RESEARCH ARTICLE OPEN ACCESS

FPGA Based Area Efficient Turbo Decoder For Wireless Communication

Mansi Rastogi*, Rajesh Mehra**
*(ME Student (ECE), NITTTR Chandigarh, UT, India
** (Associate Professor, ECE Department,NITTTR Chandigarh, UT, India)

ABSTRACT
To fulfil the extensive need of high data rate transfer in today's wireless communication systems such as WiMAX and 4G LTE (Long Term Evolution), the turbo codes gives an exceptional performance. They have allowed for near Shannon limit information transfer in modern communication systems. As the performance of these codes increases, their decoding complexity is also increases and so the power consumption. To reduce this complexity without decreasing its BER (Bit Error Rate) performance a novel modification over SOVA (Soft output Viterbi Algorithm) is proposed in this paper. The proposed model is also implemented on FPGA Xilinx Virtex 5 XC5VLX85ff676-2. The simulation results over MATLAB has been shown, indicates a comparable BER as compared to LOG-MAP with reduced complexity. The synthesis results over Xilinx FPGA shows an improvement of 12% over area utilization as compared to MAX-LOG-MAP implementation. So with reduced area and low BER, a cost effective solution proposed in this paper.

Keywords-Turbo code, FPGA, SOVA algorithm, Log MAP algorithm, FEC (Forward Error Correction), Convolutional code (CC), BER etc.

I. INTRODUCTION

As the technology advances in wireless communication, there are several application fields that have been strongly reinforced. Among these channel decoding is one of the most significant and interesting ones. To increase the efficiency and reliability of the transmission Forward Error Correction (FEC) methods are used. FEC used the ECC (Error Correction Codes) to detect and correct the errors occur due to the channel impairments. Turbo codes as ECC [1] was first proposed by Berrou in 1993, which close to Shannon limit of the error correction capability. Turbo codes are widely used in various communication systems due to its capability with high data transmission rate and large system throughput in LTE [2].

Turbo decoders are decoded iteratively. There are mainly two types of iteratively decoding algorithms used in turbo codes. The one is SOVA (Soft Output Viterbi Algorithm), used in this paper. It is advancement over Viterbi Algorithm with Soft-Decision Outputs [3] and the other is BCJR (Bahl, Cocke, Jelinek and Raviv) algorithm. In this algorithm a posteriori probability (APP) is maximized so also known as Maximum a posteriori (MAP) algorithm. Both the branches become workable after the evolution of the logarithmic versions. Further evolutions of MAP are Log-MAP [4] algorithm and the Max-Log-MAP [5] algorithm respectively. These algorithms reduce the complexity by replacing arithmetic operations with logarithm and max operator with Log-MAP greatly reducing the cost of implementing MAP. In this paper a modified SOVA algorithm is produced to reduce the computation complexity introduced by iterative decoders.

On the other hand, the need of VLSI technology increases due to the increased demand of miniaturizing electronic devices. These technologies have reached a development point where several technologies will come in a small microchip. In this era the need of reconfigurable devices increases rapidly. Moreover, embedded processors, digital signal processors, programmable devices, as FPGA's application specific instruction-set processors and VLSI technologies have come to the point where their power and cost should be minimized [6], [7]. In this paper a FPGA implementation of modified SOVA decoding algorithm is proposed to reduce the utilized no. of resources in terms of slices, Flip-Flop etc.
The organization of this paper is as follows. Section II contains the detailed architecture of turbo encoder and decoder. The algorithm structure of modified SOVA is discussed in III. MATLAB based simulation results and FPGA based results have been discussed in section IV. In the last conclusion and future scope is discussed in section V.

II. TURBO CODES

Turbo Codes are the forward error correction codes in which redundant information bits is added in the form of parity to the information bits. These codes are very attractive due to their error corrective capability. In this the receiver can correct most of the errors, which introduced by the channel. The turbo code system is given in figure 1.

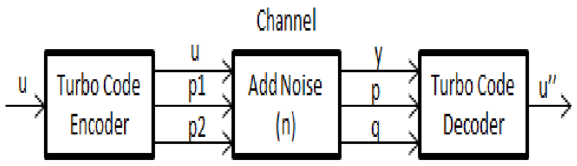

Fig.1 Turbo Code System

The information sequence u is given to the turbo encoder which generates the sequence u, p1 and p2. The p1 and p2 are parity bits in which p1 are generated by u and p2 is generated by the interleaved version of u. The channel is taken in this paper is AWGN. In which n ~ N (0, σ^2) is the random noise introduced by the channel to the information sequence. Where N (0, σ^2) is the zero mean Gaussian noise with variance σ^2=No/2. By adding noise from the AWGN channel, the input to the decoder is y, p and q corresponds to u, p1 and p2.

