CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS

PART I : Correction of Cell Defects in Integrated Memories.

PART II : Correction of Temporary and Catastrophic Errors.

C. V. Srinivasan

This work was supported by RCA Laboratories

Princeton, New Jersey

Department of Computer Science Technical Report #1
Livingston College, Rutgers University

CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS

PART I: Correction of Cell Defects in Integrated Memories.

INDEX TERMS

Memories
Integrated
Coding
Cell defects
Error Correction.

CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS

Part I : Correction of Cell Defects in Integrated Memories

ABSTRACT:

This paper introduces two schemes to correct bit errors caused by defective memory cells in high speed, random access memory systems. The schemes are addressed to word-organized memories produced by the integrated technologies. One of the two schemes calls for encoding of input information, and the other does not. The schemes are simple, economical for the technologies concerned, and exhibit a regularity which makes it possible to fabricate the necessary additional hardware within the same technology.

CODES FOR ERROR CORRECRTION IN HIGH SPEED MEMORY SYSTEMS

Part I : Correction of Cell Defects in Integrated Memories

FIGURES

CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS
PART I : Correction of Cell Defects in Integrated Memories. *

C.V. Srinivasan**

1. INTRODUCTION

Our principal concern in this paper is the correction of bit errors caused by defective memory cells in high speed, random access memory systems. The choice of schemes discussed here is dictated by the following considerations:

The correction process should not unduly increase the memory access time -- the delay should be less than 5 to 10 percent of the access time. Thus, for memories with 100 to 150 nanos (nanoseconds) cycle time (40 to 70 nanos access time), assuming 2 to 4 nanos delay per gate (typically, a 5-input threshold gate) we get that the error correcting net should be less than 3 or 4 layers deep. For memory words of the order of 40 to 100 bits, the only codes satisfying this requirement are the multiple redundancy majority codes, and the double redundancy parity codes discussed here.

We shall consider ways of using these codes with the maximum effectiveness. A word-organized memory scheme provides the ideal environment for their application. Memories of this type are commonly produced by integrated technologies of thin-film [1], transistor [2], cryogenic [3], and ferrite-sheet [4] memories. It is shown that cell defects of upto 10 to 20 percent, per word could be corrected. The defective cells may appear over a plane either in clusters, or be distributed as isolated bit defects. In both cases correction is possible. To best exploit the codes it is necessary to identify the defective

---

* This work was partly supported by Air Force Cambridge Research Laboratories Office of Aerospace Research, Bedford, Mass., under contract No. AF19(604)-8423. Most of this work was done while the author was at the RCA Laboratories, Princeton, N.J. 08540.
** The author is presently with the Department of Computer Science, Rutgers, The state University, New Brunswick, N.J., 08903.

cell locations on a plane, and depending upon their distribution, permute the output data wires emanating from the plane so that all the errors caused by the defective cells would be corrected by the decoding net. Schemes for finding the appropriate permutations are discussed in this paper. One may view this as a process of matching a defective plane to the decoding net. The excess redundancy available in the codes provide the necessary additional degree of freedom to perform this matching.

The decoding nets are assumed to operate in parallel on all the bits of an output word. The decoding schemes are very simple. They exhibit a regularity enabling their easy integration. Also, with very little additional cost and delay, they may be made to be fault tolerant to gate failures, to a certain extent [6].

The schemes promise to offer two dividends: They would improve the manufacturing yield by enabling one to use the defective planes with as much as 20 percent defective cells; also, they might encourage one to attempt fabrication of large area memories of sizes larger than what had been so far considered feasible.

For the kinds of errors considered, and for the given domain of application, the schemes discussed here are the only feasible ones currently known.* The schemes use fairly well known concepts in correction logic and coding techniques. The novelty is only in their mode application.

2. THE STRUCTURE OF A WORD-ORGANIZED MEMORY PLANE:

A schematic diagram of a word-organized memory plane is shown in figure 1. Let it contain w words, each having n information bits and k check bits. Let (n + k) be the number of unbroken (having electrical continuity) digit lines on the plane. The plane has (n + k) input data lines and (n + k) output

*For a good survey of coding schemes with application to memories see reference[5].

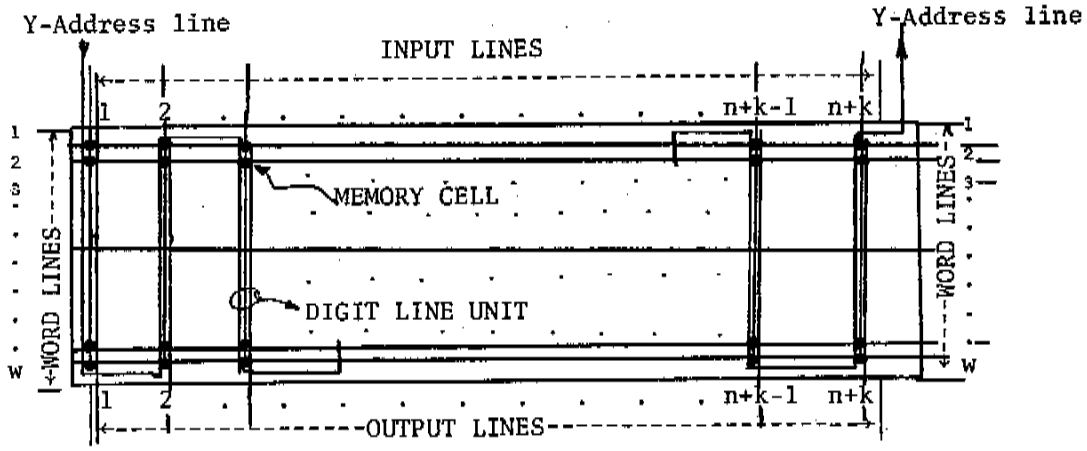


Figure 1. Schematic diagram of a word-organized memory plane.

data lines. A memory may consist of a stack of s such planes, containing in all a storage capacity for sw words in the memory.

[Figure 1]

The memory cells on the plane are located at the points of intersection of the digit lines and the word lines (see figure 1). A cell is said to be defective if its output does not always match with what was stored in it, under normal operating conditions. We shall call an unbroken digit line, an erroneous digit line , if it contains at least one defective cell.

We shall assume that a pair of corresponding digit lines in adjacent planes need not be physically contiguous to each other; they could be located anywhere in their respective planes. Thus,a facility to cross connect wires between adjacent planes is assumed to be available.

3. ERROR CORRECTION SCHEMES:

3.1 ERROR CORRECTION WITHOUT INPUT ENCODING

The simplest scheme of this type is the substitution scheme; the erroneous digit lines are eliminated by using only the error free lines for interconnections within the stack. For a given redundancy, k , of unbroken digit lines per plane, one may then correct for at most

$$e = k \qquad (1)$$

erroneous digit lines per plane. Clearly, in this scheme it may be necessary to cross connect digit line wires between adjacent planes.

By using 3-input majority gates as decoders, and storing some bits of each input word in multiple locations on a plane, one can correct more digit line errors per plane than in (1). However, this would require that the error distribution on each plane (the distribution of the defective cells or clusters) be well scattered (random), in the sense that not many of the erroneous digit lines have errors along the same word line. The error correction procedure would make use of the following principle:

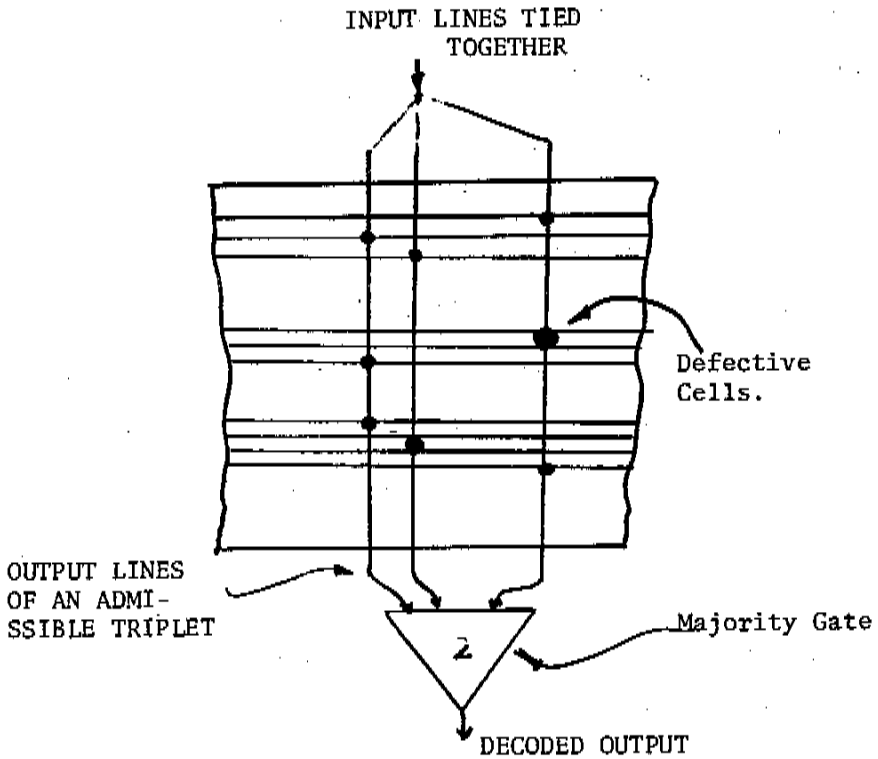


Figure 2. Illustrating the simple error correcting principle.

Suppose (d_1,d_2,d_3) are three erroneous digit lines, no two of which have defective cells along any given word line, as shown in figure 2. If the same respective information bits are stored along all the three digit lines, one for each word on the plane, then the majority of their outputs would always equal the stored input value. This situation is shown in figure 2. We shall call such a triplet, an admissible triplet.

[Figure 2]

Let us assume that a plane contains t such admissible triplets, all of which are disjoint, and h erroneous lines outside the t triplets, among which no admissible triplets exist. Let us assume that these h lines are replaced by the available error free lines on the plane. Then, to get n information bit storage capacity per word on the plane with k redundant lines, it is easy to show that

$$2t + h \leq k . \tag{2}$$

When $h = 0$, we get the maximum possible value for the number of correctable erroneous digit lines per plane,

$$e = 3k/2. \tag{3}$$

An interconnection scheme for the input and output digit line connections between the various planes of a stack may be easily constructed. One may assume that the planes $P_1,P_2,\ldots,P_s$ of a memory stack have each, for $1 \leq e \leq s$, t_i admissible triplets and h_i erroneous lines outside the triplets. The planes in the stack may be arranged in the order of ascending t_i, $t_1 \leq t_2 \leq \ldots \leq t_s$. The interconnection scheme would use one 3-input majority gate for each corrected output from the stack, as shown in figure 3.

[Figure 3]

Let me now consider the following digit line permutation problem: Let $\{d_1,d_2,\ldots,d_e\}$ be the set of all erroneous digit lines on a plane. A subset of these lines, say $T_i = \{d_{i_1},d_{i_2},\ldots,d_{i_u}\}$ for $u = 2m + 1$ for some

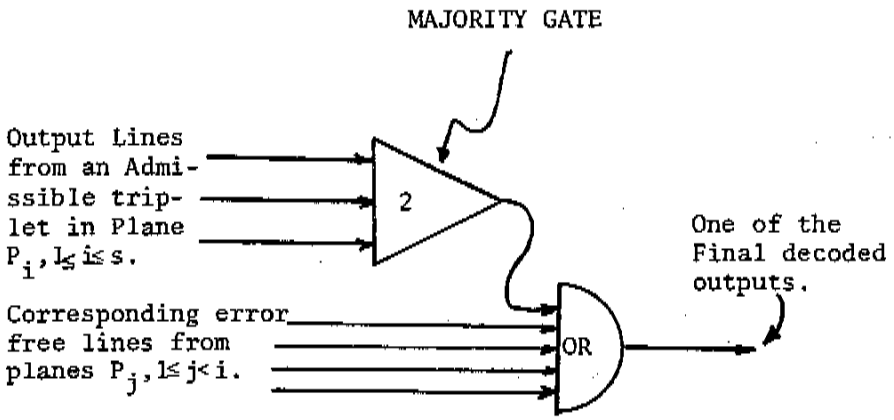


Figure 3. Decoding net for a typical output bit from a memory stack for scheme 2.

$m \geq 1$, is said to be an admissible subset if there exists a decoding strategy which could produce a single corrected output bit for every set of u output values obtained from T_i. A $(2m + 1)$-input majority gate would clearly do this if all the digit lines in T_i had the same respective bit values stored in them along each word line. Let $\{T\} = \{T_1, T_2, \ldots, T_r\}$, $r \geq 0$, be the set of all admissible subsets (not necessarily disjoint) among e erroneous digit lines on a plane. Let these subsets be listed as shown in Table 1 below, where $e = 9$, $r = 8$, and $m = 1$. Let T_i be the name of the ith row in Table 1. Each T_i is a string of 1's and 0's. Let $\{L\}$ be the subset of $\{T\}$ such that

1. all the subsets in $\{L\}$ are disjoint, and
2. $\{L\}$ contains the largest possible number of members of $\{T\}$.

Our task is to construct $\{L\}$ given $\{T\}$. The necessary digit line permutations at the input and output ends of a memory plane are directly specified by the members of $\{L\}$.

Table 1
Table of Admissible subsets.

Admissible subsets \ Erroneous digit lines	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
T_1	1	0	0	0	1	0	0	0	1
T_2	0	1	1	0	0	0	0	1	0
T_3	1	0	0	1	1	0	0	0	0
T_4	0	0	0	0	0	1	1	1	0
T_5	0	1	0	0	1	0	0	1	0
T_6	1	0	0	1	0	0	0	0	1
T_7	0	1	1	0	0	1	0	0	0
T_8	0	0	0	1	0	0	1	0	1

We shall use the following definitions:

DEFINITION 1: $[A] = a_{ij}$, $1 \leq i,j \leq r$, $a_{ij} = 0$ if $i \neq j$ and $T_i.T_j \geq 1$, $a_{ij} = 1$ if $i = j$ or $T_i.T_j = 0$, where $T_i.T_j$ is the dot-product of the rows T_i and T_j of Table 1. The matrix [A] corresponding to Table 1 is :

$$[A] = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \end{bmatrix} \quad \ldots \qquad (5)$$

Clearly, [A] is a symmetric matrix.

