DAA / AMES 94 pages

FINAL REPORT

ERROR CONTROL FOR RELIABLE DIGITAL DATA TRANSMISSION AND STORAGE SYSTEMS

NASA Grant NAG 2-202

(NASA-CR-176935) ERROR CONTROL FOR RELIABLE DIGITAL DATA TRANSMISSION AND STORAGE SYSTEMS Final Report (Illinois Inst. of Tech.) 94 p CSCL 09B

N86-29552

Unclas

G3/61 43354

Daniel J. Costello, Jr.

Robert H. Deng

Department of Electrical & Computer Engineering
Illinois Institute of Technology
Chicago, IL 60616

September 30, 1985

D.J. Costello, Jr. & R.H. Deng
NASA Grant NAG 2-202

FINAL REPORT

PART I

(Decoding of Extended Single-and-Double-Error-Correcting Reed Solomon Codes)

DECODING OF EXTENDED SINGLE-AND-DOUBLE-ERROR-CORRECTING REED SOLOMON CODES*

by

Huijie Deng and Daniel J. Costello, Jr.

Department of Electrical & Computer Engineering
Illinois Institute of Technology
Chicago, Illinois 60616

(312) 567-3404

Revised

June 1985

Submitted to the IEEE Transactions on Computers

* This work was supported by NASA under Grant NAG 2-202.

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 paper we present some special decoding techniques for extended single-and-double-error-correcting RS codes which are capable of high speed operation. These techniques are 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. Two codes are considered: 1) a $d_{min} = 4$ single-byte-error-correcting (SBEC), double-byte-error-detecting (DBED) RS code; and 2) a $d_{min} = 6$ double-byte-error-correcting (DBEC), triple-byte-error-detecting (TBED) RS code.

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×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 paper 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 bottle-neck 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 paper we investigate some special high-speed decoding techniques for extended single-and-double-byte-error-correcting RS codes. These techniques are 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 $w(\underline{s})$. Let E be the number of byte errors. In decoding a single-byte-error-correcting (SBEC), double-byte-error-detecting (DBED) RS code with 3 partiy symbols, if $w(\underline{s}) = 1$, we show that $E = 1$. If $w(\underline{s}) = 2$, then $E \geq 2$. If $w(\underline{s}) = 3$, a simple test is required to determine if $E = 1$ or $E \geq 2$. Thus decoding can be carried out in parallel, which in effect increases the decoding speed. The detailed procedure is presented in Section II. Section III contains our main result, where a double-byte-error-correcting (DBEC), triple-byte-error-detecting (TBED) RS code decoding technique is described. Double byte error correction is done by forming a quadratic equation $x^2 + x + K = 0$, the solution of which gives the two error byte 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 as two entries corresponding to each value of K, the decoding speed can be made even higher. Finally, in Section IV, we summarize our results.

II. DECODING OF A $d_{min} = 4$ SBEC-DBED CODE

In this section we present a method for decoding an extended Reed-Solomon (RS) code over GF(2^m) with minimum distance $d_{min} = 4$. This code can be used to correct any single byte error and simultaneously detect any double byte error. Thus it is called an SBEC-DBED code. Fast encoding and decoding can be acheived due to some nice features of the code described below.

The $d_{min} = 4$ Extended RS Code and Its Properties

It has been shown [11] that there exists an (n+3, n) $d_{min} = 4$ extended RS code over GF(2^m) with parity-check matrix given by

$$\underline{H} = [\ I_{3\times 3} \mid \underline{H}_1\ ], \tag{1}$$

where $\underline{I}_{3\times 3}$ is the 3×3 identity matrix,

$$\underline{H}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \cdots & \alpha^{2n-2} \end{bmatrix}, \tag{2}$$

α is a primitive element of GF(2^m), and $n \leq 2^m-1$. Because $d_{min} = 4$, the code can be used for correcting any single byte error and simultaneously detecting all double byte errors [1].

From (1) and (2) we see that the $\underline{H}$ matrix has the following important properties.

1) $\underline{H}$ is in systematic form. Hence $\underline{G}$ - the generator matrix - is also in systematic form:

$$\underline{G} = [\ \underline{H}_1^T \mid \underline{I}\ ], \tag{3}$$

where $\underline{H}_1^T$ is the transpose of $\underline{H}_1$, and where $\underline{I}$ is the n×n identity matrix. This implies that the parity-check symbols and the syndrome can be generated by the same circuit.

2) The first nonzero element of every column of $\underline{H}$ is the unit element $\alpha^0=1$. (The advantage of this property will be seen later.)

3) The number of nonzero elements in each row of $\underline{H}$ is equal.

Property 3) simplifies the implementation of the encoder and the decoder [1].

Error Correction and Error Detection

Let $\underline{v} = (v_0, v_1, \cdots, v_{n+2})$ be a code vector that is written into memory. Let $\underline{r} = (r_0, r_1, \cdots, r_{n+2})$ be the corresponding (possibly noisy) vector that is read from memory. Because of possible chip failures, $\underline{r}$ may be different from $\underline{v}$. The modulo-2 vector sum

$$\underline{e} = \underline{r} + \underline{v} = (e_0, e_1, \cdots, e_{n+2}), \tag{4}$$

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 $\underline{r}$ is read, the decoder computes the syndrome $\underline{s}$,

$$\underline{s}^T = \underline{r}\ \underline{H}^T = (\underline{v}+\underline{e})\underline{H}^T = (s_0, s_1, s_2). \tag{5}$$

Since $\underline{v}\ \underline{H}^T = \underline{0}$, the syndrome $\underline{s}$, computed from the vector $\underline{r}$, depends only on the error pattern $\underline{e}$, and not on the code vector $\underline{v}$.

a) Single byte error correction

Let $\underline{s}_s$ denote the syndrome corresponding to a single byte error. Then from (5) we have

$$\underline{s}_s = e_i\underline{h}_i = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix}, \tag{6}$$

where e_i is the error value at location i, and h_i is the $i^{\underline{th}}$ column of $\underline{H}$, $0 \leq i \leq n+2$. Note that the first nonzero element of every column of $\underline{H}$ is the unit element α^0, and $e_i \alpha^0 = e_i$. Therefore the error value e_i is given directly by the first nonzero element of the syndrome.

The problem of locating the error is reduced to finding a column $\underline{h}_i$ of $\underline{H}$ which satisfies (6) (see Chien [12]). This can be done in the following way. Check the elements of the syndrome $\underline{s}$ to see

1) if $s_0 \neq 0$, $s_1 = s_2 = 0$, then $i = 0$,
2) if $s_1 \neq 0$, $s_0 = s_2 = 0$, then $i = 1$,
3) if $s_2 \neq 0$, $s_0 = s_1 = 0$, then $i = 2$.

Otherwise, since

$$\underline{s}_s = e_i \underline{h}_i = e_i \begin{bmatrix} 1 \\ \alpha^{i-3} \\ \alpha^{2(i-3)} \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix} , \quad \text{for } 3 \leq i \leq n+2,$$

we have

$$\alpha^{i-3} = \frac{s_1}{s_0} = \frac{s_2}{s_1} , \tag{7}$$

and i gives the error location. Define

$$u \triangleq s_1^2 + s_0 s_2 , \tag{8}$$

and note that (7) is equivalent to

$$u = 0 \quad \text{for} \quad s_j \neq 0 \quad , \quad j \varepsilon \{0, 1, 2\}. \tag{9}$$

b) Double byte error detection

Let $\underline{s}_d$ denote the syndrome corresponding to a double byte error. Because the code is SBEC and DBED, it follows that [1]

$$\underline{s}_s \neq \underline{s}_d , \tag{10}$$

for any single and double byte errors. Double byte error detection can be done in the following way. If

$$s_{i_1} = 0,\ s_{i_2} \neq 0,\ s_{i_3} \neq 0, \quad \text{where} \quad i_1, i_2, i_3 \varepsilon \{0, 1, 2\}, \tag{11}$$

or if

$$s_i \neq 0, \quad \text{for} \quad i = 0, 1, 2, \quad \text{and} \quad \frac{s_1}{s_0} \neq \frac{s_2}{s_1}, \tag{12}$$

then two or more byte errors are detected. Note that (12) implies that $u = s_1^2 + s_0 s_2 \neq 0$.

Summarizing the above discussion, we have the following Decoding Scheme for the SBEC-DBED Code (see Fig. 1):

1) If $w(\underline{s}) = 0$, no errors are detected. If $w(\underline{s}) = 1$, $E = 1$, and error correction is done in step 2). If $w(\underline{s}) = 2$, $E \geq 2$, and errors are detected. If $w(\underline{s}) = 3$, $E \geq 1$, and decoding proceeds in step 3).
2) If $s_j \neq 0$, $j \varepsilon \{0,1,2\}$, then $e_i = s_j$ and $i = j$.
3) Set $u = s_1^2 + s_0 s_2$. If $u = 0$, $E = 1$, and calculating $\alpha^{i-3} \triangleq s_1/s_0$ gives the error location i. Set the error value $e_i = s_0$. If $u \neq 0$, $E \geq 2$, and errors are detected.

Figure 2 is a block diagram of the SBEC-DBED decoder.

III. DECODING OF A $d_{min} = 6$ DBEC-TBED CODE

In this section we first present a special high speed decoding technique for the DBEC-TBED RS code with $d_{min} = 6$. Then a slightly modified technique is applied to decoding the extended RS code with two extra information symbols.

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

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

$$g(x) = \sum_{i=-2}^{2} (x+\alpha^i), \tag{13}$$

where α is a primitive element of $GF(2^m)$. The parity-check matrix, $\underline{H}_2$, of the code specified by (13) can be written as

$$\underline{H}_2 = \begin{bmatrix} 1 & \alpha^{-2} & (\alpha^{-2})^2 & \cdots & (\alpha^{-2})^{n-1} \\ 1 & \alpha^{-1} & (\alpha^{-1})^2 & \cdots & (\alpha^{-1})^{n-1} \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & (\alpha)^2 & \cdots & (\alpha)^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-1} \end{bmatrix}, \qquad (14)$$

where $n \leq 2^m-1$. 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].

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

$$\underline{s}^T = \underline{r}\ \underline{H}_2^T = (\underline{v}+\underline{e})\underline{H}_2^T = \underline{e}\ \underline{H}_2^T = (s_{-2}, s_{-1}, s_0, s_1, s_2). \qquad (15)$$

The syndrome corresponding to a single byte error is

$$s_{-2} = e_i\alpha^{-2i} \quad , \qquad (16.1)$$

$$s_{-1} = e_i\alpha^{-i} \quad , \qquad (16.2)$$

$$s_0 = e_i \quad , \qquad (16.3)$$

$$s_1 = e_i\alpha^i \quad , \qquad (16.4)$$

$$s_2 = e_i\alpha^{2i} \quad , \qquad (16.5)$$

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

$$s_{-2} = e_i\alpha^{-2i} + e_j\alpha^{-2j} , \tag{17.1}$$

$$s_{-1} = e_i\alpha^{-i} + e_j\alpha^{-j} , \tag{17.2}$$

$$s_0 = e_i + e_j , \tag{17.3}$$

$$s_1 = e_i\alpha^{i} + e_j\alpha^{j} , \tag{17.4}$$

$$s_2 = e_i{}^{2i} + e_j\alpha^{2j} , \tag{17.5}$$

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

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

Property 1. Let $\underline{s}_d = (s_{-2}, s_{-1}, s_0, s_1, s_2)$ be the syndrome corresponding to a double byte error. Let N denote the number of zero elements in $\underline{s}_d$. Then

$$N \leq 2, \tag{18}$$

and the 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 $\underline{s}_d = (s_{-2}, s_{-1}, s_0, s_1, s_2)^T$. Then

$$s_2 s_{-2} + s_0^2 \neq 0 , \tag{19.1}$$

$$s_1 s_{-2} + s_{-1} s_0 \neq 0, \tag{19.2}$$

$$s_0 s_1 + s_2 s_{-1} \neq 0, \tag{19.3}$$

for all double byte errors.

Proof: This can be obtained directly from property 1.

Decoding Using the Quadratic Equation

In this subsection we 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 (17.1) and (17.3) we have

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

From (17.2) and (17.3) we have

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

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

$$s_{-1}(\alpha^i+\alpha^j) + s_{-2}\alpha^i\alpha^j + s_0 = 0. \qquad (20)$$

In the same way, from (17.3)-(17.5), we can obtain

$$s_1(\alpha^i+\alpha^j) + s_0\alpha^i\alpha^j + s_2 = 0. \qquad (21)$$

Now define

$$\gamma_1 \triangleq s_0^2 + s_{-1}s_1, \qquad (22.1)$$

$$\gamma_2 \triangleq s_2s_{-2} + s_0^2 , \qquad (22.2)$$

$$\gamma_3 \triangleq s_1s_{-2} + s_{-1}s_0, \qquad (22.3)$$

$$\gamma_4 \triangleq s_0s_1 + s_2s_{-1}. \qquad (22.4)$$

Solving (20) and (21) for $\alpha^i+\alpha^j$ and $\alpha^i\alpha^j$, we obtain

$$b \triangleq \alpha^i+\alpha^j = \frac{\gamma_2}{\gamma_3}, \tag{23.1}$$

$$c \triangleq \alpha^i\alpha^j = \frac{\gamma_4}{\gamma_3}, \tag{23.2}$$

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

$$y^2 + by + c = 0. \tag{24}$$

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 (24) can be rewritten as

$$x^2 + x + K = 0, \tag{25}$$

by letting

$$y = xb, \tag{26}$$

where

$$K \triangleq c/b^2. \tag{27}$$

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 [10,13,14,15]. The method given in [14] is probably the best approach, and we present 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 (16.1)-(16.5) we see that

$$s_i \neq 0, \quad \text{for} \quad i = -2, -1, 0, 1, 2, \tag{28.1}$$

and

$$\frac{s_{-1}}{s_{-2}} = \frac{s_0}{s_{-1}} = \frac{s_1}{s_0} = \frac{s_2}{s_1} = \alpha^i. \tag{28.2}$$

Note that (28.2) 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 (16.3) and (16.4) we have

$$\alpha^i = \frac{s_1}{s_0} , \qquad (29.1)$$

$$e_i = s_0 , \qquad (29.2)$$

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 (22.2)-(22.4) we know that $\gamma_2 \neq 0$, $\gamma_3 \neq 0$, and $\gamma_4 \neq 0$. Therefore b and c in (23.1) and (23.2) exist. Hence (24) 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 $GF(2^m)$, (17.3) and (17.4) imply that

$$e_i = \frac{\det\begin{vmatrix} s_0 & 1 \\ s_1 & \alpha^j \end{vmatrix}}{\det\begin{vmatrix} 1 & 1 \\ \alpha^i & \alpha^j \end{vmatrix}} = \frac{s_0\alpha^j + s_1}{\alpha^i + \alpha^j} , \qquad (30.1)$$

and

$$e_j = s_0 + e_i , \qquad (30.2)$$

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

Let $\underline{s}_t$ denote the syndrome corresponding to a triple byte error. Then [1]

$$\underline{s}_s \neq \underline{s}_d \neq \underline{s}_t . \qquad (31)$$

Based on (31) and properties 1 and 2, we see that if more than two elements of the syndrome $\underline{s} = (s_{-2}, s_{-1}, s_0, s_1, s_2)^T$ equal zero, but at least one of them does not equal zero; or if $\gamma_2, \gamma_3, \gamma_4$ are not all equal to zero, but at least one of them does equal zero; or if the decoding equation (24) does not have roots in $GF(2^m)$; then at least three byte errors have occurred.

Decoding Scheme for the DBEC-TBED RS Code (see Figure 3):

Read $\underline{r}$, and calculate the syndrome $\underline{s}^T = \underline{r}\,\underline{H}_2^T = (s_{-2}, s_{-1}, s_0, s_1, s_2)$. Let $w(\underline{\gamma}')$ and $w(\underline{\gamma}'')$ denote the Hamming weights of $\underline{\gamma}' \triangleq (\gamma_1, \gamma_3, \gamma_4)$ and $\underline{\gamma}'' \triangleq (\gamma_2, \gamma_3, \gamma_4)$, respectively.

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

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

3) Compute $\underline{\gamma}''$. If $w(\underline{\gamma}'') = 3$, compute K and $T_2(K)$. If $T_2(K) = 0$, $E = 2$, and we must solve (25) 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 $w(\underline{\gamma}'') \neq 3$, or $T_2(K) = 1$, $E \geq 3$ errors are detected.

Figure 4 is a block diagram of the DBEC-TBED decoder.

Decoding of the Extended Code

The parity-check matrix $\underline{H}_2$ given in (14) can be extended to form a new parity-check matrix given by

$$\underline{H}_3 = \begin{bmatrix} & 1 & 0 \\ & 0 & 0 \\ \underline{H}_2 & 0 & 0 \\ & 0 & 0 \\ & 0 & 1 \end{bmatrix} . \qquad (32)$$

The code specified by $\underline{H}_3$ is an (n+2, n-3) $d_{min} = 6$ extended RS code, where $n \leq 2^m-1$ [16,17,18]. If the errors are confined to location 0 through n-1, all the previous results apply.

Now assume that errors occur at location n or n+1. Then we obtain the following results.

1) If $\quad s_{-2} \neq s_{-1} = s_0 = s_1 = s_2 = 0, \qquad (33)$

then a single byte error occurred with error value $e_n = s_{-2}$ at location n.

2) If $\quad s_2 \neq s_{-2} = s_{-1} = s_0 = s_1 = 0, \qquad (34)$

then a single byte error occurred with error value $e_{n+1} = s_2$ at location n+1.

3) If $\quad s_i \neq 0, \quad$ for $\quad i = -1, 0, 1, 2, \qquad (35.1)$

and

$$\frac{s_{-2}}{s_{-1}} \neq \frac{s_{-1}}{s_0} = \frac{s_0}{s_1} = \frac{s_1}{s_2} , \qquad (35.2)$$

then a double byte error occurred with error values $e_i = s_0$ and $e_n = s_{-2} + s_0\alpha^{-2i}$ at locations i and n, respectively, where i is obtained from $\alpha^i = \frac{s_1}{s_0}$. Note that (35.2) implies that $\gamma_1 = 0$, $\gamma_3 \neq 0$, and $\gamma_4 = 0$.

4) If $\quad s_i \neq 0, \quad$ for $\quad i = -2, -1, 0, 1, \qquad (36.1)$

and

$$\frac{s_{-1}}{s_{-2}} = \frac{s_0}{s_{-1}} = \frac{s_1}{s_0} \neq \frac{s_2}{s_1} , \qquad (36.2)$$

i.e., $\gamma_1 = 0$, $\gamma_3 = 0$, and $\gamma_4 \neq 0$, then a double byte error occurred with error values $e_i = s_0$ and $e_{n+1} = s_2 + s_0\alpha^{2i}$ at locations i and n+1, respectively, where i is obtained from $\alpha^i = \frac{s_1}{s_0}$.

5) If $s_{-2} \neq 0$, $s_2 \neq 0$, and $s_{-1} = s_0 = s_1 = 0$, (37)

then a double byte error occurred with error values $e_n = s_{-2}$ and $e_{n+1} = s_2$ at locations n and n+1, respectively.

Now we combine the discussion in this subsection with that of the previous subsection to obtain the following.

Decoding Scheme for the Extended DBEC-TBED Code (see Figure 5):

From the vector $\underline{r}$, compute the syndrome $\underline{s}^T = \underline{r}\,\underline{H}_3^T = (s_{-2}, s_{-1}, s_0, s_1, s_2)$. Again let $w(\underline{s})$, $w(\underline{\gamma}')$, and $w(\underline{\gamma}'')$ denote the Hamming weights of $\underline{s} = (s_{-2}, s_{-1}, s_0, s_1, s_2)^T$, $\underline{\gamma}' = (\gamma_1, \gamma_3, \gamma_4)$, and $\underline{\gamma}'' = (\gamma_2, \gamma_3, \gamma_4)$ respectively.

1) If $w(\underline{s}) = 0$, no errors are detected. If $w(\underline{s}) = 1$, $E \geq 1$, and decoding proceeds in step 2). If $w(\underline{s}) = 2$, $E \geq 2$, and decoding proceeds in step 3). If $w(\underline{s}) = 3$, $E \geq 2$, and decoding proceeds in step 6). If $w(\underline{s}) = 4$, $E \geq 2$, and decoding proceeds in step 5). If $w(\underline{s}) = 5$, $E \geq 1$, and decoding proceeds in step 4).

2) If $s_{-2} \neq 0$, $E = 1$, and a single byte error is corrected with error value $e_n = s_{-2}$ at location n. If $s_2 \neq 0$, $E = 1$, and a single byte error is corrected with error value $e_{n+1} = s_2$ at location n+1. Otherwise, $E \geq 3$ errors are detected.

3) If $s_{-2} \neq 0$ and $s_2 \neq 0$, $E = 2$, and a double byte error is corrected with error values $e_n = s_{-2}$ and $e_{n+1} = s_2$ at locations n and n+1, respectively. Otherwise, $E \geq 3$ errors are detected.

4) Compute $\underline{\gamma}'$. If $w(\underline{\gamma}') = 0$, $E = 1$, and computing $\alpha^i = \frac{s_1}{s_0}$ gives the error location i. Set the error value $e_i = s_0$. If $w(\underline{\gamma}') \neq 0$, go to step 5).

5) Compute $\underline{\gamma}'$. If $w(\underline{\gamma}') = 1$, then:

(i) If $\gamma_3 \neq 0$, $E = 2$, and a double byte error is corrected with error values $e_i = s_0$ and $e_n = s_{-2} + s_0\alpha^{-2i}$ at locations i and n, respectively, where i is given by $\alpha^i = \frac{s_1}{s_0}$.

(ii) If $\gamma_4 \neq 0$, $E = 2$ and a double byte error is corrected with error values $e_i = s_0$ and $e_{n+1} = s_2 + s_0\alpha^{2i}$ at locations i and n+1, respectively, where $\alpha^i = \frac{s_1}{s_0}$.

(iii) If $\gamma_1 \neq 0$, $E \geq 3$ errors are detected.

If $w(\underline{\gamma}') \neq 1$, $E \geq 2$ errors are detected, and decoding proceeds in step 6).

