Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems | School of Computing and Information Systems

11-1987

Decoding of DBEC-TBED Reed-Solomon Codes

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Daniel J. Jr. COSTELLO
University of Notre Dame

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Information Security Commons

Citation

DENG, Robert H. and COSTELLO, Daniel J. Jr.. Decoding of DBEC-TBED Reed-Solomon Codes. (1987). *IEEE Transactions on Computers*. 36, (11), 1359-1363.

Available at: https://ink.library.smu.edu.sg/sis_research/149

This Journal Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Decoding of DBEC-TBED Reed-Solomon Codes

DENG, Robert H. & COSTELLO, Daniel J. Jr.

Published in IEEE Transactions on Computers, 1987 November, 36 (11), 1359-1363
https://doi.org/10.1109/TC.1987.5009476

Abstract: A problem in designing semiconductor memories is to provide some measure of error control without requiring excessive coding overhead or decoding time. In LSI and VLSI technology, memories are often organized on a multiple bit (or byte) per chip basis. For example, some 256K bit DRAM's are organized in 32K ?? 8 bit-bytes. Byte-oriented codes such as Reed-Solomon (RS) codes can provide efficient low overhead error control for such memories. However, the standard iterative algorithm for decoding RS codes is too slow for these applications. In this correspondence we present a special decoding technique for double-byte-error-correcting (DBEC), triple-byte-error-detecting (TBED) RS codes which is capable of high-speed operation. This technique is designed to find the error locations and the error values directly from the syndrome without having to use the iterative algorithm to find the error locator polynomial.

Keywords: Byte error correction and detection, byte-organized memory systems, error control coding, Reed-Solomon codes, VLSI memory systems

I. Introduction

Error control has long been used to improve the reliability of computer memory systems [1]. The most common approach has been to use a variation of the Hamming codes such as the single-error-correcting and double-error-detecting (SEC–DED) binary codes first introduced by Hsaio [2]. These codes are particularly effective for correcting and detecting errors in memories with a 1 bit per chip organization. In these memories a single chip failure can affect at most one bit in a codeword.

Large scale integration (LSI) and very large scale integration (VLSI) memory systems offer significant advantages in size, speed, and weight over earlier memory systems. These memories are normally packaged with a multiple bit (or byte) per chip organization. For example, some 256K bit dynamic random access memories (DRAM's) are organized in 32K $\times$ 8 bit-bytes. In this case, a single chip failure can affect several or all of the bits in a byte, thus exceeding the error correcting and detecting capability of SEC–DED codes.

Several papers have been written recently trying to extend the SEC–DED codes to include byte errors [3]–[9]. In this correspondence we investigate the use of Reed–Solomon (RS) codes for correcting and detecting byte errors in computer memories. RS codes are a class of nonbinary codes with symbols in the Galois field of 2^m elements (GF(2^m)). These codes are maximum distance separable (MDS), and thus can provide efficient low overhead error control for byte-organized memories, since symbol error correction in GF(2^m) is equivalent to correcting an m-bit byte.

For computer memory applications, decoding must be fast and efficient. A typical RS decoding procedure is to first calculate the error syndromes, then use the iterative algorithm [10] to form an error locator polynomial, and finally to search for the roots of the error locator polynomial, find the error values, and make the actual corrections. The calculation of the error locator polynomial is a major step in decoding RS codes, and it remains a bottleneck for high-speed decoding, since most errors are single errors and checking for multiple errors is time consuming. High-speed decoding can be achieved by using the table-lookup method [1]. However, even for moderate code lengths, the implementation of table-lookup decoding is impractical, since either a large amount of storage or very complex logical circuitry is needed.

In this correspondence we investigate a special high-speed decoding technique for double-byte-error-correcting (DBEC), triple-byte-error-detecting (TBED) RS codes. This technique is designed to locate and correct the errors directly without having to find the error locator polynomial. The occurrences of errors are determined by directly testing the weight of the syndrome, denoted by $w(s)$. In decoding the DBEC–TBED RS code with five parity symbols, if $w(s) = 1$ or 2, we show that the number of byte errors $E \geqslant 3$. If $w(s) = 3$ or 4, a simple test is required to determine if $E = 2$ or $\geqslant 3$. If $w(s) = 5$, the decoder quickly determines if $E = 1$, 2, or 3. Thus, decoding can be carried out in parallel, which in effect increases the decoding speed. Double byte error correction is done by forming a quadratic equation $x^2 + x + K = 0$, the solution of which gives the two byte error locations. The constant K can be determined directly from the syndrome. In this equation, only K contains information about the error locations. If a short table is used, with two error locations corresponding to each value of K, the decoding speed can be made even higher.

