TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 16, No. 1, October 2015, pp. 111 ~ 130
DOI: 10.11591/telkomnika.v16i1.8626

Comprehensive Algorithmic Review and Analysis of LDPC Codes

Waheed Ullah*[1], Abid Yahya[2]
[1] Nanjing University of Aeronautics and Astronautics, Nanjing China
[2] Universiti Malaysia Perlis (UniMAP), Malaysia
*Corresponding author,e-mail: uet_waheed@yahoo.com

Abstract

Due to the increasing popularity of LDPC codes and its demand for future applications, first time in this paper, LDPC coding techniques have been systematically summarized and analyzed. The paper gives the comprehensive review of LDPC encoder, decoder and its architecture for simulation and implementation. The paper is specially intended for giving an insight of the algorithmic overview of the LDPC encoder, decoder and its architecture for research and practical purposes. The original belief propagation algorithm (BPA), logarithmic model of BPA, and the other simplified form of the logarithmic sum product algorithms (SPA) has been elaborated and analyzed for medium and short length codes under AWGN channel.

Keywords: *LDPC Sum Product Algorithms (SPA), Layered LDPC Decoding, LDPC Encoding, Belief Propagation Algorithm, LowDensity Parity Check (LDPC) Codes Decoding.*

1. General Introduction to LDPC Codes

Low density Parity check (LDPC) codes, also known asGallager codes are a type of linear block codes , first proposed by Gallager[1] and were scarcely considered in the three decades that followed due to its computational complexity and limited computational ability of the receiver at that time. LDPC codes were reinvented by Mackay and Neal[2, 3] and have taken considerable attention recently due to their Shannon limits performance[2, 4, 5] with belief propagation decoding algorithm. Before Mackay and Neal , a notable work was done by Tanner[6] in which Tanner generalized LDPC codes and introduced a graphical representation of LDPC codes and now called Tanner graph.

1.1. Generator Matrix

In general, a generator matrix [7] for $k \times n$ array is defined as:

$$G = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_k \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1n} \\ V_{21} & V_{22} & \cdots & V_{2n} \\ \vdots & \vdots & & \vdots \\ V_{k1} & V_{k2} & \cdots & V_{kn} \end{bmatrix} \quad (1)$$

Code vectors, by convention, are usually designated as row vectors. For a message s, a sequence of k message bits is shown below as row vector ($1 \times k$ matrix having one row and k columns):

$$S = S_1, S_2, \ldots\ldots\ldots S_k \quad (2)$$

The codeword C is generated as the matrix product s and G andis written as:

$$C = sG \quad (3)$$

Received July 2, 2015; Revised September 6, 2015; Accepted September 21, 2015

A systematic linear block will have the generator matrix in the following form:

$$G = \left[P \vdots I_k \right] \tag{4}$$

Where I_k is a $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix. The codeword generated by the systematic generator matrix can be simply divided into the two parts of *k* message bits and $(n-k)$ parity check bits.

1.2. Parity Check Matrix

The parity check matrix is used to decode the receive sequence. For each $(n \times k)$ generator matrix *G*, there exists an $(n \times k) \times n$ matrix H such that each row of *G* is orthogonal to the rows of *H*; i.e. $GH^T = 0$ where H^T is transpose of *H* and 0 is a $k \times (n-k)$ all zero matrix. *H* is called the parity check matrix and for systematic linear block code is written in form:

$$H = \left[I_{n-k} \ \vdots \ P^T \right] \tag{5}$$

H is a parity check matrix because it can be used to test if the codeword is valid or not codeword is valid only if $H.code^T = 0$

1.3. Parity Check Matrix and Tanner Graph for LDPC Codes

A low density parity check codes are defined by parity check matrix that is sparse[1, 3]. LDPC code can be denoted in general as (N, d_v, d_c) where N is the length of the code equal to the number of the column in a parity check matrix, d_v is the number of ones(1s) in a column of a parity check matrix and d_c is the number ones(1s) is a row and a parity check matrix. LDPC codes can be regular or irregular. If the number of ones (1s) in each row and column of a parity check matrix are the same, then it is called regular and if the number of ones (1s) in row or column are not the same then it is called irregular. The restriction that $d_v < d_c$ is needed to ensure more than just all-zero codeword satisfies all of the constraints or equivalently, to ensure that a nonzero code rate. For regular parity check matrix or regular LDPC code, the number of ones in *H* satisfies the following condition:

$$M.d_c = N.d_v \tag{6}$$

Where *M* and *N* are the number of row and columns of a parity check matrix respectively. The code rate is given as:

$$r = 1 - \frac{M}{N} = \frac{N - M}{N} \tag{7}$$

Or equivalently,

$$r = 1 - d_v / d_c = d_c - d_v / d_c \tag{8}$$

Here *M* rows are assumed to be linearly independent and $d_v < d_c$ .Also for best performance, the number of ones (1s) in a column is set as $d_v \geq 3$ .The code is valid only if

$$H.code^T = 0 \tag{9}$$

Where *H* is the sparse parity check matrix and code is the codeword obtained from the

generator matrix(*G*) and message bits (s). Consider a parity check matrix *H*, such that $d_v = 2$ and $d_c = 4$

The sparse parity check matrix is best represented by a Tanner graph [8, 9]. The one in each row or column shows the connectivity between variable and check nodes. The set of bit nodes connecting to check nodes m is denoted by $N(m) = \{n \mid h_{mn} = 1\}$ and the set of check nodes connecting to bit node n is given by $M(n) = \{m \mid h_{mn} = 1\}$. A typical Tanner graph is shown in the Figure 1. This graph is for (8, 2, 4) regular LDPC code.The relationship of check node and variable node in the Figure 1 is expressed in algebraic form as shown in equation (10).

The work of the Luby et al [9, 10] have demonstrated that irregular parity check matrices generally outperform their regularcounterparts but it's quite difficult to construct an irregular parity check matrix in case when high girth is required. Also irregular codes are difficult to implement and design hardware for it.

$$H = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix} \quad (10)$$

$$c_1 : 0 = v_2 + v_4 + v_5 + v_8$$
$$c_2 : 0 = v_1 + v_2 + v_3 + v_6$$
$$c_3 : 0 = v_3 + v_6 + v_7 + v_8$$
$$c_4 : 0 = v_1 + v_4 + v_5 + v_7 \quad (11)$$

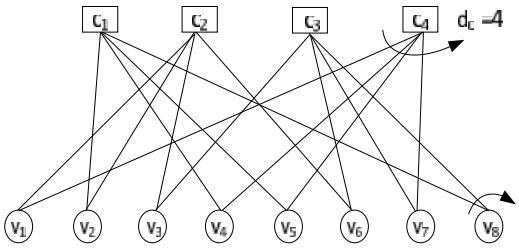


Figure 1. Tanner graph Representation Of H Matrix

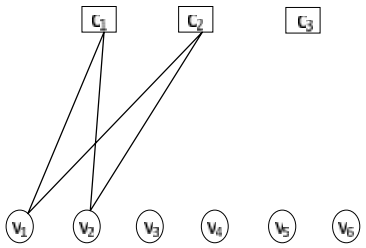


Figure 2. Short Cycles Example

Mackay et al [9] discovered that cycles especially short ones, tended to degrade decoding performance of LDPC codes. Therefore it is of high importance that short cycles be avoided in the construction of good LDPC codes. One example of short cycle (four cycles) is shown in the Figure 2.