6) Compute $\underline{\gamma}''$. If $w(\underline{\gamma}'') = 3$, compute K and $T_2(K)$. If $T_2(K) = 0$, $E = 2$ and we must solve (25) to find the roots α^i and α^j and then correct a double byte error with error values $e_i = (s_0\alpha^j + s_1)/(\alpha^i+\alpha^j)$, $e_j = s_0 + e_i$ at locations i and j, respectively. If $w(\underline{\gamma}'') \neq 3$, or $T_2(K) = 1$, $E \geq 3$ errors are detected.

IV. CONCLUSIONS

We have presented new decoding techniques for two byte oriented RS codes. These decoding techniques are based directly on the syndrome, and do 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 correction and detection in byte-organized computer memory ssytems such as LSI and VLSI chips.

The $d_{min} = 4$ code is capable of single-byte-error-correction (SBEC) and double-byte-error-detection (DBED) and can be extended to include three addi-

V. REFERENCES

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

[2] M.Y. Hsiao, "A Class of Optimal Minimum Odd-Weight-Column SEC-DED Codes", *IBM J. Res. Dev.*, 14, pp. 395-401, July 1970.

[3] D.C. Bossen, "b-Adjacent Error Correction", *IBM J. Res. Develop.*, 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.*, C-27, pp. 201-204, March 1978.

[5] S.M. Reddy, "A Class of Linear Codes for Error Control in Byte-per-Card Organized Digital Systems", *IEEE Trans. Comput.*, 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.*, 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.*, 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.*, C-32, pp. 535-542, June 1983.

[9] C.L. Chen, "Error-Correcting Codes with Byte Error-Detection Capability", *IEEE Trans. Comput.*, C-32, pp. 615-621, July 1983.

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

[11] F.J. MacWilliams and N.J.A. Sloane, *Theory of Error-Correcting Codes*, North Holland, Amsterdam, 1978.

[12] R.T. Chien, "Block-Coding Techniques for Reliable Data Transmission", *IEEE Trans. on Comm. Tech.*, COM-19 (Part II), pp. 743-751, Oct. 1971.

[13] R.T. Chien, "Cyclic Decoding Procedures for BCH Codes", *IEEE Trans. Inform. Theory*, IT-10, pp. 357-363, Oct. 1964.

[14] C.L. Chen, "Formulas for the Solutions of Quadratic Equations", *IEEE Trans. Inform. Theory*, IT-28, pp. 792-794, Sept. 1982.

[15] E.R. Berlekamp, H. Ramsey, and G. Solomon, "On the Solution of Algebraic Equations Over Finite Fields", *Inform. Contr.*, 18, pp. 553-564, Oct. 1967.

[16] T. Kasami, S. Lin, and W.W. Peterson, "Some Results on Weight Distributions of BCH Codes", *IEEE Trans. Inform. Theory*, IT-12, p. 274, April 1966.

[17] T. Kasami, S. Lin, and W.W. Peterson, "Some Results on Cyclic Codes Which are Invariant Under the Affine Group", Scientific Report AFCRL-66-662, Air Force Cambridge Research Labs., Bedford, MA, 1966.

[18] J.K. Wolf, "Adding Two Information Symbols to Certain Nonbinary BCH Codes and Some Applications", Bell Sys. Tech. J., 48, pp. 2405-2424, 1969.

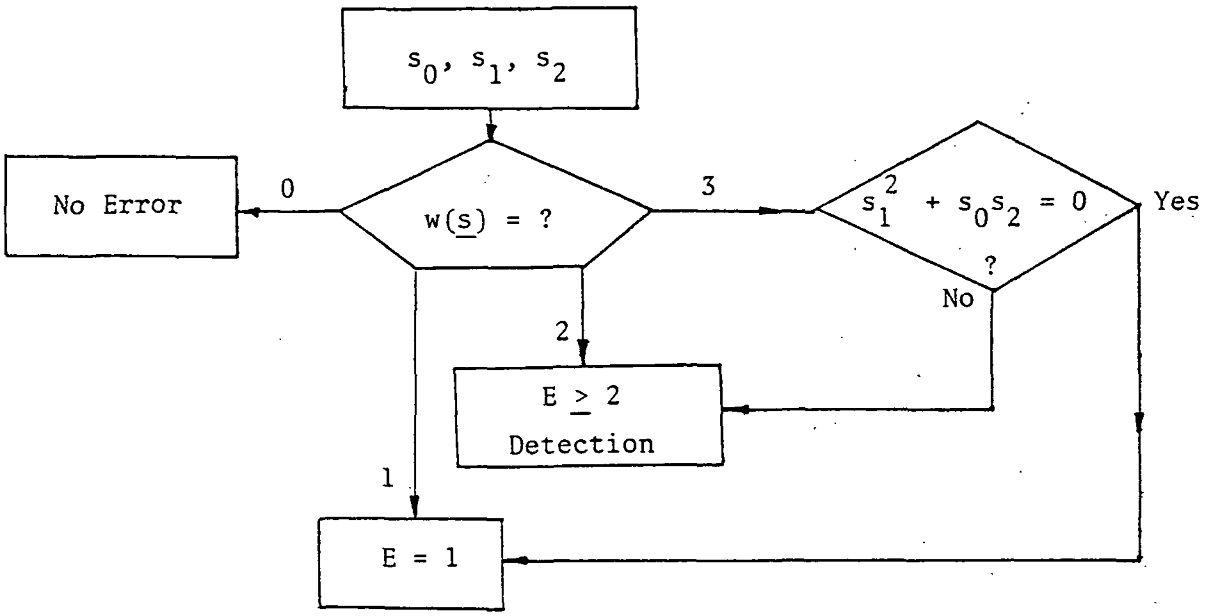


Figure 1. SBEC-DBED decoder error location calculator

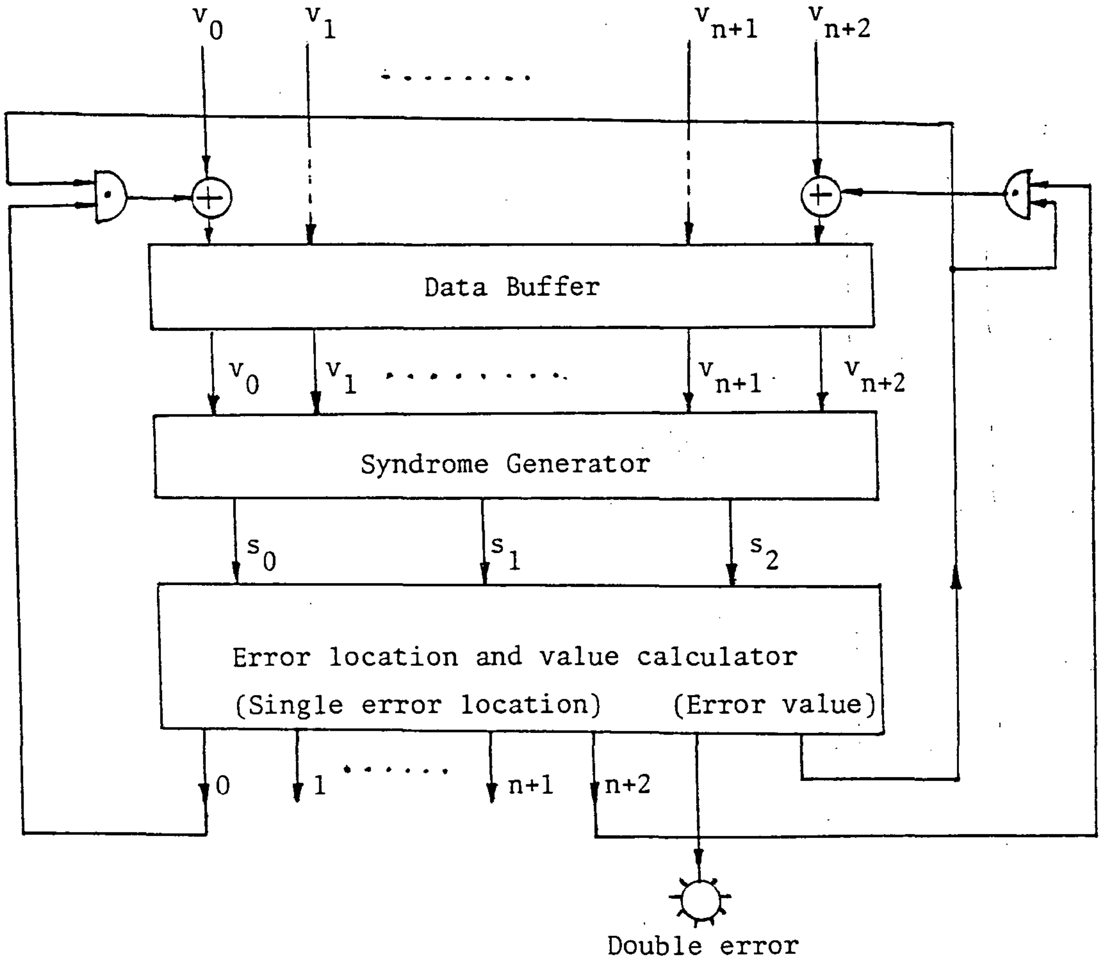


Figure 2. Block diagram of a SBEC-DBED decoder

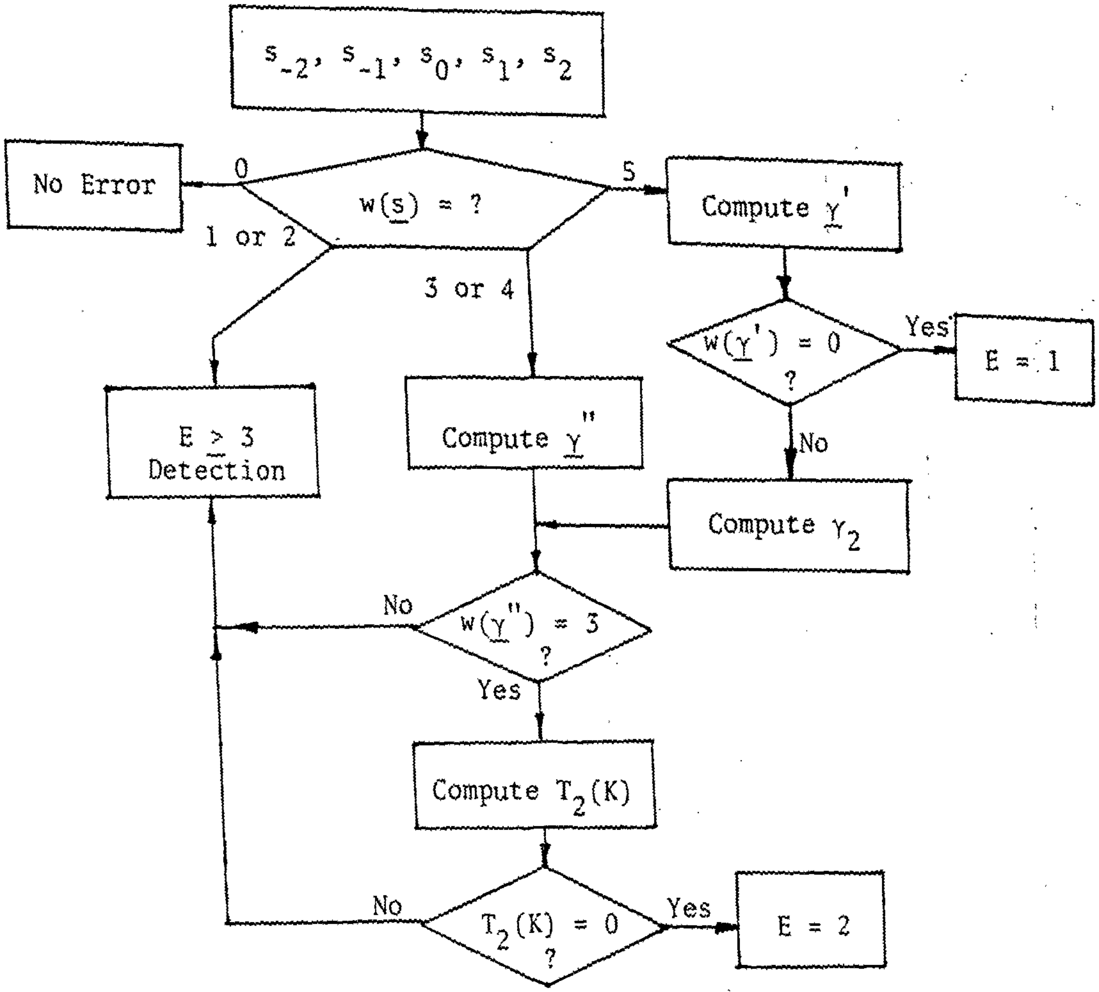

$s_{-2}, s_{-1}, s_0, s_1, s_2$
No Error
0
$w(\underline{s}) = ?$
5
Compute $\underline{\gamma}'$
1 or 2
3 or 4
$w(\underline{\gamma}') = 0$?
Yes
E = 1
$E \geq 3$ Detection
Compute $\underline{\gamma}''$
No
Compute γ_2
No
$w(\underline{\gamma}'') = 3$?
Yes
Compute $T_2(K)$
No
$T_2(K) = 0$?
Yes
E = 2

Figure 3. DBEC-TBED decoder error location calculator

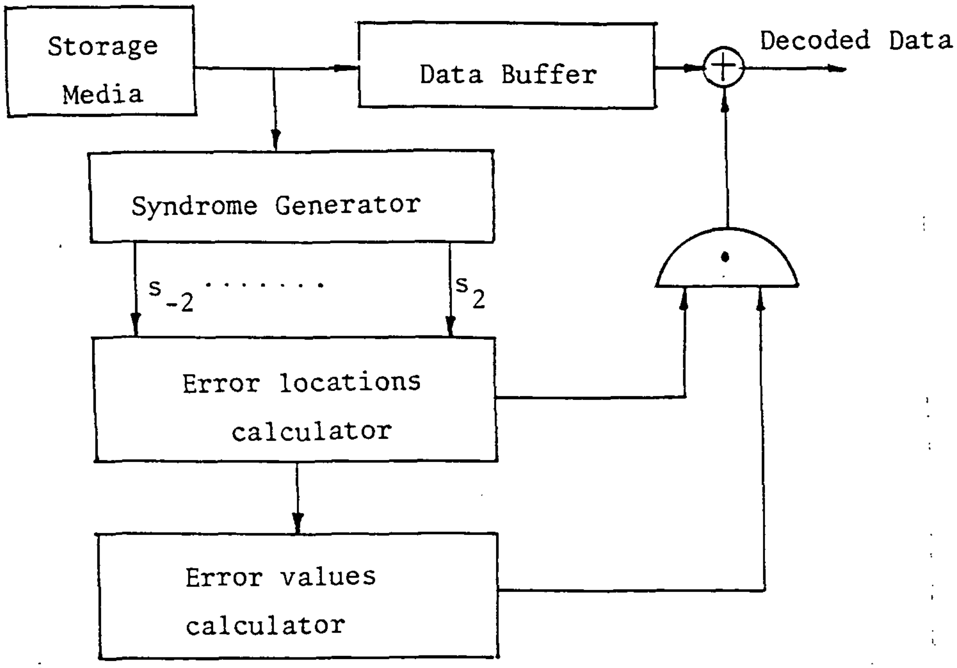


Figure 4. Block diagram of a DBEC-TBED decoder

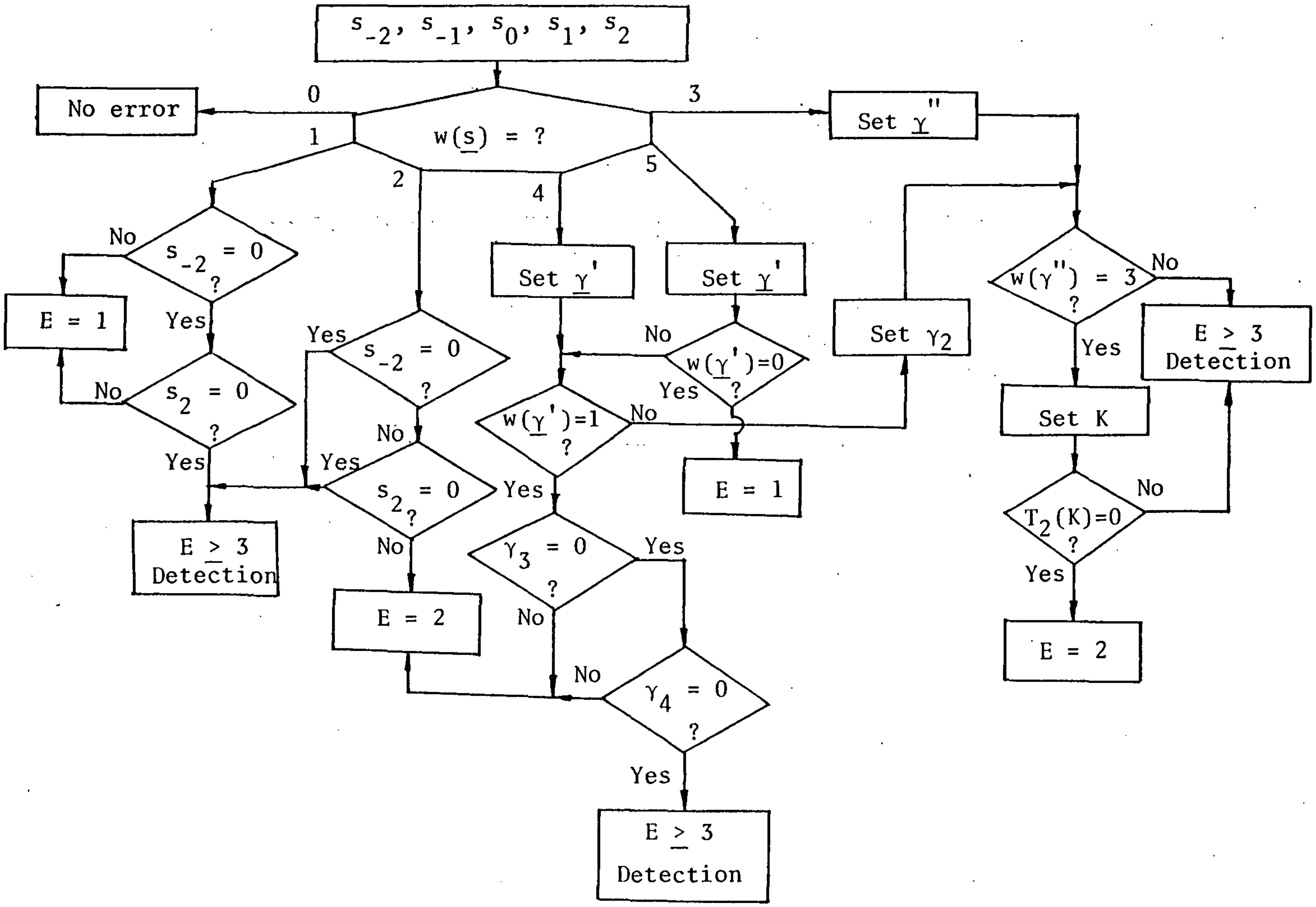


Figure 5. Extended DBEC-TBED decoder error locations calculator

APPENDIX A

Proof of Property 1: It can easily been 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) $(\alpha^{-i}, \alpha^{-j})$, $(\alpha^{2i}, \alpha^{2l})$;

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 (17.1)-(17.5), 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 $\beta_1 \neq 0$, $\beta_2 \neq 0$, $\beta_1, \beta_2 \varepsilon GF(2^m)$, 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 (17.2) and (17.5) 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 (25) which is based on [14].

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 [14]. Equation (25) has solutions in $GF(2^m)$ if and only if $T_2(K) = 0$ [10,15]. For even m, define

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

If (25) has solutions, $T_4(K)$ is either zero or one [14].

Suppose $T_2(K) = 0$, i.e., (25) has solutions. Let x_1 be a solution of (25). Then $x_2 = 1 + x_1$ is the other solution, and we have the following results [14]:

1) m odd

$$x_1 = \sum_{j\varepsilon J} K^{2^j} = \sum_{i\varepsilon I} K^{2^i}$$

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

2) $m \equiv 2$ modulo 4

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

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

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

3) $m \equiv 0$ modulo 4

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

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 (25), where z_1 is obtained from (B.5). For m = 4, 8, 12, (B.5) reduces to the following forms:

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

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

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

D.J. Costello, Jr. & R.H. Deng

NASA Grant NAG 2-202

FINAL REPORT

PART II

(Optimal Schortened BCH Codes for Computer Memory Systems)

OPTIMAL SHORTENED BCH CODES FOR COMPUTER MEMORY SYSTEMS

Abstract: This paper presents a method for constructing optimal and nearly optimal shortened BCH codes which are suitable for applications to computer memory systems. The optimal codes we found minimize the hardware required for implementation compared with all the other shortened BCH codes.

I. INTRODUCTION

Error-correcting codes are widely used to improve the system-level reliability of computer main storage or control storage. The IBM system 7030, built in 1961, was the first IBM computer system to use a single-error-correcting and double-error-detecting (SEC-DED) Hamming code with minimum distance $d_{min}=4$ for its core memory [1]. However, core memories are very reliable, especially since the technology has advanced to a mature state. In the 1970's, semiconductor memories were used to replace core memories. Semiconductor memories are faster than core memories in speed; however, they are less reliable than core memories due to their high density per chip and their exposure to radiation, which induces soft failures (errors). As a result, the use of error-correcting codes for improving semiconductor memory reliability became a standard design feature. The improvement in reliability is especially evident when the memory system is organized on 1-bit-per-card basis (bit-oriented memory). With this organization, most error patterns (or multiple-bit failures caused by a malfunction) on each card appear as if they were single errors.

The most commonly used error-correcting codes have been the minimum odd-weight-column SEC-DED codes first constructed by Hsiao [2]. Hsiao's construction is optimal in the sense that its parity-check matrix, denoted by $\underline{H}_o$, satifies the following requirements.

1. The total number of 1's in the $\underline{H}_0$ matrix is a minimum.
2. The number of 1's in each row of $\underline{H}_0$ is equal (or at least close) to the average number (i.e., the total number of 1's in $\underline{H}_0$ divided by the number of rows in $\underline{H}_0$).

