MAJORITY LOGIC DECODABLE SYMMETRIC DIRECT PRODUCT CODES

by

George P. Papoulias

(August, 1971)

Submitted to the Department of Electrical Engineering
in partial fulfilment of the requirements for the
degree of Master of Science.

Department of Electrical Engineering
Faculty of Pure and Applied Science
University of Ottawa
Ottawa, Canada

1971.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my supervisor Prof. S.G.S. Shiva for his invaluable help through numerous discussions, suggestions, constructive criticism and his continuous and sincere encouragement.

I also wish to thank my colleagues of the Communication System Research for their helpful suggestions and heated conversations.

My thanks are also directed towards Mme L. LeBlanc and Miss Robin R. Francis for their efforts to produce such a neat manuscript.

Finally, I am gratefull for the financial assistance of the National Research Council .

ABSTRACT

A subclass of binary direct product codes, called symmetric direct product codes, is investigated. These codes are low rate codes. A decoding procedure is presented for the case when the component code is majority logic decodable. This decoding algorithm has a random error correcting capability which is a compromise between cascade decoding and straight majority logic decoding.

It has been proved that a large number of error patterns of weight $(2n-t)t$ or less, where n and t are the length and the error correcting capability of the component code respectively, may be corrected.

Symmetry is the price paid for the superior overall error correcting capability.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D.)

CHAPTER 1.

INTRODUCTION

One of the first problems in communications is to transmit information or data at as high a rate as possible with the least possible errors. General theory indicates that if the rate of the source is less than the channel capacity, it is possible to encode the data in such a way that this can be recovered from the noise received signal with an arbitrarily small error probability. This probability approaches zero as the length of the code approaches infinity. The fact that the information must be coded properly is the significant result attributed to Shannon in information theory [1]. The communication channel has a definite capacity as well as a definite information rate for typical information sources.

From the practical point of view our attention has been concentrated on the coding and decoding techniques proposed today, though we may also improve the channel. Thus efforts have been directed towards the designing of good coding and decoding schemes which could be easily implemented.

Although it is not possible to combat the channel errors completely with the use of coding, it is possible to reduce their undesirable influence, keeping the information rate as high as possible.

In coding theory we are faced with the basic problem of finding "good" codes with practical encoding and decoding techniques.

It is possible, by using appropriate codes to send information at any rate less than the channel capacity so that we can decode with a much smaller probability of errors. It depends on the communication channel we are dealing with, whether the errors occur independently,

in well-defined bursts, or in random multiple bursts [2] . A code is designed to correct the errors which often occur in the channel under consideration rather than all of the errors that might possibly occur in the channel. Therefore codes have been constructed which can correct any pattern of t or fewer errors in a block of n symbols. Consequently, codes for correcting bursts of errors are required.

Since the cyclic codes can be easily implemented, they have been used in most error control systems to date. This has significant advantages in both one-way and two-way channels [3,4,5,6].

The Bose-Chaudhuri-Hocquenghem [7,8] (BCH) codes and the Euclidian Geometry Codes [9] can apply to channels, in which the errors occur independently, with an optimum error-correcting capbility, but they prove not to be good burst correctors because they are not designed for that purpose. For well-defined bursts of errors, i.e., for bursts or length b or less, the Fire [10] codes appear to be very suitable but they have poor random error-correcting abilities, since they are not designed to satisfy this situation.

The Reed-Solomen [11] codes are useful for multiple-burst correction (MBC) as well as for single-burst correction (SBC). Other MBC codes have been proposed by Kasahara [12] , Corr [13], Stone [14, 15], Tavares and Shiva [16] and Shiva and Sheng [17] .

Due to the structure of the cyclic codes, encoders can be developed without much difficulty [3,6] . Fairly simple decoders can be built for Fire codes [3,6], for BCH codes when t is small [18.20] and for some special codes belonging to the set of Euclidian Geometry codes [9], while the disadvantage of the Reed-Solomon codes is that the decoding operations in general, are nonbinary operations.

We follow two basic approaches to the problem of finding practical

encoding and decoding schemes, namely the algebraic approach and the probabilistic one.

Following the first approach we use modern algebra to construct codes associated with efficient decoding algorithms. The BCH codes form the most important class of codes derived with respect to this approach.

In the second approach we use a probabilistic decoding algorithm, something like the method of maximum likelihood decoding. The sequential decoding associated with the outstanding class of convolutional codes [4,21] is an immediate result of this approach. The use of long codes [1] is necessary for efficient error correction and none of the codes mentioned so far satisfies this situation. It seems that the BCH codes are useless for long length in the sense that either the ratio of the minimum distance over the length of the code or the information rate goes to zero (see definitions 2.2.5 and 2.2.7).

It is possible to get long codes by combining the two approaches mentioned above. These codes are called concatenated codes [22]. It has been proved that at all rates less than channel capacity, the decoding error probability approaches zero exponentially with code length, while the complexity of the decoder increases slowly.

We can combine two or more codes to a more powerful code. The idea was first used in IBM computers [3] where an iterated code was used for error detection on mangnetic-tape units. Following a construction first introduced by Elias [23], it is sometimes advantageous to write the coefficients of a code-word of length n in a two-dimensional array in which each column is a code-word in one code (component code) and each row is a code-word in another (not necessarily different) provided that the length n is a composite number. Owing to the construction of these codes they are called direct product codes.

This idea can be readily extended to an arbitrary number of dimensions.

The product codes appear to be useful on channels such as the switched telephone systems, magnetic tapes etc., in which errors tend to be both random and clustered. Thus these codes combine the usefulness of burst-correcting and random-error correcting codes. This reason and the fact that the properties of a product code depend on the properties of its component codes, give the answer to the question why the iteration of codes attracted the attention of many researchers recently.

The term "product code" was first introduced by Slepian [24].

He has proved that the generator matrix of a product code is combinatorially equivalent to the tensor product of the generator matrices of its component codes.

Burton and Weldon [25] showed that it is possible to obtain a cyclic product code by using cyclic component codes under certain conditions. Abramson [26] introduced the interleaving argument to investigate cyclic product codes. This argument gives a simple proof of Burton - Weldon result and provides a simple form for the generator polynomial of the cyclic product code. The argument suggests burst correction.

Wolf [27] proved that by using the idea of tensor product of the check matrices of binary and non-binary codes we can construct a class of codes capable of correcting random bursts and bursts of bursts of errors [2].

Chien and Tang [28] proved that efficient product codes can be constructed offering different degrees of protection against independent errors, burst errors and multiple burst errors. It is also shown that the implementation of low-redundancy codes (see definition 2.5.4.) is accomplished with little extra cost.

The factorization of product codes has been considered by Goethals [29]

and it is, in a sense, the converse problem of the one considered by Burton and Weldon [25] . He shows how irreducible cyclic codes of composite block length may always be factorized, provided that the factors are relatively prime and his results include these of Assmus and Mattson [30] as a particular case.

Three decoding methods have been suggested for product codes.

The first method proposed is to use a decoder designed for the product code. This method corrects all errors guaranteed by the minimum distance of the product code but necessitates a complex decoder.

In the second method first the column code (row code) is decoded and then the row code (column code) is decoded. This method is known as cascade decoding. The advantages of this method are that one can use the decoders of the component codes and the error-correction performance may be beyond the usual Elias bound for product codes [23] . A general problem associated with this technique is that errors guaranteed correctable by the minimum distance of the product code are not necessarily correctable by the cascade decoding. Abramson [31] termed these kind of errors "permanent errors". He applies the cascade decoding procedure to find an upper bound to the probability that an error pattern will be permanent.

Elspas [32] used the burst detecting properties of subcodes to correct a "spot" error, i.e., a two-dimensional error pattern within a product code array. He showed that many error patterns could be corrected by detecting errors through column decoding, then erasing the columns with detected errors and using the row component code to fill in the erased symbols.

Elspas' spot-error correction was generalized by Bridwell and Wolf [33] to multiple-spot correction. They introduced codes which corrects a single, triple and quadruple bursts and five single errors.

A third method, proposed recently, combines the advantages of both the methods described previously, provided that at least one of the component codes is majority logic decodable [MLD] [21].

Bahl and Chien [34] used p one-dimensional single-parity-check codes of relatively prime block lengths to construct a class of cyclic product codes which are completely orthogonal and have certain MBC capability by one step threshold decoding. Algorithms to decode iterated codes, when at least one of the component codes is majority decodable, are given by Lin and Weldon [35] and Reddy [36]. Gore [37] proved that if the component codes are L_1 and L_2-step orthogonalizable, then the product code is $(L_1 + L_2 - 1)$-step orthogonalizable.

The subject of this thesis is concerned with the symmetric direct product codes (SDPC). It is shown how we can construct such codes and that they have low information rate. A general procedure concerning the construction of product code words under certain restrictions is also developed. It seems that by applying the cascade decoding procedure for direct product code whose each of the component codes has even minimum distance, its error correcting capability increases.

Emphasis has been given on decoding of SDPC's. A decoding algorithm has been developed in the case where the component code of the SDPC is MLD. This algorithm is a slight variation of cascade decoding.

It is shown that, in regards to error correcting capability of the SDPC, the procedure is a compromise between cascade decoding and majority logic decoding.

Chapter 2 is devoted to the problem of transmission of information. The basic background material concerning error correcting codes is also presented.

Chapter 3 deals with the large class of two-dimensional codes. The subclass of direct-product codes is discussed more exclusively.

In Chapter 4 the results of the research concerning symmetric direct product codes are presented, and a new decoding procedure for these codes is described.

Finally, Chapter 5 contains the concluding remarks.

CHAPTER 2

TRANSMISSION AND INFORMATION

2.1 The Digital Communication System

A general communication system consists of three major parts, namely, the transmitter, the channel of the receiver. Due to the fact that the digital computers process information in a digital form, we use digital techniques to record and transmit information. A digital communication system is a transmission system which transmits information or data in a digital format. A typical digital communication system is shown in figure 2.1. In this thesis we will be concerned solely with this type of system.

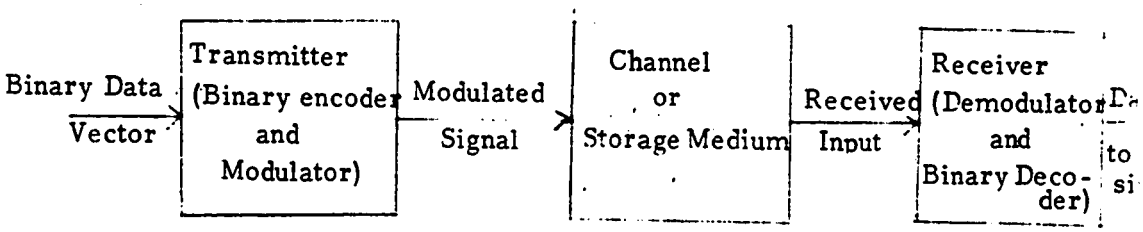


Figure 2.1

In any event the function of the channel is to reproduce at its output the message presented to it at its input. A real channel accomplishes this only approximately. Many reasons such as noise (gaussian or impulse), linear distortion, nonlinearities, frequency offset, etc., can cause in the output of the modulator-demodulator channel to be different from the input. The errors inherent in the received sequence are unavoidable.

2.2 Coding for transmission of information.

We may reduce the undesirable effects of the channel errors with the use of coding. Suppose that we wish to transmit a message M consisting of k message digits which we received from a digital information souce in the form of a data vector $a = (a_1, \ldots a_k)$, say, $a = (0010)$.

The vector a enters the (binary) encoder (see Fig. 2.2a) where r check digits are annexed and the encoder transmits the vector $V = (a_1, \ldots, a_n)$ consisting of $n = k + r$ channel digits. Assuming that the channel noise changes sufficiently few of these n transmitted channel digits, the r check digits may provide the receiver with sufficient information to enable it to detect and correct the channel errors. Given any particular sequence of k message digits, the encoder selects the r parity-check digits according to a predetermined rule. This is called the encoding problem. For the particular sequence a the encoder will annex the three digits 0, 1 and 1 and will transmit the entire block of 7 digits, i.e., the vector $V = (0010011)$.

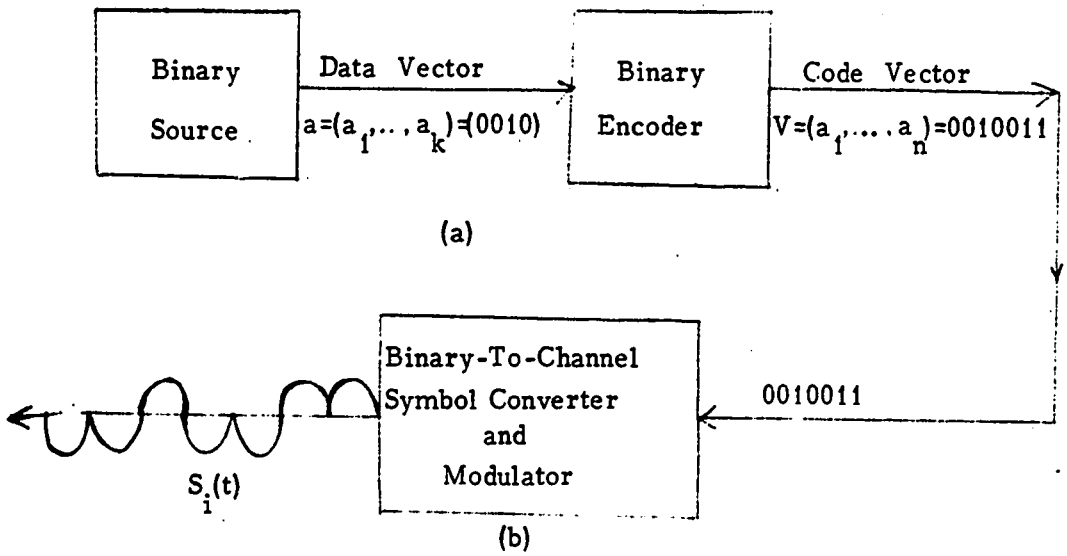


Figure 2.2.

DEFINITION 2.2.1.

Any particular binary n-tuple vector $V = (a_1, a_2, \ldots, a_n)$ which the encoder might transmit is called a code word.

Since the r check digit within any codeword are completely determined by the k message digits only 2^k out of 2^n possible binary sequences are codewords.

DEFINITION 2.2.2.

The set C consisting of the 2^k codewords of length n is called the code.

