DRA

(NASA-CR-166549) BCH CODES FOR LARGE IC RANDOM-ACCESS MEMORY SYSTEMS Final Report (Hawaii Univ., Honolulu.) 28 p
HC A03/MF A01 CSCL 09B
N84-27460
Unclas
G3/61 13636

BCH Codes for Large IC Random-Access Memory Systems

Shu Lin
Department of Electrical Engineering
University of Hawaii
Honolulu, Hawaii 96822

Daniel J. Costello, Jr.
Department of Electrical Engineering
Illinois Institute of Technology
Chicago, Illinois 60616

Final Report

Phase I

NASA Grant NAG 2-202

June, 1983

1. Introduction

The Bose, Chaudhuri and Hocquenghem (BCH) codes form a large class of random-error correcting cyclic codes [1-4]. For any positive integers m ($m \geq 3$) and t ($t < 2^{m-1}$), there exists a binary t-error-correcting BCH code of length $n = 2^m - 1$ and no more than mt parity-check bits. BCH codes or shortened BCH codes are widely used for error control in data storage and communication systems. In this report, we present some shortened BCH codes for possible applications to large IC random-access memory systems. These codes are given by their parity-check matrices. Encoding and decoding of these codes are discussed.

2. Encoding and Decoding of Linear Block Codes

An (n, k) linear block code is specified by either a k×n generator matrix G or an (n-k)×k parity-check matrix H. In systematic form, the generator and parity-check matrices have the following forms:

$$G = [P\ I_k]$$

$$= \left[\underbrace{\begin{matrix} b_{00} & b_{01} & \cdots & b_{0,n-k-1} \\ b_{10} & b_{11} & \cdots & b_{1,n-k-1} \\ \vdots & & & \vdots \\ b_{k-1,0} & b_{k-1,1} & \cdots & b_{k-1,n-k-1} \end{matrix}}_{P} \quad \underbrace{\begin{matrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{matrix}}_{I_k} \right] \qquad (1)$$

and

$$H = [I_{n-k}P^T]$$

$$= \left[\underbrace{\begin{matrix} 1 & 0 & 0 & \cdot\cdot\cdot & 0 \\ 0 & 1 & 0 & \cdot\cdot\cdot & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ 0 & 0 & 0 & \cdot\cdot\cdot & 1 \end{matrix}}_{I_{n-k}} \; \underbrace{\begin{matrix} b_{00} & b_{10} & \cdot\cdot\cdot & b_{k-1,0} \\ b_{01} & b_{11} & \cdot\cdot\cdot & b_{k-1,1} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ b_{0,n-k-1} & b_{1,n-k-1} & \cdot\cdot\cdot & b_{k-1,n-k-1} \end{matrix}}_{P^T} \right] \tag{2}$$

where P^T denotes the transpose of P. Encoding can be performed based on either the generator or the parity-check matrices. However, decoding (syndrome computation) is normally done based on the parity-check matrix. In some applications, such as applications to IC random-access memory systems, it is preferred that both encoding and decoding are based on the parity-check matrix.

Consider a systematic (n, k) code with parity-check matrix given by (2). Let $\overline{m} = (m_0, m_1, \cdots, m_{k-1})$ be the message to be encoded. The corresponding codeword is

$$\begin{aligned} \overline{v} &= (v_0, v_1, \cdots, v_{n-1}) \\ &= (v_0, v_1, \cdots, v_{n-k-1}, m_0, m_1, \cdots, m_{k-1}) \end{aligned} \tag{3}$$

where the k rightmost bits are identical to the k message bits and the n-k leftmost bits are the parity-check bits. The parity-check bits can be obtained from the parity-check matrix H by using the following theorem: A vector $\overline{v}$ is a codeword if and only if $\overline{v} \cdot H^T = \overline{0}$. From (2) and (3), the n-k parity-check bits are given by the following n-k parity-check equations:

$$\begin{aligned} v_0 &= m_0b_{00} + m_1b_{10} + \cdots + m_{k-1}b_{k-1,0} \\ v_1 &= m_0b_{01} + m_1b_{11} + \cdots + m_{k-1}b_{k-1,1} \\ &\cdot \\ &\cdot \\ &\cdot \\ v_{n-k-1} &= m_0b_{0,n-k-1} + m_1b_{1,n-k-1} + \cdots + m_{k-1}b_{k-1,n-k-1} \end{aligned} \tag{4}$$

where the coefficients b_{ij}'s are the entries of the parity-check matrix H. Hence, each parity bit is a linear sum of the message bits. An encoder which accepts k message bits in parallel and forms the n-k parity bits in parallel is shown in Figure 1.

Let $\bar{r} = (r_0, r_1, \cdots, r_{n-1})$ be the vector received from a communication system (or read from a memory system). Due to channel or memory noise, $\bar{r}$ may differ from the word $\bar{v}$ transmitted (or stored) and hence $\bar{r}$ may contain errors. The difference between the received word $\bar{r}$ and the transmitted word $\bar{v}$ is defined as the vector sum

$$\begin{aligned} \bar{e} &= (e_0, e_1, \cdots, e_{n-1}) \\ &= \bar{r} + \bar{v} \qquad (5) \\ &= (r_0 + v_0, r_1 + v_1, \cdots, r_{n-1} + v_{n-1}) \end{aligned}$$

where $r_i + v_i$ is the modulo-2 sum of r_i and v_i. We see that

$$e_i = \{ \begin{matrix} 0, & \text{if } r_i = v_i \\ 1, & \text{if } r_i \neq v_i \end{matrix}$$

The vector $\bar{e}$ is called the <u>error vector</u> (or <u>error pattern</u>), the ones in $\bar{e}$ indicate errors. From (5), we have

$$\bar{r} = \bar{v} + \bar{e} \ . \qquad (6)$$

The receiver does not know either $\bar{v}$ or $\bar{e}$. Upon receiving $\bar{r}$, the decoder must first determine whether $\bar{r}$ contains errors. If the presence of errors is detected, the decoder takes actions to locate and correct the errors.

Error detection is carried out by computing the <u>syndrome</u> of the received word $\bar{r}$ which is defined as follows:

$$\begin{aligned} \bar{s} &= (s_0, s_1, \cdots, s_{n-k-1}) \\ &= \bar{r} \cdot H^T \ . \end{aligned} \qquad (7)$$

If $\bar{s} = \bar{0}$, $\bar{r}$ is a codeword. In this case the decoder assumes that $\bar{r}$ is error-free and accepts it. If $\bar{s} \neq \bar{0}$, $\bar{r}$ is not a codeword and the presence of errors is detected. From (2) and (7), the n-k syndrome bits are given by the following n-k syndrome equations:

$$
\begin{aligned}
s_0 &= r_0 + r_{n-k} b_{00} + r_{n-k+1} b_{10} + \cdots + r_{n-1} b_{k-1,0} \\
s_1 &= r_1 + r_{n-k} b_{01} + r_{n-k+1} b_{11} + \cdots + r_{n-1} b_{k-1,1} \\
&\;\,\vdots \\
s_{n-k-1} &= r_{n-k-1} + r_{n-k} b_{0,n-k-1} + r_{n-k+1} b_{1,n-k-1} + \cdots + r_{n-1} b_{k-1,n-k-1}
\end{aligned}
\tag{8}
$$

From (8), we see that the syndrome $\bar{s}$ is simply the vector sum of the received parity bits $(r_0, r_1, \cdots, r_{n-k-1})$ and the parity bits recomputed from the received message bits $r_{n-k}, r_{n-k+1}, \cdots, r_{n-1}$. Therefore, the syndrome can be formed by a circuit similar to the encoding circuit. A syndrome circuit consisting of a replica of encoding circuit is shown in Figure 2.

