IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 29, 2018, accepted March 3, 2018, date of publication March 13, 2018, date of current version April 4, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2815640

Area-Optimized Fully-Flexible BCH Decoder for Multiple GF Dimensions

BYEONGGIL PARK[1], (Student Member, IEEE), JONGSUN PARK[2], (Senior Member, IEEE), AND YOUNGJOO LEE[3], (Member, IEEE)

[1]Engineer in Foundry Division, Samsung Electronics, Hwaseong 18448, South Korea
[2]Electrical Engineering Department, Korea University, Seoul 136-701, South Korea
[3]Electrical Engineering Department, Pohang University of Science and Technology, Pohang 37673, South Korea

Corresponding author: Youngjoo Lee (youngjoo.lee@postech.ac.kr)

This work was supported by the National Research Foundation of Korea through the Korean government (Ministry of Science and ICT) under Grant 2016R1C1B1007593 and Grant 2016R1A2B4015329.

ABSTRACT Recently, there are increasing demands for fully flexible Bose–Chaudhuri–Hocquenghem (BCH) decoders, which can support different dimensions of Galois fields (GF) operations. As the previous BCH decoders are mainly targeting the fixed GF operations, the conventional techniques are no longer suitable for multiple GF dimensions. For the area-optimized flexible BCH decoders, in this paper, we present several optimization schemes for reducing hardware costs of multi-dimensional GF operations. In the proposed optimizations, we first reformulate the matrix operations in syndrome calculation and Chien search for sharing more common sub-expressions between GF operations having different dimensions. The cell-based multi-m GF multiplier is newly introduced for the area-efficient flexible key-equation solver. As case studies, we design several prototype flexible BCH decoders for digital video broadcasting systems and NAND flash memory controllers managing different page sizes. The implementation results show that the proposed fully-flexible BCH decoder architecture remarkably enhances the area-efficiency compared with the conventional solutions.

INDEX TERMS Bose-Chaudhuri-Hocquenghem (BCH) decoder, Galois-field (GF) arithmetic, optimization, low complexity, VLSI design.

I. INTRODUCTION

In digital communication and storage systems, error correction codes (ECCs) are essential to improve data reliability. Due to the strong and guaranteed error-correcting performance with reasonable hardware costs, among the various ECCs, Bose–Chaudhuri–Hocquenghem (BCH) codes have been continuously adopted in practical applications [1]–[10]. Targeting the message frame of k bits, an (n, k, t) BCH code is defined to recover up to t random bit errors in the transferred codeword of n bits. To find the t error locations, three decoding steps are generally used as illustrated in Fig. 1, i.e., syndrome calculation (SC) block, key-equation solver (KES) stage, and Chien search (CS) step [11]. In the last decade, targeting the recent high-speed applications, the high-throughput BCH decoders associated with the massive parallelism have been developed and optimized in numerous literatures [12]–[17]. Although the previous optimization techniques are quite effective to relax the hardware complexity, they are only applied to the single BCH code, where all the internal arithmetic operations are based on the fixed dimension of Galois field (GF). Recently, specifications of error-correction capabilities and the data lengths are increasingly diversified, and it is necessary for making BCH decoders be flexible to cope with the increased requirements of reconfigurability. Targeting the multi-standard solutions, only few studies have been reported for sharing the processing elements in multi-t BCH decoders, which can change their correcting conditions within only the same GF dimension m [18]–[20]. For the sake of simplicity, therefore, we denote this multi-t fixed-m BCH decoder as the partially-flexible BCH decoder.

In order to provide more flexibilities on error-correction capabilities, it is more common to change the dimension of GF, leading to the fully-flexible BCH decoder. For example, the NAND flash memory uses a variety of page sizes such as 512, 1024, 2048, and 4096 bytes, BCH decoders in the controller system are required to operate in GF(2^{13}), GF(2^{14}), GF(2^{15}) and even GF(2^{16}) for each page size [6]. For the case

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

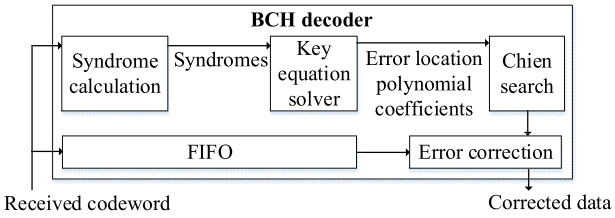

FIGURE 1. The conventional BCH decoder having 3 processing steps [11].

of digital video broadcasting (DVB) systems, in additions, recent specifications require two kinds of BCH decoders to support different sizes of frame structures [7]–[9]. For the conventional fully-flexible BCH decoder architecture, which can change both t and m, multiple BCH decoders targeting different GF dimensions are designed separately and integrated into the same system, drastically increasing the hardware complexity. Resource-sharing schemes may relax the hardware costs by reutilizing the registers and multiplexors at the higher-m BCH decoder [21], [22]. However, the area saving caused by this technique is naturally limited as there is no consideration in GF operators, which dominate the overall hardware complexity.

For the area-efficient fully-flexible BCH decoder architecture, this paper presents several optimization techniques for supporting multiple GF dimensions. More precisely, we reformulate the parallel SC and CS architectures having different field dimensions into a single matrix operator to find the maximum number of common sub-expressions (CSEs). Based on the iterative search [25], the hardware resources for different GF dimensions can be shared so that the complexities of parallel SC and CS blocks are minimized. For general GF multiplications, in addition, we newly define the cell-based multiplier architecture to accept arbitrary field dimensions. Based on the proposed universal GF multiplier, the area-time complexity of folded-KES block is remarkably reduced compared to the previous register-sharing approach. For validating the proposed optimization schemes, we also designed prototype decoders in 65nm CMOS process, targeting DVB systems and NAND flash memory controllers. Implementation results show that the proposed work successfully realizes the area-optimized fully-flexible BCH decoder, which enhances the area-efficiency by more than two times compared to the previous approaches.

The rest of this paper is organized as follows. Section II describes the methods and drawbacks of the previous architectures supporting multiple GF dimensions. Section III describes the proposed methods to maximize the effects of CSE-sharing for parallel SC and CS blocks. Section VI shows the architecture of the proposed multi-m GF multiplier for the folded-KES block. Implementation results are presented and analyzed in Section V, and the conclusions are finally drawn in Section VI.

II. THE PREVIOUS FULLY-FLEXIBLE BCH DECODERS

In order to support different BCH decoding operations for multiple GF dimensions, it is generally to design the dedicated BCH decoders for each GF dimension, requiring a huge amount of hardware resources. Only few literatures have been reported the area-efficient solutions for flexible decoder architectures [21], [22]. Targeting j different GF dimensions, in general, the previous works are based on the sharing methods of pipelined registers as depicted in Fig. 2. More precisely, the previous fully-flexible BCH decoder is firstly designed for supporting the highest GF dimension, i.e., m_1 in the figure, and then the decoders for lower dimensions are added by sharing the internal pipelined registers, eliminating additional storing elements [21], [22]. This register-sharing method is effective to reduce the amount of sequential logics as the decoders for lower dimension in general necessitate the fewer number of registers.

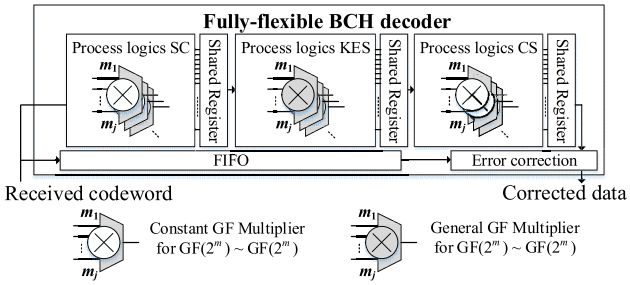

FIGURE 2. The previous fully-flexible BCH decoder architecture with register-sharing method [21], [22].

For combinational logics, as shown in Fig. 2, the GF operators, i.e., constant and general GF multiplications, cannot be shared and should be utilized individually. To overcome the limitation of register sharing in previous study, that tried to work on sharing GF computing logics between different GF fields [21]. As shown in Fig. 3(a), the shift-and-add parts of general multiplier are shared for various GF dimensions, and the following modulo parts are individually designed for each dimension. Note that the final result is selected from the outputs of different field dimensions by using the j-to-1 multiplexor. As the complexity of the modulo process dominates the overall multiplier, unfortunately, the effects of sharing are quite limited, still necessitating a huge amount of hardware resources. Moreover, this method cannot be used for reducing the hardware complexity of constant GF multipliers, which require numerous XOR operations [13]. As shown in Fig. 3(b), therefore, the previous constant multiplier covering multiple GF dimensions utilizes a dedicated GF multiplier for each dimension [21].

