NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

MAJORITY LOGIC DECODING OF A CLASS OF COMPOSITE CODES

BY

V. MIMIS

THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES, UNIVERSITY OF OTTAWA, IN PARTIAL FULFILMENT FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITY OF OTTAWA

OTTAWA, ONTARIO

JUNE, 1981

ABSTRACT

Traditionaly, block codes have been used as an error correcting tool to ensure reliable and unambiguous exchange of digital information between a transmitter and a receiver. The emergence of spread spectrum systems has created a need for long codes, to provide for the spectral spreading and disspreading of the information sequence. In the most commonly used form of spread spectrum system; namely the Direct Sequence Spread Spectrum System, the spreading code is used to provide for the synchronization of the receiver with the transmitted signal. A valuable system performance criterion is the "first attempt probability" of successful synchronization.

Among known various synchronization techniques in a low signal to noise environment "Majority Logic Decoding Synchronization" provides the highest probability of "first attempt synchronization".

A useful class of codes in the context of spread spectrum systems is the class of composite codes. Each word is constructed according to the rule:

$$V^{(\mu)} = \sum_{i=0}^{\mu} V_i(x) \frac{1+x^n}{1+x^{n_\mu}}$$

where $V^{\mu}(x)$ is a word of the composite code and $V_i(x)$, $i = 1, 2, \ldots, \mu$ is a word of the component code. The aim

of this work is to prove that this class of composite codes can be majority logic decoded under certain conditions and as a result "Majority Logic Synchronization" can be used as a synchronization technique.

ACKNOWLEDGEMENT

The author is grateful to Professor S.G.S. Shiva, his thesis advisor, for his counsel and support throughout the author's graduate studies.

The author pursued most of his graduate studies while being a full time employee of Telesat Canada. Telesat's policy of encouraging advanced studies is gratefully acknowledged. The author has also gained from his association with some of the highly talented professionals at Telesat. Finally, the author wishes to thank L. Jones for reading the manuscript and making useful suggestions and Mrs. Y. Rowe for her typing.

TABLE OF CONTENTS

Chapter			Page
	ABSTRACT		ii
	ACKNOWLEDGEMENT		iv
	TABLE OF CONTENTS		v
I	INTRODUCTION		1
	1.1	General	1
	1.2	Spread Spectrum Systems	3
	1.3	Spread Spectrum Sequences	5
	1.4	Binary Random Sequences	6
	1.5	Pseudo Random Sequences	8
	1.6	Linear Sequences	8
	1.7	Maximal Length Sequences	11
	1.8	Legendre Sequences	11
	1.9	Non Linear Shift Register Sequences	14
	1.10	Jet Propulsion Laboratories (JPL) Composite Codes	15
	1.11	Code Acquisition and Synchronization	16
	1.12	Problem Definition	18
	1.13	Thesis Outline	18
II	MAJORITY LOGIC DECODING		20
	2.1	Linear Block Codes	20
	2.2	Majority Logic Decoding	22
	2.3	Majority Logic Decodable Codes	29
		2.4.1 Maximum Length Codes	29
		2.4.2 Difference Set Codes	30
		2.4.3 Classes of L Step Majority Logic Decodable Codes	31
III	COMPOSITE CODES		32
	3.1	Introduction	32
	3.2	Construction of a Class of Composite Codes	32
	3.3	Majority Logic Decoding of Composite Codes	34
IV	CONCLUDING REMARKS		66
	APPENDIX A		68
	REFERENCES		84

CHAPTER I

INTRODUCTION

1.1 General

Our modern world has an ever growing demand for information exchange at all levels, ranging from mass media communications, to highly encrypted military data communications. The object of communications engineering is to satisfy this demand by effecting reliable and unambiguous transmission of information from one point to the other by means of electrical signals.

Information is subjected to various levels of processing before being rendered into a form suitable for transmission through the communications channel which may be a pair of wires, an optic fiber or the free space. Information can be transmitted in an analogue form such as voice, in a digital form such as digitized voice, or as data from a computer. Information transmitted through a channel in a digital form can be received as reliably as desired, without reducing the rate of information, by utilizing the appropriate bandwidth and/or power.

Such a channel is characterized by a quantity called the channel capacity (C).

The error probability at the receiver can be made arbitrarily small by proper encoding and decoding of the data,

when, and only when, the rate (R) of information transmission is less than C.

Communications engineers, continously strive to utilize the available power and bandwidth in the most effective way, in order to achieve reliable transfer of information from the transmitter to the receiver.

As a result, a number of bandwidth and/or power efficient modulation and coding schemes have been developed.

The current trend is to minimize the bandwidth used to provide the communication link either through improved techniques or through an administrative decision.

Reducing assigned bandwidth has its limitations. Over reduction of utilized bandwidth can cause degradation of system performance or result in a requirement for more costly equipment.

Another approach which may lead to better frequency spectrum utilization is a new and evolving technique known as "spread spectrum". It is based on a principle which is in antithesis to reducing bandwidth. Instead it utilizes modulated signal spectra that employ large transmission bandwidths to send information requiring a bandwidth much less than is transmitted.

However, spread spectrum techniques, through the properties of coded modulation, can provide systems which produce low interference to other systems, have high interference rejection

capability, and provide multiple access capability. Finally, spread spectrum techniques are probably the only solution to the deep space communication problem where long propagation paths are encountered and low power transmitters are utilized, resulting in a very low received signal to noise ratio (SNR).

1.2 Spread Spectrum Systems (S.S.S.)

Spread spectrum systems [2] utilize a total bandwidth much wider than the required for transmission of the information bearing data bits, with no incurred loss in communication performance due to the increase in bandwidth. A classical measure of a digital communications system's performance is the ratio E_b/N_o [25] where E_b corresponds to the received energy of each data bit and $N_o/2$ is the power spectral density of the receiver band pass noise which is assumed to be Gaussian with a flat spectral density. The fact that makes the ratio E_b/N_o an important measure of a communication system's performance is that it is independent of the actual bandwidth utilized for the transmission of the signal.

Spread spectrum systems [3] are based on three basic approaches. (1) Direct Sequence (DS), (2) Frequency Hopping (FH), and (3) Time Hopping (TH). There are other forms of SSS which are derived from one of the basic approaches or from their hybrid combinations.

The DS signal derives its name from the fact that each data bit consists of a pseudo-random digital code sequence. Each digit of the code is referred to as a "chip". The DS signal is characterized by a continuous spectrum over the bandwidth but with spectral shaping controlled by the digital code stream.

A data source produces binary data at a rate of R bits per second. A pseudorandom generator produces a stream of binary chips at a rate of N chips per bit. The data stream and the chip stream are modulo-2 added to result in DSSS modulation. The RF bandwidth of the resulting waveform is increased by a factor of N. The spread factor N provides a processing gain G equal to 10 $\log_{10}$N; that is an uncorrelated jammer is forced to generate N times (G db higher) as much power as the victim system transmitter power, assuming equal propagation loss for both the jammer signal and the desired signals.

The FH and TH signals are characterized by the instantaneous signal energy hopping around in frequency-time space. For example, FH signals consist of carrier tones but the instantaneous tone frequency is shifted pseudo-randomly with a dwell time usually equal to the bit period T. The instantaneous bandwidth is roughly equal to the information bandwidth 1/T. The processing gain of the FH system is equal to the number of frequencies among which the system hops. The

TH signals involve turning the transmitter off and on in a pseudo-random manner. The bit interval is divided into N sub-intervals, during one of which the signal is present. The specific segment in which the signal is present changes from symbol to symbol.

1.3 Spread Spectrum Sequences

As mentioned earlier, spread spectrum systems use long sequences in order to achieve spectrum spreading. Some of the important parameters for the selection of these sequences are

a) code length,

b) code autocorrelation properties,

c) code generator complexity.

The sequence length is important in the sense that the longer the sequence, the smaller its repetition frequency is, and thus less interference on the information bits results.

A direct sequence spread spectrum receiver [5] operates in a sequence selective manner, e.g., it correlates the received signal with locally generated sequence which permits it to discriminate and select the desired signal while rejecting the undesired interferring signals. In this mode of operation it is desirable that the sequence employed have a two level autocorrelation in order for the receiver to be able to

discriminate between signals on a Yes-No basis.

In this context, note that autocorrelation refers to the degree of correspondence between a code word and a phase shifted replica of itself. Some of the sequences that meet the above three requirements, and have proven to be the best of the known codes for spread spectrum systems, are discussed on the following pages.

1.4 Binary Random Sequences [6, 7, 9]

In applications, such as spread spectrum systems for communication and ranging, there is a need for binary random sequences. In most cases, it is sufficient to devise a simple method of generating sequences which look random, even if upon closer inspection they can be shown to have certain regularities. In fact, no finite sequence is truly random. The best that can be achieved is to identify certain properties as being associated with randomness, and to accept any sequence which has these properties as a random sequence.

The most familiar example of a random binary sequence arises from tossing an unbiased coin consecutively identifying heads as + 1, and tails as - 1. The following properties are then associated with randomness.

1. The number of heads is approximately equal to the number of tails.

2. Runs of consecutive heads or of consecutive tails often occur, with short runs being more frequent than long runs. More precisely, about one half of the runs have length one, one fourth have length 2, one-eighth have length 3 and so forth.
3. The auto-correlation function, peaks in the middle and decreases rapidly at the ends.

1.4.1 Auto-correlation

If $\{a_n = a_0, a_1, a_2 \ldots\}$ is any sequence of real terms, its autocorrelation function $C(\tau)$ is defined as:

$$C(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} a_n a_{n+\tau} \qquad (1.1)$$

If $\{a_n\}$ is a periodic sequence with period P equation (1.1) reduces to the finite sum.

$$C(\tau) = \frac{1}{P} \sum_{n=1}^{P} a_n a_{n+\tau} \qquad (1.2)$$

The variable τ can be considered as a phase shift of the sequence $\{a_n\}$. $C(\tau)$ measures the amount of similarity between the sequence and its phase shift. The highest value of $C(\tau)$ occurs at $\tau=0$ and if $\{a_n\}$ is random $C(\tau)$ is very small for most other values of τ.

1.5 Pseudo Random Sequences

Any binary sequence that has the properties described in the three following postulates is called a Pseudo Random sequence.

Postulate 1 (R-1): in every period, the number of +1's and -1's differ at most by 1.

Postulate 2 (R-2): in every period, half the runs have length one, one-fourth have length two, one-eighth have length three, etc., as long as the number of runs so indicated exceeds 1.

Postulate 3 (R-3): the auto-correlation function $C(\tau)$ is two valued.

$$C(\tau) = \frac{1}{P} \sum_{n=1}^{P} a_n a_{n+\tau} = \begin{cases} 1 & \text{if } \tau = 0 \\ K/P & \text{if } 0 < \tau < P \end{cases}$$

Pseudo random sequences can be generated by a shift register using the necessary external feedback logic.

1.6 Linear Sequences

The succession of states in a r state shift register is periodic with period $P \leq 2^r - 1$ and for every r there exist feedback arrangements which lead to sequences of length 2^r-1. If the feedback logic consists of modulo-2 additions of the contents of selected stages of the shift register then the generated sequence is called linear. Linear feedback logic incorporated in a r stage shift register can generate sequences

of period 2^r-1 or shorter. In order to determine the necessary feedback logic to generate linear recurring sequences which satisfy some or all of postulates R.1, R.2 and R.3 the following definition is necessary.

Definition:

A polynomial f(x) is said to have exponent P iff it divides X^P-1 but no polynomial of the form X^n-1 for $n < P$.

The associated with a particular periodic sequence of period P feedback logic is completely characterized by a mod-2 polynomial f(x) of degree $r > 1$ with $f(0) = 1$. The polynomial f(x) divides $1-X^P$ and therefore every root of f(x) is also a root of $1-X^P$. Thus every shift register polynomial is simply the equation satisfied by certain P^{th} root of unity.

The periods of irreducible polynomials are always odd because they divide 2^r-1.

In the complex plane the P^{th} roots of unity can be written $Z=e^{2n\pi j/P}$, with n = 1, 2, ..., P if n/P is an irreducible fraction, the corresponding P^{th} root is called primitive. The roots $e^{2n\pi j/P}$ are separated into sets of roots. The polynomials f(x), that are irreducible factors of $1-X^P$, and all their roots are primitive, have the exponent P and are called primitive polynomials.

The number of primitive polynomials is given by $n = \phi\frac{(2^r-1)}{r}$ where $\phi(2^r-1)$ is the Euler-phi function. Evaluation of $\phi(2^r-1)$ is faciliated by the following:

a) If $P = 2^r-1$ is a prime, then $\phi(P) = P - 1$. For example for r=5, P=31 is a prime and therefore, $\phi(31) = 30$. Hence, the number of different primitive polynomials is 30/5 = 6.

b) If P can be factored into powers of distinct primes, that is $P = P_1^{e_1}P_2^{e_2} \cdots P_n^{e_n}$ where P_i are primes raised to powers e_i then

$$\phi(P) = P \times \prod_{i=1}^{n} \frac{P_i - 1}{P_i}$$

For example, for r=12, $2^r-1 = 4095 = 3^2 \times 5^1 \times 7^1 \times 13^1$ then

$$\phi(4095) = 4095 \times \frac{2 \times 4 \times 6 \times 12}{3 \times 5 \times 7 \times 13} = 1728$$

Hence the number of different primitive polynomials of degree 12 is 1728/12 = 144.

Peterson [8] has listed all irreducible polynominals of degree 34 or less. These polynomials can be used as the feedback logic to produce linear recurring sequences. The exact periods of sequences generated by a r-stage linear shift register are determined by the nature of their associated polynominal. This relationship is derived by the following rules.

a) If f(x) is reducible (factorable), then the sequences generated by f(x) may or may not have identical periods.

b) If f(x) is irreducible, then all the sequences generated by f(x) have identical periods (not necessarily maximum). For example, $x^3 + x + 1$ and $x^4 + x + 1$ are irreducible.

c) If f(x) is primitive, then all sequences generated by f(x) have maximum period.

1.7 Maximal Length Sequences

Maximal length sequences of period $P = 2^r - 1$ are generated by shift registers. The feedback logic is prescribed by a primitive polynomial of degree r which is the simplest to generate.

Maximal length sequences observe postulates R-1, R-2, and R-3. Figure 1.1 gives an example of generating a maximal length sequence with a 5-stage shift register.

1.8 Legendre Sequences [8, 9]

An integer i is a quadratic residue of a prime P provided that there exists an integer x such that

$$x^2 \equiv i \bmod P$$

For example, 1 is a quadratic residue of all primes. If r is a quadratic residue of P, then

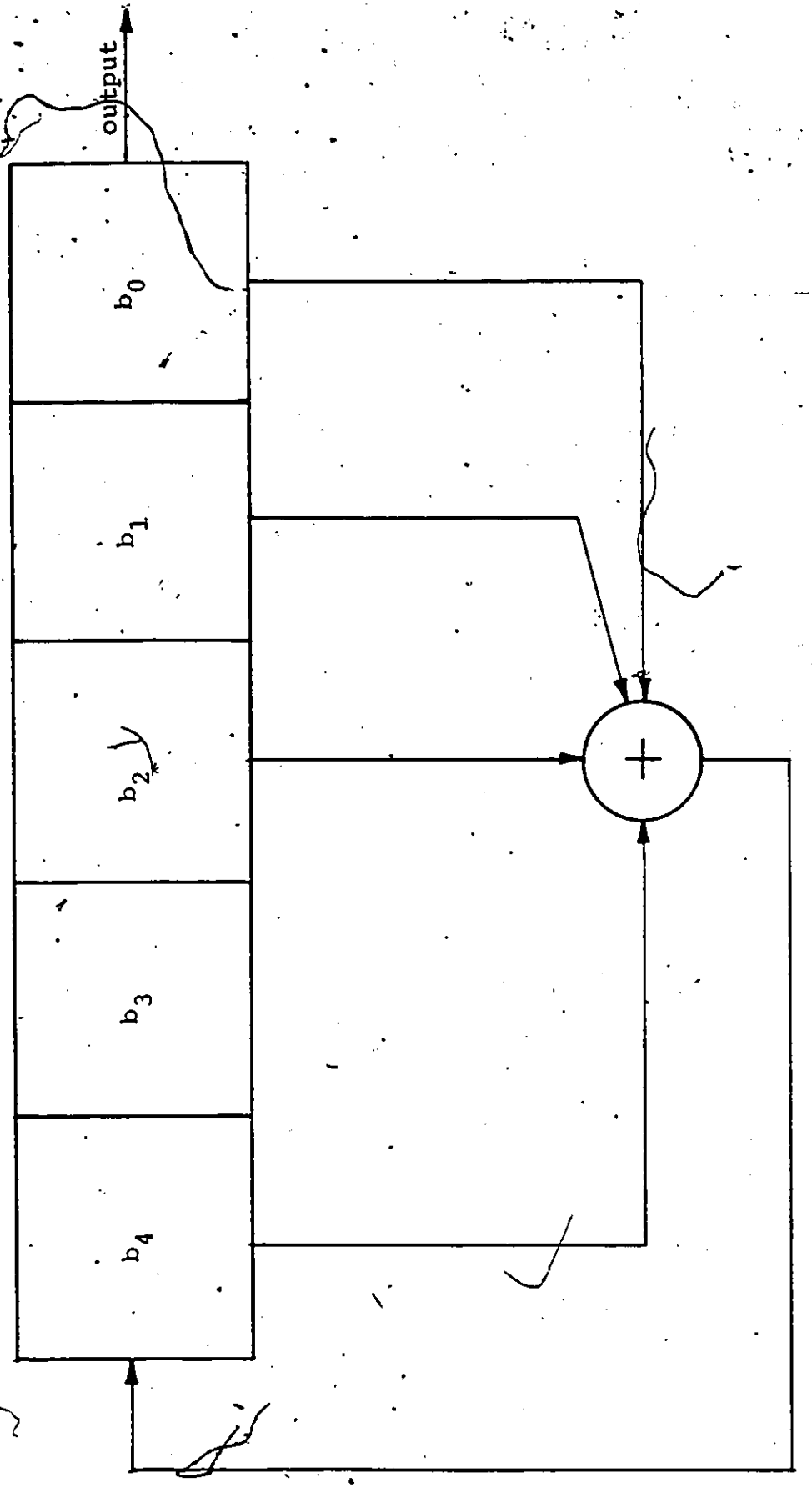


Figure 1.1

Five-stage Maximal Length Sequence Generator. $f(x) = 1+x+x^2+x^4+x^5$

$$r^{(P-1)/2} \equiv x^{P-1} \equiv 1 \bmod P$$

and if

$$r^{(P-1)/2} \not\equiv 1 \bmod P$$

then the number is a nonresidue.

If P is an odd prime the Legendre symbol $(\frac{i}{P})$ is defined by:

$$(\frac{i}{P}) = \begin{cases} 1 \text{ if } i \text{ is a quadratic residue mod } P \\ 0 \text{ otherwise.} \end{cases}$$

Thus, a two-level sequence can be defined by:

$$a_i = (\frac{i}{P})$$

This is the reason these sequences are referred to as Legendre sequences.

The sequences obtained by this method are nonmaximal except for P=3 and P=7 in which case the obtained Legendre sequences are identical to a maximal sequence of the same period P.

If P is a prime of the form 4n-1 then the corresponding sequence has the property R-1 and R-3. For example, if P = 4n-1 = 31, the quadratic residues are:

0, 1, 2, 4, 8, 16, 9, 18, 5, 10, 20, 19, 7, 14, 28, 25.

and the corresponding Legendre sequence is

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

In some cases, it is possible to mechanize the Legendre sequences as modulo-2 sum of m sequences. For example,

	1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0
	1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 1 0 1 1 1 1
	1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 1 1 0 1 1
+	1 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 0

Obviously, generating a Legendre sequence is a more complex process and requires more hardware than a m sequence with no advantage with respect to its properties.

1.9 Nonlinear Shift Register Sequences [7, 9]

It is well known that sequences of length 2^r-1 can always be obtained from an r-stage shift register by means of feedback logic consisting entirely of modulo-2 additions. The number of modulo-2 logics (or linear logics) yielding the maximum length of 2^r-1 is known to be exactly $\phi(2^r-1)/r$. Sometimes a linear logic for maximum length involves only 2 taps.

Removal of the restriction that the feedback logic be linear increases the number of maximum length shift register codes of degree n to $2^{2r-1}/2^r$. This formula discovered by deBruijn suggests an astronomical increase in the number of codes generated by an r-stage shift register.

Sequences of length 2^r-1 can be obtained with as few as 2 taps, whereas length 2^n requires all n taps. As a result, there seems little justification in trying to mechanize length 2^n. Also $C(\tau) \neq 0$ for sequences of length 2^n, although the distance from zero will frequently be small, there will be a drawback due to the fact that their auto-correlation is not a two level one.

The use of nonlinear shift register sequences, due to the associated with the design of the feedback logic difficulties are used only when protection of information for military applications is necessary.

1.10 Jet Propulsion Laboratories (JPL) Composite Codes [2]

The combination of linear maximal sequences produce composite code sequences [9]. Composite codes generated in this way, have special properties that make their use very attractive under certain circumstances. For example, the JPL codes, though constructed from maximal sequences, are not maximal and exhibit special correlation properties that permit rapid synchronization. More specifically, the JPL codes are constructed by modulo-2 addition of two or more maximal linear sequences, whose lengths are relatively prime. The advantages of this technique are as follows:

a) very long codes useful for unambiguous ranging over long ranges are available,

b) long codes are generated by a small number of shift registers,

c) receiver synchornization can be accomplished by separate operations on the component codes. This can greatly reduce the time required for synchronization.

1.11 Code Acquisition and Synchronization [9, 10, 11]

In most communication systems it is necessary for the receiver to be synchronized (in time, frequency, phase or any combination of these) with the received waveform. The receiver must therefore conduct a search to achieve synchronization before any communications can be established.

In spread spectrum systems [2] code synchronization is necessary because the code is the key to despreading the desired information and to spreading any undesired signals. Several techniques have developed for code synchronization for all spread spectrum systems. Attention will be focused on the techniques used mainly in direct sequence spread spectrum systems, since those are the most commonly used.

The initial synchronization or acquisition is the most difficult to achieve. When synchronization has already occurred it is often possible to base a subsequent synchronization on the knowledge of timing gained.