DEFINITION 2: $\{M_i\}$ is a <u>maximal subset</u> of $\{T\}$, if and only if, for $1 \leq i \leq r$, T_i is in $\{M_i\}$, for T_u, T_v both in $\{M_i\}$, $u \neq v$, $T_u.T_v = 0$ (T_u and T_v do not have common digit lines), and for any T_j not in $\{M_i\}$, there exists a T_u in $\{M_i\}$ such that $T_j.T_u \neq 0$ (T_j and T_u have a common digit line).

Among all possible $\{M_i\}$ we wish to choose the one containing the largest number of members of $\{T\}$. Let

$$X_i = \bigwedge_{\{j \mid T_j \in M_i\}} A_j, \quad \ldots \qquad (6)$$

where A_j is the jth row of [A] (See Eq. (5)). X_i is the componentwise logical AND of all A_j for j such that T_j is in $\{M_i\}$.

LEMMA 1 : $\{M_i\}$ is maximal if and only if X_i has 1's in every position j such that T_j is in $\{M_i\}$, and has 0's in all other positions.
The proof is straightforward.

LEMMA 2 : $\{M_i\}$ is maximal if and only if for every T_u, T_v in $\{M_i\}$, $A_u.A_v \geq \ell$, and $A_u.X_i = \ell$. For every T_u in $\{M_i\}$, and T_v not in $\{M_i\}$, $A_u.A_v < \ell$, where ℓ is the number of elements in $\{M_i\}$.

The proof follows from Lemma 1.

Let w_i be the number of 1's in the ith row (column) of [A]; $w_i = A_i . A_i$. Let $[A/A_i]$ be the submatrix of [A] defined as follows: If A_i has 1's in positions $(i, i_1, i_2, \ldots, i_u)$, $u = w_i - 1$, then $[A/A_i]$ consists of the rows and columns $(i, i_1, i_2, \ldots, i_u)$ of [A]. Clearly, $[A/A_i]$ will have w_i rows and w_i columns. The columns and rows of $[A/A_i]$ shall be so arranged that for $j < p$ the number of 1's in the jth row (column) of $[A/A_i]$ is not less than the number of 1's

in the pth row (column). The first row (column) of $[A/A_i]$ will be, clearly, all 1's. Let the sequence $(i,i_1,i_2,\ldots,i_u)$ be the row (column) numbers of [A] in the order of their appearence in $[A/A_i]$. Let $[A/A_i]^2 = [P_i]$. All the elements of $[P_i]$ will be $\leq w_i$. For example, for the [A] shown in equation (5), $w_3 = 4$,

$$[A/A_3] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} , \tag{7}$$

and the selected rows and columns of [A] are (3,2,4,7) in the order shown. Finally,

$$[P_3] = \begin{bmatrix} 4 & 2 & 2 & 2 \\ 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 \end{bmatrix} . \tag{8}$$

Let $[LP_i]$ be the <u>labelled</u> $[P_i]$, defined as follows:

$$[LP_i] = \begin{bmatrix} 0 & i & i_1 & i_2 & \cdots & i_u \\ i & & & & & \\ i_1 & & & [P_i] & & \\ i_2 & & & & & \\ \vdots & & & & & \\ i_u & & & & & \end{bmatrix} \tag{9}$$

where $(i,i_1,i_2,\ldots,i_u)$ are arranged in the same order in which the columns and rows of [A] were selected to form $[A/A_i]$. Thus, in our example,

$$[LP_i] = \begin{bmatrix} 0 & 3 & 2 & 4 & 7 \\ 3 & 4 & 2 & 2 & 2 \\ 2 & 2 & 2 & 1 & 1 \\ 4 & 2 & 1 & 2 & 1 \\ 7 & 2 & 1 & 1 & 2 \end{bmatrix} . \tag{10}$$

From Lemma 2 and the conventions above, we have

LEMMA 3: If $\{M_i\}$ contains g members of {T} then there exists a permutation of the rows and columns of $[P_i]$ such that the new matrix $[P_i']$ so obtained satisfies the form:

$$[P_i'] = \left[\begin{array}{c|c} [Q] & [R] \\ \hline [S] & [V] \end{array}\right] \tag{11}$$

where all elements of [Q] are $\geq$ g and every row of [S] (Column of [R])

has an element $<g$. Also [Q] is a $g \times g$ matrix.

We shall make use of Lemma 3 and the matrix $[LP_i]$ to select the desired $\{L\}$. The algorithm for selecting $\{L\}$ is based on the following three principles:

(i) The number of elements in $\{M_i\}$ is always $\leq w_i$, the number of 1's in A_i. Hence, one may begin by considering the rows of [A] with the largest number of 1's.

(ii) If $\{M_i\}$ has g members of $\{T\}$ and for all $j \neq i$ either $w_j \leq g$ or else $\{M_j\}$ has $<g$ members of $\{T\}$, then $\{M_i\} = \{L\}$.

(iii) For $\{M_i\}$ to be $\{T_i, T_{i_1}, \ldots, T_{i_{g-1}}\}$ the submatrix of $[LP_i]$ consisting of the rows and columns of $[P_i]$ with labels $(i, i_1, i_2, \ldots, i_{g-1})$ should have every element $\geq g$.

Using these principles the algorithm may be easily constructed.

3.2 ERROR CORRECTION WITH INPUT ENCODING

The schemes proposed in the previous section may be used only if $e \leq 3k/2$, where k is the number of redundant lines per plane (See Eq.3). If $e > 3k/2$ then some form of input encoding is essential. The coding scheme proposed here makes use of the following strategy:

The input information bits are encoded into a code which has a potential to correct a relatively large (compared to the length of the encoded word) bursts of errors in each code word. (A code word is said to have a burst error of length b if all the errors in the word lie within a substring of length b. The first digit of the code word is considered to be logically adjacent to the last digit.) The burst error correcting capacity gives a rough measure of the total error correcting power of the code. The randomly distributed defective cells on a plane are rearranged by permutation of the digit lines at the input and output ends, in such a way that along each word on the plane all bit errors caused by the defective cells would appear distributed in the output code word

in a manner, correctable by the decoding scheme. Decoding simplicity and high speed are achieved by using a relatively large number of redundant bits in the code word.

The encoding scheme used is fairly well known: Each memory plane is to contain $2n$ cells per word for $n \geq 3$; n of these would be <u>information</u> cells and the remaining n would be the redundant <u>check</u> cells. Let $x_0, x_1, \ldots, x_{n-1}$ be the information bits, and $a_0, a_1, \ldots, a_{n-1}$, the check bits. For $0 \leq i \leq n-1$, let

$$a_i = x_i \oplus x_{i \ominus_n 1}, \quad i = 0,1,\ldots,n-1, \tag{12}$$

where $\oplus$ denotes modulo 2 addition and $\ominus$ denotes modulo n addition (We

The burst error correcting capability of such a code is given by the following Lemma:

LEMMA 5: The code defined by (12) and (13), with digits arranged as per rules R1 and R2 is capable of correcting all burst errors of length b or less, where

$$b = \lceil (2n/5) - 1, \tag{16}$$

$\lceil (2n/5)$ being the least integer $\geq (2n/5)$, and $2n$ is the length of the code.

<u>Proof</u>: Let M be a one-one mapping from digits of the code onto integers modulo $2n = \nu$, say:

$$[x_i \overset{M}{\leftrightarrow} 2i] \text{ and } [a_i \overset{M}{\leftrightarrow} 2i+1] \quad i = 0,1,\ldots,n-1. \quad \ldots\ldots \tag{17}$$

For any i, $0 \leq i \leq n-1$, M maps the elements of D_i onto the sets:

$$M(D_i) = \{2i \ominus_\nu 2,\ 2i \ominus_\nu 1,\ 2i,\ 2i \ominus_\nu 1,\ 2i \ominus_\nu 2\}. \tag{18}$$

Consider for any j, $0 \leq j \leq \nu-1$, the following string of integers:

$$S_j = [j\ (j \ominus_\nu 5)\ (j \ominus_\nu 10)\ \ldots\ (j \ominus_\nu 5(b-1))] \tag{19}$$

where b is given by equation (16). The length of S_j is b. Any two integers of S_j differ by atleast 5, modulo $2n$. Thus, from (18) it follows

$[\hat{x}_i, \hat{x}_{i\oplus_n 1}, \hat{a}_i]$ or $[\hat{x}_i, \hat{x}_{i\oplus_n 1}, \hat{a}_{i\oplus_n 1}]$, or (iii) all the bits in $\hat{D}_i$ are in error.

The proof is straightforward. For $n > 5$, the above error correcting property may be exploited to obtain a burst error correcting feature, through suitable rearrangement of information and check digits. This burst error correcting feature of the code is a good measure of the error correcting potential of the code, which may be usefully employed for the correction of errors caused by cell defects in a memory plane. Consider for example the code for $n = 8$. Let the information and check bits be arranged as shown below:

$$[x_0 a_2 x_5 a_7 x_2 a_4 x_7 a_1 x_4 a_6 x_1 a_3 x_6 a_0 x_3 a_5] \ .$$

In this arrangement no two bits belonging to any substring of length 3 occur together in any D_i for $0 \leq i \leq 7$. Therefore, by Lemma 4, if all the errors in the code word lie within a substring of length ≤ 3, then they would be correctable. Such errors are called burst errors of length ≤ 3. The above arrangement of digit for $n = 8$ would thus correct all burst errors of length ≤ 3. The general rules for arranging a $2n$ digit code, defined by (12) to obtain the corresponding burst error correcting capability may be stated as follows:

Digit Arrangement Rules:

R1) If n is not divisible by 5 then for $0 \leq i \leq n-1$ each x_i in the code word is to be followed by $a_{i\oplus 2}$, and each a_i by $x_{i\oplus 3}$. The initial digit (the leftmost one) may be chosen arbitrarily. (The code arrangement given above for $n = 8$ follows this rule.)

R2) If n is divisible by 5 then for any i, $0 \leq i \leq n-1$, let the segment ξ_i be defined as follows:

$$\xi_i = [x_i a_{i\oplus_n 2} x_{i\oplus_n 5} a_{i\oplus_n 7} \cdots x_{i\oplus_n (n\ominus 5)} a_{i\oplus_n (n\ominus 3)}] . \tag{14}$$

Then the code word is

$$W = \xi_i \xi_{i\oplus_n 1} \xi_{i\oplus_n 2} \xi_{i\oplus_n 3} \xi_{i\oplus_n 4} \ . \tag{15}$$

The reader may easily verify that the segments all contain distinct digits.

The burst error correcting capability of such a code is given by the following Lemma:

LEMMA 5: The code defined by (12) and (13), with digits arranged as per rules R1 and R2 is capable of correcting all burst errors of length b or less, where

$$b = \lceil (2n/5) - 1, \tag{16}$$

$\lceil (2n/5)$ being the least integer $\geq (2n/5)$, and $2n$ is the length of the code.

Proof: Let M be a one-one mapping from digits of the code onto integers modulo $2n = \nu$, say:

$$[x_i \overset{M}{\leftrightarrow} 2i] \text{ and } [a_i \overset{M}{\leftrightarrow} 2i+1] \quad i = 0,1,\ldots,n-1. \quad \ldots\ldots \tag{17}$$

For any i, $0 \leq i \leq n-1$, M maps the elements of D_i onto the sets:

$$M(D_i) = \{2i\theta_\nu 2,\ 2i\theta_\nu 1,\ 2i,\ 2i\theta_\nu 1,\ 2i\theta_\nu 2\}. \tag{18}$$

Consider for any j, $0 \leq j \leq \nu-1$, the following string of integers:

$$S_j = [j\ (j\theta_\nu 5)\ (j\theta_\nu 10)\ \ldots\ (j\theta_\nu 5(b-1))] \tag{19}$$

where b is given by equation (16). The length of S_j is b. Any two integers of S_j differ by atleast 5, modulo $2n$. Thus, from (18) it follows that no two elements of S_j can together belong to any one (D_i), for any i, $0 \leq i \leq n-1$. Let

$$M^{-1}(S_j) = M^{-1}(j)\ M^{-1}(j\theta_\nu 5)\ \ldots\ M^{-1}(j\theta_\nu 5(b-1)) \tag{20}$$

It is easily verified that for all j, $0 \leq j \leq 2n-1$, the sequence $M^{-1}(S_j)$ satisfies rule R1 of the code arrangement. Therefore, when 5 does not divide n, no substring of length b or less of the code word contain two or more elements from any one D_i, $0 \leq i \leq n-1$. Thus, all burst errors of length $\leq b$ could be corrected.

In case 5 divides n let $n = 5g$. In this case the code word is by rule R2, $\xi_i \xi_{i\theta_n 1} \xi_{i\theta_n 2} \xi_{i\theta_n 3} \xi_{i\theta_n 4}$, for some i, $0 \leq i \leq n-1$, where ξ_i is given by equation (14). The equation for ξ_j, for $i \leq j \leq i\theta_n 4$, may be re-

written as,

$$\xi_j = [x_j a_{j\theta_n 2} x_{j\theta_n 5} a_{j\theta_n 7} \cdots x_{j\theta_n 5r}\ a_{j\theta_n 5r\theta_n 2} \cdots x_{j\theta_n 5(m-1)}\ a_{j\theta_n 5(m-1)\theta_n 2}], \quad (21)$$

where $2m = (b+1)$ is the length of ξ_j. Now,

$$M(\xi_j) = [2j\ 2j\theta_\nu 5 \ldots 2j\theta_\nu 10r\ 2j\theta_\nu 10r\theta_\nu 5 \ldots 2j\theta_\nu 5b]. \quad (22)$$

Consider the set $M(D_t)$ for $t = j\theta_n 5r\theta_n 1$, for $0 \le r \le m-1$.

