The bulk of the theory on error control codes has been developed under the fault assumption of random (symmetric) errors, where $1 \rightarrow 0$ and $0 \rightarrow 1$ errors are equally likely. In the past few years, several applications have emerged in which the observed errors are highly asymmetric. This has prompted the study of codes that offer a combination of symmetric and asymmetric error control capabilities. This research is a part of this ongoing study. The main results of the research are listed below.

1. New upper bounds on $t$-unordered codes. Exact bounds are established in some cases.
2. A new method for constructing constant weight distance four codes that gives the best known bounds in several cases.

3. A new method for constructing single asymmetric error correcting codes. The method establishes several new lower bounds.

4. A construction for symmetric error correcting code. The code is suited for a photon channel and other highly asymmetric channels because it has far fewer 1's than 0's. The code uses one extra bit of redundancy over the BCH code in almost all cases, and it is relatively easy to encode and decode.

5. A new construction for systematic double asymmetric error correcting code. The resulting code is easier to decode than the BCH code and is optimal in several cases. The code has fewer 1's than 0's.

6. A new construction for double symmetric error correcting linear code. The resulting code is easier to decode than the BCH code and is optimal in several cases.

7. A new construction for linear codes. The construction yields best known codes in many cases.
New Bounds and Constructions for Error Control Codes

by

Nasir Ali Al-Darwish

A Thesis

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Completed March 6, 1989

Commencement June 1989
APPROVED:

Redacted for Privacy

Professor of Computer Science in charge of major

Redacted for Privacy

Chairman of Department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented March 28, 1989

Typed by Nasir Ali Al-Darwish for Nasir Ali Al-Darwish
ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my advisor Prof. Bella Bose for his guidance and support throughout this research. Also, I would like to extend my thanks and respect to other members in my committee: Prof. Walter Rudd, Prof. Tom Dietterich, Prof. Toshi Minoura, from the CS Dept., and Prof. David Allstott from the ECE Dept.

Also, I would like to thank Prof. Paul Cull and Prof. Ted Lewis, and all other faculty and staff from the CS Dept. who made my stay here a good memory for years to come.

Also, I would like to acknowledge Oregon State University for giving me the opportunity to pursue my graduate work, and King Fahad University of Petroleum and Minerals, Dhahran, Saudi Arabia, for its financial support.

Finally, my love and appreciation to my wife Um Nadeem and my children Nadeem and Zahra for putting up with the pressure of my work and being an integral and meaningful part in my life.
# TABLE OF CONTENTS

## 1. Introduction

1.1. The Coding Problem

1.2. Error and Channel Types

1.3. Applications of Asymmetric/Unidirectional Error Control Codes

1.4. Preliminary Definitions and Results

1.5. Thesis Outline

## 2. Bounds on $t$-Unordered Codes

2.1. Problem Definition

2.2. Review of Literature on $t$-Unordered Codes

2.3. Bounds on $t$-Unordered Codes

2.4. Other Bounds on $t$-Unordered Codes

2.5. A Construction for Constant Weight Distance Four Codes

## 3. Asymmetric/Unidirectional Error Control Codes

3.1. Introduction

3.2. Literature Review on Asymmetric/Unidirectional Error Control Codes

3.3. On the relationship among Asymmetric/Unidirectional/Symmetric Error Correcting Codes
3.4. A Construction for Asymmetric Distance Two Codes

4. Efficient Double Asymmetric Error Correcting Codes
   4.1. Introduction
   4.2. Codes for Photon Communication
   4.3. Construction I
   4.4. Construction II
   4.5. Analysis of the Proposed Codes
   4.6. Concluding Remarks

5. New Constructions for Linear Codes
   5.1. Introduction
   5.2. Construction I
   5.3. Construction II
   5.4. Construction III
   5.5. Analysis of the Proposed Codes

6. Conclusion
   6.1. Summary
   6.2. Future Research

Bibliography
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Block diagram of a typical communication channel</td>
<td>2</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.1</td>
<td>Best known lower bounds for constant weight distance four codes</td>
</tr>
<tr>
<td>3.1</td>
<td>A summary of constructions for asymmetric distance two codes</td>
</tr>
<tr>
<td>4.1</td>
<td>The complete code for Example 4.1</td>
</tr>
<tr>
<td>4.2</td>
<td>A summary of code redundancy for Construction II</td>
</tr>
<tr>
<td>4.3</td>
<td>Values of $\beta$ and $\gamma$ parameters</td>
</tr>
<tr>
<td>5.1</td>
<td>The complete code for Example 5.1</td>
</tr>
<tr>
<td>5.2</td>
<td>A summary of code redundancy for Construction I</td>
</tr>
<tr>
<td>5.3</td>
<td>$[n,k,d]$ values obtained from Constructions I &amp; II</td>
</tr>
</tbody>
</table>
1.1. The Coding Problem

The basic unit of information inside a computer system is called a bit. A bit can have a value of either "0" or "1". Higher levels of information are encoded by sequences of 0's and 1's. An example of this is the ASCII code, which uses seven bits to represent ordinary digits and letters. The ASCII code is one of numerous kinds of codes in use in modern computer systems. However, this code by itself does not offer any protection from errors because each of the 7-bit binary sequences represents a valid piece of information. Codes that enable the detection and/or correction of errors are called error detecting/correcting codes. Error detecting/correcting codes have become an integral part of today's computer communication and storage systems. The transmission and storage of digital information have much in common. The term channel is used to refer to either a communication medium such as a telephone line or a storage medium such as an optical disk. The basic structure of a communication channel is shown in Fig. 1.1.
Fig. 1.1. Block diagram of a typical communication channel.

Between the time it is sent (or recorded) and the time it is received (or retrieved), the digital data can be corrupted by "noise" in the channel. The noise may be an electrical fluctuation, lightning, equipment malfunction, and so on. The object of an error correcting code is to encode the data, by adding a certain amount of redundancy, so that the original data can be recovered if not too many errors have occurred. It is important that a code be efficient (uses a small amount of redundancy) and easy to encode and decode. The design of good codes is the main problem in coding theory. To be able to characterize the optimality of these codes, it is important to know the maximum number of codewords a code can have. Therefore, an equally important problem is to determine good lower and upper bounds on the size of a code that satisfies certain constraints.

1.2. Error and Channel Types

In this section we define various channel models and error types. We use the terms 0-error and 1-error to refer to an error of type (0 → 1) and (1 → 0),
respectively.

**Definition 1.1.** If both 0-errors and 1-errors occur in a word with equal probability, then the channel is called a *binary symmetric channel*, and the errors are called *symmetric errors* or *random errors*.

**Definition 1.2.** If the probability of one type of error is much smaller than the other, then the channel is an *asymmetric channel*. In an ideal asymmetric channel only a particular type of error can occur. These errors are called *asymmetric errors*.

**Definition 1.3.** If both 0-errors and 1-errors can occur, but in any particular word the errors are all of one type, then these errors are called *unidirectional errors*.

### 1.3. Applications of Asymmetric/Unidirectional Error Control Codes

The development of coding theory began in the late 1940’s with classical papers by Shannon [SHAN 48] and Hamming [HAMM 50]. Since then, an extensive theory of error control codes has been developed [PETE 72, MACW 77]. The bulk of the theory has been developed under the fault assumption of symmetric errors. However, in the last two decades several applications have emerged where the observed errors are highly asymmetric. Some of these applications are surveyed next.

In many types of recently developed LSI circuits, the error failures in the cells of these memories are most likely caused by leakage of the charge, since a charge cannot be created except by a rewrite process. Therefore, these memory cells are apt to exhibit unidirectional failures. Although the rest of the memory system — the
address decoding, the cell selection, the gating logic, and so on — is subject to symmetric failures, for the overall memory system the probability of 1-error is significantly greater than 0-error. On the other hand, it has been established that the various faults in many digital devices are the sources of unidirectional errors [WAKE 75, WAKE 78, PRAD 80a, MAK 82, WONG 83]. Typical digital devices which exhibit unidirectional errors include data transmission systems, shift-register and magnetic-recording mass memories, and LSI/VLSI circuits such as ROM's and PLA's.

Unidirectional errors are particularly characteristic of the shift register type of memory, such as magnetic bubble memory or the CCD (charge coupled device) memory [RAO 89]. In this case, a stuck-at-1 or stuck-at-0 fault in a single cell will result in a constant 1 or 0 output from the corresponding shift register. This is because every bit on the track has to be circulated through the defective point in order to complete one shift cycle. In tape or disk memories, burst errors due to dust particles, minute scratches, and defects in coating will also sometimes result in unidirectional errors.

Another application of the asymmetric/unidirectional error control codes is found in optical communication [TAKA 76, LEIS 84, UEYM 88]. In an optical communication channel, the photon is used to transmit the information. The source is a photon transmitter, and the receiver is a photon counter. Because the number of received photons is never more than that of transmitted photons, the photon channel
is an asymmetric channel. For this type of channel it is desirable that codewords have as few 1's as possible. This reduces the code error probability, as well as the energy required for photon communication.

1.4. Preliminary Definitions and Results

In this section we give some basic definitions and theorems. These are illustrated by some examples. Also in this section, we introduce the notations, abbreviations, and other important keywords (shown in italic) that are used in subsequent chapters.

In this research we are exclusively concerned with binary block codes. A binary block code \( C \) of length \( n \) is just a set of binary \( n \)-tuples (also called words or vectors). In most cases, the code is a mapping (one-to-one) from the set of all binary vectors of length \( k \), called the information bits, into a subset of the binary vectors of length \( n \), where \( n \geq k \). The extra \( (n - k) \) bits are called the check bits. The ratio \( k/n \) is referred to as the information rate. The code is systematic if the information bits are separate from the check bits and are not modified during encoding.

As we noted earlier, the majority of the error control codes have been designed to cope with random errors. The literature on this class of codes is plentiful; the book by McWilliams and Sloane [MACW 77] is a very good reference. Outstanding among these codes are the Hamming code for single error correction, and the extended Hamming code for single error correction and double error detection. The BCH code
is a generalization of the Hamming code and can be designed to detect and/or correct any number of errors. These codes are equivalent to cyclic codes, for which efficient means of encoding and decoding have been developed.

The theory of error detection/correction relies on the assumption that the probability of no error is greater than that of at least one error, and the probability of a single error is greater than that of two errors, etc. Therefore the codewords are chosen such that any pair of codewords will remain distinguishable in the presence of a predetermined number of errors. The received word is decoded to the codeword to which it is nearest. This is known as the maximum likelihood decoding principle or the majority decoding rule.

**Definition 1.4.** The *Hamming weight* (or simply called weight) of a vector \( X = (x_1, x_2, \ldots, x_n) \), denoted by \( \text{weight}(X) \), counts the number of nonzero component \( x_i \)'s in \( X \).

**Definition 1.5.** The *Hamming distance* (or simply called distance) between two vectors \( X \) and \( Y \), denoted by \( d(X,Y) \), is the number of positions in which the two vectors differ.

**Definition 1.6.** For two binary vectors \( X \) and \( Y \), \( N(X,Y) \) denotes the number of \( 1 \to 0 \) cross-overs from \( X \) to \( Y \), i.e. in set notation, \( N(X,Y) = |X - Y| \). \( X \) and \( Y \) are unordered iff \( N(X,Y) > 0 \) and \( N(Y,X) > 0 \); otherwise they are ordered, and \( X \) covers \( Y \) if \( N(Y,X) = 0 \).
Definition 1.7. The asymmetric distance between two binary vectors $X$ and $Y$, denoted by $d_A(X,Y)$, is defined as the maximum of $\{ N(X,Y), N(Y,X) \}$. For two binary vectors $X$ and $Y$, $d(X,Y) = N(X,Y) + N(Y,X)$, and $d(X,Y) = d_A(X,Y)$ iff $X$ and $Y$ are ordered.

Definition 1.8. A code $C$ is said to have a minimum distance (or asymmetric distance) $d$ if $d(X,Y) \geq d$ (or $d_A(X,Y) \geq d$) for all distinct codewords $X, Y \in C$.

The following theorems give the necessary and sufficient conditions for the detection and/or correction of symmetric errors using block codes [Hamm 50, Pete 72, Macw 77, Rao 89].

Theorem 1.1. A code can detect $t$ errors iff its minimum distance is $t+1$.

Theorem 1.2. A code can correct $t$ errors iff its minimum distance is $2t+1$.

Theorem 1.3. A code can correct $t$ errors and simultaneously detect $d$ errors ($d \geq t$) iff its minimum distance is $t+d+1$.

Example 1.1 (Distance-2 code)

The code \{ 00, 11 \} can detect a single error. If one error occurs, the received word will be 01 or 10, and the error is detected. However, if two errors occur, then one codeword becomes another, and the errors will not be detected. The above code is an even parity code. It uses one check bit. The check is chosen such that the total number of 1's in the codeword is even. This code can detect a single error since the distance between any two codewords is at least 2. In fact, this code can detect any
odd number of errors since the weight of the resulting word will be odd.

**Example 1.2** (Distance-3 code)

The code \{ 000, 111 \} can be used as either a single error correcting (SEC) or double error detecting (DED) code. If one error occurs in a transmitted word, it can be corrected by a majority decoding. If two errors occur, say for codeword 000, the received word will be closer to 111 and therefore will be decoded incorrectly, and the majority decoding fails. However, if no correction is attempted, but *detect-only* mode is employed, then double errors can be detected.

The next two theorems give the necessary and sufficient conditions for the detection and correction of asymmetric errors [CONS 79, BORD 82].

**Theorem 1.4.** A code \( C \) can detect \( t \) asymmetric (or unidirectional) errors iff for all distinct codewords \( X, Y \in C \), either \( X \) and \( Y \) are unordered, or \( d(X,Y) \geq t+1 \).

**Theorem 1.5.** A code can correct \( t \) asymmetric errors iff its minimum asymmetric distance is \( t+1 \).

From Theorem 1.4 it follows that if any two codewords are unordered then the code is capable of detecting all (any number of) unidirectional errors. Such a code is called an *unordered code*. If a code is unordered, then no codeword can become another by 1-errors (or 0-errors) only. On the other hand, for any two words \( X, Y \), such that \( X \) covers \( Y \), \( X \) can be changed to \( Y \) by 1-errors, and \( Y \) can be changed to \( X \) by 0-errors. Therefore, the condition that the code be unordered is both necessary
and sufficient.

