ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

Graphical User Interface for Simulating Convolutional Coding with Viterbi Decoding in Digital Communication Systems using Matlab

Ezeofor C. J.[1], Ndinechi M.C.[2]

Lecturer, Department of Electronic and Computer Engineering, University of Port Harcourt, Rivers State, Nigeria[1]

Associate Professor, Department of Electrical & Electronic Engineering, Federal University of Technology, Owerri, Imo State, Nigeria[2]

ABSTRACT: This paper presents Graphical User Interface (GUI) for simulating convolutional coding with Viterbi decoding in digital communication system using MATLAB. Digital transmission is not free from channel impairments such as noise, interference and fading which cause signal distortion and degradation in signal to noise ratio. These introduce a lot of errors on the information bits sent from one place to another. To address these problems, Convolutional coding is introduced at the transmitter side and Viterbi decoding at the receiver end to ensure consistent error free transmission. In order to visualize the effect and evaluate the performance of the coding and decoding used, simulation programs that encode and decode digital data were designed, written and tested in MATLAB. The generated bit error rate (BER) is plotted against Energy per bit to noise spectral density (E_b/N_o) for different digital input data. It is seen that as E_b/N_o increases, bit error rate decreases thereby increasing the performance of the convolutional code rate used in the transmission channel at both end. Further analysis and discussion were made based on the MATLAB graph results obtained.

KEYWORDS: MATLAB, GUI, Convolutional coding, BER, SNR, Viterbi decoding

I. INTRODUCTION

The main aim of a digital communication system is to transmit information reliably over a channel [1]. The channel can be coaxial cables, microwave links, space, fiber optics etc. and each of them is subject to various types of noise, distortion and interference that lead to errors. Shannon proves that there exist channel-encoding methods which enable information to be transmitted reliably when source information rate R is less than channel capacity C. It is possible to design a communication system for that channel and with the help of error-control coding such as convolutional coding, one can achieve a very small probability of output error for that channel. As mentioned in [2], some forms of error control encoding that are used to recover some corrupted information are discussed. Convolutional coding is one of the channel coding extensively used for real time error detection and correction as shown in figure 1.1 [3].

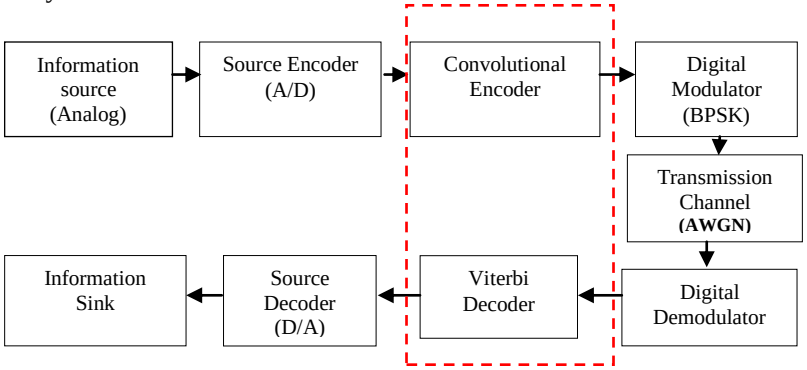

Figure 1.1: Convolutional Encoder/Decoder block diagram in digital communication system [3]

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

II. RELATED WORK

The related researched works are not limited to:

a. Performance Analysis of Convolutional Encoder Rate Change by Sw Bawane and W Gohoker (2014) which explained the modified FPGA scheme for the convolutional encoder in OFDM baseband processing systems. It shows convolutional encoder with constraint length of 9.

b. Design of a high speed parallel encoder for convolutional codes by A Msir, F Monteiro, and A Dandache (2004). In their paper, the design of high speed parallel architectures for convolutional encoders and its implementation on FPGA devices were done.

c. FPGA design and implementation of a convolutional encoder and a Viterbi decoder based on 802.11a for OFDM by Y Sun, and Z Ding (2012). They carried out a modified FPGA scheme for the convolutional encoder and Viterbi decoder based on the IEEE 802.11a standards of WLAN in OFDM based processing systems.

III. METHODOLOGY

This research work considered convolutional encoder of (n=2, k=1, K=3) as shown in figure 3.1. The generator polynomials for the chosen encoder are $g_0(D)= 1+D+D2 = 111_2$ or 7_8 and $g_1(D)=1+D2=101_2$ or 5_8. The generator polynomials that would be used depend on the convolutional encoder rate being considered.

