A Search-less DEC BCH Decoder for Low-Complexity Fault-Tolerant Systems

Injae Yoo and In-Cheol Park
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea
{ijyoo.ics@gmail.com, icpark@kaist.edu}

***Abstract*—This paper proposes a new decoding algorithm and its decoder architecture to completely remove the parallel Chien search in double error correcting (DEC) BCH decoders. The proposed algorithm called search-less decoding utilizes a quadratic formula to efficiently compute the roots of an error-location polynomial in the finite field. Since the parallel Chien search block dominates the overall complexity of a conventional DEC BCH decoder, the proposed algorithm is effective in mitigating the hardware complexity. Furthermore, a search-less (44, 32, 2) BCH decoder architecture is proposed for fault-tolerant embedded systems. Compared to the conventional decoder associated with 16-parallel Chien search, the proposed decoder decreases the hardware complexity by 51% without sacrificing the decoding throughput.**

Keywords—BCH decoders; Chien search; double error correcting; fault-tolerant systems; low complexity

I. INTRODUCTION

Fault-tolerant systems work normally in the presence of a certain level of faults. Especially for computer systems included in life-critical applications such as vehicles and medical equipment, fault tolerance is very important [1]. Among various fault-tolerant methods, information redundancy is one of the most important ways to protect the system against errors. Particularly, error-correcting codes, which add parity bits to the data to detect and correct the errors, are widely employed to realize fault-tolerant systems [1]. For embedded systems, the Hamming code that can correct one-bit error has been widely used due to its simplicity. As the feature size and supply voltage are decreased in advanced semiconductor technologies, however, stronger codes are now required to protect the systems from radiation-induced soft errors [2][3]. For the systems consisting of embedded processors and random access memories (RAM), the double-error correcting (DEC) BCH code is in the limelight as an alternative to the Hamming code [2][4].

The BCH code, which is one of the most widely used algebraic codes, can correct multiple errors [5]. Especially, the DEC BCH code has been employed in multi-level cell (MLC) NOR flash memories [6-8] as well as fault-tolerant systems [2][4], because it can be decoded much faster than t-bit-correcting codes where t is greater than two. To decode a t-bit-correcting BCH code, the iterative Berlekamp-Massey (BM) algorithm is usually applied to obtain an error-location polynomial by solving the key equation. In the decoding of the DEC BCH code, on the other hand, the number of unknowns is reduced to two, and thus the error-location polynomial can be derived using the syndrome values [9].

While most of the previous works have mainly focused on reducing the decoding latency, it is also crucial to reduce the hardware complexity of the corresponding decoder. For example, an automotive microcontroller unit (MCU) should be fault-tolerant and implementable with a low complexity because tens of automotive MCUs are included in a car these days. In the decoding of a DEC BCH code, the key equation solver (KES) can be simplified, but the complexity of Chien search that finds the roots of the error-location polynomial becomes significant, since an extremely parallel structure is required to minimize the decoding latency [6-8]. As the block exhaustively searches all the code bits for two faulty locations, its complexity is proportional to the code length when realized with fully parallel architecture.

To remove the parallel Chien search and relax the hardware complexity, this paper proposes a search-less decoding algorithm for DEC BCH codes. The proposed algorithm completely removes the exhaustive search process by solving the error-location polynomial algebraically. Furthermore, a low-complexity architecture for search-less DEC BCH decoding is proposed for fault-tolerant systems. The proposed decoder shows the effectiveness of the proposed search-less algorithm.

The rest of the paper is organized as follows. The conventional DEC BCH decoding algorithm and the proposed search-less decoding algorithm are described in Sections II and III, respectively. Section IV presents the DEC BCH decoder architecture based on the proposed algorithm and its implementation results. Conclusion remarks are made in Section V.

II. CONVENTIONAL DEC BCH DECODING ALGORITHM

The t-error-correcting binary BCH code is conventionally decoded by passing through three major steps. Let the received codeword be represented in a polynomial form as follows,

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2010-0028680).

