Modified Log Domain Decoding Algorithm for LDPC Codes over GF (q)

Md. Murad Hossain, Mohammad. Rakibul Islam, Md.Jahidul Islam, Asif Ahmed, Md.Shakir Khan, S.M.Ferdous

Islamic university of Technology, BoardBazar, Gazipur-1704, Dhaka,Bangladesh.

***Abstract*— A modified log domain decoding algorithm of Low density parity check (LDPC) codes over *GF(q)* using permutation to simplify the parity check equation is presented in this paper which is different from the conventional log domain decoding algorithm of Low Density Parity Check (LDPC) codes over *GF(q)* [11]. Modified Log domain is mathematically equivalent to the conventional log domain decoding but modified log-domain has advantages in terms of implementation, computational complexity and numerical stability. Further a variant of Log domain decoding defined as modified min-sum decoding is proposed, yielding a lower computational complexity and a little performance loss. The proposed algorithms and the conventional log domain decoding algorithm are compared both in terms of simulated BER performance of rate ½ LDPC code over GF(8) with N=204 and computational complexity.**

***Index Terms*— LDPC, GF (q), Log Domain Decoding, Min-sum decoding, sum-product algorithm, Iterative decoding.**

I. Introduction

Low density parity check codes (LDPC) is a linear block code which is defined by a sparse parity check matrix and it approaches Shannon –limit performance for binary field and long code lengths [1],[2],[4] ,[5]. But performance of binary LDPC code is degraded when the code word length is small or moderate, or when higher order modulation is for transmission [6]. LDPC codes designed over Galois Field *GF* (q>2) (also known as non-binary LDPC codes) have shown great performance for these cases [7]–[10]. But decoding complexity increases with q which makes the use of non-binary LDPC (NB-LDPC) codes is limited still today.

The belief propagation can be extended to decode NB-LDPC code but it has computational complexity dominated by $O(q^2)$ operation for each check node processing [7]. The reduction in complexity of BP can be done by carrying out the computations in log domain [11], Fourier domain [3], and mixed domain [12]. When the Galois field is a binary extension field with order $q=2^p$, the Fourier transform is easy to compute which reduces the decoding complexity to $O(p^{2P})$. [8], [3] report results for 2^p =256. Log- domain decoder combined with a Fast Fourier Transform (FFT) at the check node point with a look up table (LUT) of required operations is presented in [13]. The decoding complexity can be further reduced by the log-domain extended Min-sum (EMS) [14] and Min-max [15] algorithms. Although only additions are performed and it is independent of channel information, its complexity remains $O(q^2)$. The EMS algorithm employs 'sum' instead of 'max'in the check node processing. As a result, it can achieve better error correcting performance with higher complexity than the Min-max algorithm.

In this paper we propose two algorithms for decoding NB-binary LDPC. First algorithm is in the Log-domain. We efficiently employ permutation to simplify the parity check equation. It involves only sum operations which make this algorithm computationally less complex. We then expeditiously extend this algorithm to its min sum version which requires less sum operations than first one and it makes this algorithm very attractive for practical purpose.

The paper is organized as follows. In the next section we briefly discuss LDPC codes. We also present sum product algorithm and log-domain decoding algorithm over GF(q) in Section III and IV respectively to make the paper self contained. We develop our algorithm in section V. Section VI presents simulation results and section VI concludes this paper.

Manuscript received June 10, 2011.

Md. Murad Hossain is with the Electrical and Electronic Department, Islamic university of Technology as Lecturer. (phone: +8801717373701 and email: muradhossain87@gmail.com)

Dr. Mohammad. Rakibul Islam is with the Electrical and Electronic Department, Islamic university of Technology as Associate Professor. (phone: +8801819299389 and email: rakibultowhid@yahoo.com)

Md.Jahidul Islam is with the Electrical and Electronic Department, Islamic university of Technology. (e-mail: parthib_1090@yahoo.com.)
Asif Ahmed is with the Electrical and Electronic Department, Islamic university of Technology. (e-mail: asifahmed.iut@gmail.com).
Md.Shakir Khan is with the Electrical and Electronic Department, Islamic university of Technology. (e-mail: shakir_k@ymail.com).

S.M.Ferdous is with the Electrical and Electronic Department, Islamic university of Technology. (e-mail: tanzir68@gmail.com).