Example 1: Consider the (7,4) linear code which is specified by the following parity-check matrix

$$
H = \left[\underbrace{\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}}_{I_3}\;\underbrace{\begin{matrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{matrix}}_{P^T}\right].
$$

The three parity-check bits are given by the following parity-check equations:

$$
\begin{aligned}
v_0 &= m_0 \phantom{{}+m_1} + m_2 + m_3 , \\
v_1 &= m_0 + m_1 + m_2 \phantom{{}+m_3} , \\
v_2 &= \phantom{m_0 + {}} m_1 + m_2 + m_3 .
\end{aligned}
$$

A parallel encoding circuit is shown in Figure 3. Let $\overline{r} = (r_0, r_1, r_2, r_3, r_4, r_5, r_6)$ be the vector received or read from a memory system. The bits b_0, b_1 and b_2 are the received parity bits; the bits r_3, r_4, r_5 and r_6 are the received message bits. The 3 syndrome bits are given by the following 3 syndrome equations:

$$\begin{array}{c} s_0 \\ s_1 \\ s_2 \end{array} \begin{array}{c} = \\ = \\ = \end{array} \underset{\substack{\uparrow \\ \text{Received} \\ \text{parity} \\ \text{bits}}}{\boxed{\begin{array}{c} r_0 \\ r_1 \\ r_2 \end{array}}} \begin{array}{c} + \\ + \\ \end{array} \underset{\substack{\uparrow \\ \text{Parity bits recomputed} \\ \text{from the received} \\ \text{message bits}}}{\boxed{\begin{array}{lllllll} r_3 & & & + & r_5 & + & r_6 \\ r_3 & + & r_4 & + & r_5 & & \\ & + & r_4 & + & r_5 & + & r_6 \end{array}}}$$

A syndrome circuit is shown in Figure 4.

There are 2^n possible error patterns. However, every (n, k) linear code is capable of correcting 2^{n-k} error patterns which are called the correctable error patterns. There exists a one-to-one correspondence between a correctable error pattern and an(n-k)-bit syndrome $\overline{s}$ [1-4]. A table can be set up to show this correspondence. The table consists of 2^{n-k} correctable error patterns and their corresponding syndromes as shown in Figure 5. This table can be used for decoding. The decoding consists of three steps:

Step 1. Compute the syndrome $\overline{s}$ of the received word $\overline{r}$,

$$\overline{s} = \overline{r} \cdot H^T.$$

Step 2. From the table, determine the error pattern $\overline{e}$ which corresponds to the syndrome computed in Step 1. Then $\overline{e}$ is assumed to be the error pattern caused by the noise.

Step 3. Decode the received word $\overline{r}$ into the codeword $\overline{v} = \overline{r} + \overline{e}$.

The above decoding scheme is called table-lookup decoding.

The association of the syndrome to an error pattern can be implemented with either a combinational logic circuit or a read-only memory (ROM). A general decoder based on the table-lookup scheme is shown in Figure 6. The table-lookup decoder is fast in decoding speed, however its complexity grows exponentially with n-k (or with the number of error patterns to be corrected, 2^{n-k} of them). For large n-k, this decoder becomes impractical. However, if n-k is not too large and if we do not intend to correct all the 2^{n-k} correctable error patterns, the table-lookup decoding can be implemented practically.

If a (n, k) linear code with minimum distance d is used for random error correction, then all the error patterns with $t = \lfloor\frac{d-1}{2}\rfloor$ or fewer errors are correctable, i.e., the code is capable of correcting t or fewer errors in the received word [1-4]. The number of these error patterns is

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t},$$

which is in general much smaller than 2^{n-k} for large n-k. However these are the error patterns which are most likely to occur. If we only intend to correct these most probable error patterns, we may set up a decoding table which only shows the correspondence between these error patterns and their syndromes. The decoding is then carried out as follows:

<u>Step 1</u>. Compute the syndrome $\bar{s}$ of the received word $\bar{r}$.

<u>Step 2</u>. Check whether the syndrome $\bar{s}$ corresponds to an error pattern of t or fewer errors.

<u>Step 3</u>. If the syndrome $\bar{s}$ corresponds to an error pattern $\bar{e}$ of t or fewer errors, then the received word $\bar{r}$ is decoded into the codeword $\bar{v} = \bar{r} + \bar{e}$.

<u>Step 4</u>. If the syndrome $\bar{s}$ does not correspond to an error pattern of t or fewer errors, errors are detected. In this case, either a retransmission or a re-read from the memory system is requested.

For moderate n and small t (say $t = 1\sim5$), the above modified table-lookup decoding can be practically implemented and results in a fast decoder which is desired in large IC random-access memory systems.

3. BCH Codes

For any positive integers $m(m \geq 3)$ and $t(t < 2^{m-1})$, there exists a binary BCH code with the following parameters:

Length: $n = 2^m - 1$,

Number of parity bits: $n - k \leq mt$,

Minimum distance: $d = 2t + 1$.

This code is capable of correcting all the error patterns of t or fewer errors, and is called a t-error-correcting BCH code. The code is cyclic and is uniquely specified by a generator polynomial $\overline{g}(x)$ of degree n-k [1-4]. Let $\overline{v} = (v_0, v_1, \cdots, v_{n-1})$ be a binary vector. Let $\overline{v}(x) = v_0 + v_1 x + \cdots + v_{n-1} x^{n-1}$ be a binary polynomial corresponding to $\overline{v}$. Clearly $\overline{v}(x)$ is a polynomial of degree n-1 or less. For a cyclic code with generator polynomial $\overline{g}(x)$, a vector $\overline{v}$ is a codeword if and only if its corresponding polynomial $\overline{v}(x)$ is divisible by $\overline{g}(x)$, i.e., a multiple of $\overline{g}(x)$.

Let $GF(2^m)$ be a Galois field of 2^m elements. Let α be a primitive element in $GF(2^m)$. Then the generator polynomial $\overline{g}(x)$ of a binary primitive t-error-correcting BCH code of length $n = 2^m - 1$ is the lowest-degree polynomial with binary coefficients which has

$$\alpha, \alpha^2, \cdots, \alpha^{2t}$$

as roots, i.e., $g(\alpha^i) = 0$ for $i = 1, 2, \cdots, 2t$. Generator polynomials of binary primitive BCH codes of length up to $n = 1023$ are given by Lin and Costello [4].

Example 2: For $m = 7$ and $t = 2$, there exists a double-error-correcting BCH code of length $n = 2^7 - 1 = 127$ and 14 parity bits. Hence it is a (127,113) code. Its generator polynomial is

$$\bar{g}(x) = x^{14} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1.$$

Encoding of a BCH code is normally performed in serial manner using a shift register with feedback connections based on its generator polynomial. However in some applications, parallel encoding is preferred. For parallel encoding, we need to determine the parity-check matrix H. Dividing x^{n-k+i} by the generator polynomial $\bar{g}(x)$ for i = 0, 1, 2, ⋯, k-1, we obtain

$$x^{n-k+i} = \bar{a}_i(x)\bar{g}(x) + \bar{b}_i(x) ,$$

where $\bar{b}_i(x)$ is the remainder with the following form

$$\bar{b}_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i,n-k-1}x^{n-k-1} .$$

Then the parity-check matrix in systematic form is given below:

$$H = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & b_{00} & b_{10} & \cdots & b_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & b_{01} & b_{11} & \cdots & b_{k-1,1} \\ \cdot & & & & \cdot & \cdot & & & \cdot \\ \cdot & & & & \cdot & \cdot & & & \cdot \\ \cdot & & & & \cdot & \cdot & & & \cdot \\ 0 & 0 & 0 & \cdots & 1 & b_{0,n-k-1} & b_{1,n-k-1} & \cdots & b_{k-1,n-k-1} \end{bmatrix}$$

Example 3: For m = 4 and t = 2, there exists a (15,7) double-error-correcting BCH code with generator polynomial

$$\bar{g}(x) = x^8 + x^7 + x^6 + x^4 + 1.$$

Dividing x^{8+i} by $\bar{g}(x)$ for i = 0, 1, ⋯ , 6, we obtain

$$\bar{b}_0(x) = 1 + x^4 + x^6 + x^7,$$

$$\bar{b}_1(x) = 1 + x + x^4 + x^5 + x^6,$$

$$\bar{b}_2(x) = x + x^2 + x^5 + x^6 + x^7,$$

$$\bar{b}_3(x) = 1 + x^2 + x^3 + x^4,$$

$$\bar{b}_4(x) = x + x^3 + x^4 + x^5,$$

$$\bar{b}_5(x) = x^2 + x^4 + x^5 + x^6,$$
$$\bar{b}_6(x) = x^3 + x^5 + x^6 + x^7.$$

The parity-check matrix is given by

$$H = \begin{bmatrix} 1&0&0&0&0&0&0&0&1&1&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0&0&1&1&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1 \\ 0&0&0&0&1&0&0&0&1&1&0&1&1&1&0 \\ 0&0&0&0&0&1&0&0&0&1&1&0&1&1&1 \\ 0&0&0&0&0&0&1&0&1&1&1&0&0&1&1 \\ 0&0&0&0&0&0&0&1&1&0&1&0&0&0&1 \end{bmatrix}$$

In system design, if a code of suitable natural length n or suitable number k of message digits cannot be found, it may be desirable to shorten a code to meet the requirements. Let C be an (n, k) linear block code with parity-check matrix $H = [I_{n-k}\ P^T]$,,where P^T is an (n-k) matrix. If we delete ℓ columns from P^T with $0 \leq \ell \leq k$, we obtain an $(n-k)x(n-\ell)$ parity-check matrix $H_\ell = [I_{n-k}\ P_\ell^{\ T}]$. This matrix H_ℓ generates an $(n-\ell, k-\ell)$ linear code which is called a <u>shortened code</u> of C. Any shortened code of C has <u>at least</u> the same error-correcting capability as the original code C [1-4].

