Bit Reliability-Based Decoders for Non-Binary LDPC Codes

Qin Huang and Shuai Yuan

***Abstract*—Message-passing decoders typically perform well for nonbinary low-density parity-check (NB-LDPC) codes with large computational complexity. As another type of simplified decoders, symbol-reliability-based decoders further reduce the computational complexity. However, the previously proposed algorithms suffer severe error performance degradation for NB-LDPC codes with low column weights. In this paper, a weighted bit-reliability based (wBRB) decoder for NB-LDPC codes is developed and implemented with efficient layered partial-parallel structure. It not only balances the tradeoff between complexity and error performance, but also reduces the memory usage significantly. Furthermore, to enhance the performance of the wBRB decoder, a full bit-reliability-based (FBRB) decoder is proposed. The FBRB decoder is derived based on the binary matrix representation of the nonzero entries in the parity-check matrix. Since more bit-reliability values are passed through the edges of the Tanner graph, the FBRB decoder can achieve better error performance and faster convergence rate than the wBRB decoder. Both of the decoders are implemented on a Xilinx Virtex-5 XC5VLX155T FPGA device for a (403,226) code over GF(2^5). The results shows that they achieve 118.98 and 95.73 Mbps throughput with 15 iterations, respectively.**

***Index Terms*—Non-binary LDPC codes, reliability-based decoding, soft bit-reliability, hard-reliability, memory consumption.**

I. Introduction

BINARY *low-density parity-check* (LDPC) codes were first discovered by Gallager in 1962 [1], and re-gained interests in late 1990's [2]–[4]. In the 2000's, researchers found that *non-binary LDPC* (NB-LDPC) codes [5]–[7] over *Galois fields* (GF) could provide better error performance than their binary counterparts. However, decoding for NB-LDPC codes over GF(q) requires substantial computational complexity and memory consumption by using the q-ary *sum-product algorithm* (QSPA) [8], [9], which is an obstacle for the application of NB-LDPC codes. For example, a min-max partial-parallel decoder with 9.3 Mbps throughput for a (744,653) NB-LDPC code over GF(2^5) requires 47341 slices and 180 *block RAMs* (BRAMs) on a Xilinx Virtex-II Pro FPGA device [10], while a binary partial-parallel decoder with 28 Mbps throughput for a similar binary LDPC code only takes 2430 slices and 70 BRAMs on a Xilinx XC2V8000 device [11]. To resolve this issue, extensive research studies have been conducted in the past decade. As a result, two classes of simplified algorithms and their hardware architectures have been proposed to reduce the complexity of decoding NB-LDPC codes.

The first class of simplified algorithms is based on message-passing decoding, including the extended min-sum (EMS) algorithm [12], the min-max algorithm [13], the trellis based EMS (T-EMS) algorithm [14], the trellis based min-max (T-MM) algorithm [15], and the Max-Log-QSPA algorithm [16]. By truncating q messages for one symbol into the n_m most reliable messages, the complexity of EMS and min-max is determined by n_m rather than q. By truncating the trellis into n_c reliable paths, the complexity of T-EMS and T-MM is determined by n_c rather than q. It has been reported that many efficient hardware designs [10], [15]–[17] have incorporated these algorithms. In [17], messages were compressed during message-passing to reduce the complexity of T-MM decoder. In [15] and [16], the forward-backward operations were efficiently implemented such that both area and latency of non-binary decoders were improved. However, the error performance of these algorithms degrades as n_m or n_c decreases, limiting further reduction of the computational complexity.

The second class of simplified algorithms is derived from symbol-reliability based *majority logic decoding* (MLgD), including the *iterative soft-reliability based* (ISRB) algorithm [18] and its variations such as the *improved ISRB* (IISRB) [19], the *iterative hard-reliability based* (IHRB) [18] and the *enhanced IHRB* (E-IHRB) algorithms [20]. Since these algorithms pass only one symbol-reliability value through the edge of the Tanner graphs for LDPC codes, the complexity is significantly reduced. For example, an IHRB partial-parallel decoder with 90.7 Mbps throughput for a (403,226) NB-LDPC code over GF(2^5) costs only 7841 LUTs and 529 registers on a Xilinx Virtex-5 XC5VLX200T device [20]. Moreover, their memory consumptions are less than those of EMS or min-max decoders, though still determined by q. For instance, the above IHRB decoder takes 56 BRAMs, and the min-max decoder for the same code takes 95 BRAMs [20]. However, these symbol-reliability based decoders suffer significant performance degradation, more than 1dB, when applying for NB-LDPC codes with small column weights.

In [21], Huang et al. proposed a *weighted bit-reliability based* (wBRB) MLgD algorithm for NB-LDPC codes, which balanced the trade-off between complexity and error performance. The wBRB algorithm passes bit-reliability value through each edge of the Tanner graph for NB-LDPC codes, which is more

Manuscript received March 7, 2015; revised July 18, 2015 and October 2, 2015; accepted November 10, 2015. Date of publication November 17, 2015; date of current version January 14, 2016. This work was supported by NSAF under Grant U1530117, and National Natural Science Foundation of China under Grant 61201156. The associate editor coordinating the review of this paper and approving it for publication was D. Declercq. *(Corresponding author: Shuai Yuan.)*

Qin Huang is with the Qian Xuesen Laboratory of Space Technology, China Academy of Space Technology, Beijing 100094, China, and also with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: qhuang.smash@gmail.com).

S. Yuan is with the Qian Xuesen Laboratory of Space Technology, China Academy of Space Technology, Beijing 100094, China (e-mail: yuanshuai@qxslab.cn).

Digital Object Identifier 10.1109/TCOMM.2015.2501298

efficient than symbol-reliability based MLgD algorithms. Thus, it performs closely well compared to the EMS algorithm for NB-LDPC codes with small column weights, that is less than 0.3 dB. On the other hand, its computational complexity and memory consumption are similar to or even less than soft symbol-reliability based algorithms. This paper investigates the architecture design and hardware implementation of the partial-parallel wBRB decoder for the first time. First, this paper organizes the wBRB algorithm into a novel format such that it can be implemented with an efficient layered structure. As a result, it is possible to overlap the CN updates and the VN updates of the wBRB decoder that leads to a higher throughput of decoding and higher efficient memory usage under the layered structure. Second, the use of all bit-reliability values of a symbol rather than the minimum bit-reliability value is utilized by introducing matrix representations. The enhanced algorithm, called *full bit-reliability based* (FBRB) algorithm, outperforms the wBRB algorithm about 0.3 dB and is able to achieve higher throughput in terms of average iterations. As demonstrated in this paper, the wBRB and the FBRB decoders for a (403,226) NB-LDPC code take 10517 and 36147 slice LUTs, respectively, by using a Xilinx Virtex-5 XC5VLX155T device. The wBRB decoder achieves 118.98 Mbps while the FBRB decoder achieves 95.73 Mbps throughputs with fixed 15 iterations. In addition, they achieve 220.46 Mbps and 395.8 Mbps throughputs, respectively, with the average iterations of E_b/N_0 at 4.5 dB. It is worth mentioning that each only requires 40 and 49 BRAMs, which are much less than those of the existing min-max decoder and the hard-decision decoder IHRB.

The rest of this paper is organized as follows. Section II describes the wBRB algorithm. The layered wBRB and the FBRB algorithms are given in Section III and IV, respectively. Section V then presents the hardware implementation of the two algorithms. Finally, Section VI summarizes this paper.

II. Background

A. Quasi-Cyclic NB-LDPC Codes

Consider an NB-LDPC code $\mathcal{C}$ given by the null space of an $m \times n$ *sparse* parity-check matrix $\mathbf{H} = [h_{i,j}]$, $0 \le i < m$ and $0 \le j < n$, over GF(q). Without loss of generality, assume $q = 2^r$. If the parity-check matrix $\mathbf{H}$ consists of zero matrices and shifted identity matrices, the NB-LDPC code is considered *quasi-cyclic* (QC) [22]. Due to the regularity of their parity-check matrices, QCNB-LDPC decoders with partial-parallel structure are very efficient in hardware implementation [10], [23], [24]. Thus, most research studies focus on implementation of NB-LDPC codes decoding by utilizing QC decoders.

B. wBRB Decoding Algorithm

The decoding of an NB-LDPC code is based on its Tanner graph, which is a bipartite graph consisting of two types of nodes, variable nodes (VNs) and check nodes (CNs). If the entry $h_{i,j}$ in the parity-check matrix $\mathbf{H}$ is nonzero, there exists an edge-(i, j) that connects the i-th check node CN_i and the j-th variable node VN_j in the Tanner graph. The index sets of nonzero entries in the i-th row $\mathbf{h}_{i,*}$ and in the j-th column $\mathbf{h}_{*,j}$ are denoted as $M_j = [i : h_{i,j} \neq 0, 0 \le i < m]$ and $N_i = [j : h_{i,j} \neq 0, 0 \le j < n]$, respectively.

Algorithm A. The wBRB decoding algorithm for non-binary LDPC codes

Initialization: $R_{j,t}^{(0)} = q_{j,t}$; $E_{j\to i,t}^{(0)} = R_{j,t}^{(0)}$; $z_{j,t}^{(0)} = (R_{j,t}^{(0)} < 0)?1 : 0$, where $0 \le i < m$, $0 \le j < n$ and $0 \le t < r$; set $\{\theta_0, \theta_1, \ldots, \theta_r\}$ for $\Theta(\cdot)$.

