RESEARCH ARTICLE OPEN ACCESS

Implementation of Convolution Encoder and Viterbi Decoder for Constraint Length 7 and Bit Rate 1/2

Mr. Sandesh Y.M*, Mr. Kasetty Rambabu**
*(Department of Electronics & Communication, Nagarjuna College of Engineering & Technology, Bengaluru)
** (Department of Electronics & Communication, Nagarjuna College of Engineering & Technology,Bengaluru)

ABSTRACT
Convolutional codes are non blocking codes that can be designed to either error detecting or correcting. Convolution coding has been used in communication systems including deep space communication and wireless communication. At the receiver end the original message sequence is obtained from the received data using Viterbi decoder. It implements Viterbi Algorithm which is a maximum likelihood algorithm, based on the minimum cumulative hamming distance it decides the optimal trellis path that is most likely followed at the encoder. In this paper I present the convolution encoder and Viterbi decoder for constraint length 7 and bit rate 1/2.

Keywords **-** Convolution Encoder, trellis diagram, Verilog HDL, Viterbi algorithm, Viterbi decoder.

I. INTRODUCTION

Elias introduced convolutional codes in 1955[1]. Convolution coding has been used in communication systems including deep space communication and wireless communication. An advantage of convolutional coding is that it can be applied to a continuous data stream as well as block of data. Convolutional coding scheme correlates information elements by means of exclusive -or (XOR) operation, resulting in the increases of transmission redundancy[2]. Convolutional codes are used in applications that require good performance with low implementation cost. Several practical procedures have been developed for decoding.

In 1967 A.J Viterbi introduced "Viterbi decoding" based on the maximum likelihood algorithm. At the receiver, the actual received encoded data plus the noise is compared with the encoded data sequence for each of the possible outputs of the convolution encoder. The closest hypothetical encoded data sequence will be optimum received encoded data sequence[1]. Viterbi algorithm is the most resource consuming, efficient and robust.

II. CONVOLUTION ENCODER

Encoding of convolutional codes can be accomplished using simple registers. In convolutional encoder, the message stream continuously runs through the encoder unlike in the block coding schemes where the message is first divided into long blocks and then encoded. Thus the convolutional encoder requires very little buffering and storage hardware[3]. In this paper for a convolutional encoder, the following notations are used.

c = number of output bits.
x = number of input bits entering at a time.
m = number of stages of shift register.
L = number of bits in a message sequence.
j = number of modulo 2 adders.
Constraint Length: K = (m + 1) digits.
Bit Rate: r = x / c

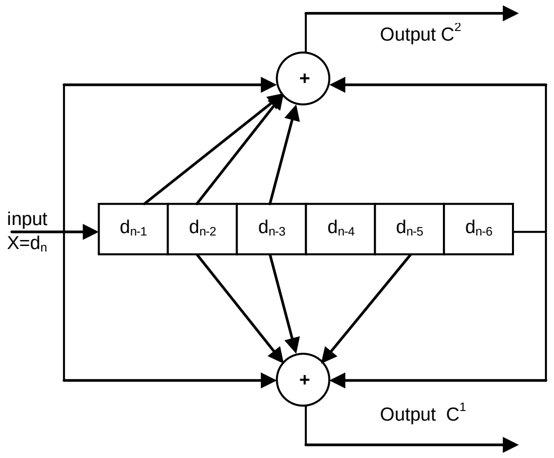


Figure 1. Convolution encoder for constraint length (K)=7,bit rate (r)=1/2

The block diagram of convolution encoder is shown in figure 2. To generate the output, the encoder uses 7 values of the input signal (1 present input bit and 6 previous input bits). The set of values of input data in the shift register is called a state. The number of input data values used to generate the code is called the constraint length. Each set of outputs is generated by XOR ing a pattern of current and shifted values of input data.

