EASILY DECODED ERROR CORRECTING CODES

by

R. J. G. SMITH

A Dissertation submitted for the degree of Doctor of Philosophy in the Faculty of Natural Sciences at the University of Kent at Canterbury

Electronics Laboratories August 1978

To Alison.

CONTENTS

ABSTRACT

This thesis is concerned with the decoding aspect of linear block error-correcting codes. When, as in most practical situations, the decoder cost is limited an optimum code may be inferior in performance to a longer sub-optimum code of the same rate. This consideration is a central theme of the thesis.

The best methods available for decoding short optimum codes and long B.C.H. codes are discussed, in some cases new decoding algorithms for the codes are introduced.

Hashim's "Nested" codes are then analysed. The method of nesting codes which was given by Hashim is shown to be optimum - but it is seen that the codes are less easily decoded than was previously thought.

"Conjoined" codes are introduced. It is shown how two codes with identical numbers of information bits may be "conjoined" to give a code with length and minimum distance equal to the sum of the respective parameters of the constituent codes but with the same number of information bits. A very simple decoding algorithm is given for the codes whereby each constituent codeword is decoded and then a decision is made as to the correct decoding. A technique is given for adding more codewords to conjoined codes without unduly increasing the decoder complexity.

Lastly, "Array" codes are described. They are formed by making parity checks over carefully chosen patterns of information bits arranged in a two-dimensional array. Various methods are given for choosing suitable patterns. Some of the resulting codes are self-orthogonal and certain of these have parameters close to the optimum for such codes. A method is given for adding more codewords to array codes, derived from a process of augmentation known for product codes.

ACKNOWLEDGEMENTS

Thanks to Dr. P.G. Farrell for supervising my research.

Thanks to my wife, family, and friends for their encouragement.

Thanks to Mrs. Sue Smith for her typing of this thesis, and to Motorola for their understanding and their help during its composition.

Thanks to Professor R.C. Jennison and the S.R.C. for facilities and financial support for my work.

ABBREVIATIONS

α	Primitive element of a Galois Field.
A.R.Q.	Automatic Repeat Request.
B.C.H.	Bose-Chaudhuri-Hocquenghem.
C.	Channel capacity, or a Code.
C(G).	Code described by generator matrix G .
$^{n}C_{r} = \frac{n}{r} = \frac{n!}{(n-r)!\ r!}$	
d.	Hamming distance.
d_{min}	Minimum Hamming distance.
F.E.C.	Forward error correction.
F.S.R.	Feedback shift register.
G	Generator matrix.
g(x)	Generator polynomial.
GF(q)	Galois Field with q elements.
H	Parity check matrix.
h(x)	Parity check polynomial.
I/P	Input.
k	Number of information digits in a block code.
n	Block length of a code.
(n,k)	Block code with parameters n, and k.
(n,k,d)	Block code with parameters n, k, and d.
O/P	Output.
P_e	Probability of a random error occuring..
P.R.O.M.	Programmable read-only memory.
R.A.M.	Random access memory.

$r(x)$	Polynomial representation of a received n-tuple.
S	Syndrome.
t	number of random errors a code can correct per block.
$\|X\|$	Determinant X
$\lceil X \rceil$	Largest integer $\leq X$.
$[X]$	Matrix X
$[X]^T$	Transpose of X .

INTRODUCTION

Shannon (1948) provided a mathematical basis for the theory of communication. He showed how a sequence of n symbols, each chosen from a set of q symbols, may contain a certain maximum amount of information. If only a subset of the possible sequences of n symbols are allowed then less information is containable by that subset. Each sequence contains less than the maximum possible amount of information, and is said to contain "redundancy". If the subset of sequences is carefully chosen, it is possible to recognise the sequences even when some of the symbols have been corrupted.

In the practical situation of transmitting information over a noisy channel, it is therefore possible to map sequences of information symbols onto longer sequences which contain redundancy and then to recognise these sequences after they have been corrupted by the channel, subsequently recovering the original information sequences. The ratio of the length of the information sequences to the length of the longer sequences is termed the "rate" of the code, or the "transmission rate". The set of longer sequences is termed a "code", and each member of the set is a "codeword". The number of symbols in a codeword is its "length", and is also the "length" of the code.

Shannon (1948) showed that for transmission rates less than a figure C, (the "channel capacity") it is possible to recognise corrupted codewords (or to correctly "decode" corrupted codewords) with an arbitrarily high certainty, provided that the code is

sufficiently long and is well chosen. He did not give any method for chosing the code.

Coding theory is concerned with methods of chosing codes with desirable characteristics. In practical situations it is not always necessary to find codes fulfilling Shannon's "promise" of negligible error rates at channel capacity; what is required are codes capable of giving desired maximum error rates at specified transmission rates, with economical decoding algorithms. It is unfortunate that the codes with economical, and so simple, decoding methods tend to be of lower rate than other codes giving the same output error rate.

The two main goals of coding theory are to find codes with as high a rate as possible for a given length and error correcting power; and to find simple decoding algorithms for codes or codes with simple decoding algorithms.

This thesis is concerned with codes which are easily decoded. Consideration is confined to the "block" codes, and in chapter one a background is given of the basic properties of those codes, together with the terms used in their description and evaluation. At the end of chapter one the importance of any trade-off in codelength and ease of decoding for a given transmission rate and output error rate is discussed.

In chapter two the area of interest is further narrowed to include only linear codes, and their properties are reviewed. At this stage consideration is narrowed again to include only binary codes, and important classes of such codes are reviewed, together with known decoding techniques for short codes of this kind. Chapter three draws upon these techniques, and adds others,

when the best available methods for decoding short codes with moderate error correcting power are discussed. Decoding methods for selected longer codes are then briefly discussed in chapter three, then chapter four is devoted to a consideration of decoding methods for one of the most important known classes of codes, the long BCH codes (Bose (1959) Chaudhein Hocquengen (1960)). Particular attention is paid to double error correcting BCH codes.

Chapter five discusses the importance of a class of codes which claim to be easily decodable for high rates, lengths, and error correcting powers - the nested codes discovered by Hashim (1974).

There is certainly a need for codes which, although not the best in terms of rate for a given length and error correcting power, are asily decoded. This requirement, discussed in chapter one, is met by the codes described in chapters six and seven.

Chapter six describes conjoined codes, which are codes formed by a combining operation on two or more other codes. A simple method of decoding these codes is given, and their relation to the Reed-Muller (1954) codes is demonstrated.

Chapter seven describes a class of codes with higher rates and a more interesting structure - the array codes. The codes are based on a two dimensional arrangement of the information symbols - redundancy being added on the basis of selected patterns over the two dimensional array. Various types of construction are given, with varying trade-offs between decoding complexity and transmission rate for a given length.

At the end of the thesis is a summary of the work presented, and a discussion of possible topics for further research based upon this work.

CHAPTER 1

Basic Properties of Error correcting block codes

1.1. The Coding Theorem

Shannon (1948) has shown that for any transmission rate less than a figure C, the "capacity" of the channel, it is possible to transmit data with a probability of error at the receiver which tends to zero. This rate, C, depends upon the noise on the channel.

The derivation of Shannon's discovery does not give any clue as to how the channels capacity may be fully used in practice. One solution might be to transmit data as extremely long blocks of digits, and then to use a correlation process at the receiver. This method has severe problems in realisation, however, in terms of the storage required at the transmitter and receiver, and in terms of the delay inevitable at the decoder.

The more promising approach is to add redundant digits to information digits, in order that these redundant digits may help the decoder at the receiver to recognise which information digits were transmitted. This concept is the basis of coding theory. In the case of block codes, information digits are divided into blocks of k digits and then redundancy digits are added to these k digits. The redundant digits are a function of the information digits only. In the case of non-block or convolutional codes the redundancy digits still follow information digits, but this time are not dependant on any one block of information digits.
Much work has been done (e.g. Viterbi (1967), Wozencraft (1968) and Forney (1969)), on convolutional codes, but they are not considered in this thesis. At present convolutional codes are

extremely attractive for situations in which it is acceptable to have rates (i.e. ratios of transmitter information digits to total number of transmitted digits) of 1/N where N is an integer. For higher rates than 1/2 block codes are of most interest since the process of decoding convolutional codes then rapidly becomes more complex.

1.2. Definition

An error correcting block code consists of a set of N n-tuples, called the codebook, where an n-tuple is a row vector of n symbols chosen from a set of q elements (the code alphabet), and is termed a codeword.

1.3. Metrics

In order to correct errors in a corrupted codeword it is necessary that each codeword is sufficiently different from all others in the codebook that the corrupted codeword is "closer" in some sense to the uncorrupted codeword than to any other in the codebook. It is this quality of "closeness" that is termed the "distance" between codewords, and the manner in which the distance is determined is called the "metric".

If codewords are corrupted in such a way that symbols are changed at random by a random degree, then the "Hamming distance" is a useful metric. The Hamming distance between two codewords is defined as the number of symbols by which they differ.

The Hamming metric is not the only type of metric. Depending upon the nature of interference to codewords - for example the form taken by the channel interference and the way in which the channel carrier is modulated by the codewords - various other metrics may give a more useful definition of distance between codewords. As an example, the Lee metric gives the "Lee distance" between two codewords where symbols are chosen from a field of q elements. The Lee distance between the codewords is defined as the sum of the distances between each respective symbol of the codewords, where the distance between each symbol is the modulo -q sum of the values of the symbols (Lee 1958, Berlekamp 1968).

The Hamming metric is suited best to orthogonal modulation schemes, whilst the Lee metric is best suited to phase modulation systems. Either metric is only an approximation to the perfect metric for any practical situation, but such approximations are generally adequate to provide practical solutions.

In the binary case; that is, when the symbol alphabet consists of two values, the Lee and Hamming metrics are identical. In this thesis only the Hamming metric will be considered, and in the main only binary codes dealt with.

1.4. Minimum distance and error correcting power

The two codewords which are closest together set a limit to the error correcting power of a given code. The distance between those two codewords is called the "Minimum distance" of the code, and is often written "dmin".

If a code has minimum distance dmin then, errors will be correctable provided they number at most $\frac{dmin - 1}{2}$, since the nearest codeword to such a corrupted codeword will be at least a distance of $\frac{dmin + 1}{2}$ away, except for the uncorrupted codeword which will be at most $\frac{dmin - 1}{2}$ away.

1.5. Detecting Errors

This thesis is concerned with the correction of errors. It is nevertheless possible to use codes to detect errors. A code with minimum distance dmin can detect up to dmin -1 errors in a codeword, since by definition a codeword cannot be corrupted into another codeword without at least dmin errors occurring. In fact the power of a code to detect errors is generally much greater than this. If a codeword is corrupted by any number of errors into an n-tuple which is not a codeword then the error will clearly be detected. Since there are q^n n-tuples and only q^k of these are codwords, it is to be expected that the majority of errors will be detected for codes with $n > k$.

As an example, Peterson & Weldon (1972) state that a particular binary block code with n = 1023 and k = 1002 (usually abbreviated to (1023,1002)), will not only detect any error pattern of weight 5 but also any error pattern confined to 21 cyclically consecutive positions of the codeword, and all but 0.00005 percent of all other error patterns.

This property of codes makes the "Automatic Repeat Request" (ARQ) method of data transmission attractive whenever practicable. In this method, data is transmitted in block-coded form, and whenever errors are detected within a block a repeat is requested of that block, and a repeat continues to be requested until an error free block is received. Many variations on this theme are possible, the most importance perhaps being that where a small number of errors are corrected at the receiving end, larger number of errors just being detected and a repeat requested. By this method the data rate is increased at the expense of increased error rate. The trade-off is decided upon by a knowledge of the channel characteristics. If the code used has minimum distance dmin, and up to t errors ($t \leq (dmin - 1)/2$) are to be corrected, whilst up to e errors ($e \geq t$) are to be detected then $dmin = 2t + e + 1$. (Peterson & Weldon 1972).

1.6. Bounds on n, k, and dmin

It is clearly of interest to know the maximum value of dmin possible for given values of n and k. In order to do this many bounds on dmin have been found. Given here are the most important of these bounds. Three give upper bounds on dmin, and one gives a lower bound.

1.6.1. Hamming Bound

This bound was first found by Rao (1947) in connection with

experimental designs, it was applied to coding theory by Hamming in 1950 and is sometimes known as the "Hamming-Rao Volume Bound".

Consider a block code of length n and with symbols chosen from an alphabet of q symbols. If the code has an information rate of R then there are q^{nR} codewords, whilst the total number of possible n-tuples is q^{n}.

The number of n-tuples which are not codewords is therefore $q^{n}-q^{nR}$

If the code is to be capable of correcting all error patterns of weight t or less then each of these error patterns applied to each codeword must be associated with a distinct non-codeword. Then $V(t) \leqslant q^{n(1-R)}$ where V(t) is the number of n-tuples which are at a distance of less than or equal to t from any codeword in the code.

1.6.2. The Plotkin Bound

The Plotkin (1960) bound is based upon the observation that the minimum distance of a code cannot exceed the average distance between all pairs of distinct codewords. The result obtained by Plotkin for the Hamming metric is

$$(n - k) \geqslant \frac{q(dmin - 1)}{(q-1)} - \text{Log}_q(dmin - 1)$$

This bound is tight for high rate codes but weak at low rates.

1.6.3. The Elias Bound

This bound, first given by Elias (1960) and later refined by Wagner (1965) combines the Plotkin and Hamming bound to give a bound which is tight at medium rates. The bound shows that the Plotkin observation is a weak one, that in fact the average distance between pairs of most distinct codewords is considerably smaller than the average distance between all pairs of codewords. In the binary case, over the Hamming metric, the bound is

$$d \leqslant 2l\ (1 - l/n)(m/(m-1))$$

where l is an integer such that $\sum_{j=0}^{l} \binom{n}{j} > 2^{n-k}$

and m is the smallest integer for which

$$m \geqslant \sum_{j=0}^{l} \binom{n}{j} / 2^{n-k}$$

1.6.4. The Varshamov Gilbert Bound

This is a constructive lower bound on dmin, found by Gilbert (1952) and Sacks (1958), is a refinement of a bound proposed by Varshamov (1957). It gives dmin bounded by:-

$$\sum_{i=0}^{dmin-2} \binom{n}{i} (q-1)^i \geqslant q^{n-k}$$

1.6.5. Bounds for specific n, k, dmin

The above bounds are clearly useful in defining what it is possible to achieve with codes of given parameters. For specific values of n, k, and dmin it may be possible to tighten these bounds considerably by taking into consideration known optimum codes. That is, when it is known what is the best dmin achievable for some values of n and k, it is possible to give bounds on codes with other n and k because of known methods of or limitations on the combination of codes with one another to make new codes. This aspect of bound on dmin has been used by Helgert & Stinaff (1973), to give a table of bounds or dmin for values of n up to 63 and t up to 11 for binary codes, and a similar table has been compiled by McWilliams & Sloane (1977) for binary and ternary codes of linear or non-linear construction.

1.7. Perfect Codes

Perfect codes are those codes which meet the Hamming bound with equality. Therefore they are codes which will correct all errors of weight up to a maximum of t, and none of higher weight. There are only a few known perfect codes. The only perfect codes with medium rates are short, and the only long perfect codes are of either very high or very low rates. Examples of perfect codes are the Hamming codes (Hamming 1950); the repetition codes; and the Golay (23,12) binary triple error correcting code and (11,6) ternary double error correcting

code (Golay 1949).

Tietavainen (1977) has shown that there are no unknown linear perfect codes, and that if the code alphabet is of q sumbols and $q = p_1^r p_2^s$ where p_1, p_2 are distinct primes and r,s are positive integers then no perfect codes exist over this alphabet for error correction power $t \geqslant 3$.

For other non power of prime values of q it is still an open question whether perfect codes exist.

1.8. Quasi-perfect codes

Quasi perfect codes are those which correct all errors of weight t, a maximum of weight t + 1, and none of higher weight. As an example, all double-error-correcting BCH codes are quasi-perfect. (Gorenstein et al 1960) . Other examples are given by Peterson & Weldon (1972).

1.9. Optimum Codes

"Optimum" is a much confused term when applied to codes. One definition follows that of perfect and quasi-perfect codes, and terms optimum any code that corrects all errors of weight t or less with t as large as possible and as many errors as possible of weight t + 1. (See for example Bose and Kuebler (1958)).

A more lax definition, but one perhaps more useful is that an optimum code is that code with the shortest length for a given number of codewords and minimum distance (see for example Berlekamp 1968).

By this definition, codes formed by puncturing subspaces of different dimensions from a maximal length feedback shift register code are optimum (see for example Solomon and Stiffler (1965) and work on anticodes by Farrell & Faraq (1970, 1974 and 1976)).

The strictest definition of optimum is that given, for example, by Peterson and Weldon (1972) which is that an optimum code is one which gives the lowest probability of error over a random channel, for a given length and number of codewords.

1.10. Good Codes

For many practical applications of error correcting codes, where the decoding scheme now corrects up to (dmin -1)/2 errors and no more, it is most useful to know which codes have the maximum number of codewords for a given length and minimum distance of the code. For the lack of a better term, these codes may be called "good codes". See for example page 123 of Peterson and Weldon (1972) and the table of McWilliams and Sloane (1977).

1.11. The practical importance of optimality

It is easy to overestimate the importance of optimum codes, as defined in the previous paragraphs. In very many cases of practical data transmission there are three major considerations: the rate of transmission of information, the probability of errors

in recovered data, and the cost of recovering the data. From the above bounds on dmin it may be deduced that the longer a code the greater the rate may be for a given error rate in the recovered data. By sacrificing the gain available in rate it is often possible to construct a code which is not optimal but which nevertheless allows data to be recovered more cheaply because a particularly simple decoding scheme applies, even though for a given rate the code will be longer.

The major argument against this design philosophy is that a longer code incurs a greater delay before received data can be output from the decoder. If such a consideration is important in an application, then obviously the use of optimal codes must be considered; although even in this case it is possible that the decoding scheme required is so computationally lengthy that less decoding time is required by a longer, non-optimal code.

In conclusion, if the decoding cost of a data transmission system is limited then an optimal code may not provide the best solution - a non-optimal code of greater length but the same rate may afford a lower error output rate for the same decoder cost. If decoder cost is of no consequence, however, then the longest optimal code (in the sense of best error rate performance) should be used commiserant with acceptable delay in recovery of the data.

CHAPTER 2

Review of Linear Block Codes

2.1 Introduction

This thesis is almost entirely concerned with linear block codes. These codes have a mathematical structure which simplifies the calculation of their properties, and allows comparatively simple encoding and decoding processes to be devised.

A definition of linear block codes is as follows:

A linear block code of length n has symbols chosen from a field of q elements, and is a subspace, of order q^k of the linear vector space of all n-tuples; where k is the number of information symbols in the code.

Note that since the symbols of the codewords must be chosen from a field, q must be a prime or a positive integral power of a prime.

In this thesis, codewords and other sequences will be described variously as vectors, n-tuples, and polynomials. The vector description is self explanatory. An n-tuple is simply a sequence of n symbols. When a sequence is described as a polynomial it will be termed a function of a variable, e.g. f(x), and each symbol in the sequence will be identified as a coefficient of a power of that variable.

E.g. the sequence 32109 would be written and treated as

$$f(x) = 3x^4 \times 2x^3 + 2x^3 + x^2 + 9.$$

The number of non zero symbols in a sequence will be termed its Hamming weight, being identical to the Hamming distance of the sequence from the all zero sequence.

2.2. Generator Matrix

Since the codewords form a subspace, of the order q^k, it is possible to describe the entire code by any k linearly independent codewords. Any codeword can be formed by any linear combination of these codewords. Equivalently, if k independent codewords are arranged as row vectors in a k x n matrix then the rowspace of the matrix is the codebook of q^k codewords. Such a matrix is termed the generator matrix of the code.

If the matrix is put in its reduced echelon form and a k symbol row vector is multiplied by the matrix, the result is k symbols identical to the original row vector, followed by (n-k) other symbols. The codebook formed by the matrix is then said to be systematic, and the first k symbols are information symbols. The remaining (n-k) symbols are often termed parity check symbols, although strictly speaking they are only checks on parity when q=2.

2.3. The Parity Check Matrix.

A second important matrix used in the description and analysis of codes is the parity check matrix, which is defined as the matrix which has the codebook as its nullspace. That is, codeword vectors multiplied by the transpose of the parity check matrix give an all zero vector result, whereas all other n-tuples give a non-zero result.

If the generator matrix, G, of a code is given by

$$G = \left[I_k \mid P \right]$$

then the parity check matrix, H, is given by

$$H = \left[P^T \mid I_{n-k} \right]$$

or

$$H^T = \begin{bmatrix} P \\ I_{n-k} \end{bmatrix}$$

where P has dimension k x (n-k)

I_n has dimension n x n

and I_k has dimension k x k.

Therefore it can be seen that

$$GH^T = \left[I_k \mid P \right] \cdot \begin{bmatrix} P \\ I_{n-k} \end{bmatrix} = 0$$

2.4. The Syndrome

If an n-tuple is multiplied by H^T, the effect is as if the first k symbols are encoded by the generator matrix, and the "parity" symbols so obtained are added symbol-by-symbol to the remaining (n-k) symbols to given an (n-k) symbol result. It can be seen that if those remaining

(n-k) symbols are the parity check symbols of the encoding of the first k symbols then the result is all zero, and otherwise it is non-zero. This confirms the assertion that the result is zero if the n-tuple is a codeword, and non-zero otherwise. Such results are termed the "syndromes" of the n-tuples.

The syndrome of an n-tuple is related to the difference between the n-tuple and codewords. If a codeword is described by a vector c, and the vector e represents "errors" added to this codeword to give a "received" n-tuple r, then

$$r = c + e$$

then since the syndrome s, of the n-tuple, r, is given by

$$s = r H^T = (c + e) H^T$$

thus

$$s = c H^T + e H^T.$$

but, since,

$$cH^T = 0$$

$$s = e H^T$$

this relationship between a syndrome and an error vector is the basis of almost all decoding methods. A received, corrupted, codeword is decoded by finding s, and then satisfying the equation with a vector e of lowest possible Hamming weight in the case of random errors. There are as many solutions of the equation as there are codewords, and it is by finding the minimum weight solution for e that the codeword closest to r is found.

The set of possible solutions for e is called the "coset" of s, and the minimum weight solution is termed the "coset leader" of s. Thus the coset leader of a syndrome of an n-tuple is the vector which when added to the n-tuple gives as a result the nearest codeword.

2.5. Minimum Hamming distance of linear codes

The Hamming distance between codewords is defined as the number of symbols by which they differ. This is equivalent to the Hamming weight of the sum of the two codewords.

The minimum Hamming distance of a code is defined as the minimum Hamming distance between any two different codewords. This, then, is equivalent to the minimum Hamming weight of the sum of any two different codewords. It is a property of linear block codes that the sum of two different codewords is also a codeword. Therefore the minimum Hamming distance of a linear block code is equal to the minimum non-zero Hamming weight of any codeword in the code.

It is easier to find the minimum weight of a codebook than it is to find the minimum distance, since only N weight calculations are then required, compared to ${}^{N}C_{2} = N\ (N-1)/2$ weight calculations required to determine minimum distance directly. Furthermore, methods of analysis of the weight

distribution of linear codes have been found (e.g. MacWilliams (1963a), Berlekamp 1968) and from these may be deduced the distance properties of the codes. In the case of many specific linear block codes the weight distribution may be determined by mathematical analysis, again giving information about the distance properties of the codes.

2.6. Important classes of binary block error-correcting-codes

2.6.1 Repetition Codes

A binary repetition code has only two codewords, the all zero n-tuple and the all one n-tuple. There is therefore but one information bit, and the minimum distance of the code, dmin, is equal to n. A repetition code with n odd is a perfect, t= (n-1)/2, error correcting code of rate 1/n.

2.6.2 Hamming Codes

These codes, described first by Hamming (1950) are capable of correcting single errors.

The columns of the parity check matrix comprise all possible distinct non-zero n-tuples.

For any code, if one error occurs in a codeword then the syndrome is equivalent to one column of the parity check matrix, corresponding to the position in which the error occurred. Since the parity check matrix of a Hamming code consists of distinct n-tuples as columns, it is possible to identify the error position within a codeword from the syndrome provided only one error occurs.

If more than one error occurs then the syndrome obtained will be the bit-by-bit modulo-2 sum of the respective columns of the parity-check matrix. Clearly the modulo-2 bit by bit sum of n-tuples will give another n-tuple, either all zero or else an n-tuple corresponding to a column of the parity check matrix (since all n-tuples are represented as columns). More than one error can therefore only be interpreted by any decoding scheme as either no error at all or one error only.

The code is seen to be perfect, since it is capable of correcting exactly one error per codeword; no errors of weight greater than one being correctable.

2.6.3 The Golay Codes

The Hamming bound, in the linear binary case, reduces to

$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{(n-k)}$$

Golay (1949) noticed that for n = 23, k = 12, and t = 3 this bound is met exactly, suggesting the existence of a perfect (23, 12) triple error correcting code. Such a code was indeed found by him. It is a code which, as might be expected, has many connections with important aspects of combinatrics, notably the Mathieu group.

Golay also found an (11,6) ternary code, which is a perfect double error correcting code, following a similar observation on the Hamming bound in the linear ternary case.

Generator matrices for these two codes are given in fig. 2.1 and fig. 2.2.

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

Fig 2. 1. Generator Matrix of the (23,12) Golay Code.

$$\begin{bmatrix}
1&0&0&0&0&0&1&1&1&1&1\\
0&1&0&0&0&0&1&0&1&-1&-1\\
0&0&1&0&0&0&-1&1&0&1&-1\\
0&0&0&1&0&0&-1&-1&1&0&1\\
0&0&0&0&1&0&1&-1&-1&1&0\\
0&0&0&0&0&1&0&1&-1&-1&1
\end{bmatrix}$$

Fig 2.2. Generator Matrix of the (11,6) Golay Code.

2.6.4. Reed Muller Codes

(n, k, d) Reed Muller codes (Muller (1954), Reed (1954)) exist for any m and r m for which

$n = 2^m$ = length

$k = \sum_{i=0}^{r} \binom{m}{i}$ = no. of information bits

$n - k = \sum_{j=0}^{m-r-1} \binom{m}{j}$ = redundancy

$d = 2^{m-r}$ = minimum distance

The generator matrix of a Reed Muller (R.M.) code comprises k linearly independent rows, each row is a 2^m - tuple.

The i th row of the matrix consists of alternate groups of 2^{m-i} "ones" and 2^{m-i} "Zeros", the row starting with the "Ones". (The first row is therefore all ones; and the last, mth, row is a row of alternate ones and zeros).

The R.M. codes formed a basis for the much wider class of "Majority logic decodable" codes, and have been found to be equivalent to cyclic codes with an added overall parity check.

2.6.5. Finite Geometry Codes

It is possible to construct codes based upon the properties of finite geometries. The codes may be constructed using Euclidean geometries or projective geometries. (Weldon 1967, Kasami et al, 1966). The properties of the finite geometries allow the codes to

be majority-logic-decoded as outlined in Section 2.7.4.
For short code-lengths the finite geometry codes are equivalent to the BCH codes (section 2.6.8.) and far greater lengths are related to the BCH codes (Kasami et al 1966). Reed Muller codes are a subset of the projective geometry codes.

Short geometry codes are easily decoded, and are often the best known codes, or close to the best known. As the length increases, however, the rates of the codes become considerably worse than other known codes and the decoding algorithms become much more complex than those for other, better, codes, (for example the BCH codes see Chapter 4).

The Majority logic approach to decoding finite geometry codes was first introduced by Rudolph (1967) and a considerable simplification for certain of the codes has been found by L. E. Wright (1977).

2.6.6. Cyclic Codes

Cyclic codes are to date the most extensively studied class of error correcting codes. This class does not promise to contain good long codes, that is, codes that meet or even approach the Varshamov-Gilbert bound. In fact the most attractive feature of cyclic codes, their considerable mathematical structure, suggests that long cyclic codes may be relatively poor. Nevertheless, this structure enables codes to be constructed for many parameters, and enables good codes to be formulated which have practically achievable encoding and

decoding procedures for quite large block lengths.

A cyclic code is a code in which if a codeword $c(x) = \sum_{i=0}^{n-1} c_i x^i$ is a codeword then its cyclic shift $c(x) = \sum_{i=0}^{n-1} c_{(i+1) \bmod n} x^i$ is also a codeword. The codewords of any cyclic code of length n may all be described as the 2^k distinct multiples, modulo $x^n + 1$, of some polynomial g(x) where g(x) is a divisor of $x^n - 1$. See, for example, Petersen and Weldon (1972). The polynomial g(x) is known as the generator polynomial of the code, and has order (n-k), where k is the number of information bits in the codewords. The polynomial $(x^n - 1)/g(x)$ has order k and is the generator polynomial of the dual of the code generated by g(x).

The generator matrix of a cyclic code may be written as in fig. 2.3, since $x^i g(x)$ is clearly linearly independent of $x^j g(x)$ for all i, $j < n$, $i \neq j$. The reduced echelon form of this matrix is given in fig. 2.4; see for example Lucky et al (1968) for a detailed explanation.

$$\begin{bmatrix} x^{k-1} g(x) & & & & \\ & \ddots & & & \\ & & x^2 g(x) & & \\ & & & x\, g(x) & \\ & & & & g(x) \end{bmatrix}$$

fig. 2.3 Generator matrix of code described by g(x).

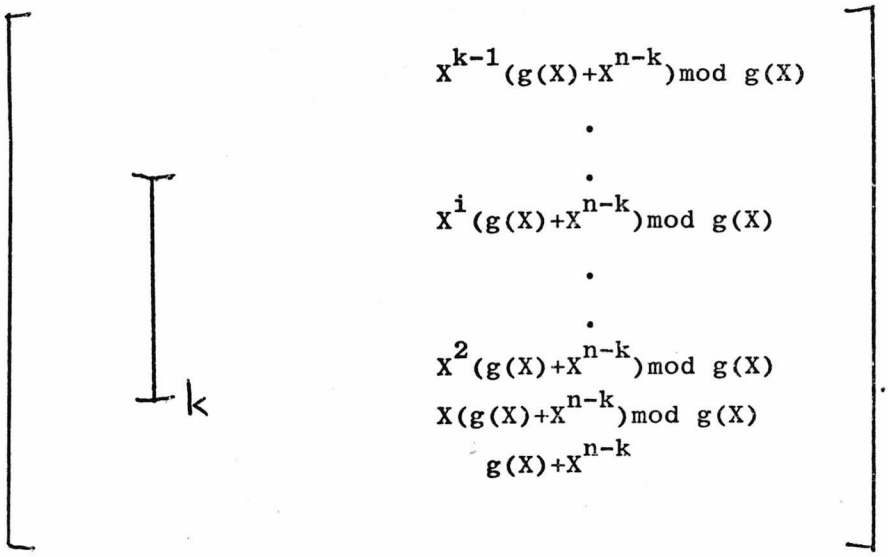


Fig 2.4 Reduced Echelon form of generator matrix of code described by g(x)

2.6.7. Quasi-cyclic codes

A code with n= mn , and k = mk is a quasi-cyclic code if every codeword shifted cyclically by M bits is also a codeword. Very powerful quasi-cyclic codes have been found for medium rates by Chen (1969) and the existence of very long quasi-cyclic codes that meet the Gilbert bound has been shown by Chen, Peterson and Weldon (1969). It is not easy to construct quasi-cyclic codes. Townsend and Weldon (1967) presented self-orthogonal quasi-cyclic codes, which are formed from difference sets; the codes are of low power but the ease with which the self-orthogonal codes may be decoded makes them of some interest nevertheless. Other work on quasi-cyclic codes has been published by Karlin (1969) and Hoffner and Reddy (1970).

2.6.8. B.C.H. Codes

B.C.H. codes are an important sub-class of the cyclic codes (Bose & Ray-Chaudhuri (1960), Hocquengam 1959). They are specified for a very large range of block lengths and rates, for all error correcting powers. For block lengths up to sixty-five there are only seventeen cyclic non-BCH codes which have a larger minimum distance than a BCH code of the same length and rate. Although for very long codes it has been shown that BCH codes are relatively poor (Lin & Weldon 1967), it is reasonable to

assume that for moderate lengths the BCH codes are generally the best cyclic codes.

Due to their wide range of parameters, and their good performance, the B.C.H. codes have become a "standard" against which other classes of error correcting codes are compared; also the decoding algorithms for the codes (see Chapter 4) are used as a gauge of the complexity of the decoding algorithms of other codes.

A cyclic code generated by the polynomial g(x) is a B.C.H. code iff g(x) is the lowest degree polynomial for which α^{mo}, α^{mo+1}, ... $\alpha^{mo+do-2}$ are an element of GF (2^m), that is, the Galois field of 2^m elements.

The length of a BCH code is equal to the lowest common multiple of the orders of the roots, and the minimum distance of the code is guaranteed to be greater than the longest number of consecutive integers, module n, in the set e= (e_1, e_2,...en-k) where α^{e1}, α^{e2}, ... $\alpha^{e\ n-k}$ are the roots of the generator polynomial of the code.

The most important BCH codes are those for which m=1, α is a primitive element of GF (2^m), and do= 2to +1. Then the generator polynomial of the code is given by:

$$g(x) = \text{LCM}\ (m_1(x),\ m_3(x),\ .....m_{2to-1}(x))$$

where $m_i(x)$ is the minimal polynomial of α^i.

The $m_i(x)$ have degree at most m, and therefore g(x) is of degree at most mt. Such a BCH code therefore has at most mto parity checks, and may correct up to to errors.

The theory of BCH codes is treated in depth by Berlekamp (1968) and Peterson and Weldon (1972).

2.6.9. Residue Codes

Residue codes are a class of cyclic codes which have been shown by Karlin (1969) and Chen et al (1969) to be related to quasi-cyclic codes.

If the order of q mod n divides (n-1)/e, then the eth residue codes of length n over GF (q) are defined (Berlekamp 1968) as the cyclic codes whose generator polynomials are

$$g_1(x) = \prod_{r \in R_0} (x - \alpha^r) \quad \text{(augmented code)}$$

and $g_2(x) = (x-1)g_1(x)$ (expurgated code)

where Ro is the set of eth residues mod n and α is a primitive nth root of unity in an extension field of GF(q).

(Note that if n is prime and if e divides (n-1) then the integer r, $1 \leq r < n$, is an eth residue mod n if and only if the equatron $x^e \equiv r \bmod n$ has solutions).

Of the residue codes, it is the quadratic residue codes (i.e. where e=2) which have been found to be the most important. For these codes it has been shown (E.g. by Berlekamp 1968) that the minimum distance, d, is lower bounded by $d^2 > n$. Furthermore, if $n \equiv -1 \bmod 4$ then the bound is improved to $d^2 - d + 1 \geq n$ (Mattson and Solomon 1961) for the augmented codes. In fact quadratic residue codes generally have large minimum distances for their length and rate, up to at least moderate block length. For example, the (23,12) binary Golay perfect triple-error-correcting code is a quadratic residue

code. A table of quadratic residue codes is given by Berlekamp (1968) in which minimum distances are given for block lengths up to 97 and upper bounds on minimum distance given for block lengths up to 48817.

Quadratic residue codes are invariant under the transformation $X \rightarrow X^r$ where r is a quadratic residue. This enables many quadratic residue codes to be decoded by a method known as permutation decoding (MacWilliams 1963). This method of decoding is nevertheless difficult to design for particular codes, and is somewhat complex in its realisation. In general, then, quadratic residue codes are hard to decode.

2.6.10. Product Codes

Product codes are formed by encoding information bits which are arranged as a two dimensional vector, or matrix. Each row of the matrix is encoded according to a code C_1 and each column encoded by a code C_2. The codewords may thus be visualised as in fig. 2.5.

Such a product code has a minimum distance equal to the product of the minimum distances of C_1 and C_2, a length equal to the product of the lengths of C_1 and C_2, a rate equal to the product of the rates of C_1 and C_2, and a generator matrix equal to the tensor product of the generator matrices of C_1 and C_2. Furthermore, in the case of a binary symetric channel, if C_1 has a probability of decoding error $f_1(p)$ and C_2 a probability of decoding error $f_2(p)$ then the product code is capable of being decoded with a probability of error at

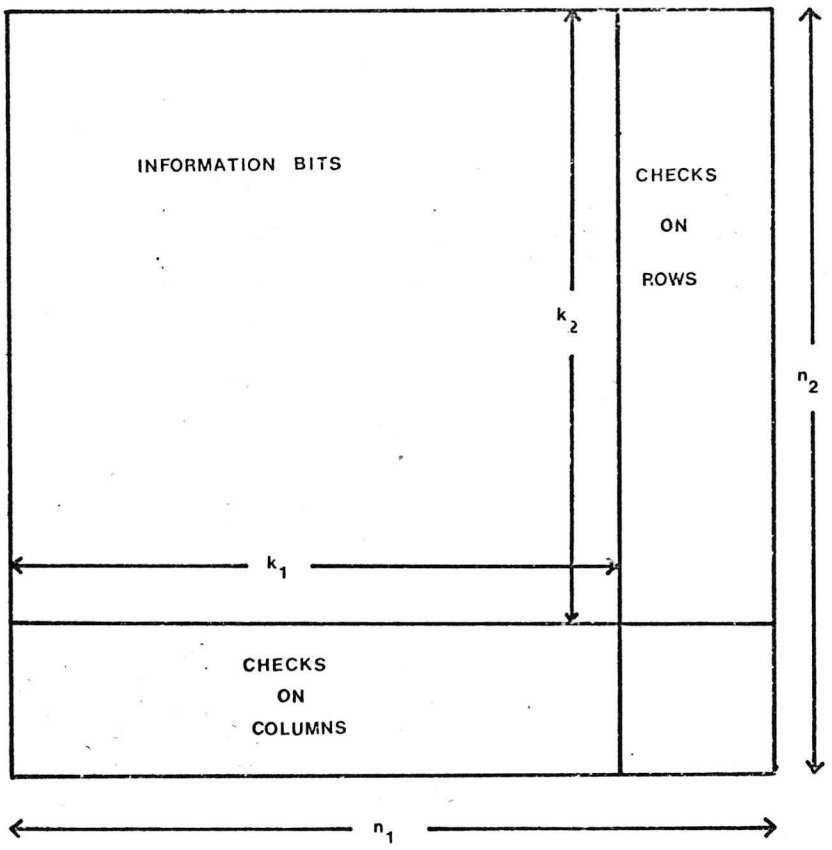


Fig. 2.5. A Product Codeword.

most(f_2 (f_1(p)) simply by decoding column by column and then row by row.

Elias (1954) has used the above relationships to show that by iterating the process of generation of product codes, using as each constituent code a Hamming dmin = 4 code, to achieve a product code of dimension as large as required, it is possible to construct codes whose probabilities of error approach zero as the length of the codes approach infinity whilst the rate remains finite. Although the rate falls far short of what is known to be possible the construction is one of the few known with this property.

In certain cases the product of two cyclic codes may be cyclic. Burton and Weldon (1965) have shown it to be sufficient that

$$n_1 p_1 + n_2 p_2 = 1 \bmod n_1 n_2$$

for $g(x) = \gcd(a(x^{p_2 n_2}) b (x^{p_1 n_1}), x^{n_1 n_2} - 1)$

to be the generator polynomial of a cyclic code which is the product of two cyclic codes of lengths n_1 and n_2.

Product codes have a relatively low rate for a given length and minimum distance, which limits their usefulness in many applications. Sugiyama et al (1976) have described an augmentation process for product codes which raises the rate of product codes considerably, at the cost of increased complexity in decoding the codes. The codes so formed are the best known for certain parameters; furthermore a form of this augmentation is used in the development of codes described later in this thesis (see chapters 6 and 7). For this reason the augmentation

process will be described in detail.

An augmented product code is shown in fig. 2.6. A codeword from a code with symbols chosen from a field of $2^{(n_1-k_1)}$ elements is superimposed on the row parity check bits, each symbol superimposed in binary form on a separated row codeword parity check section. The augmenting code has length equal to $n_a = n_2$, the length of the column code, and the minimum distance, d, equal to that of the product code. The augmented code has $k_1k_2 + (n_1-k_1)k_a$ information bits where k_a is the number of information symbols in the augmenting code, since one information symbol from the augmenting code will represent (n_1-k_1) binary information bits. It will now be proved that the augmented code has minimum distance $d= 2t+1$ when the product code has minimum distance $d=2t+1$. This will be accomplished by showing that if t errors or less occur in the codewords of the augmented code they may be correctly decoded.

An augmented code is decoded as follows. Firstly the parity check bits of the product code C are "reconstructed" by re-encoding the received k_1k_2 information bits of code C. The reconstructed parity check bits are then added modulo-2 to the respective received parity check bits to give an estimate of the augmenting codeword symbols. It can be seen that if at most t errors occurred in the received bits then at most t of the reconstructed symbols will be in error; therefore the reconstructed augmenting codeword may be correctly decoded. The decoded augmenting codeword is then added modulo-2 to the received codeword to give the unaugmented

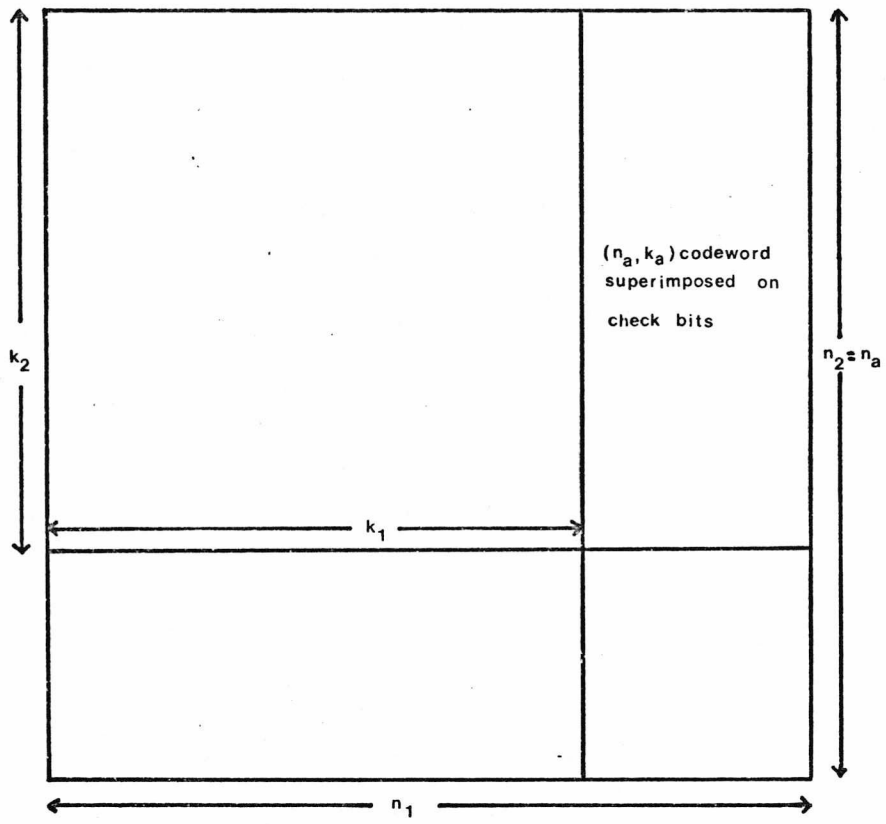


Fig. 2.6. An augmented product codeword.

product codeword corrupted in the same positions as the received codeword. This codeword may then be decoded in the normal way, to give a completely decoded word. Since up to t errors may be cleared from a received augmented codeword, the augmented code must have minimum distance at least 2t + 1. But if the all zero product codeword is augmented by the minimum weight codeword of the augmenting code, then the resulting augmented codeword will also have that weight, 2t + 1, and therefore the minimum distance of the augmented code is exactly 2t +1.

