AFCRL-66-471

"On Linear Product Codes and Their Duals"

by

Lih-Jyh Weng

Northeastern University
360 Huntington Avenue
Boston, Massachusetts

Contract No. AF19(628)-3312

Project No. 4610

Task No. 461003

Scientific Report No. 4

June 1966

Charles F. Hobbs, CRBK

Distribution of this document is unlimited

This research was supported in part by the National Aeronautics and Space Administration under Grant NGR-22-011-013

Prepared

for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

AFCRL-66-471

"On Linear Product Codes and Their Duals"

by

Lih-Jyh Weng

Northeastern University
360 Huntington Avenue
Boston, Massachusetts

Contract No. AF19(628)-3312

Project No. 4610

Task No. 461003

Scientific Report No. 4

June 1966

Charles F. Hobbs, CRBK

Distribution of this document is unlimited

This research was supported in part by the National Aeronautics and Space Administration under Grant NGR-22-011-013

Prepared

for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

ABSTRACT

In this report the value of studying the tensor product of linear codes is demonstrated. The pertinent problems concerning these product codes are outlined. Algebraic techniques for determining the null space of a tensor product space are developed. The understanding of the null space of a product space is useful not only in the development of this report, but also for future research work. Decomposition of the procedure and implementation of encoding and decoding of a product code into those of its component codes are shown. In the case where the component codes are cyclic, the product code has the special feature that its encoder and syndrome calculator can be easily converted to those of its dual codes by programmed switching. A simple decoding scheme; namely, permutation decoding, which is capable of correcting a large fraction of all the correctable errors of a systematic cyclic code, is investigated. It is suggested that it be used either as a part of the correction-detection scheme or in combination with an auxiliary scheme to attain full error correction capability. Finally, the minimum distances of product codes, and suitable communication channels for employing such codes, are discussed.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Hamming[7], Golay[8] and Slepian[9,10] are the founders of linear algebraic coding theory. Ever since the idea was brought up, the developments and advancement of this area have progressed steadily. Among linear codes, the cyclic code suggested by Prange[11] and a wide class discovered by Bose and Ray-Chaudhuri[12,13] and independently by Hocquenghem[14] are the most important ones in the sense that the implementation is relatively simple. Peterson[2] gives a unified treatment of algebraic coding theory. His excellent text book benefits many newcomers who are interested in this area. The recent research work in linear algebraic coding theory can be roughly divided into the following five areas:

(i) To discover new codes which are particularly suitable for a variety of communication channels.

(ii) To form new and simple decoding procedures and algorithms for the existing codes. For example, new decoding procedures and algorithms for the B-C-H codes have been discovered recently by several authors[15,16,17,18].

(iii) To analyze the existing codes so that their properties such as algebraic structure, error control capability, comma-freedom and weight distribution, etc. can be understood more thoroughly.

(iv) To generate new recursive sequences which are applicable to synchronization, ranging and orbit determination purposes.

(v) To combine existing codes according to certain algebraic rules to form a new compound code. This will enlarge our vocabulary of codes to suit special error-control capabilities required by the user. Furthermore, it will perform some special functions, such as error-locating, which are otherwise impossible by means of a simple component code alone.

Actually, there are no clear-cut ways to divide the above five areas. To solve a problem in any one area usually requires the knowledge of other areas. Sometimes one may even have to solve the problems in other areas first before attacking the problem itself. Nevertheless, the above division is adequate for the sake of convenience.

In order to fully understand and perform further research work in the fifth area which is the main aim of this report, it is necessary to have extensive knowledge of the other four areas. On the other hand, a thorough understanding of the fifth area is helpful in research work in all the other areas. The following shows some interrelations between the fifth area and the other ones.

(i) and (v) The more new codes we know, the more combined codes we will have. Conversely, the combined codes stimulate research work in the first area with the purpose of inventing new competitive codes, or designing component codes particularly suited for combination.

(ii) and (v) So far the decoding schemes for most of the existing codes are not immediately feasible. It is more promising to implement long and complex codes through a combination of short and simple codes, because the decoding procedures for a combined code can be decomposed into those of its omponent codes.

(iii) and (v) The relation between these two areas is evident. The more we know about the properties of the component codes, the more we understand the combined code. The study of such properties presents many new challenging problems. Furthermore, a thorough understanding of the properties of a combined code will help us understand a class of equivalent codes.* For example, the properties such as weight distribution, minimum distance, etc. are identical for two equivalent codes.

*The definition of an equivalent code is given in Chapter 2.

(iv) and (v) Presently, the maximum-length linear shift register sequences and their combined versions are studied for use in ranging and orbit determination. Also the Barker sequences[29] are used for synchronization of code words. It is believed that the knowledge of combining codes may be applicable to the combination of these sequences to generate new sequences which are more suitable for these purposes.

The tensor product is the algebraic rule adopted in this report for combining linear codes. The tensor product code is the linear algebraic code

(i) whose generator matrix is the tensor product of the generator matrices of two known codes;

or (ii) whose parity-check matrix is the tensor product of the parity-check matrices of two given codes.

In this chapter, the background of product codes will be briefly reviewed. Next, the algebra of the tensor product of matrices is illustrated. Then the definition and some properties of tensor product codes are given. Finally, the salient problems in this area are stated and the contents of the following chapters are introduced.

1.1 <u>Historical Background</u>

Elias[6] first studied the iterated codes. Calabi, Haefeli[20], Hobbs[21] and Kautz[22] extended the work of iterated codes. Slepian[1] introduced the tensor product of generator matrices to represent the iterated codes.

Wolf and Elspas[23] suggested the idea of error-locating codes. Wolf[24,25] and Chang and Weng[26] independently discovered that the error-locating codes can be represented by a parity-check matrix which is a tensor product of two parity-check matrices. Burton and Weldon[27] found a way to construct a cyclic code by permuting the code words of a product code with cyclic component codes of relative prime lengths. Chang and Weng[28] initiated the study of dual product code which can be used as variable redundancy codes with high flexibility.

1.2 Algebra of Tensor Product of Matrices

The tensor product of matrices will be used extensively in this report. It is desirable to give some fundamental rules for operation with tensor products.

Let the matrices A and B be given as follows

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{k1} & a_{k2} & \cdots & a_{kn} \end{bmatrix} \tag{1.1}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ b_{h1} & b_{h2} & \cdots & b_{hm} \end{bmatrix}, \tag{1.2}$$

where a_{ij} and b_{ij} are assumed, for the purpose of this report, as elements of GF(q), q being power of a prime. Then the tensor product of A and B denoted by $A \otimes B$ is given as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & & & \\ a_{k1}B & a_{k2}B & \cdots & a_{kn}B \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} & \cdots & a_{11}b_{1m} & \cdots & a_{1n}b_{11} & \cdots & a_{1n}b_{1m} \\ & \vdots & & \vdots & & \vdots & \\ a_{11}b_{h1} & \cdots & a_{11}b_{hm} & \cdots & a_{1n}b_{h1} & \cdots & a_{1n}b_{hm} \\ & \vdots & & \vdots & & \vdots & \\ a_{k1}b_{11} & \cdots & a_{k1}b_{1m} & \cdots & a_{kn}b_{11} & \cdots & a_{kn}b_{1m} \\ & \vdots & & \vdots & & \vdots & \\ a_{k1}b_{h1} & \cdots & a_{k1}b_{hm} & \cdots & a_{kn}b_{h1} & \cdots & a_{kn}b_{hm} \end{bmatrix} . \quad (1.3)$$

Note that A is a $k \times n$ matrix and B an $h \times m$ matrix, and $A \otimes B$ becomes a $kh \times nm$ matrix.

Some formal rules for operating with tensor products are listed below for future use without proof.

$$A \otimes 0 = 0 \otimes B = 0. \quad (1.4)$$

$$I_k \otimes I_h = I_{kh} , \quad (1.5)$$

where I_k is a $k \times k$ identity matrix.

$$(A_1 + A_2) \otimes B = (A_1 \otimes B) + (A_2 \otimes B). \quad (1.6)$$

$$A \otimes (B_1 + B_2) = (A \otimes B_1) + (A \otimes B_2). \quad (1.7)$$

$$aA \otimes bB = ab(A \otimes B). \quad (1.8)$$

$$(A \otimes B)^T = A^T \otimes B^T ,$$

where T denotes transpose.

$$(A_1A_2) \otimes (B_1B_2) = (A_1 \otimes B_1)\ (A_2 \otimes B_2). \quad (1.10)$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} , \quad (1.11)$$

provided A and B are nonsingular square matrices. The proofs of these relations can be found in the literature.[19]

1.3 Definition and Properties of Tensor Product Codes

The generator matrix of a linear code is the matrix whose row vectors are the basis of the code space. The parity-check matrix of a linear code is the matrix whose row space is the null space of the code space. There are two basic kinds of tensor product codes derivable from two given component codes. The first one is the code whose generator matrix is the tensor product of the generator matrices of its component codes. The other code is defined by its parity-check matrix which is a tensor product of the parity-check matrices of its component codes. The dual code of a linear code is a code whose generator matrix is equal to the parity-check matrix of the known code, or equivalently, whose parity-check matrix is equal to the generator matrix of the known code. From two component codes and their duals, eight product codes may be developed.

Therefore, it is an effective way of forming new codes from given codes. In the case of cyclic codes, another definition of dual codes in terms of the generator polynomials will be given in Chapter 4.

Due to the fact that either the generator matrix or the parity-check matrix of a product code can be expressed as a tensor product of two matrices, the entire code word block can be divided into subblocks each of which has a length equal to that of the second component code. With the subblock structure, the product codes can be understood somewhat easier. Each subblock is controlled by the second component code while the whole block is controlled by the first component code if each subblock is interpreted as a digit of the first component code. A useful concept of error control, the error-locating property, is a direct consequence of this subblock structure. If s or fewer erroneous subblocks, each having no more than t errors in any received code word, can be located, the code is said to have the error-locating property (s,t).

For the case of product codes with cyclic component codes, it is relatively simple to implement an encoder at the transmitting terminal and a syndrome calculator at the receiving end with eight operating modes possessing various capabilities in error control. All the operating modes have an identical code word length which is equal to the product of the lengths of the two component codes. The amounts of redundancy are, of course, all different for the eight modes. If the code word length is allowed to vary, we can employ the same encoder and syndrome calculator to operate 12 modes which include 8 product codes, 2 original component codes and the duals of the component codes. This is a highly flexible system and is particularly suitable for time-varying channels.

Another interesting property of the product code is that it is possible to correct both random errors and burst errors.[27]

1.4 Problems to be Solved

Following is a list of pertinent problems concerning product codes.

(i) To find the null space of a tensor product space: this is desirable from the viewpoint of obtaining a thorough mathematical understanding of tensor product spaces, hence product codes. The result should help one to visualize better the structure of product codes.

(ii) Logic implementation: this is a practical problem. It is desirable that the implementation of a product code can be achieved by simple logic combinations of the implementations of its component codes.

(iii) Study of product codes with cyclic component codes: since the cyclic codes are the easiest ones to be implemented, special attention should be given to the product codes with cyclic components.

(iv) Calculation of probabilities of error in data transmission for tensor product codes under various rates of transmission.

(v) Calculation of the weight distribution of product codes and determination of its relation with those of the component codes.

(vi) Study of the code or codes which can be iterated in a product manner indefinitely with a non-zero data transmission rate. The iteration may include the use of convolutional codes as components.

Chapter 2 is devoted to the null space of a tensor product space (Item i). The possibility of decomposition of encoding and decoding processes of a product code into those of its component codes is shown in Chapter 3 (Item ii). In Chapter 4, we discuss product codes with cyclic component codes and emphasize the dual product codes as variable redundancy codes (Item iii). Some decoding schemes are studied in Chapter 5. In the last chapter, miscellaneous results and concluding remarks are included. Items (iv), (v) and (vi) are reserved for future studies.

CHAPTER 2

THE NULL SPACE OF A TENSOR PRODUCT SPACE

A linear code can be completely specified by either its generator matrix or its parity-check matrix or both. A product code is determined by two original code spaces and the method of combining them. For example, one of the product codes is a code whose generator matrix is the tensor product of the generator matrices of two given code spaces. In order to have a better understanding of a product code, it is desirable to know both its generator matrix and its parity-check matrix. The knowledge of both facilitates the determination of the weight distribution of the code and the implementation of encoder and decoder. Furthermore, it provides better insight into the code structure.

The row vectors of the generator and check matrices of a code are the null spaces of each other. In the case of a product code, either its generator matrix or check matrix is expressed as a tensor product of two component matrices. Therefore, it is desirable to develop some techniques of finding the null space of a tensor product space. Slepian[1] found a simple way to evaluate the null space of an equivalent code space of the tensor product code space instead of that of the tensor product code space itself; he also concluded that there seems to be no simple expression for the parity-check matrix for a product code of two component codes in terms of the parity-check matrices of the component codes.

The direct method of Section 2.1 is an extension of the technique suggested by Slepian.[1] Section 2.2, the algebraic approach, makes use the algebraic structure of the tensor product space to express its null space in terms of the null spaces of the component code spaces. For completeness the determination of the null space of a tensor product space involving a special type of translation of fields is discussed in Section 2.3. Such translation is desirable in designing more efficient error-locating codes.

Since the generator matrix G and the parity-check matrix H of a code space will be used very frequently in the following sections, and because of the dual property of these two matrices, we introduce a gp (stands for generator-parity-check) matrix D and a pg (stands for parity-check-generator) matrix E. The gp matrix D can be either a generator matrix G or a parity-check matrix H of a code space; the row space of the pg matrix E gives the null space of the gp matrix D. In other words, if we set D = G, then E = H; and if we set D = H, then E = G. The relation between D and E can be described mathematically by:

$$DE^T = 0$$

or equivalently $$ED^T = 0. \tag{2.1}$$

If the given gp matrix D of a code can be transformed to the reduced-echelon form D' by elementary row operations, then the following theorems (after Peterson[2]) give a simple way of finding the pg matrix E of the code:

Theorem 2.1

If one matrix is obtained from another by a succession of elementary row operations, both matrices have the same row space.

Theorem 2.2

If U is the row space of the gp matrix $D' = [I_h\ P]$, where I_h is an $h \times h$ identity matrix and P is an $h \times (n-h)$ matrix, then U is the null space of $Q = [-P^T\ I_{n-h}]$, where I_{n-h} is an $(n-h) \times (n-h)$ identity matrix.

With the help of these theorems the null space of any gp matrix D, which can be transformed to the reduced-echelon form D' by elementary row operations, can be easily found. Unfortunately, in general, the gp matrix of a product code cannot be transformed to a reduced-echelon form without resorting to the permutation of columns. Although the permutation of columns of a gp matrix does not change the minimum distance or the weight distribution of the code, it will, however, destroy other code properties such as being cyclic, orthogonal, and comma free.

The following sections give some techniques to determine the null space of the gp matrix D of a product code without disturbing its original ordering of columns. Since the null space is specified by the pg matrix E corresponding to the known gp matrix, D, all the techniques in this chapter show how to evaluate E from D. The same techniques may be used in evaluating D from E.

2.1 The Direct Method

Two linear (n,k) codes are said to be equivalent if each code word of one code can be obtained by a certain permutation of code symbols of one and only one code word of the other code. This equivalence relation can be defined alternatively as either

(i) two linear (n,k) codes are equivalent if and only if the generator matrix of one code can be obtained by permuting the columns of a generator matrix of the other code;

or (ii) two linear (n,k) code are equivalent if and only if the parity-check matrix of one code can be obtained by permuting the columns of a parity-check matrix of the other code.

The purpose of investigating the equivalent codes in this section is to find the pg matrix E of a product code when its gp matrix D is known through using one of the equivalent codes as an intermediate step.

Before proceeding any further, we first define a permutation matrix π. The π matrix is an $n \times n$ square matrix whose columns (or rows) are obtained from a certain permutation of the columns (or rows) of an $n \times n$ unity matrix I. It is evident that when a $k \times n$ matrix A is postmulitplied by π, the resultant matrix $A' = A\pi$ can be recognized as a matrix whose columns is a certain permutation of that of the matrix A. The pattern of permuting the columns of the matrix A to obtain A' is identical to that of permuting the unity matrix I to form π.

<u>Lemma 2.1</u>

A permutation matrix π is an orthogonal matrix, i.e., $\pi^T = \pi^{-1}$.

Proof

The permutation matrix π is obtained by permuting the rows of the unity matrix I; hence each row of the matrix π can be considered as an n-tuple of the form (0,0,...,0,1,0,...,0), where only one component is one and the rest are zeros. We write π as

$$\pi = \begin{bmatrix} \underline{a_1} \\ \underline{a_2} \\ \cdot \\ \cdot \\ \cdot \\ \underline{a_n} \end{bmatrix}^{*}, \qquad (2.2)$$

where

$$\underline{a_i}\left(\underline{a_j}\right)^T = \underline{a_i}\, a_j] = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \,. \end{cases}$$

Since $\underline{a_i}$'s are row vectors of the unity matrix I and the row vectors of a unity matrix are orthogonal, then

$$\pi\,\pi^T = \begin{bmatrix} \underline{a_1} \\ \underline{a_2} \\ \cdot \\ \cdot \\ \cdot \\ \underline{a_n} \end{bmatrix} \Big[a_1]\ a_2]\ \ldots\ a_n] \Big]$$

$$= [\delta_{ij}]$$

$$= I, \qquad (2.3)$$

* $\underline{\qquad}$ denotes a row vector.

which implies $$\pi^T = \pi^{-1} .$$ Q.E.D.

Theorem 2.3

Let D and E be the gp and pg matrices respectively of code C, then $D\pi$ is a gp matrix of the code C', which is equivalent to C, if and only if $E\pi$ is a pg matrix of the code C'.

Proof

$$\begin{aligned} & (D\pi)(E\pi)^T \\ &= D\pi\,\pi^{-1}\,E^T && \text{by Lemma 2.1} \\ &= DI\,E^T \\ &= DE^T \\ &= 0 , && \text{by hypothesis.} \end{aligned}$$

Also, $\text{Rank}(D\pi) = \text{Rank}(D)$

$\text{Rank}(E\pi) = \text{Rank}(E)$.

Since the rank of a matrix is preserved under permutation of the columns of the matrix, therefore, if $D\pi$ is the gp matrix of the equivalent code C', then $E\pi$ must be the corresponding pg matrix. Q.E.D.

Theorem 2.4

Let D be a gp matrix of a code C. Assume $D\pi$ is the gp matrix of an equivalent code C'. If E' is the pg matrix of C', then $E = E'\pi^{-1} = E'\pi^T$ is the pg matrix of C.

Proof

$$(D\pi)(E')^T = 0 \qquad \text{by hypothesis} \qquad (2.4)$$

$$\begin{aligned} DE^T &= D(E'\pi^T)^T \\ &= D(\pi^T)^T(E')^T \\ &= (D\pi)(E')^T = 0 \qquad \text{by Equation } (2.4) \end{aligned}$$

And $\text{Rank}(D) = \text{Rank}(D\pi)$

$\text{Rank}(E) = \text{Rank}(E'\pi^T) = \text{Rank}(E')$ Q.E.D.

Theorems 2.1, 2.2, 2.3 and 2.4 suggest a way to evaluate the pg matrix E when the gp matrix D of a code is given. The procedures are as follows:

1. Let D be the $h \times n$ gp matrix of a linear (n,k) code C, then by proper elementary row operations D can be transformed to an echelon form D' which contains h columns each of which consists of one unit component and the rest of the components are zeros. This step can always be accomplished because an $h \times n$ gp matrix D consists of at least one set of h independent columns if $h < n$, and rank $D = h$. It can be easily shown that any one of these sets of h independent columns can be transformed to h columns, each equal to one and only one column of the unity $h \times h$ matrix I_h.
2. Choose a suitable permutation matrix π such that $D'' = D'\pi = [I_h, P]$, the reduced-echelon form of D.

3. Form $E'' = [-P^T, I_{n-h}]$. By Theorem 2.2, we know that $D''(E'')^T = 0$.

4. Let $E = E''\pi^T$. By Theorem 2.4, E is the pg matrix of the code C corresponding to the gp matrix D'; but D' is obtained from D by elementary row operations, hence D and D' generates the same space. Therefore, E is the pg matrix of the code C corresponding to the gp matrix D.

<u>Example</u>

$$D = \begin{bmatrix} 1&0&1&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0&0 \\ 1&0&1&1&0&1&0&0&0 \\ 0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&1&1 \end{bmatrix} . \qquad (2.5)$$

1. The first row is added to the third row

$$D' = \begin{bmatrix} 1&0&1&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0&0 \\ 0&0&0&1&0&1&0&0&0 \\ 0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&1&1 \end{bmatrix} . \qquad (2.6)$$

2. The first, second, fourth, fifth, seventh, and the eighth eolumns should be gathered together to become the first sixth columns, hence we choose

$$\pi = \begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \end{bmatrix} . \qquad (2.7)$$

The last three columns of Equation (2.7) can be interchanged among themselves.

$$D'' = D'\pi = \begin{bmatrix} 1&0&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&1&0&0&1 \end{bmatrix} . \tag{2.8}$$

3. From Equation (2.8), we have

$$P = \begin{bmatrix} 1&0&0 \\ 1&0&0 \\ 0&1&0 \\ 0&1&0 \\ 0&0&1 \\ 0&0&1 \end{bmatrix} \tag{2.9}$$

$$-P^T = \begin{bmatrix} 1&1&0&0&0&0 \\ 0&0&1&1&0&0 \\ 0&0&0&0&1&1 \end{bmatrix} , \tag{2.10}$$

then

$$E'' = \left[-P^T, I_{n-h} \right]$$

$$= \begin{bmatrix} 1&1&0&0&0&0&1&0&0 \\ 0&0&1&1&0&0&0&1&0 \\ 0&0&0&0&1&1&0&0&1 \end{bmatrix} . \tag{2.11}$$

4.

$$E = E''\pi^T = \begin{bmatrix} 1&1&1&0&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&1&1&1 \end{bmatrix} . \tag{2.12}$$

The method described in this section is straightforward but tedious. Since it does not refer to the component codes or the manner of their combination, the resulting pg matrix E found in this manner usually is not in a form which gives the best insight of the code structure. Therefore, it is necessary to resort to some other techniques which will lead to a more meaningful result.

2.2 The Algebraic Approach - Elements of Two Original Component Codes and Elements of the Resultant Code Expressed in the Same Field.

The algebraic approach relies on the algebra of the tensor product of matrices as well as the understanding of vector spaces. The following lemmas and theorems are first developed to facilitate the discussion.

Lemma 2.2

Let A and B be two square matrices of order r and s respectively, then

$$|A \otimes B| = |A|^s \, |B|^r \, .$$

Proof

It is known that every square matrix can be transformed into a triangular matrix* by a non-singular

*By triangular matrix, we mean the matrix of the form either

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ 0 & 0 & \ddots & \\ \vdots & \vdots & & \ddots \\ 0 & 0 & 0 & a_{nn} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} b_{11} & 0 & 0 & \cdots & 0 \\ b_{21} & b_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \\ \vdots & \vdots & & \ddots & \\ b_{n1} & b_{n2} & & & b_{nn} \end{bmatrix} .$$

square matrix in the following way:

$$Q_1^{-1} A Q_1 = T_A \tag{2.13}$$

$$Q_2^{-1} B Q_2 = T_B , \tag{2.14}$$

where T_A and T_B are triangular matrices and $|Q_1| \neq 0$, $|Q_2| \neq 0$. Then

$$\begin{aligned} |A \otimes B| &= |Q_1(Q_1^{-1} A Q_1) Q_1^{-1} \otimes Q_2(Q_2^{-1} B Q_2) Q_2^{-1}| \\ &= |Q_1 T_A Q_1^{-1} \otimes Q_2 T_B Q_2^{-1}| \\ &= |Q_1 \otimes Q_2| |T_A \otimes T_B| |Q_1^{-1} \otimes Q_2^{-1}| \\ &= |T_A \otimes T_B| . \end{aligned} \tag{2.15}$$

But

$$|T_A| = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ 0 & a_{22} & \cdots & a_{2r} \\ 0 & 0 & \ddots & \\ 0 & 0 & & a_{rr} \end{vmatrix} = \prod_{i=1}^{r} a_{ii} \tag{2.16}$$

$$|T_B| = \begin{vmatrix} b_{11} & b_{12} & \cdots & b_{1s} \\ 0 & b_{22} & \cdots & b_{2s} \\ 0 & 0 & \ddots & \\ 0 & 0 & & b_{ss} \end{vmatrix} = \prod_{i=1}^{s} b_{ii} . \tag{2.17}$$

By Equations (2.15), (2.16) and (2.17), we have

$$|A \otimes B| = |T_A \otimes T_B|$$

$$= \begin{vmatrix} a_{11}T_B & a_{12}T_B & \cdots & a_{1r}T_B \\ 0 & a_{22}T_B & \cdots & a_{2r}T_B \\ 0 & 0 & \ddots & \\ 0 & \cdots & 0 & a_{rr}T_B \end{vmatrix}$$

$$= \begin{vmatrix} a_{11}b_{11} & & & & & & & & & & & \\ & a_{11}b_{22} & & & & & & & \bigcirc & & & \\ & & \ddots & & & & & & & & & \\ & & & a_{11}b_{ss} & & & & & & & & \\ & & & & a_{22}b_{11} & & & & & & & \\ & & & & & a_{22}b_{22} & & & & & & \\ & & & & & & \ddots & & & & & \\ & & \bigcirc & & & & & a_{22}b_{ss} & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & a_{rr}b_{11} & & \\ & & & & & & & & & & \ddots & \\ & & & & & & & & & & & a_{rr}b_{ss} \end{vmatrix}$$

$$= \left(\prod_{i=1}^{r} a_{ii}\right)^{s} \left(\prod_{i=1}^{s} b_{ii}\right)^{r} = |T_A|^{s}\,|T_B|^{r}$$

$$= |A|^{s}\,|B|^{r} \; .$$

Q.E.D.

<u>Lemma 2.3</u>

Let $D = P \otimes Q$ and $D' = P' \otimes Q'$. If P' and Q' are obtained from P and Q respectively by a succession of elementary row operations, then D and D' have the same row space.

<u>Proof</u>

This is a direct consequence of Theorem 2.1. The row space of $D = P \otimes Q$ contains all the elements of the form $\underline{d} = \sum_{i=1}^{r} \underline{p_i} \otimes \underline{q_i}$, where $\underline{p_i} \in$ row space of P = row space of P', $\underline{q_i} \in$ row space of Q = row space of Q'. Hence, $\underline{d} \in$ row space of D', or row space of $D \subseteq$ row space of D'. By symmetry, we have row space of $D \supseteq$ row space of D'. Therefore, row space of D = row space of D'. Q.E.D.

From the above lemma, we can see that the rank of the matrix $D = P \otimes Q$ is invariant if elementary row operations are performed on the matrices P and Q.

<u>Lemma 2.4</u>

Let $D = D_1 \otimes D_2$, then

$$\text{Rank } D = (\text{Rank } D_1)(\text{Rank } D_2).$$

Proof

Let A and B be the square matrices of maximum order contained in D_1 and D_2 respectively, such that

$$|A| \neq 0, \quad |B| \neq 0.$$

Then it is clear that

order of A = rank D_1 and let it be denoted by r_1

order of B = rank D_2 and let it be denoted by r_2.

Then the square matrix $A \otimes B$ must be contained in $D = D_1 \otimes D_2$. But, by Lemma 2.2,

$$|A \otimes B| = |A|^{r_2} |B|^{r_1} \neq 0.$$

Hence, Rank D $\geq$ order of $A \otimes B$ = (order of A)(order of B)

= (Rank D_1)(Rank D_2).

By Lemma 2.3, the rank D is unchanged, if the elementary row operations are performed on D_1 and D_2. But D_1 can be reduced to r_1 non-zero rows since its rank is r_1; similarly, D_2 can be reduced to r_2 non-zero rows. Hence, after performing suitable elementary row operations on D_1 and D_2, $D = D_1 \otimes D_2$ contains exactly $r_1 r_2$ non-zero rows. Therefore,

$$\text{Rank } D \leq r_1 r_2 = (\text{Rank } D_1)(\text{Rank } D_2).$$

Combining with the above result, we have

$$\text{Rank } D = (\text{Rank } D_1)(\text{Rank } D_2). \qquad \text{Q.E.D.}$$

Theorem 2.5

Let D_1 and E_1 be the gp and pg matrices respectively of code C_1, and D_2 and E_2 be gp and pg matrices respectively of code C_2. If D_1 is an $h_1 \times n_1$ matrix and D_2 is an $h_2 \times n_2$ matrix, then both row spaces of $I_{n_1} \otimes E_2$ and $E_1 \otimes I_{n_2}$ are contained in the null space of the matrix D, where $D = D_1 \otimes D_2$.

Proof

$$
\begin{aligned}
& D(I_{n_1} \otimes E_2)^T \\
&= (D_1 \otimes D_2)(I_{n_1}{}^T \otimes E_2)^T \\
&= D_1 I_{n_1} \otimes D_2 E_2{}^T \\
&= D_1 \otimes 0, \quad \text{since} \quad D_2 E_2{}^T = 0 \\
&= 0
\end{aligned}
$$

Likewise, $D(E_1 \otimes I_{n_2})^T = 0$. Q.E.D.

By elementary row operations, the identity matrix I_{n_1} can be transformed into the form $\begin{bmatrix} \bar{E}_1 \\ E_1 \end{bmatrix}$ in such a way* that

(i) row space I_{n_1} = row space $\begin{bmatrix} \bar{E}_1 \\ E_1 \end{bmatrix}$, this is always true since $\begin{bmatrix} \bar{E}_1 \\ E_1 \end{bmatrix}$ can be obtained from I_{n_1} by elementary row operations;

*$\bar{E}$ may be called a complementary space of E with respect to I_{n_1}

(ii) (row space of $\bar{E}_1$) $\cap$ (row space of E_1) = $(0,0,\ldots,0)$, the all zero n_1-tuple. (2.18)

The above two conditions can be described mathematically as row space of I_{n_1} = (row space of $\bar{E}_1$) $\oplus$ (row space of E_1), where $\oplus$ is the notation for the direct sum.

Similarly, the identity matrix I_{n_2} can be transformed into the form $\begin{bmatrix} \bar{E}_2 \\ E_2 \end{bmatrix}$ by elementary row operation such that row space of I_{n_2} = (row space of E_2) $\oplus$ (row space of E_2). (2.19)

Corollary 2.5.1

The row spaces of $\bar{E}_1 \otimes E_2$, $E_1 \otimes E_2$, and $E_1 \otimes \bar{E}_2$ are subspaces of the null space of the matrix $D = D_1 \otimes D_2$. The proof is similar to that of Theorem 2.5, hence is omitted here.

Lemma 2.5

Let A_1, A_2, B_1 and B_2 be $h \times n$, $k \times n$, $\ell \times m$ and $r \times m$ matrices respectively, then

$$\text{Rank} \begin{bmatrix} A_1 \otimes B_1 \\ A_2 \otimes B_2 \end{bmatrix} = (\text{Rank } A_1)(\text{Rank } B_1) + (\text{Rank } A_2)(\text{Rank } B_2),$$

provided one of the following three conditions is satisfied:

(i) $\text{Rank} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \text{Rank } A_1 + \text{Rank } A_2$,

(ii) $\text{Rank} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \text{Rank } B_1 + \text{Rank } B_2$,

(iii) both (i) and (ii) are satisfied.

Proof

In the following, it is assumed that the condition (i) is satisfied. The proof for the case that condition (ii) is satisfied is similar and the proof for the condition (iii) is evident. Hence, the proofs for conditions (ii) and (iii) will be omitted.

By Lemma 2.4, we have

$$\operatorname{Rank}(A_1 \otimes B_1) = (\operatorname{Rank} A_1)(\operatorname{Rank} B_1) \tag{2.20}$$

$$\operatorname{Rank}(A_2 \otimes B_2) = (\operatorname{Rank} A_2)(\operatorname{Rank} B_2). \tag{2.21}$$

To complete our proof, we need only to show that none of the non-zero row vectors of $A_1 \otimes B_1$ can be expressed as a linear combination of row vectors of $A_2 \otimes B_2$ and vice versa. Let

$$A_1 \otimes B_1 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{h1} & a_{h2} & \cdots & a_{hn} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ b_{\ell 1} & b_{\ell 2} & \cdots & b_{\ell m} \end{bmatrix}$$

and let $\underline{a_i} \otimes \underline{b_j}$ be a row of $A_1 \otimes B_1$:

$$\underline{a_i} \otimes \underline{b_j} = (a_{i1}b_{j1},\ a_{i1}b_{j2},\ \ldots,\ a_{i1}b_{jm}, a_{i2}b_{j1}, a_{i2}b_{j2}, \ldots$$
$$\ldots,\ a_{i2}b_{jm},\ \ldots,\ a_{in}b_{j1}, a_{in}b_{j2},\ \ldots,\ a_{in}b_{jm}).$$

Define a linear mapping

M_δ: row space of $I_n \otimes I_m \longrightarrow$ row space of I_n such that only δth, $(m+\delta)$th, $(2m+\delta)$th ... $[(n-1)m+\gamma]$th components are retained after mapping.

Therefore,

$$M_\delta[\underbracket{a_i} \otimes \underbracket{b_j}] = (a_{i1}b_{j\delta}, a_{i2}b_{j\delta}, \ldots, a_{in}b_{j\delta})$$

$$= b_{j\delta}(a_{i1}, a_{i2}, \ldots, a_{in})$$

$$= b_{j\delta}\, \underbracket{a_i}\,.$$

M_δ can be easily shown to be a homomorphism. Let

$$A_2 \otimes B_2 = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \alpha_{k1} & \alpha_{k2} & \cdots & \alpha_{kn} \end{bmatrix} \otimes \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \beta_{r1} & \beta_{r2} & \cdots & \beta_{rm} \end{bmatrix}$$

and suppose

$$\underbracket{a_i} \otimes \underbracket{b_j} = C_{11}(\underbracket{\alpha_1} \otimes \underbracket{\beta_1}) + C_{12}(\underbracket{\alpha_1} \otimes \underbracket{\beta_2}) + \cdots$$

$$\cdots + C_{1r}(\underbracket{\alpha_1} \otimes \underbracket{\beta_r}) + \cdots + C_{k1}(\underbracket{\alpha_k} \otimes \underbracket{\beta_1}) + \cdots$$

$$+ C_{kr}(\underbracket{\alpha_k} \otimes \underbracket{\beta_r}) ,$$

where $C_{11}, C_{12}, \ldots, C_{kr}$ are constants and at least one of them is not zero, and $(\underbracket{\alpha_1} \otimes \underbracket{\beta_1})$, ..., $(\underbracket{\alpha_k} \otimes \underbracket{\beta_r})$ are rows of $A_2 \otimes B_2$. Then

$$M_\delta[\underline{a_i}\otimes\underline{b_j}] = M_\delta[c_{11}(\underline{\alpha_1}\otimes\underline{\beta_1}) + c_{12}(\underline{\alpha_1}\otimes\underline{\beta_2}) + \cdots + c_{kr}(\underline{\alpha_k}\otimes\underline{\beta_r})],$$

or

$$b_{j\delta}\,\underline{a_i} = c_{11}\beta_{1\delta}\,\underline{\alpha_1} + c_{12}\beta_{2\delta}\,\underline{\alpha_1} + \cdots + c_{kr}\beta_{r\delta}\,\underline{\alpha_k} \tag{2.22}$$

for $\delta = 1,2,3,\ldots,m$, and $j = 1,2,3,\ldots,\ell$, since B_1 is not a matrix with all zero elements. There exists for each $\underline{a_i}$, at least one non-zero element $b_{j\delta}$ such that Equation (2.22) is satisfied. In other words, $\underline{a_i}$ can be obtained from a linear combination of $\underline{\alpha_1}$, $\underline{\alpha_2}$,..., $\underline{\alpha_k}$. This is contradictory to our assumption that

$$\text{Rank}\begin{bmatrix}A_1\\A_2\end{bmatrix} = \text{Rank } A_1 + \text{Rank } A_2$$

which implies that no rows (containing at least one non-zero element) of matrix A_1 can be obtained from a linear combination of the rows of matrix A_2.

Through a similar procedure, we can show that no rows of $A_2 \otimes B_2$ can be obtained from $A_1 \otimes B_1$ by elementary row operations. Therefore, we have

$$\text{Rank}\begin{bmatrix}A_1\otimes B_1\\A_2\otimes B_2\end{bmatrix} = \text{Rank}(A_1\otimes B_1) + \text{Rank}(A_2\otimes B_2). \tag{2.23}$$

The proof is completed by combining Equations (2.20), (2.21), and (2.23).

<u>Theorem 2.6</u>

Let D_1 and E_1 be the gp and pg matrices respectively of code C_1 with block length n_1, and let D_2 and E_2 be the gp and pg matrices respectively of code C_2 with block length n_2. If the gp matrix of the product code is given by $D = D_1 \otimes D_2$, then the pg matrix of the product code is

$$E = \begin{bmatrix} E_1 \otimes \overline{E}_2 \\ E_1 \otimes E_2 \\ \overline{E}_1 \otimes E_2 \end{bmatrix},$$

where row space of I_{n_1} = (row space of E_1) $\oplus$ (row space of $\overline{E}_1$), and row space I_{n_2} = (row space of E_2) $\oplus$ (row space of $\overline{E}_2$).

<u>Proof</u>

By Corollary 2.5.1, we know that the row spaces of $E_1 \otimes \overline{E}_2$, $E_1 \otimes E_2$, and $\overline{E}_1 \otimes E_2$ are all contained in the null space of $D = D_1 \otimes D_2$. Hence,

$$\text{row space of} \begin{bmatrix} E_1 \otimes \overline{E}_2 \\ E_1 \otimes E_2 \\ \overline{E}_1 \otimes E_2 \end{bmatrix} \subseteq \text{row space of } E.$$

The dimension of the row space of D = Rank D = (Rank D_1)(Rank D_2).

The dimension of row space of E = Rank E

$$= n_1n_2 - \text{Rank } D$$
$$= n_1n_2 - (\text{Rank } D_1)(\text{Rank } D_2).$$

Since the block length of the product code = n_1n_2, all we need to show now is that the dimension of

$$\text{the row space of} \begin{vmatrix} E_1 \otimes \overline{E}_2 \\ E_1 \otimes E_2 \\ \overline{E}_1 \otimes E_2 \end{vmatrix} = \text{Rank} \begin{vmatrix} E_1 \otimes \overline{E}_2 \\ E_1 \otimes E_2 \\ \overline{E}_1 \otimes E_2 \end{vmatrix} = \text{Rank } E.$$

This is true, since

$$\text{Rank}\begin{bmatrix} E_1 \otimes \bar{E}_2 \\ E_1 \otimes E_2 \\ \bar{E}_1 \otimes E_2 \end{bmatrix} = \text{Rank}\begin{bmatrix} E_1 \otimes \begin{bmatrix} \bar{E}_2 \\ E_2 \end{bmatrix} \\ \bar{E}_1 \otimes \ E_2 \end{bmatrix}$$

$$= \text{Rank}\begin{bmatrix} E_1 \otimes I_{n_2} \\ \bar{E}_1 \otimes E_2 \end{bmatrix}$$

$$= (\text{Rank } E_1)(\text{Rank } I_{n_2}) + (\text{Rank } \bar{E}_1)(\text{Rank } E_2) \tag{2.24}$$

by Lemma 2.5, since Rank $\begin{bmatrix} \bar{E}_1 \\ E_1 \end{bmatrix}$ = Rank $\bar{E}_1$ + Rank E_1.

$$\left.\begin{aligned} \text{But}\quad & \text{Rank } E_1 = n_1 - \text{Rank } D_1 \\ & \text{Rank } I_{n_2} = n_2 \\ & \text{Rank } E_1 = \text{Rank } I_{n_1} - \text{Rank } E_1 = n_1 - (n_1 - \text{Rank } D_1) \\ & \qquad = \text{Rank } D_1 \\ & \text{Rank } E_2 = n_2 - \text{Rank } D_2 \ . \end{aligned}\right\} \tag{2.25}$$

Substituting Equation (2.25) to Equation (2.24), we have

$$\text{Rank}\begin{bmatrix} E_1 \otimes \bar{E}_2 \\ E_1 \otimes E_2 \\ \bar{E}_1 \otimes E_2 \end{bmatrix} = (n_1 - \text{Rank } D_1)\, n_2 + (\text{Rank } D_1)(n_2 - \text{Rank } D_2)$$

$$= n_1 n_2 - n_2 \text{Rank } D_1 + n_2 \text{Rank } D_1 - (\text{Rank } D_1)(\text{Rank } D_2)$$

$$= n_1 n_2 - (\text{Rank } D_1)(\text{Rank } D_2)$$

$$= \text{Rank } E$$

Q.E.D.

