International Journal of Engineering Research and General Science Volume 3, Issue 1, January-February, 2015
ISSN 2091-2730

An Efficient Turbo Decoder for Wireless Sensor Networks

Saranya P Krishnan, Gnana Sheela K, Christina Thomas

Department of ECE, Toc H Institute of Science and Technology, CUSAT, Kerala, India

Email: saranya.p.krishnan91@gmail.com

Abstract— Error correcting codes are an unavoidable part in digital communication such as data encoding, data decoding and data storage systems. Turbo codes are a class of error correcting codes that have recently been considered for energy constrained wireless communication applications. This paper focuses on the simulation of efficient turbo encoder and decoder for wireless sensor networks. Simulation of turbo encoder and decoder is done using Model Sim. The turbo encoder is simulated using flip flop method. Turbo decoder decodes the data using various MAP algorithms. The algorithms such as LUT Log MAP algorithm and Constant Log MAP algorithm are compared for simulating turbo decoder. The simulation results shows that turbo decoder simulated using constant Log MAP algorithm requires lesser area and hence lesser power consumption. The simulation of turbo encoder using flip flop method and decoder using constant Log MAP algorithm makes this decoder more efficient.

Keywords— Decoder, Energy efficient, Error correcting code, Log MAP algorithm, Soft information, Turbo codes, Wireless sensor networks.

INTRODUCTION

Over the years, there has been an increase in the use of digital communication in the field of computer communication, cellular and wireless sensor networks (WSN). In digital communication, error correction codes are an essential component. To ensure robust operation of digital applications like data encoding, decoding and data storage systems require error correction. Wireless communications require highly reliable data transfer in the presence of data corrupting noise. All these require good error correcting codes. There are different types of forward error correcting codes like block codes, convolutional codes, linear codes and turbo codes. Forward error correction (FEC) or channel coding is a technique which is used for controlling errors in data transmission over unreliable or noisy communication channels. The main idea is the sender encodes his message in a redundant way by using an error-correcting code (ECC).

In the last years, wireless communication systems coped with the problem of delivering reliable information while granting high throughput and better efficiency. In telecommunication, error detection and correction are the techniques that enable reliable delivery of digital data over unreliable communication channel. Among the different error correcting codes like block codes, convolutional codes and linear codes, turbo codes are found to be less complex and more efficient. The superior performance of turbo codes is due to a combination of parallel concatenated coding, iterative decoding, large interleaver size, etc. The large frame size of turbo codes and the iterative decoding process results in large decoding latency and more area consumption. The decoding latency and area has to be reduced and efficiency need to be increased in order to make Turbo-based systems acceptable for real-time voice communication and other applications that require instant data processing, like optical transmission and hard disk storage.

Turbo codes are a class of error correcting codes that come closer to Shannon's limit than any other type of error correcting codes. It has been recognized as a milestone in the channel coding theory. Due to their outstanding error correcting capabilities, turbo codes are highly appreciated in wireless communications as well as in encoding and decoding algorithms. These codes are a recent development in the field of forward-error-correction channel coding. The codes make use of some ideas like interleaving to provide better weight distribution, soft decoding to enhance decoder decisions and maximize the gain from decoder interaction**.**

Wireless sensor networks are more energy constrained, since the sensors are operated for extended periods of time and relying on batteries that are small, in expensive and light weight. In environmental monitoring WSNs for example, even though employing low transmission duty cycles and low average throughputs of less than 1 Mbits/sec the sensor energy consumption is dominated by the transmission energy E_b^{tx} [1],[2]. For this reason, turbo codes have recently found application in these scenarios, since their near-capacity coding gain facilitates reliable communication when using reduced transmission energy E_b^{tx}. Even if the transmission energy of turbo codes is less, it is offset by turbo encoders and turbo decoder energy consumption. Therefore, turbo codes designed for energy constrained scenarios have to minimize the overall energy consumption.

In 1974, Bahl, Cocke, Jelinek and Raviv presented the decoding algorithm based on a posteriori probabilities which was later known as the BCJR, Maximum a Posteriori (MAP) or forward-backward algorithm. The MAP algorithm was not used in practical

implementation for the last 20 years. The situation changed with the advent of turbo codes in 1993. The process of turbo code decoding starts with the formation of a posteriori probabilities (APPs) for each data bit, which is then followed by choosing the data-bit value that corresponds to the MAP probability for that data bit. Upon receiving a corrupted code-bit sequence, the process of decision making done with APPs allows the MAP algorithm to determine the most likely information bit to have been transmitted at each bit time.

RELATED WORK

Robertson P et al (1997) presented a comparison between log-MAP, max-log MAP and soft output Viterbi algorithm (SOVA) [3]. The comparison shows that SOVA is 0.7dB inferior to log-MAP and max-log-MAP lying in between SOVA and log-MAP. The comparative analysis is done for these algorithms in terms of number of additions, multiplication and look-up tables required.

Gross W J et al (1998) developed a simplified MAP algorithm suitable for the implementation of turbo decoder [4]. The simplification eliminates the need for a ROM or multiplexor-tree lookup table and replaces it with a constant value. The results show that the performance of turbo decoders is not adversely affected by this simplification.

