Low-Power Parallel Chien Search Architecture Using a Two-Step Approach

Hoyoung Yoo, *Student Member, IEEE*, Youngjoo Lee, *Member, IEEE*, and In-Cheol Park, *Senior Member, IEEE*

***Abstract*—This brief proposes a new power-efficient Chien search (CS) architecture for parallel Bose–Chaudhuri–Hocquenghem (BCH) codes. For syndrome-based decoding, the CS plays a significant role in finding error locations, but exhaustive computation incurs a huge waste of power consumption. In the proposed architecture, the searching process is decomposed into two steps based on the binary matrix representation. Unlike the first step accessed every cycle, the second step is activated only when the first step is successful, resulting in remarkable power saving. Furthermore, an efficient architecture is presented to avoid the delay increase in critical paths caused by the two-step approach. Experimental results show that the proposed two-step architecture for the BCH (8752, 8192, 40) code saves power consumption by up to 50% compared with the conventional architecture.**

***Index Terms*—Bose–Chaudhuri–Hocquenghem (BCH) codes, Chien search (CS), low power, two-step approach.**

I. Introduction

AMONG various error-correction codes used to recover corrupted code words in communications and storage systems, the Bose–Chaudhuri–Hocquenghem (BCH) code [1], [2] is one of the most widely used algebraic codes due to its powerful error-correction performance and affordable hardware complexity. The binary BCH code has been employed in diverse systems such as advanced solid-state storages [3], [4] and optical fiber communication systems [5], and most of these applications are continuously demanding ever higher decoding throughput and ever larger error-correction capability. Since a massive computation is inevitable in satisfying high throughput and strong error-correction capability, power-efficient structure becomes more important in BCH decoding.

In general, a BCH decoder that can correct t bits at maximum is composed of three main blocks, namely, syndrome calculation (SC), key-equation solving (KES), and Chien search (CS) [1], [2]. Given a received code word $R(x)$, the SC computes $2t$ syndromes, and the KES generates the error locator polynomial $\Lambda(x)$ using the syndromes. Finally, error position $E(x)$ is determined by finding the roots of $\Lambda(x)$ based on the CS algorithm. In a parallel BCH decoder, the CS is a major contributor to the power consumption and takes up to a half of overall power consumption [6].

Manuscript received March 13, 2015; revised June 12, 2015 and August 14, 2015; accepted September 5, 2015. Date of publication October 7, 2015; date of current version February 25, 2016. This work was supported in part by the Ministry of Science, ICT and Future Planning as Global Frontier Project/(CISS-2011-0031860) and in part by the IC Design Education Center. This brief was recommended by Associate Editor M. S.-W. Chen.

H. Yoo and I.-C. Park are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: hyyoo.ics@gmail.com; icpark@kaist.edu).

Y. Lee is with the Department of Electronic Engineering, College of Electronics and Information Engineering, Kwangwoon University, Seoul 139-701, Korea (e-mail: yjlee@kw.ac.kr).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2015.2482958

Many studies have proposed efficient structures to lessen the power consumption of the CS. Early termination techniques presented in [6] and [7] are to eliminate redundant computations after finding the last error. An additional error counter is incremented whenever an error is found, and the CS is turned off when the counter matches the number of errors detected in the KES. Although the early termination is simple to implement and effective in the BCH decoder dealing with a small number of errors, its power saving is insignificant when the error-correction capability is not small. In [8], a more efficient method called polynomial order reduction (POR) was proposed to reform the error locator polynomial whenever an error is found. The order of the locator polynomial is decreased by one at a time and eventually becomes zero when all errors are detected. The POR [8] gradually disables the CS by powering down the circuitry associated with one polynomial factor at a time. Although the POR was successful for serial BCH decoders, it is hard to apply the technique to the parallel architecture because of the complicated polynomial update. Furthermore, the power saving of all the previous algorithms, including the early termination [6], [7] and the POR [8], depends on the position of errors. For instance, if errors are located at the end of a code word, the power saving is not as significant as the case that errors are located at the beginning.

