International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 42 • 2016

Design and Development of Convolutional Encoder in Various Environments

Bhavana Godavarthi[a] M.D. Khadir[a] C.Srihari[a] and J. Sravana[a]

[a]Department of Electronics and Communication, Institute of Aeronautical Engineering Hyderabad, India

E-mail: Bhavana.bhanu402@gmail.com, Shaikhkhadir456@gmail.com,Sri4b6@gmail.com, Sravana02@gmail.com

Abstract: Communication systems plays a major role in our lives especially in the informational revolution era, nowadays communication systems reached to extent high data rates; however it's important to ensure that data reach destination correctly, for this purpose channel coding is used by adding redundancy bits, to ensure that any error occurs during transmission will be detected and then corrected. Maintenance of the quality of data is the most important thing in communication. There are various factors that affect the quality of data when it is transferred over a communication channel like noise, fading etc. To overcome these effects channel coding schemes are introduced. In this project, one type of channel coding is described namely Convolutional Codes. There are applications in which the message bits come in serially rather than in large blocks, in which case the use of a buffer may be undesirable. In such situations, the use of convolutional coding may be the preferred method. A convolutional encoder generates redundant bits by using modulo-2 convolutions, hence the name of the coder. Implementation of the Convolutional Encoder in Various environments namely MATLAB, XILINX and Lab VIEW has been done successfully.

Keywords: *Lab View, MATLAB, XILINX, Encoder.*

1. INTRODUCTION

Communication has been one of the deepest needs of the human race throughout recorded history. It is essential to forming social unions, to educating the young, and to expressing a myriad of emotions and needs. Good communication is central to a civilized society.

In digital communication systems to improve the quality of data at output, channel coding is employed. It deals with various numbers of techniques that are being used for the improvement of performance of our communication system. It increases the information transfer rate at a fixed error rate or error rate can be reduced with a fixed information transfer rate [1]. Forward Error Correction (FEC) and Automatic Repeat Request (ARQ) are two basic errors correcting schemes, being used in communications systems[2]. Choice of these schemes is application dependent. Amongst various types of channel coding convolutional codes are the one of those, which provides better performance [3].

Convolutional codes are represented by three parameters *n, k, K*. where *K* represents the number of shift registers used in the encoding part[2][3]. In convolutional codes the input message is not divided into streams of *k* bits like block codes instead a continuous stream of data is used at the encoder's input. The coded sequence of n bits obtained after encoding not only depends on the *k* bit information message but also on the previous information bits that is transmitted. Convolutional codes are same as block codes but encoder has an additional structure. Convolutional code is a linear code and its mapping is objective [2] [3].

The convolutional codes are useful in dealing with random errors instead of burst errors. Convolutional codes can be used with block codes in order to provide good performance in case of burst errors, as block codes are good against burst errors[4][5]. Linear (*n, k*) block codes take *k* data symbols at a time and encode them into n code symbols [3]. Long data sequences are broken up into blocks of k symbols and each block is encoded independently of all others. Convolutional encoders, on the other hand, convert an entire data sequence, regardless of its length, into a single code sequence by using convolution and multiplexing operations. In general, it is convenient to assume that both the data sequences (,,…...) and the code sequences (are semi-infinite sequences and to express them in the form of a power series.

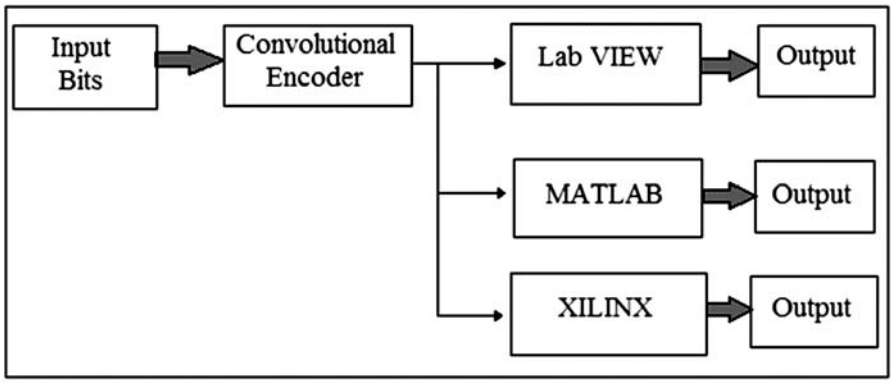


Figure 1: Design Block Diagram

2. LITERATURE SURVEY

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". Convolution codes were first introduced by Elias in 1955. He proved that redundancy could be added to an information stream through the use of linear shift register. In 1961, Wozen craft and Reiffen describe the first practical decoding algorithm for convolution codes. 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 [8] [9]. 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. Channel coding is the process of adding the redundancy information [4]. 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 [4] [5]. 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 [10]. 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* [10][12]. 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.

