Copyright © by

RICHARD RUTHERFORD GREEN

1969

DECODING COSETS OF FIRST-ORDER REED-MULLER CODES

Thesis by

Richard R. Green

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1969

(Submitted May 13, 1969)

ACKNOWLEDGEMENTS

The patient guidance and willingness to listen of my research advisor, Dr. H. C. Martel, are deeply appreciated. I am grateful for the financial support provided by the Tektronix Foundation Fellowship and the National Science Foundation Traineeship which I held as a graduate student. Mere words fail to express my gratitude for the typing of Mrs. Doris Schlicht, without whose extraordinary efforts the deadline for this thesis could never have been met.

ABSTRACT

Proper encoding of transmitted information can improve the performance of a communication system. To recover the information at the receiver it is necessary to decode the received signal. For many codes the complexity and slowness of the decoder is so severe that the code is not feasible for practical use. This thesis considers the decoding problem for one such class of codes, the comma-free codes related to the first-order Reed-Muller codes.

A factorization of the code matrix is found which leads to a simple, fast, minimum memory, decoder. The decoder is modular and only n modules are needed to decode a code of length 2^n. The relevant factorization is extended to any code defined by a sequence of Kronecker products.

The problem of monitoring the correct synchronization position is also considered. A general answer seems to depend upon more detailed knowledge of the structure of comma-free codes. However, a technique is presented which gives useful results in many specific cases.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

One approach to evaluating the performance of communication systems is to examine the rate of transfer of information as a function of the probability of making an error. Shannon's results demonstrate that if the information to be transmitted is encoded into a form more suited to the channel, an improvement in performance can be realized [1]. If this technique is used, then to recover the desired information at the receiver, it is necessary to decode the received signals.

The decoding operation is usually well specified and straight forward mathematically. However, in real systems the size, complexity, and hence slow operation of the equipment necessary to implement the mathematics, are frequently so overwhelming as to virtually prohibit the use of coded transmission.

Coding schemes and decoding techniques specifically to avoid this problem are of much current interest. Among other methods being studied are sequential decoding [2], majority-logic decodable codes [3], and threshold decoding [4].

A class of codes which have useful information handling properties are the first-order Reed-Muller codes [5], and the comma free codes based on them. This thesis examines the decoding problem for these codes. The objective is, where possible, to provide general purpose decoding equipment which will deal effectively with any such code. When this is not possible, insight into the underlying problems is sought.

CHAPTER II

DEFINING THE DECODING PROBLEM

2.1. System Model.

The communication system model which will be used to discuss decoding is very simple. Figure 2.1 gives the block diagram for the model. Specifically, it will be assumed that all the frequency translation, modulation, detection, etc., which are parts of a real communication system can be considered to be part of the channel connecting the encoder and decoder.

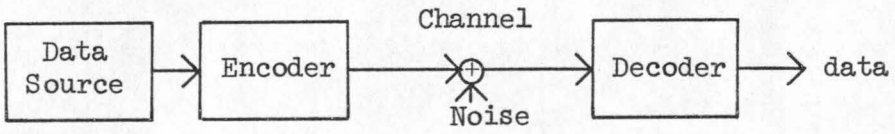

Figure 2.1

It will be assumed that the data source is a binary source whose output is considered n bits at a time. This means that the source may be regarded as emitting 2^n possible data vectors of length n. These possible vectors will be taken to be equally probable.

The encoder will be taken to provide a one-to-one mapping between the input data vectors and a set of 2^n binary code words of length m. To avoid confusion, components of the binary data vectors will be referred to as bits while components of the code words will be called symbols. The set of code words may be represented as a $2^n \times m$ matrix of ± 1's, K, where each row represents a code word. It is

assumed that each code word to be transmitted is converted to a binary function of time and transmitted over the channel. The time required to transmit the symbols will be taken to be τ seconds per symbol, and hence $m\tau$ seconds per code word. All the time functions generated by the encoder are presumed to be transmitted with equal energy. In this case, the 1:1 mapping between the data vectors and the code words may be chosen arbitrarily.

The noise added by the channel will be white Gaussian noise.

The decoder is to examine the received signal, call it $x(t)$, and make a decision about which one of the code words was transmitted, and hence what corresponding data vector appeared at the source. The decoder is to be optimum in the sense that it is to select that code word which is most likely to have been transmitted. This is called maximum likelihood decoding.

It will be further assumed that the decoder has perfect knowledge of symbol timing. In other words it is known exactly when one symbol has been completed and the next symbol begins. In actual communication systems this information is usually provided by either coherence between the symbol frequency and one of the subcarriers or the carrier involved, or by a separate tracking loop monitoring the symbol timing itself. Of course in reality these estimates are not perfect and do lead to a source of errors which will not be considered here.

Initially it will be assumed that the decoder also knows word timing precisely. Later, comma-free codes will be assumed, to remove this restriction.

Under these assumptions, it is known [6] that the maximum

likelihood decoder selects j such that $1 \le j \le 2^n$ and

$$|y_j| = \max_{1 \le i \le 2^n} \{|y_i|\} \qquad \text{where}$$

$$y_i \triangleq \frac{1}{m\tau} \int_0^{m\tau} x(t) k_i(t) dt \quad \text{and} \quad k_i(t) \text{ is the}$$

binary waveform corresponding to the i^{th} row of K. By suitable scaling, it may be assumed that $k_i(t) = \pm 1$ for all t. Let x_s be defined by:

$$x_s \triangleq \frac{1}{m\tau} \int_{(s-1)\tau}^{s\tau} x(t) dt \quad \text{for} \quad 1 \le s \le m$$

then

$$y_i = \sum_{s=1}^{m} k_{is} x_s, \quad k_{is} = \text{the } s^{th} \text{ symbol of } k_i(t).$$

Write both x and y as vectors, x a vector of m real numbers and y a vector of 2^n real numbers. (As done by Gale [7], no distinction will generally be made between the artificial concepts of row vectors and column vectors.) Then:

$$y = Kx$$

where, in order to do maximum likelihood decoding, we wish to know which component of y has the maximum magnitude.

The block diagram for the first part of the decoder, for any dictionary K, is shown in Figure 2.2.

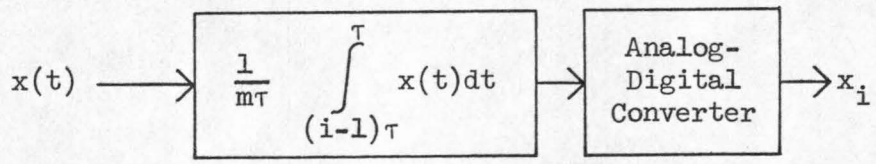

Figure 2.2

As it is anticipated that the operation Kx will be evaluated using digital techniques, an analog to digital converter is shown. This device takes the analog output of the integrator and converts it to a digital binary number suitable for further digital processing.

2.2. First-Order Reed-Muller Codes.

A particularly interesting class of codes are the first-order Reed-Muller codes. There are two reasons for this. First, they exhibit a structure over GF(2), the finite field of 2 elements, which makes them particularly easy to manipulate. The code words may be taken to be the rows of a Hadamard matrix. Second, Stiffler has found a class of comma-free codes which are closely related to the first-order Reed-Muller codes [8]. As the word synchronization properties of comma-free codes can be used to eliminate power wasted transmitting synchronization information, techniques for decoding them are well worth studying.

The Hadamard matrices of size 2^n by 2^n may be defined and

constructed by using the Kronecker product. The 2 x 2 Hadamard matrix, denoted H_1, is:

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

Larger Hadamard matrices may be defined inductively

$$H_n = H_{n-1} \otimes H_1$$

Here, $\otimes$ denotes the Kronecker product, which is defined as $A \otimes B = (a_{ij}B)$ for two matrices A and B. It should be noted that these matrices are symmetric and that their rows are mutually orthogonal. Indeed, $H_n^{-1} = 2^{-n}H_n$. Since the 2^n rows of H_n are mutually orthogonal and are elements of Euclidian 2^n-space, they form a basis.

The set of all binary vectors of length 2^n, written with their elements +1 and -1, form an Abelian group under the operation of term by term multiplication. The rows of H_n form a subgroup under the same operation. The identity element, or identity vector, of this group is the all 1's vector, $(1,1,\cdots,1)$.

Suppose that some binary vector η of length 2^n which is not a row of H_n, is term by term multiplied with every vector which is a row of H_n. The resulting set of binary vectors is called a coset of the subgroup defined by the rows of H_n. As the identity vector is one of the rows of H_n, namely the first, the vector η itself is a member of the coset. η is referred to as the coset leader.

The coset can be represented easily in matrix form. Let Λ_η be a diagonal matrix with all elements not on the main diagonal zero and the diagonal elements the components of η, in order. Then if h_i is one of the rows of H_n and c_i is the element of the coset which results when h_i and η are multiplied, $c_i = h_i\Lambda_\eta$. Thus if C is a matrix whose rows are the elements of the coset $C = H_n\Lambda_\eta$. Note that the subgroup defined by the rows of H_n may be considered to be a coset also, by letting η be the identity vector. In this case $\Lambda_\eta = I$ and $C = H_n$.

If the code dictionary discussed in 3.1 is selected to be a first-order Reed-Muller code or a coset of a first-order Reed-Muller code, then $K = H_n$ or $K = H_n\Lambda_\eta$ respectively, and $m = 2^n$. The decoding operation for these cases is $y = H_n x$ and $y = H_n\Lambda_\eta x$ respectively.

2.3. Comma-Free Codes.

Recall that in defining the system model in Section 2.1, it was assumed that the decoder had exact information about when one transmitted code word ended and the next one began. In actual practice, this synchronization information is quite frequently transmitted on a separate, low power, subcarrier. The power required for this, and consequently not available for transmitting information, can be as much as 10% of the available transmitter power. This power loss can be avoided, however, by the use of comma-free or self-synchronizing codes.