Considering the complexity breakdown of each building block, as depicted in Fig. 4, the impact of the previous only register sharing method is naturally limited as the portion of sequential parts is negligible compared to that of GF operators. Note that the complexity shown in Fig. 4 is based on an 8-parallel (65535, 65343, 12) BCH decoder, which is designed in a 65nm CMOS process at the speed of 500 MHz. In order to provide more realistic analysis, in addition, we apply the recent optimization techniques, which can reduce the complexity of the fixed-m decoder architecture [13], [16], [23]. Considering the various demands from

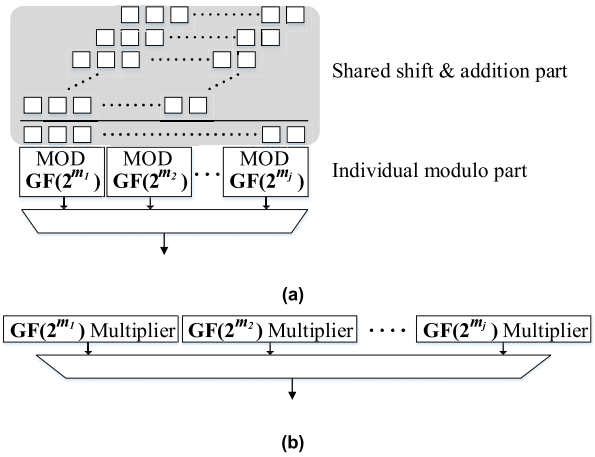

FIGURE 3. The detailed multiplier architectures of the previous flexible BCH decoder [21], [22]. (a) The general multiplier having limited shared units and (b) the constant GF multiplier with no sharing parts.

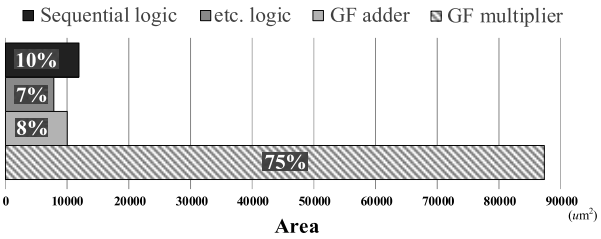

FIGURE 4. The complexity breakdown of a (65535, 65343, 12) BCH decoder.

practical applications [6]–[10], [24], therefore, it is urgently required to develop new area-efficient design methods for the fully-flexible BCH decoder architectures. For the sake of simplicity, in Section III and IV, we first describe new optimization schemes for area-efficient dual-m BCH decoders. Note that the proposed methods are easily extended to support more GF dimensions, even providing the fully-flexible solution.

III. THE PROPOSED SC AND CS ARCHITECTURES FOR MULTIPLE GF DIMENSIONS

It is widely known that constant GF multiplications are actively used in the processing of SC and CS stages [11]. Similar to the reformulated matrix operations for the fixed-m SC and CS blocks in [13] and [16], in this section, we propose construction steps for a single matrix operation supporting different GF dimensions at the same time. As the common sub-expressions (CSEs) of the given matrix can be shared during the realization step [25], the proposed single-matrix form naturally relaxes the hardware overheads by maximizing the search area of CSEs among the different GF dimensions.

In order to construct the single matrix form for dual-m parallel SC and CS blocks, firstly, it is necessary to find an efficient way to construct matrix forms including different GF dimensions. For the given GF dimension of m, $y = x\alpha^i$ is expressed as

$$\begin{aligned}\mathbf{y} &= \begin{bmatrix} \mathrm{x}_0 & \mathrm{x}_1 & \cdots & \mathrm{x}_{m-1} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \alpha_0^i & \alpha_1^i & \cdots & \alpha_{m-1}^i \\ \alpha_0^{i+1} & \alpha_1^{i+1} & \cdots & \alpha_{m-1}^{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{i+m-1} & \alpha_1^{i+m-1} & \cdots & \alpha_{m-1}^{i+m-1} \end{bmatrix} \\ &= \mathbf{x} \times \mathbf{A}_i(m). \end{aligned} \tag{1}$$

where $\mathbf{y}$ and $\mathbf{x}$ represent elements in GF(2^m), and an $m \times m$ matrix $\mathbf{A}_i(m)$ stands for the corresponding constant multiplication over GF(2^m). Considering the previous register-sharing method for the dual-m BCH decoder [21], [22], the input operand $\mathbf{x}$ in (1) can belong to either GF(2^{m_1}) or GF(2^{m_2}). By assuming $m_1 > m_2$, we set the unused bits for the smaller GF dimension to be zero, i.e., $x_i = 0$ for $m_2 \le i < m_1$. Hence, two constant multiplications can be applied to the same input $\mathbf{x}$ as follows.

$$\begin{aligned}&\begin{bmatrix} \mathbf{y}(m_1) & \mathbf{y}(m_2) \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{x}_0 & \mathrm{x}_1 & \cdots & \mathrm{x}_{m_1-1} \end{bmatrix} \\ &\times \begin{bmatrix} \alpha_{m_1,0}^i & \cdots & \alpha_{m_1,m_1-1}^i & \alpha_{m_2,0}^i & \cdots & \alpha_{m_2,m_2-1}^i & 0 & \cdots \\ \alpha_{m_1,0}^{i+1} & \cdots & \alpha_{m_1,m_1-1}^{i+1} & \alpha_{m_2,0}^{i+1} & \cdots & \alpha_{m_2,m_2-1}^{i+1} & 0 & \cdots \\ & & & \vdots & \ddots & \vdots & \vdots & \ddots \\ \vdots & \ddots & \vdots & \alpha_{m_2,0}^{i+m_2-1} & \cdots & \alpha_{m_2,m_2-1}^{i+m_2-1} & 0 & \cdots \\ & & & 0 & \cdots & 0 & 0 & \cdots \\ \alpha_{m_1,0}^{i+m_1-1} & \cdots & \alpha_{m_1,m_1-1}^{i+m_1-1} & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix} \\ &= \mathbf{x} \times \begin{bmatrix} \mathbf{A}_i(m_1) & \mathbf{A}_i(m_2) \end{bmatrix}. \end{aligned} \tag{2}$$

where $\mathbf{y}(m_1)$ and $\mathbf{y}(m_2)$ are the multiplication results over GF(2^{m_1}) and GF($2^{m_2}2^{m_2}$), respectively. In (2), more precisely, an $m_1 \times m_1$ binary matrix $\mathbf{A}_i(m_1)$ is used for calculate the $1 \times m_1$ binary matrix $\mathbf{y}(m_1)$ in GF(2^{m_1}). Note that $\mathbf{A}_i(m_2)$ is also an $m_1 \times m_1$ binary matrix even though it is targeting the constant multiplication over GF(2^{m_2}). To represent the valid result, $\mathbf{y}(m_2)$ contains m_2 bits from its left-most position, and the rest of positions are filled with zeros to make the same matrix size to $\mathbf{y}(m_1)$. This assumption is reasonable as we will also apply the register-sharing architecture in Fig. 2 to our fully-flexible decoder, which prepares all the registers based on the largest GF dimension, m_1. In other words, all the data in the lower dimensions are zero-extended in our architecture. As shown in (2), two multiplicands can be merged into a single matrix form whose size is $m_1 \times (m_1 + m_1)$ binary matrix. As the merged matrix allows to find CSEs between $\mathbf{A}_i(m_1)$ and $\mathbf{A}_i(m_2)$, it is natural that the hardware complexity for realizing (2) is relaxed significantly, compared to the individual realizations of two different constant GF multiplications.

Fig. 5 exemplifies how the proposed optimization method reduces the number of XOR operations in the flexible constant GF multiplier. In order to design the previous flexible multiplier performing $y = x\alpha^{15}$ over both GF(2^8) and GF(2^{10}), as depicted in Fig. 5(a), two multiplications are

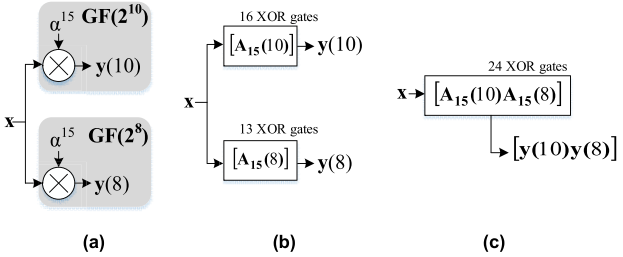

FIGURE 5. (a) Constant GF multipliers for GF(2^{10}) and GF(2^8), (b) the matrix form of each multiplier [16], (c) the proposed single matrix form.

individually realized by using matrix operations denoted as $\mathbf{A}_{15}(8)$ and $\mathbf{A}_{15}(10)$ in Fig. 5(b), respectively. More precisely, two matrices are expressed as

$$\mathrm{A}_{15}(10) = \begin{bmatrix} 0&0&0&0&0&1&0&0&1&0\\ 0&0&0&0&0&0&1&0&0&1\\ 1&0&0&1&0&0&0&1&0&0\\ 0&1&0&0&1&0&0&0&1&0\\ 0&0&1&0&0&1&0&0&0&1\\ 1&0&0&0&0&0&1&0&0&0\\ 0&1&0&0&0&0&0&1&0&0\\ 0&0&1&0&0&0&0&0&1&0\\ 0&0&0&1&0&0&0&0&0&1\\ 1&0&0&1&1&0&0&0&0&0 \end{bmatrix},$$

$$\mathrm{A}_{15}(8) = \begin{bmatrix} 0&1&1&0&0&1&0&0\\ 0&0&1&1&0&0&1&0\\ 0&0&0&1&1&0&0&1\\ 1&0&1&1&0&1&0&0\\ 0&1&0&1&1&0&1&0\\ 0&0&1&0&1&1&0&1\\ 1&0&1&0&1&1&1&0\\ 0&1&0&1&0&1&1&1 \end{bmatrix}. \quad (3)$$