4. Shortened BCH Codes for Table Look-Up Decoding

In Table 1, we give a list of 8 shortened BCH codes which have been constructed for fast syndrome computation and table look-up decoding. Four of these codes have $d_{min} = 6$, while the other four have $d_{min} = 8$. For all but the (45, 32) code with $d_{min} = 6$ and the (86, 64) code with $d_{min} = 8$, the maximum number of 1's in any row of the H matrix is either equal to or slightly less than a power of 2. This minimizes the number of logic levels needed to compute the syndrome, assuming a two-input exclusive-or gate tree-like implementation. In addition, the number of 1's in each row of the H matrix is either equal to, or nearly equal to, the average number. This facilitates a fast parallel computation of the syndrome bits. Although we have not done an exhaustive search, we feel that the codes listed in Table 1 are nearly optimal with respect to minimizing the total number of 1's in the H matrix.

The construction procedure followed was essentially a trial-and-error approach. A summary description of the construction procedure for the $d_{min} = 6$ codes now follows.

Consider the (127, 113) $d_{min} = 5$ BCH code, which has generator polynomial $\overline{p}(x) = (1 + x^3 + x^7)(1 + x + x^2 + x^3 + x^7)$. Let $\overline{g}(x) = (1 + x)\overline{p}(x) = 1 + x^3 + x^4 + x^7 + x^8 + x^{10} + x^{14} + x^{15}$. Then $\overline{g}(x)$ generates a (127, 112) $d_{min} = 6$ code. Dividing $x^{n-k+i} = x^{15+i}$ by $\overline{g}(x)$ for $i = 0, 1, 2, \cdots, 111$, we obtain

$$x^{15+i} = \overline{a}_i(x) + \overline{b}_i(x)$$

where the remainder $\overline{b}_i(x)$ has the following form:

$$\overline{b}_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i,14}x^{14} .$$

Then the parity-check matrix for the (127, 112) $d_{min} = 6$ code is given by:

$$H = [I_{15\times15}\ P^T] = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & b_{00} & b_{10} & b_{20} & \cdots & b_{111,0} \\ 0 & 1 & 0 & \cdots & 0 & b_{01} & b_{11} & b_{21} & \cdots & b_{111,1} \\ 0 & 0 & 1 & \cdots & 0 & b_{02} & b_{12} & b_{22} & \cdots & b_{111,2} \\ & \cdot & & & & \cdot & & & & \cdot \\ & \cdot & & & & \cdot & & & & \cdot \\ & \cdot & & & & \cdot & & & & \cdot \\ 0 & 0 & 0 & \cdots & 1 & b_{0,14} & b_{1,14} & b_{2,14} & \cdots & b_{111,14} \end{bmatrix}$$

By deleting an appropriate set of 48 columns from the H matrix above, we obtained a 15×79 matrix H_1, which is the parity-check matrix of a (79, 64) d_{min} = 6 linear code. The matrix H_1 is shown in Fig. 7. (In order to conserve space, the matrix is given in octal notation.) Let $w(h_i)$ denote the number of 1's in the ith row of the matrix H_1. From Fig. 7 we see that:

$$w(h_0) = 30, \quad w(h_1) = 30, \quad w(h_2) = 30, \quad w(h_3) = 31, \quad w(h_4) = 31$$

$$w(h_5) = 30, \quad w(h_6) = 31, \quad w(h_7) = 30, \quad w(h_8) = 30, \quad w(h_9) = 30$$

$$w(h_{10}) = 30, \; w(h_{11}) = 30, \; w(h_{12}) = 30, \; w(h_{13}) = 30, \; w(h_{14}) = 30$$

and $w(h_i) < 2^5 = 32.$

By deleting 32 columns from the matrix H_1, we obtained a 15×47 matrix H_2, which is the parity-check matrix of a (47, 32) d_{min} = 6 linear code. The matrix H_2 is shown in Fig. 8. From Fig. 8 we see that:

$$w(h_0) = 15, \quad w(h_1) = 15, \quad w(h_2) = 15, \quad w(h_3) = 13, \quad w(h_4) = 15$$

$$w(h_5) = 15, \quad w(h_6) = 15, \quad w(h_7) = 15, \quad w(h_8) = 14, \quad w(h_9) = 15$$

$$w(h_{10}) = 15, \; w(h_{11}) = 14, \; w(h_{12}) = 15, \; w(h_{13}) = 15, \; w(h_{14}) = 15$$

and $w(h_i) < 2^4 = 16.$

Deleting 16 columns from H_2 results in a 15×31 matrix H_3, which is the parity-check matrix of a (31, 16) d_{min} = 6 linear code, and is shown in Fig. 9. From Fig. 9 we see that:

$$w(h_0) = 7, \quad w(h_1) = 7, \quad w(h_2) = 8, \quad w(h_3) = 7, \quad w(h_4) = 6$$

$$w(h_5) = 7, \quad w(h_6) = 7, \quad w(h_7) = 7, \quad w(h_8) = 8, \quad w(h_9) = 7$$

$$w(h_{10}) = 7, \; w(h_{11}) = 8, \; w(h_{12}) = 7, \; w(h_{13}) = 7, \; w(h_{14}) = 7$$

and $w(h_i) \leq 2^3 = 8.$

Note that every column in the matrices H_1, H_2 and H_3 contains an odd number of 1's.

We also constructed a (45, 32) d_{min} = 6 code from the (63, 51) d_{min} = 5 BCH code, whose generator polynomial is given by $\overline{p}(x) = (1 + x + x^6)(1 + x +$

$x^2 + x^4 + x^6$), by multiplying $\overline{p}(x)$ by $(x + 1)$ and then following the same procedure described above. The number of 1's in some rows of the parity-check matrix H_4 obtained in this case exceeds $2^4 = 16$, however. The parity-check matrix H_4 of the (45, 32) $d_{min} = 6$ code is shown in Fig. 10. From Fig. 10 we see that:

$$w(h_0) = 17, \quad w(h_1) = 18, \quad w(h_2) = 16, \quad w(h_3) = 17, \; w(h_4) = 17$$
$$w(h_5) = 18, \quad w(h_6) = 17, \quad w(h_7) = 16, \quad w(h_8) = 18, \; w(h_9) = 17$$
$$w(h_{10}) = 16, \; w(h_{11}) = 18, \; w(h_{12}) = 16 \; .$$

The construction procedure for the $d_{min} = 8$ codes is similar to that described above for the $d_{min} = 6$ codes. The parity-check matrices are shown in figures 11-14.

The most efficient $d_{min} = 6$ code in terms of minimizing the number of parity-check bits is the (45, 32) code. This code is capable of correcting all double error patterns and detecting all triple error patterns. A computer analysis of all weight 4 error patterns has been performed for this code. We have found that out of $\binom{45}{4} = 148{,}995$ weight 4 error patterns, only 28,485 are undetectable, i.e., they have the same syndrome as a correctable error pattern. Hence

$$1 - \frac{28{,}485}{148{,}995} = 1 - .19118 = 80.882\%$$

of the weight 4 error patterns are detectable for this code. We have also included as an Appendix to this report a 38 page computer printout of the decoding table for this code. Listed are the syndromes and their corresponding coset leaders for the

$$\binom{45}{1} + \binom{45}{2} = 45 + 990 = 1035$$

correctable error patterns. The remaining syndromes for the detectable error patterns are not listed.

REFERENCES

1. E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

2. W.W. Peterson and E.J. Weldon, Jr., Error-Correcting Codes, Second Edition, MIT Press, Cambridge, Massachusetts, 1972.