2. Encoding Methods of LDPC Codes

The encoding of LDPC codes involves two basic tasks before we transmit the data.

a) Constructing the sparse parity check matrix.
b) b) Generate codewords using that matrix.

The method for generating sparse parity check matrix has explained well by Mackay [11] and Neal[12]and has given a library of codes. The straightforward approaches for encoding the LDPC code are stated as explained below:

2.1. Method 1

This method is used for regular LDPC codes [13, 14]. For a given $m \times n$ sparse parity check matrix for the code x holds the following condition

$$H.x^T = 0 \tag{12}$$

Partitioning the parity check matrix H into $m \times m$ matrix A and $m \times (n-m)$ matrix B, after rearranging columns if necessary to make A nonsingular, we can write the H as:

$$H = [A \mid B] \tag{13}$$

Similarly we partitioned the codeword x into information bits s and parity bits p such that:

$$x = [p \mid s] \tag{14}$$

Now equation (14) becomes:

$$[A \mid B][p \mid s]^T = 0 \tag{15}$$

Thus Equation (15) becomes:

$$Ap + Bs = 0 \tag{16}$$

Hence

$$p = A^{-1}Bs \tag{17}$$

It may be faster to compute c into two steps:

1) Compute $z = (Bs)$ in time proportional to $(n-m)$, exploiting the sparseness of B
2) Compute $p = (A^{-1}z)$, in time proportional to $(n-m)^2$

To exploit the sparseness of A with view to optimize step (2), we find the *LU* decomposition that satisfies:

$$A = LU \tag{18}$$

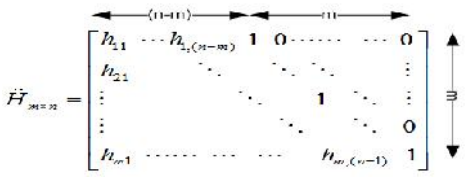
Where L is sparse lower triangular matrix and U is sparse upper triangular matrix.

The previous process reduced the equation $AC = z$ to $Uc = y$. Now Equation (16) becomes:

$$Ly = z \tag{19}$$

$$Up = y \tag{20}$$

We can solve easily Equation (19) and (20) by forward substitution and backward substitution respectively. The pivoting and bit reversing (PABR) algorithm[13] can be used to find the non-singular of binary matrix A over GF(2).



(21)

2.2. Method 2

The Gaussian elimination can transfer the initial sparse parity check matrix $H_{m\times n}$, which is afull rank, into an equivalent lower triangular form $\tilde{H}_{m\times n}$ as shown in Equation (21).

This is the simple way to get the parity bits but this matrix is no longer sparse. This is a pre-processing before encoding the actual data. To encode the data, the codeword is written in form:

$$x = (s, p) \tag{22}$$

Where $s = (s_1, s_2 \ldots\ldots\ldots s_{n-m})$ the information is bits and $p = (p_1, p_2 \ldots\ldots\ldots p_m)$ represents the parity bits which we get by back substitution when the column of the original matrix is shuffled for Gaussian elimination. The parity bits can be obtained in two steps:

a) Initialize

$$p_1 = \sum_{j=1}^{n-m} h_{1,j} s_j \tag{23}$$

b) The next $p_2, p_3 \ldots\ldots\ldots p_m$ parity bits are computed iteratively using the following equation.

$$p_l = \sum_{j=1}^{n-m} h_{l,j} s_j + \sum_{j=1}^{l-1} h_{l,j+(n-m)} p_j \tag{24}$$

Where $l = 2, 3 \ldots\ldots m$. This scheme offers high encoding complexity but is used in simulation.

2.3. Method 3

T.J.Richardson and Urbanke [15] proposed a relatively low complexity efficient encoding [15] scheme. Instead of Gaussian elimination which resulted in a dense lower triangular matrix, the matrix $H_{m\times n}$ is brought to approximate lower triangular matrix $\tilde{H}_{m\times n}$ by row and column shifting only. The Richardson proposed approximate lower triangular matrix is composed of small sub-matrices as:

$$\tilde{H}_{m\times n} = \begin{bmatrix} A & B & T & \begin{matrix}1 & 0\\ & 1\end{matrix} \\ C & D & E \end{bmatrix} \tag{25}$$

Where the dimension of the each matrix is as follow:

$A \rightarrow (m - g) \times (n - m)$
$B \rightarrow (m - g) \times g$
$T \rightarrow (m - g) \times (m - g)$; the lower triangular matrix
$C \rightarrow g \times (n - m)$
$D \rightarrow g \times g$
$E \rightarrow g \times (m - g)$

The matrix $\tilde{H}_{m\times n}$ is obtained by the column shifting of the original matrix $H_{m\times n}$. The columns are selected in such order as to give the reasonable depth of matrix E because the

number of columns of A and C depends on $n-m$, so the only variable is the dimension g which is kept as small as possible to reduce the complexity and make the encoding efficient. Thecodeword $x=(s, p_1, p_2)$ is generated in the following way:

$$p_1^T = -U^{-1}(-ET^{-1}A+C)s^T \quad (26)$$

$$p_2^T = -T^{-1}(As^T + Bp_1^T) \quad (27)$$

Where $U=-ET^{-}B+D$ and must be invertible. Once we get the triangular matrix T, then we can interchange the columns of D to make U invertible. Since the resulting matrix $\tilde{H}_{n\times m}$ is obtained by just column shifting of the parity check matrix $H_{n\times m}$ and each of the sub-matrices are sparse, so this method offer a linear complexity in order.

2.4. Method 4

In this method[16] if the parity check matrix has one part regular systematic and one part is the identity matrix such as:

$$H_{m\times n'} = \left[\bar{H}_{m\times n} \;\; I_{m\times m}\right] \quad (28)$$

The matrix $H_{m\times n'}$ is irregular systematic where $n'=n+m$. The parity equation for this type of the matrix is found simply as:

$$p_i = \sum_{j=1}^{N-M} h_{i,j}s_j \quad (29)$$

This need not to find the inverse andThe codeword is now given by $X=[s \;\; p]$.

3. LDPC Decoding Algorithms

The message passing algorithm [1, 3], [17-20] is a decoding algorithm in which messages are passed from node to node through the tanner graph used for complicated calculation using distributed hardware. When the graph is cycle-free, the message passing algorithm is recursive algorithm that always converge, after a finite number of messages have been passed, the true a posteriori probability (APP) log-likelihood ratios(LLR) is defined as:

$$\alpha_n = \log\frac{\Pr[x_n=1\mid y]}{\Pr[x_n=0\mid y]} \quad (30)$$

Where $Y=X+n$ and y is the received sequence, x_n is the transmitted codeword sequence and n is the additive white Gaussian noise. α_n is the APP log likelihood ratio (LLR).

