Single Asymmetric Error Correcting Codes:

Improved Code Size

By

Raed Yacoub Radwan Shammas

May 2011
Single Asymmetric Error Correcting Codes:
Improved Code Size

BY
Raed Yacoub Radwan Shammas

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
COMPUTER SCIENCE

May 2011
This thesis, written by RAED YACOUB RADWAN SHAMMAS under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN COMPUTER SCIENCE.

Thesis Committee

Dr. Sultan Almuhammadi (Thesis Advisor)

Dr. Mohammad H. Alsuwaiyel

Dr. Farag Azzedin

Dr. Adel Ahmed
Department Chairman

Dr. Salam Zummo
Dean of Graduate Studies

Date: 26/6/11
Dedicated to

My PARENTS,

ADVISOR, BROTHERS

and SISTERS
ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Sultan Almuhammadi, for his guidance, support, encouragement, and patience throughout my graduate study. Sincere thanks go to both the members of my advisors committee, Dr. Mohammad H. Aliuwaitel and Dr. Farag Azzedin for their constructive guidance and technical support.

Special thanks are due to my senior colleagues at the university, who were always there to provide thoughtful solutions to the various problems encountered in my research. I would also like to thank all my friends, for a wonderful company and good memories that will last a life time.

Finally, thanks are due to my parents, my sisters and brothers as well as all my family members for their emotional and moral support throughout my academic career and also for their love, forbearance, encouragement and prayers.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ................................................................................................. IV

TABLE OF CONTENTS ................................................................................................. V

LIST OF TABLES ............................................................................................... VII

LIST OF FIGURES ........................................................................................... VIII

THESIS ABSTRACT ........................................................................................ IX

Chapter 1 Introduction ............................................................................................. 1

1.2 Preface ........................................................................................................... 1

1.3 Motivation ..................................................................................................... 4

1.4 Objectives ..................................................................................................... 5

1.5 Contributions ................................................................................................. 5

1.6 Thesis Outlines and Organizations ............................................................... 6

Chapter 2 Background of ASEC Codes .................................................................. 8

2.1 An Overview ................................................................................................. 8

2.2 The Z-Channel .............................................................................................. 9

2.3 Codes ........................................................................................................... 11

2.4 Asymmetric Distance and Hamming Distance ........................................... 12

2.5 Construction Method .................................................................................. 16

2.5.1 Cartesian Product Construction Method .............................................. 16
2.5.2 B-Partitions ................................................................. 23

2.5.3 A-Partitions ................................................................. 25

Chapter 3 Literature Review .................................................. 35

Chapter 4 New Single Asymmetric Error Correcting Codes and A-Partitions .... 42

4.1 Improving A-Partitions ..................................................... 43

4.2 The Proposed Codes ......................................................... 49

Chapter 5 Conclusion and Future Work .................................... 53

5.1 Conclusion ................................................................. 53

5.2 Future Work ................................................................. 54

References ........................................................................... 56

Vita ...................................................................................... 59
LIST OF TABLES

Table 2.1 A single asymmetric error correcting code for dimension = 6. ............ 22
Table 2.2 B-partitions, even vector classes with hamming distance 4 .............. 24
Table 2.3 A-partitions using Abelian group partitioning (Γp)............. 27
Table 2.4 Improved A-partition using Cartesian product method [2] .......... 30
Table 3.1 Existing single asymmetric error correcting codes.................. 38
Table 4.1 New single Asymmetric error correcting Codes....................... 44
Table 4.2 New size of A-partitions for $p = 6, 7$ ......................................... 49
Table 4.3 A-partition for $p = 6$.............................................................. 50
Table 4.4 A-partition for $p = 7$.............................................................. 51
Table 4.5 Proposed codes with the dimensions of A- and B-partitions........ 52
LIST OF FIGURES

Figure 1.1 A typical data transmission or storage system ........................................ 2
Figure 2.1 The binary asymmetric channel (Z-Channel) ..................................... 10
Figure 2.2 Constructing A-partition for p = 6 with rotation ............................. 30
Figure 2.3 Constructing an A-partition without rotation ................................ 31
Figure 2.4 Graph coloring method for Example 2.5 ....................................... 33
THESIS ABSTRACT

NAME: Raed Yacoub Radwan Shammas

TITLE: Single Asymmetric Error Correcting Codes: Improved Code Size

MAJOR: Information Technology and Computer Sciences

DATE: May 2011

Error correcting codes have been studied extensively since 1950’s in the field of information theory. Such codes are used in storage devices and digital data transmission systems to increase data reliability. This thesis studies binary asymmetric error-correcting codes on $Z$-channel. Failure in such channels normally affects 1’s in digital data and rarely affects 0’s.

Previous research on asymmetric error correcting codes has given upper bounds and provided several construction methods to improve the lower bounds. However, these lower bounds are still much less than the upper bounds, which motivates the research in constructing new codes and improving the lower bounds.

This thesis proposes new single asymmetric error correcting codes with improved code sizes. The construction method of the proposed codes is based on the Cartesian product of two sets of partitioned codes of smaller dimensions. Some useful partitions for the construction method were obtained in this thesis. These partitions were used to construct new codes of dimensions 14, 15, 16, 17 and 19, and improve the sizes of the existing codes for these dimensions.

Master of Science Degree

King Fahd University of Petroleum and Minerals

Dhahran, Saudi Arabia

May 2011
ملخص الرسالة

الاسم: رائد عقوق رضوان شماس

العنوان: شيفرات جديدة لتصحيح الخطأ المتناطر: تحسين حجم الكود

التخصص: تكنولوجيا المعلومات وعلوم الحاسب الآلي

التاريخ: رجب 1432 هـ

بدأت دراسة علم الشيفرات وانتشار الأخطاء وتصحيحها منذ خمسينيات القرن الماضي، وبعد علم
الشيفرات وانتشار الأخطاء وتصحيحها من أحد فروع علم نظرية المعلومات، وهو يعني بكيفية تمثيل
البيانات على أجهزة التخزين وفي قنوات الاتصال لهدف تحقيق مصداقية واعتمادية أكبر لتلك البيانات
и أنظمتها، وهذه الرسالة تبحث في نظم الشيفرات وانتشار الأخطاء وتصحيحها لقنوات الرقمية غير
المتماثلة، حيث يفترض في هذا النوع من القنوات حدوث أخطاء غير متماثلة فقط.

توجد أبحاث كثيرة في حل الترميز وانتشار الأخطاء وتصحيحها لقنوات غير المتماثلة، منها ما أنجز
تحديد الحد الأعلى لتلك الشيفرات. كما يوجد عدد أخر من الأبحاث لتواليد شرائح لاكتشاف الأخطاء غير
المتماثلة مما يعني الحد الأدنى لهذه الشيفرات. ولكن الحد الأدنى لهذه الشيفرات المولد به بعيد نوعا ما
عن الحد الأعلى، مما يتيح الفرصة لمزيد من البحث حول توليد شرائح جديدة وتحسين الحد الأدنى لها.

هذه الرسالة تقدم شرائح بحجم يفوقة الشريكات الموجودة من قبل لقنوات غير المتماثلة. وتعتمد طريقة
تكوين هذه الشيفرات على عملية الضرب الكارتيزي لجزء من مقامات الشريكات ذات أطوال أصغر،
и تعرض مقدارا مفيدا لعملية الضرب الكارتيزي. وقد استخدمت هذه المقدار لتواليد شرائح جديدة لكل
من الأطوال التالية: 14، 15، 16، 17، 19، تفوق الشريكات الموجودة لهذه الأطوال من قبل.

درجة الماستر في العلوم

جامعة الملك فهد للبترول و المعادن

الظهران-المملكة العربية السعودية

رجب 1432 هـ
Chapter 1

Introduction

1.2 Preface

Recently, there has been an increasing demand for efficient and reliable digital storage systems and data transmission. This demand has been accelerated by the appearance of high-speed, large scale data networks for the exchange, storage of
digital data and processing in the research agency, educational, commercial, health and governmental applications [1].

The storage and transmission of digital data have much in common. Both activities transfer data from a source of information to a destination [1]. These systems can be represented as in the following block diagram (Figure 1.1).

![Figure 1.1 A typical data transmission or storage system](image)

Computer data should always remain correct in all sorts of processing, i.e. when it is written into memory or output device, stored, read from memory or input device, communicated and manipulated. The growing complexity of new computers makes
it very impractical to depend on reliability of components and devices for reliable operations. Some redundancy is needed to ensure the detection and/or correction of errors which invariably occur as information is being stored, transferred or manipulated. In Figure 1.1, blocks in dashed-lines represent the stages where error correcting codes are used.

An extensive theory of error-control coding has been developed since the 1950's. The primary emphasis of this theory is the design of reliable communication systems. The problem of reliable computation differs significantly from the problem of reliable communication. For example, communication error control schemes usually assume a perfectly reliable computing and processing at the transmitter and receiver, and have less severe restraints on computation time for error correction. In addition, these schemes are subjected to different statistics of error occurrences than those that occur in computer systems. The principles that have been discovered by communication coding theorists are so fundamental that they are also basic to the understanding and design of error control for reliable computation.