The proposed by Ward method of synchronization is most widely used in D.S.S.S.

In Ward's method, the transmitter and the receiver each contain identical shift registers so that identical sequences are formed.

The system makes its best estimate of the first n received bits, loads the receiver sequence generator with that estimate and starts operation of the sequence generator and the tracking circuits. If the correct estimate is loaded, tracking will occur. At the same time, a correlation is performed between the input signal and the signal from the local sequence, generator. If the correlation indicates the receiver contains the correct sequence, no further action is taken. If the cross-correlation indicates that an incorrect estimate was made, a new estimate of the input is made and the same procedure is followed until a correct acquisition has occurred.

In 1973 [11], C.C. Kilgus proposed to use Majority Logic Decoding to correct the received sequence bits if any errors have occurred, before loading it into the shift register. Depending on the number of estimates on each received bit, a significant improvement in the probability of correct acquisition in the first attempt was achieved as shown in Figure 1.2.

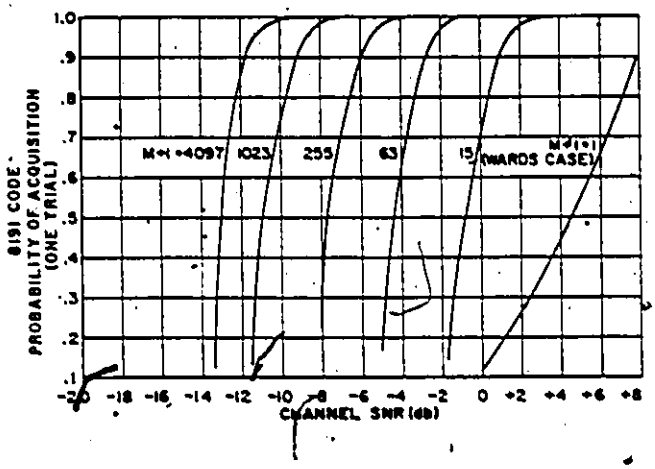


Figure 1.2 - One Trial Acquisition Probability

1.12 Problem Definition

In the context of the above discussions, the aim of this work is to prove that a class of composite codes, proven useful in the spread spectrum transmission technique can be majority logic decoded and as a result, Kilgus method can be used as an acquisition technique.

1.13 Thesis Outline

This thesis deals mainly with the majority logic decoding problem of a class of composite codes and some useful properties of this class.

In Chapter II, majority logic as a decoding technique is reviewed.

Chapter III describes how a class of composite codes, parallel to the JPL codes, are derived certain known and useful properties as these codes are discussed.

Finally, it is proved that this class of codes can be majority logic decoded.

Chapter IV contains the concluding remarks.

CHAPTER II

MAJORITY LOGIC DECODING

Among the various known decoding schemes MLD (majority logic decoding) is one of the simplest to implement.

As was mentioned, MLD can be used to obtain code acquisition in low SNR circumstances, and provides a significant improvement in the performance of a DSSS. Unfortunately, very few codes are known to be efficientIy majority logic decodable [12], [13].

In this chapter, we will define what a linear block code is, how it is generated, and mention some of its characteristics. Subsequently, the basic concepts of MLD will be introduced with reference to some known classes of majority logic decodable codes.

2.1 Linear Block Codes [14]

Let us assume that information originating from a binary information source is applied at the input of the encoder. The function of the encoder is a two-step one. It segments the information into message blocks, each block consisting of k successive information bits and transforms the message blocks into longer blocks of n (n > k) binary digits. These binary n-tuples are called code words, and are formed according to specific rules predetermined by

the encoding procedure. Since each message block consists of k binary digits, there are 2^k possible distinct message blocks, and therefore 2^k possible code words at the output of the encoder. The set of 2^k code words is called a block code. A code word is frequently called a code vector because it is an n-tuple from the vector space V_n of all n-tuples

A linear (n, k) block code whose generator polynomial is a factor of x^n+1 and has exponent n, i.e.,

$$x^n+1 = g(x)h(x) \qquad (2.1)$$

is called a cyclic code, and has the following property:

If an n-tuple (v) is a code vector of C, then the n-tuple obtained by shifting v cyclically one place to the right is also a code vector of C.

The error correcting capability of a code is determined by its minimum distance d_{min}. In order to define what minimum distance is, the following two definitions are necessary.

Definition 2.1: The Hamming weight of n-tuple v w(v) is defined as the number of non zero components of v, i.e., if v = 1 0 1 0 1 1 0 0 1, w(v) = 5.

Let u and v be two n-tuples.

Definition 2.2: The Hamming distance between u and v, d(u, v) is defined as the number of components in which they differ; i.e. if

u = 1 0 0 1 0 1 1 0 0 0 1

v = 1 1 0 0 1 0 1 0 1 0 1

then d(u, v) = 5.

Since the Hamming distance is the only distance of interest in the context of this thesis, the words distance and Hamming distance have identical meaning.

Given a linear code, we can calculate the distances between all possible pairs of code words; the smallest distance is called the minimum distance of the code (d_{min}).

2.2 Majority Logic Decoding [12, 13, 17]

In general, a code word V(x) transmitted through a communications channel is corrupted by noise. As a result of this process, the received code word R(x) differs from the transmitted code word V(x). A word V(x) belonging to V and a received sequence R(x) are related by:

$$R(x) = V(x) + E(x) \tag{2.2}$$

where E(x) is the error polynominal.

In order to decode correctly, the received code word R(x), E(x) has to be evaluated. In order to evaluate E(x) from R(x), the decoder must find the error syndrome.

The error syndrome can be expressed as a (modulo-2) sum of certain error digits (e.g., received bits). If we consider a modulo-2 sum of syndrome bits of the following

form:

$$A = a_0 s_0 + a_1 s_1 + \ldots + a_{n-k-1} s_{n-k-1} \qquad (2.3)$$

where a_i is either 0 or 1.

Since the syndrome bits are sums of error digits, it follows that A of equation 2.3 is also a sum of error bits.

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

where b_i is either 1 or 0. This sum A of error digits is called a parity check sum. The error digit e_1 is said to be checked by A if the coefficient b_1 of e_1 in A is 1.

If e_1 is checked by a set of parity check sums $A_1, A_2, \ldots A_J$, and no other error digit is checked by more than one check sum, then one may say that this set of parity check sums is orthogonal on the error digit e_1.

If there are d-1 parity check sums (equations) orthogonal on a single e_i, then the code is 1-step majority logic decodable, e.g., the error digit can be evaluated in one step and therefore only one majority element is used by the decoder.

Example 2.1

Consider the (15, 7) cyclic code with generator polynominal: $g(x) = 1 + x^4 + x^6 + x^7 + x^8$ and distance d=5.

The d-1 = 4 parity check sums orthogonal on e_{14} are as follows:

$$
\begin{aligned}
A_1 = s_3 &= e_3 + e_{11} + e_{12} + e_{14} \\
A_2 = s_1 + s_5 &= e_1 + e_5 + e_{13} + e_{14} \\
A_3 = s_0 + s_2 + s_6 &= e_0 + e_2 + e_6 + e_{14} \\
A_4 = s_7 &= e_7 + e_8 + e_{10} + e_{14}
\end{aligned} \tag{2.13}
$$

Therefore the (15, 7) code is completely orthogonalizable in one step.

Since the code is cyclic, a check of all the noise digits is made by cyclically shifting the received code word. Errors of weight $t = \frac{d-1}{2} = 2$ are correctable, as well as some errors of weight $t > 2$ if no more than two equations are affected by these errors. In Figure 2.1, the MLD circuit for decoding the (15, 7) cyclic code is shown.

The steps in the operation of the decoder of Figure 2.1 follow:

Step 1 - With Gate 1 on and Gate 2 off, the received vector is read into the buffer register.

Step 2 - The 4 parity-check sums orthogonal on e_{14} are formed by summing appropriate received digits as determined by the check equations. This is done by 4 modulo-2 adders.

Step 3 - The 4 orthogonal check sums are fed into a majority logic gate. The first received digit is read out of the buffer and corrected by the output of the majority gate. The correction is accomplished by modulo-2 adding the output of the majority gate to the received digit.

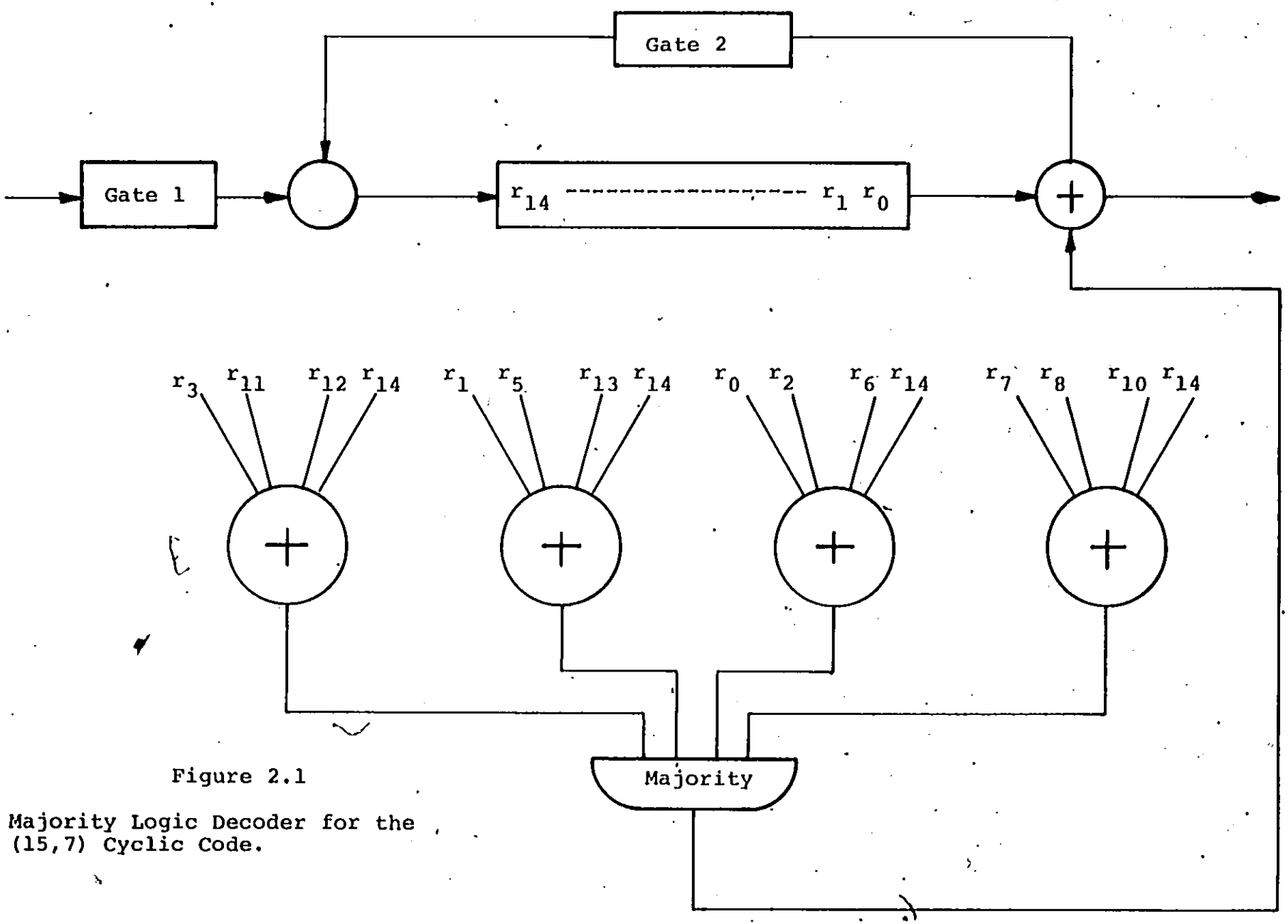


Figure 2.1

Majority Logic Decoder for the (15,7) Cyclic Code.

Step 4 - Following Step 3, the buffer register has been shifted one place to the right. Now the second received digit is in the first stage of the buffer register, and will be corrected in exactly the same manner as the first digit. The decoder repeats Steps 2 and 3.

Step 5 - The received vector is decoded digit by digit in the above manner, until a total of 15 digits have been decoded. Correct decoding is guaranteed, provided that 2 or fewer errors occurred in the received vector.

In several cases we can form d-1 equations orthogonal on selected sums of information noise bits. In this case, majority logic decoding can be used to estimate these sums, and subsequently to assume that they are known and treat them as additional parity checks. Additional parity checks are then combined with the original parity checks to produce a new set of parity checks until d-1 parity checks orthogonal on one noise digit, can be formed.

If this process can be carried out for all e_j $j = 1, 2, 3 \ldots k$ then one assumes that the code can be L-step orthogonalized. Such a code will require L-levels of majority logic gating and will be capable of correcting all errors of weight less than or equal to t. An example of how this decoding algorithm can be applied to a 2-step majority logic decodable code is given next.

Example 2.2

Consider the (7, 4) Hamming [28] code with d=3 which is generated by $g(x) = 1 + x + x^3$.

The two sets of orthogonal check equations on $\{e_5, e_6\}$ and $\{e_4, e_6\}$ are as follows:

$$A_1 \quad \begin{aligned} s_1' &= e_0 + e_3 + e_5 + e_6 = s_0 \\ s_2' &= e_2 + e_4 + e_5 + e_6 = s_2 \end{aligned}$$

and (2.16)

$$A_2 \quad \begin{aligned} s_1'' &= e_0 + e_1 + e_4 + e_6 = s_0 + s_1 \\ s_2'' &= e_2 + e_4 + e_5 + e_6 = s_2 \end{aligned}$$

A_1 and A_2 are orthogonal on e_6; therefore, e_6 can be estimated from A_1 and A_2. if a single error has occurred in the received vector, then the estimations of A_1 and A_2 are correct as is the estimation of e_6.

The circuit for the decoder is given in Figure 2.2. The circuit of Figure 2.2 performs the following operating steps.

Step 1 - With Gate 1 opened and Gate 2 closed, the received vector is read into the buffer register.

Step 2 - Parity-check sums orthogonal on the sets of error digits $e_5 + e_6$, and $e_4 + e_6$ are formed. This set of check sums is fed into the first level majority gates. The outputs of the 2-first level of majority logic gates are

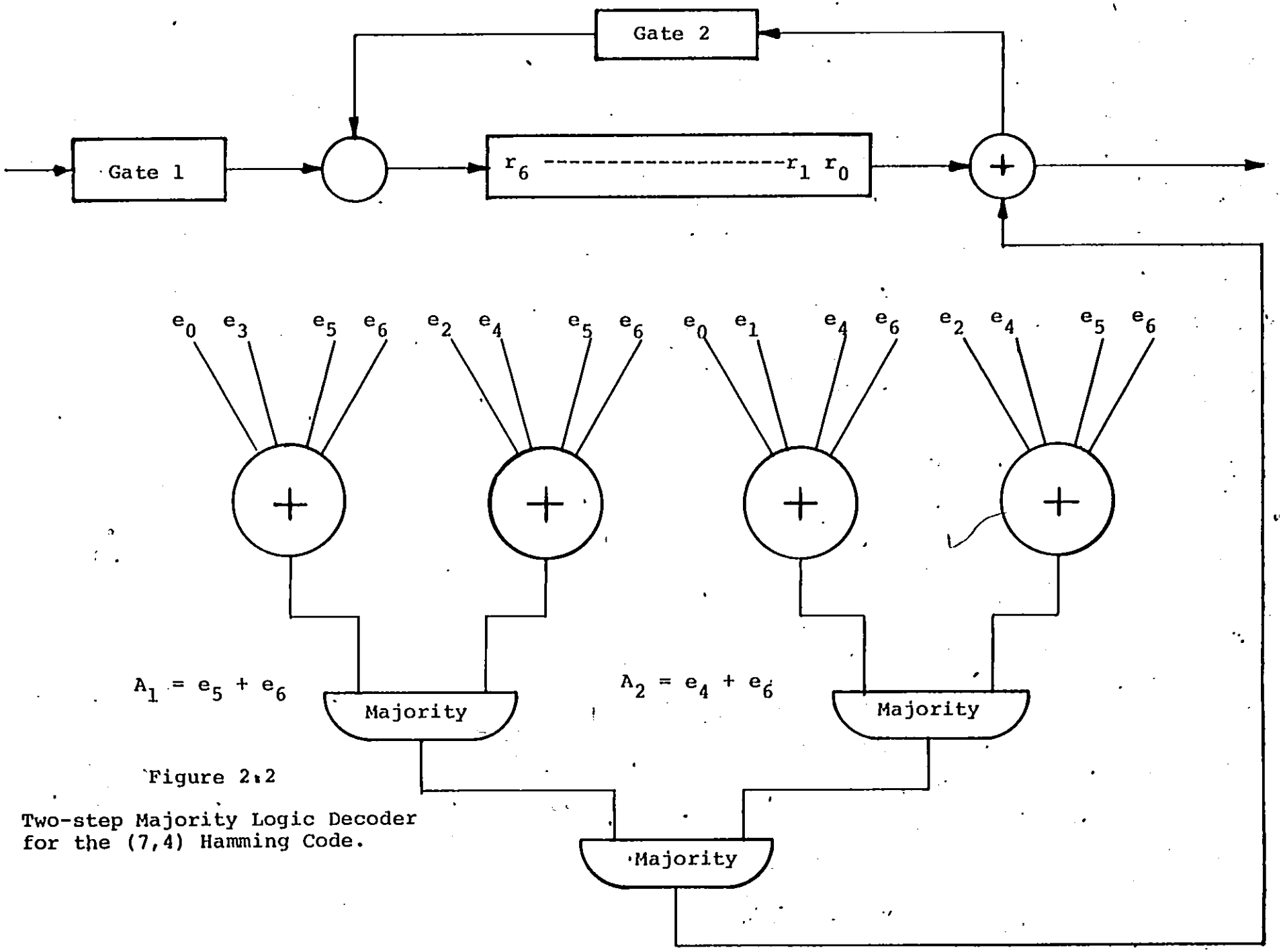


Figure 2.2

Two-step Majority Logic Decoder for the (7,4) Hamming Code.

then fed into the second level majority logic gate. The 2 inputs to this gate are parity check sums orthogonal to the received digit r_6.

Step 3: The first received digit is read out of the buffer and is corrected by the output of the last level majority gate. Correction is done by a modulo-2 addition.

2.4 Majority Logic Decodable Codes

A code with minimum distance d is considered majority logic decodable if d-1 orthogonal check sums on one error digit can be formed. If the orthogonal estimates are obtained in one step, the code is one step ML decodable. The code is L-step majority logic decodable if obtained in L-steps. Following some classes of codes that are known to be majority logic decodable are presented.

2.4.1 Maximum Length Codes

These codes derive their name from the fact that the code word can be considered as the first 2^m-1 output symbols of a maximal length m-stage shift register having the m information symbols as initial conditions. The parameters of maximum length codes.

Block length	$n = 2^m-1$
Number of information digits	$k = m$
Minimum distance	$d = 2^{m-1}$

The generator polynomial of these codes is $g(x) = x^n+1/P(x)$ where the parity polynominal $P(x)$ is any primitive polynomial of degree m.

The maximum length codes are known to be 1-step majority logic decodable and the parity check equations obey the following rules:

If k is odd, the number J of parity check equation orthogonal on any one digit is a multiple of 3. If k is even, (J-1) is a multiple of 3 on the basis of the above, all the parity checks of a maximal length code can be written in the form

$$a_i = a_{i+x} + a_{i+y} \qquad \text{modulo-2} \qquad (2.19)$$

where a = codeword bit

$i = 0, 1, 2, \ldots, (n-1)$ for $n = 2^k-1$.

x, y nonzero integers $x \neq y$ and subscript additions are performed modulo-n.

2.4.2 Difference Set Codes (DS)

Given a set $P = \{0, \ell_1, \ell_2, \ldots \ell_{2s}\}$ which is a perfect difference set it is possible to obtain a generator polynomial g(x) which will generate a DS code with the following parameters.

$$\begin{aligned} n &= 2^{2s} + 2^s + 1 \\ n - k &= 3^s + 1 \\ d &= 2^s + 2 \end{aligned} \qquad (2.20)$$

The codes with the parameters described by 2.20 are known to be 1-step MLD.

2.4.3 Classes of L-step Majority Logic Decodable Codes

One of several classes of codes which are L-step MLD is the Hamming codes. These are cyclic codes with minimum distance d=3 having $n=2^m-1$ and $k=2^m-m-1$. The Hamming codes can be completely orthogonalized in L-steps with $L \leq m-1$.

The first order Reed Muller codes (RM) are of the form ($n=2^m$, $k=m+1$) and can be orthogonalized in 2 steps.

An extension of RM codes are the Finite Projective Geometry codes, and the Euclidean Geometry codes which can be completely orthogonalized. As their structure is complex, they lay outside the scope of this work [26].

CHAPTER III

COMPOSITE CODES

3.1 Introduction

This chapter shall demonstrate how a class of composite codes is constructed. Certain useful properties of this class of codes will be reviewed and finally, proof provided that this class of codes is Majority Logic Decodable under certain conditions.

3.2 Construction of a Class of Composite Codes and Some of their Properties

Let v_i be an (n_i, k_i) binary linear code, [22] $i = 1, 2, 3, \ldots \mu$ where $(n_i, n_j) = 1$. Then $v^{(\mu)}$ is the composite code of length $n = n_1 n_2 n_3 \ldots n_\mu$, such that every word $v^{(\mu)}(x)$ of $v^{(\mu)}$ is given by [24]:

$$v^{(\mu)}(x) = \sum_{i=1}^{\mu} v_i(x) \frac{1+x^n}{1+x^{n_i}} \tag{3.1}$$

Theorem 3.1

With $v_1, v_2, \ldots v_\mu$ and $v^{(\mu)}$ as previously defined if $v_i \varepsilon v_1$ has weight w_i $i = 1, 2, 3, \ldots \mu$, then the corresponding word $v^{(\mu)}$ of $v^{(\mu)}$ has weight

$$W^{(\mu)} = W^{(\mu-1)}(n_\mu - w_\mu) + W\left(\frac{n}{n_\mu} - W^{(\mu-1)}\right) \tag{3.5}$$

where $W^{(\mu)}$ the weight of the μ-component code and $W^{(\mu-1)}$ the weight of the μ-1 component composite code.

