RESEARCH ARTICLE OPEN ACCESS

Analysis and Implementation of Hard-Decision Viterbi Decoding In Wireless Communication over AWGN Channel

Muhammad Asif[1] , H.M.Rehan Afzal[1], Anum Nusrat[1,2], Arslan Akbar[3]
1. School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
2. Information Management and Information System
3. University of Science and Technology Beijing

Abstract
Convolutional codes are also known as Turbo codes because of their error correction capability. These codes are also awarded as Super product codes, because these codes have replaced the backward error correction codes. Turbo codes are much more efficient than previous backward error correction codes because these are Forward error correction (FEC) codes and there is no need for a feedback link to request the transmitter for retransmission of data, when bits are corrupted in the information channel. A Viterbi decoder decodes stream of digital data bits that has been encoded by Convolutional encoder. In this paper we introduce a RSC (Recursive Systematic Convolutional) encoder with constraint length of 2 code rate of 1/3. The RSC encoder and Viterbi decoder both are implemented on paper, as well as in MATLAB. Simulation results are also presented by using MATLAB.
Keywords- Convolutional codes; RSC encoder; Viterbi decoder; Trellis diagram; Hamming distance; Viterbi algorithm

I. INTRODUCTION

The turbo codes can utilize an iterative decoding process to achieve the near Shannon limit performance [1]. These codes are also known as convolutional codes. Because of their efficient error correction ability over a noisy channel, these codes were awarded as Super product codes. Turbo codes [5, 9] are superior over previous backward error correction codes. Because, In Backward error correction techniques, feedback link is required to request the transmitter for retransmission of data when bits are corrupted over noisy channel. Many standards adopt turbo codes as their forward error correction techniques [2]. In Forward error correction, there is no need for feedback link. Instead of sending a retransmission request to transmitter, in forward error correction the decoder has the ability to correct the errors as well as to decode the original data. Convolutional coding is used in deep space communication as well as in wireless communication. These codes are replacing the previous backward error correction codes and block codes. Because, block codes can be implement only on blocks of data but not on continuous stream of data. But, Turbo codes can be implemented for Block codes s well as for continuous stream of data bits. A third generation wireless standard (CDMA), under preparation, plans to adopt turbo coding [3].

The Viterbi algorithm was introduced and analyzed by Viterbi in 1967 [4]. It is widely used as decoding technique for convolutional codes and also for bit detection. The algorithm works on the basis of trellis diagram which is collected from state diagram of encoder. It performs maximum likelihood decoding and mostly used for Forward error correction decoding. At receiving end, Hamming distance is calculated between transmitted bit sequence and expected received bit sequence. By comparing Hamming distance of each path, only that path is selected for which hamming distance is minimum. Actually, Hamming distance is the difference between transmitted sequence and received sequence. The general overview of RSC encoder and Viterbi decoder is shown in figure 1.

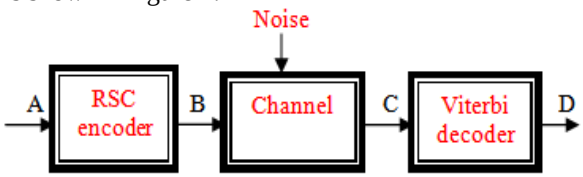

Figure1. RSC encoder and Viterbi decoder

Here, A is the input data which is given to the RSC encoder, B is the encoded data collected from encoder, The encoded data B is given to the information channel in the presence of noise and produces C which is noise added encoded data. This noise added data is given to the Viterbi decoder which produces the final decoded data D at receiver.

II. IMPLEMENTATION OF RSC ENCODER

Convolutional codes are very efficient for secure transmission of digital data on any noisy channel. Convolutional codes are generally described by three

factors: number of input data bits (k), number of output data bits (n) and number of memory registers (m). The ratio of k to n is called code rate (r), r = k/n. The length of encoder is called constraint length (K) given by following relation:

$$K = k (m-1)$$

Where 'k' is number of input data bits and 'm' is the number of memory registers. In this paper we will implement RSC (Recursive Systematic Convolutional) encoder and Viterbi decoder. First, we will encode input data by using RSC encoder and then we will decode the same data at receiving end by using Viterbi algorithm. The block diagram of RSC encoder is given below:

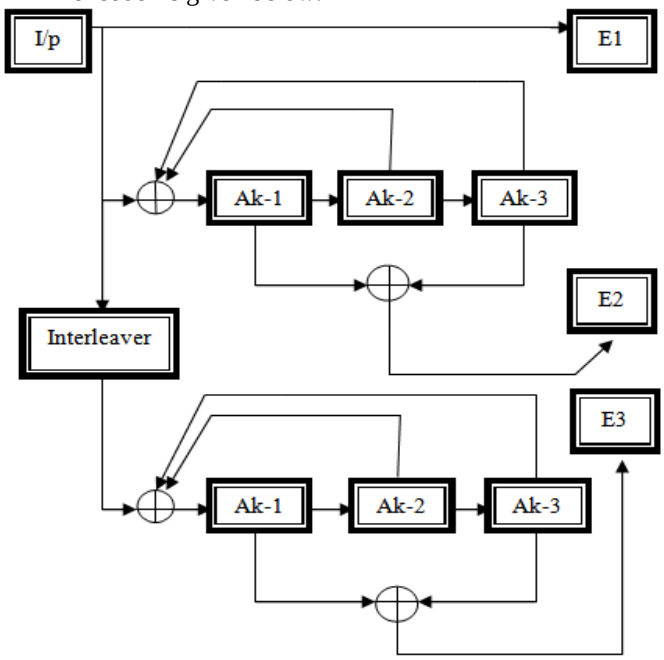

Figure 2. RSC encoder diagram

This is the block diagram of RSC encoder with constraint length (K) 2 and code rate 1/3. There are two parallel encoders for encoding the input data. In this work, we have input data stream, A=[1 0 1 1 0 1 0 1]. There are three outputs of RSC encoder: Systematic output E1which is same like input, output from encoder1 (E2), output from encoder2 (E3). First, we will multiplex all these three data streams, after multiplexing we will throw this information over information channel.

There are three memory registers (Ak-1, Ak-2, Ak-3). Initially, we assumed that the initial state of Ak-2 and Ak-3 is 0. First, we take the XOR of Ak-2 and Ak-3, output from Ak-2 and Ak-3 is operated with incoming input data bit and this output is stored in first register Ak-1. Finally, we will take the XOR of Ak-1 and Ak-3, this is the first encoded bit. In next step, we clip the Ak-3 and move Ak-1 and Ak-2 towards right. We have to continue the same operation until the all data bits are encoded. After encoding all input data bits we get the output from encoder1 which is, E2= [0 1 0 11 1 10 1 1]. In next step, first we interleaved the input data by using Pseudo random interleaver and get the interleaved data I= [1 1 1 0 1 1 1 0 0]. After interleaving, we encode the interleaved data by using same operation like encoder1 and we will get the encoded data from encoder2 which is E3= [0 1 0 10 1 0 1 0 0]. Finally, we have multiplexed the encoder outputs and get multiplexed data which is B= [E1 E2 E3]. This multiplexed data is passed over the noisy channel. At receiving end, we will receive this multiplexed data. First we will demultiplex the data, and then by using Viterbi decoder we will decode the input data that has been encoded by RSC encoder at transmitting end.

In digital communication transmission, we only deal with binary data bits (0 and 1). Initially, we will apply '0' to all states of encoder and observe the possible outputs then we apply '1' to all states of encoder and calculate the possible outputs. The number of possible states of any convolutional encoder can be calculated by 2^K, where K is the constraint length of RSC encoder which is 2. So, there are four initial possible states of RSC encoder in this case {00, 01, 10, 11}. We will draw the state diagram table which is given below:

TABLE I. STATE DIAGRAM TABLE

INPUT	CURRENT STATE	NEXT STATE	OUTPUT
0	0 0	0 0	0
0	0 1	1 0	0
0	1 0	1 1	1
0	1 1	0 1	1
1	0 0	1 0	1
1	0 1	0 0	1
1	1 0	0 1	0
1	1 1	1 1	0

This is the state diagram table; it represents the all possible states of RSC encoder with input bits 0 and 1. From state diagram table we can also observe the all possible output data bits that have been encoded by the RSC encoder. By using this table, we can draw the state diagram in this scenario.

