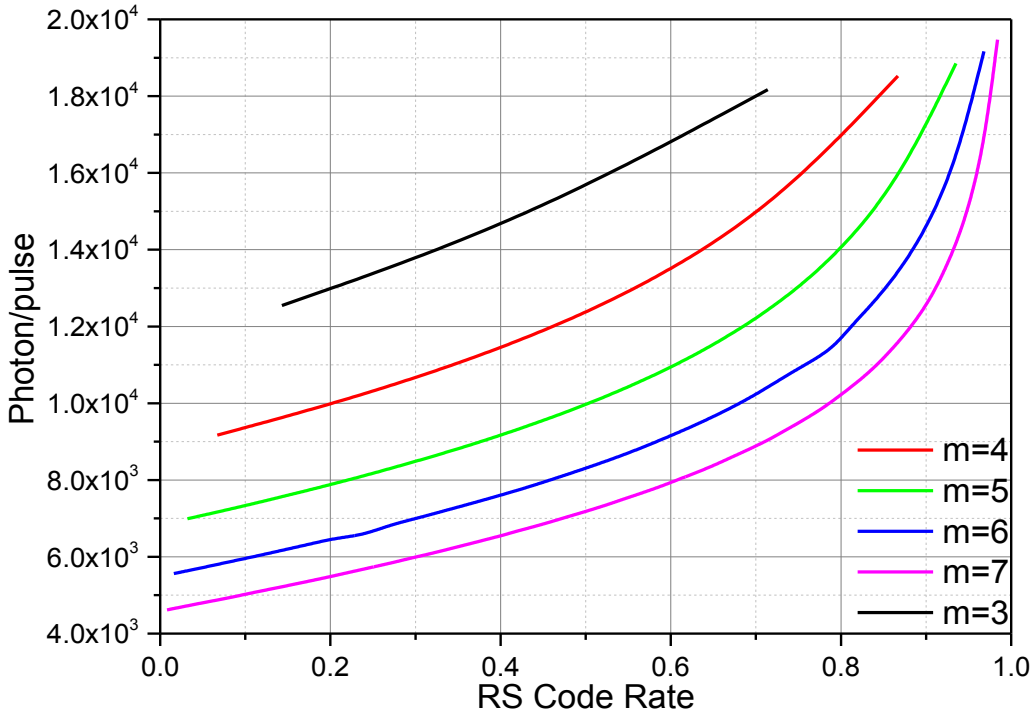


Figure 4.6 Number of photons for the coded DiPPM system function of the RS code rate at different RS codeword length using the central detection method (fn=1.8)

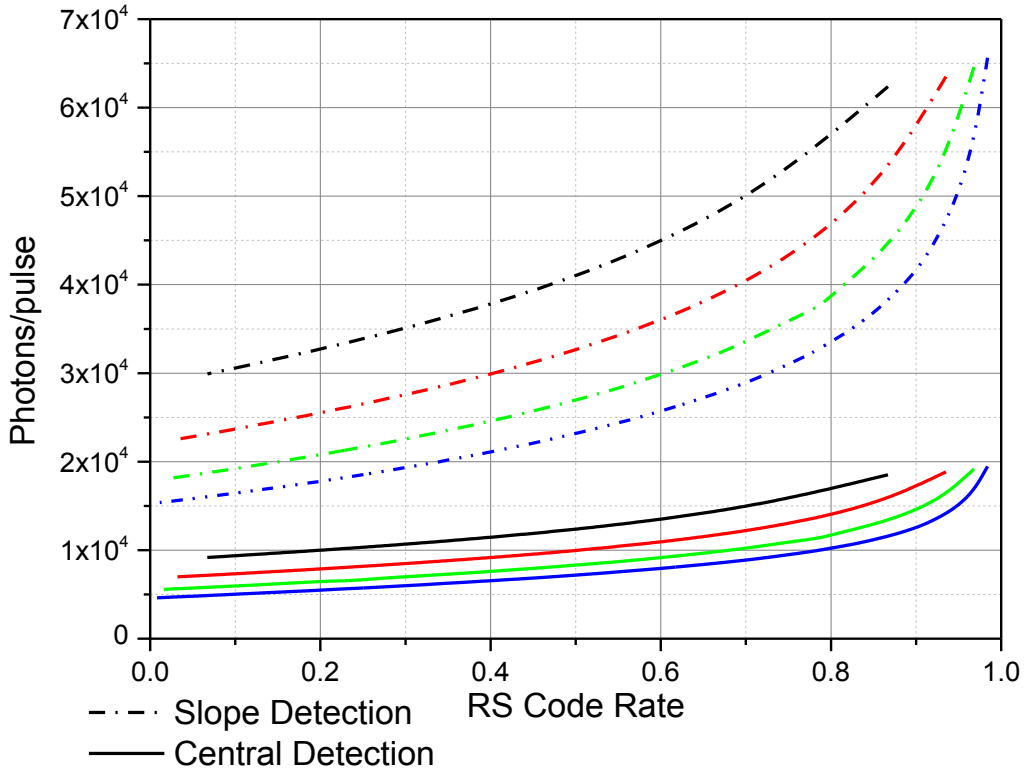


Figure 4.7 Comparison between detection methods in term of the number of photons for coded DiPPM system at different RS codeword length (fn=1.8)

Figure 4.8, and 4.9 clearly showed that there is an optimum code rate approximately 3/4, which is achieved maximum transmission efficiency. When the DiPPM coded system is operating below this optimum, the number of redundant symbols increases and, as predicted by equation (4.3), performance is degraded. Above the optimum coding rate, the number of redundant symbols is decreased, which means the number of correcting symbols is also decreased and this reduces the transmission efficiency. Figure 4.10, shows the outperform of the central detection over slope detection method. The central detection method achieved a lower number of photons in all the normalised bandwidth ranges.

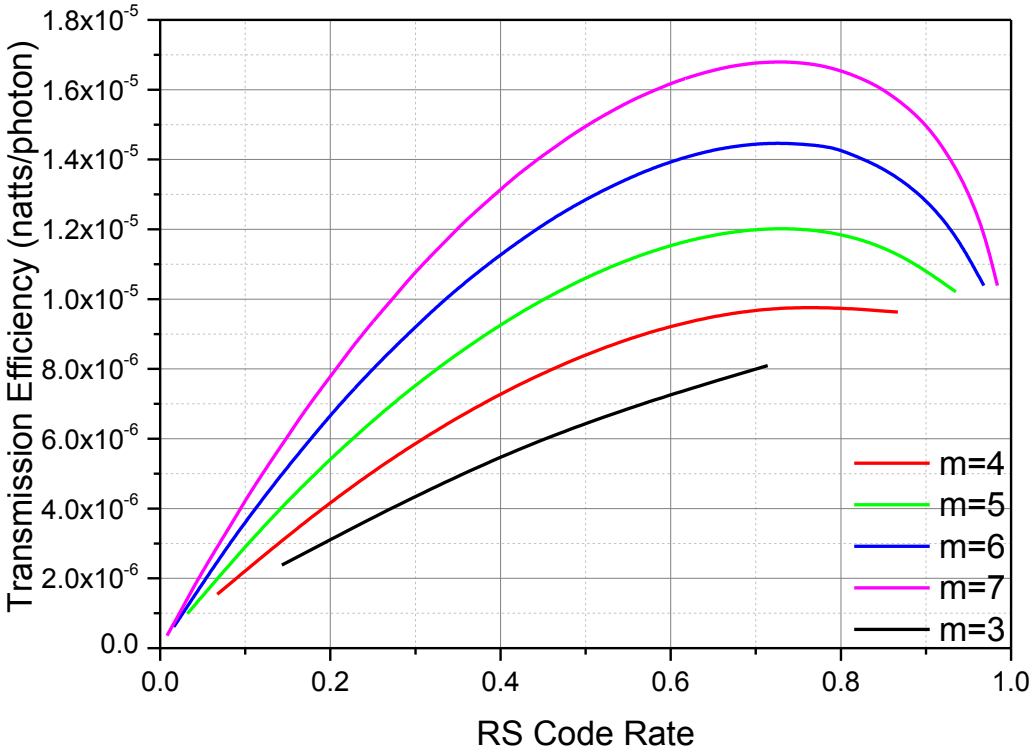


Figure 4.8 Transmission efficiency of the code DiPPM system function of the RS code rate at different RS codeword length using the slope detection method (fn=1.8)

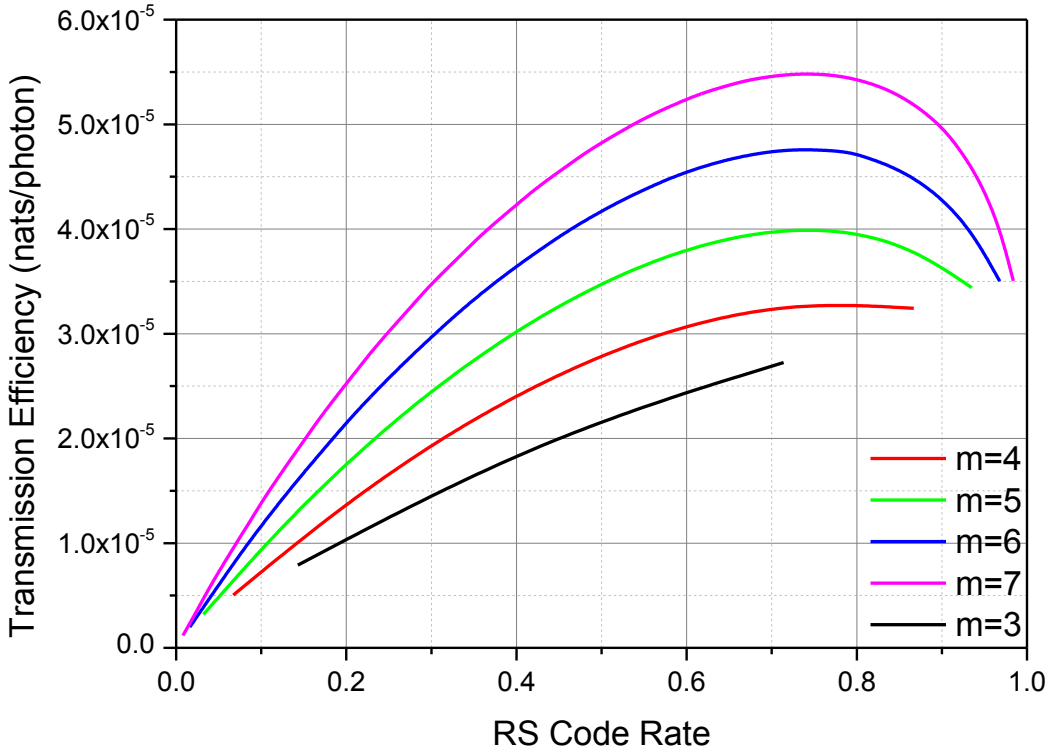


Figure 4.9 Transmission efficiency for coded DiPPM system function of the RS code rate at different RS codeword length using the central detection method (fn=1.8)

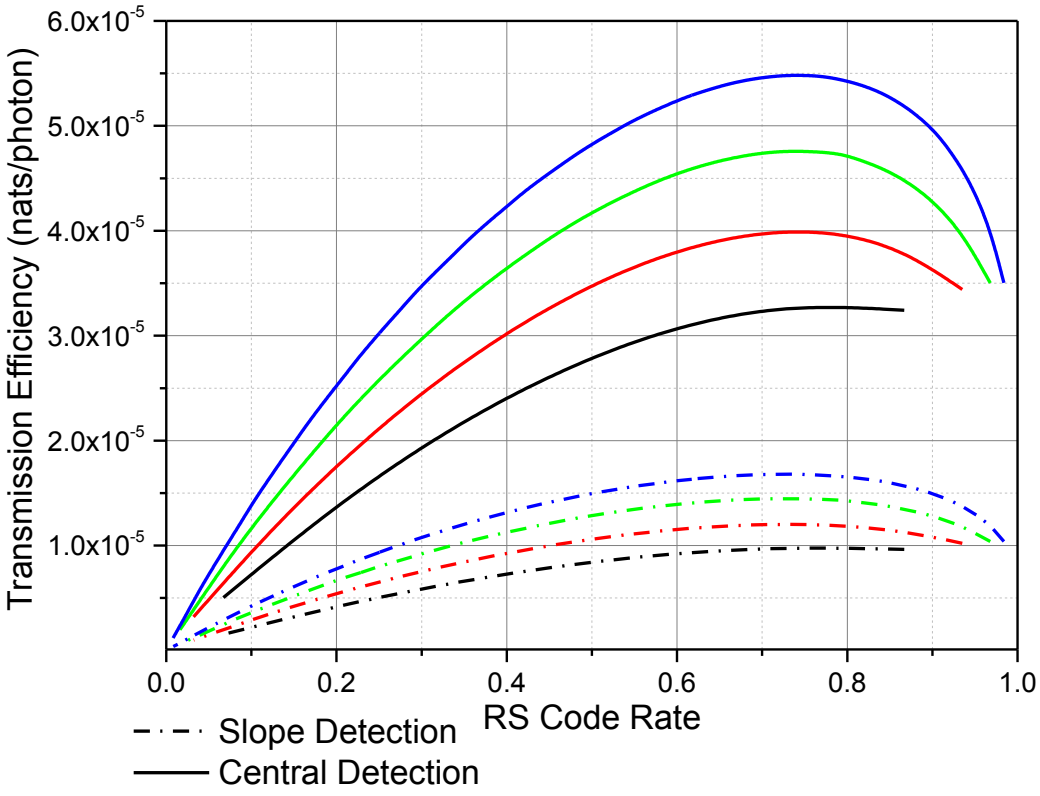


Figure 4.10 Comparison between detection methods in term of the transmission efficiency for coded DiPPM system at different RS codeword length (fn=1.8)

Figure 4.11 compares the uncoded and coded DiPPM system in terms of the number of photons at a different bit error rate by using central and slope detection methods. The RS code length varied from 15-128 symbols per codeword, and at the code rate equal approximately 3/4. The figure 4.12 shows the transmission efficiency of the DiPPM coded system, by using 31 symbols RS code length at different bit error rate. The results confirm that the RS has a slightly same optimum code rate even when the system works at a different bit error rate.

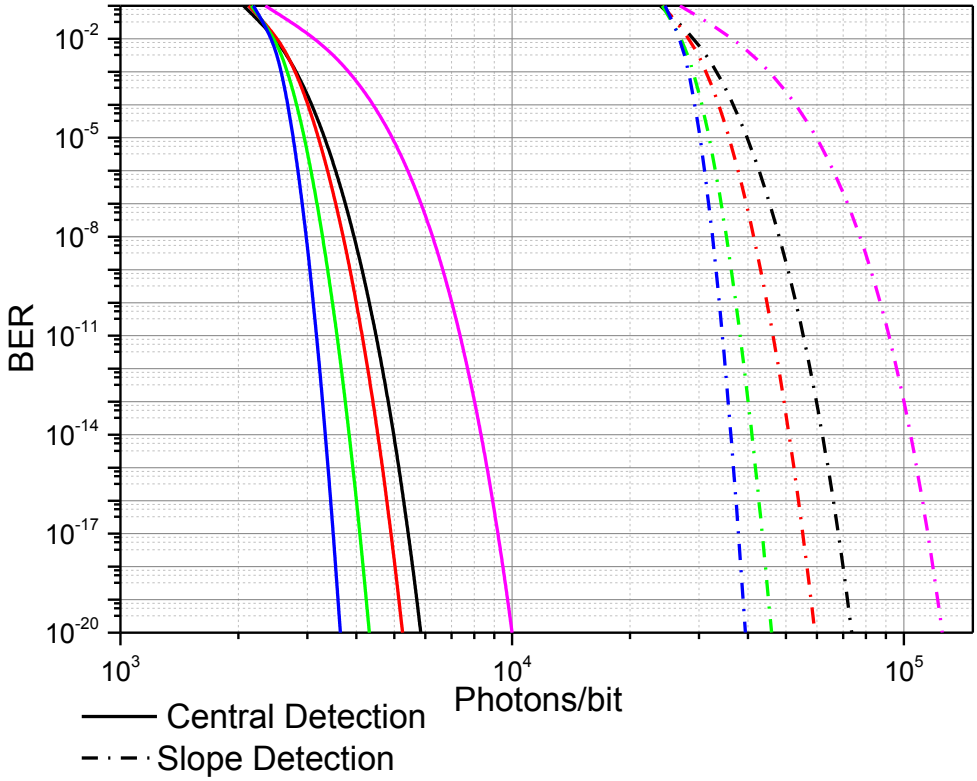


Figure 4.11 Comparison between detection methods in term of the number of photons for coded DiPPM system at different BER and RS codeword length (fn=5)

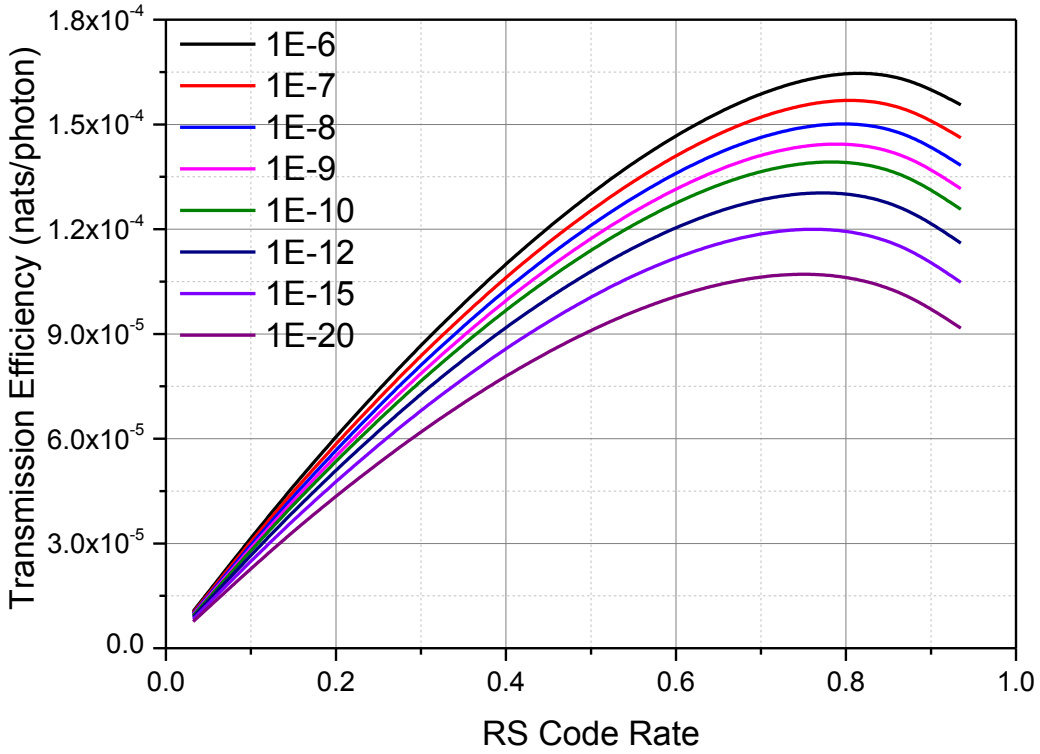


Figure 4.12 Transmission efficiency of the code DiPPM system function of the RS code rate at different BER using the central detection method (fn=5)

4.3.2. DiPPM Employing RS vs DiPPM Employing MLSD

The central detection method has been used to compute the results of this section. Table 4.1 shows the numbers of photons per pulse for DiPPM systems when it is operating with and without MLSD or RS code. The computing of numbers of photons is held with the variance in normalised fibre bandwidth. The starting operating bandwidth is varied from system to another, at a bandwidth below 0.4 times the PCM data rate only the MLSD system can operate which it requires 4×10^5 photons per pulse.

Table 4.1 DiPPM system with and without MLSD or RS

f_n	Uncoded DiPPM	DiPPM with MLSD	DiPPM with RS
0.46	658.4×10^3	40.2×10^3	76.3×10^3
1	95.8×10^3	14.3×10^3	11.4×10^3
1.8	25.6×10^3	10.5×10^3	4.5×10^3
10	4.7×10^3	4.6×10^3	1.4×10^3
100	2.1×10^3	2.1×10^3	0.9×10^3

Fig 4.13 shows that RS code required only 14.3 x 10^3 photons per pulse when it is operating at bandwidth equal or above 0.9 times the PCM data rate. The MLSD system achieves a reduction in the number of photons per pulse when it operates at bandwidth less than 1 normalised. Thus the DiPPM with RS code system outperforms on DiPPM with MLSD system when it operates at a high bandwidth, because the RS system is expanding the operating bandwidth for DiPPM system depending on RS code rate.

Fig 4.14 shows the transmission efficiency as a function of normalised bandwidth for uncoded and coded DiPPM system. The coded DiPPM using RS code realises higher transmission efficiency when its work in a low dispersive channel. This is due to the expansion in bandwidth that the RS code consumes by adding the redundancy symbols.

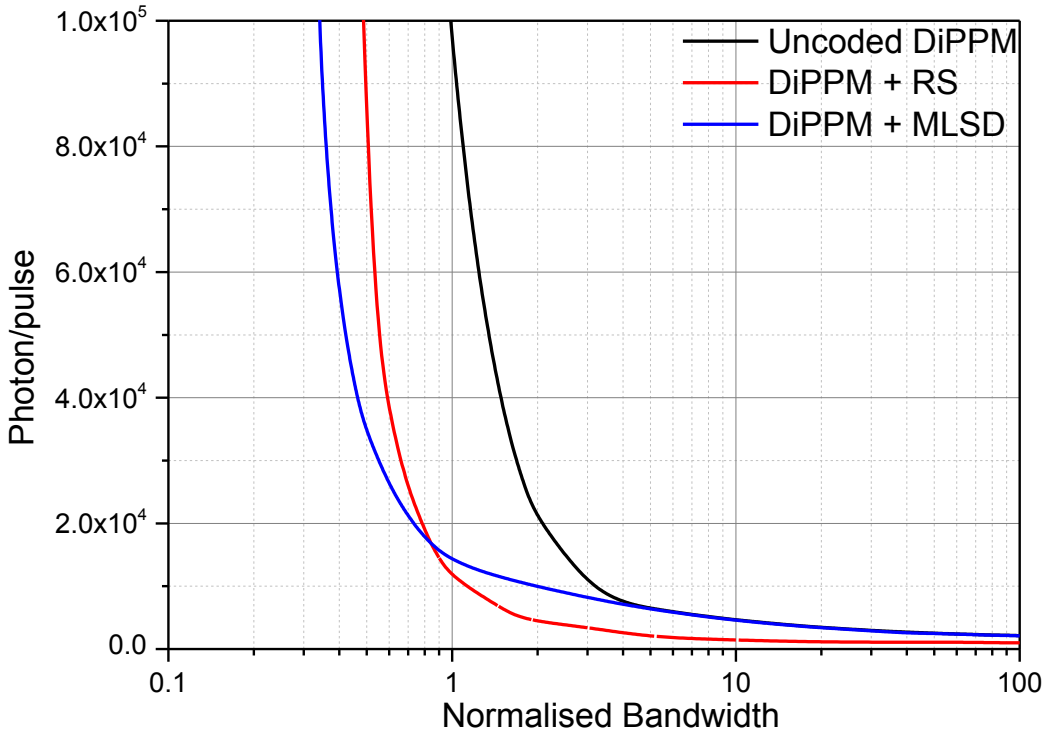


Figure 4.13 Numbers of photons per pulse as a function of normalised bandwidth

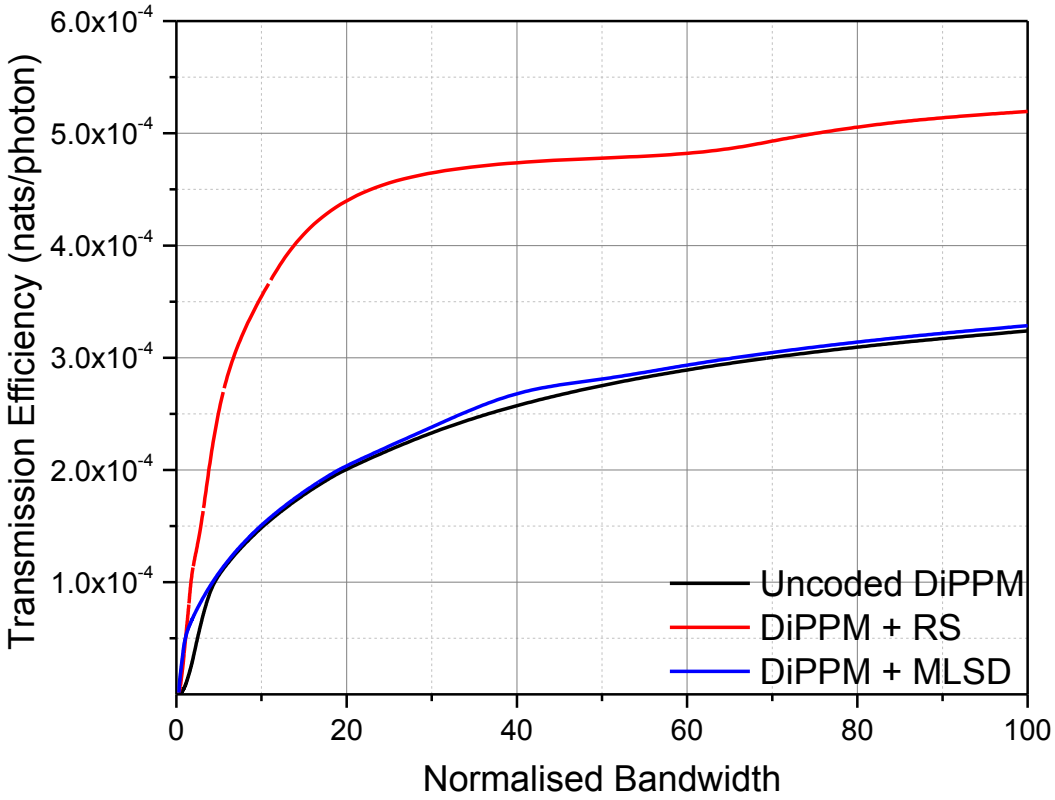


Figure 4.14 Transmission efficiency as a function of normalised bandwidth

Figure 4.15 shows the relation of the probability of error for using the wrong slot, erasure, and false alarm error probability. Wrong slot error is the dominant error in low bandwidth. When the probability of wrong slot error is reduced, the other two probabilities are increased to maintain the system performance by reducing the pulse energy. For this reason, an improvement can be seen in the transmission efficiency as the fibre bandwidth increased. This improvement continues until the wrong slot error probability is negligible. In the slope detection method the improvement lasts until 1.8 times the normalised bandwidth and then the transmission efficiency starts decreasing, while in the central detection method the improvement continues with the continuing of increasing bandwidth.

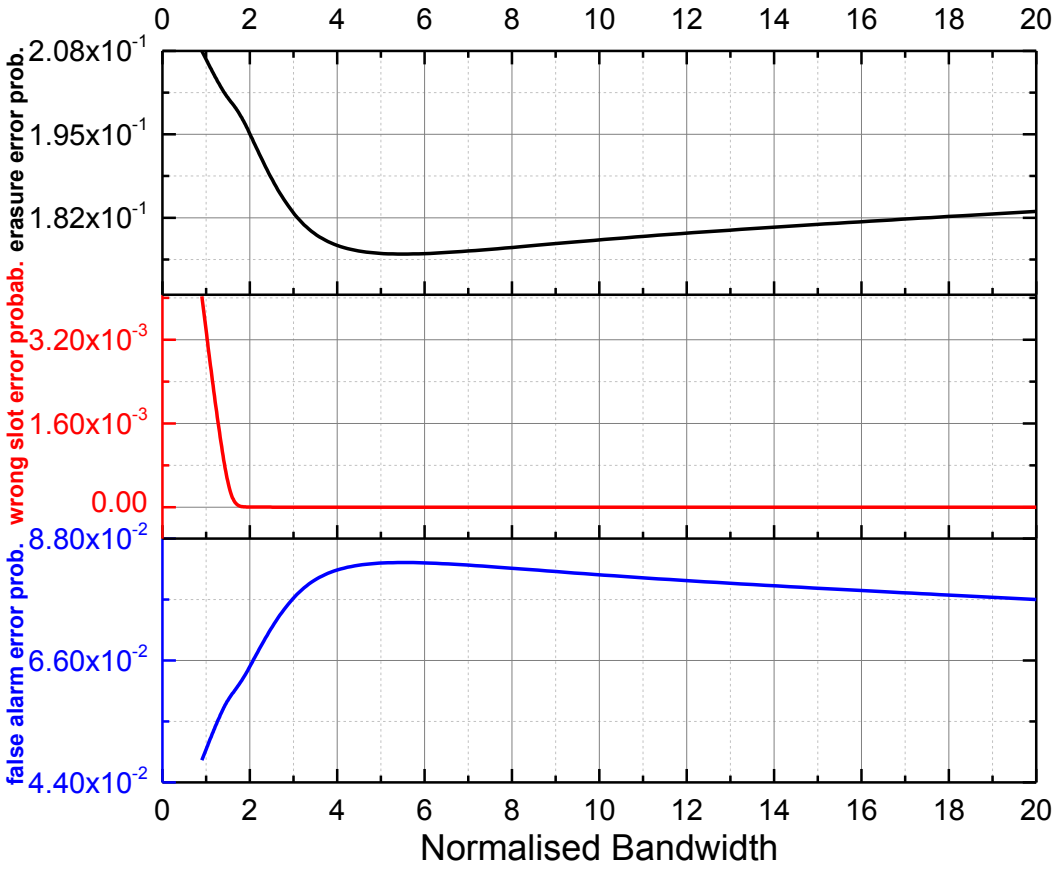


Figure 4.15 Erasure, False Alarm, and Wrong Slot Error Probabilities for the coded DiPPM system

4.3.3. PCM Employing RS

Figure 4.16 depicts the received PCM pulses, (1s and 0s), levels on the output of the detection filter. Figure 4.17 compares the number of photons per pulse for many normalise bandwidths f_n at a different RS code rate. From this figure, it can be seen that as the RS code rate is increased, the number of photons required per pulse will also increase for a particular bandwidth. Figure 4.18 clarifies that the PCM employing RS code system has approximately the same optimum code rate of DiPPM employing RS code system which is about 3/4. The transmission efficiency for the PCM employing RS code system is computed through equation (4.12).

$$\rho = r\frac{\ln 4}{b} \qquad \left(\frac{nats}{photon}\right) \tag{4.12}$$

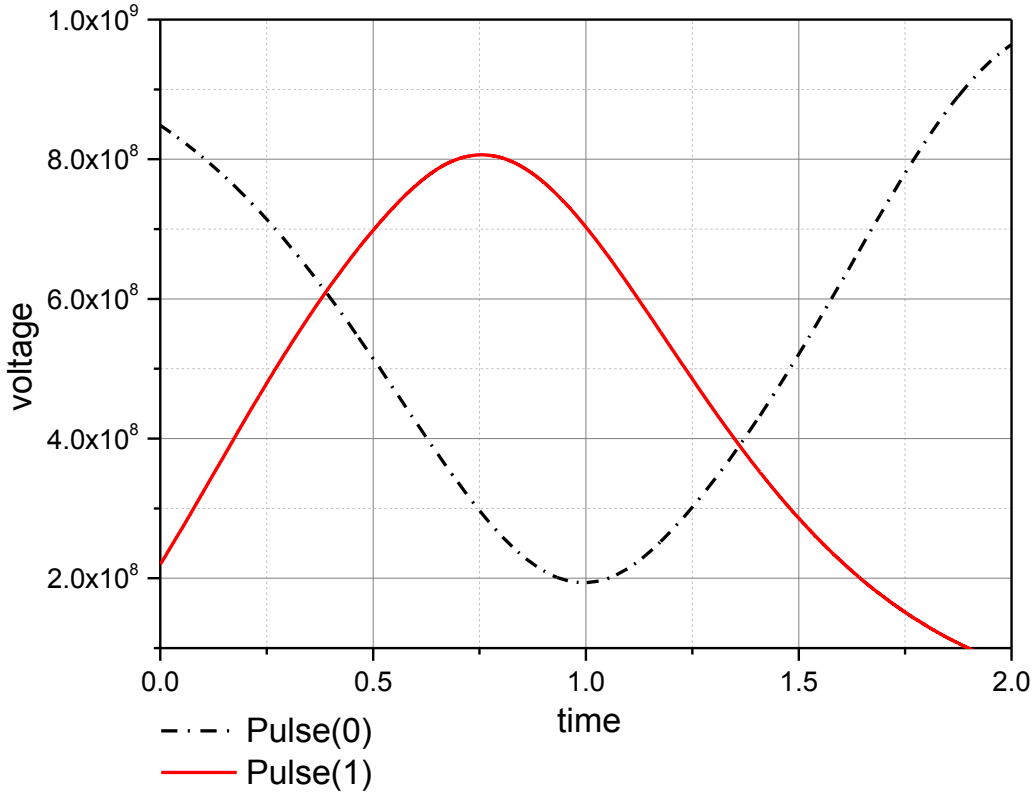


Figure 4.16 The received DiPPM signal

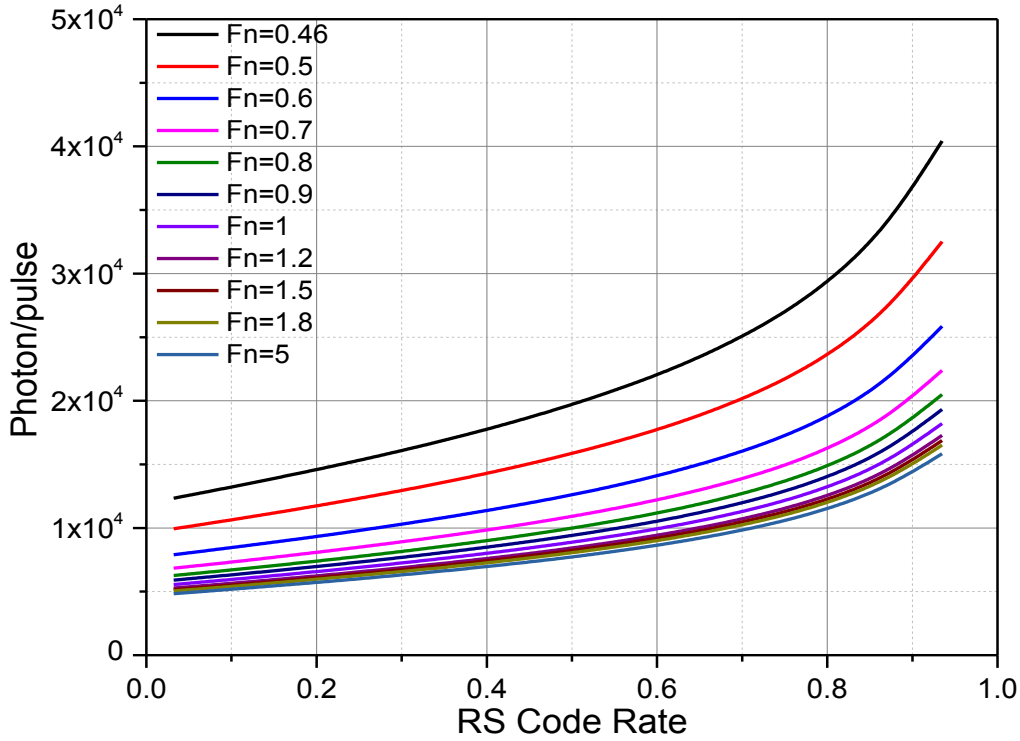


Figure 4.17 Number of photons for the coded PCM system as function of RS code rate at different normalised bandwidth (BER=1.10^{-9})

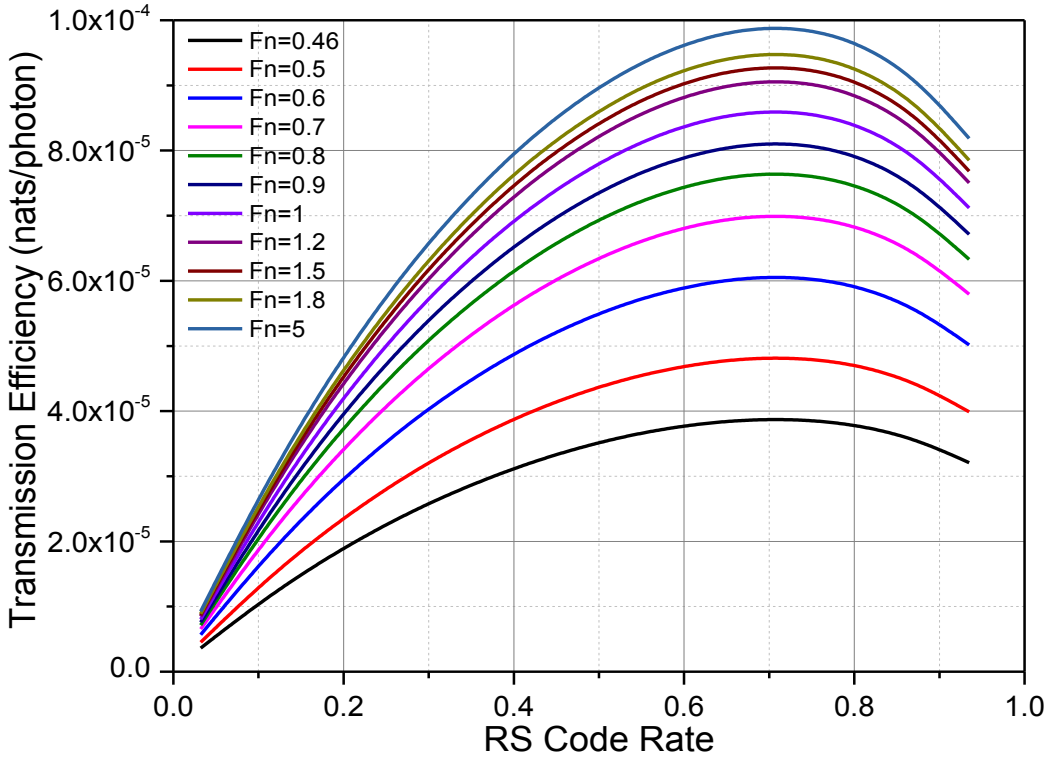


Figure 4.18 Transmission efficiency for the coded PCM system as function of RS code rate at different normalised bandwidth (BER=1.10-9)

Figure 4.19 gives a picture of the system transmission efficiency with the change of the RS code rate at different codeword length. Figures 4.19 & 4.20 illustrate that the optimum RS code rate does not change with the codeword length, or system bit error rate (BER), and the system performance improved with the increasing the RS codeword length and decreasing the BER. However, the RS system design complication is proportional to the codeword length. Figure 4.21 shows that the RS code improves the signal to noise ratio factor ,Q, when it is added to PCM system. So, Q value of 2.8 for coded PCM comparing with 6 for uncoded PCM in an error rate of 1 bit in 10^{-9}. Figure 4.22 confirms the outperforming of the DiPPM system over PCM at different BER.

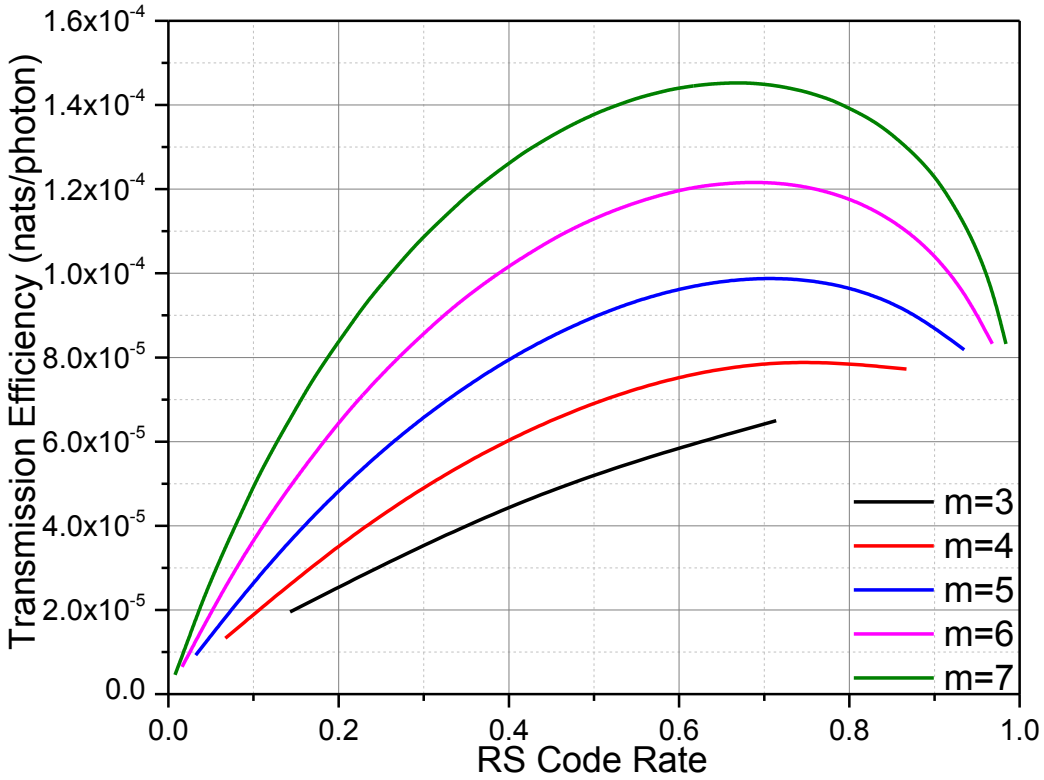


Figure 4.19 Transmission efficiency for coded PCM system as a function of RS code rate at different RS codeword length using the central detection method (fn=5)

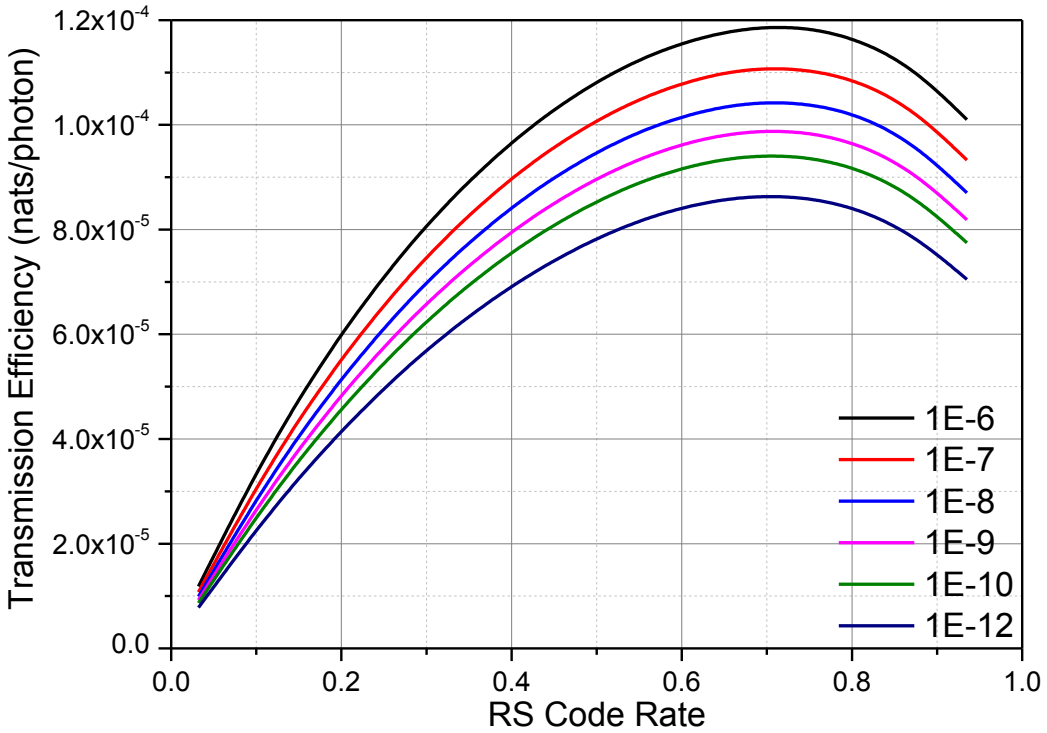


Figure 4.20 Transmission efficiency for coded PCM system as a function of RS code rate at a different BER (fn=5)

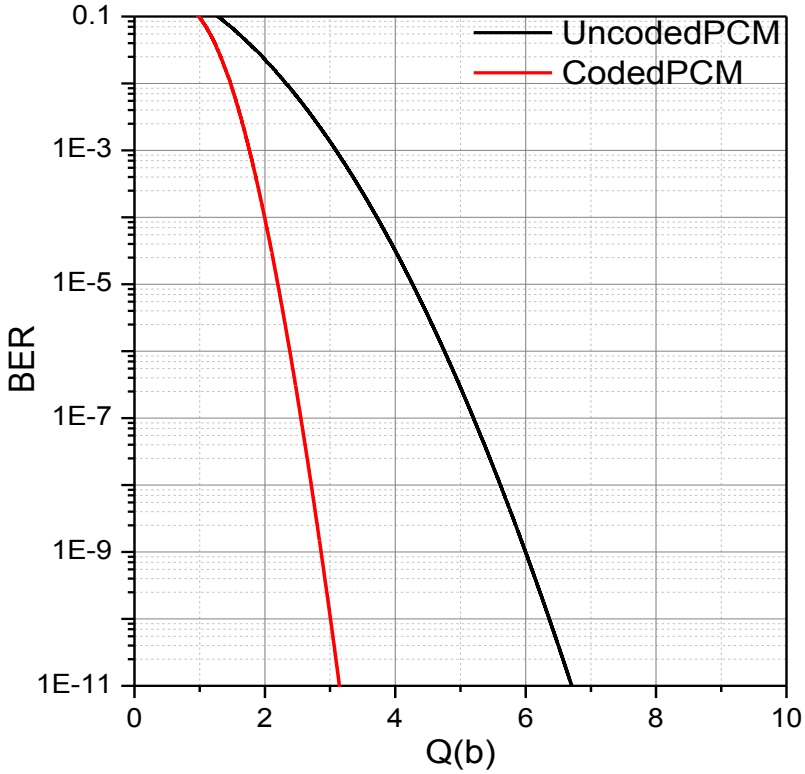


Figure 4.21 BER against signal-to-noise ratio parameter, Q, at normalise BW= 100

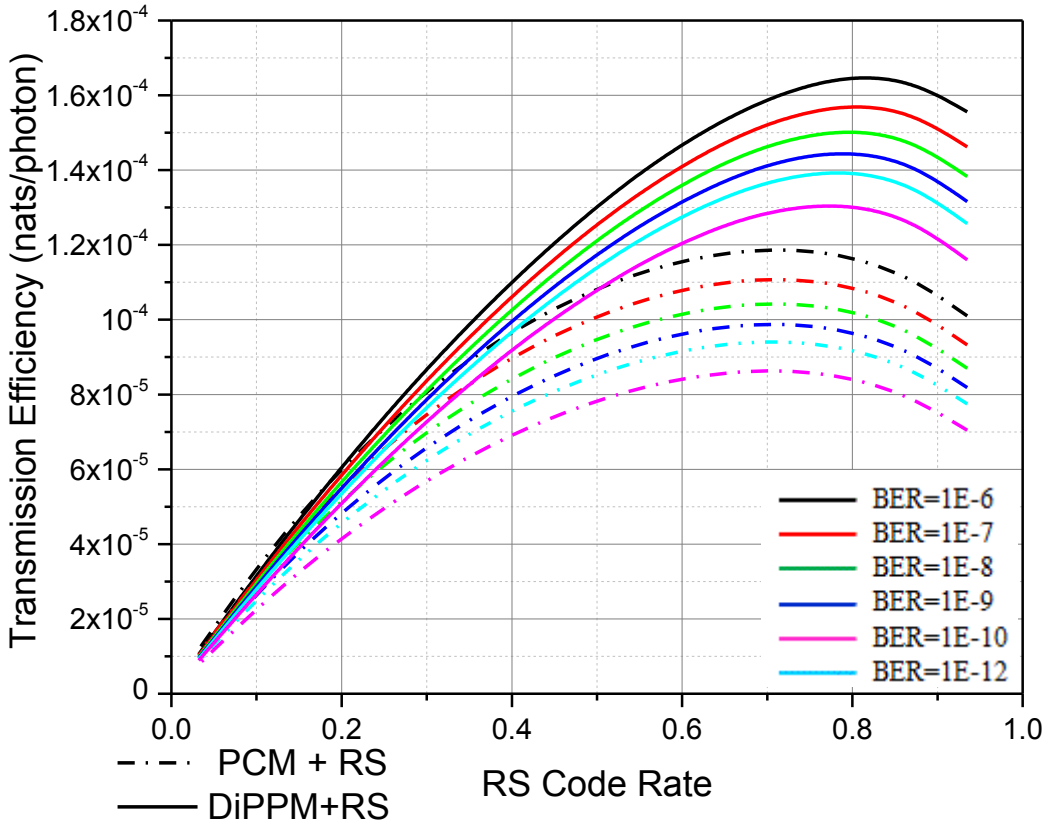


Figure 4.22 comparison between coded DiPPM and coded PCM in term of transmission efficiency at different BER, normalised BW=5

4.4. Summary

This chapter has examined the use of Reed Solomon (RS) codes with Dicode Pulse Position Modulation (DiPPM) in terms of transmission efficiency, bandwidth expansion and number of photons required per pulse. The slope detection and central detection methods have been used to detect the received signal. The simulation results show that the use of RS codes can greatly increase the transmission efficiency of DiPPM by reducing the number of photons. The outcomes have demonstrated that the DiPPM coded system offers a 5.12dB improvement over the uncoded system when it operates at the optimum code rate of (3/4). The system performance improves with the increasing the RS codeword and decreasing the BER. Moreover, the results clarify that the RS optimum code rate does not related to the varying in system parameters or the type of coding scheme.

MATLAB SIMULATION FOR THE DICODE PULSE POSITION MODULATION SYSTEM WITH REED SOLOMON CODE

5.1. Introduction

In this thesis, Matlab software has been used as a bridge to connect between the system simulation and implementation. Matlab software is engaged to simulate the system, the DiPPM with the RS code, figure 4.1. The simulation was developed through four versions. Although there was a Matlab simulation for the DiPPM system (Charitopoulos, 2009), a new version of DiPPM (coder & decoder) simulation has been presented in this chapter. The reason for that is to produce a DiPPM system working with a random input sequence and harmonic with RS system. In the second version, the RS system has been employed with the DiPPM system in order to prevent the errors that affect the system. Then, a noise is injected into the channel to generate the errors. In the fourth version, a PCM binary sequence was replaced by a picture's data to analyse the transmission performance of the system.

5.2. DiPPM System Simulation

The Matlab software has been used to simulate the DiPPM system, Appendix (2) section (10.2.1). The system design was dependent on the DiPPM truth table, table 3.1. The DiPPM system programme contains two main sections, DiPPM coder and DiPPM decoder. The first step is a clock and a random binary PCM signal generating. The generated PCM signal is changing every running of the simulation to produce a different binary PCM signal. Thus, different DiPPM pulses are being shaped.

The second step is calling the DiPPM coder subroutine. The DiPPM coder subroutine was used to create the DiPPM signal (SET & RESET) from the binary PCM signal. Each change from zero to one in PCM sequence gives SET in DiPPM signal, and the alternate from one to zero in PCM sequence produces a RESET pulse in DiPPM. No pulse IS generated in the DiPPM signal when the PCM sequence does not change state.

The third step in this programme was used to regenerate the original PCM sequence from the DiPPM sequence (DiPPM decoder). The programme is going to produce a binary

one in PCM sequence when it receives a SET pulse, and it continues until a RESET pulse is received to produce a binary zero.

The fourth step of the programme is employed to change the binary sequence (one & zero) to pulse shape. Plots output for the DiPPM coder and decoder system were set in the last part of the program. Figure 5.1 & 5.2, shows the DiPPM system results for two different PRBS PCM sequences. Each run simulation produces four line output plot, clock sequence in the first line, then the PCM sequence and DiPPM and Decoded PCM sequence are coming respectively. It is clear from the figure that the system is working as the DiPPM theory mentioned, chapter three.

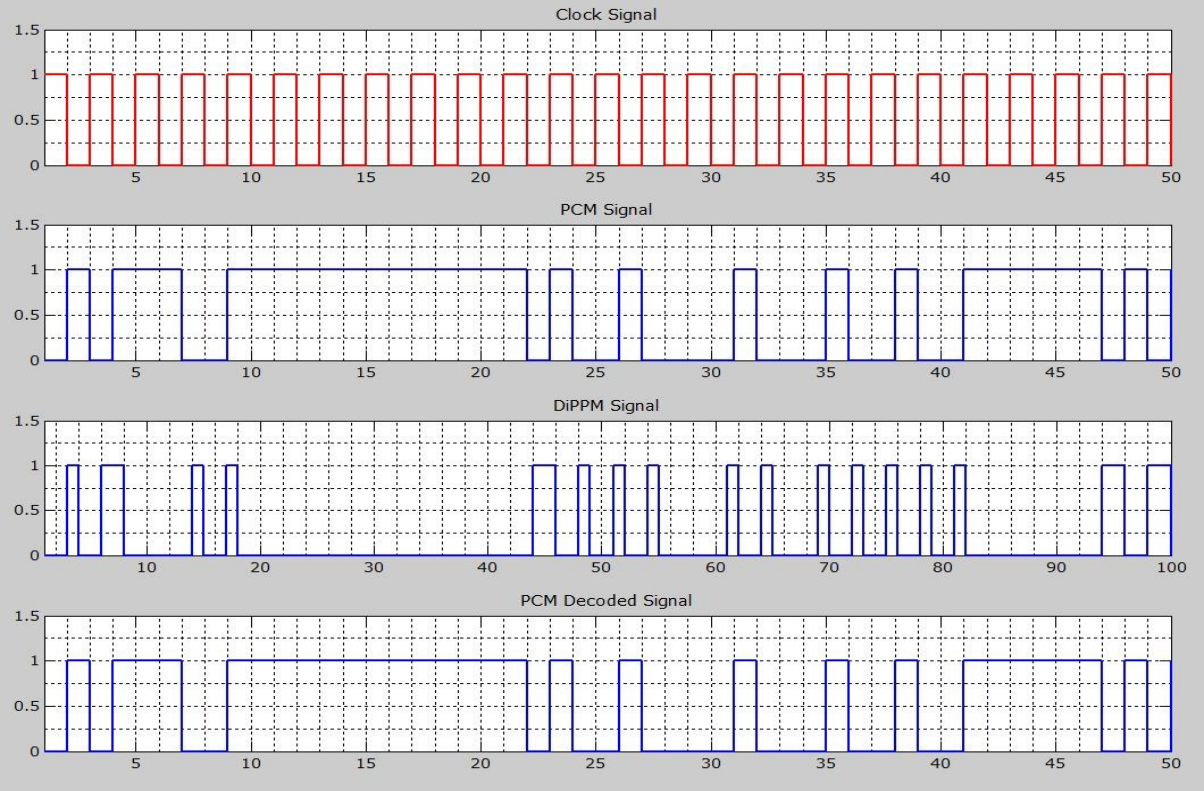


Figure 5.1 DiPPM System for a different PCM Sequence

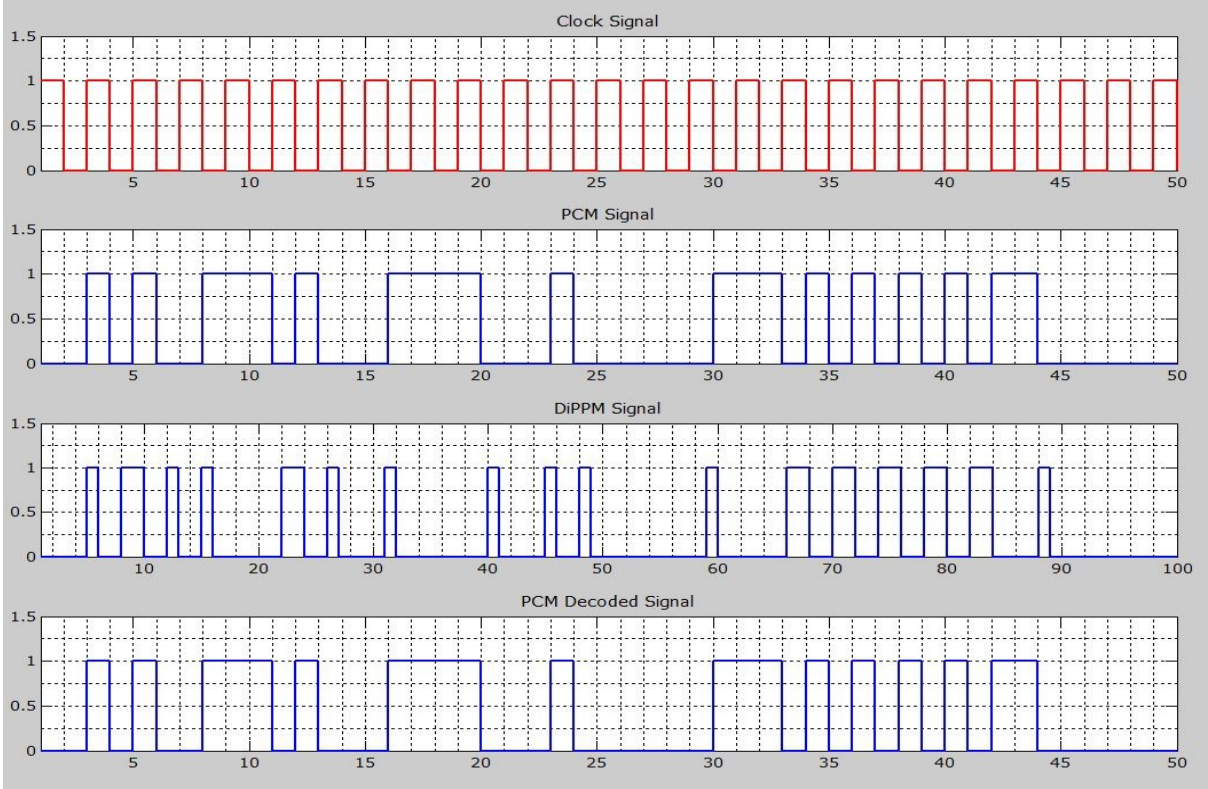


Figure 5.2 DiPPM System for a different PCM Sequence

5.3. DiPPM With RS Code System Simulation

In this programme version, the RS code system has been added to the previous DiPPM system, Appendix (2) section (10.2.4). The MathWorks team has produced two functions to simulate the RS code system:-

$$coder = rsenc(msg, n, k, genpoly)$$

$$decoded = rsdec(code, n, k, genpoly)$$

The first function is for RS encoder and the second function for RS decoder. The encoder function encodes the message in (*msg*) using an [n,k] Reed Solomon code and specifies the generator polynomial (*genpoly*) for the code. The message is a Galois array of symbols having *m* bits each. Each *K* element row of *MSG* represents a message word, where the leftmost symbol is the most significant symbol. *N* is at most 2^m*-1*.

The generator polynomial is a Galois row vector that lists the coefficients, in order of descending powers, of the generator polynomial. The generator polynomial must have degree *n-k*, and n-k must be an even integer. The output genpoly is a Galois row vector

that represents the coefficients of the generator polynomial in order of descending powers. The narrow-sense generator polynomial is

$$\left(X - Alpha^{1}\right)\left(X - Alpha^{2}\right)...\left(X - Alpha^{2t}\right)$$

where:

Alpha represents a root of the default primitive polynomial for the field *GF(*n+1*)*, and *t* represents the code's error-correction capability, *(n-k)/2*.

The decoded function attempts to decode the received signal in `code` using an *[*n*,*k*]* Reed-Solomon decoding process with the narrow-sense generator polynomial. `Code` is a Galois array of symbols having *m* bits each. Each `n`-*element* row of `code` represents a corrupted systematic codeword, where the parity symbols are at the end, and the leftmost symbol is the most significant symbol. `n` is at most 2^m*-1*. If `n` is not exactly 2^m*-1*, `RS decoder` assumes that `code` is a corrupted version of a shortened code.

In the Galois array `decoded`, each row represents the attempt at decoding the corresponding row in `code`. Decoding *failure* occurs if `the RS decoder` detects more than *(n-k)/2* errors in a row of `code`. In this case, `the RS decoder` forms the corresponding row of `decoded` by merely removing *n-k* symbols from the end of the row of code.

The Matlab code for DiPPM with the RS system, Appendix (2) section (10.2.4), contains three main parts, transmitter side, channel, and the receiver side. The first step in transmitter is generating the PRBS PCM sequence (integer message generator). Each codeword encloses 2^m*-1* symbols, for our design *m=5* and the message *k=23* symbols (see chapter four). The next step is using the RS coder function to encode the message. After that the output of the RS coder was fed to the DiPPM coder. A subroutine, Appendix (2) section (10.2.5), has been programmed to convert from a Galois array to a decimal array. Then the decimal array has been converted to a binary array using the function below:-

$$dec2bin\left(RS\_code\_dec\right)$$

The final step in the transmitter side is calling the DiPPM coder subroutine through the function shown below:-

$$DiPPM\_seq = DiPPM\_Encoder\_B\left(RS\_code\_bin\_3\right)$$

Figure 5.3, shows the system transmitter side results. The clock is shown in the first line, and the PRBS PCM sequence displayed in the second line. The RS coded signal shown in the third line, while the redundancy bits (*n-k*) shown in the fourth line. The final line presented the DiPPM (SET & RESET) sequence.

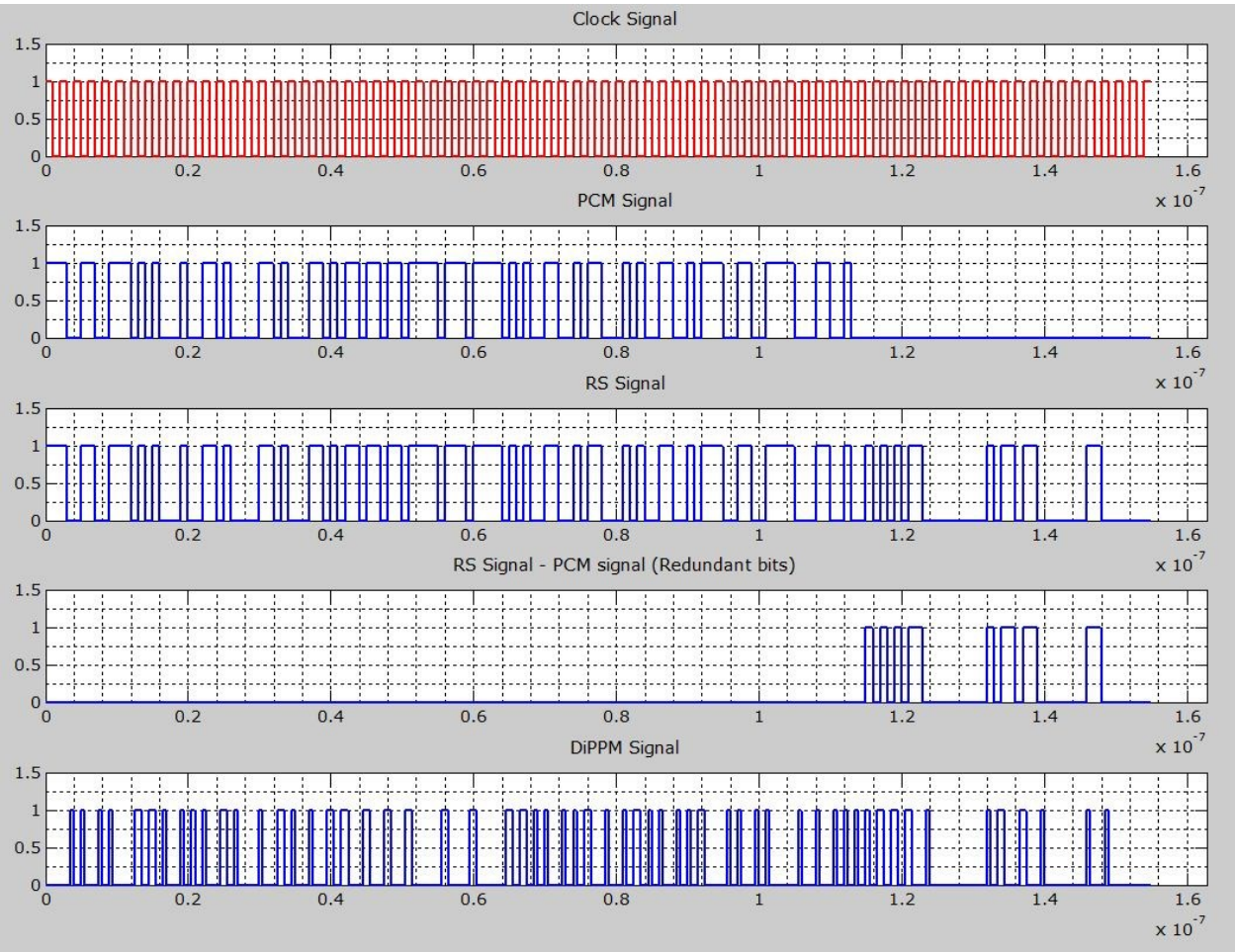


Figure 5.3 DiPPM with RS code transmitter output waveform

The output of the transmitter passes through the channel, error's symbols were injected into the transmitted codeword. The DiPPM decoder receives the transmitted codeword, in order to convert the codeword to its original scheme, PCM. This will be done through the function that calling the DiPPM subroutine.

$$DiPPM\_decoded\_seq = DiPPM\_Decoder\_B(DiPPM\_seq)$$

The next step is converting the output of DiPPM from binary form to decimal form, and then to Galois array form in order to make it in a form proper as input to RS decoder. Finally, the RS decoder is going to deal with the received codeword to extract the original message and fix any error or erasure happened via a transmission operation.

Figure 5.4, shows the output of the receiver side. The output of the DiPPM decoder is displayed in the first line. The second line represents the error number, zero in this case. The third line shows the output of RS decoder, the last line is zero when the process of decoding is successful.

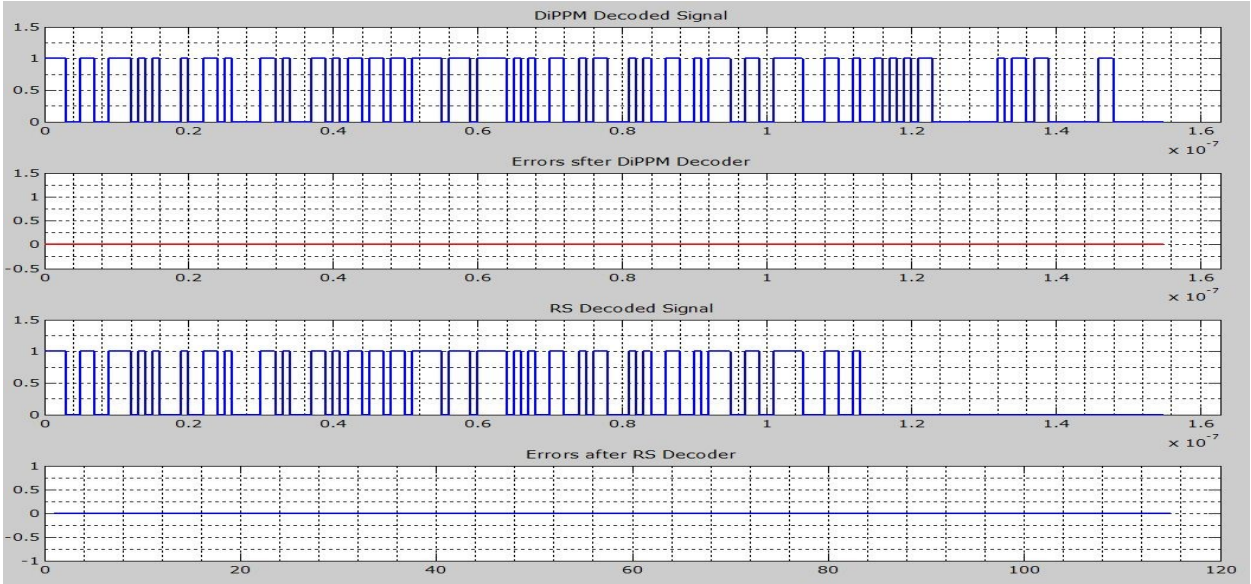


Figure 5.4 DiPPM with RS code receiver output waveform

The software has been run many times, each time the number of errors that inject to the channel is changed (0,1,2,3,4,….,8). Figures 5.5 and 5.6 display the output results for transmitter and receiver system with five bits error. It is clear that the system successfully corrected these errors and produced the original message. Figures 5.7 and 5.8 show the output results for transmitter and receiver system with symbol errors greater than four samples errors. The system fails to produce the original message because it is out of limit.

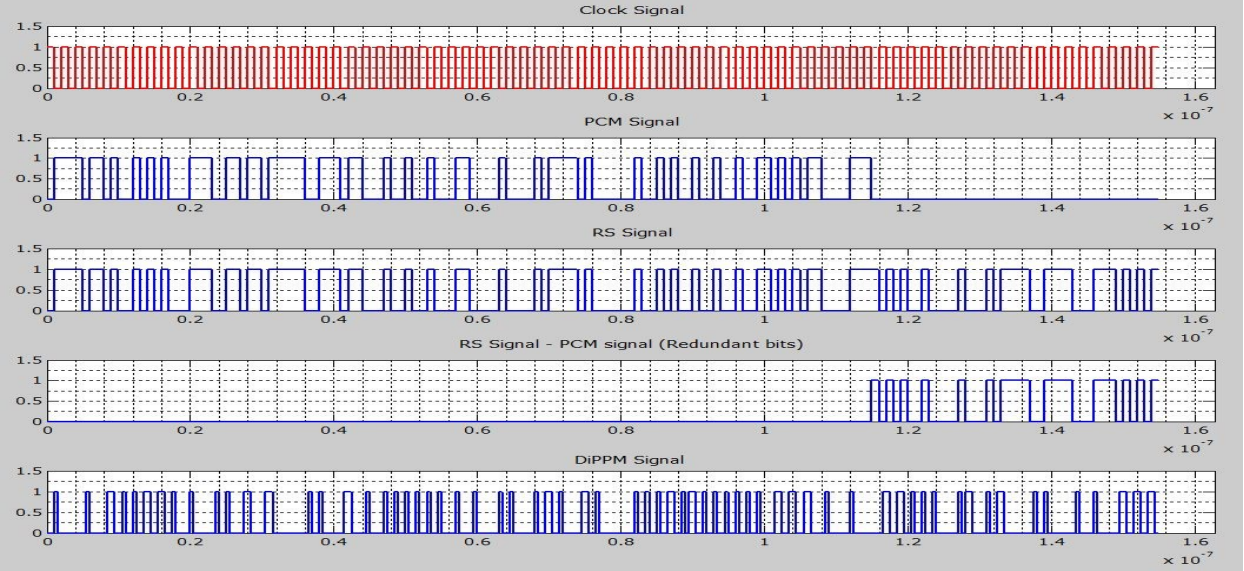


Figure 5.5 DiPPM with RS code Tx output waveform

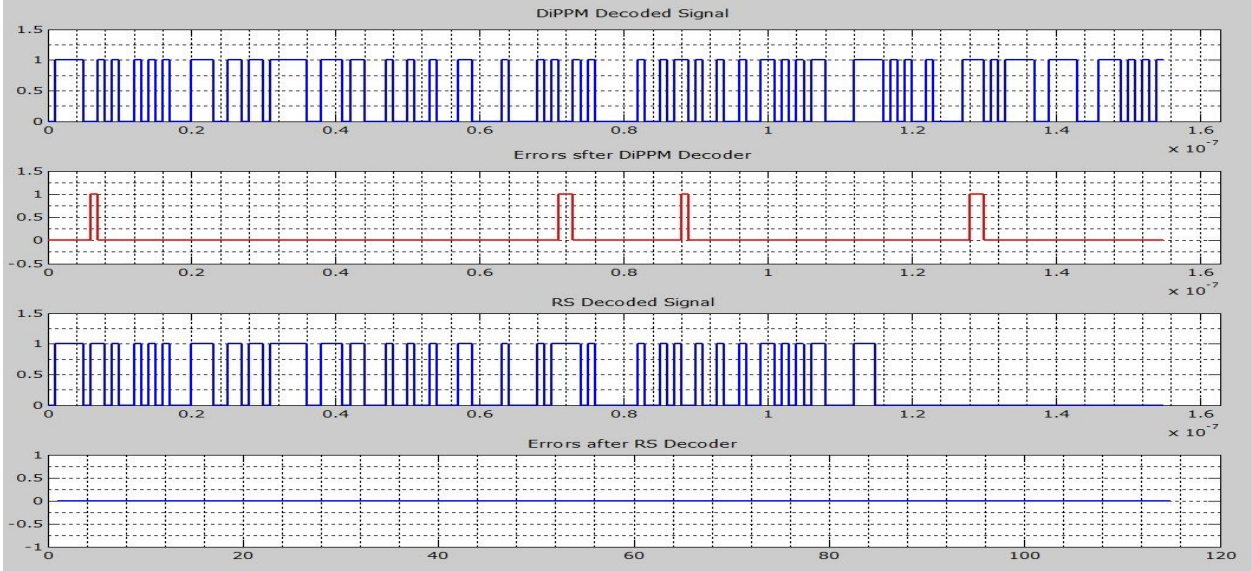


Figure 5.6 DiPPM with RS code Rx output waveform

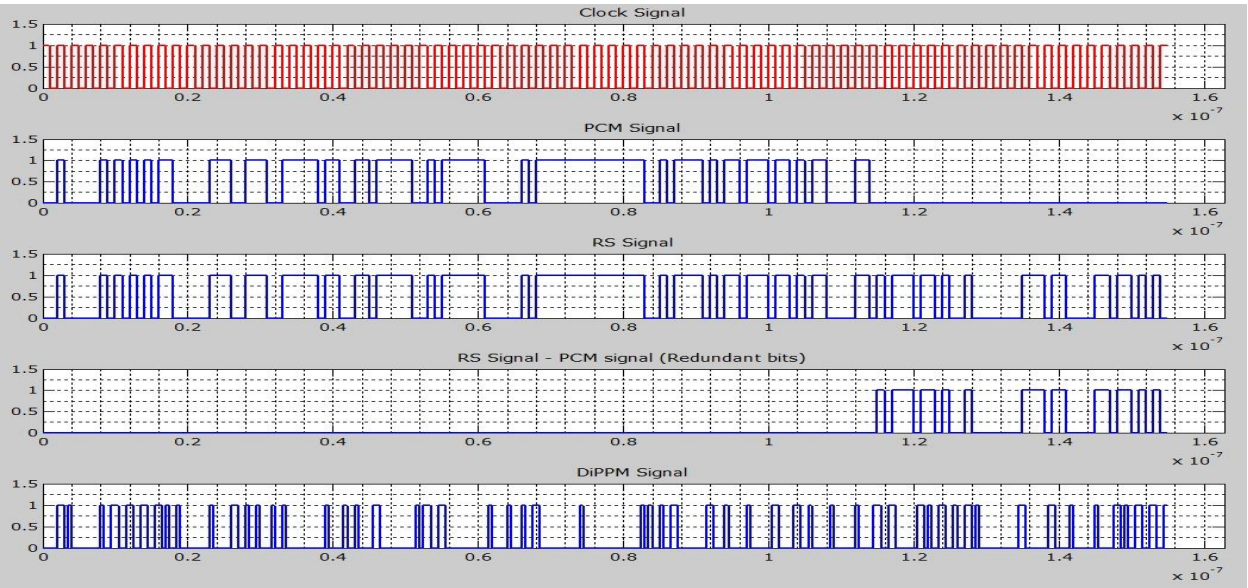


Figure 5.7 DiPPM with RS code Tx output waveform

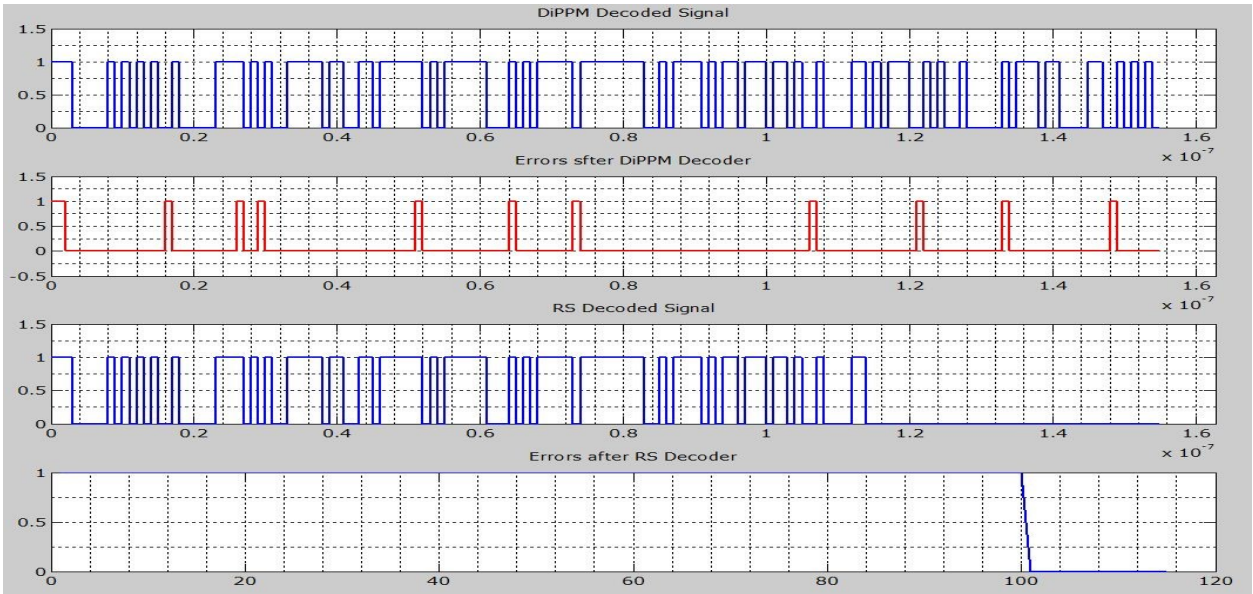


Figure 5.8 DiPPM with RS code Rx output waveform

5.4. DiPPM WITH RS SYSTEM IN AWGN CHANNEL

In this programme version, Additive White Gaussian Noise (AWGN) is added into the channel to affect on the transmitted DiPPM pulses. The MathWorks team has produced function to add AWGN.

$$y = awgn(x, snr)$$

where:

"y" is the signal after adding the AWGN, "x" is the transmited signal, and "snr" is the signl to noise ratio.

The detection errors that the DiPPM suffers from are going to appear in the received signal due to channel noise. The number of errors depends on the SNR, errors increase when SNR decrease and vise versa. The software has been run many times, each time the number of senior is changed to generate a different error number. The simulation results show that the system succeeds to decode the original data when the SNR is equal or above 12dB. Figures 5.10, 13, 16, 19 display the Tx And Rx signals for different snr values.