Iteration:

for $k = 0 : I_{max}$

Stop iteration, if $\mathbf{z}^{(k)} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{0}$ or $k = I_{max}$

for $i = 0 : m-1$

for $j \in N_i$

A1: $\boldsymbol{\sigma}_{i,j}^{(k)} = h_{i,j}^{-1} \sum_{j' \in N_i \setminus j} h_{i,j'} \cdot e_{j'\to i}^{(k)}$

A2: $\phi_{i,j}^{(k)} = \min_{j' \in N_i \setminus j} \min_{t} |E_{j'\to i,t}^{(k)}|$

for $t = 0 : r-1$

A3: $R_{i\to j,t}^{(k)} = \begin{cases} \Theta(H(\mathbf{z}_j^{(k)}, \boldsymbol{\sigma}_{i,j}^{(k)})) \cdot \phi_{i,j}^{(k)} b_{i,j,t}^{(k)} = 0 \\ -\Theta(H(\mathbf{z}_j^{(k)}, \boldsymbol{\sigma}_{i,j}^{(k)})) \cdot \phi_{i,j}^{(k)} b_{i,j,t}^{(k)} = 1 \end{cases}$

for $j = 0 : n-1$

for $i \in M_j$

A4: $\mathbf{R}_j^{(k+1)} = \mathbf{R}_j^{(0)} + \sum_{i \in M_j} \mathbf{R}_{i\to j}^{(k)}$

A5: $\mathbf{E}_{j\to i}^{(k+1)} = \mathbf{R}_j^{(k+1)} - \mathbf{R}_{i\to j}^{(k)}$

for $t = 0 : r-1$

A6: $z_{j,t}^{(k+1)} = (R_{j,t}^{(k+1)} < 0)?1 : 0$

The wBRB decoding algorithm [21] is an iterative algorithm that is based on bit-reliability updates of CNs and VNs. For the k-th iteration, the r bit-reliability values of the j-th codeword symbol, $0 \le j < n$, are denoted by $\mathbf{R}_j^{(k)} \triangleq [R_{j,0}^{(k)}, R_{j,1}^{(k)}, \ldots, R_{j,r-1}^{(k)}]$ and the hard-decision vector of this symbol is denoted by $\mathbf{z}_j^{(k)} \triangleq [z_{j,0}^{(k)}, z_{j,1}^{(k)}, \ldots, z_{j,r-1}^{(k)}]$. The extrinsic information passed from the j-th VN_j to the i-th CN_i, $i \in M_j$, is denoted by $\mathbf{E}_{j\to i}^{(k)} \triangleq [E_{j\to i,0}^{(k)}, E_{j\to i,1}^{(k)}, \ldots, E_{j\to i,r-1}^{(k)}]$. The wBRB decoding algorithm for NB-LDPC codes is described in Algorithm A.

At the beginning of the decoding, $R_{j,t}^{(0)}$ is initialized with the channel information $q_{j,t}$ and $E_{j\to i,t}^{(0)}$ is initialized with $R_{j,t}^{(0)}$, $t = 0, 1, \ldots, r-1$. Each iteration contains two phases, CN updates and VN updates.

The CN updates include three steps - A1, A2 and A3, as shown in Fig. 1. For CN_i, assume the vector, $[b_{i,j,0}^{(k)}, b_{i,j,1}^{(k)}, \ldots, b_{i,j,r-1}^{(k)}]$, is the binary representation of $\boldsymbol{\sigma}_{i,j}^{(k)}$, A1 computes its extrinsic information-sum $\boldsymbol{\sigma}_{i,j}^{(k)}$ from $e_{j\to i}^{(k)}$- hard-decision of $\mathbf{E}_{j\to i}^{(k)}$, $j \in N_i$; A2 calculates its *soft bit-reliability* $\phi_{i,j}^{(k)}$ from the minimum absolute value of extrinsic bit-reliability values of $\mathbf{E}_{j\to i}^{(k)}$; Step A3 computes the message vector $\mathbf{R}_{i\to j}^{(k)} = [R_{i\to j,0}^{(k)}, R_{i\to j,0}^{(k)}, \ldots, R_{i\to j,r-1}^{(k)}]$

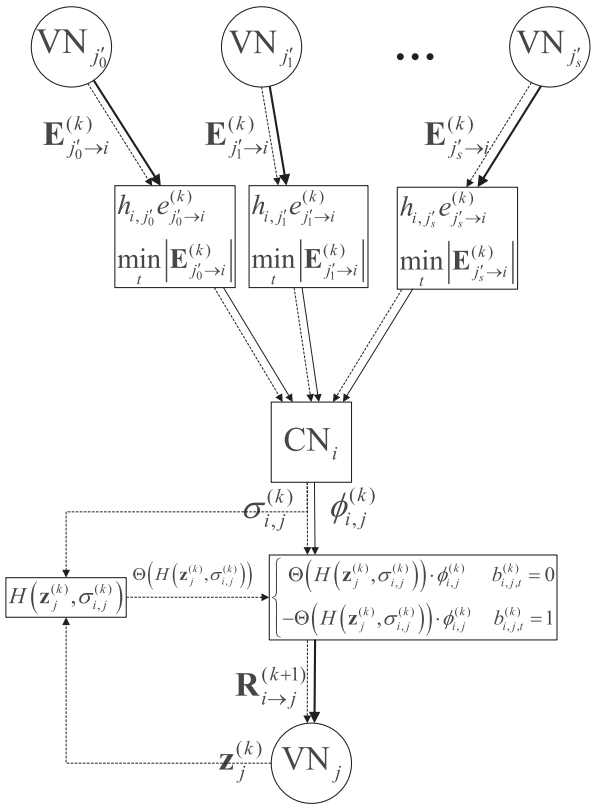

Fig. 1. CN update of wBRB.

passed from CN$_i$ to VN$_j$ (c-to-v). The signs of r c-to-v messages are determined by the corresponding extrinsic information-sum $\boldsymbol{\sigma}^{(k)}_{i,j}$. All the r absolute values are assigned with the product of the soft bit-reliability $\phi^{(k)}_{i,j}$ and a weighting factor $\Theta(H(\mathbf{z}^{(k)}_j, \boldsymbol{\sigma}^{(k)}_{i,j}))$, where $H(\mathbf{z}^{(k)}_j, \boldsymbol{\sigma}^{(k)}_{i,j})$ denotes the Hamming distance between the extrinsic information-sum $\boldsymbol{\sigma}^{(k)}_{i,j}$ and the hard-decision of the j-th symbol $\mathbf{z}^{(k)}_j$, $H(\mathbf{z}^{(k)}_j, \boldsymbol{\sigma}^{(k)}_{i,j}) = 0, 1, \ldots, r$. To distinguish it with soft bit-reliability $\phi^{(k)}_{i,j}$, we call the weighting factor $\Theta(H(\mathbf{z}^{(k)}_j, \boldsymbol{\sigma}^{(k)}_{i,j}))$ *hard-reliability*, where $\Theta(u) = \theta_u (u \in \{0, 1, \ldots, r\}, \theta_u \in \mathbb{R})$. The smaller $H(\mathbf{z}^{(k)}_j, \boldsymbol{\sigma}^{(k)}_{i,j})$ is, the more reliable $\boldsymbol{\sigma}^{(k)}_{i,j}$ is, the larger $\Theta(H(\mathbf{z}^{(k)}_j, \boldsymbol{\sigma}^{(k)}_{i,j}))$ is.

The VN updates also include three steps - A4, A5 and A6. For VN$_j$, A4 updates the bit-reliability of the codeword symbol $\mathbf{R}^{(k+1)}_j$ with the initial reliability $\mathbf{R}^{(0)}_j$ and the corresponding c-to-v messages $\mathbf{R}^{(k)}_{i\to j}$'s, $i \in M_j$; A5 computes its extrinsic information $\mathbf{E}^{(k+1)}_{j\to i}$ for the next iteration with the difference of $\mathbf{R}^{(k+1)}_j$ and $\mathbf{R}^{(k)}_{i\to j}$; A6 calculates the new hard-decision $\mathbf{z}^{(k+1)}_j$.

III. Layered wBRB Decoder

Similar to the traditional min-sum decoder, the wBRB decoder faces two challenges in hardware implementation. First, the CN updates and the VN updates cannot be processed at the same time, otherwise the efficiency of the decoder will be sacrificed. On one hand, the CN$_i$ update must be processed after all VN$_j$ ($j \in N_i$) updates are completed as new extrinsic information $\mathbf{E}^{(k)}_{j\to i}$'s from VN$_j$'s are needed. On the other hand, the VN$_j$ update must be processed after the completion of all CN$_i$'s ($i \in M_j$) updates as new c-to-v messages $\mathbf{R}^{(k)}_{i\to j}$'s are needed. Second, the *cross-addressing* problem of RAM leads to more BRAM consumption when the partial-parallel decoder is implemented in an FPGA device. Typically, LDPC decoders use BRAMs in FPGAs to store the exchange information between CNs and VNs. Exchanges RAMs are addressed by the rows of the parity-check matrix $\mathbf{H}$ in the CN updates, while in the VN updates they are addressed by the columns. As a result, the depths of the BRAMs, i.e. the numbers of stored words, are superficial such that a large number of BRAMs are required. In order to solve the above two problems, the wBRB algorithm is re-organized into layered structure as inspired by the layered binary LDPC decoders [25]–[27].