This thesis studies error-control codes that are suitable for especial kind of channels, called Z-Channel (explained in Section 2.2), capable of correcting a single asymmetric error. It proposes new single asymmetric correcting error codes with more codewords than the existing codes. In the past few decades, most of the research took place under the assumption of symmetric errors. Unlike symmetric
errors, the issues of asymmetric errors have not been well studied in literature yet, in spite of the extensive research done so far. The class of asymmetric error correcting codes was considered recently in the theory of error control coding. This thesis is an extension of the research done in this area.

In order to use a code, it should be first constructed. Therefore, the construction method of the proposed codes is explained in this thesis. The basic idea of the construction method of the proposed codes is to form the code using the Cartesian product of two sets of smaller partitioned codes. The method is quite sensitive to the sizes of the smaller partitions [2]. Indeed, the better partitions used in this method, the more codewords constructed. Therefore, the issue of obtaining better partitions is considered in this thesis. Moreover, a useful method introduced to make partitions of small codes to construct other larger codes, and new useful partitions are obtained using this method.

1.3 Motivation
The field of error control coding system in coding theory has gained the interest of researchers in various aspects. Some of those aspects are: data rate in the codeword versus the redundancy bit which affects the number of codewords in a code, the capability of correcting errors in the codeword, construction of the code, and how to encode and decode data.
Various problems exist in the field of information theory of error correcting code. The problem of maximizing the size of single asymmetric error correcting codes is open. Most of the existing single asymmetric error correcting codes are of sizes much lower than the established upper bounds. This motivates us to develop new single asymmetric error correcting codes with better sizes and improve the existing lower bounds.

1.4 Objectives

1. Study the Cartesian product construction method, and the relation between the size of the code and the size of the $A$-partition and $B$-partition used in the construction.

2. Design a heuristic algorithm to form new $A$-partitions, which improves the existing ones.

3. Apply the proposed algorithm to generate $A$-partitions of dimensions 6 and 7, and improve the sizes of the existing codes.

1.5 Contributions

The main consideration of this proposed work is to “propose new single asymmetric error correcting codes and improve the exiting lower bounds”.

The contributions of this work include:
1. Improving the code size of a single asymmetric error correcting code of dimensions 14, 15, 16, 17 and 19.

2. Constructing the proposed codes for the above dimensions.


**1.6 Thesis Outlines and Organizations**

Thesis outlines are summarized as follows:

1. Define the Asymmetric Error Correcting Codes.

2. Review the existing single asymmetric error correcting codes.

3. Study the construction method of single asymmetric error correcting codes.

4. Propose a new method to improve the A-partitions used in construction method.

5. Propose new single asymmetric error correcting codes using Cartesian product method.

This thesis is organized as follows: In Chapter 2, the asymmetric error correcting code is reviewed. Preliminaries and definitions on single asymmetric error correcting code are introduced. The Cartesian product method for constructing single asymmetric error correcting codes is described, and the concept of code partitioning is discussed. In Chapter 3, a summary of the existing single asymmetric error correcting codes is given, and previous techniques for forming and constructing
asymmetric error correcting codes are explored. In Chapter 4, new codes are proposed and the construction is explained. The sizes and the dimensions of the constructed codes are summarized and compared to the existing codes. Finally, conclusion and future work are given in Chapter 5.
Chapter 2

Background of ASEC Codes

2.1 An Overview

When digital data is stored on storage device or transmitted over a channel, it is important to have a mechanism that allows detecting and correcting possible errors. In general, digital data contains blocks of 0’s and 1’s known as bits. Each block is encoded by adding a number of extra bits (redundancy bits). When data is retrieved
from a storage device or received from a sender, the original data block should be reconstructed (decoding process). In general, decoding process scenario has two stages: error detection and error correction. Error detection checks a possible corruption in data. Whereas error correction makes a decision to correct the error if possible and extract the original data block, or declare that the data is corrupted.

The set of all possible messages (codewords) that can be encoded in order to be corrected later is called an error-correcting code. The field of error correcting codes has begun since 1940’s by the work of Shannon and Hamming, and since then, extensive research in this area has been conducted.

There are two types of errors in the media of storage/channel: symmetric errors and asymmetric errors. The scope of this thesis concentrates on single asymmetric errors.

2.2 The Z-Channel

In many digital communication systems, the probabilities of the crossovers $0 \rightarrow 1$ and $1 \rightarrow 0$ are approximately the same, and the systems are well modeled by the binary symmetric channel (BSC) error correcting codes. The BSC’s have been studied extensively.

In some communication systems, the probability of a $1 \rightarrow 0$ crossover is much larger than the probability of a $0 \rightarrow 1$ crossover. Examples of such systems include: data storing devices, and optical communication systems. Neglecting the low
probability $0 \rightarrow 1$ crossover, that system is modeled by the Z-channel (Figure 2.1). Labels on arrows represented the probability of crossover between states. $p$ here represents the probability of $1 \rightarrow 0$ error. Error correcting codes for the Z-channel have been studied recently compared to the research done on the BSC codes.

Definition 2.1

The binary asymmetric channel (the Z-channel) is a channel with \{0, 1\} as input and output alphabets, where the error $1 \rightarrow 0$ occurs with positive probability $p$, whereas the $0 \rightarrow 1$ error never occurs [3].

Positive probability $p$ here is assumed to be very low so that it is highly unlikely to have two errors in the same codeword, i.e. the probability of two errors $= 1 / p^2$ can be neglected.
Interchanging the position of “1” and “0” (complementation) we get a complementary Z-channel. Any complementation of code for the Z-channel gives a code with the same properties for the complementary channel. However, it turns out that a code for the Z-channel will be a code with the same error correcting capabilities for the complementary Z-channel also without complementation.

Definition 2.2

Let \( x = (x_1, x_2, \ldots, x_n) \), \( y = (y_1, y_2, \ldots, y_n) \) and \( x_i, y_i \in \{0, 1\} \), the number of positions where \( x \) has a 1 and \( y \) has a 0 is defined by:

\[
N(x, y) = |\{i : x_i = 1 \text{ and } y_i = 0\}| \quad [3].
\]

Definition 2.3

For \( x = (x_1, x_2, \ldots, x_n) \), and \( x_i \in \{0, 1\} \), \( w(x) \) is known as the Hamming Weight of \( x \), is the count of 1’s in \( x \),

\[
w(x) = |\{i : x_i = 1\}| \quad [3].
\]

The hamming weight of \( x \) can be defined as \( N(x, y) \) where \( y = (y_1, y_2, \ldots, y_n) \), and \( y_i = 0 \), using Definition 2.2.

### 2.3 Codes

A code is a rule for converting block of information into another representation, and the new representation of data is used in a way that is more resistant to errors in
storage/transmission medium. That code is called an error-correcting code (or error control code), and usually it works by adding redundant signals (bits) than needed.

Definition 2.4

A code of dimension $n$, $C_n$, is a subset of $\{0, 1\}^n$, i.e. $C_n \subseteq \{0, 1\}^n$, and the code size, is denoted by $|C_n|$, equals to the number of codewords in $C_n$ [3].

Definition 2.5

A code $C$ is a $t$-code (that is $t$ asymmetric error correcting code) if it can correct up to $t$ errors, that is, there exists a rule (a decoding method) such that if $x \in C$ and $v$ is obtained from $x$ by changing at most $t$ 1’s in $x$ into 0’s, then the rule recovers $x$ from $v$. The set of all codewords in the $t$-code of dimension $n$ is denoted by $A(n, t)$ [4].

This thesis deals with single asymmetric error correcting codes, which means $t = 1$.

### 2.4 Asymmetric Distance and Hamming Distance

Definition 2.6

Let $x$ and $y$ be two codewords of dimension $n$, the asymmetric distance of $x$ and $y$ is defined by:

$$d_a(x, y) = \max \{N(x, y), N(y, x)\} \quad [3].$$
Where \( N(x, y) \) is given in Definition 2.2.

**Definition 2.7**

The *minimum asymmetric distance of a code* \( C \), is denoted by \( D_a(C) \), is

\[
D_a(C) = \min\{ d_a(x, y) : x, y \in C \text{ and } x \neq y \}.
\]

**Remark 2.1**

A code \( C \) can correct \( t \) asymmetric errors or fewer if \( D_a(C) > t \) (see Theorem 2.1), i.e \( D_a(A(n, t)) > t \).

**Remark 2.2**

The proposed code has minimum asymmetric distance two so it is capable of correcting a single asymmetric error.

**Definition 2.8**

Let \( x \) and \( y \) be two codewords of dimension \( n \), the *Hamming distance* of \( x \) and \( y \) is defined by:

\[
d_h(x, y) = N(x, y) + N(y, x) \ [3].
\]

**Definition 2.9**

The *minimum Hamming distance of a code* \( C \) is defined by

\[
D_h(C) = \min\{ d_h(x, y) : x, y \in C \text{ and } x \neq y \} \ [3].
\]
Example 2.1

Suppose there is a code \( C = \{v_1, v_2, v_3\} \), \( v_1 = 1011000 \), \( v_2 = 1100101 \), and \( v_3 = 1001011 \).

Dimension of \( C = 7 \), Size of \( C = 3 \).

\( w(v_3) = 4 \).