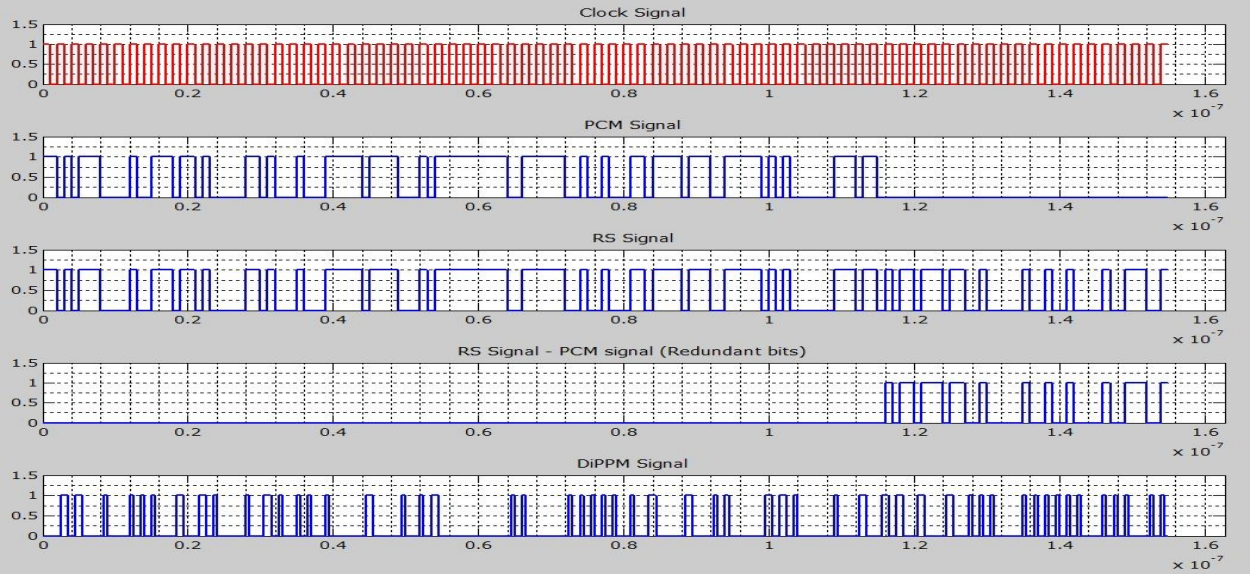


Figure 5.9 DiPPM with RS code Tx output waveform

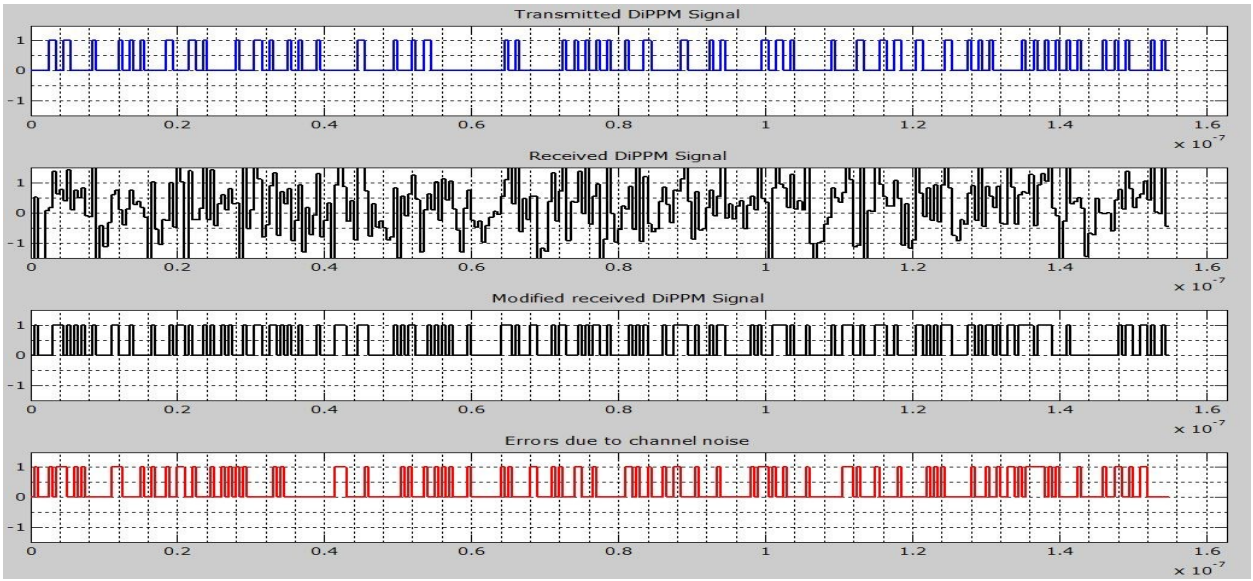


Figure 5.10 channel signals at snr=1dB

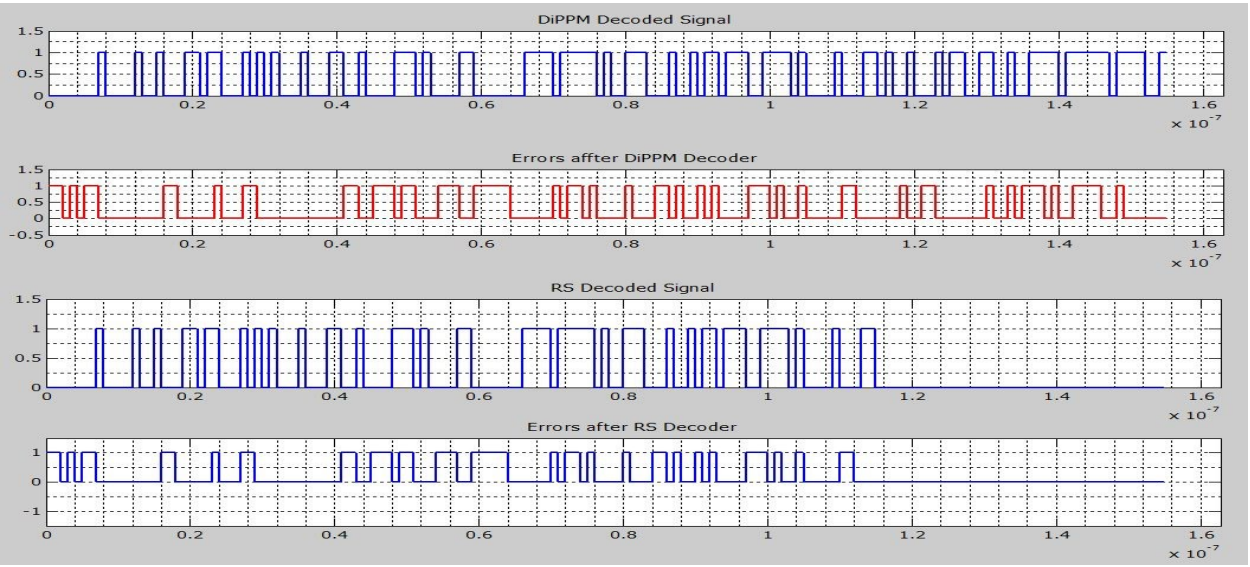


Figure 5.11 DiPPM with RS code Rx output waveform

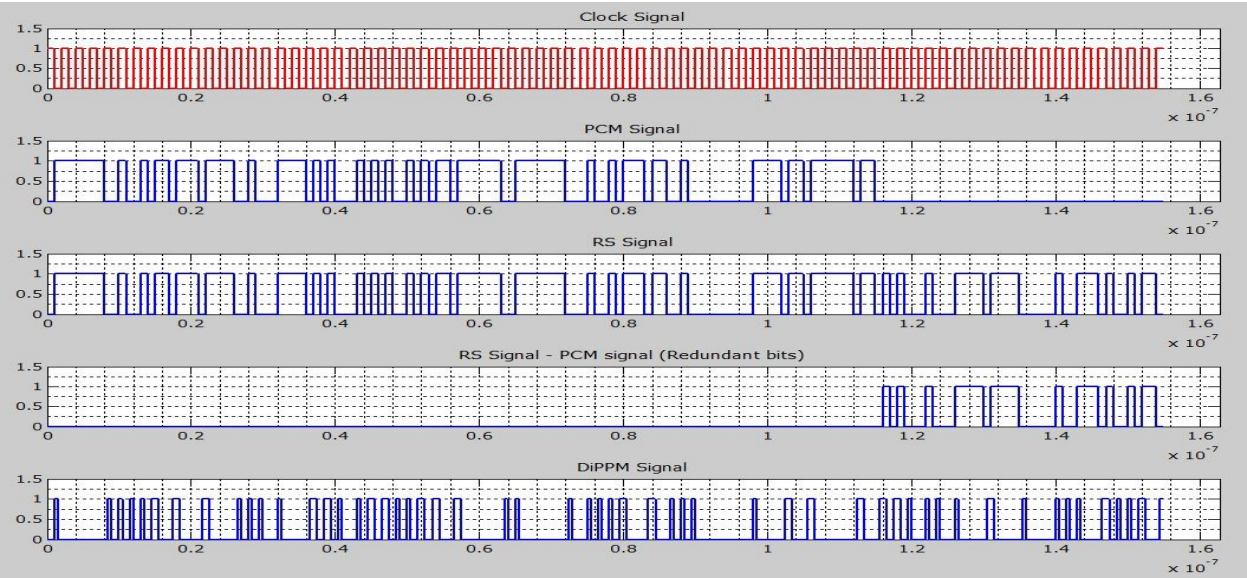


Figure 5.12 DiPPM with RS code Tx output waveform

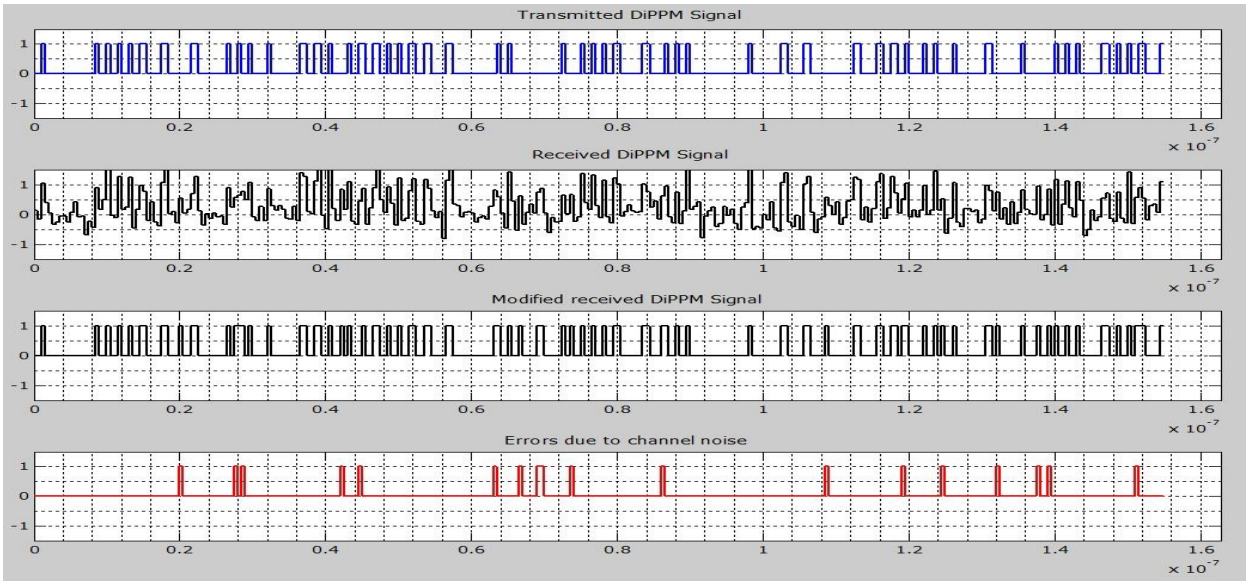


Figure 5.13 channel signals at snr=10dB

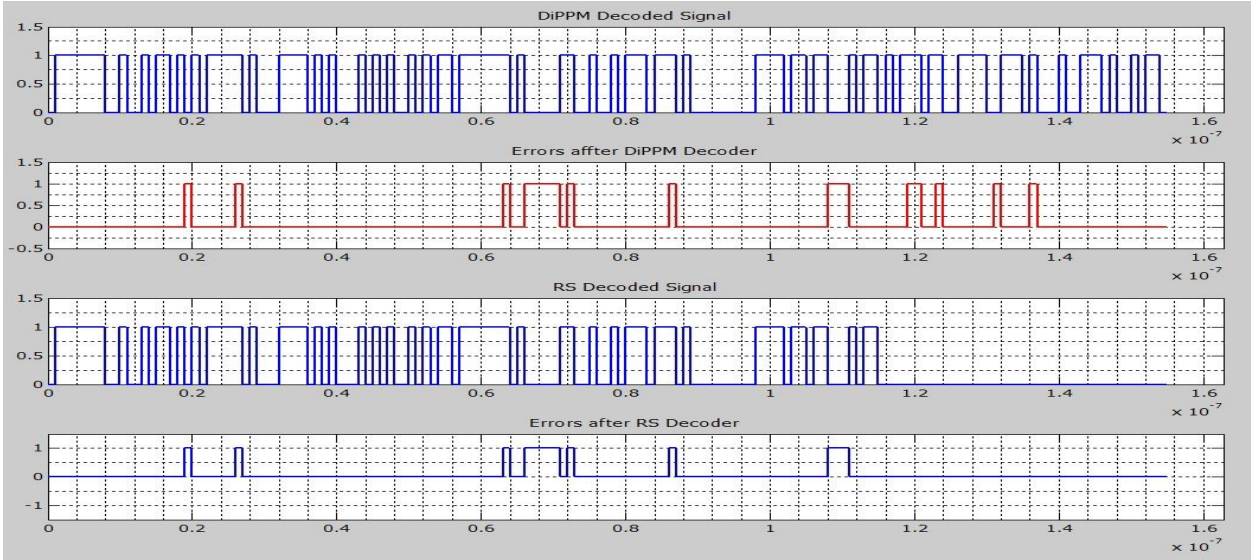


Figure 5.14 DiPPM with RS code Rx output waveform

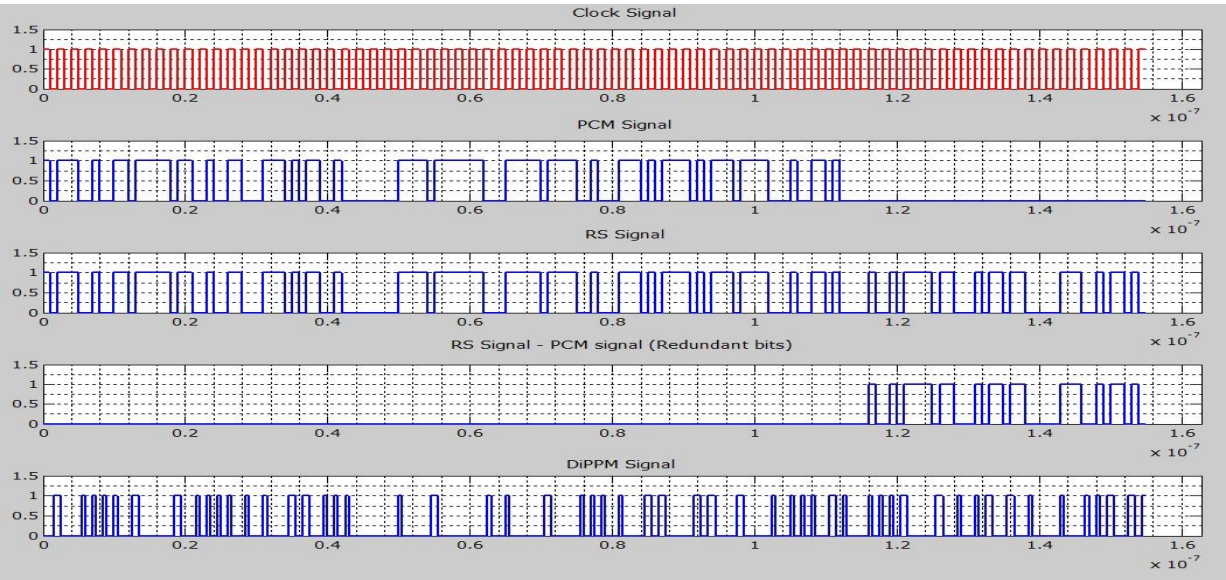


Figure 5.15 DiPPM with RS code Tx output waveform

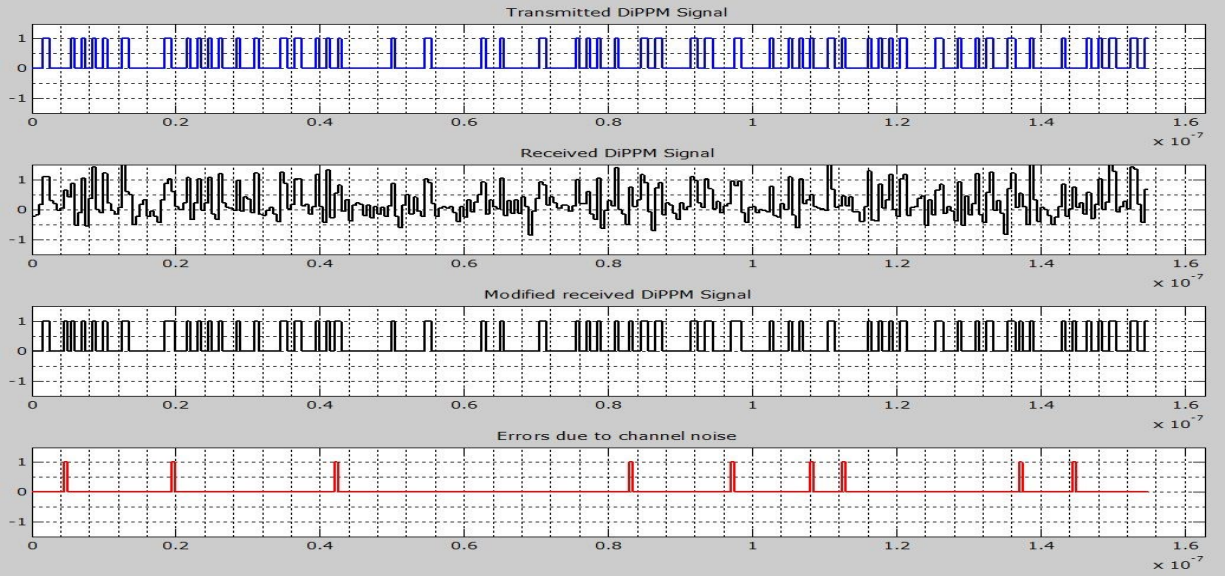


Figure 5.16 channel signals at snr=11dB

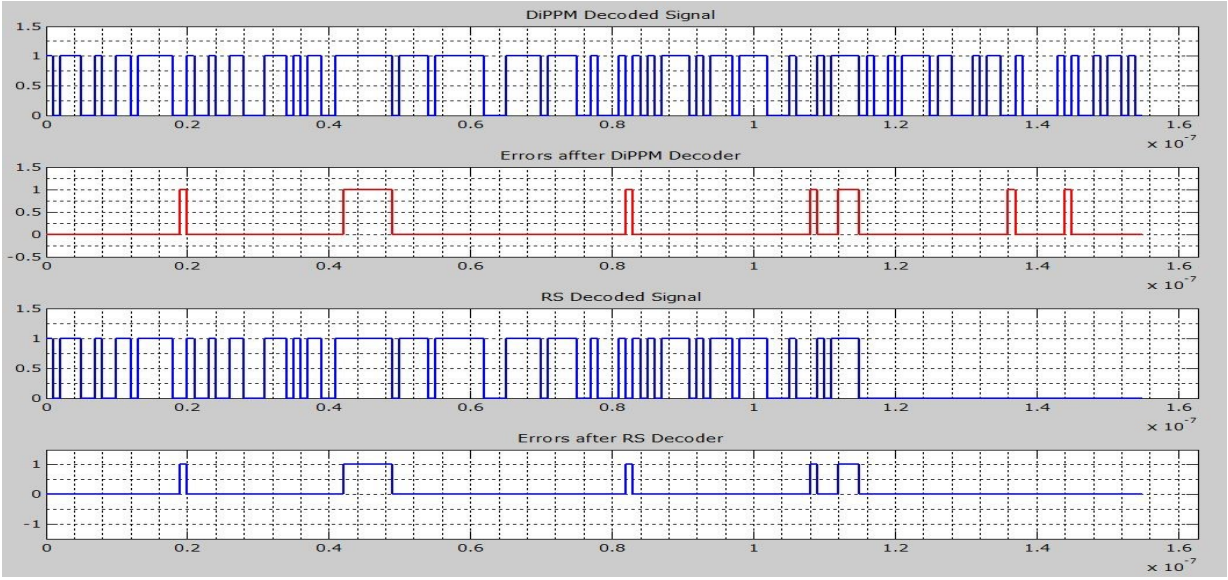


Figure 5.17 DiPPM with RS code Rx output waveform

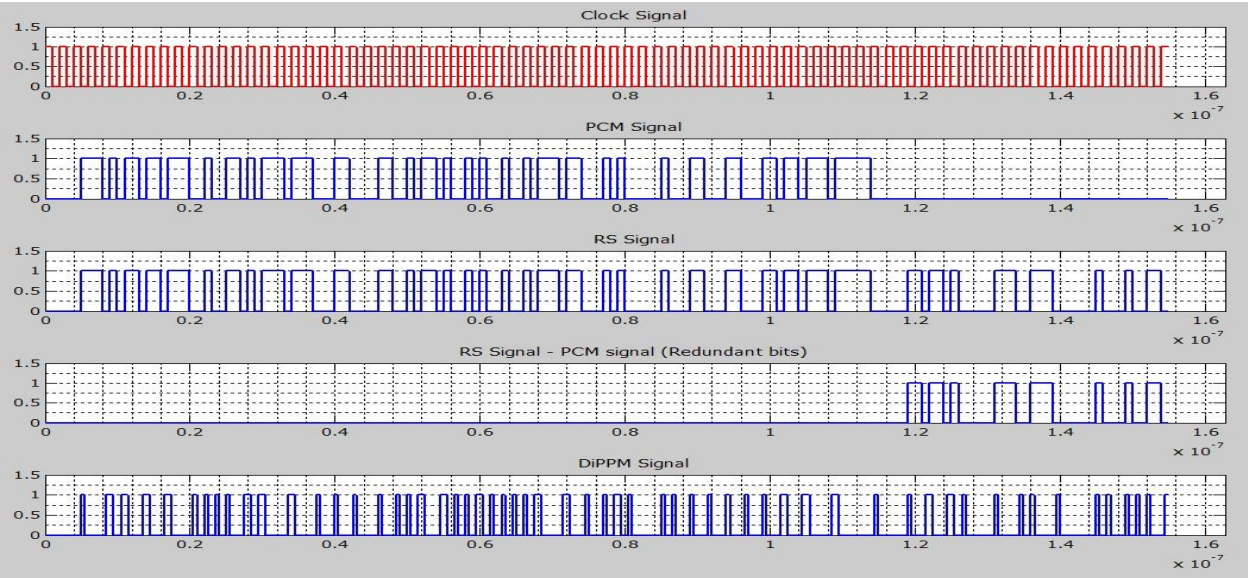


Figure 5.18 DiPPM with RS code Tx output waveform

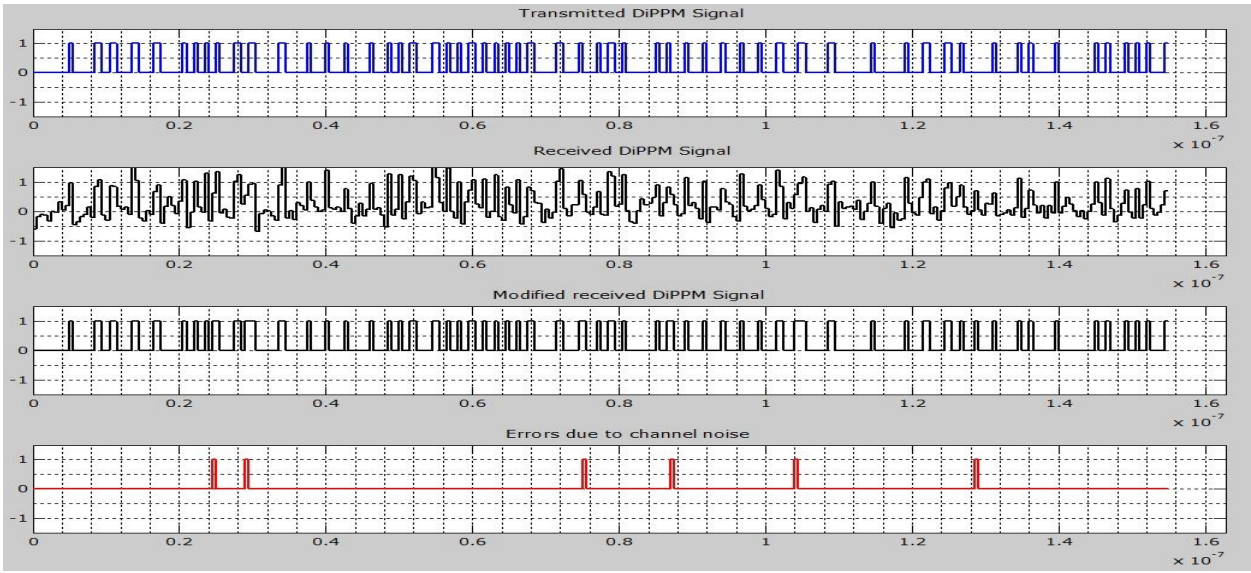


Figure 5.19 channel signals at snr=12dB

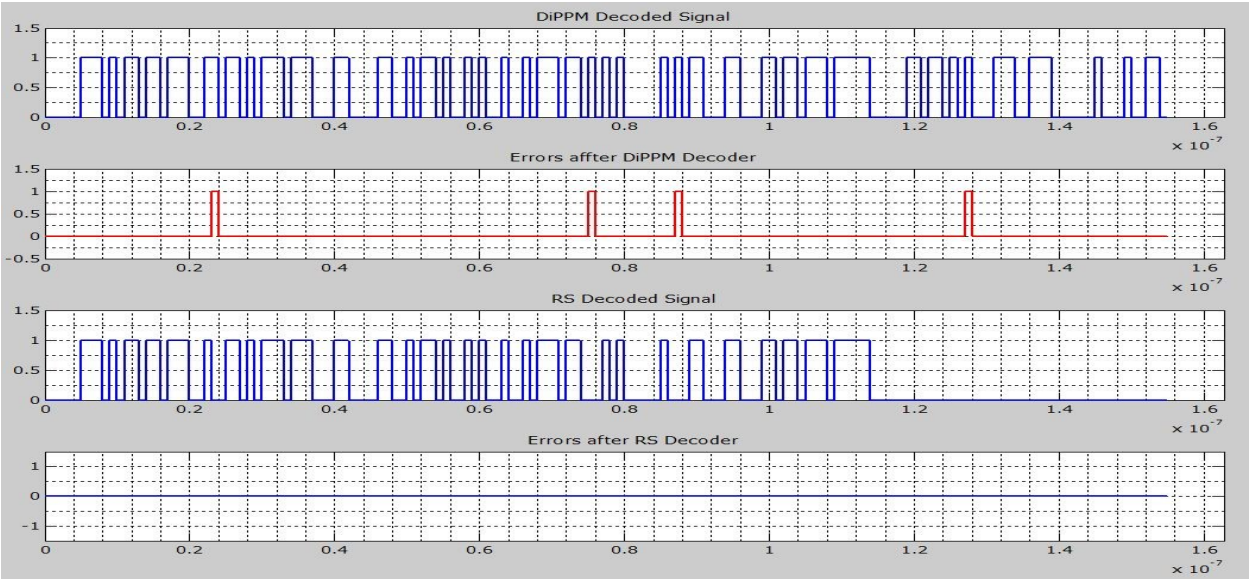


Figure 5.20 DiPPM with RS code Rx output waveform

5.5. DiPPM WITH RS SYSTEM (UPGRADED VERSION)

An upgraded version of the DiPPM employing RS code system is presented in this thesis. The RRBS PCM generator has been replaced by image read function in order to generate the message data.

$$imread('img.jpg')$$

After reading the image data, the message is processed by the RS coder and then sent through the channel to reach the receiver side. In the receiver side the original message is processed by the DiPPM decoder and RS decoder, and finally the image reshapes from the output of the RS decoder by using the function below:-

$$reshape(output\_Img, x\_size, y\_size)$$

The system has been run for many times and the number of symbol errors is changed. Figures 5.21, and 5.22 shows the output image sample, it is clear that the system effectively reshapes the original image when the number of symbol errors is equal to or less than four. The efficiency of the system starts decreasing when the number of error symbols is greater than its limit.

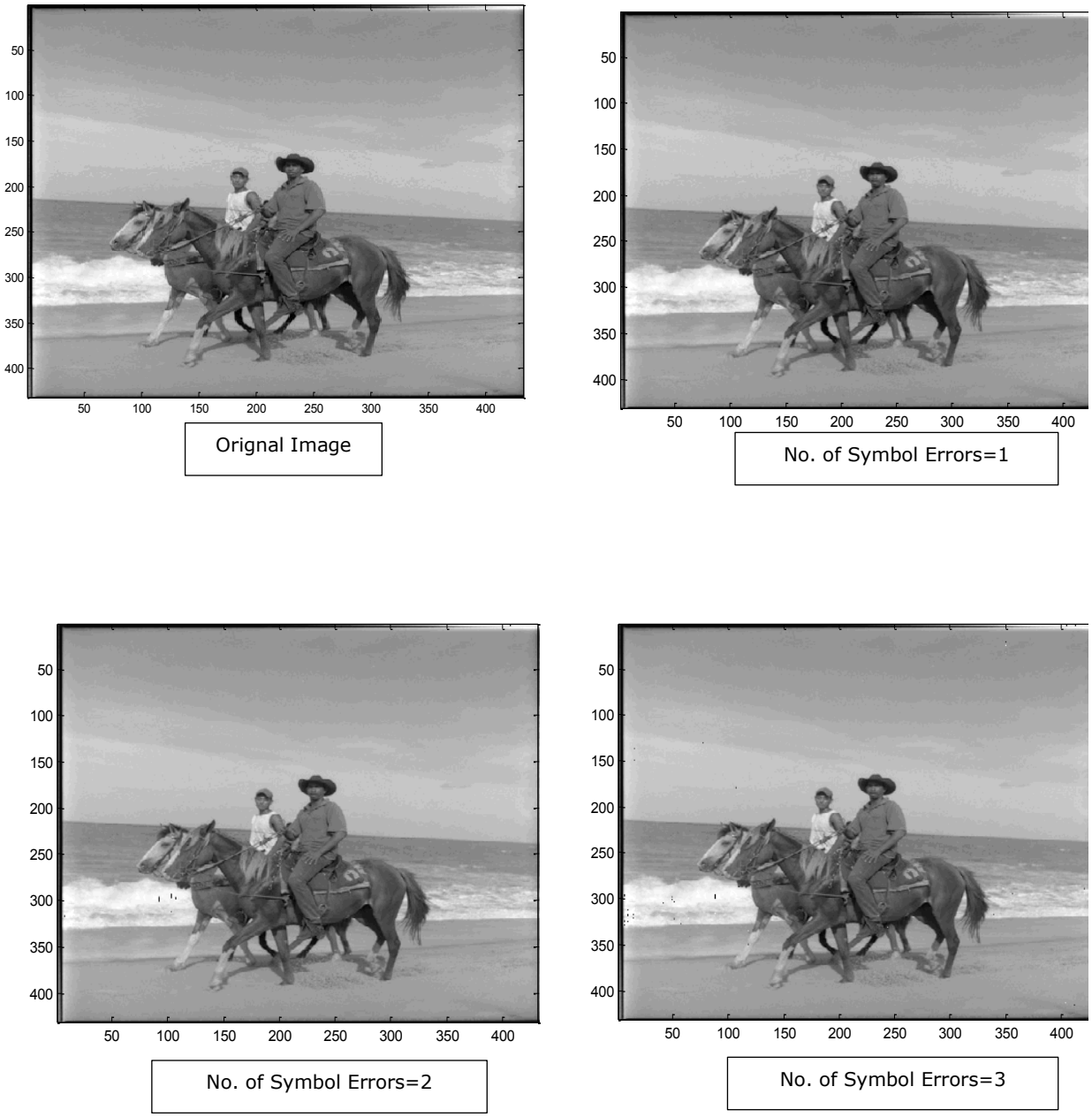
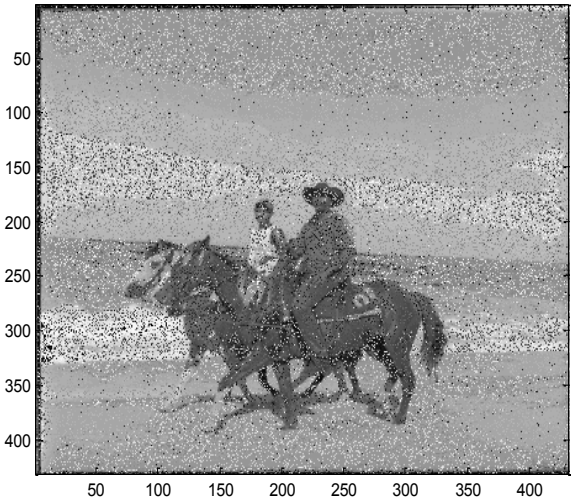


Figure 5.21 DiPPM with RS code received images

No. of Symbol Errors=4

No. of Symbol Errors=5

No. of Symbol Errors=8



No. of Symbol Errors=16

Figure 5.22 DiPPM with RS code received images

5.6. Summary

The DiPPM employing RS code system programme has been designed using Matlab software. The DiPPM system results achieved the theory of the DiPPM scheme. Adding the RS code system help the DiPPM scheme to overcome the errors that affect the transmitted data when the SNR is equal or above 12dB. However the RS code system should work in its optimum code rate.

VHDL SOURCE CODE AND SIMULATION ENVIRONMENT FOR THE DICODE PULSE POSITION MODULATION SYSTEM WITH REED SOLOMON CODE

6.1. Introduction

In this chapter, a very high speed integrated circuit (VHSIC) hardware description language (VHDL) source code for the DiPPM system employing (31,23) RS error correcting code system is given. A schematic and a full block description of the system is shown in the second section. Modelsim_Altera version (6.5b) software is used to simulate the system.

6.2. System Schematic

The DiPPM system employing RS code system schematic is shown in figure 6.1. In the transmitter side, the PRBS block is used to generate a random PCM message (*k=23 symbols*) sequence. The PCM message is coded by using (*31,23*) RS coder by adding redundancy symbols (*n-k=8symbols*). The bridge coder is used to convert the parallel output of the RS coder to serial, in order to be appropriate input for the DiPPM coder. In the receiver side, the DiPPM decoder receives the message, which is in form of DiPPM pulses (SET, RESET), to change it into PCM form. Then, the serial PCM converts to parallel by using the bridge decoder. The final stage is a (*31,23*) RS decoder used to extract the original message. A description for each part is given in the next subsections.

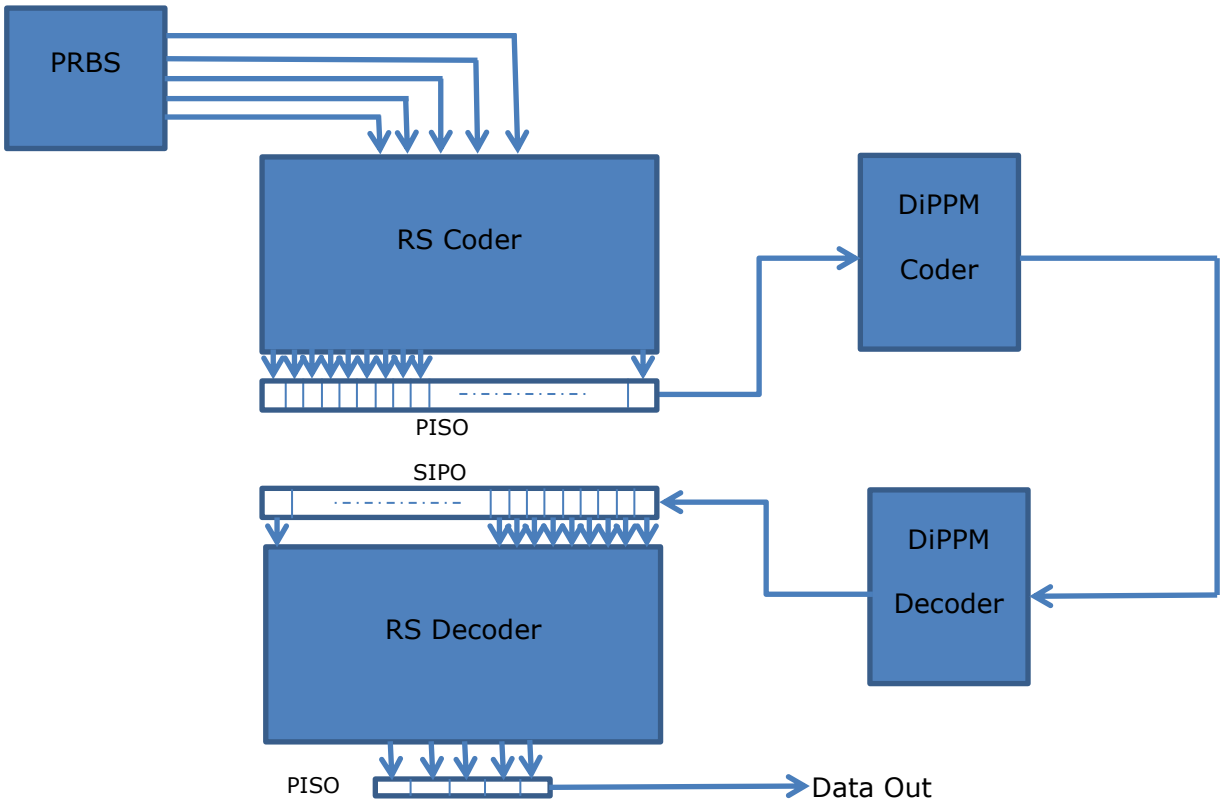


Figure 6.1 DiPPM and Reed Solomon System Schematic

6.2.1. Pseudo Random Binary Sequence (PRBS)

Linear feedback shift registers (LFSRs) are the logic circuits used to generate PRBS. The generated binary sequence has *N* bits length, $N=2^M-1$, which begins from *A0, A1, A2…A (N–1)*. The logic circuit consists of *M* registers, as the sum $\sum Aj$ binary ones and *N – M* binary zeros, where *j=0, 1, 2 …N-1. A* primitive polynomial creates a maximal length sequence, where the LFSR transitions through 2^M-1 states before repeating (Katz & Boriello, 2005; Lala, 1996).

The LFSR can be used to implement both serial and parallel outputs of PRBS. In this chapter a parallel PRBS VHDL source code has been designed to generate a message of *23 symbols*, each symbol contains *5 bits*. The Pseudo Random Binary Sequence (PRBS) source code is shown in table 6.1, Appendix(3) section (10.3.1).

Table 6.1 PRBS Source Code

file name	Description
pbrs.vhd	PBRS top module
pbrs_pkg.vhd	PBRS module package

The PRBS top block view is depicted in figure 6.2, while the input/output signals are displayed in table 6.2.

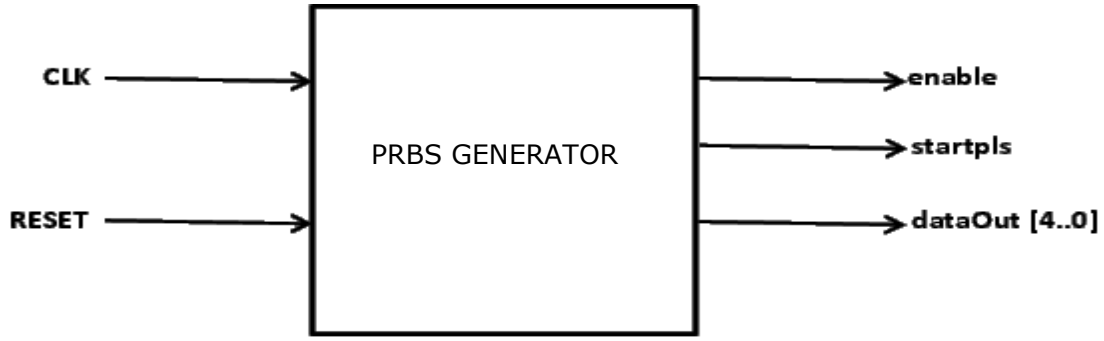


Figure 6.2 PRBS top block view

Table 6.2 PRBS input/output Signals

Signal Name	I/O	Description
CLK	I	System clock
RESET	I	System reset
Enable	O	Output data enable
startPls	O	Output start pulse
dataOut [4:0]	O	Output data

The PRBS input and output waveform for many codewords are shown in figure 6.3. Figure 6.4 shows one codeword *23 symbols* data output, the clock signal in the first line, and the second line for RESET signal, the third and fourth lines are for the ENABLE and START PULSE signals respectively, the fifth line for PRBS signal. The simulation runs at clock *100 MHz*, the generated message needs *23 clocks* to produce *23 symbols*, *8 clocks* are left for the encoding process. The PRBS is repeated every *170 clock* in order to give enough time for the decoding process.

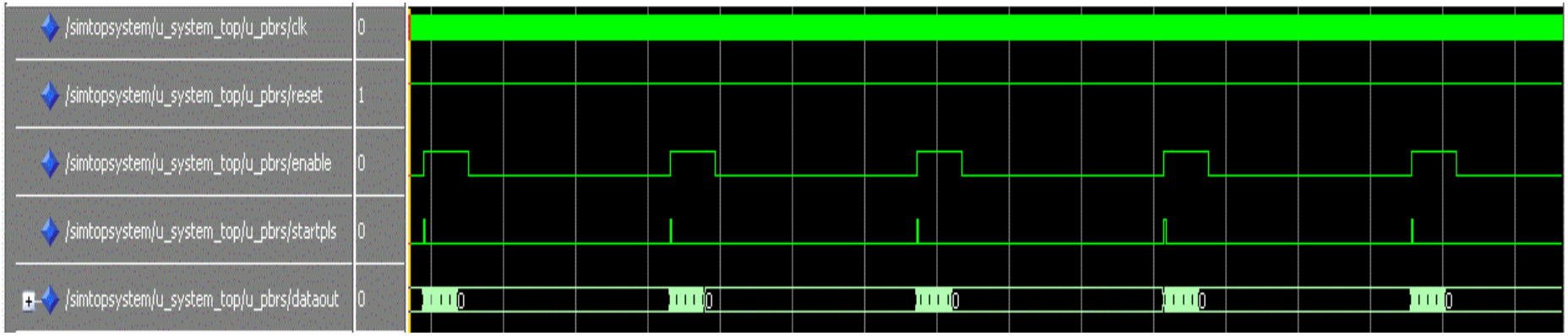


Figure 6.3 PRBS input/output waveform

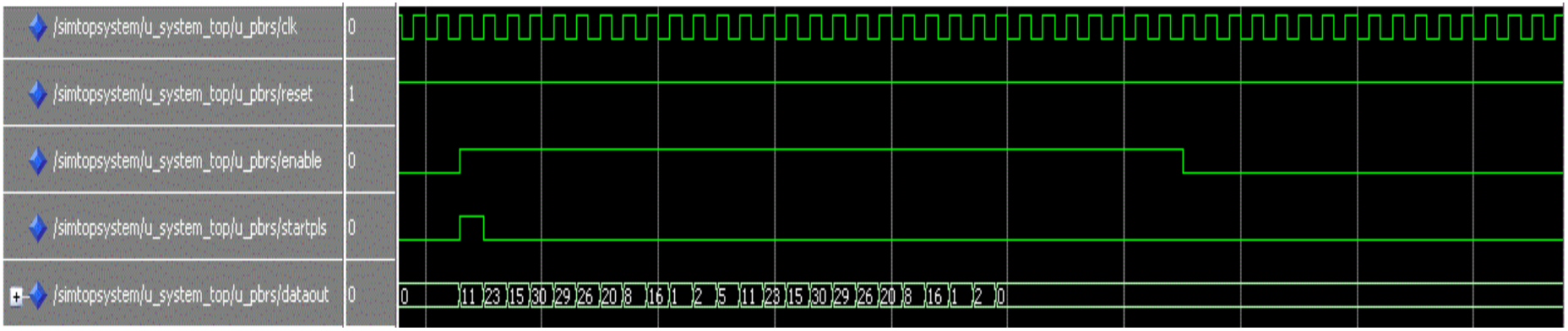


Figure 6.4 PRBS input/output one codeword zoom

6.2.2. Reed Solomon Coder

The RS code is specified as RS(*n, k*) with *m*-bit symbols. The block length contains *n* symbols; each symbol consists of *m* bits. In other words, the RS encoder combines *k* symbols data with parity symbols (redundancy) *2t* to produce an *n* symbols codeword (Sklar, 2001).

The RS encoder receives the message symbols and adds a parity *2t* symbols to create encoded blocks consisting of $n=2^m-1$ symbols each, where *m* is the symbol size in bits as described previously. Each message block is equivalent to a message polynomial of degree *k-1*, from linear algebra that any *k* distance points uniquely determine a polynomial of degree at most *k-1,* where *k* is the message length in symbols as

$$m(x) = m_0 + m_1 X + m_2 X^2 + ... + m_{k-1} X^{k-1} \tag{6.1}$$

Where the coefficients $m_0, m_1, m_2, \ldots, m_{k-1}$ of the polynomial *m(X)* are the symbols of a message block. Moreover, these coefficients are elements of $GF(2^m)$. Thus, the information sequence is mapped into an abstract polynomial by setting the coefficients equal to the symbol value.

The RS codeword is generated using a generator polynomial. The generating polynomial for an RS code takes the following form:

$$\begin{aligned} g(X) &= (X+\alpha)(X+\alpha^2)(X+\alpha^3)....(X+\alpha^{2t}) \\ g(X) &= g_0 + g_1 X + g_2 X^2 + ... + g_{2t-1} X^{2t-1} + X^{2t} \end{aligned} \tag{6.2}$$

Where α is a primitive element in $GF(2^m)$, and $g_0, g_1, g_2, \ldots, g_{2t-1}$ are the coefficients from $GF(2^m)$. The degree of the generator polynomial is equal to the number of parity check symbols. A general RS encoder circuit can be implemented using the generator polynomial as shown in fig 6.6 (Lin, & Costello, 1983; Sklar, 2001).

Hence, the RS coder will be shifting the message symbols sequence *m(X)* by *n-k* symbols and then dividing the result by generator polynomial *g(X)* to produce the codeword *c(X)* as in equations below:

$$c(X) = X^{n-k} m(X) + p(X) \tag{6.3}$$

$$p(X) = (X^{n-k} m(X)) \bmod g(X) \tag{6.4}$$

where *p(X)* is the remainder polynomial.

The polynomial generator used in this work to generate the field of RS(31,23) is X^5+X^2+1. The RS(31,23) coder polynomial, which is driven by using equation (6.2), can be written as below:

$$g(x) = \propto^4 + \propto^{16} X + \propto^{23} X^2 + \propto^{10} X^3 + \propto^{30} X^4 + \propto^{21} X^5 + \propto^{13} X^6 + \propto^{8} X^7 + X^8 \qquad (6.5)$$

Figure 6.5 shows the designed RS(31,23) coder circuit, which consists 8 registers to generate the codeword.

The Reed Solomon RS(31,23) coder VHDL source code is shown in table 6.3, Appendix (3) section (10.3.2).

Table 6.3 RS (31,23) coder source code

file name	Description
rscoder_31_23_top.vhd	RS(31,23) coder top module
rscoder_31_23_top_pkg.vhd	RS(31,23) coder top module package

The RS(31,23) coder top block view is depicted in figure 6.5, while the input/output signals are displayed in table 6.4.

Figure 6.5 RS(31,23) coder top block view

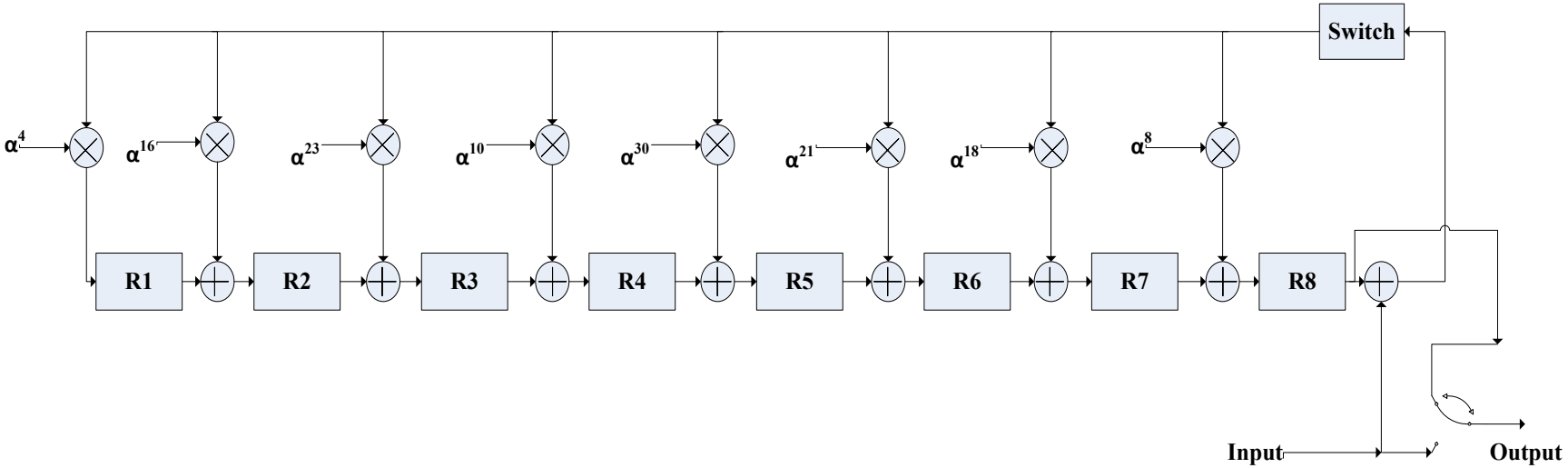


Figure 6.6 RS(31,23) coder circuit

Table 6.4 RS(31,23) coder IO signals

Signal Name	I/O	Description
CLK	I	System clock
RESET	I	System reset
Enable	I	Input data enable
startPls	I	Input start pulse
dataIn[4:0]	I	Input data
dataOut [4:0]	O	Output data

The RS(31,23) coder inner block view is depicted in the following figure 6.7.

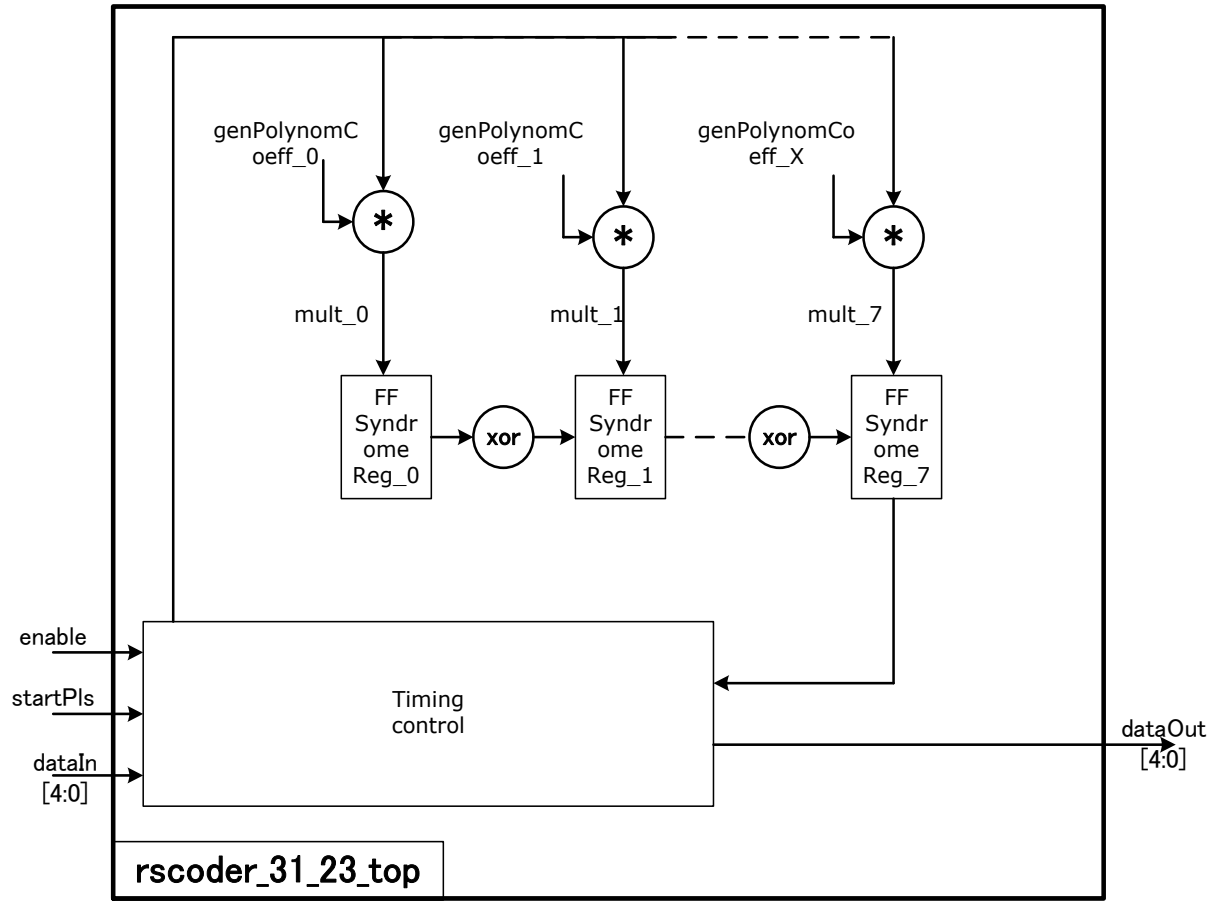


Figure 6.7 RS(31,23) coder inner block view

The RS(31,23) coder timing chart is illustrated in the following figure 6.8. The RS(31,23) coder input and output waveform simulation for many codewords are shown in figure 6.9, while the figure 6.10 shows one codeword *31 symbols* data output.

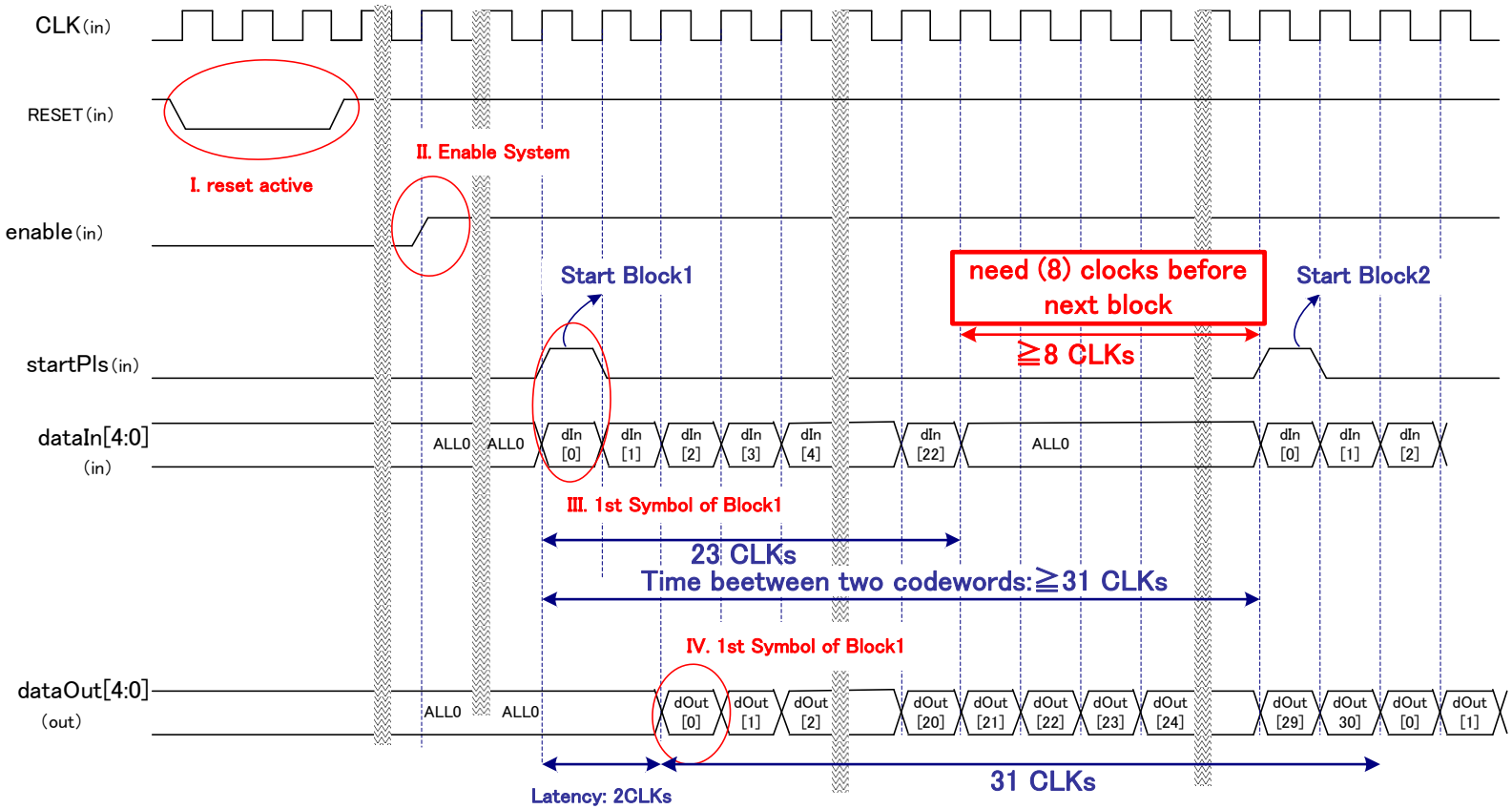


Figure 6.8 RS(31,23) coder timing chart

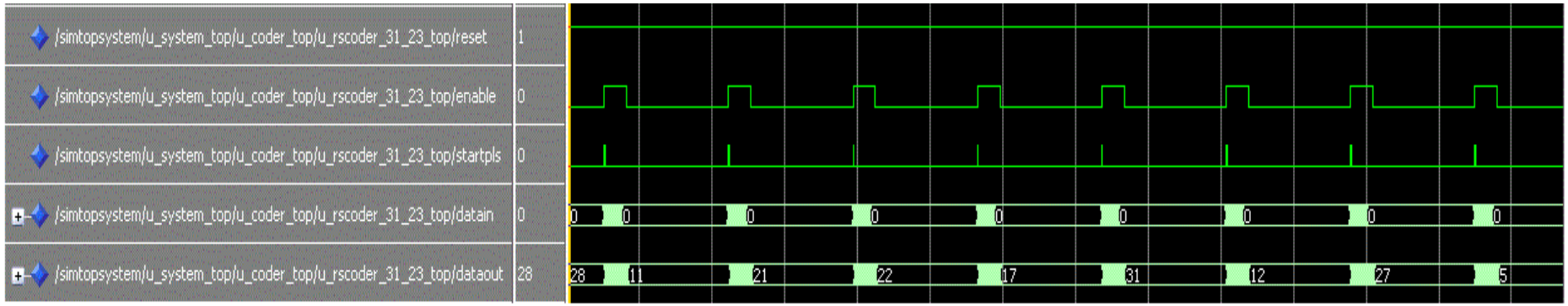


Figure 6.9 RS(31,23) coder input/output waveform

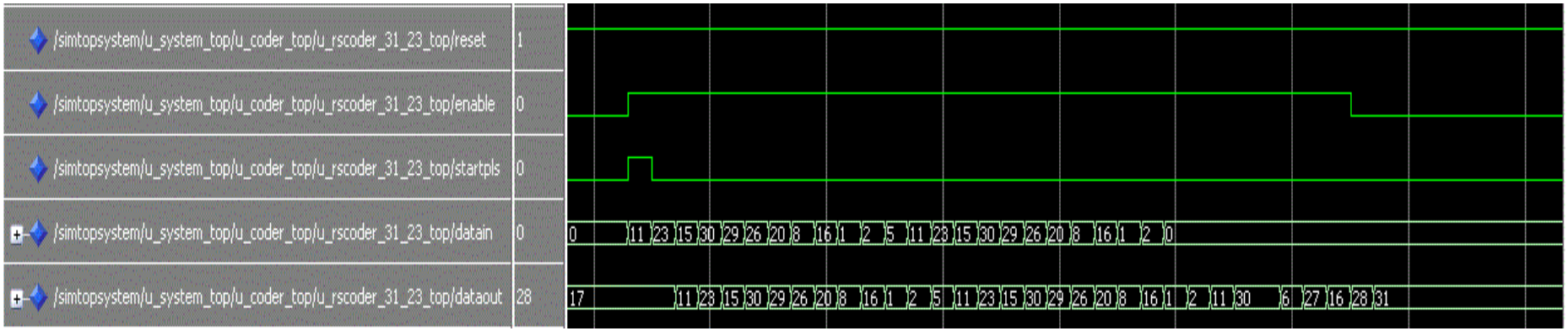


Figure 6.10 RS(31,23) coder input/output one codeword zoom

6.2.3. Bridge Coder (Parallel Input Serial Output)

The PISO is a shift register circuit which changes the data from parallel to serial. All the shift register inputs of the data bits enter the parallel input pins simultaneously (Hetzel, 1988). The reading of the input data takes place in a sequential order inside the PISO register in a shift-right mode (Maini, 2007). The output of the data comes out 1 bit each time on every clock cycle (Crowell & Press, 2004). The bridge coder source code is shown in table 6.5, Appendix (3) section (10.3.3).

Table 6.5 Bridge Coder Source Code

file name	Description
bridgecoder_top.vhd	Bridge coder top module
bridgecoder_dpram.vhd	Bridge coder dual port ram memory
bridgecoder_top_pkg.vhd	Bridge coder module package

The bridge coder top block view is depicted in figure (6.11), while the input/output signals are displayed in table (6.6).

Figure 6.11 Bridge coder top block view

Table 6.6 Bridge coder I/O signals

Signal Name	I/O	Description
CLK	I	System clock
RESET	I	System reset
Enable	I	Input data enable
startPls	I	Input start pulse
dataIn[4:0]	I	Input data
dataOut	O	Output data
startplsOut	O	Output start pulse
enOut	O	Output data enable

The bridge coder input and output waveform for many codewords are shown in figure 6.12, while the figure 6.13 shows one codeword data output.

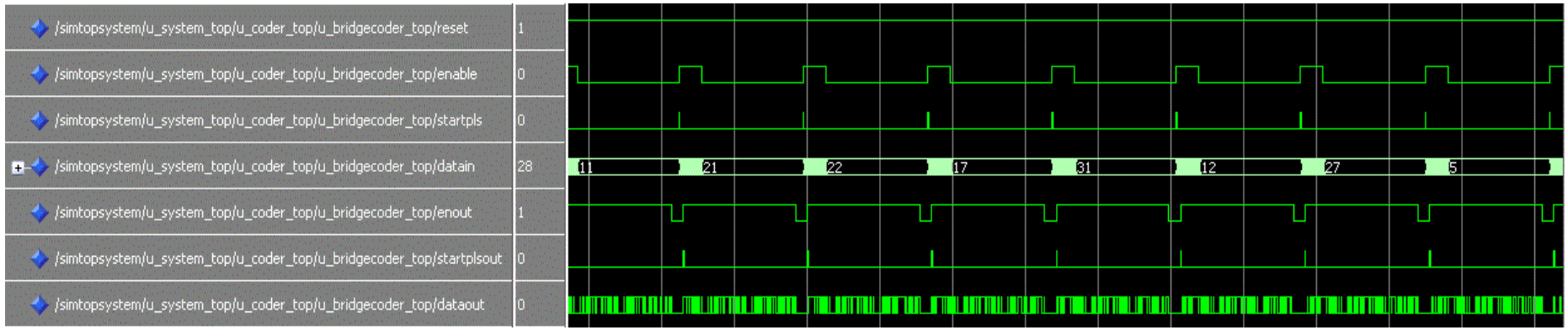


Figure 6.12 Bridge coder I/O waveform

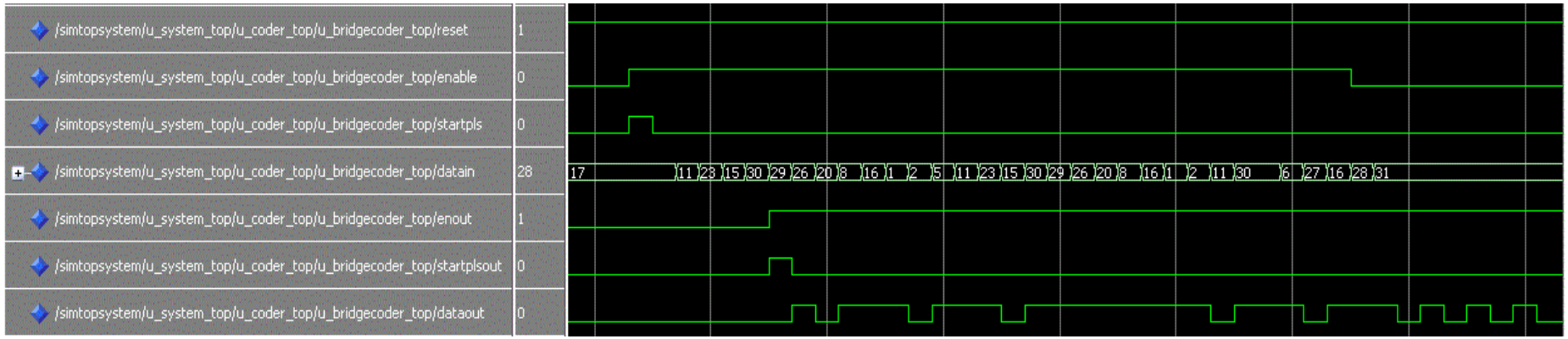


Figure 6.13 Bridge coder I/O waveform one codeword zoom

6.2.4. DiPPM Coder

The DiPPM coder is the stage of the DiPPM system located at the transmitter side in which the PCM input data is coded into the DiPPM format. The DiPPM coder circuit is composed from two Flip-Flops and five NOR gates as shown in figure 3.6 (Charitopoulos, 2009).

The two Flip-Flops use the input PCM format and the CLK pulses to generate the SET and RESET pulses. Then both pulses combine using the NOR gate to produce the final DiPPM sequence. Although the VHDL code for the DiPPM coder was designed (Charitopoulos, 2009), a modification is done to deliver a new source code, table 6.7, Appendix (3) section (10.3.4), in order to be compatible with RS.

Table 6.7 DiPPM coder source code

file name	Description
DiPPMcoders.vhd	DiPPM coder top module
DiPPMcoders_top_pkg.vhd	DiPPM coder module package

The DiPPM coder top block view is depicted in figure 6.14, while the input/output signals are displayed in table 6.8.

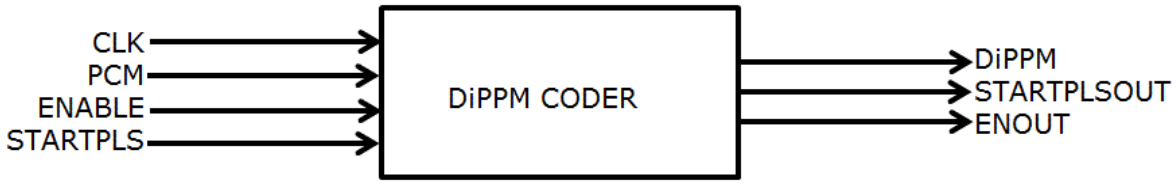


Figure 6.14 DiPPM coder top block view

Table 6.8 DiPPM I/O signals

Signal Name	I/O	Description
CLK	I	System clock
PCM	I	Input data
enableIn	I	Input data enable
startPlsIn	I	Input start pulse
DiPPM	O	Output data
enableOut	O	Output data enable
startPlsOut	O	Output start pulse

The DiPPM coder input and output waveform for many codewords are shown in figure 6.15, while the figure 6.16 shows one codeword data output.

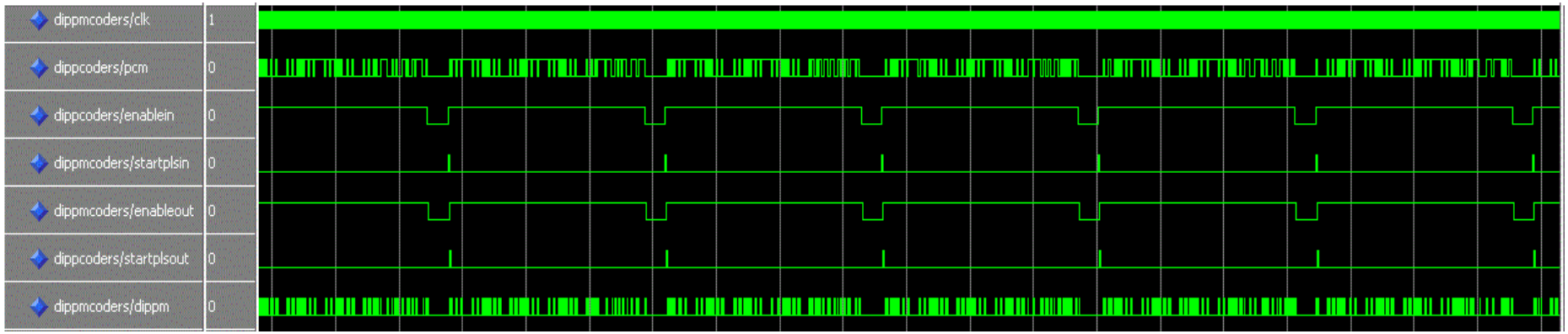


Figure 6.15 DiPPM coder I/O waveform

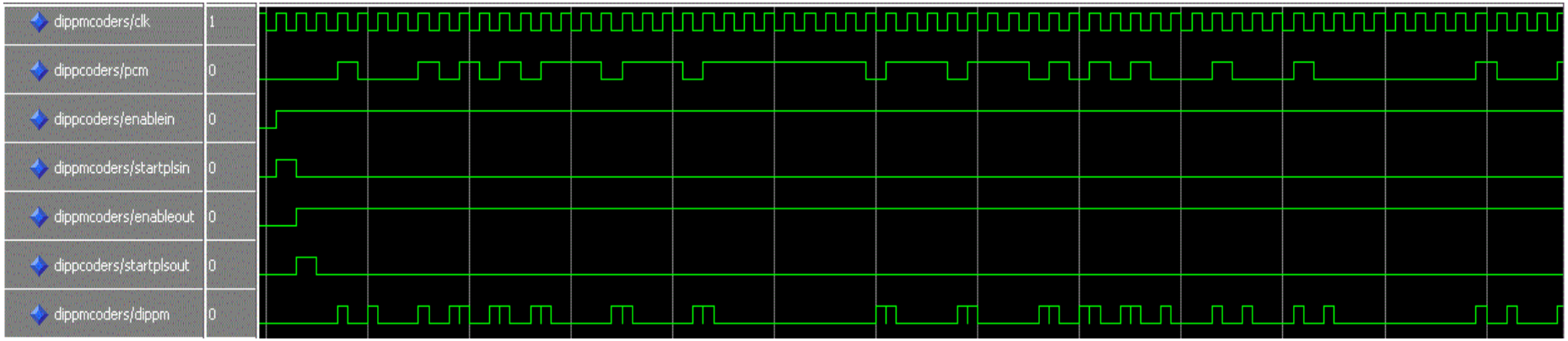


Figure 6.16 DiPPM coder I/O waveform one codeword zoom

6.2.5. DiPPM Decoder

The DiPPM format is converted to its original PCM format by using DiPPM decoder. The DiPPM decoder constructed from a one OR gate, four NOR gates, a D Flip-Flop and a Direct Set/Clear component as shown in figure 3.9 (Charitopoulos,2009).

The DiPPM decoder VHDL code was design by Charitopoulos, a modifications is made to deliver a new source code table 6.9, Appendix (3) section (10.3.6).

Table 6.9 DiPPM decoder source code

file name	Description
DiPPMdecoder.vhd	DiPPM decoder top module
DiPPMdecoder_top_pkg.vhd	DiPPM decoder module package

The DiPPM decoder top block view is depicted in figure 6.17, while the input/output signals are displayed in table 6.10.

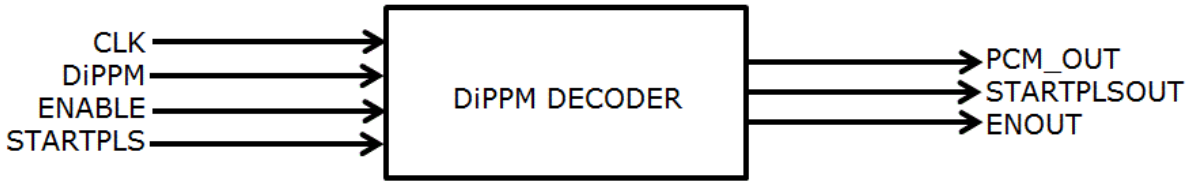


Figure 6.17 DiPPM decoder top block view

Table 6.10 DiPPM decoder I/O signals

Signal Name	I/O	Description
CLK	I	System clock
DiPPM	I	Input data
enableIn	I	Input data enable
startPlsIn	I	Input start pulse
PCM_Out	O	Output data
enableOut	O	Output data enable
startPlsOut	O	Output start pulse

The DiPPM coder input and output waveform for many codewords are shown in figure 6.18, while the figure 6.19 shows one codeword data output.

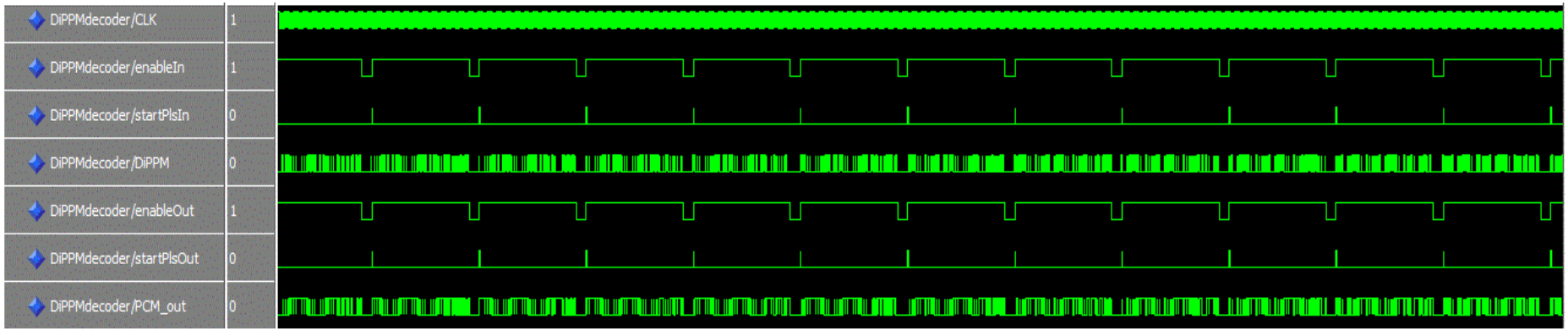


Figure 6.18 DiPPM decoder I/O waveform

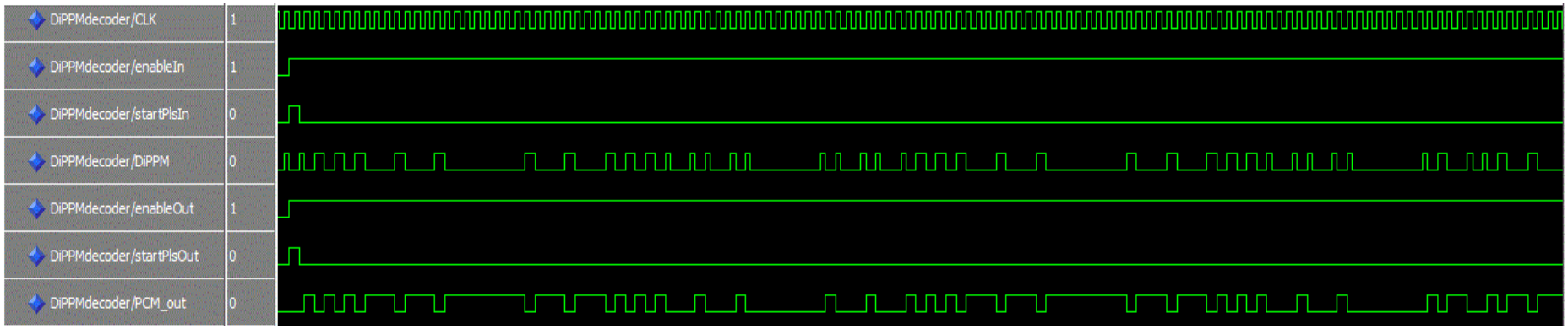


Figure 6.19 DiPPM decode I/O waveform one codeword zoom

6.2.6. Bridge Decoder (Serial Input Parallel Output)

A serial-in/parallel-out shift register converts data from serial format to parallel format. If five data bits are shifted in by five clock pulses via a single wire at data-in, the data becomes available simultaneously on the five Outputs after the fifth clock pulse, (Maini, 2007). The bridge decoder source code is shown in table 6.11, Appendix (3) section (10.3.7).

Table 6.11 Bridge decoder source code

file name	Description
bridgedecoder_top.vhd	Bridge decoder top module
bridgedecoder_dpram.vhd	Bridge decoder dual port ram memory
bridgedecoder_top_pkg.vhd	Bridge decoder module package

The bridge coder top block view is depicted in figure 6.20, while the input/output signals are displayed in table 6.12.

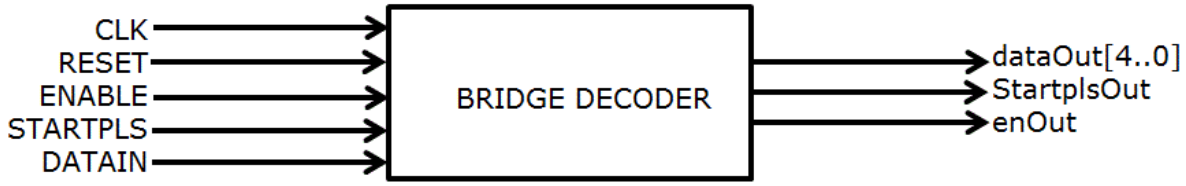


Figure 6.20 Bridge decoder top block view

Table 6.12 Brige decoder I/O signals

Signal Name	I/O	Description
CLK	I	System clock
RESET	I	System reset
Enable	I	Input data enable
startPls	I	Input start pulse
dataIn	I	Input data
dataOut[4..0]	O	Output data
startplsOut	O	Output start pulse
enOut	O	Output data enable

The bridge coder input and output waveform for many codewords are shown in figure 6.21, while the figure 6.22 shows one codeword data output.

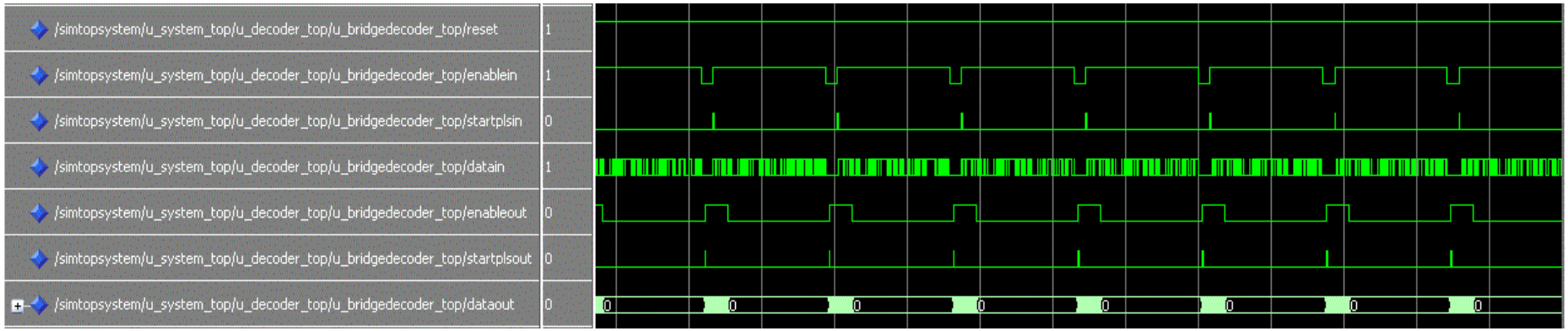


Figure 6.21 Bridge decoder I/O waveform

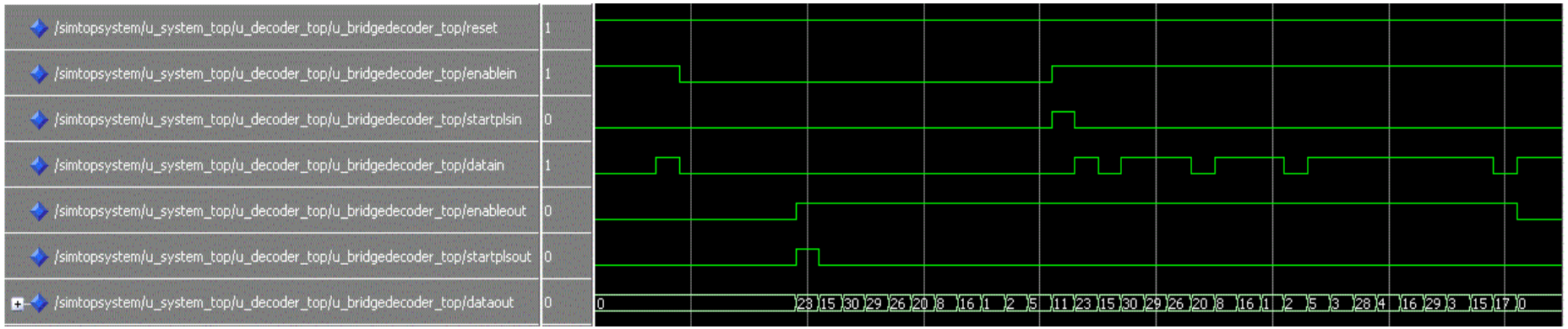


Figure 6.22 Bridge decoder I/O waveform one codeword zoom

6.2.7. Reed Solomon Decoder

The RS coder receives the transmitted codeword and then processes it to produce the message polynomial *m(X)* (Tocci, 2006). The received codeword may be corrupted during the transmission; the received corrupted codeword polynomial is then represented by adding the error pattern polynomial *e(X)* to the transmitted codeword polynomial *c(X)*:

$$r(X) = c(X) + e(X) \tag{6.6}$$

where *r(X)* is the received corrupted codeword polynomial. A RS decoder can correct up to *t* symbols that contain errors in a codeword, where *2t =n-k*. This means that the RS decoder has the ability to correct up to *t* symbols that contain errors in a codeword, where *t* can be defined as

$$t = \left[\frac{n-k}{2} \right] \tag{6.7}$$

Figure 6.23, shows a typical RS codeword (this is known as a Systematic code because the data is left unchanged and the parity symbols are appended) (Sklar, 2001):

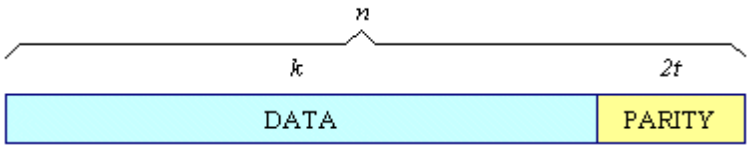


Figure 6.23 Typical RS codeword

Equation (6.7) illustrates that for the case of R-S codes, correcting *t* symbol errors requires no more than 2*t* parity symbols (Brown, & Vranesic, 2009). Equation 6.7 lends itself to the following intuitive reasoning. One can say that the decoder has *n-k* redundant symbols to "spend," which is twice the amount of correctable errors. For each error, one redundant symbol is used to locate the error, and the other redundant symbol is used to find its correct value. The erasure-correcting capability, γ, of the code is (Lu, Willi & Serge, 2005):

$$\gamma = d_{\min} - 1 = n - k \tag{6.8}$$

Where d_{min} is the code minimum distance:

$$d_{\min} = n - k - 1 \tag{6.9}$$

Simultaneous error-correction and erasure-correction capability can be expressed as follows:

$$2\sigma + \gamma < d_{\min} < n - k \text{ `} \tag{6.10}$$

Where σ is the number of symbols-error patterns that can be corrected and γ is the number of symbol erasure patterns that can be corrected. Any linear code is capable of correcting *n-k* symbol erasure patterns if the *n–k* erased symbols all happen to lie on the parity symbols. However, RS codes have the property that they are able to correct *any* set of *n-k* symbol erasures within the block. RS codes can be designed to have any redundancy. However, the complexity of a high-speed implementation increases with redundancy. Thus, the most attractive RS codes have high code rates (low redundancy).

The decoding process consists of four main steps. First, calculating the syndromes, and then finding the coefficients of the error location polynomial. After that calculating the error patterns and finally finding the error values and correcting the received codeword. Figure 6.24, shows the four stages of RS decoder (Sklar, 2001)

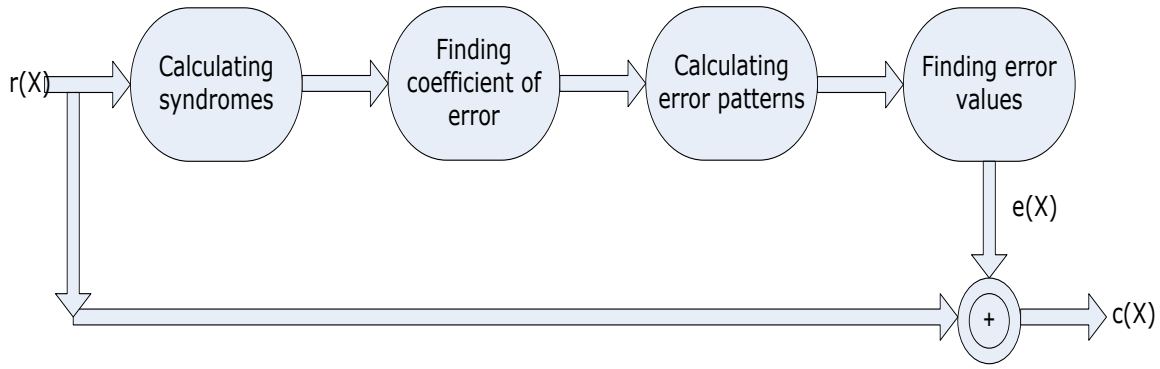


Figure 6.24 Reed Solomon decoder stages

The RS decoder can correct any number of symbol errors, within the boundaries of design, without regard to the type of damage suffered by the symbol. In other words, when a decoder corrects a symbol, it exchanges the incorrect symbol with the correct one, whether the error was caused by one bit being corrupted or all the bits being corrupted. That is why RS codes perform well against burst error and also gives RS codes a burst noise advantage over binary codes (Sklar, 2001). The Reed Solomon RS(31,23) decoder source code is shown in table 6.13, Appendix (3) section (10.3.8).

