# Performance Evaluation of LDPC decoding Algorithm for GF (2)

[1]Patel Jigisha, [2]Patel Ruchitakumari
Assistant Lecturer,
E & C Department,
Sardar Vallabhai National Institute of Technology, Surat, Gujarat, India.

***Abstract* - Low Density Parity Check (LDPC) codes are one of the block coding techniques that can approach the Shannon's limit within a fraction of a decibel for high block lengths. In many digital communication systems, these codes have strong competitors of turbo codes for error control. LDPC codes performance depends on the excellent design of parity check matrix and many diverse research methods have been used by different study groups to evaluate the performance. Unlike many other classes of codes, LDPC codes are already equipped with a fast, probabilistic decoding algorithm. This makes LDPC codes not only attractive from a theoretical point of view, but also very suitable for practical applications. This paper throws hard decision and soft decision algorithm of LDPC. The hard decision algorithm contains Bit flipping algorithm and Soft decoding algorithm contains Sum product algorithm which is used to correct burst error. Sum Product algorithm is also called as belief propagation. Bit flipping and Sum Product algorithms are explained and can be implemented for the generated for LDPC code with BPSK modulation.**

***Index Terms* - Encoder, Bit Flipping Algorithm, Sum Product Algorithm, AWGN channel**

---

## I. INTRODUCTION

Low-density parity-check (LDPC) codes are a class of linear block codes with implementable decoders, which provide near-capacity performance on a large set of data transmission and data-storage channels. Low-density parity-check (LDPC) codes were first introduced by R.G.Gallager[1] and rediscovered by MacKay in 1996[2]. Davey and Mackay rediscovered that by extending the order of Galois Field GF (q)[3]. This class of codes is known as non-binary LDPC codes. These codes are showing more and more competitive and have become some error-correcting code standards or the candidate standard in communication system.

LDPC codes were invented by Gallager in his 1960 doctoral dissertation and were mostly ignored during the 35 years that followed[2]. One notable exception is the important work of Tanner in 1981, in which Tanner generalized LDPC codes and introduced a graphical representation of LDPC codes, now called a Tanner graph[4]. The study of LDPC codes was resurrected in the mid 1990s with the work of MacKay, Luby, who noticed, apparently independently of Gallager's work, the advantages of linear block codes with sparse (low-density) parity-check matrices[5].

The Bit-Flipping (BF) decoding algorithm[6], which was proposed by Gallager in 1962, and rediscovered by Mackay and Neal in 1996, is based on the hard decision of symbol. BF is simple and easy to be implemented, but has only limited decoding performance. Well designed LDPC codes decoded with iterative decoding based on belief propagation algorithm, such as the sum product algorithm (SPA)[7] achieve performance close to the Shannon limit 0.0045[2].

The paper is organized as follows: Section I the introduction of LDPC is explained. Section II is containing encoding method of LDPC for generating codeword that is transmitted on channel. Section III decoding methods of LDPC in this section Bit Flipping (Hard Decision) and Sum Product (soft Decision) algorithm of the decoder is explained. Section IV Results and analysis for different decoding algorithm explained. Section V contains Conclusion.

## II. ENCODING

There are plenty of methods used for generating parity check matrix and Generator matrix. Parity check matrix is generated using Base matrix which consists of many sub matrices within it. Memory requirement is quite crucial issue during the VLSI implementation of LDPC codes. Generation of Parity check matrix using Base matrix requires less memory. Parity check matrix is obtained after expansion of Base matrix. Base matrix has elements like 0, -1 and any number between 0 to Z-1, where Z is an expansion factor. During expansion, -1 is replaced by Z x Z all zero matrix and any number between 0 to Z-1 is replaced by Z x Z identity matrix circular shifted by that number that shown in Fig. 1. Many of the standards like 802.11n, 802.16e etc. uses this method. Generator matrix is generated from parity check matrix using Gauss elimination method [5].

### *A.* Gauss Elimination Method

The conventional method of encoding with the generator matrix is derived from parity check matrix by (modulo-2) Gauss elimination. This method is used for finding the unknown generator matrix G from the Parity Check Matrix (PCM) achieved through row permutations, modulo-2 sums of rows and also some column permutation [5].