Moreover we have:

\[
N(v_1, v_2) = 2 \quad \text{as follows} \quad v_1: 10\underbrace{11}\underbrace{000} \\
\quad v_2: 11\underbrace{001011}
\]

\[
N(v_2, v_1) = 3 \quad \text{as follows} \quad v_2: 1\underbrace{1001011} \\
\quad v_1: 1\underbrace{011000} \\
\]

\[
d_a(v_1, v_2) = \max\{N(v_1, v_2), N(v_2, v_1)\} = 3, \quad d_a(v_1, v_3) = 2, \quad \text{and} \quad d_a(v_3, v_2) = 2
\]

\[
D_a(C) = \min\{d_a(v_1, v_2), d_a(v_1, v_3), d_a(v_3, v_2)\} = 2
\]

\[
d_h(v_1, v_2) = N(v_1, v_2) + N(v_2, v_1) = 5, \quad d_h(v_1, v_3) = 3, \quad \text{and} \quad d_h(v_3, v_2) = 4
\]

\[
D_h(C) = \min\{d_h(v_1, v_2), d_h(v_1, v_3), d_h(v_3, v_2)\} = 3
\]

The relation between error correcting code and the number of asymmetric error that can be corrected depends on the asymmetric distance of the code itself. Any binary
code $C$ of asymmetric distance $D_a(C) = \Delta$ can correct $\Delta - 1$ or fewer asymmetric errors, and hence it is called a $(\Delta-1)$ asymmetric error correcting code.

**Theorem 2.1**

Any binary code $C$ of asymmetric distance $\Delta$ can correct $\Delta - 1$ or fewer asymmetric errors. It is therefore called a $(\Delta-1)$ asymmetric error correcting code.

**Proof:**

Without loss of generality, we can assume that asymmetric errors are of the type $(1 \rightarrow 0)$. For any codeword $x \in C$, let $S_x$ denote the set of all vectors obtained from $x$ by introducing $t$ errors of the type $(1 \rightarrow 0)$, for $0 \leq t \leq \Delta - 1$.

Consider two codewords $x, y \in C$: since $d_a(x, y) \geq \Delta$, without loss of generality we can assume $N(x, y) \geq \Delta$. Clearly, $y$ cannot become $x$ by any number of $(1 \rightarrow 0)$ errors less than or equal to $\Delta - 1$. Also, $\Delta - 1$ or fewer $(1 \rightarrow 0)$ errors cannot take $x$ to $y$ or into $S_y$. That $S_x$ and $S_y$ are disjoint and the code can correct up to $\Delta - 1$ asymmetric errors.

By Theorem 2.1, we conclude that any code $C$ with minimum asymmetric distance greater than or equal to two can correct a single asymmetric error. For this reason, the goal of this work is to find for a given dimension $n$ the largest possible code size of dimension $n$ such that $D_a(C_n) \geq 2$. 
2.5 Construction Method

The field of single asymmetric error correcting codes (SAECC) has been studied extensively in the last few decades. It is used to enable systems with Z-channel to detect and correct one error and make these systems more reliable than the previous one. One goal of introducing new single asymmetric error correcting codes is to improve data rate (code size) of the existing code of same dimension.

The construction of the proposed codes uses the Cartesian product of two partitions $A$-partition and $B$-partition, where the size of the constructed code depends on the size of classes in each partition. In this chapter, the Cartesian product construction method of single asymmetric error correcting codes is described and the issues related to the partitions are discussed.

2.5.1 Cartesian Product Construction Method

The construction method which has been used in this work is based on the Cartesian product of two sets of partitioned codes of smaller dimensions. Although the Cartesian product of two sets is well-known, and it was used by many researchers as mentioned in [2], it has a slightly different meaning when the two sets are codes.
Definition 2.10

The *Cartesian Product of two codes*, $X$ and $Y$, is the code $X \times Y$ such that every pair $(x, y) \in X \times Y$ is a codeword which is the concatenation of the codeword $x \in X$ and the codeword $y \in Y$.

According to Definition 2.10, $X \times Y$ is not equal to $Y \times X$ in general.

Example 2.2

Suppose $X = \{00, 11\}$ and $Y = \{001, 011, 111\}$ be two codes. The Cartesian product of $X$ and $Y$ is the code $X \times Y = \{00001, 00011, 00111, 11001, 11011, 11111\}$.

Consider the first codeword in $X \times Y$, which is 00001. Clearly it is the concatenation of the codeword 00 $\in X$ and the codeword 001 $\in Y$. This is true for every codeword in $X \times Y$.

According to Definition 2.10, if the code $X$ is of dimension $p$ and the code $Y$ is of dimension $q$ then the code $X \times Y$ is of dimension $p + q$. This means that a code of larger dimension can be formed by the *Cartesian product* of two codes of smaller dimensions. Before going further in explaining of the construction method, consider the following definition.
Definition 2.11

Let $X$ be a set of codewords of dimension $n$, and let $X_1, X_2, \ldots, X_m$ be $m$ subsets of $X$. The set $\{X_1, X_2, \ldots, X_m\}$ is called a partition of $X$ of dimension $n$ if the following two conditions hold [2]:

1. $X_i \cap X_j = \emptyset$ for $i \neq j$, and

2. $\bigcup_{i=1}^{m} X_i = X$.

The subsets $X_1, X_2, \ldots, X_m$ are called classes; and the set $X$ is said to be partitioned into $m$ classes.

The construction method of a single asymmetric error correcting code is based on the Cartesian product of two sets of partitioned codes of smaller dimensions, called $A$-partition and $B$-partition. These partitions are defined as follows:

Definition 2.12

Let $A$ be the set of all the $2^p$ binary vectors of dimension $p$ and let $\{A_1, A_2, \ldots, A_k\}$ be a partition of $A$, such that $D_a(A_i) \geq 2$ for $1 \leq i \leq k$. Then, the partition $\{A_1, A_2, \ldots, A_k\}$ is called an $A$-partition of dimension $p$ [2].
Definition 2.13

Let $B$ be the set of the $2^{q-1}$ even weight binary vectors of dimension $q$ and let

$\{B_1, B_2, \ldots, B_s\}$ be a partition of $B$ such that $D_a(B_i) \geq 2$ for $1 \leq i \leq s$.

Then, the partition $\{B_1, B_2, \ldots, B_s\}$ is called a $B$-partition of dimension $q$ [2].

To construct a single asymmetric error correcting code of dimension $n$, two sets of partitioned codes, namely: $A$-partition = $\{A_1, A_2, \ldots, A_k\}$ and $B$-partition = $\{B_1, B_2, \ldots, B_s\}$ defined as above are involved. The dimension; say $p$ and $q$ of these two partitions satisfy $p + q = n$. The constructed code, denoted by $C_n$, of dimension $n$ is the union of the Cartesian products of all pairs $(A_i \times B_i)$ in these two partitioned codes, i.e.

$$C_n = A_1 \times B_1 \cup A_2 \times B_2 \cup A_3 \times B_3 \cup \ldots \cup A_\alpha \times B_\alpha$$

(2.1)

Where $\alpha = \min\{k, s\}$

The size of the code is computed by:

$$|C_n| = |A_1| \ast |B_1| + |A_2| \ast |B_2| + \ldots + |A_\alpha| \ast |B_\alpha|$$

(2.2)
Theorem 2.2

The code $C_n$ of dimension $n = p + q$ which has been obtained by using Cartesian product of two partitions: $A$-partition and $B$-partition is a single asymmetric error correcting code [2], see Equation (2.1).
Proof:

Let \( x, y \in C_n \) and \( x \neq y \). Let \( x = x'x'' \) and \( y = y'y'' \) where \( x' \in A_i \), \( x'' \in B_i \), \( y' \in A_i \), and \( y'' \in B_i \).

Case 1, \( i = j \):

Either \( x' \neq y' \Rightarrow d_a(x', y') \geq 2 \Rightarrow d_a(x, y) \geq 2 \)

or \( x'' \neq y'' \Rightarrow d_a(x'', y'') \geq 2 \Rightarrow d_a(x, y) \geq 2 \).

Case 2, \( i \neq j \):

Here we have \( d_a(x', y') \geq 1 \) since \( x' \neq y' \) and \( d_a(x'', y'') \geq 2 \) since \( x'' \neq y'' \) and \( x'' \) and \( y'' \) are both even, therefore \( d_a(x, y) \geq 3 \Rightarrow d_a(x, y) \geq 2 \).

Form case 1 and case 2 we can say that the minimum asymmetric distance of code \( C_n \) satisfies \( D_a(C_n) \geq 2 \).

Example 2.3

To construct a single asymmetric error correcting code \( C_6 \), let \( p = 2 \) and \( q = 4 \). Then \( A = \{00, 01, 10, 11\} \) can be partitioned into \( A_1 = \{00, 11\}, \ A_2 = \{01\} \) and \( A_3 = \{10\} \), and \( B = \{0000, 0001, 0010, 0011, \ldots, 1110, 1111\} \), the even vectors in \( B \) equal 8 and it can be partitioned into \( B_1 = \{0000, 0011, 1100, 1111\}, \ B_2 = \{0101, 1010\} \) and
$B_3 = \{0110, 1001\}$. We obtain a code $C_6$ of dimension $2 + 4 = 6$, where $C_6 = A_1 \times B_1 \cup A_2 \times B_2 \cup A_3 \times B_3$, having $(2 \times 4) + (1 \times 2) + (1 \times 2) = 12$ codewords as shown in (Table 2.1).