3. F.J. MacWilliams and N.J.A. Sloane, The Theory of Error-Correcting Codes, North-Holland, Amsterdam, 1977.

4. S. Lin and D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, New Jersey, 1982.

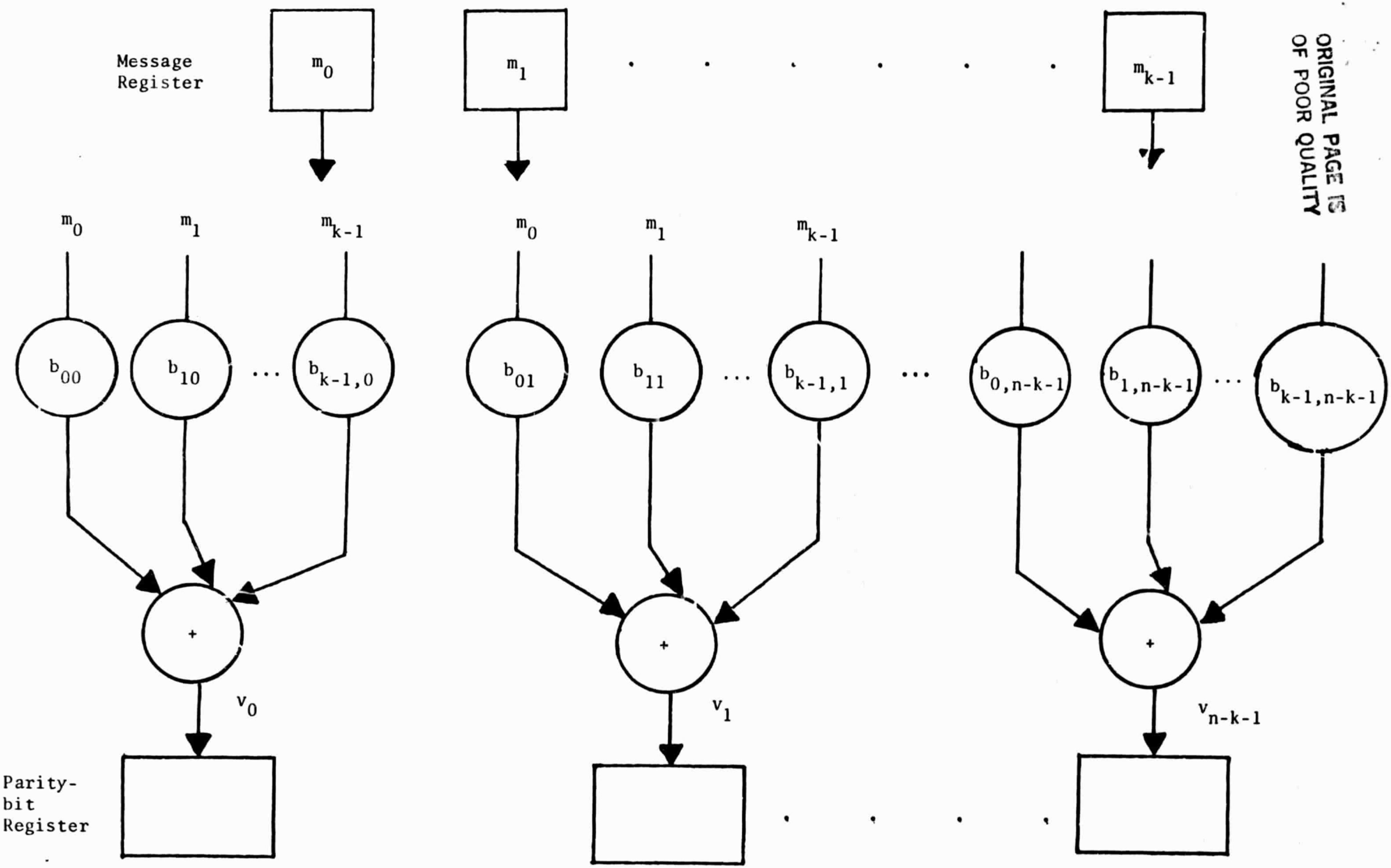

Fig. 1 A Parallel Encoding Circuit for an (n, k) Linear Systematic Code

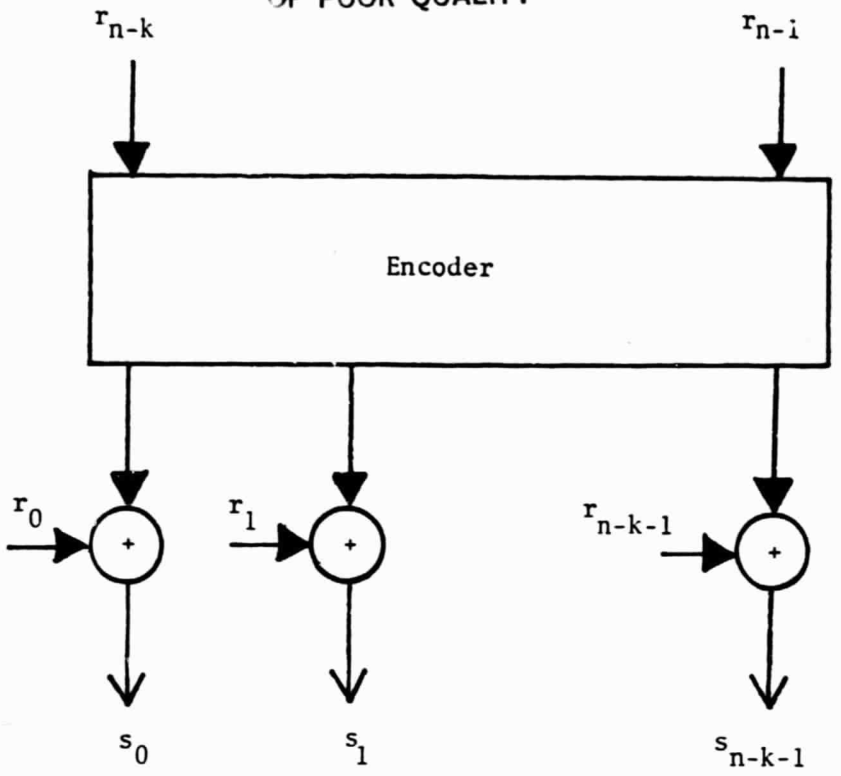

Fig. 2 A Syndrome Circuit for an (n, k) Linear Systematic Code