Let N_i be the number of 1's in the ith row of the parity-check matrix. Let $L_i^{(p)}$ be the number of logic levels required to generate the ith parity-check bit with b-input modulo-2 adders (or b-input XOR gates). Let $L_i^{(s)}$ be the number of logic levels required to generate the ith syndrome bit with b-input modulo-2 adders. Then we have [1,2]

$$L_i^{(p)} = \lceil \log_b (N_i-1) \rceil \tag{1}$$

$$L_i^{(s)} = \lceil \log_b N_i \rceil \tag{2}$$

where $\lceil X \rceil$ denotes the smallest integer greater than or equal to X. From (1) and (2), we see that the two requirements mentioned above minimize the number of logic levels required to generate the parity and syndrome bits, and hence hardware required for implementation of the code.

In practice, some computer memory systems require higher reliability than that the SEC-DED codes can provide. These codes should have minimum distance $d_{min} > 4$, and should also satisfy the above two requirements, so that fast encoding and decoding, which are the most critical on-line processes in the memory operations,can be achieved. Unfortunately, general methods for constructing such codes are still unknown. In this paper we present an algorithm for shortening the BCH codes [1]. In section II, we give some lower bounds on the total number of 1's in a parity-check matrix $\underline{H}$, and on the average number of 1's in each row of $\underline{H}$. In Section III, we present our algorithm, along with the shortened BCH codes we found. Finally, we summarize our results in Section IV.

II. LOWER BOUNDS ON THE NUMBER OF 1'S IN THE PARITY-CHECK MATRIX

Throughout this paper, we only consider systematic linear codes, because they are the most commonly used codes in practice. The code construction is best described in terms of the parity-check matrix $\underline{H}$. An (n,k) systematic linear code is uniquely specified by its parity-check matrix , which is given by

$$\underline{H} = [\ \underline{I}_{n-k} \,\vdots\, \underline{Q}\ ], \tag{3}$$

where $\underline{I}_{n-k}$ is the (n-k)x(n-k) identity matrix,

$$\underline{Q} = [\ \underline{q}_1, \underline{q}_2, \ldots, \underline{q}_k\ ] \tag{4}$$

is an (n-k)xk matrix, and $\underline{q}_i$, $1 \le i \le k$, is the $\underline{ith}$ column of $\underline{Q}$.

$\underline{\text{TH}}$: Let T and A denote the total number of 1's in $\underline{H}$ and the average number of 1's in each row of $\underline{H}$, respectively. Let m be the unique positive integer such that

$$\sum_{i=d_{min}-1}^{\ell} \binom{n-k}{i} + m = k, \tag{5}$$

where ℓ satisfies

$$\sum_{i=d_{min}-1}^{\ell} \binom{n-k}{i} < k \le \sum_{i=d_{min}-1}^{\ell+1} \binom{n-k}{i}, \tag{6}$$

and where d_{min} is the minimum distance of the code. Then T and A can be lower bounded by

$$T \geq k(\, d_{min} - 1\,) + (\, n-k\,), \text{ if } k \leq \binom{n-k}{d_{min}-1},$$

$$\geq \sum_{i=d_{min}-1}^{\ell} i \binom{n-k}{i} + (\,\ell+1\,)m + (\,n-k),$$

$$\text{if} \sum_{i=d_{min}-1}^{\ell} \binom{n-k}{i} < k \leq \sum_{i=d_{min}-1}^{\ell+1} \binom{n-k}{i}. \qquad (7)$$

$$A \geq \frac{k(\, d_{min}-1\,)}{(n-k)} + 1, \text{ if } k \leq \binom{n-k}{d_{min}-1}.$$

$$\geq \frac{\sum_{i=d_{min}-1}^{\ell} i \binom{n-k}{i} + (\ell+1)m}{(n-k)} + 1,$$

$$\text{if} \sum_{i=d_{min}-1}^{\ell} \binom{n-k}{i} < k \leq \sum_{i=d_{min}-1}^{\ell+1} \binom{n-k}{i}. \qquad (8)$$

Proof: Since the code has minimum distance d_{min}, the number of 1's in each column of $\underline{Q}$ must be at least $d_{min}-1$. If $k \leq \binom{n-k}{d_{min}-1}$, the total number of 1's in $\underline{Q}$ is at least $k(\, d_{min}-1\,)$. Adding the (n-k) 1's in the identity part of (3) to $k(\, d_{min}-1\,)$, we obtain the first part of (7). The second part can be proved in a similar way. (8) follows from the fact that $\underline{H}$ has (n-k) rows.

An (n,k) systematic linear code is said to be optimal, for a given d_{min}, if the total number of 1's in its parity-check matrix meet the lower bound of (7), and the maximum number of 1's in each row is $\lceil A \rceil$, where A meets the lower bound of (8). As mentioned in section I, no general method is known for constructing optimal codes with $d_{min} > 4$. Instead, we seek a method for shortening the BCH codes in the hope that shortened BCH codes can satisfy the two requirements of section I.

Let

$$\underline{H}_1 = [\ \underline{I}_{n-k} \ \vdots\ Q_1\ ] \tag{9}$$

be the parity-check matrix of an (n,k) systematic BCH code with minimum distance d_{min}, where

$$\underline{Q}_1 = [\ \underline{q}_1^{(1)}, \underline{q}_2^{(1)}, \ldots, \underline{q}_k^{(1)}\ ] \tag{10}$$

is an (n-k)xk matrix. The shortened BCH code is an (n-w,k-w) code specified by its parity-check matrix

$$\underline{H}_{w+1} = [\ \underline{I}_{n-k} \ \vdots\ \underline{Q}_{w+1}\ ], \tag{11}$$

where

$$\underline{Q}_{w+1} = [\ \underline{q}_1^{(w+1)}, \underline{q}_2^{(w+1)}, \ldots, \underline{q}_{k-w}^{(w+1)}\ ] \tag{12}$$

is an (n-k)x(k-w) matrix which is obtained by deleting w columns from $\underline{Q}_1$, where $0<w<k$. The shortened code has minimum distance at least d_{min}.

Define

$$\underline{CW}^{(1)} = (CW_1^{(1)}, CW_2^{(1)}, \ldots, CW_k^{(1)})$$

as a <u>column weight vector</u>, where $CW_i^{(1)}$ is the Hamming weight of the <u>ith</u> column of $\underline{Q}_1$. Arrange $CW_i^{(1)}$, $1 \leq i \leq k$, in such a way that

$$CW_{i_1}^{(1)} \leq CW_{i_2}^{(1)} \leq \ldots \leq CW_{i_k}^{(1)},$$

If the i_{k-w+1}<u>th</u>, i_{k-w+2}<u>th</u>,..., i_k<u>th</u> columns, i.e., the w heaviest columns, are deleted from $\underline{Q}_1$, then the total number of 1's in $\underline{H}_{w+1}$ is minimized. Let T_1 and A_1 denote the total number of 1's in $\underline{H}_{w+1}$ and the average number of 1's in each row of $\underline{H}_{w+1}$, respectively. Then we have

$$T_1 \geq \sum_{j=1}^{k-w} CW_{i_j}^{(1)} + n-k, \tag{13}$$

and

$$A_1 \geq [\sum_{j=1}^{k-w} CW_{i_j}^{(1)} \;/\; (n-k)] +1 . \tag{14}$$

A shortened (n-w, k-w) systematic BCH codes is said to be optimal for a given d_{min} if T_1 meets the lower bound of (13) and the maximum number of 1's in each row of $\underline{H}_{w+1}$ is $\lceil A_1 \rceil$, where A_1 meets the lower bound of (14).

III SHORTENED BCH CODE CONSTRUCTION ALGORITHM

$\binom{k}{w}$ shortened (n-w, k-w) systematic BCH codes can be obtained from an (n,k) systematic BCH code. To obtain the best code, the most straightforward approach is to try the $\binom{k}{w}$ possibilities, and pick up the best one. Unfortunately, even for moderate k and w, this approach is impossible due to the huge amount of computation. In the following, we present an algorithm for constructing shortened BCH codes. Our results show that some optimal and many nearly optimal shortened BCH codes can be found easily by this algorithm.

Define

$$\underline{RW}^{(1)} = (RW_1^{(1)}, RW_2^{(1)}, \ldots, RW_{n-k}^{(1)})$$

as a row weight vector, where $RW_i^{(1)}$ is the Hamming weight of the ith row of $\underline{H}_1$. The algorithm first sets w, the number of columns to be deleted from $\underline{Q}_1$, and then selects an arbitrary nonnegative real number $\mu \geq 0$.

Step 1. Calcualte $\underline{RW}^{(1)}$ and $\underline{CW}^{(1)}$.

Set

$$RW_{max}^{(1)} = \max_j \{ RW_j^{(1)}, 1 \leq j \leq n-k \}$$

and calculate

$$\lambda_j^{(1)} = \frac{RW_j^{(1)}}{RW_{max}^{(1)}} \cdot \mu \;, \; 1 \leq j \leq n-k,$$

$$\underline{\lambda}^{(1)} = \begin{bmatrix} e^{\lambda_1^{(1)}} \\ e^{\lambda_2^{(1)}} \\ \vdots \\ e^{\lambda_{n-k}^{(1)}} \end{bmatrix} ,$$

and

$$\phi_\ell^{(1)} = CW_\ell^{(1)} \cdot \underline{\underline{q}}_\ell^{(1)T} \cdot \underline{\lambda}^{(1)}, \quad 1 \leq \ell \leq k,$$

where $\underline{\underline{q}}_\ell^{(1)T}$ is the transpose of $\underline{\underline{q}}_\ell^{(1)}$.

If

$$\phi_\ell^{(1)} \geq \phi_j^{(1)} \ , \quad 1 \leq j \leq k,$$

then the ℓth column of $\underline{Q}_1$ is deleted. By deleting the ℓth column of $\underline{Q}_1$, we obtain an (n-k)x(k-1) matrix

$$\underline{Q}_2 = [\underline{q}_1^{(2)}, \underline{q}_2^{(2)}, \ldots, \underline{q}_{k-1}^{(2)} \ ]$$

and a new (n-k)x(n-1) parity-check matrix

$$\underline{H}_2 = [\underline{I}_{n-k} \ \vdots \ \underline{Q}_2].$$

In general, after i < w steps, we have an (n-k)x(n-i) parity-check matrix

$$\underline{H}_{i+1} = [\underline{I}_{n-k} \ \vdots \ \underline{Q}_{i+1}],$$

where

$$\underline{Q}_{i+1} = [\underline{q}_1^{(i+1)}, \underline{q}_2^{(i+1)}, \ldots, \underline{q}_{k-i}^{(i+1)}],$$

which is an (n-k)x(k-i) matrix obtained by deleting i columns from $\underline{Q}_1$.

Step i+1. Calculate

$$\underline{RW}^{(i+1)} = (RW_1^{(i+1)}, RW_2^{(i+1)}, \ldots, RW_{n-k}^{(i+1)}),$$

and

$$\underline{CW}^{(i+1)} = (CW_1^{(i+1)}, CW_2^{(i+1)}, \ldots, CW_{k-i}^{(i+1)}).$$

Set

$$RW_{max}^{(i+1)} = \max_j \{RW_j^{(i+1)}, 1 \leq j \leq n-k\},$$

and calulate

$$\lambda_j^{(i+1)} = \frac{RW_j^{(i+1)}}{RW_{max}^{(i+1)}} \cdot \mu, \ 1 \leq j \leq n-k,$$

$$\underline{\lambda}^{(i+1)} = \begin{bmatrix} e^{\lambda_1^{(i+1)}} \\ e^{\lambda_2^{(i+1)}} \\ \vdots \\ e^{\lambda_{n-k}^{(i+1)}} \end{bmatrix} ,$$

and

$$\phi_\ell^{(i+1)} = CW_\ell^{(i+1)} \cdot \underline{q}_\ell^{(i+1)^T} \cdot \underline{\lambda}^{(i+1)}, \ 1 \leq \ell \leq k-i.$$

If

$$\phi_\ell^{(i+1)} \geq \phi_j^{(i+1)}, \ 1 \leq j \leq k-i,$$

then the ℓth column of $\underline{Q}_{i+1}$ is deleted. Form an (n-k)x(n-i-1) parity-check matrix

$$\underline{H}_{i+2} = [\underline{I}_{n-k} \ \vdots \ \underline{Q}_{i+2}]$$

where

$$\underline{Q}_{i+2} = [\underline{q}_1^{(i+2)}, \underline{q}_2^{(i+2)}, \ldots, \underline{q}_{k-i-1}^{(i+2)}],$$

is an (n-k)x(k-i-1) matrix obtained by deleting the ℓth column from $\underline{Q}_{i+1}$.

If i+1 < w, the above procedure is continued. If i+1=w, $\underline{H}_{i+2}$ is the desired parity-check matrix of an (n-w, k-w) shortened BCH code. A block diagram of the algorithm is shown in Fig. 1.

In this algorithm, μ is a key parameter in finding the best shortened BCH codes. The larger the value of μ, the smaller the maximum row weight $RW_{max}^{(w+1)}$, but the total number of 1's in the parity-check matrix is larger. On the other hand, if μ=0, the w columns of $\underline{H}_1$ with the most 1's will be deleted, thus minimizing the total number of 1's in the parity-check matrix of the shortened BCH code. But the requirement that the number of 1's in each row be equal usually can't be satisfied. Which requirement is more important depends on the particular situation, and the value of μ can be adjusted accordingly.

Table 1 gives a list of the parameters of the shortened BCH codes found by the algorithm, where $n' = n-w$, $k' = k-w$. The (47,32) and (81,64) codes are seen to be optimal according to the definition from Section II, and the other codes are nearly optimal. Fig. 2 - Fig. 5 show the parity-check matrices of the shortened BCH codes with minimum distance d_{min}=5, 6, 7, and 8, respectively. Fig. 6 is a block diagram of the decoder. The encoder is identical to the upper part of the decoder with the input parity bits deleted.

IV CONCLUSION

In this paper we have presented a method of constructing optimal and nearly optimal shortened BCH codes. Some lower bounds on the parameters of the parity-check matrix were given. An efficient algorithm for shortening the BCH codes was obtained. Some of the codes found by the algorithm were optimal in the sense of meeting the lower bounds on the parameters of the parity-check matrix. The algorithm was computed on a VAX/780 computer. For moderate values of n and w running time was only a few seconds.

REFERENCES

[1] S. Lin, and D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, Inc., New Jersey, 1983.

[2] M.Y. Hsiao "A Class of Optimal Minimum Odd-Weight-Column SEC-DED Codes", IBM J. Res. Dev., 14, July 1970.

TABLE 1 PARAMETERS OF A LIST OF SHORTENED BCH CODES

n'	k'	w	d_{min}	Total number of 1's in $\underline{H}$ (T_1)	Lower bound on T_1	Lower bound on T	Average number of 1's per row (A_1)	Lower bound on A_1	Lower bound on A	Maximum number of 1's per row
28	16	35	5	88	86	76	7.33	7.17	6.33	8
44	32	19	5	188	187	140	15.67	15.58	11.67	17
46	32	81	5	180	180	142	12.86	12.86	10.14	14
78	64	49	5	401	394	270	28.64	28.14	19.29	30
80	64	175	5	368	367	272	23.00	22.94	17.00	24
29	16	34	6	101	101	93	7.77	7.77	7.15	9
45	32	18	6	215	213	173	16.54	16.38	13.31	18
47	32	80	6	215	215	175	14.33	14.33	11.67	15
79	64	48	6	445	439	335	29.67	29.27	22.33	31
81	64	174	6	409	409	337	24.06	24.06	19.82	25
34	16	29	7	124	120	114	6.89	6.67	6.33	8
50	32	13	7	242	238	210	13.44	13.22	11.67	17
85	64	42	7	575	571	405	27.38	27.19	19.29	29
88	64	167	7	591	586	408	24.63	24.42	17.00	26
35	16	28	8	133	131	131	7.00	6.89	6.89	8
51	32	12	8	261	257	243	13.74	13.53	12.79	16
86	64	41	8	656	656	470	29.82	29.82	21.36	32
89	64	166	8	673	663	473	26.92	26.52	18.92	28

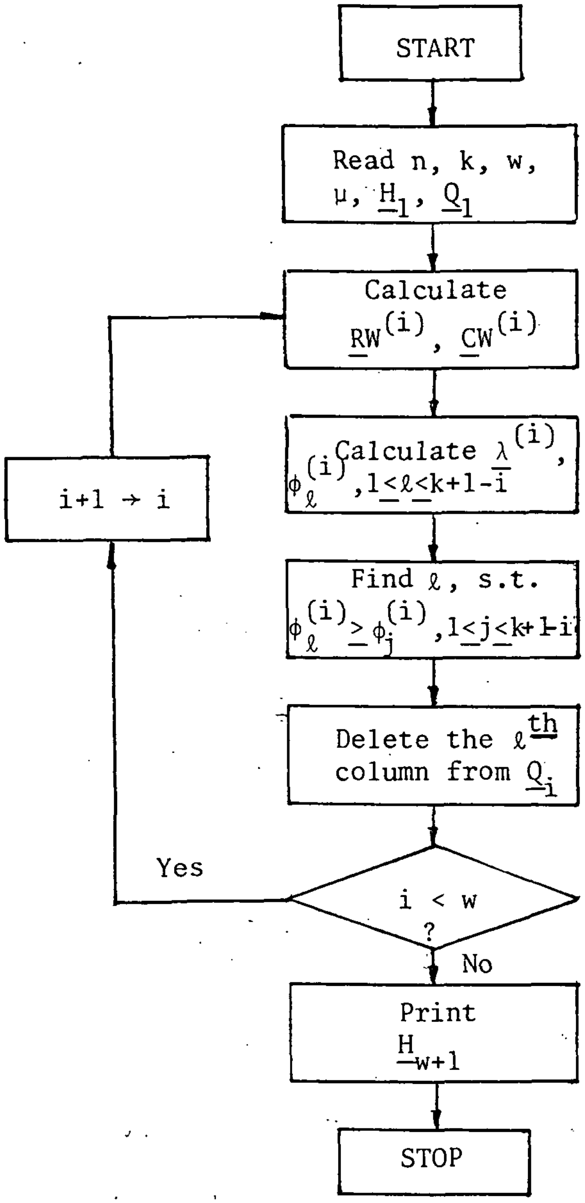

START
Read n, k, w,
μ, $\underline{H}_1$, $\underline{Q}_1$
Calculate
$\underline{RW}^{(i)}$, $\underline{CW}^{(i)}$
Calculate $\underline{\lambda}^{(i)}$,
$\phi_\ell^{(i)}$, $1 \le \ell \le k+1-i$
Find ℓ, s.t.
$\phi_\ell^{(i)} \ge \phi_j^{(i)}$, $1 \le j \le k+1-i$
Delete the ℓth
column from $\underline{Q}_i$
i < w
?
Yes
No
i+1 → i
Print
$\underline{H}_{w+1}$
STOP

Figure 1. Block diagram of the algorithm for shortening the BCH codes.

$$\underline{H} = \left[\begin{array}{ccccccc}
 & 6 & 2 & 7 & 0 & 4 & 0 \\
 & 1 & 0 & 1 & 4 & 6 & 4 \\
 & 0 & 5 & 0 & 6 & 3 & 4 \\
 & 4 & 0 & 3 & 3 & 1 & 4 \\
 & 0 & 2 & 4 & 5 & 0 & 0 \\
 & 4 & 2 & 5 & 2 & 0 & 0 \\
\underline{I}_{12} & 6 & 1 & 0 & 5 & 0 & 4 \\
 & 7 & 1 & 4 & 2 & 0 & 0 \\
 & 1 & 6 & 1 & 1 & 4 & 4 \\
 & 2 & 6 & 2 & 4 & 6 & 0 \\
 & 1 & 1 & 4 & 2 & 3 & 4 \\
 & 4 & 5 & 6 & 1 & 1 & 0
\end{array}\right]$$

Fig. 2.1 Parity-check matrix of a (28,16) $d_{min}=5$ code

$$\underline{H} = \left[\begin{array}{cccccccccccc}
 & 5 & 2 & 5 & 1 & 1 & 7 & 3 & 0 & 2 & 3 & 0 \\
 & 2 & 5 & 2 & 4 & 4 & 5 & 1 & 4 & 7 & 1 & 4 \\
 & 1 & 1 & 4 & 2 & 2 & 2 & 4 & 6 & 1 & 5 & 6 \\
 & 5 & 6 & 2 & 0 & 0 & 4 & 7 & 3 & 0 & 4 & 6 \\
 & 7 & 5 & 4 & 1 & 1 & 5 & 0 & 5 & 4 & 0 & 2 \\
 & 6 & 6 & 2 & 1 & 5 & 3 & 1 & 2 & 0 & 3 & 0 \\
\underline{I}_{12} & 3 & 3 & 1 & 0 & 6 & 7 & 4 & 5 & 0 & 0 & 4 \\
 & 1 & 7 & 5 & 4 & 3 & 1 & 4 & 2 & 0 & 1 & 2 \\
 & 5 & 5 & 2 & 7 & 0 & 1 & 5 & 1 & 6 & 2 & 4 \\
 & 2 & 4 & 5 & 3 & 4 & 2 & 6 & 4 & 7 & 1 & 2 \\
 & 4 & 0 & 6 & 4 & 7 & 4 & 4 & 2 & 5 & 6 & 4 \\
 & 2 & 3 & 2 & 2 & 3 & 6 & 2 & 1 & 4 & 6 & 2
\end{array}\right]$$