Corollary 2.6.1

Theorem 2.6 is equivalent to the statement that if $D = D_1 \otimes D_2$, then

$$E = \begin{bmatrix} I_{n_1} \otimes E_2 \\ E_1 \otimes I_{n_2} \end{bmatrix} .$$

Proof

By Theorem 2.6,

$$\text{row space of } E = \text{row space of } \begin{bmatrix} E_1 \otimes \overline{E}_2 \\ E_1 \otimes E_2 \\ \overline{E}_1 \otimes E_2 \end{bmatrix} = \text{row space of } \begin{bmatrix} E_1 \otimes \overline{E}_2 \\ E_1 \otimes E_2 \\ E_1 \otimes E_2 \\ \overline{E}_1 \otimes E_2 \end{bmatrix}$$

$$= \text{row space of } \begin{bmatrix} E_1 \otimes \begin{bmatrix} \overline{E}_2 \\ E_2 \end{bmatrix} \\ \begin{bmatrix} E_1 \\ \overline{E}_1 \end{bmatrix} \otimes E_2 \end{bmatrix} = \text{row space of } \begin{bmatrix} E_1 \otimes I_{n_2} \\ I_{n_1} \otimes E_2 \end{bmatrix},$$

$$\text{since row space of } I_{n_1} = \text{row space of } \begin{bmatrix} \overline{E}_1 \\ E_1 \end{bmatrix}$$

$$= \text{row space of } \begin{bmatrix} E_1 \\ \overline{E}_1 \end{bmatrix} . \qquad \text{Q.E.D.}$$

Theorem 2.6 gives a method of obtaining a set of basis of the null space of the matrix $D = D_1 \otimes D_2$. The basis is divided into three parts; namely $E_1 \otimes \overline{E}_2$, $E_1 \otimes E_2$, and $\overline{E}_1 \otimes E_2$. The row space of E can be expressed as the direct sum of its subspaces

(row space of $E_1 \otimes \overline{E}_2$) $\oplus$ (row space of $E_1 \otimes E_2$) $\oplus$ (row space of $\overline{E}_1 \otimes E_2$).

It should be noted that the spaces of the matrices $\overline{E}_1$ and $\overline{E}_2$ may be equal to those of the matrices D_1 and D_2 respectively*, but that, in general, they are not. When the spaces of the $\overline{E}$ matrices differ from the spaces of the D matrices, it is sometimes a rather tedious procedure to evaluate the $\overline{E}$ matrices from the E matrices which are not in echelon form. On the other hand, Corollary 2.6.1 gives the null space of $D = D_1 \otimes D_2$ with a minimum amount of calculation and no unknown matrices to be evaluated. The row vectors of this representation of the null space are, however, not linearly independent.

<u>Example</u>

Given

$$D_1 = \begin{bmatrix} 1&0&0&1&1&1 \\ 0&1&0&1&0&1 \\ 0&0&1&0&1&1 \end{bmatrix}, \quad D_2 = \begin{bmatrix} 1&0&1 \\ 0&1&1 \end{bmatrix}$$

$$D = D_1 \otimes D_2 = \begin{bmatrix} 1&0&1&0&0&0&0&0&0&1&0&1&1&0&1&1&0&1 \\ 0&1&1&0&0&0&0&0&0&0&1&1&0&1&1&0&1&1 \\ 0&0&0&1&0&1&0&0&0&1&0&1&0&0&0&1&0&1 \\ 0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1 \\ 0&0&0&0&0&0&1&0&1&0&0&0&1&0&1&1&0&1 \\ 0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&1&1 \end{bmatrix}.$$

*This means that the complementary spaces of E_1 and E_2 coincide with their null spaces.

It can be easily shown that

$$E_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\bar{E}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$E_2 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\bar{E}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} .$$

By Theorem 2.6, we have

$$E = \begin{bmatrix} E_1 \otimes \bar{E}_2 \\ E_1 \otimes E_2 \\ \bar{E}_1 \otimes E_2 \end{bmatrix}$$

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

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

By Corollary 2.6.1, we have

$$E = \begin{bmatrix} I_{n_1} \otimes E_2 \\ E_1 \otimes I_{n_2} \end{bmatrix}$$

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

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

Although this representation has 15 rows, the actual rank of E remains to be 12.

2.3 <u>The Determination of the Null Space of a Tensor Product Space Involving the Translation of Fields</u> - Elements of Two Original Component Codes and Elements of the Resultant Code Expressed in Different Fields

When the gp matrices of two original component codes D_1 and D_2 are over $GF(p^\ell)$, the elements of the resultant tensor product code given by $D_1 \otimes D_2$ are first expressed in terms of field elements of $GF(p^\ell)$, then translated into the field elements of GF(p) according to certain rules. The technique described in Section 2.2 is not applicable, in general, for this case since a change of the rank of the matrix may occur due to the field translation.

As the companion matrix plays an important role in the development of this section, a brief explanation on this matter is warranted. Let $r(x) = r_0 + r_1x + \dots + r_{\ell-1}x^{\ell-1} + x^{\ell}$ be a monic polynomial, then the companion matrix of the polynomial is defined as

$$T_c = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ -r_0 & -r_1 & -r_2 & -r_3 & \dots & -r_{\ell-1} \end{bmatrix}. \qquad (2.26)$$

Sometimes the companion matrix is defined as the transpose of the expression in (2.26)

$$T_o = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & -r_0 \\ 1 & 0 & 0 & \dots & 0 & -r_1 \\ 0 & 1 & 0 & \dots & 0 & -r_2 \\ 0 & 0 & 1 & \dots & 0 & -r_3 \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ 0 & 0 & 0 & \dots & 1 & -r_{\ell-1} \end{bmatrix}. \qquad (2.27)$$

For convenience, both matrices of (2.26) and (2.27) are called the companion matrix of r(x). They can be distinquished by the notations T_c and T_o.

There are two methods of obtaining p-ary codes through the fields translation from the q-ary to the p-ary. The first method is shown by

Cocke[35] who shows the following way of fields translation: the parity-check matrix H_q of a q-ary code is first chosen, then each element is expanded into an $\ell \times \ell$ p-ary square matrix, where $q = p^\ell$, in the following manner:

$$\begin{aligned}
0 &\Longrightarrow [0_{\ell\times\ell}] \\
\beta^0 &\Longrightarrow T_0^0 = I_\ell \\
\beta^1 &\Longrightarrow T_0 \\
&\vdots \\
\beta^i &\Longrightarrow T_0^i \\
&\vdots \\
\beta^{q-2} &\Longrightarrow T_0^{q-2} ,
\end{aligned}$$

where, β is a primitive element of GF(q), and T_0 is the companion matrix of the primitive polynomial over GF(p) with β as its primitive root.
It can be shown* that the expanded p-ary code is generated by the generator matrix which is obtained by the expanded G_q, the generator matrix of the original q-ary code, in the following manner:

$$\begin{aligned}
0 &\Longrightarrow [0_{\ell\times\ell}] \\
\beta^0 &\Longrightarrow T_c^0 = I_\ell \\
\beta^1 &\Longrightarrow T_c \\
&\vdots \\
\beta^i &\Longrightarrow T_c^i \\
&\vdots \\
\beta^{q-2} &\Longrightarrow T_c^{q-2} .
\end{aligned}$$

*See Appendix, Theorem A-4.

It has been pointed out that[33] a code obtained from the fields translation in this manner is suitable for burst-error correction. It should be noted that the minimum distances, the total number of code words and the data transmission rate of both the q-ary code and its expanded p-ary code are identical, but an increase of code word length by ℓ times results from the translation. The second method is useful in the design of a p-ary Bose-Chaudhuri-Hocquenghem code. In such a design it is convenient to express first the elements of the parity-check matrix in terms of the roots, which are elements of GF(q) with $q = p^{\ell}$, of the code generator polynomial g(x) over GF(p), and then expand the matrix into its p-ary representation according to the following rules:

$$0 \Rightarrow \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

$$\beta^0 \Rightarrow \beta^0\Big] = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

$$\beta^1 \Rightarrow \beta^1\Big] = T_0\beta^0\Big]$$

$$\beta^2 \Rightarrow \beta^2\Big] = T_0{}^2\beta^0\Big]$$

$$\vdots$$

$$\beta^i \Rightarrow \beta^i\Big] = T_0{}^i\beta^0\Big]$$

$$\vdots$$

$$\beta^{q-2} \Rightarrow \beta^{q-2}\Big] = T_0{}^{q-2}\beta^0\Big] ,$$

where β^1 and T_0 are defined as before. If the row space of the original q-ary matrix is cyclic, the cyclic property is unchanged by the field translation made in the second method. However, such is not the case under the field translation of the first method. Furthermore, it is possible although less convenient to obtain the p-ary B-C-H parity-check matrix without employing any field translation. This is not feasible if the first method is used.

An efficient error-locating code can be obtained by employing the technique of field translation. Its structure can be understood from two different viewpoints:

(i) The parity-check matrix of size $r_1 \times n_1$ of a q-ary t-error correcting is first designed. Each q-ary element is then mapped into an $r_2 \times n_2$ p-ary matrix, where $r_2 < n_2$, instead of a square matrix in the following manner:

$$
\begin{aligned}
0 &\Rightarrow [0_{r_2 \times n_2}] \\
\beta^0 &\Rightarrow A = \left[I_{r_2} \,\vdots\, P \right] \quad \text{or} \quad \left[I_{r_2} \,\vdots\, P \right] \pi \\
\beta^1 &\Rightarrow T_0 A \\
\beta^2 &\Rightarrow T_0{}^2 A \\
&\;\vdots \\
\beta^i &\Rightarrow T_0{}^i A \\
&\;\vdots \\
\beta^{q-2} &\Rightarrow T_0{}^{q-2} A \,,
\end{aligned}
$$

where A is the p-ary parity-check matrix of an s-error-detecting code. It can be easily shown that the row space of $T_0{}^iA$ is identical to that of A. This viewpoint does not require the concept of the tensor product of codes.

(ii) The parity-check matrix is considered as the p-ary representation of the tensor product of two q-ary parity-check matrices, denoted by $H = [H_{1q} \otimes H_{2q}]_p$. The p-ary representation of H_{2q} obtained by using the second method is identical to the matrix A as described in (i).

It is easier to evaluate the code space from the tensor product viewpoint, especially when H_{2q} is convertible into a single-rowed matrix. This conversion can be easily performed when the second component code is cyclic. A cyclic p-ary code with s rows in its parity-check matrix can be made identical to a cyclic code with a one-rowed p^ℓ-ary check matrix, provided $\ell = rs$ and a suitable mapping is performed from $GF(p^r)$ to $GF(p^\ell)$. The following theorem is useful in determining the code space of such a product code.

<u>Theorem 2.7</u>

Let D_{1q} and E_{1q} be respectively the gp and pg matrices of a q-ary code C_1 of length n_1, where $q = p^\ell$. If

(i) D_{2q} is a $1 \times n_2$ pg matrix of a cyclic code C with $n_2 \geq \ell$;

(ii) with suitable mapping, D_{2q} is of the form

$$\left[\beta^0\ \beta^1\ \beta^2 \ldots \beta^{\ell-1} \;\vdots\; \beta^{i_\ell}\ \beta^{i_{\ell+1}} \ldots \beta^{i_{n_2-1}}\right],$$

where β^1 is a primitive element of GF(q);

and (iii) the row space of E_{2p} is the null space of $[D_{2q}]_p$, then the null space of $[D_{1q} \otimes D_{2q}]_p$ is

$$\begin{bmatrix} I_{n_1} \otimes E_{2p} \\ E_{1q} \otimes \left[\begin{array}{c|c} \beta^0 & \\ \beta^1 & \\ \vdots & 0_{COM} \\ \beta^{\ell-1} & \end{array}\right] \end{bmatrix}_{\lfloor p \rfloor},$$

where $[\quad]_{\lfloor p \rfloor}$ is the matrix with each element in the bracket expressed as a row vector over GF(p) in the following manner

$$\beta^i \Rightarrow \lfloor \beta^i \rfloor = \left.\beta^i\right]^T ,$$

and 0_{COM} is introduced for the sake of compensating the length of each submatrices in the following manner:

$$\left[\beta^i \left[\begin{array}{c|c} \beta^0 & \\ \beta^1 & \\ \vdots & 0_{COM} \\ \beta^{\ell-1} & \end{array}\right]\right]_{\lfloor p \rfloor} = \left[\begin{array}{c|c} \lfloor \beta^i \rfloor & \\ \lfloor \beta^{i+1} \rfloor & \\ \vdots & 0_{\ell\times(n_2-\ell)} \\ \lfloor \beta^{i+\ell-1} \rfloor & \end{array}\right].$$

The proofs of this theorem and its preliminary lemmas are given in the Appendix.

We close this section with an example to illustrate Theorem 2.7. Let

$$D_{1q} = \begin{bmatrix} \beta^0 & 0 & \beta^0 & \beta^2 & \beta^5 & \beta^6 \\ 0 & \beta^0 & \beta^2 & \beta^5 & \beta^6 & \beta^6 \end{bmatrix}$$

$$D_{2q} = \begin{bmatrix} \beta^0 & \beta^1 & \beta^2 & \beta^3 & \beta^4 \end{bmatrix}$$

$$\left.\beta^0\right] = \begin{bmatrix}1\\0\\0\end{bmatrix}; \quad \left.\beta^1\right] = \begin{bmatrix}0\\1\\0\end{bmatrix}; \quad \left.\beta^2\right] = \begin{bmatrix}0\\0\\1\end{bmatrix}; \quad \left.\beta^3\right] = \begin{bmatrix}1\\1\\0\end{bmatrix};$$

$$\left.\beta^4\right] = \begin{bmatrix}0\\1\\1\end{bmatrix}; \quad \left.\beta^5\right] = \begin{bmatrix}1\\1\\1\end{bmatrix}; \quad \left.\beta^6\right] = \begin{bmatrix}1\\0\\1\end{bmatrix}; \quad \left.\beta^7\right] = \left.\beta^0\right].$$

Therefore,

$$D_{2p} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} = \text{binary representation of } D_{2q}$$

$$E_{2p} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

The row space of D_{2p} and that of E_{2p} are the null spaces of each other.

$$E_{1q} = \begin{bmatrix} \beta^0 & \beta^2 & \beta^0 & 0 & 0 & 0 \\ \beta^2 & \beta^5 & 0 & \beta^0 & 0 & 0 \\ \beta^5 & \beta^6 & 0 & 0 & \beta^0 & 0 \\ \beta^6 & \beta^6 & 0 & 0 & 0 & \beta^0 \end{bmatrix}$$

$$\left[D_{1q}\otimes D_{2q}\right]_p = \left[\begin{array}{c|c|c|c|c|c}
\begin{matrix}10010\\01011\\00101\end{matrix} & \bigcirc & \begin{matrix}10010\\01011\\00101\end{matrix} & \begin{matrix}01011\\01110\\10111\end{matrix} & \begin{matrix}11100\\10010\\11001\end{matrix} & \begin{matrix}11001\\00101\\10010\end{matrix} \\ \hline
\bigcirc & \begin{matrix}10010\\01011\\00101\end{matrix} & \begin{matrix}01011\\01110\\10111\end{matrix} & \begin{matrix}11100\\10010\\11001\end{matrix} & \begin{matrix}11001\\00101\\10010\end{matrix} & \begin{matrix}11001\\00101\\10010\end{matrix}
\end{array}\right].$$

Then the null space of $\left[D_{1q}\otimes D_{2q}\right]_p$ is, by Theorem 2.7,

$$\left[\begin{array}{c}
I_{n_1}\otimes E_{2p} \\
\left[E_{1q}\otimes\left[\begin{array}{c|c}\beta^0 & \\ \beta^1 & 0_{COM} \\ \beta^2 & \end{array}\right]\right]
\end{array}\right]_p$$

$$= \left[\begin{array}{c|c|c|c|c|c}
\begin{matrix}11010\\01101\end{matrix} & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc \\ \hline
\bigcirc & \begin{matrix}11010\\01101\end{matrix} & \bigcirc & \bigcirc & \bigcirc & \bigcirc \\ \hline
\bigcirc & \bigcirc & \begin{matrix}11010\\01101\end{matrix} & \bigcirc & \bigcirc & \bigcirc \\ \hline
\bigcirc & \bigcirc & \bigcirc & \begin{matrix}11010\\01101\end{matrix} & \bigcirc & \bigcirc \\ \hline
\bigcirc & \bigcirc & \bigcirc & \bigcirc & \begin{matrix}11010\\01101\end{matrix} & \bigcirc \\ \hline
\bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \begin{matrix}11010\\01101\end{matrix} \\ \hline
\begin{matrix}10000\\01000\\00100\end{matrix} & \begin{matrix}00100\\11000\\01100\end{matrix} & \begin{matrix}10000\\01000\\00100\end{matrix} & \bigcirc & \bigcirc & \bigcirc \\ \hline
\begin{matrix}00100\\11000\\01100\end{matrix} & \begin{matrix}11100\\10100\\10000\end{matrix} & \bigcirc & \begin{matrix}10000\\01000\\00100\end{matrix} & \bigcirc & \bigcirc \\ \hline
\begin{matrix}11100\\10100\\10000\end{matrix} & \begin{matrix}10100\\10000\\01000\end{matrix} & \bigcirc & \bigcirc & \begin{matrix}10000\\01000\\00100\end{matrix} & \bigcirc \\ \hline
\begin{matrix}10100\\10000\\01000\end{matrix} & \begin{matrix}10100\\10000\\01000\end{matrix} & \bigcirc & \bigcirc & \bigcirc & \begin{matrix}10000\\01000\\00100\end{matrix}
\end{array}\right].$$

2.4 Remarks

The direct method of finding the null space is straightforward but tedious. However, it is the one method which remains applicable when the other two methods fail. When the elements of the two original component codes and the elements of the resultant product code are expressed in different fields, usually the resultant code elements are in the ground field and the elements of component codes are in one of the extension fields. Theorem A-3 of the Appendix gives the row space of

$$\begin{bmatrix} E_{1p} \otimes I_{n_2} \\ I_{n_1} \otimes E_{2p} \end{bmatrix}$$

only as a subspace of the null space of $\left[D_{1q} \otimes D_{2q} \right]_p$. In this case the direct method is recommended for determining the whole null space over the ground field, and Theorem A-3 can be used to lend some insight of the code structure.

It is interesting to note that, by Corollary 2.6.1 and Theorem A-3 that while

$$\begin{bmatrix} I_{n_1} \otimes E_{2p} \\ E_{1p} \otimes I_{n_2} \end{bmatrix}$$

completely specifies the null space of $\left[D_{1p} \otimes D_{2p} \right]$, the same sapce

$$\begin{bmatrix} I_{n_1} \otimes E_{2p} \\ E_{1p} \otimes I_{n_2} \end{bmatrix}$$

is just a subspace of the null space of $\left[D_{1q} \otimes D_{2q} \right]_p$. It can further be shown that the row space of $\left[D_{1q} \otimes D_{2q}\right]_p$ is a subspace of $[D_{1q}]_p \otimes [D_{2q}]_p$. While the circuits for implementing $[D_{1q} \otimes D_{2q}]_p$ and $[D_{1q}]_p \otimes [D_{2q}]_p$ are of about the same order of complexity, the code space specified by the latter is larger than that of the former. Therefore, the adoption of the latter as the G matrix of a product code is preferred. By similar reasoning, a product code with parity check matrix $H = [H_{1q} \otimes H_{2q}]_p$ is better than that with $H' = [H_{1q}]_p \otimes [H_{2q}]_p$.

Theorem 2.6 can be extended to be applicable for higher order tensor product codes as follows. The null space of the row space of

$$D = D_1 \otimes D_2 \otimes D_3$$

is the row space of

$$E = \begin{bmatrix} E_1 \otimes E_2 \otimes E_3 \\ \bar{E}_1 \otimes E_2 \otimes E_3 \\ E_1 \otimes \bar{E}_2 \otimes E_3 \\ E_1 \otimes E_2 \otimes \bar{E}_3 \\ \bar{E}_1 \otimes \bar{E}_2 \otimes E_3 \\ E_1 \otimes \bar{E}_2 \otimes \bar{E}_3 \\ \bar{E}_1 \otimes E_2 \otimes \bar{E}_3 \end{bmatrix},$$

where the row space of E_i is the null space of D_i for $i = 1,2,3$. Likewise, Corollary 2.6.1 can be extended to read that the null space of the row space of

$$D = D_1 \otimes D_2 \otimes D_3$$

is the row space of

$$\begin{bmatrix} I_{n_1} \otimes I_{n_2} \otimes E_3 \\ I_{n_1} \otimes E_2 \otimes I_{n_3} \\ E_1 \otimes I_{n_2} \otimes I_{n_3} \end{bmatrix} .$$

It is not difficult to generalize the above results to the case of $D = D_1 \otimes D_2 \otimes \dots \otimes D_j$ for any j.

Theorem 2.7 is useful to determine the null space of the row space of $[D_{1q} \otimes D_{2q}]_p$, provided D_{2q} is the one-rowed q-ary gp matrix of a cyclic code. In the Appendix, this theorem has been extended to cover the case when the second component code is not cyclic. Also the difficulties are pointed out when D_{2q} is a multi-rowed matrix.

CHAPTER 3

ENCODING AND DECODING OF TENSOR PRODUCT CODES

Usually, there are two schemes for encoding a linear code. The first scheme is simply choosing either prestored basis code vectors or a linear combination of such vectors. The other method is to calculate the parity-check digits from the given information digits according to the parity-check matrix of the code, then the check digits and the information digits are combined together to form a code word.

There are also two general methods for decoding a linear code. One is to compare the received sequence with the code word dictionary. The code word which has the smallest distance from the received sequence is considered to be the original code word sent by the transmitter. The other method is to check whether the received sequence satisfies the code parity-check matrix. Let $\underline{r}$ be the received sequence. The syndrome of any received sequence is defined to be

$$\underline{s} = \underline{r}\, H^T \tag{3.1}$$

where H is the parity-check matrix. The decoding procedure is to evaluate the syndrome first, then to determine the error pattern or to detect the errors from the calculated syndrome.

The encoding and decoding of a product code can be achieved by similar schemes. But in so doing, we do not take full advantage of the product code. Hence an alternate scheme should be used. The main purposes of this chapter are as follows:

(i) to design the encoding circuit for a product code by a logic combination of the encoding circuits of its component codes;

(ii) to design the syndrome calculator for a product code by a logic combination of the syndrome calculators of its component codes;

(iii) to decompose the decoding procedure of a product code into the decoding procedures of its component codes.

The designs will first be directed to the type of product codes in which $H = H_1 \otimes H_2$, then to the type in which $G = G_1 \otimes G_2$.

The discussion and results in this chapter are intended to be as general as possible, hence they are applicable to any product code with linear component codes. Specific examples will be deferred until later chapters. It is believed that the encoding and decoding schemes presented here for the product codes with parity-check matrices $H = H_1 \otimes H_2$ and those with $H = [H_{1q} \otimes H_{2q}]_p$ have not been worked out in such a detail elsewhere. It is also believed that the decoding scheme described in Section 3.8 is the most straightforward technique in comparison with other existing schemes.

3.1 Encoding by the Parity-Check Matrix H

Systematic encoding* for a linear code whose parity-check matrix is specified can be accomplished by the check digits calculator of the code

*See Hamming[7] for the definition of systematic code.

shown in Fig. 3.1. The input sequence consists of information digits

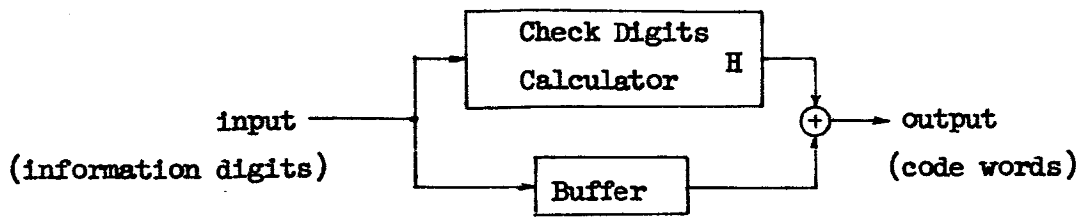


Fig. 3.1 Check Digits Calculator for Encoding a Linear Code

and all zeros in checking positions. The check digits calculator gives the check digits for the input information digits. The check digits are added to the input sequence to form the encoded output sequence or a code word. For simplicity, Fig. 3.1 is symbolically replaced by Fig. 3.2.

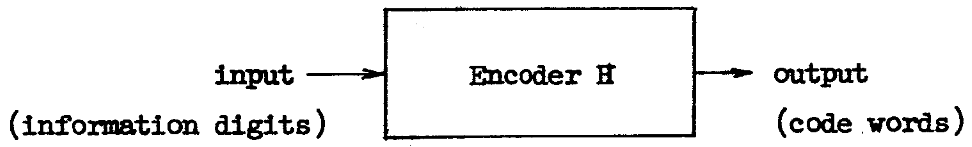


Fig. 3.2 A Block Representation of Fig. 3.1

3.2 Encoding of a Product Code with Parity-Check Matrix $H = H_1 \otimes H_2$

In this section and the following two sections we deal only with the case that the elements of the resultant product code and the elements of its component codes are in the same field. The case involving translation of fields will be discussed in Sections 3.9 and 3.10.

Fig. 3.3 shows the encoder for a product code with parity-check matrix $H = H_1 \otimes H_2$. The inputs to the scalar multipliers,

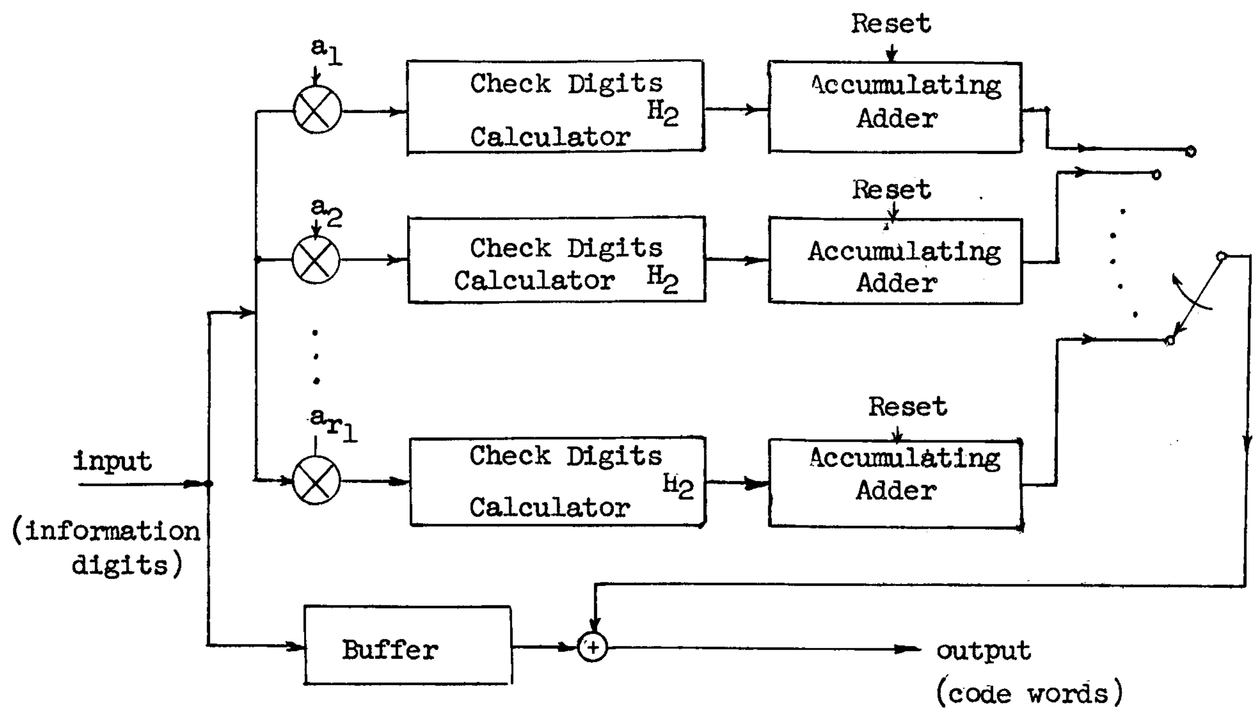


Fig. 3.3 An Encoder for the Product Code with Parity-Check Matrix $H = H_1 \otimes H_2$

$a_1, a_2, \ldots, a_{r_1}$, are set equal to the components of $(h_{1j}, h_{2j}, \ldots, h_{r_1 j})^T$ respectively when the n_1-j+1, $n_1-j+2, \ldots, n_1-j+n_2$'th digits of the input sequence are fed into the encoder. $(h_{1j}, h_{2j}, \ldots, h_{r_1 j})^T$ is the j^{th} column of the matrix H_1, an $r_1 \times n_1$ matrix. The check digit calculators H_2 calculate the check digits once for every subblock containing n_2 digits of the input sequence. The accumlating adders over GF(q) are adders with memories; each new input to the adder is added (over GF(q)) to the value previously stored. At the end of each code word length the adders should be reset to zero. The outputs of the adders are added in proper order to the read-out version of the buffered input sequence. For the sake of explanation, let us assume H_1 and H_2 are the parity-check matrices of systematic codes, in other words

$$H_1 = [I_{r_1}\ P_1] \tag{3.2}$$

$$H_2 = [I_{r_2}\ P_2], \tag{3.3}$$

then

$$H_1 \otimes H_2 = \left[\begin{array}{ccccc|c} [I_2\ P_2] & & & & & \\ & [I_2\ P_2] & & & & \\ & & [I_2\ P_2] & & \bigcirc & \\ & & & \cdot & & P_1 \otimes H_2 \\ & \bigcirc & & & \cdot & \\ & & & & \cdot & \\ & & & & [I_2\ P_2] & \end{array}\right]. \tag{3.4}$$

It is evident that the columns of $H_1 \otimes H_2$ containing only one single non-zero element (which is usually the unity element) are the positions of the parity-check digits. They correspond to the columns in which the unity matrices I_2's appear in the matrix of Equation (3.4). To these positions the outputs of the accumlating adders are attached. These check digits, called for by a synchronized switch, are included in the final code word output.

3.3 Syndrome Calculation of the Product Code with $H = H_1 \otimes H_2$

The first step of decoding is to calculate the syndrome of the sequence to be decoded. The syndrome calculator is similar to the main part of the encoder. A syndrome calculator for the product code $H = H_1 \otimes H_2$ is shown in Fig. 3.4.

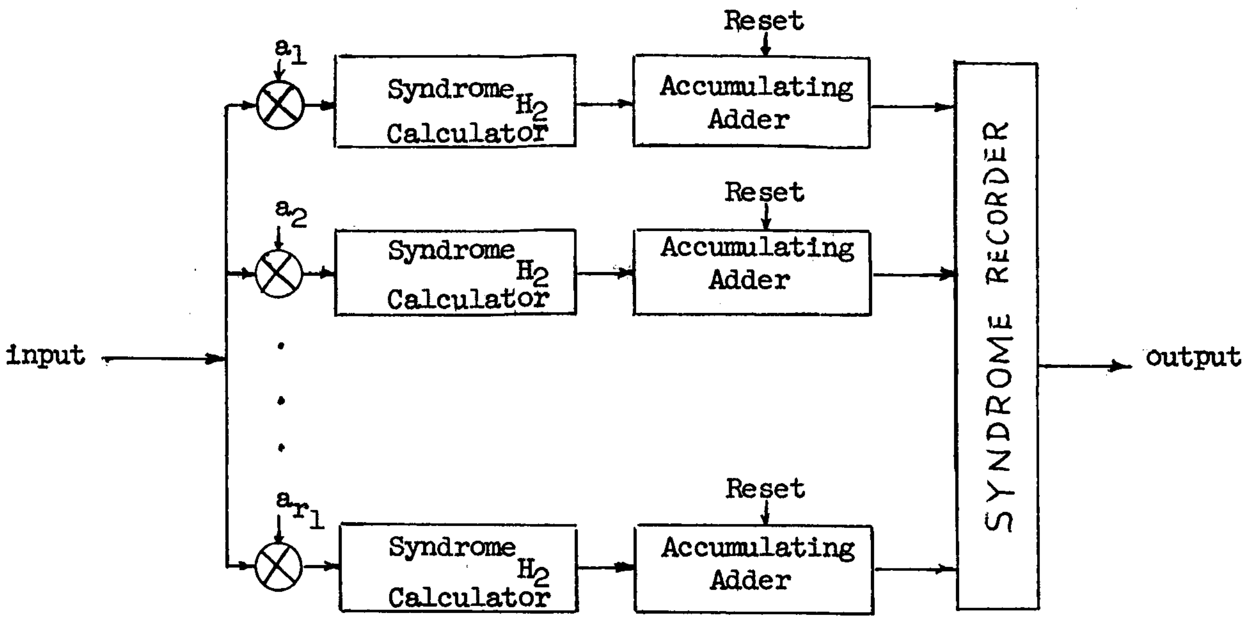


Fig. 3.4 An Syndrome Calculator for the Product Code $H = H_1 \otimes H_2$

The interpretation of the functions of each block or box for Fig. 3.3 is also applicable to Fig. 3.4. The syndrome recorder is either a buffer circuit to store the syndrome for future uses or a switch which sends out the "subsyndromes" out of each adder in a certain proper order.

3.4 Decoding of a Product Code $H = H_1 \otimes H_2$ After Syndrome Calculation

A special feature of the product code with parity-check matrix $H = H_1 \otimes H_2$ is that it may be conveniently used as an error-locating code. Let H_1 and H_2 be $r_1 \times n_1$ and $r_2 \times n_2$ matrices respectively. Then a whole code word of length $n_1 n_2$ can be divided into n_1 subblocks, each of length n_2. The parity-check matrix H_2 of the second component code specifies the maximum detectable errors in each corrupted subblock. The first component code parity-check matrix H_1 pinpoints the positions of the erroneous subblocks without correcting the erroneous digits within the subblocks.

When the syndrome of any received sequence is not a vector with all zero components or coordinates, there are some erroneous digits in the received sequence. We will be able to locate the erroneous subblocks from the calculated syndrome. Let H_2 be expressed as a single row matrix, namely,

$$H_2 = \left[\beta^0 \; \beta^{i_1} \; \beta^{i_2}, \ldots, \beta^{i_{n_2-1}}\right]$$

where β^1 is a primitive element of $GF(p^\ell)$. Also let H_1 be the parity-check matrix expressed as elements over $GF(p^\ell)$, of a t-error-correcting code, then $H = H_1 \otimes H_2$ is the t-subblock-error-locating code parity-check matrix provided the number of erroneous digits in each corrupted subblock does not exceed the error detectability of the second component code with parity check matrix H_2. The reason of error-locatability is as follows: Suppose the received code word is divided into n_1 subblocks each with length n_2, where n_1 is the code length of the first component code and n_2 that of the second component code. Then any error-pattern in any subblock which is detectable by H_2 will yield the non-zero element β^j as a syndrome. Hence if we consider each subblock as a digit (over $GF(p^\ell)$) represented by the syndrome based on H_2, the calculation of the new syndrome (based on H_1) of the sequence of n_1 digits (subblocks) will pinpoint the particular digits (subblocks) in error.

Let us now consider the case where H_2 is a multiple-row matrix with elements in $GF(q)$:

$$H_2 = \begin{bmatrix} h_{11}^{(2)} & h_{12}^{(2)} & \cdots & h_{1n_2}^{(2)} \\ h_{21}^{(2)} & h_{22}^{(2)} & \cdots & h_{2n_2}^{(2)} \\ \vdots & & & \\ h_{r_21}^{(2)} & h_{r_22}^{(2)} & \cdots & h_{r_2n_2}^{(2)} \end{bmatrix} = \begin{bmatrix} \underline{h_1} \\ \underline{h_2} \\ \vdots \\ \underline{h_{r_2}} \end{bmatrix} . \tag{3.5}$$

Hence,

$$H = H_1 \otimes H_2 = H_1 \otimes \begin{bmatrix} \underline{h_1} \\ \underline{h_2} \\ \vdots \\ \underline{h_{r_2}} \end{bmatrix} , \tag{3.6}$$

but

$$\text{row space of } H_1 \otimes \begin{bmatrix} \underline{h_1} \\ \underline{h_2} \\ \vdots \\ \underline{h_{r_2}} \end{bmatrix} = \text{row space of } \begin{bmatrix} H_1 \otimes \underline{h_1} \\ H_1 \otimes \underline{h_2} \\ \vdots \\ H_1 \otimes \underline{h_{r_2}} \end{bmatrix} . \tag{3.7}$$

Hence,

$$H = \begin{bmatrix} H_1 \otimes \underline{h_1} \\ H_1 \otimes \underline{h_2} \\ \vdots \\ H_1 \otimes \underline{h_{r_2}} \end{bmatrix} . \tag{3.8}$$

Equation (3.8) can be obtained from Equation (3.6) by row permutation. Each submatrix $H_1 \otimes \underline{h_i}$ for $i = 1,2,\ldots,r_2$ can be treated as the H matrix of the previous paragraph. It should be borne in mind that in the previous paragraph $\lfloor e \rfloor H_2^T \neq [0]$ for any detectable error pattern $\lfloor e \rfloor$ in each subblock. In the present case it is possible that $\lfloor e \rfloor h_i] = [0]$ for some i and certain detectable error pattern $\lfloor e \rfloor$. However, there exists at least one i such that $\lfloor e \rfloor h_i] \neq [0]$ for any detectable error pattern $\lfloor e \rfloor$ in each subblock. Otherwise we will have $\lfloor e \rfloor H_2 = [0]$, which is contradictory to the definition of detectable error pattern. We first look at $H \otimes \lfloor h_1 \rfloor$ to determine how many subblocks are in error. Then we investigate $H_1 \otimes \lfloor h_2 \rfloor$, $H_1 \otimes \lfloor h_3 \rfloor, \ldots, H_1 \otimes \lfloor h_{r_2} \rfloor$. Any subblock whose error pattern is detectable by the parity-check matrix H_2 can be located by at least one of the submatrices $H_1 \otimes \lfloor h_i \rfloor$ for $i = 1,2,3,\ldots,r_2$. The decoding procedure is now summarized as follows:

(1) Calculate the syndrome of the received code word according to the parity-check matrix $H = H_1 \otimes H_2$.

(2) Rearrange the components of the syndrome in the following manner

$$S \Rightarrow S' , \tag{3.9}$$

where

S = original syndrome

$$= \Big[S_1\ S_2, \ldots, S_{r_2}\ S_{r_2+1}\ S_{r_2+2}, \ldots, S_{r_2+r_2}, \ldots,$$

$$S_{(r_1-1)r_2+1}\ S_{(r_1-1)r_2+2}, \ldots, S_{r_1 r_2} \Big] , \tag{3.10}$$

and

S' = rearranged new syndrome

$$= \Big[S_1 \; S_{r_2+1} \; S_{2r_2+1}, \ldots, S_{(r_1-1)r_2+1} \;\vdots\; S_2 \; S_{r_2+2} \; S_{2r_2+2}, \ldots, S_{(r_1-1)r_2+2} \;\vdots\; \cdots \;\vdots\; S_{r_2} \; S_{2r_2}, \ldots, S_{r_1 r_2} \Big]$$

$$= \Big[S'_{b_1} \; S'_{b_2}, \ldots, S'_{b_i}, \ldots, S'_{b r_2} \Big] , \tag{3.11}$$

where

$$S'_{b_i} = \Big[S_i \; S_{r_2+i}, \ldots, S_{(r_1-1)r_2+i} \Big] \tag{3.12}$$

$$\text{for } i = 1,2,\ldots,r_2,$$

i.e., S'_{b_i} is the syndrome obtained from $H_1 \otimes \underline{h_i}$.

(3) Evaluate the erroneous digit positions of the first component code with the parity-check matrix H_1 by assigning S'_{b_i}'s as its syndromes for all i's.

(4) Identify the digit positions of the component code H_1 with the subblock positions of the product code.

