J Sign Process Syst (2015) 78:209–222
DOI 10.1007/s11265-013-0816-5

Block-Layered Decoder Architecture for Quasi-Cyclic Nonbinary LDPC Codes

Chang-Seok Choi · Hanho Lee

Received: 21 February 2013 / Revised: 10 June 2013 / Accepted: 20 June 2013 / Published online: 7 July 2013

Abstract This paper presents a block-layered decoder architecture and efficient design techniques for quasi-cyclic nonbinary low-density parity-check (QC-NB-LDPC) codes. Based on a Min-Max decoding algorithm, an efficient block-layered decoder architecture for QC-NB-LDPC codes is proposed for fast decoder convergence. Further, a novel two-way merging Min-Max algorithm, which significantly reduces decoding latency, is proposed for check node processing. The NB-LDPC decoder using the proposed algorithm can provide a considerably higher throughput rate than that using a conventional Min-Max algorithm. The proposed (225, 165) NB-LDPC decoder over GF(2^4) is synthesized using a 90-nm CMOS process. It can operate at a clock rate of 400 MHz and achieve a data processing rate of 24.6 Mbps under 10 decoding iterations.

Keywords Nonbinary · Low-density parity-check (LDPC) codes · Min-Max decoding · Two-way merging · Block-layered · VLSI

1 Introduction

Since Shannon first introduced the noisy channel coding theorem in 1948, progress on error correcting codes has been gradually approaching the Shannon limit. Within this trend, low-density parity check (LDPC) codes, originally discovered by Gallager in the early 1960s [1], have received a lot of attention over the last two decades. Their powerful error correcting capabilities have been validated, which has led to their inclusion in various standards as a channel coding scheme. Nonbinary LDPC (NB-LDPC) codes constructed over finite fields GF(q) (q>2) were recently proposed by Davey and MacKay [2]. Binary LDPC (B-LDPC) codes with a very large code length tend to approach the Shannon limit; on the other hand, NB-LDPC codes can improve their error correcting capability by increasing q for moderate code lengths.

Recently, NB-LDPC codes [3–12] have attracted a tremendous amount of research interest owing to their excellent error correction capabilities. However, the significant improvement in error correction capabilities introduces a penalty of high decoding complexity. The belief propagation (BP) algorithm for B-LDPC codes can be extended for NB-LDPC codes [2, 3]. In addition, the BP algorithm can be efficiently implemented using a fast Fourier transform (FFT) in the probability domain [4, 5]. The computational complexity of these algorithms is dominated by $O(q^2)$ and $O(q \log_2 q)$ for each check node process. While probability domain algorithms are optimal, a large number of additions and multiplications can lead to an exponential increase in hardware complexity. Several decoding schemes have been proposed to deal with this problem. The channel probability can be changed using log-likelihood ratios (LLRs). A log-domain decoding algorithm called log-BP, also called a log sum-product algorithm (SPA), was presented by Wymeersch et al. [6]. This log-BP algorithm has been shown to be equivalent to the original BP algorithm, and the log-BP decoding scheme has advantages in terms of both decoding complexity and numerical robustness.

Research has been conducted on an efficient decoder architecture design for NB-LDPC codes. Spagnol et al. [7] proposed a mixed domain decoder architecture, which requires FFT computation, for decoding NB-LDPC codes. This architecture complicates product operation, which increases decoder complexity. Generally, sub-optimal decoding algorithms have been

C.-S. Choi · H. Lee (✉)
Department of Information and Communication Engineering,
Inha University, Incheon 402-751, Republic of Korea
e-mail: hhlee@inha.ac.kr

considered for implementing practical NB-LDPC decoder architectures. They can provide reduced hardware complexity using an approximated version of check node processing. Following this trend, the extended Min-Sum (EMS) [8, 9] and Min-Max algorithms [10] have been a subject of great interest for very-large scale integration (VLSI) implementation. Both of these algorithms use LLR values to decode channel messages, and as a result, complicated multiplications are replaced by additions in the log-domain. The EMS algorithm, which is an extended version of the Min-Sum (MS) algorithm for B-LDPC codes, was further developed by Voicila et al. [11], who proposed a truncated value of $n_m<q-1$ to maintain the reliability of each message. Only a limited number of n_m values are used the check node. Thus, the proposed scheme provides a significant reduction in the number of memory elements [12]. The BCJR algorithm computation scheme for turbo decoding is geared to the Min-Max algorithm, which is similar to the butterfly algorithm proposed by Muller et al. [13]. Sayin [10] exchanged the sum operation in the EMS algorithm with a max operation for check node processing. This provides advantages for VLSI implementation, because a max operation can easily be implemented using a comparator instead of a sum operation. For this reason, some NB-LDPC decoder architectures based on the Min-Max algorithm were proposed [14–17]. Although the past few years have shown a significant growth in research on NB-LDPC codes and decoding algorithms, there have been very few publications on NB-LDPC decoder implementations.

In this paper, an efficient block-layered decoder architecture for quasi-cyclic NB-LDPC (QC-NB-LDPC) codes is presented. A novel two-way merging Min-Max algorithm, which significantly reduces the decoding latency, is proposed for check node processing. Non-overlapped rows can be grouped in NB-LDPC codes, facilitating a hardware-friendly, block-layered decoding scheme, which provides fast decoder convergence. A new check node unit (CNU) architecture using a two-way merging Min-Max algorithm is then proposed to improve throughput.

The remainder of this paper is organized as follows. In Section 2, the research background of NB-LDPC codes is briefly introduced. Code construction methods and a brief review of a block-layered decoding algorithm for NB-LDPC codes are presented in Section 3. Section 4 presents the two-way merging Min-Max algorithm and the corresponding NB-LDPC decoder architecture. Specifically, the development of an efficient block-layered decoder architecture and the two-way merging Min-Max algorithm for a CNU is described in detail. Hardware cost estimation and throughput comparisons with other related NB-LDPC decoder designs are presented in Section 5. Finally, some conclusions are given in Section 6.

2 Research Background of NB-LDPC Codes and Architectures

The NB-LDPC code, where parity check matrix **H** is defined over GF(q), is a subset of LDPC codes. Non-zero elements of **H** are mapped to the GF elements. NB-LDPC codes typically show a superior performance compared to existing channel code schemes [18–20]. Various precedence studies have analyzed the performance of NB-LDPC codes. For example, Zhou et al. [18, 19] proposed some algebraic construction methods for NB-LDPC codes and showed a performance comparison with Reed-Solomon (RS) codes. Their comparison shows that the constructed QC-NB-LDPC codes significantly outperform corresponding RS codes. A fading channel was considered in simulations for one of the studies [19]. Peng and Chen [20] investigated the application of NB-LDPC codes to multiple-input multiple-output (MIMO) channels. The results showed that an NB-LDPC coded system achieves superior performance compared to B-LDPC coded systems. Therefore, NB-LDPC codes have great potential to replace the channel coding scheme for mobile wireless communication and storage systems.

Another application of NB-LDPC codes is in the field of optical communications. In fact, higher rate optical transport systems require a powerful forward error correction (FEC) scheme because FECs are considered the most cost-effective solution to improve the optical signal-to-noise ratio (OSNR) deficit. For this reason, NB-LDPC codes were investigated for a next generation FEC scheme by Arabaci et al. [21]., where the studies demonstrated that high-rate regular QC-NB-LDPC codes are very suitable for optical communications [22]. As mentioned above, many studies have shown an interest in NB-LDPC codes and their applications.

Very few NB-LDPC decoder architectures have been reported thus far. Lin et al. [14] and Zhang and Cai [16] proposed decoder architectures using the selective Min-Max algorithm. Although the selective Min-Max algorithm can reduce the computational complexity by almost half, it may require a more complicated control logic and non-fixed memory elements. Lin et al. [14] presented a direct selection algorithm to find an appropriate LLR value when a K-bit LLR is given. In addition, the NB-LDPC decoder architecture was briefly described based on the selection algorithm.

Zhang and Cai [16] presented an efficient partially parallel NB-LDPC decoder architecture, based on the Min-Max algorithm, for QC-NB-LDPC codes. When the check node degree is not small, the proposed architecture by Zhang and Cai can achieve higher efficiency using an overlapped

method for check node processing [16]. Overlapped scheduling is used among the different layers to further speed up the decoding process. A path construction scheme–based NB-LDPC decoder architecture was also presented by Zhang and Cai [17]. That proposed scheme significantly reduces the computation complexity and eliminates the memory requirement for storing intermediate messages generated from the forward and backward processes. The standard Min-Max-based NB-LDPC decoder architecture was originally discussed by Lin et al. [15]. The authors provided an improved decoding algorithm to remove the GF multiplications at the elementary step for Min-Max decoding. As a result, efficient CNU and variable node unit (VNU) architectures were presented. Recently, a configurable decoder architecture was presented by Chen et al. [23]. This proposed decoder supports both regular and irregular codes, and as a result, a single decoder can be used to decode any code of a given GF size. An efficient layered decoding architecture was also proposed by Ueng et al. [24]. Instead of a switching network, a barrel-shifter-based permutation network that can control the permutation related to multiplications over the GF is used. In addition, detailed full-chip implementation results using a 90-nm CMOS technology were provided.