Suppose now that the symbols 0 and 1 represent the functions f_1 and f_2 respectively, as they are shown in Fig. 2.3, then, the encoded sequence V = (0010011) would cause the modulator to transmit the function $S_1(t)$ shown in Fig. 2.2b. The resulting time function is used to control the amplitude of the transmitted signal, which, for example, might be the voltage on a wire or the instantaneous power emitted by a radio transmitter.

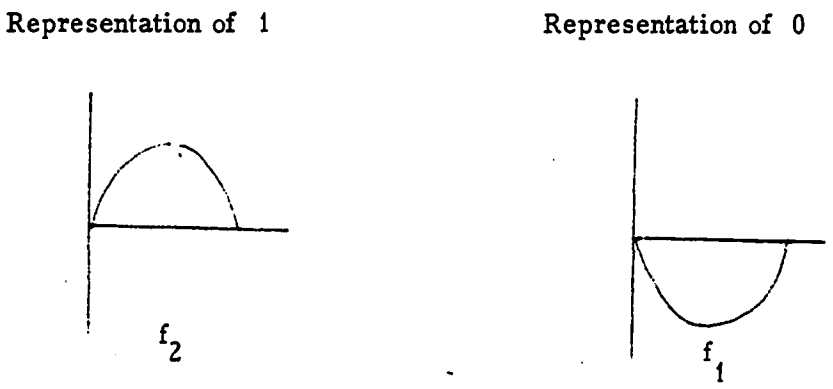


Figure 2.3.

Transmitting the function $S_i(t)$ through the propagation medium, additive noise N(t) might then cause the received signal from the modulator to appear as the function $Z_i(t) = S_i(t) + N(t)$. Suppose now that the noisy version $Z_i(t)$ is the one shown in Fig.2.4a. Then, the signal $Z_i(t)$ enters the receiver where we can obtain the time-quantized version of the noisy representation. Each segment of this signal is then demodulated into a '0' or a '1' whichever is more likely to occur. The resulting sequence received is $R = (b_1, b_2 \ldots, b_n) = (0010101)$. If we examine the transmitted sequence V and the received sequence R we will see that they differ in the 5th and 6th positions. Thus the received sequence R is presented to the decoder, which would then attempt to correct the errors occurring in the fifth and sixth digits (Fig. 2.4b).

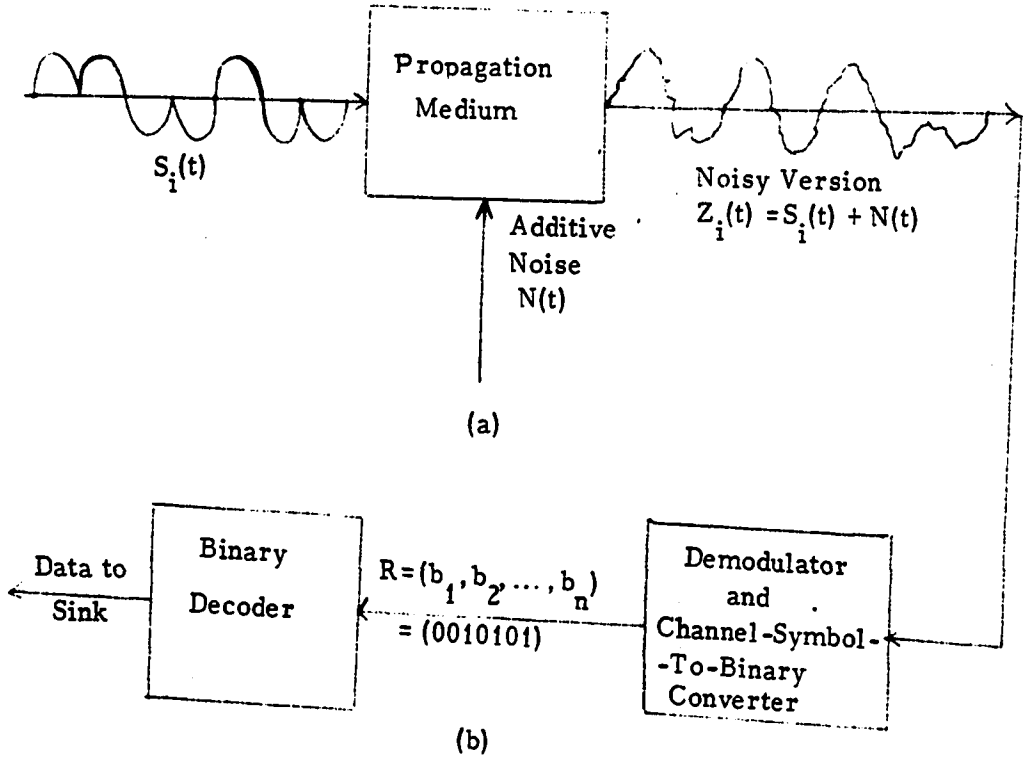


Figure 2.4.

From figures 2.2 and 2.4 one can see that the encoder feeds sequences of 0's and 1's into the modulator and the decoder gets these corrupted versions sequences from the demodulator.

Let us now form the modulo 2 addition of the transmitted sequence 0010011 and the received sequence 0010101.

$$\begin{array}{l} 0010011 \\ \oplus\ \underline{0010101} \\ 0000110 \end{array}$$

As a result we get a third n-tuple $E = (0000110)$ which is called the error sequence or error vector. We observe that the non-zero elements of the error sequence are located at the fifth and sixth positions. Hence, if we take the modulo 2 addition of the received vector and the error vector then we can obtain the true codeword which was transmitted.

DEFINITION 2.2.3.

If the decoder receives the vector $R = (b_1 b_2, \ldots, b_n)$ and it determines that the error vector is $E = (e_1, e_2, \ldots, e_n)$ then the transmitted codeword is the n-tuple

$$V = R + E \qquad 2.2.1$$

where from now on the symbol + stands for modulo 2 addition in case we add binary vectors.

DEFINITION 2.2.4.

Let $V = (a_1 a_2, \ldots, a_n)$ be a binary u-tuple. The weight of V, denoted $|V|$, is the numbers of 1's in V, e.g., if $V = (0010011)$ then $|V| = 3$.

DEFINITION 2.2.5.

If V_1 and V_2 are two binary code vectors, the Hamming distance [38], denoted by $d(V_1, V_2)$, is $|V_1 + V_2|$.

Hence, $d(V_1, V_2)$ is the number of positions in which V_1 and V_2 differ, e.g., if $V_1 = (10101)$ and $V_2 = (10011)$ then $d(V_1, V_2) = 2$ since $V_1 + V_2 = (00110)$ and $|V_1 + V_2| = 2$.

From the above definition we can see that if V is an n-tuple, then $|V| = d(V, 0)$ where the 0 stands for the 0 n-tuple.

DEFINITION 2.2.6.

A block code over the finite alphabet $A_L = (a_0, \ldots, a_{L-1})$ which consists of L distinct letters is a set of M distinct codewords of length n where $2 \leq M \leq L^n$. e.g. with $A_2 = (0, 1)$, $C = (000, 110, 101, 011)$ is a block code of length $n = 3$ containing $M = 4$ codewords.

The codes which are considered in this thesis are binary block codes.

DEFINITION 2.2.7.

The information rate of a code, denoted by R and given in bits of information per encoded digit, is the ratio of the number of information digits to the number of encoded (transmitted) digits, i.e.,

$$R = \frac{k}{n} \qquad 2.2.2.$$

2.3. Some Simple Codes.

The simplest examples of binary codes are the repetition codes, with $k = 1$, r arbitrary and $M = 2$. The two codewords are the sequence of n zeros and the sequence of n ones. The decoder uses the majority rule to decide which codeword was sent. If the number of zeros equals the number of ones, the decoder does not decide and it will commit a decoding failure. If the channel noise changes more than half of the digits in any one block, the decoder will commit a decoding error:

i.e., it will decode the received word into the wrong codeword. Of course, we would prefer correct decoding instead of having a decoding failure or decoding error. A complete decoding algorithm for these codes may be formulated without difficulty and for large n the probability of decoding error is very small but the information rate is very low.

Extreme example of high-rate codes are the single-parity-check codes, with r = 1 and k arbitrary. The check digit is taken to be the mod-2 sum of the codes k information digits. Hence, the weight of every codeword is even and therefore, if the weight of the received word is odd the decoder will commit decoding failure. Any nonzero even number of channel errors will cause a decoding error. thus, the decoder will decode correctly if no channel errors occur in the transmitted codeword.

In 1950 Hamming developed a class of single error correcting (SEC), double error detecting (DED) codes based on the concept of the minimum distance of a code defined as follows:

DEFINITION 2.3.1.

Let C represent a code, which contains M codewords, that is, $C = (V_1, V_2, \ldots, V_M)$ with $2 \leq M \leq 2^n$. The minimum distance of the code is defined to be

$$d_{min} = \min d(V_1, V_2), \quad i \neq j \text{ and } i = 1, \ldots, M; \; j = 1, \ldots, M$$

Hamming developed a relation between the error correcting capability of a code t and the concept of the minimum distance between codewords. He proved that if

$$d_{min} = 2t + 1 \qquad 2.3.1.$$

then the code can correct every error pattern E satisfying the property that

$$|E| \leq t \qquad 2.3.2.$$

A code C denoted as (n,k,t) is a code in which among the n digits of each codeword, there are k information digits and whose error correcting capability is t.

2.4. Linear codes.

The SEC Hamming codes fall into the class of linear codes which are defined as follows:

DEFINITION 2.4.1.

A code C is said to be a linear code if its codewords are the set of all vectors V which satisfy the equation

$$H V^T = 0 \qquad 2.4.1.$$

V^T stands for the transpose of V and H is an r by n matrix which is called the parity-check matrix for C.

Since in the Galois field of n elements, GF(2), the set of all n-tuples forms a vector space, a linear code is a subspace of the space of all n-tuples. Hence, the minimum distance for a linear code equals the minimum weight of its nonzero codewords.

Each row of H corresponds to a parity-check equation that codewords must satisfy and thus, the code C is the null space of the matrix H. If all sets of d - 1 or less columns of H form a linearly independent set, then C is a t error correcting code [3] where

$$t = \left[\frac{d-1}{2}\right] \qquad 2.4.2.$$

DEFINITION 2.4.2.

Let S be an r-tuple vector satisfying the equation

$$HR^T = S^T \qquad 2.4.3.$$

where R is any n-tuple word. Then the vector S is called the syndrome of R.

A coset consists of all the words having a given syndrome. Within each coset, a word having the least weight is chosen as coset leader. A powerful tool for analyzing codes is the so-called standard array [3,4,5,6] which can be constructed by choosing different n-tuples for coset leaders. If the standard array is used as a decoding table, then a received vector R will be decoded correctly into the transmitted vector V if the error patter V + R is a coset leader. This decoding process is extremely laborious in the case where the length of the code is large [3,4] .

DEFINITION 2.4.3.

Let C be a binary linear (n,k) code. Then, the k x n matrix G is called a generator matrix of the code C if the rows of G form a basis for C.

The matrix G can be obtained from H [3] . If we wish to transmit a k information vector I, then we would first multiply it by G to obtain the corresponding codeword

$$V = I G \qquad 2.4.4.$$

Hence, a binary linear code has 2^k words.

A class of codes which can correct t or less random errors was developed indepedently by Hocquenyhem [8] in 1959 and by Bose and Chaudhuri [7] in 1960. This class is a subset of the class of cyclic codes. The class of cyclic codes is a subset of the class of linear codes.

DEFINITION 2.4.4.

Let C be a linear code (n,k). If whenever $(a_0, a_1, \ldots, a_{n-1}) \in C$, $(a_{n-1}, a_0, a_1, \ldots a_{n-2})$ also belongs to C then C is called a cyclic code.

2.5. Polynomial Representation of Cyclic Codes.

We can make use of polynomials to represent codewords. This is done by associating the polynomial

$$V(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$$

with the n-tuple

$$V = (a_0, a_1, \ldots, a_{n-1})$$

where $a_1 \in G\ F(2)$. Then, there is a one-to-one correspondence between the binary sequence V and the coefficients of $V(x)$.

A cyclic code is completely specified by a polynomial $g(x)$ that divides $1 + x^n$ which is called the generator polynomial of the code [3,4,5,6] If C is a cyclic code and $V(x) \in C$ then $x^i V(x) \bmod 1 + x^n$ is again in C. Furthermore every $V(x) \in C$ can be written as

$$V(x) = g(x)\, I(x) \qquad 2.5.1.$$

where $\deg V(x) \leq n-1$

$\deg g(x) = n-k$

and $\deg I(x) \leq k-1$

The polynomial $I(x)$ is called the information polynomial.

The BCH codes are an important class of cyclic codes. The error correcting capability of the BCH cyclic codes is specified by the roots of the generator polynomial $g(x)$ [3,4,5,6] . Furthermore, let $P_i(x)$'s be all the irreducible factors of the polynomial $1 + x^n$ with

$$nc = 2^q - 1$$

where c and q are positive integers. If α is a primitive root in $G\ F(2^q)$ then

$$\alpha^c, \alpha^{2c}, \alpha^{3c}, \ldots, \alpha^{(n-1)c}$$

are the roots of $P_i(x)$'s and

$$g(x) = L\ C\ M\ [P_1(x), P_3(x), \ldots, P_{2t-1}(x)]$$

$$= P_1(x)\ P_3(x) \ldots, P_{2t-1}(x) \qquad 2.5.2.$$

The code generated by the generator polynomial g(x) which is derived from formula 2.5.2 will be a t error correcting code. In general for any q and t there is a BCH binary code of length $2^q - 1$ which corrects all combinations of t or fewer errors and has no more than qt parity-check symbols.

The generator matrix G of an (n, k) cyclic code generated by g(x) is given by

$$G = \begin{bmatrix} g(x) \\ xg(x) \\ \cdot \\ \cdot \\ \cdot \\ x^{k-1}g(x) \end{bmatrix} \qquad 2.5.3.$$

and the row space of that matrix is the code C.

A cyclic code can as well be defined by the parity-check polynomial

$$h(x) = \frac{1+x^n}{g(x)}$$

of degree k. The parity-check matrix of the code will be

$$H = \begin{bmatrix} h^*(x) \\ xh^*(x) \\ \cdot \\ \cdot \\ \cdot \\ x^{n-k-1}h^*(x) \end{bmatrix} \qquad 2.5.4.$$

where the $h^*(x)$ stands for the reciprocal of h(x) .

DEFINITION 2.5.1.

Let C be an (n, k) cyclic code generated by g(x). Then the code C^* is called the dual of C if it is generated by $h^*(x)$.

Hence the code C^* has dimension n-k and it is (n, n-k). The row space of C is the null space of C^* and conversely.