Worm A et al (2000) presented a VLSI high speed MAP architecture with optimized memory size and power consumption for decoding the turbo codes [5]. The log-MAP and max log-MAP algorithm is used in the process. Memory size is reduced by minimizing the FIFO memory size. For maximum throughput a fully pipelined architecture is considered. The area decreases by up to 11% and power consumption by up to 15% in case of a Log-MAP decoder and for a Max Log-MAP decoder, even an 18% area decrease and a 20% power decrease.

Wang Z (2002) introduced variety of area efficient parallel turbo decoding schemes [6].Turbo decoders inherently have large decoding latency and low throughput because of iterative decoding. To reduce the latency and increase the throughput, high-speed decoding schemes are employed. Thus the techniques like segmented sliding window approach and two other area-efficient parallel turbo decoding schemes are used .The comparison on storage requirements; number of computation units and overall decoding schemes are made. Also in order to reduce the storage bottleneck partial storage of state metrics approach is also presented.

In 2003, Elassal M et al proposed a method to decrease the power consumption of turbo decoder [7]. In turbo decoder, decoding is done by iteratively exchanging the extrinsic information. In this proposed method the iteration is terminated when the extrinsic information exceeds a particular threshold and then a predefined value is terminated. This reduced memory access for inter leaver and state metrics and thus power was reduced. 25% reduction of power consumption with energy per bit to noise power spectral density ratio E_b/N_o= 1.5 dB is achieved compared to conventional architecture.

Elmasry M et al (2004) designed rate 1/3, 8-state log-MAP turbo decoder architecture [8]. The simplified log-MAP algorithm is used for the component soft-in soft-out decoder (SISO). Several logic and architectural level techniques are applied through the design process to reduce power consumption, area and increase throughput of the turbo decoder Parallelism, quantization, resource sharing, logic reduction and normalization are applied to reduce area, power and throughput. 0.18µ CMOS technology is used. The developed turbo decoder has a core area of 0.6mm^2, clock frequency of 100MHz, power consumption of 63mW and energy efficiency of 2.5n J/b/iteration.

Atluri I et al (2005) formulated the implementation of a low power Log-MAP decoder with reduced storage requirement and based on the optimized MAP algorithm that calculates the reverse state metrics in the forward recursive manner [9]. The new low power derivatives of this decoder through a variation in the percentage of memory savings are presented. Three low power architectures of the Log-MAP decoder not employing the sliding window technique have been developed and post layout power savings of approximately 44%, 40% and 36% with respect to the conventional implementation have been observed.

In 2008, Shah S presented a comparison between viterbi decoder and turbo decoder. It shows that iteration decoders perform better than Viterbi decoders [10]. The trade-off between bit error rate (BER) and energy per bit to noise power spectral density ratio (E_b/No) will always exist in the wireless communication. This helps in reducing the transceiver power. The modulation techniques and BER performance of viterbi and turbo decoders are compared. The BER for Viterbi algorithm is 1.36 x 10^{-5}; and turbo decoder the values are 3.09 x 10^{-8}, 8.00 x 10^{-8}, and 4.35 x 10^{-8} for number of iteration = 2, block size = 512; number of iteration = 8, block size = 512; and number if iteration = 8, block size = 2048 respectively. The results show that turbo decoder is more powerful than viterbi decoder with ~10^3 times improvements in BER.

Reddy P et al (2010) proposed a low power technique for turbo decoding implementation [11]. The digital base band implementation, high performance, energy efficiency, flexibility and low power are major requirements for channel decoding. The proposed techniques

help meet the major requirements. The optimization techniques show an interesting gain in normalized energy efficiency between 4% and 54%. This approach can be extended for turbo decoder implementations in terms of area and throughput.

Li L et al (2013) proposed a framework that can be employed at an early design stage to estimate the processing energy consumption of the turbo decoder architecture [12]. This method reduced overall energy consumption that is transmission energy and processing energy. BCJR algorithm is used. By considering both the transmission energy consumption E_b^{tx} and the decoding energy consumption E_b^{pr} have to be considered right from the commencement of the design .The importance of optimizing the turbo codes at an early design stage is discussed.

Li L et al (2013) proposed low-complexity energy-efficient Turbo decoder architecture [13]. Turbo codes have recently been considered for energy-constrained wireless communication applications, since they have low transmission energy consumption. However, for reducing the overall energy consumption, Look-Up-Table-Log-BCJR (LUT-Log-BCJR) architectures having low processing energy consumption are required. The proposed architecture achieves a low area and hence a low energy consumption. In this approach the LUT-Log-BCJR algorithm is decomposed into its most fundamental ACS operations. The architecture was validated by implementing an LTE turbo decoder and 71% energy reduction was achieved.

Martina M et al (2014) proposed a simplified n-input max* approximation algorithm for very low complexity turbo decoder hardware architectures [14]. The results show that the proposed architecture is simpler by 30%, on average, than the constant Log-MAP in terms of chip area with the same delay. However, when applying scaling to the extrinsic information, the proposed algorithm achieves almost same Log-MAP turbo code performance for both binary and double-binary turbo codes, without increasing the implementation complexity.