Table 6.13 RS(31,23) decoder source code

file name	Description
rsdecoder_31_23_chien.vhd	RS(31,23) decoder chien algorithm
rsdecoder_31_23_degree.vhd	RS(31,23) decoder polynomial degree calculation
rsdecoder_31_23_delay.vhd	RS(31,23) decoder dual port ram memory controller
rsdecoder_31_23_dpram.vhd	RS(31,23) decoder dual port ram memory
rsdecoder_31_23_erasure.vhd	RS(31,23) decoder erasure polynomial calculation
rsdecoder_31_23_euclide.vhd	RS(31,23) decoder euclide algorithm
rsdecoder_31_23_inv.vhd	RS(31,23) decoder inverse in Galois field
rsdecoder_31_23_Mult.vhd	RS(31,23) decoder multiplication in Galois field
rsdecoder_31_23_pkg.vhd	RS(31,23) decoder package
rsdecoder_31_23_polymul.vhd	RS(31,23) decoder syndrome and erasure polynomial calculation
rsdecoder_31_23_shift_omega.vhd	RS(31,23) decoder omega polynomial shift
rsdecoder_31_23_syndrome.vhd	RS(31,23) decoder syndrome polynomial calculation
rsdecoder_31_23_top.vhd	RS(31,23) decoder top module
rsdecoder_31_23_top_pkg.vhd	RS(31,23) decoder top module package

The RS(31,23) decoder top block view is depicted in figure 6.25, while the input/output signals are displayed in table 6.14.

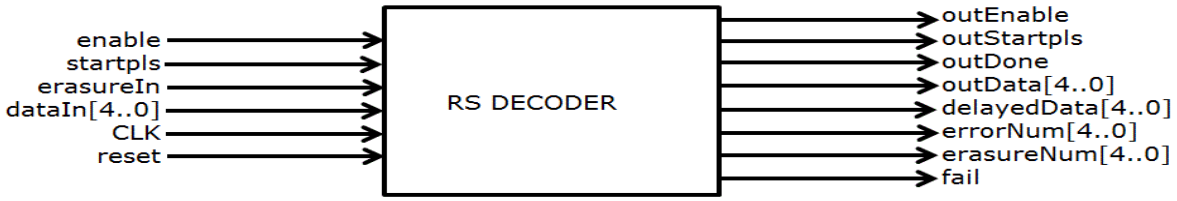


Figure 6.25 RS(31,23) decoder top block view

Table 6.14 RS(31,23) decoder I/O signals

Signal Name	I/O	Description
CLK	I	System clock
RESET	I	System reset (active low)
Enable	I	Input data enable (active high)
startPls	I	Input start pulse (active high)
dataIn[4:0]	I	Input data
erasureIn	I	Input erasure signal (‘0’ : no erasure, ‘1’ :erasure)
outData[4:0]	O	Output data
delayedData[4:0]	O	Delayed input data
outEnable	O	Output enable
outStartPls	O	Output start pulse
outdone	O	Output done, last data of a codeword (active high)
Fail	O	Output pass/fail flag（0:sucess, 1:failure）
errorNum[4:0]	O	Corrected amount of errors (valid only when decoding process is succesful)
erasureNum[4:0]	O	Corrected amount of erasures (valid only when decoding process is succesful)

The RS(31,23) decoder inner block view is depicted in the following figure 6.26.

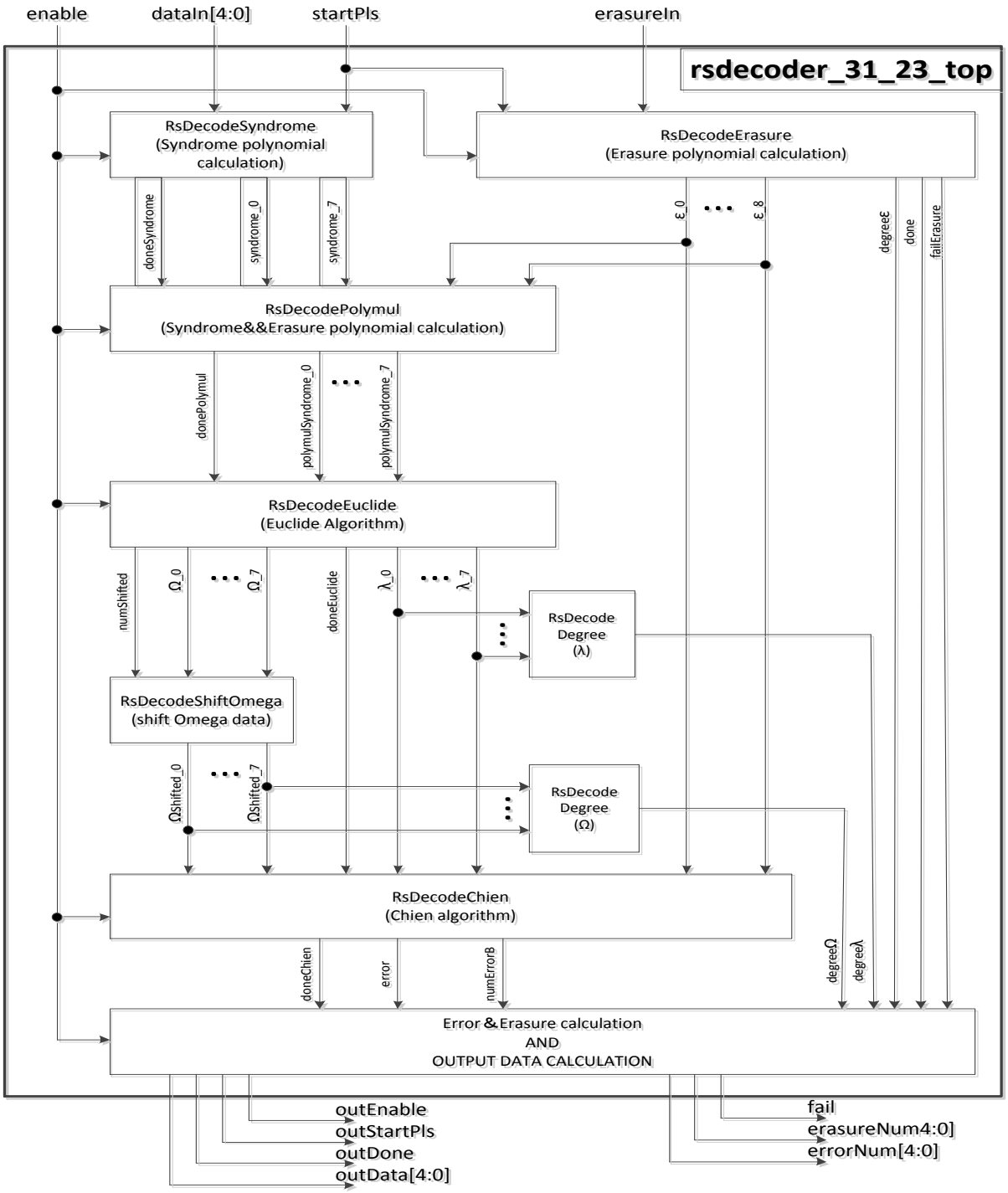


Figure 6.26 RS(31,23) decoder inner block view

The RS(31,23) decoder timing chart is illustrated in the following figure (6.27).

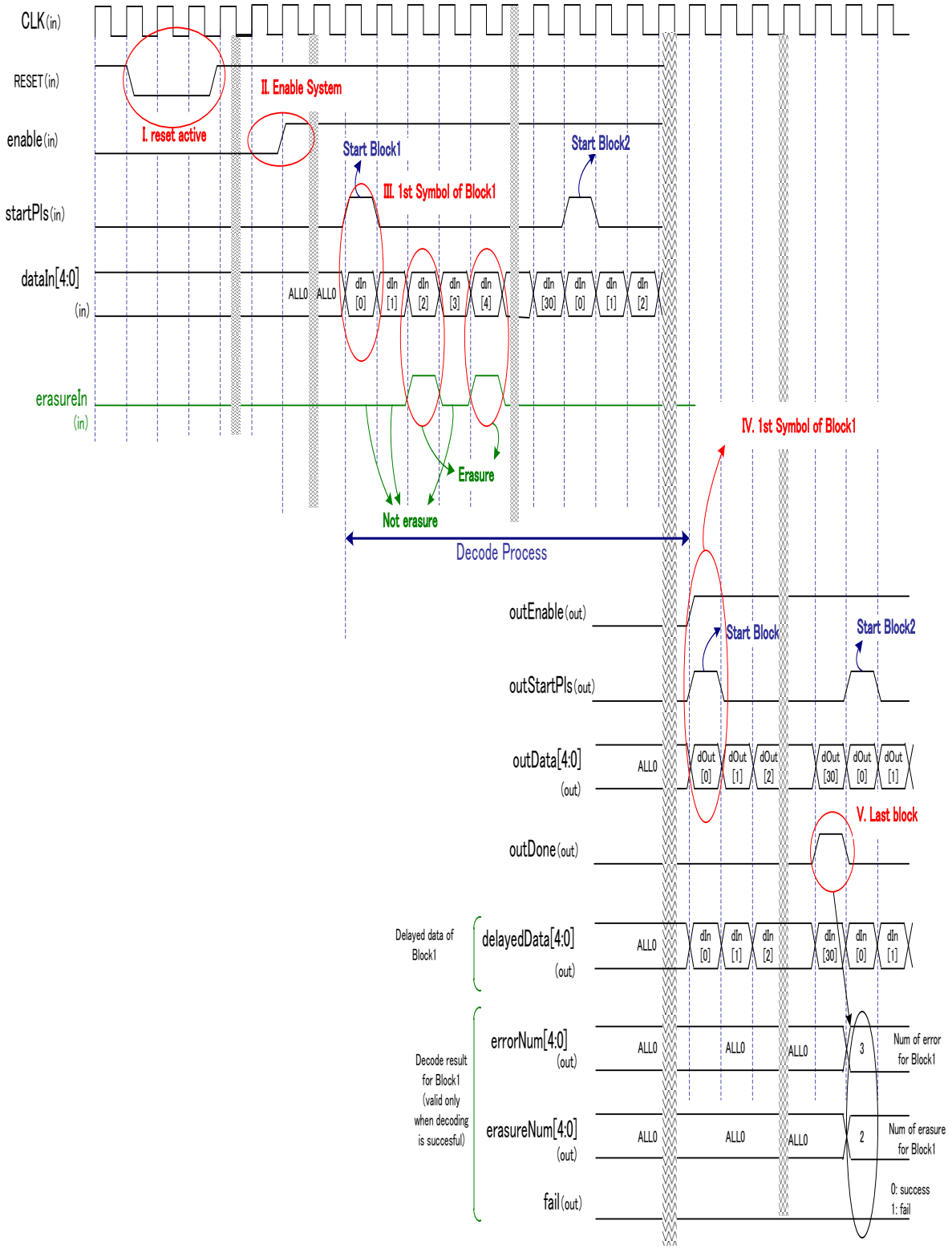


Figure 6.27 RS(31,23) decoder timing chart

The RS(31,23) decoder input and output waveform for many codewords are shown in figure 6.28, while the figure 6.29 shows one codeword data output.

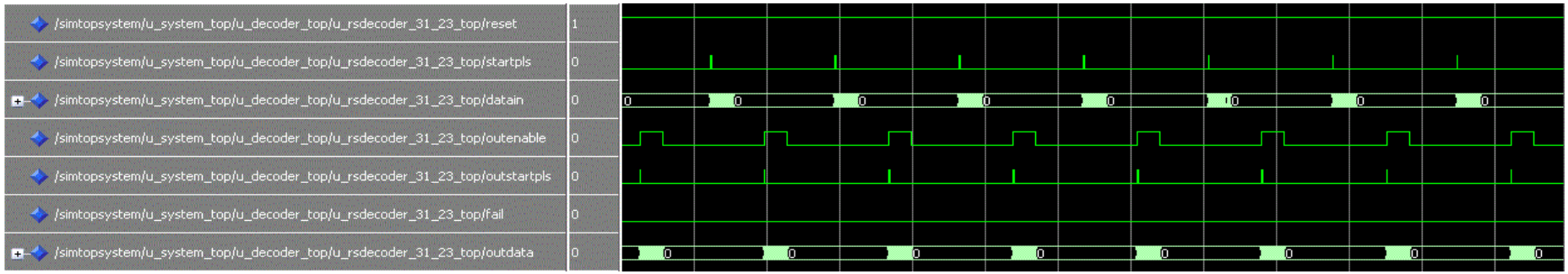


Figure 6.28 RS(31,23) decoder I/O waveform

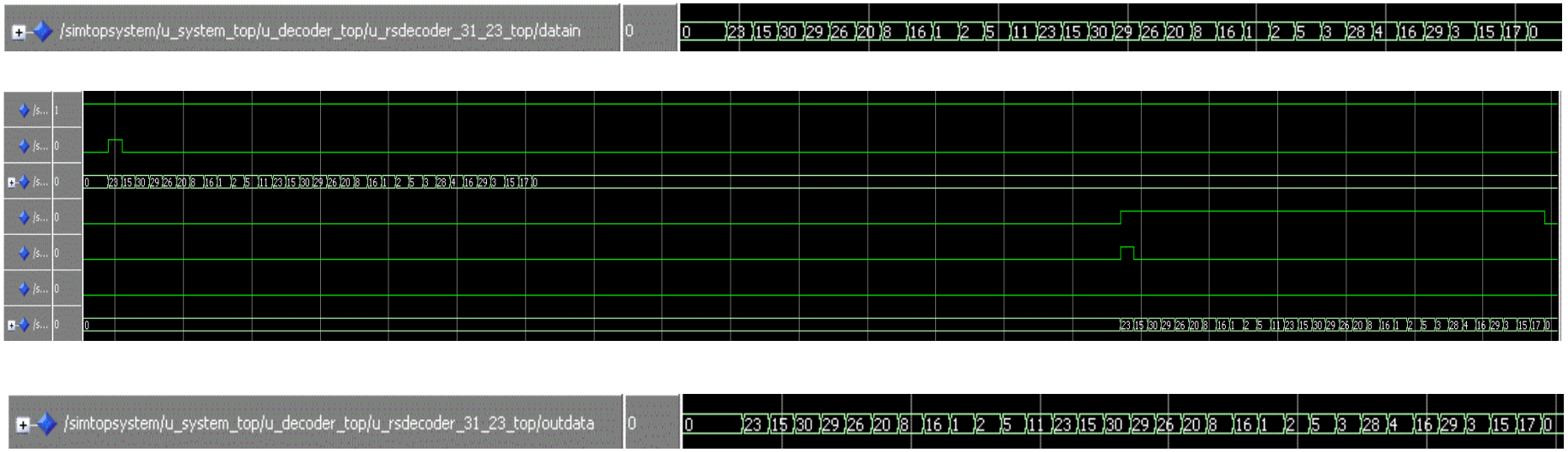


Figure 6.29 RS(31,23) decoder I/O waveform one codeword zoom

6.3. Summary

This chapter has presented the system source code. Each part of the system has been described using VHDL. The optimum Reed Solomon code parameters have been used to design the system. The simulation results showed that all the system parts are working correctly and agreed with the system theory.

ERASURE AND ERROR SIMULATION ENVIRONMENTS FOR THE DICODE PULSE POSITION MODULATION SYSTEM WITH REED SOLOMON CODE

7.1. Introduction

In this chapter, three test bench environments: erasure only, error only, and erasure and error are applied to the designed system, chapter 6. A Modelsim_Altera version (6.5b) software is to be used to simulate the system. The system has shown that it has the ability to detect, correct erasure, and error symbols when they overcome its limitation.

7.2. Erasure Only Test Bench

A VHDL test bench program, Appendix (4) section (10.4.1), was built to provide an environment where erasure errors can be injected into the system. According to equation (6.8), the designed system can correct up to 8 erasure errors only. Above this number, the system will fail to decode the original message. Figure 7.1 shows the flow chart for the erasure only test bench.

The following two test scenarios are likely to take place:

7.2.1. Correctable codeword

The number of erasure error symbols that are erased,compatible with the capacity of the decoder. In this test design, the number of erasure error symbols must be less or equal to 8 symbols per codeword. Figures 7.2 and 7.3 show the system input/output signals. Figure 7.2 shows the performance of the system when the number of erasure symbols equal 8 per codeword, while the figure 7.3 displays the system signals when the number of erasure symbols equal 5. In these figures, the fail output signal is logic 0. This means that the system has successfully decoded the original codeword. Erasures can be added or deleted by updating lines 264 and 265 inside the code.

7.2.2. Uncorrectable codeword

The number of erasure error symbols that are erased, greater than the capacity of the decoder to recover the original data. In this test design, the number of erasure symbols is greater than 8 symbols per codeword. Figure 7.4 shows the system input/output signals when the number of erasure symbols equal 9 per codeword. In this figure, the fail output signal is logic 1, which means that the system has failed to decode the original codeword. We can add or delete erasures by updating lines 264 and 265.

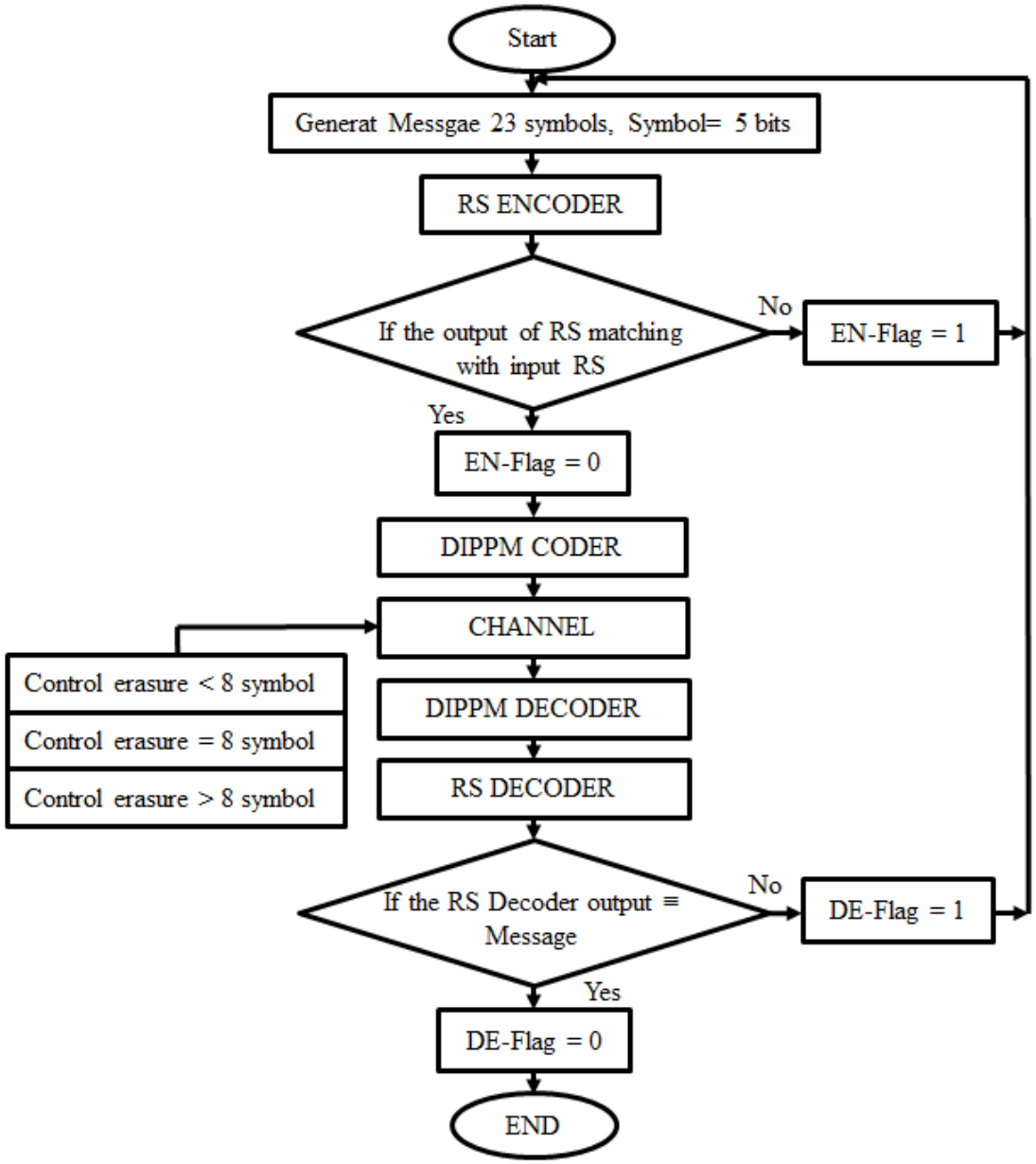


Figure 7.1 Erasure only test bench flowchart

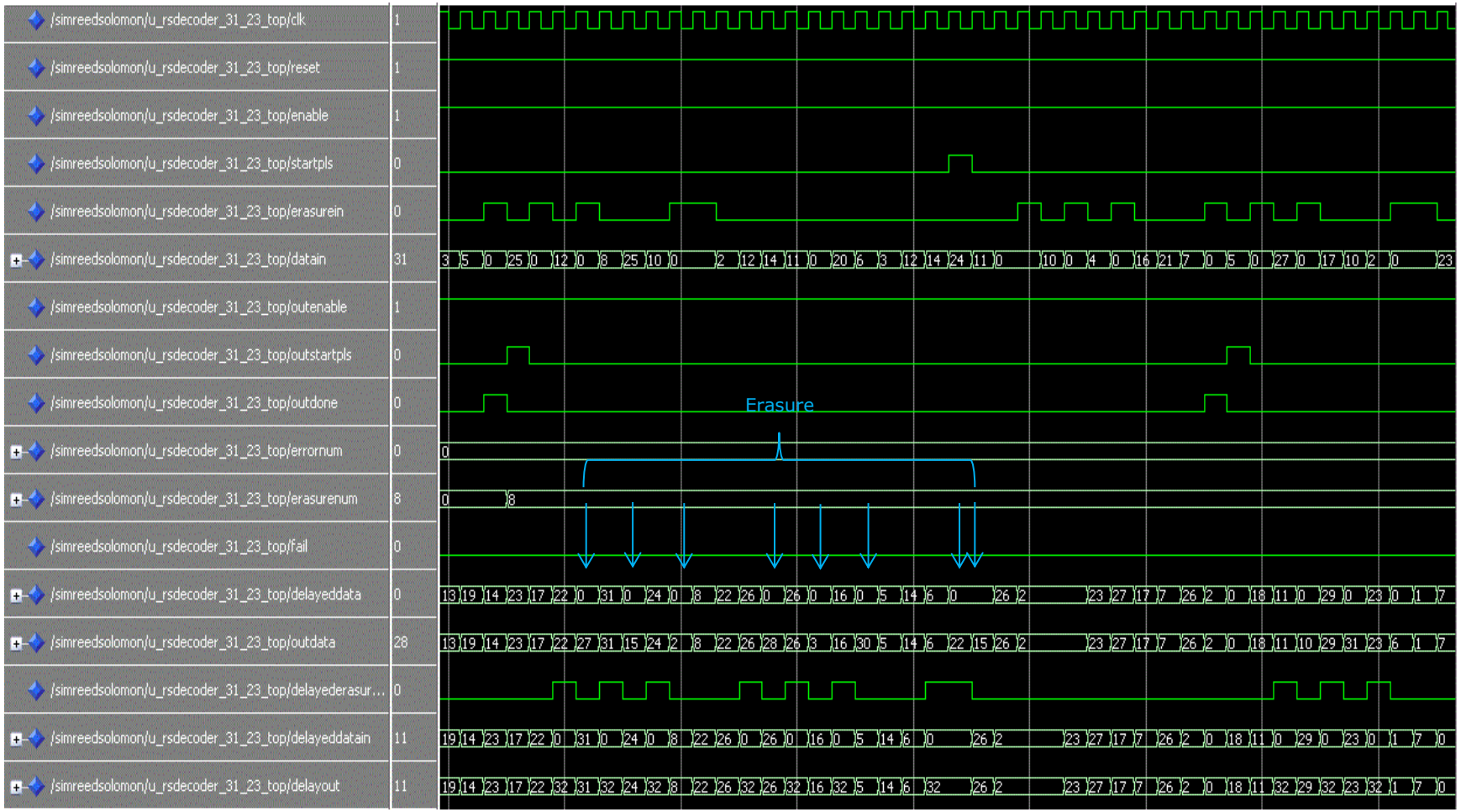


Figure 7.2 System input/output signals with 8 erasure symbols

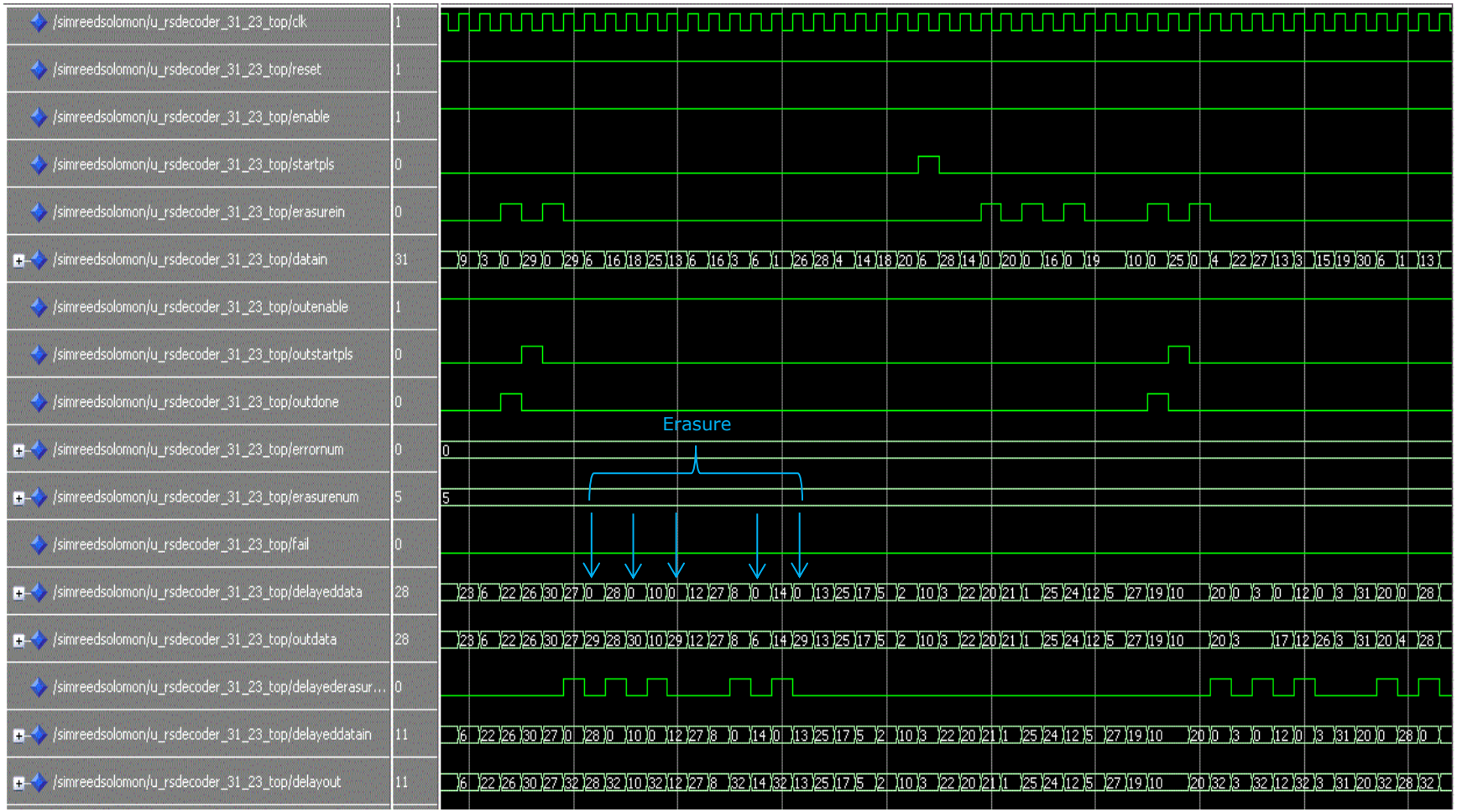


Figure 7.3 System input/output signals with 5 erasure symbols

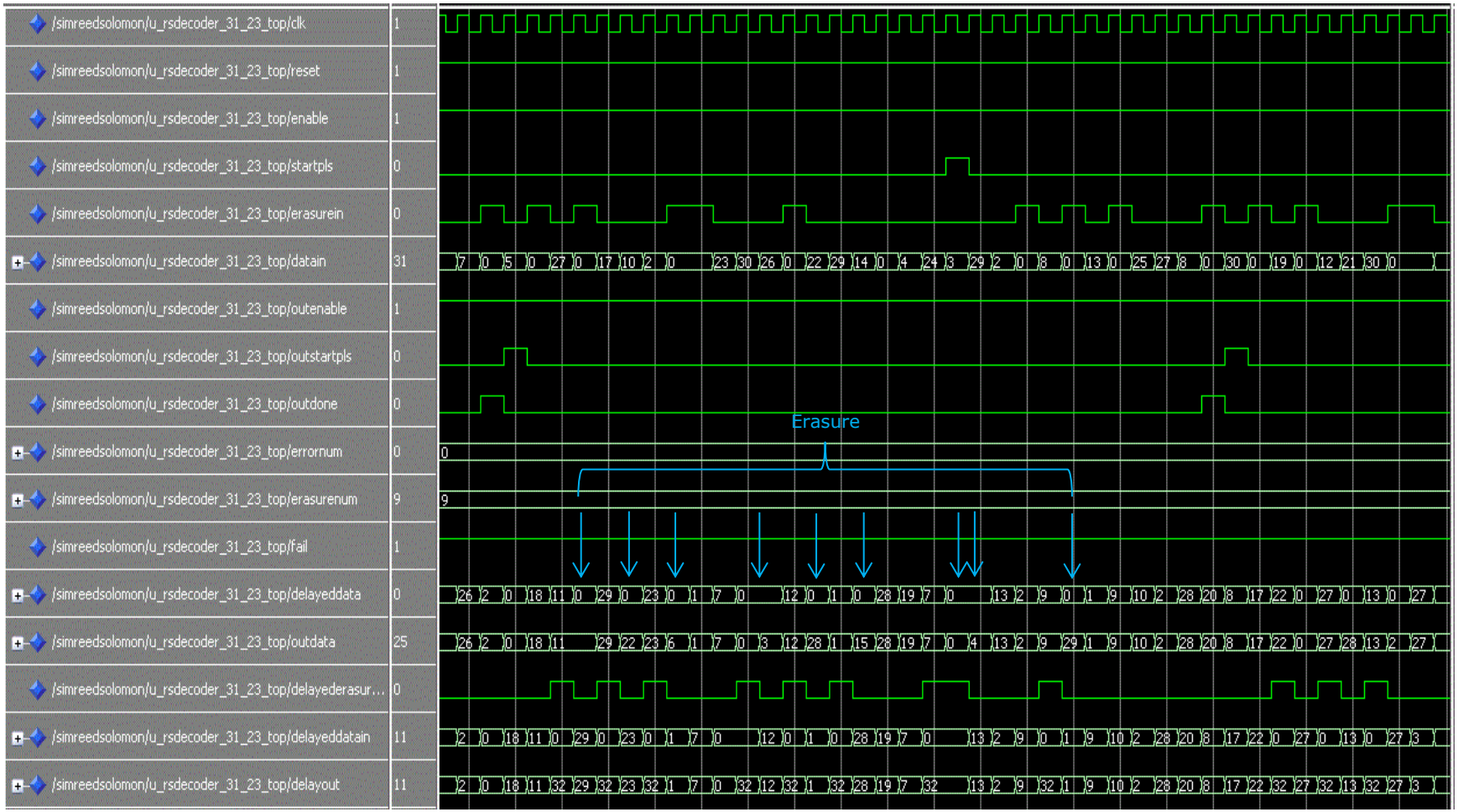


Figure 7.4 System input/output signals with 9 erasure symbols

7.3. Error Only Test Bench

A VHDL test bench program, Appendix (4) section (10.4.2), has been built to provide an environment where errors can be injected into the system. According to equation (6.7), the designed system has the ability to correct up to 4 errors only. Above this number, the system will fail to decode the original message. Figure 7.5 shows the flow chart for the error only test bench.The following two test scenarios are likely to take place:

7.3.1. Correctable codeword

The number of error symbols are compatible with the capacity of the decoder to recover the original data. In this test design, the number of error symbols must be less or equal to 4 symbols per codeword. Figures 7.6, & 7.7 show the system input/output signals. Figure 7.6 shows the performance of the system when the number of error symbols equal 4 per codeword. The figure 7.7 displays the system signals when the number of error symbols equal 2. In these figures, the fail output signal is logic 0. This means that the system has successfully decoded the original codeword. We can add or delete errors by updating line 264.

7.3.2. Uncorrectable codeword

The number of error symbols is greater than the capacity of the decoder to recover the original data. In this test design, the number of error symbols is greater than 4 symbols per codeword. Figure 7.8 shows the system input/output signals when the number of error symbols equal 5 per codeword. In this figure, the fail output signal is logic 1 which means that the system has failed to decode the original codeword. Errors can be added or deleted by updating line 264.

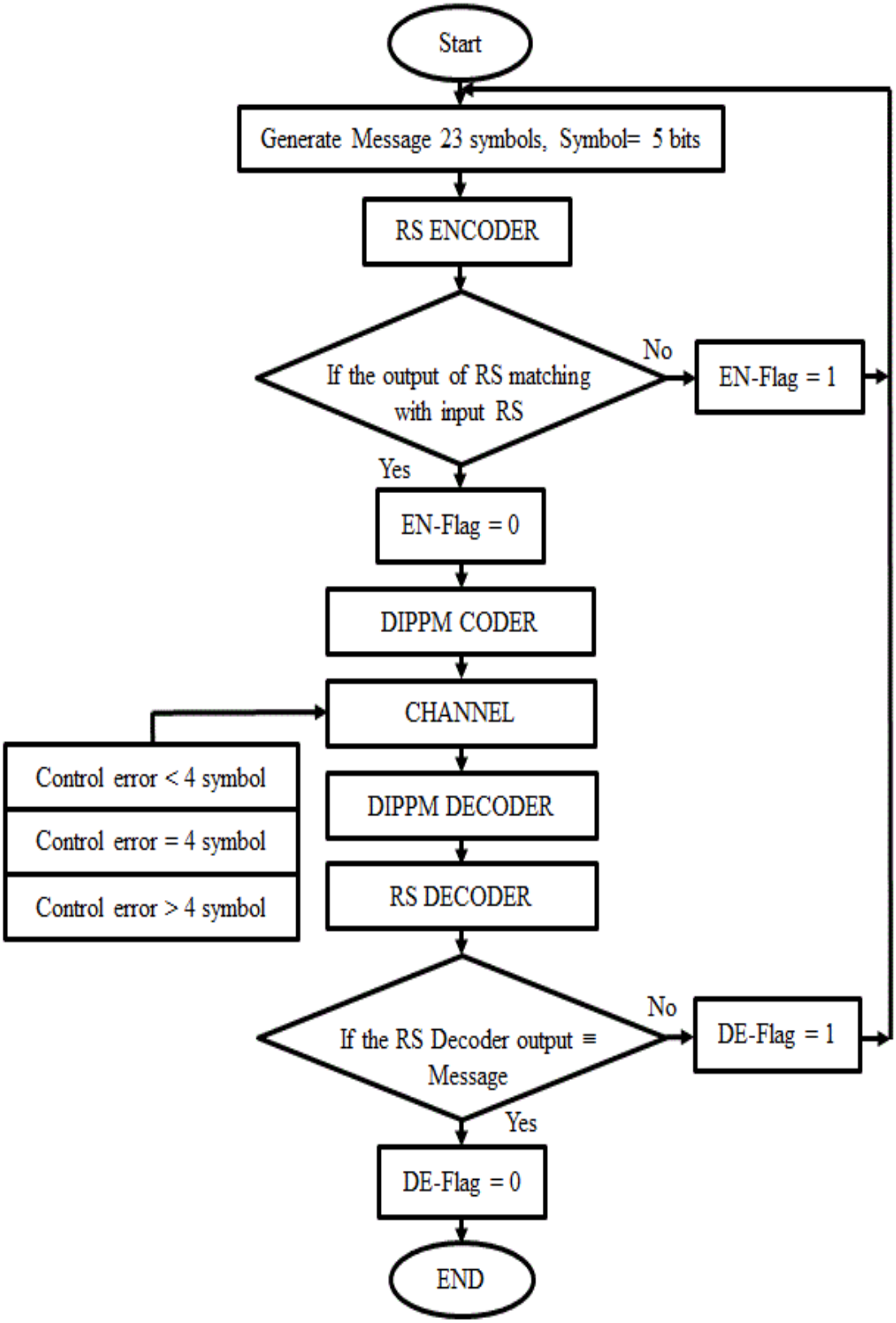


Figure 7.5 Error only test bench flowchart

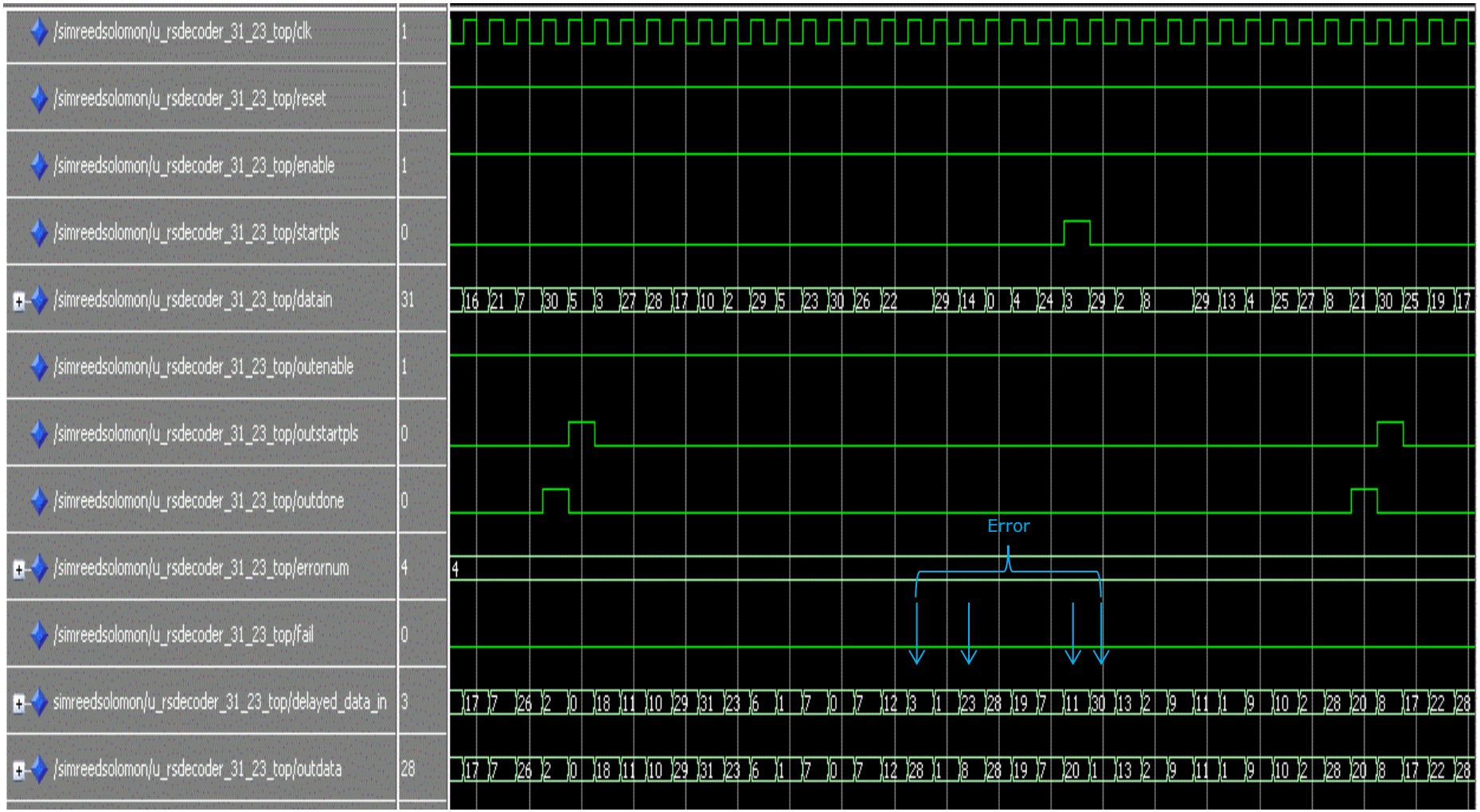


Figure 7.6 System input/output signals with 4 error symbols

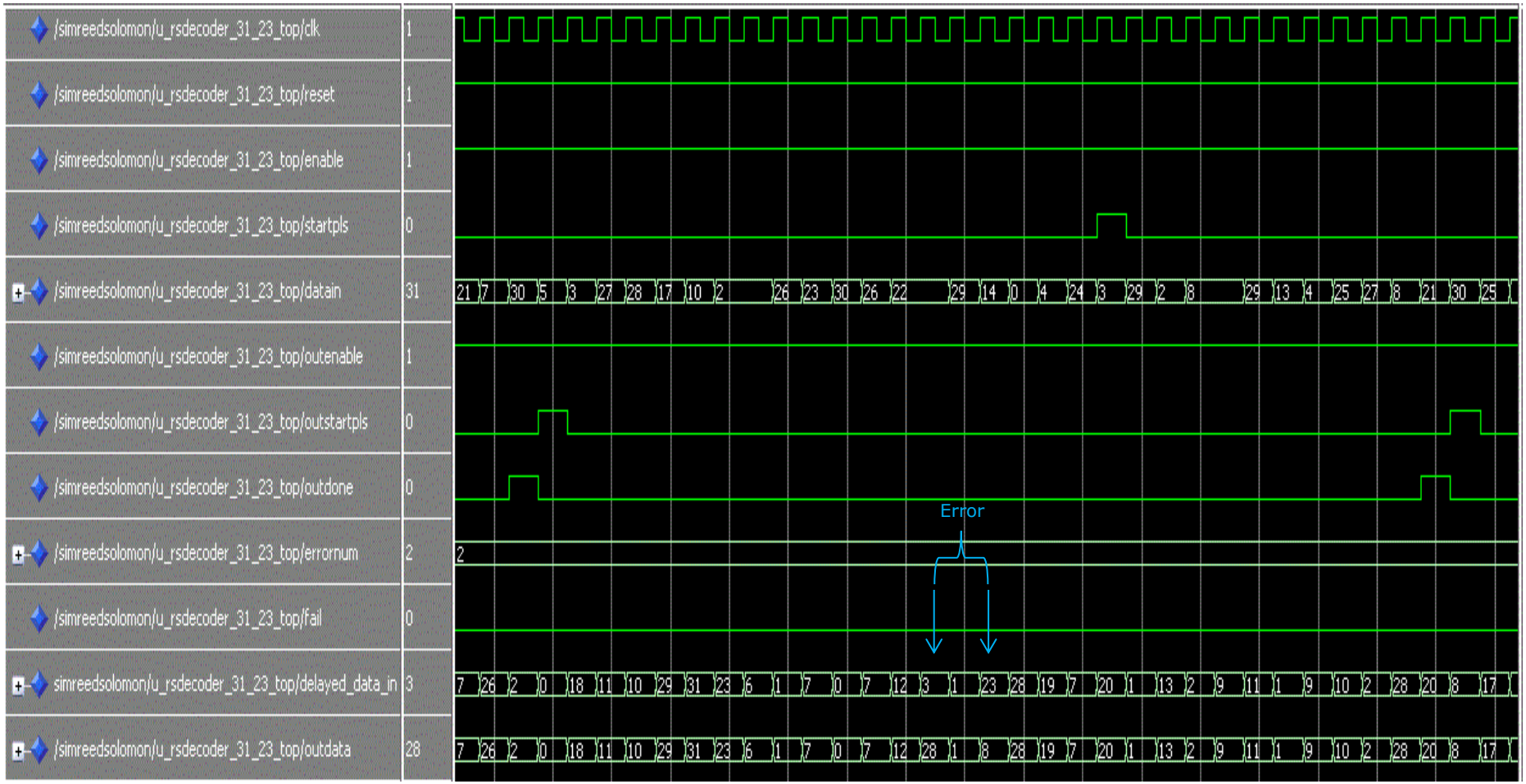


Figure 7.7 System input/output signals with 2 error symbols

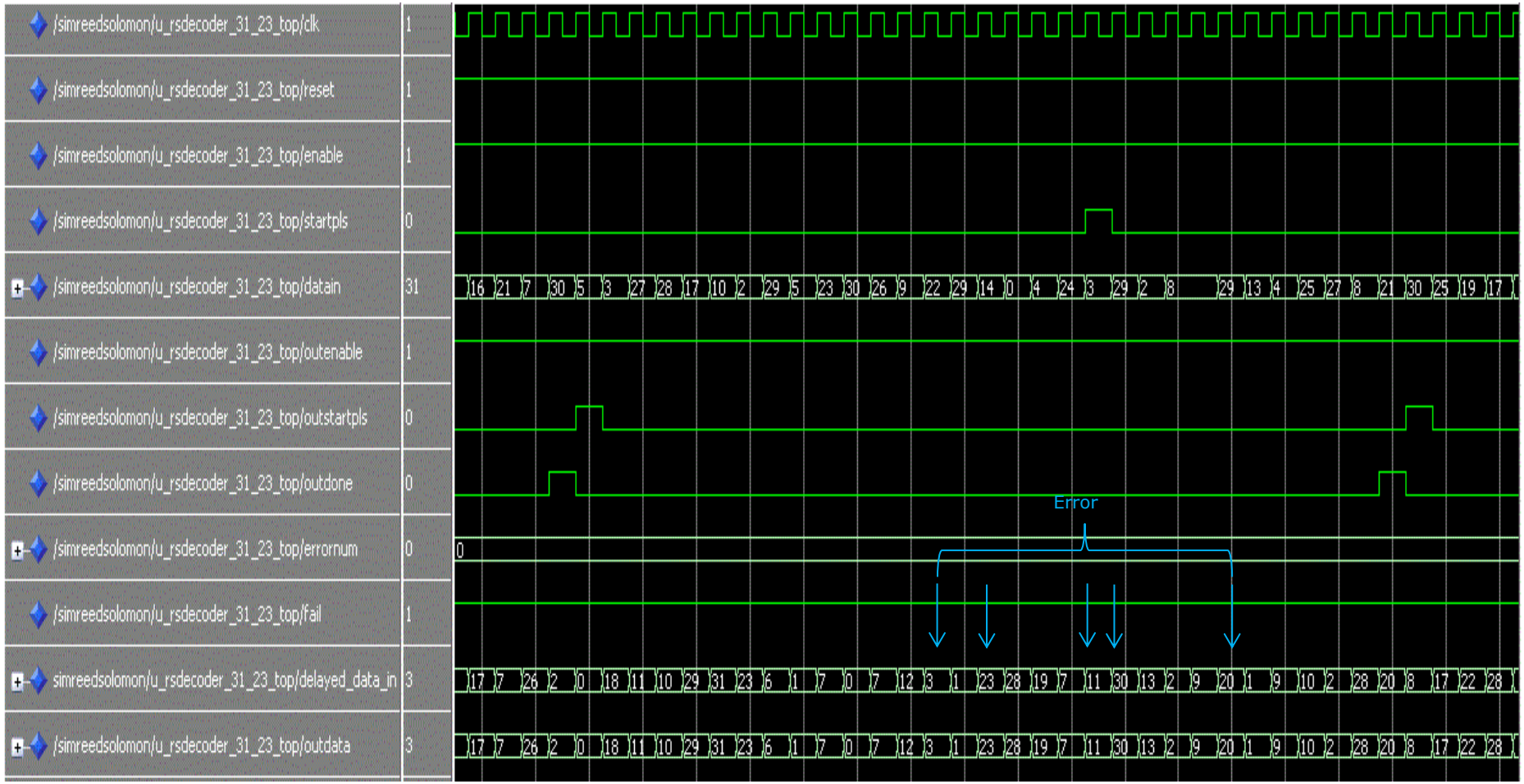


Figure 7.8 System input/output signals with 5 error symbols

7.4. Erasure and Error Test bench

A VHDL test bench program, Appendix (4) section (10.4.3), has been built to provide an environment to inject erasure and error symbols into the system. According to equation (6.10), the designed system has the ability to correct up to 4 erasure and 2 error symbols only. Above this number, the system will fail to decode the original message. Figure 7.9 shows the flow chart for the erasure only test bench.The following two test scenarios take place:

7.4.1. Correctable codeword

The number of error symbols are compatible with the capacity of the decoder to recover the original data. In this test design, the number of error symbols must be less or equal to 4 erasures and 2 error symbols per codeword. Figure 7.10 shows the system input/output signals. In this figure, the fail output signal is logic 0 which means that the system has successfully decoded the original codeword. Erasures and errors can be added or deleted by updating lines 261 & 265 inside the code.

7.4.2. Uncorrectable codeword

The number of error symbols is greater than the capacity of the decoder to recover the original data. In this test design, the number of error symbols is greater than 4 erasures or 2 error symbols per codeword. Figure 7.12 shows the system input/output signals when the number of error symbols equal 3 per codeword, while the figure 7.13 shows the system performance when the number of erasure symbols exceeds the system capability. In these figures, the fail output signal is logic 1 which means that the system has failed to decode the original codeword. We can add or delete erasures and errors by updating lines 261 & 265 inside the code.

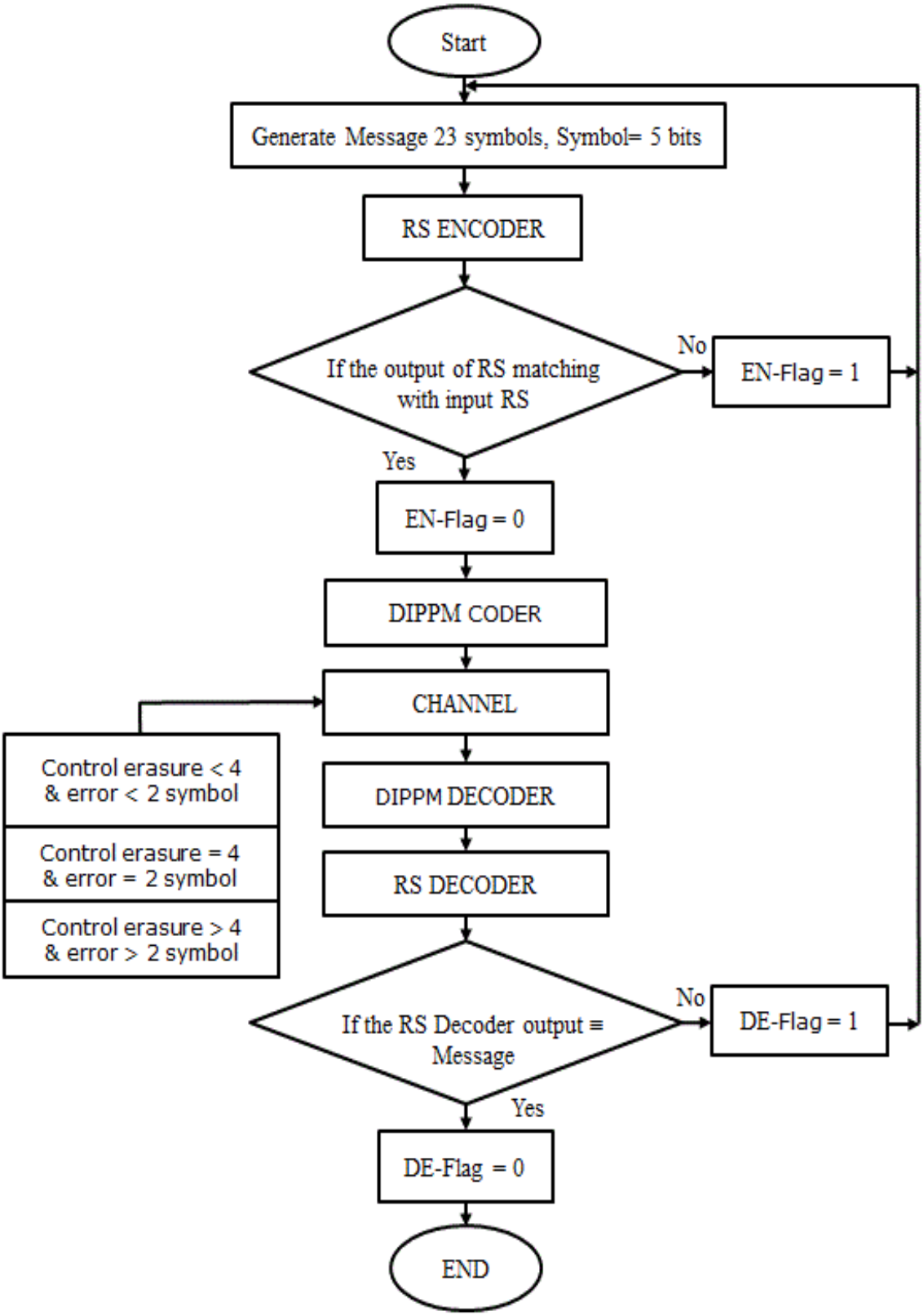


Figure 7.9 Erasure and error test bench flow chart

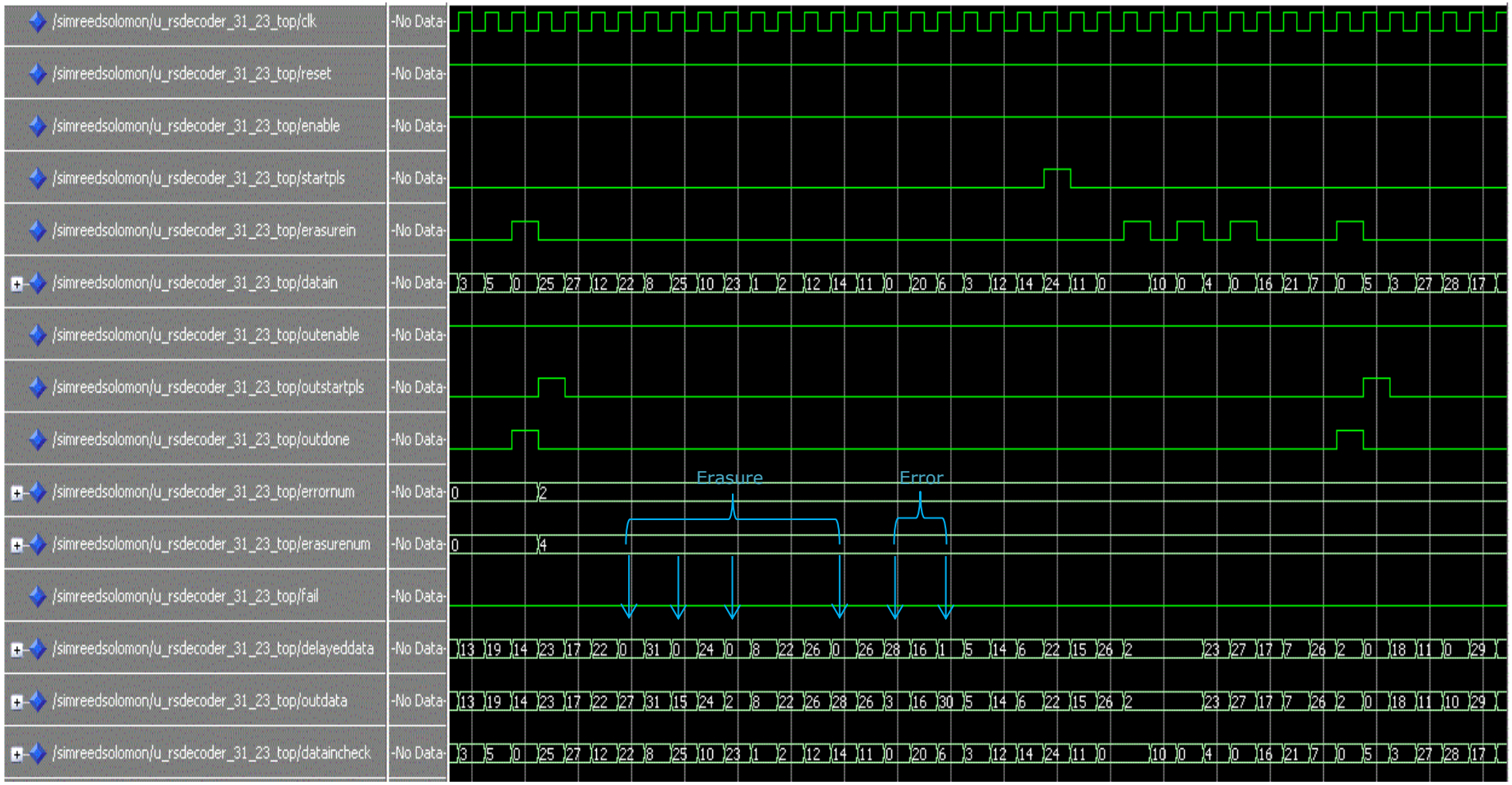


Figure 7.10 System input/output signals with 4 erasure and 2 error symbols

Figure 7.11 System original codeword

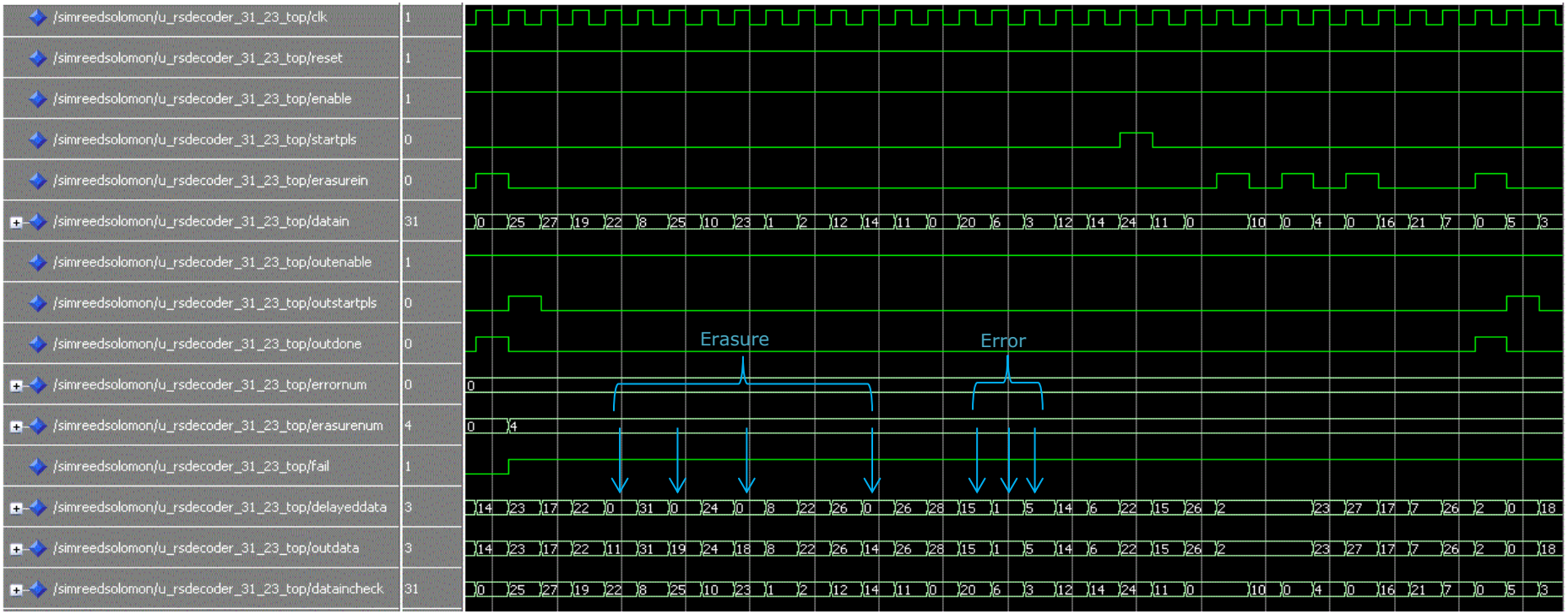


Figure 7.12 System input/output signals with 4 erasure and 3 error symbols

Figure 7.13 System input/output signals with 5 erasure and 2 error symbols

7.5. Summary

This chapter has presented three test bench environments, erasure only, error only, and erasure and error, on the designed system. Modelsim_Altera version (6.5b) software is used to simulate the system. The system has shown that it has the ability to detect and correct erasure and error symbols when they do not overcome its limitation.

DiPPM EMPLOYING RS CODE SYSTEM IMPLEMENTATION BY USING FPGA

8.1. Introduction

The VHDL source code for the DiPPM system employing RS codes was designed in chapter six. In this chapter, Altera Quartus II software and Cyclone III Field Programmable Gate Array-(FPGA) based DSP development board were utilised to implement the system (Altera, 2010). The optical transmitter, receiver, and comparator were designed and constructed. The SMA breakout cables interface was employed to get the digital input and output signals from the FPGA. Figure 8.1 shows the test bench equipment, while figure 8.2 shows the experimental design layout.

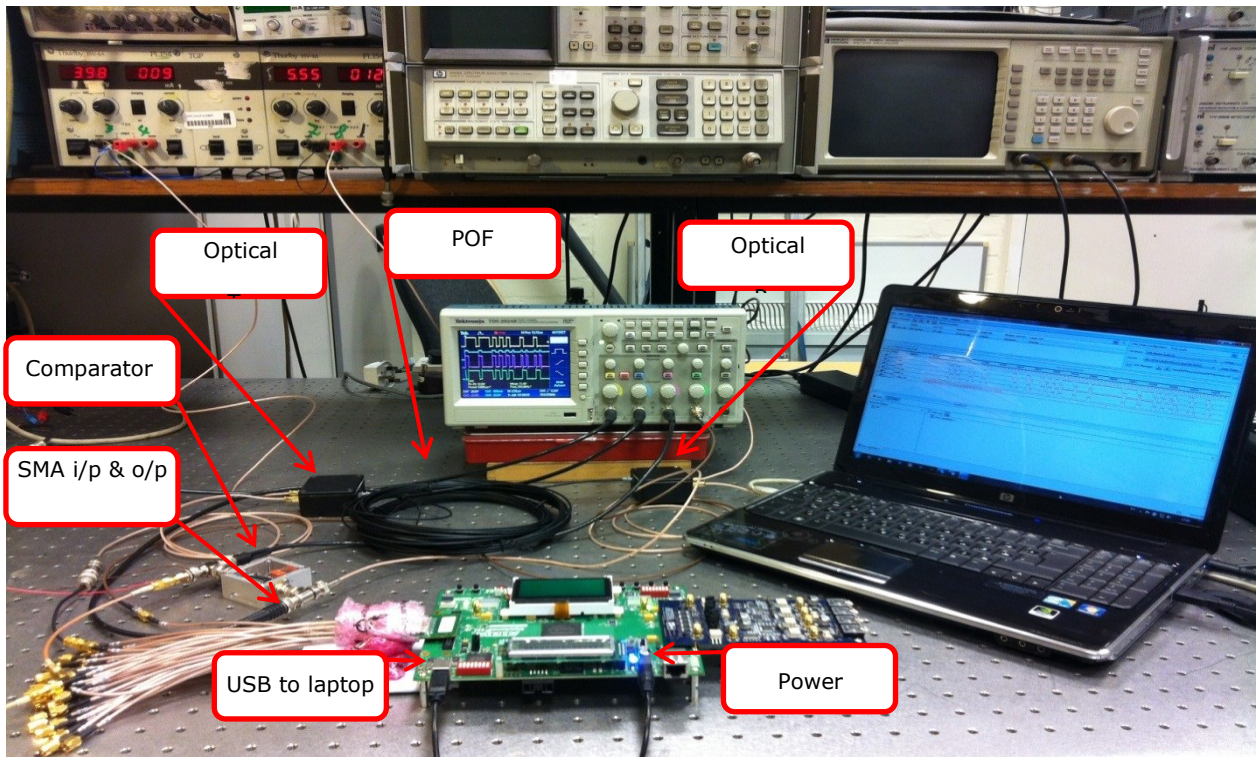


Figure 8.1 Laboratory testing facility of a design on cyclone III DSP board

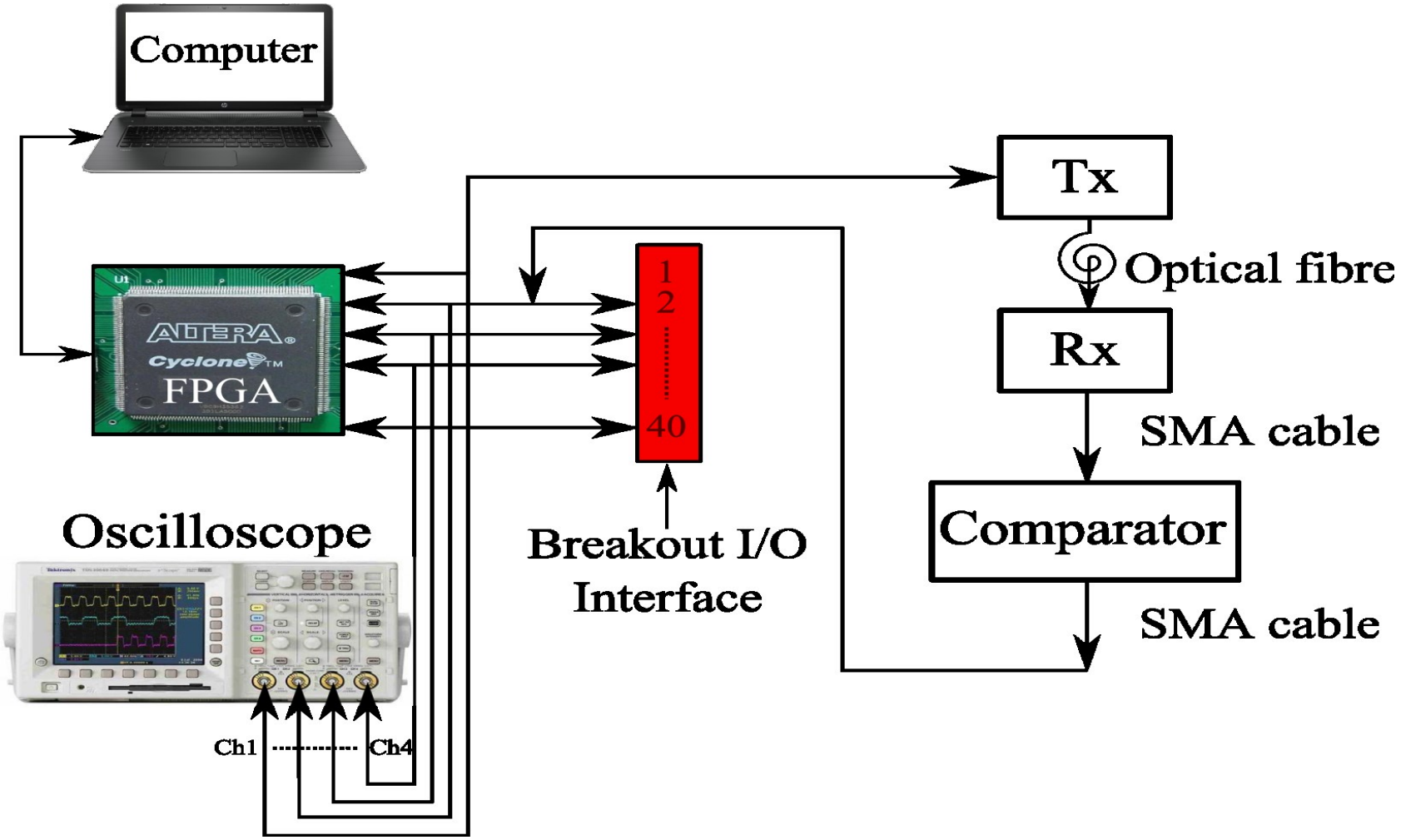


Figure 8.2 DiPPM with RS testing block diagram

8.2. Experiment Hardware Resources

This section specifies the detailed laboratory requirements of the experiment and gives justification for the level of resource requested.

8.2.1. Cyclone III Development Board

The Cyclone III development board presents hardware in the establishment and prototyping of the high-volume, the low-power, and feature-supported designs. In figure 8.3, the block diagram for the Cyclone III development board is shown. This Cyclone III development board can be used for wireless, video, and image processing, alongside high-bandwidth parallel processing systems. For the Cyclone III development board, various features are apparent, including (Altera, 2010):

- A high logic density for securing more functions.
- An embedded memory to support high-bandwidth designs.
- Expandability via two Altera High-Speed Mezzanine Card (HSMC) connectors.
- 256-MB of dual channel DDR2 SDRAM alongside a 72-bit data width.
- Able to support high-speed external memory systems as well as dual-channel DDR SDRAM and low-power SRAM.
- Four user push-button switches.
- Eight user LEDs.
- Power consumption display.

For the Cyclone III development board, different benefits are experienced:

- A unique combination of low-cost, low-power Cyclone III FPGA that assists in securing high-volume and memory-intensive standards.
- Most efficient multiplier-to-logic ratio FPGA in the industry.
- Lowest cost but density-and power-optimized FPGA.
- Quartus II development software's power optimization function.

8.2.1.1. Board Component Blocks

In figure 8.4, a top view of the Cyclone III development board is presented. The board includes different major component elements (Altera, 2010):

- 780-pin Altera Cyclone III EP3C120 FPGA.
- On-board memory.
- FPGA configuration circuitry.
- On-board clocking oscillators to ensure Cyclone III device user logic.

- SMA connector to support external clock input and output.
- Eight user LEDs.
- One user reset push-button (CPU reset).
- Four general user push-buttons.
- One system reset push-button (user configuration).
- One factory push-button switch (factory configuration).
- One MAX control DIP switch.
- One JTAG control switch.
- Eight user DIP switches.
- 128 × 64 graphics LCD, and 16 × 2 line character LCD.
- Power supply, 14 V – 20 V DC input.

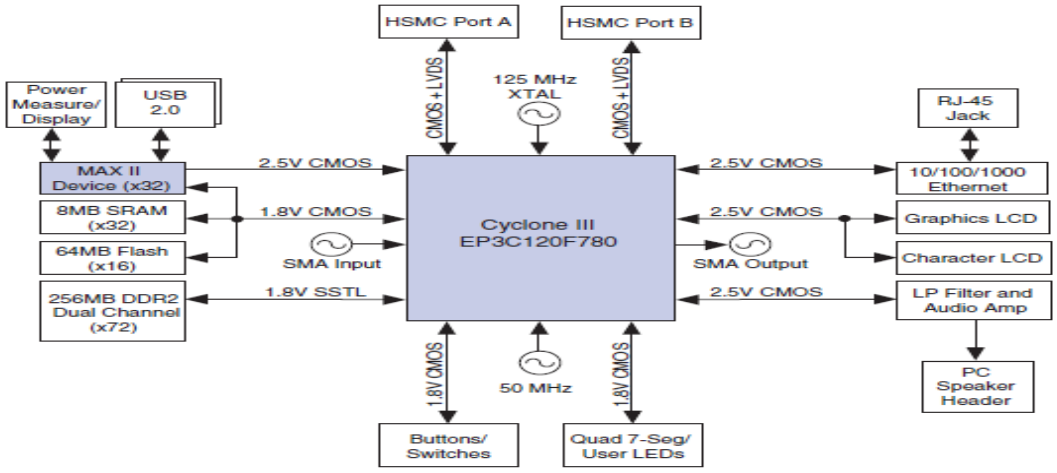


Figure 8.3 Cyclone III Development Board Block Diagram (Altera, 2010)

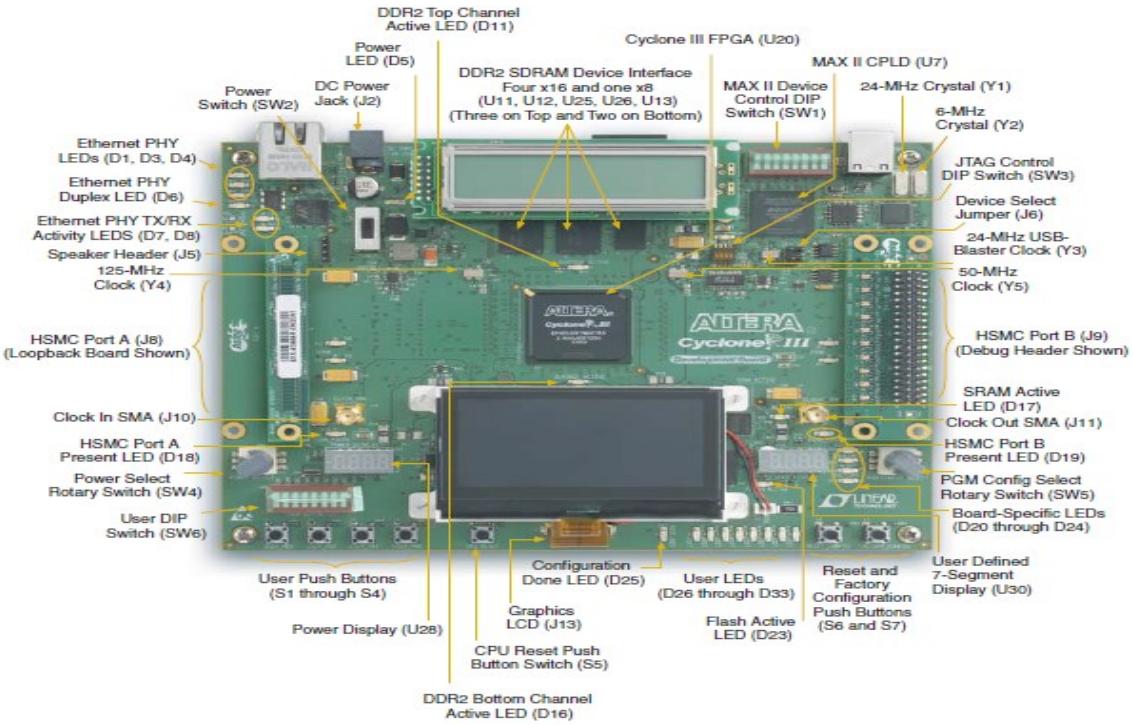


Figure 8.4 Top View of the Cyclone III Development Board (Altera, 2010)

8.2.2. SMA Breakout Cables

The Cyclone III development board includes two HSMC (High Speed Mezzanine Cards) interfaces, known as Port A and Port B. These interfaces enhance the single-ended and differential signalling. The connector part number is Samtec ASP-122953-01. The HSMC interface also supports JTAG, SMBus, clock outputs and inputs, including power for compatible HSMC cards. The HSMC is an Altera-developed system which allows users to improve the coverage of the development board with the use of the daughter cards (HSMC cards) (Altera, 2009).

The HSMC connector includes 172 total pins, as well as 120 signal pins, 39 power pins, and 13 ground pins. The ground pins are seen between two rows of signal and power pins, being both the shield and the reference. There are three banks in this system, including Bank 1, Bank 2 and Bank 3 (Altera, 2009).

The Cyclone III development board does not support Bank 1 transceiver signals meant for clock-data-recover (CDR) systems including PCI Express and Rapid I/O. These 32 pins are allowed to float. Banks 2 and 3 are fully functional and can be applied in two different configurations, see figure 8.5 (Altera, 2009).

Altera and Samtec, Appendix (6) section (10.6.1), created a breakout cable, figure 8.6, for use in managing a single bank of the 3-bank HSMC connector to SMA cables. Differential pins are used in this case. While the use can be challenging, the cable has high signal integrity and has a completely flexible connection system by applying the SMA connectors.

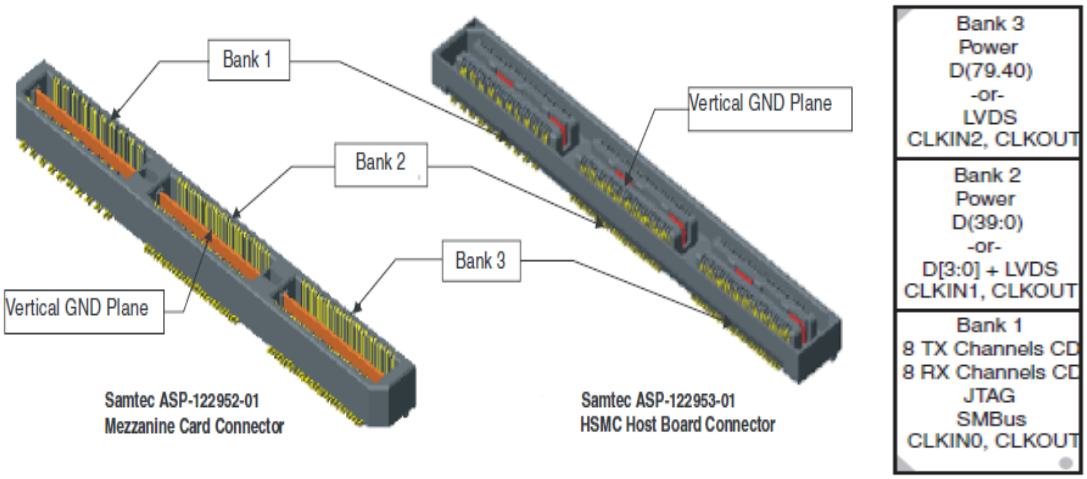


Figure 8.5 HSMC Connectors (Altera, 2009)

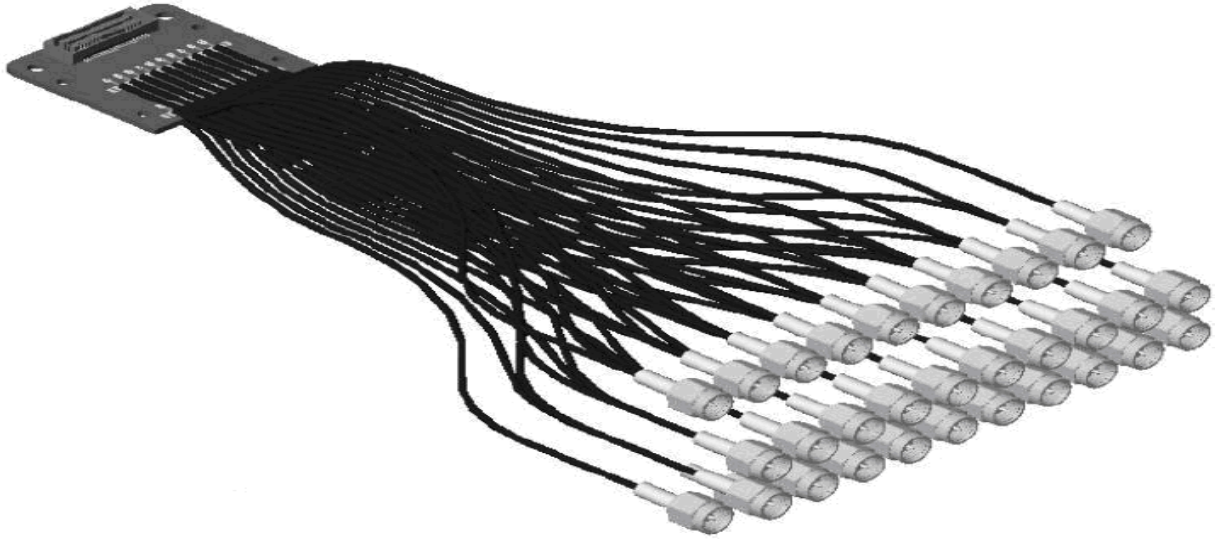


Figure 8.6 SMA Breakout Cable (Altera, 2009)

8.2.3. Optical Fibre Communication System

The optical fibre communication system is similar to any other type of communication system in relation to the principle parts which form the system. It has a source of light which is sent into a fibre as a channel in order to deliver it to an optical receiver which then converts the modulated light into an electric signal (Senior & Jamro, 2009).

8.2.3.1. Optical Transmitter

The optical transmitter is a light source whose output is modulated using on-off keying. This technique can be achieved by varying the drive current in the transmitter circuit. This then causes a proportional change in the transmitter output optical power (Sibley, 1995, Senior, 2009).

The main part of the transmitter is a semiconductor diode which could be a light emitting diode or a laser semiconductor. It is a forward-biased diode where the output light intensity is coupling with the semiconductor diameter using an optical fibre (Senior & Jamro, 2009; Sibley, 1995).

Figure 8.7 shows the deigned optical transmitter system. RC-LED and HFRB-1527Z was used to provide high optical power which could support a long length of POF cable. The HFRB-1527Z transmitter operates at a signal rate from 1 to 125 megabaud over 1 mm

diameter plastic optical fibre or 200 µm diameter hard clad silica glass optical fibre, see Appendix (6) section (10.6.2).

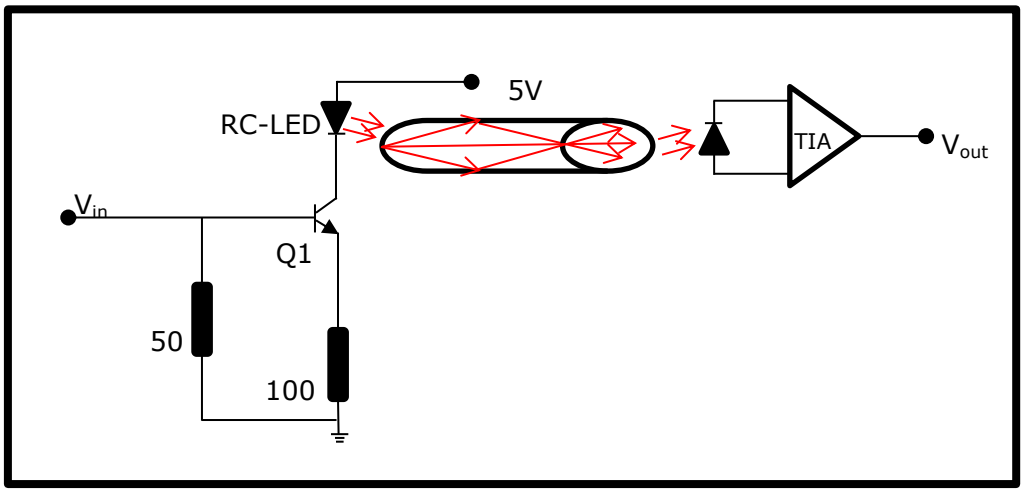


Figure 8.7 Optical Transmitter Circuit

8.2.3.2. Optical Receiver

The main part of the optical receiver is the photo detector which acts as demodulator converting the optical signal into an electrical signal. There are minimum performance requirements which the photo detector should have. There are many types of photo detectors. They differ in term of operation and in terms of materials, but the selection of these types must be decided according to the application requirements (Senior & Jamro, 2009; Sibley, 1995).