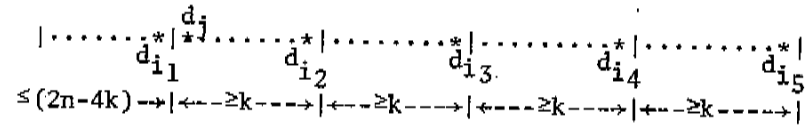
$$M(D_t) = \{2j\theta_\nu 10r\theta_\nu 4,\ 2j\theta_\nu 10r\theta_\nu 3,\ 2j\theta_\nu 10r\theta_\nu 2,\ 2j\theta_\nu 10r\theta_\nu 1,\ 2j\theta_\nu 10r\} \quad (23)$$

The number $(2j\theta_\nu 10r)$ in $M(D_t)$ is a member of $M(\xi_j)$ (See equation (22)). In $M(\xi_j)$, $(2j\theta_\nu 10r)$ is followed by $(b-2r)$ other elements. The second element, $(2j\theta_\nu 10r\theta_\nu 3)$, of $M(D_t)$ is in segment $M(\xi_{j\theta_n 1})$, and $(2j\theta_\nu 10r\theta_\nu 3)$ is the 2rth element in $M(\xi_{j\theta_n 1})$. Thus, $(2j\theta_\nu 10r)$ and $(2j\theta_\nu 10r\theta_\nu 3)$ belonging to $M(D_t)$ are seperated in $M(\xi_j)M(\xi_{j\theta_n 1}) \ldots M(\xi_{j\theta_n 4})$ by a distance of b. Similarly, one may verify that for all pairs of elements in $M(D_t)$ the distance between them is $\ge b$. Thus all burst errors of length $\le b$ can be corrected. QED

DEFINITION 4: A $B_{\nu t}$-code is one that can correct all burst errors of length $\le t$. For $m > t$, a $B_{\nu m}$-code is said to have a larger burst error correcting capability than a $B_{\nu t}$-code.

LEMMA 6: Among all $B_{\nu t}$-codes that one can derive by rearranging the digits of the code defined by equation (12), for $\nu = 2n$, the $B_{\nu b}$-code for $b = \lceil (2n/5) - 1$, has the maximum burst error correcting capability.

Proof: We shall prove this lemma by showing that m is always less than $(2n/5)$. As $\lceil (2n/5) - 1$ is the largest integer less than $(2n/5)$ the proof will follow from Lemma 5. Let k be the largest integer such that a $B_{\nu k}$-code may be derived from the code with $2n$ digits. Consider any D_i for $0 \le i \le n-1$. The elements of D_i (let us denote them by $d_{i_1}, d_{i_2}, d_{i_3}, d_{i_4}, d_{i_5}$) must lie on the code word as shown below, since they should be seperated from each other by atleast k digits. The digit d_{i_1} must, therefore, lie within the first $(2n-4k)$ digits. This is true for each D_i.



Consider now the (2n-4k+1)st digit, indicated in the above figure by d_j. Let d_j be included in the neighborhood set D_j. d_j is at a distance $\leq(2n-4k)$ from another element of D_j that is in the first (2n-4k) digits of the code. Thus,

$$k \leq 2n-4k \quad \text{or} \quad k \leq 2n/5.$$

Now suppose that $k = 2n/5$, when n is divisible by 5. Let the first $2n/5$ digits of the code be

$$S_1 = [c_1 c_2 \ \ldots \ c_{2n/5}],$$

In this case $M(S_1) = M(c_1)M(c_2) \ \ldots \ M(c_{2n/5})$ will all be congruent modulo 5. However, $M(c_{2n/5+1})$, of the (2n/5+1)st digit of the code will not be congruent modulo 5 to any of the digits in $M(S_1)$. Thus, there will exist a c_j and c_k, $1 \leq j,k \leq 2n/5$, such that

$$M(c_j)+5 = M(c_k) \quad \text{and} \quad M(c_j) < M(c_{2n/5+1}) < M(c_k).$$

Thus, $c_{2n/5+1}$ will belong to two sets D_r and D_t, $0 \leq r,t \leq n-1$, such that $c_{2n/5+1}, c_j$ are in D_r and $c_{2n/5+1}, c_k$ are in D_t. In such a case, either c_j or c_k (or both) must be at a distance $<2n/5$ from $c_{2n/5+1}$. Hence, $k < 2n/5$. QED

The reader may notice that in the above arguments we have not made use of the correction features of the code specified by items (ii) and (iii) of Lemma 4. Clearly, one could not use them for the correction of permanent errors on a plane. They might, however, prove useful for protection against temporary errors in the outputs from a plane. Let me now discuss the problem of rearrangement of output lines of a plane to match a defective plane to the decoding net of the code.

3.2.1 DIGIT LINE PERMUTATION SCHEME FOR THE $B_{\nu b}$-CODE.

Let $c_0 \ c_1 \ \ldots \ c_{\nu-1}$, for $\nu = 2n$, be the digits of the code word with n information digits and n check digits. Let Q_i be an admissible quintu-

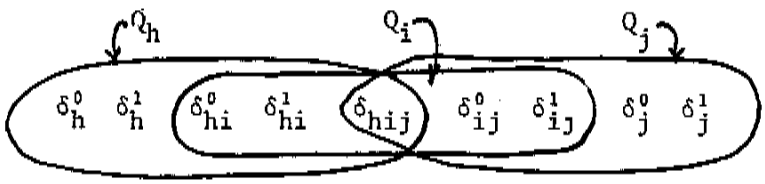


Figure 4 . The structure of a valid neighborhood sequence $(Q_h Q_i Q_j)$

plet,

$$Q_i = \{\ \delta_i^0,\ \delta_i^1,\ \delta_i^2,\ \delta_i^3,\ \delta_i^4\ \}, \tag{24}$$

where each δ_i is an output digit line of a memory plane, such that no two of them have defective cells along a same word line. For the permutation scheme to work, clearly, it is necessary that we find n such quintuplets. The necessary and sufficient conditions may be stated as follows:

DEFINITION 5: For $h \neq i \neq j,\ j \neq h$, the sequence $(Q_h\ Q_i\ Q_j)$ is said to form a valid neighborhood sequence if

$$\begin{aligned} Q_h \cap Q_i &= \{\delta_{hi}^0\ \delta_{hi}^1\ \delta_{hi}^2\}, \\ Q_i \cap Q_j &= \{\delta_{ij}^0\ \delta_{ij}^1\ \delta_{ij}^2\},\ Q_i \cap Q_j \cap Q_h = \{\delta_{ijh}\}, \end{aligned} \tag{25}$$

where $\cap$ denotes set intersection. This requires the valid neighborhood sequence to have the structure shown in figure 4.

[Figure 4]

DEFINITION 6: $Q_0Q_1\ldots.Q_{n-1}$ is said to be a valid complete sequence of length n if for $0 \leq i \leq n-1$ every subsequence $(Q_iQ_{i\theta_n 1}Q_{i\theta_n 2})$ is a valid neighborhood sequence, and the number of digit lines in the set union of all Q_i, for $i = 0,1,\ldots,n-1$, is 2n.

Let the code defined by (12) with digits arranged as shown below be called the cannonical 2n-code:

$$\begin{aligned} W &= x_0a_0x_1a_1\ldots.x_{n-1}a_{n-1} \\ \hat{W} &= \hat{x}_0\hat{a}_0\hat{x}_1\hat{a}_1\ldots.\hat{x}_{n-1}\hat{a}_{n-1}. \end{aligned} \tag{26}$$

DEFINITION 7: A valid permutation of 2n digit lines of a memory plane is one which causes the digit errors in all possible outputs, produced by the defective cells on a plane, to appear distributed over the output word in such a manner that for every output word all the defective cell errors in the word are correctable by the cannonical 2n-code.

LEMMA 7: A valid permutation of 2n lines exists if and only if there exists a valid complete sequence of length n.

Proof: Let $d_{j_0}, d_{j_1}, \ldots, d_{j_{\nu-1}}$, for $\nu = 2n$, be a valid permutation For $0 \le i,k,r \le \nu-1$ let

$$\beta(d_{j_i}) = c_i, \tag{27}$$

be the ith digit of the code word, and let

$$\beta(d_{j_i}, d_{j_k}, \ldots, d_{j_r}) = \beta(d_{j_i})\beta(d_{j_k}, \ldots, d_{j_r}). \tag{28}$$

Also, for $0 \le i = 2r \le \nu-2$, let

$$Q_{i/2} = \{d_{j_{i\theta_\nu 2}}, d_{j_{i\theta_\nu 1}}, d_{j_i}, d_{j_{i\theta_\nu 1}}, d_{j_{i\theta_\nu 2}}\} \tag{29}$$

Since the permutation is valid, it follows that $\beta(Q_i)$ is a neighborhood set satisfying condition (i) of Lemma 4. Hence Q_i is an admissible quintuplet, and $(Q_0Q_1 \ldots Q_{n-1})$ is a valid complete sequence. This proves the first part of Lemma 7.

Now assume that $(Q_0Q_1 \ldots Q_{n-1})$ is a valid complete sequence. For $0 \le i,j,k \le n-1$, $i \ne j \ne k$, and $i \ne k$, let*

$$\begin{aligned} \delta_{ijk} &= Q_i \cap Q_j \cap Q_k \\ \delta_{jk} &\in Q_j \cap Q_k \\ \text{and} \quad \delta_{jk} &\notin Q_u \cap Q_v \end{aligned} \tag{30}$$

for any u and v, $u \ne v$ and $uv \ne jk$. Then the digit lines belonging to $Q_0Q_1 \ldots Q_{n-1}$ may be arranged as, $\delta_{(n-1)01}\, \delta_{01}\, \delta_{012}\, \delta_{12} \ldots \delta_{(n-2)(n-1)0}\, \delta_{(n-1)0}$. Now define the mapping,

$$\zeta(\delta_{ijk}) = x_j, \quad \zeta(\delta_{jk}) = a_j \tag{31}$$

Then ζ of the above sequence of δ's is the cannonical 2n-code of (26). Since each Q_i is an admissible quintuplet, it follows that every $\zeta(Q_i) = \{\zeta(\delta) \mid \delta \in Q_i\}$ is a neighborhood set satisfying condition (i) of Lemma 4. Also, for

---

* Notice that definition of δ's in (30) is possible only because for each i, $0 \le i \le n-1$, $(Q_{i\theta_n 1} Q_i Q_{i\theta_n 1})$ is a valid neighborhood sequence, and in addition the total number of elements in the union of all Q_i's is 2n.

each admissible subset D_i, $\beta^{-1}(D_i) = \{ \beta^{-1}(d) \mid d \in D_i \} = Q_i$. Thus for every possible output word the errors in the word could be corrected using (13). Hence $\beta^{-1}(W)$, for W obtained from (30) and (31), and the sequence of δ's shown above, is a valid permutation scheme. QED

This Lemma may be used to construct the digit line permutation algorithm: Given a distribution of defective cells over a plane all that one has to do is to find a valid complete sequence of length n. This sequence, in fact, would then exhibit the necessary valid permutation of the 2n digit lines of the plane. The following points are noteworthy:

Suppose a valid permutation of the 2n digit lines is now rearranged to get the output word in the order specified by rules R1 and R2, instead of maintaining it in the cannonical form of W. Then the errors in the output code word would not all be necessarily burst errors of length $\leq \lceil (2n/5) - 1$. For example, for the $n = 8$ case the code arranged according to rule R1 is $x_0a_2x_5a_7x_2a_4x_7a_1x_4a_6x_1a_3x_6a_0x_3a_5$. Now, if a_2, a_4, a_6, a_0 are all in error, the errors would still be correctable by (13). However, this error is not a burst error of length ≤ 3 in the arrangement above. The proof of lemma 7 allows for the capture of all such anomalous distributions of errors, since it makes use of only the condition (i) of lemma 4, and does not require the output errors to be always in the burst form.

4. CONCLUDING REMARKS

We have discussed basically three schemes, as summarized in Table II. As we move from scheme 1 to 3 in Table II the schemes get progressively more complex. This increasing complexity is the result of increasing randomness of error distributions in the three cases. The decoder per bit (notice that the decoder is a parallel decoder) of the output information for case 2 is shown in figure 3, and the decoder for case 3 is shown in figure 5.

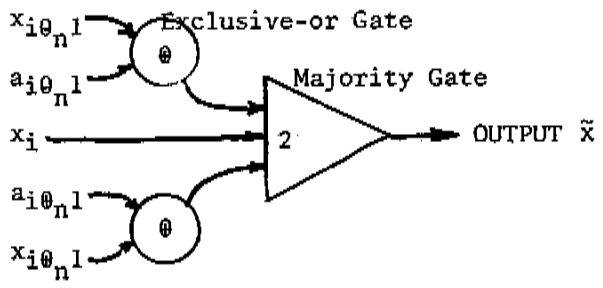


Figure 5. The decoding net for a typical output bit for scheme 3.

In scheme three there is a residual error correcting capability as indicated by rules (ii) and (iii) of Lemma 4. This residual capability would come in handy for protection against temporary errors. We have not, however, estimated its usefulness for this purpose. It would be useful to compare the improvement in yield that one might hope to achieve in the three schemes for various assumptions on defective cell distribution over a plane. This study has not been so far made.

In the second part of this paper I shall discuss certain new classes of burst error and module error correcting codes well suited for parallel decoding. The module error correcting codes may be used to protect against catastrophic failures in a system with modular organization. I shall also discuss a class of self-orthogonal [7] codes suitable to correct a small number of randomly distributed intermittent errors.