2.1 Turbo Code Encoder

Turbo code encoder consists of two RSC (recursive systematic convolutional) encoders and an interleaver. It produces a recursive Parallel concatenated Convolutional Code (PCCC) known as turbo code. Here the first constituent encoder receives input bits directly, whereas the second constituent encoder is fed with input bits through the interleaver as shown in figure 2. In this paper 16-state RSC encoders are used. Eachconstituent encoder is built around a shift register of length =4 resulting in 2^4 =16 states. As shown in the figure the code rate of the turbo encoder is 1/3, thus for each input, three outputs are generated and the state values is also depends on the input bits. The input block length is indicated as B. As shown, the encoder generates the three outputs (systematic + tail) bits u, parity bits p1 and p2 of block length B from constituent encoder 1 and 2.In an interleaver the input and output bits are same as input but in a different temporal order. In this the input bits are permuted in a matrix, such that the interleaver output is a pseudo-random string of the input bits. Basically the Interleaving converts by linkingeasily error-prone code and burst errors together with error free code. If a burst error is crooked a block of letters in the middle of the information message. It is impossible to recover that information signal without interleaver. By using interleaving, it spreads out the signal before sending it to the noisy channel and the text ismuch easier to derive afterdeinterleaving. The permutation matrix is based on the number of bitsper block, which is equal to B in this paper.

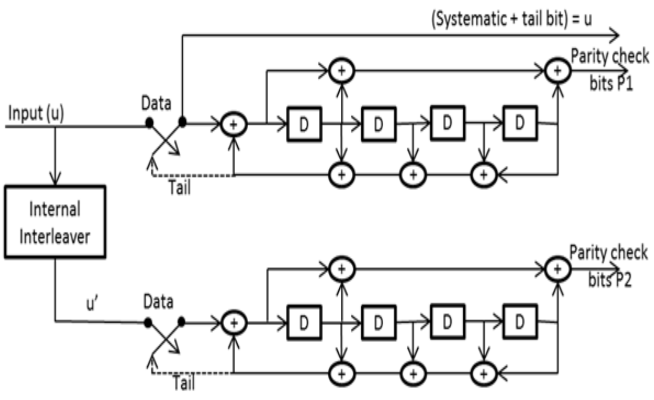

Fig. 2 Turbo Code Encoder

2.2 Turbo Code Decoder

The task of the decoder is to restore the information sequence sent to the transmitter from the systematic and parity bits generated by the encoder, as these bits are irrupted by noise.In the turbo decoder two iteratively SISO (Soft input Soft Output) decoders are used. As systematic and parity bits u, p1 and p2 are corrupted by noise and the noisy systematic and parity bits reached to the decoder are y, p and q. These bits are separated using demultiplexer. Its function is to map the parity check bits and systematic bits into even and odd columns respectively. As shown in figure 3 SISO decoder 1 uses the systematic bits y, parity checkbits 1 (p) and a priori information from SISO Decoder 2 (zero at the first iteration) to estimate a bit sequence. This processing results in two outputs as extrinsic information L^1_{ex} and alog-likelihood ratio (LLR) L^1_{lr}. The extrinsic information L^1_{ex} is interleaved by using the same interleaver as encoder to the SISOdecoder 2, where it is used as a priori information. Then decoder 2 do the same processing as decoder 1 to generate the extrinsic information $L^2_{ex}$and a log-likelihood ratio (LLR) $L^2_{lr}$by using the parity check bits 2 (p),interleaved versions of received noisy systematic bits and priori information from SISO Decoder 1(L^1_{ex}). The output of SISO decoder 2 L^2_{ex} is deinterleaved and isused as a priori information in SISO decoder 1 for a second iteration on the same systematic andparity check bits as for the first iteration. After a number of iterations the two LLR outputs from decoder 1 (L^1_{lr}) and decoder 2 (L^2_{lr})are used to make a hard decision on the bit sequence. The number of iterations needed in turbo decoder to provide a good evaluation of information sequence depends on the encoder properties.