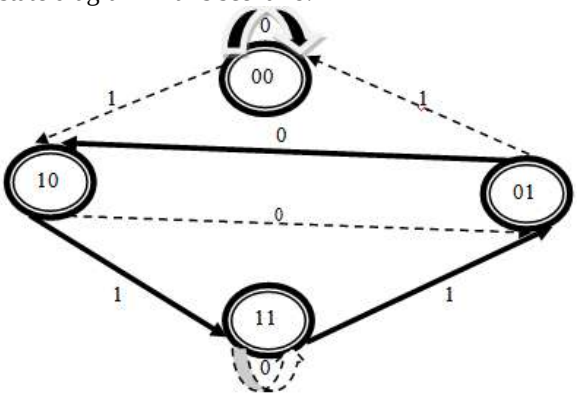

Figure 3. RSC Encoder State Diagram

This is called state diagram of RSC encoder. A black circles give the information about the state c RSC the encoder. This is stored in the shift register All solid and dotted lines between these state represents the outputs for incoming input data bits. A solid line represents the output when incoming inpu data bit is 0, and a dotted line represents the output fc incoming bit 1. Each incoming input bit cause transition from one state to another state of encode We know that there are four possible states c encoder, when we apply the input bit 0 to all states c encoder then it will leads to a transition sequence S {00, 10, 11, 01}and produce encoded output sequenc E= [0 0 1 1]. For incoming data bit 1, we get transitio sequence S= {10, 00, 01, 11} and produce outpu encoded bits E= [1 1 0 0]. This state diagram will help us for drawing trellis diagram when we will implement Viterbi decoder at receiving end.

III. VITERBI ALGORITHM

The Viterbi decoder uses Viterbi algorithm [6-8], for decoding the input data information that has been encoded by convolutional encoder at transmitting end. The Viterbi algorithm is also known maximum likelihood decoding algorithm. The Viterbi decoding is divided into two categories, Hard decision decoding and Soft decision decoding. In Hard decision decoding, the received sequence is converted into only two levels, either 0 or 1. But, in Soft decision decoding, the received sequence is converted into more than two levels. Soft decoding is complex and much more expensive than hard decoding. Here, we will deal only with hard decoding.

When a sequence of digital data is received at receiver, it is desired to retrieve the original data and correct the bits that have been corrupted by noise in the information channel. The Viterbi algorithm uses trellis diagram to calculate the Hamming distance between received data sequence and possible transmitted sequence. Actually, Hamming distance is the difference between received and transmitted sequence. According to Viterbi algorithm, we select that path in trellis diagram for which Hamming distance will be minimum.

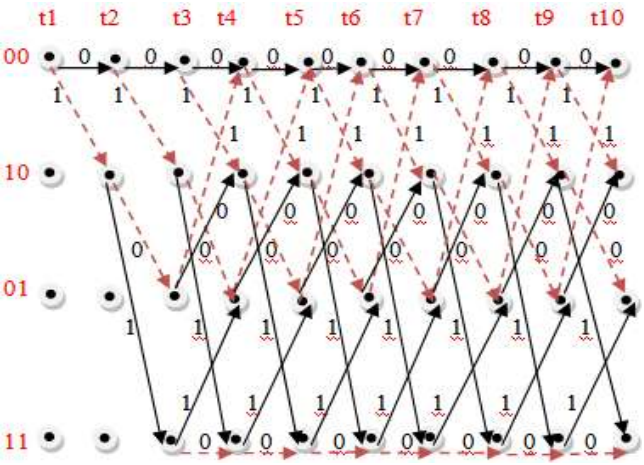

Figure 4. Viterbi Trellis diagram

This is called Viterbi trellis diagram. We draw this trellis diagram with the help of state diagram in figure 3. It can be seen from trellis diagram that there are four states, s= {00, 10, 01, 11}. All dotted lines represent the outputs for input bit 1 and solid lines represent the outputs for input bit 0. This trellis diagram shows the all possible encoded transmitted sequences. It is important to note that the number of trellis path in trellis diagram is directly proportional to number states of RSC encoder. When number of states of encoder increased, trellis diagram becomes more complex.

IV. IMPLEMENTATION OF VITERBI DECODER

The RSC encoder and Viterbi decoder are implemented by using MATLAB. The Viterbi decoder uses Viterbi algorithm for decoding original input information. The design of flow chart diagram of Viterbi decoder is given below in figure 5.

