ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 2, February 2016

Performance Evaluation of Low Density Parity Check codes with Hard and Soft decision Decoding

Shalini Bahel, Jasdeep Singh

***Abstract*— The Low Density Parity Check (LDPC) codes have received a considerable attention since recent years due to their ability to perform near to Shannon's limit. The Decoding of LDPC codes is carried out iteratively. The iterative decoding may further be classified into Hard decision decoding and Soft decision decoding depending upon the type of decision (Hard or Soft) taken by the demodulator. In this paper, performance of LDPC codes is evaluated over AWGN channel based upon both Hard and Soft decision decoding. For Hard decision decoding Bit Flipping Algorithm (BFA) is used and for Soft decision decoding Sum Product Algorithm (SPA) is used. Simulations show that the performance of LDPC codes with Soft decision decoding is better than that with Hard decision decoding however Soft decision decoding takes comparatively longer time for decoding compared to Hard decision decoding.**

***Index Terms*— LDPC, Hard decision, Soft decision, BFA, SPA**

I. INTRODUCTION

The Low-density parity-check (LDPC) codes were purposed for the first time by Gallager [1, 2] in his PhD thesis at MIT in 1962. Initially, they were ignored for a long time because their computational complexity was high for the hardware technology available at that time. But later on the discovery and great success of Turbo codes [3, 4] led to the rediscovery of LDPC codes by McKay and Neal in 1996 [5]. McKay and Neal also showed that similar to Turbo codes, LDPC codes performed near to Shannon's limit [6, 7]. Further research on LDPC codes made them best codes to perform near to Shannon's limit. It was shown by Chug et al. [8] that LDPC have came within 0.0045dB of Shannon's limit, beating the best known Turbo codes at that time. Several important factors such as the capability of parallel decoding, less complexity per iteration in comparison to that of turbo decoders, natural stopping criterion, error floor at much lower Bit Error Rate than turbo codes and the flexibility in the choice of rate and code length, have led them to appear in several standards, such as IEEE 802.16, IEEE 802.11, IEEE 802.3 and DBV-RS2.

Although a LDPC code is defined by a sparse parity check matrix, a bipartite graph also known as *Tanner Graph* [9] can be used to represent the code. A bipartite graph is a graph whose nodes can be divided into two sets such that each node in one set is connected to a node in other set. The connections are represented by lines called as edges of the Tanner Graph. The two sets of nodes in a Tanner Graph are called *check nodes* and *variable nodes* representing rows and columns respectively.

The field of LDPC codes is very vast. In this paper the focus is mainly on decoding of LDPC codes. The decoding is the process of de-mapping the received information sequence at the channel output and using the redundancy in the received sequence to correct errors. The decoding process in case of LDPC codes is based upon passing messages between the variable and the check nodes on the Tanner graph of the code, so called message pass decoding or Iterative decoding. The Iterative decoding can further be classified into two main categories: 1) Hard decision decoding and 2) Soft decision decoding, depending upon the type of decision (Hard or soft) taken by the demodulator in the communication system. To implement Hard and Soft decision decoding, various algorithms have been developed and implemented. Some of those algorithms are belief propagation algorithm, bit flipping algorithm, sum product algorithm, min-sum decoding algorithm etc.

In this paper, for evaluating performance of LDPC codes, Bit Flipping Algorithm (BFA) is used for hard decision decoding and Sum Product Algorithm (SPA) is used for soft decision decoding. The performance is evaluated by simulating a communication channel in the MATLAB.

II. HARD VERSUS SOFT DECISION

In a communication system, the information sequence passed over the channel is received by the demodulator which takes decision on the received information sequence. The decision taken by the demodulator can either be hard-decision or soft decision as explained in previous section. In case of Hard decision a decision regarding a bit equals to 0 or 1 (binary) is taken by the demodulator and the resultant binary sequence is forward towards the decoder which decodes the information sequence. On the other hand, in case of soft decision decoding, the received sequence is not approximated into 0's and 1's but instead the probabilities of received code words are directly forwarded towards the decoder which decodes the sequence depending upon the algorithm used for decoding. The probabilities which are passed towards the decoder by the demodulator are called as *a priori probabilities* and the probabilities returned by the decoder are called as *a posteriori probabilities*.

