AN ABSTRACT OF THE THESIS OF

Madhusudhanan Anantha for the degree of Doctor of Philosophy in Computer Science presented on June 1, 2007.

Title: Gray Codes and Their Applications

Abstract approved: ______________________________

Bella Bose

An n-bit Gray code is an ordered set of all 2^n binary strings of length n. The special property of this listing is that Hamming distance between consecutive vectors is exactly 1. If the last and first codeword also have a Hamming distance 1 then the code is said to be *cyclic*. This dissertation addresses problems dealing with the design and applications of new and existing types of both binary and non-binary Gray codes. It is shown how properties of certain Gray codes can be used to solve problems arising in different domains. New types of Gray codes to solve specific types of problems are also designed. We construct Gray codes over higher integral radices and show their applications. Applications of new classes of Gray codes defined over residue classes of Gaussian integers are also shown. We also propose new classes of binary Gray codes and prove some important properties of these codes.

Gray Codes and Their Applications

by

Madhusudhanan Anantha

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 1, 2007
Commencement June 2008

ACKNOWLEDGMENTS

I would like to thank my parents and my grand mother for being so supportive and helpful throughout the course of my doctoral study. Without their support none of this would have been possible. I would also like to record my sincere thanks to the invaluable and able help and guidance of my first teacher, my late grand father. He was instrumental in directing me to choose computer science as my field of study. My extended family has also played an important role in helping me during my study and I convey my sincerest thanks to them.

I would like to express my sincere appreciation to my major professor, Dr. Bose, for his valuable ideas which put me in the right direction. Also, He was available anytime despite his busy schedule to help me out. He was also instrumental in showing me how analyze and work with a research problem.

I would like to express my sincere gratitude to Dr. Mary Flahive, for her valuable help and guidance and a careful reading of my thesis to improve the quality of presentation and for serving in my committee.

I would also like to express my sincere gratitude to Dr. Ben Lee and Dr. Thinh Nguyen, for serving in my committee.

I would like to convey my sincere thanks to my girl friend, Melissa Morales-Warming. I would also like thank my fellow graduate students and friends, Madhusudhanan Srinivasan, Sriraam Natarajan, Robin Abraham for their help in making my stay at corvallis all the more pleasant and enjoyable with their delightful company.

I want to thank all the helpful staff members of the School of E.E.C.S, particularly, Nancy Brown, Ferne Simendinger for their invaluable assistance in scheduling exams and Nancy Bremner for taking care of my payroll.

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

GRAY CODES AND THEIR APPLICATIONS

CHAPTER 1

INTRODUCTION

An n-bit binary Gray code is an ordered set of all 2^n binary strings of length n. The special property of this listing is that Hamming distance (number of bits that are not equal) between consecutive vectors is exactly 1. If the last and first codewords also have a Hamming distance 1 then the code is said to be *cyclic*. For higher radix (≥ 3) Gray codes, any two consecutive radix k strings differ in exactly one digit by ± 1.

Over the last 4-5 decades binary Gray codes have found applications in diverse areas [21, 38]: VLSI testing [37], signal encoding [33], ordering of documents on the shelves [31], data compression [36], statistics [20], graphics and image processing [2], processor allocation in the hypercube [17], hashing [23], computing the permanent [34], information retrieval [15], puzzles such as the Chinese rings and towers of Hanoi [25], designing efficient combinatorial algorithms [34, 41, 35], etc. A large number of Gray code related patents in diverse areas have been granted in the last three decades and almost all of them are, in one way or another, based on the original reflected Gray code ideas.

This dissertation addresses problems dealing with the design of new types of Gray codes for certain problems and new applications of existing types of both binary and non-binary Gray codes over various distance metrics. It is shown how properties of certain Gray codes can be used to solve problems arising in different domains. Design of new types of Gray codes to solve specific types of problems is also shown. Results already obtained in this direction are shown.

This dissertation is organized as follows. Chapter 2 introduces all the background information required to understand the results obtained in this proposal. We then detail the results which we have been obtained so far in Chapters 3, 4, 5, 6, 7 and 8. We then conclude in the final section.