Further, the augmentation process may be extended to include the column check symbols of the product code, by superimposing a codeword from a code of length k with symbols chosen from a field of $2^{(n_2 - k_2)}$ elements expressed in binary form, and minimum distance 2t +1, on the column check symbols in the same manner as described above for row checks. Note that the column checks on row checks are not augmented since they are already augmented by the previous method. Proof that the augmentation method is valid follows that of the proof for row check augmentation. Decoding is performed first on the column check augmentation, to obtain unaugmented column checks, and then decoding may be performed on the row check augmentation and finally upon the unaugmented product code.

2.6.11. Concatenated Codes

These codes, devised by Forney (1966) may be visualised as two dimensional codes, in a similar manner to product codes.

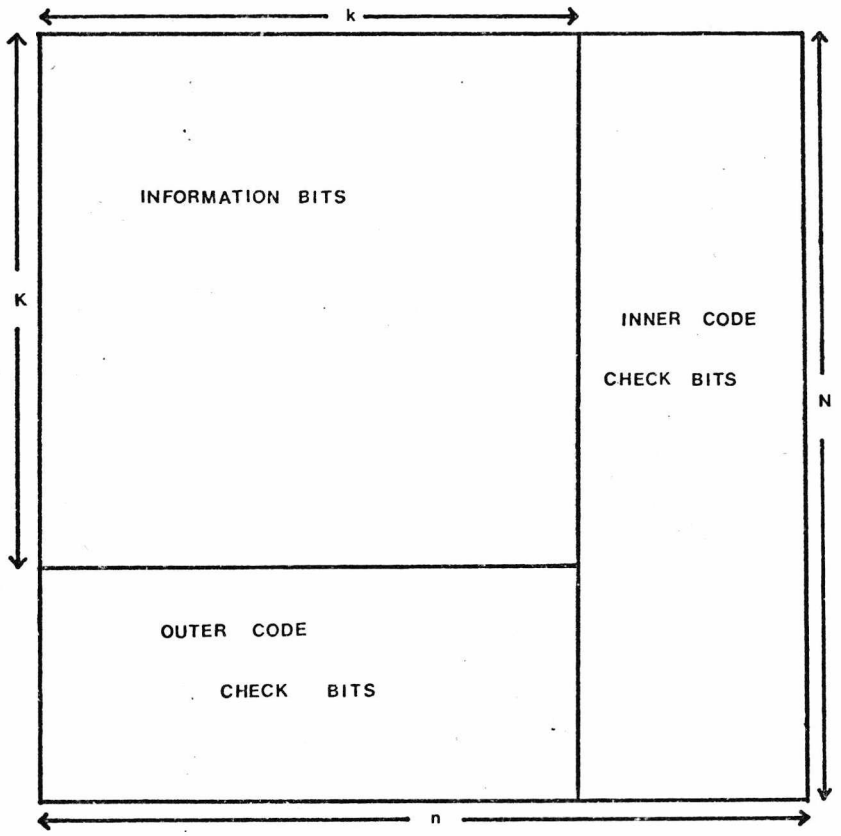


Fig. 2.7. A concatenated code word.

A concatenated code is represented in this way by fig. 2.7. Information symbols from GF(q) are arranged in K rows of k symbols. Each row is considered as a symbol in GF (q^k) and the K such symbols are encoded into an N symbol codeword in a code over GF (q^k) called the outer code. Each of the additional symbols from this codeword are arranged as rows of symbols in GF(q) underneath the information symbols as shown in fig. 2.7. and each row of the array is then encoded into n symbol codewords from a code over GF(q) called the inner code. The entire resulting codewords have length nN, with kK information bits. Forney has shown that with correct choice of inner and outer codes the probability of decoding error of concatenated codes approaches zero exponentially with block length for all rates below the channel capacity.

Sugiyama et al (1976) have shown how many more information symbols may be added to concatenated codes to give codes which are occasionally the best known. The method of augmentation is very similar indeed to their method of augmentation of product codes described above.

2.6.12. Justesen Codes

The Justesen codes (Justesen 1972) are a generalisation of concatenated codes, where the outer code is a Reed-Solomon code (Reed-Solomon (1960)) over GF (2^m) with length $N=2^m$ and the Ninner codes are distinct codes which may be shown to be the codes in Wozencrafts ensemble of randomly shifted codes (described by Massey (1963)). The codes are assymptotically good, with

$$\lim_{n\to\infty} \inf \frac{d}{n} \geq (1-r^{-1} R)\, H^{-1} (1-r) > 0$$

where r is the maximum of 1/2 and the solution of

$$R = r^2 \left[1 + \log_2 (1-H^{-1}(1-r)\right]^{-1}$$

and $0 < R < 1$ whilst the rate of the code, Rc, is greater than R

(H(x) is the binary entropy function).

Another use of a multi-level code in a pseudo-concatenated coding scheme is described in Appendix B.

2.6.13. Srivastava Codes

Srivastava codes are linear codes which are algebraically decodable but are not cyclic. The parity check matrix of a Srivastava code is given by

$$\begin{bmatrix} \frac{b_1^{\,l}}{1-a_1 b_1} & \frac{b_2^{\,l}}{1-a_1 b_2} & \cdots & \frac{b_n^{\,l}}{1-a_1 b_n} \\ \frac{b_1^{\,l}}{1-a_2 b_1} & \frac{b_2^{\,l}}{1-a_2 b_2} & \cdots & \frac{b_n^{\,l}}{1-a_2 b_n} \\ \frac{b_1^{\,l}}{1-a_{d-1} b_1} & \frac{b_2^{\,l}}{1-a_{d-1} b_2} & \cdots & \frac{b_n^{\,l}}{1-a_{d-1} b_n} \end{bmatrix}$$

Where l is any integer, $a_1, a_2 \ldots a_{d-1}$ are distinct elements from GF (q^m) and $b_1, b_2, \ldots b_n$ are the elements in GF $(q^m) - 0 - \{a_1^{-1}, a_2^{-1}, \ldots a^{-1}{}_{d-1}\}$ The length of the code is $n = q^m - d$ and the minimum distance of the code is at least d (Berlekamp 1968)

2.6.14. Goppa Codes

The Goppa codes are a class of linear codes which have an algebraic formulation and decoding algorithm, but which are not generally cyclic (Goppa 1970). A Goppa code is defined as the set of all vectors C that satisfy the condition.

$$\sum_{\gamma \in L} \frac{C_\gamma}{z - \gamma} \equiv 0 \bmod g(z)$$

where g(z) is any polynomial with coefficients in GF (q^m), q a prime power, and L is the subset of elements of GF (q^m) that are not roots of g(z).

Long Goppa codes have minimum distances which asymptotically meet the Gilbert bound (Berlekamp 1973) Primitive BCH codes are a special case of the Goppa codes (Goppa 1971) as are Srivastava codes.

That Goppa codes may be decoded using the Berlekamp algorithm for BCH codes is shown by Chien and Choy (1975) and Retter (1975). The complexity of decoding Goppa codes is considered by Sanvate (1977) who shows that they may be "erasures and errors" decoded in O (n $\log^2$ n) arithmetic operations, when using a decoding algorithm discovered by Sugiyama et al (1976) (See also corrections to this paper, 1976). Mandelbaum has shown that Goppa codes may be decoded by a method involving the theory of continued fractions (1977).

2.6.15. Alternant Codes

This class of codes contains all BCH codes, all Goppa codes, and Srivastava codes as well as generalisations of BCH codes made by Chien and Choy (1975). It has been shown that by a linear transformation of the syndromes, these codes may be decoded by

the application of the Berlekamp algorithm for the decoding of BCH codes (Helgert 1977).

2.7. Decoding Technique

There follows a resumé of the most useful decoding methods found for linear binary codes used over the binary symmetric channel.

2.7.1. Exhaustive Search

This is the most obvious method of decoding, in which a received vector is compared with every possible code vector and the "closest" in terms of Hamming distance is considered to be that most likely to have been transmitted. This method of decoding is suitable only for very short and/or low rate codes; the decoder will need to store the entire codebook of 2^{nR} n-tuples where n is the code length and R the information rate, and even for moderate values of nR this becomes very unmanageable in terms of storage hardware and of decoding time. Nevertheless, the method allows the most likely transmitted codeword to be chosen in every case (such a decoding method is termed "maximum likelihood"decoding) and so is often preferable to a decoding method which corrects only those errors guaranteeed correctable by the minimum distance of a code.

2.7.2. Syndrome Decoding

As previously described, it is possible to form an n(1-R) bit syndrome, s, from a received vector, which is related to the error vector, e, imposed on it by the equation

$$s = e \ H^T$$

where H^T is the transpose of the parity check matrix. The decoding aim is then to find a vector e of minimum possible weight which satisfies the equation. A conceptionally simple means of achieving this aim is to compile a table of minimum weight error vectors for each syndrome, that is a table of coset leaders, and to use the table when decoding. To construct the table of $2^{n(1-R)}$ error vectors (which are n-tuples), it is generally necessary to follow an exhaustive procedure described in detail by Peterson and Weldon (1972) and Lucky Salz and Weldon (1968) for example. The decoding method then consists of three stages:-

i) form a syndrome by multiplying the received vector by H^T.

ii) look up the coset leader corresponding to the syndrome.

iii) add the coset leader to the received vector to give the most likely transmitted codeword.

This decoding technique is only suitable for codes which are of high rate, small n-k, or both. Again, the method is a maximum likelihood decoding method. It is less complex than exhaustive search decoding iff $R < 1/2$.

2.7.3. Symbol by Symbol decoding

This decoding algorithm, discovered by L.D. Rudolph (1976) is the only general decoding method known which makes specific use of the linearity of linear codes - that is, it uses the property that linear codes have codewords from a subspace of a vector space.

The algorithm may use reliability information on each received bit, and in this form it is an optimum method of decoding. That is, no decoding method is better than symbol-by-symbol decoding.

The complexity of the decoder is related to the number of codewords in the dual code of the code to be decoded, and therefore the method is of most use for codes of extremely high rate, i.e. with a small number of parity check bits. Unfortunately the application of the algorithm is therefore limited to codes with low minimum distance and/or short length.

2.7.4. Majority-Logic decoding

Reed (1954) first suggested majority-logic decoding, in connection with the Reed-Muller codes. The technique was refined by others including Massey (1963) and Rudolph (1967).

Majority-logic decoding is based upon the concept of "orthogonality" defined as follows:

Given a collection E, of parity check equations or sums of parity check equations; if a set, P, of positions in the codewords is checked by each member of E, but no other position is checked by more than one member of E, then the set E is said to be orthogonal on P.

If it is possible to find on each separate information position in a codeword a set of J orthogonal parity - check equations or sums thereof, then it is possible to correct t= J/2 errors in the codewords. This is seen to be so since the parity-check bits of a codeword represent the calculation of the parity check equations of the code and so J independent estimates of each information bit may be made from the received parity check bits. If at most t errors occurred then at most half of the estimates can be incorrect. A decoding algorithm for such a code is therefore to make the J estimates of each information bit and invert a received bit only if it differs from more than one half of its estimates.

When the above properties apply to a code, the code is said to be one step majority-logic decodable up to distance d=J+1. For many codes, some of them the best known, this distance is equal to the actual minimum distance of the code. Such codes are termed "completely orthoganlisable in one step". The estimates of the information bits may be constructed directly from the received block (type I decoding) or from the syndrome of the received block (type II decoding). The type II decoding is preferable for codes with a rate somewhat greater than one half in these cases, such as with cyclic codes, where the syndrome is particularly early calculated.

If a set of J orthogonal check sums can be made on a set, P, of more than one information bit, then it is possible to determine the value of the sum of the bits in those positions

in the same way as described for just one bit in the one-step majority-logic case. It may be possible to deduce the values of several such sums of bits and then these may be used together with the original set of possible parity-check sums to give a larger number of known check sums. At this stage it may be possible to determine each bit in the codeword from J orthogonal sums - in which case the code is said to be two-step majority-logic decodable up to minimum distance $d = J + 1$, - or it may be that by repeating the process the bits may finally be determined in which case the code is L-step decodable where L is the number of stages of majority logic required. Again, some codes are decodable up to their actual minimum distance. It has been shown by Peterson (1972) that for an (n,k) code whose dual code has minimum distance $\bar{d}$, a maximum number of t_L errors can be corrected in each codeword by L-step majority-logic decoding where

$$t_L \qquad \frac{n - \lceil \bar{d}/2 \rceil}{2(\bar{d} - \lceil (\bar{d}/2) \rceil)} .$$

Majority-logic decoding is particularly attractive when it may be applied to cyclic codes, since in this case only one information position needs to be determined, the remainder then found by shifting the codeword cyclically.

Majority-logic decoding becomes more complex exponentially with L, the number of steps required. Thus it becomes extremely complex for many codes of even moderate length, and even when the codes are cyclic. One step majority logic is very simple, however. Long one -step majority logic codes of moderate or high rate are longer than other known codes with the same rate;

nevertheless the simplicity of their decoding may outweigh this consideration (see e.g. Townsend and Weldon (1967) and chapter 7 of this thesis). A considerable simplification of majority-logic decoding has been found for certain majority-logic decodable codes by L. E. Wright (1977).

2.7.5. Decoding cyclic codes

The considerable mathematical structure of cyclic codes is not only useful in the construction of good error correcting codes; it is also useful in the formulation of decoding algorithms for them. As a result, the cyclic codes are as a class the most widely used error correcting block codes.

Two subsets of long cyclic codes have feasible decoding algorithms - those which are majority logic decodable and those which are BCH codes. The decoding of BCH codes is discussed in chapter 4 and majority logic decoding has been discussed in section 2.5.4. Short cyclic codes may be decoded by methods which will be briefly reviewed here.

A major simplification of any decoding process for cyclic codes is that the syndrome of a received sequence may be calculated in a very simple manner. The syndrome S of a received sequence r is given by

$$s = r H^T$$

where H^T is the transpose of the parity check matrix of the code. In the case of cyclic codes the ith row of H^T is equivalent to the remainder from dividing x^{n-1-j} by g(x); therefore the calculation of rH^T may be performed by expressing r as a polynomial and dividing it by g(x). The remainder is rH^T

(Peterson and Weldon 1972). The division may be accomplished in practice by the use of a feedback shift register wired to divide by g(x) as a result of shifting the received sequence into the register. Full descriptions and explanations of such division circuits may be found in Peterson and Weldon (1972), Lucky et al (1968) and Berlekamp (1968).

Another major simplification in the case of many cyclic code decoders arises from the fact that the syndrome s^1 of a sequence r^1, where r^1 is a single cyclic shift of a sequence r, is obtained simply by shifting the syndrome, s, of r once in the syndrome forming register. This allows any circuit which is capable of correcting just one bit of a received sequence by an operation on the syndrome to correct all of the bits simply by shifting the syndromes in their registers.

Particular forms of decoder for short cyclic codes may now be discussed.

(a) Meggitt Decoder

In this form of decoder, a combinatorial logic circuit is used to "recognise" those syndromes corresponding to an error in the first bit in a received block. Thus if the output of the logic circuit is a "one", the first bit in a buffer register containing the received block is inverted. The syndrome and the buffer register may then be shifted once to enable, by virtue of the properties described above, the second and subsequent bits to be decoded. The major complexity of the decoder lies in the complexity of the combinatorial logic circuitry. (See Peterson and Weldon (1972) page 235 but note that the complexity of the decoder.

for a (23,12) triple error correcting code is overestimated, since the number of possible triple error patterns with a "one" in the last position is half of that given).

It has been suggested (Rocha 1976) that the combinatorial circuit may be replaced by a programmable read only memory (PROM) which with accelerating advances in microcircuitry is becoming feasible for ever longer codes. The technique of using PROMS may be extended to correct more than one bit of a received sequence at a time, depending upon the storage capability of the PROM. In the limiting case the decoder then becomes a syndrome search decoder in which the minimum weight error pattern corresponding to each syndrome is stored.

(b) Error trapping

This form of decoding, due to Rudolph (1967), relies upon the ability to detect when all errors occur in the parity check bits of a received corrupted codeword.

For any linear block code, if a codeword is corrupted only in its parity check bits then the syndrome of the corrupted codeword is identical to the error pattern. This may be seen from the construction of the parity check matrix, H , as will now be shown.

If the generator matrix, G, of a code is given by

$$G = \left[I \mid P \right]$$

where I has dimensions k x k and P has dimensions k x (n-k) then H^T, the transpose of the parity check matrix, is given by

$$H^T = \begin{bmatrix} P \\ - \\ I \end{bmatrix}$$

The syndrome, S, of a codeword corrupted by an error pattern, e, is given by

$$s = e\,H^T = e\begin{bmatrix} P \\ - \\ I \end{bmatrix}$$

therefore, if the first k bits of e are zero,

$$s = e''\,(I) = e''$$

where e" are the last (n-k) bits of e. Trivially, if at most t errors occur then the weight of S is at most t.

Now consider the case where at least one bit of e^1, the first k bits of e, is non zero - say i bits where $i > o$. Then the syndrome, s, is the bit by bit modulo. - 2 addition of e^1P and e"I.

If the minimum distance of the code is d, then x G, where x is any non-zero k-tuple, must have weight at least d. Let xG = c, and c^1 , c" be the first k, and the final (n-k) bits, respectively, of c. Then clearly $c^1 = x$ (the code is systematic), and if x has weight w so does c^1 have weight w. Therefore c" must have weight at least d-w.

But, c" = xP therefore if we let c" = e^1, it is seen that e^1P has weight at least d-i.

Hence s is the bit-by-bit modulo-2 sum of e^1 (which has weight at least d-i) and e'' (which has weight at most t-i provided that at most t errors have occurred). This sum has weight at least (d-i)-(t-i) = d-t.

Since $d = 2t + 1$ the syndrome has weight at least $t + 1$.

It has therefore been shown above that if and only if all the errors in the systemmatic linear block code occur in the parity check positions, the syndrome will have weight at most t provided that at most t errors occurred, and that the syndrome will then be identical to the error pattern.

This fact is the basis of error trap decoding. The syndrome of a received block is tested to see if its weight is less than $t + 1$. If so the syndrome is added bit by bit to the parity check positions of the block, and correction is assumed to be complete. If not, the received block is shifted once, cyclically, and the syndrome shifted once in its register to give the syndrome of the shifted block. The test is again made on the weight of the syndrome, and if possible correction made. This process may be executed n times, and if the errors in the original block were confined to (n-k) consecutive positions, they will eventually be corrected. In order that the errors be guaranteed correctable, it is necessary and sufficient that the rate, R, of the code be bounded by

$$R \leqslant 1/t$$

The decoding method is therefore seen to be useful only for low rate codes.

(c) Error trapping with windows

This method of decoding is a modification of the error trapping method, which allows codes of rate higher than 1/t to be corrected.

It is possible that although all t errors may not be confined in (n-k) positions, they may be confined to the final (n-k) positions and one of relatively few other positions termed windows.

The decoding procedure may then be:

(1) test for the syndrome weight $< t + 1$.

(2) if the test is positive decode as for the error trapping case.

(3) otherwise invert each "window" bit in turn repeating step (1) each time.

(4) if still the errors are not "trapped", shift the syndrome once, and shift the received block one, then repeat steps (1), (2), and (3).

The procedure may be carried out n times, by which time if less than t + 1 errors occurred on the received block correction will have been made.

The additional complexity of "error-trapping with windows" over "error trapping" depends of course on the number of windows to be tested.

From a design point of view, "windows" decoders are complicated in that the choice of window positions is difficult (see Kasami (1966)).

(d) Permutation Decoding

As explained earlier, a cyclic shift of a codeword in a cyclic code remains a member of the code. Thus a cyclic shift may be termed a "code-preserving permutation" and the code may be described as being invariant under the cyclic permutation.

Some cyclic codes are invariantunder other permutations, and if a set of permutations can be found which will in at least

one case transform any t errors in a codeword to (n-k) consecutive positions, that permutated word may then be error trapped as described earlier, and then re-permuted to give the original uncorrupted codeword. If a set of i permutations (including the cyclic permutation) is sufficient to allow this to be done, the code is called i-step permutation decodable (MacWilliams 1964).

For i-step permutation where i is even moderately high, the decoding process becomes very time consuming because in the worst case ni error trapping attempts must be made. Also, it is not known in general how to find suitable permutations.

The decoding method is therefore restricted to short codes.

CHAPTER 3

Decoding of Short Error Correcting Codes

3.1. Introduction

In this chapter the best method of decoding linear block error correcting codes of length less than 32 will be discussed, for error correction ability, t, less than seven. The codes considered will be those with the largest number of information bits, for a given length, that are known to the author. A table of these codes is given below in Table 3.1. Selected longer codes will also be considered.

The codes will be considered in order of error correcting power.

3.2. Single Error Correcting Codes

The most efficient linear block single error correcting codes are the Hamming codes, and their shortened versions.

The immediately obvious method of decoding the Hamming codes, which are cyclic codes, is by error trapping. Given at most one error per block it is possible to shift the error into the parity check section of the codeword when the shifted syndrome will have a weight of one and will be identical to the error pattern in the parity check bits. Adding this syndrome to the parity check bits of the shifted word and then shifting the word back to its original arrangement gives a corrected version of the received word.

It is of interest that, for a single error correcting cyclic code, the error may be trapped by shifting the received word, and the syndrome, by r bits at a time where r is the number of parity check bits in the codeword. In this way it might be possible to decode a received codeword with less delay than in the classical manner - but the clock

Table 3.1. - Codes with largest k for given n and t

n \ t	1	2	3	4	5	6
3	1					
4	1					
5	2	1				
6	3	1				
7	4	1	1			
8	4	2	1			
9	5	2	1	1		
10	6	3	1	1		
11	7	4	2	1	1	
12	8	4	2	1	1	
13	9	5	3	1	1	1
14	10	6	4	2	1	1
15	11	7	5	2	1	1
16	11	8	5	2	1	1
17	12	9	6	3	2	1
18	13	9	7	3	2	1
19	14	10	8	4	2	1
20	15	11	9	5	3	2
21	16	12	10	5	3	2
22	17	13	11	6	4	2
23	18	14	12	6	5	2
24	19	14	12	7	5	3
25	20	15	12	8	6	3
26	21	16	13	9	7	4
27	22	17	14	9	7	5
28	23	18	14	10	8	5
29	24	19	15	11	9	6
30	25	20	16	12	10	6
31	26	21	16	12	11	6

circuitry is made more complex, and the reduction in the delay would normally be of little value.

Another method of decoding Hamming codes is by majority logic decoding (e.g. Peterson & Weldon 1972). A Hamming code of length $2^m - 1$ is decodable in m - 1 steps by this method. This means that for anything but the shorted codes the decoding logic becomes rather complex. The decoding circuitry required for the (7,4) Hamming code is shown in Fig. 3.2., this is a two step majority logic decodable (m.l.d.) code.

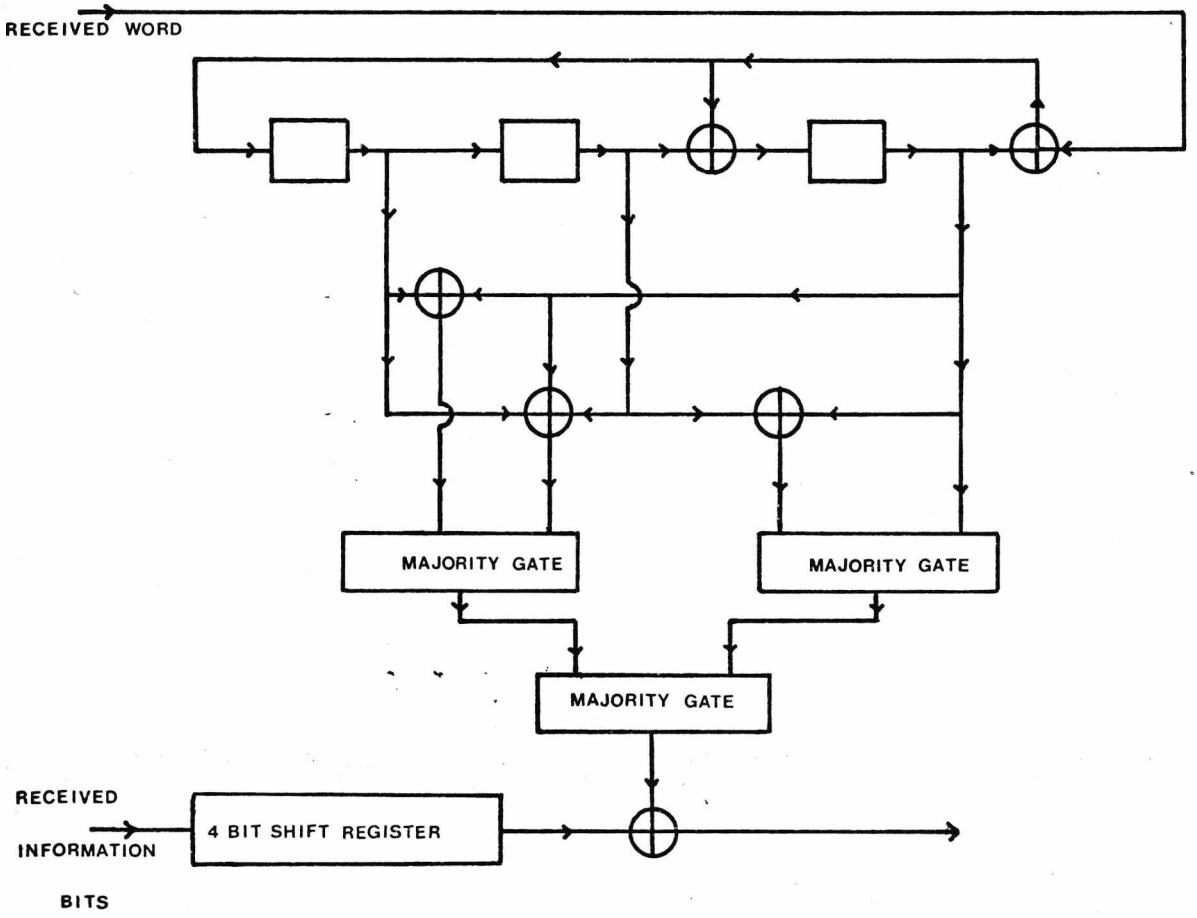


Fig. 3.2. - A 2-Step Majority Logic Decoder for the (7,4) Long Cyclic Hamming Code

A third, and most satisfactory, method of decoding, is that of Meggittdecoding (Peterson & Weldon 1972). Here a logic circuit recognises those syndrome patterns associated with an error in the final bit of a cyclic codeword. By progressive shifting of a received word the errors are cleared as they enter the final bit position. In the case of single error correcting codes there is only one syndrome to recognise, and that is a 'one' preceded by m - 1 'zeros'.

As the length of Hamming code increases, the decoding complexity increases only slowly - apart from the syndrome calculating circuit and a shift register to store the received word, all that is required is an 'm' input 'nand' gate and one inverter. The decoder for the (7,4) Hamming code is shown in Fig. 3.3, and can be seen to be simpler than that of Fig. 3.2. whilst also simpler than, but similar in its mode of operation to, the error trapping circuit.

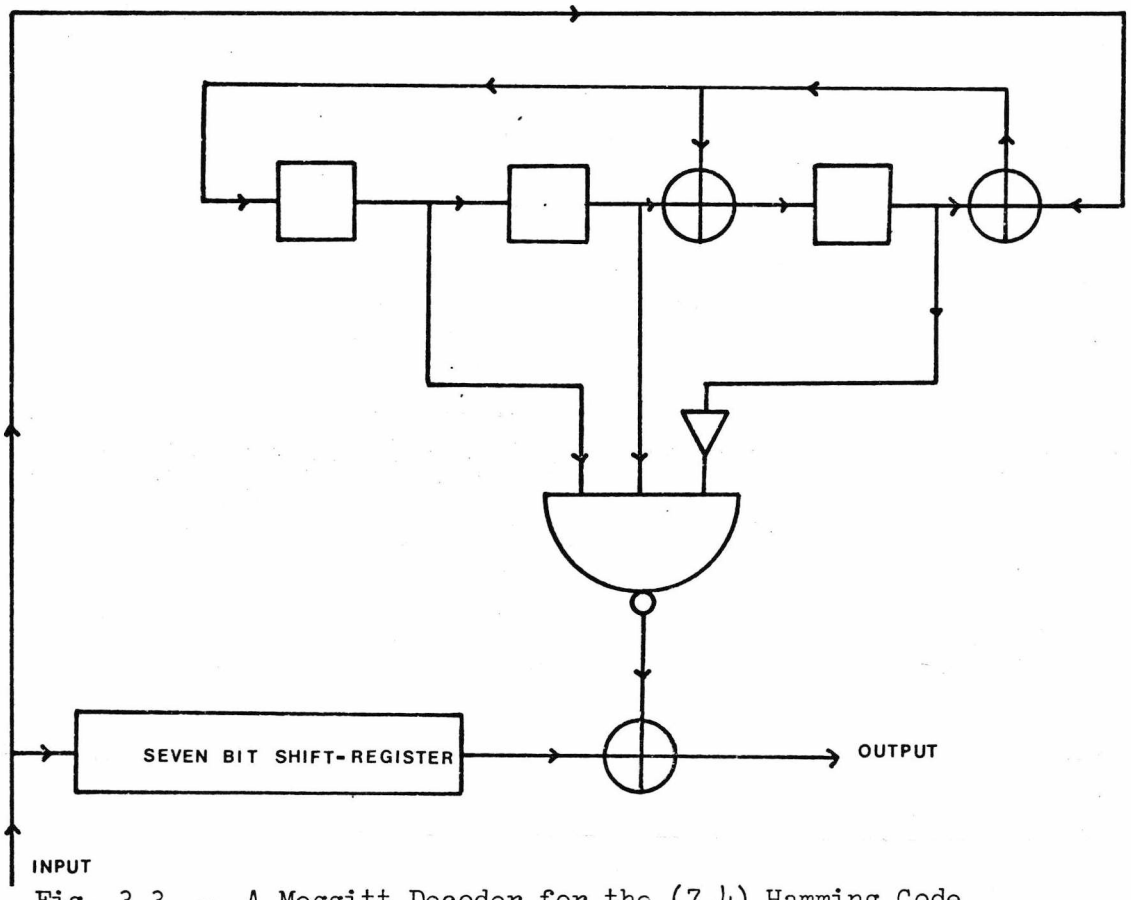


Fig. 3.3 - A Meggitt Decoder for the (7,4) Hamming Code

Shortened Hamming codes may be decoded by any of the preceding methods simply by inserting dummy 'zero' information bits in the missing 'shortened' positions. A more elegant method of achieving the same object is to modify the syndrome calculating feedback shift register such that the incoming bits are pre-multiplied by X^i where i is the number of bits by which the cyclic code has been shortened. Then, where r(X) is the polynomial representation of the received bits, the syndrome of $X^{n-k+i} r(X)$ is calculated instead of the usual $X^{n-k} r(X)$, which then, when used with any of the decoders described for Hamming codes, causes the first received symbol to be the first corrected - no insertion of zeros now being required. The technique for modification of the shift register is given in detail by Peterson (1972 page 242). A decoder for the (10,6) shortened Hamming Code, using the above method applied to a Meggit decoder for the 15,11 Hamming code is shown in Fig. 3.4.

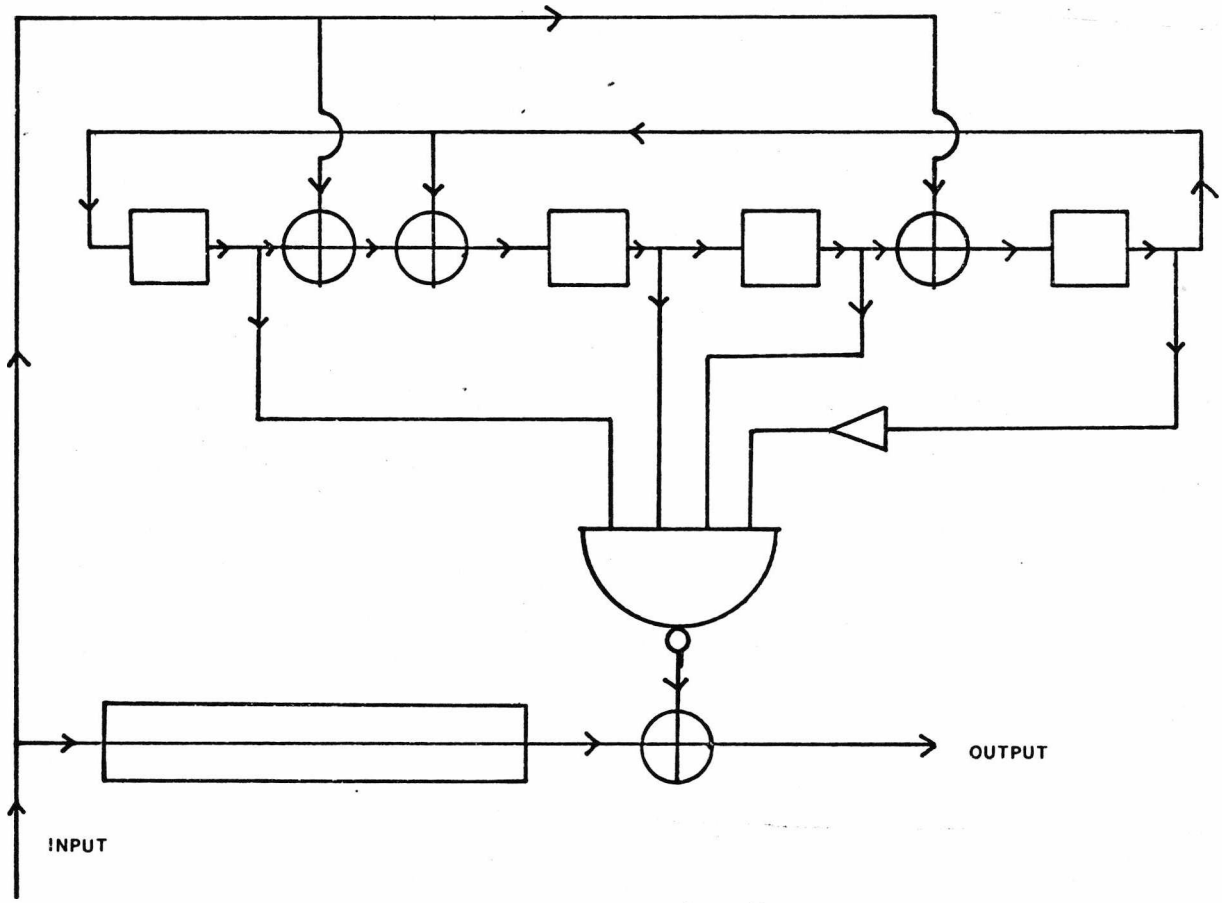


Fig. 3.4. - A Meggitt Decoder for the (10,6) Shortened Hamming Code

In conclusion it may be said that the most efficient type of decoder for single error correcting linear codes is the Meggitt type, modified when necessary for the shortened codes.

3.3. Double Error Correcting Codes

Unlike single error correcting codes, the most efficient linear double error correcting codes are of varying types. The codes will be considered in order of their number of information bits.

(a) The (5,1) Repetition Code

This is most simply decoded by a majority logic decoder - if more than two of the received bits are 'one' then five 'ones' are output, otherwise five 'zeros'.

(b) The (8,2) Code

This code is not cyclic. Two possible methods of decoding this code present themselves. The first is that of a search, since there are only four codewords, of length eight bits. This would require either suitable logic circuitry or a 256 word programmable read only memory (PROM) - with each address corresponding to a possible received word and the eight bit content being the corrected word. Suitable PROM's are now readily and cheaply available.

The second method is that of one step majority logic decoding. The circuitry required for this may be determined by first considering a possible generator matrix G, for the 8,2 code which is :

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

then, if a codeword C, from this code is written $C = k_1\ k_2\ p_1\ p_2\ p_3\ p_4\ p_5\ p_6$ where the k_i are information bits and the p_i are parity check bits, the p_i may be expressed in terms of the k_i as follows :

$$p_1 = k_1 \qquad p_2 = k_1 + k_2 \qquad p_3 = k_2$$

$$p_4 = k_1 \qquad p_5 = k_2 \qquad p_6 = k_1 + k_2$$

This defines the majority logic decoding circuiting, which is given in Fig. 3.5. The received word is shifted into register A, and the corrected information bits shifted out of register B. Register B is a two bit parallel-in serial-out register.

Clearly the latter method is the most cheaply implemented, requiring as it does only four modulo-2 adders and two majority logic gates. As the price of PROM's becomes lower, however, the the former method may become the cheaper.

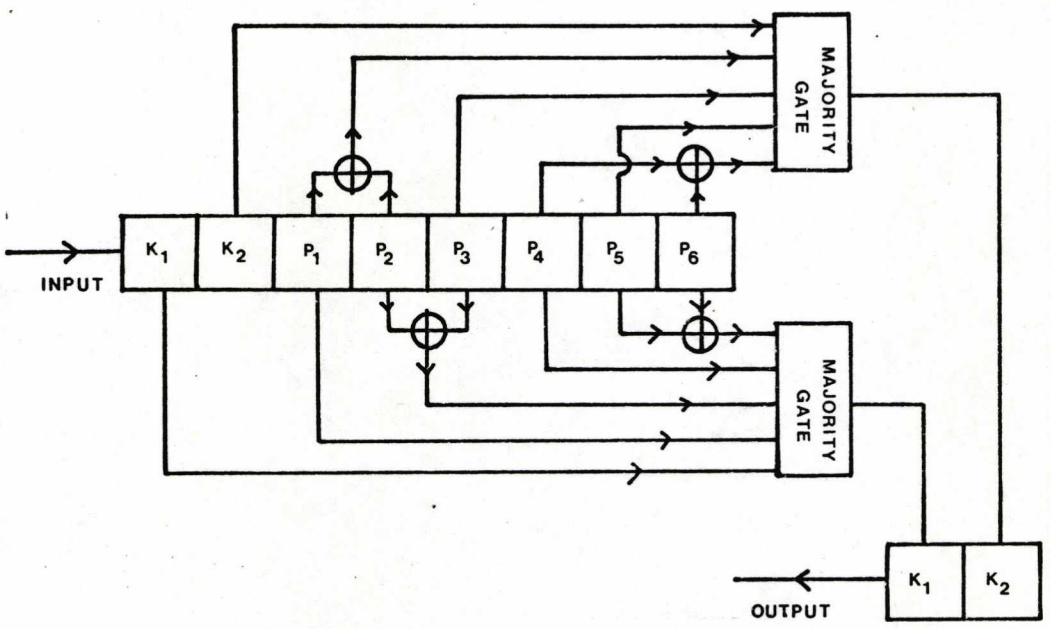


Fig. 3.5 - Majority Logic Decoder for the 8,2 Double Error Correcting Code

(c) The (10,3) Code

This code may decode by a shortened form of the decoder for the 11,4 code.

(d) The (11,4) Code

This code is not cyclic, but it may be constructed to be one step majority logic decodable.

The generator matrix G is then given by :

$$G = \begin{matrix} 1\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1 \end{matrix}$$

If a codeword is written as $(k_1, k_2, k_3, k_4, p_1, p_2, \ldots p_6, p_7)$ then the following equations, orthogonal on the k_i, may be formed :

$$k_1 = p_3 + p_4 \qquad k_2 = p_1 + k_4$$
$$k_1 = p_1 + p_5 \qquad k_2 = p_3 + k_3$$
$$k_1 = k_4 + p_6 \qquad k_2 = p_5 + p_6$$
$$k_1 = k_2 + p_2 + p_7 \qquad k_2 = p_2 + p_7 + k_1$$

$$k_3 = p_1 + p_2 \qquad k_4 = p_1 + k_2$$
$$k_3 = p_3 + k_2 \qquad k_4 = p_2 + p_3$$
$$k_3 = p_6 + p_7 \qquad k_4 = p_6 + k_1$$
$$k_3 = p_4 + p_5 + k_4 \qquad k_4 = p_4 + p_5 + k_3$$

hence a decoder of the same form as that in Fig. 3.5 may be constructed.

(e) The (13,5) and (14,6) Codes

These are best decoded as shortened versions of the (15,7) code. Peterson (1972) gives methods for modifying the decoder as required.

(f) The (15,7) Code

This code may be constructed as a B.C.H. cyclic code, and as such may be decoded by various schemes put forward for B.C.H. codes in general. There are, however, much simpler ways of decoding this particular B.C.H. code.

Since the code is cyclic and of rate less than $1/t$ where t is the error correcting power, it may be error trap decoded in the normal way. Furthermore, because the code has length $2k + 1$ where k is the number of information bits, the trapping of the errors may be speeded up on average by shifting the received bits and syndrome by two bits at a time during the trapping operation. This improvement is at the cost of complexity of the clocking circuitry and, since the maximum number of shifts required to guarantee the trapping of a double error remains the same as in the conventional decoder, the method would normally offer no advantage.

This code may also be one step majority logic decoded, as shown by Massey (1963). The parity check matrix of the code is given in Fig. 3.6, from which it can be seen that the four arrangements of the columns which are orthogonal on k_1 are

$$p_1 + k_2 + k_4, \quad p_2 + p_3 + p_5, \quad k_5 + p_6 + p_7, \quad k_3 + k_6 + p_8$$

where the k_i, p_i are as indicated in the figure.

This code is short and of rate less than one half. A Type II majority logic decoder is therefore the simplest to construct since no syndrome forming feedback register is then required. Such a decoder is shown Fig. 3.7. Note that feedback from the decoder output to the received bit register is used to allow correction of some error patterns of weight greater than two.

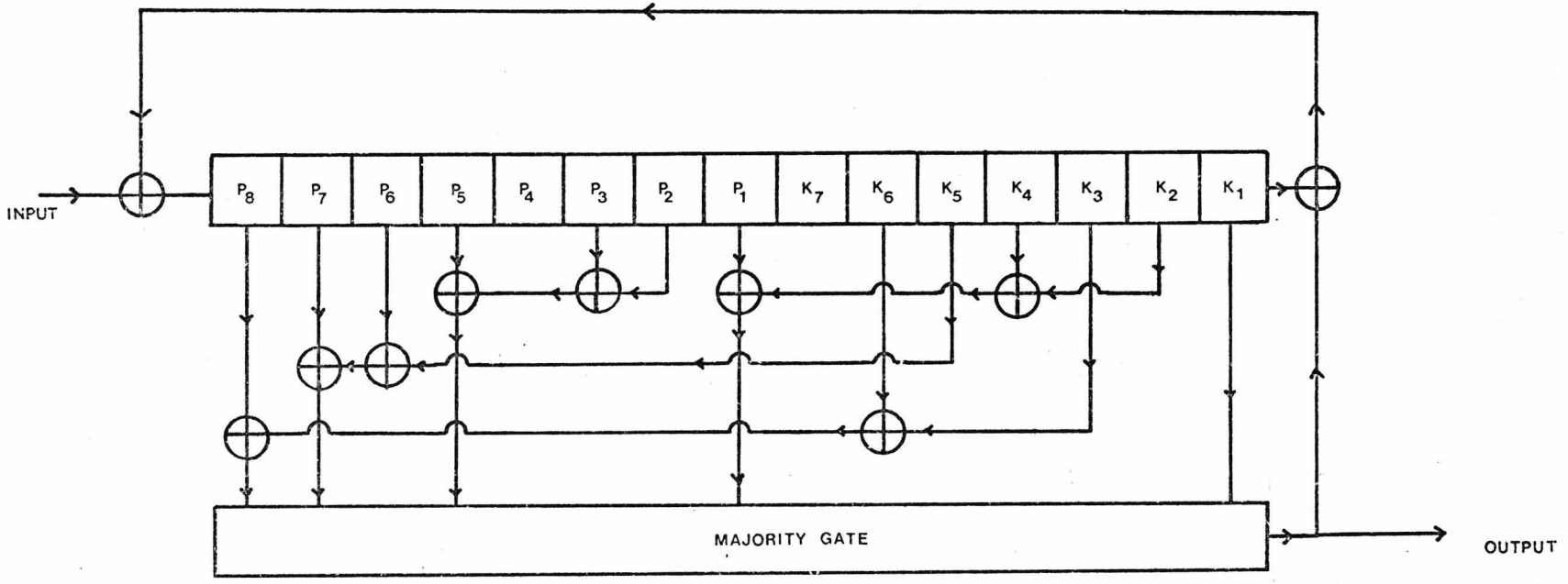


Fig. 3.7 A Type I Decoder for the 15,7 Double Error Correcting Code.

$$
G = \begin{array}{c}
\begin{matrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 \end{matrix} \\
\left[\begin{matrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{matrix}\right]
\end{array}
$$

Fig. 3.6 - The generator matrix of the cyclic, one step majority logic decodable (15,7) double error correcting code

(h) The (17,9) Code

This code may be formed as a non-primitive B.C.H. cyclic code.