Table II. Summary of schemes discussed. The number of redundant digit lines per plane is n, the number of information digits per word is n, and the number of words per plane is w.

No.	Schemes	Max. No. of digit line errors correctable.	Max. No. of digit errors correctable per plane.	Percentage errors per plane.
1.	replacement	n	$w.n$	50%
2.	Majority	$(3/2)n$	$(1/2)w.n$	25%
3.	coding	$2n$	$(2/5)w.n$	20%

ACKNOWLEDGEMENT

I am thankful to George Cooke for several discussions we had during the development of proofs for Lemmas 5 and 6.

REFERENCES:

1. Rajchman, J.A. "Integrated Memories and Superconductive Memories - A Survey of Techniques, Results, and Processes." Proc. IFIP 1965, Vol. 1 123.

2. Sander, W.B. "Semiconductor Memory Circuits and Technology" Proc. FJCC Vol. 33, part 2, 1968 pp. 1205-1211.

3. Gange, R.A. "Cryogenic Hybrid System for Very Large Random Access Memory" Proc. IEEE, Oct. 1968, Vol. 56, #10, pp. 1679-1690.

4. Shabender, R. "Laminated-Ferrite Memories - Review and Evaluation" RCA-Review, Vol. 29, #2, June 1968, pp. 180-198.

5. Tang D.T. and Chien R.T. "Coding for Error Control" IBM System J1., Vol. 8, #1, 1969, pp. 48-86.

6. Pierce, W.H. "Failure Tolerant Computer Design" New York, Academic Press, 1965.

7. Massey, J. "Threshold Decoding" The M.I.T. Press, Cambridge, Mass. 1963.

FOOTNOTES

Page 1.

* Manuscript received April 1970. Revised manuscript received Oct. 1970. This work was partly supported by Air Force Cambridge Research Laboratories, Office of Aerospace Research, Bedford, Mass., under contract number AF 19(604)-8423. Most of this work was dome while the author was at the RCA Laboratories, Princeton, N.J. 08540.

** The author is presently with the Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswic, N.J. 08903.

Page 2.

* For a good survey of coding schemes with application to memories see reference [5].

Page 15.

* Notice that definition of δ's in (30) is possible only because, for each i, $0 \leq i \leq n-1$, $(Q_{i\theta_n 1} Q_i Q_{i\theta_n 1})$ is a valid neighborhood sequence, and in addition the total number of elements in the union of all Q_i's is $2n$.

APPENDIX I:

Algorithm for Constructing the maximal subset, {L}, of the set of all admissible subsets

We shall make use of the following variables in the Algorithm:

1). {I} is a set of integers. Initially {I} is equal to ϕ, the empty set.

2). $T_1, T_2, \ldots, T_a$ are the admissible subsets of digit lines.

3). [A] is the matrix defined in definition 1.

4). A_i is the ith row of [A].

5). w_i is the weight of (the number of 1's in) A_i.

6). $[P_i]$ and $[LP_i]$ are the matrices defined in the discussion immediately following Lemma 2.

7). $\{M_i\}$ is the maximal subset defined in defn. 2.

8). ℓ_i is either equal to zero or equal to the number of elements in $\{M_i\}$.

9). For $1 \leq j \leq r$, and $b>0$, $[Q_{bj}]$ is a submatrix of P_i consisting of the rows and columns of P_i with labels, say

$$(i_{j_1}, i_{j_2}, \ldots, i_{j_b})$$

in $[LP_i]$, such that all the elements of $[Q_{bj}]$ are $\geq b$.

10). $\{C_{bj}\} = \{i_{j_1}, i_{j_2}, \ldots, i_{j_b}\}$ the set of labels in $[LP_i]$ defining the submatrix $[Q_{bj}]$ of $[P_i]$ for some $b>0$ and $1 \leq j \leq r$.

ALGORITHM:

STEP 1: Construct $T_1, T_2, \ldots, T_a$ for some $a \geq 0$. If $a=0$ then $\{L\} = \phi$, STOP. If $a>0$ go to Step 2.

STEP 2: Construct [A]. Select A_i such that for all $j \neq i$, $w_j \leq w_i$.

STEP 3: $\{I\} \leftarrow \{I\} \cup \{i\}$

Construct $[LP_i]$.

Choose the largest integer b such that $[Q_{bj}]$ exists for $j = 1, 2, \ldots, r., r \geq 1$. Let $\{C_{bj}\}$, $j = 1,2,\ldots,r$, be the labels defining $[Q_{bj}]$.

STEP 4: If $b > \ell_k$ for all $k \in \{I\}$ go to STEP 5. Else set $\ell = 0$ and go to STEP 6.

STEP 5: Construct $\{C_{bj}\}$, $\{M_{ij}\} = \{T_k \mid k \in \{C_{bj}\}\}$, and

$$X_{ij} = \bigwedge_{\{k \in \{C_{bj}\}\}} (A_k), \text{ for } j = 1,2,\ldots,r.$$

Select a j, $1 \leq j \leq r$, such that X_{ij} satisfies Lemma 1. If such a j exists then set

$$\{M_i\} = \{M_{ij}\},$$

and $$\ell_i = b$$

If $\ell_i = w_i$ then set

$$\{L\} = \{M_i\}$$

and STOP. If $\ell_i < w_i$ go to STEP 6.

If no such j exists then go to STEP 7.

STEP 6: Choose a row A_m of A such that

$w_m = w_i$, and $m \notin \{I\}$.

If such an m exists set i=m and go to STEP 3.

If no such m exists then choose an m such that

m $\in\{I\}$, $w_m < w_i$, and for all $k \neq m$, and $k \notin \{I\}$, $w_k \leq w_m$. See whether $w_m > \ell_j$ for all $j \in \{I\}$. If it is, then set i=m and go to STEP 3. Else, choose $\{L\} = \{M_j\}$ for j such that $j \in \{I\}$ and for all $k \in \{I\}$, and $k \neq j$ $\ell_j \geq \ell_k$.

STEP 7:

Set $b \leftarrow (b-1)$

and go to STEP 4.

CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS
Part II: Correction of Temporary and Catastrophic Errors.

C. V. Srinivasan

Index Terms

Memories

Coding

Error Correction

Temporary errors

Catastrophic errors

CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS

PART II: CORRECTION OF TEMPORARY AND CATASTROPHIC ERRORS.

C. V. Srinivasan

Abstract:

A few classes of codes, suitable for error correction in high speed memory systems, are presented. The codes have relatively simple parallel decoding nets. The codes may be used both for the correction of temporary and catastrophic errors.

CODES FOR ERROR CORRECTION IN HIGH SPEED MEMORY SYSTEMS
PART II: CORRECTION OF TEMPORARY AND CATASTROPHIC ERRORS*

C.V.Srinivasan**

1. Introduction:

We shall assume that the requirements enumerated in Part I of this paper [1] are still valid. We shall, however, relax the high speed requirement, to a certain extent. The schemes discussed here would have decoding delays of 50 - 200 nanoseconds. Thus, as per our earlier assumption, the codes would be suitable for memories having 0.5 - 2 microseconds access time.

In the case of temporary errors we shall assume that the number of errors per word would be relatively small, say one to five per word of 100 bits. Also, when multiple errors occur, we shall consider two cases: In one case the erroneous bits are assumed to show clustering along the length of a word, and in the other, they are assumed to be randomly distributed. To cover the first case, a class of non-cyclic burst-error correcting codes for short bursts is proposed. For the second case, a class of majority codes [2,3] is proposed. To protect against catastrophic failures a modular memory organization is proposed, in which a failure of any one of a set of memory modules might be corrected for by the use of an error correcting code. We shall refer to the code, as the module-error correcting code. All the codes proposed have simple, parallel decoding nets. The nets may be easily made to be error-masking, to protect against gate failures in the decoding nets.

We shall consider the burst and module error correcting codes together, since they are closely related. The class of majority codes are

similar to the self-arthogonal codes of Townsend and Weldon [2].

2. The Non-Cyclic Burst-Error Correcting (BEC) Code

2.1. Some Remarks on BEC-Codes

Let a b-BEC-Code be one that can correct all errors distributed within a set of any b consecutive digits of a code-word, when the digits outside the particular b digits are error free. For $1<b\leq 3$ such codes were first studied by Abramson [4,5]. The codes of Abramson are easy to implement and use much less redundancy than what is necessary to correct uncorrelated double and triple errors. These codes are particular instances of the general class of codes proposed by Fire [6], who gave a simple procedure to construct b-BEC-Codes for any $b>1$. Optimal and near optimal b-BEC codes have been studied by several authors [7,8,9,10,11] for $b\leq 5$. Also, bounds for burst error correction have been obtained [12] which relate the number of necessary check digits, k, in a b-BEC-Code of length, N, when the code is a linear code. It is easily shown that the number of necessary check digits in a b-BEC linear code is at least 2b [12]. If the distinct burst error patterns are counted in a "closed-loop" fashion (where the first and last digits of a code-word are treated as adjacent digits of the word) then it is true that $k > \text{Log}_2 (N+1) + b-1$ [8]. The code is said to be optimal if $k=\text{Log}_2(N+1)+b-1$. Clearly, for $b > 5$ the optimal codes will be quite long.

The usefulness of a BEC-Code for high speed memory applications, to correct transient errors, may be judged from the following three considerations.

a) The encoding and decoding circuit complexity for parallel instrumentation,

b) The burst coefficient, $\beta = (b/N)$.

and c) The redundancy, $\rho = (k/N)$.

Most of the known BEC-Codes (Codes of Metzner [13] are the only exception)

are linear cyclic codes. The Fire Codes [6] form the largest known class of linear BEC-Codes. These codes have very simple shift register circuitry [12] for serial encoders and decoders. These codes are also near optimal, and hence have low redundancy. However for $b > 3$ these codes have a low burst coefficient, β. Of course, the value of β could be increased by using shortened cyclic codes [12]. Such a shortened code is obtained by fixing at value zero some of the high order information digits in the code-word (the digits towards the right end of a code-word, written in the usual form). These zero valued digits may then be considered as not being part of the code-word. However, the parallel encoding and decoding circuitry for these codes are far from being simple. These codes were designed essentially for serial operation. Thus, the cost of encoding and decoding delays would rule out these codes as possible candidates for application to high speed memory systems.

In this paper I shall introduce a class of non-cyclic BEC-Codes having characteristics better suited for application to high speed memory systems. The codes have simple parallel instrumentation having delays not greater than about 10% of memory access time (say, of the order of 50 to 200 nanoseconds), about 1/3 redundancy, and a burst coefficient of about 1/k, where k is the number of check digits in the code.

A variant of these codes may be used as module error correcting codes to protect against catastrophic failure of a memory module, in a memory system with modular organization.

All the codes are here considered only for the binary case.

2.2. The Basic Concepts

The BEC and the MEC (Module error correcting) codes discussed in this paper are derived by suitable rearrangement of the digits of a specific class of

single error correcting codes, belonging to the class of codes known as, low-density parity check codes, first introduced by Gallagar [14]. These codes are also, self-orthogonal as per definition of Massey [3], and hence have very simple threshold decoders for error correction. For our purpose, the codes are best defined as follows:

Let $\{I_n\}$ be the set of integers, $\{0,1, \ldots,n-1\}$. For $0 \leqslant j \leqslant (k-1)$ let S_j be a proper subset of $\{I_n\}$, satisfying the following two (self-orthogonal) properties:

P1). For each i, $0 \leqslant i \leqslant n-1$, there exist exactly two subsets, $S'(i)$ and $S''(i)$, say $S'(i) = S_j$ and $S''(i) = S_\ell$, $j \neq \ell$, for some j and ℓ, $0 \leqslant j,\ell \leqslant (k-1)$, such that

$$S'(i) \cap S''(i) = \{i\}. \tag{1}$$

P2) Every pair of subsets S_j, S_ℓ, $0 \leqslant j,\ell \leqslant k-1$, $j \neq \ell$, has atmost one integer in common between them.

Thus, there will be a one-one correspondence between integers i, $0 \leqslant i \leqslant n-1$, and pairs of subsets (S_j, S_ℓ), $j \neq \ell$. Also, some pairs of subsets may not have any common elements. Therefore, we have

$$k(k-1) \geqslant 2n \tag{2}$$

When $k(k-1) = 2n$ we shall call the Code Maximal

Let $W = (x_0\ x_1 \ \ldots\ x_{n-1}\ a_0\ a_1 \ \ldots\ a_{k-1})$ be the code-word with n information digits $(x_0, \ldots, x_{n-1})$, and k check digits, $(a_0, \ldots a_{k-1})$. For $0 \leqslant j \leqslant k-1$, the check digit a_j is defined by

$$a_j = \sum_{i\ \epsilon S_j} x_i,\ j=0,1, \ldots, (k-1), \tag{3}$$

where Σ denotes modulo 2 summation (pair-wise exclusive-or). We shall denote by a'(i) and a''(i) the two check digits, dependent on x.: a'(i) and a''(i) are defined by the subsets S'(i) and S''(i), respectively, as per equation (3).

Let $\hat{W}$ $(\hat{x}_0\ \hat{x}_1 \ldots \hat{x}_{n-1}\ \hat{a}_0\ \hat{a}_1 \ldots \hat{a}_{k-1})$ denote a code-word read out of a memory.

Depending upon the error distribution over the word, either $\hat{x}_i = x_i$ or $\hat{x}_i \oplus 1$, where $\oplus$ denotes modulo 2 addition. In the latter case $\hat{x}_i$ is said to be in error. Let $\tilde{x}_i$ denote the decoded output information digit, $\tilde{x}_i = x_i$ if the output code-word had a correctable error distribution. For the class of codes defined by (3), where the set of all S_j's satisfy P1 and P2, the decoding rule is given by,

$$\tilde{x}_i = \text{MAJORITY } (\hat{x}_i, x_i', x_i'') \quad \ldots (4)$$

where

$$x_i' = \sum_{j \in S'(i)} \hat{x}_j \oplus a'(i) \oplus \hat{x}_i \quad \ldots (5)$$

and

$$x_i'' = \sum_{j \in S''(i)} \hat{x}_j \oplus a''(i) \oplus \hat{x}_i \quad \ldots (6)$$

Since the above codes are low-density codes (having a sparse parity-check matrix (See [15])) one might explore the following possibility: By suitably rearranging the digits of the code-word, W, and adopting the decoding scheme shown in figure 1, without unduly complicating the decoding network, may one obtain a useful BEC capability.?