In general, the error correcting capability of a code depends on its minimum distance, as a result of that the minimum distance is an important parameter.

Finding an analytical expression for the minimum distance of an μ-component composite code is next to impossible. For this work it suffices to find an analytical expression for the minimum distance of a 2-component composite code, and to use this expression recursively to find the minimum distance for the cases of interest in the context of this thesis.

Theorem 3.2 [21]

Let d_i and D_i be the minimum and maximum weights in V_i for i = 1, 2 then:

$$d = \text{Min}\{n_2 d_1,\ n_1 d_2,\ n_2 D_1 + n_1 D_2 - 2D_1 D_2\}$$

and

$$D = \text{Max}\{n_2 D_1,\ n_1 d_2,\ n_2 D_1 + n_1 D_2 - 2D_1 D_2\}$$

Corollary 3.2.1

If V_2 is an m-sequence code then:

$$d = \text{Min}\{n_2 d_1,\ \frac{n_1(n_2+1)}{2} - D_1\}$$

and

$$D = \text{Max}\{n_2 D_1,\ \frac{n_1(n_2+1)}{2}\}$$

A very important class of codes is the class of cyclic codes especially when it comes to majority logic decoding. As stated earlier, one can majority logic decode all the noise digits of such a code by simply shifting a code word in a shift register. The following theorem gives us the means to finding the generator polynomial of a composite code if all the component codes are cyclic.

Theorem 3.3 [21]

Let V_i be a cyclic code with generator polynomial $g_i(x)$, $i = 1, 2, 3, \ldots \mu$ then $v^{(\mu)}$ is cyclic with generator polynomial.

$$g(x) = \gcd\{g_1(x)\ \frac{1+x^n}{1+x^{n_1}},\ g_2(x)\ \frac{1+x^n}{1+x^{n_2}} + \ldots$$

$$+ g_\mu(x)\ \frac{1+x^n}{1+x^{n_\mu}}$$

Figure 3.1 gives the circuit that implements the (21, 5) composite code composed of the (3,2) m sequence code and the (7,3) m sequence code.

Its generator polynomial is $g(x) = 1+x+x^2+x^3+x^4+x^6+x^8+x^{11}+x^{12}+x^{16}$, and its parity check polynomial is $h(x) = 1+x+x^5$.

3.3 Majority Logic Decoding of Composite Codes

As stated earlier, MLD is a powerful decoding technique, easily implemented and useful in the context of attaining

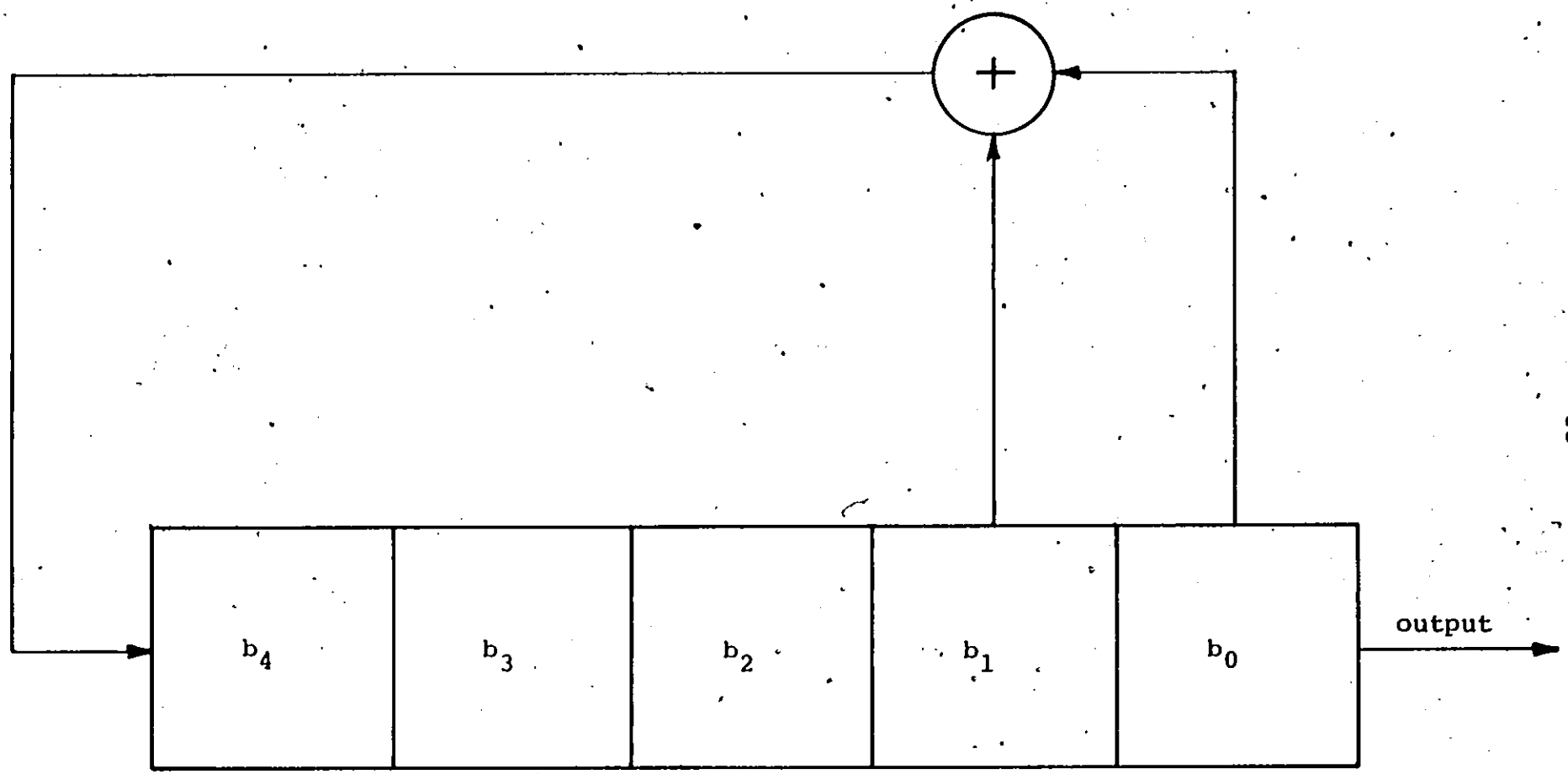


Figure 3.1

Five-stage Encoding Circuit for the (21,5) Cyclic Composite Code.

code acquisition in DSSS.

The following paragraphs will prove that under certain conditions the class of composite codes discussed in Section 3.2 can be majority logic decoded.

Theorem 3.4 [31]

Let V_1 be an (n_1, k_1) binary cyclic code which has minimum distance d_1 and is L-step majority logic decodable, and let V_2 be an $(n_2 = 2^{m_2}-1, k_2 = m_2)$ m-sequence code where $(n_1, n_2) = 1$. The composite code $V^{(2)}$ can correct by L+1 step majority logic decoding $\frac{n_2 d_1 - 1}{2}$, or fewer errors if

$$d_1 \leq \frac{n_1(n_2-1)}{2n_2}, \text{ and } \frac{n_1(n_2-1) - 2}{2}$$

or fewer errors if

$$d_1 \geq \frac{n_1(n_2-1)}{2n_2}$$

Proof: A word of $V^{(2)}$ $V^{(2)}(x)$ can be written as follows

$$V^{(2)}(x) = A(x) \frac{1+x^n}{1+x^{n_1}} + B(x) \frac{1+x^n}{1+x^{n_2}} \qquad (3.12)$$

where

$$A(x) = a_0 + a_1(x) + a_2x^2 + \ldots + a_{n-1}X^{n_1-1} \varepsilon V_1 \qquad (3.13)$$

$$B(x) = b_0 + b_1x + b_2X^2 + \ldots + bn_2-1\, X^{n_2-1} \varepsilon V_2 \qquad (3.14)$$

$$V^{(2)}(x) = v_0 + v_1x + v_2x^2 + v_3x^3 + \ldots + v_{n-1}X^{n-1} \varepsilon V \qquad (3.15)$$

Since V_2 is an m-sequence code, there are $\frac{n_2-1}{2}$ orthogonal check sums, on b_0, of the form

$$b_0 = b_a + b_b \tag{3.16}$$

Since $(n_1, n_2) = 1$, with respect to (3.12), (3.13), (3.14) there are $n = n_1 n_2$ equations of the form where

$$v_i = a_\gamma + b_\delta$$

where for every pair of γ and δ, there is a different i. This means that, by adding two appropriate equalities of (3.17), we can express b_0 of (3.16) as the sum of two v_j's. But then each a_j occurs n_1 times in (3.17) and there are n_2-1/2 orthogonal check sums (3.16). Therefore, $\frac{n_1(n_2-1)}{2}$ check sums on b_0, of the form

$$b_0 = v_p + v_q \tag{3.18}$$

can be formed. Thus, every b_j can be determined by taking a majority note on $\frac{n_1(n_2-1)}{2}$ estimates, since the codes are all cyclic.

Using these estimates in (3.13)

$$v_i + b_\delta = v_i' = a_\delta$$

where v_i' is known. Since V_1 is an L-step majority logic decodable code, L sets of $n_2 + n_2(d_1-1) = n_2 d_1$ orthogonal

check equations are formed on selected sums of a_i of the form

$$v'_q + v'_p = a_a + a_b + a_c + \ldots + a_\ell \qquad (3.19)$$

Therefore every a_i can be estimated in L-steps in the presence of $\frac{n_2 d_1 - 1}{2}$ or fewer errors.

After estimating all a_i's and b_j's every V_i can be reconstructed by taking appropriate linear combinations of a's and b's.

Tables 1 and 2 give listings of L+1 step majority logic decodable composite codes composed of one L-step majority logic decodable code and one m-sequence code.

Example

As an example, consider the (21, 6) d=7 composite code composed of V_1, where V_1 is the (7, 4) 2-step MLD Hamming code and V_2 the (3, 2) 1-step majority logic decodable m sequence code.

The received noise digits of the composite code as linear combinations of a's and b's are as follows:

$$\begin{array}{lll}
v_0 = a_0 + b_0 & v_7 = a_0 + b_1 & v_{14} = a_0 + b_2 \\
v_1 = a_1 + b_1 & v_8 = a_1 + b_2 & v_{15} = a_1 + b_0 \\
v_2 = a_2 + b_2 & v_9 = a_2 + b_0 & v_{16} = a_2 + b_1 \\
v_3 = a_3 + b_0 & v_{10} = a_3 + b_1 & v_{17} = a_3 + b_2 \\
v_4 = a_4 + b_1 & v_{11} = a_4 + b_2 & v_{18} = a_4 + b_0 \\
v_5 = a_5 + b_2 & v_{12} = a_5 + b_0 & v_{19} = a_5 + b_1 \\
v_6 = a_6 + b_0 & v_{13} = a_6 + b_1 & v_{20} = a_6 + b_2
\end{array}$$

Table 1

3-STEP MAJORITY LOGIC DECODABLE CODES COMPOSED OF ONE 2-STEP MAJORITY LOGIC DECODABLE CODE AND ONE m-SEQUENCE CODE

v_1			v_2			$v^{(2)}$		
n	k	d	n	k	d	n	k	d
31	6	15	3	2	2	93	8	31
			7	3	4	217	9	93
			15	4	8	465	10	217
			63	6	32	1953	12	945
			127	7	64	3937	13	1905
			255	8	128	7905	14	3825
			511	9	256	15841	15	7665
			2047	11	1024	63457	17	30705
			4095	12	2048	126945	18	61425
			8191	13	4096	253921	19	122865
63	48	5	31	5	16	1953	53	155
			127	7	64	8001	55	635
			2047	11	1024	128961	59	10235
			8191	13	4096	516033	61	40955
15	5	7	7	3	4	105	8	45
			31	5	16	465	10	217
			127	7	64	1905	12	889
			511	9	256	7665	14	3577
			2047	11	1024	30705	16	14329
			8191	13	4096	122865	18	57337
15	10	4	7	3	4	105	13	28
			31	5	16	465	15	124
			127	7	64	1905	17	508
			511	9	256	7665	19	2044
			2047	11	1024	30705	21	8188
			8191	13	4096	122865	23	32764
63	7	31	127	7	64	8001	14	3937
			2047	11	1024	128961	18	63457
			8191	13	4096	516033	20	253921

Table 2

4-STEP MAJORITY LOGIC DECODABLE CODES COMPOSED OF ONE 3-STEP MAJORITY LOGIC DECODABLE CODE AND ONE m-SEQUENCE CODE

v_1			v_2			$v^{(2)}$		
n	k	d	n	k	d	n	k	d
31	16	7	3	2	2	93	18	21
			7	3	4	217	19	49
			15	4	8	465	20	105
			63	6	32	1953	22	441
			127	7	64	3937	23	889
			255	8	128	7905	24	1785
			511	9	256	15841	25	3577
			2047	11	1024	63457	27	14329
			4095	12	2048	126943	28	28665
			8191	13	4096	253921	29	57337
31	25	4	3	2	2	93	27	12
			7	3	4	217	28	28
			15	4	8	465	29	60
31	25	4	63	6	32	1953	31	252
			127	7	64	3937	32	508
			255	8	128	7905	33	1020
			511	9	256	15841	34	2044
			2047	11	1024	63457	36	8188
			4095	12	2048	126945	37	16380
			8191	13	4096	253921	38	32764
63	42	7	127	7	64	8001	49	889
			2047	11	1024	128961	53	14329
			8191	13	4096	516033	55	57337
63	22	15	127	7	64	8001	29	1905
			2047	11	1024	128961	33	30705
			8191	13	4096	316033	35	122865

The seven orthogonal check sums on b_0 are

$$\begin{array}{llll} v_1 + v_8 = b_0 & v_7 + v_{14} = b_0 & v_{13} + v_{20} = b_0 & v_{19} = v_5 = b_0 \\ v_4 + v_{11} = b_0 & v_{10} + v_{17} = b_0 & v_{16} + v_2 = b_0 & \end{array}$$

The two sets of 9 orthogonal equations on $\{a_5, a_6\}$ and $\{a_4, a_6\}$ are:

$$\begin{array}{lll} v'_5 + v'_6 = v'_7 + v'_{10} = v'_2 + v'_4 = a_5 + a_6 \\ v'_0 + v'_3 = v'_{19} + v'_{20} = v'_9 + v'_{11} = a_5 + a_6 \\ v'_{12} + v'_{13} = v'_{14} + v'_{17} = v'_{16} + v'_{18} = a_5 + a_6 \end{array}$$

$$\begin{array}{lll} v'_0 + v'_1 = v'_4 + v'_6 = v'_2 + v'_5 = a_4 + a_6 \\ v'_7 + v'_{18} = v'_{11} + v'_{13} = v'_9 + v'_{12} = a_4 + a_6 \\ v'_{14} + v'_{15} = v'_{18} + v'_{18} = v'_{16} + v'_{19} = a_4 + a_6 \end{array}$$

Theorem 3.5

Let V_1 be an (n_1, k_1) binary cyclic code which has minimum distance d_1 and is 1-step majority logic decodable and let v_2 be an $(n_2 = 2^{m_2}-1, k_2 = m_2)$ M-sequence code where $(n_1, n_2) = 1$. Then the composite code $V^{(2)}$ can correct, by 2-step majority logic, $\frac{n_2 d_1 - 1}{2}$ or fewer errors if

$$d_1 \leq \frac{n_1(n_2-1)}{2n_2} \text{ and } \frac{n_1(n_2-1) - 2}{4} \text{ or fewer errors if}$$

$$d_1 \geq \frac{n_1(n_2-1)}{2n_2} .$$

Proof: Let $b_0b_1b_2 \cdots b_{n1-1}$ be a word of V_1 and $a_0a_1a_2 \cdots a_{n2-1}$ a word of V_2. Let $v_0v_1v_2 \cdots v_{n-1}$ be the resulting word of $V^{(2)}$. Since $(n_1, n_2) = 1$, every v_p can be expressed in the form

$$v_p = b_q + a_r \tag{3.20}$$

Where there is only one p for any given pair of q, r. This means, since r occurs in (3.20) n_1 times and V_2 is an m-sequence code, that we can form $n_1 \frac{(n_2-1)}{2}$ orthogonal check sums on a_0 of the form

$$a_0 = v_s + v_t \tag{3.21}$$

implying, since V_2 is cyclic, that every a_r can be estimated by majority logic if the number of errors is $\frac{n_1n_2-n_1-2}{4}$ or smaller.

After estimating each a_r we can rewrite (3.20) in the form

$$b_q = v_p + a_r = v'_p \tag{3.22}$$

where v'_p is known. Since V_1 is 1-step majority logic decodable, from (3.22) we can form $n_2+n_2(d_1-1) = n_2d_1$ orthogonal check sums on b_0, of the form

$$b_0 = v'_x + v'_y \tag{3.23}$$

Therefore if $\frac{n_1(n-1)}{2} \geq n_2 d_1$ or if $d_1 \leq \frac{n_1(n_2-1)}{2n_2}$, $n_2 d_1$ orthogonal equations can be termed and as a result we can correct $\frac{n_2 d_1 - 1}{2}$ or fewer errors with two-step majority logic decoding. Similarly, if $d_1 \geq \frac{n_1(n_2-1)}{2n_2}$ $\frac{n_1(n_2-1)}{2}$ orthogonal equations is formed and as a result, can correct $\frac{n_1 n_2 - n_1 - 2}{4}$ or fewer errors by two-step majority logic decoding, this concludes the proof of theorem 3.5.

As special case to the theorem we have the following 2 corollaries.

Corollary 3.5.1

If, with reference to theorem 3.6, V_1 has the word of all one's so that $D_1 = n_1$ and if $d_1 \leq \frac{n_1(n_2-1)}{2n_2}$, then $V^{(2)}$ is 2-step majority logic decodable.

Proof: Having established by means of corollary 3.2.1 that for $V^{(2)}$ its minimum distance is

$$d = \min\{n_2 d_1, \frac{n_1(n_2+1)}{2} - D_1\}$$

since $D_1 = n_1$ we have

$$d = \min\{n_2 d_1, \frac{n_1 n_2 n_1}{2}\}$$

$$d = \min\{n_2 d_1, \frac{n_1(n_2-1)}{2}\}$$

and since $n_2 d_1 \leq \frac{n_1(n_2-1)}{2}$, $d = n_2 d_1$. Therefore the number of

orthogonal estimates coincides with the minimum distance of the code.

Corollary 3.5.2

If, with reference to theorem 3.6, V_1 is even weighted with $D_1 = n_1 - d_1 + 1$, and if $d_1 \leq \frac{n_1(n_2-1)-2}{2(n_2-1)}$, then $V^{(2)}$ is two-step majority logic decodable.

Proof: For $V^{(2)}$ we can write:

$$d = \min\{n_2 d_1, \frac{n_1(n_2+1)}{2} - n_1 + d_1 - 1\}$$

or $$d = \min\{n_2 d_1, \frac{n_1(n_2-1) + 2d_1 - 2}{2}\} \text{ since}$$

$$(n_2-1)d_1 \leq \frac{n_1(n_2-1) - 2}{2(n_2-1)}$$

$$d = n_2 d_1$$

and as a result of Theorem 3.5, the minimum distance of the code coincides with the number of the orthogonal parity check equations. Table 3 provides a list of 2-step majority logic decodable codes composed of one 1-step majority logic decodable and one m-sequence code. Subsequently, consider $V^{(2)}$ where $V^{(2)}$ is the (105, 10) d=35 composite code such that each code word in V is of the form

$$V(x) = B(x) \frac{X^{105} + 1}{X^{15} + 1} + A(x) \frac{X^{105} + 1}{X^{7} + 1}$$

Table 3

2-STEP MAJORITY LOGIC DECODABLE CODES COMPOSED OF ONE 1-STEP MAJORITY LOGIC DECODABLE CODE AND ONE m-SEQUENCE CODE

v_1			v_2			$v^{(2)}$		
n	k	d	n	k	d	n	k	d
15	7	5	7	3	4	105	10	35
			31	5	16	465	12	155
			127	7	64	1905	14	635
			511	9	256	7665	16	2555
			2047	11	1024	30705	18	10235
			8191	13	4096	122865	20	40955
21	11	6	31	5	16	651	16	186
			127	7	64	2667	18	762
			2047	11	1024	42987	22	12282
			8191	13	4096	172011	24	49146
63	37	9	127	7	64	8001	44	1143
			2047	11	1024	128961	48	18423
			8191	13	4096	516033	50	73719
63	37	9	31	5	16	1953	48	279
63	13	21	31	5	16	1953	18	651
			127	7	64	8001	20	2667
			2047	11	1024	128961	24	42987
			8191	13	4096	516033	26	172011
63	1	63	31	5	16	1953	6	945
			127	7	64	8001	8	3969
			2047	11	1024	128961	12	64449
			8191	13	4096	516033	14	257985
15	1	15	7	3	4	105	4	45
			31	5	16	465	6	225
			127	7	64	1905	8	945
			511	9	256	7665	10	3825
			2047	11	1024	30705	12	15345
			8191	13	4096	122865	14	61425
31	1	31	3	2	2	93	3	31
			7	3	4	217	4	93
			15	4	8	465	5	217
			63	6	32	945	7	961
			127	7	64	3937	8	1953
			255	8	128	7905	9	3937
			511	9	256	15841	10	5355
31	1	31	2047	11	1024	63457	12	31713
			4095	12	2048	126945	13	63457
			8191	13	4096	253921	14	126945
255	21	85	7	3	4	1785	24	595
			31	5	16	7905	26	2635
			127	7	64	32385	28	10795
			511	9	256	130305	30	43435

and B(x) is a code word in B where B is the (15, 7) d=5 cyclic code and A(x) is a code word in A where A is the (7, 3) d=4 m-sequence code. The 45 orthogonal parity check sums on a_0 are as shown in 3.24 and the 35 orthogonal check sums on b_0 are shown in 3.25. Since every v_i in V is a linear combination of a_i and b_j every v_i can be determined correctly in the presence of 17 or fewer errors in a block of 105 bits. Equation 3.26 shows the v_i's as linear combinations of a's and b's.

