CONVOLUTION ENCODER FOR FORWARD ERROR CORRECTION

AHMAD TERMIZI BIN MOHD AZMI

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG ◆

BORANG PENGESAHAN STATUS TESIS

JUDUL: **CONVOLUTION ENCODER FOR FORWARD ERROR CORRECTION**

SESI PENGAJIAN: **2010/2011**

Saya **AHMAD TERMIZI BIN MOHD AZMI (880222-08-6611)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

594 BENDANG KERAJAAN,
33300, GERIK
PERAK DARUL RIDZUAN

NOR FARIZAN BINTI ZAKARIA
(Nama Penyelia)

Tarikh: **30 NOVEMBER 2010**

Tarikh: : **30 NOVEMBER 2010**

CATATAN:
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

CONVOLUTION ENCODER FOR FORWARD ERROR CORRECTION

AHMAD TERMIZI BIN MOHD AZMI

This thesis is submitted as partial fulfillment of the requirement for the award of the Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronic Engineering

University Malaysia Pahang

NOVEMBER, 2010

Signature : _____

Author : AHMAD TERMIZI BIN MOHD AZMI

Date : 30 NOVEMBER 2010

“Specially dedicated to

My beloved parents, brothers, sister, and all friends”

"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering
(Electronic)"

Signature : _____

Name : NOR FARIZAN BINTI ZAKARIA

Date : 30 NOVEMBER 2010

ACKNOWLEDGMENT

Alhamdulillah, the highest thanks to God because with His Willingness I can complete the final year project in time.

I would like to express my gratitude to my dedicated supervisor, Madam Nor Farizan binti Zakaria for guiding this project with clarity and that priceless gift of getting things done by sharing her valuable ideas as well as her knowledge.

I also would like to thank to my family, UMP lecturers, electrical technicians, and my best colleagues at that have provide assistance at various occasions. Their views are useful indeed.

The great cooperation, kindheartedness and readiness to share worth experiences that have been shown by them will be always appreciated and treasured by me. Once again, thank you very much.

ABSTRACT

Nowadays bandwidth demands are totally increase and the tolerance for errors and latency decreases, designers of data-communication systems are looking for new ways to expand available bandwidth and improve the quality of transmission. One solution isn't actually new, but has been around for a while. Nevertheless, it could prove quite useful. Called forward error correction (FEC), this design technology has been used for years to enable efficient, high-quality data communication over noisy channels, such as those found in satellite and digital cellular-communications applications. The big attraction of FEC technology is how it adds redundant information to a data stream. This enables a receiver to identify and correct errors without the need for retransmission and the data will be transfer faster than ever.

ABSTRAK

Pada zaman serba canggih sekarang ini keperluan jalur lebar yang benar-benar meningkat dan kesungguhan untuk mengurangkan kesalahan dan latensi, pereka sistem komunikasi data telah mencari cara baru untuk memperluaskan jalur lebar yang telah sedia ada dan mempertingkatkan lagi kualiti penghantaran maklumat. Salah satunya adalah kaedah yang lama tetapi telah di pertingkatkan penggunaanya untuk kemudahan yang lebih luas. Kaedah yang digunakan adalah telah terbukti sangat berguna. “Forward Error Correction” (FEC), adalah teknologi yang telah dicipta dan telah digunakan selama bertahun-tahun untuk membolehkan penghantaran komunikasi yang lebih cekap, data komunikasi yang lebih berkualiti tinggi apabila melalui gangguan saluran, seperti yang ditemui dalam satelit dan digital-aplikasi komunikasi bimbit. Kelebihan utama yang terdapat pada teknologi FEC ini adalah bagaimana ia dapat menambah maklumat secara berlebihan untuk satu aliran data. Hal ini membolehkan penerima untuk mengenal pasti dan memperbaiki kesalahan tanpa memerlukan penghantaran semula dan pemindahan data akan lebih cepat daripada sebelumnya.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

FEC	-	Called Forward Error Correction
VHDL	-	Very High Speed Integrated Circuit Hardware Description
QoS	-	Quality of Service
ARQ	-	Automatic Repeat Request
XOR	-	Exclusive OR, XOR-gates
CAD	-	Computer-Aided Design
AWGN	-	Additive White Gaussian Noise
$G1$	-	Generator Polynomial 1
$G2$	-	Generator Polynomial 2
R	-	Rate
k	-	Input
n	-	Output
L	-	Constraint Length
m	-	Memory Register

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

Convolution encoder is a code that have been widely used in numerous applications in order to achieve reliable data transfer, including digital video broadcasting, digital audio broadcasting, satellite communication, cellular mobile, and satellite communication. As the capabilities of FEC increase, the number of errors that can be corrected also increases. The advantage is obvious. Noisy channels create a relatively large number of errors. The ability to correct these errors means that the noisy channel can be used reliably. This enhancement can be parlayed into several system improvements, including bandwidth efficiency, extended range, higher data rate, and greater power efficiency, as well as increased data reliability.

Convolution code is a type of error correcting code that is normally used in telecommunication. On the other hand, this convolution encoding is used to encode data prior to transmission over a channel. The received data is decoded by the classic Viterbi decoder. In a basic convolution encoder, two or three bits (depending on the encoder output rate) are transmitted over the channel for every input bit.

Its popularity of using the convolution encoder for forward error correction is came from the structure and availability that is easy and simple to implement. The purposes of convolution code are to improve channel capacity during the transmission and the other is to mitigate burst error occurs the transmission.

In developing digital system design, a main techniques use is by using Very High Speed Integrated Circuit Hardware Description Language (VHDL) in order to programmed it in software where simulation can be perform to do analysis and then the result will be compared to the analysis result that have been perform by using MATLAB software. Xilinx ISE 10.1 and MATLAB software are use in order to encode the data and develop a convolution encoder.

1.2 Problem Statement

Modern digital communication system requirements are becoming more and more stringent with respect to error-free transmission. Next generation systems would likes to offer Quality of Service (QoS) guarantees to users, this cannot be done unless more efficient error correction schemes can be implemented. There is also exponential growth in the Wireless industry for the same demands but that require less power.

The Convolution Encoder for Forward Error Correction (FEC) is used to implement and solve this problem. This method will allow the receiver to detect and correct the errors (within some bound) without the need to ask the sender for additional data, compared to Automatic Repeat Request (ARQ) method which is if the sender does not receive an acknowledgment before the timeout, it will re-transmits the frame/packet data until the sender receives an acknowledgment or exceeds a predefined number of re-transmission

1.3 Objectives of the project

The objectives of this project:

i) To developed and design the convolution encoder by using Very High Speed Integrated Circuit Hardware Description Language (VHDL) in Xilinx ISE 10.1 software.

ii) To compare the result with convolution encoder used in MATLAB and Xilinx software

1.4 Scope of project

The scope of the project has been narrow down from the objective to ensure the goal target is achieved when the result are conclude. The scope of the project has been specified as below:

i) The data out that was scramble out by MATLAB software will be verified again by using Very High Speed Integrated Circuit Hardware Description Language (VHDL) in Xilinx software to get the same data output

ii) Basic convolution encoder rate 1/2 with constraint length 7 will be use

iii) Same input data Bernoulli Binary Generator are use in both simulation process

In other word, these scopes create a basic convolution code that demonstrates the detection of the error in the transmission of data in communication system.

1.5 Thesis Outline

This section will give an outlines of the structure of the thesis. This thesis will consist of five chapters including this chapter. The following is an explanation for each chapter:

Chapter 2 discusses the previous work that been done around the world about the convolution encoder, in term of definition, algorithm, and modeling system design. Literature that been done will cover, for instance, history, algorithm design and others.

Chapter 3 explain on methodology of this project. In this chapter, each step in the work methodology flow chart starting from modelling of the convolution encoder block was explained.

.

Chapter 4 consists of experimental results and results analysis. Comparison between each graphically result was done.

Lastly, Chapter 5 summarizes the overall conclusion for this thesis and a few suggestion and recommendation for future development.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

This part will explain the research information that is related to completing this project. All the research sources are from books, journals, websites and some articles.

2.2 FEC

Forward error correction (FEC) is techniques that introduce redundancy to allow for correction of error without transmission. This technique are used in system where a reverse channel is not available for requesting retransmission, the delay with

retransmission would be excessive, the expected number of error would require a large number of retransmission, or retransmission would be awkward to implement [4].

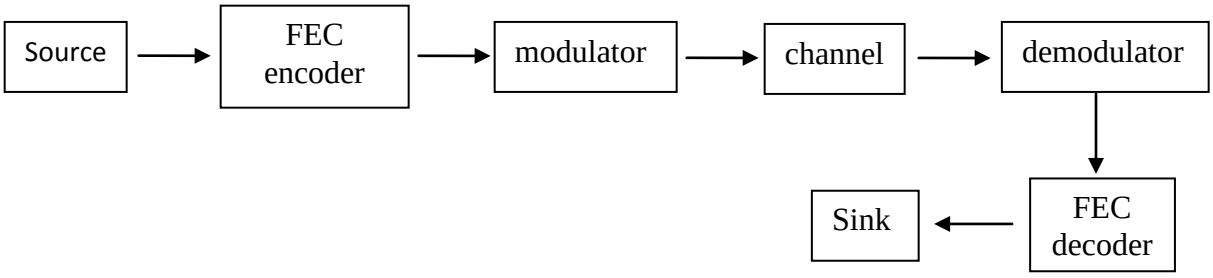


Figure 2.0: Process Flow Data Transmit and Receive

The FEC code acts on a discrete data channel comprised of all system elements between the encoder output and decoder input. The encoder maps the source data to q-ary code symbols that are modulated and transmitted. During the transmission, the signal can be corrupted, causing errors to arise in the demodulated symbol sequence. The FEC decoder attempts to correct these errors and restore the original source data [4].

2.2.1 Convolution Encoder

Shannon's Noisy Channel Coding Theorem says that "With every channel we can associate a 'channel capacity' C (bits/sec). There exist such error control codes that information can be transmitted at a rate below C (bits/sec) with an arbitrarily low bit error rate" [3].

Convolution codes were first introduced by Elias [14] in 1955. He proved that redundancy could be added to an information stream through the use of linear shift

register. In 1961, Wozencraft and Reiffen describe the first practical decoding algorithm for convolution codes [15]. The algorithm was based on sequential decoding, however sub-optimal for decoding convolution codes.

Several other algorithms were developed off of Wozencraft and Reiffen initial work. In 1967, Viterbi proposed a maximum likelihood decoding scheme for decoding convolution codes. The importance of the Viterbi algorithm is that it proved to be relatively easy to implement given the encoder has a small number of memory elements [16].

Channel coding is the process of adding the redundancy information. Convolution coding and block coding are two major forms of channel coding. Convolutional codes operate on serial data, one or few bits at a time while the block codes operate on relatively large message blocks [1].

The encoding process of convolutional codes is significantly different to that of block encoding. Block codes are developed through the use of algebraic techniques. Block encoders group information bits into length k blocks. These blocks are then mapped into codeword's of length n. A convolutional encoder converts the entire input stream into length n codeword's independent of the length k. The development of convolutional codes is based mostly on physical construction techniques. The evaluation and the nature of the design of convolutional codes depends less on an algebraic manipulation and more on construction of the encoder [3].

Convolutional codes are described by two parameters: the code rate R=k/n, expressed as a ratio of the number of input bits of the convolutional encoder (k) to the number of channel symbols in the output of the convolutional encoder (n), and the

constraint length L, indicating how many k-bit stages are available to feed the combinatorial logic (exclusive OR, XOR-gates) that produces the output symbols [6].

