JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.18, NO.5, OCTOBER, 2018
https://doi.org/10.5573/JSTS.2018.18.5.609
ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

Area-optimized Syndrome Calculation for ReedSolomon Decoder

Youngjoo Lee[1], In-Cheol Park[2], and Hoyoung Yoo[3,*]

***Abstract*—In this paper an efficient structure to compute syndromes is proposed for Reed-Solomon decoders. The proposed method formulates all the computations relevant to syndromes as a matrix multiplication so as to enlarge the search area for common sub-expressions. In contrast to the conventional architecture, the proposed method completely removes finite-field adders as they are embedded in the single matrix multiplication. The hardware resources are drastically reduced by sharing common sub-expressions and eliminating finite-field adders. Syndrome calculation blocks for various RS codes are synthesized with a 0.13μm CMOS technology, and experimental results show that the proposed method reduces the hardware complexity of parallel syndrome calculation by approximately 50% compared to the conventional structure.**

***Index Terms*—ReedSolomon codes, error correction codes, syndrome calculation, area optimization**

I. INTRODUCTION

Error-correction codes are widely used in communication systems to recover corrupted codewords from the noisy channel. Among a variety of error correction codes, the Reed-Solomon (RS) code forms the core of the most powerful algebraic codes due to its superior error correction performance in both burst and random errors [1, 2]. The RS code that corrects t error-symbols has been commonly employed for error control in diverse systems such as digital broadcasting, optical communication, and digital storage [2, 3]. Most of those applications have continuously demanded ever higher decoding throughput as well as ever larger error-correction capability, and these requirements necessitate high speed-RS decoders that can be realized by adopting architectural innovations such as parallel processing [4, 5] and pipelining [6-8]. Massive-parallel RS decoding is recently inevitable, but the hardware overhead resulting from the high parallel factor is a serious burden in implementation.

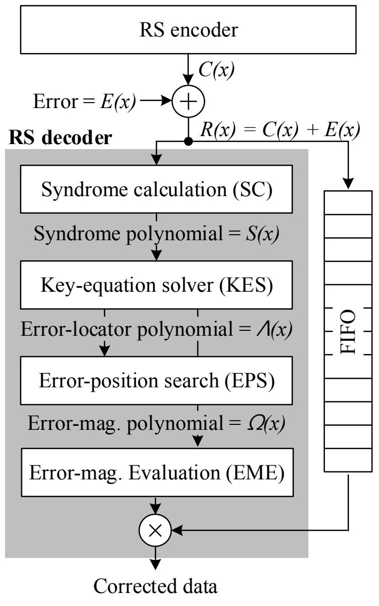
RS encoder
C(x)
Error = E(x)
R(x) = C(x) + E(x)
RS decoder
Syndrome calculation (SC)
Syndrome polynomial = S(x)
Key-equation solver (KES)
Error-locator polynomial = Λ(x)
Error-position search (EPS)
Error-mag. polynomial = Ω(x)
Error-mag. Evaluation (EME)
FIFO
Corrected data

Fig. 1. The typical RS decoder with 4 pipeline stages.

Manuscript received Apr. 9, 2018; accepted Aug. 12, 2018
[1] Department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang 37673, Korea
[2] Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34134, Korea
[3] Department of Electronics Engineering, Chungnam National University, Deajeon 34134, Korea
E-mail : hyyoo@cnu.ac.kr

Table 1. Area analysis on a RS (255, 239, 8) decoder

Components	Gate counts
SC	100,800 (23.2%)
KES	156,000 (35.8%)
EPS + EME	178,000 (40.9%)
Total	434,800 (100 %)

A RS codeword is usually decoded by conducting four steps as shown in Fig. 1: syndrome calculation (SC), key-equation solving (KES), Error-position search (EPS), and Error-magnitude evaluation (EME). Especially, Chein and Forney algorithms are widely used for EPS and EME, respectively. Many optimization methods such as folding [7, 8] and common substructure sharing [9-13] have been presented to relax the complexities of Chien search and Forney processing. Though the SC block has been regarded as being associated with a relatively less complexity, those optimizations make it have almost the same complexity as the other blocks. In [4], the massive-parallel SC block is reported to take almost a quarter of the whole decoder area as shown in Table 1.

