

Hardware Implementation of Convolutional Encoder

Bhaskar Nandy,
M.Tech. Scholar,

Department of Electronics and Communication Engineering,
Heritage Institute of Technology, Kolkata ,West Bengal , India.

Abstract: Occurrence of errors in any communication system is the most intimidating factor since it makes the system vulnerable to security threats. A system should be designed to detect, prevent and amend the errors. Convolutional codes is a supreme forward error correcting (FEC) code with a least code rate.

In this paper FPGA is configured with the best performance design with least cost for 1/2 rate convolutional encoder with constraint length 3. FPGA itself will operate as the convolutional encoder showed with user input.

Keywords: Constraint length; Code Rate; Generator Polynomial; Poly2trellis structure; Convolutional Encoder; MATLAB Communication tool box; FPGA Basys2board; Verilog; Xilinx ISE design suite 14.1.

I. INTRODUCTION:

Convolutional codes were invented by Peter Elias in the year 1955[1], .Convolutional codes belong to the FEC(Forward Error Correcting) scheme. Design and Implementation of Convolutional encoding is a process of channel coding. BPSK is the best modulation scheme. Later on Jack Wozencraft recognized that the tree structure of convolutional codes[2]. Then in the year 1967, Andy Viterbi introduced Viterbi algorithm (VA) as an “asymptotically optimal” decoding algorithm for convolutional codes[3]. Ever since then, Convolutional coding with Viterbi decoding has been the prominent forward error correction technique used in Satellite Communication.

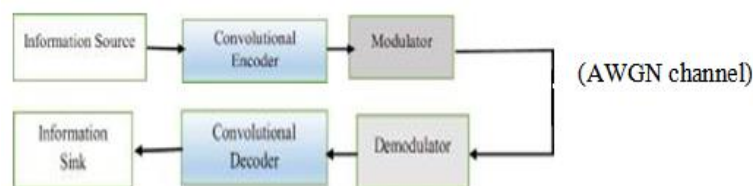


Figure1: Convolutional encoder/decoder position in a digital communication system.

A. Convolutional Encoder:

We know that (n,k) block code where code words are constructed independently of each other. Convolution code differ from block codes in the encoder output that is to say, it is not constructed from single input but by using, previous encoder input bits. Clearly memory registers are required to store the input for future use. Code rate of convolutional code, is calculated as k/n which also defines the efficiency of the encoder ,while quantity L is called the constraint length of code which is defined as no. of shifts over which a single message bit can influence the encoder outputs. calculated as $L = m+1$. where, 'm' denotes no. of memory elements used, 'n' denotes no. of outputs and 'k' denotes no. of inputs to the encoder.

Convolution encoders can be classified into two types recursive and non - recursive. or systematic and non - systematic types. We say it's a non systematic code if the output don't contain any ordered input bit stream. The block diagram of convolution encoder is shown in Fig 2. To generate the output, the encoder uses three values of the input signal, one present and two previous input values. The set of past values of input data is called a state.. Each set of outputs is generated by EX-ORing a pattern of current and shifted values of input data.

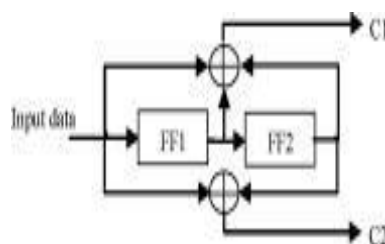


Figure 2: Block diagram of convolutional encoder

B. Generator Polynomial:

A polynomial forming the generator polynomial should be at most K degree and specifies the connections between the shift registers and the modulo-2 adders or EXOR gates..

Two generator polynomials are $g_2(x) = 1+x^2$, $g_1(x) = 1+x+x^2$ which give $g_2(x) = (101)$ and $g_1(x) = (111)$.

C. Impulse Response of the Encoder:

An encoder can be accessed on the basis of its impulse response. Impulse response is that response of the encoder when a single '1' bit or '0' bit is passed through it. Consider the contents of the register shown in table as a '1' moves through it:

Table 1: Impulse Response of Encoder

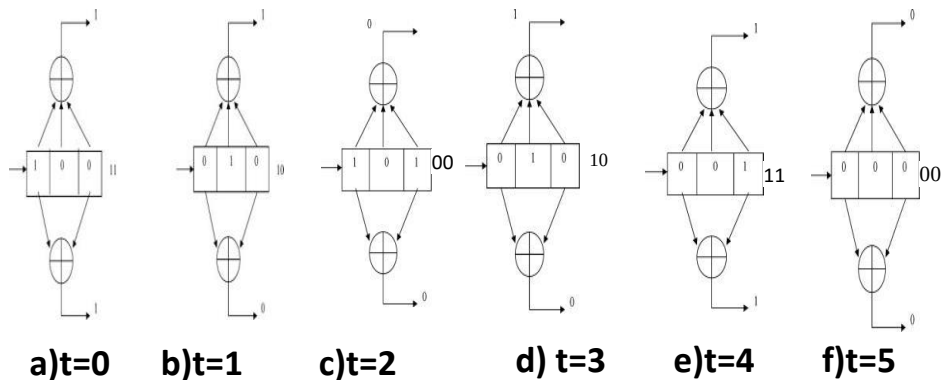
Memory Contents	Output(C1)	Output(C2)
1 0 0	1	1
0 1 0	1	0
1 0 1	0	0