As the code is of rate only slightly greater than one half it may be decoded quite simply by "error trapping with windows"; in fact only one "window" is required since only one error pattern, with its cyclic shifts, is not trappable.

An alternative decoding scheme would be to use the Meggit decoder. A PROM could be used to replace the logic circuiting. This PROM would need $2^8 = 256$ locations each corresponding to one of the possible syndromes of the received words. Each location would need to contain only one bit, this being the error value of the last bit of a received word with the respective syndrome. Since such a PROM is not expensive this may well be the cheapest method of implementation of the code.

(j) The (19,10) and (20,11) Codes

These codes may be decoded using suitably modified decoders for the (21,12) code.

(k) The (21,12) Code

This code may be constructed as a non-primitive B.C.H. code. As such it is cyclic, and the "error trapping with windows" decoding method may be applied - only one "window" is required. Alternatively a Meggitt decoder using a PROM may be used, or complete "syndrome decoding" employed using a 2^a(= 512) nine-bit location PROM. These latter two decoding methods are probably comparable in complexity; and both involve less decoding delay than the first method. Unfortunately nine-bit per location PROMs are not standard items at present. Nevertheless, the low cost of PROMs make the "syndrome decoding" approach attractive in this case.

(l) The (22,13) Code

This code may be decoded as a shortened version of the (22,14) code.

(m) The (23,14) Code

This code is not cyclic nor known to be m.l.d. and therefore its decoding presents considerable difficulty. The code has 2^9 = 512 syndromes, and therefore the simplest decoding method so far known should appear to be a matrix multiplication circuit to calculate the syndrome of a received word, followed

by a PROM of 512 locations each of 14 bits, which would correct errors in the 14 information bits.

It is in cases such as this, where the best known code has little known structure and is not very high or very low rate that decoding circuits are most complex. It can be seen that for codes somewhat longer than this one, a PROM suitable for syndrome decoding of the code would be inordinately large.

(p) The (31,21) Code

Again, this may be constructed as a B.C.H. cyclic code. The standard decoding algorithms for B.C.H. codes are applicable, but are not the simplest methods of decoding this code.

The code is not error trap decodable, as its rate is greater than one half. By decoding with error trapping plus windows, as in the 17,9 code, it is necessary to have windows in two positions of the codeword.

An alternative decoding method is that of permutation decoding. This code is two-step permutation-decodable. To decode the code in this manner an attempt is first made to "trap" errors in the normal way, by cyclic permutations of the codeword, and weighing of the resulting syndromes. If this process is unsuccessful then a second set of permutations is made, whereby if the received word is described as a polynomial $a_0x^{n-1} + a_1x^{n-2} + \ldots\ldots + a_{n-1}x + a_n$, it is permuted according to the rule $a_ix^j \rightarrow a_i\, x^{2j}$.

This permutation guarantees that the errors will now be correctable by applying "error trapping" to the newly created word, and then permutating back to the original codeword. This method would appear more simple than the "windows" method, it

will also cause less decoding delay.

3.4. Triple Error Correcting Codes

Again these will be taken in order of the number of information bits they contain.

(a) The (7,1) Code

This code may be one step majority logic decoded in the obvious manner.

(b) The (11,2) Code

It is possible to form an (11,2) triple error correcting code which may be one step majority logic decoded in a cyclic manner, even though the code itself is not cyclic. The generator matrix, G, of the code is given by :

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

If the information bits k_i and parity check bits p_i are labelled as in previous examples, the following orthogonal equations for k_1 and k_2 are obtained :

$k_1 = p_2, p_4, p_6, p_8 + p_3, p_9 + p_5, p_1 + p_7$

$k_2 = p_3, p_5, p_7, p_8 + p_2, p_9 + p_4, p_1 + p_6$

note that the estimates of k_2 are related in a simple way to those of k_1 - if the p_i are changed to p_{i+1} (mod 9) then the k_2 estimates are obtained from those for k_1.

The circuiting for such an arrangement is shown in Fig. 3.8. It operates as follows :

With S_1 closed, the received bits are shifted into register A, and the majority logic decodes information bit k_1, then switch S_1 is opened and the register A is shifted once to give $p_i = p_{i+1}$

(mod 9) which then allows the same circuiting to calculate information bit k_2. The information bit k_1 is at the same time shifted along register B, so that when k_2 has been calculated the register B contains both information bits, which may then be shifted out. Note that the first two bits of register A, containing the information bits received, are also shifted so that the correct k_i is automatically input to the majority gate for each bit decoded.

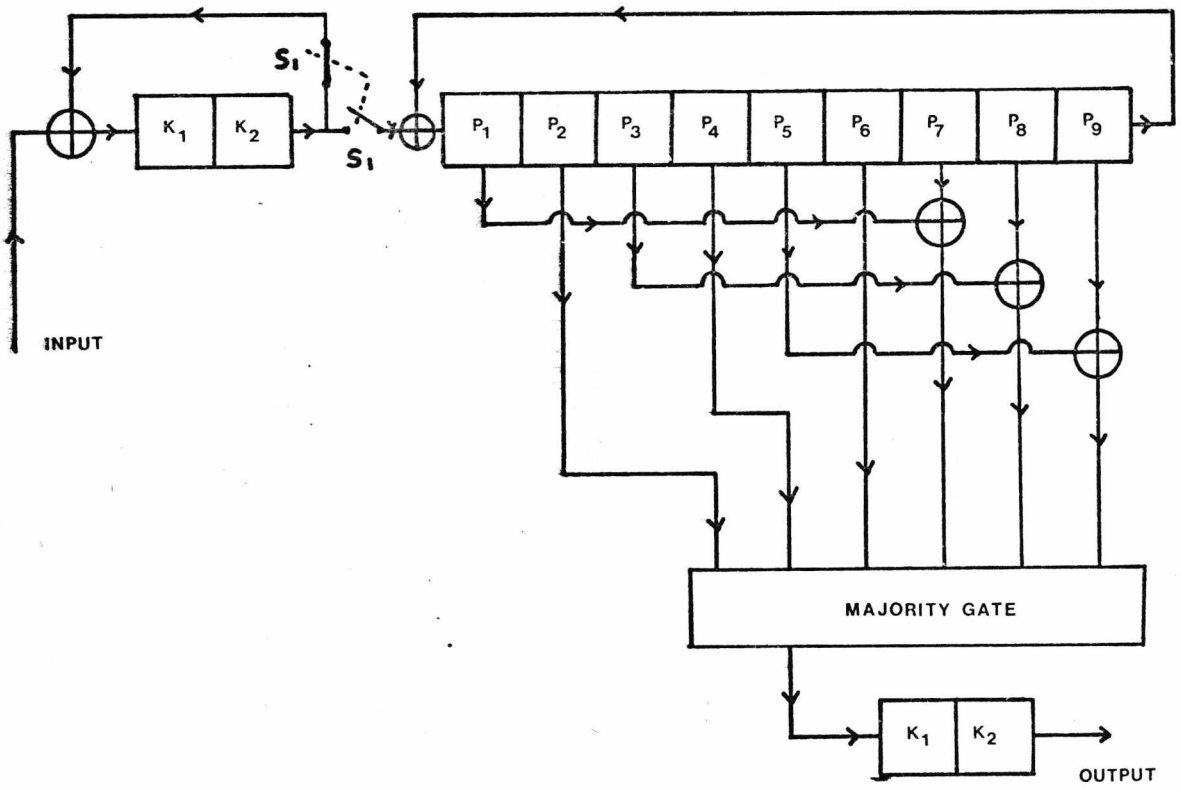


Fig. 3.8 - 1 Step Majority Logic Decoder for the (11,2) Triple Error Correcting Code

(c) The (13,9) and (14,4) Codes

These are best decoded as shortened versions of the (15,5) triple error correcting code.

(d) The (15,5) Code

This code may be constructed as a B.C.H. cyclic code.

Although the code could be decoded by the various algorithms available for the general class of B.C.H. codes, it may be decoded for more easily by other methods.

Firstly, since it is a cyclic code of rate exactly $^1/_t$ where t is the error correcting power of the code, it may be error trap decoded.

Secondly, it may be two-step majority-logic decoded, as shown by Massey (1963). The circuit for this decoder is shown in Fig. 3.10, and the generator matrix in Fig. 3.9. Note that a total of 47 'exclusive or' gates, and four majority gates are required, with a 15 bit shift register. This illustrates the considerable complexity of L step majority logic decoding even for short codes with limited error correcting power.

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

Fig. 3.9 - The Generator Matrix of the (15,5) Triple-Error-Correcting B.C.H. Code

(e) The (17,6), (18,7), (19,8), (20,9), (21,10), and (22,11) Codes

These codes may best be decoded as shortened versions of the (23,12) Golay code.

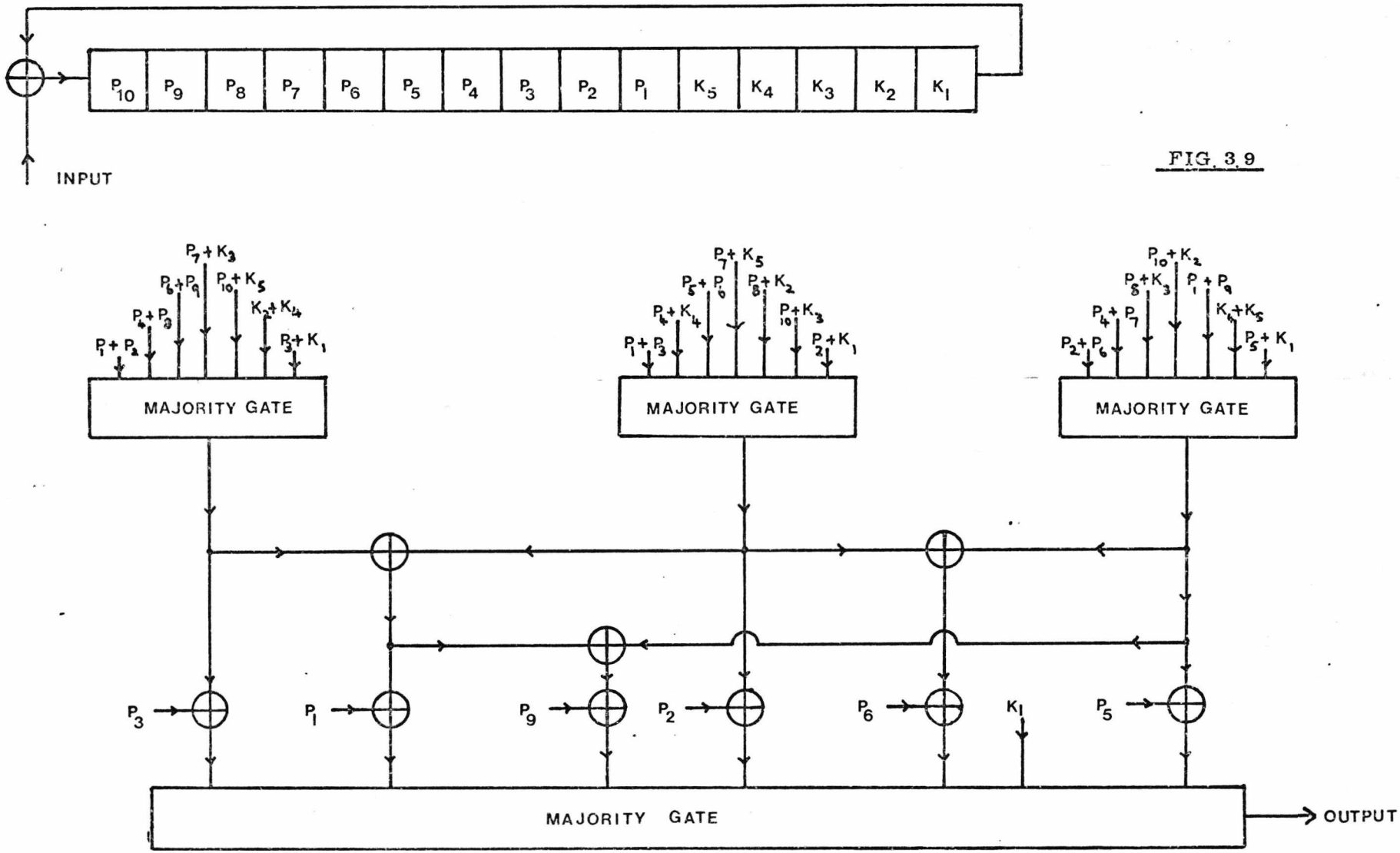

P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 K5 K4 K3 K2 K1
INPUT
P7+K3 P6+P9 P10+K5 K2+K4 P4+P8 P1+P2 R3+K1
MAJORITY GATE
P7+K5 P5+P6 P8+K2 P4+K4 P10+K3 P1+P3 P2+K1
MAJORITY GATE
P10+K2 P8+K3 P1+P9 P4+P7 K6+K5 P2+P6 P5+K1
MAJORITY GATE
P3 P1 P9 P2 P6 K1 P5
MAJORITY GATE
OUTPUT

FIG. 3.9

(f) The (23,12) Code

This code is the Golay perfect triple-error-correcting code, and may be constructed as a cyclic non-primitive B.C.H. code, equivalent to a quadratic-residue code.

The two most suitable decoding methods for this code are permutation decoding, and "error trapping with windows".

Permutation decoding of the code requires four steps (see for example Peterson and Weldon (1972) and Yip el al (1974)). This method is therefore rather complex and certainly time consuming.

The "error trapping with windows" method is less complex - requiring only two "windows" and therefore involving less decoding time. (Kasami 1964).

(g) The (31,16) Code

This code may be constructed as a quadratic residue code, in which form it may be decoded by permutation decoding (see MacWilliams 1963). In this form the code is also a primitive B.C.H. code and may therefore be decoded by the means described in Chapter 4 for triple-error-correcting B.C.H. codes. Also, as a cyclic code, the (31,16) code may be corrected by the "error trapping with windows" method (Kasami 1964).

Furthermore, the code may be constructed as a shortened Reed Muller Code; in which form it is two-step majority-logic decodable, and may also be decoded in the manner described in Chapter 6 for augmented conjoined codes of which shortened Reed Muller codes are examples.

Permutation decoding is a somewhat slow and complex method of decoding, not least in terms of the control circuitry

required. The "error trapping with windows" method is similarly complex and slow in this case owing to the number of "windows" required. The majority logic decoding algorithm therefore appears to be the simplest available.

As mentioned, the code may be orthogonalised in two steps. The decoding circuitry may be realised using one buffer register of 31 bits, seven majority gates, and 36 modulo-2 adders. (See Berlekamp 1968). However, it has been shown by Rudolph and Hartmann (1973) how certain cyclic codes, including the (31,16) code, may be majority-logic decoded in a somewhat different manner involving what they term "sequential code reduction"; in this way the decoding time is doubled, but the hardware required is reduced to two buffers, two majority gates and twelve modulo-2 adders together with the control circuitry common to both approaches. A further slight simplification of the circuitry required by the sequential code reduction method is claimed by Schmandt (1976).

Decoding the (31,16) code as an augmented conjoined code required decoding circuits for the (15,11) $d_{min} = 3$; (16,11) $d_{min} = 4$; and (15,5) $d_{min} = 7$ codes, all of which may be of the very simple error trapping type, together with some simple control circuitry. The decoder is treated in detail in Chapter 6 and can be seen to be comparable with the complexity of, but somewhat slower than, the majority-logic decoders.

3.5. Four-Error-Correcting Codes

(a) The (9,1) Code

This code may best be decoded by one step majority logic.

(b) The (14,2) Code

This code may be one step majority logic decoded in the same manner as the (11,2) triple-error-correcting code. The generator matrix is now given by :

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

and the orthogonal parity check equations are :

$$k_1 = p_2 = p_4 = p_6 = p_8 = p_1 + p_9 = p_{10} + p_3 = p_{11} + p_5 = p_{12} + p_7$$

and

$$k_2 = p_3 = p_5 = p_7 = p_9 = p_2 + p_{10} = p_{11} + p_1 = p_{12} + p_6 = p_1 + p_8$$

the decoding circuiting is arranged in a similar manner to that for the (11,2) code.

(c) The (17,3) Code

This code may not be constructed as a cyclic code and is therefore not decodable by error trapping.

There are only eight possible codewords, and therefore the code might be decoded by comparing a received sequence with each in turn.

More simply, the code may be decoded with one-step majority-logic. Also, the parity checks may be arranged such that the decoding may be performed in a cyclic manner, similar to that of the decoding of the (11,2) triple-error-correcting, and (14,2) four-error-correcting codes.

The generator matrix of the code is then given by :

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

if a codeword, C, is written

$$C = k_1, k_2, k_3, p_1, p_2, \ldots p_{13}, p_{14}$$

then the orthogonal parity checks are :

$$k_1 = p_4, p_{11}, p_1 + p_5, p_8 + p_{12}, p_3 + p_6, p_{10} + p_{13}, p_2 + p_7, p_9 + p_{14}$$

$$k_2 = p_3, p_{10}, p_1 + p_7, p_8 + p_{14}, p_2 + p_5, p_9 + p_{12}, p_4 + p_6, p_{11} + p_{13}$$

$$k_3 = p_2, p_9, p_1 + p_6, p_8 + p_{13}, p_4 + p_7, p_{11} + p_{14}, p_3 + p_5, p_{10} + p_{12}$$

the cyclic nature of these orthogonal equations suggests the decoder shown in Fig. 3.14.

The decoder operates as follows. The received sequence is shifted into the Register A, and switches SA are then opened, switches SB closed. The logic circuiting is then calculating the majority estimate of k_1, which is entered into Register B. The sections of the Register A marked 'shift' are then shifted once, so that now the majority estimate of k_2 is calculated and entered into Register B. A further shift gives k_3, and then the decoded information bits may be shifted out of Register B.

(d) The (19,4) Code

This code may be decoded as a shortened version of the (20,5) code.

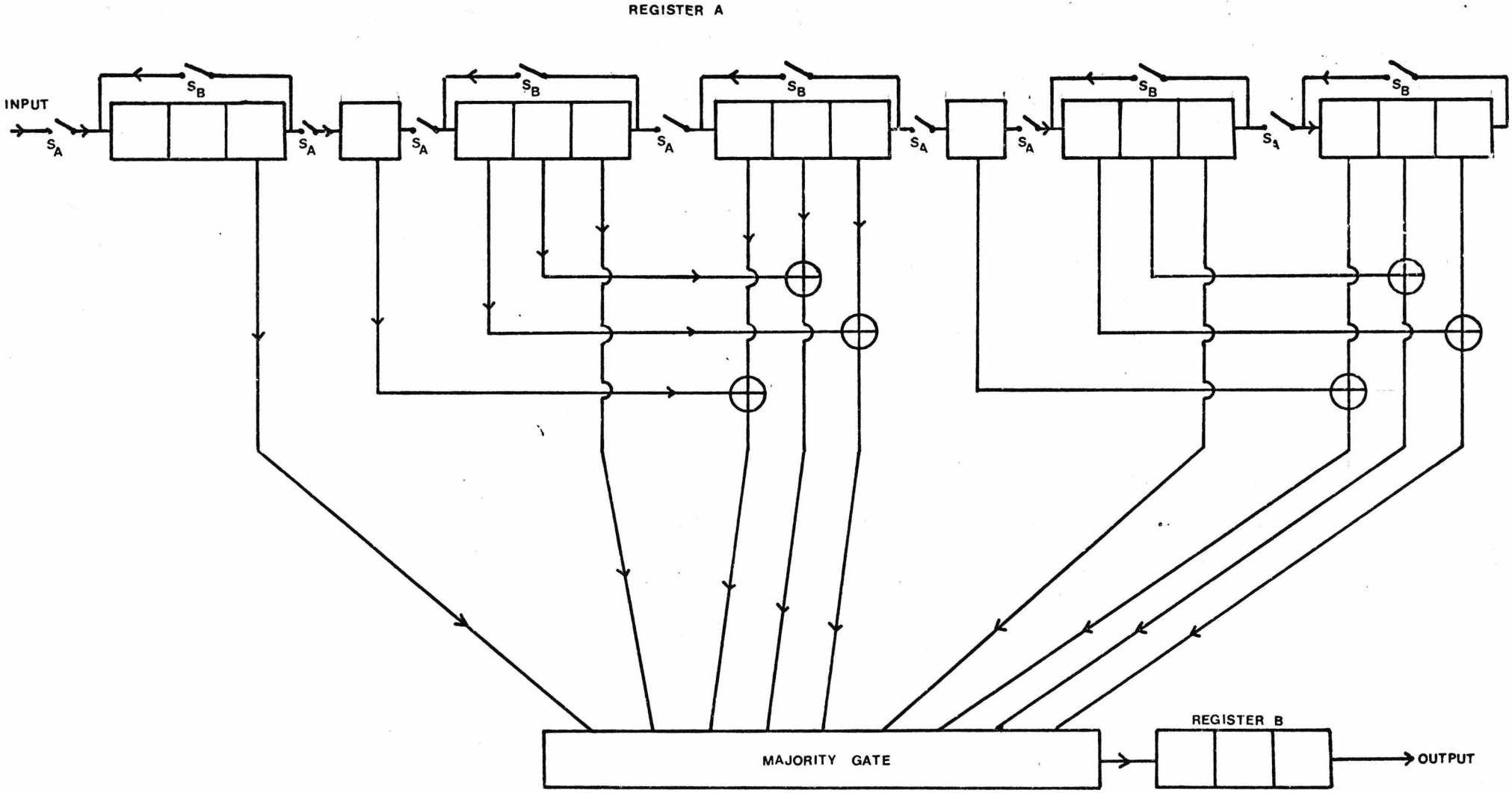


Fig. 3.14 - One Step Majority Logic Decoder for the (17,3) Four-Error-Correcting Code

(e) The (20,5) Code

This code is obtained by omitting a parity check bit from the (21,5) non-primitive B.C.H. code with minimum distance ten. A decoding procedure for this code may be devised as follows :

A (21,5) code is capable of correcting four errors and detecting five. When a (20,5) codeword derived from this code is received corrupted by at most four errors, an erroneous correction will not result from guessing the "missing" parity check bit value since this will introduce only one more error at most. If the guess is correct, or if not but nevertheless at most four errors then exist in the extended word, it will be possible to correct all the errors in the word. If the guess is incorrect and thereby introduces a fifth error, the presence of five errors is detected by the decoder. The basis of a decoding algorithm is therefore clear - a guess at the missing parity check bit is made, and correction attempted; if correction is successful no further action is required, if five errors are detected then the "guessed" bit is inverted and correction again attempted. Correction will then be possible provided at most four errors occurred in the original received word.

The problem of decoding the (20,5) code has now been reduced to that of decoding the (21,5) code with minimum distance ten. This decoding is very simple since the code is cyclic and of rate less than one quarter. The code may therefore be error trap decoded - a failure to be able to error trap implying that five errors have occurred.

The final decoding algorithm therefore consists of five major sections :-

(a) Construct a (21,5) vector from the (20,5) received vector by appending the "missing" parity check bit.

(b) Attempt to error trap the (21,5) vector - if successful deliver the corrected word to sink, otherwise procede to (c).

(c) Invert the appended parity check bit, and repeat the error trapping procedure. If successful deliver the corrected word to sink, otherwise five or more errors have been detected.

(f) The (22,6) Code

The existence of this code was shown by Calabi and Myrvaagnes (1964). No properties of the code are known which would enable the code to be decoded in any way other than those common to all binary linear block codes. Therefore, since the code is of low rate, the simplest decoding algorithm is to compare each received block with every codeword in the codebook (the "exhaustive search" method), delivering the closest word to the sink. Such a decoder would need to store or generate 2^6 (= 64) twenty-two bit words and to compare each of these with each received block.

(g) The (24,7) and (25,8) Codes

These codes may be decoded most simply as shortened versions of the (26,9) code.

(h) The (26,9) Code

This code has been found by Hashim (1974) by a computer search procedure. No decoding procedure based upon the structure of the code is known and therefore the decoding methods available are limited to those common to all binary linear

block codes. As the code is of low rate the least complex decoder is the exhaustive search decoder. The decoder must store or generate 512 twenty-six bit words and compare each with each received block.

(i) The (28,10) and (29,11) Code

These codes may be decoded most efficiently as shortened versions of the (30,12) code. Although the (29,11) code may be obtained by removing any two parity check bits from the (31,11) B.C.H. code with minimum distance 11, but the writer knows of no way in which a decoder of the (31,11) code may be modified to decode the (29,11) code.

(j) The (30,12) Code

This code is another which was found by Hashim (1974) as a result of a computer search. Again no structure is known which would enable a decoder to be designed which is less complex than the exhaustive search decoder. For this code an exhaustive search decoder needs to store or to generate 2^{12}(= 4096) thirty bit words for comparison with every received word.

3.6 Five-Error Correcting Codes

(a) The (11,1) Code

This code consists of the all-ones and the all-zeros 11-tuples. The decoding algorithm may be to output all zeros if more than five received bits are zero, otherwise output all ones.

(b) The (17,2) Code

Following the method used for the triple-and-quadruple-error correcting codes, this code may be constructed so as to be majority logic decodable. The code then has a generator matrix, G, given by :

$$G = \begin{array}{cccccccccccccccc} k_1 & k_2 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

If the information bits and the parity bits are labelled as shown, it may be seen that

$k_1 = p_2 = p_4 = p_6 = p_8 = p_{10} = p_1 + p_{11} = p_{12} + p_3 = p_{13} + p_5$
$= p_{14} + p_7 = p_{15} + p_9$

$k_2 = p_3 = p_5 = p_7 = p_9 = p_{11} = p_1 + p_{10} = p_{12} + p_2 = p_{13} + p_4$
$= p_{14} + p_6 = p_{15} + p_8$

the decoding circuiting may be arranged in the same manner as for the (11,2) triple-error-correcting code.

(c) The (20,3) Code

This code may be constructed by omitting any one parity check from the (21,3) nom-primitive B.C.H. code with minimum distance twelve. It may therefore be decoded in a similar manner to that proposed for the (20,5) four error correcting code. That is, the "missing" parity check bit may be guessed at the receiver and decoding of the resulting corrupted (21,3) codeword attempted; if decoding is unsuccessful then the guessed bit is inverted so guaranteeing correction provided that no more than five errors occurred in the received word. The (21,3) code is error trap decodable, and so the entire decoding algorithm consists of two error trap attempts; if the first

fails then the missing bit is inverted and then the second attempt is sure to succeed.

(d) The (22,4) Code

This code may be decoded as a shortened (23,5) code.

(e) The (23,5) Code

This code may be constructed by omitting any parity check bit from the (24,5) code of minimum distance 12 found by MacDonald (1958). The code may also be formed as an augmented conjoined code (see Chapter 6). The code is formed by conjoining the (11,4) double-error correcting code with the (12,4) code of minimum distance six and then augmenting an additional codeword from the (11,1) code. The complexity of the decoder is only a little greater than the complexity of decoders for the three constituent codes.

The code may also be decoded by the exhaustive - search method. The thirty two possible codewords would need to be stored or generated, and compared with the received block.

(f) The (25,6) Code

This code may be decoded as a shortened (26,7) code.

(g) The (26,7) Code

This code was found by Hashim (1974) as a result of a computer search. Little is known of its structure, and therefore the only decoding algorithms that may be considered are those common to all linear binary block codes. The code is of low rate, and therefore the exhaustive search decoder is the

obvious choice.

(h) The (27,7), (28,8) (29,9), and (30,10) Codes

These codes may be decoded as shortened versions of the (31,11) code.

(i) The (31,11) Code

This code may be constructed as a primitive B.C.H. code. In this form the code is cyclic, and is decodable by the "error-trapping with windows" method, (see Kasami 1964).

3.7 Six-Error-Correcting Codes

(a) The (13,1) Code

This is the repetition code, with two codewords : the all-zero and the all-one 13-tuples. Decoding is made on a majority principle - if more than six bits of a received block are 'one' then the all-one codeword is delivered to sink, otherwise the all-zero codeword.

(b) The (20,2) Codeword

This code, in common with the (11,2), (14,2), and (17,2) codes already described, may be constructed as a one-step majority-logic decodable code, with a pseudo-cyclic decoding algorithm. The generator matrix is given by :

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

and the orthogonal parity check equations are :

$$k_1 = p_2 = p_4 = p_6 = p_8 = p_{10} = p_{12} = p_1 + p_{13} = p_3 + p_{14}$$
$$= p_5 + p_{15} = p_7 + p_{16} = p_9 + p_{17} = p_{11} + p_{18}$$

$k_2 = p_3 = p_5 = p_7 = p_9 = p_{11} = p_{13} = p_2 + p_{14} = p_4 + p_{15}$
$= p_6 + p_{16} = p_8 + p_{17} = p_{10} + p_{18} = p_{12} + p_1$

The decoding circuitry arrangement follows that for the (11,2) code.

(c) The (24,3) Code

This code may be constructed as a code which is one-step majority-logic decodable in much the same manner as the (17,3) four-error correcting code. Thus, although the code is not cyclic it may be decoded in a cyclic manner.

The generator matrix, G, is given by :

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

if a codeword, C, is written

$$C = k_1, k_2, k_3, p_1, p_2, \ldots p_{20}, p_{21}$$

then the orthogonal parity checks are :

$k_1 = p_4, p_{11}, p_{17}, p_1 + p_5, p_8 + p_{12}, p_3 + p_6, p_{10} + p_{13}, p_2 + p_7,$
$p_a + p_{14}, p_{15} + p_{20}, p_{16} + p_{19}, p_{21} + p_{18}$

$k_2 = p_3, p_{10}, p_{16}, p_1 + p_7, p_8 : p_{14}, p_2 + p_5, p_a + p_{12}, p_4 + p_6,$
$p_{11} + p_{13}, p_{17} + p_{19}, p_{15} + p_{18}, p_{21} + p_{20}$

$k_3 = p_2, p_a, p_{15}, p_1 : p_6, p_8 + p_{13}, p_4 + p_7, p_{11} + p_{14}, p_3 + p_5,$
$p_{10} + p_{12}, p_{16} + p_{18}, p_{17} + p_{20}, p_{21} + p_{19}$

The similarity to the (17,3) code is readily seen, and a decoder will only need circuitry to calculate the additional four orthogonal checks given above for each information bit. This additional circuitry may be arranged to be of a cyclic nature in the same way as for the (17,3) code.

(d) The (26,4) Code

This code may be decoded as a shortened (27,5) code.

(e) The (27,5) Code

This code may be constructed by dropping any parity check bit from the (28,5) code with minimum distance fourteen discovered by MacDonald (1958).

It may also be constructed by conjoining the (14,4) triple-error-correcting code with the (13,4) code obtained by omitting a check bit, and then augmenting with a codeword from the (13,1) repetition code, (see Chapter 6).

The code may be decoded by the method shown in Chapter 6 for conjoined codes, or the exhausive search method may be used.

(f) The (29,6) Code

This code may be constructed as an augmented conjoined code (Chapter 6). The constituent codes are the (15,5) triple-error-correcting code, a (14,5) code with minimum distance six constructed by omitting any one parity check bit from the (15,5) code, and the (13,1) repetition code. As shown in Chapter 6, the code may be decoded with a complexity only slightly greater than that of decoding each of the constituent codes.

3.8 Longer Codes

For lengths much greater than thirty it becomes considerably more difficult to decode optimum codes. It is then more realistic to use near-optimum codes with simpler decoding algorithms. For example the (63,41) sub-optimum triple-error-correcting projective geometry code is generally preferable to the optimum (63,47) code

because it is much more easily decoded. Even so, the projective geometry code requires 343 sets of modulo 2 adders, and 57 seven input majority gates to be decoded by a standard majority-logic decision method; or 27 majority gates, two counters and three binary flags to be decoded by a simpler method, involving "orbits" (L.E. Wright 1977). The (255,218) projective geometry triple-error-correcting code is even more complex to decode - the simplified "orbit" method requires 120 thirty-one input majority gates, three binary flags and a thirty-five entry decoding table, whereas the standard majority logic decision method needs nearly three thousand seven-input majority gates and so is impracticable for most purposes. The long B.C.H. codes are sub-optimum in most cases, but they do have practical decoding algorithms for the correction of small numbers of errors. Chapter 4 discusses the decoding algorithms available for these codes, and it is seen that for codes correcting more than three errors the decoders required are very complex.

In order that long codes with large error correcting power may be easily decoded it is evidently necessary that the codes be of considerable worse rate than the best known. For example the one-step majority logic decodable codes such as the self-orthogonal quasi-cyclic codes discovered by Townsend and Weldon (1967), are very poor in rate compared to even the projective geometry codes - they are, however, much more easily decoded. For the same cost a much longer self-orthogonal-quasi-cyclic, than projective-geometry code of the same rate may be decoded. The increased length code sometimes has better performance than the shorter code (particularly if many more errors may be corrected than guaranteed by the minimum distance) and therefore is preferable in certain applications to the shorter and more newly optimum code.

3.9 The Use of Microprocessors in Decoding Error Correcting Codes

Since the decoders for many error correcting codes require what is in effect a special purpose computer, the microprocessor might seem an ideal component for their realisation.

The versatility of microprocessors is offset, however, by the considerable time required for the execution of each step of computation. It seems reasonable to assume that a programmable system will always be slower than a "hand wired" system.

There are three major ways in which the versatility of microprocessors may be usefully employed. Firstly they may be used in applications where speed is unimportant - for example in paging systems and private-line squelch systems in radio communication systems. Secondly they may be used in parallel to achieve high speed, provided that cost is of secondary importance. Thirdly they may be used to control the operation of sub-sections of the decoder which are fast "hard wired" computation elements.

CHAPTER 4

Decoding B.C.H. Codes

4.1. Introduction

In this Chapter the best method of decoding B.C.H. codes is considered, for various error correction powers of the codes. Decoders for double error correcting B.C.H. codes are first compared, and then their generalisations to more powerful B.C.H. codes are discussed. The decoders described are those considered most suitable for long B.C.H. codes - shorter B.C.H. codes may often be decoded by other more general methods such as error trapping, majority logic decision, and permutation decoding.

4.2. Decoders for Double Error Correcting B.C.H. Codes

(a) Berlekamp's Method

The most common form of decoder for a double-error-correcting (d.e.c.) B.C.H. code is that described by E.R. Berlekamp (1968). This is reviewed briefly here.

The basis of this method lies in the fact that an n-tuple $c(x)$ is a codeword polynomial of an (n,k) d.e.c. B.C.H. code iff x and x^3 are roots of the generator polynomial, $g(x)$, of the code, where x is a primitive n th root of unity in GF (2^m), where $n = 2^m - 1$.

If a codeword is denoted by $c(x)$ and a corrupting error pattern by $e(x)$ then a corrupted codeword, $r(x)$, may be written

$$r(x) = c(x) + e(x)$$

Since $c(x)$ is by definition a multiple of $g(x)$, x and x^3 must be roots of $c(x)$. Therefore, by substitution of x and x^3 in the above equation, the following two equations are obtained :

$r(\alpha) = c(\alpha) + e(\alpha) = e(\alpha)$ and

$r(\alpha^3) = c(\alpha^3) + e(\alpha^3) = e(\alpha^3)$

It is usual to term $r(\alpha^i)$ the i th syndrome, S_i, of the corrupted codeword.

If the error positions are labelled as powers of x, that is if an error in the fifth bit of a codeword is termed $X = x^4$, then the syndromes S_1 and S_3 give two simultaneous equations in the two error positions X_1 and X_2, viz =

$$S_1 = X_1 + X_2 \quad \ldots (i)$$

and $$S_3 = X_1^3 + X_2^3 \quad \ldots (ii)$$

these two equations may be solved for X_1 and X_2 as follows :

from (i); $X_1 = S_1 \quad X_2 \ldots (iii)$

substituting (iii) in (ii); $S_3 = X_2^3 + (S_1 - X_2)^3$

hence, $S_3 = S_1^3 + S_1^2 X_2 + S_1 X_2^2 \ldots (iv)$

similarly, $S_3 = S_1^3 + S_1^2 X_1 + S_1 X_1^2 \ldots (v)$

therefore, if X represents either X_1 or X_2 we have

$S_3 = S_1^3 + S_1^2 X + S_1 X^2 \ldots (vi)$

Decoding may then be accomplished by substituting all possible values for X into equation (vi); those values which are roots of the equation are the error positions the values of which must be inverted in order to complete the decoding.

Note that equation (vi) differs from the equation derived by Berlekamp in that it has not been "divided through" by S_1. As a result the variables to be calculated are S_1, S_1^2, S_1^3, and S_3 compared to S_1, S_1^2, S_3/S_1, and S_3 required by Berlekamp. The importance of this difference is that if S_3/S_1 is found by means of a look-up table, $2^{2(n-k)}$ storage locations are required whereas a look-up table to determine S_1^3 would require only $2^{(n-k)}$ locations. Thus equation (vi) affords a

considerable saving in storage requirements, or may allow a look up table to be used where a time consuming sequential circuit might otherwise have been necessary. Equation (vi) has also been suggested as an improvement by Harari (1974).

It may be shown, for example by Berlekamp (1968), that S_i may be calculated by dividing the received sequence $r(\alpha)$ by M_i, the minimal polynomial of α^i to give $r_i(\alpha)$ and then transferring by a matrix multiplication to $r_i(\alpha^i)$ which is equal to S_i. These operations may be realised using feed-back shift register for the division, and arrays of exclusive or gates for the matrix multiplication (see Peterson and Weldon 1972) and Berlekamp 1968).

In order to solve equation (vi) in practice, the Chien search (Chien 1964) is normally used. In order that the first received bit is the first decoded, it is convenient to arrange for the equation (vi) to be manipulated to give as roots the "inverse locations numbers"; that is, for $X = \alpha^i$ to be a root if the (n - i)th bit is in error; because the first receive bit is the highest order co-efficient of the received polynomial. The equation transformation is very straightforward, with the result (after the simplifying multiplication through by S_1) of $(S_3 + S_1^3)X^2 + S_1^2X + S_1 = 0$. The Chien search then operates as follows .

One feedback shift register is wired to multiply its contents by α with each clock pulse, another is wired to multiply its contents by α^2 with each clock pulse. Initially the first register is loaded with S_1^2 and the second with $S_3 + S_1^3$. The contents of these registers are added together, and added to S_1; the result is the enumeration of $(S_3 + S_1^3)\alpha^2 + S_1^2\alpha + S_1$.

If the result is zero then $X = \alpha$ is a root of equation (vii) and therefore the first received bit is in error. The two feedback shift registers are then shifted once, to give the enumeration of equation (vii) for $X = \alpha^2$ and if the result is zero the second received bit is in error. This clocking procedure continues to check all possible roots of the equation. Note that at each stage it is necessary also to check that $S_1 = 0$ since if $S = 0$ there are no further errors, and no further inversions must take place.

A block diagram of a complete decoder of the type described above is given in Fig. 4.2(a).

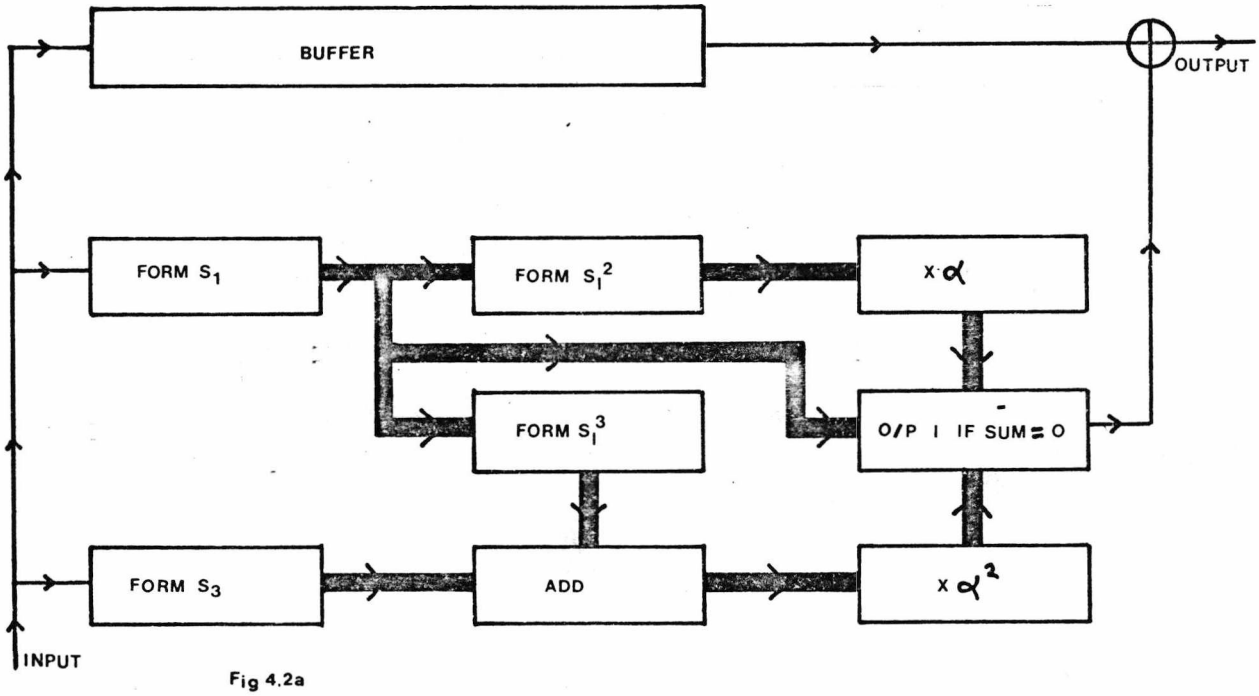


Fig 4.2a

(b) A Simpler Method

Described here is a somewhat less complex decoder for double error correcting B.C.H. codes. To understand its operation consider first the relationship between S_1^3 and S_3 in equation (vi) of the preceding section.

If one error has occurred in the received sequence then $X = 0$ is one root of equation (vi), therefore $S_1^3 = S_3$, also $S_1 \neq 0$ and $S_3 \neq 0$.

If two errors have occurred in the received sequence then since $S_1 = X$ (in fact $S_1 = X_1 + X_2$ from equation (i)) it can be seen from equation (vi) that $S_1^3 \neq S_3$.

Also $S_1 \neq 0$, $S_3 \neq 0$.

If three errors have occurred in the received sequence then the sequence is at distance of least two from any codeword. Therefore, again, $S_1^3 \neq S_3$ and S_1, $S_3 \neq 0$.

If no errors have occurred in the received squence then both S_1 and S_3 equal zero.

Now consider the effect of inverting one bit of a received sequence :

If originally one error had occurred then there now will be either no errors, in which case $S_1^3 = S_3 = 0$, or two errors in which case $S_1^3 \neq S_3$.

