International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

A Robust Turbo Codec Design for Satellite Communications

Dr. V Sambasiva Rao
Professor,
ECE Department
PES University, India

Ranjitha S
M.Tech Student,
ECE Department
PES University, India

***Abstract*— Satellite communication systems require forward error correction techniques for their correct functioning. Satellite communications necessitate an efficient error control technique which can provide high reliability, low coding redundancy, and high coding gain. Coding technique is needed to send or receive information from ground station with minimal or no error. In this paper, turbo coding, a very dominant and robust error correcting coding scheme is presented. The reformed Viterbi algorithm called SOVA (Soft Output Viterbi Algorithm) which is used at the decoder is also presented. Efficiently designing the turbo codec based on CCSDS defined standards and parameters has been the focus of this paper.**

The modern satellite communication systems also demand low power and lesser area design techniques. Optimization of area reduces the delay and henceforth increases the device speed which is a critical parameter while designing satellites and employing the error control techniques in spacecraft. Implementing the turbo codec on FPGA Virtex 5 by optimizing the logic design and minimizing the area requirements for the codec design has been thereby addressed in this paper.

Keywords*— Forward error Correction, Coding Gain, Turbo Encoder, SOVA Decoder, Satellite Communications, FPGA.

I. INTRODUCTION

Turbo codes from past few years have consistently shown to hold the ability to achieve near Shannon-limit error correction performance with noticeably very reasonable encoding and decoding complexity when compared to other typical error controlling codes which were conventionally employed in satellite communications [1][5].Consultative Committee for Space Data Systems (CCSDS) has therefore adopted turbo codes as the new standard for telemetry channel coding based on its outstanding performance [6]. Turbo codecs are being widely employed in satellite communications at present since they achieve high gain with much lesser complexity compared to other primitive codes like convolutional codes [8].Turbo codes are not derived from completely new concepts but have been developed by modification of existing concepts like convolutional coding and Viterbi decoding. The outstanding properties of this coding technique has thereby paved way to wider research on turbo codecs and hence in this paper a conducive work on simplifying the turbo codec design and optimizing the algorithm for satellite communications has been addressed. Turbo codes in order to be suitable for satellite systems have to be capable of being implementable on hardware with the design holding low processing power and minimized physical area requirements but at the same time codec should be able to provide high throughput and high coding to make it robust. Therefore maintaining the trade-off between performance of the codec and the area requirement is the key part.

Algorithmic implementations are mostly suitable on reconfigurable hardware for satellite communications. Among various reconfigurable hardware available, FPGAs take the prime place. FPGAs are being extensively used for rapid prototyping and likewise for implementation of various satellite applications without conceding power, area and speed performance with significantly reduced time-to-market and low cost factors [2]. Therefore this paper focuses on implementing the turbo codec design on FPGA. This paper presents the design and simulation result of the turbo codec and also discusses the technique employed in implementing the constituent encoder and decoder on the FPGA by optimizing the design and minimizing the area.

The turbo encoder and decoder are designed on MATLAB-Simulink R2015a using system blocks. The deterministic interleaving technique standardised by CCSDS was employed for this design. Simulation is carried out using Xilinx ISE simulator tool and codec has been implemented on FPGA Virtex 5. Design is written using Verilog HDL.

II. TURBO ENCODER

A turbo encoder is constructed by parallel concatenation of two identical convolutional codes of special type, such as, recursive systematic convolutional encoders (RSC) [3][5]. Each RSC is termed as component encoder. Both the component encoders are disjointed by an interleaver. Using turbo encoder with interleaver is to predominantly reduce the low-weight code words. One of the two encoders may possibly perhaps intermittently produce a low-weight code output, but the probability that both of the component encoders in chorus generate a low-weighted output is extremely trivial.