Input sequence: 1 0 1 Output sequence: 111000.

The symbol '11' is known as the impulse response of the encoder.

II. ENCODER OPERATION

The encoding of the 4-bit input sequence [1 0 1 0] for the 6 sequential time periods is as shown in Figure 3.



The result of the above encoding at each of these 6 sequential time periods is as shown in Table 2.

Table 2: Result showing the working of a (2, 1, 3) convolutional encoder.

Time	Input Bits	Output Bits	Encoder Bits
0	1	1 1	00
1	0	1 0	1 0
2	1	0 0	0 1
3	0	1 0	1 0
4	0	1 1	0 1
5	0	0 0	0 0

Thus the encoded output sequence for the 4-bit input sequence [1 0 1 0] is: 11 10 00 10 .

Last two '0' bits at inputs are called " flush bits" to reset the memory states of encoder to 0.

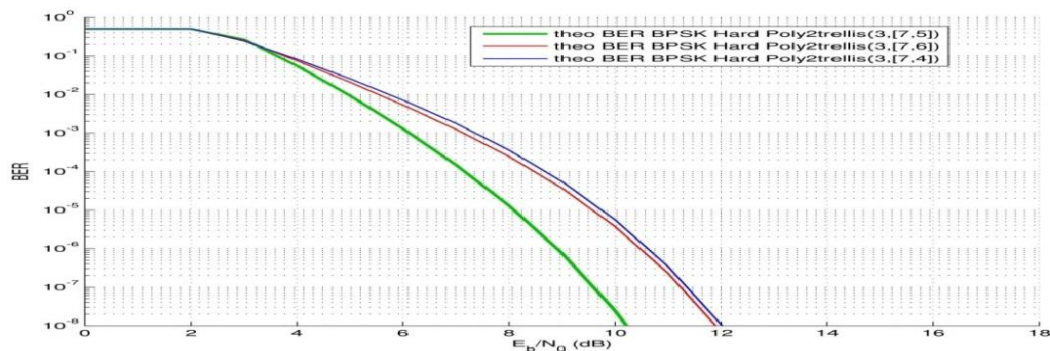


Figure 4: BER analysis for Different Circuit of 1/2 Code Rate

Out of these circuits, for a constant BER, the circuit with poly2trellis (3, [7, 5]) has the best performance. This indicates we need to have an encoder circuit with three memory elements and $g_1 = (111)$ and $g_0 = (101)$ as their impulse response.

III. ENCODER DESIGN

In our design of the encoder, first we studied and tried to implement it with PSPICE (schematic editor), then with Xilinx ISE design suite.

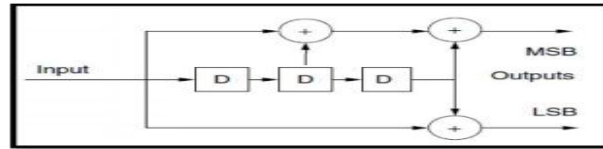


Figure 5: Block diagram for the proposed design (Non-Recursive, Non Systematic Convolutional Encoder (2, 1, 3))

Table 4: Hardware Description for the convolutional encoder

ENCODER PARAMETERS	DESCRIPTION
Rate of the encoder	1/2
Constraint length of the encoder	3
Number of inputs	1 bit input at a time
Generator polynomials	[1 1 1] [1 0 1]