3.1. Probability Domain Decoding Algorithm

A codeword $X=\{x_1, x_2.....x_n\}$ is transmitted after BPSK modulation over a noisy AWGN channel , the received sequence is $Y=X+n$ where $Y=\{y_1, y_2, ...y_n\}$, n is the additive white.

Gaussian noise with zero mean and variance $\pm^2$ {n~N(0, $\pm^2$)} as shown in Figure (3).

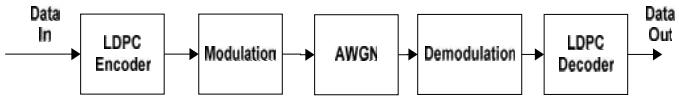


Figure 3. LDPC encoder and decoder in a communication system

Now the LDPC decoding algorithm can be stated in the following steps for parity check $H_{m\times n}$ where m is the number of rows and n is the number of columns. The following notation is introduced to represent the decoding algorithms effectively.

$j = n$ =the number of columns in parity check matrix $H_{m\times n}$

$i = m$ = the number of rows in parity check matrix $H_{m\times n}$

$N(j) = \{i : h_{ji} = 1\}$: The set of column locations of the ones (1's) in the ith row of H .

$N(j) \setminus i$: The set of column locations of the ones (1's) in the ith row of H excluding location j .

$M(i) = \{j : h_{ji} = 1\}$: The set of row locations of the ones (1's) in the jth column of H .

$M(i) \setminus j$: The set of row locations of the ones (1's) in the jth column of H excluding location i.

Now the LDPC decoding algorithm can be demonstrated in the following steps.

Initialization: The a posteriori probabilities(APP) are initialized to channel output as.

For s (i, j) , set:

$$q_{ji}^{0} = f_i^{0} = \frac{1}{1 + \exp(\frac{2y_i}{\pm^2})} \tag{31}$$

$$q_{ji}^{1} = f_i^{1} = \frac{1}{1 + \exp(\frac{-2y_i}{\pm^2})} \tag{32}$$

$$f_i = \frac{f_i^{0}}{f_i^{1}} \tag{33}$$

Set the number of iteration as $I_{\max}$

1) Parity node update(Horizontal processing): Update r_{ji}

$$¢\, r_{ji} = \prod_{i' \in N(j)\setminus i} (q_{ji'}^{0} - q_{ji'}^{1}) \tag{34}$$

$$r_{ji} = \frac{1 + ¢\, r_{ji} / (q_{ji}^{0} - q_{ji}^{1})}{1 - ¢\, r_{ji} / (q_{ji}^{0} + q_{ji}^{1})} \tag{35}$$

2) Bit node update (Vertical processing): Updating the variable node as:

$$\overline{R_i} = \prod_{j' \in M(i)\setminus j} r_{j'i} \tag{36}$$

$$s_{ji} = f_i \frac{\overline{R_i}}{r_{ji}} \tag{37}$$

Or equivalently,

$$s_{ji} = \frac{q_{ji}^{0}}{q_{ji}^{1}} \tag{38}$$

Updating q_{ji}^{0} and q_{ji}^{0} as follow:

$$q_{ji}^{0} = \frac{s_{ji}}{1+s_{ji}} \quad \text{and} \quad q_{ji}^{1} = 1 - q_{ji}^{0} \tag{39}$$

3) Calculating estimated codeword: In this step , the APP likelihood ratio R_i is computed and then the estimated codeword is calculated.

$$R_i = f_i \overline{R}_i \tag{40}$$

$$\hat{X} = \begin{cases} 0 & \text{for } R_i > 0 \\ 1 & \text{else} \end{cases} \tag{41}$$

R_i Comparison depends on the modulation scheme used .Here it is for typical BPSK modulation.

4) Stop condition : If the parity check equation is satisfied

$$H.(\hat{x}_1, \hat{x}_2 .....\hat{x}_i)^T = 0 \tag{42}$$

Or the maximum iteration $(\mathrm{I}_{\max})$ is reached then terminates the decoding or otherwise go to step2. Alternative way for Step 3 to Step 5:this can also be written in the form:

Step 3: In step 3, Equation (36) and equation (40) take the following form:

First calculate

$$Q_i(0) = K_i(1-P_i) \prod_{j' \in M(i) \setminus j} r_{j'i}(0) \tag{43}$$

$$Q_i(1) = K_i P_i \prod_{j' \in M(i) \setminus j} r_{j'i}(1) \tag{44}$$

Now Update

$$q_{ji}(0) = K_{ji}(1-P_i) \prod_{i' \in N(j) \setminus i} r_{i'j}(0) \tag{45}$$

$$q_{ji}(1) = K_{ji}(P_i) \prod_{i' \in N(j) \setminus i} r_{i'j}(1) \tag{46}$$

The constant K has the value such that $Q_i(0) + Q_i(1) = 1$ and $P_i = Pr(x_i = 1 | y_i)$ is the probability of $x_i = 1$ under that the knowledge of the received signal *y* is known, while the code structure is not considered. Set $q_{ji}(0) = 1 - P_i$ and $q_{ji}(1) = P_i$ for all i, j for which $h_{ji} = 1$.

Step 4: For every column index i , compute

$$\hat{X} = \begin{cases} 1, \text{if } Q_i(1) \geq 0.5 (or > Q_i(0)) \\ 0, \text{else} \end{cases} \quad (47)$$

If $rem(\hat{X}.H^T, 2) = 0$, or if the maximum number of iterationsis reached, then stop, else,continue iteration from Step 2.

3.2. Log Domain Decoding Algorithm

Due to many multiplications involved in BP SPA which can make it numerically unstable, therefore log domain SPA is preferred. We define the following notations:

$$L(f_i) = L(\frac{f_i^0}{f_i^1}) = \log\left(\frac{\Pr(x_i = 0 \mid y_i)}{\Pr(x_i = 1 \mid y_i)}\right) = \log\left(\frac{1/1 + \exp\left(2y_i / \pm^2\right)}{1/1 + \exp\left(-2y_i / \pm^2\right)}\right) = \frac{2y_i}{\pm^2}$$

$$L(r_{ji}) = \log\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right); \quad L(q_{ji}) = \log\left(\frac{q_{ji}(0)}{q_{ji}(1)}\right); \; L(R_i) = \log\left(\frac{Q_i(0)}{Q_i(1)}\right)$$

1) Initialization : Set the maximum number of iterations $(I_{\max})$ and initialize as follows:

$$L(q_{ji}) = L(f_i) = \frac{2y_i}{\pm^2} \quad (48)$$

2) Parity node update(Horizontal process): For initial derivation purpose for log-domain SPA,

