Comparative Analysis of Turbo and LDPC Codes for Reduced Storage and Retrieval of Data

MS. N. GOPIKA RANI[1], DR. G. SUDHA SADASIVAM[2], MR.R.M.SURESH[3]
Department of Computer Science and Engineering
PSG College of Technology Coimbatore
INDIA
[1]gopika79@yahoo.com, [2]sudhasadhasivam@yahoo.com, [3]sureshrmse@gmail.com

Abstract

Turbo codes are the channel coding scheme used in wireless cellular networks as they are able to reach close to the Shannon limit. This paper proposes the use of turbo codes and LDPC codes for storage of data. Turbo encoding can be performed by using parallel Recursive Systematic Convolutional (RSC) encoder and an interleaver while turbo decoding is based on Bahl Cocke Jelinek and Raviv (BCJR) algorithm, the Maximum Aposterior Algorithm (MAP).Low Density Parity-Check (LDPC) codes encoding technique are based on the generator matrix value of the original code word to be identified. In LDPC decoding Hard-decision decoding algorithm is followed. Finally, a comparative analysis on turbo and LDPC codes is presented. Theoretical and experimental results show turbo codes perform better than LDPC codes.

Key-Words: - BCJR algorithm, Check nodes, encoding algorithm, Hard-decision decoding algorithm, LDPC codes, Turbo codes, Variable nodes.

1 Introduction

Over the years, there has been a spectacular growth in digital communications especially in the fields of cellular/PCS, satellite, and computer communication. In these communication systems, information is represented as a sequence of binary bits. This information is then mapped, modulated and transported in communication channels, which introduce noise and losses. For these reasons it is important to conceive efficient coding algorithms. LDPC and turbo codes enables telecommunication channels to transfer error free data. Error correction will be single bit in the case of hamming code. In a real world the error can be introduced at many places while data gets transmitted across the communication channel. So by using turbo code the user will be able to correct multiple bits of errors. The accuracy of the turbo codes is higher than hamming code.

Turbo codes are a recent development in the field of error control coding. These codes are very attractive due to their outstanding performance, clear reliable communication limit given by Shannon limit. Turbo codes involve iterative soft-input soft-output (SISO) decoding of concatenated codes separated by an interleaver. The discovery of turbo codes and the subsequent rediscovery of low-density parity-check (LDPC) codes represent major milestones in the field of channel coding. These two classes of codes can achieve realistic bit error rates (BERs), and with signal-to noise ratios (SNRs) that are only slightly above the minimum SNR for a given channel and code rate established by Shannon's original capacity theorems.

Low-Density Parity-Check (LDPC) codes are a class of linear block codes. The name comes from the characteristic of their parity-check matrix which contains only a few 1's in comparison to the amount of 0's. Basically there are two different possibilities to represent LDPC codes. They are linear block codes described via matrices and graphical representation. This paper proposes the use of turbo codes and LDPC codes for storage of data.

The rest of the paper is organized as follows: section II describes the related work about recursive convolutional code, interleaving process and hard-

decision decoding algorithm. Section III and section IV explains about the error correcting codes and system design. Section V presents the sample solution. Section VI provides experimental result about turbo and LDPC encoding and decoding followed by conclusion and references.

2 Related Work

The turbo encoding is built using two identical Recursive Systematic Convolutional (RSC) code with parallel concatenation. The encoder used in our experimentation is an RSC with r = ½. The two block encoders are separated by an interleaver. Only one of the systematic outputs from the two component encoders is used, because the systematic output from the other component encoder is just a permuted version of the chosen systematic output. Figure 1 shows three outputs, two of them issued by RSC encoder1 where c_1 is the systematic sequence and c_2 the recursive convolutional sequence [1]. Also, it is important to notice that RSC encoder 2 issues a convolutional part c_3, the other output is discarded. Turbo encoder can detect and correct multiple bits of errors.

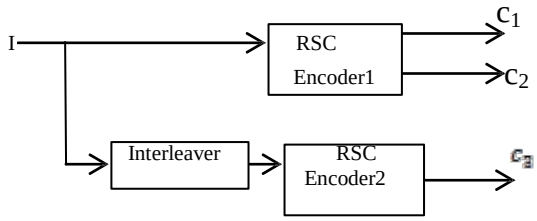

Fig. 1 Turbo Encoder Circuit

The turbo decoding is built using Bahl Cocke Jelinek and Raviv (BCJR) algorithm. It consists of a pair of decoders which work cooperatively in order to refine and improve the estimate of the original bits. The decoder belongs to class of Maximum aposteriori probability (MAP) algorithms. The interleaving function denoted by π^{-1} is used to break low weight input sequences, and hence increases the code free hamming distance. Deinterleaving denoted by π is used to recover original message. The figure 2 shows the structure of a parallel turbo decoder.