From the view of a layered structure, all the rows in a parity-check matrix can be divided into L layers, $\mathbf{H} = \{\mathbf{H}_l\}$, $l = 0, 1, \ldots, L-1$, in which the column weight of any $\mathbf{H}_l$ is no more than one. The l-th layer has τ_l rows and their indices in $\mathbf{H}$ are $[m_{l,0}, m_{l,2}, \ldots, m_{l,\tau_l-1}]$. Consequently, the layered decoder can be considered as a serial concatenation of L sub-decoders based on $\mathbf{H}_l$.

In order to transform the wBRB algorithm into a layered structure, the update of the bit-reliability value $\mathbf{R}^{(k)}_j$ is decomposed into L layers. Designate the bit-reliability value in the l-th layer of k-th iteration as $\mathbf{R}^{(l,k)}_j = [R^{(l,k)}_{j,0}, R^{(l,k)}_{j,1}, \ldots, R^{(l,k)}_{j,r-1}]$. $\mathbf{E}^{(k)}_{j\to i}$ in the l-th sub-decoder is computed by $\mathbf{R}^{(l,k)}_j$ from the $(l-1)$-th sub-decoder and the previous $\mathbf{R}^{(k)}_{i\to j}$ from the $(k-1)$-th iteration:

$$\mathbf{E}^{(k)}_{j\to i} = \mathbf{R}^{(l,k)}_j - \mathbf{R}^{(k)}_{i\to j}, \tag{1}$$

where $i \in [m_{l,0}, m_{l,2}, \ldots, m_{l,\tau_l-1}]$ and $j \in N_j$. Then $\mathbf{R}^{(l+1,k)}_j$ is updated from $\mathbf{E}^{(k)}_{j\to i}$ and the new $\mathbf{R}^{(k+1)}_{i\to j}$ to the $(l+1)$-th sub-decoder,

$$\mathbf{R}^{(l+1,k)}_j = \mathbf{E}^{(k)}_{j\to i} + \mathbf{R}^{(k+1)}_{i\to j}. \tag{2}$$

Thus, the reliability value $\mathbf{R}^{(l,k)}_j$ can be updated in each layer. Different from the original Algorithm A, the proposed algorithm does not need to wait for all the CNs to finish their reliability update. As a result, in the layered decoder, the CN updates and the VN updates can work simultaneously as long as the processes of the adjacent two layers are overlapped. In addition, there is no need to store the extrinsic information $\mathbf{E}^{(k)}_{j\to i}$'s as they are needed only in one layer. Therefore, the layered decoder no longer has the cross-addressing problem.

IV. Full Bit-Reliability Decoding Algorithm

Different from binary LDPC codes, entries in the parity-check matrix $\mathbf{H}$ for an NB-LDPC code are symbols over GF(2^r). It is clear that nonzero entries have various impacts on both the hard-reliability and the soft bit-reliability for symbols that pass through the edges of the Tanner graph. For simplicity, the wBRB algorithm only considers the impacts on the

hard-reliability and uses the minimum absolute value of the r bit-reliability values for one symbol. Thus, it compromises the performance to some extent.

In order to further improve the performance, the following section considers the nonzero entries over GF(2^r) as their binary matrix representations. A new algorithm, called FBRB algorithm, is derived from these $r \times r$ matrices over GF(2), which can keep all the r bit-reliability values for one symbol in reliability updates.

A. CN Update With Matrix Representations

In this subsection, the impact of the matrix representations of nonzero entries in $\mathbf{H}$ on bit-reliability updates is analyzed. Consider GF(2^r) given by the primitive polynomial $p(x) = p_0 + p_1 x + \ldots + p_{r-1}x^{r-1} + x^r$. The companion matrix of $p(x)$ is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \ldots & 0 & p_0 \\ 1 & 0 & \ldots & 0 & p_1 \\ 0 & 1 & \ldots & 0 & p_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & p_{r-1} \end{bmatrix}. \tag{3}$$

As demonstrated in [28], [29], $\mathbf{A}$ is the matrix representation of the primitive element α in GF(2^r). The power of $\mathbf{A}$ represents the nonzero elements in this field, i.e., $\mathbf{A}^k$ is the matrix representation of α^k, where $k = 0, 1, \ldots, 2^r - 2$. As a result, each nonzero element in GF(2^r) can be represented as a power of $\mathbf{A}$, while all the nonzero elements in GF(2^r) can be represented as a $r \times r$ matrix over GF(2).

Consider a codeword $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$. The following exists for the i-th parity-check equation of $\mathbf{H}$:

$$\sum_{j \in N_i} h_{i,j} \cdot x_j = 0, \tag{4}$$

which can be described by the vector representations of codeword symbols x_j's and matrix representations of $h_{i,j}$'s, $j \in N_i$. Denote $\mathbf{x}_j = [x_{j,0}, x_{j,1}, \ldots, x_{j,r-1}]$ as the vector representation of the j-th codeword symbol, $j \in N_i$, and $\mathbf{A}^{i,j} = [A^{i,j}_{s,t}]$ as the matrix representation of $h_{i,j}$, $0 \le s < r$, $0 \le t < r$. The index sets of non-zero entries in the s-th row $\mathbf{A}^{i,j}_{s,*}$ and in the t-th column $\mathbf{A}^{i,j}_{*,t}$ are designated as $T^{i,j}_s = [t : A^{i,j}_{s,t} = 1, 0 \le s < r, j \in N_i]$ and $S^{i,j}_t = [s : A^{i,j}_{s,t} = 1, 0 \le t < r, j \in N_i]$. Then equation (4) can be rewritten as

$$\sum_{j \in N_i} \mathbf{A}^{i,j} \cdot \mathbf{x}_j^{\mathrm{T}} = \mathbf{0}. \tag{5}$$

Expanding (5), we have

$$\begin{cases} \sum_{j \in N_i} \mathbf{A}^{i,j}_{0,*} \cdot \mathbf{x}_j^{\mathrm{T}} = 0 \\ \sum_{j \in N_i} \mathbf{A}^{i,j}_{1,*} \cdot \mathbf{x}_j^{\mathrm{T}} = 0 \\ \qquad \vdots \\ \sum_{j \in N_i} \mathbf{A}^{i,j}_{r-1,*} \cdot \mathbf{x}_j^{\mathrm{T}} = 0 \end{cases} \tag{6}$$

Based on the r parity check equations over GF(2), the CN$_i$ of $\mathbf{H}$ consists of r binary sub-CNs. VN$_j$'s are also decomposed into r sub-VNs, $j \in N_i$. Similar to the binary min-sum algorithm [22], these binary sub-CNs and sub-VNs work as follows.

If $A^{i,j}_{s,t} = 1$, the soft bit-reliability $\hat{\phi}^k_{i,j,(s,t)}$ from the s-th sub-CN of CN$_i$ to the t-th sub-VN of VN$_j$ is

$$\hat{\phi}^{(k)}_{i,j,(s,t)} = \prod_{\substack{j' \in N_i, t' \in T^{i,j'}_s \\ j' \neq j \text{ or } t' \neq t}} \operatorname{sign}(E^{(k)}_{j' \to i,t'}) \cdot \min_{\substack{j' \in N_i, t' \in T^{i,j'}_s \\ j' \neq j \text{ or } t' \neq t}} \left| E^{(k)}_{j' \to i,t'} \right|. \tag{7}$$

Then, the overall soft bit-reliability values $\phi^{(k)}_{i,j,t}$ from the r sub-CNs to the t-th sub-VN of VN$_j$ are the summation of the above messages,

$$\phi^{(k)}_{i,j,t} = \sum_{s \in S^{i,j}_t} \hat{\phi}^{(k)}_{i,j,(s,t)}. \tag{8}$$

However, the r sub-VNs of VN$_j$ may connect to various number of sub-CNs. Thus, averaging is helpful to keep numerically stable

$$\phi^{(k)}_{i,j,t} = \text{average}\left\{ \hat{\phi}^{(k)}_{i,j,(s,t)} : s \in S^{i,j}_t \right\}. \tag{9}$$

In some simplified situation, we can set attenuate factors instead of averaging with little performance loss.

The above CN updates are different from those of wBRB, which allocates the same minimum bit-reliability value to all r bit-reliability values of a symbol. The updates method using matrix representations pass r bit-reliability values of each symbol via the nonzero entries in the matrix representations. In other words, the soft bit-reliability vector $\mathbf{\Phi}^{(k)}_{i,j} = [\phi^{(k)}_{i,j,0}, \phi^{(k)}_{i,j,1}, \ldots, \phi^{(k)}_{i,j,r-1}]$ transfers various reliability values for different bits in one symbol. As a result, the CN updates using matrix representations are more effective than those of wBRB.

B. The FBRB Decoding Algorithm

Based on the CN updates with matrix representations, the *full bit-reliabilty based* (FBRB) decoding algorithm with layered structure is proposed.

For the l-th layer in the k-th iteration, the extrinsic bit-reliability values $\mathbf{E}^{(k)}_{j \to i}$'s are calculated by (1), and the extrinsic -sums $\boldsymbol{\sigma}^{(k)}_{i,j}$'s are computed as the same as the wBRB algorithm.