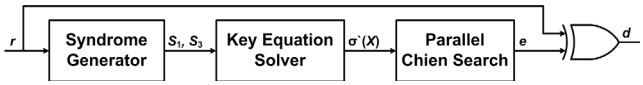

Fig. 1. A block diagram of conventional DEC BCH decoding.

$$\mathbf{r}(X) = r_0 + r_1 X + \cdots + r_{n-1} X^{n-1}. \tag{1}$$

First, the syndrome values of the codeword, $\mathbf{S} = (S_1, S_2, \ldots, S_{2t})$, are computed by

$$\begin{aligned} S_i &= \mathbf{r}(\alpha^i) \\ &= r_0 + r_1 \alpha^i + \cdots + r_{n-1} \alpha^{(n-1)i} \end{aligned}, \tag{2}$$

where α^i is an element of GF(2^m) and n representing the code length is 2^m-1.

The overall structure of a conventional DEC BCH decoder is shown in Fig. 1. Since t is two in case of the DEC code, the syndrome computation is not complicated. Moreover, it is sufficient to compute only odd-indexed syndromes because even-indexed syndromes can be derived by squaring the odd-indexed syndromes.

Second, an error-location polynomial $\boldsymbol{\sigma}(X) = 1 + \sigma_1 X + \cdots + \sigma_v X^v$ is determined by solving the key equation [9]. With assuming $v = t$ errors and setting S_0 to 1, the key equation is

$$S_j = \sum_{i=1}^{t} \sigma_i S_{j-i}, \quad \begin{matrix} j = 1, 3, \ldots, 2t-1 \\ j - i \geq 0 \end{matrix}. \tag{3}$$

The key equation for the DEC BCH code can be solved directly instead of using the conventional BM algorithm [9]. For the codeword with two errors at most, (3) is simply solved as

$$\begin{aligned} \sigma_1 &= S_1, \\ \sigma_2 &= S_1^2 + \frac{S_3}{S_1}. \end{aligned} \tag{4}$$

Therefore, the error-location polynomial is

$$\boldsymbol{\sigma}(X) = 1 + S_1 X + \left(S_1^2 + \frac{S_3}{S_1} \right) X^2. \tag{5}$$

To remove the complex division in (5), both sides are multiplied by S_1 [6-8], which results in a more efficient error-location polynomial,

$$\boldsymbol{\sigma}`(X) = S_1 + S_1^2 X + \left(S_1^3 + S_3 \right) X^2. \tag{6}$$

The KES block in Fig. 1 computes (6).

Lastly, the error locations, x_1 and x_2, are determined by finding the roots of $\boldsymbol{\sigma}`(X)$. Conventionally, the Chien search is utilized to check whether $\boldsymbol{\sigma}`(\alpha^i) = 0$ for $0 \leq i < n$. If there is any error in the codeword and α^l is a root of $\boldsymbol{\sigma}`(X)$, then the exponent of the reciprocal α^{n-l} is the error location. In other words, the received digit r_{n-l} is faulty. As depicted in Fig. 1, the errors in the binary BCH code can be corrected by simply flipping the erroneous bits.

The fully parallel Chien search block evaluates n elements simultaneously, while the serial one checks each element at a time by adopting Horner's rule. Hence, the parallel factor of the block should be determined between the fully parallel and the serial architecture with considering the latency requirement. Since the DEC BCH decoder is required to be very fast in general, the Chien search block has to be realized with a highly parallel structure. Therefore, the search block dominates the overall complexity [6][7]. In [4], on the other hand, the KES and the Chien search blocks are replaced with a lookup table (LUT) that contains all the possible pairs of syndromes and their corresponding error patterns. Though the LUT approach is straightforward, the table size is inhibitive in most cases. To avoid the exhaustive searching, a new DEC BCH decoding algorithm is proposed in this paper.

III. Proposed Search-less DEC BCH Decoding Algorithm

The proposed DEC BCH decoding algorithm is called search-less, since it algebraically solves the error-location polynomial based on the quadratic formula in GF(2^m) [10]. First of all, the number of errors should be derived from two syndrome values, S_1 and S_3. Since α and α^3 of GF(2^m) are the roots of any error-free codeword polynomial, the syndrome values can be rewritten as

$$\begin{aligned} S_1 &= \mathbf{r}(\alpha) = \mathbf{e}(\alpha) = \alpha^{j_1} + \alpha^{j_2} \\ S_3 &= \mathbf{r}(\alpha^3) = \mathbf{e}(\alpha^3) = (\alpha^{j_1})^3 + (\alpha^{j_2})^3 \end{aligned}, \tag{7}$$