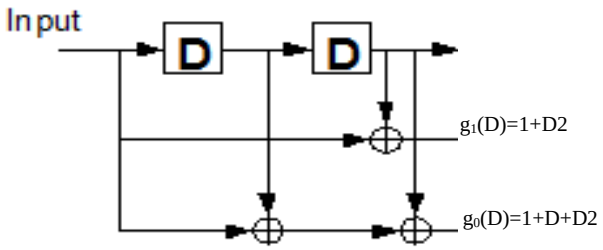

Figure 3.1: ½ Rate of Convolutional Encoder [11]

3.1. How convolutional coding is done

This can be understood using set of sequence of information bits sent serially into the ½ convolutional encoder diagram shown in figure 3.1. Let's look at the digitized sample input information bits $k = [1111010]_2 = [172]_8$. This would be carried out into different stages at different clock input. The encoder first initializes its shift-register memory D=0 and D2=0 at clock input.

Stage 1: Input k =1, O/P=11

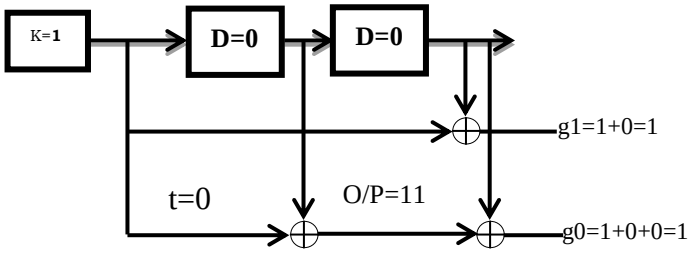

Figure 3.2a: Encoder state at stage 1

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

At stage 1, t=0: encoder takes the first input bit k=1at first clock input (from the sequence of information bits, starting from the most significant bit) and Ex-OR it with the value found in the memory D2= 0 to get $g_1(D)$ = g(1)=1+0=1. At the same time, takes the same input bit k=1 and Ex-OR with the values found in the memory D=0, then uses the result and Ex-OR with the value found in memory D2=0 to get g0(D)= g(0)=1+0+0=1. The output code word would be 11_2 as shown in figure 3.2a. The encoder shifts k value into D and D value into D2. The new D=1 and D2=0.

Stage 2: Input k=1, O/P=01

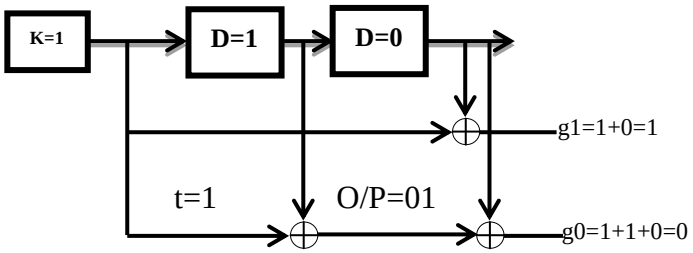

Figure 3.2b: Encoder state at stage 2

At stage 2, t=1, encoder takes the second input bit k=1 at next clock input (from the sequence of information bits, starting from the most significant bit) and Ex-OR it with the value found in the memory D2= 0 to get $g_1(D)$ = g(1)=1+0=1. At the same time, takes the same input bit k=1 and Ex-OR with the values found in the memory D=1, then uses the result and Ex-OR with the value found in memory D2=0 to get g0(D)= g(0)=1+1+0=0. The output code word would be 01_2 as shown in figure 3.2b. The encoder shifts k value into D and D value into D2. The new D=1 and D2=1.

Stage 3: input k=1, O/P=10

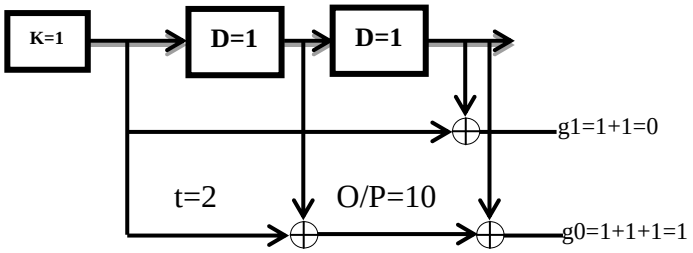

Figure 3.2c: Encoder state at stage 3

At stage 3, t=2, encoder takes the second input bit k=1(from the sequence of information bits, starting from the most significant bit) and Ex-OR it with the value found in the memory D2= 1 to get $g_1(D)$ = g(1)=1+1=0. At the same time, takes the same input bit k=1 and Ex-OR with the values found in the memory D=1, then uses the result and Ex-OR with the value found in memory D2=1 to get g0(D)= g(0)=1+1+1=1 as shown in figure 3.2c. The output code word would be 10_2. The encoder shifts k value into D and D value into D2. The new D=1 and D2=1.