In 1961 W.W. Peterson [19] provided a decoding procedure for the BCH codes. These codes are designed for random error correction and are very efficient for such correction but they have poor burst error correcting capability.

DEFINITION 2.5.2.

An error of the form, $(00\ldots1\, a_{i_2}\, a_{i_3} \ldots a_{i_{L-1}}\, 100\ldots0)$ with $a_{i_k} = 0$ or 1, is called a burst of length L starting in position i_1.

Burst errors occur very frequently in telephone and recording systems.

Linear codes are a special case of group codes which are defined as follows:

DEFINITION 2.5.3.

A code C whose elements form a group [39,40,41] under modulo 2 addition is called a group code.

In the case where the elements of C are binary n-tuples a group code is a linear code [3] . Hence, since in this thesis we will be mainly concerned with binary codes we will use the terms group codes and linear codes interchangeably.

We close this chapter with the following

DEFINITION 2.5.4.

The number of parity-check digits within a codeword belonging to the code C and usually denoted by r is called the redundancy of the code. This equals the dimension n-k of the parity-check matrix H or, alternately, the degree of the generator polynomial g(x) of the code C.

e.g. The redundancy r of the (7,4) code is r = n-k = 7 - 4 = 3.

DEFINITION 2.5.5.

The dimension of a code is the number of its information symbols k. This equals the dimension of the generator matrix G or, alternately, the degree of the parity-check polynomial h(x) of the code C, e.g. The dimension k of the (7,4) code is $k = 4$.

CHAPTER 3

DIRECT-PRODUCT CODES AND RELATED CODES

3.0. Introduction.

This chapter is dedicated to the large class of two-dimensional codes. It is shown that we can modify or combine other codes to make up a more powerful code such as tensor-product codes, direct-sum codes, direct-product codes, etc. . The class of direct-product codes has been discussed more extensively. It has been stated that the cyclic product codes have the advantages of cyclic codes and the structural properties of product codes. The product of two cyclic codes is itself a cyclic code provided that certain conditions are satisfied, and, furthermore, their generator polynomial can be easily derived from the generator polynomials of the component codes. Cyclic product codes are capable of unambiguous correction of both bursts and random errors. Using,for example, the cascade decoding procedure for product codes, i.e., decoding the component codes one at a time, it has been shown [31] that the product codes do not reach their error correcting capability guaranteed by their minimum distance. Instead, it is possible to correct many error-patterns beyond the error correcting capability of the product code. Hence, having a cascade decoder rather than a conventional one we get the profit of correcting a large number of errors, by paying the price of not being able to correct a small percentage of error patterns guaranteed correctable by using a conventional decoder in a straightforward manner. If a cyclic product code guarantees correction of any error pattern containing 4 single errors or less by using any of the conventional decoding algorithms, then by using the cascade decoding algorithm we may correct any error pattern containing 3 single errors or less, 99.97 percent of the error patterns containing 4 single errors, 99.85 percent of the error patterns containing 5 single errors,..., 99.37 percent of the error

patterns containing 10 single errors, etc. It is possible to correct any pattern of 4 errors only in case where at least, one of the component codes is 1-step MLD. Again, many other errors of weight greater than 4 will be corrected.

Hence, the product codes are good burst correctors. Consequently these codes effect a compromise between strictly burst-correcting and random-error-correcting codes and should, therefore allow the communications engineer some additional freedom in designing error-control systems. Nevertheless, these codes can be used on channels in which both types of errors occur as in switched telephone points, magnetic tapes, etc. .

3.1. Kronecker Product of codes.

Let (n_1, k_1) and (n_2, k_2) be two linear binary codes C_1 and C_2 respectively. If $V_1 \in C_1$ and $V_2 \in C_2$ are two codes words then the Kronecker product of V_1 by V_2, denoted by $V_1 \times V_2$, is a vector.

$$V = V_1 \times V_2$$

which is formed by substituting V_2 for 1 and $\bar{V}_2$ (the complement of V_2) for 0 in V_1.

The codeword V has length $n_1 \times n_2$. Therefore, it is sometimes advantageous to write the coefficients of V in an $n_1 \times n_2$ array. A column of the array of its complement will belong to C_1 and a row or its complement will belong to C_2.

EXAMPLE 3.1.1.

Let V_1 =(1011000) and V_2 = (10110) be two code words of the C_1 =(7,4) and C_2 = (5, 3) codes respectively.

Then, the Kronecker product V of the V_1 and V_2 is

V = 10110010011011010110010010100101001

or, by making use of a 7 x 5 array, V can be written as

$$V = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}_{7 \times 5}.$$

Hence, we can make use of the two dimensional representation to express a one-dimensional code word.

The two-dimensional codes make up a large class of codes derived by modifying or combining other codes. For example, there are the tensor-product codes, the direct-sum codes, the direct-product codes, etc.

3.2. Tensor product codes.

Let us say that H_1 is a parity check matrix of dimension r x n over $GF(2^m)$. This matrix may be expanded [4] into a parity-check matrix H_2 of dimension mr x n over GF(2). If the matrices H_1 and H_2 are the parity-check matrices of the codes C_1' and C_2' respectively, then the code C_2' over GF(2) is said to be the expansion of C_1' over $GF(2^m)$. Alternately, the C_1' is said to be the contraction of C_2'.

DEFINITION 3.2.1.

If C_1 and C_2 are codes over $GF(2^m)$ and C_2 is the expansion of C_3 over GF(2), then the tensor product of the codes C_1 and C_3, denoted by $C_1 \otimes C_3$ is the expansion of the dual of the direct product (see definition 3.4.1) of the duals of C_1 and C_2; i.e.,

$$C_1 \otimes C_3 = \text{expansion of } (C_1^* \times C_2^*)^*$$

where the codes C_1^* and C_2^* are the duals of the codes C_1 and C_2 respectively. If the codes of C_1 and C_3 have lengths n_1 and n_3 and redundancies r_1 and r_3 respectively then, the tensor product code $C_1 \otimes C_3$ has length $n_1 n_3$ and redundancy $r_1 r_3$.

In practice, C_3 is usually taken to be a code with $r_3 = m$. C_1 is often taken to be a Reed-Solomon Code.

Wolf [27] has shown that by using certain tensor product codes it is possible to correct multiple burst errors. Tensor product codes may be used in a two-way Communication System. Wolf and Elspas [42], and Wolf [43], have suggested that the tensor product codes can be treated as error-locating codes (EL codes), and these codes are proposed for a third coding technique which is called error-location (EL) coding.

EL codes permit the location of digit errors to within a sub-block of the received massage block without specifying the precise location of erroneous digit positions. The receiver is able to specify which particular sub-blocks contain errors .. so that it could then request the sender to retransmit the erroneous received sub-blocks. The error-location coding lies midway between error-detection and error-correction coding.

3.3. Direct Sum Codes.

If V_1 and V_2 are two binary sequences, say, $V_1 = a_1 a_2 \ldots a_j$ and $V_2 = b_1 b_2 \ldots b_k$, then we may form the concatenation [4] of V_1 and V_2, denoted by $V_1 * V_2$, as

$$V_1 * V_2 = a_1 a_2 \ldots a_j \; b_1 b_2 \ldots b_k$$

Let (n_1, k_1) and (n_2, k_2) be two codes C_1 and C_2 respectively over the same field.

If $V_i \in C_1$ and $V_j \in C_2$ represent two codewords, then the direct-sum code C of the codes C_1 and C_2, denoted by $C_1 \oplus C_2$, consists of the set of all words of the form $V_i * V_j$.

The direct-sum code C is the code $(n_1+n_2,\ k_1+k_2)$ and its minimum distance is the minimum of the minimum distances of C_1 and C_2.

Suppose that the code D is equivalent under some permutation to a code C, which can be written as a direct-sum $C = C_1 \oplus C_2$.

Then the code D is said to be a decomposable code.

It has been shown [24] that there is no decomposable linear code with larger minimum distance or smaller probability of error than the best indecomposable linear code of the same rate and block length.

3.4. DIRECT PRODUCT CODES.

3.4.0. Introduction .

The class of direct product codes was first introduced by Elias [23]. This is the class of two-dimensional codes which has been studied most extensively.

DEFINITION 3.4.1.

If A is a linear code of length n_1 and B is a linear code of length n_2, the direct product C of the codes A and B denoted by

$$C = A \times B$$

is the two-dimensional code consisting of all codewords of length $n = n_1 n_2$ which satisfy the following two conditions:

1. For all j, $(C_{0,j}, C_{1,j}, \ldots, C_{n_1-1,j}) \in A,\ j = 0, 1, \ldots, n_2-1$

2. For all i, $(C_{i,0}, C_{i,1}, \ldots, C_{i,n_2-1}) \in B,\ i = 0, 1, \ldots, n_1-1$

3.4.0.1.

According to the above definition the conditions 3.4.0.1. can be translated into the following matrix of dimension $n_1 \times n_2$ of which the entries will be $C_{i,j}$ with $i = 0, 1, \ldots, n_1-1$ and $j = 0, 1, \ldots, n_2-1$.

$$M = \begin{bmatrix} C_{0,0} & C_{0,1} & \cdots & C_{0,n_2-1} \\ C_{1,0} & C_{1,1} & \cdots & C_{1,n_2-1} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & C_{i,j} & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ C_{n_1-1,0} & C_{n_1-1,1} & \cdots & C_{n_1-1,n_2-1} \end{bmatrix} \qquad 3.4.0.2.$$

Hence, every column of the matrix 3.4.0.2. is a code word from the component code A and every row is a code word from the component code B, that is, if $A_j(X)$ and $B_i(X)$ are code words from the codes A and B respectively then

$$A_j(X) = C_{0,j}+C_{1,j}X+C_{2,j}X^2+\ldots+C_{n_1-1,j}X^{n_1-1}, \quad j = 0,1,\ldots,n_2-1$$

and

$$B_i(X) = C_{i,0}+C_{i,1}X+C_{i,2}X^2+\ldots+C_{i,n_2-1}X^{n_2-1}, \quad i = 0,1,\ldots n_1-1 \qquad 3.4.0.3.$$

Let $V(X)$ be a code word of the product code C. Then

$$V(x) = C_0+C_1X+C_2X^2+\ldots+C_\ell X^\ell+\ldots+C_{n_1n_2-1}X^{n_1n_2-1} \qquad 3.4.0.4.$$

or, using one-dimensional representation for the coefficients of $V(x)$,

$$V = (C_0,C_1,C_2,\ldots,C_\ell,\ldots C_{n_1n_2-1}) \qquad 3.4.0.5.$$

If the digits of the product code word V of the matrix 3.4.0.2 are to be transmitted consecutively, then the one-dimensional nature of time forces us to identify the components of that $n_1 \times n_2$ array with the components of the above $1 \times n_1n_2$ array 3.4.0.5. This can be done in various ways.

3.4.1. Canonical ordering.

There is a way of arranging the 3.4.0.5. into the matrix 3.4.0.2 which is called canonical ordering [4]. For every $C_{i,j}$ corresponds a unique C_ℓ

$$C_{i,j} \longleftrightarrow C_\ell$$

and, in order to identify $C_{i,j}$ with C_ℓ we must define a mapping $m(i,j)$ such that it relates the (i,j) element of the array V with the coefficient of the $(1+\ell)$th term of the polynomial $V(X)$, i.e.

$$\ell = m(i,j) \qquad 3.4.1.1.$$

The mapping m is defined as

$$\ell = in_2 + j \qquad 3.4.1.2.$$

Hence, canonical ordering is obtained by choosing i and j as the quotient and remainder when ℓ is divided by n_2. Applying the mapping m to the positions of the components of the vector V we get the following array

$$V_1 = \begin{bmatrix} C_0 & C_1 & \cdots & C_{n_2-1} \\ C_{n_2} & C_{n_2+1} & \cdots & C_{2n_2-1} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ C_{(n_1-1)n_2} & C_{(n_1-1)n_2+1} & \cdots & C_{n_1n_2-1} \end{bmatrix} \qquad 3.4.1.3$$

EXAMPLE 3.4.1.1.

If $n_1 = 3$ and $n_2 = 7$, we have

$$V_1 = \begin{bmatrix} C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\ C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} \\ C_{14} & C_{15} & C_{16} & C_{17} & C_{18} & C_{19} & C_{20} \end{bmatrix}$$

Hence, the order in which the digits of the array are transmitted is row by row. If we apply the mapping m' (j,i), where

$$\ell = m'(j,i)$$

with

$$\ell = jn_2 + i \qquad 3.4.1.4.$$

then the array 3.4.1.3 becomes

$$V_2 = V_1^T \qquad 3.4.1.5$$

where V_1^T stands for the transpose of V_1.

In this case the order in which the digits of the array are transmitted is column by column.

Practically speaking, when the digits of the array are arranged in canonical order, these are assumed to be transmitted either row or column by column. Another way or ordering the coefficients of the product codeword into a two-simensional array, the cyclic ordering, will be discussed later. Until then, some basic theorems concerning direct-product codes are presented. Let (n_1,k_1) be the column-component code with distance d_1 and (n_2,k_2) the row component code with minimum distance d_2. The following theorem specifies the minimum distance of the product code with regard to the minimum distances of its component codes.

THEOREM 3.4.1.

The minimum distance d of a direct product code equals the product of the minimum distances of the component codes; that is,

$$d = d_1 d_2 \qquad 3.4.1.6.$$

Proof: Suppose V in 3.4.0.2. is not an all-zero matrix. Then there must be at least one row, say B_i, with at least d_2 ones. The columns

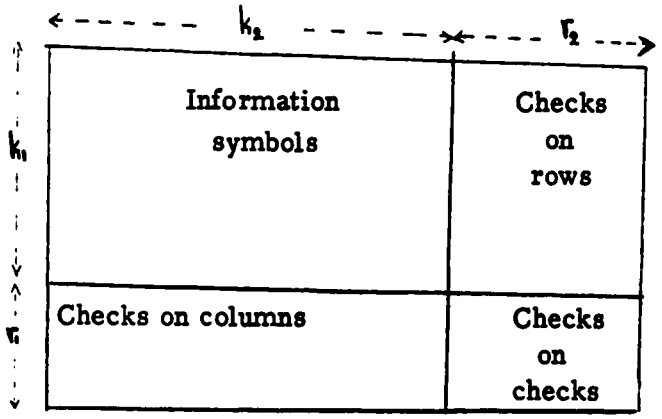
containing these d_2 ones are obviously not all-zero columns. Therefore each of these columns must have at least d_1 ones. Thus, there are at least d_2 columns each containing d_1 ones and hence the minimum distance is $d = d_1 d_2$ Q.E.D.