For the case of binary product codes with parity-check matrix $H = H_1 \otimes H_2$, the decoding procedures become much simpler. No rearrangement of the syndrome components is required in this case. Let us divide the original syndrome S in the following manner:

$$S = \left[S_1\ S_2, \ldots, S_{r_2} \,\vdots\, S_{r_2+1}\ S_{r_2+2}, \ldots, S_{r_2+r_2} \,\vdots\, \cdots \,\vdots\, S_{(r_1-1)r_2+1}\ S_{(r_1-1)r_2+2}, \ldots, S_{r_1 r_2} \right]$$

$$= \left[S_{a_1}\ S_{a_2}, \ldots, S_{a_i}, \ldots, S_{a_{r_1}} \right] , \qquad (3.13)$$

where

$$S_{a_i} = \left[S_{(i-1)r_2+1}\ S_{(i-1)r_2+2}, \ldots, S_{(i-1)r_2+r_2} \right] \text{ for } i = 1,2,\ldots,r_1 . \qquad (3.14)$$

Let us define a mapping by

$$\left. \begin{array}{l} S_{a_i} \Rrightarrow 0 \text{ if } S_{(i-1)r_2+k} = 0 \text{ for } k = 1,2,\ldots,r_2 \\ S_{a_i} \Rrightarrow 1 \text{ if there exists at least one } S_{(i-1)r_2+k} = 1 \end{array} \right\} . \qquad (3.15)$$

For example, let $r_2 = 3$, $r_1 = 7$ and

$$S = \left[010 \,\vdots\, 000 \,\vdots\, 011 \,\vdots\, 111 \,\vdots\, 000 \,\vdots\, 001 \,\vdots\, 000 \right] ,$$

then

$$S_1 = [010] \Rrightarrow 1$$

$$S_2 = [000] \Rrightarrow 0$$

$$S_3 = [011] \Rrightarrow 1$$

$$S_4 = [111] \Rrightarrow 1$$

$$S_5 = [000] \Rrightarrow 0$$

$$S_6 = [001] \Rrightarrow 1$$

$$S_7 = [000] \Rrightarrow 0 .$$

Let S_{H_1} denote the syndrome obtained from S by mapping S_{a_i} into either 0 or 1. Treat S_{H_1} as the syndrome of the first component code. We can then evaluate the erroneous digit positions of the first component code which correspond to the corrupted subblocks:

3.5 Encoding by the Generator Matrix G

It is well known that the rows of the generator matrix G of a linear code are nothing but a set of the basis vectors of the code space. Any code word of the code space can be expressed as a linear combination of the basis vectors. Therefore, the encoder can be constructed as shown in Fig. 3.5, where $v_1, v_2, \ldots, v_k$ are prestored row vectors of the generator matrix G. $G_1, G_2, \ldots, G_k$ are gated amplifiers whose gains are controlled by the input information sequence. $T_1, T_2, \ldots, T_{k-1}$ are the proper delay units so that all the prestored row vectors, each of them multiplied by a value depending on the input sequence, are fed simultaneously into the adder over GF(q). Symbolically Fig. 3.5 may be represented as in Fig. 3.6.

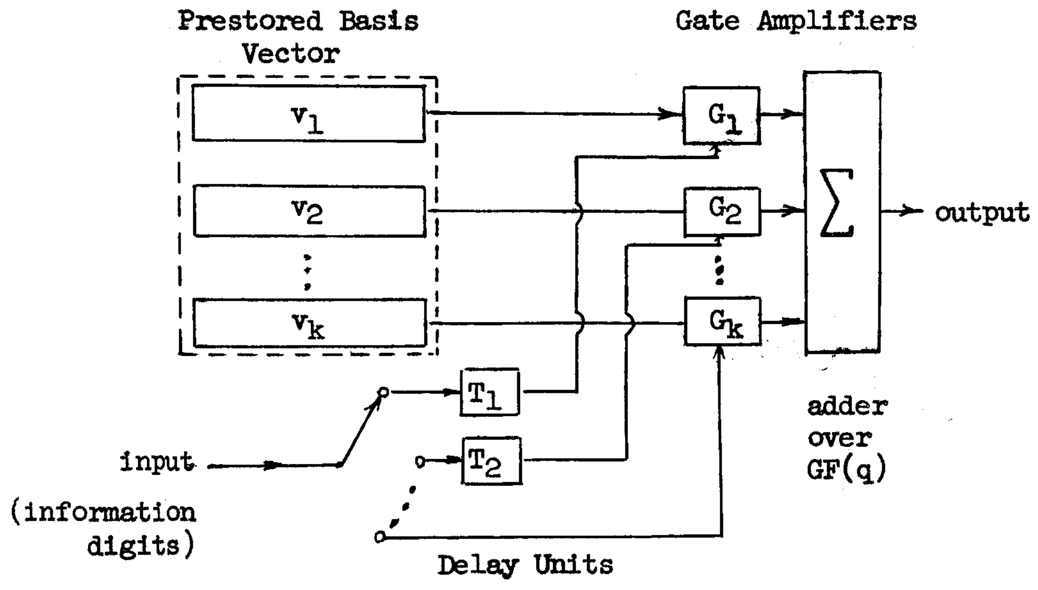


Fig. 3.5 Encoder with Specified Generator Matrix G

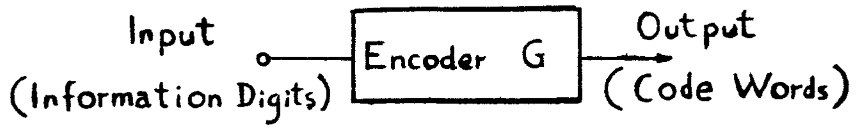


Fig. 3.6 Symbolic Representation of an Encoder with Specified Generator Matrix G

3.6 Encoding of a Product Code with Generator Matrix $G = G_1 \otimes G_2$

Fig. 3.7 is the encoder for a product code with generator matrix $G = G_1 \otimes G_2$ if the first component code is systematic. Let

$$G_1 = \begin{bmatrix} P_1 & I_{k_1} \end{bmatrix}, \tag{3.16}$$

then

$$G = G_1 \otimes G_2 = \begin{bmatrix} P_1 & I_{k_1} \end{bmatrix} \otimes G_2$$

$$= \left[P_2 \otimes G_2 \;\vdots\; I_k \otimes G_2 \right]$$

$$= \left[\begin{array}{c|cccc} & G_2 & & & \\ & & G_2 & & \\ P_2 \otimes G_2 & & & \ddots & \\ & & & & G_2 \end{array} \right]. \tag{3.17}$$

Among the first k_1 subblocks, the information digits are distributed evenly in each subblock and the encoder G_2 fills the parity-check digits for that subblock. All the first digits of subblocks $k_1 + 1$, $k_1 + 2$, $\ldots, n_1 - 1$, n_1 are the parity-check digits given by encoder G_1 with all the first digits of each of the first k_1 subblocks as information digits. The rest digits of each of the last $n_1 - k_1$ subblocks are obtained in a similar manner. Therefore, the encoder shown in Fig. 3.7 is the proper circuit for encoding. If the first component code is not systematic, some rearrangement of the ordering of the switches shown in Fig. 3.7 is necessary, but the rest of the circuit will remain unchanged.

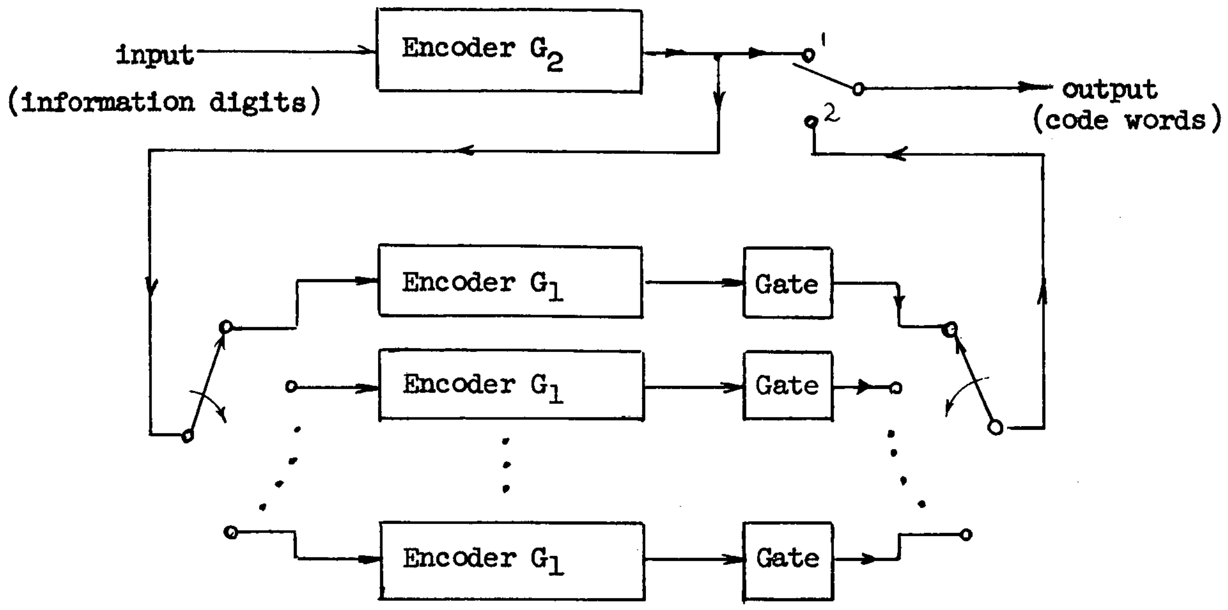


Fig. 3.7 The Encoder for a Product Code with Generator Matrix $G = G_1 \otimes G_2$

3.7 Syndrome Calculation of a Product Code with $G = G_1 \otimes G_2$

By Theorem 2.6 and Corollary 2.6.1 of Chapter II, the parity-check matrix H of the product code whose generator matrix $G = G_1 \otimes G_2$ can be expressed as either

$$H = \begin{bmatrix} H_1 \otimes \bar{H}_2 \\ H_1 \otimes H_2 \\ \bar{H}_1 \otimes H_2 \end{bmatrix} \qquad (3.18)$$

or

$$H = \begin{bmatrix} I_{n_1} \otimes H_2 \\ H_1 \otimes I_{n_2} \end{bmatrix}. \tag{3.19}$$

The row vectors of Equation (3.18) are linearly independent. Hence they form a set of basis vectors of the null space of the code space. Equation (3.19), however, gives a form which leads to a most convenient way of decoding the product code. Even though the row vectors are not all linearly independent, the union of these vectors (with overlapping of r_1r_2 rows) do generate the whole null space of the code space. The decoding scheme discussed in this and the following sections will be based entirely on Equation (3.19).

It can be easily seen that the syndrome calculation of

$$H = \begin{bmatrix} I_{n_1} \otimes H_2 \\ H_1 \otimes I_{n_2} \end{bmatrix} \tag{3.20}$$

can be accomplished by implementing two parallel but separate syndrome calculators, one for $I_{n_1} \otimes H_2$ and the other for $H_1 \otimes I_{n_2}$. The results of Section 3.3 can be applied directly here to find the syndrome. Fig. 3.8 shows the complete circuit for syndrome calculation. As in Section 3.3, the inputs to the scalar multiplier, a_1 $a_2, \ldots, a_r$, are controlled by H_1.

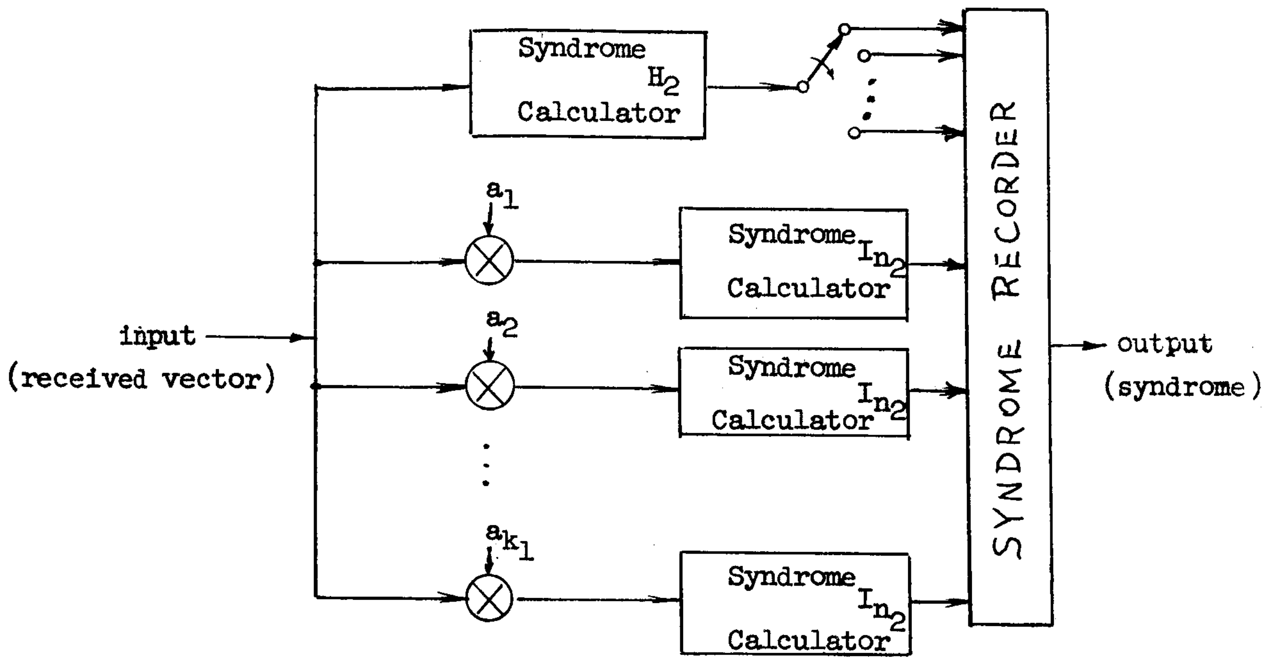


Fig. 3.8 Syndrome Calculator for the Product Code with $G = G_1 \otimes G_2$

3.8 Decoding of a Product Code $G = G_1 \otimes G_2$ After a Syndrome Calculation

Let the first component code be a t_1-error-correcting code and the second one a t_2-error-correcting code. The minimum distances of the former and the latter are assumed respectively to be $d = 2t_1 + 1$ and $d_2 = 2t_2 + 1$. It can be shown* that the minimum distance d of the product code with generator matrix $G = G_1 \otimes G_2$ is equal to the product of the minimum distances of its components.

*See Reference 2, Chapter 5, Theorem 5.3.

Thus,

$$\begin{aligned} d &= d_1 d_2 \\ &= (2t_1 + 1)(2t_2 + 1) \\ &= 4t_1t_2 + 2(t_1 + t_2) + 1. \end{aligned} \tag{3.21}$$

Therefore, the product code can either detect $4t_1t_2 + 2(t_1 + t_2)$ random errors or correct any error pattern containing $\frac{d-1}{2} = 2t_1t_2 + t_1 + t_2$ or fewer errors.

The procedure for error detection is simple. If the output of the syndrome calculator is an all-zero sequence, the received sequence is considered error-free, otherwise there are $4t_1t_2 + 2(t_1 + t_2)$ or fewer errors in the sequence. An alarm may be set off or a request for retransmission may be made if a feedback channel is available.

The error-correcting procedure is far more complicated. In the cases of non-binary codes it is necessary to employ a systematic searching technique for correcting certain error patterns. We first illustrate the decoding steps for the binary product codes, to be followed by a description of necessary modification in decoding non-binary codes.

Let G_1, H_1 be the generator and parity-check matrices, respectively, of the binary code C_1 with k_1 information digits in each code block n_1. The minimum distance of code C_1 is $d_1 = 2t_1 + 1$, hence it is capable of correcting t_1 random errors. G_2, H_2, k_2, n_2, d_2 and t_2 are defined similarly for the binary code C_2. If we form the tensor product code C whose generator matrix $G = G_1 \otimes G_2$, by Corollary 2.6.1, we have the parity-check matrix

$$H = \begin{bmatrix} I_{n_1} \otimes H_2 \\ H_1 \otimes I_{n_2} \end{bmatrix}. \tag{3.22}$$

As mentioned earlier, the minimum distance is

$$d = d_1 d_2 = 4t_1t_2 + 2(t_1 + t_2) + 1. \tag{3.23}$$

Hence the product code is capable of correcting

$$t = \frac{d-1}{2} = 2t_1t_2 + t_1 + t_2$$

random errors.

Each received code block contains n_1n_2 binary digits. It is convenient to arrange these digits in an $n_1 \times n_2$ array as follows:

$$\begin{array}{c} n_1 \\ \text{rows} \end{array} \left\{ \underbrace{\begin{array}{ccccc}
a_1 & a_2 & a_3 & \cdots & a_{n_2} \\
a_{n_2+1} & a_{n_2+2} & a_{n_2+3} & \cdots & a_{2n_2} \\
a_{2n_2+1} & a_{2n_2+2} & a_{2n_2+3} & \cdots & a_{3n_2} \\
\vdots & \vdots & \vdots & & \vdots \\
a_{(i-1)n_2+1} & a_{(i-1)n_2+2} & a_{(i-1)n_2+3} & \cdots & a_{in_2} \\
\vdots & \vdots & \vdots & & \vdots \\
a_{(n_1-1)n_2+1} & a_{(n_1-1)n_2+2} & a_{(n_1-1)n_2+3} & \cdots & a_{n_1n_2}
\end{array}}_{n_2 \text{ columns}} \right.$$

By investigating the parity-check matrix and the above array, it is clear that each row of the array should satisfy the parity-check matrix H_2 of the second component code C_2 and thus be a code word of C_2. Similarly, each column of the array should be a code word of the first component code C_1.

The detailed decoding procedure is as follows:

Step 1

Arrange the received sequence in an array as shown in the previous paragraph.

Step 2

Correct errors in each row by the parity-check matrix H_2.

Step 3

Correct errors in each column by the parity-check matrix H_1.

Step 4

Correct the remaining errors again in each row by the parity-check matrix H_2.

Step 5

Feed the corrected sequence into the syndrome calculator. If the new syndrome is an all zero sequence, go to Step 6; otherwise go to Step 7.

Step 6

Compare the received sequence with the corrected sequence. If the number of errors corrected is less than or equal to $t = 2t_1t_2 + t_1 + t_2$, the procedure is completed and the errors are assumed to have been corrected; otherwise go to Step 7.

Step 7

Repeat Steps 2 - 6, but perform column checks before row checks, then column checks again. If the test of Step 6 is satisfied, then all the errors are assumed to have been corrected; otherwise go to Step 8.

Step 8

Use H_1 and H_2 respectively as the parity-check matrices of $2t_1$ - and $2t_2$ - error detecting codes to detect all the rows and columns which are in error.

Step 9

Complement all the digits at the intersections of the erroneous rows and columns. Then repeat Steps 2 to 7. Compare the corrected sequence with the received sequence. If the number of errors is less than t, stop the procedure; otherwise go to Step 10.

Step 10

Auxiliary decoding schemes are necessary to correct all the error patterns which are not corrigible by the procedure of the previous 9 steps. These should be designed according to certain properties of the code. These error patterns which are, in general, a very small fraction of the total number of the corrigible error patterns.

It is convenient to explain the reasons behind the above 10 steps by an example. Let $t_1 = t_2 = 2$, then $t = 2t_1t_2 + t_1 + t_2 = 12$. The following cases are handled separately.

(i) If the number of errors in each row and each column is always less than 3, then either the row checks or the column checks are sufficient to correct all the errors. Hence this is a trivial case.

(ii) The error pattern of 12 errors is of the form:

$$\begin{array}{ccccc}
 & \times & \times & \times & \times \\
 & \times & & \times & \\
 & \times & \times & & \\
\times & \times & \times & & \\
 & \times & & &
\end{array}$$

($t_1 = 2$, or fewer rows contain more than $t_2 = 2$ errors).
Applying row checks, we may "transform" the error pattern to the form shown in the following:

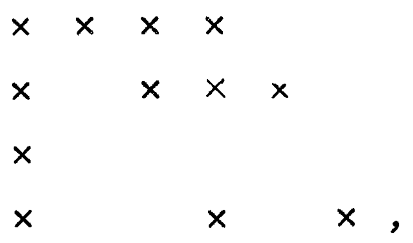
$$\begin{array}{ccccc}
\overset{\times}{\triangle} & \times & \times & \times & \times \\
 & \triangle & & \triangle & \\
 & \triangle & & \triangle & \\
\times & \times & \times & \overset{\times}{\triangle} & \overset{\times}{\triangle} \\
 & \triangle & & &
\end{array}$$

where $\triangle$ denotes the position where an error is corrected by the row checks, and $\overset{\times}{\triangle}$ denotes the position where an extra error is introduced by the wrong correction based on the row checks.
Applying column checks next, we obtain

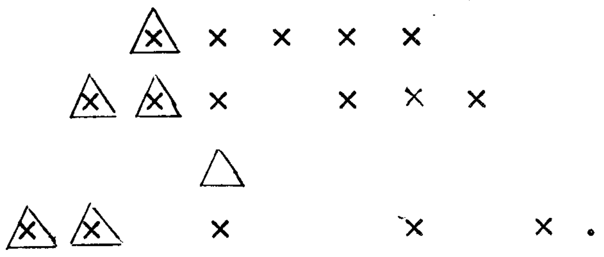
$$\begin{array}{ccccc}
\boxed{\triangle} & \square & \square & \square & \square \\
 & \triangle & & \triangle & \\
 & \triangle & \triangle & & \\
\square & \square & \square & \square & \square \\
 & \triangle & & &
\end{array}$$

which completes the correction of all 12 errors.

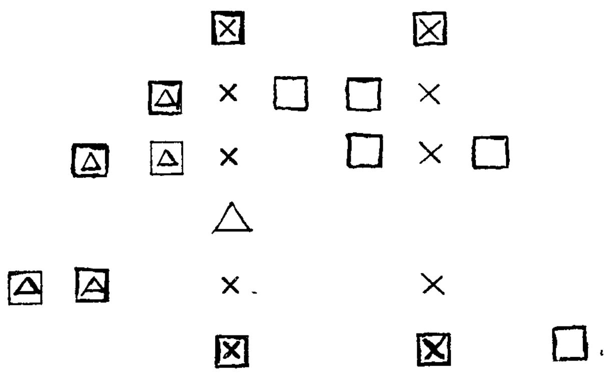
(iii) The error pattern of 12 errors is of the form



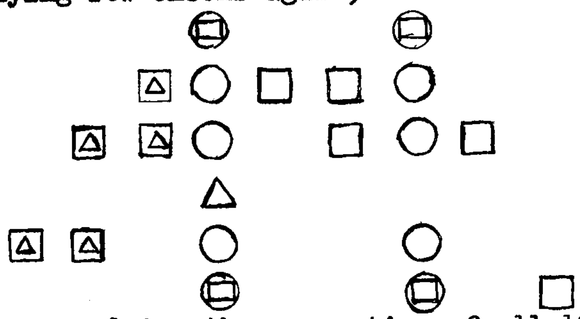
(t_2 = 2, or fewer columns contain more than t_1 = 2 errors). Applying row checks, we have



Applying column checks, we have



Applying row checks again, we obtain



which completes the correction of all 12 errors.

(iv) The error pattern is of the form

$$\begin{array}{ccccc}
 & \times & \times & \times & \times \\
\times & & \times & \times & \\
\times & \times & & & \times \\
 & \times & & & \times
\end{array}$$

($t_2 = 2$, or fewer columns contain more than $t_1 = 2$ errors). Applying row checks, we have

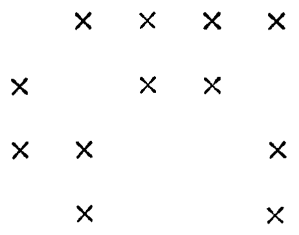
$$\begin{array}{ccccc}
\overset{\triangle}{\times} & \times & \times & \times & \times \\
\times & \overset{\triangle}{\times} & \times & \times & \overset{\triangle}{\times} \\
\times & \times & \overset{\triangle}{\times} & \overset{\triangle}{\times} & \times \\
 & \triangle & & & \triangle
\end{array} .$$

Applying column checks, we have

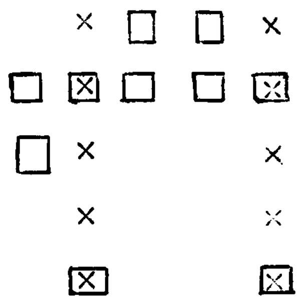
$$\begin{array}{ccccc}
\boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\
\boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} & \boxed{\times} \\
\overset{\triangle}{\times} & \times & \times & \times & \times \\
\times & \overset{\triangle}{\times} & \times & \times & \overset{\triangle}{\times} \\
\times & \times & \overset{\triangle}{\times} & \overset{\triangle}{\times} & \times \\
 & \triangle & & & \triangle
\end{array} .$$

We then count the "corrected" errors. There are 16 in number, hence we know that this is not the right error pattern. The correcting procedure is started all over again with column checks first as:

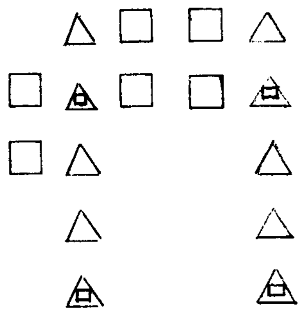
Original error pattern



Column checks

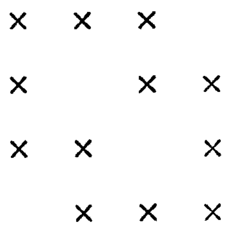


Row checks



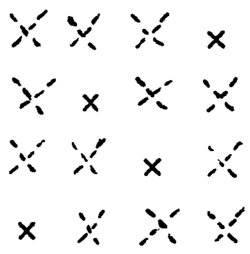
which succeeds in correcting all 12 errors.

(v) The error pattern (of 12 errors) is of the form



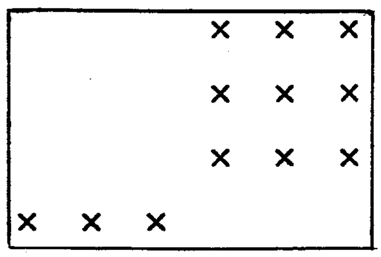
Apply either row checks first or column checks first.

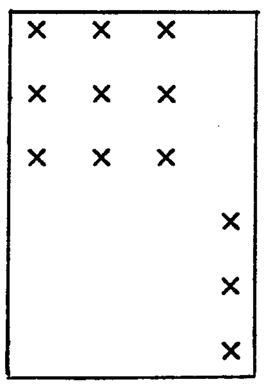
The "corrected" sequence will differ from the original received sequence for more than 12 positions. Therefore, we go to Step 8 and detect all the rows and columns which are in error. By counting them, we know that there are four rows and four columns in error. According to Step 9, we complement all the digits at the intersections of the erroneous rows and columns. The new error pattern becomes



where denotes the position which was in error before complementing. It is evident that the complemented errors can be easily corrected by either row or column checks.

(iv) The only error patterns which cannot be corrected by Steps 2 through 9, are some permutations of columns and/or rows of the following two patterns:

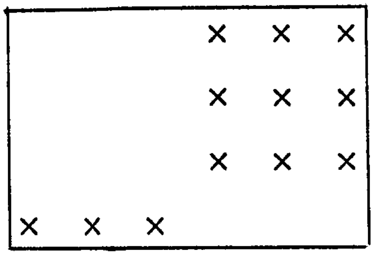


 .

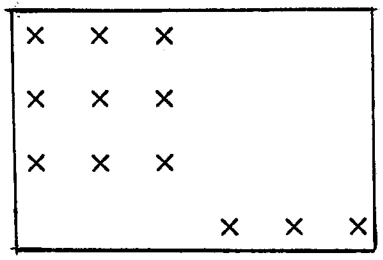
The pattern on the right is obtained from that on the left by a 90° rotation. Hence, only the left-hand side pattern needs to be considered in the following discussion.

It should be noted that, for this error pattern, the complements of the digits at the intersections of erroneous rows and columns form a mirror image of the original error pattern as follows:

Original Pattern

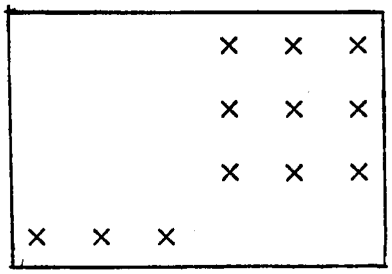


Complemented Pattern

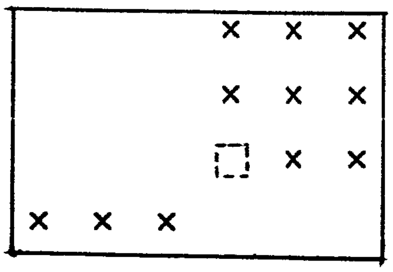


The method of complimentation fails due to the fact that there are an equal number of correct and erroneous digits at the intersections of those rows and columns. Therefore, it is necessary to break this "balanced" situation of correct and incorrect positions. One way of doing it is to change an odd number of aribtrarily chosen digits among these intersections. Then the number of correct digits and that of erroneous digits will differ at least by one. Let us take the smallest odd number, 1, and proceed with the correction procedure as follows:

The original pattern is repeated here for reference.

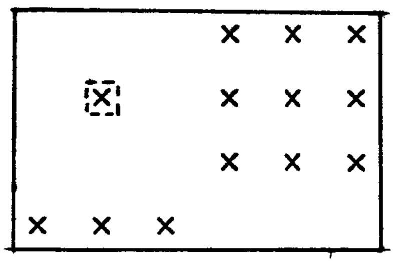


If the assumed single error is among one of the erroneous digits, then the new pattern is

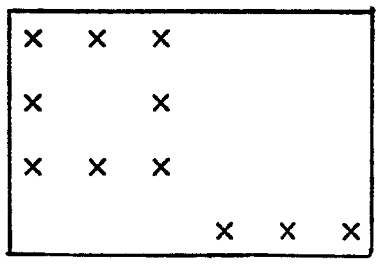


where $\boxed{}$ denotes the original erroneous position which was corrected by the trial-and-error method. The above pattern can be easily corrected by applying column checks first, followed by the row checks.

If the assumed single error is not among one of the erroneous digits, then an additional error is introduced, the new pattern becomes



where $\boxed{X}$ is the newly introduced error by the method of trial-and-error. Because of this additional error, any row checks and column checks cannot completely correct the error pattern, no matter which one is applied first. We then complement the whole matrix and obtain the new pattern as follows:

 .

This error pattern can be corrected easily by applying column checks followed by row checks.

At the end of each correcting procedure we should count the number of positions changed to vindicate whether the corrected "errors" are real errors. Suppose there are e errors in a received sequence and $e \leq t$. It is well known that the minimum distance d of a t-error-correcting code is given by $d \geq 2t + 1$. Hence in dealing with a received sequence with e errors, two possible situation may occur: we may either correct all the errors, or add some other erroneous digits together with the original received errors to form a code word which satisfies both the row checks and the column checks. In order to form a code word which is d or more digits distant from the true code word, a weight of no less than $d-e(\geq t+1)$ must be added to the original error pattern. Therefore, at the end of each correction procedure, if it is found that the number of errors corrected is less than t, the corrected code word is the true code word. An alternate decoding scheme should be used if this number exceeds t.

In Steps 8 and 9, we have assumed that among the possibly erroneous positions, either more than half are in error, or the pattern of the correct positions is such that if it were an error pattern it could be easily corrected by the row and column checks. Hence, after detecting the error locations, we take the complement of these possibly erroneous digits and proceed to correct all the errors in the complemented pattern of the original errors.

The auxiliary decoding scheme cited in Step 10 and the illustration in the example varies from code to code depending upon some special properties of the codes and the error patterns.

It should be noted that Steps 8 and 9 do not apply to non-binary codes. Complementing in a non-binary field will not, in general, eliminate the original error pattern but will introduce new errors. Hence either a trial-and-error technique should be used, or a new auxiliary decoding scheme should be adopted for the non-binary codes.

The main advantage of the tensor product code is that one may employ the error-correction capabilities of the component codes to correct the errors in the product space. The design involved in the repetitive procedure of the component code is usually simpler than that required to correct a larger number of errors by means of a single code. Take as an example the product code C (49,16), which is a tensor product of two cyclic (7,4) codes C_1 and C_2. The (49,16) code is capable of correcting 4 errors. Were it not a product code, a single 4 error-correction code would require solving 4 simultaneous equations in a binary extension field. Now, as a product code, it requires only the application of single error-correcting procedure to rows and columns of 7 digits each which does not involve the calculation in any of the extension fields. A saving in the decoding procedure results.

3.9 Encoding of a Product Code with Parity-Check Matrix $H = [H_{1q} \otimes H_{2q}]_p$

Since $H = [H_{1q} \otimes H_{2q}]_p$ is a p-ary representation of $[H_{1q} \otimes H_{2q}]$; the syndrome obtained from H should be the syndrome derivable from $[H_{1q} \otimes H_{2q}]$ by proper translation from GF(q) to GF(p).

Because of the complexity of the decoding procedure in higher extension fields, we shall assume that H_{2q} is a $1 \times n_2$ matrix. The encoding procedure can be described as follows:

(i) Fill the checking digit positions with zeros.

(ii) Divide the entire sequence, including the information digits and the all-zero checking digits, into n_1 subblocks with n_2 digits in each subblocks.

(iii) Separately feed each subblock into a syndrome calculator with parity-check matrix $[H_{2q}]_p$.

(iv) Interpret the syndrome of each subblock as a q-ary digit. Then feed these n_1 q-ary digits into a syndrome calculator with parity-check matrix H_{1q}.

(v) Interpret the syndrome output of the calculator with matrix H_{1q} in terms of its p-ary representation. Then add the p-ary representation of each q-ary digit to its corresponding positions which were filled with zeros before.

The encoding circuit is shown in Fig. 3.9.

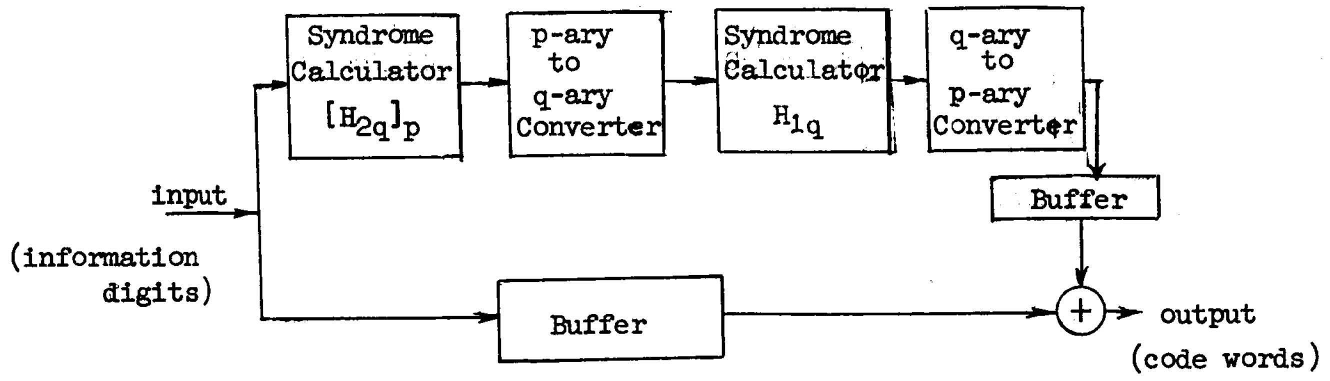


Fig. 3.9 Encoder for the Product Code with $H = [H_{1q} \otimes H_{2q}]_p$

3.10 Decoding of a Product Code with Parity-Check Matrix $H = [H_{1q} \otimes H_{2q}]_p$

The syndrome calculator of a product code is very similar to the encoder of the code. Fig. 3.10 is a block diagram of the syndrome calculator

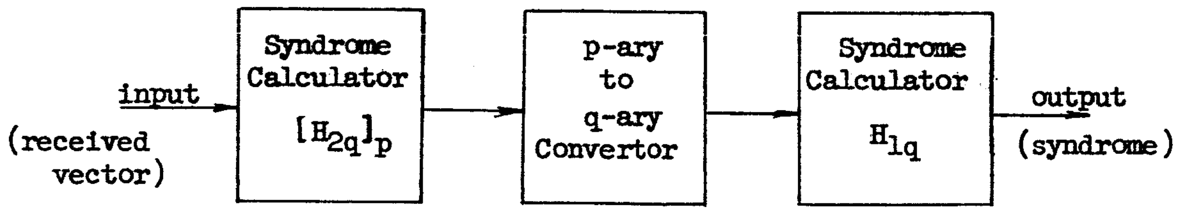


Fig. 3.10 Syndrome Calculator for the Product Code with $H = [H_{1q} \otimes H_{2q}]_p$

The output of the syndrome calculator should be given as vectors with components in GF(q). With this syndrome, we determine the location of erroneous subblocks. In locating the errors, we go through the error correcting procedure for the q-ary component code with parity-check matrix H_{1q} up to the point of determining the error location. It is not necessary to evaluate the exact error pattern within each erroneous subblock, since the product code is employed as an error-locating code.

3.11 Remarks

Let us first compare the decoding procedure of a product code whose parity-check matrix is $H = H_1 \otimes H_2$ with that of a product code specified by the parity-check matrix $H' = [H_{1q} \otimes H_{2q}]_p$ in the following table:

Parity-Check Matrix	$H_1 \otimes H_2$	$[H_{1q} \otimes H_{2q}]_p$
Number of rows of second parity-check matrix	p	1 in q-ary expanded to p in p-ary
Data transmission rate	Lower	Higher
Total number of code words	Smaller	Larger
Fields translation	No	Yes
Implementation	Less complex	More complex

The choice of produce codes with $H = H_1 \otimes H_2$ and $H = [H_{1q} \otimes H_{2q}]_p$ depends on the complexity of circuits allowed and the number of code words required. For the same code word length and same redundancy, one has to pay the price of employing more complicated circuits in order to have a larger code word space.

In the previous sections, we have considered the product code whose parity-check matrix is the tensor product of the check matrices of two component codes as an error-locating code. However, the code can also achieve some other functions. Let the first component code have a minimum distance of $d_1 = 2t_1 + 1$. It is capable of correcting t_1 or fewer random or independent errors. Let the minimum distance of the second component code be $d_2 = 2t_2 + 1$. Then the product code is capable of performing any one of the following functions.

(i) Detecting $\min(2t_1, 2t_2)$ random errors.

(ii) Correcting $\min(t_1, t_2)$ random errors.

(iii) Detecting $2t_1$ corrupted subblocks, each of which containing no more than $2t_2$ errors.

(iv) Locating t_1 corrupted subblocks, each of which containing no more than $2t_2$ errors.

(v) Correcting t_1 corrupted subblocks, each of which contains no more than t_2 errors.

It should be noted that the product code of the type $H = H_1 \otimes H_2$ as well as that of the type $H' = [H_{1q} \otimes H_{2q}]_p$ can perform any one of the above 5 functions.

The bounds given in (i) and (ii) are due to the fact that the minimum distance d of this product code $H = H_1 \otimes H_2$ is $\min(d_1, d_2)$, a property to be further examined in Chapter 6. It is evident that this kind of product code is not particularly suited for random-error correction or random-error detection because of its relatively small minimum distance. It seems that the minimum distance is not always the best criterion for judging the utility of a code because there are occasions when the knowledge about the relation between the error patterns and the subblock structures is important.

The product code whose generator matrix is the tensor product of the generator matrices of its component codes has also an interesting alternative error-control capability. Since the parity-check matrix of this product code is given as

$$H = \begin{bmatrix} H_1 \otimes I_{n_2} \\ I_{n_1} \otimes H_2 \end{bmatrix},$$

the upper submatrix $H_1 \otimes I_{n_2}$ can be considered as the parity-check matrix which is capable of locating t_1 subblocks (assuming H_1 is the parity-check matrix of a t_1-error-correcting code) with each of them containing no more than n_2-1 errors. This is due to the fact that I_{n_2} is the parity-check matrix of a code whose minimum distance is n_2. The lower submatrix $I_{n_1} \otimes H_2$ is capable of locating up to $\frac{n_1-1}{2}$ corrupted subblocks, each containing no more than $2t_2$ errors (assuming H_2 is the parity-check matrix of a $2t_2$-error-detecting code). Therefore, we can employ this product code to locate errors distributed in the following two ways:

(i) Errors are clustered in no more than t_1 random subblocks with at least one correct digit in each.

(ii) Errors are distributed in $\frac{n_1-1}{2}$ subblocks, each containing no more than $2t_2$ errors.

$H_1 \otimes I_{n_2}$ and $I_{n_1} \otimes H_2$ can be considered as the parity-check matrices of the codes whose functions are listed under the discussion of error-locating codes of this section. A meaningful combination of them is that the product code is capable of

(i) correcting t_1 subblocks containing $\frac{n_2-1}{2}$ or fewer errors,

(ii) locating $\frac{n_1-1}{2}$ subblocks containing $2t_2$ or fewer errors.

Another useful application of the product code specified by $G = G_1 \otimes G_2$ over GF(q) is to correct, by simple procedures described in Section 3.8, Step 1 - 8, all the following four kinds of error patterns:

(i) fewer than t_2 + 1 errors in each row,

(ii) fewer than t_1 + 1 errors in each column,

(iii) no more than t_1 rows contain more than t_2 errors,

(iv) no more than t_2 columns contain more than t_1 errors.

The receiver will ask for retransmission or locate the errors whenever the error patterns do not belong to any of the above four categories. It should be noted that the above four kinds of error patterns occupy a very large fraction of total corrigible error patterns.

