



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Design and Implementation of Convolution Encoder with Viterbi Decoder

Rohan M. Pednekar, Dayanand B M

Crucible of Research and Innovation

PES Institute of Technology, Bangalore, Karnataka, India

Abstract: In communication system, error detection and correction has importance to issue reliability and efficiency of data transmission. The convolutional encoder with Viterbi decoder is more powerful structure for forward error correction technique. The maximum likelihood is the basic principle of Viterbi algorithm. The main aim of this paper is to understand the viterbi algorithm and design and implementation of a convolutional encoder and viterbi decoder with constraints length 3 and code rate 1/2. The proposed design of encoder and viterbi decoder has been realized using matlab.

Keywords: Convolutional Encoder, Viterbi Decoder, Viterbi algorithm, Matlab.

I. Introduction

Convolution coding is used in communication such as satellite and space communication to improve communication efficiency. The convolution code has enriched with the gift of a linear code with quality properties that it can operate on serial data as well as block codes. The convolution encoder with viterbi decoder is a forward error correction method is mostly suited to a channel in which the transmitted signal is corrupted because of additive white Gaussian noise. IS-95, a wireless cellular standard for CDMA (Code Division Multiple Access), employs convolution coding which is the part of convolution coding [1, 2].

The Viterbi decoding was developed by Andrew J. Viterbi who is a founder of Qualcomm Inc. His seminar paper on the technique is "Error Bound for Convolution Codes and Asymptotically Optimum Decoding Algorithm", published in IEEE Transaction on Information Theory, in April, 1967. Since then, other researchers have expanded on his work by finding good convolution codes, exploring the performance limits of the technique, and varying decoder design parameters to optimize the implementation of the technique in hardware and software [3]. The viterbi algorithm has now used in large number of applications. This algorithm works with trellis structure, which gives trace back for decoding the received information [3].

The fixed decoding time is the great advantage of the viterbi decoder because of this it is well suited to hardware decoder implementation. The convolutional coding with Viterbi decoding has been the predominant FEC technique used in space communication, particularly in geostationary satellite communication networks, such as VSAT (very small aperture terminal) networks [3].

Convolution encoder with viterbi decoder is a powerful method for forward error correction technique. The principle of viterbi algorithm is maximum likelihood decoding. Basic block diagram for convolution encoder followed by viterbi decoder with addition of Additive White Gaussian Noise (AWGN) is shown in the Figure1. Input data stream is fed to the convolution encoder, which produces encoded output stream according to designed encoder specification. The encoded data stream travels through channel having presence of noise, produces the new encoded stream with noise. Finally, this noisy data is given to the viterbi decoder that produces the corrected data which is applied to the encoder as input [2].

The motivation of this paper is to understand basic concept of both convolution encoder and viterbi decoder using matlab tool with both script language and simulink having constraint length 3.

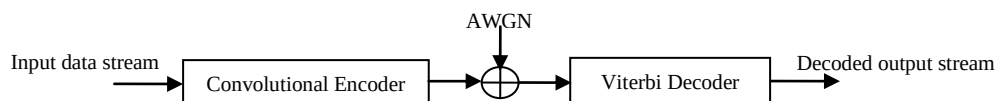


Figure1. Block diagram of Encoder and Decoder

II. Convolutional Encoder

Convolution code are commonly specified by three parameters (n , k , m), where n is number of output bits, k is number of input bits and m is number of memory registers. The efficiency of convolution code is measured in terms of code rate, which is calculated as k/n . Some other manufactures of convolution code chips specify the code by (n , k , L), where n and k are same as previous parameters while quantity L is called the constraint length of code and this parameter defined as: Constraint Length, $L = m+1$, as defined in this paper. [3, 4]. The convolutional encoder is easy to draw from its parameters shown in Figure2. FF1 and FF2 shows the shift

registers as memory registers. The selection of which bits are to be added to produce the output bits is called generator polynomial for output bit. In this paper, C1 and C2 represent generator polynomials as output bits of two modulo-2 adders. The generator polynomials C1 and C2 are of (111, 101) i.e.

$C1 = \text{mod2}(\text{input bit} + FF1 + FF2)$. $C2 = \text{mod2}(\text{input bit} + FF2)$.

The polynomials give the code its unique protection quality [4]. Figure2 shows encoder of code rate $\frac{1}{2}$ means each input bit is coded into 2 output bits. The constraints length of code is three. The output stream of encoder is as C1 C2 C1 C2 C1 C2 and so on. If this stream is not maintained then decoder will not give error corrected output hence we have to take care of bit synchronization. In this paper, we are considering that decoder is receiving synchronized input bits.