2.2.2 Error - Control Coding

In this section, an example is shown to show how the encoded sequence is by hand. So that, a clear understanding how the encoded sequence is obtained without using calculator. The same method shown in [12] can be used to calculate the example below:

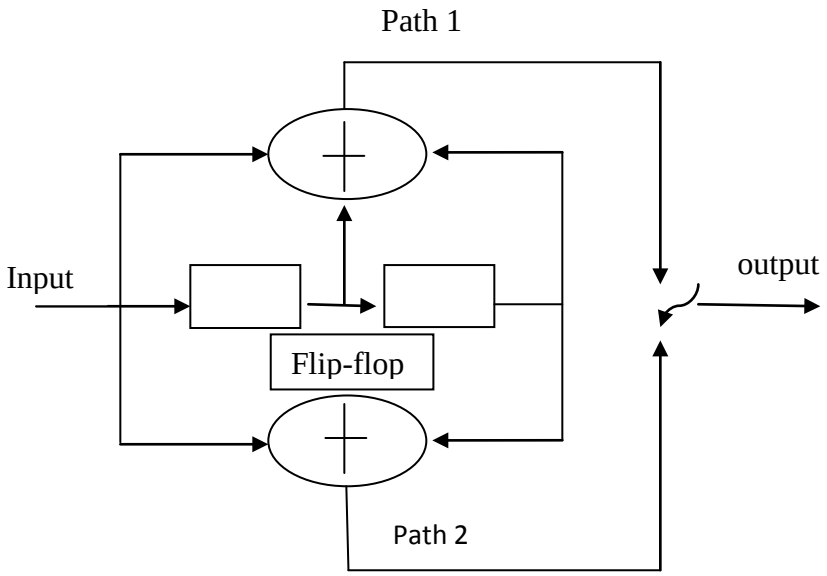


Figure 2.1: Constraint Length 3, and 1/2 convolution rate

Example:

Consider the convolution encoder figure 2.1 which has two paths numbered 1 and 2 for convenience of reference. The impulse response of path 1 is (1,1,1). Hence the corresponding generator polynomial is given by

$$g^{(1)}(D) = 1 + D + D^2$$

The impulse response of path 2 is (1,0,1). Hence the corresponding generator polynomial is given by

$$g^{(2)}(D) = 1 + D^2$$

For the message sequence (10011), say we have the polynomial representation

$$m(D) = 1 + D^3 + D^4$$

As with Fourier transformation, convolution in the domain is transformed into multiplication in the D-domain. Hence the output polynomial of path 1 is given by

$$\begin{aligned} c^{(1)}D &= g^{(1)}(D)m(D) \\ &= (1 + D + D^2)(1 + D^3 + D^4) \\ &= 1 + D + D^2 + D^3 + D^6 \end{aligned}$$

From this we immediately deduce that the output sequence of path 1 is (1111001). Similarly, the output polynomial of path 2 is given by

$$\begin{aligned} c^{(2)}(D) &= g^{(2)}(D)m(D) \\ &= (1 + D^2)(1 + D^3 + D^4) \\ &= 1 + D^2 + D^3 + D^4 + D^5 + D^6 \end{aligned}$$

The output sequence of path 2 is therefore (1011111). Finally, multiplexing the two output sequences path 1and 2, we get the encoded sequence

$$c = (11,10,11,11,01,01,11)$$

2.3 VHDL

VHDL is an industry standard language for modeling digital circuits. The original version, adopted in 1987, called IEEE standard 1076. IEEE 1164, a revised standard, was adopted 1n 1993. Although originally intended for design documentation and simulation, today VHDL is also used in computer-aided design (CAD) design entry [13].

The first step is to consider the specification of requirement that the algorithm is satisfy. In other word, the developers have to consider the limitation of the input for instants the same rate, memory register, and the constraint length so that the designed system is capable to operate [2].

VHDL is one of three popular modern HDL languages. A second HDL is Verilog, it was developed to have syntax similar to the C programming language. The third HDL is system C which has developed on 2000's by several companies [5].

VHDL stands for Very High Speed Integrated Circuit Hardware Description Language. This VHDL language can be used in several goals in mind. It may be used for the system description and documentation, synthesis of digital circuits, simulation of digital system, or verification and validation of digital systems [2].

Table 2.0: List of main keywords of VHDL

Constructs for	**VHDL**
Entity declaration	**Entity** circuit **is port (** a: **in** std_logic; b: **out** std_logic; c: **in** std_logic_vector (3 downto 0); d: **out** std_logic_vector (0 to 7); end circuit;
Internal signals, variables, constant	**Signal inta**: std_logic; **Signal intb**: std_logic_vector (3 downto 0); **Signal** counter: integer range -127 to 127; **Variable** temp: std_logic_vector (0 to 7) **Constant** C: std_logic_vector (3 downto 0):= “000”;
Component instantiations	**Architecture** sys_arch **of** system 1 **is** **Component** comp1 **port(** a: std_logic; b: std_logic); **end component;** **begin** U_comp1: comp1 **port map** (A,B); **End** sys_arch:
Concurrent signal assignment	Dataout <= Datain;
Sequential block	**Process** (a) **Begin** …… **End process;**
Control flow a) If b) If … else c) Case	 **If** (en = ‘1’) **then** f <= x1; **end if;** **If** (sel = ‘0’) **then** f <= x1; g <= x2; **Else** f<= x2; g <= x1; **End if;** **Case** y **is** **When** “00” => f <= state A; **When** “01” => f <= state B; **When** others => f <= state C; **End case;**

VHDL offer several advantages to the designer, which it used a standard language, already have an available tools, powerful and versatile description language, have an multiple mechanism to support design hierarchy and also allow the behavior of the required system to be destined and verified before translate into the real hardware [8].

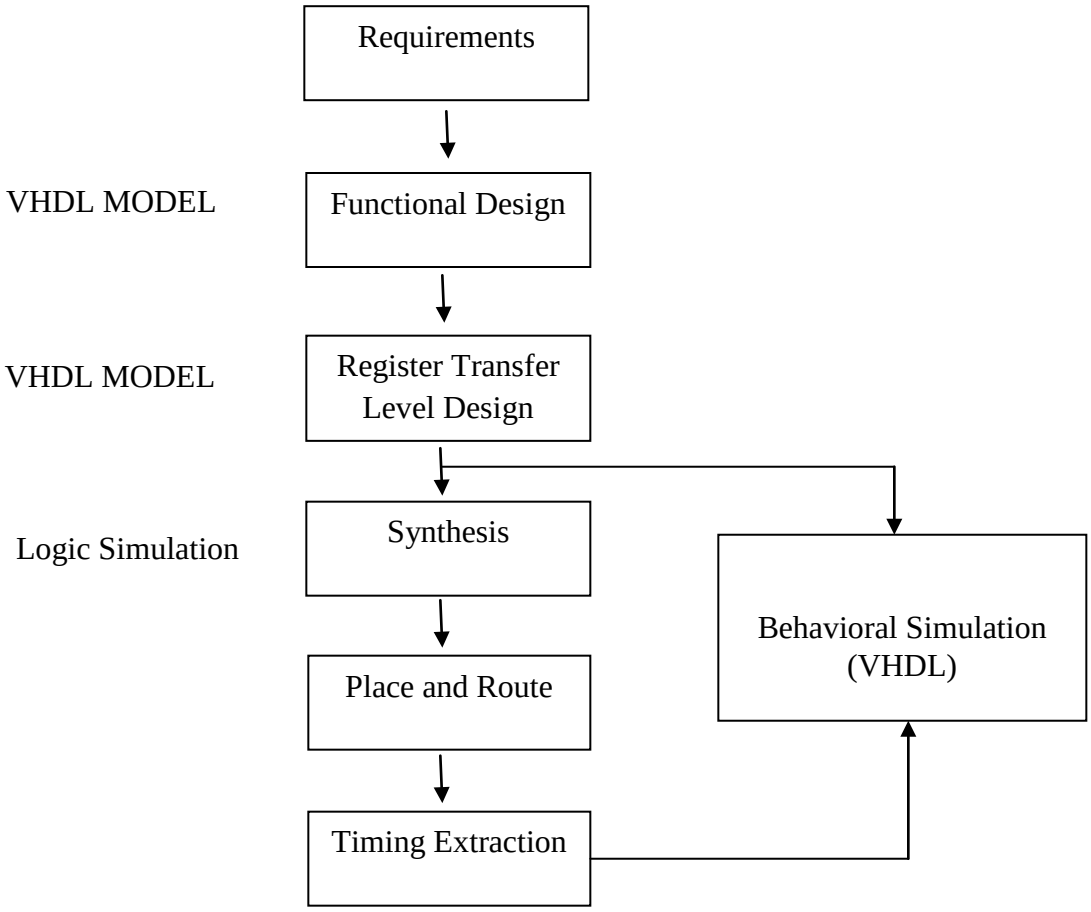


Figure 2.2: A Synthesis Design Flow

2.3.1 Basic of VHDL

Three data objects are used to represent information in VHDL programs. These are signals, variables, and constants. Signals are very common in logic circuits since they provide wires (connections) in the circuit. Input and output signals are attached to the input and output ports, respectively. Input signals are used inside the modules, whereas output signals are assigned values inside the modules. In particular, architecture assigns value to an output signal through a *signal assignment statement* denoted by the symbol " <= " [13]. For example,

$$x \leq' 1'; \quad binary\ value\ 1\ is\ assigned\ to\ x$$

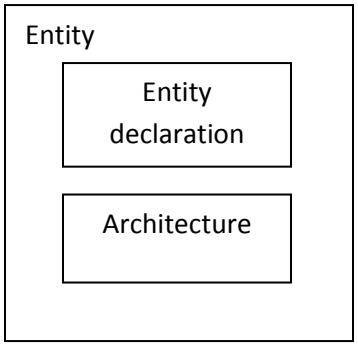
$$x \leq u + y; \ \ value\ of (u + y) is\ assigned\ to\ x$$

The values of signals can correspond to different data types, such as INTEGER, REAL, BIT, and nonnumeric sets. Data types are characterized by a name, a set of values, and a set of operations [13]. VHDL data types that are predefined include:

- TYPE INTEGER IS RANGE -2, 147,483, 647, to 2, 147, 483, 647
- TYPE REAL IS RANGE -1.0E + 38 to 1.0E + 38
- TYPE BOOLEAN IS (FALSE, TRUE)
- TYPE BITE IS ('0', '1')
- TYPE CHARACTER IS (…, 'A', 'B', 'C', …, 'a', 'b', 'c', …, '0', '1',...)

A digital circuit or system describe in VHDL is called a *design entity*, or just *entity*. It has two blocks: the *entity declaration*, which specifies the input and output connections (ports) to the hardware, and the *architecture,* which defines the behavior of

the hardware entity being designed. The general structure of a VHDL design entity modeling a digital circuit is shown in figure 2.3 [13].



Entity entity_name **is**

Port (signal_name: [**mode**] type_name);

End entity_name;

Architecture architecture_name **of** entity_name **is**

[**type; signal; constant; component** declarations]

[**attribute** specifications]

Begin

{**component** instantiation statement;}

{**concurrent assignment** statement;}

{**Process** statement;}

{**generate** statement;}

End [architecture_name];

Figure 2.3: A typical VHDL Design Entity

2.3.2 Operators in VHDL

Table 2.1: The VHDL operators (used for synthesis)

Operator Class	Operator
Miscellaneous	NOT
Multiplying	*, /
Sign	+, -
Adding	+, -, & (concatenate)
Relational	=, /=, <, <=, >, >=
Logic	AND, OR, NAND, NOR, XOR, XNOR

VHDL provides a usual range of operators useful for synthesizing logic circuits [13], which includes:

i) Logical operators,
ii) Relational operators,
iii) Arithmetic operators,
iv) Concatenate operator,
v) Shift and rotate operators

They are categorized in an unusual way, as shown in table 2.1, according to the precedence of the operators. Note that operators in the same category do not have precedence over one another. Thus, consider the VHDL expression,

$$x1\ AND\ x2\ AND\ x3\ OR\ x4$$

Since there is no precedence among any Boolean operators, this can be imply to the $x1x2x3 + x4$ expression. But to be legal and have desired meaning, this VHDL expression should be written as:

$$(x1\ AND\ x2\ AND\ x3)OR\ x4$$

2.4 MATLAB

MATLAB is software for mathematical computation, analysis, visualization, and algorithm development. Its flexible language is interactive and lets the problems, ideas, and solution be express naturally and more quickly than traditional languages. It can access more than 600 mathematical, scientific, and engineering functions [7].