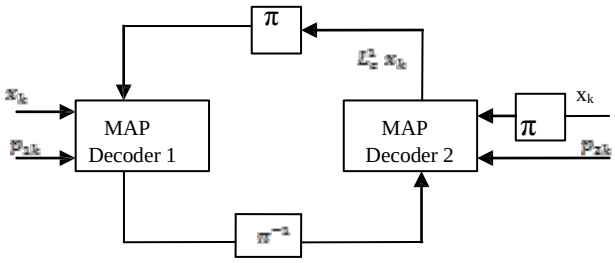

Fig. 2 Turbo Decoder Circuit

2.1.1 LDPC Encoding Methods

Step 1:
In the LDPC codes the encoding is performed by the input string from user, and then converts that input string to equivalent binary code.

Step 2:
After convert string to binary code value, multiply with generator matrix ($G= [P|I_k]$) to identify original code word.

Step 3:
Then generate error in original code word by changing any one of binary bits to convert a 0 to 1 or 1 to 0.

Step 4:
Check that code word have any error bits then if any error occurs, then send that data bits to decoder to recover the original message bits [7].

Step 5:
In this paper, for decoding, Hard- decision decoding algorithm technique is be followed in LDPC decoding.

Step 6:
The parity check matrix n x m for (8, 4) code is shown in Eqn 1.

$$H= \begin{pmatrix} 0\,1\,0\,1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,1 \\ 1\,0\,0\,1\,1\,0\,1\,0 \end{pmatrix}$$

Eqn. 1 Parity Check Matrix

Figure (3) shows the tanner graph which is formed from above parity check matrix. It is a bipartite graph. The two types of nodes in tanner graph are variable nodes (v-nodes) and check nodes(c-nodes). Check node f_i is connected to variable node c_j if the element h_{ij} of H is a 1.

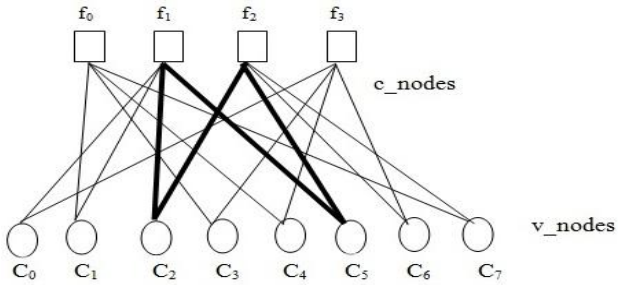

Fig. 3 Tanner Graph

2.1.2 LDPC Decoding Method

Step 1:
In decoding, first step all v-nodes c_i send a "message" to their c-nodes containing the bit they believe to be the correct one for them ($c_i \rightarrow f_i$).

Step 2:
Then the every check nodes f_j calculate a response to every connected variable node ($f_j \rightarrow c_i$). This step is show in below Table 1.

Table 1 Message Received and Sends by c-nodes

c-nodes	*Received/sent*				
f_0	*Received:*	$C_1 \rightarrow 1$	$C_3 \rightarrow 1$	$C_4 \rightarrow 0$	$C_7 \rightarrow 1$
	Sent:	$0 \rightarrow C_1$	$0 \rightarrow C_3$	$1 \rightarrow C_4$	$0 \rightarrow C_7$
f_1	*Received:*	$C_0 \rightarrow 1$	$C_1 \rightarrow 1$	$C_2 \rightarrow 0$	$C_5 \rightarrow 1$
	Sent:	$0 \rightarrow C_0$	$0 \rightarrow C_1$	$1 \rightarrow C_2$	$0 \rightarrow C_5$
f_2	*Received:*	$C_2 \rightarrow 0$	$C_5 \rightarrow 1$	$C_6 \rightarrow 0$	$C_7 \rightarrow 1$
	Sent:	$0 \rightarrow C_2$	$1 \rightarrow C_5$	$0 \rightarrow C_6$	$1 \rightarrow C_7$
f_3	*Received:*	$C_0 \rightarrow 1$	$C_3 \rightarrow 1$	$C_4 \rightarrow 0$	$C_6 \rightarrow 0$
	Sent:	$1 \rightarrow C_0$	$1 \rightarrow C_3$	$0 \rightarrow C_4$	$0 \rightarrow C_6$

Step 3:
The v-nodes receive the messages from the check nodes and use this additional information to decide if their originally received bit is OK ($c_i \rightarrow f_j$). Repeat the above steps for all data bits is shown Table 2.