The HFBR-2526Z, Appendix (6) section (10.6.2), optical receiver was used in the tests. The receiver contains a PIN photodiode with integrated transimpedance amplifier. This type of optical receiver is suitable for POF application providing high bandwidth and high transimpedance gain. Figure 8.8 displays the power supply filter which was used to provide the input voltage to the receiver as well as its biases in the PIN photodiode (inverse biased).

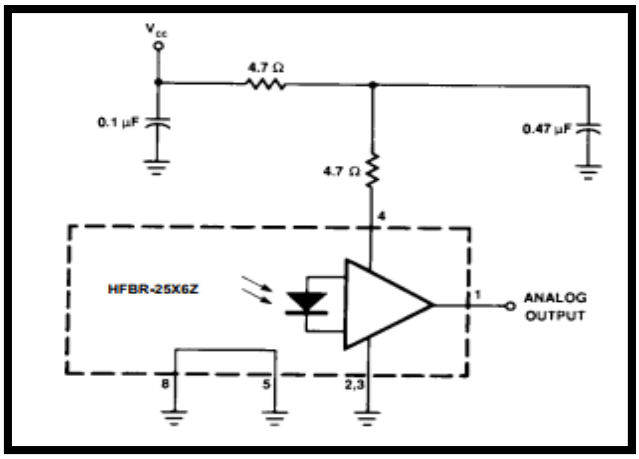


Figure 8.8 Recommended Filter Circuit For Optical Reciever

8.2.3.3. The Comparator

A comparator circuit differentiates between two voltages and outputs from 1 or 0 in order to determine which is bigger. These are usually applied in order to assess whether an input has secured a predetermined value. In most instances, a dedicated comparator IC is used. However, op-amps may be applied as another option (Senior & Jamro, 2009; Sibley, 1995).

The MAX941, Appendix (6) section (10.6.3), was employed as a comparator. Figure 8.9 shows a MAX941 comparator connected as a simple line transceiver. The output is a clean square wave signal at the input frequency. The output amplitude is equal to V+. See figure 8.10 for pin configurations.

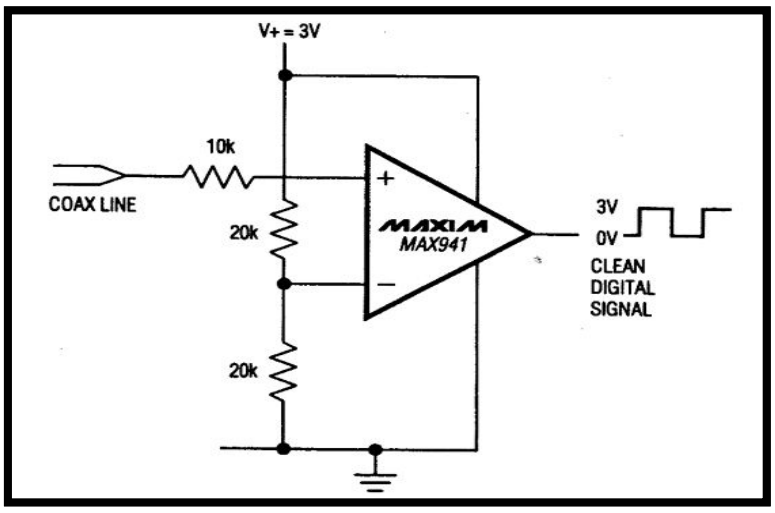


Figure 8.9 The Comparator Circuit

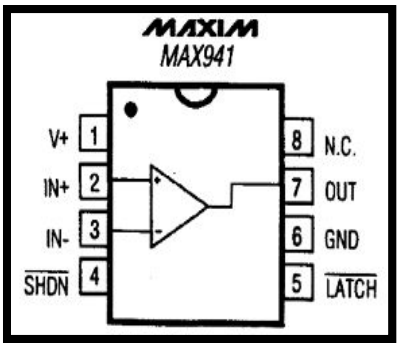


Figure 8.10 MAX941 Pin Confugarations

8.2.3.4. Plastic Optical Fibre (POF)

The plastic optical fibre is composed of organic polymers for the core and cladding. It is easily used as well as cheap to manufacture. However, it is limited in terms of the infrared, which makes gives the POF limited applications. In recent years, the POF has been made from polymethyl methacrylate and fluorinated acrylic (PMMA). Such a type of fibre has losses of 110dB/km within the visible wavelength. The loss mechanism of POF is likened to a glass fibre because of absorption and because of Rayleigh scattering which are associated to the density changes and anisotropic structure of polymers (Senior & Jamro, 2009; Sibley, 1995).

A 10 meter of HFBR-R, figure 8.11, single mode plastic optical fibre was used to transfer data between the optical transmitter and receiver. For further details and specifications of this type of POF, see Appendix (6) section (10.6.4).

Figure 8.11 Single Mode Plastic Optical Fibre

8.3. Test One: Implementation of DiPPM System

In this test, the optical dicode pulse position system was implemented for the first time. The DiPPM source code was used to build the DiPPM coder and decoder entities. The optical DiPPM system was fully designed, figure 8.12, using Altera Quartus II version 9.1software and downloaded onto the development board from a laptop via a USB cable as shown in figure 8.1. A SMA breakout cables interface was employed to get the input and output signals from the FPGA. The Altera Quartus software offers many MegaCore functions to parameterize the function that the engineer requires. The PPL MegaCore function wizard allows the setting of the clock frequency.

In the transmitter side, a 50 MHz internal clock was fed to the PLL to generate 1 MHz clock frequency. A PRBS was built to produce a random parallel of 23 symbols as a PCM sequence. The bridgecoder entity was used to convert the parallel PCM sequence to serial form. The output pin of the DiPPM coder was connected to the optical transmitter. A 10 meter plastic optical fibre (POF) was used as a channel to transfer the data.

In the receiver side, a comparator was designed as a first stage to receive the signal from the optical receiver. Then, the received signal was passed to the DiPPM decoder in order to regenerate the original PCM sequence.

Figure 8.13 shows in the first channel (Yallow Signal) the PRBS sequence output of the bridgecoder entity, while the DiPPM sequence is seen in channel two (Blue Signal). Channels two and three (Blue and Purple) compare the received DiPPM sequence before and after the comparator. The decoded PRBS sequence is shown in channel four (Green Signal). The designed optical DiPPM system, figure 8.12, produces the correct DiPPM through FPGA and correctly decodes it back to the original PCM with only a half clock cycle delay.

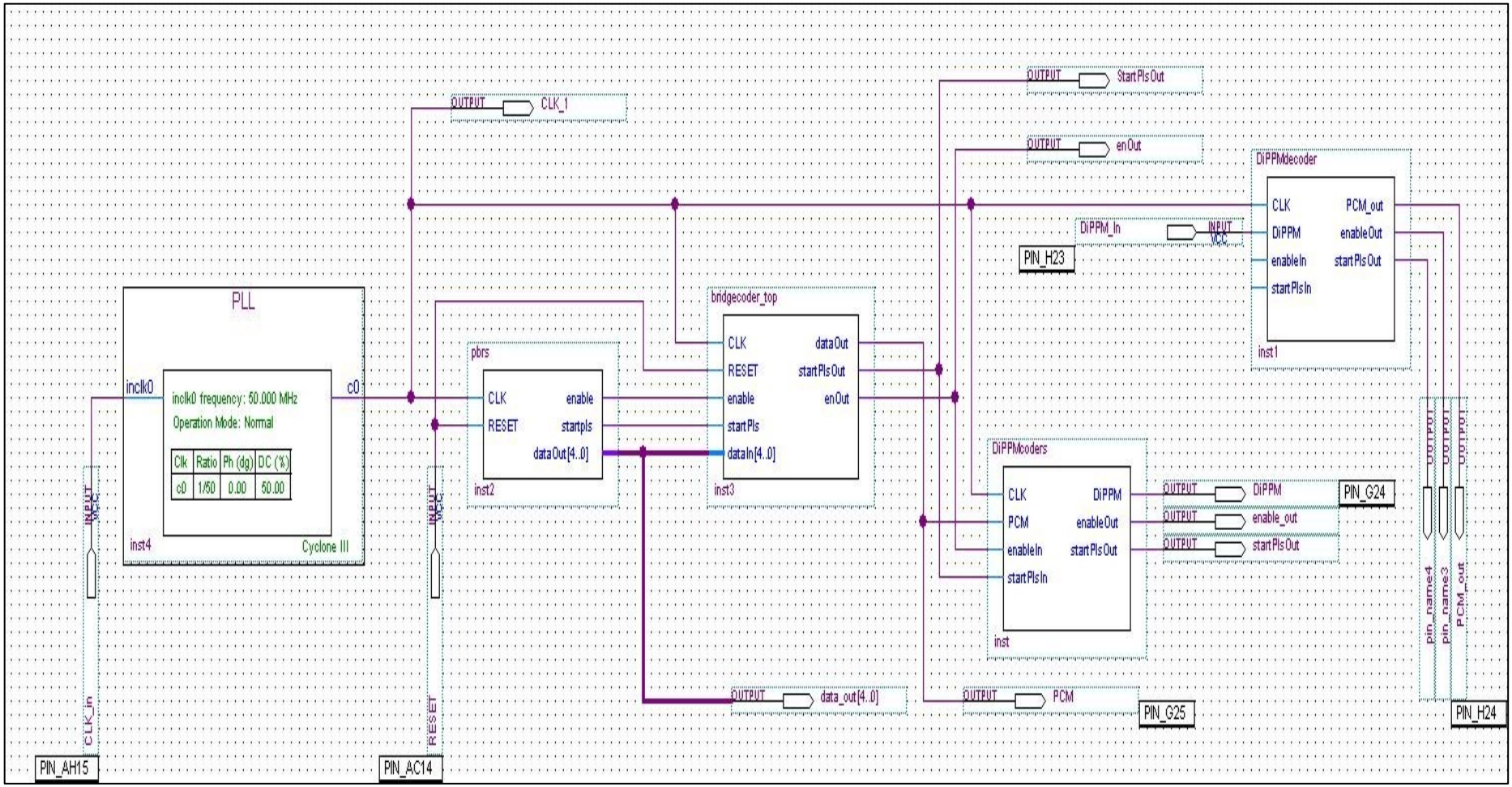


Figure 8.12 DiPPM system design on Altera Quartus II

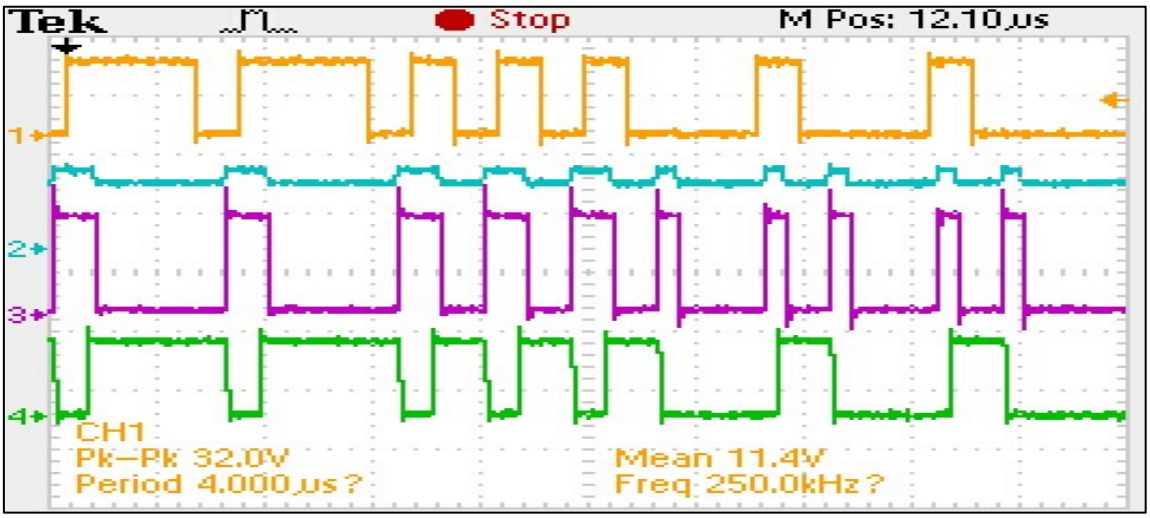


Figure 8.13 The DiPPM Optical System Waveform

8.4. Test Two Implementation of DiPPM with RS code System

The DiPPM with (31, 23) RS code system was implemented for the first time using Altera Quartus II software, figure 8.14. A 50 MHz on board clock oscillator was used to provide input clock to the system and one of the eight IP switches was employed to provide a RESET signal to the system. The transmitter side was divided into five stages, which are: PLL to provide 1 MHz clock, PRBS generator, (31,23) RS coder, Bridgecoder to convert the RS output from parallel data to serial, and the DiPPM coder. The output of the transmitter is sent via a designed channel entity. On the receiver side, the received data process in four stages, including the DiPPM decoder, bridgedecoder to alter the DiPPM decoder output data from serial to parallel form, and (31, 23) the RS decoder. The output waveform of each stage of the transmitter and receiver was gathered and compared to the simulation results (chapter 6).

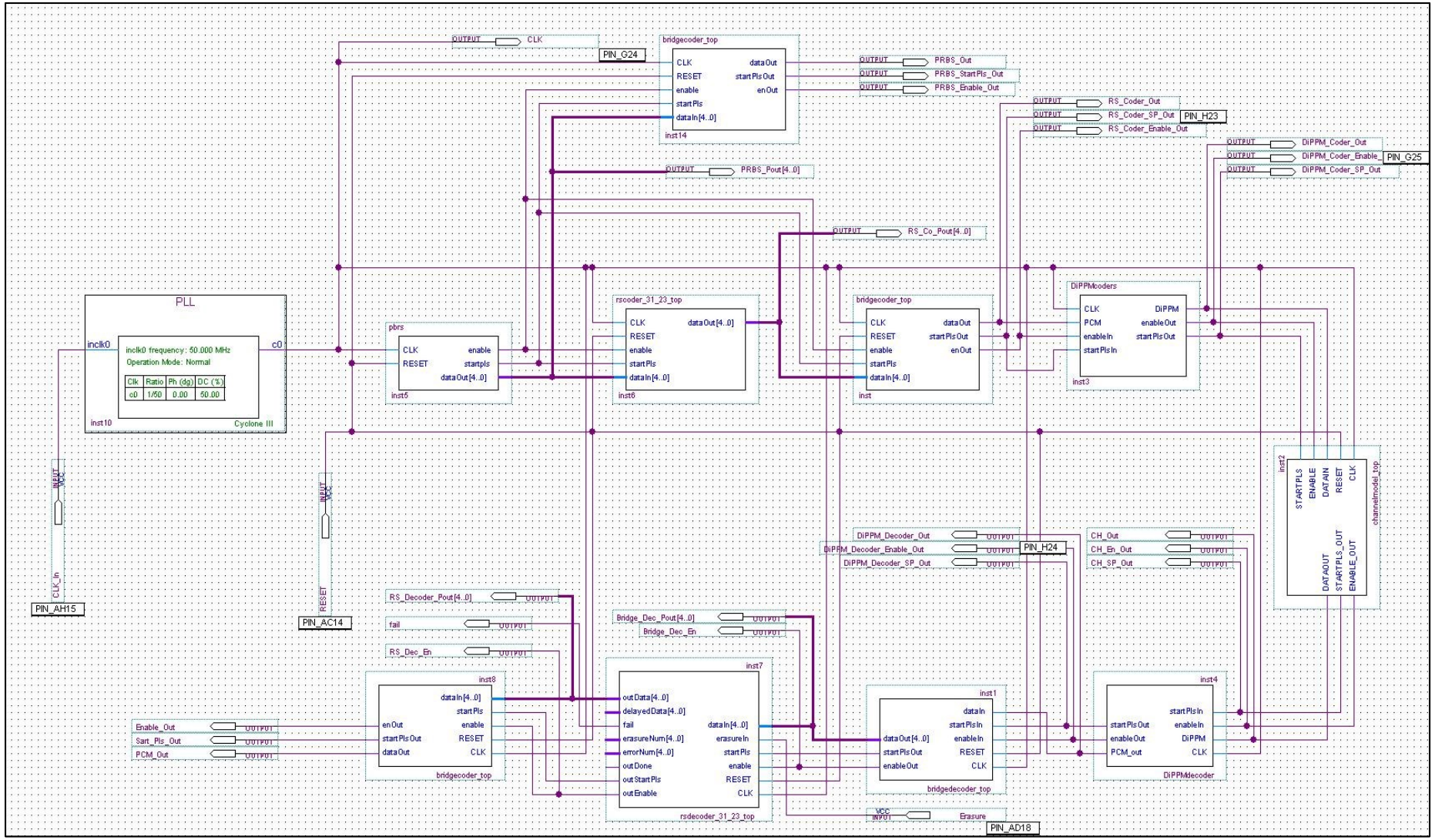


Figure 8.14 The DiPPM with (31,23) RS Code System

8.4.1. PRBS Entity

The PRBS generator waveform output was collected by using an oscilloscope and using the Signal Tap Analyzer (STA) tool that Altera provided with the Quartus software which allowed the reading of the parallel data. Figures 8.15 and 8.16 show the PRBS waveform output by using the oscilloscope for multi codewords and single codeword respectively. Figures 8.17 and 8.18 on the other hand, display the PRBS waveform output by using the STA for multi codewords and single codeword, including the parallel form.

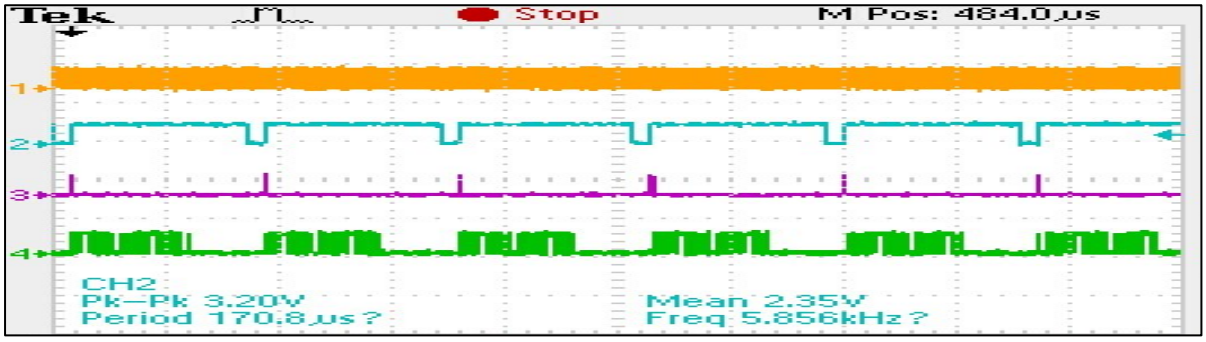


Figure 8.15 The PRBS Waveform Output Multi Codewords

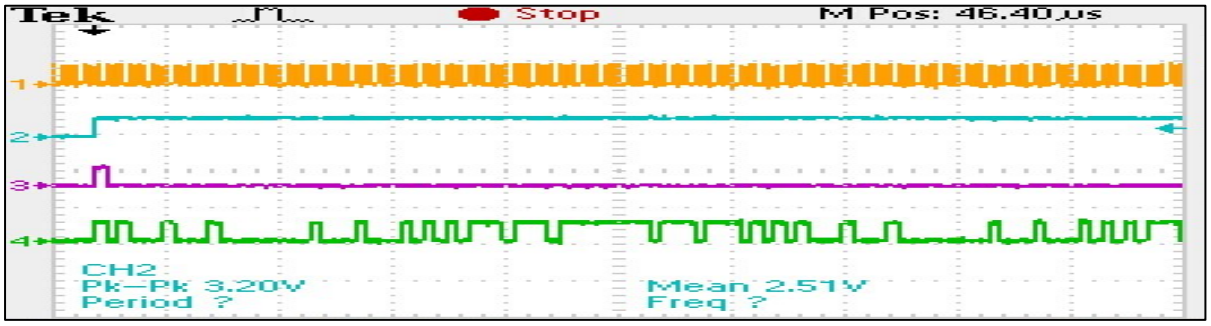


Figure 8.16 Figure 8.15 The PRBS Waveform Output Single Codeword

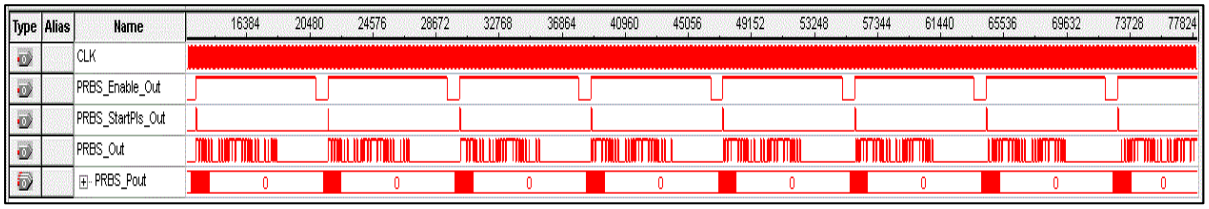


Figure 8.17 The PRBS Waveform Output Multi Codewords Using the STA

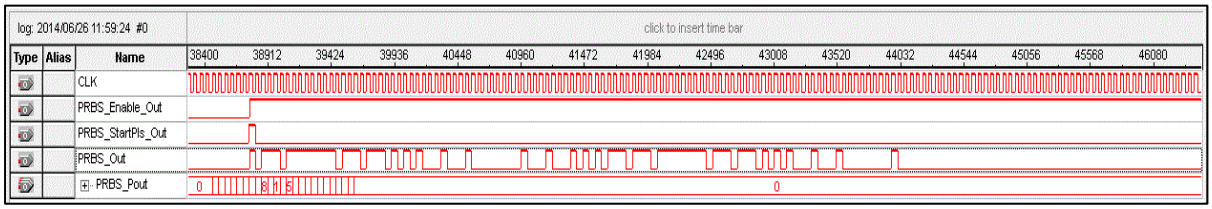


Figure 8.18 The PRBS Waveform Output Single Codeword Using the STA

8.4.2. RS Coder Entity

The (31,23) RS coder waveform output is displayed in figures 8.19, 8.20, 8.21, and 8.22. Figures 8.19 and 8.20 show the RS coder serial waveform in both multi and single codeword mode. The serial data was collected after converting the parallel RS coder output by using the bridgecoder. Figures 8.21 and 8.22 present the output waveform for the RS coder in both serial and parallel style.

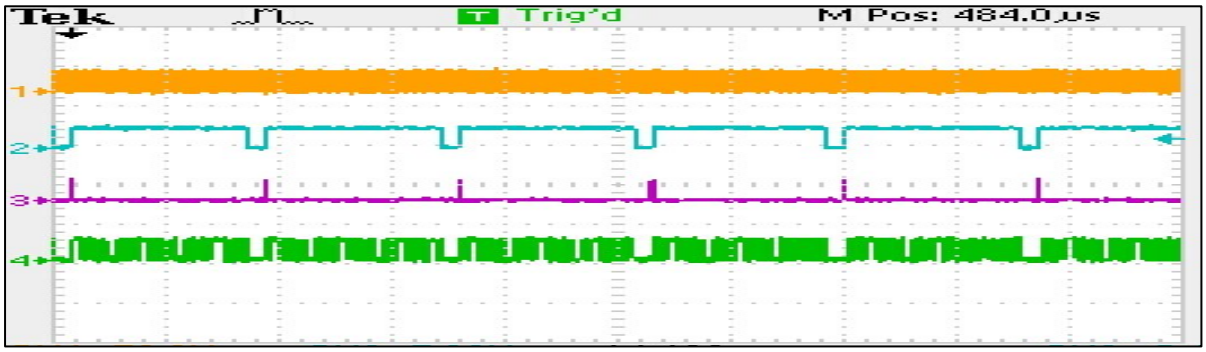


Figure 8.19 The (31,23) RS Coder Waveform Output Multi Codewords

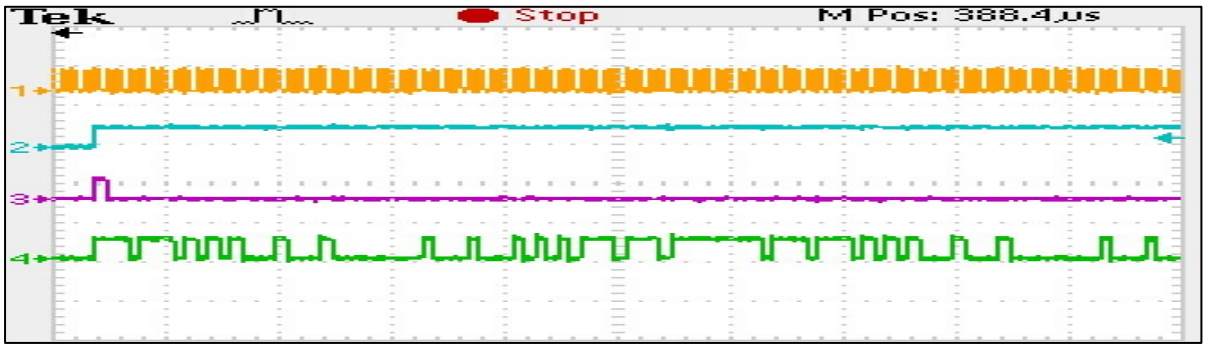


Figure 8.20 The (31,23) RS Coder Waveform Output Single Codeword

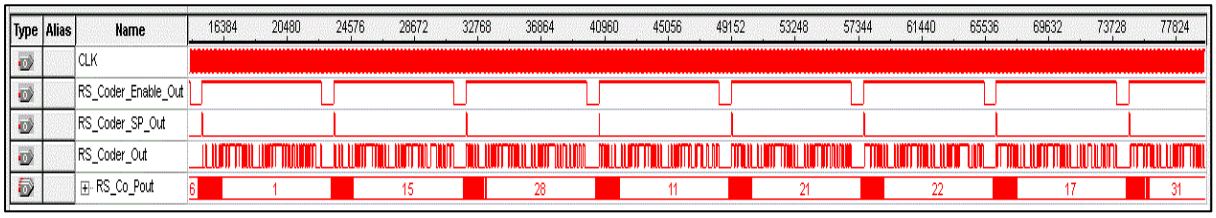


Figure 8.21 The (31,23) RS Coder Waveform Output Multi Codewords by Using STA

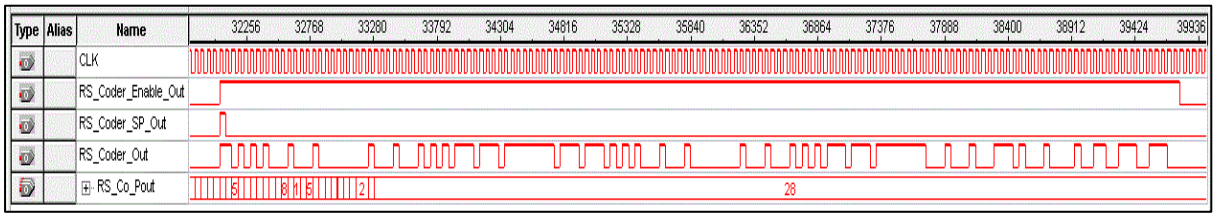


Figure 8.22 The (31,23) RS Coder Waveform Output Single Codeword by Using STA

8.4.3. DiPPM Coder Entity

The output pins of bridgecoder were connected with the input pins of the DiPPM coder as shown in figure 8.14. The DiPPM coder waveform output is displayed in figures 8.23 and 8.24, for the multi codewords and single codeword. In both figures, channel one (Yellow) represents the clock signal, while channels 2, 3 and 4, (Blue, Purple, Green) display the enable, the start packet, and the DiPPM coder output waveform respectively.

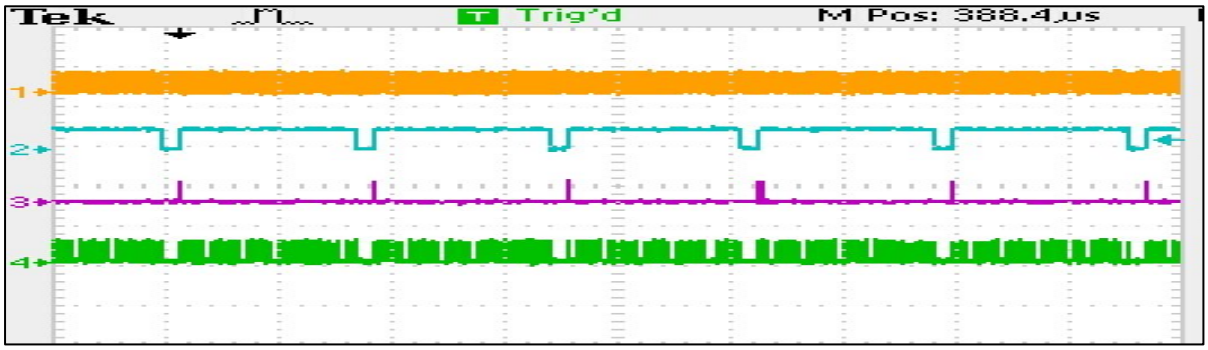


Figure 8.23 The DiPPM Coder Waveform Multi Codewords Output

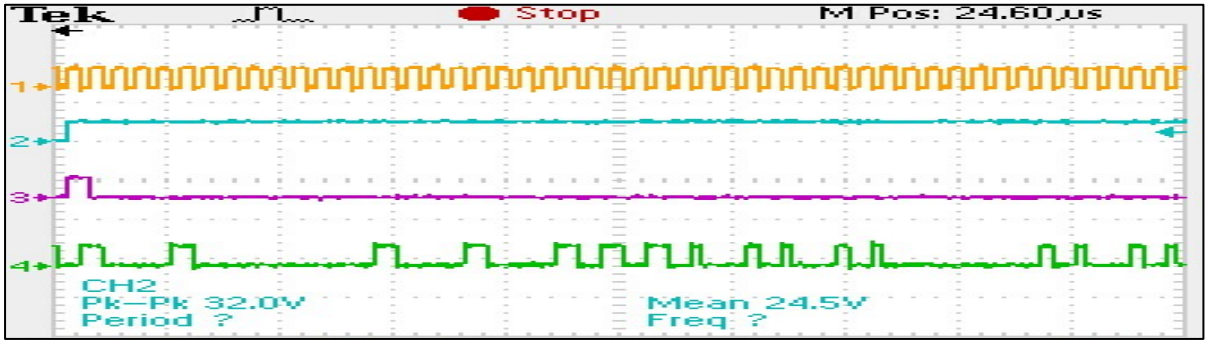


Figure 8.24 The DiPPM Coder Waveform Single Codeword Output

The system transmitter output waveform is shown in figures 8.25 and 8.26 for the multi and single codewords. The clock signal is displayed on channel one (Yellow), the PRBS output waveform is shown on channel two (Blue), the RS coder output waveform is shown on channel three (Purple), and the DiPPM coder output waveform is shown on channel four (Green).

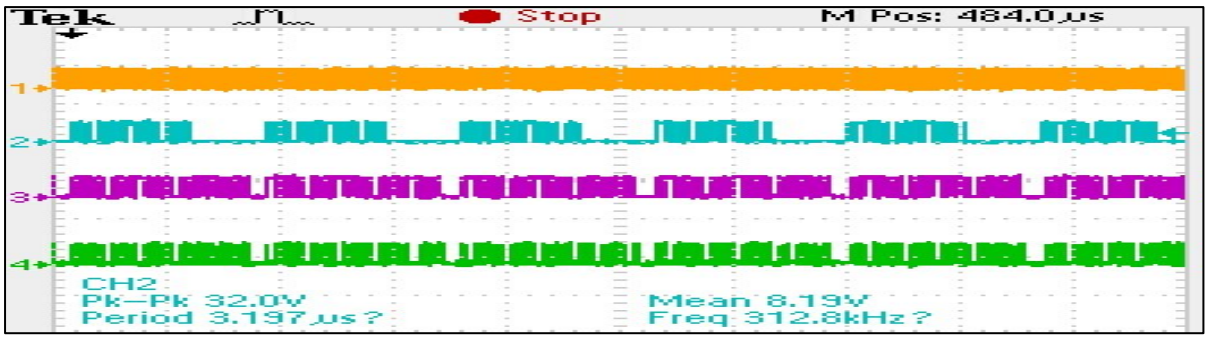


Figure 8.25 The System Transmitter Waveform Multi Codewords Output

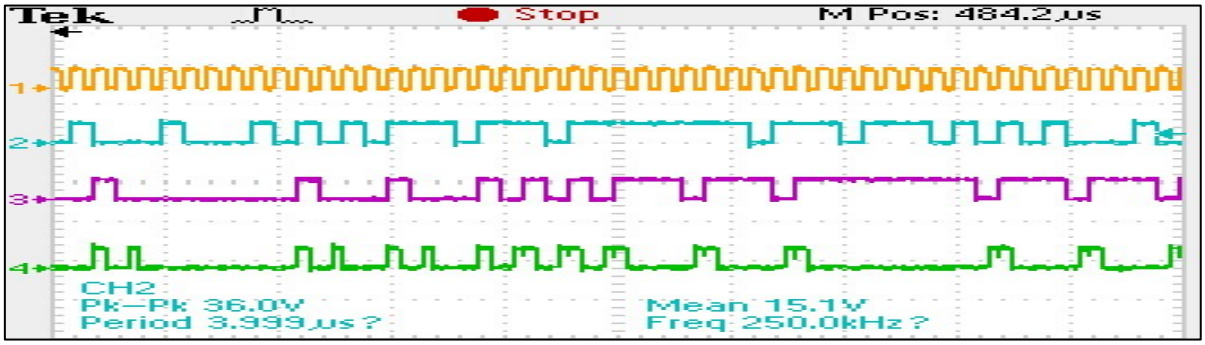


Figure 8.26 The System Transmitter Waveform Single Codeword Output

8.4.4. DiPPM Decoder Entity

The received signal, after detection by the optical receiver, goes to the DiPPM decoder entity pin. The DiPPM decoder attempts to reconstruct the original RS coder output waveform in order to permit the RS decoder to regenerate the original PCM codeword.

Figure 8.27 and 8.28 display the output waveform for the DiPPM decoder for the multi and single codewords. Figures 8.29 and 8.30 illustrate a comparison between RS coder, the DiPPM coder, and the DiPPM decoder multi and single codewords waveform output.

It is clearly noticed from figure 8.30 that the DiPPM system (coder & decoder) succeeded in modulating and demodulating the original RS coder output with only half clock cycle delay.

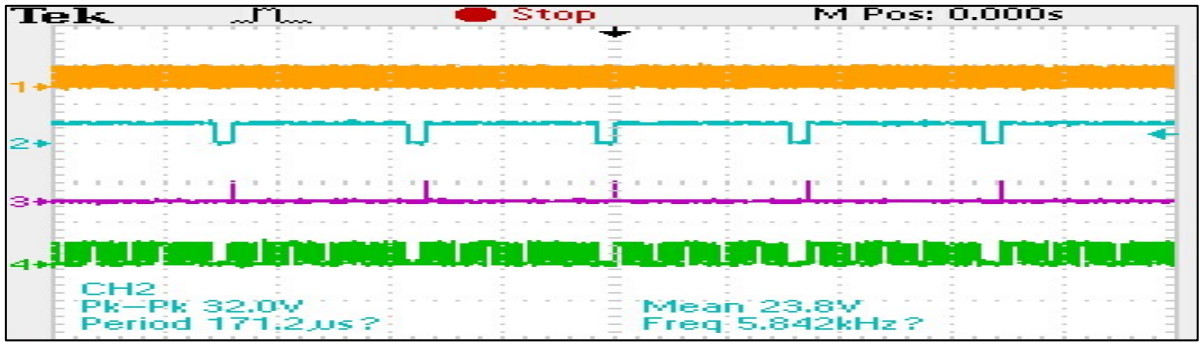


Figure 8.27 The DiPPM Decoder Waveform Multi Codewords Output

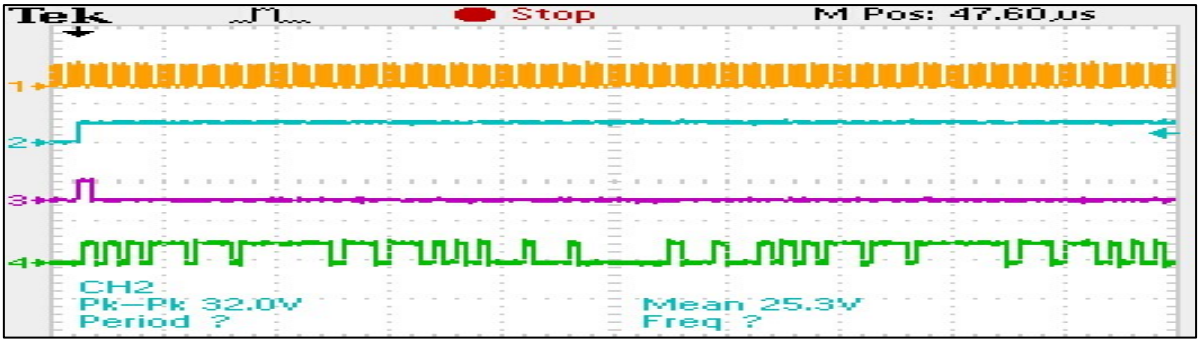


Figure 8.28 The DiPPM Deoder Waveform Single Codeword Output

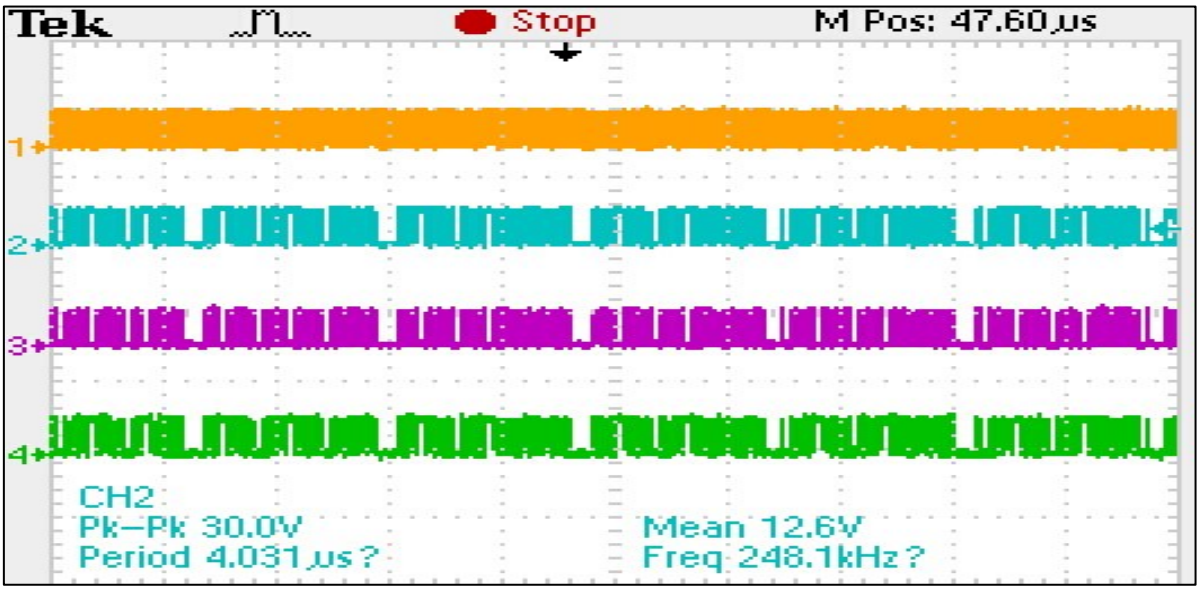


Figure 8.29 Comparison between RS Coder (blue), DiPPM Coder (purple), and DiPPM Decoder (green) Multi codewords output waveforms

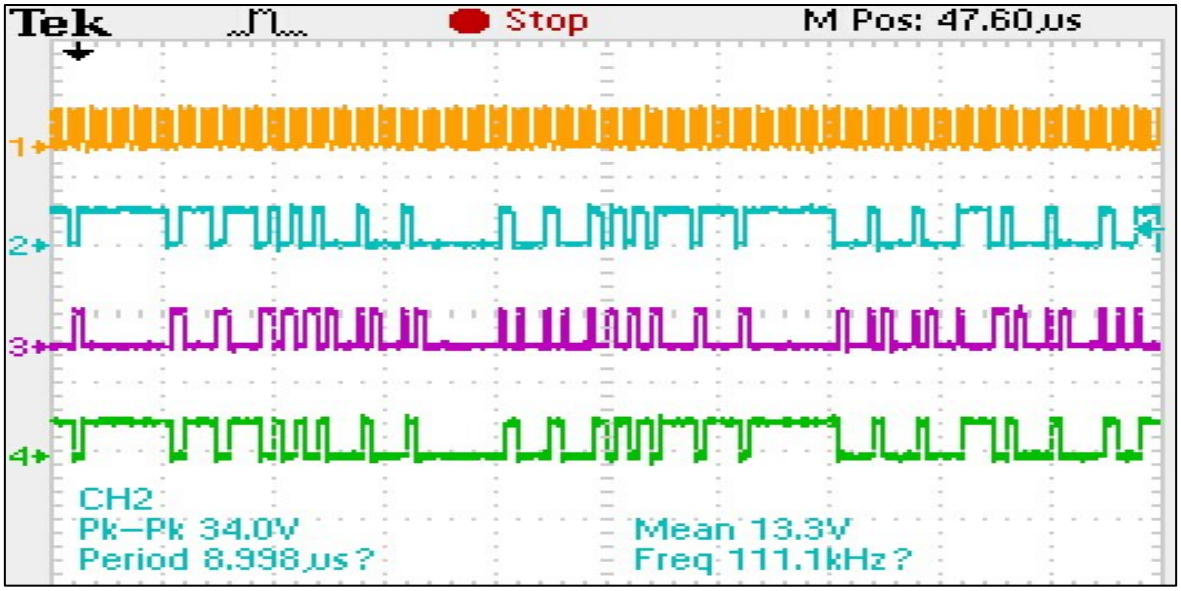


Figure 8.30 Comparison between RS Coder (blue), DiPPM Coder (purple), and DiPPM Decoder (green) Single codeword output waveforms

8.4.5. RS Decoder Entity

The (31,23) RS decoder entity is the final stage of the receiver side. This would try to regenerate the original PCM data, detect, and correct any errors occurring during the transmission process. However, the number of erasure and error symbols must be within its capability (see equation 6.10).

Figures 8.31 and 8.32 show the output waveform for the PRBS, RS coder, and the RS decoder for the multi and single codewords in serial format. Figures 8.33 and 8.34 display the output waveform for the PRBS, RS coder, and the RS decoder for the multi and single codewords in parallel format by using STA. It is noticeable from figure 8.32 and 8.34 that the RS decoder has successfully regenerated the PCM data even as there was a delay in the decoded process.

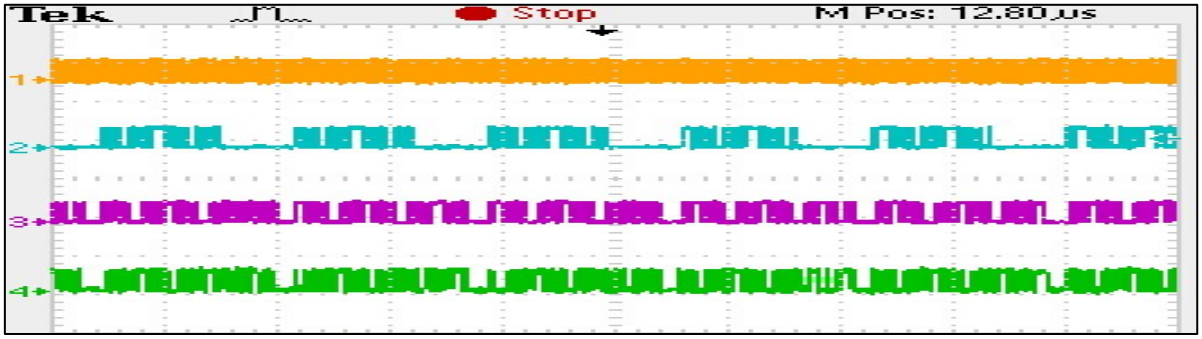


Figure 8.31 The PRBS (blue), RS Coder (purple), and RS Decoder (green) multi codewords output waveform

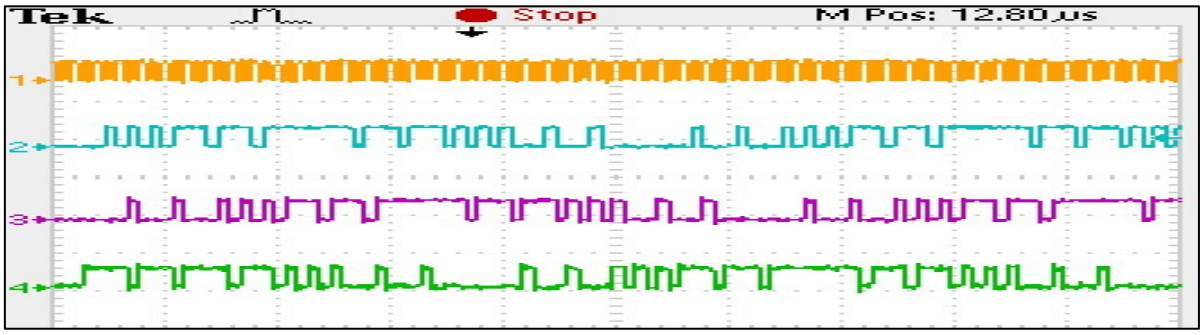


Figure 8.32 The PRBS (blue), RS Coder (purple), and RS Decoder (green) single codeword output waveform

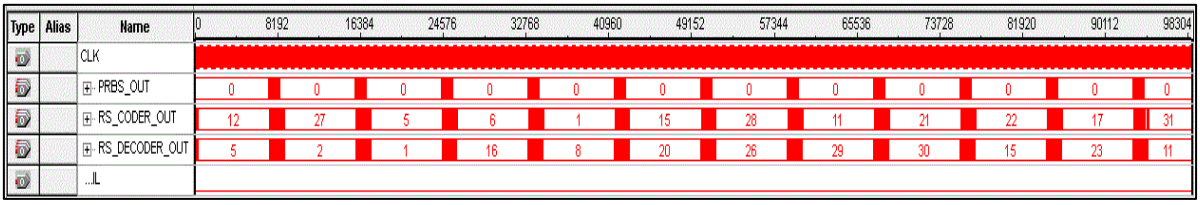


Figure 8.33 The PRBS, RS Coder, and RS Decoder multi codewords output using STA

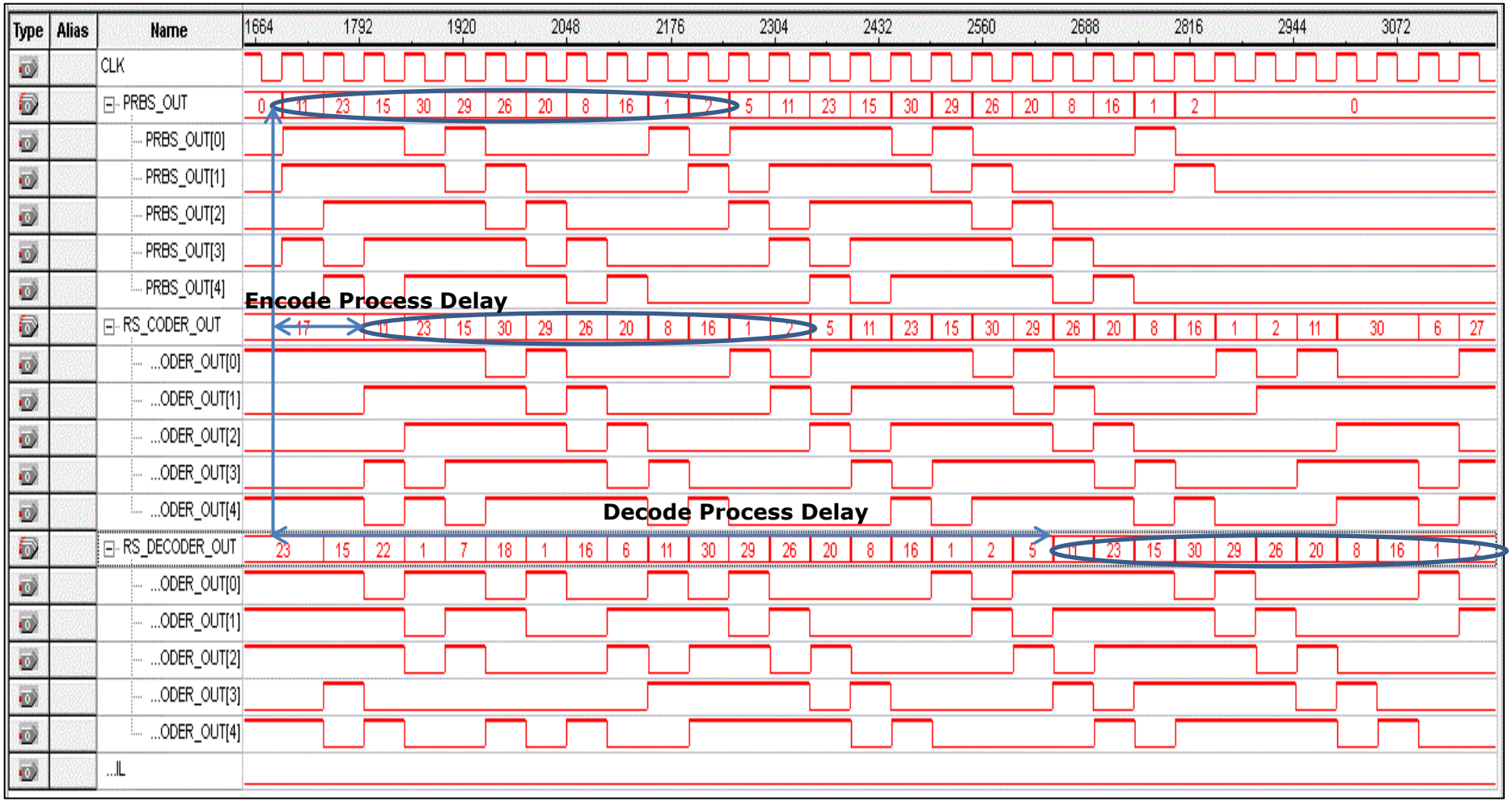


Figure 8.34 The PRBS, RS Coder, and RS Decoder multi codewords output using STA

8.4.6. Summary

This chapter has presented a practical implementation of the designed system by using Altera Quartus II software, and Cyclone III Field Programmable Gate Array (FPGA) based DSP development board. The implementation of the optical system transceiver is done as well. The output results agreed with the obtained simulation results of chapter six.

CONCLUSION AND FURTHER WORK

The following section provides a conclusion to the work done in this project and a discussion of the possible further work.

9.1. Conclusion

The Dicode PPM (DiPPM) was found to be a rather easy technique to be implemented more so because it involved the use of two slots in the transmission process that allowed the passage of one bit of PCM. In addition, the technique also provided greater sensitivity and the slot rate were found to be twice as high compared to that of the original PCM. Even as DiPPM is highly effective in optical communications, it does have its share of problems in the form of three major types of errors which have affected the functioning of the technique. These errors include the wrong-slot errors, the erasure errors, and the false alarm errors.

The major aim of this research was to develop and investigate DiPPM with the Reed Solomon (RS) Code to reduce the occurrence of errors encountered in the DiPPM technique. The RS decoder used will help in the correction of symbol errors found within its boundary without taking into account the type of error caused to the symbol. For example, while decoding a damaged byte the RS Code simply replaces the incorrect byte with a correct byte without considering whether the original errors was caused by the corruption of a single bit or all the eight bits.

The results from the simulation tests have revealed that when the RS decoder is used, it increases the transmission efficiency of the DiPPM to a large extent by decreasing the number of photons. In addition the system using the RS code has also been shown to provide an improvement of 5.12 dB as compared to the systems which do not employ the RS code. Such an improvement is observed when the code functions at the optimum rate of (3/4).

Further, the results have also shown that at this optimum code rate, the DiPPM system achieves maximum transmission efficiency. However, when the system operates below this optimum level, there is an increase in the number of redundant symbols which in turn negatively affects the performance of the system. It is only above the optimum coding rate that the redundant symbols are decrease which implies that the amount of correct symbols also decrease thereby reducing the transmission efficiency.

From the results it is also evident that the DiPPM system while using the RS code required only about 14.3 x 10^3 photons per pulse when it is operated at a bandwidth equal to or above 0.9 times the PCM data rate. On a comparative basis when the DiPPM system uses the MLSD system, it achieves a reduction in the number of photons per pulse when it is operated at a bandwidth of less than 1 times the PCM data rate. From this, it is evident that the DiPPM system when using the RS code outperforms that of the MLSD system, when it is operated at a high bandwidth. This is essentially due to the expansion of the operating bandwidth for the system based on the RS code rate.

The DiPPM system using the RS decoder has been designed using the Matlab software. The DiPPM system confirms what has been predicted from them. With the further addition of the RS decoder, the DiPPM scheme has been able to overcome the errors that had initially caused damage to the transmitted message. That has been done by using the RS decoder optimum code rate.

The VHDL has been used for the design and synthesis of a digital systems simulation. Source code has been described using VHDL for every part of the system. The optimum RS code rate has been incorporated into the system design and the results of the simulation with theory.

A test bench environment comprising of erasure only, error only, and erasure and error, has been desgined to examine the system. The system has the ability to detect and correct the erasure and error symbols when it is not overcome by their limitations.

The system VHDL source code has been downloaded on a Cyclone III Field Programmable Gate Array (FPGA) based DSP development board by using the Altera Quartus II software. has been done. The implementation of the optical system transceiver has also been carried out. The practical designed system has been tested and the output results are in agreement with the obtained simulation results.

9.2. Further Work

The following is a proposal for future research to be carried out in this area:

- The Mathcad simulation of the system can be extended by computing the sensitivity for different bit error rate, and code rate. The system can be further analysed using other filter types like the tunable filter. The results obtained could be compared with those of the coded Digital PPM system.
- The Altera DSP builder under the Matlab software can be used in the construction of the RS code Simulink system (figure 9.1) along with an optical fibre package which could help in determining the bit error rate performance of the RS decoder in a noisy communication channel. The DiPPM technique is to be used for this simulation evaluation of the coded communication system.

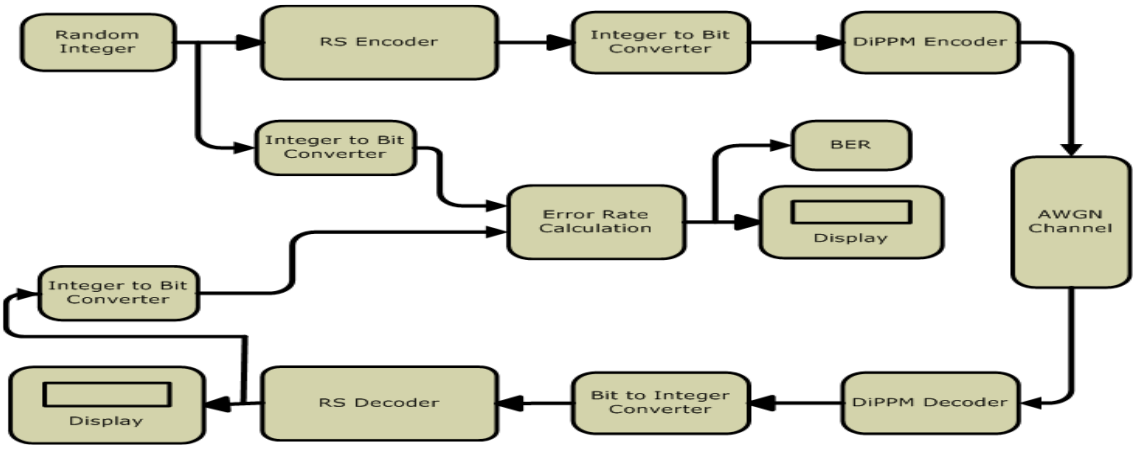


Figure 9.1 Communication System Model with RS Encoder/ RS Decoder over AWGN Channel

- Further the Matlab program in chapter 5 can be upgraded in order to send and receive an audio video data, and to measure the optical spectrum of the system.
- The RS code can be upgraded by using a rate-adaptive transmission scheme with variable-rate forward error correction codes along with a fixed signal constellation and a fixed symbol rate. This would help to quantify the variation of the bit rates with distance in a long-haul fiber system. The FEC scheme should use the serially concatenated RS codes.
- The practical implementation of the system can be improved by the addition of a timing extraction circuit. Further, if high frequencies need to be reached a double clock frequency can be used for the purpose. This will allow the system to function as a positive end, even while emulating working for the system on both edges of a single clock.

APPENDICES

10.1. Appendix 1

10.1.1. DiPPM & RS Mathcad simulation for slope detection method.

DiPPM using a PINBJT and sub-optimum - GAUSSIAN

1.Set up the scan limits

$i := 0, 1 .. 5 \qquad n := 10$

$v_i := v_{off} + \frac{i}{range}$

$v = \begin{pmatrix} 0.69 \\ 0.691 \\ 0.692 \\ 0.693 \\ 0.694 \\ 0.695 \end{pmatrix}$

$x := 0..n$ This gives the row of the matrix

$y := 0..n$ This gives the column of the matrix

2.Preamplifier terms

$f_p := 1 \cdot 10^9$ Preamp bandwidth

$\omega_p := 2 \cdot \pi \cdot f_p$

$S_o := 24\,10^{-24}$ Preamp noise at input

$\omega_n := 1 \cdot 10^9$ Noise corner frequency

$B := 1 \cdot 10^9$ Bit rate

$T_b := \frac{1}{B}$ PCM bit time

$T_s := \frac{T_b}{2 + gu}$ Slot time

$n := 10$ Number of like symbols in PCM

$\eta q := 1.6\,10^{-19}$ Quantum energy

$\lambda := 1.55\,10^{-6}$ This is the wavelength of operation

$photon\_energy := \frac{6.63\,10^{-34} \cdot 3 \cdot 10^8}{\lambda}$

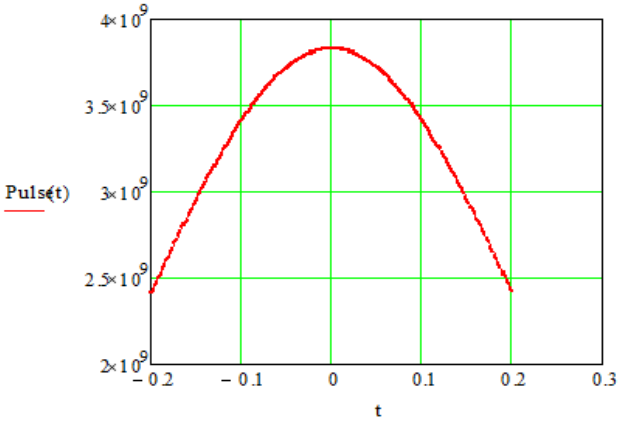
$R_o := \frac{\eta q}{photon\_energy}$ $R_o = 1.247$

3.Pulse shape terms

$$\alpha_p := \frac{0.1874 T_b}{f_n}$$

$$\alpha_{pn} := \frac{\alpha_p}{T_s}$$

$$Pulse(t) := \frac{1}{\sqrt{2\cdot\pi}\cdot\alpha_{pn}\cdot T_s}\cdot\exp\left(\frac{-t^2}{2\cdot\alpha_{pn}^2}\right)$$



4.Filter terms

$$t := -0.2, -0.199 .. 0.2$$

$$\alpha_G := \frac{\alpha_p}{k_G}$$

$$\omega_G := \frac{\sqrt{\ln(2)}}{\alpha_G}$$

$$\alpha_{Gn} := \frac{\alpha_{pn}}{k_G}$$

$$\omega_{Gn} := \frac{\sqrt{\ln(2)}}{\alpha_{Gn}}$$

$$\tau_R := \alpha_G$$

$$\omega_{pn} := 2\cdot\pi\cdot f_p\cdot T_s$$

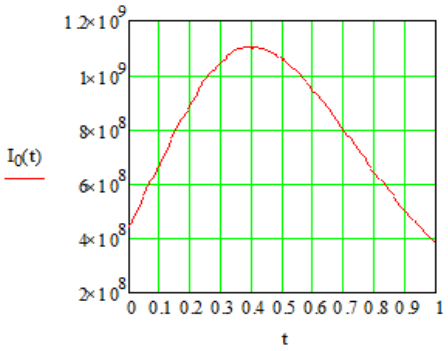
5.Pulse shape

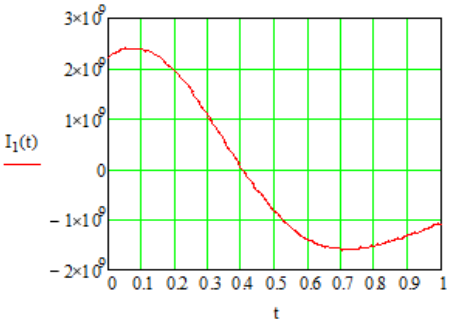
$$I_0(t) := \frac{\omega_{pn}}{2\cdot T_s}\cdot\frac{1}{\sqrt{2\cdot\pi}\cdot\alpha_{Gn}}\cdot\exp\left(\frac{\omega_{pn}^{\,2}\cdot\alpha_{pn}^{\,2}}{2}\right)\cdot\int_{-4}^{t}\exp(-\omega_{pn}\cdot\tau)\cdot\operatorname{erfc}\left(\frac{\alpha_{pn}\cdot\omega_{pn}}{\sqrt{2}}-\frac{\tau}{\sqrt{2}\cdot\alpha_{pn}}\right)\cdot\exp\left[\frac{-(t-\tau)^2}{2\cdot\alpha_{Gn}^{\,2}}\right]d\tau$$

$$I_1(t) := \frac{\omega_{pn}}{2\cdot T_s}\cdot\frac{1}{\sqrt{2\cdot\pi}\cdot\alpha_{Gn}}\cdot\exp\left(\frac{\omega_{pn}^{\,2}\cdot\alpha_{pn}^{\,2}}{2}\right)\cdot\exp(-\omega_{pn}\cdot t)\cdot\operatorname{erfc}\left(\frac{\alpha_{pn}\cdot\omega_{pn}}{\sqrt{2}}-\frac{t}{\sqrt{2}\cdot\alpha_{pn}}\right)\ldots$$

$$+\frac{\omega_{pn}}{2\cdot T_s}\cdot\frac{1}{\sqrt{2\cdot\pi}\cdot\alpha_{Gn}^{\,3}}\cdot\exp\left(\frac{\omega_{pn}^{\,2}\cdot\alpha_{pn}^{\,2}}{2}\right)\cdot\int_{-4}^{t}-\exp(-\omega_{pn}\cdot\tau)\cdot\operatorname{erfc}\left(\frac{\alpha_{pn}\cdot\omega_{pn}}{\sqrt{2}}-\frac{\tau}{\sqrt{2}\cdot\alpha_{pn}}\right)\cdot(t-\tau)\cdot\exp\left[\frac{-(t-\tau)^2}{2\cdot\alpha_{Gn}^{\,2}}\right]d\tau$$

$t := 0, 0.01 .. 1$





$$noise := \frac{\omega_p}{2}\cdot erfc\left(\alpha_G\cdot\omega_p\right)\cdot exp\left(\alpha_G^{\,2}\cdot\omega_p^{\,2}\right) + \frac{\omega_p^{\,2}}{\pi\cdot\omega_n^{\,2}}\cdot\int_0^{1\cdot10^{13}} exp\left(-\left(\alpha_G^{\,2}\cdot\omega^2\right)\right)\cdot\frac{\omega^2}{\omega_p^{\,2}+\omega^2}\,d\omega$$

$$noise = 2.39910^{10}$$

6.modified pulse definitions

ISI pulse for pulse in slot S - sequence is S xN R yN S xN etc

current prior post

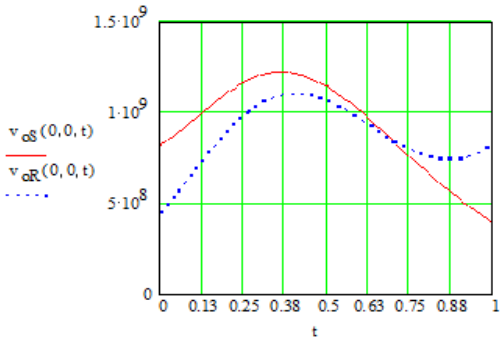
$$v_{oS}(x,y,t) := I_0(t) + (y \le limi)\cdot I_0[t + [(y+1)\cdot(gu+2) - 1]] + (x < limi)\cdot I_0[t - [(x+1)\cdot(gu+1) + 2]]$$

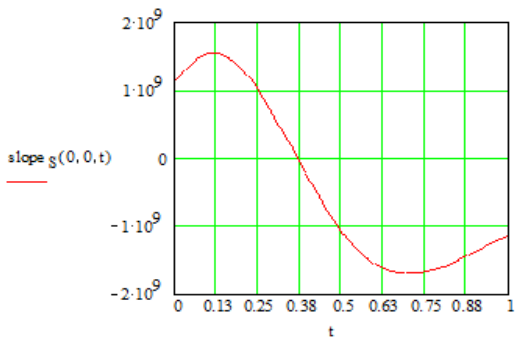
$$slope_S(x,y,t) := I_1(t) + (y \le limi)\cdot I_1[t + [(y+1)\cdot(gu+2) - 1]] + (x < limi)\cdot I_1[t - [(x+1)\cdot(gu+1) + 2]]$$

$$v_{oR}(x,y,t) := I_0(t) + (x < limi)\cdot I_0[t + [(x+1)\cdot(gu+1) + 2]] + (y \le limi)\cdot I_0[t - [(y+1)\cdot(gu+2) - 1]]$$

$$slope_R(x,y,t) := I_1(t) + (x < limi)\cdot I_1[t + [(x+1)\cdot(gu+1) + 2]] + (y \le limi)\cdot I_1[t - [(y+1)\cdot(gu+2) - 1]]$$

$t := 0, 0.01..1$





7.Determine the peak pulse amplitude

Pulse in S first

$t := 0.3$ Guess at the peak time

$t_{pS_{x,y}} := \text{root}\left(\text{slope}_S(x,y,t)\cdot T_s^{\,2}, t\right)$ This is the peak time for pulse in slot S as a function of x and y

$t_{pS_{0,0}} = 0.365$

$v_{pS_{x,y}} := v_{oS}\left(x, y, t_{pS_{x,y}}\right)$ This is the peak voltage for pulse in slot S as a function of x and y

$v_{pS_{0,0}} = 1.22310^{9}$

Pulse in R

$t_{pR_{x,y}} := \text{root}\left(\text{slope}_R(x,y,t)\cdot T_s^{\,2}, t\right)$ This is the peak time for pulse in slot R as a function of x and y

$t_{pR_{0,0}} = 0.404$

$v_{pR_{x,y}} := v_{oR}\left(x, y, t_{pR_{x,y}}\right)$ This is the peak voltage for pulse in slot R as a function of x and y

$v_{pR_{0,0}} = 1.10510^{9}$

8.Determine the decision time

$t := -0.05$ Guess at the decision time

$v_{oS}(0,0,t) = 8.02110^{8}$ $v_{oS}\left(0,0,t_{pS_{0,0}}\right) = 1.223\times 10^{9}$

$$t_{d_i} := \text{root}\left[\left(v_{oS}(0,0,t) - v_i \cdot v_{oS}\left(0,0,t_{pS_{0,0}}\right)\right)\cdot T_s^{1.5}, t\right]$$

$$v_{d_i} := v_{oS}\left(0,0,t_{d_i}\right)$$ This is the decision level voltage

t_{d_i} =

0.017
0.018
0.019
0.02
0.021
0.022

v_{d_i} =

$7.772 \cdot 10^{8}$
$7.782 \cdot 10^{8}$
$7.791 \cdot 10^{8}$
$7.801 \cdot 10^{8}$
$7.811 \cdot 10^{8}$
$7.821 \cdot 10^{8}$

$v_i \cdot v_{oS}\left(0,0,t_{pS_{0,0}}\right)$ =

$7.742 \cdot 10^{8}$
$7.754 \cdot 10^{8}$
$7.766 \cdot 10^{8}$
$7.779 \cdot 10^{8}$
$7.791 \cdot 10^{8}$
$7.803 \cdot 10^{8}$

v_i =

0.633
0.634
0.635
0.636
0.637
0.638

9.Determine the slope at the decision time

$$slopeS_{x\cdot(n+1)+y,i} := slope_S\left(x,y,t_{d_i}\right)$$

$$slopeR_{x\cdot(n+1)+y,i} := slope_R\left(x,y,t_{d_i}\right)$$

10.Modify the peak pulse amplitude

$$t_{pkR_{x\cdot(n+1)+y,i}} := t_{pR_{x,y}}\cdot\left(t_{pR_{0,0}} - t_{d_i} \le 0.5\right) + \left(t_{d_i} + \frac{1}{2}\right)\cdot\left(t_{pR_{0,0}} - t_{d_i} > 0.5\right)$$

$$t_{pkR_{x\cdot(n+1)+y,i}} := t_{pR_{x,y}}$$

$$t_{pkS_{x\cdot(n+1)+y,i}} := t_{pS_{x,y}}\cdot\left(t_{pS_{0,0}} - t_{d_i} \le 0.5\right) + \left(t_{d_i} + \frac{1}{2}\right)\cdot\left(t_{pS_{0,0}} - t_{d_i} > 0.5\right)$$

$$t_{pkS_{x\cdot(n+1)+y,i}} := t_{pS_{x,y}}$$ $\mathbf{gu} \equiv \mathbf{0}$

11.Error sources

Pulse in S wrong slotting to adjacent slot R - sequence is S xN R yN S xN R

$$Q_{SR_{x\cdot(n+1)+y,i}} := \eta q \cdot \left(\frac{1}{2}\right) \cdot \frac{slopeS_{x\cdot(n+1)+y,i}}{\sqrt{S_o \, noise}}$$

$$P_{sSR}(x,y,b,i) := \frac{1}{2}\,erfc\left[\frac{Q_{SR_{x\cdot(n+1)+y,i}} \cdot b}{\sqrt{2}}\right]$$ This is the prob. of a SET wrong slotting to adjacent RESET

$$P_{ewsSR}(b,i) := \sum_{y=0}^{n-1}\sum_{x=0}^{n-1}\left[\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{sSR}(x,y,b,i)\cdot(x+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{sSR}(n,y,b,i)\cdot(n+1)\right] \ldots$$
$$+ \sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sSR}(x,n,b,i)\cdot(x+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sSR}(n,n,b,i)\cdot(n+1)$$

Pulse in R wrong slotting to prior slot S

$$Q_{RS_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{1}{2} \cdot \frac{slopeR_{x\cdot(n+1)+y,i}}{\sqrt{S_o \, noise}}$$

$$P_{sRS}(x,y,b,i) := \frac{1}{2}\,erfc\left[\frac{Q_{RS_{x\cdot(n+1)+y,i}} \cdot b}{\sqrt{2}}\right]$$ This is the prob of a RESET going to preceding SET

$$P_{ewsRS}(b,i) := \sum_{y=0}^{n-1}\sum_{x=0}^{n-1}\left[\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{sRS}(x,y,b,i)\cdot(y+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{sRS}(n,y,b,i)\cdot(y+1)\right] \ldots$$
$$+ \sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sRS}(x,n,b,i)\cdot(n+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sRS}(n,n,b,i)\cdot(n+1)$$

Pulse in R wrong slotting to following slot G

$$Q_{RG_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{1}{2} \cdot \frac{slopeR_{x\cdot(n+1)+y,i}}{\sqrt{S_o \cdot noise}}$$

$$P_{sRG}(x,y,b,i) := \frac{1}{2} \cdot erfc\left[\frac{Q_{RG_{x\cdot(n+1)+y,i}} \cdot b}{\sqrt{2}}\right]$$

This is the prob of a RESET going to following Guard

$$P_{ewsRG}(b,i) := \sum_{y=0}^{n-1}\sum_{x=0}^{n-1}\left[\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{sRG}(x,y,b,i)\cdot(y+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{sRG}(n,y,b,i)\cdot(y+1)\right] \ldots$$
$$+ \sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sRG}(x,n,b,i)\cdot(n+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sRG}(n,n,b,i)\cdot(n+1)$$

$$P_{es}(b,i) := P_{ewsSR}(b,i) + P_{ewsRS}(b,i) + P_{ewsRG}(b,i)$$

Erasure of pulse in S

$$Q_{eS_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{oS}\left[x,y,t_{pkS_{x\cdot(n+1)+y,i}}\right] - v_{d_i}}{\sqrt{S_o \cdot noise}}$$

$$P_{eS}(x,y,b,i) := \frac{1}{2} \cdot erfc\left[\frac{Q_{eS_{x\cdot(n+1)+y,i}} \cdot b}{\sqrt{2}}\right]$$

$$P_{eeS}(b,i) := \sum_{y=0}^{n-1}\sum_{x=0}^{n-1}\left[\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{eS}(x,y,b,i)\cdot(x+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{eS}(n,y,b,i)\cdot(n+1)\right] \ldots$$
$$+ \sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{eS}(x,n,b,i)\cdot(x+1)\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{sRG}(n,n,b,i)\cdot(n+1)$$

Erasure of pulse in R

$$Q_{eR_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{oR}\left[x, y, t_{pkR_{x\cdot(n+1)+y,i}}\right] - v_{d_i}}{\sqrt{S_o \cdot noise}}$$

$$P_{eR}(x, y, b, i) := \frac{1}{2} \cdot erfc\left[\frac{Q_{eR_{x\cdot(n+1)+y,i}} \cdot b}{\sqrt{2}}\right]$$

$$P_{eeR}(b, i) := \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} \left[\left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{eR}(x, y, b, i) \cdot (y+1)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{eR}(n, y, b, i) \cdot (y+1)\right] \ldots$$
$$+ \sum_{x=0}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{eR}(x, n, b, i) \cdot (n+1)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{eR}(n, n, b, i) \cdot (n+1)$$

$$P_{er}(b, i) := P_{eeS}(b, i) + P_{eeR}(b, i)$$

False alarm

False alarm in prior frame slot R. Sequence is S N R
ISI voltage made up of prior S signal and R pulse

$$v_{oR\_4Ts_{x\cdot(n+1)+y,i}} := v_{oR}\left[x, y, t_{d_i} - (2 + gu)\right]$$

$$Q_{R\_4Ts_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{d_i} - v_{oR\_4Ts_{x\cdot(n+1)+y,i}}}{\sqrt{S_o \cdot noise}}$$

$$P_{R\_4Ts}(b, i, x, y) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left[\frac{b \cdot Q_{R\_4Ts_{x(n+1)+y,i}}}{\sqrt{2}}\right]$$

$$P_{eR\_4Ts}(b, i) := \sum_{y=0}^{n-1} \sum_{x=1}^{n-1} \left[\left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{R\_4Ts}(b, i, x, y)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{R\_4Ts}(b, i, n, y)\right] \ldots$$
$$+ \sum_{x=1}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{R\_4Ts}(b, i, x, n)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{R\_4Ts}(b, i, n, n)$$

False alarm in immediate slot S - Pulse in slot R
ISI voltage made up of prior S, current R, following S
Sequence is S xN R yN S

$$v_{oR\_Ts_{x\cdot(n+1)+y,i}} := v_{oR}\left(x, y, t_{d_i} - 1\right)$$

$$Q_{R\_Ts_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{d_i} - v_{oR\_Ts_{x\cdot(n+1)+y,i}}}{\sqrt{S_o \cdot noise}}$$

$$P_{R\_Ts}(b, i, x, y) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left[\frac{b \cdot Q_{R\_Ts_{x\cdot(n+1)+y,i}}}{\sqrt{2}}\right]$$

$$P_{eR\_Ts}(b, i) := \sum_{y=0}^{n-1} \sum_{x=0}^{n-1} \left[\left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{R\_Ts}(b, i, x, y) \cdot (y+1)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{R\_Ts}(b, i, n, y) \cdot (y+1)\right] \ldots$$
$$+ \sum_{x=0}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{R\_Ts}(b, i, x, n) \cdot (n+1)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{R\_Ts}(b, i, n, n) \cdot (n+1)$$

False alarm in following frame slot S - Pulse in slot R
ISI voltage made up of current R, following S

$$v_{oR3Ts_{x\cdot(n+1)+y,i}} := v_{oR}\left(x, y, t_{d_i} + (1 + gu)\right]$$

$$Q_{R3Ts_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{d_i} - v_{oR3Ts_{x\cdot(n+1)+y,i}}}{\sqrt{S_o \cdot noise}}$$

$$P_{R3Ts}(b, i, x, y) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left[\frac{b \cdot Q_{R3Ts_{x\cdot(n+1)+y,i}}}{\sqrt{2}}\right]$$

$$P_{eR3Ts}(b, i) := \sum_{y=1}^{n-1} \sum_{x=0}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{R3Ts}(b, i, x, y) \cdot y + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{R\_Ts}(b, i, n, y) \cdot y\right] \ldots$$
$$+ \sum_{x=0}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{R\_Ts}(b, i, x, n) \cdot n\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{R\_Ts}(b, i, n, n) \cdot n$$

False alarm between R and S pulses - N to SET

$$Q_{NS}(b,i) := b \cdot \eta q \cdot \frac{v_{d_i}}{\sqrt{S_o \cdot noise}}$$

$$P_{NS}(b,i) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{Q_{NS}(b,i)}{\sqrt{2}}\right)$$

$$P_{eNS}(b,i) := \sum_{y=3}^{n-1} \sum_{k=2}^{y-1} \left[\left(\frac{1}{2}\right)^{y+3} \cdot P_{NS}(b,i) \cdot (y+1-k)\right] + \sum_{k=2}^{n-1} \left[\left(\frac{1}{2}\right)^{n+2} \cdot P_{NS}(b,i) \cdot (n+1-k)\right]$$

False alarm in prior frame slot S with pulse in slot S
ISI voltage made up of prior R signal and current S pulse

$$v_{oS\_4Ts_{x\cdot(n+1)+y,i}} := v_{oS}\left(x, y, t_{d_i} - (2+gu)\right]$$

$$Q_{S\_4Ts_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{d_i} - v_{oS\_4Ts_{x\cdot(n+1)+y,i}}}{\sqrt{S_o \cdot noise}}$$

$$P_{S\_4Ts}(b,i,x,y) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left[\frac{b \cdot Q_{S\_4Ts_{x\cdot(n+1)+y,i}}}{\sqrt{2}}\right]$$

$$P_{eS\_4Ts}(b,i) := \sum_{y=2}^{n-1} \sum_{x=0}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{S\_4Ts}(b,i,x,y) + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{y+2} \cdot P_{S\_4Ts}(b,i,n,y)\right] \ldots$$
$$+ \sum_{x=0}^{n-1} \left[\left(\frac{1}{2}\right)^{x+2} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{S\_4Ts}(b,i,x,n)\right] + \left(\frac{1}{2}\right)^{n+1} \cdot \left(\frac{1}{2}\right)^{n+1} \cdot P_{S\_4Ts}(b,i,n,n)$$

False alarm in following frame slot R with pulse in slot S

$$v_{oS5Ts_{x\cdot(n+1)+y,i}} := v_{oS}\left[x, y, t_{d_i} + (3 + gu)\right]$$

$$Q_{S5Ts_{x\cdot(n+1)+y,i}} := \eta q \cdot \frac{v_{d_i} - v_{oS5Ts_{x\cdot(n+1)+y,i}}}{\sqrt{S_o \cdot noise}}$$

$$P_{S5Ts}(b,i,x,y) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left[\frac{b \cdot Q_{S5Ts_{x\cdot(n+1)+y,i}}}{\sqrt{2}}\right]$$

$$P_{eS5Ts}(b,i) := \sum_{y=0}^{n-1}\sum_{x=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{S5Ts}(b,i,x,y)\cdot x + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot P_{S5Ts}(b,i,n,y)\cdot n\right] \ldots$$
$$+ \sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{S5Ts}(b,i,x,n)\cdot x\right] + \left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot P_{S5Ts}(b,i,n,n)\cdot n$$

False alarm between S and R pulses - N to R

$$Q_{NR}(b,i) := b \cdot \eta q \cdot \frac{v_{d_i}}{\sqrt{S_o \cdot noise}}$$

$$P_{NR}(b,i) := \frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{Q_{NR}(b,i)}{\sqrt{2}}\right)$$

$$P_{eNR}(b,i) := \sum_{x=3}^{n-1}\sum_{k=2}^{x-1}\left[\left(\frac{1}{2}\right)^{x+3}\cdot P_{NR}(b,i)\cdot(x+1-k)\right] + \sum_{k=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n+2}\cdot P_{NR}(b,i)\cdot(n+1-k)\right]$$

12.Calculating DiPPM probability of error

$$P_{eftotal}(b,i) := P_{eR\_4Ts}(b,i) + P_{eR\_Ts}(b,i) + P_{eR3Ts}(b,i) + P_{eNS}(b,i) + P_{eS\_4Ts}(b,i) + P_{eS5Ts}(b,i) + P_{eNR}(b,i)$$

$$P_{eb}(b,i) := P_{es}(b,i) + P_{eftotal}(b,i) + P_{er}(b,i) \qquad b := 9010^3 \quad \text{Guess number of photons}$$

13.Calculating RS probability of error

$m := 4 \quad t := 2$

$$r := \frac{2^m - 1 - 2t}{2^m - 1} \qquad r = 0.733$$

$$pe(b,i) := \frac{1}{2^m - 1} \cdot \sum_{j=t+1}^{2^m-1} \left[j \cdot \left[\frac{(2^m - 1)!}{j! \cdot \left[(2^m - 1)! - (j-1)! \right]} \cdot (P_{eb}(b,i))^{j-1} \cdot (1 - P_{eb}(b,i))^{(2^m-1)-(j-1)} \right] \right]$$

$$peb(b,i) := \frac{2^{m-1}}{2^m - 1} \cdot pe(b,i)$$

Calculating number of photons

$$pc(b,i) := \left(\log(peb(b,i) \cdot 10^5) + 4 \right)$$ Set for 1 in 10^9 errors