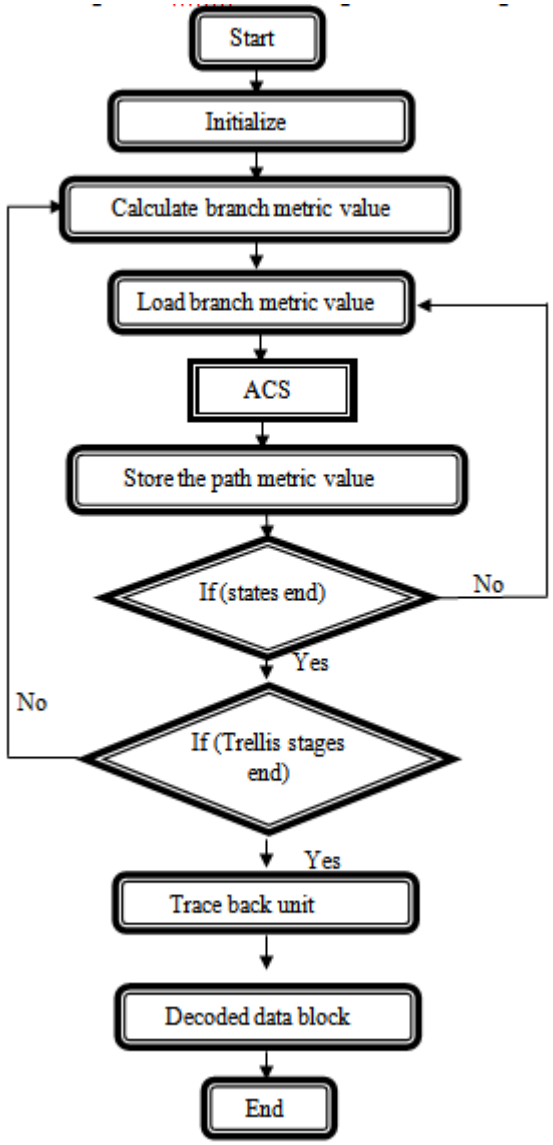

Figure 5. Flow chart diagram of Viterbi decoder

The Viterbi decoder [10-12], consist of three main functional units, Branch metric unit, Path metric unit and Survivor memory unit. In Branch metric unit, the difference between received sequence and expected transmitted sequence is calculated, which is called Hamming distance in case of hard decision is decoding. In Path metric unit, the Hamming distance of each path is calculated and compared with other paths. By comparing, that path is selected for which Hamming distance is minimum. The Path metric unit is also called ACS (Add Compare Select) unit. Then, corresponding bit decision is transferred to the Survivor memory unit. The Survivor memory collects the bit decision from Path metric unit and produce decoded bit sequence.

Now, first we will draw Viterbi trellis diagram with Hamming distance, and then we will perform step by step decoding process by using Viterbi algorithm.

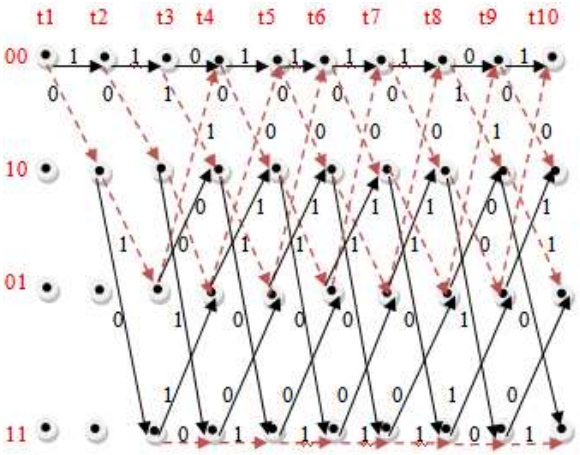

Figure 6. Viterbi trellis diagram with Hamming distance

The above figure 6 represents the Viterbi trellis diagram with Hamming distance. All dotted and solid lines are representing the Hamming distance between received and transmitted sequence. For each time instant't' we will calculate and compare the Hamming distance between all states.

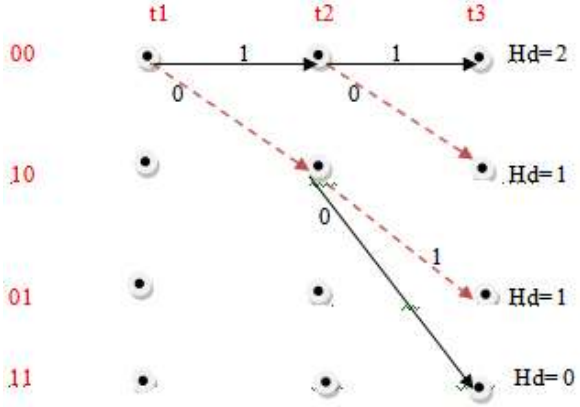

Figure 7. Viterbi decoding Step one

From figure 7, it can be seen that there are two paths between t1 and t2 to reach states at t3 from states at time t1. But, we can see that the total Hd (Hamming Distance) of solid line path is 3 and Hd for dotted line is 1. So, according to Viterbi algorithm dotted line path is selected because its Hd is less than the solid line path.

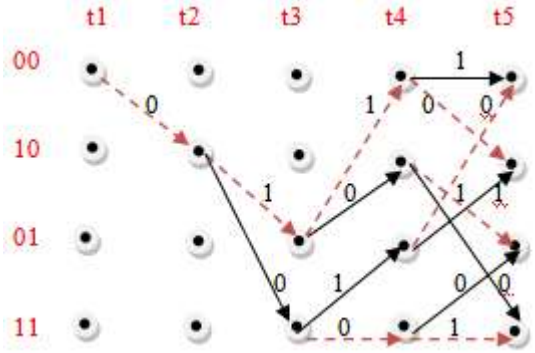

Figure 8. Viterbi decoding Step two

From above figure it can be seen that there is only single path between time stages t1 and t2, so this is the first decoded bit. we know that dotted line represents the input bit 1, so 1 is our first decoded bit. Now, from figure we can see that there are again two paths between t2 and t3 to reach at t4 states. But, the Hd of dotted line at t4 is 3 and Hd of solid line path is 1.So, we will select solid line path and ignore dotted line path to reach at t4.