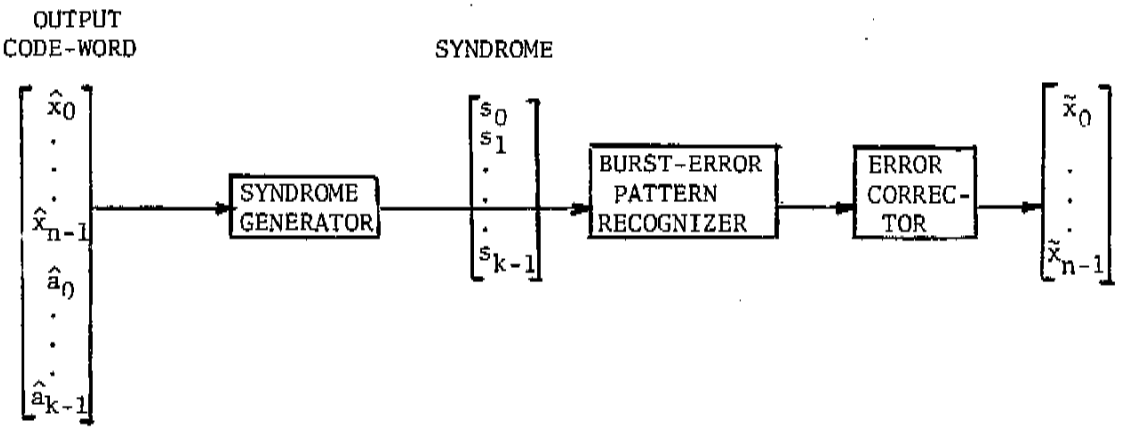


Fig. 1: Decoding Schema for Burst-error Correction

In the next two sections we shall discuss the BEC and MEC (Module-error Correcting) Codes so generated.

2.3. The BEC Codes

2.3(A) The BEC Codes derived from the maximal single error correcting low-density codes.

These codes have $2n = k(k-1)$ for odd $k > 5$. We shall denote by (n,k)-b-BEC code, a BEC code with n information digits and k check digits, capable of correcting all burst-errors of length $\leq b$ for some $b > 1$. The examples in Table I show the subsets defining the (78,13)-5-BEC, (36,9)-3-BEC and (21,7)-2-BEC codes, together with the digit arrangement in their respective code-words. Let us hereafter refer to the matrix of integers defining the subsets S_j for j=0,1, ..., k-1, as the code matrix. Table I shows the code matrices for k=13,9 and 7. The symbols λ in the code matrices, denote the null digits. Let me first develop the conditions for codes of this type to be b-BEC for some $b > 1$.

For $k > 5$ and odd, and $n = k(k-1)/2$ let $W^m(k)$ (m, for maximal) be the code-word of the (n,k)-b-BEC code, defined as follows:

$$W^m(k) = B_0^m(k)\ B_1^m(k) \ldots B_{k-1}^m(k) \qquad \ldots (6)$$

where $B_j^m(k)$ for $0 \leq j \leq k-1$ is a block of digits of the code-word, of the following form:

$$B_j^m(k) = x_{j(k-1)/2}\ x_{j(k-1)/2+1} \cdots x_{j(k-1)/2+(k-3)/2}\ a_{j+(k-1)/2} \qquad \ldots (7)$$

for $j = 0,1,\ldots,(k-1)$, where the subscript $(j+(k-1)/2)$ in $a_{j+(k-1)/2}$ is computed modulo k. The reader may easily verify that the code-words for k=13, 9 and 7, in Table I, satisfy the rules specified by (6) and (7). Let us denote the j^{th} digit in the i^{th} block of a code-word $W^m(k)$, by $b_{ij}^m(k)$.

In Table I notice that the subscripts of information digits belonging to anyone of the blocks of a code-word, all come from a single column in the code matrix defining the subsets of the code. This scheme, together with the concept

Table I: The Maximal BEC-Code for k=13.

S_0 = {	0	6	13	20	27	34	41	λ	53	58	63	68	73}
S_1 = {	1	6	12	19	26	33	40	47	λ	59	64	69	74}
S_2 = {	2	7	12	18	25	32	39	46	53	λ	65	70	75}
S_3 = {	3	8	13	18	24	31	38	45	52	59	λ	71	76}
S_4 = {	4	9	14	19	24	30	37	44	51	58	65	λ	77}
S_5 = {	5	10	15	20	25	30	36	43	50	57	64	71	λ}
S_6 = {	λ	11	16	21	26	31	36	42	49	56	63	70	77}
S_7 = {	5	λ	17	22	27	32	37	42	48	55	62	69	76}
S_8 = {	4	11	λ	23	28	33	38	43	48	54	61	68	75}
S_9 = {	3	10	17	λ	29	34	39	44	49	54	60	67	74}
S_{10} = {	2	9	16	23	λ	35	40	45	50	55	60	66	73}
S_{11} = {	1	8	15	22	29	λ	41	46	51	56	61	66	72}
S_{12} = {	0	7	14	21	28	35	λ	47	52	57	62	67	72}

Check Bits

$a_i = \sum_{j \in S_i} x_j$

i = 0, 1,..., 12.

λ denotes the "empty symbol"

Code-word: (91 digits long)	$(x_0 \ldots x_5\ a_6$	$x_6 \ldots x_{11}\ a_7$	.....	$x_{36} \ldots x_{41}\ a_{12}$	$x_{42} \ldots x_{47}\ a_0$	.....	$x_{72} \ldots x_{77}\ a_5$
	Block 1	Block 2		Block 7	Block 8		Block 13

(Table I Continued on next page)

Table I (continued) The Maximal BEC-Codes for k=9, and 7.

(36,9)-3-EBEC-Code

$S_0 = \{0\ 4\ 9\ 14\ 19\ \lambda\ 27\ 30\ 33\}$

$S_1 = \{1\ 4\ 8\ 13\ 18\ 23\ \lambda\ 31\ 34\}$

$S_2 = \{2\ 5\ 8\ 12\ 17\ 22\ 27\ \lambda\ 35\}$

$S_3 = \{3\ 6\ 9\ 12\ 16\ 21\ 26\ 31\ \lambda\}$

$S4 = \{\lambda\ 7\ 10\ 13\ 16\ 20\ 25\ 30\ 35\}$

$S_5 = \{3\ \lambda\ 11\ 14\ 17\ 20\ 24\ 29\ 34\}$

$S_6 = \{2\ 7\ \lambda\ 15\ 18\ 21\ 24\ 28\ 33\}$

$S_7 = \{1\ 6\ 11\ \lambda\ 19\ 22\ 25\ 28\ 32\}$

$S_8 = \{0\ 5\ 10\ 15\ \lambda\ 23\ 26\ 29\ 32\}$

Check Bits: $a_i = \sum_{j \epsilon S_i} x_j$, i=0, 1, ..., 8

Code-Word: $(x_0 \cdots x_3 a_4 \mid x_4 \cdots x_7 a_5 \mid \cdots \cdots \mid \cdots \mid x_{16} \cdots x_{19} a_8 \mid x_{20} \cdots x_{23} a_0 \mid \cdots \cdots \mid x_{32} \cdots x_{35} a_3)$

(21,7)-2-EBEC-Code

$S_0 = \{0\ 3\ 7\ 11\ \lambda\ 17\ 19\}$

$S_1 = \{1\ 3\ 6\ 10\ 14\ \lambda\ 20\}$

$S_2 = \{2\ 4\ 6\ 9\ 13\ 17\ \lambda\}$ Check Bits

$S_3 = \{\lambda\ 5\ 7\ 9\ 12\ 16\ 20\}$

$S_4 = \{2\ \lambda\ 8\ 10\ 12\ 15\ 19\}$ $a_i = \sum_{j \epsilon S_i} x_j$

$S_5 = \{1\ 5\ \lambda\ 11\ 13\ 15\ 18\}$ i = 0,1,...6

$S_6 = \{0\ 4\ 8\ \lambda\ 14\ 16\ 18\}$

$(x_0 x_1 x_2 x_3 \mid x_3 x_4 x_5 x_4 \mid \cdots\cdots \mid x_9 x_{10} x_{11} a_6$ Code-Word

$x_{12} x_{13} x_{14} a_0 \mid \cdots\cdots \mid x_{18} \cdots x_{20} a_2)$

of symmetry of integer arrangements in each column of the code matrix, as shown in Table I, in fact, defines this class of codes for all $k \geq 5$. In Table I the i^{th} column of a code matrix for $1 \leq i \leq (k-1)$, has the symmetry pattern, which is the i-digit downward cyclic shift of the 0^{th} column.

Let the syndrome for the code be defined by

$$y^m(k) = (y_0^m(k)\ y_1^m(k)\ \ldots\ y_{k-1}^m(k)) \quad \ldots \quad (8)$$

where

$$y_j^m(k) = \sum_{i \in S_j} \hat{x}_i \oplus \hat{a}_j$$

$$j = 0,1, \ldots, (k-1). \quad \ldots \quad (9)$$

Clearly, $y_j^m(k) = 1$, if there is an error in $\hat{x}_i$ for some $i \in S_j$, or an error in $\hat{a}_j$. Notice that the code-word is so arranged that for no two information digits, say x_i, x_j, belonging to a same block of the code-word, the subscript pair (i,j) belong to a same subset defined by the code matrix.

In the ensuing text we shall refer exclusively to codes of this type, when we use the phrase 'k-MBEC Code' (M for maximal). Every k-MBEC Code would contain $k(k-1)/2$ information digits, where k is odd and greater than 5, and would correct all burst-errors of length $\leq$ b, for some $b > 1$. The value of b corresponding to a given k, is as yet undetermined.

Let Y_{ij}^m (k,b) for $j = 0,1, \ldots, (k-1)/2$ and $i = 0,1, \ldots, (k-1)$ be the sets of syndromes associated with burst-errors of length $\leq$ b, which begin at the digit b_{ij}^m (k) (the j^{th} digit of the i^{th} block) of the code-word. Each $Y_{ij}^m(k,b)$ should contain 2^{b-1} distinct syndromes if the k-MBEC code is to correct all burst-errors of length $\leq$ b. We shall, hereafter, simply use Y_{ij}^m to denote Y_{ij}^m (k,b), whenever such use is unambiguous. To obtain a convenient representation of the members of Y_{ij}^m, let me introduce some additional notation.

Let $\alpha, \beta, \nu, \delta$, etc., with or without subscripts, denote arbitrary bits, 0 or 1. Let $\alpha^{(j)}$ denote any bit string of length j. Finally, let $\dot{\alpha}^{(j)}$ denote

$\alpha^{(j)}$ in reverse order. Thus, if (1001100) is $\alpha^{(7)}$ then (0011001) is $\dot{\alpha}^{(7)}$. Let $0^{(i)}$ ($1^{(i)}$) denote a string of i zeros (ones). The reader may now easily verify the following lemmas.

LEMMA 1: The syndromes belonging to $Y^m_{0j}(k,b)$ for $0 \leq j \leq (k-1)/2$, are of the form shown in Table II. For $0 \leq i \leq (k-1)$, $Y^m_{ij}(k,b)$ is given by,

$$Y^m_{ij} = R_i(Y^m_{0j}), \qquad \ldots (10)$$

for $0 \leq j \leq (k-1)/2$ and $0 \leq i \leq (k-1)$,

where $R_i(Y^m_{0j})$ is the set obtained by i-digit cyclic right shift of each member of Y^m_{0j}.

Table II. The Syndrome Sets Y^m_{0j} (k,b)

Range of j	Syndrome bit Patterns	Values of indicies used in bit patterns
$0 \leq j < (k-2b+1)/2$	$0^{(j)}\ 1\ \alpha^{(b-1)}\ 0^{(i)}\ \dot{\alpha}^{(b-1)}\ 1\ 0^{(j)}$	$i = (k-2b-2j)$
$j = (k-2b+1)/2$	$0^{(j)}\ 1\ \alpha^{(b-2)}\beta\ \ \dot{\alpha}^{(b-2)}\ 1\ 0^{(j)}$	———
$(k-2b+1)/2 < j < (k-1)/2$	$\alpha\ \alpha\ \beta^{(i)} 0\ 1\ \gamma^{(m)} \dot{\gamma}^{(m)}\ 1\ 0^{(d)}\ \dot{\beta}^{(i)}$	$m = (k-1)/2-(j+1)$ $i = b-m-3$ $d = (k-2b)$
$j = (k-1)/2$	$\alpha\alpha\ \gamma^{(i)}\ 0\ 1\ 0^{(d)}\ \dot{\gamma}^{(i)}$	$i = (b-2)$ $d = (k-2b)$

Thus, a member of $Y^m_{01}(13,5)$

is,

$$(0\ 1\ \alpha^{(4)}\ 0\ \dot{\alpha}^{(4)}\ 1\ 0)$$

and the corresponding member of $Y^m_{21}(13,5)$ is

$$R_2(01\ \alpha^{(4)}\ 0\ \dot{\alpha}^{(4)}\ 10) = (1001\alpha^{(4)}\ 0\ \dot{\alpha}^{(4)}) \quad \ldots \quad (11)$$

The syndrome sets $Y^m_{0j}(13,5)$ for $j = 0, 1, \ldots, 6$, are shown in Table III.