MATERIALS AND METHODS USED

Reliable data transmission in wireless communication systems requires sophisticated channel coding schemes and corresponding high-throughput, low-area, and energy-efficient decoder implementations. Forward-error-correcting (FEC) channel codes are commonly used to improve the energy efficiency of wireless communication systems. FEC encoder on the transmitter side will add redundancy to the data in the form of parity information. Then at the receiver, an FEC decoder is able to exploit the redundancy in such a way that a reasonable number of channel errors can be corrected. There are different types of error correcting codes like linear codes, convolutional codes and turbo codes. The significance of turbo codes are its features, like less complexity and better efficiency compared to other error correcting codes. In information theory, turbo codes are a class of high-performance forward error correction codes developed during 1993, which were the first practical codes to closely approach the channel capacity, the theoretical maximum for the code rate at which reliable communication is still possible given a specific noise level.

The decoding algorithm employed in the decoders is the maximum a posteriori probability (MAP) algorithm. The MAP algorithm provides a reliability metric, known as the log-likelihood ratio (LLR), on the transmitted code symbols. The LLR output is employed by other constituent decoders, which attempt to improve their LLR estimates iteratively.

PROPOSED METHODOLOGY

In wireless sensor networks, for reliable data transmission, the data need to be encoded at the transmitter and then decoded at the receiver. The input data first enters the turbo encoder. From the turbo encoder the encoded data passes through the noisy channel, then to the turbo decoder. The turbo decoder produces the decoded output.

Block Diagram of Turbo Encoder

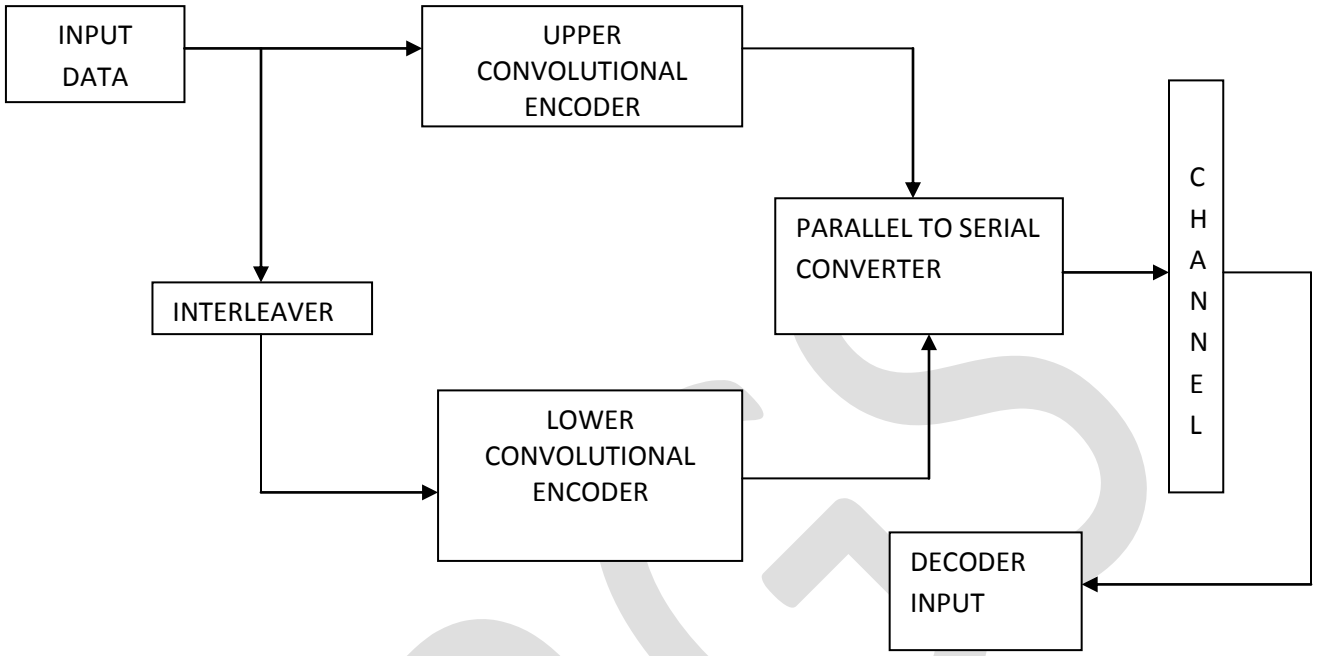

Fig 1: Block Diagram of Turbo Encoder

Fig 1. shows block diagram of turbo encoder. It mainly consists of 4 blocks such as upper convolutional encoder, lower convolutional encoder, interleaver and parallel to serial converter. The input data is binary values. The input data enters the upper convolutional encoder. At the same time information bits enter the interleaver module. The interleaver output is then fed to lower convolutional encoder. The output of upper convolutional and lower convolutional encoder's are passed through a parallel to serial converter and then through the channel.

Each input bit entering the turbo encoder is encoded as 4 bits. The convolutional encoder consists of number of memory elements. It can be considered to be made of flip-flops. The input data that enters the upper convolutional encoder produces two encoded bits and they are system bit 1 and parity bit 1. The input data at the same time enters the interleaver. Interleaver module is the one that rearrange the order of input bits. The interleaved bits are then made to enter the lower convolutional encoder. In lower convolutional encoder, corresponding to each bit entering into it, the encoder produces two encoded bits and they are system bit 2 and parity bit 2. Both the encoders produce completely different sequence. Thus for each input bit, there will be 4 encoded bits. These sequence of encoded data is passed through a parallel to serial converter and then through the channel.