THEOREM 3.4.2.

The dimension k of a direct product code equals the product of the dimensions of the component codes; that is,

$$k = k_1 k_2 \qquad 3.4.1.7.$$

Proof: Consider $k_1 k_2$ information symbols arranged in a rectangular array in the upper left-hand corner of the figure 3.1. and distributed among k_1 rows and k_2 columns. Applying the parity-check rules for the row code we fill in the $k_1 r_2$ check digits in the upper right-hand corner of the $n_1 \times n_2$ array. In a similar manner we fill in the $k_2 r_1$ check



digits in the lower left-hand corner of that array, by applying the parity-check rules for the column code. The $r_1 \times r_2$ digits in the lower right-hand corner can be filled in as checks on rows and will be consistent as checks on columns, or vice versa. If they are filled in as checks on rows, then each of the last r_2 columns is actually a linear combination of the first k_2 columns that contain information symbols. Each of these r_2 parity-check columns, has parity symbols added to it to make it a codeword and, since they are linear combinations of code vecotrs for the column code, they are also code vectors for the column code. In an analogous manner it can be verified that the last r_1 parity-check rows constitute code vectors

belonging to the row-component code.

Q.E.D.

From theorems 3.4.1. and 3.4.2. one concludes that if an (n,k) code is the product of an (n_1,k_1) column code and an (n_2,k_2) row code, then it has block length $n = n_1n_2$, number of information symbols $k = k_1k_2$ and minimum Hamming distance $d = d_1d_2$. Clearly the code is capable of correcting t errors where

$$t = \left[\frac{d_1d_2 - 1}{2}\right] \qquad 3.4.1.8.$$

Hence, an important observation is that the properties of the product code depend on the properties of the component codes. If R_1 and R_2 are the transmission rates of the component codes then the transmission rate R of the product code is the product of R_1 and R_2, for

$$R = \frac{k}{n} = \frac{k_1k_2}{n_1n_2} = R_1R_2 \qquad 3.4.1.9.$$

Then 3.4.1.9. indicates that if we want to construct a product code with high rate it is required to use component codes with even higher rates. Although it is difficult to construct a product code with high transmission rate by using very powerful component codes, it is possible to obtain a product code with moderate transmission rate by using reasonably powerful component codes.

The following theorem gives the conditions on the array 3.4.0.2. and a definition of the mapping 3.4.1.1. which result in a cyclic product code having cyclic component codes.

THEOREM 3.4.3. [Burton-Weldon (1965)]

The product of two cyclic codes is itself cyclic code provided the

lengths of the component codes are relatively prime, i.e., $an_1 + bn_2 = 1$ for some integers a and b and, the mapping m(i,j) being defined as

$$m(i,j) \equiv (j-i)\, an_1 + i \bmod n_1 n_2,\ m(i,j) = 0, 1 \ldots, n_1 n_2 - 1 \qquad 3.4.1.10.$$

Before proceeding with the proof of this theorem it is convenient to prove the following lemma:

LEMMA 3.4.1.

$$m(i,j) = m(i',j') \qquad 3.4.1.11.$$

if and only if

$$i \equiv i' \bmod n_1 \text{ and } j \equiv j' \bmod n_2 \qquad 3.4.1.12.$$

Proof:

3.4.1.11. 3.4.1.12.

The m(i,j) is defined in 3.4.1.10. According for m(i',j') we get

$$m(i',j') \equiv (j'-i')\, an_1 + i' \bmod n_1 n_2 \qquad 3.4.1.13.$$

Combining the 3.4.1.10. and 3.4.1.13. we have

$$m(i,j) - m(i',j') \equiv (j-i)\, an_1 + i - (j'-i')\, an_1 - i' \bmod n_1 n_2$$

The later relationship yields because of the 3.4.1.11.,

$$(j-i)\, an_1 + i - (j'-i')\, an_1 - i' \equiv 0 \bmod n_1 n_2$$

Rearranging, and since $1 - an_1 = bn_2$, this is equivalent to

$$(j-j')\, an_1 + (i-i')\, bn_2 \equiv 0 \bmod n_1 n_2 \qquad 3.4.1.14.$$

Therefore, the left-hand side of the above relationship must be a multiple of $n_1 n_2$ and, hence,

$n_1 / i-i'$ and $n_2 / j-j'$

or $i-i' \equiv 0 \bmod n_1$ and $j-j' \equiv 0 \bmod n_2$, from which the 3.4.1.12 follows.

3.4.1.11 3.4.1.12.

The statements 3.4.1.12 yield

$$i-i' \equiv \text{mod} n_1 \text{ and } j-j' \equiv 0 \text{ mod } n_2$$

or

$$(i-i')\, bn_2 \equiv 0 \text{ mod } n_1 n_2 \text{ and } (j-j')\, an_1 \equiv 0 \text{ mod } n_1 n_2$$

Hence,

$$(i-i')\, bn_2 + (j-j,)\, an_1 \equiv 0 \text{ mod } n_1 n_2 \text{ or, since } bn_2 = 1-an_1 ,$$

$$[(j-i)\, an_1 + i] - [(j-i')\, an_1 + i] \equiv 0 \text{ mod } n_1 n_2.$$

Therefore,

$$(j-i)\, an_1 + i \equiv (j'-i')\, an_1 + i' \text{ mod } n_1 n_2$$

or

$$m(i,j) \equiv m(i',j') \text{ mod } n_1 n_2$$

But $m(i,j),\ m(i',j') < n_1 n_2$ so that

$$m(i,j) = m(i',j').$$

Proof of theorem 3.4.3. Lemma 3.4.1. leads to the conclusion that if the lengths n_1 and n_2 of the component codes are relatively prime then the mapping relating position (i,j) of the two-dimensional array with $(1 + \ell)$th term of the one-dimensional code word is unique. In order to prove that the code is cyclic it is sufficient to show that a cyclic shift of a code word is again a code word or, alternately, that every row and column is a cyclic shift of a row and column of the original array. Therefore,it is sufficient to show that

$$m(i,j) + 1 \equiv m(i+1,\ j+1) \text{ mod } n_1 n_2 \qquad 3.4.1.15$$

where i+1 and j+1 are to be interpeted as (i+1) mod n_1 and (j+i) modn_2 respectively.

According to the definition 3.4.1.10 the right-hand side of 3.4.1.15. yields

$$\begin{aligned} m(i+1,\ j+1) &\equiv (j-i)\, an_1 + i+1 \text{ mod } n_1 n_2 \\ &\equiv [\lceil (j-i)\, an_1 + i \rceil \text{ mod } n_1 n_2 + 1] \text{ mod } n_1 n_2 \\ &\equiv m(i,j) + 1 \text{ mod } n_1 n_2 \end{aligned}$$

Q.E.D.

Before proceeding further it is wise to simplify the mapping 3.4.1.10. by defining another equivalent mapping.

DEFINTION 3.4.2.

The mapping m such that

$$\ell = m(i,j),$$

with ℓ and m(i,j) defined as in section 3.4.1., is defined as

$$\ell \equiv i \bmod n_1$$

and $\ell \equiv j \bmod n_2$ 3.4.1.16.

provided that

$$(n_1, n_2) = 1$$

ASSERTION 3.4.1.

Definitions 3.4.1.10 and 3.4.1.16 are equivalent.

Proof: 3.4.1.10, $\Longrightarrow$ 3.4.1.16.

Since $\ell = m(i,j)$ the 3.4.1.10 yields

$$\ell \equiv j\,an_1 + ibn_2 \bmod n_1 n_2$$

or

$$\ell = qn_1 n_2 + jan_1 + ibn_2 \quad \text{for some } q$$

or $\ell \equiv j\,an_1 \bmod n_2$

or $\ell \equiv j \bmod n_2$, for $an_1 + bn_2 = 1$.

Similarly we get $\ell \equiv i \bmod n_1$.

3.4.1.10. $\Longleftarrow$ 3.4.1.16.

From 3.4.1.16 we get

$$\ell = q_1 n_1 + i$$

and $\ell = q_2 n_2 + j$ 3.4.1.17.

for some q_1 and q_2.

Combining these together we obtain:

$$(j-i)\,an_1 - q_1 an_1^2 \equiv o \bmod n_1 n_2 \qquad 3.4.1.18.$$

Then $an_1 + bn_2 = 1$ in combination with the first of 3.4.1.17 yields that:

$$q_1 an_1^2 = \ell - i - q_1 bn_1 n_2$$

and substituting into 3.4.1.18 we have

$$(j - i)\, an_1 + i \equiv \ell - q_1 bn_1 n_2 \bmod n_1 n_2$$

or $(j - i)\, an_1 + i \equiv \ell \bmod n_1 n_2$

i.e., $m(i,j) \equiv \ell \bmod n_1 n_2$

But $m(i,j),\ \ell < n_1 n_2$

Hence, $m(i,j) = \ell$

Q.E.D.

3.4.2. Cyclic ordering.

In Section 3.4.1. we showed that a product code can always be arranged in canonical order. Theorem 3.4.3. enables us to use another way of ordering the coefficients of the product code word V into the two-dimensional array V, provided that the product code is cyclic. This ordering is called cyclic ordering [4,25] and it is defined by the mapping 3.4.1.10. or its equivalent 3.4.1.16. Hence, the cyclic ordering is obtained by choosing i and j as the remainders when ℓ is divided by n_1 and n_2 respectively. The cyclic ordering maps only one value of ℓ, $0 \le \ell < n_1 n_2$, into any given pair (i,j) of the two-dimensional array, $0 \le i < n_1$, $0 \le j < n_2$. If n_1 and n_2 are not relatively prime, the cyclic ordering is undefined.

EXAMPLE 3.4.2.1. Let us take again $n_1 = 3$ and $n_2 = 7$. We apply the mapping 3.4.1.16 to the product code word 3.4.5. to get the following array in cyclic order :

$$V_2 = \begin{bmatrix} C_0 & C_{15} & C_9 & C_3 & C_{18} & C_{12} & C_6 \\ C_7 & C_1 & C_{16} & C_{10} & C_4 & C_{19} & C_{13} \\ C_{14} & C_8 & C_2 & C_{17} & C_{11} & C_5 & C_{20} \end{bmatrix}$$

EXAMPLE 3.4.2.2. If $n_1 = 3$ and $n_2 = 5$ then the one-dimensional product code word $V(X)$ will be $V(X) = C_0 + C_1X + C_2X^2 + C_3X^3 + ..$

$.. + C_{13}X^{13} + C_{14}X^{14}$.

If we want to arrange that in canonical ordering we apply the mapping 3.4.1.2. to obtain the following two-dimensional representation:

$$V_1 = \begin{bmatrix} C_0 & C_1 & C_2 & C_3 & C_4 \\ C_5 & C_6 & C_7 & C_8 & C_9 \\ C_{10} & C_{11} & C_{12} & C_{13} & C_{14} \end{bmatrix}$$

If cyclic transmission is assumed then we apply mapping 3.4.1.16. to obtain the following cyclic ordering :

$$V_2 = \begin{bmatrix} C_0 & C_6 & C_{12} & C_3 & C_9 \\ C_{10} & C_1 & C_7 & C_{13} & C_4 \\ C_5 & C_{11} & C_2 & C_8 & C_{14} \end{bmatrix}$$

The following important theorem is also due to Burton and Weldon [25] .

THEOREM 3.4.4.

Let $g_1(x)$ and $g_2(x)$ be the generator polynomials of the component codes A and B respectively. Then, the generator polynomial $g(x)$ of the product code $C = A \times B$ is given by

$$g(x) = gCd \left[g_2(x^{an_1}) \, g_1(x^{bn_2}),\ 1 + x^{n_1 n_2} \right]$$

where gcd stands for the greatest common divisor and $an_1 + bn_2 = 1$

Proof: Let $A_j(x) \in A$ and $B_i(x) \in B$ denote the jth column and ith row of the array 3.4.0.2. Then, equations 3.4.0.3 can be written as

$$A_j(x) = \sum_{i=o}^{n_1-1} C_{ij} x^i, \quad j = 0, 1, \ldots, n_2-1$$

and

$$B_i(x) = \sum_{j=o}^{n_2-1} C_{ij} x^j, \quad i = 0, 1, \ldots, n_1-1 \qquad 3.4.1.19$$

Also, if $V(x) \in C$ then,

$$V(x) = \sum_{i=o}^{n_1-1} \sum_{j=o}^{n_2-1} C_{ij} x^{m(i,j)} \qquad 3.4.1.20.$$

Substituting 3.4.1.10. into 3.4.1.20. and using the second equations of 3.4.1.19. we get after a few steps that

$$V(x) \equiv \sum_i \sum_j C_{ij} x^{(j-i)an_1+i} \mod 1 + x^{n_1 n_2}$$

$$\equiv \sum_i x^{ibn_2} \sum_j C_{ij} x^{jan_1} \mod 1 + x^{n_1 n_2}$$

$$\equiv \sum_i x^{ibn_2} \beta_i(x^{an_1}) \mod 1 + x^{n_1 n_2}$$

Hence,

$$V(x) = g(x)\,(1+x^{n_1 n_2}) + \sum_i x^{ibn_2}\,\beta_i(x^{an_1}) \text{ for some } q(x)$$

$$= g(x)\,I(x)$$

But, since $g_2(x) \,/\, \beta_i(x)$ then $g_2(x^{an_1}) \,/\, \beta_i(x^{an_1})$

and therefore the $\gcd\,[\,g_2(x^{an_1}),\ 1+x^{n_1 n_2}\,]$ divides every code word $V(x)$ of the product code C

Thus,

$$\gcd\,[\,g_2(x^{an_1}),\ 1+x^{n_1 n_2}\,] \,/\, g(x) \qquad 3.4.1.21.$$

Similarly we get, if the first equation of 3.4.1.19. has been taken into account, that

$$V(x) \equiv \sum_j x^{jan_1} \sum_i C_{ij}\,x^{ibn_2} \mod 1+x^{n_1 n_2}$$

$$\equiv \sum_j x^{jan_1}\,A_j(x^{bn_2}) \mod 1+x^{n_1 n_2}$$

Hence,

$$V(x) = g(x)\,(1+x^{n_1 n_2}) + \sum_j x^{jan_1}\,A_j(x^{bn_2})$$

But, $g_1(x^{bn_2}) \,/\, A_j(x^{bn_2})$) because $g_1(x) \,/\, A_j(x)$.