Other than that the MATLAB also can create and interactively edit 2-D and 3-D plots, images, surfaces, volume and vector visualization, color and rendering, animation, camera control, sound, and lightning control [7].

2.4.1 Simulink

Simulink can be used an interactive block diagram environment with extensive predefined blocks that can be used for building graphical model of systems using the drag and crop option. This tool also capable to build and customize block diagrams of embedded systems, signal processing algorithms, mechanical systems, power systems, and wireless systems with blocksets [7].

2.4.2 Communication Blocksets

This communication blockset contain a block and convolutional coding supporting Reed-Solomon, Hamming, a posteriori probability (APP), and Viterbi decoder. The specification also includes block and convolutional interleaving libraries, channels models includes additive white Gaussian Noise (AWGN), Rayleigh, and Rician fading [7].

CHAPTER III

METHODOLOGY

3.1 Introduction

This chapter discuss about the convolution encoder design by using Very High Speed Integrated Circuit Hardware Description Language (VHDL) method to complete the project. This chapter also includes the overview of the whole method used like the block diagram building by using MATLAB software.

3.2 Work Methodology

The project planning was dividing into 3 major parts:

i. Develop the flow chart
ii. Develop the block diagram algorithm model
iii. Verify the algorithm

3.3 Flow Chart

For this project, there were some steps or stage must be done to make it happen follow the path. When the program is executed, random input data signal Bernoulli Binary Generator will be inserted into the convolution encoder to encode the data by using a VHDL language with specific algorithm equation.

The algorithm equation was created by using a parameter rate 1/2 with input, k=1 and output, n = 2, a constraint length, L=7 and the generator polynomial g1= 171 and g2=133. Then the data will be scrambled out as output data signal, so the result will be compared with the output data produced by convolution in MATLAB. These procedures are make to justify whether the algorithm that are used with VHDL language in Xilinx ISE 10.1 are compatible with the convolution that have been coded in MATLAB to produce the same output data signal with the same rate and input signal.

Hence, if the output data signals produced are not equivalent for both techniques, some error has occurred in the convolution encoder design. Then convolution encoder must be redesigned by troubleshooting from algorithm equation until the data. Output signals produced matches, then the convolution encoder are ready to be uses and proved its effectiveness according to the data sending to the network. The flowchart is as in Figure 3.1.

Flowchart

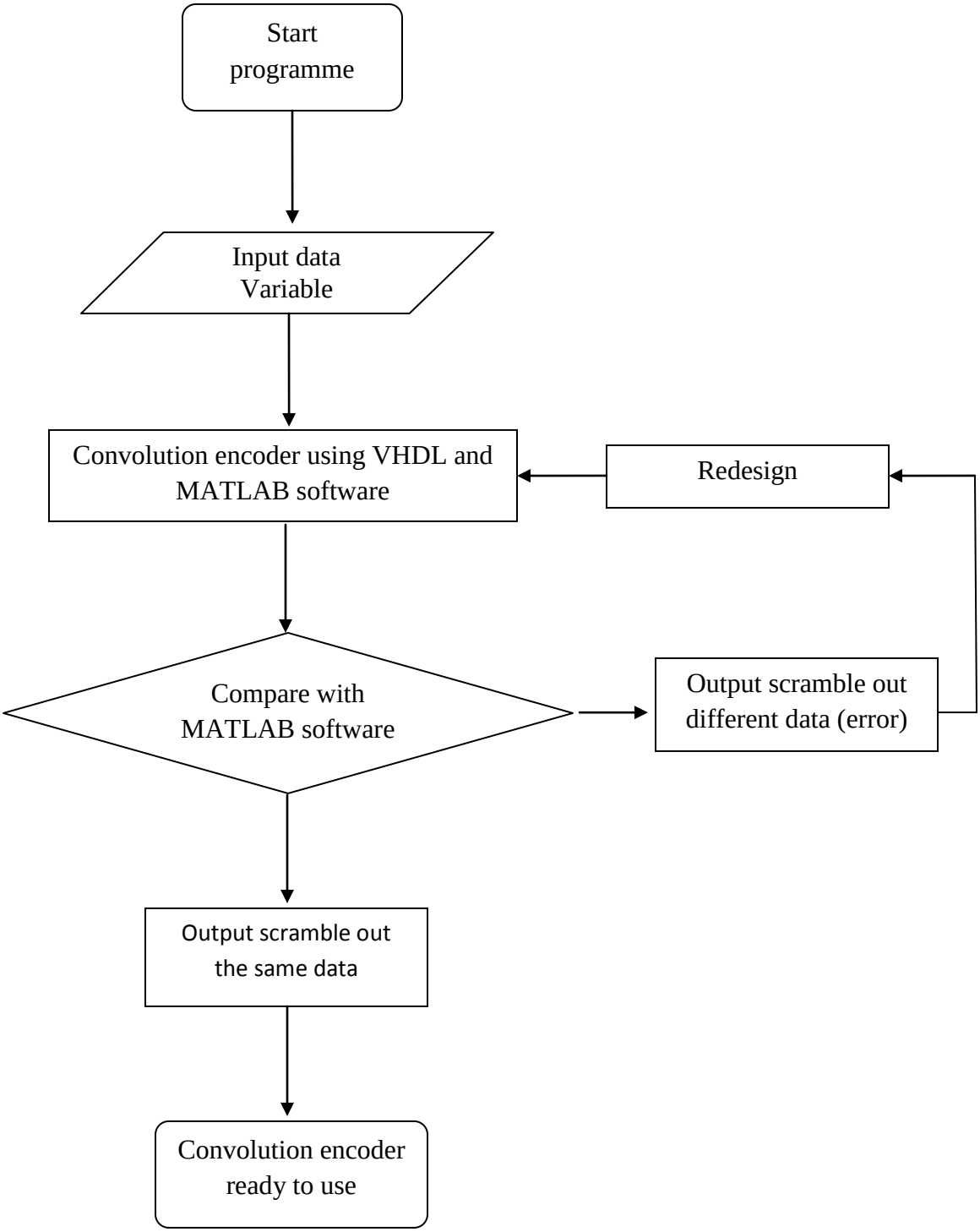


Figure 3.1: Convolution Encoder Design Flow Chart

3.4 Block Diagram

In order to get the result to be compared with the VHDL language, some stages need to be done first by using MATLAB Toolbox software. The toolbox to created the block diagram can be found in the MATLAB > Simulink Library Browser > Communication Blockset.

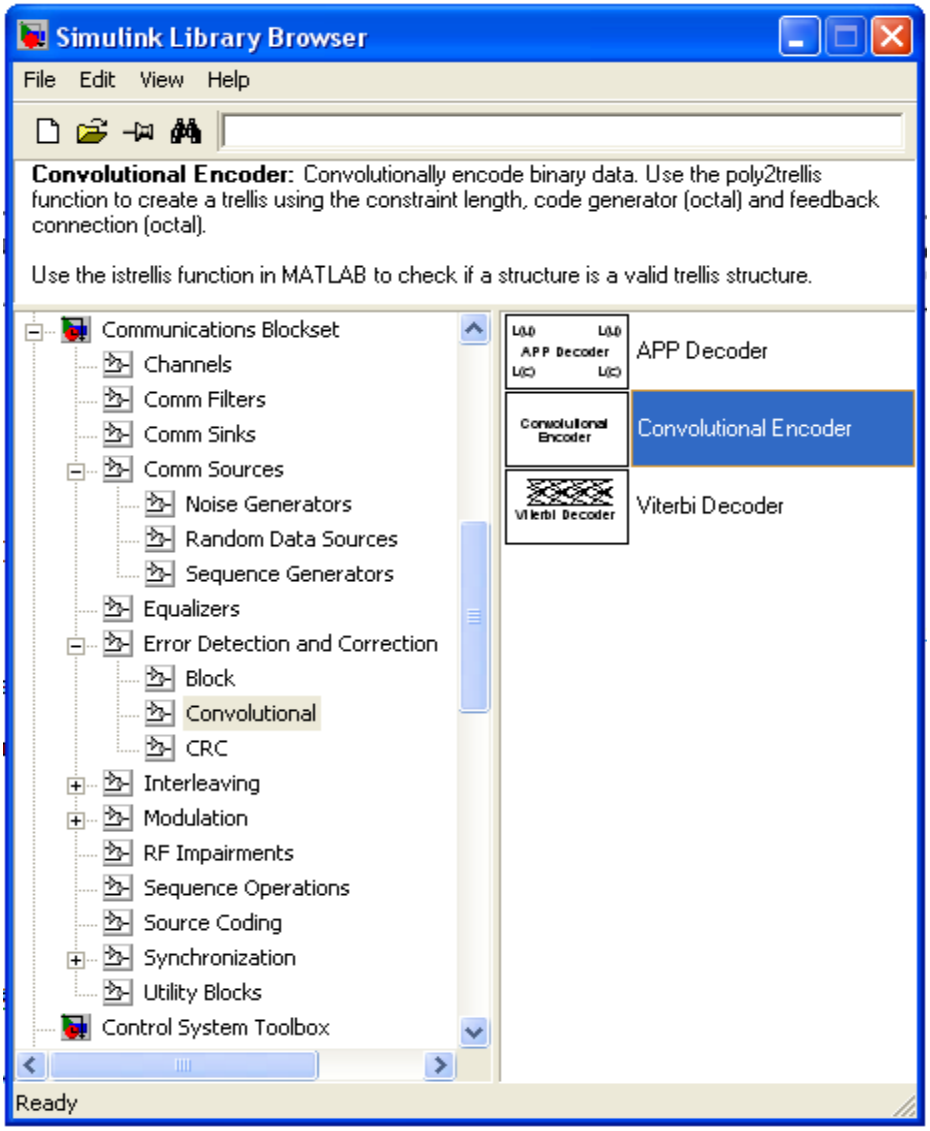


Figure 3.2: Simulink browser

3.5 Convolution encoder system design

This section described the methods used to develop and design the convolution encoder by using Very High Speed Integrated Circuit Hardware Description Language (VHDL) and also to compare the output result with convolution encoder used in MATLAB software. The methods explained in this chapter are very important procedure in order to ensure the flow of research move smoothly as planned. The methodology of this research is divided into five major sections:

i. Verification of parameter model design.
ii. Modeling of convolution encoder block in MATLAB Toolbox
iii. Develop the convolution encoder model using Simulink in MATLAB communication blockset.
iv. Develop the convolution encoder using VHDL
v. Data collection and analysis of convolution encoder design

3.6 Verification of parameter model design

For this project, modeling is the first step needed to represent the system that been focused in order to do an analysis and further work**.** This involves obtaining the parameter characteristic value of the system. With the parameter analysis, it can be used to generate and develop the algorithm that will be verified with MATLAB Simulink and VHDL at the next stage. This process is needed to determine the right rate $R = {}^{k}/_{n}$, ($k = input$ and $n = output$) that must be used to build the right algorithm.

From the parameters that have been analyzed with the convolution block in communication blockset, the parameter values shown as in table 3.0.