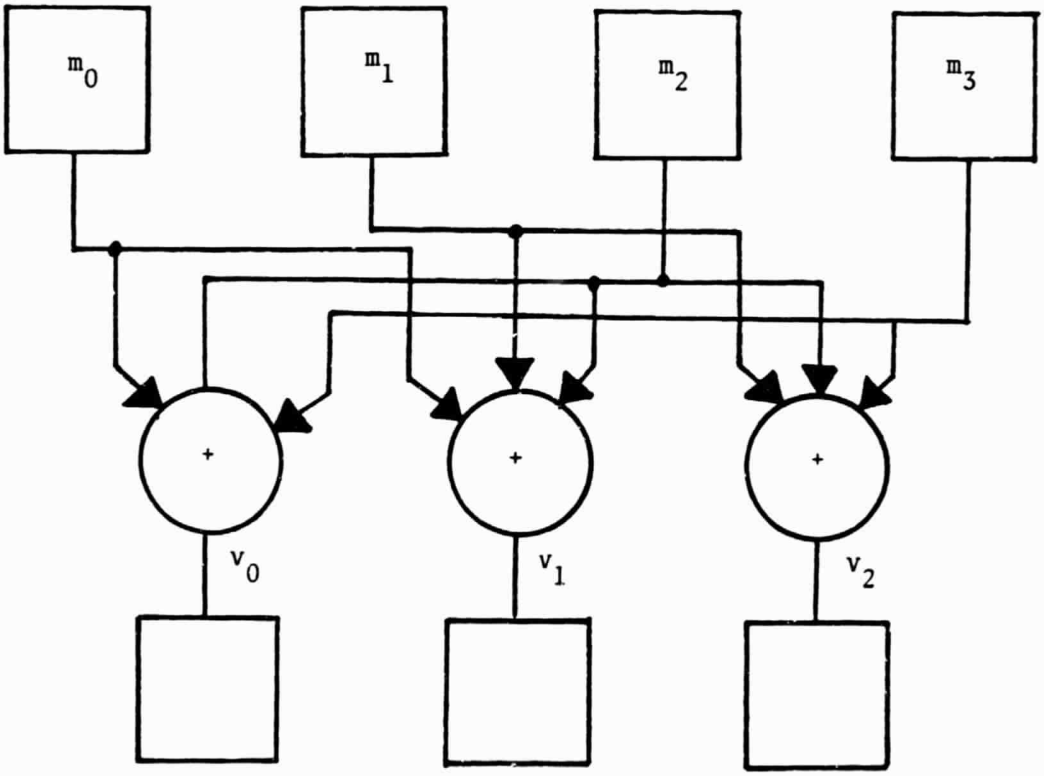

Figure 3

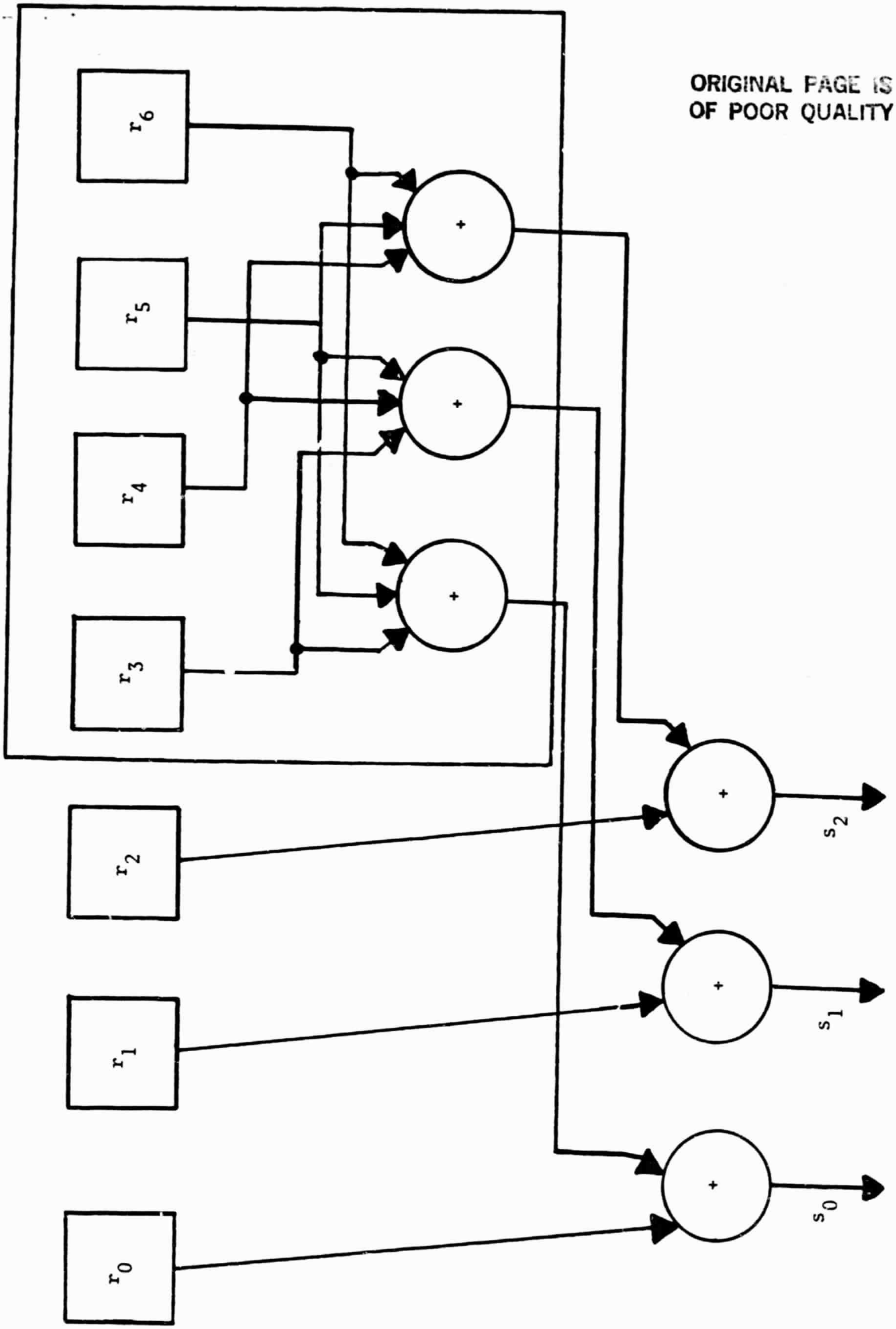

Figure 4

$$\bar{s}_1 \longleftrightarrow \bar{e}_1$$

$$\bar{s}_2 \longleftrightarrow \bar{e}_2$$

$$\vdots$$

$$\bar{s}_{2^{n-k}} \longleftrightarrow \bar{e}_{2^{n-k}}$$

Fig. 5 A Table-Lookup Decoding Table

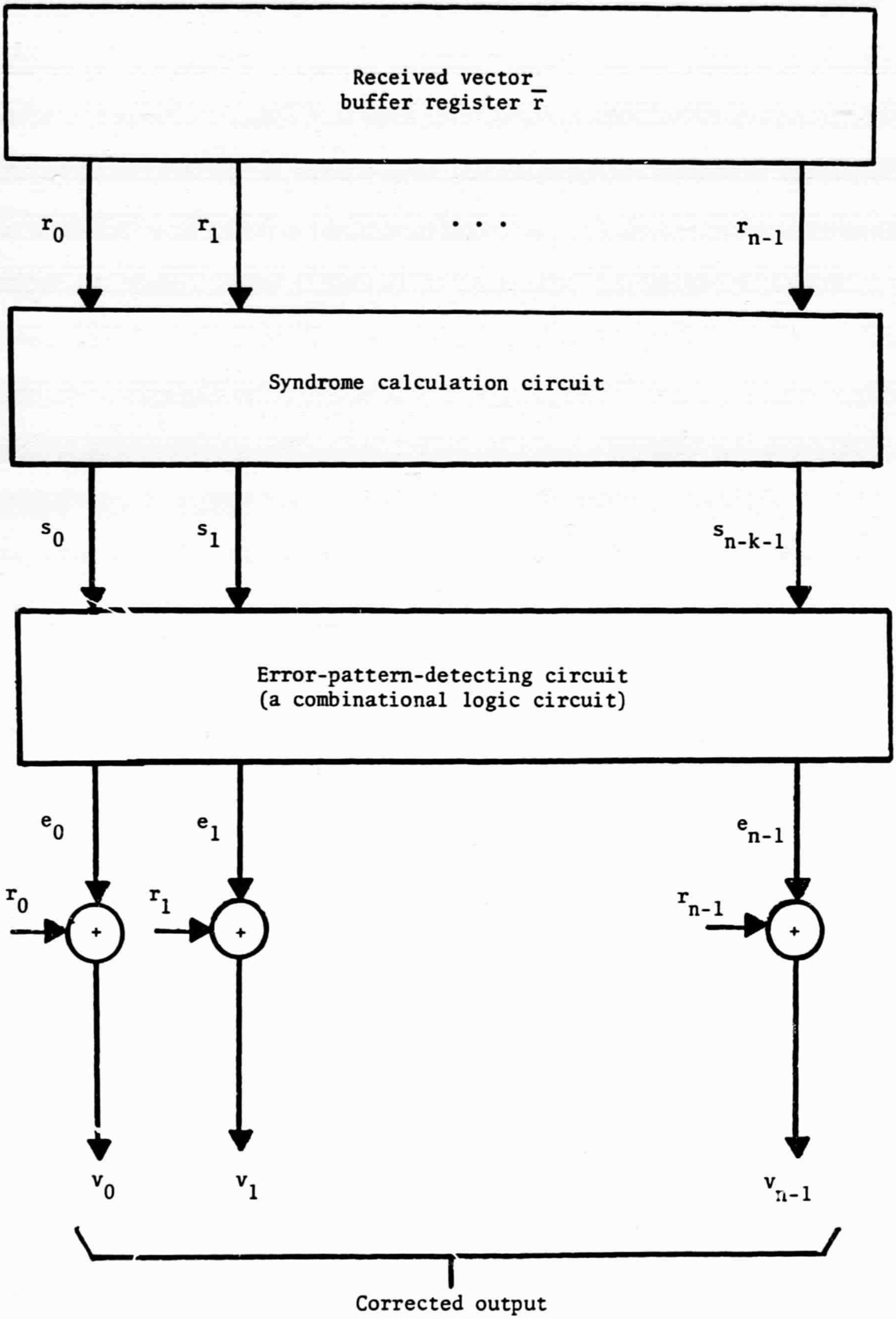

Fig. 6 General decoder for a linear block code

TABLE 1.