Fig. 2.2 Parity-check matrix of a (44, 32) $d_{min}=5$ code

$$\underline{H} = \left[\begin{array}{cccccccccccc}
 & 7 & 2 & 2 & 2 & 5 & 1 & 4 & 1 & 1 & 0 & 0 \\
 & 4 & 5 & 1 & 1 & 3 & 6 & 2 & 1 & 0 & 4 & 0 \\
 & 1 & 2 & 4 & 7 & 1 & 6 & 1 & 0 & 4 & 2 & 0 \\
 & 0 & 5 & 2 & 0 & 0 & 4 & 0 & 5 & 7 & 1 & 0 \\
 & 6 & 2 & 5 & 0 & 0 & 2 & 4 & 2 & 3 & 4 & 4 \\
 & 0 & 1 & 0 & 4 & 0 & 3 & 2 & 1 & 5 & 6 & 0 \\
 & 2 & 0 & 6 & 0 & 1 & 1 & 1 & 0 & 3 & 7 & 2 \\
\underline{I}_{14} & 0 & 0 & 1 & 2 & 4 & 5 & 4 & 4 & 5 & 7 & 4 \\
 & 2 & 0 & 0 & 5 & 6 & 0 & 2 & 2 & 2 & 7 & 4 \\
 & 6 & 0 & 2 & 2 & 2 & 0 & 5 & 1 & 0 & 3 & 6 \\
 & 3 & 0 & 1 & 1 & 1 & 2 & 2 & 5 & 0 & 1 & 6 \\
 & 1 & 6 & 2 & 6 & 0 & 5 & 1 & 2 & 4 & 0 & 6 \\
 & 1 & 5 & 1 & 1 & 5 & 1 & 0 & 4 & 6 & 0 & 2 \\
 & 4 & 4 & 4 & 7 & 2 & 6 & 0 & 2 & 2 & 0 & 2
\end{array}\right]$$

Fig. 2.3 Parity-check matrix of a (46,32) d_{min}=5 code

$$\underline{H} = \left[\begin{matrix} \underline{I}_{14} \end{matrix} \quad \begin{matrix}
1 & 3 & 0 & 5 & 4 & 7 & 1 & 5 & 5 & 2 & 2 & 5 & 2 & 4 & 6 & 0 & 2 & 1 & 4 & 0 & 2 & 4 \\
0 & 6 & 4 & 2 & 2 & 0 & 5 & 4 & 6 & 6 & 3 & 5 & 7 & 6 & 1 & 0 & 7 & 1 & 2 & 0 & 1 & 4 \\
4 & 0 & 2 & 4 & 5 & 2 & 3 & 3 & 2 & 1 & 3 & 2 & 5 & 3 & 0 & 4 & 5 & 5 & 1 & 0 & 3 & 0 \\
5 & 0 & 1 & 2 & 2 & 5 & 1 & 6 & 0 & 1 & 5 & 7 & 2 & 5 & 0 & 2 & 6 & 6 & 4 & 4 & 2 & 4 \\
7 & 7 & 0 & 1 & 5 & 4 & 5 & 0 & 0 & 2 & 4 & 1 & 7 & 6 & 2 & 1 & 1 & 2 & 6 & 2 & 0 & 4 \\
2 & 4 & 4 & 1 & 2 & 0 & 3 & 2 & 0 & 2 & 0 & 6 & 5 & 3 & 5 & 0 & 6 & 4 & 7 & 1 & 1 & 4 \\
7 & 1 & 2 & 0 & 1 & 7 & 0 & 0 & 0 & 3 & 2 & 7 & 0 & 1 & 4 & 4 & 1 & 3 & 7 & 4 & 5 & 0 \\
4 & 4 & 5 & 1 & 0 & 2 & 4 & 5 & 5 & 1 & 5 & 0 & 4 & 0 & 6 & 2 & 4 & 5 & 7 & 6 & 2 & 4 \\
6 & 1 & 2 & 1 & 0 & 2 & 3 & 4 & 3 & 7 & 4 & 2 & 0 & 4 & 1 & 1 & 0 & 3 & 3 & 7 & 1 & 4 \\
2 & 3 & 5 & 0 & 0 & 7 & 0 & 7 & 4 & 4 & 4 & 6 & 2 & 6 & 2 & 4 & 6 & 0 & 1 & 7 & 4 & 0 \\
6 & 1 & 6 & 5 & 0 & 2 & 4 & 6 & 2 & 3 & 2 & 1 & 1 & 3 & 1 & 2 & 3 & 0 & 0 & 7 & 6 & 0 \\
5 & 0 & 7 & 6 & 4 & 1 & 2 & 3 & 4 & 0 & 5 & 2 & 4 & 5 & 4 & 5 & 1 & 4 & 0 & 3 & 6 & 0 \\
3 & 4 & 3 & 6 & 2 & 4 & 5 & 2 & 7 & 1 & 2 & 5 & 2 & 2 & 4 & 2 & 0 & 6 & 0 & 1 & 5 & 0 \\
2 & 6 & 1 & 3 & 1 & 6 & 2 & 1 & 2 & 4 & 5 & 2 & 5 & 1 & 0 & 1 & 4 & 3 & 0 & 0 & 7 & 4
\end{matrix} \right]$$

Fig. 2.4 Parity-check matrix of a (78, 64) d_{min}=5 code

$$
\underline{H} = \left[\begin{matrix} \underline{I}_{16} \end{matrix} \quad \begin{matrix}
4 & 1 & 1 & 3 & 4 & 5 & 0 & 0 & 6 & 4 & 5 & 2 & 0 & 3 & 0 & 0 & 1 & 2 & 4 & 1 & 0 & 4 \\
6 & 0 & 4 & 4 & 2 & 2 & 4 & 0 & 1 & 6 & 2 & 7 & 0 & 1 & 4 & 0 & 1 & 3 & 2 & 2 & 4 & 0 \\
7 & 0 & 2 & 0 & 1 & 1 & 2 & 0 & 0 & 7 & 5 & 1 & 4 & 1 & 6 & 0 & 0 & 7 & 1 & 0 & 2 & 4 \\
3 & 5 & 0 & 0 & 0 & 0 & 5 & 0 & 2 & 3 & 3 & 6 & 6 & 2 & 7 & 0 & 0 & 5 & 0 & 4 & 1 & 0 \\
1 & 7 & 4 & 2 & 0 & 0 & 2 & 4 & 3 & 1 & 5 & 5 & 3 & 0 & 3 & 4 & 0 & 0 & 4 & 2 & 0 & 0 \\
4 & 6 & 6 & 2 & 4 & 1 & 1 & 2 & 1 & 0 & 2 & 4 & 5 & 5 & 1 & 6 & 1 & 0 & 2 & 0 & 0 & 0 \\
6 & 2 & 2 & 0 & 2 & 0 & 4 & 5 & 2 & 0 & 0 & 0 & 2 & 6 & 4 & 7 & 1 & 0 & 5 & 2 & 0 & 0 \\
3 & 0 & 0 & 2 & 5 & 0 & 2 & 2 & 5 & 0 & 0 & 2 & 1 & 1 & 2 & 3 & 4 & 4 & 6 & 7 & 0 & 4 \\
1 & 4 & 1 & 0 & 6 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 0 & 4 & 5 & 1 & 7 & 6 & 3 & 1 & 4 & 4 \\
0 & 6 & 0 & 5 & 3 & 5 & 0 & 4 & 1 & 4 & 0 & 0 & 4 & 2 & 2 & 4 & 6 & 1 & 1 & 5 & 6 & 0 \\
0 & 3 & 1 & 3 & 1 & 2 & 4 & 2 & 0 & 2 & 4 & 2 & 2 & 0 & 1 & 2 & 2 & 4 & 0 & 4 & 7 & 0 \\
4 & 0 & 5 & 4 & 4 & 4 & 2 & 1 & 4 & 5 & 3 & 1 & 1 & 2 & 0 & 5 & 0 & 0 & 4 & 1 & 3 & 0 \\
2 & 1 & 3 & 4 & 2 & 6 & 1 & 0 & 2 & 2 & 0 & 4 & 4 & 6 & 0 & 2 & 4 & 0 & 6 & 3 & 5 & 0 \\
1 & 1 & 4 & 5 & 5 & 2 & 0 & 4 & 1 & 5 & 0 & 0 & 2 & 0 & 0 & 1 & 3 & 0 & 3 & 2 & 6 & 0 \\
0 & 4 & 7 & 3 & 2 & 4 & 0 & 2 & 2 & 2 & 5 & 0 & 1 & 0 & 0 & 0 & 4 & 2 & 1 & 6 & 3 & 4 \\
0 & 2 & 2 & 7 & 1 & 2 & 0 & 1 & 5 & 1 & 2 & 4 & 0 & 6 & 0 & 0 & 2 & 5 & 0 & 4 & 1 & 4
\end{matrix} \right]
$$

Fig. 2.5 Parity-check matrix of a (80,64) d_{min}=5 code

$$\underline{H} = \left[\begin{array}{ccccccc}
 & 5 & 6 & 0 & 4 & 1 & 0 \\
 & 1 & 5 & 1 & 6 & 0 & 4 \\
 & 6 & 0 & 5 & 3 & 0 & 0 \\
 & 6 & 4 & 3 & 5 & 4 & 0 \\
 & 5 & 6 & 0 & 2 & 6 & 0 \\
 & 2 & 7 & 0 & 5 & 3 & 0 \\
\underline{I}_{13} & 4 & 1 & 5 & 2 & 4 & 4 \\
 & 6 & 4 & 6 & 1 & 2 & 0 \\
 & 6 & 2 & 2 & 4 & 5 & 0 \\
 & 2 & 3 & 0 & 2 & 2 & 4 \\
 & 0 & 1 & 5 & 1 & 1 & 0 \\
 & 1 & 0 & 7 & 0 & 5 & 4 \\
 & 3 & 4 & 3 & 0 & 2 & 4
\end{array}\right]$$

Fig. 3.1 Parity-Check Matrix of a (29, 16) d_{min}=6 code

$$
\underline{H} = \left[\begin{array}{cccccccccccc}
 & 4 & 2 & 3 & 4 & 3 & 0 & 2 & 0 & 6 & 6 & 2 \\
 & 0 & 3 & 3 & 2 & 3 & 6 & 3 & 0 & 5 & 1 & 6 \\
 & 6 & 1 & 4 & 1 & 1 & 7 & 1 & 4 & 2 & 4 & 6 \\
 & 5 & 2 & 5 & 0 & 6 & 5 & 6 & 6 & 7 & 0 & 4 \\
 & 6 & 5 & 3 & 4 & 2 & 0 & 5 & 3 & 3 & 4 & 6 \\
 & 3 & 2 & 5 & 6 & 1 & 2 & 2 & 5 & 5 & 6 & 2 \\
\underline{I}_{13} & 7 & 7 & 0 & 3 & 2 & 7 & 1 & 2 & 0 & 1 & 2 \\
 & 5 & 7 & 5 & 1 & 4 & 3 & 4 & 5 & 0 & 0 & 4 \\
 & 6 & 5 & 4 & 4 & 4 & 3 & 6 & 2 & 2 & 6 & 4 \\
 & 3 & 0 & 4 & 6 & 1 & 3 & 5 & 1 & 7 & 1 & 4 \\
 & 3 & 6 & 0 & 3 & 3 & 5 & 4 & 4 & 1 & 2 & 4 \\
 & 1 & 5 & 2 & 1 & 7 & 4 & 4 & 2 & 6 & 7 & 0 \\
 & 0 & 4 & 7 & 0 & 4 & 4 & 0 & 1 & 5 & 5 & 6
\end{array}\right]
$$

Fig. 3.2 Parity-check matrix of a (45, 32) d_{min}=6 code

$$\underline{H} = \left[\begin{array}{cccccccccccc}
 & 6 & 4 & 4 & 4 & 3 & 0 & 4 & 2 & 1 & 7 & 4 \\
 & 3 & 2 & 3 & 2 & 1 & 0 & 6 & 3 & 0 & 2 & 6 \\
 & 1 & 5 & 0 & 1 & 0 & 4 & 3 & 1 & 6 & 5 & 2 \\
 & 4 & 2 & 0 & 4 & 7 & 6 & 1 & 6 & 6 & 0 & 0 \\
 & 6 & 1 & 5 & 2 & 0 & 3 & 4 & 5 & 2 & 2 & 0 \\
 & 1 & 0 & 7 & 1 & 0 & 5 & 6 & 2 & 5 & 0 & 2 \\
 & 0 & 0 & 2 & 4 & 4 & 2 & 7 & 3 & 6 & 1 & 6 \\
\underline{I}_{15} & 4 & 0 & 5 & 2 & 1 & 1 & 3 & 5 & 0 & 6 & 4 \\
 & 4 & 0 & 6 & 1 & 3 & 0 & 5 & 4 & 7 & 0 & 2 \\
 & 2 & 0 & 2 & 4 & 5 & 4 & 2 & 6 & 1 & 5 & 4 \\
 & 5 & 4 & 5 & 2 & 1 & 6 & 5 & 1 & 3 & 0 & 0 \\
 & 0 & 6 & 2 & 5 & 0 & 7 & 2 & 4 & 1 & 1 & 4 \\
 & 0 & 7 & 0 & 2 & 4 & 7 & 1 & 2 & 4 & 1 & 2 \\
 & 2 & 3 & 5 & 1 & 2 & 3 & 0 & 5 & 0 & 1 & 6 \\
 & 5 & 1 & 3 & 0 & 6 & 1 & 0 & 0 & 7 & 6 & 0
\end{array}\right]$$

Fig. 3.3 Parity-check matrix of a (47, 32) d_{min}=6 code

$$\underline{H} = \left[\begin{array}{cccccccccccccccccccccccc}
 & 7 & 2 & 2 & 4 & 3 & 0 & 6 & 0 & 6 & 0 & 4 & 7 & 1 & 4 & 2 & 7 & 1 & 5 & 6 & 5 & 2 & 0 \\
 & 3 & 5 & 1 & 4 & 0 & 4 & 3 & 0 & 3 & 0 & 6 & 1 & 4 & 6 & 1 & 3 & 4 & 6 & 5 & 3 & 7 & 0 \\
 & 2 & 6 & 4 & 2 & 0 & 2 & 1 & 4 & 5 & 4 & 3 & 0 & 6 & 3 & 1 & 5 & 6 & 7 & 2 & 4 & 5 & 4 \\
 & 4 & 1 & 0 & 3 & 3 & 1 & 6 & 6 & 0 & 6 & 5 & 1 & 2 & 5 & 6 & 0 & 6 & 2 & 1 & 6 & 0 & 4 \\
 & 5 & 0 & 6 & 3 & 6 & 4 & 1 & 3 & 6 & 3 & 2 & 1 & 4 & 6 & 4 & 7 & 2 & 0 & 4 & 2 & 2 & 0 \\
 & 0 & 4 & 3 & 7 & 4 & 2 & 0 & 5 & 7 & 1 & 5 & 2 & 6 & 3 & 3 & 2 & 5 & 0 & 0 & 0 & 3 & 0 \\
 & 0 & 0 & 1 & 3 & 7 & 1 & 0 & 2 & 7 & 4 & 6 & 7 & 3 & 1 & 5 & 5 & 2 & 4 & 2 & 1 & 1 & 4 \\
\underline{I}_{15} & 4 & 0 & 2 & 1 & 4 & 4 & 2 & 1 & 1 & 6 & 3 & 6 & 4 & 0 & 5 & 0 & 4 & 7 & 5 & 5 & 6 & 4 \\
 & 4 & 0 & 3 & 6 & 4 & 2 & 7 & 0 & 2 & 7 & 1 & 0 & 3 & 4 & 1 & 3 & 3 & 6 & 0 & 2 & 5 & 0 \\
 & 1 & 0 & 1 & 7 & 3 & 1 & 3 & 4 & 1 & 3 & 4 & 4 & 1 & 6 & 0 & 4 & 5 & 3 & 2 & 1 & 0 & 4 \\
 & 6 & 6 & 2 & 3 & 4 & 4 & 3 & 6 & 2 & 5 & 2 & 5 & 1 & 3 & 2 & 4 & 3 & 0 & 1 & 4 & 4 & 0 \\
 & 0 & 3 & 1 & 3 & 5 & 2 & 1 & 7 & 1 & 2 & 1 & 2 & 4 & 5 & 4 & 2 & 1 & 0 & 2 & 7 & 2 & 0 \\
 & 2 & 3 & 4 & 1 & 6 & 5 & 0 & 7 & 0 & 5 & 4 & 5 & 2 & 2 & 6 & 1 & 0 & 0 & 3 & 2 & 7 & 0 \\
 & 3 & 1 & 6 & 4 & 6 & 2 & 4 & 3 & 4 & 2 & 2 & 2 & 5 & 1 & 2 & 0 & 4 & 4 & 3 & 5 & 1 & 4 \\
 & 6 & 4 & 5 & 4 & 0 & 1 & 4 & 1 & 4 & 1 & 1 & 6 & 3 & 0 & 7 & 6 & 3 & 3 & 5 & 2 & 4 & 4
\end{array}\right]$$

Fig. 3.4 Parity-check matrix of a (79, 64) d_{min}=6 code

$$\underline{H} = \left[\begin{array}{cc} \underline{I}_{17} & \begin{matrix}
4 & 6 & 6 & 3 & 2 & 7 & 0 & 6 & 2 & 1 & 0 & 4 & 4 & 0 & 3 & 3 & 0 & 0 & 0 & 4 & 4 & 0 \\
0 & 1 & 1 & 0 & 5 & 1 & 4 & 5 & 3 & 0 & 7 & 2 & 6 & 0 & 2 & 5 & 4 & 0 & 0 & 6 & 2 & 0 \\
3 & 0 & 1 & 6 & 0 & 0 & 6 & 4 & 5 & 5 & 0 & 5 & 3 & 0 & 1 & 1 & 6 & 0 & 0 & 7 & 1 & 0 \\
1 & 2 & 2 & 4 & 0 & 1 & 3 & 4 & 0 & 6 & 6 & 2 & 5 & 4 & 0 & 0 & 7 & 0 & 0 & 3 & 4 & 4 \\
0 & 5 & 0 & 1 & 2 & 0 & 5 & 0 & 2 & 3 & 3 & 5 & 2 & 6 & 0 & 4 & 3 & 4 & 0 & 1 & 2 & 0 \\
0 & 0 & 1 & 3 & 7 & 7 & 2 & 0 & 1 & 0 & 5 & 2 & 1 & 3 & 0 & 1 & 1 & 6 & 0 & 0 & 5 & 0 \\
4 & 4 & 1 & 4 & 5 & 5 & 5 & 0 & 0 & 4 & 2 & 1 & 0 & 5 & 4 & 0 & 4 & 7 & 0 & 0 & 6 & 4 \\
4 & 0 & 6 & 4 & 2 & 0 & 6 & 0 & 0 & 3 & 1 & 0 & 4 & 2 & 6 & 7 & 2 & 3 & 4 & 0 & 3 & 0 \\
6 & 6 & 6 & 1 & 1 & 4 & 3 & 0 & 0 & 0 & 6 & 0 & 2 & 1 & 0 & 0 & 5 & 1 & 6 & 4 & 1 & 4 \\
3 & 7 & 0 & 2 & 4 & 5 & 1 & 2 & 2 & 0 & 1 & 0 & 1 & 0 & 5 & 0 & 2 & 4 & 7 & 2 & 0 & 4 \\
3 & 7 & 4 & 0 & 2 & 2 & 4 & 1 & 3 & 0 & 2 & 4 & 4 & 4 & 3 & 4 & 1 & 2 & 3 & 1 & 0 & 0 \\
5 & 3 & 6 & 3 & 1 & 2 & 2 & 2 & 5 & 4 & 0 & 2 & 2 & 2 & 0 & 2 & 0 & 5 & 1 & 0 & 4 & 0 \\
4 & 1 & 1 & 1 & 4 & 2 & 1 & 3 & 2 & 7 & 0 & 1 & 1 & 1 & 2 & 6 & 0 & 2 & 4 & 0 & 6 & 0 \\
2 & 0 & 1 & 4 & 4 & 7 & 0 & 1 & 7 & 2 & 5 & 0 & 0 & 4 & 6 & 4 & 0 & 1 & 2 & 0 & 7 & 0 \\
6 & 0 & 2 & 6 & 2 & 0 & 4 & 2 & 5 & 5 & 0 & 4 & 0 & 2 & 2 & 2 & 0 & 0 & 5 & 4 & 3 & 4 \\
1 & 2 & 5 & 3 & 3 & 4 & 2 & 1 & 0 & 7 & 7 & 2 & 0 & 1 & 0 & 2 & 0 & 0 & 2 & 2 & 1 & 4 \\
3 & 5 & 6 & 4 & 5 & 2 & 1 & 4 & 4 & 2 & 4 & 1 & 0 & 0 & 5 & 6 & 0 & 0 & 1 & 1 & 0 & 4
\end{matrix} \end{array}\right]$$

Fig. 3.5 Parity-check matrix of a (81, 64) d_{min}=6 code

$$\underline{H} = \left[\begin{array}{cccccccc}
 & 1 & 0 & 1 & 3 & 2 & 4 \\
 & 4 & 4 & 0 & 0 & 5 & 4 \\
 & 2 & 6 & 0 & 5 & 0 & 0 \\
 & 0 & 7 & 0 & 5 & 4 & 0 \\
 & 0 & 3 & 5 & 4 & 0 & 4 \\
 & 4 & 1 & 6 & 2 & 0 & 0 \\
 & 3 & 0 & 7 & 0 & 4 & 0 \\
 & 4 & 4 & 3 & 1 & 4 & 4 \\
 & 2 & 6 & 0 & 6 & 2 & 0 \\
\underline{I}_{18} & 5 & 3 & 0 & 2 & 5 & 0 \\
 & 6 & 1 & 5 & 1 & 0 & 0 \\
 & 6 & 0 & 6 & 0 & 4 & 0 \\
 & 3 & 0 & 2 & 4 & 2 & 4 \\
 & 1 & 4 & 1 & 0 & 5 & 0 \\
 & 1 & 2 & 1 & 2 & 2 & 4 \\
 & 1 & 1 & 1 & 6 & 3 & 0 \\
 & 4 & 0 & 4 & 0 & 7 & 4 \\
 & 2 & 0 & 2 & 7 & 1 & 0
\end{array}\right]$$