Now the v-nodes can send another message with their (hard) decision for the correct value to the check nodes. An error free received code word would be e.g. c = [1 0 0 1 0 1 0 1]. Let's suppose that we have a Binary Symmetric Channel (BSC) channel and the received code word with one error bit flipped to 1. LDPC codes can correct multiple bits of errors compared to turbo codes. So by using LDPC codes, an error free data can be transmitted through a communication channel.

The probability for occurrence of error can be high between sender and receiver side, when data is transmitted in sequential method. The operation relies on soft decision information learned from each other. To reduce the error rate and to increase the efficiency without data loss turbo codes and LDPC codes can be used. The decision value in Table 2 is the decoded value. If the decision value matches the received code word, there is no error otherwise an error has occurred in corresponding bit position.

Table 2 Decoding Algorithm

v-node	y_i *received*	*Message from check nodes*		*Decisions*
C_0	1	$f_1 \rightarrow 0$	$f_3 \rightarrow 1$	1
C_1	1	$f_0 \rightarrow 0$	$f_1 \rightarrow 0$	0
C_2	0	$f_1 \rightarrow 1$	$f_2 \rightarrow 0$	0
C_3	1	$f_0 \rightarrow 0$	$f_3 \rightarrow 1$	1
C_4	0	$f_0 \rightarrow 1$	$f_3 \rightarrow 0$	0
C_5	1	$f_1 \rightarrow 0$	$f_2 \rightarrow 1$	1
C_6	0	$f_2 \rightarrow 0$	$f_3 \rightarrow 0$	0
C_7	1	$f_0 \rightarrow 1$	$f_2 \rightarrow 1$	1

Hamming code is well known for its single-bit error detection and correction. To provide such a capability, it introduces 4 redundancy bits in a 7-bit data item [1]. These redundancy bits are to be interspersed at bit positions (number of bits = 0, 1, 2, 3) with the original data bits. After error detection and correction, the data bits are reassembled by removing the redundancy bits. Reed-Muller and Reed Solomon Codes are non-binary cyclic error correcting code, and it is suitable as multiple-burst bit-error correcting codes [14]. They describe a systematic way of building a code that could detect and correct multiple random symbol errors. By adding t check symbols to the data, an RS code can detect any combination of up to t erroneous symbols or correct up to [t/2] symbols.

Convolutional codes encode bits based upon a state which is determined by summing a fixed set of previous bits. Each input bit is manipulated in a few different ways to produce several outputs bits [12].

Therefore each output bit conveys the combined information of many different input bits. The state is initialized to a key that is initially passed from encoder to decoder. Each m-bit information symbol to be encoded is transformed into an n-bit symbol, where m/n is the code rate ($n \geq m$) and the transformation is a function of the last k information symbols, where k is the constraint length of the code.

Linear codes are used in forward error correction and are applied in methods for transmitting symbols on a communications channel so that, if errors occur in the communication, some errors can be corrected or detected by the recipient of a message block. The code words in a linear block code are blocks of symbols which are encoded using more symbols than the original value to be sent. A linear code of length n transmits blocks containing n symbols. As a consequence, up to two errors per code word can be detected and a single error can be corrected. The number of code words 2_k has 2_k distinct messages. The set of vectors are linearly independent since we must have a set of unique code words. Local Decoding and Testing of codes is single bits of the message can be probabilistically recovered by only looking at a small number of positions of a code word, has been corrupted at some constant fraction of positions. Locally testable codes are error correcting codes for which it can be checked probabilistically whether a signal is close to a code word by only looking at a small number of positions of signal.

In coding theory, Reed–Solomon (RS) codes are non-binary cyclic error-correcting codes invented by Irving S. Reed and Gustave Solomon. They described a systematic way of building codes that could detect and correct multiple random symbol errors. By adding t check symbols to the data, an RS code can detect any combination of up to t erroneous symbols, or correct up to $\lfloor t/2 \rfloor$ symbols. As an erasure code, it can correct up to t known erasures, or it can detect and correct combinations of errors and erasures. Furthermore, RS codes are suitable as multiple-burst bit-error correcting codes, since a sequence of b + 1 consecutive bit errors can affect at most two symbols of size b. The choice of t is up to the designer of the code, and may be selected within wide limits.

In the proposed method the turbo and LDPC encoding and decoding are to be performed for storage of data. In hamming code minimum number of error bits can be detected and corrected. By using turbo codes and LDPC codes we can detect and correct multiple numbers of bits. So turbo codes and LDPC codes are more advance and efficient in detecting and correcting more number of errors while storing and retrieval of data.

3. System Design

The proposed system aims at using turbo codes and LDPC codes for storage. This section discusses in detail about the proposed approaches.