In this brief, we propose a new approach in which the parallel CS is decomposed into two steps. The first step is accessed every cycle, but the second step is activated only when the first step is successful, resulting in a less number of access. The proposed two-step approach is conceptually similar to that in [9]. Although the two-step approach, in general, leads to the increase in critical path delay and latency, the drawbacks are resolved in this brief by employing an efficient pipelined structure. Unlike the previous architectures [6]–[8], the proposed architecture can save the power consumption regardless of error locations.

II. Parallel CS Architecture

Let us consider a binary BCH (n, k, t) code over GF(2^m), where n is the code length, k is the message length, and t is the maximal number of correctable error bits. More precisely, $n = k + mt$, where m is the field dimension that satisfies $2^m - 1 \geq n$. During the syndrome-based decoding [1], [2], the error locator polynomial delivered by the KES is expressed as

$$\Lambda(x) = \sum_{j=1}^{t} \lambda_j x^j + 1 = Y(x) + 1. \quad (1)$$

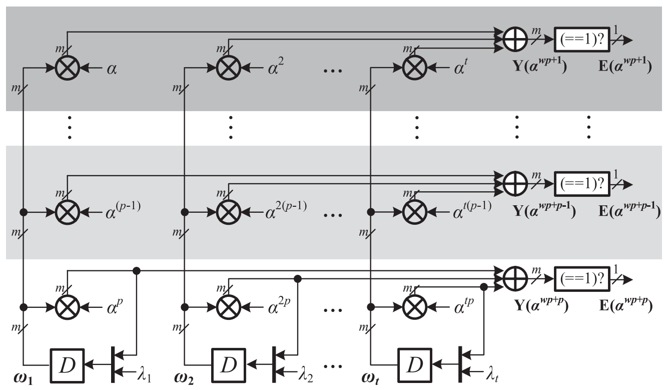

(==1)?
$Y(\alpha^{wp+1})$
$E(\alpha^{wp+1})$
$Y(\alpha^{wp+p-1})$
$E(\alpha^{wp+p-1})$
$Y(\alpha^{wp+p})$
$E(\alpha^{wp+p})$
α α^2 α^t
$\alpha^{(p-1)}$ $\alpha^{2(p-1)}$ $\alpha^{t(p-1)}$
α^p α^{2p} α^{tp}
D
ω_1 ω_2 ω_t
λ_1 λ_2 λ_t
m
1

Fig. 1. Conventional p-parallel CS architecture.

To determine the error position $E(x)$, the CS iteratively substitutes α^i into (1) for $1 \le i \le n$ and detects the presence of an error when $\Lambda(\alpha^i) = 0$ or $Y(\alpha^i) = 1$. In practice, p-parallel CS architecture is widely implemented to achieve a high throughput, where the parallel factor p is the number of α^i substitutions performed at the same time. Fig. 1 describes the p-parallel CS architecture that reduces the number of cycles from n to $\lceil n/p \rceil$ by calculating

$$Y(\alpha^{wp+i}) = \sum_{j=1}^{t} \lambda_j \alpha^{wpj} \alpha^{ij} = \sum_{j=1}^{t} \omega_j(w) \alpha^{ij} \text{ for } 1 \le i \le p. \quad (2)$$

As shown in Fig. 1, an intermediate value ω_j in the jth registers is simultaneously fed to p finite field multipliers (FFMs) located in the same column. As a result, the p-parallel structure is composed of pt FFMs, p t-input m-bit finite field adders, p m-bit comparators, t m-bit registers, and t m-bit multiplexers.