Each component encoder is designed to accept only one bit of input message stream at a given instant of time and thereby will produce encoded output bits which are to be transmitted over the defined channel. In this run-through, for one input bit the encoder produces more than one output bit and these extra bits in output configuration makes the transmitted data more protected against the noise in the channel. These extra bits encompassed with information bits are called redundant bits, which are used to detect and correct the errors in the received bits arrangement [5]. The code rate of each RSC encoder is ½.The RSC encoding process therefore will twofold the number of input bits at the output for a code rate of ½. For instance, 8-bit input stream will be converted into a 16-bit output stream.

The two constituent encoders are actually encoding the same information bits stream, both in a different order. For each input binary information word, at the output the systematic output is obtained as it is, followed by the parity check bit from the 1st RSC encoder, which is then followed by the parity check bit from the 2nd RSC encoder. All these symbols are then multiplexed in order to form the following turbo-coded sequence: {..., x-Þ1-Þ2, x- Þ1-Þ2, x-Þ1-Þ2...} where x is the systematic output bit, Þ1 is the parity bit from 1st RSC encoder, Þ2 is the parity bit from 2nd RSC encoder. Code rate consequently becomes 1/3 instead of ½.

Here, the turbo encoder uses two RSC encoders each of short constraint length in order to avoid excessive decoding complexity. Important design parameters considered:
1.Type of component codes= Recursive systematic convolutional codes
2. Code type= Systematic parallel concatenation turbo code
3. Number of RSC encoders used=2
4. Interleaver type= Algorithmic
5. Code rate=1/3
6. Constraint length=3
(CCSDS has recommended constraint lengths of 3, 5, and 7 for turbo codecs but to simplify the design for hardware implementation constraint length of 3 is considered)

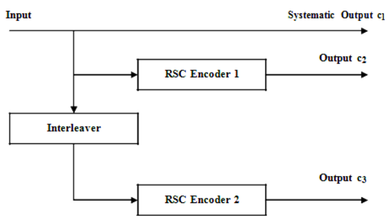


Fig.1 Turbo Encoder Structure

In Fig.1 the 1st RSC encoder outputs two sequences one is the systematic output *c1* and the other is the recursive convolutional encoded output sequence *c2* and the 2nd RSC encoder discards its systematic outputs and will only output the recursive convolutional encoded output sequence *c3*.

A. Recursive Systematic Convolutional (RSC) encoder

The RSC encoder is realized from the common conformist non-recursive non- systematic convolutional encoder by doing a simple operation of providing a feedback from output to input i.e., feeding back one of its encoded outputs again back to its input. The conventional encoder can be transformed into an RSC encoder by feeding back the first output to the input. The generator matrix of the encoder then becomes G = [1, *g*1/*g*2] where g1, g2 are the generator polynomials used to specify the hardware connections. Here g1= [111] and g2 = [101]. A “1” denotes a connection and a “0” denotes no connection [3].

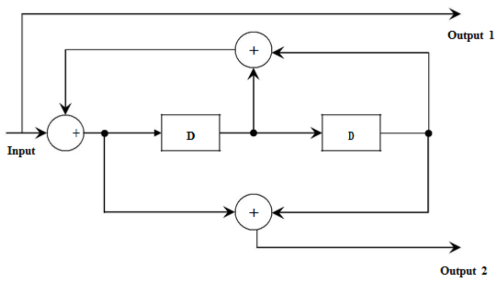


Fig.2 RSC encoder obtained from the conventional convolution encoder with code rate r = 1/2 and K = 3

B. State Diagram Representation

Fig.3 shows the state diagram representation of a component encoder. The state diagram exemplifies each of the state information of the encoder. The state information of a recursive convolutional encoder is stored in the shift registers.

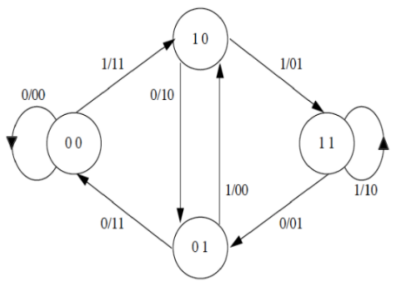


Fig.3 State diagram representation of the encoder

C. Interleaver Design

The important goal for using the interleaver is to deliver randomness to the input sequences and also to increase the code weights. The reason for interleaving is to shield the data from burst errors [3][8]. This can be described by looking at the interleaving step as a progressive permutation of information bits. Specifically for satellite application, interleaver using memories are avoided to save look-up table space, simple algorithmic interleaver is employed. Flowchart for the interleaver design is given below in fig.4.