In Chapter 3, we show how the transition sequence of the Binary Reflected Gray code can be used to identify the type of error in unidirectional channels when Bose-Lin codes are used. This is an interesting connection between the characteristics of two seemingly unrelated constructions.

Chapter 4 discusses new constructions of mixed radix Gray codes and shows how they correspond to Hamiltonian cycles in a corresponding mixed radix toroidal network in Lee distance metric. The construction of independent sets of Gray codes and how these correspond to edge disjoint Hamiltonian cycles in their corresponding tori is shown when the number of dimensions, n, is a power of 2.

We show how the minimal change property of Gray codes proposed in Chapter 4 can be used to obtain efficient algorithms to visit each of the $k_n \times k_{n-1} \times k_2 \times k_1$ length n mixed radix vectors in turn in Chapter 5.

In Chapter 6, we define a new class of binary Gray codes using a simple mapping of the digits of a radix 4 Lee distance Gray code to binary vectors. The resultant code is of even length. We prove that the important property of minimal change in the number of bits going from one vector to another of same weight in the new binary Gray code holds. We also show how the mapping function proposed for radix 4 Lee distance Gray code can be extended to higher radices ($k \geq 4$ and $k = 2^r$) for some $r, r > 2$.

The approach we use to solve the problem of finding edge disjoint Hamiltonian cycles in toroidal networks with arbitrary number of dimensions, n, where $n > 0$ and $n \neq 2^r$ is explained in Chapter 7. In particular, we show how we can use the construction of edge disjoint Hamiltonian cycles when $n = 2$ and 3 to recursively solve the problem for arbitrary values of $n \geq 4$.

The problem of defining a subclass of Circulant graphs using Gaussian integers is discussed in Chapter 8. We are interested in embedding a Hamiltonian cycle in this graph and finding edge disjoint Hamiltonian cycles. The proposed solution to this problem is by defining a new class of Gray codes over equivalence classes modulo a Gaussian integer. We show a simple construction of edge disjoint Hamiltonian cycles for this graph as a Gray code over the equivalence classes modulo a Gaussian integer . We also give efficient algorithms for generating these Gray codes.

CHAPTER 2

BACKGROUND

2.1 Automatic Repeat Request (ARQ) protocols

There are two major error control techniques in any data communication: 1) Forward error correction (FEC) and 2) Automatic repeat request (ARQ). ARQ protocols can be classified into three main types: stop-and-wait ARQ, go-back-N ARQ, and selective repeat ARQ [30, 40]. They are briefly summarized below. The proposed error control techniques can be used with any of these basic protocols.

- Stop-and-Wait protocol: This is the simplest of the ARQ protocols. After transmitting the codeword, the transmitter will wait for an acknowledgment. If the transmitter receives ACK, it then sends the next codeword. If either the timeout time expires without receiving an acknowledgment, or the transmitter receives NAK, the transmitter re-transmits the same codeword. This procedure continues until ACK is received. So, buffering is not necessary at both the receiver and the transmitter. The main disadvantage of using this scheme is that the transmitter is idle while waiting for the acknowledgment, resulting in a low throughput performance. Stop-and-wait ARQ is useful in some computer applications such as interprocessor transfer in multiprocessing systems, where the round trip delay is extremely low. Figure 2.1 explains this protocol.

- Go-Back-N ARQ protocol: If there is some buffering available in the transmitter and not necessarily at the receiver, go-back-N ARQ protocol can be used. In this protocol, the transmitter sends the codewords in a continuous stream without waiting for an acknowledgment from the receiver. If the receiver detects an error in a received word, it requests a retransmission for this word by sending NAK to the transmitter. At this point, all subsequent incoming words are ignored until the transmitter retransmits the requested word and the receiver receives it. Therefore, buffering is not necessary at the receiver. When the transmitter resends a word, it also resends all subsequent words (which were ignored by the receiver after detecting the first erroneous word). This makes buffering necessary at the transmitter. Figure 2.2 explains this protocol.