Since all elements over GF(2^m) can be expressed as a $1 \times m$ matrix, the computation in the CS can be formulated by using the binary matrices [10]–[12]. In this brief, ${\alpha^i}_{(a:b)}$ for $0 \le b \le a \le m-1$ is used to precisely denote a part of element α^i ranging from the bth bit to the ath bit. In particular, ${\alpha^i}_{(a)} = {\alpha^i}_{(a:a)}$ indicates a particular bit, and α^i implicitly denotes ${\alpha^i}_{(m-1:0)}$. For example, ${\alpha^4}_{(3:2)} = 1 \times \alpha^3 + 0 \times \alpha^2$ and ${\alpha^4}_{(3)} = {\alpha^4}_{(1)} = {\alpha^4}_{(0)} = 1$ for $\alpha^4 = {\alpha^4}_{(3:0)} = 1 \times \alpha^3 + 0 \times \alpha^2 + 1 \times \alpha^1 + 1 \times \alpha^0$ over GF(2^4). According to [10]–[12], the FFM located on the ith row and jth column in Fig. 1 can be transformed to a binary matrix multiplication as

$$\begin{aligned}
\text{FFM}_{ij} &= \omega_j \alpha^{ij} \\
&= \left(\omega_{j,(m-1)}\alpha^{m-1} + \omega_{j,(m-2)}\alpha^{m-2} + \cdots + \omega_{j,(0)}\right)\alpha^{ij} \\
&= \omega_{j,(m-1)}\alpha^{ij+m-1} + \omega_{j,(m-2)}\alpha^{ij+m-2} + \cdots + \omega_{j,(0)}\alpha^{ij} \\
&= \left[\omega_{j,(m-1)}\omega_{j,(m-2)} \cdots \omega_{j,(0)}\right] \\
&\quad \times \begin{bmatrix} \alpha^{ij+m-1}_{(m-1)} & \alpha^{ij+m-1}_{(m-2)} & \cdots & \alpha^{ij+m-1}_{(0)} \\ \alpha^{ij+m-2}_{(m-1)} & \alpha^{ij+m-2}_{(m-2)} & \cdots & \alpha^{ij+m-2}_{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{ij}_{(m-1)} & \alpha^{ij}_{(m-2)} & \cdots & \alpha^{ij}_{(0)} \end{bmatrix} \\
&= \Omega_j A_{ij}
\end{aligned} \quad (3)$$

where ω_j and α^{ij} are represented as a binary $1 \times m$ matrix Ω_j and a binary $m \times m$ matrix A_{ij}, respectively. By combining pt FFMs based on (2) and (3), the entire p-parallel CS can be reformulated as

$$\begin{aligned}
Y(w) &= \left[Y(\alpha^{wp+1}) \cdots Y\left(\alpha^{wp+(p-1)}\right) Y(\alpha^{wp+p})\right] \\
&= [\Omega_1\ \Omega_2 \cdots \Omega_t] \begin{bmatrix} A_{11} & \cdots & A_{(p-1)1} & A_{p1} \\ A_{12} & \cdots & A_{(p-1)2} & A_{p2} \\ \vdots & \ddots & \vdots & \vdots \\ A_{1t} & \cdots & A_{(p-1)t} & A_{pt} \end{bmatrix} \\
&= \Omega(w) A_Y
\end{aligned} \quad (4)$$

where the iteration index w varies from 0 to $\lceil n/p \rceil - 1$. All the computations of the parallel CS are formulized into a single matrix multiplication of the $1 \times mt$ binary matrix $\Omega(w)$ and the $mt \times mp$ binary constant matrix A_Y [12]. In the parallel CS, the computational complexity is proportional to the parallel factor, the field dimension, and the error-correction capability, and the computation is iteratively processed $\lceil n/p \rceil$ times.