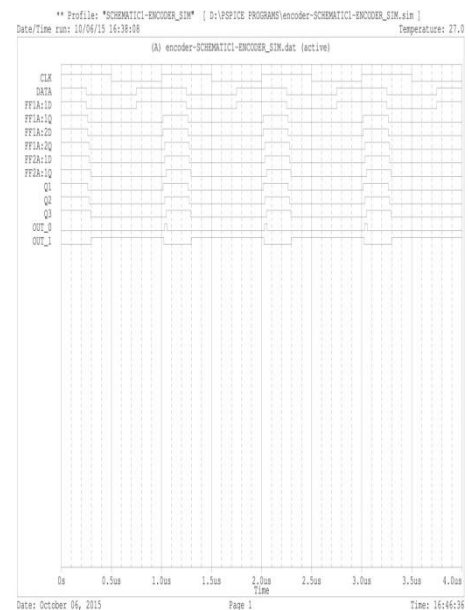
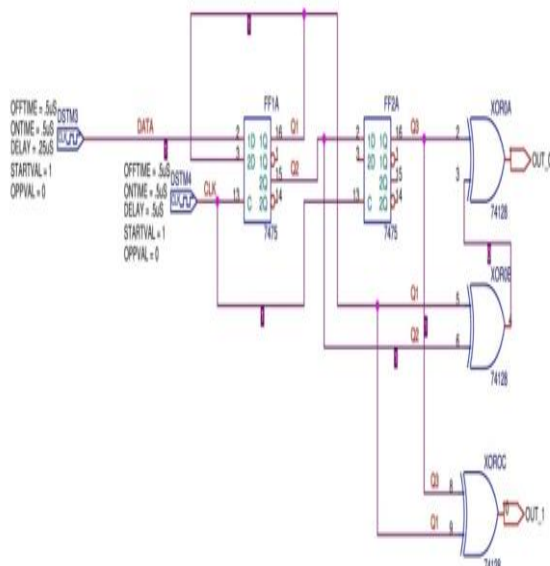


Figure 6: Encoder design using Orcad's schematic capture **Figure 7: Output generated from Schematic**

TOP LEVEL VERILOG PROGRAM in ISE DESIGN SUITE [4,5]:

timescale 1ns / 1ps

```
module encode_top_level(
```

input wire m_bit,

input wire clock,

```
input wire reset1,
```

output m_out0,

```

1  output m_out1

```

$$);$$

```
wire w1,w2,w3,w4,w5;
```

```
clk_comp u0(.clk_in(clock),.reset(reset1),.clk_out(w5));
```

```

D FF u1 (.clk(w5), .rst(reset1), .d(m bit), .q(w1));

```

```
D_FF u2 (.clk(w5), .rst(reset1), .d(w1), .q(w2));
```

```
D_FF u3 (.clk(w5), .rst(reset1), .d(w2), .q(w3));
```

```
// Slowclock module to produce 0.5hz
```

```
//Dflipflip module
```

```

xor_2 u4 (.a(w1), .b(w2), .y(w4));
xor_2 u5 (.a(w3), .b(w4), .y(m_out0));
xor_2 u6 (.a(w1), .b(w3), .y(m_out1));
endmodule

```

//XOR gate module

UCF file created for user input:

```

NET "clock" LOC = B8;
NET "m_bit" LOC = P11;
NET "reset1" LOC = A7;
NET "m_out0" LOC = M11;
NET "m_out1" LOC = M5;

```

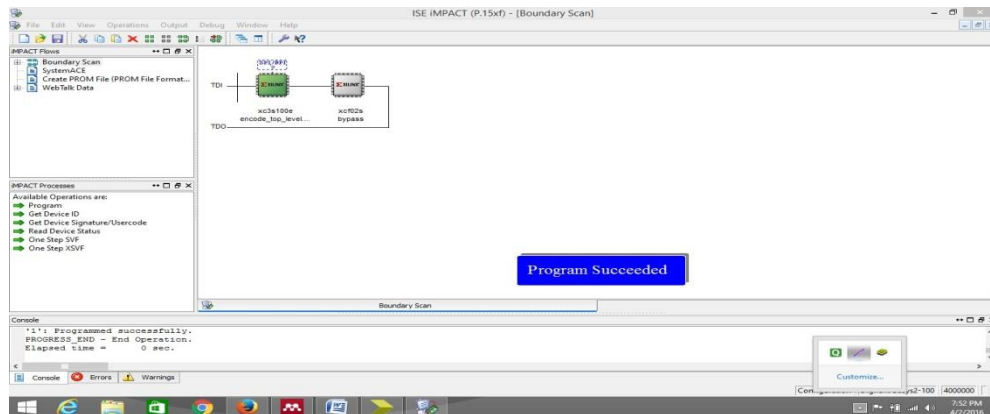


Figure 8: Program burned to FPGA kit (BASYS 2 XCS100E).