As $r_{ji}(0) = 1 - r_{ji}(1)$ and $\tanh\left[\frac{1}{2}\log(\frac{p_0}{p_1}\right] = p_0 - p_1 = 1 - 2p_1$, we can write:

$$\tanh\left(\frac{1}{2}L(r_{ji})\right) = \prod_{i' \in N(j) \setminus i} \tanh\left(\frac{1}{2}L(q_{ji'})\right) \quad (49)$$

Now factorizing $L(q_{ij})$ into sign and magnitude components as follow:

$$L(q_{ij}) = \sim_{ji} \; _{ji}, \text{ where } \sim_{ji} = sign\left[L(q_{ji})\right] \text{ and } \; _{ji} = \mid L(q_{ji}) \mid$$

Re-writing Equation (49) as:

$$\tanh\left(\frac{1}{2}L(r_{ji})\right) = \prod_{i' \in N(j) \setminus i} \sim_{ji'} . \prod_{i' \in N(j) \setminus i} \tanh\left(\frac{1}{2} \; _{ji'}\right)$$

$$L(r_{ji}) = \prod_{i'} \sim_{ji'} . 2\tanh^{-1}\left(\prod_{i'} \tanh\left(\frac{1}{2} \; _{i'j}\right)\right)$$

$$= \prod_{i'} \sim_{ji'} . 2\tanh^{-1}\log^{-1}\log\left(\prod_{i'} \tanh\left(\frac{1}{2} \; _{i'j}\right)\right)$$

$$= \prod_{i'} \sim_{ji'} . 2\tanh^{-1}\log^{-1}\sum_{i'} \log\left(\tanh\left(\frac{1}{2} \; _{i'j}\right)\right)$$

$$= \prod_{i' \in N(j) \setminus j} \sim_{ji'} . \Phi\left(\sum_{i' \in N(j) \setminus j} \Phi\left(\; _{ji'}\right)\right) \quad (50)$$

Here $\Phi(x) = \Phi(x)^{-1} = -\log[\tanh(x/2) = \log\left[\frac{e^x+1}{e^x-1}\right]$ such that $x > 0$. This function can be implemented as look up table.

1) Bit node update (Vertical processing): Updating the variable node. Dividing equation (45) by (46) and then taking the logarithm of both sides, we get:

$$L(q_{ji}) = L(f_i) + \sum_{j' \in M(i) \setminus j} L(r_{j'i}) \tag{51}$$

2) Updating the final Log-likelihood ratio (LLR):
Similarly we get from Equation (43) and (44)

$$L(R_i) = L(f_i) + \sum_{j \in M(i)} L(r_{ji}) \tag{52}$$

3) Calculating estimated codeword and Stop condition:
For every column index i, we calculate

$$\hat{X} = \begin{cases} 0 & \text{for } \mathrm{L}(R_i) > 0 \\ 1 & \text{else} \end{cases} \tag{53}$$

If $rem(\hat{X}.H^T, 2) = rem(\hat{x}_1, \hat{x}_2, ...\hat{x}_n.H^T, 2) = 0$, or if the maximum number of iterations is reached, then stop, else, continue iteration from Step 2.

3.3. Approximated Log-domain SPA algorithm

The approximation to Log-domain SPA knows as min-sum algorithm (MSA) [17], greatly reduces the complexity but with reduced performance. In the log-domain SPA, the complexity is at the step 2. Min sum decoding algorithm reduces the complexity by approximating the magnitude of the initial LLR ($L(q_{ji})$). From equation (50), we get the approximation as:

$$L(r_{ji}) = \prod_{i' \in N(j) \setminus j} \sim_{ji'} .\Phi\left(\sum_{i' \in N(j) \setminus j} \Phi\left(\quad_{ji'} \right) \right) \cong \prod_{i' \in N(j) \setminus i} \sim_{ji'} .Min_{i' \in N(j) \setminus i} |\left(\quad_{ji'} \right)| \tag{54}$$

After this modification, all the steps are repeated in the same way in as in Log-domain SPA in section (3.2).

3.4. Min-Sum Decoding Algorithm

Here, first of all, we define some notations as:

X = the transmitted actual sequence; Y = the received sequence; $L_i = L(f_i)$ = initial channel LLR

$V_{ji} = L(q_{ji})$ =variable node message; $C_{ji} = L(r_{ji})$ = check node message

$\hat{L}_i^k$ =The final LLR used for calculating the estimated codeword ($\hat{X}$)

The min-sum decoding algorithm is now stated in the steps below for parity check matrix.

1) Initialization: Set $L_i = Y$ as the initial log likelihood ratio (LLR) as no priori information[18, 20] about the AWGN is required. and for each $(j,i) \in \{(m,n) \mid h_{mn} = 1\}$, we initialize the variable node as:

$$V_{ji} = L_i \tag{55}$$

Set the maximum number of iterations ($I_{\max}$) as $k = 0$ to $I_{\max}$

2) Parity node update(Horizontal processing): Message passing from variable to parity(check) node for $j = 0$ to $M - 1(d_v)$ and C_{ji}^k is updated for each $i \in N(j)$ as follows:

$$C_{ji}^k = \prod_{i' \in N(j) \backslash i} sign(V_{ji'}^{k-1}).Min_{\, i' \in N(j) \backslash i} |V_{ji'}^{k-1}| \tag{56}$$

3) Bit node updates (Vertical processing):Check node to variable node
For $i = 0$ to $N - 1(d_c)$, calculate the final LLR

$$\hat{L}_i^k = L_i + \sum_{j \in M(i)} C_{ji}^k \tag{57}$$

Updating the variable (bit) node message for each $j \in M(i)$

$$V_{ji}^k = \hat{L}_i^k - C_{ji}^k \quad \text{or} \ (V_{ji}^k = L_i + \sum_{j' \in M(i) \backslash j} C_{j'i}^k) \tag{58}$$

4) Hard decision: Estimating the codeword by hard decision as follows,

$$\hat{X} = \begin{cases} 0 & \text{for } \hat{L}_i > 0 \\ 1 & \text{else} \end{cases} \tag{59}$$

5) Stop condition: If the parity check equation is satisfied i.e.

$$H.\hat{X}^T = 0 \tag{60}$$

Or the iteration $I_{\max}$ is reached then terminates the decoding or otherwise go back to step 2.

The message passing between check node and variable node in step 2 and step 3 can also be represented in a graphical way as shown in Figure 4 and Figure 5.