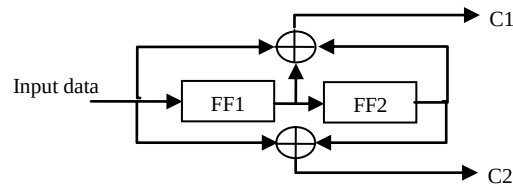


Figure2. Block diagram of convolutional encoder

Graphically, there are three ways to understand operation of the encoder. These are state diagram, tree diagram and trellis diagram [4].

A. State Diagram

A state diagram for encoder, Figure3, is of four states corresponding to the binary contents of the memory registers FF1 and FF2 of the encoder. The lines joining states indicates transition due to input of single information bit. Dashed line corresponds to input bit '1' while solid line represents input transition bit as '0'.

B. Tree Diagram

Figure4 shows another method of understanding operation of convolutional encoder. It is somewhat better than a state diagram but still not preferred approach for representing convolutional code [4]. The first branch indicates the arrival of a '0' or a '1' bit. Assume starting state is "00". If a '0' bit is received, transition will go upward direction while if a '1' bit is received, transition will go downward direction. The bits outside the parenthesis are output bits (C1C2) and the bits inside the parenthesis are represents next state i.e. present available bits in memory registers (FF1, FF2).

C. Trellis Diagram

Trellis diagrams are little complex than state and tree diagram still they are more preferable for higher constraint length [4]. The structure and working function of trellis diagram is in section III-A with Figure5.

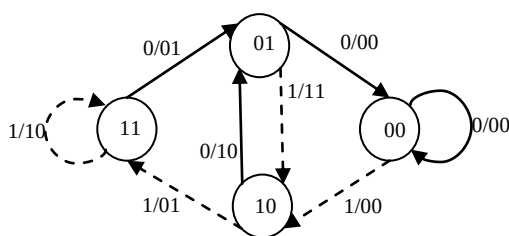


Figure3. State diagram of (2, 1, 3) code

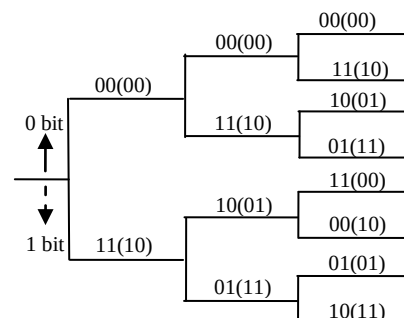


Figure4. Tree diagram of (2, 1, 3) code

III. Viterbi decoder

There are two basic categories to decoding convolutional codes. These are sequential decoding and maximum likelihood decoding based on Fano algorithm and Viterbi algorithm respectively [4].

The principle of viterbi algorithm is maximum likelihood decoding. Viterbi decoding was first shown to be an efficient and practical decoding technique for short constraint length codes by Heller. Forney and Omura demonstrated that the algorithm was in fact maximum likelihood [1]. It uses trellis diagram to compute branch metric and path metric from received signal to possible transmitted signal. Figure5 gives idea about trellis diagram. It has discrete time on horizontal x-axis while possible states on vertical y-axis. Number of possible states is depends on constraints length of encoder [4], here; constraint length (L) is 3 so possible states are

$2^{(L-1)}$ i.e four. Trace-back length (T) is five to six times of constraint length. As per algorithm decoding started at state 0. When a sequence of data is received from channel, it is desirable to estimate the original sequence that has been sent. And this process can be done with the help of trellis.

The bubble shows the states at each time interval. The solid line connecting dots in the diagram represents the state transitions when input bit is zero and the dotted line represents state transitions when input bit is one. The bits which are corresponding with solid and dotted lines represent encoder polynomial generator outputs. For example, at state 00, if input bit is '0' then state transition from $t=1$ to 2 with output '00' while if input bit is '1' then transition with output '11'.

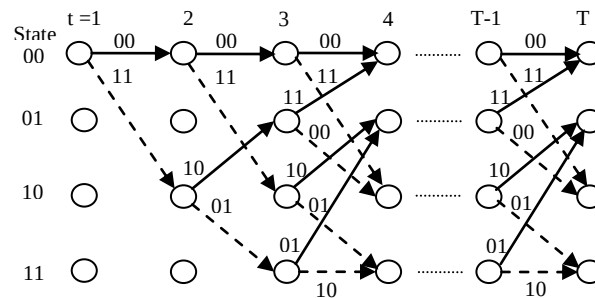


Figure5. Trellis diagram for hard decision viterbi decoder

A. Structure of Viterbi Decoder

The general structure of viterbi decoder is shown in figure2 which has mainly three parts: BMU (Branch Metric Unit), PMU (Path Metric Unit) and SMU (Survivor Memory Unit) [6]. Decoding started with hamming distance calculation shown in Figure2 as block HD.