Table 3.0: Properties of MATLAB Convolution block

Input, k	Output, n	Generator Polynomial, g_1	Generator Polynomial, g_2	$L = (m + 1)$
1	2	1111001	1011011	7

The parameters properties:

i. Rate using, $R = {}^{1}/_{2}$
ii. Random input data = Bernoulli binary generator
iii. Convolution encoder function block parameter: Poly2trellis (7, [171 133])
iv. Generator polynomial 1 = 171
v. Generator polynomial 2 = 133
vi. Constraint length = 7

3.6.1 Bernoulli Binary Generator

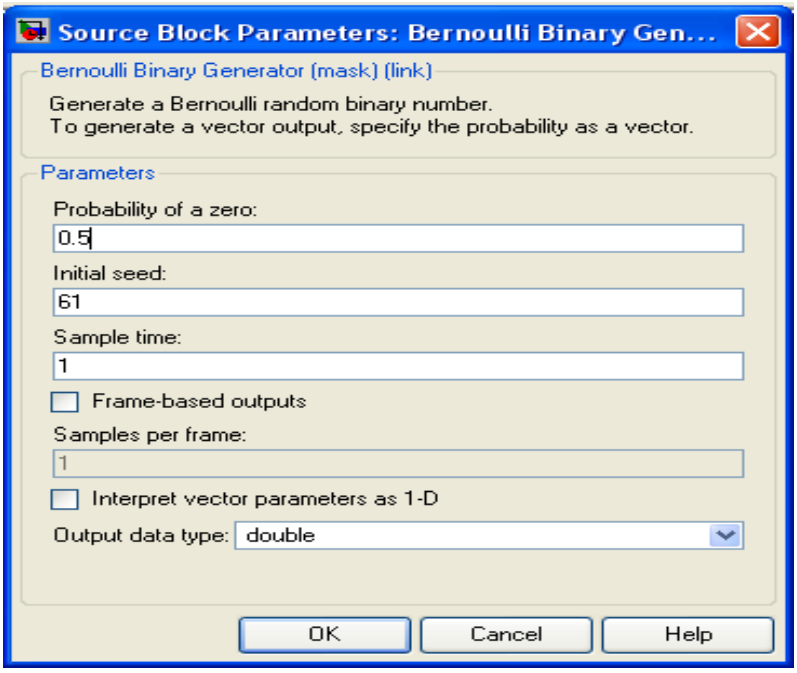


Figure 3.3: Bernoulli binary generator parameter

Figure 3.4 below show the input data for Bernoulli binary generator in graphically. These inputs will used in order to get the same data when inserted to the convolution encoder that have been developed.

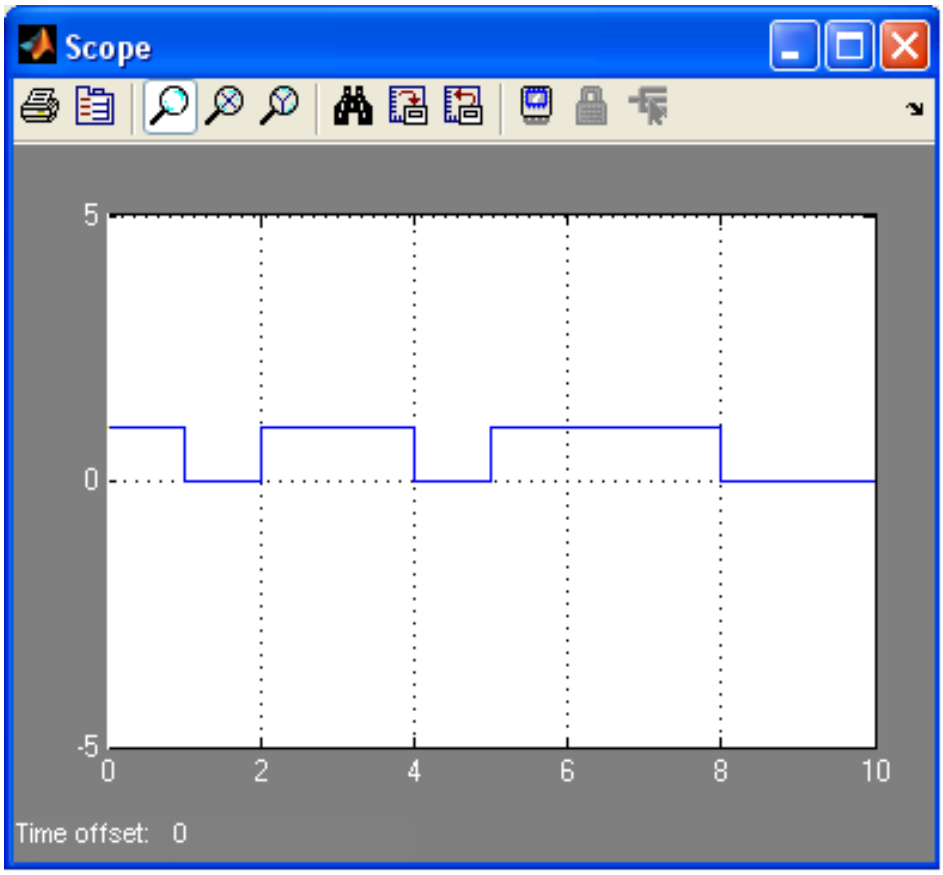


Figure 3.4: Random input Bernoulli Binary Generator

The functions of the source block parameters for Bernoulli binary generator are as below:

- Probability of a zero
  - The probability with which a zero output occurs.

- Initial seed
  - The initial seed value for the random number generator. The seed can be either a vector of the same length as the Probability of a zero parameter, or a scalar.

- Sample time
  - The period of each sample-based vector or each row of a frame-based matrix.

- Frame-based outputs
  - Determines whether the output is frame-based or sample-based. This box is active only if Interpret vector parameters as 1-D are unchecked.

- Samples per frame
  - The number of samples in each column of a frame-based output signal. This field is active only if Frame-based outputs are checked.

- Interpret vector parameters as 1-D
  - If this box is checked, then the output is a one-dimensional signal. Otherwise, the output is a two-dimensional signal. This box is active only if Frame-based outputs are unchecked.

- Output data type
  - The output type of the block can be specified as a Boolean, int8, uint8, int16, uint16, int32, uint32, single, or double. By default, the block sets this to double. Single outputs may lead to different results when compared with double outputs for the same set of parameters.

3.6.2 Poly2trellis

The poly2trellis function accepts a polynomial description of a convolutional encoder and returns the corresponding trellis structure description. The output of poly2trellis is suitable as an input to the convolution encoder and viterbi decoder functions, and as a mask parameter for the Convolutional Encoder, Viterbi Decoder, and APP Decoder blocks in the Communications Blockset [7].Trellis = poly2trellis (Constraint Length, Code Generator) performs the conversion for a rate k/n feedforward encoder. Constraint Length is a 1-by-k vector that specifies the delay for the encoder's k input bit streams.

The function parameters are as below:

- Trellis structure
  - MATLAB structure that contains the trellis description of the convolutional encoder

- Reset
  - Determines whether and under what circumstances the encoder resets to the all-zeros state before processing the input data. Choices are None, On each frame, and On nonzero Rst input. The last option causes the block to have a second input port, labeled Rst

3.6.3 Generator polynomials

Code Generator is a k-by-n matrix of octal numbers that specifies the n output connections for each of the encoder's k input bit streams [7]. If the encoder diagram has

k inputs and n outputs, then the code generator matrix is a k-by-n matrix. The element in the i^{th} row and j^{th} column indicates how the i^{th} input contributes to the j^{th} output [7].

3.6.4 Constraint length

The constraint lengths of the encoder form a vector whose length is the number of inputs in the encoder diagram. The elements of this vector indicate the number of bits stored in each shift register, including the current input bits. In the figure 3.5, the constraint length is seven. It is a scalar because the encoder has one input stream, and its value is one plus the number of shift registers for that input.

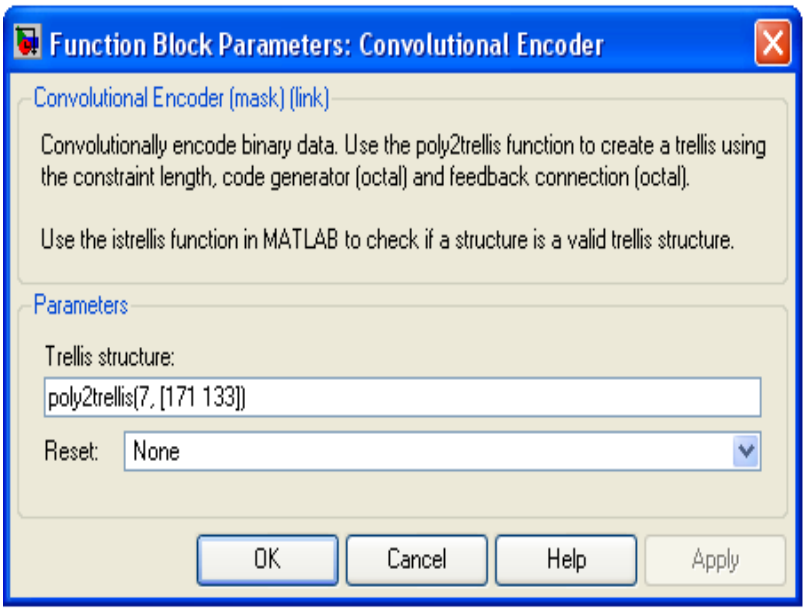


Figure 3.5: convolution encoder parameter

3.7 1st Stage: Modeling of convolution encoder block in MATLAB Toolbox

In the first stage of designing the convolution encoder model, the block diagram was build as shown in figure 3.5. The Bernoulli Binary Generator block generates random input data binary numbers using a Bernoulli distribution. The Bernoulli distribution with parameter p produces zero with probability p and one with probability 1-p. The Bernoulli distribution has mean value 1-p and variance p(1-p). The Probability of a zero parameter specifies p, and can be any real number between zero and one.

Then the input data will be process with the convolution encoder to get the scramble result. The Convolution Encoder block will encode a sequence of binary input vectors to produce a sequence of binary output vectors. This block can also process multiple symbols at a time. The block parameter used in this convolution encoder is set by using the trellis structure parameter which is poly2trellis (7, [171 133], 171) as refer to table 3.0.

The output data will be presented in graphs figure as shown by the scope block function. The Scope block will display the input with respect to simulation time. The Scope block can have multiple axes (one per port), all axes have a common time range with independent y-axes. The scope allows adjusting the amount of time and the range of input values displayed.

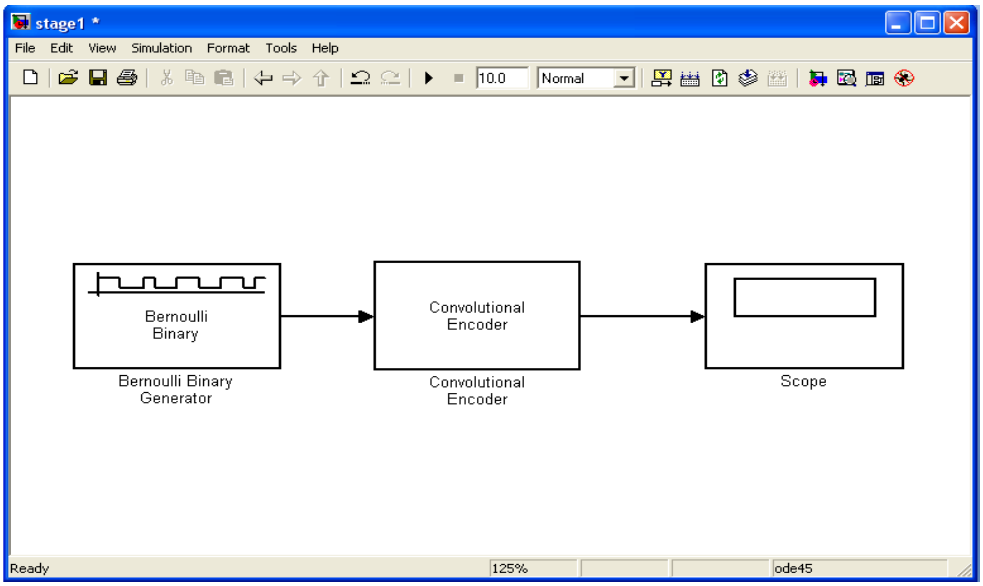


Figure 3.6: MATLAB Convolution Block

3.8 2st Stage: Develop the convolution encoder model using MATLAB Simulink

Based on the parameters verification, all the values were used to develop the next stage of convolution encoder block. The incoming data is brought into the constraint register a bit at a time, and the output bits are generated by modulo-2 addition of the required bits from the constraint register. The bits to be XORed are selected by the convolution codes with a serial stream of data are shifted into the device, and an encoded serial data stream is shifted out of the device. Refer to Figure as shown in Figure 3.7.