Therefore the $\gcd\,[g_1(x^{bn_2}),\ 1+x^{n_1 n_2}]$ divides every code word $V(x) \in C$ and thus,

$$\gcd\,[\,g_1(x^{bn_2}),\ 1+x^{n_1 n_2}\,] \,/\, g(x) \qquad 3.4.1.22.$$

From 3.4.1.21. and 3.4.1.22. one concludes that if $L(x)$ is their least common mutliple (ℓ cm) then

$$L(x) = \ell\text{cm}\,\{\gcd\,[\,g_2(x^{an_1}), 1+x^{n_1 n_2}], \gcd\,[g_1(x^{bn_2}), 1+x^{n_1 n_2}]\}/g(x)$$

3.4.1.23.

Suppose now that

$$g_1(x) = \sum_i p_i x^i$$

and

$$g_2(x) = \sum_j q_j x^j \qquad 3.4.1.24.$$

and let us consider the array with column

$$A_i(x) = g_j g_1(x)$$

This array has rows 3.4.1.25.

$$B_i(x) = p_i g_2(x)$$

Hence,

$$\begin{aligned} V(x) &\equiv \sum_i x^{ibn_2} \beta_i(x^{an_1}) \mod 1 + x^{n_1 n_2} \\ &\equiv \sum_i x^{ibn_2} p_i g_2(x^{an_1}) \mod 1 + x^{n_1 n_2} \\ &\equiv g_2(x^{an_1}) \sum_i p_i x^{ibn_2} \mod 1 + x^{n_1 n_2} \\ &\equiv g_2(x^{an_1}) g_1(x^{bn_2}) \mod 1 + x^{n_1 n_2}, \end{aligned}$$

for 3.4.1.25 and the first of 3.4.1.24 yield respectively that

$$\beta_i(x^{an_1}) = p_i g_2(x^{an_1}) \text{ and } g_1(x^{bn_2}) = \sum_i p_i x^{ibn_2}.$$

Therefore,

$$V(x) = q(x)(1+x^{n_1 n_2}) + g_2(x^{an_1}) q_1(x^{bn_2})$$

for some $q(x)$ and since $g(x) / V(x)$ and $g(x) / 1+x^{n_1 n_2}$ then $g(x) / g_2(x^{an_1}) g_1(x^{bn_2})$ and so we conclude that

$$g(x) / G(x) = \gcd [g_2(x^{an_1}) g_1(x^{bn_2}), 1+x^{n_1 n_2}] \qquad 3.4.1.26.$$

So far, it has been proved that

$$L(x) \,/\, g(x) \quad \text{and} \quad g(x) \,/\, G(x) \qquad 3.4.1.27.$$

Hence,

$$L(x) \,/\, G(x)$$

Comparing 3.4.1.26 with 3.4.1.23 one concludes that

$$L(x) = G(x)$$

if no factor of $1 + x^{n_1 n_2}$ is also a factor of both $g_1(x^{an_1})$ and $g_2(x^{bn_2})$ so as not to have greater multiplicity in G(x) than in L(x).

Indeed, suppose that the code words are polynomials with coefficients from the prime field with p elements, and suppose $n_1 n_2 = p^s n$ where n and p are relatively prime and s is a positive integer or zero. Then

$$1 + x^{n_1 n_2} = (1 + x^n)^{p^s} . \qquad 3.4.1.28.$$

Since $1 + x^n$ has no repeated factors then every irreducible factor of $1 + x^{n_1 n_2}$ is repeated exactly p^s times. But $p^s / n_1 n_2$ and $(n_1 n_2) = 1$. Hence, we can write that either

$$n_1 = p^s n_1' \quad \text{with } (n_2, p^s) = 1$$

or

$$n_2 = p^s n_2' \quad \text{with } (n_1, p^s) = 1 .$$

In the case where $n_1 = p^s n_1'$, we obtain that

$$g_2(x^{an_1}) = [\, g_2(x^{an_1'}) \,]^{p^s} \qquad 3.4.1.29.$$

and in the case where $n_2 = p^s n_2'$ we obtain that

$$g_1(x^{bn_2}) = [\, g_1(x^{bn_2'}) \,]^{p^s} \qquad 3.4.1.30.$$

Therefore,

$$g_2(x^{an_1})\, g_1(x^{bn_2}) = [g_2(x^{an_1'})\, g_1(x^{bn_2'})]^{p^s} \qquad 3.4.1.31.$$

From 3.4.1.28, 3.4.1.29 and 3.4.1.30 concludes that every common factor of the left-hand side terms of the above statements has multiplicity p^s whereas it has multiplicity $2p^s$ in 3.4.1.31. From 3.4.1.23. and 3.4.1.26 we conclude that these common factors have also multiplicity p^s in L(x) and G(x). Hence, L(x) = G(x) and 3.4.1.27. yields

$$G(x) \,/\, g(x) \quad \text{and} \quad g(x) \,/\, G(x)$$

Finally

$$g(x) = G(x)$$

Q.E.D.

COROLLARY 3.4.1.

In the notation of theorem 3.4.4

$$g(x) = \text{lcm}\,[\, G_1(x^{n_2}),\ G_2(x^{n_1})\,]$$

where

$$G_1(x) = \gcd\,[g_1(x^b),\ 1 + x^{n_1}]$$

$$G_2(x) = \gcd\,[g_2(x^a),\ 1 + x^{n_2}]$$

EXAMPLE 3.4.4.1.

Consider the cyclic product code C whose component codes A and B are the (15, 10) and (7,3) cyclic binary codes respectively and each of minimum distance 4. Then $n_1 = 15$, $k_1 = 10$, and

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

$$= 1 + x^2 + x^4 + x^5$$

Also, $n_2 = 7$, $k_2 = 3$, and

$$g_2(x) = (1 + x + x^3)(1 + x)$$
$$= 1 + x^2 + x^3 + x^4 .$$

Consequently, the product code has length $n_1 n_2 = 105$ and has $K_1 K_2 = 30$ in formation symbols.

We have

$$(1)\ 15 + (-2)7 = 1$$

that is $a = 1$ and $b = -2$

Thus,

$$g_1(x^{bn_2}) = g_1(x^{-14}) = 1 + x^{-28} + x^{-56} + x^{-70}$$

$$= x^{-70}\left[1 + x^{14} + x^{42} + x^{70}\right]$$

Also,

$$g_2(x^{an_1}) = g_2(x^{15}) = 1 + x^{30} + x^{45} + x^{60}$$

We apply corollary 3.4.1 to get

$$g(x) = \text{lcm}\left\{\gcd\left[1+x^{30}+x^{45}+x^{60},\ 1+x^{105}\right], \gcd\left[x^{-70}(1+x^{14}+x^{42}+x^{70}),\ 1+x^{105}\right]\right\}$$

$$= \text{lcm}\left\{1 + x^{30} + x^{45} + x^{60},\ 1 + x^7 + x^{21} + x^{35}\right\}$$

$$= \frac{(1 + x^{30} + x^{45} + x^{60})(1 + x^7 + x^{21} + x^{35})}{\gcd\{1 + x^{30} + x^{45} + x^{60},\ 1 + x^7 + x^{21} + x^{35}\}}$$

$$= (1+x+x^2)(1+x^3+x^4)(1+x+x^2+x^3+x^4)$$
$$\cdot (1+x+x^2+x^4+x^6)(1+x^4+x^6+x^7+x^9+x^{10}+x^{12})$$
$$\cdot (1+x+x^2+x^4+x^7+x^8+x^9+x^{10}+x^{12})$$
$$\cdot (1 + x^7 + x^{21} + x^{35}).$$

Since deg g(x) = n-k = 75 then k = n-75 = 30 as it was expected. Finally the product code has minimum distance $d = d_1 d_2 = 16$ and it can correct $t = \left[\frac{d-1}{2} \right] = 7$ errors.

Finally, we note that, if $h_1(x)$ and $h_2(x)$ represent the parity polynomials of the column and row component codes respectively, it has been shown [35] that the parity polynomial h(x) of the product code is given by:

$$h(x) = \gcd \left[H_1(x^{n_2}),\ H_2(x^{n_1}) \right]$$

where

$$H_1(x) = \gcd \left[h_1(x^b),\ 1 + x^{n_1} \right]$$

$$H_2(x) = \gcd \left[h_2(x^a),\ 1 + x^{n_2} \right] \quad .$$

3.4.3. Cascade decoding of direct-product codes.

Since each row and each column of the array in 3.4.0.2. consists of a code word, we may decode each of the column (row) code words separately and then decode each of the row column code words. The complexity of the decoder we use depends upon the way of ordering the coefficients of the product code word into the two-dimensional array. The most obvious way in which to decode a canonically ordered direct-product code is to decode the component codes one at a time, as shown in Fig. 3.1.

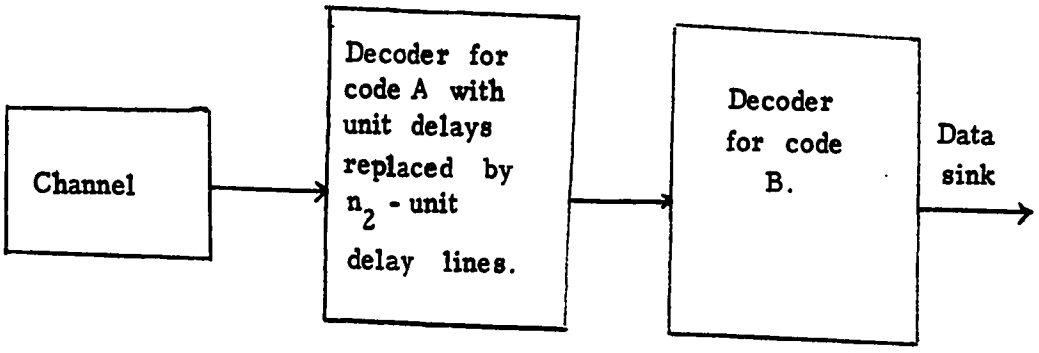


Figure 3.1.

In corollary 3.4.1. we showed that the generator polynomial g(x) of the product code is given by

$$g(X) = \text{lcm}[G_1(X^{n_2}), G_2(X^{n_1})]$$

Hence, as far as cyclic ordering is concerned, the polynomial g(X) will result in the code words from $G_1(X) = \gcd[g_1(X^b), X^{n_1}-1]$ interlaced on every n_1 th digit of the product code, and code words from $G_2(X) = \gcd[g_2(X^a), X^{n_2}-1]$ interlaced on every nth digit of the product code. The general outline of a decoder for a cyclic product code differs slightly from the cascaded decoder shown in Fig. 3.1.

If the code A_1 is generated by $G_1(X)$ and B_1 is generated by $G_2(X)$ a general outline of the cascade decoder for direct-product code transmitted in the cyclic order is shown in Fig. 3.2.

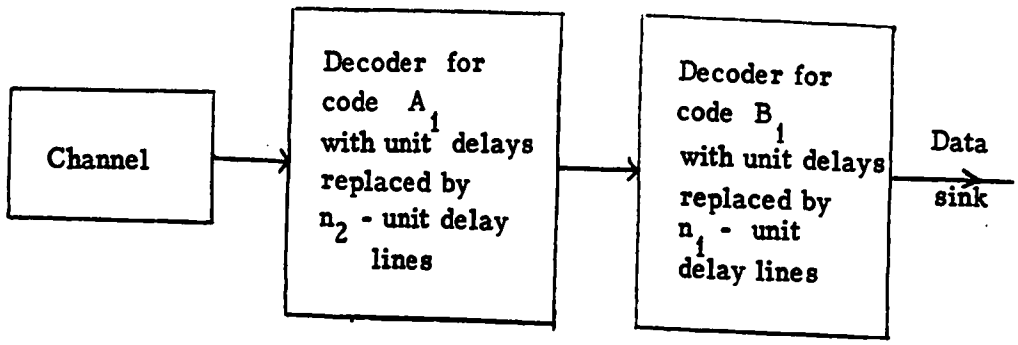


Figure 3.2.

If a pattern of errors should occur during the transmission of a direct-product code word, then, independently of the kind of ordering used, it will be distributed among both the code's rows and the code's columns. Hence, in accordance with the two-dimensional array of a product code word we also refer to an error pattern as a two-dimensional error pattern. Accordingly, we have the following.

DEFINITION 3.4.3.

A two-dimensional pattern of errors within a product code array is called a spot error [33]. If the spot error consists of j random errors it is also referred to as j-error pattern, e.g. the error pattern in Fig. 3.3. is a spot error or a 4-error pattern. It is evident that the weight of a spot error equals the number of single errors contained within it.

3.4.4. Error-correcting properties of cascade decoders.

If the error correcting capability of the product code is t, the cascade algorithm will not decode all error patterns of t or fewer errors. For example if $d_1 = d_2 = 3$, the distance of the product code is d = 9, and the product code is capable of correcting any error pattern of four or fewer errors. However, the error pattern shown in Fig. 3.3. will not be corrected by the cascade decoding.

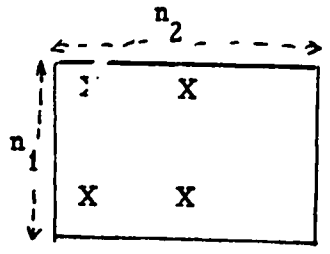


Figure 3.3.

DEFINITION 3.4.4.

An error pattern which cannot be corrected by the cascade decoding procedure is called a permanent error pattern [31] .

e.g. the error pattern shown in Fig. 3.3. is a permanent error pattern.

Although the algorithm will not correct all error patterns of t or fewer errors, it will correct the overwhelming majority of patterns of t errors, and, in addition, will correct many of the patterns of $t + 1, t + 2, \ldots,$ errors.

Therefore, the overall error probability for this decoding procedure can be significantly lower than the error probability for a conventional decoding procedure that corrects all errors up to the guaranteed error-correcting ability of the code and none above this level. In spite that, the fact that there are permanent error patterns is a defect of this decoding procedure paid as a price for the ease of implementation.

Let t_1 and t_2 be the error-correcting capabilities of the column component code and row component code respectively. If we assume the distance of the product code is odd it can theoretically correct any spot error of weight e or less, where

$$e = 2t_1 t_2 + t_1 + t_2 \qquad (d_1, d_2 \text{ odd}) \qquad 3.4.4.1.$$

whereas, if the Hamming distance is even

$$e = 2t_1 t_2 + 2 (t_1 + t_2) + 1 \quad (d_1, d_2 \text{ even}) \qquad 3.4.4.2.$$

Let τ_1 be the largest number of errors contained in the product code word array that will always be corrected by the cascade decoding procedure we have described. One can see by examining the various error patterns, that the maximum number of random errors contained in a correctable spot error cannot be more than $(t_1 + 1)(t_2 + 1) - 1$.