III. Proposed Two-Step CS Architecture

As indicated in (4), the p-parallel CS examines p error positions simultaneously, each of which generates a $1 \times m$ binary matrix denoting a Galois field (GF) element by computing

$$Y(\alpha^{wp+i}) = \sum_{j=1}^{t} \text{FFM}_{ij} = \sum_{j=1}^{t} \Omega_j A_{ij} = [\Omega_1\ \Omega_2 \cdots \Omega_t] \begin{bmatrix} A_{i1} \\ A_{i2} \\ \vdots \\ A_{it} \end{bmatrix} \quad (5)$$

where i ranges from 1 to p. The CS determines the presence of an error when $Y(\alpha^{wp+i})$ is 1, which implies that α^{wp+i} is a root of the error locator polynomial. In the GF of dimension m, the multiplicative identity element, α^0 or α^{2^m-1}, is defined as 1, i.e., $0_{(m-1:1)}1_{(0)}$, more precisely. The main idea comes from the fact that the absence of errors is guaranteed if some bits of $Y(\alpha^{wp+i})$ are not equal to those of $0_{(m-1:1)}1_{(0)}$. In the case of GF(2^4), for example, no presence of errors is guaranteed if $Y(\alpha^{wp+i})_{(3:2)} \ne 0$. Similar to [9], a two-step approach is employed for early detection. In other words, the possibility of error presence is found by searching only the l MSBs rather than the entire m bits. Using this property, (5) can be decomposed into two matrix multiplications as (6)

$$\begin{aligned}
Y(\alpha^{wp+i}) &= [\Omega_1\ \Omega_2\ \cdots\ \Omega_t] \\
&\quad \times \left(\left[\begin{array}{c|c} A_{i1,(m-1:m-l)} & 0_{(m-l-1:0)} \\ A_{i2,(m-1:m-l)} & 0_{(m-l-1:0)} \\ \vdots & \vdots \\ A_{it,(m-1:m-l)} & 0_{(m-l-1:0)} \end{array}\right] + \left[\begin{array}{c|c} 0_{(m-1:m-l)} & A_{i1,(m-l-1:0)} \\ 0_{(m-1:m-l)} & A_{i2,(m-l-1:0)} \\ \vdots & \vdots \\ 0_{(m-1:m-l)} & A_{it,(m-l-1:0)} \end{array}\right] \right) \\
&= \text{concat}\left\{ \Omega(w) \begin{bmatrix} A_{i1,(m-1:m-l)} \\ A_{i2,(m-1:m-l)} \\ \vdots \\ A_{it,(m-1:m-l)} \end{bmatrix}, \Omega(w) \begin{bmatrix} A_{i1,(m-l-1:0)} \\ A_{i2,(m-l-1:0)} \\ \vdots \\ A_{it,(m-l-1:0)} \end{bmatrix} \right\}
\end{aligned} \quad (6)$$