II. Decoding of a $d_{\min} = 6$ DBEC–TBED Code

The $d_{\min} = 6$ RS Code and its Properties

The generator polynomial for the $d_{\min} = 6$ RS code is given by

$$g(x) = \sum_{i=-2}^{2} (x + \alpha^i), \tag{1}$$

where α is a primitive element of GF(2^m). The parity-check matrix of the code specified by (1) can be written as

$$H = \begin{bmatrix} 1 & \alpha^{-2} & (\alpha^{-2})^2 & \cdots & (\alpha^{-2})^{n-1} \\ 1 & \alpha^{-1} & (\alpha^{-1})^2 & \cdots & (\alpha^{-1})^{n-1} \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & (\alpha)^2 & \cdots & (\alpha)^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \end{bmatrix}, \tag{2}$$

where $n \leqslant 2^m - 1$. Because the code has $d_{\min} = 6$, it is capable of correcting any two or fewer byte errors and simultaneously detecting any combination of three byte errors [1].

Let $\boldsymbol{v} = (v_0, v_1, \cdots, v_{n-1})$ be a code vector that is written into memory. Let $\boldsymbol{r} = (r_0, r_1, \cdots, r_{n-1})$ be the corresponding (possibly noisy) vector that is read from memory. Because of possible chip failures, $\boldsymbol{r}$ may be different from $\boldsymbol{v}$. The vector difference[1]

$$\boldsymbol{e} \triangleq \boldsymbol{r} - \boldsymbol{v} = \boldsymbol{r} + \boldsymbol{v} = (e_0, e_1, \cdots, e_{n-1}), \tag{3}$$

where $e_i \neq 0$ for $r_i \neq v_i$ and $e_i = 0$ for $r_i = v_i$, is called the *error pattern*.

When $\boldsymbol{r} = \boldsymbol{v} + \boldsymbol{e}$ is read, the decoder computes the syndrome

$$s^T = \boldsymbol{r}\boldsymbol{H}^T = (\boldsymbol{v} + \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{e}\boldsymbol{H}^T = (s_{-2}, s_{-1}, s_0, s_1, s_2). \tag{4}$$

The syndrome corresponding to a single byte error is

$$s_{-2} = e_i\alpha^{-2i}, \tag{5.1}$$

$$s_{-1} = e_i\alpha^{-i}, \tag{5.2}$$

$$s_0 = e_i, \tag{5.3}$$

$$s_1 = e_i\alpha^{i}, \tag{5.4}$$

$$s_2 = e_i\alpha^{2i}, \tag{5.5}$$

where e_i is the error value and i is the error location, $0 \leq i \leq n - 1$, and the syndrome corresponding to a double byte error is

$$s_{-2} = e_i\alpha^{-2i} + e_j\alpha^{-2j}, \tag{6.1}$$

$$s_{-1} = e_i\alpha^{-i} + e_j\alpha^{-j}, \tag{6.2}$$

$$s_0 = e_i + e_j, \tag{6.3}$$

$$s_1 = e_i\alpha^{i} + e_j\alpha^{j}, \tag{6.4}$$

$$s_2 = e_i\alpha^{2i} + e_j\alpha^{2j}, \tag{6.5}$$

where $0 \leq i \leq j \leq n - 1$.

Before proceeding, we need to prove some properties of the $d_{\min} = 6$ RS code which will be used later.

Property 1: Let $s_d = (s_{-2}, s_{-1}, s_0, s_1, s_2)^T$ be the syndrome corresponding to a double byte error. Let N denote the number of zero elements in s_d. Then $N \leq 2$, and equality holds in only two cases:

1) $s_{-1} = s_2 = 0$;

2) $s_1 = s_{-2} = 0$.

Proof: See Appendix A.