Figure 1 illustrates the Hard and Soft decision taken by the demodulator.

***First Author name**, Shalini Bahel, Assistant Professor, department of electronics technology, Guru Nanak Dev University, Amritsar (Punjab).*
***Second Author name**, Jasdeep singh, Assistant Professor, department of electronics technology, Guru Nanak Dev University, Amritsar (Punjab).*

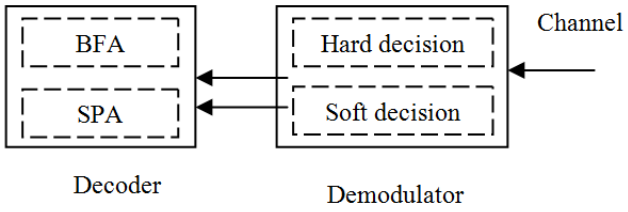

Figure 1 Hard and soft decision of LDPC codes

III. Algorithms for Decoding

The two algorithms Bit Flipping Algorithm (BFA) and Sum Product Algorithm (SPA) are explained below

a) Bit Flipping Algorithm (BFA)

In case of BFA, a binary (hard) decision about each received bit is made by the demodulator and this is passed to the decoder. For the bit-flipping algorithm the messages passed along the Tanner graph edges are also binary: a variable node sends a message declaring if it is a one or a zero, and each check node sends a message to each connected variable node, declaring what value the bit is based on the information available to the check node. The check node determines that its parity-check equation is satisfied if the modulo-2 sum of the incoming bit values is zero. However, if the parity check equations are not satisfied (mod-2 sum is not zero) the variable nodes which participate in maximum number of incorrect parity checks are flipped. This process is repeated until all of the parity-check equations are satisfied, or until some maximum number of decoder iterations has passed and the decoder gives up.

The bit-flipping decoder can be immediately terminated whenever a valid code word has been found by checking if all of the parity-check equations are satisfied.

b) Sum Product Algorithm (SPA)

On the other hand in case of SPA, Log likelihood ratios are passed between variable and check nodes. The variable nodes receive the outputs of the channel and pass the likelihood of the code word components to the check nodes. Each check node updates the received likelihoods using the information it has gathered from all variable nodes that are connected to it and sends updated likelihoods back the check nodes. This process is repeated until a predetermined maximum number of iterations is achieved or until a code word is decoded i.e. all the check equations are satisfied. If the received code word in BPSK format (in the form of -1' and +1) is *y* then the Log likelihood ratio (LLR) $L(x)$ is given by equation

$$L(x) = \log\{p(x=0)/p(x=1)\} \quad (1)$$

Here $p(x = 0)$ is probability of received symbol equal to zero and $p(x = 1)$ is probability of received symbol equal to one. In terms of Bit energy the LLR is given by

$$L(x) = 4\sqrt{E/N_o} \quad (2)$$

Here, E is the energy of the bit so that 0 is mapped to $-\sqrt{E}$ and 1 is mapped to $+\sqrt{E}$. The sum product algorithm is initialized at each message node i, $1 \le i \le n$, by sending the likelihood values $L(x)$ to all the check nodes $j \in M(i)$, where $M(i)$ denotes the check nodes connected to the variable node i. In other words, for all $1 \le i \le n$, and for all $j \in M(i)$, the message passed from node i to node j is

$$M_{i\text{-}j} = (4\sqrt{E/N_o})\, x \quad (3)$$

Also the check nodes j, after receiving all messages from variable nodes connected to it, computes the messages to be sent to node i, $1 <= i <= n$, using the relation given by

$$E_{j,i} = 2\tanh^{-1} \{ \prod_{i' \in B_j, i' \sim= i} \tanh (M_{j,i'} / 2) \} \quad (4)$$

Where, B_j denotes the variable nodes connected to check nodes. After this the variable nodes update their information based on the received information from check nodes. In this step, variable node i sends check node $j \in M(i)$ the updated loglikelihood given by

$$M_{j,i} = LLR(P^{int}) = \sum_{j' \in A_i, j' \sim=j} E_{j',i} + r_i \quad (5)$$

where r_i denotes input probabilities given by

$$r_i = \log \{ (p(c_t = 0) / p(c_t = 1) \}$$

Then finally the codeword is detected using

$$z_i = 1 \text{ for } L_i \le 0 \text{ and}$$
$$z_i = 0 \text{ for } L_i > 0 \quad (6)$$

In this way the code word is decoded using SPA.