The process continues until stage 4 at t=3, stage 5 at t=4, stage 6 at t=5, and stage 7 at t=6 are done. After the sequence of bits has been encoded, the encoder needs to be flushed or reset to retain its 00 state. Stage 8 and stage 9 perform encoder reset process. The output code word would be 00_2. Thus the encoded output sequence for the 7-bits input sequence [1111010] is [11011010010010] plus flushing bits added as shown in table 3.1. This is done at the transmitter's side before it would be sent to the channel.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

Table 3.1: Encoder values and State Transition Table look up

Time t	Input bit k	m_0	m_1	Output g0	Output g1	Output Code word	Current State	Next State	Output Bits
$t_{0=0}$	1	0	0	1	1	11	00	10	11
$t_{1=1}$	1	1	0	0	1	01	10	11	01
$t_{2=2}$	1	1	1	1	0	10	11	11	10
$t_{3=3}$	1	1	1	1	0	10	11	11	10
$t_{4=4}$	0	1	1	0	1	01	11	01	01
$t_{5=5}$	1	0	1	0	0	00	01	10	00
$t_{6=6}$	0	1	0	1	0	10	10	01	10
Flushing bits	0	0	1	1	1	11	01	00	11
	0	0	0	0	0	00	00	00	00

The convolutional encoder uses look-up tables called state transition table to do the encoding which are shown in table 3.2. With this, the encoder knows the current and next states during operation.

Table 3.2: State Transition Table for Current & Next

	NEXT STATE, IF	
CURRENT STATE	INPUT =0;	INPUT =1;
00	00	10
01	00	10
10	01	11
11	01	11

3.2 How Viterbi decoding is done

Viterbi decoding uses trellis diagram to decode convolutional encoded data at the receiver's end. It has the encoding knowledge of convolutional encoder and that enable it to perform its decoding. Two forms of Viterbi decoding are hard and soft decision Viterbi decoding. Hard decision Viterbi decoding also known as Soft Input Viterbi decoding technique (SIVD) uses a path metric called the hamming distance metric to determine the survivor path and the decoder output through the trellis. Soft decision Viterbi decoding calculates the distance between the received symbol and the probable transmitted symbol and determine its output. That is, if transmitted coded bit is 0, Euclidean distance is,

$$ed_0=(y--\sqrt{E_c})^2 = (y^2+2y\sqrt{E_c}+E_c)^2 \quad (2.1)$$

If transmitted coded bit is 1, Euclidean distance is $ed_1=(y-\sqrt{E_c})^2 = (y^2-2y\sqrt{E_c}+E_c)^2$ (2.2)

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

The terms 2,y^2, $\sqrt{E_c}$ and E_c are common in both the equations they can be ignored. The simplified Euclidean distance is,

ed_0 = +y and ed_1 = -y. (2.3)

Let's assume that at the receiver, the bits were [1101101001010]. Error would be detected at t=5 as shown in the table 3.4.

Table 3.4: Input, output & Received bits with Errors

Time	t0	t1	t2	t3	t4	t5	t6
Encoder Input	1	1	1	1	0	1	0
Encoder Output	11	01	10	10	01	00	10
Received (assumed)	11	01	10	10	01	01	10
Errors (assumed)						x	

The table 3.4 comprises of the encoder input, encoder output, and assumed received bits with error marked in red colour. The Trellis diagram drawn for the 7-bits input stream [1111010] is in figure 3.4 for each time tick based on the example considered.

From the trellis diagram in figure 3.4, S0 to S3 represents state of the encoder, the hamming distance of state 0 through state 3 are calculated thus:

At t0

Moving from state 00 to state 00, the output = 00 and the received bits =11, the hamming distance = 2

Moving from state 00 to state 10, the output = 11 and the received bits =11, the hamming distance = 0. Therefore the shortest hamming distance is chosen and that is 0. At t0, HD=S2=0

At t1

Hamming distance S0= S0+S0 =2 +1 =3

Hamming distance S1= S0-S2 +S2-S1= 0+2 =2

Hamming distance S2= S0-S0 +S0-S2= 2+1 =3

Hamming distance S3= S0-S2 +S2-S3= 0+0 =0

At t1, HD=S3=0

At t2

Hamming distance S0=3+1=4 or 0+2+1=3; 3 (the highest is discarded while the least chosen)