Here in this work, as given fig.4 flow chart, length of each input frame or sequence is choosen as N=8.
Prime number 3 is chosen=>p=3.
Original sequence order q= {0, 1, 2, 3, 4, 5, 6, 7}. Using equation (p*q) modN=> the interleaved sequence bit positions is given by (3*q) mod8 => q*= {0, 3, 6, 1, 4, 7, 2, 5}.

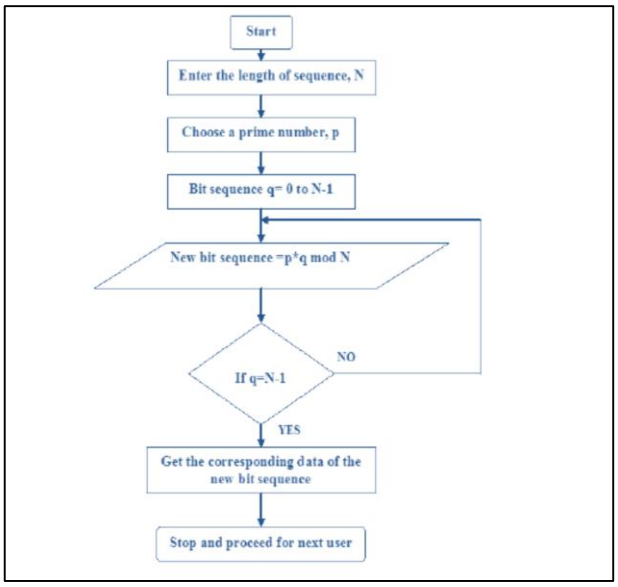

Start
Enter the length of sequence, N
Choose a prime number, p
Bit sequence q= 0 to N-1
New bit sequence =p*q mod N
If q=N-1
NO
YES
Get the corresponding data of the new bit sequence
Stop and proceed for next user

Fig.4 Deterministic interleaving flow chart

III. TURBO DECODER

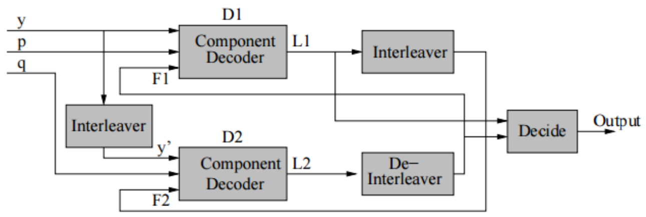


Fig.5 Turbo decoder structure

In general, turbo decoder will contain two component decoders as discussed in earlier sections as given in fig.5. The data stored in the flip flops of a component decoder can be regarded as decoder state, with state changes defined via a state machine which starts from state 0. The state machine inputs a bit and outputs the parity bit together with input bit at each time step. After input bit followed by parity bits are sent through a channel, they ultimately arrive at the turbo decoder as received values possibly altered during transmission across the channel. The received bits maybe be altered by the channel noise, and hence the received values may not match their transmitted counterparts. The turbo decoder attempts to reconstruct transmitted bits through a series of decoding steps, a typical turbo decoder consists of two identical component decoders, D1 and D2, interleaver/de-interleaver blocks, and an output decision block.

A. SOVA decoding algorithm

Soft Decision Viterbi Decoding (SOVA) is a decoding method akin to the Viterbi algorithm [4][7]. There are two main differences between the conventional Viterbi algorithm and the SOVA [2][7].

• Firstly, the path metrics are transformed to use a “priori information” when determining the path through the trellis that is most likely.

• Secondly, the soft output has reliability information about the decoded output. This reliability information is the “a posteriori log-likelihood ratios”.