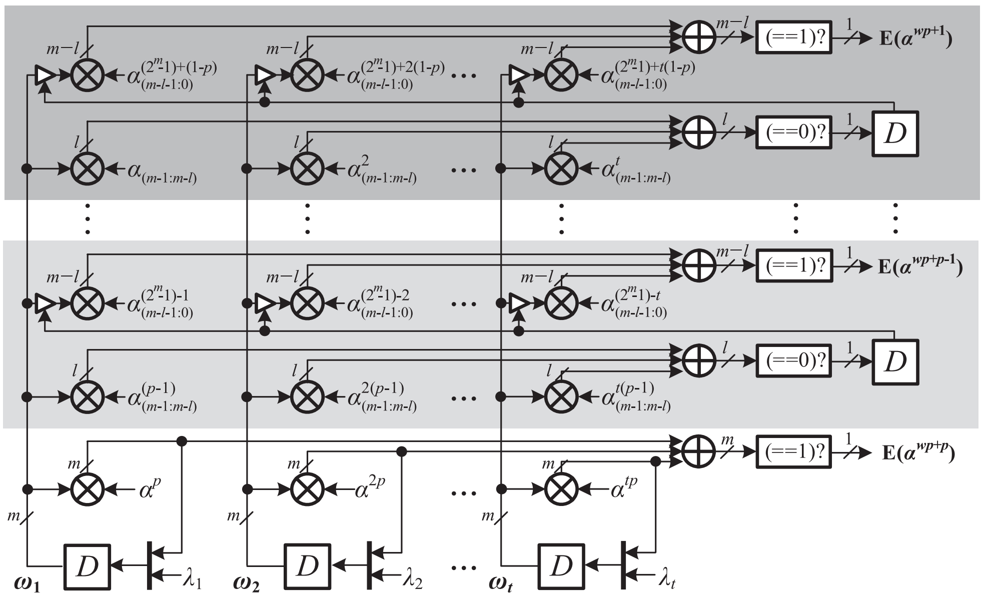


Fig. 2. Proposed two-step structure for p-parallel CS.

$$Y(\alpha^{wp+i}) = \sum_{j=1}^{t} \omega_j(w)\alpha^{pj}\alpha^{ij-pj} = \sum_{j=1}^{t} \omega_j(w+1)\alpha^{(i-p)j}$$
$$= \sum_{j=1}^{t} \omega_j(w+1)\alpha^{(2^m-1)+(i-p)j} \quad (7)$$

$$Y(\alpha^{wp+i}) = \text{concat}\left\{ \Omega(w)\begin{bmatrix} A_{i1,(m-1:m-l)} \\ A_{i2,(m-1:m-l)} \\ \vdots \\ A_{it,(m-1:m-l)} \end{bmatrix}, \Omega(w+1)\begin{bmatrix} A_{(2^m-1)+(i-p)1,(m-l-1:0)} \\ A_{(2^m-1)+(i-p)2,(m-l-1:0)} \\ \vdots \\ A_{(2^m-1)+(i-p)t,(m-l-1:0)} \end{bmatrix} \right\} \quad (8)$$

where concat$\{a, b\}$ stands for the concatenation of two binary matrices a and b. The former and the latter matrix multiplications are responsible for the l MSBs and the $m-l$ LSBs of $Y(\alpha^{wp+i})$, respectively.

Except the FFMs in the pth row, which is in fact used to update the registers, the two-step approach can be applied to the other FFMs in the p-parallel CS shown in Fig. 1. The two-step approach, in general, induces the longer critical path since one computation is decomposed into two small computations in series. To resolve the problem, the long critical path can be broken by inserting delay elements, which makes the two computations operate in a pipelined manner. Thus, the partial FFM for the LSBs is activated at the next clock cycle only when the partial FFM for the MSBs results in zero. Since the intermediate values in the registers are updated every cycle, the straightforward pipelining method is to latch all the intermediate values into separate registers to provide them to the partial FFM for the LSBs at the next cycle. However, this method demands a large amount of hardware resources. To prevent the increase in hardware complexity, in [9], the update of intermediate values was postponed when the former condition is satisfied. Hence, additional clock cycles are inevitable as one cycle is additionally taken whenever one of the $p-1$ former computations is successful.

Unlike the straightforward methods presented in [9], we save only activation signals, which are to enable the inputs to be fed to the latter partial FFMs. For this, (2) is modified to (7), which processes the same computation as (2) with different intermediate values. Note that the multiplication by the multiplicative identity α^{2^m-1} is to make the exponent a positive integer. Using (6) and (7), the two-step approach is finally reformulated as (8). As a result, the proposed two-step approach can detect the case of no errors early without degrading the critical path delay or the performance.

Fig. 2 illustrates the low-power CS architecture based on the proposed two-step approach. According to (8), the m-bit FFMs in the conventional CS are replaced with the pipelined two partial FFMs except for those in the pth row. Given the intermediate values from the registers, the first partial FFM processes the l MSBs and activates the second partial FFM responsible for the remaining $m-l$ LSBs at the next clock cycle only when the output of the former is 0. Otherwise, we can reduce the dynamic switching power by disabling the latter partial FFMs. Since each intermediate register can hold one of all possible GF elements, the latter partial FFM is activated once every 2^l clock cycles on the average. Furthermore, it is worth noting that the sum of the hardware complexity for the former and the latter partial FFMs is almost the same as the conventional FFM. Therefore, additionally required in the proposed architecture are the p 1-bit registers and the $(p-1)t$ m-bit buffers.