CHAPTER 4

IMPLEMENTING PRODUCT CODES WITH DUAL CYCLIC COMPONENT CODES

The ease with which cyclic codes can be implemented and the results of the last chapter showing that the encoding and decoding of a product code can be decomposed into those of its component codes lead to the conclusion that product codes with cyclic component codes should be relatively easy to realize. This chapter is devoted to studying the implementation of the encoders and syndrome calculators of the product codes with cyclic component codes. A cyclic decoding scheme of the cyclic codes after obtaining the syndromes will be discussed in Chapter 5.

First, the implementation of cyclic codes by shift-registers is reviewed, then dual cyclic codes are studied. Next, the encoding and syndrome calculations of product codes with cyclic component codes are shown. Finally, the problem of implementing product codes with dual cyclic component codes is investigated. The high degree of flexibility in redundancy possessed by the product codes with dual component codes is very attractive in applications such as space-vehicle-to-ground communications.

4.1 Cyclic Codes[2]

A subspace V of n-tuples over GF(q) is called a cyclic subspace or a cyclic code if for each vector $\underline{f} = (a_0, a_1, \ldots, a_{n-1})$ in V, the vector $\underline{f}' = (a_{n-1}, a_0, a_1, \ldots, a_{n-2})$ obtained by shifting the components of $\underline{f}$ cyclically one unit to the right is also in V. Corresponding to each n-tuple $(a_0, a_1, \ldots, a_{n-1})$ there is a polynomial of degree less than n, $f(x) \equiv a_0 + a_1x + \ldots + a_{n-1}x^{n-1}$, mod x^n-1.

Let $g(x)$ be a monic polynomial* of degree r which divides x^n-1. Then an n-tuple is a code word of the cyclic code V if and only if its corresponding polynomial $f(x)$ is divisible by $g(x)$. A cyclic code is, therefore, specified completely by a polynomial $g(x)$ that divides x^n-1, and it is said that the cyclic code is generated by $g(x)$. Alternately, the code can be specified by its recursion polynomial $h(x) = (x^n-1)/g(x)$.

4.2 Encoding for a Systematic (n,k) Cyclic Code**

Encoding for the systematic (n,k) cyclic code generated by $g(x)$, a polynomial of degree n-k, can be accomplished by using a device called a shift-register generator as shown in Fig. 4.1, with connections corresponding to

$$h(x) = (x^n-1)/g(x) = h_0 + h_1x + h_2x^2 + \dots + h_{k-2}x^{k-2} + h_{k-1}x^{k-1} + x^k .$$

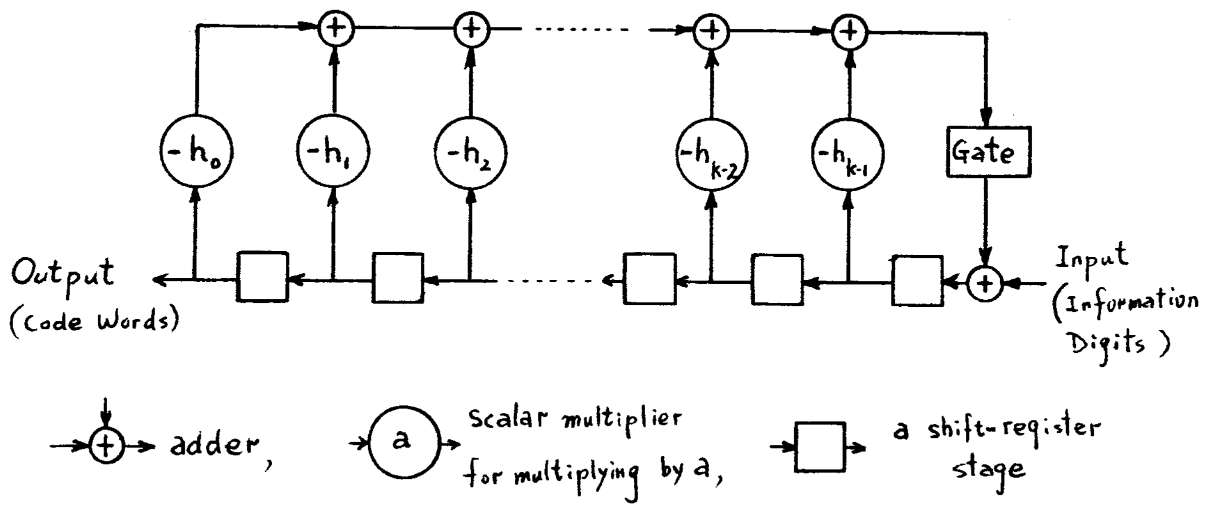


Fig. 4.1 A Shift-Register Generator for Encoding the (n,k) Code Employing k Shift-Register Stages

*A polynomial $f(x)$ is called monic if the coefficient of the highest power of x is 1.

**Peterson[2] Chapter 7.

The gate is open during the period of feeding in the k information digits. The device is made to shift n times immediately after all the information digits are fed in. The gate is closed during the first n-k shifts, then opened again for the last k shifts in order to clear out all the shift-registers. The first k symbols to come out will be the information symbols, and the last n-k symbols will be a set of check symbols that make the whole n-tuple a code vector. Fig. 4.2 is a modification of Fig. 4.1 in which the constant delay of k digits is eliminated.

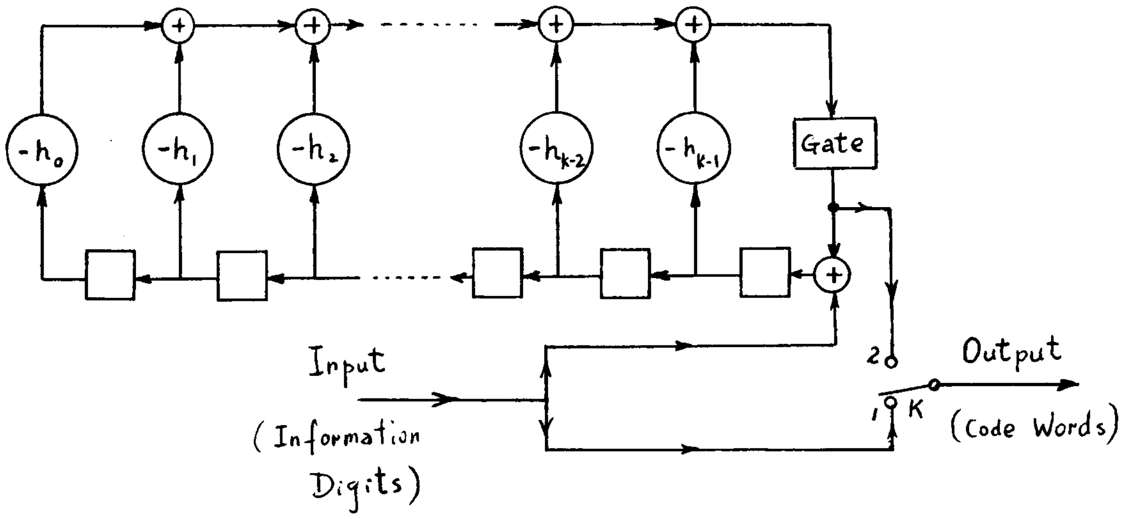


Fig. 4.2 An Encoder for the *n,k) Cyclic Code Employing k Shift-Register Stages

The gate is open for the first k shifts, and closed for the last n-k shifts. The switch k is at position 1 for the first k shifts then set to position 2 for the last n-k checking symbols.

Example

For the (7,4) binary code, $g(x) = 1 + x + x^3$ and $h(x) = (x^7+1)/g(x) = 1 + x + x^2 + x^4$. The shift-register shown in Fig. 4.3 can be used for encoding.

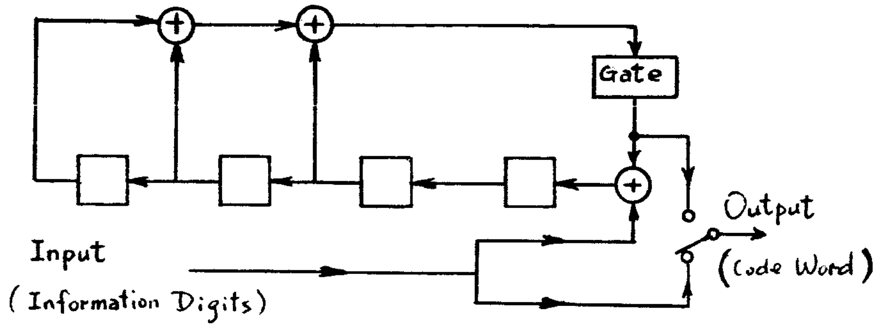


Fig. 4.3 Shift-Register for Encoding the (7,4) Code

Let $f_0(x)$ be a polynomial in which the k coefficients of terms involving $x^{n-1}, x^{n-2}, \ldots, x^{n-k}$ are arbitrary information symbols, and the coefficients of terms of degree less than n-k are 0. This corresponds to a vector for which the first n-k components are 0. The last k components are arbitrary information symbols. Then by the division algorithm:

$$f_0(x) = g(x)\, q(x) + r(x), \tag{4.1}$$

where r(x) has degree less than n-k, the degree of g(x). Then

$$f_0(x) - r(x) = g(x)\, q(x). \tag{4.2}$$

Hence the n-tuple corresponding to $f_0(x) - r(x)$ is a code vector for which the first n-k components are checking digits, the last k components arbitrary information symbols. Therefore, the encoding for an (n,k) code can also be achieved by a divider shown in Fig. 4.4, with connections corresponding to $g(x) = g_0 + g_1x + \cdots + g_{n-k-1}x^{n-k-1} + x^{n-k}$.

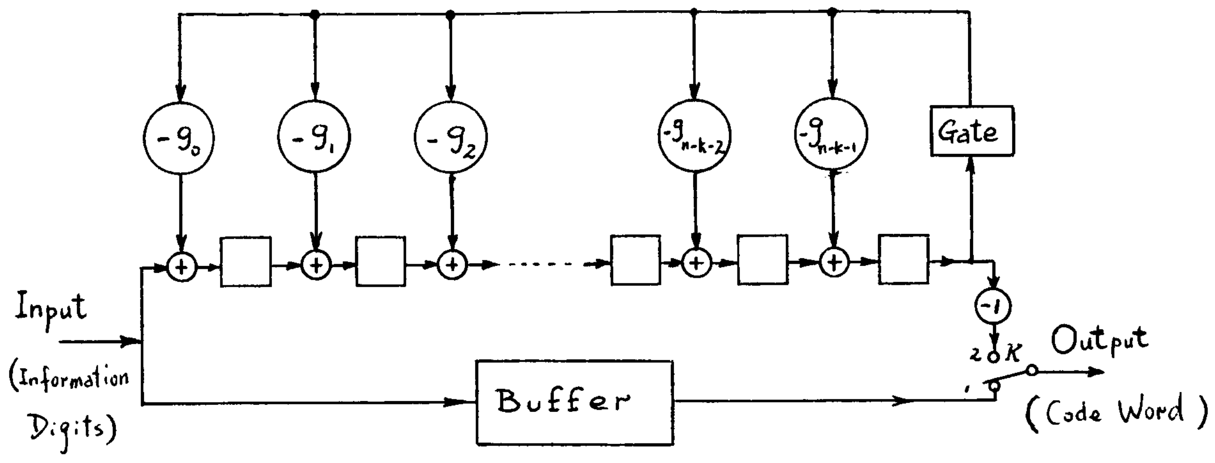


Fig. 4.4 A Divider as the Encoder for a Cyclic (n,k) Code Employing n-k Shift-Registers

The input sequence is fed into both the divider and the buffer which is used to store the sequence for a duration of n-k units of time. The gate is originally open and closes when all the information digits are fed into the divider. It opens again n-k shifts after the last information digit is sent into the divider so that all the storage units in the divider will be cleared out and they be ready for a new information sequence. The output takes information digits from the buffer and checking digits from the divider. The buffer of Fig. 4.4 can be removed by multiplying the information sequence by the factor x^{n-k} before feeding it into the divider. This can be achieved by introducing the information sequence at the output of the $(n-k)^{th}$ shift register, as shown in Fig. 4.5.

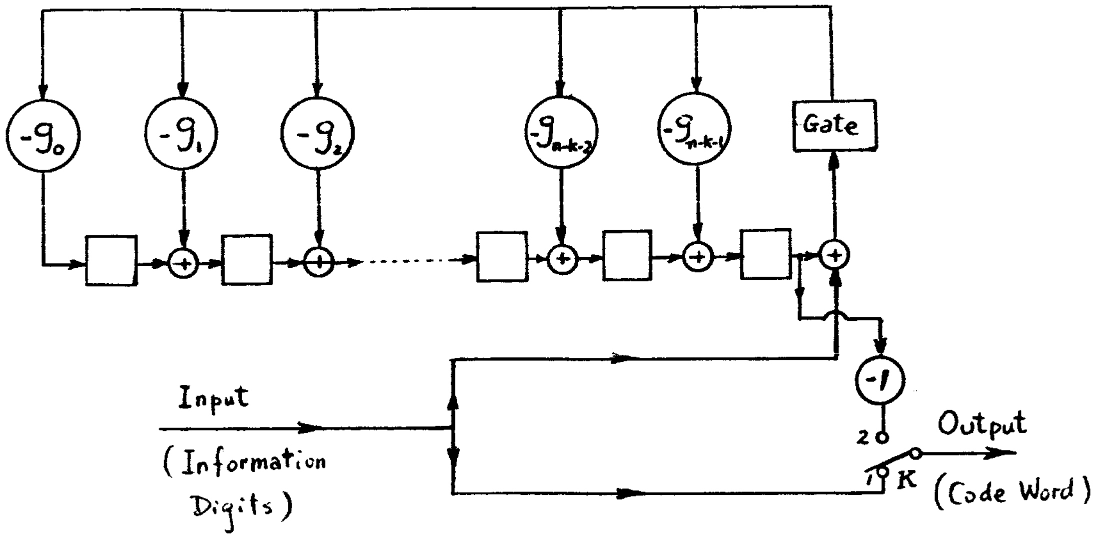


Fig. 4.5 A Divider as the Encoder for a Cyclic (n,k) Code Without Buffer Stage

The gate is closed during the feeding in of information digits. It is opened immediately after all the information digits have been fed into the divider. In this chapter, for the purpose of faciliating the implementation of dual codes, an equivalent circuit of Fig. 4.5 is used as the encoder for the (n,k) code which is shown in Fig. 4.6.

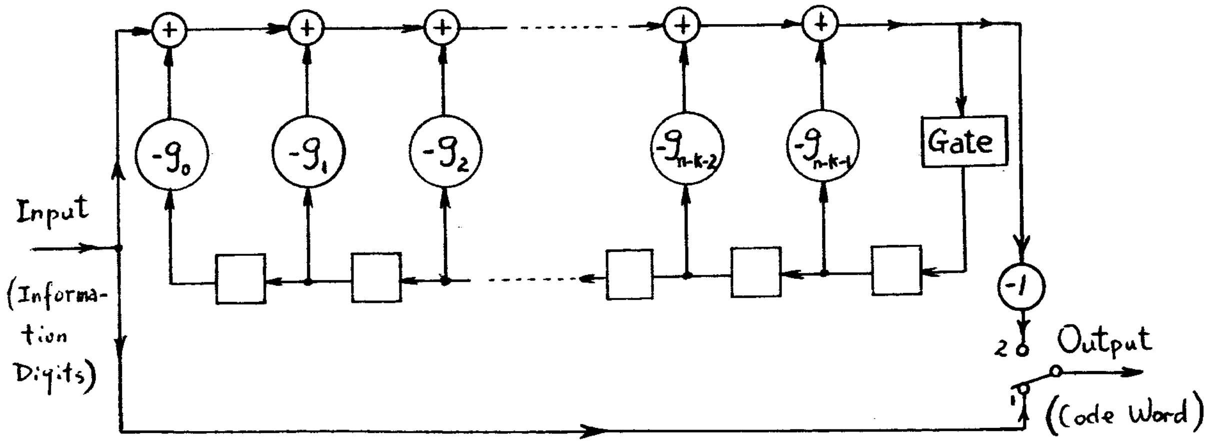


Fig. 4.6 An Equivalent Circuit for Fig. 4.5

The gate is closed during the first k digits and open for the last n-k digits.

Example

For the (7,4) binary cyclic code, $g(x) = 1 + x + x^3$, the encoding circuit styled after Fig. 4.6 is shown in Fig. 4.7.

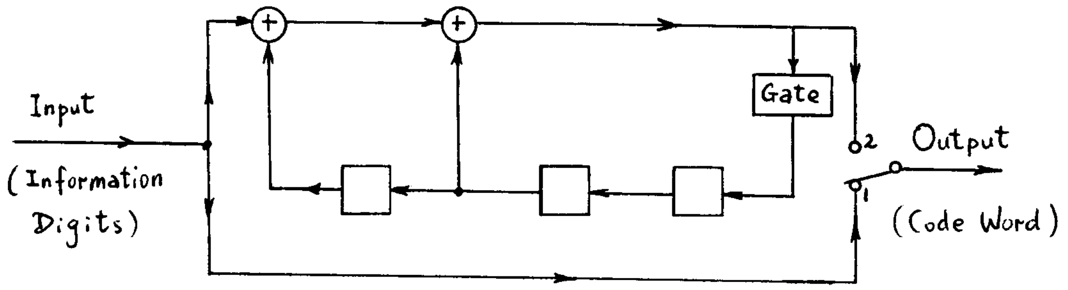


Fig. 4.7 An Encoder for (7,4) Code

4.3 Syndrome Calculating for a Systematic (n,k) Cyclic Code

Let $\underline{r} = (r_0, r_1, r_2, \ldots, r_{n-1})$ be a received vector. The corresponding polynomial r(x) for $\underline{r}$ is

$$r(x) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \ldots + r_{n-k}x^{n-k} + \ldots + r_2x^2 + r_1x + r_0 .$$

Let

$$f_0(x) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \ldots + r_{n-k}x^{n-k}$$

be the polynomial whose first k coefficients are identical to those of r(x) and whose last n-k coefficients are zeros. Then

$$f_0(x) = q(x)\ g(x) + r'(x), \tag{4.3}$$

where g(x) is the generator polynomial of degree n-k for an (n,k) cyclic code, $q(x)$ is the quotient polynomial, and $r'(x)$ is the remainder whose degree is $< n-k$. It can be easily seen that

$$f(x) = f_0(x) - r'(x) \tag{4.4}$$

is a polynomial whose first k coefficients are identical to $r(x)$, thus the corresponding vector $\underline{f}$ of $f(x)$ is a code word. It can also be shown that the syndrome $\underline{s}$ for the received vector $\underline{r}$ is given as

$$\underline{\underline{s}\ 00 \ldots 0} = \underline{r} - \underline{f} \tag{4.5}$$

or in polynomial form

$$s(x) = r(x) - f(x). \tag{4.6}$$

Hence the encoder shown in Fig. 4.1 can be modified to be the syndrome calculator as shown in Fig. 4.8.

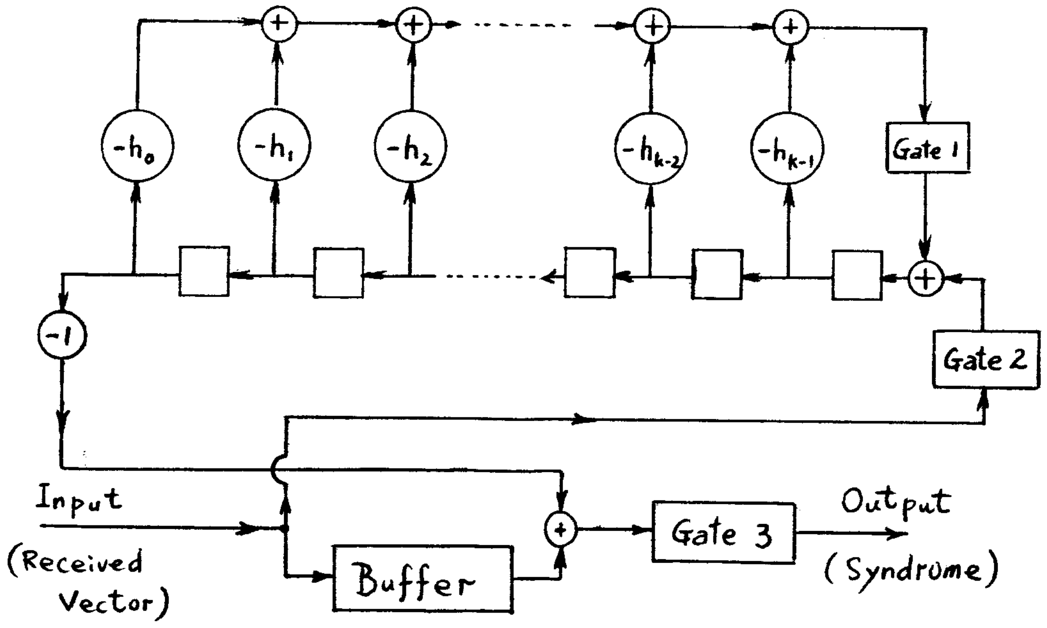


Fig. 4.8 A Syndrome Calculator for an (n,k) Cyclic Code, a Modification of Fig. 4.1

Gate 1 is opened for the first k digits, closed for the next n-k digits, then opened again for the last k digits. Gate 2 is closed for the first k digits and opened for the rest of the n digits. The buffer serves to delay the input for k digits. The first k digits of the input sequence $\underline{r}$ are fed into the encoder to generate the code word $\underline{f}$. This is then subtracted from the input sequence $\underline{r}$ to form $\underline{s}$. Here the first k zero components are deleted by the action of gate 3.

Since the first k digits of r(x) and those of f(x) are identical, we can ignore them completely without going through the trouble of subtracting the latter from the former. Hence Fig. 4.8 can be simplified further as shown in Fig. 4.9.

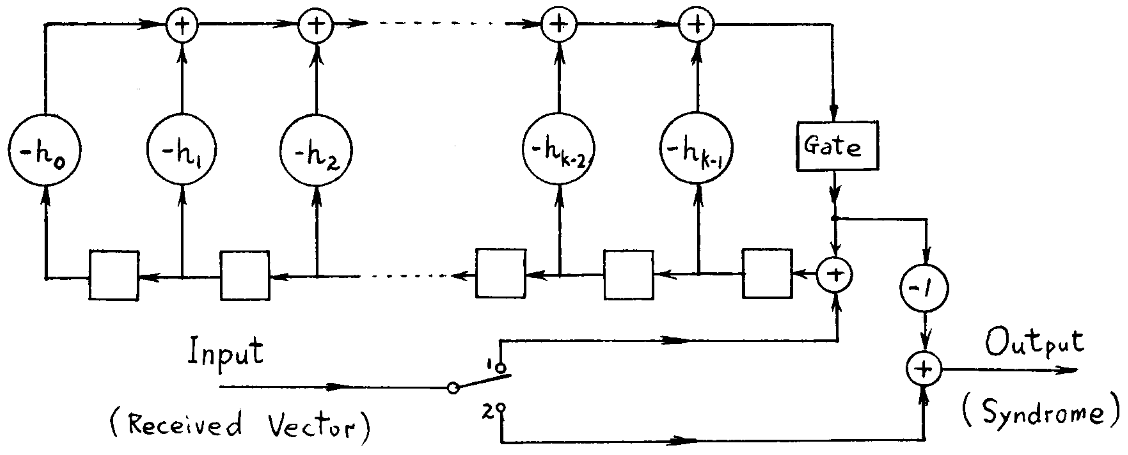


Fig. 4.9 An Equivalent Circuit for Fig. 4.8

The gate in Fig. 4.9 is the same as Gate 1 in Fig. 4.8. Switch K is at position 1 for the first k digits and at position 2 for the next n-k digits, and no buffer is required.

Let the polynomial r(x) corresponding to the received sequence $\llcorner r \lrcorner$ be split as follows:

$$r(x) = f_0(x) + r_0(x), \tag{4.7}$$

where

$$f_0(x) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \ldots + r_{n-k}x^{n-k} , \tag{4.8}$$

and

$$r_0(x) = r_{n-k-1}x^{n-k-1} + r_{n-k-2}x^{n-k-2} + \ldots + r_1 x + r_0 . \tag{4.9}$$

Then

$$\frac{r(x)}{g(x)} = \frac{f_0(x)}{g(x)} + \frac{r_0(x)}{g(x)} , \tag{4.10}$$

where the degree $r_0(x) <$ degree $g(x)$. (4.11)

Let

$$f_0(x) = q_0(x)\ g(x) + r'(x). \tag{4.12}$$

It can be shown that the syndrome is

$$s(x) = r'(x) + r_0(x). \tag{4.13}$$

Therefore, Fig. 4.6 can be modified to be the syndrome calculator as shown in Fig. 4.10. Switch K is at position 1 for the first k digit, then is shifted to position 2 for the last n-k digits. Gate 1 is closed for the first k digits only, and Gate 2 is closed for the last n-k digits only.

Since the encoder and the syndrome calculator of a systematic (n,k) cyclic code can be implemented by shift-registers with connections corresponding to either its generator polynomial g(x) or its recursion polynomial h(x), it is possible to implement the encoding and syndrome calculation for a cyclic code by a circuit and its dual. This will be the subject of discussion in the next section.

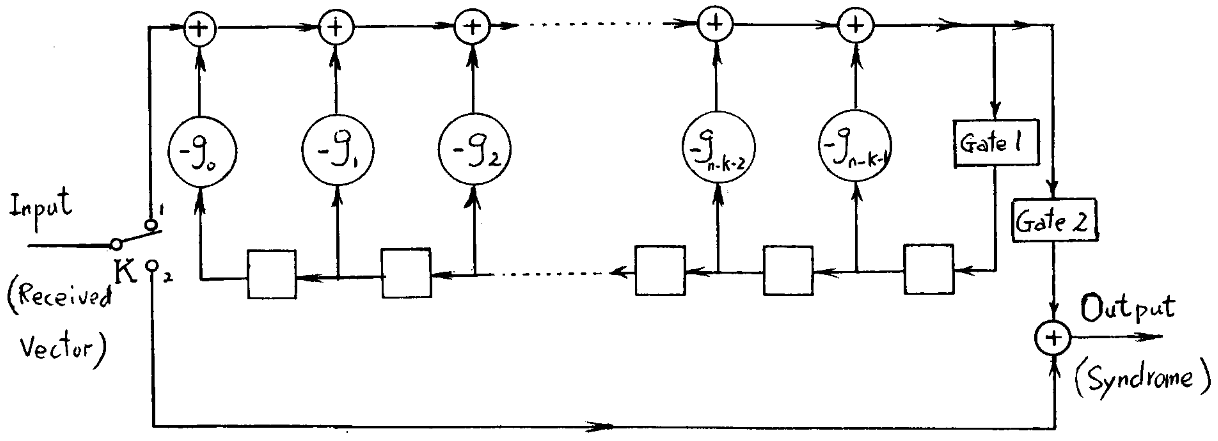


Fig. 4.10 Syndrome Calculator Obtained by Modifying Fig. 4.6

4.4 Cyclic Dual Codes as Variable Redundancy Codes[*]

Let g(x) of degree n-k be the generator polynomial of an (n,k) cyclic code C and $h(x) = (x^n-1)/g(x)$ be the recursion polynomial. Then the code C' with generator polynomial h(x) and recursion polynomial g(x) is called the dual code of code C. Or code C and code C' are dual to each other.

By combining Figs. 4.1, 4.6, 4.9, and 4.10, we are able to implement a circuit which is an (n,k) cyclic code at one time and the dual of the (n,k) code or a (n,n-k) code at another time if a mode-chainging signal is given. Since the redundancy of the two modes is different we may use dual codes as variable redundancy codes. Fig. 4.11 shows the encoding and syndrome-calculating circuit for the variable redundancy code. The timing for the switches is listed as follows:

[*]See Tang, Reference 4.

Mode 1: The code is generated by $g(x)$

Encoder

Switch \ Time Position	$0^+ \to kT^-$	$kT^+ \to nT^-$
K_1	a	a
K_2	1	2
K_3	1	2
K_4	a	a

Syndrome Calculator

Switch \ Time Position	$0^+ + t_D \to kT^- + t_D$	$kT^+ + t_D \to nT^- + t_D$
K_6	a	a
K_7	1	2
K_9	1	2
K_{10}	2	1
K_{11}	a	a

where T is the digit duration,

iT^+ is the beginning time of the i^{th} digit,

iT^- is the instant just before the beginning time of the i^{th} digit, and t_D is the delay time required to transmit the signal from the encoder to the syndrome calculator.

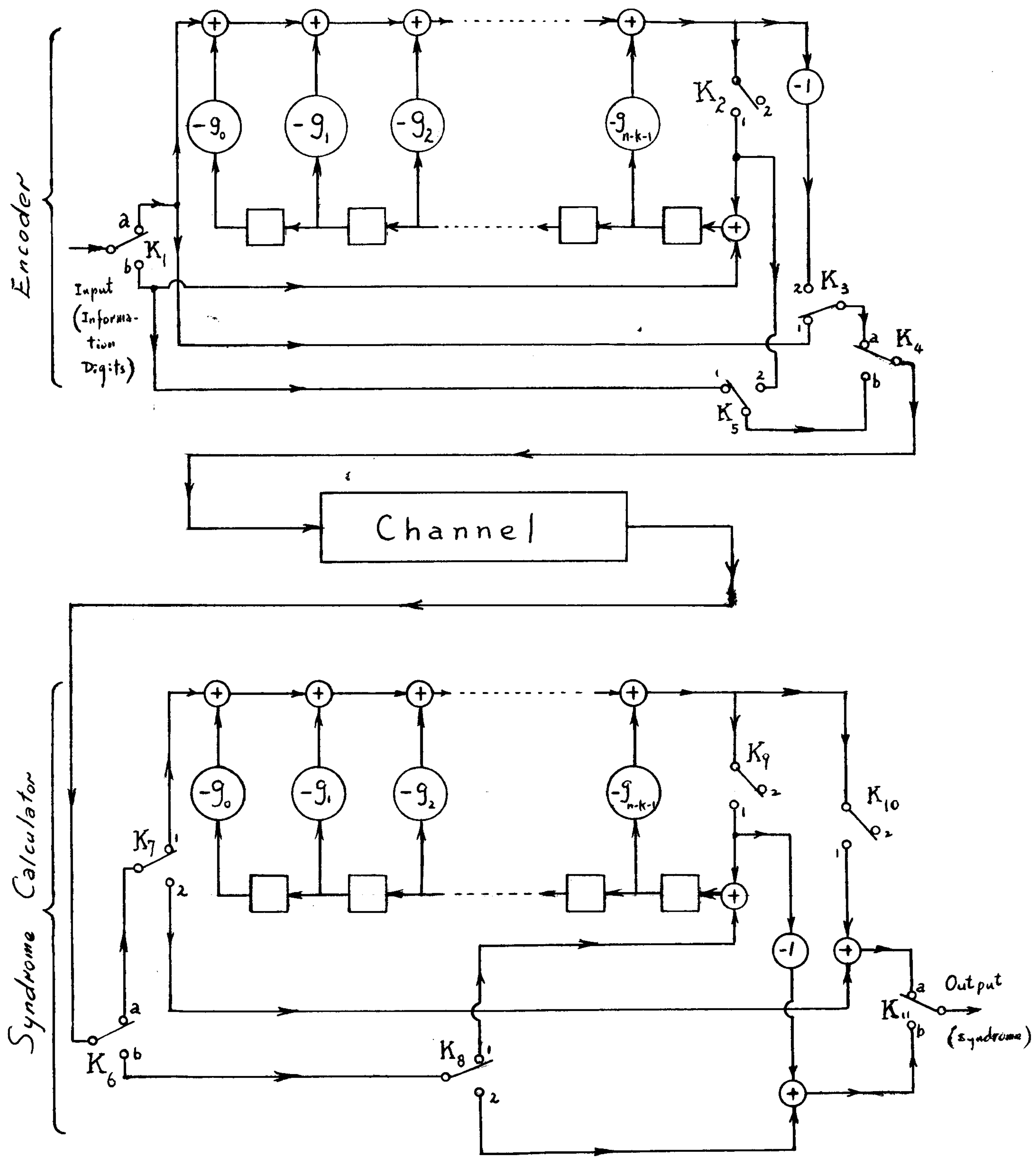


Fig. 4.11 Encoder and Syndrome Calculator for Dual Cyclic Codes

Mode 2: The code is generated by h(x)

Encoder

Switch \ Time Position	$0^+ \rightarrow kT^-$	$kT^+ \rightarrow nT^-$
K_1	b	b
K_2	2	1
K_4	b	b
K_5	1	2

Syndrome Calculator

Switch \ Time Position	$0^+ + t_D \rightarrow kT^- + t_D$	$kT^+ + t_D \rightarrow nT^- + t_D$
K_6	b	b
K_8	1	2
K_9	2	1
K_{11}	b	b

Fig. 4.11 can be represented in block diagram manner by Fig. 4.12.

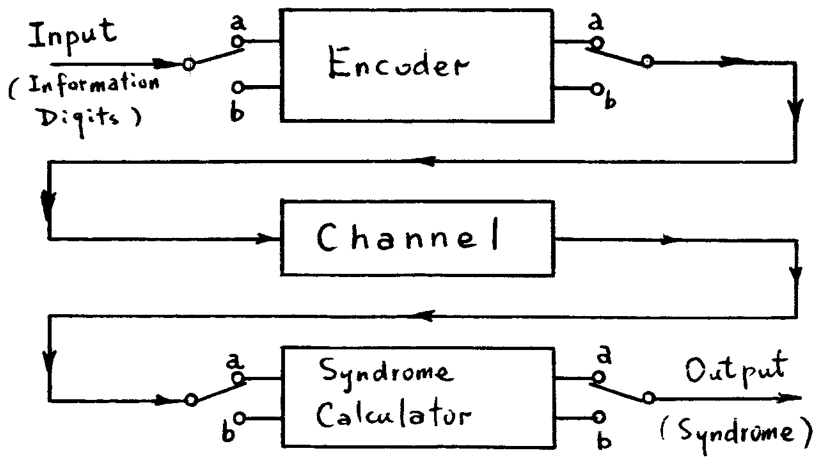


Fig. 4.12 A Block Diagram Representation of Fig. 4.11

When all the switches are at position a, the system is in mode 1. In this case the polynomial $g(x)$ is the generator polynomial of the code. When all the switches are at position b, the system is in mode 2. The code is now generated by the polynomial $h(x) = (x^n-1)/g(x)$.

4.5 Encoding of a Cyclic - Component Product Code with Parity-Check Matrix $H = H_1 \otimes H_2$

Let $g_1(x) = g_{10} + g_{11}x + g_{12}x^2 + \dots + g_{1(r_1-1)}x^{r_1-1} + x^{r_1}$ be the generator polynomial of the first cyclic component code C_1 with parity check matrix H_1. Let $g_2(x) = g_{20} + g_{20}x + \dots + g_{2(r_2-1)}x^{r_2-1} + x^{r_2}$ be the second cyclic component code C_2 with check matrix H_2. Then Fig. 3.3 of Chapter 3 can be replaced by Fig. 4.13. It should be noted that the accumulating adders over GF(q) do not appear in Fig. 4.13, since their functions are performed by the dividers. Each divider has a period of n_2, the length of the second component. Therefore, whatever is appearing at the shift-registers now will reappear n_2 shifts later provided there is no input. In the case of an input, what appear at the shift-registers n_2 shifts later is their original content plus the shifted version of the input which arrived in the interval of n_2 shifts, since the circuit is linear. The gate of each divider is always closed except during the interval when the remainder is shifted out to become the check digits. The multiplier controlling signals $a_1, a_2, \dots, a_{r_2}$ are the outputs taken at proper points shown in Fig. 4.14 with $(1,0,0,\dots,0)$ as the initial state, where $g_1^*(x) = g_{10}^* + g_{11}^*x + \dots + g_{1(r_1-1)}^*x^{r_1-1} + x^{r_1}$ is the reciprocal polynomial of $g_1(x)$. It should be noted that the clock frequency of Fig. 4.14

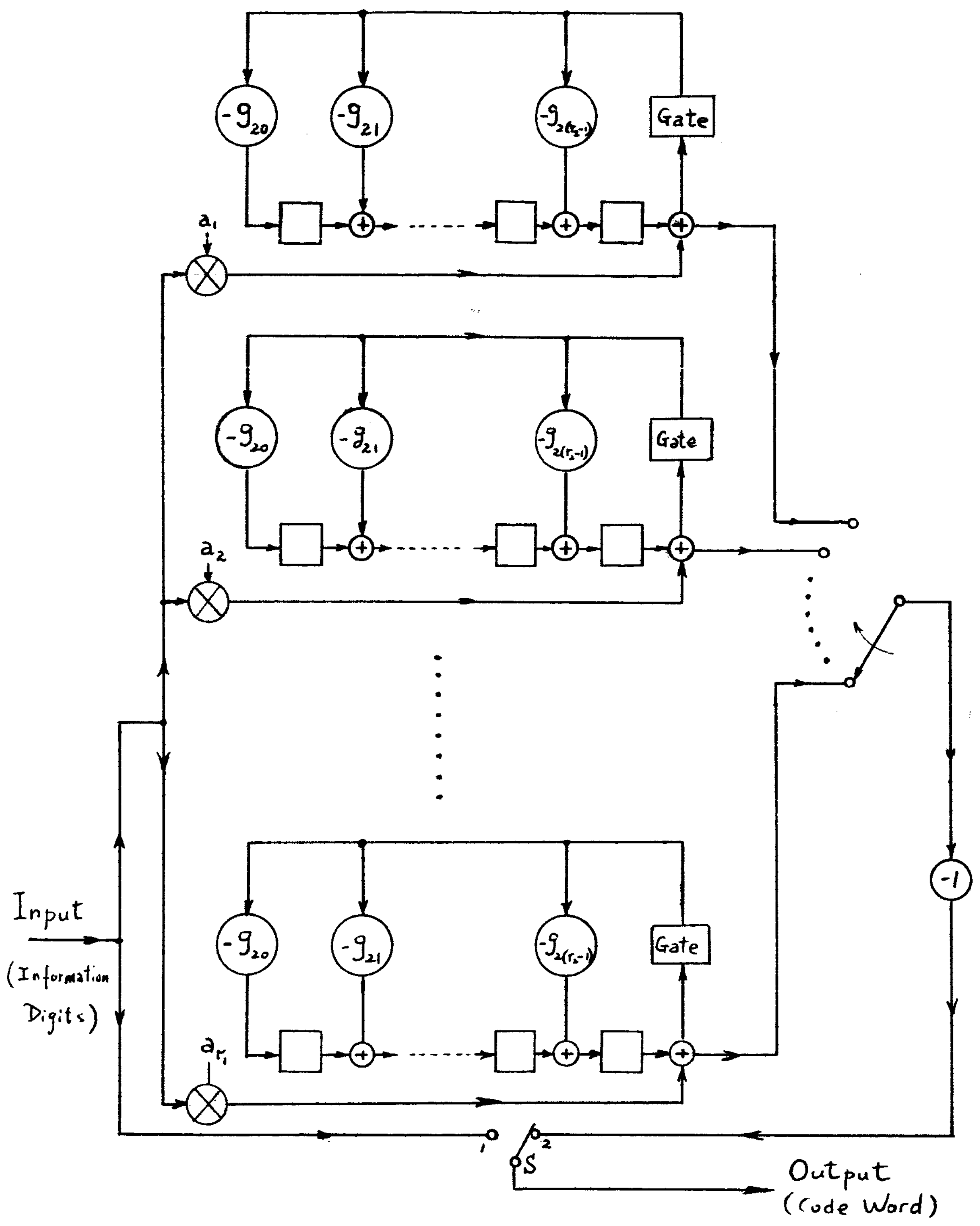


Fig. 4.13 Encoder for the Cyclic-Component Product Code with Parity-Check Matrix $H = H_1 \otimes H_2$

S is at 1 for the duration of information digits, at 2 for the duration of check digits.

is $1/n_2$ of the clock frequency of Fig. 4.13. Fig. 4.14 can be replaced by other storage devices or timing circuits.