- Selective-Repeat ARQ protocol: If some buffering is available at both the transmitter and the receiver, Selective-Repeat ARQ protocol can be used. In this protocol, the transmitter sends the words in a continuous stream without waiting for acknowledgment from the receiver. If the receiver detects an error in one of the received words, it requests a retransmission for this word by sending NAK to the transmitter. At this point, the transmitter resends the required word and then resumes transmitting the new codewords. So, buffering is necessary at both sides. Figure 2.3 explains this protocol.

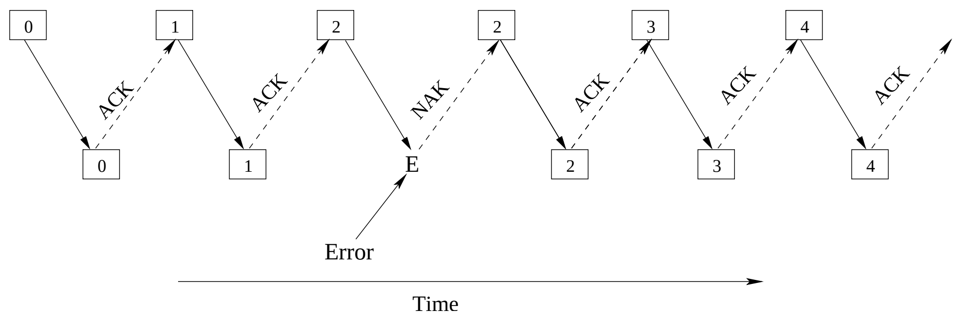


FIGURE 2.1: Stop-and-Wait Protocol.

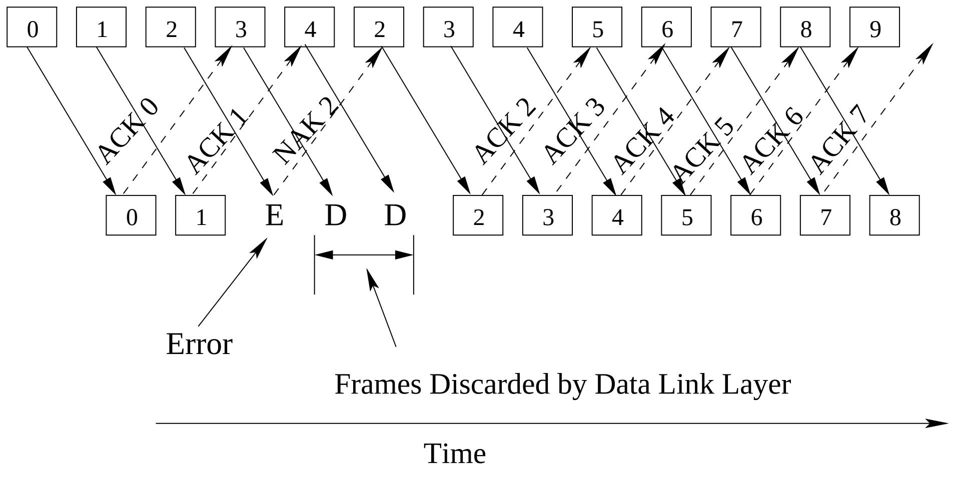


FIGURE 2.2: Go-Back-N Protocol.

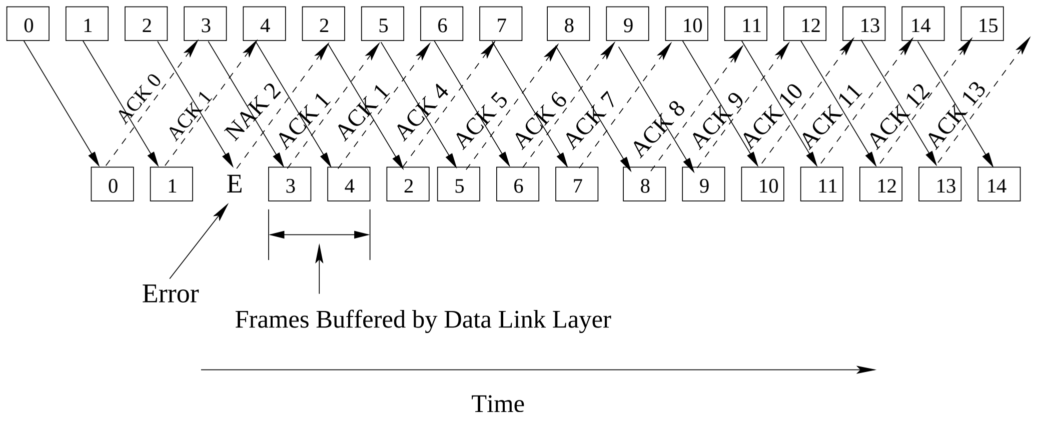


FIGURE 2.3: Selective Repeat Protocol.