IV. Implementation of LDPC codes

For implementation of LDPC codes, a parity check matrix of size 1000 x 2000 is created using Mackay-Neal's method. In this paper instead of Encoding every time using generator matrix, an initial message consisting entirely of zeroes is used, which was also used by Luby [10-12]. Also it was proved [13] that if the symmetry conditions are satisfied by the channel then the Probability of error is independent of the code word. Hence an all zero code word can be used. The modulation technique used here is BPSK. The channel used in the simulations is AWGN. The channel is realized in MATLAB by creating random noise and adding it to the all zero sequence in the channel. The decoding is performed using BFA and SPA with different number of iterations. For simulation, 1000 frames were sent and Bit Error Rate (BER), Frame Error Rate (FER) and average time taken for decoding a frame of LDPC codes was calculated. The various results obtained are discussed in the next section.

V. Results and Discussion

Figure 2(a) and 2(b) shows the BER (Bit Error Rate) and FER (Frame error Rate) of LDPC codes with BFA algorithm. It is clear from the Figures 2(a) and 2(b) that with increase in number of iterations the performance of LDPC codes improves. The dotted line shows the performance of uncoded data and solid lines are for coded data with different number of iterations as indicated by the legend. It is clear from the diagrams that with one iteration both BER and FER performances are poor than uncoded data but as the number of iterations are increased both BER and FER performances improves. Also as the iterations are increased from 10 to 20 the improvement in performance is very small than as compared with previous improvements with small number of iterations. This could be due to the reason that most of the errors have been corrected with initial iterations and only a small number of errors are left. Also at very low E_b/N_0 values performance of coded data is poor than uncoded data but as E_b/N_0 increases the performance improves over uncoded data. Figure 2(c) shows the Average time taken for decoding a frame using BFA algorithm. It is observed from the Figure that, at low values of E_b/N_0 the time taken for decoding a frame increases almost linearly with increase in number of iterations but at higher values of E_b/N_0 the Time taken for decoding do not increase linearly with increase in number of iterations. It might be due to the reason that at low value of E_b/N_0 there are comparatively large number of errors which could have occurred in the frames. These errors would require large number of iterations for getting corrected and every specified iteration would consume its normal time for rectifying errors. Due to this the time taken increases with increase in number of iterations at low E_b/N_0. On the other hand, at high values of E_b/N_0 the number of errors which could have occurred in the frames is comparatively small in number. These errors could be rectified with smaller number of iterations, taking their normal time for decoding. But if more number of iterations is specified, the higher iterations do not take part in correcting errors, because almost all the errors would have been corrected with lower iterations. Due to this time is only consumed by lower iterations and hence the slope of the curve falls for higher iterations at high values of E_b/N_0.

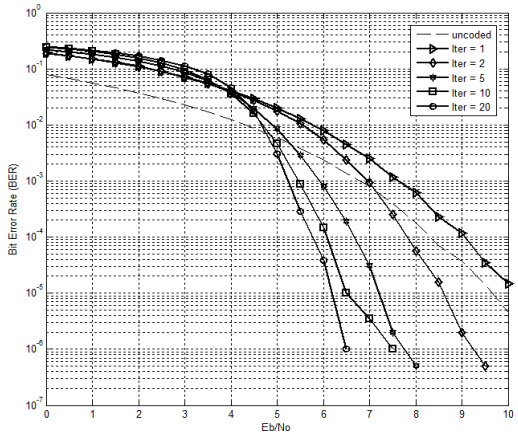

Figure 2(a) BER performances of LDPC codes with different number of iterations using BFA