3.1.1 Turbo Encoding and Decoding

Turbo Encoding is built using two identical recursive systematic convolutional (RSC) code with parallel concatenation. One input is sent through an interleaver and another is sent directly to RSC encoder. The output of RSC encoder is sent back as input to a turbo encoder in cyclic manner. Then the results of turbo encoding is stored in disks parity check bits are generated. The coded values are then decoded. In decoding BCJR algorithm, the maximum aposteriori algorithm (MAP) is followed to recover the original message bits. It consists of a pair of decoders which work cooperatively in order to refine and improve the estimate of original bits. The decoded information is cycled around loop until the soft decisions converge on stable set of values. Fig. 4 shows the flow of encoding and decoding of turbo codes.

3.1.2 LDPC Encoding and Decoding

LDPC encoder converts the input string to equivalent binary bits. This is multiplied with generator matrix ($G= [P|I_k]$) to identify the original code word. If the code word has any error bits, then the data bits are sent to decoder to recover the original message bits. LDPC codes can generate a single bit or multiple bit error by flipping any one bit. The encoding is easier to implement when compared to LDPC decoding. Fig 5 shows detailed description of LDPC encoding.

The input should be given in the form of dataset, and then it should be converted into equivalent binary bits from a character to binary value. Multiply with generator matrix to identify error bits during the data transmission. If error is identified then send it to decoder to recover the original message bits, if no error found stop the encoding process after sending the message bits to decoder to recover original message bits.

LDPC decoding should be performed for the purpose of to recover the original message bits during data transmission through the