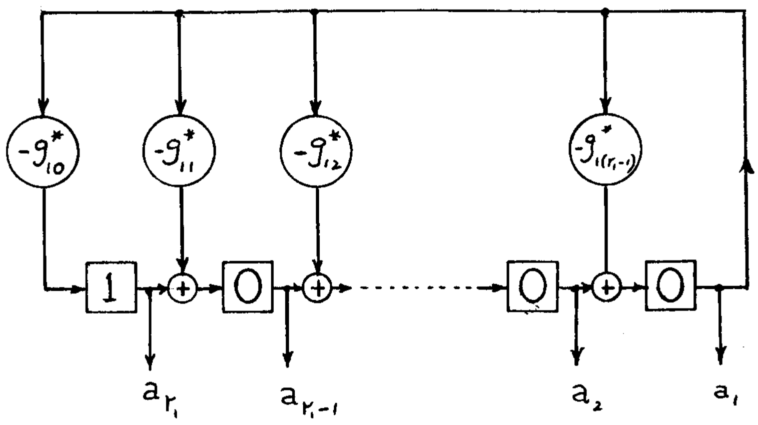


Fig. 4.14 Multiplier Controlling Signals

As far as economy of the circuitry is concerned, Fig. 4.13 is best for encoding the cyclic-component product code with parity-check matrix $H = H_1 \otimes H_2$, since no buffer stages and no accumulating adder over GF(q) are required, and there is no delay. But, unfortunately the flexibility of code word redundancy of a product code using Fig. 4.13 as encoder is limited. We can only change the first component code with parity-check matrix H_1 by varying the number of multipliers r_1, whereas we can do nothing about the second component code with parity-check matrix H_2. Therefore, we shall develop some alternative encoding circuits for use in later sections, such as those shown in Figs. 4.15 and 4.16.

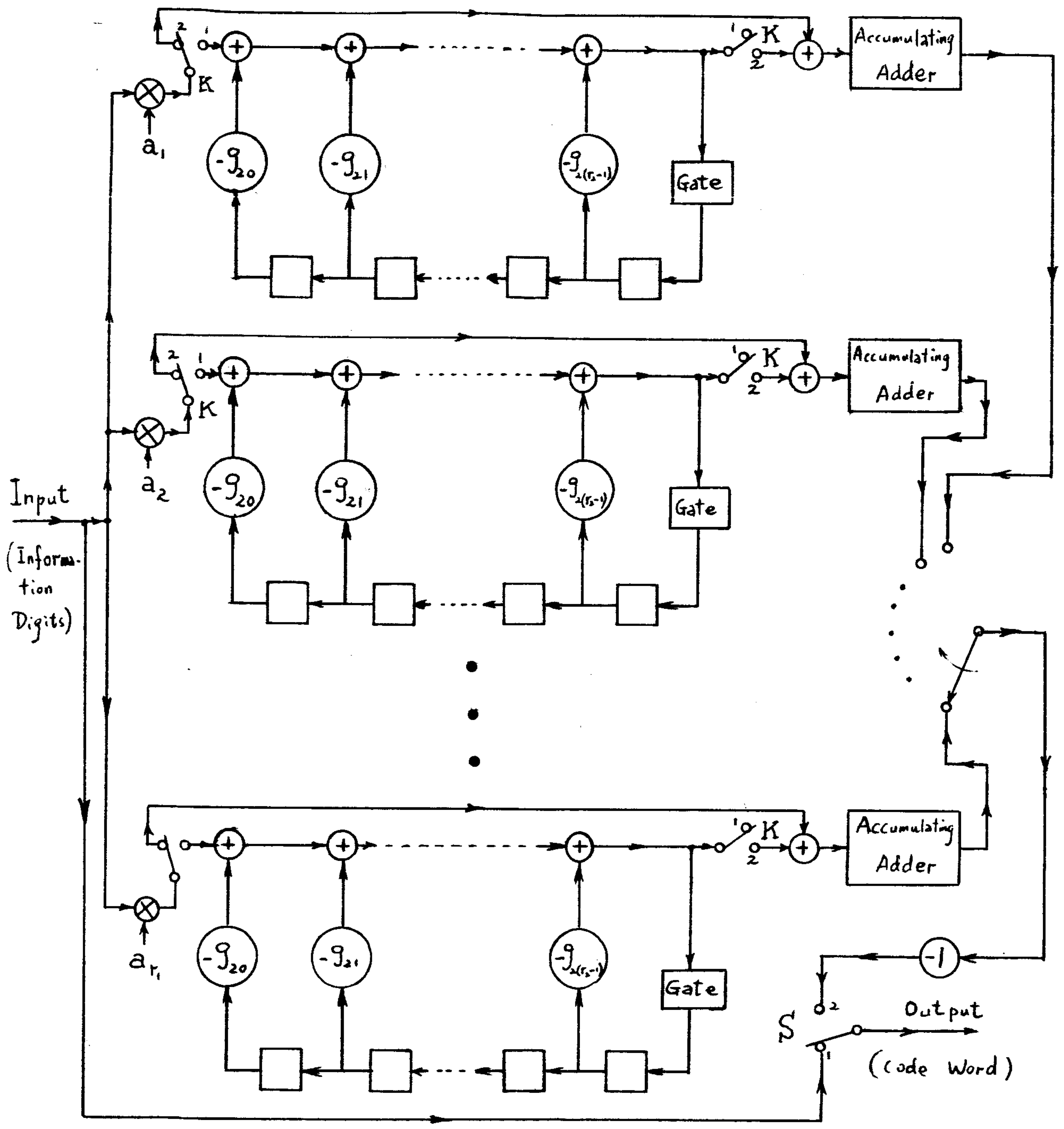


Fig. 4.15 An Encoder for the Cyclic-Component Code with Parity-Check Matrix $H = H_1 \otimes H_2$

K's are at position 1 for the first k_2 digits for each subblock and at position 2 for the last $n_2 - k_2$ digits. S is at position 1 for information digits and position 2 for check digits.

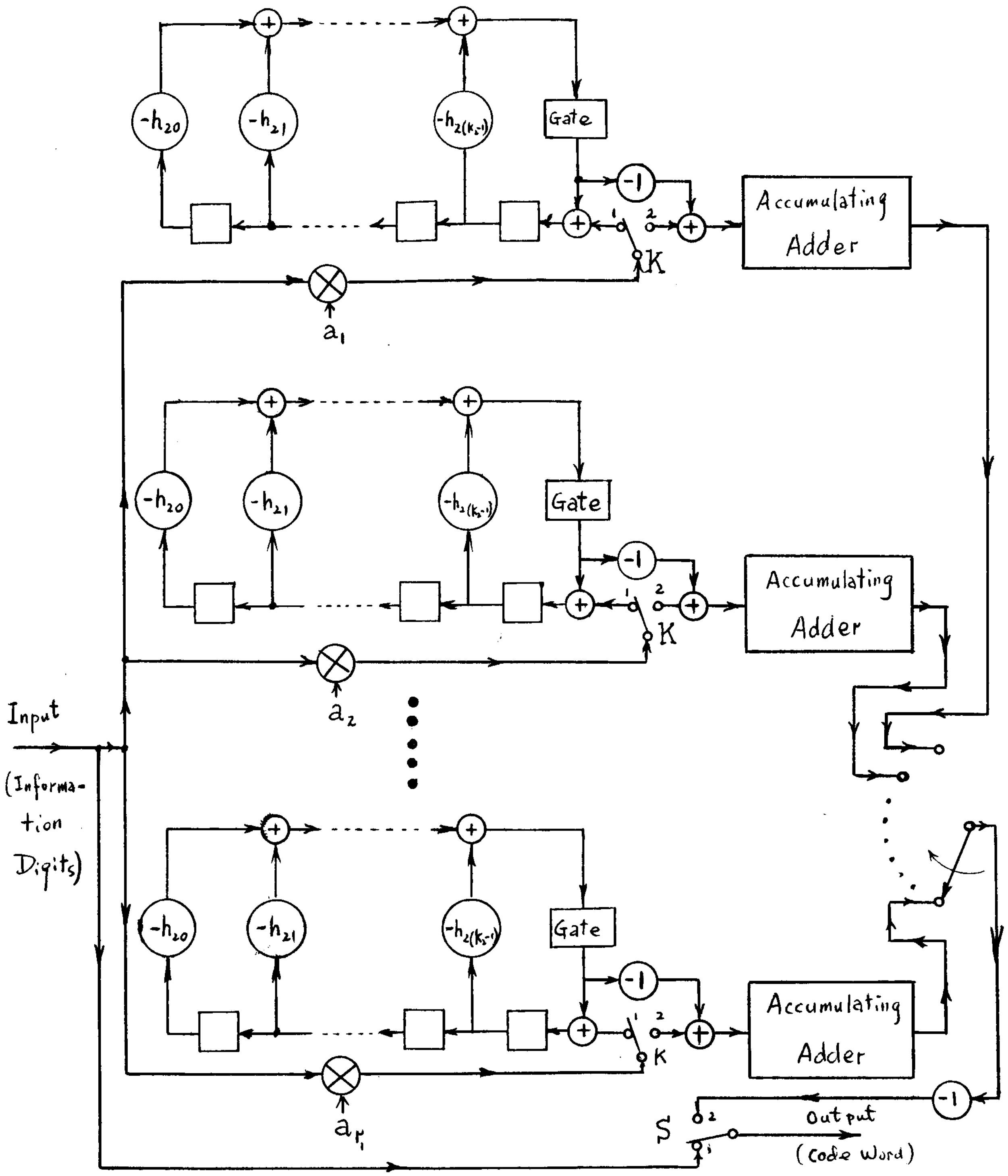


Fig. 4.16 An Encoder for the Cyclic-Component Code with Parity-Check Matrix $H = H_1 \otimes H_2$

Timing for K's and S are specified in Fig. 4.15.

There are several ways to implement the accumulating adders over GF(q) shown in Figs. 4.15 and 4.16. Since shift-registers are widely used for the encoding and decoding circuits for the cyclic codes, Fig. 4.17 suggests a way to achieve the accumulating adder by employing shift-registers. $m_0, m_1, \ldots, m_{r_2-1}$, and 1 are coefficients of a polynomial $m(x) = m_0 + m_1 + m_2x^2 + \ldots + m_{r_2-1}x^{r_1-1} + x^{r_1}$ which generates a recursive sequence of period n_2. Switch k is at position 2 most of the time except while output is being taken from the adder. It can be easily seen that if the input sequences at t_0, and $t_0 + n_2T$ are $(a_0, a_1, a_2, \ldots, a_{r_2-1})$ and $(b_0, b_1, b_2, \ldots, b_{r_2-1})$ respectively then at $t_0 + 2n_2T$ the output sequence should be

$$(a_0 + b_0, a_1 + b_1, \ldots, a_{r_2-1} + b_{r_2-1}) ,$$

where T is the time required for each shift. The polynomial m(x) may be the generator polynomial $g_2(x)$ of the second component code, but so far as the economy of circuitry is concerned, m(x) should be the polynomial of degree n-k with minimum number of non-zero coefficients which generates a recursive sequence of period n_2.

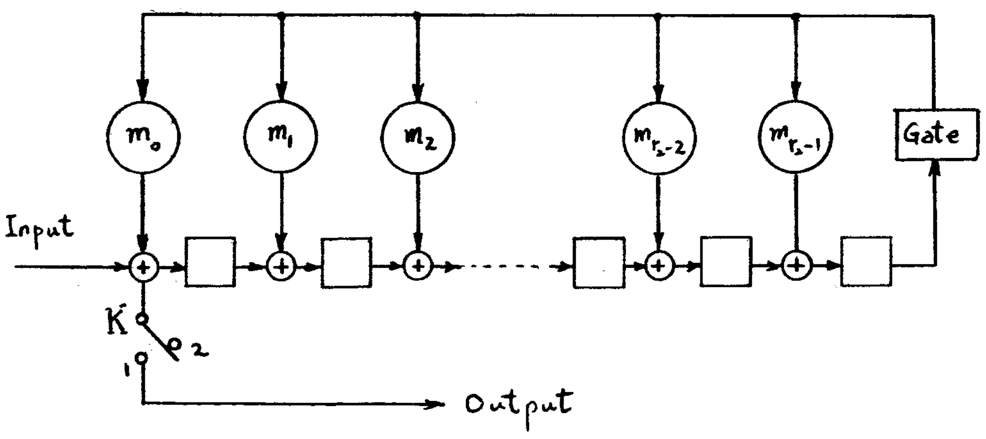


Fig. 4.17 An Accumulating Adder Over GF(q)

4.6 Syndrome Calculation of a Cyclic-Component Product Code with Parity-Check Matrix $H = H_1 \otimes H_2$

Figs. 4.13, 4.15 and 4.16 can be modified to serve as syndrome calculators of the cyclic-component product code with parity-check matrix $H = H_1 \otimes H_2$. The modified circuits are shown in Figs. 4.18, 4.19 and 4.20 respectively.

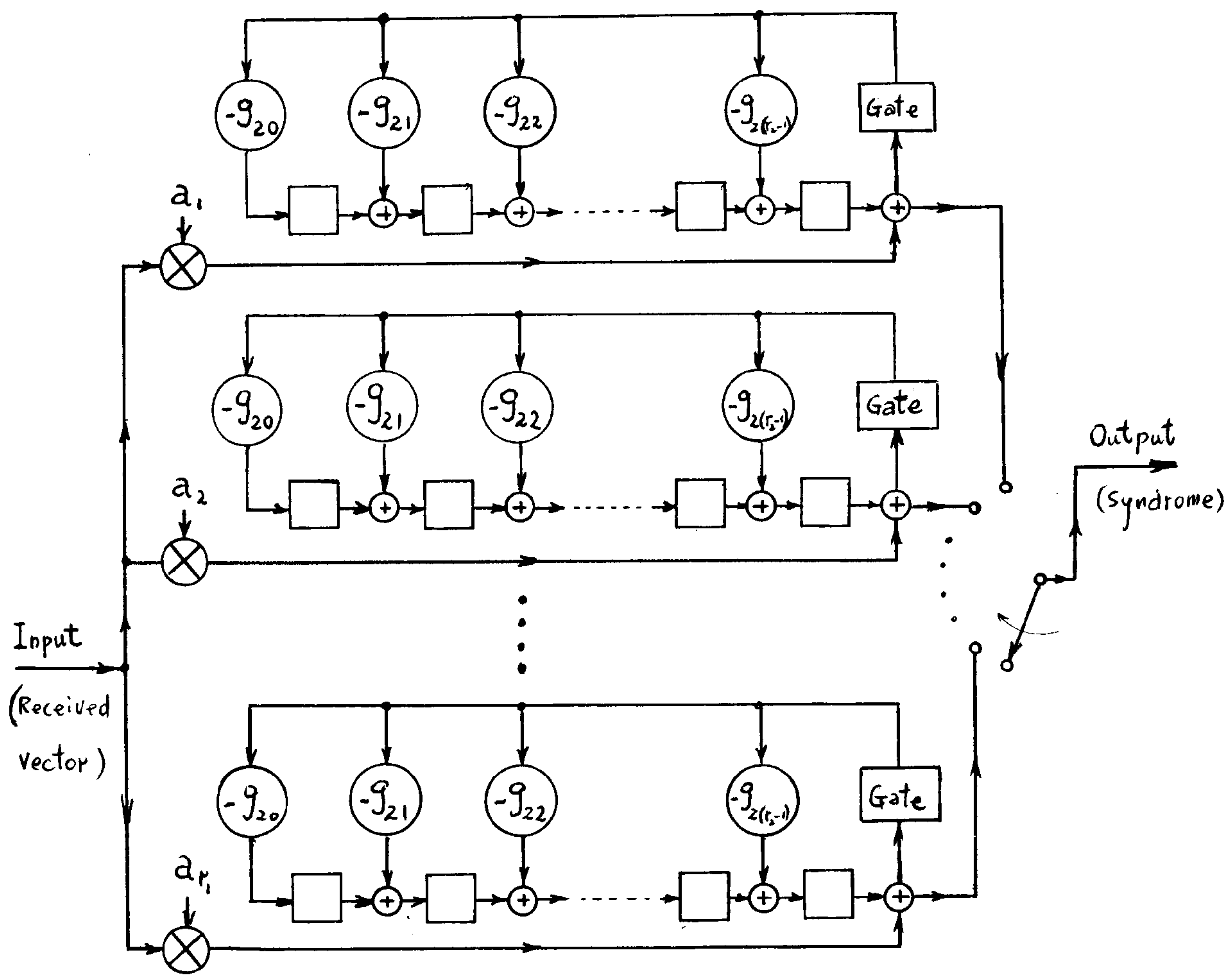


Fig. 4.18 Syndrome Calculator Corresponding to the Encoder of Fig. 4.13

The gate of each divider is open only when the remainder of that particular divider is shifting out as output.

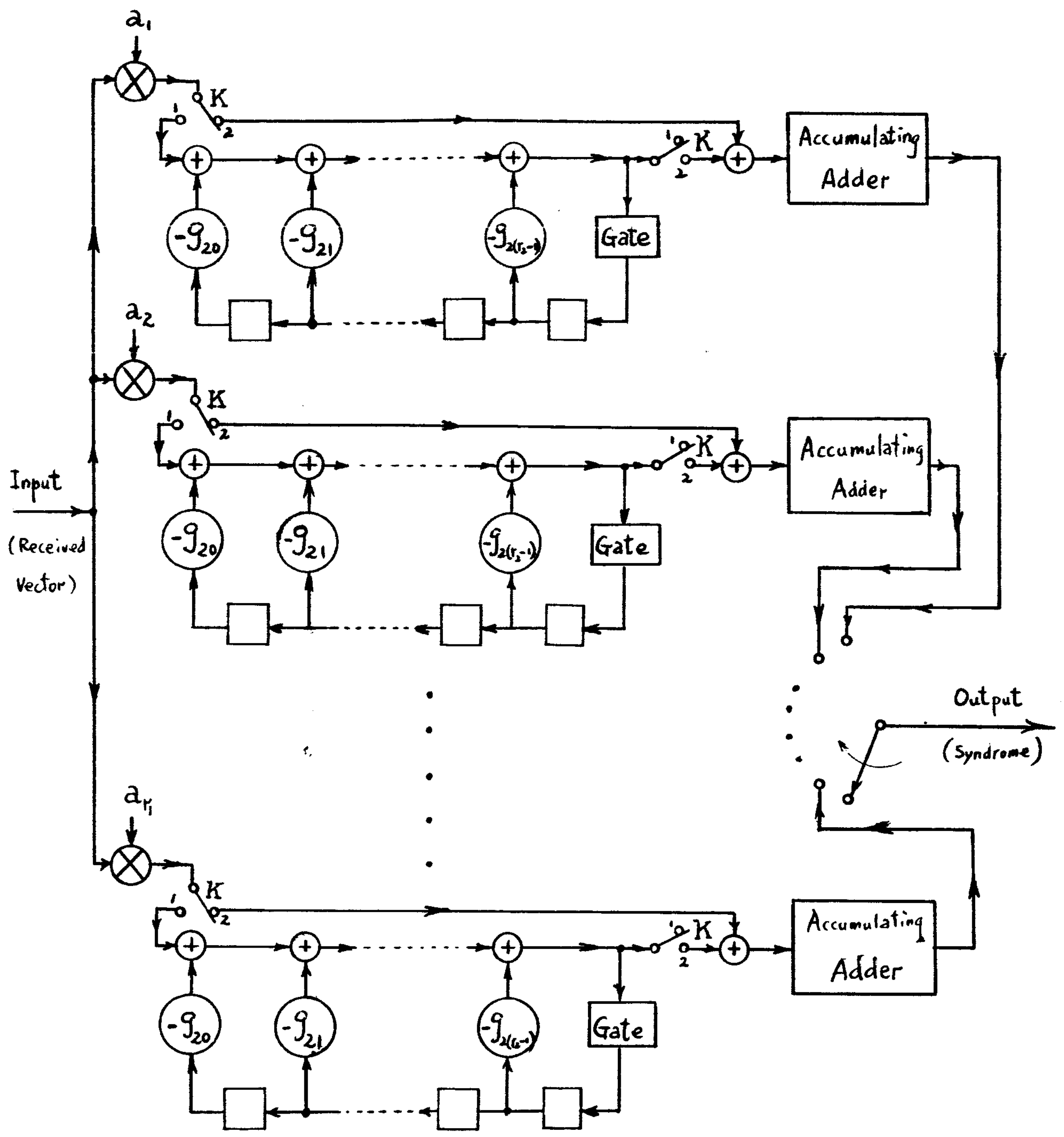


Fig. 4.19 Syndrome Calculator Corresponding to the Encoder of Fig. 4.15. Timing for the Switches and Gates is the same as that of Fig. 4.15.

Switches K are at position 1 for the first k_2 digits of each subblock of length n_2 and at position k the rest of time. The gate of each divider is open only when the remainder of that particular divider is shifting out as output.

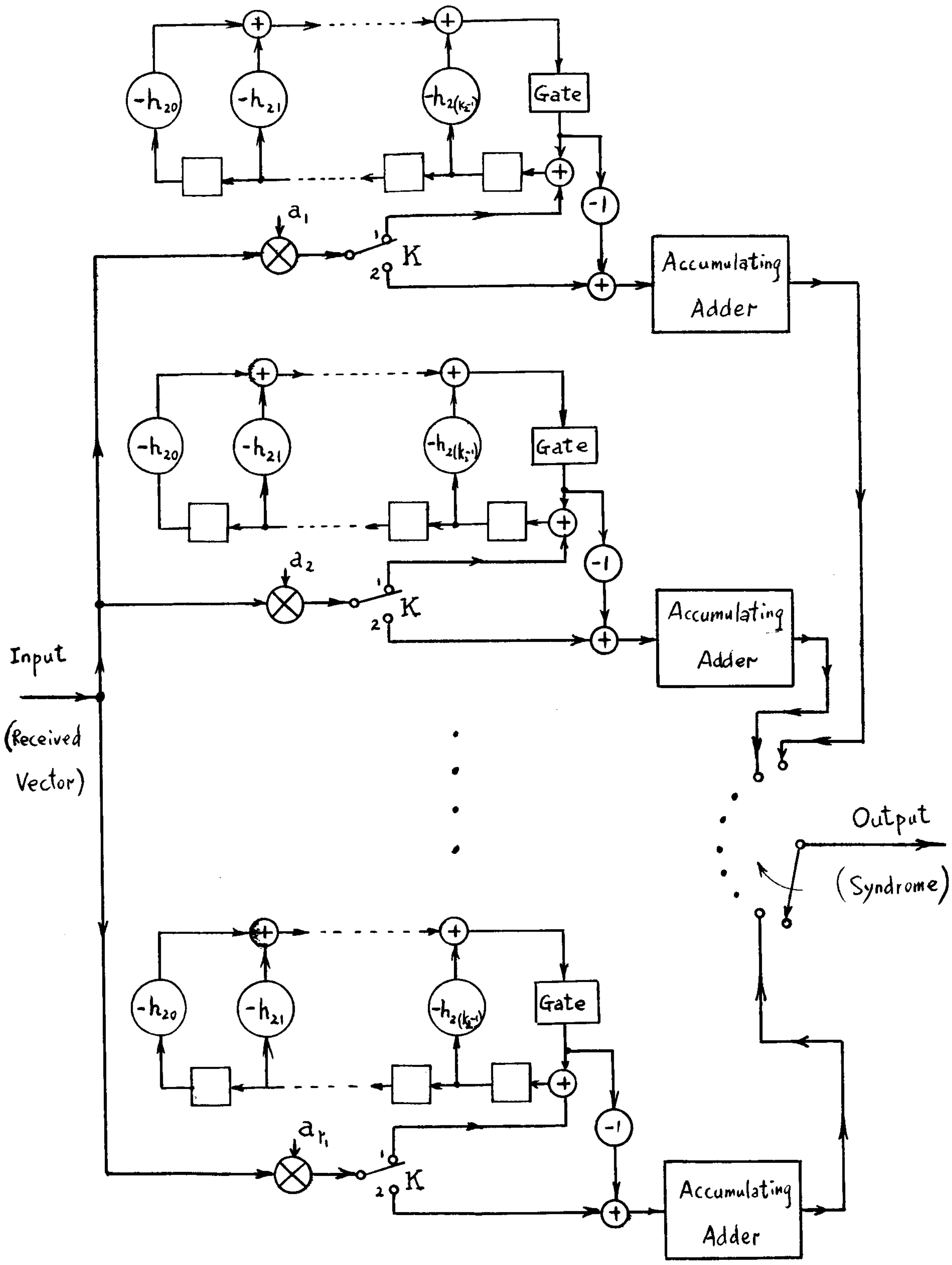


Fig. 4.20 Syndrome Calculator Corresponding to the Encoder of Fig. 4.16

4.7 Encoding of a Cyclic-Component Product Code with Generator Matrix $G = G_1 \otimes G_2$

Fig. 3.7 can be replaced by Fig. 4.21 for the case of cyclic-component codes. Two equivalent circuits are shown in Figs. 4.22 and 4.23. The generator matrix is $G = G_1 \otimes G_2$.

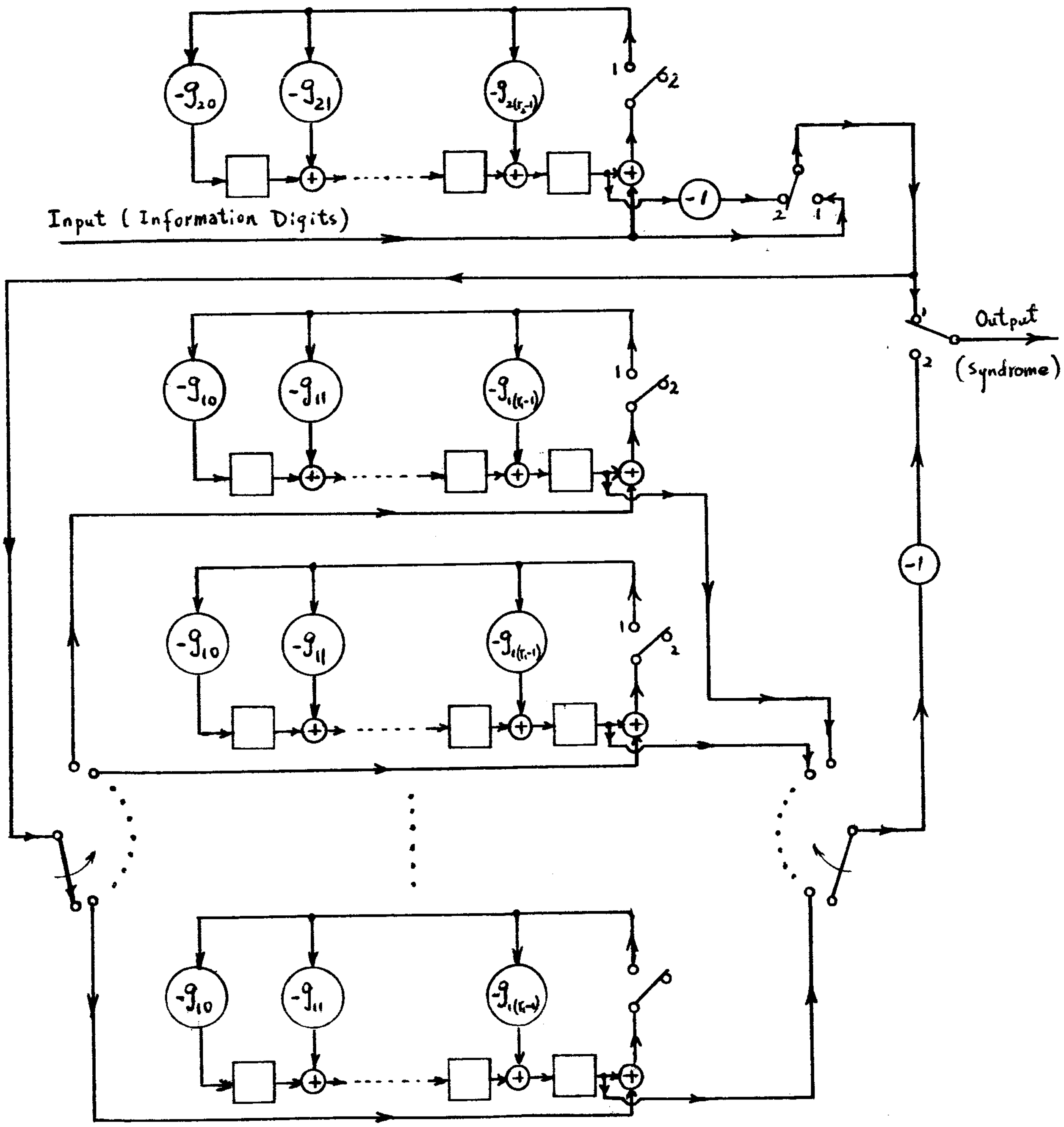


Fig. 4.21 Encoder for a Cyclic-Component Product Code with Generator Matrix $G = G_1 \otimes G_2$

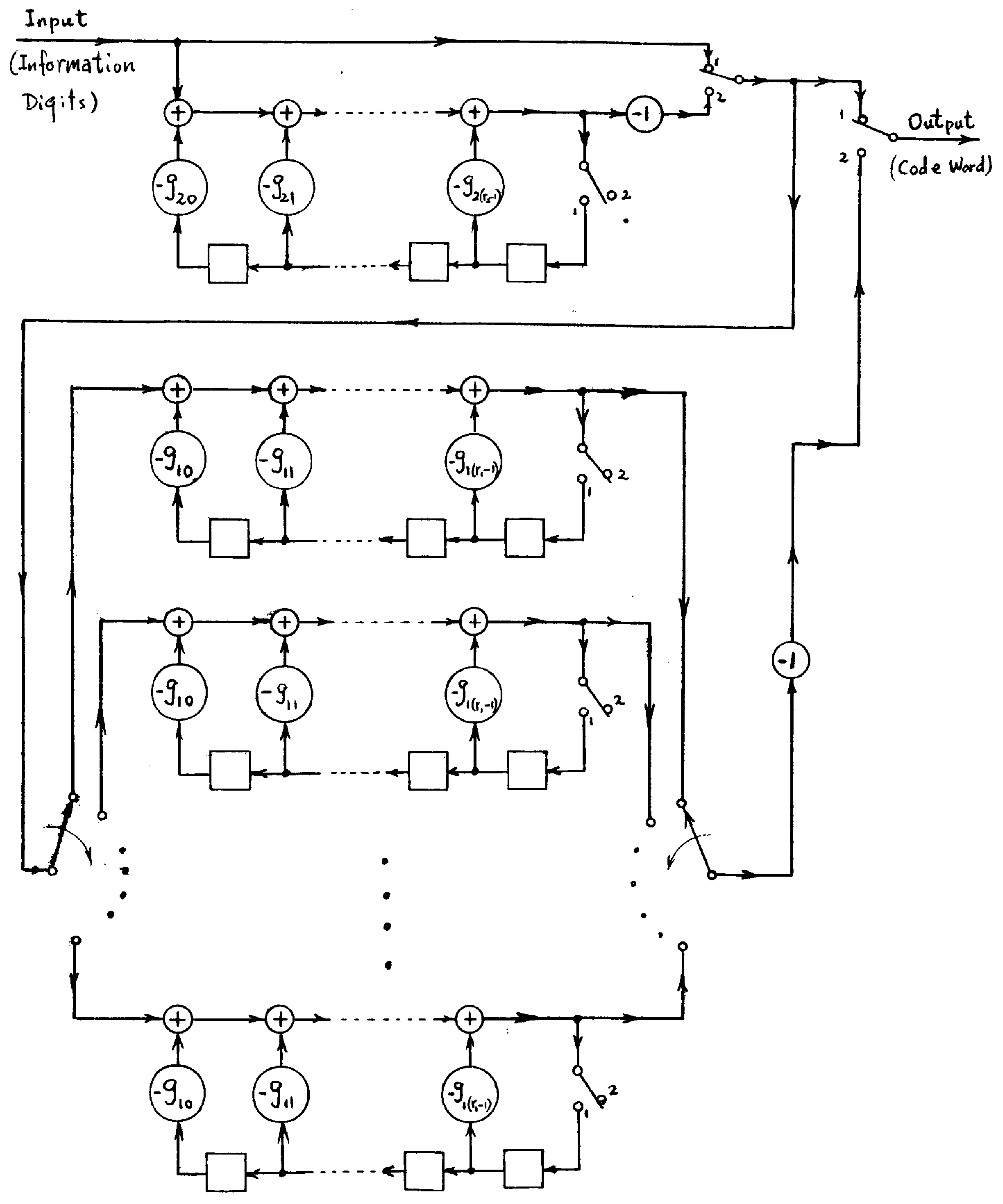


Fig. 4.22 An Equivalent Circuit of Fig. 4.21

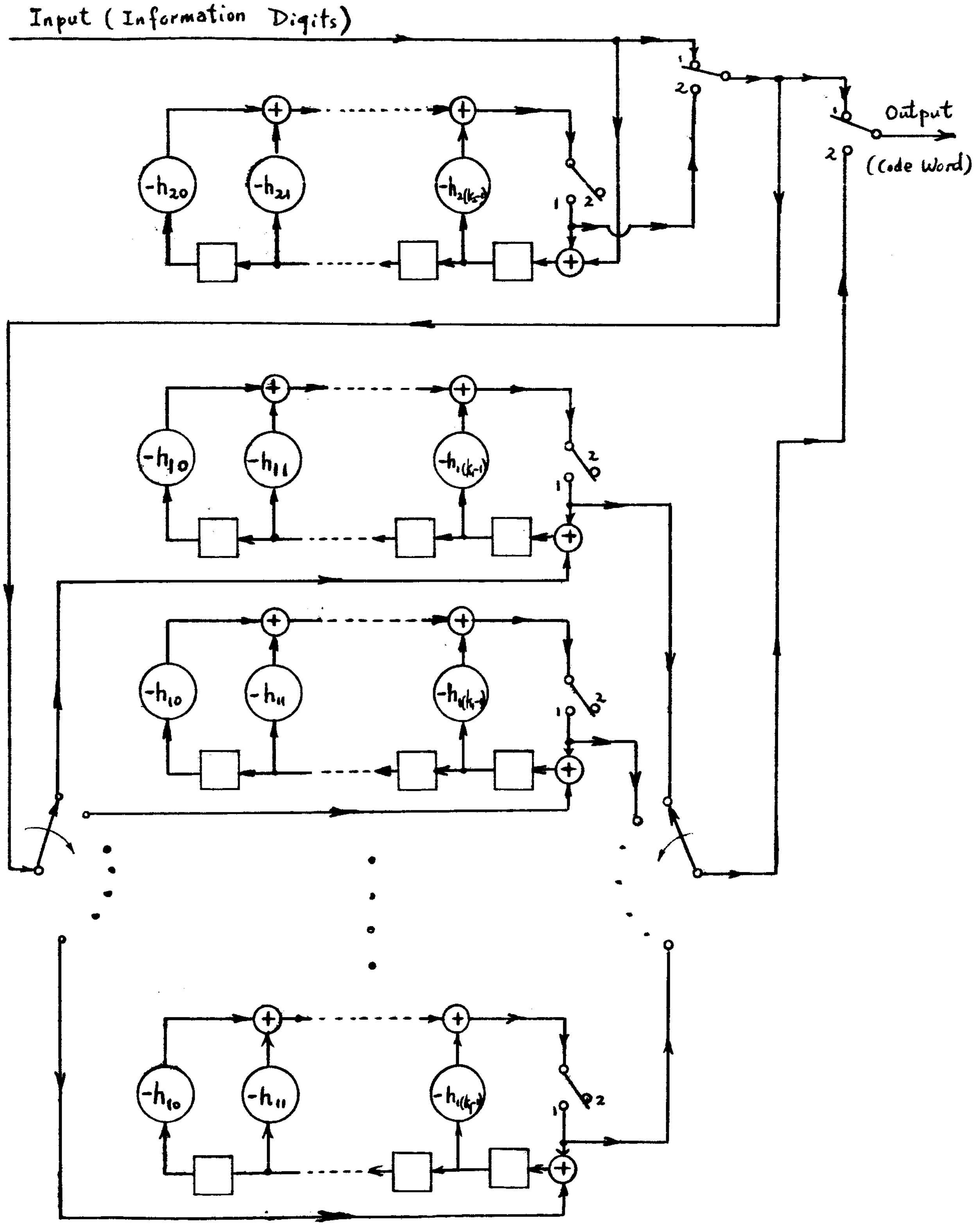


Fig. 4.23 Another Equivalent Circuit of Fig. 4.21

Another alternative of implementing the cyclic-component code with generator matrix $G = G_1 \otimes G_2$ is shown in Fig. 4.24.

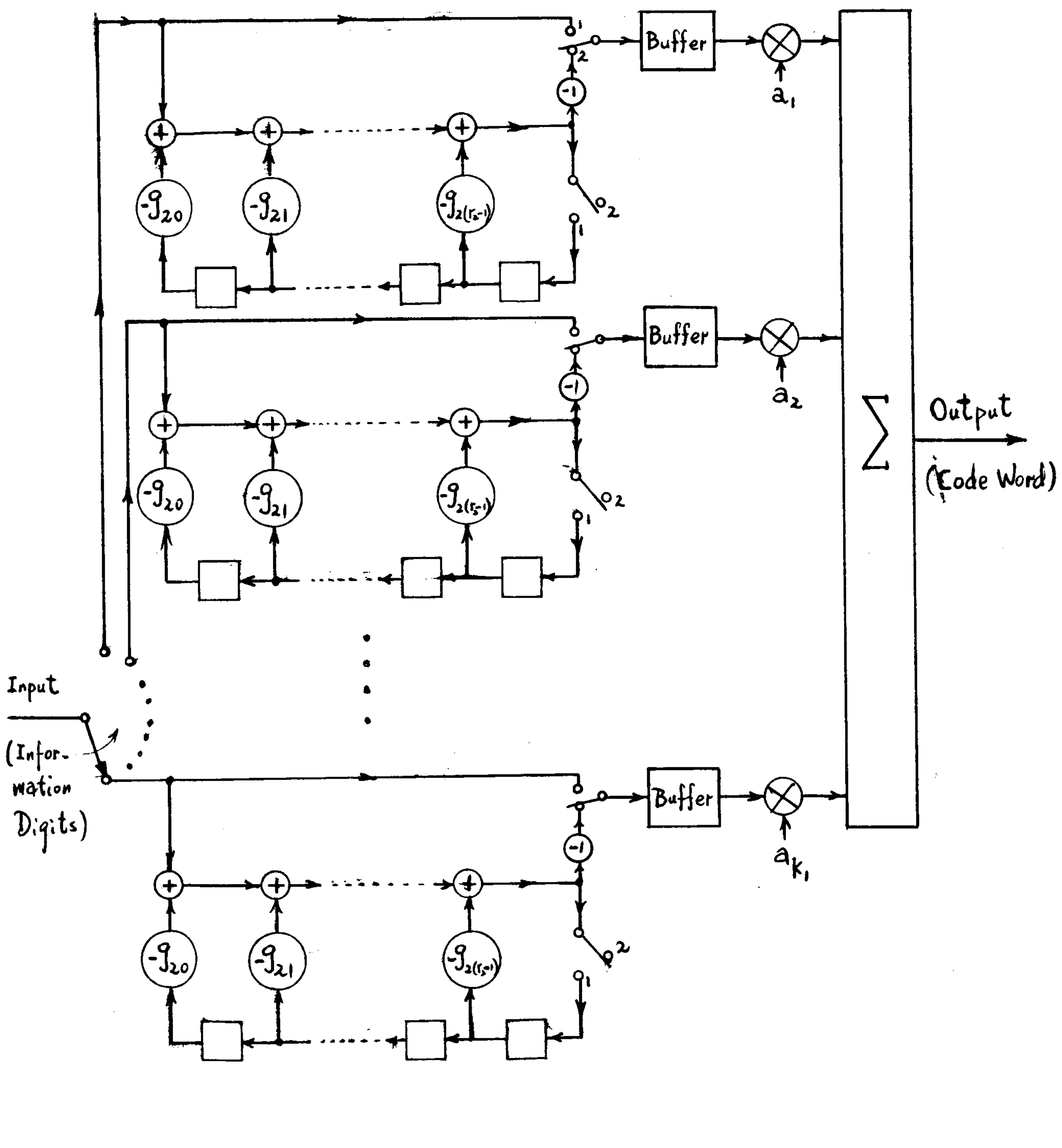


Fig. 4.24 Another Alternative Encoder for the Cyclic-Component Product Code with Generator Matrix $G = G_1 \otimes G_2$

There are k_1 encoders connected according to $g_2(x)$, the generator polynomial of the second component code. The output of each encoder is stored in a buffer stage or a repeater so that the sequence out of the second component encoder is repeated n_1 times. For the first n_2 digits, $(a_1, a_2, \ldots, a_{k_1})^T$ is the last column of G_1 while for the second n_2 digits, $(a_1, a_2, \ldots, a_{k_1})^T$ takes the value of the $(n_1-1)^{th}$ column of G_2, etc. An equivalent circuit of Fig. 4.24 is shown in Fig. 4.25.