It is important to decide how many l bits are appropriate for the former partial FFMs. The more bits are examined in the former, the latter partial FFMs will be accessed less resulting in a large power reduction, but the former partial FFM will suffer from the increased power dissipation, and vice versa. To find an optimal bit width of the first processing, the ratio of the power saving achieved by the two-step approach is simply modeled as

$$P(p,l,m) = \frac{1}{p}\times\frac{m}{m} + \frac{p-1}{p}\times\frac{l}{m} + \frac{p-1}{p}\times\frac{m-l}{m}\times\frac{1}{2^l}. \quad (9)$$

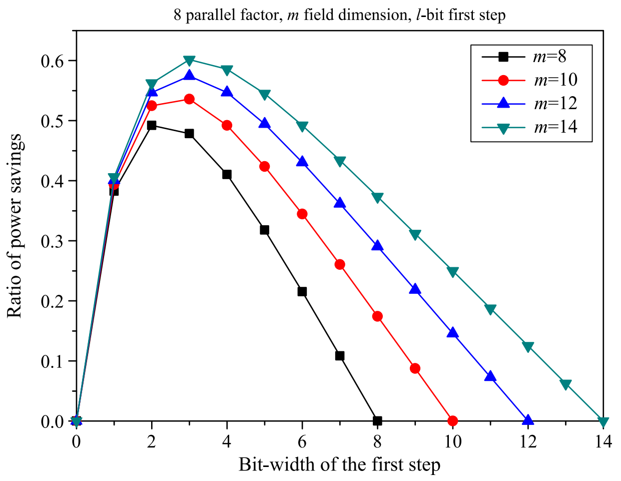


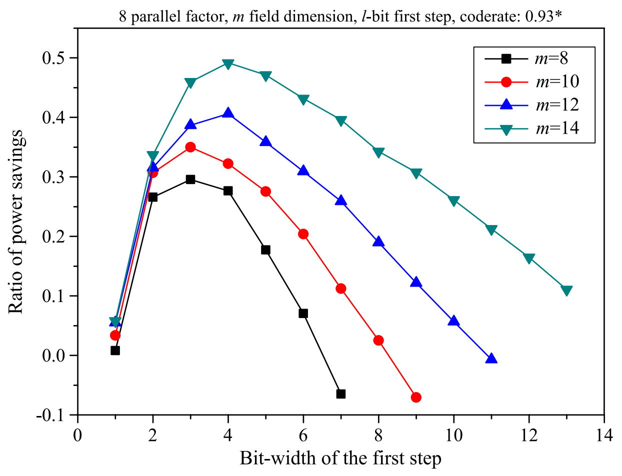
Fig. 3. Estimated power savings versus the bit width of the first step for various field dimensions.

The simplified model is based on two assumptions: 1) The power dissipation mainly comes from FFMs; and 2) the power consumed in an FFM is proportional to the bit width and the number of access. Although the model is highly simplified, it is quite proper to estimate the overall tendency, as the power dominance of FFMs has been observed in a similar application [13]. As a matter of fact, the simplified model is considerably accurate as described in Section IV. The first term stands for the FFMs in the pth row, and the second and third terms denote the first and second partial FFMs for the other FFMs. Based on (9), Fig. 3 describes the ratio of power saving, and the maximum indicates the optimal power saving of a configuration. For example, in the case of $m = 14$, 60% power saving is expected compared with the conventional architecture when the first step processes three MSBs. It is recommended in a practical realization to find a more accurate optimal bit width by investigating several candidate bit widths near the bit width resulting from the model.