In order to pass r bit-reliability values of $\mathbf{E}^{(k)}_{j \to i}$'s, we represent the nonzero entries $h_{i,j}$'s in the i-th row of $\mathbf{H}$ with the matrix representations $\mathbf{A}^{i,j}_{s,t}$, $j \in N_i$, $0 \le s < r, 0 \le t < r$. According to (7), the soft bit-reliability $\hat{\phi}^{(k)}_{i,j,(s,t)}$'s, $t \in T^{i,j}_s$ can be calculated within r sub-CN updates. Then, the overall soft bit-reliability values $\phi^{(k)}_{i,j,t}$'s, $t = 0, 1, \ldots, r-1$, are computed according to (9).

The remaining steps of FBRB such as the hard-reliability weighting and the $\mathbf{R}^{(l+1,k)}_j$ updates are similar to those of

Algorithm B. The FBRB decoding algorithm for non-binary LDPC codes

Initialization: $R_{j,t}^{(0,0)} = q_{j,t}$; $R_{i\to j,t}^{(0)} = 0$; $z_{j,t}^{(0)} = (R_{j,t}^{(0)} < 0)?1:0$, where $0 \le i < m$, $0 \le j < n$ and $0 \le t < r$; set $\{\theta_0, \theta_1, \ldots, \theta_r\}$ for $\Theta(\cdot)$.

Iteration:

for $k = 0 : I_{max}$

Stop iteration, if $\mathbf{z}^{(k)} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{0}$ or $k = I_{max}$

for $l = 0 : L-1$

for $i \in [m_{l,0}, m_{l,2}, \ldots, m_{l,\tau_l-1}]$

for $j \in N_i$

B1: $\mathbf{E}_{j\to i}^{(k)} = \mathbf{R}_j^{(l,k)} - \mathbf{R}_{i\to j}^{(k)}$

for $j \in N_i$

B2: $\sigma_{i,j}^{(k)} = h_{i,j}^{-1} \sum_{j' \in N_i \setminus j} h_{i,j'} \cdot e_{j'\to i}^{(k)}$

for $s = 0 : r-1$

for $j \in N_i$

for $t \in T_s^{i,j}$

B3: $\hat{\phi}_{i,j,(s,t)}^{(k)} = \prod_{\substack{j' \in N_i, t' \in T_s^{i,j'} \\ j' \ne j \text{ or } t' \ne t}} \mathrm{sign}(E_{j'\to i,t'}^{(k)}) \cdot \min_{\substack{j' \in N_i, t' \in T_s^{i,j'} \\ j' \ne j \text{ or } t' \ne t}} |E_{j'\to i,t'}^{(k)}|$

for $j \in N_i$

for $t = 0 : r-1$

B4: $\phi_{i,j,t}^{(k)} = \mathrm{average}\{\hat{\phi}_{i,j,(s,t)}^{(k)} : s \in S_t^{i,j}\}$

B5: $R_{i\to j,t}^{(k+1)} = \Theta(H(\mathbf{z}_j^{(k)}, \sigma_{i,j}^{(k)})) \cdot \phi_{i,j,t}^{(k)}$

B6: $\mathbf{R}_j^{(l+1,k)} = \mathbf{E}_{j\to i}^{(k)} + \mathbf{R}_{i\to j}^{(k+1)}$

B7: $z_{j,t}^{(k+1)} = (R_{j,t}^{(L,k)} < 0)?1:0$

B8: $\mathbf{R}_j^{(0,k+1)} = \mathbf{R}_{j,t}^{(L,k)}$

wBRB. The FBRB decoding algorithm for NB-LDPC codes is summarized in Algorithm B.

In [30], Jiang et al. proposed a high-performance decoding algorithm for Reed-Solomon codes by adapting the parity-check matrix (ADP) in different iterations. Inspired by ADP, the matrix representations of rows in $\mathbf{H}$ are adapted similarly to enhance the performance of FBRB. In each iteration, the nonzero entry corresponding to the least reliable symbol in each row is normalized to "1". In other words, the nonzero entries in the rows are multiplied by their corresponding inverse values.

C. Computational Complexity and Performance Analysis

In this subsection, the complexity of FBRB, wBRB, IISRB, EMS, T-EMS, IHRB and E-IHRB is compared first. Then, the error performance and convergence rate of these algorithms for two NB-LDPC codes are presented. It is shown that FBRB provides a good trade-off between complexity and performance.

Without loss of generality, a regular NB-LDPC code over GF(2^r) given by the null space of an $m \times n$ matrix is used for complexity analysis. The column and row weights are denoted as γ and ρ respectively. The number of edges in the Tanner graph for this code is $\delta = n\gamma = m\rho$.

The computational complexity per iteration of FBRB is analyzed as follows. According to Algorithm B, the complexity is determined by B1-B6. In B1, the extrinsic bit-reliability values $\mathbf{E}_{j\to i}^{(k)}$'s on δ edges result in δr integer additions. B2 computes the extrinsic information-sum $\sigma_{i,j}^{(k)}$ with 2δ multiplications and $2\delta - m$ additions over Galois field. B3 consists of two parts: the sign computation and the absolute value computation of the soft bit-reliability values. The sign computation takes δr^2 bit-level operations; The absolute values are assigned with the first and second minimum values in the rows of a set of matrix representations so that $mr[(\rho r - 1) + (\rho r - 2)]$ integer comparisons are required. In B4, the overall soft bit-reliability value calculations need $\delta(r^2 - r)$ integer additions. B5 takes δr integer multiplications for weighting. In B6, the reliability updates for codeword symbols cost δr integer additions. Furthermore, the step of adapting matrices mentioned in the above subsection takes δ GF multiplications, $\delta(r-1)$ integer additions and $m(\rho - 1)$ integer comparisons.

The computational complexity of FBRB, wBRB, IHRB, E-IHRB, IISRB, EMS and T-EMS is listed in Table I.

As described in the above subsection, r bit-reliability values instead of only the minimum ones in each symbol are passed through a $r \times r$ binary matrix in FBRB. Therefore, the number of integer comparisons and multiplications in FBRB are about r times than those of wBRB, according to Table I.

It is shown that EMS and T-EMS need much more computational complexity than FBRB. IISRB needs less complexity per iteration, but it requires a more complex initialization. For the hard-decision algorithm IHRB and its variant E-IHRB, they cost much less computational complexity than FBRB, but they both suffer more performance compromise.

The error performance of FBRB, wBRB, EMS, ISRB, IISRB, E-IHRB and IHRB is compared next. All the simulations are conducted over the BI-AWGN channel with BPSK modulation $0 \to +1$ and $1 \to -1$. Parameters of different decoders are optimized by using search algorithms.

Example 1: Consider a (1000, 900) NB-LDPC code $\mathcal{C}$ over GF(2^5) with $\gamma = 4$ and $\rho = 40$. Set $\{5.0, 3.0, 0.25, \ldots, 0.25\}$ for $\Theta(\cdot)$ in wBRB and $\{2.5, 1.25, 0.25, \ldots, 0.25\}$ for $\Theta(\cdot)$ in FBRB. Set $\xi_1 = 4$ and $\xi_2 = 1$ for IISRB, $n_m = 16$ for EMS, $n_c = 3$ and $n_r = 2$ for T-EMS, respectively. All the maximum iteration numbers are set to 50. The performance results of these algorithms are shown in Fig. 2, and the corresponding average iterations are shown in Fig. 3.

For Code $\mathcal{C}$, the performance gap between FBRB and EMS is only 0.3 dB at the BER of 10^{-6}. FBRB outperforms T-EMS, wBRB, IISRB, E-IHRB and IHRB for 0.1 dB, 0.3 dB, 0.45 dB, 0.9 dB and 2.1 dB at the BER of 10^{-6}, respectively. Moreover, IISRB, E-IHRB and IHRB have error-floors at the BER of 10^{-6}, while FBRB still keeps a steep curve down to the BER of 10^{-9}. According to Fig. 3, FBRB converges much faster than T-EMS, wBRB, IISRB, E-IHRB and IHRB. At $E_b/N_0 = 5.0$ dB, the average iterations used for T-EMS, wBRB, IISRB, E-IHRB and IHRB are 1.26, 3.26, 2.96, 3.53 and 12.66 times

TABLE I
NUMBER OF OPERATIONS OF VARIOUS NON-BINARY DECODING ALGORITHMS

Decoding Algorithm		Number of Operations				
		GA	GM	IA/RA	IC/RC	IM
FBRB		$2\delta - m$	3δ	$(r^2+2r-1)\delta$	$(2r^2-1)\delta-(3r-1)m$	δr
wBRB		$2\delta - m$	2δ	$2\delta r$	$(2r+1)\delta-3m$	δ
IHRB		$2\delta - m$	2δ	$\delta+3nq-2n$		
E-IHRB	per iter.	$3\delta - m$	2δ	$2\delta+3nq-2n+(n_m-1)(n_m-2)$		
	init.			$nq(n_m+r-1)-nn_m(n_m+1)/2+nn_m$		
IISRB	per iter.	$2\delta - m$	2δ	2δ		
	init.			$nrq+mn(3\rho-6)(q-1)$		nq
EMS	per iter.	$9n_m(\delta-2m)$	$2\delta n_m$	$5nn_m(3\gamma-5)+\log_2 2n_m(3\delta-4n)$ $+n_m\log_2 n_m(9\delta-12m-4n)$		
	init.			$nq(n_m+r-1)-nn_m(n_m+1)/2$		
T-EMS	per iter.	$3\delta q-\delta+$ $m\sum_{u=1} n_c Tconf_\eta(n_r,u)$	$2\delta q$	$(n_r+5)\delta q-(n_r+3)q$ $+m\sum_{u=1} n_c Tconf_\eta(n_r,u)u$		
	init.			$nq(r-1)$		