$f_n = 1.8$ $\quad P_{er}(b,0) = 3.33610^{-5}$ $\quad P_{eftotal}(b,0) = 1.72 \cdot 10^{-5}$

$P_{es}(b,0) = 3.73210^{-7}$ $\quad P_{eR\_Ts}(b,0) = 0$

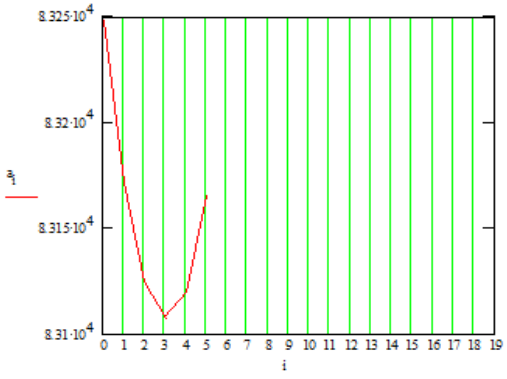
$a_i := root(pc(b,i), b)$ Find the root to give 1 in 10^9

$a_i =$

$8.325 \cdot 10^4$
$8.317 \cdot 10^4$
$8.312 \cdot 10^4$
$8.311 \cdot 10^4$
$8.312 \cdot 10^4$
$8.317 \cdot 10^4$

minimum= min(a)

minimum = $8.311 \cdot 10^4$



limit= 8 $\text{minimum} \cdot 10^{-3} = 83.108$

$v_{off} = 0.633$ $gu = 0$ $f_n \equiv 1.8$ $i := 3$ $range \equiv 1000$

$k_G \equiv 0.75$

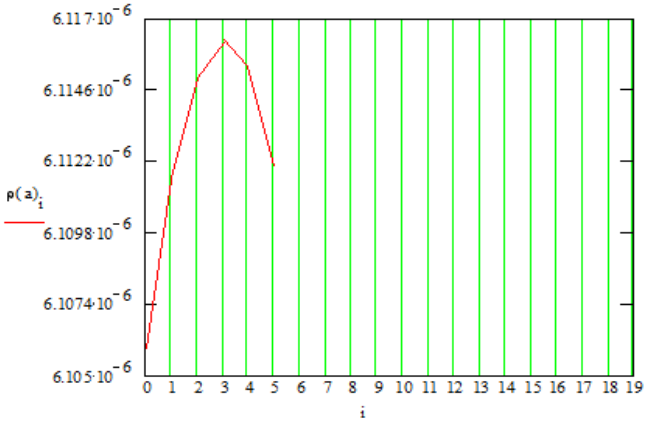
$\frac{\omega_G}{2 \cdot \pi} = 9.545 \times 10^{8}$

$v_i = 0.693$ $t_{d_i} = 0.02$

Calculating Transmission efficiency

$$\rho(a) := \frac{r \cdot \ln(2)}{a}$$

$$\rho(a) = \begin{bmatrix} 6.10610^{-6} \\ 6.11210^{-6} \\ 6.11510^{-6} \\ 6.11610^{-6} \\ 6.11510^{-6} \\ 6.11210^{-6} \end{bmatrix}$$



$$dBm := 10 \log\left[\text{minimum}\frac{photon\_energy}{10^{-3}} \cdot \left[\frac{n+1}{2 \cdot (gu+2) \cdot n}\right] \cdot B\right]$$

$dBm = -25.327$

10.1.2. DiPPM & RS Mathcad simulation for central detection method.

DiPPM using standard detection and matched filter

$i := 0, 1 .. 10$ $n := 10$ Set up the scan limits

$v_i := v_{off} + \frac{i}{range}$

$x := 0..n$ This gives the row of the matrix

$y := 0..n$ This gives the column of the matrix

1.Preamplifier terms

$S_o := 50\,10^{-24}$ Preamp noise at input - double sided Philips CGY2110CU

$f_p \equiv 10\,10^{9}$

$B := 1 \cdot 10^{9}$ Bit rate

$T_b := \frac{1}{B}$ PCM bit time

$T_s := \frac{T_b}{2}$ Slot time

$\eta q := 1.610^{-19}$ Quantum energy

2.Pulse shape

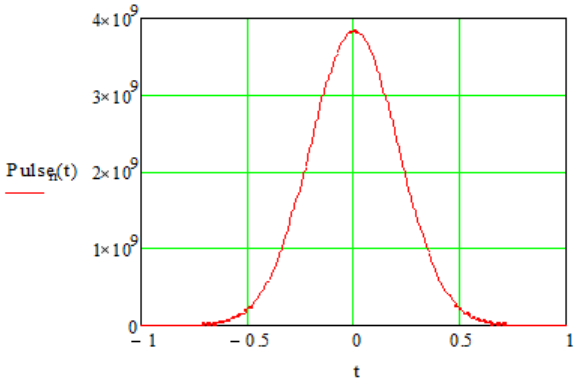
$\alpha := \frac{0.1874 T_b}{f_n}$ fn is the fibre bandwidth normalised to the bit rate

$\alpha_n := \frac{0.1874 T_b}{f_n \cdot T_s}$ $\omega_{cn} := 2 \cdot \pi \cdot f_p \cdot T_s$ $\omega_c := 2 \cdot \pi \cdot f_p$

$\tau_R := \alpha$ $\tau_{Rn} := \alpha_n$

$$Pulse_n(t) := \frac{1}{\sqrt{2\cdot\pi}\cdot\alpha_n\cdot T_s}\cdot\exp\left(\frac{-t^2}{2\cdot\alpha_n^{\,2}}\right)$$

$t := -1, -0.99 .. 1$



3.Pulse shape Matched filter only

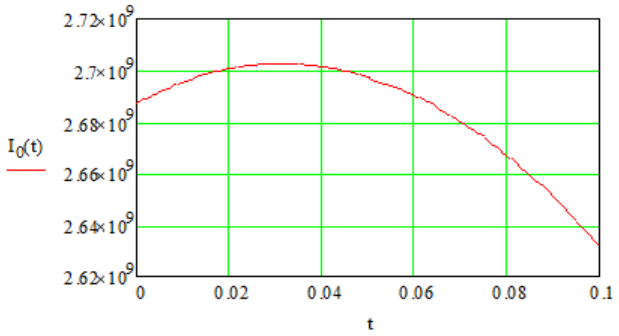
$$I_0(t) := \frac{\omega_{cn}}{2\cdot T_s}\cdot\exp\left[\left(\alpha_n\cdot\omega_{cn}\right)^2\right]\cdot\exp\left(-\omega_{cn}\cdot t\right)\cdot\operatorname{erfc}\left(\alpha_n\cdot\omega_{cn}-\frac{t}{2\cdot\alpha_n}\right)$$

$$I_1(t) := \frac{-1}{2}\cdot\frac{\omega_{cn}^{\,2}}{T_s}\cdot\exp\left(\alpha_n^{\,2}\cdot\omega_{cn}^{\,2}\right)\cdot\exp\left(-\omega_{cn}\cdot t\right)\cdot\left(1+\operatorname{erf}\left(-\alpha_n\cdot\omega_{cn}+\frac{1}{2}\cdot\frac{t}{\alpha_n}\right)\right)\ldots$$
$$+\frac{1}{2}\cdot\frac{\omega_{cn}}{T_s}\cdot\exp\left(\alpha_n^{\,2}\cdot\omega_{cn}^{\,2}\right)\cdot\frac{\exp\left(-\omega_{cn}\cdot t\right)}{\pi^{\left(\frac{1}{2}\right)}}\cdot\frac{\exp\left[-\left(-\alpha_n\cdot\omega_{cn}+\frac{1}{2}\cdot\frac{t}{\alpha_n}\right)^2\right]}{\alpha_n}$$

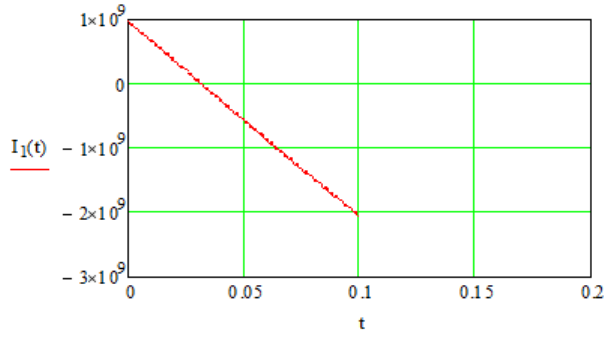
$$I_0(t) := -.11283791670955125738\cdot\omega_{cn}\cdot\frac{\left(20\cdot\alpha_n^{\,4}\cdot\omega_{cn}^{\,2}-20\cdot\alpha_n^{\,2}\cdot\omega_{cn}\cdot t+5\cdot t^2-7\cdot\alpha_n^{\,2}\right)}{\left[\left(-2\cdot\alpha_n^{\,2}\cdot\omega_{cn}+t\right)^3\cdot T_s\right]}\cdot\exp\left(-.250000000000000000000\cdot\frac{t^2}{\alpha_n^{\,2}}\right)$$

$$I_1(t) := 5.641895835477562869 \cdot 10^{-2} \cdot \left[\begin{array}{l} 40\alpha_n^{\,6} \cdot \omega_{cn}^{\,2} \cdot (1 - \omega_{cn} \cdot t) \ldots \\ + \alpha_n^{\,4} \cdot \omega_{cn} \cdot t \cdot (60\omega_{cn} \cdot t - 26) \ldots \\ + 3 \cdot \alpha_n^{\,2} \cdot t^2 - 30\alpha_n^{\,2} \cdot \omega_{cn} \cdot t^3 + 5 \cdot t^4 - 42\alpha_n^{\,4} \end{array} \right] \cdot \left[\exp\left[-\left(\frac{t}{2 \cdot \alpha_n} \right)^2 \right] \cdot \frac{\omega_{cn}}{\left(-2 \cdot \alpha_n^{\,2} \cdot \omega_{cn} + t \right)^4 \cdot \alpha_n \cdot T_s} \right]$$

$t := 0, 0.001 .. 0.1$



$noise := I_0(0)$



4.Peak pulse amplitude - isolated pulse

$t_{p0} :=$ | $t \leftarrow t_{pmin}$ | $t_p \leftarrow \text{root}\left(I_1(t) \cdot T_s^{\,2}, t\right)$ | t_p

$t_{pmin} := 0.0032$

$t_{p0} = 3.203 \times 10^{-3}$

5.Threshold level - isolated pulse

$t_g := -0.005$

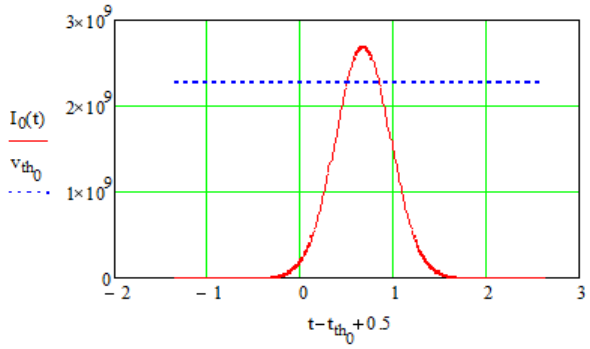
$t_{th_i} :=$ | $t \leftarrow t_g$ | $t \leftarrow \text{root}\left(I_0(t) - v_i \cdot I_0(t_{p0})\right), t$ | t

$t_{th_0} = -0.141$

$v_{th_i} := I_0\left(t_{th_i}\right)$ This is the decision level voltage

$t_{d0_i} := t_{th_i}$

$t := -2, -1.99 .. 2$



6.Modify peak of single pulse

$t_p := \text{if}\left[\left(t_{p0} - t_{th_0} + 0.5 \geq 1\right), \left(t_{th_0} + 0.5\right), t_{p0}\right]$

$t_p = 3.203 \times 10^{-3}$

$t_{p0} := t_p$

$t_{p0} - t_{th_0} + 0.5 = 0.645$

7.Pulse definitions

$v_{o4}(a,b,c,d,t) := I_0(t+a) + I_0(t+b) + I_0(t+c) + I_0(t+d)$

$v_{o3}(a,b,c,t) := I_0(t+a) + I_0(t+b) + I_0(t+c)$

$v_{o2}(a,b,t) := I_0(t+a) + I_0(t+b)$

$v_{14}(a,b,c,d,t) := I_1(t+a) + I_1(t+b) + I_1(t+c) + I_1(t+d)$

$v_{13}(a,b,c,t) := I_1(t+a) + I_1(t+b) + I_1(t+c)$

$v_{12}(a,b,t) := I_1(t+a) + I_1(t+b)$

$v_{oRSRS}(t) := v_{o4}(4,3,0,-1,t)$

$v_{oRS}(t) := v_{o2}(0,-1,t)$

$v_{oRSRNS}(t) := v_{o4}(4,3,0,-3,t)$

$v_{oRSNRNS}(t) := v_{o4}(6,5,0,-3,t)$

$v_{oRNS}(t) := v_{o2}(0,-3,t)$

$v_{oRSR}(t) := v_{o3}(4,3,0,t)$

$v_{oRSNR}(t) := v_{o3}(6,5,0,t)$

$v_{oRNSRS}(t) := v_{o4}(6,3,0,-1,t)$

$v_{oRNSRNS}(t) := v_{o4}(6,3,0,-3,t)$

$v_{oRNSR}(t) := v_{o3}(6,3,0,t)$

$v_{oSRS}(t) := v_{o3}(3,0,-1,t)$

$v_{oSRNS}(t) := v_{o3}(3,0,-3,t)$

$v_{oSR}(t) := v_{o2}(3,0,t)$

$v_{1RSRS}(t) := v_{14}(4,3,0,-1,t)$

$v_{1RS}(t) := v_{12}(0,-1,t)$

$v_{1RSRNS}(t) := v_{14}(4,3,0,-3,t)$

$v_{1RSNRNS}(t) := v_{14}(6,5,0,-3,t)$

$v_{1RNS}(t) := v_{12}(0,-3,t)$

$v_{1RSR}(t) := v_{13}(4,3,0,t)$

$v_{1RSNR}(t) := v_{13}(6,5,0,t)$

$v_{1RNSRS}(t) := v_{14}(6,3,0,-1,t)$

$v_{1RNSRNS}(t) := v_{14}(6,3,0,-3,t)$

$v_{1RNSR}(t) := v_{13}(6,3,0,t)$

$v_{1SRS}(t) := v_{13}(3,0,-1,t)$

$v_{1SRNS}(t) := v_{13}(3,0,-3,t)$

$v_{1SR}(t) := v_{12}(3,0,t)$

$v_{o8NRS}(t) := v_{o3}(5,0,-1,t)$

$v_{o8RSR}(t) := v_{o4}(4,1,0,-3,t)$

$v_{o8NR}(t) := v_{o2}(5,0,t)$

$v_{o8RS2}(t) := v_{o3}(4,1,0,t)$

$v_{o8NRSR}(t) := v_{o4}(6,1,0,-3,t)$

$v_{o8NRS2}(t) := v_{o3}(6,1,0,t)$

$v_{oRSR2}(t) := v_{o3}(1,0,-3,t)$

$v_{oRS2}(t) := v_{o2}(1,0,t)$

$v_{oRNSR2}(t) := v_{o3}(3,0,-3,t)$

$v_{o8R2}(t) := v_{o2}(0,-3,t)$

$v_{o8RNS2}(t) := v_{o3}(6,3,0,t)$

$v_{o8RSNR}(t) := v_{o4}(4,1,0,-5,t)$

$v_{o8NRSNR}(t) := v_{o4}(6,1,0,-5,t)$

$v_{oRNS2}(t) := v_{o2}(3,0,t)$

$v_{1SNRS}(t) := v_{13}(5,0,-1,t)$

$v_{1SRSR}(t) := v_{14}(4,1,0,-3,t)$

$v_{1SNR}(t) := v_{12}(5,0,t)$

$v_{1SRS2}(t) := v_{13}(4,1,0,t)$

$v_{1SNRSR}(t) := v_{14}(6,1,0,-3,t)$

$v_{1SNRS2}(t) := v_{13}(6,1,0,t)$

$v_{1RSR2}(t) := v_{13}(1,0,-3,t)$

$v_{1RS2}(t) := v_{12}(1,0,t)$

$v_{1RNSR2}(t) := v_{13}(3,0,-3,t)$

$v_{1SR2}(t) := v_{12}(0,-3,t)$

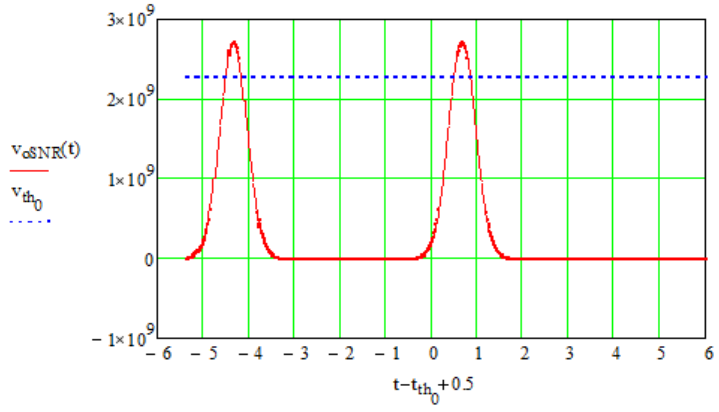
$v_{1SRNS2}(t) := v_{13}(6,3,0,t)$

$v_{1SRSNR}(t) := v_{14}(4,1,0,-5,t)$

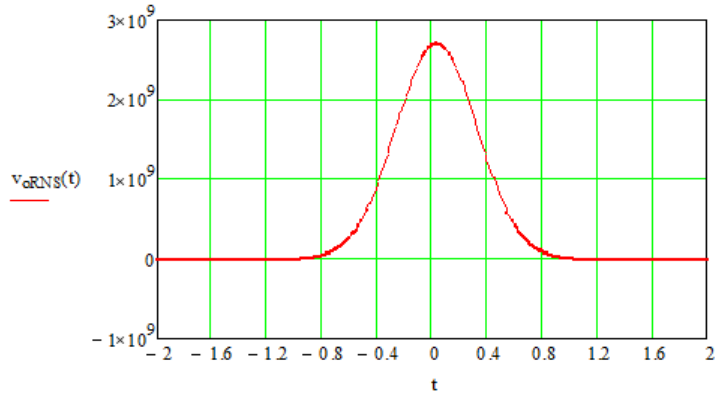
$v_{1SNRSNR}(t) := v_{14}(6,1,0,-5,t)$

$v_{1RNS2}(t) := v_{12}(3,0,t)$

$t := -6,-5.99..6$



$t := -6, -5.99 .. 6$



8.Peak value times

$$t_{p4}(a,b,c,d,t_{pmin}) := \begin{| t \leftarrow t_{pmin} \\ t_{peak} \leftarrow \text{root}\left(v_{14}(a,b,c,d,t)\cdot T_s^{\,2}, t\right) \\ t_p \leftarrow \text{if}\left[\left(t_{peak} - t_{th_0} + 0.5 \geq 1\right), \left(t_{th_0} + 0.5\right), t_{peak}\right] \\ t_p}$$

$$t_{p3}(a,b,c,t_{pmin}) := \begin{| t \leftarrow t_{pmin} \\ t_{peak} \leftarrow \text{root}\left(v_{13}(a,b,c,t)\cdot T_s^{\,2}, t\right) \\ t_p \leftarrow \text{if}\left[\left(t_{peak} - t_{th_0} + 0.5 \geq 1\right), \left(t_{th_0} + 0.5\right), t_{peak}\right] \\ t_p}$$

$$t_{p2}(a,b,t_{pmin}) := \begin{| t \leftarrow t_{pmin} \\ t_{peak} \leftarrow \text{root}\left(v_{12}(a,b,t)\cdot T_s^{\,2}, t\right) \\ t_p \leftarrow \text{if}\left(\left(t_{peak} - t_{th_0} + 0.5 \geq 1\right), \left(t_{th_0} + 0.5\right), t_{peak}\right] \\ t_p}$$

$t_g := 0.025 \qquad t_{g1} := 0.025 \qquad t_{g2} := 0.025$

$t_{pRSRS} := t_{p4}(4,3,0,-1,t_{g1})$ $\quad t_{pRSRS} - t_{th_0} + 0.5 = 0.676$

$t_{pRS} := t_{p2}(0,-1,t_{g1})$ $\quad t_{pRS} - t_{th_0} + 0.5 = 0.676$

$t_{pRSRNS} := t_{p4}(4,3,0,-3,t_g)$ $\quad t_{pRSRNS} - t_{th_0} + 0.5 = 0.673$

$t_{pRSNRNS} := t_{p4}(6,5,0,-3,t_g)$ $\quad t_{pRSNRNS} - t_{th_0} + 0.5 = 0.673$

$t_{pRNS} := t_{p2}(0,-3,t_g)$ $\quad t_{pRNS} - t_{th_0} + 0.5 = 0.673$

$t_{pRSR} := t_{p3}(4,3,0,t_g)$ $\quad t_{pRSR} - t_{th_0} + 0.5 = 0.673$

$t_{pRSNR} := t_{p3}(6,5,0,t_g)$ $\quad t_{pRSNR} - t_{th_0} + 0.5 = 0.673$

$t_{pRNSRS} := t_{p4}(6,3,0,-1,t_{g1})$ $\quad t_{pRNSRS} - t_{th_0} + 0.5 = 0.676$

$t_{pRNSRNS} := t_{p4}(6,3,0,-3,t_g)$ $\quad t_{pRNSRNS} - t_{th_0} + 0.5 = 0.673$

$t_{pRNSR} := t_{p3}(6,3,0,t_g)$ $\quad t_{pRNSR} - t_{th_0} + 0.5 = 0.673$

$t_{pSRS} := t_{p3}(3,0,-1,t_{g2})$ $\quad t_{pSRS} - t_{th_0} + 0.5 = 0.676$

$t_{pSRNS} := t_{p3}(3,0,-3,t_g)$ $\quad t_{pSRNS} - t_{th_0} + 0.5 = 0.673$

$t_{pSR} := t_{p2}(3,0,t_g)$ $\quad t_{pSR} - t_{th_0} + 0.5 = 0.673$

$t_{pSNRS} := t_{p3}(5,0,-1,t_{g2})$ $\quad t_{pSNRS} - t_{th_0} + 0.5 = 0.676$

$t_{pSNR} := t_{p2}(5,0,t_g)$ $\quad t_{pSNR} - t_{th_0} + 0.5 = 0.673$

$t_{pSRSR} := t_{p4}(4,1,0,-3,t_{g2})$ $\quad t_{pSRSR} - t_{th_0} + 0.5 = 0.669$

$t_{pSRS2} := t_{p3}(4,1,0,t_{g2})$ $\quad t_{pSRS2} - t_{th_0} + 0.5 = 0.669$

$t_{pSNRSR} := t_{p4}(6,1,0,-3,t_{g2})$ $\quad t_{pSNRSR} - t_{th_0} + 0.5 = 0.669$

$t_{pSNRS2} := t_{p3}(6,1,0,t_g)$ $\quad t_{pSNRS2} - t_{th_0} + 0.5 = 0.669$

$t_{pRSR2} := t_{p3}(1,0,-3,t_g)$ $\quad t_{pRSR2} - t_{th_0} + 0.5 = 0.669$

$t_{pRS2} := t_{p2}(1,0,t_{g2})$ $\quad t_{pRS2} - t_{th_0} + 0.5 = 0.669$

$t_{pRNSR2} := t_{p3}(3,0,-3,t_g)$ $\quad t_{pRNSR2} - t_{th_0} + 0.5 = 0.673$

$t_{pSR2} := t_{p2}(0,-3,t_g)$ $\quad t_{pSR2} - t_{th_0} + 0.5 = 0.673$

$t_{pSRNS2} := t_{p3}(6,3,0,t_g)$ $\quad t_{pSRNS2} - t_{th_0} + 0.5 = 0.673$

$t_{pSRSNR} := t_{p3}(4,1,0,t_{g2})$ $\quad t_{pSRSNR} - t_{th_0} + 0.5 = 0.669$

$t_{pSNRSNR} := t_{p3}(5,0,-1,t_{g2})$ $\quad t_{pSNRSNR} - t_{th_0} + 0.5 = 0.676$

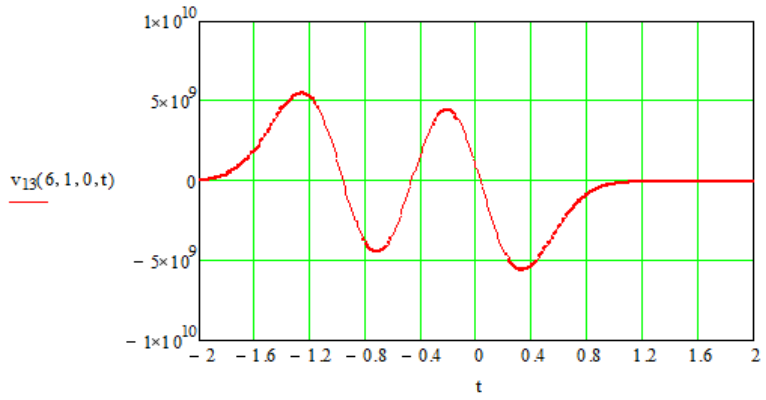
$t_{pRNS2} := t_{p2}(3,0,t_g)$ $\quad t_{pRNS2} - t_{th_0} + 0.5 = 0.673$

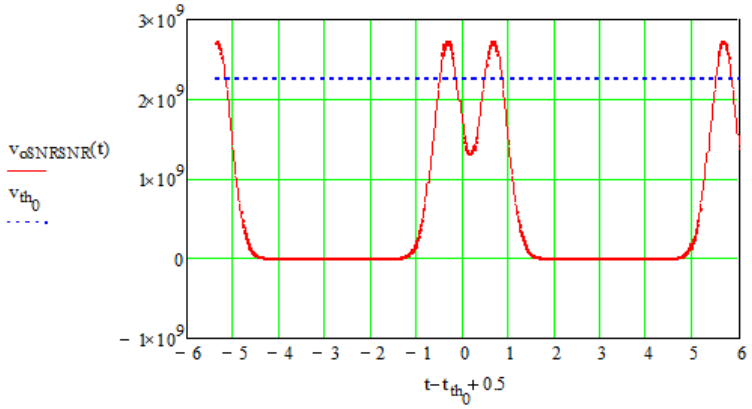
$t_{pRSRNS} = 0.032$

$t_{pRSRS} = 0.035$

$t_{pSRS} = 0.035$

$t := -6, -5.99 .. 6$


$v_{13}(6, 1, 0, t)$
1×10^{10}
5×10^{9}
0
-5×10^{9}
-1×10^{10}
−2 −1.6 −1.2 −0.8 −0.4 0 0.4 0.8 1.2 1.6 2
t


$v_{oSNRSNR}(t)$
v_{th_0}
3×10^{9}
2×10^{9}
1×10^{9}
0
-1×10^{9}
−6 −5 −4 −3 −2 −1 0 1 2 3 4 5 6
$t - t_{th_0} + 0.5$

9.Erasure AND wrong slot errors

Erasure/W.S. (either way) of R gives y+1 errors in sequence
R y1N S xN R yN S

$errors_{R_y} := y + 1$

Sequence R S xN R S

y1 = 0, x = 0, y = 0 ONLY

$$\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 256$$

$$Q_i := \eta q\cdot\frac{v_{oRSRS}\left(t_{pRSRS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\left(\frac{1}{2}\right)\cdot\frac{v_{1RSRS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er1}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 0, x = 1..n, y = 0 ONLY

$$\left[\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^x\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right] + \frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^n\cdot 1\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 256$$

$$Q_i := \eta q\cdot\frac{v_{oRS}\left(t_{pRS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\left(\frac{1}{2}\right)\cdot\frac{v_{1RS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er2}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s2}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

$$P_{erRinRSxNRS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) \qquad P_{sRinRSxNRS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i)$$

y1 = 0, x = 0..n and y = 1..n - Erasure of R in sequence RS xN R yN S

y1 = 0, x = 0, y = 1 ONLY

$$\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(1+1)\right]\cdot 8192 = 256$$

$$Q_i := \eta q\cdot\frac{v_{oRSRNS}\left(t_{pRSRNS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RSRNS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er1}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 0, x = 1, y = 1 ONLY

$$\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(1+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oRSNRNS}\left(t_{pRSNRNS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RSNRNS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er2}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s2}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 0, x = 2..n, y = 1 ONLY

$$\left[\left[\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^x\cdot\frac{1}{2}\left(\frac{1}{2}\right)^1\cdot\frac{1}{2}\cdot(1+1)\right] + \frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^n\cdot 1\cdot\left(\frac{1}{2}\right)^1\cdot\frac{1}{2}\cdot(1+1)\right]\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oRNS}\left(t_{pRNS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RNS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er3}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s3}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 0, x = 0, y = 2..n ONLY

$$\left[\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\cdot 8192 = 511.5$$

$$Q_i := \eta q\cdot\frac{v_{oRSR}\left(t_{pRSR}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RSR}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er4}(b,i) := \frac{511.5}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s4}(b,i) := \frac{511.5}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 0, x = 1, y = 2..n ONLY

$$\left[\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\cdot 8192 = 255.75$$

$$Q_i := \eta q\cdot\frac{v_{oR1SNR}\left(t_{pR1SNR}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1R1SNR}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er5}(b,i) := \frac{255.75}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s5}(b,i) := \frac{255.75}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 0, x = 2..n, y = 2..n ONLY

$$\left[\begin{array}{l}\sum_{x=2}^{n-1}\left[\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\ldots\\ +\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\end{array}\right]\cdot 8192 = 255.75$$

$$Q_i := \eta q\cdot\frac{I_0\left(t_{p0}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{I_1\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er6}(b,i) := \frac{255.75}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s6}(b,i) := \frac{255.75}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

$$P_{erRinRSxNRyNS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) + P_{er3}(b,i) + P_{er4}(b,i) + P_{er5}(b,i) + P_{er6}(b,i)$$

$$P_{sRinRSxNRyNS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i) + P_{s3}(b,i) + P_{s4}(b,i) + P_{s5}(b,i) + P_{s6}(b,i)$$

$P_{erRinRSxNRyNS}(b,10) = 0.067$ $\quad$ $P_{sRinRSxNRyNS}(b,10) = 4.138 \times 10^{-3}$

Erasure of R in sequence R y1N S xN R yN S

y1 = 1..n, x = 0 and y = 0..n - ERASURE OF R in sequence R y1N S R yN S

y1 = 1, y = 0

$$\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oRNSRS}\left(t_{pRNSRS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RNSRS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er1}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 1, y = 1

$$\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot(1+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oRNSRNS}\left(t_{pRNSRNS}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RNSRNS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er2}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s2}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 1, y = 2..n

$$\left[\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\cdot 8192 = 255.75$$

$$Q_i := \eta q\cdot\frac{v_{oRNSR}\left(t_{pRNSR}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Qs_i := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RNSR}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{esi}(b,i) := \frac{255.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{esi}(b,i) := \frac{255.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Qs_i}{\sqrt{2}}\right)\right)$$

y1 = 2..n, y = 0

$$\left[\sum_{y1=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot(0+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oSRS}\left(t_{pSRS}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Qs_i := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SRS}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{esi}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{esi}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Qs_i}{\sqrt{2}}\right)\right)$$

y1 = 2..n, y = 1

$$\left[\sum_{y1=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot(1+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot(1+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q \cdot \frac{v_{oSRNS}(t_{pSRNS}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1SRNS}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{er5}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s5}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 2..n, y = 2..n

$$\left[\sum_{y1=2}^{n-1}\left[\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\dots\right]\cdot 8192 = 255.75$$
$$\left[+\sum_{y=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]$$

$$Q_i := \eta q \cdot \frac{v_{oSR}(t_{pSR}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1SR}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{er6}(b,i) := \frac{255.75}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s6}(b,i) := \frac{255.75}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

$$P_{erRinSRNS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) + P_{er3}(b,i) + P_{er4}(b,i) + P_{er5}(b,i) + P_{er6}(b,i)$$

$$P_{sRinSRNS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i) + P_{s3}(b,i) + P_{s4}(b,i) + P_{s5}(b,i) + P_{s6}(b,i)$$

$P_{erRinSRNS}(b,10) = 0.045$ $\qquad$ $P_{sRinSRNS}(b,10) = 2.748 \times 10^{-3}$

y1 = 1..n, x = 1..n and y = 0 - ERASURE OF R in sequence R y1N S xN R S

y1 = 1..n, x = 1, y = 0

$$\left[\sum_{y1=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q \cdot \frac{v_{oSNRS}(t_{pSNRS}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1SNRS}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{er1}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot \operatorname{erfc}\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot \operatorname{erfc}\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 1..n, x = 2..n, y = 0

$$\left[\sum_{y1=1}^{n-1}\left[\sum_{x=2}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{y1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^{x} \cdot \frac{1}{2} \cdot \frac{1}{2}(0+1)\right] + \frac{1}{4}\left(\frac{1}{2}\right)^{y1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^{n} \cdot 1 \cdot \frac{1}{2}(0+1)\right] \ldots\right] \cdot 8192 = 128$$
$$+\sum_{x=2}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{n} \cdot 1 \cdot \left(\frac{1}{2}\right)^{x} \cdot \frac{1}{2} \cdot \frac{1}{2}(0+1)\right] + \frac{1}{4}\left(\frac{1}{2}\right)^{n} \cdot 1 \cdot \left(\frac{1}{2}\right)^{n} \cdot 1 \cdot \frac{1}{2}(0+1)\Bigg]$$

$$Q_i := \eta q \cdot \frac{v_{oRS}(t_{pRS}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1RS}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{er2}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot \operatorname{erfc}\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s2}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot \operatorname{erfc}\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

$$P_{erRinRNSNRS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) \qquad P_{sRinRNSNRS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i)$$

y1 = 1..n, x = 1..n and y = 1..n - ERASURE OF R in sequence R y1N S xN R yN S

y1 = 1, x = 1, y = 1..n

$$\left[\sum_{y=1}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^{1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^{y} \cdot \frac{1}{2}(y+1)\right] + \frac{1}{4}\left(\frac{1}{2}\right)^{1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^{1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^{n} \cdot 1 \cdot (n+1)\right] \cdot 8192 = 191.875$$

$$Q_i := \eta q \cdot \frac{v_{oSNR}(t_{pSNR}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1SNR}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{gi1}(b,i) := \frac{191.875}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{si1}(b,i) := \frac{191.875}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 1..n, x = 2..n, y = 1

$$\left[\sum_{y1=1}^{n-1}\left[\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot(1+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot(1+1)\right]\cdots\right]\cdot 8192 = 128$$
$$\left[+\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot(1+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}(1+1)\right]$$

$$Q_i := \eta q \cdot \frac{v_{oRNS}(t_{pRNS}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1RNS}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{gi2}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{si2}(b,i) := \frac{128}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 2..n, x = 1, y = 1..n

$$\left[\sum_{y1=2}^{n-1}\left[\sum_{y=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\cdots\right]\cdot 8192 = 191.875$$
$$\left[+\sum_{y=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]$$

$$Q_i := \eta q \cdot \frac{v_{oSNR}(t_{pSNR}) - v_{th_i}}{\sqrt{S_o \cdot noise}} \qquad Q_{S_i} := \eta q \cdot \frac{1}{2} \cdot \frac{v_{1SNR}(t_{d0_i})}{\sqrt{S_o \cdot noise}}$$

$$P_{gi3}(b,i) := \frac{191.875}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{si3}(b,i) := \frac{191.875}{8192} \cdot \left(\frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

y1 = 1..n, x = 2..n, y = 2..n, ERRORS

$$\left[\sum_{y1=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left[\sum_{x=2}^{n-1}\left[\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(y+1)\right]+\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{n}\cdot(n+1)\right]\cdots\right.\right.\right.$$
$$\left.\left.\left.+\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(y+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{n}\cdot(n+1)\right]\right]\cdots\right]\cdot 8192 = 255.75$$
$$+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left[\sum_{x=2}^{n-1}\left[\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(y+1)\right]+\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{n}\cdot(n+1)\right]\cdots\right.$$
$$\left.+\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(y+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{n}\cdot(n+1)\right]$$

$$Q_i := \eta q\cdot\frac{I_0(t_{p0}) - v_{th_i}}{\sqrt{S_o\cdot noise}}$$

$$Qs_i := \eta q\cdot\frac{1}{2}\cdot\frac{I_1(t_{d0_i})}{\sqrt{S_o\cdot noise}}$$

$$P_{er4}(b,i) := \frac{255.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right)$$

$$P_{s4}(b,i) := \frac{255.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Qs_i}{\sqrt{2}}\right)\right)$$

$$P_{erRinSNRNS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) + P_{er3}(b,i) + P_{er4}(b,i)$$

$$P_{sRinSNRNS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i) + P_{s3}(b,i) + P_{s4}(b,i)$$

$$P_{eeR}(b,i) := P_{erRinRSxNRS}(b,i) + P_{erRinRSxNRyNS}(b,i) \cdots$$
$$+ P_{erRinSRNS}(b,i) + P_{erRinRNSNRS}(b,i) + P_{erRinSNRNS}(b,i)$$

$$P_{eeR}(b,10) = 0.179$$

$$P_{wsR}(b,i) := 2\cdot\left(P_{sRinRSxNRS}(b,i) + P_{sRinRSxNRyNS}(b,i) \cdots + P_{sRinSRNS}(b,i) + P_{sRinRNSNRS}(b,i) + P_{sRinSNRNS}(b,i)\right)$$

$$P_{wsR}(b,10) = 0.022$$

Erasure/w.s. of pulse in S $\quad errors_{S_x} := x + 1$

ERASURE/W.S. OF PULSE IN S, sequence Sx1NRyNSxNR with x1 = 0..n, x = 0..n and y = 0

x1 = 0, y = 0, x = 0

$$\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 256$$

$$Q_i := \eta q\cdot\frac{v_{oSRSR}\left(t_{pSRSR}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SRSR}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{e1}(b,i) := \frac{256}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{256}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 0, y = 0, x = 1..n

$$\left[\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^x\cdot\frac{1}{2}\cdot(x+1)\right] + \frac{1}{4}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^n\cdot 1\cdot(n+1)\right]\cdot 8192 = 767.5$$

$$Q_i := \eta q\cdot\frac{v_{oSRS2}\left(t_{pSRS2}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SRS2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{e2}(b,i) := \frac{767.5}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s2}(b,i) := \frac{767.5}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 1, y = 0, x = 0

$$\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^1\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oSNRSR}\left(t_{pSNRSR}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SNRSR}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{e3}(b,i) := \frac{128}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s3}(b,i) := \frac{128}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 1, y = 0, x = 1..n

$$\left[\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\cdot 8192 = 383.75$$

$$Q_i := \eta q\cdot\frac{v_{oSNRS2}\left(t_{pSNRS2}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SNRS2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er4}(b,i) := \frac{383.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s4}(b,i) := \frac{383.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 2..n, y = 0, x = 0

$$\left[\sum_{x1=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 128$$

$$Q_i := \eta q\cdot\frac{v_{oRSR2}\left(t_{pRSR2}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RSR2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er5}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s5}(b,i) := \frac{128}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 2..n, y = 0, x = 1..n, ERRORS

$$\left[\begin{array}{l}\sum_{x1=2}^{n-1}\left[\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\ldots \\ +\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\end{array}\right]\cdot 8192 = 383.75$$

$$Q_i := \eta q\cdot\frac{v_{oRS2}\left(t_{pRS2}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RS2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er6}(b,i) := \frac{383.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s6}(b,i) := \frac{383.75}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

$$P_{eSinRS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) + P_{er3}(b,i) + P_{er4}(b,i) + P_{er5}(b,i) + P_{er6}(b,i)$$

$$P_{sSinRS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i) + P_{s3}(b,i) + P_{s4}(b,i) + P_{s5}(b,i) + P_{s6}(b,i)$$

$$P_{sSinRS}(b,i) := 0$$

ERASURE OF PULSE IN S, sequence SR with x = 0 and y = 1..n

x1 = 0..n, y = 1, x = 0

$$\left[\sum_{x1=0}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right] + \frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot(0+1)\right]\cdot 8192 = 256$$

$$Q_i := \eta q\cdot\frac{v_{oRNSR2}\left(t_{pRNSR2}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Qs_i := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1RNSR2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er1}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{256}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Qs_i}{\sqrt{2}}\right)\right)$$

x1 = 0..n, y = 2..n, x = 0, ERRORS

$$\left[\sum_{x1=0}^{n-1}\left[\sum_{y=2}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot\frac{1}{2}(0+1)\right] + \frac{1}{4}\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}(0+1)\right]\ldots\right]\cdot 8192 = 256.25$$
$$\left[+\sum_{y=1}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot\frac{1}{2}(0+1)\right] + \frac{1}{4}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}(0+1)\right]$$

$$Q_i := \eta q\cdot\frac{v_{oSR2}\left(t_{pSR2}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Qs_i := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SR2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er2}(b,i) := \frac{256.25}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s2}(b,i) := \frac{256.25}{8192}\cdot\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Qs_i}{\sqrt{2}}\right)\right)$$

$$P_{eSinSR}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) \qquad P_{sSinSR}(b,i) := P_{s1}(b,i) + P_{s2}(b,i)$$

ERASURE OF PULSE IN S ALONE with x = 1..n and y = 1..n

x1 = 0, y = 1, x = 1..n

$$\left[\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x+1)\right]+\frac{1}{4}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\cdot 8192 = 383.75$$

$$Q_i := \eta q\cdot\frac{v_{oSRNS2}\left(t_{pSRNS2}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{v_{1SRNS2}\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{e1}(b,i) := \frac{383.75}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s1}(b,i) := \frac{383.75}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 0..n, y = 2..n, x = 1..n

$$\left[\sum_{x1=0}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\left[\sum_{x=1}^{n-1}\left[\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(x+1)\right]+\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{n}\cdot(x+1)\right]\ldots\right]\right]\ldots\right]\cdot 8192 = 767.25$$

$$+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left[\sum_{x=2}^{n-1}\left[\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(x+1)\right]+\left(\frac{1}{2}\right)^{x+1}\cdot\left(\frac{1}{2}\right)^{n}\cdot(x+1)\right]\ldots\right]$$

$$+\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(n+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{n}\cdot(n+1)$$

$$+\sum_{y=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{y+1}\cdot(n+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\left(\frac{1}{2}\right)^{n}\cdot(n+1)$$

$$Q_i := \eta q\cdot\frac{I_0\left(t_{p0}\right)-v_{th_i}}{\sqrt{S_o\cdot noise}} \qquad Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{I_1\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{e0}(b,i) := \frac{767.25}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right) \qquad P_{s0}(b,i) := \frac{767.25}{8192}\left(\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

x1 = 1..n, y = 1, x = 1..n

$$\left[\sum_{x1=1}^{n-1}\left[\sum_{x=1}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\dots\right]\cdot 8192 = 383.5$$
$$\left[+\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]$$

$$Q_i := \eta q\cdot\frac{v_{oRNS2}\left(t_{pRNS2}\right) - v_{th_i}}{\sqrt{S_o\cdot noise}}$$

$$Q_{S_i} := \eta q\cdot\frac{1}{2}\cdot\frac{I_1\left(t_{d0_i}\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{er3}(b,i) := \frac{383.5}{8192}\cdot\left(\frac{1}{2}\cdot\operatorname{erfc}\left(\frac{b\cdot Q_i}{\sqrt{2}}\right)\right)$$

$$P_{s3}(b,i) := \frac{383.25}{8192}\cdot\left(\frac{1}{2}\cdot\operatorname{erfc}\left(\frac{b\cdot Q_{S_i}}{\sqrt{2}}\right)\right)$$

$$P_{erS}(b,i) := P_{er1}(b,i) + P_{er2}(b,i) + P_{er3}(b,i)$$

$$P_{sS}(b,i) := P_{s1}(b,i) + P_{s2}(b,i) + P_{s3}(b,i)$$

$$P_{eeS}(b,i) := P_{eSinRS}(b,i) + P_{eSinSR}(b,i) + P_{erS}(b,i)$$

$$P_{wsS}(b,i) := P_{sSinRS}(b,i) + P_{sSinSR}(b,i) + P_{sS}(b,i)$$

$$P_{eeS}(b,10) = 0.178$$

$$P_{wsS}(b,10) = 5.517\times 10^{-3}$$

$$P_{eeR}(b,10) = 0.179$$

$$P_{wsR}(b,10) = 0.022$$

$$P_{er}(b,i) := P_{eeS}(b,i) + P_{eeR}(b,i)$$

$$P_{ws}(b,i) := P_{wsS}(b,i) + P_{wsR}(b,i)$$

False alarm - Sequence S x1N R yN S xN R - FALSE ALARM OF R IN S xN R

x1 = 0, y = 0, x = 1, k = 1, ONE error, N to R in SRS N R

$$\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot 1\right]\cdot 8192 = 128$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oSRSNR}\left(t_{d0_i} + 2.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{f1}(b,i) := \frac{128}{8192} \cdot \left(\frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

x1 = 1, y = 0, x = 1, k = 1, ONE error, N to R in SNRS N R

$$\left[\frac{1}{4}\left(\frac{1}{2}\right)^1 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^1 \cdot \frac{1}{2} \cdot 1\right] \cdot 8192 = 64$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oSNRSNR}\left(t_{d0_i} + 2.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{f2}(b,i) := \frac{64}{8192} \cdot \left(\frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

x1 = 2..n, y = 0, x = 1, k = 1, ONE error, N to R

$$\left[\sum_{x1=2}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{x1} \cdot \frac{1}{2}\left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^1 \cdot \frac{1}{2} \cdot 1\right] + \frac{1}{4}\left(\frac{1}{2}\right)^n \cdot 1 \cdot \left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^1 \cdot \frac{1}{2} \cdot 1\right] \cdot 8192 = 64$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oRSNR}\left(t_{d0_i} + 2.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{f3}(b,i) := \frac{64}{8192} \cdot \left(\frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

$$P_{fRSRR}(b,i) := P_{f1}(b,i) + P_{f2}(b,i) + P_{f3}(b,i)$$

y = 0, k = 1, x1 and x variable N to R in S xN R as part of S x1N R S xN R

x1 = 0, y = 0, x = 2..n, k = 1

$$\left[\sum_{x=2}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^x \cdot \frac{1}{2}(x - 1 + 1)\right] + \frac{1}{4}\left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^0 \cdot \frac{1}{2}\left(\frac{1}{2}\right)^n \cdot 1 \cdot (n - 1 + 1)\right] \cdot 8192 = 383.5$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oSRS2}\left(t_{d0_i} + 2.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{f1}(b, i) := \frac{383.5}{8192}\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

x1 = 1, y = 0, x = 2..n, k = 1

$$\left[\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x-1+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n-1+1)\right]\cdot 8192 = 191.75$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oSNRS2}\left(t_{d0_i} + 2.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{f1}(b, i) := \frac{191.75}{8192}\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

x1 = 2..n, y = 0, x = 2..n, k = 1

$$\left[\sum_{x1=2}^{n-1}\left[\left[\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x-1+1)\right]\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{x1}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n-1+1)\right]\ldots\right]\cdot 8192 = 191.75$$
$$\left[+\sum_{x=2}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{x}\cdot\frac{1}{2}\cdot(x-1+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left(\frac{1}{2}\right)^{0}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n-1+1)\right]$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oRS2}\left(t_{d0_i} + 2.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{f3}(b, i) := \frac{191.75}{8192}\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

$$P_{fRSR}(b, i) := P_{f1}(b, i) + P_{fX}(b, i) + P_{f3}(b, i)$$

y = 0, x = 2..n, k = x, ONE error, N to R, S x1N R S xN R

$$\left[\sum_{x=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{0+2}\cdot 1\right]+\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{0+2}\cdot 1\right]\cdot 8192 = 256$$

$$Q_{R_i} := \eta q\cdot\frac{v_{th_i}-0}{\sqrt{S_o\cdot noise}}$$

$$P_{fR\_R}(b,i) := \frac{256}{8192}\cdot\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

y = 0, x = 3..n, k = 2..x-1, errors, N to R, S x1N R S xN R

$$\left[\sum_{x=3}^{n-1}\sum_{k=2}^{x-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{0+2}\cdot(x-k+1)\right]+\sum_{k=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{0+2}\cdot(n-k+1)\right]\right]\cdot 8192 = 757$$

$$Q_{R_i} := \eta q\cdot\frac{v_{th_i}}{\sqrt{S_o\cdot noise}}$$

$$P_{fNR}(b,i) := \frac{757}{8192}\cdot\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

$$P_{efNtoR\_RSNR}(b,i) := P_{fRSRR}(b,i) + P_{fRSR}(b,i) + P_{fR\_R}(b,i) + P_{fNR}(b,i)$$

$$P_{efNtoR\_RSNR}(b,10) = 0.021$$

y = 1..n, x = 1, k = 1, ONE error, N to R, S x1N R yN S xN R

$$\left[\sum_{y=1}^{n-1}\left[\left(\frac{1}{2}\right)^{1+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot 1\right]+\left(\frac{1}{2}\right)^{1+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot 1\right]\cdot 8192 = 256$$

$$Q_{R_i} := \eta q\cdot\frac{v_{th_i}-v_{oSNR}\left(t_{d0_i}-2.5\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{fSRR}(b,i) := \frac{256}{8192}\cdot\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

y = 1..n, x = 2..n, k = 1, errors, N to R, S x1N R yN S xN R

$$\left[\sum_{y=1}^{n-1}\left[\sum_{x=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot(x-1+1)\right]+\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot(n-1+1)\right]\ldots\right]\cdot 8192 = 767.5$$
$$\left[+\sum_{x=1}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot(x-1+1)\right]+\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot(n-1+1)\right]$$

$$Q_{R_i} := \eta q\cdot\frac{v_{th_i} - I_0\left(t_{d0_i}+3\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{fSNR}(b,i) := \frac{767.5}{8192}\cdot\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

y = 1..n, x = 2..n, k = x, ONE error

$$\left[\sum_{y=1}^{n-1}\left[\sum_{x=2}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot 1\right]+\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot 1\right]+\sum_{x=1}^{n-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot 1\right]+\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot 1\right]\cdot 8192 = 256.5$$

$$Q_{R_i} := \eta q\cdot\frac{v_{th_i} - 0}{\sqrt{S_o\cdot noise}}$$

$$P_{fR\_R}(b,i) := \frac{256.5}{8192}\cdot\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

y = 1..n, x = 3..n, k = 2..x-1, errors

$$\left[\sum_{y=1}^{n-1}\left[\sum_{x=3}^{n-1}\sum_{k=2}^{x-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot(x-k+1)\right]+\sum_{k=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{y+2}\cdot(n-k+1)\right]\right]\ldots\right]\cdot 8192 = 757$$
$$\left[+\left[\sum_{x=3}^{n-1}\sum_{k=2}^{x-1}\left[\left(\frac{1}{2}\right)^{x+2}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot(x-k+1)\right]+\sum_{k=2}^{n-1}\left[\left(\frac{1}{2}\right)^{n+1}\cdot\left(\frac{1}{2}\right)^{n+1}\cdot(n-k+1)\right]\right]\right]$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i}}{\sqrt{S_o \cdot noise}}$$

$$P_{fNR}(b,i) := \frac{757}{8192}\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

$$P_{efNtoR\_SNR}(b,i) := P_{fSRR}(b,i) + P_{fSNR}(b,i) + P_{fR\_R}(b,i) + P_{fNR}(b,i)$$

$$P_{efNtoR}(b,i) := P_{efNtoR\_RSNR}(b,i) + P_{efNtoR\_SNR}(b,i)$$

$$P_{efNtoR}(b,10) = 0.041$$

Sequence R yN S xN R - false S pulse in between S and R - NO ERRORS EXCEPT FOR FALSE S BEFORE THE LAST R - MAKES IT LOOK LIKE AN S

Sequence R y1N S xN R yN S - going to R y1N S xN S yN S

y1 = 0..n, x = 0, y = 0..n

$$\left[\sum_{y1=0}^{n-1}\left[\sum_{y=0}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]\dots\right]\cdot 8192 = 2.047\times 10^{3}$$
$$\left[+\sum_{y=0}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{y}\cdot\frac{1}{2}\cdot(y+1)\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\cdot(n+1)\right]$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - v_{oSR}\left(t_{d0_i} - 0.5\right)}{\sqrt{S_o \cdot noise}}$$

$$P_{fNtoSinSR}(b,i) := \frac{2047}{8192}\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right) \qquad y1 = 0..n,\ x = 1..n,\ y = 0..n$$

$$\left[\sum_{y1=0}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\left[\sum_{x=1}^{n-1}\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{y+1}(y+1)\right]+\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{n}(n+1)\right]\ldots\right]+\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{y+1}(y+1)\right]+\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{n}(n+1)\right]\right]\right]\ldots\right]\cdot 8192 = 2.047\times 10^{3}$$

$$+\frac{1}{4}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left[\left[\sum_{x=1}^{n-1}\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{y+1}(y+1)\right]+\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{n}(n+1)\right]\ldots\right]+\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{y+1}(y+1)\right]+\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{n}(n+1)\right]\right]\right]$$

$$Q_{R_i} := \eta q\cdot\frac{v_{th_i} - I_0\left(t_{d0_i} - 0.5\right)}{\sqrt{S_o\cdot noise}}$$

$$P_{fNtoSinR}(b,i) := \frac{2047}{8192}\left(\frac{T_s}{\tau_R}\cdot\frac{1}{2}\cdot erfc\left(\frac{b\cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

Sequence S xN R yN S - false R pulse in between R and S - NO ERRORS

False alarm - Sequence R y1N S xN R yN S - FALSE ALARM OF S IN R xN S

Effectively pulse in isolation

Take R pulse and slot immediately after it - k = 0

y1 = 0..n, x = 0..n, y = 0..n, k = 0 - errors of y

$$\left[\sum_{y1=0}^{n-1}\left[\frac{1}{4}\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\left[\sum_{x=0}^{n-1}\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{y+1}(y)\right]+\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{n}(n)\right]\ldots\right]+\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{y+1}(y)\right]+\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{n}(n)\right]\right]\right]\ldots\right]\cdot 8192 = 2.045\times 10^{3}$$

$$+\frac{1}{4}\left(\frac{1}{2}\right)^{n}\cdot 1\cdot\left[\left[\sum_{x=1}^{n-1}\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{y+1}(y)\right]+\left(\frac{1}{2}\right)^{x+1}\left(\frac{1}{2}\right)^{n}(n)\right]\ldots\right]+\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{y+1}(y)\right]+\left(\frac{1}{2}\right)^{n}\left(\frac{1}{2}\right)^{n}(n)\right]\right]\right]$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - I_0\left(t_{d0_i} + 0.5\right)}{\sqrt{S_o \cdot noise}}$$

THIS GIVES ERRORS

$$P_{fRfalse}(b, i) := \frac{2045}{8192}\left(\frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

Take R pulse and slots after it - k = 1..n

y1 = 0..n, x = 0..n, y = 0..n, k = 1..n - errors of y-k+1

$$\left[\sum_{y1=0}^{n-1}\left[\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{y1}\cdot\frac{1}{2}\cdot\left[\sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\cdot\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{y+1}\cdot\sum_{k=1}^{y}(y-k+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\sum_{k=1}^{n}(n-k+1)\right]\right]\cdots\right]\cdots\right]+\left(\frac{1}{2}\right)^{n}\cdot\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{y+1}\cdot\sum_{k=1}^{y}(y-k+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\sum_{k=1}^{n}(n-k+1)\right]\right]+\frac{1}{4}\cdot\left(\frac{1}{2}\right)^{n}\cdot 1\left[\left[\sum_{x=0}^{n-1}\left[\left(\frac{1}{2}\right)^{x+1}\cdot\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{y+1}\cdot\sum_{k=1}^{y}(y-k+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\sum_{k=1}^{n}(n-k+1)\right]\right]\cdots\right]+\left(\frac{1}{2}\right)^{n}\cdot\left[\sum_{y=0}^{n-1}\left[\left(\frac{1}{2}\right)^{y+1}\cdot\sum_{k=1}^{y}(y-k+1)\right]+\left(\frac{1}{2}\right)^{n}\cdot\sum_{k=1}^{n}(n-k+1)\right]\right]\right]\cdot 8192 = 5.096\times 10^{3}$$

$$Q_{R_i} := \eta q \cdot \frac{v_{th_i} - 0}{\sqrt{S_o \cdot noise}}$$

$$P_{ffalse}(b, i) := \frac{5096}{8192}\left(\frac{T_s}{\tau_R} \cdot \frac{1}{2} \cdot erfc\left(\frac{b \cdot Q_{R_i}}{\sqrt{2}}\right)\right)$$

$$P_{efNtos}(b, i) := P_{fNtoSinSR}(b, i) + P_{fNtoSinR}(b, i) + P_{fRfalse}(b, i) + P_{ffalse}(b, i)$$

$$P_{ef}(b, i) := P_{efNtoR}(b, i) + P_{efNtos}(b, i)$$

$$P_{eb}(b, i) := P_{ef}(b, i) + P_{er}(b, i) + P_{ws}(b, i)$$

10.Reed Solomon Code

$m := 5$ $t := 4$ $r := \frac{2^m - 1 - 2 \cdot t}{2^m - 1}$ $r = 0.742$

$$pe(b,i) := \frac{1}{2^m - 1} \cdot \sum_{j=t+1}^{2^m-1} \left[j \cdot \left[\frac{(2^m - 1)!}{j! \cdot (2^m - 1)! - (j-1)!} \cdot (P_{eb}(b,i))^{j-1} \cdot (1 - P_{eb}(b,i))^{(2^m-1)-(j-1)} \right] \right]$$

$pc(b,i) := (\log(pe(b,i)) + 9)$ Set for 1 in 10^9 errors

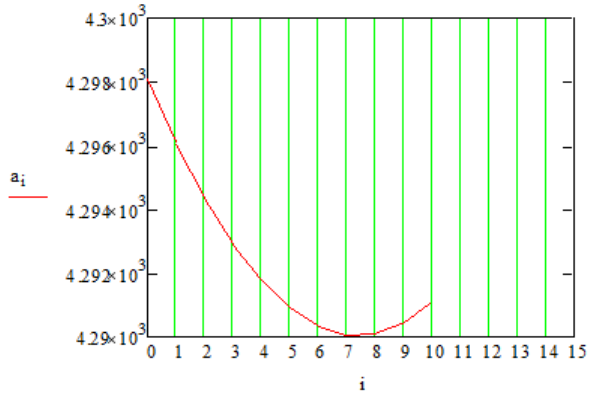
$P_{ef}(b,10) = 0.414$ $P_{er}(b,10) = 0.357$ $P_{ws}(b,10) = 0.028$

11.Number of photons and transmission efficiency

$a_i := root(pc(b,i), b)$ Find the root to give 1 in 10^9 $\rho(a) := \frac{r \cdot \ln(2)}{a}$

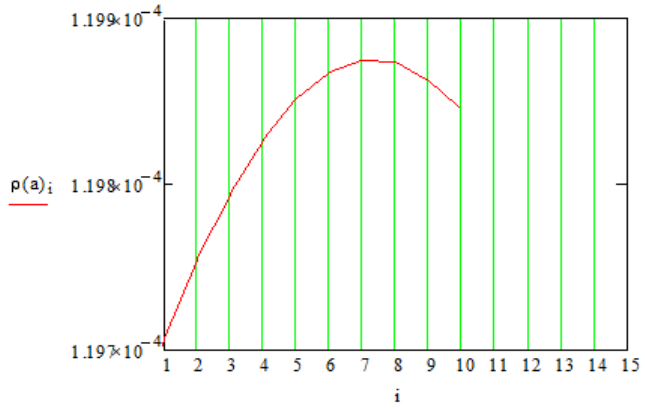
minimum= min(a) $b \equiv 2.110^3$ $v_{off} \equiv 0.847$ $number_i := a_i$

minimum= 4.29×10^3 range ≡ 1000 $f_n \equiv 1.8$ $\min(number \cdot 10^{-3}) = 4.29$



a =

	0
0	$4.298 \cdot 10^3$
1	$4.296 \cdot 10^3$
2	$4.294 \cdot 10^3$
3	$4.293 \cdot 10^3$
4	$4.292 \cdot 10^3$
5	$4.291 \cdot 10^3$
6	$4.29 \cdot 10^3$
7	$4.29 \cdot 10^3$
8	$4.29 \cdot 10^3$
9	$4.29 \cdot 10^3$
10	$4.291 \cdot 10^3$



ρ(a) =

	0
0	$1.197 \cdot 10^{-4}$
1	$1.197 \cdot 10^{-4}$
2	$1.198 \cdot 10^{-4}$
3	$1.198 \cdot 10^{-4}$
4	$1.198 \cdot 10^{-4}$
5	$1.199 \cdot 10^{-4}$
6	$1.199 \cdot 10^{-4}$
7	$1.199 \cdot 10^{-4}$
8	$1.199 \cdot 10^{-4}$
9	$1.199 \cdot 10^{-4}$
10	$1.198 \cdot 10^{-4}$

$i := 7$ $v_i = 0.854$ $t_{p0} = 3.203 \times 10^{-3}$ $t_{pmin} = 3.2 \times 10^{-3}$

b := minimum

$b = 4.29 \times 10^{3}$

$P_{ef}(b, i) = 0.067$

$P_{er}(b, i) = 0.223$

$P_{efNtoR\_SNR}(b, i) \cdot 10^{10} = 1.009 \times 10^{5}$

$P_{efNtoR\_RSNR}(b, i) \cdot 10^{10} = 1.009 \times 10^{5}$

$P_{fNtoSinSR}(b, i) \cdot 10^{10} = 5.572 \times 10^{5}$

$P_{fNtoSinR}(b, i) \cdot 10^{10} = 5.572 \times 10^{5}$

$P_{fRfalse}(b, i) \cdot 10^{10} = 6.666 \times 10^{8}$

$P_{ffalse}(b, i) \cdot 10^{10} = 2.525 \times 10^{5}$

$P_{eeR}(b, i) \cdot 10^{10} = 1.117 \times 10^{9}$

$P_{erRinRSxNRS}(b, i) \cdot 10^{10} = 1.368 \times 10^{8}$

$P_{erRinRSxNRyNS}(b, i) \cdot 10^{10} = 4.219 \times 10^{8}$

$P_{erRinSRNS}(b, i) \cdot 10^{10} = 2.783 \times 10^{8}$

$P_{erRinRNSNRS}(b, i) \cdot 10^{10} = 6.841 \times 10^{7}$

$P_{erRinSNRNS}(b, i) \cdot 10^{10} = 2.12 \times 10^{8}$

$P_{eeS}(b,i)\cdot 10^{10} = 1.113\times 10^{9}$

$P_{eSinRS}(b,i)\cdot 10^{10} = 5.466\times 10^{8}$

$P_{eSinSR}(b,i)\cdot 10^{10} = 1.401\times 10^{8}$

$P_{erS}(b,i)\cdot 10^{10} = 4.261\times 10^{8}$

$P_{wS}(b,i) = 2.127\times 10^{-5}$

$P_{wsR}(b,i)\cdot 2\cdot 10^{10} = 3.393\times 10^{5}$

$P_{sRinRSxNRS}(b,i)\cdot\left(2\cdot 10^{10}\right) = 2.027\times 10^{4}$

$P_{sRinRSxNRyNS}(b,i)\cdot\left(2\cdot 10^{10}\right) = 6.455\times 10^{4}$

$P_{sRinSRNS}(b,i)\cdot\left(2\cdot 10^{10}\right) = 4.241\times 10^{4}$

$P_{sRinRNSNRS}(b,i)\cdot\left(2\cdot 10^{10}\right) = 1.014\times 10^{4}$

$P_{sRinSNRNS}(b,i)\cdot\left(2\cdot 10^{10}\right) = 3.227\times 10^{4}$

10.2. Appendix 2

10.2.1. DiPPM Matlab simulation.

```
%% DiPPM Coder
%% Initialaization
clear all
close all
clc
%% step one: generating a random binary PCM signal and clock
seq_length = 30;
PCM_seq = randi([0 1], 1, seq_length);
clock_seq = repmat([1 0],1,seq_length/2);
%% DiPPM encoder
%input: PCM_signal
%output: DiPPM_signal
DiPPM_seq = DiPPM_Encoder_B(PCM_seq);
%% DiPPM Decoder
%input: DiPPM_seq
%output: PCM_Decoded_seq
PCM_Decoded_seq = zeros(1,length(DiPPM_seq)/2);
flag1 = false;
flag2 = true;
for ii=3:2:length(DiPPM_seq)
    pcm_index = (ii+1)/2;
    if(DiPPM_seq(ii) == DiPPM_seq(ii+1))
        if(pcm_index>1)
            PCM_Decoded_seq(pcm_index) = PCM_Decoded_seq(pcm_index-1);
        end
    elseif(DiPPM_seq(ii) == 0 && DiPPM_seq(ii+1)==1)
        flag1 = true;
        if(pcm_index>1)
            PCM_Decoded_seq(pcm_index-1)=1;
        end
        PCM_Decoded_seq(pcm_index)=0;
    elseif(DiPPM_seq(ii) == 1 && DiPPM_seq(ii+1)==0)
        flag1 = true;
        if(pcm_index>1)
            PCM_Decoded_seq(pcm_index-1)=0;
        end
        PCM_Decoded_seq(pcm_index)=1;
    end
    if(flag1&&flag2)
        flag1 = false;
        flag2 = false;
        PCM_Decoded_seq(1:pcm_index-1)=PCM_Decoded_seq(pcm_index-1);
    end
end
%% converting form binary to pulses
clock_freq = 1*10^9;
pulse_width = 1/(2*clock_freq);
t0=0;
t1 = pulse_width * seq_length;
samples_per_pulse = 200;
```

```
total_no_samples = samples_per_pulse * seq_length;
t = t0:(t1/total_no_samples):t1-(t1/total_no_samples);
pulse_width_DiPPM = pulse_width/2;
samples_per_pulse_DiPPM = samples_per_pulse/2;
PCM_signal = zeros(size(t));
DiPPM_signal = zeros(size(t));
for i=1:length(PCM_signal)
    PCM_signal(i)=PCM_seq(ceil(i/samples_per_pulse));
    clock_signal(i)=clock_seq(ceil(i/samples_per_pulse));
    DiPPM_signal(i)=DiPPM_seq(ceil(i/samples_per_pulse_DiPPM));
    PCM_Decoded_signal(i)=PCM_Decoded_seq(ceil(i/samples_per_pulse));
end
%% Ploting the signals:
figure(1)
subplot(3,1,1),plot(t,clock_signal,'r','LineWidth',1.5);
axis([0 1.05*t1 0 1.5]);
title('Clock Signal');
grid on
grid minor
subplot(3,1,2), plot(t,PCM_signal,'LineWidth',1.5);
axis([0 1.05*t1 0 1.5]);
title('PCM Signal');
grid on
grid minor
subplot(3,1,3), plot(t,DiPPM_signal,'LineWidth',1.5);
axis([0 1.05*t1 0 1.5]);
title('DiPPM Signal');
grid minor
grid on
figure(2)
subplot(3,1,1),plot(t,clock_signal,'r','LineWidth',1.5);
axis([0 1.05*t1 0 1.5]);
title('Clock Signal');
grid on
grid minor
subplot(3,1,2), plot(t,DiPPM_signal,'LineWidth',1.5);
axis([0 1.05*t1 0 1.5]);
title('DiPPM Signal');
grid minor
grid on
subplot(3,1,3), plot(t,PCM_Decoded_signal,'LineWidth',1.5);
axis([0 1.05*t1 0 1.5]);
title('PCM Decoded Signal');
grid minor
grid on
%% ploting using stairs
figure(3)
subplot(4,1,1),stairs(clock_seq,'r','LineWidth',1.5);
axis([1 seq_length 0 1.5]);
    title('Clock Signal');
    grid on
    grid minor
    subplot(4,1,2), stairs(PCM_seq,'LineWidth',1.5);
    axis([1 seq_length 0 1.5]);
    title('PCM Signal');
    grid on
    grid minor
```

```
108.      subplot(4,1,3), stairs(DiPPM_seq,'LineWidth',1.5);
109.      axis([1 seq_length*2 0 1.5]);
110.      title('DiPPM Signal');
111.      grid minor
112.      grid on
113.      subplot(4,1,4), stairs(PCM_Decoded_seq,'LineWidth',1.5);
114.      axis([1 seq_length 0 1.5]);
115.      title('PCM Decoded Signal');
116.      grid minor
117.      grid on
```

10.2.2. Function of DiPPM coder.

```
function DiPPM_seq = DiPPM_Encoder_B(PCM_seq)
%% DiPPM encoder
%input: PCM_signal
%output: DiPPM_signal
seq_length = length(PCM_seq);
DiPPM_seq = zeros(1,seq_length*2);
for i=2:seq_length

    if(PCM_seq(i) == PCM_seq(i-1))
        DiPPM_seq(2*i-1) = 0;
        DiPPM_seq(2*i) = 0;
    elseif(PCM_seq(i) == 0 && PCM_seq(i-1)==1)
        DiPPM_seq(2*i-1) = 0;
        DiPPM_seq(2*i) = 1;
    elseif(PCM_seq(i) == 1 && PCM_seq(i-1)==0)
        DiPPM_seq(2*i-1) = 1;
        DiPPM_seq(2*i) = 0;
    end
end
end
```

10.2.3. Function of DiPPM decoder.

```
%% DiPPM Decoder
%input: DiPPM_seq
%output: PCM_Decoded_seq
function PCM_Decoded_seq = DiPPM_Decoder_B(DiPPM_seq)
PCM_Decoded_seq = zeros(1,length(DiPPM_seq)/2);
flag1 = false;
flag2 = true;
for ii=3:2:length(DiPPM_seq)

    pcm_index = (ii+1)/2;
    if(DiPPM_seq(ii) == DiPPM_seq(ii+1))
        if(pcm_index>1)
            PCM_Decoded_seq(pcm_index) = PCM_Decoded_seq(pcm_index-1);
        end
    elseif(DiPPM_seq(ii) == 0 && DiPPM_seq(ii+1)==1)
        flag1 = true;
        if(pcm_index>1)
            PCM_Decoded_seq(pcm_index-1)=1;
        end
        PCM_Decoded_seq(pcm_index)=0;
    elseif(DiPPM_seq(ii) == 1 && DiPPM_seq(ii+1)==0)
        flag1 = true;
        if(pcm_index>1)
            PCM_Decoded_seq(pcm_index-1)=0;
        end
        PCM_Decoded_seq(pcm_index)=1;
    end
    if(flag1&&flag2)
        flag1 = false;
        flag2 = false;
        PCM_Decoded_seq(1:pcm_index-1)=PCM_Decoded_seq(pcm_index-1);
    end

end
```

10.2.4. DiPPM & RS Matlab Simulation.

```
%% Complete RS_DiPPM Tx Rx system.
%% Initialaization
clear all
close all
clc


%% Transmitter Side Tx

% Parameters:

k=23;
m = 5;
n = 2^m -1;
seq_length = 300;
pri_poly = 37;


% step 1: integer massage generator
biggest_int = 2^m -1;
PCM_seq = randi([0 biggest_int], 1, seq_length);


% Step 2:The RS Encoding
msg = gf(PCM_seq(1:k),m);
genpoly = rsgenpoly(n,k);
RS_code = rsenc(msg,n,k,genpoly);


% Step 3: converting to binary
RS_code_dec = gf2dec(RS_code,m,pri_poly);
RS_code_bin = dec2bin(RS_code_dec);
% RS_code_bin_2 = reshape(RS_code_bin,1,m*n);
RS_code_bin_3 = zeros(1,155);
tt=1;
for i=1:n
    for j=1:5
        RS_code_bin_3(tt) = str2double(RS_code_bin(i,j));
        tt = tt+1;
    end
end


% Step 4: DiPPM Encoding
DiPPM_seq = DiPPM_Encoder_B(RS_code_bin_3);


%% Channel

%changing bitwise
random_seq = ones(size(DiPPM_seq));
number_of_changes = 4;
```

```
random_indices = randi([1 length(random_seq)], 1, number_of_changes);
% DiPPM_seq(random_indices) = not(DiPPM_seq(random_indices));

%changing sample wise;
noc = 4;
random_indices_1 = zeros(1,noc);
interval_step = floor(length(DiPPM_seq)/noc);
for ii=1:noc
    temp = randi([((ii-1)*interval_step)+1 ii*interval_step]);
    random_indices_1(ii) = temp;
end
DiPPM_seq(random_indices_1) = not(DiPPM_seq(random_indices_1));

%% Reciver Side Rx

% Step 1: DiPPM_Decoder
DiPPM_decoded_seq = DiPPM_Decoder_B(DiPPM_seq);

DiPPM_decoded_seq_1 = zeros(n,m);
% Step 2: Convert to Decimal sequence
% DiPPM_decoded_seq_1 = reshape(DiPPM_decoded_seq,n,m);
tt=1;
for i=1:n
    for j=1:5
    DiPPM_decoded_seq_1(i,j) = (DiPPM_decoded_seq(tt));
    tt = tt+1;
    end
end
DiPPM_decoded_seq_dec = num2str(DiPPM_decoded_seq_1);
DiPPM_decoded_seq_dec = bin2dec(DiPPM_decoded_seq_dec);

% Step 3: RS_Decoder
msg2  = gf(DiPPM_decoded_seq_dec,m);
[RS_decoded_seq,cnumerr] = rsdec(msg2',n,k,genpoly);

RS_decoded_dec = gf2dec(RS_decoded_seq,m,pri_poly);


%% Display the error between the original message and the received
figure(1)
subplot(2,1,1), stairs(PCM_seq(1:k),'LineWidth',2);
title('original msg (blue) vs RS decoded');
hold on
stairs(RS_decoded_dec,'r','LineWidth',1.5);
grid on
    subplot(2,1,2), stairs(PCM_seq(1:k)-
  RS_decoded_dec,'LineWidth',1.5);
    grid on

    figure(2)
    subplot(2,1,1), stairs(DiPPM_decoded_seq_dec(1:k),'LineWidth',2);
    hold on
    title('DiPPM decoded(blue) vs RS decoded');
    stairs(RS_decoded_dec,'r','LineWidth',1.5);
    grid on
```

```
109.      subplot(2,1,2), stairs(DiPPM_decoded_seq_dec(1:k)'-
   RS_decoded_dec,'LineWidth',1.5);
110.      grid on
111.
112.      figure(3)
113.      stairs(RS_code_dec - DiPPM_decoded_seq_dec','k','LineWidth',1.5);
114.      title('difference between sent RS code and the received code');
115.      grid minor
116.
117.      %% drawing the pulses Tx
118.      k2 = 30;
119.      clock_seq = repmat([1 0],1,(m*k2)/2);
120.
121.      % convert from decimal to binary
122.      PCM_binary = dec2bin(PCM_seq(1:k2));
123.      % PCM_binary = reshape(PCM_binary,1,m*k2);
124.      PCM_bin_2 = zeros(1,k2*m);
125.      tt=1;
126.      for i=1:length(PCM_binary)
127.          for j=1:5
128.          PCM_bin_2(tt) = str2double(PCM_binary(i,j));
129.          tt = tt+1;
130.          end
131.      end
132.
133.
134.      clock_freq = 1*10^9;
135.      pulse_width = 1/(2*clock_freq);
136.      t0=0;
137.      t1 = 2*pulse_width * m*k2;
138.      samples_per_pulse = 50;
139.
140.      total_no_samples = samples_per_pulse * m*k2;
141.      t = t0:(t1/total_no_samples):t1-(t1/total_no_samples);
142.      %
143.      pulse_width_DiPPM = pulse_width/2;
144.      samples_per_pulse_DiPPM = samples_per_pulse/2;
145.      %
146.      clock_signal = zeros(size(t));
147.      PCM_signal = zeros(size(t));
148.      RS_signal = zeros(size(t));
149.      DiPPM_signal = zeros(size(t));
150.      for i=1:length(PCM_signal)
151.          PCM_signal(i)=PCM_bin_2(ceil(i/samples_per_pulse));
152.          clock_signal(i)=clock_seq(ceil(i/samples_per_pulse));
153.          RS_signal(i) = RS_code_bin_3(ceil(i/samples_per_pulse));
154.          DiPPM_signal(i)=DiPPM_seq(ceil(i/samples_per_pulse_DiPPM));
155.      %
   PCM_Decoded_signal(i)=PCM_Decoded_seq(ceil(i/samples_per_pulse));
156.
157.      end
158.
159.      figure(4)
160.      subplot(5,1,1),plot(t,clock_signal,'r','LineWidth',1.5);
161.      axis([0 1.05*t1 0 1.5]);
162.      title('Clock Signal');
163.      grid on
```

```
164.      grid minor
165.      subplot(5,1,2), plot(t,PCM_signal,'LineWidth',1.5);
166.      axis([0 1.05*t1 0 1.5]);
167.      title('PCM Signal');
168.      grid on
169.      grid minor
170.      subplot(5,1,3), plot(t,RS_signal,'LineWidth',1.5);
171.      axis([0 1.05*t1 0 1.5]);
172.      title('RS Signal');
173.      grid on
174.      grid minor
175.      subplot(5,1,4), plot(t,abs(RS_signal-
   PCM_signal),'LineWidth',1.5);
176.      axis([0 1.05*t1 0 1.5]);
177.      title('RS Signal - PCM signal');
178.      grid on
179.      grid minor
180.      subplot(5,1,5), plot(t,DiPPM_signal,'LineWidth',1.5);
181.      axis([0 1.05*t1 0 1.5]);
182.      title('DiPPM Signal');
183.      grid minor
184.      grid on
185.      %%  drawing the pulses Rx
186.      % convert from decimal to binary
187.      RS_decoded_binary = dec2bin(RS_decoded_dec);
188.      % PCM_binary = reshape(PCM_binary,1,m*k2);
189.      RS_decoded_binary_1 = zeros(1,k2*m);
190.      tt=1;
191.      for i=1:k
192.          for j=1:5
193.          RS_decoded_binary_1(tt) = str2double(RS_decoded_binary(i,j));
194.          tt = tt+1;
195.          end
196.      end
197.
198.
199.      DiPPM_decoded_signal = zeros(size(t));
200.      RS_decoded_signal = zeros(size(t));
201.      % RS_signal = zeros(size(t));
202.      DiPPM_signal = zeros(size(t));
203.      for i=1:length(PCM_signal)
204.
   DiPPM_decoded_signal(i)=DiPPM_decoded_seq(ceil(i/samples_per_pulse));
205.          RS_decoded_signal(i) =
   RS_decoded_binary_1(ceil(i/samples_per_pulse));
206.      %     DiPPM_signal(i)=DiPPM_seq(ceil(i/samples_per_pulse_DiPPM));
207.      %
   PCM_Decoded_signal(i)=PCM_Decoded_seq(ceil(i/samples_per_pulse));
208.
209.      end
210.
211.      figure(5)
212.      subplot(4,1,1),plot(t,DiPPM_decoded_signal,'LineWidth',1.5);
213.      axis([0 1.05*t1 0 1.5]);
214.      title('DiPPM Decoded Signal');
215.      grid on
216.      grid minor
```

```
217.      subplot(4,1,2),plot(t,abs(DiPPM_decoded_signal-
   RS_signal),'r','LineWidth',1.5);
218.      axis([0 1.05*t1 -.5 1.5]);
219.      title('DiPPM Decoded Signal - RS Signal');
220.      grid on
221.      grid minor
222.      subplot(4,1,3),plot(t,RS_decoded_signal,'LineWidth',1.5);
223.      axis([0 1.05*t1 0 1.5]);
224.      title('RS Decoded Signal');
225.      grid on
226.      grid minor
227.      subplot(4,1,4),plot(RS_decoded_signal(1:k*m)-
   PCM_signal(1:m*k),'LineWidth',1.5);
228.      % axis([0 1.05*t1 0 1.5]);
229.      title('RS Decoded vs PCM signal');
230.      grid on
231.      grid minor
 1.
```

10.2.5. Function of Galois field to decimal transformation.

```
1. function [DecOutput] = gf2dec(GFInput,m,prim_poly)
2. GFInput = GFInput(:)';% force a row vector
3. GFRefArray = gf([0:(2^m)-1],m,prim_poly);
4. for i=1:length(GFInput)
5.     for k=0:(2^m)-1
6.         temp = isequal(GFInput(i),GFRefArray(k+1));
7.         if (temp==1)
8.             DecOutput(i) = k;
9.         end
10.    end
11.end
```

10.3. Appendix 3

10.3.1. PRBS VHDL source code.

```
1. ================================================================================
2. -- Project Name: DiPPM & RS
3. -- Name: pbrs. vhd
4. ================================================================================
5. -- Description: Pseudo Random Binary Generator Sequence
6. ================================================================================
7. --         libraries
8. ================================================================================
9. library ieee;
10.use ieee.std_logic_1164.all;
11.use ieee.numeric_std.all;
12.library work;
13.use work.pbrs_pkg.all;
14.================================================================
15.--         TOP instantiation
16.================================================================
17.entity pbrs is
18.    port (
19.        CLK        : in  std_logic;                    -- system clock
20.        RESET      : in  std_logic;                    -- system reset
21.        -- outputs
22.        enable     : out std_logic;                    -- enable signal
23.        startpls   : out std_logic;                    -- start pulse
   signal
24.        dataOut    : out std_logic_vector(4 downto 0)  -- data out
25.);
26.end pbrs;
27.
28.
29.================================================================
30.--         RTL Architecture
31.================================================================
32.architecture rtl of pbrs is
33.----------------------------------------------------------------
34.-- Signals
35.----------------------------------------------------------------
36.    signal shiftreg        : std_logic_vector(4 downto 0);
37.    signal dataOut_inner   : std_logic_vector(4 downto 0);
38.    signal ctrl_cnt        : std_logic_vector(7 downto 0);
39.    signal enable_inner    : std_logic;
40.    signal startpls_inner  : std_logic;
41.begin
42.    ---------------------------------------------
43.    -- Write Pointer
44.    pbrs_p : process (CLK,RESET) is
45.    begin
46.        if RESET = '0' then
47.            shiftreg <= "10000";
48.            enable_inner   <= '0';
49.            ctrl_cnt <= (others => '0');
```

```
50.         elsif rising_edge(CLK) then
51.             --------------
52.             if (ctrl_cnt < "00011111") then
53.                 enable_inner   <= '1';
54.             else
55.                 enable_inner   <= '0';
56.             end if;
57.             if (ctrl_cnt < "00010111") then
58.                 shiftreg (0) <= shiftreg(1) xor shiftreg(2) xor
   shiftreg(4);
59.                 shiftreg (1) <= shiftreg (0);
60.                 shiftreg (2) <= shiftreg (1);
61.                 shiftreg (3) <= shiftreg (2);
62.                 shiftreg (4) <= shiftreg (3);
63.                 dataOut_inner <= shiftreg;
64.             else
65.                 dataOut_inner <= (others => '0');
66.             end if;
67.             if (ctrl_cnt = "10101010") then -- 10011011
68.                 ctrl_cnt <= "00000000";
69.             else
70.                 ctrl_cnt <=  std_logic_vector(unsigned(ctrl_cnt) + 1);
71.             end if;
72.             if (ctrl_cnt = "00000000") then
73.                 startpls_inner <= '1';
74.             else
75.                 startpls_inner <= '0';
76.             end if;
77.         end if;
78.     end process;
79.     -- Output ports
80.     enable   <= enable_inner;
81.     startpls <= startpls_inner;
82.     dataOut  <= dataOut_inner;
83. end rtl;
```