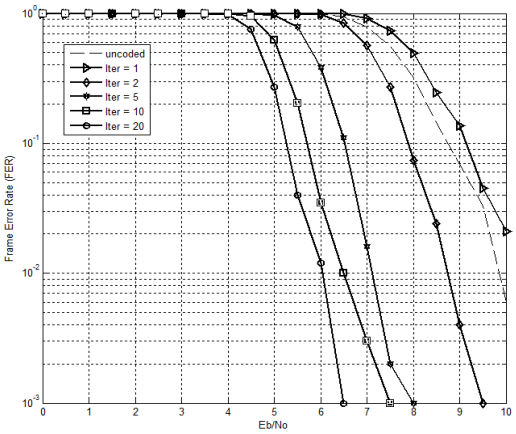

Figure 2(b) FER performances of LDPC codes with different number of iterations using BFA

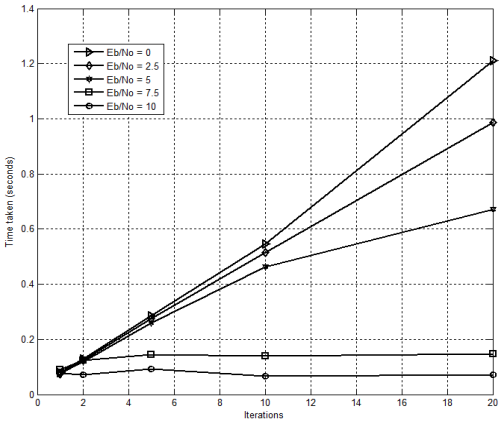

Figure 2(c) Average Time taken for decoding LDPC frame with different number of iterations using BFA

Figure 3(a) and 3(b) shows the BER and FER for SPA algorithm. From Figures it is clear that similar to BFA, the performance improves with increase in number of iterations however the improvement in this case is more than that with BFA.

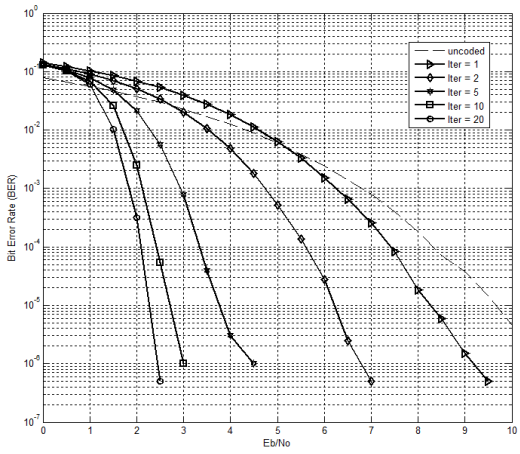

Figure 3(a) BER performances of LDPC codes with different number of iterations using SPA

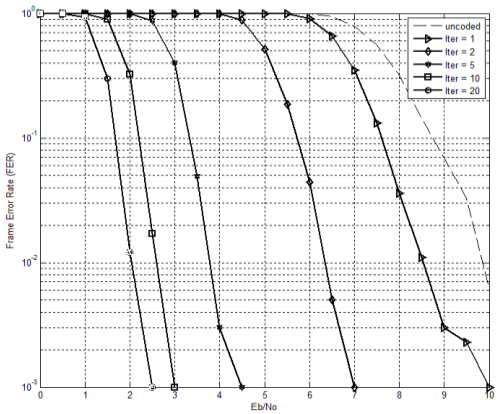

Figure 3(b) FER performances of LDPC codes with different number of iterations using SPA

Figure 3(c) shows the average time taken by SPA algorithm for decoding a frame of LDPC codes. It is clear from the Figure that similar to BFA at low E_b/N_0 the average time taken increases linearly with number of iterations and at higher E_b/N_0 average the time taken do not increase linearly with iterations but the curve bends on higher iterations.

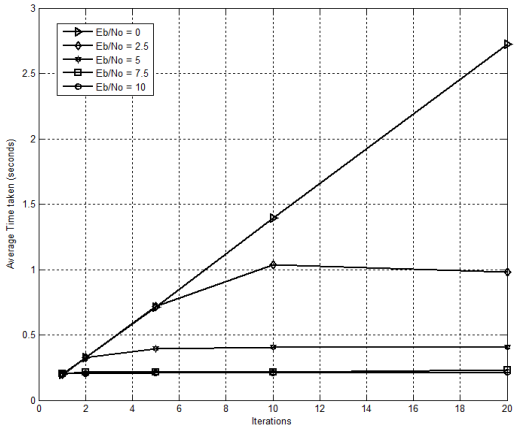

Figure 3(c) Average Time taken for decoding LDPC codes with different number of iterations using SPA

For comparing the BFA and SPA algorithm, the BER performance and Average time taken for decoding are compared for both the algorithms.