Hamming distance S1= 2+1+0=3 or 0+0+2=2;2

Hamming distance S2= 2+1+1=4 0r 0+2+1=3;3

Hamming distance S3= 2+1+2=5 or 0+0+0=0;0

At t2, HD=S3=0

At t3

Hamming distance S0=0+2+1+1=4 or 0+0+2+1=3;3

Hamming distance S1= 0+2+1+0=3 or 0+0+0+2=2;2

Hamming distance S2= 0+0+2+1=3 or 0+2+1+1=4;3

Hamming distance S3= 0+2+1+2=5 or 0+0+0+0=0;0

At t3, HD=S3=0

At t4

Hamming distance S0=0+0+2+1+1=4 or 0+0+0+2+1=3;3

Hamming distance S1= 0+0+2+1+2=5 0r 0+0+0+0+0=0;0

Hamming distance S2= 0+0+2+1+1=4 or 0+0+0+2+1=3;3

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

Hamming distance S3= 0+0+2+1+0=3 or0+0+0+0+2=2;2
At t4, HD=S1=0

At t5

Hamming distance S0=0+0+0+2+1+1=4 or 0+0+0+0+0+1=1;1
Hamming distance S1= 0+0+0+0+2+0=2 or 0+0+0+2+1+2=5;2
Hamming distance S2= 0+0+0+0+0+1=1 or 0+0+0+2+1+1=4;1
Hamming distance S3= 0+0+0+2+1+0=3 or 0+0+0+0+2+2=4;3
At t0, HD=S2=1

At t6

Hamming distance S0=0+0+0+0+0+1+1=2 or 0+0+0+0+2+0+1=3;2
Hamming distance S1= 0+0+0+0+0+1+0=1 or 0+0+0+2+1+0+2=5;1
Hamming distance S2= 0+0+0+0+2+0+1=3 or 0+0+0+0+0+1+1=2;2
Hamming distance S3= 0+0+0+0+0+1+2=3 or 0+0+0+2+1+0+0=3;3
At t0, HD=S1=1
At this stage, the decoder would detect that error occurred at state 1 at time t=5.