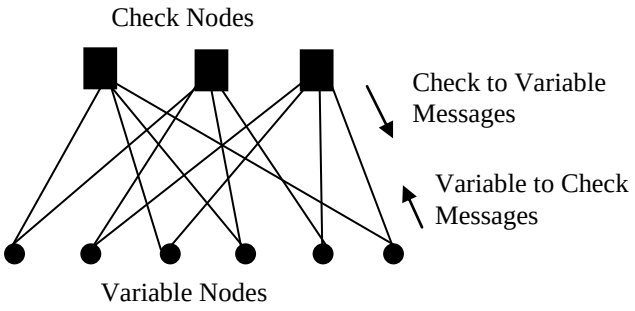

Fig. 1.Tanner graph representation of parity check matrix with row weight=3 and column weight=2.

II. LOW DENSITY PARITY CHECK CODES

LDPC code is defined by a sparse parity check matrix of *M* rows and *N* columns and the code rate is defined by $R \geq \frac{N-M}{N}$. A vector **c** is a codeword if and only if it satisfies:

$$c\,H^T = 0 \tag{1}$$

LDPC code is represented by a Tanner graph Fig. 1 which consists of *N* bit nodes and *M* check nodes that represent *N* bits of a codeword and *M* parity constraints. The graph has an edge between the nth bit node and mth check node if and only if bit is involved in the mth check i.e if $H_{mn} = 1$. Decoding algorithms of LDPC codes are iterative message passing decoders based on a factor graph.

In NB-LDPC code, the parity check matrix *H* of size $M \times N$ whose elements belong to a finite field *GF(q)* defining code C_H such that $C_H = \{\hat{c}\ \varepsilon\ GF\ (q)^N\ |H\hat{c}^T = 0\}$. The rank of *H* is *N - K* with $K \geq N - M$ and the code rate $R = \frac{K}{N} \geq 1 - \frac{M}{N}$ as in binary case.

The tanner graph Fig. 2 representation of a set of variable nodes belonging to *GF (q)* fully connected to a set of parity check nodes. We denote d_v the column weight of a symbol node and d_c is the row weight of a check node. d_v and d_c vary according to the symbol index and check index respectively in irregular LDPC code. A single parity check equation in NB-LDPC code involving d_c and codeword symbols c_n is as follows:

$$\sum_{n=0}^{d_c-1} h_{mn} c_n = 0 \tag{2}$$

in *GF (q)* and $m = \{1, 2, \ldots d_v\}$.Where h_{mn} is a nonzero value of the parity matrix *H*.

The messages in NB-LDPC codes can be probability weights vectors or Log Likelihood Ratio (LLR) vectors. But the use of LLR is more advantageous than probability weights vectors because it avoids use of multiplication and addition operations sum operations and symbols are less sensitive to quantization error [16].

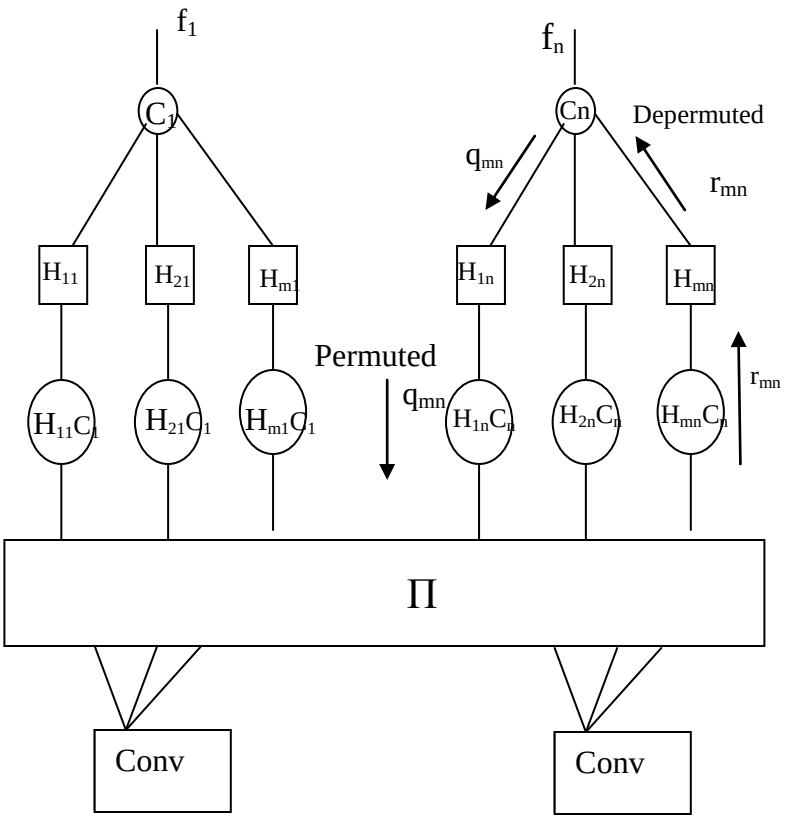

Fig. 2. Generalized factor graph of a NB-LDPC code [21].