PARAMETERS OF A LIST OF SHORTENED BCH CODES

n	k	n-k	d_{min}	Total Number of 1's in H	Average Number of 1's per row	Maximum Number of 1's per row
79	64	15	6	453	30.2	31
47	32	15	6	221	14.7	15
31	16	15	6	107	7.1	8
45	32	13	6	221	17	18
35	16	19	8	133	7	8
51	32	19	8	263	13.8	16
86	64	22	8	683	31	34
89	64	25	8	675	27	30

$$H_1 = \left[\begin{array}{cccccccccccccccccccccccc}
 & 6 & 4 & 5 & 0 & 2 & 4 & 3 & 0 & 2 & 0 & 4 & 7 & 4 & 6 & 1 & 7 & 1 & 5 & 6 & 5 & 2 & 0 \\
 & 3 & 2 & 3 & 0 & 0 & 2 & 1 & 4 & 3 & 0 & 6 & 2 & 6 & 3 & 0 & 3 & 4 & 6 & 5 & 3 & 7 & 0 \\
 & 1 & 5 & 0 & 4 & 1 & 1 & 0 & 6 & 1 & 4 & 3 & 0 & 3 & 1 & 4 & 5 & 6 & 7 & 6 & 4 & 5 & 4 \\
 & 4 & 2 & 0 & 6 & 2 & 4 & 7 & 3 & 0 & 6 & 5 & 2 & 5 & 2 & 7 & 0 & 6 & 2 & 5 & 6 & 0 & 4 \\
 & 6 & 1 & 4 & 7 & 3 & 2 & 0 & 5 & 6 & 3 & 2 & 2 & 6 & 3 & 2 & 7 & 2 & 0 & 4 & 2 & 2 & 0 \\
 & 1 & 0 & 7 & 7 & 5 & 1 & 0 & 2 & 7 & 1 & 5 & 1 & 3 & 1 & 5 & 2 & 5 & 0 & 4 & 0 & 3 & 0 \\
 & 0 & 0 & 2 & 7 & 7 & 4 & 4 & 1 & 3 & 4 & 6 & 5 & 5 & 4 & 6 & 5 & 2 & 4 & 2 & 1 & 1 & 4 \\
I_{15\times15} & 4 & 0 & 4 & 3 & 5 & 2 & 1 & 0 & 5 & 6 & 3 & 5 & 2 & 0 & 2 & 0 & 4 & 7 & 5 & 5 & 6 & 4 \\
 & 4 & 0 & 7 & 5 & 5 & 1 & 3 & 4 & 2 & 7 & 1 & 0 & 1 & 6 & 0 & 3 & 3 & 6 & 0 & 2 & 5 & 0 \\
 & 2 & 0 & 3 & 6 & 7 & 4 & 5 & 6 & 1 & 3 & 4 & 4 & 0 & 7 & 0 & 4 & 5 & 3 & 2 & 1 & 0 & 4 \\
 & 5 & 4 & 4 & 7 & 0 & 2 & 1 & 7 & 2 & 5 & 2 & 4 & 4 & 5 & 5 & 4 & 3 & 0 & 5 & 4 & 4 & 0 \\
 & 0 & 6 & 2 & 7 & 4 & 5 & 0 & 7 & 5 & 2 & 1 & 3 & 2 & 2 & 6 & 2 & 1 & 0 & 2 & 7 & 2 & 0 \\
 & 0 & 7 & 0 & 3 & 7 & 2 & 4 & 3 & 4 & 5 & 4 & 4 & 5 & 1 & 3 & 1 & 0 & 0 & 3 & 2 & 7 & 0 \\
 & 2 & 3 & 5 & 1 & 7 & 1 & 2 & 1 & 4 & 2 & 2 & 3 & 2 & 4 & 5 & 0 & 4 & 4 & 3 & 5 & 1 & 4 \\
 & 5 & 1 & 3 & 0 & 4 & 0 & 6 & 0 & 4 & 1 & 1 & 7 & 1 & 4 & 3 & 6 & 3 & 3 & 5 & 2 & 4 & 4
\end{array}\right]$$

Fig. 7 Parity-check matrix of a (79,64) $d_{min} = 6$ code

$$H_2 = \left[\begin{array}{ccccccccccccc}
 & & 5 & 1 & 1 & 0 & 4 & 3 & 0 & 4 & 5 & 7 & 2 \\
 & & 6 & 4 & 4 & 4 & 2 & 1 & 4 & 6 & 2 & 3 & 4 \\
 & & 3 & 2 & 0 & 2 & 2 & 0 & 6 & 3 & 2 & 5 & 6 \\
 & & 0 & 4 & 1 & 1 & 5 & 2 & 3 & 5 & 1 & 0 & 4 \\
 & & 4 & 3 & 0 & 4 & 3 & 7 & 1 & 2 & 2 & 6 & 0 \\
 & & 2 & 1 & 6 & 2 & 0 & 7 & 4 & 5 & 3 & 2 & 0 \\
 & & 0 & 0 & 7 & 1 & 1 & 3 & 6 & 6 & 4 & 5 & 0 \\
 I_{15\times15} & & 0 & 1 & 2 & 4 & 0 & 6 & 7 & 3 & 4 & 1 & 6 \\
 & & 0 & 1 & 6 & 2 & 6 & 1 & 3 & 1 & 0 & 3 & 4 \\
 & & 4 & 0 & 7 & 1 & 2 & 0 & 5 & 4 & 6 & 4 & 6 \\
 & & 3 & 1 & 0 & 4 & 3 & 3 & 2 & 2 & 7 & 4 & 0 \\
 & & 1 & 4 & 7 & 2 & 3 & 4 & 5 & 1 & 0 & 2 & 0 \\
 & & 1 & 6 & 2 & 5 & 1 & 6 & 2 & 4 & 7 & 0 & 0 \\
 & & 4 & 7 & 2 & 2 & 5 & 6 & 1 & 2 & 1 & 1 & 0 \\
 & & 2 & 2 & 6 & 1 & 4 & 4 & 0 & 1 & 5 & 6 & 6
\end{array}\right]$$

Fig. 8 Parity-check matrix of a (47,32) $d_{min} = 6$ code

$$H_3 = \left[\begin{array}{ccccccc}
 & 2 & 2 & 4 & 3 & 0 & 4 \\
 & 5 & 1 & 2 & 1 & 4 & 0 \\
 & 6 & 4 & 1 & 0 & 6 & 4 \\
 & 1 & 0 & 4 & 6 & 3 & 0 \\
 & 0 & 6 & 2 & 1 & 1 & 0 \\
 & 4 & 3 & 1 & 1 & 4 & 0 \\
 & 0 & 1 & 4 & 5 & 6 & 0 \\
I_{15\times15} & 0 & 2 & 2 & 0 & 7 & 4 \\
 & 0 & 3 & 1 & 3 & 3 & 0 \\
 & 0 & 1 & 4 & 4 & 5 & 4 \\
 & 6 & 2 & 2 & 1 & 2 & 0 \\
 & 3 & 1 & 5 & 0 & 5 & 0 \\
 & 3 & 4 & 2 & 4 & 2 & 0 \\
 & 1 & 6 & 1 & 2 & 1 & 0 \\
 & 4 & 5 & 0 & 6 & 0 & 4
\end{array}\right]$$

Fig. 9 Parity-check matrix of a (31,16) $d_{min} = 6$ code

$$H_4 = \left[\begin{array}{ccccccccccccc} & 6 & 4 & 7 & 4 & 3 & 2 & 1 & 0 & 3 & 4 & 6 \\ & 4 & 6 & 6 & 2 & 3 & 7 & 1 & 4 & 2 & 3 & 4 \\ & 2 & 3 & 1 & 1 & 1 & 5 & 4 & 6 & 1 & 1 & 6 \\ & 6 & 5 & 3 & 0 & 6 & 4 & 5 & 3 & 3 & 1 & 0 \\ & 3 & 2 & 7 & 4 & 2 & 2 & 2 & 5 & 5 & 1 & 4 \\ & 0 & 5 & 3 & 6 & 1 & 3 & 3 & 2 & 6 & 4 & 6 \\ I_{13\times13} & 7 & 6 & 0 & 3 & 2 & 7 & 4 & 5 & 0 & 2 & 4 \\ & 3 & 7 & 2 & 1 & 4 & 3 & 4 & 2 & 4 & 1 & 2 \\ & 7 & 3 & 0 & 4 & 4 & 3 & 7 & 1 & 1 & 5 & 2 \\ & 4 & 1 & 1 & 6 & 1 & 1 & 6 & 4 & 7 & 3 & 2 \\ & 5 & 4 & 1 & 3 & 3 & 4 & 6 & 2 & 0 & 5 & 2 \\ & 5 & 2 & 5 & 1 & 7 & 4 & 2 & 1 & 3 & 6 & 2 \\ & 5 & 1 & 7 & 0 & 4 & 4 & 2 & 0 & 6 & 3 & 6 \end{array} \right]$$