For any coset of a first-order Reed-Muller code, the correlation between any pair of elements in the coset appears as a term in:

$$CC^T = H_n\Lambda_\eta(H_n\Lambda_\eta)^T = H_n\Lambda_\eta\Lambda_\eta H_n^T$$

$$= H_nH_n^T = H_nH_n$$

It is thus apparent that the correlation properties, assuming word synchronization, are the same for the coset as for the original subgroup. However, if it is assumed that word synchronization is not available, then evaluating $y = Kx$ will amount to correlating words in the dictionary with received words formed by taking the last i symbols from one unknown code word and the first 2^n-i symbols from another unknown code for an unknown value of i. This out of phase correlation property will, in general, be different for different cosets. Stiffler has found that there exist cosets for which every word in the coset differs from every possible word formed by out of phase sequences from two joined words in at least ρ positions, for some integer ρ. Such cosets, when used as code dictionaries, are called comma-free codes. ρ is called the index of comma freedom.

In order to decode one of these comma-free codes, the decoding apparatus must first find the correct word synchronization position. If the decoder is only capable of evaluating $y = Kx$ for the received vector x, a search technique must be used to achieve synchronization. One method of searching is to assume some arbitrary position is the correct phase. The vector y is computed for some predetermined number, W, of input vectors x. A sum, μ, is formed from the maximum magnitude components of each y, i.e. letting the W vectors y be denoted $y^1, y^2, \cdots, y^W$ and letting the maximum

magnitude component of y^i be y_m^i, then $\mu = \sum_{i=1}^{W} |y_m^i|$. These computations are repeated for each of the 2^n-1 other possible phase positions. After this search has been completed, the most probable location of the correct synchronization point is that for which μ is the largest. The question of probability of a correct decision as a function of index of comma freedom, W, and signal to noise ratio has been examined [9].

Once a decision has been reached, the decoder assumes that word synchronization is fixed. Decoding then proceeds as discussed in Section 2.1.

2.4. The Decoding Problem.

In order to decode any coset of a first-order Reed-Muller code it is only necessary to evaluate $y = Kx = H_n \Lambda_\eta x$. However, as in many engineering problems, while it is clear what needs to be done, there are practical difficulties involved which render the job all but impossible.

CHAPTER III

A METHOD FOR DECODING

3.1. Motivation.

The major portion of the arithmetic manipulation involved in decoding is in calculating the vector $y = Kx$. The complexity of this calculation becomes significant for a combination of two reasons. First, in cases of interest, K is very large and has no zero elements. Since K is $2^n \times 2^n$, if the operation Kx were evaluated directly, there would need to be $2^n(2^n-1)$ additions or subtractions performed. The smallest value of n for which η can be found to make K a comma-free dictionary is $n = 4$ [10], and in order to more fully exploit the improvement in error probability offered by these codes, it is desirable to have $n = 7$ or 10 or 15. Even for $n = 10$, direct evaluation of Kx would involve more than 10^6 additions or subtractions.

Even these large numbers of calculations could be accommodated, given an adequate length of time, but for the second complicating reason, the data rate. For systems of current practical interest, the data rates used imply typical code word rates of 10^3 to 10^4 code words per second. This allows only 100 to 1000 microseconds to calculate Kx if decoding is to be accomplished in real time. For example, the Mariner Mars fly-by mission in 1969 includes a coded system using $n = 5$, data rate = 16,200 bits per second, 2700 code words per second, and thus 3×10^6 additions or subtractions per second.

This high calculation rate places the job of direct evaluation of Kx beyond the current level of technology of general purpose digital computers. Both the size of K and its complexity make the construc-

tion of special purpose equipment with sufficient speed to evaluate Kx directly, term by term, prohibitively expensive. Thus, it becomes of interest to investigate more subtle approaches to the evaluation of Kx. Since special purpose digital equipment is almost always faster for a given task than a general purpose computer, any different approach should be directed toward an algorithm which could be easily implemented with hardware.

There is at least one intuitive reason to believe that a reduction in the number of calculations in Kx is possible. The first addition performed in computing the first component, y_1, of $y = Kx = H_n \Lambda_\eta x$ would be $\eta_1 x_1 + \eta_2 x_2$. The first addition for y_2 would be $\eta_1 x_1 - \eta_2 x_2$. For subsequent components of y, say y_i, for i odd the first sum is always $\eta_1 x_1 + \eta_2 x_2$; for i even, $\eta_1 x_1 - \eta_2 x_2$. Instead of computing $\eta_1 x_1 \pm \eta_2 x_2$ separately for each component of y, it seems more reasonable to compute each of the 2 sums once, store them, and use them as needed. Since all components of K are either $+1$ or -1, there are only 4 possible sums for any pair of components of x, namely x_i+x_j, x_i-x_j, $-x_i+x_j$, and $-x_i-x_j$. Thus, the same type of argument used for x_1 and x_2 can be extended to any pair of components of x. In other words, the idea is to try to store, keep track of, and use intermediate results of the computation process to eliminate duplication of individual calculations.

For the first-order Reed-Muller codes, and thus for any coset of the first-order Reed-Muller codes, a reduction in the number of calculations required in computing Kx is possible. As will be shown, the reduction is achieved by factoring H_n into the matrix

product of n different $2^n \times 2^n$ matrices, which will be denoted $M_n^{(1)}$, $M_n^{(2)}$, $\cdots$, $M_n^{(n)}$. Each of these matrices, $M_n^{(i)}$, will have only 2 nonzero elements per row.

3.2. Kronecker Product and Notation.

Before proceeding further, there are some notation standards which should be explicitly mentioned. Capital letters are used to represent square matrices, with I being used exclusively for identity matrices and Λ for other diagonal matrices. Lower case Roman letters used as subscripts on matrices denote $\log_2$ of the dimension of the matrix; e.g. A_n implies A is $2^n \times 2^n$. Note that if $n = 0$ then A_n is a 1×1 matrix or just a real number. In particular, $I_o = 1$. Lower case Greek letters used as subscripts on diagonal matrices denote the diagonal elements as a vector; e.g., if $\eta = (\eta_1, \eta_2, \cdots, \eta_m)$ then Λ_η is defined by $\lambda_{ii} = \eta_i$ and $\lambda_{ij} = 0$ for $i \neq j$. Vectors are represented by lower case letters.

The Kronecker product of matrices is used in deriving the properties of the factor matrices, $M_n^{(i)}$. Recall that the Kronecker product of two matrices, say A and B, is defined by $A \otimes B \triangleq (a_{ij}B)$. This product is associative, i.e. $(A \otimes B) \otimes C = A \otimes (B \otimes C)$, and it is clearly not commitative, i.e. $A \otimes B \neq B \otimes A$ in general. If the dimensions are correct for the necessary ordinary matrix products to be defined, $(A \otimes B)(C \otimes D) = AC \otimes BD$ [11].

From the foregoing, the following useful relations can be easily proved:

1. $I_m \otimes I_n = I_{m+n}$

2. $(I_m \otimes A_n)(I_m \otimes B_n) = I_m \otimes A_n B_n$

3. $(A_n \otimes I_m)(B_n \otimes I_m) = A_n B_n \otimes I_m$

4. $(A_n \otimes I_m)(I_n \otimes B_m) = A_n \otimes B_m$
$= (I_n \otimes B_m)(A_n \otimes I_m)$

3.3. Analysis.

Define a new matrix $M_n^{(i)}$ as follows:

$$M_n^{(i)} \triangleq I_{n-i} \otimes H_1 \otimes I_{i-1} \quad \text{for} \quad 1 \le i \le n \ .$$

An example of one of these matrices for $n = 3$ and $i = 2$ is as follows:

$$M_3^{(2)} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

Figure 3.1

Note that this is a block diagonal matrix, with the diagonal sub-matrix being $H_1 \otimes I_{i-1}$, which is repeated on the main diagonal as many times as there are 1's in I_{n-i}. Also, it is important that

there are only 2 nonzero elements per row of $M_n^{(i)}$ for any n and any i.

The first important property these matrices possess is that any two of them commute under ordinary matrix multiplication.

Theorem: $M_n^{(i)} M_n^{(j)} = M_n^{(j)} M_n^{(i)}$

Proof: If $i = j$ there is nothing to prove. Assume $i > j$ then;

$$M_n^{(i)} M_n^{(j)} = (I_{n-i} \otimes H_1 \otimes I_{i-1})(I_{n-j} \otimes H_1 \otimes I_{j-1})$$

$$= ([I_{n-i} \otimes H_1 \otimes I_{i-j-1}] \otimes I_j)(I_{n-j} \otimes [H_1 \otimes I_{j-1}])$$

$$= I_{n-i} \otimes H_1 \otimes I_{i-j-1} \otimes H_1 \otimes I_{j-1}$$

$$= (I_{n-i+1} \otimes [I_{i-j-1} \otimes H_1 \otimes I_{j-1}])([I_{n-i} \otimes H_1] \otimes I_{i-1})$$

$$= (I_{n-j} \otimes H_1 \otimes I_{j-1})(I_{n-i} \otimes H_1 \otimes I_{i-1})$$

$$= M_n^{(j)} M_n^{(i)} \qquad \text{Q.E.D.}$$

Thus, when discussing matrix products of the $M_n^{(i)}$, it is not necessary to keep track of the order in which the matrices appear. That these matrices commute is useful in discussing the result of multiplying together the first m of them.

Theorem: $\prod_{i=1}^{m} M_n^{(i)} = I_{n-m} \otimes H_m$

where $1 \le m \le n$.

Proof: This theorem is very simply proved by induction. For $m = 1$ we have:

$$\prod_{i=1}^{1} M_n^{(i)} = M_n^{(1)} \triangleq I_{n-1} \otimes H_1 \otimes I_o = I_{n-1} \otimes H_1$$

Assume the result is true for m and prove that it follows for $m + 1$:

$$\prod_{i=1}^{m+1} M_n^{(i)} = M_n^{(m+1)} \prod_{i=1}^{m} M_n^{(i)} = (I_{n-m-1} \otimes H_1 \otimes I_m)(I_{n-m} \otimes H_m)$$

$$= I_{n-m-1} \otimes H_1 \otimes H_m$$

$$= I_{n-(m+1)} \otimes H_{m+1}$$

Thus, by induction, the result is true for any value of m between 1 and n. Q.E.D.