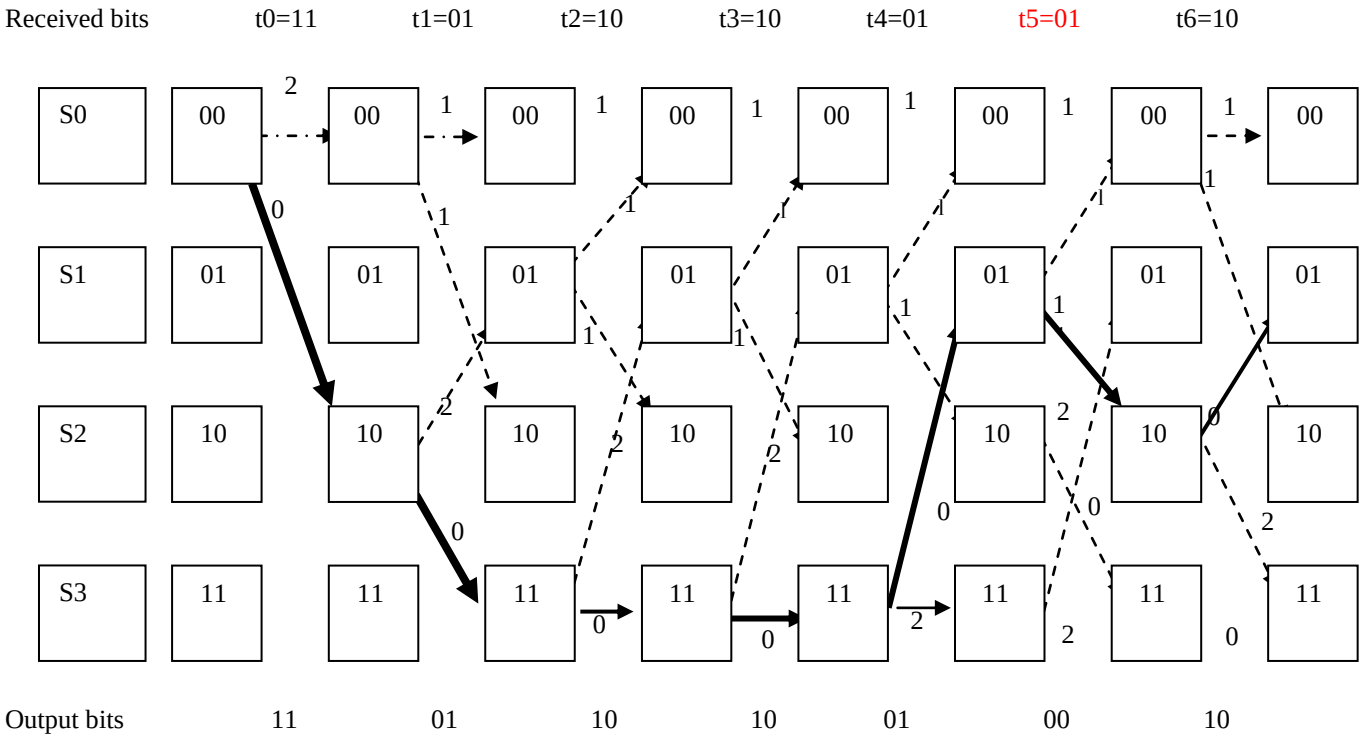

Figure 3.4: Trellis diagram for decoding transmitted 7 input bits message

3.3 Branch and Path Metrics Computation

The path and branch metrics for all the states from 0 to 3 can be calculated as shown in equation 3.1 through equation 3.12. The movement from one state to another is clearly stated in indicated in figure 3.5.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

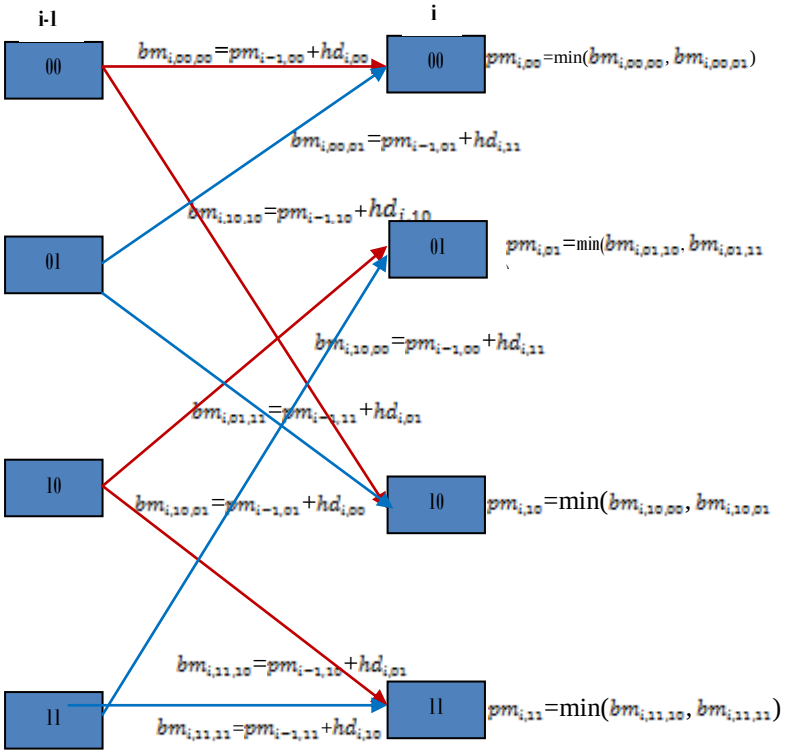

Figure 3.5: Branch and path metrics block diagram

i. State 00 can be reached from two branches

(a) State 00 with output 00. The branch metric for this transition is,

$$bm_{i,00,00}= pm_{i-1,00}+hd_{i,00} \quad (2.4)$$

(b) State 01 with output 11. The branch metric for this transition is,

$$bm_{i,00,01}= pm_{i-1,01}+hd_{i,11}. \quad (2.5)$$

The path metric for state 00 is chosen based which is the minimum out of the two.

$$pm_{i,00}= \min (bm_{i,00,00}, bm_{i,00,01}) \quad (2.6)$$

The survivor path for state 00 is stored in survivor path metric.

ii. State 01 can be reached from two branches:

(c) State 10 with output 10. The branch metric for this transition is,

$$bm_{i,10,10}=pm_{i-1,10}+hd_{i,10} \quad (2.7)$$

(d) State 11 with output 01. The branch metric for this transition is,

$$bm_{i,01,11}=pm_{i-1,11}+hd_{i,01} \quad (2.8)$$

The path metric for state 01 is chosen based which is the minimum out of the two.

$$pm_{i,01}=\min (bm_{i,01,10}, bm_{i,01,11}) \quad (2.9)$$

The survivor path for state 01 is stored in survivor path metric.

iii State 10 can be reached from two branches:

(e) State 00 with output 11. The branch metric for this transition is,

$$bm_{i,10,00}=pm_{i-1,00}+hd_{i,11} \quad (2.10)$$

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

(f) State 01 with output 00. The branch metric for this transition is,

$bm_{i,10,01}=pm_{i-1,01}+hd_{i,00}$ (2.11)

The path metric for state 10 is chosen based on which is the minimum out of the two.