The following notation will be used for an LLR vector of a random variable $z\ \varepsilon\ GF\ (q)$:

$\boldsymbol{\lambda}(z) = [\,\lambda[0] \ldots \lambda[q]$ and

$$\lambda\,[i] = log\,\frac{P(z=a_i)}{P(z=0)} \tag{3}$$

with $P(z = a_i)$ being the probability that the random variables *z* takes on the values $a_i \in \{1,2,\ldots\ldots q\}$.

The LLR messages at the channel output are *q-1* dimensional vectors in general denoted by $\boldsymbol{\lambda_{ch}} = [\lambda_{ch}[k]_{k\varepsilon\ \{1,2,\ldots q\}}]$. Each symbol of the codeword c_n , $n \in \{0,\ldots,N-1\}$ can be converted into a sequence of $log_2(q)$ bits $c_{ni} \in GF\ (2)$, $i \in \{0,\ldots, log_2(q) - 1\}$ for a binary additive white Gaussian noise channel (AWGN) i.e. a block of *Kb* bits is converted to a sequence of *K GF* (2^b) symbols according to some mapping, φ: $(GF(2))^b \rightarrow GF\ (2^b)$. Then message word $\boldsymbol{b} \in (GF(2^b))^k$, is encoded using the generator matrix, resulting in a codeword $c \in C$ *i.e.* $\boldsymbol{c = G^T b}$. The codeword is now converted to a vector $\boldsymbol{t}$ using signal constellation mapping. We are assuming BPSK signaling but it is possible to use higher order modulation [17]. Thus ψ: $GF\ (q) \rightarrow \Omega^b$, with $\Omega = \{-1,+1\}$ such that $\psi\ (c_k) = [t_{kb}, \ldots t_{(k+1)(b-1)}]^T$ and $t_i \in \Omega$ [11].

After BPSK mapping, the codeword is sent on the AWGN Channel:

$$\boldsymbol{x = t + n} \quad (4)$$

with $\boldsymbol{x}$ being the received noisy BPSK signal and $\boldsymbol{n}$ being a real white Gaussian noise random variable with variance $\sigma^2 = \frac{N_0}{2\,E_b\,R}$ where $\frac{E_b}{N_0}$ is the SNR per information bit.
The following notations will be used throughout the paper.

Notations:

- $n \in \{1,2,\ldots N\}$ a variable node of *H*.
- $m \in \{1,2,\ldots M\}$ a check node of *H*.
- $\mathcal{N}_m$, set of neighbor variables nodes of the check node *m*.
- $\mathcal{M}_{n,}$, set of neighbor check nodes of the variable node *n*.
- $\mathcal{N}_{m,n}$, set of neighbor variables nodes except n of the check node *m*.
- $\mathcal{M}_{\mathrm{n,m}}$ set of neighbor check nodes except *m* of the variable node *n*.
- *L (m)*, set of Local configurations verifying the check node *m* i.e. the set of sequences of *GF(q)* symbols verifying the linear constraint.
- $L(m \mid a_n = a)$ set of local configurations verifying *m*, such that $a_n = a$; for given $n \in \mathcal{N}_m$ and $a \in GF(q)$.
- $\lambda_n\,(a)$, the a priori information of the variable node *n* concerning the symbol *a*.
- $\widehat{\lambda_n}\,(a)$, the a posteriori information of the variable node n concerning the symbol *a*.

III. Sum product algorithm [1]

The decoding algorithm has following four stages:
Initialization.. f_n^a is used to initialize all messages q_{mn}^a passing from a variable node to a check node with probability $p(r_i|c_i = a)$.
Updating Check Node. All messages coming from the check nodes are updated with:

$$r_{mn}^a = \sum_{x':x'_{n=a}} p(z_m\,|x')\Pi_{j\in,\mathcal{N}_{m,n},} q_{mj}^{x'_j} \quad (5)$$

where $p(z_m\,|x') \in \{0,1\}$ based on $\boldsymbol{x'}$ satisfies check m or not.
Updating Bit Node. All messages coming from the variable nodes are updated with

$$q_{mn}^a = \alpha_{mn}\, f_n^a\, \Pi_{j\in\,\mathcal{M}_{n,m}}\; r_{jn}^a \quad (6)$$

Where α_{mn} is a normilzed factor chosen such that $\sum_{a=0}^{q-1} q_{mn}^a = 1$

Tentative decoding. An estimation of variable node is made with :

$$\widehat{c_n} = \underset{a}{max}\; f_n^a\;\; \Pi_{j\in\mathcal{M}_{n,m}} r_{jn}^a \quad (7)$$