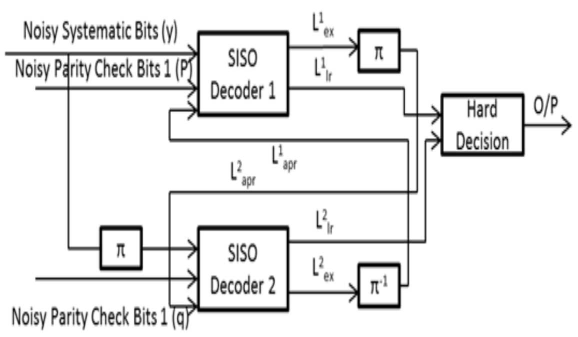

Fig. 3 Block Diagram of Convolutional Turbo Decoder

III. Modified SOVA

There are two types of SOVA algorithm: Traceback Method and Register Exchange Method. In this paper the traceback method is used. The SOVA uses the same idea as VA (Viterbi Algorithm) except that it generates the soft output, represents reliability of the bit decision. The input values to the decoder are y and p. For accurate results y and p are not hard decoded. Rather they have a continuous value depends on the channel reliability. When used in turbo codes the decoders also used a priori information termed as L_a (alpha), Which after combining with systematic and parity bits is converted to complete information L-all. The extrinsic information L_e can be generated by subtracting priori and systematic bits from L_all, considered as the input for second decoder. The result is multiplied by the channel reliability to compensate for the distortion.

As far as the SOVA decoding algorithm is concerned. It is analogues to VA by using some modifications. The basic architecture block of traceback SOVA decoder is shown in figure 4. This architecture is mainly divided in three blocks BMU (branch metric unit), ACS (add-compare and select), and traceback unit.

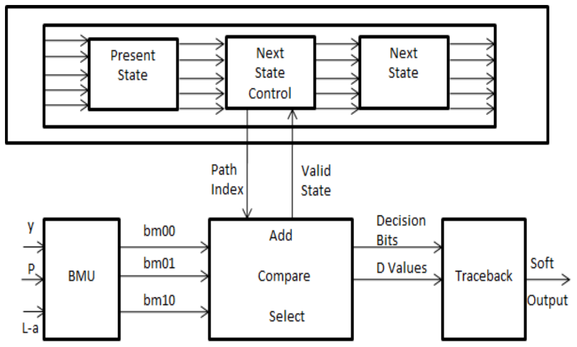

Fig. 4 Basic Architecture Block of Trace back SOVA

3.1 BMU

The branch metric unit in VA calculates the hamming distance between the received parity and systematic bits and the branch word associated with this edge. This branch word is generated by the encoder. The BM (Branch metric) output for VA is given as:

$$BM = \text{Hamming Distance}(\{y_k, p_k\},\{u_k', p_k'\}) \quad (1)$$

While in SOVA, BM output is given as:

$$BM= (1/2)\, u_k'\, L\_a\, (u_k't) + L\_c/2\, (y_k u_k' + p_k\, p_k') \quad (2)$$

Where:
(y_k, p_k) = Received from the channel at stage k.
(u_k', p_k') = Associated with this edge
L_a $(u_k't)$ = A priori information
L_c= Channel reliability measure
The value of u_k' can be +1 for binary 1 or -1 for binary 0.

3.2 ACS

This unit will take the previous path metric from memory and add the branch metric to it calculated by BMU, results in new path metric. The equation for this is given as:

$$M(S_k) = M(S_{k-1}) + (1/2)\, u_k'\, L\_a\, (u_k't) + L\_c/2\, (y_k u_k' + p_k\, p_k') \quad (3)$$

For the values of uk' as +1 or -1 for 1 or 0, two metric values are calculated: one for the binary "0" (Mk0) and one for the binary "1" (Mk1).

$$M(S_{k0}) = M(S_{k-1}) - (1/2)\, L\_a\, (u_k't) + L\_c/2\, (y_k u_k' + p_k\, p_k') \quad (4)$$

$$M(S_{k1}) = M(S_{k-1}) + (1/2)\, L\_a\, (u_k't) + L\_c/2\, (y_k u_k' + p_k\, p_k') \quad (5)$$

These two values are compared, and the highest value, difference between the two (Mdiff=Δ) and its corresponding binary value is stored. This is done for an entire frame block, which corresponds to going forward in the trellis diagram while calculating the metric of each path. In SOVA a path metric represents the likelihood that the path is the decoded path and a larger metric represents increased likelihood. So when many paths converge then highest metric path is considered as the decoded path and its metric and survivor is stored. The other paths are discarded. The trellis diagram for SOVA algorithm is shown in figure 5. Three paths are shown in the figure path 0, 1 and 2. The difference between the two adjacent paths coming to a node is given by Δ.