Note that the input vector $\mathbf{x}$ in Fig. 5(b) consists of 10 bits for targeting the larger dimension GF(2^{10}), but only 8 bits are involved to process the multiplications over GF(2^8). Based on (3) 23 and 14 XOR gates are used for the straightforward realization of $\mathbf{A}_{15}(10)$ and $\mathbf{A}_{15}(8)$, respectively. The iterative searching algorithm in [16] shares CSEs of each matrix operator as many as possible, leading to the following results.

$$\mathrm{A}_{15}(10)_{\mathrm{CSE}}: \begin{bmatrix} 0&0&0&0&0&1&0&0&1&0&0\\ 0&0&0&0&0&0&1&0&0&1&0\\ 0&0&0&0&0&0&0&1&0&0&1\\ 0&1&0&0&1&0&0&0&1&0&0\\ 0&0&1&0&0&1&0&0&0&1&0\\ 1&0&0&0&0&0&1&0&0&0&0\\ 0&1&0&0&0&0&0&1&0&0&0\\ 0&0&1&0&0&0&0&0&1&0&0\\ 0&0&0&1&0&0&0&0&0&1&0\\ 0&0&0&0&1&0&0&0&0&0&1\\ 1&0&0&1&0&0&0&0&0&0&0 \end{bmatrix},$$

$$\mathrm{A}_{15}(8)_{\mathrm{CSE}}: \begin{bmatrix} 0&1&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0&0&1&0&0\\ 0&0&0&0&1&0&0&1&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1&1&0&0&0\\ 1&0&0&0&0&0&1&0&1&0&0&0\\ 0&0&0&1&0&1&0&1&0&0&1&0\\ 0&0&0&0&1&0&0&0&0&0&0&1\\ 0&0&1&1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&1&0&0&0&0&0\\ 0&0&1&0&0&1&0&0&0&0&0&0 \end{bmatrix}. \quad (4)$$

Note that the optimized matrices in (4), denoted as $\mathbf{A}_{15}(10)_{\mathrm{CSE}}$ and $\mathbf{A}_{15}(8)_{\mathrm{CSE}}$, necessitate 16 and 13 XOR gates, respectively. Therefore, total 29 XOR gates are used to realize the previous flexible constant multiplier in Fig. 5(b). In the proposed scheme, as depicted in Fig. 5(c), we reformulate two matrix operations in (3) into a single matrix operator

$$\left[\mathrm{A}_1(10)\ \ \mathrm{A}_{15}(8)\right]_{\mathrm{CSE}}: \begin{bmatrix} 0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&1&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\ 1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0 \end{bmatrix}. \quad (6)$$

as follows.

$$\begin{bmatrix} \mathbf{A}_{15}(10) & \mathbf{A}_{15}(8) \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&1&0&0&1&0&0&1&1&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&1&0&0&1&1&0&0&1&0&0&0 \\ 1&0&0&1&0&0&0&1&0&0&0&0&0&1&1&0&0&1&0&0 \\ 0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&1&0&0&0&0 \\ 0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&1&0&0&0 \\ 1&0&0&0&0&0&1&0&0&0&0&0&1&0&1&1&0&1&0&0 \\ 0&1&0&0&0&0&0&1&0&0&1&0&1&0&1&1&1&0&0&0 \\ 0&0&1&0&0&0&0&0&1&0&0&1&0&1&0&1&1&1&0&0 \\ 0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}. \tag{5}$$

As described in (2), the merged matrix is zero-extended so that the portion corresponding to $\mathbf{A}_{15}(8)$ becomes equal to the size of $\mathbf{A}_{15}(10)$. By applying the iterative searching algorithm, similar to (4), we can share all the CSEs to generate the optimized matrix in (6), as shown at the bottom of the previous page.

Note that only 24 XOR gates are enough to implement the single matrix operation in (6), relaxing the complexity by 17% compared to the previous structure. As the matrix becomes dense for supporting the larger multiplicand, moreover, the proposed scheme is more effective to reduce the overall complexity with increased number of common terms. Note that this concept can be further extended to optimize the parallel dual-m SC and CS architectures, which will be described in the following subsections.

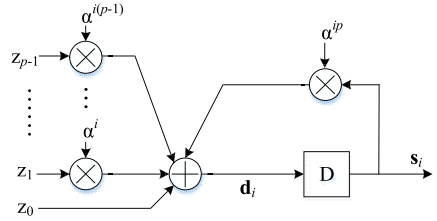

FIGURE 6. The conventional p-parallel SC unit [11].

A. THE PROPOSED PARALLEL DUAL-m SC ARCHITECTURE

When the n-bit BCH codeword $(r_0, r_1, r_2, \ldots, r_{n-1})$ is received, the i-th syndrome $\mathbf{s}_i$ $(0 < i \leq 2t)$ is calculated as follows [11].

$$\mathbf{s}_i = R(\alpha^i) = \sum_{j=0}^{n-1} r_j \alpha^{ij} = r_0 + r_1\alpha^i + \cdots + r_{n-1}\alpha^{i(n-1)}. \tag{7}$$

In the conventional p-parallel SC architecture, in general, $2t$ different syndrome units are utilized and, in each processing cycle, each of syndrome unit accepts p bits from the received codeword, which is denoted as $z_0, z_1, \ldots, z_{p-1}$ in Fig. 6. Hence, the p-parallel SC architecture necessitates n/p processing cycles to compute $2t$ syndromes. To reduce the hardware complexity of p-parallel fixed-m SC, only t odd-indexed SC units are introduced and other even-indexed

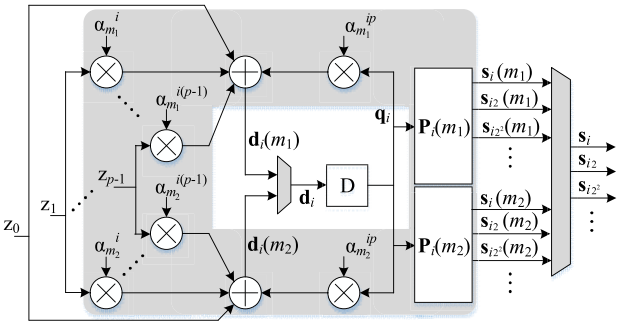

FIGURE 7. The proposed dual-m p-parallel SC unit. GF operations in the shared region are grouped to generate a single matrix operator.

syndromes are calculated by using power-operations without using the original syndrome units [15]. In additions, all the power operations and constant GF multiplications can be reorganized into a single matrix to share the maximum number of CSEs [25].

Based on the previous optimizations, the proposed p-parallel i-th dual-m SC unit, which supports two GF dimensions m_1 and m_2 $(m_1 > m_2)$, is exemplified in Fig. 7, where i is an odd number less than $2t$. Note that GF operations in the shadowed region in Fig. 7 are organized into a single matrix operator to share their internal logics. The proposed syndrome unit contains one temporal register having m_1 bits, where the currently-stored value and the next value for the following cycle are denoted as $\mathbf{q}_i$ and $\mathbf{d}_i$, respectively. Based on the p-bit input $(z_0, z_1, \ldots, z_{p-1})$ and m_1-bit stored-data $\mathbf{q}_i$, the proposed dual-m SC unit computes two candidates of next value, denoted as $\mathbf{d}_i(m_1)$ and $\mathbf{d}_i(m_2)$. According to the current GF dimension, $\mathbf{d}_i$ is determined by selecting one of these two candidates over GF(2^{m_1}) and GF(2^{m_2}), and the stored-value $\mathbf{q}_i$ becomes the i-th syndrome $\mathbf{s}_i$ after p/n processing cycles. For the case of smaller dimension, m_2, some parts of $\mathbf{s}_i$ are filled with zeros to make the regular data size be used at the fully-flexible BCH decoder. Similar to the works in [15], even-indexed syndromes can be calculated after obtaining odd-indexed $\mathbf{s}_i$ based on the power operations. As depicted in Fig. 7, two matrices, denoted as $\mathbf{P}_i(m_1)$ and $\mathbf{P}_i(m_2)$, are reserved to perform the power operations for each GF dimension and the final syndromes are generated by selecting the proper dimension between $\mathbf{s}_x(m_1)$ and $\mathbf{s}_x(m_2)$.