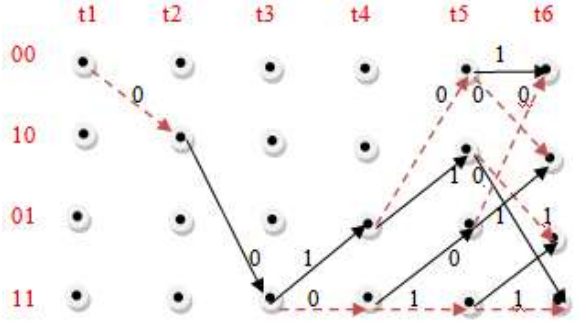

Figure 9. Viterbi Decoding Step three

From above figure we can see that again there is a single path between t2 and t3, so, this is our second decoded bit. As we know that the solid line represents the input bit 0.So, our second decoded bit is 0. There are two paths between t3 and t4 to be reach at t5. But, we select only dotted line path because its Hd is less than solid line path. When there is only single path between two states of trellis, then this path will represent the decoded bit at receiving end. If there is single dotted line between two states then it means that the decoded bit is 1. If there is solid line, then it represents that the decoded bit is 0 because solid line represents the input bit 0. The Viterbi algorithm is also Known as Maximum likelihood decoding algorithm. We will continue the same procedure as described above for selecting survivor path at each time instant‘t’ until our original data is retrieved. At the end, most likely path between first and last stage will represent the decoded sequence.

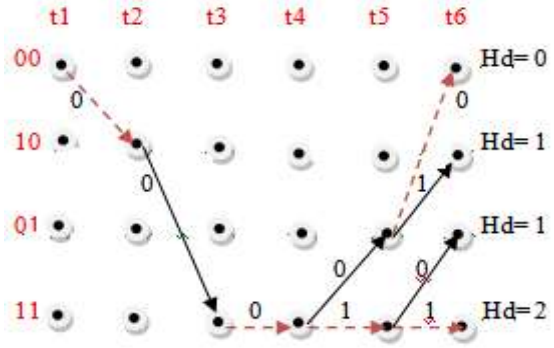

Figure 10. Viterbi Decoding Step four

The above figure represents that there are two paths between t4 and t5 to be reached the states at t6. By adding Hd of two paths, we see that the total Hamming distance of solid line path is 1 and Hd for dotted line path is 3. Therefore, we will select solid line path.

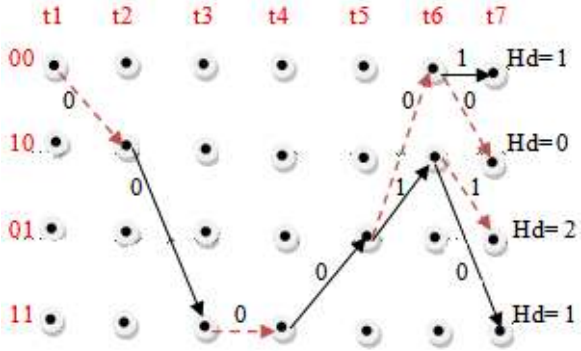

Figure 11. Viterbi Decoding Step five

From figure 11, we can see that again there are two paths between t5 and t6 to reach the states at t7. The total Hd of dotted line is 1 and Hd for solid line is 3 to be reached at t7. So, we will choose dotted line and this will be our 5th decoded bit in our desired sequence.

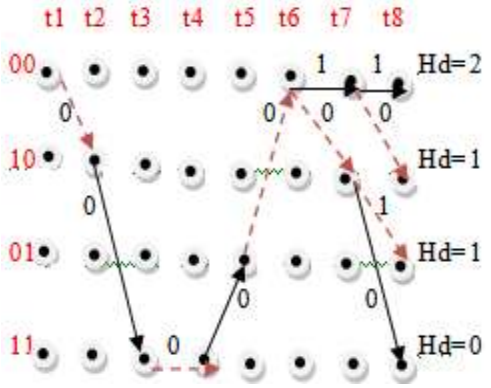

Figure12.Viterbi Decoding Step seven

Above figure shows that again there are two paths between t6 and t7 to be reached the states at t8. By comparing Hd of two paths, we conclude that Hd of solid line is 3 and Hd for dotted line is 1. So, we will select dotted line path and this will be our 6th decoded bit.