If $H\widehat{c_n} = 0$ then stop. Otherwise if no. of iteration < maximum no . of iteration , loop to updating check node.
Otherwise, declare decoding failure and **stop.**

IV. Log domain decoding over *GF(Q)*

The Log domain decoding [11] has following steps:
Initialization. All messages passing form variables node to a check node are initialized with the LLR vector from the channel model.

$$\boldsymbol{\lambda_{m\leftarrow n} = \lambda_{ch}}(c_n) = \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} \quad (8)$$

Updating Check Node. All the messages coming from the check nodes are updated with:

$$\boldsymbol{\lambda}(\mathbf{H}_{\mathrm{m,n_{m,k}}}^{-1}\boldsymbol{\sigma}_{\mathrm{m,n_{m,k-1}}} + \boldsymbol{\rho}_{\mathrm{m,n_{m,k+1}}}) \quad (9)$$

The value of σ and ρ is calculated recursively.

$$\boldsymbol{\lambda}(\sigma_{m,n_{m,l-1}}) = \boldsymbol{\lambda}\,(\sigma_{m,n_{m,l-1}} + H_{m,n_{m,l}} c_{n_{m,l}}) \quad (10)$$

$$\boldsymbol{\lambda}(\rho_{m,n_{m,l}}) = \boldsymbol{\lambda}(\rho_{m,n_{m,l+1}} + H_{m,n_{m,l}} c_{n_{m,l}}) \quad (11)$$

Updating Bit Node. All messages coming from the variable nodes are updated with:

$$\boldsymbol{\lambda_{m\leftarrow n}} = \boldsymbol{\lambda_{ch}}(\mathrm{c_n}) + \sum_{\mathrm{j}\in\mathcal{M}_{\mathrm{n,m}}} \boldsymbol{\lambda}\,(\mathrm{j} \rightarrow \mathrm{n}) \quad (12)$$

Tentative decoding. An estimation of variable node is made with :

$$\widehat{c_n} = \underset{a}{argmax}(\,\lambda_{m\leftarrow n}(a)) \quad (13)$$

If $H\widehat{c_n} = 0$ then stop. Otherwise if no. of iteration < maximum no . of iteration , loop to updating check node.
Otherwise, declare decoding failure and **stop.**

V. Proposed algorithm

In this section, we develop our proposed algorithm which is known as modified log –domain decoding. Then the modified log-domain is extended to modified min-sum decoding which is less computationally complex. $p(\,c_n = a|r)$ is the channel posterior probability of a ∈ GF (q). Ratio of channel posterior probability of a ∈ GF (q) and channel posterior probability of o is defined as

$$\lambda_n\,(c_n{=}a|\boldsymbol{r}){=}log\,\frac{p(\,c_n{=}a|r)}{p(\,c_n{=}0|r)}{=}log\;\frac{p(\,c_n{=}a|r_n,\{r_i,i\neq n\})}{p(\,c_n{=}0|r_n,\{r_i,i\neq n\})} \quad (14)$$

Applying byes rule in numerator

$$p(\,c_n = a|r_n\{r_i, i \neq n\}) = \frac{p(r_n,c_n=a,\{r_i,i\neq n\})}{p(r_n,\{r_i,i\neq n\})}$$

$$= \frac{p(\,c_n{=}a|r_n\{r_i,i\neq n\})p(c_n{=}a,\{r_i,i\neq n\})}{p(r_n|\{r_i,i\neq n\})p(\{r_i,i\neq n\})}$$

$$= \frac{p(r_n|c_n=a)p(c_n=a,\{r_i,i\neq n\})}{p(r_n|\{r_i,i\neq n\})p(\{r_i,i\neq n\})}$$

$$= \frac{p(r_n|c_n=a)p(c_n=a|\{r_i,i\neq n\})}{p(r_n|\{r_i,i\neq n\})}$$

Similar way for denominator

$$p(c_n = 0|r_n\{r_i, i \neq n\}) = \frac{p(r_n|c_n=0)p(c_n=0|\{r_i,i\neq n\})}{p(r_n|\{r_i,i\neq n\})}$$

Now for (14)

$$\lambda_a(c_n|\boldsymbol{r}) = log\frac{p(r_n|c_n=a)*p(c_n=a|\{r_i,i\neq n\})}{p(r_n|c_n=0)*p(c_n=0|\{r_i,i\neq n\})}$$

$$= \underbrace{log\ \frac{p(r_n|c_n=a)}{p(r_n|c_n=0)}}_{\text{Intrinsic}} + \underbrace{log\ \frac{p(c_n=a|\{r_i,i\neq n\})}{p(c_n=0|\{r_i,i\neq n\})}}_{\text{Extrinsic}} \quad (15)$$

$\lambda_a(c_n|\boldsymbol{r})$ consists of two terms which can be divided as intrinsic term and extrinsic term. Intrinsic term being logarithmic ratio of likelihood of non-binary symbols is calculated from the channel information i.e r_n affecting the bit c_n. Likelihood of non-binary ($GF(2^q)$) symbols is calculated from binary likelihood values provided by the channel.