To construct a huge but single matrix operation for the proposed p-parallel dual-m SC unit, we first formulate the matrix operation for calculating the next value over GF(2^{m_1}), $\mathbf{d}_i(m_1)$, as follows.

$$\begin{aligned} \mathbf{d}_i(m_1) &= \begin{bmatrix} z_0\mathbf{J}(m_1) & z_1\mathbf{J}(m_1) & \cdots & z_{p-1}\mathbf{J}(m_1) & \mathbf{q}_i \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{A}_0(m_1) \\ \mathbf{A}_i(m_1) \\ \vdots \\ \mathbf{A}_{(p-1)i}(m_1) \\ \mathbf{A}_{ip}(m_1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z} & \mathbf{q}_i \end{bmatrix} \times \begin{bmatrix} \mathbf{C}_i(m_1) \\ \mathbf{A}_{ip}(m_1) \end{bmatrix} \end{aligned} \tag{8}$$

where $\mathbf{J}(m_1)$ represents $1 \times m_1$ all-ones matrix. Considering the zero-extended GF processing over GF(2^{m_2}), two internal candidates of $\mathbf{d}_i$, i.e. $\mathbf{d}_i(m_1)$ and $\mathbf{d}_i(m_2)$, are calculated by using the same inputs as follows.

$$\begin{bmatrix} \mathbf{d}_i(m_1) & \mathbf{d}_i(m_2) \end{bmatrix} = \begin{bmatrix} \mathbf{z} & \mathbf{q}_i \end{bmatrix} \times \begin{bmatrix} \mathbf{C}_i(m_1) & \mathbf{C}_i(m_2) \\ \mathbf{A}_{ip}(m_1) & \mathbf{A}_{ip}(m_2) \end{bmatrix}. \tag{9}$$

By grouping t odd-indexed syndrome units, it is possible to generate a single matrix operator as follows.

$$\begin{bmatrix} \mathbf{D}_{\text{odd}}(m_1) & \mathbf{D}_{\text{odd}}(m_2) \end{bmatrix} = \begin{bmatrix} \mathbf{z} & \mathbf{Q}_{\text{odd}} \end{bmatrix} \times \begin{bmatrix} \mathbf{C}_{\text{odd}}(m_1) & \mathbf{C}_{\text{odd}}(m_2) \\ \mathbf{X}_{\text{D}}(m_1) & \mathbf{X}_{\text{D}}(m_2) \end{bmatrix}. \tag{10}$$

where $\mathbf{D}_{\text{odd}}(m_1)$, $\mathbf{Q}_{\text{odd}}$, and $\mathbf{C}_{\text{odd}}(m_1)$ are constructed by serially appending the corresponding matrices in each odd-indexed syndrome unit as follows.

$$\mathbf{C}_{\text{odd}}(m_1) = \begin{bmatrix} \mathbf{C}_1(m_1) & \mathbf{C}_3(m_1) & \cdots & \mathbf{C}_{2t-1}(m_1) \end{bmatrix} \tag{11}$$

$$\mathbf{D}_{\text{odd}}(m_1) = \begin{bmatrix} \mathbf{d}_1(m_1) & \mathbf{d}_3(m_1) & \cdots & \mathbf{d}_{2t-1}(m_1) \end{bmatrix} \tag{12}$$

$$\mathbf{Q}_{\text{odd}} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_3 & \cdots & \mathbf{q}_{2t-1} \end{bmatrix}. \tag{13}$$

On the other hand, the matrix $\mathbf{X}_{\text{D}}(m_1)$ is obtained by diagonally appending each constant GF multiplication as follows.

$$\mathbf{X}_{\text{D}}(m_1) = \begin{bmatrix} \mathbf{A}_{1p}(m_1) & 0 & \cdots & 0 \\ 0 & \mathbf{A}_{3p}(m_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{A}_{(2t-1)p}(m_1) \end{bmatrix}. \tag{14}$$

Note that the grouped matrices for the second GF dimension, i.e., $\mathbf{C}_{\text{odd}}(m_2)$, $\mathbf{D}_{\text{odd}}(m_2)$, and $\mathbf{X}_{\text{D}}(m_2)$, are defined by using the same way. After taking n/p cycles, $\mathbf{D}_{\text{odd}}(m_1)$ and $\mathbf{D}_{\text{odd}}(m_2)$ becomes odd-indexed syndromes for two different field dimensions, i.e., GF(2^{m_1}) and GF(2^{m_2}), respectively.

Based on the odd-indexed syndromes, it is necessary to generate even-indexed syndromes by applying power operations. For the given i-th odd-indexed syndrome unit, the related even-indexed syndromes over GF(2^{m_1}) are obtained as follows.

$$\begin{aligned} &\begin{bmatrix} \mathbf{s}_i(m_1) & \mathbf{s}_{i\times 2}(m_1) & \mathbf{s}_{i\times 2^2}(m_1) & \cdots \end{bmatrix} \\ &= \mathbf{q}_i \times \begin{bmatrix} \mathbf{I}(m_1) & \mathbf{B}_2(m_1) & \mathbf{B}_{2^2}(m_1) & \cdots \end{bmatrix} \\ &= \mathbf{q}_i \times \mathbf{P}_i(m_1) \end{aligned} \tag{15}$$

where $\mathbf{B}_w(m_1)$ is an $m_1 \times m_1$ binary matrix stands and stands for the power operations $y = x^w$ over GF(2^{m_1}) [15]. Note that $\mathbf{I}(m_1)$ is an $m_1 \times m_1$ identity matrix so that $\mathbf{q}_i$ becomes $\mathbf{s}_i(m_1)$ directly as we discussed previously. By grouping t $\mathbf{P}_i(m_1)$ matrices, therefore, it is possible to generate all the $2t$ syndromes over GF(2^{m_1}) as follows.

$$\begin{aligned} \mathbf{S}(m_1) &= \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_3 & \cdots & \mathbf{q}_{2t-1} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{P}_1(m_1) & 0 & \cdots & 0 \\ 0 & \mathbf{P}_2(m_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{P}_{2t-1}(m_1) \end{bmatrix} \\ &= \mathbf{Q}_{\text{odd}} \times \mathbf{X}_{\text{S}}(m_1). \end{aligned} \tag{16}$$

Considering two equations (10) and (16), finally, we can develop a huge single matrix equation that can calculate all the syndromes over two different GF dimensions as follows.

$$\begin{aligned} &\begin{bmatrix} \mathbf{D}_{\text{odd}}(m_1) & \mathbf{D}_{\text{odd}}(m_2) & \mathbf{S}(m_1) & \mathbf{S}(m_2) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z} & \mathbf{Q}_{\text{odd}} \end{bmatrix} \times \begin{bmatrix} \mathbf{C}_{\text{odd}}(m_1) & \mathbf{C}_{\text{odd}}(m_2) & 0 & 0 \\ \mathbf{X}_{\text{D}}(m_1) & \mathbf{X}_{\text{D}}(m_2) & \mathbf{X}_{\text{S}}(m_1) & \mathbf{X}_{\text{S}}(m_2) \end{bmatrix}. \end{aligned} \tag{17}$$

Fig. 8 represents the proposed dual-m SC hardware architecture supporting (17). Note that we added multiplexors to select the target GF dimension after performing a huge matrix operation for different GF dimensions. If we perform the iterative CSE searching algorithm on this matrix [25], the search area of CSEs is theoretically maximized, leading to the minimum number of XOR gates for realizing the parallel dual-m SC block.

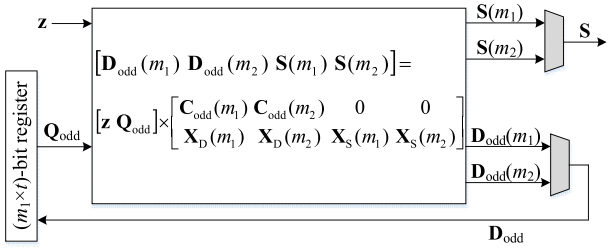

FIGURE 8. The proposed dual-m p-parallel SC block after constructing a single matrix operator.

B. THE PROPOSED PARALLEL DUAL-m CS ARCHITECTURE

Since the parallel CS block also uses numerous constant GF multipliers [11], the proposed optimization steps for the dual-m architecture can be summarized in the similar way to that of the proposed parallel SC block. In the parallel CS block, which is the last step of BCH decoding in Fig. 1, up to t error positions are determined by evaluating the error locator polynomial, which is generated by the KES block. More precisely, the t-order error locator polynomial over GF(2^{m_1}) is characterized by $t+1$ m_1-bit coefficients denoted as $1 \times m_1$ binary matrix forms, $\lambda_0, \lambda_1, \ldots, \lambda_t$, and α^i becomes the root of this polynomial if the error location is represented by i ($1 \le i \le n$).

Fig. 9 illustrates the conventional p-parallel CS architecture, taking n/p cycles by evaluating p consecutive positions at each processing cycle [16]. At the first processing cycle,

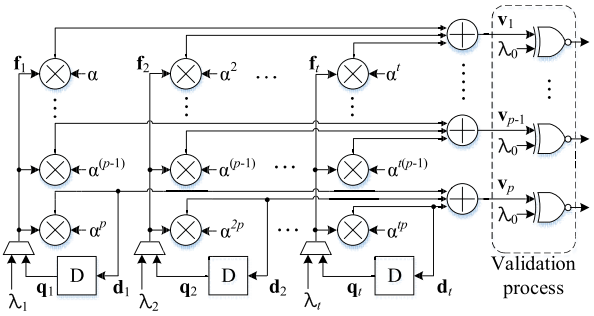

FIGURE 9. The conventional *p*-parallel CS unit [11].