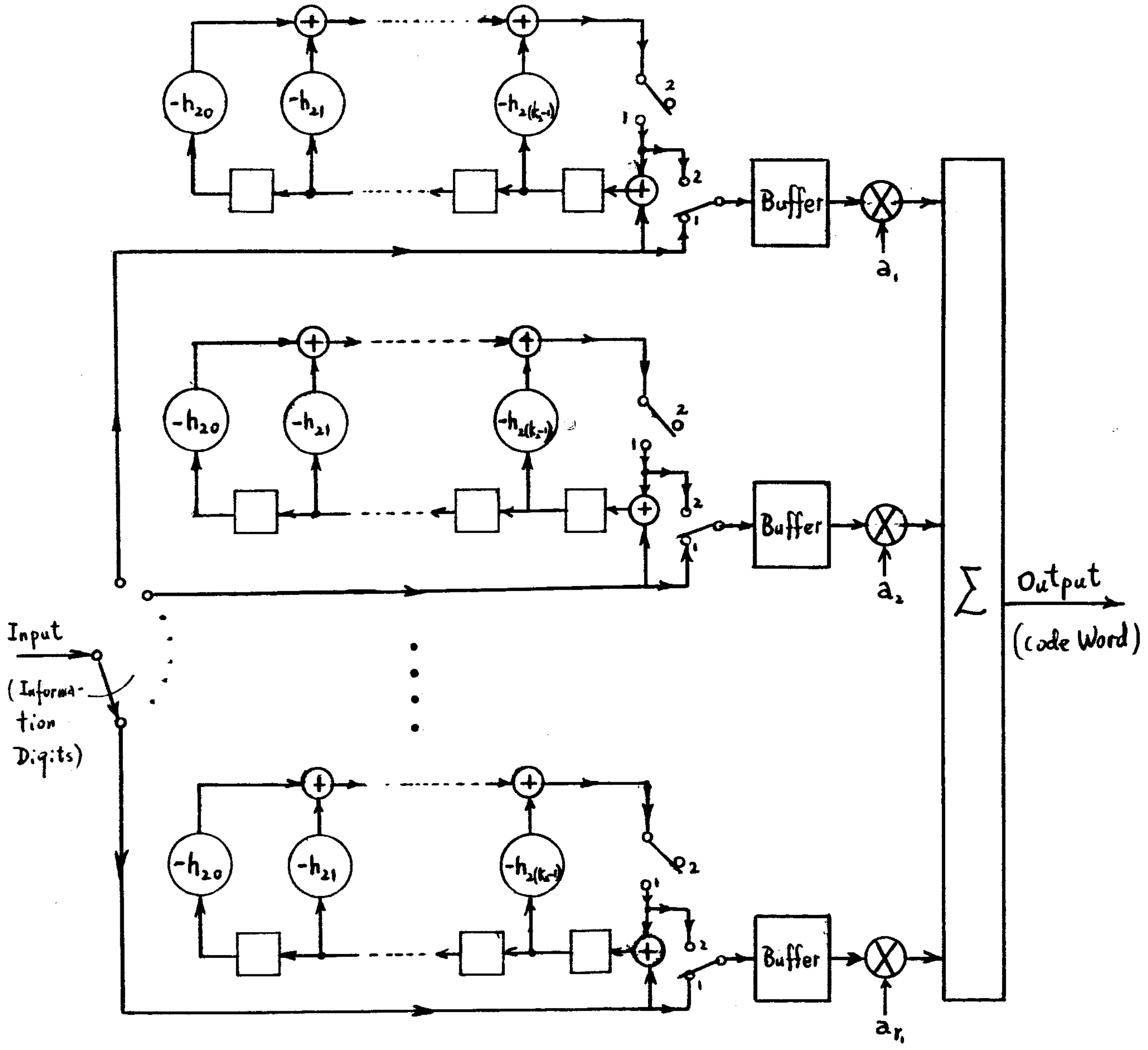


Fig. 4.25 An Equivalent Circuit of Fig. 4.24

4.8 Syndrome Calculation of a Cyclic-Component Product Code with Generator Matrix $G = G_1 \otimes G_2$

Let $\llcorner r \lrcorner$ be a received $n_1 n_2$-tuple and $\llcorner f \lrcorner$ be the code vector whose information digits are identical to those in $\llcorner r \lrcorner$. Then

$$\llcorner s' \lrcorner = \llcorner r \lrcorner - \llcorner f \lrcorner ,$$

where $\llcorner s' \lrcorner$ is the vector obtained by complementing the syndrome $\llcorner s \lrcorner$ with zeros at information positions. Hence the syndrome of the cyclic-component product code with generator matrix $G = G_1 \otimes G_2$ can be calculated by the circuit shown in Fig. 4.26. The switch K is at position 1 when check digits are fed in and at position 2 when information digits are fed in. Any circuit shown in Figs. 4.21, 4.22, 4.23, 4.24 and 4.25 can be used as the check digit calculator in Fig. 4.26.

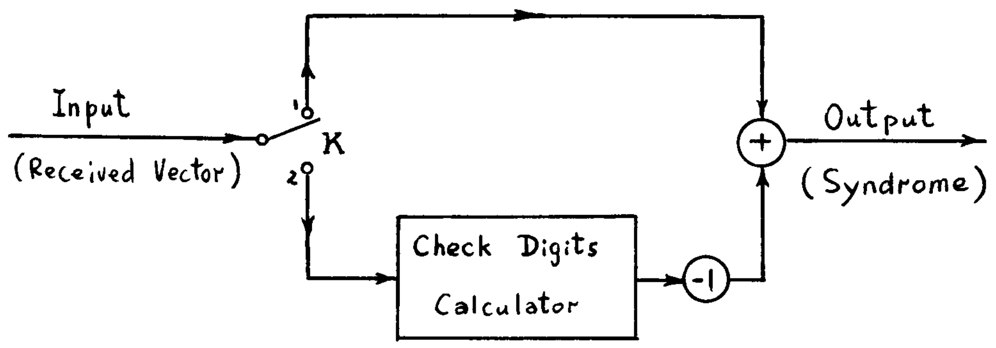


Fig. 4.26 Syndrome Calculator of the Cyclic-Component Code with Generator Matrix $G = G_1 \otimes G_2$

Usually, it is more convenient to have the syndrome given as in Equation (4.15). It can be shown that the syndrome shown in

$$\llcorner s \lrcorner = \llcorner r \lrcorner H^T = \llcorner r \lrcorner \begin{bmatrix} H_1 & \otimes I_{n_2} \\ I_{n_1} & \otimes H_2 \end{bmatrix}^T \tag{4.15}$$

can be obtained from the output of Fig. 4.26 by passing it through a linear circuit whose output vector components are linear combinations of those of the input vector. We may consider this linear transform circuit as part of the decoding circuit after syndrome calculation.

From the discussion of the previous sections in this chapter, we know that the circuit of a cyclic-component product encoder is quite similar to that of its syndrome calculator. Therefore, in the following sections of this chapter, we discuss only the encoder of a cyclic-component product code. But the results are also valid for the syndrome calculator.

4.9 Encoding of Cyclic-Component Product Codes Specified by $H = H_1 \otimes H_2$ with Dual Component Codes

Let C_1 be the systematic cyclic code of length n_1 generated by the polynomial $g_1(x)$ of degree $n_1 - k_1$. Then the dual code C_1' of C_1 is generated by the polynomial $h_1(x) = (x^{n_1} - 1)/g_1(x)$ which is of degree k_1. Similarly, let C_2 be the systematic cyclic code of length n_2 generated by the polynomial $g_2(x)$ of degree $n_2 - k_2$. Then the dual code C_2' of C_2 is generated by the polynomial $h_2(x) = (x^{n_2} - 1)/g_2(x)$ which is of degree k_2. Let the parity-check matrices of codes C_1, C_1', C_2 and C_2' be respectively H_1, H_1', H_2 and H_2'.

From Figures 4.13, 4.15 and 4.16, we know that the cyclic-component product codes with parity-check matrices $H_1 \otimes H_2$ and $H_1' \otimes H_2$ can be encoded by the same circuit, except that the second input of each multiplier a_i or a_i' is different for the code with parity-check matrix $H_1 \otimes H_2$ from that with $H_1' \otimes H_2$. The values of a_i's or a_i''s can be prestored in two separate storage devices. The operating mode of the encoder depends on which output of the storage devices is taken. Or, alternatively, Fig. 4.14 is used to generate one sequence, say the a_i's for the code with check matrix $H_1 \otimes H_2$, and a linear combination of a_i's gives the a_i''s for the code with check matrix $H_1' \otimes H_2$, as shown in Fig. 4.27. A detailed discussion of obtaining one recursive sequence from the other can be found in the literature.[5]

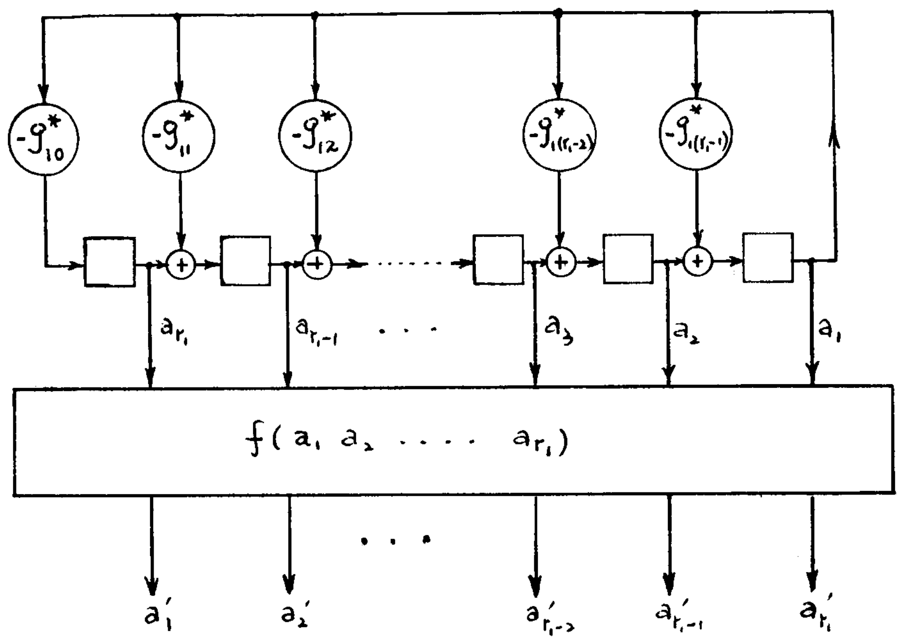


Fig. 4.27 A Circuit Generating Two Recursive Sequences

By examining the similarity between Fig. 4.15 and Fig. 4.16, we know that it is possible to begin by implementing the encoder for the cyclic-component product code with parity-check matrix $H_1 \otimes H_2$ according to Fig. 4.15. Then when the channel characteristics change we switch the inputs to, and take outputs from, each encoder of the second component code as shown in Fig. 4.16. In other words, we have the encoder changed to a new mode —— a cyclic-component code with parity-check matrix $H_1 \otimes H_2'$.

From the above discussion we can conclude that a single encoder can be used to encode four cyclic-component product codes with parity-check matrices $H_1 \otimes H_2$, $H_1' \otimes H_2$, $H_1 \otimes H_2'$ and $H_1' \otimes H_2'$. The interchange of first component code with its dual can be achieved by properly selecting the recursive sequence corresponding to H_1 or H_2. The choice of the second component code and its dual is determined by the input and output position of each second component encoder.

4.10 Encoding of Cyclic-Component Product Codes Generated by $G = G_1 \otimes G_2$ with Dual Component Codes

Let C_1, C_1', C_2 and C_2' be the codes specified in the last section, and their generator matrices be respectively G_1, G_1', G_2 and G_2'.

Examining the similarity between Fig. 4.22 and fig. 4.23, we know that a single circuit can be the encoder of the cyclic-component product codes with generator matrices $G_1 \otimes G_2$, $G_1 \otimes G_2'$, $G_1' \otimes G_2$ and $G_1' \otimes G_2'$. The change of modes is controlled by the input and output positions of the first and second component codes.

By reasoning similar to that given in the last section, the similarity between Fig. 4.24 and Fig. 4.25 can also be employed to design a single circuit with different modes, each of which corresponds to one cyclic-component product code.

4.11 Encoding of 8 Dual Cyclic-Component Product Codes by a Single Encoder

Examining the similarity between Fig. 4.15 and Fig. 4.24, and that between Fig. 4.16 and Fig. 4.25, we know that it is possible to encode the cyclic-component product with parity-check matrix $H = H_1 \otimes H_2$ and the cyclic-component product code with generator matrix $G = G_1 \otimes G_2$ by a single circuit, the change of mode being achieved by varying the input and output positions of both the first and second component codes. Also, from the discussions of the last two sections, we conclude that all eight cyclic-component product codes with duality relationship among either or all of their component codes can be implemented by a single encoder with switches used to change from one mode to the other.

A detailed example is given as follows.

Let C_1 be the (15,5) B-C-H 3-error-correcting code. The generator polynomial g(x) is

$$g_1(x) = (x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$$

$$= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

$$\text{and} \quad h_1(x) = \frac{x^{15}-1}{g_1(x)} = x^5 + x^3 + x + 1 \,.$$

The parity-check matrix is

$$H_1 = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&1&0&0&0&0&0&0&0&0&1&1&1&1&1 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&1&0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&1 \end{bmatrix} .$$

The generator matrix is

$$G_1 = \begin{bmatrix} 1&1&1&0&1&1&0&0&1&0&1&0&0&0&0 \\ 0&1&1&1&0&1&1&0&0&1&0&1&0&0&0 \\ 1&1&0&1&0&1&1&1&1&0&0&0&1&0&0 \\ 0&1&1&0&1&0&1&1&1&1&0&0&0&1&0 \\ 1&1&0&1&1&0&0&1&0&1&0&0&0&0&1 \end{bmatrix} .$$

Let C_1' be the dual of C_1, then C_1' is a (15, 10) 2-adjacent-error-correcting code with generator polynomial

$$g_1'(x) = h_1(x) = x^5 + x^3 + x + 1$$

and

$$h_1'(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Then,

$$H_1' = \begin{bmatrix} 1&0&0&0&0&1&0&1&0&0&1&1&0&1&1 \\ 0&1&0&0&0&1&1&1&1&0&1&0&1&1&0 \\ 0&0&1&0&0&0&1&1&1&1&0&1&0&1&1 \\ 0&0&0&1&0&1&0&0&1&1&0&1&1&1&0 \\ 0&0&0&0&1&0&1&0&0&1&1&0&1&1&1 \end{bmatrix}$$

$$G_1' = \begin{bmatrix} 1&1&0&1&0&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&1&0&1&0&0&0&0&0&0&0&0 \\ 1&1&1&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&1&1&1&0&0&0&0&1&0&0&0&0&0 \\ 1&1&0&0&1&0&0&0&0&0&1&0&0&0&0 \\ 1&0&1&1&0&0&0&0&0&0&0&1&0&0&0 \\ 0&1&0&1&1&0&0&0&0&0&0&0&1&0&0 \\ 1&1&1&1&1&0&0&0&0&0&0&0&0&1&0 \\ 1&0&1&0&1&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}.$$

Let C_2 be the (7,4) single-error-correcting cyclic Hamming code; C_2' be its dual. Then

$$g_2(x) = x^3 + x + 1$$

$$h_2(x) = x^4 + x^2 + x + 1$$

$$H_2 = \begin{bmatrix} 1&0&0&1&0&1&1 \\ 0&1&0&1&1&1&0 \\ 0&0&1&0&1&1&1 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 1&1&0&1&0&0&0 \\ 0&1&1&0&1&0&0 \\ 1&1&1&0&0&1&0 \\ 1&0&1&0&0&0&1 \end{bmatrix}$$

$$g_2'(x) = x^4 + x^2 + x + 1$$

$$h_2'(x) = x^3 + x + 1$$

$$H_2' = \begin{bmatrix} 1&0&0&0&1&0&1 \\ 0&1&0&0&1&1&1 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&0&1&1 \end{bmatrix}$$

$$G_2' = \begin{bmatrix} 1&1&1&0&1&0&0 \\ 0&1&1&1&0&1&0 \\ 1&1&0&1&0&0&1 \end{bmatrix} .$$

The eight cyclic component product codes are listed as follows:

Code	n-k	Min. Distance	Parity-Check Matrix	Generator Matrix
PC_1	30	3	$H_1 \otimes H_2$	
PC_2	40	4	$H_1 \otimes H_2'$	
PC_3	15	3	$H_1' \otimes H_2$	
PC_4	20	4	$H_1' \otimes H_2'$	
PC_5	85	21		$G_1 \otimes G_2$
PC_6	90	28		$G_1 \otimes G_2'$
PC_7	65	9		$G_1' \otimes G_2$
PC_8	75	12		$G_1' \otimes G_2'$

Table 4.1 8 Product Codes with Cyclic Component Codes

The overall length of each product code is $n_1n_2 = 105$.

An encoder which possesses all these eight modes is shown in Fig. 4.28.

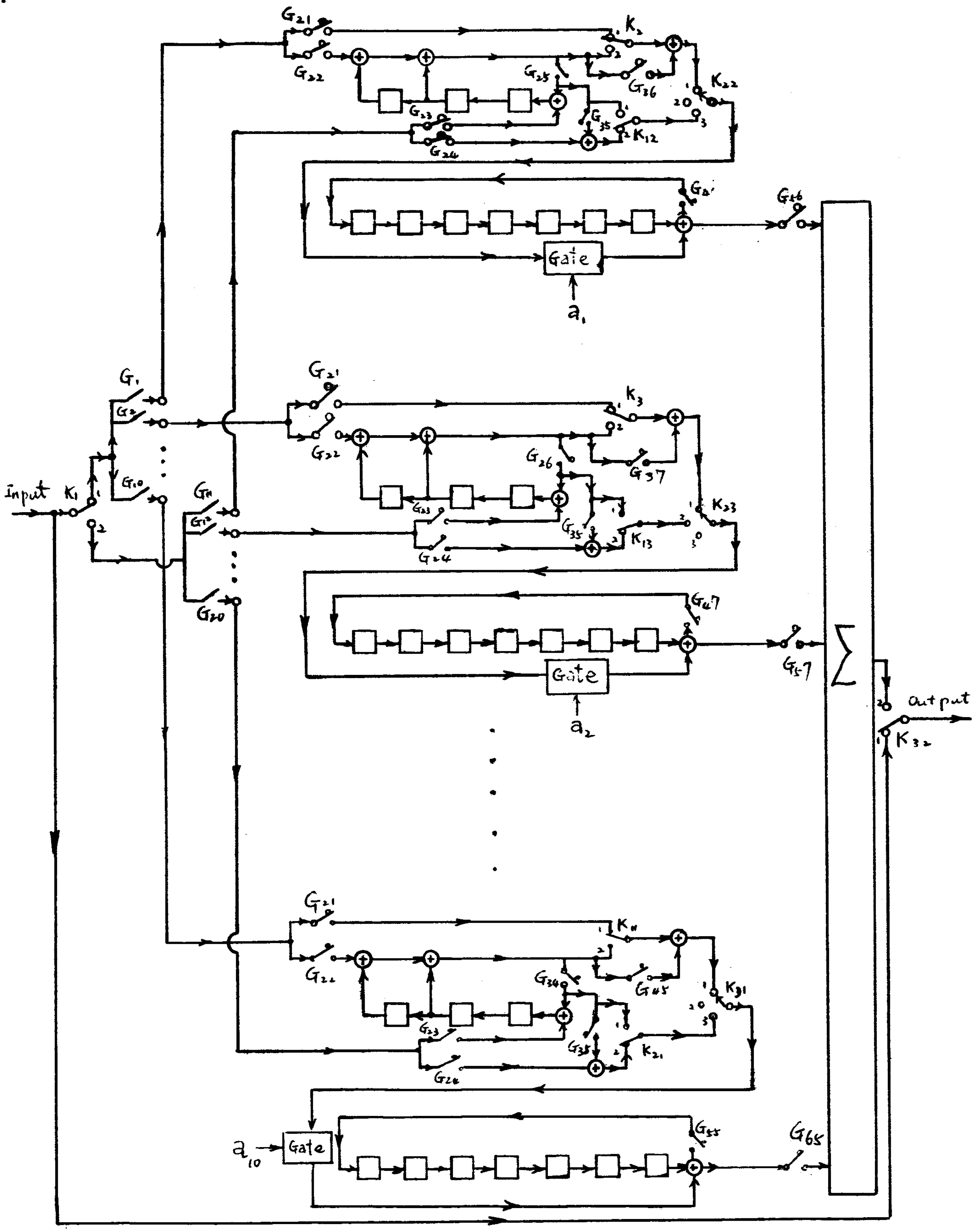


Fig. 4.28 An Encoder with Eight Operating Modes

The timing for the switches and gates in the PC_1 and PC_5 modes is given below:

For the PC_1 Code

	$7iT^+ \rightarrow (7i+4)T^-$	$(7i+4)T^+ \rightarrow (7i+7)T^-$	$i = 0,1,2,\ldots,14$
K_1	1	1	
$G_1 - G_{10}$	Closed	Closed	
$G_{11} - G_{20}$	Open	Open	
G_{21}	Open	Closed	
G_{22}	Close	Open	
$G_{25} - G_{34}$	Close	Open	
$K_2 - K_{11}$	1	1	
$G_{36} - G_{45}$	Open	Closed	
$K_{22} - K_{31}$	1	1	

K_{32} is at position 1 for $0T^+ - 39T^-$; $42T^+ - 46T^-$; $49T^+ - 53T^-$; $56T^+ - 60T^-$; $63T^+ - 67T^-$; $70T^+ - 74T^-$; $77T^+ - 81T^-$; $84T^+ - 88T^-$; $91T^+ - 95T^-$ and $98T^+ - 102T^-$. K_{32} is at position 2 for the rest of the time in a code word period.

$G_{46} - G_{55}$ are closed most of the time and G_{56} - G_{65} are open most of the time except during the following intervals.

$102T^+ - 105T^-$	:	G_{46} is open and G_{56} is closed.
$95T^+ - 98T^-$	:	G_{47} is open and G_{57} is closed.
$88T^+ - 91T^-$	:	G_{48} is open and G_{58} is closed.
$81T^+ - 84T^-$	:	G_{49} is open and G_{59} is closed.
$74T^+ - 77T^-$	:	G_{50} is open and G_{60} is closed.
$67T^+ - 70T^-$	:	G_{51} is open and G_{61} is closed.
$60T^+ - 63T^-$	:	G_{52} is open and G_{62} is closed.

$53T^+ - 56T^-$: G_{53} is open and G_{63} is closed.

$46T^+ - 49T^-$: G_{54} is open and G_{64} is closed.

$39T^+ - 42T^-$: G_{55} is open and G_{65} is closed.

For the PC_5 Code

The following gates are open throughout the code word period:

$$G_6 - G_{20}, \quad G_{23} - G_{24}, \quad G_{35} - G_{45}, \quad G_{61} - G_{65}.$$

The following gates are closed all the time

$$G_{21}, \; G_{22}, \; G_{56} - G_{60} .$$

K_1 is at position 1 all the time while K_{32} stays at position 2.

$G_{46} - G_{50}$ are closed for $0T^+ - 98T^-$ and open for $98T^+ - 105T^-$.

G_1 & G_{25} are closed only for $28T^+ - 32T^-$.

G_2 & G_{26} are closed only for $21T^+ - 25T^-$.

G_3 & G_{27} are closed only for $14T^+ - 18T^-$.

G_4 & G_{28} are closed only for $7T^+ - 11T^-$.

G_5 & G_{29} are closed only for $0T^+ - 4T^-$.

K_{i+1} is at position 1 when G_i is closed, and goes to position 2 when G_i is open for i = 1,2,3,4, and 5.

$K_{22} - K_{26}$ are at position 2 most of the time, except during the following intervals:

K_{22} is at position 1 only for $28T^+ - 35T^-$.

K_{23} is at position 1 only for $21T^+ - 28T^-$.

K_{24} is at position 1 only for $14T^+ - 21T^-$.

K_{25} is at position 1 only for $7T^+ - 14T^-$.

K_{26} is at position 1 only for $0T^+ - 7T^-$.

K_{32} is a position 2 all the time.

It should be noted that the multipliers of Figs. 4.15, 4.16, 4.24 and 4.25 can be replaced by gates for the binary case. The gate-controlling signals can be taken from the outputs of the shift registers as shown in Fig. 4.29. For the PC_1 code all the gates are closed, and the preset sequence for the shifts registers is

$$(1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1) = (a_{10}, a_9, a_8, a_7, a_6, \ldots, a_1).$$

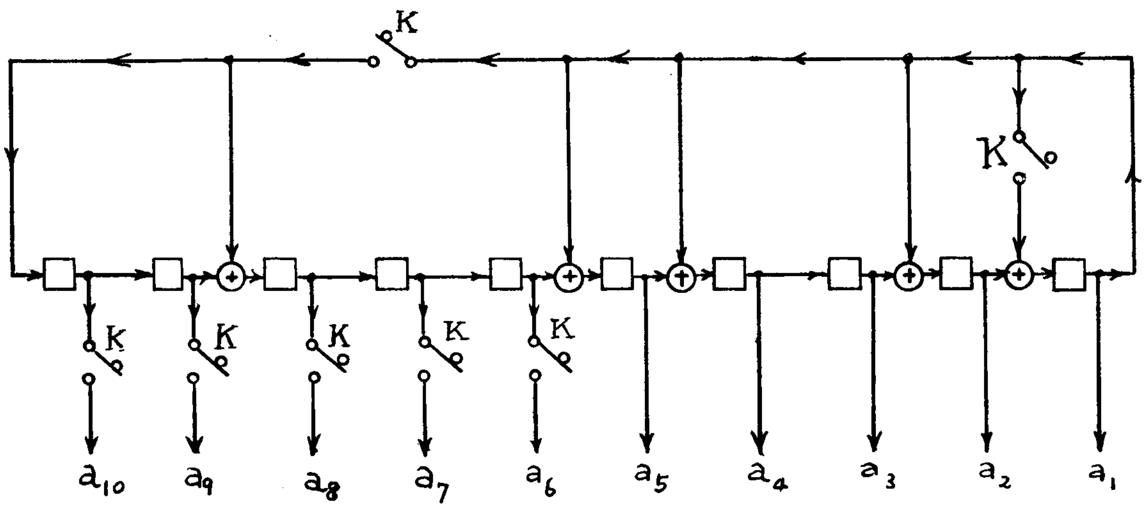


Fig. 4.29 Gates Controlling Signals

For the K's closed the polynomial is

$$x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1,$$

the reciprocal of $g_1(x)$. For the K's open the polynomial is

$$x^5 + x^3 + x + 1 = h_1(x).$$

For the PC_5 code, all the gates are open, and the preset sequence for the shift registers is

$$(1\ 0\ 0\ 0\ 0) = (a_5, a_4, a_3, a_2, a_1).$$

Other modes can be set up in a similar manner by proper assignment of the switching operations.

4.12 Remarks

Sections 4.1 through 4.3 are a brief review of the encoding and syndrome calculating circuits for the cyclic codes. Section 4.4 is similar to Tang's work[4], but it is pointed out in this section that both the encoder and the syndrome calculator of a cyclic code can be connected corresponding to either $g(x)$ or $h(x)$ whichever has the lower degree. Fig. 4 "A general purpose decoder for dual codes" of reference (4) is actually a syndrome calculator which can be a decoder only if the codes under considerations are Hamming single-error-correcting codes.

Sections 4.5 through 4.11 are original. Let us examine Table 4.1: The product codes PC_1 and PC_3 have the same minimum distance but different redundancy. It seems to imply that code PC_3 is better than code PC_1. This is actually not so, because PC_1 is capable of locating 3 independent corrupted subblocks containing 2 or fewer errors while PC_3 can locate only 2-adjacent erroneous subblocks containing 2 or fewer errors. A similar argument applies to codes PC_2 and PC_4. The variations of minimum distance among codes PC_5, PC_6, PC_7 and PC_8 are much larger than among the other 4 codes listed in Table 4.1.

If we only intend to design an encoder for 4 cyclic-component product codes whose generator matrices are given as $G_1 \otimes G_2$, $G_1' \otimes G_2$, $G_1 \otimes G_2'$ and and $G_1' \otimes G_2'$, then Figures 4.22 and 4.23 appear to be much simpler than Figures 4.24 and 4.25, since the former require no buffers, no multipliers and no restored sequences. But Figures 4.22 and 4.23 cannot be transformed to encode product codes with parity-check matrix $H_1 \otimes H_2$ by simply changing the input and output positions of the component codes.

If codes PC_5, PC_6, PC_7 and PC_8 in Table 4.1 are respectively replaced by their equivalent codes each with generator matrix $H_1' \otimes G_2$, $H_1' \otimes G_2'$, $H_1 \otimes G_2$ and $H_1 \otimes G_2'$, some saving in generating the gate controlling signals may result

The high degree of flexibility in code redundancy is quite useful in space-vehicle-to-ground communication, since the channel characteristics are affected by several factors such as distance, vehicle velocity and weather. It requires a coding scheme with variable error-correctability at different times so that the transmission error probability may always be kept within tolerance while the transmission rate is correspondingly adjusted. Since the decoders are usually much more complicated, it is conjectured that the vehicle will contain only the transmitter and decoder to provide one-way communication.

Having presented the syndrome calculations, we shall discuss some decoding schemes in the next chapter.

CHAPTER 5

A NOTE ON DECODING SYSTEMATIC CYCLIC CODES

The first error-correction procedure for the binary Bose-Chaudhuri-Hocquenghem codes was found by Peterson.[32] The extension to the decoding for non-binary codes was discovered by Gorenstein and Zierler.[33] Recently, some new decoding procedures and algorithms for the B-C-H codes have been suggested by several other authors.[15,16,17,18] Along another line, Prange[30] discovered the permutation decoding procedure for the systematic cyclic codes in general. This scheme was carried on further by MacWilliams.[31]

The procedure of cyclic permutation decoding is straightforward, and the circuit for implementing it is simple. Unfortunately, this decoding scheme is capable of correcting only a part of all the correctable error patterns of the code. While all the other decoding procedures mentioned in the last paragraph are tedious and the circuits complex, they are designed to correct all the errors specified by the code capability. Therefore, we suggest decoding a cyclic code mainly by the cyclic permutation procedure, which can decode a large fraction of all the received vectors; whenever the cyclic permutation fails, an auxiliary decoding scheme is used to correct the remaining error patterns. In so doing, the total decoding time will be significantly less than when employing any other decoding scheme alone since most of the time the cyclic permutation decoding procedure will be used and the auxiliary circuits will be idle.

In this chapter, the cyclic permutation decoding scheme is first described. The circuit to implement such a scheme is then discussed.

A detailed example of decoding the B-C-H (15,5) code employing cyclic permutation, and a simple auxiliary scheme are illustrated. It is believed that the combined scheme for decoding the (15,5) code is the simplest one of all the existing procedures. It is doubtful that such simple auxiliary schemes will always be available to complement the capability of the cyclic permutation decoding.

5.1 Cyclic Permutation Decoding of Systematic Cyclic Codes

The decoding procedure for cyclic codes, in general, involves many difficult steps beyond syndrome calculation. In particular it involves computations over the extension field $GF(q^m)$. The decoding scheme suggested here makes full use of the cyclic property of a cyclic code. With a straightforward division (i.e., syndrome calculation) this method is capable of correcting any pattern of information digits in a code word. For a t-error-correcting (n,k) cyclic code, if the errors are so distributed that there exist k consecutive error-free digits (k' digits at the end of the code word and k-k' digits at the beginning of the code word are also considered as k consecutive digits) then t or fewer errors can be corrected.

It is well known that the parity check matrix H of an (n,k) systematic cyclic code can be written as

$$H = [I_{n-k}\ P], \tag{5.1}$$

where P is an $(n-k) \times k$ matrix. Let $\underline{r}$ be the received vector, then

$$\underline{r} = \underline{f} + \underline{e}, \tag{5.2}$$

where $\underline{f}$ is the correct code vector and $\underline{e}$ is the error pattern. The resulting syndrome is

$$\begin{aligned} \underline{s} = \underline{r}\, H^T &= (\underline{f} + \underline{e})\, H^T \\ &= \underline{f}\, H^T + \underline{e}\, H^T \\ &= \underline{e}\, H^T . \end{aligned} \tag{5.3}$$

Since $$\underline{f}\, H^T = 0. \tag{5.4}$$

If the error pattern is such that the last k places are all zero, i.e.,

$$\underline{e} = (e_0, e_1, e_2, \ldots, e_{n-k-1}, 0\ 0, \ldots, 0), \tag{5.5}$$

then
$$\underline{s} = \underline{e}\, H^T = (e_0, e_1, \ldots, e_{n-k-1}). \tag{5.6}$$

Thus the syndrome exhibits the original error pattern. Now suppose that $\underline{e}$ is of the form

$$\underline{e} = (\, e_0, e_1, \ldots, e_{i-1}, \underbrace{00000, \ldots, 0}_{k \text{ zeros}}, e_{i+k}, e_{i+k+1}, \ldots, e_{n-1}). \tag{5.7}$$

Let $\underline{\pi}_c$ be the cyclic permutation matrix, then

$$(a_0, a_1, \ldots, a_{n-1})\, \pi_c = (a_{n-1}, a_0, a_1, \ldots, a_{n-3}, a_{n-2}). \tag{5.8}$$

By the fact that the cyclic permutation of a code word is still a code word, the syndrome of the received word after n-i-k shifts is

$$\begin{aligned} \underline{S_{n-i-k}} = \underline{r}\, \underline{\pi}^{n-i-k} H^T &= \underline{f}\, \underline{\pi}^{n-i-k} + \underline{e}\, \pi^{n-i-k} H^T \\ &= \underline{e}\, \underline{\pi}^{n-i-k} H^T \\ &= (e_{i+k}, e_{i+k+1}, \ldots, e_{n-1}, e_0, e_1, \ldots, e_{i-1}) \end{aligned} \tag{5.9}$$

Thus the new syndrome exhibits the permuted error pattern.

Therefore, whenever the received code word contains k or more consecutive error-free digits there exists at least one syndrome of permutation which exhibits exactly the erronerous digit positions.

Let $\underline{e}$ contain e random errors with $e \leq t$. Then it can be shown that the weight of the syndrome $\underline{s} = \underline{e}\, H^T$, denoted by $W(\underline{s})$, is either:

$$\left.\begin{array}{lll} & \text{(a)} & W(\underline{s}) = e \\ \text{or} & \text{(b)} & W(\underline{s}) \geq d-e, \end{array}\right\} \qquad (5.10)$$

where d is the minimum weight of non-zero code words ($d \geq 2t+1$ for a t-error-correcting code). To show the validity of (a) and (b), note that $\underline{e} - \underline{s}'$ is a code word*, where $\underline{s}'$ is the vector derived from $\underline{s}$ by adding k zeros to the right of $\underline{s}$, i.e., if

$\underline{s} = (s_0, s_1, \ldots, s_{n-k-1})$, $\underline{s}' = (s_0, s_1, s_2, \ldots, s_{n-k-1}, 0, 0, \ldots, 0)$.

Hence

$$W(\underline{e} - \underline{s}') \leq W(\underline{e}) + W(\underline{s}') = W(\underline{e}) + W(\underline{s}). \qquad (5.11)$$

But $\underline{e} + \underline{s}'$ is a code word,

$$W(\underline{e} - \underline{s}') = 0 \text{ or } \geq d. \qquad (5.12)$$

---

*Here $\underline{s}' = \underline{\underline{s}, \underline{0}}$ and $H^T = \begin{bmatrix} I \\ P \end{bmatrix}$, $\underline{s}' H^T = \underline{s}\, I + \underline{0}\, P = \underline{s}$.

Also, $\underline{e}\, H^T = \underline{s}$, $[\, \underline{e} - \underline{s}'\,]\, H^T = \underline{0}$, $\underline{e} - \underline{s}'$ is a code word.

From the above discussion and Equations (5.10), (5.11), we have either (a) or (b) of Equation (5.10), where

(a) occurs if and only if the last k digits of $\underline{r}$ is free of error, and

(b) occurs if and only if at least one error occurs in the last k digits.

The decoding procedure just described can be summarized in the following 3 steps.

(1) Feed the received vector $\underline{r}$ into the syndrome calculator and count the weight of the syndrome $W(\underline{s})$. If $W(\underline{s}) \leq t$, we correct the received vector to obtain a code word vector as

$$\underline{f} = \underline{r} - \underline{s'}$$

where $\underline{s'}$ has been described in the previouw paragraph. If $W(\underline{s}) > t$, go to Step (2).

(2) Feed the shifted vector $\underline{r}\,\pi_c$ into the syndrome calculator. Then go through Step (1) for the new syndrome $\underline{s_1} = \underline{r}\,\pi_c H^T$.
If $W(\underline{s_1}) > t$, try $\underline{r}\,\pi_c^i$, for $i = 2,3,\ldots,n-1$, successively. Stop the process whenever $W(\underline{s_i}) \leq t$. The permuted error pattern is exhibited by $\underline{s_i}$, and the corrections may be made.

This decoding scheme is capable of correcting any pattern of errors less than n/k, since if $e < n/k$, there exists at least one run of k consecutive error-free digits in the received vector $\underline{r}$.

(3) For the range of e, $t \geq e \geq \frac{n}{k}$, we require that there exists at least one run of consecutive k error-free digits in the received code word. Otherwise, some auxiliary means should be used in order to correct all the error patterns with t or less errors.

A further thought on Step (2) will help a great deal in simplifying the circuit. Let r(x) be the corresponding polynomial of degree n-1 to the received vector $\llcorner r \lrcorner$.

Then $r(x)\, x^i (\text{mod } x^n-1)$ is the corresponding polynomial for $\llcorner r \lrcorner\, \pi_c^i$. And let g(x) of degree n-k be the generator polynomial of the systematic code. The syndrome s(x) is given as

$$r(x) \equiv q_1(x)\, g(x) + s(x),\ \text{mod}(x^n-1), \tag{5.13}$$

where

$$\text{degree } s(x) < n-k \tag{5.14}$$

and

$$\begin{aligned} r(x)\, x^i &\equiv q_1(x)\, g(x)\, x^i + s(x)\, x^i,\ \text{mod}(x^n-1) \\ &\equiv q_2(x)\, g(x) + q_3(x)\, q(x) + s_1(x),\ \text{mod}(x^n-1), \end{aligned} \tag{5.15}$$

where

$$q_2(x) = x^i q_1(x) \tag{5.16}$$

and

$$s(x)\, x^i \equiv q_3(x)\, q(x) + s_1(x), \text{mod}(x^n-1) \tag{5.17}$$

with

$$\text{degree } s_1(x) < n-k\ . \tag{5.18}$$

It is clear that $s_1(x)$ is the syndrome for $r(x)\, x^i$ and $s_1(x)$ is also the remainder after dividing $s(x)\, x^i$ by g(x). In other words, $s_1(x)$ can be obtained by shifting s(x) i times in a divider connected according to g(x) with s(x) as remainder in the shift registers. Hence the decoding procedure can be revised as follows:

(1) Feed the received vector $\underline{r}$ into the syndrome calculator connected according to $g(x)$ and count the weight of the syndrome $W(\underline{s})$. If $W(\underline{s}) \leq t$, we correct the received vector to obtain a code word vector as

$$\underline{f} = \underline{r} - \underline{s'} .$$

If $W(\underline{s}) > t$, go to Step (2).

(2) Disconnect the input and shift the remainder $\underline{s}$ in the shift-registers. And for each shift we count the weight of the shifted new syndrome. Stop the shifting whenever $W(\underline{s_i}) \leq t$. The correct code word is then given by

$$\underline{r} - \underline{\underline{s_i}\ \underline{0}}\,\pi_c^i .$$

(3) Call for an auxiliary decoding scheme if $W(\underline{s_i}) > t$ for $i = 1,2,\ldots,n$.

5.2 Decoding of the Binary 3-Error-Correcting (15,5) B-C-H Code

The binary 3-error-correcting (15,5) B-C-H code contains $2^5 = 32$ code words, and has a minimum distance of 7. Hence it is capable of correcting any error pattern consisting of 3 or fewer random errors. A detailed decoding procedure of such a code is given as follows: Let the generator polynomial of the (15,5) code be

$$\begin{aligned} g(x) &= (x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1) \\ &= x^{10}+x^8+x^5+x^4+x^2+x+1. \end{aligned} \tag{5.19}$$

The parity-check matrix H and the generator matrix G are then

$$H = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&1&0&0&0&0&0&0&0&0&1&1&1&1&1 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&1&0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&1 \end{bmatrix} \tag{5.20}$$

and

$$G = \begin{bmatrix} 1&1&1&0&1&1&0&0&1&0&1&0&0&0&0 \\ 0&1&1&1&0&1&1&0&0&1&0&1&0&0&0 \\ 1&1&0&1&0&1&1&1&1&0&0&0&1&0&0 \\ 0&1&1&0&1&0&1&1&1&1&0&0&0&1&0 \\ 1&1&0&1&1&0&0&1&0&1&0&0&0&0&1 \end{bmatrix} . \tag{5.21}$$

Since $x^4+x^3+x^2+x+1$ is a factor of $g(x)$, any code word satisfying the parity-check matrix H of Equation (5.20) should also satisfy the following matrix H' corresponding to $x^4+x^3+x^2+x+1$.

$$H' = \begin{bmatrix} 1&0&0&0&1&1&0&0&0&1&1&0&0&0&1 \\ 0&1&0&0&1&0&1&0&0&1&0&1&0&0&1 \\ 0&0&1&0&1&0&0&1&0&1&0&0&1&0&1 \\ 0&0&0&1&1&0&0&0&1&1&0&0&0&1&1 \end{bmatrix} . \tag{5.22}$$

The decoding procedure can now be listed as follows.