In particular, letting $m = n$ in this theorem yields: $\prod_{i=1}^{n} M_n^{(i)} = I_{n-n} \otimes H_n = I_o \otimes H_n = H_n$. Thus, H_n has been factored into the product of n different matrices. The original problem of computing $y = Kx = H_n \Lambda_\eta x$ may now be written $y = M_n^{(n)} M_n^{(n-1)} \cdots M_n^{(2)} M_n^{(1)} \Lambda_\eta x$, where the $M_n^{(i)}$ could be written in any arbitrary order because of their commutivity relationship. If we let $z \triangleq \Lambda_\eta x$, $z^1 \triangleq M_n^{(1)} z$, $z^2 \triangleq M_n^{(2)} z^1$, etc., and finally $y = M_n^{(n)} z^{n-1}$, it becomes apparent that if a piece of equipment can be constructed which takes a vector of numbers at its input, multiplies them by $M_n^{(i)}$ for some value of i between 1 and n, and outputs the resulting vector, that a cascade of n of these will accomplish the transformation $y = H_n z$. That such a piece of equipment not only can be constructed but is relatively simple is, at least in part, due to the simple structure of $M_n^{(i)}$.

Theorem: $M_n^{(j)}$ has 2 and only 2 nonzero elements in each row for any value of j between 1 and n.

Proof: $M_n^{(j)} = I_{n-j} \otimes (H_1 \otimes I_{j-1})$ for $1 \le j \le n$. So letting $A_j \triangleq H_1 \otimes I_{j-1}$ gives $M_n^{(j)} = I_{n-j} \otimes A_j$. Thus $M_n^{(j)}$ is a block diagonal matrix with the off-diagonal blocks all identically zero. Therefore, $M_n^{(j)}$ has 2 and only 2 nonzero elements per row if and only if A_j has 2 and only 2 nonzero elements per row. But

$$A_j = H_1 \otimes I_{j-1} = \begin{bmatrix} I_{j-1} & I_{j-1} \\ I_{j-1} & -I_{j-1} \end{bmatrix}$$

which is a block matrix. Any row of A_j is constructed from the juxtaposition of 2 rows of $\pm I_{j-1}$ and each row of any identity matrix has exactly one nonzero element. Thus each row of A_j has exactly 2 nonzero elements and hence so does $M_n^{(j)}$. Q.E.D.

Since $M_n^{(i)}$ is a $2^n \times 2^n$ matrix having exactly 2 nonzero elements per row, there are 2^n additions necessary to calculate z^i given z^{i-1} by $z^i = M_n^{(i)} z^{i-1}$. Calculating $y = H_n z$ requires n such operations, giving the total number of calculations necessary to produce y, given z, as $n2^n$. Letting $g_d = 2^n(2^n-1)$, the number of calculations necessary to produce y by direct evaluation of $y = H_n z$, and $g_f = n2^n$, the number of calculations necessary to produce y using the $M_n^{(i)}$ factor matrices, we have $g_f = \frac{n}{2^n-1} g_d$.

3.4. Implementation.

Since arithmetic manipulations on digital binary number require much less hardware if the numbers are in serial form rather than in parallel form, special purpose digital equipment is usually designed

for serial numbers. Parallel operation is used only where extreme calculation speed is required and then only when the much greater cost can be justified. For the symbol rates of current interest in practical communication systems, viz. Mariner Mars 1969, the current level of technology in digital hardware allows serial number representation in the decoder. There is nothing inherent in the mathematics or in the block diagram (to be discussed) of the decoder which would prohibit construction using parallel techniques if the need arose. However, the simplicity of the equipment is most striking in the serial case. Thus, in the following discussion, it will be assumed that the components of the vector x are available sequentially, one component every symbol time, τ, and are represented as serial binary numbers of q bits each. Since the decoder produces the components of y by adding all 2^n components of x, the equipment must have a digital world length of at least m bits where $m \geq n + q$. In this form, the components of x are represented as the q least significant bits of the m bit word.

The first operation to be performed is to compute $z = \Lambda_\eta x$. The i^{th} component of z is $z_i = \eta_i x_i$. One simple method of implementation is:

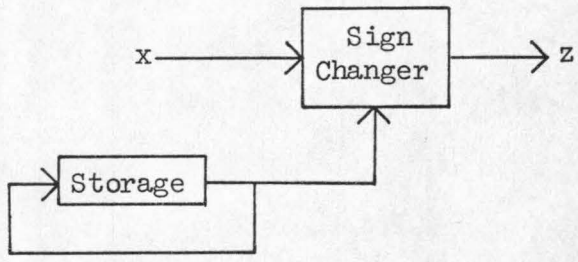

Figure 3.2

Here the coset leader, η, is stored in a n bit cyclic binary shift register which is shifted every τ seconds and timed so that η_1 is at the output of the shift register simultaneously with the appearance of the first bit of x_1 at the input of the sign changer. The sign changer block consists of one flip-flop and the necessary gating to change the sign of the j^{th} component of x if the j^{th} bit of η is -1.

Given z, the next task is to generate $z^1 = M_n^{(1)} z$, then $z^2 = M_n^{(2)} z^1$, $\cdots$, $y = M_n^{(n)} z^{n-1}$. Fortunately a general block diagram can be exhibited, Figure 3.3, which is a realization of $M_n^{(i)}$ for a general i. The memory element is 2^{i-1} m-bit words of serial memory which could be constructed, for example, from delay lines or integrated circuit shift registers. The gating structure, where $\cdot$ indicates an "and" gate and $+$ an "or" gate, serves to connect the input line to the memory input and the output line to the memory output when the signal W is "true". When W is "false", the gating connects the memory input to the subtractor output and the output line to the adder output. The signals W and $\overline{W}$ come from the opposite sides of a flip-flop which changes its state every $\tau 2^{i-1}$ seconds, i.e. every 2^{i-1} digital word times, and is timed so that W goes true simultaneously with the appearance of the first bit of the first component of z^{i-1} on the input line. This signal can be taken as the output of the i^{th} flip-flop in a binary counter of n flip-flops which is pulsed every τ seconds.

The serial binary adder and subtractor are very simple pieces of digital equipment. They take two binary numbers at their input and

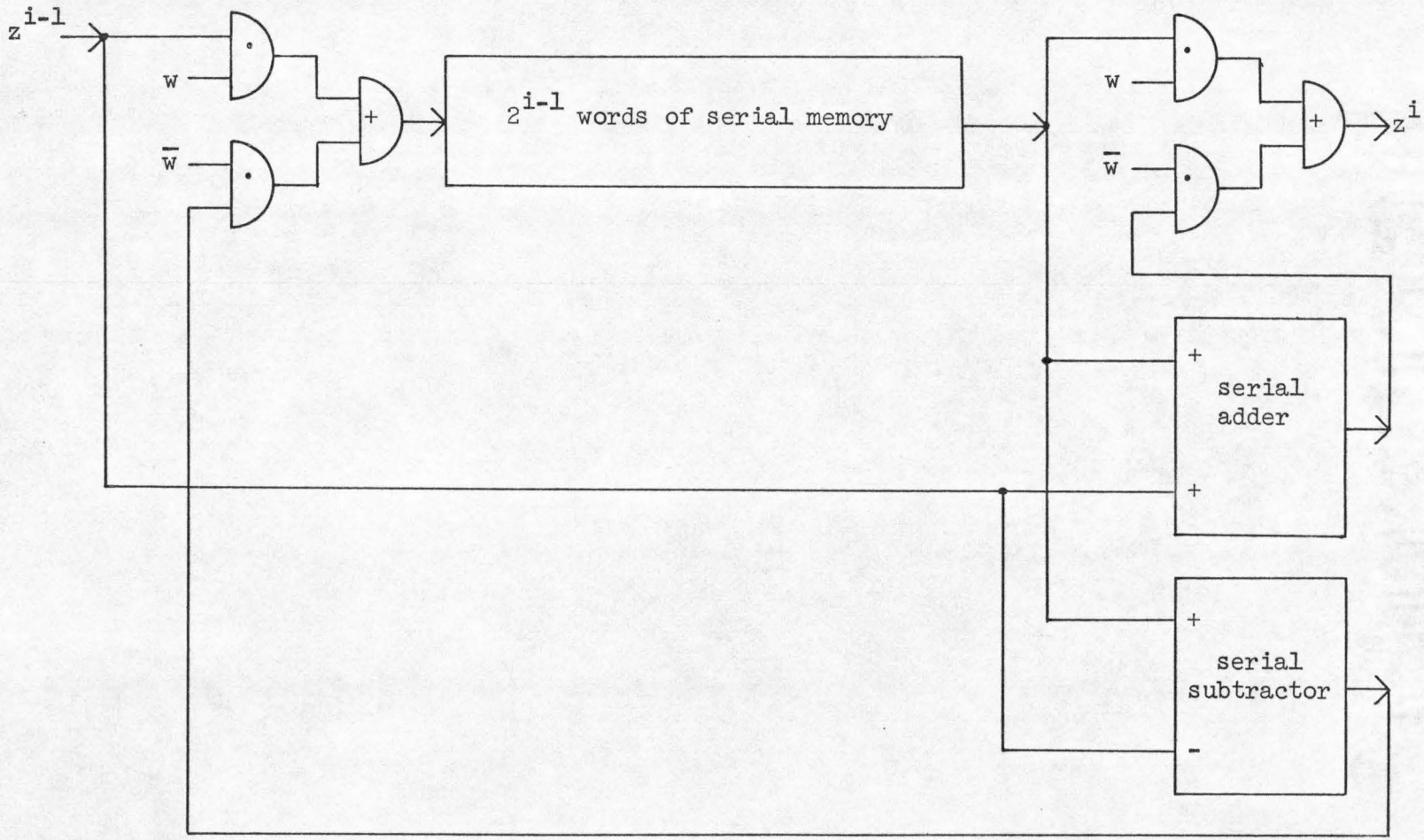
z^{i-1}
w
$\overline{w}$
2^{i-1} words of serial memory
w
$\overline{w}$
z^i
serial adder
serial subtractor

Figure 3.3

produce the sum or difference as binary numbers at their output.