Convolutional codes are described by two parameters: the code rate R = *k*/*n* [4][5], 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 9 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[5].

3. DESIGN AND IMPLEMENTATION

We used two approaches in our design to the convolutional encoder; one of them is top-down approach that we can include any number of modules (*i.e*. blocks in Verilog) to write the code. Referring to figure 2.

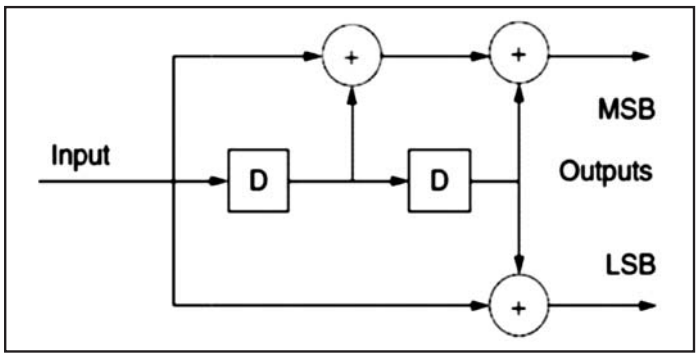


Figure 2: Convolutional encoder with rate 1/2

Technically we assumed two D flip-flops, each of which represents a block (module as well as the top-level module, which is the en coder module) in the code, binary input, and binary output with length of two bits the operation of the encoder is just XORing the binary input with the values of the **states** of the two flip flops (taking into account that each flip flop represent a module) in order to generate the output stream. The operation of the encoder is a matrix multiplication of the generator polynomials ([1 1 1], [1 0 1]) with the data (input + the states of the flip-flops) in order to generate the output stream. This approach of programming in which we used one module in the code is called a ***co-simulation*** method.

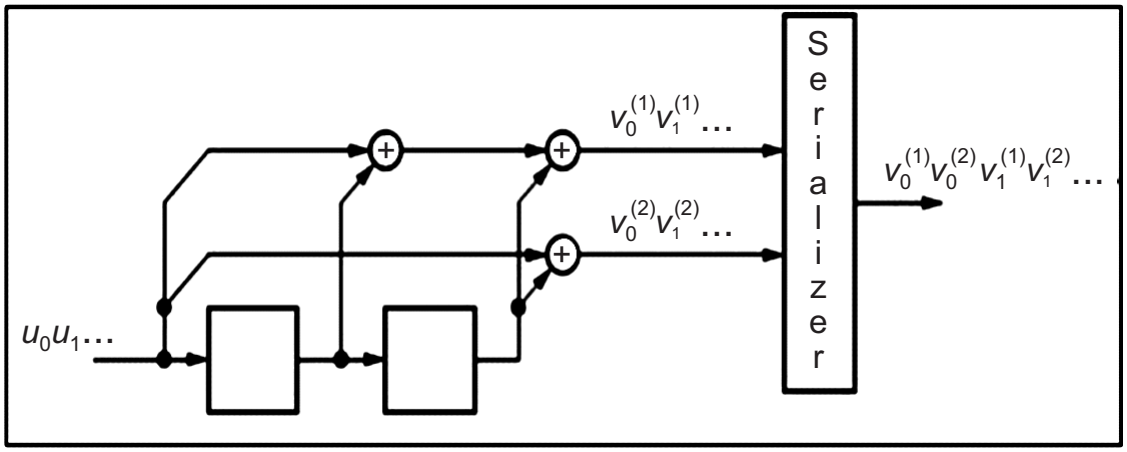


Figure 2.1: An encoder with Binary rate R = 1/2 convolutional code

The information digits $u = u_0\ u_1$ ….. are not as in the previous section separated into blocks. Instead, they form an infinite sequence that is shifted into a register, in our example, of length or memory $m = 2$. The encoder has two linear output functions. The two output sequences $v^{(1)} = v_0^{(1)}\ v_1^{(1)}$ ….. and $v^{(2)} = v_0^{(2)}\ v_1^{(2)}$

….. are interleaved by a serializer to form a single output sequence $v_0^{(1)}\ v_0^{(2)}\ v_1^{(1)}\ v_1^{(2)}$ …. That is transmitted over the channel. For each information digit that enters the encoder, two channel digits are emitted. Thus, the code rate of this encoder is R = ½ bits/channel use. Assuming that the content of the register is zero at time $t = 0$, we notice that the two output sequences can be viewed as a convolution of the input sequence and the two sequences 11100 …..and 10100….., respectively. These latter sequences specify the linear output functions; that is, they specify the encoder. The fact that the output sequences can be described by convolutions is why such codes are called convolutional codes.