**Example 1.3 (Unordered code)**

Consider the code which consists of all binary words of length $n$ and weight $k$ (usually called $k$-out-of-$n$ code). In this code any two codewords are unordered. Thus this code can detect all unidirectional errors. The errors are detected by the fact that in case of errors, the received word will have weight $\neq k$. As we shall see in the next chapter, the largest unordered code of length $n$ is achieved by taking all words of weight $n/2$.

Other terms that we will be using in this thesis are defined next. A code of length $n$ is *balanced* if all codewords are of weight $\lfloor n/2 \rfloor$ (or $\lceil n/2 \rceil$).

Also the following notation is used.

- $|X|$ : cardinality of the set $X$.
- $\lfloor X \rfloor$ : integer part of $X$.
- $\lceil X \rceil$ : the largest integer $\leq X$.
- $\lfloor X \rfloor$ : the smallest integer $\geq X$.
- $\text{GF}(q)$ : Galois field with $q$ elements.

**1.5. Thesis Outline**

In this research we study several problems concerned with finding good (lower and upper) bounds for different kinds of codes. These codes have a combination of asymmetric/unidirectional/symmetric error detection/correction capabilities. The presentation is divided into independent topics which are covered in Chapters 2 through 5. The background and literature review for each topic are covered in its
In Chapter 2 we introduce \textit{t-unordered codes}, which are a generalization of unordered codes. A \textit{t-unordered code} can correct \( t-1 \) errors and simultaneously detect all unidirectional errors (\((t-1)\)-EC/AUED). We derive some bounds on the size of these codes, and we establish the exact bound in some specific cases. A method for constructing constant weight distance four codes is given. These give rise to \(2\)-unordered codes which are of practical interest.

In Chapter 3 we investigate some of the properties of asymmetric/unidirectional error detecting/correcting codes. We prove a theorem relating asymmetric and unidirectional error correcting codes. Also in this chapter, we give a method for constructing single asymmetric error correcting codes. The method establishes several new lower bounds.

In Chapter 4 we consider the design of codes for optical communication channels, where it is important that most codewords have small weight. We give two constructions for double asymmetric error correcting codes. The first construction uses a subset of a double error correcting BCH code. The code is semisystematic, but it offers a considerable improvement in the weight distribution at the expense of one extra bit of redundancy (in almost all cases) over the BCH code. The code has the same degree of encoding/decoding complexity as the BCH code. The second construction is completely systematic and has twice the information rate as the BCH code in several cases. This code is easier to decode than the BCH code and has fewer
1's than 0's.

In Chapter 5 we give new constructions for linear codes. The first two constructions given are for double error correcting linear codes. These codes have higher information rates than the BCH code in many cases, and yet they are easier to decode. The first construction uses $2r+1$ check bits for information length up to $2'^r - r - 2$, and $2r$ check bits for information length $= 2'^r - 1 - r$. The second construction gives one more extra bit of information, but it is slightly more difficult to decode. A generalization of the latter construction gives a $t$ error correcting code that would use at most $tr+1$ check bits for information length up to $2'^r - 1 - (t-1)r$ bits.

In Chapter 6 we give a summary and point out directions for future research.
Chapter 2

Bounds on t-Unordered Codes

In this chapter we address the problem of determining the maximum number of codewords in a t-unordered code. In Section 2.1 we give a formal definition of this code and characterize its error correcting/detecting capabilities. Also in this section, we state the problem and the motivation. In Section 2.2 we give a review of the literature on t-unordered codes. In Sections 2.3 and 2.4 we derive some bounds on the size of these codes and establish the exact bound in some specific cases. Finally, in Section 2.5 we give a method for constructing 2-unordered codes.

2.1. Problem Definition

Definition 2.1. A code $C$ is $t$-unordered if and only if $N(X,Y) \geq t$ for all distinct codewords $X, Y \in C$. If $t = 1$, then the code is unordered.

Unordered codes are useful for symmetric error correction and unidirectional error detection, as given by the next theorem [PRAD 77, BOSE 80, BOSE 82a].

Theorem 2.1. A code $C$ is $(t-1)$-EC/AUED if and only if it is $t$-unordered.

We will refer to a $t$-unordered code of length $n$ as $(n,t)$-unordered. We use $U(n,t)$ to denote the maximum number of codewords in an $(n,t)$-unordered code and $U_l(n,t)$ to denote the lower bound on $U(n,t)$. As customary, let $A(n,d)$ denote the maximum number of codewords in a binary code of length $n$ and distance $d$, and let
A(n,d,w) denote the maximum number of codewords in a binary code of length n, distance d, and weight w. Also, let A₁(n,d,w) denote the lower bound on A(n,d,w).

Bounds and constructions for A(n,d) and A(n,d,w) can be found in [MACW 77, BEST 78, GRAH 80, SLOA 89]. Clearly, a constant weight code with distance 2t is t-unordered. On the other hand, since the distance between any two codewords in an (n,t)-unordered code is at least 2t, U(n,t) is at most A(n,2t). Therefore, the following lemma is obvious.

Lemma 2.1. \( \{ \text{maximum over } w \text{ of } A_1(n,2t,w) \} \leq U(n,t) \leq A(n,2t) \).

Our problem here is to get bounds tighter than those given by the above lemma. For t = 1, the problem is solved. In this case, the exact bound is \( \binom{n}{n/2} \) and is achieved by taking all words of weight \([n/2]\). This is known as Sperner's lemma [SPER 28]. The literature is abundant with proofs for Sperner’s lemma. Some of these can be found in [FRIE 62, KLEI 74]. Most of the proofs for Sperner’s lemma are given from a set theory standpoint. A collection (set) of subsets from a given set such that no subset is contained in another is known as Sperner’s family. A Sperner’s family is said to be an antichain, where a chain of length k is a sequence of k elements in which the first element covers the second, the second covers the third, etc. Sperner’s lemma gives the maximum number of elements in a Sperner’s family. We can think of t-unordered codes (where t ≥ 2) as a generalization (further restrictions on the elements of a Sperner’s family) of a Sperner’s family. Actually, a
Sperner's family has been generalized by relaxing the conditions on the elements; in a
$k$-family the maximum length of a chain is $k$. The maximum number of elements in a
$k$-family is known [GREE 74]. The maximum number of codewords in an
$(n,t)$-unordered code is the same as the maximum number of subsets of a set with $n$
elements such that for any two subsets there are at least $t$ elements in one subset but
not in the other, and vice versa. In [IKEN 71] and [NAKA 72], it is conjectured that
the maximum is achieved by a balanced code. If the subsets are restricted to a given
cardinality $k$, then this number is the same as $A(n,2t,k)$, for which the exact bound
is generally unknown. The tables given in [GRAH 80] and [SLOA 89] show that
$A_t(16,6,6) > A_t(16,6,7)$; thus it is still a research problem [MACW 77, p. 691] to
show:

Is it true that if $w_1 < w_2 \leq n/2$ then $A(n,d,w_1) \leq A(n,d,w_2)$?

2.2. Review of Literature on Unordered Codes

As we have stated, unordered codes are useful for symmetric error correction
and all unidirectional error detection. Most of the work reported in the literature is
on the design of good unordered codes and their applications. Early work on unor-
dered codes is by Berger [BERG 61], who designed systematic unordered codes.
These have the advantage of easy encoding and decoding and are optimal (use as few
check bits as possible) [FRIE 62]. Unordered codes have been shown to be useful in
maintaining data integrity in optical disks [LEIS 84]. For fiber optic and electromag-
netic wire links, the class of unordered codes which is balanced has desirable properties, such as each codeword being dc-free [TAKA 76, WIDM 83]. Knuth [KNUT 86] gave a method for encoding information bits into a balanced code. Further extensions to Knuth's method are given in [BOSE 87, ALBA 89b]. A variation of unordered codes for $d$ unidirectional error detection is given by Bose and Lin [BOSE 85].

Another class of unordered codes of practical interest is single error correcting all unidirectional error detecting (SEC/AUED) codes. These codes can be compared with the popular SEC/DED codes that now dominate applications in computer memories. The early work on SEC/AUED codes is of the nonsystematic type by Pradhan and Reddy [PRAD 77], Bose [BOSE 80], and Bose and Rao [BOSE 82a]. Systematic SEC/AUED and more general $t$-EC/AUED have been studied by Niklos [NIKO 84, NIKO 86], Blaum [BLAU 87], Lin [LIN 87], Tao [TAO 88], and most recently by Bruck [BRUC 89] and Albassam [ALBA 89a].

2.3. Bounds on $t$-Unordered Codes

In this section and the next we prove several upper bounds on $t$-unordered codes and establish the exact bound in some cases.

Lemma 2.2. $U(n,t) \leq 2U(n-1,t)$.

Proof:

Consider a particular column in an $(n,t)$-unordered code. Those words which have 1 (0) in that column form an $(n-1,t)$-unordered code which, by definition, can
have at most $U(n-1,t)$ codewords.

**Theorem 2.2.** If $n < 4t$ then $U(n,t) \leq 2[2t/(4t-n)]$.

**Proof:**

In an $(n,t)$-unordered code having $m$ codewords, the number of 1→0 crossovers must be at least $\left(\frac{m}{2}\right)^2 2t$, i.e. for each pair of words we must have $2t$ crossovers. On the other hand, if the number of 1's in the $i$th column is $x_i$, then the number of crossovers in that column is $x_i(m-x_i)$. Therefore, the following inequality must be satisfied by the code:

$$\sum_{i=1}^{n} x_i(m-x_i) \geq m(m-1)t \quad (2.1)$$

Since the maximum number of crossovers in the $i$th column is attained when $x_i = m/2$ (if $m$ is even), or $x_i = (m-1)/2$ (if $m$ is odd), then we have:

for even $m$:

$$n(m/2)(m/2) \geq m(m-1)t \implies m \leq 4t/(4t-n) \leq 2[2t/(4t-n)] \quad (2.1-a)$$

for odd $m$:

$$n(m-1)(m+1)/4 \geq m(m-1)t \implies m \leq n/(4t-n),$$

and because $n < 4t$, we get

---

1 This bound, known as Plotkin's bound [MACW 77], is applicable for $A(n,d)$. Here we use a slightly different proof approach, which is also used in proving a subsequent theorem.
\[ m \leq \frac{4t}{(4t-n)} - 1 \quad \Rightarrow \quad m \leq \frac{4t}{(4t-n)} - 1 \leq 2\frac{2t}{(4t-n)} \]  
\hspace{1cm} (2.1-b)

For \( n \geq 4t \), Lemma 2.2 can be applied repeatedly until the above theorem can be applied. For \( n = 4t \), we get \( U(4t,t) \leq 8t \). This is in general an exact bound for \( A(4t,2t) \), but can be improved (Theorem 2.3) when the code must also be \( t \)-unordered.

**Definition 2.2.** An \((n,t)\)-unordered code \( C \) with \( m \) codewords is **perfect \( t \)-unordered** iff \[ n \frac{\lfloor m/2 \rfloor \lfloor m/2 \rfloor}{2} = \binom{m}{2} 2t. \]

**Lemma 2.3.** For a perfect \((n,t)\)-unordered code \( C \) with \( m \) codewords,

(a) \( N(X,Y) = N(Y,X) = t \) for any two distinct codewords \( X, Y \in C \).

(b) All words are of the same weight.

(c) If \( m \) is even then so is \( n \).

**Proof :**

(a) From the definition, the number of cross-overs between any two codewords cannot be more than \( 2t \), and since the code is \( t \)-unordered, then any two codewords must be exactly \( t \)-unordered.

(b) This follows from the fact that words of different weight cannot be exactly \( t \)-unordered.

(c) If \( m \) is even, then from (a) any column has exactly \( m/2 \) 1's. But from (b) all words in \( C \) are of the same weight. Therefore, any word must have exactly
\( n/2 \) 1's. This can be the case only if \( n \) is even.

**Lemma 2.4.** \( U(3t, t) = 3 \ (4) \) if \( t \) is odd (even).

**Proof:**

From Theorem 2.2, we get \( U(3t, t) \leq 4 \). For \( t = 2 \), we can construct the following (6,2)-unordered code \( C \):

- 001 011
- 010 101
- 100 110
- 111 000

For \( t = 2k \), we can concatenate \( k \) copies of \( C \). This gives a (6k,2k)-unordered code having 4 codewords. Definition 2.2 shows that for \( n = 3t \) and \( m = 4 \), the code is a perfect \( t \)-unordered code, and thus \( 3t \) has to be even since \( m \) is even. Therefore, when \( t \) is odd, \( m \) can be at most 3. In fact, such a code is also perfect and can be constructed by concatenating \( t \) copies of the 1-out-of-3 code, or, to get a balanced code, by concatenating \( t \) codes alternating between 1-out-of-3 and 2-out-of-3 codes.

**Theorem 2.3.**

1. (a) \( U(4t, t) \leq 8t - 2 \).

(b) \( U(4t-1, t) \leq 4t - 1 \).

(c) \( U(4t-2, t) \leq 2t \).

2. The bounds above are exact if a Hadamard matrix of order \( 4t \) exists.

**Proof:**
From Theorem 2.2 1 (c) follows, and 1 (a) follows from 1 (b) and Lemma 2.2. We prove 1 (b) by contradiction. Suppose that a \((4t-1,t)\)-unordered code containing \(4t\) words exists. In accordance with (2.1-a), the maximum number of cross-overs in such a code is \((4t-1)(2t)(2t)\), while the number of cross-overs that need to be taken care of is \(4t(4t-1)t\). These are equal; therefore, in each column we must have exactly \(2t\) 1's (and \(2t\) 0's).\(^2\) If the weight of the top word is \(w\), then we can have at most a total of \((4t-1)(w-t)\) 1's in these columns in the remaining words, since we must have at least \(t\) 0's in the positions of every word under the 1's of the top word. On the other hand, we need precisely \(w(2t-1)\) 1's to ensure that each of these columns ends up having \(2t\) 1's. Therefore, the inequality \((4t-1)(w-t) \geq w(2t-1)\) must be satisfied. This implies that \(w \geq (4t-1)/2\). When we consider the columns where the top word has 0's, a similar argument gives the inequality \((n-w) \geq (4t-1)/2\), which is same as \(w \leq (4t-1)/2\). Since \(4t-1\) is odd, no integer value of \(w\) can satisfy both inequalities. Therefore, a \((4t-1,t)\)-unordered code having \(4t\) words cannot exist.