Fig. 4.1 Parity-check matrix of a (34, 16) d_{min}=7 code

$$\underline{H} = \left[\begin{matrix} \underline{I}_{18} \end{matrix} \quad \begin{matrix} 4 & 4 & 4 & 0 & 6 & 3 & 1 & 2 & 2 & 7 & 0 \\ 2 & 2 & 6 & 0 & 3 & 0 & 4 & 6 & 7 & 4 & 4 \\ 1 & 1 & 7 & 0 & 3 & 5 & 2 & 1 & 5 & 1 & 2 \\ 0 & 0 & 3 & 4 & 1 & 5 & 4 & 3 & 4 & 2 & 4 \\ 4 & 0 & 1 & 6 & 4 & 4 & 2 & 0 & 2 & 1 & 2 \\ 2 & 0 & 0 & 7 & 0 & 2 & 0 & 1 & 5 & 1 & 4 \\ 5 & 4 & 4 & 3 & 6 & 0 & 4 & 2 & 0 & 3 & 6 \\ 2 & 2 & 6 & 1 & 5 & 1 & 6 & 2 & 2 & 6 & 6 \\ 5 & 1 & 3 & 0 & 0 & 6 & 3 & 1 & 5 & 2 & 2 \\ 6 & 4 & 1 & 4 & 0 & 2 & 4 & 7 & 0 & 2 & 0 \\ 7 & 2 & 0 & 6 & 6 & 1 & 2 & 2 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 1 & 0 & 4 & 1 & 4 & 1 & 0 \\ 1 & 4 & 4 & 1 & 0 & 4 & 3 & 1 & 2 & 0 & 4 \\ 0 & 6 & 2 & 0 & 6 & 0 & 4 & 4 & 5 & 0 & 2 \\ 0 & 7 & 1 & 0 & 5 & 2 & 3 & 2 & 6 & 4 & 0 \\ 4 & 7 & 0 & 4 & 6 & 6 & 1 & 6 & 1 & 5 & 0 \\ 2 & 3 & 0 & 2 & 1 & 0 & 5 & 5 & 6 & 0 & 4 \\ 1 & 1 & 0 & 1 & 0 & 7 & 2 & 5 & 5 & 6 & 2 \end{matrix} \right]$$

Fig. 4.2 Parity-check matrix of a (50, 32) d_{min}=7 code

$$\underline{H} = \left[\begin{array}{cccccccccccccccccccccccc}
 & 6 & 3 & 2 & 3 & 0 & 0 & 5 & 0 & 5 & 5 & 6 & 0 & 2 & 1 & 5 & 0 & 4 & 0 & 1 & 1 & 3 & 4 \\
 & 5 & 2 & 3 & 0 & 4 & 0 & 2 & 4 & 7 & 3 & 5 & 0 & 3 & 1 & 2 & 4 & 6 & 0 & 1 & 4 & 5 & 0 \\
 & 2 & 5 & 5 & 6 & 2 & 0 & 5 & 2 & 3 & 4 & 2 & 4 & 1 & 4 & 1 & 2 & 3 & 0 & 0 & 7 & 0 & 4 \\
 & 5 & 2 & 2 & 5 & 1 & 0 & 3 & 5 & 1 & 7 & 1 & 2 & 0 & 6 & 5 & 1 & 1 & 4 & 0 & 2 & 7 & 0 \\
 & 6 & 5 & 5 & 2 & 4 & 4 & 0 & 6 & 5 & 6 & 4 & 5 & 0 & 3 & 2 & 0 & 4 & 6 & 0 & 1 & 0 & 4 \\
 & 5 & 1 & 0 & 4 & 2 & 2 & 0 & 3 & 6 & 2 & 4 & 2 & 6 & 0 & 5 & 4 & 6 & 3 & 1 & 0 & 7 & 4 \\
 & 4 & 7 & 6 & 1 & 1 & 1 & 4 & 1 & 3 & 4 & 0 & 1 & 1 & 1 & 7 & 2 & 7 & 1 & 5 & 4 & 0 & 0 \\
 & 4 & 0 & 5 & 3 & 4 & 4 & 2 & 0 & 0 & 2 & 6 & 0 & 6 & 5 & 2 & 1 & 7 & 4 & 7 & 6 & 0 & 4 \\
 & 0 & 3 & 4 & 4 & 6 & 2 & 4 & 0 & 5 & 5 & 1 & 0 & 1 & 3 & 5 & 4 & 3 & 6 & 2 & 6 & 0 & 4 \\
 & 4 & 1 & 2 & 2 & 3 & 1 & 2 & 0 & 2 & 7 & 4 & 4 & 0 & 5 & 2 & 6 & 1 & 7 & 1 & 2 & 2 & 0 \\
\underline{I}_{21} & 6 & 0 & 5 & 3 & 1 & 4 & 4 & 0 & 1 & 2 & 6 & 2 & 0 & 2 & 4 & 7 & 0 & 7 & 4 & 5 & 3 & 0 \\
 & 1 & 3 & 4 & 6 & 4 & 6 & 2 & 0 & 5 & 0 & 1 & 1 & 2 & 0 & 2 & 7 & 0 & 3 & 7 & 2 & 5 & 0 \\
 & 6 & 6 & 0 & 0 & 2 & 3 & 4 & 0 & 6 & 0 & 2 & 4 & 7 & 1 & 0 & 7 & 0 & 1 & 6 & 4 & 2 & 0 \\
 & 3 & 3 & 4 & 0 & 1 & 1 & 6 & 0 & 3 & 0 & 5 & 2 & 3 & 4 & 4 & 3 & 4 & 0 & 7 & 2 & 1 & 0 \\
 & 3 & 6 & 0 & 3 & 0 & 4 & 3 & 0 & 4 & 1 & 0 & 5 & 3 & 7 & 3 & 5 & 2 & 0 & 2 & 5 & 3 & 0 \\
 & 3 & 4 & 6 & 0 & 4 & 2 & 1 & 4 & 6 & 5 & 2 & 2 & 7 & 6 & 0 & 6 & 1 & 0 & 0 & 3 & 6 & 0 \\
 & 1 & 6 & 3 & 2 & 2 & 1 & 4 & 6 & 2 & 2 & 5 & 1 & 3 & 7 & 0 & 3 & 0 & 4 & 0 & 1 & 4 & 0 \\
 & 2 & 4 & 3 & 6 & 1 & 0 & 3 & 3 & 4 & 5 & 4 & 4 & 7 & 6 & 5 & 1 & 0 & 2 & 1 & 0 & 4 & 4 \\
 & 3 & 1 & 3 & 4 & 0 & 4 & 4 & 5 & 2 & 6 & 0 & 2 & 1 & 6 & 6 & 4 & 0 & 1 & 1 & 4 & 3 & 4 \\
 & 1 & 4 & 1 & 4 & 0 & 2 & 6 & 2 & 4 & 2 & 0 & 1 & 0 & 7 & 7 & 2 & 0 & 0 & 4 & 7 & 1 & 4 \\
 & 4 & 6 & 4 & 6 & 0 & 1 & 2 & 1 & 3 & 1 & 4 & 0 & 4 & 3 & 3 & 1 & 0 & 0 & 2 & 2 & 7 & 4
\end{array}\right]$$

Fig. 4.3 Parity-check matrix of a (85, 64) d_{min}=7 code

$$\underline{H} = \left[\ \underline{I}_{24} \quad \begin{matrix}
4 & 4 & 0 & 0 & 1 & 5 & 0 & 6 & 4 & 1 & 4 & 2 & 3 & 6 & 5 & 4 & 2 & 1 & 0 & 6 & 6 & 4 \\
6 & 2 & 0 & 0 & 0 & 2 & 4 & 5 & 2 & 0 & 6 & 1 & 1 & 1 & 2 & 6 & 1 & 0 & 6 & 5 & 3 & 4 \\
3 & 1 & 0 & 0 & 1 & 0 & 2 & 0 & 1 & 0 & 7 & 2 & 4 & 4 & 5 & 3 & 0 & 5 & 3 & 4 & 1 & 4 \\
5 & 4 & 4 & 0 & 0 & 0 & 1 & 6 & 0 & 0 & 3 & 7 & 0 & 4 & 6 & 1 & 6 & 2 & 7 & 6 & 0 & 0 \\
2 & 6 & 2 & 0 & 0 & 1 & 0 & 1 & 4 & 4 & 5 & 5 & 4 & 0 & 3 & 0 & 5 & 0 & 3 & 1 & 6 & 0 \\
1 & 3 & 1 & 0 & 1 & 5 & 4 & 1 & 2 & 2 & 6 & 4 & 4 & 0 & 1 & 4 & 2 & 4 & 1 & 4 & 5 & 0 \\
0 & 1 & 4 & 4 & 1 & 2 & 6 & 7 & 1 & 0 & 3 & 0 & 2 & 0 & 1 & 2 & 3 & 2 & 2 & 0 & 6 & 4 \\
0 & 0 & 6 & 2 & 1 & 0 & 3 & 2 & 4 & 1 & 5 & 4 & 2 & 6 & 4 & 5 & 1 & 4 & 1 & 0 & 7 & 0 \\
0 & 0 & 3 & 1 & 0 & 1 & 1 & 2 & 2 & 4 & 2 & 4 & 0 & 5 & 2 & 6 & 4 & 7 & 2 & 0 & 5 & 0 \\
0 & 0 & 1 & 4 & 5 & 4 & 4 & 2 & 1 & 2 & 1 & 2 & 1 & 6 & 4 & 7 & 0 & 2 & 7 & 0 & 4 & 0 \\
4 & 4 & 0 & 6 & 2 & 2 & 2 & 1 & 0 & 0 & 0 & 5 & 2 & 3 & 3 & 7 & 4 & 1 & 1 & 0 & 6 & 4 \\
2 & 2 & 0 & 3 & 1 & 1 & 1 & 5 & 4 & 4 & 0 & 0 & 6 & 3 & 1 & 3 & 4 & 0 & 6 & 4 & 3 & 0 \\
1 & 5 & 0 & 1 & 5 & 0 & 4 & 3 & 6 & 2 & 0 & 0 & 0 & 3 & 0 & 1 & 6 & 0 & 3 & 6 & 1 & 0 \\
4 & 6 & 4 & 0 & 7 & 5 & 2 & 4 & 7 & 0 & 4 & 2 & 0 & 1 & 0 & 0 & 7 & 0 & 1 & 5 & 4 & 0 \\
6 & 3 & 2 & 0 & 2 & 6 & 5 & 2 & 7 & 1 & 2 & 3 & 2 & 2 & 0 & 0 & 3 & 5 & 0 & 2 & 0 & 0 \\
3 & 1 & 5 & 0 & 0 & 2 & 2 & 1 & 3 & 4 & 1 & 1 & 6 & 1 & 0 & 0 & 3 & 6 & 6 & 1 & 2 & 4 \\
1 & 0 & 6 & 4 & 1 & 0 & 1 & 1 & 5 & 6 & 4 & 6 & 4 & 0 & 4 & 0 & 1 & 6 & 1 & 0 & 3 & 0 \\
4 & 0 & 3 & 2 & 0 & 5 & 0 & 3 & 6 & 7 & 6 & 1 & 0 & 6 & 3 & 4 & 0 & 6 & 0 & 0 & 1 & 0 \\
2 & 0 & 1 & 5 & 1 & 2 & 4 & 2 & 7 & 6 & 7 & 0 & 5 & 5 & 1 & 2 & 0 & 3 & 0 & 2 & 0 & 0 \\
1 & 0 & 0 & 6 & 5 & 4 & 2 & 5 & 3 & 3 & 7 & 4 & 0 & 0 & 4 & 1 & 0 & 1 & 4 & 7 & 0 & 0 \\
0 & 0 & 0 & 3 & 2 & 3 & 1 & 0 & 1 & 5 & 3 & 6 & 2 & 2 & 6 & 4 & 6 & 1 & 4 & 3 & 4 & 0 \\
4 & 0 & 0 & 1 & 4 & 4 & 4 & 7 & 0 & 6 & 1 & 7 & 1 & 5 & 2 & 2 & 1 & 0 & 4 & 3 & 0 & 0 \\
2 & 0 & 0 & 0 & 6 & 6 & 2 & 0 & 0 & 7 & 0 & 5 & 4 & 6 & 4 & 1 & 0 & 5 & 2 & 7 & 2 & 4 \\
1 & 0 & 0 & 0 & 2 & 2 & 1 & 4 & 0 & 3 & 0 & 0 & 7 & 5 & 3 & 0 & 4 & 2 & 5 & 5 & 5 & 4
\end{matrix}\ \right]$$

Fig. 4.4 Parity-check matrix of a (88, 64) d_{min}=7 code

$$\underline{H} = \left[\begin{array}{ccccccc}
 & 0 & 1 & 1 & 5 & 1 & 0 \\
 & 4 & 5 & 0 & 1 & 2 & 4 \\
 & 2 & 3 & 4 & 4 & 5 & 0 \\
 & 1 & 0 & 6 & 2 & 4 & 4 \\
 & 0 & 0 & 6 & 3 & 3 & 0 \\
 & 4 & 0 & 3 & 0 & 1 & 4 \\
 & 2 & 1 & 0 & 2 & 5 & 4 \\
 & 5 & 4 & 0 & 1 & 2 & 4 \\
 & 2 & 2 & 5 & 0 & 2 & 0 \\
\underline{I}_{19} & 5 & 0 & 3 & 2 & 6 & 0 \\
 & 6 & 5 & 0 & 2 & 2 & 0 \\
 & 7 & 2 & 0 & 4 & 1 & 0 \\
 & 3 & 0 & 4 & 3 & 0 & 4 \\
 & 1 & 5 & 2 & 6 & 0 & 0 \\
 & 0 & 6 & 1 & 1 & 4 & 0 \\
 & 0 & 7 & 5 & 4 & 1 & 0 \\
 & 4 & 6 & 2 & 1 & 4 & 4 \\
 & 2 & 2 & 5 & 6 & 0 & 0 \\
 & 1 & 0 & 2 & 6 & 6 & 0
\end{array}\right]$$

Fig. 5.1 Parity-check matrix of a (35, 16) $d_{min}=8$ code

$$\underline{H} = \left[\begin{array}{cccccccccccc}
 & 4 & 2 & 1 & 3 & 0 & 5 & 6 & 1 & 1 & 2 & 0 \\
 & 2 & 1 & 1 & 7 & 4 & 2 & 5 & 1 & 4 & 5 & 0 \\
 & 1 & 0 & 4 & 5 & 6 & 1 & 7 & 4 & 6 & 2 & 4 \\
 & 0 & 4 & 2 & 6 & 7 & 0 & 2 & 6 & 3 & 1 & 2 \\
 & 4 & 0 & 1 & 4 & 3 & 0 & 1 & 3 & 0 & 6 & 4 \\
 & 2 & 0 & 0 & 0 & 1 & 4 & 4 & 4 & 4 & 3 & 2 \\
 & 5 & 2 & 1 & 1 & 0 & 2 & 6 & 2 & 3 & 3 & 4 \\
 & 2 & 5 & 1 & 2 & 4 & 0 & 7 & 1 & 1 & 5 & 6 \\
 & 5 & 0 & 5 & 2 & 2 & 4 & 5 & 4 & 5 & 4 & 6 \\
\underline{I}_{22} & 6 & 6 & 2 & 4 & 1 & 6 & 0 & 6 & 3 & 4 & 2 \\
 & 7 & 1 & 0 & 5 & 0 & 2 & 1 & 2 & 0 & 4 & 0 \\
 & 3 & 4 & 4 & 0 & 4 & 1 & 5 & 4 & 0 & 2 & 0 \\
 & 1 & 6 & 2 & 2 & 2 & 0 & 0 & 7 & 0 & 1 & 0 \\
 & 0 & 7 & 0 & 5 & 1 & 1 & 6 & 2 & 4 & 0 & 4 \\
 & 0 & 3 & 5 & 4 & 4 & 4 & 0 & 1 & 2 & 0 & 2 \\
 & 4 & 3 & 7 & 1 & 2 & 7 & 1 & 0 & 4 & 2 & 0 \\
 & 2 & 1 & 7 & 4 & 5 & 2 & 5 & 5 & 2 & 1 & 0 \\
 & 1 & 0 & 7 & 4 & 2 & 5 & 4 & 6 & 5 & 0 & 4 \\
 & 0 & 4 & 3 & 6 & 1 & 3 & 2 & 2 & 2 & 4 & 2
\end{array}\right]$$

Fig. 5.2 Parity-check matrix of a (51, 32) d_{min}=8 code

$$
\underline{H} = \left[\begin{array}{cccccccccccccccccccccccc}
 & & 5 & 6 & 1 & 3 & 0 & 7 & 2 & 3 & 3 & 7 & 0 & 1 & 2 & 6 & 0 & 0 & 3 & 4 & 0 & 7 & 6 & 0 \\
 & & 2 & 7 & 0 & 5 & 4 & 3 & 7 & 3 & 5 & 7 & 4 & 2 & 5 & 3 & 0 & 0 & 0 & 6 & 0 & 0 & 7 & 4 \\
 & & 6 & 1 & 5 & 0 & 6 & 7 & 5 & 6 & 4 & 0 & 6 & 0 & 4 & 1 & 4 & 0 & 2 & 7 & 0 & 6 & 5 & 0 \\
 & & 1 & 6 & 6 & 4 & 3 & 2 & 4 & 7 & 3 & 0 & 3 & 2 & 6 & 0 & 6 & 0 & 1 & 3 & 4 & 2 & 2 & 0 \\
 & & 2 & 1 & 3 & 2 & 1 & 5 & 2 & 2 & 4 & 4 & 1 & 5 & 3 & 2 & 3 & 0 & 0 & 5 & 6 & 1 & 1 & 4 \\
 & & 6 & 0 & 4 & 6 & 0 & 1 & 7 & 0 & 0 & 5 & 0 & 5 & 3 & 3 & 1 & 4 & 2 & 6 & 7 & 6 & 2 & 0 \\
 & & 3 & 6 & 2 & 3 & 4 & 1 & 5 & 6 & 1 & 2 & 4 & 0 & 1 & 7 & 4 & 6 & 1 & 3 & 3 & 4 & 1 & 0 \\
 & & 3 & 1 & 1 & 1 & 6 & 1 & 4 & 4 & 0 & 5 & 2 & 2 & 4 & 7 & 6 & 3 & 1 & 5 & 5 & 7 & 0 & 0 \\
 & & 3 & 4 & 4 & 4 & 3 & 0 & 6 & 3 & 0 & 2 & 5 & 1 & 2 & 1 & 7 & 1 & 5 & 6 & 6 & 6 & 4 & 4 \\
 & & 4 & 2 & 3 & 0 & 5 & 2 & 1 & 1 & 7 & 4 & 2 & 5 & 3 & 4 & 7 & 4 & 4 & 3 & 3 & 6 & 4 & 0 \\
\underline{I}_{22} & & 2 & 1 & 1 & 4 & 2 & 5 & 0 & 6 & 7 & 4 & 1 & 2 & 1 & 4 & 3 & 6 & 2 & 1 & 5 & 7 & 2 & 4 \\
 & & 6 & 4 & 5 & 4 & 5 & 5 & 6 & 2 & 0 & 3 & 0 & 6 & 6 & 0 & 1 & 7 & 2 & 4 & 6 & 0 & 3 & 0 \\
 & & 1 & 4 & 2 & 6 & 6 & 6 & 7 & 0 & 0 & 1 & 4 & 3 & 7 & 0 & 0 & 7 & 5 & 2 & 3 & 1 & 1 & 4 \\
 & & 7 & 4 & 0 & 0 & 7 & 4 & 1 & 6 & 3 & 5 & 6 & 2 & 1 & 2 & 0 & 3 & 4 & 1 & 1 & 1 & 2 & 4 \\
 & & 4 & 4 & 1 & 3 & 7 & 0 & 2 & 5 & 2 & 3 & 7 & 2 & 2 & 1 & 0 & 1 & 5 & 4 & 4 & 3 & 3 & 0 \\
 & & 0 & 2 & 0 & 4 & 3 & 4 & 3 & 1 & 4 & 1 & 7 & 7 & 5 & 0 & 4 & 0 & 6 & 6 & 2 & 2 & 5 & 4 \\
 & & 5 & 7 & 1 & 0 & 5 & 0 & 1 & 4 & 4 & 5 & 7 & 4 & 0 & 0 & 2 & 0 & 1 & 7 & 1 & 4 & 4 & 4 \\
 & & 5 & 3 & 5 & 7 & 2 & 3 & 2 & 5 & 0 & 5 & 7 & 5 & 2 & 4 & 1 & 0 & 2 & 3 & 4 & 0 & 4 & 0 \\
 & & 0 & 3 & 6 & 7 & 1 & 0 & 5 & 0 & 5 & 2 & 7 & 4 & 5 & 0 & 0 & 4 & 1 & 1 & 6 & 0 & 2 & 4 \\
 & & 5 & 1 & 6 & 1 & 4 & 2 & 2 & 4 & 1 & 0 & 3 & 7 & 4 & 0 & 0 & 2 & 2 & 0 & 7 & 5 & 7 & 0 \\
 & & 0 & 4 & 7 & 0 & 2 & 1 & 1 & 3 & 1 & 6 & 1 & 5 & 6 & 0 & 0 & 1 & 1 & 0 & 3 & 6 & 7 & 4 \\
 & & 7 & 0 & 2 & 7 & 1 & 6 & 4 & 4 & 7 & 2 & 0 & 7 & 5 & 4 & 0 & 0 & 7 & 0 & 1 & 1 & 5 & 4
\end{array} \right]
$$