10.3.2. RS coder VHDL source code.

```
1. ================================================================
2. -- Project Name: RSIP
3. -- Name: rscoder_top_31_23. vhd
4. ================================================================
5. -- Description: RS (31,23) encoder module
6. ================================================================
7. --         libraries
8. ================================================================
9. library ieee;
10.use ieee.std_logic_1164.all;
11.use ieee.numeric_std.all;
12.library work;
13.use work.rscoder_31_23_top_pkg.all;
14.================================================================
15.--         TOP instantiation
16.================================================================
17.entity rscoder_31_23_top is
18.    port (
19.        CLK      : in  std_logic;                    -- system clock
20.        RESET    : in  std_logic;                    -- system reset
21.        enable   : in  std_logic;                 -- rs encoder enable
   signal
22.        startPls : in  std_logic;                 -- rs encoder sync
   signal
23.        dataIn   : in  std_logic_vector(4 downto 0); -- rs encoder data
   in
24.        -- Data output
25.        dataOut  : out std_logic_vector(4 downto 0)  -- rs encoder data
   out
26.);
27.end rscoder_31_23_top;
28.
29.================================================================
30.--         RTL Architecture
31.================================================================
32.architecture rtl of rscoder_31_23_top is
33.----------------------------------------------------------------
34.-- Signals
35.----------------------------------------------------------------
36.    signal count      : std_logic_vector(4 downto 0);
37.    signal dataValid : std_logic;
38.    signal feedbackReg: std_logic_vector(4 downto 0);
39.    signal mult_0: std_logic_vector(4 downto 0);
40.    signal mult_1: std_logic_vector(4 downto 0);
41.    signal mult_2: std_logic_vector(4 downto 0);
42.    signal mult_3: std_logic_vector(4 downto 0);
43.    signal mult_4: std_logic_vector(4 downto 0);
44.    signal mult_5: std_logic_vector(4 downto 0);
45.    signal mult_6: std_logic_vector(4 downto 0);
46.    signal mult_7: std_logic_vector(4 downto 0);
47.    signal syndromeReg_0: std_logic_vector(4 downto 0);
48.    signal syndromeReg_1: std_logic_vector(4 downto 0);
49.    signal syndromeReg_2: std_logic_vector(4 downto 0);
```

```
50.    signal syndromeReg_3: std_logic_vector(4 downto 0);
51.    signal syndromeReg_4: std_logic_vector(4 downto 0);
52.    signal syndromeReg_5: std_logic_vector(4 downto 0);
53.    signal syndromeReg_6: std_logic_vector(4 downto 0);
54.    signal syndromeReg_7: std_logic_vector(4 downto 0);
55.    signal dataReg: std_logic_vector(4 downto 0);
56.    signal syndromeRegFF: std_logic_vector(4 downto 0);
57.    signal wireOut: std_logic_vector(4 downto 0);
58.    signal dataOutInner : std_logic_vector(4 downto 0);
59.begin
60.    ---------------------------------------------
61.    -- count
62.    rs_count : process (CLK,RESET) is
63.    begin
64.        if RESET = '0' then
65.            count <= (others => '0');
66.        elsif rising_edge(CLK) then
67.            if (enable ='1') then
68.                if (startPls ='1') then
69.                    count <= "00001";
70.                elsif ((count ="00000") or (count ="11111")) then
71.                    count <= (others => '0');
72.                else
73.                    count <=  std_logic_vector(unsigned(count) + 1);
74.                end if;
75.            end if;
76.        end if;
77.    end process;
78.    ---------------------------------------------
79.    -- dataValid
80.    dataValid <= '1' when  ((startPls = '1') or (count < "10111")) else
   '0';
81.    ---------------------------------------------
82.    -- mulitpliers
83.    mult_7(0) <=  feedbackReg(3);
84.    mult_7(1) <=  feedbackReg(4);
85.    mult_7(2) <=  feedbackReg(0) xor feedbackReg(3);
86.    mult_7(3) <=  feedbackReg(1) xor feedbackReg(4);
87.    mult_7(4) <=  feedbackReg(2);
88.    mult_2(0) <=  feedbackReg(2);
89.    mult_2(1) <=  feedbackReg(3);
90.    mult_2(2) <=  feedbackReg(2) xor feedbackReg(4);
91.    mult_2(3) <=  feedbackReg(0) xor feedbackReg(3);
92.    mult_2(4) <=  feedbackReg(1) xor feedbackReg(4);
93.    mult_4(0) <=  feedbackReg(0) xor feedbackReg(1) xor feedbackReg(2)
   xor feedbackReg(3);
94.    mult_4(1) <=  feedbackReg(0) xor feedbackReg(1) xor feedbackReg(2)
   xor feedbackReg(3) xor feedbackReg(4);
95.    mult_4(2) <=  feedbackReg(0) xor feedbackReg(4);
96.    mult_4(3) <=  feedbackReg(0) xor feedbackReg(1);
97.    mult_4(4) <=  feedbackReg(0) xor feedbackReg(1) xor feedbackReg(2);
98.    mult_5(0) <=  feedbackReg(2) xor feedbackReg(3);
99.    mult_5(1) <=  feedbackReg(3) xor feedbackReg(4);
100.        mult_5(2) <=  feedbackReg(0) xor feedbackReg(2) xor
   feedbackReg(3) xor feedbackReg(4);
101.        mult_5(3) <=  feedbackReg(0) xor feedbackReg(1) xor
   feedbackReg(3) xor feedbackReg(4);
```

```
102.          mult_5(4) <=  feedbackReg(1) xor feedbackReg(2) xor
   feedbackReg(4);
103.          mult_6(0) <=  feedbackReg(2) xor feedbackReg(3);
104.          mult_6(1) <=  feedbackReg(3) xor feedbackReg(4);
105.          mult_6(2) <=  feedbackReg(0) xor feedbackReg(2) xor
   feedbackReg(3) xor feedbackReg(4);
106.          mult_6(3) <=  feedbackReg(0) xor feedbackReg(1) xor
   feedbackReg(3) xor feedbackReg(4);
107.          mult_6(4) <=  feedbackReg(1) xor feedbackReg(2) xor
   feedbackReg(4);
108.          mult_1(0) <=  feedbackReg(0) xor feedbackReg(2) xor
   feedbackReg(3) xor feedbackReg(4);
109.          mult_1(1) <=  feedbackReg(0) xor feedbackReg(1) xor
   feedbackReg(3) xor feedbackReg(4);
110.          mult_1(2) <=  feedbackReg(0) xor feedbackReg(1) xor
   feedbackReg(3);
111.          mult_1(3) <=  feedbackReg(0) xor feedbackReg(1) xor
   feedbackReg(2) xor feedbackReg(4);
112.          mult_1(4) <=  feedbackReg(1) xor feedbackReg(2) xor
   feedbackReg(3);
113.          mult_3(0) <=  feedbackReg(0) xor feedbackReg(2) xor
   feedbackReg(4);
114.          mult_3(1) <=  feedbackReg(0) xor feedbackReg(1) xor
   feedbackReg(3);
115.          mult_3(2) <=  feedbackReg(1);
116.          mult_3(3) <=  feedbackReg(0) xor feedbackReg(2);
117.          mult_3(4) <=  feedbackReg(1) xor feedbackReg(3);
118.          mult_0(0) <=  feedbackReg(1) xor feedbackReg(3);
119.          mult_0(1) <=  feedbackReg(0) xor feedbackReg(2) xor
   feedbackReg(4);
120.          mult_0(2) <=  feedbackReg(0);
121.          mult_0(3) <=  feedbackReg(1);
122.          mult_0(4) <=  feedbackReg(0) xor feedbackReg(2);
123.          ------------------------------------------
124.          -- syndromeReg
125.          rs_syndrome : process (CLK,RESET) is
126.          begin
127.              if RESET = '0' then
128.                  syndromeReg_0 <= (others => '0');
129.                  syndromeReg_1 <= (others => '0');
130.                  syndromeReg_2 <= (others => '0');
131.                  syndromeReg_3 <= (others => '0');
132.                  syndromeReg_4 <= (others => '0');
133.                  syndromeReg_5 <= (others => '0');
134.                  syndromeReg_6 <= (others => '0');
135.                  syndromeReg_7 <= (others => '0');
136.              elsif rising_edge(CLK) then
137.                  if (enable ='1') then
138.                      if (startPls ='1') then
139.                          syndromeReg_0 <= mult_0 (4 downto 0);
140.                          syndromeReg_1 <= mult_1 (4 downto 0);
141.                          syndromeReg_2 <= mult_2 (4 downto 0);
142.                          syndromeReg_3 <= mult_3 (4 downto 0);
143.                          syndromeReg_4 <= mult_4 (4 downto 0);
144.                          syndromeReg_5 <= mult_5 (4 downto 0);
145.                          syndromeReg_6 <= mult_6 (4 downto 0);
146.                          syndromeReg_7 <= mult_7 (4 downto 0);
```

```
147.                     else
148.     syndromeReg_0 <= mult_0 (4 downto 0);
149.     syndromeReg_1 <= syndromeReg_0(4 downto 0) xor mult_1(4 downto
   0);
150.     syndromeReg_2 <= syndromeReg_1(4 downto 0) xor mult_2(4 downto
   0);
151.     syndromeReg_3 <= syndromeReg_2(4 downto 0) xor mult_3(4 downto
   0);
152.     syndromeReg_4 <= syndromeReg_3(4 downto 0) xor mult_4(4 downto
   0);
153.     syndromeReg_5 <= syndromeReg_4(4 downto 0) xor mult_5(4 downto
   0);
154.     syndromeReg_6 <= syndromeReg_5(4 downto 0) xor mult_6(4 downto
   0);
155.     syndromeReg_7 <= syndromeReg_6(4 downto 0) xor mult_7(4 downto
   0);
156.                     end if;
157.                 end if;
158.             end if;
159.         end process;
160.
161.
162.         ---------------------------------------------
163.         -- feedbackReg
164.         feedbackReg <= (dataIn(4 downto 0)) when  (startPls = '1')
   else
165.                        (dataIn(4 downto 0) xor syndromeReg_7(4 downto
   0)) when  (dataValid = '1') else "00000";
166.         ---------------------------------------------
167.         -- dataReg syndromeRegFF
168.         rs_dataReg : process (CLK,RESET) is
169.         begin
170.             if RESET = '0' then
171.                 dataReg       <= (others => '0');
172.                 syndromeRegFF <= (others => '0');
173.             elsif rising_edge(CLK) then
174.                 if (enable ='1') then
175.                     dataReg       <= dataIn(4 downto 0);
176.                     syndromeRegFF <= syndromeReg_7(4 downto 0) ;
177.                 end if;
178.             end if;
179.         end process;
180.         ---------------------------------------------
181.         -- wireOut
182.         wireOut <= (dataReg(4 downto 0)) when  (count <= "10111")
   else syndromeRegFF(4 downto 0);
183.         ---------------------------------------------
184.         -- dataOutInner
185.         rs_dataOut : process (CLK,RESET) is
186.         begin
187.             if RESET = '0' then
188.                 dataOutInner <= (others => '0');
189.             elsif rising_edge(CLK) then
190.                 dataOutInner <= wireOut;
191.             end if;
192.         end process;
193.         ---------------------------------------------------------------
```

```
194.          -- Output ports
195.          dataOut <= dataOutInner;
196.      end rtl;
```

10.3.3. Parallel to searial bridge VHDL source code.

```
1. ================================================================
2. -- Project Name: bridge coder
3.  -- Name: bridgecoder_top. vhd
4.  --
   ================================================================
5.  -- Description: bridge for encoder
6.  --
   ================================================================
7.  --         libraries
8. ================================================================
9.  library ieee;
10. use ieee.std_logic_1164.all;
11. use ieee.numeric_std.all;
12. library work;
13. use work.bridgecoder_top_pkg.all;
14. use work.bridgecoder_dpram_pkg.all;
15.================================================================
16. --         TOP instantiation
17.================================================================
18. entity bridgecoder_top is
19.     port (
20.         CLK  : in  std_logic;                    -- system clock
21.         RESET: in  std_logic;                    -- system reset
22.        enable: in  std_logic;                   -- enable signal
23.      startPls: in  std_logic;                    -- sync signal
24.        dataIn: in  std_logic_vector(4 downto 0); -- data in
25.         -- Data output
26.       dataOut: out std_logic;                       -- data out
27.   startPlsOut: out std_logic;                  -- start pulse out
28.         enOut: out std_logic                      -- enable out
29. );
30. end bridgecoder_top;
31.================================================================
32. --         RTL Architecture
33.================================================================
34. architecture rtl of bridgecoder_top is
35. ----------------------------------------------------------------
36. -- Signals
37. ----------------------------------------------------------------
38.     signal enable_ff1  : std_logic;
39.     signal enable_ff2  : std_logic;
40.     signal enable_ff3  : std_logic;
41.     signal writePointer: std_logic_vector(4 downto 0);
42.     signal readPointer : std_logic_vector(4 downto 0);
43.     signal rdcnt_5clk  : std_logic_vector(2 downto 0);
44.     signal rdcnt_5clk_ff1 : std_logic_vector(2 downto 0);
45.     signal rdcnt_5clk_ff2 : std_logic_vector(2 downto 0);
46.     signal rd_enable   : std_logic;
47.     signal dpramRdData : std_logic_vector(4 downto 0);
48.     signal dataOutInner : std_logic;
49.     signal rd_enable_ff1 : std_logic;
50.     signal rd_enable_ff2 : std_logic;
51.     signal rd_enable_ff3 : std_logic;
```

```
52.     signal startPls_ff1  : std_logic;
53.     signal startPls_ff2  : std_logic;
54.     signal startPls_ff3  : std_logic;
55.     signal startPls_ff4  : std_logic;
56.     signal startPls_ff5  : std_logic;
57.     signal startPls_ff6  : std_logic;
58. begin
59.     ---------------------------------------------
60.     -- RAM memory instantiation
61.     u_bridgecoder_dpram : bridgecoder_dpram
62.     port map(
63.         w_clk   => CLK,
64.         w_en    => enable_ff2,
65.         w_addr  => writePointer,
66.         w_data  => dataIn,
67.         r_clk   => CLK,
68.         r_en    => rd_enable,
69.         r_addr  => readPointer,
70.         r_data  => dpramRdData
71.     );
72.     ---------------------------------------------
73.     -- write side process
74.     wr_side_p : process (CLK,RESET) is
75.     begin
76.         if RESET = '0' then
77.             writePointer <= (others => '0');
78.             enable_ff1 <= '0';
79.             enable_ff2 <= '0';
80.             enable_ff3 <= '0';
81.         elsif rising_edge(CLK) then
82.             enable_ff1 <= enable;
83.             enable_ff2 <= enable_ff1;
84.             enable_ff3 <= enable_ff2;
85.             -----------
86.             if (enable_ff2 ='1') then
87.                 if (writePointer = "11110") then
88.                     writePointer <= "00000";
89.                 else
90.                     writePointer <=
   std_logic_vector(unsigned(writePointer) + 1);
91.                 end if;
92.             end if;
93.         end if;
94.     end process;
95.     -- read side process
96.     rd_side_p : process (CLK,RESET) is
97.     begin
98.         if RESET = '0' then
99.             readPointer  <= (others => '0');
100.                rdcnt_5clk   <= (others => '0');
101.                rd_enable    <= '0';
102.                dataOutInner <= '0';
103.                rd_enable_ff1<= '0';
104.                rd_enable_ff2<= '0';
105.                rd_enable_ff3<= '0';
106.                startPls_ff1 <= '0';
107.                startPls_ff2 <= '0';
```

```
108.                    startPls_ff3 <= '0';
109.                    startPls_ff4 <= '0';
110.                    startPls_ff5 <= '0';
111.                    startPls_ff6 <= '0';
112.                elsif rising_edge(CLK) then
113.                    if ((enable_ff3='0') and (enable_ff2='1')) then
114.                        rd_enable <= '1';
115.                    elsif ((readPointer = "11110") and (rd_enable ='1')
    and (rdcnt_5clk = "100")) then
116.                        rd_enable <= '0';
117.                    end if;
118.                    if (rd_enable ='1') then
119.                       if (rdcnt_5clk = "100") then
120.                          rdcnt_5clk <= (others => '0');
121.                       else
122.                          rdcnt_5clk <=
    std_logic_vector(unsigned(rdcnt_5clk) + 1);
123.                       end if;
124.                    end if;
125.                    if ((rd_enable ='1') and (rdcnt_5clk = "100")) then
126.                        if (readPointer = "11110") then
127.                            readPointer <= "00000";
128.                        else
129.                            readPointer <=
    std_logic_vector(unsigned(readPointer) + 1);
130.                        end if;
131.                    end if;
132.                    rdcnt_5clk_ff1 <= rdcnt_5clk;
133.                    rdcnt_5clk_ff2 <= rdcnt_5clk_ff1;
134.                    if (rd_enable_ff2='1') then
135.                        case (rdcnt_5clk_ff2) is
136.                            when "000" => dataOutInner <=
    dpramRdData(4);
137.                            when "001" => dataOutInner <=
    dpramRdData(3);
138.                            when "010" => dataOutInner <=
    dpramRdData(2);
139.                            when "011" => dataOutInner <=
    dpramRdData(1);
140.                            when others => dataOutInner <=
    dpramRdData(0);
141.                        end case;
142.                    else
143.                        dataOutInner <= '0';
144.                    end if;
145.                    rd_enable_ff1 <= rd_enable;
146.                    rd_enable_ff2 <= rd_enable_ff1;
147.                    rd_enable_ff3 <= rd_enable_ff2;
148.                    startPls_ff1 <= startPls;
149.                    startPls_ff2 <= startPls_ff1;
150.                    startPls_ff3 <= startPls_ff2;
151.                    startPls_ff4 <= startPls_ff3;
152.                    startPls_ff5 <= startPls_ff4;
153.                    startPls_ff6 <= startPls_ff5;
154.                end if;
155.            end process;
156.
```

```
157.          -- Output ports
158.          dataOut      <= dataOutInner;
159.          startPlsOut  <= startPls_ff6;
160.          enOut        <= rd_enable_ff3;
161.      end rtl;
```

10.3.4. DiPPM coder VHDL source code.

```
1. ================================================================================
2. -- Project Name: DiPPM
3.  -- Name: DiPPMcoders. vhd
4. ================================================================================
5.  -- Description: DiPPM coder
6. ================================================================================
7.  --         libraries
8. ================================================================================
9.  Library IEEE;
10. use IEEE.STD_LOGIC_1164.all;
11.    entity DiPPMcoders is
12. port(
13. CLK   : in std_logic;
14. PCM : in std_logic;
15. enableIn : in std_logic;
16. startPlsIn : in std_logic;
17. DiPPM:out std_logic;
18. enableOut:out std_logic;
19. startPlsOut:out std_logic
20. );
21. end DiPPMcoders;
22.   architecture beh of DiPPMcoders is
23. Signal DiPPM_inner:std_logic;
24. Signal enable_inner:std_logic;
25. Signal startpls_inner:std_logic;
26. signal DiPPMss:std_logic;
27. signal DiPPMr:std_logic;
28. signal DiPPMrr:std_logic;
29. signal DiPPMrrr:std_logic;
30. Signal R:std_logic;
31. Signal S:std_logic;
32.
33. begin
34.
35.        dummy_process :process(CLK) is
36.     begin
37.         if rising_edge(CLK) then
38.             enable_inner   <= enableIn;
39.             startpls_inner <= startPlsIn;
40.         end if;
41.       end process;
42.
43.  process
44. begin
45.
46. wait until clk='0' and clk'event;
47. DiPPMss<=PCM;
48. end process;
49.
50.   S<= '1' when PCM='1' and DiPPMss='0' else
51. '0';
52.   DiPPMr<='1' when PCM='0'else
53. '0';
```

```
54.
55. process
56. begin
57. wait until clk='0' and clk'event;
58.   DiPPMrr<=DiPPMr;
59. end process;
60.
61. process
62. begin
63. wait until clk='1'and clk'event;
64. DiPPMrrr<=DiPPMrr;
65. end process;
66.
67.   R<='1' when DiPPMrrr='0' and DiPPMrr='1' else
68. '0';
69.   DiPPM_inner<= '1' when S='1' and R='0'else
70. '1' when S='0' and R='1'else
71. '0';
72.
73.     DiPPM <= DiPPM_inner;
74.     enableOut <= enable_inner;
75.     startPlsOut <= startpls_inner;
76.
77. end beh;
```

10.3.5. Channel model VHDL source code.

```
1. ================================================================
2. -- Project Name: Channel
3.  -- Name: channelmodel_top. vhd
4. ================================================================
5.  -- Description: Channel Model
6. ================================================================
7.  --         libraries
8. ================================================================
9.  library ieee;
10. use ieee.std_logic_1164.all;
11. use ieee.numeric_std.all;
12.
13. use work.channelmodel_top_pkg.all;
14.
15.================================================================
16. --         TOP instantiation
17.================================================================
18. entity channelmodel_top is
19.     port (
20.         CLK         : in  std_logic;                    -- system
   clock
21.         RESET       : in  std_logic;                    -- system
   reset
22.         DATAIN      : in  std_logic;                    -- enable
   signal
23.         ENABLE      : in  std_logic;                    -- sync signal
24.         STARTPLS    : in  std_logic;                    -- data in
25.         -- Data output
26.         ENABLE_OUT   : out std_logic;                    -- enable
   signal
27.         STARTPLS_OUT : out std_logic;                    -- sync
   signal
28.         DATAOUT      : out std_logic                     -- data out
29. );
30. end channelmodel_top;
31.
32.
33.================================================================
34. --         RTL Architecture
35.================================================================
36. architecture rtl of channelmodel_top is
37.
38. ----------------------------------------------------------------
39. -- Signals
40. ----------------------------------------------------------------
41.     signal enable_inner   : std_logic;
42.     signal startPls_inner : std_logic;
43.     signal data_inner     : std_logic;
44.
45. begin
46.
47.
48.     ------------------------------------------
```

```
49.     -- channel model process
50.     channel_model_p : process (CLK,RESET) is
51.     begin
52.         if RESET = '1' then
53.           enable_inner   <= ENABLE;
54.             startPls_inner <= STARTPLS;
55.             data_inner     <= DATAIN;
56.
57.         elsif RESET = '0' then
58.           enable_inner   <= '0';
59.           startPls_inner <= '0';
60.           data_inner     <= '0';
61.
62.         end if;
63.     end process;
64.
65.     -------------------------------------------------------------
66.     -- Output ports
67.     DATAOUT      <= data_inner;
68.     ENABLE_OUT   <= enable_inner;
69.     STARTPLS_OUT <= startPls_inner;
70.
71. end rtl;
```

10.3.6. DiPPM decoder VHDL source code.

```
1. Project Name: DiPPM
2. -- Name: DiPPMdecoder. vhd
3. ================================================================================
4.  -- Description: DiPPM decoder
5. ================================================================================
6.  --         libraries
7. ================================================================================
8.  library ieee;
9.  use ieee.std_logic_1164.all;
10. use ieee.std_logic_arith.all;
11. use ieee.std_logic_unsigned.all;
12.  entity DiPPMdecoder is
13. port(
14. CLK : in std_logic;
15. DiPPM: in std_logic;
16. enableIn : in std_logic;
17. startPlsIn : in std_logic;
18. PCM_out:out std_logic;
19. enableOut:out std_logic;
20. startPlsOut:out std_logic
21. );
22. end DiPPMdecoder;
23.   architecture beh of DiPPMdecoder is
24.
25. Signal enable_inner:std_logic;
26. Signal startpls_inner:std_logic;
27. signal R:std_logic;
28. signal S:std_logic;
29. Signal PCM_inner:std_logic;
30. Signal nclk:std_logic;
31.
32. begin
33.   nclk<= clk nor clk;
34.   process(clk)
35. begin
36. if rising_edge(clk) then
37.   enable_inner<= enableIn;
38.   startpls_inner <= startPlsIn;
39.   end if;
40.   end process;
41.
42. process
43. begin
44. wait until nclk='1' and nclk'event;
45. S<=DiPPM;
46. end process;
47.
48. R<='1' when DiPPM='1' and clk='0' else
49. '0';
50. process (S,R)
51. begin
52. if S='1' then
53. PCM_inner<='1';
```

```
54. elsif R='1' then
55. PCM_inner<='0';
56. end if;
57. end process;
58.
59.     enableOut   <= enable_inner;
60.     startPlsOut <= startpls_inner;
61.     PCM_out <= PCM_inner;
62. end beh;
```

10.3.7. Searial to parallel bridge VHDL source code.

```
1. ================================================================
2. -- Project Name: bridge decoder
3.  -- Name: bridgedecoder_top. vhd
4. ================================================================
5.  -- Description: bridge for decoder
6. ================================================================
7.  --         libraries
8. ================================================================
9.  library ieee;
10. use ieee.std_logic_1164.all;
11. use ieee.numeric_std.all;
12.
13. library work;
14.
15. use work.bridgedecoder_top_pkg.all;
16. use work.bridgedecoder_dpram_pkg.all;
17.
18.
19.================================================================
20. --         TOP instantiation
21.================================================================
22. entity bridgedecoder_top is
23.     port (
24.         CLK         : in  std_logic;                    -- system
   clock
25.         RESET       : in  std_logic;                    -- system
   reset
26.         enableIn    : in  std_logic;                    -- enable
   signal
27.         startPlsIn  : in  std_logic;                    -- sync signal
28.         dataIn      : in  std_logic;                    -- data in
29.         -- Data output
30.         enableOut   : out std_logic;                    -- enable
   signal
31.         startPlsOut : out std_logic;                    -- sync signal
32.         dataOut     : out std_logic_vector(4 downto 0)  -- data out
33. );
34. end bridgedecoder_top;
35.
36.
37.
38.================================================================
39. --         RTL Architecture
40.================================================================
41. architecture rtl of bridgedecoder_top is
42.
43.
44. ----------------------------------------------------------------
45. -- Signals
46. ----------------------------------------------------------------
47.     -----
48.     signal cnt_5clk : std_logic_vector(2 downto 0);
49.     -----
```

```
50.     signal writePointer: std_logic_vector(4 downto 0);
51.     signal writeEn     : std_logic;
52.     signal wrdataIn    : std_logic_vector(4 downto 0);
53.     -----
54.     signal go_rd_process : std_logic;
55.     -----
56.     signal rd_enable : std_logic;
57.     signal readPointer : std_logic_vector(4 downto 0);
58.     signal dpramRdData : std_logic_vector(4 downto 0);
59.     -----
60.     signal rd_enable_ff1 : std_logic;
61.     signal rd_enable_ff2 : std_logic;
62.     signal rd_enable_ff3 : std_logic;
63.     -----
64.     signal dataOut_inner : std_logic_vector(4 downto 0);
65.     signal startPlsOut_inner : std_logic;
66.     signal enableOut_inner : std_logic;
67.
68. begin
69.
70.
71.     ---------------------------------------------
72.     -- RAM memory instantiation
73.     u_bridgedecoder_dpram : bridgedecoder_dpram
74.     port map(
75.         w_clk   => CLK,
76.         w_en    => writeEn,
77.         w_addr  => writePointer,
78.         w_data  => wrdataIn,
79.         r_clk   => CLK,
80.         r_en    => rd_enable,
81.         r_addr  => readPointer,
82.         r_data  => dpramRdData
83.     );
84.
85.     ---------------------------------------------
86.     -- bridge process
87.     brdigedec_p : process (CLK,RESET) is
88.     begin
89.         if RESET = '0' then
90.             cnt_5clk      <= (others => '0');
91.             writePointer  <= (others => '0');
92.             writeEn       <= '0';
93.             wrdataIn      <= (others => '0');
94.             go_rd_process <= '0';
95.             rd_enable     <= '0';
96.             readPointer   <= (others => '0');
97.             rd_enable_ff1 <= '0';
98.             rd_enable_ff2 <= '0';
99.             rd_enable_ff3 <= '0';
100.              elsif rising_edge(CLK) then
101.                  -- counter 5 clk
102.                  if (enableIn ='1') then
103.                     if (startPlsIn ='1') then
104.                        cnt_5clk <= "001";
105.                     elsif(cnt_5clk >= "100") then
106.                        cnt_5clk <= (others => '0');
```

```
107.                     else
108.                        cnt_5clk <=
   std_logic_vector(unsigned(cnt_5clk) + 1);
109.                     end if;
110.                 end if;
111.                 -- writeEn
112.                 if (startPlsIn ='1') then
113.                     writeEn      <= '0';
114.                 elsif ((enableIn ='1') and (cnt_5clk = "100")) then
115.                     writeEn      <= '1';
116.                 else
117.                     writeEn  <= '0';
118.                 end if;
119.                 -- writePointer
120.                 if (startPlsIn ='1') then
121.                     writePointer <= "00000";
122.                 elsif (writeEn ='1') then
123.                     writePointer <=
   std_logic_vector(unsigned(writePointer) + 1);
124.                 end if;
125.                 -- writeDataIn
126.                 if (enableIn ='1') then
127.                     wrdataIn (4 downto 0) <= wrdataIn (3 downto 0) &
   dataIn;
128.                 end if;
129.                 -- go_rd_process
130.                 if ((writeEn ='1') and (writePointer="11110")) then
131.                     go_rd_process <= '1';
132.                 elsif ((rd_enable ='1') and (readPointer="11101"))
   then
133.                     go_rd_process <= '0';
134.                 end if;
135.                 -- rd_enable
136.                 rd_enable <= go_rd_process;
137.                 -- readPointer
138.                 if (rd_enable ='1') then
139.                     if (readPointer < "11110") then
140.                       readPointer <=
   std_logic_vector(unsigned(readPointer) + 1);
141.                     else
142.                         readPointer <= (others => '0');
143.                     end if;
144.                 else
145.                     readPointer <= (others => '0');
146.                 end if;
147.                 -- ffs
148.                 rd_enable_ff1 <= rd_enable;
149.                 rd_enable_ff2 <= rd_enable_ff1;
150.                 rd_enable_ff3 <= rd_enable_ff2;
151.                 -- startPlsOut_inner
152.                 if ((rd_enable_ff2='1') and (rd_enable_ff3='0'))
   then
153.                     startPlsOut_inner <= '1';
154.                 else
155.                     startPlsOut_inner <= '0';
156.                 end if;
157.                 -- enableOut_inner
```

```
158.                 enableOut_inner <= rd_enable_ff2;
159.                 -- dataOut_inner
160.                 if (rd_enable_ff2 ='1') then
161.                     dataOut_inner <= dpramRdData;
162.                 else
163.                     dataOut_inner <= "00000";
164.                 end if;
165.             end if;
166.         end process;
167.
168.
169.
170.         ------------------------------------------------------------
171.         -- Output ports
172.         dataOut     <= dataOut_inner;
173.         enableOut   <= enableOut_inner;
174.         startPlsOut <= startPlsOut_inner;
175.
176.     end rtl;
```

10.3.8. RS decoder VHDL source code.

```
1. ================================================================================
2. Project Name: RSIP
3.  -- Name: rsdecoder_top_31_23. vhd
4. ================================================================================
5.  -- Description: RS (31,23) decoder module
6. ================================================================================
7.  --         libraries
8. ================================================================================
9.  library ieee;
10.       use ieee.std_logic_1164.all;
11.       use ieee.numeric_std.all;
12.
13.       library work;
14.       use work.rsdecoder_31_23_top_pkg.all;
15.       use work.rsdecoder_31_23_pkg.all;
16.
17.
18.      ================================================================
19.       --         TOP instantiation
20.      ================================================================
21.       entity rsdecoder_31_23_top is
22.           port (
23.            CLK: in  std_logic;                          --system clock
24.           RESET: in  std_logic;                         --system reset
25.          enable: in  std_logic;              --rs decoder enable signal
26.       startPls: in  std_logic;                 --rs decoder sync signal
27.      erasureIn: in  std_logic;              -- rs decoder erasure input
28.         dataIn: in  std_logic_vector(4 downto 0); --rs encoder data in
29.                -- Data output
30.      outEnable: out  std_logic;    -- rs decoder data out valid signal
31.      outStartPls : out  std_logic;                                  -- rs
   decoder first decoded symbol trigger
32.                outDone     : out  std_logic;                       -- rs
   decoder last symbol decoded trigger
33.                errorNum    : out std_logic_vector(4 downto 0);    -- rs
   decoder number of errors corrected
34.                erasureNum  : out std_logic_vector(4 downto 0);    -- rs
   decoder number of erasure corrected
35.                fail        : out  std_logic;                      -- rs
   decoder decoding failure signal
36.                delayedData : out std_logic_vector(4 downto 0);    -- rs
   decoder delayed input data
37.                outData     : out std_logic_vector(4 downto 0)     -- rs
   encoder data out
38.       );
39.       end rsdecoder_31_23_top;
40.
41.
42.
43.      ================================================================
44.       --         RTL Architecture
45.      ================================================================
46.       architecture rtl of rsdecoder_31_23_top is
```

```
47.
48.
49.     ------------------------------------------------------------------
50.     -- Signals
51.     ------------------------------------------------------------------
52.         signal dataInCheck: std_logic_vector(4 downto 0);
53.
54.         signal syndrome_0: std_logic_vector(4 downto 0);
55.         signal syndrome_1: std_logic_vector(4 downto 0);
56.         signal syndrome_2: std_logic_vector(4 downto 0);
57.         signal syndrome_3: std_logic_vector(4 downto 0);
58.         signal syndrome_4: std_logic_vector(4 downto 0);
59.         signal syndrome_5: std_logic_vector(4 downto 0);
60.         signal syndrome_6: std_logic_vector(4 downto 0);
61.         signal syndrome_7: std_logic_vector(4 downto 0);
62.         signal doneSyndrome : std_logic;
63.
64.         signal epsilon_0: std_logic_vector(4 downto 0);
65.         signal epsilon_1: std_logic_vector(4 downto 0);
66.         signal epsilon_2: std_logic_vector(4 downto 0);
67.         signal epsilon_3: std_logic_vector(4 downto 0);
68.         signal epsilon_4: std_logic_vector(4 downto 0);
69.         signal epsilon_5: std_logic_vector(4 downto 0);
70.         signal epsilon_6: std_logic_vector(4 downto 0);
71.         signal epsilon_7: std_logic_vector(4 downto 0);
72.         signal epsilon_8: std_logic_vector(4 downto 0);
73.         signal degreeEpsilon: std_logic_vector(3 downto 0);
74.         signal failErasure : std_logic;
75.         signal doneErasure : std_logic;
76.
77.         signal polymulSyndrome_0: std_logic_vector(4 downto 0);
78.         signal polymulSyndrome_1: std_logic_vector(4 downto 0);
79.         signal polymulSyndrome_2: std_logic_vector(4 downto 0);
80.         signal polymulSyndrome_3: std_logic_vector(4 downto 0);
81.         signal polymulSyndrome_4: std_logic_vector(4 downto 0);
82.         signal polymulSyndrome_5: std_logic_vector(4 downto 0);
83.         signal polymulSyndrome_6: std_logic_vector(4 downto 0);
84.         signal polymulSyndrome_7: std_logic_vector(4 downto 0);
85.         signal donePolymul : std_logic;
86.
87.         signal lambda_0: std_logic_vector(4 downto 0);
88.         signal lambda_1: std_logic_vector(4 downto 0);
89.         signal lambda_2: std_logic_vector(4 downto 0);
90.         signal lambda_3: std_logic_vector(4 downto 0);
91.         signal lambda_4: std_logic_vector(4 downto 0);
92.         signal lambda_5: std_logic_vector(4 downto 0);
93.         signal lambda_6: std_logic_vector(4 downto 0);
94.         signal lambda_7: std_logic_vector(4 downto 0);
95.         signal lambda_8: std_logic_vector(4 downto 0);
96.         signal omega_0: std_logic_vector(4 downto 0);
97.         signal omega_1: std_logic_vector(4 downto 0);
98.         signal omega_2: std_logic_vector(4 downto 0);
99.         signal omega_3: std_logic_vector(4 downto 0);
100.        signal omega_4: std_logic_vector(4 downto 0);
101.        signal omega_5: std_logic_vector(4 downto 0);
102.        signal omega_6: std_logic_vector(4 downto 0);
103.        signal omega_7: std_logic_vector(4 downto 0);
```

```
104.          signal doneEuclide : std_logic;
105.          signal numShifted: std_logic_vector(3 downto 0);
106.          signal degreeEpsilonReg: std_logic_vector(3 downto 0);
107.
108.          signal epsilonReg_0: std_logic_vector(4 downto 0);
109.          signal epsilonReg_1: std_logic_vector(4 downto 0);
110.          signal epsilonReg_2: std_logic_vector(4 downto 0);
111.          signal epsilonReg_3: std_logic_vector(4 downto 0);
112.          signal epsilonReg_4: std_logic_vector(4 downto 0);
113.          signal epsilonReg_5: std_logic_vector(4 downto 0);
114.          signal epsilonReg_6: std_logic_vector(4 downto 0);
115.          signal epsilonReg_7: std_logic_vector(4 downto 0);
116.          signal epsilonReg_8: std_logic_vector(4 downto 0);
117.
118.          signal epsilonReg2_0: std_logic_vector(4 downto 0);
119.          signal epsilonReg2_1: std_logic_vector(4 downto 0);
120.          signal epsilonReg2_2: std_logic_vector(4 downto 0);
121.          signal epsilonReg2_3: std_logic_vector(4 downto 0);
122.          signal epsilonReg2_4: std_logic_vector(4 downto 0);
123.          signal epsilonReg2_5: std_logic_vector(4 downto 0);
124.          signal epsilonReg2_6: std_logic_vector(4 downto 0);
125.          signal epsilonReg2_7: std_logic_vector(4 downto 0);
126.          signal epsilonReg2_8: std_logic_vector(4 downto 0);
127.
128.
129.          signal epsilonReg3_0: std_logic_vector(4 downto 0);
130.          signal epsilonReg3_1: std_logic_vector(4 downto 0);
131.          signal epsilonReg3_2: std_logic_vector(4 downto 0);
132.          signal epsilonReg3_3: std_logic_vector(4 downto 0);
133.          signal epsilonReg3_4: std_logic_vector(4 downto 0);
134.          signal epsilonReg3_5: std_logic_vector(4 downto 0);
135.          signal epsilonReg3_6: std_logic_vector(4 downto 0);
136.          signal epsilonReg3_7: std_logic_vector(4 downto 0);
137.          signal epsilonReg3_8: std_logic_vector(4 downto 0);
138.
139.          signal omegaShiftedReg_0: std_logic_vector(4 downto 0);
140.          signal omegaShiftedReg_1: std_logic_vector(4 downto 0);
141.          signal omegaShiftedReg_2: std_logic_vector(4 downto 0);
142.          signal omegaShiftedReg_3: std_logic_vector(4 downto 0);
143.          signal omegaShiftedReg_4: std_logic_vector(4 downto 0);
144.          signal omegaShiftedReg_5: std_logic_vector(4 downto 0);
145.          signal omegaShiftedReg_6: std_logic_vector(4 downto 0);
146.          signal omegaShiftedReg_7: std_logic_vector(4 downto 0);
147.          signal omegaShiftedReg_8: std_logic_vector(4 downto 0);
148.
149.          signal degreeEpsilonReg2: std_logic_vector(3 downto 0);
150.          signal degreeEpsilonReg3: std_logic_vector(3 downto 0);
151.          signal degreeEpsilonReg4: std_logic_vector(3 downto 0);
152.          signal degreeEpsilonReg5: std_logic_vector(3 downto 0);
153.          signal doneShiftReg : std_logic;
154.          signal doneChien : std_logic;
155.          signal doneReg: std_logic_vector(2 downto 0);
156.          signal numErasureReg: std_logic_vector(3 downto 0);
157.          signal doneShift : std_logic;
158.          signal numShiftedReg: std_logic_vector(3 downto 0);
159.
160.          signal lambdaReg_0: std_logic_vector(4 downto 0);
```

```
161.          signal lambdaReg_1: std_logic_vector(4 downto 0);
162.          signal lambdaReg_2: std_logic_vector(4 downto 0);
163.          signal lambdaReg_3: std_logic_vector(4 downto 0);
164.          signal lambdaReg_4: std_logic_vector(4 downto 0);
165.          signal lambdaReg_5: std_logic_vector(4 downto 0);
166.          signal lambdaReg_6: std_logic_vector(4 downto 0);
167.          signal lambdaReg_7: std_logic_vector(4 downto 0);
168.          signal lambdaReg_8: std_logic_vector(4 downto 0);
169.
170.          signal omegaReg_0: std_logic_vector(4 downto 0);
171.          signal omegaReg_1: std_logic_vector(4 downto 0);
172.          signal omegaReg_2: std_logic_vector(4 downto 0);
173.          signal omegaReg_3: std_logic_vector(4 downto 0);
174.          signal omegaReg_4: std_logic_vector(4 downto 0);
175.          signal omegaReg_5: std_logic_vector(4 downto 0);
176.          signal omegaReg_6: std_logic_vector(4 downto 0);
177.          signal omegaReg_7: std_logic_vector(4 downto 0);
178.
179.          signal omegaShifted_0: std_logic_vector(4 downto 0);
180.          signal omegaShifted_1: std_logic_vector(4 downto 0);
181.          signal omegaShifted_2: std_logic_vector(4 downto 0);
182.          signal omegaShifted_3: std_logic_vector(4 downto 0);
183.          signal omegaShifted_4: std_logic_vector(4 downto 0);
184.          signal omegaShifted_5: std_logic_vector(4 downto 0);
185.          signal omegaShifted_6: std_logic_vector(4 downto 0);
186.          signal omegaShifted_7: std_logic_vector(4 downto 0);
187.
188.          signal degreeOmega: std_logic_vector(3 downto 0);
189.
190.          signal lambdaReg2_0: std_logic_vector(4 downto 0);
191.          signal lambdaReg2_1: std_logic_vector(4 downto 0);
192.          signal lambdaReg2_2: std_logic_vector(4 downto 0);
193.          signal lambdaReg2_3: std_logic_vector(4 downto 0);
194.          signal lambdaReg2_4: std_logic_vector(4 downto 0);
195.          signal lambdaReg2_5: std_logic_vector(4 downto 0);
196.          signal lambdaReg2_6: std_logic_vector(4 downto 0);
197.          signal lambdaReg2_7: std_logic_vector(4 downto 0);
198.
199.          signal degreeLambda: std_logic_vector(3 downto 0);
200.          signal degreeOmegaReg: std_logic_vector(3 downto 0);
201.          signal error: std_logic_vector(4  downto 0);
202.          signal degreeLambdaReg: std_logic_vector(3 downto 0);
203.          signal delayedErasureIn : std_logic;
204.
205.          signal delayOut: std_logic_vector(5 downto 0);
206.          signal delayIn: std_logic_vector(5 downto 0);
207.
208.          signal delayedDataIn: std_logic_vector(4 downto 0);
209.
210.          signal startReg: std_logic_vector(3 downto 0);
211.          signal OutputValidReg : std_logic;
212.          signal numErrorLambdaReg: std_logic_vector(3 downto 0);
213.          signal degreeErrorReg : std_logic;
214.          signal numErrorReg: std_logic_vector(3 downto 0);
215.          signal failErasureReg : std_logic;
216.          signal failErasureReg2 : std_logic;
217.          signal failErasureReg3 : std_logic;
```

```
218.          signal failErasureReg4 : std_logic;
219.          signal failErasureReg5 : std_logic;
220.          signal failReg : std_logic;
221.
222.          signal DataOutInner: std_logic_vector(4 downto 0);
223.          signal DelayedDataOutInner: std_logic_vector(4 downto 0);
224.
225.          signal enableFF : std_logic;
226.          signal startRegInner : std_logic;
227.          signal doneRegInner : std_logic;
228.
229.          signal failRegInner : std_logic;
230.          signal OutputValidRegInner : std_logic;
231.          signal numErrorChien: std_logic_vector(3 downto 0);
232.
233.          signal numErrorRegInner: std_logic_vector(4 downto 0);
234.          signal numErasureRegInner: std_logic_vector(4 downto 0);
235.
236.          signal temp_add: std_logic_vector(4 downto 0);
237.          signal temp_add1: std_logic_vector(4 downto 0);
238.          signal temp_add2: std_logic_vector(4 downto 0);
239.
240.      begin
241.
242.
243.          ---------------------------------------------
244.          -- dataInCheck (assign to 0 if Erasure)
245.          dataInCheck <= dataIn when (erasureIn = '0') else "00000";
246.
247.
248.          ---------------------------------------------
249.          -- syndrome_0,...,syndrome_7, doneSyndrome
250.          -- RS Syndrome calculation
251.          u_rsdecoder_31_23_syndrome : rsdecoder_31_23_syndrome
252.          port map (
253.            CLK        => CLK,
254.            RESET      => RESET,
255.            enable     => enable,
256.            sync       => startPls,
257.            dataIn     => dataInCheck,
258.            syndrome_0 => syndrome_0,
259.            syndrome_1 => syndrome_1,
260.            syndrome_2 => syndrome_2,
261.            syndrome_3 => syndrome_3,
262.            syndrome_4 => syndrome_4,
263.            syndrome_5 => syndrome_5,
264.            syndrome_6 => syndrome_6,
265.            syndrome_7 => syndrome_7,
266.            done       => doneSyndrome
267.          );
268.
269.
270.          ---------------------------------------------
271.          -- syndrome_0,...,syndrome_7, doneSyndrome
272.          -- RS Erasure calculation
273.          u_rsdecoder_31_23_erasure : rsdecoder_31_23_erasure
274.          port map (
```

```
275.           CLK       => CLK,
276.           RESET     => RESET,
277.           enable    => enable,
278.           sync      => startPls,
279.           erasureIn => erasureIn,
280.           epsilon_0 => epsilon_0,
281.           epsilon_1 => epsilon_1,
282.           epsilon_2 => epsilon_2,
283.           epsilon_3 => epsilon_3,
284.           epsilon_4 => epsilon_4,
285.           epsilon_5 => epsilon_5,
286.           epsilon_6 => epsilon_6,
287.           epsilon_7 => epsilon_7,
288.           epsilon_8 => epsilon_8,
289.           numErasure => degreeEpsilon,
290.           fail       => failErasure,
291.           done       => doneErasure
292.         );
293.
294.
295.          ---------------------------------------------
296.          -- polymulSyndrome_0,..., polymulSyndrome_7
297.          -- RS Polymul calculation
298.          u_rsdecoder_31_23_polymul : rsdecoder_31_23_polymul
299.          port map (
300.            CLK           =>  CLK,
301.            RESET         =>  RESET,
302.            enable        =>  enable,
303.            sync          =>  doneSyndrome,
304.            syndromeIn_0  =>  syndrome_0,
305.            syndromeIn_1  =>  syndrome_1,
306.            syndromeIn_2  =>  syndrome_2,
307.            syndromeIn_3  =>  syndrome_3,
308.            syndromeIn_4  =>  syndrome_4,
309.            syndromeIn_5  =>  syndrome_5,
310.            syndromeIn_6  =>  syndrome_6,
311.            syndromeIn_7  =>  syndrome_7,
312.            epsilon_0     =>  epsilon_0,
313.            epsilon_1     =>  epsilon_1,
314.            epsilon_2     =>  epsilon_2,
315.            epsilon_3     =>  epsilon_3,
316.            epsilon_4     =>  epsilon_4,
317.            epsilon_5     =>  epsilon_5,
318.            epsilon_6     =>  epsilon_6,
319.            epsilon_7     =>  epsilon_7,
320.            epsilon_8     =>  epsilon_8,
321.            syndromeOut_0 =>  polymulSyndrome_0,
322.            syndromeOut_1 =>  polymulSyndrome_1,
323.            syndromeOut_2 =>  polymulSyndrome_2,
324.            syndromeOut_3 =>  polymulSyndrome_3,
325.            syndromeOut_4 =>  polymulSyndrome_4,
326.            syndromeOut_5 =>  polymulSyndrome_5,
327.            syndromeOut_6 =>  polymulSyndrome_6,
328.            syndromeOut_7 =>  polymulSyndrome_7,
329.            done          =>  donePolymul
330.          );
331.
```

```
332.
333.          ---------------------------------------------
334.          -- polymulSyndrome_0,..., polymulSyndrome_7
335.          -- RS euclide
336.          u_rsdecoder_31_23_euclide : rsdecoder_31_23_euclide
337.          port map (
338.            CLK        => CLK,
339.            RESET      => RESET,
340.            enable      => enable,
341.            sync        => donePolymul,
342.            syndrome_0  => polymulSyndrome_0,
343.            syndrome_1  => polymulSyndrome_1,
344.            syndrome_2  => polymulSyndrome_2,
345.            syndrome_3  => polymulSyndrome_3,
346.            syndrome_4  => polymulSyndrome_4,
347.            syndrome_5  => polymulSyndrome_5,
348.            syndrome_6  => polymulSyndrome_6,
349.            syndrome_7  => polymulSyndrome_7,
350.            numErasure  => degreeEpsilonReg,
351.            lambda_0    => lambda_0,
352.            lambda_1    => lambda_1,
353.            lambda_2    => lambda_2,
354.            lambda_3    => lambda_3,
355.            lambda_4    => lambda_4,
356.            lambda_5    => lambda_5,
357.            lambda_6    => lambda_6,
358.            lambda_7    => lambda_7,
359.            omega_0     => omega_0,
360.            omega_1     => omega_1,
361.            omega_2     => omega_2,
362.            omega_3     => omega_3,
363.            omega_4     => omega_4,
364.            omega_5     => omega_5,
365.            omega_6     => omega_6,
366.            omega_7     => omega_7,
367.            numShifted  => numShifted,
368.            done        => doneEuclide
369.          );
370.
371.
372.          ---------------------------------------------
373.          -- epsilonReg_0, ..., epsilonReg_8
374.          rs_epsilon : process (CLK,RESET) is
375.          begin
376.              if RESET = '0' then
377.                  epsilonReg_0 <= (others => '0');
378.                  epsilonReg_1 <= (others => '0');
379.                  epsilonReg_2 <= (others => '0');
380.                  epsilonReg_3 <= (others => '0');
381.                  epsilonReg_4 <= (others => '0');
382.                  epsilonReg_5 <= (others => '0');
383.                  epsilonReg_6 <= (others => '0');
384.                  epsilonReg_7 <= (others => '0');
385.                  epsilonReg_8 <= (others => '0');
386.                  degreeEpsilonReg<= (others => '0');
387.              elsif rising_edge(CLK) then
388.                  if ((enable ='1') and (doneErasure ='1')) then
```

```
                epsilonReg_0 <= epsilon_0;
                epsilonReg_1 <= epsilon_1;
                epsilonReg_2 <= epsilon_2;
                epsilonReg_3 <= epsilon_3;
                epsilonReg_4 <= epsilon_4;
                epsilonReg_5 <= epsilon_5;
                epsilonReg_6 <= epsilon_6;
                epsilonReg_7 <= epsilon_7;
                epsilonReg_8 <= epsilon_8;
                degreeEpsilonReg<= degreeEpsilon;
                end if;
            end if;
        end process;


        ------------------------------------------
        -- epsilonReg2_0,..., epsilonReg2_8
        rs_epsilon2 : process (CLK,RESET) is
        begin
            if RESET = '0' then
                epsilonReg2_0 <= (others => '0');
                epsilonReg2_1 <= (others => '0');
                epsilonReg2_2 <= (others => '0');
                epsilonReg2_3 <= (others => '0');
                epsilonReg2_4 <= (others => '0');
                epsilonReg2_5 <= (others => '0');
                epsilonReg2_6 <= (others => '0');
                epsilonReg2_7 <= (others => '0');
                epsilonReg2_8 <= (others => '0');
                degreeEpsilonReg2<= (others => '0');
            elsif rising_edge(CLK) then
                if ((enable ='1') and (donePolymul ='1')) then
                epsilonReg2_0 <= epsilonReg_0;
                epsilonReg2_1 <= epsilonReg_1;
                epsilonReg2_2 <= epsilonReg_2;
                epsilonReg2_3 <= epsilonReg_3;
                epsilonReg2_4 <= epsilonReg_4;
                epsilonReg2_5 <= epsilonReg_5;
                epsilonReg2_6 <= epsilonReg_6;
                epsilonReg2_7 <= epsilonReg_7;
                epsilonReg2_8 <= epsilonReg_8;
                degreeEpsilonReg2<= degreeEpsilonReg;
                end if;
            end if;
        end process;


        ------------------------------------------
        -- omegaShiftedReg_0,..., omegaShiftedReg_8
        rs_omegashifted : process (CLK,RESET) is
        begin
            if RESET = '0' then
                epsilonReg3_0 <= (others => '0');
                epsilonReg3_1 <= (others => '0');
                epsilonReg3_2 <= (others => '0');
                epsilonReg3_3 <= (others => '0');
                epsilonReg3_4 <= (others => '0');
```

```
446.                epsilonReg3_5 <= (others => '0');
447.                epsilonReg3_6 <= (others => '0');
448.                epsilonReg3_7 <= (others => '0');
449.                epsilonReg3_8 <= (others => '0');
450.                degreeEpsilonReg3<= (others => '0');
451.                numShiftedReg<= (others => '0');
452.            elsif rising_edge(CLK) then
453.                if ((enable ='1') and (doneEuclide ='1')) then
454.                epsilonReg3_0 <= epsilonReg2_0;
455.                epsilonReg3_1 <= epsilonReg2_1;
456.                epsilonReg3_2 <= epsilonReg2_2;
457.                epsilonReg3_3 <= epsilonReg2_3;
458.                epsilonReg3_4 <= epsilonReg2_4;
459.                epsilonReg3_5 <= epsilonReg2_5;
460.                epsilonReg3_6 <= epsilonReg2_6;
461.                epsilonReg3_7 <= epsilonReg2_7;
462.                epsilonReg3_8 <= epsilonReg2_8;
463.                degreeEpsilonReg3<= degreeEpsilonReg2;
464.                numShiftedReg <= numShifted;
465.                lambdaReg_0 <= lambda_0;
466.                lambdaReg_1 <= lambda_1;
467.                lambdaReg_2 <= lambda_2;
468.                lambdaReg_3 <= lambda_3;
469.                lambdaReg_4 <= lambda_4;
470.                lambdaReg_5 <= lambda_5;
471.                lambdaReg_6 <= lambda_6;
472.                lambdaReg_7 <= lambda_7;
473.                lambdaReg_8 <= lambda_8;
474.                omegaReg_0 <= omega_0;
475.                omegaReg_1 <= omega_1;
476.                omegaReg_2 <= omega_2;
477.                omegaReg_3 <= omega_3;
478.                omegaReg_4 <= omega_4;
479.                omegaReg_5 <= omega_5;
480.                omegaReg_6 <= omega_6;
481.                omegaReg_7 <= omega_7;
482.                end if;
483.            end if;
484.        end process;
485.
486.
487.        -------------------------------------------
488.        --
489.        rs_epsilon4 : process (CLK,RESET) is
490.        begin
491.            if RESET = '0' then
492.                degreeEpsilonReg4<= (others => '0');
493.            elsif rising_edge(CLK) then
494.                if ((enable ='1') and (doneShiftReg ='1')) then
495.                degreeEpsilonReg4<= degreeEpsilonReg3;
496.                end if;
497.            end if;
498.        end process;
499.
500.
501.        -------------------------------------------
502.        --
```

```
503.          rs_epsilon5 : process (CLK,RESET) is
504.          begin
505.              if RESET = '0' then
506.                  degreeEpsilonReg5<= (others => '0');
507.                  numErasureReg<= (others => '0');
508.              elsif rising_edge(CLK) then
509.                  if ((enable ='1') and (doneChien ='1')) then
510.                      degreeEpsilonReg5<= degreeEpsilonReg4;
511.                  end if;
512.                  if ((enable ='1') and (doneReg(0)='1')) then
513.                      numErasureReg   <= degreeEpsilonReg5;
514.                  end if;
515.              end if;
516.          end process;
517.
518.
519.         -------------------------------------------
520.          --
521.          rs_doneShift : process (CLK,RESET) is
522.          begin
523.              if RESET = '0' then
524.                  doneShift<= '0';
525.              elsif rising_edge(CLK) then
526.                  if ((enable ='1')) then
527.                      doneShift<= doneEuclide;
528.                  end if;
529.              end if;
530.          end process;
531.
532.
533.          ---------------------------------------------
534.          --
535.          u_rsdecoder_31_23_shiftomega : rsdecoder_31_23_shiftomega
536.          port map (
537.            omega_0         => omegaReg_0,
538.            omega_1         => omegaReg_1,
539.            omega_2         => omegaReg_2,
540.            omega_3         => omegaReg_3,
541.            omega_4         => omegaReg_4,
542.            omega_5         => omegaReg_5,
543.            omega_6         => omegaReg_6,
544.            omega_7         => omegaReg_7,
545.            omegaShifted_0  => omegaShifted_0,
546.            omegaShifted_1  => omegaShifted_1,
547.            omegaShifted_2  => omegaShifted_2,
548.            omegaShifted_3  => omegaShifted_3,
549.            omegaShifted_4  => omegaShifted_4,
550.            omegaShifted_5  => omegaShifted_5,
551.            omegaShifted_6  => omegaShifted_6,
552.            omegaShifted_7  => omegaShifted_7,
553.            numShifted      => numShiftedReg
554.          );
555.          -- 1 clk ff for omegashifted and lambdareg
556.          rs_doneShift2 : process (CLK,RESET) is
557.          begin
558.              if RESET = '0' then
559.                  omegaShiftedReg_0 <= (others => '0');
```

```
560.                omegaShiftedReg_1 <= (others => '0');
561.                omegaShiftedReg_2 <= (others => '0');
562.                omegaShiftedReg_3 <= (others => '0');
563.                omegaShiftedReg_4 <= (others => '0');
564.                omegaShiftedReg_5 <= (others => '0');
565.                omegaShiftedReg_6 <= (others => '0');
566.                omegaShiftedReg_7 <= (others => '0');
567.                lambdaReg2_0 <= (others => '0');
568.                lambdaReg2_1 <= (others => '0');
569.                lambdaReg2_2 <= (others => '0');
570.                lambdaReg2_3 <= (others => '0');
571.                lambdaReg2_4 <= (others => '0');
572.                lambdaReg2_5 <= (others => '0');
573.                lambdaReg2_6 <= (others => '0');
574.                lambdaReg2_7 <= (others => '0');
575.            elsif rising_edge(CLK) then
576.                if ((enable ='1')) then
577.                omegaShiftedReg_0 <= omegaShifted_0;
578.                omegaShiftedReg_1 <= omegaShifted_1;
579.                omegaShiftedReg_2 <= omegaShifted_2;
580.                omegaShiftedReg_3 <= omegaShifted_3;
581.                omegaShiftedReg_4 <= omegaShifted_4;
582.                omegaShiftedReg_5 <= omegaShifted_5;
583.                omegaShiftedReg_6 <= omegaShifted_6;
584.                omegaShiftedReg_7 <= omegaShifted_7;
585.                lambdaReg2_0 <= lambdaReg_0;
586.                lambdaReg2_1 <= lambdaReg_1;
587.                lambdaReg2_2 <= lambdaReg_2;
588.                lambdaReg2_3 <= lambdaReg_3;
589.                lambdaReg2_4 <= lambdaReg_4;
590.                lambdaReg2_5 <= lambdaReg_5;
591.                lambdaReg2_6 <= lambdaReg_6;
592.                lambdaReg2_7 <= lambdaReg_7;
593.                end if;
594.            end if;
595.         end process;
596.
597.
598.         ---------------------------------------------
599.         -- omega degree
600.         rsdecoder_31_23_degree_1  : rsdecoder_31_23_degree
601.         port map (
602.           polynom_0   => omegaShiftedReg_0,
603.           polynom_1   => omegaShiftedReg_1,
604.           polynom_2   => omegaShiftedReg_2,
605.           polynom_3   => omegaShiftedReg_3,
606.           polynom_4   => omegaShiftedReg_4,
607.           polynom_5   => omegaShiftedReg_5,
608.           polynom_6   => omegaShiftedReg_6,
609.           polynom_7   => omegaShiftedReg_7,
610.           degree      => degreeOmega
611.         );
612.         -- lambda degree
613.        rsdecoder_31_23_degree_2  : rsdecoder_31_23_degree
614.        port map (
615.           polynom_0   => lambdaReg2_0,
616.           polynom_1   => lambdaReg2_1,
```

```
617.           polynom_2   => lambdaReg2_2,
618.           polynom_3   => lambdaReg2_3,
619.           polynom_4   => lambdaReg2_4,
620.           polynom_5   => lambdaReg2_5,
621.           polynom_6   => lambdaReg2_6,
622.           polynom_7   => lambdaReg2_7,
623.           degree      => degreeLambda
624.         );
625.
626.
627.          ---------------------------------------------
628.          -- degree reg
629.          rs_degreeOmegaLambda : process (CLK,RESET) is
630.          begin
631.              if RESET = '0' then
632.                  degreeOmegaReg  <= (others => '0');
633.                  degreeLambdaReg <= (others => '0');
634.              elsif rising_edge(CLK) then
635.                  if ((enable ='1') and (doneShiftReg ='1')) then
636.                      degreeOmegaReg  <= degreeOmega;
637.                      degreeLambdaReg <= degreeLambda;
638.                  end if;
639.              end if;
640.          end process;
641.
642.
643.          ---------------------------------------------
644.          -- doneShiftReg
645.          rs_doneShiftReg : process (CLK,RESET) is
646.          begin
647.              if RESET = '0' then
648.                  doneShiftReg  <= '0';
649.              elsif rising_edge(CLK) then
650.                  if ((enable ='1')) then
651.                      doneShiftReg  <= doneShift;
652.                  end if;
653.              end if;
654.          end process;
655.
656.
657.          ---------------------------------------------
658.          -- RSChien
659.          u_rsdecoder_31_23_chien : rsdecoder_31_23_chien
660.          port map(
661.            CLK          => CLK,
662.            RESET        => RESET,
663.            enable       => enable,
664.            sync         => doneShiftReg,
665.            erasureIn    => delayedErasureIn,
666.            lambdaIn_0   => lambdaReg2_0,
667.            lambdaIn_1   => lambdaReg2_1,
668.            lambdaIn_2   => lambdaReg2_2,
669.            lambdaIn_3   => lambdaReg2_3,
670.            lambdaIn_4   => lambdaReg2_4,
671.            lambdaIn_5   => lambdaReg2_5,
672.            lambdaIn_6   => lambdaReg2_6,
673.            lambdaIn_7   => lambdaReg2_7,
```

```
674.           omegaIn_0    => omegaShiftedReg_0,
675.           omegaIn_1    => omegaShiftedReg_1,
676.           omegaIn_2    => omegaShiftedReg_2,
677.           omegaIn_3    => omegaShiftedReg_3,
678.           omegaIn_4    => omegaShiftedReg_4,
679.           omegaIn_5    => omegaShiftedReg_5,
680.           omegaIn_6    => omegaShiftedReg_6,
681.           omegaIn_7    => omegaShiftedReg_7,
682.           epsilonIn_0  => epsilonReg3_0,
683.           epsilonIn_1  => epsilonReg3_1,
684.           epsilonIn_2  => epsilonReg3_2,
685.           epsilonIn_3  => epsilonReg3_3,
686.           epsilonIn_4  => epsilonReg3_4,
687.           epsilonIn_5  => epsilonReg3_5,
688.           epsilonIn_6  => epsilonReg3_6,
689.           epsilonIn_7  => epsilonReg3_7,
690.           epsilonIn_8  => epsilonReg3_8,
691.           errorOut     => error,
692.           numError     => numErrorChien,
693.           done         => doneChien
694.         );
695.
696.
697.          ---------------------------------------------
698.          -- Rs Decode Delay
699.          u_rsdecoder_31_23_delay : rsdecoder_31_23_delay
700.          port map(
701.              CLK     => CLK,
702.              RESET   => RESET,
703.              enable  => enable,
704.              dataIn  => delayIn,
705.              dataOut => delayOut
706.          );
707.
708.
709.          ---------------------------------------------
710.          -- delayIn, delayedErasureIn, delayedDataIn
711.          delayIn          <= erasureIn & dataInCheck;
712.          delayedErasureIn <= delayOut(5);
713.          delayedDataIn    <= delayOut(4 downto 0);
714.
715.
716.          ---------------------------------------------
717.          -- OutputValidReg
718.          rs_OutputValidReg : process (CLK,RESET) is
719.          begin
720.              if RESET = '0' then
721.                  OutputValidReg  <= '0';
722.              elsif rising_edge(CLK) then
723.                  if ((enable ='1')) then
724.                      if ((startReg(1) ='1')) then
725.                          OutputValidReg  <= '1';
726.                      elsif (doneReg(0) ='1') then
727.                          OutputValidReg  <= '0';
728.                      end if;
729.                  end if;
730.              end if;
```

```
731.          end process;
732.
733.
734.          -------------------------------------------
735.          -- startReg, doneReg
736.          rs_startReg : process (CLK,RESET) is
737.          begin
738.              if RESET = '0' then
739.                  startReg  <= (others => '0');
740.                  doneReg   <= (others => '0');
741.              elsif rising_edge(CLK) then
742.                  if ((enable ='1')) then
743.                      startReg  <= doneShiftReg & startReg(3 downto
   1);
744.                      doneReg   <= doneChien    & doneReg(2 downto 1);
745.                  end if;
746.              end if;
747.          end process;
748.
749.
750.          -------------------------------------------
751.          -- numErrorLambdaReg
752.          rs_numErrorLambdaReg : process (CLK,RESET) is
753.          begin
754.              if RESET = '0' then
755.                  numErrorLambdaReg   <= (others => '0');
756.              elsif rising_edge(CLK) then
757.                  if ((enable ='1') and (startReg(1)='1')) then
758.                      numErrorLambdaReg  <= degreeLambdaReg;
759.                  end if;
760.              end if;
761.          end process;
762.
763.
764.          -------------------------------------------
765.          -- temp_add
766.          temp_add1 <= ('0' & degreeLambdaReg);
767.          temp_add2 <= ('0' & degreeEpsilonReg4);
768.          temp_add <= std_logic_vector(unsigned(temp_add1) +
   unsigned(temp_add2));
769.
770.
771.          -------------------------------------------
772.          -- degreeErrorReg
773.          rs_degreeErrorReg : process (CLK,RESET) is
774.          begin
775.              if RESET = '0' then
776.                  degreeErrorReg   <= '0';
777.              elsif rising_edge(CLK) then
778.                  if ((enable ='1') and (startReg(1)='1')) then
779.                      if ('0' & degreeOmegaReg) <= (temp_add) then
780.                          degreeErrorReg   <= '0';
781.                      else
782.                          degreeErrorReg   <= '1';
783.                      end if;
784.                  end if;
785.              end if;
```

```
786.          end process;
787.
788.
789.          -------------------------------------------
790.          -- numErrorReg
791.          rs_numErrorReg : process (CLK,RESET) is
792.          begin
793.              if RESET = '0' then
794.                  numErrorReg   <= (others => '0');
795.              elsif rising_edge(CLK) then
796.                  if ((enable ='1') and (doneReg(0)='1')) then
797.                      numErrorReg  <= numErrorChien;
798.                  end if;
799.              end if;
800.          end process;
801.
802.
803.          -------------------------------------------
804.          -- failErasureReg
805.          rs_failErasureReg : process (CLK,RESET) is
806.          begin
807.              if RESET = '0' then
808.                  failErasureReg  <= '0';
809.                  failErasureReg2  <= '0';
810.                  failErasureReg3  <= '0';
811.                  failErasureReg4  <= '0';
812.                  failErasureReg5  <= '0';
813.              elsif rising_edge(CLK) then
814.                  if ((enable ='1') and (doneErasure='1')) then
815.                      failErasureReg  <= failErasure;
816.                  end if;
817.                  if ((enable ='1') and (donePolymul='1')) then
818.                      failErasureReg2  <= failErasureReg;
819.                  end if;
820.                  if ((enable ='1') and (doneEuclide='1')) then
821.                      failErasureReg3  <= failErasureReg2;
822.                  end if;
823.                  if ((enable ='1') and (doneShiftReg='1')) then
824.                      failErasureReg4  <= failErasureReg3;
825.                  end if;
826.                  if ((enable ='1') and (startReg(1)='1')) then
827.                      failErasureReg5  <= failErasureReg4;
828.                  end if;
829.              end if;
830.          end process;
831.
832.
833.          -------------------------------------------
834.          -- failReg
835.          rs_failReg : process (CLK,RESET) is
836.          begin
837.              if RESET = '0' then
838.                  failReg  <= '0';
839.              elsif rising_edge(CLK) then
840.                  if ((enable ='1') and (doneReg(0)='1')) then
841.                      if ((numErrorLambdaReg = numErrorChien) and
   (degreeErrorReg='0') and (failErasureReg5='0')) then
```

```
842.                         failReg  <= '0';
843.                      else
844.                         failReg  <= '1';
845.                      end if;
846.                 end if;
847.             end if;
848.         end process;
849.
850.
851.         -------------------------------------------
852.         -- DataOutInner
853.         rs_DataOutInner : process (CLK,RESET) is
854.         begin
855.             if  RESET = '0' then
856.                 DataOutInner        <= (others => '0');
857.                 DelayedDataOutInner <= (others => '0');
858.                 enableFF            <= '0';
859.                 startRegInner       <= '0';
860.                 doneRegInner        <= '0';
861.                 numErrorRegInner    <= (others => '0');
862.                 numErasureRegInner  <= (others => '0');
863.                 failRegInner        <= '0';
864.             elsif rising_edge(CLK) then
865.                 DataOutInner        <= delayedDataIn xor error;
866.                 DelayedDataOutInner <= delayedDataIn;
867.                 enableFF            <= enable;
868.                 startRegInner       <= startReg(0);
869.                 doneRegInner        <= doneReg(0);
870.                 numErrorRegInner    <= '0' & numErrorReg(3 downto
    0);
871.                 numErasureRegInner  <= '0' & numErasureReg(3 downto
    0);
872.                 failRegInner        <= failReg;
873.             end if;
874.         end process;
875.
876.
877.         -------------------------------------------
878.         -- OutputValidRegInner
879.         rs_OutputValidRegInner: process (CLK,RESET) is
880.         begin
881.             if RESET = '0' then
882.                 OutputValidRegInner  <= '0';
883.             elsif rising_edge(CLK) then
884.                 if ((enableFF ='1')) then
885.                     OutputValidRegInner  <= OutputValidReg;
886.                 else
887.                     OutputValidRegInner  <= '0';
888.                 end if;
889.             end if;
890.         end process;
891.
892.
893.         ------------------------------------------------------------
    -----
894.         -- Output ports
895.         outEnable   <= OutputValidRegInner;
```

```
896.          outStartPls <= startRegInner;
897.          outDone     <= doneRegInner;
898.          outData     <= DataOutInner;
899.          errorNum    <= numErrorRegInner;
900.          erasureNum  <= numErasureRegInner;
901.          delayedData <= DelayedDataOutInner;
902.          fail        <= failRegInner;
903.
904.      end rtl;
```

10.4. Appendix 4

10.4.1. Erasure only test bench VHDL source code.

```
1. Erasure only test bench VHDL source code.
2. ================================================================================
3.  -- Project Name         : RSIP
4.  -- Name                 : simReedSolomon.vhd
5.  -- Actual Version       : v0.1
6. ================================================================================
7.  -- Description          : erasue only test environment
8. ================================================================================
9.  --         libraries
10.      ================================================================
11.       library ieee;
12.       use ieee.std_logic_1164.all;
13.       use ieee.numeric_std.all;
14.       use ieee.std_logic_textio.all;
15.       library std;
16.       use std.textio.all;
17.       library work;
18.       use work.rscoder_31_23_top_pkg.all;
19.       use work.rsdecoder_31_23_top_pkg.all;
20.      ================================================================
21.       --         TOP instantiation
22.      ================================================================
23.       entity simReedSolomon is
24.       end simReedSolomon;
25.
26.      ================================================================
27.       --         RTL Architecture
28.      ================================================================
29.       architecture TB of simReedSolomon is
30.
31.       constant CLK_PER :   time := 10 ns; -- 100Mhz
32.
33.       ----------------------------------------------------------------
34.       -- Signals
35.       ----------------------------------------------------------------
36.       -- decoder --
37.           signal CLK        : std_logic;
38.           signal RESET     : std_logic;
39.           signal rsdecEnable     : std_logic;
40.           signal rsdecSync   : std_logic;
41.           signal rsdecErasureIn   : std_logic;
42.           signal rsdecDataIn   : std_logic_vector(4 downto 0);
43.           signal rsdecOutStartPls   : std_logic;
44.           signal rsdecOutDone   : std_logic;
45.           signal rsdecOutData   : std_logic_vector(4 downto 0);
46.           signal rsdecErrorNum   : std_logic_vector(4 downto 0);
47.           signal rsdecErasureNum   : std_logic_vector(4 downto 0);
48.           signal rsdecFail   : std_logic;
49.           signal rsdecOutEnable   : std_logic;
50.           signal rsdecDelayedData   : std_logic_vector(4 downto 0);
```

```
51.
52.           signal rsencEnable   : std_logic;
53.           signal rsencStartPls   : std_logic;
54.           signal rsencDataIn   : std_logic_vector(4 downto 0);
55.           signal rsencDataOut   : std_logic_vector(4 downto 0);
56.
57.           signal rsdecOutEnableFF   : std_logic;
58.           signal rsdecOutDataFF   : std_logic_vector(4 downto 0);
59.           signal rsdecErasureNumFF   : std_logic_vector(4 downto 0);
60.           signal rsdecErrorNumFF   : std_logic_vector(4 downto 0);
61.           signal rsdecFailFF   : std_logic;
62.
63.
64.           signal simStart   : std_logic;
65.           signal simStart_ff1   : std_logic;
66.           signal simStart_ff2   : std_logic;
67.           signal simStart_ff3   : std_logic;
68.           signal rd_once_decin   : std_logic;
69.           signal rd_once_decout   : std_logic;
70.           signal rsdecOutEnable_ff1   : std_logic;
71.
72.           signal rsDecDataFlag   : std_logic;
73.           signal rsDecNGDataFlag   : std_logic;
74.           signal rsDecErasureFlag   : std_logic;
75.           signal rsDecNGErasureFlag   : std_logic;
76.           signal rsDecErrorFlag   : std_logic;
77.           signal rsDecNGErrorFlag   : std_logic;
78.           signal rsDecFailPinFlag   : std_logic;
79.           signal rsDecNGFailPinFlag   : std_logic;
80.
81.           signal rsdec0_sig : std_logic_vector(23 downto 0);
82.
83.           signal rsdecExpData_sig    : std_logic_vector(7 downto 0);
84.           signal rsdecExpNumErasure  : std_logic_vector(4 downto 0);
85.           signal rsdecExpNumError    : std_logic_vector(4 downto 0);
86.           signal rsdecExpFailFlag    : std_logic;
87.           signal rsdecExpData        : std_logic_vector(4 downto 0);
88.           signal rsdecExpDelayedData : std_logic_vector(4 downto 0);
89.
90.           signal rsdecOutData_ff1  : std_logic_vector(4 downto 0);
91.           signal rsdecErrorNum_ff1    : std_logic_vector(4 downto 0);
92.           signal rsdecFail_ff1    : std_logic;
93.           signal rsdecErasureNum_ff1        : std_logic_vector(4
   downto 0);
94.
95.           signal data_count  : std_logic_vector(5 downto 0);
96.       -- coder --
97.           signal rd_once_in   : std_logic;
98.           signal rd_once_out   : std_logic;
99.           signal rsEncPassFailFlag : std_logic;
100.          signal rsEncFailFlag : std_logic;
101.          signal rsenc0_sig     : std_logic_vector(15 downto 0);
102.          signal rsencEnable_ff1   : std_logic;
103.          signal rsencEnable_ff2   : std_logic;
104.          signal rsencStartPls_ff1   : std_logic;
105.          signal rsencStartPls_ff2   : std_logic;
106.          signal rsencExpData_sig     : std_logic_vector(4 downto 0);
```

```
107.      begin
108.
109.          -- RS  Decoder Top module Instantiation
110.          u_rsdecoder_31_23_top : rsdecoder_31_23_top
111.          port map(
112.            CLK          => CLK,               -- system clock
113.            RESET        => RESET,             -- system reset
114.            -- IN
115.            enable       => rsdecEnable,       -- RSdec enable in
116.            startPls     => rsdecSync,         -- RSdec sync signal
117.            erasureIn    => rsdecErasureIn,    -- RSdec erasure in
118.            dataIn       => rsdecDataIn,       -- RSdec data in
119.            -- OUT
120.            outEnable    => rsdecOutEnable,    -- RSdec enable out
121.            outStartPls  => rsdecOutStartPls,  -- RSdec start pulse
   out
122.            outDone      => rsdecOutDone,      -- RSdec done out
123.            errorNum     => rsdecErrorNum,     -- RSdec error number
124.            erasureNum   => rsdecErasureNum,   -- RSdec Erasure number
125.            fail         => rsdecFail,         -- RSdec Pass/Fail flag
126.            delayedData  => rsdecDelayedData,  -- RSdec delayed data
127.            outData      => rsdecOutData       -- Rsdec data out
128.          );
129.
130.          -- RS Encoder Top module Instantiation
131.          u_rscoder_31_23_top: rscoder_31_23_top
132.          port map(
133.            CLK      => CLK,           -- system clock
134.            RESET    => RESET,         -- system reset
135.            enable   => rsencEnable,   -- RSenc enable signal
136.            startPls => rsencStartPls, -- RSenc sync signal
137.            dataIn   => rsencDataIn,   -- RSenc data in
138.            rsdataOut  => rsencDataOut   -- RSenc data out
139.          );
140.
141.          -- Generate clock
142.          CLK_p:process
143.          begin
144.              CLK <= '0';
145.              wait for (CLK_PER/2);
146.              CLK <= '1';
147.              wait for (CLK_PER/2);
148.          end process;
149.
150.          rs_encsim : process (CLK,RESET) is
151.          variable file_status_in : file_open_status;
152.          variable file_status_out: file_open_status;
153.          file     mem_file_in        : TEXT;
154.          file     mem_file_out       : TEXT;
155.          variable mem_line_in     : line;
156.          variable mem_line_out    : line;
157.          variable rsenc0     : std_logic_vector(15 downto 0);
158.          variable rsencExpData  : std_logic_vector(7 downto 0);
159.          ---- DECODER
160.          variable file_status_decin : file_open_status;
161.          variable file_status_decout: file_open_status;
162.          file     mem_file_decin        : TEXT;
```

```
163.          file     mem_file_decout       : TEXT;
164.          variable mem_line_decin     : line;
165.          variable mem_line_decout    : line;
166.          variable rsdec0     : std_logic_vector(23 downto 0);
167.          variable rsdecExp  : std_logic_vector(7 downto 0);
168.
169.          begin
170.              if RESET = '0' then
171.                  rsencStartPls <= '0';
172.                  rsencEnable   <= '0';
173.                  rsencDataIn   <= (others => '0');
174.                  simStart_ff1  <= '0';
175.                  simStart_ff2  <= '0';
176.                  simStart_ff3  <= '0';
177.                  rd_once_in    <= '0';
178.                  rd_once_out   <= '0';
179.                  rsEncPassFailFlag <= '0';
180.                  rsEncFailFlag   <= '0';
181.                  -----------------------
182.                  rsdecSync          <= '0';
183.                  rsdecEnable        <= '0';
184.                  rsdecDataIn        <= (others => '0');
185.                  rd_once_decin      <= '0';
186.                  rd_once_decout     <= '0';
187.                  rsdecOutEnable_ff1 <= '0';
188.                  rsDecDataFlag      <= '0';
189.                  rsDecNGDataFlag    <= '0';
190.                  rsDecErasureFlag   <= '0';
191.                  rsDecNGErasureFlag <= '0';
192.                  rsDecErrorFlag     <= '0';
193.                  rsDecNGErrorFlag   <= '0';
194.                  rsDecFailPinFlag   <= '0';
195.                  rsDecNGFailPinFlag <= '0';
196.
197.                  rsdecExpData_sig        <= (others => '0');
198.                  rsdecExpNumErasure      <= (others => '0');
199.                  rsdecExpNumError        <= (others => '0');
200.                  rsdecExpFailFlag        <= '0';
201.                  rsdecExpData            <= (others => '0');
202.                  rsdecExpDelayedData     <= (others => '0');
203.
204.                  rsdecOutData_ff1      <= (others => '0');
205.                  rsdecErrorNum_ff1     <= (others => '0');
206.                  rsdecFail_ff1         <= '0';
207.                  rsdecErasureNum_ff1   <= (others => '0');
208.
209.                  rsencStartPls_ff1 <= '0';
210.                  rsencStartPls_ff2 <= '0';
211.                  data_count <=  (others => '0');
212.              elsif rising_edge(CLK) then
213.                  ---------------
214.                  simStart_ff1 <= simStart;
215.                  simStart_ff2 <= simStart_ff1;
216.                  simStart_ff3 <= simStart_ff2;
217.      ---------------------------------
218.                  --  ENCODER INPUT  ---
219.      ---------------------------------
```

```
220.                 if ((simStart_ff1 ='0') and (simStart ='1') and
   (rd_once_in='0')) then
221.                     file_open(file_status_in, mem_file_in,
   "RsEncIn.hex", READ_MODE);
222.                     rd_once_in   <= '1';
223.                 end if;
224.                 if ((simStart ='1')) then
225.                     readline(mem_file_in, mem_line_in);
226.                     hread(mem_line_in, rsenc0);
227.                 end if;
228.                 if (simStart ='1') then
229.                     rsenc0_sig    <= rsenc0;
230.                     rsencStartPls <= rsenc0(12);
231.                     rsencEnable   <= rsenc0(8);
232.                     rsencDataIn   <= rsenc0(4 downto 0);
233.                 end if;
234.     ----------------------------------
235.                 --  ENCODER OUTPUT  ---
236.     ----------------------------------
237.                 rsencEnable_ff1   <= rsencEnable;
238.                 rsencEnable_ff2   <= rsencEnable_ff1;
239.                 rsencStartPls_ff1 <= rsencStartPls;
240.                 rsencStartPls_ff2 <= rsencStartPls_ff1;
241.                 if ((rsencEnable_ff1 ='0') and (rsencEnable ='1')
   and (rd_once_out='0')) then
242.                     file_open(file_status_out, mem_file_out,
   "RsEncOut.hex", READ_MODE);
243.                     rd_once_out   <= '1';
244.                 end if;
245.                 if ((simStart_ff2 ='1')) then
246.                     readline(mem_file_out, mem_line_out);
247.                     hread(mem_line_out, rsencExpData);
248.                     rsencExpData_sig <= rsencExpData(4 downto 0);
249.                 end if;
250.                 if ((simStart_ff3 ='1')) then
251.                     if (rsencDataOut = rsencExpData_sig) then
252.                         rsEncPassFailFlag <= '0';
253.                     else
254.                         rsEncPassFailFlag <= '1';
255.                         rsEncFailFlag   <= '1';
256.                     end if;
257.                 end if;
258.     ----------------------------------
259.                 --  DECODER INPUT  ---
260.     ----------------------------------
261.                 if ((simStart ='1')) then
262.                     rsdecSync      <= rsencStartPls_ff2;
263.                     rsdecEnable    <= rsencEnable_ff2;
264.                     if ((data_count="000010") or
   (data_count="000100") or (data_count="000110") or (data_count="001010")
   or
265.                         (data_count="001100") or
   (data_count="001110") or (data_count="010010") or
   (data_count="010011")) then
266.                         rsdecErasureIn <= '1';
267.                         rsdecDataIn    <= (others => '0');
268.                     else
```

```
269.                         rsdecErasureIn <= '0';
270.                         rsdecDataIn    <= rsencDataOut;
271.                     end if;
272.                 end if;
273.                 if ((rsencEnable_ff2='1')) then
274.                     if (rsencStartPls_ff2='1') then
275.                         data_count <=  (others => '0');
276.                     else
277.                        data_count <=
   std_logic_vector(unsigned(data_count) + 1);
278.                     end if;
279.                 end if;
280.     ---------------------------------
281.                 --  DECODER OUTPUT  ---
282.     ---------------------------------
283.                 rsdecOutData_ff1      <= rsdecOutData;
284.                 rsdecErrorNum_ff1     <= rsdecErrorNum;
285.                 rsdecFail_ff1         <= rsdecFail;
286.                 rsdecErasureNum_ff1   <= rsdecErasureNum;
287.                 rsdecOutEnable_ff1 <= rsdecOutEnable;
288.                 if ((rsdecOutEnable_ff1 ='0') and (rsdecOutEnable
   ='1') and (rd_once_decout='0')) then
289.                     file_open(file_status_decout, mem_file_decout,
   "RsDecOut.hex", READ_MODE);
290.                     rd_once_decout   <= '1';
291.                 end if;
292.                 if ((rsdecOutEnable ='1')) then
293.                     readline(mem_file_decout, mem_line_decout);
294.                     hread(mem_line_decout, rsdecExp);
295.                 end if;
296.                 if ((rsdecOutEnable ='1')) then
297.                     rsdecExpData        <= rsdecExp(4 downto 0);
298.                 end if;
299.                 ---- Data Pin ----
300.                 if (rsdecOutEnable_ff1 = '1') then
301.                     if (rsdecOutData_ff1 = rsdecExpData) then
302.                         rsDecDataFlag <= '0';
303.                     else
304.                         rsDecDataFlag   <= '1';
305.                         rsDecNGDataFlag <= '1';
306.                     end if;
307.                 else
308.                     rsDecDataFlag <= '0';
309.                 end if;
310.             end if;
311.         end process;
312.     -------------------------------------------
313.     -- Generate reset and tb enable stimulus
314.         stimulus_p:process
315.         begin
316.            simStart  <= '0';
317.            RESET     <= '1';
318.            wait for 20 ns;
319.            RESET           <= '0';
320.            wait for 20 ns;
321.            RESET           <= '1';
322.            wait for 200 ns;
```

```
323.             simStart            <= '1';
324.             wait for 1000 ms;
325.             assert false report "End of simulation !" severity
   failure;
326.             wait;
327.          end process;
328.
329.      end TB;
```