The turbo decoder is built on this modified Viterbi algorithm that incorporates soft-input and soft-output values beside the channel reliability values to improve decoding performance. From conventional SOVA algorithm one major change made is elimination of unlikely paths to save area during implantation.

B. SOVA Hardware Architecture

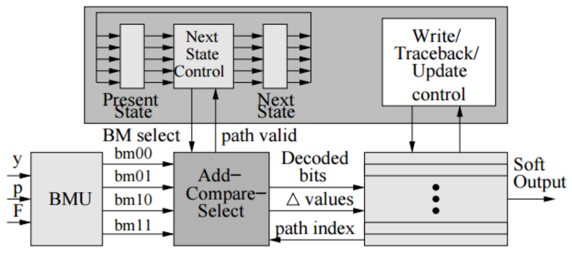


Fig.6 Constituent decoder architecture

- The Branch metric unit (BMU) calculates the branch metrics along each of the trellis stage
- The Add-Compare-Select unit (ACSU) will take the previous path metric from memory and add the branch metric to it calculated by BMU, results in new path metric which is stored in the survivor memory unit. ACSU then compares all the path metrics along each of the trellis stage by using a threshold value
- One major design strategy followed is elimination of unlikely paths. Trellis paths which are not likely to lead to decoded output sequence is eliminated and paths thus preserved are the survival paths thus incredibly saving the memory space
- TBU will calculate the path with highest metric and store it in SMU
- TBU will use the control signal to check if the path is valid and if yes, then the decoded bits are obtained based on the sign of likelihood ratio
- If the sign is detected as positive the bit is decoded as “1”, v hi9i9i9iand if it is negative the bit is decoded as “0”.

The proposed architecture uses relatively less memory since at every instant unlikely and least probabilistic paths are eliminated and also only the current metrics are stored and previous path metrics at each stage is automatically updated once the present metrics are calculated, this saves the use of numerous registers and simplifies the design and saves memory space.

C. Trellis Diagram Representation

Trellis is like a tree with remerging structured branches. A trellis diagram of a component encoder is acquired from its state diagram as given in fig.7. In the trellis diagram, each node signifies a distinct state at a given instant of time and designates an imaginable pattern of recently received bits [2]. The shifts which take place between the states are specified

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 12, December-2016

by the branches that are regarded as with the corresponding output. The solid lines in the trellis diagram denote sequence when input bit “0” and the dotted lines denote sequence when an input bit “1” [2][4][8].

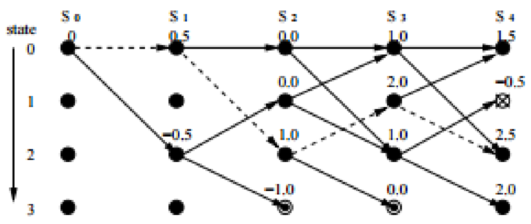


Fig.7 Trellis diagram for SOVA component decoder

IV. IMPLEMENTATION RESULTS

. The device utilization summary of synthesis report shows that the area utilized by both the decoder architecture is relatively very less, thus making it an efficient architecture. The implementation summary shown in table 1 and table 2 reports efficient utilization of area on the device, in comparison with previously reported architectures. Efficient area utilization in-turn resulting in optimized design of the turbo codec has been one of the major highlights of this work.