Property 2: Let $s_d = (s_{-2}, s_{-1}, s_0, s_1, s_2)^T$. Then

$$s_2 s_{-2} + s_0^2 \neq 0,$$

$$s_1 s_{-2} + s_{-1} s_0 \neq 0,$$

$$s_0 s_1 + s_2 s_{-1} \neq 0,$$

for all double byte errors.

[1] Addition and subtraction are equivalent over GF(2^m).

Proof: This can be obtained directly from (6.1)–(6.5).

Decoding Using the Quadratic Equation

We now show that the well-known quadratic equation over GF(2^m) can be used to decode the $d_{\min} = 6$ RS code. If α is a primitive element of GF(2^m), then $\alpha^{-i} + \alpha^{-j} \neq 0$, $0 \leq i < j \leq 2^m - 2$. From (6.1) and (6.3) we have

$$e_i = \frac{\det\begin{bmatrix} s_0 & 1 \\ s_{-2} & \alpha^{-2j} \end{bmatrix}}{\det\begin{bmatrix} 1 & 1 \\ \alpha^{-2i} & \alpha^{-2j} \end{bmatrix}} = \frac{s_{-2}+s_0\alpha^{-2j}}{(\alpha^{-i}+\alpha^{-j})^2}.$$

From (6.2) and (6.3) we have

$$e_i = \frac{\det\begin{bmatrix} s_0 & 1 \\ s_{-1} & \alpha^{-j} \end{bmatrix}}{\det\begin{bmatrix} 1 & 1 \\ \alpha^{-i} & \alpha^{-j} \end{bmatrix}} = \frac{s_{-1}+s_0\alpha^{-j}}{\alpha^{-i}+\alpha^{-j}}.$$

Therefore,

$$\frac{s_{-1}+s_0\alpha^{-j}}{\alpha^{-i}+\alpha^{-j}} = \frac{s_{-2}+s_0\alpha^{-2j}}{(\alpha^{-i}+\alpha^{-j})^2}.$$

After multiplying both sides by $(\alpha^{-i} + \alpha^{-j})^2 \neq 0$ and simplifying, the above equation becomes

$$s_{-1}(\alpha^i+\alpha^j)+s_{-2}\alpha^i\alpha^j+s_0=0. \tag{7}$$

In the same way, from (6.3)–(6.5), we obtain

$$s_1(\alpha^i+\alpha^j)+s_0\alpha^i\alpha^j+s_2=0. \tag{8}$$

Now define

$$\gamma_1 \triangleq s_0^2+s_{-1}s_1, \tag{9.1}$$

$$\gamma_2 \triangleq s_2 s_{-2}+s_0^2, \tag{9.2}$$

$$\gamma_3 \triangleq s_1 s_{-2}+s_{-1}s_0, \tag{9.3}$$

$$\gamma_4 \triangleq s_0 s_1+s_2 s_{-1}. \tag{9.4}$$

Solving (7) and (8) for $\alpha^i + \alpha^j$ and $\alpha^i\alpha^j$, we obtain

$$b \triangleq \alpha^i+\alpha^j=\frac{\gamma_2}{\gamma_3}, \tag{10.1}$$

$$c \triangleq \alpha^i+\alpha^j=\frac{\gamma_4}{\gamma_3}, \tag{10.2}$$

for $\gamma_3 \neq 0$. Therefore, α^i and α^j are the roots of

$$y^2+by+c=0. \tag{11}$$

This is the well-known quadratic equation over GF(2^m). We will see later that it plays an important role in decoding. Therefore, we call it the *decoding equation*. Equation (11) can be rewritten as

$$x^2+x+K=0 \tag{12}$$

by letting

$$y=xb, \tag{13}$$

where

$$K \triangleq c/b^2. \tag{14}$$

The formula for the roots of the quadratic equation is $(-b \pm \sqrt{b^2 - 4c})/2$. Unfortunately, for finite fields of characteristic two, this formula is not applicable because the denominator is zero. However, there are several known approaches to solving this equation [10]–[13]. The method given in [12] is probably the best approach, and we summarize it in Appendix B.