where $\mathbf{e}$ denotes the error pattern, and j_1 and j_2 ($0 \leq j_1 < j_2 < n$) represent the error locations. If S_1 is zero, S_3 also becomes zero according to the definition (7) so that the received codeword is error free. If S_1 is not zero and S_1^3 is equal to S_3, it is regarded as a single-error case. In this case, the error-location polynomial (5) is reduced to

$$\boldsymbol{\sigma}(X) = 1 + S_1 x, \tag{8}$$

so that we can find the error location with ease. Otherwise, a case of double errors is detected and the quadratic formula is evaluated to find error locations.

The error-location polynomial (5) can be reformulated as

$$\omega^2 + \omega = \mu, \tag{9}$$

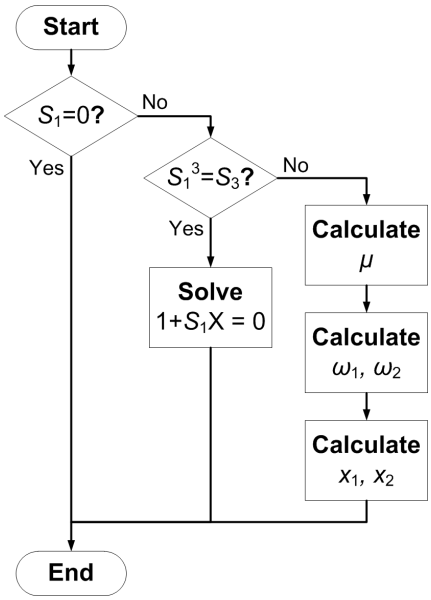

Fig. 2. Overall procedure of the proposed search-less DEC BCH decoding algorithm.

where

$$\omega = \frac{S_1^3 + S_3}{S_1^2} x, \quad \mu = \frac{S_1^3 + S_3}{S_1^3}. \tag{10}$$

Note that (9) is exactly equivalent to (5) because both S_1 and $S_1^3+S_3$ are not zero in the double-error case. Therefore, there must be two solutions for x and accordingly for ω. One solution is computed by

$$\omega_1 = \mu\theta^2 + \left(\mu + \mu^2\right)\theta^{2^2} + \cdots + \left(\mu + \mu^2 + \cdots + \mu^{2^{m-2}}\right)\theta^{2^{m-1}}, \tag{11}$$

where θ is any element of GF(2^m) whose trace Tr(θ) is one, and the other solution is

$$\omega_2 = \omega_1 + 1, \tag{12}$$

as $(\omega_1+1)^2+(\omega_1+1) = \omega_1^2+\omega_1 = \mu$.

Finally, the two roots of (5), x_1 and x_2, can be generated from ω_1 and ω_2 based on (10). For the sake of completeness, the definition of the trace and a few lemmas needed to prove (9)-(11) are described in Appendix.

The overall flow of the proposed decoding algorithm is depicted in Fig. 2. When two bits are faulty, μ, ω_1 and ω_2 are computed in sequel and then the error locations x_1 and x_2 are obtained by using them. The faulty bits are easily corrected by flipping them as in the conventional algorithm. Although calculating (11) seems to be complicated, it can be implemented with a few logic gates as θ is a predetermined constant. In the next section, we will describe how to implement the proposed algorithm for fault-tolerant systems and its efficiency over the conventional DEC BCH decoder that employs the Chien search.

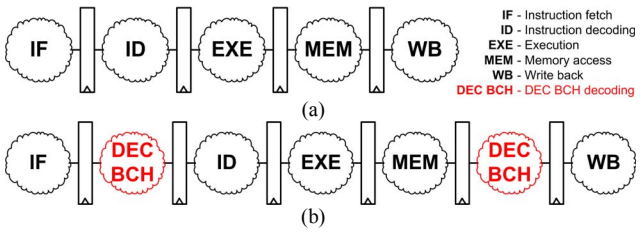

Fig. 3. (a) An example of conventional 5-stage pipeline and (b) its variant considering the DEC BCH code.