If $g_1^{(j)}$, $g_2^{(j)}$, $g_3^{(j)}$, ... $g_7^{(j)}$ denote the "**impulse responses**" also called "**generator sequences**" of input-output path through 'jth' modulo-2 Adder. Then the encoder generates 'j' number of output sequences denoted by $C^{(1)}$, $C^{(2)}$. From definition of discrete convolution[3], we have

$$C_n^{(j)} = \sum_{i=0}^{m} d_{n-i} * g_{i+1}^{(j)} \qquad \text{- - -(1)}$$

Therefore, $g^{(1)}$ and $g^{(2)}$ for the convolution encoder shown in figure1 is given by [2][3]
$g^{(1)} = g_1^{(1)} g_2^{(1)} g_3^{(1)} g_4^{(1)} g_5^{(1)} g_6^{(1)} =$ **1 0 1 1 0 1 1** $g^{(2)} = g_1^{(2)} g_2^{(2)} g_3^{(2)} g_4^{(2)} g_5^{(2)} g_6^{(2)} =$ **1 1 1 1 0 0 1**

The 2 output bits are generated by XOR ing the following bits.

$$C^1 = d_n \oplus d_{n-2} \oplus d_{n-3} \oplus d_{n-5} \oplus d_{n-6} \qquad \text{- - -(2)}$$
$$C^2 = d_n \oplus d_{n-1} \oplus d_{n-2} \oplus d_{n-3} \oplus d_{n-6} \qquad \text{- - -(3)}$$

A. State Diagram

The state of an encoder is defined as its shift register contents. Each new 'x' bit input results in a new state. Therefore for one bit entering the encoder there are $2^1 = 2$ possible branches for every state. If the Constraint length k=7, then the size of shift register would be m=6 which results in 2^m states. Therefore $2^6 = 64$ states are named from S0 to S63.

In the figure 2, 'State Diagram' is shown, here all the 64 states are represented and labelled as S0,S1,.......S63. Consider the state S0 (000000). With '0' input, the shift register remains at same state S0 and is shown as a dotted loop starting from S0 and ending at S0. With '1' input, the shift register moves to state S1 (100000) and is shown as a solid line starting from S0 and ending at S1. To make easy for tracking the transition two different types of line are used. Solid line represents the transition when the input bit is '1' and dotted line represents the transition when the input bit is '0'.

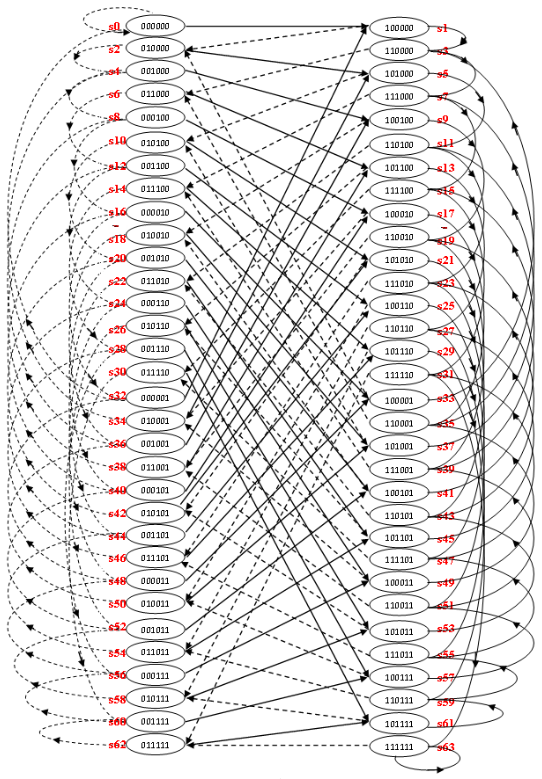


Figure 2. State diagram of K=7, r=1/2 convolution encoder

B. Code Tree

In the state diagram shown in Figure 3. It is very difficult to follow the paths because too many paths leads to confusion hence for better understanding state diagram can be re-drawn as 'code-tree'. The following rules are followed while constructing the code tree.

If the input is a '0', then the upper path is followed and if the input is a '1', then lower path is followed, the circles represents the 'node' and lines represents the 'branch'. The output code [$C^{(1)}$ $C^{(2)}$] for each input is shown on the branches.

Consider the input sequence 1 0 1 1 0 1 1 0 as the input to the convolution encoder, then the code tree for the above input is as shown below.