Decoding the DBEC–TBED Code

Suppose that a single-byte error with error value e_i at location i occurs. From (5.1)–(5.5) we see that

$$s_i \neq 0, \qquad \text{for } i=-2,\ -1,\ 0,\ 1,\ 2, \tag{15.1}$$

and

$$\frac{s_{-1}}{s_{-2}}=\frac{s_0}{s_{-1}}=\frac{s_1}{s_0}=\frac{s_2}{s_1}=\alpha^i. \tag{15.2}$$

Note that (15.2) is equivalent to $\gamma_1 = \gamma_3 = \gamma_4 = 0$. That is, whenever a single-byte error occurs, $s_i \neq 0$ for $i = -2, -1, 0, 1, 2$, and $\gamma_1 = \gamma_3 = \gamma_4 = 0$. From (5.3) and (5.4) we have

$$\alpha^i=\frac{s_1}{s_0}, \tag{16.1}$$

$$e_i=s_0, \tag{16.2}$$

where i gives the error location and e_i is the error value of a single byte error.

If a double-byte error occurs, from property 2 and (9.2)–(9.4) we know that $\gamma_2 \neq 0$, $\gamma_3 \neq 0$, and $\gamma_4 \neq 0$. Therefore, b and c in (10.1) and (10.2) exist. Hence, (11) has two roots, α^i and α^j. In other words, whenever a double byte error occurs, its error locations can be found by solving the decoding equation.

Since $\alpha^i + \alpha^j \neq 0$, for $0 \leq i < j \leq 2^m - 2$, when α is a primitive element of GF(2^m), (6.3) and (6.4) imply that

$$e_i = \frac{\det\begin{bmatrix} s_0 & 1 \\ s_1 & \alpha^{j} \end{bmatrix}}{\det\begin{bmatrix} 1 & 1 \\ \alpha^{i} & \alpha^{j} \end{bmatrix}} = \frac{s_0\alpha^j+s_1}{\alpha^i+\alpha^j}, \tag{17.1}$$

and

$$e_j=s_0+e_i, \tag{17.2}$$

where e_i and e_j are the error values at locations i and j of the double byte error.

Let s_s denote the syndrome corresponding to a single byte error and s_t denote the syndrome corresponding to a triple byte error. Then [1]

$$s_s \neq s_d \neq s_t. \tag{18}$$

Based on (18) and properties 1 and 2, we see that if more than two elements of the syndrome $s = (s_{-2}, s_{-1}, s_0, s_1, s_2)^T$ equal zero, but at least one of them does not equal zero, or if γ_2, γ_3, and γ_4 are all not equal to zero, but at least one of them does equal zero, or if the decoding equation (11) does not have roots in GF(2^m), then at least three byte errors have occurred.

Decoding Scheme for the DBED–TBED RS Code (see Fig. 1)

Read r, and calculate the syndrome $s^T = rH_2^T = (s_{-2}, s_{-1}, s_0, s_1, s_2)$. Let $w(\gamma')$ and $w(\gamma'')$ denote the Hamming weights of $\gamma' \triangleq (\gamma_1, \gamma_3, \gamma_4)$ and $\gamma'' \triangleq (\gamma_2, \gamma_3, \gamma_4)$, respectively.

1) If $w(s) = 0$, no errors are detected. If $w(s) = 1$ or 2, $E \geq 3$ errors are detected. If $w(s) = 3$ or 4, $E \geq 2$, and decoding proceeds in step 3). If $w(s) = 5$, $E \geq 1$, and decoding proceeds in step 2).

2) Compute γ'. If $w(\gamma') = 0$, $E = 1$, and calculating $\alpha^i = s_1/s_0$ gives the error location i. Set the error value $e_i = s_0$. If $w(\gamma') \neq 0$, $E \geq 2$ errors are detected, and decoding proceeds in step 3).