This paper proposes an efficient structure to curtail the hardware complexity of parallel syndrome calculation. The proposed method transforms all the computations of parallel SC into a matrix multiplication to enlarge the search space for common sub-expressions (CSEs). As a result, the proposed method maximizes the sharing of CSEs and drastically reduces the hardware complexity of parallel SC.

II. Syndrome Calculation

For RS (n, k, t) code, every codeword consists of k message symbols and $2t$ parity symbols, and thus the codeword length n is $k + 2t$. Given a message, an n-symbol codeword is constructed based on a generator polynomial $g(x)$ that has α^i ($1 \le i \le 2t$) as its roots. An n-th order polynomial $C(x)$ is a valid codeword only when $C(x)$ is a multiple of $g(x)$.

In the syndrome based RS decoding, the received vector (r_0, r_1, $\cdots$, r_{n-1}) is interpreted as a polynomial, $R(x) = r_0 + r_1x^1 + \cdots + r_{n-1}x^{n-1}$, where r_i is an element of GF(2^m), and m is the smallest positive integer that satisfies $2^m - 1 \ge n$. We first check if there are any errors in the received vector by computing $2t$ syndromes as follows;

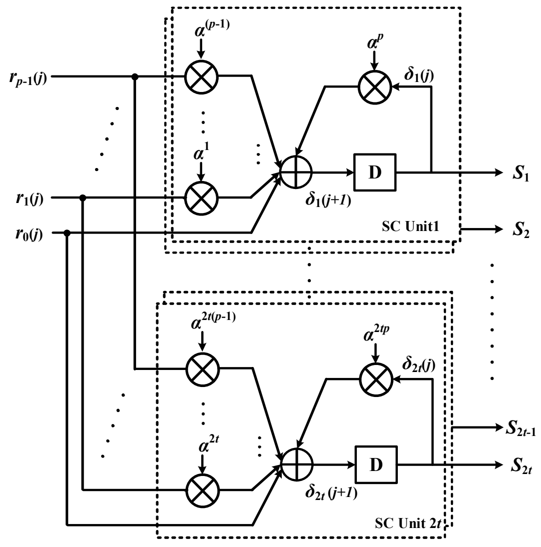

Fig. 2. The conventional SC block that processes p symbols in parallel.

$$S_i = R(\alpha^i) = \sum_{j=0}^{n-1} r_j \alpha^{ij} = r_0 + r_1\alpha^i + \cdots + r_{n-1}\alpha^{i(n-1)}, \ (1 \le i \le 2t) \tag{1}$$

If there is any non-zero syndrome, the received vector is regarded as being corrupted. Notice that the $2t$ syndromes are dependent solely on error patterns and independent of the transmitted codeword, since all the valid codewords are dividable by the generator polynomial $g(x)$.

As a symbol-serial SC unit processes a received symbol in a cycle, it takes n cycles. To reduce the decoding latency, more-than-one symbols should be simultaneously processed in cycle. If the number of received symbols processed in a cycle, so-called the parallel factor, is p, the processing latency can be improved by a factor of p as expressed in (2),

$$S_i = \sum_{k=0}^{n/p-1} \sum_{l=0}^{p-1} r_{l+pk} \alpha^{i(l+pk)} = \sum_{k=0}^{n/p-1} \alpha^{ipk} \sum_{l=0}^{p-1} r_{l+pk} \alpha^{il}, \ (1 \le i \le 2t) \tag{2}$$

where p is assumed to be a divisor of n.

In general, $2t$ syndromes are computed in parallel to achieve a high-throughput decoder. A conventional p-parallel SC block consisting of $2t$ SC units is depicted in Fig. 2. Each SC unit enclosed by a dashed box is responsible for one syndrome. In the symbol-serial SC block in which p is 1, one SC unit consists of a finite-field multipliers (FFMs), a two-input finite-field adder (FFA), and an m-bit register. However, a p-parallel SC unit takes p FFMs, a (p+1)-input FFA, and an m-bit register. Since the FFM is the most hardware consuming operation in the SC unit, the overall hardware complexity of the p-parallel SC unit is increased by almost p times compared to the symbol-serial SC unit. Hence, the hardware complexity is increased significantly if multiple symbols are processed in parallel to increase the decoding throughput.