If originally two errors had occurred then there will be now either one error, in which case $S_1^3 = S_3$, or three errors in which case $S_1^3 \neq S_3$.

The basic decoding algorithm may now be seen. First, Syndromes S_1 and S_3 are calculated. If $S_1 = S_3 = 0$ then the received sequence is assumed to be error free and is output to the sink unchanged. Otherwise, the first received bit is inverted,

and Syndromes S_1 and S_3 recalculated. S_1 is cubed, and if $S_1^3 = S_3$ the inverted bit is assumed to have been in error and so left inverted; if $S_1^3 \neq S_3$ the bit is assumed to have been correct and is therefore re-inverted. This process is continued by inverting the remaining bits in turn and following the above procedure until all of the errors in the received sequence have been cleared.

Considerable simplification of the above algorithm is possible because B.C.H. codes are cyclic. As a result the syndrome of a cyclic shift of a received sequence may be obtained by shifting once the syndrome of the unshifted sequence in the syndrome calculating feedback register. Furthermore, as with all linear block codes, if a received sequence r, (X) given by $r_1(X) = C(X) + e_1(X)$ is further corrupted by another error pattern $e_2(X)$ to give $r_3(X) = C(X) + e_1(X) + e_2(X)$, then the syndrome of $r_3(X)$ is the sum of the syndromes of $r_1(X)$ and $e_2(X)$. Therefore the syndrome of a received sequence with its first received bit inverted is equal to the sum of the syndrome of the received sequence with the syndrome of a sequence of n - 1 zeros preceded by a 'one', where n is the length of the B.C.H. code.

Combining the above two facts it can be seen that if the syndrome of a received sequence is calculated then the syndrome of that sequence with its first bit inverted may be obtained by adding the syndrome of 'one followed by all zeros' to the calculated syndrome, and the syndromes of the sequence with other bits invested are found simply by shifting the 'uninverted' syndrome in its register and still adding the syndrome of the 'one followed by all zeros'.

Example

Consider the (15,7) double error correcting B.C.H. code. The syndrome S_1 of a one followed by fourteen zeros is

1 0 0 1 and so a circuit which calculates the syndrome of a received sequence inverted in its first, second, and so on to its fifteenth bit position, and gives a 'one' output if the result is zero is given in Fig. 4.2.(b)

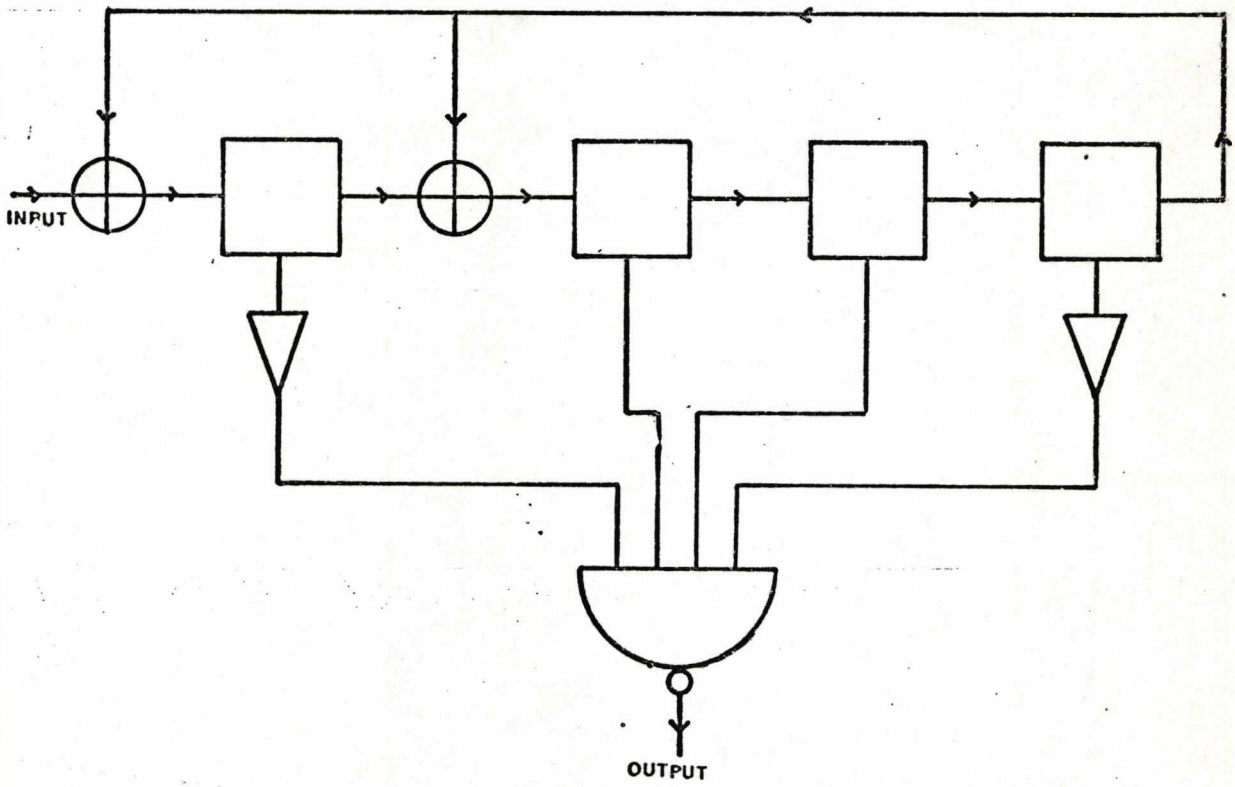


Fig. 4.2.(b).

A block diagram is given in Fig. 4.2.(c) of a decoder, for a double error correcting B.C.H. code, using this algorithm. Note that in this decoder a 'NAND' gate is used to test S_1 for 'all zero' - if S_1 is zero then, provided not more than two errors have occurred in the received sequence, so will S_3 be zero. If no errors have occurred in the received sequence then the test $S_1^3 = S_3$ will be positive after an inversion has been made in the syndromes. Therefore by adding the output of the 'NAND' gate modulo 2 to the output of the $S_1^3 = S_3$ test circuit it is ensured that the received sequence is delivered to the sink unaltered, when no errors have occurred during transmission over the channel. The circuitry operates as follows : Firstly n received bits are clocked simultaneously into the buffer register and the two syndrome forming feedback shift registers. Thus syndromes S_1 and S_3 are formed. The remaining circuitry then determines whether or not the first received bit is correct, and the corrected bit is delivered to sink on the next clock pulse. This clock pulse also shifts once the syndrome shift registers, enabling the correctness of the second received bit to be determined. This process is continued until all of the n received bits have been delivered to sink, and then a new received block is shifted into the decoder. The process is then repeated as above. Note that, as with all Meggitt type decoders, the decoder must be clocked at twice line rate in order to prevent any buffer overflow.

The advantage of this circuit over that of the decoder of section (a) is clearly that of obviating the need to calculate S_2, and that the Chien Search Stage is effectively performed by the syndrome forming registers, thus reducing hardware costs.

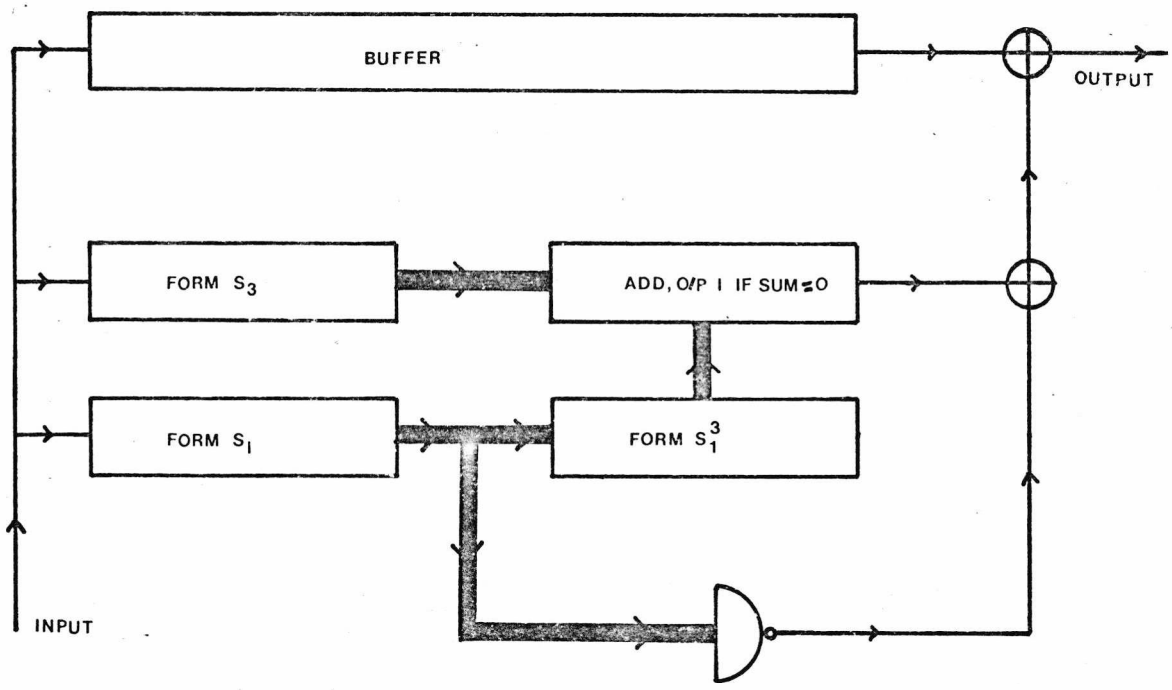


Fig 4.2c

The decoding scheme presented above may be seen to be equivalent to a special case of the step-by-step decoding algorithm for B.C.H. codes given by Massey (1965), and this generalisation of the decoder will be described in the next section, as it is a very suitable method of decoding B.C.H. codes of minimum distance seven.

4.3. Decoders for the Correction of Multiple Errors

B.C.H. codes capable of correcting more than two errors are considerably more complex to decode than are the d.e.c. versions. For moderate error correcting power, the simplest decoding technique is that of Massey (1965), but for larger t methods put forward by Peterson (1960) and Berlekamp (1965, 1968) are preferable. These algorithms are described here to illustrate the variations in their complexity for differing error correcting powers.

(a) Massey's Step-by-Step Algorithm

This decoding algorithm for B.C.H. codes is based upon two theorems.

The first is that the determinant of the matrix

$$L^t = \begin{bmatrix} S_1 & 1 & 0 & 0 \ldots 0 \\ S_3 & S_2 & S_1 & 1 \ldots 0 \\ S_5 & S_4 & S_3 & S_2 \ldots 0 \\ & & \vdots & \\ S_{2t-1} & S_{2t-2} & S_{2t-3} & S_{2t-4} \ldots S_t \end{bmatrix}$$

is zero if the weight of the error pattern, e, affecting a codeword is t - 1 or less, and non-zero if e is of weight t or t + 1; provided the B.C.H. code is t error-correcting.

The second theorem is that if a syndrome bit is inverted, the syndrome becomes that of the original sequence modified by inverting one of the parity check bits.

The decoding algorithm based on these two theorems begins with the following steps :

(1) The syndromes S_1, S_2, ... S_{2t-1} are formed

(2) The determinant of Lt is calculated

(3) If det (Lt) = 0 then successive bits of the syndromes are inverted until det (Lt) $\neq$ 0.

Step (3) ensures that, at the following step, the received sequence contains exactly t errors (provided that the original received sequence does not contain more than t errors). By the first theorem, if det (Lt) = 0 then (t - 1) or less errors have occurred; and so, by the second theorem, the inverting of successive bits of the syndromes ensures that when det (Lt) first becomes non-zero, there are exactly t errors in the modified sequence.

The decoding algorithm continues with the following steps :

(4) The syndromes of the sequence V_o, where V_o is a sequence of (n - 1) zeros preceded by a single 'one', are added to the syndromes modified by step (3). The results are used as syndromes in step (5).

(5) Det (Lt) is again calculated.

(6) If det (Lt) is now zero, the first received bit is inverted and delivered to the sink, otherwise the first received bit is delivered to the sink unchanged.

Step (4) gives the syndromes of the modified sequence modified a second time by inverting the first received bit (which is the first information bit). Therefore, if the first bit was in error, the result of step five will be zero since

the syndromes will be of a sequence with only t - 1 errors. If, however, the first bit was not in error, the syndromes will be of a sequence containing t + 1 errors, and therefore by the first theorem the value of det (Lt) will be non-zero. This explains the step (6) which is the error correcting step. The algorithm is concluded by :

(7) The syndrome forming registers are shifted once, and steps 4 and 5 repeated, step (6) is then carried out, but on the second received bit.

(8) The process continues for the third, fourth, ... n th received bits.

Steps (6), (7), and (8) are possible because of the cyclic nature of B.C.H. codes, thus the syndromes achieved by step (7) are those of the modified sequence shifted by one bit, hence the effect of the second execution of step (4) is to calculate the syndrome of the modified sequence with the second bit inverted. Continuing the process corrects all n received bits in order.

Calculation of the S_i , $0 < i < 2t$ is straight forward, the most complex operation in the algorithm is that of calculating det (Lt). This calculation is carried out at least n times, and at most n + 2t times. It is the complexity of this stage which limits the highest value of t for which the algorithm is useful. The terms of det (Lt) are given in the table below, for $t \leq 5$, from which can be seen the complexity of the calculations required for increasing t.

t	det (Lt)
2	$S_1^3 + S_3$
3	$S_1^6 + S_1^3 S_3 + S_1 S_5 + S_3^2$
4	$S_1^{10} + S_1^7 S_3 + S_1^5 S_5 + S_1^3 S_7 + S_1^2 S_3 S_5 + S_1 S_3^3 + S_3 S_7 + S_5^2$
5	$S_1^{15} + S_1^{12} S_3 + S_1^8 S_7 + S_1^7 S_3 S_5 + S_1^6 S_3^3 + S_1^6 S_9 + S_1^5 S_5^2 S_1^5 S_3 S_7 + S_1^4 S_3^2 S_5 + S_1^3 S_5 S_7 + S_1^3 S_3 S_9 + S_1^2 S_3 S_5^2 + S_1^2 S_3^2 S_7 + S_1 S_5 S_9 + S_1 S_7^2 + S_3^2 S_9 + S_3^5 + S_5^3$

The calculation of det (L_2) is quite simple - the only complex operation required is the cubing of S_1. This case $(t = 2)$ is equivalent to the double-error-correcting decoder given in Section 4.2.(b), except that in that section circuitry has been included to make the steps (2) and (3) of the Massey algorithm unnecessary. This circutry is much simpler than the additional inverting and clocking circutry otherwise required to carry out those additional steps, and the decoder of Section 4.2.(b) becomes a Meggitt type decoder with very modest clock circutry requirements.

The calculation of det(L_3), required for the decoding of triple error correcting B.C.H. codes, is somewhat more complex. To evaluate det (L_3) the terms $S_1^6 + S_1^3 S_3 + S_1 S_5 + S_3^2$ must be calculated. S_3^2 may be found from S_3 using a matrix of exclusive or gates as described, for example, by Berlekamp (1968). S_1^3 must be calculated either by means of a look up table or by a combinatorial circuit. S_1^6 may then be found, again using a matrix of exclusive or gates, from S_1^3 since $S_1^6 = (S_1^3)^2$. The

terms $S_1^3 S_3$ and $S_1 S_5$ must be calculated by a combinational logic multiplier circuit, or by a look-up table. Clearly, by using a sequential programme, only one squaring, one cubing, and one multiplying circuit need be used for the calculation in this case, indeed, by calculating S_1^3 as $S_1^2 S_1$, the cubing circuitry could also be omitted. It is important to note, however, that the calculation of det (L_3) must be made as many as $(n + 6)$ times per decoded block. Therefore the time taken for the calculation is very important, and it is for this reason that sequential circuitry for the multimplication of syndromes is not considered.

To apply the algorithm to B.C.H. codes capable of correcting four errors would clearly require some considerable amount of circuitry, since seven multiplications, and three raising-to-power operations are required in this case. Certainly, for codes correcting five or more errors, the decoding circuitry will be excessively complex unless a sequential programme is used for the determinant evaluation - which would only be possible if the time available for decoding is considerable.

The choice between combinational circuitry and look-up tables for the calculation of powers or products of syndromes requires some consideration. A look-up table capable of giving the product of syndromes requires considerably more storage than one giving an odd power of a syndrome. For example, a code of length 256 would require $2^8 = 256$ storage locations for the calculation of S_1^3, but $2^{16} = 65,536$ storage locations for the calculation of the product of two syndromes. Therefore, with present technology, the product of two syndromes is best found using a combinatorial logic circuit. In

Appendix 'A' a method of constructing suitable logic circuits first given by Bennett & Stein (1963) is described, and bounds on the number of gates required for their fabrication are derived. Also given are the actual numbers of gates required for multipliers of syndromes of certain selected lengths.

(b) Error Location Polynomial Decoding

The method described earlier for the correction of double error correcting B.C.H. codes by the solution of an algebraic equation may also be applied to multi-error correcting B.C.H. codes (Berlekamp 1968). That is, a polynomial (the "error location polynomial") with algebraic combinations of partial syndromes as co-efficients may be formed which has as roots the locations of errors expressed as elements of $GF(2^m)$ where the code has length $2^m - 1$. Such a method of decoding divides naturally into three parts; information of the partial syndromes, formation of the co-efficients of the error location equation, and solution of the polynomial.

A partial syndrome S_i is formed as in the double-error correcting case by dividing the received polynomial by M_i, the minimal polynomial of α^i and multiplying the remainder, $r_i(\alpha)$, by a binary matrix to give $r_i(\alpha^i) = S_i$. For a t error correcting B.C.H. code, S_i may be found for $0 < i \leqslant 2t - 1$.

The error location polynomial is more conveniently arranged to have the inverses of the error locations as roots, so that in a practical realisation of the decoder the first received bit may be the first location to be tested as a root of the polynomial, and so the decoding delay is minimised.

The inverse error location numbers may be defined as B_j, $0 < j \leqslant t$. Then it is known (Berlekamp 1968) that the B_j and the S_i are related by

$$S_i = \sum_{j=1}^{t} B_j^{\,i}$$

and the error location polynomial is defined as :

$$(PX) = \prod_{j=1}^{t} (1 + B_j X) = R_o + R_1 X + R_2 X^2 + \ldots . R_t X^t$$

from which it is possible to define the R_i in terms of the S_i. The most efficient means of achieving this relationship is the "Berlekamp Iterative algorithm" (Berlekamp 1968). The algorithm may be visualised (Lin 1970) as the filling in of a table (see Table 4.1.) for which the first two rows are always as shown. To complete the table, for the (M + 1)th row :

(a) If dM = 0 then $P^{(M+1)}(X) = P^{(M)}(X)$

(b) If dM = 0 find a row preceding the Mth, say the Vth, where 2V - jM (in the last column) is as large as possible with dV non-zero.

Then $P^{(M+1)}(X) = P^{M}(X) - dM\, dV^{-1} X^{2(M-V)} P^{V}(X)$

In both cases j(M + 1) is equal to the degree of $P^{(M+1)}(X)$

and $d(M+1) = S_{2M+3} + P_1^{(M+1)} S_{2M+2} + P_2^{M+1} S_{2M+1} + \ldots . + P_{j(M+1)}^{M+1} S_{2M+3} - j(M+1)$

then P(X) is given by $P^{(t)}(X)$.

The roots of P(X) may be found in a 'brute force' way by substitution of each of the n possible inverse error location numbers (i.e. non-zero members of $GF(2^m)$ where $n = 2^m - 1$) into the error location polynomial and testing for a zero result. This search, known as the Chien search and described earlier for double error correcting codes, is easily realised as hardware

in the form of feedback shift registers (Chien 1963).

The requirement for inversion in Berlekamp's iterative algorithm is eliminated by a modification of the algorithm, given by Burton (1971). The simplification of the decoder is minor, however.

The Berlekamp algorithm is seen to be very complex in every way. Of most importance is the complexity of any hardware realisation of such an algorithm. This complexity dictates that its use is only advantageous over that of the Massey algorithm for values of t of 4 or above.

Peterson (1960) put forward a decoding method based upon error location equations in a similar way to Berlekamp. The solution of the equations was achieved by the inversion of a matrix of syndromes. The method, described in detail by Peterson and Weldon (1972) and compared to the Berlekamp method by Berlekamp (1968) is more complex than the Berlekamp method for large t, but for t less than about five may be preferable. It is nevertheless more complex than the step-by-step method for t less than four.

4.4 Applications of B.C.H. Decoding Algorithms

B.C.H. decoding algorithms have importance outside the realm of decoding B.C.H. codes. It has been shown by Chien and Choy (1975) that Goppa codes and Srivastava codes may be decoded by a B.C.H. decoder; C.T. Retter (1975) has also published methods of decoding Goppa codes with B.C.H. decoders. More recently Helgert (1977) has shown that, by a linear transformation of the syndromes, Alternant codes may be decoded by the Berlekamp B.C.H. decoding algorithm. The Alternant codes contain B.C.H., Goppa, and Srivastava

codes as sub classes.

A major advance in the decoding of B.C.H. codes would result from an ability to calculate quickly the logarithms of elements of Galois fields. At present the most efficient method of realising B.C.H. decoders of any type would appear to be microprocessor control of PROM type Galois field arithmetic units.

CHAPTER 5

Hashim's Nested Codes

5.1. Introduction

Presented here is a description of the Hashim nested codes, which differs from that given by Hashim (1974).

This description gives a clearer insight into the properties of nested codes, and the decoding alogorithm applicable to nested codes is readily deduced from the description.

The conditions under which a nested code will decode a codeword corrupted by more than $(d - 1)/2$ errors, where d is the minimum distance of the code, are noted, and the consequences of repeated nesting of the codes are considered, especially the increase in length, rate, and decoder complexity which results.

Further, a decoder algorithm put forward by Hashim for low rate nested codes, and for nested codes formed from low rate nested codes, is investigated.

5.2. Structure of the Nested Codes

The parity check matrix, H, of a nested code is given by Hashim as Fig. 5.1.

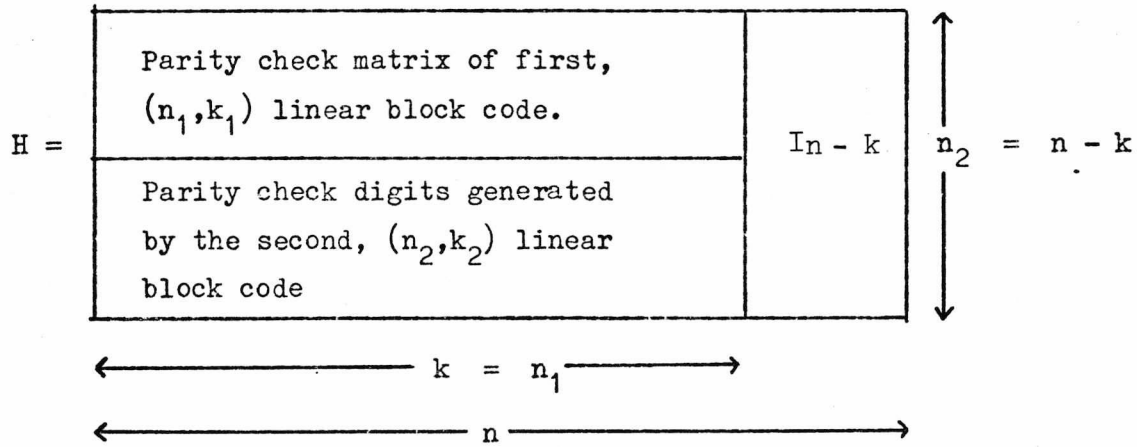


Fig 5.1 - Parity Check Matrix of a Nested Code

The first k columns of the parity check matrix of this (n,k) nested code are therefore codewords of the second, (n_2,k_2) code which is either a Hamming code with minimum distance 4, or another nested code.

The (n_1,k_1) code has an odd minimum distance d, and then the (n_2,k_2) code must have minimum distance (d - 1). The resulting (n,k) nested code then has minimum distance d, with $n = n_1 + n_2$ and $k = n_1$.

5.3. Proof of the Minimum Distance of a Nested Code

Consider first a code C_A with a generator matrix G_A given by :

$$G_A = \left[I \mid H_1^T \right]$$

where H_1^T is the transpose of the parity check matrix of another code C_1.

It will be proved that if the code C_1 has minimum distance d then code C_A will correct up to (d - 1)/2 errors provided that they occur only in the information digits of the code.

Proof

Let a codeword, C_A, from the code C_A, be corrupted by an error pattern e_A. Let $e_A = e_k \mid e_r$, where e_k is the error pattern in the information digits and e_r that in the redundancy digits of the corrupted codeword.

Then the syndrome, S_A, of this corrupted codeword is given by :

$$S_A = (C_A + e_A) H_A^T$$

where H_A is the parity check matrix of the code C_A.

hence $S_A = C_A H_A^T + e_A H_A^T = e_A H_A^T$

now $e_A = e_k \mid e_r$

and since $G_A = \left[I \mid H_1^T \right]$

then $H_A = \left[H_1 \mid I \right]$

hence $e_A H_A^T$ is given by

$$e_A H_A^T = e_k \mid e_r \left[\frac{H_1^T}{I} \right]$$

Now, the row vector e_k has length equal to the column length of H_1^T, and $e_r = 0$ since all errors are in the information digit section of the codeword.

Therefore $s_A = e_A H_A^T = e_k H_1^T$, provided all errors are in the information digits of c_A.

If the weight, $w(e_k)$ of e_k is bounded by $w(e_k) \leqslant (d - 1)/2$ then the solution of the equation $s_A = e_k H_1^T$ is precisely the decoding of a codeword from C_1 which is corrupted by at most $(d - 1)/2$ errors. That is to say, if s_A is treated as a syndrome of a corrupted codeword from C_1 and an error pattern is found from this by the decoding rules of C_1, then the error pattern so found is precisely e_k, provided that the weight of e_k is at most the error correcting power of C_1, i.e. provided $w(e_k) \leqslant (d - 1)/2$.

This completes the proof that C_A will correct up to $t = (d - 1)/2$ errors provided that they all occur in the information digits of the codewords. The manner in which this is achieved is seen from the proof to be to use s_A as a syndrome of the code C_1, and the error pattern corresponding to this syndrome, found by the decoding algorithm of C_1, is the error pattern e_k, corrupting the information digits.

In order to convert the code C_A into one which will correct $(d - 1)/2$ errors wherever they occur in the codeword, the parity check digits may be encoded with another linear systematic block code, C_2, which has even minimum distance $(d - 1)$. As a result of this encoding, the parity

check digits of the converted codewords are themselves codewords from code C_2. Hence if at most $t - 1 = \frac{d - 2}{2}$ errors occur in the parity check digits they are correctable - allowing the decoding algorithm previously described to clear the remaining errors in the information digits - whereas if t errors occur in the parity check digits then they are detected and the information digits may be assumed to be uncorrupted, since it is accepted that t errors at most have occurred overall.

In order to ensure that the parity check digits of codewords from C_A are encoded according to the rules of code C_2, it is sufficient that the rows of H_1^T, in the generator matrix G_A, are encoded by the (n_2,k_2) code C_2.

The generator matrix, G, of the resulting (n,k) code C is then given by :

$$G = \left[\begin{array}{c|c|c} I & H_1^T & \begin{array}{l}\text{digits obtained by encoding rows of} \\ H_1^T \text{ with code } C_2\end{array} \end{array}\right]$$

The last n_2 digits of each row are then codewords from code C_2. The codewords from C are of course simply all the possible combinations of sums of rows of G - therefore the last n_2 bits of all the codewords of C are always the sum of some codewords from C_2 and since C_2 is linear they are always a codeword from C_2.

Finally, by manipulating G, the parity check matrix, H, of the code C is seen to be given by :

$H = 1$

H_1	
Digits obtained by encoding columns of H_1 with the code C_2	I

where H_1 is the parity check matrix of code C_1 which has a minimum distance one greater than that of the even minimum distance code C_2.

This, of course, describes precisely a Hashim nested code. The distance properties of the code have therefore been proved as required.

5.4. The Decoding Algorithm

Directly from the proof has come the decoding algorithm, which is to first decode the parity check bits of a received codeword, according to the decoding algorithm of C_2 (if t errors are detected here then the information digits are considered error-free) and then to correct errors in the information digits by treating the first $(n_1 - k_1)$ digits of the syndrome of the received word as the syndrome of a corrupted codeword from C_1. The error pattern corresponding to this syndrome, found by a decoder for the code C_1, is the error pattern in the information digits of the codeword from the code C.

5.5. The Optimality of the Construction of Code C_A

The efficiency of the code C can be seen to be dependent upon that of the code C_A. It will be shown that the construction of G_A as :

$$G_A = \left[I \mid H_1^T \right]$$

is the optimum construction for a code capable of correcting error patterns occurring only in the information positions of a codeword, provided that H_1 is the parity check matrix of an optimum random error

correcting code.

The syndrome S_A, when treated as a syndrome of a corrupted codeword from code C_1, has already been shown to correspond to the error pattern in the information digits of a corrupted codeword, as long as this error pattern has Hamming weight at most $(d - 1)/2$ where d is the minimum distance of the code C_1.

It may further be seen that any error pattern corresponding to a coset leader of the code C_1 may be corrected provided the syndrome S_A is used by a maximum likelihood decode for code C_1. That is to say, those error patterns which are correctable by code C_1 are also correctable by code C_A if they occur only in the information digits of the codewords. Thus if the code C_1 is optimum, that is corrects the optimum set of error patterns for use on a Gaussian Channel, then code C_A is also optimum in that it can correct the optimum set of error patterns occurring in the information digits, when used on a random channel.

Hence it has been shown that the code G_A is constructed in the most efficient manner possible for use in a random error correcting nested code, and so nested codes cannot be made more efficient by the choice of a different construction of G_A.

5.5.ii Correction of Error Patterns of Weight $> (d - 1)/2$

The ability of nested codes to correct error patterns of weight in excess of that guaranteed by their minimum distances may be seen immediately from the decoding algorithm. A received codeword is correctly decoded provided that the following conditions are met :

(a) All errors are cleared from the parity check digits by code C_2.

(b) All errors are then cleared from the information digits by code C_A.

Alternatively, a received word is also corrected if the information

digits are error-free and the code C_2 detects any remaining errors.

Therefore a codeword is correctly decoded provided these are at most $d = (d - 1)/2$ errors in the information bits and at most $t - 1$ errors in the parity check bits; and it is also correctly decoded when the information digits are error-free and more than $t - 1$ errors occur in the parity check digits, but are nevertheless detected by the decoder for code C_2.

From the previous section it can be seen that the former requirement reduces to the need for the errors in the information digits to be correctable by code C_1, and those in the parity check digits to be correctably by code C_2.

5.6. Easily Decodable Nested Codes

In order for nested codes to be useful, it is necessary that the be easily decodable. Since the complexity of decoding a nested code is approximately equal to the sum of the complexities of the decoders for the constituent codes, it is therefore necessary that the codes C_1 and C_2 be easily decodable. For this to be so, C_1 and C_2 should be either nested codes themselves, or easily decodable in some other way, for example majority-logic decodable or error trappable.

As an example, consider a nested code constructed from a (23,12) linear binary block code with minimum distance seven, (i.e. the Golay perfect three-error-correcting code) and a (21,11) linear binary block code with minimum distance six (e.g. a first-order projective geometry code). This code will be a (44,23) nested code with minimum distance seven, and may be easily decoded since the (23,12) code may be decoded by the "error trapping with windows" method (Lucky, Salz, and Weldon 1968), and the (21,11) code is one-step majority-logic decodable (Peterson 1972).

Hashim (1974) implies that for a nested code of rate at most one half it is necessary only that C_2 be easily decodable, and that the information bits of the code may be found by solving the parity check equations of the nested code. Hashim thus introduces a class of "decodable nested codes" found by nesting such codes with other nested codes C_2 and then repeating this nesting procedure as often as desired. It will be shown later that this decoding algorithm is fallacious. First, more general properties of nested codes are examined.

5.7. The Rate of Nested Codes

Hashim correctly states that the decoding complexity of nested codes increases linearly with their length. This criterion, however, may give misleadingly optimistic view of the power of nested codes. Consider for example the (44,23) nested code which was described above. This code has length nearly twice that of the (23,12) constituent code, whilst the decoding complexity is also somewhat less than twice the complexity of decoding the (23,12) code (to be more precise, approximately the sum of the complexities of the decoders of the (23,12) code and the (21,11) code). The rate of the new code, however, is 0.523 compared to the (23,12) code which has rate 0.522 and the same minimum distance as the new code. Evidently, the (44,23) code, although having increased in decoding complexity linearly with the increase in length from a (23,12) code, has increased negligibly in rate.

In view of this doubt as to the usefulness of nested codes, a calculation is now made of the efficiency, R, of a nested code in terms of the efficiencies R_1, R_2, of the constituent codes C_1 and C_2 respectively.

It is known, from the construction of the codes, that the following three relations hold between the parameters of the nested code and its

constituent codes :

$$n = n_1 + n_2 \quad .. \quad .. \quad .. \text{ (i)}$$

$$k = n_1 \quad .. \quad .. \quad .. \text{ (ii)}$$

$$k_2 = n_1 - k_1 \quad .. \quad .. \quad .. \text{ (iii)}$$

From equations (i) and (ii)

$$n = k + n_2 \quad .. \quad .. \quad .. \text{ (iv)}$$

and from (ii) and (iii)

$$k_2 = k - k_1$$

$$\therefore \frac{k_2 n_2}{n_2} = k - \frac{k k_1}{n_1} \text{ since } k = n_1$$

$$\therefore n_2 = \frac{k}{R_2} (1 - R_1) \quad .. \text{ (v)}$$

since $R_1 = \frac{k_1}{n_1}$ and $R_2 = \frac{k_2}{n_2}$

hence, from (iv) and (v)

$$n = k + \frac{k}{R_2} (1 - R_1)$$

$$\therefore \frac{n}{k} = 1 + \frac{1 - R_1}{R_2}$$

$$\therefore R = \frac{R_2}{1 - R_1 + R_2} \quad \text{since } R = \frac{k}{n}$$

This gives the required relationship between R, R_1 and R_2.

In order that a nested code be useful, it is necessary that it have a greater rate than that of the constituent code C_1 which has the same minimum distance as itself.

This gives the condition :

$$\frac{R_2}{1 - R_1 + R_2} > R_1$$

i.e. $R_2 > R_1 (1 - R_1) + R_1 R_2$

$$\therefore R_2 (1 - R_1) > R_1 (1 - R_1)$$

hence, since $1 - R_1 > 0$,

$$\underline{R_2 > R_1}$$

For nested codes with high rates C_2 will be much shorter than C_1, and this consideration will overide that of minimum distance, making C_1 higher in rate than C_2. In this situation it is obviously pointless to construct a nested code, since the code C_1 will then have a higher rate than, and the same minimum distance as, the nested code and yet be shorter. Hence C_1 would give a lower probability of error than the nested code, on a random error channel, unless the nested code is able to construct a considerable number of errors of weight greater than its guaranteed error correction capability. Also, C_1 would be much easier to decode than the nested code constructed from it.

At low rate, however, when code C_2 will be the same length as, or longer than, code C_1 the situation is different. Since code C_2 has a minimum distance one less than that of C_1 its rate will be higher than that of C_1 and so the nested code will also have a rate higher than that of C_1.

Nested codes which are constructed with a C_1 which is the best known linear binary block code with minimum distance d and C_2 another nested code with minimum distance d - 2 extended by the addition of an overall parity check bit to give it minimum distance d - 1, are listed by Hashim for minimum distances five to eleven.

The longest of these for which nesting gives an improvement in rate over that of the code C_1 are listed below in Table 5.1.

TABLE 5.1

Minimum Distance	Longest useful nested code when C_1 is best known	R
5	37,23	0.62
7	38,15	0.39
9	73,30	0.41
11	91,31	0.34

A similar table has been compiled (Table 5.2.) for nested codes constructed from a code C_1 which is the best known linear binary block code with minimum distance d and a code C_2 which is the best known binary group code with minimum distance d - 1.

It can be seen that in either case the longest nested codes that may be constructed are at best of rate slightly greater than one half.

TABLE 5.2.

Minimum Distance	Longest useful nested code with C_1 & C_2 the best known	R
5	37,23	0.62
7	60,34	0.57
9	89,47	0.53
11	95,45	0.47

As stated above Hashim claims that with nested codes of rate one half or less it is not necessary to decode C_1 - the information digits may be calculated by clearing the errors from the parity check digits by using code C_2 and then solving the parity check equations of code C_1. This process may be less complex than decoding a corrupted code-word from C_1, in which case nested codes of rate one half or less would certainly be useful in that they would have a rate higher than that of C_1 and may be easily decoded.

Also useful are nested codes in which C_1 is a nested code of rate one half or less, or in which C_1 is a nested code that in turn is constructed from a nested code, and where in either case code C_2 is also a similarly constructed nested code.

These easily decodable repeatedly nested codes may be constructed with high rates and for any desired odd minimum distance. They are, however, very long compared to other classes of error correcting codes, particularly for minimum distances greater than five and at high rates.

This is shown graphically in Fig. 5.2. where the decodable nested codes with minimum distances 5, 7, and 9 are compared with self-orthogonal quasi-syclic codes, and with B.C.H. codes, of the same minimum distances. The parameters of the nested codes are taken from a list given by Hashim (1974) which has been corrected for some small arithmetic errors. This corrected list is given in Table 5.3.

5.8. Decoding Nested Codes with rate less than or equal to half

Hashim's claim that nested codes with rate at most one half may be decoded by clearing the errors from the (n - k) parity check digits and then finding the error pattern in the information digits by solving k parity check equations is now investigated.

Hashim (1974) states that since the (n - k) rows of the parity check matrix H of a nested code are linearly independent, then the (n - k) syndrome equations are also linearly independent and therefore the decoding scheme above may be used. In fact, however, it is necessary that the rows formed from k of the first (n - k) columns of the H matrix be linearly independent; and this property is not held by the parity check matrix of a nested code, as will now be shown.

Consider firstly the parity check matrix, H, of a half rate nested code, which may be written as :

$$H = [M \mid I]$$

Then the equations obtained for the calculation of the error pattern in the information digits are given by the solution of the equation :

$$eH^T = Me = s$$

where e is a column vector representing the error pattern, and s a row vector which is the syndrome given by multiplying the received, corrupted, codeword by H^T.

Then e is given by the solution of the equation :

$$e = sM^{-1}$$

and is uniquely soluble iff M is a non-singular matrix.

For M to be non-singular, the rows of M must be linearly independent. In the case of nested codes, the matrix M is given by

$$M = \left[\begin{array}{c} H_1 \\ \hline \text{Columns obtained by encoding} \\ \text{columns of } H_1 \text{ according to code } C_2 \end{array}\right]$$

and therefore the lower $(n_2 - k_2)$ rows of M are linear combinations of the upper rows.

Hence the matrix M must be singular, and e cannot be uniquely determined - hence the decoding procedure is not valid for codes of rate one half.

Furthermore, in nested codes of rate less than one half, the matrix M must still contain $(n_2 - k_2)$ rows which are linearly dependent on the remainder, since the k columns must still be chosen from linearly dependent rows of the H matrix.

Therefore, whatever the rate of a nested code, the decoding algorithm put forward by Hashim is not applicable. This may be illustrated by a simple example.

A (13,5) nested code with minimum distance five may be constructed with code C_1 a 5,1 repetition code and code C_2 an (8,4) Hamming code with minimum distance 4.

Thus code C_1 has the parity check matrix H_1 given by :

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

and by encoding the columns of H_1 according to the rules of code C_2 and adjoining the result to an identity matrix of order eight, the parity check matrix H_1 of the nested code is constructed.

Now the generator matrix, G_2, of the (8,4) code C_2 is given by :

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

and so H is given by :

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

Thus G, the generater matrix of the nested code, is given by :

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

In Table 5.4. is given a list of all the codewords in this code, i.e. the row space of G, from which it can be seen that any parity check vector will always correspond to two different information vectors - hence an error pattern in the information vector cannot be calculated solely from a knowledge of the uncorrupted parity check bits.

Information	Parity Checks	Information	Parity Checks
0 0 0 0 0	0 0 0 0 0 0 0 0 0	1 0 0 0 0	1 1 1 1 1 1 1 1 1
0 0 0 0 1	0 0 0 1 1 1 1 1 0	1 0 0 0 1	1 1 1 0 0 0 0 1
0 0 0 1 0	0 0 1 0 1 1 1 0 1	1 0 0 1 0	1 1 0 1 0 0 1 0
0 0 0 1 1	0 0 1 1 0 0 1 1	1 0 0 1 1	1 1 0 0 1 1 0 0
0 0 1 0 0	0 1 0 0 1 0 1 1	1 0 1 0 0	1 0 1 1 0 1 0 0
0 0 1 0 1	0 1 0 1 0 1 0 1	1 0 1 0 1	1 0 1 0 1 0 1 0
0 0 1 1 0	0 1 1 0 0 1 1 0	1 0 1 1 0	1 0 0 1 1 0 0 1
0 0 1 1 1	0 1 1 1 1 0 0 0	1 0 1 1 1	1 0 0 0 0 1 1 1
0 1 0 0 0	1 0 0 0 0 1 1 1	1 1 0 0 0	0 1 1 1 1 0 0 0
0 1 0 0 1	1 0 0 1 1 0 0 1	1 1 0 0 1	0 1 1 0 0 1 1 0
0 1 0 1 0	1 0 1 0 1 0 1 0	1 1 0 1 0	0 1 0 1 0 1 0 1
0 1 0 1 1	1 0 1 1 0 1 0 0	1 1 0 1 1	0 1 0 0 1 0 1 1
0 1 1 0 0	1 1 0 0 1 1 0 0	1 1 1 0 0	0 0 1 1 0 0 1 1
0 1 1 0 1	1 1 0 1 0 0 1 0	1 1 1 0 1	0 0 1 0 1 1 0 1
0 1 1 1 0	1 1 1 0 0 0 0 1	1 1 1 1 0	0 0 0 1 1 1 1 0
0 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1	0 0 0 0 0 0 0 0

TABLE 5.4. - Codewords from the (13,5) Nested Code

5.9. Useful Decodable Nested Codes

It would seem that in order for a nested code to be both useful and easily decodable, the constituent codes C_1 and C_2 should both be easily decodable, and of course the rate of code C_2 should be considerably higher than that of C_1

This latter requirement would normally imply that the nested code be of low rate, as has been shown earlier. If the code C_1 is a nested code, however, then the rate may be increased. In such cases, unfortunately, the resultant nested codes will become extremely long for only moderate to high rates and small error correction ability.

This latter requirement would normally imply that the nested code be of low rate, as has been shown earlier. If the code C_1 is a nested code, however, then the rate may be increased. In such cases, unfortunately, the resultant nested codes will become extremely long for only moderate to high rates and small error correction ability.

An example of a useful nested code would be the (35,15) triple error correcting code constructed from the (15,5) triple error correcting B.C.H. code, which may be error trap decoded, and the (20,10) code of minimum distance six obtained by shortening the Euclidean geometry one-step majority logic decodable (21,11) code.

This (35,15) nested code may then be nested with the (31,20) code of minimum sistance six, obtained by shortening the (31,21) double error correcting B.C.H. code and adding an overall parity check bit, to give a (66,35) triple error correcting code, which has a rate greater than one half.

Whatever codes are used, when they are repeatedly nested the resulting codes will tend to become very long compared to other known classes of easily decodable codes such as the self-orthogonal quasi-cyclic (SOQC) codes and therefore they will be proportionately difficult to decode.

5.10. Conclusions

The structure and properties of nested codes have been described in a clear and easy to understand manner. The codes have been seen to be interesting, but evidently of low rate. When of higher rate they are of lower efficiency than their constituent codes, hence being of no practical value, or are considerably longer than other easily decodable codes.