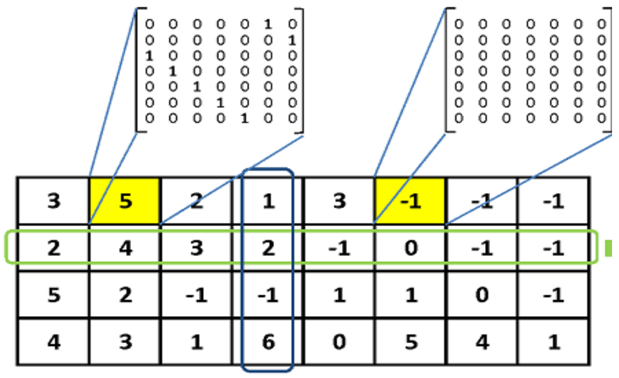

Fig. 1 Base matrix having expansion factor 7

The generator matrix G is given by G = {$A^T$| $I_k$}. It is used to reduce H into row-echelon form by applying elementary row operations. The encoded codeword c is derived from the matrix multiplication c = u x G, where u is the message to be transmitted. The encoding process consists of normal matrix multiplication.

## III. Decoding Algorithm of LDPC Coder

The algorithm used to decode LDPC codes was discovered independently several times and as a matter of fact comes under different names. The most common ones are the belief propagation algorithm, the message passing algorithm and the sum-product algorithm [7]. In order to explain this algorithm, a very simple variant which works with hard decision [6], will be introduced first. Later on the algorithm will be extended to work with soft decision which generally leads to better decoding results.

### A. Hard Decision (Bit Flipping) Algorithm

Tanner designed LDPC codes with a graphical representation method which called Tanner graph [4]. Tanner graph can express the decoding process of LDPC codes effectively and completely. For an example of Tanner graph with (8, 4) LDPC code is given in Figure 2 and it can describe a regular parity check matrix H intuitively in Eq. 1 Tanner graph has two different nodes which are the variable nodes (v-node) and the check nodes (c-node). Each row of a parity check matrix H is to match each check node of Tanner graph, and each column of a parity check matrix H is to match each variable node of Tanner graph. Check node i is connected to neighbor variable node j when $h_{ij}$ in H is assigned to 1. Otherwise, it is not connected the value of this element is 0. Therefore, n variable nodes and m = n-k check nodes are existed in a parity check matrix H and are matched to Tanner graph.

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{1}$$

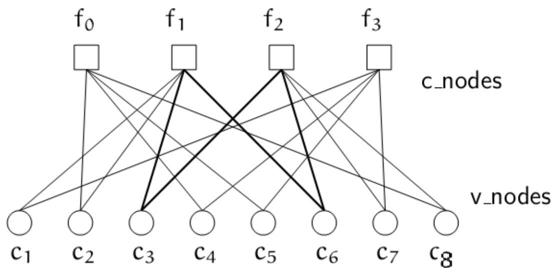

Fig. 2 Graphical representation of tanner graph for LDPC code

An error free codeword of H is c = [1 0 0 1 0 1 0 1]. Suppose we receive y = [1 1 0 1 0 1 0 1]. So $c_2$ was flipped. The algorithm is as follow

Step 1. All v-nodes $c_i$ send a”message” to their c-nodes $f_j$ containing the bit they believe to be the correct one for them. At this stage the only information a v-node $c_i$ is the corresponding received ith bit of c. That means for example, that $c_1$ sends a message containing 1to $f_1$ and $f_3$, node $c_2$ sends messages containing $y_1$ (1) to $f_0$and $f_1$, and so on.

Step 2. Every check nodes calculate a response to their connected message nodes using the messages they receive from step 1. The response message in this case is the value (0 or 1) that the check node believes the message node has based on the information of other message nodes connected to that check node. This response is calculated using the parity-check equations which force all message nodes connect to a particular check node to sum to 0 (mod 2).