We define symbols likelihood values by

$$P(r_n|c_n = a) = p\,(r_n|[c_{b1}, \ldots\ldots, c_{bq}][a_1, \ldots\ldots, a_q])$$

$$= \prod_{k=1}^{q} p\,(r_n|c_{ik} = a_k) \quad (16)$$

Where $a\ \varepsilon\ GF\ (q)$ and a_i is the ith bit of the binary representation of a.

Now, (16) transforms to

$$log\frac{p(r_n|c_n=1)}{p(r_n|c_n=0)} = \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} \quad (17)$$

The extrinsic term is determined by the information provided by all the other observations and the code structure. The parity check equations for non-binary case are of following form:

$$z_m = \sum_{i\varepsilon\,\mathcal{N}_m} h_{mi}c_i \quad (18)$$

Where h_{mi} & $c_i \in GF(q)$, m is the checks number. The parity check equation for check *1* is satisfied when:

$$h_{11}c_1 + h_{12}c_2 + \cdots + h_{1n}c_n = 0 \quad (19)$$

The parity check equation for $c_i=x$ and $x\ \epsilon\ GF\ (q)$ is satisfied when:

$$h_{m1}c_1 + h_{m2}c_2 + \cdots + h_{mn}c_n = h_{mi}c_i \quad (20)$$

$$c_n \rightarrow \begin{bmatrix} q_{mn}(0) \\ q_{mn}(1) \\ . \\ . \\ . \\ q_{mn}(\alpha^{q-2}) \end{bmatrix} \alpha c_n \rightarrow \begin{bmatrix} q_{mn}(0) \\ q_{mn}(\alpha^{q-2}) \\ q_{mn}(1) \\ . \\ . \\ . \end{bmatrix} \alpha^2 c_n \rightarrow \begin{bmatrix} q_{mn}(0) \\ . \\ . \\ . \\ q_{mn}(\alpha^{q-2}) \\ q_{mn}(1) \end{bmatrix}$$

Fig. 3. Permutation of the likelihood vectors

But calculation of (20) is more complicated than binary case because now we have non-binary parity check matrix and each coded symbol has *q* likelihoods associated with it.

But this multiplication is easily accomplished by cyclically shifting downwards this column vector of likelihoods with the exception of first likelihoods. The power of primitive elements that is multiplied with the coded symbol is equal to the number of cyclic shifts. This process is known as permutation [3] Fig. 3.

This permutation transforms the parity check (20) to:

$$c_1 + c_2 + \cdots + c_n = c_i \quad (21)$$

This is more similar to binary parity check equations. When the likelihoods are cyclically shifted upwards, process is known as depermutation.

The parity check computed using the *m*th check associated with c_n, except for c_n is denoted by

$$z_{m,n} = \sum_{i\varepsilon\,\mathcal{N}_{m,n}} h_{mi}c_i \quad (22)$$

After doing permutation, (22) is transformed to:

$$z_{m,n} = \sum_{i\varepsilon\,\mathcal{N}_{m,n}} c_i \quad (23)$$

If $c_n=a$, then $z_{m,n} + c_n = 0$; that is , $z_{m,n}$ = a for all the checks $m\ \epsilon\ \mathcal{M}_n$ in which c_n participates. Now, extrinsic term in (15) is written as following way

$$log\frac{p(c_n=1|\{r_i,i\neq n\})}{p(c_n=0|\{r_i,i\neq n\})} = log\frac{p(z_{m,n}=a\ for\ all\ m\ \epsilon\ \mathcal{M}n|\{r_i,i\neq n\})}{p(z_{m,n}=0\ for\ all\ m\ \epsilon\ \mathcal{M}n|\{r_i,i\neq n\})} \quad (24)$$