3.1. Time Varying Convolutional Codes

So far we have considered only time-invariant or fixed convolutional codes, that is, convolutional codes encoded by time-invariant generator matrices. When it is too difficult to analyze the performance of a communication system using time-invariant convolutional codes, we can often obtain powerful results if we study time varying convolution codes instead.

Assuming polynomial generator matrices, we have

$$u_t = u_t G_0 + u_{t-1} G_1 + u_{t-2} G_2 + \ldots\ldots + u_{t-m} G_m$$

Where G_i, $0 \le i \le m$, is a binary $b \times c$ time invariant matrix.

In general, a rate R = b/c, binary convolutional code can be time varying.

Then it becomes

$$u_t = utG_0(t) + u_{t-1} G_1(t) + u_{t-2} G_2(t) + \ldots\ldots + u_{t-m} G_m(t)$$

Where $G_i(t)$, $i = 0, 1\ldots\ldots m$, is a binary $b \times c$ time varying matrix.

Illustrate a general time-varying polynomial convolutional encoder. As a counter part to the semi-infinite matrix G given, we have

$$G_t = \begin{pmatrix} G_0(t)G_1(t+1) & \cdots & G_m(t+m) \\ G_0(t+1)G_1(t+2) & \cdots & G_m(t+1+m) \\ \vdots & \vdots & \vdots \end{pmatrix}$$

We have the general ensemble of binary, rate R = b/c, time varying convolutional codes with generator matrices of memory in which each digits in each of the matrices $G_i(t)$ for $0 \le i \le m$ and $t = 0, 1, 2\ldots$ is chosen independently and is equally likely to be 0 and 1.

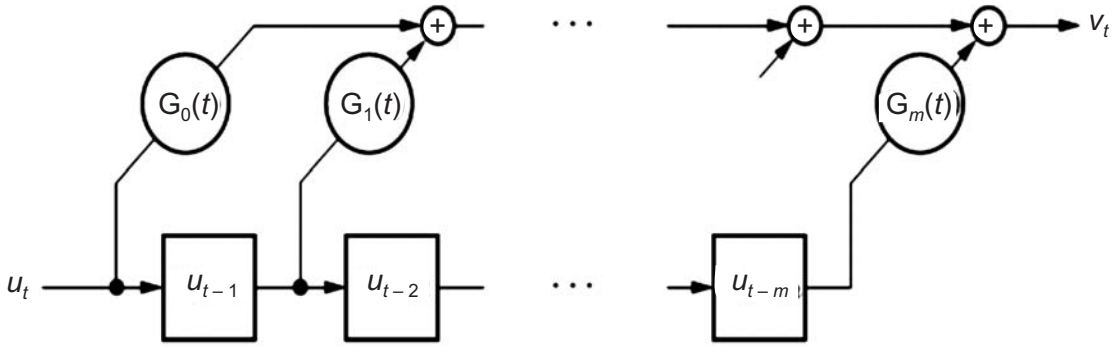


Figure 2.2: A general time-varying polynomial convolutional encoder

3.2. Circuit Description

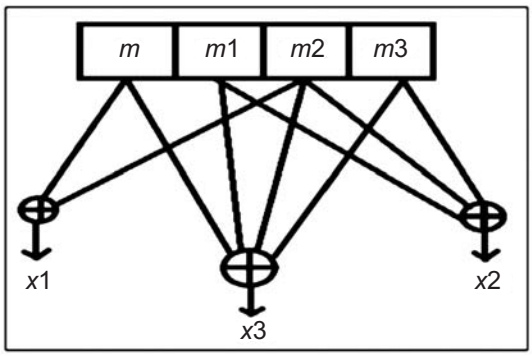


Figure 3: Encoder Circuit

Table 1
Encoder outputs

Input	*M*	*m1*	*m2*	*m3*	*x1*	*x2*	*x3*
	0	0	0	0			
1	1	0	0	0	1	0	1
0	0	1	0	0	0	1	1
0	0	0	1	0	1	1	1
1	1	0	0	1	1	1	0
0	0	1	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	1	1	0	1

Convolutional codes are commonly specified by three parameters; (*n*, *k*, *m*).

n = Number of output bits

k = Number of input bits

m = Number of memory registers

The quantity *k/n* called the code rate is a measure of the efficiency of the code. Commonly *k* and *n* parameters range from 1 to 8, *m* from 2 to 10 and the code rate from 1/8 to 7/8 except for deep space applications where code rates as low as 1/100 or even longer have been employed.