To see that the i^{th} stage of the decoder really realizes $M_n^{(i)}$, consider the example for $n = 3$, $i = 2$ illustrated in Figure 3.1. For this example the memory consists of $2^{2-1} = 2^1 = 2$ words of memory, and the W flip-flop changes state energy 2τ seconds or every 2 digital word times. The result of allowing this stage to run through 7 input numbers is presented in Figure 3.4. Notice that after 7τ seconds the equipment is in the same state as after 3τ seconds except that the subscripts are now greater by 4. Thus, the decoder's i^{th} stage really implements the diagonal block of $M_n^{(i)}$, $H_1 \times I_{i-1}$, and repeats this operation time and again.

To actually construct the decoder, n pieces of equipment, or stages, would be constructed, following the block diagram of Figure 3.3, one for each value of i, $1 \leq i \leq n$. The operation of decoding would then be done by cascading the n stages of the decoder and the sign changer as shown in Figure 3.5. This decoder does the $n2^n$ arithmetic operations necessary to calculate $y = Kx$. Since there are τ seconds available for processing each of the m bit digital numbers making up x, the digital bit rate must be at least $\frac{m}{\tau}$ bits per second to do the decoding in real time. Expressed another way, if we let $s \triangleq$ the digital speed in bits per second, $r \triangleq$ the source data rate in bits per second then the maximum data rate this decoder can handle for orthogonal codes is $r_{max} = \frac{ns}{m2^n}$. For example, let $n = 7$, $q = 7$, $m = n + q = 14$, and $s = 10^7$ (implying 10 megacycle logic, which is only moderately fast at the present development of digital technology) then $r_{max} = 39062.5$ bits per second.

Time in τ sec.	W	$\overline{W}$	Memory Input Connection	Output Line Connection	Input Number	Memory Contents	Adder Output	Subtractor Output	Output Number
1	T	F	Input	Memory	z_1^1	?, ?	?	?	?
2	T	F	Input	Memory	z_2^1	z_1^1, ?	?	?	?
3	F	T	Subtractor	Adder	z_3^1	z_2^1, z_1^1	$z_1^1+z_3^1$	$z_1^1-z_3^1$	$z_1^1+z_3^1=z_1^2$
4	F	T	Subtractor	Adder	z_4^1	$z_1^1-z_1^3, z_2^1$	$z_2^1+z_4^1$	$z_2^1-z_4^1$	$z_2^1+z_4^1=z_2^2$
5	T	F	Input	Memory	z_5^1	$z_2^1-z_Y^1, z_1^1-z_3$	don't care	don't care	$z_1^1-z_3^1=z_3^2$
6	T	F	Input	Memory	z_6^1	$z_5^1, z_2^1-z_4^1$	don't care	don't care	$z_2^1-z_4^1=z_4^2$
7	F	T	Subtractor	Adder	z_7^1	z_6^1, z_5^1	$z_5^1+z_7^1$	$z_5^1-z_7^1$	$z_5^1+z_7^1=z_5^2$

Figure 3.4

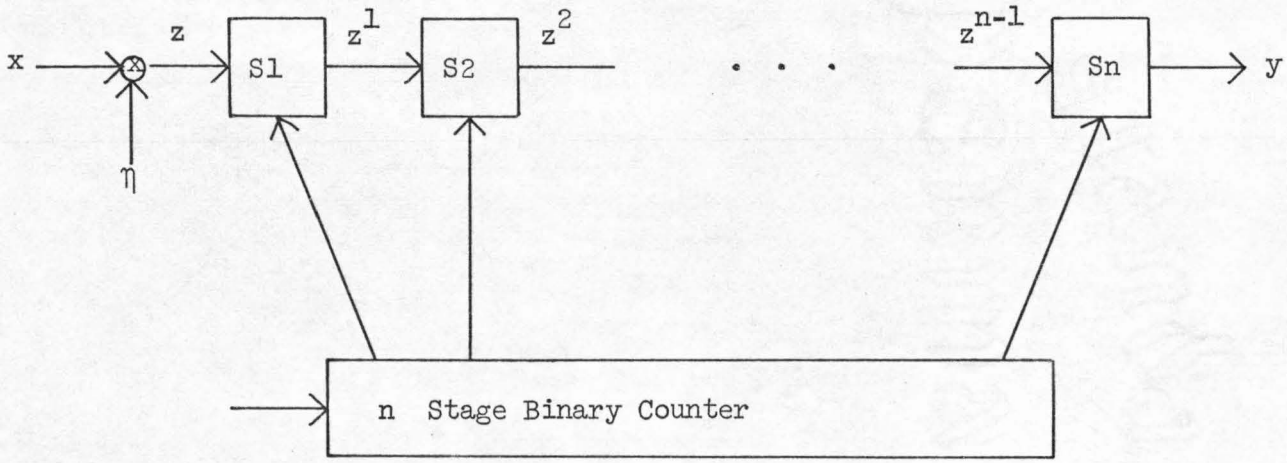

Figure 3.5

Considering r to be a fixed quantity and s the variable yields a different look at the decoder. Naturally, $s = \frac{rm2^n}{n}$. But, symbol rate $= 2^n \cdot$ code word rate $= 2^n \cdot \frac{1}{n} \cdot$ data rate $= \frac{2^n r}{n}$, thus $s = \frac{m}{\tau}$. In other words the logic speed need only be great enough to perform one addition of two m-bit binary numbers every τ seconds or every symbol time, τ.

This decoder is an optimum decoder with respect to the amount of memory involved. In Appendix B it is shown that any decoder for these codes which produces the components of y by taking linear combinations of the components of x and outputs the components of y sequentially must have at least 2^n-1 storage registers. This decoder involves 2^{i-1} storage registers in the i^{th} stage and there are n such stages, thus total storage $= \sum_{i=1}^{n} 2^{i-1} = 2^n-1$.

The final step in actually doing maximum-likelihood decoding of these codes is to determine the component of y which is the largest in magnitude. If, for example, y_5 were determined to be the largest, then the most likely code word to have been transmitted is the sequence of ± 1's of row k_5 of the dictionary matrix K. However, the quantity of immediate interest is the n bit data vector associated with the code word, not the code word itself. Recall, that the encoding algorithm, which is a 1:1 mapping between the 2^n possible data vectors of n bits each and the 2^n code words of K, may be assigned in an arbitrary fashion when all the data vectors are equally probable.

As the decoder in Figure 3.5 operates, the binary counter from which the W signals to each stage are derived, cyclically runs through all the possible 2^n combinations of its n binary bits.

Thus, on successive iterations of the operation of the decoder, whenever y_i appears at the output, the counter is always in the same state and this state is unique to the i^{th} component of y. The particular state in any actual decoder would depend upon the details of construction. In any case, this cyclic counter action serves to define a particular 1:1 mapping between the group of binary n vectors and the code words.

Hence if the encoding mapping is designated to be the mapping defined by the decoder counter, recovery of the actual data bits is very simple. As each component of y emerges from the output, it is examined to determine whether it is larger in magnitude than the previous largest (of course y_1 is always so chosen). Whenever a component of y is determined to be the largest in magnitude, the counter value is examined and recorded in place of that value associated with the previous largest. After all 2^n components of y have been so examined, the currently recorded counter value is the most likely transmitted binary data vector.

This operation could take place inside a general purpose computer attached to the decoder output, y, and to the counter, or it could be implemented with more special purpose equipment. The special purpose equipment required to do these operations is again quite simple. There would need to be 3 storage registers, 2 of length 2^n and 1 of length n, and a few decision elements to perform the magnitude comparison.

3.5. Unexploited Properties.

Recall that it has been proved that $M_n^{(i)} M_n^{(j)} = M_n^{(j)} M_n^{(i)}$. The direct implication of this result on the hardware of the decoder is that the n stages of the decoder may be connected together in any arbitrary order without disturbing the output. This would not be a surprising result if the various stages were identical, but as no two are alike, it does seem to merit examination. It appeals to engineering intuition that there should be some question, the answer to which would be to connect the stages of the decoder in some particular order.

This feeling that there were yet unexploited properties of the decoder led to consideration of the subject matter of Chapter IV. Unfortunately, the conclusions of that chapter only place one small constraint on the ordering of the stages of the decoder and this constraint does not seem to be something which can be proved in general. Therefore, there is still a strong feeling that further research into decoding techniques for first-order Reed-Muller codes would yield interesting results.

CHAPTER IV

ADJACENT SYMBOL MONITORING TECHNIQUES

4.1. Importance of Adjacent Symbol Monitoring.

After word synchronization has been obtained in an actual telemetry system, and decoding of the transmitted information is proceeding, it is possible that word syncronization may be lost. As mentioned by Stiffler in his doctoral thesis, if this loss of syncronization occurs it is most probably lost to a symbol adjacent to the formerly correct position [12]. One of the reasons that this can occur is due to the use of phase-lock loops in receiving systems.

In typical telemetry systems there is a high frequency carrier which is modulated by, perhaps, several lower frequency subcarriers. The subcarrier or subcarriers are, in turn, modulated by the information to be transmitted, in either coded or uncoded form. Reception of these signals, in current practice, employs phase lock loops. The first phase lock loop is used to track and remove the carrier. This loop is followed by a set of loops, one to track and remove each of the subcarriers. Finally, in a coded system there frequently would be a squaring or Costas loop to provide an estimate of symbol timing [15]. It is a characteristic of phase lock loops that they occasionally slip cycles. When this phenomenon occurs, it is most probable that they slip by one cycle. If this happens in the symbol timing loop, of course the correct syncronization position is immediately slipped to one of the adjacent symbols. In a telemetry system using the carrier and subcarrier loop frequency estimates to derive symbol timing, the cycle slipping in these loops may lead to a loss of synchronization.

It would, therefore, be advantageous to continuously monitor not only the synchronization position believed to be correct, but also each of the two adjacent positions. If this could be done, the decision to change estimates of the correct synchronization position could be made at the optimum moment. Additionally, no time would be lost, and hence data lost, while using a search algorithm to find the new phase estimate.