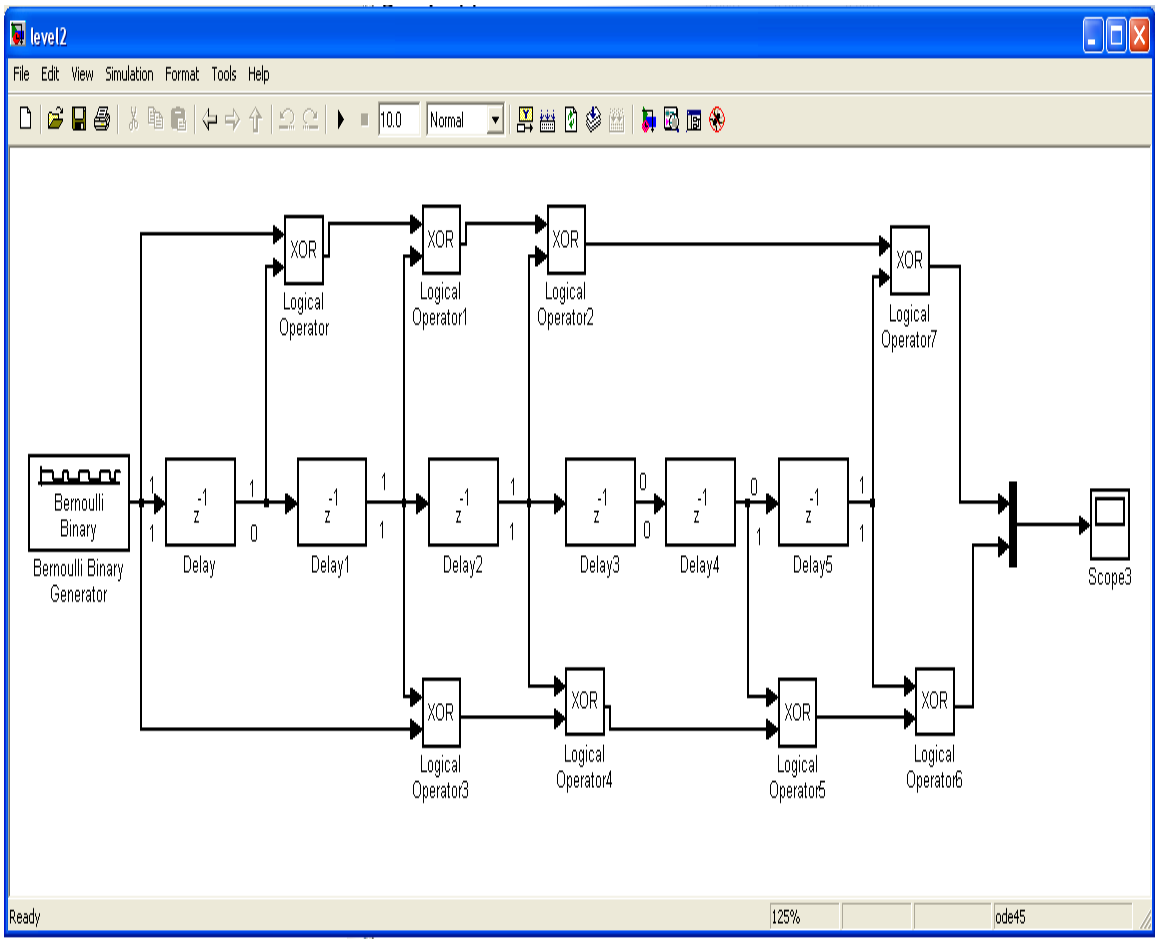


Figure 3.7: Convolution encoder design using MATLAB

The encoder structure is described by a pair of binary numbers, having the same length as the code's constraint length, that specify the connections from the delay cells to modulo-2 addition nodes. The binary number for the upper addition node is 1111001. A 1 indicates that the bit in the corresponding delay cell (reading from left to right) is sent to the addition node, and a 0 indicates that the bit is not sent. The binary number for the lower addition node is 1011011. Converting these two binary numbers to octal gives the pair [171,133].

3.8.1 XOR Logical Operator

When a logical XOR is placed between two logical expressions, the result is TRUE if only one of the expressions on either side of the XOR is TRUE. This is a bitwise logical operation. The following truth table shows all the possible outcomes of an XOR operation:

Table 3.1: Logical XOR

A	B	C
Zero	Zero	0
Zero	Nonzero	1
Nonzero	Zero	1
Nonzero	Nonzero	0

The configurations of XOR logical operator in Simulink Tool model is shown in figure 3.8.

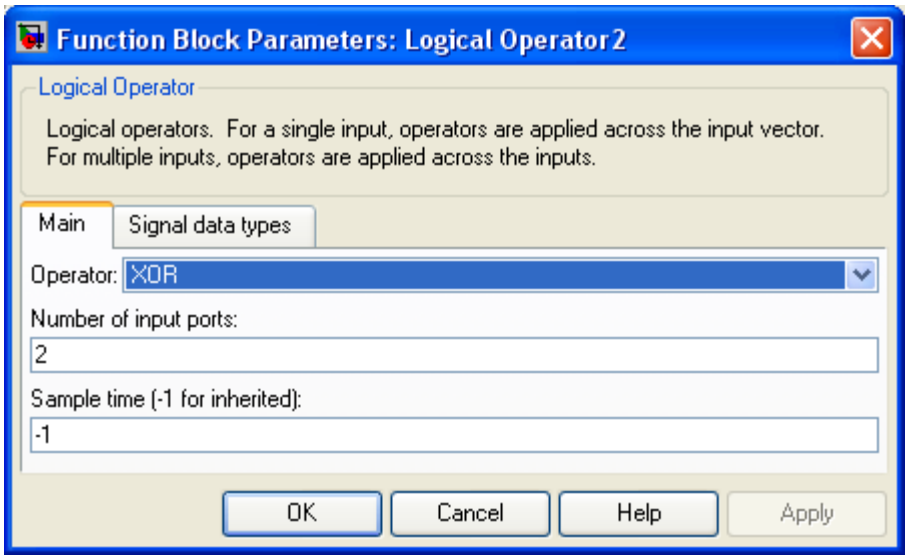


Figure 3.8: Configuration of Logical XOR

Convolutional coding is a special case of error-control coding. Unlike a block coder, a convolutional coder is not a memoryless device. Even though a convolutional coder accepts a fixed number of message symbols and produces a fixed number of code symbols, its computations depend not only on the current set of input symbols but on some of the previous input symbols.

3.9 3st Stage: Develop the convolution encoder model using VHDL

To developed the top level VHDL file that completes a hierarchical design, the design must uses a package that contain all the component declarations that has completed as part of the architecture.

3.9.1 XOR Architecture

The architecture for the Exclusive OR gates connections has shown in Figure 3.9 below. The coding can be referring in appendix B.

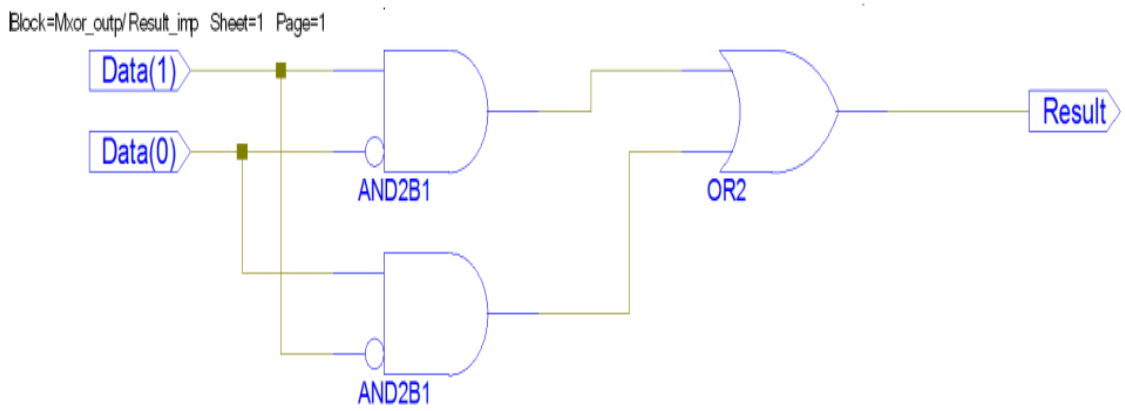


Figure 3.9: Exclusive OR gates

The test bench waveforms of the exclusive OR were done in a simulation as shown in figure 3.9.1.

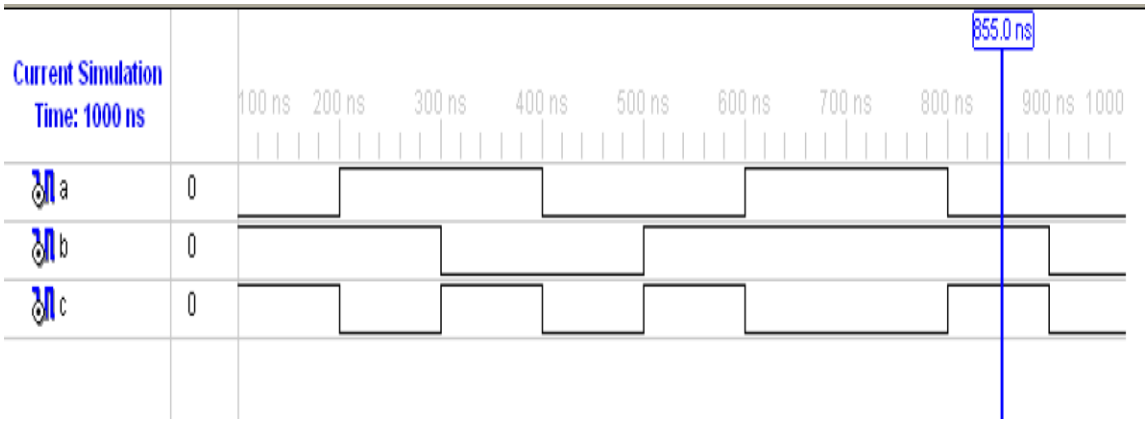


Figure 3.9.1: XOR result simulation

3.9.2 D Flip - Flop Architecture

The architecture for the D flip – flop connections has shown in Figure 3.9 below. The coding can be referred in appendix B.