As mentioned above, a lot of effort has been spent in the design of efficient decoding algorithms and architectures for NB-LDPC codes. However, only a few NB-LDPC decoder architectures have been proposed owing to the inherently large complexity of NB-LDPC decoding algorithms. Therefore, a study on hardware-friendly decoding algorithms and efficient decoder architectures for NB-LDPC codes is currently needed.

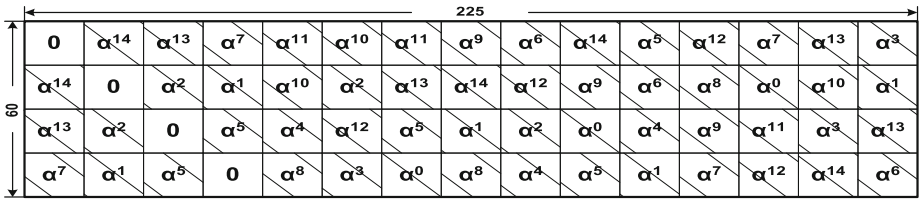
3 NB-LDPC Code and Decoding Algorithm

3.1 Code Generation of NB-LDPC Code

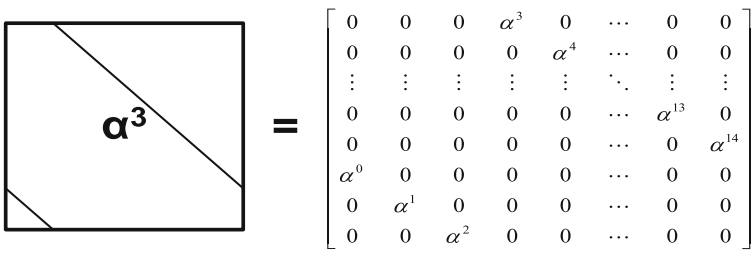
The algebraic properties of regular NB-LDPC codes generated using their binary images were presented by Poulliat et al. [25]. Most NB-LDPC codes are constructed by taking a known binary LDPC code and replacing its non-zero elements with randomly generated GF elements.

To generate good-quality NB-LDPC codes, some structured construction methods were proposed by Zhou et al. [19, 26]. In 2008, they presented two new algebraic construction methods based on array-dispersions of matrices over nonbinary subgroups [26]. The base matrix of both the RS and row-column (RC)–constrained codes can be extended using an array dispersion technique. One class of array dispersion–based NB-LDPC codes exhibits very good performance. RC-constrained arrays [26] are used to generate our selected (225, 165) NB-LDPC code over GF(2^4), which provides a greatly reduced hardware complexity compared to an NB-LDPC code over GF(2^5). The property of an RC-constraint is a constraint on the rows and columns of the H-matrix. The detailed construction process was described in 2008 [26]. The property of an RC-constraint is a constraint on the rows and columns of the H-matrix. Using the method proposed in 2008 [26], a 15×15 $H^{(1)}$ matrix is generated. Set column weight γ=4 and row weight ρ=5; the 4×5 sub-array $H^{(1)}(4, 15)$ is then a 60×225 matrix over GF(2^4).

The generated $H^{(1)}(4,15)$ matrix is extended using an α-multiplied circulant permutation matrix (CPM). Figure 1(a) and (b) show the H-matrix of the selected (225, 165) NB-LDPC code over GF(2^4) and an α-multiplied CPM for α^3,



(a)



$$\begin{bmatrix} 0 & 0 & 0 & \alpha^3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \alpha^{13} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \alpha^{14} \\ \alpha^0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \alpha^1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \alpha^2 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

(b)

Figure 1 **a** H-matrix for a (225, 165) NB-LDPC code over GF(2^4) with code rate 0.733. **b** Example of α-*multiplied* CPM for α^3.

respectively. The selected (225, 165) NB-LDPC code has a constant row weight, $d_c=14$, and a two-column weight, $d_v=3$ or 4.

3.2 Block-layered Decoding Algorithm

In layered LDPC decoding, if non-overlapped k-rows can be grouped into subsets, each column of these subsets has a weight of at most 1. The known advantage of the layered decoding is its reduced computation complexity, as well as a faster decoding convergence [24, 28]. This layered decoding scheme has been commonly used to implement B-LDPC decoders [29–31].

Figure 2 shows the dataflow of the layered NB-LDPC decoding. The initialization of the decoder is achieved using soft values from the channel LLRs in the bit update block, and the decoder starts updating messages with the initial channel message $\gamma^{(f)}{}_n(a)$ and the variable node message $\beta^{(f)}{}_{m,n}(a)$, where m is the check node degree and n is the variable node degree. The check node message $\alpha_{m,n}(a)$ is used to compute the check node update, and as a result, $\beta^{(f+1)}{}_{m,n}(a)$ is stored in memory. At the same time, the updated posterior messages are computed by adding $\beta^{(f+1)}{}_{m,n}(a)$ and $\alpha_{m,n}(a)$.

Then $\gamma'{}_n(a)$ is passed through the normalization block, and finally $\gamma^{(f+1)}{}_n(a)$ is used to compute the bit update block again. If the NB-LDPC code has a CPM-based H-matrix, the block layers can easily be applied to implement the NB-LDPC decoder. In the (225, 165) NB-LDPC code, each of the 15 rows can be divided into four subsets, as the CPM is defined over GF(2^4).

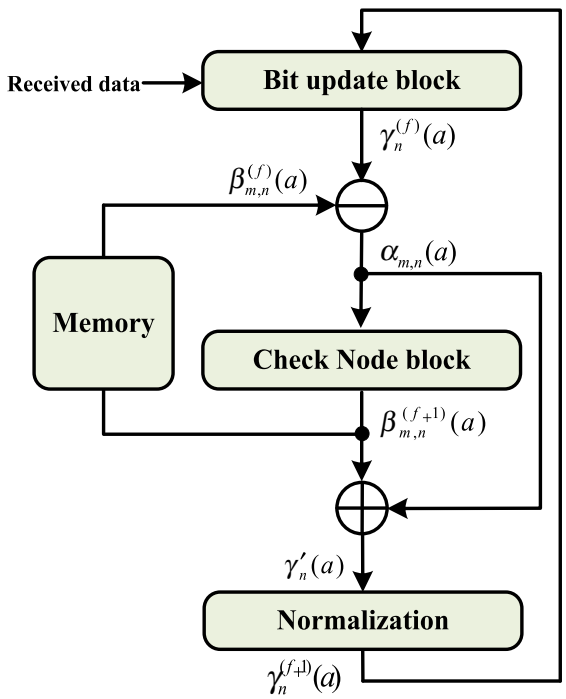


Figure 2 Dataflow of block-layered NB-LDPC decoding.

The proposed block-layered decoding algorithm is given as follows:

Block-layered decoding algorithm

Initialization :

$$\gamma_n(a) = \ln(\Pr(x_n = s_n \mid channel) / \Pr(x_n = a \mid channel))$$

where s_n *is the most likely symbol for* x_n

for $f = 0$ until $f = I_{max}$ or *Parity check* = 0

begin

for all m in subset k of the rows,

begin

Step 1 : $\alpha_{m,n}(a) = \gamma_n^{(f)}(a) - \beta_{m,n}^{(f)}(a)$

Step 2 : *Check node processing*

$$\beta_{m,n}^{(f+1)}(a) = \min_{\substack{(a_{n'})_{n'} \in H(m) \\ \in L(m \mid a_n = a)}} \left(\max_{n' \in H(m \setminus \{n\})} \alpha_{m,n'}(a_{n'}) \right)$$

Step 3 : $\gamma'_n(a) = \alpha_{m,n}(a) + \beta_{m,n}^{(f+1)}(a)$

Normalization

$$\gamma'_n = \min_{a \in GF(q)} \gamma'_n(a),$$

$$\gamma_n^{(f+1)}(a) = \gamma'_n(a) - \gamma'_n$$

end

end

Let GF(q)={0, α^0, α^1, ⋯, α^{q-2}} be GFs with q elements. Let s_n be the most-likely symbol of c_n; set L_n has $q-1$ elements including one zero value element. The zero value elements can be an LLR for the most-likely symbol, while the others are all positive. A $q-1$ LLR vector is used for the initialization step. For check node processing, Min-Max decoding is applied. Min-Max decoding is performed using well-known forward-backward metrics. When the variable to check node computation is performed in Step 3, a normalization step is necessary to maintain computational stability. These steps are continued until the number of iterations reaches I_{max}, or until the parity check is ok.

3.3 Bit Error Rate Performance

When a practical hardware implementation is considered, quantization becomes an important issue that needs to be solved [27]. If a large bit-size for the decoder is used, not

only will decoder performance improve, its hardware complexity will also increase. For this reason, finding the best trade-off between decoding performance and hardware complexity is a very important issue for the Min-Max algorithm.

To find the optimal bit-size, we tried different kinds of quantization schemes to find the best trade-off between the decoding performance and hardware complexity of the (225, 165) NB-LDPC code. The notation (q, f) is used to represent the quantization bit-size of the LLR values, in which q bits and f bits are the total bit size and the bit size of the fractional part, respectively.