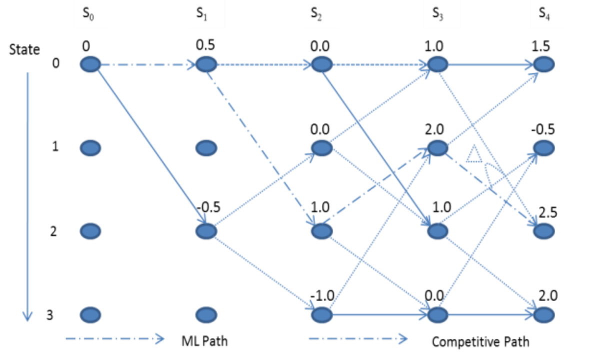

Fig. 5 Trellis Diagram for SOVA Component Decoder

3.2.1 Modified SOVA

The complexity of MSOVA is reduced as compared to SOVA by pruning some metric states. The performance of modified SOVA is close to that of LOG-MAP, when compensated for the loss using the expectation and scaling factor. In MSOVA two new parameters Nmax and T are used to prune the metric paths. T represents the threshold value and Nmax represents the maximum no. of survivor path allowed in the decoding trellis. At each trellis stage, only those paths whose metric values satisfy equation 6 will be survive.

$$M(S_k) = \max \{M(S_{k1})\} + T \qquad (6)$$

When the number of preserved path is more than Nmax then only the best Nmax path will be kept. The trellis diagram of modified SOVA for T=-2 and Nmax=3 is given in figure 6, where the numbers on top of each node represents paths metrics. The node obstructed by an open circle represents a path pruned using threshold *T*. The nodes encircled by cross indicate paths pruned using Nmax. As it is unlikely to become a low metric path to a ML path later, pruning the bad paths has only small possibility to change the ML path. As a result Modified SOVA generates the decoded sequence with less complexity by using less number of metrics.

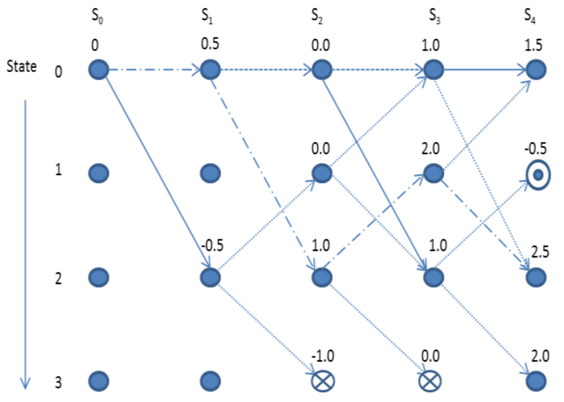

Fig. 6 Trellis for MSOVA Decoderfor T = -2 and Nmax = 3

3.3 Traceback

Traceback is done to find the most likely sequence of bits. But first the most likely state for each decoder has to be found. This can be done by standing at the end of the trellis diagram and looking at each of the final states. It is trying to find the one with the highest metric of likelihood. This is easily done for decoder 1, as its trellis is terminated to an all zero state in the encoder, causing the final state in the decoder to be the all zero state as well. While for decoder 2, the last state is not the all zero state due to the interleaver. It will change the sequence of the information data. In this the decoder compares the metric for all the states at the end of the trellis, results in the highest metric as the most likely. Now as starting point is known to the decoder, it is possible to start the traceback through the trellis to find the most likely sequence. A sequence is estimated based on the stored survivor values and their corresponding metric values (M_{Sk0}), and (M_{Sk1}). This estimation is then compared with the competitive path within the range of δ. Where δ is the window size of SOVA. After this a log likelihood ratio (LLR) is calculated to determine whether the competitive and estimated sequence differ from each other. This is based upon the Δ. The estimated sequence is then converted to -1 for binary 0 and 1 for binary 1 and multiplied with the LLR. It results in a negative LLR value when the estimated bit value is zero and a positive LLR when the estimated bit is one. This is the soft-output result of the turbo decoder.