A Hadamard matrix of order \(n\) is an \(n \times n\) matrix of +1's and -1's, such that any two distinct rows (or columns) are orthogonal. In such a matrix any two distinct rows (columns) differ in exactly \(n/2\) places. The matrix is normalized if the first row

\(^2\) This code is a perfect unordered code, and since it has \(4t\) words (i.e. an even number), then the word length must be even (Lemma 2.3), which is also a contradiction.
and the first column consist of just +1's. It is proven [PETE 72, MACW 77] that for a normalized Hadamard matrix of order \( n > 2, n = 4t \). Moreover, each row (column) except the first has exactly \( 2t +1 \)'s and \( 2t -1 \)'s, and any two distinct rows (columns) have exactly \( t +1 \)'s in common.

Let \( C \) denote the code obtained after dropping the first row and the first column in a normalized Hadamard matrix of order \( 4t \) and replacing \(-1 \)'s by 0's. This code is of length \( 4t-1 \) and has \( 4t-1 \) words. All codewords have the same weight \( (2t-1) \), and any two distinct codewords differ in exactly \( 2t \) places; thus they are exactly \( t \)-unordered.\(^3\) This is the code for 1 (b). Let \( \overline{X} \) denote the complement of a codeword \( X \in C \). It is clear that \( X \) and \( \overline{X} \) are \( t \)-unordered. Also, because any two distinct codewords in \( C \) have exactly \( t (t-1) \) 0's (1's) in common, \( N(\overline{X}, Y) = t \) and \( N(Y, \overline{X}) = t-1, \forall Y \in C \) and \( Y \neq X \). Therefore, the code for 1 (a) can be obtained by choosing both \( C \) and its complement and then adding an extra column with 1's (0's) appended to the words in \( C \) (\( C \)'s complement). In \( C \) any column has exactly \( 2t \) 0's; therefore, by taking all words which have 0 in a particular column and dropping that column we obtain the code for 1 (c). Note that all of the above codes are balanced.

It is conjectured that Hadamard matrices can be constructed for any order \( n = 4t \). There are known construction methods for Hadamard matrices of order \( n \)

\(^3\) This is another example of a perfect \( t \)-unordered code.
for \( n \) powers of 2, or \( n-1 \) a prime number [MACW 77].

### 2.4. Other Bounds on \( t \)-Unordered Codes

Other bounds on \( U(n,t) \) can be established by considering the set of unordered words that can be obtained from a \( t \)-unordered code and noting that such a set is bounded by \( \binom{n}{\lceil n/2 \rceil} \).

**Lemma 2.5.** If \( X \) and \( Y \) are two \( t \) unordered words of length \( n \) and \( i + j \leq t-1 \), then \( E_X \cup E_Y \) contains \( |E_X| + |E_Y| \) unordered words, where \( w_X \) denotes the weight of \( X \) and \( E_X \) denotes the set of words that results from \( i \) 1-errors and \( j \) 0-errors in \( X \), i.e.,

\[
|E_X| = \binom{w_X}{i} \binom{n-w_X}{j}. \quad (2.2)
\]

**Proof:**

Either of the sets \( E_X \) or \( E_Y \) consists of unordered words, since each set consists of distinct words all of which are of the same weight. Also, for \( \hat{X} \in E_X \) and \( \hat{Y} \in E_Y \), \( \hat{X} \) and \( \hat{Y} \) are unordered because

\[
N(\hat{X}, \hat{Y}) \geq N(X,Y) - \{ \text{ max 1-errors in } X \} - \{ \text{ max 0-errors in } Y \} = t - i - j \geq 1,
\]

\[
N(\hat{Y}, \hat{X}) \geq N(Y,X) - \{ \text{ max 1-errors in } Y \} - \{ \text{ max 0-errors in } X \} = t - i - j \geq 1.
\]
Theorem 2.4. $U(n,t) \leq \binom{n}{\lfloor n/2 \rfloor} / \binom{\lfloor n/2 \rfloor}{t-1}$.\footnote{Der Jei Lin gave a similar proof.}

Proof:

From Lemma 2.5, an $(n,t)$-unordered code $C$ must satisfy the following constraint:

$$\sum_{X \in C} \binom{w_X}{t-1} \leq \binom{n}{\lfloor n/2 \rfloor}.$$  \hspace{1cm} (2.3)

Now consider the total number of 1's in $C$. If this number is less than $\lfloor mn/2 \rfloor$, we can replace $C$ by its complement and thus ensure that the number of 1's is at least $\lfloor mn/2 \rfloor$. So we can have the following constraint on $C$.

$$\sum_{X \in C} w_X \geq \lfloor mn/2 \rfloor.$$ \hspace{1cm} (2.4)

It is known \cite{TOME 85, p. 7} that the minimum of the left side in (2.3) subject to the constraint (2.4) occurs when all of the $w_X$'s are as equal as possible, i.e. the minimum is when $w_X = \lfloor mn/2 \rfloor / n = \lceil n/2 \rceil$, which establishes the theorem.

The bound given by Theorem 2.4 is better than known $A(n,d)$ in few cases ($t = 2$ and $n \geq 14$). The next bound does better for higher values of $t$ and $n$. 
Theorem 2.5. Let $w$ denote the smallest weight in the code.

a) If $t$ is odd, and letting $u = (t-1)/2$, then

$$U(n,t) \leq \binom{n}{n/2} / \sum_{i=0}^{u} \binom{w}{i} \binom{n-w}{i}.$$

b) If $t$ is even, and letting $u = t/2$, then

$$U(n,t) \leq \binom{n}{n/2} / \sum_{i=1}^{u} \binom{w}{i} \binom{n-w}{i-1}.$$

Proof:

In either case we are generating constant weight words from each codeword. The words resulting from different codewords are unordered due to the fact that we are allowing no more than $t-1$ symmetric errors in a codeword, and at the same time ensuring that 1-errors in one codeword and 0-errors in another do not add up to $t$ (Lemma 2.5). It can be shown that the number of words generated for each codeword is the least when one of $\{w, n-w\}$ is as small as it can be.

Lemma 2.6. In an optimal $(n,t)$-unordered code $C$ (i.e. with $U_l(n,t)$ codewords), the minimum weight $w$ must satisfy the following constraint:

$$U(n-w,t) \sum_{i=0}^{w-t} \binom{w}{i} \geq U_l(n,t) - 1. \quad (2.5)$$

Proof:

Consider a codeword $X$ of weight $w$. For each of the remaining codewords to be $t$-unordered with $X$, it can have at most $w-t$ 1's in the columns where $X$ has 1's, and thus there are $P$ distinct possible binary values for these $w$ bits, where
\[ P = \sum_{i=0}^{w-t} \binom{w}{i} . \]

This results in at least \((U(n,t)-1)/P\) codewords which agree in \(w\) columns, but this number is at most \(U(n-w,t)\).

**Example 2.1**

Let us determine an upper bound on a 3-unordered code of length 16, i.e. \(U(16,3)\).

From Lemma 2.6, and by letting \(U(16,3) = A(16,6,8) = 120\) [GRAH 80], it can be verified that \(w \geq 5\). Then, by using Theorem 2.5, we get

\[
U(16,3) \leq \binom{16}{8} / \sum_{i=0}^{5} \binom{5}{i} \binom{11}{i} = 229 .
\]

This is better than \(A(16,6) = 256\) [BEST 78].

**Theorem 2.6.** \(U(9,2) = 18\) and \(U(10,2) = 36\).

**Proof:**

There are constant weight balanced codes which achieve these bounds [GRAH 80], and it suffices to rule out the existence of a \((9,2)\)-unordered code having 19 words to complete the proof. Let \(C\) be a \((9,2)\)-unordered code having 19 codewords. The maximum number of cross-overs in \(C\) is \(n \lfloor m/2 \rfloor \lceil m/2 \rceil = 9 \times 10 \times 9 = 810\). From Lemma 2.6, it can be verified that the weight of any codeword in \(C\) must be either 4 or 5. If \(C\) contains \(x\) words of weight 4 and \(y\) words of weight 5, then the following
inequality must hold:

$$810 \geq 4 \left(\frac{x}{2}\right) + 4 \left(\frac{y}{2}\right) + 5 xy .$$

(2.6)

For a particular word $X$ with weight 4, the maximum number of codewords $X$ of weight 4 or 5 that are exactly 2-unordered with $X$, i.e. $N(X,X) = 2$ and $N(X,X) = 2$ (3) if weight($X$) = 4 (5), which themselves are 2-unordered is at most

$$\binom{4}{2} U(5,2) = 6 \times 2 = 12 .$$

(2.7)

Similarly for a codeword with weight 5, we can argue, when considering its zeros, that the maximum number of codewords that are exactly 2-unordered with it which themselves are 2-unordered is 12. Therefore, for any codeword we have $(19 - 13)$ codewords which are not exactly 2-unordered with it. Therefore, each codeword participates in at least 6 pairs; each pair results in at least 2 extra cross-overs. This results in at least $(19 \times 6 \times 2) / 2 = 114$ cross-overs which have yet to be accounted for in the right side of (2.6) above. Thus $C$ must satisfy:

$$810 \geq 4 \left(\frac{x}{2}\right) + 4 \left(\frac{y}{2}\right) + 5 xy + 114 .$$

(2.8)

Without loss of generality we can assume that $x > y$. It can be easily verified that there is no integer solution for $x$ and $y$, $10 \leq x \leq 18$, and $y = 19 - x$, that satisfies the above equation. Thus $C$ cannot exist.
2.5. A Construction for Constant Weight Distance Four Codes

In this section we present a method for constructing constant weight distance four codes. The method attains the best known lower bound in several cases. The method is based on utilizing all possible codes of shorter length which are usually obtained via group theoretic methods. These in turn are meshed to obtain a larger length code. The results of this section are summarized in Tables 2.1.

Let $P(n,w)$ denote a partition of the set $w$ out-of $n$ binary words into classes $G_0, G_1, \ldots$ etc., where each $G_i$ is a distance four code. For $G_i \in P(n_1,w_1)$ and $H_i \in P(n_2,w_2)$, let $G_i \times H_i$ denote the set of words which results from catenating each word in $G_i$ to every word in $H_i$. Since each $G_i$ and each $H_i$ is a distance four code, it is clear that $G_i \times H_i$ is also a distance four code. Also, for $i \neq j$, $(G_i \times H_i) \cup (G_j \times H_j)$ is also a distance four code, since a word from the first set and a word from the second set will be distance two in the first $n_1$ positions and also distance two in the last $n_2$ positions. It is obvious that in order to maximize the number of words generated, it is best to pair large $G_i$'s with large $H_i$'s. Therefore, for our purpose we will assume that the partitions in a given $P(n,w)$ are listed in order of non-increasing size.

Given $P(n_1,w_1) = \{ G_0, G_1, \ldots, G_j \}$, and $P(n_2,w_2) = \{ H_0, H_1, \ldots, H_k \}$, let $M(n_1,w_1,n_2,w_2)$ denote the set of words which results from $\bigcup_{i=0}^{\min(j,k)} (G_i \times H_i)$. 
Lemma 2.7. $M(n_1, w_1, n_2, w_2)$ is a distance four code of length $n_1 + n_2$ and weight $w_1 + w_2$.

Theorem 2.7. The set of words which results from $\bigcup_i M(n_1, w_1 + 2i, n_2, w_2 - 2i)$ for $i = 0, 1, -1, 2, -2, \ldots$ etc., where $0 \leq w_1 + 2i \leq n_1$ and $0 \leq w_2 - 2i \leq n_2$, is a distance four code of length $n_1 + n_2$ and weight $w_1 + w_2$.

Proof:

From Lemma 2.7, the resulting mesh for a particular $i$ is a distance four code of length $n_1 + n_2$, and weight $(w_1 + 2i + w_2 - 2i) = w_1 + w_2$. For $i \neq j$, consider two words $X, Y$ where $X \in M(n_1, w_1 + 2i, n_2, w_2 - 2i)$ and $Y \in M(n_1, w_1 + 2j, n_2, w_2 - 2j)$. Without loss of generality, assume $i > j$. Then in the first $n_1$ positions, $N(X, Y) \geq (w_1 + 2i) - (w_1 + 2j) = 2(i - j) \geq 2$, while in the last $n_2$ positions, $N(Y, X) \geq (w_2 - 2j) - (w_2 - 2i) = 2(i - j) \geq 2$. Therefore, $d(X, Y) \geq 4$. 


Example 2.2 (Distance four code of length 8 and weight 4)

Let $P(4,2) = \{ \{1100, 0011\}, \{1010, 0101\}, \{1001, 0110\} \}$. Clearly, $P(4,4) = \{ \{1111\} \}$ and $P(4,0) = \{ \{0000\} \}$. Thus, by using $M(4,2+2i,4,2-2i)$, we get:

For $i = 0$:

| 1100 | 1100 |
| 1100 | 0011 |
| 0011 | 1100 |
| 0011 | 0011 |
| 1010 | 1010 |
| 1010 | 0101 |
| 0101 | 1010 |
| 0101 | 0101 |
| 1001 | 1001 |
| 1001 | 0110 |
| 0110 | 1001 |
| 0110 | 0110 |

For $i = 1$:

| 1111 | 0000 |

For $i = -1$:

| 0000 | 1111 |

2.5.1. Generation of P(n,w) Partitions

Let $F^n_w$ denote the set of all binary vectors of length $n$ and weight $w$, and let $(G_n,+)$ be an abelian group of order $n$ whose elements are $\{g_0, g_1, \ldots, g_{n-1}\}$. Consider the map

$$T : F^n_w \rightarrow G,$$

whose value at $X = (x_0, \ldots, x_{n-1}) \in F^n_w$ is

$$T(X) = \sum_{i=1} g_i . \tag{2.9}$$

This mapping partitions $F^n_w$ into classes $C_i$'s, where $C_i = \{ X \mid T(X) = g_i \}$ for
It is proven that any $C_i$ is a distance four code [GRAH 80, BOSE 82].

For some given $n$ and $w$ we seek to partition $F^n_w$ into as few and large partitions as possible. We have found that the choice of an abelian group which is a direct product of the groups corresponding to the prime decomposition of $n$ (i.e. for $n = p_1 \times \cdots \times p_k$, where $p_i$ is a prime, use the abelian group $Z_{p_1} \times \cdots \times Z_{p_k}$) gives good results. In all subsequent constructions such an abelian group is used.