(1) Feed the received vector into the syndrome calculator as shown in Fig. 5.1. If the weight indicator shows that the weight, i.e., the total number of 1's of the syndrome is zero, then there is no error in the received vector. If the weight shown is no more than 3, there is no error in the first k = 5 information digits. Thus the errors are in the check digits and no correction is required.

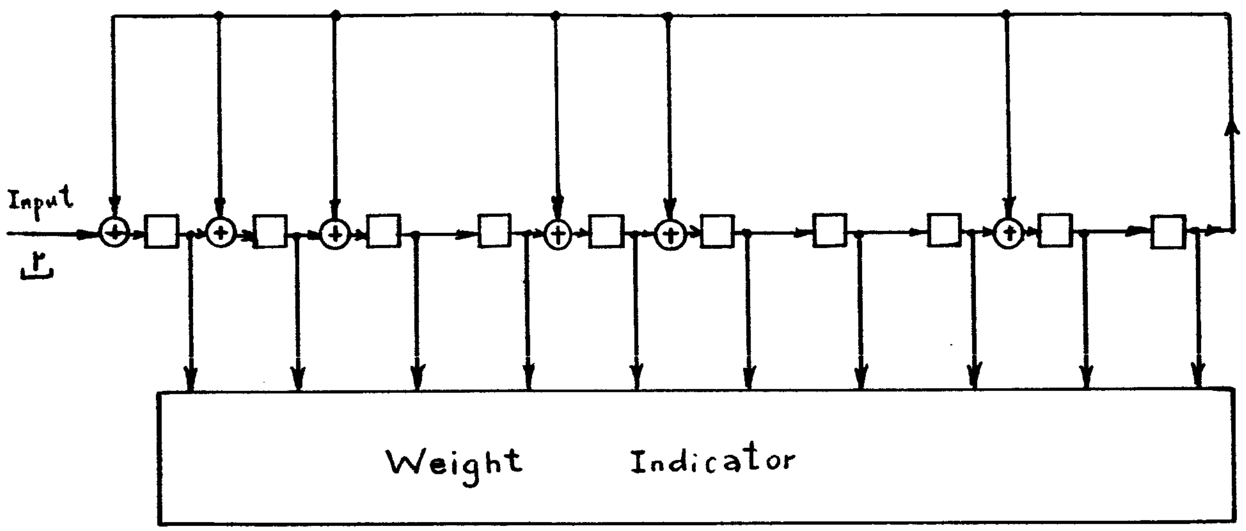


Fig. 5.1 Syndrome Calculator

(2) If the weight of the syndrome is greater than 3, there is at least one error in the first k = 5 information digits. Allow the syndrome to shift by itself with no further input until at the ith shift the weight indicator shows a weight of 3 or less. Then take this shifted "syndrome", denoted by $\underline{S_i}$, and form $\underline{e'} = \underline{\underline{S_i}\ 0\ 0\ 0\ 0\ 0}$. The true error pattern can be obtained by shifting e' backward i times.

(3) If the weight of each of the 15 shifted "syndromes" is greater than three, we know there are exactly three errors, evenly spaced. Then we feed the received code vector into the auxiliary "subsyndrome" calculator shown in Fig. 5.2 whose parity-check matrix

is given by Equation (5.22).

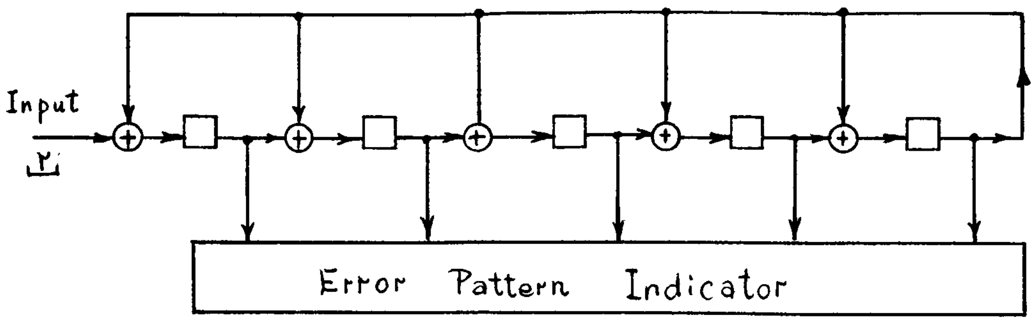


Fig. 5.2 The Auxiliary Subsyndrome Calculator

The error pattern indicator "transforms" the subsyndromes into the error patterns in the following manner:

$$(1000) \Rightarrow (10000\ 10000\ 10000)$$
$$(0100) \Rightarrow (01000\ 01000\ 01000)$$
$$(0010) \Rightarrow (00100\ 00100\ 00100)$$
$$(0001) \Rightarrow (00010\ 00010\ 00010)$$
$$(1111) \Rightarrow (00001\ 00001\ 00001).$$

The following examples will help to clariy the decoding procedure.

<u>Example 1</u>

Let $\underline{r}$ = (0 1 1 0 1 0 0 0 0 0 1 0 0 0 0) be the received vector. Then the syndrome $\underline{s}$ is

$$\underline{s} = \underline{r}\, H^T = (1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0).$$

The weight of the syndrome

$$W[\underline{s}] = 3.$$

Hence the information digits (10000) are correct. No correction is required.

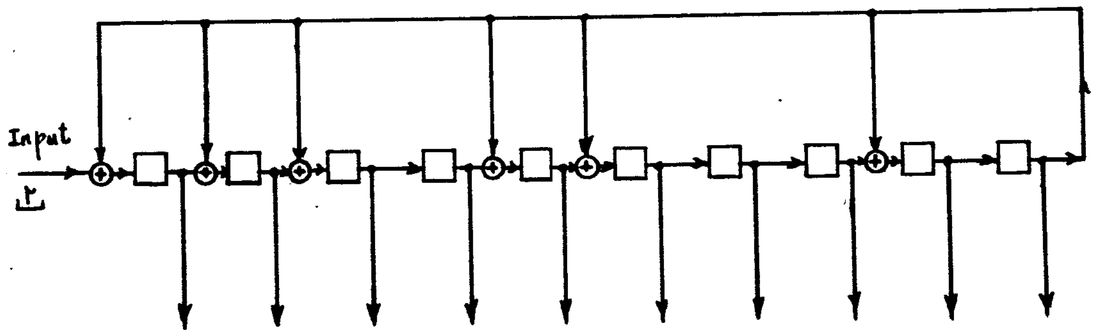
<u>Example 2</u>

$$\underline{r} = (0\;1\;1\;0\;0\;1\;1\;0\;0\;1\;0\;1\;1\;0\;0)$$

$$\underline{s} = \underline{r}\,H^T = (1\;1\;0\;0\;0\;1\;1\;1\;1\;0)$$

$$W[\underline{s}] = 6 > 3.$$

Hence we should shift the syndrome in the syndrome calculator as shown in Fig. 5.3.



											weight
	1	1	0	0	0	1	1	1	1	0	6
1st shift	0	1	1	0	0	0	1	1	1	1	6
2nd shift	1	1	0	1	1	1	0	1	0	1	7
3rd shift	1	0	0	0	0	0	1	0	0	0	$\boxed{2}$

Fig. 5.3 Cyclic Shifts of a Syndrome

Therefore,

$$\underline{S_3} = (1\;0\;0\;0\;0\;0\;1\;0\;0\;0), \text{ since } W[\underline{S_3}] = 2 < 3$$

and

$$\underline{e'} = (\underbrace{1\;0\;0\;0\;0\;0\;1\;0\;0\;0}_{\underline{S_3}}\;0\;0\;0\;0\;0)$$

$$\underline{e} = 3 \text{ backward shifts of } \underline{e'}$$

$$= (0\;0\;0\;1\;0\;0\;0\;0\;0\;0\;0\;0\;1\;0\;0).$$

Hence the correct code word $\underline{f}$ is

$$\underline{f} = \underline{r} + \underline{e}$$
$$= (0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0).$$

Example 3

$$\underline{r} = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)$$
$$\underline{s} = \underline{r}\, H^T = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$$
$$W[\underline{s}] = 7 > 3.$$

The weight of each of the 15 shifts of the syndrome is greater than 3. Hence, upon trying $\underline{s} = \underline{r}\, H' = (0\ 0\ 1\ 0)$, we know the error pattern should be

$$\underline{e} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0)$$
$$\underline{f} = \underline{r} + \underline{e} = (1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0).$$

5.3 Remarks

The cyclic permutation decoding procedure is particularly suitable for high-redundancy short cyclic codes and for B-C-H codes generated by non-primitive elements. The following table shows the number of random errors correctable by the permutation decoding scheme, e_c, and the code error-correctability, t, of some of this class of binary Bose-Chaudhuri-Hocquenghem codes.

n	k	t	e_c	$t-e_c$
21	12	2	1	1
	6	3	3	0
	4	4	4	0
23	12	3	1	2

n	k	t	e_c	$t-e_c$
25	5	2	2	0
27	3	4	4	0
33	13	2	2	0
35	11	2	2	0
	8	3	3	0
	4	7	7	0
39	15	3	2	1
	3	6	6	0
43	15	3	2	1
45	29	2	1	1
	23	3	1	2
	11	4	4	0
	7	7	6	1
	5	10	8	2
47	24	2	1	1
49	7	3	3	0
	4	10	10	0
51	35	2	1	1
	27	4	1	3
	19	5	2	3
	11	8	4	4
	9	5	5	0

n	k	t	e_c	$t-e_c$
55	15	2	2	0
	5	5	5	0
57	21	2	2	0
	3	9	9	0
65	41	2	1	1
	29	3	2	1
	17	5	3	2
	5	6	6	0
69	36	2	1	1
	14	7	4	3
	3	11	11	0
71	36	3	1	2
73	55	2	1	1
	46	4	1	3
	37	5	1	4
	28	6	2	4
	19	8	3	5
	10	12	7	5 .

MacWilliams[31] also has suggested the following permutation transform

$$U: \quad i \rightarrow qi \quad \text{for } i = 1,2,\ldots,n-1,$$

where i is the position index of the component a_i of a code word

$$(a_0, a_1, \ldots, a_i, \ldots, a_{n-1}) \text{ over } GF(q).$$

With this permutation, some of the cyclic codes are capable of correcting all the correctable error patterns. But the implementation of such a permutation is not simple and for each U-transform of the received sequence it is required to repeat the three decoding steps mentioned in Section 5.1. Therefore, we suggest combining the technique described in Section 5.1 with other decoding schemes rather than going through the U-transform.

The cyclic permutation combined with the U-transform fails to decode completely the binary 3-error-correcting (15,5) B-C-H code as described in Section 5.2.

In certain applications, it may be desirable to have an extremely simply decoding procedure which corrects a major fraction of theoretically correctable error patterns and sounds an alarm whenever the error pattern falls outside of that major fraction. The permutation decoding procedure is particularly suitable for such an applications. Take an error-locating code for example. If we take the binary 5-error-correcting B-C-H (31,11) code as the first component code, then permutation decoding is capable of correcting two errors which corresponds to two subblocks in the error-locating code. Therefore, if there are two or fewer subblocks in error, we locate them and either consider such errors as tolerable or ask for retransmission of these two subblocks only.

If there are more than two corrupted subblocks, we may not be able to locate erroneous subblocks by the permutation method and in this case we ask for retransmission of the entire code block regardless of whether we may be able to locate the erroneous subblocks by other methods.

In general, it can be shown that a t-error-correcting code with minimum distance $d > 2t$ is capable of correcting errors of magnitude $e_c < t$ and simultaneously detect errors of magnitude less than or equal to $d-e_c-1 > t$. Hence, a t-error-correcting code with minimum distance $d = 2t + 1$ can perform one of the following functions:

(1) detecting 2t errors;

(2) correcting $e_c \left(\leq \frac{n-1}{k}\right)$ errors and simultaneously detecting $2t-e_c$ errors;

(3) correcting t errors.

CHAPTER 6

MISCELLANEOUS RESULTS AND CONCLUSIONS

In this chapter the results of Chapter 2 are used to find the minimum distance of the product code specified by $H = H_1 \otimes H_2$. Suitable channels for employing product codes are discussed. Conclusions and recommendations for future research are presented.

6.1 Minimum Distances of Product Codes

It has been shown* that the minimum distance of a product code generated by $G = G_1 \otimes G_2$ is the product of the minimum distances of its component codes. We now proceed to find the minimum distance for the product code specified by $H = H_1 \otimes H_2$. The following lemma is useful in determining the minimum distance for such a product code.

Lemma 6.1

> Let V be a subspace of the space containing all n-tuples over $GF(q)$. Let $\overline{V}$ be the complementary space of V, i.e., $\overline{V} \oplus V =$ whole space of n-tuples over $GF(q)$. Also let $v \in V$, $\overline{v} \in \overline{V}$, and both v and $\overline{v}$ be non-zero vectors. Then $d[v, \overline{v}] \geq 1$, where $d[v, \overline{v}]$ denotes the distance between v and $\overline{v}$.

*See Reference 2, Chapter 5, Theorem 5.3.

Proof

Suppose $d[v, \bar{v}] = 0$, then $v = \bar{v} \in V \cap \bar{V} = 0$, by the definition of the direct sum. This is contradictory to the assumption that $v \neq 0$ and $\bar{v} \neq 0$, therefore, we conclude that $d[v, \bar{v}] \geq 1$. Q.E.D.

Theorem 6.1

The minimum distance, d, of a product code specified by $H = H_1 \otimes H_2$ is given by $d = \min[d_1, d_2]$, where d_1 is the minimum distance of the first component code and d_2 the minimum distance of the second component code.

Proof

By Theorem 2.6, the generator matrix of the product code is given by

$$G = \begin{bmatrix} G_1 \otimes G_2 \\ \bar{G}_1 \otimes G_2 \\ G_1 \otimes \bar{G}_2 \end{bmatrix},$$

which can be transformed by elementary row operations into the following form:

$$G = \left[\begin{array}{c} I_{n_1} \otimes G_2 \\ \hdashline G_1 \otimes \bar{G}_2 \end{array}\right].$$

Without loss of generality, let us assume

$$G_2 = [P \,\vdots\, I_{k_2}],$$

then it can be easily shown that one can choose the complementary space of G_2 as row space of

$$\bar{G}_2 = [I_{n_2-k_2} \; 0].$$

Let $\underbrace{g_2}^{i}$ be any code word of the second component code. Then the code word generated by the generator matrix $I_{n_1} \otimes G_2$ is of the form

$$\underbrace{\underbrace{g_2}^{1} \underbrace{g_2}^{2}, \ldots, \underbrace{g_2}^{n_1}}.$$

The minimum distance of such code word is d_2 and the minimum weight of each subblock is either 0 or d_2. Let $\underbrace{\overline{g}_2}$ be a code word generated by the generator matrix $\overline{G}_2$. Then the code word generated by the generator matrix $G_1 \otimes \overline{G}_2$ is

$$\underbrace{g_{11} \underbrace{\overline{g}_2}, \; g_{12} \underbrace{\overline{g}_2}, \ldots, g_{1n_1} \underbrace{\overline{g}_2}},$$

where $(g_{11}, g_{12}, \ldots, g_{1n_1})$ is a code word of the first component code. The minimum distance of the code words generated by $G_1 \otimes \overline{G}_2$ is $d_1 \times 1 = d_1$. Hence the minimum distance d of the code word generated by G is $d \geq \min[d_1, d_2]$.

We now need to show that a linear combination of code words from $I_{n_1} \otimes G_2$ and from $G_1 \otimes \overline{G}_2$ has a weight greater than $\min[d_1, d_2]$. This is true, since the combined code word is of the form

$$\underbrace{a \underbrace{g_2}^{1} + b\, g_{11} \underbrace{\overline{g}_2}, \; a \underbrace{g_2}^{2} + b\, g_{12} \underbrace{\overline{g}_2}, \ldots, a \underbrace{g_2}^{n_1} + b\, g_{1n_1} \underbrace{\overline{g}_2}},$$

where a and b are non-zero field elements.

For each $g_{1i} = 0$:

$$\begin{aligned} w &\equiv W[a_1 \underbracket{g_2^i} + b_1 g_{1i} \underbracket{\bar{g}_2}] \\ &= W[a_1 \underbracket{g_2^i}] \\ &= W[\underbracket{g_2^i}]. \end{aligned}$$

When $g_{1i} \neq 0$, we have the following 2 cases:

(1) $W[\underbracket{g_2^i}] = 0$

$w = W[\underbracket{\bar{g}_2}] \geq 1$, since $W[\underbracket{\bar{g}_2}] \neq 0$.

(2) $W[\underbracket{g_2^i}] \neq 0$

$w = \min d[\underbracket{g_2^i}, \underbracket{\bar{g}_2}] \geq 1$,

by Lemma 6.1.

There are at least d_1 g_{1i}'s which are not zero, hence the minimum distance is larger than d_1.

Therefore,

$$d = \min[d_1, d_2]. \qquad \text{Q.E.D.}$$

Even though the minimum distance of a product code specified by $H = H_1 \otimes H_2$ is equal to the smaller minimum distance of its two component codes, the error-control capability of the product code is far superior to that of its component codes. Therefore, the minimum distance of a code is not always a good criterion in itself for judging the code capability.

6.2 Suitable Channels for Employing Product Codes

In general, a product code is suitable for a certain communication channel whenever its component codes are the proper codes for this particular channel. The reason for choosing product codes in such a channel is that we can have more varieties of code lengths and error control capabilities at our command.

The product codes generated by $G = G_1 \otimes G_2$ are particularly suitable for the channels where random errors and burst errors may occur simultaneously, while the product codes specified by $H = H_1 \otimes H_2$ and $H^1 = [H_{1q} \otimes H_{2q}]_p$ are appropriate for the channels where errors tend to cluster in subblocks. For the channels with multi-zone detection, the error-locating codes whose parity-check matrices are given by $H = H_1 \otimes H_2$ or $H^1 = [H_{1q} \otimes H_{2q}]_p$ are particularly suitable for error-control. The multi-zone detection scheme indicates the possible erroneous digits throughout the entire code word block, while the error-locating decoder will locate the subblocks containing errors. Hence those digits indicated as erroneous by the former scheme which lie in the subblocks located by the latter technique are the highly questionable digits. In this way we may narrow the erroneous "area" from a whole subblock to several digits.

6.3 Concluding Remarks

The techniques of finding the null space of the tensor product space are developed in Chapter 2. This algebraic approach yields the results which are expressed as much as possible in terms of the null spaces of the original "component" spaces or their bases. This results in better insight of the product code structure and the structure of its null space. It also enables us to perform further research in determining the properties of product codes, such as weight distribution, possibility of being comma free, etc. The investigation of the decomposibility of a code into two component codes whose tensor product gives the original code, relies heavily on the understanding of the null space of a tensor product space.

The decomposition of the encoding and decoding procedures of a product code into those of its component codes, and the capabilities of product codes, are discussed in Chapter 3. It is also shown in this chapter that the implementation of a product code can be achieved by a simple logic combination of those of its component codes. All the discussions in Chapter 3 are kept as general as possible and are applicable to any product code with linear component codes.

In Chapter 4, the product codes with cyclic component codes are discussed. Since cyclic codes can be most simply implemented, the encoder and syndrome calculator of a cyclic code can be easily converted to those of its dual code by switching, the implementation of a product code with cyclic component codes is simpler than that of other product codes. The encoder and syndrome calculator of such a product code provide eight operating modes, thus providing a highly flexible communication system.

A simple decoding scheme for the systematic cyclic code, namely permutation decoding, is discussed in Chapter 5. This simple scheme is capable of correcting a large fraction of the correctable errors of the code. It is suggested that an auxiliary decoding scheme be used whenever permutation decoding fails to correct errors. A simple and straightforward method of decoding the binary (15,5) B-C-H code is illustrated as an example.

In the last chapter, the minimum distances of product codes and suitable channels for employing the product codes are discussed. An important result obtained from consideration of the minimum distances of product codes is that the minimum distance alone is not always a good criterion for judging the code capability.

6.4 Recommended Further Research

The following are a direct extension and promising areas for further research concerning product codes:

(i) Polynomial description of product codes with cyclic component codes. Some of the work has been done by Yüill[34] and Burton and Weldon[27], but extension of their work is necessary in order to have a better understanding of product codes.

(ii) Calculation of probabilities of error in data transmission of tensor product codes under various rates of transmission.

(iii) Calculation of the weight distribution of product codes and determination of its relation to those of the component codes.

(iv) Study of the code or codes which can be iterated in a product manner indefinitely with a non-zero data transmission rate.

(v) Searching for different ways of combining two or more codes to form new codes.

APPENDIX

THE DETERMINATION OF NULL SPACE OF A TENSOR PRODUCT SPACE INVOLVING THE TRANSLATION OF FIELDS

The preliminary lemmas and theorems lead to Theorem 2.7 of Chapter 2 are given in this appendix. The results are extended somewhat beyond the case that the second component code is cyclic.

Let r(x) be a monic primitive irreducible polynomial of degree ℓ over GF(q), where $q = p^\ell$, and let β^i, for $i = 0,1,2,\ldots,q-1$, be non-zero elements of GF(q). The p-ary representations of the β^i's are given as

$$\left.\beta^0\right] = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \quad \left.\beta^1\right] = \left.T_0\beta^0\right], \quad \left.\beta^2\right] = \left.T_0\beta^1\right] = \left.T_0^2\beta^0\right]$$

$$\ldots, \beta^i = \left.T_0^i\beta^0\right], \ldots \tag{A-1}$$

and

$$\left.\beta^{k+i}\right] = \left.T_0^i\beta^k\right] = \left.T_0^k\beta^i\right]$$

$$\text{for all } k \text{ and } i \bmod(p^\ell-1). \tag{A-2}$$

T_c has the similar property if the p-ary representations of β^i's are given as row vectors $\underline{\beta^i}$'s.

A matrix A_p with elements in GF(p) is said to be a p-ary representation of a matrix A_q with elements in GF(q), $q = p^\ell$, if A_p is obtained from A_q by expressing each of its elements as a column vector with ℓ p-ary components.

Lemma A.1

Let D_p and D_p' be the p-ary representations of D_q and $\beta^i D_q$ respectively, where β^i is a non-zero element of $GF(q)$, $q = p^{\ell}$. Then

$$D_p' E_p^T = 0 \text{ for all } E_p \text{ such that}$$

$$D_p E_p^T = 0.$$

Proof

Let

$$D_q = \begin{bmatrix} (b_{11}\beta^{a_{11}}) & (b_{12}\beta^{a_{12}}) & \cdots & (b_{1n}\beta^{a_{1n}}) \\ (b_{21}\beta^{a_{21}}) & (b_{22}\beta^{a_{22}}) & \cdots & (b_{2n}\beta^{a_{2n}}) \\ \vdots & \vdots & & \vdots \\ (b_{h1}\beta^{a_{h1}}) & (b_{h2}\beta^{a_{h2}}) & \cdots & (b_{hn}\beta^{a_{hn}}) \end{bmatrix}, \qquad \text{(A-3)}$$

where $\beta^{a_{ij}}$'s are the non-zero elements of $GF(q)$, and $b_{ij} = \beta^0$ or 0. Then

$$\beta^i D_q = \begin{bmatrix} (b_{11}\beta^{a_{11}+i}) & (b_{12}\beta^{a_{12}+i}) & \cdots & (b_{1n}\beta^{a_{1n}+i}) \\ (b_{21}\beta^{a_{21}+i}) & (b_{22}\beta^{a_{22}+i}) & \cdots & (b_{2n}\beta^{a_{2n}+i}) \\ \vdots & \vdots & & \vdots \\ (b_{h1}\beta^{a_{h1}+i}) & (b_{h2}\beta^{a_{h2}+i}) & \cdots & (b_{hn}\beta^{a_{hn}+i}) \end{bmatrix} \qquad \text{(A-4)}$$

The p-ary representations of matrices (A-3) and (A-4) are

$$D_p = \begin{bmatrix} [b_{11}\beta^{a_{11}}] & [b_{12}\beta^{a_{12}}] & \cdots & [b_{1n}\beta^{a_{1n}}] \\ [b_{21}\beta^{a_{21}}] & [b_{22}\beta^{a_{22}}] & \cdots & [b_{2n}\beta^{a_{2n}}] \\ \vdots & \vdots & & \vdots \\ [b_{h1}\beta^{a_{h1}}] & [b_{h2}\beta^{a_{h2}}] & \cdots & [b_{hn}\beta^{a_{hn}}] \end{bmatrix}, \tag{A-5}$$

$$D_p' = \begin{bmatrix} [T_0{}^i b_{11}\beta^{a_{11}}] & [T_0{}^i b_{12}\beta^{a_{12}}] & \cdots & [T_0{}^i b_{1n}\beta^{a_{1n}}] \\ [T_0{}^i b_{21}\beta^{a_{21}}] & [T_0{}^i b_{22}\beta^{a_{22}}] & \cdots & [T_0{}^i b_{2n}\beta^{a_{2n}}] \\ \vdots & \vdots & & \vdots \\ [T_0\ b_{h1}\beta^{a_{h1}}] & [T_0\ b_{h2}\beta^{a_{h2}}] & \cdots & [T_0\ b_{hn}\beta^{a_{hn}}] \end{bmatrix}^*$$

*T_0 is the companion matrix of the primitive polynomial which generates the fields containing β^i's. The matrix T_0 is given in Equation (2.27).

or

$$D_p' = \begin{bmatrix} T_0{}^i\left[[b_{11}\beta^{a_{11}}] & b_{12}\beta^{a_{12}}] & \cdots & b_{1n}\beta^{a_{1n}}] \right] \\ T_0{}^i\left[[b_{21}\beta^{a_{21}}] & b_{22}\beta^{a_{22}}] & \cdots & b_{2n}\beta^{a_{2n}}] \right] \\ \vdots & \vdots & & \vdots \\ T_0\left[[b_{h1}\beta^{a_{h1}}] & b_{h2}\beta^{a_{h2}}] & & b_{hn}\beta^{a_{hn}}] \right] \end{bmatrix} \quad \text{(A-6)}$$

The statement that $D_pE_p{}^T = 0$ is equivalent to

$$\left[[b_{j1}\beta^{a_{j1}}] \quad b_{j2}\beta^{a_{j2}}] \cdots b_{jn}\beta^{a_{jn}}] \right] E_p{}^T = 0$$

for $j = 1,2,3,\ldots,h$.

Furthermore,

$$T_0{}^i\left[[b_{j1}\beta^{a_{j1}}] \quad b_{j2}\beta^{a_{j2}}] \cdots b_{jn}\beta^{a_{jn}}] \right] E_p{}^T = 0$$

for $j = 1,2,3,\ldots,h$.

Consequently, we have

$$D_p'E_p{}^T = 0. \qquad \text{Q.E.D.}$$

In coding terminology, we say that if a gp matrix D_p is the p-ary representation of β^iD_q over GF(q), where β^i is any non-zero element of GF(q) and $q = p^\ell$, then the code space D_p is invariant under the changing of the element β^i.

Theorem A-1

Let D_{1q} and D_{2q} be gp matrices over GF(q), $q = p^{\ell}$, of sizes $h_1 \times n_1$ and $h_2 \times n_2$ respectively. Let D_{2p} be the p-ary representation of D_{2q}, and let the row spaces of E_{2p} be the null space of D_{2p}, i.e., $D_{2p}E_{2p}^T = 0$. If D_p is the p-ary representation of $D_q = D_{1q} \otimes D_{2q}$, then the row space of $I_{n_1} \otimes E_{2p}$ over GF(p) is a subspace of the null space of the matrix D_p.

Proof

By hypothesis

$$D_{2p}E_{2p}^T = 0. \tag{A-7}$$

We need only to show

$$D_p(I_{n_1} \otimes E_{2p})^T = 0.$$

But

$$D_q = D_{1q} \otimes D_{2q}$$

$$= \begin{bmatrix} a_{11}D_{2q} & a_{12}D_{2q} & \cdots & a_{1n_1}D_{2q} \\ a_{21}D_{2q} & a_{22}D_{2q} & \cdots & a_{2n_1}D_{2q} \\ \vdots & \vdots & & \vdots \\ a_{h_11}D_{2q} & a_{h_12}D_{2q} & \cdots & a_{h_1n_1}D_{2q} \end{bmatrix} \tag{A-8}$$

where the a_{ij}'s are elements of GF(q).

From Equation (A-8), we can visualize that D_p contains $h_1 \times n_1$ submatrices. Each of them is either a p-ary representation of $a_{ij}D_{2p}$ or a 0-matrix. Let D_{ijp} =

p-ary representation of $a_{ij}D_{2q}$, which is a zero matrix if and only if $a_{ij} = 0$, since we assume D_{2q} is not a zero matrix. Hence,

$$D_p = \begin{bmatrix} D_{11p} & D_{12p} & \cdots & D_{1n_1p} \\ D_{21p} & D_{22p} & \cdots & D_{2n_1p} \\ \vdots & \vdots & & \vdots \\ D_{h_11p} & D_{h_12p} & \cdots & D_{h_1n_1p} \end{bmatrix} \tag{A-9}$$

and

$$D_p(I_{n_1} \otimes E_{2p})^T$$

$$= D_p(I_{n_1} \otimes E_{2p}^T)$$

$$= \begin{bmatrix} D_{11p} & D_{12p} & \cdots & D_{1n_1p} \\ D_{21p} & D_{22p} & \cdots & D_{2n_1p} \\ \vdots & \vdots & & \vdots \\ D_{h_11p} & D_{h_12p} & \cdots & D_{h_1n_1p} \end{bmatrix} \underbrace{\begin{bmatrix} E_{2p}^T & & & \\ & E_{2p}^T & & \bigcirc \\ & & \ddots & \\ \bigcirc & & & E_{2p}^T \end{bmatrix}}_{n_1} \left.\vphantom{\begin{bmatrix} E \\ E \\ \ddots \\ E \end{bmatrix}}\right\} n_1$$

$$= \begin{bmatrix} D_{11p}E_{2p}^T & D_{12p}E_{2p}^T & \cdots & D_{1n_1p}E_{2p}^T \\ D_{21p}E_{2p}^T & D_{22p}E_{2p}^T & \cdots & D_{2n_1p}E_{2p}^T \\ \vdots & \vdots & & \vdots \\ D_{h_11p}E_{2p}^T & D_{h_12p}E_{2p}^T & & D_{h_1n_1p}E_{2p}^T \end{bmatrix}$$

$$= 0 ,$$

since, by Lemma A-1

$$D_{ijp}E_{2p}^T = 0$$

$$\text{for } i = 1,2,\ldots,h_1$$
$$j = 1,2,\ldots,n_1 \;.$$ Q.E.D.

<u>Theorem A-2</u>

Let D_{1q} and D_{2q} be the gp matrices over GF(q), $q = p^{\ell}$, of sizes $h_1 \times n_1$ and $h_2 \times n_2$ respectively. Let $D_{1p} = [D_{1q}]_p$ = the p-ary representation of D_{1q}, $D_{2p} = [D_{2q}]_p$, and the row space of E_{1p} is the null space of D_{1p}, i.e., $D_{1p}E_{1p}^T = 0$. Then the row spaces of $E_{1p} \otimes I_{n_2}$ is a subspace of the null space of $D_p = [D_{1q} \otimes D_{2q}]_p$.

<u>Proof</u>

Let

$$D_{1q} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n_1} \\ a_{21} & a_{22} & \cdots & a_{2n_1} \\ \vdots & \vdots & & \vdots \\ a_{h_11} & a_{h_12} & \cdots & a_{h_1n_1} \end{bmatrix} \qquad \text{(A-10)}$$

$$D_{2q} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n_2} \\ b_{21} & b_{22} & \cdots & b_{2n_2} \\ \vdots & \vdots & & \vdots \\ b_{h_21} & b_{h_22} & \cdots & b_{h_2n_2} \end{bmatrix} . \qquad \text{(A-11)}$$

Define $[A_{ij}]_{rs}$ as the $h_2 \times n_2$ matrix with only one non-zero element a_{ij}, positioned at the intersection of the rth row and sth column, e.g.

$$[A_{36}]_{24} = \underbrace{\left.\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \dots & 0 \\ 0 & 0 & 0 & a_{36} & 0 \dots & 0 \\ 0 & 0 & 0 & 0 & 0 \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0 \dots & 0 \end{bmatrix}\right\}}_{n_2 \text{ columns}} h_2 \text{ rows} \; . \tag{A-12}$$

Then

$$D_{1q} \otimes D_{2q} = \sum_{i=1}^{h_2} \sum_{j=1}^{n_2} b_{ij} \begin{bmatrix} [A_{11}]_{ij} & [A_{12}]_{ij} & \dots [A_{1n_1}]_{ij} \\ [A_{21}]_{ij} & [A_{22}]_{ij} & \dots [A_{2n_1}]_{ij} \\ \vdots & \vdots & \vdots \\ [A_{h_1 1}]_{ij} & [A_{h_1 2}]_{ij} & \dots [A_{h_1 n_1}]_{ij} \end{bmatrix} . \tag{A-13}$$

Note that the row space of

$$\left[b_{ij} \begin{bmatrix} [A_{11}]_{ij} & [A_{12}]_{ij} & \dots & [A_{1n_1}]_{ij} \\ [A_{21}]_{ij} & [A_{22}]_{ij} & \dots & [A_{2n_1}]_{ij} \\ \vdots & \vdots & & \vdots \\ [A_{h_1 1}]_{ij} & [A_{h_1 2}]_{ij} & \dots & [A_{h_1 n_1}]_{ij} \end{bmatrix} \right]_p$$

is exactly the same as the row space of

$$\left[b_{kj} \begin{bmatrix} [A_{11}]_{kj} & [A_{12}]_{kj} & \dots & [A_{1n_1}]_{kj} \\ [A_{21}]_{kj} & [A_{22}]_{kj} & \dots & [A_{2n_1}]_{kj} \\ \vdots & \vdots & & \vdots \\ [A_{h_1 1}]_{kj} & [A_{h_1 2}]_{kj} & \dots & [A_{h_1 n_1}]_{kj} \end{bmatrix} \right]_p$$

by Lemma A-1.

Hence,

$$\text{row space of } \left[D_{1q} \otimes D_{2q}\right]_p = \text{row space of} \sum_{j=1}^{n_2} \begin{bmatrix} [A_{11}]_{ij} & [A_{12}]_{ij} & \cdots & [A_{1n_1}]_{ij} \\ [A_{21}]_{ij} & [A_{22}]_{ij} & \cdots & [A_{2n_1}]_{ij} \\ \vdots & \vdots & & \vdots \\ [A_{h_1 1}]_{ij} & [A_{h_1 2}]_{ij} & \cdots & [A_{h_1 n_1}]_{ij} \end{bmatrix}_p \tag{A-14}$$

since at least there is one non-zero element in each column of D_{2q}.

Define $\underline{i}$ as the i^{th} row vector of I_{n_2}. It is evident that

$$\begin{bmatrix} [A_{11}]_{ij} & [A_{12}]_{ij} & \cdots & [A_{1n_1}]_{ij} \\ [A_{21}]_{ij} & [A_{22}]_{ij} & \cdots & [A_{2n_1}]_{ij} \\ \vdots & \vdots & & \vdots \\ [A_{h_1 1}]_{ij} & [A_{h_1 2}]_{ij} & \cdots & [A_{h_1 n_1}]_{ij} \end{bmatrix}_p \begin{bmatrix} \\ E_{1p} \otimes \underline{i} \\ \\ \end{bmatrix}^T = 0 \tag{A-15}$$

if $i \neq j$,

since the first matrix has non-zero elements only in column $j, 2j, \ldots, n_2 j$, while the j^{th}, $2j^{th}$, and $n_2 j^{th}$ columns of $E_{1p} \otimes \underline{i}$ are all zeros. For $i = j$

$$\begin{bmatrix} [A_{11}]_{ij} & [A_{12}]_{ij} & \cdots & [A_{1n_1}]_{ij} \\ [A_{21}]_{ij} & [A_{22}]_{ij} & \cdots & [A_{2n_1}]_{ij} \\ \vdots & \vdots & & \vdots \\ [A_{h_1 1}]_{ij} & [A_{h_1 2}]_{ij} & \cdots & [A_{h_1 n_1}]_{ij} \end{bmatrix}_p [E_{1p} \otimes \underline{j}]^T$$

$$= [D_{1p}][E_p]^T = 0. \tag{A-16}$$

Therefore,

$$[D_{1q} \otimes D_{2q}]_p \, [E_{1p} \otimes \underline{i}]^T = 0 \text{ for } i = 1,2,\ldots,n_2$$

which implies

$$[D_{1q} \otimes D_{2q}]_p \, [E_{1p} \otimes I_{n_2}]^T = 0. \qquad \text{(A-17)}$$

In other words, $E_{1p} \otimes I_{n_2}$ generates a subspace contained in the null space of $[D_{1q} \otimes D_{2q}]_p$. Q.E.D.

The results of Theorems A-1 and A-2 can be combined in the following theorem.

<u>Theorem A-3</u>

Let D_{1p} and D_{2p} be the p-ary representations of the gp matrices D_{1q} and D_{2q} of q-ary codes C_1 and C_2 respectively with $q = p^{\ell}$. The sizes of D_{1q} and D_{2q} are $h_1 \times n_1$ and $h_2 \times n_2$ respectively. If the null space of row space D_{1p} is specified by E_{1p} and that of D_{2p} by E_{2p}, then the row space of

$$\begin{bmatrix} E_{1p} \otimes I_{n_2} \\ I_{n_1} \otimes E_{2p} \end{bmatrix}$$

is a subspace of the null space of $D_p = [D_{1q} \otimes D_{2q}]_p$.

It can be easily shown that

$$\text{rank}[D_p] \le \ell h_1 h_2 \, . \qquad \text{(A-18)}$$

Hence the dimension of the null space of $D_p \ge n_1 n_2 - \ell h_1 h_2$. But from Theorem 2.6 and Corollary 2.6.1 we know that

$$\text{rank} \begin{bmatrix} E_{1p} \otimes I_{n_2} \\ I_{n_1} \otimes E_{1p} \end{bmatrix} = n_1 n_2 - (\ell h_1)(\ell h_2) = n_1 n_2 - \ell^2 h_1 h_2 \, . \qquad \text{(A-19)}$$

Therefore, the row space of

$$\begin{bmatrix} E_{1p} \otimes I_{n_2} \\ I_{n_1} \otimes E_{2p} \end{bmatrix}$$

is a subspace of the null space of D_p.

So far, no systematic method of evaluating the matrix whose row space is the complete null space of $D_p = [D_{1q} \otimes D_{2q}]_p$ has been developed.

Fortunately, the most useful codes whose gp matrices are the p-ary representations of the tensor products of two q-ary gp matrices are the codes with cyclic q-ary component codes. A cyclic p^r-ary code with s rows in its gp matrix is identical to a cyclic code of the same length with a one-rowed p^{ℓ}-ary gp matrix, $\ell = rs$, by suitably expressing elements in the p^{ℓ}-ary field in terms of vectors with components in the p^r-ary field. The following lemma and theorems are developed for the tensor product of cyclic component codes.

<u>Lemma A-2</u>

Let $r(x) = r_0 + r_1x + r_2x^2 + \dots + r_{\ell-1}x^{\ell-1} + x^{\ell}$ be a primitive irreducible polynomial over GF(p). Let T_0 be its companion matrix, then the following mapping

$$M: GF(p^{\ell}) \rightarrow F(A)$$

is a homomorphism, where F(A) contains all $\ell \times \ell$ square matrices over GF(p).