$$
\begin{array}{llll}
v_{30} + v_{45} = v_{60} + v_{90} = v_{15} + v_{75} = a_0 \\
v_{16} + v_{31} = v_{46} + v_{76} = v_{1} + v_{61} = a_0 \\
v_{2} + v_{17} = v_{32} + v_{62} = v_{92} + v_{47} = a_0 \\
v_{93} + v_{3} = v_{18} + v_{48} = v_{78} + v_{33} = a_0 \\
v_{79} + v_{94} = v_{4} + v_{34} = v_{64} + v_{19} = a_0 \\
v_{65} + v_{80} = v_{95} + v_{20} = v_{50} + v_{5} = a_0 \\
v_{51} + v_{66} = v_{81} + v_{6} = v_{36} + v_{96} = a_0 \\
v_{37} + v_{52} = v_{67} + v_{97} = v_{22} + v_{82} = a_0 \\
v_{23} + v_{38} = v_{53} + v_{83} = v_{8} + v_{68} = a_0 \\
v_{9} + v_{24} = v_{39} + v_{69} = v_{99} + v_{54} = a_0 \\
v_{100} + v_{10} = v_{25} + v_{55} = v_{85} + v_{40} = a_0 \\
v_{86} + v_{101} = v_{11} + v_{41} = v_{71} + v_{26} = a_0 \\
v_{72} + v_{87} = v_{102} + v_{27} = v_{57} + v_{12} = a_0 \\
v_{58} + v_{73} = v_{88} + v_{13} = v_{43} + v_{103} = a_0 \\
v_{44} + v_{59} = v_{74} + v_{104} = v_{29} + v_{89} = a_0
\end{array}
\qquad (3.24)
$$

$$
\begin{array}{llll}
v'_0 & & & v'_{75} \\
v'_{91} + v'_{63} + v'_7 & & & v'_{61} + v'_{33} + v'_{82} \\
v'_{98} + v'_{84} + v'_{56} & & & v'_{68} + v'_{54} + v'_{26} \\
v'_{77} + v'_{21} + v'_{14} & & & v'_{19} + v'_{12} + v'_{103} \\
v'_{49} + v'_{42} + v'_{28} & & & v'_{47} + v'_{96} + v'_{89} \\
 & & & = b_0 \qquad (3.25) \\
v'_{15} & & & v'_{90} \\
v'_1 + v'_{78} + v'_{22} & & & v'_{76} + v'_{48} + v'_{97} \\
v'_8 + v'_{99} + v'_{71} & & & v'_{83} + v'_{69} + v'_{41} \\
v'_{64} + v'_{57} + v'_{43} & & & v'_{34} + v'_{27} + v'_{13} \\
v'_{92} + v'_{36} + v'_{29} & & & v'_{62} + v'_6 + v'_{104} \\
 & & & \\
v'_{30} & & & \\
v'_{16} + v'_{93} + v'_{37} & & & \\
v'_{23} + v'_9 + v'_{86} & = b_0 & & \\
v'_{79} + v'_{72} + v'_{58} & & & \\
v'_2 + v'_{51} + v'_{44} & & & \\
 & & & \\
v'_{45} & & & \\
v'_{31} + v'_3 + v'_{52} & & & \\
v'_{38} + v'_{24} + v'_{101} & & & \\
v'_{94} + v'_{87} + v'_{73} & & & \\
v'_{17} + v'_{66} + v'_{59} & & & \\
 & & & \\
v'_{60} & & & \\
v'_{46} + v'_{18} + v'_{67} & & & \\
v'_{53} + v'_{39} + v'_{11} & & & \\
v'_4 + v'_{102} + v'_{88} & & & \\
v'_{32} + v'_{81} + v'_{74} & & &
\end{array}
$$

$$
\begin{array}{llll}
v_0 = a_0+b_0 & v_{15} = a_1+b_0 & v_{30} = a_2+b_0 & \\
v_{91} = a_0+b_1 & v_1 = a_1+b_1 & v_{16} = a_2+b_1 & \\
v_{77} = a_0+b_2 & v_{92} = a_1+b_2 & v_2 = a_2+b_2 & \\
v_{63} = a_0+b_3 & v_{78} = a_1+b_3 & v_{93} = a_2+b_3 & \\
v_{49} = a_0+b_4 & v_{64} = a_1+b_4 & v_{79} = a_2+b_4 & \\
v_{35} = a_0+b_5 & v_{50} = a_1+b_5 & v_{65} = a_2+b_5 & \\
v_{21} = a_0+b_6 & v_{36} = a_1+b_6 & v_{51} = a_2+b_6 & \\
v_7 = a_0+b_7 & v_{22} = a_1+b_7 & v_{37} = a_2+b_7 & \\
v_{98} = a_0+b_8 & v_8 = a_1+b_8 & v_{23} = a_2+b_8 & \\
v_{84} = a_0+b_9 & v_{99} = a_1+b_9 & v_9 = a_2+b_9 & \\
v_{70} = a_0+b_{10} & v_{85} = a_1+b_{10} & v_{100} = a_2+b_{10} & \\
v_{56} = a_0+b_{11} & v_{71} = a_1+b_4 & v_{86} = a_2+b_{11} & \\
v_{42} = a_0+b_{12} & v_{57} = a_1+b_{12} & v_{72} = a_2+b_{12} & \\
v_{28} = a_0+b_{13} & v_{43} = a_1+b_{13} & v_{58} = a_2+b_{13} & \\
v_{14} = a_0+b_{14} & v_{29} = a_1+b_{14} & v_{44} = a_2+b_{14} &
\end{array}
\qquad (3.26)
$$

$$
\begin{array}{llll}
v_{45} = a_3+b_0 & v_{60} = a_4+b_0 & v_{75} = a_5+b_0 & v_{90} = a_6+b_0 \\
v_{31} = a_3+b_1 & v_{46} = a_4+b_1 & v_{61} = a_5+b_1 & v_{76} = a_6+b_1 \\
v_{17} = a_3+b_2 & v_{32} = a_4+b_2 & v_{47} = a_5+b_2 & v_{62} = a_6+b_2 \\
v_3 = a_3+b_3 & v_{18} = a_4+b_3 & v_{33} = a_5+b_3 & v_{48} = a_6+b_3 \\
v_{94} = a_3+b_4 & v_4 = a_4+b_4 & v_{19} = a_5+b_4 & v_{34} = a_6+b_4 \\
v_{80} = a_3+b_5 & v_{95} = a_4+b_5 & v_5 = a_5+b_5 & v_{20} = a_6+b_5 \\
v_{66} = a_3+b_6 & v_{81} = a_4+b_6 & v_{96} = a_5+b_6 & v_6 = a_6+b_6 \\
v_{52} = a_3+b_7 & v_{67} = a_4+b_7 & v_{82} = a_5+b_7 & v_{97} = a_6+b_7 \\
v_{38} = a_3+b_8 & v_{53} = a_4+b_8 & v_{68} = a_5+b_8 & v_{83} = a_6+b_8 \\
v_{24} = a_3+b_9 & v_{39} = a_4+b_9 & v_{54} = a_5+b_9 & v_{69} = a_6+b_9 \\
v_{10} = a_3+b_{10} & v_{25} = a_4+b_{10} & v_{40} = a_5+b_{10} & v_{55} = a_6+b_{10} \\
v_{101} = a_3+b_{11} & v_{11} = a_4+b_{11} & v_{26} = a_5+b_{11} & v_{41} = a_6+b_{11} \\
v_{87} = a_3+b_{12} & v_{102} = a_4+b_{12} & v_{12} = a_5+b_{12} & v_{27} = a_6+b_{12} \\
v_{73} = a_3+b_{13} & v_{88} = a_4+b_{13} & v_{103} = a_5+b_{13} & v_{13} = a_6+b_{13} \\
v_{59} = a_3+b_{14} & v_{74} = a_4+b_{14} & v_{89} = a_5+b_{14} & v_{104} = a_6+b_{14}
\end{array}
$$

Theorem 3.7 [21]

If V_1 and V_2 are respectively the $(n_1 = 2^{m_1}-1,\ k_1 = m_1)$ and $(n_2 = 2^{m_2} - 1,\ k_2 = m_2)$ m-sequence codes, then $V^{(2)}$ can correct any error of weight $\frac{n_1 n_2 - n_1 - 2}{4}$ or less by 2-step majority logic decoding, where $n_1 < n_2$.

Proof: A word of $V^{(2)}$ $V^{(2)}(x)$ can be written as follows:

$$V^{(2)}(x) = A(x)\,\frac{1+x^n}{1+x^{n_1}} + B(x)\,\frac{1+x^n}{1+x^{n_2}} \qquad (3.27)$$

where

$$A(x) = a_0 + a_1(x) + a_2x^2 + \ldots a_{n_1-1}x^{n_1-1} \in V_1 \qquad (3.28)$$

$$B(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n_2-1}x^{n_2-1} \in V_2 \qquad (3.29)$$

$$V^{(2)}(x) = v_0 + v_1x + v_2x^2 + \ldots + v_{n-1}x^{n-1} \in V \qquad (3.30)$$

Since V_2 is an m-sequence code, there are $\frac{n_2-1}{2}$ orthogonal check-sums, on b_0, of the form

$$b_0 = b_\alpha + b_\beta. \qquad (3.31)$$

Since $(n_1, n_2)=1$, with respect to (3.28), (3.29), (3.30) we have $n = n_1n_2$ equations of the form

$$v_i = a_\gamma + b_\delta, \qquad (3.32)$$

where for every pair of γ and δ, there is a different i. This means that, by adding two appropriate equalities of (3.32) one can express b_0 of (3.31) as the sum of two v_j's. But then each a_j occurs n_1 time in (3.32) and there are $\frac{n_2-1}{2}$

orthogonal check-sums (3.32) a_0. Therefore, is concluded that we can form $\frac{n_1(n_2-1)}{2}$ check-sums, on b_0, of the form

$$b_0 = v_p + v_q. \tag{3.33}$$

Thus it is possible to determine every b_j by taking a majority vote on $\frac{n_1(n_2-1)}{2}$ estimates, since the codes are all cyclic.

Using these estimates in (3.32) then

$$v_1 + b_\delta = v_i' = a_\delta, \tag{3.34}$$

where v_i, importantly, is known. Since v_1 is an m-sequence code, one can form, on a_0, $\frac{n_1-1}{2}$ orthogonal check-sums of the form

$$a_0 = a_\theta + a_\lambda, \tag{3.35}$$

where $\theta \neq 0$, $\lambda \neq 0$. This means, since each a_j in (3.34) occurs n_2 times, that, by adding two appropriate equalities we can form $\frac{(n_1-1)n_2}{2}$ orthogonal estimates of the form

$$a_0 = v_x' + v_y', \tag{3.36}$$

where, emphatically, $x \neq 0$, $y \neq 0$. Again, in (3.34) a_0 occurs n_2 times. Thus, altogether, $n_2 + \frac{(n_1-1)n_2}{2} = \frac{n_1n_2+n_2}{2}$ orthogonal estimates on a_0.

Recalling that $\frac{n_1n_2-n_1}{2}$ orthogonal estimates on b_0, one now concludes, since $\frac{n_1n_2-n_1}{2} < \frac{n_1n_2+n_1}{2}$, that $V^{(2)}$ is 2-step

majority-logic-decodable up to, and including, $\frac{n_1n_2-n_1-2}{4}$ errors.

This concludes the proof of Theorem 3.7. Table 4 gives some 2-step majority-logic-decodable codes composed of two m-sequence codes.

Corollary 3.7.1

With reference to Theorem 3.4, if $n_1=3$, then $V^{(2)}$ is 2-step majority-logic-decodable.

Proof: The minimum distance of $^{(2)}$ from Corrolary 3.2.1 is

$$d = \min\{n_2d_1, \frac{n_1(n_2+1)}{2} - D_1\}$$

or

$$d = \min\{\frac{n_2(n_1+1)}{2}, \frac{n_1n_2-1}{2}\}$$

or

$$d = \frac{n_1n_2-1}{2}$$

if $n_1=3$, then in order for the code to be completely majority-logic-decodable $d-1 = \frac{n_1n_2-1}{2} - 1 = \frac{3n_2-3}{2}$ orthogonal equations on a sum of two noise digits should be formed. Because we can form $\frac{n_1n_2-n_1}{2}$ orthogonal equations, it follows that for $n_1=3$, it is possible to form $\frac{3n_2-3}{2} = d-1$ orthogonal equations.

Example 3.7.1

As an example of corollary 3.7.1 consider the case when V_1 is the $(n_1=3, k_1=2)$ m-sequence code with $g_1(x) = 1+x$ and

Table 4

2-STEP MAJORITY LOGIC DECODABLE CODES
COMPOSED OF TWO m-SEQUENCE

v_1			v_2			$v^{(2)}$		
n	k	d	n	k	d	n	k	d
3	2	2	7	3	4	21	5	10
3	2	2	31	5	16	93	7	46
3	2	2	127	7	64	381	10	190
3	2	2	511	9	256	1533	11	766
3	2	2	2047	11	1024	6141	13	3070
3	2	2	8191	13	4096	24573	15	12286
3	2	2	524287	19	262144	1572861	21	786430

V_2 is the $(n_2=7,\ k_2=3)$ m-sequence code with $g_2(x) = (1+x)(1+x+x^3)$. With reference to theorem 3.3, $V^{(2)}$ is the $(n=n_1n_2 = 21,\ k=k_1+k_2=5)$ code with $g(x) = (1+x)(1+x+x^3)(1+x+x^2+x^4+x^6)(1+x^2+x^4+x^5+x^6) = 1+x+x^2+x^3+x^4+x^6+x^8+x^{11}+x^{12}+x^{16}$.

If $A(x) = a_0 + a_1x + a_2x^2 \ \varepsilon V_1$

and $B(x) = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + b_6x^6 \ \varepsilon V_2$

Then a word $V^{(2)}(x)$ of $V^{(2)}$ is a combination of (a_i, b_j) as follows:

$$V^{(2)}(x) = a_0+a_1x+a_2x^2)\ \frac{1+x^{21}}{1+x^3} + (b_0+b_1x+b_2x^2+b_3x^3+b_4x^4+b_5x^5+b_6x^6\ \frac{1+x^{21}}{1+x^3}$$

and

$$\begin{aligned} V^2(x) = {} & (a_0+b_0) + (a_1+b_1)x + (a_2+b_2)x^2 + (a_0+b_3)x^3 \\ & + (a_1+b_4)x^4 + (a_2+b_5)x^5 + (a_0+b_6x^6) + (a_1+b_0)x^7 \\ & + (a_2+b_1)x^8 + (a_0+b_2)x^9 + (a_1+b_3)x^{10} + (a_2+b_4)x^{11} \\ & + (a_0+b_5)x^{12} + (a_1+b_6)x^{13} + (a_2+b_0)x^{14} \\ & + (a_0+b_1)x^{15} + (a_1+b_2)x^{16} + (a_2+b_3)x^{17} \\ & + (a_0+b_4)x^{18} + (a_1+b_5)x^{18} + (a_2+b_6)x^{20} \end{aligned}$$

Therefore all received v_i's are linear combinations of a's and b's as follows:

$$
\begin{array}{lll}
v_0 = a_0 + b_0 & v_7 = a_1 + b_0 & v_{14} = a_2 + b_0 \\
v_1 = a_1 + b_1 & v_8 = a_2 + b_1 & v_{15} = a_0 + b_1 \\
v_2 = a_2 + b_2 & v_9 = a_0 + b_2 & v_{16} = a_1 + b_2 \\
v_3 = a_0 + b_3 & v_{10} = a_1 + b_3 & v_{17} = a_2 + b_3 \\
v_4 = a_1 + b_4 & v_{11} = a_2 + b_4 & v_{18} = a_0 + b_4 \\
v_5 = a_2 + b_5 & v_{12} = a_0 + b_5 & v_{19} = a_1 + b_5 \\
v_6 = a_0 + b_6 & v_{13} = a_4 + b_6 & v_{20} = a_2 + b_6
\end{array} \qquad (3.37)
$$

where all the v_i's are the received digits. The parity check equations for the (3, 2) m-sequence code is $a_0 = a_1 + a_2$ and for the (7, 3) m-sequence code the parity check equations are

$$b_0 = b_2 + b_3,\ b_0 = b_4 + b_6,\ b_0 = b_1 + b_5$$

By adding two appropriate equalities of 3.37 we obtain

$$
\begin{array}{lll}
v_2 + v_{17} = b_0 & v_6 + v_{18} = b_0 & \\
v_4 + v_{13} = b_0 & v_{12} + v_{15} = b_0 & \\
v_1 + v_{19} = b_0 & v_0 + v_7 + v_{14} = b_0 & \\
v_3 + v_9 = b_0 & & \\
v_{11} + v_{20} = b_0 & & \\
v_5 + v_8 = b_0 & & \\
v_{10} + v_{16} = b_0 & &
\end{array}
$$

Therefore all the b_i's can be correctly estimated in the presence of 4 or fewer errors. Substituting those estimates in 3.37.

$$
\begin{array}{lll}
v_0 + b_0 = a_0 = v'_0 & v_7 + b_0 = a_1 = v'_7 & v_{14} + b_0 = a_2 = v'_{14} \\
v_1 + b_1 = a_1 = v'_8 & v_8 + b_1 = a_2 = v'_8 & v_{15} + b_1 = a_0 = v'_{15} \\
v_2 + b_2 = a_2 = v'_2 & v_9 + b_2 = a_0 = v'_9 & v_{16} + b_2 = a_1 = v'_{16} \\
v_3 + b_3 = a_0 = v'_3 & v_{10} + b_3 = a_1 = v'_{10} & v_{17} + b_3 = a_2 = v'_{17} \\
v_4 + b_4 = a_1 = v'_4 & v_{11} + b_4 = a_2 = v'_{11} & v_{18} + b_4 = a_0 = n'_{18} \\
v_5 + b_5 = a_2 = v'_5 & v_{12} + b_5 = a_0 = v'_{12} & v_{19} + b_5 = a_1 = v'_{19} \\
v_6 + b_5 = a_0 = v'_6 & v_{13} + b_6 = a_1 = v'_{13} & v_{20} + b_6 = a_2 = v'_{20}
\end{array} \qquad (3.38)
$$

from 3.38 it follows

$$
\begin{array}{lll}
v'_0 = a_0 & v'_7 + v'_8 = a_0 & v'_{15} = a_0 \\
v'_1 + v'_2 = a_0 & v'_9 = a_0 & v'_{16} + v'_{17} = a_0 \\
v'_3 = a_0 & v'_{10} + v'_{11} = a_0 & v'_{18} = a_0 \\
v'_4 + v'_5 = a_0 & v'_{12} = a_0 & v'_{14} + v'_{20} = a_0 \\
v'_6 = a_0 & v'_{13} + v'_{14} = a_0 &
\end{array} \qquad (3.38A)
$$

The 14 equations in 3.38A are orthogonal on a_0 therefore all a's can be estimated in the presence of 6 or fewer errors.

By using linear combinations of a's and b's an estimate of the received digits of $V^{(2)}(x)$ from equation 3.37 can be made.

Subsequently, consider the $V^{(3)}$ composite code for which $\mu=3$ and V_i, $i = 1, 2, 3$ are all m-sequence codes.

Theorem 3.8

Let V_1, V_2, V_3 be respectively the $(n_1 = 2^{m_1}-1, k_1 = m_1)$, $(n_2 = 2^{m_2}-1, k_2 = m_2)$ and $(n_3 = 2^{m_3}-1, k_3 = n_3)$ m-sequence codes. Then $V^{(3)}$ can correct any errors of weight $[\frac{n_1 n_2 n_3 - n_3(n_1-2)-2}{4}]$ or less by 3-step majority-logic-decoding where $n_1 < n_2 < n_3$ and

$[n_1, n_2, n_3] = 1$.

Proof: A word of $V^{(3)}$, $V^{(3)}(x)$ can be written as follows:

$$V(x) = A(x)\frac{1+x^n}{1+x^{n_1}} + B(x)\frac{1+x^n}{1+x^{n_2}} + C(x)\frac{1+x^n}{1+x^{n_3}} \tag{3.39}$$

where

$$A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n_1-1}x^{n_1-1} \quad \varepsilon V_1 \tag{3.40}$$

$$B(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n_2-1}x^{n_2-1} \quad \varepsilon V_2 \tag{3.41}$$

$$C(x) = c_0 + c_1x + c_2x^2 + \ldots + b_{n_3-1}x^{n_3-1} \quad \varepsilon V_3 \tag{3.42}$$

$$V^{(3)}(x) = v_0 + v_1x + v_2x^2 + \ldots + b_{n-1}x^{n-1} \quad \varepsilon V^{(3)} \tag{3.43}$$

Since V_3 is an m-sequence code we have $\frac{n_3-1}{2}$ orthogonal check sums on c_0 of the form

$$c_0 = c_\alpha + c_\beta \tag{3.44}$$

Since $[n_1, n_2, n_3] = 1$ with respect to (3.40), (3.41), (3.42) one has $n = n_1n_2n_3$ equations of the form

$$v_i = a_\gamma + b_\delta + c_\varepsilon \tag{3.45}$$

where for every different triplet of γ, δ, and ε, there is a different i. This means that by adding two appropriate equalities of (3.45) one can express c_0 of (3.44) as the sum of two v_i's but since each b_j occurs n_2 times in (3.45), and a_j occurs n_1 times in (3.45), and there are $n_3-1/2$ orthogonal check sums (3.44) one therefore concludes that $n_1n_2(n_3-1)/2$

check sums are formed on c_0 of the form

$$c_0 = v_p + v_q \tag{3.46}$$

Thus a determination can be made of every c_j by taking a majority vote on $n_1n_2(n_3-1)/2$ estimates since the code is cyclic.