<table>
<thead>
<tr>
<th>Classes $A \times B$</th>
<th>A-partition</th>
<th>B-partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1 \times B_1$</td>
<td>00 0000</td>
<td>00 0011</td>
</tr>
<tr>
<td></td>
<td>00 1100</td>
<td>00 1111</td>
</tr>
<tr>
<td></td>
<td>11 0000</td>
<td>11 0011</td>
</tr>
<tr>
<td></td>
<td>11 1100</td>
<td>11 1111</td>
</tr>
<tr>
<td>$A_2 \times B_2$</td>
<td>01 0101</td>
<td>01 1010</td>
</tr>
<tr>
<td>$A_3 \times B_3$</td>
<td>10 0110</td>
<td>10 1001</td>
</tr>
</tbody>
</table>

The size of asymmetric error correcting code $C$ with dimension $n$, which is constructed using Cartesian product method, is affected directly by the choice of $p$ and $q$ for $A$-partition and $B$-partition respectively, and by the number and the sizes of the classes in these partitions. Therefore, in order to maximize the size of the code $C_n$ of dimension $n$, appropriate values of $p$ and $q$, satisfying $n = p + q$ should be selected. Without loss of generality, any partition $A = \{A_1, A_2, \ldots, A_k\}$ is assumed to satisfy $|A_i| \geq |A_{i+1}|$ for $1 \leq i < k$. Once $p$ and $q$ are chosen, "good" $A$- and $B$-partitions should be obtained. Suppose that there are two $A$-partitions for the
same dimension, \( \{A_1, A_2, \ldots, A_k\} \) and \( \{A'_1, A'_2, \ldots, A'_k\} \), with \( k \leq k' \). We can say that \( \{A_1, A_2, \ldots, A_k\} \) is better than \( \{A'_1, A'_2, \ldots, A'_k\} \), in general, if \( |A_i| \geq |A'_i| \) for all \( i \) such that \( 1 \leq i \leq m \), for some integer \( m \leq k \). The size of the constructed code depends on the sizes of the largest \( m \) classes used in the Cartesian product method. Therefore, in this example, the partition \( \{A_1, A_2, \ldots, A_k\} \) yields more codewords than \( \{A'_1, A'_2, \ldots, A'_k\} \) in the Cartesian product method.

In general, the better partition for any dimension \( n \) should have as few numbers of classes as possible, and the sizes of the classes have to be maximized. After the selection of the \( A \)- and \( B \)-partitions, the code is formed by using the Cartesian product of the largest class in \( A \)-partition with the largest class in \( B \)-partition, then the second largest with the second largest and so on.

**2.5.2 B-Partitions**

The \( B \)-partitions shown in (Table 2.2) are obtained from the partitioning of the constant weight vectors into classes with \textit{Hamming distance} 4 (see [2, 5]). For example, the entries for \( q = 4 \) which are 4, 2, and 2 are obtained as follows: First, the vectors of weight 0 are partitioned into one class. Namely \( \{0000\} \); the vectors of weight 2 are partitioned into three classes: \( \{0011, 1100\} \), \( \{1001, 0110\} \), and \( \{1010, 0101\} \); and the vectors of weight 4 have one class which is \( \{1111\} \). The eight
even weight codewords of dimension 4 can then be partitioned into three classes of sizes 4, 2, and 2 respectively as follows: \( \{0000, 0011, 1100, 1111\} \), \( \{1001, 0110\} \), and \( \{1010, 0101\} \), where each class is of asymmetric distance two. The constant weight partitions of different weights are listed in [5] for binary vectors of dimensions up to 14. Partitions of larger even weight vectors can be obtained using the procedure given by Brouwer in [5], and partitions of different even weights can be constructed (as given before) to obtain classes of \( B \)-partitions of all even weight vectors of the desired dimension.

<table>
<thead>
<tr>
<th>( q )</th>
<th>( B_1 )</th>
<th>( B_2 )</th>
<th>( B_3 )</th>
<th>( B_4 )</th>
<th>( B_5 )</th>
<th>( B_6 )</th>
<th>( B_7 )</th>
<th>( B_8 )</th>
<th>( B_9 )</th>
<th>( B_{10} )</th>
<th>( B_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>22</td>
<td>20</td>
<td>20</td>
<td>18</td>
<td>16</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>33</td>
<td>32</td>
<td>32</td>
<td>13</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>72</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>62</td>
<td>60</td>
<td>54</td>
<td>40</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>125</td>
<td>124</td>
<td>118</td>
<td>117</td>
<td>110</td>
<td>101</td>
<td>100</td>
<td>94</td>
<td>79</td>
<td>46</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>248</td>
<td>246</td>
<td>234</td>
<td>234</td>
<td>224</td>
<td>198</td>
<td>192</td>
<td>176</td>
<td>136</td>
<td>94</td>
<td>66</td>
</tr>
</tbody>
</table>

In [6], it is mentioned that the number of classes that can be used in the construction of a partition of even vectors of dimension \( q \) is equal to \( q - 1 \) classes when \( q = 2^i \), or \( q = 3 \times 2^i \) for \( i \geq 1 \) and even when \( q = 5 \times 2^i \) for \( i \geq 1 \). For example, as given in
(Table 2.2), when \( q = 4 \), it gives a partition with three classes of the following sizes: 4, 2 and 2. The existing \( B\)-partitions are very tight, and it seems to be hard to get any significant improvement over there.

### 2.5.3 A-Partitions

The A-partition of a code of a given dimension can be obtained using several methods like: Abelian group partitioning [7], Cartesian product of using smaller partitions with special constrains [2], and coloring method [8]. In general, partitions can contain one or more classes that have the same sizes for that dimensions. However, there is no proof for whether a given partition is optimal or not.

The Abelian group partitioning method, given by Varshamov in 1973 [9], was improved and used in 1979 by Constantin and Rao [7]. In this method, a code \( A \) of some dimension \( p \) is partitioned into \( p + 1 \) disjoint sets, \( A_1, A_2, \ldots, A_{p+1} \) such that \( D_s(A_i) \geq 2 \) for \( 1 \leq i \leq p + 1 \).

Given a set \( A \) of binary vectors of dimension \( p \), the group partition \((\Gamma p)\) of \( p + 1 \) classes is constructed as follows:

- **Algorithm:** Group Partitioning
- **Input:** set of binary vectors \( A \)
- **Output:** the \( A\)-partition \( \Gamma p \)
1. Initialize $\Gamma p = \{A_1, A_2, \ldots, A_{p+1}\}$ where $A_i = \emptyset$ for all $i$.

2. For every codeword $c = [c_1, c_2, c_3, \ldots, c_p] \in A$, where $c_i \in \{0, 1\}$, do
   - Compute the sum $k = (\sum_{j=1}^{p} j \cdot c_j) \mod (p + 1)$
   - Add $c$ to the class $A_{k+1}$

3. Return $\Gamma_p$

End.

It is shown in [7] that this algorithm constructs $p + 1$ classes of asymmetric distance greater than or equal to 2. Of course, the largest class is at least of size $2^p / (p + 1)$.

Table 2.3 shows $A$-partitions that are obtained using Abelian group partitioning method.

The Cartesian product method introduced in [2] has been used to construct $A$-partitions from smaller sets of partitions, it is quite sensitive to the sizes of the smaller partitions. The better partitions used in this method, the more codewords constructed.
Table 2.3 A-partitions using Abelian group partitioning ($\Gamma_p$)

<table>
<thead>
<tr>
<th>$P$</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$A_5$</th>
<th>$A_6$</th>
<th>$A_7$</th>
<th>$A_8$</th>
<th>$A_9$</th>
<th>$A_{10}$</th>
<th>$A_{11}$</th>
<th>$A_{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>28</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>52</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>52</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>94</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
<td>93</td>
</tr>
</tbody>
</table>

Recall the construction method given in Section 2.5.1 for constructing single asymmetric error correcting codes. A procedure similar to this method can be used to construct $A$-partitions. The idea behind this constructed method is to use different combination ($p + 1$ combinations) of Cartesian product of classes for the same $A$-partition and $B$-partition (even vector $B$-partition and odd vector $B$-partition) which gives a different code that constructs a new $A$-partition of dimension $p$. It gives better partition than the group partition for some values of $p$, like 6, 10 and 11.

In order to obtain a partition of all binary vectors of dimension $p$, two numbers $s$ and $t$ are chosen such that

$$s = \left\lfloor \frac{p - 1}{2} \right\rfloor$$
and

\[ t = p - s \]

This implies:

\[ t = \left\lfloor \frac{p + 1}{2} \right\rfloor \quad (2.3) \]

Then, all classes in A-partition of vectors of dimension \( s \), and all classes in B-partitions of the odd as well as of the even weight vectors of dimension \( t \) are employed in different distinct combinations to produce the desired partitions of dimension \( p \). It is always possible to get a partition with \( s + 1 \) classes of the vectors of dimension \( s \), and \( t \) classes of all odd (or even) weight vectors of dimension \( t \). This is true because the first one is the same as the A-partitions, and the second one is similar to the B-partitions.

According to Equation (2.3), it follows that

\[ t = \begin{cases} 
\frac{(p + 1)}{2} & \text{if } p \text{ is odd} \\
\frac{(p + 2)}{2} & \text{if } p \text{ is even}
\end{cases} \]

This implies:

\[ 2t = \begin{cases} 
(p + 1) & \text{if } p \text{ is odd} \\
(p + 2) & \text{if } p \text{ is even}
\end{cases} \]
Therefore, it is always possible to obtain $2t$ classes of binary vectors of dimension $p$.