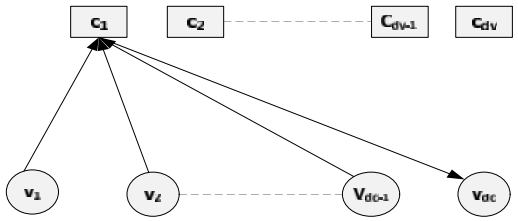


Figure 4. Horizontal processing: bit nodes to check nodes

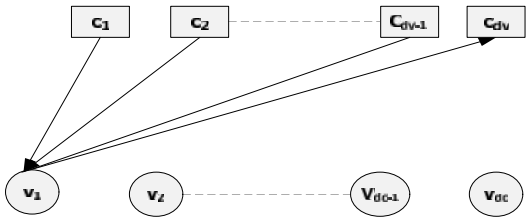


Figure 5. Vertical Processing: check nodes to bit nodes

3.5 Enhanced Min-Sum Decoding Algorithm

Min-Sum algorithm (MSA)provides a significant decrease in decoding complexity and is also independent of the noise variance, so no a priori information of the AWGN channel is required. But MSA causes performance degradation reasonable because of check node messages overestimation in comparison to SPA.

For a certain edges v and c denote L_1 and L_2 as the values of $L(r_{ji})$ and C_{ji}^k computed by SPA and MSA respectively for the same messages during the previous iteration.

Then the following statement holds about the relationship [20, 21]of L_1 and L_2.

1) The values L_1 and L_2 have the same sign i.e. $sign(L_1) = sign(L_2)$

2) The magnitude of L_2 is always greater than L_1 i.e. $L_2 > L_1$

The normalization factor is required to correct the magnitude overestimation and bring L_2 close to the value of L_1 which significantly improve the decoding performance. Several modification algorithms have been proposed in order to improve the performance of reduced complexity min-sum decoding algorithm (also known as universal most powerful (UMP) algorithms [18].Two approaches [20, 21] are most popular for the check node message update in the equation (56) as under:

a) First Approach :Normalized Min-Sum Decoding Algorithm

$$C_{ji}^{k} = sf. \prod_{i' \in N(j) \backslash i} sign(V_{ji'}^{k-1}).Min_{\, i' \in N(j) \backslash i} |V_{ji'}^{k-1}| \tag{61}$$

The scaling factor (sf) is used as normalization factor to correcting the variable message overestimation during the check node update in step 2. The value of scaling factor is $0 < sf \leq 1$.

b) Second Approach :Offset Min-Sum Decoding Algorithm

$$C_{ji}^{k} = \prod_{i' \in N(j) \backslash i} sign(V_{ji'}^{k-1}).Max_{\, i' \in N(j) \backslash i} |V_{ji'}^{k-1} - f, 0| \tag{62}$$

All the extrinsic messages with reliability values smaller than the offset factor f , are set to 0, such that they have no contribution to the preceding bit node processing. In both the approaches, the normalization factor and the offset values should vary with each iteration number for achieving better performance but for making the processing complexity simple, it is kept constant. All other steps are repeated in the same way as in section (3.4).

4. Effect of Normalization Factor on Performance

In Equation (61), the check node message is updated as follow:

$$C_{ji}^{k} = sf. \prod_{\substack{i' \in N(j) \\ i' \neq i}} sign(V_{ji'}^{k-1}). \min_{\substack{i' \in N(j) \\ i' \neq i}} |V_{ji'}^{k-1}| \tag{62}$$

The range for scaling factor (*sf*) value is *0< sf <1*. This is called the single factor or single way normalized min-sum decoding algorithm. It uses one scaling factor for updating the check node message during the row processing. The normalized min-sum algorithm has been simulated here to show the scaling factor effect on error performance when either code rate or length is changed. Three types of codes have been selected for the results validation and comparative analysis. Regular medium length codes (1024, 512) and (1296, 864) are chosen to show the effect for code rate change and a short length code (684,324) is chosen to show effect when the code length is changed. The simulation results in figures (6), (7) and (8)signify that error performance is affected greatly with scaling factor and it varies with code rate and length. Actually adaptive normalization can greatly improve the performance but then it becomes very complex for real time applications and makes it practically impossible although in theory it can shows good results.

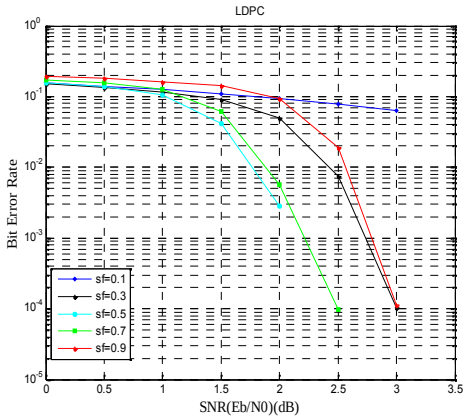


Figure 6. Performance of normalized min-sum for (1296, 864) LDPC code for various sf values, iterations=10

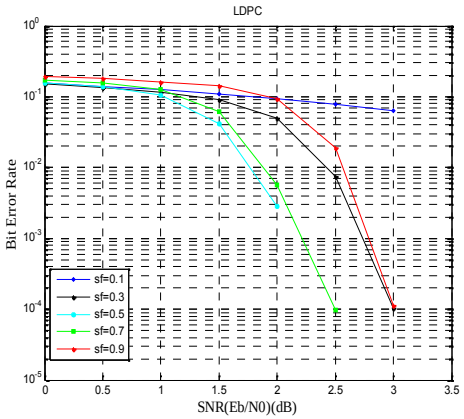


Figure 7. BER performance ofnormalizedmin-sum for (648, 324) LDPC codefor various *sf* values, iteration=10

5. Quasi Cyclic LDPC Codes

Quasi- cyclic (QC) LDPC codes are the good candidate to solve the memory problem as their parity check matrices consists of circulant permutation matrices [22] or the zero matrices. QC LDPC codes have also shown performance [23] close to Shannon limit. QC LDPC is becoming popular among the hardware implementation related researchers and many efficient encoding [24-28] methods for implementation has been proposed.

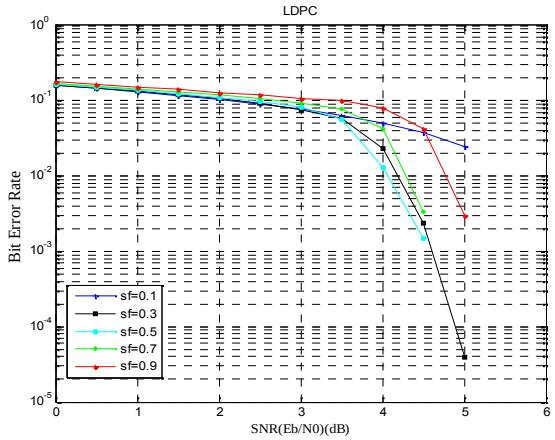


Figure 8. BER performance of one-way normalized min-sum for (1024, 512) LDPC code for various sf values, iterations=10

5.1. Constructing Quasi Cyclic Matrices

The structure of the QC LDPC codes depends on the arrangement of the constituents' sub-matrices and their shift values. Two constraints are always kept in mind while designing LDPC codes; 1) High girth to improve the performance, 2) low complexity in implementation. Some of the methods include finite geometry [29, 30], algebraic construction [31], finite field approach [32].