IV. MATLABSimulation Based Results

The simulation result of proposed decoder for different block length in terms of FER (Frame Error Rate) Vs. SNR (Signal to Noise Ratio) in dB has been shown in figure 7. It is clear from the graph that as the number of input bits i.e. blocks length increases, the performance increases simultaneously. In this two block length are considered 500 and 1500. The generator polynomial used is g = [7, 5]. The numbers of iteration used are equal to 6. In figure 7 a graph is drawn for different number of iterations 1 to 6. The parameters used for the graph are FER vs. SNR. It is drawn for the 1024 bits. The generator polynomial is used as g = [7, 5]. It is shown in the graph that as the number of iterations increases, the FER performance increases. In turbo decoder the performance is increases if the decoder operates multiple times. But to process the decoder multiple times the delay and complexity increases vice-versa. The MATLAB used for the above simulation is r2013a.

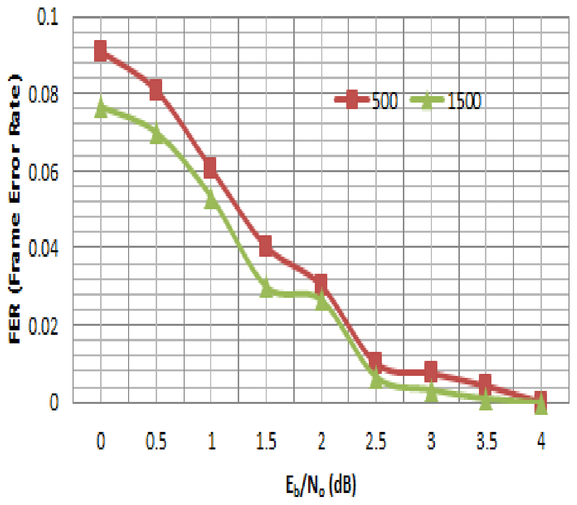

Fig. 7 FER vs. SNR for different block length

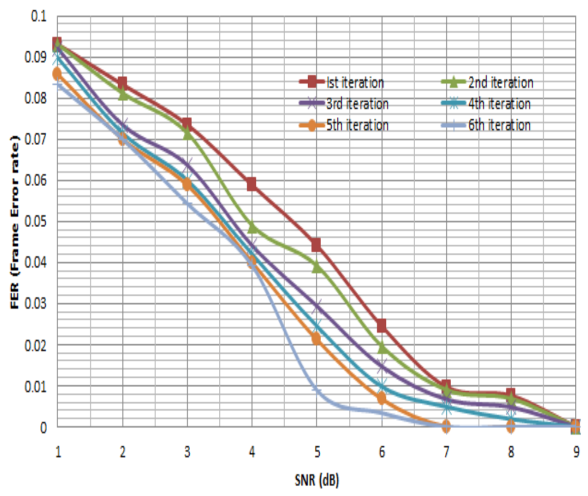

Fig. 8 FER vs. SNR for different iterations

In figure 8 a comparison graph is given between the proposed algorithm (Modified SOVA) and LOG-MAP [8] in terms of BER vs. SNR. Both the algorithms are used for turbo codes. The results are calculated for code rate R=1/3, Block Length = 1500 bits and the number of iterations are 4. The generator polynomial used for the proposed design is g = [31, 27]. As shown in figure BER for MSOVA is slightly improved over LOG-MAP. The BER performance is slightly improved for low range of E_b/N_o values, while for higher range of Eb/N0 values BER performance is equivalent to LOG-MAP. The proposed SOVA have much reduced complexity than LOG-MAP [9]. So MSOVA is a preferred choice in terms of performance and complexity for wireless communication.