A. Implementation Result of Turbo Encoder

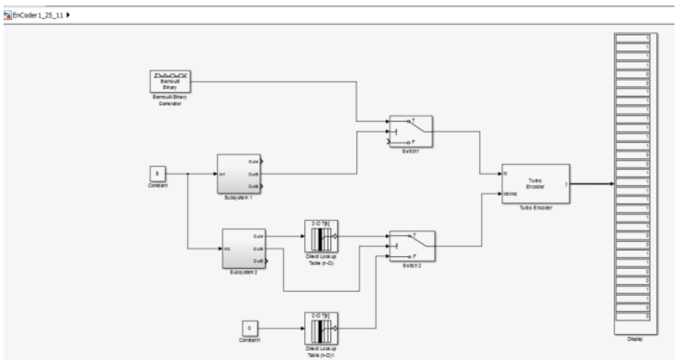


Fig.8 Simulink model of turbo encoder showing the encoded bits

- Bernoulli Binary Generator – This block is used for generating the input bit stream. In this work, 8 bits stream is used as the input the Encoder.
- Constant Block – This block has a value of 8.This block specifies the input word length.
- Subsystem 1 and Subsystem 2 – This block is used for generating the signals like new code word and valid output, which will indicate the validity of the input signal like the input codeword and address for interleaver.
- Direct Lookup Table – This is the lookup table which contains the address of the interleaver. This address is used by the interleaver for mixing of the input bits. The second direct lookup table is used to store the default sequence to the interleaver. The sequence is calculated using CCSDS algorithm as specified in the previous section
- Switch 1 and Switch 2 – These switches are used for letting the input pass or stop w.r.t the signal generated by the subsystem 1 and 2.
- Turbo Encoder – This block takes the input generated i.e, input bit stream and the input interleaver address and produces the encoded signal. This block uses the concept of tail biting; tail bits are used to move coder to known state (state of all zeros).
- Display – This block is used for displaying the output produced by the turbo encoder.

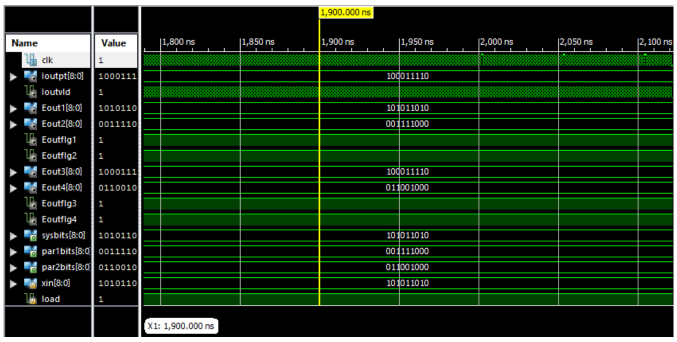
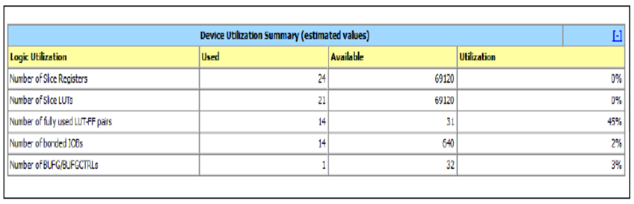

```
Name            Value     1,800 ns  1,850 ns  1,900 ns  1,950 ns  2,000 ns  2,050 ns  2,100 ns
clk             1
Ioutpt[8:0]     1000111                       100011110
Ioutvld         1
Eout1[8:0]      1010110                       101011010
Eout2[8:0]      0011110                       001111000
Eoutflg1        1
Eoutflg2        1
Eout3[8:0]      1000111                       100011110
Eout4[8:0]      0110010                       011001000
Eoutflg3        1
Eoutflg4        1
sysbits[8:0]    1010110                       101011010
par1bits[8:0]   0011110                       001111000
par2bits[8:0]   0110010                       011001000
xin[8:0]        1010110                       101011010
load            1
```

Fig.9 Waveform output for turbo encoder design

TABLE I: DEVICE UTILIZATION SUMMARY OF THE TURBO ENCODER



Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	24	69120	0%
Number of Slice LUTs	21	69120	0%
Number of fully used LUT-FF pairs	14	31	45%
Number of bonded IOBs	14	640	2%
Number of BUFG/BUFGCTRLs	1	32	3%