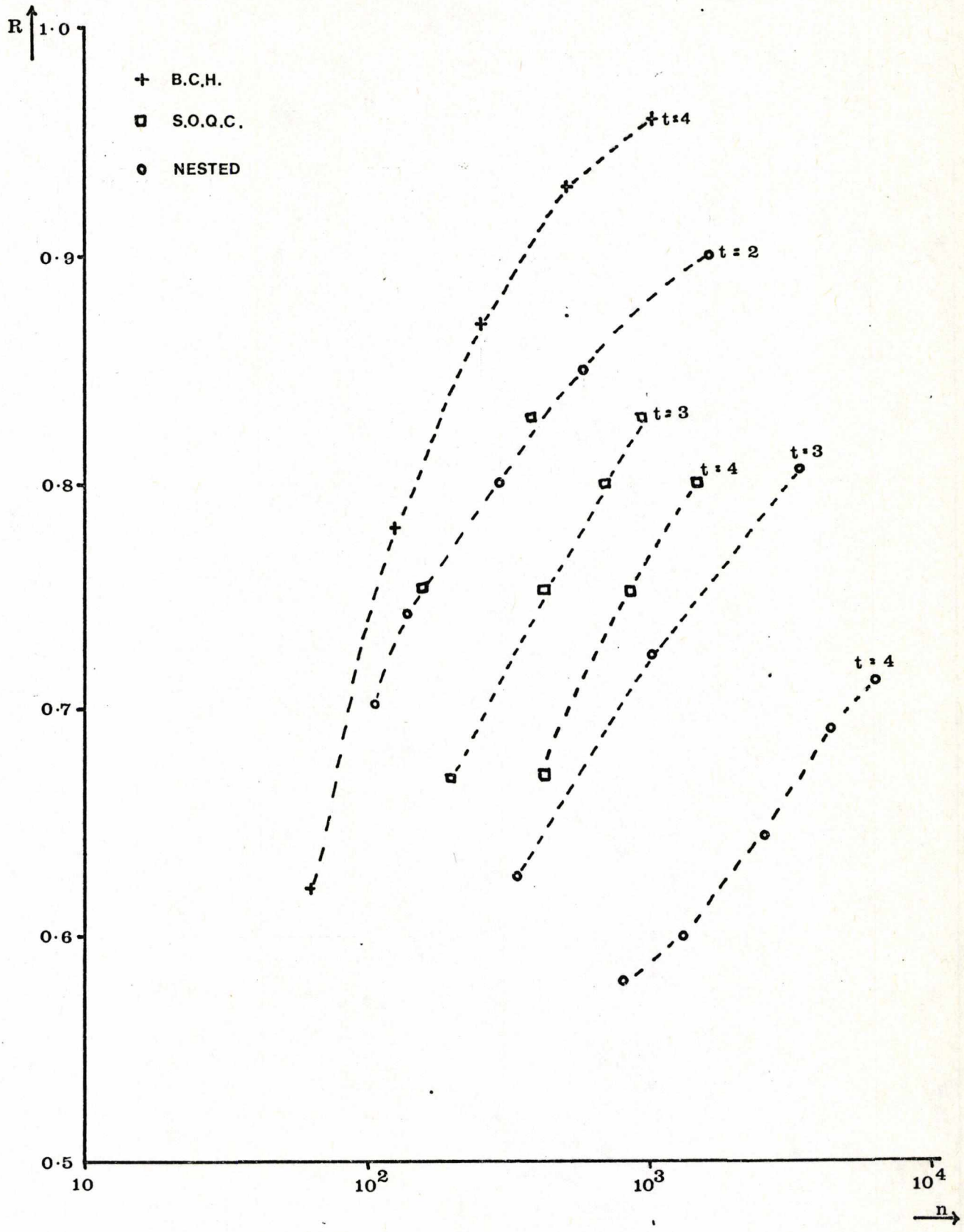

R
1·0
0·9
0·8
0·7
0·6
0·5
+ B.C.H.
S.O.Q.C.
NESTED
t=4
t = 2
t = 3
t = 4
t = 3
t = 4
10
10^2
10^3
10^4
n

FIG. 5.2

Minimum Distance	Length	No. Inf. Bits	Rate
5	26	13	0.5
5	49	30	0.612
5	105	74	0.704
5	143	106	0.738
5	296	238	0.803
5	608	518	0.851
5	1626	1468	0.902
5	3453	3214	0.93
7	76	38	0.5
7	161	91	0.565
7	343	215	0.626
7	1024	740	0.722
7	3609	2910	0.806
7	5336	4415	0.826
9	94	47	0.5
9	793	459	0.578
9	1301	793	0.609
9	4425	3054	0.69
9	6223	4425	0.71

Table 5.3 Parameters of "Easily Decoded" Nested codes

CHAPTER 6

Conjoined Codes

6.1. Introduction

In this chapter the properties of codes with generator matrices of the form $G = [G_1 \mid G_2]$ are investigated, where G_1, G_2 are generator matrices of other, shorter, linear codes, all having an equal number of information bits.

A codeword of such a code will consist of two codewords arranged consecutively, i.e. $C = C_1 \mid C_2$, and will have length $N = N_1 + N_2$, whilst $K = K_1 = K_2$; from which it is easy to deduce that the rate, R, is given by $R = \dfrac{R_1 R_2}{R_1 + R_2}$

The codes will be termed 'conjoined' codes.

Theorem

The minimum distance of C (G) is given by :

$$d \geqslant d_1 + d_2$$

Proof

A codeword of C (G) is given by $C = C_1 \mid C_2$

The weight of C_1 is either 0 or $\geqslant d_1$

The weight of C_2 is either 0 or $\geqslant d_2$

If the weight of C_1 is 0 then, since the codes are linear, and the information bits of C_2 are identical to those of C_1, the weight of C_2 is also 0.

Hence the weight of C is either 0 or $\geqslant d_1 + d_2$; therefore, since the conjoined code is linear, the minimum distance of C is $\geqslant d_1 + d_2$.

The most useful construction of the type described consists of combining a code of even minimum distance with one of odd minimum distance. This gives a conjoined code of odd minimum distance, which may be increased by one, if required, by adding an overall parity check

bit.

Example

We may choose C (G_1) to be the (8,4) code of minimum distance 4, and C (G_2) to be an identity matrix - i.e. a (4,4) code of minimum distance 1.

The resulting code is a (12,4) code of minimum distance 5.

No linear code of length 12 and minimum distance 5 has more than four information bits.

Example

Choose C (G_1) to be a (6,3) code with minimum distance 3, and C (G_2) to be a (7,3) code with minimum distance 4.

The resulting code is a (13,3) code of minimum distance 7.

No linear code of length 13 and minimum distance 7 has more than three information bits.

The codes do, unfortunately, become less efficient than some other codes of the same length and minimum distance, for increasing values of K.

Example

Choose C (G_1) to be the (24,12) code with minimum distance 8, and C (G_2) to be the (12,12) code of minimum distance 1.

The resulting code is a (36,12) code with minimum distance 9. This may appear to compare adversely with the best known linear code of the same length and minimum distance, which has sixteen information bits. Nevertheless, the parameters of the conjoined code are identical with those of the shortened code derived from the primitive BCH code of length 63 and minimum distance 9. The equivalent rate self-orthogonal quasi-cyclic code with minimum distance 9 has length 78.

The disadvantage of these codes having low rate is offset by the simple decoding algorithm which may be applied to them; which moreover

is capable of correcting some error patterns of weight greater than t.

6.2. The Decoding Algorithm for Conjoined Codes of Odd Minimum Distance

A. Decode the two constituent codewords separately, using a suitable decoder for each code;

B. If the decoder of the even minimum distance code detects an uncorrectable error pattern, then the decoded word of the odd minimum distance code is considered a correct decoding;

C. Otherwise attach to each decoded sequence $\hat{C}_1$, $\hat{C}_2$, an 'unreliability' number Y_1, Y_2 respectively, given by $Y_1 = M_1 + \frac{d_2 - d_1 - 1}{2}$

$$\text{and } Y_2 = M_2$$

Where M_1, M_2 are the number of bits inverted by the decoders of r_1, r_2 (i.e. the estimates of the numbers of errors in C_1 and C_2). The decoded sequence with the lowest unreliability number Y is considered to be a correct decoding. If $Y_1 = Y_2$ then the decoded sequence from the code of largest minimum distance is assumed correct. That this decoding procedure is valid will now be proved. The decoding situation may be divided into four possibilities :

(a) both decoded sequences are correct;

(b) the code of even minimum distance detects an uncorrectable error pattern;

(c) one of the decoded sequences is incorrect;

(d) both decoded sequences are incorrect.

It will first be shown that condition (d) never occurs unless $> t$ errors have corrupted the codeword C.

Theorem

It is impossible for both decoded sequences $\hat{C}_1$, $\hat{C}_2$ to be incorrect, given at most t errors in the received sequence $r_1 \mid r_2$.

Proof

If $\hat{C}_1$ is incorrect, then the number of errors, e_1 in r_1 is bounded by :

$$e_1 \geqslant \frac{d_1 + 1}{2}$$

and the number of errors in r_2 is bounded by :

$$e_2 \leqslant t - e_1$$

provided t errors have occurred overall.

i.e. $$e_2 \leqslant \frac{d_1 + d_2 - 1}{2} - \frac{d_1 + 1}{2}$$

hence $$e_2 \leqslant \frac{d_2 - 2}{2}$$

and therefore, since $\hat{C}_2$ is correct for $e_2 \leqslant \frac{d_2 - 1}{2}$, $\hat{C}_2$ must be a correct decoding.

Similarly, if $\hat{C}_2$ is an incorrect decoding, then C_1 must be correct. ∎

It will next be shown that in situation (b), the codeword from the code of odd minimum distance will be correctly decoded. Without loss of generality, it can be assumed that C (G_1) has even minimum distance and C (G_2) odd minimum distance.

Theorem

If r_1 is found to have an incorrectable error pattern then r_2 has a correctable error pattern, given at most t errors overall.

Proof

If r_1 has an uncorrectable error pattern then $e_1 \geqslant d_1/2$

but $e_2 \leqslant t - e_1$

hence $$e_2 \leqslant \frac{d_1 + d_2 - 1}{2} - \frac{d_1}{2}$$

i.e. $$e_2 \leqslant \frac{d_2 - 1}{2}$$

and therefore r_2 may be correctly decoded. ∎

Clearly, in case (a), the decoding algorithm cannot fail, since whichever sequence is taken to be correct there will be no decoding error.

Case (c) is left, where only one of $\hat{C}_1$, $\hat{C}_2$, is correct. It must be shown that the unreliability numbers Y_1, Y_2, will indicate the correct decoding. It may be assumed, without loss of generality, that C (G_1) has the lower minimum distance.

Theorem

If $Y_1 < Y_2$ then $\hat{C}_1$ is a correct decoding.

If $Y_1 \geqslant Y_2$ then $\hat{C}_2$ is correct, given at most t errors overall

Proof

Consider the two cases :

(i) $\hat{C}_1$ is correct, $\hat{C}_2$ incorrect

(ii) $\hat{C}_2$ is correct, $\hat{C}_1$ incorrect

(i) $\hat{C}_1$ is correct

Then $e_2 \geqslant \frac{d_2 + 1}{2}$

and $e_1 \leqslant t - e_2$

but $e_1 \leqslant \frac{d_1 - 1}{2}$

hence $M_1 \leqslant t - e_2$

and $M_2 \geqslant d_2 - e_2$

then, if $\frac{d_2 - d_1 - 1}{2}$ is positive - i.e. $d_2 > d_1 -$

$$Y_1 \leqslant \frac{d_1 + d_2 - 1}{2} - e_2 + \frac{d_2 - d_1 - 1}{2}$$

therefore $Y_1 \leqslant d_2 - e_2 - 1$

and $Y_2 \geqslant d_2 - e_2$;

hence if $\hat{C}_1$ is correct, $Y_1 < Y_2$

(ii) $\hat{C}_2$ is correct

then $e_1 \geqslant \frac{d_1 + 2}{2}$

and $e_2 \leqslant t - e_1$

hence $M_1 \geqslant d_1 - e_1$

and $M_2 \leqslant t - e_1$

but $M_2 \leqslant \frac{d_1 - 1}{2}$

hence $Y_1 \geqslant d_1 - e_1 + \frac{d_2 - d_1 - 1}{2}$

provided $\frac{d_2 - d_1 - 1}{2}$ is positive, i.e. $d_2 > d_1$

Therefore $Y_1 \geqslant \frac{d_2 + d_1 - 1}{2} - e_1$

but $Y_2 \leqslant \frac{d_1 + d_2 - 1}{2} - e_1$

so that if $\hat{C}_2$ is correct $Y_1 \geqslant Y_2$. ▌

This completes the justification of the decoding algorithm. Note that the algorithm does not require the constituent codes to be linear - hence non-linear and linear codes may be conjoined in any desired combination.

The practical realisation of the decoders is simple - consider for example a conjoined code with d_1 even, d_2 odd, and $d_1 > d_2$.

r_1 is first decoded, and an updown counter is incremented by one for each received, bit inverted. If an uncorrectable error pattern is detected then C_2 is decoded and delivered to the sink, otherwise as C_2 is decoded, the counter is decremented by one for each error in r_2 corrected.

If the final content of the counter is found to be greater than $(d_1 - d_2 - 1)/2$ then $\hat{C}_2$ is output to sink, otherwise $\hat{C}_1$ is output.

The counting of errors is particularly simple if, as occurs in very many cases, the decoders for C_1 and C_2 are of the type where the received sequence is stored in a buffer register, and the decoding logic arranged so as to correct the bits as they emerge - since then the output from the decoding logic may be taken not only to the inverting gate, for error correction, but also to the counting circuit for error enumeration.

Conjoined codes of even minimum distance may be constructed from two codes of either odd or even minimum distance. In this case the decoding algorithm must be modified to enable error patterns of weight d/2 to be detected. This modification is now described.

6.3. Decoding Procedure for Conjoined Codes of Even Minimum Distance

Whether an even minimum distance conjoined code is formed from two odd or two even minimum distance codes, the decoding procedure for correcting the maximum number, $(d - 2)/2$, of errors per block is the same as for odd minimum distance conjoined codes except that Y_1 is now given by :

$$Y_1 = M_1 + \frac{d_2 - d_1}{2}$$

and the proof that this algorithm is valid follows through in the same way. It remains to give a procedure for detecting d/2 errors. The two classes of even minimum distance conjoined codes are considered separately.

6.3.1. Class (a) Codes Formed by Conjoining Two Odd Minimum Distance Codes

Here it is noted that the number of correctable errors is given by :

$$t = \left\lceil \frac{d_1 + d_2 - 1}{2} \right\rceil = \frac{d_1 + d_2 - 2}{2} = \frac{d_1 - 1}{2} + \frac{d_2 - 1}{2} = t_1 + t_2$$

but, $\frac{d}{2} = t + 1$

hence $d/2 = t_1 + t_2 + 1$

Therefore, given d/2 errors overall, either $\hat{C}_1$ or $\hat{C}_2$ must be a correct decoding, but not both. We may assume without loss of generality that $d_2 \geqslant d_1$, and then consider the two possible cases :

Case 1 - $\hat{C}_1$ is a Correct Decoding

Then $M_1 = e_1$

and $M_2 \geqslant d_2 - e_2$

but $d/2 = e_1 + e_2$

Therefore $M_2 \geqslant d_2 - \frac{(d_1 + d_2)}{2} + e_1$

Therefore $M_2 \geqslant \frac{d_2 - d_1}{2} + e_1$ Hence $Y_1 = e_1 + \frac{d_2 - d_1}{2}$

and $Y_2 = M_2 \geqslant \frac{d_2 - d_1}{2} + e_1$

hence $Y_2 \geqslant Y_1$

Therefore, by the decoding algorithm proposed, if $Y_2 > Y_1$ then $\hat{C}_1$ is rightly assumed to be a correct decoding. Otherwise, $Y_2 = Y_1$ and $\hat{C}_1 \neq \hat{C}_2$ and then d/2 errors are detected.

Case 2 - $\hat{C}_2$ is Decoded Correctly

The argument here is similar to that of Case 1 above, with the result :

$$Y_1 \geqslant Y_2$$

So that if $Y_1 > Y_2$, $\hat{C}_2$ is rightly assumed to be correct, otherwise $Y_1 = Y_2$ and $C_2 \neq C_1$ when d/2 errors are again detected.

To complete the justification of the decoding

algorithm modification it remains to be shown that $Y_2 = Y_1$ and $\hat{C}_1 \neq \hat{C}_2$ does not occur for less than d/2 errors overall. This case is covered by the following argument.

For $\hat{C}_1 \neq \hat{C}_2$ we must have either $e_1 > t_1$ or $e_2 > t_2$.

If $e_1 > t_1$ then $Y_1 > t_1 + \frac{d_2 - d_1}{2}$

Therefore $Y_1 > \frac{d_2 - 1}{2}$

but $Y_2 = M_2 = e_2 < t_2$

hence $Y_2 < \frac{d_2 - 1}{2}$

So $Y_1 > Y_2$

Similarly if $e_2 > t_2$ then $Y_2 > Y_1$

So for $e_1 + e_2 < d/2$ $Y_1 \neq Y_2$

This completes the justification.

6.3.2. Class (b): Conjoined Codes Formed from Codes of Even Minimum Distance

Here the unreliability number Y_1 is again modified to $Y_1 = M_1 + \frac{d_2 - d_1}{2}$ and d/2 errors are detected if $Y_1 = Y_2$ and $\hat{C}_1 \neq \hat{C}_2$, but d/2 errors are also detected if $t_1 + 1$ errors are detected in r_1 at $t_2 + 1$ errors in r_2. This additional situation arises since :

$$t_1 = \frac{d_1 - 2}{2} \text{ and } t_2 = \frac{d_2 - 2}{2}$$

while $t = \left\lceil \frac{d_1 + d_2 - 1}{2} \right\rceil = \frac{d_1 + d_2 - 2}{2} = t_1 + t_2 + 1$

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

allowing the possibility, when d/2 errors have occurred, of $t_1 + 1$ errors occurring in r_1, and $t_2 + 1$ errors occurring in r_2.

TABLE 6.1

SOME UNAUGMENTED CONJOINED CODES

Conjoined Code $n,k,d,$	C_1 n_1,k_1,d_1	C_2 n_2,k_2,d_2	Best Known Code $N,K,D,$
11,2,7	5,2,3	6,2,4	11,2,7
13,3,7	6,3,3	7,3,4	13,3,7
15,4,7	7,4,3	8,4,4	15,5,7
23,7,7	11,7,3	12,7,4	23,12,7
17,2,11	8,2,5	9,2,6	17,2,11
20,3,11	7,3,4	13,3,7	20,3,11
23,2,15	5,2,3	18,2,12	23,2,15
27,3,15	6,3,3	21,3,12	27,3,15
29,4,15	14,4,7	15,4,8	29,4,15
43,10,15	21,10,7	22,10,8	43,11,15
48,3,27	21,3,12	27,3,15	48,3,27
52,4,27	23,4,12	29,4,15	52,4,27

n, n_1, n_2, N = Length of conjoined code, C_1, C_2, best known code.

k, k_1, k_2, K = Number of information bits in conjoined code, C_1, C_2, best known code.

d, d_1, d_2, D = Minimum distance of conjoined code, C_1, C_2, best known code.

Justification of this algorithm otherwise follows the pattern of that for the previous classes of conjoined code.

The constituent codes of a conjoined code are both shorter and of a lower minimum distance than the resultant code, and are consequently much more easily decoded than most codes having the parameters of the resultant code. The total complexity of the conjoined code decoder is only slightly greater than the sum of the complexities of the constituent decoders, and hence at the expense of efficiency the conjoined code is easily decodable.

Since the rate of a conjoined code is given by

$$R = \frac{R_1 R_2}{R_1 + R_2}$$

conjoined codes are restricted to $R < \frac{1}{2}$. Fortunately the codes may be augmented by adding further codewords, in a simple manner, to give codes where $\frac{1}{2} \leqslant R < 1$.

A list of unaugmented conjoined codes is given in Table 6.1.

6.4. Augmentation

Consider a conjoined code in which G_1 contains x columns identical to x columns of G_2. Then each codeword will contain x bits in C_1 which are identical to those x bits of C_2 in the positions corresponding to identical columns in G_1 and G_2.

Let the two sets of bits related in this way be the x tuples x_1 and x_2.

Now, if such a codeword is received with errors in at most t positions, it is clear that x_2 will differ from x_1 in no more than

t positions.

Hence, if a codeword, C_3, from a code of length x and minimum distance $d_1 + d_2$, is added modulo-2 to x_2 it may be recovered at the receiving end by adding modulo-2 the received x_1 to the received $C_3 \oplus x_2$, to give C_3 corrupted in at most t bits. C_3 may then be decoded in the normal way.

Having recovered C_3 it may be modulo-2 added to the received $C_3 \oplus x_2$ to give the original unaugmented codeword, corrupted by no more than t bits, which may then be decoded by the algorithm previously described. In this way the original code has been augmented by the number of information bits in C_3. Note that the augmented code is not systematic. The augmentation technique does not require C_1, C_2, or C_3 to be linear, and so again non-linear and linear codes may be combined in any way.

Example

Consider the (6,3) shortened Hamming code with minimum distance 3 which has the generator matrix.

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

and the (5,3) code with minimum distance 2 obtained by puncturing the final column of G_1, giving :

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

clearly the bits of a codeword, C_2, from G_2, would be identical to the first five bits of a codeword, C_1, from G_1 obtained by encoding the same data bits.

Hence the code obtained by conjoining G_1 and G_2 which has minimum distance 5, may be augmented by a codeword from the (5,1) repetition

TABLE 6.2

SOME AUGMENTED CONJOINED CODES

Augmented Conjoint Code n,k,d,	C_1 n,k,d,	C_2 n,k,d,	C_3 n,k,d,	Best Known Code n,k,d,
11,4,5	5,3,2	6,3,3	5,1,5	11,4,5
15,5,7	8,4,4	7,4,3	7,1,7	15,5,7
19,6,7	10,5,4	9,5,3	9,1,7	19,8,7
23,9,7	12,7,4	11,7,3	11,2,7	23,12,7
31,16,7	16,11,4	15,11,3	15,5,7	31,16,7
47,30,7	24,18,4	23,18,3	23,12,7	47,31,7
63,42,7	32,26,4	31,26,3	31,16,7	63,46,7
16,5,8	8,4,4,	8,4,4,	8,1,8	16,5,8
32,16,8	16,11,4	16,11,4	16,5,8	32,16,8
64,42,8	32,26,4	32,26,4	32,16,8	64,46,8
33,10,9	17,9,5	14,9,4	9,1,9	33,14,9
41,14,11	21,11,6	20,11,5	20,3,11	41,18,11
47,19,11	24,14,6	23,14,5	23,5,11	47,24,11
63,32,11	32,21,6	31,21,5	31,11,11	63,36,11
31,6,15	16,5,8	15,5,7	15,1,15	31,6,15
47,14,15	24,12,8	23,12,7	23,2,15	47,15,15
63,22,15	32,16,8	31,16,7	31,6,15	63,28,15
32,6,16	16,6,8	16,6,8	16,1,16	32,6,16
64,22,16	32,16,8	32,16,8	32,6,16	64,28,16
45,7,19	23,6,10	22,6,9	22,1,19	45,7,19
47,6,23	24,5,12	23,5,11	23,1,23	47,6,23
53,8,23	27,7,12	26,7,11	26,1,23	53,8,23
55,6,27	28,5,13	27,5,13	27,1,27	55,6,27

code with generator matrix given by :

$$G_3 = [1\ 1\ 1\ 1\ 1]$$

hence by conjoining G_1 and G_2 and augmenting with G_3 an (11.4) code is obtained with minimum distance 5 and generator matrix given by :

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

No linear binary code of length 11 and minimum distance 5 may have more than 4 information bits.

The complexity of the decoder for an augmented conjoined code is little more than the sum of the complexities of the decoders for C_1, C_2, and C_3.

The decoder for C_3 will generally be the most complex, since it has a minimum distance equal to that of the conjoined code. It is, however, less than half the length of the conjoined code - and will therefore have a lower rate. This will generally make C_3 considerably easier to decode. Often, for example, C_3 may be an error trappable cyclic, or shortened cyclic, code.

A list of some augmented conjoined codes is given in Table 6.2. The codes listed are of two types - in the first C_1 is simply C_2 with an added overall parity check bit, so that N_1 of the columns of G_1 and G_2 are identical, enabling a code of length N_1 to be added to the conjoined code, whereas in the second case the augmenting codeword is only K bits long and added to the information bits of C_2 - the columns of G_1 and G_2 being identical in the information positions provided C_1 and C_2 are systematic codes.

For low, and very low, rates many of the codes formed are optimum, and even for high rates many good codes may be constructed. Notable

amongst these are codes with identical parameters to the Reed Muller codes, and the connection between the codes will now be demonstrated.

6.5. R.M. Codes and Augmented Conjoined Codes

To begin with, it will be shown that the first order R.M. codes, (which are identical to the maximal length shift register codes), may be formed by a conjoining and augmenting process, starting with a (2,2) code of minimum distance 1, with the generator matrix :

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

and augmenting with $(2^m,1)$ repetition codes of minimum distance 2^m, when required.

A Reed Muller code of order one has length 2^m, minimum distance $2^m - 1$ and has m + 1 information bits. The columns of the generator matrix of such a code consist of each possible m-tuple, preceded by a 1, as shown below for m = 3.

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

Therefore G = 1 1 is the generator matrix of the first order R.M. code with m = 1 and hence k = 2, n = 2, d = 1.

The remaining R.M. codes may be formed from this as follows :

Firstly it is conjoined with itself :

$$G^1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$ to give a (4,2) code with d = 2

then it is augmented by a (2.1) repetition code $G^{11} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$ to give

a (4,3) code with d = 2. This is the first order R.M. code with m = 2. This is then conjoined with itself and augmented by a (4,1) repetition code to give the first order R.M. code :

$$G^{111} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

This process may be continued indefinitely, to give all of the first order R.M. codes.

Note that $G = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ may be transformed to $G = \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$ by a simple row operation - and the latter, more familiar matrix for a 'code' of minimum distance 1 may be used to form codes with identical n, k, d, and also weight distribution, to the R.M. first order codes.

This explanation will now be extended to show that an rth order R.M. code of length 2^m may be formed by conjoining two rth order R.M. codes of length $2^m - 1$ and augmenting with an $(r - 1)$th order R.M. code of length $2^m - 1$.

It will be recalled that an rth order R.M. code is formed by using as a basis the vectors V_o, V_1, ... V_m and all vector products of these vectors r or fewer at a time, (Peterson & Weldon 1972), where V_o is a row of 2^m ones, V_1 is a row of 2^m alternate zeros and ones, and V_i is a row of 2^m alternate groups of 2^i zeros and ones, as is illustrated in Fig. 6.1 for m = 4.

If the V_i for a R.M. code of length 2^m are termed $V_i(m)$ then it is easy to see that if $V_i(m)$ is repeated it becomes $V_i(m + 1)$.

e.g. $V_3(3)$ is given by 0 0 0 0 1 1 1 1

which repeated becomes 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1

which is $V_3(4)$.

It is also seen that if any vector product $V_j(m)V_k(m)$ is repeated, it becomes $V_j(m+1)\,V_k(m+1)$ by the nature of vector products.

e.g. $V_2(3)V_3(3)$ is given by :

(0 0 1 1 0 0 1 1) (0 0 0 0 1 1 1 1)

= 0 0 0 0 0 0 1 1

which repeated becomes 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1

Now, $V_2(4)V_3(4)$ is given by :

(0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1) (0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1)

= 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 = $V_2(3)V_3(3)$ repeated.

V_o	=	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
V_1	=	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
V_2	=	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
V_3	=	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
V_4	=	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
$V_1\,V_2$	=	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
$V_1\,V_3$	=	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$V_1\,V_4$	=	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$V_2\,V_3$	=	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$V_2\,V_4$	=	0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
$V_3\,V_4$	=	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
$V_1\,V_2\,V_3$	=	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
$V_1\,V_3\,V_4$	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
$V_1\,V_2\,V_4$	=	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
$V_2\,V_3\,V_4$	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
$V_1\,V_2\,V_3\,V_4$	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Fig. 6.1 - Basis Vectors for R.M. Codes with m = 4

Therefore it can be seen that a generator matrix composed of rows which are vectors $V_o(m)$, $V_1(m)$, $V_2(m)$, ... $V_m(m)$ and all vector products of these vectors r or fewer at a time (that is, the generator

matrix of an rth order R.M. code of length 2^m), when conjoined with itself, becomes a generator matrix composed of rows which are the vectors $V_o(m+1)$, $V_1(m+1)$, ... $V_m(m+1)$, and all vector products of these vectors taken r or fewer at a time.

Now consider rows augmenting the generator matrix in the manner previously described. Each will consist of 2^m zeros followed by 2^m bits corresponding to a row of the augmenting code generator matrix.

It is next necessary to consider the effect of prefixing a $V_i(m)$ by 2^m zeros.

This is seen to be equivalent to the vector product of $V_i(m+1)$ with $V_{m+1}(m+1)$.

e.g. $V_3(3)$ prefixed by 2^3 zeros becomes

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

and $V_3(4)V_4(4)$ is given by

(0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1) (0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1)

= 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1

Similarly, any product $V_j(m)\ V_k(m)$ prefixed by 2^m zeros is equivalent to the vector multiple of $V_j(m+1)$ with $V_{m+1}(m+1)$.

Hence if the augmenting code is an $(r-1)$th order R.M. code of length 2^m; that is, a code with a generator matrix comprised of rows which are the vectors $V_o(m)$, $V_1(m)$, ... $V_m(m)$ and vector products of these vectors taken $(r-1)$ or fewer at a time; then the rows augmented to the conjoined generator matrix are equivalent to the vector products $V_{m+1}(m+1)\ V_o(m+1)$, $V_{m+1}(m+1)\ V_1(m+1)$, ... $V_{m+1}(m+1)\ V_m(m+1)$ and all vector products of $V_{m+1}(m+1)$ with the vector multiples of $V_o(m+1)$, $V_1(m+1)$, ... $V_m(m+1)$ taken $(r-1)$ or fewer at a time. Note that $V_{M+1}(m+1)\ V_o(m+1) = V_{m+1}(m+1)$.

Therefore the row of the augmented conjoined generator matrix will now comprise the vectors $V_o(m+1)$, $V_1(m+1)$, ... $V_{m+1}(m+1)$ and all vector products of these vectors taken r or fewer at a time.

This is precisely the definition of an rth order R.M. code of length $2^m + 1$.

Thus it has been shown that an rth order R.M. code of length 2^m may be constructed by conjoining two rth order R.M. codes of length $2^m - 1$ and augmenting with an $(r - 1)$th order R.M. code of length $2^m - 1$.

This decomposition of R.M. codes enables simple decoders to be constructed for some high order codes.

Consider, for example, the (32,16) second order R.M. code with minimum distance seven. This may be constructed as two (16,11) second order R.M. codes (these are equivalent to the Hamming single error correcting double error detecting codes) conjoined and augmented by the (16,5) first order R.M. code (which is equivalent to the BCH (15,5) triple-error-correcting code with an additional parity check bit).

Therefore the (32,16) code may be decoded by decoding two (16,11) codes and one (16,5) code, all of which may be error trap decoded.

Clearly, the decoder for the (16,11) code may be used twice, and so the decoding hardware consists of little more than an error trap decoder for the (16,11) code and one for the (16,5) code.

CHAPTER 7

Array Codes

7.1. Introduction

In this chapter, classes of self-orthogonal codes are described which are based upon the arrangement of information bits into two-dimensional arrays. Parity checks are then made on certain carefully chosen patterns on the arrays. These codes are easily decodable, and some are of a rate close to the optimum for self-orthogonal codes. Moreover, the codes may be augmented with additional codewords, to give codes of rate higher than that achievable with self-orthogonal codes of the same length and minimum distance. The extension of the codes by annexing additional check bits is also considered - thus increasing the minimum distance of the codes at the expense of the ease with which they may be decoded.

7.2. Construction A

Consider points on a two-dimensional surface. No two straight lines in the plane of the points will pass through more than one common point. That is, no two points lie on more than one straight line. This is the basis of the self-orthogonal codes to be constructed.

It is well known (e.g. Massey 1963) that a code is self-orthogonal with minimum distance d if no two information bits are involved together in more than one parity check equation, and each information bit is involved in a least (d - 1) parity check equations.

The analogy is clear. If information bits are considered as points on a two-dimensional array, and parity check equations as straight lines passing through those points, then a self-orthogonal code of minimum distance d is described by a two-dimensional array of points, with sufficient

straight lines passing through the points such that each point is crossed by at least (d - 1) lines.

In order to make such codes as efficient as possible, each straight line should pass through as many points as possible. The least number of lines is then used - that is, the least number of parity check bits is achieved.

Consider as a starting point, information bits arranged in a square array. The most efficient straight lines that can be drawn through the points are horizontal and vertical lines. If there are S^2 points in the square, then there are 2S such lines, and each point is crossed by just two lines. This arrangement therefore describes a self-orthogonal code with S^2 information bits, 2S parity check bits, and minimum distance 3.

The next most efficient set of lines are the diagonals - which may be in two directions - where 2S - 1 diagonals in one direction will cover the entire array. Hence 2S + (2S - 1).2 lines will cross each point four times - giving a code with S^2 information bits, 6S - 2 check bits, and minimum distance 5.

There are four groups of lines of the next most efficient type - which are "knights move" lines (as in chess) across the array, as shown in Fig. 7.1. These lines, with the previous ones, give codes of minimum distance 7 if two "knights move" lines are taken, or minimum distance 9 if all are used. The number of parity check bits for S^2 information bits are 12S - 6 and 18S - 10 respectively.

A table of codes of minimum distances 3, 5, and 7 are given in Table 7.1, from which it can be seen that they are of poor efficiency for a given length and minimum distance.

This disadvantage is offset by the ease with which the codes may be decoded, and the fact that a very large number of errors of weight

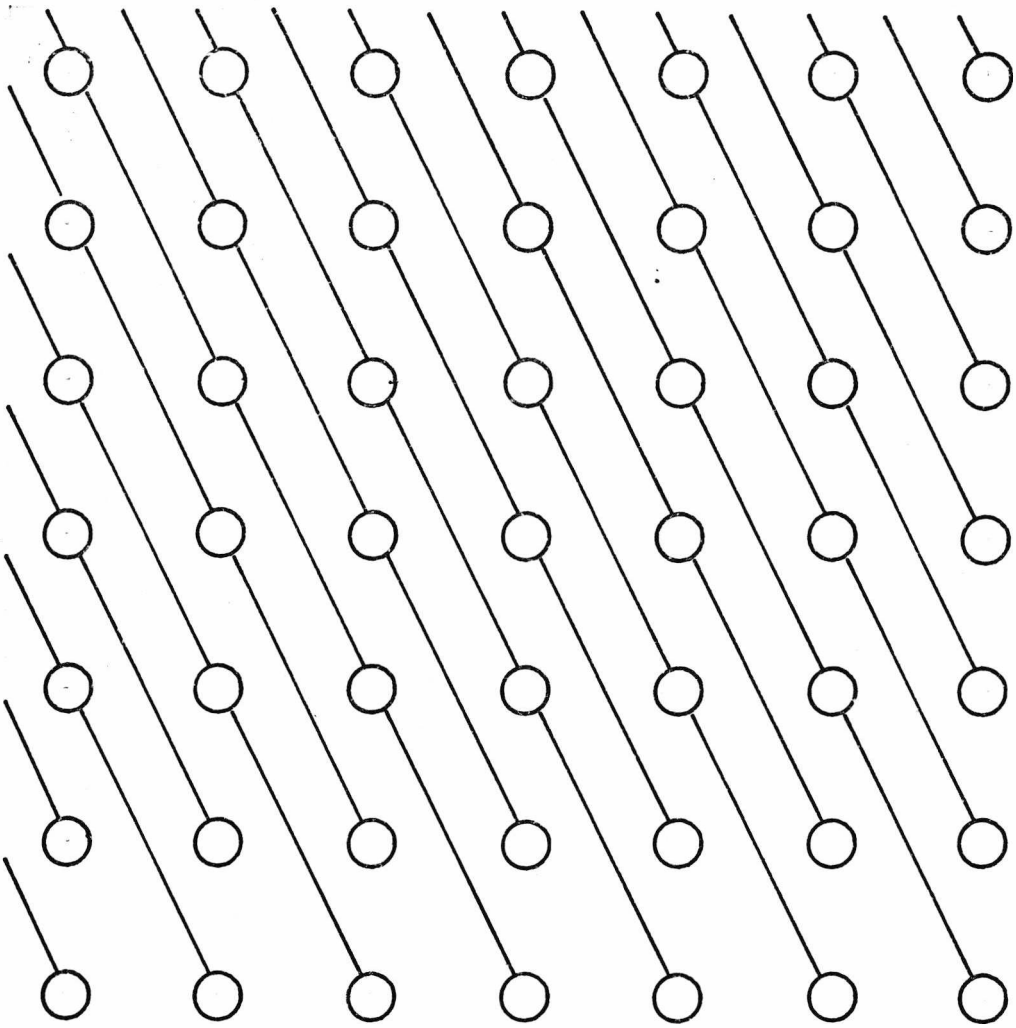


Fig 7.1.

TABLE 7.1

CODES OF CONSTRUCTION A

Minimum Distance	n	k	R
3	$S^2 + 2S$	S^2	k/n
3	3	1	0.333
3	8	4	0.5
3	15	9	0.6
3	24	16	0.667
3	35	25	0.714
3	48	36	0.75
3	63	49	0.778
3	80	64	0.8
5	$S^2 + 6S - 2$	S^2	k/n
5	5	1	0.2
5	14	4	0.286
5	38	16	0.42
5	70	36	0.514
5	110	64	0.582
5	158	100	0.633
5	214	144	0.673
7	$S^2 + 12S - 6$	S^2	
7	7	1	0.143
7	22	4	0.182
7	58	16	0.276
7	102	36	0.353
7	154.	64	0.415
7	214	100	0.467
7	282	144	0.511

greater than $(d - 1)/2$ are correctable.

7.2. Augmentation of the Codes

With codes formed as above, the parity check bits may be divided into $(d - 1)$ sets, each set represented by lines which pass through every point in the array, but each point being passed through by only one line in the set. Examples of such sets are the sets of horizontal, vertical, diagonal, and "knights move" lines referred to in the examples of construction of the codes. Each set will be termed a "direction set".

Within each direction set, all the parity check bits are independent - which is to say no parity check bit is dependent upon any information bits which contribute to the determination of any other parity check bit within the same direction set.

Let the number of parity checks within a direction set D_i $(1 \leqslant i \leqslant d - 1)$ be n_i. Then a codeword from a code of length n_i and minimum distance d may be bit-by-bit modulo -2 added to the parity check bits in the direction set D_i, for all i, $1 \leqslant i \leqslant d - 1$ to give an array codeword augmented by k_i information bits, where k_i is the number of information bits in the code of length n_i. The minimum distance of the code remains d, and the length of the augmented code is that of the unaugmented code.

That such an augmentation is permissible will be shown by explaining how the received augmented codeword may be decoded.

First, an estimate of the unaugmented parity check bits is made by re-encoding the received data bits. Since each parity check bit is independent of others within the same direction set, then given at most $t = (d - 1)/2$ errors in the received codeword there can be at most t errors in the estimated parity checks within each direction set.

If these estimated bits are now added modulo -2 to the respective

received bits, the result will be the augmented codeword bits, corrupted in at most t bits in each codeword. These "estimated codewords" may therefore be decoded, since they are from codes of minimum distance $d = 2t + 1$. The corrected words may then be modulo -2 added to the received parity check bits to give an unaugmented codeword corrupted in at most t bits. The unaugmented code is of minimum distance $2t + 1$ and so the bits in error may be found.

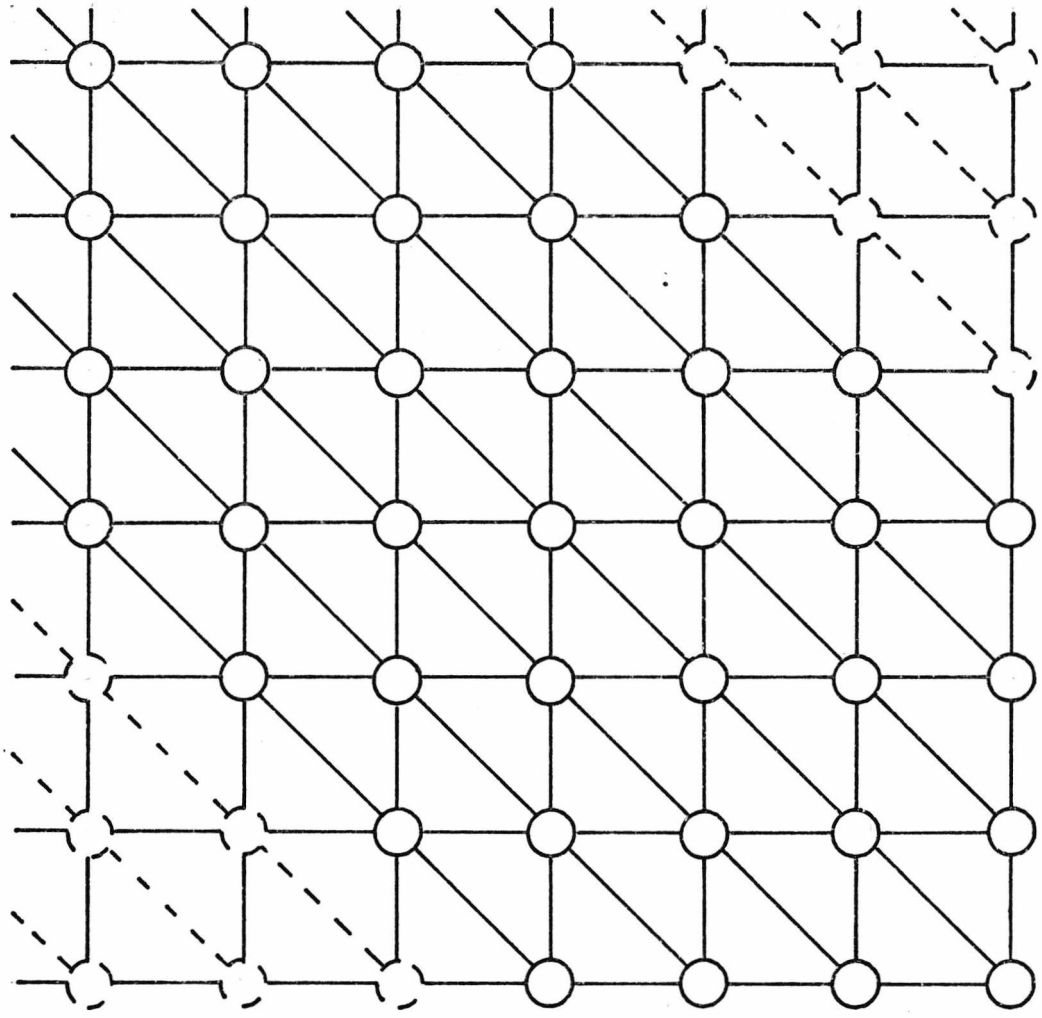
A form of this process of augmentation is used by Kasahara, et al (1976) to augment concatenated and product codes - but the augmentation codewords in this case must in general be from multilevel codes, and the resulting codes, although powerful, are not easily decodable.

A list of double-error-correcting augmented codes is given in Table 7.2, from which can be seen the great improvement in the achieved rate. The complexity of decoding the augmented codes is little more than the complexity of decoding the augmenting codes, whose lengths vary approximately as the square root of the length of the code, plus that of decoding the unaugmented code.