IV. Architecture for Search-less DEC BCH Decoding

A. Decoder Specification

A typical embedded system consists of at least one microprocessor and a memory, usually static RAM (SRAM). By expanding the width of the memory, each data word can be DEC BCH encoded before being stored. In the case, two bit-errors in a word can always be corrected by decoding the memory output. However, since accessing the memory is relatively slow compared to the processing speed of a conventional pipelined microprocessor, the DEC BCH decoding should be processed in additional pipeline stages so as not to deteriorate the system performance. This is visualized with a 5-stage pipeline in Fig. 3. Two pipeline-stages are added in Fig. 3(b) because the instruction fetch as well as the data access are relevant to memory accesses. The decoding latency of a DEC BCH decoder should be less than the clock period of the target microprocessor to keep the number of additional pipeline stages minimal.

To specify the DEC BCH code and the requirement of decoding latency, three state-of-the-art MCU products for automotive powertrain control are considered, which are listed in Table I. As all the high-end products are based on 32-bit architecture, the message length should be 32 bits wide. Additional parameters such as code length (n) and Galois field extension (m) are derived from the message length. The final specification of the DEC BCH code is summarized in Table II. Notice that 19 bits are shortened from the 63-bit codeword. In addition, since the clock speed of the recent product approaches 300 MHz, the decoding latency is specified to be

TABLE I. Specifications of Commercial Automotive MCUs

Corporation	MCU	Core	Maximum Clock Speed
Infineon Technologies	AURIX TC299TX (in development)	32-bit TriCore	300 MHz
Freescale Semiconductor	Qorivva MPC5674F	32-bit PowerPC e200z7	264 MHz
STMicroelectronics	SPC564A70L7	32-bit PowerPC e200z4	150 MHz

TABLE II. SPECIFICATION OF THE DEC BCH CODE[a]

Parameter	Value
Galois field extension (m)	6[b]
Shortened code length (n)	44 bits[c]
Message length	32 bits
Parity length	12 bits

[a] The DEC BCH code is generated by $g(X)=1+X^3+X^4+X^5+X^8+X^{10}+X^{12}$.

[b] GF(2^6) is generated by $\varphi(X)=1+X+X^6$.

[c] Original code length is 63($=2^m$-1) bits.

less than 3.33 ns.

B. Decoder Architecture

The proposed search-less decoding algorithm enables a fault-tolerant embedded system to be built with a much lower hardware complexity than the conventional Chien-search-based one.

The overall block diagram of the proposed decoder is shown in Fig. 4. The syndrome generator is designed with a fully parallel structure [3]. On the other hand, the quadratic formula solver (QFS) that generates the error pattern based on S_1 and S_3, which is a core component of the proposed decoder, consists of four components: μ calculator, ω calculator, root calculator, and error-pattern generator. Note that the figure is depicted with focusing on the double-error case, since those blocks dealing with the single-error case are insignificant in terms of the latency and complexity.

The detailed structure of the μ calculator that computes μ as defined in (10) is shown in Fig. 5(a). The division operation in GF(2^m) is complicated, while cubing a syndrome value and adding two syndromes are relatively simple. For the division, a LUT is utilized to convert the 6-bit vector representation of a field element into its 6-bit index value. In Fig. 5, the table is denoted as log() LUT, because the conversion is equivalent to taking the discrete logarithmic operation. After acquiring the indices of the numerator and the denominator, the division in GF(2^6) is achieved by subtracting them. If the subtraction result is negative, it is normalized by adding n, which is equivalent to multiplying 1($=\alpha^n$) to the quotient. The normalized result representing the index of μ, which is notated as log(μ), is used for the calculation of roots. Meanwhile, another table denoted as exp() LUT is employed to convert the final index to the corresponding vector representation. Moreover, since the size of GF(2^6) is relatively small, a LUT is realized with only 384 bits and the lookup latency is negligible.

Following the μ calculator, the ω calculator generates ω_1 and ω_2 according to (11) and (12), respectively. As stated in the previous section, the computation can be realized with a very small hardware. If θ in (11) is α^5, for example, (11) is reduced to

$$\omega_1 = \mu\alpha^{62} + \mu^2\alpha^{24} + \mu^4\alpha^{44} + \mu^8\alpha + \mu^{16}\alpha^{34}. \qquad (13)$$

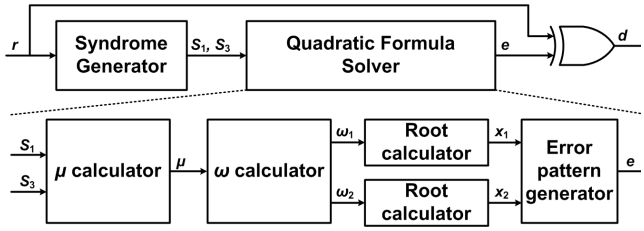

Fig. 4. The architecture of the proposed search-less DEC BCH decoding.