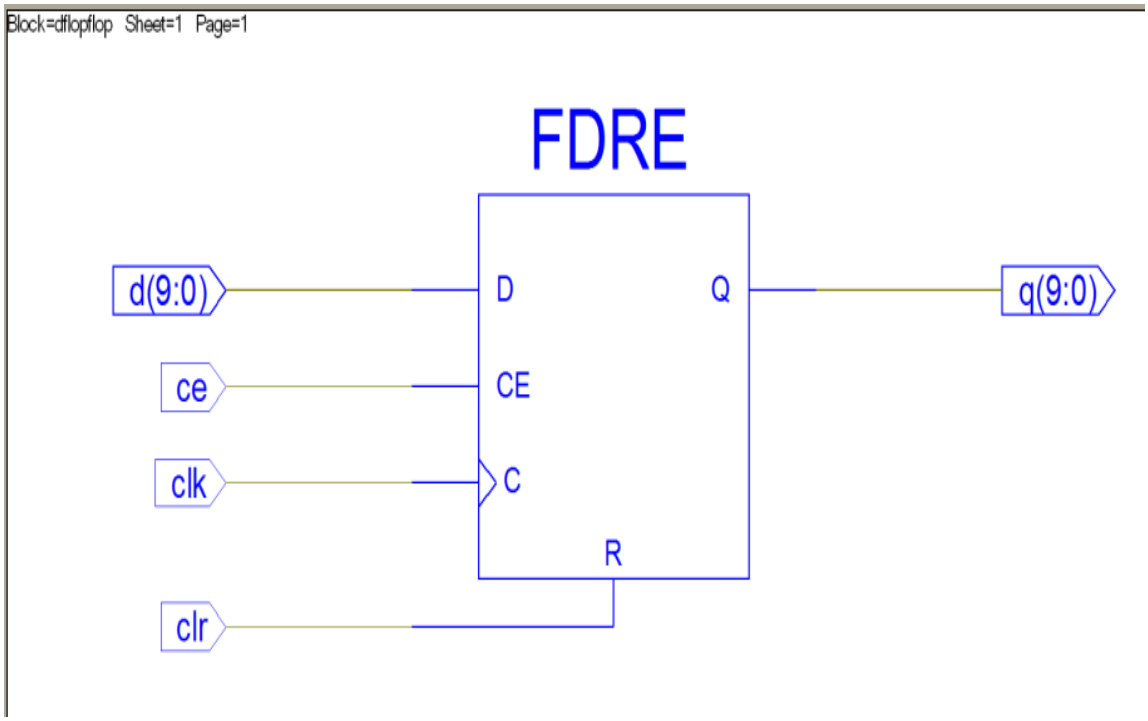


Figure 3.9.2: D Flip – Flop Architecture

The test bench waveforms of the exclusive OR were done in a simulation as shown in figure 3.9.1.

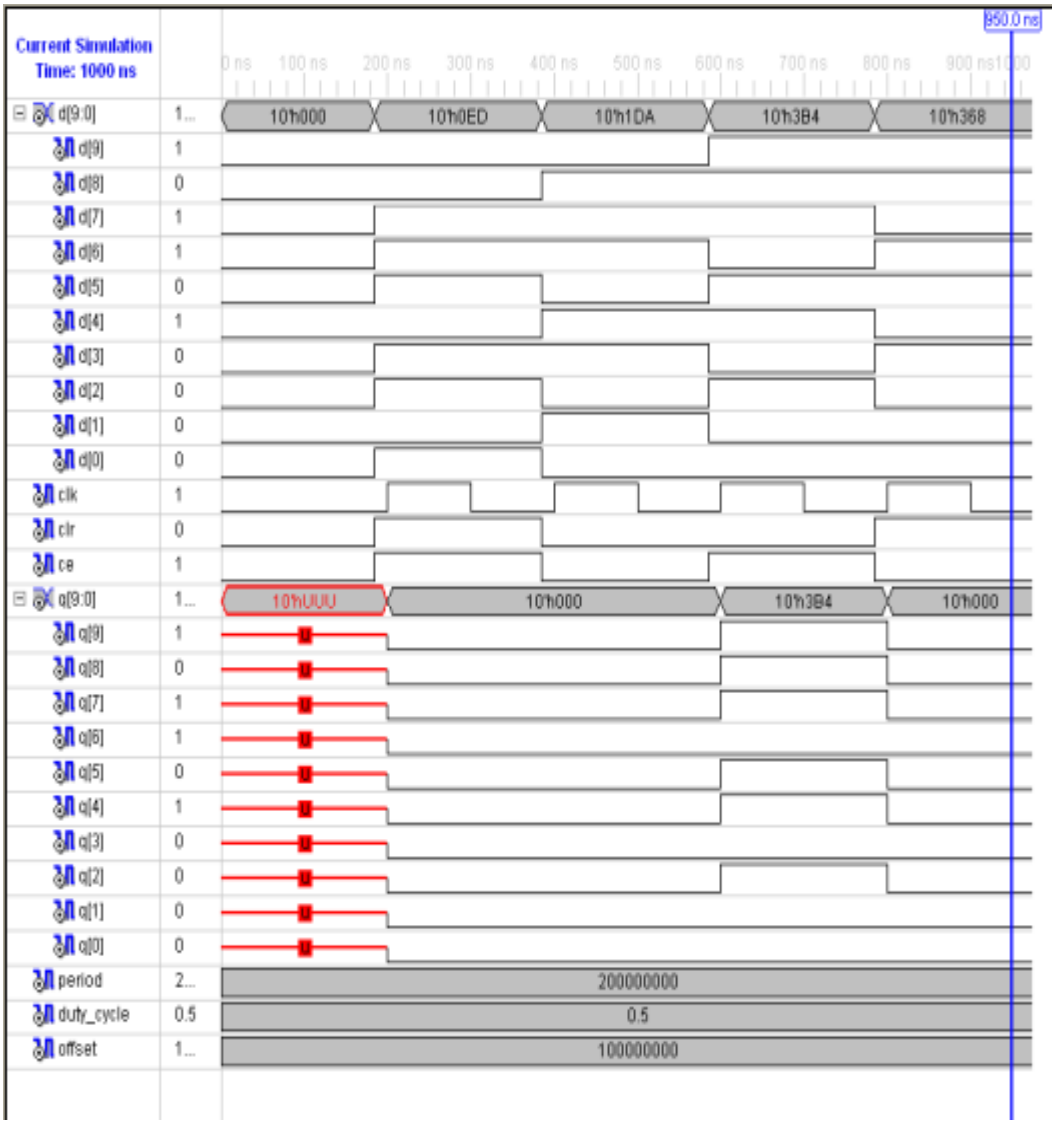


Figure 3.9.3: D Flip – Flop simulation

The D flip-flop was combined with the XOR gate architecture in the TOP level module to get the whole combination architecture as shown in figure 3.9.1. The coding can be referred in appendix B.

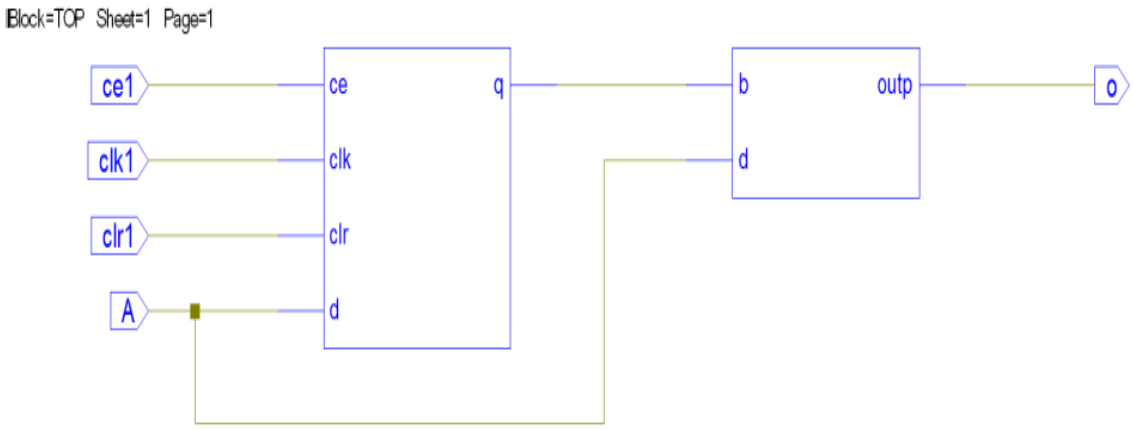


Figure 3.9.4: Top Level Combination D flip-flop and XOR gates

CHAPTER IV

RESULTS AND ANALYSIS

4.1 Introduction

This chapter will discuss all the results that have been simulated. Each of the results were presented and analyzed. The simulation studies were done by using MATLAB 7.1 and Xilinx ISE 10.1.

4.2 Result using Convolution Encoder in MATLAB Toolbox

As refer from the previous chapter, the random input data taken from Bernoulli Binary Generator (1011011100) was used with poly2trellis (7, [171 133]) function to construct the output response value from the Convolution Encoder toolbox block.

As refer to Figure 4.1, after the inputs have been inserted, the output response values were shown by 1101010101.

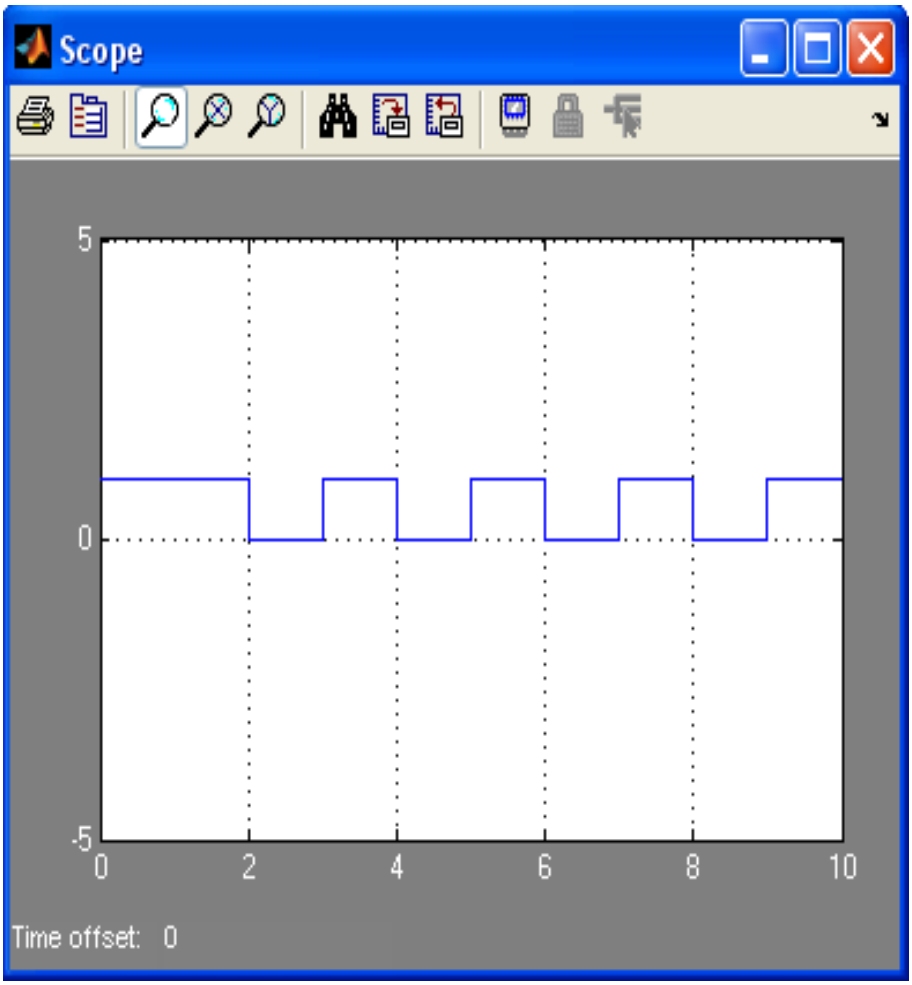


Figure 4.1: Output response for generator polynomial [171]

As refer to Figure 4.2, after the inputs have been inserted, the output response values were shown by 1101010101.