Of course, this adjacent symbol monitoring could be accomplished by constructing 3 decoders as discussed in Chapter III. However, the three received vectors for one symbol before and one symbol after the correct phase, call them ${}^{1}x$, and x^{1} respectively, each share a common set of $2^{n}-1$ components with the vector for the correct phase, x. Presumably, $y = Kx$ is already computed or at least the equipment necessary to compute y is at hand. The new task is to compute $y^{1} \triangleq Kx^{1}$ and ${}^{1}y \triangleq K{}^{1}x$. Again it seems intuitively reasonable that since the components of K are all ± 1's, and hence many of the individual pairwise calculations involved in computing y^{1} and ${}^{1}y$ must be identical with those in computing y, proper use of intermediate results and use of y itself may reduce the number of calculations required.

4.2. Formulation and Analysis.

In this chapter there is no use made of the Kronecker product of matrices. Therefore keeping track of the dimension of individual matrices is not so critical and the logarithmic dimension subscript notation will be dropped, i.e. H_{n} will be written simply H. All matrices will be assumed to be $2^{n} \times 2^{n}$ unless explicitly stated

otherwise.

The first task in formulating the adjacent symbol monitoring problem is to express x^1 and ${}^1x$ in convenient form. If a typical sequence of successive outputs from the integrator is enumerated as $s_o, s_1, s_2, s_3, \cdots$, and the current estimate of word synchronization identifies $x = (s_1, s_2, \cdots, s_{2^n})$ then ${}^1x = (s_o, s_1, \cdots, s_{2^n-1})$ and $x^1 = (s_2, s_3, \cdots, s_{2^n+1})$. (For simplicity in this chapter, define $m \triangleq 2^n$.) Define a permutation matrix P by $p_{m,1} = 1$, $p_{i,i+1} = 1$ for $i = 1, \cdots, m-1$, $p_{ij} = 0$ otherwise. Then $x^1 = Px + e^m(s_{m+1} - s_1)$ where e^i is a column vector with all elements zero except the i^{th}, which is $+1$. For example, let $n = 2$ then:

$$x^1 = \begin{bmatrix} s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} (s_5 - s_1)$$

The reason $p_{m,1}$ is chosen to be $+1$ instead of zero is to avoid having P be a singular matrix. Also, ${}^1x = P^T x + e^1(s_o - s_m)$.

The desired calculation may now be written

$$y^1 = Kx^1 = K[Px + e^m(s_{m+1} - s_1)]$$

$$= KPx + Ke^m(s_{m+1} - s_1)$$

and

$$^{1}y = K^{1}x = KP^{T}x + Ke^{1}(s_{o}-s_{m}))$$

As before, only linear combinations of the components of x, y, and the extra necessary outputs of the integrator are considered in trying to calculate y^{1} and ^{1}y. Clearly, y^{1} and ^{1}y are functions of x and the integrator outputs, and indeed the functions are linear. The adjacent symbol monitoring problem will be to investigate the functional dependence of y^{1} and ^{1}y upon x, the integrator outputs, and y, where attention is restricted to linear functions. The problem may then be formulated, for y^{1} as:

$$y^{1} = Q[Ay + Bx + v(s_{m+1}-s_{1})]$$

where the matrices A, B, and Q and the vector v are to be selected. Strictly, the matrix Q is unnecessary and could be dispensed with, however it simplifies the understanding of the problem to include it.

Substituting y = Kx, equating the two expressions for y^{1} and grouping terms gives:

$$KPx + Ke^{m}(s_{m+1}-s_{1}) = Q(AK+B)x + Qv(s_{m+1}-s_{1})$$

Equating like terms gives two characteristic equations for this problem:

$$KP = Q(AK+B) \quad \text{and} \quad Ke^{m} = Qv$$

If exactly the same steps were performed for ${}^{1}y$ the only difference in the above equations would be; P^T would replace P and e^1 would replace e^m. Therefore, only the problem involving y^1 will be discussed, since if it can be solved so can the problem for ${}^{1}y$.

Since all that is required is to monitor the adjacent symbol position the quantity of interest is $|y^1{}_j| = \max_{1 \le i \le m} \{|y^1_i|\}$. If it were necessary to actually decode the adjacent position, the value of j for which the maximum is achieved would be needed. However, since only monitoring is required and the value of j is unimportant, any permutation of y^1 would serve as well as y^1 itself. If Q is restricted to be a permutation matrix, then $z \triangleq Q^{-1}y = Q^T y$ serves as well as y^1 itself for the output of the monitor. Further, since it is the magnitude of the components of y^1, and hence z, that are of interest, the nonzero components of Q may be -1 instead of $+1$ if desired.

The second characteristic equation can now be solved for the vector v;

$$v = Q^{-1}Ke^m = Q^T Ke^m = Q^T H \Lambda_\eta e^m = \eta_m Q^T He^m$$

$$= \eta_m Q^T h^m$$

where h^m represents the m^{th}, last, column of H.

Selecting the matrices Q, A, and B is not so simple. Indeed, it is here that the first of two major problems preventing a general solution, or even a general approach to a solution, to adjacent symbol monitoring is encountered. The difficulty is that A and B are

required to be "simple" where a precise mathematical definition of "simple" cannot be given.

Of course there are many possible choices of the three matrices which will satisfy the first characteristic equation. Indeed given any signed premutation matrix Q and any other matrix for A, a feasible matrix B is obtained from $B = Q^{T}KP-AK$ since there is no requirement that either A or B be non-singular. However, to be useful, A and B must be of a form which may be implemented with less equipment than would be required to build a whole new decoder. In order to place such a constraint on the matrices, it would be necessary to somehow give a general mathematical characterization of all simple, constructable, linear digital machines. This task would make a formidable research project in its own right.

One way to proceed to obtain an answer to the problem is to over-constrain either or both of the matrices to forms for which simple mechanizations are known. Of course, taking this approach, while it does yield useful results, provides no guarantee that some different set of constraints would not give an even better result.

Examples of the type of constraints which have been investigated are; A must be an identity matrix, B an identity matrix, A must be a band matrix with bandwidth $k \ll m$, B a band matrix with $k \ll m$. Although Q has been constrained to be a signed permutation matrix, there are $m!\, 2^{m}$ such matrices possible. The particular Q matrix selected from among the possibilities will also have a profound effect upon the usefulness of the resulting A and B matrices. However, whatever constraints were selected, they all led both to

useful results and, ultimately, to the second major stumbling block. In view of this only the approach which gave the most useful results in the examples considered in Section 4.3 will be examined here.

Let A be an identity matrix, then the first characteristic equation becomes:

$$KP = Q(K+B) \qquad \text{or}$$

$$B = Q^{T}KP-K$$

Recall that B expressed the contribution of x to the evaluation of y^{1}. Any component of x that is used must be stored. So, to minimize storage, it seems reasonable to investigate the implications of requiring a column of B, say b^{s}, to be all zeros. Clearly, this will impose some requirements on Q which may or may not conflict with the constraint that Q be a signed permutation matrix.

Recall that $K = H\Lambda_{\eta}$ and let $Q^{T} = D\Lambda_{\eta}P^{-1}K^{-1}$. (Clearly any permutation matrix, Q^{T}, can be written in this form by picking $D = Q^{T}KP\Lambda_{\eta}$.) Then:

$$B = D\Lambda_{\eta}P^{-1}K^{-1}KP-H\Lambda_{\eta}$$

$$= D\Lambda_{\eta}-H\Lambda_{\eta}$$

Letting the j^{th} column of a matrix A be denoted as either $(A)^{j}$ or a^{j} and the i^{th} row be $(A)_{i}$ or a_{i}, gives, for the j^{th} column of B:

$$b^j = (D\Lambda_\eta)^j - (H\Lambda_\eta)^j$$

$$= \eta_j d^j - \eta_j h^j = \eta_j (d^j - h^j)$$

$$= 0 \quad \text{if and only if} \quad d^j \equiv h^j$$

Suppose then, that we specify $d^j = h^j$ for $1 \le j \le n$; i.e. pick $D = H$ and hence $B \equiv 0$. Then $Q^T = KP^{-1}K^{-1} = \frac{1}{m} H\Lambda_\eta P^T \Lambda_\eta H$. Note that $\Lambda_\eta P^T \Lambda_\eta = P_s$ where P_s is a matrix with $(P_s)_{ij} = 0$ if and only if $(P^T)_{ij} = 0$ and $(P_s)_{ij} = \pm 1$ when $(P^T)_{ij} = +1$. Therefore $Q^T = \frac{1}{m} (HP_s)H$, but for an arbitrary coset leader η, a row of HP_s is not in general some row (column) of H. Since the columns of H form a maximal orthogonal set in Euclidian m-space, $\frac{1}{m} HP_sH$ is not a permutation matrix. Thus, it is clear that there is no possibility of letting $d^j = h^j$ for all j. The best which can be done is to try to maximize the number of columns for which this choice can be made.

Now examine the requirement that the i^{th} row of Q^T be $\pm e_j$ for some j.

$$(Q^T)_i = (D\Lambda_\eta P^T \Lambda_\eta H^{-1})_i$$

$$= \frac{1}{m} (D)_i \, (\Lambda_\eta P^T \Lambda_\eta)H$$

$$= \frac{1}{m} d_i (\Lambda_\eta P^T \Lambda_\eta)H$$

Here it is necessary to introduce two operators. If $a = (a_1, a_2, \cdots, a_m)$ is a vector, then define a shift operator by $a^c = (a_2, a_3, \cdots, a_m, a_1)$ and $a^{-c} = (a_m, a_1, a_2, \cdots, a_{m-1})$. Note that $(a^c)^{-c} = (a^{-c})^c = a$.

This leads to several easily verified identities: $Pa = a^{c}$ $P^{T}a = a^{-c}$, $aP = a^{-c}$, $aP^{T} = a^{c}$, $P\Lambda_{a} = \Lambda_{a^{c}}P$, $P\Lambda_{a^{-c}} = \Lambda_{a}P$, $P^{T}\Lambda_{a} = \Lambda_{a^{-c}}P^{T}$, $P^{T}\Lambda_{a^{c}} = \Lambda_{a}P^{T}$. If a and b are both vectors then define their term by term product by $a \cdot b = (a_1 b_1, a_2 b_2, \cdots, a_m b_m)$. Using these operators we may now write:

$$(Q^{T})_i = \frac{1}{m} d_i P^{T} \Lambda_{\eta^{c}} \Lambda_{\eta} H$$

$$= \frac{1}{m} d_i^{c} \cdot \eta^{c} \cdot \eta \, H$$

$$= e_j \quad \text{for some } j \text{ if and only if:}$$

$$d_i^{c} \cdot \eta^{c} \cdot \eta = h_j$$

or:

$$d_i^{c} = \eta^{c} \cdot \eta \cdot h_j$$

or:

$$d_i = \eta \cdot \eta^{-c} \cdot h_j^{-c}$$

Specifying a $d^{j} = h^{j}$ for some j implies that $d_{ij} = h_{ij}$. Thus, the question of interest is to select as many bits of d_i as possible as the i^{th} bit of h^{j} while still satisfying the preceding equation.