Fig. 5.3 Parity-check matrix of a (86, 64) d_{min}=8 code

$$\underline{H} = \left[\ \underline{I}_{25} \quad \begin{matrix}
5 & 4 & 5 & 3 & 4 & 3 & 6 & 4 & 2 & 1 & 2 & 6 & 3 & 1 & 1 & 0 & 3 & 0 & 2 & 0 & 6 & 0 \\
2 & 2 & 3 & 4 & 2 & 4 & 0 & 6 & 5 & 0 & 7 & 1 & 2 & 5 & 1 & 4 & 2 & 4 & 1 & 2 & 5 & 0 \\
4 & 1 & 4 & 5 & 0 & 2 & 5 & 6 & 2 & 4 & 0 & 4 & 5 & 3 & 0 & 6 & 0 & 2 & 2 & 7 & 4 & 4 \\
6 & 0 & 3 & 0 & 5 & 4 & 2 & 2 & 3 & 2 & 0 & 6 & 3 & 4 & 4 & 3 & 1 & 1 & 1 & 3 & 6 & 0 \\
2 & 0 & 0 & 6 & 2 & 1 & 1 & 5 & 1 & 4 & 3 & 1 & 2 & 7 & 4 & 1 & 6 & 4 & 4 & 7 & 1 & 0 \\
5 & 4 & 4 & 1 & 0 & 3 & 1 & 7 & 0 & 6 & 1 & 0 & 4 & 3 & 5 & 0 & 5 & 2 & 0 & 1 & 4 & 4 \\
6 & 2 & 3 & 2 & 5 & 4 & 6 & 3 & 4 & 3 & 1 & 0 & 0 & 0 & 2 & 4 & 3 & 5 & 0 & 2 & 0 & 0 \\
6 & 1 & 4 & 4 & 6 & 2 & 0 & 4 & 4 & 1 & 5 & 4 & 1 & 1 & 7 & 2 & 0 & 6 & 4 & 1 & 0 & 0 \\
6 & 0 & 2 & 0 & 6 & 0 & 7 & 3 & 4 & 0 & 5 & 6 & 0 & 4 & 7 & 5 & 0 & 3 & 2 & 2 & 0 & 0 \\
3 & 4 & 4 & 1 & 7 & 0 & 0 & 1 & 2 & 0 & 0 & 7 & 1 & 3 & 1 & 6 & 6 & 1 & 5 & 3 & 0 & 0 \\
0 & 2 & 2 & 1 & 3 & 4 & 6 & 0 & 1 & 1 & 3 & 1 & 6 & 5 & 7 & 7 & 0 & 0 & 6 & 5 & 0 & 0 \\
1 & 1 & 1 & 2 & 1 & 3 & 6 & 1 & 4 & 5 & 4 & 2 & 6 & 2 & 2 & 7 & 4 & 0 & 1 & 2 & 4 & 0 \\
0 & 0 & 4 & 6 & 5 & 1 & 0 & 0 & 4 & 2 & 7 & 3 & 0 & 1 & 4 & 3 & 5 & 0 & 0 & 7 & 6 & 0 \\
0 & 4 & 7 & 0 & 2 & 3 & 5 & 4 & 6 & 0 & 2 & 3 & 4 & 1 & 6 & 1 & 5 & 4 & 0 & 1 & 5 & 0 \\
1 & 6 & 2 & 5 & 4 & 1 & 7 & 2 & 7 & 1 & 2 & 1 & 4 & 1 & 0 & 0 & 5 & 6 & 2 & 0 & 2 & 4 \\
4 & 3 & 0 & 0 & 3 & 4 & 5 & 0 & 5 & 5 & 4 & 0 & 6 & 1 & 6 & 0 & 1 & 7 & 1 & 2 & 1 & 0 \\
3 & 1 & 4 & 3 & 0 & 1 & 5 & 4 & 2 & 6 & 4 & 2 & 3 & 0 & 6 & 0 & 0 & 7 & 4 & 5 & 0 & 4 \\
1 & 4 & 2 & 0 & 5 & 4 & 6 & 7 & 5 & 2 & 2 & 5 & 2 & 4 & 5 & 0 & 2 & 3 & 4 & 0 & 0 & 0 \\
1 & 2 & 0 & 0 & 6 & 7 & 0 & 6 & 0 & 4 & 0 & 6 & 6 & 3 & 6 & 4 & 2 & 1 & 6 & 0 & 2 & 0 \\
5 & 5 & 5 & 0 & 2 & 7 & 2 & 6 & 4 & 2 & 0 & 1 & 1 & 0 & 2 & 2 & 2 & 0 & 7 & 0 & 7 & 0 \\
2 & 2 & 6 & 7 & 1 & 0 & 0 & 3 & 2 & 1 & 1 & 0 & 5 & 4 & 3 & 1 & 3 & 0 & 3 & 4 & 3 & 4 \\
4 & 5 & 3 & 0 & 5 & 0 & 1 & 1 & 3 & 0 & 4 & 6 & 0 & 6 & 6 & 4 & 6 & 4 & 1 & 4 & 3 & 4 \\
2 & 6 & 0 & 4 & 2 & 6 & 7 & 1 & 1 & 4 & 0 & 7 & 1 & 2 & 0 & 2 & 0 & 2 & 2 & 6 & 7 & 4 \\
5 & 3 & 5 & 2 & 0 & 6 & 4 & 1 & 4 & 7 & 1 & 5 & 4 & 4 & 0 & 1 & 0 & 1 & 1 & 1 & 5 & 4 \\
3 & 1 & 2 & 7 & 1 & 7 & 3 & 0 & 4 & 2 & 7 & 0 & 6 & 2 & 2 & 0 & 6 & 0 & 4 & 4 & 0 & 4
\end{matrix} \ \right]$$

Fig. 5.4 Parity-check matrix of a (89, 64) d_{min}=8 code

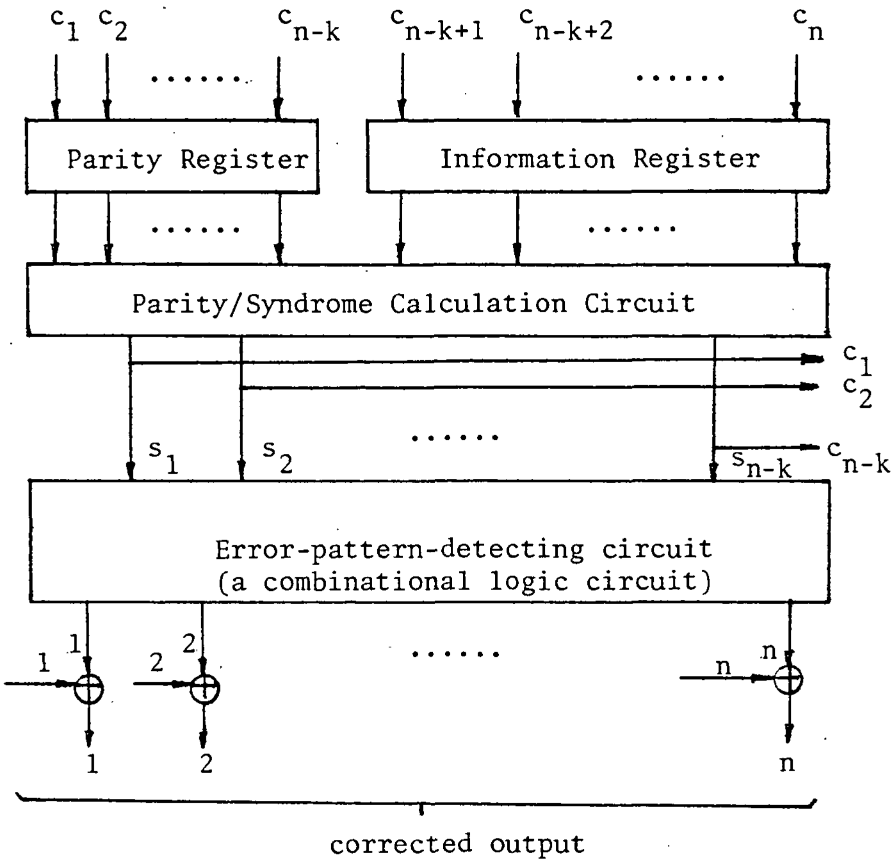


Figure 6. Block diagram of the decoder.

D.J. Costello, Jr. & R.H. Deng
NASA Grant NAG 2-202

FINAL REPORT

PART III

(Burst Error Correction in Laser Memory Systems)

BURST ERROR CORRECTION IN LASER MEMORY SYSTEMS

I. INTRODUCTION

In computer memory systems, errors often occur in bursts. For example, in NASA's laser memory system, the disk can contain a double burst error with lengths $a \leq 8$ bits and $b \leq 11$ bits or a single burst error with length $c \leq 14$ bits. In this paper, we show that by interleaving two $d_{min} = 6$ Reed-Solomon (RS) codes with symbols from $GF(2^m)$, almost all double burst errors of lengths $a \leq m+1$ and $b \leq 2m+1$ can be corrected using erasure decoding, as well as any single burst error of length $c \leq 3m+2$.

Code interleaving distributes the error detection and correction burden among the two component codes and thus lowers the overall redundancy requirement. Erasure decoding uses the information inherent in the interleaving scheme to achieve further improvement in code performance [1]. More importantly, higher speed decoding is possible as opposed to using more powerful codes.

The generator polynomial for the $d_{min} = 6$ RS code is given by [2,3]

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

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

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

where $n \leq 2^m-1$ is the code length. If the code is used for error correction and detection, we say that the decoder operates in the normal mode. The code is then capable of correcting any two or fewer errors and simultaneously detecting any combination of three errors [2,3]. A high-speed decoding procedure for the decoder operating in the normal mode is given in [4]. If, however, the code is used for correcting errors as well as erasures, then all combinations of t errors and e erasures are correctable provided that [3]

$$2t + e < d_{min}, \tag{3}$$

and we say that the decoder is operating in the erasure mode.

Let $v_0^i v_1^i \cdots v_{n-1}^i$ be a codeword generated by the i-th encoder. As shown in Figure 1, code symbols from each encoder are sent alternately using a multiplexer resulting in the sequence $v_0^1 v_0^2 v_1^1 v_1^2 \cdots v_{n-1}^1 v_{n-1}^2$. This sequence is then read into the memory where burst noise is introduced.

At the memory output, each code symbol v_j^i is replaced by its noise corrupted version r_j^i. The memory output sequence $r_0^1 r_0^2 r_1^1 r_1^2 \cdots r_{n-1}^1 r_{n-1}^2$ is then demultiplexed into the respective codewords for decoding. The two decoders can operate in either the normal mode or the erasure mode. When the memory output sequence is deinterleaved, assume decoder 1 first operates in the normal mode, i.e., it does independent decoding. If the errors in $\underline{r}^1 = (r_0^1 r_1^1 \cdots r_{n-1}^1)$ are correctable, then after error correction the error location information is fed into decoder 2 through the mode selector, and decoder 2 will operate in the erasure mode. That is, it does erasure decoding by using the error location information provided by decoder 1. On the other hand, if the errors in $\underline{r}^1$ are not correctable but can be detected, decoder 1 stops decoding and decoder 2 starts operating in the normal mode. If the errors in $\underline{r}^2 = (r_0^2, r_1^2, \cdots, r_{n-1}^2)$ are correctable, then the error location information from decoder 2 is fed into decoder 1. By using this information, decoder 1 will do erasure decoding and the errors in $\underline{r}^1$ will be corrected.

To have a better idea of how the decoder works, let us look at some particular cases. Assume that $\underline{r} = (r_0^1 r_0^2 r_1^1 r_1^2 \cdots r_{n-1}^1 r_{n-1}^2)$ contains two bursts of lengths $a \leq m+1$ and $b \leq 2m+1$. Then the burst of length $a \leq m+1$ can affect at most two symbols in $\underline{r}$ and the burst of length $b \leq 2m+1$ can affect at most three symbols in $\underline{r}$, as shown below.

$$\underline{r} = (r_0^1 r_0^2 r_1^1 r_1^2 \cdots \underset{\text{in error}}{\underset{\uparrow\quad\uparrow}{r_i^1 \, r_i^2}} \cdots r_{j-1}^1 \underset{\text{in error}}{\underset{\uparrow\quad\uparrow\quad\uparrow}{r_{j-1}^2 r_j^1 r_j^2}} \cdots r_{n-1}^1 r_{n-1}^2)$$

The two code words that result from deinterleaving $\underline{r}$ are shown below.

$$\underline{r}^1 = (r_0^1 r_1^1 \cdots \underset{\text{in error}}{\underset{\uparrow}{r_i^1}} \cdots\cdots\cdots \underset{\text{in error}}{\underset{\uparrow}{r_j^1}} \cdots r_{n-1}^1)$$

$$\underline{r}^2 = (r_0^2 r_1^2 \cdots \underset{\text{in error}}{\underset{\uparrow}{r_i^2}} \cdots\cdots\cdots \underset{\text{in error}}{\underset{\uparrow\quad\uparrow}{r_{j-1}^2 r_j^2}} \cdots r_{n-1}^2)$$

Let decoder 1 operate in the normal mode first. Because $\underline{r}^1$ contains two errors at locations i and j, respectively, they can always be corrected. Then the symbols at locations i-1, i, j-1, and j in $\underline{r}^2$ are erased. Note that the symbols at location i, j-1, and j are in error. By operating decoder 2 in the erasure mode, the four erasures in $\underline{r}^2$ can be corrected. If, however, decoder 2 operates in the normal mode first instead of decoder 1, because $\underline{r}^2$ contains three errors, they will be detected but not be corrected. Then decoder 2 stops decoding, and decoder 1 starts operating in the normal mode. The same decoding procedure as described above will now follow.

As another example, again suppose that $\underline{r}$ contains two bursts, but the first burst only affects r_i^2, and the second burst only affects r_{j-1}^2 and r_j^2, as shown below.

$$\underline{r} = (r_0^1\, r_0^2\, r_1^1\, r_1^2\, \cdots\, r_i^1\, \underset{\substack{\uparrow \\ \text{in error}}}{r_i^2}\, \cdots\, r_{j-1}^1\, \underset{\uparrow}{r_{j-1}^2}\, \underset{\substack{ \\ \text{in error}}}{r_j^1}\, \underset{\uparrow}{r_j^2}\, \cdots\, r_{n-1}^1\, r_{n-1}^2)$$

Let decoder 1 operate in the normal mode first. Then no errors will be detected in $\underline{r}^1$, and when decoder 2 starts operating in the normal mode the three errors in $\underline{r}^2$ will be detected but cannot be corrected. Fortunately, situations like this are rare, as we will show in the next section.

The decoding procedure for a decoder operating in the normal mode is presented in [4]. Therefore, in the following, we focus our attention only on erasure decoding.

II. ERASURE DECODING PROCEDURE

Let $\underline{r}^i = \underline{v}^i + \underline{e}^i$ be the input to the i-th decoder. The decoder first computes the syndrome $\underline{s}^{(i)}$,

$$\begin{aligned} \underline{s}^{(i)} &= \underline{r}^i \underline{H}^T = (\underline{v}^i + \underline{e}^i)\underline{H}^T = \underline{e}^i \underline{H}^T \\ &= (s_{-2}^{(i)},\ s_{-1}^{(i)},\ s_0^{(i)},\ s_1^{(i)},\ s_2^{(i)}). \end{aligned} \tag{4}$$

Without loss of generality, assume that after the deinterleaving of the output sequence, decoder 1 operates in the normal mode. Let t and e denote the number of errors and the number of erasures in $\underline{r}^2$, respectively. Then there exists the following possibilities:

(1) No errors are detected in $\underline{r}^1$. If this is the case, then decoder 2 starts operating in the normal mode. If $t \leq 2$, the errors will be corrected. If, however, $t = 3$, the decoder will fail. Suppose that each error bit within a burst takes on a value of 1 with probability 1/2. Then this failure probability is given by

$$P_1 = \frac{1}{m+1} \{ \sum_{i=1}^{m} (\tfrac{1}{2})^{m+1-i} [1 - (\tfrac{1}{2})^{i}] \}$$

$$\cdot \frac{1}{2m+1} \{ \sum_{i=1}^{m} [1 - (\tfrac{1}{2})^{i}] (\tfrac{1}{2})^{m+1-i} \} (\tfrac{1}{2})^{m}$$

$$= \frac{(\frac{1}{2})^{m}}{(m+1)(2m+1)} \{ \sum_{i=1}^{m} (\tfrac{1}{2})^{m+1-i} [1 - (\tfrac{1}{2})^{i}] \}. \qquad (5)$$

(2) One error at location k in $\underline{r}^1$ is detected and corrected. The symbols at locations k-1 and k in $\underline{r}^2$ are erased. Then decoder 2 starts operating in the erasure mode. Three cases can occur

(i) $e = 2$, $t = 0$.

$$s_\ell^{(2)} = e_{k-1}\alpha^{\ell(k-1)} + e_k\alpha^{\ell k} \quad , \quad \ell = -2, -1, 0, 1, 2. \qquad (6)$$

From (6), the error values at locations k-1 and k are given by

$$e_{k-1} = \frac{\begin{vmatrix} s_0^{(2)} & 1 \\ s_1^{(2)} & \alpha^k \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \alpha^{k-1} & \alpha^k \end{vmatrix}} = (s_1^{(2)} + \alpha^k s_0^{(2)}) \ / \ \alpha^k(1+\alpha^{-1}), \qquad (7.1)$$

and

$$e_k = s_0^{(2)} + e_{k-1}, \qquad (7.2)$$

respectively.

(ii) $e = 2$, $t = 1$, where the error is at location $j \neq k-1, k$.

$$s_\ell^{(2)} = e_{k-1}\alpha^{\ell(k-1)} + e_k\alpha^{\ell k} + e_j\alpha^{\ell j} \quad , \quad \ell = -2, -1, 0, 1, 2. \qquad (8)$$

Define

$$T_1 = s_1^{(2)} + \alpha^k(1+\alpha^{-1})s_0^{(2)} + \alpha^{2k-1}\ s_{-1}^{(2)},$$

$$T_2 = s_2^{(2)} + \alpha^k(1+\alpha^{-1})s_1^{(2)} + \alpha^{2k-1}\ s_0^{(2)}.$$

Then the error locator α^j and the error value e_j can be found from [3]:

$$\alpha^j = T_2/T_1 \ , \tag{9}$$

and

$$e_j = \frac{\begin{vmatrix} 1 & 1 & s_0^{(2)} \\ \alpha^{k-1} & \alpha^k & s_1^{(2)} \\ \alpha^{2(k-1)} & \alpha^{2k} & s_2^{(2)} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 \\ \alpha^{k-1} & \alpha^k & \alpha^j \\ \alpha^{2(k-1)} & \alpha^{2k} & \alpha^{2j} \end{vmatrix}} = \frac{T_2\alpha^k(\alpha^{-1} + 1)}{(\alpha^k+\alpha^{k-1})(\alpha^j+\alpha^{k-1})(\alpha^j+\alpha^k)}$$

$$= \frac{T_2}{(\alpha^j+\alpha^{k-1})(\alpha^j+\alpha^k)} \ . \tag{10.1}$$

The other error values are given by

$$e_{k-1} = \frac{\begin{vmatrix} e_j \quad + s_0^{(2)} & 1 \\ e_j\alpha^j + s_1^{(2)} & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \alpha^{k-1} & \alpha^k \end{vmatrix}}$$

$$= (e_j + s_0^{(2)} + e_j\alpha^j + s_1^{(2)}) \ / \ \alpha^k(\alpha^{-1} + 1), \tag{10.2}$$

and

$$e_k = s_0^{(2)} + e_{k-1} + e_j. \tag{10.3}$$

(iii) $e = 2$, $t = 2$, where the errors are at locations j and i ($\neq$ k-1, k), respectively. Because

$$2t+e = 4+2 = 6 = d_{min},$$

this error pattern in $\underline{r}^2$ cannot be corrected, and the probability of this failure occurring is given by

$$P_2 = \frac{1}{m+1} \{ \sum_{i=1}^{m} [1 - (\frac{1}{2})^m] \} \cdot \frac{1}{2m+1} \{ \sum_{i=1}^{m} [1 - (\frac{1}{2})^i] (\frac{1}{2})^{m+1-i} \}. \tag{11}$$

(3) Two adjacent errors in $\underline{r}^1$ are detected and corrected. Then $\underline{r}^2$ contains at most two byte errors, which can always be corrected by operating decoder 2 in the normal mode.

(4) Two nonadjacent errors in $\underline{r}^1$ at locations i and j are detected and corrected. Then the symbols at locations i-1, i, j-1, and j in $\underline{r}^2$ are erased, and decoder 2 does erasure decoding.

$$s_\ell^{(2)} = e_{i-1}\alpha^{\ell(i-1)} + e_i\alpha^{\ell i} + e_{j-1}\alpha^{\ell(j-1)} + e_j\alpha^{\ell j},$$

$$\ell = -2, -1, 0, 1, 2. \tag{12}$$

Solving (12) for the error values yields

(i) $e_{i-1} = A/B,$ (13.1)

where

$$\begin{aligned} A = {} & (s_{-1}^{(2)}\alpha^{i+2j-1} + s_2^{(2)})(\alpha^{j-1} + \alpha^i)(\alpha^j + \alpha^i)\alpha^j(1 + \alpha^{-1}) \\ & + s_0^{(2)}[\alpha^{5j-2}(1 + \alpha^{-1}) + \alpha^{2i+3j}(1 + \alpha^{-3}) + \alpha^{3i+2j}(1 + \alpha^{-1})] \\ & + s_1^{(2)}[\alpha^{4j-1}(1 + \alpha^{-2}) + \alpha^{i+3j}(1 + \alpha^{-1}) + \alpha^{3i+j}(1 + \alpha^{-1})], \end{aligned}$$

and

$$B = \alpha^{j}(1 + \alpha^{-2})(\alpha^{2i} + \alpha^{2j})(\alpha^{j} + \alpha^{i-1})(\alpha^{j-1} + \alpha^{i});$$

(ii) $$e_i = C/D, \tag{13.2}$$

where

$$C = (s_0^{(2)} + e_{i-1})\alpha^{3j-1} + (s_1 + e_{i-1}\alpha^{i-1})\alpha^{2j}(1 + \alpha^{-1})$$
$$+ (s_2 + e_{i-1}\alpha^{2(i-1)})\alpha^{j},$$

$$D = (\alpha^{j-1} + \alpha^{i})(\alpha^{j} + \alpha^{i})\alpha^{j};$$

(iii) $$e_{j-1} = [\alpha^{j}(s_0^{(2)} + e_{i-1} + e_i) + s_1^{(2)} + \alpha^{i}(e_{i-1}\alpha^{-1} + e_i)] / \alpha^{j}(1 + \alpha^{-1}); \tag{13.3}$$

and

(iv) $$e_j = s_0^{(2)} + e_{i-1} + e_i + e_j. \tag{13.4}$$

(5) Three errors are detected in $\underline{r}^1$. Then decoder 2 starts operating in the normal mode, and the error location information is fed into decoder 1. Decoder 1 then does erasure decoding, as described in (1)-(4), the only difference being that if an error at location i in $\underline{r}^2$ is corrected, the symbols in $\underline{r}^1$ at locations i and i+1 rather than at locations i-1 and i are erased.