The bit error rates (BERs) under different quantization methods and decoding algorithms are shown in Figs. 3 and 4. The simulation was performed over an additive white Gaussian noise (AWGN) channel with binary phase-shift-keying modulation. Figures 3 and 4 show the BER performance of both (5, 2) and (6, 2) input LLRs, respectively (i.e. a 2-bit fraction part is included for the input LLR). In addition, the number of maximum iterations is limited to 25. First, optimal FFT-BP and sub-optimal Min-Max decoding are performed to observe the maximum performance. For a floating-point simulation, Min-Max decoding degrades performance by about 0.1-dB compared to FFT-BP decoding. For 5-bit input LLR, the quantization level for the inner bits of the decoder is changed from 6-bit to 8-bit. As we can see in Fig. 3, 5-bit input LLR and decoder inner bit size (8, 2) show performance loss of 0.1 dB compared to floating point Min-Max decoding due to the quantization effect. When 6-bit input LLR and decoder inner bit size (8, 2) is applied, it can be seen that the proposed (8, 2) quantization fixed-point Min-Max scheme can achieve a similar error performance compared to the floating point FFT-BP scheme, and error performance of the proposed scheme is close to that of floating-point Min-Max decoding. Thus, the simulation results show that the optimum quantization scheme is 6-bit (6, 2) for input LLR and 8-bit (8, 2) for the inner bit size of the decoder.

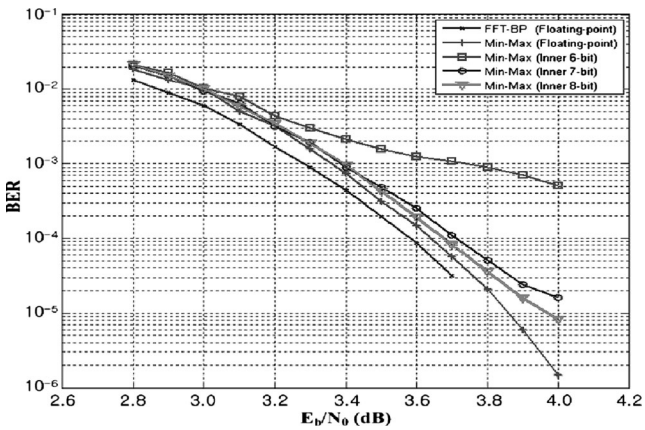


Figure 3 BERs of a (225, 165) NB-LDPC code over GF(2^4) under 5-bit input LLR.

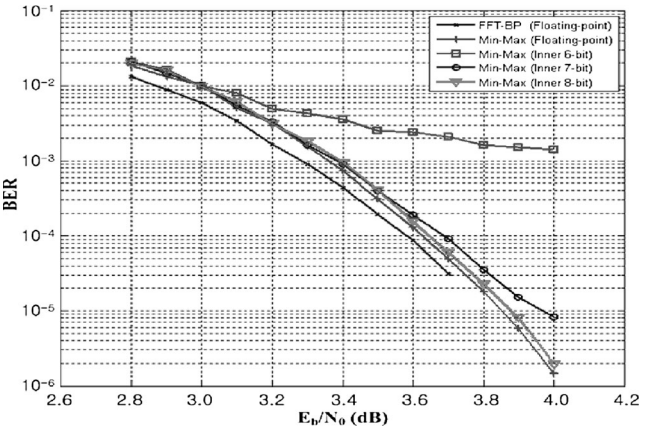


Figure 4 BERs of a (225, 165) NB-LDPC code over GF(2^4) under 6-bit input LLR.

4 Proposed Block-layered NB-LDPC Decoder Architecture

In this section, we present an efficient block-layered decoder architecture using the two-way merging Min-Max algorithm. This two-way merging Min-Max algorithm is proposed to reduce latency during forward-backward computations.

4.1 MMC and MMB Architectures

During check node processing, the min-max computations occur during the forward, backward, and merging steps. Equations (1) through (3) describe the elementary min-max computation steps; (1) is the forward metric, (2) is the backward metric, and (3) is the merging step, in which the forward and backward metrics process $F_i(a)_{i=1\ to\ d-1}$ and $B_i(a)_{i=2\ to\ d}$, respectively, where d is the check node degree.

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

$$B_i(a) = \min_{\substack{a', a'' \in GF(q) \\ a' + h_{m,n_i} a'' = a}} \left(\max\left(B_{i+1}(a'), \alpha_{m,n_i}(a'')\right)\right) \tag{2}$$

$$\beta_{m,n_i}(a) = \min_{\substack{a', a'' \in GF(q) \\ a' + a'' = -h_{m,n_i} a}} \left(\max\left(F_{i-1}(a'), B_{i+1}(a'')\right)\right) \tag{3}$$

Actually, the min-max computation finds the minimum value among the appropriate maximum values. Figure 5(a) shows the proposed min-max computation (MMC) architecture. The LLRs, $L_a[0],\cdots, L_a[q-1]$ and $L_b[0],\cdots, L_b[q-1]$, are fed to the MMC in parallel. Each L_a and L_b value is 8-bit, as determined through the quantization simulation. The output values $L_{out}[0]$, $\cdots$, $L_{out}[q-1]$ are computed in parallel. Figure 5(b) shows the min-max block (MMB) architecture, which performs one CPM computation in the **H**-matrix. The

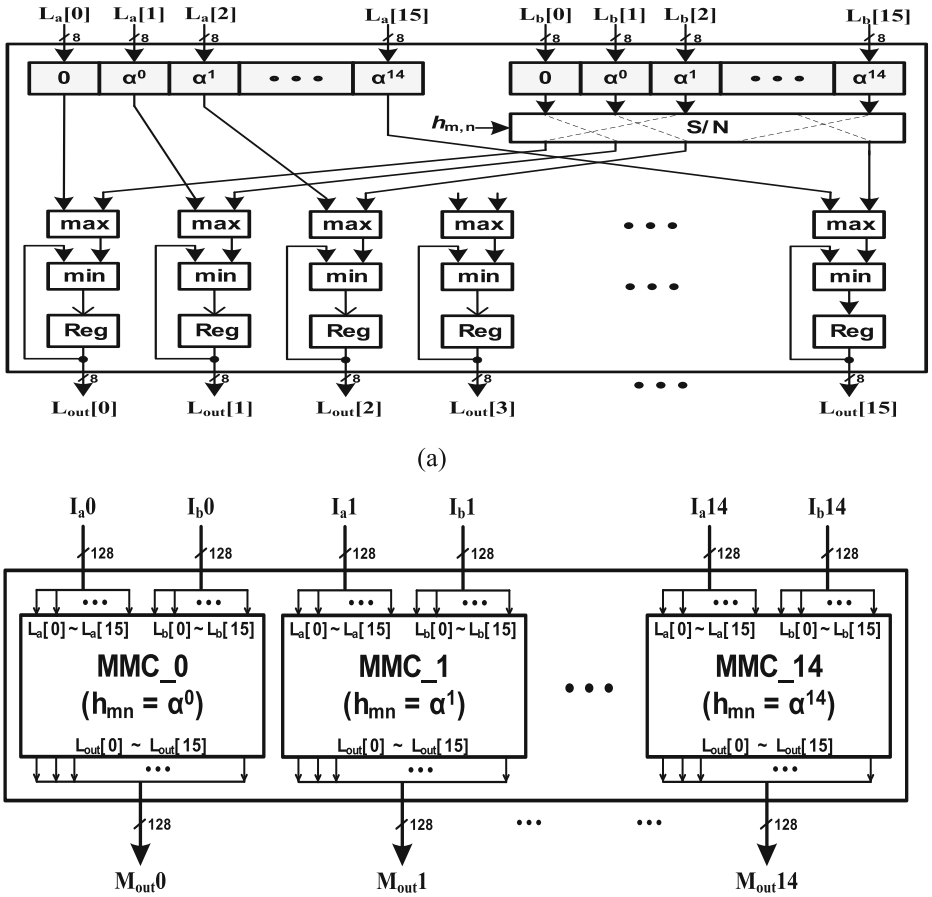


Figure 5 **a** MMC, and (**b**) MMB architectures.

MMB architecture consists of 15 MMC blocks, where $h_{m,n}$ defines a different S/N scheduling for each block.

Since our selected (225, 165) NB-LDPC code was defined over GF(2^4), all input and output vectors include 16 LLRs. The switch network (S/N) in the MMC must provide appropriate connections corresponding to $h_{m,n}$. The S/N scheme is defined by Eqs. (4) and (5) for the forward-backward metrics and the merging step, respectively.

$$a' + h_{m,n}a'' = a \ : \ for\ the\ forward{-}backward\ metrics \quad (4)$$

$$a' + a'' = -h_{m,n}a \ : \ for\ the\ merging\ step \quad (5)$$

Symbols a, a', and a'' are the corresponding GF(q) symbols. Using Eqs. (4) and (5), we can find all possible combinations of $a' - a''$. Suppose that $f(0), f(\alpha^0), \cdots, f(\alpha^{14})$ are computed when $h_{m,n}$ is α^5. The two unknown symbols a and $h_{m,n}$ are then defined, allowing us to determine all combinations by substituting the GF symbols into a' and a''. For example, when computing $f(\alpha^0)$, Eq. (4) can be rewritten as Eq. (6).

$$a' + \alpha^5 a'' = \alpha^0 \quad (6)$$

In Eq. (6), every corresponding symbol for a'' can be found by increasing a' in order. As a result, a total of 15 pairs, $0 - \alpha^{11}$, $\alpha^0 - \alpha^{14}$, $\alpha^1 - 0$, $\alpha^2 - \alpha^0$, $\cdots$, $\alpha^{14} - \alpha^2$, are possible. Figure 6 shows an example of all possible $a' - a''$ combinations when $h_{m,n}$ is α^5.