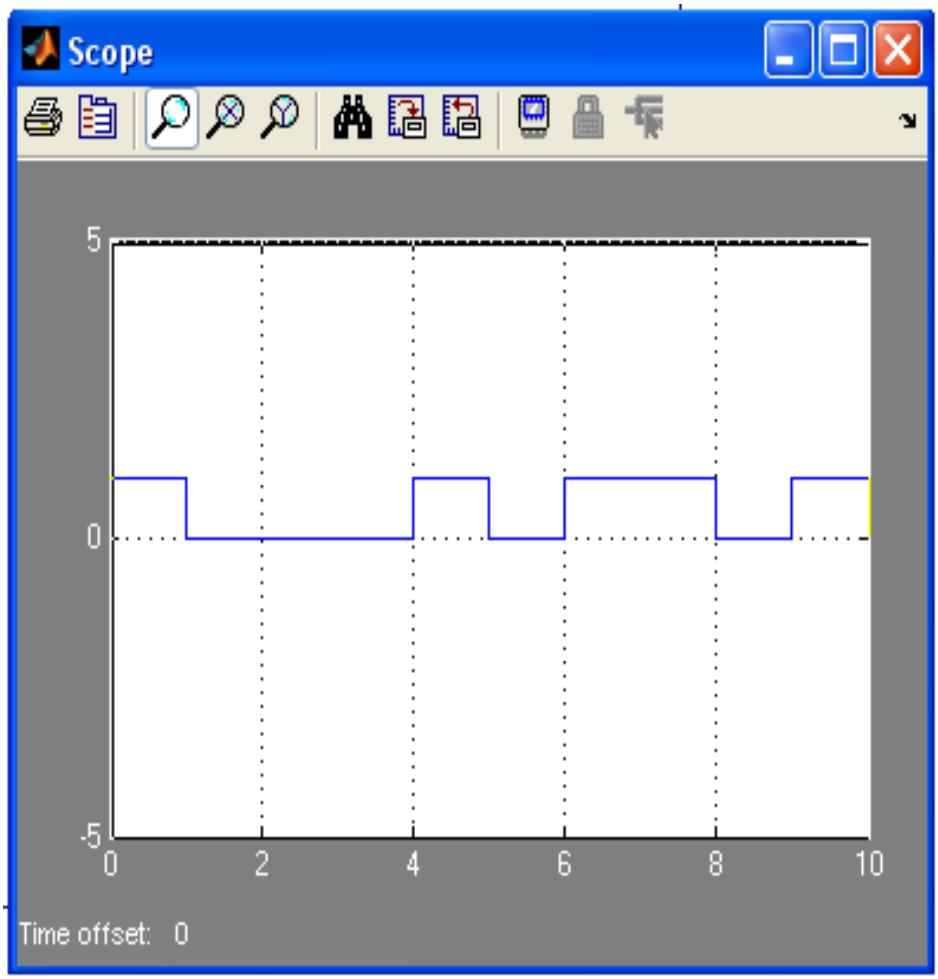


Figure 4.2: Output response for generator polynomial [133]

Figure 4.3 shows the output response of a MATLAB toolbox blocks build with poly2trellis (7, [171 133]) and constant rate 1/2. The final result was a combination of the two output response as get in Figure 4.1 and Figure 4.2. The two graphs were multiplexed to get the combination of the output response.

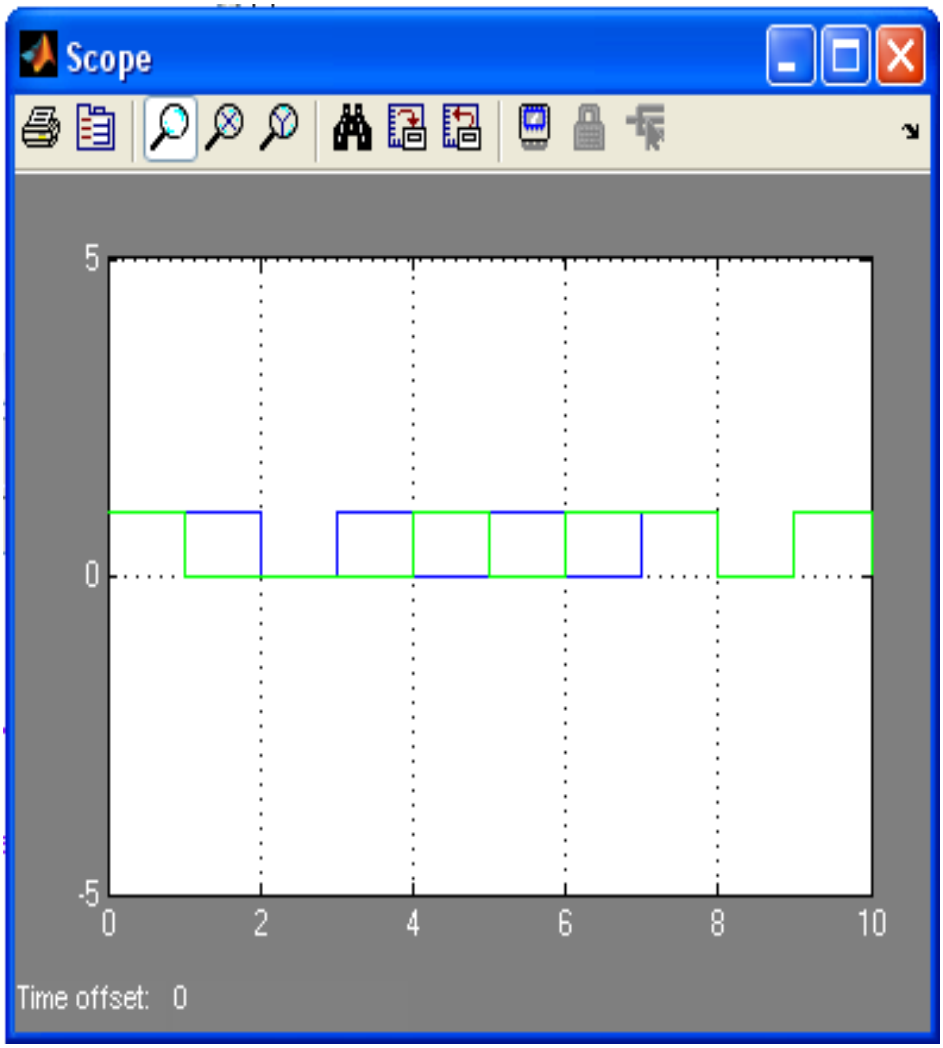


Figure 4.3: Combination of output response G1 and G2

- Blue line = generator polynomial 1, g1= [171]
- Green line = generator polynomial 2, g2 = [133]

.

4.3 Result of developing convolution encoder model using MATLAB Simulink

By refer from the previous chapter, the incoming data taken from Bernoulli Binary Generator which is 1011011100 (refer Figure 3.4) was brought into the constraint register a bit at a time, and the output bits are generated by using modulo-2 addition of the required bits from the constraint register which is 1111001 and 1011011 (refer Figure 3.5). The bits to be XORed are selected by the convolution codes as shown in Figure 3.7.

As refer to Figure 4.4, after the inputs have been XORed by the constraint register 1111001, the output response values were shown by 1101010101

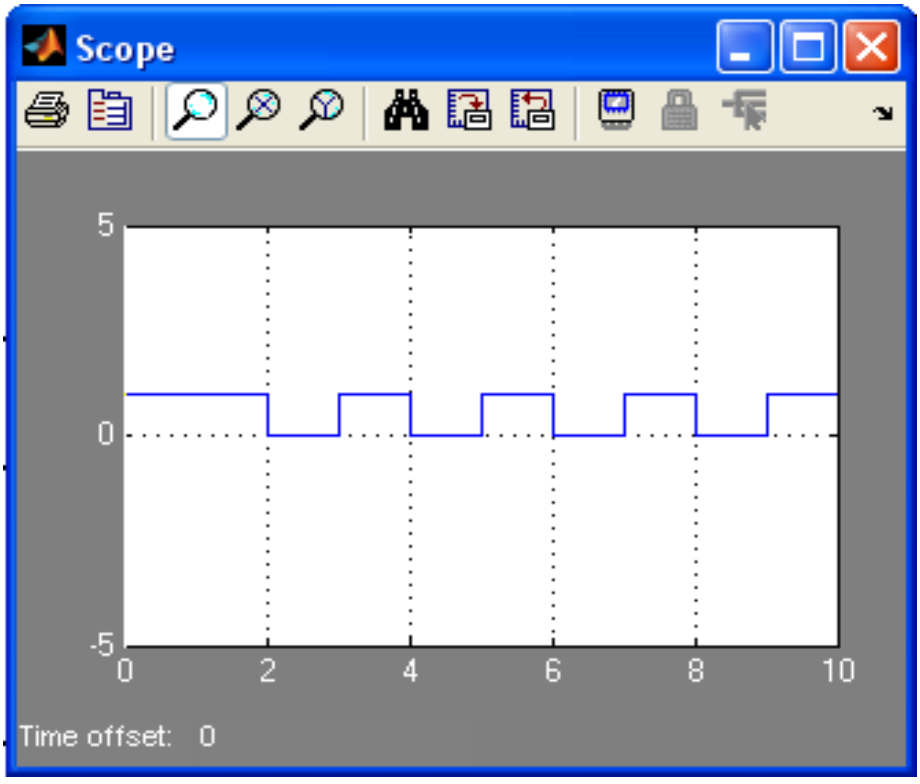


Figure 4.4: Scope 1 (generator polynomial 1)

As refer to Figure 4.5, after the inputs have been XORed by the constraint register 1111001, the output response values were shown by 1101010101.

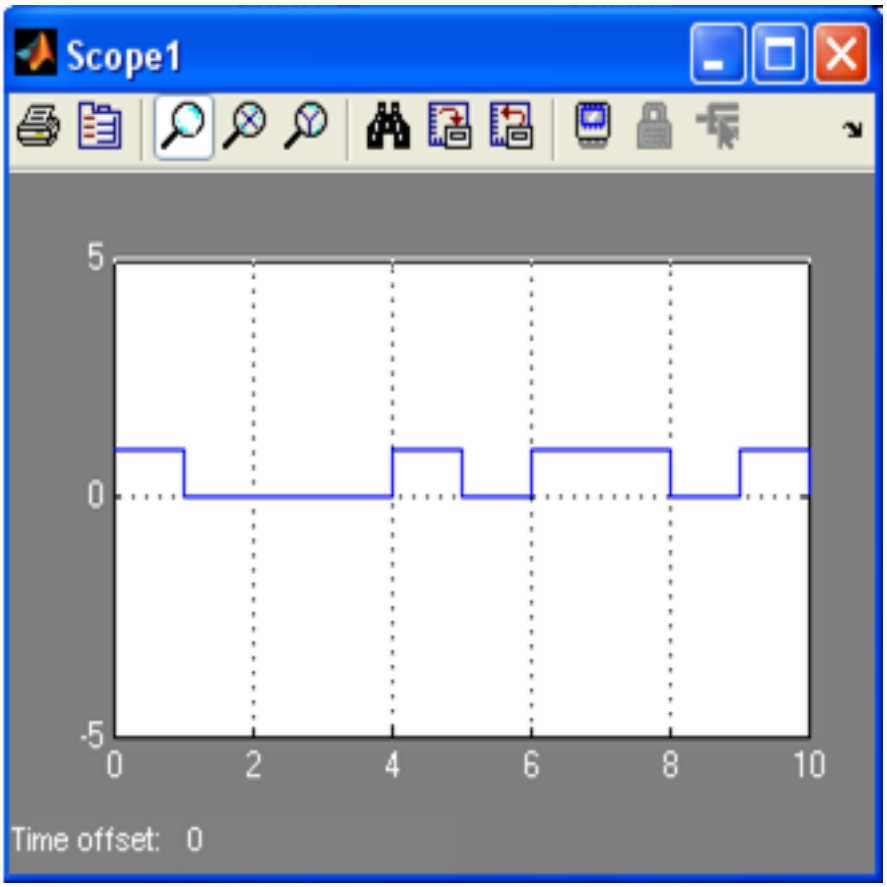


Figure 4.5 Scope 2 (generator polynomial 2)

Figure 4.6, show the combination of output response in Figure 4.4 and Figure 4.5 after it have been multiplexed together. The outputs response were encoded by using the block diagram that has been developed using MATLAB Simulink.