2.5.2. Lower Bounds for Some 2-Unordered Codes

In Table 2.1 we give the best known lower bounds for some constant weight distance four codes that are obtained by the proposed method. The entries for odd values of $n$ are obtained by shortening. The other entries are obtained by using a mesh of the type $M(n_1, w_1 + 2i, n_2, w_2 - 2i)$, where, as illustrated by Example 2.2, $n$ and $w$ are split evenly in $n_1$, $n_2$ and $w_1$, $w_2$, respectively.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$w$</th>
<th>lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>158</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>316</td>
</tr>
</tbody>
</table>

**Table 2.1.** Best known lower bounds for constant weight distance four codes (length $n$ and weight $w$).
In this chapter we investigate the properties of unidirectional/asymmetric error control codes. In Section 3.1 we give the basic theorems for the detection/correction of asymmetric/unidirectional errors. In Section 3.2 we give a literature review on asymmetric/unidirectional error correcting codes. In Section 3.3 we prove a theorem relating asymmetric and unidirectional error correcting codes. Finally, in Section 3.4 we give a method for constructing single asymmetric error correcting codes that achieves several new lower bounds.

3.1. Introduction

The basic theorem on the detection of asymmetric and unidirectional errors was given in Chapter 1 (Theorem 1.4).

From this theorem it can be seen that the two classes of asymmetric and unidirectional errors are equivalent under error detection. Not only that, the condition is far less restrictive than that for the detection of random errors, namely that the minimum distance of the code be $t+1$. For example, in the systematic case we can use $t$ check bits to design a Berger code that can detect any number of unidirectional errors in a code of length $2^t-1+t$, while by using $t$ check bits, at most $t$ random errors can be detected, since a systematic code with $t$ check bits has distance of at most $t+1$. In the nonsystematic case, Borden [BORD 82] proved that to detect $t$ or
fewer unidirectional errors, the optimal code of length $n$ is achieved by taking all binary words of weight congruent to $[n/2] \mod (t+1)$.

In the rest of this chapter we are concerned with codes that can correct (rather than detect) asymmetric or unidirectional errors. The basic theorem on the correction of asymmetric errors was given in Chapter 1 (Theorem 1.5). The next theorem states the necessary and sufficient conditions for the correction of unidirectional errors [BOSE 80, BOSE 82a].

**Theorem 3.1.** A code $C$ is capable of correcting $t$ unidirectional errors iff for any two distinct codewords $X, Y \in C$, either $X$ and $Y$ are unordered and $d_A(X, Y) \geq t+1$, or $d(X, Y) \geq 2t+1$.

It is clear from Theorem 1.5 and the above theorem that the two classes of asymmetric and unidirectional error correcting codes are not equivalent. As Theorem 3.1 states, for unidirectional error correction any two ordered codewords must satisfy a stronger condition (as the one for symmetric error correction) than that for asymmetric error correction. In designing a code for correcting a particular class of errors, one may begin with an error correcting code from a different class, which is then modified to cope with errors from the first class. Therefore, it is worthwhile to consider the relationship among the various classes of error correcting codes. This is the subject of Section 3.3, where we introduce a new theorem relating asymmetric and unidirectional error correcting codes.
3.2. Literature Review on Asymmetric/Unidirectional Error Correcting Codes

As we noted earlier, the majority of the error correcting codes have been designed to cope with random errors. The basic work on asymmetric and unidirectional error correcting codes is of the nonsystematic type aimed at constructing the largest codes for these classes of errors. Early work of this type is by Kim and Frie man [KIM 59], Varshamov [VARS 73] and McEliece [MCEL 73]; later work by Constantin and Rao [CONS 79], McEliece and Rodemich [MCEL 80], Delsarte and Piret [DELS 81], and Cunningham [CUNN 82]. The most recent work of this type is by Weber, de Vroedt, and Boekee [WEBE 88]. All of these studies deal exclusively with asymmetric errors. Upper bounds for asymmetric error correcting codes have been studied by Delsarte and Piret [DELS 81], Klove [KLOV 81a] and by Weber, de Vroedt, and Boekee [WEBE 87a]. Bounds and constructions for unidirectional error correcting codes have been recently given in [WEBE 87b].

Most of the other work reported in the literature deals with systematic codes originally designed for random errors which are modified to offer some protection against asymmetric/unidirectional errors. Among these are the \( t \)-EC/AUED codes that we reviewed in the previous chapter, and the \((t \text{-EC}/d\text{-UED})\) codes given by Lin and Bose [LIN 88] and most recently by Albassam [ALBA 89a].
3.3. On the Relationship among Asymmetric/Unidirectional/Symmetric Error Correcting Codes

The following theorems establish some relations between asymmetric (unidirectional) and symmetric error correcting codes [WEBE 87b].

**Theorem 3.2.** Any $t$-asymmetric error correcting code can be made into a $t$-symmetric error correcting code by appending at most $t$ bits.

**Theorem 3.3.** Any $t$-unidirectional error correcting code can be made into a $t$-symmetric error correcting code by appending at most $t - 1$ bits.

Next we will show that any asymmetric error correcting code can be made into a unidirectional error correcting code by appending at most three bits.

**Lemma 3.1.** Let $X$ and $Y$ be two binary words such that $d_A(X, Y) > t + 1$. If $|\text{weight}(X) - \text{weight}(Y)| \leq t$, then $X$ and $Y$ are unordered.

**Proof:**

If $X$ and $Y$ are ordered, i.e. say $X$ covers $Y$, and $\text{weight}(X) - \text{weight}(Y) \leq t$, then $d_A(X, Y)$ is at most $t$, which is a contradiction.

**Theorem 3.4.** Any $t$-asymmetric error correcting code can be made into a $t$-unidirectional error correcting code by appending at most $t$ bits if $t \leq 2$, or 3 bits if $t \geq 3$.

**Proof:**
Given a $t$-asymmetric error correcting code $C$, consider the following assignment of check bits to the codewords in $C$.

Case 1: $t \leq 2$.

$$\text{check}(X) = 2^t - 1 - \lfloor \text{weight}(X)/(t+1) \mod 2^t \rfloor \text{ in binary, } \forall X \in C. \quad (3.1)$$

The check bits are assigned according to (3.1). This is illustrated below for $t = 2$.

<table>
<thead>
<tr>
<th>weight</th>
<th>check</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1,2</td>
<td>11</td>
</tr>
<tr>
<td>3,4,5</td>
<td>10</td>
</tr>
<tr>
<td>6,7,8</td>
<td>01</td>
</tr>
<tr>
<td>9,11,11</td>
<td>00</td>
</tr>
<tr>
<td>12,13,14</td>
<td>11</td>
</tr>
</tbody>
</table>

It is easy to check that a group of $t+1$ consecutive weights are assigned the same check and that checks cycle every $2^t$ groups. Consider any two distinct codewords $X, Y \in C$. If $\text{check}(X) = \text{check}(Y)$, then either $|\text{weight}(X) - \text{weight}(Y)| \leq t$ or $|\text{weight}(X) - \text{weight}(Y)| > (2^t-1)(t+1)$. In the first case, $X$ and $Y$ are unordered by Lemma 3.1; the latter $d(X,Y)$ is at least the minimum distance required for $t$ symmetric error correction (i.e. $2t+1$). If $\text{check}(X) \neq \text{check}(Y)$, then we consider two cases:

1) For $t = 1$, $d(X,Y) \geq d_A(X,Y) = 2$. Also, $X$ and $Y$ differ in the check. This gives the minimum distance of 3 required for single symmetric error correction.
2) For $t = 2$, without loss of generality assume $\text{weight}(X) > \text{weight}(Y)$. If $\text{weight}(\text{check}(X)) > \text{weight}(\text{check}(Y))$, then, with the addition of the check bit, $X$ and $Y$ become unordered. If $\text{weight}(\text{check}(X)) > \text{weight}(\text{check}(Y))$, then out of the possible weight values, the closest $X$ and $Y$ can be is in one of the following:

(a) $\text{check}(X) = 11$ and $\text{check}(Y) = 00$

(b) $\text{check}(X) = 11$ and $\text{check}(Y) = 01$

(c) $\text{check}(X) = 10$ and $\text{check}(Y) = 00$

In case (a), with the addition of the check bits, we get:

$$d(X,Y) \geq d_A(X,Y) + 2 = 3 + 2 = 5.$$

In cases (b) and (c) we have a group of $t+1$ consecutive weights coming in between $X$ and $Y$ (e.g. in (b) the group check symbol is 00), so the distance between $X$ and $Y$ is at least $t+2$. With the addition of the check bits, the distance becomes at least $t+3 = 5$.

Case 2: $t \geq 3$.

Here we compute the check using (3.2).

$$\text{check}(X) = \lfloor \text{weight}(X)/(t+1) \rfloor \mod 3. \quad (3.2)$$

Then the check symbol is assigned according to the mapping

$$\{ 0 \rightarrow 001, 1 \rightarrow 010, 2 \rightarrow 100 \}.$$
With this assignment, we need to check that codewords which are assigned the same check are either unordered or have a minimum distance of $2t+1$. Without loss of generality, consider two codewords $x$ and $y$ in $C$, such that $\text{weight}(X) > \text{weight}(Y)$ and both are assigned the check 001. If $\text{weight}(X) - \text{weight}(Y) \leq t$, then they are unordered by Lemma 3.1. If $\text{weight}(X) - \text{weight}(Y) > t$, then at least two groups of codewords with weights higher than $\text{weight}(Y)$ have been assigned the checks $\{010, 100\}$; therefore, $\text{weight}(X) - \text{weight}(Y) > 2(t+1)$, which implies that $d(X,Y) > 2t+2$.

### 3.4. A Construction for Asymmetric Distance Two Codes

Let $P(n, w)$ be a partition of the set $w$ out-of $n$ into classes $G_0, G_1, \ldots$ etc., where each class is a distance four code. Let $Q(m)$ be a partition of the set of all binary words of length $m$ into classes $H_0, H_1, \ldots$ etc., where each class is a code with minimum asymmetric distance $= 2$. Then clearly, $G_i \times H_i$ is a code of length $n + m$ and minimum asymmetric distance $= 2$. Also, for $i \neq j$, $(G_i \times H_i) \cup (G_j \times H_j)$ is a code with minimum asymmetric distance $= 2$, since a word from the first set and a word from the second set will be distance two in the first $n$ positions and distinct in the last $m$ positions.

Given $P(n_1, w_1) = \{G_0, G_1, \ldots, G_j\}$ and $Q(n_2) = \{H_0, H_1, \ldots, H_k\}$, let $M(n_1, w_1, n_2)$ denote the set of words which results from $\bigcup_{i=0}^{\min(j,k)} (G_i \times H_i)$. 
Lemma 3.2. $M(n_1, w_1, n_2)$ is a code of length $n_1 + n_2$ and minimum asymmetric distance $= 2$.

Theorem 3.5. The set of words which results from $\bigcup_i M(n_1, w_1 + 2i, n_2)$ for $i = 0, 1, -1, 2, -2, \ldots$ etc., where $0 \leq w_1 + 2i \leq n_1$, is a code of length $n_1 + n_2$ and minimum asymmetric distance $= 2$.

Proof:

From Lemma 3.1, the resulting mesh for a particular $i$ is asymmetric distance two code of length $n_1 + n_2$. For $i \neq j$, consider two words $X, Y$ where $X \in M(n_1, w_1 + 2i, n_2)$ and $Y \in M(n_1, w_1 + 2j, n_2)$. Without loss of generality we can assume $i > j$. Then in the first $n_1$ positions, $N(X, Y) \geq (w_1 + 2i) - (w_1 + 2j) = 2(i - j) \geq 2$; therefore, $d_A(X, Y) \geq 2$.

Example 3.1 (Asymmetric distance two code of length 6)

Let us use $M(4, 2+2i, 2)$. Thus $Q(2) = \{ \{00, 11\}, \{01\}, \{10\} \}$. Let $P(4, 2) = \{ \{1100, 0011\}, \{1010, 0101\}, \{1001, 0110\} \}$. Clearly, $P(4, 4) = \{ \{1111\} \}$ and $P(4, 0) = \{ \{0000\} \}$. Thus we get:
For $i = 0$ :

- 1100 00
- 1100 11
- 0011 00
- 0011 11
- 1010 01
- 0101 01
- 1001 10
- 0110 10

For $i = 1$ :

- 1111 00
- 1111 11

For $i = -1$ :

- 0000 00
- 0000 11

Partitions of the type $P(n,w)$ are obtained using the method given in the previous chapter (2.5.1).

### 3.4.1. Generation of $Q(n)$ Partitions

Let $F^n$ denote the set of all binary vectors of length $n$, and let $(G_{n+1},+)$ be an abelian group of order $n+1$ whose elements are \{ $g_0$, $g_1$, ..., $g_n$ \}, where $g_0$ is the zero element. Consider the map

$$ T : F^n \rightarrow G, $$

whose value at $X = (x_1,x_2,...,x_n) \in F^n$ is

$$ T(X) = \sum_{z=1}^{n} g_i. $$

This mapping partitions $F^n$ into classes $C_i$'s, where $C_i = \{ X \mid T(X) = g_i \}$ for $0 \leq i \leq n$. It is proven that any $C_i$ is asymmetric distance two code [CON 79].

To partition $F^n$, the choice of an abelian group which is a direct product of the
groups corresponding to the prime decomposition of $n+1$ gives good results. In all subsequent constructions such an abelian group is used, unless otherwise specified.

### 3.4.2. New Lower Bounds for Asymmetric Distance Two Codes

In this section we explain specific constructions that improve upon some existing bounds for asymmetric distance two codes. An asymmetric distance two code of length $n$ is built by using a mesh of the type $M(n_1, w_1+2i, n_2)$, where, as a general strategy, $n_1 = \lfloor n/2 \rfloor$, $n_2 = \lfloor n/2 \rfloor$ and $w = \lfloor n_1 \rfloor$. The results of this section are summarized in Table 3.1.
1. $A_2(13,2) \geq 588$.

The set $F^6$ is partitioned into 7 classes; one of size 10 and 6 of size 9 each. Therefore, by using $M(7,4,6)$ we get:

For $i = 0$:

The set 4 out-of 7 is partitioned into 7 classes of size 5 each. Thus we get $(10 \times 5 + 6 \times 9 \times 5) = 320$ words.