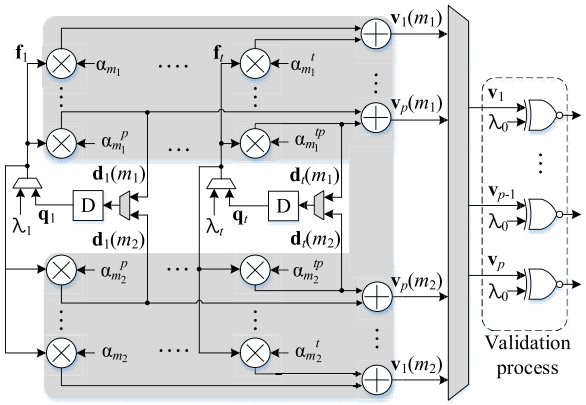

FIGURE 10. The proposed dual-*m* *p*-parallel CS unit. Processing parts in the shaded-region are grouped to generate a single matrix operator.

in this architecture, the received t coefficients are selected to $\mathbf{f}_j$ $(1 \le j \le t)$ for checking the first p consecutive positions, and the temporal values $\mathbf{d}_j$ are calculated to prepare the next evaluation at the same time. At the following cycles, these $\mathbf{d}_j$ values are updated to $\mathbf{q}_j$, which are selected to $\mathbf{f}_j$ for the rest of processing cycles. To reduce the complexity of p-parallel fixed-m CS block, all the constant GF multipliers and GF adders are grouped into a single matrix operation for sharing CSEs as many as possible [16].

Based on the previous optimizations, the proposed p-parallel dual-m CS architecture, which supports two GF dimensions m_1 and m_2 $(m_1 > m_2)$, is exemplified in Fig. 9. Based on the selected $\mathbf{f}_j$ inputs, in the proposed architecture, we calculate two different temporal values, $\mathbf{d}_j(m_1)$ and $\mathbf{d}_j(m_2)$, over different fields, GF(2^{m_1}) and GF(2^{m_2}), respectively. As shown in Fig. 10, the proposed CS architecture contains t m_1-bit temporal register, which means that the temporal values for the lower dimension, i.e., $\mathbf{d}_j(m_2)$, are zero extended. Similar to the dual-m SC case, the proposed work generates a single matrix operator by grouping all the GF operators inside of the shaded region in Fig. 10, leading to the maximum number of CSEs.

To derive the proposed optimization steps for p-parallel dual-m CS architecture, we first group t constant GF multipliers for calculating temporal values $\mathbf{d}_j(m_1)$ over GF(2^{m_1}) as follows.

$$\begin{aligned}\mathbf{D}(m_1) &= \begin{bmatrix}\mathbf{d}_1(m_1) & \mathbf{d}_2(m_1) & \cdots & \mathbf{d}_t(m_1)\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_t\end{bmatrix} \\ &\quad \times \begin{bmatrix}\mathbf{A}_{1p}(m_1) & 0 & \cdots & 0 \\ 0 & \mathbf{A}_{2p}(m_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{A}_{tp}(m_1)\end{bmatrix} \\ &= \mathbf{F} \times \mathbf{G}_{\mathrm{D}}(m_1). \end{aligned} \tag{18}$$

By considering the lower GF dimension, all the temporal values for dual-m CS block can be derived as follows.

$$\begin{bmatrix}\mathbf{D}(m_1) & \mathbf{D}(m_2)\end{bmatrix} = \mathbf{F} \times \begin{bmatrix}\mathbf{G}_{\mathrm{D}}(m_1) & \mathbf{G}_{\mathrm{D}}(m_2)\end{bmatrix}. \tag{19}$$

Note that $\mathbf{G}_{\mathrm{D}}(m_2)$ contains zero-extended multiplications over GF(2^{m_2}) as depicted in (2).

To find the error positions, as shown in Fig. 10, the evaluation results $\mathbf{v}_k(m_1)$ and $\mathbf{v}_k(m_2)$ are firstly calculated based on $\mathbf{f}_j$ values, where $1 \le k \le p$. Then, the proper $\mathbf{v}_{\mathrm{k}}$ values are selected by taking account of the current dimension, and the final validation process is performed by comparing p $\mathbf{v}_k$ values with the constant coefficient of error locator polynomial, i.e., λ_0. For the case of m_1 dimension, the calculation process of $\mathbf{v}_k(m_1)$ can be expressed as follows.

$$\begin{aligned}\mathbf{v}_k(m_1) &= \sum_{j=1}^{t} \mathbf{f}_j \mathbf{A}_{ij}(m_1) \\ &= \begin{bmatrix}\mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_t\end{bmatrix} \times \begin{bmatrix}\mathbf{A}_{i1}(m_1) \\ \mathbf{A}_{i2}(m_1) \\ \vdots \\ \mathbf{A}_{it}(m_1)\end{bmatrix} \end{aligned} \tag{20}$$

By grouping p $\mathbf{v}_{\mathrm{k}}$ values, denoted as $\mathbf{V}(m_1)$, we can construct the a single matrix operator, denoted as $\mathbf{G}_{\mathrm{V}}(m_1)$, which includes all the GF operations for p-parallel evaluating processes over GF(2^{m_1}) as follows.

$$\begin{aligned}\mathbf{V}(m_1) &= \begin{bmatrix}\mathbf{v}_1(m_1) & \mathbf{v}_2(m_1) & \cdots & \mathbf{v}_p(m_1)\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_t\end{bmatrix} \\ &\quad \times \begin{bmatrix}\mathbf{A}_{1\times1}(m_1) & \mathbf{A}_{2\times1}(m_1) & \cdots & \mathbf{A}_{p1}(m_1) \\ \mathbf{A}_{1\times2}(m_1) & \mathbf{A}_{2\times2}(m_1) & \cdots & \mathbf{A}_{p2}(m_1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{1t}(m_1) & \mathbf{A}_{2t}(m_1) & \cdots & \mathbf{A}_{pt}(m_1)\end{bmatrix} \\ &= \mathbf{F} \times \mathbf{G}_{\mathrm{V}}(m_1). \end{aligned} \tag{21}$$

It is obvious that the similar way can be applied for the case of lower dimension. The only difference is that we have to consider the zero-extended multiplications as depicted in (2). As a result, all the evaluation parts supporting two different field dimensions can be formulated into a single matrix as follows.

$$\begin{bmatrix}\mathbf{V}(m_1) & \mathbf{V}(m_2)\end{bmatrix} = \mathbf{F} \times \begin{bmatrix}\mathbf{G}_{\mathrm{V}}(m_1) & \mathbf{G}_{\mathrm{V}}(m_2)\end{bmatrix}. \tag{22}$$

By combing two parts, i.e., calculating the temporal values in (19) and evaluating the error positions in (22), we can

finally obtain the single and huge matrix operator for p-parallel dual-m CS architecture as follows.

$$\begin{bmatrix} \mathbf{V}(m_1) & \mathbf{V}(m_2) & \mathbf{D}(m_1) & \mathbf{D}(m_2) \end{bmatrix} = \mathbf{F} \times \begin{bmatrix} \mathbf{G}_{\mathrm{V}}(m_1) & \mathbf{G}_{\mathrm{V}}(m_2) & \mathbf{G}_{\mathrm{D}}(m_1) & \mathbf{G}_{\mathrm{D}}(m_2) \end{bmatrix}. \quad (23)$$

Fig. 11 illustrates the proposed hardware architecture based on (23). Similar to the proposed dual-m SC case, the proposed dual-m CS architecture enlarges the search area of CSEs as much as possible, leading to the area-efficient solution as described at the following subsection.

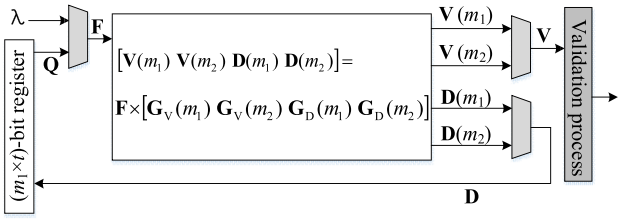

FIGURE 11. The proposed dual-m p-parallel CS unit with the singe-matrix operator.