4.2 Two-way Merging CNPU Architecture

The conventional Min-Max algorithm is performed using forward-backward metrics and a merging step [10]. As shown in Eqs. (1) – (3), both the forward-backward metrics and merging step are computed sequentially. That is, the merging step is started after the forward-backward metrics are executed. Thus, $2d_c$ cycles are usually required to finish the min-max computation for each layer, creating a long latency for the decoder. Although an overlapped check node processing scheme was proposed by Zhang and Cai [16], it still requires $2d_c$ cycles with three CNUs to finish one block layer (i.e., one cycle is the computation time of the MMC). The conventional min-max decoding architecture generally requires three CNUs for the forward, backward, and merging steps.

If one more CNU is added for the merging step, the merging computation can be divided into right-merging and left-merging computations, which are referred to as a two-way merging step. In a two-way merging scheme for check node processing, the right and left-merging computations for the same check node are

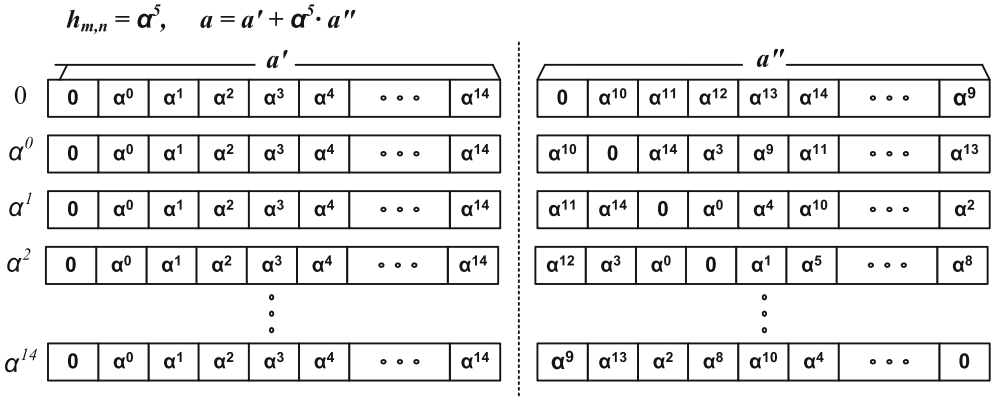


Fig. 6 Example of switch network scheduling for $h_{m,n}=\alpha^5$.

independent, and can be carried out in parallel. The proposed two-way merging Min-Max algorithm provides reduced latency compared to that of a conventional Min-Max algorithm. However, a two-way merging min-max computation requires four CNUs for the forward-backward and two-way merging computations. The two-way merging Min-Max algorithm is described as follows:

Two-way merging Min-Max algorithm

Let m be a check node and $\mathbf{H}(m)=\{n_1, n_2, n_3, ..., n_d\}$, be the set of variable nodes connected to m in a Tanner graph.

Then

$(F_i)_{i=1,d-1}$ and $(B_i)_{i=2,d}$ are computed sequentially.

Forward metrics :

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

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

Backward metrics :

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

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

Two-way merging step :

$$\beta_{m,n_1}(a)=B_2(a) \quad \text{and} \quad \beta_{m,n_d}(a)=F_{d-1}(a)$$

Left-merging step :

$$k=\lfloor d/2 \rfloor \quad \text{to} \quad k=2$$

$$\beta_{m,n_k}(a)=\min_{\substack{a',a''\in GF(q)\\ a'+a''=-h_{m,n_k}a}}(\max(F_{k-1}(a'),B_{k+1}(a'')))$$

Right-merging step :

$$j=\lfloor d/2 \rfloor+1 \quad \text{to} \quad j=d-1$$

$$\beta_{m,n_j}(a)=\min_{\substack{a',a''\in GF(q)\\ a'+a''=-h_{m,n_j}a}}(\max(F_{j-1}(a'),B_{j+1}(a'')))$$

An example of the two-way merging min-max computation procedure is shown in Fig. 7. Since the value of d_c is 14 in the proposed (225, 165) NB-LDPC decoder, F1 through F13 for the forward metrics and B14 through B2 for the backward metrics are computed sequentially at each cycle. The left-merging and right-merging are started right after the F7 and B8 computations are finished (i.e., in the middle of the forward and backward processes). That is, M7 can be calculated using (F6, B8), and M8 can be calculated using (F7, B9). Because k=7 and j=8 in the two-way merging Min-Max algorithm, two-way merging step pairs M7–M8, M6–M9, •••, M2–M13 can be computed simultaneously. The numbers provided in the small boxes in Fig. 7 indicate the processing order during the forward-backward and merging steps.

Figure 8 shows the scheduling for the two-way merging check node processing for each block layer. The forward and backward metrics are computed during $d_c - 1$ cycles. During the two-way merging process, the right and left-merging processes are started in the middle of the forward and backward processes. Thus, $(d_c/2) - 1$ cycles are needed to process both the right- and left-merging metrics. Since the forward-backward metrics and merging step are completed at the same time, a total of $d_c - 1$ cycles are taken to complete one block layer check node process. As a result, the two-way merging Min-Max algorithm provides reduced latency compared to the conventional Min-Max algorithm.

The CNU architecture consists of one S/N and one MMB, as shown in Fig. 9(a) and (b). However, the CNU architecture for the forward and backward metrics is slightly different from that of the merging step. As shown in Fig. 9(a), the forward and backward metrics are computed sequentially (i.e., F_{i-1} and B_{i+1} are fed to F_i and B_i, respectively). Thus, the CNU architectures for the forward and backward metrics have feedback loops and D flip-flops at the end of the output ports. As shown in Fig. 9(b), the CNU architecture for the merging step accepts two input vectors, and the output values are fed to the shift-registers. All input and output ports are 128-bit, as each port consists of 16 8-bit LLRs.

The S/N for the CNUs provides dynamic wiring to the MMB architecture. Since the MMB architecture processes

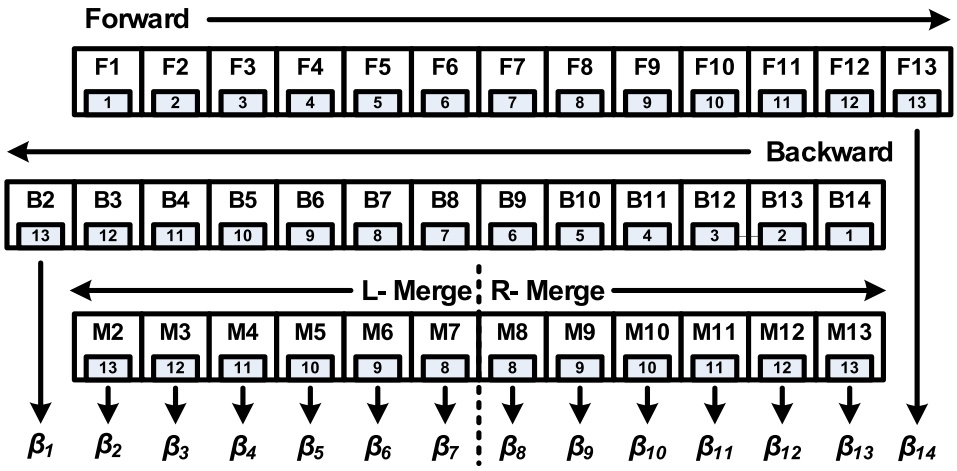


Figure 7 Computation procedure of two-way merging Min-Max algorithm.

one block column, re-ordering is required for the input ports. When the forward metrics are computed for the first block layer, the current outputs are always used for the next input. As each output is obtained from the MMCs in the MMB, appropriate input ports should be assigned during every computation cycle.

Since the proposed MMB architecture can process one block layer at a time, the S/N for the CNUs should provide an appropriate dynamic network for the forward-backward metrics and the left-right merging computation for every cycle. For the forward computation procedure, the S/N scheduling for the first block layer is shown in Fig. 10. As defined in the **H**-matrix, the processing order of the first layer is $\alpha^{14} \rightarrow \alpha^{13} \rightarrow \alpha^{7} \rightarrow \cdots \rightarrow \alpha^{7} \rightarrow \alpha^{13}$. The second layer is $\alpha^{0} \rightarrow \alpha^{14} \rightarrow \alpha^{8} \rightarrow \cdots \rightarrow \alpha^{8} \rightarrow \alpha^{14}$. Finally, the processing order of the fifteenth layer can be defined as $\alpha^{13} \rightarrow \alpha^{12} \rightarrow \alpha^{6} \rightarrow \cdots \rightarrow \alpha^{6} \rightarrow \alpha^{12}$. Note that an all-zero matrix is ignored during this processing order.