Block Diagram of Turbo Decoder

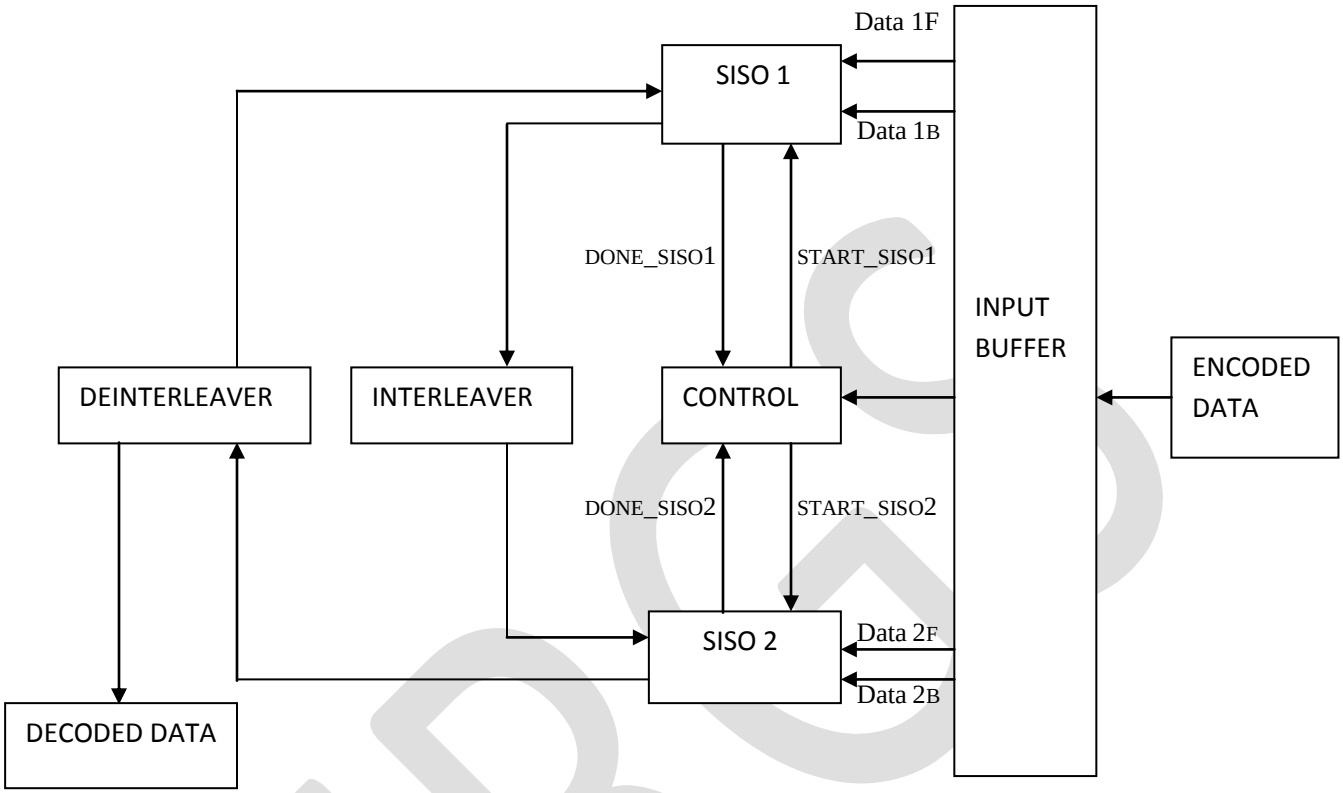

Fig 2: Block Diagram of Turbo Decoder

Fig 2. shows the block diagram of Turbo decoder. It consists of 6 modules such as input buffer, SISO 1, SISO 2, Control, interleaver and deinterleaver

The turbo decoder always works with Log likelihood ratios (LLR). This means ratio of probability that the received bit is 0 or 1. Input buffer is the module that stores the received data's. SISO 1 and SISO 2 are soft input soft output modules. The algorithm used is performed in these modules. The soft input soft output module (SISO) works with soft information. Soft information means the probability that the bit is 0 or 1. The interleaver module is used to interleave the data. Interleaver works by storing the data using interleaved address and reading it using normal address. The deinterleaver module stores the data using normal address and read it back using interleaved address. The controller module controls all other modules.

The encoded data that passes through the noisy channel enters the input buffer. Input buffer stores the received information. The 2 blocks of data enters SISO 1 and SISO 2. Data 1F and data 1B enter SISO 1. Each of the data blocks is 8 bit wide. In data 1F the 4 bits MSB is filled by the system bit 1 and 4 LSB bits are filled by parity bit 1.The data 1B is just reverse order of data 1F. Data 2F and data 2B enter SISO 2. The 4 MSB bits in data 2F is system bit 2 and 4 LSB bits are parity bit 2. When the control module sends the signal START_SISO1, SISO 1 starts the iteration. Inside SISO 1 the LLR value corresponding to each bit is calculated using the algorithm. Once it completes the first iteration, it sends back DONE_SISO1 signal to control module. This makes control module to send the START_SISO2 signal and SISO 2 starts the iteration. The input of SISO 2 is data 2F and data 2B from input buffer and the output of SISO 1 after passing through the interleaver. Once SISO 2 completes the iteration, it sends DONE_SISO2 signal to the control module. Similarly the input to SISO 1 is data 1F and data 1B from input buffer and the output of SISO 2 after passing it through deinterleaver. The SISO 1 and SISO 2 modules perform the iteration using the algorithm. During the iteration, parameters like α, β, γ are calculated. The final output is taken from the deinterleaver after a particular number of iteration. The final output will also be LLR values. If this LLR value obtained is negative then it will be decoded as 1 and if it is positive it is decoded as 0.