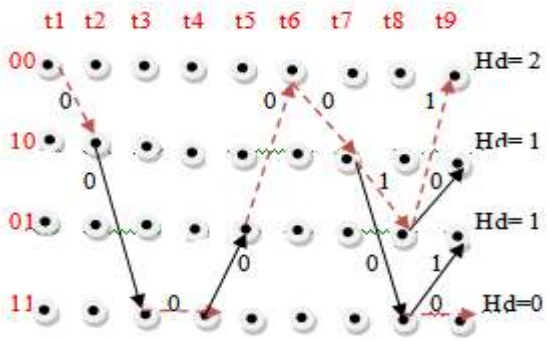

Figure 13. Viterbi Decoding Step seven

From figure 13 we can see that there are two paths between t7 and t8 to be reached the states at t9. By calculating Hd of both paths, we came to know that the Hd of dotted line is 3 and for solid line is 1. Therefore, we will select the solid line path and this will be our 7th decoded bit.

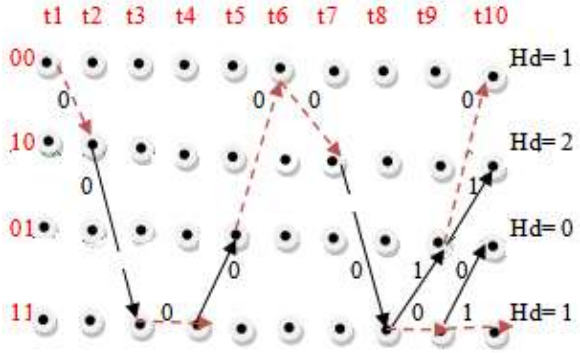

Figure 14. Viterbi Decoding Step eight

The above figure also shows that again there are two paths between t8 and t9 to be reached the states at t10.But, the Hd of solid line path is 3 and for dotted line path is 1. We will select the dotted line and this is our final 8th decoded bit because length of our input signal was eight bits.

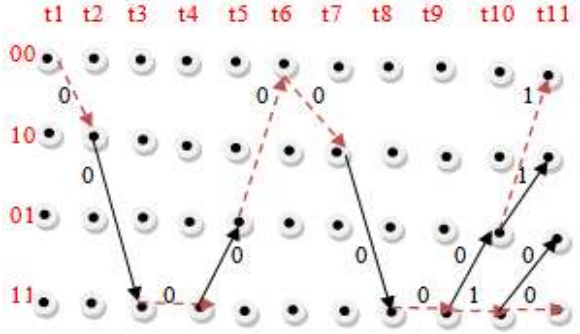

Figure 15.Viterbi Decoding Step nine

The above diagram is our final trellis because our eight bits have been decoded and maximum likelihood path between t1 and t9 is representing our decoded bits sequence which is our desired information that was encoded by RSC encoder at transmitting end. In maximum likelihood path, dotted line is representing for input bit 1 and solid line for bit 0. It is important to note that we have encoded the input bits from right to left order, so our first decoded bit will also be placed at right most position in decoded sequence. So, our decoded sequence is D= [1 0 1 1 0 1 0 1]. When we compare this sequence with input data, A= [1 0 1 1 0 1 0 1], we came to know that it is exactly matched with input data which was our required information at receiving end.

V. SIMULATION RESULTS AND DISCUSSION

In this work, the RSC encoder and Viterbi decoder both are implemented on paper, as well as by using MATLAB. The simulation results are taken by using MATLAB GUI (Graphical User Interface). The simulation results of RSC encoder and Viterbi decoder are given.

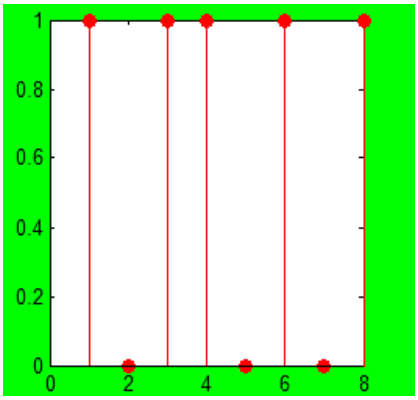

Figure 16. Input Data

The above figure represents the input data information which we to transmit on information channel after encoding. We will pass this input information through RSC encoder 1.