3.2.1. Convolutional Codes and their Encoders

In general, the rate R = *b* = *c*, *bc*, convolutional encoder input (information) sequence $u = : : : u_1u_0u_1u_2 : : :$, where $u_i = (u_i^{(1)} u_i^{(2)} : : : u_i^{(b)})$, and output (code) sequence $v = : : : v_1v_0v_1v_2 : : :$, where $v_j = (v_j^{(1)}v_j^{(2)} : : : v_j^{(c)})$, must start at some finite time (positive or negative) and may or may not end. It is often convenient to express them in terms of the delay operator D (D-transforms):

$$u(D) = + u_1D^1 + u_0 + u_1D + u_2D^2 + \quad (2.1)$$

$$v(D) = + v_1D^1 + v_0 + v_1D + v_2D^2 + \quad (2.2)$$

The convolutional encoder can be represented as a finite state machine (FSM) with given number of shift register stages, *e.g.* if there is a K-stage shift registers (usually D flip flops) with rate of encoder input bits to its output is 1/n and a L bit length message; the coded output sequence will have length of n (L + K) bits.

3.2.2. Encoder Design

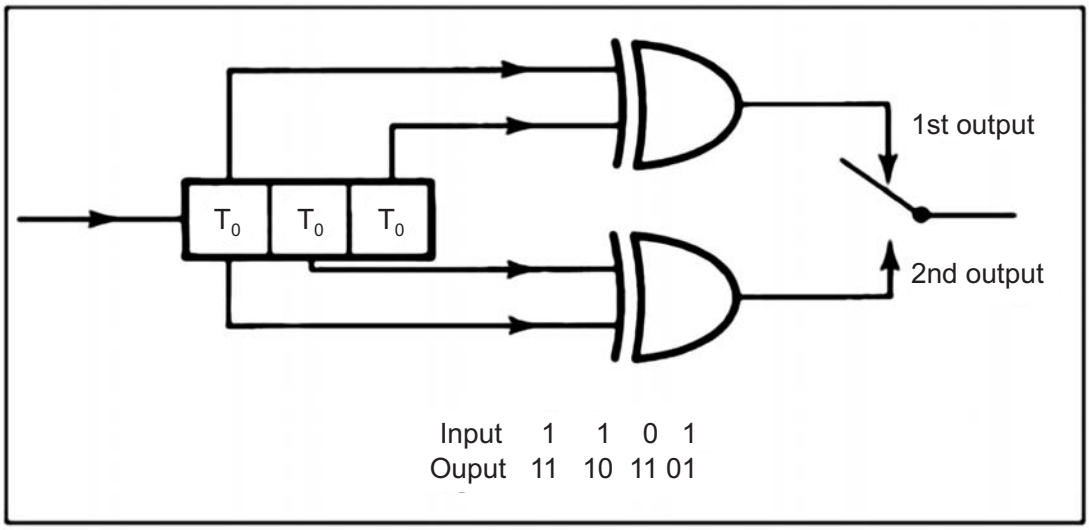


Figure 3.1: Convolutional Encoder

4. ALGORITHM & FLOWCHART

1. Initialize the shift register with zero

   $m = m1 = m2 = m3 = 0$

2. Store the incoming data bit in the first flip flop

   m = data

3. Calculate the outputs

   $x1$, $x2$, and $x3$

4. Perform shifting operation

   $m3 = m2$,

   $m2 = m1$,

   $m1 = m$

5. Repeat the above steps 2, 3, 4.

FLOWCHART

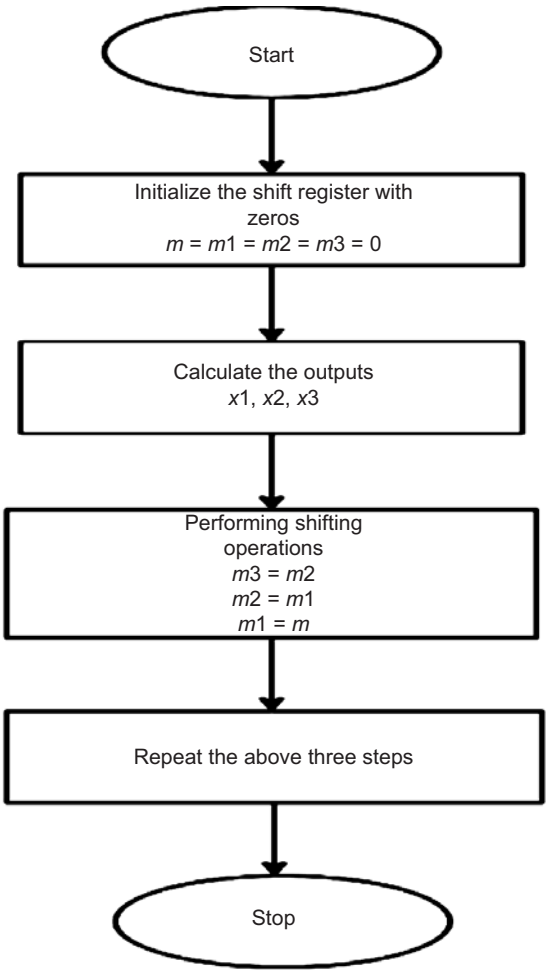


Figure 3.2: Flow chart

5. OUTPUT AND RESULTS

MATLAB

```
clc;
clear all;
close all;
I = input('enter input sequence'); %%enter input sequence
tic; %%start stopwatch timer
m = 0; %%initialise the registers with zero
m1 = 0;
m2 = 0;
m3 = 0;
for L= 1:length(I)
m = I(L);
x1 = bitxor(m, m2); %%calculate the output
```