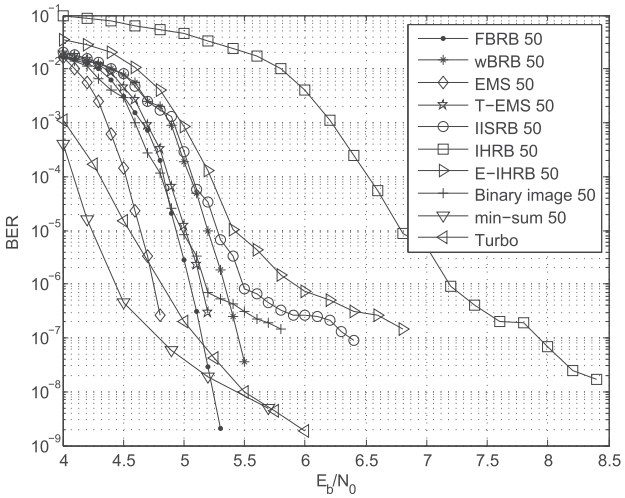

Fig. 2. Performance of the 32-ary (1000,900) code decoded with various decoding algorithms.

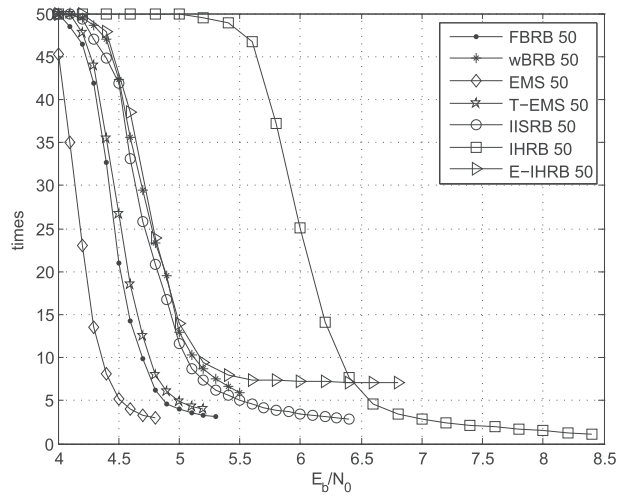

Fig. 3. Average number of iterations for the 32-ary (1000,900) code with wBRB and FBRB.

of that of FBRB, respectively. It is worth mentioning that the performance of using the binary min-sum algorithm (without hard-reliability weighting) over binary images is poor because of short cycles in binary images, as shown in Fig. 2.

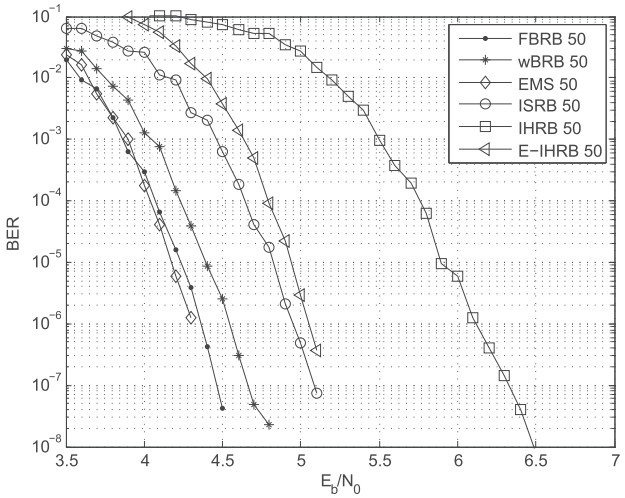

Fig. 4. Performance of the 32-ary (403,226) code decoded with various decoding algorithms.

In addition, we compare the FBRB performance for Code $\mathcal{C}$ with two binary codes. Consider a binary (5040, 4536) LDPC code $\mathcal{C}$ with the same column and row weights. According to Fig. 2, $\mathcal{C}$ performs better in the waterfall region. However, it suffers an error-floor at the BER of 10^{-7}. Compared with another binary code - a 16-state Turbo code $\mathcal{C}$ with the same code rate, FBRB performs worse than $\mathcal{C}$ at the beginning. However, the BER curve of FBRB is much steeper so that it outperforms $\mathcal{C}$ down to the BER of 10^{-8}.

Example 2: Consider a (403, 226) NB-LDPC Code $\mathcal{C}$ over GF(2^5) with $\gamma = 8$ and $\rho = 13$. Set {5.5, 1.25, 0.25, . . . , 0.25} for $\Theta(\cdot)$ in wBRB and {4.0, 0.75, 0.25, . . . , 0.25} for $\Theta(\cdot)$ in FBRB. Set $\lambda = 15$ for ISRB, and $n_m = 16$ for EMS, respectively. The performance of these algorithms for $\mathcal{C}$ are shown in Fig. 4, and the average iterations are shown in Fig. 5.

For Code $\mathcal{C}$, the performance gap between FBRB and EMS is only 0.05 dB at the BER of 10^{-6}. Moreover, FBRB outperforms wBRB, ISRB, E-IHRB and IHRB about 0.2 dB, 0.6 dB, 0.7 dB and 1.8 dB, respectively, at the BER of 10^{-6}.

As shown in Fig. 5, FBRB converges faster than wBRB, ISRB, E-IHRB and IHRB. At $E_b/N_0 = 4.0$ dB, the average iterations of wBRB, ISRB, E-IHRB and IHRB are 2.65, 5.12, 6.13 and 8.13 times of FBRB, respectively.

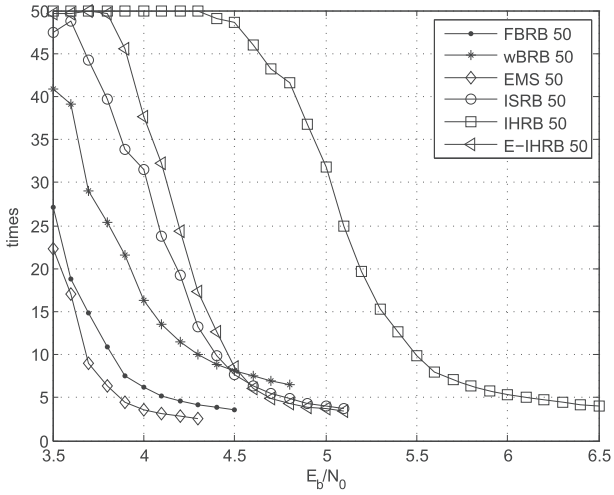

Fig. 5. Average number of iterations required for decoding the 32-ary (403,226) code with wBRB and FBRB.

V. Implementation of the Decoders

In this section, the wBRB and FBRB decoders with partial-parallel structure are designed for QCNB-LDPC codes. Thanks to the layered structure demonstrated in Section III, the CN updates and VN updates can be merged in one computation module. Secondly, the adjacent layers in the proposed decoders can be processed with overlap. Thus, the wBRB and FBRB decoders can achieve high throughputs by using relatively less hardware resources. Since the same top architecture and schedule are shared, they are not distinguished until their computation modules are discussed.

For the sake of simplicity, all sub-matrices are considered to be nonzero shifted identity matrices. The QCNB-LDPC decoders in other cases are similar.

A. Top Architecture

Since the sub-matrices in the parity-check matrix $\mathbf{H}$ of a QCNB-LDPC code are shifted identify matrices, one row of the sub-matrices of $\mathbf{H}$ will be considered as one layer in the layered structure. In the proposed partial-parallel decoders, all the rows in one layer are processed simultaneously. Assume that $\mathbf{H}$ consists of $M \times N$ shifted identity matrices of size $d \times d$. Then, the wBRB and FBRB decoders have M layers and their degrees of parallelism are d in both cases.

The top architecture is illustrated in Fig. 6, which consists of three types of modules, the reliability FIFOs, the message RAMs, and the processing units. Since the degree of parallelism is d, each type has d modules.

- **FIFO:** In the process of the l-th layer in the k-th iteration, the p-th reliability FIFO L_p, $p = 0, 1, \ldots, d-1$, stores the bit-reliability values of codeword symbols $\mathbf{R}_j^{(l,k)}$'s, $j \in N_{m_{l,p}}$. The codeword symbols reliability values $\mathbf{R}_j^{(l,k)}$'s for one block column (d columns) of $\mathbf{H}$ are stored at the same address. Therefore, the depth of these FIFOs are N. The d reliability values for the same block column can be read/written from/to the d reliability FIFOs.