Table 1. Check nodes activities for hard-decision Decoder for code of Fig. 2

| check nodes | activities | | | | |
|---|---|---|---|---|---|
| $f_0$ | receive | $c_2 \rightarrow 1$ | $c_4 \rightarrow 1$ | $c_5 \rightarrow 0$ | $c_8 \rightarrow 1$ |
| | send | $0 \rightarrow c_2$ | $0 \rightarrow c_4$ | $1 \rightarrow c_5$ | $0 \rightarrow c_8$ |
| $f_1$ | receive | $c_1 \rightarrow 1$ | $c_2 \rightarrow 1$ | $c_3 \rightarrow 0$ | $c_6 \rightarrow 1$ |
| | send | $0 \rightarrow c_1$ | $0 \rightarrow c_2$ | $1 \rightarrow c_3$ | $0 \rightarrow c_6$ |
| $f_2$ | receive | $c_3 \rightarrow 0$ | $c_6 \rightarrow 1$ | $c_7 \rightarrow 0$ | $c_8 \rightarrow 1$ |
| | send | $0 \rightarrow c_3$ | $1 \rightarrow c_6$ | $0 \rightarrow c_7$ | $1 \rightarrow c_8$ |
| $f_3$ | receive | $c_1 \rightarrow 1$ | $c_4 \rightarrow 1$ | $c_5 \rightarrow 0$ | $c_7 \rightarrow 0$ |
| | send | $1 \rightarrow c_1$ | $1 \rightarrow c_4$ | $0 \rightarrow c_5$ | $0 \rightarrow c_7$ |

In Table 1, check node $f_1$ receives 1 from $c_4$, 0 from $c_5$, 1 from $c_8$ thus it believes $c_2$ has 0 (1+0+1+0=0), and sends that information back to $c_2$. Similarly, it receives 1 from $c_2$, 1 from $c_4$, 1 from $c_8$ thus it believes $c_5$ has 1 (1+1+1+1=0), and sends 1 back to $c_5$. At this point, if all the equations at all check nodes are satisfied, meaning the values that the check nodes calculate match the values that receive, the algorithm terminates. If not, we move on to step 3.

Table 2. Message nodes decisions for hard decision Decoder for code of Fig. 2

| message nodes | $y_i$ | messages from check nodes | | decision |
|---|---|---|---|---|
| $c_1$ | 1 | $f_2 \rightarrow 0$ | $f_4 \rightarrow 1$ | 1 |
| $c_2$ | 1 | $f_1 \rightarrow 0$ | $f_2 \rightarrow 0$ | 0 |
| $c_3$ | 0 | $f_2 \rightarrow 1$ | $f_3 \rightarrow 0$ | 0 |
| $c_4$ | 1 | $f_1 \rightarrow 0$ | $f_4 \rightarrow 1$ | 1 |
| $c_5$ | 0 | $f_1 \rightarrow 1$ | $f_4 \rightarrow 0$ | 0 |
| $c_6$ | 1 | $f_2 \rightarrow 0$ | $f_3 \rightarrow 1$ | 1 |
| $c_7$ | 0 | $f_3 \rightarrow 0$ | $f_4 \rightarrow 0$ | 0 |
| $c_8$ | 1 | $f_1 \rightarrow 1$ | $f_3 \rightarrow 1$ | 1 |

Step 3. In this step, the message nodes use the messages they get from the check nodes to decide if the bit at their position is a 0 or a 1 by majority rule. The message nodes then send this hard-decision to their connected check nodes. Table 2 illustrates this step. To make it clear, let us look at message node $c_2$. It receives 2 0's from check nodes $f_1$ and $f_2$. Together with what it already has $y_2 = 1$, it decides that its real value is 0. It then sends this information back to check nodes $f_1$ and $f_2$.

Step 4. Repeat step 2 until either exit at step 2 or a certain number of iterations has been passed. In this example, the algorithm terminates right after the first iteration as all parity check equations have been satisfied. $c_2$ is corrected to 0.

### *A.* Soft-Decision Algorithm

The above description of hard-decision decoding was mainly for educational purpose to get an overview about the idea. Soft-decision decoding of LDPC codes [7], which is based on the concept of belief propagation, yields in a better decoding performance and is therefore belief propagation the preferred method.

The soft-decision decoder operates with the same principle as the hard-decision decoder, except that the messages are the conditional probability that the received bit is a 1 or a 0 given the received vector y.

let $P_i = P_r(c_i = 1|y_i)$ be the conditional probability that $c_i$ is a 1 given the value of y and let $Pr(c_i = 1|y_i) = 1- P_i$. $q_{ij}$ is a message sent by the variable node $c_i$ to the check node $f_j$. Every message contains always the pair $q_{ij}(0)$ and $q_{ij}$ (1) which stands for the amount of belief that $y_i$ is a"0" or a "1".