III. Proposed Area-optimized SC

The main concept of the proposed method is to share common sub-expressions (CSEs) appearing more than once to reduce the hardware complexity. Briefly speaking, a CSE is evaluated only once, and then the result is used to replace such sub-expressions. For the syndrome calculation of RS codes, this paper applies an efficient optimization technique called the iterative matching algorithm [9] for the first time, and proposes a new method that enlarges the search area to find more common sub-expressions by considering all the calculations relevant to syndromes as a whole.

1. Iterative Matching Algorithm

A constant FFM in Fig. 2 that multiplies a received symbol r_j by a constant α^{ij} over GF(2^m), where $1 \le i \le 2t$, $0 \le j \le p$, can be transformed to a binary matrix multiplication as follows;

$$\begin{aligned} r_j\alpha^{ij} &= (r_{j,0} + r_{j,1}\alpha + \cdots + r_{j,m-1}\alpha^{m-1})\alpha^{ij} \\ &= r_{j,0}\alpha^{ij} + r_{j,1}\alpha^{ij+1} + \cdots + r_{j,m-1}\alpha^{ij+m-1} \\ &= \begin{bmatrix} r_{j,0} \\ r_{j,1} \\ \vdots \\ r_{j,m-1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \alpha_0^{ij} & \alpha_1^{ij} & \cdots & \alpha_{m-1}^{ij} \\ \alpha_0^{ij+1} & \alpha_1^{ij+1} & \cdots & \alpha_{m-1}^{ij+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{ij+m-1} & \alpha_1^{ij+m-1} & \cdots & \alpha_{m-1}^{ij+m-1} \end{bmatrix} \\ &= \mathrm{R}_j \mathrm{A}_{ij} \end{aligned} \quad (3)$$

where $\mathbf{R}_j$ is a binary $1\times m$ matrix and $\mathbf{A}_{ij}$ is a binary constant $m\times m$ matrix. Note that each element in GF(2^m) is represented with m bits. As the modulo-2 addition and modulo-2 multiplication in (3) are implemented with XOR gates, the complexity of a binary matrix multiplication is usually represented by the number of XOR gates required to implement it. A constant matrix multiplication can be optimized by finding and sharing CSEs, and such methods have been developed for filter synthesis [9] and Chien search [10, 11]. This bit-level optimization, which is called iterative matching algorithm (IMA), is iteratively applied to find CSEs. A similar approach can be applied separately to each constant FFM appearing in Fig. 2.

2. Search Space Expansion

The computational complexity of p-parallel SC can be reduced by eliminating CSEs according to the IMA algorithm. As this approach has multiple search domains each of which is searched independently, it may be possible to find more CSEs if all the search domains are merged together. The search space is first expanded by considering a p-parallel SC unit as a whole, instead of considering each FFM independently like in the previous IMA algorithm. As shown in Fig. 2, the p-parallel SC unit processes p symbols in parallel. By combining p constant FFMs based on (2) and (3), the p-parallel SC unit can be represented as

$$\delta_i(j+1) = \begin{bmatrix} \mathrm{R}_{n-pj-p}^{\mathrm{T}} \\ \mathrm{R}_{n-pj-(p-1)}^{\mathrm{T}} \\ \vdots \\ \mathrm{R}_{n-pj-1}^{\mathrm{T}} \\ \delta_i(j)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathrm{A}_{i,0} \\ \mathrm{A}_{i,1} \\ \vdots \\ \mathrm{A}_{i,p-1} \\ \mathrm{A}_{i,p} \end{bmatrix} = \left[\mathrm{R}(j)\ \delta_i(j)\right] \begin{bmatrix} \mathrm{X}_{i,\mathrm{R}} \\ \mathrm{A}_{i,p} \end{bmatrix} \quad (4)$$

where $\delta_i(j)$ represented by a binary $1\times m$ matrix is the intermediate value of i-th syndrome shown in Fig. 2 and $\mathbf{R}(j)$ is an $1\times mp$ binary matrix representing p symbols of the received codeword to be fed at the j-th iteration. The matrix $\mathbf{X}_{i\mathrm{R}}$ is a $mp\times$m binary matrix relevant to the input symbols. In (4), we can see that all the computations in a single SC unit are collected into a single matrix multiplication as $\delta_i(j)$ becomes S_i at the last iteration.