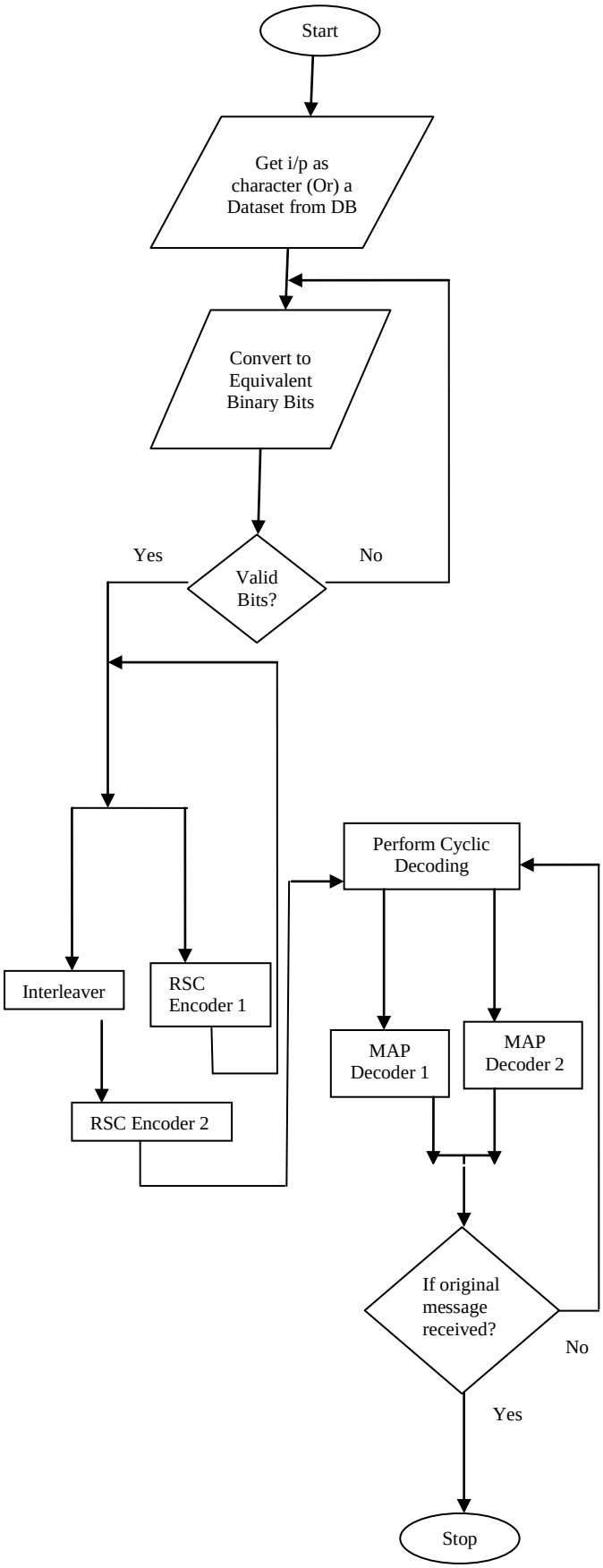

Fig. 4 Flowchart for Turbo Encoding and Decoding

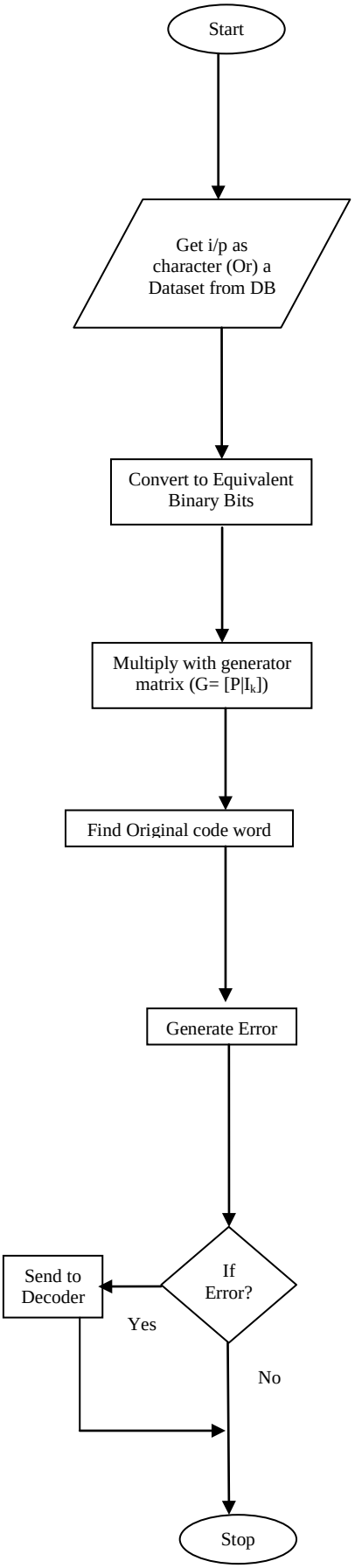

Fig. 5 Flowchart for LDPC Encoding

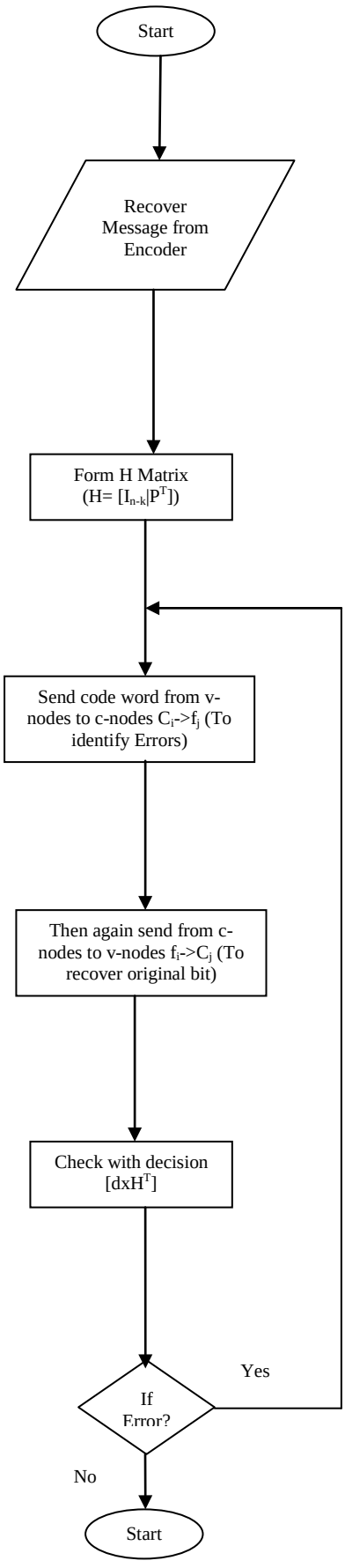

Fig. 6 Flowchart for LDPC Decoding

communication channel. Based on the Hard Decision Decoding value the decoding process should be done in loop until the original message to be recovered.

In Hard-Decision decoding algorithm, H matrix ($H= [In-k|P^T]$) is formed to perform decoding operation to recover the original message bits as shown in Fig 6. In the first step all v-nodes ci send a "message" to their c-nodes f_j containing the bit they believe to be the correct one for them (ci → fj). In the second step every check node calculates a response to every connected variable node (fj→ ci). Then the v-nodes receive the messages from the check nodes and use this additional information to decide if their originally received bit is OK (ci → fj). Finally decision value is multiplied with inverse of H matrix to identify the position of error occurred.