In many cases this procedure produces $A$-partitions which are at least as good as (and in many cases better than) those obtained using the group method.

Example 2.4

In this example, the $A$-partition for $p = 6$ of seven classes of the sizes: 12, 10, 10, 8, 8, 8 and 8 is illustrated. Here, $s = \lfloor (6-1)/2 \rfloor = 2$ and $t = \lceil (6+1)/2 \rceil = 4$.

Recall that one can partition all binary vectors of dimension 2, $S = \{00, 01, 10, 11\}$, into $S_1 = \{00, 11\}$, $S_2 = \{01\}$ and $S_3 = \{10\}$.

The eight even weight binary vectors of dimension 4, $T = \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$, can be partitioned into:

$T_1 = \{0000, 0011, 1100, 1111\}$, $T_2 = \{0101, 1010\}$ and $T_3 = \{0110, 1001\}$.

The eight odd weight vectors, $T' = \{0001, 0010, 0100, 0111, 1000, 1011, 1101, 1110\}$, can be partitioned into four classes:

$T_1' = \{0001, 1110\}$, $T_2' = \{0010, 1101\}$, $T_3' = \{0100, 1011\}$ and $T_4' = \{1000, 0111\}$

Now the seven classes of the $A$-partition of all the $2^6$ binary vectors can be obtained as illustrated in (Figure 2.2). Notice that $A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_7$

\[ \text{Figure 2.2} \]
contain all the 64 binary vectors of dimension 6, \( A_i \cap A_j = \emptyset \) when \( i \neq j \), and \( D_s(A_i) \geq 2 \) for \( 1 \leq i \leq 7 \).

\[
A_1 = S_1 \times T_1 \cup S_2 \times T_2 \cup S_3 \times T_3 \text{ of size } 12 \\
A_2 = S_1 \times T_2 \cup S_2 \times T_3 \cup S_3 \times T_1 \text{ of size } 10 \\
A_3 = S_1 \times T_3 \cup S_2 \times T_1 \cup S_3 \times T_2 \text{ of size } 10 \\
A_4 = S_1 \times T_1' \cup S_2 \times T_2' \cup S_3 \times T_3' \text{ of size } 8 \\
A_5 = S_1 \times T_2' \cup S_2 \times T_3' \cup S_3 \times T_4' \text{ of size } 8 \\
A_6 = S_1 \times T_3' \cup S_2 \times T_4' \cup S_3 \times T_1' \text{ of size } 8 \\
A_7 = S_1 \times T_4' \cup S_2 \times T_1' \cup S_3 \times T_2' \text{ of size } 8
\]

Figure 2.2 Constructing A-partition for \( p = 6 \) with rotation

The sizes of these seven classes are: 12, 10, 10, 8, 8, 8, and 8. As (Figure 2.2) shows.

Table 2.4 shows the size of improved partitions by using Cartesian product method.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( A_1 )</th>
<th>( A_2 )</th>
<th>( A_3 )</th>
<th>( A_4 )</th>
<th>( A_5 )</th>
<th>( A_6 )</th>
<th>( A_7 )</th>
<th>( A_8 )</th>
<th>( A_9 )</th>
<th>( A_{10} )</th>
<th>( A_{11} )</th>
<th>( A_{12} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>104</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>90</td>
<td>88</td>
<td>84</td>
<td>84</td>
<td>84</td>
<td>82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>180</td>
<td>180</td>
<td>176</td>
<td>172</td>
<td>172</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>164</td>
<td>164</td>
<td></td>
</tr>
</tbody>
</table>

This procedure may be deemed as a generalized version of the code construction method proposed in Section 2.5.1. Clearly, each \( A_i \) of dimension \( p \) in the above example is obtained in the same way \( C_n \) is obtained only using different
combinations of $S$, $T$ and $T'$ partitions. These combinations have a special characteristics; no class in $S$ is in the Cartesian product in all combinations more than one time with the same class in $T$ and $T'$ partitions. One way to get these combinations is by a simple rotation strategy of the classes to get $A$-partition. In some cases, other combination strategies would give better $A$-partition than simple rotation method, especially when the number of the classes in each of $S$, $T$ and $T'$ is even. For example if there are four classes in each of $S$, $T$ and $T'$, one of the combinations that is given in (Figure 2.3) could give better partition than those obtained by simple rotation technique

\[ A_1 = S_1 \times T_1 \cup S_2 \times T_2 \cup S_3 \times T_3 \cup S_4 \times T_4 \]
\[ A_2 = S_1 \times T_2 \cup S_2 \times T_1 \cup S_3 \times T_4 \cup S_4 \times T_3 \]
\[ A_3 = S_1 \times T_3 \cup S_2 \times T_4 \cup S_3 \times T_1 \cup S_4 \times T_2 \]
\[ A_4 = S_1 \times T_4 \cup S_2 \times T_3 \cup S_3 \times T_2 \cup S_4 \times T_1 \]
\[ A_5 = S_1 \times T'_1 \cup S_2 \times T'_2 \cup S_3 \times T'_3 \cup S_4 \times T'_4 \]
\[ A_6 = S_1 \times T'_2 \cup S_2 \times T'_1 \cup S_3 \times T'_4 \cup S_4 \times T'_3 \]
\[ A_7 = S_1 \times T'_3 \cup S_2 \times T'_4 \cup S_3 \times T'_1 \cup S_4 \times T'_2 \]
\[ A_8 = S_1 \times T'_4 \cup S_2 \times T'_3 \cup S_3 \times T'_2 \cup S_4 \times T'_1 \]

Figure 2.3 Constructing an A-partition without rotation
Another technique, which is used to improving *A-partitions*, is Graph coloring technique that given in [10]. Briefly, to construct an *A-partition* of dimension *p*, a graph $G = (V, E)$ of $2^p$ nodes is constructed, where $V$ is the set of all $2^p$ binary vectors and $E = \{(x, y) : x, y \in V \text{ and } d_a(x, y) = 1\}$.

The nodes of the graph are colored using *m* colors, such that $\forall x, y \in V$, if $(x, y) \in E$ then *x* and *y* have different colors. The basic idea behind this method that the set of all nodes having color *k*, say $A_k$, satisfies $D_a(A_k) \geq 2$. This means that graph coloring can be used to find *A-partition*. The following example shows how graph-coloring can be used to partition a set of binary vectors.

Example 2.5

In this example, graph-coloring is used to partition the set $V = \{0011, 0110, 1100, 0001, 0000, 1001\}$, First, the graph $G = (V, E)$ is constructed where nodes are all elements of $V$, has an edge between *x* and *y* $\in V$, if and only if $d_a(x, y) = 1$. The constructed graph, shown in Figure 2.4, has the following set of edges:

$E = \{(0011, 0110), (0011, 0001), (0011, 1001), (1100, 0110), (1100, 1001), (0000, 0001), (0001, 1001)\}$. 
This graph can be colored with three different colors using first-fit algorithm such that there is no two adjacent nodes having the same color, Figure 2.4 shows coloring assignment and the partitions of set $V$ into three disjoint subsets as follows:

$$V_1 = \{0011, 1100, 0000\}, \ V_2 = \{1001\} \ and \ V_3 = \{0110, 1000\}$$

Notice that $D_i(V_i) \geq 2$, for $i = 1, 2, 3$.

![Figure 2.4 Graph coloring method for Example 2.5](image)

In order to obtain the $A$-partitions that can be used in the Cartesian product method, all the $2^n$ codewords should be included in the set $V$, where $p$ is the dimension of the
$A$-partitions. Then, $V$ is partitioned into some subsets of minimum asymmetric distances $D_a \geq 2$ using graph-coloring method. Coloring a graph of $2^p$ nodes is not an easy task since it is an $NP$-complete problem. Therefore, a modified algorithm called Coloring In-Limited-Backtracking Algorithm (CILBA) [11] was designed to solve the graph coloring method. CILBA indeed was used to construct a new $A$-partition for $p = 7$ with cardinalities 18, 18, 18, 18, 17, 16, 13, and 10. It gives a much better partition for $p = 7$ than Abelian group partition.
The theory and construction of asymmetric error correcting codes have been studied since the late 1950's. In 1959, Kim and Freiman proposed a construction method of asymmetric error correcting codes using “prefix/suffix” constructions of code [12]. The construction of a code of a given dimension gives a lower bound for the size of
all codes of the given dimension. In 1964, Varshamov gave an explicit upper bound for asymmetric error correcting codes [13].

In 1971, Goldbaum obtained tighter upper bounds using integer programming techniques. Two years later, Varshamov used algebraic group theory to construct codes for correcting asymmetric errors [14]. In 1979, Constantin and Rao improved the same method and used it to construct codes for asymmetric channel [7].

In 1981, Klove [4] and Delsarte and Piret [15] improved the upper bounds that was obtained by Goldbaum in 1971 by adding more constraints to the integer programming technique. Delsarte and Piret also have introduced the "expurgating/puncturing" construction method for asymmetric error correcting codes. They used the idea of constructing a code of dimension $n$ and asymmetric distance $d$ by modifying an initial code with good (Hamming) distance properties by successive judicious deletions of coordinates and vectors [15]. A year later, in 1982, Shiozaki presented a construction method of a $t$-fold asymmetric error-correcting code of dimension $n - 1$ by expurgating and puncturing any $t$-fold symmetric error-correcting code of dimension $n$.