For $i = 1$:

Here we can get all of $F^6$ by pairing each class in $F^6$ with a word from the set 6 out-of 7. Thus we get 64 words.

For $i = -1$:

The set 2 out-of 7 is partitioned into 7 classes of size 3 each. Thus we get $(10 \times 3 + 6 \times 9 \times 3) = 192$ words.

For $i = -2$:

Here we use the largest possible partition in any $Q(6)$, i.e. $A_2(6,2) = 12$.

Thus this code will have $(320 + 64 + 192 + 12) = 588$ words.
2. $A_2(14,2) \geq 1108$.

The set $F^6$ is partitioned into 7 classes; one of size 10 and 6 of size 9 each.

Therefore, by using $M(8,4,6)$ we get:

For $i = 0$:

The set 4 out-of 8 is partitioned into 8 classes; one class of size 14 and 7 classes of size 8 each. Thus we get $(10 \times 14 + 6 \times 9 \times 8) = 572$ words.

For $i = 1$ or $-1$:

The set 6 (2) out-of 8 is partitioned into 7 classes of size 4 each. Thus we get $(10 \times 4 + 6 \times 9 \times 4) = 256$ words.

For $i = 2$ or $-2$:

Here we use the largest possible partition in any $Q(6)$, i.e. $A_2(6,2) = 12$.

Thus this code will have $(572 + 2 \times 256 + 2 \times 12) = 1108$ words.
3. $A_2(15,2) \geq 2052$.

The set $F^7$ is partitioned into 8 classes of size 16 each. Therefore, by using $M(8,4,7)$ we get:

For $i = 0$:

The set 4 out-of 8 is partitioned into 8 classes; one class of size 14 and 7 classes of size 8 each. Thus we get $(16 \times 14 + 7 \times 16 \times 8) = 1120$ words.

For $i = 1$ or $-1$:

The set 6 (2) out-of 8 is partitioned into 7 classes of size 4 each. Thus we get $(16 \times 4 + 6 \times 16 \times 4) = 448$ words.

For $i = 2$ or $-2$:

Here we use the largest possible partition in any $Q(7)$, i.e. $A_2(7,2) = 18$.

Thus this code will have $(1120 + 2 \times 448 + 2 \times 18) = 2052$ words.
4. $A_2(17,2) \geq 7300$.

The set $P^8$ is partitioned into 9 classes; one class of size 32 and 8 classes of size 28 each. Therefore, by using $M(9,4,8)$ we get:

For $i = 0$:

The set 4 out-of 9 is partitioned into 9 classes of size 14 each. Thus we get $(32 \times 14 + 8 \times 28 \times 14) = 3584$ words.

For $i = 1$:

The set 6 out-of 9 is partitioned into 9 classes; one class of size 12 and 8 classes of size 9 each. Thus we get $(32 \times 12 + 8 \times 28 \times 9) = 2400$ words.

For $i = -1$:

The set 2 out-of 9 is partitioned into 9 classes of size 4 each. Thus we get $(32 \times 4 + 8 \times 28 \times 4) = 1024$ words.

For $i = 2$:

Here we can use all of $P^8$ by pairing each class in $Q(8)$ with a word from 8 out-of 9. Thus we get 256 words.

For $i = -2$:

Here we use the largest possible partition in any $Q(8)$, i.e. $A_2(8,2) = 36$. 
5. $A_2(19,2) \geq 26242$.

The set $F^9$ is partitioned into 10 classes; two classes of size 52 each and 8 classes of size 51 each. Therefore, by using $M(10,4,9)$ we get:

For $i = 0$ or 1:

The set 4 (6) out-of 10 is partitioned into 10 classes; 5 classes of size 22 each, and 5 classes of size 20 each. Thus we get $(2 \times 52 \times 22 + 3 \times 51 \times 22 + 5 \times 51 \times 20) = 10754$ words.

For $i = -1$ or 2:

The set 2 (8) out-of 10 is partitioned into 10 classes; 5 classes of size 5 each and 5 classes of size 4 each. Thus we get $(2 \times 52 \times 5 + 3 \times 51 \times 5 + 5 \times 51 \times 4) = 2305$ words.

For $i = -2$ or 3:

Here we use the largest possible partition in any $Q(9)$, i.e. $A_2(9,2) = 62$.

Thus this code will have $(2 \times 10754 + 2 \times 2305 + 2 \times 62) = 26242$ words.
6. $A_2(21,2) \geq 95340$.

The set $F_{10}$ is partitioned into 11 classes; one class of size 94 and 10 classes of size 93 each. Therefore, by using $M(11,5,10)$ we get:

For $i = 0$:

The set 5 out-of 11 is partitioned into 11 classes of size 42 each. Thus we get

$$(94 \times 42 + 10 \times 93 \times 42) = 43008 \text{ words.}$$

For $i = 1$:

The set 7 out-of 11 is partitioned into 11 classes of size 30 each. Thus we get

$$(94 \times 30 + 10 \times 93 \times 30) = 30720 \text{ words.}$$

For $i = -1$:

The set 3 out-of 11 is partitioned into 11 classes of size 15 each. Thus we get

$$(94 \times 15 + 10 \times 93 \times 15) = 15360 \text{ words.}$$

For $i = 2$:

The set 9 out-of 11 is partitioned into 11 classes of size 5 each. Thus we get

$$(94 \times 5 + 10 \times 93 \times 5) = 5120 \text{ words.}$$

For $i = -2$:

Here we can pair each word in 1 out-of 11 with one partition in $F_{10}$. Thus we get 1024 words.

For $i = 3$:
Here we use the largest possible partition in any $Q(10)$, i.e. $A_2(10,2) \geq 108$.

Thus this code will have $(43008 + 30720 + 15360 + 5120 + 1024 + 108) = 95340$ words.
7. $A_2(23,2) \geq 349600$.

The set $F^{11}$ is partitioned into 12 classes; 4 classes of size 172 each and 8 classes of size 170 each. Therefore, by using $M(12,6,11)$ we get:

For $i = 0$:

The set 6 out-of 12 is partitioned into 12 classes; 3 classes of size 80 each, 6 classes of size 78 each, and 3 classes of size 72 each. Thus we get $(3 \times 172 \times 80 + 172 \times 78 + 5 \times 170 \times 78 + 3 \times 170 \times 72) = 157716$ words.

For $i = 1$ or $-1$:

The set 8 (4) out-of 12 is partitioned into 12 classes; 3 classes of size 45 each and 9 classes of size 40 each. Thus we get $(3 \times 172 \times 45 + 172 \times 40 + 8 \times 170 \times 40) = 84500$ words.

For $i = 2$ or $-2$:

The set 10 (2) out-of 12 is partitioned into 12 classes; 9 classes of size 6 each and 3 classes of size 4 each. Thus we get $(4 \times 172 \times 6 + 5 \times 170 \times 6 + 3 \times 170 \times 4) = 11268$ words.

For $i = 3$ or $-3$:

Here we use the largest possible partition in any $Q(11)$, i.e. $A_2(11,2) \geq 174$.

Thus this code will have $(157716 + 2 \times 84500 + 2 \times 11268 + 2 \times 174) = 349600$ words.
Table 3.1. A summary of constructions for asymmetric distance two codes of length \( n \). Old bounds are from [WEBE 87b, WEBE 88].

<table>
<thead>
<tr>
<th>( n )</th>
<th>( n_1 )</th>
<th>( w_1 )</th>
<th>( n_2 )</th>
<th>( i = 0 )</th>
<th>( i = \pm 1 )</th>
<th>( i = \pm 2 )</th>
<th>( i = \pm 3 )</th>
<th>new bound</th>
<th>old bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>320</td>
<td>64,192</td>
<td>12</td>
<td></td>
<td>588</td>
<td>586</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>572</td>
<td>2 (256)</td>
<td>2 (12)</td>
<td></td>
<td>1108</td>
<td>1096</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1120</td>
<td>2 (448)</td>
<td>2 (18)</td>
<td></td>
<td>2052</td>
<td>2048</td>
</tr>
<tr>
<td>17</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>3584</td>
<td>2400,1024</td>
<td>256,36</td>
<td></td>
<td>7300</td>
<td>7296</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>4</td>
<td>9</td>
<td>10754</td>
<td>10754,2305</td>
<td>2305,62</td>
<td>62</td>
<td>26242</td>
<td>26216</td>
</tr>
<tr>
<td>21</td>
<td>11</td>
<td>5</td>
<td>10</td>
<td>43008</td>
<td>30720,15360</td>
<td>5120,1024</td>
<td>108</td>
<td>95340</td>
<td>95326</td>
</tr>
<tr>
<td>23</td>
<td>12</td>
<td>6</td>
<td>11</td>
<td>157716</td>
<td>2 (84500)</td>
<td>2 (11268)</td>
<td>2 (174)</td>
<td>349600</td>
<td>349536</td>
</tr>
</tbody>
</table>
Chapter 4

Efficient Double Asymmetric Error Correcting Codes

4.1. Introduction

In certain types of applications the errors are often of known polarity. For example, in optical communication photons may vanish or fail to be detected (1 → 0 error) but the creation of spurious photons (0 → 1 error) is impossible. For these kinds of applications the asymmetric channel model, where the errors are assumed to be of one type, is more appropriate. The expectation is that we might be able to design better codes (codes that use less redundant bits and are easier to encode and decode) than those used for the symmetric channel. While it is true that we can get more codewords for a given length, it seems that when we require that these codes be systematic then in most cases we cannot do better in terms of code size than systematic symmetric error correcting codes. For example, the only known systematic single asymmetric error correcting code with a higher information rate than that of the Hamming code is an (8,5)-code [CUNN 82], i.e. with 3 check bits 5 information bits can be accommodated, while the Hamming code would require 4 check bits. Also a look at a table of bounds for asymmetric and symmetric error correcting codes [BEST 78, DELS 81, WEBE 87b, WEBE 87c] shows that only in very few cases the lower bound for \( t \) asymmetric error correcting code of a given length is at least twice the corresponding bound for the symmetric case.
In light of these facts, we will be satisfied if we are able to design asymmetric error correcting codes that are easier to decode and encode than their counterparts for the symmetric channel. Also of interest is that these codes have smaller code error probabilities and, for photon communication, have higher information rates per photon [UYEM 88].

In this chapter, we present two constructions for double asymmetric error correcting codes. These codes have fewer 1's than 0's; a property which linear codes (or those which are union of cosets of a linear code) lack. The first construction is based on the BCH double error correcting code. In this case, one information bit is sacrificed to determine which one of a pair of codewords from the BCH code will be in the new code. The resulting code has far fewer 1's than 0's compared to a BCH code having the same number of information bits. This code is semisystematic but the complexity of the encoding/decoding process is comparable to that of the BCH code. The second construction is completely systematic. It uses $2r+1$ check bits to accommodate up to $2^r - r - 2$ information bits. A variation of this code uses $2r$ check bits to accommodate $2^{r-1} - r$ information bits. This code has information rate that is optimal or close to optimal in many cases, and is easier to decode than the BCH code. Also the resulting code has fewer 1's than 0's.

The chapter is organized as follows. In Section 4.2, we review some important parameters for codes used in optical communication channels. Construction I and II are given in Sections 4.3 and 4.4, respectively. In Section 4.5, we give an analysis of
the proposed codes. Section 4.6 contains some concluding remarks. Summaries of the relevant parameters of these codes are given in Tables 4.1, 4.2 and 4.3 at the end of this chapter.

4.2. Codes for Photon Communication

In the Photon Communication System [UYEM 88], the photon itself is used to transmit the information. Because the number of received photons is never more than that of transmitted photons, the photon channel is an asymmetric channel. To compensate for the energy losses in the optical channel, error correcting codes must be employed. Two criteria for good error correcting codes have been proposed [UYEM 88]. One criterion is that the average code error probability be minimized, and the other is that the information rate per photon be maximized. These two criteria are important for the reliability and the efficiency of the photon communication system.

Let $C$ be an asymmetric error correcting code with fixed $(n,k,t)$, where $n$, $k$ and $t$ denote the code length, the number of information bits, and the number of errors which the code can correct, respectively. Let $n_w$ be the number of codewords with weight $w$. Let $p$ denote the probability of a single $(1 \rightarrow 0)$ error. The average code error probability, $P_e$, is defined by

$$P_e = \left( \sum_{w=l+1}^{n} n_w \sum_{i=l+1}^{w} \binom{w}{i} p^i \right) / |C|.$$  

(4.1)
This can be approximated as:

\[ P_e \approx p^{t+1} \left( \sum_{w=0}^{n} n_w \binom{w}{t+1} \right) / |C|. \] (4.2)

Since \( \binom{w}{t+1} \) is an increasing function of \( w \), it is necessary that most codewords have small weights in order to minimize the code error probability.

The average weight, \( W_a \), of the code can be expressed as:

\[ W_a = \left( \sum_{w=0}^{n} w n_w \right) / |C|. \] (4.3)

Using the average weight, the information rate per photon is defined by

\[ I_B = k / W_a \ \text{(bit/photon)}. \] (4.4)

From the preceding discussion, it is apparent that for some given \( k \) and \( t \) we are seeking a code of length \( n \) which gives small values for the following parameters:

\[ \beta(n,k,t) = \sum_{w=0}^{n} n_w \binom{w}{t+1}, \] (4.5)

and

\[ \gamma(n,k,t) = \sum_{w=0}^{n} w n_w. \] (4.6)

These parameters are computed for the codes constructed here and are given in Table 4.3. Note that (4.6) gives the total \# of 1's in the code. For a linear code, this is half the code.
4.3. Construction I

Let $C$ be an $(n,k)$ double error correcting BCH code, where the information bits are the leftmost $k$ bits. Let $W \in C$ be a codeword with the maximum weight. Without loss of generality we can assume that $w_1$ (leftmost bit in $W$) = 1. For any codeword $X$ for which $x_1 = 0$, let $Y = W + X$. $Y \neq X$ since $y_1 = 1$. Also $Y \in C$ since $C$ is a linear code. Consider the code $\hat{C}$ defined by (4.7).

$$\hat{C} = \left\{ Y \middle| \begin{array}{l}
Y = X \text{ if } \text{weight}(X) \leq \text{weight}(W + X), \\
Y = W + X \text{ otherwise, where } X \in C \text{ and } x_1 = 0
\end{array} \right\} \quad (4.7)$$

It is obvious that $\hat{C} \subseteq C$ and $|\hat{C}| = |C| / 2$ since $C$ is a linear code. If $W$ is the all 1's word, then any word in $\hat{C}$ is of weight $\leq n/2$. The encoding of a BCH code usually uses a linear feedback shift register (LFSR) based on the generator polynomial of an equivalent cyclic code. The new code is encoded similarly. The encoding circuitry for the new code will include a counter (modulo at least $n$), a comparator, and some additional registers to store the words $W$, $X$, and $W + X$. If speed is important, two counters must be used in order to compute the weights of $X$ and $W + X$ in parallel. The decoding of this code is same as that of the original code plus one extra step. After the received word is corrected, it is added with the word $W$ only if the leftmost bit of the corrected word is 1. Then the original information sent is given by the 2nd through the $k$th bits.
4.4. Construction II

Let $V^n$ denote the set of all binary vectors of length $n$. Let \( \{ \alpha_1, \alpha_2, \ldots, \alpha_n \} \) be a set of nonzero elements from $\text{GF}(2^r)$ such that $2^r > n$. Consider the mapping

$$ M: V^n \rightarrow \text{GF}(2^r), $$

whose value at $X = x_1 x_2 \cdots x_n$ is

$$ M(X) = \sum_{i=1}^{n} x_i \alpha_i. \quad (4.8) $$

It is well known that this mapping partitions $V^n$ into distance three codes. In particular, if $C = \{ X \in V^n \mid M(X) = 0 \}$, then $C$ is the well known Hamming code.