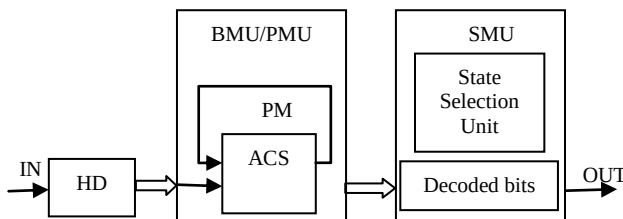


Figure6. General structure of Viterbi Decoder

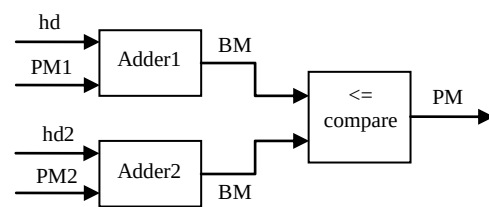


Figure7. Branch and Path metric calculation at a node

There are two types of viterbi decoder, hard decision and soft decision. A hard decision decoder uses 1- bit quantization, and uses hamming distance metric to find the distance between expected signal and actual received signal. Other hand soft decision decoder uses multi bit quantization and uses Euclidean distance as metric. The soft decision decoding is expensive and requires more memory than hard decision. This paper focuses on hard decision decoder.

B. Hamming Distance

There are four possible hamming distances. These are depends on how many polynomial generators used during encoder design. As shown in Figure2, encoder has two generators polynomial so there are four possible cases to calculate hamming distances; these are 00, 01, 10, and 11. Hence, zero, one and two, these are possible hamming distances.

C. Branch Metric Unit

The branch metric is done with hamming distance and previous state path metric [$BM = \text{Pre. PM} + \text{hd}$]. Initially, at $t=1$ the branch metric is nothing but hamming distance because it is considered as previous path metric is zero. At each node of state there are two branch metrics. So, addrer1 and addrer2 gives two branch metrics BM1 and BM2 respectively as shown in Figure7.

D. Path Metric Unit

The minimum of branch metrics is a path metric for each node [$PM = \min(BM1, BM2)$]. In viterbi decoder, the path metric is calculated for all state nodes as per constraints length which is further helpful during trace-back for that it is stored in path memory. If this path history memory is fixed amount, then the decoder can output the oldest bit on an arbitrary path each time it steps one level deeper into the trellis [5].

E. Survivor Memory Unit

The survivor memory stores the state having minimum path metric. And finally these states are used to get decoded bits, i.e. original data. The path comparisons made for paths entering each state require the calculation of the likelihood of each path involved for the particular received information [5].

IV. Proposed design simulation in Matlab

Viterbi decoder presented here is totally based on flow graph shown in Figure8. Operation can be discussed as shown in the flow graph; taking constraints length (L) is equal to 3, in case of Figure2 as an example.

1. Decoder is start with initialization of resetting all registers to 0's. Then as discussed in section III-C, hamming distance has been calculated. The branch metric for the two paths entering at state 00 are calculated by adding the previous path metric and hamming distance of the previous states 00 and 01. Here, the first part of Add-Compare-Select i.e. Add has done. Same addition is done using adders at each state node.
2. Compare-Select is the next part in which comparison between branch metrics computed. The minimum of branch metric is stored as new path metric of that state node. This Add-Compare-Select operation performed for the all state nodes. Here, total states are $2^{(L-1)} = 2^{(3-1)} = 4$ and the trace-back length (t) is 15 [considered as 5 times of constraints lengths, $L*5 = 3*5=15$]. So total states of the metric are $4*15$ i.e. four rows of states with 15 columns of discrete time from $t=1$ to $t=15$.
3. After having path metric for all state node, the state of minimum path metric for all $t=15$ to $t=1$ are calculated. Once, all state are available for trace-back length the final step of trace-back has been performed to get corrected decoded output. Finding state is depends on the previous state. Illustration, at $t=15$, the state selection is the minimum of path metric between all state nodes. Next at $t=14$, the state selection is depends on the previously selected state with the reference of Figure 5, If t is the 15th state then we can see that if the minimum state is 1 or 3, then state selection at $(t-1)$ will be between state 1 and 2 only. Also if the minimum state is 2 or 4, then state selection at $(t-1)$ will be between 3 and 4. This procedure will followed till state $t=2$, because as we have continuous data stream, then we cannot say that trace-back will stop at state '00' only. So here is a solution that to find state at $t=1$, use $t=15$ as a previous state and then using state at $t=2$ find out the state at $t=1$. Now here, is all state available to find decoded bits Figure16.