That is, if E is a spot error with

$$|E| \leq \tau_1 \qquad 3.4.4.3.$$

where

$$\tau_1 = t_1 t_2 + t_1 + t_2 \qquad 3.4.4.4.$$

then this error pattern will be corrected with probability one.

3.4.5. Majority-Logic Decoding for Cyclic Codes.

The majority-logic decoding is an effective decoding scheme for certain classes of cyclic codes.

Let $\acute{V} = (V_0, V_1, \ldots, V_{n-1})$ denote the transmitted code vector of the (n, k) code and let $\acute{R} = (r_0, r_1, \ldots, r_{n-1})$ denote the received vector. Their sum

$$\acute{V} + \acute{R} = \acute{E} = (e_0, e_1, \ldots, e_{n-1}) \qquad 3.4.5.1.$$

is the error vector added by the channel.

The above equation yields

$$\begin{aligned} \acute{R}\, H^T &= \acute{E}\, H^T \\ &= \acute{S} \\ &= (S_0, S_1, \ldots, S_{n-k-1}) \end{aligned} \qquad 3.4.5.2.$$

where S is the syndrome corresponding to $\acute{E}$ and H^T is the transpose of the parity matrix H of the code. Each syndrome bit is a sum of certain error digits. If $B = (b_0, b_1, \ldots, b_{n-1})$ is any n-tuple in the null space of the code, then equation 3.4.5.1. yields that the inner product of $\acute{B}$ with R is identical to the inner product of B with $\acute{E}$. Let A represent such an inner product, i.e.,

$$A = b_0 e_0 + b_1 e_1 + \ldots + b_{n-1} e_{n-1} \qquad 3.4.5.3.$$

where b_i is either 0 or 1. This sum is called parity-check sum.

The error digit e_i is said to be checked by A if the coefficient b_i of e_i in A is 1. Suppose now that we form a set of parity-check sums $A_1, A_2, \ldots, A_J$. This set is said to be orthogonal on the error digit e_i if

1) e_i is checked by each check sum of the set, and
2) no other error digit is checked by more than one check sum [21]. For example, the following three check sums are orthogonal on the error digit e_0:

$$A_1 = e_0 + e_2$$

$$A_2 = e_0 + e_1$$

$$A_3 = e_0 + e_3 + e_4$$

Each check sum orthogonal on e_i is of the form

$$A_\ell = \sum_{j \neq i} e_j + e_i, \quad \ell = 1, \ldots, J \qquad 3.4.5.4.$$

If $e_j = 0$ for $j \neq i$ then

$$e_i = A_\ell \qquad 3.4.5.5.$$

Therefore, we can estimate e_i by using the parity-check sums orthogonal on e_i and Eq. 3.4.5.5.. If it is possible to form J parity-check orthogonal on e_i and there are $[J/2]$ or fewer errors in the error vector $E = (e_0, e_1, \ldots, e_{n-1})$, then we can get the estimate of the error digit e_i by majority logic: The error digit e_i is decoded as 1 if a clear majority of the check sums is 1; otherwise, e_i is decoded as 0. In the case of ties, e_i is taken to be 0. Once we can form no more than J parity check sums orthogonal on e_{n-1}, then it is possible to form J parity checks sums orthogonal on any other message error digit $e_{n-2}, e_{n-3}, \ldots, e_{n-k}$. This decoding algorithm is called one-step majority-logic decoding (1 - step MLD), and if this is effective for a code then, it can correct any error pattern of $[J/2]$ or fewer errors. Hence,

for $J = d-1$ the code can correct t or less random errors.

DEFINITION 3.4.5.1.

A cyclic code with minimum distance d is said to be one-step majority-logic decodable (1 - step MLD) or to be completely orthogonalizable in one step if and only if it is possible to form $J = d-1$ parity-check sums orthogonal on every error digit. The (15, 7, 2) code is the only known BCH code which is 1 - step MLD. Shortening 1 bit the $(n, q+1, 2^{q-2}-1)$ BCH code we get the class $(n-1, q, 2^{q-2}-1)$ shortened cyclic codes which are 1 - step MLD. e.g. the shortened cyclic codes (14, 4, 3) and (6, 3, 1) belong to this category.

Let

$$E = \{e_{i_1}, e_{i_2}, \ldots, e_{i_M}\} \qquad 3.4.5.6.$$

be a set of M error digits. The number of error digits in E is called the size of the set E.

DEFINITION 3.4.5.2.

A set of parity check sums is said to be orthogonal on the set E if and only if: (1) every error digit in E is checked by every check sum and (2) no other error digit is checked by more than one check sum [21] .

For example, the three check sums shown below are orthogonal on the check sum $e_0 + e_1$

$$A_1 = e_0 + e_1 + e_2$$
$$A_2 = e_0 + e_1 + e_3$$
$$A_3 = e_0 + e_1 + e_4 + e_5 \ .$$

It is possible to proceed in such a way that we can estimate sums from sums of larger size. This process is called orthogonalization [21]. If, by using this process, we can construct J or more parity-check sums

orthogonal on only a single error digit e_i then, this error digit can be estimated by majority logic and the code can correct any error pattern of $[\ J/2\ ]$ random errors or less.

DEFINITION 3.4.5.3.

A code is said to be L-step MLD or L-step orthogonalizable if L steps of orthogonalization are required to make a decoding decision on an error digit e_i. The code is said to be completely L-step orthogonalizable if J is equal to d-1, where d is the minimum distance of the code [21].

It has been shown [21] that the $(2^q, 2^q-q-1)$ Hamming code is completely orthogonalizable in q-1 steps. The subclass $(2^q-1, q+1, 2^{q-2}-1)$ BCH codes for $q \geq 3$ are 2-step orthogonalizable and the (31, 16, 3) code is 3-step orthogonalizable.

3.4.6. Majority-Logic Decoding of Product Codes.

Using majority logic decoding for product codes we can achieve their error correcting capability guaranteed by their minimum distance.

It has been proved [35] that if the column-component code A is L-step MLD and the row-component code B is 1-step MLD then, their product code C is L-step MLD. Actually, this result is a special case of the following theorem which has been proved by Gore [37]

THEOREM 3.4.6.1.

The product of an L_1-step orthogonalizable code with an L_2-step orthogonalizable code produces an $(L_1 + L_2 - 1)$-step orthogonalizable code. In applying MLD on a product code word of C, the word is treated as one dimensional and not as two dimensional. We will refer to this decoding method as straight majority logic decoding. This method has, of course, a random error correcting capability of $t = \left[\ \frac{d_1 d_2 - 1}{2}\ \right]$ and, because of MLD, can also correct many error patterns not guaranteed as correctable

by the minimum distance of the product code. Thus the price paid for achieving the capability of the product code to correct any error pattern of t or fewer errors is that A and B are MLD. It is known [36] that for achieving the error correcting capability of the product code it is enough if either C_1 or C_2 is one-step MLD. Thus the restriction of ML decodability can be relaxed on one of the two component codes. The decoding algorithm is necessarily more complex than the cascade decoding and/or the straight majority logic decoding. Also, the decoding algorithm cannot correct as many error patterns, not guaranteed as correctable by the minimum distance of the product code, as the previous two methods can.

CHAPTER 4

4.1. CONSTRUCTION OF DIRECT PRODUCT CODES

4.1.1. Construction of Symmetric Direct Product Codes (SDPC)

Let $A = B = V$ be an (n, k) group code. Let us consider the matrix

$$M = \begin{array}{c} \\ R_1 \\ R_2 \\ \vdots \\ R_k \\ \vdots \\ R_n \end{array} \begin{array}{c} \begin{array}{cccccccc} C_1 & C_2 & & & C_k & & & C_n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2k} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} & \cdots & a_{kn} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} & \cdots & a_{nn} \end{bmatrix} \end{array} \qquad 4.1.1.1.$$

which is formed in the following way :

Choose $R_1 = C_1 = V_1 \in V,$

$R_2 = C_2 = V_2 \in V,$

$\vdots$

$R_k = C_k = V_k \in V.$

Now consider C_{k+1}. In this column the first k bits are already fixed. Therefore, there is only one r-tuple, with $r = n-k$, which when appended to the already existing entries will give a word of V. Choose this word V_{k+1} for C_{k+1} and R_{k+1}. Next consider C_{k+2}. Here the first k+1 entries are already fixed. Choose this word V_{k+1} for C_{k+1} and R_{k+1}. Next consider C_{k+2}. Here the first k+1 entries are already fixed. Choose (r-1)-tuple such that this, together with the already existing

entries, gives a word of V. Choose this word V_{k+2} for C_{k+2} and R_{k+2}. Continue the process till the entire M is full. Every column of the matrix M belongs to A and every row belongs to B. Hence, according to the definition 3.4.1. the array 4.1.1.1 is a product codeword arranged in canonical ordering; clearly the matrix M is symmetric. Therefore, either we use row by row or column by column transmission, the product codeword produced is the same. Hereafter a product code with this property will be referred to as a symmetric DP code. The number of product code words produced is a subset of the set of all product codewords. This is so, since in order to form the product codewords we use the codewords of a component code available rather than make use of the conventional procedure, in which the check digits are annexed to the information digits to form each code word. If N_s and N_t are the number of product code words produced and the number of total product code words respectively, then

$$N_s = N_t \, 2^{-\frac{k(k-1)}{2}} \qquad 4.1.1.2.$$

where k is the dimension of each component code.

That this is true can be seen as follows:

The number of code words in each component code A and B is 2^k. For $C_1 = R_1$ we have 2^k choices; for $C_2 = R_2$, 2^{k-1} choices etc. Hence,

$$N_s = 2^k . 2^{k-1} \ldots 2^1$$

Since A = B = (n, k) the dimension of the direct product code is $k_1 k_2 = k^2$ and hence, it has

$$N_t = 2^{k^2}$$

codewords.

Therefore, $$\frac{N_s}{N_t} = \frac{2^{\frac{k(k+1)}{2}}}{2^{k^2}} = 2^{-\frac{k(k-1)}{2}}, \qquad 4.1.1.3.$$

hence, the ratio N_s/N_t decreases exponentially with k.

It is also true that the information rate R_s decreases. Indeed, if R is the information rate corresponding to the case where the matrix M in 4.1.1.1. is not symmetric, we have that

$$\frac{R_s}{R} = \frac{k(k+1)/2n}{k^2/n} = \frac{1}{2} + \frac{1}{2k} \qquad 4.1.1.4.$$

and the information rate in case of symmetry, drops down to one half if the number of information symbols in the component codes is pretty large. This shows that symmetric product codes are low rate codes. It is, therefore, desirable that V has as high a rate as possible.

EXAMPLE 4.1.1.1. Let V be the (5, 3) group code with A = B = V. The generator matrix G for V may be taken to be

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

and the $2^k = 2^3 = 8$ codewords of each component code A and B will be

$V_1 = 1\ 0\ 0\ 1\ 1$
$V_2 = 0\ 1\ 0\ 1\ 0$
$V_3 = 0\ 0\ 1\ 0\ 1$
$V_4 = 1\ 1\ 0\ 0\ 1$
$V_5 = 1\ 0\ 1\ 1\ 0$
$V_6 = 0\ 1\ 1\ 1\ 1$
$V_7 = 1\ 1\ 1\ 0\ 0$
$V_8 = 0\ 0\ 0\ 0\ 0$.

We may start with any 3 vectors to construct the matrix M: e.g., starting with V_1, V_2 and V_3 as the first 3 rows and first 3 columns in matrix M then, the V_4 codeword must be chosen to fill in the 4th row and 4th column of the array; We complete M by choosing V_5 as the 5th row and 5th column. The product code word produced will be the following.

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

If the code vectors V_3, V_6 and V_7 are chosen for the first 3 rows and the first 3 columns then, the code vector V_2 must be chosen for the 4th row and 4th column and the code vector V_4 must be chosen for the last row and last column. Hence, the code word produced will be the following

$$M_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The number of product codewords produced will be 2^6 while the total number of product codewords is 2^9.

EXAMPLE 4.1.1.2. Let A = B = V be the (7,4) code generated by

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

Each of the component codes will consist of the following $2^4 = 16$ code words:

$V_1 = 1\ 1\ 0\ 1\ 0\ 0\ 0$ $\quad$ $V_9 = 0\ 1\ 1\ 1\ 0\ 0\ 1$

$V_2 = 0\ 1\ 1\ 0\ 1\ 0\ 0$ $\quad$ $V_{10} = 0\ 0\ 1\ 0\ 1\ 1\ 1$

$V_3 = 0\ 0\ 1\ 1\ 0\ 1\ 0$ $\quad$ $V_{11} = 1\ 0\ 1\ 0\ 0\ 0\ 1$

$V_4 = 0\ 0\ 0\ 1\ 1\ 0\ 1$ $\quad$ $V_{12} = 1\ 1\ 1\ 1\ 1\ 1\ 1$

$V_5 = 1\ 0\ 1\ 1\ 1\ 0\ 0$ $\quad$ $V_{13} = 0\ 1\ 0\ 0\ 0\ 1\ 1$

$V_6 = 1\ 1\ 1\ 0\ 1\ 1\ 0$ $\quad$ $V_{14} = 1\ 0\ 0\ 0\ 1\ 1\ 0$

$V_7 = 1\ 1\ 0\ 0\ 1\ 0\ 1$ $\quad$ $V_{15} = 1\ 0\ 0\ 1\ 0\ 1\ 1$

$V_8 = 0\ 1\ 0\ 1\ 1\ 1\ 0$ $\quad$ $V_{16} = 0\ 0\ 0\ 0\ 0\ 0\ 0$

If we choose the codewords V_1, V_5, V_{13} and V_7 to fill in the first four rows and first four columns then,

V_8, V_{10} and V_3 must be chosen for the last three rows and last three columns. The product code word M will be the following.

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

If V_1, V_{11}, V_8 and V_5 are chosen to fill in the first, second, third and fourth rows (columns) of the matrix M respectively then, in order to fill in the fifth, sixth and seventh rows (columns), V_3, V_{10} and V_{13} must be chosen resulting in another product codeword.