Once again, all the $2t$ SC units are combined into a matrix multiplication to expand the search space further. The enlarged matrix multiplication that computes $2t$ intermediate values $\delta_i(j)$ for the next iteration is

$$
\begin{aligned}
\Delta(j+1) &= \left[\delta_1(j+1)\ \delta_2(j+1)\ \cdots\ \delta_{2t}(j+1)\right] \\
&= \begin{bmatrix} \mathrm{R}^{\mathrm{T}}_{n-pj-p} \\ \mathrm{R}^{\mathrm{T}}_{n-pj-(p-1)} \\ \vdots \\ \mathrm{R}^{\mathrm{T}}_{n-pj-1} \\ \delta_1(j)^{\mathrm{T}} \\ \delta_2(j)^{\mathrm{T}} \\ \vdots \\ \delta_{2t}(j)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}
\begin{bmatrix} \mathrm{A}_{1,0} & \mathrm{A}_{2,0} & \cdots & \mathrm{A}_{2t,0} \\ \mathrm{A}_{1,1} & \mathrm{A}_{2,1} & \cdots & \mathrm{A}_{2t,1} \\ \vdots & \vdots & \cdots & \vdots \\ \mathrm{A}_{1,p-1} & \mathrm{A}_{2,p-1} & \cdots & \mathrm{A}_{2t,p-1} \\ \mathrm{A}_{1,p} & 0 & \cdots & 0 \\ 0 & \mathrm{A}_{2,p} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \mathrm{A}_{2t,p} \end{bmatrix} \qquad (5) \\
&= \left[\mathrm{R}_j\ \Delta(j)\right] \begin{bmatrix} \mathrm{X}_{\mathrm{R}} \\ \mathrm{X}_{\Delta} \end{bmatrix}
\end{aligned}
$$

where $\mathbf{\Delta}(j)$ is a binary $1\times 2tm$ matrix containing $2t$ intermediate values of the j-th iteration. $\mathbf{X_R}$ is a binary $mp\times 2tm$ matrix associated with the received symbols, and $\mathbf{X_\Delta}$ is a binary $2tm\times 2tm$ matrix denoting $2t$ constant multiplications. Therefore, all the $\delta_i(j+1)$ values can be obtained by multiplying the input vector $[\mathbf{R}(j)\ \mathbf{\Delta}(j)]$ and the constant matrix consisting of $\mathbf{X_R}$ and $\mathbf{X_\Delta}$. $[\mathbf{R}(j)\ \mathbf{\Delta}\ (j)]$ is dependent on the iteration index j, but $\mathbf{X_R}$ and $\mathbf{X_\Delta}$ are constant matrices independent of j. As the single matrix multiplication in (5) covers all the computations relevant to the $2t$ syndromes, we can find more CSEs from (5) than the direct IMA application does. Therefore, a low-complexity SC block can be achieved by employing the reformulated Eq. (5). Compared to the previous architectures associated with multiple matrix multiplications, the proposed architecture has only a single matrix multiplication.

3. Implementation Techniques

Now, we will describe how to find common sub-expressions appearing in the constant matrix multiplication of (5). The overall procedure is very similar to that of IMA [9] as both of them deal with a single matrix multiplication. The only difference is the wideness of search area. To find the best common sub-expressions, we examine every possible pair of two rows in $[\mathbf{X_R}^{\mathrm{T}}\ \mathbf{X_\Delta}^{\mathrm{T}}]^{\mathrm{T}}$ to count the number of bit-wise common 1's between the two rows, and then select a pair associated with the highest count. The addition of the two input terms corresponding to the selected pair is the most common sub-expression in the $[\mathbf{X_R}^{\mathrm{T}}\ \mathbf{X_\Delta}^{\mathrm{T}}]^{\mathrm{T}}$ matrix.