Now the second major difficulty is encountered. In order to proceed further it is necessary to know something about the properties of coset leaders which generate cosets of the first-order Reed-Muller codes which are comma-free. Unfortunately very little is known at this time. This question is presently being actively investigated by, among others, L. Baumert and H. Rumsey [13]. As there is insufficient

knowledge of the general properties of comma-free coset leaders to be able to deal with the selection of d_i analytically, it seems hopeless to try to say in general what the maximum number of zero columns is for B. Lacking any general theory relating to this, taking any particular coset leader leads to an enormously complicated, not very enlightening bit matching task. However, there is a different, less elegant but very straight-forward approach to selecting B and Q when $A = I$ which leads to useful answers.

As part of his work Baumert has found coset leaders for all the cosets for $n = 4$ and $n = 5$ which have maximum index of comma freedom [13][14]. For $n = 4$, the maximum index of comma freedom is 2 and there are 372 distinct cosets which achieve this bound. For $n = 5$ the maximum index is 7 and there are 32 distinct cosets achieving the bound. This provides a usefully large number of examples for evaluating the effectiveness of the following approach.

Instead of trying to get the maximum number of zero columns in B, the problem of inserting a large number of zeros into B can be approached by maximizing the number of zeros in each row (and hoping that they will naturally tend to fall into columns). Recall that:

$$B = Q^T KP - K$$

Thus, asking for the maximum number of zeros in some row of B, say b_i, is equivalent to trying to pick Q^T such that $(Q^T KP)_{ij} = k_{ij}$ for the largest number of values of j. Since Q is a signed

permutation matrix, this is then equivalent to finding that value of ℓ for which for the largest number of values of j either

$$(KP)_{\ell j} = k_{ij}$$

or

$$(KP)_{\ell j} \neq k_{ij}$$

This may be simply done by evaluating the matrix $KP^T K^T \triangleq F$. Then

$$f_{ij} = (k_i, ((KP)^T)^j) = (k_i, (KP)_j)$$

and Q^T may be selected by:

$$q_{ij} = 0 \quad \text{if} \quad |f_{ij}| < |f_{ik}| \quad \text{for some} \quad k, \quad 1 \leq k \leq m$$

$$q_{ij} = +1 \quad \text{if} \quad f_{ij} > 0 \quad \text{and} \quad |f_{ij}| \geq |f_{ik}| \quad \text{for all} \quad k, \quad 1 \leq k \leq m$$

$$q_{ij} = -1 \quad \text{if} \quad f_{ij} < 0 \quad \text{and} \quad |f_{ij}| \geq |f_{ik}| \quad \text{for all} \quad k, \quad 1 \leq k \leq m$$

In those cases where the above process results in exactly one nonzero element per row, it is a remarkable and as yet unexplained fact (related of course to the general structure of comma-free coset leaders) that the resulting matrix is a signed permutation matrix in every case tried. In those cases where the process results in more than one nonzero element per row it is then necessary to delete

sufficient nonzero entries to make Q^T a signed permutation matrix. Again it is true that in all cases tried it is possible to perform the deletions so that a signed permutation matrix results, although some selections may result in more identically zero columns of B than others.

Since the largest magnitude element in a row of F is always greater than zero in magnitude, there are always more than 2^{n-1} zeros in each row of B. And since B is the difference between two matrices, all of whose elements are ± 1, the only possible values for elements of B are 0, or ± 2.

4.3. An Adjacent Symbol Monitor for $n = 5$.

The previously discussed technique has been applied to all 32 maximum index cosets for $n = 5$ and to several of the maximum index cosets for $n = 4$. The example given here is one for which the maximum magnitude element in each row of F is unique. For this example the coset leader was:

$$\eta = (1,-1,-1,-1,1,-1,-1,-1,1,1,-1,-1,1,1,-1,1,-1,1,-1,-1,1,-1,-1,-1,-1,$$
$$-1,1,-1,1,-1,-1,1)$$

The resulting matrix Q^T is shown in Figure 4.1. This implies the matrix B is as shown in Figure 4.2. Notice that there are 20 identically zero columns in B. Thus only 12 components of x need to be stored for use in computing y^1. Notice, also, that columns $1,5,9,\cdots,29$ have nonzero elements whose values are constant over sets of 4 rows, this can be exploited to further reduce the equipment needed. Since the

