BER Performance Analysis of LDPC Codes for Min-Sum Decoding Algorithm

Rinku N. Patel1 Manish I. Patel2
1,2Department of Electronics & Communication
1,2SPCE, Visnagar-384315, India

Abstract—Low density parity check (LDPC) codes are adopted in many applications due to their Shannon limit approaching error correcting performance. In this paper, we analyze the Bit Error Ratio (BER) performance of LDPC codes with different codeword length and different number of iterations over Additive White Gaussian Noise (AWGN) channel. The encoding of LDPC codes is done with standard method and it is implemented using MATLAB. Codewords have been constructed for code rate 1/2. LDPC codes can be decoded using iterative decoding algorithm to improve the performance. Here, the min-sum decoding algorithm is implemented and desired simulation results were obtained using MATLAB. The min-sum algorithm is mathematically equivalent to the sum-product algorithm and it greatly reduces the computational complexity of sum-product algorithm. The simulation results show that higher the number of iterations, better the code's performance.

Key words: LDPC, Iterative decoding, MSA

I. INTRODUCTION

Low density parity check codes are first discovered by Gallager in 1963 in his PhD thesis[1]. They were unnoticed for long time by researchers because their computational complexity was high at that time, LDPC codes are reinvented by Mackay and Neal[2]. As their name suggests, LDPC codes are linear block codes with parity-check matrices that contain only a very small number of non-zero entries, that's why the name is "LOW DENSITY PARITY-CHECK" codes. The sparseness of parity-check matrix H, which guarantees both a decoding complexity which increases linearly with the code length, and a minimum distance which also increases linearly with the code length[3]. In any coding scheme larger block-length codes provide superior performance, but need more computing power[6]. The Performance of any code is measured through its bit error rate (BER) vs. signal to noise ratio (Eb/No) graph in dB. The curve of a good code will show a spectacular drop in BER as the SNR increases. LDPC codes can approach the performance near Shannon limit. That's why LDPC codes have been adopted in many applications such as satellite based digital video broadcasting(DVB), IEEE wireless local area network standard, Optical communication standards, 3rd generation mobile telephony, 10GBase-T Ethernet, IEEE 802.11n (Wi-Fi), 802.16e (Wi-Max) and others[5].

Low density parity check codes use the generator matrix G in their encoder and parity check matrix H in their decoder. The improvement of LDPC codes over other block codes is their error correcting performance. The parity check matrix H and generator matrix G are orthogonal to each other. The parity check matrix H has M rows and N columns, where M represents the check nodes and N represents the variable nodes. This matrix is generated by random construction techniques. The Information bits are based on the check nodes and the codeword bits are based on the variable nodes. The parity check matrix H shown in figure. 1a can be graphically represented by Tanner graph[4]. It is a pictorial way of representing the parity check matrix of LDPC codes. The rows of a parity check matrix are represented by the check nodes, and the columns are represented by variable nodes as shown in fig.1b. The edges are used to show the connection between variable nodes and check nodes. Fig. 1b shows a Tanner graph of parity check matrix H of size 5x6.

\[
H = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Fig. 1a: Example of parity check matrix H of size 5x6

Check nodes

C0 C1 C2 C3 C4

F0 F1 F2 F3 F4 F5

Variable nodes

Fig. 1b: Graphical representation of parity check matrix H

Parity-check matrix H contains only a small number of 1's and a lot of 0's. If there is the same no. of 1's in each columns and same no. of 1's in each rows, then the parity check matrix H is called regular parity check matrix. Otherwise the parity check matrix is called irregular. The formation of the parity check matrix is important for BER performance. The biggest dissimilarity between LDPC codes and other classical block codes is their decoding techniques. The Classical block codes are generally decoded with Maximum Likelihood decoding algorithms and so they are frequently short and intended to make their task less complex. LDPC codes are decoded iteratively by using a graphical representation of their parity-check matrix H and therefore they are designed with the properties of H matrix.

This paper is organized as follows. Section 2 represents the LDPC algorithm. The proposed LDPC encoder and min-sum decoding algorithm with design methodology are presented in section 3. Section 4 presents the corresponding simulation results in MATLAB. Finally, conclusions and future work are drawn in section 5 and 6 respectively.
II. LDPC ALGORITHM

A codeword $C$ is generated from generator matrix $G$ as,

$$C= k \cdot G$$  \hspace{1cm} (1)

where, $k$= information bits vector, $G$=generator matrix.