Table III. The Syndrome Sets Y^m_{0j} (13,5)

j	Syndrome bit Patterns
0	$1\ \alpha^{(4)}\ 0^{(3)}\ \dot{\alpha}^{(4)}\ 1$
1	$0\ 1\ \alpha^{(4)}\ 0\ \dot{\alpha}^{(4)}\ 1\ 0$
2	$0\ 0\ 1\ \alpha^{(3)}\ 1\ \dot{\alpha}^{(3)}\ 1\ 0\ 0$
3	$\alpha\ \alpha\ 0\ 1\ \beta^{(2)}\gamma\ \dot{\beta}^{(2)}\ 1\ 0\ 0\ 0$
4	$\alpha\ \alpha\ \beta\ 0\ 1\ \gamma\ \delta\ \gamma\ 1\ 0\ 0\ 0\ \beta$
5	$\alpha\ \alpha\ \beta^{(2)}\ 0\ 1\ \gamma\ 1\ 0\ 0\ 0\ \dot{\beta}^{(2)}$
6	$\alpha\ \alpha\ \beta^{(3)}\ 0\ 1\ 0\ 0\ 0\ \dot{\beta}^{(3)}$

LEMMA 2 A k-MBEC code is a b-BEC code for some $b > 1$, if and only if for $i = 0,1, \ldots, (k-1)$ and $j = 0,1, \ldots, (k-1)/2$, it is true that,

i) The number of distinct members of $Y^m_{ij}(k,b)$ is 2^{b-1}, and

ii)
$$Y^m_{ij}(k,b) \cap R_t(Y^m_{0\ell}(k,b)) = \Phi \quad \ldots \quad (12)$$
$$\text{for } 0 \leq t \leq (k-1)$$

where Φ is the empty set, except when $i = t$ and $j = \ell$.

In the case of the k-MBEC codes, for obvious reasons*, conditions (i) and (ii) of Lemma 2 restricts interesting cases (cases with significant values of b) to prime values of k. For prime k upto 79 Table IV lists the corresponding values of b, for which the k-MBEC code is a b-BEC code.

The values of b shown in Table IV were determined by exhaustive computer search: For each k, Table IV shows the largest value of b for which the conditions of Lemma 2 were not violated. We have not been so far able to obtain an analytical expression for b in terms of k.

An interesting generalization of codes of this type appears when the number of information digits in a code-word is restricted to be less than k(k-1)/2. This class of codes is discussed in the next section.

2.3.(B): The Non-Cyclic, k-NCBEC Codes

Given an odd k > 7, the k-NCBEC (NC for Non Cyclic) code with k check digits is defined as follows:

Choose

$$b = \lfloor ((k-2)/3) \qquad \ldots \quad (13)$$

where $\lfloor ((k-2)/3) \rfloor$ is the largest integer less than or equal to (k-2)/3 (called the floor of (k-2)/3). Let

$$[M_0(k)]^T = [0\ 1\ \ldots (b-1)\ \underbrace{\lambda\ \lambda\ \ldots\ \lambda}_{k-2b}\ (b-1)\ \ldots\ 1\ 0] \qquad \ldots \quad (14)$$

where $[M_0(k)]^T$ is the transpose (T for transpose) of the column vector $M_0(k)$. Given an integer i, let $(M_0(k) + i)$ be the vector obtained by adding i to all the non-null digits of $M_0(k)$. Also, let $D_i(M_0(k))$ denote the vector obtained by i-digit downward cyclic shift of $M_0(k)$. Thus, $[D_i\ (M_0(k))]^T$ would be

---

* If k is not a prime then syndromes would include periodic sequences, which would violate both conditions of Lemma 2.

Table IV: The burst length b, correctable by a k-MBEC Code, for prime values of k, $11 \leqslant k \leqslant 79$.

Number of check digits: k	Number of information digits: $\frac{k(k-1)}{2}$	Max. correctible burst length
11	55	2
13	78	5
17	136	5
19	171	7
23	253	7
29	406	9
31	465	10
37	666	11
41	820	13
43	903	14
47	1081	14
53	1378	17
59	1711	19
61	1830	20
67	2211	23
71	2485	16
73	2628	18
79	3081	22

$$[D_i(M_0(k))]^T = [(i-1)\ (i-2)\ \ldots\ 1\ 0\ 0\ 1\ \ldots (b-1)\ \underbrace{\lambda\ \lambda \ldots\ \lambda}_{k-2b}$$
$$(b-1)(b-2)\ \ldots\ i] \qquad \ldots \quad (15)$$

For $1 \leq i \leq (k-1)$ let

$$M_i(k) = D_i(M_0(k) + b.i) \qquad \ldots \quad (16)$$

and let the code-matrix, [M (k)] be composed of the k column vectors $M_i(k)$ for $0 \leq i \leq (k-1$ as shown below:

$$[M(k)] = M_0(k)\ M_1(k)\ \ldots\ M_{k-1}(k). \qquad \ldots \quad (17)$$

Also, for $0 \leq j \leq k-1$ let the jth row of [M(k)] be the subset, S_j, which defines the check bit, a_j, of the code-word as per equation (3). Let the code-word itself be again composed of (k-1) blocks,

$$W(k) = B_o(k)\ B_1(k)\ \ldots\ B_{k-1}(k), \qquad \ldots \quad (18)$$

where for $0 \leq i \leq (k-1)$,

$$B_i(k) = x_{bi}\ x_{(bi+1)}\ \cdots\ x_{(bi+b-1)}\ a_{(k-b+i-1)} \qquad \ldots \quad (19)$$

where the subscript k-b+i-1 of $a_{(k-b+i-1)}$ is computed modulo k. For k=9 the code-word would be,

$$W(9) = (x_0x_1a_6x_2x_3a_7x_4x_5a_8x_6x_7a_0$$
$$x_8x_9a_1x_{10}x_{11}a_2x_{12}x_{13}a_3x_{14}x_{15}a_4x_{16}x_{17}a_5) \qquad \ldots \quad (20)$$

The following lemmas are true for the NCBEC codes:

LEMMA 3: The syndromes belonging to the sets $Y_{0j}(k,b)$, for $0 \leq j \leq b$, are of the form shown in Table V. For $0 \leq i \leq k-1$,

$$Y_{ij}(k,b) = R_i\ (Y_{0j}(k,b)) \qquad \ldots \quad (21)$$
$$j = 0,1,2,\ldots,b.$$

Table V The Syndrome Sets Y_{0j} (k,b)

Range of j		Syndrome bit Patterns $b=\lfloor((k-2)/3\rfloor$	# of distinct sequences	Value of indicies used in bit Patterns
$j = 0$	Y_{00}	$\{1\ \alpha^{(b-1)}\ 0^{(i)}\ \dot{\alpha}^{(b-1)}\ 1\}$	2^{b-1}	
$0 < j < b$	Y_{0j}	$\{\alpha\ \alpha\beta^{(j-2)}\ 1\ \gamma^{(b-1-j)}\ 0^{(i-1)}\delta\ \dot{\gamma}^{(b-i-j)}1\ 0\ 0\ \dot{\beta}^{(j-2)}\}$	$2^{(b-1)}$	$i = (k-2b)$
$j = b$	Y_{0b}	$\{\alpha\ \alpha\ \beta^{(b-2)}0^{(i-1)}100\dot{\beta}^{(b-2)}\}$	2^{b-1}	

LEMMA 4: A k-NCBEC code is a (kb,k)-b-BEC code if and only if for $0 \leq i \leq (k-1)$ and $0 \leq j \leq b$,

i) The number of distinct elements of $Y_{ij}(k,b)$ is 2^{b-1}, and for all t and ℓ

ii) $$Y_{ij}(k,b) \cap Y_{t\ell}(k,b) = \Phi \quad \ldots \quad (22)$$

except when $i = t$ and $j = \ell$.

From lemmas 3 and 4, lemma 5, follows:

LEMMA 5: A k-NCBEC code is a (kb,k)-b-BEC code if and only if for $0 \leq i \leq (k-1)$ and $0 \leq j \leq b$ it is true that

i) The number of distinct elements of $Y_{ij}(k,b)$ is 2^{b-1} and

ii) $$Y_{0j}(k,b) \cap R_t(Y_{0\ell}(k,b)) = \Phi \quad \ldots \quad (23)$$

for $0 \leq \ell \leq b$, except when $t = 0$ and $j = \ell$.

Using lemma 5, and the forms of syndromes shown in Table V, it can be shown that,

THEOREM 1: A k-NCBEC code is a (kb,k)-b-BEC code for $b=\lfloor((k-2)/3)$.

The proof of theorem 1 is shown in Appendix I. NCBEC codes have simple parallel decoding networks. For details on the decoders for these codes please see [15].

3. The Module Error Correcting Codes

These are actually variants of the k-MBEC codes, where k is a prime number, $k > 5$. The number of information digits in a code-word is $k(k-3)/2$. A typical code for $k = 11$ is shown in Table VI. Here also the code-words consists of k blocks,

$$W'(k) = B'_0(k) \quad B'_1(k) \quad \ldots \quad B'_{k-1}(k), \quad \ldots \quad (24)$$

Table VI: The k-MEC Code for k = 11

CODE MATRIX: $[M](k)] =$

$$\begin{bmatrix} 0 & 4 & 9 & 14 & 19 & \lambda & \lambda & \lambda & 35 & 38 & 41 \\ 1 & 4 & 8 & 13 & 18 & 23 & \lambda & \lambda & \lambda & 39 & 42 \\ 2 & 5 & 8 & 12 & 17 & 22 & 27 & \lambda & \lambda & \lambda & 43 \\ 3 & 6 & 9 & 12 & 16 & 21 & 26 & 31 & \lambda & \lambda & \lambda \\ \lambda & 7 & 10 & 13 & 16 & 20 & 25 & 30 & 35 & \lambda & \lambda \\ \lambda & \lambda & 11 & 14 & 17 & 20 & 24 & 29 & 34 & 39 & \lambda \\ \lambda & \lambda & \lambda & 15 & 18 & 21 & 24 & 28 & 33 & 38 & 43 \\ 3 & \lambda & \lambda & \lambda & 19 & 22 & 25 & 28 & 32 & 37 & 42 \\ 2 & 7 & \lambda & \lambda & \lambda & 23 & 26 & 29 & 32 & 36 & 41 \\ 1 & 6 & 11 & \lambda & \lambda & \lambda & 27 & 30 & 33 & 36 & 40 \\ 0 & 5 & 10 & 15 & \lambda & \lambda & \lambda & 31 & 34 & 37 & 40 \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \\ S_9 \\ S_{10} \end{bmatrix}$$

CODE-WORD: $W'(k) = (x_0x_1x_2x_3a_5x_4x_5x_6x_7a_6 \quad \ldots \quad x_{24}x_{25}x_{26}x_{27}a_0 \quad \ldots \quad x_{40}x_{41}x_{42}x_{43}a_4)$

where

$$B_i'(k) = (x_{i.(k-3)/2} \quad x_{i.(k-3)/2+1} \quad \cdots \quad x_{i.(k-3)/2+(k-5)/2} \quad a_{(k-1)/2+i}) \quad \cdots \quad (25)$$

where the subscript $((k-1)/2 + i)$ in $a_{(k-1)/2+i}$ is computed modulo k. This code can be used to correct all errors which are confined to any one of the blocks in the code-word. We shall call such errors as single block (or module) errors. Thus, if a memory is organized in terms of k modules each module containing a block of a code-word, then this code may be used to protect against a catastrophic failure of any one of the modules. Typically for k=19, one would have 19 modules, each containing a block of the code-word consisting of 8 information bits and 1 check bit.

We have for the MEC codes the following theorem, the proof of which is similar to the proof of Theorem 1 in Appendix I.

THEOREM 2: A k-MEC code for prime values of k, is a single block error correcting code, where the blocks are defined as shown in (25).

The decoding net complexity of a k-MEC code, for parallel decoding, compares very favourably with that of the single error correcting Hamming [12] codes. To get a fair comparison, codes in both cases were chosen to have the same number of information digits.

The following assumptions were made:

a) The basic building block for the decoding net was a 5-input majority gate, with a gate delay of δ nanoseconds,

b) Each 3-input (or 2-input) modulo 2 summer is equivalent to two such gates, with a total delay of 2δ nano-seconds, and

c) k-input summers were built out of 3-input summers as building blocks.

Let S(k) be the number of 3-input summers necessary to build a k-input summer. Then, it can be shown that S(k) is bounded by

$$1/2\ (3^{m-1}-1) + (m-2) < S(k) < 1/2\ (3^{m}-1) \qquad \ldots \quad (26)$$

where $m = \lceil \text{Log}_3 k \rceil$, the smallest integer greater than $\text{Log}_3 k$. Similarly, let T(k) be the gate count for a k-input OR-function using 5-input OR-gates as building blocks. Then

$$1/2\ (5^{n-1}-1) + (n-4) < T(k) < 1/2\ (5^{n}-1) \qquad \ldots \quad (27)$$

where $n = \lceil \text{Log}_5 k \rceil$.

The delay in the case of the k-input summer would be,

$$D_1(k) < 2\,\delta\ \text{Log}_3 k + 2, \qquad \ldots \quad (28)$$

and in the case of the k-input OR-net would be

$$D_2(k) < \delta\ \text{Log}_5 k + 1 \qquad \ldots \quad (29)$$

With these basic observations, it can be shown that a parallel decoder for a k-MEC single module error correcting decoder would have a total of N(k) gates, with a total delay of D(k) nanoseconds, where

$$N(k) = 2kS(k-2) + kT(k+1/2) + kT(k-1) + 2k(k-3) \qquad \ldots \quad (30)$$

and

$$D(k) = \delta\ (\text{Log}_3(k-2) + \text{Log}_5(k+1/2) + \text{Log}_5(k-1) + 8) \qquad \ldots \quad (31)$$

For k = 19,

$$N(19) = 1140$$

and

$$D(19) = 15\delta\ , \qquad \ldots \quad (32)$$

and in this case the k-MEC code would have 152 information digits. For the same number of information digits a Hamming single error correcting code would

have 8 check bits. A parallel decoder for this single error correcting code would also have about 1140 gates. The decoding delay would be 12δ.