Proof

We define mapping S as

$$S: \quad GF(p^{\ell}) \rightarrow \text{p-ary representation}$$

$$S(0) = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

$$S\beta^{i} = \beta^{i}\Big] .$$

It is evident that S is an isomorphism. But

$$\beta^{i}\Big] = T_{o}{}^{i}\beta^{o}$$

and

$$\left(\beta^{i}\beta^{j}\right)\Big] = \beta^{i+j}\Big] = T_{o}{}^{i+j}\beta^{o}\Big] = T_{o}^{i}T_{o}^{j}\beta^{o}\Big]$$

$$\left(\beta^{i}+\beta^{j}\right)\Big] = \beta^{i}\Big] + \beta^{j}\Big] = T_{o}{}^{i}\beta^{o}\Big] + T_{o}{}^{j}\beta^{o}\Big] = \left(T_{o}{}^{i}+T_{o}{}^{j}\right)\beta^{o}\Big]$$

and

$$\begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = 0_{\ell\times\ell}\,\beta^{o}\Big] .$$

Define U such that $U\left(\beta^{i}\Big]\right) = T_{o}{}^{i}$, then M = US which is a homomorphism. Q.E.D.

Theorem A-4

Let D_{1q} and E_{1q} be the gp and pg matrices of a q-ary code, where $q = p^{\ell}$ and the sizes of D_{1q} and E_{1q} be $h \times n$ and $s \times n$ respectively, $h + s = n$. Then

$$\left[D_{1q} \otimes [\beta^0 \beta^1 \ldots \beta^{\ell-1}]\right]_p \left[E_{1q}^T \otimes [\beta^0 \beta^1 \ldots \beta^{\ell-1}]\right]_p = 0$$

where β^1 is a primitive element of GF(q).

Proof

Let

$$D_{1q} = \beta^{-1} \begin{bmatrix} b_{11}\beta^{a_{11}} & b_{12}\beta^{a_{12}} & \ldots & b_{1n}\beta^{a_{1n}} \\ b_{21}\beta^{a_{21}} & b_{22}\beta^{a_{22}} & \ldots & b_{2n}\beta^{a_{2n}} \\ \vdots & \vdots & & \vdots \\ b_{h1}\beta^{a_{h1}} & b_{h2}\beta^{a_{h2}} & \ldots & b_{hn}\beta^{a_{hn}} \end{bmatrix} \qquad \text{(A-20)}$$

where b_{ij} is either 0 or β^0, and let

$$E_{1q} = \beta^{-1} \begin{bmatrix} t_{11}\beta^{m_{11}} & t_{12}\beta^{m_{12}} & \ldots & t_{1n}\beta^{m_{1n}} \\ t_{21}\beta^{m_{21}} & t_{22}\beta^{m_{22}} & \ldots & t_{2n}\beta^{m_{2n}} \\ \vdots & \vdots & & \vdots \\ t_{s1}\beta^{m_{s1}} & t_{s2}\beta^{m_{s2}} & \ldots & t_{sn}\beta^{m_{sn}} \end{bmatrix}, \qquad \text{(A-21)}$$

where t_{ij} is either 0 or β^0. By hypothesis, we have

$$D_{1q}E_{1q}^T = 0, \text{ equivalently, } \beta^{-2}D_{1q}E_{1q}^T = 0,$$

which implies

$$b_{i1}t_{j1}\beta^{a_{i1}}\beta^{m_{j1}} + b_{i2}t_{j2}\beta^{a_{i2}}\beta^{m_{j2}} + \ldots + b_{in}t_{jn}\beta^{a_{in}}\beta^{m_{jn}} = 0 \qquad \text{(A-22)}$$

for $i = 1,2,\ldots,h$, and $j = 1,2,\ldots,s$.

By Lemma A-2, Equation (A-22) is equivalent to

$$b_{11}t_{j1}T_o^{a_{11}}T_o^{m_{j1}} + b_{12}t_{j2}T_o^{a_{12}}T_o^{m_{j2}} + \dots + b_{1n}t_{jn}T_o^{a_{1n}}T_o^{m_{jn}} = 0.$$

Writing this in matrix form, we have

$$\begin{bmatrix} b_{11}T_o^{a_{11}} & b_{12}T_o^{a_{12}} & \dots & b_{1n}T_o^{a_{1n}} \\ b_{21}T_o^{a_{21}} & b_{22}T_o^{a_{22}} & \dots & b_{2n}T_o^{a_{2n}} \\ \vdots & \vdots & & \vdots \\ b_{h1}T_o^{a_{h1}} & b_{h2}T_o^{a_{h2}} & \dots & b_{hn}T_o^{a_{hn}} \end{bmatrix} \begin{bmatrix} t_{11}T_o^{m_{11}} & \dots & t_{s1}T_o^{m_{s1}} \\ t_{12}T_o^{m_{12}} & \dots & t_{s2}T_o^{m_{s2}} \\ \vdots & & \vdots \\ t_{1n}T_o^{m_{1n}} & \dots & t_{sn}T_o^{m_{sn}} \end{bmatrix} = 0. \qquad \text{(A-23)}$$

Now

$$T_o = [\beta^1\ \beta^2\ \beta^3,\dots,\beta^{\ell}]_p$$
$$= [\beta^1(\beta^0\ \beta^1,\dots,\beta^{\ell})]_p\ .$$

Therefore, the first matrix of Equation (2.53) is just the p-ary representation of $D_{1q}\otimes[\beta^0\ \beta^1,\dots,\beta^{\ell-1}]$, and the second matrix is the p-ary representation of $E_{1q}^T\otimes[\beta^0\ \beta^1,\dots,\beta^{\ell-1}]$. Q.E.D.

It can be easily deduced that, for any i and j

$$\left[D_{1q}\otimes[\beta^i\ \beta^{i+1}\ \beta^{i+2},\dots,\beta^{i+\ell-1}]\right]_p \left[E_{1q}^T\otimes[\beta^j\ \beta^{j+1},\dots,\beta^{j+\ell-1}]\right]_p = 0 \qquad \text{(A-24)}$$

provided $D_{1q}E_{1q}^T = 0$. But the form $[\beta^0\ \beta^1,\dots,\beta^{\ell-1}]$, which corresponds to $i = j = 0$, is the simplest one, since

$$[\beta^0\ \beta^1,\dots,\beta^{\ell-1}]_p = I_{\ell}\ .$$

It should be noted that the null space of $\left[D_{1q} \otimes [\beta^0\ \beta^1, \ldots, \beta^{\ell-1}]\right]_p$ is the row space of

$$\left[E_{1q} \otimes \begin{bmatrix} \beta^0 \\ \beta^1 \\ \vdots \\ \beta^{\ell-1} \end{bmatrix}_{\llcorner p \lrcorner} \right],$$

where the notation $[\ \ ]_{\llcorner p \lrcorner}$ is the matrix with each element in the bracket expressed as a row vector over GF(p).

<u>Thoerem A-5</u> (identical to Theorem 2.7)

Let D_{1q} and E_{1q} be the respectively gp and pg matrices of a q-ary code C_1 of length n_1, where $q = p^{\ell}$. If D_{2q} is a $1 \times n_2$ pg matrix of a cyclic q-ary code C_2 with $n_2 \geq \ell$ and if, with suitable mapping, D_{2q} is of the form

$$\left[\beta^0\ \beta^1\ \beta^2, \ldots, \beta^{\ell-1} \,\middle|\, \beta^{i_\ell}\ \beta^{i_{\ell+1}}, \ldots, \beta^{i_{n_2-1}}\right],$$

where β^1 is a primitive element of GF(q); also, if the row space of E_{2p} is the null space of $[D_{2q}]_p$, then the null space of $[D_{1q} \otimes D_{2q}]_p$ is

$$\begin{bmatrix} I_{n_1} \otimes E_{2p} \\ \left[E_{1q} \otimes \left[\begin{array}{c|c} \begin{matrix} \beta^0 \\ \beta^1 \\ \vdots \\ \beta^{\ell-1} \end{matrix} & O_{COM} \end{array}\right] \right]_{\llcorner p \lrcorner} \end{bmatrix},$$

where 0_{COM} is introduced for the sake of compensating the length of each submatrices in the following manner

$$\left[\beta^i \begin{bmatrix} \beta^0 \\ \beta^1 \\ \cdot \\ \cdot \\ \cdot \\ \beta^{\ell-1} \end{bmatrix} \begin{array}{|c} \\ 0_{COM} \\ \\ \end{array}\right]_{\lfloor p \rfloor} = \left[\begin{matrix} \lfloor \beta^i \rfloor \\ \lfloor \beta^{i+1} \rfloor \\ \cdot \\ \cdot \\ \cdot \\ \lfloor \beta^{i+\ell-1} \rfloor \end{matrix} \begin{array}{|c} \\ 0_{\ell \times (n_2-\ell)} \\ \\ \end{array}\right]$$

provided rank $D_{1q} = h_1$, rank$[D_{1q} \otimes D_{2q}]_p = \ell h_1$, rank $D_{2p} = \ell$, and rank $E_{2p} = n_2-\ell$.

Proof

By Theorem A-1, we know the row space of $[I_{n_1} \otimes E_{2p}]$ is contained in the null space of $[D_{1q} \otimes D_{2q}]_p$. And

$$[D_{1q} \otimes D_{2q}]_p \left[\left[E_{1q} \otimes \begin{bmatrix} \beta^0 \\ \beta^1 \\ \cdot \\ \cdot \\ \cdot \\ \beta^{\ell-1} \end{bmatrix} \begin{array}{|c} \\ 0_{COM} \\ \\ \end{array}\right]_{\lfloor p \rfloor}\right]^T$$

$$= \left[D_{1q} \otimes \left[\beta^0\ \beta^1, \ldots, \beta^{\ell-1} \,\middle|\, 0_{1 \times (n_2-\ell)}\right]\right]_{\lfloor p \rfloor} \left[\left[E_{1q} \otimes \begin{bmatrix} \beta^0 \\ \beta^1 \\ \cdot \\ \cdot \\ \cdot \\ \beta^{\ell-1} \end{bmatrix} \begin{array}{|c} \\ 0_{COM} \\ \\ \end{array}\right]_{\lfloor p \rfloor}\right]^T = 0$$

by Theorem A-4.

Hence,

$$E = \begin{bmatrix} I_{n_1} \otimes E_{2p} \\ \left[E_{1q} \otimes \left[\begin{array}{c|c} \beta^0 & \\ \beta^1 & \\ \vdots & 0_{COM} \\ \beta^{\ell-1} & \end{array} \right] \right]_p \end{bmatrix} \qquad \text{(A-25)}$$

is indeed contained in the null space of the row space of $[D_{1q} \otimes D_{2q}]_p$. It remains to show that Rank $E = n_1 n_2 - \ell h_1$. Now,

$$\text{Rank } I_{n_1} \otimes E_{2p} = n_1 \text{ Rank } E_{2p} = n_1(n_2 - \ell_1) = n_1 n_2 - \ell n_1 \qquad \text{(A-26)}$$

$$\text{Rank} \left[E_{1q} \otimes \left[\begin{array}{c|c} \beta^0 & \\ \beta^1 & \\ \vdots & 0_{COM} \\ \beta^{\ell-1} & \end{array} \right] \right]_p = \left(\text{Rank } E_{1q} \right) \left\{ \text{Rank} \left[\begin{array}{c|c} \beta^0 & \\ \beta^1 & \\ \vdots & 0_{COM} \\ \beta^{\ell-1} & \end{array} \right]_p \right\}$$

$$= (n_1 - h_1)\ell = \ell n_1 - \ell h_1. \qquad \text{(A-27)}$$

We need only to show

$$\text{Rank } E = n_1 n_2 - \ell h_1 \, . \qquad \text{(A-28)}$$

Given

$$D_{2p} = [D_{2q}]_p = \left[\beta^0 \; \beta^1, \ldots, \beta^{\ell-1} \,\middle|\, \beta^{i_\ell} \; \beta^{i_{\ell+1}}, \ldots, \beta^{i_{n_2}} \right]_p$$

$$= [I_\ell \;\, P], \qquad \text{(A-29)}$$

then

$$E_{2p} = \left[-P^T \; I_{n_2-\ell}\right] \tag{A-30}$$

$$I_{n_1} \otimes E_{2p} = \begin{bmatrix} \left[-P^T \; I_{n_2-\ell}\right] & 0 & 0 \cdots 0 \\ 0 & \left[-P^T \; I_{n_2-\ell}\right] & \\ 0 & 0 & \ddots \\ \vdots & \vdots & \\ 0 & 0 & \left[-P^T \; I_{n_2-\ell}\right] \end{bmatrix} . \tag{A-31}$$

Now

$$\left[E_{1q} \otimes \left[\begin{array}{c|c} \beta^0 & \\ \beta^1 & \\ \vdots & 0_{COM} \\ \beta^{\ell-1} & \end{array} \right] \right]_p = \begin{bmatrix} [E_{11}] & [E_{12}] & \cdots & [E_{1n_1}] \\ [E_{21}] & [E_{22}] & \cdots & [E_{2n_1}] \\ \vdots & \vdots & & \vdots \\ [E_{(n_1-h_1)n_1}] & [E_{(n_1-h_1)n_1}] & \cdots & [E_{(n_1-h_1)n_1}] \end{bmatrix} . \tag{A-32}$$

Any non-zero $[E_{ij}]$ is of the form

$$\begin{bmatrix} \underline{\beta^{a_{ij}}} & \\ \underline{\beta^{a_{ij+1}}} & \\ \vdots & 0_{\ell(n_2-\ell)} \\ \underline{\beta^{a_{ij+\ell-1}}} & \end{bmatrix} = T_c^{\,a_{ij}} \left[I_\ell \; 0_{\ell(n_2-\ell)} \right] . \tag{A-33}$$

Since T_c is the companion matrix, it defines a linear mapping, hence the rank of $\begin{bmatrix} I_\ell & 0_{\ell(n_2-\ell)} \end{bmatrix}$ is invariant under such a transformation. Therefore,

$$\text{row space of } T_c^{a_{1j}} \begin{bmatrix} I_\ell & 0_{\ell(n_2-\ell)} \end{bmatrix} = \text{row space of} \begin{bmatrix} I_\ell & 0_{\ell(n_2-\ell)} \end{bmatrix} . \qquad \text{(A-34)}$$

Hence,

$$\text{rank} \begin{bmatrix} -P^T & I_{n_2-\ell} \\ T_c^{a_{1j}} \begin{bmatrix} I_\ell & 0_{\ell(n_2-\ell)} \end{bmatrix} \end{bmatrix} = \text{rank} \begin{bmatrix} -P^T & I_{n_2-\ell} \\ I_\ell & 0_{\ell(n_2-\ell)} \end{bmatrix}$$

$$= \text{rank} \begin{bmatrix} 0 & I_{n_2-\ell} \\ I_\ell & 0_{\ell(n_2-\ell)} \end{bmatrix} \text{by elementary row operations}$$

$$= \text{rank } I_{n_2-\ell} + \text{rank } I_\ell = n_2-\ell+\ell = n_2. \qquad \text{(A-35)}$$

This shows that none of the rows of $I_{n_1} \otimes E_{2p}$ can be obtained from

$$\begin{bmatrix} E_{1q} \otimes \begin{bmatrix} \beta^0 \\ \beta^1 \\ \vdots \\ \beta^{\ell-1} \end{bmatrix} & \vdots & 0_{COM} \end{bmatrix}_{\llcorner p \lrcorner}$$

and vice versa; hence,

$$\text{Rank } E = \text{Rank} \begin{bmatrix} I_{n_1} \otimes E_{2p} \end{bmatrix} + \text{Rank} \begin{bmatrix} E_{1q} \otimes \begin{bmatrix} \beta^0 \\ \beta^1 \\ \vdots \\ \beta^{\ell-1} \end{bmatrix} & \vdots & 0_{COM} \end{bmatrix}_{\llcorner p \lrcorner}$$

$$= n_1 n_2 - \ell n_1 + \ell n_1 - \ell h_1 = n_1 n_2 - \ell h_1 \qquad \text{Q.E.D.}$$

The technique described in this section is valid only when the gp matrix of the second component code is a $1 \times n_2$ matrix in GF(q) containing a submatrix $\beta^i[\beta^0\ \beta^1,\ldots,\beta^{\ell-1}]$. In Theorem 2.11, the submatrix is put at the left end of the gp matrix D_{2q}. Actually, the submatrix $\beta^i[\beta^0\ \beta^1,\ldots,\beta^{\ell-1}]$ can be put any place, if D_{2q} is of the form

$$D_{2q} = \left[\beta^{i1}\ \beta^{i2},\ldots,\beta^{ij} \;\vdots\; \beta^0\ \beta^1,\ldots,\beta^{\ell-1} \;\vdots\; \beta^{i\ell+j}\ \beta^{i\ell+j+1},\ldots,\beta^{in2}\right],$$

since the null space of $[D_{1q} \otimes D_{2q}]_p$ is

$$\begin{bmatrix} I_{n_1} \otimes E_{2p} \\ \left[E_{1q} \otimes \left[\begin{array}{c|c|c} 0_{COM_1} & \begin{matrix}\beta^0\\ \beta^1\\ \cdot\\ \cdot\\ \beta^{\ell-1}\end{matrix} & 0_{COM_2}\end{array}\right]\right]_p \end{bmatrix},$$

where 0_{COM_1} and 0_{COM_2} compensate the submatrix in the following manner

$$\left[\beta^i \left[\begin{array}{c|c|c} 0_{COM_1} & \begin{matrix}\beta^0\\ \beta^1\\ \cdot\\ \cdot\\ \beta^{\ell-1}\end{matrix} & 0_{COM_2}\end{array}\right]\right]_p = \left[\begin{array}{c|c|c} 0_{\ell\times j} & \begin{matrix}\lfloor\beta^i\rfloor\\ \lfloor\beta^{i+1}\rfloor\\ \cdot\\ \cdot\\ \lfloor\beta^{i+\ell-1}\rfloor\end{matrix} & 0_{\ell\times(n-j\ell)}\end{array}\right].$$

For the case that the second component code is cyclic, it is always possible to choose a proper extension field such that

$$D_{2q} = [\beta^0\ \beta^1\ \beta^2,\ldots,\beta^{n_2}].$$

The null space of the product space can then be determined.

If there is no submatrix in D_{2q} which is of the form $\beta^i[\beta^0\ \beta^1,\ldots,\beta^{\ell-1}]$, then it is necessary to find a permutation matrix π such that

$$D_{2q}\pi = \left[\begin{array}{c|c}\beta^0\ \beta^1,\ldots,\beta^{\ell-1} & \beta^{i\ell},\ldots,\beta^{in2}\end{array}\right].$$

By Theorem 2.11, the null space of $\left[D_{1q}\otimes(D_{2q}\pi)\right]_p$ is

$$\left[\begin{array}{c} I_{n_1}\otimes(E_{2p}\,\pi) \\ \left[\begin{array}{c|c} E_{1q}\otimes\begin{bmatrix}\beta^0\\ \beta^1\\ \vdots\\ \beta^{\ell-1}\end{bmatrix} & O_{COM}\end{array}\right]_p \end{array}\right].$$

It can be easily shown that the null space of $\left[D_{1q}\otimes D_{2q}\right]_p$ is

$$\left[\begin{array}{c} I_{n_1}\otimes E_{2p} \\ \left[\begin{array}{c|c} E_{1q}\otimes\begin{bmatrix}\beta^0\\ \beta^1\\ \vdots\\ \beta^{\ell-1}\end{bmatrix} & O_{COM}\end{array}\right]_p \end{array}\ \underbrace{\begin{bmatrix}\pi & & & \\ & \pi & & \bigcirc \\ & & \ddots & \\ \bigcirc & & & \pi\end{bmatrix}}_{n_1\ \pi\text{'s}}\right].$$

If the gp matrix D_{2q} contains more than one row, we first find the null space E_1 of the product space $\left[D_{1q}\otimes D'_{iq}\right]_p$, where D'_{iq} is the submatrix contains i^{th} row of D_{2q}. The entire null space E_p of $\left[D_{1q} \otimes D_{2p}\right]_p$ can be expressed as

$$\text{row space of } E_p = \bigcap_{\text{all } i} \text{row space of } E_i, \tag{A-36}$$

where $\bigcap$ denotes intersection. To evaluate the null space E_p by determining the intersection of the row spaces of E_i's is a very tedious procedure. Therefore, when D_{2q} contains more than one row, the direct method discussed in Section 2.1 is recommended to calculate the pg matrix E_p when the gp matrix is given as a tensor product of $\left[D_{1q} \otimes D_{2q}\right]_p$.

REFERENCES

1. D. Slepian, "Some Further Theory of Group Codes", The Bell System Technical Journal, 39, (September 1960), pp. 1219-1252.

2. W. W. Peterson, Error-Correcting Codes, The M.I.T. Press and John Wiley & Sons, Inc., 1961.

3. G. Birkhoff and S. MacLane, A Survey of Modern Algebra, Macmillan, New York, 1961.

4. D. T. Tang, "Dual Codes as Variable Redundancy Codes", 1965 IEEE International Conventional Record, Part 7, pp. 220-226.

5. S. W. Golomb (Editor), Digital Communications with Space Applications, Prentice-Hall, Inc., 1964.

6. P. Elias, "Error-Free Coding", IRE Trans. PGIT-4, 1954, pp. 29-37.

7. R. W. Hamming, "Error Detecting and Error Correcti ng Codes", Bell System Technical Journal, 29, 1950, pp. 147-160.

8. M. J. E. Golay, "Notes on Digital Coding", Proc. I.R.E., 37, Correspondence, 1949, p. 657.

9. D. Slepian, "A Class of Binary Signalling Alphabets", Bell System Technical Journal, 35, 1956, pp. 203-234.

10. D. Slepian, "A Note on Two Binary Signalling Alphabets", I.R.E. Trans. IT-2, 1956, pp. 84-86.

11. E. Prange, Cyclic Error-Correcting Codes in Two Symbols, AFCRC-TN-57-103, Air Force Cambridge Research Center, Cambridge, Mass. (September 1957).

12. R. C. Bose and D. K. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes", Information and Control, 3, 1960, pp. 68-79.

13. R. C. Bose and D. K. Ray-Chaudhuri, "Further Results on Error Correcting Binary Group Codes", Information and Control, 3, 1960, pp. 279-290.

14. A. Hocquenghem, "Codes Correcteurs d'erreurs", Chiffres, Vol. 2, (September 1959), pp. 147-156.

15. R. T. Chien, "Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes", IEEE Trans. on Information Theory, Vol. IT-10, (October 1964), pp. 357-363.

16. G. D. Forney, Jr., "On Decoding B-C-H Codes", IEEE Trans. on Information Theory, Vol. IT-11, (October 1965), pp. 549-557.

17. E. R. Berlekamp, "On Decoding Binary Bose-Chaudhuri-Hocquenghem Codes", IEEE Trans. on Information Theory, Vol. IT-11, (October 1965), pp. 577-579.

18. J. L. Massey, "Step-by-Step Decoding of the Bose-Chaudhuri-Hocquenghem Codes", IEEE Trans. on Information Theory, Vol. IT-11, (October 1965), pp. 580-585.

19. P. R. Halmos, Finite-Dimensional Vector Spaces 2nd Edition, D. Van Nostrand Company, Inc., Princeton and New York, 1958.

20. L. Calabi and H. G. Haefeli, "A Class of Binary Systematic Codes Correcting Errors at Random and in Bursts", IRE Trans. CT-6 Special Supplement, 1959, pp. 79-94.

21. C. Hobbs, Hobbs' Code is mentioned in Reference 20.

22. W. H. Kautz, "A Class of Multiple-Error-Correcting Codes for Data Transmission and Recording", Stanford Research Institute Technical Report No. 5 (SRI Project 2124), Palo Atlo, California, (August 1959).

23. J. K. Wolf and B. Elspas, "Error-Locating Codes - A New Concept in Error Control", IEEE Trans. on Information Theory, Vol. IT-9, No. 2, (April 1963), pp. 113-117.

24. J. K. Wolf, "On Codes Derivable from the Tensor Product of Check Matrices", IEEE Trans. on Information Theory, Vol IT-11, No. 2, (April 1965), pp. 281-284.

25. J. K. Wolf, "On an Extended Class of Error-Locating Codes", Information and Control, Vol. 8, No. 2, (April 1965), pp. 163-169.

26. S. H. Chang and L. J. Weng, "Error-Locating Codes", 1965 IEEE International Convention Record Part 7, pp. 252-258.

27. H. O. Burton and E. J. Weldon, Jr., "Cyclic Product Codes", IEEE Transactions on Information Theory, Vol. IT-11, No. 2, (July 1965), pp. 433-439.

28. S. H. Chang and L. J. Weng, "Dual Product Codes", IEEE International Symposium on Information Theory, January 31 - February 2, 1966 at UCLA.

29. R. H. Barker, "Group Synchronizing of Binary Digital Systems", in Communication Theory, W. Jackson (Editor), New York, Academic Press Inc., 1953, pp. 273-287.

30. E. Prange, "The Use of Information Sets in Decoding Cyclic Codes", IRE Trans. IT-8, (September 1962).

31. J. MacWilliams, "Permutation Decoding of Systematic Codes", The Bell System Technical Journal, Vol. XLIII, No. 1, Part 2, (January 1964), pp. 485-505.

32. W. W. Peterson, "Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes", IRE Trans. IT-6, 1960, pp. 459-470.

33. D. Gorenstein and N. Zierler, "A Class of Error-Correcting Codes in p^m Symbols", J. Soc. Indust. Appl. Math., Vol. 9, No. 2, (June 1961), pp. 207-214.

34. S. J. Yuill, "An Algebraic Structure for Error-Locating Codes", an unpublished note, Communication Theory Group, Northeastern University, Boston, Massachusetts.

35. J. Cocke, "Lossless Symbol Coding with Nonprimes", IRE Trans. on Information Theory IT-5, (March 1959), pp. 33-34.

36. A. A. Albert, Fundamental Concepts of Higher Algebra, The University of Chicago Press, 1959.

Distribution List

USASDRL (SIGRA/SL-NAC)
Fort Monmouth, New Jersey
Attn: Mr. G. Balano

USASDRL (SIGRA/SLNAC)
Fort Monmouth, New Jersey
Attn: Mr. J. Bartow

Signatron, Incorporated
594 Marrett Road
Lexington, Massachusetts 02173
Attn: Dr. Phillip A. Bello

Signatron, Incorporated
Miller Building
594 Marrett Road
Lexington, Massachusetts
Attn: Dr. Julian Bussgang

New York University
School of Engineering Science
Department of Electrical Engineering
University Heights
Bronx, New York 10453
Attn: Dr. Robert F. Cotellessa

General Dynamics/Fort Worth
Chief Librarian
Post Office Box 748
Fort Worth, Texas 76101
Attn: P. R. De Tonnancour

Northeastern University
Office of Research Administration
360 Huntington Avenue
Boston, Massachusetts 02115
Attn: M. W. Essigmann

Polytechnic Institute of Brooklyn
Research Coordinator
333 Jay Street
Brooklyn, New York 11201
Attn: Jerome Fox

Massachusetts Institue of Technology
Librarian, Lincoln Laboratory
Post Office Box 73
Lexington, Massachusetts
Attn: Mary A. Granese

The Johns Hopkins University
School of Engineering Science
34th and Charles
Baltimore, Maryland 21218
Attn: W. H. Huggins

The Rand Corporation
1700 Main Street
Santa Monica, California
Attn: Dr. R. E. Kalaba

Michigan State University
Electrical Engineering Department
Engineering Building
East Lansing, Michigan
Attn: Dr. W. Kilmer

U. S. Army Electronics R & D Labs.
Commanding Officer SELRA/XC
Fort Monmouth, New Jersey 07703
Attn: Mr. Mattes

Parke Mathematical Laboratories, Inc.
One River Road
Carlisle, Massachusetts 01741
Attn: Dr. Nathan Grier Parke, III

Syracuse University
Syracuse, New York
Attn: Professor F. Reza

RCA - Defense Electronic Products
Staff Engineer - Bldg. 10-Floor 7
Organization of Chief Tech. Admin.
Camden, New Jersey
Attn: Mr. Harold J. Schrader

Bell Telephone Laboratories
1600 Osgood Street
North Andover, Massachusetts 01845
Attn: Mr. David Shnidman

General Electric Company
Research Laboratory
Post Office Box 1088
Schenectady, New York 12301
Attn: Dr. R. L. Shuey

Sylvania Electronic Systems
Applied Research Laboratory
40 Sylvan Road
Waltham, Massachusetts 02154
Attn: Dr. Seymour Stein

Harvard University
199 Pierce Hall
Oxford Street
Cambridge, Massachusetts 02138
Attn: Dr. D. W. Tuft

Litton Systems, Incorporated
335 Bear Hill Road
Waltham, Massachusetts 02154
Attn: Dr. David Van Meter

Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025
Attn: Mr. W. R. Vincent

Director - AFOSR
Research Information Office
Washington, D. C.
Attn: Dr. Harold Wooster

Philco Corporation
Plant No. 50 - Systems Engineering
4700 Wissahickon Avenue
Philadelphia, Pennsylvania 19144
Attn: Mr. S. Zebrowitz

AFCRL (CRW) Stop 30
L. G. Hanscom Field
Bedford, Massachusetts 01731

Autonetics, Division of North
American Aviation, Incorporated
3370 Miraloma Avenue - Bldg. 202
Anaheim, California 92803
Attn: Main Tech. Library D/503-31

HRB Singer, Incorporated
1517 Science Avenue
State College, Pennsylvania

ITT Federal Laboratories
500 Washington Avenue
Nutley, New Jersey
Attn: Technical Library

Massachusetts Institute of Technology
Engineering Library
Room 10-550
Cambridge, Massachusetts 02139

Motorola, Incorporated
Post Office Box 5409
Phoenix, Arizona 85010
Attn: Technical Librarian

National Security Agency
Fort George G. Meade, Maryland
Attn: Director C3/TDL

Office of Naval Research
Department of the Navy
Washington, D. C.
Attn: Code 427 - Electronics Branch

RADC (EMCRS)
Griffiss Air Force Base, New York 13442

Rand Corporation
1700 Main Street
Santa Monica, California 90406
Attn: Library

Sandia Corporation - Sandia Base
Post Office Box 5800
Albuquerque, New Mexico
Attn: Classified Document Division

Space Technology Laboratories, Inc.
STL Technical Library
Document Acquisitions
Post Office Box 95001
Lost Angeles, California

The University of Michigan
Institute of Science and Technology
Box 618
Ann Arbor, Michigan 48107
Attn: Technical Documents Service

HQ., AFCRL, OAR (CRBK) Stop 30
L. G. Hanscom Field
Bedford, Massachusetts 01730

AFCRL (CRMXLR) Stop 29
L. G. Hanscom Field
Bedford, Massachusetts 01730
Attn: Mrs. Cora Gibson

AFCRL (CRMXLR) Stop 29
L. G. Hanscom Field
Bedford, Massachusetts 01730

AFCRL (CRMXRD) Stop 30
L. G. Hanscom Field
Bedford, Massachusetts 01730

AFCRL (CRMXRA) Stop 39
L. G. Hanscom Field
Bedford, Massachusetts 01730

AFCRL (CRN) Stop 30
L. G. Hanscom Field
Bedford, Massachusetts 01730

AFCRL (CRTE) Stop 30
L. G. Hanscom Field
Bedford, Massachusetts 01730

AFCRL (CRTPM) Stop 30
L. G. Hanscom Field
Bedford, Massachusetts 01730

ADC
Operations Analysis Office
Ent. Air Force Base, Colorado

AEDC (ARO, Inc.)
Arnold Air Force Station
Tennessee 37389
Attn: Library/ Documents

AFETR
Technical Library - MU - 135
Patrick Air Force Base
Florida 32925

AFIT (MCLI, Library)
Building 125 - Area B
Wright-Patterson Air Force Base
Ohio 45433

AFSC-STLO (RSTAL)
AF Unit Post Office
Los Angeles, California 90045

AFSC-STLO (RTSAB)
Waltham Federal Center
424 Trapelo Road
Waltham, Massachusetts 02154

AFSC-STLO (RTSAS)
452 Deguigne
Sunnyvale, California 94086

AFSC-STLO (RTSUM)
68 Albany Street
Cambridge, Massachusetts 02139

AFSWC
Technical Library
Kirtland Air Force Base
New Mexico

Director, Air University Library
Maxwell Air Force Base
Alabama 36112
Attn: AUL3T

APGC
(PGBPS-12)
Eglin Air Force Base
Florida 32542

AWS (AWSSS/TIPD)
Scott Air Force Base, Illinois

BSD
Technical Library
Norton Air Force Base, California

OAR (RRY)
1400 Wilson Boulevard
Arlington, Virginia 22209

RADC (EMTDL)
Documents Library
Griffiss Air Force Base
New York 13440

RTD
Scientific Director
Bolling Air Force Base
Washington, D. C.

RTD (APX)
Wright-Patterson Air Force Base
Ohio 45433

RTD (AWX)
Wright-Patterson Air Force Base
Ohio 45433

RTD (FDX)
Wright-Patterson Air Force Base
Ohio 45433

Systems Engineering Group (RTD)
Wright-Patterson Air Force Base
Ohio 45433
Attn: SEPIR

SAC
Operations Analysis Office
Offutt Air Force Base, Nebraska

SSD (SST) Space Systems Division
Los Angeles Air Force Station
Air Force Unit Post Office
Los Angeles, California 90045

TAC
Operations Analysis Office
Langley Air Force Base, Virginia

USAF Academy
Library
Colorado Springs, Colorado

USAF Academy
FJSRL
Colorado 80840

USAFSS (ORD)
San Antonio, Texas 78241

Army Electronic Proving Ground
Technical Library
Fort Huachuca, Arizona

Army Missile Command
Redstone Scientific Information Center
Redstone Arsenal, Alabama 35809
Attn: Chief, Documents Section

U. S. Army Research Office
3045 Columbia Pike
Arlington, Virginia 22204
Attn: Technical Library

Los Alamos Scientific Laboratories
Los Alamos, New Mexico

U. S. Army Electronic Command
Technical Document Center
Fort Monmouth, New Jersey 07703
Attn: AMSEL-RD-MAT

Bureau of Naval Weapons
(DLI-31 - Library)
Washington, D. C.

Bureau of Ships
(Code 320)
Washington, D. C.

Chief of Naval Operations
(OP-413-B21)
Washington, D. C.

Director
Naval Research Laboratory
Washington, D. C. 20390
Attn: 2027

Naval Air Development Center
Library
Johnsville, Pennsylvania

Naval Missile Center
Library
Point Mugu, California

Naval Ordnance Laboratory
Technical Library
White Oak, Silver Spring, Maryland

Naval Post Graduate School
Technical Library
Monterey, California

Commanding Officer and Director
U. S. Navy Electronics
Laboratory (Library)
San Diego, California 92152

Commander (Code 753)
U. S. Naval Ordnance Test Station
China Lake, California 93555
Attn: Technical Library

Commanding Officer
Office of Naval Research Branch Office
Box 39 - Fleet Post Office
New York 09510

U. S. Naval Academy
Library
Annapolis, Maryland

ARPA
Library
The Pentagon
Washington, D. C.

Central Intelligence Agency
Washington, D. C. 20505
Attn: OCR/DD/STD. Distribution

Director
Defense Atomic Support Agency
Washington, D. C. 20301
Attn: Technical Library Section

Defense Documentation Center (DDC)
Cameron Station Alexandria, Virginia 22314

Environmental Sciences Services Adm.
Library
Boulder Laboratories
Boulder, Colorado 80302

NSF
1951 Constitution Avenue, N.W.
Washington, D. C. 20235

ODDR+E (Library)
Room 3C-128
The Pentagon
Washington, D. C. 20301

Smithsonian Astrophysical
Observatory - Library
60 Garden Street
Cambridge, Massachusetts 02138

U. S. Atomic Energy Commission
Hq., Library - Room G-017
Reports Section
Washington, D. C. 20545

J. S. Weather Bureau
Library - ESSA - Room 806
8060 13th Street
Silver Spring, Maryland 20910

AIAA - Library
Technical Information Service
750 Third Avenue
New York, New York

Aerospace - Library
2400 East El Segundo Boulevard
El Segundo, California

DIA
(DIAAP - 142)
Washington, D. C. 20301

FAA
Bureau of Research & Development
300 Independence Avenue, S.W.
Washington, D. C. 20553

Government Printing Office
Library
Division of Public Documents
Washington, D. C.

Library of Congress
Aerospace Technical Division
Washington, D. C. 20540

Library of Congress
Exchange & Gift Division
Washington, D. C. 20540

NAS/NRC - Library
Executive Secretary
Advisory Committee to AFSC
2101 Constitution Avenue, N.W.
Washington, D. C. 20418

NASA Representative (S-AK/DL)
Scientific & Technical Information Facility
Post Office Box 5700
Bethesda, Maryland 20014

NASA-Ames Research Center
Technical Library
Moffett Field, California

NASA-Flight Research Center
Library
P. O. Box 273
Edwards, California 93523

NASA-Goddard Institute for Space Studies
(Library)
2880 Broadway
New York, New York 10025

NASA-Goddard Space Flight Center
Technical Library
Greenbelt, Maryland

NASA-JPL - Library
4800 Oak Grove Drive
Pasadena, California

NASA-Langley Research Center
Technical Library
Langely Station
Hampton, Virginia

NASA-Lewis Research Center
Library - Mail Stop 60-3
21000 Brookpark Road
Cleveland, Ohio 44135

NASA-Manned Spacecraft Center
Technical Library
Houston, Texas 77058

IIT Research Institute
Document Library
10 West 35th Street
Chicago, Illinois 60616

Massachusetts Institute of Technology
Department of Meteorology
Cambridge, Massachusetts
Attn: Professor Henry G. Houghton

Rockefeller Institute
New York, New York
Attn: Dr. Mark Kac

Stanford University
Department of Physics
Stanford, California
Attn: Professor Leonard Schiff

University of California
Department of Physics
Los Angeles, California
Attn: Dr. Joseph Kaplan

University of Illinois
Department of Physics
Urbana, Illinois
Attn: Professor Frederick Seitz

British Defence Staffs
British Embassy
Scientific Information Officer
3100 Massachusetts Avenue, N.W.
Washington, D. C. 20008

Battelle Memorial Institute
Library
505 King Avenue
Columbus, Ohio 43201

Boeing Aero-Space Division
Post Office Box 3707
Seattle, Washington
Attn: Dr. N. L. Krisberg

The Mitre Corporation
Post Office Box 208
Bedford, Massachusetts 01700
Attn: Library

NCAR
Boulder, Colorado
Attn: Library

The Rand Corporation
1700 Main Street
Santa Monica, California 90406
Attn: Library-D

TRW Systems
Director, Physical Research Center
One Space Park
Redondo Beach, California 90278
Attn: David B. Langmuir

The Johns Hopkins University
2500 West Rogers Avenue
Baltimore, Maryland 21215
Attn: Dr. Carl Kaplan

Chief, Canadian Defence Research Staff
2450 Massachusetts Avenue, N.W.
Washington, D. C. 20008
(Technical & Scientific Reports will
be released for military purposes only
and any proprietary rights which may
be involved are protected by United
States/United Kingdon & Canadian
Government Agreements)

National Research Council
Library
Ottawa 2, Ontario
Canada

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Northeastern University 360 Huntington Avenue Boston, Massachusetts	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

"On Linear Product Codes and Their Duals"

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific Interim Report

5. AUTHOR(S) *(Last name, first name, initial)*

Weng, Lih-Jyh

6. REPORT DATE June 1966	7a. TOTAL NO. OF PAGES 177	7b. NO. OF REFS 36

8a. CONTRACT OR GRANT NO. AF19(628)-3312 Grant NGR-22-011-013	9a. ORIGINATOR'S REPORT NUMBER(S) Scientific Report No. 4
b. PROJECT AND TASK NO. 4610-03	
c. DOD ELEMENT 62405304	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* AFCRL-66-471
d. DOD SUBELEMENT 674610	

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES Supported in part by the National Aeronautics and Space Administration	12. SPONSORING MILITARY ACTIVITY Hq. AFCRL, OAR (CRB) United States Air Force L.G.Hanscom Field,Bedford, Mass.

13. ABSTRACT

In this report the value of studying the tensor product of of linear codes is demonstrated. The pertinent problems concerning these product codes are outlined. Algebraic techniques for determining the null space of a tensor product space are developed. The understanding of the null space of a product space is useful not only in the development of this report, but also for future research work. Decomposition of the procedure and implementation of encoding and decoding of a product code into those of its component codes are shown. In the case where the component codes are cyclic, the product code has the special feature that its encoder and syndrome calculator can be easily converted to those of its dual codes by programmed switching. A simple decoding scheme; namely, permutation decoding, which is capable of correcting a large fraction of all the correctable errors of a systematic cyclic code, is investigated. It is suggested that it be used either as a part of the correction-detection scheme or in combination with an auxiliary scheme to attain full error correction capability. Finally, the minimum distances of product codes, and suitable communication channels for employing such codes, are discussed.

DD FORM 1 JAN 64 1473

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Algebraic Coding Theory Linear Codes Product Codes Dual Codes						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
______________________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
______________________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
______________________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, *or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

Unclassified
Security Classification