4 Sample Solutions

4.1.1 Turbo encoding

Step 1
Convert a character to equivalent binary bits. For example consider the character 'a' and its equivalent binary value is:

$$a=01100001$$

Step 2
In Turbo encoder is perform as based on RSC encoder and a random interleaver. After convert a character to binary value perform interleaver function to reduce hamming bit distance vale.

After interleaver function= 00101101001100100011

Step 3
The final Turbo RSC Encoder result,

Xor with 1 operation C_1 00101101
Xor with 2 operation C_1 01100110
Xor with 1 operation C_2 00110111
Xor with 1 operation C_2 10110001
Final output is C_1 00101101
Final output is C_2 01100110
Final output is C_3 01110011
Process time for encoding 100ms

After performing the encoding operation then add the error by changing any one bit from 0 to 1 or 1 to 0 and send to decoder to recover the original bits.

4.1.2 Turbo decoding

The turbo decoding is based on BCJR algorithm to recover an original message bits. The turbo decoding operation to be performed in loop until the original message is to recover for transfer error free data through communication channel

Error occurred in the position 4
Error corrected at the position 4
The value of r_k (s1, s) is 4.840457
DECODING value 24.83

4.1.3 BER chart for Turbo codes

The turbo decoding is based on BCJR algorithm by give a turbo encoding result as input to the turbo decoding to recover the original message bits. Below figure shows Bit Error Rate (BER) for turbo encoder. Based on the BER chart also, the efficiency is identified.

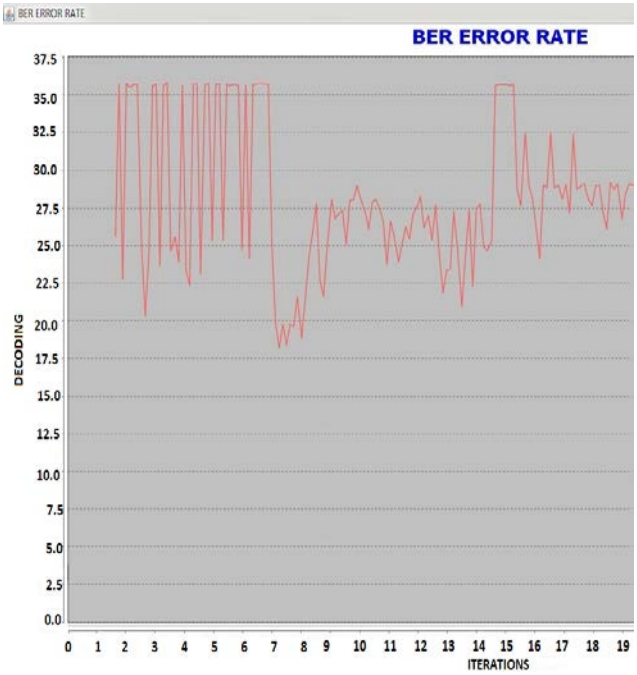

Fig.7 Bit Error Rate for turbo encoding

Based on the above result turbo codes can correct multiple bits of error. To this end, we just had to perform soft decoding at the end of each iteration to see that the decoded message is getting closer and closer to the initial message when the number of iteration is increasing.

4.1.4 LDPC encoding

Step 1

Convert a character to equivalent binary bits. For example consider the character 'a' and its equivalent binary value is:

a=01100001

Step 2

Create LDPC matrix as based on iteration

Frame: 1
Message encoded.
Elapsed time is 0.519324 seconds.

Step 3

The parity check matrix to be created,

$$H=\begin{pmatrix} 0\,1\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,1\,1\,1\,0\,0\,0 \\ 1\,1\,0\,0\,0\,0\,1\,0 \\ 1\,1\,1\,1\,1\,0\,0\,0 \end{pmatrix}$$

After performing the encoding operation, the result is passed to the decoder recover the original bits.

4.1.5 LDPC decoding

In LDPC the decoding operation is performed as based on Hard-Decision decoding algorithm. The original messages are to be recovered as based on column wise is shown below,

Columns 1

1 0 1 1 0 1 1 0

4.1.6 BER chart for LDPC codes

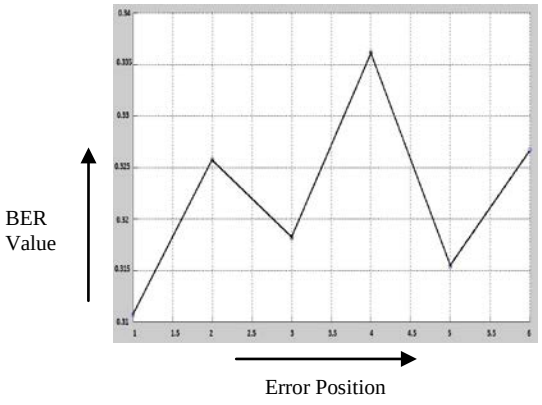

Fig.8 LDPC Bit Error Rate Result

The BER for LDPC codes is formed as based on the number of error occurring position and with corresponding bit error rate value. As based on BER error rate the performance of error correcting codes are to be identified. By using LDPC codes multiple bits of error can be detected and corrected.