$pm_{i,10}$=min ($bm_{i,10,00}$, $bm_{i,10,01}$). (2.12)

The survivor path for state 10 is stored in survivor path metric.

iv. State 11 can be reached from two branches:

(g) State 10 with output 01. The branch metric for this transition is,

$bm_{i,11,10}=pm_{i-1,10}+hd_{i,01}$ (2.13)

(h) State 11 with output 10. The branch metric for this transition is,

$bm_{i,11,11}=pm_{i-1,11}+hd_{i,10}$ (2.14)

The path metric for state 11 is chosen based which is the minimum out of the two.

$pm_{i,11}$=min ($bm_{i,11,10}$, $bm_{i,11,11}$) (2.15)

The survivor path for state 11 is stored in survivor path metric.

3.4 How does Viterbi detect error and correct them during decoding

Viterbi decoder always has knowledge of coding tactics used by the convolutional encoder before it can decode any information bits. The encoder detects error by comparing the output symbol from the received bits at the receiver's side. When the bits symbol is the same, it detects no error but when they are not the same, it detects error. It corrects the error based on the coding information values stored and that makes it intelligent.

The accumulated metrics for the full 7-bit message at each time t (plus two flushing bits) are listed in table 3.5.

Table 3.5: Accumulated Metric Value for State 0 to State 3

Current states	**PREVIOUS STATES (PREDECESSORS)**						
Current states	t_0	t_1	t_2	t_3	t_4	t_5	t_6
State 00_2	0	0	0	1	1	1	1
State 01_2	0	0	2	3	3	3	3
State 10_2	0	0	0	1	1	1	1
State 11_2	0	0	2	3	3	3	2

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

Table 3.6: States selected when tracing the path back

Time	t0	t1	t2	t3	t4	t5	t6
State 00_2	0	2	3	3	3	3	1
State 01_2	0	0	2	2	2	0	2
State 10_2	0	0	3	3	3	3	1
State 11_2	0	0	0	0	0	2	3

3.5. Traceback Unit

Once the survivor path is computed $\frac{N}{2}$+K-1 times (N is the number of output and K is the constraint length), the decoding algorithm can start trying to estimate the input sequence. Thanks to presence of tail bits (additional K-1 zeros), it is known that the final state following Convolutional code is State 00. So, start from the last computed survivor path at index $\frac{N}{2}$+K-1 for State 00. From the survivor path, find the previous state corresponding to the current state. From the knowledge of current state and previous state, the input sequence can be determined from the given table. Continue tracking back through the survivor path and estimate the input sequence till index = 1 and the states selected when tracing the path back through the survivor paths is shown in table 3.6.

IV. RESULT AND DISCUSSION

The graphical user interface designed in MATLAB is as shown in figure 4.1. Simulation was run for ½ Convolutional codes with different input messages. Error performance analysis is checked by plotting bit error-rate versus energy per bit to noise power spectral density (Eb/No) for AWGN channel. Soft decision Viterbi decoding offers better performance results than hard decision Viterbi decoding. As the number of input message increases, Convolutional coding and decoding performs better. BER of 10^{-5} was taken for both hard and soft decision and the transmitting power for soft decision is 6dB while that of hard is 8dB. Also the coding gain of soft decision decoding is greater than hard decision decoding which proved that soft decision decoding is always at least 2dB better responses as to compare to hard decision decoding. Thus to achieve the same BER; the soft decision decoding will require a lower signal to noise ratio, that is, lower transmitter power compared to its counterpart. More explanations were done on each of the plotted graph.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

Now using the simulation GUI interface, different digit numbers were entered but only 350 bits information is shown in figure 4.1.