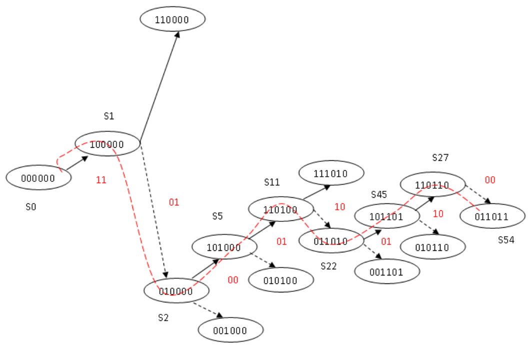


Figure 3. Code tree for the input 1011011

III. VITERBI DECODER

When a sequence of data is received from the channel, it is required to estimate the original sequence that has been sent. The process of identifying original message sequence from the received data can be done using the diagram called "trellis"[4]. A Viterbi decoder uses the Viterbi algorithm for decoding a bit stream that has been encoded using Forward error correction based on a convolutional code[5].Figure 4 shows the block diagram of Viterbi decoder.

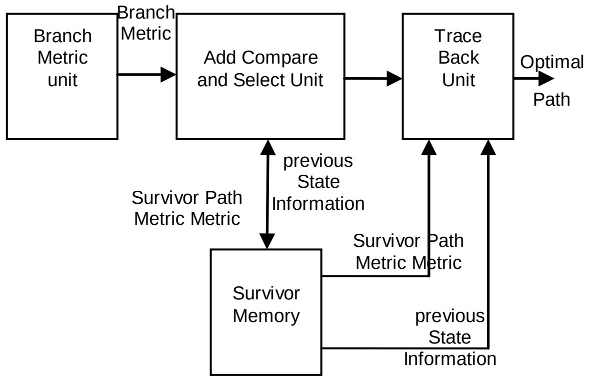

Branch Metric unit
Branch Metric
Add Compare and Select Unit
Trace Back Unit
Optimal Path
previous State Information
Survivor Path Metric Metric
Survivor Path Metric Metric
Survivor Memory
previous State Information

Figure 4. Block diagram of Viterbi decoder

It consists of following functional units.

a) Branch Metric Unit (BMU)
b) Add Compare and Select Unit (ACS)
c) Survivor Memory Unit
d) Trace Back Unit (TBU)

Using the functional units of Viterbi decoder the Viterbi Algorithm is computed and the original message sequence is obtained by following the below mentioned steps[2].

Step 1: Two parallel binary bits are inputted into the Viterbi decoder and then the hamming distance computation module (BMU) calculates sixty four set of hamming distance. Each set consists of two values because each current state can be reached by two possible paths.

Step 2: Cumulative hamming distance till the last branch is added to hamming distance of the new branches by ACS module. After adding operation each current state gets two new cumulative hamming distances, now ACS module compares the size of the two cumulative distances and selects the smaller one as a survivor. The smaller cumulative hamming distance becomes the benchmark for the next computation. Survivor paths of all the sixty four states are stored in RAM blocks of Survivor memory unit.

Step 3: sixty four survivor paths are stored in the RAM blocks at each stage. When there are no more encoded bits to process the detection of a node having minimum path metric is done by comparing all the sixty four cumulative hamming distances at the last stage.

Step 4: Using the minimum path metric of last stage the Trace back Unit starts back tracing survivor paths which are stored in the memory unit. according to survivor path values the original transmitted message is determined.

A. Branch Metric Unit

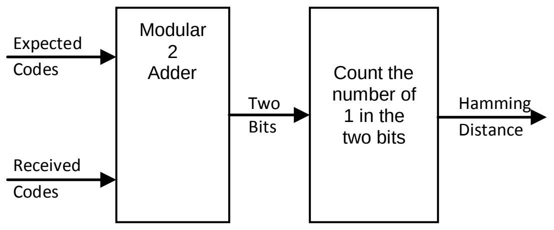


Figure 5. Block diagram of Branch Metric Unit

In this unit hamming distance computation is done. This Unit compares the received codes with the expected codes of the current state and calculates the hamming distance between them. The block diagram of BMU is shown in figure 5. Hamming distance is calculated by giving the received codes and expected codes to modular 2 Adder and number of one's present in the resulting two bits is checked. This gives the hamming distance between those codes.