The output of MMC_14 (i.e., corresponding to α^{14}), which is included in the MMB for the forward metrics, is fed to the input of MMC_13 (α^{13}) at the first cycle. At the second computation cycle, the output of MMC_13 (i.e., corresponding to α^{13}) is fed to MMC_7 (α^{7}) as an input. The rest of the computational procedures for the third, fourth, ⋯, and fifteenth layers are shown in Fig. 10. In the same way, the S/N scheduling for the backward metrics can be defined based on the corresponding $h_{m,n}$.

Figure 11 shows the proposed check node processing unit (CNPU) architecture, which employs the two-way merging Min-Max algorithm. A total of four CNUs (i.e., two CNUs for the forward-backward metrics and two CNUs for the two-way merging step) are used to implement the proposed CNPU architecture. At every cycle, the proposed CNPU accepts two input vectors, A0 to A14 and B0 to B14. First, the forward and backward metrics are processed during d_c–1 cycles, and the computation results, (F1, F2, ⋯, F13) and (B14, B13, ⋯, B2), are then generated consecutively. During the forward and backward computations, the first seven results, (F1–F7) and (B14–B8), are stored in F_MEM and B_MEM, respectively. For this process, the MUX control signal in the dotted box is kept at "1" during the first seven cycles. The control signal is then toggled to "0", such that the remaining outputs, (F8–F13)

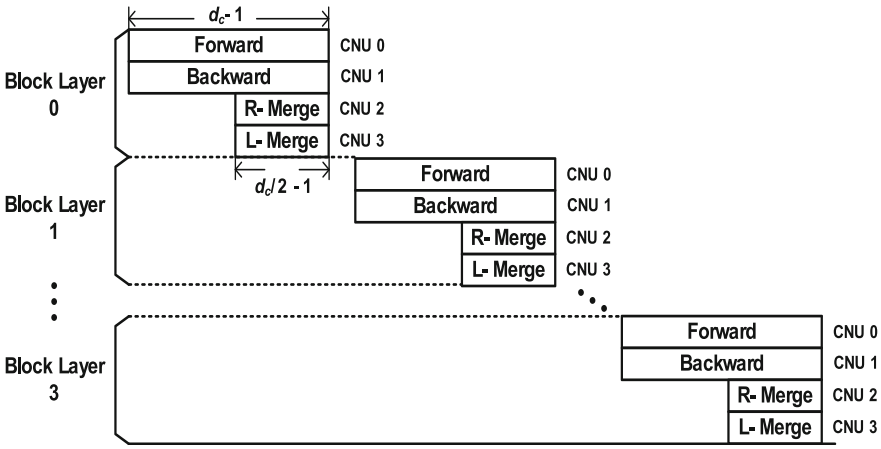


Figure 8 Scheduling of two-way merging check node processing.

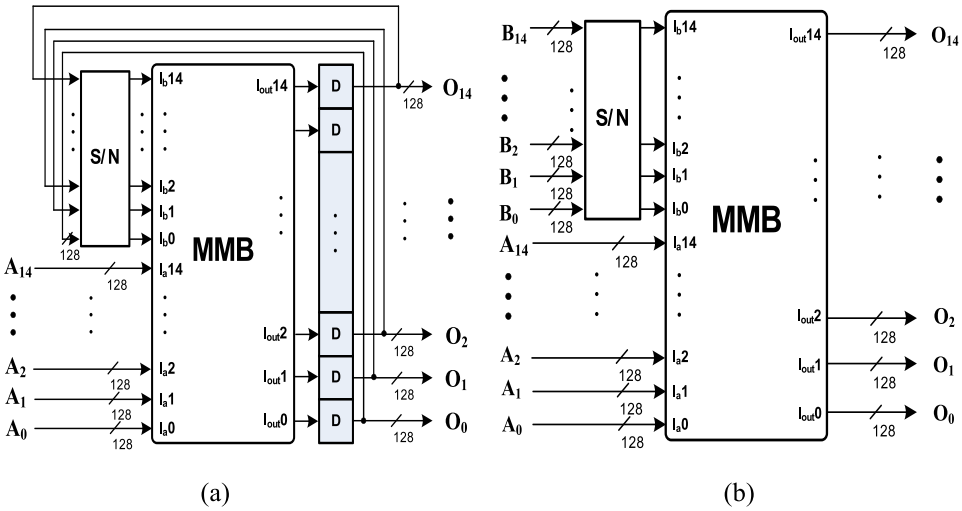


Figure 9 **a** CNU architecture for the forward and backward metrics. **b** CNU architecture for the merging step.

and (B7–B2), are directly fed to the CNUs for the left- and right-merging steps. For left- and right-merging, one of the input vectors comes from F_MEM, and the other comes from B_MEM. The output data of the left-merging step (M7–M1) and right-merging step (M8–M14) are stored in the shift registers. The memory depth of F_MEM and B_MEM are sufficient to cover $(d_c/2) - 1$, since only half of the forward and backward results need to be stored.

4.3 Overall Decoder Architecture

The proposed block-layered NB-LDPC decoder architecture, which employs the two-way merging Min-Max algorithm, is shown in Fig. 12. The proposed NB-LDPC decoder architecture consists of one CNPU architecture, 15 CN_MEMs, 15 VN_MEMs, 15 shift-registers, and the CNPU scheduler, etc. The channel extrinsic messages are inputted initially into the VN_MEMs. Because a total of 15 block columns exist, VN_MEMs (VN_MEM 0 – VN_MEM 14) are used to store the corresponding LLRs for each block. The shift-register blocks are used to buffer both the addition and subtraction operations, as well as the pipelining stage.

The CNPU accepts two data pairs, A0 to A14 and B0 to B14, at every processing cycle. The CNPU scheduler in front of the CNPU is used to manage the input scheduling. After CNPU processing, the outputs of the CNPU are stored in the CN_MEMs. At the same time, 15 parallel adders are used to check to variable node updates. A normalization step is then performed before writing to memory. As mentioned above, the normalization step is essential for maintaining computational stability. Finally, the updated data are written to the CN_MEMs; the decoder continuously runs until either the parity check is completed or the number of iterations reaches a pre-determined maximum number.

A total of 15 CN_MEMs and 15 VN_MEMs are used to store the check-to-variable and variable-to-check processing

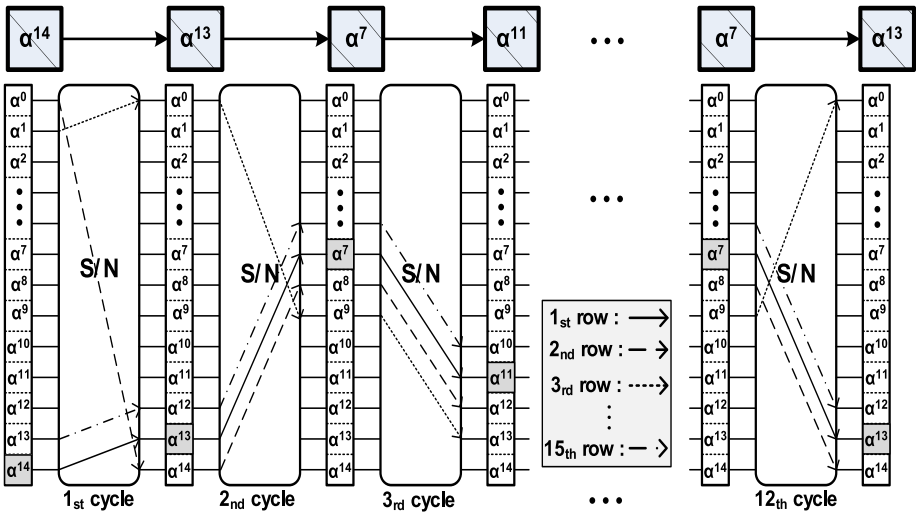


Figure 10 Switch network (S/N) scheduling for the first block layer.