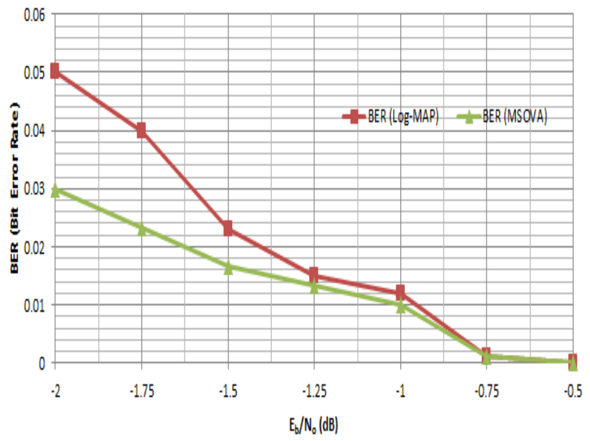

Fig. 9 Comparison of BER vs. Eb/No for proposed and LOG-
MAP algorithm

V. FPGA ImplementationBased Results

The Turbo encoder and decoder using proposed algorithm was implemented using Verilog hardware description language, which offers high abstraction level during the implementation. The Verilog description was synthesized using Xilinx Virtex 5XC5VLX85ff676 FPGA with a speed grade of -2. It has used ISE 13.4 synthesis tool to measure the performance of the implemented Turbo encoder and decoder in terms of utilized area.Table 1 show the parameter used in the hardware implementation of the proposed decoder.

TABLE I
PARAMETER USED FOR TURBO CODEC

No. of Input Sequence	k	1
Length of Input Sequence	h	1024
No. of Output Sequence	n	2
Code Rate	R	½
Constraint Length	υ	5
Polynomial Generator in Octal Notation	G	$(31, 27)_8$
Total No. of States	$2^{\upsilon-1}$	16
Threshold	T	-3
Maximum No. of Path	N_{max}	4

Table 2 shows the hardware utilization of the proposed decoder in terms of area used, frequency and delay. The parameters used for area utilization are number of slice registers, number of slices, LUT, bonded IOBs and number of BUFG etc. In terms of timing utilization are frequency and delay. From the table it was shown that proposed decoder is area efficient.

TABLE 2
HARDWARE AND TIMING PERFORMANCE OF TURBO CODES

PROPOSED TURBO DECODER	
Device Utilization Summary	
No. of Slice Registers	644 out of 51840 (1%)
No. of Slices	230 out of 12960 (1%)
No. of LUTs	919 out of 51840 (2%)
No. of Fully used LUT-FF pairs	341 out of 1222 (27%)
No. of Bonded IOBs	60 out of 440 (13%)
No. of BUFG/BUFGCTRLs	1 out of 32 (3%)
Timing summary	
Frequency	97.448 MHz
Delay	10.262 ns

In figure 10 the comparison between the proposed turbo codec and turbo codec using MAX-LOG-MAP with scaling [10]. The parameters include for the comparison are number of slice registers, LUT and number of slices. The blue lines is for existing and red is for proposed decoder.

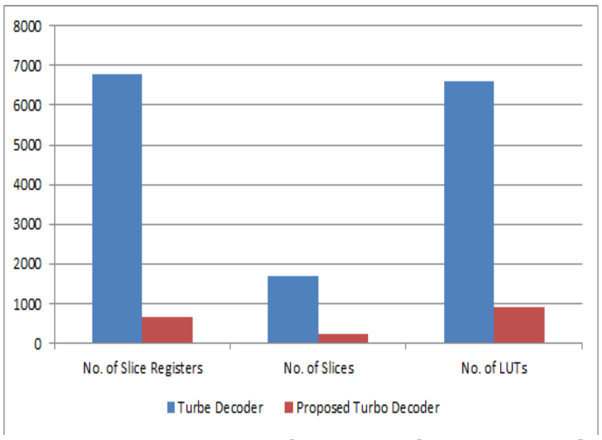

Fig. 10 Comparison Graph Between the Existing and Proposed Decoder

As shown in the table the utilized number of slices, LUT, slice registers are less as compared to existing algorithms. So it results in optimized area performance. Its frequency is less as compared to hardware implementation of MAX-LOG-MAP. The simulation waveform for the proposed decoder has been shown in figure 11.