2.2 Code Combining

ARQ protocols, as discussed above, retransmit a packet repeatedly until it is received correctly. For noisy channels repeated retransmissions can decrease the throughput efficiency of the system. Packets which cause retransmission requests can be stored

i	Y_i	Z_i
0		0000000000000
1	$0100\underline{0}1\underline{0}010\underline{0}01$	$0100\underline{0}1\underline{0}010\underline{0}01$
2	$01001\underline{0}1010\underline{0}01$	$0100111010\underline{0}01$
3	$01001\underline{0}10\underline{0}0101$	0100111010101

TABLE 2.1: Code Combining for asymmetric errors

and combined with additional retransmissions of the packet, thus creating a single packet that is likely to be the correct version of the transmitted word. In code combining, individual symbols from identical copies of a packet are combined to create a packet with more reliable constituent symbols.

Consider for example, code combining for *asymmetric errors*. In case of binary asymmetric errors only one type of error, that is, either $1 \rightarrow 0$ or $0 \rightarrow 1$ is possible. This type of error characteristic is typical of optical channels where only $1 \rightarrow 0$ errors are possible. In this case, we use a register Z initialized to the all-zero vector. Each time an erroneous word, Y, is received, a bit-by-bit OR operation is performed with Z and the result is stored back in Z. This stops until Z or Y is free of errors. Note that Z stores the results of previous retransmissions of Y.

Example 1. *Let the transmitted word be* $X = 0100111010101$ *and* Y_i*,* $i = 1, 2, 3, \ldots$ *be the received word at the i-th time assuming errors in all* Y_j*,* $j = 1, 2, 3, \ldots, i-1$*. Each time a received* Y *has errors code combining is done as shown in Table 2.1. Thus, after 3 transmissions all errors are corrected. The underlined bits in each transmission are the bits in error.*

In the case of *symmetric errors* only portions of a packet which was received in error are re-transmitted and in these cases code combining is not used.

2.3 Lee Distance Metric

2.3.1 Single Radix Case

Let $A = (a_n a_{n-1} \cdots a_1)$ be an n digit radix k vector. The Lee weight of A is defined as

$$\begin{aligned} W_L(A) &= \sum_{i=1}^{n} |a_i|, \\ \text{where} \qquad |a_i| &= \min(a_i, k - a_i) \end{aligned}$$

The Lee distance between two vectors A and B is denoted by $D_L(A, B)$ and is defined to be $W_L(A - B)$. That is, the Lee distance between two vectors is the Lee weight of their digit-wise difference **mod** k. For example, when $k = 4$, $W_L(321) = \min(3, 4 - 3) + \min(2, 4 - 2) + \min(1, 4 - 1) = 1 + 2 + 1 = 4$, and $D_L(123, 321) = W_L(202) = 4$.

Let $D_H(A, B)$ be the Hamming distance between two vectors A and B, i.e. the number of positions in which A and B differ. Then $D_L(A, B) = D_H(A, B)$ when $k = 2$ or 3, and $D_L(A, B) \geq D_H(A, B)$ when $k > 3$. In the rest of the paper, if the value of k is not mentioned then it is assumed that $k \geq 3$; when $k = 2$, it will be explicitly mentioned.

In a Lee distance Gray code C, the set of k^n vectors are arranged in a sequence such that two adjacent vectors are at a Lee distance one. In this sequence, if the first and the last words are also at a distance of 1, then the code is called a *cyclic Gray code*; if not it is a *non-cyclic Gray code*. Cyclic Lee distance Gray codes correspond to Hamiltonian cycles in k-ary n-cube [12].