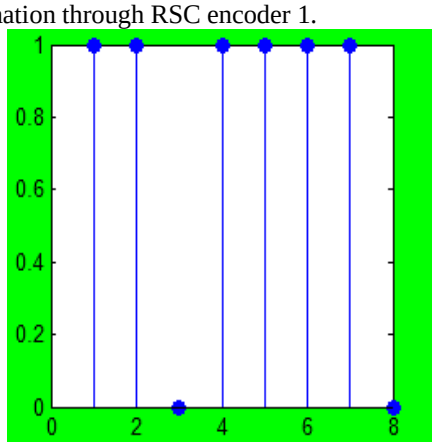

Figure 17. RSC Encoder 1

This is the output from RSC encoder 1. When we interleave the input data by using Pseudo random interleaver then we will get interleaved data, I= [1 1 1 0 1 1 1 0 0]. Now, we will apply this interleaved data to RSC encoder 2.

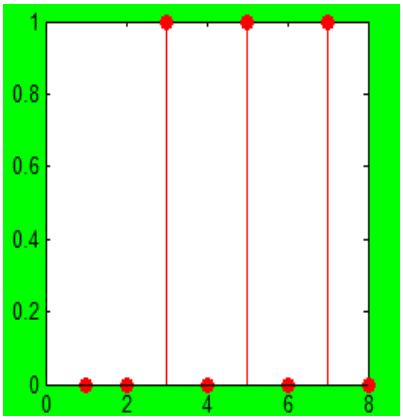

Figure 18. RSC Encoder 2

This is the output from RSC encoder 2. Now, we will multiplex systematic input, RSC 1 output and RSC 2 output.

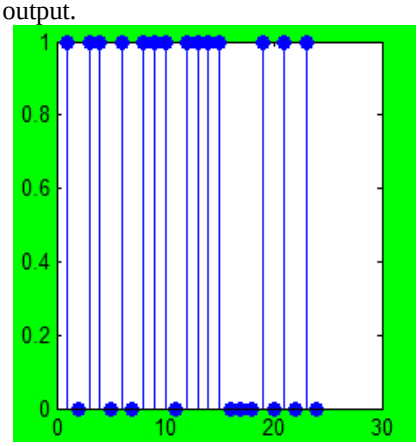

Figure 19. Multiplexed Data

This is the multiplexed data, now we will through this multiplexed data over AWGN (Additive White Gaussian Noise) channel.

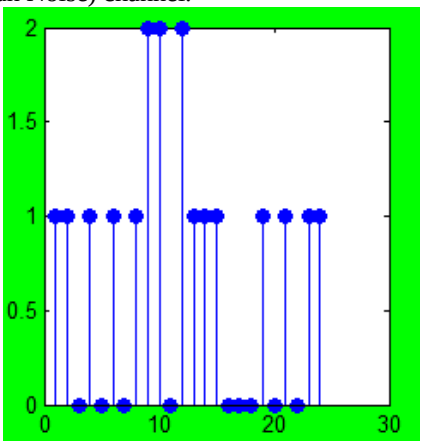

Figure 20. Received Data

This is the received data at receiving end. After passing through AWGN channel, we can see that out of 24 bits 6 bits are corrupted. We have used two parallel Viterbi decoders to decode the input data at receiving end.

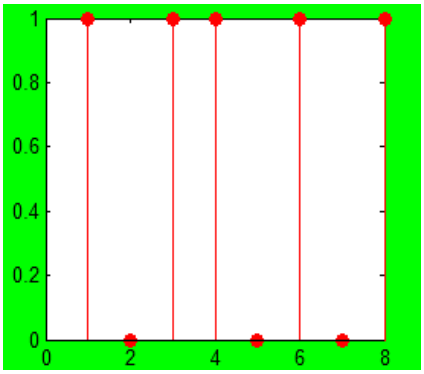

Figure 21. Viterbi Decoder 1

This is the output from Viterbi decoder 1. It is important to note that the Viterbi decoder 1 will deal with RSC encoder 1 and Viterbi decoder 2 deals with RSC encoder 2.

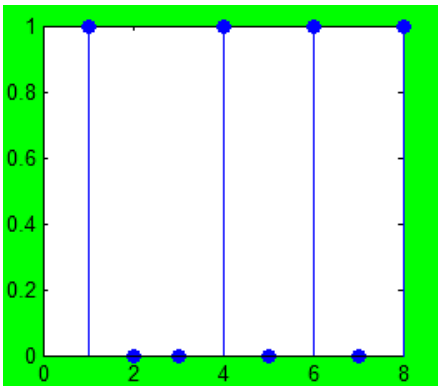

Figure 22. Viterbi Decoder 2

This is the output from Viterbi decoder 2, Viterbi decoder 2 will only deal with RSC encoder 1. Now, we will apply simple probability on systematic data, Viterbi decoder1 and Viterbi decoder 2. According to this logic, if two bits are 0 and one bit is 1 then it means that decoded output is 0 and vice versa. Finally we got the following the following sequence.