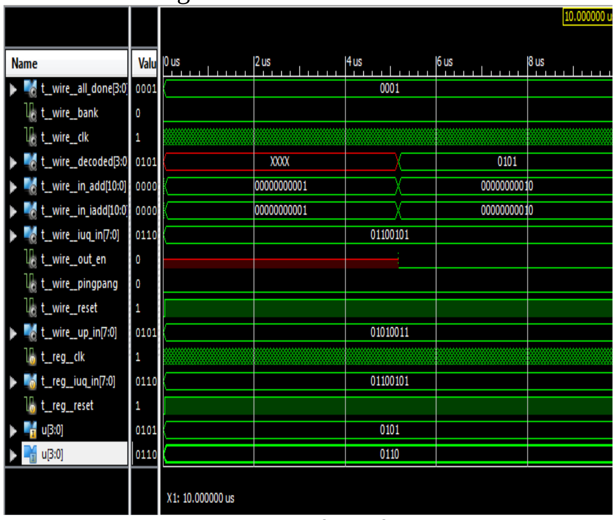
10.000000 us
Name
Value
0 us
2 us
4 us
6 us
8 us
t_wire_all_done[3:0] 0001 0001
t_wire_bank 0
t_wire_clk 1
t_wire_decoded[3:0] 0101 XXXX 0101
t_wire_in_add[10:0] 0000 00000000001 00000000010
t_wire_in_iadd[10:0] 0000 00000000001 00000000010
t_wire_iuq_in[7:0] 0110 01100101
t_wire_out_en 0
t_wire_pingpang 0
t_wire_reset 1
t_wire_up_in[7:0] 0101 01010011
t_reg_clk 1
t_reg_iuq_in[7:0] 0110 01100101
t_reg_reset 1
u[3:0] 0101 0101
u[3:0] 0110 0110
X1: 10.000000 us

Fig. 11 Simulation Waveform for the Proposed Decoder

VI. CONCLUSION

Turbo Code has become one of the best choices to deal with errors induced from high-noise communication channels due to the superior error correction performance. A modification over SOVA algorithm has been proposed in this paper. The modification over SOVA is done by pruning the paths with low path metrics compared to other path metric at that time. With this the complexity is reduced as the total number of path metric is reduced at each time. The proposed algorithm results in slightly better BER performance than LOG-MAP algorithm. The proposed architecture is quite flexible to support multiple code lengths. The FPGA utilization of the proposed decoder is compared with a CTC decoder using MAX-LOG-MAP algorithm. It is shown that proposed CTC decoder is superior to existing CTC decoder in hardware utilization and also having a comparable performance. So an area efficient, better performance and a cost effective solution has been proposed in this paper for wireless communication systems.

REFERENCES

[1] Qu Wei, Information theory and coding theory, Sciences Publishing House, 2005.
[2] J. Haqenanner, P. Hoer, "A Viterbi Algorithm with Soft-Decision Outputs and Its Applications," IEEE Globecom'89, 1989, pp.1680–1686.
[3] Robertson, E. Villegrum, P. Hoeher, "A Comparison of Optimal and Sub-optimal MAP Decoding Algorithm Operating in the Log Domain," Proceedings IEEE international Conference on Communication, 1995, pp. 1009-1013.
[4] C. Berrou, A. Gliavieux, and P. Thitimajshima, "Near Shannon limiterror-correcting coding and decoding: turbo codes (1)," in Proceedings IEEE International Conference Communications, pp. 1064-1070, 1993.
[5] W. Zhongfeng Wang, Zhipei Chi and Keshab K. Parhi, "Area-Efficient High-Speed Decoding Scheme for Turbo Decoders," *IEEE Trans. Very Large Scale Integrated (VLSI) Systems*, vol. 10, pp. 902–912, 2002.
[6] Olaf J. Joeressen, Martin Vaupel, and Heinrich Meyr, "High-Speed VLSI Architectures for Soft-Output Viterbi Decoding," *Proceedings of IEEE International Conference Application Specific Array Processors*, pp. 373-384, Aug. 1992.
[7] Engling Yeo, Stephanie A Augsburger, W. Rhett Davis, Borivoje Nikolic, "A 500-Mb/s Soft-Out Viterbi Decoder," in *IEEE J. Solid State Circuits*, Vol. 38, pp.1234-1241, July 2003.
[8] Wang Huahua and Liu Wenwen, "Analysis of Turbo Decoding Algorithm in LTESystem," IEEE International Conference on Fuzzy Systems andKnowledge Discovery (FSKD), pp. 1741-1744, 2012
[9] J. P. Woodard and L. Hanzo, "Comparative Study of Turbo Decoding Techniques: an overview," *IEEE Transaction on Vehicular Technology*,vol. 49, pp. 2208–2233, 2000.
[10] Alptekin Pamuk, "An FPGA Implementation Architecture forDecoding of Polar Codes," International Symposium on Wireless Communication Systems (ISWCS), pp. 437-441, November 2011.