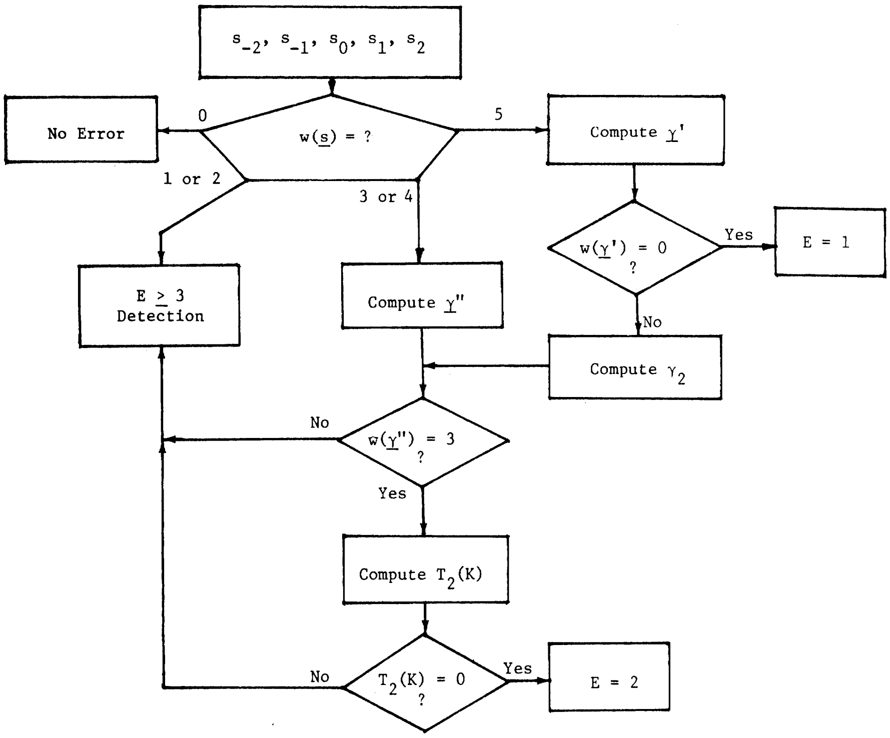

Fig. 1. DBEC–TBED decoder error location calculator.

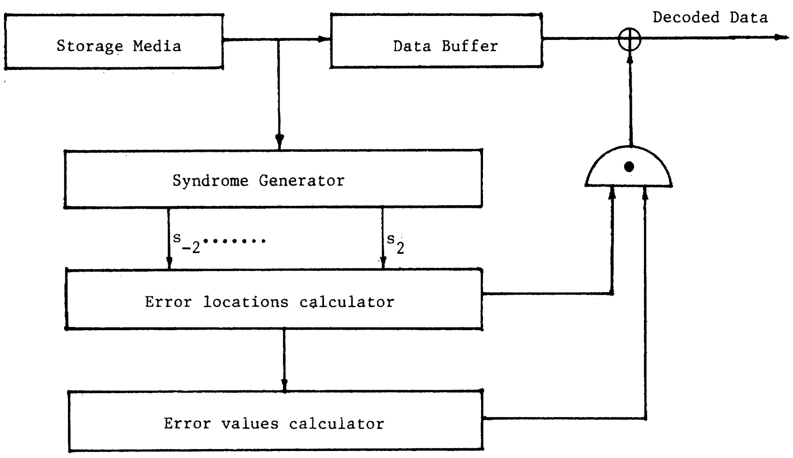

Fig. 2. Block diagram of a DBEC–TBED decoder.

3) Compute γ''. If $w(\gamma'') = 3$, compute K and $T_2(K)$. If $T_2(K) = 0$, $E = 2$, and we must solve (12) to find the roots α^i and α^j. Compute $e_i = (s_0\alpha^j + s_1)/(\alpha^i + \alpha^j)$ and $e_j = s_0 + e_i$, and correct a double-byte error with error values e_i and e_j at locations i and j, respectively. If $w(\gamma'') \neq 3$, or $T_2(K) = 1$, $E \geq 3$ errors are detected.

Fig. 2 is a block diagram of the DBEC–TBED decoder.

III. Conclusions

We have presented a new decoding technique for double-byte-error-correcting (DBEC), triple-byte-error-detecting (TBED) RS codes. This decoding technique is based directly on the syndrome, and does not involve applying the iterative algorithm to find the error locator polynomial. Hence, high-speed decoding can be achieved, making these codes well suited for error correction and detection in byte-organized computer memory systems such as LSI and VLSI chips.

Code efficiency is high since only five parity symbols are used in the code. In addition, the basic code length n can be selected to match the organization of the memory (as long as $n \leqq 2^m - 1$) without changing the decoding method. However, efficiency is maximized when $n = 2^m - 1$ is chosen.