Thus, for a small increase in the decoding delay and about twice the number of check bits, one could obtain single module error correction instead of single bit error correction. It should be pointed out that the k-MEC code could also detect several of the uncorrectable multiple module errors, and this error detection feature may be introduced with little additional cost to the parallel decoding net.

The k-MEC codes are probably ideal candidates for use in cache transactions, between cache-memory and main memory, in systems with the cache memory organizations [16]. Their only disadvantage is that they need a prime number of modules instead of a number which is a power of 2.

4. Multiple Error Correcting Majority Codes.

We shall refer to these simply as the M-Codes. The examples shown in Tables VII through X, illustrate the basic schemes for the construction of these codes.

The code in Table VII has the following characteristics:

Let t be the number of randomly distributed digit errors to be corrected in a code-word. For any prime $p \geq (2t-1)$, one may then construct a M-code consisting of

$$n = p^2 + 2t.\ \lfloor((p-1)/(2t-1)\rfloor \quad \ldots \quad (33)$$

information digits, and

$$k = 2tp \quad \ldots \quad (34)$$

check digits. For the code in Table VII $t = 2$, $p = 5$, $n = 29$ and $k = 20$. (The code for $t = 2$ and $p = 3$, following the same scheme is shown in Table VIII).

Table VII: M-Code for t = 2, and p = 5.

Block 1		Block 2	
S_1	= { 1, 2, 3, 4, 5, 26}	S_6	= {1, 6, 11, 16, 21, 27}
S_2	= { 6, 7, 8, 9, 10, 26}	S_7	= {2, 7, 12, 17, 22, 27}
S_3	= { 11, 12, 13, 14, 15, 26}	S_8	= {3, 8, 13, 18, 23, 27}
S_4	= { 16, 17, 18, 19, 20, 26}	S_9	= {4, 9, 14, 19, 24, 27}
S_5	= { 21, 22, 23, 24, 25 }	S_{10}	= {5, 10, 15, 20, 25, }

Block 3		Block 4	
S_{11}	= { 1, 10, 14, 18, 22, 28}	S_{16}	= {1, 9, 12, 20, 23, 29}
S_{12}	= { 2, 6, 15, 19, 23, 28}	S_{17}	= {2, 10, 13, 16, 24, 29}
S_{13}	= { 3, 7, 11, 20, 24, 28}	S_{18}	= {3, 6, 14, 17, 25, 29}
S_{14}	= { 4, 8, 12, 16, 25, 28}	S_{19}	= {4, 7, 15, 18, 21, 29}
S_{15}	{ 5, 9, 13, 17, 21 }	S_{20}	= {5, 8, 11, 19, 22, }

For $j = 1, 2, \ldots, 20$ the check digit,

$$a_j = \sum_{i \in S_j} x_i$$

Code-Word: $(x_1 x_2 \ldots . x_{29} a_1 a_2 \ldots a_{20})$

Table VIII: M-Code for t=2 and p=3.

Block 1	Block 2
$S_1 = \{1, 2, 3\}$	$S_4 = \{1, 4, 7\}$
$S_2 = \{4, 5, 6\}$	$S_5 = \{2, 5, 8\}$
$S_3 = \{7, 8, 9\}$	$S_6 = \{3, 6, 9\}$
Block 3	**Block 4**
$S_7 = \{1, 6, 8\}$	$S_{10} = \{1, 5, 9\}$
$S_8 = \{2, 4, 9\}$	$S_{11} = \{2, 6, 7\}$
$S_9 = \{3, 5, 7\}$	$S_{12} = \{3, 4, 8\}$

Code-Word: $(x_1x_2 \ldots x_9a_1a_2 \ldots a_{12}$

Table IX: M'-Code for t=2 and m = 3.

Block 1	Block 2
$S_1 = \{1, 2, 3, 10\}$	$S_4 = \{1, 4, 7, 11\}$
$S_2 = \{4, 5, 6, 10\}$	$S_5 = \{2, 5, 8, 11\}$
$S_3 = \{7, 8, 9, 10\}$	$S_6 = \{3, 6, 9, 11\}$
Block 3	**Block 4**
$S_7 = \{1, 6, 8, 12\}$	$S_{10} = \{1, 5, 9, 13\}$
$S_8 = \{2, 4, 9, 12\}$	$S_{11} = \{2, 6, 7, 13\}$
$S_9 = \{3, 5, 7, 12\}$	$S_{12} = \{3, 4, 8, 13\}$
$S_{13} = \{10,11,12,13\}$.	

Code-Word: $(x_1x_2 \ldots x_{13}a_1a_2 \ldots a_{13})$

The pattern of arrangement of integers in Table VII may be followed by noting the distribution of the underlined integers, 1, 6, 11, 16 and 21 in the table. The check bits for the code are again defined by equation (3), in terms of the subsets, S_j, for $0 \leq j \leq k-1$. The arrangement of digits in the various subsets satisfy the following conditions:

a) Each digit appears in exactly 2t subsets,

b) No two subsets have more than one digit in common between them.

These conditions make the codes self-orthogonal. Hence, the following decoding scheme may be adopted (See [4]):

For a given information digit x_i, $0 \leq i \leq n-1$, consider the 2t subsets $S^1(i)$, $S^2(i)$ ..., $S^{2t}(i)$, which contain the integer i. Let $a^1(i)$, $a^2(i)$, . . ., $a^{2t}(i)$ be the check digits defined by these subsets, as per equation (3). Let $\hat{x}_i$, $\hat{a}^1(i)$, $\hat{a}^2(i)$, . . ., $\hat{a}^{2t}(i)$ be the digits obtained from a code-word, read out of a memory. Then, the decoder can make the following (2t +1) independent estimates of x_i:

$$
\begin{aligned}
E_1 &: \hat{x}_i \quad , \\
E_2 &: \hat{a}^1(i) \oplus \sum_{\{j \in S^1(i) \wedge j \neq i\}} \hat{x}_j \quad , \\
E_3 &: \hat{a}^2(i) \oplus \sum_{\{j \in S^2(i) \wedge j \neq i\}} \hat{x}_j \quad , \qquad \ldots \quad (35) \\
&\vdots \\
E_{2t} &: \hat{a}^{2t}(i) \oplus \sum_{\{j \in S^{2t}(i) \wedge j \neq i\}} \hat{x}_j \quad ;
\end{aligned}
$$

These (2t+1) estimates are mutually independent because, by conditions a) and b) given above, a digit used in making one of the estimates is never used again in any of the other estimates. Since, the code is expected to correct atmost only t errors, it follows that, in any given instance, only t of the above (2t+1) estimates could be in error. Hence, the majority of these (2t+1) estimates should give the correct value of x_i, under the assumed error conditions. The parallel decoder for the code would then consist of 2t estimators and a (2t+1)-input majority gate, for each one of the information digits in the code-word.

The scheme presented in Table IX is for the special case where (2t-1) is itself a prime number. Thus, in the case t=2, and m=(2t-1) = 3, the code in Table IX shows a variant of the code in Table VIII. The code of Table VIII has 12 check digits and 9 information digits, and that in Table IX has 13 each of information and check digits. Here, by adding an additional check digit, we have been able to accomodate four additional information digits. A similar code for m=(2t-1) = 5 and t = 3 is shown in Table X. These codes have the following characteristics:

i) m = (2t-1) is a prime number.

ii) The number of check digits is,

$$k = 2tm + 1$$

and iii) The number of information digits is

$$n = m^2 + 2t = k$$

I believe that for a given t and k, the above scheme accomodates the maximum possible number of information digits in the code-word, if the code itself satisfies the self-orthogonal conditions a) and b) given above. I have not so far been able to prove this contention.

Table X M'-Code for t = 3 and m = 5

Block 1

S_1 = { 1, 2, 3, 4, 5, 26}
S_2 = { 6, 7, 8, 9, 10, 26}
S_3 = {11, 12, 13, 14, 15, 26}
S_4 = {16, 17, 18, 19, 20, 26}
S_5 = {21, 22, 23, 24, 25, 26}

Block 2

S_6 = {1, 6, 11, 16, 21, 27}
S_7 = {2, 7, 12, 17, 22, 27}
S_8 = {3, 8, 13, 18, 23, 27}
S_9 = {4, 9, 14, 19, 24, 27}
S_{10} = {5, 10, 15, 20, 25, 27}

Block 3

S_{11} = {1, 10, 14, 18, 22, 28}
S_{12} = {2, 6, 15, 19, 23, 28}
S_{13} = {3, 7, 11, 20, 24, 28}
S_{14} = {4, 8, 12, 16, 25, 28}
S_{15} = {5, 9, 13, 17, 21, 28}

Block 4

S_{15} = {1, 9, 12, 20, 23, 29}
S_{17} = {2, 10, 13, 16, 24, 29}
S_{18} = {3, 6, 14, 17, 25, 29}
S_{19} = {4, 7, 15, 18, 21, 29}
S_{20} = {5, 8, 11, 19, 22, 29}

Block 5

S_{21} = {1, 8, 15, 17, 24, 30}
S_{22} = {2, 9, 11, 18, 25, 30}
S_{23} = {3, 10, 12, 19, 21, 30}
S_{24} = {4, 6, 13, 20, 22, 30}
S_{25} = {5, 7, 14, 16, 23 30}

Block 6

S_{26} = {1, 7, 13, 19, 25, 31}
S_{27} = {2, 8, 14, 20, 21, 31}
S_{28} = {3, 9, 15, 16, 22, 31}
S_{29} = {4, 10, 11, 17, 23, 31}
S_{30} = {5, 6, 12, 18, 24, 31}

S_{31} = {26, 27, 28, 29, 30, 31}

Code-Word: $(x_1x_2 \ldots x_{31}a_1a_2 \ldots a_{31})$

Acknowledgement:

My discussions with R. O. Winder were very helpful in developing and proving the scheme for the NCBEC-Codes. Winder constructed counter examples to refute several of my earlier conjectures on the NCBEC-Codes, and also suggested the organization of the proof presented in Appendix I.

References

1. Srinivasan, C. V., "Code for Error Correction in High Speed Memory Systems: Part I correction of cell defects in integrated Memories": Submitted to IEEE-EC in April 1970, for possible publication.

2. Townsend, R. L. and Weldon, E. J., Jr., "Self-orthogonal Quasi-Cyclic Codes" IEEE - Vol. IT-12, #2, Ap. 1966, p. 278.

3. Massey, J. L., "Threshold Decoding" M.I.T. Press, Cambridge, Mass. 1963.

4. Abramson, M. M., "Error Correcting Codes from Linear Sequential Networks", Proc. 4th London Symposium on Information Theory, Ed. by C. Cherry, Butterworths, Washington, D. C., 1961.

5. Abramson, M. M., "A Note on Single Error Correcting Binary Codes", IRE-PGIT-6, pp. 502-503, September 1960.

6. Fire, P., "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors", Technical Report No. 55, Stanford Research Lab., Stanford, California, April 24, 1959.

7. Elspas, B., and Short, R. A., "A Note on Optimum Burst Error Correcting Codes", IRE-PGIT-8, pp. 39-42, January 1962.

8. Elspas, B., "A Note on P-nary Adjacent Error Correcting Codes", IRE-PGIT-G, pp. 13-15, (1960).

9. C. R. Foulk, "Some Properties of Maximally Efficient Cyclic BEC-Codes and Results of A Computer Search for Such Codes", Digital Computer Lab. Report, File No. 375, University of Illinois, Urbana, June 12, 1961.

10. Gross, A. J., "Binary Group Codes which Correct Bursts of Three or Less for Odd Redundancy", IRE-PGIT-8, pp. 356-359, October 1962.

11. Gross, A. J., "A Note on Some Binary Group Codes Which Correct Bursts of Four or Less", IRE-PGIT-8, pp. 384, October 1962.

12. Peterson, W. W., "Error-Correcting Codes", the MIT Press, Massachusetts Institute of Technology, Cambridge, Mass., 1961.

13. Metzner, John J., "Burst Error-Correction for Randomly Chosen Binary Group Codes, "IEEE Trans., IT-9, No. 4, pp. 281-285, October 1963.

14. Gallager, R. G., "Low-density Parity Check Codes", M.I.T. Press, Cambridge, Mass. 1963

15. Srinivasan, C. V.: Patent No. 3478 313, "System for Automatic Correction of Burst Errors"

16. Conti, C. J., Gibson, D. H. and Pitkowsky, S. H., "Structural Aspects of System 360, Model 85", IBM Systems Journal, Vol. 7, # 1, 1968.

APPENDIX I

Proof Of Theorem 1

In Table V we have basically two types of sequences. :

1. Sequences of even parity of the forms a) and b) shown below:

a) $Y_{00} = \{1\ \alpha^{(b-1)}\ 0^{(c+1)}\ \dot{\alpha}^{(b-1)}\ 1\}$ . . . (I-1)

and* b) $Y_{jj} = \{00\ \beta^{(j-1)}\ \dot{\beta}^{(j-1)}\ 1\ \alpha^{(b-j-1)}\ 0^{(c+1)}\ \dot{\alpha}^{(b-j-1)}$. (I-2)

where $0 < j < b$ and $c = k-2b-1$, c even.

2) Sequences of odd parity of the form

$$Y_{(b+c+1)j} = 0^{(c)}\ 1\ \alpha^{(b-j)}\ 00\ \beta^{(j-1)}\ \dot{\beta}^{(j-1)}\ \dot{\alpha}^{(b-j)} \quad . . . \quad (I-3)$$

For $c = k-2b-1$, c even, and $0<j\leqslant b$. In (I.3) when $j = b$, $\alpha^{(b-j)} = \lambda$, the null sequence.