Using this approach, the set $V^k$ can be made into distance three linear code by appending $r$ extra bits if $k < 2^r-r$. The check bits are assigned an additive basis, i.e. the field elements $1, \alpha, \alpha^2, \ldots, \alpha^{r-1}$ where $\alpha$ is a primitive element. The information bits can be assigned distinct nonzero elements other than those used for the additive basis. If the $i$th information bit is assigned the element $\alpha_i$ then the check is computed as:

$$ \sum_{i=1}^{k} x_i \alpha_i. \quad (4.9) $$

Construction II is systematic and uses two checks which we will refer to as $\text{check1}$ and $\text{check2}$. $\text{check1}$ is used to encode the information into an even parity distance four code. Then $\text{check2}$ is appended to insure that the asymmetric distance of the code is at least 3.
Let the bit positions of the information and check1 be assigned distinct nonzero elements from GF(2^f). Denote these bits for a particular word X by \( x_1x_2 \cdots x_n \), and let \( \alpha_i \) denote the element assigned to the ith position. Let check1 be computed such that

\[
\sum_{i=1}^{n} x_i \alpha_i = 0. 
\]  

(4.10)

Now let check2 be computed as

\[
\text{check2} = \prod_{i=1}^{n} \alpha_i^{x_i}. 
\]  

(4.11)

**Theorem 4.1.** The proposed code can correct up to two asymmetric errors.

**Proof:**

We will show that the asymmetric distance of the code is at least 3. Let \( X, Y \) be two distinct codewords, where \( X = \hat{X}\text{check2}(X) \) and \( Y = \hat{Y}\text{check2}(Y) \). The proposed construction insures that \( d(\hat{X}, \hat{Y}) \geq 4 \). If \( \text{check2}(X) \neq \text{check2}(Y) \) then \( d(X, Y) \geq 5 \) which implies that \( d_A(X, Y) \geq 3 \). If \( \text{check2}(X) = \text{check2}(Y) \) then the only possibility that needs to be ruled out is that of \( \hat{X} \) and \( \hat{Y} \) differing in four coordinates \( i, j, k, l \) as shown:

\[
\begin{array}{cccc}
  & i & j & k & l \\
\hat{X} & \vdots & 1 & 1 & \ldots & 0 & 0 & \ldots \\
\hat{Y} & \vdots & 0 & 0 & \ldots & 1 & 1 & \ldots \\
\end{array}
\]

This, by adding the check equations for \( X \) and \( Y \), would imply that
\[ \alpha_i + \alpha_j = \alpha_k + \alpha_l, \]
\[ \alpha_i \cdot \alpha_j = \alpha_k \cdot \alpha_l. \]

But this means that a second degree polynomial over a field has four distinct roots which is impossible.

4.4.1. Encoding

Calculation of Check 1

For this part we would like to obtain a distance four systematic code of length \( n \) such that with the assignment of distinct nonzero field elements \( \alpha_i \)'s to bit positions, the following condition is satisfied.

\[ \sum_{i=1}^{n} x_i \cdot \alpha_i = 0. \] \hspace{1cm} (4.12)

One way of doing this is by having the check part generate all field elements in both parities. Then, depending on the parity of the information, the appropriate check is assigned so that the overall parity is even and this would insure that the minimum distance is 4. This can be done in GF\( (2^r) \) by using \( r+1 \) elements, \( r \) of these elements constitute an additive basis, and the last element is the sum of any two elements, say the first two, from the chosen basis. Then one parity representation is obtained by using the basis and an opposite parity representation is obtained by setting the last bit to "1" and complementing the first two bits.
This approach of calculating check1 means that the information length can be up to $2^r - r - 2$.

**Calculation of Check2**

For a given information word $X$, let $X_{\text{check1}}(X)$ be $x_1 x_2 \cdots x_n$. To facilitate encoding and decoding check2 will be defined, instead of (4.11), as:

$$
\text{check2} = ( \prod_{i=1}^{n} \alpha_i^{x_i} )^{-1}.
$$

(4.13)

Let the bits in check2 be assigned the multiplicative basis \{ $\alpha^{2^r-1}, ..., \alpha^2, \alpha$ \} where $\alpha$ is a primitive element. The calculation of check2 can be carried out by addition $\text{mod } 2^r-1$ of the log (base $\alpha$) of the field elements assigned to the information and check1 where the corresponding bit is 1. If this value is $S$ then check2 is the binary representation of $-S \text{ mod } 2^r-1$. In fact, this operation is equivalent to doing bit addition and then complementing the final result which gives check2; thus it can be easily implemented in hardware. In this case any codeword (including check2) $X = x_1 x_2 \cdots x_m$ satisfies:

$$
\prod_{i=1}^{m} \alpha_i^{x_i} = 1.
$$

(4.14)

It is to be noted that using a multiplicative basis the element $\alpha^0 = 1$ can be represented as either all 1's or all 0's. The all 0's representation should be used in order to obtain a code with fewer 1's.
Example 4.1 (Encoding)

To get a double asymmetric error correcting code with 4 information bits, we will use GF($2^4$) generated by $\alpha^4 + \alpha + 1 = 0$ [1, p. 85]. Note that the rightmost bit in check1 has been assigned $1 + \alpha = \alpha^4$.

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|}
& \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^2 & \alpha \\
\hline
\text{information} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{check1} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
\text{check2} & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
$$

In encoding "0001", check1 is first calculated as "10100" since $\alpha^8 = 1 + \alpha^2$. This results in an odd parity over the information and check1. So the first (leftmost) two bits in check1 are complemented and the rightmost bit in check1 is set to "1". Then check2 is computed as, summing the powers of $\alpha$ where the corresponding bit is "1", $-(8 + 1 + 2 + 4) \mod 15 = 0$. Thus check2 can be set to either all 0's or all 1's. For check2 of the next information word we have $-(7 + 0 + 1 + 3) \mod 15 = 4$. Thus the position corresponding to $\alpha^4$ in check2 is set to "1" or alternatively check2 can be viewed as the binary representation of 4. The complete code is given in Table 4.1. This code has a minimum nonzero weight = 4. Any symmetric double error correcting code is equivalent to a code with a minimum nonzero weight $\geq 5$.

4.4.2. Decoding

For a received word $X = x_1 x_2 \cdots x_n$, calculate the syndromes:

$$
S_1 = \sum_{i=1}^{n-r} x_i \alpha_i .
$$

(4.15)
Also calculate the parity $P$ over the bits of the information and check1.

$$P = (\sum_{i=1}^{n-r} x_i) \mod 2. \quad (4.17)$$

Then consider the following cases:

1) $S_1 = 0$ and $S_2 = 1$. Declare $X$ to be error free.

2) $S_1 = 0$. If $P = 0$, then the information part of $X$ is error free; otherwise, conclude that more than two asymmetric errors have occurred.

3) $S_1 \neq 0$. If $P = 1$, then complement the position assigned $S_1$ in the information part, if any. If $P = 0$, then at least two errors have occurred, and if the locations of two errors correspond to $\alpha_1$ and $\alpha_2$, then

$$S_1 = \alpha_1 + \alpha_2,$$
$$S_2 = \alpha_1 \alpha_2.$$

Therefore, the $\alpha_i$'s can be determined as the roots of the polynomial

$$f(x) = x^2 - S_1 x + S_2 \quad (\text{in case of } 1 \rightarrow 0 \text{ errors, } S_2^{-1} = \alpha_1 \alpha_2 \text{ and } f(x) = x^2 - S_1 x + S_2^{-1}).$$

If there is no solution then more than two asymmetric errors have occurred; otherwise, complement the leftmost positions assigned $\alpha_1$ and $\alpha_2$.

Example 4.2 (Decoding)
Suppose that the code of Example 4.1 is used where errors are only of type $1 \rightarrow 0$. Suppose that the word "0111 01101 1100" is received. For this word we calculate $S_1 = \alpha^{10}, S_2^{-1} = \alpha^5$ and $P = 0$. Therefore, we have to compute the roots of the polynomial $f(x) = x^2 - \alpha^{10} x + \alpha^5$. This polynomial is factored into $(x - 1)(x - \alpha^5)$. Therefore, the erroneous locations correspond to the positions assigned 1 and $\alpha^5$. Therefore, the corrected information is "1111".

4.4.3. An Alternative Method for Calculating Check1

The purpose of check1 is to encode the information into an even parity distance four code. For a given redundancy $r$, this can be achieved by using all the binary vectors that have "1" in a fixed coordinate. This generates a set of $2^{r-1}$ vectors over GF($2^r$). The basis for such a set, read column-wise, is shown below.

\[
\begin{array}{cccccc}
1 & 1 & \ldots & 1 \\
0 \\
: & L_{r-1} \\
0
\end{array}
\]

The basis is assigned to check1 and the remaining $2^{r-1}-r$ vectors are assigned to the information bits. Check2 is same as before. This approach of calculating check1 means that we will be using a total of $2r$ check bits and can accommodate up to $2^{r-1}-r$ information bits. In the previous method, by using GF($2^{r-1}$), $2r-1$ check bits are used to accommodate $2^{r-1}-(r-1)-2 = 2^{r-1}-r-1$. So the only case in which the new method does better is when the number of information bits is exactly $2^{r-1}-r$ for some integer $r$. 
4.5. Analysis of the Proposed Codes

4.5.1. Construction I

To accommodate \( k \) information bits, the code in Construction I is derived from a BCH code that accommodates \( k + 1 \) information bits. If \( k = 2^r - 2r - 1 \), then we have to use a BCH code that, in general, would use \( 2(r+1) \) check bits. Thus the proposed code uses \( 2r + 3 \) check bits for this case. On the other hand, for the given \( k \) we can use a BCH code that uses \( 2r \) check bits. Therefore, for this particular case the code may not be appropriate because it uses a high amount of redundancy (3 more check bits over BCH) and offers little improvement in the weight distribution. In all other cases, the code uses only one extra check bit and, as can be seen from Table 4.3, offers a considerable improvement in the weight distribution. The table shows that the \( \beta \) parameter (so is the code error probability) is reduced by about 50\%, whereas the \( \gamma \) parameter is reduced by about 15\%, which corresponds to an equal increase in the information rate per photon.

4.5.2. Construction II

In this construction, by using \( \text{GF}(2^r) \), a double asymmetric error correcting code can be designed that uses \( 2r + 1 \) check bits to accommodate up to \( 2^r - r - 2 \) information bits. If the information length is in the range \([2^r - 2r, 2^r - r - 2]\), then a BCH code would require in general \( 2r + 2 \) check bits. Thus the proposed code has twice the information rate as the BCH code in this range. Though Preparata code (a union of
cosets of a linear code) [PREP 68] achieves this even for a wider range, our code is
easier to encode and decode. As we have seen, the encoding can be performed using
only binary addition; likewise, the computation of the syndromes. The location of a
single error is readily determined. In case of double errors, the syndromes directly
give the symmetric functions which in turn determine a quadratic equation whose
roots identify the erroneous locations. This is easier than the BCH code, where the
computation of the second symmetric function is more involved. A summary of the
code redundancy for information length up to 121 is given in Table 4.2.

Because part of the check is never all 1's, this code will in general have fewer
1's than 0's. It is expected that this code will have a better weight distribution than
the BCH code when the information length is in the range \([2^r - 2r, 2^r - r - 2]\), since it
uses fewer check bits. This can be seen from Table 4.3 for [8-10] information bits.
As a matter of fact, even in cases where the code uses more redundant bits, we found
that it has as good weight distribution as some of the less redundant codes. For
example, in reference to Example 4.1, by assigning the information bits the elements
which are sums of three basis elements (i.e. \(\alpha^7, \alpha^{10}, \alpha^{11}, \alpha^{13}\)), we get \(\beta(13,4,2) = 434\)
and \(\gamma(13,4,2) = 92\). This is better than those parameters for the less redundant BCH
code with 4 information bits where we get \(\beta(12,4,2) = 480\) and \(\gamma(12,4,2) = 96\).

4.6. Concluding Remarks

In this chapter, we have given constructions for double asymmetric error
correcting codes. These codes are preferred over the known codes for the symmetric case because they offer a smaller code error probability, and for photon communication, a higher information rate per photon. While having these features, the proposed codes have good information rates and are relatively easy to encode and decode.