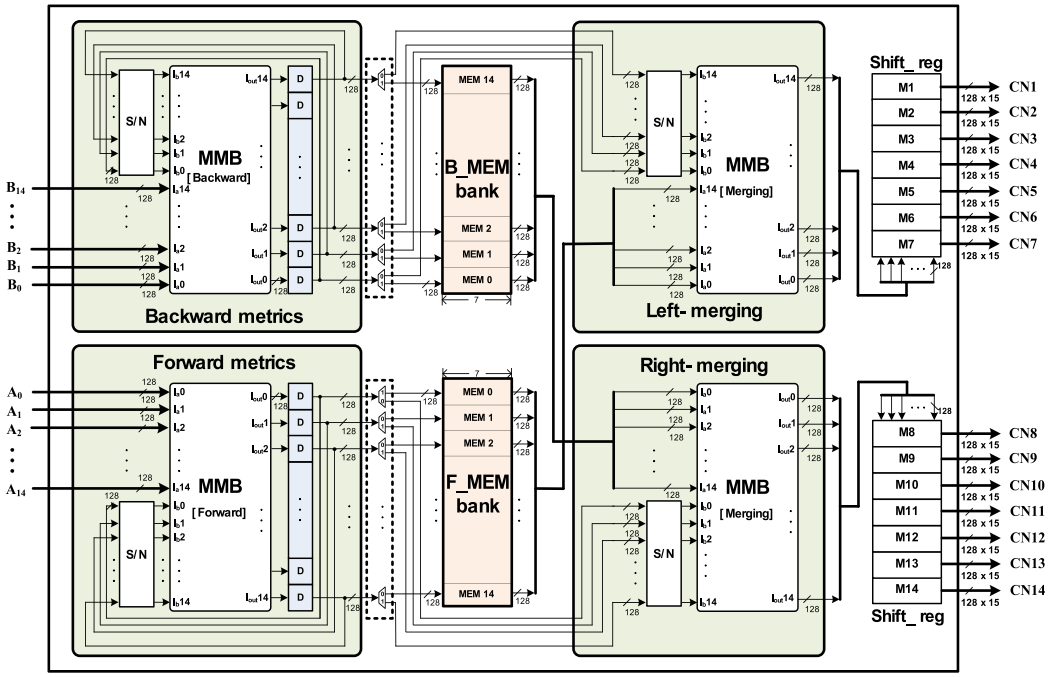


Figure 11 Proposed two-way merging CNPU architecture.

data, respectively. The size of the elementary memory for each CN_MEM and VN_MEM bank is 16×8-bit, where the memory depth is 16, since each memory block has to store all LLRs corresponding to GF(2^4).

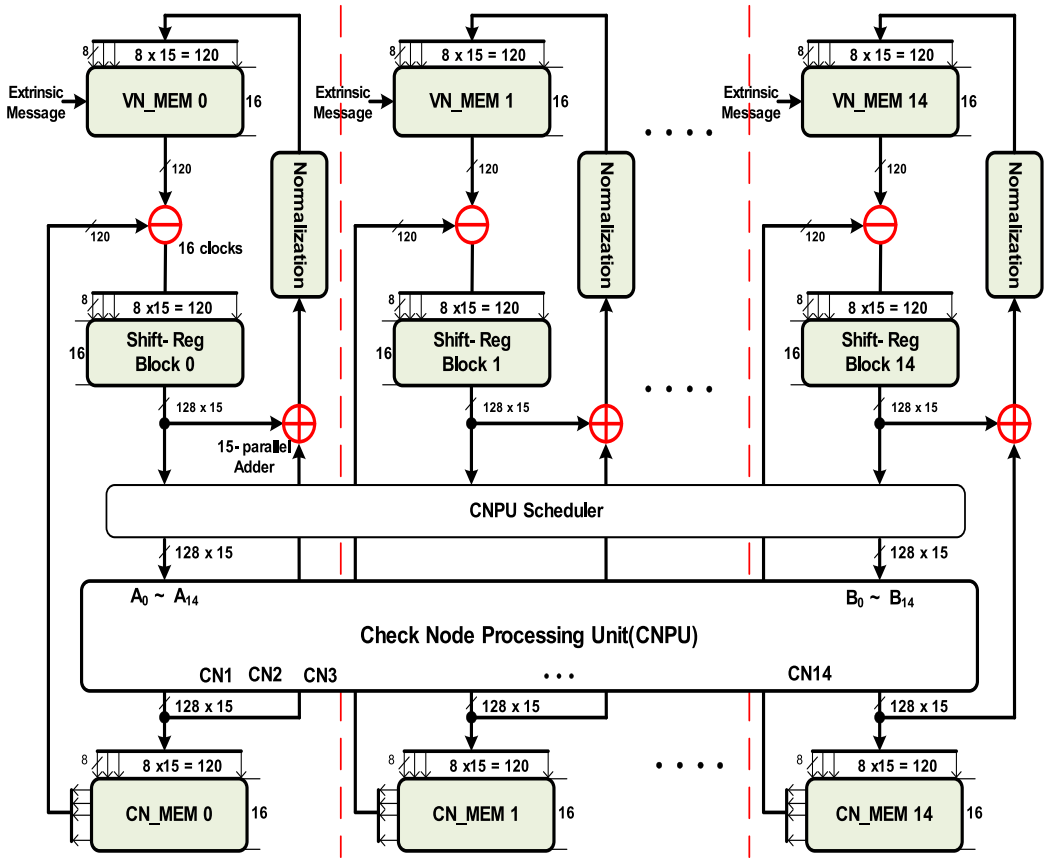


Figure 12 Proposed block-layered (225, 165) NB-LDPC decoder architecture.

5 Implementation Results and Comparisons

5.1 Implementation Results

The proposed block-layered (225, 165) NB-LDPC decoder using the two-way merging Min-Max algorithm was modeled in Verilog HDL and then simulated to verify functionality using a test pattern generated from a C simulator. After complete verification of its design functionality, the NB-LDPC decoder was synthesized using the appropriate time and area constraints. Both the simulation and synthesis steps were carried out using the SYNOPSYS design tool and a TSMC 90-nm CMOS standard cell library optimized for a 1.0 V supply voltage. The logic synthesis and the static timing analysis were performed with the typical_1.0V_25C PVT (process/voltage/temperature) corner. The synthesis result also includes the gate count of memories. The block-layered decoding method was used to improve the speed of convergence and memory requirements. For quantization, a (6, 2) input LLR and (8, 2) quantization scheme for an inner bit size were adopted. Figure 3 shows the BER performance for both floating and fixed-point simulation. The performance loss due to the quantization effect is less than 0.07 dB.

The synthesis results of the proposed (225, 165) NB-LDPC decoder architecture are shown in Table 1. The proposed NB-LDPC decoder architecture consists of one CNPU architecture, 15 CN_MEMs, 15 VN_MEMs, 15 shift-registers, and the CNPU scheduler etc., in which the CNPU architecture has almost 473 K gates, and the other component block has 97 K gates. The total number of gates for the proposed NB-LDPC decoder architecture is almost 570 K gates, which includes the estimated memory area (gate count) of 79 K-bits memory. The single CNPU architecture has one B_MEM bank and one F_MEM bank, in which one memory bank size is $128\times7\times14=12{,}544$ bits. Thus, a total of 12,544 bits$\times2=25{,}088$ bits are used for two memory banks. The total bit size of 15 VN_MEMs is 26,880 bits. Also, 15 CN_MEMs have the same 26,880 bits. As a result, $25{,}088+26{,}880+26{,}880=78{,}848$ bits of memory are used for the proposed NB-LDPC decoder architecture. The gate count of memories has been estimated from (79 K-bits SRAM core area/NAND gate area) using a 90 nm CMOS library. From the pre-layout simulation, the proposed NB-LDPC decoder architecture can operate at a clock rate of 400 MHz and has a data processing rate of 24.6 Mbps.

5.2 Comparison with Other NB-LDPC Decoders

Lin et al. [15] defined (620, 310) NB-LDPC codes over GF(2^5). The normalized code size for a B-LDPC code is (3100, 1550). This NB-LDPC decoder is implemented using a 0.18-μm CMOS process; the occupied gate counts are 14.9 M and the throughput is 60 Mbps at a clock rate of 200 MHz. The applied decoding scheme for CNP is based on the standard Min-Max algorithm. A partially parallel QC-NB-LDPC decoder architecture based on the Min-Max algorithm was proposed by Zhang and Cai [16]. A (744, 653) NB-LDPC code was also defined over GF(2^5). They proposed an efficient CNU architecture and the overlapped scheme for check node processing. From the synthesis results, the clock rate was found to reach 106 MHz with a throughput of 9.3 Mbps, targeted on a Vertex-2 Pro FPGA device. In a separate paper [17], Zhang and Cai presented an (837, 726) QC-NB-LDPC decoder over GF(2^5) using the selective-input Min-Max algorithm. They proposed a path construction algorithm for reducing the check node processing scheme. Ueng et al. [24] presented a (620, 310) NB-LDPC decoder architecture over GF(2^5) based on the selective-input layered Min-Max algorithm. They proposed a barrel-shifter-based permutation network and a minimum value filter in order to increase throughput.

Table 1 compares the proposed decoder with these earlier NB-LDPC decoders, where efficiency is defined by the throughput-to-gate-count ratio (Mbps/M gates). The throughput rate is related to check node degree and codeword size, as a larger codeword can transmit more bits at once. Since the Min-Max algorithm is operated sequentially, a larger check node causes a longer latency of the NB-LDPC decoder. Although the proposed NB-LDPC decoder shows a higher clock rate, a lower throughput rate was reported owing to a small codeword size and higher check node degree. However, it can be seen that the proposed decoder has a very low hardware complexity owing to GF(2^4), and it can provide much better efficiency than the decoders presented by Lin et al. [15], Zhang and Cai [17], and Ueng et al. [24].