IV. Experimental Results

The low-power CS architecture based on the proposed two-step approach is compared with the conventional architecture for various configurations of field dimension, parallel factor, and error-correction capability. All the CS blocks are synthesized in a 130-nm CMOS technology at the operating frequency of 200 MHz, and the equally probable error model [7], [8] is adopted for the power consumption simulations. More precisely, when v errors occur in the BCH (n, k, t) codes, the average bit distance between two adjacent errors becomes n/v since the model assumes that each bit in a received code word is corrupted with the same error probability.

Fig. 4 shows how the field dimension and the bit width of the first step affect the power-saving ratio. For fair comparison, all the BCH codes are designed with a code rate of 0.93 and the parallel factor is set to 8. As shown in Fig. 4, the improvement resulting from the proposed architecture becomes more significant as the field dimension increases, and a small number of bits are sufficient in maximizing the power saving. For instance, the proposed two-step architecture for the BCH



* The BCH (255, 239, 2), BCH (1023, 953, 7), BCH (4095, 3831, 22) and BCH (8752 ,8192, 40) codes are used for m=8, 10, 12, and 14, respectively.

Fig. 4. Simulated power savings versus the bit width of the first step for various field dimensions.

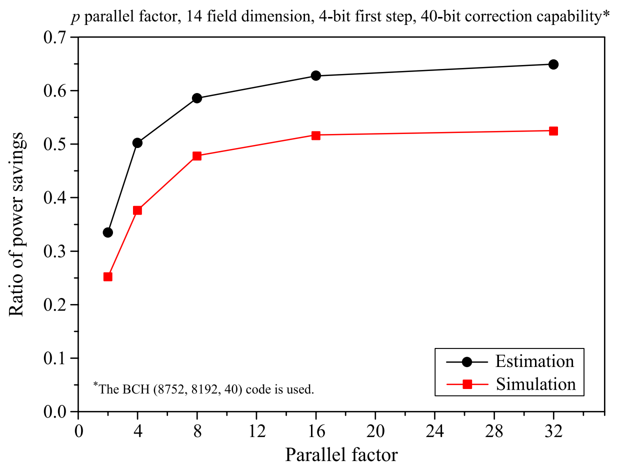


Fig. 5. Power-saving ratio versus parallel factor.

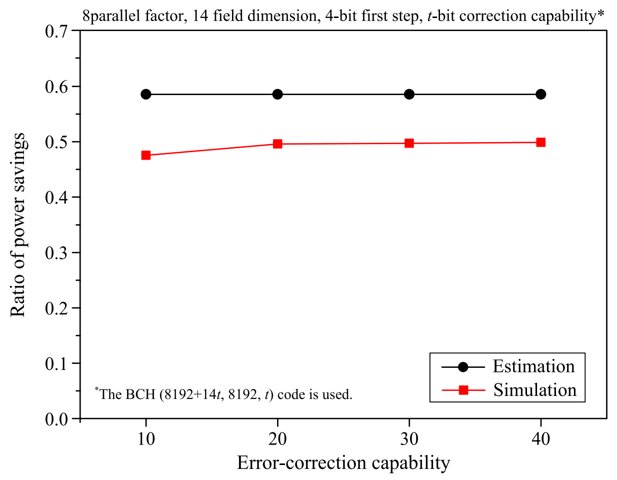


Fig. 6. Power-saving ratio versus error-correction capability.

(8752, 8192, 40) code over GF(2^{14}) achieves 49.3% power saving when the first partial FFM processes four MSBs early. Moreover, Figs. 5 and 6 illustrate how the parallel factor and