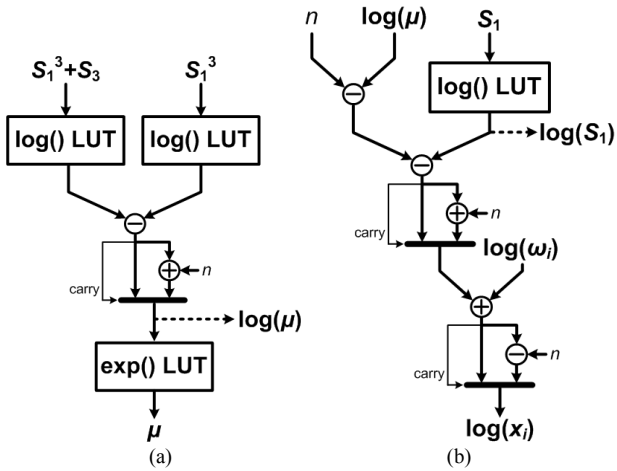

Fig. 5. Structures of (a) the μ calculator and (b) the root calculator.

Since every element in GF(2^6) can be represented with a linear combination of 6 basis elements, *i.e.*, 1, α, α^2, …, α^5, let μ be

$$\mu = a_0 + a_1\alpha + \cdots + a_5\alpha^5, \qquad (14)$$

where a_0, a_1, …, a_5 are binary coefficients. By applying (14) to (13),

$$\omega_1 = a_0\alpha^{21} + a_1\alpha^{47} + a_2\alpha^{31} + a_3\alpha^{14} + a_4\alpha^{62}. \qquad (15)$$

As a result, the transformation from μ to ω_1 can be expressed as

$$\omega_1 = \begin{bmatrix} a_0 a_1 a_2 a_3 a_4 a_5 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \qquad (16)$$

Note that the ω calculator is a direct implementation of (16) and (12) so that it can be realized with a few XOR and AND gates.

Given a ω value, the root calculator generates a root of the error-location polynomial. To compute the root, (10) is reformulated as

TABLE III. COMPARISON OF DEC BCH DECODER IMPLEMENTATIONS[d]

(44, 32, 2) BCH decoders	Decoding latency[f]	Equivalent gate count	Normalized decoder efficiency[g]
Conventional 32-parallel[e]	1.42 ns	13,414	0.60
Proposed search-less	**2.94 ns**	**3,829**	**1.00**
Conventional 16-parallel	2.96 ns	7,734	0.50
Conventional 8-parallel	6.00 ns	4,500	0.42
Conventional 4-parallel	11.2 ns	3,292	0.31
Conventional 2-parallel	23.0 ns	2,519	0.20
Conventional serial	46.4 ns	2,347	0.10

[d] Synthesized in a 65nm CMOS process.

[e] Conventional architecture with a 32-parallel Chien search block.

[f] Multi-cycle decoding except the conventional 32-p. and proposed search-less decoders.

[g] (Decoding latency)$^{-1}$ / equivalent gate count.

$$x = \frac{S_1^{\,2}}{S_1^{\,3} + S_3}\omega = \frac{1}{\mu S_1}\omega\,. \tag{17}$$

When there are double errors in the codeword, S_1 is not zero and $S_1^{\,3}$ is not equal to S_3, which validates the reformulation. Note that μ is reused to reduce the computing delay as well as the hardware complexity. The detailed structure of the root calculator is depicted in Fig. 5(b), where an additional log() LUT is employed to obtain the index of S_1, and the negative index is normalized as in the μ calculator shown in Fig. 5(a). The index of S_1 can be utilized in the single-error case, too. If the summed index obtained by adding two indices is out of the range, the excessive result is normalized by subtracting n, which is the same as dividing the result by $1(=\alpha^n)$.