The QC-LDPC parity check matrix has the general structure as:

$$H_{QC} = [C_1, C_2 ..... C_i] \tag{64}$$

Where $C_1, C_2 ..... C_i$ are all circular shifted matrices such that the parity check matrix is full rank. Shifted identity matrices are easily obtained by shifting the row of an identity matrix to

the right or left by some amount. Some arrangement of the identity matrices are shown as:

$$\begin{pmatrix} I_{11} & I_{12} & I_{13} & I_{14} \\ I_{21} & I_{22} & I_{23} & I_{24} \\ I_{31} & I_{32} & I_{33} & I_{34} \end{pmatrix} \quad (65)$$

$$\begin{pmatrix} I_{11} & O & I_{13} & I_{14} \\ O & I_{22} & I_{23} & I_{24} \\ I_{31} & I_{32} & O & I_{34} \\ I_{41} & I_{42} & I_{43} & O \end{pmatrix} \quad (66)$$

Equation (65) is with all non-zero sub-matrices and Equation (66) is with zero sub-matrices. Each sub-matrix in a row shows the weight one and similarly for each column. In the Equation (65), the row and column weights are 3 and 4 respectively while in the equation (66), the row and column weights are 3 and 3 respectively due to the placement of the zero sub-matrices.

The circulant matrix of size $m \times m$ can be shown as:

$$C = \begin{pmatrix} c_0 & c_1 & \cdots & c_{m-1} \\ c_{m-1} & c_0 & \cdots & c_{m-2} \\ \vdots & \vdots & \cdots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{pmatrix} \quad (67)$$

A circulant matrix is uniquely specified by a polynomial formed by the entries of the first row.

$$c(x) = c_0 + c_1 x + c_2 x^2 + \cdots\cdots c_{m-1} x^{m-1} \quad (68)$$

For a rate $1/k$ systematic QC code has the $m \times mp$ generator matrix [28] of the form:

$$G_{QC} = [I_m, p_1, p_2 \cdots p_{k-1}] = [I_{m \times m} \quad P_{m \times mp}] \quad (69)$$

$$p_j = s_1 G_{1,j} + s_2 G_{2,j} + \cdots\cdots s_m G_{m,j} \quad (70)$$

Where I_m is the $m \times m$ identity matrix and C_i is the $m \times m$ binary circulant matrices.After getting the generator matrix, the encoding is simply done as:

$$X = s.G_{QC} = s.[I \quad P] = [s \;\; p \;] \quad (71)$$

Where s contains the information bits.

5.2. Encoding Techniques for QC LDPC Codes

The parity check matrix for QC LDPC codes[25] is represented as:

$$H_{QC} = [C_1, C_2 \ldots\ldots C_i] \quad (72)$$

This matrix can be decomposed into two parts:

$$H_{QC} = [M \quad D] \quad (73)$$

Where D is a square matrix and must be invertible. D and M both are quasi cyclic, composed of circulant. The desired generator matrix G has the following form:

$$G_{QC} = [I \quad P] \tag{74}$$

The necessary and sufficient condition for the generator matrix is that:

$$G_{QC} H_{QC} = [0] \tag{75}$$

The codeword $X = [s \;\; p]$ has two parts; the information bits s and the parity bits p such that Thus X is a codeword if and only if:

$$\begin{bmatrix} I \\ P \end{bmatrix} [M \quad D] = 0 \tag{76}$$

Or equivalently,

$$MI^T + DP^T = 0 \tag{77}$$

$$\Rightarrow P^T = D^{-1} MI = D^{-1} M \quad \text{(Mod 2)} \tag{78}$$

If a matrix is circular or composed of circular permutationmatrices, the inverse is also a circular matrix[33] which can be obtained as follows:

The cyclic matrix in Equation (67) is such that $C^T = C$ and $X^T = (x_1, x_2, \cdots, x_n)^T$ is a matrix of polynomial such that:

$$CX^T = (1, 0, 0, \cdots\cdots, 0)^T \tag{79}$$

This can be written in the polynomial form as:

$$\begin{aligned} &c_0 x_1 + c_2 x_2 + \cdots\cdots + c_{m-1} x_n = 1 \\ &c_{m-1} x_1 + c_0 x_2 + \cdots + c_{m-2} x_n = 0 \\ &\vdots \;\; \vdots \;\; \vdots \;\; \vdots \;\; \vdots \;\; \vdots \;\; \vdots \;\; \vdots \\ &c_1 x_1 + c_2 x_2 + \cdots\cdots\cdots + c_0 x_n = 0 \end{aligned} \tag{80}$$

Solving the Equation (80), we obtainthe inverse C^{-1} of C written in the form:

$$C^{-1} = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_1 \\ \vdots & \vdots & & \vdots \\ x_n & x_1 & \cdots & x_{n-1} \end{pmatrix} \tag{81}$$

This can be verified by multiplying the Equation (67) and (81) such that:

$$C^{-1} C = I(\textit{indentity matrix}) \tag{82}$$

$$\begin{pmatrix} c_0 & c_1 & \cdots & c_{m-1} \\ c_{m-1} & c_0 & \cdots & c_{m-2} \\ \vdots & \vdots & \cdots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_1 \\ \vdots & \vdots & & \vdots \\ x_n & x_1 & \cdots & x_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \tag{83}$$

This method can be used for obtaining the D^{-1} in Equation (77).

The structure of the D matrix plays an important role in the QC LDPC parity check matrix. In [34] it was shown that QC LDPC codes based on D-matrix and Q-matrix, designed by modified PEG, are more suitable to be used than identity matrix as they outperform than that of identity matrix.

5.3. QC LDPC Encoder Hardware Implementation

QC–LDPC codes have encoding advantage over conventional LDPC codes and their encoding can be implemented by shift register [35, 36] called shift register adder accumulate(SRAA) with complexity linearly proportional to the number of parity bits as shown in the Equation (69) of the code .Also QC-LPDC codes require less amount of memory as compared to the general LDPC codes, since their parity check matrices consist of the circulant permutation matrices or the zero matrices.

6. QC LDPC Decoder Hardware Design

QC LDPC codes are the family of implementation oriented codes. QC LDPC codes have not only efficient encoding characteristic but has also solved the problem of decoding complexity. QC LDP codes have got the following advantages;

1) Memory efficient;
2) Less Hardware Complexity;
3) High Convergence Speed through Layered decoding;
4) High Throughput

6.1. Layered and Non-layered LDPC Codes

Layered schedule considers the parity check matrix as layers of check equations and update the variable node information right after check node information of current layer. Two types of layer decoding has been considered in [37]. Many different approaches has been made to improve it further in terms of memory, energy (power) and throughput [38-43].