Using these estimates in (3.45) it follows that

$$v_i + c_\varepsilon = a_\gamma + b_\delta = v_i' \tag{3.47}$$

where v_i' is known. Since n_2 is an m-sequence code on b_0 $n_2-1/2$ orthogonal check sums of the form

$$b_0 = b_\theta + b_\gamma$$

can be made, where $\theta \neq 0$, $\lambda \neq 0$. Since b_j occurs n_1n_3 times, by adding two appropriate equalities of (3.47) one can derive $\frac{n_1n_3(n_2-1)}{2}$ equations of the form $b_0 = v_x' + v_y'$ where $x \neq 0$, $y \neq 0$, again in (3.47) b_0 occurs n_1n_3 times. By taking n_3 sums of n_1 terms each of the form $v_x' + v_y' + v_z' + \ldots + v_t'$ and since the number of v's is n_1, where n_1 is odd, there are all together

$$n_3 + \frac{n_1n_3(n_2-1)}{2} \tag{3.48}$$

orthogonal estimates on b_0. Using b_0 estimates in (3.47)

$$v_i' + b_\delta = v_i'' = a_\gamma \tag{3.49}$$

develops. Since v_1 is an m-sequence code we can form $n_1-1/2$ orthogonal check sums of the form

$$a_0 = a_\theta + a_\lambda \tag{3.50}$$

where $\theta \neq 0$, $\lambda \neq 0$. This means that since each a_j in (3.37) occurs, n_2n_3 times, $(\frac{n_1-1}{2})n_2n_3$ orthogonal estimates can be derived of the form

$$a_0 = v''_x + v''_y \tag{3.51}$$

Again in (3.49) a_0 occurs n_2n_3 times thus all together:

$$n_2n_3 + (\frac{n_1-1}{2})n_2n_3 = \frac{n_1n_2n_3+n_2n_3}{2}$$

orthogonal estimates on a_0 are apparent. Recalling that $\frac{n_1n_2-n_1n_2}{2}$ orthogonal estimates on c_0 and $\frac{2n_3+n_1n_2n_3-n_1n_3}{2}$ on b_0 and since

$$\frac{n_1n_2n_3-n_3(n_1-2)}{2} < \frac{n_1n_2n_3-n_1n_2}{2} < \frac{n_1n_2n_3+n_2n_3}{2}$$

concludes that $v^{(3)}$ can correct by 3-step majority-logic-decoding $\frac{n_1n_2n_3-n_3(n_1-2)-2}{4}$ or fewer errors.

Table 5 gives a listing of 3-step majority-logic-decodable codes composed of 3 m-sequence codes.

Corollary 3.8.1

With reference to Theorem 3.5 if $n_1=3$ then $v^{(3)}$ is 3-step majority-logic-decodable.

Table 5

3-STEP MAJORITY LOGIC DECODABLE CODES
COMPOSED OF THREE M-SEQUENCE CODES

v_1			v_2			v_3			$v^{(3)}$		
n	k	d	n	k	d	n	k	d	n	k	d
3	2	2	31	5	16	127	7	64	11811	14	5842
3	2	2	31	5	16	511	9	256	47523	16	23506
3	2	2	31	5	16	2047	11	1024	190371	18	94162
3	2	2	31	5	16	8191	13	4096	761763	20	376786
3	2	2	7	3	4	31	5	16	651	10	310
3	2	2	7	3	4	127	7	64	2667	12	1270
3	2	2	7	3	4	2047	11	1024	42987	16	20470
3	2	2	7	3	4	8191	13	4096	172011	18	81910
3	2	2	7	3	4	131071	17	65536	2752491	22	1310710
3	2	2	7	3	4	524287	19	262144	11010027	24	5242870
3	2	2	127	7	64	511	9	256	194691	18	97090
3	2	2	127	7	64	2047	11	1024	779907	20	388930
3	2	2	127	7	64	8191	13	4096	3120771	22	1556290
3	2	2	511	9	256	2047	11	1024	3138051	22	1568002

Proof: If $n_1=3$ one can form at least $\frac{n_3 n_1 n_2 - n_3}{2}$ orthogonal estimates or $\frac{(3n_2-1)}{2} n_3$, on the other hand, the minimum distance of $V^{(3)}$ is

$$d_{min} = \frac{(n_1 n_2 - 1)}{2} = \frac{3n_2-1}{2} n_3$$

Therefore the number of orthogonal equations equals to d_{min}.

Example 3.8.2

As an example of corollary 3.8.1, consider the (651, 10) composite code with $d_{min} = 310$ and composed of the following m-sequence codes: V_1 is the ($n_1=3$, $k_1=2$) m-sequence code, V_2 is the ($n_2=7$, $k_2=3$) m-sequence code and V_3 is the ($n_2=31$, $k_3=5$) m-sequence code.

The orthogonal check equations on a_0, b_0 and c_0 are given in Appendix A.

Theorem 3.9

Let V_1 be an (n_1, k_1) binary cyclic code which has minimum distance d_1, and is 1-step majority-logic-decodable. Let V_2 be an ($n_2=2^{m_2}-1$, $k_2=m_2$) m-sequence code, and let $V^{(3)}$ be an ($n_3=2^{m_3-1}$, $k_3=m_3$) m-sequence code, where n_1, n_2 and n_3 are pairwise prime and $n_3 > n_2$. Then, the composite code $V^{(3)}$ can correct by 3-step majority-logic-decoding $[\frac{n_2 n_3 d_1 - 1}{2}]$ or fewer errors if $d_1 \leq \frac{n_1 n_2 - n_1 + 2}{2}$, and $[\frac{n_1 n_2 n_3 - n_3 n_1 + 2n_3 - 2}{4}]$ or fewer errors if $d_1 \geq \frac{n_1 n_2 - n_1 + 2}{2n_2}$.

Proof: Let $a_0a_1a_2 \ldots a_{n_1-1}$ be a word of v_1, $b_0b_1n_2 \ldots b_{n_2-1}$ be a word of v_2, and $c_0c_1c_2 \ldots c_{n_3-1}$ a word of v_3; then every word $v_0v_1v_2 \ldots v_{n-1}$ of $v^{(3)}$ is constructed according to the rule:

$$V(x) = A(x)\frac{1+x^n}{1+x^{n_1}} + B(x)\frac{1+x^n}{1+x^{n_2}} + C(x)\frac{1+x^n}{1+x^{n_3}} \quad V(x)\varepsilon V^{(3)}$$

and since n_1, n_2, n_3 are pairwise prime every v_p of $V(x)$ can be expressed in the form

$$v_p = a_q + b_r + c_s \tag{3.52}$$

and there is only one p for every given triplet q, r, s. Every q occurs in (3.52) n_1 times, every r occurs in (3.52) n_2 times. Therefore, for each one of n_3 c_i's where, i = 0, 1, 2, 3, ... n_3 one can form n_1n_2 sums of the form

$$v_{kn_3+i} = a_{k+i} + b_{n_1k+i} + c_i \tag{3.53}$$

where k = 0, 1, 2, 3, ..., n_1n_2

i = 0, 1, 2, 3, ..., n_3

and subscripts of v_{kn_3+i} are mod $(1+x^n)$

subscripts of a_{k+i} are mod $(1+x^{n_1})$

subscripts of b_{n_1k+1} are mod $(1+x^{n_2})$.

Since V_3 is an m-sequence code we can form on each c_i $n_3-1/2$ orthogonal check sums of the form

$$c_i = c_j + c_d \tag{3.54}$$

Therefore by adding two appropriate equalities of (3.53) one may derive $\frac{n_1 n_2 (n_3-1)}{2}$ orthogonal check sums on each c_i of the form

$$c_i = v_n + v_t \tag{3.55}$$

and as a result every c_i can be estimated by majority, if the number of errors is $\frac{n_1 n_2 n_3 - n_1 n_2 - 2}{4}$ or less.

After estimating each c_i, (3.53) can be rewritten in the form

$$v_{kn_3+i} + c_i = a_{k+i} + b_{n_1 k} + i = v'_{kn_3+i}$$

or

$$v'_{kn_3} + i = a_{k+i} + b_{n_1 k+i} \tag{3.56}$$

Every $bk_{n_1}+i$ occurs in (3.56) $n_1 n_3$ times. Since V_2 is an m-sequence code, $n_2-1/2$ orthogonal check sums on each $b_{n_1 k+i}$ can be made of the form

$$b_{n_1+1} = b_\theta + b_\lambda$$

Therefore $\frac{n_1 n_3 (n_2-1)}{2}$ orthogonal check sums on each $b_{n_1 k+i}$ can be formed. Subsequently, by taking n_3 sums of n_1 terms each all together $n_3 + \frac{n_1 n_3 (n_2-1)}{2}$ orthogonal check sums on each $b_{n_1 k+i}$ can be formed. As a result, each $b_{n_1 k+i}$ can be majority-logic-decoded if $\frac{2n_3 + n_1 n_3 (n_2-1) - 2}{4}$ or fewer errors have occurred. Using these estimates (eq. 3.56) can be written as

$$V''_{kn_3+i} = a_{k+i} \tag{3.58}$$

V_1 is one-step majority-logic-decodable, therefore

d_1-1 orthogonal check sums on each a_{k+i} can be formed as

$$a_{k+i} = a_c + a_j + \ldots + a_x \tag{3.59}$$

Since every a_{k+i} occurs in 3.58, n_2n_3 times, $(d_1-1)n_2n_3$ orthogonal check sums on each a_{k+i} can be developed of the form

$$a_{k+i} = v''_x + v''_y + \ldots + v''_z \tag{3.60}$$

Again every a_{k+i} occurs n_2n_3 times in (3.58), therefore, $n_2n_3 + (d_1-1)n_2n_3 = n_2n_3d_1$ orthogonal estimates on every a_{k+i} can be formed, and every a_{k+i} can be majority-logic-decoded in the presence of $\frac{n_2n_3d_1-1}{2}$ or fewer errors.

Two corollaries are developed from the theorem.

Corollary 3.9.1

If with reference to the theorem, V_1 has the word of all ones, so that $D_1 = n_1$ and if $d_1 \leq \frac{n_1n_2-n_1}{2n_2}$, then $V^{(3)}$ is 3-step majority-logic-decodable.

Proof: In order to find the minimum distance of $V^{(3)}$, we proceed as follows: the minimum distance d' of V', where ' is the composite code of V_2 and V_3, and since both V_2 and V_3 are m-sequence codes is given by $d' = \frac{n_2n_3-1}{2}$ and its maximum distance D' is given by $D' = \frac{n_2+1}{2} n_3$.

From Theorem 3.2 it is known that for a two component composite code in general, d is given by

$$d = \min\{n_2 d_1,\ n_1 d_2,\ n_2 d_1 + n_1 D_2 - \overline{2D_1 D_2}\}$$

Therefore the composite code V" of V' and V_1, which is the code $V^{(3)}$ will have minimum distance.

$$d = \min\{n_2 n_3 d_1,\ \frac{n_1 n_2 n_3 - n_1}{2},\ \frac{n_1 n_2 n_3 - n_3(2D_1 - n_1)}{2}\} \qquad (3.60)$$

Since $D_1 = n_1$ substituting D_1 in 3.60 we obtain

$$d = \min\{n_2 n_3 d_1,\ \frac{n_1 n_2 n_3 - n_1}{2},\ \frac{n_1 n_2 n_3 - n_3 n_1}{2}\} \qquad (3.61)$$

From 3.61 it is obvious that

$$d = \min\{n_2 n_3 d_1,\ \frac{n_1 n_3 n_3 - n_3 n_1}{2}\}$$

Since

$$d_1 \leq \frac{n_1 n_2 - n_1}{2n_2}$$

or $n_2 d_1 \leq \frac{n_1 n_2 - n_1}{2}$

or $n_2 n_3 d_1 \leq \frac{n_1 n_2 n_3 - n_1 n_3}{2}$, it follows that

$d \leq n_2 n_3 d_1$ which coincides with the smallest number of orthogonal check sums that can possibly be formed. Therefore, the code is 3-step majority-logic-decodable.

Corollary 3.9.2

If with reference to the theorem V_1 is even weighted with $D_1 = n_1 - d_1 + 1$, and if $d_1 \leq \frac{n_1(n_2-1)-2}{2(n_2-1)}$, then $V^{(3)}$ is 3-step majority-logic-decodable.

Proof: From equation 3.60

$$d = \min\{n_2 n_3 d_1, \frac{n_1 n_2 n_3 - n_1}{2}, \frac{n_1 n_2 n_3 - n_3(n_1 - 2d_1 + 2)}{2}\}$$

or $$d = \min\{n_2 n_3 d_1, \frac{n_1 n_2 n_3 - n_1}{2}, \frac{n_1 n_2 n_3 - n_3[n_1 - 2(d_1 - 1)]}{2}\}$$

from which

$$d = \min\{n_2 n_3 d_1, \frac{n_1 n_2 n_3 - n_3[n_1 - 2(d_1 - 1)]}{2}\}$$

since

$$d_1 \leq \frac{n_1(n_2 - 1) - 2}{2(n_2 - 1)}$$

thus

$$d_1(n_2 - 1) \leq \frac{n_1(n_2 - 1) - 2}{2}$$

$$d_1 n_2 n_3 \leq \frac{n_3 n_1(n_2 - 1) - 2n_3 + 2d_1 n_3}{2}$$

$$d_1 n_2 n_3 \leq \frac{n_1 n_2 n_3 - n_3[n_1 - 2(d_1 - 1)]}{2} \qquad (3.62)$$

Equation 3.62 suggests that the minimum number of orthogonal check equations coincides with the minimum distance of the code. Therefore $v^{(3)}$ is 3-step majority-logic decodable.

CHAPTER 4

CONCLUDING REMARKS

This thesis introduced a class of composite codes. These codes are parallel to the codes devised by Jet Propulsion Laboratories, and were used during the Marriner experiment for picture transmission from deep space. The question of the composite code dimensions was discussed, and a recursive expression for calculating the weight distribution of an μ-component composite code was stated. An analytical expression for the minimum distance of a 2-component composite code was also discussed.

Proof was provided that a two component composite code can be majority-logic-decoded in L+1 steps, given that V_1 is an L-step majority-logic-decodable code, and V_2 is an m-sequence code.

Subsequently, proof was given that a two component composite code, where one of the codes is an m-sequence code and the other is any one-step majority-logic decodable code, that majority-logic-decoding of the composite code, can be accomplished in two steps.

The above result was extended for the case of a 3 component composite code.

Proof was also provided that in the case of a 2 component composite code where the two codes are m-sequence codes, the composite code can correct $\frac{n_1 n_2 - n_1 - 2}{4}$ or fewer errors with two-step majority-logic-decoding, where n_1 and n_2 are the lengths of the two component codes. It was also proven that for the case $n_1=3$ the composite code is 2-step majority-logic-decodable. For the case of the 3 m-sequence composite code it was proven that we can correct $\frac{n_1 n_2 n_3 - n_3(n_1-2)}{4}$ or fewer errors with 3-step majority-logic-decoding, and if $n_1=3$ it was proven that the composite code is 3-step majority-logic-decodable.

The author submits that the stated composite codes can be used in the context of spread spectrum systems for code synchronization and acquisition.

APPENDIX A

DECODING OF THE (651, 10) d=310 COMPOSITE CYCLIC CODE

Equation A1 below gives all v_i's as linear combinations of a's, b's and c's of the component codes

$$
\begin{array}{lllll}
v_0 = a_0+b_0+c_0 & v_{31}=a_1+b_3+c_0 & v_{62}=a_2+b_6+c_0 & v_{93} = a_0+b_2+c_0 & v_{124}=a_1+b_5+c_0 \\
v_1 = a_1+b_1+c_1 & v_{32}=a_2+b_4+c_1 & v_{63}=a_0+b_0+c_1 & v_{94} = a_1+b_3+c_1 & v_{125}=a_2+b_6+c_1 \\
v_2 = a_2+b_2+c_2 & v_{33}=a_0+b_3+c_2 & v_{64}=a_1+b_1+c_2 & v_{95} = a_1+b_4+c_2 & v_{126}=a_0+b_0+c_2 \\
v_3 = a_0+b_3+c_3 & v_{34}=a_1+b_6+c_3 & v_{65}=a_2+b_2+c_3 & v_{96} = a_0+b_5+c_3 & v_{127}=a_1+b_1+c_3 \\
v_4 = a_1+b_4+c_4 & v_{35}=a_2+b_0+c_4 & v_{66}=a_0+b_3+c_4 & v_{97} = a_1+b_6+c_4 & v_{128}=a_2+b_2+c_4 \\
v_5 = a_2+b_5+c_5 & v_{36}=a_0+b_1+c_5 & v_{67}=a_1+b_4+c_5 & v_{98} = a_2+b_0+c_5 & v_{129}=a_0+b_3+c_5 \\
v_6 = a_0+b_6+c_6 & v_{37}=a_1+b_2+c_6 & v_{68}=a_2+b_5+c_6 & v_{99} = a_0+b_1+c_6 & v_{130}=a_1+b_4+c_6 \\
v_7 = a_1+b_0+c_7 & v_{38}=a_2+b_3+c_7 & v_{69}=a_0+b_6+c_7 & v_{100}=a_1+b_2+c_7 & v_{131}=a_2+b_5+c_7 \\
v_8 = a_2+b_1+c_8 & v_{39}=a_0+b_4+c_8 & v_{70}=a_1+b_0+c_8 & v_{101}=a_2+b_3+c_8 & v_{132}=a_0+b_6+c_8 \\
v_9 = a_0+b_2+c_9 & v_{40}=a_1+b_5+c_9 & v_{71}=a_2+b_1+c_9 & v_{102}=a_0+b_4+c_9 & v_{133}=a_1+b_0+c_9 \\
v_{10}=a_1+b_3+c_{10} & v_{41}=a_0+b_6+c_{10} & v_{72}=a_0+b_2+c_{10} & v_{103}=a_1+b_5+c_{10} & v_{134}=a_2+b_1+c_{10} \\
v_{11}=a_2+b_4+c_{11} & v_{42}=a_0+b_0+c_{11} & v_{73}=a_1+b_3+c_{11} & v_{104}=a_2+b_6+c_{11} & v_{135}=a_0+b_2+c_{11} \\
v_{12}=a_0+b_5+c_{12} & v_{43}=a_1+b_1+c_{12} & v_{74}=a_2+b_4+c_{12} & v_{105}=a_0+b_0+c_{12} & v_{136}=a_1+b_3+c_{12} \\
v_{13}=a_1+b_6+c_{13} & v_{44}=a_2+b_2+c_{13} & v_{75}=a_0+b_3+c_{13} & v_{106}=a_1+b_1+c_{13} & v_{137}=a_2+b_4+c_{13} \\
v_{14}=a_2+b_0+c_{14} & v_{45}=a_0+b_3+c_{14} & v_{76}=a_1+b_6+c_{14} & v_{107}=a_2+b_2+c_{14} & v_{138}=a_0+b_5+c_{14} \\
v_{15}=a_0+b_1+c_{15} & v_{46}=a_1+b_4+c_{15} & v_{77}=a_2+b_0+c_{15} & v_{108}=a_0+b_3+c_{15} & v_{139}=a_1+b_6+c_{15} \\
v_{16}=a_1+b_2+c_{16} & v_{47}=a_2+b_5+c_{16} & v_{78}=a_0+b_1+c_{16} & v_{109}=a_1+b_4+c_{16} & v_{140}=a_2+b_0+c_{16} \\
v_{17}=a_2+b_3+c_{17} & v_{48}=a_0+b_6+c_{17} & v_{79}=a_1+b_2+c_{17} & v_{110}=a_1+b_5+c_{17} & v_{141}=a_0+b_1+c_{17} \\
v_{18}=a_0+b_4+c_{18} & v_{49}=a_1+b_0+c_{18} & v_{80}=a_2+b_3+c_{18} & v_{111}=a_0+b_6+c_{18} & v_{142}=a_1+b_2+c_{18} \\
v_{19}=a_1+b_5+c_{19} & v_{50}=a_2+b_1+c_{19} & v_{81}=a_0+b_4+c_{19} & v_{112}=a_1+b_0+c_{19} & v_{143}=a_2+b_3+c_{19} \\
v_{20}=a_2+b_6+c_{20} & v_{51}=a_0+b_2+c_{20} & v_{82}=a_1+b_5+c_{20} & v_{113}=a_2+b_1+c_{20} & v_{144}=a_0+b_4+c_{20} \\
v_{21}=a_0+b_0+c_{21} & v_{52}=a_1+b_3+c_{21} & v_{83}=a_2+b_6+c_{21} & v_{114}=a_0+b_2+c_{21} & v_{145}=a_1+b_5+c_{21} \\
v_{22}=a_1+b_1+c_{22} & v_{53}=a_2+b_4+c_{22} & v_{84}=a_0+b_0+c_{22} & v_{115}=a_1+b_3+c_{22} & v_{146}=a_2+b_6+c_{22} \\
v_{23}=a_2+b_2+c_{23} & v_{54}=a_0+b_5+c_{23} & v_{85}=a_1+b_1+c_{23} & v_{116}=a_2+b_4+c_{23} & v_{147}=a_0+b_0+c_{23} \\
v_{24}=a_0+b_3+c_{24} & v_{55}=a_1+b_6+c_{24} & v_{86}=a_2+b_2+c_{24} & v_{117}=a_0+b_5+c_{24} & v_{148}=a_1+b_1+c_{24} \\
v_{25}=a_1+b_4+c_{25} & v_{56}=a_2+b_0+c_{25} & v_{87}=a_0+b_3+c_{25} & v_{118}=a_1+b_5+c_{25} & v_{149}=a_2+b_2+c_{25} \\
v_{26}=a_2+b_5+c_{26} & v_{57}=a_0+b_1+c_{26} & v_{88}=a_1+b_4+c_{26} & v_{119}=a_2+b_0+c_{26} & v_{150}=a_0+b_3+c_{26} \\
v_{27}=a_0+b_6+c_{27} & v_{58}=a_1+b_2+c_{27} & v_{89}=a_2+b_5+c_{27} & v_{120}=a_0+b_1+c_{27} & v_{151}=a_1+b_4+c_{27} \\
v_{28}=a_1+b_0+c_{28} & v_{59}=a_2+b_3+c_{28} & v_{90}=a_0+b_6+c_{28} & v_{121}=a_1+b_2+c_{28} & v_{152}=a_2+b_3+c_{28} \\
v_{29}=a_2+b_1+c_{29} & v_{60}=a_0+b_1+c_{29} & v_{91}=a_1+b_0+c_{29} & v_{122}=a_2+b_3+c_{29} & v_{153}=a_0+b_6+c_{29} \\
v_{30}=a_0+b_2+c_{30}. & v_{61}=a_1+b_5+c_{30} & v_{92}=a_2+b_1+c_{30} & v_{123}=a_2+b_4+c_{30} & v_{154}=a_1+b_0+c_{30}
\end{array}
\qquad \text{(A1)}
$$