Algorithm

The algorithm that is used in turbo decoder is maximum a posteriori (MAP) algorithm. There are different versions of MAP algorithm. Based on the algorithm used, the area required and power consumed by the turbo decoder varies.

In turbo decoder, the SISO modules undergo a number of iteration using the algorithm. During the iteration certain parameters like α, β, γ are calculated. γ is branch metrics. α and β are node metrics. Using γ, the node metrics α and β are calculated.

For example:

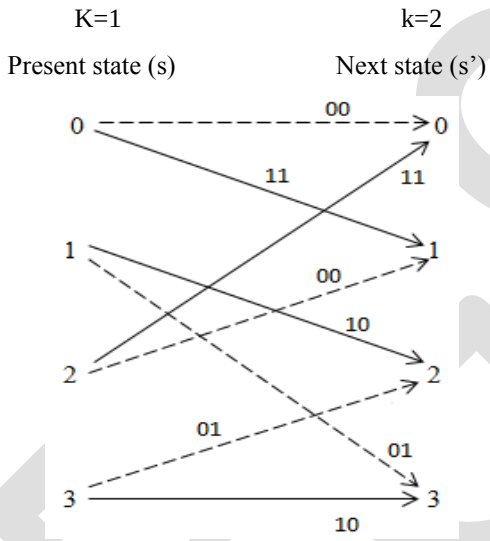

Fig 3: State Diagram

This is a state diagram with present state s and next state s'. The dashed arrow shows the transition from one state to another when 0 input is given. The bold arrow shows the transition for 1 input. Using branch metrics, the node metrics are calculated as follows:

$$\gamma_k^{i,m} \gamma = 0.5 \exp (x_k u_k + y_k v_k) \quad (1)$$

where 'm' is the state and 'i' is the input 0 or 1. u_k and v_k are the outputs when moving from one state to another. x_k and y_k are the input values that enter into the SISO modules. γ is branch metrics.

$$\alpha_{k+1}(s') = \max^*_{s \text{ to } s'} (\alpha_k(s) + \sum_{i=1 \text{ to } 2} \gamma(s,s')) \quad (2)$$

$$\beta_{k-1}(s) = \max^*_{s \text{ to } s'} (\alpha_k(s') + \sum_{i=1 \text{ to } 2} \gamma(s,s')) \quad (3)$$

where α is calculated in the forward direction of state diagram. β is calculated in backward direction. In (2) and (3), to obtain the values of α and β max* operation is required. This operation is performed using MAP algorithm. Using the node metrics and branch metrics, the iteration is performed.

LUT Log MAP Algorithm

This is a MAP algorithm. In turbo decoder, when SISO modules perform iteration, parameters like α, β, γ need to be calculated. α, β are called node metrics and γ is branch metrics. For determining the values of node metrics (α, β), max* operation is required. The max* operation associated with look-up table Log MAP (LUT Log MAP) algorithm is as follows.

$$\text{Max*}(x,y) = \max(x,y) + \begin{cases} 0.75, \text{ if } |y-x| = 0 \\ 0.5, \text{ if } |y-x| = (0.25, 0.5, 0.75) \\ 0.25, \text{ if } |y-x| = (1, 1.25, 1.5, 1.75, 2) \end{cases} \quad (4)$$

In this max* operation, maximum of x and y is determined and then based on the range where |y-x| lies, correction factors like 0.75, 0.5 or 0.25 is added.

Constant Log MAP Algorithm

This is another version of MAP algorithm. To determine node metrics (α, β) during SISO iteration, max* operation is required. The max* operation associated with constant Log MAP algorithm is as follows.

$$\text{Max*}(x,y) = \max(x,y) + \begin{cases} 0, \text{ if } |y-x| > T \\ C, \text{ if } |y-x| <= T \end{cases} \quad (5)$$

where T is the threshold value = 1.5,C=0.5

In this max* operation after obtaining maximum of x and y, based on the range of |y-x| correction factor 0.5 is added.

SIMULATION RESULTS

The simulation work has been done in ModelSim. ModelSim is a powerful simulator that can be used to simulate the behaviour and performance of logic circuits. The simulator allows the user to apply inputs to the designed circuit, usually referred to as test vectors, and to observe the outputs generated in response. The user can use the waveform editor to represent the input signals as waveforms.