Fig. 10 Parity-check matrix of a (45,32) $d_{min} = 6$ code

$$H_5 = \left[\begin{array}{ccccccc}
 & 1 & 3 & 0 & 5 & 0 & 4 \\
 & 0 & 7 & 4 & 1 & 5 & 0 \\
 & 4 & 1 & 6 & 0 & 6 & 4 \\
 & 2 & 2 & 7 & 2 & 2 & 0 \\
 & 0 & 0 & 3 & 3 & 1 & 4 \\
 & 0 & 0 & 1 & 4 & 4 & 4 \\
 & 5 & 1 & 0 & 2 & 2 & 4 \\
 & 2 & 6 & 4 & 1 & 1 & 0 \\
I_{19\times19} & 4 & 2 & 2 & 4 & 5 & 0 \\
 & 3 & 0 & 1 & 6 & 3 & 0 \\
 & 4 & 5 & 0 & 2 & 1 & 0 \\
 & 6 & 0 & 4 & 0 & 0 & 4 \\
 & 7 & 2 & 2 & 3 & 0 & 0 \\
 & 3 & 5 & 1 & 2 & 4 & 0 \\
 & 1 & 4 & 4 & 5 & 2 & 0 \\
 & 1 & 5 & 2 & 4 & 4 & 4 \\
 & 0 & 4 & 5 & 1 & 2 & 0 \\
 & 4 & 0 & 2 & 6 & 4 & 0 \\
 & 2 & 2 & 1 & 2 & 3 & 0
\end{array}\right]$$

Fig. 11 Parity-check matrix of a (35,16) $d_{min} = 8$ code

$$H_6 = \left[\begin{array}{cccccccccccc}
 & 4 & 2 & 1 & 5 & 4 & 2 & 7 & 4 & 4 & 2 & 0 \\
 & 2 & 1 & 0 & 7 & 6 & 1 & 2 & 6 & 6 & 5 & 0 \\
 & 1 & 0 & 4 & 6 & 7 & 0 & 7 & 2 & 3 & 2 & 4 \\
 & 0 & 4 & 2 & 3 & 3 & 4 & 1 & 5 & 1 & 1 & 2 \\
 & 4 & 0 & 0 & 6 & 1 & 4 & 0 & 3 & 4 & 6 & 4 \\
 & 2 & 0 & 0 & 4 & 0 & 6 & 2 & 0 & 2 & 3 & 2 \\
 & 5 & 2 & 1 & 4 & 4 & 1 & 3 & 5 & 1 & 3 & 4 \\
 & 2 & 5 & 0 & 5 & 2 & 0 & 3 & 6 & 4 & 5 & 6 \\
 & 5 & 0 & 5 & 1 & 1 & 2 & 2 & 2 & 2 & 4 & 6 \\
I_{19\times19} & 6 & 6 & 3 & 2 & 0 & 7 & 0 & 5 & 1 & 4 & 2 \\
 & 7 & 1 & 0 & 6 & 4 & 1 & 0 & 7 & 0 & 4 & 0 \\
 & 3 & 4 & 4 & 0 & 2 & 0 & 6 & 2 & 0 & 2 & 0 \\
 & 1 & 6 & 2 & 1 & 1 & 0 & 0 & 1 & 4 & 1 & 0 \\
 & 0 & 7 & 1 & 2 & 4 & 4 & 7 & 1 & 2 & 0 & 4 \\
 & 0 & 3 & 4 & 2 & 2 & 2 & 0 & 4 & 5 & 0 & 2 \\
 & 4 & 3 & 7 & 0 & 5 & 3 & 4 & 6 & 2 & 2 & 0 \\
 & 2 & 1 & 7 & 6 & 2 & 5 & 2 & 2 & 5 & 1 & 0 \\
 & 1 & 0 & 7 & 6 & 1 & 2 & 6 & 1 & 2 & 0 & 4 \\
 & 0 & 4 & 3 & 7 & 0 & 5 & 5 & 1 & 1 & 4 & 2
\end{array}\right]$$

Fig. 12 Parity-check matrix of a (51,32) d_{min} = 8 code

Fig. 13 Parity-check matrix of a (86,64) $d_{min} = 8$ code

$$H_7 = \left[\begin{array}{cccccccccccccccccccccccc}
 & 5 & 6 & 7 & 0 & 5 & 7 & 0 & 6 & 3 & 3 & 2 & 0 & 6 & 7 & 0 & 0 & 1 & 6 & 0 & 4 & 5 & 0 \\
 & 2 & 7 & 3 & 4 & 2 & 7 & 4 & 1 & 1 & 7 & 7 & 0 & 3 & 1 & 4 & 0 & 0 & 3 & 0 & 1 & 2 & 4 \\
 & 6 & 5 & 2 & 6 & 4 & 0 & 6 & 5 & 7 & 4 & 5 & 4 & 3 & 0 & 6 & 0 & 1 & 3 & 4 & 2 & 2 & 0 \\
 & 1 & 2 & 5 & 3 & 2 & 0 & 3 & 2 & 7 & 6 & 0 & 6 & 1 & 4 & 5 & 0 & 0 & 5 & 6 & 1 & 1 & 0 \\
 & 2 & 5 & 2 & 5 & 5 & 4 & 1 & 4 & 3 & 5 & 0 & 3 & 4 & 5 & 1 & 4 & 0 & 2 & 7 & 5 & 6 & 4 \\
 & 6 & 4 & 2 & 2 & 3 & 5 & 0 & 3 & 2 & 5 & 6 & 1 & 4 & 5 & 4 & 6 & 1 & 3 & 3 & 4 & 0 & 0 \\
 & 3 & 2 & 1 & 1 & 1 & 7 & 4 & 3 & 5 & 2 & 5 & 0 & 2 & 3 & 6 & 3 & 0 & 5 & 5 & 3 & 2 & 0 \\
 & 3 & 5 & 0 & 4 & 4 & 2 & 6 & 0 & 6 & 5 & 2 & 4 & 1 & 3 & 7 & 1 & 4 & 6 & 6 & 6 & 5 & 0 \\
 & 3 & 6 & 4 & 2 & 2 & 5 & 3 & 2 & 3 & 0 & 5 & 2 & 4 & 4 & 7 & 4 & 6 & 7 & 3 & 5 & 0 & 4 \\
 & 4 & 1 & 5 & 1 & 4 & 0 & 5 & 3 & 2 & 7 & 0 & 5 & 4 & 6 & 3 & 6 & 2 & 1 & 5 & 3 & 1 & 0 \\
 & 2 & 0 & 6 & 4 & 6 & 5 & 2 & 4 & 5 & 3 & 4 & 2 & 2 & 2 & 1 & 7 & 1 & 0 & 6 & 7 & 4 & 4 \\
I_{22\times22} & 6 & 6 & 4 & 2 & 6 & 1 & 5 & 4 & 1 & 4 & 6 & 1 & 3 & 4 & 0 & 7 & 5 & 2 & 3 & 1 & 3 & 0 \\
 & 1 & 3 & 2 & 1 & 3 & 5 & 6 & 7 & 0 & 4 & 3 & 0 & 5 & 4 & 0 & 3 & 6 & 5 & 1 & 6 & 7 & 4 \\
 & 7 & 3 & 2 & 0 & 0 & 1 & 7 & 7 & 7 & 3 & 3 & 4 & 0 & 1 & 0 & 1 & 6 & 0 & 4 & 4 & 4 & 4 \\
 & 4 & 3 & 2 & 0 & 5 & 2 & 7 & 0 & 4 & 4 & 7 & 6 & 2 & 4 & 4 & 0 & 6 & 6 & 2 & 6 & 7 & 0 \\
 & 0 & 1 & 5 & 0 & 2 & 5 & 3 & 7 & 2 & 0 & 3 & 7 & 5 & 0 & 2 & 0 & 3 & 3 & 1 & 3 & 3 & 4 \\
 & 5 & 6 & 1 & 4 & 4 & 0 & 5 & 1 & 6 & 1 & 3 & 7 & 0 & 0 & 1 & 0 & 0 & 7 & 4 & 7 & 0 & 4 \\
 & 5 & 1 & 7 & 6 & 7 & 6 & 2 & 0 & 4 & 1 & 7 & 7 & 6 & 6 & 0 & 4 & 1 & 1 & 6 & 4 & 1 & 0 \\
 & 0 & 4 & 7 & 7 & 3 & 3 & 1 & 3 & 2 & 2 & 5 & 7 & 3 & 0 & 0 & 2 & 0 & 4 & 7 & 6 & 0 & 4 \\
 & 5 & 4 & 4 & 7 & 0 & 2 & 4 & 2 & 6 & 2 & 0 & 7 & 7 & 0 & 0 & 1 & 1 & 0 & 3 & 6 & 7 & 0 \\
 & 0 & 6 & 2 & 3 & 4 & 0 & 2 & 3 & 3 & 3 & 0 & 3 & 7 & 4 & 0 & 0 & 4 & 4 & 1 & 2 & 3 & 4 \\
 & 7 & 5 & 6 & 1 & 3 & 6 & 1 & 6 & 6 & 6 & 4 & 1 & 5 & 2 & 0 & 0 & 3 & 4 & 0 & 3 & 6 & 4
\end{array}\right]$$