The error pattern generator is the last major component of the QFS. The error pattern is a 32-bit vector which has at most two ones at the error locations. Some applications may require that faulty parity bits be corrected, but the proposed decoder is designed to correct only the message bits. Since the root calculator provides the index values of faulty bits, it is straightforward to generate the error pattern.

C. Implementation Results

A number of conventional (44, 32, 2) BCH decoders that are all based on the Chien search are implemented by varying the parallel factors so as to compare them with the proposed search-less decoder. All the designs have the same syndrome generator and are synthesized in a 65nm CMOS process. The results are summarized in Table III. Note that only the fully parallel (32-parallel) Chien-search-based decoder and the proposed decoder are possible options for the automotive MCU applications shown in Table I, as the other conventional structures take multiple cycles and thus demand deeper pipelines. For instance, the decoding of a codeword takes 2 cycles in the 16-parallel Chien-search-based decoder and 4 cycles in the 8-parallel one, and so on. The decoding latency shown in Table III represents the overall time to decode a codeword containing a 32-bit message. The equivalent gate count is also presented in the table. In short, the proposed search-less decoder reduces the hardware complexity of the fully parallel conventional decoder by almost 71%, while satisfying the 3.33 ns latency requirement imposed by recent automotive MCUs.

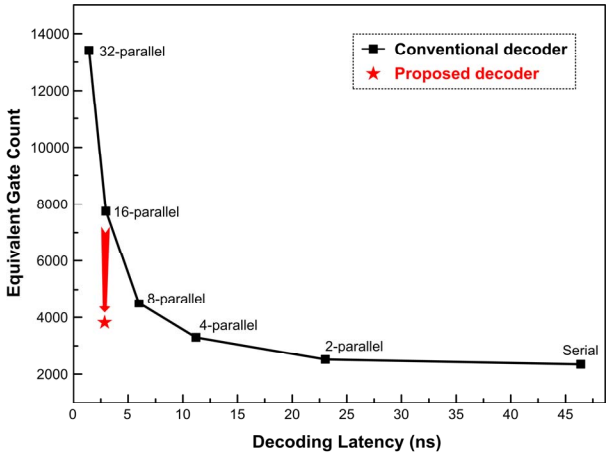

Fig. 6. Implementation results of the proposed search-less decoder and the conventional decoder.

In addition, it is important to note that the proposed architecture can be pipelined without increasing the overall decoding latency much. Therefore, the proposed work can be employed in many other applications with various timing requirements. To generally compare the decoders regardless of the target application, the design efficiency of each decoder, which is defined as the decoding performance per gate, is calculated and normalized by that of the proposed one. In short, the proposed search-less decoder is almost twice as efficient as the conventional decoders. To visualize the implementation results, the equivalent gate count and the decoding latency are drawn in Fig. 6. The result of the proposed architecture is marked with a star, and those of conventional decoders are indicated with squares. In particular, the decoding latency of the 16-parallel Chien-search-based decoder is almost the same as that of the proposed decoder, but the proposed decoder reduces the hardware complexity by almost half thanks to the search-less architecture.

V. CONCLUSION

This paper has presented a new DEC BCH decoding algorithm and its hardware architecture for low-complexity fault-tolerant systems. The proposed algorithm is called search-less because error locations are found by solving a quadratic polynomial equation rather than directly searching for the roots of the error-location polynomial by enumerating all the possible elements. Since the conventional DEC BCH decoder suffers from the high complexity caused by the parallel Chien search, the proposed algorithm is a promising way to reduce the hardware complexity of a DEC BCH decoder. Moreover, the search-less decoder was implemented for error-resilient digital systems. Compared to the conventional decoder with 16-parallel Chien search, the proposed decoder reduces the hardware complexity by half without degrading the decoding performance.

APPENDIX
PROOF OF (9)-(11)

Let K be a Galois field GF(q^m) and its subfield F be GF(q). If α is an element of K, its trace relative to F is defined as

$$\mathrm{Tr}(\alpha) = \alpha + \alpha^q + \cdots + \alpha^{q^{m-1}} . \quad (18)$$

Some important properties of the trace are given here without proofs [10]. For all $\alpha, \beta \in K$,

1) $\mathrm{Tr}(\alpha+\beta) = \mathrm{Tr}(\alpha) + \mathrm{Tr}(\beta)$

2) $\mathrm{Tr}(\alpha^q) = \mathrm{Tr}(\alpha)$

3) The trace function maps K onto F

The properties are essential to prove (9)-(11). The equations to be proved are generalized and rewritten as Theorem 1, which will be proved by using Lemma 1 and Lemma 2 described in [10].