7.3. Shortening of the Codes by Omitting Certain Information and Parity Check Bits

When the information bits of a code of construction A are represented by a square array, certain parity checks are very inefficient. The diagonal lines passing through corner points, for example, represent parity check equations involving only one information bit.

It is possible to avoid these wasteful parity checks, to give codes of improved rate, by omitting certain points in the square array so that lines passing through relatively few points may also be omitted. This process is illustrated in Fig. 7.2 for the case of a code of minimum distance 4, where the most efficient shape of array is found to be a



INFORMATION BIT

PARITY CHECK

DELETED INFORMATION BIT

DELETED PARITY CHECK

Fig 7.2

TABLE 7.2

DOUBLE ERROR CORRECTING AUGMENTED CONSTRUCTION A CODES

n	k unaug	n_a, k_a	k aug	R aug
25	9	5,1;5,1	11	0.44
38	16	6,1;6,1	18	0.474
53	25	9,2;9,2;5,1;5,1	31	0.585
70	36	11,4;11,4;6,1;6,1	46	0.657
89	49	13,5;13,5;7,1;7,1	61	0.685
110	64	15,7;15,7;8,2;8,2	82	0.746
133	81	17,9;17,9;9,2;9.2	103	0.775
158	100	19,10;19,10;10,3;10,3	126	0.797
185	121	21,12;21,12;11,4;11,4	153	0.827
214	144	23,14;23,14;12,4;12,4	180	0.841

k unaug = Number of information bits in unaugmented code.

k aug = Number of information bits in augmented code.

R aug = Rate of augmented code.

n_a = Lenths of augmenting codes.

k_a = Number of information bits in augmenting codewords.

hexagon with equal numbers of points on each side. The omitted points and lines are shown dotted. A table of shortened codes of minimum distance 4 is given in Table 7.3 together with some of minimum distance 5. The latter are most efficient with the array in the form of an octagon with an equal number of points on each side. It is conjectured that the array should be a regular (in the sense that each side has an equal number of points) $2(d-1)$-agon for a code of minimum distance d.

The improvement in rate of shortened codes over unshortened codes is illustrated by the double-error-correcting case. For the unshortened case $n = S^2 + 6S - 2$ and $k = S^2$, and the efficiency is tightly lower bounded by this expression.

$$36R\,(1 - R)^{-2} < n$$

For the shortened case, a regular octagon with S points to each side will contain $7S^2 - 10S + 4$ points - and the number of horizontal, vertical, and diagonal lines required to cross each point four times is $14S - 10$. Hence a shortened code has length n^1 and number of information bits k^1 given by $n^1 = 7S^2 + 4S - 6$ and $k^1 = 7S^2 - 10S + 4$ from which it may be deduced that the length of a shortened double error correcting code of given rate R is tightly upper bounded by :

$$n^1 > \frac{8R + 20}{(1 - R)^2}$$

which for high rates is 7/9 the length of the unshortened codes.

7.4. Augmentation of the Shortened Codes

The parity check bits are still arranged in direction sets, and so augmentation of the codes is carried out in an identical manner to that for the unshortened codes. The rate of the codes for a given length and minimum distance is greatly increased as may be judged from the list of double-error-correcting augmented shortened codes given in Table 7.4.

TABLE 7.3

SOME SHORTENED CODES OF CONSTRUCTION A

Minimum Distance	n	k	R
4	$3S^2 + 3S - 2$	$3S^2 - 3S + 1$	k/n
4	16	7	0.438
4	34	19	0.543
4	58	37	0.638
4	88	61	0.693
4	124	91	0.734
4	160	127	0.765
4	214	169	0.790
5	$7S^2 + 4S - 6$	$7S^2 - 10S + 4$	k/n
5	30	12	0.4
5	69	37	0.536
5	122	76	0.623
5	189	129	0.683
5	270	196	0.726
5	365	277	0.759
5	474	372	0.785
5	597	481	0.806
5	734	604	0.823

TABLE 7.4

AUGMENTED SHORTENED DOUBLE ERROR CORRECTING ARRAY CODES

OF CONSTRUCTION A

Length	k unaug	n_a, k_a	k aug	R aug
30	12	5,1;5,1	14	0.467
69	37	9,2;9,2;7,1;7,1	43	0.623
122	76	13,5;13,5;10,3;10.3	92	0.754
189	129	17,9;17,9;13,5;13,5	157	0.831
270	196	21,12;21,12;16,8;16.8	236	0.874
365	277	25,15;25,15;19,10;10,10	327	0.896
474	372	29,19;29,19;22,13;22,13	436	0.920
597	481	33,22;33,22;25,15;25,15	555	0.930

k unaug = Number of information bits in unaugmented code.

n_a = Lengths of augmenting codewords.

k_a = Number of information bits in augmenting codewords.

R aug = Rate of augmented code.

k aug = Number of information bits in augmented code

Note, however, that the rates of the augmented shortened double-error-correcting codes are little better than those of comparable augmented unshortened codes.

7.5. Decoding Unaugmented Unshortened Codes of Construction A

As has been mentioned, the self-orthogonal class of codes is decodable with majority logic decision elements, and by this technique many error patterns of a weight not guaranteed correctable by the minimum distance of a code may be corrected.

The codes may be one-step majority logic decoded by virtue of the (d - 1) parity checks orthogonal on each information bit, as a result of which (d - 1) estimates of each information bit may be made, with no more than (d - 1)/2 of these estimates being incorrect.

In practice estimates of each bit are not made, but the parity checks are recalculated from the received information bits and compared with the received parity check bits - and information bits are inverted if a clear majority of the parity check equations in which they are involved fail. This is seen to be equivalent to making estimates of the received bits and then making a majority decision.

Note that if the received information bits are compared to the received parity check bits by modulo -2 addition a '1' result indicates failure of the parity check equation, and the operation is completely equivalent to forming the usual syndrome of the received word. Thus the decoding operation reduces to making majority decisions based on the values of selected bits in the syndrome of the received word.

If the received words are decoded sequentially, that is one bit at a time, then only one (d - 1) input majority gate is required, giving a logical '1' output if more than half of the inputs are at logical '1'. Arrangements must then be made to apply the correct syndrome bits to the inputs of the majority gate according which received

bit is being decoded.

The description of this process is facilitated if a precise description of the parity check equations is formulated. In order to do this the rows and columns of the S x S array are numbered 0 to S - 1. Each information bit in the square array is now labelled by the row, r, and the column, C, in which it lies as k(r,C). Those information bits in the same parity check equation, within each direction set D_i, are then taken to be those where the values of a function, D_i (r,C), are equal.

Example 1

Define D_1 (r,C) = r for $0 \leqslant r \leqslant S - 1$ then D_1 (r,C) takes the same value for all information bits lying in the same row. The parity check equations are therefore those checking rows.

Example 2

Define D_2 (r,C) = r + C for $0 \leqslant r \leqslant S - 1$ and $0 \leqslant C \leqslant S - 1$, then D_2 (r,C) takes the same value for all information bits lying on the same diagonal - the parity check equations are therefore those checking on a diagonal.

Now the parity check bits may be labelled by the values taken by the equation describing function D_i. For example, in the cases given in the above examples $D_1 = 3$ would describe the parity check equation involving those bits in the third row. To avoid confusion between the equations and the check bits, the check bits will be described as $P(D_i = x)$. In a similar manner, the syndrome bits, obtained by modulo -2 adding the received parity check bits to the recalculated parity check bits will be described as $S(D_i = x)$.

The decoding procedure may now be described. It is assumed the information bits are received in row by row order. That is, k(0,0) is received first, followed by k(0,1) and so on to the last information bit, k(S - 1, S - 1).

As the received bits are clocked in, two binary counters are incremented. The first is a column counter, which operates modulo S, and the second a row counter which also operates modulo S. The state of the counters at any time will therefore give the row and the column to which the received information bit belongs.

A total of (d - 1) binary arithmetic units now calculate the value of $D_i(r,C)$ for each direction set, and hence the identity of the (d - 1) parity check equations to which the received bit belongs. The results in binary form are used to specify the store locations in a memory, in which are kept running modulo -2 totals of received bits relating to the parity check equations they represent.

Hence, all of the information bits have been received, the store locations contain the recalculated parity check bits required as the first step in the calculation of the syndrome.

At the same time as these parity checks are recalculated, the received bits are shifted into a dynamic buffer shift register of length equal to the block length of the codeword, and continue to be shifted as the remaining, parity check bits of the received word are clocked into the decoder.

As the received check bits are clocked in they are modulo -2 added to the contents of the store locations of the respective re-calculated parity check bits - thus the contents of the memory are now the syndrome of the received word, and the received word is about to emerge from the dynamic shift register.

At this stage, the two binary counters are reset to zero, and, as the received bits are clocked out of the register, are incremented as before. Again the (d - 1) binary arithmetic units calculate the value of $D_i(r,C)$ for each, now emerging, received information bit. These values are now the locations of the syndrome bits relating to this information bit. The values are therefore used to access these

bits from the memory, where they are applied to the imputs of a(d - 1) input majority gate. Iff greater than (d - 1)/2 of these imputs are of logical '1' then the output is at '1'. Therefore by modulo 2 adding the output of this gate to the emerging received information bit the errors are corrected and the corrected bit may then be delivered to the sink.

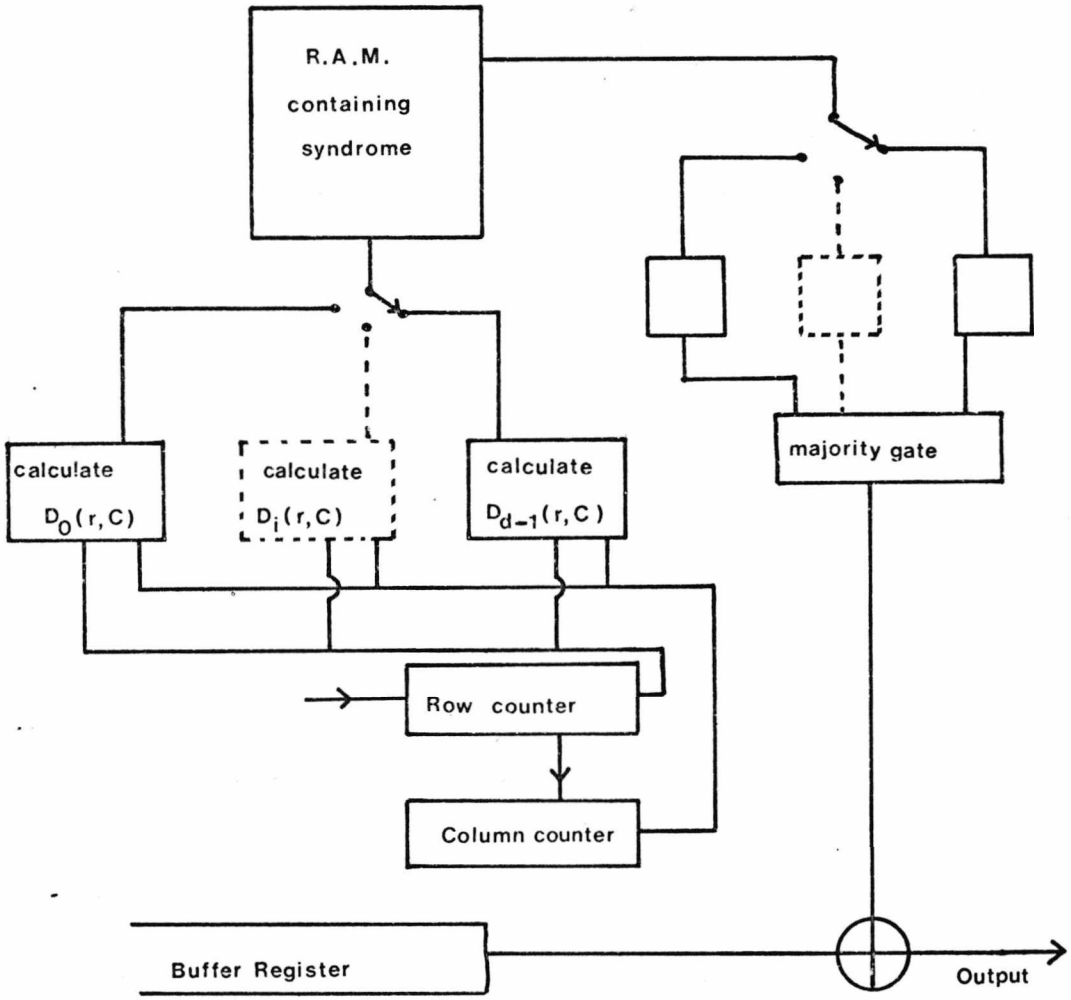
A block diagram of this process is shown in Fig. 7.3. Also given are the expressions for $D_i(r,C)$ for codes of minimum distance up to 11; the extension to codes of minimum distance greater than this is obvious.

The hardware requirements for such a decoder comprise mainly an n-bit serial register, two binary counters, (d - 1) hard wired arithmetic units, and sufficient random access storage for the syndrome bits, together with the one majority gate with (d - 1) inputs.

The major complication in the decoder is the routing of the received parity check bits to the correct recalculated parity check bit locations. By transmitting the check bits in the correct order, however, they may be added modulo -2 to the storage locations in a straightforward sequential manner. That is, the first received parity check bit would relate to $S(D_o = 0)$ the next to $S(D_o = 1)$ and so on to $S(D_{d-2}$ = maximum value attained).

The decoding process may be improved in various ways common to all majority logic decodable codes. Feedback of the decoding information to the syndrome of a received word may be applied simply by inverting not only the emerging received bit when an error is calculated, but also the respective syndrome bits, the locations of which are of course those of the bits applied to the inputs of the majority gate. The decoding may also be carried out using a variable threshold majority gate, where the threshold is variable from (d + 1)/2 to (d - 1). An attempt to decode a received word is made first with the threshold set to (d - 1).

The syndrome feedback technique described is also employed. If no corrections are made then the threshold is lowered by one otherwise the decoding cycle is repeated until a cycle is executed with no corrections



$D_i(r,C) = r$

$D_i(r,C) = C$

$D_i(r,C) = r+C$

$D_i(r,C) = S-(r+C)$

$D_i(r,C) = 2r+C$

$D_i(r,C) = 2C+r$

$D_i(r,C) = S-(2r+C)$

$D_i(r,C) = S-(C+2r)$

$D_i(r,C) = S-(2C-r)$

$D_i(r,C) = S-(C-2r)$

made, and then the threshold lowered by one. Another decoding cycle is then executed. If corrections are made then the threshold is increased by one, after which it is again lowered by one regardless of whether or not corrections were made. Otherwise, if no correction is made in the decoding cycle then the threshold is again lowered by one and another decoding cycle made. This process is continued until the threshold reaches $(d + 1)/2$ at which point decoding is considered complete and the information bits delivered to sink. If, after $2(d - 1)$ complete decoding cycles, the threshold has not reached $(d + 1)/2$ then it is assumed that the threshold is ocsillating between two levels, and so the threshold is forced to $(d + 1)/2$ and a final decoding cycle made, after which the decoded information bits are delivered to sink.

This modification to the decoding algorithm corrects many more errors of weight greater than $(d - 1)/2$, but at the expense of a considerable increase in decoding time - requiring approximately $n + (2(d - 1)+1)k$ shifts per decoded block, compared to about $n + k$ shifts of the original, fixed threshold version. Even for double-error-correcting codes this is nearly a tenfold increase.

Nevertheless, the decoding processes for array codes lend themselves very well to simulation by computer, using a high level programming language. Moreover they could be easily impremented using a microprocesser.

7.6. Decoding Shortened Array Codes of Construction A

The basic procedure for decoding the shortened codes is clearly the same as above. The process may be modified in two possible ways, in order to deal with the shortened case.

Firstly, dummy (zero) information bits could be inserted at the receiver, before they enter the decoder. Secondly, the counters in the decoder could be hand wire programmed to "skip" those states relating to

the missing information bits. Neither modification is particularly complex.

7.7. Performance of the Codes

In order to check the performance of the array codes, decoders for certain examples of the codes with minimum distance 5 have been simulated on a computer, using the BASIC programming language. After observing that the codes do, indeed, correct all errors of weight less than three, the proportion of errors of weight 3, 4, and 5 which are corrected was found for the random error case; that is, where the errors have a binomial distribution.

Fixed threshold decoding schemes have been simulated, and results are shown for the algorithm with and without feedback to the syndrome. The summary of results is in Table 7.5.

7.8. Contruction B

In this construction, consideration is taken of the fact that the lines representing parity check equations, which pass through the array of information bits, do not have to be straight. The only requirement is that two lines pass through more than one common point.

As a result of this consideration, the parity check equations which in construction A codes are inefficient, because they check relatively few information bits, may be made more efficient by including more information bits without relaxing the self-orthogonal requirement.

To deomonstrate that this is possible, consider the code with sixteen information bits and minimum distance four, described by straight lines passing through the rows, columns, and diagonals of a square array as shown in Fig. 7.4.

TABLE 7.5. - Decoding Results (Blocks in error/no. of blocks tested)

3 Errors

With feedback :

45,25	augmented 4 x 5	263/1000
77,51	augmented 5 x 8	249/1000
110,82	augmented 8 x 8	211/1000
45,20	unaugmented 4 x 5	90/1000
77,40	unaugmented 5 x 8	109/1000
110,64	unaugmented 8 x 8	96/1000

No feedback :

augmented 4 x 5	436/1000
augmented 5 x 8	610/1000
augmented 8 x 8	431,1000
unaugmented 4 x 5	271/1000
unaugmented 5 x 8	260/1000
unaugmented 8 x 8	300/1000

4 Errors

With feedback :

42,25	augmented 4 x 5	743/1000
77,51	augmented 5 x 8	870/1000
110,82	augmented 8 x 8	853/1000
45,20	unaugmented 4 x 5	330/1000
77,40	unaugmented 5 x 8	294/1000
110,64	unaugmented 8 x 8	303/1000

No feedback :

45,25	augmented 4 x 5	808/1000
77,71	augmented 5 x 8	894/1000
110,82	augmented 8 x 8	873/1000
45,25	unaugmented 4 x 5	586/1000
77,51	unaugmented 5 x 8	613/1000
110,82	unaugmented 8 x 8	605/1000

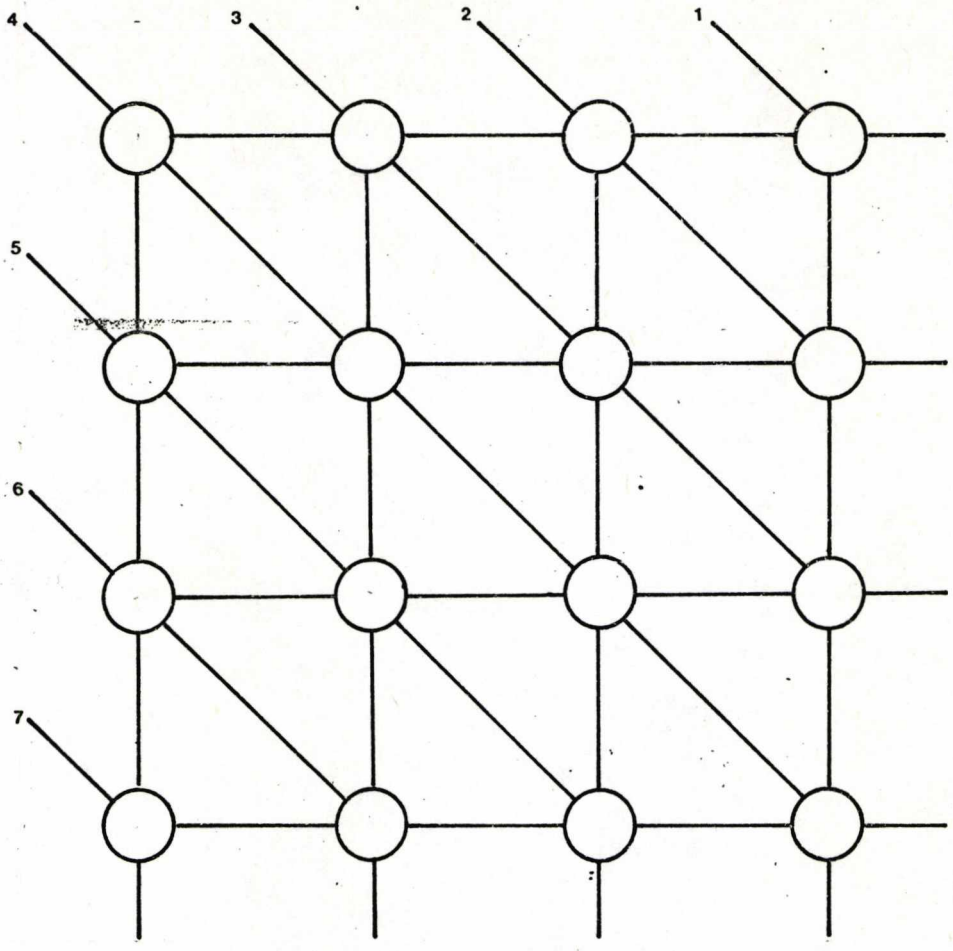


Fig 7.4.

The diagonal lines are numbered one to seven. If lines one and five, two and six, and three and seven are combined, as shown in Fig. 7.5, then it can be seen that the arrangement still has the required property that no two points lie together on more than one line. Thus a $(31,16)d = 4$ code has been improved in rate to a $(28,16)d = 4$ code.

It is not easy, however, to see how extra lines might be economically added to the diagram in order to increase the minimum distance of the code represented. A set of lines in the other diagonal direction would not be suitable, since then some pairs of points would lie both on one of these diagonals and on a "combined diagonal". Neither would

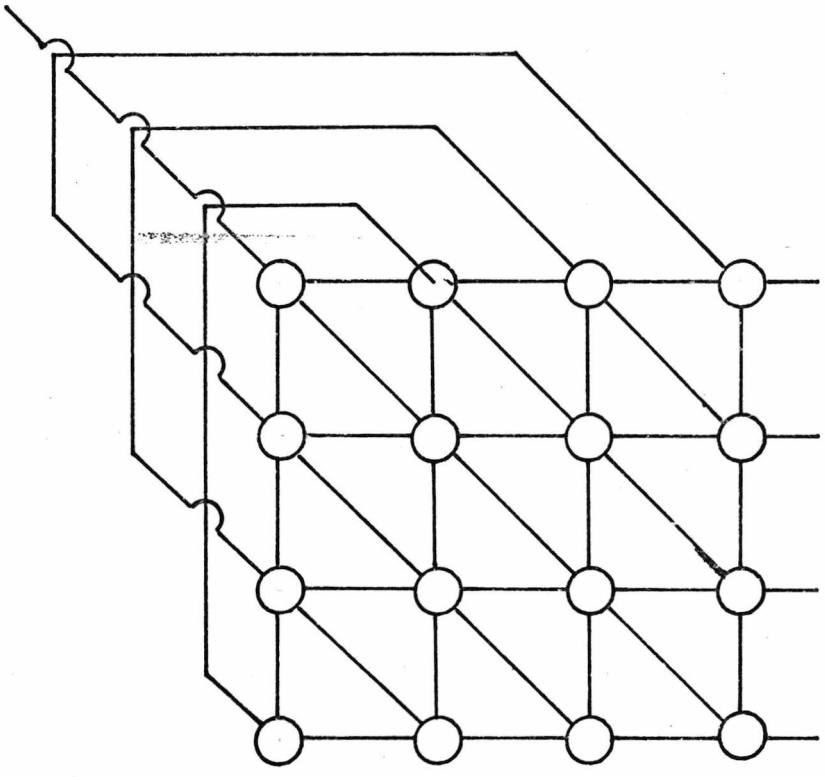


Fig 7.5.

a "Knights Move" set of lines be adequate. Clearly an intuitive approach to the choice of line sets is not effective.

In order to describe a set of self-orthogonal codes with rate improved over that of construction A, the numerical description of parity check equations and information bits, which was described earlier, is again used. Applying this to the above example we have the arrangement of Fig. 7.6.

$D_1 = 0$ $D_2 = 0$ $D_0 = 0$	$D_1 = 0$ $D_2 = 1$ $D_0 = 1$	$D_1 = 0$ $D_2 = 2$ $D_0 = 2$	$D_1 = 0$ $D_2 = 3$ $D_0 = 3$
$D_1 = 1$ $D_2 = 1$ $D_0 = 0$	$D_1 = 1$ $D_2 = 2$ $D_0 = 1$	$D_1 = 1$ $D_2 = 3$ $D_0 = 2$	$D_1 = 1$ $D_2 = 0$ $D_0 = 3$
$D_1 = 2$ $D_2 = 2$ $D_0 = 0$	$D_1 = 2$ $D_2 = 3$ $D_0 = 1$	$D_1 = 2$ $D_2 = 0$ $D_0 = 2$	$D_1 = 2$ $D_2 = 1$ $D_0 = 3$
$D_1 = 3$ $D_2 = 3$ $D_0 = 0$	$D_1 = 3$ $D_2 = 0$ $D_0 = 1$	$D_1 = 3$ $D_2 = 1$ $D_0 = 2$	$D_1 = 3$ $D_2 = 2$ $D_0 = 3$

FIG 7.6.

Clearly $D_0(r,C)$ has been described as $D_0(r,C) = C$ and $D_1(r,C)$ described as $D_1(r,C) = r$.

It is not difficult to see that $D_2(r,C)$ is described by $D_2(r,C) = (r + C) \bmod 4$.

Armed with the mathematical description of the diagrams representing the array codes, it is possible to give a precise formulation of the requirement that the codes be self-orthogonal. Since it is required that no two information bits are together in more than one parity check equation we have :

If $D_j(r_1,C_1) = D_j(r_2,C_2)$ then $D_k(r_1,C_1) \neq D_k(r_2,C_2)$ $\quad j \neq k$.

The codes of construction B, see also Smith (1977), will now be

formulated, as codes with $D_i = (r + iC) \bmod S$ - $i < S$ where S^2 is the number of information bits in the square array.

For a code of minimum distance d, i must take (d - 1) values $0 \leqslant i \leqslant d - 2$.

Therefore, for the construction to be valid, i.e. $i < S$, it is necessary that

$$d - 2 \leqslant S - 1$$

i.e. $S \geqslant d - 1$

It will now be shown that the codes of construction B are self-orthogonal.

It must be shown that if $D_i\ (r,C) = (r + iC) \pmod S$ then when

$D_j\ (r_1,C_1) = D_j\ (r_2,C_2)$ - (1)

$D_k\ (r_1,C_1) \neq D_k\ (r_2,C_2)$ - (2)

for all $j \neq k$ given $j, k < S$

now, (1) implies :

$$(r_1 + jC_1) \pmod S \equiv (r_2 + jC_2) \bmod S)$$

$$(r_2 - r_1) \ (\text{mos } S) \equiv (j(C_1 - C_2)) \pmod S$$

Similarly (2) implies :

$$(r_2 - r_1) \pmod S \equiv (k(C_1 - C_2)) \pmod S$$

hence, since $r_1 = r_2$, $C_1 = C_2$,

$$j \pmod S \equiv k \pmod S$$

i.e., for $j, k < S$, $j = k$. This completes the proof of self-orthogenality. Note that if S were not prime, the calculations would be invalid.

As shown above, the number of parity check bits is given by the number of values taken by the $D_i\ (r,C)$. To enumerate this consider a column C of the array. Then $D_i\ (r,C) = (r + iC) \pmod S$ where iC is a constant within the column. Hence, since r takes all values 0 to s - 1, $D_i\ (r,C)$ must take all values 0 to S -1. This is true of every column. Therefore it is clear that all $D_i\ (r,C)$ take just S values.

There are $(d - 1)$ of the D_i (r,C) and each takes S values, hence the number of parity check bits on S^2 information bits is $S(d - 1)$. The parameters of the codes of construction B are threfore :

$$n = S^2 + S(d - 1) \text{ and } k = S^2$$

from which it may easily be deduced that :

$$n = \frac{(d - 1)^2 R}{(1 - R)^2}$$

which compares well with the best that can be achieved with self-orthogonal codes, which is shown by Townsend and Weldon (1967) to be

$$n \geqslant \frac{R(d - 1)(d - 2)}{(1 - R)^2} + \frac{1}{1 - R}$$

In fact, for codes of rate greater than one half and minimum distance greater than seven, the codes are generally shorter than equivalent rate self-orthogonal quasi-cyclic (s.o.q.c.) codes (Townsend and Weldon (1967)) of the same minimum distance. Note also that, with a knowledge of the prime numbers, codes with construction B are entirely constructive, whereas s.o.q.c. codes are in very many cases found only by a computer search.

Example

Choose $d = 8$, $S = 29$

then $n = (29)^2 + 7 \times 29 = 1047$

and $k = (29)^2 = 841$

This code has rate 0.803 and may be compared with the s.o.q.c. code of minimum distance 8 and rate 0.8 which has a length of 1205.

Note that the requirement, derived above, that $S \geqslant d - 1$, together with the equality $R = \frac{k}{n} = \frac{S^2}{S(d - 1) + S^2}$

gives $R > 0.5$

and therefore the codes may not be constructed for low rates.

A table of selected codes of construction B is given in Table 7.6

TABLE 7.6

UNAUGMENTED CONSTRUCTION B CODES COMPARED WITH

SELF-ORTHOGONAL QUASI-CYCLIC CODES

CONSTRUCTION B					SELF-ORTHOGONAL QUASI-CYCLIC			
n	k	d min	R		n	k	d min	R
45	25	5	0.56		26	13	5	0.5
77	49	5	0.64		78	52	5	0.67
165	121	5	0.73		156	117	5	0.75
357	289	5	0.81		265	212	5	0.8
437	361	5	0.83		390	325	5	0.83
55	25	7	0.45		62	31	7	0.5
187	121	7	0.65		201	134	7	0.67
391	289	7	0.74		420	315	7	0.79
667	529	7	0.80		695	556	7	0.8
1015	841	7	0.83		936	780	7	0.83
105	49	9	0.47		114	57	9	0.5
425	289	9	0.68		420	280	9	0.67
713	529	9	0.74		836	627	9	0.75
1209	961	9	0.80		1460	1168	9	0.8
231	121	11	0.52		182	91	11	0.5
759	529	11	0.70		681	454	11	0.67
1271	961	11	0.76		1380	1035	11	0.75
253	121	13	0.49		266	133	13	0.5
805	529	13	0.66		1041	694	13	0.67
351	169	15	0.48		366	183	15	0.5

s.o.q.c. codes of similar rates and minimum distances are also listed for comparison.

7.9. Decoding Codes of Construction B

Codes of construction B are, of course, one-step majority logic decodable by the same principles used for codes of construction A. The same improvements to the algorithm, feedback to the syndrome and variable threshold decoding, are applicable to construction B codes. In the case of these codes, however, the decoding algorithm is particularly simple, owing to the following two properties :

(a) $D_i\ (r + 1, C) \equiv (1 + D_i(r,C))\ (\text{mod } S)$

(b) $D_i\ (0, C + 1) \equiv (1 + i + D_i(S - 1, C))\ (\text{mod } S)$

These properties may easily be verified by expansion of the terms of the equations.

To see how these properties facilitiate decoding, consider received information bits being input to the decoder in column by column order. By property (a) the value of D_i for a particular bit within each column, for a particular bit, is given by incrementing by one, modulo S, the value of D_i for the preceeding bit. That is to say that the parity check equations to which information bits, in each column, belong are arranged in numerical order, modulo S. Hence the syndrome may be calculated by storing the $S(D_i(r,C))$ in a feedback shift register for each column, and modulo 2 adding the received bits to the contents as the information bits are input, and finally the received parity check bits, giving then the syndrome bits in the register. The arrangement is shown in Fig. 7.7.

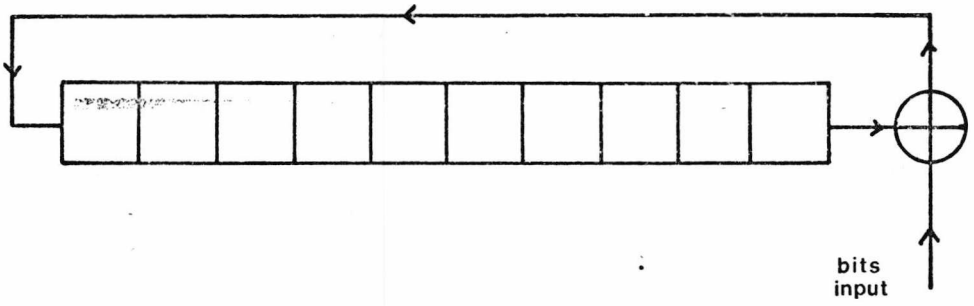


Fig 7.7

Despite the convenient modular arrangement of the D_i for each column, when the received bits change columns there is a discontinuity in the progression of values of the D_i. The size of the discontinuity is dependent upon the value of i, and is given by property (b). In order that the shift register contents keep in step with the received bits it is therefore necessary for the registers to be shifted by the required number of bits between receiving a bit from the end of one column and that from the beginning of the next. The number of shifts required is given by (b) as (1 + i), and clearly when i S/2 an improvement in time required is given by shifting backwards S - (1 + i) rather than forwards by 1 + i.

Alternatively the syndrome registers may be constructed in such a way that the received bits may enter the register at a variable point in the register, as shown in Fig. 7.8 for the D_2 register for a code with S = 5. The entrance point for the information bits is altered at the end of each column.

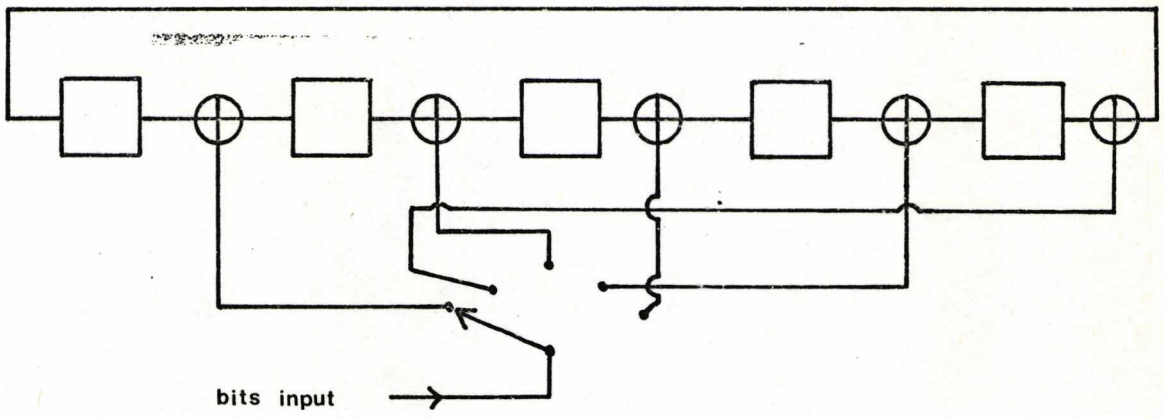


Fig 7.8

The same properties enable the same arrangement of shift registers and taps to be used to assess the correct (d - 1) syndrome bits for the decoding of each information bits as they emerge from the storage registers after the syndrome has been formed.

7.10. Augmentation of Codes of Construction B

Since the parity check bits of codes of construction B are again arranged in direction sets, within which they are independent, the codes

may be augmented in an identical manner to the augmentation of codes of construction A. Again the complexity of the decoding is little more than the sum of the complexities of the decoders for the constituent codes.

A list of augmented codes of construction B is given in Table 7.7. Note that if desired the augmenting codes could in many cases be array codes.

7.11. Construction C

These codes are essentially the same as construction B codes, but are defined with arrays of side S^x where x is any positive integer and S is prime. The D_i are again defined as $d_i(r,C) = r + iC$, but in this case the arithmetic is over GF (S^x) which, in the case of $x \neq 1$, is not modulo arithmetic.

As a consequence, the construction C codes may be defined over a wider range of values of k, but are not so easily decodable, since the property of convenient ordering of values of $D_i(r,C)$ is not held by these codes. Nevertheless if the codes are to be "parallel decoded", that is each information bit decoded at the same time, then these codes may well be attractive, since they usually have, for $t > 2$, a better rate than comparable s.o.q.c. codes. Also, as in previous constructions, the codes may be augmented by modulo -2 adding an augmenting codeword to each direction set of parity checks. (See also Smith, 1977).

A table of codes of construction C, together with some s.o.q.c. codes for comparison, is given in Table 7.8. and a list of augmented codes in Table 7.9.

Proof of the properties of these codes follows through in an identical manner to that for codes of construction B, save that the arithmetic operations are made over GF (S^x) and not modulo S.

TABLE 7.7

AUGMENTED CODES OF CONSTRUCTION B

n	k unaug	d min	na, ka	k aug	R aug
45	25	5	5,1	29	0.64
162	121	5	11,4	137	0.85
357	289	5	17,9	325	0.91
187	121	7	11,2	133	0.71
667	529	7	23,12	601	0.90
1015	841	7	29,14	925	0.91
425	289	9	17,3	313	0.74
1209	961	9	31,11	1049	0.87
231	121	11	11,1	131	0.56
1271	961	11	31,11	1071	0.84
805	529	13	23,2	553	0.69

n = Length of code.

k unaug = No. of information bits in unaugmented codeword.

d min = Minimum distance of code.

Na = Length of augmenting codewords.

ka = No. of information bits in augmenting codewords.

k aug = No. of information bits in augmented codeword.

R aug = Rate of augmented codeword.

TABLE 7.8

CODES OF CONSTRUCTION C COMPARED WITH

SELF-ORTHOGONAL QUASI-CYCLIC CODES

CONSTRUCTION C					SELF-ORTHOGONAL QUASI-CYCLIC			
n	k	d min	Rate		n	k	d min	Rate
32	16	5	0.5		26	13	5	0.5
96	64	5	0.67		78	52	5	0.67
320	256	5	0.8		265	212	5	0.8
775	625	7	0.81		695	556	7	0.8
128	64	9	0.5		114	57	9	0.5
384	256	9	0.67		420	280	9	0.67
825	625	9	0.76		836	627	9	0.75
1280	1024	9	0.8		1460	1168	9	0.8
171	81	11	0.47		182	91	11	0.5
1344	1024	11	0.76		1380	1035	11	0.75
925	625	13	0.68		1041	694	13	0.67
480	256	15	0.53		366	183	15	0.5

TABLE 7.9

AUGMENTED CODES OF CONSTRUCTION C

n	k unaug	d min	na, ka	k aug	R aug
96	64	5	8,2	72	0.75
320	256	5	16,8	288	0.9
775	625	7	25,12	697	0.90
384	256	9	16,2	272	0.71
825	625	9	25,8	689	0.84
1280	1024	9	32,11	1112	0.90
1344	1024	11	32,11	1134	0.84
925	625	13	25,3	661	0.71
480	256	15	16,1	270	0.56

n = Length of code.

k unaug = No. of information bits in unaugmented codeword.

d min = Minimum distance of code.

Na = Length of augmenting codewords.

Ka = No. of information bits in augmenting codewords.

k aug = No. of information bits in augmented code.

R aug = Rate of augmented code.

7.12. Construction 'D' Codes

7.12.1. Introduction

By the addition of overall parity check bits (i.e. parity checks on all information bits), the minimum distances of codes of Construction 'A' can be increased. The resultant codes are of considerably higher efficiency than those of Construction 'A'; but with the exception of the double-error-correcting case, the exact minimum distances of the codes has not been proved nor a practical decoding scheme found for them. Nevertheless, a proof of the minimum distance of the codes has been found which relies upon the truth of a strong geometrical conjucture, and the codes contain considerable mathematical structure. Therefore, since it is possible that a simple decoding algorithm might be found, possibly based upon a proof of the above mentioned conjecture, the general construction of the codes is described here together with a proof of the minimum distance of, and a description of a simple decoding algorithm for, the double-error-correcting case.

7.12.2. Construction of the Codes

The codes of Construction 'D' are formed by simply annexing overall parity check bits to the codewords of codes of Construction 'A'. The number of additional parity check bits to annex is given as $(Q - 1)$ where Q is the number of direction sets comprising the parity check bits of the Construction 'A' code. It will be recalled that a Construction 'A' code with minimum distance d_A has parity check bits composed of $d_A - 1$ direction sets.

It is conjectured here that a code constructed in such a way has minimum distance $d_A + (Q-1)$. Thus for example a construction 'A' code of minimum distance 4 would become, with the addition of a single overall parity check bit, a Construction 'D' code of minimum distance 5. A Construction 'A' code of minimum distance 5 would become a Construction 'D' code with minimum distance 7 by the addition of two overall parity check bits. Trivially, a Construction 'A' code of minimum distance 3 will become a minimum distance 4 Construction 'D' code by the addition of one overall parity check bit - this Construction 'D' code is equivalent to the product code formed by the product of two single parity check codes.

That the construction method described above is valid for the double-error-correcting case will now be proved, followed by a description of a simple decoding algorithm for this case. The validity of the construction method for codes of greater minimum distance will then be discussed.

7.12.3 Double Error Correcting Codes of Construction 'D'

It will be proved here that by annexing two overall parity check bits to the codewords from Construction 'A' codes of minimum distance four, double error correcting Construction 'D' codes are obtained. Consider first the general case of systematic, linear, self-orthogonal block codes.

It will be recalled that a self-orthogonal code is one in which not two information bits are found together in more than one parity check equation. Also, since the

codes are linear, a codeword with an information vector of weight w is given by the modulo -2 sum of the codewords formed by encoding each of those w vector of weight one whose own modulo -2 sum is the information vector. This latter statement is equivalent to the remark that a systematic linear codebook is the rowspace of a reduced echelon form of generator matric. A further, obvious, property of the codes is that a codeword formed by encoding an information vector of weight one will have minimum weight d, the minimum distance of the code.

From the above properties it can be seen that parity checks formed by encoding two different information vectors of weight one will have "ones" in at most one common position. Therefore an information vector of weight two will have a minimum weight of (2d - 2) which in the case of d = 4 is equal to six.

As an example of this situation, consider the code having a generator matric, G, given by

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

the encoding of the information vector 0 1 1 0 is clearly equal to the sum of the two codewords

0 1 0 0 1 0 0 1 1 0

↕——(ones in common position)

and 0 0 1 0 0 1 0 1 0 1

which gives 0 1 1 0 1 1 0 0 1 1 which has weight 6.

In a similar way to the above, the minimum weight of codewords having information vectors of weight 3 can be deduced. It can be seen that a codeword having an information vector of weight one will, in all but at most two positions, have parity checks which are 'one' in differing positions. For example, consider the codeword 1 1 0 0 0 1 1 1 1 0 from the above code and another : 0 0 0 1 0 0 1 0 1 1. The 'ones' in the parity check bits coincide only in the third and fifth positions. The sum of these codewords therefore gives a codeword, 1 1 0 1 0 1 0 1 0 1, which is of weight six. It can be seen that this is the minimum weight possible for a codeword with information bits of weight three, from a code of minimum distance four.

It is clear that a codeword with an information vector of weight four, when it is from a linear code of minimum distance four, will have a weight of at least four. Also, that a codeword with information bits of weight five will have weight at least five.

Now consider the effect of adding an overall parity check to the above codewords. An overall parity check will be a 'one' if the weight of the information vector is odd. Therefore codewords with information vectors of odd weight will themselves be increased in weight by one. Thus the minimum weights of the new codewords will be as given in Table 7.10.

Weight of Information Vector	Minimum Weight of New Codeword
1	5
2	6
3	10
4	4
5	6

Table 7.10

Therefore, if a self orthogonal code of weight four extended in the above manner is to have minimum distance five it is sufficient that codewords with information vectors of weight four have minimum weight five. It will now be shown that self-orthogonal array codes of construction 'A' fulfil this requirement. It will be remembered that a Construction 'A' code with minimum distance four has checks in three directions across a square array of information bits. In order that all parity checks be zero it is therefore necessary that all directional checks meet on even number of non-zero information bits.