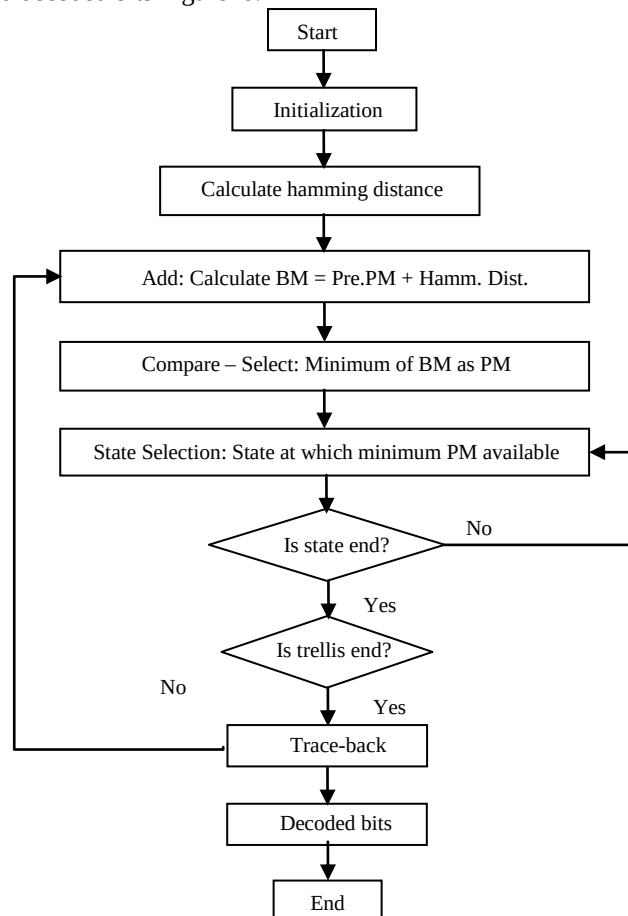


Figure8. Design flow graph of viterbi decoder

V. Result analysis In Matlab

The results for proposed viterbi decoder with corrected outputs with intermediate steps are discussed in this section which is very useful to understand the design. The PN sequence generator of polynomial [19 18 17 14 0] generate random binary signal. The signals generated at each block shown in figure 10. Noise introduced in the channel is of 7dB. Path metric calculation for each state and each time instance has done with module shown in figure 11 with add-compare-select unit. Unit delay is to get previous path metric for further calculation of path metric.

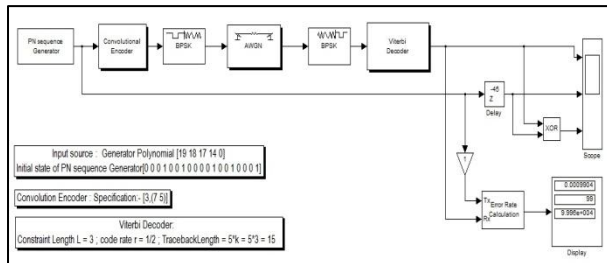


Figure9. Proposed design of system for convolution encoder with viterbi decoder.

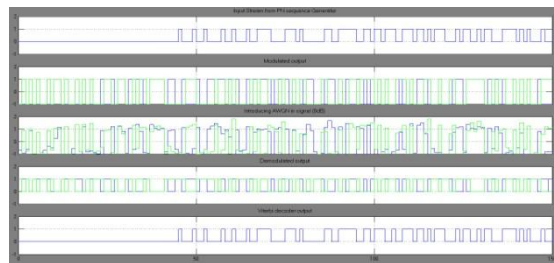


Figure10. Delayed PN sequence generator output, modulated and demodulated signal with introducing noise, viterbi decoder output.