Appendix A

Proof of Property 1: It can easily be seen that the vectors $(\alpha^{-2i}, \alpha^{-2j})$, $(\alpha^{-i}, \alpha^{-j})$, $(1, 1)$, (α^i, α^j), and $(\alpha^{2i}, \alpha^{2j})$, where $0 \leq i < j \leq 2^m - 2$, are always pairwise linearly independent except for the following two pairs:

1) $(\alpha^{-i}, \alpha^{-j})$, $(\alpha^{2i}, \alpha^{2j})$;

2) (α^i, α^j), $(\alpha^{-2i}, \alpha^{-2j})$.

These two pairs are linearly independent for some values of i and j.

First we show that if $s_0 = 0$, then $s_k \neq 0$, $k = -2, -1, 1, 2$. Suppose $s_k = 0$ for some $k \neq 0$. From (6.1)–(6.5), we have

$$\begin{bmatrix} s_0 \\ s_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = e_i \begin{bmatrix} 1 \\ \alpha^{ki} \end{bmatrix} + e_j \begin{bmatrix} 1 \\ \alpha^{kj} \end{bmatrix},$$

where $e_i \neq 0$, $e_j \neq 0$, and $k = -2, -1, 1, 2$. But $(1, 1)$ and $(\alpha^{ki}, \alpha^{kj})$ are linearly independent, and this implies that the above equation is impossible. Hence, $s_k \neq 0$, $k = -2, -1, 1, 2$.

Next we show that if $s_{-1} = 0$ (or $s_2 = 0$), then $s_k \neq 0$, $k = -2, 0, 1$, and s_2 (or s_{-1}) can be either zero or nonzero. It is easy to show that $s_k \neq 0$, $k = -2, 0, 1$, in the same way as above. Because $(\alpha^{-i}, \alpha^{-j})$ and $(\alpha^{2i}, \alpha^{2j})$ are linearly dependent for some i and j, there exists $\beta_1 \neq 0$, $\beta_2 \neq 0$, $\beta_1, \beta_2 \in GF(2^m)$, and some $i < j$, such that

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \beta_1 \begin{bmatrix} \alpha^{-i} \\ \alpha^{+2i} \end{bmatrix} + \beta_2 \begin{bmatrix} \alpha^{-j} \\ \alpha^{+2j} \end{bmatrix}.$$

Let $e_i = \beta_1$ and $e_j = \beta_2$. From (6.2) and (6.5) we see that the above equation becomes

$$\begin{bmatrix} s_{-1} \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = e_i \begin{bmatrix} \alpha^{-i} \\ \alpha^{+2i} \end{bmatrix} + e_j \begin{bmatrix} \alpha^{-j} \\ \alpha^{+2j} \end{bmatrix},$$

Therefore, $s_{-1} = s_2 = 0$ for some i and j.

By exactly the same argument as above, we can prove that if s_1 (or s_{-2}) $= 0$, then $s_k \neq 0$, $k = -1, 0, 2$, and s_{-2} (or s_1) can be either zero or nonzero. This completes the proof that $N \leqq 2$. Q.E.D.

Appendix B

In this Appendix we present a method for solving the quadratic equation (12) which is based on [12].

Let β be any element of $GF(2^m)$, and define

$$T_2(\beta) \triangleq \sum_{i=0}^{m-1} \beta^{2^i}. \tag{B.1}$$

$T_2(\beta)$ is known as the *trace* of β. It is either zero or one [12]. Equation (12) has solutions in $GF(2^m)$ if and only if $T_2(K) = 0$ ([10], [13]). For even m, define

$$T_4(\beta) \triangleq \sum_{i=0}^{(m-2)/2} \beta^{2^{2i}}. \tag{B.2}$$

If (12) has solutions, $T_4(K)$ is either zero or one [12].