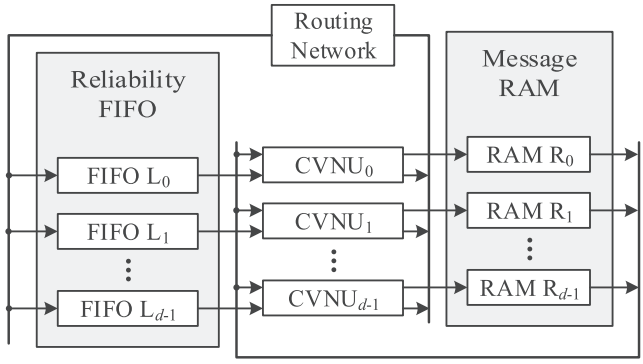

Fig. 6. Top architecture of the wBRB decoder.

- **RAM:** The p-th message RAM R_p, $p = 0, 1, \ldots, d-1$, stores the corresponding c-to-v messages $\mathbf{R}_{m_{l,p}\to j}^{(k)}$'s, $j \in N_{m_{j,p}}$. A block of d c-to-v messages for one shifted identity matrix are stored at the same address for parallel processing. Consider $M \cdot N$ shifted identity matrices in $\mathbf{H}$. Since all the c-to-v messages in the k-th iteration are required in the $(k+1)$-th iteration, the depth of these RAMs are $M \cdot N$.
- **CVNU:** As demonstrated in Section III, the VN updates can be immediately carried out after the CN updates in the layered decoding algorithm. Thus, the proposed decoders have only one type of computation module for both the CN updates and VN updates rather than two types of modules, CN units (CNUs) and VN units (VNUs), which are commonly adopted in traditional LDPC decoders [10], [11], [13]. d such modules are employed in the decoders, named as *check and variable nodes* (CVN) units. In the process of the l-th layer in the k-th iteration, the p-th CVN unit, $p = 0, 1, \ldots, d-1$, processes the p-th row of this layer - row $m_{l,p}$. When the computation of the l-th layer starts, the d CVN units import $\mathbf{R}_j^{(l,k)}$'s from the reliability FIFOs and $\mathbf{R}_{m_{l,p}\to j}^{(k)}$'s from the message RAMs block by block. After N clock cycles, the new reliability values $\mathbf{R}_j^{(l+1,k)}$'s for the next layer and the new c-to-v messages $\mathbf{R}_{m_{l,p}\to j}^{(k+1)}$'s for the next iteration are created by the d CVN units and are ready to be written into the reliability FIFOs and the message RAMs again.
- **Routing network:** Since different layers are related to different shifted identity matrices, the reliability values of the codeword symbols have to be routed into proper order for processing each layer. As any shifted identity matrix can be denoted by an offset integer, the routing can be implemented by barrel shifters [10].

  Remark: In the previous design [10], [20], a routing network is used to route the reliability values $\mathbf{R}_j^{(l,k)}$'s into proper order before they are imported into the CNUs. After the new reliability values are updated, another routing network is used to reverse the routing. In order to save logical resources for routing and reduce critical path delay, the proposed decoder uses only one routing network to route the reliability values according

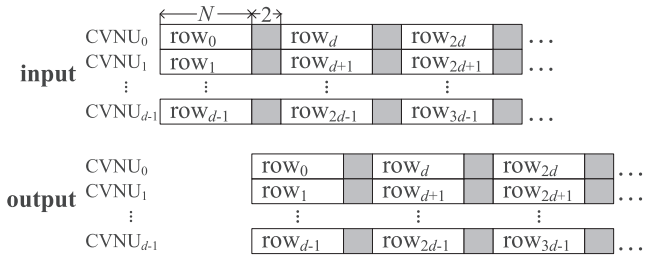

Fig. 7. Schedules of the wBRB and FBRB decoders.

to the relative offsets of the upper and lower adjacent blocks [31].

The schedule of the proposed decoders is illustrated in Fig. 7. In order to achieve higher throughput, the l-th layer output and the $(l+1)$-th layer input are overlapped. The output of the l-th layer takes N clock cycles. Two clock cycles after the CVN units export the new reliability values $\mathbf{R}_j^{(l+1,k)}$'s, the CVN units can import these values from the LLR FIFOs that are going to be processed in the next layer.

B. CVN Unit for the wBRB Decoder

The CVN unit is the core computation unit of the wBRB decoder. Its architecture is described in Fig. 8. To simplify the implementation, all the numerical signals are designed in the sign-magnitude format. In the l-th layer of the k-iteration, the computation of one CVN unit is carried out following the following steps:

First, the new extrinsic information $\mathbf{E}_{j\to i}^{(k)}$'s $(j \in M_i)$ are computed by the r subtracters in the sign-magnitude format.

Second, the signs and magnitudes of $\mathbf{E}_{j\to i}^{(k)}$'s are grouped into two branches. The signs go to the branch to calculate $\boldsymbol{\sigma}_{i,j}^{(k)}$'s. The magnitudes go to the branch to calculate the soft bit-reliability $\boldsymbol{\phi}_{i,j}^{(k)}$'s. The two branches are processed simultaneously.

- Sign-computation: In order to simplify the computation of $\boldsymbol{\sigma}_{i,j}^{(k)}$'s, the equation in Step B2 can be rewritten as $\boldsymbol{\sigma}_{i,j}^{(k)} = h_{i,j}^{-1}(h_{i,j}\cdot e_{j\to i}^{(k)} + \sum_{j'\in N_i} h_{i,j'}\cdot e_{j'\to i}^{(k)})$. As a result, the information-sum $\sum_{j'\in N_i} h_{i,j'}\cdot e_{j'\to i}^{(k)}$ only needs to be calculated once for the entire row computation. The addition over Galois field can be implemented by bitwise XORs. Therefore, an XOR-register loop is designed to accumulate the information-sum. After N clock cycles, the information-sum is worked out and latched. Adding the latched information-sum to the cached products $h_{i,j}\cdot e_{j\to i}^{(k)}$, and multiplying with $h_{i,j}^{-1}$, the $\boldsymbol{\sigma}_{i,j}^{(k)}$ is derived.
- Magnitude-computation: An r-input comparer is used to compute the minimum magnitude of $E_{j\to i,t}^{(k)}$ $(t = 0, 1, \ldots, r-1)$, while a comparer-register loop is designed to serially calculate the minimum and the second minimum soft bit-reliability values. Then, $\boldsymbol{\phi}_{i,j}^{(k)}$'s are produced by a 2-to-1 multiplexer. For the j-th symbol, $\boldsymbol{\phi}_{i,j}^{(k)}$ is assigned to the second minimum reliability value if j equals to the index of the minimum reliability value, or it is assigned to the minimum reliability value otherwise.

Third, $\boldsymbol{\phi}_{i,j}^{(k)}$ is weighted by the distance of $\boldsymbol{\sigma}_{i,j}^{(k)}$ and the hard-decision of the reliability $\mathbf{R}_j^{(l,k)}$. Then the new c-to-v message $\mathbf{R}_{i,j}^{(k+1)}$ is the combination of $\boldsymbol{\sigma}_{i,j}^{(k)}$ and the weighted $\boldsymbol{\phi}_{i,j}^{(k)}$. The new reliability value $\mathbf{R}_j^{(l+1,k)}$ is the cached sum of $\mathbf{E}_{j\to i}^{(k)}$ and the new $\mathbf{R}_{i,j}^{(k+1)}$.

It is worth mentioning that since the $\mathbf{R}_j^{(l,k)}$'s, $h_{i,j}\cdot e_{j\to i}^{(k)}$'s and $\mathbf{E}_{j\to i}^{(k)}$'s are required in more than one steps, cache is used to temporarily store these signals for N clock cycles.

C. CVN Unit for the FBRB Decoder

The architecture of the CVN unit used in the FBRB decoder is illustrated in Fig. 9. Similarly to wBRB, the computation of the CVN unit in the FBRB decoder consists of two parts - sign computation and magnitude computation. The sign-computation branch is similar to that of wBRB. However, the magnitude computation is split into r branches corresponding to the r sub-CNs.

For the s-th magnitude-computation branch, r absolute values of $\mathbf{E}_{j\to i}^{(k)}$ are imported after filtered by the s-th row of the binary image $\mathbf{A}_{s,*}^{i,j}$. The t-th input of this branch is assigned with $|E_{j\to i,t}^{(k)}|$ if $A_{s,t}^{i,j} = 1$, or it is assigned with the allowable maximum value if $A_{s,t}^{i,j} = 0$. Similar to the wBRB structure, a comparer-register loop is employed to compute the minimum and the second minimum values of input. After N clock cycles, the results are latched. Then, r sub soft bit-reliability values of $\hat{\boldsymbol{\phi}}_{i,j,(s,t)}^{(k)}$'s, $t = 0, 1, \ldots, r-1$, are exported by r 2-to-1 multiplexers.

As shown in Fig. 9, in order to calculate the overall soft bit-reliability $\boldsymbol{\phi}_{i,j,t}^{(k)}$, the t-th column of the binary image $\mathbf{A}_{*,t}^{i,j}$ is introduced to filter the r inputs of the t-th average module. The s-th input is assigned with $\hat{\boldsymbol{\phi}}_{i,j,(s,t)}^{(k)}$ if $A_{s,t}^{i,j} = 1$, or it is assigned with 0 otherwise.

If the matrix representations of all the nonzero entries of $\mathbf{H}$ are directly stored, a large amount of memory resources will be consumed. In the following subsection, an efficient way to generate these matrix representations from their binary vector representations is derived by simple logical operations.