B. Add Compare and Select Unit

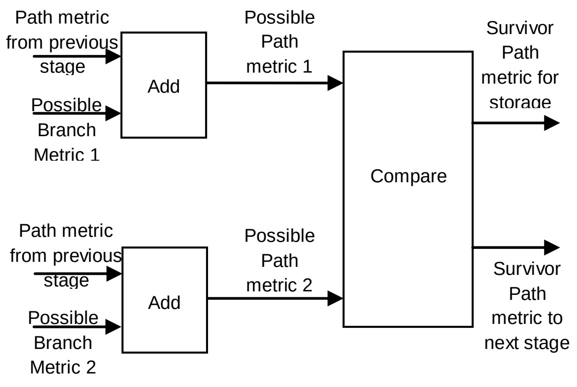

Path metric from previous stage
Possible Branch Metric 1
Add
Possible Path metric 1
Path metric from previous stage
Possible Branch Metric 2
Add
Possible Path metric 2
Compare
Survivor Path metric for storage
Survivor Path metric to next stage

Figure 6. Block diagram of Add Compare and Select Unit

The hardware architecture of the ACS module is shown in figure 6. Path metric of the node/state is found by adding the path metric from the previous stage and the present branch metrics. Since there are two possible way to reach any node/state two path metrics are obtained, these two are compared to select the one with the least path metric. The selected least path metric is sent for storage as well as it is used as benchmark for calculating the path metric of next stage.

C. Survivor Memory Unit

This unit is used for storing the survivor path values of the ACS modules. Each stage there are 64 survivor paths and number of such stages vary depending on the length of encoded bits received. another memory is reserved for trace back depth which defines the maximum number of stages that is allowed during the decoding process.

D. Trace Back Unit

Once the minimum path metrics of all the nodes at each stage is calculated, the minimum path metric at the last stage is found. The node having the minimum path metrics at the last stage is given as input to Trace Back Unit and then it starts trace backing the survival paths from that node and outputs the corresponding bit which has caused the transition of that path. In this paper decoding depth of 64 is set in the decoder. The below figure 7 shows the trace back procedure followed.

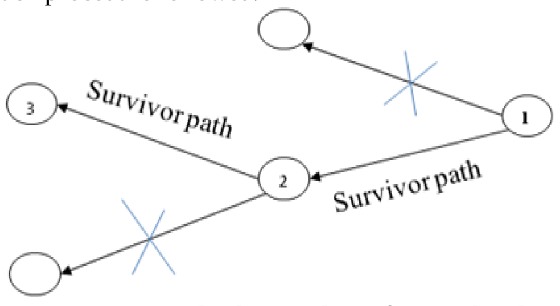


Figure 7. Trace back procedure of optimal path

Butterfly Diagram

As mentioned earlier there are always two possible path to reach any node/state. The representation of this is done using the figure 8 as shown below which is popularly referred as butterfly diagram.

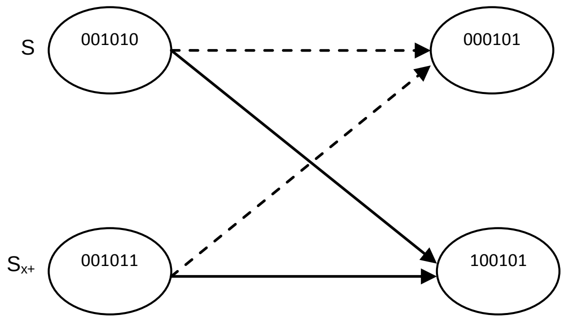


Figure 8. Butterfly diagram

In the diagram the state S_{40} ('000101') is taken as example, this state can be reached from two different states namely S_{20} and S_{52}. During computation out of these two any 'one' state will have the minimum path metric. The transition path having the minimum path metric is labelled as survivor path and stored in the memory. While tracing back if at any point this node comes then the survivor path to that node is followed to reach the previous stage and the bit causing that transition is taken as output. In the above figure the transition caused due to '0' is marked in dotted line and the transition caused due to '1' is marked as solid line.

Trellis Diagram

The original message sequence is encoded following a path which has to be determined at the decoder part of receiver to get the original message sequence from the encoded data. For the representation of this path a 'trellis diagram' is used. An example of trellis structure for K=3 and r=1/2 is given below in figure 9.

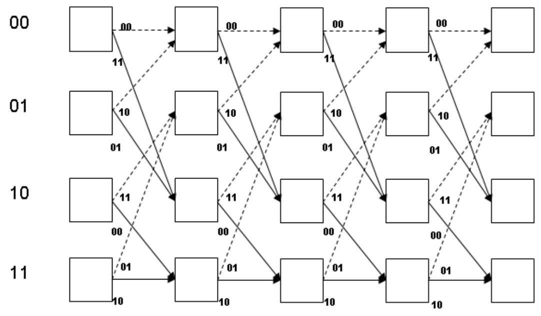


Figure 9. Trellis diagram for K = 3 and r = 1/2

IV. IMPLEMENTATION

Implementation of Convolution Encoder and Viterbi Decoder for constraint length 7 and bit rate 1/2 is done using Verilog HDL. The design is simulated and synthesized using ModelSim PE Student Edition10.2a and Xilinx ISE Design Suit 14.3 respectively. The design is tested for different trials by introducing the errors manually.