4.1.2. Construction of product code words arranged in canonical order.

Let A and B be the (n_1,k_1) and (n_2,k_2) cyclic codes generated by the polynomials $g_1(x)$ and $g_2(x)$ respectively, where

$$\deg\ g_1(x) = n_1-k_1 = r_1 \quad \text{and} \quad \deg\ g_2(x) = n_2-k_2 = r_2$$

Suppose

$$g_1(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_{r_1} x^{r_1} = \sum_{\ell=o}^{r_1} p_\ell x^\ell \qquad 4.1.2.1.$$

and

$$g_2(x) = q_0 + q_1 x + q_2 x^2 + \ldots + q_{r_2} x^{r_2} = \sum_{\ell'=o}^{r_2} q_{\ell'} x^{\ell'} \qquad 4.1.2.2.$$

and let $A_j(x)$ and $B_i(x)$ be the polynomial representation of the code-words of A and B respectively. If V(x) is a product codword arranged in canonical ordering then

$$V(x) = n_1 \updownarrow \overset{\begin{matrix} A_o(x) & A_1(x). & . & . & A_{n_2-1}(x) \end{matrix}}{\begin{bmatrix} x & x & . & . & . & x \\ x & x & . & . & . & x \\ . & & & & & \\ . & . & . & . & . & . \\ . & & & & & \\ x & x & . & . & . & x \end{bmatrix}} \begin{matrix} B_0(x) \\ B_1(x) \\ . \\ . \\ . \\ B_{n_1-1}(x) \end{matrix} \qquad 4.1.2.3.$$

$$\longleftarrow n_2 \longrightarrow$$

and, since each column belongs to the code A and each row belongs to the code B, V(x) can be expressed either column-wise as

$$V(x) = A_o(x^{n_2}) + xA_1(x^{n_2}) + x^2 A_2(x^{n_2}) + \ldots + x^{n_2-1} A_{n_2-1}(x^{n_2}) \qquad 4.1.2.4.$$

or row-wise as

$$V(x) = B_o(x) + x^{n_2} B_1(x) + x^{2n_2} B_1(x) + x^{2n_2} B_2(x) + \ldots + x^{(n_1-1)n_2} B_{n_1-1}(x)$$

4.1.2.5.

If $I_j(x)$ and $I_i(x)$ are the corresponding information polynomials of the codewords $A_j(x)$ and $B_i(x)$ respectively, then according to formula 2.5.1. we can write

$$A_j(x) = g_1(x)\, I_j(x), \quad j = 0, 1, \ldots, n_2-1$$

and 4.1.2.6.

$$B_i(x) = g_2(x)\, I_i(x), \quad j = 0, 1, \ldots, n_1-1$$

where $\deg I_j(x) \leq k_1-1$ and $\deg I_i(x) \leq k_2-1$.

Therefore,

$$A_j(x^{n_2}) = g_1(x^{n_2})\, I_j(x^{n_2}), \quad j = 0, 1, \ldots, n_2-1 \qquad 4.1.2.7.$$

and substituting 4.1.2.7. and the second of 4.1.2.6. into 4.1.2.4. and 4.1.2.5, they yield respectively

$$V(x) = g_1(x^{n_2}) I_o(x^{n_2}) + x g_1(x^{n_2}) I_1(x^{n_2}) + \ldots + x^{n_2-1} g_1(x^{n_2}) I_{n_2-1}(x^{n_2})$$

$$= g_1(x^{n_2}) \sum_{j=o}^{n_2-1} x^j I_j(x^{n_2}) \qquad 4.1.2.8.$$

and

$$V(x) = g_2(x)\, I_o(x) + x^{n_2} g_2(x)\, I_1(x) + \ldots + x^{(n_1-1)n_2} I_{n_1-1}(x)$$

$$= g_2(x) \sum_{i=o}^{n_1-1} x^{in_2} I_i(x) \qquad 4.1.2.9.$$

Since the left-hand sides of the equations 4.1.2.8.and 4.1.2.9. represent the same product codewords we have

$$g_1(x^{n_2}) \sum_{j=o}^{n_2-1} x^j I_j(x^{n_2}) = g_2(x) \sum_{i=o}^{n_1-1} x^{in_2} I_i(x) \qquad 4.1.2.10.$$

Let

$$I_j(x) = a_{j,0} + a_{j,1}x + a_{j,2}x^2 + \ldots + a_{j,k_1-1}x^{k_1-1}$$

$$= \sum_{m=o}^{k_1-1} a_{j,m} x^m \qquad 4.1.2.11.$$

and

$$I_i(x) = b_{i,0} + b_{i,1}x + b_{i,2}x^2 + \ldots + b_{i,k_2-1}x^{k_2-1}$$

$$= \sum_{m'=o}^{k_2-1} b_{i,m'} x^{m'} \qquad 4.1.2.12.$$

From 4.1.2.1. and 4.1.2.11. we get

$$g_1(x^{n_2}) = \sum_{\ell=o}^{r_1} p_\ell x^{\ell n_2} \qquad 4.1.2.13.$$

and

$$I_j(x^{n_2}) = \sum_{m=o}^{k_1-1} a_{j,m} x^{mn_2} \qquad 4.1.2.14.$$

Substituting 4.1.2.2., 4.1.2.12., 4.1.2.13. and 4.1.2.14. into 4.1.2.10. yields

$$\sum_{j=o}^{n_2-1} \sum_{\ell=o}^{r_1} \sum_{m=o}^{k_1-1} p_\ell a_{j,m} x^{(\ell+m)n_2+j} = \sum_{i=o}^{n_1-1} \sum_{\ell'=o}^{r_2} \sum_{m'=o}^{k_2-1} q_{\ell'} b_{i,m'} x^{in_2+\ell'+m'}$$

with ℓ, m, ℓ' and m' $\geq$ 0 4.1.2.15.

Equation 4.1.2.10 or its equivalent 4.1.2.15 is the key equation we can use to find an appropriate set of codewords from code B such that their product with codewords from code A will give us a direct-product codeword A x B, provided that the product codeword is arranged in canonical order.

Indeed, each side of equation 4.1.2.15. is a polynomial of degree n_1n_2, and since these polynomials are identical we may equalize the corresponding coefficients of the same degree terms to obtain n_1n_2 equations. Each of these equalities will represent a relationship among some of p'_ℓ s, $a'_{j,m}$ s $q'_{\ell'}$ s and $b'_{i,m'}$ s depending on the degree of the corresponding term. The n_1n_2 equalities may be partitioned into n_1 sets each of which will contain n_2 equalities. From the first k_2 equalities of each of the first k_1 sets we can calculate the digits $b_{im'}$, $i = 0,1,\ldots,k_1-1$, $m' = 0,1,\ldots,k_2-1$ because the p'_ℓs, $\ell = 0,1,\ldots,r_1$ and $q'_{\ell'}$ s, $\ell' = 0,1,\ldots, r_2$ are known and the $a_{j,m}$, $j = m'$, $m = i$ will be predetermined and, hence, k_1k_2 digits have to be chosen arbitrarily. The remaining digits $b_{im'}$, $i = k_1,\ldots, n_1$, $m' = 0,1,\ldots, k_2-1$ may be found from the first k_2 equalities of each of the remaining r_1 sets.

EXAMPLE 4.1.2.1. Let $A = (7,4)$ and $B = (3,1)$ be the cyclic codes with generator polynomials $g_1(x) = 1+x+x^3$ and $g_2(x) = 1+x+x^2$ respectively. Applying equation 4.1.2.15. for $k_1 = 4$, $r_1 = 3$, $k_2 = 1$, $r_2 = 2$, $n_1 = 7$ and $n_2 = 3$ we get 21 equalities. Choosing $(a_{00}, a_{01}, a_{02}, a_{03})$ to be, say, (1, 0, 0, 1) we obtain that the information polynomials with respect to B ought to be

$$I_0(x) = I_1(x) = I_4(x) = I_6(x) = 1$$

and $I_2(x) = I_3(x) = I_5(x) = 0$

Hence,

$$B_0(x) = B_1(x) = B_4(x) = B_6(x) = 1 + x + x^2$$

and $B_2(x) = B_3(x) = B_5(x) = 0$

and, therefore, the product codeword will be

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

4.2. DECODING OF BINARY DIRECT PRODUCT CODES

4.2.1. Extension of Cascade Decoding.

If the received codeword R(x) is given and we proceed in such a way that the error pattern E(x) may be specified then the transmitted codeword V(x) may be found by observing that

$$V(x) = R(x) + E(x) \qquad 4.2.1.1.$$

Given an R(x) there is a unique V(x) and there is a unique E(x) which caused this R(x) if the E(x) is correctable, i.e., if

$$|E(x)| \leq t \qquad 4.2.1.2.$$

and

$$|V(x)| \geq d \qquad 4.2.1.3$$

with

$$d = 2t + 1 \qquad 4.2.1.4.$$

the minimum distance of the code C, $V(x) \in C$.
Therefore, if V(x) belongs to a group code V with even minimum distance d and E(x) has weight $\frac{d}{2}$ or less, for a given R(x) there is at most only one E'(x) of weight $\frac{d}{2} - 1$ or less such that R(x)+E'(x) belongs to V; E(x) = E'(x). If no such E'(x) exists, then E(x) has weight $\frac{d}{2}$. This comment provides us with a test for finding whether R(x) has a correctable, or merely detectable, error.

Based on the above discussion it is possible to modify the cascade decoding procedure in such a way, that the error correcting capability of the product code increases, if each of its component codes has even minimum distance. The received product codeword has to be passed through the system of Fig. 3.1 (or Fig. 3.2) twice rather than once while a certain condition has to be stipulated at the end of second pass in the case where the spot error still has not been corrected. If we start decoding the product array columnwise the aboe extension guarantees the correction of any spot error whose weight is not more than

$$\tau' = t_1 t_2 + 2t_1 + t_2 + 1 \qquad 4.2.1.5.$$

so that, if 3.4.4.4. is taken into account, we get

$$\Delta\tau = \tau' - \tau = t_1 + 1 \qquad 4.2.1.6$$

and, hence, we succeed in correcting t_1+1 more random errors. Table 4.1 summarizes the results derived in the case where the component codes are the same. Many other spot errors of weight much greater than the one guaranteed by table 4.1. will be corrected For example, if A = B = (15,7,2) with $d_1 = d_2 = 6$, then the extension of the cascade decoding guarantees that spot errors of weight up to 30 or less, 45 or less, 48 or less, 56 or less and 57 or less may be corrected; if A = B = (15, 5,3) with $d_1 = d_2 = 8$, then spot errors of weight 45 or less, 55 or less, 60 or less, 71 or less, 81 or less and 82 or less may be corrected while the error correcting capability of the product code is 31.

Table 4.1.

Minimum distance of the component codes $d_1 = d_2$	Highest weight of the spot errors always correctable by cascade decoding τ	Highest weight of spot errors always correctable by extending the cascade decoding τ'	Guaranteed spot error correcting capability of the product code t
2	0	1	1
4	3	5	7
6	8	11	17
8	15	19	31
10	24	29	49
12	35	41	71

Accordingly, it has been proved that the overall error probability for a spot error to be permanent decreases. Results derived for the case of $n_1 = n_2 = 100$ and $d_1 = d_2 = 4$ are included in Table 4.2 compared to the results taken by Abramson [31].

Table 4.2

Weight of the spot error j	Probability that a j-error pattern will not be correctable for $n_1 = n_2 = 100$ and $d_1 = d_2 = 4$	
	Cascade Decoding Procedure	Extending the Cascade Decoding Procedure.
1	0	0
2	0	0
3	0	0
4	3×10^{-4}	0
5	1.5×10^{-3}	0
6	4.5×10^{-3}	15.1×10^{-6}
7	1.1×10^{-2}	4.1×10^{-9}

It has to be noted that these results correspond to the most pessimistic conditions of the single error distribution into the product array.

In conclusion, it seems clear that extending the cascade decoding procedure we can increase the error correcting capability of the code.

4.2.2. DECODING OF SMLD DP CODES.

In section 4.1.1. we show that, in the case of symmetric DP code, the matrix M is symmetric that is, the jth column and the jth row are the same word, say, $V_j \in V$; let us designate them as the jth pair.

Let V_{j1} be the estimate of V_j, obtained by applying majority logic decoding on the jth column. Similarly let V_{j2} be the estimate of V_j, obtained by applying MLD on the jth row. Let m_{pq} represent the pqth entry in M. Suppose we treat m_{pq} as belonging to the pth row for $p < q$ and as belonging to the qth column for $p > q$. Then we can have d orthogonal estimates on m_{pq} treating m_{pq} as belonging to pth row or qth column as the case may be and we can have d orthogonal estimates on m_{pq} as belonging to qth row or pth column, where d stays for the minimum distance of each component code. Since $m_{pq} = m_{qp}$ and the two sets of estimates, taken together, are again othogonal, we can have 2d-1 orthogonal estimates on $m_{pq} = m_{qp}$. On the basis of these estimates, we can have a third estimate V_{j3} of V_j. Thus, altogether we can have three estimates of V_j.

Now let us consider the following decoding procedure:

Step 1. For the jth pair we get the three estimates V_{j1}, V_{j2}, V_{j3}. If at least two of them agree, then we take this estimate to be true and make the corrections accordingly in the jth column and the jth row. If not, we leave the jth column and the jth row as they are. We repeat the whole process for $j = 1, 2, \ldots, n$.

Step 2. Repeat Step 1 and stop.

4.2.3. Random Error Correction.

For the decoding of the jth pair, in step 1, to end in incorrect result or in indecision, there must be at least d errors in the jth pair. In view of this, if the total number of errors is $\tau_2 = \frac{d(d-1)}{2}$ or less, then at the end of step 1, there are at most $\frac{d-1}{2}$ pairs which are not error free. Moreover, the number of errors per pair is d-1 or less. Therefore, in step 2 all these errors are corrected. It is easy to construct error patterns of weight $\tau_2 + 1$ which are not correctable. Thus the decoding procedure can correct all error patterns of weight τ_2 or less, where

$$\tau_2 = 2t^2 + t \qquad 4.2.3.1.$$

and $t = t_1 = t_2 = \left[\frac{d-1}{2}\right]$

If we apply cascade decoding for a symmetric DP code, equation 3.4.4.4. yeilds that any error pattern of weight τ_1 or less with

$$\tau_1 = t^2 + 2t \qquad 4.2.3.2.$$

will be corrected. If $t > 1$ then, clearly,

$$\tau_1 < \tau_2 \qquad 4.2.3.3.$$

and, hence, the presented decoding procedure improves the error correcting capability of the product code in regards with cascade decoding.

On the other hand, if e_s is the maximum number of random errors a SDPC can correct, equation 3.4.4.1. yields

$$e_s = 2t^2 + 2t \qquad 4.2.3.4.$$

This guaranteed error correcting capability of the SDPC is achievable by applying straight majority logic decoding. Comparing 4.2.3.1. and 4.2.3.4. we get

$$\tau_2 < e_s \qquad 4.2.3.5.$$

Hence, as far as the error correcting capability is concerned the decoding procedure presented is a compromise between cascade decoding and straight majority logic decoding.