This codeword $C$ can be valid if only if it satisfy following condition,

$$C \cdot H^T = 0$$ \hspace{1cm} (2)

where, $H$= low density parity check matrix.

If above equation is not satisfy then the codeword is not valid. And it generate a non-zero syndrom vector $S$.

$$Y \cdot H^T = S$$ \hspace{1cm} (3)

where, $Y$ is invalid codeword. Syndrom vector $S$ is used in decoding process for error correction. The syndrome vector $S$ indicates which row in the $H$ is not zeroed by vector $Y$ and some bits have to be repaired in the decoder$^{[9]}$.

III. LDPC ENCODING AND ITERATIVE DECODING ALGORITHM

A. LDPC Encoder Design:

We have designed LDPC encoder using standard method. The input information bits are applied to encoder and it converts them into codeword using generator matrix $G$ as per equation(4)$^{[10]}$. The generator matrix is generated using parity check matrix $H$. It adds some parity bits in information bits and generate specific codeword. So, the main part of our encoder design is to construct the appropriate parity check matrix $H$$^{[7]}$.

1) Construction of Parity Check Matrix $H$:

1) Step- 1: Generate a matrix of size $M \times N$ which contains all zero elements in it.
2) Step- 2: In matrix $H$, flip the no. of bits in each column (with column weight $W_c$) to arrange them in distinct rows.
3) Step- 3: Re-arrange the no. of 1’s between rows in matrix $H$ to make row weight $W_r$ as uniform as possible.
4) Step- 4: If any row contains 1’s less than two then add one 1’s in it, because the code bit will be zero if only one 1’s in any row.
5) Step- 5: Re-arrange the columns in such a way that no two row of each column in matrix $H$ are overlapped.

2) Construction Of Codeword:

1) Step- 1: Re-arrange the parity check matrix into systematic form.
2) Step- 2: Generate the Generator matrix $G$ from the re-arranged Parity check matrix $H$.
3) Step- 3: After getting generator matrix $G$, multiply it with message bits to generate encoded codeword by performing modulo-2 operations.

$$C= k \cdot G$$  \hspace{1cm} (4)

The relation between generator matrix $G$ and parity check matrix $H$ is defined as below:

$$G \cdot H^T = 0$$

The generated codeword is modulated using BPSK modulation and it converts the codeword $C = (c_1, c_2, c_3, ..., c_n)$ into sequence $s = (s_1, s_2, s_3, ..., s_n)$ according to the equation $s_n = 2c_n-1$, for $n = 1, 2, ..., N$, which maps the binary input $[0,1] \rightarrow [-1,1]$$^{[11]}$. This generated sequence is then transmitted over AWGN channel where noise gets added in it.

B. LDPC Decoder Design:

Now at the receiver side, the received sequence corresponding to $s_n$ after de-modulation is, $Y_n = S_n + N_n$, where $N_n$ is a random gaussian noise with zero mean and variance $\sigma^2$. Then the de-modulated signal is applied to LDPC decoder. Decoder is very important part of any error correcting system. Here LDPC decoder decodes the received signal and gets back the original information signal. There are many decoding algorithms available for LDPC codes but the sum-product algorithm has best error correcting performance but it has high complexity and computations so it is difficult for hardware implementation. So we have used the min-sum algorithm which is originally approximation of sum-product algorithm. The min-sum algorithm can be easily implemented and greatly reduces the decoder complexity and computations but it has little performance degradation$^{[12]}$.

1) Min-Sum Decoding Algorithm:

For ease of later use, let us define some notations as follows,

$L(P_n) = $ The log-likelihood ratio(LLR) of bit $n$ which is obtained from received value of $Y_n$

$L(F_{m,n}) = $ The LLR of bit which is sent from check node $m$ to variable node $n$.

$L(F_{n,m}) = $ The LLR of bit which is sent from variable node $n$ to check node $m$.

$L(P_n) = $ The posteriori LLR of bit $n$ computed at each iteration.

![Flow-chart of min-sum decoding algorithm](image)

Fig. 2: Flow-chart of min-sum decoding algorithm

1) Step - 1 Initialization

A priori information $L(P_n)$ is the LLR of bit $n$ which is obtained from the received value of $Y_n$, and used for variable node initialization.

$$L(F_{n,m}) = L(P_n)$$

Variable to check node initialization:

where,
BER Performance Analysis of LDPC Codes for Min-Sum Decoding Algorithm

(IJSRD/Vol. 3 Issue 03/2015/154)

All rights reserved by www.ijsrd.com

643

2) Step - 2 Iterative Decoding

During this decoding process, each iteration consists of updating and transferring the extrinsic messages between neighbouring variable nodes and check nodes.