Simulation Results of Turbo Encoder

Turbo encoder using 2 flip-flops

Input given: 111110
Output obtained: ex1 system bit 1 -111110

ey1 Parity bit 1-110001

ex2 System bit 2- 110111

ey2 Parity bit 2 - 111010

System bit 1 and parity bit 1 are the encoded bits from upper convolutional encoder and system bit 2 and parity bit 2 are the encoded bits from lower convolutional encoder. So for 6 bits of input data, total 24 encoded bits are produced.

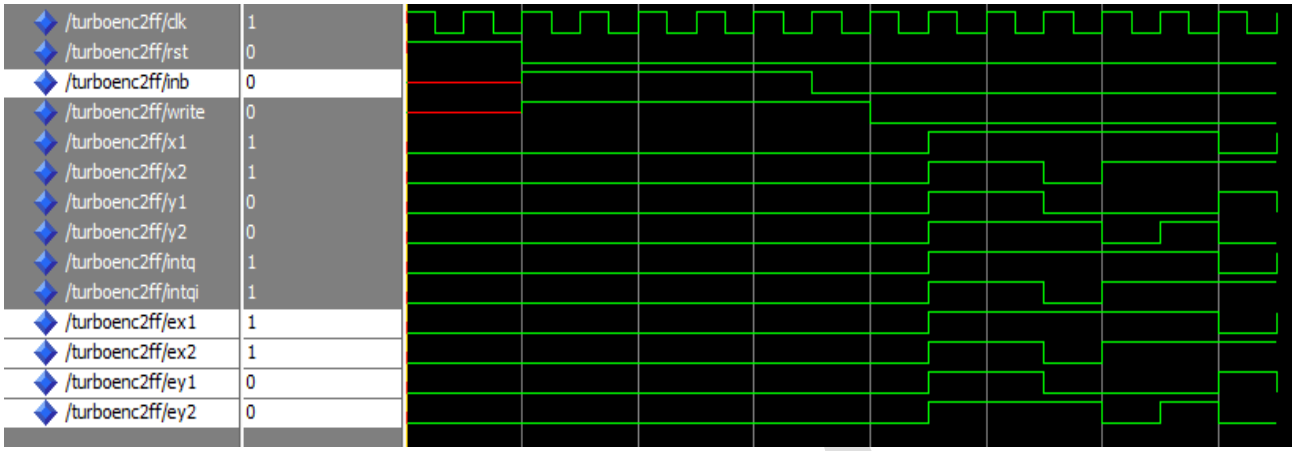

Fig 4: Output waveform of turbo encoder using 2 flip-flops

LUT Max* operation in LUT Log MAP Algorithm

Input given: a=000100010
b=000011100
Output obtained: z=000100011

The output z is obtained from Max*(a,b) calculated based on equation(4).

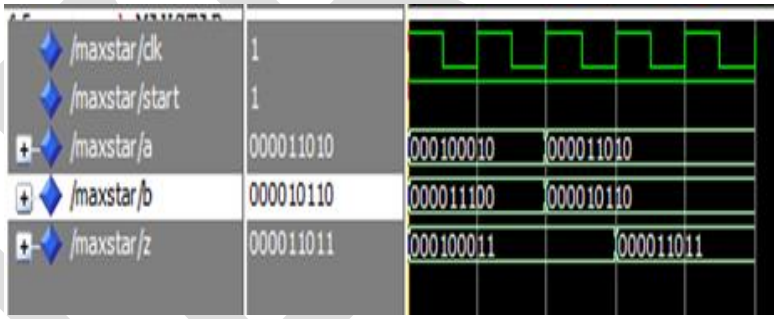

Fig 5: Output waveform of LUT Max* operation

Constant Max* operation in constant Log MAP Algorithm

Input given: a=000011010
b=000010110
Output obtained: z=000011100

The output z is obtained from Max*(a,b) calculated based on equation(5).

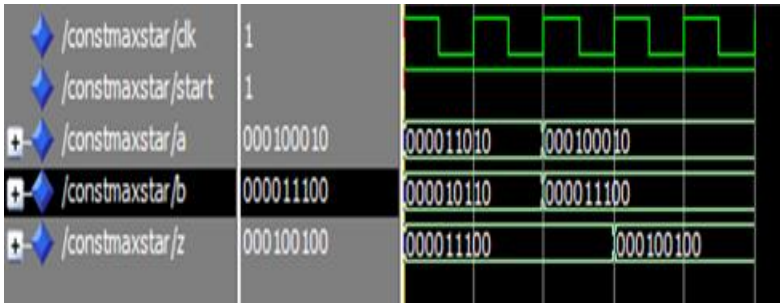

Fig 6: Output waveform of Constant Max* operation

Comparison of LUT Log MAP and Constant Log MAP algorithm

In order to determine the best algorithm, LUT Log MAP algorithm and constant Log MAP algorithm are synthesized using Xilinx ISE and device utilization values are estimated.

The below tables shows the estimated number of components required for both the algorithms.