V. RESULTS

The Convolution Encoder and Viterbi Decoder for constraint length 7 and bit rate 1/2 has been developed and synthesis is done. Figure 10 shows the output of encoder.

Input = 1 0 1 1 0 1 1
output = 1101000110011000

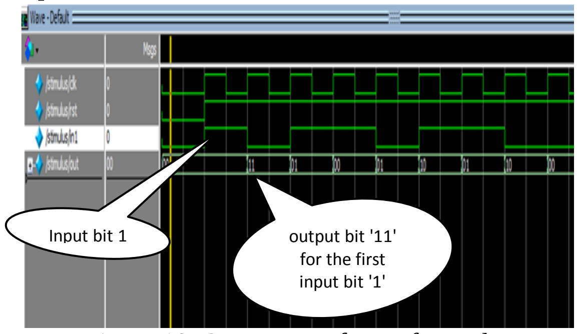


Figure 10. Output waveform of encoder

Considering the effect of noise suppose if the encoded data is corrupted then also decoder should be able to retrieve the original message sequence. Figure 11 shows the output of decoder if the error has occurred at second bit of the encoded sequence.

Input = 1<u>**0**</u>01000110011000
output = 1 0 1 1 0 1 1

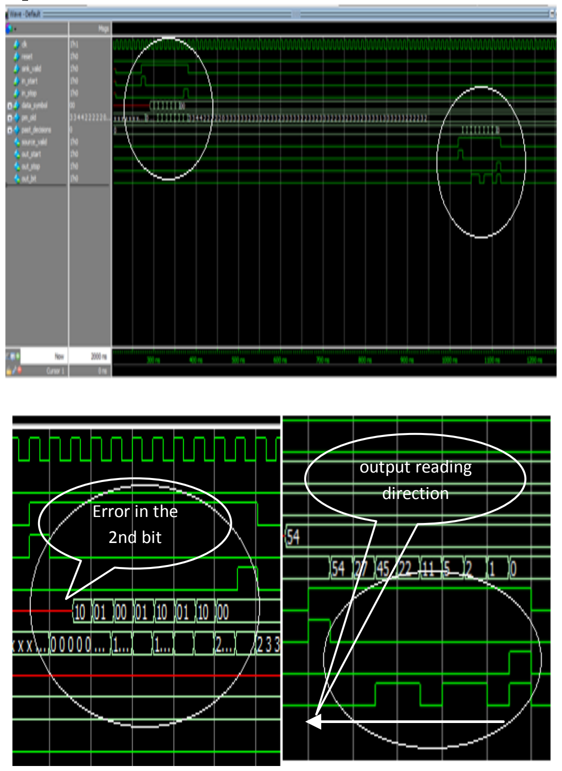
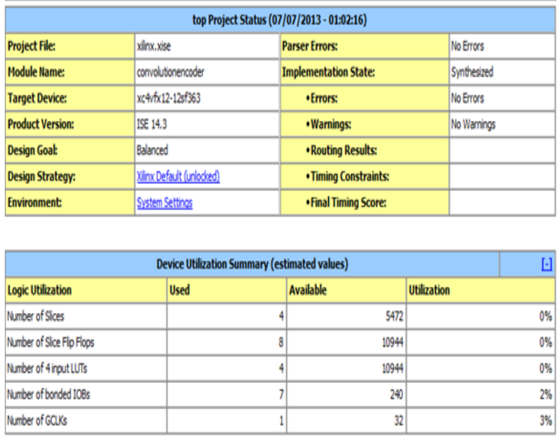


Figure 11. Output waveform of decoder

In the figure we can observe the trace backing process results in the original message sequence (output is read in the direction or arrow) rectifying the error.

The synthesis report gives the device utilization table for both convolution encoder and Viterbi decoder as shown in table 1 and table 2.