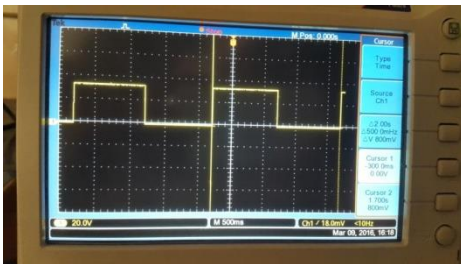


Figure 9: Slow clock tested to be 0.5 Hz in DSO

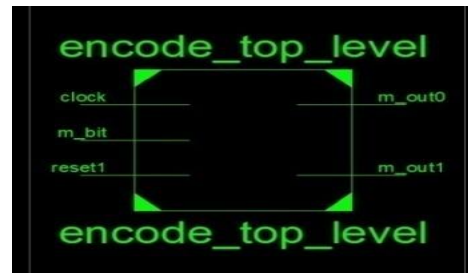


Figure10: RTL schematic (top level)

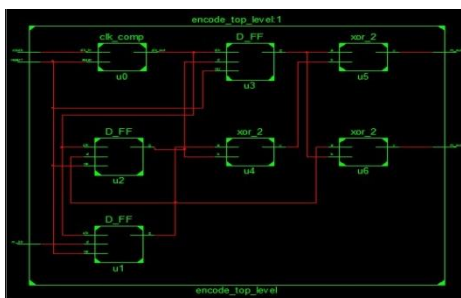


Figure11: RTL schematic of the actual circuit



Figure12: RTL schematic of the slow clock

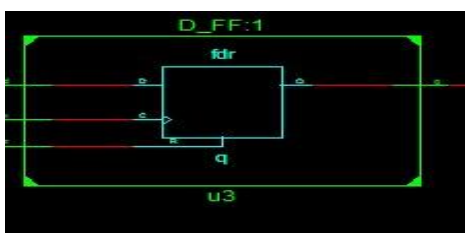


Figure13: RTL schematic of the D FlipFlop



Figure14: RTL schematic of the XOR gate

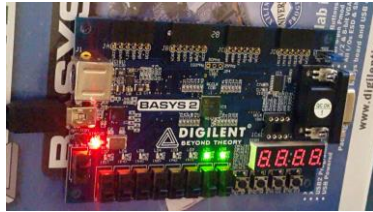


Figure 15: OUTPUT > 11 when INPUT > 1 ; t=0

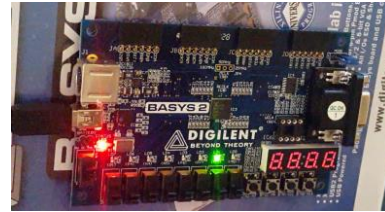


Figure 16: OUTPUT > 10 when INPUT > 0; t=1



Figure 17: OUTPUT > 00 when INPUT > 1; t=2



Figure 18: OUTPUT > 10 when INPUT > 0; t=3

If the Uncoded data stream: 1010 , obtained Encoded Output stream: 11 10 00 10 , as shown above which matches theoretically shown in section II of Encoder Operation.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	32	1,920	1%	
Number of 4 input LUTs	38	1,920	1%	
Number of occupied Slices	37	960	3%	
Number of Slices containing only related logic	37	37	100%	
Number of Slices containing unrelated logic	0	37	0%	
Total Number of 4 input LUTs	65	1,920	3%	
Number used as logic	38			
Number used as a route-thru	27			
Number of bonded I/Os	5	83	6%	
Number of BUFMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.14			

Figure 19: Device Utilization summary

IV. CONCLUSION

Keeping code rate fixed at 1/2 we had found three different circuits out of which from BER (Bit Error Rate) analysis we found the circuit with poly2trellis (3,[7,5]) has the best performance. As we have designed it in Basys 2 FPGA kit we found that the device utilization to be very less, which would enable us to fabricate a SOC (System on Chip) system with a very least cost.

REFERENCES

- [1] P. Elias, "Coding for noisy channels," IRE Conv. Rec., pt. 4, pp. 37-46, Mar. 1955. Reprinted in Key Papers in the Development of Information Theory (D. Slepian, ed.), IEEE Press, 1973; Key Papers in the Development of Coding Theory (E. R. Berlekamp, ed.), IEEE Press, 1974; The Electron and the Bit (J. V. Guttag, ed.), EECS Dept., MIT, Cambridge, MA, 2005.
- [2] J. M. Wozencraft and B. Reiffen, Sequential Decoding. Cambridge, MA: MIT Press, 1961.
- [3] Viterbi.A.J, "Convolution codes and their performance in communication systems," IEEE Transaction on Communications, vol.com-19, pp. 751 to 771, October 1971.
- [4] Samir Palnitkar, "Verilog HDL: A Guide to Digital Design And Synthesis, Second Edition", PHP.
- [5] Hardware Implementation of Viterbi Decoder for Wireless Applications Bhupendra Singh, Sanjeev Agarwal2 and Tarun Varma3, Deptt. of Electronics and Communication Engineering ,Amity School of Engineering and Technology, Noida, India International Journal of Computer Communication and Information System (IJCCIS)- Vol2. No1. ISSN: 0976-1349 July - Dec2016.