10.4.2. Error only test bench VHDL source code.

```
1. ================================================================================
2.  -- Project Name         : RSIP
3.  -- Name                 : simReedSolomon.vhd
4.  -- Actual Version       : v0.1
5. ================================================================================
6.  -- Description          : error only test environment
7. ================================================================================
8.  --         libraries
9. ================================================================================
10.      library ieee;
11.       use ieee.std_logic_1164.all;
12.       use ieee.numeric_std.all;
13.       use ieee.std_logic_textio.all;
14.       library std;
15.       use std.textio.all;
16.       library work;
17.       use work.rscoder_31_23_top_pkg.all;
18.       use work.rsdecoder_31_23_top_pkg.all;
19.      ================================================================
20.       --         TOP instantiation
21.      ================================================================
22.       entity simReedSolomon is
23.       end simReedSolomon;
24.      ================================================================
25.      --         RTL Architecture
26.      ================================================================
27.      ============architecture TB of simReedSolomon is
28.      constant CLK_PER :   time := 10 ns; -- 100Mhz
29.       ----------------------------------------------------------------
30.      -------------- Signals
31.       ----------------------------------------------------------------
32.      -------------- decoder --
33.           signal CLK        : std_logic;
34.           signal RESET     : std_logic;
35.           signal rsdecEnable     : std_logic;
36.           signal rsdecSync   : std_logic;
37.           signal rsdecErasureIn   : std_logic;
38.           signal rsdecDataIn   : std_logic_vector(4 downto 0);
39.           signal rsdecOutStartPls   : std_logic;
40.           signal rsdecOutDone   : std_logic;
41.           signal rsdecOutData   : std_logic_vector(4 downto 0);
42.           signal rsdecErrorNum   : std_logic_vector(4 downto 0);
43.           signal rsdecErasureNum   : std_logic_vector(4 downto 0);
44.           signal rsdecFail   : std_logic;
45.           signal rsdecOutEnable   : std_logic;
46.           signal rsdecDelayedData   : std_logic_vector(4 downto 0);
47.
48.           signal rsencEnable   : std_logic;
49.           signal rsencStartPls   : std_logic;
50.           signal rsencDataIn   : std_logic_vector(4 downto 0);
51.           signal rsencDataOut   : std_logic_vector(4 downto 0);
52.
53.           signal rsdecOutEnableFF   : std_logic;
```

```
54.          signal rsdecOutDataFF   : std_logic_vector(4 downto 0);
55.          signal rsdecErasureNumFF   : std_logic_vector(4 downto 0);
56.          signal rsdecErrorNumFF   : std_logic_vector(4 downto 0);
57.          signal rsdecFailFF   : std_logic;
58.
59.          signal simStart   : std_logic;
60.          signal simStart_ff1   : std_logic;
61.          signal simStart_ff2   : std_logic;
62.          signal simStart_ff3   : std_logic;
63.          signal rd_once_decin   : std_logic;
64.          signal rd_once_decout   : std_logic;
65.          signal rsdecOutEnable_ff1   : std_logic;
66.
67.          signal rsDecDataFlag   : std_logic;
68.          signal rsDecNGDataFlag   : std_logic;
69.          signal rsDecErasureFlag   : std_logic;
70.          signal rsDecNGErasureFlag   : std_logic;
71.          signal rsDecErrorFlag   : std_logic;
72.          signal rsDecNGErrorFlag   : std_logic;
73.          signal rsDecFailPinFlag   : std_logic;
74.          signal rsDecNGFailPinFlag   : std_logic;
75.
76.          signal rsdec0_sig : std_logic_vector(23 downto 0);
77.
78.          signal rsdecExpData_sig    : std_logic_vector(7 downto 0);
79.          signal rsdecExpNumErasure  : std_logic_vector(4 downto 0);
80.          signal rsdecExpNumError    : std_logic_vector(4 downto 0);
81.          signal rsdecExpFailFlag    : std_logic;
82.          signal rsdecExpData        : std_logic_vector(4 downto 0);
83.          signal rsdecExpDelayedData : std_logic_vector(4 downto 0);
84.
85.          signal rsdecOutData_ff1  : std_logic_vector(4 downto 0);
86.          signal rsdecErrorNum_ff1    : std_logic_vector(4 downto 0);
87.          signal rsdecFail_ff1    : std_logic;
88.          signal rsdecErasureNum_ff1        : std_logic_vector(4
   downto 0);
89.
90.          signal data_count  : std_logic_vector(5 downto 0);
91.      -- coder --
92.          signal rd_once_in   : std_logic;
93.          signal rd_once_out   : std_logic;
94.          signal rsEncPassFailFlag : std_logic;
95.          signal rsEncFailFlag : std_logic;
96.          signal rsenc0_sig     : std_logic_vector(15 downto 0);
97.          signal rsencEnable_ff1   : std_logic;
98.          signal rsencEnable_ff2   : std_logic;
99.          signal rsencStartPls_ff1   : std_logic;
100.         signal rsencStartPls_ff2   : std_logic;
101.         signal rsencExpData_sig     : std_logic_vector(4 downto 0);
102.     begin
103.         -- RS  Decoder Top module Instantiation
104.         u_rsdecoder_31_23_top : rsdecoder_31_23_top
105.         port map(
106.           CLK          => CLK,              -- system clock
107.           RESET        => RESET,            -- system reset
108.           -- IN
109.           enable       => rsdecEnable,      -- RSdec enable in
```

```
110.           startPls     => rsdecSync,         -- RSdec sync signal
111.           erasureIn    => rsdecErasureIn,    -- RSdec erasure in
112.           dataIn       => rsdecDataIn,       -- RSdec data in
113.           -- OUT
114.           outEnable    => rsdecOutEnable,    -- RSdec enable out
115.           outStartPls  => rsdecOutStartPls,  -- RSdec start pulse
   out
116.           outDone      => rsdecOutDone,      -- RSdec done out
117.           errorNum     => rsdecErrorNum,     -- RSdec error number
118.           erasureNum   => rsdecErasureNum,   -- RSdec Erasure number
119.           fail         => rsdecFail,         -- RSdec Pass/Fail flag
120.           delayedData  => rsdecDelayedData,  -- RSdec delayed data
121.           outData      => rsdecOutData       -- Rsdec data out
122.         );
123.
124.          -- RS Encoder Top module Instantiation
125.          u_rscoder_31_23_top: rscoder_31_23_top
126.          port map(
127.            CLK      => CLK,           -- system clock
128.            RESET    => RESET,         -- system reset
129.            enable   => rsencEnable,   -- RSenc enable signal
130.            startPls => rsencStartPls, -- RSenc sync signal
131.            dataIn   => rsencDataIn,   -- RSenc data in
132.            dataOut  => rsencDataOut   -- RSenc data out
133.         );
134.
135.         -- Generate clock
136.          CLK_p:process
137.          begin
138.              CLK <= '0';
139.              wait for (CLK_PER/2);
140.              CLK <= '1';
141.              wait for (CLK_PER/2);
142.          end process;
143.
144.          rs_encsim : process (CLK,RESET) is
145.          variable file_status_in : file_open_status;
146.          variable file_status_out: file_open_status;
147.          file     mem_file_in        : TEXT;
148.          file     mem_file_out       : TEXT;
149.          variable mem_line_in     : line;
150.          variable mem_line_out    : line;
151.          variable rsenc0     : std_logic_vector(15 downto 0);
152.          variable rsencExpData  : std_logic_vector(7 downto 0);
153.          ---- DECODER
154.          variable file_status_decin : file_open_status;
155.          variable file_status_decout: file_open_status;
156.          file     mem_file_decin        : TEXT;
157.          file     mem_file_decout       : TEXT;
158.          variable mem_line_decin     : line;
159.          variable mem_line_decout    : line;
160.          variable rsdec0     : std_logic_vector(23 downto 0);
161.          variable rsdecExp  : std_logic_vector(7 downto 0);
162.
163.          begin
164.              if RESET = '0' then
165.                  rsencStartPls <= '0';
```

```
166.                 rsencEnable   <= '0';
167.                 rsencDataIn   <= (others => '0');
168.                 simStart_ff1  <= '0';
169.                 simStart_ff2  <= '0';
170.                 simStart_ff3  <= '0';
171.                 rd_once_in    <= '0';
172.                 rd_once_out   <= '0';
173.                 rsEncPassFailFlag <= '0';
174.                 rsEncFailFlag   <= '0';
175.                 ----------------------
176.                 rsdecSync          <= '0';
177.                 rsdecEnable        <= '0';
178.                 rsdecDataIn        <= (others => '0');
179.                 rd_once_decin      <= '0';
180.                 rd_once_decout     <= '0';
181.                 rsdecOutEnable_ff1 <= '0';
182.                 rsDecDataFlag      <= '0';
183.                 rsDecNGDataFlag    <= '0';
184.                 rsDecErasureFlag   <= '0';
185.                 rsDecNGErasureFlag <= '0';
186.                 rsDecErrorFlag     <= '0';
187.                 rsDecNGErrorFlag   <= '0';
188.                 rsDecFailPinFlag   <= '0';
189.                 rsDecNGFailPinFlag <= '0';
190.
191.                 rsdecExpData_sig        <= (others => '0');
192.                 rsdecExpNumErasure      <= (others => '0');
193.                 rsdecExpNumError        <= (others => '0');
194.                 rsdecExpFailFlag        <= '0';
195.                 rsdecExpData            <= (others => '0');
196.                 rsdecExpDelayedData     <= (others => '0');
197.
198.                 rsdecOutData_ff1      <= (others => '0');
199.                 rsdecErrorNum_ff1     <= (others => '0');
200.                 rsdecFail_ff1         <= '0';
201.                 rsdecErasureNum_ff1   <= (others => '0');
202.
203.                 rsencStartPls_ff1 <= '0';
204.                 rsencStartPls_ff2 <= '0';
205.                 data_count <=  (others => '0');
206.             elsif rising_edge(CLK) then
207.                 ---------------
208.                 simStart_ff1 <= simStart;
209.                 simStart_ff2 <= simStart_ff1;
210.                 simStart_ff3 <= simStart_ff2;
211.     ----------------------------------
212.                 --  ENCODER INPUT  ---
213.     ----------------------------------
214.                 if ((simStart_ff1 ='0') and (simStart ='1') and
    (rd_once_in='0')) then
215.                     file_open(file_status_in, mem_file_in,
    "RsEncIn.hex", READ_MODE);
216.                     rd_once_in   <= '1';
217.                 end if;
218.                 ---------------
219.                 if ((simStart ='1')) then
220.                     readline(mem_file_in, mem_line_in);
```

```
221.                     hread(mem_line_in, rsenc0);
222.                 end if;
223.                 ---------------
224.                 if (simStart ='1') then
225.                     rsenc0_sig    <= rsenc0;
226.                     rsencStartPls <= rsenc0(12);
227.                     rsencEnable   <= rsenc0(8);
228.                     rsencDataIn   <= rsenc0(4 downto 0);
229.                 end if;
230.     ---------------------------------
231.                 --  ENCODER OUTPUT  ---
232.     ---------------------------------
233.                 rsencEnable_ff1   <= rsencEnable;
234.                 rsencEnable_ff2   <= rsencEnable_ff1;
235.                 rsencStartPls_ff1 <= rsencStartPls;
236.                 rsencStartPls_ff2 <= rsencStartPls_ff1;
237.                 if ((rsencEnable_ff1 ='0') and (rsencEnable ='1')
   and (rd_once_out='0')) then
238.                     file_open(file_status_out, mem_file_out,
   "RsEncOut.hex", READ_MODE);
239.                     rd_once_out   <= '1';
240.                 end if;
241.                 ---------------
242.                 if ((simStart_ff2 ='1')) then
243.                     readline(mem_file_out, mem_line_out);
244.                     hread(mem_line_out, rsencExpData);
245.                     rsencExpData_sig <= rsencExpData(4 downto 0);
246.                 end if;
247.                 ---------------
248.                 if ((simStart_ff3 ='1')) then
249.                     if (rsencDataOut = rsencExpData_sig) then
250.                         rsEncPassFailFlag <= '0';
251.                     else
252.                         rsEncPassFailFlag <= '1';
253.                         rsEncFailFlag   <= '1';
254.                     end if;
255.                 end if;
256.     ---------------------------------
257.                 --  DECODER INPUT  ---
258.     ---------------------------------
259.                 ---------------
260.                 if ((simStart ='1')) then
261.                     rsdecSync      <= rsencStartPls_ff2;
262.                     rsdecEnable    <= rsencEnable_ff2;
263.                     rsdecErasureIn <= '0';
264.                     if ((data_count="001100") or
   (data_count="001110") or (data_count="010010") or
   (data_count="010011")) then
265.                         rsdecDataIn    <= not rsencDataOut;
266.                     else
267.                         rsdecDataIn    <= rsencDataOut;
268.                     end if;
269.                 end if;
270.                 ---------------
271.                 if ((rsencEnable_ff2='1')) then
272.                     if (rsencStartPls_ff2='1') then
273.                         data_count <=  (others => '0');
```

```
274.                     else
275.                         data_count <=
   std_logic_vector(unsigned(data_count) + 1);
276.                     end if;
277.                 end if;
278.     -------------------------------
279.                 --  DECODER OUTPUT  ---
280.     ---------------------------------
281.                 rsdecOutData_ff1      <= rsdecOutData;
282.                 rsdecErrorNum_ff1     <= rsdecErrorNum;
283.                 rsdecFail_ff1         <= rsdecFail;
284.                 rsdecErasureNum_ff1   <= rsdecErasureNum;
285.                 rsdecOutEnable_ff1 <= rsdecOutEnable;
286.                 if ((rsdecOutEnable_ff1 ='0') and (rsdecOutEnable
   ='1') and (rd_once_decout='0')) then
287.                     file_open(file_status_decout, mem_file_decout,
   "RsDecOut.hex", READ_MODE);
288.                     rd_once_decout   <= '1';
289.                 end if;
290.                 ---------------
291.                 if ((rsdecOutEnable ='1')) then
292.                     readline(mem_file_decout, mem_line_decout);
293.                     hread(mem_line_decout, rsdecExp);
294.                 end if;
295.                 if ((rsdecOutEnable ='1')) then
296.                     rsdecExpData        <= rsdecExp(4 downto 0);
297.                 end if;
298.                 ---- Data Pin ----
299.                 if (rsdecOutEnable_ff1 = '1') then
300.                     if (rsdecOutData_ff1 = rsdecExpData) then
301.                         rsDecDataFlag <= '0';
302.                     else
303.                         rsDecDataFlag   <= '1';
304.                         rsDecNGDataFlag <= '1';
305.                     end if;
306.                 else
307.                     rsDecDataFlag <= '0';
308.                 end if;
309.             end if;
310.         end process;
311.
312.     -------------------------------------------
313.     -- Generate reset and tb enable stimulus
314.         stimulus_p:process
315.         begin
316.            simStart  <= '0';
317.            RESET     <= '1';
318.            wait for 20 ns;
319.            RESET           <= '0';
320.            wait for 20 ns;
321.            RESET           <= '1';
322.            wait for 200 ns;
323.            simStart           <= '1';
324.            wait for 1000 ms;
325.            assert false report "End of simulation !" severity
   failure;
326.            wait;
```

```
327.          end process;
328.
329.
330.      end TB;
```

10.4.3. Erasure & Error test bench VHDL source code.

```
1. ================================================================================
2. -- Project Name         : RSIP
3. -- Name                 : simReedSolomon.vhd
4. -- Actual Version       : v0.1
5. ================================================================================
6.  -- Description          : erasure & error test environment
7. ================================================================================
8.  --         libraries
9. ================================================================================
10.       library ieee;
11.       use ieee.std_logic_1164.all;
12.       use ieee.numeric_std.all;
13.       use ieee.std_logic_textio.all;
14.       library std;
15.       use std.textio.all;
16.       library work;
17.       use work.rscoder_31_23_top_pkg.all;
18.       use work.rsdecoder_31_23_top_pkg.all;
19.      ================================================================
20.       --         TOP instantiation
21.      ================================================================
22.       entity simReedSolomon is
23.       end simReedSolomon;
24.      ================================================================
25.       --         RTL Architecture
26.      ================================================================
27.       architecture TB of simReedSolomon is
28.       constant CLK_PER :   time := 10 ns; -- 100Mhz
29.       ----------------------------------------------------------------
30.       -- Signals
31.       ----------------------------------------------------------------
32.       -- decoder --
33.           signal CLK        : std_logic;
34.           signal RESET     : std_logic;
35.           signal rsdecEnable     : std_logic;
36.           signal rsdecSync   : std_logic;
37.           signal rsdecErasureIn   : std_logic;
38.           signal rsdecDataIn   : std_logic_vector(4 downto 0);
39.           signal rsdecOutStartPls   : std_logic;
40.           signal rsdecOutDone   : std_logic;
41.           signal rsdecOutData   : std_logic_vector(4 downto 0);
42.           signal rsdecErrorNum   : std_logic_vector(4 downto 0);
43.           signal rsdecErasureNum   : std_logic_vector(4 downto 0);
44.           signal rsdecFail   : std_logic;
45.           signal rsdecOutEnable   : std_logic;
46.           signal rsdecDelayedData   : std_logic_vector(4 downto 0);
47.
48.           signal rsencEnable   : std_logic;
49.           signal rsencStartPls   : std_logic;
50.           signal rsencDataIn   : std_logic_vector(4 downto 0);
51.           signal rsencDataOut   : std_logic_vector(4 downto 0);
52.
53.           Signal rsdecOutEnableFF: std_logic;
```

```
54.           signal rsdecOutDataFF   : std_logic_vector(4 downto 0);
55.           signal rsdecErasureNumFF   : std_logic_vector(4 downto 0);
56.           signal rsdecErrorNumFF   : std_logic_vector(4 downto 0);
57.           signal rsdecFailFF   : std_logic;
58.
59.           signal simStart   : std_logic;
60.           signal simStart_ff1   : std_logic;
61.           signal simStart_ff2   : std_logic;
62.           signal simStart_ff3   : std_logic;
63.           signal rd_once_decin   : std_logic;
64.           signal rd_once_decout   : std_logic;
65.           signal rsdecOutEnable_ff1   : std_logic;
66.
67.           signal rsDecDataFlag   : std_logic;
68.           signal rsDecNGDataFlag   : std_logic;
69.           signal rsDecErasureFlag   : std_logic;
70.           signal rsDecNGErasureFlag   : std_logic;
71.           signal rsDecErrorFlag   : std_logic;
72.           signal rsDecNGErrorFlag   : std_logic;
73.           signal rsDecFailPinFlag   : std_logic;
74.           signal rsDecNGFailPinFlag   : std_logic;
75.
76.           signal rsdec0_sig : std_logic_vector(23 downto 0);
77.
78.           signal rsdecExpData_sig    : std_logic_vector(7 downto 0);
79.           signal rsdecExpNumErasure  : std_logic_vector(4 downto 0);
80.           signal rsdecExpNumError    : std_logic_vector(4 downto 0);
81.           signal rsdecExpFailFlag    : std_logic;
82.           signal rsdecExpData        : std_logic_vector(4 downto 0);
83.           signal rsdecExpDelayedData : std_logic_vector(4 downto 0);
84.
85.           signal rsdecOutData_ff1  : std_logic_vector(4 downto 0);
86.           signal rsdecErrorNum_ff1    : std_logic_vector(4 downto 0);
87.           signal rsdecFail_ff1    : std_logic;
88.           signal rsdecErasureNum_ff1        : std_logic_vector(4
   downto 0);
89.
90.           signal data_count  : std_logic_vector(5 downto 0);
91.       -- coder --
92.           signal rd_once_in   : std_logic;
93.           signal rd_once_out   : std_logic;
94.           signal rsEncPassFailFlag : std_logic;
95.           signal rsEncFailFlag : std_logic;
96.           signal rsenc0_sig     : std_logic_vector(15 downto 0);
97.           signal rsencEnable_ff1   : std_logic;
98.           signal rsencEnable_ff2   : std_logic;
99.           signal rsencStartPls_ff1   : std_logic;
100.          signal rsencStartPls_ff2   : std_logic;
101.          signal rsencExpData_sig     : std_logic_vector(4 downto 0);
102.      begin
103.          ---------------------------------------------
104.          -- RS  Decoder Top module Instantiation
105.          u_rsdecoder_31_23_top : rsdecoder_31_23_top
106.          port map(
107.            CLK          => CLK,               -- system clock
108.            RESET        => RESET,             -- system reset
109.            -- IN
```

```
110.           enable       => rsdecEnable,       -- RSdec enable in
111.           startPls     => rsdecSync,         -- RSdec sync signal
112.           erasureIn    => rsdecErasureIn,    -- RSdec erasure in
113.           dataIn       => rsdecDataIn,       -- RSdec data in
114.           -- OUT
115.           outEnable    => rsdecOutEnable,    -- RSdec enable out
116.           outStartPls  => rsdecOutStartPls,  -- RSdec start pulse
   out
117.           outDone      => rsdecOutDone,      -- RSdec done out
118.           errorNum     => rsdecErrorNum,     -- RSdec error number
119.           erasureNum   => rsdecErasureNum,   -- RSdec Erasure number
120.           fail         => rsdecFail,         -- RSdec Pass/Fail flag
121.           delayedData  => rsdecDelayedData,  -- RSdec delayed data
122.           outData      => rsdecOutData       -- Rsdec data out
123.         );
124.          -- RS Encoder Top module Instantiation
125.          u_rscoder_31_23_top: rscoder_31_23_top
126.          port map(
127.            CLK      => CLK,           -- system clock
128.            RESET    => RESET,         -- system reset
129.            enable   => rsencEnable,   -- RSenc enable signal
130.            startPls => rsencStartPls, -- RSenc sync signal
131.            dataIn   => rsencDataIn,   -- RSenc data in
132.            dataOut  => rsencDataOut   -- RSenc data out
133.         );
134.         -- Generate clock
135.          CLK_p:process
136.          begin
137.              CLK <= '0';
138.              wait for (CLK_PER/2);
139.              CLK <= '1';
140.              wait for (CLK_PER/2);
141.          end process;
142.             rs_encsim : process (CLK,RESET) is
143.          variable file_status_in : file_open_status;
144.          variable file_status_out: file_open_status;
145.          file     mem_file_in        : TEXT;
146.          file     mem_file_out       : TEXT;
147.          variable mem_line_in     : line;
148.          variable mem_line_out    : line;
149.          variable rsenc0     : std_logic_vector(15 downto 0);
150.          variable rsencExpData  : std_logic_vector(7 downto 0);
151.          ---- DECODER
152.          variable file_status_decin : file_open_status;
153.          variable file_status_decout: file_open_status;
154.          file     mem_file_decin        : TEXT;
155.          file     mem_file_decout       : TEXT;
156.          variable mem_line_decin     : line;
157.          variable mem_line_decout    : line;
158.          variable rsdec0     : std_logic_vector(23 downto 0);
159.          variable rsdecExp  : std_logic_vector(7 downto 0);
160.
161.          begin
162.              if RESET = '0' then
163.                  rsencStartPls <= '0';
164.                  rsencEnable   <= '0';
165.                  rsencDataIn   <= (others => '0');
```

```
166.                 simStart_ff1  <= '0';
167.                 simStart_ff2  <= '0';
168.                 simStart_ff3  <= '0';
169.                 rd_once_in    <= '0';
170.                 rd_once_out   <= '0';
171.                 rsEncPassFailFlag <= '0';
172.                 rsEncFailFlag   <= '0';
173.                 -----------------------
174.                 rsdecSync          <= '0';
175.                 rsdecEnable        <= '0';
176.                 rsdecDataIn        <= (others => '0');
177.                 rd_once_decin      <= '0';
178.                 rd_once_decout     <= '0';
179.                 rsdecOutEnable_ff1 <= '0';
180.                 rsDecDataFlag      <= '0';
181.                 rsDecNGDataFlag    <= '0';
182.                 rsDecErasureFlag   <= '0';
183.                 rsDecNGErasureFlag <= '0';
184.                 rsDecErrorFlag     <= '0';
185.                 rsDecNGErrorFlag   <= '0';
186.                 rsDecFailPinFlag   <= '0';
187.                 rsDecNGFailPinFlag <= '0';
188.
189.                 rsdecExpData_sig        <= (others => '0');
190.                 rsdecExpNumErasure      <= (others => '0');
191.                 rsdecExpNumError        <= (others => '0');
192.                 rsdecExpFailFlag        <= '0';
193.                 rsdecExpData            <= (others => '0');
194.                 rsdecExpDelayedData     <= (others => '0');
195.
196.                 rsdecOutData_ff1      <= (others => '0');
197.                 rsdecErrorNum_ff1     <= (others => '0');
198.                 rsdecFail_ff1         <= '0';
199.                 rsdecErasureNum_ff1   <= (others => '0');
200.
201.                 rsencStartPls_ff1 <= '0';
202.                 rsencStartPls_ff2 <= '0';
203.                 data_count <=  (others => '0');
204.             elsif rising_edge(CLK) then
205.
206.                 simStart_ff1 <= simStart;
207.                 simStart_ff2 <= simStart_ff1;
208.                 simStart_ff3 <= simStart_ff2;
209.     ----------------------------------
210.                 --  ENCODER INPUT  ---
211.     ----------------------------------
212.                 if ((simStart_ff1 ='0') and (simStart ='1') and
   (rd_once_in='0')) then
213.                     file_open(file_status_in, mem_file_in,
   "RsEncIn.hex", READ_MODE);
214.                     rd_once_in   <= '1';
215.                 end if;
216.                 ---------------
217.                 if ((simStart ='1')) then
218.                     readline(mem_file_in, mem_line_in);
219.                     hread(mem_line_in, rsenc0);
220.                 end if;
```

```
221.                 ---------------
222.                 if (simStart ='1') then
223.                     rsenc0_sig    <= rsenc0;
224.                     rsencStartPls <= rsenc0(12);
225.                     rsencEnable   <= rsenc0(8);
226.                     rsencDataIn   <= rsenc0(4 downto 0);
227.                 end if;
228.     ---------------------------------
229.                 --  ENCODER OUTPUT  ---
230.     ---------------------------------
231.                 rsencEnable_ff1   <= rsencEnable;
232.                 rsencEnable_ff2   <= rsencEnable_ff1;
233.                 rsencStartPls_ff1 <= rsencStartPls;
234.                 rsencStartPls_ff2 <= rsencStartPls_ff1;
235.                 if ((rsencEnable_ff1 ='0') and (rsencEnable ='1')
   and (rd_once_out='0')) then
236.                     file_open(file_status_out, mem_file_out,
   "RsEncOut.hex", READ_MODE);
237.                     rd_once_out   <= '1';
238.                 end if;
239.                 ---------------
240.                 if ((simStart_ff2 ='1')) then
241.                     readline(mem_file_out, mem_line_out);
242.                     hread(mem_line_out, rsencExpData);
243.                     rsencExpData_sig <= rsencExpData(4 downto 0);
244.                 end if;
245.                 ---------------
246.                 if ((simStart_ff3 ='1')) then
247.                     if (rsencDataOut = rsencExpData_sig) then
248.                         rsEncPassFailFlag <= '0';
249.                     else
250.                         rsEncPassFailFlag <= '1';
251.                         rsEncFailFlag   <= '1';
252.                     end if;
253.                 end if;
254.     ---------------------------------
255.                 --  DECODER INPUT  ---
256.     ---------------------------------
257.                           if ((simStart ='1')) then
258.                     rsdecSync      <= rsencStartPls_ff2;
259.                     rsdecEnable    <= rsencEnable_ff2;
260.                     -- insert erasures
261.                     if ((data_count="000010") or
   (data_count="000100") or (data_count="000110") or
   (data_count="001010")) then
262.                         rsdecErasureIn <= '1';
263.                         rsdecDataIn    <= (others => '0');
264.                     -- insert errors
265.                     elsif ((data_count="001100") or
   (data_count="001110")) then
266.                         rsdecErasureIn <= '0';
267.                         rsdecDataIn    <= not rsencDataOut;
268.                     -- insert normal data
269.                     else
270.                         rsdecErasureIn <= '0';
271.                         rsdecDataIn    <= rsencDataOut;
272.                     end if;
```

```
273.                 end if;
274.                 ---------------
275.                 if ((rsencEnable_ff2='1')) then
276.                     if (rsencStartPls_ff2='1') then
277.                         data_count <=  (others => '0');
278.                     else
279.                        data_count <=
   std_logic_vector(unsigned(data_count) + 1);
280.                     end if;
281.                 end if;
282.     ---------------------------------
283.                 --  DECODER OUTPUT  ---
284.     ---------------------------------
285.                 rsdecOutData_ff1      <= rsdecOutData;
286.                 rsdecErrorNum_ff1     <= rsdecErrorNum;
287.                 rsdecFail_ff1         <= rsdecFail;
288.                 rsdecErasureNum_ff1   <= rsdecErasureNum;
289.                 rsdecOutEnable_ff1 <= rsdecOutEnable;
290.                 if ((rsdecOutEnable_ff1 ='0') and (rsdecOutEnable
   ='1') and (rd_once_decout='0')) then
291.                     file_open(file_status_decout, mem_file_decout,
   "RsDecOut.hex", READ_MODE);
292.                     rd_once_decout   <= '1';
293.                 end if;
294.                 ---------------
295.                 if ((rsdecOutEnable ='1')) then
296.                     readline(mem_file_decout, mem_line_decout);
297.                     hread(mem_line_decout, rsdecExp);
298.                 end if;
299.                 if ((rsdecOutEnable ='1')) then
300.                     rsdecExpData        <= rsdecExp(4 downto 0);
301.                 end if;
302.                 ---- Data Pin ----
303.                 if (rsdecOutEnable_ff1 = '1') then
304.                     if (rsdecOutData_ff1 = rsdecExpData) then
305.                         rsDecDataFlag <= '0';
306.                     else
307.                         rsDecDataFlag   <= '1';
308.                         rsDecNGDataFlag <= '1';
309.                     end if;
310.                 else
311.                     rsDecDataFlag <= '0';
312.                 end if;
313.             end if;
314.         end process;
315.     -------------------------------------------
316.     -- Generate reset and tb enable stimulus
317.         stimulus_p:process
318.         begin
319.            simStart  <= '0';
320.            RESET     <= '1';
321.            wait for 20 ns;
322.            RESET           <= '0';
323.            wait for 20 ns;
324.            RESET           <= '1';
325.            wait for 200 ns;
326.            simStart           <= '1';
```

```
327.             wait for 1000 ms;
328.             assert false report "End of simulation !" severity
   failure;
329.             wait;
330.          end process;
331.
332.      end TB;
```

10.5. Appendix 5

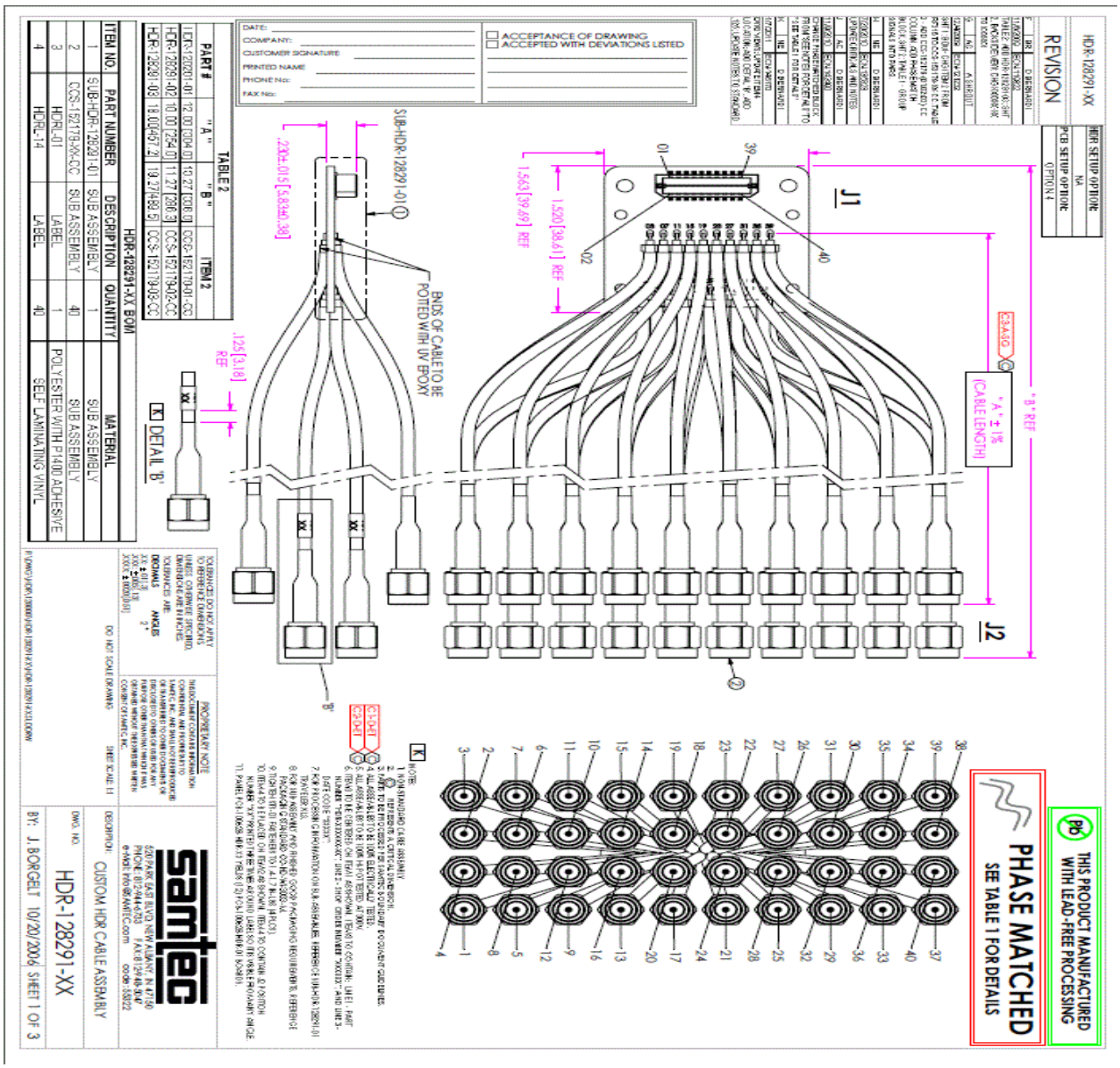
10.5.1. Field of (31,23)RS code.

Polynomial Generator is:- X^5+X^2+1

Power	Polynomial	a_0	a_1	a_2	a_3	a_4
0	0	0	0	0	0	0
1	1	1	0	0	0	0
α	α	0	1	0	0	0
α^2	α^2	0	0	1	0	0
α^3	α^3	0	0	0	1	0
α^4	α^4	0	0	0	0	1
α^5	α^2+1	1	0	1	0	0
α^6	$\alpha^3+\alpha$	0	1	0	1	0
α^7	$\alpha^4+\alpha^2$	0	0	1	0	1
α^8	$\alpha^3+\alpha^2+1$	1	0	1	1	0
α^9	$\alpha^4+\alpha^3+\alpha$	0	1	0	1	1
α^{10}	α^4+1	1	0	0	0	1
α^{11}	$\alpha^2+\alpha+1$	1	1	1	0	0
α^{12}	$\alpha^3+\alpha^2+\alpha$	0	1	1	1	0
α^{13}	$\alpha^4+\alpha^3+\alpha^2$	0	0	1	1	1
α^{14}	$\alpha^4+\alpha^3+\alpha^2+1$	1	0	1	1	1
α^{15}	$\alpha^4+\alpha^3+\alpha^2+\alpha+1$	1	1	1	1	1
α^{16}	$\alpha^4+\alpha^3+\alpha+1$	1	1	0	1	1
α^{17}	$\alpha^4+\alpha+1$	1	1	0	0	1
α^{18}	$\alpha+1$	1	1	0	0	0
α^{19}	$\alpha^2+\alpha$	0	1	1	0	0
α^{20}	$\alpha^3+\alpha^2$	0	0	1	1	0
α^{21}	$\alpha^4+\alpha^3$	0	0	0	1	1
α^{22}	$\alpha^4+\alpha^2+1$	1	0	1	0	1
α^{23}	$\alpha^3+\alpha^2+\alpha+1$	1	1	1	1	0
α^{24}	$\alpha^4+\alpha^3+\alpha^2+\alpha$	0	1	1	1	1
α^{25}	$\alpha^4+\alpha^3+1$	1	0	0	1	1
α^{26}	$\alpha^4+\alpha^2+\alpha+1$	1	1	1	0	1
α^{27}	$\alpha^3+\alpha+1$	1	1	0	1	0
α^{28}	$\alpha^4+\alpha^2+\alpha$	0	1	1	0	1
α^{29}	α^3+1	1	0	0	1	0
α^{30}	$\alpha^4+\alpha$	0	1	0	0	1

10.6. Appendix 6

10.6.1. SMA breakout cables data sheet.



HDR-128291-XX

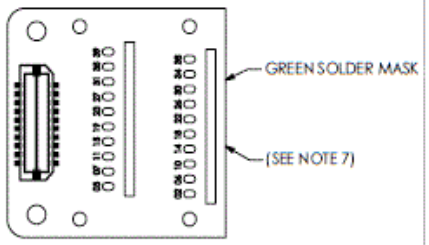
REVISION K

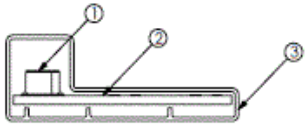
THIS PRODUCT MANUFACTURED WITH LEAD-FREE PROCESSING

HDR SETUP OPTION:
SUB-MQCD-01-1 BANK
PCB SETUP OPTION:
OPTION 4

J1

SUB-HDR-128291-01

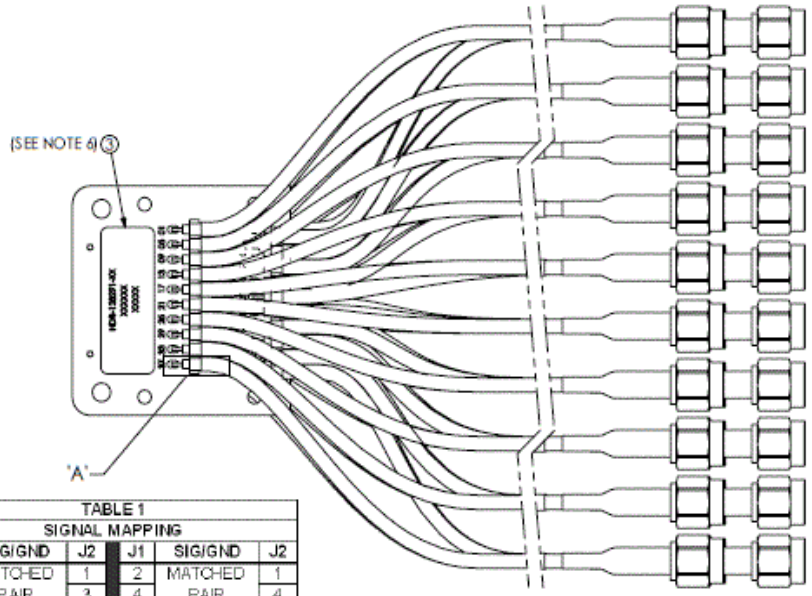




SUB-HDR-128291-01 BOM			
ITEM NO.	PART NUMBER	QUANTITY	MATERIAL
1	DTH-020-01-H-D-DP-A	1	TERMINAL
2	PCB 100628 HDR 01	1	PCB
3	PT-1-24-01-157	.06667	TUBE
4	TP-07	.13333	PLUG

BACKSIDE VIEW

PHASE MATCHED
SEE TABLE 1 FOR DETAILS



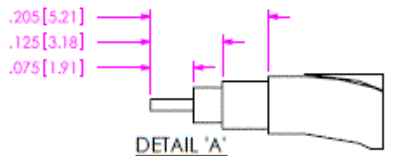
(SEE NOTE 10)

TABLE 1
SIGNAL MAPPING

J1	SIG/GND	J2	J1	SIG/GND	J2
1	MATCHED	1	2	MATCHED	1
3	PAIR	3	4	PAIR	4
5	MATCHED	5	6	MATCHED	6
7	PAIR	7	8	PAIR	8
9	MATCHED	9	10	MATCHED	10
11	PAIR	11	12	PAIR	12
13	MATCHED	13	14	MATCHED	14
15	PAIR	15	16	PAIR	16
17	MATCHED	17	18	MATCHED	18
19	PAIR	19	20	PAIR	20
21	MATCHED	21	22	MATCHED	22
23	PAIR	23	24	PAIR	24
25	MATCHED	25	26	MATCHED	26
27	PAIR	27	28	PAIR	28
29	MATCHED	29	30	MATCHED	30
31	PAIR	31	32	PAIR	32
33	MATCHED	33	34	MATCHED	34
35	PAIR	35	36	PAIR	36
37	MATCHED	37	38	MATCHED	38
39	PAIR	39	40	PAIR	40

CAUTION: EACH PAIR OF CABLES MUST BE PHASE MATCHED TO WITHIN +/- 5 PICO-SECONDS. IF RE-WORKED, THE ENTIRE PAIR MUST BE REPLACED

STRIP DIMENSIONS



DETAIL 'A'
SCALE 4:1

F:\DWGS\HDR\128000\HDR-128291-XX\HDR-128291-XX.SLDDRW

samtec
520 PARK EAST BLVD, NEW ALBANY, IN 47150
PHONE: 812-944-6733 FAX: 812-948-5047
e-Mail: info@SAMTEC.com code: 53822

SHEET SCALE: 1:1

DESCRIPTION: CUSTOM HDR CABLE ASSEMBLY

DWG. NO. HDR-128291-XX

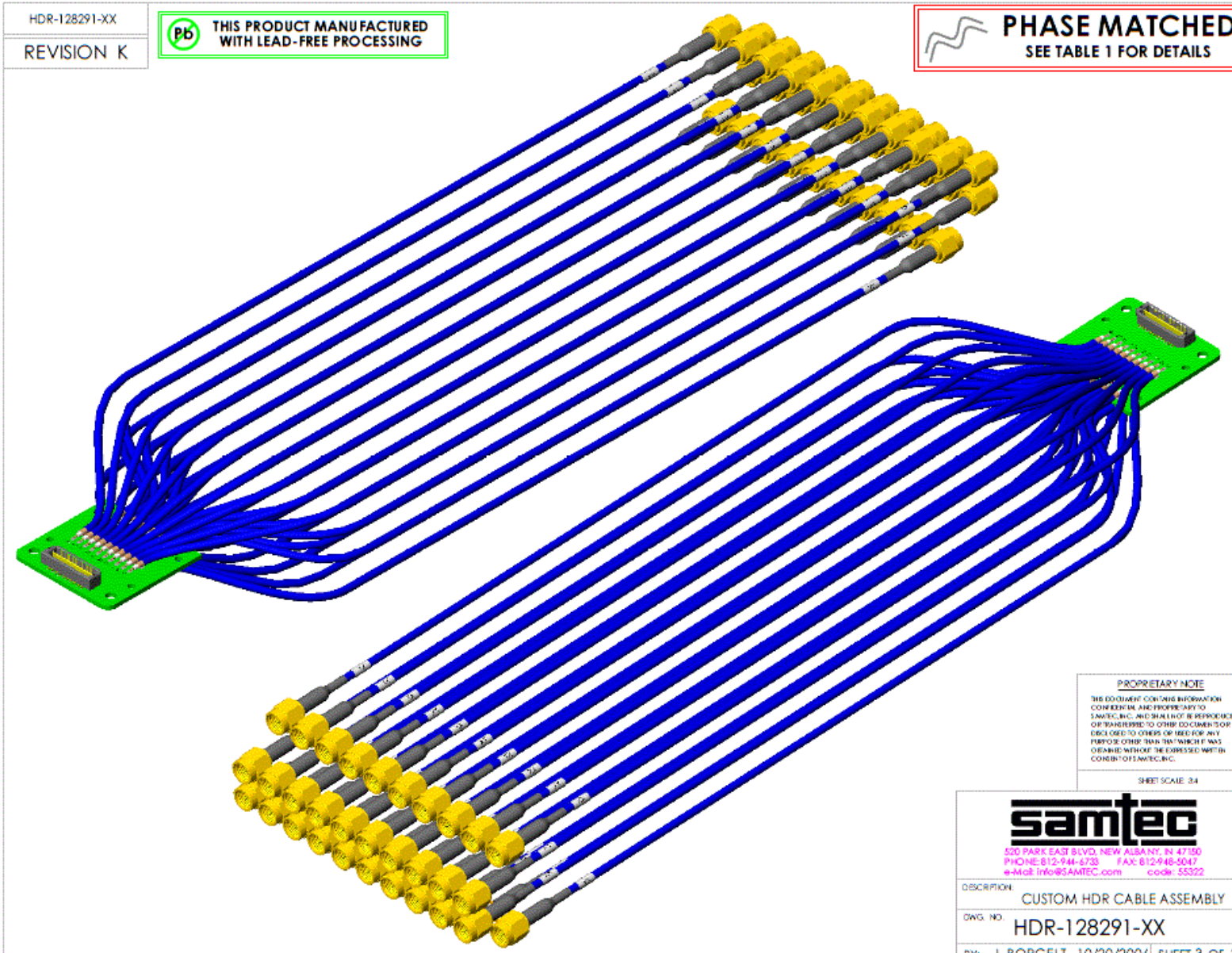
BY: J. BORGELT 10/20/2006 SHEET 2 OF 3

HDR-128291-XX

REVISION K

THIS PRODUCT MANUFACTURED WITH LEAD-FREE PROCESSING

PHASE MATCHED
SEE TABLE 1 FOR DETAILS



PROPRIETARY NOTE

THIS DOCUMENT CONTAINS INFORMATION CONFIDENTIAL AND PROPRIETARY TO SAMTEC, INC. AND SHALL NOT BE REPRODUCED OR TRANSFERRED TO OTHER DOCUMENTS OR DISCLOSED TO OTHERS OR USED FOR ANY PURPOSE OTHER THAN THAT WHICH IT WAS ORIGINALLY WITHOUT THE EXPRESSED WRITTEN CONSENT OF SAMTEC, INC.

SHEET SCALE: 3/4

samtec
520 PARK EAST BLVD, NEW ALBANY, IN 47150
PHONE: 812-944-6733 FAX: 812-948-5047
e-Mail: info@SAMTEC.com code: 55322

DESCRIPTION: CUSTOM HDR CABLE ASSEMBLY

DWG. NO: HDR-128291-XX

BY: J. BORGELT 10/20/2006 SHEET 3 OF 3

F:\DWGS\HDR\128000\HDR-128291-XX\HDR-128291-XX.SLDDRW

10.6.2. Optical transmitter & receiver data sheet.

HFBR-0507Z Series
HFBR-15X7Z Transmitters
HFBR-25X6Z Receivers

125 Megabaud Versatile Link
The Versatile Fiber Optic Connection

Data Sheet

Description

The 125 MBd Versatile Link (HFBR-0507Z Series) is the most cost-effective fiber-optic solution for transmission of 125 MBd data over 100 meters. The data link consists of a 650 nm LED transmitter, HFBR-15X7Z, and a PIN/preamp receiver, HFBR-25X6Z. These can be used with low-cost plastic or silica fiber. One mm diameter plastic fiber provides the lowest cost solution for distances under 25 meters. The lower attenuation of silica fiber allows data transmission over longer distance, for a small difference in cost. These components can be used for high speed data links without the problems common with copper wire solutions, at a competitive cost.

The HFBR-15X7Z transmitter is a high power 650 nm LED in a low cost plastic housing designed to efficiently couple power into 1 mm diameter plastic optical fiber and 200 µm Hard Clad Silica (HCS®) fiber. With the recommended drive circuit, the LED operates at speeds from 1-125 MBd. The HFBR-25X6Z is a high bandwidth analog receiver containing a PIN photodiode and internal transimpedance amplifier. With the recommended application circuit for 125 MBd operation, the performance of the complete data link is specified for of 0-25 meters with plastic fiber and 0-100 meters with 200 µm HCS® fiber. A wide variety of other digitizing circuits can be combined with the HFBR-0507Z Series to optimize perfor-mance and cost at higher and lower data rates.

Features

- RoHS-compliant
- Data transmission at signal rates of 1 to 125 MBd over distances of 100 meters
- Compatible with inexpensive, easily terminated plastic optical fiber, and with large core silica fiber
- High voltage isolation
- Transmitter and receiver application circuit schematics and recommended board layouts available
- Interlocking feature for single channel or duplex links, in a vertical or horizontal mount configuration

Applications

- Intra-system links: board-to-board, rack-to-rack
- Telecommunications switching systems
- Computer-to-peripheral data links, PC bus extension
- Industrial control
- Proprietary LANs
- Digitized video
- Medical instruments
- Reduction of lightning and voltage transient susceptibility

HCS® is a registered trademark of Spectran Corporation.

HFBR-0507Z Series
125 MBd Data Link
Data link operating conditions and performance are specified for the HFBR-15X7Z transmitter and HFBR-25X6Z receiver in the recommended applications circuits shown in Figure 1. This circuit has been optimized for 125 MBd operation. The Applications Engineering Department in the Avago Optical Communication Division is available to assist in optimizing link performance for higher or lower speed operation.

Recommended Operating Conditions for the Circuits in Figures 1 and 2.

Parameter	Symbol	Min.	Max.	Unit	Reference
Ambient Temperature	T_A	0	70	°C	
Supply Voltage	V_{CC}	+4.75	+5.25	V	
Data Input Voltage – Low	V_{IL}	V_{CC}-1.89	V_{CC}-1.62	V	
Data Input Voltage – High	V_{IH}	V_{CC}-1.06	V_{CC}-0.70	V	
Data Output Load	R_L	45	55	Ω	Note 1
Signaling Rate	f_S	1	125	MBd	
Duty Cycle	D.C.	40	60	%	Note 2

Link Performance: 1-125 MBd, BER ≤ 10^{-9}, under recommended operating conditions with recommended transmit and receive application circuits.

Parameter	Symbol	Min.[3]	Typ.[4]	Max.	Unit	Condition	Reference
Optical Power Budget, 1 m POF	OPB_{POF}	11	16		dB		Note 5,6,7
Optical Power Margin, 20 m Standard POF	$OPM_{POF,20}$	3	6		dB		Note 5,6,7
Link Distance with Standard 1 mm POF	l	20	27		m		
Optical Power Margin, 25 m Low Loss POF	$OPM_{POF,25}$	3	6		dB		Note 5,6,7
Link Distance with Extra Low Loss 1 mm POF	l	25	32		m		
Optical Power Budget, 1 m HCS	OPB_{HCS}	7	12		dB		Note 5,6,7
Optical Power Margin, 100 m HCS	$OPM_{HCS,100}$	3	6		dB		Note 5,6,7
Link Distance with HCS Cable	l	100	125		m		

Notes:
1. If the output of U4C in Figure 1, page 4 is transmitted via coaxial cable, terminate with a 50 Ω resistor to V_{CC}- 2 V.
2. Run length limited code with maximum run length of 10 µs.
3. Minimum link performance is projected based on the worst case specifications of the HFBR-15X7Z transmitter, HFBR-25X6Z receiver, and POF cable, and the typical performance of other components (e.g. logic gates, transistors, resistors, capacitors, quantizer, HCS cable).
4. Typical performance is at 25°C, 125 MBd, and is measured with typical values of all circuit components.
5. Standard cable is HFBR-RXXYYYZ plastic optical fiber , with a maximum attenuation of 0.24 dB/ m at 650 nm and NA = 0.5.
   Extra low loss cable is HFBR-EXXYYYZ plastic optical fiber, with a maximum attenuation of 0.19 dB/ m at 650 nm and NA = 0.5.
   HCS cable is HFBR-H/ VXXYYY glass optical fiber, with a maximum attenuation of 10 dB/ km at 650 nm and NA = 0.37.
6. Optical Power Budget is the difference between the transmitter output power and the receiver sensitivity, measured after 1 meter of fiber. The minimum OPB is based on the limits of optical component performance over temperature, process, and recommended power supply variation.
7. The Optical Power Margin is the available OPB after including the effects of attenuation and modal dispersion for the minimum link distance: OPM = OPB - (attenuation power loss + modal dispersion power penalty). The minimum OPM is the margin available for long term LED LOP degradation and additional fixed passive losses (such as in-line connectors) in addition to the minimum specified distance.

Plastic Optical Fiber (1 mm POF) Transmitter Application Circuit: Performance of the HFBR-15X7Z transmitter in the recommended application circuit (Figure 1) for POF; 1-125 M Bd, 25°C.

Parameter	Symbol	Typical	Unit	Condition	Note
Average Optical Power 1 mm POF	P_{avg}	-9.7	dBm	50% Duty Cycle	Note 1, Fig 3
Average Modulated Power 1 mm POF	P_{mod}	-11.3	dBm		Note 2, Fig 3
Optical Rise Time (10% to 90%)	t_r	2.1	ns	5 M Hz	
Optical Fall Time (90% to 10%)	t_f	2.8	ns	5 M Hz	
High Level LED Current (On)	$I_{F,H}$	19	mA		Note 3
Low Level LED Current (Off)	$I_{F,L}$	3	mA		Note 3
Optical Overshoot - 1 mm POF		45	%		
Transmitter Application Circuit Current Consumption - 1 mm POF	I_{CC}	110	mA		Figure 1

Hard Clad Silica Fiber (200 µm HCS) Transmitter Application Circuit: Performance of the HFBR-15X7Z transmitter in the recommended application circuit (Figure 1) for HCS; 1-125 M Bd, 25°C.

Parameter	Symbol	Typical	Unit	Condition	Note
Average Optical Power 200 µm HCS	P_{avg}	-14.6	dBm	50% Duty Cycle	Note 1, Fig 3
Average Modulated Power 200 µm HCS	P_{mod}	-16.2	dBm		Note 2, Fig 3
Optical Rise Time (10% to 90%)	t_r	3.1	ns	5 M Hz	
Optical Fall Time (90% to 10%)	t_f	3.4	ns	5 M Hz	
High Level LED Current (On)	$I_{F,H}$	60	mA		Note 3
Low Level LED Current (Off)	$I_{F,L}$	6	mA		Note 3
Optical Overshoot - 200 µm HCS		30	%		
Transmitter Application Circuit Current Consumption - 200 µm HCS	I_{CC}	130	mA		Figure 1

Notes:

1. Average optical power is measured with an average power meter at 50% duty cycle, after 1 meter of fiber.
2. To allow the LED to switch at high speeds, the recommended drive circuit modulates LED light output between two non-zero power levels. The modulated (useful) power is the difference between the high and low level of light output power (transmitted) or input power (received), which can be measured with an average power meter as a function of duty cycle (see Figure 3). Average Modulated Power is defined as one half the slope of the average power versus duty cycle:

$$\text{Average Modulated Power} = \frac{[P_{avg}@80\%\text{ duty cycle} - P_{avg}@20\%\text{ duty cycle}]}{(2)\,[0.80 - 0.20]}$$

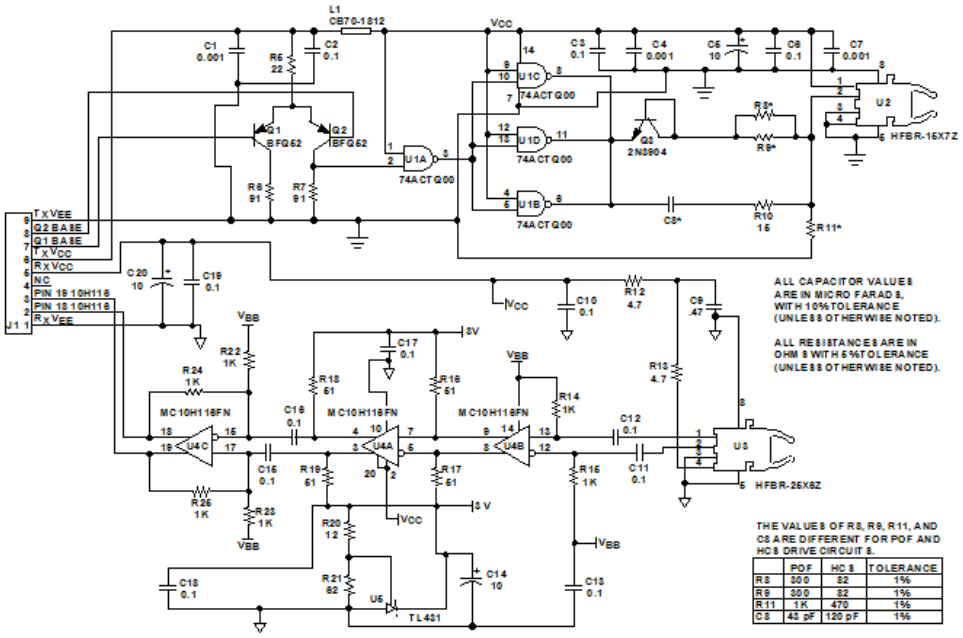
3. High and low level LED currents refer to the current through the HFBR-15X7Z LED. The low level LED "off" current, sometimes referred to as "hold-on" current, is prebias supplied to the LED during the off state to facilitate fast switching speeds.

Plastic and Hard Clad Silica Optical Fiber Receiver Application Circuit: Performance[4] of the HFBR-25X6Z receiver in the recommended application circuit (Figure 1); 1-125 MBd, 25°C unless otherwise stated.

Parameter	Symbol	Typical	Unit	Condition	Note
Data Output Voltage - Low	V_{OL}	V_{CC} -1.7	V	R_L = 50 Ω	Note 5
Data Output Voltage - High	V_{OH}	V_{CC} -0.9	V	R_L = 50 Ω	Note 5
Receiver Sensitivity to Average Modulated Optical Power 1 mm POF	P_{min}	-27.5	dBm	50% eye opening	Note 2
Receiver Sensitivity to Average Modulated Optical Power 200 µm HCS	P_{min}	-28.5	dBm	50% eye opening	Note 2
Receiver Overdrive Level of Average Modulated Optical Power 1 mm POF	P_{max}	-7.5	dBm	50% eye opening	Note 2
Receiver Overdrive Level of Average Modulated Optical Power 200 µm HCS	P_{max}	-10.5	dBm	50% eye opening	Note 2
Receiver Application Circuit Current Consumption	I_{CC}	85	mA	R_L = ∞	Figure 1

Notes:

4. Performance in response to a signal from the HFBR-15X7Z transmitter driven with the recommended circuit at 1-125 MBd over 1 meter of HFBR-RZ/ EXXYYYZ plastic optical fiber or 1 meter of HFBR-H/ VXXYYYH hard clad silica optical fiber.
5. Terminated through a 50 Ω resistor to V_{CC} - 2 V.
6. If there is no input optical power to the receiver, electrical noise can result in false triggering of the receiver. In typical applications, data encoding and error detection prevent random triggering from being interpreted as valid data. Refer to Applications Note 1066 for design guidelines.



	POF	HCS	TOLERANCE
R8	300	82	1%
R9	300	82	1%
R11	1K	470	1%
C8	43 pF	120 pF	1%

Figure 1. Transmitter and receiver application circuit with + 5 V ECL inputs and outputs.

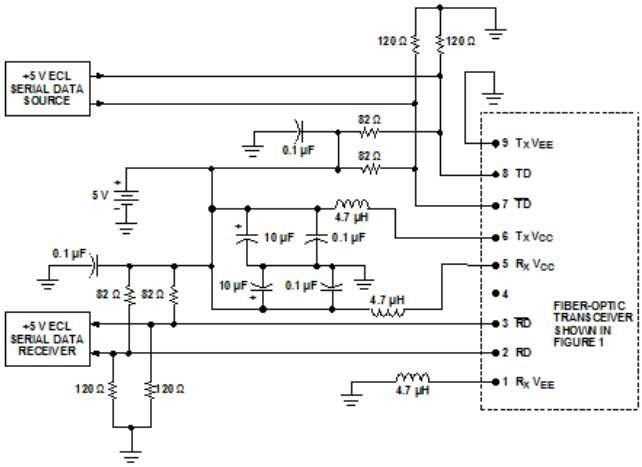


Figure 2. Recommended power supply filter and + 5 V ECL signal terminations for the transmitter and receiver application circuit of Figure 1.

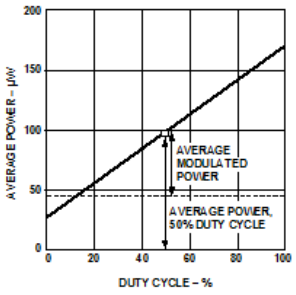


Figure 3. Average modulated power.

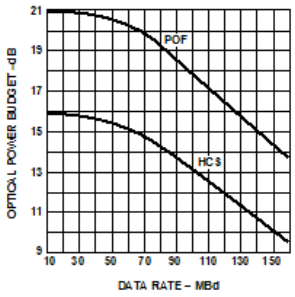


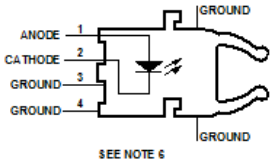
Figure 4. Typical optical power budget vs. data rate.

125 Megabaud Versatile Link Transmitter

HFBR-15X7Z Series

Description

The HFBR-15X7Z transmitters incorporate a 650 nanometer LED in a horizontal (HFBR-1527Z) or vertical (HFBR-1537Z) gray housing. The HFBR-15X7Z transmitters are suitable for use with current peaking to decrease response time and can be used with HFBR-25X6Z receivers in data links operating at signal rates from 1 to 125 megabaud over 1 mm diameter plastic optical fiber or 200 µm diameter hard clad silica glass optical fiber. Refer to Application Note 1066 for details for recommended interface circuits.



Absolute Maximum Ratings

Parameter	Symbol	Min.	Max.	Unit	Reference
Storage Temperature	T_S	-40	85	°C	
Operating Temperature	T_O	-40	70	°C	
Lead Soldering Temperature Cycle Time			260	°C	Note 1
			10	s	
Transmitter High Level Forward Input Current	$I_{F,H}$		120	mA	50% Duty Cycle ≥ 1 MHz
Transmitter Average Forward Input Current	$I_{F,AV}$		60	mA	
Reverse Input Voltage	V_R		3	V	

CAUTION: The small junction sizes inherent to the design of this component increase the component's susceptibility to damage from electrostatic discharge (ESD). It is advised that normal static precautions be taken in handling and assembly of this component to prevent damage and/or degradation which may be induced by ESD.

WARNING: WHEN VIEWED UNDER SOME CONDITIONS, THE OPTICAL PORT MAY EXPOSE THE EYE BEYOND THE MAXIMUM PERMISSIBLE EXPOSURE RECOMMENDED IN ANSI Z136.2, 1993. UNDER MOST VIEWING CONDITIONS THERE IS NO EYE HAZARD.

Electrical/ Optical Characteristics 0 to 70°C, unless otherwise stated.

Parameter	Symbol	Min.	Typ.[2]	Max.	Unit	Condition	Note
Transmitter Output Optical Power, 1 mm POF	P_T	-9.5 -10.4	-7.0	-4.8 -4.3	dBm	$I_{F,dc}$ = 20 mA, 25°C 0-70°C	Note 3
Transmitter Output Optical Power, 1 mm POF	P_T	-6.0 -6.9	-3.0	-0.5 -0.0	dBm	$I_{F,dc}$ = 60 mA, 25°C 0-70°C	Note 3
Transmitter Output Optical Power, 200 µm HCS®	P_T	-14.6 -15.5	-13.0	-10.5 -10.0	dBm	$I_{F,dc}$ = 60 mA, 25°C 0-70°C	Note 3
Output Optical Power Temperature Coefficient	$\frac{\Delta P_T}{\Delta T}$		-0.02		dB/ °C		
Peak Emission Wavelength	λ_{PK}	640	650	660	nm		
Peak Wavelength Temperature Coefficient	$\frac{\Delta \lambda}{\Delta T}$		0.12		nm/ °C		
Spectral Width	FWHM		21		nm	Full Width, Half Maximum	
Forward Voltage	V_F	1.8	2.1	2.4	V	I_F = 60 mA	
Forward Voltage Temperature Coefficient	$\frac{\Delta V_F}{\Delta T}$		-1.8		mV/ °C		
Transmitter Numerical Aperture	NA		0.5				
Thermal Resistance, Junction to Case	θ_{jc}		140		°C/ W		Note 4
Reverse Input Breakdown Voltage	V_{BR}	3.0	13		V	$I_{F,dc}$ = -10 µA	
Diode Capacitance	C_O		60		pF	V_F = 0 V, f = 1 MHz	
Unpeaked Optical Rise Time, 10% - 90%	t_r		12		ns	I_F = 60 mA f = 100 kHz	Figure 1 Note 5
Unpeaked Optical Fall Time, 90% - 10%	t_f		9		ns	I_F = 60 mA f = 100 kHz	Figure 1 Note 5

Notes:

1. 1.6 mm below seating plane.
2. Typical data is at 25°C.
3. Optical Power measured at the end of 0.5 meter of 1 mm diameter plastic or 200 µm diameter hard clad silica optical fiber with a large area detector.
4. Typical value measured from junction to PC board solder joint for horizontal mount package, HFBR-1527Z. θ_{jc} is approximately 30°C/ W higher for vertical mount package, HFBR-1537Z.
5. Optical rise and fall times can be reduced with the appropriate driver circuit; refer to Application Note 1066.
6. Pins 5 and 8 are primarily for mounting and retaining purposes, but are electrically connected; pins 3 and 4 are electrically unconnected. It is recommended that pins 3, 4, 5, and 8 all be connected to ground to reduce coupling of electrical noise.
7. Refer to the Versatile Link Family Fiber Optic Cable and Connectors Technical Data Sheet for cable connector options for 1 mm plastic optical fiber and 200 µm HCS fiber.
8. The LED current peaking necessary for high frequency circuit design contributes to electromagnetic interference (EMI). Care must be taken in circuit board layout to minimize emissions for compliance with governmental EMI emissions regulations. Refer to Application Note 1066 for design guidelines.

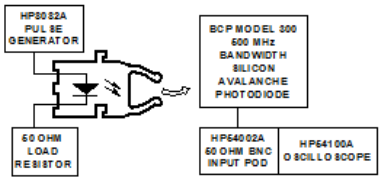


Figure 1. Test circuit for measuring unpeaked rise and fall times.

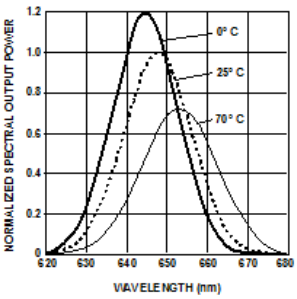


Figure 2. Typical spectra normalized to the 25°C peak.

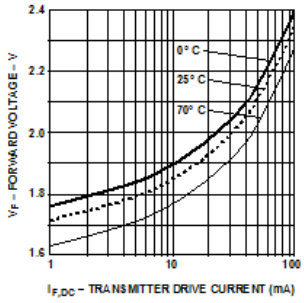


Figure 3. Typical forward voltage vs. drive current.

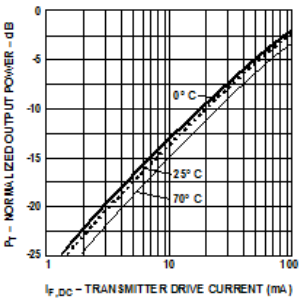


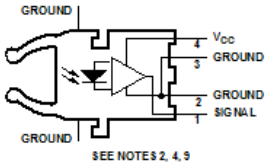
Figure 4. Typical normalized output optical power vs. drive current.

125 Megabaud Versatile Link Receiver

HFBR-25X6Z Series

Description

The HFBR-25X6Z receivers contain a PIN photodiode and transimpedance pre-amplifier circuit in a horizontal (HFBR-2526Z) or vertical (HFBR-2536Z) blue housing, and are designed to interface to 1mm diameter plastic optical fiber or 200 μm hard clad silica glass optical fiber. The receivers convert a received optical signal to an analog output voltage. Follow-on circuitry can optimize link performance for a variety of distance and data rate requirements. Electrical bandwidth greater than 65 MHz allows design of high speed data links with plastic or hard clad silica optical fiber. Refer to Application Note 1066 for details for recommended interface circuits.



Absolute Maximum Ratings

Parameter	Symbol	Min.	Max.	Unit	Reference
Storage Temperature	T_S	-40	+75	°C	
Operating Temperature	T_A	0	+70	°C	
Lead Soldering Temperature			260	°C	Note 1
Cycle Time			10	s	
Signal Pin Voltage	V_O	-0.5	V_{CC}	V	
Supply Voltage	V_{CC}	-0.5	6.0	V	
Output Current	I_O		25	mA	

CAUTION: The small junction sizes inherent to the design of this component increase the component's susceptibility to damage from electrostatic discharge (ESD). It is advised that normal static precautions be taken in handling and assembly of this component to prevent damage and/or degradation which may be induced by ESD.

Electrical/ Optical Characteristics 0 to 70°C; 5.25 V ≥ V_{CC} ≥ 4.75 V; power supply must be filtered (see Figure 1, Note 2).

Parameter	Symbol	Min.	Typ.	Max.	Unit	Test Condition	Note
AC Responsivity 1 mm POF	$R_{P,APF}$	1.7	3.9	6.5	mV/ µW	650 nm	Note 4
AC Responsivity 200 µm HCS	$R_{P,HCS}$	4.5	7.9	11.5	mV/ µW		
RMS Output Noise	V_{NO}		0.46	0.69	mV_{RMS}		Note 5
Equivalent Optical Noise Input Power, RMS - 1 mm POF	$P_{N,RMS}$		- 39	-36	dBm		Note 5
Equivalent Optical Noise Input Power, RMS - 200 µm HCS	$P_{N,RMS}$		-42	-40	dBm		Note 5
Peak Input Optical Power - 1 mm POF	P_R			-5.8 -6.4	dBm dBm	5 ns PWD 2 ns PWD	Note 6
Peak Input Optical Power - 200 µm HCS	P_R			-8.8 -9.4	dBm dBm	5 ns PWD 2 ns PWD	Note 6
Output Impedance	Z_O		30		Ω	50 MHz	Note 4
DC Output Voltage	V_O	0.8	1.8	2.6	V	P_R = 0 µW	
Supply Current	I_{CC}		9	15	mA		
Electrical Bandwidth	BW_E	65	125		MHz	-3 dB electrical	
Bandwidth * Rise Time			0.41		Hz * s		
Electrical Rise Time, 10-90%	t_r		3.3	6.3	ns	P_R = -10 dBm peak	
Electrical Fall Time, 90-10%	t_f		3.3	6.3	ns	P_R = -10 dBm peak	
Pulse Width Distortion	PWD		0.4	1.0	ns	P_R = -10 dBm peak	Note 7
Overshoot			4		%	P_R = -10 dBm peak	Note 8

Notes:

1. 1.6 mm below seating plane.
2. The signal output is an emitter follower, which does not reject noise in the power supply. The power supply must be filtered as in Figure 1.
3. Typical data are at 25°C and V_{CC} = +5 Vdc.
4. Pin 1 should be ac coupled to a load ≥ 510 Ω with load capacitance less than 5 pF.
5. Measured with a 3 pole Bessel filter with a 75 MHz, -3dB bandwidth.
6. The maximum Peak Input Optical Power is the level at which the Pulse Width Distortion is guaranteed to be less than the PWD listed under Test Condition. $P_{R,Max}$ is given for PWD = 5 ns for designing links at ≤ 50 MBd operation, and also for PWD = 2 ns for designing links up to 125 MBd (for both POF and HCS input conditions).
7. 10 ns pulse width, 50% duty cycle, at the 50% amplitude point of the waveform.
8. Percent overshoot is defined at:

$$\frac{(V_{PK} - V_{100\%})}{V_{100\%}} \times 100\%$$

9. Pins 5 and 8 are primarily for mounting and retaining purposes, but are electrically connected. It is recommended that these pins be connected to ground to reduce coupling of electrical noise.
10. If there is no input optical power to the receiver (no transmitted signal) electrical noise can result in false triggering of the receiver. In typical applications, data encoding and error detection prevent random triggering from being interpreted as valid data. Refer to Application Note 1066 for design guidelines.

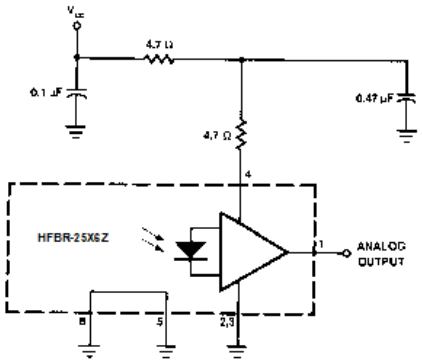


Figure 1. Recommended power supply filter circuit.

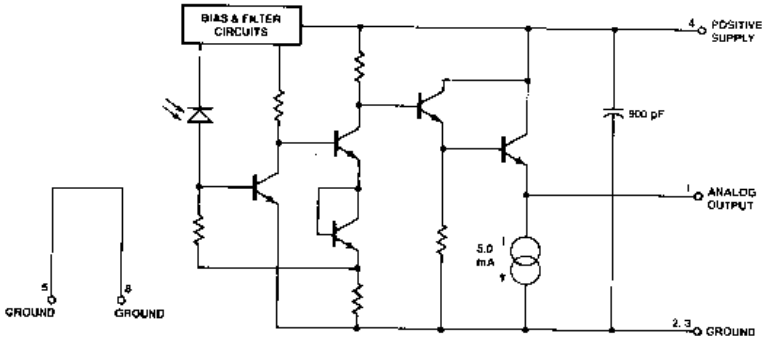


Figure 2. Simplified receiver schematic.

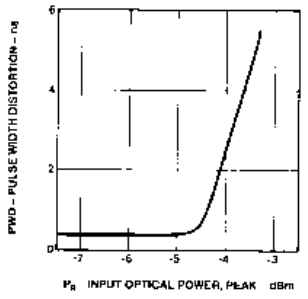


Figure 3. Typical pulse width distortion vs. peak input power.

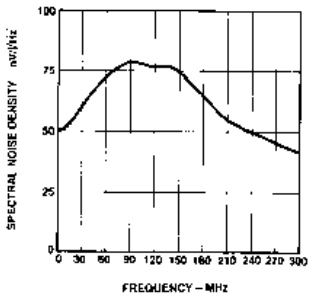


Figure 4. Typical output spectral noise density vs. frequency.

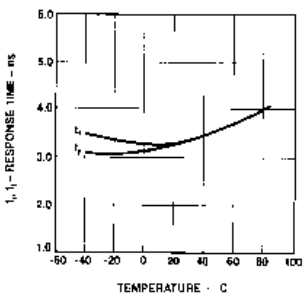
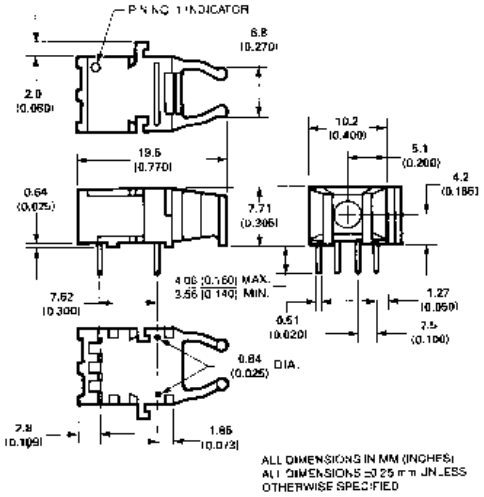


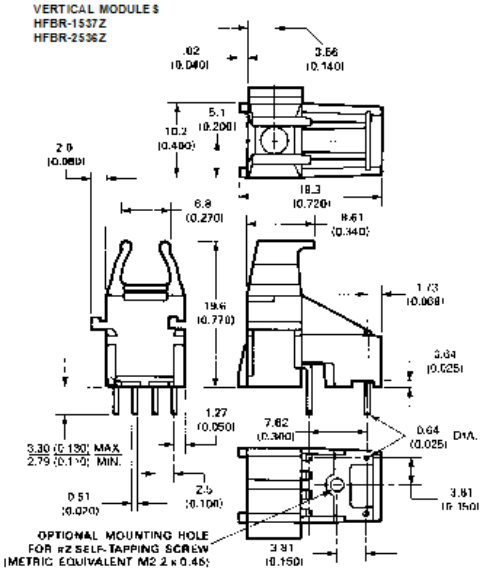
Figure 5. Typical rise and fall time vs. temperature.

Versatile Link Mechanical Dimensions

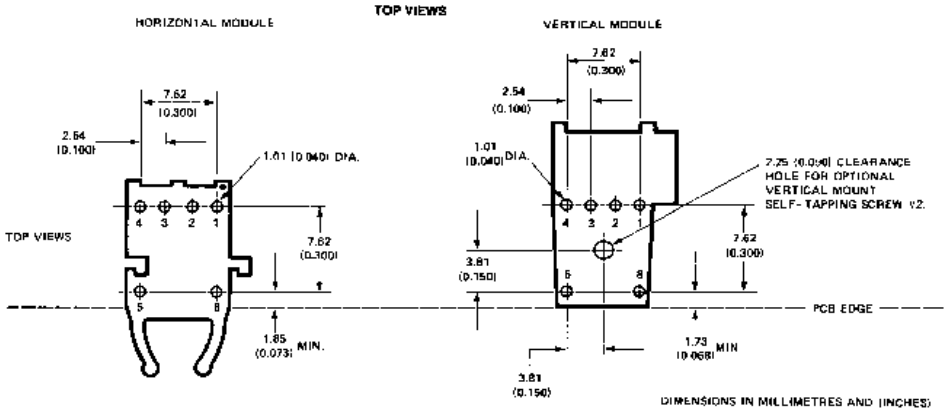
HORIZONTAL MODULES
HFBR-1527Z
HFBR-2526Z



VERTICAL MODULES
HFBR-1537Z
HFBR-2536Z



Versatile Link Printed Circuit Board Layout Dimensions



ELECTRICAL PIN FUNCTIONS

PIN NO.	TRANSMITTERS HFBR-16x7Z	RECEIVERS HFBR-26x8Z
1	ANODE	SIGNAL
2	CATHODE	GROUND
3	GROUND*	GROUND
4	GROUND*	V_{CC} (+5 V)
5	GROUND**	GROUND**
6	GROUND**	GROUND**

*NO INTERNAL CONNECTION.
**PINS 5 AND 6 CONNECTED INTERNALLY TO EACH OTHER ONLY.