Table 1 shows the values of BER performances for the two algorithms at various values of E_b/N_0 and Figure 4 shows the comparison of performance using the two algorithms. For comparing BER performance, the numbers of iterations for decoding are fixed to five. From Table and Figure, it is clear that the SPA algorithm performs much better than that with BFA algorithm i.e. SPA comparatively corrects more number of errors than that of BFA at a particular value of E_b/N_0. The reason behind this could be for this is that SPA is based upon Soft decision decoding whereas BFA is based upon Hard decision decoding.

Table 1 BER of BFA and SPA with five decoding iteration

E_b/N_0	Bit Error Rate (BER)	
	BFA (Iter = 5)	SPA (Iter = 5)
0	2.2126e-001	1.2905e-001
2.5	1.1258e-001	5.6290e-003
5	8.5419e-003	6.3860e-003
7.5	2.0000e-006	—

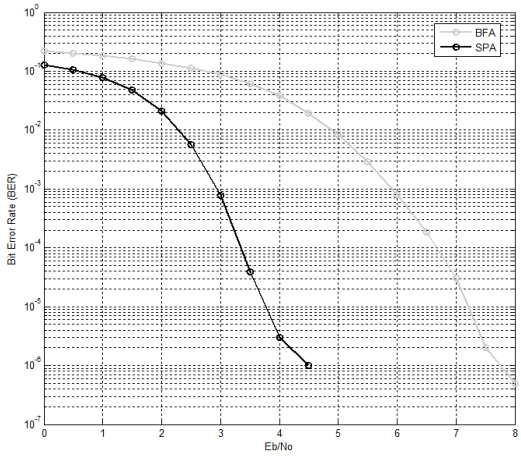

Figure 4 BER performance comparison of BFA and SPA with five decoding iteration

Table 2 and 3 shows the average time taken for decoding for the BFA and SPA respectively at various values of E_b/N_0 with different iterations and Figure 5 shows a comparison of the two algorithms based upon Average time taken for decoding. It is observed that the SPA algorithm takes comparatively longer time for decoding compared with BFA algorithm. This is due to the reason that in case of SPA, the functioning of Decoder is comparatively more complicated than that with BFA, because decoder in this case has to deal with probabilities of information than the binary symbols (i.e. 0 or 1) and working with probabilities is comparatively more time consuming. That is why the SPA consumes more time for decoding than BFA with different number of iterations. Also it is clear from the Figure that in case of SPA the curve bends more sharply than that of BFA, due to superior performance of SPA compared with BFA.

Table 2 Average time taken for decoding using BFA

E_b/N_0	Average time taken for decoding with BFA				
	Iter = 1	Iter = 2	Iter = 5	Iter = 10	Iter = 20
0	0.0802	0.1293	0.2842	0.5452	1.2122
2.5	0.0739	0.1237	0.2759	0.5154	0.9860
5	0.0776	0.1199	0.2598	0.4621	0.6717
7.5	0.0895	0.1241	0.1446	0.1393	0.1481

Table 3 Average time taken for decoding using SPA

E_b/N_0	Average time taken for decoding with BFA				
	Iter = 1	Iter = 2	Iter = 5	Iter = 10	Iter = 20
0	0.1956	0.3249	0.7126	1.3926	2.7262
2.5	0.1963	0.3252	0.7191	1.0354	0.9790
5	0.1933	0.3259	0.3938	0.4031	0.4024
7.5	0.2015	0.2154	0.2143	0.2134	0.2307