5 Experimental Results

The turbo code and LDPC code are relatively new codes that offer a wide range of flexibility in terms of performance, complexity, and code rate. The turbo codes and LDPC codes have a better error correcting rate of approximately 0.8, which is a 60% improvement in data rate over the standard convolutional code.

The turbo codes and LDPC codes are compared on the basis of Bit Error Rate (BER) and also on the basis of time complexity. LDPC codes are more accurate than turbo codes in correcting the error when transferring the data through a communication channel. The interleaver plays a major role in the performance of turbo codes. In our experiment random interleaver method is be used, with fixed random permutation to scramble the data.

Experiments are done on a cluster of about 7 systems with Pentium IV Processor: Intel® Core™2 Duo, 2.54 GHz processor, 1GB RAM, 160GB Hard Disk interconnected with Ethernet LAN capable of transmitting at a maximum speed of 4 Mbps. Results are analyzed based on time and space complexity.

5.1.1 Time Complexity Analysis

Table 3 Time Complexity Analysis

S.NO	Data Size (KB)	LDPC Codes		Turbo Codes	
		Encoding Process time (in milliseconds)	Decoding Process time (in milliseconds)	Encoding Process time (in milliseconds)	Decoding Process time (in milliseconds)
1	1	1100	654	100	81
2	10	1658	1000	345	257
3	50	4563	4230	1954	1643
4	100	9215	8115	6512	5092
5	150	16543	14700	12056	11092

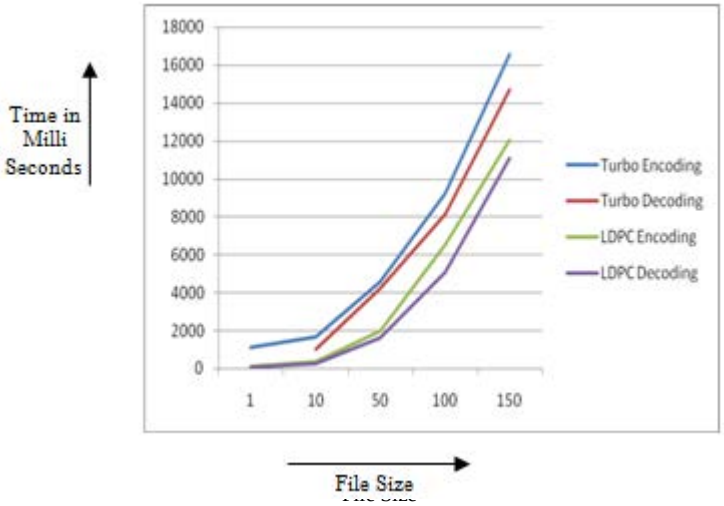

Fig.9 Comparison graph for turbo codes and LDPC codes

5.1.2 Order of Complexity

Table 4 Order Complexity Analysis

S.NO	Error Correcting Codes	Encoding	Decoding
1.	Turbo Codes	$O(n^2)$	$O(n)$
2.	LDPC Codes	$O(n^2)$	$O(n^2)$

In case of LDPC, as the number of check nodes and message nodes increases the complexity of decoding and the computation will be slower. In case of turbo codes, decoding computation will be faster as based on time complexity and space savings. In case of Low Density Parity Check decoding computation will be slower as the number of updations between

S.No	Error blocks	Turbo codes	LDPC
1	X=2	47%	33%
2	X=4	43%	33%
3	X=6	40%	33%
4	X=8	36%	33%
5	X=10	33%	33%

bit nodes and check nodes will increase.

5.1.3 Space Savings

Table 5 Space Savings

1 Stripe=10blocks

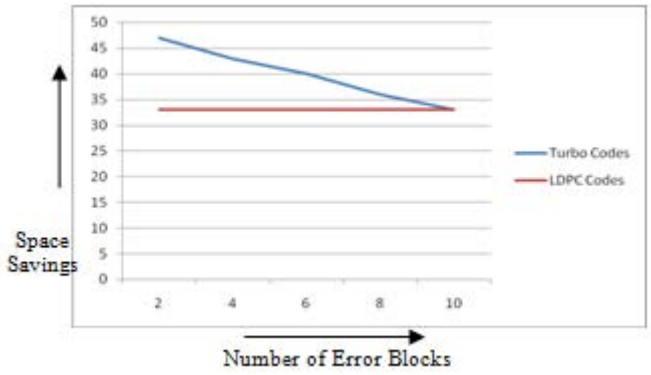

Fig.10 Space Savings of Turbo code and LDPC

In case of LDPC code for each message bit the corresponding parity bit will be present. It is like keeping 2 copies of data. Turbo codes can save lots of space without compromising the corruption probability. Figure 10 shows the space savings of turbo and LDPC.