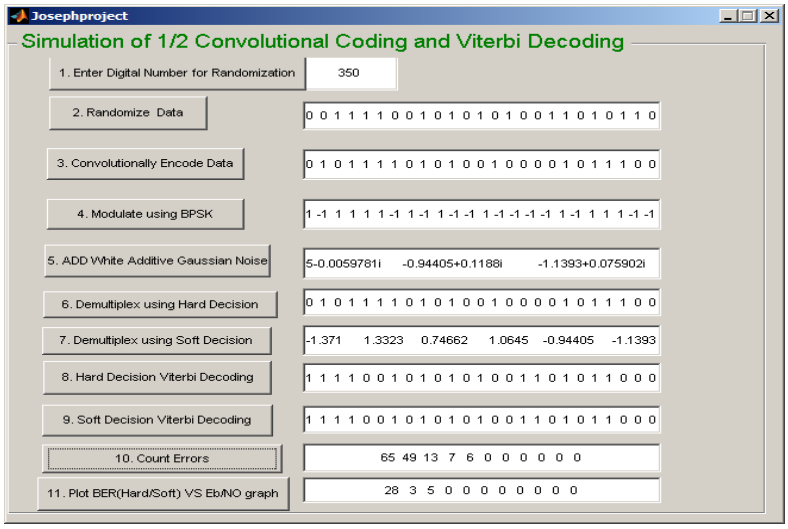

Figure 4.1: Coding and decoding Information message of 350 digit number displays

The 80000 bits message was encoded and decoded in MATLAB and the graph of BER ranging from 10^{0} to 10^{-6} versus Eb/No ranging from 0dB to 12dB plotted as shown in figure 4.2.

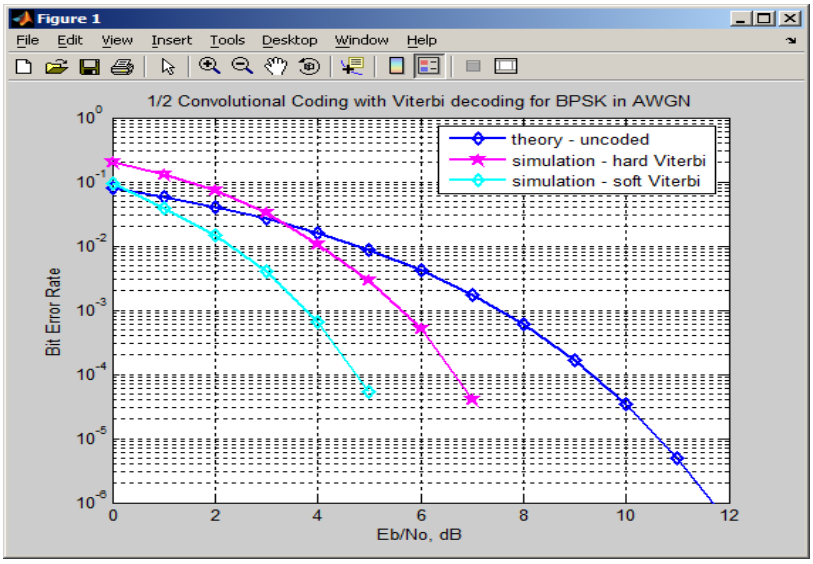

Figure 4.2: 80000 bits message; Graph of BER VS Eb/No

If Eb/No is further increased to 9.5dB, the BER of Soft Decision Viterbi Decoding (SDVD) decreases faster than Hard Decision Viterbi Decoding (HDVD). The coding gain of both can be calculated thus; Taking the measure at BER $=10^{-4}$
The Coding gain for HDVD = $SNR_{uncoded}$ - SNR_{hdvd} = 9.5dB – 6.8dB= 2.7dB
Coding gain for SDVD = $SNR_{uncoded}$ - SNR_{sdvd} = 9.5dB – 4.8dB = 4.7dB
Therefore Coding gain of SDVD – Coding gain of HDVD = 4.7dB - 2.7dB = 2dB

Another 1050000 bits message was encoded and decoded and the output graph plotted is shown in figure 4.3.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