Theorem 1: Assume that α, β are elements of K. If α is defined as

$$\alpha = \beta - \beta^q , \quad (19)$$

and such β exists, it can be obtained by

$$\beta = \alpha\theta^q + \left(\alpha + \alpha^q\right)\theta^{q^2} + \cdots + \left(\alpha + \alpha^q + \cdots + \alpha^{q^{m-2}}\right)\theta^{q^{m-1}} . \quad (20)$$

Lemma 1: Assume that α and θ are elements of K. If β is defined as

$$\beta = \alpha\theta^q + \left(\alpha + \alpha^q\right)\theta^{q^2} + \cdots + \left(\alpha + \alpha^q + \cdots + \alpha^{q^{m-2}}\right)\theta^{q^{m-1}} , \quad (21)$$

then

$$\beta - \beta^q = \alpha\left(Tr(\theta) - \theta\right) - \theta\left(Tr(\alpha) - \alpha\right) . \quad (22)$$

Proof of Lemma 1: By the definition of β,

$$\beta^q = \alpha^q\theta^{q^2} + \left(\alpha^q + \alpha^{q^2}\right)\theta^{q^3} + \cdots + \left(\alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{m-1}}\right)\theta^{q^m} . \quad (23)$$

Subtracting (23) from (21), therefore, we have

$$\beta - \beta^q = \alpha\left(\theta + \theta^{q^2} + \cdots + \theta^{q^{m-1}}\right) - \left(\alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{m-1}}\right)\theta^{q^m} . \quad (24)$$

By using the definition of the trace and the fact that $\theta^{q^m} = \theta$, (24) can be rewritten as (22). □

Lemma 2: Assume that α and β are elements of K. If α is defined as

$$\alpha = \beta - \beta^q , \quad (25)$$

the trace of α is always zero.

Proof of Lemma 2: By the first and the second properties of the trace, the trace of α is

$$\begin{aligned} \mathrm{Tr}(\alpha) &= \mathrm{Tr}\left(\beta - \beta^q\right) \\ &= \mathrm{Tr}(\beta) - \mathrm{Tr}\left(\beta^q\right) \\ &= \mathrm{Tr}(\beta) - \mathrm{Tr}(\beta)^q . \\ &= \mathrm{Tr}(\beta) - \mathrm{Tr}(\beta) \\ &= 0 \end{aligned} \quad (26)$$

□

Due to the third property of the trace, it is certain that there is an element of K, θ, whose trace is one. By applying this fact and Lemma 2 to Lemma 1, Theorem 1 is proved.

REFERENCES

[1] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*, San Francisco, CA: Morgan-Kaufman Publishers, 2007.

[2] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 305-316, Sep. 2005.

[3] G. R. Redinbo, L. M. Napolitano, Jr., and D. D. Andaleon, "Multibit correcting data interface for fault-tolerant systems," *IEEE Trans. Comput.*, vol. 42, no. 4, pp. 433-446, Apr. 1993.

[4] R. Naseer and J. Draper, "Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs," in *Proc. Eur. Solid-State Circuits Conf.*, Sep. 2008, pp. 222-225.

[5] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inform. Control*, vol. 3, no. 1, pp. 68-79, Mar. 1960.

[6] X. Wang, D. Wu, C. Hu, L. Pan, and R. Zhou, "Embedded high-speed BCH decoder for new-generation NOR flash memories," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2009, pp. 195-198.

[7] W. Xueqiang, P. Liyang, W. Dong, H. Chaohong, and Z. Runde, "A high-speed two-cell BCH decoder for error correcting in MLC NOR flash memories," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 11, pp. 865-859, Nov. 2009.

[8] X. Wang, D. Wu, L. Pan, R. Zhou, and C. Hu, "An on-chip high-speed 4-bit BCH decoder in MLC NOR flash memories," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2009, pp. 229-232.

[9] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Code*, 2nd ed. Cambridge, MA: MIT Press, 1972.

[10] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Boston, MA: Kluwer Academic, 1987.