$$
\begin{array}{lllll}
v_{155}=a_2+b_1+c_0 & v_{186}=a_0+b_4+c_0 & v_{217}=a_1+b_0+c_0 & v_{248}=a_2+b_3+c_0 & v_{279}=a_0+b_6+c_0 \\
v_{156}=a_0+b_2+c_1 & v_{187}=a_1+b_5+c_1 & v_{218}=a_2+b_1+c_1 & v_{249}=a_0+b_4+c_1 & v_{280}=a_1+b_0+c_1 \\
v_{157}=a_1+b_3+c_2 & v_{188}=a_2+b_6+c_2 & v_{219}=a_0+b_2+c_2 & v_{250}=a_1+b_5+c_2 & v_{281}=a_2+b_1+c_2 \\
v_{158}=a_2+b_1+c_3 & v_{189}=a_0+b_0+c_3 & v_{220}=a_1+b_3+c_3 & v_{251}=a_2+b_6+c_3 & v_{282}=a_0+b_2+c_3 \\
v_{159}=a_0+b_5+c_4 & v_{190}=a_1+b_1+c_4 & v_{221}=a_2+b_4+c_4 & v_{252}=a_0+b_0+c_4 & v_{283}=a_1+b_3+c_4 \\
v_{160}=a_1+b_6+c_5 & v_{191}=a_2+b_2+c_5 & v_{222}=a_0+b_5+c_5 & v_{253}=a_1+b_1+c_5 & v_{284}=a_2+b_1+c_5 \\
v_{161}=a_2+b_0+c_6 & v_{192}=a_0+b_3+c_6 & v_{223}=a_1+b_6+c_6 & v_{254}=a_2+b_2+c_6 & v_{285}=a_0+b_5+c_6 \\
v_{162}=a_0+b_1+c_7 & v_{193}=a_1+b_4+c_7 & v_{224}=a_2+b_0+c_7 & v_{255}=a_0+b_3+c_7 & v_{286}=a_1+b_6+c_7 \\
v_{163}=a_1+b_2+c_8 & v_{194}=a_2+b_5+c_8 & v_{225}=a_0+b_1+c_8 & v_{256}=a_1+b_4+c_8 & v_{287}=a_2+b_0+c_8 \\
v_{164}=a_2+b_3+c_9 & v_{195}=a_0+b_6+c_9 & v_{226}=a_1+b_2+c_9 & v_{257}=a_2+b_5+c_9 & v_{288}=a_0+b_1+c_9 \\
v_{165}=a_0+b_4+c_{10} & v_{196}=a_1+b_0+c_{10} & v_{227}=a_2+b_3+c_{10} & v_{258}=a_0+b_6+c_{10} & v_{289}=a_1+b_2+c_{10} \\
v_{166}=a_1+b_5+c_{11} & v_{197}=a_2+b_1+c_{11} & v_{228}=a_0+b_4+c_{11} & v_{259}=a_1+b_0+c_{11} & v_{290}=a_2+b_3+c_{11} \\
v_{167}=a_2+b_6+c_{12} & v_{198}=a_0+b_2+c_{12} & v_{229}=a_1+b_5+c_{12} & v_{260}=a_2+b_1+c_{12} & v_{291}=a_0+b_4+c_{12} \\
v_{168}=a_0+b_0+c_{13} & v_{199}=a_1+b_3+c_{13} & v_{230}=a_2+b_6+c_{13} & v_{261}=a_0+b_2+c_{13} & v_{292}=a_1+b_5+c_{13} \\
v_{169}=a_1+b_1+c_{14} & v_{200}=a_2+b_4+c_{14} & v_{231}=a_0+b_0+c_{14} & v_{262}=a_1+b_3+c_{14} & v_{293}=a_2+b_6+c_{14} \\
v_{170}=a_2+b_2+c_{15} & v_{201}=a_0+b_5+c_{15} & v_{232}=a_1+b_0+c_{15} & v_{263}=a_2+b_4+c_{15} & v_{294}=a_0+b_0+c_{15} \\
v_{171}=a_0+b_3+c_{16} & v_{202}=a_1+b_6+c_{16} & v_{233}=a_2+b_2+c_{16} & v_{264}=a_0+b_5+c_{16} & v_{295}=a_1+b_1+c_{16} \\
v_{172}=a_1+b_4+c_{17} & v_{203}=a_2+b_0+c_{17} & v_{234}=a_0+b_3+c_{17} & v_{265}=a_1+b_6+c_{17} & v_{296}=a_2+b_2+c_{17} \\
v_{173}=a_2+b_5+c_{18} & v_{204}=a_0+b_1+c_{18} & v_{235}=a_1+b_4+c_{18} & v_{266}=a_2+b_0+c_{18} & v_{297}=a_0+b_3+c_{18} \\
v_{174}=a_0+b_6+c_{19} & v_{205}=a_1+b_2+c_{19} & v_{236}=a_2+b_5+c_{19} & v_{267}=a_0+b_1+c_{19} & v_{298}=a_1+b_4+c_{19} \\
v_{175}=a_1+b_0+c_{20} & v_{206}=a_2+b_3+c_{20} & v_{237}=a_0+b_6+c_{20} & v_{268}=a_1+b_2+c_{20} & v_{299}=a_2+b_5+c_{20} \\
v_{176}=a_2+b_1+c_{21} & v_{207}=a_0+b_1+c_{21} & v_{238}=a_1+b_0+c_{21} & v_{269}=a_2+b_3+c_{21} & v_{300}=a_0+b_6+c_{21} \\
v_{177}=a_0+b_2+c_{22} & v_{208}=a_1+b_5+c_{22} & v_{239}=a_2+b_1+c_{22} & v_{270}=a_0+b_4+c_{22} & v_{301}=a_1+b_0+c_{22} \\
v_{178}=a_1+b_3+c_{23} & v_{209}=a_2+b_6+c_{23} & v_{240}=a_0+b_2+c_{23} & v_{271}=a_1+b_5+c_{23} & v_{302}=a_2+b_1+c_{23} \\
v_{179}=a_2+b_4+c_{24} & v_{210}=a_0+b_0+c_{24} & v_{241}=a_1+b_3+c_{24} & v_{272}=a_2+b_6+c_{24} & v_{303}=a_0+b_2+c_{24} \\
v_{180}=a_0+b_5+c_{25} & v_{211}=a_1+b_1+c_{25} & v_{242}=a_1+b_1+c_{25} & v_{273}=a_0+b_0+c_{25} & v_{304}=a_1+b_3+c_{25} \\
v_{181}=a_1+b_6+c_{26} & v_{212}=a_2+b_2+c_{26} & v_{243}=a_0+b_5+c_{26} & v_{274}=a_1+b_1+c_{26} & v_{305}=a_2+b_4+c_{26} \\
v_{182}=a_2+b_0+c_{27} & v_{213}=a_0+b_3+c_{27} & v_{244}=a_1+b_6+c_{27} & v_{275}=a_2+b_2+c_{27} & v_{306}=a_0+b_5+c_{27} \\
v_{183}=a_0+b_1+c_{28} & v_{214}=a_1+b_4+c_{28} & v_{245}=a_2+b_0+c_{28} & v_{276}=a_0+b_3+c_{28} & v_{307}=a_1+b_6+c_{28} \\
v_{184}=a_1+b_2+c_{29} & v_{215}=a_2+b_5+c_{29} & v_{246}=a_0+b_1+c_{29} & v_{277}=a_1+b_4+c_{29} & v_{308}=a_2+b_0+c_{29} \\
v_{185}=a_0+b_3+c_{30} & v_{216}=a_0+b_0+c_{30} & v_{247}=a_1+b_2+c_{30} & v_{278}=a_2+b_5+c_{30} & v_{309}=a_0+b_1+c_{30}
\end{array}
\qquad \text{(A1)}
$$

$$
\begin{array}{lllll}
v_{310}=a_1+b_2+c_0 & v_{341}=a_2+b_5+c_0 & v_{372}=a_0+b_1+c_0 & v_{403}=a_1+b_4+c_0 & v_{434}=a_2+b_0+c_0 \\
v_{311}=a_2+b_3+c_1 & v_{342}=a_0+b_6+c_1 & v_{373}=a_1+b_2+c_1 & v_{404}=a_2+b_5+c_1 & v_{435}=a_0+b_1+c_1 \\
v_{312}=a_0+b_4+c_2 & v_{343}=a_1+b_0+c_2 & v_{374}=a_2+b_3+c_2 & v_{405}=a_0+b_6+c_2 & v_{436}=a_1+b_2+c_2 \\
v_{313}=a_1+b_5+c_3 & v_{344}=a_2+b_1+c_3 & v_{375}=a_0+b_4+c_3 & v_{406}=a_1+b_0+c_3 & v_{437}=a_2+b_3+c_3 \\
v_{314}=a_2+b_6+c_4 & v_{345}=a_0+b_2+c_4 & v_{376}=a_1+b_5+c_4 & v_{407}=a_2+b_1+c_4 & v_{438}=a_0+b_4+c_4 \\
v_{315}=a_0+b_0+c_5 & v_{346}=a_1+b_3+c_5 & v_{377}=a_2+b_6+c_5 & v_{408}=a_0+b_2+c_5 & v_{439}=a_1+b_5+c_5 \\
v_{316}=a_1+b_1+c_6 & v_{347}=a_2+b_4+c_6 & v_{378}=a_0+b_0+c_6 & v_{409}=a_1+b_3+c_6 & v_{440}=a_2+b_6+c_6 \\
v_{317}=a_2+b_2+c_7 & v_{348}=a_0+b_5+c_7 & v_{379}=a_1+b_1+c_7 & v_{410}=a_2+b_4+c_7 & v_{441}=a_0+b_0+c_7 \\
v_{318}=a_0+b_3+c_8 & v_{349}=a_1+b_6+c_8 & v_{380}=a_2+b_2+c_8 & v_{411}=a_0+b_5+c_8 & v_{442}=a_1+b_1+c_8 \\
v_{319}=a_1+b_4+c_9 & v_{350}=a_2+b_0+c_9 & v_{381}=a_0+b_3+c_9 & v_{412}=a_1+b_6+c_9 & v_{443}=a_2+b_2+c_9 \\
v_{320}=a_2+b_5+c_{10} & v_{351}=a_0+b_1+c_{10} & v_{382}=a_1+b_4+c_{10} & v_{413}=a_2+b_0+c_{10} & v_{444}=a_0+b_3+c_{10} \\
v_{321}=a_0+b_6+c_{11} & v_{352}=a_1+b_2+c_{11} & v_{383}=a_2+b_5+c_{11} & v_{414}=a_0+b_1+c_{11} & v_{445}=a_1+b_4+c_{11} \\
v_{322}=a_1+b_0+c_{12} & v_{353}=a_2+b_3+c_{12} & v_{384}=a_0+b_6+c_{12} & v_{415}=a_1+b_2+c_{12} & v_{446}=a_2+b_5+c_{12} \\
v_{323}=a_2+b_1+c_{13} & v_{354}=a_0+b_4+c_{13} & v_{385}=a_1+b_0+c_{13} & v_{416}=a_2+b_3+c_{13} & v_{447}=a_0+b_6+c_{13} \\
v_{324}=a_0+b_2+c_{14} & v_{355}=a_1+b_5+c_{14} & v_{386}=a_2+b_1+c_{14} & v_{417}=a_0+b_4+c_{14} & v_{448}=a_1+b_0+c_{14} \\
v_{325}=a_1+b_3+c_{15} & v_{356}=a_2+b_6+c_{15} & v_{387}=a_0+b_2+c_{15} & v_{418}=a_1+b_5+c_{15} & v_{449}=a_2+b_1+c_{15} \\
v_{326}=a_2+b_4+c_{16} & v_{367}=a_0+b_0+c_{16} & v_{388}=a_1+b_3+c_{16} & v_{419}=a_2+b_6+c_{16} & v_{450}=a_0+b_2+c_{16} \\
v_{327}=a_0+b_5+c_{17} & v_{358}=a_1+b_1+c_{17} & v_{389}=a_2+b_4+c_{17} & v_{420}=a_0+b_0+c_{17} & v_{451}=a_1+b_3+c_{17} \\
v_{328}=a_1+b_6+c_{18} & v_{359}=a_2+b_2+c_{18} & v_{390}=a_0+b_5+c_{18} & v_{421}=a_1+b_1+c_{18} & v_{452}=a_2+b_4+c_{18} \\
v_{329}=a_2+b_0+c_{19} & v_{360}=a_0+b_3+c_{19} & v_{391}=a_1+b_6+c_{19} & v_{422}=a_2+b_2+c_{19} & v_{453}=a_0+b_5+c_{19} \\
v_{330}=a_0+b_1+c_{20} & v_{361}=a_1+b_4+c_{20} & v_{392}=a_2+b_0+c_{20} & v_{423}=a_0+b_3+c_{20} & v_{454}=a_1+b_6+c_{20} \\
v_{331}=a_1+b_2+c_{21} & v_{362}=a_2+b_6+c_{21} & v_{393}=a_0+b_1+c_{21} & v_{424}=a_1+b_4+c_{21} & v_{455}=a_2+b_0+c_{21} \\
v_{332}=a_2+b_3+c_{22} & v_{363}=a_0+b_6+c_{22} & v_{394}=a_1+b_2+c_{22} & v_{425}=a_2+b_5+c_{22} & v_{456}=a_0+b_1+c_{22} \\
v_{333}=a_0+b_4+c_{23} & v_{364}=a_1+b_0+c_{23} & v_{395}=a_2+b_3+c_{23} & v_{426}=a_0+b_6+c_{23} & v_{457}=a_1+b_2+c_{23} \\
v_{334}=a_1+b_5+c_{24} & v_{365}=a_2+b_1+c_{24} & v_{396}=a_0+b_4+c_{24} & v_{427}=a_1+b_0+c_{24} & v_{458}=a_2+b_3+c_{24} \\
v_{335}=a_2+b_6+c_{25} & v_{366}=a_0+b_2+c_{25} & v_{397}=a_1+b_5+c_{25} & v_{428}=a_2+b_1+c_{25} & v_{459}=a_0+b_4+c_{25} \\
v_{336}=a_0+b_0+c_{26} & v_{367}=a_1+b_3+c_{26} & v_{398}=a_2+b_6+c_{26} & v_{429}=a_0+b_2+c_{26} & v_{460}=a_1+b_5+c_{26} \\
v_{337}=a_1+b_1+c_{27} & v_{368}=a_2+b_4+c_{27} & v_{399}=a_0+b_0+c_{27} & v_{430}=a_1+b_3+c_{27} & v_{461}=a_2+b_6+c_{27} \\
v_{338}=a_2+b_2+c_{28} & v_{369}=a_0+b_5+c_{28} & v_{400}=a_1+b_1+c_{28} & v_{431}=a_2+b_4+c_{28} & v_{462}=a_0+b_0+c_{28} \\
v_{339}=a_0+b_3+c_{29} & v_{370}=a_1+b_6+c_{29} & v_{401}=a_2+b_2+c_{29} & v_{432}=a_0+b_5+c_{29} & v_{463}=a_1+b_1+c_{29} \\
v_{340}=a_1+b_4+c_{30} & v_{371}=a_2+b_0+c_{30} & v_{402}=a_0+b_3+c_{30} & v_{433}=a_1+b_6+c_{30} & v_{464}=a_2+b_2+c_{30}
\end{array}
\qquad \text{(A1)}
$$

$v_{465}=a_0+b_3+c_0$ $v_{496}=a_1+b_6+c_0$ $v_{527}=a_2+b_2+c_0$ $v_{558}=a_0+b_5+c_0$ $v_{589}=a_1+b_1+c_0$

$v_{466}=a_1+b_4+c_1$ $v_{497}=a_2+b_0+c_1$ $v_{528}=a_0+b_3+c_1$ $v_{559}=a_1+b_6+c_1$ $v_{590}=a_2+b_2+c_1$

$v_{467}=a_2+b_5+c_2$ $v_{498}=a_0+b_1+c_2$ $v_{529}=a_1+b_4+c_2$ $v_{560}=a_2+b_0+c_2$ $v_{591}=a_0+b_3+c_2$

$v_{468}=a_0+b_6+c_3$ $v_{499}=a_1+b_2+c_3$ $v_{530}=a_2+b_5+c_3$ $v_{561}=a_0+b_1+c_3$ $v_{592}=a_1+b_4+c_3$

$v_{469}=a_1+b_0+c_4$ $v_{500}=a_2+b_3+c_4$ $v_{531}=a_0+b_6+c_4$ $v_{562}=a_1+b_2+c_4$ $v_{593}=a_2+b_5+c_4$

$v_{470}=a_2+b_1+c_5$ $v_{501}=a_0+b_4+c_5$ $v_{532}=a_1+b_0+c_5$ $v_{563}=a_2+b_3+c_5$ $v_{594}=a_0+b_6+c_5$

$v_{471}=a_0+b_2+c_6$ $v_{502}=a_1+b_5+c_6$ $v_{533}=a_2+b_1+c_6$ $v_{564}=a_0+b_4+c_6$ $v_{595}=a_1+b_0+c_6$

$v_{472}=a_1+b_3+c_7$ $v_{503}=a_2+b_6+c_7$ $v_{534}=a_0+b_2+c_7$ $v_{565}=a_1+b_5+c_7$ $v_{596}=a_2+b_1+c_7$

$v_{473}=a_2+b_4+c_8$ $v_{504}=a_0+b_0+c_8$ $v_{535}=a_1+b_3+c_8$ $v_{566}=a_2+b_6+c_8$ $v_{597}=a_0+b_2+c_8$

$v_{474}=a_0+b_5+c_9$ $v_{505}=a_1+b_1+c_9$ $v_{536}=a_2+b_4+c_9$ $v_{567}=a_0+b_0+c_9$ $v_{598}=a_1+b_3+c_9$

$v_{475}=a_1+b_6+c_{10}$ $v_{506}=a_2+b_2+c_{10}$ $v_{537}=a_0+b_5+c_{10}$ $v_{568}=a_1+b_1+c_{10}$ $v_{599}=a_2+b_4+c_{10}$

$v_{476}=a_2+b_0+c_{11}$ $v_{507}=a_0+b_3+c_{11}$ $v_{538}=a_1+b_6+c_{11}$ $v_{569}=a_2+b_2+c_{11}$ $v_{600}=a_0+b_5+c_{11}$

$v_{477}=a_0+b_1+c_{12}$ $v_{508}=a_1+b_4+c_{12}$ $v_{539}=a_2+b_0+c_{12}$ $v_{570}=a_0+b_3+c_{12}$ $v_{601}=a_1+b_6+c_{12}$

$v_{478}=a_1+b_2+c_{13}$ $v_{509}=a_2+b_5+c_{13}$ $v_{540}=a_0+b_1+c_{13}$ $v_{571}=a_1+b_4+c_{13}$ $v_{602}=a_2+b_0+c_{13}$

$v_{479}=a_2+b_3+c_{14}$ $v_{510}=a_0+b_6+c_{14}$ $v_{541}=a_1+b_2+c_{14}$ $v_{572}=a_2+b_5+c_{14}$ $v_{603}=a_0+b_1+c_{14}$

$v_{480}=a_0+b_4+c_{15}$ $v_{511}=a_1+b_0+c_{15}$ $v_{542}=a_2+b_3+c_{15}$ $v_{573}=a_0+b_6+c_{15}$ $v_{604}=a_1+b_2+c_{15}$

$v_{481}=a_1+b_5+c_{16}$ $v_{512}=a_2+b_1+c_{16}$ $v_{543}=a_0+b_4+c_{16}$ $v_{574}=a_1+b_0+c_{16}$ $v_{605}=a_2+b_3+c_{16}$ (A1)

$v_{482}=a_2+b_6+c_{17}$ $v_{513}=a_0+b_2+c_{17}$ $v_{544}=a_1+b_5+c_{17}$ $v_{575}=a_2+b_1+c_{17}$ $v_{606}=a_0+b_4+c_{17}$

$v_{483}=a_0+b_0+c_{18}$ $v_{514}=a_1+b_3+c_{18}$ $v_{545}=a_2+b_6+c_{18}$ $v_{576}=a_0+b_2+c_{18}$ $v_{607}=a_1+b_5+c_{18}$

$v_{484}=a_1+b_1+c_{19}$ $v_{515}=a_2+b_4+c_{19}$ $v_{546}=a_0+b_0+c_{19}$ $v_{577}=a_1+b_3+c_{19}$ $v_{608}=a_2+b_6+c_{19}$

$v_{485}=a_2+b_2+c_{20}$ $v_{516}=a_0+b_5+c_{20}$ $v_{547}=a_1+b_1+c_{20}$ $v_{578}=a_2+b_4+c_{20}$ $v_{609}=a_0+b_0+c_{20}$

$v_{486}=a_0+b_3+c_{21}$ $v_{517}=a_1+b_6+c_{21}$ $v_{548}=a_2+b_2+c_{21}$ $v_{579}=a_0+b_5+c_{21}$ $v_{610}=a_1+b_1+c_{21}$

$v_{487}=a_1+b_4+c_{22}$ $v_{518}=a_2+b_0+c_{22}$ $v_{549}=a_0+b_3+c_{22}$ $v_{580}=a_1+b_6+c_{22}$ $v_{611}=a_2+b_2+c_{22}$

$v_{488}=a_2+b_5+c_{23}$ $v_{519}=a_0+b_1+c_{23}$ $v_{550}=a_1+b_4+c_{23}$ $v_{581}=a_2+b_0+c_{23}$ $v_{612}=a_0+b_3+c_{23}$

$v_{489}=a_0+b_6+c_{24}$ $v_{520}=a_1+b_2+c_{24}$ $v_{551}=a_2+b_5+c_{24}$ $v_{582}=a_0+b_1+c_{24}$ $v_{613}=a_1+b_4+c_{24}$

$v_{490}=a_1+b_0+c_{25}$ $v_{521}=a_2+b_3+c_{25}$ $v_{552}=a_0+b_6+c_{25}$ $v_{583}=a_1+b_2+c_{25}$ $v_{614}=a_2+b_5+c_{25}$

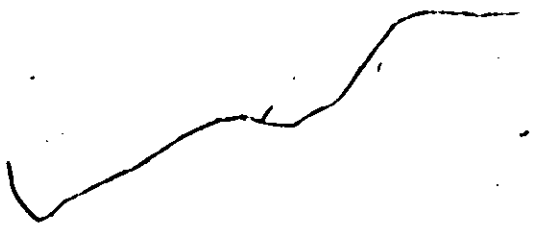
$v_{491}=a_2+b_1+c_{26}$ $v_{522}=a_0+b_4+c_{26}$ $v_{553}=a_1+b_0+c_{26}$ $v_{584}=a_2+b_3+c_{26}$ $v_{615}=a_0+b_6+c_{26}$