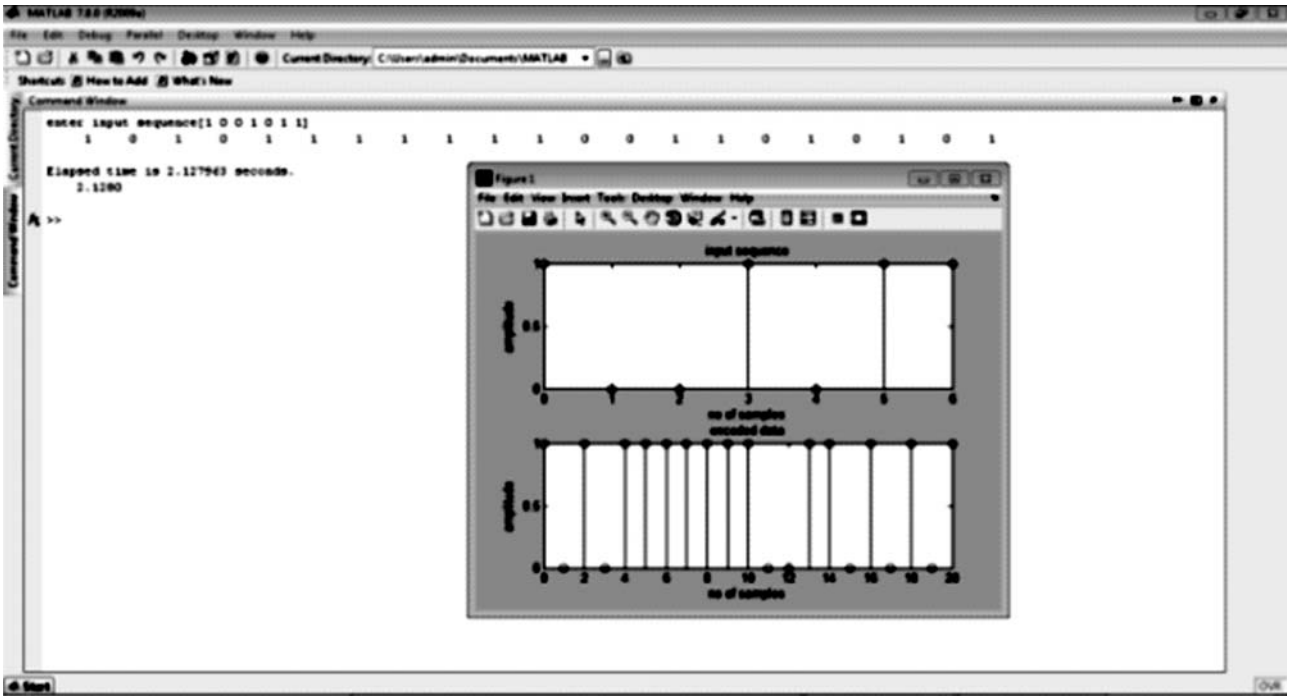


Figure 4: Matlab Output

VERILOG

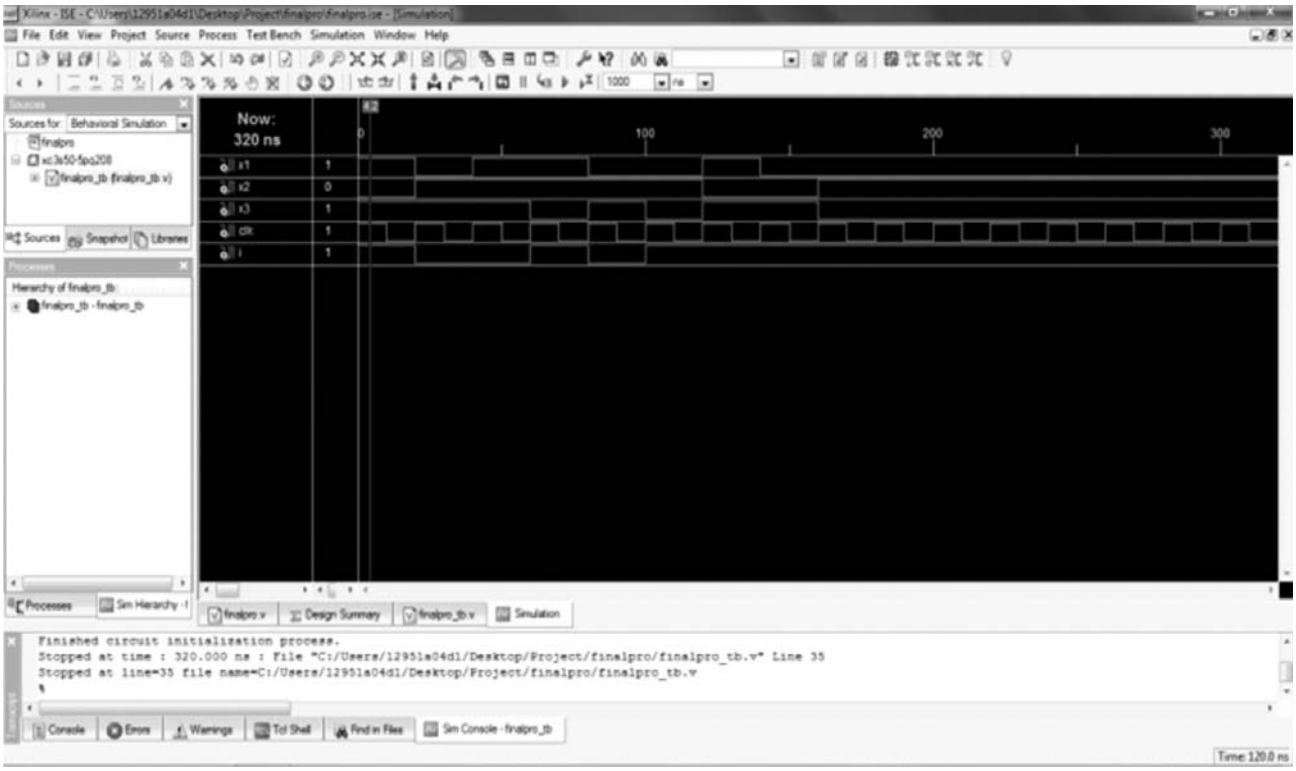


Figure 4.1: Xilinx Output

```
//XILINX//
final pro(clk, i, x1, x2, x3);
input  clk;
input  i;
output x1;
output  x2;
output x3;
reg x1, x2, x3;
reg [0:3]m = 4'b0000;
```