5.3 Normalized Performance Comparison

NB-LDPC codes have not been selected as an FEC standard for any applications and, as shown in Table 1, the existing NB-LDPC decoders use different code sizes, Galois-fields and quantization bits. Thus it is very difficult to compare with previous decoders. The (620, 310) NB-LDPC decoder over GF(2^5) was proposed by Lin et al. [15] and Ueng et al. [24]. For a fair comparison of the proposed decoder architecture with those proposals, a normalized comparison of these designs was accomplished through an architecture-level analysis. For the (620, 310) code over GF(2^5) with $d_v=3$ and $d_c=6$, the hardware requirement and normalized decoding latency of the (620, 310) NB-LDPC decoder is listed in Table 2. According to the results in Table 2, we can observe that, if we assume that all design conditions are the same, the decoder architecture by Lin et al. [15] can be implemented using a total of 310 CNUs in which one CNU consists of several MMC blocks. Since a fully parallel structure is applied and $d_v=3$, the normalized CNU complexity

Table 1 Performance comparison with other NB-LDPC decoders.

	[15]	[16]	[17]	[24]	Proposed
Code size	(620, 310)	(744, 653)	(837, 726)	(620, 310)	(225, 165)
Code rate	0.5	0.88	0.87	0.5	0.73
Galois-field	$GF(2^5)$	$GF(2^5)$	$GF(2^5)$	$GF(2^5)$	$GF(2^4)$
(d_v, d_c)	(3, 6)	(3, 24)	(4, 27)	(3, 6)	(3, 14) or (4, 14)
Algorithm	Non-selective Min-Max	Min-Max	Selective - input Min-Max	Selective-input Min-Max	Two-way merging Min-Max
Normalized to B-LDPC	(3100, 1550)	(3720, 3265)	(4185, 3630)	(3100, 1550)	(900, 660)
Quantization bits	7	–	5	7	8
Technology	180-*nm* CMOS	Virtex-2 FPGA	N/A	90-*nm* CMOS	90-*nm* CMOS
Clock rate (MHz)	200	106	150	260	400
No. of iterations	10	15	15	10	10
Memories (K-bits)	–	935	452	405	79
Gate count (NAND)	14.9 M	N/A	1.60 M	2.14 M	570 K
Throughput (Mbps)	60	9.3	10	66.6	24.6
Efficiency (Mbps/M gates)	8.84	–	6.25	31.12	43.15

for the proposed two-way merging architecture should be $124\times3=372$. The proposed two-way merging CNPU architecture can significantly reduce the decoding latency by half, thereby improving the throughput rate. If the two-way merging Min-Max algorithm is adopted for the (620, 310) NB-LDPC decoder design, data throughput can be doubled compared to the (620, 310) NB-LDPC decoders of Lin et al. [15] and Ueng et al. [24]. Let us suppose that the fully parallel check node architecture is applied for the proposed NB-LDPC decoder architecture. The latency of the proposed MMB is $T_{\mathrm{MMB}}=32$ cycles, which is equivalent to the latency of the elementary computation unit (ECU) (T_{ECU_1}) by Lin et al. [15]. Based on the proposed two-way merging Min-Max algorithm, the normalized latency of the CNU is estimated to be $T_{CNU\_C}=(d_c-1)\,T_{MMB}=5\times32=160$ cycles, and hence, the proposed decoder takes $(T_{CNU\_C}+T_{c2v}+T_{v2c})\,N_{iter}$ cycles to accomplish N_{iter} iterations. On the other hand, the normalized latency of a CNU using the non-selective Min-Max algorithm by Lin et al. [15] is $T_{CNU\_A}=3(d_c-2)T_{ECU\_1}=384$ cycles, and the latency of the decoder is $(T_{CNU\_A}+T_{sub}+T_{add})\,N_{iter}$. In total, for GF($q$), $2q$ clock cycles are required for the C2V (T_{c2v}) and the V2C memory update time (T_{v2c}). Thus both decoders take 64 cycles to process a single block layer. For Ueng et al. [24], since each ECU takes $T_{ECU\_2}=n_m$ cycles to accomplish a single elementary step, and each CNU requires $3(d_c-2)$ elementary steps to compute C2V messages, the decoder takes $T_{CNU\_B}=3(d_c-2)T_{ECU\_2}$ cycles to process a single layer. Thus, the total latency of the NB-LDPC decoder can be expressed by $T_{CNU\_B}\times d_v\times N_{iter}$.

Table 2 Normalized performance comparison for (620, 310) NB-LDPC decoder.

	Non-selective Min-Max [15]	Selective-input Min-Max [24]	Proposed two-way Merging Min-Max
Code size	(620, 310)	(620, 310)	(620, 310)
Galois-field	$GF(2^5)$	$GF(2^5)$	$GF(2^5)$
(d_v, d_c)	(3, 6)	(3, 6)	(3, 6)
Parallelism	Fully parallel	Partially parallel	Fully parallel
No. of CNUs (Hardware Complexity)	310	-	372 (normalized)
Latency of MMB (or ECU)	$T_{ECU}=32$	$T_{ECU\_2}=n_m$	$T_{MMB}=32$
Normalized latency of CNU	$T_{CNU\_A}=3\times(d_c-2)\times T_{ECU\_1}=384$ cycles	$T_{CNU\_B}=3\times(d_c-2)\times T_{ECU\_2}$	$T_{CNU\_C}=(d_c-1)\times T_{MMB}=160$ cycles
Total latency (with 10 iter.)	$(T_{CNU\_A}+T_{sub}+T_{add})\times N_{iter}=4{,}480$ cycles	$T_{CNU\_B}\times d_v\times N_{iter}$	$(T_{CNU\_C}+T_{c2v}+T_{v2c})\times N_{iter}=2{,}240$ cycles
Technology	180-*nm* CMOS	90-*nm* CMOS	180-*nm* CMOS
Clock rate (MHz)	200	260	200
Estimated throughput (Mbps)	60	66.6	138

As shown in Table 2, if we assume that both maximum iterations (N_{iter}) and clock rate are fixed at 10 times and 200 MHz, respectively, the proposed NB-LDPC decoder based on the two-way merging Min-Max algorithm takes (160+32+32)×10=2,240 cycles. The throughput rate of the proposed decoder can be calculated as follows:

$$Throughput = \frac{f_{clk} \times N_c \times (\log_2 q) \times R}{(T_{CNU\_C} + T_{c2v} + T_{v2c}) \times N_{iter}} \quad (7)$$

where f_{clk} is the clock rate, N_c is the codeword size, R is the code rate, T_{c2v} is the C2V memory update time, T_{v2c} is the V2C memory update time, and N_{iter} is the maximum number of iterations, respectively. As a result, the estimated throughput of the proposed decoder using the two-way merging Min-Max algorithm is 138 Mbps, while that of the previous decoder using the non-selective Min-Max algorithm [15] is 60 Mbps using 180-nm CMOS technology. For Ueng et al. [24], since there is no information for the latency of the ECU, it is difficult to estimate the normalized latency of the CNU and the total latency. However, they gave the clock rate and throughput as 260 MHz and 66.6 Mbps, respectively, using 90-nm CMOS technology. Thus, it is clear that our proposed decoder can provide a much higher throughput than the decoders presented by Lin et al. [15] and Ueng et al. [24] for the same code and number of iterations.

While overlapped scheduling can achieve higher efficiency when the check node degree d_v is not small, the proposed two-way merging Min-Max algorithm can provide reduced latency. The **H**-matrix of the (744, 653) NB-LDPC code has d_v=24 and d_c=3. For the (744, 653) NB-LDPC decoder from Zhang and Cai [16], 42,464 cycles are required with 15 iterations. Similarly, the decoding latency of the proposed decoder is estimated at 36,000 clock cycles (i.e., T_{CNU}=736 clocks, and as a result, the total latency is (736+32+32)×3×15=36,000-cycles). Almost 15 % of the clock cycles are saved by applying the proposed two-way merging Min-Max algorithm. In other words, applying this algorithm to implement the NB-LDPC decoder clearly improves the decoding throughput. In addition, the observed single layer processing cycle of the CNU architecture from Ueng et al. [24] is 3 (d_c-2) n_m, which is equivalent to the standard min-max processing cycle for Lin et al. [14]. The two-way merging CNPU architecture has a latency of (d_c-1) T_{MMB} cycles, which is a significant reduction. Thus, using the proposed two-way merging Min-Max algorithm, the NB-LDPC decoder can improve throughput significantly. As observed from the comparison results, our proposed decoder outperforms previous NB-LDPC decoders.

6 Conclusion

This paper presents a novel block-layered NB-LDPC decoder architecture and an efficient design technique based on the two-way merging Min-Max algorithm. To determine the appropriate bit size for the NB-LDPC decoder, a fixed-point simulation with various finite precision levels was performed. A hardware-friendly block-layered decoding algorithm was applied for fast decoder convergence. In addition, a novel two-way merging Min-Max algorithm for check node processing was proposed to provide a much higher throughput rate compared to existing Min-Max algorithms. The proposed two-way merging Min-Max algorithm and decoder architecture shows a significant reduction in decoding latency. A (255, 165) NB-LDPC decoder over GF(2^4) was designed to demonstrate the efficiency of the proposed architecture. The proposed decoder can be a powerful FEC scheme for next-generation communication systems and memory systems.