Table 1: LUT Log MAP algorithm

Device Utilization Summary (estimated values)		
Logic Utilization	**Used**	**Available**
Number of Slices	22	2448
Number of 4 input LUTs	40	4896
Number of bonded IOBs	29	92
Number of GCLKs	1	24

Table 2: Constant Log MAP algorithm

Device Utilization Summary (estimated values)		
Logic Utilization	**Used**	**Available**
Number of Slices	20	2448
Number of 4 input LUTs	37	4896
Number of bonded IOBs	29	92
Number of GCLKs	1	24

The design summary of LUT Log MAP algorithm and constant Log MAP algorithm shows that constant Log MAP algorithm requires lesser number of components. The number of 4 input LUTs required in constant Log MAP algorithm is 37 and that required in LUT Log MAP algorithm is 40. Number of slices in constant Log MAP algorithm is 20 and that in LUT Log MAP is 22. So using Constant Log MAP algorithm in turbo decoder will be more efficient.

Simulation Results of Turbo Decoder

The input to the decoder is encoded data. The decoder reads encoded data bits from the data file when write signal (writeout) = 0. The decoded output is obtained when write signal (writeout) =1.

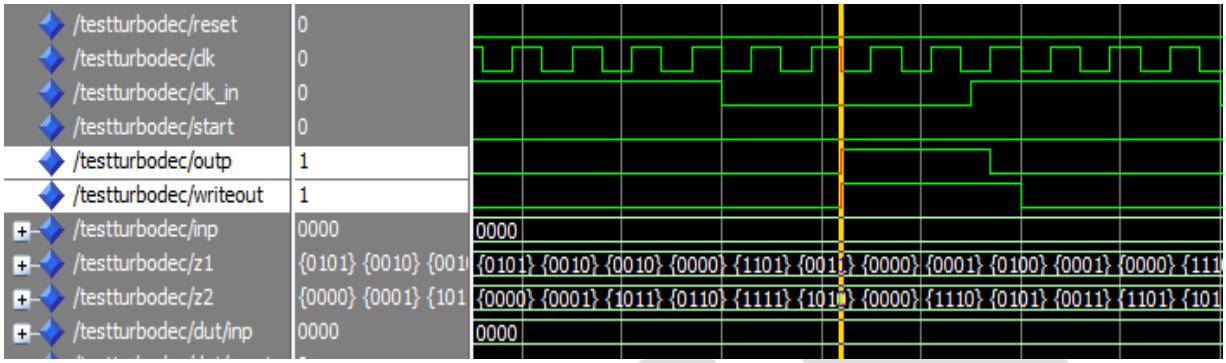

Fig 7 : Output waveform of decoder output when write signal (writeout) = 1

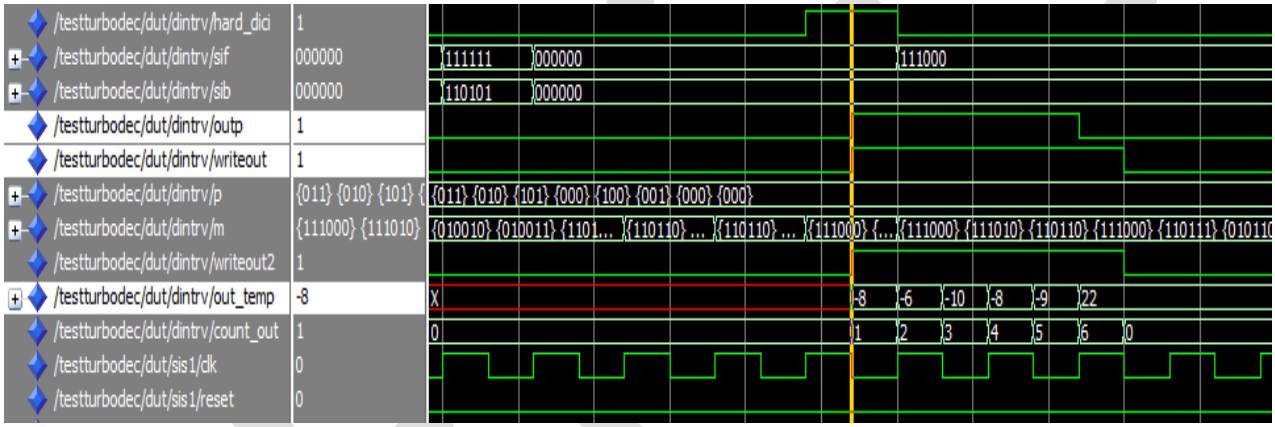

Fig 8: Output waveform of decoder output in decimal format

The out_temp signal in figure 8 is the final output of deinterleaver. These are the LLR values. Here the LLR values are shown in decimal format. When LLR value is negative, it is decoded as 1 and when LLR value is positive, it is decoded as 0. The LLR values obtained are -8, -6, -10, -8, -9 and 22. So the decoded bits are 111110 as shown by outp signal.

Comparison of Turbo decoder simulated using LUT Log MAP and Constant Log MAP algorithm

The turbo decoder was simulated using LUT Log MAP algorithm and constant Log MAP algorithm. The turbo encoder using both the algorithms were synthesised using Xilinx ISE to obtain their estimated device utilisation values and its was compared.