The straight forward horizontal layered min-sum decoding algorithm is given below:

1) Initialization: $L_{ji}^{0} = L_{i}^{0} = Y \;\&\; \mathrm{C}_{ji}^{0} = 0$ (as stated in chapter 2) where $i = 0,1,2\cdots N-1$
2) Set maximum iteration $k = 1$ to $I_{\max}$
   a) Layer initialization: (Here, in general each row of a parity check matrix is considered as one layer). Set $t = 1$ to $\mathrm{t}_{\max}$ (No. of Rows or Row Blocks)

$$V_{ji}^{t} = L_{i}^{t-1} - C_{ji}^{t-1} \tag{84}$$

   b) Check Node Update:

$$C_{ji}^{t} = s.f \prod_{i' \in N(j)\backslash i} sign(V_{Vji'}^{t}) \min_{i' \in N(j)\backslash i} |V_{Vji'}^{t}| \tag{85}$$

   c) Variable Node update:

$$L_{i}^{t} = Y + \sum C_{ji}^{t} \tag{86}$$

Go to step (a) until *M* reaches.

$$\hat{X} = \begin{cases} 1 & \mathrm{L}_{i}^{t_{\max}} < 0 \\ 0 & \mathrm{L}_{i}^{t_{\max}} \geq 0 \end{cases} \tag{87}$$

If $rem(\hat{X}.H^T,2) = rem(\hat{x}_1,\hat{x}_2,.....\hat{x}_n.H^T,2) = 0$, or if the maximum number of iterations is reached, then stop, else, continue iteration from Step 2.

6.2. Memory Efficient Layered LDPC Decoding

To minimize the interconnect complexity and memory size at the check node update (horizontal processing), the MSA can be formulated in such a way not to store all the messages but only the following four element [35-36], [40].

1) The 1st smallest magnitude
2) The 2nd smallest magnitude
3) The index of the 1st smallest magnitude
4) The signs of all the soft message of the row(variable messages).Note: This may not require in software simulation only in case quantized values are not used.

In the check to variable message passing phase, a check node c sends only the 1st smallest magnitude (*min1*), the 2nd smallest magnitude (*min2*) and the index of the 1st smallest magnitude (*index*) where *min1*≤ *min2*.Now the check node (step 2) is updated in the layered decoding in section (6.2) as:

$$C_i^t = \begin{cases} sf.sign(V_i^t).\min 1 & if \;\; i == index \\ sf.sign(V_i^t).\min 2 & ohterwise \end{cases} \tag{88}$$

7. QC LDPC Simulation and Performance Analysis

Consider a quasi cyclic parity check matrix $H_{600\times1500}$ which has the each sub-matrices (circulant) with $size(m',n') = 150$ and LDPC code length $L = 1500$, $d_v = 4$ and $d_c = 10$. The code has been simulated for SPA, MSA with normalized and offset values and with the new improved MSA [44].

$H = [M \mid D]$ where D is square matrix such that it is full rank and is represented as:

$$D = \begin{pmatrix} D_1 & \cdots & 0 \\ \vdots & D_2 & \vdots \\ 0 & \cdots & D_i \end{pmatrix}$$

The diagonal elements are designed such that the parity check matrix H is regular and the upper and lower triangular values are zero matrices.

$$H = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} & D_1 & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} & 0 & D_2 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} & 0 & 0 & D_3 & 0 \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} & 0 & 0 & 0 & D_4 \end{bmatrix} \tag{89}$$

Here $C_{i,j}$ is a circulant permutation matrix of size 150x150. The QC LDPC code is simulated for SPA, simple MSA , layered normalized MSA which is based on the memory efficient MSA as stated in section (6.2), offset MSA and new improved MSA [44]. In Figure 9, the value of the offset is chosen by search and is considered here as the approximately good one and there may be exist some other optimum or near optimum values. Also the normalized values for MSA are chosen such that it gives good performance as well as the hardware implementation is also simple.Recently adaptive normalized/offset min-sum algorithms [45, 46] are proposed which further improve the performance.

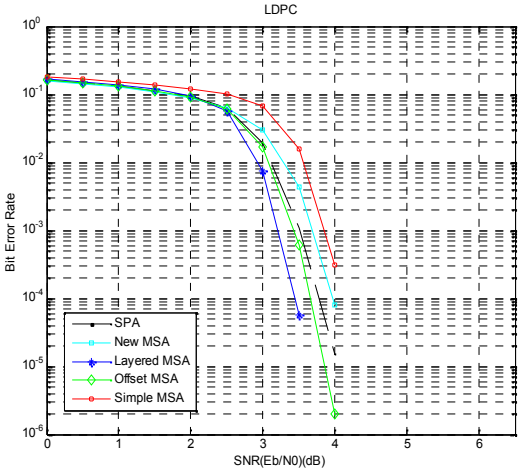


Figure 9. Simulations for QC LDPC code (1500, 4, 10), iterations=10

8. Design and Sparsity of LDPC Matrices

LDPC matrices are designed such that it can give good performance as well as easy to implement in hardware and also offer good performance for MIMO/Cooperative Communication [47]. One of the examples of LDPC matrices are illustrated in Figure 10 here for general idea.

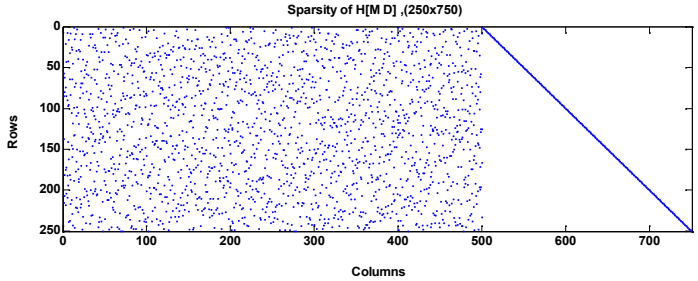


Figure 10. Distribution of 1's in H =[M D](size=250x750), where M & D are composed of Circulant Permutation Matrices

9. Conclusion

This paper summarizes the encoding and decoding algorithms of LDPC and also provides thedetail analysis to make it more comprehensible for those interested in LDPC. It also shows the recent trends and challenges in LDPC codes and indicates the future direction of the powerful coding for reliable communication. QC LDPC is specially elaborated in this paper to show the practical importance and to give better insight for hardware implementation for many applications. Better code construction; offering less delays, small memory size and fast decoding with better performance, is still open for research.

References

[1] RG Gallager. Low-Density Parity-Check Code. Cambridge. MA: MIT Press. 1963.
[2] DJ Mackay RN. Near Shannon Limit Performance of Low Density Parity Check Codes. *Electronic Letters.* 1996; 32(18): 1645-1646.
[3] DJ MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Infor. Theory.* 1999; 45: 399-431.
[4] CE Shannon. A mathematical theory of communication. *Bell System Technical Journal.* 1948; 27: 379-423 and 623-656.
[5] C Berrou AG, P Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. *IEEE Transactions on Communications.* 1993: 1064-1070.