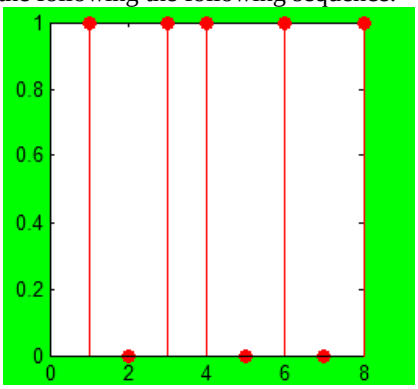

Figure 23. Decoded Data

This is our final decoded data sequence. When we compare this sequence with input data, we came to know that it is exactly matched with input data sequence which was our required data at receiver.

VI. COMPARISON

The most recent paper on convolutional encoding and Viterbi decoding using Verilog HDL [3] was published in IEEE conference in 2011. But, this research is not so valuable and effective in comparison with our research work which is explained in this paper.

In [3] research paper, simply bits are encoded using convolutional encoder at transmitting side and same bits are decoded using Viterbi algorithm at receiving end. Actually, decoding at receiving end is not a major issue but we face problem when bits are corrupted in the information channel. In [3], they have not designed an information channel also not explained the operation of Viterbi Algorithm in detail. Simply, bits are encoded at transmitting end and same bits are decoded at receiving end. They have not explained the actual operation of viterbi Algorithm as a forward error correction algorithm.

In our research work, each and every thing is explained in detail, the input data sequence has been encoded by using RSC encoder and it is transmitted over AWGN channel. A corrupted bit sequence is received at receiving end. Then by using Viterbi decoder original desired bit sequence is produced at receiver.

VII. CONCLUSION

In this paper we have presented the design and implementation of RSC encoder and Viterbi decoder. This implementation has been simulated by using MATLAB R2009a GUI (Graphical User Interface). The input data sequence has been encoded by using RSC encoder and it is transmitted over Additive White Gaussian Noise channel. A corrupted bit sequence is received at receiving end. Then by using Viterbi decoder original desired bit sequence is produced at receiver.

REFERENCES

[1.] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo-codes," in Proc. IEEE Int. Conf. Commun., May 1993, pp. 1064–1070.

[2.] K. Gracie and M.-H. Hamon, "Turbo and turbo-like codes: Principles and applications in telecommunications," Proc. IEEE, vol. 95, no. 6, pp. 1228–1254,Jun.2007.

[3.] "Implementation of Convolutional Encoder and Viterbi Decoder using Verilog HDL" V.Kavinilavu1, S. Salivahanan, V. S. Kanchana Bhaaskaran2, Samiappa Sakthikumaran, B. Brindha and C. Vinoth SSN College of Engineering, Kalavakkam, Rajiv Gandhi Salai, (Off) Chennai, Tamil Nadu, India. 978-1-4244-8679-3/11/$26.00 ©2011 IEEE.

[4.] A.J.Viterbi, "Error bounds for convolutional coding and an asymptotically optimum decoding algorithm", IEEE Tran. on Inform.Theory, Vol. 2, Pp. 260-269, Apr. 1967.

[5.] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," IEEE Trans. Commun., vol. 44, no. 10,pp. 1261–1271, Oct. 1996.

[6.] K. Ouahada and H. C. Ferreira, "Simulation study of the performance of ternary line codes under Viterbi decoding," in IEE Proc-Commun., vol. 151, no. 5, pp. 409–414, 2004.

[7.] G. David Forney, JR., "The Viterbi Algorithm," Proceedings of the IEEE, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[8.] Viterbi and J. Omura, "Principles of Digital Communication and Coding, McGraw-Hill Kogakusha LTD, Tokyo Japan, 1979.

[9.] Wong, Y.S. et.al "Implementation of convolutional encoder and Viterbi decoder using VHDL" IEEE Tran. on Inform. Theory, Pp. 22-25, Nov. 2009.

[10.] Irfan Habib, Özgün Paker, Sergei Sawitzki, "Design Space Exploration of Hard-Decision Viterbi Decoding: Algorithm and VLSI Implementation" IEEE Tran. on Very Large Scale Integration(VLSI) Systems,Vol.18,Pp.94-807,May2010.

[11.] Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error correcting Coding and decoding: Turbo codes. In Proceedings of the IEEE International Conference on Communications, Geneva, Switzerland, May 2003.

[12.] J. B. Anderson and S. M. Hladik, "An Optimal Circular Viterbi Decoder for the Bounded Distance Criterion," IEEE Trans. Commun., vol. 50, no.11, pp.1736–1742, Nov.2002.