In 1987, Weber De Vroedt and Boekee proposed new upper bounds on the size of asymmetric error correcting codes by further enhancing the constraints for the integer programming technique [15]. A year later, they improved the upper bounds and proposed constructions for asymmetric error correcting codes using a general
"expurgating/puncturing" construction method [16]. This method includes as special cases the construction method of Shiozaki and some of the constructions of Delsarte and Piret, they proposed a construction method for a code $C$ of dimension $(n - m)$, where $1 \leq m < n$, and asymmetric distance $d \geq t + 1$ which consists of expurgating and puncturing a code $C'$ of dimension $n$ and Hamming distance $h \geq 2t + 1$.

In 1992, Zhang and Xia derived new lower bounds for asymmetric single-error correcting codes. The codes were obtained by puncturing constant weight codes and by using a random coding argument. Their method improves code size from 12 to 19 at that time. Although their method is non-constructive, they used probability and counting techniques to show that the asymmetric single-error-correcting codes of the following sizes exist [17].

In 1997, Al-Bassam, Venkatesan and Al-Muhammadi. proposed a new single asymmetric error-correcting codes. These codes are better than existing codes at that time when the code dimension $n$ is greater than 10, except for $n = 12$ and $n = 15$. In many cases the constructed codes contain at least $\lceil 2^n / n \rceil$ codewords. Their method is based on the Cartesian product of two smaller partitioned codes as explained in Chapter 2. They used the Cartesian product method with combination to construct A-partitions of larger sizes [2].
Table 3.1 summarizes the upper and lower bounds and the sizes of the codes that obtain by this method [2]. All upper bounds in Table 3.1 were obtained using the integer programming techniques as described in [15].

Table 3.1 Existing single asymmetric error correcting codes

<table>
<thead>
<tr>
<th>n</th>
<th>[2^n / n] Lower Index</th>
<th>Existing Code</th>
<th>[2^n / n - 1] Upper Index</th>
<th>Upper Bound [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2^a</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2^a</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4^a</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>6^i</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>12^h</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>18^e</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>36^c</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>56</td>
<td>62^c</td>
<td>64</td>
<td>62</td>
</tr>
<tr>
<td>10</td>
<td>102</td>
<td>108^c</td>
<td>113</td>
<td>117</td>
</tr>
<tr>
<td>11</td>
<td>186</td>
<td>180^e</td>
<td>204</td>
<td>210</td>
</tr>
<tr>
<td>12</td>
<td>341</td>
<td>340^d</td>
<td>372</td>
<td>410</td>
</tr>
<tr>
<td>13</td>
<td>630</td>
<td>652^e</td>
<td>682</td>
<td>786</td>
</tr>
<tr>
<td>14</td>
<td>1170</td>
<td>1204^e</td>
<td>1260</td>
<td>1500</td>
</tr>
<tr>
<td>15</td>
<td>2184</td>
<td>2216^d</td>
<td>2340</td>
<td>2828</td>
</tr>
<tr>
<td>16</td>
<td>4096</td>
<td>4232^e</td>
<td>4369</td>
<td>5430</td>
</tr>
<tr>
<td>17</td>
<td>7710</td>
<td>8192^f</td>
<td>8192</td>
<td>10379</td>
</tr>
<tr>
<td>18</td>
<td>14563</td>
<td>14624^e</td>
<td>15420</td>
<td>19898</td>
</tr>
<tr>
<td>19</td>
<td>27594</td>
<td>28548^d</td>
<td>29127</td>
<td>38008</td>
</tr>
<tr>
<td>20</td>
<td>52428</td>
<td>53856^e</td>
<td>55188</td>
<td>73174</td>
</tr>
<tr>
<td>21</td>
<td>99864</td>
<td>101576^e</td>
<td>104857</td>
<td>140798</td>
</tr>
<tr>
<td>22</td>
<td>190650</td>
<td>195700^f</td>
<td>199728</td>
<td>271953</td>
</tr>
</tbody>
</table>

(a) code by Varshamov  
(b) code by Kim and Freiman  
(c) code by Delsarte and Piret  
(d) code by Zhang and Xia  
(e) code by Al-Bassam, Venkatesan and Al-Muhammadi [2]  
(f) code by Al-Bassam and Al-Muhammadi [10]
It is important to mention that the lower bounds (or the sizes of the existing codes) in Table 3.1 are for constructible codes, i.e. codes that can actually be constructed. Moreover in many cases the single asymmetric error-correcting codes satisfy:

\[
\left\lfloor \frac{2^n}{n} \right\rfloor \leq |C| \leq \left\lfloor \frac{2^n}{(n-1)} \right\rfloor.
\]

In 2000, S. Al-Bassam and S. Al-Muhammadi proposed a new single error correcting code of dimension \( n = 17 \) [10]. This code is constructed using a product of two codes of smaller dimensions. The proposed code is of size 8192. They applied coloring algorithm to get better partition for \( n = 7 \), and used this partition to construct the code by the Cartesian product method.

In 2003, F. Fu and C. Xing presented a general method in [18] to construct \( k \)-asymmetric error correcting codes, for \( k = 1, 2, 3 \) and 4, which extends a previous work for Xing. It depends on finite field of prime power, and shows that some previously known lower bounds for binary asymmetric error-correcting codes can be obtained from their general construction. However, Fu and C. Xing work did not improve the lower bound of single asymmetric error correcting codes. Their lower bound for code of minimum asymmetric distance two is \( A(n, \Delta) = \frac{2^n}{(n + 1)} \). In 2004, Liang, Chang and Chen developed in [18] a construction algorithm that improves the complexity of the construction method presented in [19] without improving the lower bounds. They developed a construction algorithm which
requires $O(2^n)$ in the worst case, while Fu and Xing method requires $O(n2^n)$. In most cases, the number of operations is much lower than that.

In 2008, Neri, Skantzos and Bolle derived critical noise levels for Gallager codes on asymmetric channels as a function of the input bias and the temperature [20]. They studied the space of codewords and the entropy in the various decoding regimes by using a statistical mechanics approach. Some other works were done to problem of evaluating the undetected error probability of Varshamov–Tenengol’ts codes in [21]. Computation of the undetected error probability for error detecting codes over the Z-channel for Varshamov–Tenengol’ts (VT) codes was studied. An exact formula for the probability of undetected errors was given. It was explicitly computed for small code dimensions. A comparison to the Hamming codes was given. It was further shown that heuristic arguments give a very good approximation that can easily be computed even for large dimensions. They used Monte Carlo methods to estimate performance for long code dimensions. They verified that the probability of undetected errors is almost constant in a wide region of values of the channel error probability.

Since 2005, a great deal of research has been dedicated to find lower bounds for systematic single asymmetric error correcting codes [22, 23]. A comparison of the number of codewords in the systematic single asymmetric error-correcting codes with that of the existing nonsystematic single asymmetric error-correcting codes
was conducted. In general, systematic codes have a worst coding efficiency than nonsystematic. However systematic codes often are less complex in encoding and decoding.

The error types in several communication systems and some VLSI media are of asymmetric error in nature. Some implementations of the selective-repeat ARQ (Automatic-Repeat Request) protocol suited for the communication over the $m$ ($\geq 2$)-ary asymmetric channel which makes use of all asymmetric error detecting codes that are given in [24]. For those codes, the number of retransmissions needed to receive all codewords correctly is derived, and as a special case, the number of retransmissions needed to receive codewords correctly is derived for the Z-channel.
Chapter 4

New Single Asymmetric Error

Correcting Codes and A-Partitions

Code construction is an important issue in coding theory for the code to be applied in proper applications. Assume a code of a given size does exist, it may not be used unless the construction of that code is known. So, the code construction is more useful than just showing that a code of a given size does exist. Preferably, the
construction should be easily implementable for information encoding and decoding.

In this chapter, new single asymmetric error correcting codes are proposed. Also new $A$-partitions are introduced. Table 4.1 lists the sizes of the proposed codes, the existing codes and the upper bound for a given dimension. The upper bound itself does not mean that there is a code of that size, but it is proven that there is no code with a size more than the upper bound for that dimension.

In Section 4.1, a new algorithm is used to improve $A$-partitions that are used later in the Cartesian product construction method. Then, in Section 4.2, new single asymmetric error correcting codes are introduced.