5.1.4 Comparison of turbo codes and LDPC codes

The turbo codes and LDPC codes are compared based on the time complexity and space savings. In time complexity analysis the turbo encode and decode method will take less time to detect and correct the errors for recovering original message when compared to the LDPC encoding and decoding. In space savings the turbo code occupies less space when compared to the LDPC codes. LDPC codes can also detect and correct mutiple bits of error.

6 Conclusions

This paper summaries the important concepts regarding turbo codes and low density parity-check codes (LDPC). Turbo codes and LDPC codes can detect and correct multiple bits of error. But turbo codes are more accurate and fast for encoding/decoding and for correcting multiple errors. Therefore, turbo codes are highly desirable for storage and retrieval of data for large clusters. Finally based on comparison chart the turbo method will be more accurate in detecting and correcting multiple bits of error. In the future we will try to compare with other error correcting codes for storage and retrieval of data for large clusters.

7 Acknowledgements

The authors convey their heartfelt thanks to Dr. R. Rudramoorthy, Principal, PSG College of Technology and Dr. R. Venkatesan, Professor and Head, Department of computer Science & Engineering, PSG College of Technology. This work is performed in the grid / cloud computing lab at PSG College of Technology.

References:

[1] R.Ma and S.Cheng, "*Hamming coding for multiple sources*," in Procceding ISIT, June 2010.

[2] Walid Boumerdassi, Etienne Collange and Team Space Busters, "*Turbo Codes Encoding/Decoding*", December 9, 2010.

[3] Onur atar, Murat h.sazli, Hakk Gokhan ILK, "*FPGA Implementation of Turbo Decoders*", KTTO 2011 11th International Conference on Knowledge in Telecommunication Technologies and Optics, pp. 103-108, Szczyrk, Poland, June 22-24 2011.

[4] Emilia Käsper, "*Turbo Codes*"

[5] Jakob Dahl Andersen, "Product Codes for Optical Communication", March 17, 2010 IEEE Explore.

[6] Wagner J. Okano, Fernando Ciriaco and Taufik Abrao, Member, IEEE "*Channel Reliability for Turbo DS/CDMA Systems under Rayleigh Fading and Multiple Access Interference*", ieee latin america transactions, VOL. 8, NO. 1, March 2010.

[7] Bernhard M.J. Leiner, "*LDPC Codes a brief Tutorial*, April 8, 2005.

[8] Kai Zhang, Xinming Huang, Senior Member, IEEE, and Zhongfeng Wang, Senior Member, IEEE, "*A High-Throughput LDPC Decoder Architecture With Rate Compatibility*", ieee transactions on circuits and systems-i: regular papers, vol. 58, NO. 4, April 2011.

[9] Nastaran Mobini, "*New Iterative Decoding Algorithms for Low-Density Parity-Check (LDPC) Codes"*, August, 2011.

[10] Xinmiao Zhang, Senior Member, IEEE, Fang Cai, Student Member, IEEE, and Shu Lin, Life Fellow, IEEE, "*Low Complexity Reliability- Based Message-Passing Decoder Architectures for Non-Binary LDPC Codes*", IEEE transactions on very large scale integration (VLSI) systems, VOL. 20, NO. 11, November 2012.

[11] Alexander Barg, Fellow, IEEE, and Arya Mazumdar, Student Member, IEEE, "*On the Number of Errors Correctable with Codes on Graphs*", IEEE transactions on

information theory, VOL.57, NO.2, February 2011.

[12] Mahe Jabeen, Salma Khan, "*Design of Convolution Encoder and Reconfigurable Viterbi Decoder*", International Journal of Engineering and Science ISSN: 2278-4721, Vol. 1, Issue 3 (Sept 2012), PP 15-21.

[13] Ripple Dhingra and Danvir Mandal, "*Convolutional Code Optimization for Various Constraint Lengths using PSO*", International Journal of Electronics and Communication Engineering. ISSN0974-2166 Volume 5, Number2 (2012), pp. 151-157.

[14] Jatinder Singh and Jaget Singh, "*Design of Reed-Muller Encoder for Multiple Errors Detection*", International Journal of Computer Applications (0975 – 8887 Volume 45– No.21 May 2012.