4.2.4. Spot Error Correction.

From the discussion in the previous section it is clear that what is important is that at the end of step 1 the number of pairs with errors should be $\frac{d-1}{2}$ or less and each such pair should have d-1 errors or less. In this connection let T be the set of all error patterns which are such that the errors are confined to any $\frac{d-1}{2}$ pairs or less. The decoding procedure can correct any pattern belonging to T.

If E(x) is a spot error with $E(x) \in T$ then,

$$|E(x)| \leq (d-1)\left[n - \frac{d-1}{4}\right] \qquad 4.2.4.1.$$

For example, if V is (14,4,3), this bound works out to be 75.

By way of illustration let V be the (14,4,3) 1-step MLD code. With reference to M let the error pattern of weight $\tau_2 = 21$ be as shown in Figure 4.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	⊗	⊗	⊗	⊗	x	x	x	x	x	x	x	x	x	x
2	⊗	⊗	⊗	⊗	⊗	x	x	x	x	x	x	x	x	x
3	⊗	⊗	⊗	⊗	⊗	⊗	x	x	x	x	x	x	x	x
4	⊗	⊗	⊗	x	x	x	x	x	x	x	x	x	x	x
5	x	⊗	⊗	x	x	x	x	x	x	x	x	x	x	x
6	x	x	⊗	x	x	x	x	x	x	x	x	x	x	x
7	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8	x	x	x	x	x	x	x	x	x	x	x	x	x	x
9	x	x	x	x	x	x	x	x	x	x	x	x	x	x
10	x	x	x	x	x	x	x	x	x	x	x	x	x	x
11	x	x	x	x	x	x	x	x	x	x	x	x	x	x
12	x	x	x	x	x	x	x	x	x	x	x	x	x	x
13	x	x	x	x	x	x	x	x	x	x	x	x	x	x
14	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.1.

Decoding the first pair gives $\tilde{v}_{11} = v_{12}$. So we accept this estimate and make the corrections in the first column and first row. In the process we may introduce errors all along the first column and first row. A similar thing will happen in the second and third pairs. However, in spite of these errors, in pairs $4,5,\ldots,14$, the number of errors per pair does not exceed $d-1 = 6$. Therefore, three pairs

will be decoded correctly. At the end of step 1, we will be left with errors in positions (p,q), $p = 1,2,3$ and $q = 1,2,3$. During step 2 these errors will be corrected since in each pair the number of errors is less than $d-1 = 6$. A valid point that has to be discussed at this point is the advantage of going through this decoding procedure especially when straight majority logic decoding is not only simple to implement but also has an error correcting capability up to e_s.

To answer this point let us consider the partitioning of M as shown in Fig. 4.2. The partitioning ends in a point. We note that $A_{j2}\,A_{j1}$ is a part of jth column and $A_{j2}\,A_{j3}$ is a part of jth row.

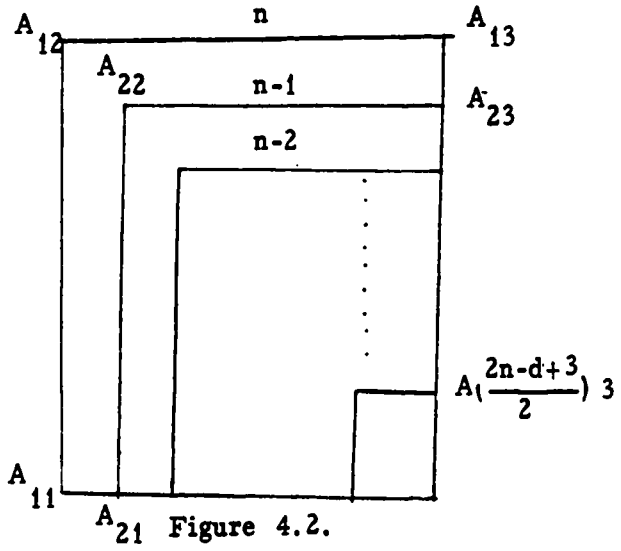


Figure 4.2.

With reference to the first step of the decoding procedure, we see that if the weight of the error pattern along $A_{11}\,A_{12}\,A_{13}$ is d-1 or less the first pair will be decoded correctly; that is at the end of decoding of the first pair, $A_{11}\,A_{12}\,A_{13}$ is error free. This means that, for correct decoding of the second pair, we can allow maximally d-1 errors along $A_{21}\,A_{22}\,A_{23}$, and at the end of decoding of second pair, $A_{21}\,A_{22}\,A_{23}$ is error free. Applying this argument successively over all pairs, we see that the first step of the decoding procedure gives

correct estimates provided the weight of the error pattern per path $A_{j1} A_{j2} A_{j3}$ is d-1 or less. To summarize, the decoding procedure can correct any error pattern which is such that the weight of errors per path $A_{j1} A_{j2} A_{j3}$ is d-1 or less. The weight W of such a pattern can be maximally

$$W = (d-1)\left(n - \frac{d-1}{2}\right) + (d-2) + (d-4) + \ldots + 1$$

that is,

$$W = (d-1)\left(n - \frac{d-1}{4}\right) \qquad 4.2.4.1.$$

$$= t\,(2n - t)$$

and the number of such error patterns with this weight is given by

$$\binom{2n-1}{d-1} \quad \binom{2n-3}{d-1} \cdots \binom{d}{d-1} \qquad 4.2.4.2.$$

Let S be the set of all error patterns which are such that the number of errors in each $A_{j1} A_{j2} A_{j3}$ is d-1 or less. With reference to M this means that heavy weight error patterns appear to be in the bottom right corner of M and scattered elsewhere. Recalling that the pairs are decoded in the order of $j = 1, 2, \ldots, n-1$, we see that we are decoding starting from the area of scattered errors and moving gradually towards the area of clustered errors. This means that if we know, a priori, in which part of M the errors tend to be clustered on the average, then we can choose the order of decoding of pairs accordingly. For example, if the clustering occurs at the top left corner, we choose $j = n, n-1, \ldots, 1$. If the clustering occurs in the middle, we choose $j = 1, n, 2, n-1, 3, n-2$ and so on. On the other hand if the clustering occurs, for instance, in the top right corner, there is no best way of choosing the order of j. It may be pointed out that this order has no bearing on the random error correcting capability of the decoding algorithm.

CHAPTER 5

CONCLUDING REMARKS

In this thesis we have discussed a decoding algorithm for symmetric majority logic decodable binary direct product codes. These codes are necessarily low rate codes. If the component code V has rate k/n, the direct product code has rate $\frac{k^2}{n^2}$ and the symmetric direct product code has a rate slightly greater than $\frac{1}{2}\ \frac{k^2}{n^2}$. The algorithm is simple to implement since it is basically majority logic decoding. It has a random error correcting capability of $\tau_2 = 2t^2 + t$ which is much larger than the capability $\tau_1 = t^2 + 2t$ of cascade decoding and which is not so bad compared to the capability $e_s = 2t^2 + 2t$ of straight majority logic decoding, t being the capability of V. Specifically $\tau_2 - \tau_1 = t(t-1)$ whereas $e_s - \tau_2 = t$.

Two sets S and T of error patterns are correctable by the decoding algorithm. These are considerably large sets and the error patterns belonging to them can be of weight (2n-t)t or less. Since the decoding algorithm is basically majority logic decoding, it can correct also a large number of error patterns of weight greater than τ_2 and not belonging to S or T. It seems that the decoding algorithm has a large overall error correcting capability as a compensation for the low rate of symmetric codes.

It is also shown how we can construct such codes and that they have low information rate. A general procedure concerning the construction of product code words under certain restrictions is also developed. It seems that by applying the cascade decoding procedure for direct product code whose each of the component codes has even minimum distance, its error correcting capability increases.

REFERENCES

[1] C.E. Shannon and W. Weaver, "The Mathematical Theory of Communication", Urbana, Ill. The University of Illinois Press, 1949.

[2] B. Mandebrot, "Electromagnetic Turbulence in Communication Systems", Int'l Conf. on Microwaves, Circuit Theory and Information Theory, Tokyo, Japan, September 1964.

[3] W.W. Peterson, Error-Correcting Codes, Cambridge, Mass: M.I.T. Press, 1961.

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

[5] R.W. Lucky, J. Salz and E.J. Weldon, Principles of Data Communication, New York: McGraw-Hill, 1968.

[6] S. Lin, An Introduction to Error Correcting Codes, New Jersey: Prentice-Hall, 1970.

[7] R.C. Bose and D.K. Ray-Chandhuri, " On a class of error correcting binary codes," Inform. Control, Vol. 3, pp. 68-79, 1960.

[8] E.J. Weldon, Jr., "Euclidean geometry cyclic codes," Elect. Eng. Dept., University of Hawaii, Rept., January 1967.

[10] P. Fire, " A class of multiple-error-correcting binary codes for nonindependent errors," Sylvania Reconnaissance Systems Lab., Mountain View, Calif., Sylvania Rept. RSL-E-2, 1959.

[11] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," J.Soc. Indust. Appl. Math., Vol. 8, pp. 300-304, 1960.

[12] M. Kasahara and V. Kasahara, "Pseudo-Cyclic Codes for Double Burst Error Correction," Electronics and Communications in Japan Vol. 50, No. 1, pp. 9-18, July 1967.

[13] F. Corr, "Multiple burst detection," IRE Proc. Vol. 49, p. 1337, 1961.

[14] J.J. Stone, "Multiple burst error correction," Information and Control, Vol. 4, pp. 324-331, 1961.

[15] - " - "Multiple burst error correction with the Chinese Remainder Theorem," J. SIAM, Vol. 11, pp. 74-81, 1963.

[16] S.E. Tavares and S.G.S. Shiva, "Detecting and correcting Multiple bursts for binary cyclic codes", IEEE Trans. Information Theory, Vol. IT-16, pp. 643-644, September 1970.

[17] S.G.S. Shiva and C.L. Sheng, "Multiple solid burst error correcting binary codes," IEEE Trans. Information Theory, Vol. It-15, pp. 188-189, January 1969.

[18] E.R. Berlekamp, "Practical BCH Decoders", unpublished preliminary version of chaps. 7 and 10 of "Algebraic Coding Theory."

[19] W.W. Peterson, "Encoding and error-correction procedures for the Bose-Chandhuri codes" IRE Trans. Information Theory,Vol. IT-6, pp. 459-470, September 1960.

[20] J.L. Massey, "Step-b7-step decoding of the Bose-Chandhuri-Hocquenghore codes," IEEE Trans. Information Theory, Vol. IT-11, pp. 580-585, October 1965.

[21] -"- Threshold Decoding. Cambridge, Mass: M.I.T. Press, 1963.

[22] G.D.Forney, Jr. "Concatenated codes", The M.I.T. Press, Cambridge, Mass. 1966.

[23] P. Elias, "Error free coding, "IRE Trans. Information Theory, Vol. PG-IT-4, pp. 30-37, September 1954.

[24] D. Slepian, "Some further theory of group codes," Bell Sys. Tech. J., Vol. 39, Sep. 1960, pp. 1219-1252.

[25] H.O. Burton and E.J. Weldon, "Cyclic product codes", IEEE Trans. Inf. Theory, Vol. PG IT-11, pp. 433-439, July 1965.

[26] N. Abramson, "Encoding and decoding cyclic code Groups", Dept. of Elec. Eng., University of Hawaii, Honolulu, Tech. Rept., February 1967.

[27] J.K.Wolf, "On codes derivable from the tensor product of check Matrices," IEEE Trans. on Inform. Theory, IT-11, No. 2, pp. 281-284, April 1965.

[28] R.T. Chien and D.T. Tang, "Cyclic Product codes and their Implementation," Inf. and Control, Vol. 9, pp. 196-209, 1966.

[29] J. Goethals, "Factorization of cyclic codes," IEEE Trans. on Inform. Theory, Vol. 13, pp. 242-246, April 1967.

[30] E.F. Assmus and H.F. Mattson, "Some cyclic codes of block length 3P", in summary scientific report, sec. 3, Sylvania Electronic Systems, Applied Research Lab., Needham, Mass., Contract AF19 (604) 8516, July 1963.

[31] N. Abramson, "Cascade decoding of cyclic product codes" IEEE Trans. Communication Technology, Vol. CIN-16, pp. 398-402, June 1968.

[32] B. Elspas, "Design and Instrumentation of error-correcting codes", Stanford Research Institute, Menlo Park, Calif. 1961.

[33] J.D. Bridwell and J.K. Wolf, "Burst distance and multiple-burst corrections" the Bell System Tech. Journal, pp. 889-909, May - June 1970.

[34] L.R. Bahl and R.T. Chien, "Multiple-Burst-Error Correction by Threshold Decoding", Inf. and Control, Vol. 15, pp. 397-406, 1969.

[35] S. Lin and E.J. Weldon, "Further results on cyclic product codes", IEEE Trans. Information Theory, Vol. IT-16, pp. 452-459, July 1970.

[36] S.M. Reddy, "On decoding iterated codes," IEEE Trans. Inf. Theory, Vol. IT-16, September 1970.

[37] W.C. Gore, "Further results on product codes," IEEE Trans. Inf. Theory, Vol. IT-16, pp. 446-451, July 1970.

[38] R.W. Hamming, "Error detecting and correcting codes," Bell System Tech. J., Vol. 29, pp. 147-160, 1950.

[39] I.N. Herstein, "Topics in Algebra," Blaisdell Publ. Co., New York, 1964.

[40] B.L. Van der Waerden, "Modern Algebra," Frederick Ungar Publishing Co. New York, 1931.

[41] A.A. Albert, "Fundamental Concepts of Higher Algebra," University of Chicago Press, Chicago, Ill., 1956.

[42] J.K. Wolf and B. Elspas, "Error-locating codes - A new concept in error control," IEEE Trans. Inf. Theory, Vol. IT-9, pp. 113-117, April 1963.

[43] J.K. Wolf, "On an extended class of Error locating codes," Inf. and Control, Vol. 8, pp 163-169, 1965.

VITA

NAME: George P. Papoulias

DATE OF BIRTH: August 21, 1940

PLACE OF BIRTH: Astros Arkadias, Greece

EDUCATION:

Secondary: Astros High School,
Astros Arkadias, Greece.

University: National University of Athens,
Athens, Greece.
School of Physics and Mathematics.

Degree : B. Sc in Physics, 1966.