$v_{492}=a_0+b_2+c_{27}$ $v_{523}=a_1+b_5+c_{27}$ $v_{554}=a_2+b_1+c_{27}$ $v_{585}=a_0+b_4+c_{27}$ $v_{616}=a_1+b_0+c_{27}$

$v_{493}=a_1+b_3+c_{28}$ $v_{524}=a_2+b_6+c_{28}$ $v_{555}=a_0+b_2+c_{28}$ $v_{586}=a_1+b_5+c_{28}$ $v_{617}=a_2+b_1+c_{28}$

$v_{494}=a_2+b_4+c_{29}$ $v_{525}=a_0+b_0+c_{29}$ $v_{556}=a_1+b_3+c_{29}$ $v_{587}=a_2+b_6+c_{29}$ $v_{618}=a_0+b_2+c_{29}$

$v_{495}=a_0+b_5+c_{30}$ $v_{526}=a_1+b_1+c_{30}$ $v_{557}=a_2+b_4+c_{30}$ $v_{588}=a_0+b_0+c_{30}$ $v_{619}=a_1+b_3+c_{30}$



$$
\begin{aligned}
v_{620} &= a_2 + b_4 + c_0 \\
v_{621} &= a_0 + b_5 + c_1 \\
v_{622} &= a_1 + b_6 + c_2 \\
v_{623} &= a_2 + b_0 + c_3 \\
v_{624} &= a_0 + b_1 + c_4 \\
v_{625} &= a_1 + b_2 + c_5 \\
v_{626} &= a_2 + b_3 + c_6 \\
v_{627} &= a_0 + b_4 + c_7 \\
v_{628} &= a_1 + b_5 + c_8 \\
v_{629} &= a_2 + b_6 + c_9 \\
v_{630} &= a_0 + b_0 + c_{10} \\
v_{631} &= a_1 + b_1 + c_{11} \\
v_{632} &= a_2 + b_2 + c_{12} \\
v_{633} &= a_0 + b_3 + c_{13} \\
v_{634} &= a_1 + b_4 + c_{14} \\
v_{635} &= a_2 + b_5 + c_{15} \\
v_{636} &= a_0 + b_6 + c_{16} \qquad \text{(A1)} \\
v_{637} &= a_1 + b_0 + c_{17} \\
v_{638} &= a_2 + b_1 + c_{18} \\
v_{639} &= a_0 + b_2 + c_{19} \\
v_{640} &= a_1 + b_3 + c_{20} \\
v_{641} &= a_2 + b_4 + c_{21} \\
v_{642} &= a_0 + b_5 + c_{22} \\
v_{643} &= a_1 + b_6 + c_{23} \\
v_{644} &= a_2 + b_0 + c_{24} \\
v_{645} &= a_0 + b_1 + c_{25} \\
v_{646} &= a_1 + b_2 + c_{26} \\
v_{647} &= a_2 + b_3 + c_{27} \\
v_{648} &= a_0 + b_4 + c_{28} \\
v_{649} &= a_1 + b_5 + c_{29} \\
v_{650} &= a_2 + b_6 + c_{30}
\end{aligned}
$$

Using Equation A1, we can form 315 orthogonal check sums on c_0 as shown in Equation A2

$$
\begin{array}{l}
v_2 + v_{317} = v_5 + v_{215} = v_{24} + v_{150} = v_4 + v_{634} = v_{10} + v_{430} = v_{17} + v_{269} = v_8 + v_{617} = v_{20} + v_{209} = v_3 + v_{807} = c_0 \\
v_{33} + v_{348} = v_{36} + v_{246} = v_{55} + v_{181} = v_{35} + v_{14} = v_{41} + v_{461} = v_{48} + v_{300} = v_{39} + v_{648} = v_{51} + v_{240} = v_{34} + v_{538} = c_0 \\
v_{64} + v_{379} = v_{67} + v_{277} = v_{86} + v_{212} = v_{66} + v_{45} = v_{72} + v_{492} = v_{79} + v_{331} = v_{70} + v_{28} = v_{82} + v_{271} = v_{65} + v_{569} = c_0 \\
v_{95} + v_{410} = v_{98} + v_{308} = v_{117} + v_{243} = v_{97} + v_{76} = v_{103} + v_{523} = v_{110} + v_{362} = v_{101} + v_{59} = v_{113} + v_{302} = v_{96} + v_{600} = c_0 \\
v_{126} + v_{441} = v_{129} + v_{339} = v_{148} + v_{274} = v_{128} + v_{107} = v_{134} + v_{554} = v_{141} + v_{393} = v_{132} + v_{90} = v_{144} + v_{313} = v_{127} + v_{631} = c_0 \\
v_{157} + v_{972} = v_{160} + v_{370} = v_{179} + v_{305} = v_{159} + v_{138} = v_{165} + v_{885} = v_{172} + v_{424} = v_{163} + v_{121} = v_{175} + v_{364} = v_{158} + v_{11} = c_0 \\
v_{188} + v_{503} = v_{191} + v_{401} = v_{210} + v_{336} = v_{190} + v_{169} = v_{196} + v_{616} = v_{203} + v_{455} = v_{194} + v_{152} = v_{206} + v_{395} = v_{189} + v_{42} = c_0 \\
v_{219} + v_{534} = v_{222} + v_{432} = v_{241} + v_{367} = v_{221} + v_{200} = v_{227} + v_{647} = v_{234} + v_{486} = v_{225} + v_{183} = v_{237} + v_{426} = v_{220} + v_{73} = c_0 \\
v_{250} + v_{565} = v_{253} + v_{463} = v_{272} + v_{398} = v_{252} + v_{231} = v_{258} + v_{27} = v_{265} + v_{517} = v_{286} + v_{214} = v_{268} + v_{457} = v_{251} + v_{104} = c_0 \\
v_{281} + v_{596} = v_{284} + v_{494} = v_{303} + v_{429} = v_{283} + v_{262} = v_{289} + v_{58} = v_{296} + v_{548} = v_{287} + v_{245} = v_{299} + v_{488} = v_{282} + v_{135} = c_0 \\
v_{312} + v_{627} = v_{315} + v_{525} = v_{334} + v_{460} = v_{314} + v_{293} = v_{320} + v_{89} = v_{327} + v_{579} = v_{318} + v_{276} = v_{330} + v_{519} = v_{313} + v_{166} = c_0 \\
v_{343} + v_7 = v_{346} + v_{556} = v_{365} + v_{491} = v_{345} + v_{324} = v_{351} + v_{120} = v_{358} + v_{610} = v_{349} + v_{307} = v_{361} + v_{550} = v_{344} + v_{197} = c_0 \\
v_{374} + v_{38} = v_{377} + v_{587} = v_{396} + v_{522} = v_{376} + v_{353} = v_{382} + v_{151} = v_{389} + v_{641} = v_{380} + v_{338} = v_{392} + v_{581} = v_{375} + v_{228} = c_0 \\
v_{405} + v_{69} = v_{408} + v_{618} = v_{427} + v_{533} = v_{407} + v_{386} = v_{413} + v_{182} = v_{420} + v_{21} = v_{411} + v_{369} = v_{423} + v_{612} = v_{406} + v_{289} = c_0 \\
v_{435} + v_{100} = v_{439} + v_{649} = v_{458} + v_{584} = v_{438} + v_{417} = v_{444} + v_{213} = v_{451} + v_{52} = v_{442} + v_{400} = v_{454} + v_{643} = v_{437} + v_{290} = c_0 \\
v_{467} + v_{131} = v_{470} + v_{29} = v_{489} + v_{615} = v_{469} + v_{448} = v_{475} + v_{244} = v_{482} + v_{83} = v_{473} + v_{431} = v_{485} + v_{23} = v_{468} + v_{321} = c_0 \\
v_{498} + v_{162} = v_{501} + v_{60} = v_{520} + v_{646} = v_{500} + v_{479} = v_{506} + v_{275} = v_{513} + v_{114} = v_{504} + v_{462} = v_{516} + v_{54} = v_{499} + v_{352} = c_0 \\
v_{529} + v_{193} = v_{532} + v_{91} = v_{551} + v_{26} = v_{531} + v_{510} = v_{537} + v_{306} = v_{544} + v_{145} = v_{535} + v_{493} = v_{547} + v_{85} = v_{530} + v_{383} = c_0 \\
v_{560} + v_{224} = v_{563} + v_{122} = v_{582} + v_{57} = v_{502} + v_{541} = v_{568} + v_{337} = v_{575} + v_{176} = v_{566} + v_{524} = v_{578} + v_{116} = v_{561} + v_{414} = c_0 \\
v_{591} + v_{255} = v_{594} + v_{153} = v_{613} + v_{88} = v_{593} + v_{572} = v_{599} + v_{368} = v_{606} + v_{207} = v_{597} + v_{555} = v_{609} + v_{147} = v_{592} + v_{445} = c_0 \\
v_{622} + v_{286} = v_{625} + v_{184} = v_{644} + v_{119} = v_{624} + v_{603} = v_{630} + v_{399} = v_{637} + v_{238} = v_{628} + v_{586} = v_{640} + v_{178} = v_{623} + v_{476} = c_0
\end{array}
\qquad \text{(A2)}
$$

$$
\begin{array}{l}
v_{16} + v_{583} = v_{9} + v_{387} = v_{6} + v_{363} = v_{1} + v_{484} = v_{18} + v_{123} = v_{12} + v_{75} = c_0 \\
v_{47} + v_{614} = v_{40} + v_{418} = v_{37} + v_{394} = v_{32} + v_{515} = v_{49} + v_{154} = v_{43} + v_{106} = c_0 \\
v_{78} + v_{645} = v_{71} + v_{449} = v_{68} + v_{425} = v_{63} + v_{546} = v_{80} + v_{185} = v_{74} + v_{137} = c_0 \\
v_{109} + v_{25} = v_{102} + v_{480} = v_{99} + v_{456} = v_{94} + v_{577} = v_{111} + v_{216} = v_{105} + v_{168} = c_0 \\
v_{140} + v_{56} = v_{133} + v_{511} = v_{130} + v_{487} = v_{125} + v_{608} = v_{142} + v_{247} = v_{136} + v_{199} = c_0 \\
v_{171} + v_{87} = v_{164} + v_{542} = v_{161} + v_{518} = v_{156} + v_{639} = v_{173} + v_{278} = v_{167} + v_{230} = c_0 \\
v_{202} + v_{118} = v_{195} + v_{573} = v_{192} + v_{549} = v_{187} + v_{19} = v_{204} + v_{309} = v_{198} + v_{261} = c_0 \\
v_{233} + v_{149} = v_{226} + v_{604} = v_{223} + v_{580} = v_{218} + v_{50} = v_{235} + v_{340} = v_{229} + v_{292} = c_0 \\
v_{264} + v_{180} = v_{257} + v_{635} = v_{254} + v_{611} = v_{249} + v_{81} = v_{266} + v_{371} = v_{260} + v_{323} = c_0 \\
v_{295} + v_{211} = v_{288} + v_{15} = v_{285} + v_{642} = v_{280} + v_{112} = v_{297} + v_{402} = v_{291} + v_{354} = c_0 \\
v_{326} + v_{242} = v_{319} + v_{46} = v_{316} + v_{22} = v_{311} + v_{143} = v_{328} + v_{433} = v_{322} + v_{385} = c_0 \\
v_{357} + v_{273} = v_{350} + v_{77} = v_{347} + v_{53} = v_{342} + v_{174} = v_{359} + v_{464} = v_{353} + v_{416} = c_0 \\
v_{388} + v_{304} = v_{381} + v_{108} = v_{378} + v_{84} = v_{373} + v_{205} = v_{390} + v_{495} = v_{384} + v_{447} = c_0 \\
v_{419} + v_{335} = v_{412} + v_{139} = v_{409} + v_{115} = v_{404} + v_{236} = v_{421} + v_{526} = v_{415} + v_{478} = c_0 \\
v_{450} + v_{366} = v_{443} + v_{170} = v_{440} + v_{146} = v_{435} + v_{267} = v_{452} + v_{557} = v_{446} + v_{509} = c_0 \\
v_{481} + v_{397} = v_{474} + v_{471} = v_{471} + v_{177} = v_{466} + v_{298} = v_{483} + v_{588} = v_{477} + v_{540} = c_0 \\
v_{512} + v_{428} = v_{505} + v_{232} = v_{502} + v_{208} = v_{497} + v_{329} = v_{514} + v_{619} = v_{508} + v_{571} = c_0 \\
v_{543} + v_{459} = v_{536} + v_{263} = v_{533} + v_{239} = v_{528} + v_{360} = v_{545} + v_{650} = v_{539} + v_{602} = c_0 \\
v_{574} + v_{490} = v_{567} + v_{294} = v_{564} + v_{270} = v_{559} + v_{391} = v_{576} + v_{30} = v_{570} + v_{633} = c_0 \\
v_{605} + v_{521} = v_{598} + v_{325} = v_{595} + v_{301} = v_{580} + v_{422} = v_{607} + v_{61} = v_{601} + v_{13} = c_0 \\
v_{636} + v_{552} = v_{629} + v_{356} = v_{626} + v_{332} = v_{621} + v_{453} = v_{638} + v_{92} = v_{632} + v_{44} = c_0
\end{array}
\qquad \text{(A2)}
$$

Equation A3 shows all 310 orthogonal check sums on b_0

$$
\begin{aligned}
v'_2 + v'_{17} &= v'_4 + v'_{13} = v'_1 + v'_{19} = b_0 \\
v'_9 + v'_{24} &= v'_{11} + v'_{20} = v'_{20} + v'_8 = b_0 \\
v'_{16} + v'_{31} &= v'_{18} + v'_{27} = v'_{15} + v'_{33} = b_0 \\
v'_{23} + v'_{38} &= v'_{35} + v'_{34} = v'_{22} + v'_{40} = b_0 \\
v'_{30} + v'_{45} &= v'_{32} + v'_{41} = v'_{29} + v'_{47} = b_0 \\
v'_{37} + v'_{52} &= v'_{39} + v'_{48} = v'_{36} + v'_{54} = b_0 \\
v'_{44} + v'_{59} &= v'_{46} + v'_{55} = v'_{43} + v'_{61} = b_0 \\
v'_{51} + v'_{66} &= v'_{53} + v'_{62} = v'_{50} + v'_{68} = b_0 \\
v'_{58} + v'_{73} &= v'_{60} + v'_{69} = v'_{57} + v'_{75} = b_0 \\
v'_{65} + v'_{80} &= v'_{67} + v'_{76} = v'_{64} + v'_{82} = b_0 \\
v'_{72} + v'_{87} &= v'_{74} + v'_{83} = v'_{71} + v'_{89} = b_0 \\
v'_{79} + v'_{94} &= v'_{81} + v'_{90} = v'_{78} + v'_{96} = b_0 \\
v'_{86} + v'_{101} &= v'_{88} + v'_{97} = v'_{85} + v'_{103} = b_0 \\
v'_{93} + v'_{108} &= v'_{95} + v'_{104} = v'_{92} + v'_{110} = b_0 \\
v'_{100} + v'_{115} &= v'_{102} + v'_{111} = v'_{99} + v'_{117} = b_0 \\
v'_{107} + v'_{122} &= v'_{109} + v'_{118} = v'_{106} + v'_{124} = b_0 \\
v'_{114} + v'_{129} &= v'_{116} + v'_{125} = v'_{113} + v'_{131} = b_0 \\
v'_{121} + v'_{136} &= v'_{123} + v'_{132} = v'_{120} + v'_{138} = b_0 \\
v'_{128} + v'_{130} &= v'_{130} + v'_{139} = v'_{127} + v'_{145} = b_0 \\
v'_{135} + v'_{150} &= v'_{137} + v'_{146} = v'_{134} + v'_{152} = b_0 \\
v'_{142} + v'_{157} &= v'_{144} + v'_{153} = v'_{141} + v'_{159} = b_0 \\
v'_{149} + v'_{164} &= v'_{151} + v'_{160} = v'_{148} + v'_{166} = b_0 \\
v'_{156} + v'_{171} &= v'_{158} + v'_{167} = v'_{155} + v'_{173} = b_0 \\
v'_{163} + v'_{178} &= v'_{165} + v'_{174} = v'_{162} + v'_{180} = b_0 \\
v'_{170} + v'_{185} &= v'_{172} + v'_{181} = v'_{169} + v'_{187} = b_0 \\
v'_{177} + v'_{192} &= v'_{179} + v'_{188} = v'_{176} + v'_{194} = b_0 \\
v'_{184} + v'_{199} &= v'_{186} + v'_{195} = v'_{183} + v'_{201} = b_0 \\
v'_{191} + v'_{206} &= v'_{193} + v'_{202} = v'_{190} + v'_{208} = b_0 \\
v'_{198} + v'_{213} &= v'_{200} + v'_{209} = v'_{197} + v'_{215} = b_0 \\
v'_{205} + v'_{220} &= v'_{207} + v'_{216} = v'_{204} + v'_{222} = b_0 \\
v'_{212} + v'_{227} &= v'_{214} + v'_{223} = v'_{211} + v'_{229} = b_0
\end{aligned}
\qquad \text{(A3)}
$$

$$
\begin{aligned}
v'_{219} + v'_{234} &= v'_{221} + v'_{230} = v'_{218} + v'_{236} = b_0 \\
v'_{226} + v'_{241} &= v'_{228} + v'_{228} = v'_{225} + v'_{243} = b_0 \\
v'_{233} + v'_{248} &= v'_{235} + v'_{244} = v'_{232} + v'_{250} = b_0 \\
v'_{240} + v'_{255} &= v'_{242} + v'_{251} = v'_{239} + v'_{257} = b_0 \\
v'_{247} + v'_{262} &= v'_{249} + v'_{258} = v'_{246} + v'_{264} = b_0 \\
v'_{254} + v'_{269} &= v'_{256} + v'_{265} = v'_{253} + v'_{271} = b_0 \\
v'_{261} + v'_{276} &= v'_{263} + v'_{272} = v'_{260} + v'_{278} = b_0 \\
v'_{268} + v'_{283} &= v'_{270} + v'_{279} = v'_{267} + v'_{285} = b_0 \\
v'_{275} + v'_{290} &= v'_{277} + v'_{286} = v'_{274} + v'_{292} = b_0 \\
v'_{282} + v'_{297} &= v'_{284} + v'_{293} = v'_{281} + v'_{299} = b_0 \\
v'_{289} + v'_{304} &= v'_{291} + v'_{300} = v'_{288} + v'_{306} = b_0 \\
v'_{296} + v'_{311} &= v'_{298} + v'_{307} = v'_{295} + v'_{313} = b_0 \\
v'_{303} + v'_{318} &= v'_{305} + v'_{314} = v'_{302} + v'_{320} = b_0 \\
v'_{310} + v'_{325} &= v'_{312} + v'_{321} = v'_{309} + v'_{327} = b_0 \\
v'_{317} + v'_{332} &= v'_{319} + v'_{328} = v'_{316} + v'_{334} = b_0 \\
v'_{324} + v'_{339} &= v'_{326} + v'_{335} = v'_{323} + v'_{341} = b_0 \\
v'_{331} + v'_{346} &= v'_{333} + v'_{342} = v'_{330} + v'_{348} = b_0 \\
v'_{338} + v'_{353} &= v'_{340} + v'_{349} = v'_{337} + v'_{353} = b_0 \\
v'_{345} + v'_{360} &= v'_{347} + v'_{356} = v'_{344} + v'_{362} = b_0 \\
v'_{352} + v'_{367} &= v'_{354} + v'_{363} = v'_{351} + v'_{369} = b_0 \\
v'_{359} + v'_{374} &= v'_{361} + v'_{370} = v'_{358} + v'_{376} = b_0 \\
v'_{366} + v'_{381} &= v'_{368} + v'_{377} = v'_{365} + v'_{383} = b_0 \\
v'_{376} + v'_{388} &= v'_{378} + v'_{384} = v'_{372} + v'_{390} = b_0 \\
v'_{380} + v'_{395} &= v'_{382} + v'_{391} = v'_{379} + v'_{397} = b_0 \\
v'_{387} + v'_{402} &= v'_{389} + v'_{398} = v'_{386} + v'_{404} = b_0 \\
v'_{394} + v'_{409} &= v'_{396} + v'_{405} = v'_{393} + v'_{411} = b_0 \\
v'_{401} + v'_{416} &= v'_{403} + v'_{412} = v'_{400} + v'_{418} = b_0 \\
v'_{408} + v'_{423} &= v'_{410} + v'_{419} = v'_{407} + v'_{425} = b_0 \\
v'_{415} + v'_{430} &= v'_{417} + v'_{426} = v'_{414} + v'_{432} = b_0 \\
v'_{422} + v'_{437} &= v'_{424} + v'_{433} = v'_{421} + v'_{439} = b_0 \\
v'_{429} + v'_{444} &= v'_{431} + v'_{440} = v'_{428} + v'_{446} = b_0 \\
v'_{436} + v'_{451} &= v'_{438} + v'_{447} = v'_{435} + v'_{453} = b_0 \\
v'_{443} + v'_{458} &= v'_{445} + v'_{454} = v'_{442} + v'_{460} = b_0
\end{aligned}
\tag{A3}
$$