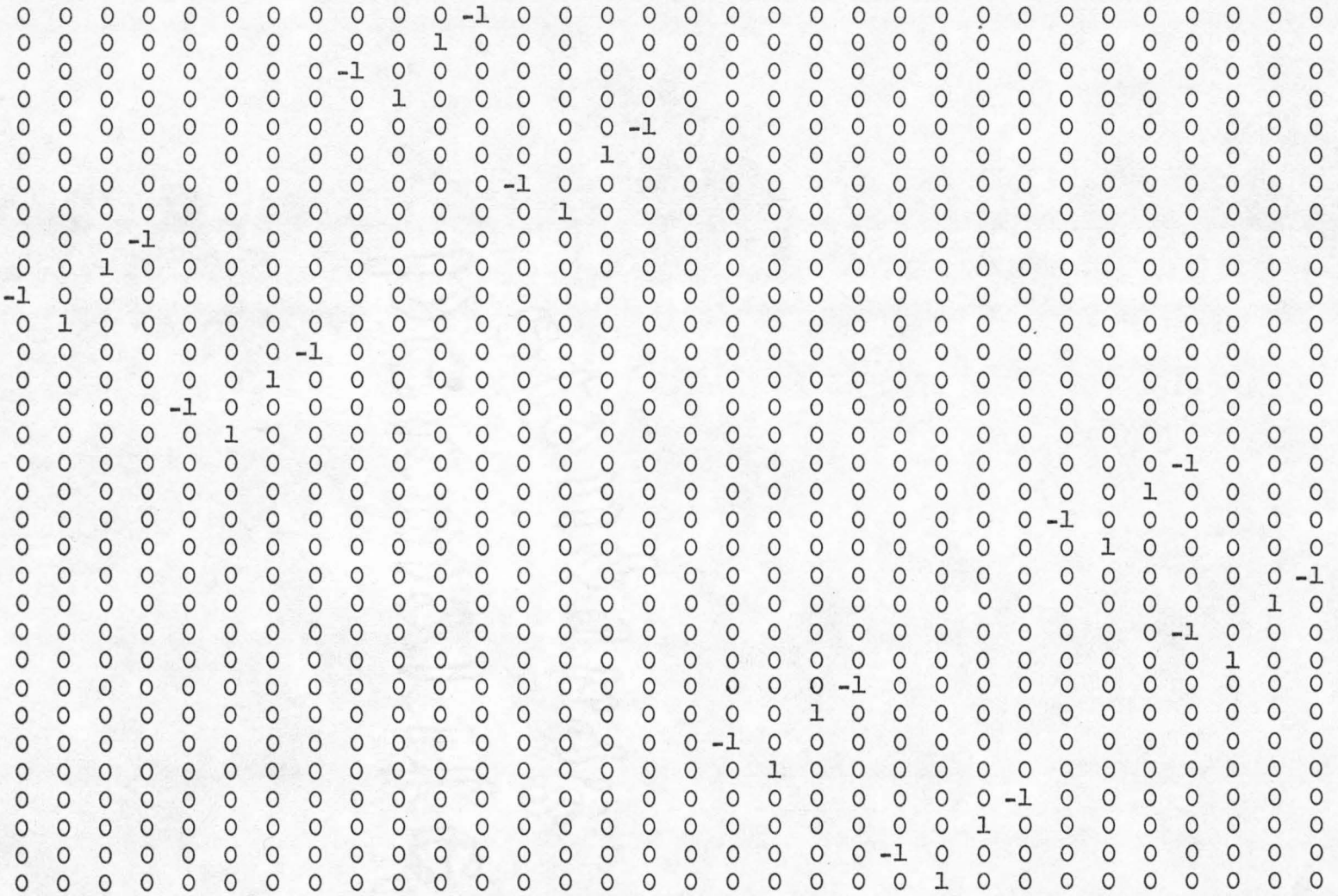

Figure 4.1

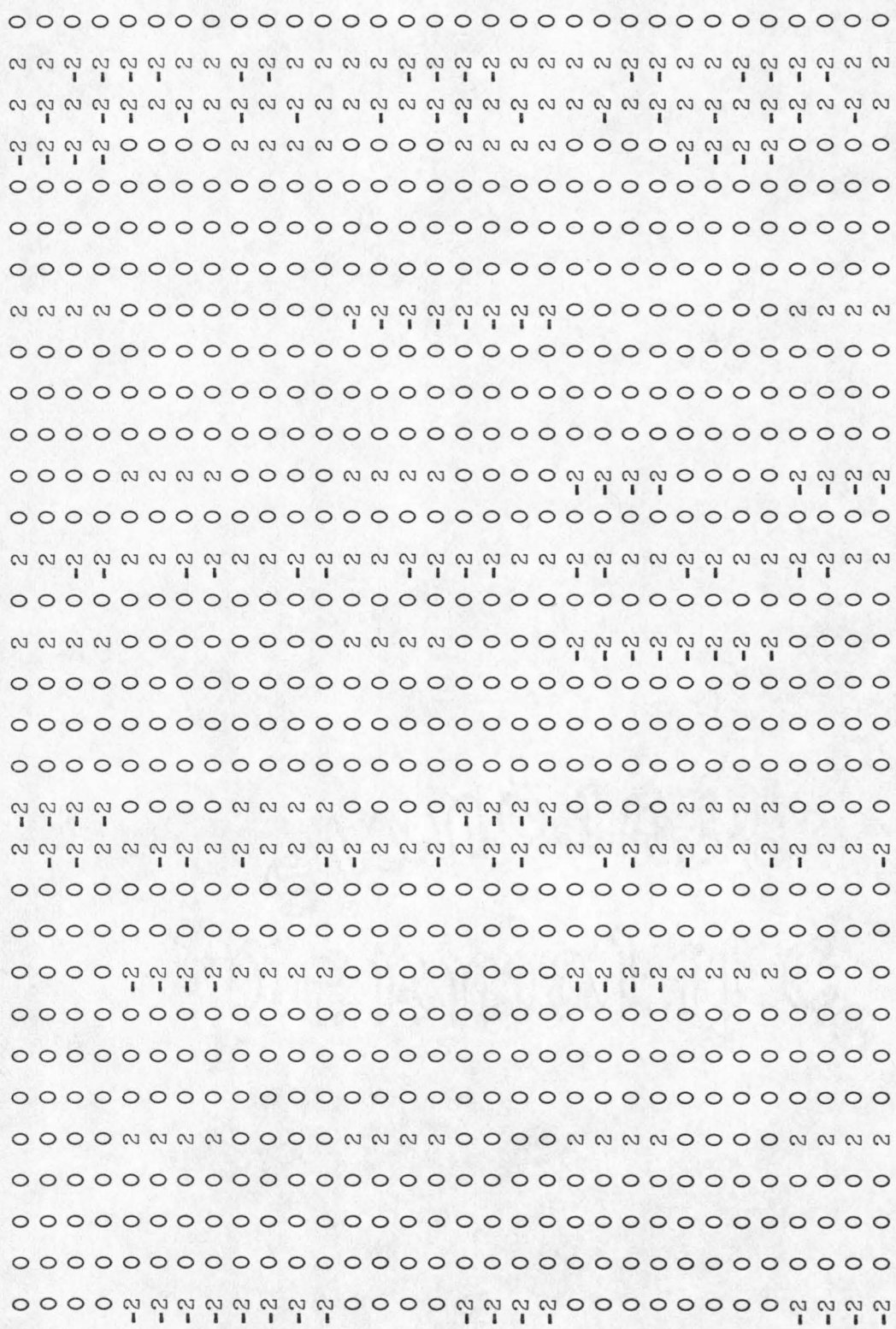

Figure 4.2

third stage of the decoder has a delay of 4 digital words, its outputs include $z_1 \pm z_5$, $z_9 \pm z_{13}$, $\cdots$, $z_{25} \pm z_{29}$. Hence its outputs include all (both) possible magnitudes of $x_1 \pm x_5$, $\cdots$, $x_{25} \pm x_{29}$ and some of the computation work can be saved by placing the third decoder stage first in line. Of course the details of these matrices, and hence their detailed implementation, depend upon which coset leader is being used.

Figure 4.3 shows a block diagram of the adjacent symbol position monitor. In this particular example there are 24 storage registers and 11 adder-subtractors in the arithmetic block. The details of the gating structure to select those components of x to be stored, the outputs of S_3 to be used, etc., can be obtained by standard digital design practice. Of course, the particular details of the structure of the monitor depend heavily on the particular coset leader chosen.

4.4. Conclusions and Future Research.

As has been seen, the question of how best to do adjacent symbol position monitoring is hampered by insufficient knowledge of the nature of coset leaders and of "simple" digital machines. In spite of these difficulties an answer, which of course may not be and probably is not optimum in any sense, can be reached. Unfortunately, the answer obtained does not place a heavy constraint on the order in which the stages of the decoder must be corrected.

Any future progress in either the understanding of coset leaders or the characterization of digital machines in terms of their input-output properties would open new possibilities on the subject at hand. Failing any new progress in either of these areas, it may be that a

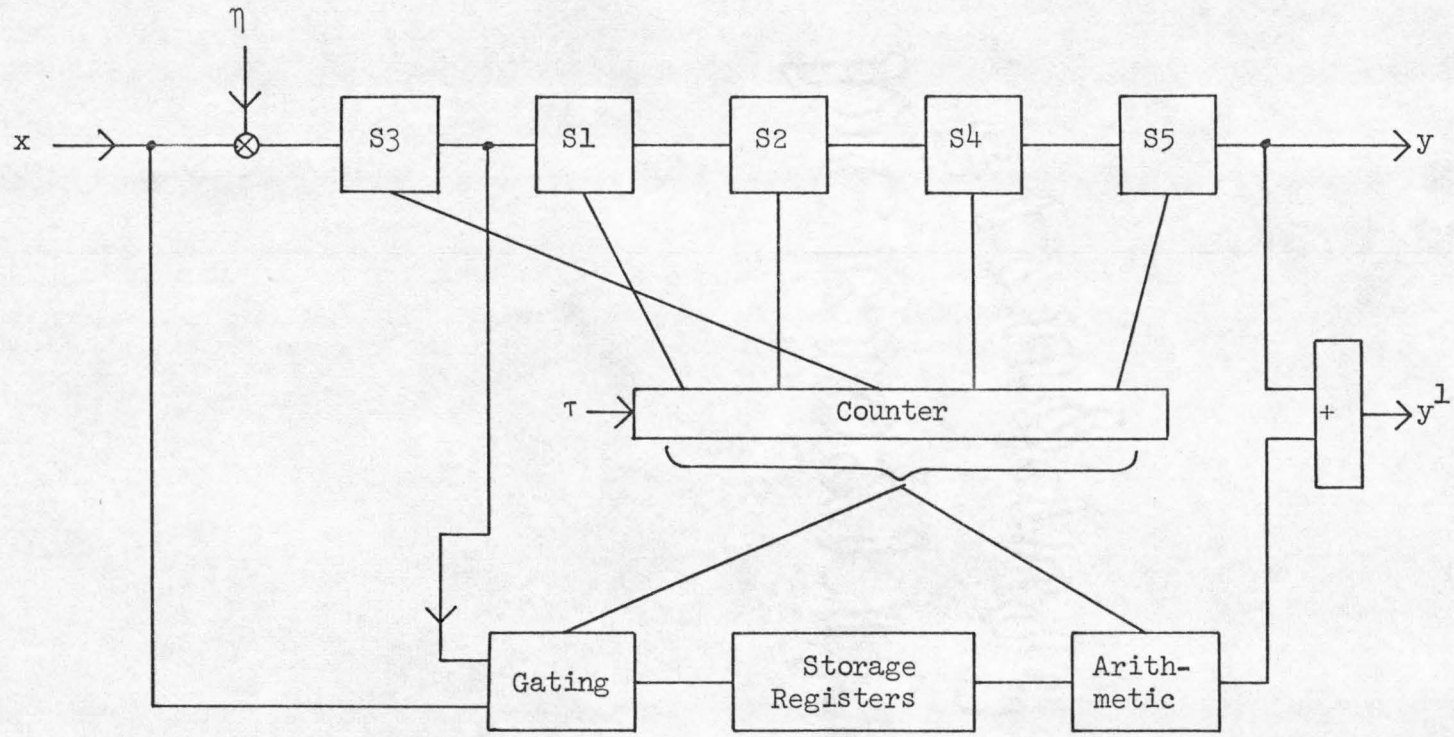
η
x
S3
S1
S2
S4
S5
y
τ
Counter
+
y1
Gating
Storage Registers
Arith-metic

Figure 3.2

different approach to selecting the matrices A, B, and Q might yield useful generalizable results.

Conceivably a whole new approach to the problem could be taken. One such approach which has been briefly considered but discarded as beyond the scope of the present work is as follows. There is really no particular reason for considering the problems of computing y^1, y, and ^1y separately. It may be that the output of the integrator should be considered as a length $m + z$ vector which is then operated upon to produce the required outputs.

CHAPTER V

CONCLUSIONS

The problem of doing the mathematical manipulations necessary to do maximum likelihood decoding of any coset of the first-order Reed-Muller codes rapidly and economically has been solved in general. The decoder exhibited is optimum in that it requires the minimum possible amount of storage. The technique used to solve the problem and construct the decoder has been extended to include codes other than the first-order Reed-Muller.

The related problem of adjacent symbol position monitoring has been considered. It has been demonstrated that a general solution to this problem awaits further development of two other areas of research. Failing a general solution, an approach which, though it does not provide generalizable insight, yields satisfactory answers in all cases examined, is presented. The resulting monitor, while not provably optimum in any sense, is simpler than building a separate decoder to monitor the adjacent symbol position.

The most striking indication that there is still some worthwhile understanding to be achieved through further research in this area comes from the decoder itself. Thus far, the only limitation that can be imposed upon the ordering of the stages of the decoder arises from a heuristic approach to the adjacent symbol monitoring problem. It appeals, very strongly, to engineering intuition, that finding a question which will impose a strict ordering to the decoder stages would yield further useful insight into the decoding problem.

APPENDIX A

MATHEMATICAL EXTENSIONS

The mathematics of Chapter III can be extended to a larger class of codes.

Let m and i be integers and for $1 \le i \le m$ let $n_i \ge 1$ be an integer. Let A_i be a square, $n_i \times n_i$, matrix of real numbers. For integers i and j, define an integer valued function $P(i,j)$ by: $P(i,j) = \prod_{i \le k \le j} n_k$ for $1 \le i \le j \le m$, $P(i,j) = 1$ for $i > j$. Let $I_{P(i,j)}$ denote the $P(i,j) \times P(i,j)$ identity matrix. (For $P(i,j) = 1$, $I_{P(i,j)} = 1$.) Finally, let a matrix K be defined by: $K = A_1 \otimes A_2 \otimes \cdots \otimes A_{m-1} \otimes A_m$. The dimension of K is $P(1,m) \times P(1,m)$ since $P(1,m) = \prod_{k=1}^{m} n_k$.

The matrix K could be the $2^m \times 2^m$ Hadamard matrix of Chapter III, for example, by letting $n_i = 2$ and $A_i = H_1$ for all i. For coding purposes, the elements of A_i are usually restricted to ± 1, however, that restriction is unnecessary for establishing the following results.