Table 3: Turbo decoder using LUT Log MAP algorithm

Device Utilization Summary (estimated values)		
Logic Utilization	**Used**	**Available**
Number of Slices	9823	14752
Number of Slice Flip Flops	1897	29504
Number of 4 input LUTs	18829	29504
Number of bonded IOBs	10	250
Number of MULT18X18SIOs	8	36
Number of GCLKs	3	24

Table 4: Turbo decoder using Constant Log MAP algorithm

Device Utilization Summary (estimated values)		
Logic Utilization	**Used**	**Available**
Number of Slices	9590	14752
Number of Slice Flip Flops	1182	29504
Number of 4 input LUTs	18058	29504
Number of bonded IOBs	10	250
Number of MULT18X18SIOs	8	36
Number of GCLKs	3	24

The design summary shows comparison of the LUT Log MAP algorithm and Constant Log MAP algorithm used for the simulation of turbo decoder, there is a large variation in number of components in both algorithms. The total number of slice required for turbo decoder using LUT Log MAP algorithm is 9823 and that using constant Log MAP algorithm is 9590. The number of slice flip flop used in LUT Log algorithm is 1897 and that using constant Log MAP algorithm is 1182. These differences in logic utilisation in turbo decoders using both the algorithms bring a large difference in its power consumption. This shows that an efficient turbo decoder can be developed using constant Log MAP algorithm.

ACKNOWLEDGMENT

The authors wish to acknowledge the Management of Toc H Institute of Science and Technology for their whole hearted support and also to the ECE department for guiding us in developing this journal.

CONCLUSION

This paper mainly focuses on simulating an efficient turbo encoder and decoder for wireless sensor networks. By using the ideas perceived from literature survey it was found that turbo decoder has better performance than viterbi decoder for its application in wireless sensor networks. The turbo encoder can be simulated using different methods. The turbo encoder was simulated by two flip flops method. For the simulation of turbo decoder, LUT Log MAP algorithm and constant Log MAP algorithm was compared. The result shows that turbo encoder using constant Log MAP algorithm was more efficient than turbo encoder using LUT Log MAP algorithm. An energy efficient turbo encoder and decoder were simulated successfully using ModelSim. The turbo decoder is used in digital communications like computer, 3G mobile communication, telemetry and deep space exploration.

REFERENCES:

[1] F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," The International Journal of Computer and Telecommunications Networking, vol. 52, pp. 292–422, 2008.
[2] P. Corke, T. Wark, R. Jurdak, H. Wen, P. Valencia, and D. Moore, "Environmental Wireless Sensor Networks," Proc. of the IEEE, vol. 98, no. 11, pp. 1903–1917, 2010.
[3] Patric Robertson, Peter Hoeher and Emmanuelle Villebrun "Optimal and Sub-Optimal Maximum a Posteriori Algorithm Suitable for Turbo Decoding," European Transactions on Telecommunication, vol.8, pp-119-125, March 1997

[4] Warren J. Gross and P. Glenn Gulak, "Simplified MAP Algorithm Suitable for Implementation of Turbo Decoders", IEEE Electronics Letter,1998

[5] Alexander Worm, Holger Lamm and Norbert When, "VLSI Architectures for High-speed MAP Decoders," IEEE transaction on VLSI design, pp: 446-453, 2000

[6] Zhongfeng Wang, Zhipei ChiandKeshab K. Parhi, "Area-Efficient High-Speed Decoding Schemes for Turbo Decoders," IEEE Transactions on VLSI, vol. 10, no. 6, pp: 902-912, 2002

[7] Muhmoud Elassal and Magdy Buyoumi, "A Low Power Turbo Decoder Architecture," IEEE transactionon signal processing systems, pp:105-110, 2003

[8] Ibrahim Al-Mohandes and Mohamed Elmasry, "A Low-Power 5 Mb/s Turbo Decoder for Third-Generation Wireless Terminals",IEEE Canadian Conference on Electrical and Computer Engineering, vol. 4, pp:2387-2390,2004.

[9] Indrajit Atluri, Ashwin K. Kumaraswamy and V.A.Chouliaras, "Energy efficient architectures for the Log-MAP decoder through intelligent memory usage", IEEE Computer Society Annual Symposium on VLSI,pp-263-265, 2005.

[10] Santosh Shah & V. Sinha, "Iterative decoding vs. Viterbi decoding: acomparison",National Conference on Communications, IIT Bombay, p. 494-497, February 2008.

[11] Pallavi Reddy, Fabien Clermidy Rasheed Al Khayat and Amer Baghdadi, "Power Consumption Analysis and Energy Efficient Optimization for Turbo Decoder Implementation", IEEE Int Symposium On System On Chip,pp:12-17, 2010.

[12] Liang Li, Robert G. Maunder, Bashir M. Al-Hashim Mark Zwolinski, and LajosHanzo, "Energy-Conscious Turbo Decoder Design: A Joint Signal Processing and Transmit Energy Reduction Approach", IEEE Transactions on Vehicular Technology, vol. 62, no. 8,pp-3627-3638 Oct 2013.

[13] Liang Li, Robert G. Maunder, Bashir M. Al-Hashimi and Lajos Hanzo, "A Low-Complexity Turbo Decoder Architecture for Energy-Efficient Wireless Sensor Networks", IEEE Transactions On VLSI, vol. 21, pp: 14-22,2013

[14] Maurizio Martina, Stylianos Papaharalabos, P.Takis Mathiopoulos, and Guido Masera, "Simplified Log-MAP Algorithm for Very Low-Complexity Turbo Decoder Hardware Architectures", IEEE Transactions on Instrumentation and Measurement, vol. 63, no. 3, pp:531-537, Mar 2014