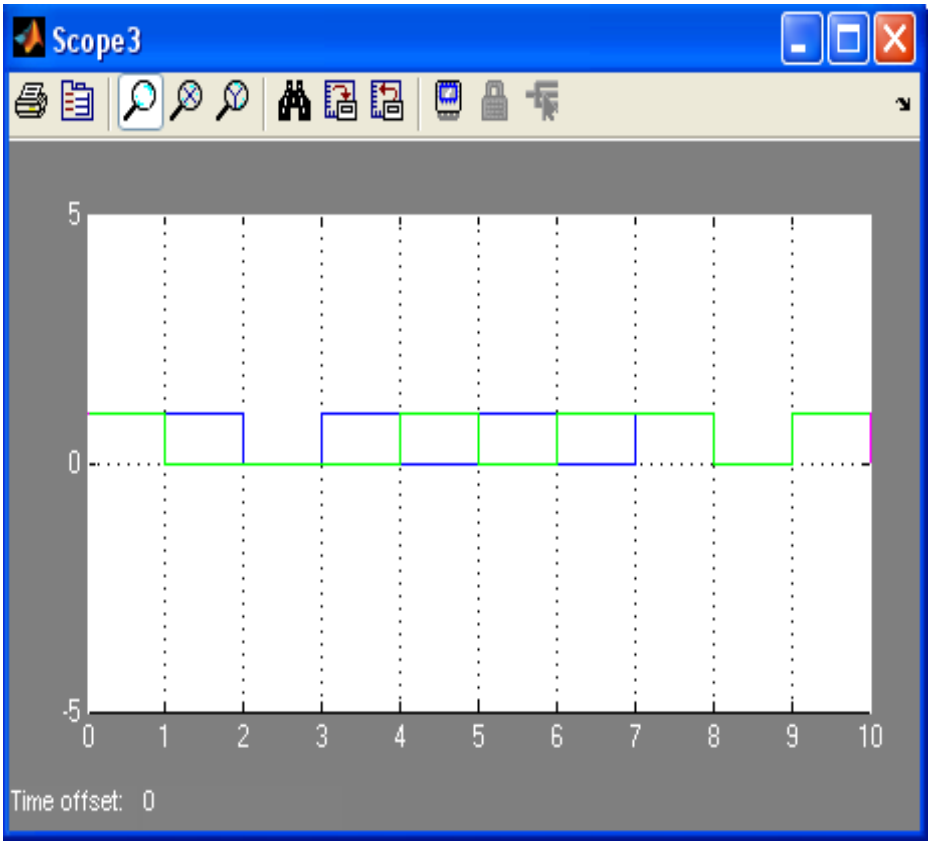


Figure 4.6: Scope 3 (combination g1and g2)

- Blue line = output response for XORed generator1polynomial 1
- Green line = output response for XORed generator polynomial 2

4.4 Result analysis

From observation on Figure 4.4, 4.5, and Figure 4.6, the similarity can be seen between this graphs and the previous technique used to get the output graph as in Figure 4.1, 4.2 and 4.3. Based on the parameters that have been verified, the new block diagram was able to be developed to get the same output response. As discussion, the similarity between these two output response from each block diagram has confirm that the

algorithm was developed by using MATLAB Simulink is obviously successful and ready to be develop again by using Very High Speed Integrated Circuit Hardware Description Language (VHDL).

The output response will be used as a reference to get the output response by using Very High Speed Integrated Circuit Hardware Description Language (VHDL) method. The two output response will be compared each other by using the convolution encoder design in Simulink and VHDL in Xilinx ISE 10.1

4.5 Result of developing convolution encoder by using VHDL

On order to developed the top level hierarchy of convolution encoder using VHDL, some part of package that contain all the component declarations that has completed as part of the architecture need to be done. Below is the part for the exclusive or gates result simulation.

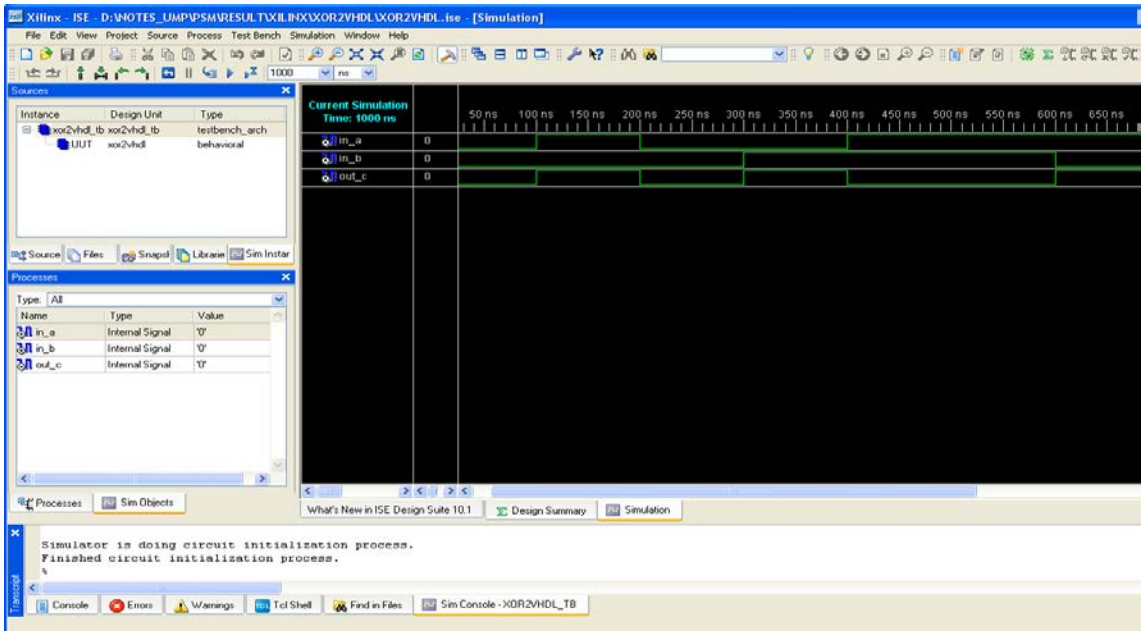


Figure 4.7: Exclusive OR result

CHAPTER V

CONCLUSION

5.1 Conclusion

The algorithm has been successfully developed to design the convolution encoder. The objectives set in this project had been half successful fulfilled such that a developing the convolution encoder using MATLAB Simulink was finished and ready to be compared. Besides, the progress to developing the convolution encoder using Very High Speed Integrated Circuit Hardware Description Language (VHDL) is still on progress. The comparison has been made so that the algorithm in MATLAB Simulink is verified to be used as the correct algorithm that can be used in VHDL.

From the simulation result obtained and discussion, some conclusion has been made. The scope of the project was achieved which is by using the rate 1/2 and constraint length 7, the algorithm are able to be develop with combination of random input data, Bernoulli binary generator block.

5.2 Limitation of the Project

In developing this project, there are some problems occur that make this project slow and did not flow smooth according to the schedule. The problems occur around the bits that need to be XORed into the constraint register. There are a few Boolean expressions that need to be configured so that the output graph of development algorithm is similar with the output response produce by MATLAB convolution encoder block.

5.3 Future work Recommendation

For future work, the constraint length can be added to be longer, because if longer constraint lengths, it will produce more powerful codes.

The encoder also can use the punctured bit set. Since the complexity of Viterbi decoding is exponential in the number of input symbols as constraint length gets large implementation complexity becomes difficult. By periodically deleting bits via a puncturing matrix these codes allow for higher rate codes, which give a higher coding gain while not suffering the implementation penalty from a large value of constraint length, *l*.

REFFERENCES

[1] A Tutorial on Convolutional Coding with Viterbi Decoding by Chip Fleming of Spectrum Applications Updated 2006-11-02

[2] Sudhakar Yalamanchili, (2005). VHDL a Starter's Guide, Pearson Prentice Hall

[3] Punctured Convolutional Coding Scheme for Multi-Carrier Multi- Antenna Wireless Systems by Christopher Lamont Taylor

[4] Jerry D.Gibson, (2005). The Communication Handbook Second Edition

[5] Frank Vahid, Roman Lysecky (2007). VHDL for Digital Design, Wiley

[6] International Scientific Conference Computer Science'2008, Software Tool for Simulating Convolutional Encoding and Decoding Used in Communication Systems, Adriana Borodzhieva, Plamen Manoilov Rousse, University "Angel Kanchev", Rousse, Bulgaria

[7] The MathWorks Family Of Products

[8] Charles H.Ruth, Jr. (1998). Digital System Design Using VHDL, PWS Publishing Company

[9] P. Elias, "Coding for Noisy Channels", IRE Conv. Record, Part 4, pp. 37-47, 1955

[10] J. Wozencraft and B. Reiffen, Sequential Decoding, MIT Press, Cambridge, Mass., 1961

[11] A. Viterbi, “Error Bounds for Convolutional and an Asymptotically Optimum Decoding Algorithm”, IEEE Transactions on Information Theory, Vol. IT-13, pp. 260-269, Apr. 1967

[12] Communication System, 4th Edition by Simon Haykin

[13] Starter’s Guide to Digital Systems VHDL and Verilog Design

APPENDIX A

Data Sheets

APPENDIX B

Program Listing

1.0 XOR GATE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
--XOR file

entity cube is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           outp : out  STD_LOGIC);
end cube;

architecture Behavioral of cube is

begin

outp <= a xor b;

end Behavioral;
```

1.2 D FLIP – FLOP

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
--D Flip – flop file

entity plop is
    Port ( d : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           clr : in  STD_LOGIC;
           ce : in  STD_LOGIC;
           q1 : out  STD_LOGIC;
                          q2 : out  STD_LOGIC);

end plop;

architecture Behavioral of plop is

begin

           process (clk)
           begin
           if (rising_edge(clk))then
                          if clr = '1' then
```

```
                q1 <= '0';
                q2 <= '0';
            elsif
                ce = '1' then
                    q1 <= d;
                    q2 <= d;
            end if;
        end if;
    end process;

end Behavioral;
```

1.3 Combination of XOR and D flip – flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity diatas is
   Port ( z : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          clr : in  STD_LOGIC;
          ce : in  STD_LOGIC;
          o : out  STD_LOGIC);
```

```
end diatas;

architecture Behavioral of diatas is
component cube is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          outp : out  STD_LOGIC);

end component;

component plop is
   Port ( d : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          clr : in  STD_LOGIC;
          ce : in  STD_LOGIC;
          q1 : out  STD_LOGIC;
q2 : out  STD_LOGIC);

end component;
signal D1: std_logic;
signal Q3: std_logic;
begin

          D1 <= z;
          u1: cube port map (a=>D1, b=>Q3, outp=>o);
          u2: plop port map (d=>z, clk=>clk, clr=>clr, ce=>ce, q2=>Q3);

end Behavioral;
```

Bernoulli Binary Generator