### 4.1 Improving A-Partitions

In this section, heuristic techniques are used to generate $A$-Partition of dimension $p$. This method mainly depends on making combinations of codewords with a minimum asymmetric distance two, which can be used in constructing $A$-partition to get partitions which are better than the existing ones.
Table 4.1 New single Asymmetric error correcting Codes

<table>
<thead>
<tr>
<th>n</th>
<th>Existing Code</th>
<th>Proposed Code</th>
<th>Upper Bound [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$2^a$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>$2^a$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>$4^a$</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>$6^a$</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>$12^b$</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>$18^c$</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>$36^c$</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>$62^c$</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>10</td>
<td>$108^c$</td>
<td>108</td>
<td>117</td>
</tr>
<tr>
<td>11</td>
<td>$180^e$</td>
<td>180</td>
<td>210</td>
</tr>
<tr>
<td>12</td>
<td>$340^d$</td>
<td>340</td>
<td>410</td>
</tr>
<tr>
<td>13</td>
<td>$652^e$</td>
<td>652</td>
<td>786</td>
</tr>
<tr>
<td>14</td>
<td>$1204^e$</td>
<td>1228*</td>
<td>1500</td>
</tr>
<tr>
<td>15</td>
<td>$2216^d$</td>
<td>2288*</td>
<td>2828</td>
</tr>
<tr>
<td>16</td>
<td>$4232^e$</td>
<td>4272*</td>
<td>5430</td>
</tr>
<tr>
<td>17</td>
<td>$8192^f$</td>
<td>8296*</td>
<td>10379</td>
</tr>
<tr>
<td>18</td>
<td>$14624^e$</td>
<td>14624</td>
<td>19898</td>
</tr>
<tr>
<td>19</td>
<td>$28548^f$</td>
<td>28688*</td>
<td>38008</td>
</tr>
<tr>
<td>20</td>
<td>$53856^e$</td>
<td>53856</td>
<td>73174</td>
</tr>
<tr>
<td>21</td>
<td>$101576^e$</td>
<td>101576</td>
<td>140798</td>
</tr>
<tr>
<td>22</td>
<td>$195700^f$</td>
<td>195700</td>
<td>271953</td>
</tr>
</tbody>
</table>

(a) code by Varshamov
(b) code by Kim and Freiman
(c) code by Delsarte and Piret
(d) code by Zhang and Xia
(e) code by Al-Bassam, Venkatesan and Al-Muhammadi [2]
(f) code by Al-Bassam and Al-Muhammadi [10]
(*) proposed code improves the existing code
Using the combination rules is a simple way to construct any class (subset) $A$ with a minimum asymmetric distance two, of dimension $p$ of any size $k \leq$ upper bound of that dimension $p$ (given in Table 4.1). Let $V = \{x_1, x_2, x_3, \ldots, x_n\}$, be a set of all $2^p$ binary vectors of dimension $p$, where $n = 2^p$. This method constructs a subset (combination) of binary vectors $A = \{y_1, y_2, y_3, \ldots, y_k\}$, such that every $y_i \in V$, and the size of $A$ is $k$. Then, every element (codeword) in the class is tested with all other elements in $A$ to satisfy $d_a(x_i, x_j) \geq 2$, $1 \leq i < k$ and $i < j \leq k$. One way to improve this method is to use techniques that cancel combinations as much as possible in every stage of constructing the $A$-partition from the set $V$ of all $2^p$ binary vectors.

Briefly, the proposed algorithm can be divided into two steps: First, it constructs a partition (subset) of vectors with dimension $n$, this partition has classes $\{v_1, v_2, v_3, \ldots, v_e\}$ of vectors such that any two vectors, $x$ and $y$, in one class have an asymmetric distance equals to one; i.e. $\forall x, y \in v_i$, $d_a(x, y) = 1$, for $1 \leq i \leq e$. Second, it uses heuristic techniques based on the combination rules to construct a new $A$-partition of classes $\{A_1, A_2, \ldots, A_f\}$ such that $D_a(A_i) \geq 2$, for $1 \leq i \leq e$. This algorithm leads to create better $A$-partitions than the existing ones. The new $A$-partitions have been used to construct a new single asymmetric error correcting codes.
The heuristic in the second step needs some expected values for the cardinalities of the classes for the *A-partition*, which are provided as inputs to the second step. Those expected values are taken from the existing *A-partition* or better. The resultant *A-partition* \( \{A_1, A_2, \ldots, A_f\} \) will eventually have classes of cardinalities \( S_1, S_2, \ldots, S_f \), such that \( \sum_{i=1}^{f} S_i = 2^p \), which includes all the binary vectors of dimension \( p \).

The proposed algorithm creates a combination of subsets provided by the first step. The \( i^{th} \) combination has a number of subsets equals to \( S_i \) where \( 1 \leq i \leq f \). Then the algorithm constructs a combination of binary vectors such that one vector from every subset is included in the combination of the subsets, i.e. \( A_i = \{y_1, y_2, y_3, \ldots, y_{s_i}\} \).

Then, the algorithm tests if \( A_i \) satisfies \( D_a(A_i) \geq 2 \). If this is true, all codewords in \( A_i \) will be cleared from their initial classes \( v_i \), and then the algorithm repeats these steps to process the next class \( A_{i+1} \) (with possible backtracking if needed), and so on, until a new *A-partition* is constructed. The pseudocode of the proposed algorithm is as follows.
Algorithm:

INPUT:

\[ p = \text{Dimension which is equal to the number of bit in the codewords} \]

\[ V = \{ x_1, x_2, x_3, \ldots, x_n \}, \quad n = 2^p, \text{ All binary vector in dimension } p \]

\[ \text{Cardinality} = \{ S_1, S_2, \ldots, S_f \} \]

OUTPUT:

\[ V = \{ A_1, A_2, \ldots, A_f \}, \text{ such that } S_i = |A_i| \]

\[ A_1 = \{ y_1, y_2, \ldots, y_{s_1} \}, \quad A_2 = \{ y_1, y_2, \ldots, y_{s_2} \}, \quad \ldots, \quad A_f = \{ y_1, y_2, \ldots, y_{s_f} \} \]

\( \forall x \in V \), initialize:

\[ w[x] = \text{number of 1's in the binary vector } x ; \]

\[ \text{Subset } [x] = \text{null} ; \]

Sort all \( x \in V \) such that \( \forall x_i, x_{i+1} \in V, \ w(x_i) < w(x_{i+1}) \) or \( w(x_i) = w(x_{i+1}) \) and \( \text{val}(x_i) < \text{val}(x_{i+1}) \);

Partition all \( x \in V \) using first fit to a subset \( v_i \) such that:
While there is a combination of a subsets \( \{v_1, v_2, \ldots, v_{|S_1|}\} \) and combination of codewords \( A_1 = \{b_1, b_2, \ldots, b_{|S_1|}\} \) such that \( b_i \in v_i \) do  //Loop 1

If \( (D_a(A_1) \geq 2) \) then  //IF 1

Update subsets \( v_1, v_2, \ldots, v_{|S_1|} \): such that \( v_i = v_j - b_i, 1 \leq i \leq |S_1| \)

While there is a combination of a subsets \( \{v_1, v_2, \ldots, v_{|S_2|}\} \) and combination of codewords \( A_2 = \{b_1, b_2, \ldots, b_{|S_2|}\} \) such that \( b_i \in v_i \) do //Loop 2

If \( (D_a(A_2) \geq 2) \) then  //IF 2

Update subsets \( v_1, v_2, \ldots, v_{|S_2|} \): such that \( v_j = v_i - b_i, 1 \leq i \leq |S_2| \)

(do the same steps as in Loop2 for other partitions)

Else // if there no partition satisfies conditions //
Update subsets $v'_1, v'_2, \ldots, v'_{|\mathcal{S}_i|}$ such that

$$v'_i = v'_i \cup b'_i, \ 1 \leq i \leq |\mathcal{S}_i| \quad \text{(Back tracking)}$$

End IF //IF 2

End while //Loop2

End IF //IF 1

End while //Loop1

End Algorithm

4.2 The Proposed Codes

The new single asymmetric error correcting codes are obtained as a result of applying the Cartesian product method discussed in Section 2.5.1. The sizes of these codes are computed by Equation (2.2). Table 4.3 and Table 4.4 represent $A$-partitions for $p = 6$ and 7 respectively. These partitions have been constructed using method discussed in Section 4.1. The proposed partitions are better than the ones found in the literature.

The cardinalities of new $A$-partitions are listed in (Table 4.2) for $p = 6$ and 7.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$A_5$</th>
<th>$A_6$</th>
<th>$A_7$</th>
<th>$A_8$</th>
<th>$A_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>16</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A_1$</td>
<td>$A_2$</td>
<td>$A_3$</td>
<td>$A_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>000000</td>
<td>000001</td>
<td>010000</td>
<td>100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100010</td>
<td>101000</td>
<td>001010</td>
<td>010010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>010100</td>
<td>000110</td>
<td>100001</td>
<td>000101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>001001</td>
<td>011100</td>
<td>111000</td>
<td>100011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>110001</td>
<td>110010</td>
<td>001101</td>
<td>110100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>101100</td>
<td>100101</td>
<td>010011</td>
<td>011001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>011010</td>
<td>001011</td>
<td>100110</td>
<td>001110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>000111</td>
<td>111001</td>
<td>011110</td>
<td>111010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>101011</td>
<td>010111</td>
<td>110101</td>
<td>101101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>011101</td>
<td>111110</td>
<td>101111</td>
<td>011111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>110110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>111111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$A_5$</th>
<th>$A_6$</th>
<th>$A_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000010</td>
<td>001000</td>
<td>001000</td>
</tr>
<tr>
<td>2</td>
<td>001100</td>
<td>000011</td>
<td>010001</td>
</tr>
<tr>
<td>3</td>
<td>110000</td>
<td>011000</td>
<td>100100</td>
</tr>
<tr>
<td>4</td>
<td>010101</td>
<td>010110</td>
<td>001111</td>
</tr>
<tr>
<td>5</td>
<td>101001</td>
<td>101010</td>
<td>110011</td>
</tr>
<tr>
<td>6</td>
<td>011011</td>
<td>100111</td>
<td>111010</td>
</tr>
<tr>
<td>7</td>
<td>101110</td>
<td>111100</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>110111</td>
<td>111011</td>
<td></td>
</tr>
</tbody>
</table>
Table 4.4 A-partition for $p = 7$