$$H_8 = \left[\begin{array}{cccccccccccccccccccccccc}
 & 7 & 1 & 2 & 7 & 0 & 7 & 7 & 4 & 4 & 0 & 1 & 3 & 2 & 5 & 6 & 0 & 7 & 0 & 2 & 0 & 4 & 0 \\
 & 0 & 4 & 7 & 0 & 4 & 0 & 2 & 2 & 6 & 0 & 4 & 6 & 5 & 3 & 3 & 0 & 6 & 4 & 1 & 2 & 6 & 4 \\
 & 4 & 3 & 1 & 4 & 2 & 7 & 0 & 5 & 7 & 0 & 2 & 0 & 2 & 0 & 1 & 4 & 2 & 2 & 2 & 7 & 5 & 0 \\
 & 0 & 0 & 6 & \text{[illegible]} & 1 & 2 & 3 & 6 & 3 & 4 & 4 & 1 & 3 & 5 & 4 & 6 & 3 & 1 & 1 & 3 & 4 & 0 \\
 & 0 & 0 & 1 & 4 & 4 & 2 & 4 & 3 & 5 & 6 & 6 & 3 & 5 & 3 & 0 & 3 & 6 & 4 & 4 & 7 & 2 & 0 \\
 & 7 & 1 & 0 & 2 & 2 & 7 & 6 & 1 & 6 & 7 & 2 & 1 & 4 & 0 & 6 & 1 & 1 & 2 & 0 & 1 & 5 & 4 \\
 & 0 & 4 & 6 & 5 & 1 & 0 & 1 & 4 & 3 & 3 & 1 & 1 & 4 & 0 & 1 & 0 & 5 & 5 & 0 & 2 & 0 & 4 \\
 & 0 & 3 & 1 & 1 & 4 & 1 & 0 & 2 & 5 & 5 & 1 & 5 & 6 & 0 & 2 & 4 & 0 & 6 & 4 & 1 & 0 & 4 \\
 & 0 & 0 & 4 & 1 & 6 & 2 & 1 & 1 & 2 & 6 & 0 & 4 & 7 & 4 & 3 & 2 & 0 & 3 & 2 & 2 & 0 & 4 \\
 & 3 & 1 & 0 & 3 & 7 & 4 & 0 & 0 & 1 & 3 & 0 & 1 & 3 & 6 & 3 & 5 & 4 & 1 & 5 & 3 & 0 & 4 \\
 & 0 & 4 & 4 & 2 & 7 & 2 & 1 & 0 & 0 & 5 & 5 & 2 & 5 & 3 & 3 & 6 & 2 & 0 & 6 & 5 & 0 & 4 \\
 & 2 & 2 & 2 & 4 & 3 & 1 & 7 & 4 & 0 & 2 & 3 & 5 & 0 & 5 & 1 & 7 & 2 & 0 & 1 & 2 & 4 & 4 \\
 & 0 & 1 & 1 & 5 & 1 & 0 & 4 & 6 & 0 & 1 & 0 & 6 & 4 & 6 & 4 & 7 & 1 & 0 & 0 & 7 & 4 & 0 \\
I_{25\times 25} & 5 & 1 & 6 & 0 & 4 & 3 & 6 & 7 & 4 & 0 & 0 & 2 & 0 & 6 & 4 & 3 & 3 & 4 & 0 & 1 & 6 & 0 \\
 & 7 & 4 & 5 & 3 & 2 & 6 & 5 & 7 & 2 & 0 & 5 & 2 & 0 & 2 & 4 & 1 & 1 & 6 & 2 & 0 & 1 & 4 \\
 & 4 & 6 & 0 & 0 & 5 & 0 & 2 & 3 & 1 & 0 & 7 & 4 & 0 & 1 & 4 & 0 & 1 & 7 & 1 & 2 & 2 & 0 \\
 & 2 & 3 & 0 & 6 & 2 & 4 & 4 & 1 & 4 & 4 & 2 & 4 & 0 & 5 & 0 & 0 & 0 & 7 & 4 & 5 & 1 & 0 \\
 & 3 & 0 & 4 & 1 & 1 & 4 & 1 & 0 & 6 & 2 & 4 & 2 & 2 & 3 & 6 & 0 & 6 & 3 & 4 & 0 & 0 & 0 \\
 & 2 & 4 & 0 & 1 & 4 & 5 & 6 & 0 & 7 & 1 & 2 & 0 & 3 & 5 & 1 & 0 & 4 & 1 & 6 & 0 & 0 & 0 \\
 & 7 & 3 & 2 & 0 & 6 & 1 & 5 & 4 & 7 & 4 & 0 & 1 & 1 & 2 & 4 & 4 & 4 & 0 & 7 & 0 & 6 & 0 \\
 & 4 & 5 & 5 & 6 & 3 & 0 & 2 & 6 & 3 & 6 & 1 & 0 & 4 & 0 & 2 & 2 & 7 & 0 & 3 & 4 & 3 & 0 \\
 & 5 & 2 & 6 & 1 & 1 & 4 & 0 & 3 & 1 & 7 & 4 & 4 & 2 & 4 & 1 & 1 & 4 & 4 & 1 & 4 & 3 & 4 \\
 & 1 & 4 & 1 & 0 & 4 & 5 & 1 & 5 & 0 & 7 & 6 & 1 & 3 & 6 & 4 & 4 & 0 & 2 & 2 & 6 & 7 & 0 \\
 & 6 & 7 & 2 & 4 & 2 & 7 & 2 & 2 & 0 & 3 & 3 & 1 & 7 & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 7 & 0 \\
 & 6 & 2 & 5 & 6 & 1 & 3 & 7 & 1 & 0 & 1 & 0 & 6 & 5 & 1 & 4 & 1 & 6 & 0 & 4 & 4 & 1 & 4
\end{array}\right]$$

Fig. 14 Parity-check matrix of a (89,64) d_{min} = 8 code

1. Introduction

The Bose, Chaudhuri and Hocquenghem (BCH) codes form a large class of random-error correcting cyclic codes [1-4]. For any positive integers m ($m \geq 3$) and t ($t < 2^{m-1}$), there exists a binary t-error-correcting BCH code of length $n = 2^m - 1$ and no more than mt parity-check bits. BCH codes or shortened BCH codes are widely used for error control in data storage and communication systems. In this report, we present some shortened BCH codes for possible applications to large IC random-access memory systems. These codes are given by their parity-check matrices. Encoding and decoding of these codes are discussed.

2. Encoding and Decoding of Linear Block Codes

An (n, k) linear block code is specified by either a k×n generator matrix G or an (n-k)×k parity-check matrix H. In systematic form, the generator and parity-check matrices have the following forms:

$$G = [P\ I_k]$$

$$= \left[\underbrace{\begin{matrix} b_{00} & b_{01} & \cdots & b_{0,n-k-1} \\ b_{10} & b_{11} & \cdots & b_{1,n-k-1} \\ \vdots & & & \vdots \\ b_{k-1,0} & b_{k-1,1} & \cdots & b_{k-1,n-k-1} \end{matrix}}_{P} \quad \underbrace{\begin{matrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{matrix}}_{I_k} \right] \qquad (1)$$

and