These two sets of bits are conditionally independent to each other

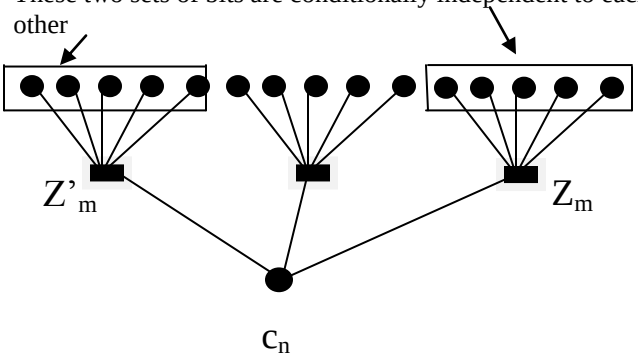

Fig. 4. Conditional independence among the set of bits. [21]

If the graph associated with the code is cycle free then the set of bits associated with $z_{m,n}$ are independent of the bits associated with $z_{m,n'}$ for $m' \neq m$ Fig. 4. Now (24) becomes

$$\log \frac{p(c_n=a|\{r_i,i\neq n\})}{p(c_n=0|\{r_i,i\neq n\})} = \log \frac{\prod_{m\in\mathcal{M}n} P(z_{m,n}=a|\{r_i,i\neq n\})}{\prod_{m\in\mathcal{M}n} P(z_{m,n}=0|\{r_i,i\neq n\})}$$

$$= \sum_{m\in\mathcal{M}n} \log \frac{P(z_{m,n}=a|\{r_i,i\neq n\})}{P(z_{m,n}=0|\{r_i,i\neq n\})} \quad (25)$$

Let, Log likelihood ratio can be defined as

$$\lambda(z_{m,n}=a|\{r_i,i\neq n\}) = \frac{P(z_{m,n}=a|\{r_i,i\neq n\})}{P(z_{m,n}=0|\{r_i,i\neq n\})} \quad (26)$$

Then from (25)

$$\sum_{m\in\mathcal{M}n} \lambda(z_{m,n}|\{r_i,i\neq n\}) = \sum_{m\in\mathcal{M}n} \lambda(\sum_{n\in\mathcal{N}_{m,n}} c_n|\{r_i,i\neq n\}) \quad (27)$$

From (15), (17) and (27), we can write

$$\lambda_n(c_n=a|\mathbf{r}) = \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} + \sum_{m\in\mathcal{M}n} \lambda(z_{m,n}|\{r_i,i\neq n\})$$

$$= \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} + \sum_{m\in\mathcal{M}n} \lambda(\sum_{n\in\mathcal{N}_{m,n}} c_n|\{r_i,i\neq n\}) \quad (28)$$

Let

$$\eta_{m,n} = \lambda(\sum_{n\in\mathcal{N}_{m,n}} c_n|\{r_i,i\neq n\}) \quad (29)$$

This is the message which is passed from the check node m to the bit *node* n. Now (28) can be written as

$$\lambda_n(c_n=a|\mathbf{r}) = \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} + \sum_{m\in\mathcal{M}n} \eta_{m,n} \quad (30)$$

It is the message which is passed from the bit node n to check node m. Now we can employ iterative decoding between (29) and (30). The different steps of our proposed algorithms is presented below.

A. Modified Log domain decoding

Modified log domain decoding uses log likelihood ratio as message and it performs the following operations:

Initialization:

Initialize $\boldsymbol{\eta}_{m,n}^{[0]} = 0$ for all (m,n) with $H(m,n) \neq 0$.

And $$\boldsymbol{\lambda}_n^{[0]} = \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} \quad (31)$$

Check node update: All the messages coming from the check nodes are updated with:

$$\boldsymbol{\eta}_{m,n} = \boldsymbol{\lambda}(\sum_{n\,\varepsilon\mathcal{N}_{m,n}} c_n|\{r_i, i\neq n\}) \quad (32)$$

Bit node Update: All messages coming from the variable nodes are updated with:

$$\boldsymbol{\lambda}_n(a) = \boldsymbol{\lambda}_n^{[0]}(a) + \sum_{m\varepsilon\mathcal{M}n} \boldsymbol{\eta}_{m,n} \quad (33)$$

Tentative decoding : tentative decision *of* $\widehat{c_n}$ *:*

$$\widehat{\boldsymbol{c_n}} = \underset{a}{argmax}\ (\boldsymbol{\lambda_n}(\boldsymbol{a})) \quad (34)$$

If $H\widehat{\boldsymbol{c_n}} = \boldsymbol{0}$ then stop. Otherwise if no. of iteration < maximum no . of iteration , loop to check node update.
Otherwise, declare decoding failure and **stop.**

B. Modified min sum decoding

The above modified log domain algorithm can be converted to a min sum algorithm which requires less operations according to procedure presented in [20].

Initialization:

Initialize $\boldsymbol{\eta}_{m,n}^{[0]} = 0$ for all (m,n) with $H(m,n) \neq 0$.