$$
\begin{aligned}
v'_{450} + v'_{465} &= v'_{452} + v'_{461} = v'_{449} + v'_{467} = b_0 \\
v'_{457} + v'_{472} &= v'_{459} + v'_{468} = v'_{456} + v'_{474} = b_0 \\
v'_{464} + v'_{479} &= v'_{466} + v'_{475} = v'_{463} + v'_{481} = b_0 \\
v'_{471} + v'_{486} &= v'_{473} + v'_{482} = v'_{470} + v'_{488} = b_0 \\
v'_{478} + v'_{493} &= v'_{480} + v'_{489} = v'_{477} + v'_{495} = b_0 \\
v'_{485} + v'_{500} &= v'_{487} + v'_{496} = v'_{484} + v'_{502} = b_0 \\
v'_{492} + v'_{507} &= v'_{494} + v'_{503} = v'_{491} + v'_{509} = b_0 \\
v'_{499} + v'_{514} &= v'_{501} + v'_{510} = v'_{498} + v'_{516} = b_0 \\
v'_{506} + v'_{521} &= v'_{508} + v'_{517} = v'_{505} + v'_{523} = b_0 \\
v'_{513} + v'_{528} &= v'_{514} + v'_{524} = v'_{512} + v'_{530} = b_0 \\
v'_{520} + v'_{535} &= v'_{522} + v'_{531} = v'_{519} + v'_{537} = b_0 \\
v'_{527} + v'_{542} &= v'_{529} + v'_{538} = v'_{526} + v'_{544} = b_0 \\
v'_{534} + v'_{549} &= v'_{536} + v'_{545} = v'_{533} + v'_{551} = b_0 \\
v'_{541} + v'_{556} &= v'_{543} + v'_{552} = v'_{540} + v'_{558} = b_0 \\
v'_{548} + v'_{563} &= v'_{550} + v'_{559} = v'_{547} + v'_{565} = b_0 \\
v'_{555} + v'_{570} &= v'_{557} + v'_{566} = v'_{554} + v'_{572} = b_0 \\
v'_{562} + v'_{577} &= v'_{564} + v'_{573} = v'_{561} + v'_{579} = b_0 \\
v'_{569} + v'_{584} &= v'_{571} + v'_{680} = v'_{568} + v'_{586} = b_0 \\
v'_{576} + v'_{591} &= v'_{578} + v'_{687} = v'_{575} + v'_{595} = b_0 \\
v'_{583} + v'_{598} &= v'_{585} + v'_{594} = v'_{582} + v'_{600} = b_0 \\
v'_{590} + v'_{605} &= v'_{592} + v'_{601} = v'_{589} + v'_{609} = b_0 \\
v'_{597} + v'_{612} &= v'_{599} + v'_{608} = v'_{596} + v'_{614} = b_0 \\
v'_{604} + v'_{619} &= v'_{606} + v'_{615} = v'_{603} + v'_{621} = b_0 \\
v'_{611} + v'_{626} &= v'_{613} + v'_{622} = v'_{610} + v'_{628} = b_0 \\
v'_{618} + v'_{633} &= v'_{620} + v'_{629} = v'_{617} + v'_{635} = b_0 \\
v'_{625} + v'_{640} &= v'_{627} + v'_{636} = v'_{624} + v'_{642} = b_0 \\
v'_{632} + v'_{647} &= v'_{634} + v'_{643} = v'_{631} + v'_{649} = b_0 \\
v'_{639} + v'_{3} &= v'_{641} + v'_{650} = v'_{638} + v'_{5} = b_0 \\
v'_{646} + v'_{10} &= v'_{648} + v'_{657} = v'_{645} + v'_{12} = b_0
\end{aligned}
\qquad \text{(A3)}
$$

$$
\begin{aligned}
v'_0 + v'_7 + v'_{14} &= v'_{336} + v'_{343} + v'_{350} = b_0 \\
v'_{21} + v'_{28} + v'_{35} &= v'_{357} + v'_{364} + v'_{371} = b_0 \\
v'_{42} + v'_{49} + v'_{56} &= v'_{378} + v'_{385} + v'_{392} = b_0 \\
v'_{63} + v'_{70} + v'_{77} &= v'_{399} + v'_{406} + v'_{413} = b_0 \\
v'_{84} + v'_{91} + v'_{98} &= v'_{420} + v'_{427} + v'_{434} = b_0 \\
v'_{105} + v'_{112} + v'_{119} &= v'_{441} + v'_{448} + v'_{455} = b_0 \\
v'_{126} + v'_{133} + v'_{140} &= v'_{462} + v'_{469} + v'_{476} = b_0 \\
v'_{147} + v'_{154} + v'_{161} &= v'_{483} + v'_{490} + v'_{497} = b_0 \\
v'_{168} + v'_{175} + v'_{182} &= v'_{504} + v'_{511} + v'_{518} = b_0 \\
v'_{189} + v'_{196} + v'_{203} &= v'_{525} + v'_{532} + v'_{539} = b_0 \\
v'_{210} + v'_{217} + v'_{224} &= v'_{546} + v'_{553} + v'_{560} = b_0 \\
v'_{231} + v'_{238} + v'_{245} &= v'_{567} + v'_{574} + v'_{581} = b_0 \\
v'_{252} + v'_{259} + v'_{266} &= v'_{588} + v'_{595} + v'_{602} = b_0 \\
v'_{273} + v'_{280} + v'_{287} &= v'_{609} + v'_{616} + v'_{623} = b_0 \\
v'_{294} + v'_{301} + v'_{308} &= v'_{630} + v'_{637} + v'_{644} = b_0 \\
v'_{315} + v'_{322} + v'_{329} &= b_0
\end{aligned}
\qquad \text{(A3)}
$$

Equation A4 shows all 434 orthogonal check sums on a_0

$$
\begin{array}{llllllll}
v''_1 + v''_2 = v''_0 & = v''_{127} + v''_{128} = v''_{126} & = v''_{253} + v''_{254} = v''_{252} & = a_0 \\
v''_4 + v''_5 = v''_3 & = v''_{130} + v''_{131} = v''_{129} & = v''_{256} + v''_{257} = v''_{255} & = a_0 \\
v''_7 + v''_8 = v''_6 & = v''_{133} + v''_{134} = v''_{132} & = v''_{259} + v''_{260} = v''_{258} & = a_0 \\
v''_{10} + v''_{11} = v''_7 & = v''_{136} + v''_{137} = v''_{135} & = v''_{262} + v''_{263} = v''_{261} & = a_0 \\
v''_{13} + v''_{14} = v''_{12} & = v''_{139} + v''_{140} = v''_{138} & = v''_{265} + v''_{266} = v''_{264} & = a_0 \\
v''_{16} + v''_{17} = v''_{15} & = v''_{142} + v''_{143} = v''_{141} & = v''_{268} + v''_{269} = v''_{267} & = a_0 \\
v''_{19} + v''_{20} = v''_{18} & = v''_{145} + v''_{146} = v''_{144} & = v''_{271} + v''_{272} = v''_{270} & = a_0 \\
v''_{22} + v''_{23} = v''_{21} & = v''_{148} + v''_{149} = v''_{147} & = v''_{274} + v''_{275} = v''_{273} & = a_0 \\
v''_{25} + v''_{26} = v''_{24} & = v''_{151} + v''_{152} = v''_{150} & = v''_{277} + v''_{278} = v''_{276} & = a_0 \\
v''_{28} + v''_{29} = v''_{27} & = v''_{154} + v''_{155} = v''_{153} & = v''_{280} + v''_{281} = v''_{279} & = a_0 \\
v''_{31} + v''_{32} = v''_{30} & = v''_{157} + v''_{158} = v''_{156} & = v''_{283} + v''_{284} = v''_{282} & = a_0 \\
v''_{34} + v''_{35} = v''_{33} & = v''_{160} + v''_{161} = v''_{159} & = v''_{286} + v''_{287} = v''_{285} & = a_0 \\
v''_{37} + v''_{38} = v''_{36} & = v''_{163} + v''_{164} = v''_{162} & = v''_{289} + v''_{290} = v''_{288} & = a_0 \\
v''_{40} + v''_{41} = v''_{39} & = v''_{166} + v''_{167} = v''_{165} & = v''_{292} + v''_{293} = v''_{291} & = a_0 \\
v''_{43} + v''_{44} = v''_{42} & = v''_{169} + v''_{170} = v''_{168} & = v''_{295} + v''_{296} = v''_{294} & = a_0 \\
v''_{46} + v''_{47} = v''_{45} & = v''_{172} + v''_{173} = v''_{171} & = v''_{298} + v''_{299} = v''_{297} & = a_0 \\
v''_{49} + v''_{50} = v''_{48} & = v''_{175} + v''_{176} = v''_{174} & = v''_{301} + v''_{302} = v''_{300} & = a_0 \\
v''_{52} + v''_{53} = v''_{51} & = v''_{178} + v''_{179} = v''_{177} & = v''_{304} + v''_{305} = v''_{303} & = a_0 \\
v''_{55} + v''_{56} = v''_{54} & = v''_{181} + v''_{182} = v''_{180} & = v''_{307} + v''_{308} = v''_{306} & = a_0 \\
v''_{58} + v''_{59} = v''_{57} & = v''_{184} + v''_{185} = v''_{183} & = v''_{310} + v''_{311} = v''_{309} & = a_0 \\
v''_{61} + v''_{62} = v''_{60} & = v''_{187} + v''_{188} = v''_{186} & = v''_{313} + v''_{314} = v''_{312} & = a_0 \\
v''_{64} + v''_{65} = v''_{63} & = v''_{190} + v''_{191} = v''_{189} & = v''_{316} + v''_{317} = v''_{315} & = a_0 \\
v''_{67} + v''_{68} = v''_{66} & = v''_{193} + v''_{194} = v''_{192} & = v''_{319} + v''_{320} = v''_{318} & = a_0 \\
v''_{70} + v''_{71} = v''_{69} & = v''_{196} + v''_{197} = v''_{195} & = v''_{322} + v''_{323} = v''_{321} & = a_0 \\
v''_{73} + v''_{74} = v''_{72} & = v''_{199} + v''_{200} = v''_{198} & = v''_{325} + v''_{326} = v''_{324} & = a_0 \\
v''_{76} + v''_{77} = v''_{75} & = v''_{202} + v''_{203} = v''_{201} & = v''_{328} + v''_{329} = v''_{327} & = a_0 \\
v''_{79} + v''_{80} = v''_{78} & = v''_{205} + v''_{206} = v''_{204} & = v''_{331} + v''_{332} = v''_{330} & = a_0 \\
v''_{82} + v''_{83} = v''_{81} & = v''_{208} + v''_{209} = v''_{207} & = v''_{334} + v''_{335} = v''_{333} & = a_0 \\
v''_{85} + v''_{86} = v''_{84} & = v''_{211} + v''_{212} = v''_{210} & = v''_{337} + v''_{338} = v''_{336} & = a_0 \\
v''_{88} + v''_{89} = v''_{87} & = v''_{214} + v''_{213} = v''_{213} & = v''_{340} + v''_{341} = v''_{339} & = a_0 \\
v''_{91} + v''_{92} = v''_{90} & = v''_{217} + v''_{218} = v''_{216} & = v''_{343} + v''_{344} = v''_{342} & = a_0
\end{array}
\qquad \text{(A4)}
$$

$$
\begin{aligned}
v''_{94} + v''_{95} &= v''_{93} = v''_{220} + v''_{221} = v''_{219} = v''_{346} + v''_{347} = v''_{345} = a_0 \\
v''_{97} + v''_{98} &= v''_{96} = v''_{223} + v''_{224} = v''_{222} = v''_{349} + v''_{350} = v''_{348} = a_0 \\
v''_{100} + v''_{102} &= v''_{99} = v''_{226} + v''_{227} = v''_{225} = v''_{352} + v''_{353} = v''_{351} = a_0 \\
v''_{103} + v''_{105} &= v''_{102} = v''_{229} + v''_{230} = v''_{228} = v''_{355} + v''_{356} = v''_{354} = a_0 \\
v''_{106} + v''_{107} &= v''_{105} = v''_{232} + v''_{233} = v''_{231} = v''_{358} + v''_{359} = v''_{357} = a_0 \\
v''_{109} + v''_{110} &= v''_{108} = v''_{235} + v''_{236} = v''_{234} = v''_{361} + v''_{362} = v''_{360} = a_0 \\
v''_{112} + v''_{113} &= v''_{111} = v''_{238} + v''_{239} = v''_{237} = v''_{364} + v''_{365} = v''_{363} = a_0 \\
v''_{115} + v''_{116} &= v''_{114} = v''_{241} + v''_{242} = v''_{240} = v''_{367} + v''_{368} = v''_{366} = a_0 \\
v''_{118} + v''_{119} &= v''_{117} = v''_{244} + v''_{245} = v''_{243} = v''_{370} + v''_{371} = v''_{369} = a_0 \\
v''_{121} + v''_{122} &= v''_{120} = v''_{247} + v''_{248} = v''_{246} = v''_{373} + v''_{374} = v''_{372} = a_0 \\
v''_{124} + v''_{125} &= v''_{123} = v''_{250} + v''_{251} = v''_{249} = v''_{376} + v''_{377} = v''_{375} = a_0 \\
v''_{379} + v''_{380} &= v''_{378} = v''_{505} + v''_{506} = v''_{504} = v''_{631} + v''_{632} = v''_{630} = a_0 \\
v''_{382} + v''_{383} &= v''_{381} = v''_{508} + v''_{509} = v''_{507} = v''_{634} + v''_{635} = v''_{633} = a_0 \\
v''_{385} + v''_{386} &= v''_{384} = v''_{511} + v''_{512} = v''_{510} = v''_{637} + v''_{638} = v''_{636} = a_0 \\
v''_{388} + v''_{389} &= v''_{387} = v''_{514} + v''_{515} = v''_{513} = v''_{640} + v''_{641} = v''_{639} = a_0 \\
v''_{391} + v''_{392} &= v''_{390} = v''_{517} + v''_{518} = v''_{516} = v''_{643} + v''_{644} = v''_{642} = a_0 \\
v''_{394} + v''_{395} &= v''_{393} = v''_{520} + v''_{521} = v''_{519} = v''_{646} + v''_{647} = v''_{645} = a_0 \\
v''_{397} + v''_{398} &= v''_{396} = v''_{523} + v''_{524} = v''_{522} = v''_{649} + v''_{650} = v''_{648} = a_0 \\
v''_{400} + v''_{401} &= v''_{399} = v''_{526} + v''_{527} = v''_{525} = a_0 \\
v''_{403} + v''_{404} &= v''_{402} = v''_{529} + v''_{530} = v''_{528} = a_0 \\
v''_{406} + v''_{407} &= v''_{405} = v''_{532} + v''_{533} = v''_{531} = a_0 \\
v''_{409} + v''_{410} &= v''_{408} = v''_{535} + v''_{536} = v''_{534} = a_0 \\
v''_{412} + v''_{413} &= v''_{411} = v''_{538} + v''_{539} = v''_{537} = a_0 \\
v''_{415} + v''_{416} &= v''_{414} = v''_{541} + v''_{542} = v''_{540} = a_0 \\
v''_{418} + v''_{419} &= v''_{417} = v''_{544} + v''_{545} = v''_{543} = a_0 \\
v''_{421} + v''_{422} &= v''_{420} = v''_{547} + v''_{548} = v''_{546} = a_0 \\
v''_{424} + v''_{425} &= v''_{423} = v''_{550} + v''_{551} = v''_{549} = a_0 \\
v''_{427} + v''_{428} &= v''_{426} = v''_{553} + v''_{554} = v''_{552} = a_0 \\
v''_{430} + v''_{431} &= v''_{429} = v''_{556} + v''_{557} = v''_{555} = a_0 \\
v''_{433} + v''_{434} &= v''_{432} = v''_{559} + v''_{560} = v''_{558} = a_0 \\
v''_{436} + v''_{437} &= v''_{435} = v''_{562} + v''_{563} = v''_{561} = a_0 \\
v''_{439} + v''_{440} &= v''_{438} = v''_{565} + v''_{566} = v''_{564} = a_0
\end{aligned}
\qquad \text{(A4)}
$$

$$
\begin{aligned}
v''_{442} + v''_{443} &= v''_{441} = v''_{568} + v''_{569} = v''_{567} = a_0 \\
v''_{445} + v''_{446} &= v''_{444} = v''_{571} + v''_{572} = v''_{570} = a_0 \\
v''_{448} + v''_{449} &= v''_{447} = v''_{574} + v''_{575} = v''_{573} = a_0 \\
v''_{451} + v''_{452} &= v''_{450} = v''_{577} + v''_{578} = v''_{576} = a_0 \\
v''_{454} + v''_{455} &= v''_{453} = v''_{580} + v''_{581} = v''_{579} = a_0 \\
v''_{457} + v''_{458} &= v''_{456} = v''_{583} + v''_{584} = v''_{582} = a_0 \\
v''_{460} + v''_{461} &= v''_{459} = v''_{586} + v''_{587} = v''_{585} = a_0 \\
v''_{463} + v''_{464} &= v''_{462} = v''_{589} + v''_{590} = v''_{588} = a_0 \\
v''_{466} + v''_{467} &= v''_{465} = v''_{592} + v''_{593} = v''_{591} = a_0 \\
v''_{469} + v''_{470} &= v''_{468} = v''_{595} + v''_{596} = v''_{594} = a_0 \\
v''_{472} + v''_{473} &= v''_{471} = v''_{598} + v''_{599} = v''_{597} = a_0 \\
v''_{475} + v''_{476} &= v''_{474} = v''_{601} + v''_{602} = v''_{600} = a_0 \\
v''_{478} + v''_{479} &= v''_{477} = v''_{604} + v''_{605} = v''_{603} = a_0 \\
v''_{481} + v''_{482} &= v''_{480} = v''_{607} + v''_{608} = v''_{606} = a_0 \\
v''_{484} + v''_{485} &= v''_{483} = v''_{610} + v''_{611} = v''_{609} = a_0 \\
v''_{487} + v''_{488} &= v''_{486} = v''_{613} + v''_{614} = v''_{612} = a_0 \qquad \text{(A4)} \\
v''_{490} + v''_{491} &= v''_{489} = v''_{616} + v''_{617} = v''_{615} = a_0 \\
v''_{493} + v''_{494} &= v''_{492} = v''_{619} + v''_{620} = v''_{618} = a_0 \\
v''_{496} + v''_{497} &= v''_{495} = v''_{622} + v''_{623} = v''_{621} = a_0 \\
v''_{499} + v''_{500} &= v''_{498} = v''_{625} + v''_{626} = v''_{624} = a_0 \\
v''_{502} + v''_{503} &= v''_{501} = v''_{628} + v''_{629} = v''_{627} = a_0
\end{aligned}
$$

REFERENCES

1. C.E. Shannon, "Mathematical Theory of Communication" Bell System, Tech. Journal 27, 379; 623 (July and October 1948).

2. R.C. Dixon, "Spread Spectrum Systems", John Wilky and Sons, New York, 1976.

3. J.A. Kumar, "Direct Sequence Spread Spectrum Systems", Ph.D. Thesis, Southern Methodist University, 1978.

4. D.R. Anderson, "Periodic and Partial Correlation Properties of Sequences", TRW 1.c 7353 1.01, July 1969.

5. T.G. Birdsell and M.P. Ristenbat, "Introduction to Linear Shift-Register Generated Sequences", Tech. Report 90, University of Michigan Research Institute, October, 1958.

6. S.W. Golomb, "Shift Register Sequences", Holden Day, San Francisco, 1967.

7. W.W. Peterson, "Error Correcting Codes", the M.I.T. Press, Massachussets Institute of Technology, Cambridge, Mass., 1968.

8. S.W. Golomb, et al., "Digital Communications with Space Applications", Prentice Hall, Englewood Cliffs, New Jersey, 1964.

9. R.B. Ward, "Acquisition of Pseudonoise Signals by Sequential Estimation", IEEE Trans. in Communications Technology, Vol. Com 13, December 1965.

10. C.C. Kilgus, "Pseudonoise Acquisition Using Majority Logic Decoding", IEEE Transactions on Communications, June, 1973.

11. J.L. Massey, "Threshold Decoding", The M.I.T. Press, Massachusetts Institute of Technology, Cambridge, Mass., 1963.

12. C. Laferriere, "Error Correcting Capability of 1-step Majority-Logic-Decoding", M.A.Sc. thesis, University of Ottawa, Ottawa, Ontario, June 1977.

13. Shu Lin, "An Introduction to Error Correcting Codes", Prentice Hall, 1970.

14. E.J. Weldon, Jr., "Some Results on Majority-Logic-Decoding", Error Correcting Codes, Editor H. Mann., John Wiley and Sons, 1968.

15. L.D. Rudolph, "Threshold Decoding of Cyclic Codes", IEEE Transactions on Information Theory, Vol. IT-15, May 1969.

16. L.D. Rudolph, "A Class of Majority-Logic-Decodable Codes", IEEE Transactions on Information Theory, Vol. IT-13, April 1967.

17. S.G.S. Shiva, "A Class of Majority-Logic-Decodable Codes with a Minimum Distance of 4", Electronics Letters, Vol. 12, No. 22, October 1976.

18. S.G.S. Shiva and S.E. Tavares, "On Binary Majority-Logic-Decodable Codes", IEEE Trans. on Information Theory, Vol. IT-20, January 1974.

19. N.Q. Duc and L.V. Skatlebal, "Algorithms for Majority-Logic Check of Maximal Length Codes", Electronics Letters, November 1969.

20. S.G.S. Shiva, P.E. Allard and G. Sequin, "A Class of Composite Codes", To Appear in IEEE Transactions on Information Theory.

21. F.J. MacWilliams and N.J.A. Sloane, "The Theory of Error Correcting Codes", North Holland Publishing Co., New York, 1978.

22. J.F. Blake and Q.C. Mullin, "The Mathematical Theory of Coding", Academic Press, New York, 1975.

23. S.G.S. Shiva et al., "A Class of Composite Codes", IEEE Transactions on Information Theory, Vol. IT-27, No. 2, Mar. 1981.

24. R.W. Lucky, J. Salz and Weldon, "Principles of Data Communications; New York, McGraw Hill, 1968.

25. E.J. Weldon, Jr., "Euclidean Geometry Cyclic Codes", E.E. Dept., University of Hawaii Report, June 1967.

26. J.L. Massey, "Step by Step Decoding of the Bose-Chandhuri-Hocquenghore Codes", IEEE Trans. on Information Theory, Vol. IT-11, October 1965.

27. R.W. Hamming, "Error Detecting and Correcting Codes", Bell System Tech J., Vol. 29, 1950.

28. G.D. Forney, Jr., "Coding and its Applications in Space Communications", IEEE Spectrum, June 1970.

29. J.J. Stiffler, "Editorial of Special Issue on Coding Theory", IEEE Trans. on Communications.

30. S.G.S. Shiva and V. Mimis, "On the Majority Logic Decoding of a Class of Composite Codes", Collection of Abstracts, 1981 International Symposium on Information Theory, IEE Catalog Number 81 CH 1609-7-IT.