$$
\begin{array}{ccc}
 & \text{Matrix} & \text{Equations} \\
 & \begin{bmatrix} \textcircled{1} & 1 & \textcircled{1} & \textcircled{1} \\ \textcircled{1} & 0 & \textcircled{1} & \textcircled{1} \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} & \begin{array}{l} b_0 = \textcircled{a_0} \wedge \textcircled{a_1} \wedge a_2 \wedge a_3 \\ b_1 = a_0 \\ b_2 = \textcircled{a_0} \wedge \textcircled{a_1} \\ b_3 = \textcircled{a_0} \wedge \textcircled{a_1} \wedge a_2 \end{array} \\
\text{Iteration 1} \downarrow & & \\
 & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ \boxed{1} & 0 & 0 & \boxed{1} & 0 \\ 1 & 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 1 & \boxed{1} & 0 \end{bmatrix} & \begin{array}{l} b_0 = \boxed{a_2} \wedge a_3 \wedge \boxed{v_1} \\ b_1 = a_0 \\ b_2 = v_1 \\ b_3 = \boxed{a_2} \wedge \boxed{v_1} \\ v_1 = a_0 \wedge a_1 \end{array} \\
\text{Iteration 2} \downarrow & & \\
 & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} & \begin{array}{l} b_0 = a_3 \wedge v_2 \\ b_1 = a_0 \\ b_2 = v_1 \\ b_3 = v_2 \\ v_1 = a_0 \wedge a_1 \\ v_2 = a_2 \wedge v_1 \end{array}
\end{array}
$$

Fig. 3. CSE search and matrix update for α^{12} constant multiplication.

After finding a common sub-expression, the matrix is modified to replace the common sub expression with a new term. For every row including the selected pair, all the bits related to the sub-expression are changed to 0, and an 1 is appended to the last column in order to represent the common sub-expression. For the other rows without the selected pair, a 0 is appended to the last column. Lastly, a new row denoting the selected sub-expression is appended to the last row. Therefore, the matrix size is increased by one in both the row and column directions. As one common sub-expression is found at a time, the above procedure is applied to the updated matrix so as to find another common sub-expression, and repeat until there is no sub-expression appearing more than once.

An example of the iterative CSE search is shown in Fig. 3. A multiplication by constant α^{12} is optimized in GF(2^4). The number of XOR gates is minimized through two iterations, and the best matches found in the iterations are marked with circles and squares. As a result, the number of XOR gates is reduced from 6 to 3. The same CSE search algorithm is applied to the matrix enlarged by the proposed method.

The implementation based on the proposed algorithm is described in Fig 4. The conventional structure shown in Fig. 2 requires $2tp$ FFMs each to multiply a received symbol by a constant, and $2t$ FFAs each to add $(p+1)$

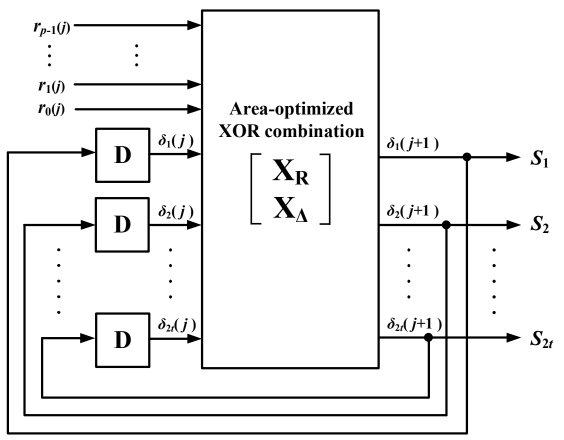

Fig. 4. The proposed *p*-parallel SC block.

values. This paper is the first one that applies the common sub-expression elimination technique to the design of a SC block for RS codes. The single matrix multiplication of (5) combines all the FFMs and completely removes the FFAs because the addition is inherent in the matrix multiplication. As depicted in Fig. 4, the proposed SC structure requires only one matrix multiplication to compute all the $2t$ syndromes. Compared to the previous architecture consisting of many GF multiplications, the proposed architecture reduces the hardware complexity significantly by providing a high chance to share CSEs.

IV. Experimental Results

For various RS codes, we have compared the hardware complexity of syndrome calculation blocks by implementing three different architectures: 1) the conventional architecture shown in Fig. 2, 2) the architecture adopting the IMA, and 3) the proposed architecture. The conventional architecture is a straightforward implementation that computes (2) directly, and the IMA-based architecture separately searches each FFM for common sub-expressions. To maximize the search area for CSEs, the proposed architecture transforms all the syndrome calculations into a matrix multiplication, enlarging the matrix size to $(mp+2mt)\times 2mt$.