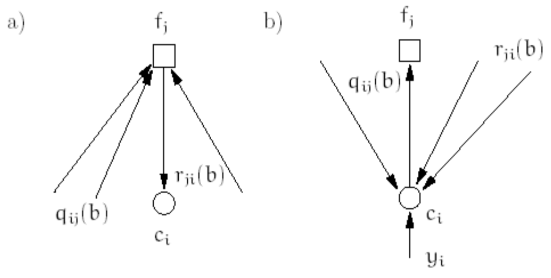

Fig. 3 VN Decoder and CN decoder

$r_{ji}$ is a message sent by the check node $f_j$ to the variable node $c_i$. Again there is a $r_{ji}(0)$ and $r_{ji}(1)$ that indicates the (current)amount of believe in that $y_i$ is a ”0” or a ”1”.The step numbers in the following description correspond to the hard decision case.

Step 1. All variable nodes send their $q_{ij}$ messages to check node from below equation

$$q_{ij}(1) = P_i \text{ and } q_{ij}(0) = 1 - P_i \tag{2}$$

Step 2. The check nodes calculate their response messages $r_{ji}$

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{i'\in vj/i}(1 - 2q_{i'j}(1)) \tag{3}$$

$$r_{ji}(1) = 1 - r_{ji}(0) \tag{4}$$

So they calculate the probability that there is an even number of 1’s among the variable nodes except $c_i$ (this is exactly what $V_{j\backslash i}$ means). This probability is equal to the probability $r_{ji}(0)$that $c_i$ is a 0. This step and the information used to calculate the responses is illustrated in Fig. 3.

Step 3. The variable nodes update their response messages to the check node ci to the variable node fj. This is done according to the following equations

$$q_{ij}(0) = k_{ij}(1 - Pi)\prod_{j'\in ci/j} r_{j'i}(0) \tag{5}$$

$$q_{ij}(1) = k_{ij}P_i \prod_{j'\in ci/j} r_{j'i}(0) \tag{6}$$

Where by the Constants $K_{ij}$ are chosen in a way to ensure that $q_{ij}(0)+q_{ij}(1) = 1$. $C_{i\backslash j}$ now means all check nodes except $f_j$.At this point the v-nodes also update their current estimation their variable ci. This is done by calculating the probabilities for 0 and 1 and voting for the bigger one. The used equations are quite similar to the ones to compute $q_{ij(b)}$ but now the information from every c-node is used.

$$Q_i(0) = K_i(1 - P_i)\prod_{j\in Ci} r_{ji}(0) \tag{7}$$

$$C_i = \begin{cases} 1 & if Q_i(1) > Q_i(0) \\ 0 & else \end{cases} \tag{8}$$

$$Q_i(1) = K_iP_i\prod_{j\in Ci} r_{ji}(1) \tag{9}$$

If the current estimated codeword fulfills now the parity check equations the algorithm terminates. Otherwise termination is ensured through a maximum number of iterations.

## IV. Result and Analysis

In this section, the BER performance for the Hard decision (Bit flipping) and Soft decision (Sum Product) algorithms with irregular (1016,508)LDPC codes with codeword length is 1016 and code rate is 0.5 are presented. The simulations are performed over an additive white Gaussian noise (AWGN) channel with BPSK modulation for different number of iterations.

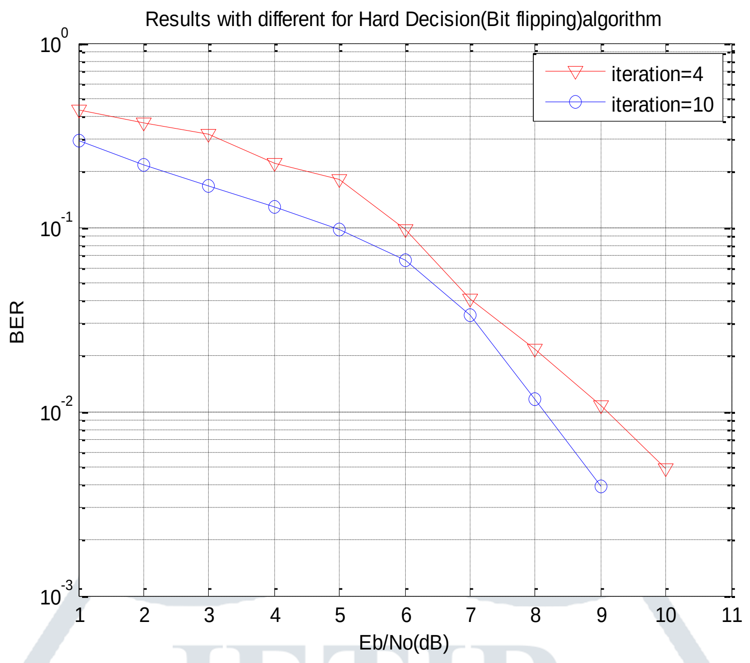

Fig. 4 Simulation result for (508,1016) for bit flipping algorithm

Fig. 4 is shows BER Vs SNR is plotted for the Hard decision (Bit flipping) algorithm for iteration 4 and iteration 10. If number of iteration is increases the BER performance improves at the cost of decoding time. So at iteration 10 BER performances is good as compare with iteration 4.