B. Implementation Result of Turbo Decoder

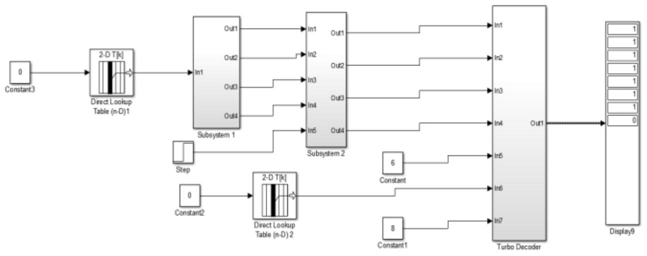


Fig.10 Simulink model of turbo decoder showing the decoded bits

- Bernoulli Binary Generator – This block is used for generating the input bit stream. Have used 24 bits stream as the input the Encoder.
- Constant Block – Constant1 block has been assigned a value of 8.This block specifies the input word length. Constant2 and Constant3 block is used to feed the input to the direct look up table.
- Subsystem 1 - This block is used for dividing the signals into small input bit streams, which are then input to the subsystem
- Subsystem 2 – This block is used for collecting the divided input bits and adding noise to the input bit streams.
- Direct Lookup Table 1 – This is the lookup table which contains the 24 bits long input bit stream. This is used as the input to the decoder for decoding.
- Direct Lookup Table 2 – This is the lookup table which contains the address of the deinterleaver. This address is used by the interleaver for demuxing of the input bits.

- Turbo Decoder – This is the block which takes the input bit stream and the estimated deinterleaver address and produces the decoded signal.
- Display – This block is used for displaying the output produced by the turbo encoder.

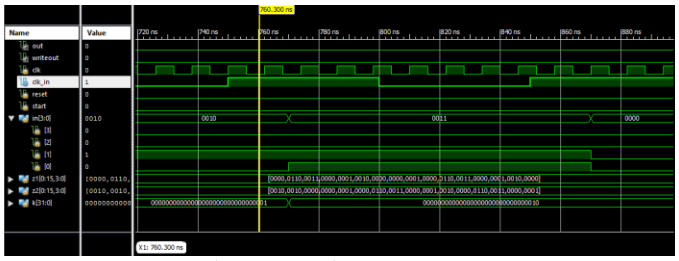
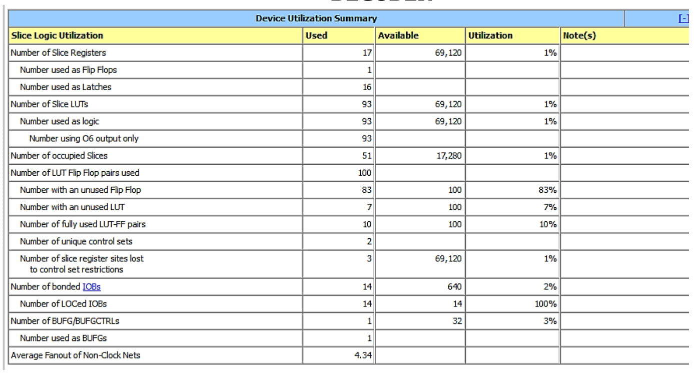


Fig.11 Waveform output of component decoder design

TABLE II: DEVICE UTILIZATION SUMMARY OF TURBO DECODER



Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	17	69,120	1%	
Number used as Flip Flops	1			
Number used as Latches	16			
Number of Slice LUTs	93	69,120	1%	
Number used as logic	93	69,120	1%	
Number using O6 output only	93			
Number of occupied Slices	51	17,280	1%	
Number of LUT Flip Flop pairs used	100			
Number with an unused Flip Flop	83	100	83%	
Number with an unused LUT	7	100	7%	
Number of fully used LUT-FF pairs	10	100	10%	
Number of unique control sets	2			
Number of slice register sites lost to control set restrictions	3	69,120	1%	
Number of bonded IOBs	14	640	2%	
Number of LOCed IOBs	14	14	100%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Average Fanout of Non-Clock Nets	4.34			

C. *BER PERFORMANCE ANALYSIS RESULT*

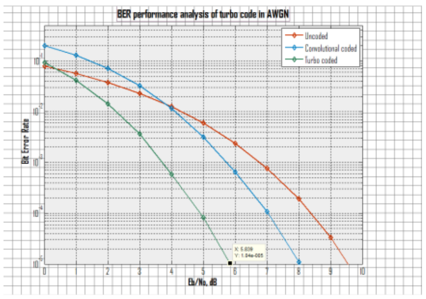


Fig.12 BER curve for the turbo codec

A comprehensive comparison was performed between the uncoded BPSK schemes, convolutionally coded BPSK scheme, Turbo coded BPSK scheme as shown in fig.12-+. From the simulations it was clearly observed that the turbo codes outperformed the convolutional codes. At BER of 10−4, a reading of Eb/No= 4.894 dB was obtained for turbo codes. A coding gain of 3.456 dB was achieved for the turbo codes architecture.