Fig. 5 illustrates how the hardware complexity changes according to the parallel factor associated with RS (255, 239, 8) code. The hardware complexity is measured by the number of synthesize equivalent gates

Table 2. Synthesis results for RS (255,239,8) codes

Design Environments	
Technology	130 nm CMOS
Clock Rate	200 MHz
Parallel factor	8
Throuhgput	12.8 Gb/s
Latency	320 ns
Hardawre Complexity Comparison*	
Conventional	22.3K / 96.1K
IMA	19.8K / 93.5K
Proposed	10.9K / 84.7K

* SC block complexity / Overall decoder complexity

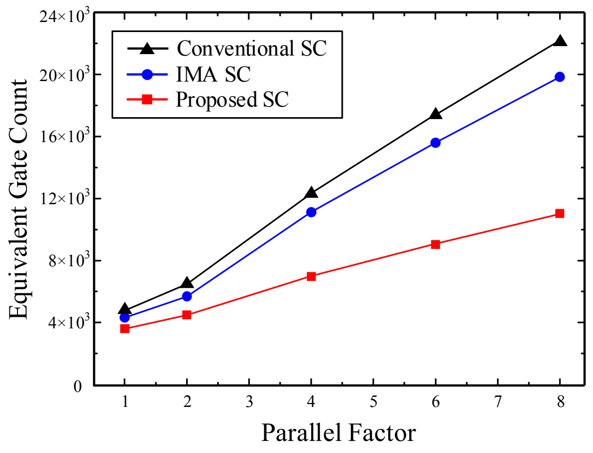

Fig. 5. The complexity of SC versus the parallel factor for RS (255, 239, 8) code.

resulting from with a 0.13μm technology at 200MHz operating frequency. In case of the RS (255, 239, 8) code with a parallel factor of 8, the proposed architecture reduces approximately 50% of XOR gates compared to the conventional architecture. By enlarging the search space, the proposed method guarantees more sharing of CSEs than the IMA-based architecture. The more sharing of CSEs leads to a significant reduction of hardware complexity. Fig. 6 shows how the error-correction capability affects the hardware complexity, which compares three different SC blocks implemented for various RS (255, 255−2*t*, *t*) codes. All the SC blocks are designed with a parallel factor of 8. Note that the proposed architecture has the smallest complexity among the three architectures regardless of the RS codes.

Furthermore, Table 2 describes detailed synthesis results for a RS (255, 239, 8) decoder so as to provide an effect on the overall decoder. Although the SC block has been considered as being associated with a relatively small complexity, the proposed method significantly

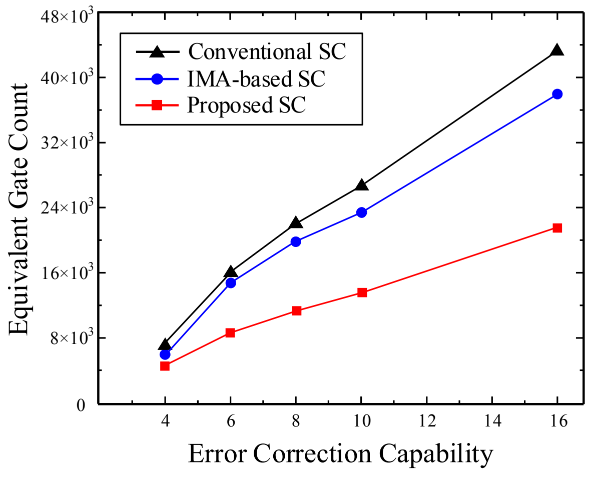

Fig. 6. The complexity of SC versus the error correction capability for RS (255, 255−2t, t) code.

reduces the hardware resources, and thus its effect is not negligible on the overall complexity.

Though the proposed method is effective in reducing hardware complexity, some CSEs have high fan-outs or loading effects, resulting in a slow realization. However, the critical path delay is normally decided by other pipelined blocks such as KES or EPS block, hence these effects are normally negligible even for massive parallel architectures. If these effects degrade the overall throughput, the CSE-elimination algorithm can be modified by giving a constraint on the maximum fan-out load. As a result, the area-optimized SC block can be always implemented by applying the proposed method without deteriorating the decoding throughput.