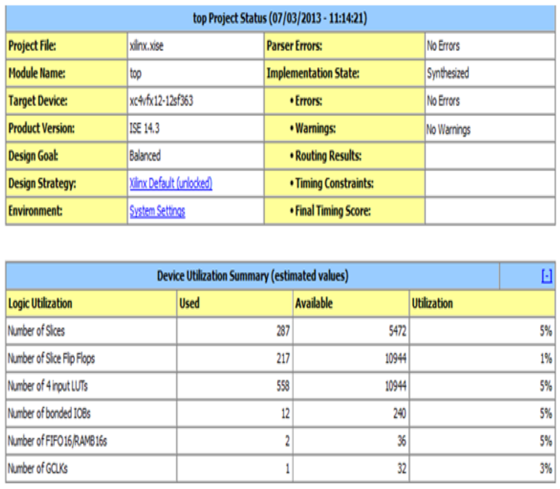
Table 1. Device utilization table for convolution encoder



top Project Status (07/07/2013 - 01:02:16)			
Project File:	xilinx.xise	Parser Errors:	No Errors
Module Name:	convolutionencoder	Implementation State:	Synthesized
Target Device:	xc4vfx12-12sf363	•Errors:	No Errors
Product Version:	ISE 14.3	•Warnings:	No Warnings
Design Goal:	Balanced	•Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	•Timing Constraints:	
Environment:	System Settings	•Final Timing Score:	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	4	5472	0%
Number of Slice Flip Flops	8	10944	0%
Number of 4 input LUTs	4	10944	0%
Number of bonded IOBs	7	240	2%
Number of GCLKs	1	32	3%

Table 2. Device utilization table for Viterbi decoder



top Project Status (07/03/2013 - 11:14:21)			
Project File:	xilinx.xise	Parser Errors:	No Errors
Module Name:	top	Implementation State:	Synthesized
Target Device:	xc4vfx12-12sf363	•Errors:	No Errors
Product Version:	ISE 14.3	•Warnings:	No Warnings
Design Goal:	Balanced	•Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	•Timing Constraints:	
Environment:	System Settings	•Final Timing Score:	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	287	5472	5%
Number of Slice Flip Flops	217	10944	1%
Number of 4 input LUTs	558	10944	5%
Number of bonded IOBs	12	240	5%
Number of FIFO16/RAMB16s	2	36	5%
Number of GCLKs	1	32	3%

VI. CONCLUSION

Convolution encoder and Viterbi decoder for constraint length 7 and bit rate 1/2 is implemented using Verilog HDL and simulated using ModelSim PE Student Edition

.

10.2a and synthesis is done using Xilinx ISE Design Suit 14.3 tool. The working of the design is cross verified for many trials with introducing errors.

REFRENCES

[1]. Madhu Vamshi Malladi, "Reconfigurable Viterbi Decoder", The University of New Brunswick, Canada, 2005.

[2]. Yan Sun, Zhizhong Ding "FPGA Design and Implementation of a Convolutional Encoder and a Viterbi Decoder Based on 802.11a for OFDM", Wireless Engineering and Technology, 2012, 3, 125-131, doi:10.4236/wet.2012.33019 Published Online July 2012

[3]. "Information Theory and Coding", by Prof. K. Giridhar, pooja publications.

[4]. V.Kavinilavu, S. Salivahanan, V. S. Kanchana Bhaaskaran, Samiappa Sakthikumaran, B. Brindha and C. Vinoth," Implementation of Convolutional Encoder & Viterbi Decoder using Verilog HDL",IEEE, 2011

[5]. HEMA.S, SURESH BABU.V, RAMESH P "FPGA Implementation of Viterbi Decoder", Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Corfu Island, Greece, February 16-19, 2007.

[6]. DR. Anubhuti Khare, Manish Saxena, Jagdish Patel "FPGA Implementation of Viterbi Decoder.

[7]. "Error Detection and Correction" at www.mathworks.in

[8]. Sherif Welsen Shaker, Salwa Hussein Elramly, Khaled Ali Shehata "FPGA Implementation of a Reconfigurable Viterbi Decoder for Wimax Receiver", International Conference on Microelectronics, 2009.

Acknowledgement

I would like to express my sincere gratitude to Mr. Kasetty Rambabu for his guidance in the course of my research. I am also grateful to Dr. K.N Hari Bhat and Mr. Ashish Gupta for their generous discussion and helps, without their help this paper is impossible to be completed.

Sandesh Y.M