C. IMPLEMENTATION RESULTS OF DUAL-m SC AND CS BLOCKS

To verify the proposed SC and CS blocks in dual-m BCH decoder, this subsection shows the implementation results of various parameters. We compared the results of the proposed multi-m BCH decoders by using Synopsys' Design Compiler in 65nm CMOS technology. Considering the broadcasting standards, we select two BCH codes over GF(2^{16}) and GF(2^{14}), where the number of correctable errors is equally set to 12. For fair comparisons, the area efficiency is used for this work, which is defined as follows [27]:

$$\text{Area efficiency (Gbps/mm}^2) = \frac{\text{Decoding throughput (Gbps)}}{\text{Area (mm}^2)}. \quad (24)$$

Fig. 12 shows the comparison between the proposed dual-m SC architecture and the previous register-sharing dual-m SC block [21]. It can be seen that the proposed structure is more effective in both area and area efficiency for most parallel factors as the proposed work shares both sequential and combinational logics. By increasing the parallel factor, the impact of the proposed optimization method is remarkably enhanced. In the case of 8-parallel architectures, for example, the proposed dual-m SC block improves the area-efficiency by 1.74 times.

The area-efficiencies of parallel dual-m CS architectures are illustrated in Fig. 13. In contrast to the impact of our dual-m SC architecture, as depicted in the figure, the proposed dual-m CS block slightly degrades the area efficiency in the case of serial, i.e., when the parallel factor is set to 1. This is because of the increased critical delay from aggressive-sharing of CSEs [30]. If the parallel is increased, however, the effects of area saving become dominant, reducing the area

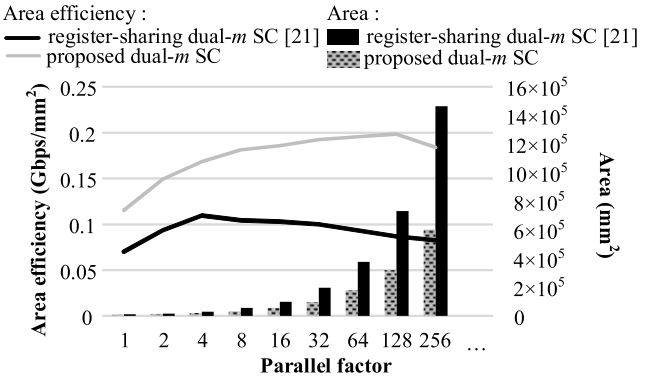

FIGURE 12. Area and area efficiencies of dual-m SC blocks depending on the parallel factor. ($m_1 = 16$, $m_2 = 14$, and $t = 12$).

efficiency remarkably. According to the number of parallel factor, moreover, the amount of effectiveness is gradually increased as depicted in Fig. 13. For example, when the parallel factor is 8, the proposed dual-m CS architecture saves the area-efficiency by 1.61 times by reducing the hardware complexity by 33% compared to the previous register-sharing architecture.

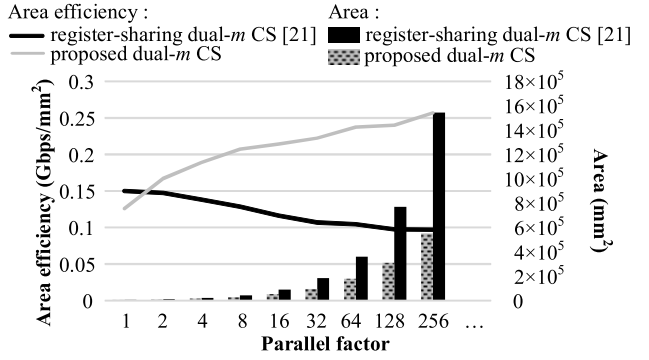

FIGURE 13. Area and Area efficiencies of dual-m CS blocks depending on the parallel factor. ($m_1 = 16$, $m_2 = 14$, and $t = 12$).

Based on the proposed optimization method, it is noticeable that the area efficiencies of parallel dual-m SC and CS architectures are superior to the previous works [21]. As the contemporary flexible BCH decoders, which will be discussed in Section V, aim more than 8-parallel architectures for increasing the decoding throughput, the proposed SC and CS architectures can be happily acceptable at the practical area-efficient designs.

IV. THE PROPOSED FOLDED-KES STRUCTURE FOR MULTIPLE GF DIMENSIONS

This section discusses the proposed optimization methods, which enable the area-efficient dual-m KES module. As the KES operation is normally based on the general GF multiplication rather than the constant ones, it is hard to directly apply the previous matrix-based sharing techniques. In order to share the combinational parts of general multiplier, therefore, we propose a new folding method for dual-m KES architecture.

A. THE PREVIOUS FOLDED-KES ARCHITECTURES

Fig. 14 illustrates the conventional KES architecture based on the simplified inverse-free Berlekamp-Massey (SiBM) algorithm, which takes t cycles for calculating the error location polynomial [23]. Due to the numerous general GF multipliers, in general, the unfolded-KES operation is considered as the most complex and area-consuming operation among three BCH decoding stages [11]. On the other hand, the processing latency of KES block is relatively shorter than the other stages. Hence, the folding technique is widely applied to reduce the hardware complexity, making the balanced pipelining process. For example, the original SiBM architecture in Fig. 14 can be folded by sharing the grey-colored processing elements (PEs), allowing more processing cycles [23]. Compared to this PE-level global folding, the recent architecture in [27] reveals that the multiplier-level local folding provides better area-efficiency by even reducing the internal critical delay as well as the hardware complexity. However, these techniques are only targeting the fixed-m KES architecture, no longer suitable for developing the area-efficient fully-flexible BCH decoder.

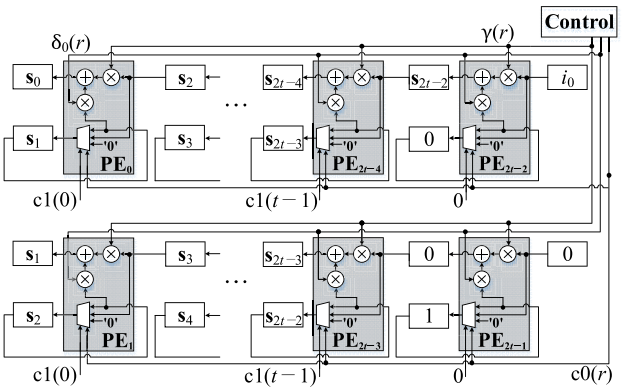

FIGURE 14. The previous unfolded-SiBM architecture [23].

Similar to the previous dual-m solution, the register sharing method from [21] and [22] has limitations by nature and it is necessary to share internal combinational logics for reducing the overall hardware complexity of KES block. To support dual-m operation in a single general multiplier without using separate multipliers, as reported in [21], the modulation parts for two dimensions can be designed individually and attached at the end of shared processing parts. However, this method is inefficient in terms of critical delay by utilizing the serially-connected logics. As the operating frequency of a whole BCH decoder is normally determined by the KES module [29], the decoding throughput can be degraded due to the increased clock period, even reducing the area efficiency of decoder. Based on the locally-folded multiplier architecture in [27], in this work, we present efficient architectures of processing cells (PCs) for supporting different field dimensions while enhancing the area efficiency remarkably.

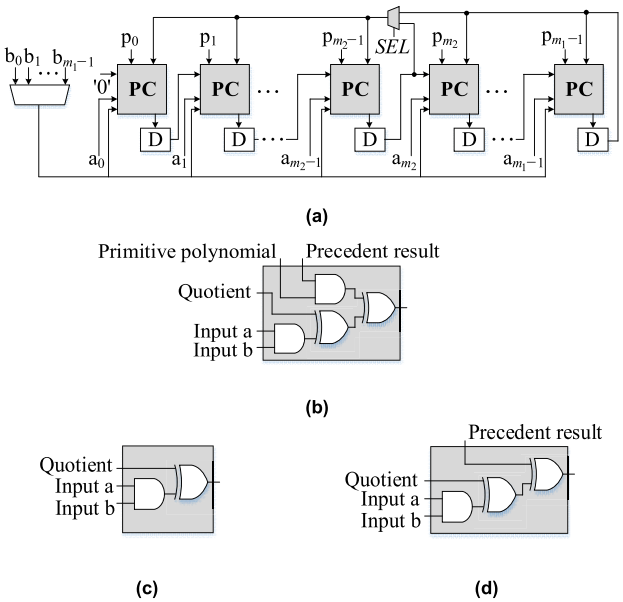

FIGURE 15. The proposed (a) dual-m general GF multiplier, (b) non-biased process cell, (c) 0-biased process cell, and (d) 1-biased process cells.

B. THE PROPOSED DUAL-m FOLDED-KES ARCHITECTURE

As reported in [27], the PC-based locally-folded general GF multiplier is effective in reducing both the critical delay and the hardware complexity of fixed-m SiBM-based KES architecture. In this subsection, we add the flexibility to the previous multiplier by presenting new PC types and simple control schemes.