C. Horizontal Step (Check Node Update Processing):

\[ L(F_m, n) = 2 \tanh^{-1} \left( \prod_{n \in R(m)/n} \tanh(F_n, m)/2 \right) \]

It can be represented as,

\[ L(F_m, n) = 2 \tanh^{-1} \left( \prod_{n \in R(m)/n} \text{sign}(F_n, m) \tanh(F_n, m)/2 \right) \]

Where, \( \prod_{n \in R(m)/n} \text{sign}(F_n, m) \) represents the sign of all messages coming from variable nodes to check nodes.

So, the equation (5) becomes as,

\[ L(F_m, n) = \prod_{n \in R(m)/n} \text{sign}(F_n, m) \tanh(F_n, m)/2 \]

Here the operations of \( \tanh \) and \( \tanh^{-1} \) are too complex for hardware implementation. So the min-sum algorithm make an inventive approximation to simplify the calculations of check node updates.

\[ \min_{n \in R(m)/n} |F_n, m| = 2 \tanh^{-1} \left( \prod_{n \in R(m)/n} \text{sign}(F_n, m) \tanh(F_n, m)/2 \right) \]

D. Vertical Step (Variable Node Update Processing):

A posteriori information is given as,

\[ L(\bar{P}_n) = L(P_n) + \sum_{m \in M(n)/n} L(F_m, n) \]

So the variable node update equation is given as,

\[ L(F_m, n) = L(\bar{P}_n) + \sum_{m \in M(n)/n} L(F_m, n) \]

3) Step - 3 Hard Decision

As per equation (3), the \( L(\bar{P}_n) \) can take two values as per mapping (0 as -1 and 1 as +1)

\[ \hat{C}_n = \begin{cases} 0, & L(\bar{P}_n) \leq 0 \\ 1, & L(\bar{P}_n) > 0 \end{cases} \]

Now, if \( \hat{C}_n = 0 \) then \( \hat{C}_n \) is the valid codeword.

If above condition does not satisfy then it will take the next iteration and continue the process until max. no of iterations has reached. But if the max. no. of iterations has reached then the codeword is declared invalid and the process stops. This min - sum algorithm greatly reduce the decoder complexity and computations but it has little performance loss[8].

IV. SIMULATION RESULTS

We have implemented LDPC encoder and min-sum decoder in MATLAB. The encoding is done by standard method which generates the codeword with code rate 1/2. Min-sum decoding algorithm is used for LDPC decoder in which messages are transferred between check nodes and variable nodes as per the value of no. of iterations.

Min-sum decoding algorithm is the approximation of sum-product algorithm which has advantages in terms of implementation and computational complexity.

Fig. 3: BER vs. Eb/No. for 2 MSA decoding iterations

We have estimated Bit Error Rate (BER) versus Signal-to-Noise Ratio (Eb/No) graph for different codeword length of 64 bits, 128 bits, and 256 bits. Here, simulation results for codeword length of 128 bits is presented. Fig. 3 shows the simulation result for iteration 2. Fig. 4 shows the simulation result for iteration 5. Fig. 5 and Fig. 6 show the simulation results for iteration 10 and 20 respectively.

Our simulation results show the impact of no. of decoding iterations where it is seen that as we increase the no. of decoding iterations, the BER performance of LDPC codes is improved. The simulation results indicated that when the BER was 0.001, the SNR under AWGN environment was 4.6 dB for 2 iterations and the SNRs were reduced to 3.4 dB, 2.75 dB and 2.28 dB for iterations 5, 10 and 20 respectively.

Fig. 4: BER vs. Eb/No. for 5 MSA decoding iterations
BER Performance Analysis of LDPC Codes for Min-Sum Decoding Algorithm
(IJSRD/Vol. 3/Issue 03/2015/154)

V. CONCLUSION

In this paper, we have implemented low density parity check (LDPC) codes in MATLAB. The LDPC encoder is designed with code rate 1/2. The min-sum decoding algorithm is used for decoding which greatly reduces the complexity and computations of sum-product algorithm. The simulation results for different codeword lengths and different no. of iterations show that as we increase the codeword length and no. of iterations, better performance is achieved in terms of BER.

VI. FUTURE WORK

In future, we will modify the min-sum decoding algorithm by introducing error optimization factor $\alpha$ at the both check node update processing and variable node update processing unit and try to improve the performance of LDPC min-sum decoder.

REFERENCES