Construction I, in general, can be used with any $t$ error correcting linear code. The reward is great when the code is derived from a linear code containing the all 1's word. Most good codes are linear and, if nonshortened, contain the all 1's word. These include Hamming, BCH and Golay codes. In the next chapter we will give constructions for linear codes that have higher information rates than the BCH code in many cases.
<table>
<thead>
<tr>
<th>information</th>
<th>check1</th>
<th>check2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^6$ $a^5$ $a^2$ $a^8$</td>
<td>$1$ $a$ $a^2$ $a^3$ $a^4$</td>
<td>$a^8$ $a^4$ $a^2$ $a$</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1 1 0 1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>1 1 0 1 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>1 0 1 1 1</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>1 1 1 1 1</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>1 0 0 1 0</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 0 1 0 1</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1 0 0 0</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0 1 0 1</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 1 0 0 0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1 1 1 1</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>0 0 0 1 0</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 1 0 1 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>0 0 1 1 1</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 0 0 0 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 0 1</td>
<td>1 1 0 0</td>
</tr>
</tbody>
</table>

**Table 4.1.** The complete code for Example 4.1.
<table>
<thead>
<tr>
<th># of info. bits</th>
<th># of check bits</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8*</td>
<td>same as BCH</td>
</tr>
<tr>
<td>5-7</td>
<td>9</td>
<td>better than BCH</td>
</tr>
<tr>
<td>8-10</td>
<td>9</td>
<td>same as BCH</td>
</tr>
<tr>
<td>11</td>
<td>10*</td>
<td>optimal - Preparata</td>
</tr>
<tr>
<td>12-21</td>
<td>11</td>
<td>same as BCH</td>
</tr>
<tr>
<td>22-25</td>
<td>11</td>
<td>same as BCH</td>
</tr>
<tr>
<td>26</td>
<td>12*</td>
<td>optimal - Preparata</td>
</tr>
<tr>
<td>27-51</td>
<td>13</td>
<td>better than BCH</td>
</tr>
<tr>
<td>52-56</td>
<td>13</td>
<td>same as BCH</td>
</tr>
<tr>
<td>57</td>
<td>14*</td>
<td>same as BCH</td>
</tr>
<tr>
<td>58-113</td>
<td>15</td>
<td>optimal - Preparata</td>
</tr>
<tr>
<td>114-121</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2. A summary of code redundancy for Construction II.

* Using the method of Section 4.4.3 to calculate check1.
<table>
<thead>
<tr>
<th>info.</th>
<th></th>
<th>BCH</th>
<th></th>
<th>Construction I</th>
<th></th>
<th>Construction II</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>check</td>
<td>β</td>
<td>γ</td>
<td>check</td>
<td>β</td>
<td>γ</td>
<td>check</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>480</td>
<td>96</td>
<td>9</td>
<td>245</td>
<td>84</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>1164</td>
<td>208</td>
<td>9</td>
<td>670</td>
<td>184</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>2880</td>
<td>448</td>
<td>9</td>
<td>1305</td>
<td>375</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>7280</td>
<td>960</td>
<td>11</td>
<td>6700</td>
<td>966</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>26080</td>
<td>2304</td>
<td>11</td>
<td>14591</td>
<td>1998</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>62016</td>
<td>4864</td>
<td>11</td>
<td>36460</td>
<td>4253</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>145920</td>
<td>10240</td>
<td>11</td>
<td>89818</td>
<td>9030</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>340480</td>
<td>21504</td>
<td>11</td>
<td>207957</td>
<td>18923</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>788480</td>
<td>45056</td>
<td>11</td>
<td>497260</td>
<td>39912</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>1813504</td>
<td>94208</td>
<td>11</td>
<td>1128358</td>
<td>83181</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4.3. Values of the β and γ parameters.

BCH codes are from (15,7)-code generated by $g(x) = x^8 + x^7 + x^4 + 1$, and (31,21)-code generated by $g(x) = x^{10} + x^8 + x^6 + x^5 + x^3 + 1$, and their shortened ones. Construction II uses GF(2^4) generated by $α^4 + α + 1 = 0$ with the information bits assigned successive powers of $α$ starting from $α^5$, for information length ≤ 10. The other entries use GF(2^6) generated by $α^5 + α^2 + 1 = 0$, and the information bits are assigned successive powers of $α$ starting from $α^5$ (the last bit in checkl is assigned $(1 + α) = α^{18}$).
Chapter 5

New Constructions for Linear Codes

5.1. Introduction

Let $V(n,q)$ denote the vector space of length $n$ over $GF(q)$ for some prime power $q$. A linear code of length $n$ is just a subspace of $V(n,q)$. In particular, a binary code is linear if and only if the sum of any two codeword is a codeword. Linear codes are the most widely studied class of codes because they possess useful properties. Linear codes are systematic. The distance of a linear code is the minimum nonzero weight. The probability of undetectable error for a linear code can be determined from the knowledge of its weight distribution.

A binary linear $(n,k)$ code is a set of $2^k$ of binary $n$-tuples that forms a subspace of $V(n,2)$. The code can be described in terms of a $k \times n$ generator matrix $G$, whose rows are linearly independent and form a basis for the code, or by an $(n - k) \times n$ parity check matrix $H$. In the latter case, the code consists of all binary $n$-tuples that are orthogonal to every row of $H$. The rows of $H$ are linearly independent; thus it is itself is a generator matrix and generates the so-called dual of the code generated by $G$. On the other hand, if any $d-1$ columns of $H$ are linearly independent, then the code with $H$ as a parity check matrix has a minimum nonzero weight (and distance) $= d$. 
A binary linear code of length $n$, $k$ information bits, and distance $d$ is denoted by $[n,k,d]$. The maximum distance achievable by a binary linear code for some fixed $n$ and $k$, denoted by $d_{\text{max}}(n,k)$, is generally unknown, despite a great deal of research on the problem. The problem appears to be difficult and no theoretical approach to a general solution has ever been proposed [HELG 73]. This has prompted the study of good lower and upper bounds on $d_{\text{max}}(n,k)$, and, as a result, a number of such bounds have been reported in the literature [HELG 73, VERH 87]. Upper bounds are usually obtained using some well known bounds such as the Hamming and Plotkin bounds [MACW 77]. Lower bounds are obtained via some general well known constructions such as the BCH code, or other specialized methods and techniques. Our work in this chapter is a further contribution to this latter effort.

In this chapter, we present constructions for $t$ error correcting linear codes, where $t \geq 2$. The first and second constructions given in Sections 5.2 and 5.3 (Constructions I & II) are for double error correcting linear codes. These codes have higher information rates than the BCH code in many cases, and yet they are simple to encode and to decode. Construction I uses $2r+1$ check bits for information length up to $2^r-r-2$ bits, and $2r$ check bits for information length $= 2^{r-1}-r$ bits. Construction II uses $2r+1$ check bits for information length up to $2^r-r-1$ bits. Section 5.4 presents a generalization of Construction II which gives a $t$ error correcting code that would use at most $tr+1$ check bits for information length up to $2^r-(t-1)r-1$ bits. The proposed codes are analyzed in Section 5.5.
5.2. Construction I

Construction I is systematic and uses two checks which we will refer to as check1 and check2. Check1 is used to encode the information into an even parity distance four code. Then check2 is appended to ensure that the distance of the code is at least 5.

Let the bit positions of the information and check1 be assigned distinct nonzero elements from GF(2^r). Denote these bits for a particular word X by x_1x_2 \ldots x_n, and let \alpha_i denote the element assigned to the i-th position. Let check1 be computed such that

\[ \sum_{i=1}^{n} x_i \alpha_i = 0. \] (5.1)

Now let check2 be computed as

\[ \sum_{i=1}^{n} x_i \alpha_i^{-1} \] (5.2)

**Theorem 5.1.** The code given by Construction I can correct up to two errors.

**Proof:**

We will show that the distance of the code is at least 5. Let X, Y be two distinct codewords, where X = \hat{X}check2(X) and Y = \hat{Y}check2(Y). The proposed construction insures that d(\hat{X}, \hat{Y}) \geq 4. If check2(X) \neq check2(Y), then d(X, Y) \geq 5. If check2(X) = check2(Y), then the only possibility that needs to be ruled out is that of \hat{X} and \hat{Y} differing in four coordinates i, j, k, l. This, by adding the check equations
for the two words and rewriting, would imply

\[ \alpha_i + \alpha_j = \alpha_k + \alpha_l \, , \]  

(5.3)

and

\[ \alpha_i^{-1} + \alpha_j^{-1} = \alpha_k^{-1} + \alpha_l^{-1} . \]  

(5.4)

Both (5.3) and (5.4) imply that

\[ \alpha_i \alpha_j = \alpha_k \alpha_l \, . \]  

(5.5)

Both (5.3) and (5.5) imply that a second degree polynomial over a field has four distinct roots which is impossible.

**Theorem 5.2.** The code given by Construction I is linear.

**Proof:**

Let \( X, Y \) be two information symbols of length \( k \). Since the code is systematic, \( X+Y \) is also an information symbol. Let \( \hat{X} = X \text{check}_1(X) = x_1 \cdots x_k \cdots x_n \) and similarly let \( \hat{Y} = Y \text{check}_1(Y) \). We have to show

1. \( \text{check}_1(X+Y) = \text{check}_1(X) + \text{check}_1(Y) \),
2. \( \text{check}_2(\hat{X}+\hat{Y}) = \text{check}_2(\hat{X}) + \text{check}_2(\hat{Y}) \).

For (1):

\[ \text{check}_1(X+Y) = \sum_{i=1}^{k} (x_i + y_i) \alpha_i = \sum_{i=1}^{k} x_i \alpha_i + \sum_{i=1}^{k} y_i \alpha_i = \text{check}_1(X) + \text{check}_1(Y). \]

For (2):
check2(\hat{X} + \hat{Y}) = \sum_{i=1}^{n} (x_i + y_i) \alpha_i^{-1}

= \sum_{i=1}^{n} x_i \alpha_i^{-1} + \sum_{i=1}^{n} y_i \alpha_i^{-1} = check2(\hat{X}) + check2(\hat{Y}).

5.2.1. Encoding

Calculation of Check1

This is the same as given in the previous chapter (4.3.1). The check part generates all field elements in both parities. Then, depending on the parity of the information, the appropriate check is assigned so that the overall parity is even, and this would insure that the minimum distance is 4. This can be done in GF(2^r) by using \(r+1\) elements, \(r\) of these elements constitute an additive basis, and the last element is the sum of two elements, say the first two, from the chosen basis. Then one parity representation is obtained by using the basis and an opposite parity representation is obtained by setting the last bit to 1 and complementing the first two bits.

This approach of calculating check1 means that the information length can be up to \(2^r - r - 2\).

Calculation of Check2

For a given information word \(X\), let \(Xcheck1(X) = x_1 x_2 \cdots x_n\).

\[
check2 = \sum_{i=1}^{n} x_i \alpha_i^{-1}. \quad (5.6)
\]
Thus check2 uses \( r \) bits which are assigned an additive basis. In this case any codeword (including check2) \( X = x_1 x_2 \cdots x_n \) satisfies:

\[
\sum_{i=1}^{n-r} x_i \alpha_i^{-1} + \sum_{i=n-r+1}^{n} x_i \alpha_i = 0.
\] (5.7)

**Example 5.1 (Encoding)**

To get a double error correcting code with 4 information bits, we will use \( \text{GF}(2^4) \) generated by \( \alpha^4 + \alpha + 1 = 0 \) [MACW 77, p. 85]. Note that the rightmost bit in check1 has been assigned \( 1 + \alpha = \alpha^4 \).

<table>
<thead>
<tr>
<th>Information</th>
<th>Check1</th>
<th>Check2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha^0 )</td>
<td>( \alpha^0 )</td>
<td>( \alpha^0 )</td>
</tr>
<tr>
<td>( \alpha^1 )</td>
<td>( \alpha^1 )</td>
<td>( \alpha^1 )</td>
</tr>
<tr>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
<td>( \alpha^2 )</td>
</tr>
<tr>
<td>( \alpha^3 )</td>
<td>( \alpha^3 )</td>
<td>( \alpha^3 )</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1 1 0</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>1 1 0 1</td>
<td>0 1 0 0</td>
</tr>
</tbody>
</table>

In encoding "0001", check1 is first calculated as "10100" since \( \alpha^8 = 1 + \alpha^2 \). This results in an odd parity over the information and check1. So the first two bits in check1 are complemented and the rightmost bit in check1 is set to "1". Then check2 is computed as, summing the multiplicative inverses of the \( \alpha \)'s where the corresponding bit is "1", \( \alpha^7 + \alpha^{14} + \alpha^{13} + \alpha^{11} = 1 \). The complete code is given in Table 5.1.

**5.2.2. Decoding**

For a received word \( X = x_1 x_2 \cdots x_n \), calculate the syndromes:

\[
S_1 = \sum_{i=1}^{n-r} x_i \alpha_i
\] (5.8)
\[ S_2 = \sum_{i=1}^{n-r} x_i \alpha_i^{-1} + \sum_{i=n-r+1}^{n} x_i \alpha_i \]  

(5.9)

Also calculate the parity \( P \) over the bits of the information and check1.

\[ P = (\sum_{i=1}^{n-r} x_i) \mod 2. \]  

(5.10)

Then consider the following cases:

1) \( S_1 = S_2 = 0 \). Declare \( X \) to be error free.

2) \( S_1 = 0 \). If \( P = 0 \), then the information part of \( X \) is error free; otherwise, conclude that more than two errors have occurred.

3) \( S_1 \neq 0 \). If \( P = 1 \), then complement the position assigned \( S_1 \) in the information part, if any. If \( P = 0 \), then at least two errors have occurred in the information and check1 combined and if the locations of two errors correspond to \( \alpha_1 \) and \( \alpha_2 \), then

\[ S_1 = \alpha_1 + \alpha_2, \]

\[ S_2 = \alpha_1^{-1} + \alpha_2^{-1}. \]

These two equations imply that

\[ S_1 / S_2 = \alpha_1 \alpha_2 \]

Therefore, the \( \alpha_i \)'s can be determined as the roots of the polynomial \( f(x) = x^2 - S_1 x + S_1 / S_2 \). If there is no solution, then more than two errors have occurred; otherwise, complement the leftmost positions assigned \( \alpha_1 \) and \( \alpha_2 \).
Example 5.2 (Decoding)

Suppose that the code of Example 5.1 is used and that the word "0111 01101 0001" is received. For this word we calculate

\[ S_1 = \alpha^6 + \alpha^7 + \alpha^8 + \alpha + \alpha^2 + \alpha^4 = \alpha^{10}, \]
\[ S_2 = \alpha^9 + \alpha^8 + \alpha^7 + \alpha^{14} + \alpha^{13} + \alpha^{11} + \alpha^3 = \alpha^5, \]

and \( P = 0 \). Therefore, we have to compute the roots of the polynomial \( f(x) = x^2 - \alpha^{10} x + \alpha^5 \). This polynomial is factored into \((x - 1)(x - \alpha^5)\). Therefore, the erroneous locations correspond to the positions assigned 1 and \( \alpha^5 \). Therefore, the corrected information is "1111".