Targeting two GF dimensions, GF(2^{m_1}) and GF(2^{m_2}), Fig. 15(a) conceptually shows the proposed dual-m locally-folded general GF multiplier. Applying the maximum number of folding factor, as shown in the figure, the proposed multiplier utilizes m_1 PCs ($m_1 > m_2$). According to the coefficients of primitive polynomials for constructing GF fields, the proposed multiplier uses three types of PCs, i.e., non-biased PC (PC_N), 0-biased PC (PC_0), and 1-biased PC (PC_1), as depicted in Fig. 15(b), (c), and (d), respectively. For m_2 left-most bit positions, PCs are shared by two general multipliers over GF(2^{m_1}) and GF(2^{m_2}). More precisely, if two coefficients of the same bit position of primitive polynomials are different from each other, then the PC_N is reserved at the corresponding position, allowing the change of primitive polynomial as depicted in Fig. 15(b). If two coefficients from different primitive polynomials are identical at a certain bit position, on the other hand, PC_0 (or PC_1) is used by reflecting the common value. Note that the rest of PCs, which are used for (m_1–m_2) right-most positions, are only considering the coefficient values of the primitive polynomial for GF(2^{m_1}). Hence, only PC_0 and PC_1 are used for these positions.

Fig. 16 exemplifies the proposed flexible general GF multiplier supporting both GF(2^5) and GF(2^4) whose primitive polynomials are given as $1 + x^2 + x^5$ and $1 + x + x^4$,

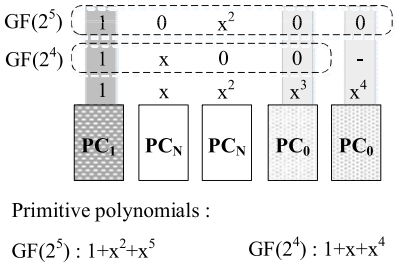

FIGURE 16. The decision example of PC types for supporting multiple primitive polynomials.

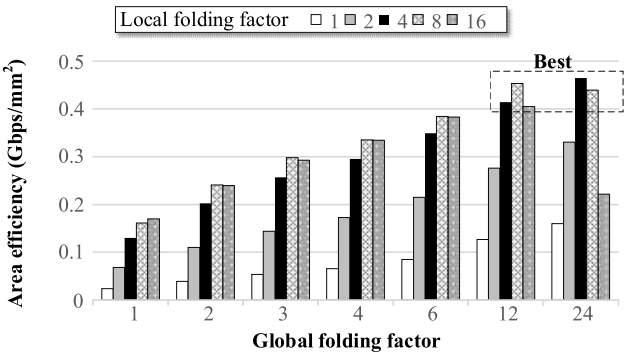

FIGURE 17. Area efficiencies of dual-m KES blocks having different folding factors. ($m_1 = 16$, $m_2 = 14$, and $t = 12$).

respectively. Note that three non-biased PCs are utilized at the first, second, and fourth bit positions as two coefficients at these positions are different from each other. For the same coefficient values, PC_1 and PC_0 are used for the zeroth and third positions, respectively, by taking account of each value. Note that the last fifth bit position, where PC_1 is introduced, is dedicated only for GF(2^5), the larger dimension.

In the proposed folded process, for the case of larger field dimension, m_1 cycles are used for multiplying two m_1-bit input operands over GF(2^{m_1}). More precisely, each bit of (b_0, $b_1, \ldots, b_{m-1}$) is serially inserted to the multiplier in order, whereas all the bits of ($a_0, a_1, \ldots, a_{m-1}$) are issued in parallel as depicted in Fig. 15(a). When the lower field dimension is selected, two operands are zero-extended, and only m_2 cycles are consumed for the general multiplication process. Note that (m_1–m_2) right-most PCs are disabled by deactivating the control signal *SEL* so that the upper bits are automatically disconnected to the current processing steps for GF(2^{m_2}).

By replacing the general GF multipliers of the previous SiBM architecture in Fig. 14 to the proposed folded multi-m multiplier in Fig. 15(a), the KES block can successfully support two different field dimensions without using more multiplier units. In addition, the locally-folded architecture may be combined to the globally-folding concept, i.e., the previous PE-level folding, as discussed in [27]. This hybrid-folding technique further enhances the area-efficiency of dual-m folded-KES block, as discusses at the following sub-section.

C. IMPLEMENTATION RESULTS OF DUAL-m KES ARCHITECTURES

Similar to the dual-m parallel SC and CS modules, as described in the previous Section, we design dual-m KES architectures based on the straightforward architecture, the previous register-sharing and partial GF logic sharing structure [21], and the proposed solution. Supporting two types of BCH codes at the same time, (65535, 65343, 12) and (16383, 16216, 12) codes, all the architectures are equally designed in 65nm CMOS process for providing fair comparison results. In addition, we assume 8-parallel SC/CS stages for finding the best folding parameters, which can optimize each KES structure in terms of area efficiency. For example, as depicted in Fig. 17, the proposed dual-m KES module achieves the maximum area-efficiency when it adopts the hybrid folding method with the local folding factor of 8 and the global folding factor of 12.

TABLE 1. Comparison of KES for dual GF dimensions GF(2^{14}) and GF(2^{16}).

	Straightforward (two KESs)	Previous sharing [21]	Proposed
Area (μm^2)	31369.6	17523.5	12505.6
Critical path(ns)	1.64	1.73	1.01
Area efficiency (Gbps/mm^2)	0.156 (1.0)	0.264 (1.7)	0.453 (2.9)

Resulting from the commercialized EDA tool, Table 1 compares three different dual-m KES architectures. Note that the proposed architecture occupies the minimum silicon area, while even having the shortest critical delay. This result is reasonable, as the proposed work internally cuts the delay paths inside of dual-m GF multipliers where the previous architectures separately utilize general multipliers for different dimensions. As a result, the proposed dual-m KES module improves the area efficiency by more than 72% compared to the register-sharing architecture, providing the most attractive solution.

V. CASE STUDIES

This section shows case studies of fully-flexible BCH decoder prototypes by targeting the broadcasting specification and storage controllers. To verify the effectiveness of the proposed studies, we implement three types of flexible decoder architectures; the straight-forward architecture having multiple dedicated BCH decoders, the previous register-sharing and partial GF logic sharing architecture [21], and the flexible decoder based on the presented optimizations. Note that all the decoder architectures consume the same number of processing cycles as no additional cycles are required for each optimization scheme. For fair comparisons, all the prototype designs are equally synthesized in a 65nm CMOS process using Synopsys' Design Compiler. In addition, the constant GF multipliers over different GF dimensions in [21] are reformulated into single binary matrix operators, which can guarantee the maximum number of common terms detected by the commercialized EDA tools.

TABLE 2. Comparison of dual-m BCH decoder for DVB-S2.

	Straightforward (decoder×2)	Previous sharing [21]	Proposed
Area(μm^2)	135787	116314	72742
Critical path(ns)	1.64	1.73	1.41
Throughput(Gbps)	4.88	4.62	5.67
Area efficiency (Gbps/mm^2)	3.59×10^{-2} (1.0)	3.97×10^{-2} (1.11)	7.79×10^{-2} (2.17)

TABLE 3. Comparison of multi-m BCH decoder for NAND flash memories.

	Straightforward (decoder×3)	Previous sharing [21]	Proposed
Area(μm^2)	5579349.9	4778828.9	2125066.3
Critical path(ns)	2.03	2.13	1.94
Throughput(Gbps)	15.76	15.02	16.49
Area efficiency (Gbps/mm^2)	2.82×10^{-3} (1.0)	3.14×10^{-3} (1.11)	7.76×10^{-3} (2.75)

A. CASE STUDY ON THE DVB-S2 SYSTEM

In the first case, the proposed fully-flexible BCH decoder is designed to support the DVB-S2 system [7]. In this application, two BCH codes over GF(2^{16}) and GF(2^{14}) dimensions are defined for managing the normal and short frames, respectively. The system defines 58320b-length message frame for a normal frame, whereas a short one consists of 14400 bits. It also requests various t parameters including 8, 10, and 12. Based on the standard implementation guide [31], the parallel factor of the proposed fully-flexible decoder is set to 8 to meet the required throughput sufficiently. Under the specified conditions, therefore, the folding factors of KES module is properly set to maintain the overall decoding throughput. Table 2 compares different BCH decoders that can fully support this DVB-S2 specification. Note that the proposed dual-m decoder takes the lowest hardware complexity while achieving the minimum level of area efficiency. In terms of the area efficiency, more precisely, the proposed solution enhances provides 2.17 and 1.96 times better results compared to the straight-forward realization and the recent register-sharing and partial GF logic sharing architecture, respectively.

B. CASE STUDY ON THE NAND FLASH MEMORY CONTROLLER

Based on the proposed optimizations for supporting multiple GF dimensions, we also design three different types of flexible BCH decoders targeting three field dimensions, which are used for the recent NAND flash memory controllers [6], [32]. In this prototype decoders, we support 512B, 1KB and 2KB user data, and up to 120bit random errors can be corrected. As shown in Table 3, the proposed optimization still provides the most attractive results compared to the other solutions. Note that the area-efficiency of the proposed fully-flexible BCH decoder is 2.47 times better than that of the previous sharing architecture. Compared to the dual-m cases depicted in Table 2, it is important that the improvement due to the proposed methods is increased by 58% by supporting one more dimension. This means that the proposed fully-flexible architecture is superior to the previous options when the practical system considers multiple types of BCH codes.