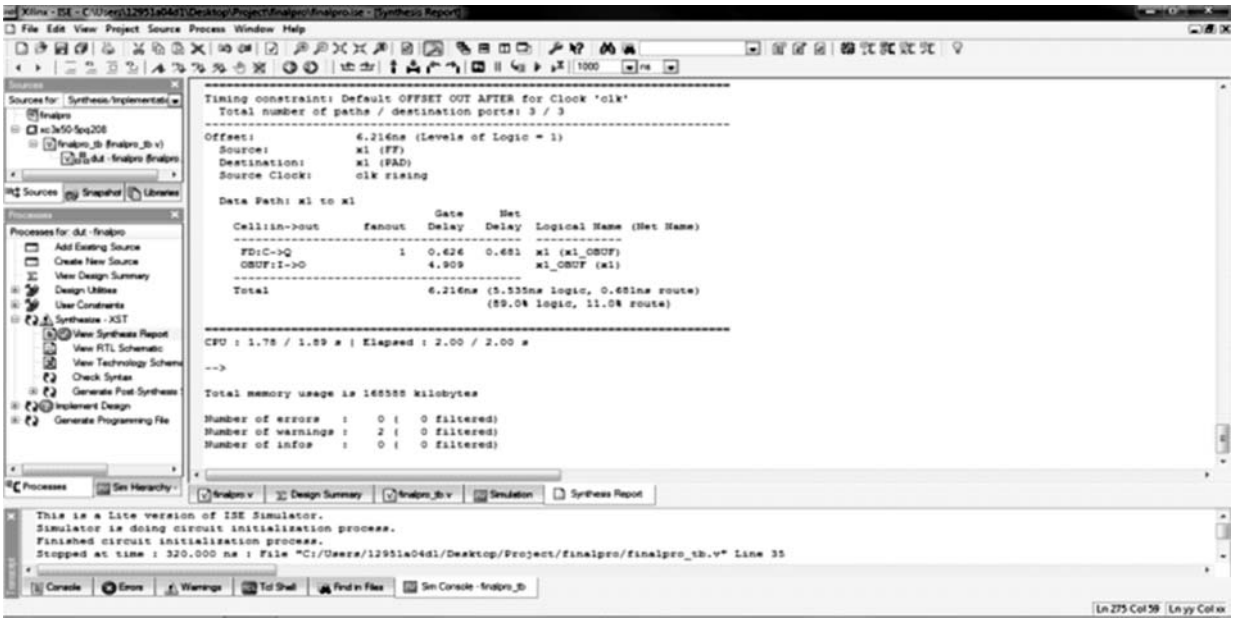
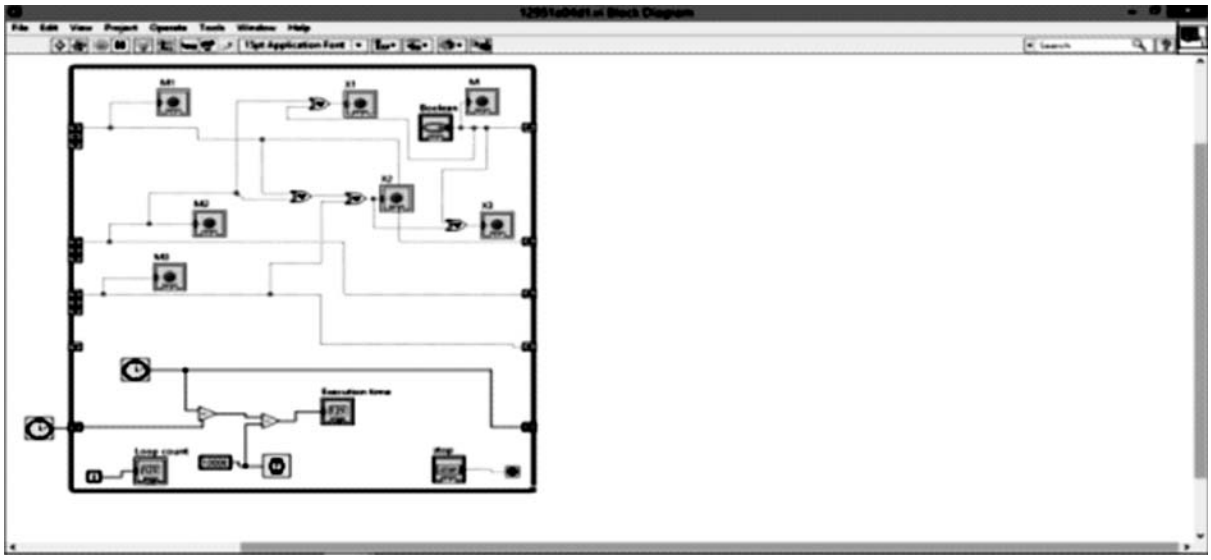