And $$\boldsymbol{\alpha_{m,n}} = \boldsymbol{\lambda_n} = \sum_{j:\psi^{-1}(a_i)_j=+1} \frac{2r_{nb+j}}{\sigma^2} \quad (35)$$

Check node update: All messages coming from the variable nodes are updated with:

$$\boldsymbol{\beta_{m,n}} = \min_{\substack{(a_{n'})n'\in\mathcal{N}_m \\ \in L(m|a_n=a)}} (\sum_{n'\in\mathcal{N}_{m,n}} c_n|\{r_i, i\neq n\}) \quad (36)$$

The check node update process can be implemented efficiently by recursive method using forward and back ward matrix.

Forward matrix:

$$F_1(a) = \alpha_{m,n_1}\left(h_{m,n_1}^{-1} a\right)$$

$$F_i(a) = \min_{\substack{a',a'' \in GF(q) \\ a'+h_{m,n_i}a''=a}} (F_{(i-1)}(a'), \alpha_{m,n_i}(a'')) \quad (37)$$

Backward matrix:

$$B_d(a) = \alpha_{m,n_d}\left(h_{m,n_d}^{-1} a\right)$$

$$B_i(a) = \min_{\substack{a',a'' \in GF(q) \\ a'+h_{m,n_i}a''=a}} (B_{(i+1)}(a'), \alpha_{m,n_i}(a'')) \quad (38)$$

Check node messages can be computed as follows:

$$\beta_{m,n_1}(a) = B_2(a) \;\&\; \beta_{m,n_d}(a) = F_{d-1}(a)$$

$$\beta_{m,n_i}(a) = \min_{\substack{a',a'' \in GF(q) \\ a'+a''=-h_{m,n_i}a}} (F_{(i-1)}(a'), B_{(i+1)}(a'')) \quad (39)$$

Bit node update: All messages coming from the variable nodes are updated with:

$$\alpha'_{m,n} = \lambda_n + \sum_{m' \varepsilon \mathcal{M}n,m} \beta_{m'n} \quad (40)$$

$$\alpha'_{m,n} = \min_{a \in GF(q)} \alpha'_{m,n} \quad (41)$$

$$\alpha_{m,n} = \alpha'_{m,n} - \alpha'_{m,n} \quad (42)$$

Tentative decoding:

$$\widehat{\lambda_n} = \lambda_n + \sum_{m \varepsilon \mathcal{M}n} \beta_{m,n} \quad (43)$$

Tentative decision of $\widehat{c_n}$:

$$\widehat{c_n} = \arg\max (\widehat{\lambda_n}(a)) \quad (44)$$

If $H\widehat{c_n} = 0$ then stop. Otherwise if no. of iteration < maximum no . of iteration , loop to check node update. Otherwise, declare decoding failure and **stop.**

VI. Result

In this section we compare three decoding algorithms for NB-LDPC: the Log domain decoding algorithm [11], modified log –domain decoding algorithm and min- sum decoding algorithm as discussed above.

Fig. 5 shows the Bit Error Rate (BER) performance for a rate ½ LDPC code from [18] over GF (8) with N=204 and BPSK modulation. The decoding process is halted after a maximum 200 iterations. The BER performance of modified log domain decoding and modified min-sum decoding is better than log domain decoding algorithms. But the BER performance of modified min-sum decoding is 0.2 dB less than the modified log domain as expected because modified min-sum is approximation of modified log domain decoding. The memory requirement is less in modified log domain but the computation complexity is much in modified log domain. The modified min sum requires less no. of iterations to converge.

VII. Conclusion

We have introduced modified log-domain algorithm and min-sum algorithm of sum-product algorithm (SPA) for LDPC codes over GF (q). A log-domain implementation has several advantages as far as practical implementation is concerned.

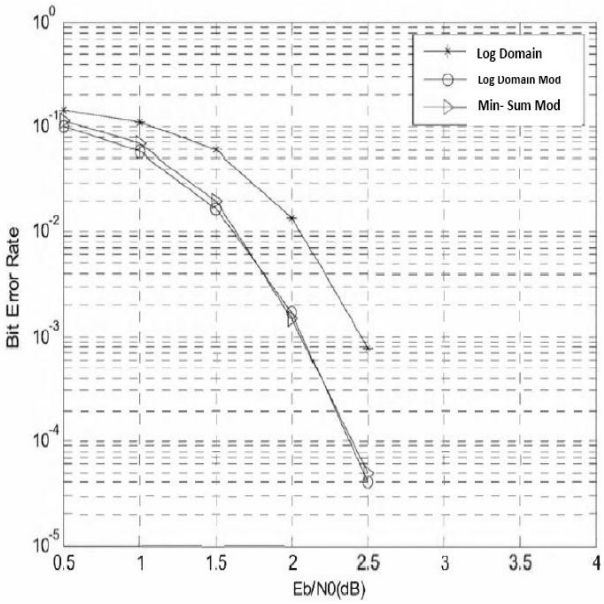

Fig. 5. BER performance for an LDPC code over *GF* (8)