Acknowledgments This research was partly supported by the IT R&D program of MOTIE/KEIT [10044092] and Basic Science Research Program through the NRF funded by the Ministry of Education, Science and Technology.

References

1. Gallager, R. G. (1963). *Low-density parity-check codes*. Cambridge: MIT Press. Ph.D. dissertation.
2. Davey, M. C., & MacKay, D. J. C. (1998). Low-density parity check codes over GF(q). *IEEE Communications Letters, 2*(6), 165–167.
3. MacKay, D. J. C. (1999). Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory, 45*(2). Mar. 1999.
4. Song, H., & Cruz, H. R. (2003). Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording. *IEEE Transactions on Magnetics, 39*(2), 1081–1087.
5. Barnault, L., and Declercq, D. (2003). Fast decoding algorithm for LDPC over GF(2^q). In *Proc. Inf. Theory Workshop*, Paris, France, Mar. 2003, pp. 70-73.
6. Wymeersch, H., Steendam, H., and Moeneclaey, M. (2004). Log-domain decoding of LDPC codes over GF(q). In *Proc. IEEE Int. Conf. Communications*, vol. 2, Jun. 2004, pp. 772–776.
7. Spagnol, C., Popovici, E., & Marnane, W. (2009). Hardware implementation of GF(2^m) LDPC decoders. *IEEE Transactions on Circuits and Systems I, 56*(12), 2609–2620.
8. Declercq, D., and Fossorier, M. (2005). Extended min-sum algorithm for decoding LDPC codes over GF(q). In Proc. Int. Symp, Inf. Theory, Adelaide, Australia, Sep. 2005, pp. 464–468.
9. Declercq, D., & Fossorier, M. (2007). Decoding algorithms for nonbinary LDPC codes over GF(q). *IEEE Transactions on Communications, 55*(5), 633–643.
10. Savin, V. (2008). Min-max decoding for nonbinary LDPC codes. In Proc. IEEE. Int. Symp. Inf. Theory, Toronto Canada, Jul. 2008, pp. 960–964.
11. Voicila, A., Declercq, D., Verdier, F., Fossorier, M., and Urard, P. (2007). Low-complexity Low-memory EMS algorithm for non-binary LDPC Codes. In Proc. IEEE Int. Conf. Communications, Jun. 2007, pp. 671–676.
12. Voicila, A., Declercq, D., Verdier, F., Fossorier, M., & Urard, P. (2010). Low-complexity decoding for non-binary LDPC codes in high order fields. *IEEE Transactions on Communications, 58*(5), 1365–1375.

13. Muller, O., Baghdadi, A. and Jezequel, M. (2010). Parallelism efficiency in convolutional turbo decoding. *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Nov. 2010.
14. Lin, J., Sha, J., Wang, Z., and Li, L. (2010). An efficient VLSI architecture for nonbinary LDPC decoders. IEEE Tran. Circuits and Syst. II, Exp. Briefs, vol. 57, no. 1, pp. 51–55, Jan. 2010.
15. Lin, J., Sha, J., Wang, Z., & Li, L. (2010). Efficient decoder design for nonbinary quasi cyclic LDPC codes. *IEEE Transactions on Circuits and Systems I, 57*(5), 1071–1082.
16. Zhang, X., & Cai, F. (2011). Efficient partial-parallel decoder architecture for quasi-cyclic nonbinary LDPC codes. *IEEE Transactions on Circuits and Systems I, 58*(2), 402–414.
17. Zhang, X., & Cai, F. (2011). Reduced-complexity decoder architecture for non-binary LDPC codes. *IEEE Transactions on Very Large Scale Integration Systems, 19*(7), 1229–1238.
18. Zhou, B., Zhang, L., Kang, J., Huang, Q., Tai, Y. Y., Lin, S., and Xu, M. (2008). Non-binary LDPC codes vs. Reed-Solomon codes. In Proc. Inf. Theory App. Workshop, Jan. 2008, pp. 175–184.
19. Zhou, B., Kang, J., Song, S., Lin, S., Ghaffar, K. A., & Xu, M. (2009). Construction of non-binary Quasi-cyclic LDPC codes by arrays and array dispersions. *IEEE Transactions on Communications, 57*(6), 1652–1662.
20. Peng, R., & Chen, R. R. (2008). Application of nonbinary LDPC cyclic codes to MIMO channels. *IEEE Transactions on Wireless Communications, 7*(6), 2020–2026.
21. Arabaci, M., Djordjevic, I. B., Saunders, R., & Marcoccia, R. M. (2009). High-rate nonbinary regular quasi-cyclic LDPC codes for optical communications. *IEEE/OSA Journal of Lightwave Technology, 27*(23), 5261–5267.
22. Djordjevic, I. B., Arabaci, M., & Minkov, L. L. (2009). Next generation FEC for high-capacity communication in optical transport networks. *IEEE/OSA Journal of Lightwave Technology, 27*(16), 3518–3530.
23. Chen, X., Lin, S., and Akella, V. (2012). Efficient configurable decoder architecture for nonbinary Quasi-Cyclic LDPC codes. *IEEE Transactions on Circuits and Systems I*, *59*(1), Jan. 2012.
24. Ueng, Y.-L., Leong, C.-Y., Yang, C.-J., Cheng, C.-C., Liao, K.-H., & Chen, S.-W. (2012). An efficient layered decoding architecture for nonbinary QC-LDPC codes. *IEEE Transactions on Circuits and Systems I, 59*(2), 385–398.
25. Poulliat, C., Fossorier, M., and Declercq, D. (2006). Design of non-binary LDPC codes using their binary image: algebraic properties. In Proc. Int. Symp, Inf. Theory, Seattle, USA, Jul. 2006, pp. 93–97.
26. Zhou, B., Zhang, L., Kang, J., Huang, Q., Lin, S., and Ghaffar, K. A. (2008). Array dispersions of matrices and constructions of Quasi-cyclic LDPC codes over non-binary fields. In Proc. IEEE Int. Symp. Inf. Theory, Toronto Canada, Jul. 2008, pp. 1158–1162.
27. Kim, S., Sobelman, G. E., and Lee, H. (2008) Adaptive quantization in min-sum based irregular LDPC decoder. In Proc. IEEE Int. Symp. Circuit and Syst., Seattle, USA, May. 2008, pp. 536–539.
28. Hocevar, D. E (2004). A reduced complexity decoder architecture via layered decoding of LDPC codes. In Proc. IEEE Workshop Signal Process. Syst., Oct. 2004, pp. 107–112
29. Sha, J., Wang, Z., Gao, M., & Li, L. (2009). Multi-Gb/s LDPC code design and implementation. *IEEE Transactions on Very Large Scale Integration Systems, 17*(2), 262–268.
30. Cui, Z., Wang, Z., & Liu, Y. (2009). High-throughput layered LDPC decoding architecture. *IEEE Transactions on Very Large Scale Integration Systems, 17*(4), 582–587.
31. Kim, S., Sobelman, G. E., & Lee, H. (2011). A reduced-complexity architecture for LDPC layered decoding schemes. *IEEE Transactions on Very Large Scale Integration Systems, 19*(6), 1099–1103.

Chang-Seok Choi Chang-Seok Choi received Ph. D and M. S. degrees, both in information & communication engineering from Inha University, Incheon, Korea, in 2012 and 2007, respectively. Since January 2012, he has been with Chief Technology Office, LG Electronics, where he is currently Senior Research engineer. His research interests VLSI architecture design for digital signal processing and communications.

Hanho Lee Hanho Lee received Ph.D. and M.S. degrees, both in Electrical & Computer Engineering, from the University of Minnesota, Minneapolis, USA, in 2000 and 1996, respectively. In 1999, he was a Member of Technical Staff-1 at Lucent Technologies, Bell Labs, Holmdel, New Jersey. From April 2000 to August 2002, he was a Member of Technical Staff at the Lucent Technologies (Bell Labs Innovations), Allentown, where he was responsible for the development of high-performance DSP multi-processor architecture. From August 2002 to August 2004, he was an Assistant Professor at the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, USA. Since August 2004, he has been with the Department of Information and Communication Engineering, Inha University, where he is currently Professor. He was a visiting researcher at Electronics and Telecommunications Research Institute (ETRI), Korea, in 2005. From August 2010 to August 2011, he was a visiting scholar at Bell Labs, Alcatel-Lucent, Murray Hill, New Jersey, USA. He authored over 120 papers and also holds 16 patents. He received the best paper award at the International SoC Design Conference (ISOCC) in 2006 and 2009. He served the Secretary General of the IEEE ISCAS2012, Special Session Chair of the ISOCC2012 and Guest Editor of the Journal of Electrical and Computer Engineering. He is an Associate Editor of the Journal of Semiconductor Technology & Science. He has served the Technical Committee Member, IEEE Signal Processing Society, Design and Implementation of Signal Processing Systems (DISPS) (Jan. 2011 ~ Dec. 2013). He is a Technical Committee Member, 2013 IEEE Workshop on Signal Processing Systems (SiPS2013). His research interest includes VLSI architecture design for digital signal processing, communications, and forward error correction architectures.