Consider first a single non-zero information bit on the array, as in Fig. 7.9. It is clearly necessary for at least three more information bits to be a 'one' in order that all parity check bits be zero. That is, one more information bit in each direction - row, column, and diagonal must be a 'one'. See Fig. 7.10 for an example.

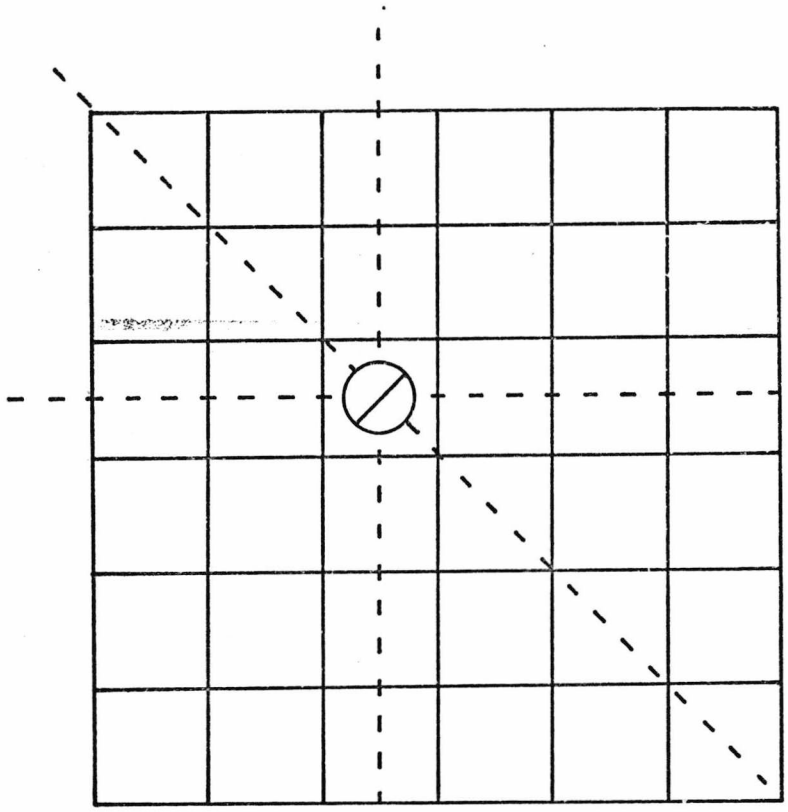


Fig. 7.9.

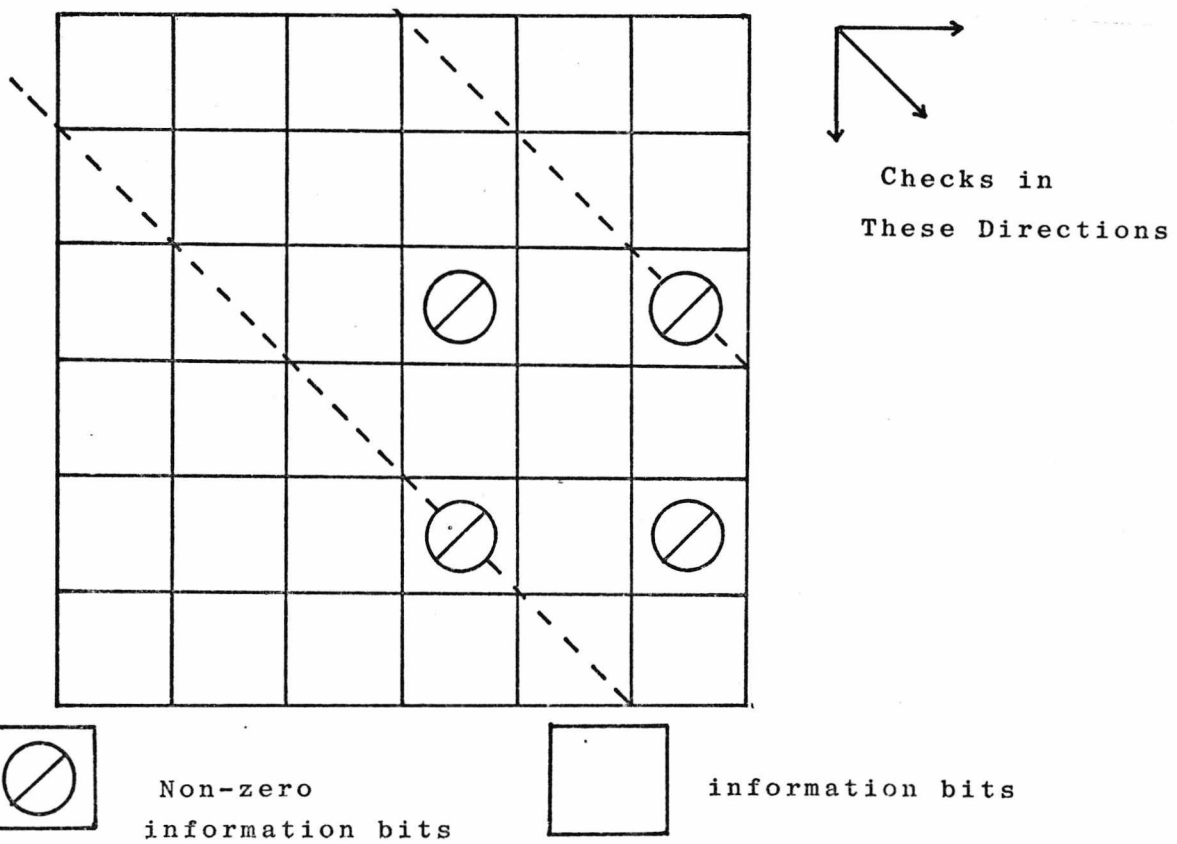


Fig. 7.10.

It can be seen that only one of these extra three bits can have all of its parity checks zero. In the case of each of the remaining two non-zero information bits there must be at lease one 'parity check failure', causing two non-zero parity check bits. Thus it is necessary for at least two more information bits to be zero in order that all parity checks are zero. That six non-zero information bits are sufficient can be seen from Fig. 7.11.

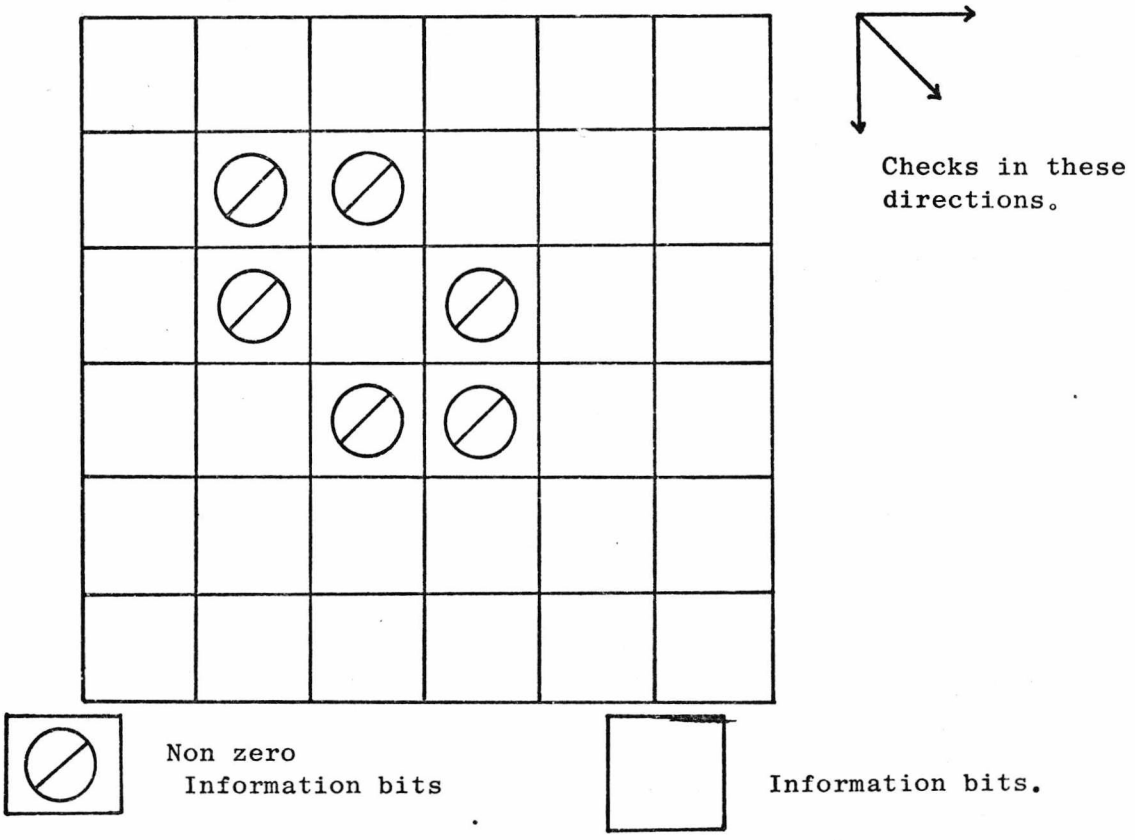


Fig. 7.11.

It has been shown, therefore, that codewords, from array codes of Construction 'A' with minimum distance four, having information vectors of weight four have minimum

weight six.

Consequently it can be seen that the Construction 'D' codes formed by annexing codewords from sum Construction 'A' codes with an overall parity check bit have minimum distance five.

7.12.4. Decoding Codes of Construction 'D'

Described here is a decoding algorithm for double-error-correcting codes of Construction 'D'.

Correctable error patterns in codewords from such a code may be of six basic types.

(i) No errors occur
(ii) One error occurs in the parity check bits.
(iii) Two errors occur in the parity check bits.
(iv) One error occurs in the information bits.
(v) Two errors occur in the information bits.
(vi) One error occurs in the information bits, and one in the parity check bits.

In order to understand the decoding algorithm, consider the parity check failures which occur in each case :

In Case (i) there will be no parity check failures, thus the received sequence is recognised as a codeword, and no correction is necessary :

In Case (ii) one parity check only will fail, that is to say the syndrome of the received sequence will have weight one. It is well known (e.g. Peterson 1972) that the weight of a syndrome is $\leqslant t$ iff all errors have occurred in the parity check bits of a codeword. Thus one parity check failure indicates that no errors have occurred in the information bits of the received sequence :

In Case (iii), as in Case (ii), the syndrome weight will be

$\leq t$, in fact two parity checks will fail. Hence the information bits will be recognised as error free :

In Case (iv) there will be four parity check failures - namely those checks on the column, row, and diagonal in which the error has occurred, and in the overall check bit :

Case (v) may be subdivided into two situations.

Situation (a) The errors do not lie on the same row, column, or diagonal :

Here, there will be six parity check failures - two of the row checks, two of the column checks, and two of the diagonal checks, corresponding to the rows, columns, and diagonals in which the errors have occurred. It can be seen that all the three directional checks on each of the two errors will fail, whereas at most two checks will fail on the other information bits. See Fig. 7.12. for an example of such a situation.

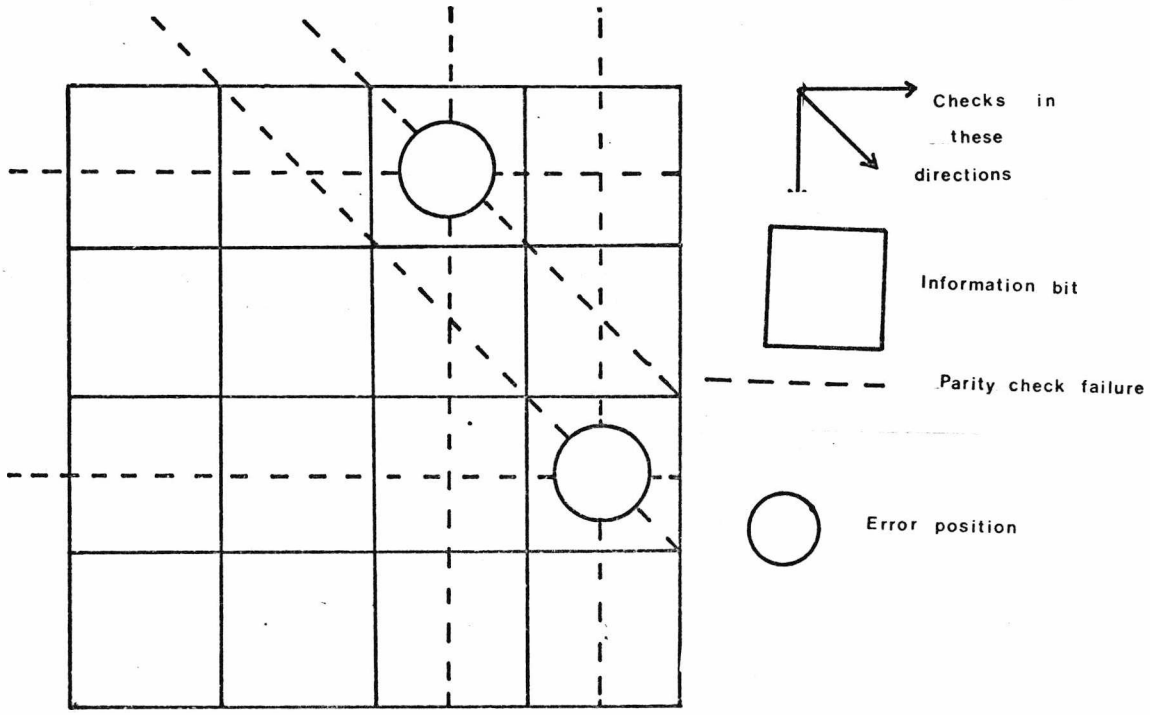


Fig. 7.12. Two errors in information bits, not lying on the same row, column, or diagonal.

Situation (b) Errors lying on the same row, column, or diagonal :

In this case there will be four parity check failures, as can be seen in the example shown in Fig. 7.13. Two of the parity checks on each of the error positions will fail - but also, in general, two parity checks will also fail on each of two other information bits. These bits are marked with an asterisk (*) in Fig. 7.13.

Consider the situation if all of the bits which have two parity check failures are inverted. The two errors will be corrected, but two different errors will be introduced - i.e. in those positions in Fig. 7.13. marked with an asterisk. The new error positions will always be on different rows, columns, and diagonals to one another. This may be seen particularly clearly if the array of squares is

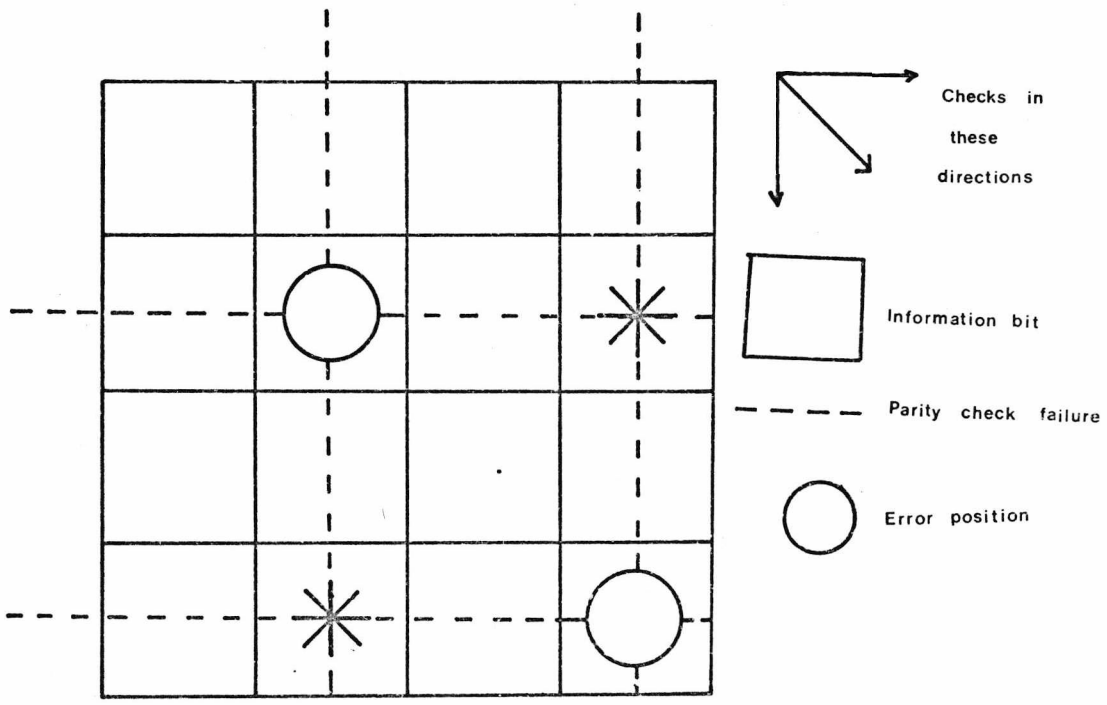


Fig. 7.13 Two errors in information bits, lying on the same diagonal.

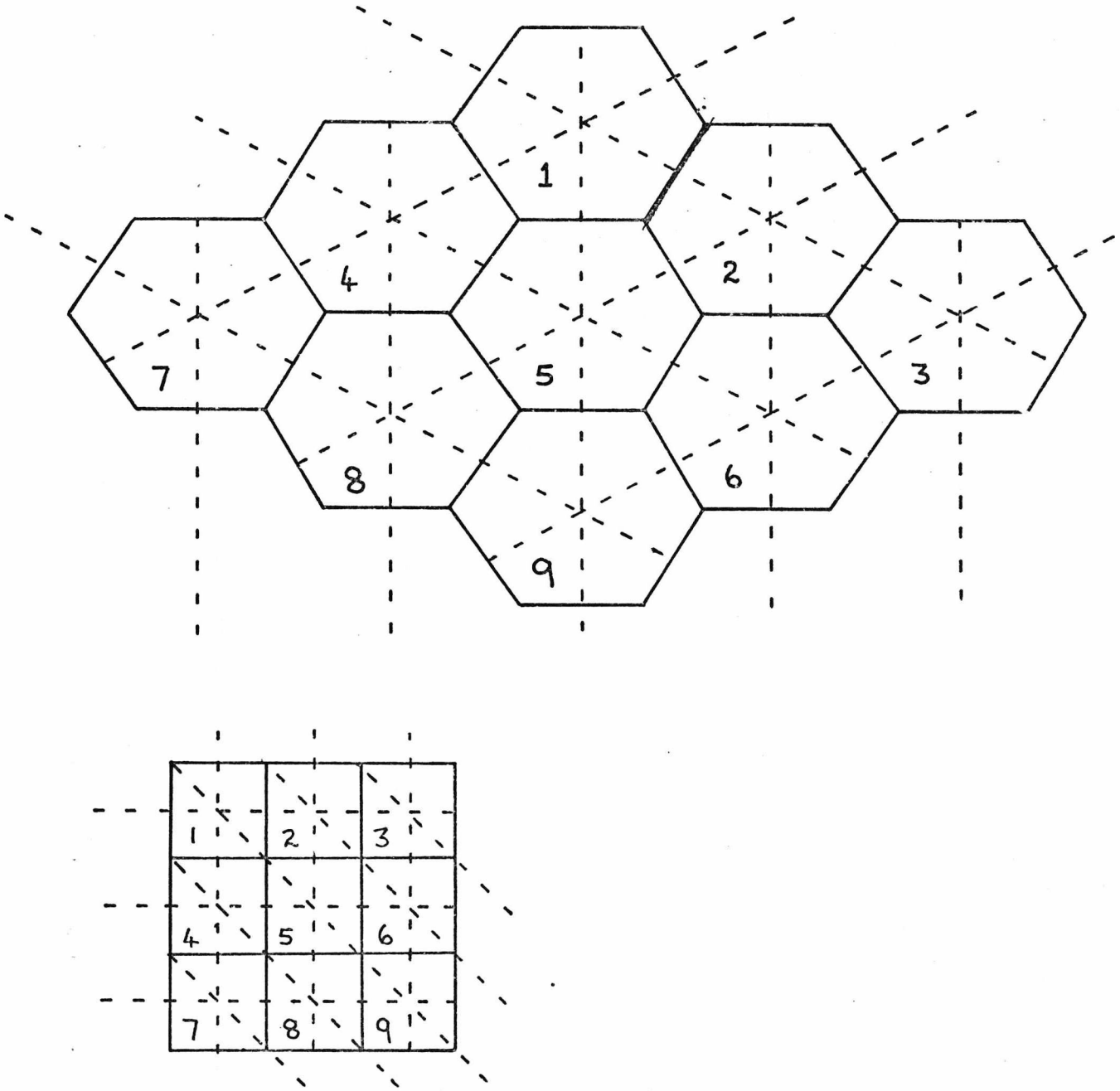


Fig. 7.14. Transformation of square array to hexagonal array

transformed to an array of hexagons as shown for the case of nine information bits in Fig. 7.14. The code remains unchanged, with an identical parity check matrix. However, rows, columns, and diagonals are clearly seen to be indistinguishable from one another, and so consideration of one situation in which two errors are checked by the same parity check bit will cover all possible cases. Such a case is illustrated in Fig. 7.15, and it is easily seen that the two information bits on which two parity checks also fail do not have any parity checks in common, i.e. are in different rows, columns, and diagonals to one another. Thus the new error positions have become of type (a).

In Case (vi) the overall parity check fails, and either two or four other checks. The two situations are illustrated in Figs. 7.13. and 7.15. they are :

(a) The parity check in error checkes the information bit which is in error, i.e. causes a parity check which should fail to succeed.

(b) The parity check bit in error does not check the information bit which is in error.

In Case (a) two of the checks on the erroreous information bit will fail, apart from the overall check. There will be no other information bits on which two parity checks fail (see Fig. 7.16. for an example).

In Case (b) all three checks on the erroreous information bit will fail, but also there will be another parity check failure. It can be seen, however, that no other information will have all of its checks failing, (see Fig. 7.17. for an example.

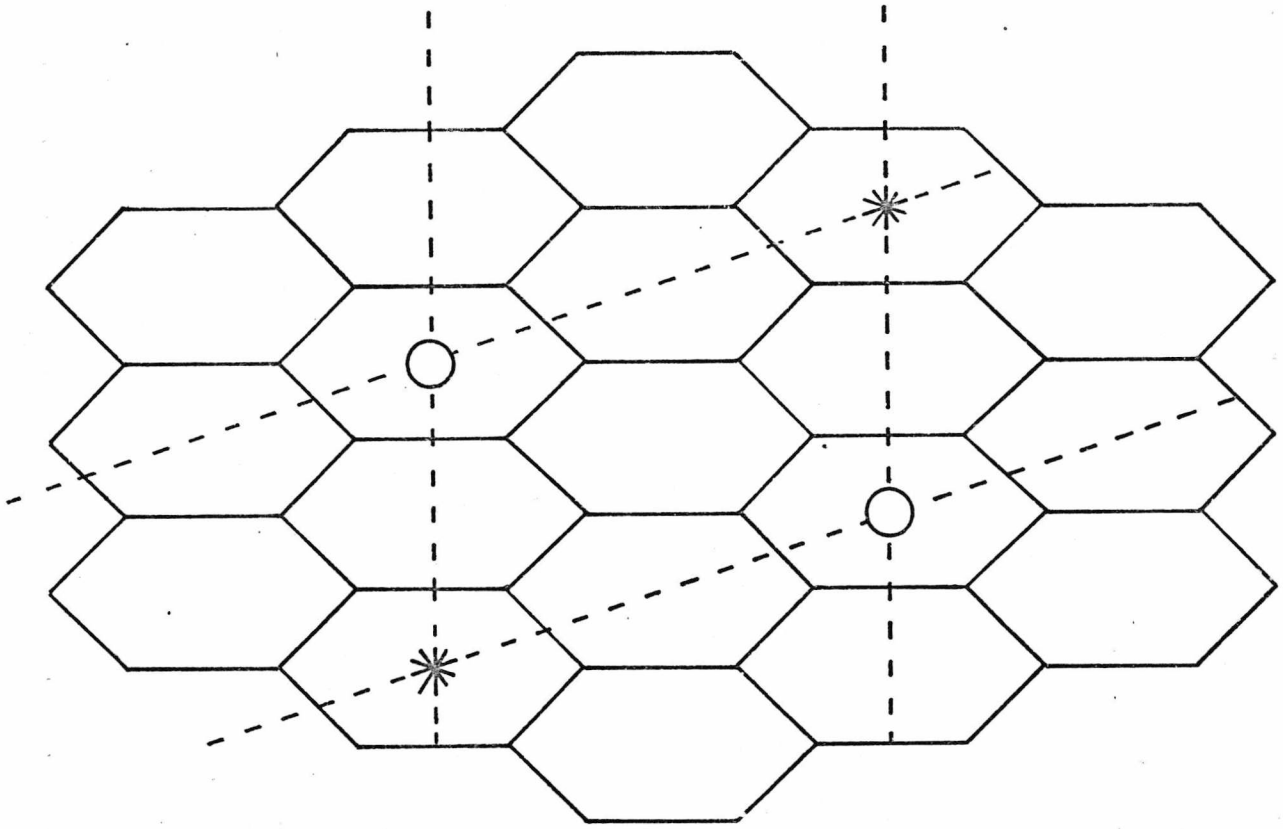


Fig. 7.15 Two information bits in error lying in the same direction line.

From the above considerations, the decoding algorithm may be seen. It may be divided in to 4 steps.

Step 1 - The syndrome is calculated.

Step 2 - If the weight of the syndrome is less than three, the information bits are assumed to be error free, and are output to the sink;

Step 3 - Otherwise the information bits are considered one at a time, and any of which have parity failures on all three directional checks are inverted. If any inversions are made, then the information bits are considered to have been corrected, and are output to the sink;

Step 4 - Otherwise if no inversions are made, the information bits are again considered one at a time, and any of which have parity failures on two of their checks are

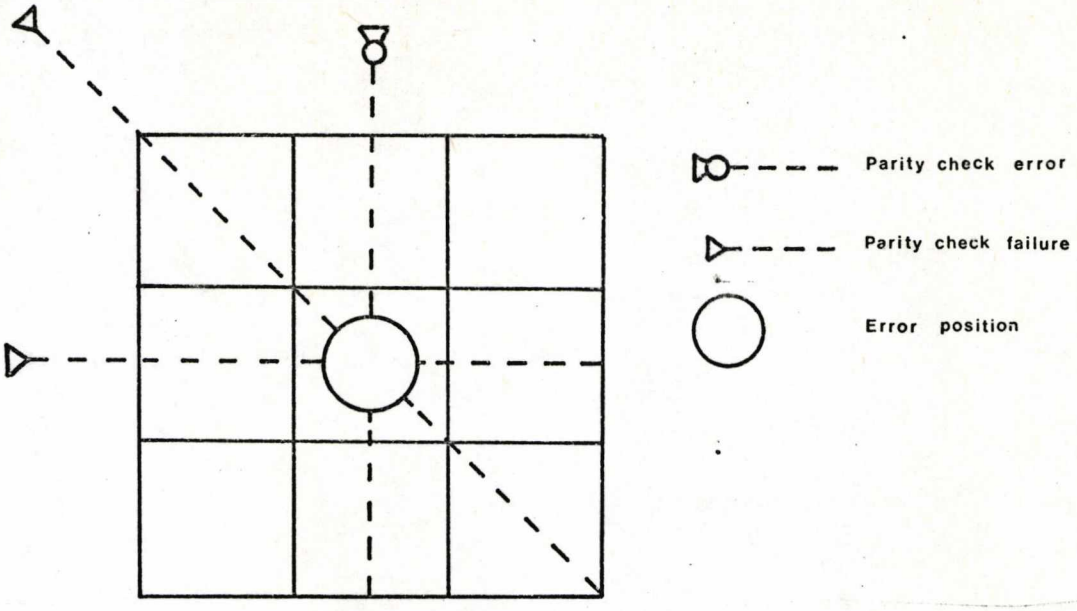


Fig. 7.16

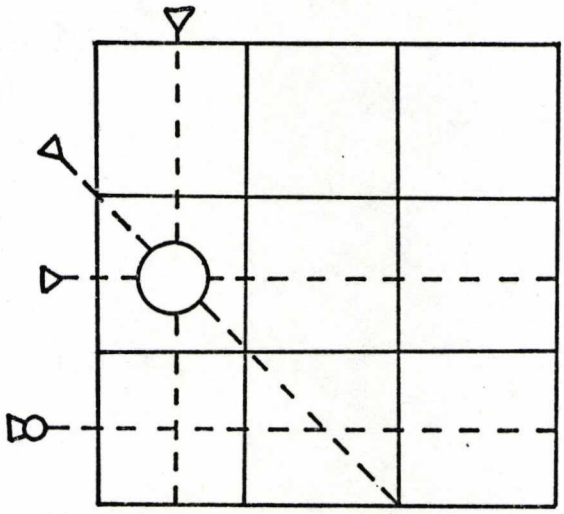


Fig. 7.17

inverted. Then the syndromes are re-calculated and Steps 2 and 3 are carried out again. If, on Step 3 repeated, no inversions are made, then it is assumed that more than two errors have occurred.

Step 2 makes use of the properties of situations (i) and (ii), and (iii), i.e. the weight of the syndrome is at

least three if there are any errors in the information bits.

Step 3 corrects any errors of type (iv), (v)(a); and (vi) (b), whilst leaving errors of type (v)(b) and (vi)(a) unchanged.

Step 4 corrects any errors of type (vi) a and transforms errors of type (v) b to those of type (v) a. Thus repeating Step 2, after re-calculating the syndromes, recognises that errors of type (vi)a have been corrected at Step 4; and repeating Step 3 corrects the errors of type (v) a produced by Step 4.

The algorithm may be recognised as consisting of a stage of variable threshold majority logic decoding preceded by a stage of error trapping. Thus the decoder would be similar to that of the codes of Construction 'A', with the addition of a variable threshold element, the calculation of the overall parity check bit, and the weighing of the syndrome before beginning variable threshold decoding. The threshold is set to 3 at the first decoding run. If no errors are corrected the threshold is set to 2 and a second decoding run is made. Then the syndrome is recalculated, error trapping is attempted, and the threshold set to 3 for a final run. The decoding algorithm is thus seen to be rather less complex than a variable threshold decoder for a code of Construction 'A', and to require less time for decoding. It is, of course, more complex and more timeconsuming in its operation than a fixed threshold decoder for Construction 'A' codes. Moreover, a fixed threshold type of decoder for the Construction

'D' codes is not realisable. Construction 'D' codes are nevertheless more efficient than Construction 'A' codes, or indeed Construction 'B' and 'C' codes, and therefore would be attractive in situations where a high rate is required.

7.12.5 Simulation of the Decoder

The decoding algorithm described above has been simulated on a computer using the BASIC programming language, and the results verify that all double-error patterns are corrected. Also found were the proportion of triple and quadruple error patterns which are incorrectly decoded, for the (45,25) double-error-correcting Construction 'D' code. Of a random sample of one thousand triple error patterns, only 397 were incorrectly decoded, that is approximately 40%. Of five hundred quadruple error patterns, 358 were incorrectly decoded, i.e. about 72%.

Results have also been obtained for a Construction 'D' code formed by the extension of a shortened Construction 'A' code. The construction of the code is shown in Fig. 7.18 , and it is seen to be a (20,8) double-error-correcting code.

With this code 411 of 1600 triple error patterns were not corrected and 795 of 1000 quadruple errors were not corrected.

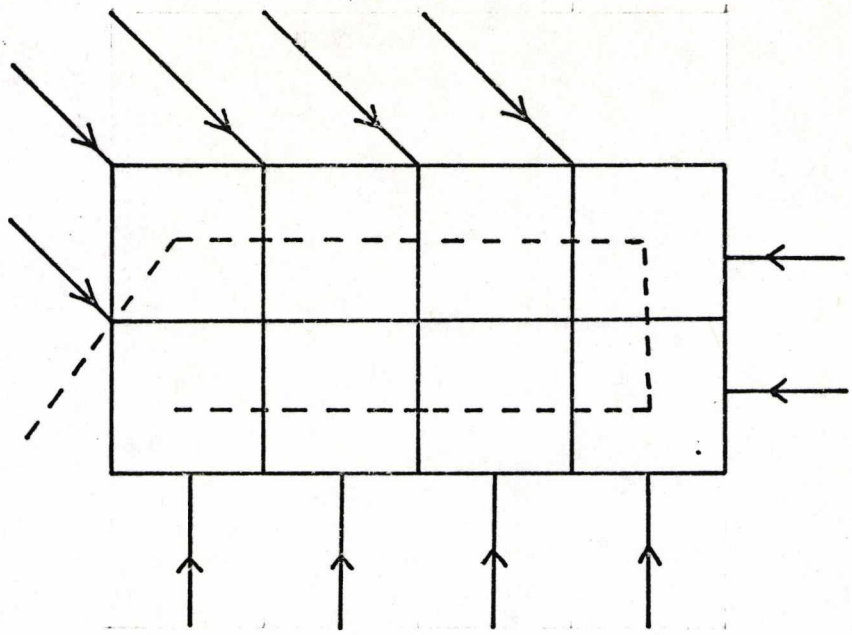


Fig. 7.18. The (20,8) Construction 'D' Double-Error-Correcting Code.

7.12.6. Multi-error correcting construction D codes

For error correction capability greater than two, no proof of this construction method exists. However, a conjecture will be made which, if true, enables a proof to be constructed that by annexing Q-1 parity check bits to construction A codewords the minimum distance of the code is increased to 2Q.

Let w(i) be the weight of a codeword with i non-zero information bits, and f(i) be the number of non-zero parity check bits in that codeword. Then trivially $w(i) = i + f(i)$. If the minimum distance of the code is d then it is necessary that $i + f(i) \geqslant d$.

Therefore, to prove that a code has minimum distance at least d it is sufficient to prove that for that code

$f(i) \geqslant d-i$

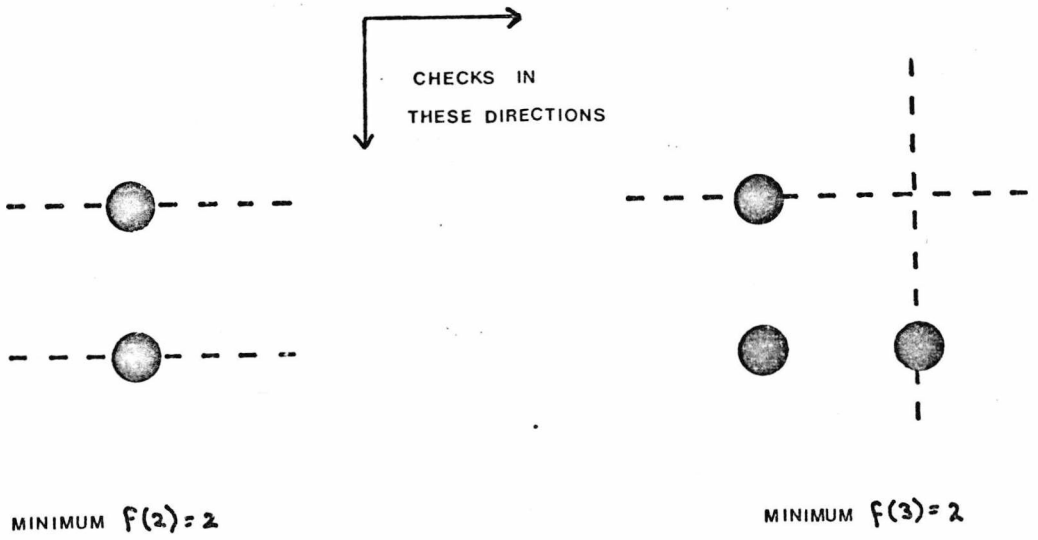


FIG 7.19

This approach will be that adopted for the proof of the minimum distance of construction D codes, based upon a conjecture. The conjecture made is that for construction A codes.

$f(Q+j) \geq 2Q-g(Q-j)$ for $0 \leq j < Q$ where $g(x) = x(d-x)$.

The validity of this conjecture will be discussed later.

Consider f(i) for any self-orthogonal code. It is known that no two information bits are involved together in more than one parity check equation, whilst each information bit is involved in a total of at least (d-1) parity check equations. It may therefore be seen that:

if i=1 $f(i) \geq (d-1)$

if i=2 $f(i) \geq 2(d-2)$

if i=3 $f(i) \geq 3(d-3)$

and in general $f(i) \geq i(d-i)$

With construction A codes $Q + 1 = d$

therefore $f(i) \geq i(Q + 1 - i)$.

and so $w(i) \geq i + i(Q + 1 - i)$

therefore $w(i) \geq i(Q + 2 - i)$

it is easily shown that for $1 < i \leq Q$

$$i(Q + 2 - i) \geq 2Q$$

and therefore $w(i) \geq 2Q$ for $1 < i \leq Q$

Now consider $i = Q + j$

then from the conjecture; for $0 \leq j < Q$

$$f(Q + j) \geq 2Q - (Q + 1 - Q - j)(Q + j)$$

i.e. $w(Q + j) \geq 3Q + j - (1-j)(Q+j)$

then $w(Q + j) \geq 2Q + Qj + j^2$

Therefore, since $2Q + Qj + j^2 > 2Q$ for $0 \leq j < Q$

$w(Q + j) \geq 2Q$ for $0 \leq j < Q$

hence $w(i) \geq 2Q$ for $1 < i < 2Q$

That $w(i) \geq 2Q$ for $i \geq 2Q$ is trivial.

It is known that $w(1) \geq Q + 1$, therefore by the addition of Q-1 overall parity checks to the code, $w(1) \geq 2Q$.

Thus it has been shown that the addition of Q-1 overall parity check bits to a construction A code is sufficient for the minimum distance to be increased from Q + 1 to 2Q, provided that

$f(Q + j) \geq 2Q - g(Q - i)$ for $0 \leq j < Q$

where $g(x) = x(Q + 1 - x)$ and $f(x) \geq g(x)$

Validity of the conjecture

(i) consider the case of Q = 2

$g(1) = 2$

$g(2) = 2$

the conjecture implies that $f(2) \geq 2Q - g(2)$

$f(3) \geq 2Q - g(1)$

i.e. that $f(2) \geq 2$ and $f(3) \geq 2$

fig. 7.19 illustrates arrangement of non-zero information bits for minimum f(2) and f(3), from which the validity of the conjecture may be seen.

(ii) consider the case Q=3

$g(1) = 3 \quad g(2) = 4 \quad g(3) = 3$

the conjecture implies that $f(3) \geq 2Q - g(3) = 3$

$f(4) \geq 2Q - g(2) = 2$

$f(5) \geq 2Q - g(1) = 3$

fig. 7.20 shows the configurations for minimum f (3), f(4), and f(5), and the conjecture is therefore seen to be valid for Q = 3.

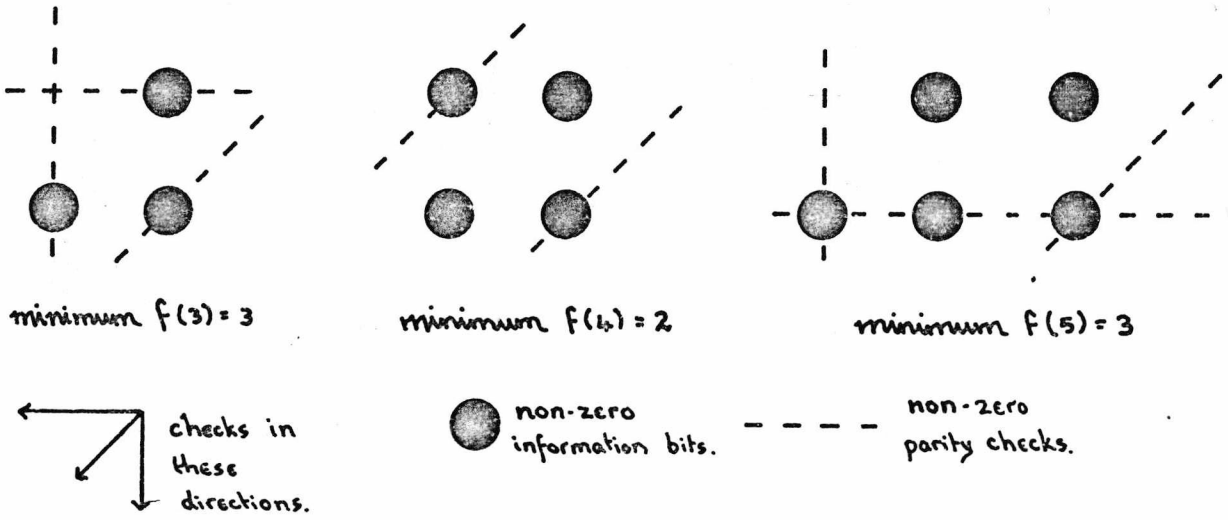


FIG 7.20

(iii) the case Q = 4

here $g(1) = 4, g(2) = 6, g(3) = 6, g(4) = 4$

then the conjecture maintains that

$f(4) \geq 2Q - g(4) = 4$

$f(5) \geq 2Q - g(3) = 2$

$f(6) \geq 2Q - g(2) = 2$

$f(7) \geq 2Q - g(1) = 4$

fig. 7.21 shows the configurations for minimum f(4), f(5), f(6) and f(7) confirming the validity of the conjecture for Q = 4

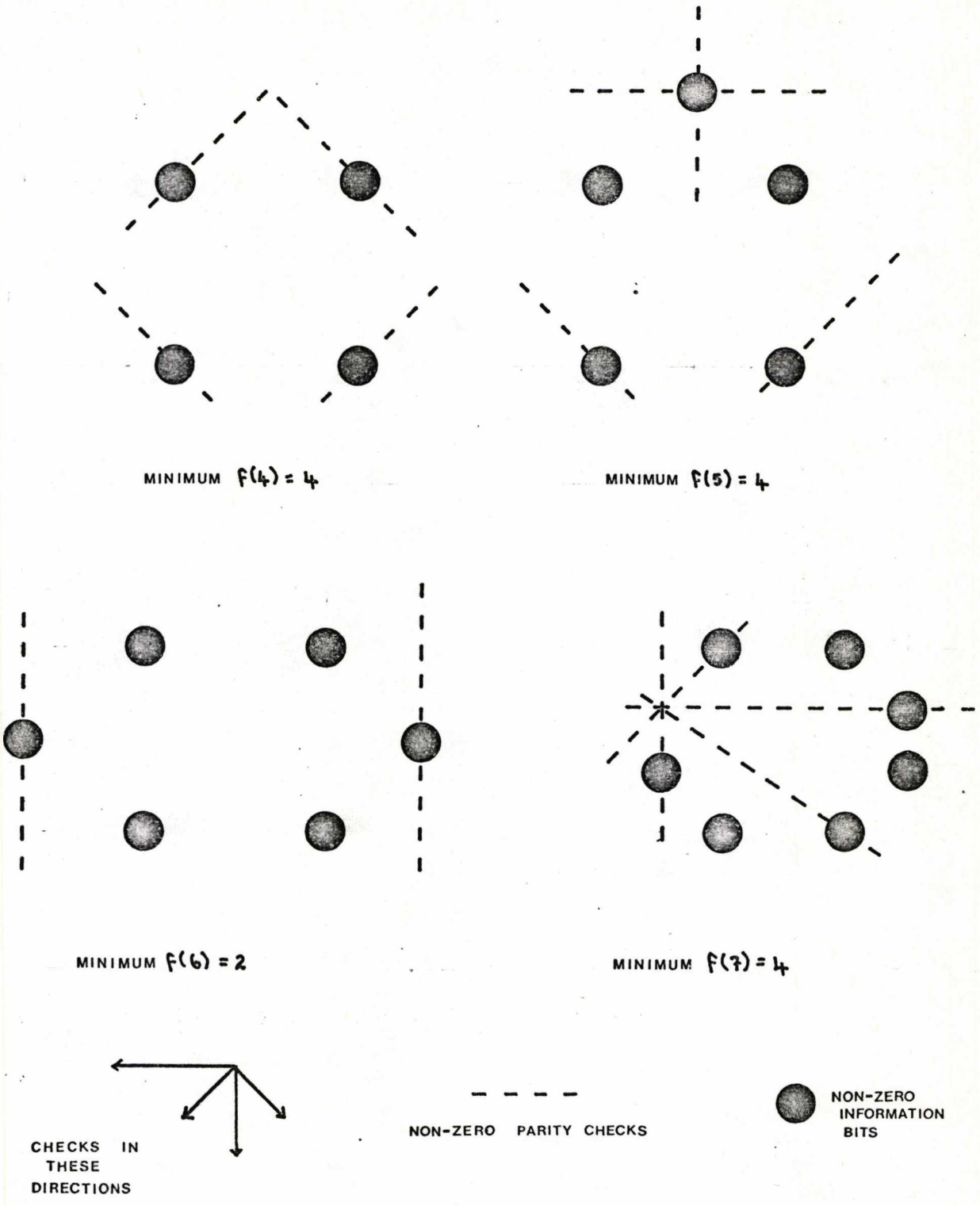

MINIMUM F(4) = 4
MINIMUM F(5) = 4
MINIMUM F(6) = 2
MINIMUM F(7) = 4
CHECKS IN THESE DIRECTIONS
NON-ZERO PARITY CHECKS
NON-ZERO INFORMATION BITS

Fig 7.21.