2.3.2 Mixed Radix Case

Let $A = a_n a_{n-1} \cdots a_1$ be an n-dimensional mixed radix vector over Z_K, where $K = k_n \times k_{n-1} \times \cdots \times k_1$, i.e., all $x_i \in Z_{k_i}$, for $i = 1, 2, \cdots, n$. The Lee weight of A in mixed radix notation is defined as

$$\begin{aligned} W_L(A) &= \sum_{i=1}^{n} |a_i|, \\ \text{where} \qquad |a_i| &= \min(a_i, k_i - a_i), \text{for } \ i = 1, 2, \cdots, n. \end{aligned}$$

The Lee distance between two vectors A and B is denoted by $D_L(A, B)$ and is defined to be $W_L(A - B)$. That is, the Lee distance between two vectors is the Lee weight of their digit-wise difference. In other words, $D_L(A, B) = \sum_{i=1}^{n} \min(a_i - b_i, b_i - a_i)$, where $(a_i - b_i)$ and $(b_i - a_i)$ are **mod** k_i operations. For example, when $K = 4 \times 6 \times 3$, $W_L(321) = \min(3, 4 - 3) + \min(2, 6 - 2) + \min(1, 3 - 1) = 1 + 2 + 1 = 4$, and $D_L(120, 321) = W_L(202) = 3$.

Let $D_H(A, B)$ be the Hamming distance between two vectors A and B, i.e. the number of positions in which A and B differ. Then $D_L(A, B) = D_H(A, B)$ when $k_i = 2$ or 3, for all i, and $D_L(A, B) \geq D_H(A, B)$ when $k_i > 3$ for some i. In the rest of the paper, if the value of k_i is not mentioned then it is assumed that $k_i \geq 3$; when $k_i = 2$, it will be explicitly mentioned.

In a Lee distance Gray code C, the set of $K = k_n \times k_{n-1} \times \cdots \times k_1$ vectors over $Z_{k_n} \times Z_{k_{n-1}} \times \cdots \times Z_{k_1}$ are arranged in a sequence such that two adjacent vectors are at a Lee distance one. In this sequence, if the first and the last words are also at a distance of 1, then the code is called a *cyclic Gray code*; if not it is a *non-cyclic Gray code*. Further, when $k_n = k_{n-1} = \cdots = k_1$ the resultant code is called *single radix Gray code*; if $k_i \neq k_j$ for some $i \neq j$ then it is called a *mixed radix Gray code*.

2.3.3 Hamiltonian Cycles in Toroidal Networks

Cyclic single radix and mixed radix Gray codes correspond to Hamiltonian cycles in the k-ary n-cube and the n-dimensional torus $T_{k_n \times k_{n-1} \times \cdots \times k_1}$ graphs, respectively, and these two graphs are described below.

A k-ary n-cube graph (C_n^k) and an n-dimensional torus ($T_{k_n, k_{n-1}, \cdots, k_1}$) are $2n$-regular graphs containing k^n and $k_n \times k_{n-1} \times \cdots \times k_1$ nodes, respectively. Each node in C_n^k is labeled with a distinct n-digit radix-k vector while each node in $T_{k_n \times k_{n-1} \times \cdots \times k_1}$ is labeled with a distinct n-digit mixed radix vector. If u and v are two nodes in the graph, then there is an edge between them *iff* $D_L(u, v) = 1$. From the definition of Lee distance, it can be seen that every node in a C_n^k or a $T_{k_n \times k_{n-1} \times \cdots \times k_1}$ shares an edge with two nodes in every dimension, resulting in a regular graph of degree $2n$. In addition, the shortest path between any two nodes, u and v has length $D_L(u, v)$. Note that C_n^k is an n-dimensional hypercube, Q_n, when $k = 2$; in this case, each node is adjacent to exactly n other nodes. It is easy to verify that cyclic single radix and mixed radix Gray codes corresponds to Hamiltonian cycles in C_k^n and $T_{k_n \times k_{n-1} \times \cdots \times k_1}$, respectively.