We have compared the BER performance and computational complexity of the log domain algorithm, modified log-domain algorithm and modified min-sum algorithm. The modified log–domain algorithm has superior BER performance than modified min-sum algorithm for small SNR as verified by the computer simulations. It is expected because modified min-sum is the approximation of modified log-domain. Modified min –sum gives rise to a small BER degradation. Both modified log domain algorithm and modified min-sum algorithm requires no message multiplications. So they may constitute a considerable saving in computational complexity as compared to the SPA. There is a scope of finding the performance of the proposed algorithm in terms of extrinsic information transfer chart (EXIT) and density evolution.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA:MIT Press, 1963.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity- approaching low-density parity check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001

[3] Barnault, L. and Declerq, D. (2003) Fast decoding algorithm for LDPC over GF(2q). IEEE Information Theory Workshop, Paris, France, pp. 70–3E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.

[4] M. G. Luby,M.Mitzenmacher,M.A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[5] 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 Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[6] Voicila, A.; Declercq, D.; Verdier, F.; Fossorier, M.; Urard, P.; , "Low-complexity decoding for non-binary LDPC codes in high order fields," *Communications, IEEE Transactions on* , vol.58, no.5, pp.1365-1375, May 2010

[7] M. Davey and D.J.C. MacKay, "Low Density Parity Check Codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, pp. 165-167,June 1998.

[8] X.-Y. Hu and E. Eleftheriou, “Binary Representation of Cycle Tanner-Graph GF(2q) Codes,” *The Proc. IEEE Intern. Conf.on Commun.*, Paris, France, pp. 528-532, June 2004.

[9] C. Poulliat,M. Fossorier and D. Declercq, “Design of non binary LDPC codes using their binary image: algebraic properties,”*ISIT’06*, Seattle, USA, July 2006.

[10] A. Bennatan and David Burshtein, ”Design and Analysis of Nonbinary LDPC Codes for Arbitrary Discrete-Memoryless Channels,” *IEEE Trans. on Inform. Theory*, vol. 52, no. 2, pp. 549-583, Feb. 2006.

[11] H. Wymeersch, H. Steendam and M. Moeneclaey, ”Log-domain decoding of LDPC codes over GF(q),” *Proc. IEEE Intl. Conf. on Commun.*, pp. 772-776, Paris, France, Jun. 2004.

[12] Spagnol, C.; Popovici, E.M.; Marnane, W.P.; , "Hardware Implementation of ${rm GF}(2^{m})$ LDPC Decoders," *Circuits and Systems I: Regular Papers, IEEE Transactions on* , vol.56, no.12, pp.2609-2620, Dec. 2009.

[13] H. Song and J. R. Cruz, “Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording,” *IEEE Trans. Magn.*, vol. 39, no. 3, pp. 1081–1087, Mar. 2003.

[14] D. Declercq, M. Fossorier, ”Decoding algorithms for nonbinary LDPC codes over GF(q)”, *IEEE Trans. on Commun.*, vol. 55, no. 4, pp. 633-643, Apr. 2007.

[15] V. Savin,“Min-Max decoding for non binary LDPC codes,” *Proc. IEEE Intl. Symp. on Info. Theory*, Toronto, Canada, Jul. 2008.

[16] L. Ping and W.K. Leung, “Decoding low density parity check codes with finite quantization bits”, *IEEE Commun. Lett.*,4(2):pp.62-64, February 2000.

[17] S. ten Brink, J. speidel and J. C. Yan. “Iterative demapping and decoding for multilevel modulation”. In *IEEE GLOBECOME’98*, 1998.

[18] D.J.C. MacKay. "Online database of low-density parity check codes". http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[19] [T. k. Moon “Error Correction Coding- Mathematical Methods and Algorithms”, wiley publications.

[20] D. Declercq and M. Fossorier, “Extended min-sum algorithm for decoding LDPC codes over GF(*q*),” in Information Theory, 2005. ISIT 2005.Proceedings. International Symposium on, 2005, pp. 464–468.

[21] R. A. Carrasco , M. Johnston ,” Non binary error control coding for wireless communication and data storage” Wiley publications”