Theorem 1: Suppose α is a primitive element of GF(2^r) generated by the primitive polynomial $p(x)$. The matrix representation $\mathbf{A}^l$ of any nonzero symbol α^l over GF(2^r) is equal to

$$\mathbf{A}^l = \mathbf{B}\odot\mathbf{a}^{(l)} \triangleq [\mathbf{B}_0\cdot\mathbf{a}^{(l)\mathrm{T}}, \mathbf{B}_1\cdot\mathbf{a}^{(l)\mathrm{T}}, \ldots, \mathbf{B}_{r-1}\cdot\mathbf{a}^{(l)\mathrm{T}}], \tag{10}$$

where $\mathbf{B} = [\mathbf{B}_i]$, $i = 0, 1, \ldots, r-1$, $\mathbf{B}_i$'s are $r\times r$ binary matrices over GF(2), and $\mathbf{a}^{(l)}$ is the vector representation of α^l.

Proof: The proof is provided in the appendix. ■

Example 3: Consider GF(2^5) generated by the primitive polynomial $p(x) = 1 + x^2 + x^5$. According to Theorem 1, the

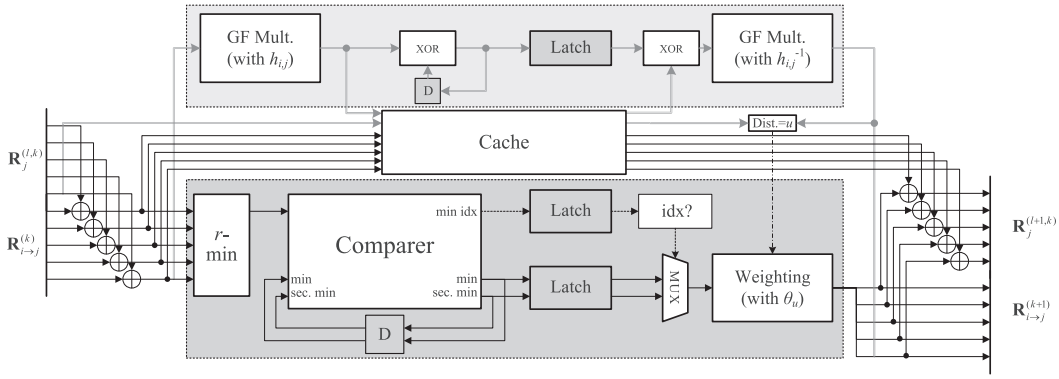

Fig. 8. Architecture of the CVN units for wBRB.

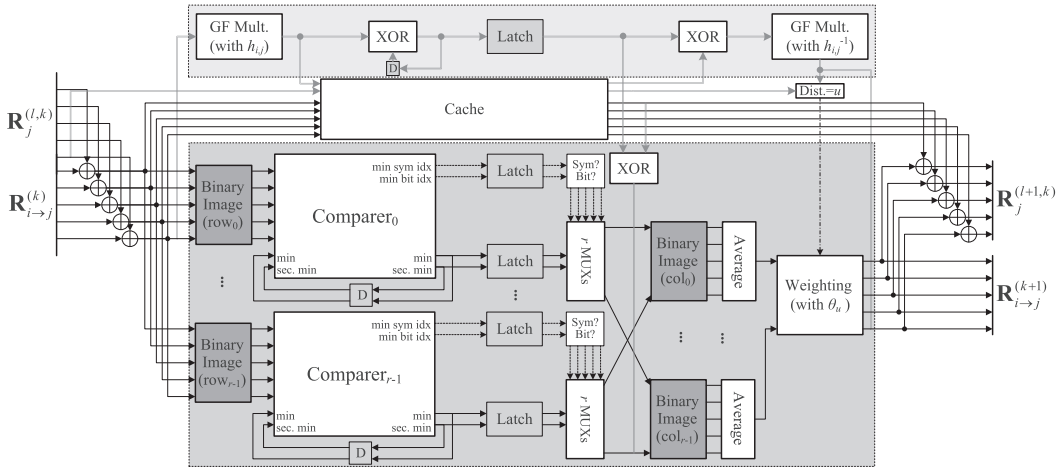

Fig. 9. Architecture of the CVN units for FBRB.

matrix representation $\mathbf{S}$ of each symbol can be obtained from its vector representation $\mathbf{s} = [s_0, s_1, s_2, s_3, s_4]$ and the matrix $\mathbf{B} = [\mathbf{B}_0|\mathbf{B}_1|\mathbf{B}_2|\mathbf{B}_3|\mathbf{B}_4]$, where

$$\mathbf{B} = \left[\begin{array}{c|c|c|c|c} 10000 & 00001 & 00010 & 00100 & 01001 \\ 01000 & 10000 & 00001 & 00010 & 00100 \\ 00100 & 01001 & 10010 & 00101 & 01011 \\ 00010 & 00100 & 01001 & 00010 & 00101 \\ 00001 & 00010 & 00100 & 01001 & 10010 \end{array}\right]. \quad (11)$$

Thus, $\mathbf{S} = [\mathbf{B}_0 \cdot \mathbf{s}^{\mathrm{T}}|\mathbf{B}_1 \cdot \mathbf{s}^{\mathrm{T}}|\mathbf{B}_2 \cdot \mathbf{s}^{\mathrm{T}}|\mathbf{B}_3 \cdot \mathbf{s}^{\mathrm{T}}|\mathbf{B}_4 \cdot \mathbf{s}^{\mathrm{T}}]$.

Since all the nonzero elements over GF(2^r) share the same $\mathbf{B}$, the matrix representations of the nonzero entries of $\mathbf{H}$ can be easily obtained from their vector representations by a simple logical circuit. The circuit for GF(2^5) is shown in Fig. 10. Thus, only vector representations of nonzero entries of $\mathbf{H}$ need to be stored rather their matrix representations.

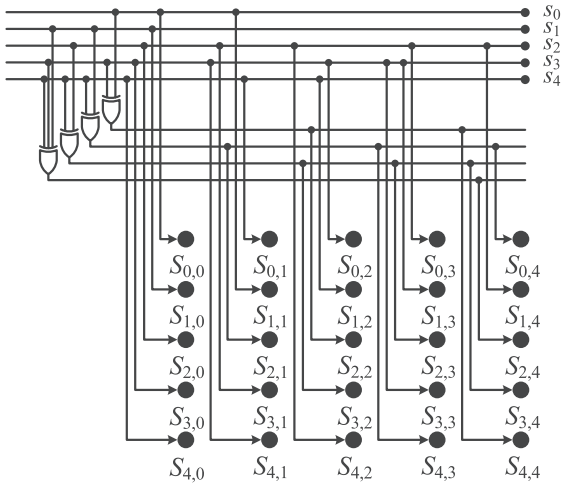

Fig. 10. The circuit generating the matrix representations of GF(2^5).

D. Analyses and Comparisons

In the previous work [20], the authors efficiently designed an IHRB decoder and an E-IHRB decoder with 3-bit quantization for a (403,226) QCNB-LDPC code over GF(2^5) and re-implemented a min-max decoder with 5-bit quantization for the same code. For the purpose of comparison, a wBRB decoder and an FBRB decoder with 5-bit quantization are designed for the same code.

The parity-check matrix $\mathbf{H}$ of this code consists of 8×13 shifted identity matrices with the dimensions of 31×31. Thus, the wBRB and FBRB decoders have 8 layers with the degree of parallelism 31, which are implemented on a Xilinx Virtex-5 XC5VLX155T device. According to the decoder schedules shown in Fig. 7, it takes the 31 CVNUs 13 clock cycles to

TABLE II
SYNTHESIS REPORT OF (403, 226) QCNB-LDPC DECODERS ON A XILINX VIRTEX-5 XC5VLX200T DEVICE

	wBRB	FBRB	IHRB [20]	E-IHRB [20]	min-max [20]
Slice Registers	1269	4556	529	716	38660
Slice LUTs	10517	36147	7841	9153	62605
BRAMs	40	49	56	71	95
Max. Freq.(Mhz)	106.28	85.52	117.6	109.8	90.9
Throughput(Mbps)	118.98	95.73	90.68	84.67	2.63

decode a layer of $\mathbf{H}$ with two clock cycles inserted between each two layers. Thus, each iteration of decoding takes $(13+2)\times 8 = 120$ clock cycles. The maximum number of iterations is set to 15, since the two decoders converge very fast as shown in Fig. 5. As a result, each received codeword costs $120 \times 15 = 1800$ clock cycles for decoding. The throughputs of the wBRB and FBRB decoders are 118.98 Mbps and 95.73 Mbps, respectively. The detailed synthesis report of these decoders is presented in Table II.

Compared with the soft-decision min-max decoder in [20], the wBRB decoder takes 16.8% slice LUTs and 42.1% BRAMs. Moreover, it achieves 45.2 times higher throughput than the min-max decoder.

Compared with the hard-decision IHRB decoder and its variant E-IHRB decoder, the wBRB decoder requires 1.43 and 1.23 times more slice LUTs, but costs 71.4% and 56.3% less BRAMs, respectively. Moreover, it achieves 1.17 and 1.26 times higher throughput, respectively.

The FBRB decoder needs more slice LUTs and BRAMs than the wBRB decoder. However, the FBRB decoder achieves better error performance and converges much faster. At E_b/N_0 of 4.5 dB, the average iterations of the FBRB and wBRB decoder are 3.628 and 8.095, respectively. The average throughputs of the FBRB and wBRB decoders are 395.8 Mbps and 220.46 Mbps, respectively.