If $j \neq b$ then $\alpha^{(b-j)}$ should end with a 1. Accordingly, the proof is in two parts.

To prove Theorem 1, it is sufficient if we prove Lemma 5: Let us denote by e_{ij} a syndrome sequence of even parity belonging to Y_{ij}. Similarly, let d_{ij} denote a syndrome sequence of odd parity belonging to Y_{ij}. Clearly, there are no e_{0b} and d_{00}. Also, let e and d denote arbitrary syndrome sequences of even and odd parity respectively.

Assertion 1: Every e has a unique characteristic block of the form $(1\ 0^{(a)}\ 1)$, where $a \geqslant (c+1)$, $c = k-2b-1$, and a is odd. (See (I-1) and (I-2)).

---

* Notice that $Y_{jj} = R_j(Y_{0j})$, where Y_{0j} is shown in Table V for $0 < j < b$. Also, in Y_{jj} above the variable δ of Table V has been set to zero.

Proof: Follows from the following observations:

i) For $0 \leq j < b$

$(b-j-1) \leq b-1 < (c+1)$ and $j+1 \leq b < (c+1)$,

ii) The pattern $(100\ \beta^{(j-1)}\dot{\beta}^{(j-1)}1)$ in (I-2) cannot include $(1\ 0^{(a)}\ 1)$ since 'a' is odd, and

iii) $(10^{(a)}\ 1)$ cannot straddle the patterns $(\dot{\beta}^{(j-1)}1\ \alpha^{(b-j-1)})$ and $(\dot{\alpha}^{(b-j-1)}\ 1\ \beta^{(j-1)})$ appearing in (I-2).

Assertion 2: $e_{0r} \neq R_\ell(e_{0t})$

for any ℓ, $0 \leq \ell \leq k-1$ and $0 \leq r,t \leq (b-1)$, unless $\ell = 0$, $r = t$ and $e_{0r} = e_{0t}$.

Proof: If $e_{0r} = R_\ell(e_{0t})$ then the characteristic blocks of the form $(1\ 0^{(a)}\ 1)$ in the two sequences should line up with each other because of assertion 1. Hence, $\ell = 0$, and $e_{0r} = e_{0t}$. Now, if we proceed from the center of this block and scan in both directions, the distance, s_r, to the farthest pair of symmetric 1's is exactly determined by the explicit 1's in (I-1) and (I-2) in the pattern $(1\ \alpha^{(b-j-1)}\ 0^{(c+1)}\ \dot{\alpha}^{(b-j-1)}\ 1)$:

$$s_r = c/2 + (b-r-1) \qquad \ldots \quad (I.4)$$

If $\ell = 0$, and $e_{0r} = e_{0t}$ then $s_r = s_t$ and by (I-4) $r = t$.

For sequences of odd parity the proof consists of several parts, and is quite long. In this proof we shall use the '1' in the pattern '$0^{(c)}\ 1$', appearing in (I-3) as a reference marker, and denote its position within a syndrome sequence by letters I or J.

Assertion 3: $d_{0r} \neq R_\ell(d_{0t})$

for any ℓ, $0 \leq \ell \leq k-1$, and $0 < r,t \leq b$, unless $\ell = 0$, $r = t$ and $d_{0r} = d_{0t}$.

Proof: Suppose

$$d_{0r} = R_\ell(d_{0t}) \qquad \ldots \quad (I\text{-}5)$$

for some ℓ. Then, let

$$d_{0r} = (0^{(c)}\ \overset{I}{1}\ \alpha^{(b-r)}\ 00\ \beta^{(r-1)}\ \dot{\beta}^{(r-1)}\ \dot{\alpha}^{(b-r)}) \qquad \ldots \quad (I\text{-}6)$$

and $$d_{0t} = (0^{(c)}\ \underset{J}{1}\ \gamma^{(b-t)}\ 00\delta\ ^{(t-1)}\ \dot{\delta}^{(t-1)}\ \dot{\gamma}^{(b-t)}), \qquad \ldots \quad (I\text{-}7)$$

both sequences being of the form (I-3). We then have

$$0^{(c)}\ I\ \alpha^{(b-r)}\ 00\ \beta^{(r-1)}\ \dot{\beta}^{(r-1)}\ \dot{\alpha}^{(b-r)} = d_{0r} = R_\ell(d_{0t}) = R_\ell(0^{(c)}{}_J\ \gamma^{(b-t)}\ 00\delta\ ^{(t-1)}\ \dot{\delta}^{(t-1)}\ \dot{\gamma}^{(b-t)}), \qquad \ldots \quad (I\text{-}8)$$

where $\alpha^{(b-r)} = \lambda$ the null sequence, or it ends in a 1, and likewise with $\gamma^{(b-t)}$.

For any segment η in d_{0t}, let $R_\ell(\eta)$ be its image in d_{0r}. Let J mark the position of the 1 in '(0^c1)', in d_{0t}, and I mark the position of the corresponding 1 in d_{0r}. The different cases of the proof depend upon the position of $R_\ell(J)$ in d_{or}.

Case A:* $R_\ell(J) = I$. $\ell = 0$.

Suppose $r > t$. Then

$$(I\text{-}8) \Rightarrow \gamma^{(b-t)} = \omega^{(b-r)}\ \xi^{(r-t)} \qquad \ldots \quad (I\text{-}9)$$

for some $\omega^{(b-r)}$ and $\xi^{(r-t)}$:

$$(I\text{-}8) \text{ and } (I\text{-}9) \Rightarrow \xi^{(r-t)}00\ \delta^{(t-1)}\ \dot{\delta}^{(t-1)}\ \xi^{(r-t)} = 00\beta^{(r-1)}\ \dot{\beta}^{(r-1)}. \qquad (I\text{-}10)$$

Now

$$\xi^{(r-t)} = \lambda \Rightarrow r=t,\ \alpha^{(b-r)} = \gamma^{(b-t)},$$

$$\beta^{(r-1)} = \delta^{(t-1)} \Rightarrow d_{0r} = d_{0t} \qquad \ldots \quad (I\text{-}11)$$

---

* The symbol ' $\Rightarrow$' is to be read as 'implies', and ' $\overset{i}{\Rightarrow}$ is to be read as 'implies by i'.

$(\xi^{(r-t)} = \lambda) \overset{I.9}{\Rightarrow} (\gamma^{(b-t)} = \lambda) \overset{\text{Hypothesis}}{\Rightarrow}$

$(\gamma^{(b-t)} = \gamma^{(b-t-1)}\,1) \overset{I.9}{\Rightarrow}$

$(\xi^{(r-t)} = \xi^{(r-t-1)}\,1) \overset{I.10}{\Rightarrow}$

Either $(\xi^{(r-t-1)}\,1 = 0\,(i)\,1\;\xi^{(r-t-i=2)}\,1)$

Or $(\xi^{(r-t-1)}\,1 = 0\,(r-t-1)\,1) \Rightarrow$

NOT (I.10), a contradiction

Case B: $R_\ell(J)$ is in $(\alpha^{(b-r)}\,00\,\beta^{(r-1)}) \Rightarrow$

Since $(b-r+2+r-1) = b+1 < (c+1)$,

$R_\ell\,(0^{(c)}\,J)$ overlaps with I in d_{0r}, a contradiction.

Case C: $R_\ell\,(J)$ is in $\dot{\beta}^{(r-1)} \Rightarrow$

$\dot{\beta}^{(r-1)} = \dot{\beta}^{(n)}\,R_\ell(J)\,\dot{\omega}^{(r-n-2)}$ for

$(n < r-1 < b-1 < c) \Rightarrow$

$\beta^{(r-1)}\,\dot{\beta}^{(r-1)} = \omega^{(r-n-2)} R_\ell(J)\,\beta^{(n)}\,\dot{\beta}^{(n)}\,R_\ell(J)\,\dot{\omega}^{(r-n-2)} \Rightarrow$

$R_\ell(0^{(c)}J)$ is in $\beta^{(n)}\,\dot{\beta}^{(n)}\,R_\ell(J) \Rightarrow$

$c < 2n \Rightarrow$

Since $(n < c)$

$\dot{\beta}^{(r-1)} = (\dot{\beta}^{(n)}\,R_\ell(J)\,\dot{\omega}^{(r-n-2)} = 0^{(n)}\,R_\ell(J)\,\dot{\omega}^{(r-n-2)})$ . . . (I-12)

We shall return to this case shortly.

Case D: (Final Case) $R_\ell(J)$ is in $\dot{\alpha}^{(b-r)} \Rightarrow$

$\dot{\alpha}^{(b-r)} \neq \lambda \overset{\text{Hypothesis}}{\Rightarrow}$

$\dot{\alpha}^{(b-r)} = 1\,\dot{\alpha}^{(b-r-1)}$.

Sine $b-1 < c$ and $R_\ell(J)$ is in $1\dot{\alpha}^{(b-r-1)}$

$$\dot{\beta}^{(r-1)} \; \dot{\alpha}^{(b-\ )} = 0^{(r-1)} \; R_\ell(J) \; \dot{\alpha}^{(b-r-1)} \qquad \ldots \quad (I\text{-}13)$$

By reversing the roles of r and t in cases C and D we have analogous cases C' and D' for R^{-1}_ℓ (I) in $\dot{\delta}^{(t-1)}$ and R^{-1}_ℓ (I) in $\dot{\gamma}^{(b-t)}$, respectively. We shall now consider three composite cases:

(C and C'), (D and D'), and (C and D') (which is the same as (C' and D).

Cases C and C':

$$\text{By (I.12)} \quad \dot{\beta}^{(r-1)} = 0^{(n)} \; R_\ell(J) \; \dot{\omega}^{(r-n-2)} \qquad \ldots \quad (I\text{-}14)$$

and by analogy with (I.12) in case C'

$$\dot{\delta}^{(t-1)} = 0^{(m)} R^{-1}_\ell(I) \; \dot{\eta}^{(t-m-2)} \qquad \ldots \quad (I\text{-}15)$$

Substituting (I-14) in the segment between J and $R^{-1}_\ell(I)$ of d_{0t} in (I.8) we get

$$((J \; \gamma^{(b-t)} 00 \eta^{(t-m-2)} \; 1 \; 0^{(2m)} \; R^{-1}_\ell(I) \;) =$$

$$(R_\ell(J) \; \dot{\omega}^{(r-n-2)} \; \alpha^{(b-r)} \; 0^{(c)} \; I) \;) \Rightarrow$$

$(m+n) = c-3$, contradicting $c \leqslant 2m, 2n$.

Cases D and D'

$$(I.13) \Rightarrow \dot{\alpha}^{(b-r)} = R_\ell(J) \; \dot{\alpha}^{(b-r-1)} \text{ and } \dot{\beta}^{(r-1)} = 0^{(r-1)} \qquad \ldots \quad (I\text{-}16)$$

and by analogy to (I.13) in case D' we have

$$\dot{\gamma}^{(b-t)} = R^{-1}_\ell(I) \; \dot{\gamma}^{(b-t-1)} \text{ and } \dot{\delta}^{(t-1)} = 0^{(t-1)} \qquad \ldots \quad (I\text{-}17)$$

Substituting (I.17) between J and $R^{-1}_\ell(I)$ in d_{0t} of (I.8) we get

$$(J \; \gamma^{(b-t-1)} 1 \; 00 \; 0^{(2(t-1))} R^{-1}_\ell(I) \;) = (R_\ell(J) \; \dot{\alpha}^{(b-r-1)} \; 0^{(c)} \; I \;) \qquad \ldots \quad (I\text{-}18)$$

Substituting (I.16) between $R_\ell(J)$ and I in d_{0r} of (I.8) we get

$$(R_\ell(I)\ \dot{\gamma}^{(b-t-1)}\ 0\ ^{(c)}\ J) =$$

$$(I\ \alpha^{(b-r-1)}\ 1\ 00\ 0^{(r-1)}\ 0^{(r-1)}\ R_\ell(J)) \qquad \ldots \quad (I\text{-}19)$$

$$(I.18) \Rightarrow \gamma^{(b-t-1)}{}_1\ 0^{(2t-c)} = \dot{\alpha}^{(b-r-1)} \qquad \ldots \quad (I\text{-}20)$$

Substituting (I.20) in (I.19) we get

$$(1\ \gamma^{(b-t-1)}\ 0^{(c)}\ 1) = (1\ 0^{(2t-c)}\ 1\ \dot{\gamma}^{(b-t-1)}{}_1$$
$$000^{(2(r-1))}{}_1)$$

which is impossible.

Cases C and D'

Substituting (I.12) and (I.17) in (I.8) as before, we get

$$(J\ \gamma^{(b-t-1)}100\ 0^{(2(t-1))}R^{-1}_\ell(I)) =$$

$$(R_\ell(J)\ \dot{\omega}^{(r-n-2)}\ \dot{\alpha}^{(b-r)}0\ ^{(c)}\ I) \qquad \ldots \quad (I\text{-}21)$$

and

$$(R^{-1}_\ell(I)\ \dot{\gamma}^{(b-t-1)}\ 0^{(c)}\ J) =$$

$$(I\ \alpha^{(b-r)}00\ \omega^{(r-n-2)}{}_1\ 0^{(2n)}\ R_\ell(J)) \qquad \ldots \quad (I\text{-}22)$$

$$(I.22) \Rightarrow \dot{\gamma}^{(b-t-1)} = \alpha^{(b-r)}\ 00\ \omega^{(r-n-2)}{}_1\ 0^{(2n-c)} \qquad \ldots \quad (I\text{-}23)$$

Substituting (I.23) in (I.21) we get

$$0^{(2n-c)}{}_1\ \dot{\omega}^{(r-n-2)}00\ \dot{\alpha}^{(b-r)}{}_1\ 0^{(2t-c)} =$$
$$\omega^{(r-n-2)}\ \alpha^{(b-r)}$$

which again is impossible. Hence Assertion 3.

Lemma 5 and Theorem 1 follow from these three assertions.