Intuitively, the conjecture looks true for $Q > 4$ but no proof of this can be offered.

A short list of conjectured construction D codes is given in table 7.10.

Table 7.11 gives a list of some codes formed from shortened construction A codes, and table 7.12 gives a list of some codes augmented by the technique described previously for the other array codes.

Minimum distance	n	k	R
7	40	16	0.4
7	112	64	0.571
7	216	144	0.667
11	62	16	0.258
11	158	64	0.405
11	286	144	0.504

Table 7.10 Construction D codes

Minimum distance	n	k	R
7	32	12	0.375
7	71	37	0.521
7	124	76	0.613
7	272	196	0.721

Table 7.11 Construction D codes from shortened construction A codes

Minimum distance	n	k	R
7	40	18	0.45
7	91	59	0.66
7	214	174	0.813

Table 7.12. Augmented Construction D. Codes.

7.13. Construction E codes

These codes are pure conjecture. It is possible that the addition of overall parity checks to construction B and C codes will increase their minimum distance. The only support for this hypothesis is that the minimum distance of such codes with various parameters have been found to be consistent with this conjecture.

Example 1.

Adding one overall parity check to the construction B code with n = 18 k = 9 and minimum distance = 4 to give a 19,9 double error correcting code.

The generator matrix of the code is

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

and the weight distribution is:

Weight of codeword	No. of this weight
1	0
2	0
3	0
4	0
5	9
6	57
7	18
8	45
9	126
10	126
11	45
12	18
13	57
14	9
19	1

Example 2

The addition of this overall parity checks to the 21,9 construction B code with minimum distance 5, to give a triple-error correcting code with n = 23 and k = 9.

The generator matrix is then

$$\begin{bmatrix}
1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1&1&0&0&1&1\\
0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&1&0&1&1\\
0&0&1&0&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0&1&1&1\\
0&0&0&1&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&1&0&1&1\\
0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&1&1&1\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&1&0&1&0&0&1&1\\
0&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&1&0&0&0&1&1&1\\
0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&1&0&0&1&0&0&1&1\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&1&0&2&0&1&1
\end{bmatrix}$$

and the weight distribution is:

Weight of Codeword	Number of codewords of that weight
1	0
2	0
3	0
4	0
5	0
6	0
7	9
8	102
11	144
12	144
15	102
16	9

7.14. Adding overall parity checks to other self-orthogonal codes

It is reasonable to consider the possibility of adding overall parity checks to self-orthogonal codes other than the array codes, in order to increase their minimum distance.

Codes may certainly be constructed for which the process is of no value. For example the 10, 4 code of minimum distance 4 with the generator matrix, G, given by

$$G = \begin{bmatrix} 1000\ 111000 \\ 0100\ 100110 \\ 0010\ 010101 \\ 0001\ 001011 \end{bmatrix}$$

is self orthogonal. The code does not benefit by the addition of overall parity check bits, however, since the codeword 1111000000 does not increase in weight as a result.

Nevertheless, the (39,26) code of minimum distance 4, one of the quasi-cyclic self-orthogonal codes found by Townsend and Weldon (1967) was investigated to see if the minimum distance could be increased by the addition of overall parity checks. As in the case of the (10,4) code described above, a codeword with four non-zero information bits and no non-zero parity check bits was found. It is concluded that there is no general method of increasing minimum distance of self-orthogonal quasi-cyclic codes by the addition of overall parity check bits.

CONCLUSION

It has been seen that short optimum codes are relatively easily decoded. Long codes, even when somewhat short of optimum, such as majority logic decodable codes or BCH codes, are much more complex to decide even for moderate error correcting powers. Nevertheless, codes far from optimum but easily decoded are a realistic consideration for practical systems because when decoder cost is limited it is possible to use a much longer simply decoded code. Further, the ability to decode well beyond the minimum distance of such codes makes them even more attractive.

Townsend and Weldon's self-orthogonal codes were presented as a useful set of such codes, whilst Hashim's nested codes were shown to be less attractive.

Conjoined codes were shown to be useful easily decoded codes where a low rate is acceptable, and they were seen to be capable of correcting more errors than guaranteed by their minimum distances. By an augmentation process the rate of the codes may be increased without making the decoding scheme over-complex. An interesting relationship between the aumented conjoined codes and the Reed-Muller codes has shown - providing an alternative decoding scheme for those codes.

Array codes have been presented. They may be constructed in various ways. The construction A, B, and C array codes are self-orthogonal and therefore very easily decoded by one-step majority decision methods. The construction B and C codes are

close to the best achievable for one-step majority-logic decodable codes, and may be constructed for any error correcting power and for a very wide range of rates above one half. All of the array codes may be aumented in a simple way - increasing their rate without unduly increasing the decoding complexity. Some of the array codes - notable construction A codes, may be shortened in a way which increases their rate for a given length, but maintains the simple decoding algorithm. Construction D codes were presented which, except in the double-error-correcting case, have no known decoding algorithm. The double-error correcting construction D codes were shown to have a decoding algorithm closely related to one step majority logic decoding with variable thresholds. Construction D codes have rates higher than those achievable for self-orthogonal codes of the same rate.

Several areas of the work presented in this thesis provide opportunity for further research. The most useful of these would probably be an attempt to find a simple decoding algorithm for the construction D array codes, this also being an attempt to prove the conjecture upon which they are based for error correction powers greater than three.

Another area of interest would be the correction of burst errors with array codes. It is probable that many burst errors are correctable using the decoding algorithms given for the codes.

Modifications to the algorithms might allow the codes to be burst-only decoded with some advantage; but burst and random error correcting codes are an important and perhaps somewhat neglected area of coding theory - the characterisation of bursts correctable by the random error correcting algorithms might be the most useful work.

Conjoined codes, also, might correct many burst errors and their similarity to interleaved codes suggests modifications to the decoding algorithms to allow burst only correction.

The relationship between conjoined codes and Reed-Muller codes has been shown. There is almost certainly a connection between the non-linear codes derived from Hadamand matrices, and the codes formed by conjoining shorter such codes, in an analogous way to the Reed-Muller case. Certainly the parameters of conjoined Hadamard matrix derived codes augmented with other such codes are identical to Hadamard matrix codes of twice the length. It is possible that such a connection would provide simplifications of decodings algorithms for such codes.

Another useful but perhaps tedious area for work is in the simulation of decoders for conjoined codes and for construction B and C array codes, in order to assess their performance compared to other classes of codes.

Appendix B of this thesis gives details of a forward-error-correction scheme for use on the H.F. channel (i.e. the radio frequencies between 3 and 30 MHz used as carriers for data). The scheme is simple, and lends itself extremely

well to simulation. It is certainly an interesting possibility for research.

APPENDIX 'A'

Combinational Circuits for the Multiplication of Element of GF (2^m)

A two level realisation of a combinational circuit for multiplication of elements of Galois fields of order 2^m would have 2m inputs and m output, and require in the order of m^3 gates. It is possible, however, to use far less gates by adopting the following method which was put forward by Bennet & Stein (1963). In this appendix, the method is described and also bounds on the number of gates required for the realisation of suitable circuits are derived. The bounds are listed in a table for various useful values of m, and the actual number of gates required is given for selected m.

Consider an element of GF(2^m), which may be represented as :

$$\sum_{i=0}^{m-1} a_i X^i$$

another may be represented as

$$\sum_{i=0}^{m-1} b_j X^j$$

the product is then :

$$\sum_{i=0}^{m-1} a_i X^i \qquad \sum_{j=0}^{m-1} b_j X^j$$

$$= \sum_{i=0}^{m-1} \qquad \sum_{j=0}^{m-1} a_i b_j X^{i+j}$$

there are therefore m^2 products of the kind $a_i b_j$, and these may be formed very simply using m^2 'AND' gates, since the a_i and b_j are either '0' or '1'.

Now, the maximum value of $X^{i + j}$ is $X^{2m - 2}$ so :

$$\sum_{i = 0}^{m - 1} \sum_{j = 0}^{m - 1} a_i \, b_j \, X^{i + j} = \sum_{k = 0}^{2m - 2} C_k \, X^k$$

The number of products of the type $a_i \, b_j$ which contribute to each C_k may be shown to be :

$$m = k - m + 1$$

and therefore each C_k may be formed from these products by adding them modulo 2. This will require m - k - m + 1 - 1 exclusive - or gates for eacg C_k. The total number of exclusive - or gates required is therefore :

$$\sum_{k = 0}^{2(m - 1)} m - 1 - k - (m - 1) = (m - 1)^2$$

now, for the result to be given as a member of $GF(2^m)$ it must be reduced modulo the irreducible polynomial used to generate the field. That is to say, the co-efficients of X^p, $p \geqslant m$, must be mapped into sums of X^q, $0 \leqslant q < m$.

i.e. $$X^p = \sum_{i = 0}^{m - 1} d_i \, X^i$$

where $d_i = 0$ or 1

Clearly, by the properties of Galoisfields, at least two of the d_i must be 1, and so the case requiring the least number of gates would have each X^p, $p \geqslant m$, mapping into a polynomial with only two terms. The most number of gates would be required for the case where one of the X^p maps to an expresion with m terms, and the remaining X^p map to expressions with (m - 1) terms.

These are two extreme cases - the actual number of terms involved depending upon the irreducible polynomial used to generate the Galoisfield.

The two cases above will, however, give an upper and a lower bound on the number of gates required to implement the multiplier.

Now, the co-efficients of X^i, $i < m$, are obtained by adding, modulo 2, those d_i just derived to the co-efficients of X^p, $p < m$.

The number of exclusive or gates required to form the co-efficients of the result is given by the number of d_i which are equal to one.

In the simplest case, the number of these is simply $2(m - 1)$.

In the most complex case, the number is $(m - 1) + (m - 1)(m - 2) = (m - 1)^2$.

Hence, the total number of gates required are :

(a) in the simplest case

m^2 two-input 'AND' gates with $(m - 1)^2 + 2(m - 1)$ exclusive - or gates making a total of $2m^2 - 1$ gates.

(b) in the most complex case

m^2 two input 'AND' gates with $(m - 1)^2 + m + (m - 1)(m - 2)$ exclusive - or gates making a total of $3m^2 - 4m + 3$ gates.

These bounds are listed in Table A.1 for various valves of m. Also listed are the actual number of gates required for selected values of m.

Example

To clarify the method of multiplication given above, the case of $m = 3$ is constructed here. $GF(2^3)$ is generated by the irreducible polynomial $X^3 + X + 1$.

Writing the multiplication, A, as $A_2 X^2 + A_1 X + A_0$ and the multiplicant, B, as $B_2 X^2 + B_1 X + B_0$ we have the product AB given by :

$$AB = A_2 B_2 X_4 + (A_2 B_1 + S_1 B_2) X^3 + (A_2 B_0 + A_1 B_1 + A_0 B_2) X^2 + (A_1 B_0 + B_1 A_0) X + A_0 B_0 \quad \ldots (i)$$

Clearly the formation of the products $A_i B_i$ require the use of $m^2 = 9$ two input 'AND' gates.

The summing of the $A_i B_j$'s required for the co-efficients of X^3, X^2, and X clearly require 4 exclusive - or gates (= $(m - 1)^2$ gates).

Now, since the GF(2^3) is generated by the equation $X^3 + X + 1$, it may be seen that $X^3 = X + 1$ and $X^4 = X^2 + X$.

Labelling the co-efficients of X^i in equation (i) as C_i we see that the co-efficients of the product

$$AB = P_2 X^2 + P_1 X + P_0$$

are given by $P_2 = C_2 + C_4$, $P_1 = C_1 + C_3 + C_4$, and $P_0 = C_0 + C_3$, and the formation of the P_i therefore requires an additional 4 exclusive - or gates. Notice this total is equal to 2(m - 1) and therefore the number of gates required is equal to the lower bound given in Section (a).

The circuit of the resulting GF(2^3) multiplier is given in Fig. A.1.

M	Lower Bound	Upper Bound	Actual No. Required
3	17	18	17
4	31	35	31
5	49	58	50
6	71	87	-
7	97	122	99
8	127	163	141
9	161	210	167
10	199	261	-

TABLE A.1.

Bounds on the Number of Gates Required for Multipliers of Elements over GF(2^m)

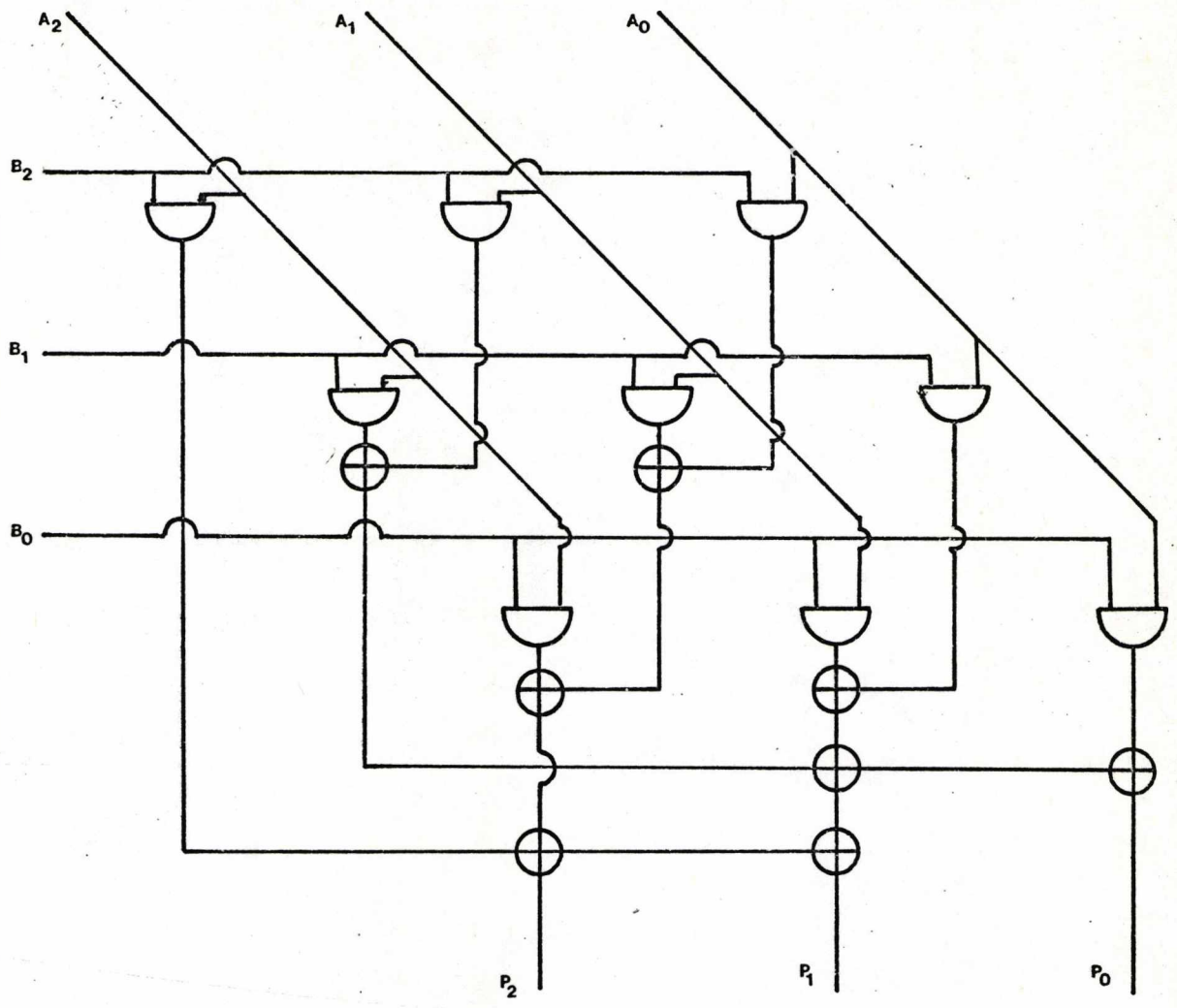


Fig A.1 : A Combinational Multiplier Circuit for Elements over $GF(2^3)$

= two input 'AND' gate

= exclusive - or gate

A_0, A_1, A_2 = co-efficients of multiplicand

B_0, B_1, B_2 = co-efficients of multiplicant

P_0, P_1, P_2 = co-efficients of product

APPENDIX B

A proposed Forward Error Correction System for H.F. Channels.

Efficient block FEC schemes are difficult to implement on relatively high-speed (1-3 Kbits/sec) HF digital channels, because it is found that most (50-75%) blocks contain errors. Even with a very high interleaving factor, the output error rate is often unacceptably high. It is characteristic of the HF channel, however, that occasional "clear patches" occur; that is, about 25-30% of blocks may be error free, or nearly so. It should be possible to make use of these error-free periods to transmit data rapidly and reliably, so that data received during the noisy periods could then effectively be discarded. There are two aspects of the problem: to find the clear patches; and to use them efficiently when found. The proposed scheme is directed towards both these aspects.

In essence the scheme consists of (see Fig. BI also):-

(i) A high-rate binary error-detecting cyclic code, with k information digits in a block of length n digits, where n is approximately equal to the average length of a clean patch.

(ii) Each cyclic code word is transmitted i times, with i chosen so that the block length, ni, of the hybrid cyclic-repetition code so formed is roughly equal to the average number of digits in one clear patch plus one noisy patch.

(iii) The information digits in one of the cyclic code words in each hybrid block of ni digits (it does not matter which, since all are the same) is encoded as a 2^k-level symbol in a Reed-Solomon (R-S) multi-level erasure-correction code (Reed 1954). Thus, C (say) out of every N hybrid blocks would consist of i transmissions of n-digit cyclic code words, the "information" digits of which represent the 2k-level parity symbols of the R-S code.

(iv) The decoder searches for a clean patch by computing syndromes over n consecutive digits, sweeping cyclically through the whole of the ni digits of the hybrid code word by shifting a digit at a time. Thus ni syndromes would be the maximum number of syndromes that might be calculated. Since the ni digits consist of i repeats of a cyclic code word, the shifting block of n consecutive digits within the ni is one of n cyclic translates of the cyclic code word, and hence is a word from the same cyclic code. Thus syndrome calculation is very simple, since the same circuit can be used for all the syndromes.

(v) As soon as a zero syndrome is found, the sweep can be stopped, and the k information digits output (in the appropriate order).

(vi) Should a zero syndrome not be found, then the cyclic code word is labelled as an erasure, and corrected by the R-S multi-level code.

(vii) Erroneous decoding can occur either if a zero cyclic code syndrome corresponds to an undetectable error pattern,

or if the R-S erasure code makes a wrong correction. The latter can be minimised by noting that the number of erasures in each block of N R-S code-word multi-level symbols can be counted, so that the erasure correction capability of the R-S code need not be exceeded.

If K = N-C, then the rate of the coding scheme is given by:

$$R = \frac{k}{ni} \quad \frac{K}{N} \quad = \frac{kK}{niN}$$

For example, if n = 100, k = 90, i = 3, N = 10 and K = 8, then R = 0.24. Thus, an overall transmission rate of 1 kbit/sec could be achieved by operating at 4.16 Kbit/sec, which is less than the binary SSB Nyquist rate for a nominal 3KHz bandwidth.

Binary syndrome calculation for cyclic codes is extremely simple. Multi-level erasure correction is also relatively simple, and is much simpler than multi-level error correction. Thus a hardware implementation of the scheme would be apparently quite practical. Simulation would be particularly easy, because actual erasure correction would not be necessary. It would also be quite easy to vary the coding parameters, so as to find a range of useful values.

The scheme could be applied to serial or parallel channels. Performance might be improved, at the expense of greater complexity, by using:

(a) a partial-error-correction cyclic code (bounded distance decoding), with or without soft-decision detection;

(b) error location within the cyclic code words, so that parts of incorrect words could be combined to obtain a correct word. This might also permit a reduction in the required value of i.

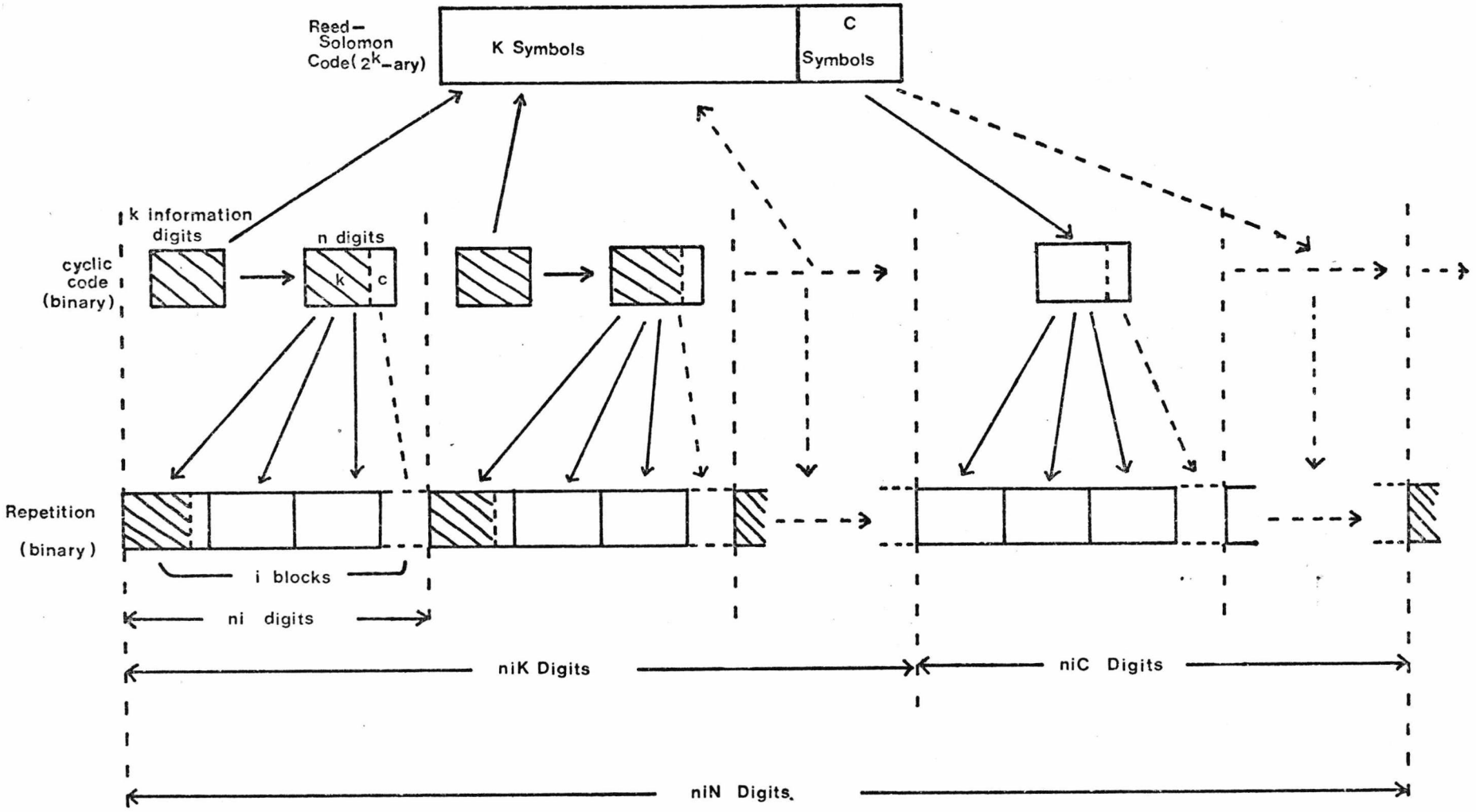

Reed–
Solomon
Code(2^K–ary)
K Symbols
C
Symbols
k information
digits
n digits
cyclic
code
(binary)
k
c
Repetition
(binary)
i blocks
ni digits
niK Digits
niC Digits
niN Digits.

Fig B.1

BIBLIOGRAPHY

Bartee T.C. and D.I. Schneider 1963 - "Computation with finite fields" Inf. and Control 6, 79 - 98.

Berlekamp, E.R. 1965 - "On Decoding Binary B se - Chaudhuri Hocquenghem Codes" IEEE Trans I.T. - 11 577 - 580.

1968 - "Algebraic Coding Theory" New York : McGraw Hill Book Company.

1973 - "Goppa Codes" IEEE Trans I.T. - 19 590 - 592.

Bose, R.C. and R.R. Kuebler Jr. 1958 - "On the construction of a class of error correcting binary signalling codes tech report univ. N. Carolina, N.C. (May)

Bose, R.C. and D.K. Ray-Chaudhuri 1960 - "On a class of error correcting binary group codes". Inf. and Control 3, 68 - 79.

Burton H.O. 1971 - "Inversionless decoding of binary B.C.H. codes" IEEE Trans I.T. - 17 464 - 466.

Burton H.O. and E.J. Weldon Jr. 1965 - "Cyclic Product Codes" IEEE Trans I.T. - 11 433 - 439.

Calabi L. and E. Myrvaagnes 1964 - "On the minimal weight of binary group codes" IEEE Trans I.T. - 10 385 - 387.

Chen C.L. 1969 - "Some results on Algebraically Structured Error-Correcting Codes". PH.D. Dissertation, University of Hawaii .

Chen C.L., W.W. Peterson, and E.J. Weldon, Jr. 1969 - "Some results on Quasi - cyclic codes" Inf. and Control 15 407 - 423

Chien R.T. 1964 - "Cyclic Decoding Procedures for Bose-Chaudhuri -Hocquenghem Codes" IEEE Trans I.T. - 10 357 - 363

Chien R.T. and Choy 1975 - "Algebrai Generalisation of B.C.H. - Goppa - Helgert codes" I.T. - 21 No. 1 P. 70 IEEE Trans.

Elias P. 1954 - "Error Free Coding" I.R.E. Trans, P.G.I.T. - 4 29 - 37

Farrag A. 1976 - "Anticodes and Optimum Error Correcting Codes" Ph.D. Thesis University of Kent at Canterbury.

Farrell P.G. 1970 - "Linear Binary Anticodes" Elec. Letters Vol. 6 No. 13 419 - 421

Farrell P.G. and Farrag A - "Further properties of Linear Binary Anticodes" Elec. Letters Vol. 10 No. 16, P. 340.

Forney G.D. Jr. 1966 - "Concatenated Codes" M.I.T. Press monograph 37.

Forney G.D. Jr. 1969 - "Algebraic structure of Convolutional Codes" presented at IEEE International Symposium of Inf. Theory, Ellenville N.Y., also published IEEE Trans I.T. -16 November.

Gilbert E.n. 1952 - " A Comparison of Signalling Alphabets" Bell System Tech J. 31, 504-522.

Golay M.J.E. 1949 - " Notes on digital coding". Proc. I.R.E. 37 P.657.

Goppa V.D. 1970 - "A New Class of Linear Correcting Codes". Probl. Peredach Inform, Vol. 6, 24-30.

Goppa V.D. 1971 - "Rational Representation of Codes and (L,g) Codes". Probl. peredach. Inform, Vol.7, 41-49.

Gorenstein D., W. W. Peterson and N. Zierler 196 - "Two-Error-Correcting Bose Chaudhuri. Codes are quasi-perfect". Inf. and control 3, 291-294.

Hamming R.W. 1950 - "Error detecting and error correcting codes". Bell System Tech. J. 29, 147-160.

Harari S. - "Unalgorithm rapide de correction pour les codes B.C.H. binaires corrigeant 2 erreurs". Revue du Cethedec (france) 1974 Vdll, Supp 1, 30-33.

Hashim A.A. 1974 ; "Methods for constructing and decoding block error-correcting codes". PhD Thesis, Imperial College, London.

Helgert. 1977 - "Decoding of Alternant Codes". IEEE Trans IT-23, No.4, 513-514.

Helgert H.J., and R. D. Stinnaff 1973 - "Minimum distance bounds for binary linear codes". IEEE Trans IT-19 No.3, May.

Hocquengham, A. 1959 - "Codes correcteurs d'erreurs". Chiffres, 2, 147-156.

Hoffner C.W., and S.M. Reddy 1970 -"Circulant Bases for Cyclic Codes". IEEE Trans, IT-16, No.4, 511-512.

Justesen J. 1972 - "A class of assymptotically good algebraic codes". IEEE Trans IT-18, 652-656.

Karlin M. 1969 - "New Binary Coding Results by Circulants". IEEE Trans It-15, No.1, Part 1, 81-92.

Kasahari, Sugiyama, Hirasawa, and Namekawa , 1975 - "An erasures and errors decoding algorithm for Goppa codes". IEEE Trans IT-22 238-241, also corrections in Trans IT-22, P.765.

Kasami T. 1964 - "A decoding procedure for multiple error-correcting codes". IEEE Trans IT-10, 134-139.

Kasami T, S. Lin, and W.W. Peterson 1966 - "Some results on cyclic codes which are invariant under the Affire group". Air Force Cambridge Research Lab. Report.

Lee, C.Y. 1958 - "Some properties of non-binary error-correcting-codes". IRE Trans. I.T. 4 77 - 82.

Lin, S. and E.J. Weldon Jr. 1967 - "Long B.C.H. codes are bad". Inf. and Control 11 P. 445.

Lin, S. 1970 - "Introduction to Error Correcting Codes". Prentice Hall Inc.

Lucky R.W., J. Salz, and E.J. Weldon Jr. 1968 - "Principles of Data Communication" N.Y. McGraw-Hill Book Company.

MacDonald J.E. 1958 - "Contructive Coding methods for the binary symmetric Independent Data Transmission Channel". M.S. Thesis, Dept. Electrical Eng. Syracuse University N.Y.

MacWilliams F.J. 1963 - "A Theorem on the Distribution of Weights in a Systematic code". Bell system Tech. J.42 79 -94.

MacWilliams F.J. 1963 - "Quadratic residue alphabets and how to decode them". Bell telephone labs. memorandum.

MacWilliams F.J. 1964 - "Permutation decoding of systematic codes". Bell system tech J. 43, 485 - 505.

MacWilliams F.J. and Sloane, N.J.A. 1977 - "The Theory of error-correting codes". Vol. I and II, North Holland.

Mandelbaum D. 1977 - "A method for decoding of generalised Goppa codes". IEEE Trans I.T. 23 No. 1 137 - 140.

Massey J.L. 1963 - "Threshold Decoding". M. I.T. Press Research Monograph 20, Cambridge, Mass.

Massey J.L. 1965 - "Step-by-Step decoding of the Base Chaudhuri Hocquenghem codes". IEEE Trans. I.T. - 11 580 - 585.

Mattson H.F. and G. Solom n 1961 - "A new treatment of Bose Chaudhuri codes". J. Soc. Indus. Appl. Math. 9.4. P.P. 654 - 669.

Meggitt J.E. 1960 - "Error correcting codes for correcting bursts of errors". I.B.M. J. Research Develop 4, 329 - 334.

Muller D.E. 1954 - "Application of Boolean Algebra to switching circuit design and error detection". I.R.E. Trans. E.C.3, 6 - 12.

Peterson W.W. - "Encoding and error correction procedures for the Bose-Chaudhuri codes". I.R.E. Trans. I.T. - b 459 - 470.

Peterson W.W. and E.J. Weldon Jr. 1972 - "Error correcting codes". M.I.T. Press.

Plotkin M. 1960 - "Binary codes with specified minimum distance I.R.E. Trans I.T. - 6 445 - 450.

Rao C.R. 1947 - "Factorial experiments derivable from combinatorial arrangements of arrays". J. Roy. Statist soc. suppl. 9 : 128 - 139; math rev. 9 : 264.

Reed I.S. 1954 - "A class of multiple-error-correcting codes and the decoding scheme". I.R.E. Trans, P.G. I.T. - 4 38 - 49

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

Retter C.T. 1975 - "Decoding Goppa codes with a B.C.H. Decoder". IEEE Trans I.T. - 21 P.112.

Rocha V.C. 1975 - "The decoding of cyclic codes using P.R.O.M.S.". Internal Report, electronics labs, University of Kent at Cantebury.

Rudolph L.D. 1967 - "A class of majority logic decodable codes". IEEE Trans. I.T. 23 No. 2 305 - 307.

Rudolph L.D. and C.R.P. Hartmann 1973 - "Decoding by sequential code reduction". IEEE Trans. I.T. - 19 No. 4 549 - 555.

Rudolph L.D. and C.R.P. Hartmann - "An optimum symbol-by-symbol decoding rule for linear codes". IEEE Trans. I.T. - 22 No. 5 514 - 518.

Sacks G.E. 1958 - "A multiple error correction by means of parity checks". I.R.E. Trans. I.T. 4 145 - 147.

Sarwate D.V. 1977 - "On the complexity of decoding Goppa codes". IEEE Trans. I.T. - 23 No. 4 515 - 516.

Shannon C.E. 1948 - "A mathematical theory of communication" Bell System Tech. J. 27 379 - 423, 623 - 656.

Schmandt F.D. 1976 - "On the practical application of sequential code Reduction". IEEE Trans I.T. - 22 482 - 483.

Smith R.J.G. 1977 - "Easily decodable efficient self-orthogonal block codes". Electronics letters Vol. 13 No. 7 173 - 174.

Solomon G. and J.J. Stiffler 1965 - "Algebraically Punctured cyclic codes" Inf. and Control 8 170 - 179.

Sugiyama, Kasahara, Hirasawa, and Namekawa 1976 - "New classes of binary codes constructed on the basis of concaternated codes and prodint codes". IEEE Trans I.T. 22 462 - 467.

Tietavainen A. 1977 - "There are no unknown perfect e, n, q, codes if q is a power of a prime". Discrete mathematics Vo. 17 No. 2 P.199.

Townsend R.L. and E.J. Weldon Jr. 1967 - "Self-orthogonal quasi-cyclic codes". IEEE Trans. I.T. 13, 183 - 195.

Varshamov R.R. 1957 - "Estimate of the number of symbols in error-correcting-codes". Doklady A.N.S.S.S.R. 117 No. 5 739 - 741.

Viterby A.J. 1967 - "Error bounds for convolutional codes and an assymptotically optimum decoding algorithm". IEEE Trans. I.T.13 260 - 269.

Wagner T.J. 1965 - "A remark concerning the minimum of distance of binary group codes". IEEE Trans. I.T. - 11 P. 458.

Weldon E.J. Jr. 1967 - "Euclidean Gemetry Cyclic Codes". Proceeding of symposium of combinational mathmatics at the University of N. Carolina.

Wozencraft J.M. and I.M. Jacobs 1965 - "Communication Engineering". Wiley.

Wright L.E. 1977 - "Mathematical structures for decoding projective geometry codes". Ph.D. thesis Imperial College London.

Yip P.W., S.G.S. Shiva and E.L. Cohen - "Permutation - decodable binary cyclic codes". Electronics letters Vo. 10 No. 22 P. 467.

EASILY DECODABLE EFFICIENT SELF-ORTHOGONAL BLOCK CODES

Indexing term: Codes

A class of self-orthogonal block codes is described, which are easy to decode and to augment. They are constructed by forming parity-check equations across certain patterns of information digits arranged in a 2-dimensional array. The codes are of a rate close to the optimum for self-orthogonal codes.

Introduction: Self-orthogonal block codes for error detection and correction are longer than some codes of similar rate and minimum distance, but make up for this defect by virtue of the ease with which they may be 1-step majority-logic decoded, and the fact that they are capable of correcting far more errors than their minimum distance would indicate.[1]

The codes described here are particularly easy to decode, and may be constructed for a wide range of information rates, which are close to the optimum for self-orthogonal codes. Further, it is possible to augment them in a simple way. They are constructed by forming parity-check equations across certain patterns of information digits arranged in a 2-dimensional array.

Construction: Consider a $p \times p$ array of squares, where p is prime. Let each square be labelled by the row r and the column c of the array in which it is situated, $0 \leqslant r, c \leqslant p-1$, and place $d-1$ numbers $D_i(r, c)$, $0 \leqslant i \leqslant d-2$, in each square.

If it can be shown that no two squares have more than one D_i of the same value, and if each square in the array is associated with an information bit, while each value which each $D_i(r, c)$ takes denotes a parity-check equation involving the information bit associated with the square (r, c), then, since no two information symbols are involved together in more than one parity-check equation, this arrangement describes a linear, binary, self-orthogonal block code[2] with minimum distance d and with p^2 information bits.

It remains to choose the $D_i(r, c)$ such that the above conditions apply, and such that the $D_i(r, c)$ take on a sufficiently small number of values that the codes formed have a reasonable rate, bearing in mind that the best that can be done for a self-orthogonal binary block code of length n, rate R and minimum distance d is [1]

$$n \geqslant \{R(d-1)(d-2)\}/(1-R)^2 + (1-R)^{-1}$$

Choose $D_i(r, c) \equiv (r+ic) \pmod p$ with the proviso that $i < p$, i.e. $d \leqslant p+1$. For any square (r_1, c_1) and any number $D_j(r_1, c_1)$ in that square it is required to prove that any other square (r_2, c_2) with $D_j(r_2, c_2) = D_j(r_1, c_1)$ has

$$D_k(r_2, c_2) \neq D_k(r_1, c_1)$$

for all $k \neq i$ and $i, j < p$.

Now, if $D_j(r_2, c_2) = D_j(r_1, c_1)$,

$$(r_2+jc_2) \pmod p \equiv (r_1+jc_1) \pmod p$$

Thus

$$(r_2-r_1) \pmod p \equiv [j(c_1-c_2)] \pmod p$$

but if $D_k(r_2, c_2) = D_k(r_1, c_1)$ this implies

$$(r_2-r_1) \pmod p \equiv [k(c_1-c_2)] \pmod p.$$

Thus

$$j \pmod p \equiv k \pmod p$$

i.e. $j = k$ for $j, k < p$

Thus it has been shown that $D_k(r_2, c_2) \neq D_k(r_1, c_1)$ for $j \neq k$ provided $j, k < p$. Note that if p were not prime, the calculations would be invalid.

To find how many values are taken by the $D_i(r, c)$, consider a column c of the array; then $D_i(r, c) \equiv (r+ic) \pmod p$. Here, ic is a constant and r takes all values 0 to $p-1$. There-

fore, since we are working modulo p, $D_i(r, c)$ takes all values 0 to $p-1$; this is true in each column.

There are $d-1$ of the $D_i(r, c)$ and each takes p values; therefore the number of parity checks on p^2 information bits is $p(d-1)$. The parameters of the codes are therefore

$$n = p^2 + p(d-1) \text{ and } k = p^2$$

Thus $n = (d-1)^2 R(1-R)^{-2}$, which compares well with the best that can be achieved with self-orthogonal codes, particularly for large values of d.

In fact, for codes of rate greater than one-half and minimum distance greater than 7, the codes are generally shorter than equivalent self-orthogonal quasicyclic (s.o.q.c.) codes.[1] Also, s.o.q.c. codes are often found only by means of a computer search for suitable difference sets, whereas the codes described here require only a knowledge of the prime numbers.

Decoding: The codes are easy to decode, owing to the properties

(*a*) $D_i(r+1, c) \equiv [1 + D_i(r, c)] \pmod{p}$

(*b*) $D_i(0, c+1) \equiv [1 + i + D_i(p-1, c)] \pmod{p}$

This allows parity checks, or syndromes, to be formed using only $(d-1)$ p-stage restricted access shift registers, with simple check and switching circuitry. Hence a decoder would require only a p^2-stage serial register, $(d-1)$ p-stage restricted-access shift register and one $d-1$ input majority gate, together with the clock and switching circuitry.

Example: Choose $d = 8$, $p = 29$; $n = (29)^2 + 7 \times 29 = 1047$ and $k = 29^2 = 841$. This code has a rate of 0·803, and may be compared with the s.o.q.c. code of minimum distance 8 and rate 0·8, which has a length of 1205.

Alternative definition: Instead of choosing p prime and defining $D_i(r, c)$ modulo p, it is possible to choose p to be a power of a prime, with the $D_i(r, c)$ now defined over the Galois field of p elements. The proofs of the code properties are identical. In this way, many more codes may be formed, although the practical realisation of the decoders will be more complex.

Example: Choose $d = 9$ and $p = 2^4 = 16$;

$$n = (16)^2 + 8 \times 16 = 384 \text{ and } k = 16^2 = 256.$$

This code has a rate of two-thirds, which may be compared with the s.o.q.c. code of minimum distance 9 and rate two-thirds, which has a length of 420.

Augmentation of the codes: Each parity-check equation associated with a value taken by a $D_i(r, c)$ is completely independent of every other parity-check equation associated with other values taken by the same $D_i(r, c)$, since no information bit is involved in more than one of the equations associated with a particular value of i of $D_i(r, c)$. Therefore, given less than $(d-1)/2$ errors in the codeword, a reconstruction from the received information digits of the parity-check bits associated with any particular value of i of $D_i(r, c)$ will be incorrect in at most $(d-1)/2$ positions.

This is precisely the property required to augment the codes by adding to each set of parity checks associated with a particular value of i of $D_i(r, c)$ a binary codeword of length p and minimum distance d, in the manner shown by Kasahara *et al.*[3] for product codes. The complexity of decoding will be approximately the sum of the complexities of the decoders for the constituent codes.

Example: Choose $d = 5$ and $p = 5$;

$$n = 5^2 + 4 \times 5 = 45 \text{ and } k = 5^2 = 25,$$

but $d-1$ codewords of length 5 and minimum distance 5 may be added. The codewords could be those from the (5, 1) repetition code. Then we have added 4 information bits, giving an augmented code with parameters $n = 45$, $k = 29$.

Acknowledgments: I wish to thank P. G. Farrell for his many helpful comments, and also the UK Science Research Council for financial support.

R. J. G. SMITH

Electronics Department
Kent University
Canterbury, Kent, England

19th February 1977

References

1 TOWNSEND, R. L., and WELDON, E. J., JUN.: 'Self-orthogonal quasi-cyclic codes', *IEEE Trans.*, 1967, **IT-13**, pp. 183–195
2 MASSEY, J. L.: 'Threshold decoding' (MIT Press, Cambridge, Mass., 1963), chap. 1, pp. 5–7
3 KASAHARA, M., SUGIYAMA, Y., HIRASAWA, S., and NAMEKAWA, T.: 'New classes of binary codes constructed on the basis of concatenated codes and product codes', *IEEE Trans.*, 1976, **IT-22**, pp. 462–467