[6] M Tanner R. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory.* 1981; 27: 533-547.
[7] B Sklar. Digital Communication: Fundamentals and Applications. 2nd Edition.
[8] FR Kschischang BJF, HALoeliger. Factor graphs and the sum-product algorithm. I*EEE Transactions on Information Theory.* 47: 498-519.
[9] DJC MacKay, STW, MC Davey. Comparison of construction of irreugualr Gallager Codes. I*EEE Trans. Communincation*. 1999; 47: 1449-1454.
[10] MG Luby MM, MA Shokrollahi, DA Spielman. Improved Low Density Parity Check Codes Using Irregular Graphs. *IEEE Trans. Information Theory.* 2001; 47: 585-598.
[11] http://www.inference.phy.cam.ac.uk/mackay/codes/.
[12] http://www.cs.utoronto.ca/~radford/ldpc.software.html.
[13] S Chae YP. *Low Complexity Encoding of Regular Low Density Parity Check Codes.* IEEE 58th Vehicular Technology Conference(VTC 2003-Fall). 2003: 1822-1826.
[14] Zongjie Tu SZ. *Overview of LDPC Codes.* 7th IEEE International Conference on Computer and Information Technology (CIT 2007). 2007: 469 - 474.
[15] TJ Richardson RLU. Efficient Encoding of Low-Density Parity-Check Codes. *IEEE TRANSACTIONS ON INFORMATION THEORY.* 2001; 47(2).
[16] Fengfan Yang JC, Peng Zong, Shunwai Zhang, Qiuxia Zhang. Joint iterative decoding for pragmatic irregular LDPC-coded multi-relay cooperations. International Journal of Electronics. 2011; 98(10) 1383-1397.
[17] JR Barry EAL, DG Messserschmitt. Digital Communication. 3rd Edition.
[18] M Fossorier MM. *R*educed Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation. *IEEE Transactions on communications.* 1999; 47.
[19] TJ Richardson RLU. The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding. *IEEE Trans.Information Theory*. 2001; 47: 599-618.
[20] J Chen MPCF. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Transactions on Communication.* 2002; 50: 406-414.
[21] Waheed Ullah, J., Fengfan Yang, S.M.Aziz, *"Two-Way Normalization of Min-Sum Decoding Algorithm for Medium and Short Length Low Density Parity Check Codes."* 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2011: p. 1-5.
[22] MPC Fossorier. Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices. *IEEE Transactions on Information theory.* 2004; 50(8):. 1788-1793.
[23] L Chen JX, I Djurdjevic, S Lin. Near Shannon Limit Quasi-Cyclic Low Density Parity- Check Codes. *IEEE Trans. on Communications.* 2004.
[24] S Myung KY, J Kim. Quasi-Cyclic LDPC Codes for Fast Encoding. *IEEE transaction on Information Theory.* 2005; 51(8): 2894-2901.
[25] Zongwang Li LC, Lingqi Zeng, Shu Lin, Wai H Fong. Efficient Encoding of QC LDPC code. *IEEE Transactions on Communications.* 2006; 54: 71-81.
[26] Li Peng GZ, Xiaoxiao Wu, Xi Yan. *The Efficient D-LDPC Encoder based on arithmetical progression.* Proceeding. IEEE GMC. Shanghai. 2007.
[27] Zhang Wenjun LC. A Novel Encoding Architecure of QC LDPC codes based on RAMs. *Information Science and System Science*. 2007.
[28] M Baldi FB, F Chiaraluce. On a Family of Circulant Matrices for Quasi-Cyclic Low-Density Generator Matrix Codes. *IEEE Transactions on Information Theory.* 2011; 57(9): 6052-6067.
[29] Y Kou SL, MPC Fossorier. LDPC codes based on Finite Geomatry. *IEEE Transaction on Information Theory.* 2001; 47: 2711-2736.
[30] H Tang JX, Y Kou, S Lin, KA Ghaffar. Codes on Finite Geometries. I*EEE transaction on Information Theory.* 2005; 51: 572-596.
[31] H Tang JX, Y Kou, S Lin, KA Ghaffar. On algebraic construction of Gallager and circulant low-density parity-check codes. *IEEE transaction on Information Theory.* 2004; 50(6): 1269-1279.
[32] L Lan LZ, YY Tai, L Chen, S Lin, K Abdel-Ghaffar. Construction of Quasi-Cyclic LDPC Codes for AWGN and Binary Erasure Channels: A Finite Field Approach. 2007; 53: 2429-2458.
[33] Nian-Liang W. The simple method of cyclic matrix Inversion. *Journal of Shangluo Teacher College.* 2002; 16(4).
[34] W Zhan GZ, L Peng, X Yan. *Quasi-cyclic LDPC codes based on D and Q matrices through progressive edge growth.* International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2007). 2007: 12-15.
[35] Hao Zhong WX, Ningde Xie, Tong Zhang. Area-Efficient Min-Sum Decoder Design for High-Rate Quasi-Cyclic Low-Density Parity-Check Codes in Magnetic Recording. *IEEE Transactions on Magnetics.* 2007; 43: 4117-4122.
[36] Nan Jiang KP, Jian Song, Chanyong Pan, Zhixing Yang. High-Throughput QC-LDPC Decoders. *IEEE TRANSACTIONS ON BROADCASTING.* 2009; 55(2):. 251-259.
[37] J Zhang MF. Shuffled iterative decoding. *IEEE Transactions on Communications.* 2005; 53: 209-213.

[38] MM Mansour NRS. High-Throughput LDPC Decoders. *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS.* 2003; 11(6): 976-996.
[39] T Mohsenin BB. *High-throughput LDPC decoders using a multiple Split-Row method.* ICASSP. 2007. 2: 13-16.
[40] Zhiqiang Cui ZW, Youjian Liu. High-Throughput Layered LDPC Decoding Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.* 2009; 17: 582-587.
[41] Kai Zhang XH, Zhongfeng Wang. *H*igh-throughput layered decoder implementation for quasi-cyclic LDPC codes. *IEEE Journal on Selected Areas in Communications*. 2009; 17(6): 985-994.
[42] Kai He JS, Li Li, Zhongfeng Wang. *Low Power Decoder Design for QC LDPC Codes.* Proceedings of 2010 IEEE International Symposinm on Circuirts and Systems (ISCAS). 2010: 3937-3940.
[43] Jie Jin CYT. An Energy Efficient Layered Decoding Architecture for LDPC Decoder. 2010; 18(8): 1185-1195.
[44] Waheed Ullah J, Yang Fengfang. *Improved min-sum decoding algorithm for moderate length low density parity check codes.* International conference on Computer, Informatics, Cybernetics and Application (ICIA-2011). LNEE Springer. 2011: 939-944.
[45] Xiaofu Wu YS, Ming Jiang, Chunming Zhao. Adaptive-Normalized/Offset Min-Sum Algorithm. *IEEE Communications Letters*, 2010; 14: 667-669.
[46] Xiaofu Wu YS, Long Cui, Ming Jiang, Chunming Zhao. *Adaptive-normalized min-sum algorithm.* 2nd International Conference on Future Computer and Communication (ICFCC2010). 2010; 2: 661-663.
[47] Waheed Ullah, Yang Fengfan, Abid Yahya. QC LDPC Codes for MIMO and Cooperative Networks using Two Way Normalized Min-Sum Decoding. 12(7).