Define: $N^{(i)} \triangleq I_{P(1,i-1)} \otimes A_i \otimes I_{P(i+1,m)}$ for $1 \le i \le m$.

Since $P(1,i-1) \cdot n_i \cdot P(i+1,m) = \prod_{k=1}^{i-1} n_k \cdot n_i \cdot \prod_{k=i+1}^{m} n_k = \prod_{k=1}^{m} n_k = P(1,m)$, $N^{(i)}$ is $P(1,m) \times P(1,m)$. This definition gives a matrix related to K in the same way that $M_n^{(i)}$, defined in Chapter III, was related to H_n, the $2^n \times 2^n$ Hadamard matrix. As might be expected,

the same type of commutivity result holds.

Theorem: $N^{(i)}N^{(j)} = N^{(j)}N^{(i)}$ for $1 \le i \le m$, $1 \le j \le m$.

Proof: If $i = j$, there is nothing to prove. Assume that $i < j$, then:

$$N^{(i)}N^{(j)} = (I_{P(1,i-1)} \otimes A_i \otimes I_{P(i+1,m)})(I_{P(1,j-1)} \otimes A_j \otimes I_{P(j+1,m)})$$

$$= ([I_{P(1,i-1)} \otimes A_i \otimes I_{P(i+1,j-1)}] \otimes I_{P(j,m)})(I_{P(1,j-1)} \otimes [A_j \otimes I_{P(j+1,m)}])$$

$$= I_{P(1,i-1)} \otimes A_i \otimes I_{P(i+1,j-1)} \otimes A_j \otimes I_{P(j+1,m)}$$

$$= (I_{P(1,i)} \otimes [I_{P(i+1,j-1)} \otimes A_j \otimes I_{P(j+1,m)}])([I_{P(1,i-1)} \otimes A_i] \otimes I_{P(i+1,m)})$$

$$= (I_{P(1,j-1)} \otimes A_j \otimes I_{P(j+1,m)})(I_{P(1,i-1)} \otimes A_i \otimes I_{P(i+1,m)})$$

$$= N^{(j)}N^{(i)}$$ Q.E.D.

This theorem shows that, as before, the order in which the matrices appear in products of $N^{(i)}$ for various values of i is unimportant. The following theorem gives the result of multiplying together an arbitrary subset of these matrices.

Theorem: For $t \le m$ define a set R by:

$R \triangleq \{r_1, r_2, \cdots, r_t\} \subset \{1, 2, \cdots, m\}$ where if $i < j$ then $r_i < r_j$. Then:

$$\prod_{i\in R} N^{(i)} = I_{P(1,r_1-1)} \otimes A_{r_1} \otimes I_{P(r_1+1,r_2-1)} \otimes A_{r_2} \otimes \cdots$$

$$\otimes A_{r_t} \otimes I_{P(r_t+1,m)}$$

Proof: This theorem is proved by induction on t. When $t = 1$, $\prod_{i\in R} N^{(i)} = N^{(r_1)}$, which is in the required form by definition. Now, assume the result is true for an arbitrary t and prove it for $t + 1$. So let R have $t + 1$ elements, i.e. $|R| = t + 1$ then $\prod_{i\in R} N^{(i)} = N^{(r_1)} \prod_{i\in R-\{r_1\}} N^{(i)}$. But $|R-\{r_1\}| = t$ thus the theorem holds by assumption for $i \in R - \{r_1\}$.

$$\prod_{i\in R-\{r_1\}} N^{(i)} = I_{P(1,r_2-1)} \otimes A_{r_2} \otimes \cdots \otimes A_{r_{t+1}} \otimes I_{P(r_{t+1}+1,m)}$$

$$= I_{P(1,r_1)} \otimes I_{P(r_1+1,r_2-1)} \otimes A_{r_2} \otimes \cdots \otimes A_{r_{t+1}}$$

$$\otimes I_{P(r_{t+1}+1,m)}$$

Thus:

$$\prod_{i\in R} N^{(i)} = [(I_{P(1,r_1-1)} \otimes A_{r_1}) \otimes I_{P(r_1+1,m)}][I_{P(1,r_1)} \otimes$$

$$(I_{P(r_1+1,r_2-1)} \otimes A_{r_2} \otimes \cdots \otimes A_{r_{t+1}} \otimes I_{P(r_{t+1}+1,m)})]$$

$$= I_{P(1,r_1-1)} \otimes A_{r_1} \otimes I_{P(r_1+1,r_2-1)} \otimes A_{r_2} \otimes \cdots$$

$$\otimes A_{r_{t+1}} \otimes I_{P(r_{t+1}+1,m)} \qquad \text{Q.E.D.}$$

In particular if $r_i = m+i-t$ this theorem gives $\prod_{i\epsilon R} N^{(i)} = I_{P(1,m-t)} \otimes A_{m+1-t} \otimes A_{m+2-t} \otimes \cdots \otimes A_m$ and if $t = m$, $\prod_{i\epsilon R} N^{(i)} = A_1 \otimes A_2 \otimes \cdots \otimes A_m = K$. Again, to get the results for the first-order Reed-Muller codes we let $n_i = 2$ and $A_i = H_1$ for all i. In this case the theorem gives $\prod_{i\epsilon R} N^{(i)} = I_{P(1,m-t)} \otimes H_t$ and for $t = m$, $\prod_{i\epsilon R} N^{(i)} = H_n$ as before.

The factorization of the code matrix K into a product of matrices of the form of $N^{(i)}$, leads to a decoder having desirable properties in the case of the first-order Reed-Muller codes. Whether or not this approach leads to a useful result in any other case depends upon the elements and dimensions of the component matrices $A_1, \cdots, A_m$.

APPENDIX B

MEMORY REQUIREMENTS FOR DECODERS

If a piece of digital equipment is constructed which does the operation $y = H_n z$, with certain restrictions on its mode of operation, there will be a lower bound on the number of storage registers required. If the machine is required to form components y only by taking linear combinations of components of z and is required to output the components of y one by one, then there must be at least 2^n-1 storage registers. This limit can be achieved only if the first component of y begins emerging from the output as the last component of z is entering the input. If, instead, it is required that the first component of y not appear at the output until after the last component of z has entered the input, then at least 2^n storage registers are required.

These facts are most easily seen by an induction type argument. Consider first the 2×2 case of $y = H_1 z$. Explicitly, $y_1 = z_1 + z_2$ and $y_2 = z_1 - z_2$. When z_1 appears at the input neither y_1 nor y_2 may yet appear at the output so z_1 must be stored. This requires $2^1-1 = 1$ storage registers. When z_2 appears at the input both y_1 and y_2 may be computed. If y_1 is immediately emitted from the output only y_2 need be stored in the single storage register. If, however, y_1 is not to be emitted until a new number appears at the input, then both y_1 and y_2 or both z_1 and z_2 must be stored, which of course requires 2 storage registers.

Assume that the result has been established for the $2^n \times 2^n$ case

$y = H_n z$ and consider the problem $y = H_{n+1} z$. Recall that $H_{n+1} = H_1 \otimes H_n = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$. Write $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ where z_1, z_2, y_1, and y_2 are all 2^n vectors, then $y = H_n z$ may be written $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$. Thus, $y_1 = H_n z_1 + H_n z_2$ and $y_2 = H_n z_1 - H_n z_2$. By assumption $H_n z_1$ requires 2^n storage registers since the components of z_2 are needed before any components of y can be emitted. Likewise, computing $H_n z_2$ requires $2^n - 1$ storage registers or 2^n storage registers, depending on which output condition is being met. In either event $2^n + 2^n - 1 = 2^{n+1} - 1$ or $2^n + 2^n = 2^{n+1}$, thus establishing the result for the $2^{n+1} \times 2^{n+1}$ operation $y = H_{n+1} z$.

Thus, by induction, any suitably constrained piece of equipment which does the mathematical manipulation involved in decoding a first-order Reed-Muller code of length 2^n must have at least $2^n - 1$ storage registers.

REFERENCES

[1] Shannon, C. E., and Weaver, W., The Mathematical Theory of Communication, University of Illinois Press, Urbana, Illinois, 1964.

[2] Wozencraft, J. M., and Jacobs, I. M., Principles of Communication Engineering, John Wiley and Sons, Inc., New York, 1967, pp. 405-454.

[3] Goethals, J. M., and Delsarte, P., "On a Class of Majority-Logic Decodable Cyclic Codes", IEEE Transactions on Information Theory, Vol. IT-14, No. 2, March 1968, pp. 182-188.

[4] Massey, J. L., Threshold Decoding, MIT Press, Cambridge, Mass., 1963.

[5] Reed, I. S., "A Class of Multiple-Error-Correcting Codes and the Decoding Scheme", IRE Transactions on Information Theory, Vol. IT-4, September 1954, pp. 38-49.

[6] Golomb, S. W., editor, Digital Communications with Space Applications, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964, pp. 108-111.

[7] Gale, D., The Theory of Linear Economic Models, McGraw-Hill Book Company, Inc., New York, 1960, pp. xii.

[8] Stiffler, J. J., "Self-Synchronizing Binary Telemetry Codes", Doctoral Thesis, California Institute of Technology, Pasadena, California, 1962.

[9] Stiffler, J. J., Theory of Synchronous Communication, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, to be published, Chapter 14.

[10] Stiffler, J. J., "Self-Synchronizing Binary Telemetry Codes", Doctoral Thesis, California Institute of Technology, Pasadena, California, 1962, pp. 60-66, 117-121.

[11] Bellman, R., Introduction to Matrix Analysis, McGraw-Hill Book Company, Inc., New York, 1960, pp. 227-228.

[12] Viterbi, A. J., Principles of Coherent Communication, McGraw-Hill Book Company, Inc., New York, 1966, pp. 286-292.

[13] Baumert, L. D., and Rumsey, H., "The Index of Comma Freedom for the Mariner '69 High Rate Telemetry Code", Space Programs Summary No. 37-46, Vol. IV, Jet Propulsion Laboratory, Pasadena, California, August 1967, pp. 221-226.

[14] Baumert, L. D., private communication, February 1968.

[15] Costas, J. P., "Synchronous Communications", Proceedings of the IRE, Vol. 44, No. 12, December 1956, pp. 1713-1718.