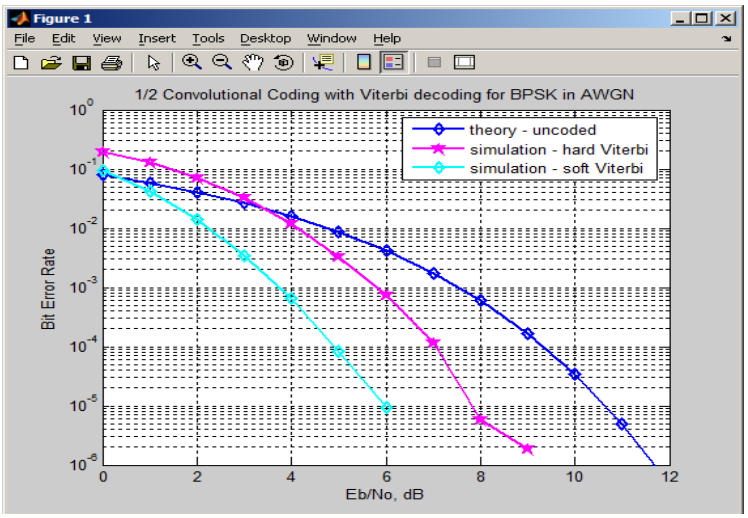

Figure 4.3: BER Vs EbNo graph plot of 1050000 information bits

From the graph shown in figure 4.3, it is seen that as the Eb/No is increased from 0 to 12dB, the BER of Soft Viterbi decoding (SDVD) decreases faster than Hard Viterbi decoding (HDVD) during decoding . This means that increase in Eb/No reduces the bit error rate of the signal thereby introduces less error in the transmission systems. Take a look at Eb/No equal to 9.5dB in figure 4.3, the coding gain of both of hard and soft Viterbi decoding can be calculated thus;
Taking measurement at BER $=10^{-5}$
Coding gain for HDVD = $SNR_{uncoded}$ - SNR_{hdvd} = 9.5dB – 6.8dB= 2.7dB
Coding gain for SDVD = $SNR_{uncoded}$ - SNR_{sdvd} = 9.5dB – 4.8dB = 4.7dB
Therefore Coding gain of SDVD – Coding gain of HDVD = 4.7dB - 2.7dB = 2dB
Therefore, the soft Viterbi decoding performs better than the Hard Viterbi decoding in decoding convolutional coded bits.

V. CONCLUSION

The objective of this paper work is to design and program MATLAB graphical User Interface for simulating ½ rate convolutional encoder with Viterbi decoder. The encoding process was demonstrated using a (2, 1, 3) convolutional encoder and decoding process demonstrated also using a hard decision Viterbi decoder and a soft decision Viterbi decoder. Different graphs of BER against Eb/No were plotted to check the number of errors that would be reduced within the transmitting powers ranging from 0dB to12dB. As was seen from the simulation graph results obtained in figure 4.2 and figure 4.3, the performance of convolutional coding with Viterbi decoding was greatly improved by the smaller code rate used.

REFERENCES

[1] Shannon, C. E., "A Mathematical Theory of Communication", Bell Syst. Technology. J., Vol. 27, pp.379- 423, 623-656, 1948.
[2] Bossert M, "Channel Coding for Telecommunications", New York, NY: John Wiley and Sons, 1999.
[3] Clark, G. C., Jr., Cain, J. B., "Error-Correction Coding for Digital Communications", Plenum Press, New York, 1981.
[4] Bernard S., "Digital Communications-Fundamental and Applications", 2nd Edition, New Jersey Prentice Hall, 2001.
[5] Benedetto S. G. Montorsi, "Design of parallel concatenated convolutional codes", IEEE Transactions on Communications, vol. 44, 1996.
[6] Berrou C., Glavieux A., Thitimajshima P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes", in Proceedings of the International Conference on Communications, (Geneva, Switzerland), pp. 1064–1070, 1993.
[7] Blahut R. E, "Theory and practice of error control codes", Reading, Massachusetts: Addison-Wesley Publishing Company, 1983.
[8] Chase D., "A class of algorithms for decoding block codes with channel measurement information", IEEE Trans. Inform. Theory, vol. IT-18, no. 1, pp. 170–182, 1972.
[9] Elias P., "Coding for noisy channels", International Convention Record (Part IV), pp. 37-46, 1955.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Volume 3, Issue 7, July 2014

[10] Forney G. D., "Convolutional codes with algebraic structure" IEEE Trans. Inform. Theory, IT-16, pp. 720-738, 1970.
[11] Gallager R. G, and Elias P., "Introduction to Coding for noisy channels", in The Electron and the Bit, J. V. Guttag, Ed. Cambridge, MA: EECS Dept., MIT Press, pp. 91–94, 2005.
[12] MacKay D., "Good error correcting codes based on very sparse matrices", IEEE Trans. Information Theory, pp. 399, 1999.
[13] Ming-Bo .L.,"New Path History Management Circuits for Viterbi Decoders", IEEE Transactions on Communications, vol. 48, pp. 1605-1608, 2000.
[14] Morelos-Zaragoza R. H, "The art of error correcting coding", John Wiley & Sons, ISBN 0471495816, 2002.
[15] Shaina S., Ch. Kranthi, Faisal Bala, "Performance Analysis of Convolutional Encoder and Viterbi Decoder Using FPGA" International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 6, December 2012.
[16] Viterbi A. J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE. Transaction of Information Theory, vol. IT-13, pp. 260-269, 1967.
[17] Wicker S. B., "Error Control Systems for Digital Communication and Storage", Prentice Hall, Englewood Cliffs, New Jersey, 1995.