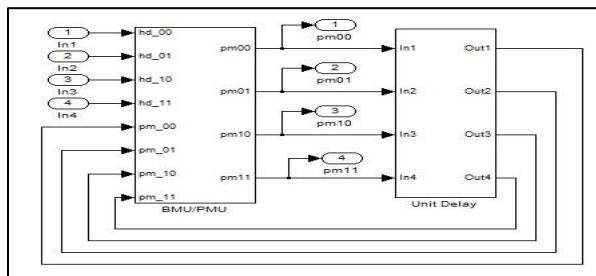


Figure11. Branch metric and path metric unit including Add-compare-select unit.

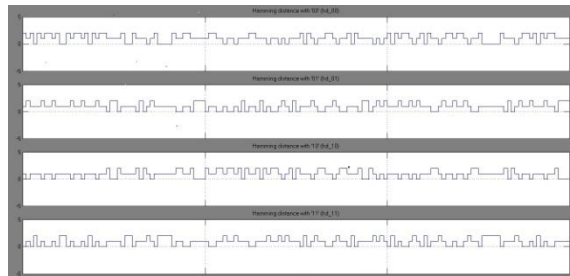


Figure12. The hamming distance for four cases "00", "01", "10" and "11" are shown respectively.

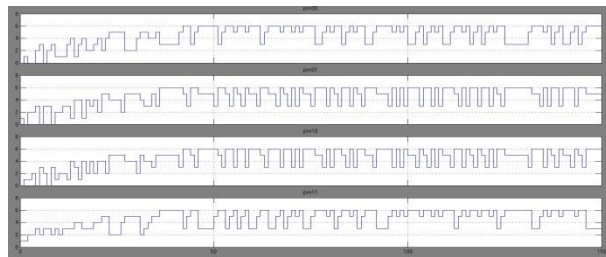


Figure13. Path metric outputs for all four state as the outputs of Add-Compare-Select Unit.

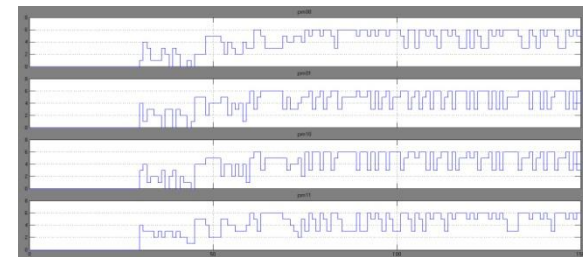


Figure14. Path metric outputs from $t=15$ to $t=1$.

As shown in the Figure1, path metric is from $t=1$ to 15. Here, for trace-back it requires from $t=15$ to $t=1$. This requirement is fulfillment and gives the following delayed output. As discussed in section IV-3, the state selection outputs are depends on the minimum path metric at each time instance. At $t=15$ the minimum path metric is directly comparison between available path metrics for all states. The path metric at $t=14$ to $t=1$ is done with previous state selection where minimum path metric stored. The minim_PM subsystem is designed to calculate minimum path metric at each time instance, Figure15. These calculated minimum path metrics are required to decide at which state these minimum path metrics are stored. Selection of states happens at state_select1 subsystem. State_select2 is designed in such a way that state at $t=1$ is to be decided and replace with state_select1 subsystem output to get required final state selection output.

There is comparison between state at $t=15$ and state at $t=14$ to get 15th decoded bit, comparing state at $t=14$ and $t=13$ to get 14th decoded bit and so on go for other decoded bits this is nothing but comparison of current state and previous state. 1st bit is getting by the comparison between state at $t=1$ and state at $t=15$. Figure16 gives working idea of the subsystem state_select2.

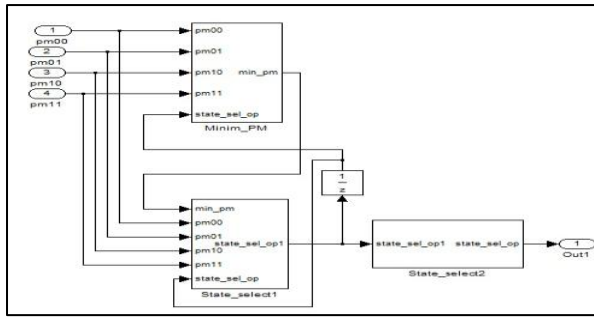


Figure15. State selection module.