Figure 4.2: Execution time of XILINX

LABVIEW



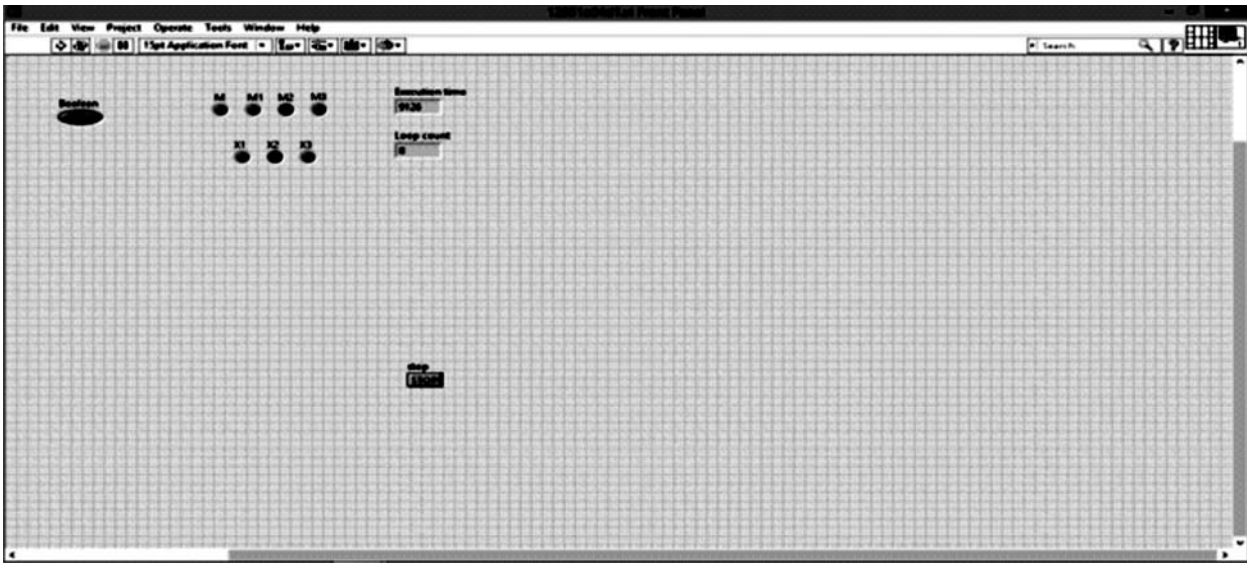


Figure 4.3: Lab view Design & Output

6. RESULT ANALYSES

The convolutional encoder is implemented in various environments. The execution time for XILINX is 6.216 nanoseconds and for MATLAB it is 2.12 seconds whereas in Lab VIEW it is 9.126 seconds.

Table 2
Execution Time Comparison

Software	*Execution Time*
XILINX	6.216ns
MATLAB	2.12s
Lab VIEW	9.126s

7. CONCLUSION

XILINX is a software tool used for synthesis and analysis of HDL designs, enabling the developer to synthesize the designs, perform timing analysis, examine RTL diagrams, and simulate a design's reaction to different stimuli. But the verilog coding in XILINX becomes little complex for larger applications as it requires to write both source code and test bench.

MATLAB is a high performance language for technical computing. Moreover it has better computation and also simulations execute fast.

Lab VIEW has fast and simple construction of GUI that provides updating of parameters and elegant presentation of results.

REFERENCES

[1] Ancheta, T. C., Jr. (1977), Bounds and techniques for linear source coding. *Ph.D. Thesis, Dept. Elec. Eng., Univ. of Notre Dame, Notre Dame, Ind.*

[2] Anderson, J. B. and Mohan, S. (1991), Source and Channel Coding—*An Algorithmic Approach.* Kluwer Academic Boston.

[3] Best, M. R., Burnashev, M. V., Levy, Y., Rabinovich, A., Fishburn, P. C., Calder-bank, A. R., and Costello, D. J., Jr. (1995), On a technique to calculate the exact performance of a convolutional code. *IEEE Trans. Inform. Theory, IT-41:441–447*.

[4] Cedervall, M. and Johannesson, R. (1989), A fast algorithm for computing distance spectrum of convolutional codes. *IEEE Trans. Inform. Theory*, IT-35:1146–1159.

[5] Engdahl, K., Lentmaier, M., Truhachev, D., and Zigangirov, K. Sh. (2000), Analytical approach to low-density convolutional codes. *Proc. IEEE Int. Symp. Inform. Theory, Sorrento*, Italy.