Suppose $T_2(K) = 0$, i.e., (12) has solutions. Let x_1 be a solution of (12). Then $x_2 = 1 + x_1$ is the other solution, and we have the following results [12]:

1) m odd

$$x_1 = \sum_{j \in J} K^{2^j} = \sum_{i \in I} K^{2^i}, \tag{B.3}$$

where

$$I = \{1, 3, 5, \cdots, m-2\}, \; J = \{0, 2, 4, \cdots, m-1\}.$$

2) $m \equiv 2$ modulo 4

$$x_1 = \sum_{i=0}^{(m-6)/4} (K+K^2)^{2^{2+4i}}, \qquad \text{for } T_4(K)=0, \tag{B.4.1}$$

$$x_1 = \alpha_1 + \sum_{i=0}^{(m-6)/4} (K+K^2)^{2^{2+4i}}, \qquad \text{for } T_4(K)=1, \tag{B.4.2}$$

where α_1 is a solution of the equation $\alpha_1^2 + \alpha_1 + 1 = 0$.

3) $m \equiv 0$ modulo 4

$$x_1 = S + S^2 + K^{2^{m-1}} \left(1 + \sum_{i=0}^{(m/4)-1} K^{2^{2i+m/2}}\right), \qquad \text{for } T_4(K)=1, \tag{B.5}$$

where

$$S = \sum_{j=1}^{(m/4)-1} \sum_{i=j}^{(m/4)-1} K^{(2^{2i-1+m/2}+2^{2j-2})}.$$

For $T_4(K) = 0$, select an element β of $GF(2^m)$ such that $T_2(\beta) = 1$, compute $K_1 = \beta + \beta^2$, and solve $z^2 + z + K_1 + K = 0$ using (B.5) with K replaced by $K_1 + K$. Then $x_1 = \beta + z_1$ is a solution of (12), where z_1 is obtained from (B.5). For $m = 4, 8, 12$, (B.5) reduces to the following forms:

$$m=4, \; x_1 = K^8 + K^{12};$$

$$m=8, \; x_1 = K^{33} + K^{66} + K^{129} + K^{132};$$

$$m=12, \; x_1 = K^{2048}(1 + K^{64} + K^{256} + K^{1024}) + K^{129} + K^{258} + K^{513} + K^{1026} + K^{516} + K^{1032}.$$

Acknowledgment

We would like to thank one of the reviewers for pointing out that similar ideas have been described in U.S. Patents 4 030 067 [14] and 4 099 160 [15].

References

[1] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

[2] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Develop.*, vol. 14, pp. 395–401, July 1970.

[3] D. C. Bossen, "b-Adjacent error correction," *IBM J. Res. Develop.*, vol. 14, pp. 402–408, July 1970.

[4] D. C. Bossen, L. C. Chang, and C. L. Chen, "Measurement and generation of error correcting codes for package failures," *IEEE Trans. Comput.*, vol. C-27, pp. 201–204, Mar. 1978.

[5] S. M. Reddy, "A class of linear codes for error control in byte-per-card organized digital systems," *IEEE Trans. Comput.*, vol. C-27, pp. 455–459, May 1978.

[6] T. T. Dao, "SEC-DED nonbinary code for fault-tolerant byte-organized memory implemented with quaternary logic," *IEEE Trans. Comput.*, vol. C-30, pp. 662–666, Sept. 1981.

[7] S. Keneda and E. Fujiwara, "Single byte error correcting–double byte error detecting codes for memory systems," *IEEE Trans. Comput.*, vol. C-31, pp. 596–602, July 1982.

[8] L. A. Dunning and M. R. Varanasi, "Code constructions for error control in byte organized memory systems," *IEEE Trans. Comput.*, vol. C-32, pp. 535–542, June 1983.

[9] C. L. Chen, "Error-correcting codes with byte error-detection capability," *IEEE Trans. Comput.*, vol. C-32, pp. 615–621, July 1983.

[10] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, 1968.

[11] R. T. Chien, "Cyclic decoding procedures for BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-10, pp. 357–363, Oct. 1964.

[12] C. L. Chen, "Formulas for the solutions of quadratic equations," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 792–794, Sept. 1982.

[13] E. R. Berlekamp, H. Ramsey, and G. Solomon, "On the solution of algebraic equations over finite fields," *Inform. Contr.*, vol. 18, pp. 553–564, Oct. 1967.

[14] T. H. Howell *et al.*, "Table lookup direct decoder for double-error correcting (DEC) BCH codes using a pair of syndromes," U.S. Patent 4 030 067, June 14, 1977.

[15] H. L. Flagg, "Error location apparatus and methods," U.S. Patent 4 099 160, July 4, 1978.