5.2.3. An Alternative Method for Calculating Check1

The purpose of check1 is to encode the information into an even parity distance four code. For a given redundancy \( r \), this can be achieved by using all the binary vectors that have "1" in a fixed coordinate. This generates a set of \( 2^r - 1 \) vectors over GF(2\(^r\)). The basis for such a set, read column-wise, is shown below.

\[
\begin{array}{cccc}
1 & 1 & \ldots & 1 \\
0 \\
\vdots & \vdots \\
0 \\
\end{array}
\]

The basis can be assigned to check1 and the remaining \( 2^r - 1 - r \) vectors can be assigned to the information bits. Check2 is same as before. This approach of calculating check1 means that we will be using a total of \( 2^r \) check bits and accommodate up to \( 2^r - 1 - r \) information bits. In the previous method, by using GF(2\(^{r-1}\)), we can
use $2r - 1$ checks and accommodate $2^{r-1} - (r - 1) - 2 = 2^{r-1} - r - 1$. So the only case in which the new method does better is when the number of information bits is exactly $2^{r-1} - r$ for some integer $r$.

5.3. Construction II:

This construction is a slight variation of Construction I. The construction is systematic and uses two checks. Check1 uses $r$ bits and is used to encode the information into distance three Hamming code. Check2 uses $r + 1$ bits and is added to insure that the resulting code has a distance of 5. The rightmost bit of check2 is a parity bit over check2.

Let the bit positions of the information and check1 be assigned distinct nonzero elements from $GF(2^r)$. Denote these bits for a particular word $X$ by $x_1x_2 \cdots x_n$, and let $\alpha_i$ denote the element assigned to the $i$th position. Let check1 be computed such that

$$\sum_{i=1}^{n} x_i \alpha_i = 0. \quad (5.11)$$

Now let check2 be computed as

$$\text{check2} = \sum_{i=1}^{n} x_i \alpha_i^3 \quad (5.12)$$

**Theorem 5.3.** The code given by Construction II can correct up to two errors.

**Proof:**
We will show that the distance of the code is at least 5. Let $X, Y$ be two distinct codewords, where $X = \hat{X} \text{check}_2(X)$ and $Y = \hat{Y} \text{check}_2(Y)$. The proposed construction insures that $d(\hat{X}, \hat{Y}) \geq 3$. If $\text{check}_2(X) \neq \text{check}_2(Y)$, then $d(X, Y) \geq 5$. If $\text{check}_2(X) = \text{check}_2(Y)$, then we need to rule out that $X$ and $Y$ differ in three or four coordinates. If $X$ and $Y$ differ in three coordinates $i, j, k$. Then by adding the check equations for $X$ and $Y$ and rewriting, we get

$$\alpha_i + \alpha_j = \alpha_k \tag{5.13}$$

and

$$\alpha_i^3 + \alpha_j^3 = \alpha_k^3 \tag{5.14}$$

By cubing (5.13) we get

$$\alpha_i^3 + \alpha_i \alpha_j (\alpha_i + \alpha_j) + \alpha_j^3 = \alpha_k^3,$$

which, by using (5.14), would simplify to

$$\alpha_i \alpha_j (\alpha_i + \alpha_j) = 0 \tag{5.15}$$

But this is not possible since $\alpha$'s are distinct and nonzero.

The other possibility that must be ruled out is that $X$ and $Y$ differ in four coordinates, say $i, j, k, l$. In this case, by adding the check equations for $X$ and $Y$ and rewriting, we get

$$\alpha_i + \alpha_j = \alpha_k + \alpha_l \tag{5.13-a}$$

and

$$\alpha_i^3 + \alpha_j^3 = \alpha_k^3 + \alpha_l^3 \tag{5.14-a}$$
By cubing (5.13-a) we get

\[ \alpha_i^3 + \alpha_i \alpha_j (\alpha_i + \alpha_j) + \alpha_j^3 = \alpha_k^3 + \alpha_k \alpha_l (\alpha_k + \alpha_l) + \alpha_l^3 , \]

which, by using (5.13-a) and (5.14-a), would simplify to

\[ \alpha_i \alpha_j = \alpha_k \alpha_l . \] (5.15-a)

The above equation and (5.13-a) imply that a second degree polynomial over a field has four distinct roots, which is impossible.

**Theorem 5.4.** The code given by Construction II is linear.

**Proof:** Similar to that of Theorem 5.2.

**5.4. Construction III:**

Construction II is a slight variation of the BCH double error correcting code. The same idea can be applied to a t-error correcting BCH code. Recall that a t-error correcting BCH binary code of length \( n = 2^r - 1 \) has a parity check matrix \( H \) of the form

\[
H = \begin{bmatrix}
\alpha_1 & \alpha_2 & \cdots & \alpha_n \\
\alpha_1^3 & \alpha_2^3 & \cdots & \alpha_n^3 \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_1^{2t-1} & \alpha_2^{2t-1} & \cdots & \alpha_n^{2t-1}
\end{bmatrix}, \tag{5.16}
\]

where \( \alpha_i \)'s are distinct nonzero elements from \( GF(2^r) \).

In Construction III, we get a \( t \) error correcting code \( C \) by using a \( (t-1) \) error correcting code \( C_1 \) based on \( GF(2^r) \) and appending an extra check of even weight.
This check uses \( r+1 \) bits and is calculated for a word \( X = x_1x_2 \cdots x_n \in C_1 \) by
\[
\sum_{i=1}^{n} x_i \alpha_i^{2t-1}.
\] (5.17)

**Theorem 5.5.** The code given by Construction III is linear and can correct up to \( t \) errors.

**Proof:** Those words which agree on the added check will be a \( t \) error correcting BCH code. Any two words which differ in the last check will have a distance \( \geq 2 + 2(t-1) + 1 = 2t+1 \). The proof of linearity is similar to the one given for Theorem 5.2.

Often a BCH code will have a an actual distance that is higher than its designed distance. This fact can be exploited here by replacing \( 2t-1 \) in (5.17) by the first integer not accounted for by the cyclotomic cosets used in the construction of \( C_1 \). Note that the code \( C \) resulting from Construction III will have a distance \( d(C) \geq \min( d(C_1) + 2, d(C^*) ) \), where \( C^* \) is the BCH code with the parity check matrix as that of \( C_1 \) extended by the extra row specified by (5.17).

**5.5. Analysis of the Proposed Codes**

**5.5.1. Construction I :**

Using \( GF(2^r) \), Construction I uses \( 2r+1 \) check bits for information length up to \( 2^r-r-2 \) bits, and \( 2r \) check bits for information length = \( 2^r-1-r \) bits. If the information length is in the range \( [2^r-2r, 2^r-r-2] \), then this code uses \( 2r+1 \) check bits,
while the BCH code would require in general $2r + 2$ checks. Thus this code has twice the information rate as the BCH code in this range. Though *Preparata* code [PREP 68] achieves this even for a wider range, the proposed code is linear and easier to encode and decode. As we have seen, the encoding is done using only binary addition; likewise the computation of the syndromes. The location of a single error is readily determined. In case of double errors, the syndromes directly give the symmetric functions of the error locations. This is easier than BCH, where the second symmetric function has to be computed from the syndromes. Table 5.2 summarize the code redundancy for information length up to 121 bits.

5.5.2. Constructions II & III:

The bounds obtained from these constructions are summarized in Table 5.3. Most of these match the best known bounds given in [HELG 73, MACW 77, VERH 87] even after being shortened several times. Note that the entry for $r = t = 5$ results from the fact that a BCH code over $\text{GF}(2^5)$ of a designed distance $= 9$, has an actual distance $= 11$ [MACW 77, p. 205]. In this case, the extra check is computed using 11 as a power of the $\alpha$'s. A similar analysis applies to the entry for $r = 5$ and $t = 6$. 
<table>
<thead>
<tr>
<th>information</th>
<th>check1</th>
<th>check2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^0$</td>
<td>$\alpha^5$</td>
<td>$\alpha^7$</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1 1 0 1</td>
<td>1 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>1 1 0 1 0</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>1 0 1 1 1</td>
<td>1 1 0 0 0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>1 1 1 1 1</td>
<td>0 1 1 1 1</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>1 0 0 1 0</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 0 1 0 1</td>
<td>0 0 1 1 1</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1 0 0 0</td>
<td>1 0 1 1 1</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0 1 0 1</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 1 0 0 0</td>
<td>0 0 1 0 0</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1 1 1 1</td>
<td>1 1 1 0 0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>0 0 1 0 0</td>
<td>0 1 1 1 0</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 1 0 1 0</td>
<td>1 1 0 1 0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>0 0 1 1 1</td>
<td>0 1 0 1 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 0 0 0 0</td>
<td>1 0 0 1 0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 0 1</td>
<td>0 0 0 1 0</td>
</tr>
</tbody>
</table>

**Table 5.1.** The complete code for Example 5.1.
<table>
<thead>
<tr>
<th>#of info. bits</th>
<th>#of check bits</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8*</td>
<td>same as BCH</td>
</tr>
<tr>
<td>5-7</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8-10</td>
<td>9</td>
<td>better than BCH</td>
</tr>
<tr>
<td>11</td>
<td>10*</td>
<td>same as BCH</td>
</tr>
<tr>
<td>12-21</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>22-25</td>
<td>11</td>
<td>optimal - Preparata</td>
</tr>
<tr>
<td>26</td>
<td>12*</td>
<td>same as BCH</td>
</tr>
<tr>
<td>27-51</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>52-56</td>
<td>13</td>
<td>better than BCH</td>
</tr>
<tr>
<td>57</td>
<td>14*</td>
<td>same as BCH</td>
</tr>
<tr>
<td>58-113</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>114-121</td>
<td>15</td>
<td>optimal - Preparata</td>
</tr>
</tbody>
</table>

Table 5.2. A summary of code redundancy for Construction I.
* Using the method of Section 5.2.3 to calculate check1.
<table>
<thead>
<tr>
<th>$r/t$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>[11,4,5]*</td>
<td>[11,1,10]*</td>
<td>[20,7,9]*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[20,11,5]*</td>
<td>[20,7,7]*</td>
<td>[20,7,9]*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>[37,26,5]*</td>
<td>[37,21,7]*</td>
<td>[37,16,9]</td>
<td>[37,11,13]*</td>
<td>[37,8,17]*</td>
</tr>
<tr>
<td>6</td>
<td>[70,57,5]*</td>
<td>[70,51,7]*</td>
<td>[70,45,9]</td>
<td>[70,39,11]</td>
<td>[70,33,13]</td>
</tr>
<tr>
<td>7</td>
<td>[135,120,5]*</td>
<td>[135,113,7]*</td>
<td>[135,106,9]*</td>
<td>[135,99,11]*</td>
<td>[135,92,13]*</td>
</tr>
</tbody>
</table>

**Table 5.3.** $[n,k,d]$ values obtained from Constructions II & III. Entries marked with * match the best known bounds given in [MACW 77, VERH 87].
Chapter 6

Conclusion

6.1. Summary

In this research we have studied several problems concerned with finding good lower and upper bounds for different kinds of codes. These codes have a combination of asymmetric/unidirectional/symmetric error detection/correction capabilities. In Chapter 2 we introduced $t$-unordered codes, which are a generalization of unordered codes. A $t$-unordered code can correct $t-1$ errors and simultaneously detect all unidirectional errors. We derived some bounds on the size of these codes and established the exact bound in some specific cases. We also gave a method for constructing constant weight distance four codes. These give rise to 2-unordered codes which are of practical interest.

In Chapter 3 we investigated some of the properties of asymmetric/unidirectional error detecting/correcting codes. We proved that any $t$ asymmetric error correcting code can be made into $t$ unidirectional error correcting code by appending three or fewer bits. Also in this chapter, we gave a method for constructing single asymmetric error correcting codes and established several new lower bounds.

We presented constructions for double asymmetric error correcting codes in Chapter 4. These codes are preferred over the known double symmetric error
correcting codes because they offer a smaller code error probability, and for photon communication, a higher information rate per photon. While having these features, the proposed codes have good information rates and are relatively easy to encode and decode.

In Chapter 5 we gave new constructions for linear codes. The first two constructions given are for double error correcting linear codes. These codes have higher information rates than the BCH code in many cases, and yet they are easier to decode. The first construction uses $2r+1$ check bits for information length up to $2^r-r-2$, and $2r$ check bits for information length $= 2^{r-1}-r$. The second construction uses $2r+1$ for information length $= 2^r-r-1$. A generalization of the latter construction gives a $t$ error correcting code that would use at most $tr+1$ check bits for information length up to $2^r-(t-1)r-1$ bits.

6.2. Future Research

The problem of finding good bounds on the size of a code has been and will continue to be a difficult research problem. The use of linear programming has proven to be successful in obtaining good bounds for $A(n,d)$ and $A(n,d,w)$ [MACW 77]. In this case the upper bound is actually a complete linear program, rather than a simple closed formula. The linear program incorporates all kinds of relevant constraints one can think of. To develop a linear program for $U(n,t)$, we would start with a linear program for $A(n,2t)$. Further constraints are then added to this linear program. The
additional constraints will include the bounds we developed in Chapter Two. It is expected that this approach will yield good bounds, especially when \( n \) is much larger than \( t \).

The other important question that needs to be addressed further is, "Are there practical asymmetric error control codes that are better than symmetric error control codes?". For error detection, the answer is yes, indeed. As we have seen, an unordered code can detect any number of asymmetric errors, while a code that can detect \( t \) symmetric errors must have a distance of \( t+1 \). On the other hand, for error correction the answer is not fully known. For single error correction, we know of only one case where there exists a systematic asymmetric error correcting code that uses one less check bit than the Hamming code (see Chapter 3). However, such a code has been generated by computer search, and the encoding/decoding of this code may not be done any easier than by table-lookup. In this thesis, the double systematic asymmetric code that was constructed in Chapter 4 (Section 4.3) gave rise to a similar systematic symmetric error correcting code (5.2), thus giving support to the statement, "The best systematic asymmetric error correcting codes are the systematic symmetric error correcting codes". It would be of great value to find a class of asymmetric error correcting codes that are as general as cyclic codes, but ones that are easier to decode.
Bibliography