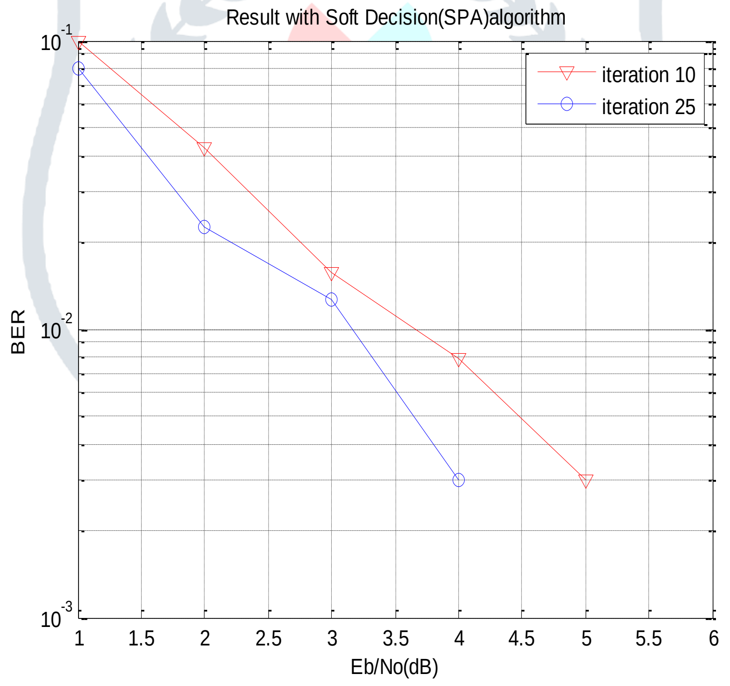

Fig. 5 Simulation result for (508,1016) for Sum Product algorithm

Fig. 5 is shows BER Vs SNR is plotted for the Soft decision (Sum Product) algorithm for iteration 10 and iteration 25. If number of iteration is increases the BER performance improves at the cost of decoding time. So at iteration 25 BER performances is good as compare with iteration 10.

## V. Conclusions

BER Performance for Sum Product Algorithm and Bit Flipping Algorithm is checked with different number of iterations. It is observed that as number of iteration increases the BER performance improves at the cost of decoding time. It can be seen from BER performance that the Sum Product Algorithm is better in comparison with Bit flipping Algorithm.

## VI. Acknowledgment

I am really thankful to my guide without which the accomplishment of the task would have never been possible. I am also thankful to all other helpful people for providing me relevant information and necessary clarifications.

## References

[1] William E. Ryan, Shu Lin, “Channel Codes-Classical and Modern”,Cambridge University Press 2009, ISBN-13 978-0-511-64182-4 eBook (NetLibrary).

[2] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” IEEE Transactions on Information Theory, vol.45, pp. 399–431, Mar.1999.

[3] S. K. Chilappagari, B. Vasi´c, and M. W. Marcellin, “Guaranteed error correction capability of codes on graphs,” in Proceedings of the Information Theory and Applications, UCSD, Feb 2009.

[4] D. Burshtein, “On the error correction of regular LDPC codes using the flipping algorithm,” IEEE Trans. Inf. Theory, vol. 54, no. 2, pp.517–530,Feb. 2008.

[5] T. Richardson and R. Urbanke, “The capacity of low-density parity check codes under messag epassing decoding,” IEEE Trans. Inform. Theory, vol. 47, pp. 599 – 618, may 2001.

[6] Zijian Dong, Tiecheng Song, “New Weighted Bit-Flipping Decoding Algorithm for LDPC Codes”,IEEE International Symposium on Computer, Communication, Control and Automation, pp.384-386,April 2013.

[7] Adrian Voicila, David Declercq, François Verdier, Marc Fossorier, and Pascal Urard. ”Low-Complexity Decoding for Non-Binary LDPC Codes in High Order Fields” IEEE transactions on communications, vol.58, no. 5,pp.1365,1375,may 2010