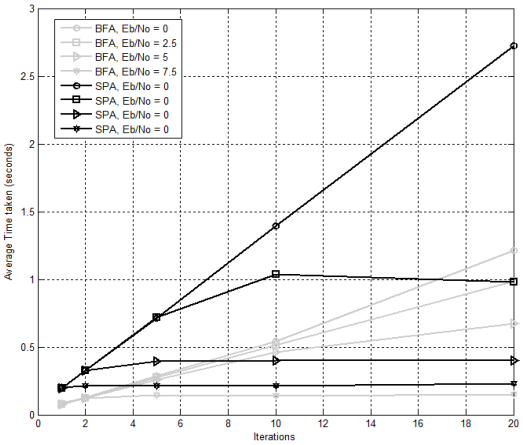

Figure 5 Comparison of Average time taken for decoding using BFA and SPA

VI. Conclusion

Performance of a LDPC based communication system with BPSK modulation and AWGN channel is evaluated using MATLAB. It has been concluded from the results that with increase in number of iterations the performance is improved in both the cases of BFA and SPA. However the performance using SPA is much better than that with BFA. This could be due to soft decoding in case of SPA compared with hard decoding in case of BFA. Also the average time taken for decoding a frame increases as the number of iterations is increased. However decoding using SPA takes more time compared to decoding using BFA due to complicated functioning of Demodulator and Decoder in case of SPA.

REFERENCES

[1] R. G. Gallager, "*Low-Density Parity-Check Codes",* IRE trans. on Information theory, Vol. 8, issue 1, pp. 21-28, January 1962.
[2] R. G. Gallager, "*Low-Density Parity-Check Codes*," Cambridge, MA: MIT Press, 1963.
[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "*Near Shannon limit error-correcting coding and decoding: Turbo-codes*," in Proc., IEEE lnt. Conf. on Communications (Geneva, Switzerland, May 1993), pp. 1064-1070.
[4] Claude Berrou and Alain Glavieux, "*Near Optimum Error Correcting Coding and Decoding: Turbo-Codes*," IEEE trans. on communication, Vol. 44, No. 10, October 1996.
[5] D. J. C. MacKay, "*Good codes based on very sparse matrices,"* IEEE proceedings on Information Theory, 29 jun-4 jul, 1997.
[6] D. J. C. MacKay and R. M. Neal, "*Near Shannon limit performance of low density parity check codes,"* IEEE electronic letters, vol. 32, No. 18, 29 August, 1996.
[7] D. J. C. MacKay and R. M. Neal, "*Near Shannon limit performance of low density parity check codes,"* IEEE electronic letters, vol. 33, No. 6, 13 March, 1997.
[8] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. L. Urbanke, "*On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit,*" IEEE Communication Letters, vol. 5, no. 2, pp. 58–60, February 2001.
[9] R. M. Tanner, "*A recursive approach to low complexity codes*," IEEE Trans. Inform. Theory, vol. 27, no. 5, pp. 533–547, September 1981.
[10] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, "*Improved low-density parity-check codes using irregular graphs and belief propagation,"* In proceedings of IEEE int. symp. on information theory, August 1998.
[11] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, "*Improved low-density parity-check codes using irregular graphs and belief propagation,"* SRC Technical Note, 1998 - 009, April 15, 1998.
[12] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, "*Improved low-density parity-check codes using irregular graphs,"* IEEE Trans. on Inform. Theory, Vol. 47, No. 2, pp. 585-598, February 2001.
[13] Thomas J. Richardson and Rüdiger L. Urbanke, "*The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding*," IEEE Trans. on Information theory, Vol. 47, No. 2, pp. 599-618, Feb. 2001.

Authors Profile

Dr. shalini Bahel

Shalini bahel is working as Assistant Professor in department of Electronics Technology at Guru Nanak Dev University, Amritsar. She did her B.E. (Electronics) from GEC (presently UIT, RGPV) Bhopal, M.Tech. (Digital Communication) from MACT (presently MANIT) Bhopal and PhD from GNDU Amritsar. She has published many papers in international and national conferences/journals.

Er. Jasdeep Singh

Jasdeep Singh bahel is working as Assistant Professor in department of Electronics Technology at Guru Nanak Dev University, Amritsar. He did his B.Tech (Electronics and Communication) from ACET amritsar and M.tech (Communication Systems) from GNDU Amritsar.