[6] Jordan, R., Johannesson, R., and Bossert, M. (2004), on nested convolutional codes and their applications to woven codes. *IEEE Trans. Inform. Theory, IT-50:380–384.*

[7] Richardson, T., and Urbanke, R. (2008), Modern Coding Theory. *Cambridge University Press, Cambridge, UK.*

[8] A. J. Viterbi and J. K. Omura, "Principles of Digital Communications and Coding", McGraw-Hill, NY, 1979.

[9] A. J. Viterbi, "Convolutional codes and their Performance in communication systems", IEEE Transaction Communication Technology, Vol.19, pp. 751-77, Oct. 1971.pp 260-269

[10] G. C. Clark Jr. and J. B. Cain, "Error-Correction Coding for Digital Communications", Plenum Press, NY, 1981.

[11] Shashank V. Maiya, Daniel J. Costello and Thomas E. Fuja, "Low latency Coding; Convolutional Codes versus LDPC", IEEE Transactions on Communications, Vol. 60, No. 5, May 2012

[12] J. Han Vinck, Senior member IEEE, Petr Dolezal and Young Gil Kim, "Convolutional Encoder State Estimation", *IEEE Transactions on Information Theory*, Vol. 44, No. 4, july 1998.

[13] Ba-Zhong Shen, Ara Patapoutian and Peter McEwen, "Punctured Recursive Convolutional Encoders and Their Application in Turbo Codes", *ISIT 2000, Sorrento, Italy*, June 25-30, 2000.

[14] J. B. Cain, G. C. Clark and J. M. Giest, "punctured convolutional codes and simplified maximum likelihood decoding", *IEEE transactions on Information Theory*, vol. IT 25, No. 1, PP 97-100, January 1979.

[15] Sven Riedel, "symbol by symbol MAP decoding algorithm for high rate convolutional codes that use reciprocal dual codes", *IEEE Journal on Selected Area in Communications*, Vol.16, No. 2, February 1998.

[16] Melinda, "Inner and outer decoding Performances of convolutional codes", Proceeding of the 2011 *IEEE international Conference on Space Science and Communication*, 12-13 July 2011, Penang, Malaysia.