2.4 Circulant Graphs

Circulant graphs are a popular interconnection network topology used for decades in the design of computer and telecommunication networks. The term *circulant* comes from the nature of its adjacency matrix; a matrix is Circulant if all its rows are periodic rotations of the first one. Recent research in design of interconnection networks for on-chip multiprocessors [19] has shown that a sub-class of circulant graphs called Dense Gaussian graphs is a suitable topology.

In the remainder of this section, we will briefly introduce circulant graphs and then give a deeper insight in to the properties of Dense Gaussian graphs.

A *circulant* graph with N vertices and jumps $\{j_1, j_2, \ldots, j_m\}$ is an undirected graph in which each vertex n, $0 \leq n \leq N-1$, is adjacent to all the vertices $n \pm j_i$, with $1 \leq i \leq m$. We denote this graph as $C_N(j_1, j_2, \ldots, j_m)$. It is clear that a circulant graph $C_N(j_1, j_2, \ldots, j_m)$ is connected if and only if $\gcd(j_1, j_2, \ldots, j_m, N) = 1$

In a degree four circulant graph there can be, at most, $4d$ different nodes at distance d from any node. Thus, for a given diameter k the maximum number of nodes of a $C_N(j_1, j_2)$ graph is:

$$N \leq 1 + 4\sum_{d=1}^{k} d = 1 + 4\left(\frac{k(k+1)}{2}\right) = 2k^2 + 2k + 1 \tag{2.1}$$

Graphs containing such a maximum number of nodes can be denoted as *dense* degree 4 circulants. Different authors have shown that $C_N(k, k+1)$ graphs with $N = 2k^2 + 2k + 1$ are dense degree 4 circulants.

We now show why such a graph with maximum number of nodes can exist. We first need the following definitions. For integers a and a' , we denote $[a, a']$ the interval of all integers n with $a \leq n \leq a'$. Under this notation the vertex set, V, is $[0, N-1]$.

We also define the set ξ_l as

$$\xi_l = \{(x, y) \in \mathbb{Z}^2 ||x| + |y| \leq l\} \tag{2.2}$$

where x and y are integers, and the accessing function f_N from $\mathbb{Z}^2$ to V as follows

$$f_N(x, y) = xj_1 + yj_2 \mod N \tag{2.3}$$

The value given by the accessing function f_N on (x, y) is the node reached from node 0 after x number of j_1-hops and y number of j_2-hops in the graph $C_N(j_1, j_2)$. Given the diameter of the $C_N(j_1, j_2)$ graph, k. The set, ξ_k, is the domain that should be considered for f_N to find out all those nodes that are within the distance k from node 0. The following Lemma is shown here for the sake of completeness.

Lemma 2.4.1. *[6] For each k, the cardinality of ξ_k is $2k^2 + 2k + 1$. Therefore, a circulant graph of degree 4 and diameter k cannot have more than $2k^2+2k+1$ nodes.*

Proof. [6] For each x with $-k \leq x \leq k$ there are $2(k - |x|) + 1$ valid values of y to have $(x, y) \in \xi_k$. Thus the cardinality of ξ_k is

$$\sum_{i=-k}^{k} 2(k - |i|) + 1 = 2k^2 + 2k + 1 \tag{2.4}$$

□

Therefore, we can obtain a circulant graph with $j_1 = k$ and $j_2 = k + 1$ with $N = 2k^2 + 2k + 1$. Note that $k > 1$ and $\text{GCD}(k, k + 1) = 1$.

For our purposes we consider just circulant graphs $C_N(j_1, j_2)$ with $N = j_1^2 + j_2^2$. In these particular cases, the circulant is connected if and only if $\gcd(j_1, j_2) = 1$. As $2k^2 + 2k + 1 = k^2 + (k + 1)^2$, dense circulant graphs $C_{k^2+(k+1)^2}(k, k + 1)$ are included in this family as a special member.