V CONCLUSION

A turbo encoder and decoder, codec pair are designed on MATLAB-Simulink. The deterministic interleaving technique standardised by CCSDS was employed for this design. The turbo encoder was designed at code rate 1/3 and constraint length used for the component encoders was K=3.

The turbo encoder and decoder was simulated using Xilinx ISE Simulator tool. The turbo encoder and decoder designs were henceforth synthesised and implemented successfully on FPGA Virtex 5. The encoded bits and decoded bits were successfully obtained and observed on LEDs of the FPGA Virtex 5 board. The design was optimized to reduce the area by simplifying the complex operations and minimizing the logic blocks. Efficient area utilization resulted in lesser processing delay. Area optimization is one of the major design issues while designing satellites as smaller components badge for better system adaptation and henceforth endure cost effectiveness. The turbo codec performance was evaluated on Matlab 2015, At BER of 10−4, a reading of Eb/No= 4.894 dB was obtained for turbo codes. A coding gain of 3.456 dB was achieved for the turbo codes.

In satellite communications, the other critical design issue is the amount of power consumption, turbo decoding involves huge amount of complex processing which have a major impact on power consumption. Therefore in the future, work can be done to minimize the power by performing a low power design of the turbo codec along with optimizing the area requirements so as to create a more robust design of the turbo codec, which can be efficiently used for satellite communications.

ACKNOWLEDGMENT

We would like to sincerely thank Dr. K N Balasubramanya Murthy, Vice-Chancellor, PES University and Dr. T S Chandar, Chairperson of ECE Dept., PES University for their encouragement, support and for providing the requisite infrastructure for carrying out this work successfully.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: turbo-codes,” Proc. ICC’93, Geneva, Switzerland, May 1993 pp. 1064-1070
[2] Habib, I. Paker, O. Sawitzki, S., “Design Space Exploration of Hard Decision Viterbi Decoding: Algorithm and VLSI Implementation”, IEEE Transaction on very Large Scale Integration (VLSI) Systems, May 2010, pp. 794-807
[3] Consultative Committee for Space Data Systems, "Recommendation for Space Data Systems Standards: Telemetry Channel Coding", CCSDS 101.0-B-5. http://www.ccsds.org.
[4] Shweta Ramteke, Yeshwantrao Chavan Coll. of Eng., Wanadongri,India, SandeepKakde, Yogesh Suryawanshi ; Mangesh Meshram “Performance analysis of Turbo decoder using Soft Output Viterbi Algorithm”, IEEE Communications and Signal Processing (ICCSP), 2015 International Conference,April201, pp.1332 - 1336
[5] Turbo Coding: Basic Principles; Schlegel, C.; Perez, L. Trellis and Turbo Coding: Iterative and Graph-Based Error Control Coding Year: 2015 Pages: 528, DOI: 10.1002/9781119106319.ch8 Referenced in: Wiley-IEEE Press eBook Chapters
[6] Closing In On the Perfect Code", IEEE Spectrum, March 2010
[7] J. Hagenauer and P. Hoeher “A Viterbi Algorithm with Soft-Decision Outputs and its Applications”, Proc. Globecom pp. 1680-1686 November 1989.
[8] S. Crozier, J. Lodge, P. Gunand and A. Hunt “Performance of Turbo-Codes with Relative Prime and Golden Interleaving Strategies” Sixth International Mobile Satellite Conference (IMSC’99), Ottowa, Canada,pp.268-275,June,1999.