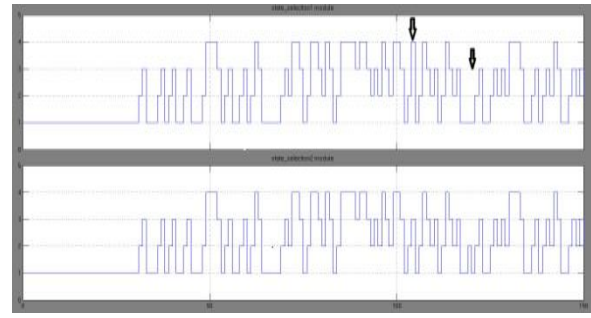


Figure16. State_select1 and State_select2 output.

Using AWGN block from simulink library bit error rate has been calculated by varying signal to noise ratio. Result for BER versus E_b/N_0 for code rate $\frac{1}{2}$ and constraint length of 3 and simulated uncoded BER is plotted in Figure17.

VI. Conclusion

The basic design and implementation of convolution encoder and viterbi decoder with the help of viterbi algorithm is explained in this paper. This paper explains step by step working of three basic modules (BMU, PMU and SMU) of viterbi decoder. It can be seen that without coding, the value of E_b/N_0 needed to achieve a BER of 10^{-3} is around 12.2dB. This error rate can be achieved with coding at E_b/N_0 around 9.9dB using hard-decision viterbi decoder. So we can express it by saying that the coding gain at a BER of 10^{-3} is 2.2dB. This paper helps beginners to understand working of viterbi algorithm those who wants to work on viterbi decoder.

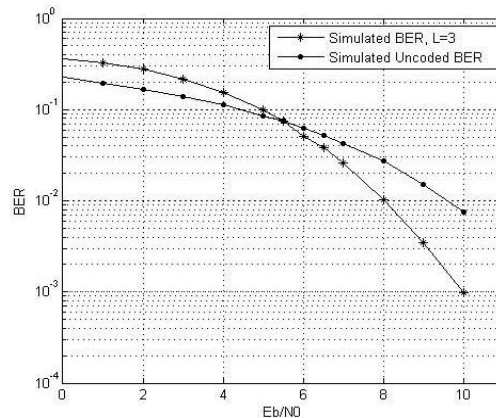


Figure17. Bit error rate versus E_b/N_0 for rate $\frac{1}{2}$ viterbi hard decoding; constraint length (L) = 3.

References

- [1] Andrew J. Viterbi and Jim K. Omura, "Principle of digital communication and coding", McGraw-Hill, international edition, 1979.
- [2] V.Kavinilavul, S. Salivahanan, V. S. Kanchana Bhaaskaran, Samiappa Sakthikumaran, B. Brindha and C. Vinoth, "Implementation of convolutional encoder and viterbi decoder using verilog HDL", *IEEE trans. ICEPT*, vol. 1, pp. 297-300, April 2011.
- [3] Chip Fleming, "A tutorial on convolutional coding with viterbi decoding", 2011. Available: <http://home.netcom.com/~chip.f/viterbi/tutorial.html>
- [4] Charan Langton, editor, "coding and decoding with convolutional codes", Available: www.complextoreal.com
- [5] Jerrold A. Heller and Irwin Mark Jacobs, "Viterbi decoding for satellite and space communication", *IEEE Trans. Commun.*, vol. COM-19, pp. 835-848, Oct., 1971.
- [6] Swati Gupta, rajesh Mehra, "FPGA Implementation Of Viterbi Decoder using Trace Back Architecture", *International Journal Of Engineering Trends and Technology*, pp. 131-134, June 2011.
- [7] Jinjin He, Huaping Ling, Zhongfeng Wang, Xinming Huang and Kai Zhang, "High-Speed Low-Power Viterbi Decoder Design for TCM Decoders", *IEEE Trans. VLSI*, Vol. 20, pp.755-759, April 2012.
- [8] Yin Sweet Wong, Wen Jian Ong, Jin Hui Chong, Chee Kyun Ng and Nor Kamariah Noordin, "Implementation of convolutional encoder and viterbi decoding using VHDL", *Proceedings of SCORed*, pp. 22-25, Nov. 2009.
- [9] Shuji Kubota, Shuzo Kato and Tsunehachi Ishitani, "Novel viterbi decoder VLSI implementation and its performance", *IEEE Trans. Commun.*, vol. 41, pp. 1170-1178, Aug., 1993.

Acknowledgments

The authors would like to express their sincere thanks to Dr. V. K. Agrawal, Director, Crucible of Research and Innovation, Dr. T. S. Chandar, HOD, Dept. of ECE, Dr. V. S. Rao, Professor, Dept. of ECE and Dr. K. N. B. Murthy, Principle, PES Institute of Technology, for the encouragement given throughout this development work.