V. Conclusion

This paper has proposed an area-optimized structure that calculates all syndromes in parallel. The p-parallel SC block is formulized as a matrix multiplication in order to optimize all the syndrome calculations as a whole. The single matrix multiplication makes it possible to find more common sub-expressions that can be shared in the implementation and completely removes additional circuits for modular additions. Experimental results show that the proposed architecture remarkably saves synthesized equivalent gates compared to the conventional and the IMA-based architectures. The proposed method can apply to any RS codes regardless of the parallel factor and error correction capability, and it is effective especially for strong RS decoders correcting a large number of errors.

Acknowledgments

This work was supported by the National Research Foundation (NRF) grand funded by the Korea government (MSIP) (2017R1C1B5015962) and by the Ministry of Science and ICT (MSIT), Korea under the Information Technology Research Center (ITRC) support program (IITP-2018-2016-0-00309). The EDA tools were supported by IDEC, Korea.

References

[1] S. Lin and D. J. Costello, *Error control coding: Fundamentals and Applications,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall Inc., 2004.

[2] S. B. Wicker and V. K. Bhargava, *Reed–Solomon Codes and Their Application*. Picataway, NJ: IEEE Press, 1994.

[3] A. Mondal, S. Thatimattala, V. K. Yalamaddi, and S. S. Garani, "Efficient Coding Architectures for Reed–Solomon and Low-Density Parity-Check Decoders for Magnetic and Other Data Storage Systems," *IEEE Trans. Magnetics,* vol. 54, no. 2, Aug. 2018.

[4] S. B. Lee, C. S. Choi and H. H. Lee, "Two-parallel ReedSolomon based FEC Architecture for Optical Communications," *IEICE Electronics Express*, vol. 5, no. 10, pp. 374-380, May 2008.

[5] J.-I. Park, K. Lee, C.-S. Choi, and H. Lee, "High-speed low-complexity Reed-Solomon decoder using pipelined Berlekamp-Massey algorithm and its folded architcture," *Journal of Semiconductor Technology and Science*, vol. 10, no. 3, pp. 193-202, 2010.

[6] H. Luo, W. Zhang, and Y. Wang, "An algorithm for improving the throughput of serial low-complexity chase soft-decision Reed–Solomon decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 12, no. 12, pp. 3539-3542, Dec. 2017.

[7] K. Seth, K. N. Viswajith, S. Srinivasan, and V. Kamakoti, "Ultra folded high-speed architecture for Reed-Solomon decoder," in Proc. *Int. Conf. VLSI Design*, Hyderabad, India, pp. 517-520, Jan. 2006.

[8] W. ji, W. Zhang, X. Peng, and Y. Liu, "High-efficient Reed–Solomon decoder design using

recursive Berlekamp–Massey architecture," *IET Commun.,* vol. 10, no. 4, pp. 381-386, March 2016.

[9] M. Potkonjak, et. al, "Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 151-165, Feb. 1996.

[10] 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.

[11] 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.

[12] C. Yu and Y-S. Su, "Two-mode Reed-Solomon decoder using a simpliifed step-by-step algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs,* vol. 62, no. 11, pp. 1093-1097, Nov. 2015.

[13] B. Park, J. Park, and Y. Lee, "Area-optimized fully-flexible BCH decoder for Multiple GF Dimensions," *IEEE Access,* vol. 6, pp. 14498-14509, Mar. 2018.

Youngjoo Lee 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.

In-Cheol Park received the B.S. degree in electronic engineering from Seoul National University, Seoul, Korea, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1988 and 1992, respectively. Since June 1996, he has been an Assistant Professor and is currently a Professor with the School of Electrical Engineering, KAIST. Prior to joining KAIST, he was with the IBM T. J. Watson Research Center, Yorktown, NY, USA, from May 1995 to May 1996. His current research interests include computer-aided design algorithms for high-level synthesis and very large scale integration architectures for general-purpose microprocessors.

Hoyoung Yoo received the B.S. degree in Electrical and Electronic Engineering from Yonsei university, Seoul, Korea in 2010, and the M.S. and Ph.D. degrees in Electronics Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2012 and 2016, respectively. Since September 2016, he has been an assistant professor in the department of Electronics Engineering at Chungnam National University. Special area of interests includes VLSI design for error correction codes and 5G communication systems.