VI. CONCLUSION

In this paper, we have presented several optimization skills for realizing area-efficient fully-flexible BCH decoders, which can support different field dimensions. By sharing the internal computing parts as much as possible, the proposed methods actually reduce the hardware complexity to the minimum level. The new folded-multiplier architecture also allows the multi-dimensional operations without using redundant hardware resources. Various case studies show that the proposed optimizations are quite effective to enhance the area efficiency while providing the flexible ECC usages, compared to the state-of-the-art designs.

REFERENCES

[1] C.-H. Yang, T.-Y. Huang, M.-R. Li, and Y.-L. Ueng, "A 5.4 μ W soft-decision BCH decoder for wireless body area networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 9, pp. 2721–2729, Sep. 2014.

[2] S. Khalid and S. Khan, "Application of compressed sensing on images via BCH measurement matrices," in *Proc. Int. Conf. Robot. Emerg. Allied Technol. Eng. (iCREATE)*, 2014, pp. 78–81.

[3] *IEEE Standard for Ethernet-Amendment 3: Physical Layer Specifications and Management Parameters for 40 Gb/s and 100 Gb/s Operation over Fiber Optic Cables*, IEEE Standard 802.3bm, 2015.

[4] A. Kumar and A. Kumar, "A cell-array-based multibiometric cryptosystem," *IEEE Access*, vol. 4, pp. 15–25, 2016.

[5] *Telecommunication Standardization Sector of ITU; Series G: Transmission Systems and Media, Digital Systems and Networks, Digital Sections and Digital Line System—Optical Fibre, Submarine Cable Systems, Forward Error Correction for High Bit-Rate DWDM Submarine Systems*, document Rec. G.975.12004.02, ITU-T, 2004.

[6] ONFI. (Feb. 2014). *Open NAND Flash Interface Specification, Revision 4.0*. [Online]. Available: http://www.onfi.org

[7] *Digital Video Broadcasting (DVB); Second Generation Framing Structure, Channel Coding and Modulation Systems for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications; Part 1: DVB-S2*, document EN 302 307-1 V1.4.1, ETSI, 2014.

[8] *Digital Video Broadcasting (DVB); Frame Structure Channel Coding and Modulation for a Second Generation Digital Terrstrial Television Broadcasting System (DVB-T2)*, document EN 302 307-1 V1.4.1, ETSI, 2015.

[9] *Digital Video Broadcasting (DVB); Frame Structure Channel Coding and Modulation for a Second Generation Digital Transmission System for Cable Systems (DVB-C2)*, document EN 302 769 V1.3.1, 2015.

[10] S. Hwang, J. Jung, D. Kim, J. Ha, I.-C. Park, and Y. Lee, "An energy-optimized (37840, 34320) symmetric BC-BCH decoder for healthy mobile storages," in *Proc. IEEE Asian Solid State Circuits Conf. (A-SSCC)*, Nov. 2017, pp. 169–172.

[11] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[12] Y. Chen and K. K. Parhi, "Small area parallel Chien search architectures for long BCH codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 545–549, May 2004.

[13] Y. Lee, H. Yoo, I. Yoo, and I.-C. Park, "High-throughput and low-complexity BCH decoding architecture for solid-state drives," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 1183–1187, May 2014.

[14] K. Lee, H.-G. Kang, J.-I. Park, and H. Lee, "100GB/S two-iteration concatenated BCH decoder architecture for optical communications," in *Proc. IEEE Workshop On Signal Process. Syst.*, Oct. 2010, pp. 404–409.

[15] Y. Lee, H. Yoo, and I.-C. Park, "Small-area parallel syndrome calculation for strong BCH decoding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1609–1612.

[16] Y. Lee, H. Yoo, and I.-C. Park, "Low-complexity parallel Chien search structure using two-dimensional optimization," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 8, pp. 522–526, Aug. 2011.
[17] C.-C. Chu, Y.-M. Lin, C.-H. Yang, and H.-C. Chang, "A fully parallel BCH codec with double error correcting capability for NOR flash applications," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1605–1608.
[18] D. Shin, J. Park, J. Park, S. Paul, and S. Bhunia, "Adaptive ECC for tailored protection of nanoscale memory," *IEEE Des. Test.*, vol. 34, no. 6, pp. 84–93, Dec. 2017.
[19] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S.-L. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," in *Proc. ACM/IEEE 38th Int. Symp. Comput. Archit. (ISCA)*, Jun. 2011, pp. 461–471.
[20] C.-H. Yang, Y.-M. Lin, H.-C. Chang, and C.-Y. Lee, "An MPCN-based BCH codec architecture with arbitrary error correcting capability," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1235–1244, Jul. 2015.
[21] B. Zhang, D. Liu, S. Wang, X. Chen, and H. Liu, "Design and implementation of area-efficient DVB-S2 BCH decoder," in *Proc. 2nd Int. Conf. Comput. Eng. Technol. (ICCET)*, vol. 3. 2010, pp. 179–184.
[22] Y.-M. Lin, C.-L. Chen, H.-C. Chang, and C.-Y. Lee, "A 26.9 K 314.5 Mb/S soft (32400,32208) BCH decoder chip for DVB-S2 system," *IEEE J. Solid-State Circuits*, vol. 45, no. 11, pp. 2330–2340. Nov. 2010.
[23] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *Proc. IEEE Workshop Signal Process. Syst. Design Implement.*, Oct. 2006, pp. 303–308.
[24] Y. M. Lin, H. T. Li, M. H. Chung, and A. Y. Wu, "Byte-reconfigurable LDPC codec design with application to high-performance ECC of NAND flash memory systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 7, pp. 1794–1804, Jul. 2015.
[25] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 15, no. 2, pp. 151–165, Feb. 1996.
[26] M. Yin, M. Xie, and B. Yi, "Optimized algorithms for binary BCH codes," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2013, pp. 1552–1555.
[27] B. Park, S. An, J. Park, and Y. Lee, "Novel folded-KES architecture for high-speed and area-efficient BCH decoders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 5, pp. 535–539, May 2017.
[28] R.-J. Chen, J.-W. Fan, and C.-H. Liao, "Reconfigurable galois field multiplier," in *Proc. Int. Symp. Biomet. Security Technol. (ISBAST)*, 2014, pp. 112–115.
[29] H. Y. Tsai, C. H. Yang, and H. C. Chang, "An efficient BCH decoder with 124-bit correctability for multi-channel SSD applications," in *Proc. IEEE Asian Solid State Circuits Conf. (A-SSCC)*, Nov. 2012, pp. 61–64.
[30] X. Zhang, N. Wu, L. Lan, and Y. Liu, "A low-delay common subexpression elimination algorithm for constant matrix multiplications over GF(2^m)," in *Proc. IEEE 10th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2015, pp. 416–421.
[31] *Digital Video Broadcasting (DVB) Implementation Guidelines for the Second Generation System for Broadcasting, Interactive Services, News Gathering and Other Broadband Satellite Applications; Part I (DVB-S2)*, document A171-1, DVB, 2015.
[32] D. Wei, L. Deng, L. Qiao, P. Zhang, and X. Peng, "PEVA: A page endurance variance aware strategy for the lifetime extension of NAND flash," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 5, pp. 1749–1760, May 2016.

BYEONGGIL PARK (S'15) received the B.S. degree in electrical information system from Hanyang University, Ansan, South Korea, in 2011, and the M.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2017.

Since 2011, he has been with ASIC&IP Development Team, Samsung electronics Inc. His research interests include error correction coding design and energy-efficient VLSI design.

JONGSUN PARK (M'05–SM'13) received the B.S. degree in electronics engineering from Korea University, Seoul, South Korea, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2000 and 2005, respectively. He joined the Electrical Engineering Faculty, Korea University, Seoul, in 2008.

He was with the Digital Radio Processor System Design Group, Texas Instruments, Dallas, TX, USA, in 2002. From 2005 to 2008, he was with the Signal Processing Technology Group, Marvell Semiconductor Inc., Santa Clara, CA, USA. His research interests focus on variation-tolerant, low-power, high-performance VLSI architectures and circuit designs for digital signal processing, and digital communications.

Dr. Park is a member of the Circuits and Systems for Communications Technical Committee of the IEEE Circuits and Systems Society. He served as a Guest Editor for the IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS. He has also served in the technical program committees for various IEEE/ACM conferences, including ISCAS, ASP-DAC, HOST, VLSI-SoC, ISOCC, and APCCAS. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS.

YOUNGJOO LEE (M'14) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2008, 2010, and 2014, respectively. Since 2017, he has been an Assistant Professor with the Department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, South Korea.

Prior to joining POSTECH, he was with Interuniversity Microelectronic Center, Leuven, Belgium, from 2014 to 2015, where he researched reconfigurable SoC platforms for software-defined radio systems. From 2015 to 2016, he was with the Faculty of the Department of Electronic Engineering, Kwangwoon University, Seoul, South Korea. His current research interests include the algorithms and architectures for embedded processors, intelligent mobile systems, advanced error-correction codes, and mixed-signal integrated circuits.