For product information and a complete list of distributors, please go to our website: www.avagotech.com

5989-4631EN September 11, 2006

10.6.3. Comparator data sheet.

19-0229; Rev 4; 9/01

MAXIM

High-Speed, Low-Power, 3V/5V, Rail-to-Rail, Single-Supply Comparators

General Description

The MAX941/MAX942/MAX944 are single/dual/quad high-speed comparators optimized for systems powered from a 3V or 5V supply. These devices combine high speed, low power, and Rail-to-Rail® inputs. Propagation delay is 80ns, while supply current is only 350µA per comparator.

The input common-mode range of the MAX941/MAX942/MAX944 extends beyond both power-supply rails. The outputs pull to within 0.4V of either supply rail without external pullup circuitry, making these devices ideal for interface with both CMOS and TTL logic. All input and output pins can tolerate a continuous short-circuit fault condition to either rail.

Internal hysteresis ensures clean output switching, even with slow-moving input signals. The MAX941 features latch enable and device shutdown.

The single MAX941 and dual MAX942 are offered in a tiny µMAX package. Both the single and dual MAX942 are available in 8-pin DIP and SO packages. The quad MAX944 comes in 14-pin DIP and narrow SO packages.

Applications

3V/5V Systems
Battery-Powered Systems
Threshold Detectors/Discriminators
Line Receivers
Zero-Crossing Detectors
Sampling Circuits

Features

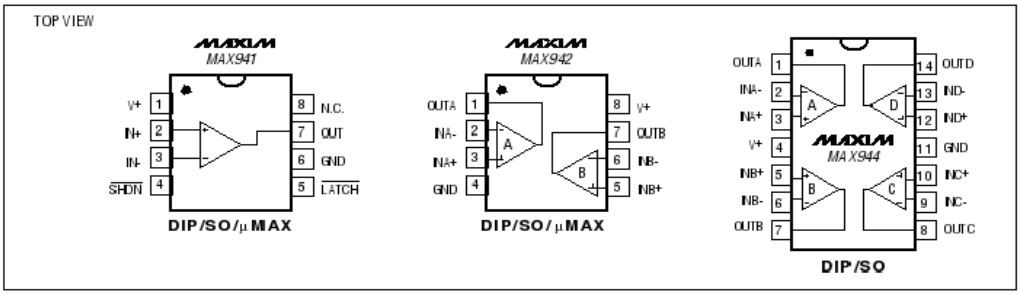
- **Available in µMAX Package**
- **Optimized for 3V and 5V Applications (operation down to 2.7V)**
- **Fast, 80ns Propagation Delay (5mV overdrive)**
- **Rail-to-Rail Input Voltage Range**
- **Low Power:**
  1mW Power Dissipation per Comparator (3V)
  350µA Supply Current
- **Low, 1mV Offset Voltage**
- **Internal Hysteresis for Clean Switching**
- **Outputs Swing 200mV of Power Rails**
- **CMOS/TTL-Compatible Outputs**
- **Output Latch (MAX941 only)**
- **Shutdown Function (MAX941 only)**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX941CPA	0°C to +70°C	8 Plastic DIP
MAX941CSA	0°C to +70°C	8 SO
MAX941EPA	-40°C to +85°C	8 Plastic DIP
MAX941ESA	-40°C to +85°C	8 SO
MAX941EUA	-40°C to +85°C	8 µMAX

Ordering Information continued at end of data sheet.

Pin Configurations



Rail-to-Rail is a registered trademark of Nippon Motorola Ltd.

High-Speed, Low-Power, 3V/5V, Rail-to-Rail, Single-Supply Comparators

ABSOLUTE MAXIMUM RATINGS

Power-Supply Ranges
Supply Voltage V+ to GND .....+6.5V
Differential Input Voltage .....-0.3V to (V+ + 0.3V)
Common-Mode Input Voltage .....-0.3V to (V+ + 0.3V)
LATCH Input (MAX941 only) .....-0.3V to (V+ + 0.3V)
SHDN Control Input (MAX941 only) .....-0.3V to (V+ + 0.3V)
Continuous Power Dissipation (T_A = +70°C)
8-Pin Plastic DIP (derate 9.09mW/°C above +70°C) ...727mW
8-Pin SO (derate 5.88mW/°C above +70°C).....471mW
8-Pin µMAX (derate 4.1mW/°C above +70°C) 330mW
14-Pin Plastic DIP (derate 10.00mW/°C above +70°C)..800mW
14-Pin SO (derate 8.33mW/°C above +70°C).....667mW
Operating Temperature Ranges
MAX94_C_ _ .....0°C to +70°C
MAX94_E_ _ .....-40°C to +85°C
Storage Temperature Range .....-65°C to +150°C
Lead Temperature (soldering, 10s) .....+300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V+ = 2.7V to 6.0V, $T_A = T_{MIN}$ to T_{MAX}, unless otherwise noted. Typical values are at T_A = +25°C.) (Note 14)

PARAMETER	SYMBOL	CONDITIONS			MIN	TYP	MAX	UNITS
Positive Supply Voltage	V+				2.7		6.0	V
Input Voltage Range	V_{CMR}	(Note 1)			-0.2		V+ + 0.2	V
Input-Referred Trip Points	V_{TRIP}	V_{CM} = 0 or V_{CM} = V+ (Note 2)	T_A = +25°C	MAX94_C_ _, MAX94_EP_, MAX94_ES_		1	3	mV
				MAX941EUA/MAX942EUA		1	4	
			$T_A = T_{MIN}$ to T_{MAX}	MAX94_C_ _, MAX94_EP_, MAX94_ES_			4	mV
				MAX941EUA/MAX942EUA			6	
Input Offset Voltage	V_{OS}	V_{CM} = 0 or V_{CM} = V+ (Note 3)	T_A = +25°C	MAX94_C_ _, MAX94_EP_, MAX94_ES_		1	2	mV
				MAX941EUA/MAX942EUA		1	3	
			$T_A = T_{MIN}$ to T_{MAX}	MAX94_C_ _, MAX94_EP_, MAX94_ES_			3	mV
				MAX941EUA/MAX942EUA			5.5	
Input Bias Current	I_B	V_{IN} = V_{OS}, V_{CM} = 0 or V_{CM} = V+ (Note 4)		MAX94_C		150	300	nA
				MAX94_E		150	400	
Input Offset Current	I_{OS}	V_{IN} = V_{OS}, V_{CM} = 0 or V+				10	100	nA
Common-Mode Rejection Ratio	CMRR	(Note 5)		MAX94_C_ _, MAX94_EP_, MAX94_ES_		80	300	µV/V
				MAX941EUA/MAX942EUA		80	800	
Power-Supply Rejection Ratio	PSRR	2.7V ≤ V+ ≤ 6.0V, V_{CM} = 0		MAX94_C_ _, MAX94_EP_, MAX94_ES_		80	300	µV/V
				MAX941EUA/MAX942EUA		80	350	
Output High Voltage	V_{OH}	I_{SOURCE} = 400µA			V+ - 0.4	V+ - 0.2		V
		I_{SOURCE} = 4mA			V+ - 0.4	V+ - 0.3		
Output Low Voltage	V_{OL}	I_{SINK} = 400µA				0.2	0.4	V
		I_{SINK} = 4mA				0.3	0.4	
Output Leakage Current	I_{LEAK}	(Note 6)					1	µA

ELECTRICAL CHARACTERISTICS (continued)

(V+ = 2.7V to 6.0V, $T_A = T_{MIN}$ to T_{MAX}, unless otherwise noted. Typical values are at T_A = +25°C.) (Note 14)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Current per Comparator	I_{CC}	V+ = 3V	MAX941		380	600	µA
			MAX942/MAX944		350	500	
		V+ = 5V	MAX941		430	700	
			MAX942/MAX944		400	600	
		MAX941 only, shutdown mode (V+ = 3V)			12	60	
Power Dissipation per Comparator	PD	(Note 7)	MAX941		1.0	4.2	mW
			MAX942/MAX944		1.0	3.6	
Propagation Delay	t_{PD+}, t_{PD-}	(Note 8)	MAX94_C		80	150	ns
			MAX94_E		80	200	
Differential Propagation Delay	dt_{PD}	(Note 9)			10		ns
Propagation Delay Skew		(Note 10)			10		ns
Logic Input Voltage High	V_{IH}	(Note 11)		$\frac{V+}{2}$ + 0.4	$\frac{V+}{2}$		V
Logic Input Voltage Low	V_{IL}	(Note 11)			$\frac{V+}{2}$	$\frac{V+}{2}$ - 0.4	V
Logic Input Current	I_{IL}, I_{IH}	V_{LOGIC} = 0 or V+ (Note 11)			2	10	µA
Data-to-Latch Setup Time	t_S	(Note 12)			20		ns
Latch-to-Data Hold Time	t_H	(Note 12)			30		ns
Latch Pulse Width	t_{LPW}	MAX941 only			50		ns
Latch Propagation Delay	t_{LPD}	MAX941 only			70		ns
Shutdown Time		(Note 13)			3		µs
Shutdown Disable Time		(Note 13)			10		µs

Note 1: Inferred from the CMRR test. Note also that either or both inputs can be driven to the absolute maximum limit (0.3V beyond either supply rail) without damage or false output inversion.

Note 2: The input-referred trip points are the extremities of the differential input voltage required to make the comparator output change state. The difference between the upper and lower trip points is equal to the width of the input-referred hysteresis zone (see Figure 1).

Note 3: V_{OS} is defined as the center of the input-referred hysteresis zone (see Figure 1).

Note 4: The polarity of I_B reverses direction as V_{CM} approaches either supply rail. See *Typical Operating Characteristics* for more detail.

Note 5: Specified over the full common-mode range (V_{CMR}).

Note 6: Applies to the MAX941 only when in shutdown mode. Specification is for current flowing into or out of the output pin for V_{OUT} driven to any voltage from V+ to GND.

Note 7: Typical power dissipation specified with V+ = 3V; maximum with V+ = 6V.

Note 8: Parameter is guaranteed by design and specified with V_{OD} = 5mV and C_{LOAD} = 15pF in parallel with 400µA of sink or source current. V_{OS} is added to the overdrive voltage for low values of overdrive (see Figure 2).

Note 9: Specified between any two channels in the MAX942/MAX944.

Note 10: Specified as the difference between t_{PD+} and t_{PD-} for any one comparator.

Note 11: Applies to the MAX941 only for both SHDN and LATCH pins.

Note 12: Applies to the MAX941 only. Comparator is active with LATCH pin driven high and is latched with $\overline{LATCH}$ pin driven low (see Figure 2).

Note 13: Applicable to the MAX941 only. Comparator is active with $\overline{SHDN}$ pin driven high and is in shutdown with $\overline{SHDN}$ pin driven low. Shutdown disable time is the delay when $\overline{SHDN}$ is driven high to the time the output is valid.

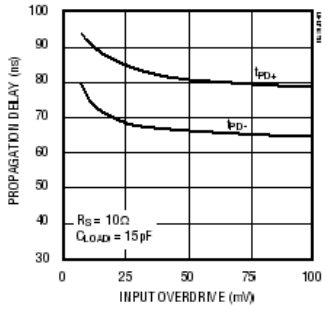
Note 14: The MAX941EUA and MAX942EUA are 100% production tested at T_A = +25°C. Specifications over temperature are guaranteed by design.

High-Speed, Low-Power, 3V/5V, Rail-to-Rail, Single-Supply Comparators

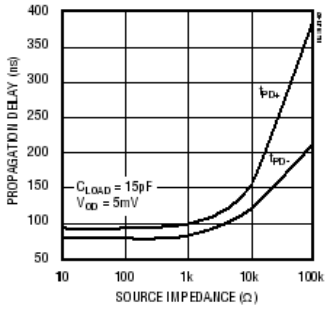
Typical Operating Characteristics

(V+ = 3.0V, T_A = +25°C, unless otherwise noted.)

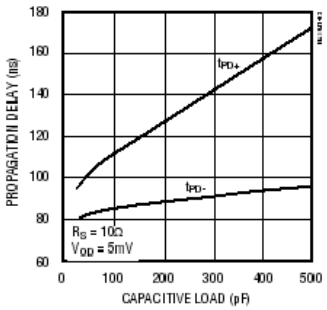
PROPAGATION DELAY vs. INPUT OVERDRIVE



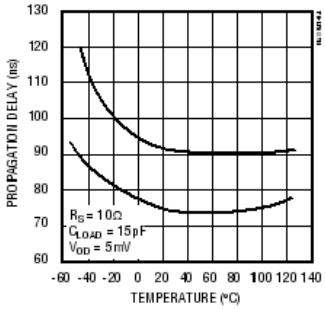
PROPAGATION DELAY vs. SOURCE IMPEDANCE



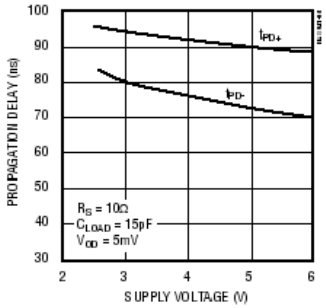
PROPAGATION DELAY vs. CAPACITIVE LOAD



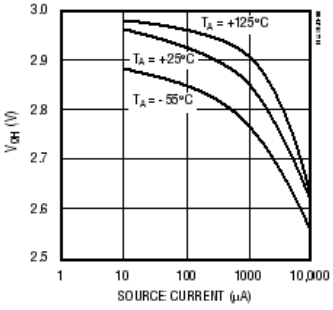
PROPAGATION DELAY vs. TEMPERATURE



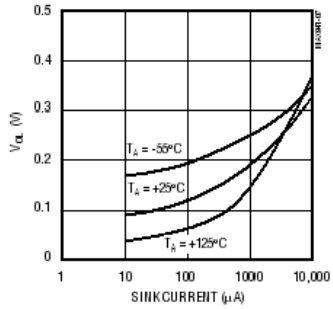
PROPAGATION DELAY vs. SUPPLY VOLTAGE



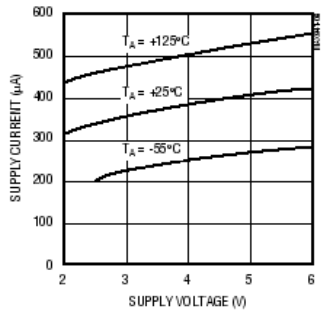
OUTPUT HIGH VOLTAGE vs. SOURCE CURRENT



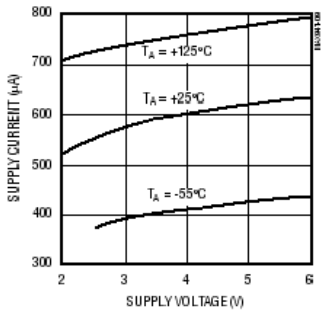
OUTPUT LOW VOLTAGE vs. SINK CURRENT



MAX941 TOTAL SUPPLY CURRENT vs. SUPPLY VOLTAGE



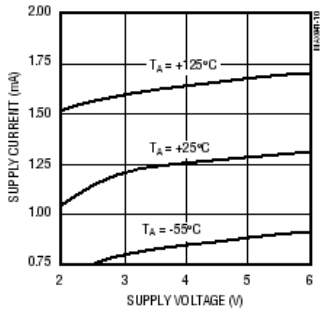
MAX942 TOTAL SUPPLY CURRENT vs. SUPPLY VOLTAGE



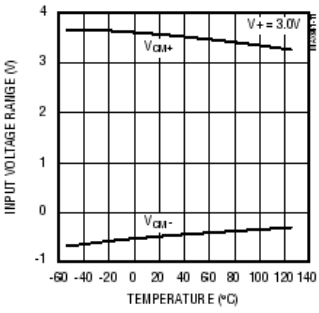
Typical Operating Characteristics (continued)

(V+ = 3.0V, T_A = +25°C, unless otherwise noted.)

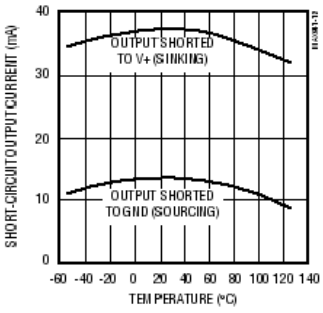
MAX944 TOTAL SUPPLY CURRENT vs. SUPPLY VOLTAGE



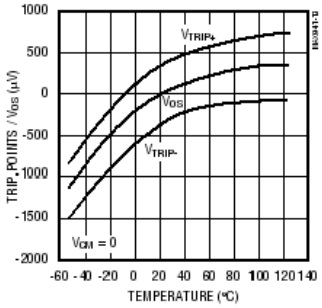
INPUT VOLTAGE RANGE vs. TEMPERATURE



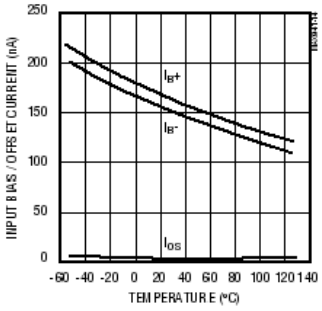
SHORT-CIRCUIT OUTPUT CURRENT vs. TEMPERATURE



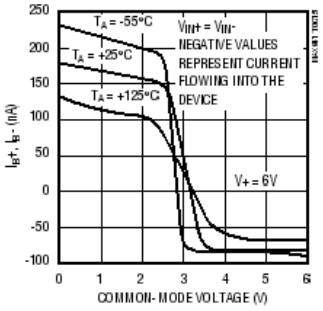
VOLTAGE TRIP POINTS/INPUT OFFSET VOLTAGE vs. TEMPERATURE



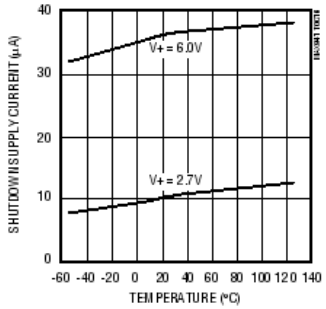
INPUT BIAS CURRENT/INPUT OFFSET CURRENT vs. TEMPERATURE



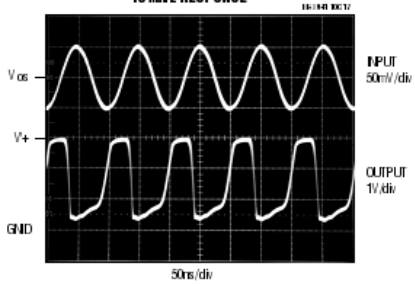
INPUT BIAS CURRENT (I_{B+}, I_{B-}) vs. COMMON-MODE VOLTAGE



MAX941 SHUTDOWN SUPPLY CURRENT vs. TEMPERATURE



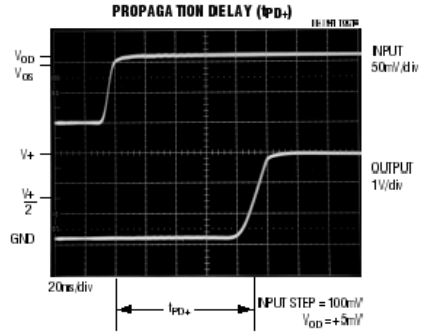
10MHz RESPONSE

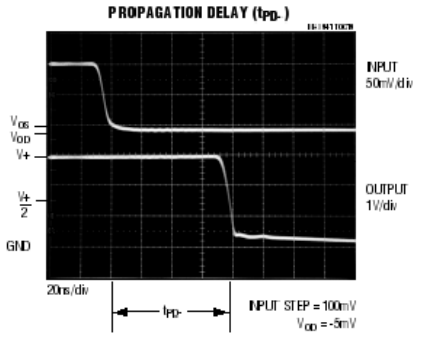


High-Speed, Low-Power, 3V/5V, Rail-to-Rail, Single-Supply Comparators

Typical Operating Characteristics (continued)

(V+ = 3.0V, T_A = +25°C, unless otherwise noted.)





Pin Description

PIN			NAME	FUNCTION
MAX941	MAX942	MAX944		
—	1	1	OUTA	Comparator A Output
—	2	2	INA-	Comparator A Inverting Input
—	3	3	INA+	Comparator A Noninverting Input
1	8	4	V+	Positive Supply (V+ to GND must be ≤ 6.5V)
—	5	5	INB+	Comparator B Noninverting Input
—	6	6	INB-	Comparator B Inverting Input
—	7	7	OUTB	Comparator B Output
—	—	8	OUTC	Comparator C Output
—	—	9	INC-	Comparator C Inverting Input
—	—	10	INC+	Comparator C Noninverting Input
6	4	11	GND	Ground
—	—	12	IND+	Comparator D Noninverting Input
—	—	13	IND-	Comparator D Inverting Input
—	—	14	OUTD	Comparator D Output
2	—	—	IN+	Noninverting Input
3	—	—	IN-	Inverting Input
4	—	—	$\overline{SHDN}$	Shutdown: MAX941 is active when $\overline{SHDN}$ is driven high; MAX941 is in shutdown when $\overline{SHDN}$ is driven low.
5	—	—	$\overline{LATCH}$	The output is latched when $\overline{LATCH}$ is low. The latch is transparent when $\overline{LATCH}$ is high.
7	—	—	OUT	Comparator Output
8	—	—	N.C.	No Connection. Not internally connected.

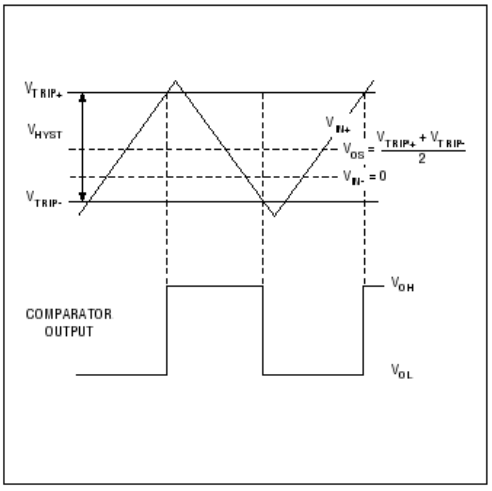


Figure 1. Input and Output Waveform, Noninverting Input Varied

Detailed Description

The MAX941/MAX942/MAX944 single-supply comparators feature internal hysteresis, high speed, and low power. Their outputs are guaranteed to pull within 0.4V of either supply rail without external pullup or pulldown circuitry. Rail-to-rail input voltage range and low-voltage single-supply operation make these devices ideal for portable equipment. The MAX941/MAX942/MAX944 interface directly to CMOS and TTL logic.

Timing

Most high-speed comparators oscillate in the linear region because of noise or undesired parasitic feedback. This tends to occur when the voltage on one input is at or equal to the voltage on the other input. To counter the parasitic effects and noise, the MAX941/MAX942/MAX944 have internal hysteresis.

The hysteresis in a comparator creates two trip points: one for the rising input voltage and one for the falling input voltage (Figure 1). The difference between the trip points is the hysteresis. When the comparator's input voltages are equal, the hysteresis effectively causes one comparator input voltage to move quickly past the other, thus taking the input out of the region where oscillation occurs. Standard comparators require hysteresis to be added with external resistors. The MAX941/MAX942/MAX944's fixed internal hysteresis eliminates these resistors and the equations needed to determine appropriate values.

Figure 1 illustrates the case where IN- is fixed and IN+ is varied. If the inputs were reversed, the figure would look the same, except the output would be inverted.

The MAX941 includes an internal latch that allows storage of comparison results. The $\overline{LATCH}$ pin has a high input impedance. If $\overline{LATCH}$ is high, the latch is transparent (i.e., the comparator operates as though the latch is not present). The comparator's output state is stored when $\overline{LATCH}$ is pulled low. All timing constraints must be met when using the latch function (Figure 2).

Shutdown Mode (MAX941 Only)

The MAX941 shuts down when $\overline{SHDN}$ is low. When shutdown, the supply current drops to less than 60µA, and the three-state output becomes high impedance. The $\overline{SHDN}$ pin has a high input impedance. Connect $\overline{SHDN}$ to V+ for normal operation. Exit shutdown with $\overline{LATCH}$ high; otherwise, the output will be indeterminate.

Input Stage Circuitry

The MAX941/MAX942/MAX944 include internal protection circuitry that prevents damage to the precision input stage from large differential input voltages. This protection circuitry consists of two back-to-back diodes between IN+ and IN- as well as two 4.1kΩ resistors (Figure 3). The diodes limit the differential voltage applied to the internal circuitry of the comparators to be no more than $2V_F$, where V_F is the forward voltage drop of the diode (about 0.7V at +25°C).

For a large differential input voltage (exceeding $2V_F$), this protection circuitry increases the input bias current at IN+ (source) and IN- (sink).

$$\text{Input Current} = \frac{(IN+ - IN-) - 2V_F}{2 \times 4.1k\Omega}$$

Input current with large differential input voltages should not be confused with input bias current (I_B). As long as the differential input voltage is less than $2V_F$, this input current is equal to I_B. The protection circuitry also allows for the input common-mode range of the MAX941/MAX942/MAX944 to extend beyond both power-supply rails. The output is in the correct logic state if one or both inputs are within the common-mode range.

High-Speed, Low-Power, 3V/5V, Rail-to-Rail, Single-Supply Comparators

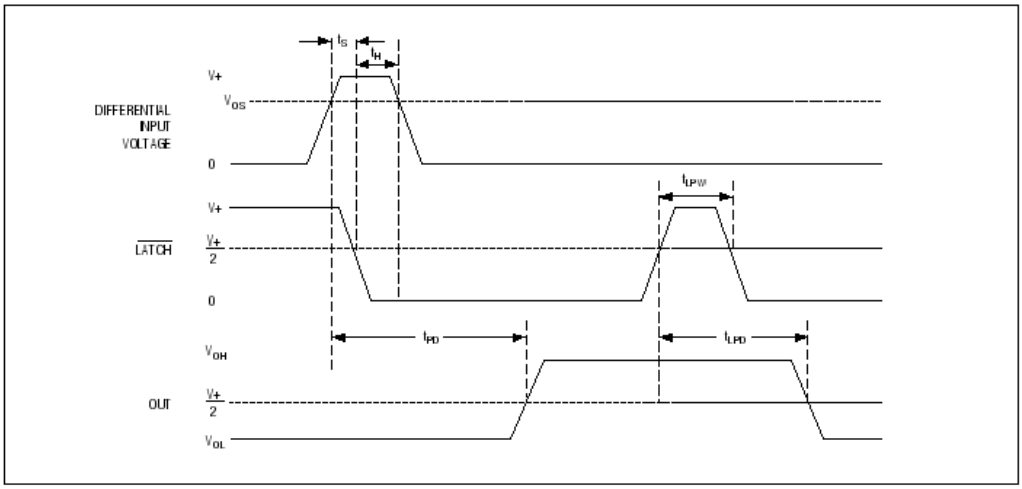


Figure 2. MAX941 Timing Diagram with Latch Operator

Output Stage Circuitry

The MAX941/MAX942/MAX944 contain a current-driven output stage as shown in Figure 4. During an output transition, I_{SOURCE} or I_{SINK} is pushed or pulled to the output pin. The output source or sink current is high during the transition, creating a rapid slew rate. Once the output voltage reaches V_{OH} or V_{OL}, the source or sink current decreases to a small value, capable of maintaining the V_{OH} or V_{OL} static condition. This significant decrease in current conserves power after an output transition has occurred.

One consequence of a current-driven output stage is a linear dependence between the slew rate and the load capacitance. A heavy capacitive load will slow down a voltage output transition. This can be useful in noise-sensitive applications where fast edges may cause interference.

Applications Information

Circuit Layout and Bypassing

The high gain bandwidth of the MAX941/MAX942/MAX944 requires design precautions to realize the comparators' full high-speed capability. The recommended precautions are:

1) Use a printed circuit board with a good, unbroken, low-inductance ground plane.
2) Place a decoupling capacitor (a 0.1µF ceramic capacitor is a good choice) as close to V+ as possible.
3) Pay close attention to the decoupling capacitor's bandwidth, keeping leads short.
4) On the inputs and outputs, keep lead lengths short to avoid unwanted parasitic feedback around the comparators.
5) Solder the device directly to the printed circuit board instead of using a socket.

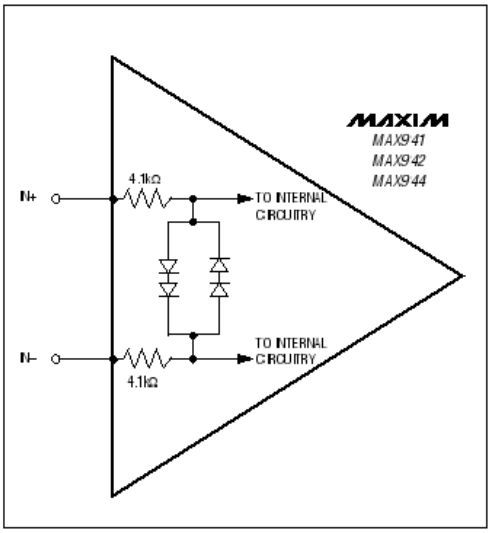


Figure 3. Input Stage Circuitry

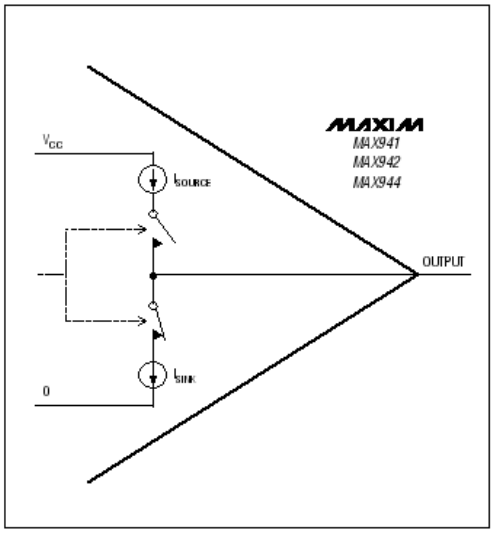


Figure 4. Output Stage Circuitry

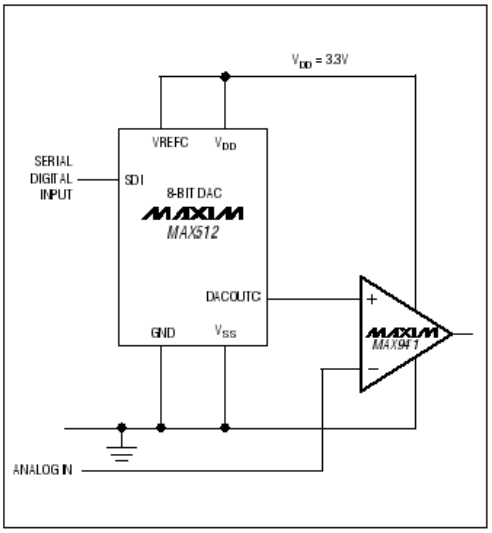


Figure 5. 3.3V Digitally Controlled Threshold Detector

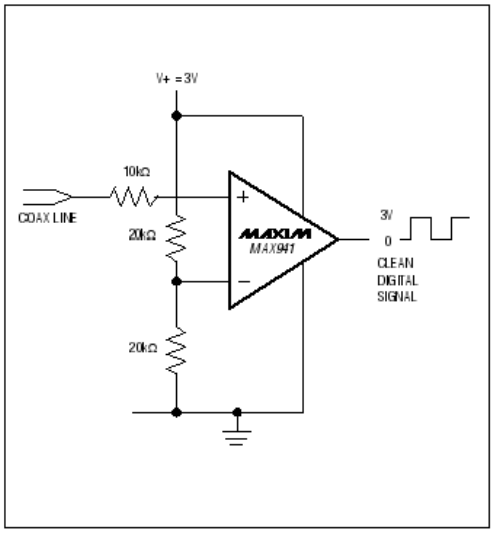


Figure 6. Line Transceiver Application

High-Speed, Low-Power, 3V/5V, Rail-to-Rail, Single-Supply Comparators

Ordering Information (continued)

PART	TEMP RANGE	PIN-PACKAGE
MAX942CPA	0°C to +70°C	8 Plastic DIP
MAX942CSA	0°C to +70°C	8 SO
MAX942EPA	-40°C to +85°C	8 Plastic DIP
MAX942ESA	-40°C to +85°C	8 SO
MAX942EUA	-40°C to +85°C	8 µMAX
MAX944CPD	0°C to +70°C	14 Plastic DIP
MAX944CSD	0°C to +70°C	14 SO
MAX944EPD	-40°C to +85°C	14 Plastic DIP
MAX944ESD	-40°C to +85°C	14 SO

Chip Information

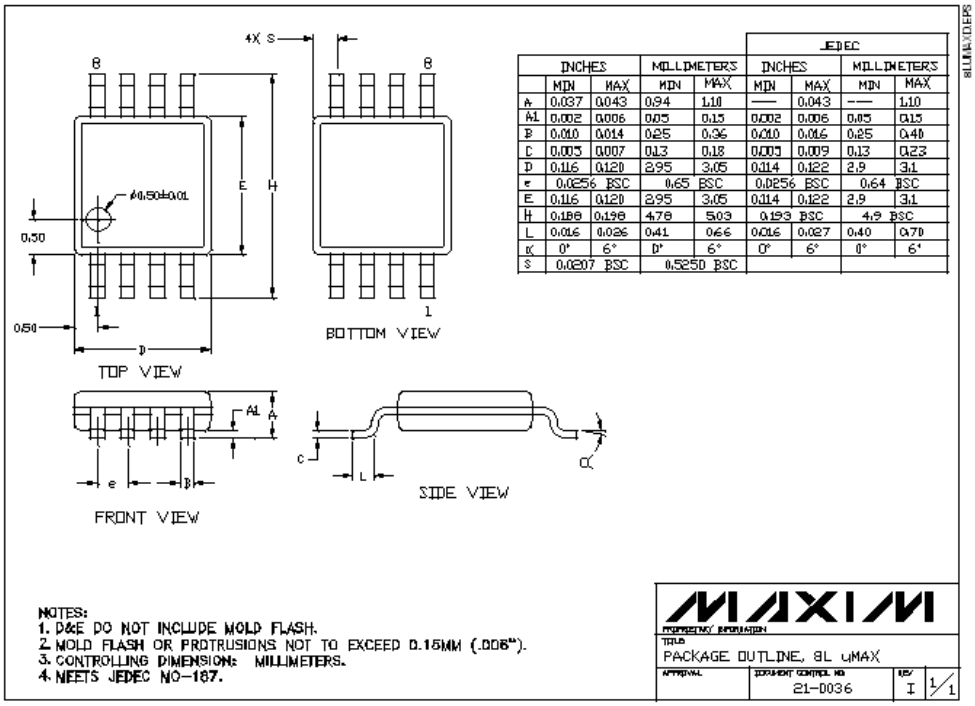
MAX941 TRANSISTOR COUNT: 192

MAX942 TRANSISTOR COUNT: 314

MAX944 TRANSISTOR COUNT: 620

PROCESS: BiPolar

Package Information



	INCHES		MILLIMETERS		JEDEC INCHES		JEDEC MILLIMETERS	
	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX
A	0.037	0.043	0.94	1.10	—	0.043	—	1.10
A1	0.002	0.006	0.05	0.15	0.002	0.006	0.05	0.15
B	0.010	0.014	0.25	0.36	0.010	0.016	0.25	0.40
C	0.005	0.007	0.13	0.18	0.005	0.009	0.13	0.23
D	0.116	0.120	2.95	3.05	0.114	0.122	2.9	3.1
e	0.0256 BSC		0.65 BSC		0.0256 BSC		0.64 BSC	
E	0.116	0.120	2.95	3.05	0.114	0.122	2.9	3.1
H	0.188	0.198	4.78	5.03	0.193 BSC		4.9 BSC	
L	0.016	0.026	0.41	0.66	0.016	0.027	0.40	0.70
α	0°	6°	0°	6°	0°	6°	0°	6°
S	0.0207 BSC		0.5250 BSC					

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH.
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED 0.15MM (.006").
3. CONTROLLING DIMENSION: MILLIMETERS.
4. MEETS JEDEC MO-187.

MAXIM

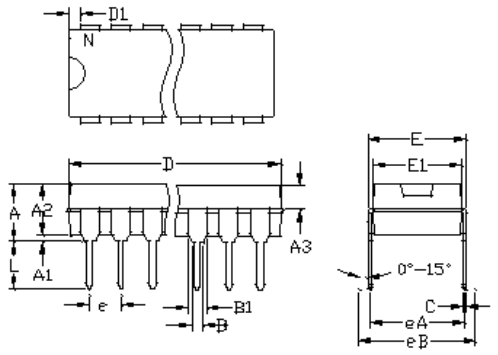
PROPRIETARY INFORMATION

TITLE: PACKAGE OUTLINE, 8L µMAX

APPROVAL | DOCUMENT CONTROL NO. 21-0036 | REV I | 1/1

Package Information (continued)

PDIPN.EPS



	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	---	0.180	---	4.572
A1	0.020	---	0.508	---
A2	0.125	0.175	3.18	4.45
A3	0.055	0.080	1.40	2.03
B	0.015	0.021	0.381	0.533
B1	0.045	0.060	1.14	1.524
C	0.009	0.014	0.229	0.355
D1	0.005	0.080	0.13	2.03
E	0.300	0.325	7.62	8.255
E1	0.275	0.295	6.985	7.493
e	0.100	---	2.54	---
eA	0.300	---	7.62	---
eB	---	0.400	---	10.16
L	0.115	0.150	2.921	3.81

	INCHES		MILLIMETERS			
	MIN	MAX	MIN	MAX	N	MS001
D	0.348	0.390	8.84	9.91	8	AB
D	0.735	0.765	18.67	19.43	14	AC
D	0.745	0.765	18.92	19.43	16	AA
D	0.885	0.915	22.48	23.24	18	AD
D	1.015	1.045	25.78	26.54	20	AE
D	1.14	1.265	28.96	32.13	24	AF
D	1.360	1.380	34.54	35.05	28	*5

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH.
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED .15MM (.006").
3. CONTROLLING DIMENSION: MILLIMETER.
4. MEETS JEDEC MS001-XX AS SHOWN IN ABOVE TABLE.
5. SIMILIAR TO JEDEC MS-095-AH.
6. N = NUMBER OF PINS.

MAXIM | PACKAGE FAMILY OUTLINE: PDIP .300" | 1/1 | 21-0043 B

10.6.4. POF data sheet.

HFBR-RXXYYYZ Series (POF)
HFBR-EXXYYYZ Series (POF)

Plastic Optical Fiber Cable and Accessories for Versatile Link

Data Sheet

Cable Description

The HFBR-R/EXXYYYZ series of plastic fiber optic cables are constructed of a single step-index fiber sheathed in a black polyethylene jacket. The duplex fiber consists of two simplex fibers joined with a zipcord web.

Standard attenuation and extra low loss POF cables are identical except for attenuation specifications. Polyethylene jackets on all plastic fiber cables comply with UL VW-1 flame retardant specification (UL file # E89328).

Cables are available in unconnectored or connectored options. Refer to the Ordering Guide for part number information.

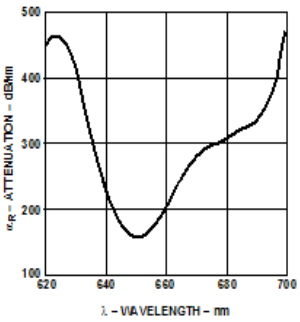


Figure 1. Typical POF attenuation vs. wavelength.

Features

- Compatible with Avago Versatile Link family of connectors and fiber optic components
- 1 mm diameter Plastic Optical Fiber (POF) in two grades: low cost standard POF with 0.22 dB/ m typical attenuation, or high performance extra low loss POF with 0.19 dB/ m typical attenuation

Applications

- Industrial data links for factory automation and plant control
- Intra-system links; board-to-board, rack-to-rack
- Telecommunications switching systems
- Computer-to-peripheral data links, PC bus extension
- Proprietary LANs
- Digitized video
- Medical instruments
- Reduction of lightning and voltage transient susceptibility
- High voltage isolation

Plastic Optical Fiber Specifications: HFBR-R/ EXXYYYZ

Absolute Maximum Ratings

Parameter		Symbol	Min.	Max.	Unit	Note
Storage and Operating Temperature		$T_{S,O}$	-55	+85	°C	
Recommended Operating Temperature		T_O	-40	+85	°C	
Installation Temperature		T_I	-20	+70	°C	1
Short Term Tensile Force	Single Channel	F_T		50	N	2
	Dual Channel	F_T		100	N	
Short Term Bend Radius		r	25		mm	3, 4
Long Term Bend Radius		r	35		mm	
Long Term Tensile Load		F_T		1	N	
Flexing				1000	Cycles	4

Mechanical/ Optical Characteristics, T_A = -40 to +85°C unless otherwise specified.

Parameter		Symbol	Min.	Typ.[5]	Max.	Unit	Condition
Cable Attenuation	Standard Cable, Type "R"	α_O	0.15	0.22	0.27	dB/ m	Source is HFBR-15XXZ (660 mm LED, 0.5 NA) ℓ = 50 meters
	Extra Low Loss, Type "E"		0.15	0.19	0.23		
Reference Attenuation	Standard Cable, Type "R"	α_R	0.12	0.19	0.24	dB/ m	Source is 650 nm, 0.5 NA monochrometer, ℓ = 50 meters Note 7, Figure 1
	Extra Low Loss, Type "E"		0.12	0.16	0.19		
Numerical Aperture		NA	0.46	0.47	0.50		>2 meters
Diameter, Core and Cladding		D_C	0.94	1.00	1.06	mm	
Diameter, Jacket		D_J	2.13	2.20	2.27	mm	Simplex Cable
Propagation Delay Constant		l/ v		5.0		ns/ m	Note 6
Mass per Unit Length/ Channel				5.3		g/ m	Without Connectors
Cable Leakage Current		I_L		12		nA	50 kV, ℓ = 0.3 meters
Refractive Index	Core	n		1.492			
	Cladding			1.417			

Notes:

1. Installation temperature is the range over which the cable can be bent and pulled without damage. Below -20°C the cable becomes brittle and should not be subjected to mechanical stress.
2. Short Term Tensile Force is for less than 30 minutes.
3. Short Term Bend Radius is for less than 1 hour nonoperating.
4. 90° bend on 25 mm radius mandrel. Bend radius is the radius of the mandrel around which the cable is bent.
5. Typical data are at 25°C.
6. Propagation delay constant is the reciprocal of the group velocity for propagation delay of optical power. Group velocity is v=c/ n where c is the velocity of light in free space ($3x10^8$ m/ s) and n is the effective core index of refraction.
7. Note that α_R rises at the rate of about 0.0067 dB/ °C, where the thermal rise refers to the LED temperature changes above 25°C. Please refer to Figure 1 which shows the typical plastic optical fiber attenuation versus wavelength at 25°C.

Plastic Fiber Connector Styles

Connector Description

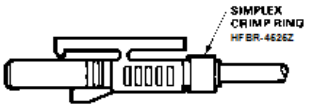
Four connector styles are available for termination of plastic optical fiber: simplex, simplex latching, duplex and duplex latching. All connectors provide a snap-in action when mated to Versatile Link components. Simplex connectors are color coded to facilitate identification of transmitter and receiver connections. Duplex connectors are keyed so that proper orientation is ensured during insertion. If the POF cable/connector will be used at extreme operating temperatures or experience frequent and wide temperature cycling effects, the cable/connector attachment can be strengthened with an RTV adhesive (see Plastic Connectoring Instructions for more detail). The connectors are made of a flame retardant VALOX UL94 V-0 material (UL file # E121562).

SIMPLEX CONNECTOR STYLES

HFBR-4501Z/ 4511Z – Simplex

The simplex connector provides a quick and stable connection for applications that require a component-to-connector retention force of 8 Newtons (1.8 lb.). These connectors are available in gray (HFBR-4501Z) or blue (HFBR-4511Z).

HFBR-4503Z/ 4513Z – Simplex Latching

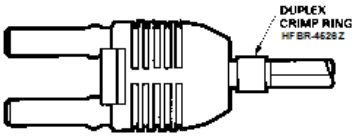


The simplex latching connector is designed for rugged applications requiring a greater retention force — 80 Newtons (18 lb.) — than provided by a simplex nonlatching connector. When inserting the simplex latching connector into a module, the connector latch mechanism should be aligned with the top surface of the horizontal modules, or with the tall vertical side of the vertical modules. Misalignment of an inserted latching connector into either module will not result in a positive latch. The connector is released by depressing the rear section of the connector lever, and then pulling the connector assembly away from the module housing.

The simplex latching connector is available in gray (HFBR-4503Z) or blue (HFBR-4513Z).

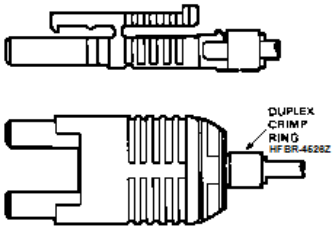
DUPLEX CONNECTOR STYLES

HFBR-4506Z – Duplex



Duplex connectors provide convenient duplex cable termination and are keyed to prevent incorrect insertion into duplex configured modules. The duplex connector is compatible with dual combinations of horizontal or vertical Versatile Link components (e.g., two horizontal transmitters, two vertical receivers, a horizontal transmitter with a horizontal receiver, etc.). The duplex nonlatching connector is available in parchment, off-white (HFBR-4506Z).

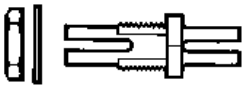
HFBR-4516Z – Duplex Latching



The duplex latching connector is designed for rugged applications requiring greater retention force than the nonlatching duplex connector. When inserting the duplex latching connector into a module, the connector latch mechanism should be aligned with the top surface of the dual combination of horizontal or vertical Versatile Link components. The duplex latching connector is available in gray (HFBR-4516Z).

Feedthrough/ Splice

HFBR-4505Z/ 4515Z Bulkhead Adapter



The HFBR-4505Z/4515Z adapter mates two simplex connectors for panel/bulkhead feedthrough of HFBR-4501Z/4511Z terminated plastic fiber cable. Maximum panel thickness is 4.1 mm (0.16 inch). This adapter can serve as a cable in-line splice using two simplex connectors. The adapters are available in gray (HFBR-4505Z) and blue (HFBR-4515Z). This adapter is not compatible with POF duplex, POF simplex latching, or HCS connectors.

Plastic Optical Fiber Connector Absolute Maximum Ratings

Parameter	Symbol	Min.	Max.	Unit	Note
Storage and Operating Temperature	$T_{S,O}$	-40	85	°C	1
Recommended Operating Temperature	T_O	-40	85	°C	1
Installation Temperature	T_I	0	70	°C	1
Nut Torque	T_N		0.7	N-m	2
HFBR-4505Z/ 4515Z Adapter			100	OzF-in.	

Notes:

1. Storage and Operating Temperatures refer to the ranges over which the connectors can be used when not subjected to mechanical stress. Installation Temperature refers to the ranges over which connectors may be installed onto the fiber and over which connectors can be connected and disconnected from transmitter and receiver modules.
2. Recommended nut torque is 0.57 N-m.

Plastic Optical Fiber Connector Mechanical/ Optical Characteristics

T_A = -40 to +85°C, Unless Otherwise Specified.

Parameter	Part Number	Symbol	Min.	Typ.[1]	Max.	Units	Temp. °C	Note
Retention Force, Connector to Versatile Link Transmitters and Receivers	Simplex, HFBR-4501Z/ 4511Z	F_{R-C}	7 3	8		N	+25 -40 to +85	2
	Simplex Latching, HFBR-4503Z/ 4513Z		47 11	80			+25 -40 to +85	
	Duplex, HFBR-4506Z		7 4	12			+25 -40 to +85	
	Duplex Latching, HFBR-4516Z		50 15	80			+25 -40 to +85	
Tensile Force, Connector to Cable	Simplex, HFBR-4501Z/ 4511Z	F_T	8.5	22		N		3
	Simplex Latching, HFBR-4503Z/ 4513Z		8.5	22				
	Duplex, HFBR-4506Z		14	35				
	Duplex Latching, HFBR-4516Z		14	35				
Adapter Connector to Connector Loss	HFBR-4505Z/ 4515Z with HFBR-4501Z/ 4511Z	α_{CC}	0.7	1.5	2.8	dB	25	4, 5
Retention Force Connector to Adapter	HFBR-4505Z/ 4515Z with HFBR-4501Z/ 4511Z	F_{R-B}	7	8		N		
Insertion Force, Connector to Versatile Link Transmitters and Receivers	Simplex, HFBR-4501Z/ 4511Z	F_I		8	30	N		6
	Simplex Latching, HFBR-4503Z/ 4513Z			16	35			
	Duplex, HFBR-4506Z			13	46			
	Duplex Latching HFBR-4516Z			22	51			

Notes:

1. Typical data are at +25°C.
2. No perceivable reduction in retention force was observed after 2000 insertions. Retention force of non-latching connectors is lower at elevated temperatures. Latching connectors are recommended for applications where a high retention force at high temperatures is desired.
3. For applications where frequent temperature cycling over temperature extremes is expected, please contact Avago Technologies for alternate connectoring techniques.
4. Minimum and maximum limit for α_{CC} for 0°C to +70°C temperature range. Typical value of α_{CC} is at +25°C.
5. Factory polish or field polish per recommended procedure.
6. Destructive insertion force was typically at 178 N (40 lb.).

Step-by-Step Plastic Cable Connectoring Instructions

The following step-by-step guide describes how to terminate plastic fiber optic cable. It is ideal for both field and factory installation. Connectors can be easily installed on cable ends with wire strippers, cutters and a crimping tool.

Finishing the cable is accomplished with the Avago HFBR-4593Z Polishing Kit, consisting of a Polishing Fixture, 600 grit abrasive paper and 3 µm pink lapping film (3M Company, OC3-14). The connector can be used immediately after polishing.

Materials needed for plastic fiber termination are:

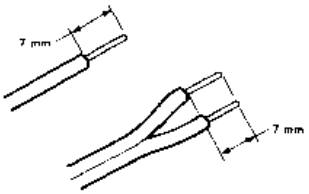
1. Avago Plastic Optical Fiber Cable (Example: HFBR-RUS500Z, HFBR-RUD500Z, HFBR-EUS500Z, or HFBR-EUD500Z)
2. Industrial Razor Blade or Wire Cutters
3. 16 Gauge Latching Wire Strippers (Example: Ideal Stripmaster™ type 45-092).
4. HFBR-4597Z Crimping Tool
5. HFBR-4593Z Polishing Kit
6. One of the following connectors:
   a) HFBR-4501Z/4503Z Gray Simplex/Simplex Latching Connector and HFBR-4525Z Simplex Crimp Ring
   b) HFBR-4511Z/4513Z Blue Simplex/Simplex Latching Connector and HFBR-4525Z Simplex Crimp Ring
   c) HFBR-4506Z Parchment (off-white) Duplex Connector and HFBR-4526Z Duplex Crimp Ring
   d) HFBR-4516Z Gray Latching Duplex Connector and HFBR-4526Z Duplex Crimp Ring

Step 1

The zip cord structure of the duplex cable permits easy separation of the channels. The channels should be separated a minimum of 100 mm (4 in) to a maximum of 150 mm (6 in) back from the ends to permit connectoring and polishing.

After cutting the cable to the desired length, strip off approximately 7 mm (0.3 in.) of the outer jacket with the 16 gauge wire strippers. Excess webbing on the duplex cable may have to be trimmed to allow the simplex or simplex latching connector to slide over the cable.

When using the duplex connector and duplex cable, the separated duplex cable must be stripped to equal lengths on each cable. This allows easy and proper seating of the cable into the duplex connector.



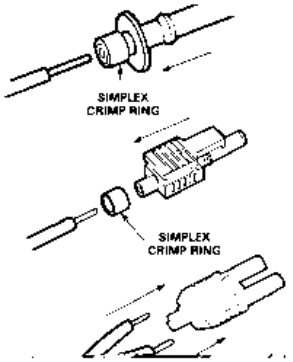
Step 2

Place the crimp ring and connector over the end of the cable; the fiber should protrude about 3 mm (0.12 in.) through the end of the connector. Carefully position the ring so that it is entirely on the connector with the rim of the crimp ring flush with the connector, leaving a small space between the crimp ring and the flange. Then crimp the ring in place with the crimping tool. One crimp tool is used for all POF connector crimping requirements.

For applications with extreme temperature operation or frequent temperature cycling, improved connector to cable attachment can be achieved with the use of an RTV (GE Company, RTV-128 or Dow Corning 3145-RTV) adhesive. The RTV is placed into the connector prior to insertion of the fiber and the fiber is crimped normally. The connector can be polished after the RTV has cured and is then ready for use.

Note: *By convention, place the gray connector on the transmitter cable end and the blue connector on the receiver cable end to maintain color coding (different color connectors are mechanically identical).*

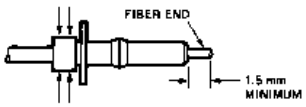
Simplex connector crimp rings cannot be used with duplex connectors and duplex connector crimp rings cannot be used with simplex connectors because of size differences. The simplex crimp has a dull luster appearance; the duplex ring is glossy and has a thinner wall.



Step 3

Any excess fiber protruding from the connector end may be cut off; however, the trimmed fiber should extend at least 1.5 mm (0.06 in) from the connector end.

Insert the connector fully into the polishing fixture with the trimmed fiber protruding from the bottom of the fixture. This plastic polishing fixture can be used to polish two simplex connectors or simplex latching connectors simultaneously, or one duplex connector.



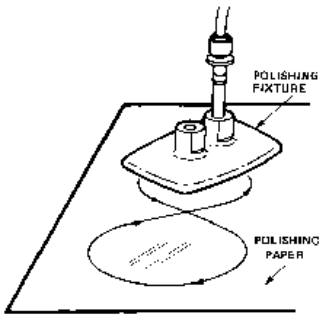
Note: *The four dots on the bottom of the polishing fixture are wear indicators. Replace the polishing fixture when any dot is no longer visible. Typically, the polishing fixture can be used 10 times; 10 duplex connectors or 20 simplex connectors, two at a time.*

Place the 600 grit abrasive paper on a flat smooth surface, pressing down on the connector, polish the fiber and the connector using a figure eight pattern of strokes until the connector is flush with the bottom of the polishing fixture. Wipe the connector and fixture with a clean cloth or tissue.

Step 4

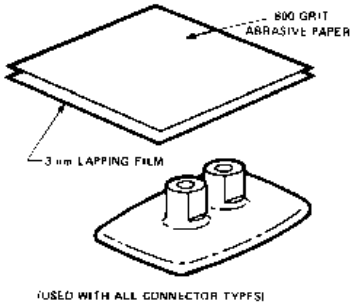
Place the flush connector and polishing fixture on the dull side of the 3 µm pink lapping film and continue to polish the fiber and connector for approximately 25 strokes. The fiber end should be flat, smooth and clean.

This cable is now ready for use.



Note: *Use of the pink lapping film fine polishing step results in approximately 2 dB improvement in coupling performance of either a transmitter-receiver link or a bulkhead/ splice over a 600 grit polish alone. This fine polish is comparable to the Avago factory polish. The fine polishing step may be omitted where an extra 2 dB of optical power is not essential, as with short link lengths. Proper polishing of the tip of the fiber/ connector face results in a tip diameter between 2.5 mm (0.098 in.) minimum and 3.2 mm (0.126 in.) maximum..*

HFBR-4593Z Polishing Kit



Ordering Guide for POF Connectors and Accessories

Plastic Optical Fiber Connectors

HFBR-4501Z	Gray Simplex Connector/Crimp Ring
HFBR-4511Z	Blue Simplex Connector/Crimp Ring
HFBR-4503Z	Gray Simplex Latching Connector with Crimp Ring
HFBR-4513Z	Blue Simplex Latching Connector with Crimp Ring
HFBR-4506Z	Parchment Duplex Connector with Crimp Ring
HFBR-4516Z	Gray Duplex Latching Connector with Crimp Ring
HFBR-4505Z	Gray Adapter (Bulkhead/Feedthrough)
HFBR-4515Z	Blue Adapter (Bulkhead/Feedthrough)

Plastic Optical Fiber Accessories

HFBR-4522Z	500 HFBR-0500Z Products Port Plugs
HFBR-4525Z	1000 Simplex Crimp Rings
HFBR-4526Z	500 Duplex Crimp Rings
HFBR-4593Z	Polishing Kit (one polishing tool, two pieces 600 grit abrasive paper, and two pieces 3 µm pink lapping film)
HFBR-4597Z	Plastic Fiber Crimping Tool

Ordering Guide for POF Cable

For Example:

HFBR-RUD500Z is a Standard Attenuation, Unconnectored, Duplex, 500 meter cable.

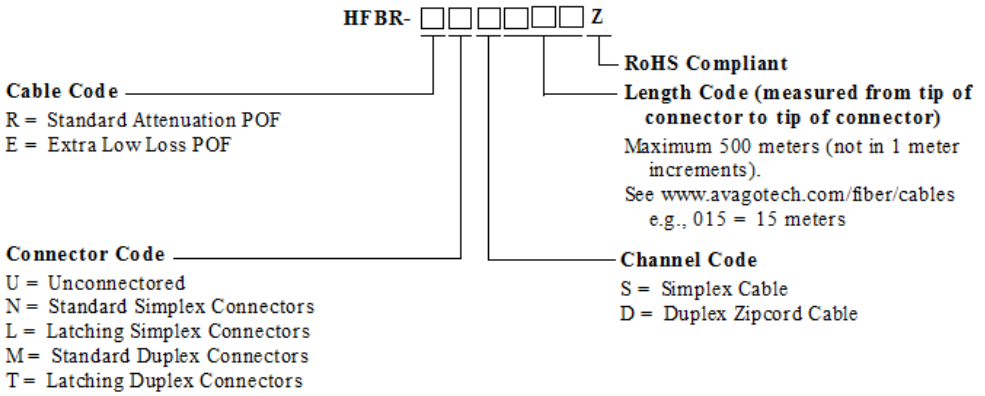
HFBR-RLS001Z is a Standard Attenuation, Latching Simplex Connectored, Simplex, 1 meter cable.

HFBR-RMD010Z is a Standard Attenuation, Standard Duplex Connectored, Duplex, 10 meter cable.

HFBR-RMD100Z is a Standard Attenuation, Standard Duplex Connectored, Duplex, 100 meter cable.

Cable Length Tolerances:
The plastic cable length tolerances are: +10%/-0%.

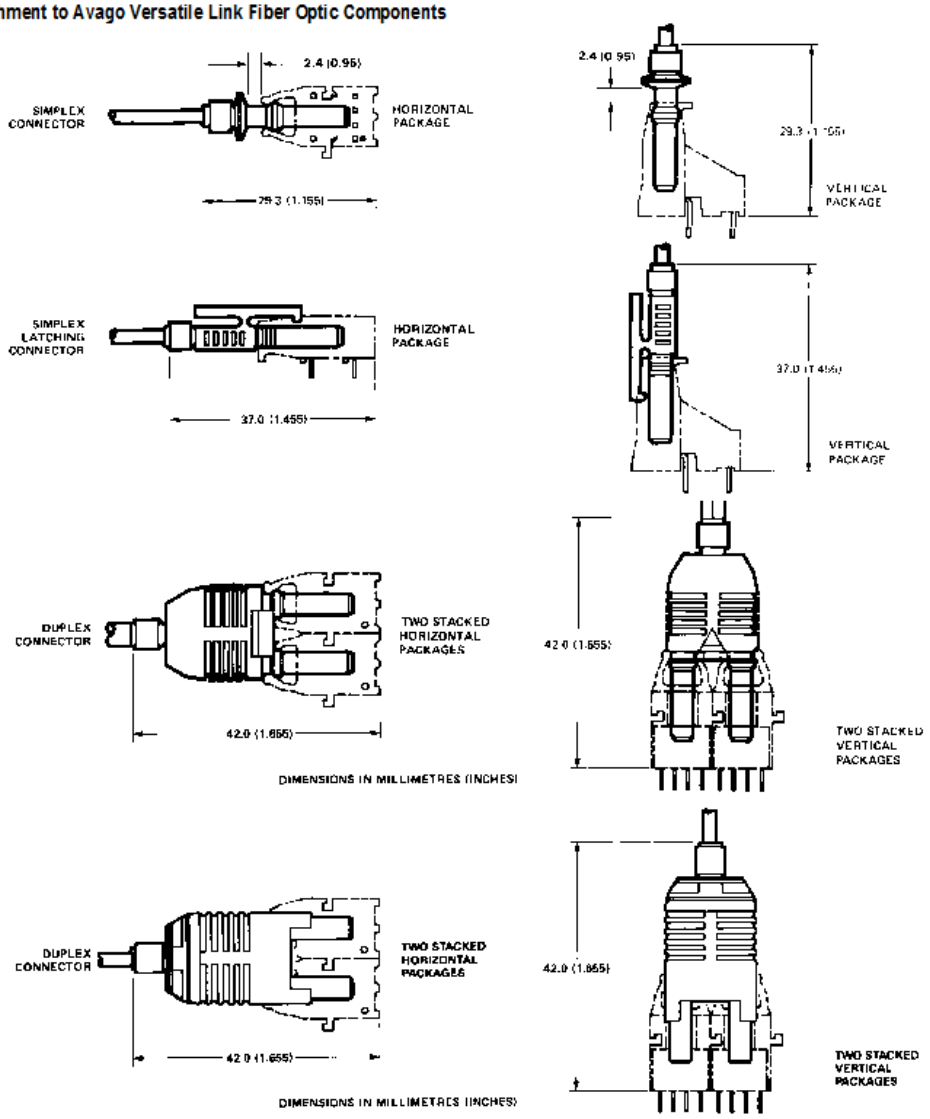
NOTE: By convention, pre-connectored simplex POF cables have gray and blue colored connectors on the opposite ends of the same fiber; although oppositely colored, the connectors are mechanically identical. Duplex POF cables with duplex connectors use color-coded markings on the duplex fiber cable to differentiate between the channel.



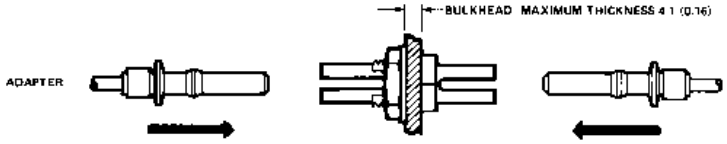
Note: Not all possible combinations reflect available part numbers. Please contact your local Avago representative for a list of current available cable part numbers.

Connector Applications

Attachment to Avago Versatile Link Fiber Optic Components



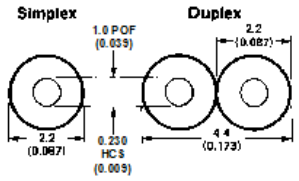
Bulkhead Feedthrough or Panel Mounting for HFBR-4501Z/ 4511Z Simplex Connectors



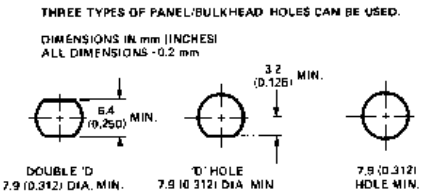
Versatile Link Mechanical Dimensions

All dimensions in mm (inches).
All dimensions ± 0.25 mm unless otherwise specified.

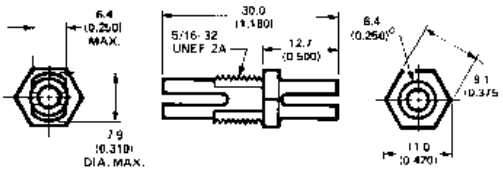
Fiber Optic Cable Dimensions



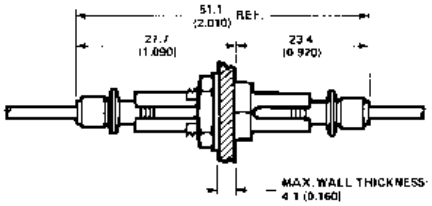
Panel Mounting – Bulkhead Feedthrough



HFBR-4505Z (Gray)/ 4515Z (Blue) Adapters



Bulkhead Feedthrough with Two HFBR-4501Z/ 4511Z Connectors



REFERENCES

Acton, Q. A. (2012). Issues in Telecommunications Research: 2011 Edition (Google eBook). Georgia: ScholarlyEditions.

Agrawal, G. (2010). Applications of Nonlinear Fiber Optics: Elsevier Science.

Al-Azzawi, A. (2006). Fiber Optics: Principles and Practices: Taylor & Francis.

Al-Suleimani, I., Phillips, A., & Woolfson, M. (2008). Performance evaluation of optically preamplified dicode pulse position modulation receivers. European Transactions on Telecommunications, 19(1), 47-52.

Aldibbiat, N., Ghassemlooy, Z., & McLaughlin, R. (2002). Dual header pulse interval modulation for dispersive indoor optical wireless communication systems. IEE Proceedings-Circuits, Devices and Systems, 149(3), 187-192.

Alexander, S. B. (1997). Optical Communication Receiver Design: SPIE Optical Engineering Press.

Alic, N., & Fainman, Y. (2004). Data-dependent phase coding for suppression of ghost pulses in optical fibers. IEEE Photonics Technology Letters, 16(4), 1212-1214.

Alpert, C. J., Mehta, D. P., & Sapatnekar, S. S. (2008). Handbook of Algorithms for Physical Design Automation: Taylor & Francis.

Altera. (2009). High Speed Mezzanine Card (HSMC) Specification Retrieved 10 Feb, 2014, from http://www.altera.co.uk/literature/ds/hsmc_spec.pdf

Altera. (2010). Cyclone III FPGA development kit Retrieved 20 Feb, 2014, from http://www.altera.com/products/devkits/altera/kit-cyc3.html

Anderson, D. R., Johnson, L. M., & Bell, F. G. (2004). Troubleshooting Optical Fiber Networks: Understanding and Using Optical Time-Domain Reflectometers: Elsevier Science.

Atkin, G., & Fung, K. (1990). Performance analysis of coded optical PPM system using direct and coherent detection. IEE Proceedings I (Communications, Speech and Vision), 137(4), 226-232.

Bagad, V. S., & Dhotre, I. A. (2009). Data Communication Systems: Technical Publications.

Band, Y. B. (2006). Light and matter: electromagnetism, optics, spectroscopy and lasers (Vol. 1). New Jersey: John Wiley & Sons.

Bandyopadhyay, N. (2014). OPTICAL COMMUNICATION AND NETWORKS: PHI Learning.

Betten, A., Braun, M., Fripertinger, H., Kerber, A., Kohnert, A., & Wassermann, A. (2006). Error-Correcting Linear Codes: Classification by Isometry and Applications: Springer.

Blogh, J. S., & Hanzo, L. L. (2002). Third-generation systems and intelligent wireless networking: smart antennas and adaptive modulation: John Wiley & Sons.

Brown, S. D., & Vranesic, Z. G. (2009). Fundamentals of digital logic with VHDL design. New York: McGraw-Hill

Charitopoulos, R. (2009). Implementation & Performance Investigation of Dicode PPM over Dispersive Optical Channels. Doctoral thesis, University of Huddersfield.

Charitopoulos, R., & Sibley, M. J. (2009). Experimental coder/decoder of dicode pulse position modulation. Paper presented at the Proceedings of Computing and Engineering Annual Researchers' Conference, Huddersfield, UK.

Charitopoulos, R., Sibley, M. J. N., & Mather, P. (2011). Maximum likelihood sequence detector for dicode pulse position modulation. IET, 5(6), 261-264.

Chatzidiamantis, N. D., Uysal, M., Tsiftsis, T. A., & Karagiannidis, G. K. (2009). EM-based maximum-likelihood sequence detection for MIMO optical wireless systems. Paper presented at the Communications, 2009. ICC'09. IEEE International Conference on.

Crisp, J. (2005). Introduction to Fiber Optics: Elsevier Science.

Crowell, G., & Press, R. (2004). Using Scan Based Techniques for Fault Isolation in Logic Devices0871708043, 9780871708045 Microelectronics Failure Analysis: Desk Reference (pp. 132-138): ASM International.

Cryan, R., & Sibley, M. J. (2006). Minimising intersymbol interference in optical-fibre dicode PPM systems. Paper presented at the Optoelectronics, IEE Proceedings-.

Cryan, R., & Unwin, R. (1990a). Heterodyne n-ary PPM employing Reed-Solomon codes. Paper presented at the Global Telecommunications Conference, 1990, and Exhibition.'Communications: Connecting the Future', GLOBECOM'90., IEEE.

Cryan, R., & Unwin, R. (1990b). Reed-Soloman coded optical fibre digital PPM: Approaching Fundamental Limits. Paper presented at the IEEE International Conference on Communication Systems, Singapore.

Cryan, R., & Unwin, R. (1992). Reed—solomon coded homodyne digital pulse position modulation. IEE Proceedings I (Communications, Speech and Vision), 139(2), 140-146.

Curtis, K., Dhar, L., Hill, A., Wilson, W., & Ayres, M. (2010). Holographic data storage. Wiley.

Cvijetic, M. (2004). Optical Transmission Systems Engineering: Artech House.

Darnell, M. (1985). Error control coding: Fundamentals and applications. Communications, Radar and Signal Processing, IEE Proceedings F, 132(1), 68.

Davidson, F. M., & Sun, X. (1989). Slot clock recovery in optical PPM communication systems with avalanche photodiode photodetectors. Communications, IEEE Transactions on, 37(11), 1164-1172.

Divsalar D, Gagliardi R.M, & J.H. Yuen. (1982). PPM Demodulation for Reed-Solomon Decoding for Optical Space Channel: Jet Propulsion Laboratory, California Institute of Technology.

Dolinar Jr, S. (1983). A near-optimum receiver structure for the detection of M-ary optical PPM signals. The Telecommunications and Data Acquisition Progress Report, 42-72.

Downing, J. N. (2004). Fiber-optic Communications: Thomson/Delmar Learning.

Elmirghani, J., Cryan, R., & Clayton, M. (1992a). Spectral characterisation and frame synchronisation of optical fibre digital PPM. Electronics Letters, 25(16), 1482-1483. doi: 10.1049/el:19920941

Elmirghani J. M. H, Cryan R. A, & Clayton F. M. (1992c, 1992). Frame synchronisation for optical fibre digital PPM.

Elmirghani, J. M., Cryan, R. A., & Clayton, F. (1992b). Optical fiber n-ary PPM: the question of slot synchronization. Paper presented at the Fibers' 92.

Fleetwood, D. M., & Schrimpf, R. D. (2004). Radiation effects and soft errors in integrated circuits and electronic devices (Vol. 34): World Scientific.

Forouzan, B. A., & Fegan, S. C. (2003). Local area networks (Vol. 1): McGraw-Hill.

Ghassemlooy, Z., & Hayes, A. (2000). Digital pulse interval modulation for IR communication systems—a review. International Journal of Communication Systems, 13(7-8), 519-536.

Gho, G.-H., & Kahn, J. M. (2012). Rate-adaptive modulation and coding for optical fiber transmission systems. Journal of Lightwave Technology, 30(12), 1818-1828.

Gho, G.-H., Klak, L., & Kahn, J. M. (2011). Rate-adaptive coding for optical fiber transmission systems. Lightwave Technology, Journal of, 29(2), 222-233.

Ghosna, F., & Sibley, M. J. (2010). Pulse position modulation coding schemes for optical intersatellite links. Electronics Letters, 46(4), 290-291.

Goff, D. (2002). Fiber Optic Reference Guide: Taylor & Francis.

Gol'Dshteyn, Y. A., & Frezinskiy, B. (1979). Investigation of the Transmission of a Multi-Position PPM Optical Signal Through a Communications Line Containing Repeaters. RADIO ENG. & ELECTRON. PHYS., 24(7), 65-71.

Goldsmith, A. (2005). Wireless communications: Cambridge university press.

Guimaraes, D. A. (2002). Local Area Networks. New York: Tata Mc-Graw Hill Education.

Guimaraes, D. A. (2010). Digital Transmission: A Simulation-Aided Introduction with VisSim/Comm: Springer.

Hecht, J. (2004). City of Light: The Story of Fiber Optics: Oxford University Press.

Held, G. (2008). Introduction to Light Emitting Diode Technology and Applications: CRC Press.

Hemmati, H. (2006). Deep Space Optical Communications (Google eBook ed.). New Jersey: John Wiley & Sons.

Herceg, M., Švedek, T., & Matić, T. (2010). Pulse interval modulation for ultra-high speed IR-UWB communications systems. EURASIP Journal on Advances in Signal Processing, 2010, 48.

Hetzel, P. (1988). Time dissemination via the LF transmitter DCF77 using a pseudo-random phase-shift keying of the carrier. Paper presented at the Proceedings of the 2nd European Frequency and Time Forum (EFTF).

Houghton, A. (2001). Error coding for engineers. Berlin: Springer, Science & Business Media.

Infocellar. (2015). Wave Division Multiplexing Computer Technology & Network Communications Retrieved June, 2015, from http://www.infocellar.com/networks/fiber-optics/

Introduction to the VHDL Language. (n.d.) Retrieved 20 Feb, 2014, from http://home.deib.polimi.it/sami/VHDL_merged.pdf

Janssen, C. (2014). Optical Communication Retrieved 20 August, 2014, from http://www.techopedia.com/definition/24942/optical-communication

Kadrid, E. (2011). An FPGA Implementation for a High-Speed Optical Link with a PCIe Interface Retrieved 15 Feb, 2014, from http://www.seas.upenn.edu/~ekadric/masters_thesis.pdf

Karp, S., & Gagliardi, R. M. (1969). The design of a pulse-position modulated optical communication system. Communication Technology, IEEE Transactions on, 17(6), 670-676.

Katz, R. H., & Borriello, G. (2005). Contemporary logic design (Second ed.). New York: Pearson Prentice Hall.

Kaur, G. (2011). VHDL: Basics to Programming. India: Dorling Kindersley.

Kim, C., Rhee, S., Kim, J., & Jee, Y. (2010). Product Reed-Solomon codes for implementing NAND flash controller on FPGA chip. Paper presented at the Computer Engineering and Applications (ICCEA), 2010 Second International Conference on.

Ko, Y. C. (2011). Pulse-Position Modulation Retrieved 2 Feb, 2014, from http://ocw.korea.edu/ocw/college-of-engineering/d1b5c2e0c774b860/lecturenote23June2.pdf

Kurzweil, H., Seidl, M., & Huber, J. B. (2011). Reduced-complexity collaborative decoding of interleaved Reed-Solomon and Gabidulin codes. arXiv preprint arXiv:1102.3126.

Kythe, D. K., & Kythe, P. K. (2012). Algebraic and stochastic coding theory: CRC Press.

Lala, P. K. (1996). Practical digital logic design and testing. USA: Prentice-Hall, Inc.

Lamba, T. S., Biswas, P., & Pathak, S. (2005). Proceedings of the Eleventh National Conference on Communications: NCC-2005, 28-30 January, 2005: Allied Publishers.

Lapstun, P. (2009). Coding pattern comprising direction codes: Google Patents.

Lazaridis, G. (2011). Pulse Position Modulation and Differential PPM Retrieved 20 August 2014, from http://www.pcbheaven.com/wikipages/Pulse_Position_Modulation/

Lee, G., & Schroeder, G. (1977). Optical pulse position modulation with multiple positions per pulsewidth. Communications, IEEE Transactions on, 25(3), 360-364.

Lin, S., & Costello, D. J. (1983). Error Control Coding—Fundamentals and Applications, Vol. 1 of Computer Applications in Electrical Engineering Series: Prentice–Hall, Englewood Cliffs, New Jersey.

Ling, G., & Gagliardi, R. M. (1986). Slot synchronization in optical PPM communications. Communications, IEEE Transactions on, 34(12), 1202-1208.

Liu, A. (2002). The Mathematical Theory of Communication Retrieved 2 Feb, 2014, from http://oldsite.english.ucsb.edu/faculty/ayliu/unlocked/shannon/mathematical-theory.html

Lu, Y., Willi, M., & Serge, V. (2005). The conditional correlation attack: A practical attack on bluetooth encryption. Paper presented at the Advances in Cryptology–CRYPTO 2005.

Maini, A. K. (2007). Digital electronics: principles, devices and applications: John Wiley & Sons, Inc.

McDaniel, T. W., & Victora, R. (1995). Handbook of Magneto-Optical Data Recording: Materials, Subsystems, Techniques: Elsevier Science.

McEliece, R. (1979). Coding for the photon channel. Paper presented at the NTC'79; National Telecommunications Conference, Volume 2.

McEliece, R. J. (1981). Practical codes for photon communication. Information Theory, IEEE Transactions on, 27(4), 393-398.

Mecherle, G. S. (1985). Impact of laser diode performance on data rate capability of PPM optical communication. Paper presented at the Military Communications Conference, 1985. MILCOM 1985. IEEE.

Mecherle, G. S. (1986). Detection alternatives for pulse position modulation (PPM) optical communication. Paper presented at the OE/LASE'86 Symp (January 1986, Los Angeles).

Mori, R., & Tanaka, T. (2010). Non-binary polar codes using Reed-Solomon codes and algebraic geometry codes. Paper presented at the Information Theory Workshop (ITW), 2010 IEEE.

Nam, H. (2006). Joint diversity combining technique and adaptive modulation in wireless communications: ProQuest.

Nguyen, T. T., & Lampe, L. (2009). Coded pulse-position modulation for free-space optical communications. Paper presented at the Communications, 2009. ICC'09. IEEE International Conference on.

Nguyen, T. T., & Lampe, L. (2010). Coded multipulse pulse-position modulation for free-space optical communications. Communications, IEEE Transactions on, 58(4), 1036-1041.

Nikolaidis, K. (2008). An Investigation of an Optical Multiple Pulse Position Modulation Link over a Dispersive Optical Channel. Dissertation/Thesis. Retrieved from http://eprints.hud.ac.uk/6980/

Osterberg, U. (2003). Signal processing in optical fibers. Mathematical Sciences Research Institute Publications, 301.

Pavert, L. (2011). REED-SOLOMON ENCODING AND DECODING Retrieved 15 Feb, 2014, from http://vandepavert.fi/uploads/media/Reed-Solomon_Encoding_and_Decoding_01.pdf

Picone, A. (2013). New lower bounds for the minimum distance of generalized algebraic geometry codes. Journal of Pure and Applied Algebra, 217(6), 1164-1172.

Pires, J., & Da Rocha, J. (1986). Digital pulse position modulation over optical fibres with avalanche photodiode receivers. Paper presented at the Optoelectronics, IEE Proceedings J.

Prösch, R. (2009). Technical Handbook for Radio Monitoring HF: Edition 2009: BoD–Books on Demand.

Prösch, R., & Daskalaki-Prösch, A. (2011). Technical Handbook for Radio Monitoring VHF/UHF: Edition 2011: BoD–Books on Demand.

Riley, M., & Richardson, I. (1998). Reed-Solomon Codes Retrieved 1 Augest, 2014, from http://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html

Sankaran, J. (2000). Reed Solomon decoder: TMS320C64x implementation. Application Report SPRA686, Texas Instruments.

Sasaoka, H. (2000). Mobile communications: IOS Press.

Schubert, E. F. (2006). Light-Emitting Diodes: Cambridge University Press.

Senior, M. J., & Jamro, Y. (2009). Optical Fiber Communications: Principles and Practice. India: Pearson Education India.

Shalaby, H. M. (1999). A performance analysis of optical overlapping PPM-CDMA communication systems. Lightwave Technology, Journal of, 17(3), 426-433.

Shirokov, G. A., & Bukhinnik, A. (1984). Evaluation of the reliability of signal transmission along digital optical fibre channels with differential pulse position keying. Telecomm. And Radio Engineering, 39(7).

Sibley, M. J. N. (1987). The design and construction of a Digital pulse position modulation coder and decoder. PostDectoral, Fellowship Report University of Huddersfield.

Sibley, M. J. N. (1993). Design implications of high-speed digital PPM. Paper presented at the SPIE's 1993 International Symposium on Optics, Imaging, and Instrumentation.

Sibley, M. J. N. (1995). Optical Communications: Components and Systems (Second ed.). UK: The Macmillan Press LTD

Sibley, M. J. N. (2003). Dicode pulse-position modulation: A novel coding scheme for optical-fibre communications. IEE Proceedings: Optoelectronics, 150(2), 125-131. doi: 10.1049/ip-opt:20030386

Sibley, M. J. N. (2004). Suboptimal filtering in a zero-guard, dicode PPM system operating over dispersive optical channels. IEE Proceedings-Optoelectronics, 151(4), 237-243.

Sibley, M. J. N. (2005). Performance analysis of a dicode PPM system, operating over plastic optical fibre, using maximum likelihood sequence detection. IEE Proceedings-Optoelectronics, 152(6), 337-343.

Sibley, M. J. N. (2012). Comparison of several pulse position modulation schemes when operating over plastic optical fibre. Paper presented at the Transparent Optical Networks (ICTON), 2012 14th International Conference on.

Sibley, M. J. N., & Massarella, A. J. (1993). Detection of digital pulse position modulation over highly/slightly dispersive optical channels. Paper presented at the Berlin-DL tentative.

Sivalingam, K. M., & Subramaniam, S. (2006). Optical WDM Networks: Principles and Practice: Springer US.

Sklar, B. (2001a). Digital communications fundamentals and applications (Second ed.): Prentice Hall Communications Engineering and Emerging Techno.

Sklar, B. (2001b). Reed-Solomon Codes Retrieved 10 Feb, 2014, from http://ptgmedia.pearsoncmg.com/images/art_sklar7_reedsolomon/elementLinks/art_sklar7_reed-solomon.pdf

Svelto, O. (2010). Principles of Lasers: Springer.

Thyagarajan, K., & Ghatak, A. (2010). Lasers: Fundamentals and Applications: Springer US.

Tocci, R. J., Widmer, N. S., & Moss, G. L. (2011). Digital Systems: Principles and Applications: Pearson Education, Limited.

Wang, H., Cheng, G., Sun, X., & Zhang, T. (2007). Performance analysis of dicode pulse position modulation for optical wireless communications. Paper presented at the Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on.

Watts, P., Waegemans, R., Glick, M., Bayvel, P., & Killey, R. (2006). An FPGA-based optical transmitter using real-time DSP for implementation of advanced signal formats and signal predisortion. The European Conference on Optical Communications, ECOC Retrieved 20 Feb, 2014, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.4224&rep=rep1&type=pdf

Wicker, S. B., & Bhargava, V. K. (1999). Reed-Solomon codes and their applications: John Wiley & Sons.

Wilson, B., & Ghassemlooy, Z. (1993). Pulse time modulation techniques for optical communications: a review. IEE Proceedings J (Optoelectronics), 140(6), 346-358.

Xu, F., Khalighi, M., & Bourennane, S. (2009). Pulse position modulation for FSO systems: capacity and channel coding. Paper presented at the Telecommunications, 2009. ConTEL 2009. 10th International Conference on.

Zhu, X., & Kahn, J. M. (2003). Markov chain model in maximum-likelihood sequence detection for free-space optical communication through atmospheric turbulence channels. Communications, IEEE Transactions on, 51(3), 509-516.

Zwillinger, D. (1988). Differential PPM has a higher throughput than PPM for the band-limited and average-power-limited optical channel. Information Theory, IEEE Transactions on, 34(5), 1269-1273.