VI. CONCLUSION

This paper presents two efficient bit-reliability based decoders, wBRB and FBRB, for NB-LDPC codes. Compared with existing symbol-reliability based decoders, the two decoders can achieve better performance with much less memory consumptions. The proposed layered structure not only improves the BRAMs usage with higher efficiency but also leads to faster decoding speed. Furthermore, the proposed routing network according to the relative offsets can also save the usage of barrel shifters. For a 32-ary (403,226) QCNB-LDPC code, the two decoders achieve 118.98 Mbps throughputs with 10517 LUTs and 40 BRAMs, and 95.73 Mbps throughputs with 36147 LUTs and 49 BRAMs, respectively, on Xilinx Virtex-5 XC5VLX155T. Finally, an efficient way is proposed to generate binary matrix representations of GF elements that is useful in applications involving matrix representations.

APPENDIX
PROOF OF THEOREM 1

Proof: Consider GF(2^r) generated by the primitive polynomial $p(x) = p_0 + p_1 x + \ldots + p_{r-1}x^{r-1} + x^r$. $\mathbf{A}$ is the companion matrix of $p(x)$. α is a primitive element of GF(2^r). $\mathbf{A}^l$ is the matrix representation of α^l. Denote $\mathbf{a}^{(l)} = [a_0^{(l)}, a_1^{(l)}, \ldots, a_{r-1}^{(l)}]$ as the vector representation of α^l:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \ldots & 0 & p_0 \\ 1 & 0 & \ldots & 0 & p_1 \\ 0 & 1 & \ldots & 0 & p_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & p_{r-1} \end{bmatrix} = [\mathbf{a}^{(1)^{\mathrm{T}}}, \mathbf{a}^{(2)^{\mathrm{T}}}, \ldots, \mathbf{a}^{(r-1)^{\mathrm{T}}}, \mathbf{a}^{(r)^{\mathrm{T}}}]$$

Rewrite $\mathbf{A}^l$ as

$$\begin{aligned} \mathbf{A}^l &= \mathbf{A}^{l-1} \cdot [\mathbf{a}^{(1)^{\mathrm{T}}}, \mathbf{a}^{(2)^{\mathrm{T}}}, \ldots, \mathbf{a}^{(r)^{\mathrm{T}}}] \\ &= \mathbf{A}^{l-2} \cdot [\mathbf{A}\cdot\mathbf{a}^{(1)^{\mathrm{T}}}, \mathbf{A}\cdot\mathbf{a}^{(2)^{\mathrm{T}}}, \ldots, \mathbf{A}\cdot\mathbf{a}^{(r)^{\mathrm{T}}}] \\ &\vdots \\ &= [\mathbf{a}^{(l)^{\mathrm{T}}}, \mathbf{a}^{(l+1)^{\mathrm{T}}}, \ldots, \mathbf{a}^{(l+r-1)^{\mathrm{T}}}] \end{aligned}$$

According to $\mathbf{a}^{(l+k)^{\mathrm{T}}} = \mathbf{A}^k \cdot \mathbf{a}^{(l)^{\mathrm{T}}}$,

$$\begin{aligned} \mathbf{A}^l &= [\mathbf{I}\cdot\mathbf{a}^{(l)^{\mathrm{T}}}, \mathbf{A}\cdot\mathbf{a}^{(l)^{\mathrm{T}}}, \ldots, \mathbf{A}^{r-1}\cdot\mathbf{a}^{(l)^{\mathrm{T}}}] \\ &\triangleq [\mathbf{I}|\mathbf{A}|\ldots|\mathbf{A}^{r-1}] \odot \mathbf{a}^{(l)^{\mathrm{T}}}. \end{aligned}$$

■

REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1. pp. 21–28, 1962.

[2] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *Proc. 5th IMA Conf. Rec. IEEE IAS Annu. Meeting Cryptography Coding*, 1995, pp. 100–111.

[3] N. Alon and M. Luby, "A linear time earsure-resilient code with nearly optimal recovery," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1732–1736, Nov. 1996.

[4] D. J. C MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 399–431, Mar. 1999.

[5] B. Zhou, J. Kang, Y.-Y. Tai, Q. Huang, and S. Lin, "High performance nonbinary quasi-cyclic LDPC codes on Euclidean geometries," in *Proc. IEEE Mil. Commun. Conf*., 2007, pp. 1–8.

[6] L. Zeng, L. Lanand, and Y.-Y. Tai, "Construction of nonbinary cyclic, quasi-cyclic and regular LDPC codes: A finite geometry approach," *IEEE Trans. Commun.*, vol. 56, no. 3, pp. 378–387, Mar. 2008.

[7] B. Zhou, J. Kang, and S. Song, "Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1652–1662, Jun. 2009.

[8] H. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," in *Proc. IEEE Inf. Theory Workshop*, 1998, pp. 70–71.

[9] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. IEEE Int. Conf. Commun*., 2004, vol. 2, pp. 772–776.

[10] X. Zhang and F. Cai, "Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 2, pp. 402–414, Feb. 2011.
[11] K. Zhang and X. Huang, "A low-complexity rate-compatible LDPC decoder," in *Proc. Asilomar Conf. Signals. Syst. Comput*., 2009, pp. 749–753.
[12] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.
[13] V. Savin, "Min-max decoding for non binary LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2008, pp. 960–964.
[14] E. Li, K. Gunnam, and D. Declercq, "Trellis based extended min-sum algorithm for non-binary LDPC codes," in *Proc. IEEE Int. Symp. Wireless Commun. Syst*., 2010, pp. 46–50.
[15] J. O. Lacruz, F. Garcia-Herrero, D. Declercq, and J. Valls, "Simplified trellis min-max decoder architecture for nonbianry low-density parity-check codes," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 9, pp. 1783–1792, Sep. 2015, doi: 10.1109/TVLSI.2014.2377194.
[16] Y.-L. Ueng, K.-H. Liao, H.-C. Chou, and C.-J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using max-log-QSPA," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2940–2951, Jun. 2013.
[17] J. O. Lacruz, F. Garcia-Herrero, and J. Valls, "Reduction of complexity for nonbinary LDPC decoders with compressed messages," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 11, pp. 2676–2679, Nov. 2015, doi: 10.1109/TVLSI.2014.2344113.
[18] C.-Y. Chen, Q. Huang, and C.-C. Chao, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. Commum.*, vol. 58, no. 11, pp. 3140–3147, Nov. 2010.
[19] C. Xiong and Z. Yan, "Improved iterative soft-reliability-based majority-logic decoding algorithm for non-binary low- density parity-check codes," in *Proc. IEEE Signals Syst. Comput*., 2011, pp. 894–898.
[20] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 11, pp. 1938–1950, Nov. 2012.
[21] Q. Huang, M. Zhang, Z. Wang, and L. Wang, "Bit-reliability based low-complexity decoding algorithms for non-binary LDPC codes," *IEEE Trans. Commum.*, vol. 62, no. 12, pp. 4230–4240, Dec. 2014.
[22] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[23] X. Chen, S. Lin, and V. Akella, "Efficient configurable decoder architecture for nonbinary quasi-cyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 1, pp. 188–197, Jan. 2012.
[24] Y.-L. Ueng, C.-Y. Leong, C.-J. Yang, C.-C. Cheng, K.-H. Liao, and S.-W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 2, pp. 385–398, Feb. 2012.
[25] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop Signal Process. Syst*., 2004, pp. 107–112.
[26] Z. Cui, Z. Wang, and Y. Liu, "High-throughput layered LDPC decoding architecture," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 4, pp. 582–587, Apr. 2009.
[27] K. K. Gunnam, G. S. Cho, and M. B. Yeary, "A parallel VLSI architecture for layered decoding for array LDPC codes," in *Proc. Int. Conf. Very Large Scale Integr. (VLSI) Des*., Bangalore, India, Jan. 2007, pp. 738–743.
[28] F. Williams and N. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North Holland, 1978.
[29] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, d_c)-LDPC codes over GF(q) using their binary images," *IEEE Trans. Commum.*, vol. 56, no. 10. pp. 1626–1635, Oct. 2008.
[30] J. Jiang and K. R. Narayanan, "Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3746–3756, Aug. 2006.
[31] S. Kim, G. E. Sobelman, and H. Lee, "A reduced-complexity architecture for LDPC layered decoding schemes," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 6, pp. 1099–1103, Jun. 2011.

Qin Huang received the B.E. and M.E. degrees from Southeast University, Nanjing, China, both in electrical engineering, and the Ph.D. degree in electrical engineering from the University of California, Davis, Davis, CA, USA, in 2005, 2007 and 2011, respectively. He joined Link-A-Media Devices Corporation, Santa Clara, CA, USA, in 2011. He is currently an Associate Professor with Beihang University (BUAA), Beijing, China. His research interests include classical and modern coding theory, signal processing, and their applications on communications and storage systems.

Shuai Yuan received the B.S. and M.S. degrees from Beihang University, Beijing, China, both in electronic and information engineering, in 2011 and 2014, respectively. He is currently a Research Assistant with Qian Xuesen Laboratory of Space Technology, Beijing, China. His research interests include error-control coding and network coding.