Let P be the probability that two bursts of lengths a = m+1 and b = 2m+1 occur. Then the probability of decoding failure, from (5) and (11), is given by

$$P_E = \frac{P}{(m+1)(2m+1)} \left\{ \sum_{i=1}^{m} [1 - (\tfrac{1}{2})^{i}](\tfrac{1}{2})^{m+1-i} \right\}$$
$$\cdot \left\{ (\tfrac{1}{2})^{m} + \sum_{i=1}^{m} [1 - (\tfrac{1}{2})^{m}] \right\}. \tag{14}$$

If m = 8, then (14) yields

$$P_E \approx 3.3P\times 10^{-3} .$$

Before proceeding, we need to prove the following property. Let e and t denote the number of erasures and the number of errors in $\underline{r}^i$, respectively:

Property: If e = 2 and

$$\Delta^{(i)} \triangleq s_0^{(i)}({s_0^{(i)}}^2 + s_1^{(i)}\ s_{-2}^{(i)}) + s_1^{(i)}(s_0^{(i)}\ s_{-1}^{(i)} + s_1^{(i)}\ s_2^{(i)})$$

$$+ s_2^{(i)}({s_{-1}^{(i)}}^2 + s_0^{(i)}\ s_2^{(i)}) = 0,$$

then t = 0 or t > 1.

Proof: We have to show t ≠ 1. It is sufficient to show that [3]

$$\Delta^{(i)} = \begin{vmatrix} s_0^{(i)} & s_{-1}^{(i)} & s_{-2}^{(i)} \\ s_1^{(i)} & s_0^{(i)} & s_{-1}^{(i)} \\ s_2^{(i)} & s_1^{(i)} & s_0^{(i)} \end{vmatrix} = 0.$$

Because

$$\Delta^{(i)} = s_0^{(i)}({s_0^{(i)}}^2 + s_1^{(i)}\ s_{-2}^{(i)}) + s_1^{(i)}(s_0^{(i)}\ s_{-1}^{(i)} + s_1^{(i)}\ s_2^{(i)})$$

$$+ s_2^{(i)}({s_{-1}^{(i)}}^2 + s_0^{(i)}\ s_2^{(i)}) = 0$$

by our assumption, the property is true.

Based on the above discussion, our decoding procedure can be summarized as follows. Assume that $\underline{r}^1$ is first decoded by decoder 1 in the normal mode.

(1) No errors are detected in $\underline{r}^1$. Decoder 2 starts decoding $\underline{r}^2$ in the normal mode.

- If two or fewer errors occur in $\underline{r}^2$, they are corrected.
- If three errors occur in $\underline{r}^2$, they are detected but cannot be corrected.

(2) One error at location k in $\underline{r}^1$ is corrected. The symbols at locations k-1 and k in $\underline{r}^2$ are erased.

- If $\Delta^{(2)} = 0$, two erasures are corrected with error values e_{k-1} and e_k given by (7.1) and (7.2), respectively.
- If $\Delta^{(2)} \neq 0$, two erasures and one error are corrected with error values e_{k-1}, e_k, and e_j given by (10.1), (10.2), and (10.3), respectively, where the error location j is given by (9).

(3) Two adjacent errors in $\underline{r}^1$ are corrected. Decoder 2 starts decoding $\underline{r}^2$ in the normal mode.

(4) Two nonadjacent errors in $\underline{r}^1$ at locations i and j are corrected. The symbols at locations i-1, i, j-1, and j in $\underline{r}^2$ are erased. Four erasures are corrected with error values e_{i-1}, e_i, e_{j-1}, and e_j given by (13.1), (13.2), (13.3), and (13.4), respectively.

(5) Three errors are detected in $\underline{r}^1$. Decoder 2 starts decoding $\underline{r}^2$ in the normal mode.

- If no errors are detected in $\underline{r}^2$, stop. The errors in $\underline{r}^1$ cannot be corrected.
- If one error at location k in $\underline{r}^2$ is corrected, the symbols at locations k and k+1 in $\underline{r}^1$ are erased. Replace $\underline{s}^{(2)}$, k-1, and k by $\underline{s}^{(1)}$, k, and k+1, respectively, in (9) and (10.1)-(10.3). Decoder 1 corrects two erasures and one error with error values e_k, e_{k+1}, and e_j given by (10.1)-(10.3), where the error location j is given by (9).

- If two errors at locations i and j in $\underline{r}^2$ are corrected, the symbols at locations i, i+1, j, and j+1 in $\underline{r}^1$ are erased. Replace $\underline{s}^{(2)}$, i-1, i, j-1, and j by $\underline{s}^{(1)}$, i, i+1, j, and j+1, respectively, in (13.1)-(13.4). Decoder 1 corrects four erasures with error values e_i, e_{i+1}, e_j, and e_{j+1} given by (10.1)-(10.4), respectively.

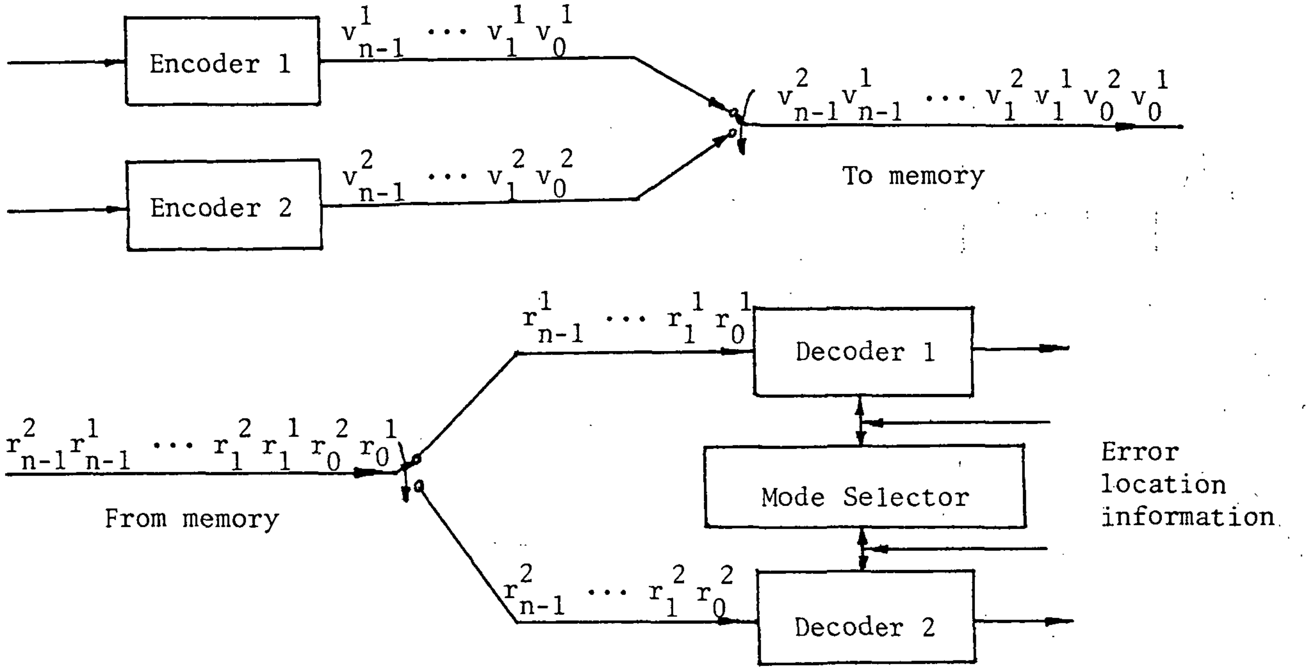
REFERENCES

[1] K.S. Leung and L.R. Welch, "Erasure Decoding in Burst-Error Channels", IEEE Trans. Inform. Theory, Vol. IT-27, No. 2, pp. 160-167, March 1980.

[2] S. Lin and D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, Inc., New Jersey, 1983.

[3] W.W. Peterson and E.J. Weldon, Error-Correcting Codes , 2nd ed., MIT Press, Cambridge, MA, 1972.

[4] H. Deng and D.J. Costello, Jr., "Reed Solomon Codes for Error Control in Byte Organized Computer Memory Systems", submitted to IEEE Trans. on Computers, August 1985.



Encoder 1
$v^1_{n-1} \cdots v^1_1 v^1_0$
Encoder 2
$v^2_{n-1} \cdots v^2_1 v^2_0$
$v^2_{n-1} v^1_{n-1} \cdots v^2_1 v^1_1 v^2_0 v^1_0$
To memory
$r^2_{n-1} r^1_{n-1} \cdots r^2_1 r^1_1 r^2_0 r^1_0$
From memory
$r^1_{n-1} \cdots r^1_1 r^1_0$
Decoder 1
Mode Selector
$r^2_{n-1} \cdots r^2_1 r^2_0$
Decoder 2
Error location information

Figure 1. Encoding-decoding block diagram

D.J. Costello, Jr. & R.H. Deng

NASA Grant NAG 2-202

FINAL REPORT

PART IV

(High Speed Decoding of Extended

Triple-Byte-Error-Correcting Reed-Solomon Codes)

HIGH SPEED DECODING OF EXTENDED TRIPLE-BYTE-ERROR-CORRECTING REED-SOLOMON CODES

I. INTRODUCTION

Reed-Solomon (RS) codes are a class of nonbinary codes with symbols or bytes from the Galois field of 2^m elements ($GF(2^m)$). They are maximum distance separable, 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. Chen et al. [1] presented a simplified high speed decoding scheme for Reed-Solomon codes capable of correcting up to three byte errors in code words made up of k data and n-k parity-check bytes. In this paper, we modify Chen's scheme to decode extended triple-byte-error-correcting (TBEC) Reed-Solomon codes.

A typical RS decoding procedure is to first claculate the syndrome, then find the error location polynomial and search for its roots, and finally compute the error values and make the actual corrections. Finding the error location polynomial remains the major bottleneck in high-speed decoding of RS codes. Some general solutions to this problem are known, such as those described in [2,3]. It is also possible to obtain particular solutions for specific applications. The method described in [1] is based on the idea of checking for single errors first and correcting them prior to checking for multiple errors. Since most errors are single errors and checking for multiple errors is time consuming, this method is not only high-speed but also simple to implement.

In practice, it is also desirable that the coding overhead be as low as possible without sacrificing the error correcting capability. Extended RS codes provide two more information bytes than ordinary RS codes, and therefore a lower coding overhead, while retaining the same error correcting capability. In this report, we modify the decoding scheme given in [1] to extended Triple-Byte-Error-Correcting (TBEC) RS codes. The decoder first tests if the second syndrome symbol $S_1 = 0$. If $S_1 \neq 0$, the assumption is made that a single byte error has occurred. The assumption is verified quickly by the decoder. If the error is not a single byte error the decoder goes on to determine if a double byte error has occurred. If not, it then goes on to determine if there is a triple byte error.

II. PRELIMINARIES

The TBEC Reed-Solomon code has 6 parity-check bytes or symbols, and minimum distance $d_{min} = 7$. The code symbols are elements of $GF(2^m)$, the finite field of 2^m elements, where m is the number of bits in each symbol. The generator polynomial g(X) of the code is [3]

$$g(X) = \prod_{i=0}^{5} (X+\alpha^i), \qquad (1)$$

where α is a primitive element of $GF(2^m)$. The code length n is assumed to be less than or equal to 2^m-1. The parity-check matrix of the code can be written as

$$\underline{H} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & \alpha & (\alpha)^2 & \cdots & (\alpha)^{n-2} & (\alpha)^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \cdots & (\alpha^2)^{n-2} & (\alpha^2)^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & \cdots & (\alpha^3)^{n-2} & (\alpha^3)^{n-1} \\ 1 & \alpha^4 & (\alpha^4)^2 & \cdots & (\alpha^4)^{n-2} & (\alpha^4)^{n-1} \\ 1 & \alpha^5 & (\alpha^5)^2 & \cdots & (\alpha^5)^{n-2} & (\alpha^5)^{n-1} \end{bmatrix} . \qquad (2)$$

Let $\underline{V} = (v_0, v_1, \cdots, v_{n-1})$ and $\underline{R} = (r_0, r_1, \cdots, r_{n-1})$ be the transmitted codeword and the received vector, respectively. The difference between $\underline{R}$ and vector,

$\underline{V}$ is the error vector $\underline{E} = (e_0, e_1, \cdots, e_{n-1})$, i.e.,

$$\underline{R} = \underline{V} + \underline{E} = (v_0+e_0, v_1+e_1, \cdots, v_{n-1} + e_{n-1}). \tag{3}$$

The syndrome vector is, by definition,

$$\underline{S} = \underline{R}\,\underline{H}^T = \underline{E}\,\underline{H}^T = (S_0, S_1, S_2, S_3, S_4, S_5), \tag{4}$$

and depends only on the error vector, not on the particular transmitted codeword. From this definition we have

$$S_i = \sum_j e_j (\alpha^j)^i.$$

Let $X_j = \alpha^j$ represent the error location j. X_j is called the error location number. Knowing X_j is equivalent to knowing the error location j. Then

$$S_i = \sum_j e_j X_j^i \quad , \quad 0 \leq i \leq 5. \tag{5}$$

The error-location polynomial $\sigma(X)$ is defined as

$$\sigma(X) = \prod_i (X-X_i) = X^e + \sigma_1 X^{e-1} + \cdots + \sigma_e, \tag{6}$$

where e is the number of errors in the received codeword. The coefficients of $\sigma(X)$ are related to the syndromes S_i by the following equations [2]:

$$S_{i+e} + \sigma_1 S_{i+e-1} + \cdots + \sigma_{e-1} S_{i+1} + \sigma_e S_i = 0 \quad , \quad 0 \leqq i \leqq 5\text{-}e. \tag{7}$$

The extended triple-byte-error-correcting Reed-Solomon code is an (n+2, n-4) code, whose parity-check matrix is given by [4,5,6]

$$\underline{H}_E = \begin{bmatrix} & 1 & 0 \\ & 0 & 0 \\ \underline{H} & 0 & 0 \\ & 0 & 0 \\ & 0 & 0 \\ & 0 & 1 \end{bmatrix}, \tag{8}$$

where $\underline{H}$ is given by (2). We again have

$$\underline{S} = \underline{R}\,\underline{H}_E^T = (r_0, r_1, \cdots, r_{n+1})\underline{H}_E^T = (e_0, e_1, \cdots, e_{n+1})H_E^T$$

$$= (S_0, S_1, S_2, S_3, S_4, S_5).$$

It is easily seen from (8) and (5) that

$$S_0 = \sum_{j=0}^{n-1} e_j + e_n \quad , \tag{9.1}$$

$$S_i = \sum_{j=0}^{n-1} e_j X_j^i \quad , \quad \text{for} \quad 1 \leq i \leq 4, \tag{9.2}$$

$$S_5 = \sum_{j=0}^{n-1} e_j X_j^5 + e_{n+1}. \tag{9.3}$$

Note that the error at location n only contributes to S_0, and the error at location n+1 only to S_5.

III. THE MODIFIED DECODING SCHEME

For the extended code, we denote e as the number of errors at locations 0 through n-1, e' as the number of errors at locations n and n+1, and E = e+e' as the total number of errors in a received vector, respectively. Obviously, if the errors are all confined to locations 0 to n-1, (7) is still valid. Then we can write out the equations of (7) explicitly for the following three cases:

Case 1) E = e = 1:

$$\begin{aligned} S_1 + \sigma_1 S_0 &= 0, \\ S_2 + \sigma_1 S_1 &= 0, \\ S_3 + \sigma_1 S_2 &= 0, \\ S_4 + \sigma_1 S_3 &= 0, \\ S_5 + \sigma_1 S_4 &= 0. \end{aligned} \tag{10}$$

Case 2) E = e = 2:

$$\begin{aligned} S_2 + \sigma_1 S_1 + \sigma_2 S_0 &= 0, \\ S_3 + \sigma_1 S_2 + \sigma_2 S_1 &= 0, \\ S_4 + \sigma_1 S_3 + \sigma_2 S_2 &= 0, \\ S_5 + \sigma_1 S_4 + \sigma_2 S_3 &= 0. \end{aligned} \tag{11}$$

Case 3) E = e = 3:

$$\begin{aligned} S_3 + \sigma_1 S_2 + \sigma_2 S_1 + \sigma_3 S_0 &= 0, \\ S_4 + \sigma_1 S_3 + \sigma_2 S_2 + \sigma_3 S_1 &= 0, \\ S_5 + \sigma_1 S_4 + \sigma_2 S_3 + \sigma_3 S_2 &= 0. \end{aligned} \tag{12}$$

Now let us consider the cases when some of the errors are located at positions n or n+1. There exist eight possibilities. It follows from (9.1)-(9.3) and (7) that:

Case 4) E = e' = 1 at location n:

$$S_0 \neq 0, \; S_i = 0, \quad \text{for} \quad 1 \leq i \leq 5. \tag{13}$$

Case 5) E = e' = 1 at location n+1:

$$S_i = 0, \quad \text{for} \quad 0 \leq i \leq 4, \; S_5 \neq 0. \tag{14}$$

Case 6) E = e' = 2 at locations n and n+1:

$$S_0 \neq 0, \quad S_i = 0, \quad 1 \leq i \leq 4, \quad S_5 \neq 0. \tag{15}$$

Case 7) E = 2, e = 1, e' = 1, at locations j, $0 \leq j \leq n-1$, and n:

$$\begin{aligned} &S_1 + \sigma_1 S_0 \neq 0, \\ &S_2 + \sigma_1 S_1 = 0, \\ &S_3 + \sigma_1 S_2 = 0, \\ &S_4 + \sigma_1 S_3 = 0, \\ &S_5 + \sigma_1 S_4 = 0. \end{aligned} \tag{16}$$

Case 8) E = 2, e = 1, e' = 1, at locations j, $0 \leq j \leq n-1$, and n+1:

$$\begin{aligned} &S_1 + \sigma_1 S_0 = 0, \\ &S_2 + \sigma_1 S_1 = 0, \\ &S_3 + \sigma_1 S_2 = 0, \\ &S_4 + \sigma_1 S_3 = 0, \\ &S_5 + \sigma_1 S_4 \neq 0. \end{aligned} \tag{17}$$

Case 9) E = 3, e = 1, e' = 2, at locations j, $0 \leq j \leq n-1$, n, and n+1:

$$\begin{aligned} &S_1 + \sigma_1 S_0 \neq 0, \\ &S_2 + \sigma_1 S_1 = 0, \\ &S_3 + \sigma_1 S_2 = 0, \\ &S_4 + \sigma_1 S_3 = 0, \\ &S_5 + \sigma_1 S_4 \neq 0. \end{aligned} \tag{18}$$

Case 10) $E = 3$, $e = 2$, $e' = 1$, at locations j, k, $0 \leq j < k \leq n-1$, and n:

$$\begin{aligned} S_2 + \sigma_1 S_1 + \sigma_2 S_0 &\neq 0, \\ S_3 + \sigma_1 S_2 + \sigma_2 S_1 &= 0, \\ S_4 + \sigma_1 S_3 + \sigma_2 S_2 &= 0, \\ S_5 + \sigma_1 S_4 + \sigma_2 S_3 &= 0. \end{aligned} \tag{19}$$

Case 11) $E = 3$, $e = 2$, $e' = 1$, at locations j, k, $0 \leq j < k \leq n-1$, and $n+1$:

$$\begin{aligned} S_2 + \sigma_1 S_1 + \sigma_2 S_0 &= 0, \\ S_3 + \sigma_1 S_2 + \sigma_2 S_1 &= 0, \\ S_4 + \sigma_1 S_3 + \sigma_2 S_2 &= 0, \\ S_5 + \sigma_1 S_4 + \sigma_2 S_3 &\neq 0. \end{aligned} \tag{20}$$

Our problem is to find a simple way to solve for σ_i based on the above equations. Let $\underline{S}_s$, $\underline{S}_d$, and $\underline{S}_t$ denote the syndrome vectors for single, double, and triple byte error patterns, respectively. Then [3]

$$\underline{S}_s \neq \underline{S}_d \neq \underline{S}_t. \tag{21}$$

The following properties of the syndromes are important to our decoding procedure. First, suppose that the error locations are confined to positions 0 through n-1.

<u>Property 1</u>: If $S_1 \neq 0$ and $S_3 S_1 + S_2^2 = 0$, then $e = 1$, or $e \geq 3$.

<u>Proof</u>: Because $S_1 \neq 0$, $e \neq 0$, and we have to show that $e \neq 2$. From [2], it is sufficient to show that the determinant

$$\Delta \triangleq \begin{vmatrix} S_2 & S_1 \\ S_3 & S_2 \end{vmatrix} = S_2^2 + S_1 S_3 = 0.$$

Since $S_2^2 + S_1S_3 = 0$ by assumption, Property 1 is true.

Property 2: If $S_1 \neq 0$, $S_3S_1 + S_2^2 = 0$, and $S_1S_4 + S_2S_3 = 0$, then $e = 1$ or $e > 3$.

Proof: From Property 1, we only have to show $e \neq 3$. From [2], it is sufficient to show that the determinant

$$\Delta_3 \triangleq \begin{vmatrix} S_2 & S_1 & S_0 \\ S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \end{vmatrix} = S_0(S_3^2 + S_2S_4) + S_1(S_2S_3 + S_1S_4) + S_2(S_2^2 + S_1S_3) = 0.$$

Since $S_1S_3 + S_2^2 = 0$ and $S_1S_4 + S_2S_3 = 0$ by assumption, $\Delta_3 = S_0(S_3^2 + S_2S_4)$. Also, since $S_1S_3 = S_2^2$ and $S_1S_4 = S_2S_3$, then $S_3 = S_2^2/S_1$ and $S_3 = S_1S_4/S_2$ (unless $S_2 = S_3 = S_4 = 0$). Therefore

$$S_3^2 = \frac{S_2^2}{S_1} \cdot \frac{S_1S_4}{S_2} = S_2S_4, \quad \text{and} \quad \Delta_3 = 0.$$

Property 3: Let $\Delta = S_2^2 + S_1S_3 \neq 0$, $\sigma_1 = (S_2S_3 + S_1S_4)/\Delta$, and $\sigma_2 = (S_2S_4 + S_3^2)/\Delta$ (σ_1 and σ_2 when $e = 2$). If $D_1 \triangleq S_2 + \sigma_1S_1 + \sigma_2S_0 = 0$ or $D_2 \triangleq S_5 + \sigma_1S_4 + \sigma_2S_3 = 0$, then $e = 2$, or $e > 3$.