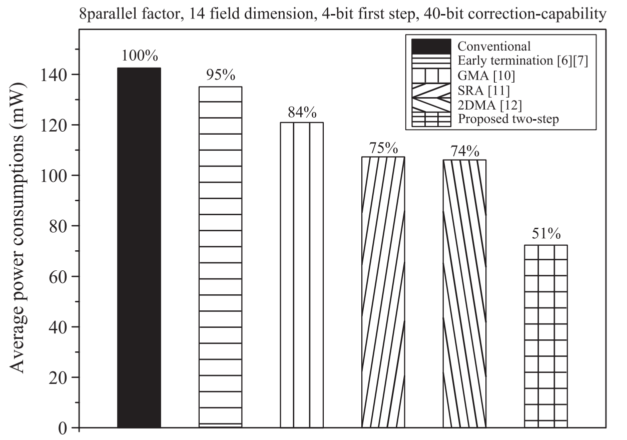


Fig. 7. Average power consumption of eight-parallel CS blocks.

the error-correction capability affect the power saving. As the parallel factor increases, the power-saving ratio converges to 50% approximately and becomes saturated when the parallel factor is 8. It is also shown that the power-saving ratio is almost independent of the error-correction capability. In Figs. 3–6, we can see that the simplified power model (9) can be used as a good estimate, although there is an offset between the estimated and the simulated power savings. Due to the additional buffers addressed in Section III, the proposed architecture increases 10% hardware complexity approximately.

The overall power consumption for the BCH (8752, 8192, 40) code is shown in Fig. 7, which is also achieved with the parallel factor of 8. The early termination [6], [7] is not effective in power saving when the number of errors is not small. Note that the POR [8] is not suitable for the parallel CS since complex computations caused by the polynomial update should be processed in parallel. The power consumption values are also measured for area-efficient architectures [10]–[12] that share common substructures. Although the previous area-efficient architectures are developed to reduce hardware complexity, they are also effective in saving power consumption. However, the power saving is limited to around 25%. On the other hand, the proposed two-step architecture saves almost half of power consumption by activating the second step only when the first step is successful. As the early termination [6], [7] and the area-efficient architectures [10]–[12] are independent of the proposed two-step method, they can be combined into the proposed method to further reduce the power consumption.

V. Conclusion

This brief has presented a new low-power architecture for parallel CS. The conventional CS is decomposed into two steps to achieve a significant power saving by reducing access to the second step. Under the equally probable error model, the low-power CS architecture is compared with the conventional architecture for various configurations of field dimension, parallel factor, and error-correction capability. Experimental results show that the proposed architecture reduces up to 50% power consumption compared with the conventional parallel CS. The power saving becomes more significant as the parallel factor or the field dimension increases. The proposed two-step CS is also applicable to other linear block codes such as the Reed–Solomon codes.

References

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

[2] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.

[3] Y. Lin, C. Yang, C. Hsu, H. Chang, and C. Lee, "A MPCN-based parallel architecture in BCH decoders for NAND Flash memory devices," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 10, pp. 682–686, Oct. 2011.

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

[5] X. Zhang and Z. Wang, "A low-complexity three-error-correcting BCH decoder for optical transport network," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 10, pp. 663–667, Oct. 2012.

[6] K. Lee, S. Lim, and J. Kim, "Low-cost, low-power and high-throughput BCH decoder for NAND flash memory," in *Proc. IEEE ISCAS*, May 2012, pp. 413–415.

[7] Y. Wu, "Low power decoding of BCH codes," in *Proc. IEEE ISCAS*, May 2004, pp. II-369–II-372.

[8] S. Wong, C. Chen, and Q. M. Wu, "Low power Chien search for BCH decoder using RT-level power management," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 2, pp. 338–341, Feb. 2011.

[9] H. Weingarten, E. Sterin, O. A. Kanter, and M. Katz, "Low Power Chien-Search Based BCH/RS Decoding System for Flash Memory, Mobile Communications Devices and Other Applications," U.S. Patent 2010 013 1831 A1, May 27, 2010.

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

[11] J. Cho and W. Sung, "Strength-reduced parallel Chien search architecture for strong BCH codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 5, pp. 427–431, May 2008.

[12] 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.

[13] H. Choi, W. Liu, and W. Sung, "VLSI implementation of BCH error correction for multilevel cell NAND Flash memory," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 5, pp. 843–847, May 2010.