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0000001</td>
<td>0000010</td>
<td>0001000</td>
<td>0010000</td>
</tr>
<tr>
<td>2</td>
<td>0100010</td>
<td>0011000</td>
<td>001001</td>
<td>001010</td>
</tr>
<tr>
<td>3</td>
<td>0001100</td>
<td>1000100</td>
<td>0110000</td>
<td>0100100</td>
</tr>
<tr>
<td>4</td>
<td>1010000</td>
<td>010001</td>
<td>1000100</td>
<td>1000001</td>
</tr>
<tr>
<td>5</td>
<td>0110100</td>
<td>1101000</td>
<td>1010100</td>
<td>1110000</td>
</tr>
<tr>
<td>6</td>
<td>1000101</td>
<td>0100110</td>
<td>0000111</td>
<td>0011001</td>
</tr>
<tr>
<td>7</td>
<td>1001010</td>
<td>0010101</td>
<td>0011010</td>
<td>0010110</td>
</tr>
<tr>
<td>8</td>
<td>0010011</td>
<td>0001011</td>
<td>0101100</td>
<td>1001100</td>
</tr>
<tr>
<td>9</td>
<td>0101001</td>
<td>1010010</td>
<td>110001</td>
<td>010011</td>
</tr>
<tr>
<td>10</td>
<td>0100111</td>
<td>1000111</td>
<td>110110</td>
<td>110101</td>
</tr>
<tr>
<td>11</td>
<td>1011001</td>
<td>011010</td>
<td>0111001</td>
<td>0111001</td>
</tr>
<tr>
<td>12</td>
<td>1101100</td>
<td>0101101</td>
<td>111000</td>
<td>0001111</td>
</tr>
<tr>
<td>13</td>
<td>1110010</td>
<td>1100011</td>
<td>100110</td>
<td>110010</td>
</tr>
<tr>
<td>14</td>
<td>0011110</td>
<td>1011001</td>
<td>010001</td>
<td>100101</td>
</tr>
<tr>
<td>15</td>
<td>1111010</td>
<td>1101101</td>
<td>101011</td>
<td>111001</td>
</tr>
<tr>
<td>16</td>
<td>1001111</td>
<td>111011</td>
<td>011110</td>
<td>011011</td>
</tr>
<tr>
<td>17</td>
<td>0111011</td>
<td>0011111</td>
<td>110110</td>
<td>1011110</td>
</tr>
<tr>
<td>18</td>
<td>1111110</td>
<td>111110</td>
<td>111110</td>
<td>110111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$A_5$</th>
<th>$A_6$</th>
<th>$A_7$</th>
<th>$A_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0100000</td>
<td>100000</td>
<td>0000000</td>
<td>0010000</td>
</tr>
<tr>
<td>2</td>
<td>0010100</td>
<td>0000110</td>
<td>0000101</td>
<td>1001110</td>
</tr>
<tr>
<td>3</td>
<td>0000011</td>
<td>0010001</td>
<td>0010010</td>
<td>0111001</td>
</tr>
<tr>
<td>4</td>
<td>1001000</td>
<td>0101000</td>
<td>1100000</td>
<td>1101110</td>
</tr>
<tr>
<td>5</td>
<td>1100010</td>
<td>1100100</td>
<td>0011000</td>
<td>1101011</td>
</tr>
<tr>
<td>6</td>
<td>1010001</td>
<td>1000011</td>
<td>0101100</td>
<td>1101111</td>
</tr>
<tr>
<td>7</td>
<td>0111000</td>
<td>1011000</td>
<td>0110001</td>
<td>1101110</td>
</tr>
<tr>
<td>8</td>
<td>0001110</td>
<td>0110010</td>
<td>1000110</td>
<td>1101100</td>
</tr>
<tr>
<td>9</td>
<td>0100101</td>
<td>0001101</td>
<td>1001001</td>
<td>1101010</td>
</tr>
<tr>
<td>10</td>
<td>1110100</td>
<td>0101101</td>
<td>0110110</td>
<td>1101111</td>
</tr>
<tr>
<td>11</td>
<td>0010111</td>
<td>0110101</td>
<td>1010101</td>
<td>1110111</td>
</tr>
<tr>
<td>12</td>
<td>0101011</td>
<td>0011011</td>
<td>1100011</td>
<td>1111001</td>
</tr>
<tr>
<td>13</td>
<td>1001101</td>
<td>1101001</td>
<td>1111000</td>
<td>1110110</td>
</tr>
<tr>
<td>14</td>
<td>1011010</td>
<td>1010110</td>
<td>1011100</td>
<td>1111011</td>
</tr>
<tr>
<td>15</td>
<td>1101110</td>
<td>1100111</td>
<td>0101110</td>
<td>1111101</td>
</tr>
<tr>
<td>16</td>
<td>0111101</td>
<td>1110101</td>
<td>1111101</td>
<td>1111111</td>
</tr>
<tr>
<td>17</td>
<td>1110011</td>
<td>1011101</td>
<td>1111101</td>
<td>1111111</td>
</tr>
<tr>
<td>18</td>
<td>1011111</td>
<td>0111111</td>
<td>1111111</td>
<td>1111111</td>
</tr>
</tbody>
</table>
These new $A$-partitions yield new single asymmetric error correcting codes with better code seizes than the existing ones. Table 4.1 shows that the codes of dimensions 14, 15, 16, 17 and 19 are improved. These improvements are mainly due to the use of the new $A$-partitions for $p = 6, 7$ (Table 4.3, Table 4.4). The Code of dimension 15 is constructed using $A$-partition of $p = 7$ and $B$-partition of $q = 8$ instead of using $p = 6$ and $q = 9$. The code of dimension 16 is constructed using $A$-partition of $p = 7$ and $B$-partition of $q = 9$ instead of using $p = 6$ and $q = 10$.

Table 4.5 Proposed codes with the dimensions of $A$- and $B$-partitions

<table>
<thead>
<tr>
<th>$n$</th>
<th>Existing Code</th>
<th>$p$</th>
<th>$q$</th>
<th>Proposed Code</th>
<th>Upper Bound [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1204</td>
<td>6</td>
<td>8</td>
<td>1228</td>
<td>1500</td>
</tr>
<tr>
<td>15</td>
<td>2216</td>
<td>7</td>
<td>8</td>
<td>2288</td>
<td>2828</td>
</tr>
<tr>
<td>16</td>
<td>4232</td>
<td>7</td>
<td>9</td>
<td>4272</td>
<td>5430</td>
</tr>
<tr>
<td>17</td>
<td>8192</td>
<td>7</td>
<td>10</td>
<td>8296</td>
<td>10379</td>
</tr>
<tr>
<td>19</td>
<td>28548</td>
<td>7</td>
<td>12</td>
<td>28688</td>
<td>38008</td>
</tr>
</tbody>
</table>

Table 4.5 shows new sizes of improved codes and the values of $p$ and $q$, which are chosen for $A$-partitions and $B$-partitions respectively, to be used in the Cartesian product method to construct these codes of dimension $p + q$. The cardinalities of $A$-partitions are listed in Table 4.2, while the $B$-partitions are listed in Table 2.2.
Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, new codes of minimum asymmetric distance two, capable of correcting a single asymmetric error, are proposed. The issue of asymmetric errors is relatively new comparing with the symmetric errors. However, many papers have been published in the area of asymmetric errors since the late 1950's due to their increasing
number of applications. Examples include: transmissions in optical fibers, LSI single transistor cell memories, and metal-nitride-oxide-silicon (MNOS) memories.

The construction method of the proposed codes is also presented. This method is based on the Cartesian product of two sets of partitioned codes, which are called $A$-partition and $B$-partition, of smaller dimensions. The method is quite sensitive to the sizes of the smaller partitions. The better partitions are used in this method; the more codewords are constructed. The issue of improving $A$-partition is discussed and new $A$-partitions are obtained for dimensions 6 and 7. Using the new $A$-partitions leads to proposing the new single asymmetric error correcting codes of sizes larger than the existing ones. It is worth noting that the code of dimension 17 obtained here has 8296 codewords, which exceeds the upper index $(2^n / n - 1)$ given in [2].

5.2 Future Work

There are several promising research directions that can be pursued based on the results of this thesis. The followings summarize some interesting directions for future work:

1. Propose a construction method to construct codes capable of correcting $k$ asymmetric errors, for $k > 1$.

2. Designing algorithms for efficient encoding/decoding of the proposed codes.
3. Using the new partitioning algorithm to find better *A-partition* for some other values of $p$. The *A-partition* for $p = 8$ seems to be a promising start.

4. Construct *systematic* asymmetric error correcting codes. For $n = 17$, the proposed code size is more than $2^{13}$. It seems promising, therefore, to construct a systematic code of $2^{13}$ codewords [24].
References


Vita

Raed Yacoub Radwan Shammas

Permanent Address:

Hebron, West Bank
Palestine
Tel. 0097022291722

Email:

raed_alshammas@yahoo.com

Bachelor of Engineering (Computer Engineering)

Palestine Polytechnic University
Hebron, Palestine
1999-2003

Master of Science (COMPUTER SCIENCE)

King Fahd University of Petroleum and Minerals
Dhahran, 31261, Saudi Arabia
2007-2011