Proof: Since $\Delta \neq 0$, $e \neq 0$, and $e \neq 1$ [2], and it is sufficient to show that $e \neq 3$. Since the determinant

$$\Delta_3 = \begin{vmatrix} S_2 & S_1 & S_0 \\ S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \end{vmatrix} = S_0(S_3^2 + S_2S_4) + S_1(S_2S_3 + S_1S_4) + S_2(S_2^2 + S_1S_3)$$

$$= S_0\sigma_2\Delta + S_1\sigma_1\Delta + S_2\Delta = \Delta \cdot D_1 = 0,$$

it follows that $e \neq 3$ [2]. Similarly, we can show that if $D_2 = 0$, $e \neq 3$.

Now let us consider the cases when the errors can occur at any location. Keep in mind that an error at location n can only affect S_0 and an error at location n+1 can only affect S_5.

Property 4: If $S_0 \neq 0$ and $S_i = 0$, $1 \leq i \leq 5$, then $E = e' = 1$ with error location at position n, or $E > 3$.

Proof: Since $S_0 \neq 0$, $E \neq 0$ and we need only show that $E \neq 2$ and $E \neq 3$. From Case 4), we see that if $E = e' = 1$ with error location at position n, then $S_0 \neq 0$ and $S_i = 0$, $1 \leq i \leq 5$. $E \neq 2$ and $E \neq 3$ follow from (21). Similar arguments can be used to obtain the next two properties.

Property 5: If $S_5 \neq 0$ and $S_i = 0$, $0 \leq i \leq 4$, then $E = e' = 1$ with error location at position n+1, or $E > 3$.

Property 6: If $S_0 \neq 0$, $S_5 \neq 0$, and $S_i = 0$, $1 \leq i \leq 4$, then $E = e' = 2$ with two error locations at positions n and n+1, respectively, or $E > 3$.

Property 7: Assume $S_1 \neq 0$, $S_3S_1 + S_2^2 = 0$, and $S_1S_4 + S_2S_3 = 0$, and let $\sigma_1 = S_2/S_1$ (σ_1 when e = 1), $T_1 \triangleq S_1 + \sigma_1 S_0$, and $T_2 \triangleq S_5 + \sigma_1 S_4$. Then

(1) If $T_1 = 0$ and $T_2 = 0$, then $E = e = 1$, or $E > 3$.

(2) If $T_1 \neq 0$ and $T_2 = 0$, then $E = 2$, $e = 1$, and $e' = 1$, with two error locations at positions j, $0 \leq j \leq n-1$, and n, respectively, or $E > 3$.

(3) If $T_1 = 0$ and $T_2 \neq 0$, then $E = 2$, $e = 1$, and $e' = 1$ with error locations at positionsj, $0 \leq j \leq n-1$, and n+1, respectively, or $E > 3$.

(4) If $T_1 \neq 0$ and $T_2 \neq 0$, then $E = 3$, $e = 1$, and $e' = 2$ with error locations at j, $0 \leq j \leq n-1$, n and n+1, respectively, or $E > 3$.

Proof: By Property 2, (1), (2), (3), and (4) follow from Case 1), Case 7), Case 8), and Case 9), respectively.

Property 8: Let $\Delta = S_2^2 + S_1S_3 \neq 0$, $\sigma_1 = (S_2S_3 + S_1S_4)/\Delta$, $\sigma_2 = (S_2S_4 + S_3^2)/\Delta$ (σ_1 and σ_2 when $e = 2$), $D_1 = S_2 + \sigma_1S_1 + \sigma_2S_0$, and $D_2 = S_5 + \sigma_1S_4 + \sigma_2S_3$. Then

(1) If $D_1 = 0$ and $D_2 = 0$, then $E = e = 2$ or $E > 3$.

(2) If $D_1 \neq 0$ and $D_2 = 0$, then $E = 3$, $e = 2$, and $e' = 1$ with error locations at positions j, k, $0 \leq j < k \leq n-1$, and n, respectively, or $E > 3$.

(3) If $D_1 = 0$ and $D_2 \neq 0$, then $E = 3$, $e = 2$, and $e' = 1$ with error locations at positions j, k, $0 \leq j < k \leq n-1$, and $n+1$, respectively, or $E > 3$.

(4) If $D_1 \neq 0$ and $D_2 \neq 0$, then $E \geq e \geq 3$.

Proof: By Property 3, (1), (2), and (3) follow from Case 2), Case 10), and Case 11), respectively. (4) follows from Property 3, and (1), (2), and (3).

Property 9: If $S_i \neq 0$ and $S_j = 0$ for some i and j such that $i \neq j$ and $1 \leq i$, $j \leq 4$, then $E \geq e \geq 2$.

Proof: First note that $e \neq 0$. If $e = 1$, then $S_i = e_jX_j^i \neq 0$, for $1 \leq i \leq 4$. Since some $S_i = 0$, $e \neq 1$. Hence $e \geq 2$ and $E \geq e \geq 2$.

Property 10: If $S_1 = S_2 = 0$ and $S_3 \neq 0$, then $E \geq 3$.

Proof: Property 10 follows from Properties 1 and 9

Property 11: If $S_0 = S_1 = S_2 = 0$ and $S_3 \neq 0$, then $E > 3$.

Proof: From Property 10, $E \geq 3$, and we need only show that $E \neq 3$. It is sufficient to show that

$$\Delta_3 = \begin{vmatrix} S_2 & S_1 & S_0 \\ S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \end{vmatrix} = 0.$$

Since $\Delta_3 = 0$ by assumption, $e \neq 3$ and $E \neq 3$.

Property 12: If $S_1 = S_2 = S_3 = 0$ and $S_4 \neq 0$, then $E > 3$.

Proof: Since S_i, $1 \leq i \leq 4$, is not affected by errors at locations n and n+1, if $S_1 = S_2 = S_3 = 0$ and $S_4 \neq 0$, it is easy to see that equations (10), (11), and (12) are not valid. Therefore $e \neq 1$, $e \neq 2$, and $e \neq 3$ and we must have $E \geq e > 3$.

Based on the properties presented above, the procedure shown in Fig. 1 can be used to calculate the coefficients of the error-location polynomial (if $e \neq 0$) and to find the error locations (if $e' \neq 0$).

(1) If $S_1 \neq 0$, then $E \geq e \geq 1$, and the next step is (2). If $S_1 = 0$ and $S_2 \neq 0$, then $E \geq e \geq 2$, and the next step is (3). If $S_1 = S_2 = 0$ and $S_3 \neq 0$, then $E \geq e \geq 3$ and the next step is (7). If $S_1 = S_2 = S_3 = 0$ and $S_4 \neq 0$, then $E > 3$ (detected errors). If $S_i = 0$, $1 \leq i \leq 4$, then $E \geq 0$ and $e = 0$, and the next step is (8).

(2) $S_1 \neq 0$. Set $\sigma = S_2/S_1$. If $S_3 + \sigma S_2 \neq 0$, then $E \geq e \geq 2$ and the next step is (3). If $S_3 + \sigma S_2 = 0$ and $S_4 + \sigma S_3 \neq 0$, then $E \geq e \geq 3$ and the next step is (5). If $S_3 + \sigma S_2 = 0$ and $S_4 + \sigma S_3 = 0$, then $E \geq e = 1$, and the next step is (6).

(3) $\Delta = S_2^2 + S_1 S_3 \neq 0$. Set $\sigma_1 = (S_2 S_3 + S_1 S_4)/\Delta$, $\sigma_2 = (S_2 S_4 + S_3^2)/\Delta$, $D_1 = S_2 + \sigma_1 S_1 + \sigma_2 S_0$, and $D_2 = S_5 + \sigma_1 S_4 + \sigma_2 S_3$.

If $D_1 = D_2 = 0$, then $E = e = 2$.

If $D_1 \neq 0$ and $D_2 = 0$, then $E = 3$, $e = 2$ and $e' = 1$, with one error location at position n.

If $D_1 = 0$ and $D_2 \neq 0$, then $E = 3$, $e = 2$, and $e' = 1$, with one error location at position n+1.

If $D_1 \neq 0$ and $D_2 \neq 0$, then $E = e = 3$ and the next step is (4).

(4) Set $\sigma_3' = D_2/D_1$, $\sigma_2' = \sigma_2 + \sigma_3'(S_1S_2 + S_0S_3)/\Delta$, and $\sigma_1' = \sigma_1 + \sigma_3'(S_0S_2 + S_1^2)/\Delta$.

(5) Set $\Delta' = S_2S_3 + S_1S_4$ and $D_1' = S_0(S_3^2 + S_2S_4) + S_1\Delta'$.

If $D_1 = 0$, then $E > 3$ (detected errors).

If $D_1 \neq 0$, then $E = e = 3$, and we set

$$\sigma_3 = [(S_3^2 + S_2S_4)S_3 + \Delta'S_4]/D_1', \quad \sigma_1 = (S_2S_4 + S_1S_5)/\Delta', \text{ and}$$

$$\sigma_2 = (\sigma_3S_0 + \sigma_1S_2 + S_3)/S_1.$$

(6) Set $T_1 = S_1 + \sigma S_0$ and $T_2 = S_5 + \sigma S_4$.

If $T_1 = 0$ and $T_2 = 0$, then $E = e = 1$.

If $T_1 \neq 0$ and $T_2 = 0$, then $E = 2$, $e = 1$, and $e' = 1$, with one error location at position n.

If $T_1 = 0$ and $T_2 \neq 0$, then $E = 2$, $e = 1$, and $e' = 1$, with one error location at position n+1.

If $T_1 \neq 0$ and $T_2 \neq 0$, then $E = 3$, $e = 1$, and $e' = 2$, with two error locations at positions n and n+1, respectively.

(7) $S_1 = S_2 = 0$, $S_3 \neq 0$.

If $S_0 = 0$, then $E > 3$ (detected errors).

If $S_0 \neq 0$, then $E = e = 3$, and we set

$$\sigma_1 = S_4/S_3, \ \sigma_2 = (S_5 + \sigma_1 S_4)/S_3, \text{ and } \sigma_3 = S_3/S_0.$$

(8) $S_1 = S_2 = S_3 = S_4 = 0$.

If $S_0 = S_5 = 0$, then $E = 0$ (no errors).

If $S_0 \neq 0$ and $S_5 = 0$, then $E = e' = 1$, with the error location at position n.

If $S_0 = 0$ and $S_5 \neq 0$, then $E = e' = 1$, with the error location at position n+1.

If $S_0 \neq 0$ and $S_5 \neq 0$, then $E = e' = 2$, with the two error locations at positions n and n+1, respectively.

An overall diagram for the decoding of extended TBEC Reed-Solomon codes is shown in Fig. 2. The symbols read from the storage media are stored in the data buffer while decoding proceeds. In the first step of decoding, syndromes S_0, S_1, S_2, S_3, S_4, and S_5 are generated. The syndromes are then stored in the syndrome buffer. The OR gates at the output of the syndrome buffer determine if errors exist. If errors exist, the decoder starts the error correction procedure. The error-location polynomial calculator calculates the error-location polynomial using the procedure shown in Fig. 1. The error-location calculator then finds the roots of the error-location polynomial. This can be done by using the methods described in References 7 and 8. Finally the error values calculator determines the value of the errors at each error location. The error locations and the error values are used to modify the symbols at the output of the data buffer.

IV. CONCLUSION

In this report we have presented a decoding scheme for extended triple-byte-error-correcting Reed-Solomon codes. The scheme is especially applicable to computer systems where high speed decoding is the basic requirement.

REFERENCES

[1] C.L. Chen and M.Y. Hsiao, "High Speed Decoding of Reed-Solomon Codes", U.S. Patent 4, 142, 174, February 27, 1979.

[2] W.W. Peterson and E.J. Weldon, Jr., Error-Correcting Codes, second ed. Cambridge, MA: MIT Press, 1972.

[3] S. Lin and D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, New Jersey, 1983.

[4] T. Kasami, S. Lin, and W.W. Peterson, "Some Results on Weight Distributions of BCH Codes", IEEE Trans. Inf. Theory, IT-12(2), p. 274, April 1966.

[5] T. Kasami, S. Lin, and W.W. Peterson, "Some Results on Cyclic Codes Which Are Invariant Under the Affine Group", Scientific Report AFCRL-66-62, Air Force Cambridge Research Labs., Bedford, MA, 1966.

[6] J.K. Wolf, "Adding Two Information Symbols to Certain Nonbinary BCH Codes and Some Applications", Bell Syst. Tech. J., 48, pp. 2405-2424, 1969.

[7] C.L. Chen, "Formulas for the Solutions of Quadratic Equations", IEEE Trans. Inform. Theory, IT-28, pp. 792-794, September 1982.

[8] L.R. Welch, "Fast Algorithms for Finding Roots of Cubic Polynomials over Finite Fields", Presented at 1985 IEEE International Symposium on Information Theory, June 1985, Brighton, England.

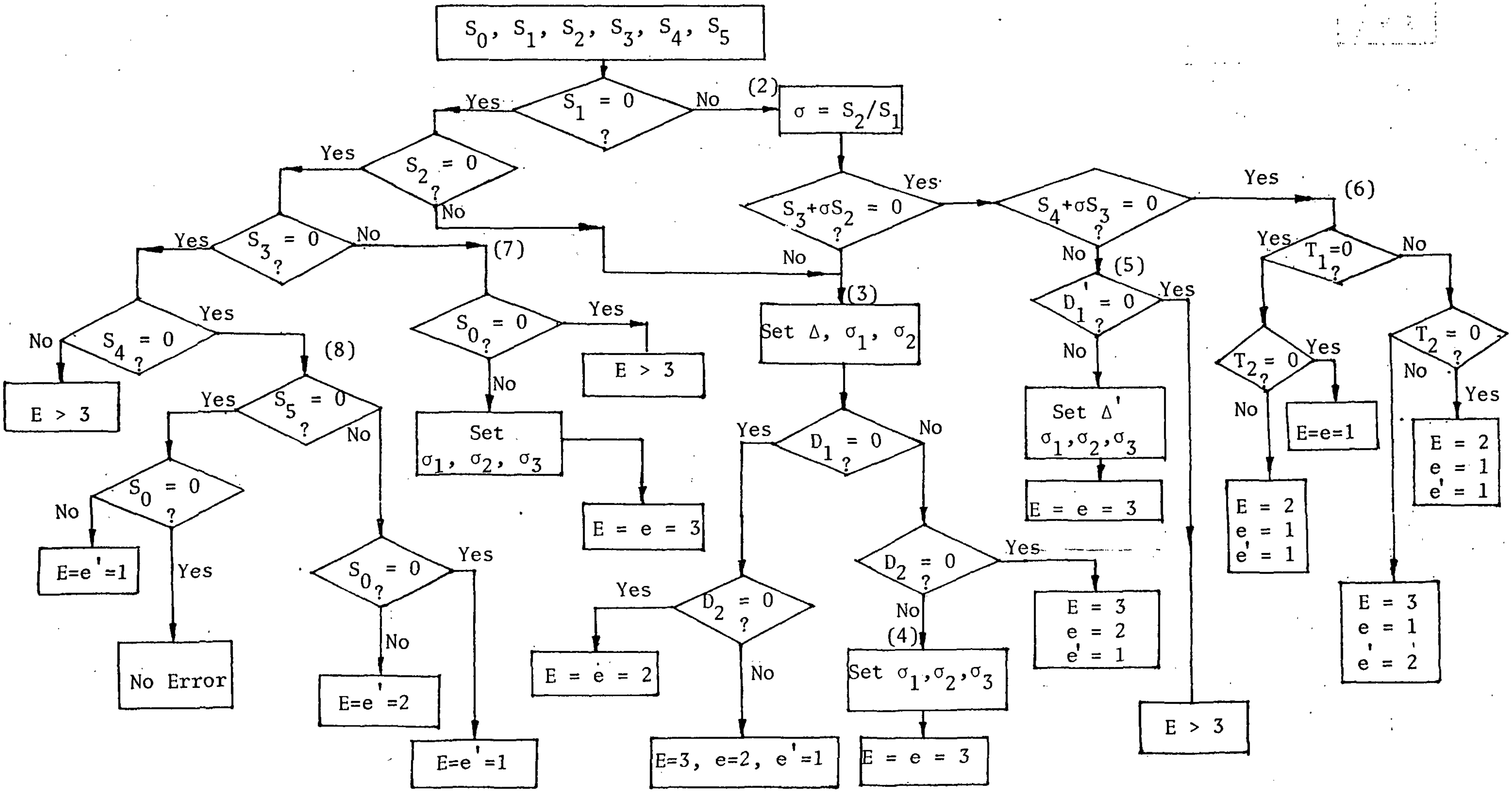

$S_0, S_1, S_2, S_3, S_4, S_5$
$S_1 = 0$?
Yes
No
(2)
$\sigma = S_2/S_1$
$S_2 = 0$?
Yes
No
$S_3+\sigma S_2 = 0$?
Yes
No
$S_4+\sigma S_3 = 0$?
Yes
No
(6)
$S_3 = 0$?
Yes
No
(7)
(3)
Set Δ, σ_1, σ_2
(5)
$D_1' = 0$?
Yes
No
$T_1=0$?
Yes
No
$S_4 = 0$?
Yes
No
$E > 3$
(8)
$S_0 = 0$?
Yes
No
$E > 3$
Set σ_1, σ_2, σ_3
$E = e = 3$
$D_1 = 0$?
Yes
No
Set Δ' $\sigma_1, \sigma_2, \sigma_3$
$E = e = 3$
$T_2 = 0$?
Yes
No
$E=e=1$
$E = 2$ $e = 1$ $e' = 1$
$T_2 = 0$?
Yes
No
$E = 2$ $e = 1$ $e' = 1$
$S_5 = 0$?
Yes
No
$S_0 = 0$?
Yes
No
$E=e'=1$
No Error
$S_0 = 0$?
Yes
No
$E=e'=2$
$E=e'=1$
$D_2 = 0$?
Yes
No
$E = e = 2$
$E=3$, $e=2$, $e'=1$
$D_2 = 0$?
Yes
No
(4)
Set $\sigma_1, \sigma_2, \sigma_3$
$E = e = 3$
$E = 3$ $e = 2$ $e' = 1$
$E > 3$
$E = 3$ $e = 1$ $e' = 2$

Figure 1.

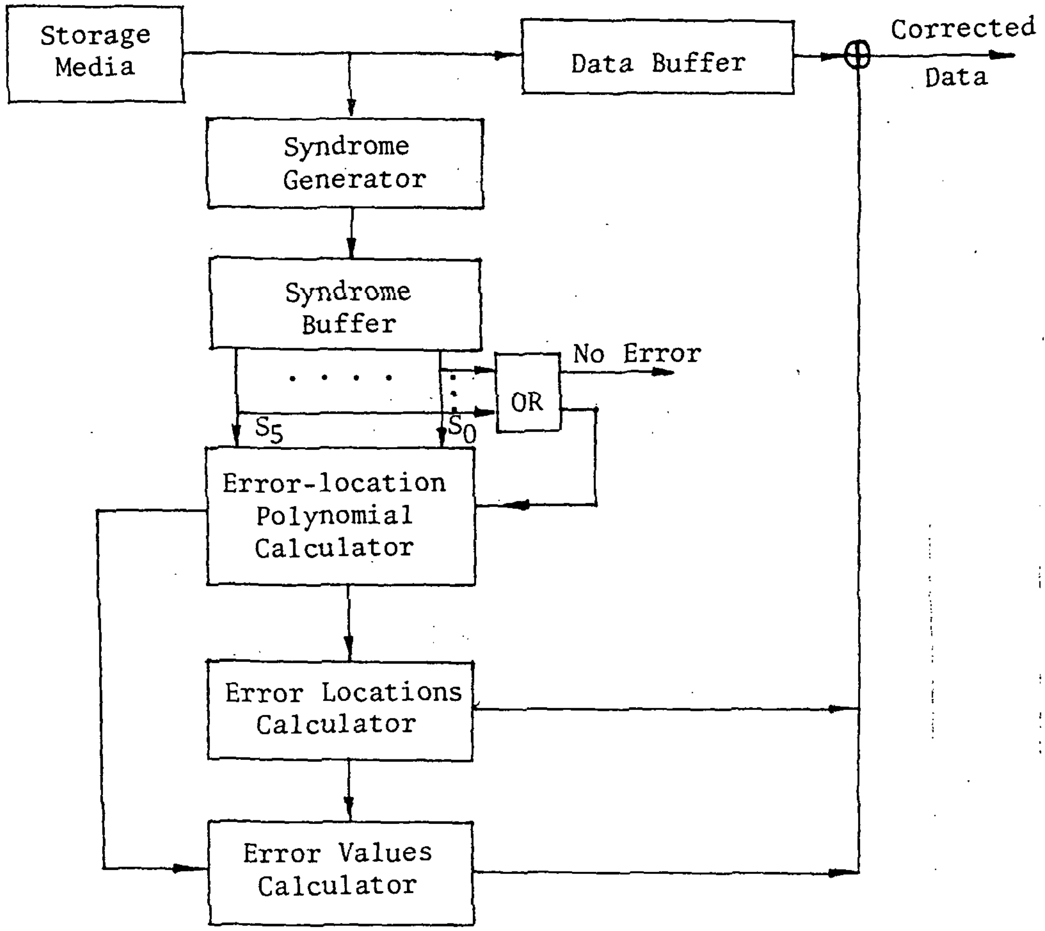

Storage Media
Data Buffer
Corrected Data
Syndrome Generator
Syndrome Buffer
No Error
OR
S_5
S_0
Error-location Polynomial Calculator
Error Locations Calculator
Error Values Calculator

Figure 2.

tional information symbols. The d_{min} = 6 code is capable of double-byte-error-correction (DBEC) and triple-byte-error-detection (TBED) and can be extended to include two additional information symbols. The decoding method applies to the extended codes with only slight modification.

Code efficiency is high since only three parity symbols are used in the d_{min} = 4 code and only five in the d_{min} = 6 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.