CHAPTER 3

CODE COMBINING FOR UNIDIRECTIONAL ERRORS

The main idea is briefly explained below. Let X be the transmitted codeword and let Y_i, $i = 1, 2, \ldots,$ be the received word at the i-th time, assuming errors in each Y_j's, for all $j = 1, 2, \ldots, i-1$. It is assumed that the errors are independent from one transmission to the next. The code combining operation is done as follows.

$$\begin{aligned} Z_0 &= (0\ldots0), \quad \left.\begin{array}{c} Z_i = Y_i \vee Z_{i-1} \\ Z'_i = Z'_{i-1} \end{array}\right\} \text{ for } 1 \to 0 \text{ errors, and} \qquad (3.1) \\ Z'_0 &= (1\ldots1), \quad \left.\begin{array}{c} Z'_i = Y_i \wedge Z'_{i-1} \\ Z_i = Z_{i-1} \end{array}\right\} \text{ for } 0 \to 1 \text{ errors, for } i = 1, 2, 3, \ldots. \end{aligned}$$

Here, $\vee$ and $\wedge$ are digit-by-digit OR and AND operations, respectively. The receiver requests retransmission of a codeword until Z_i or Z'_i is not in error. Code combining can reduce the average number of retransmissions in noisy channels [32].

Example 2. *We consider a system using a binary Borden code [10] of length 10, capable of detecting up to 4 unidirectional errors. Thus, the weights of the codewords are 0, 5 and 10. Suppose a codeword of weight 5, 1111100000, is transmitted. When no more than 2 unidirectional errors occur, the received word will be of weight 3, 4, 5, 6 or 7. So, a received word of any of these weights can only correspond to a transmitted codeword of weight 5. The receiver can also easily decide if $1 \to 0$ or $0 \to 1$ errors have occurred. Then, the bit-by-bit OR or bit-by-bit AND operations*

i	Y_i	Z_i	Z'_i	error type
0		00000 00000	11111 11111	
1	$111\underline{00}\ 00000$	$111\underline{00}\ 00000$	11111 11111	$1 \to 0$
2	$11111\ \underline{1}0\underline{1}00$	$111\underline{00}\ 00000$	$11111\ \underline{1}0\underline{1}00$	$0 \to 1$
3	$11\underline{0}1\underline{0}\ 00000$	$1111\underline{0}\ 00000$	$11111\ \underline{1}0\underline{1}00$	$1 \to 0$
4	$\underline{00}111\ 00000$	11111 00000	$11111\ \underline{1}0\underline{1}00$	$1 \to 0$

TABLE 3.1: Unidirectional error correction example

can be performed as shown in Table 3.1. Thus, after 4 transmissions, the errors are corrected.

Note that after each step of code combining, the number of errors in the resultant word will either decrease or remain the same but will never increase.

The main advantage of the t-unidirectional error detecting (t-UED) codes is that by using only a few check bits, a large (exponential, in terms of these check bits) number of asymmetric/unidirectional errors can be detected. This is not the case for symmetric error detection. In fact, a t (symmetric) error detecting code of length n requires approximately $\frac{t}{2}\log_2 n$ check bits, whereas a t-UED code, regardless of the length of the code, requires only $O(\log t)$ check bits. Thus, the proposed error control techniques provide low cost alternative methods.

The operations required for code combining in case of unidirectional errors are as already defined above. The hardware logic circuits to perform these operations are shown in Figure 3.1. A system using an ARQ protocol with a t-UED code is capable of correcting up to $E_{max} = \left\lfloor \frac{t}{2} \right\rceil$ unidirectional errors. This is achieved by code combining. Before proving this in Theorem 3.0.2, we state some required definitions and a useful theorem.

Let X and X' be any two n-bit vectors. Then X and X' are *unordered* if $N(X, X') \geq 1$ and $N(X', X) \geq 1$, where $N(Y, Z)$ is the number of 1 to 0 crossovers from an n-bit vector Y to another n-bit vector Z. Then the Hamming distance between Y and Z is $D_H(Y, Z) = N(Y, Z) + N(Z, Y)$. If $N(X, Y) = 0$ we say Y *covers* X.

Theorem 3.0.1. *[10, 9] A code $\mathbb{C}$ is capable of detecting t unidirectional errors iff $\forall X, X' \in \mathbb{C}$, X and X' are either unordered or $D_H(X, X') \geq t + 1$.*

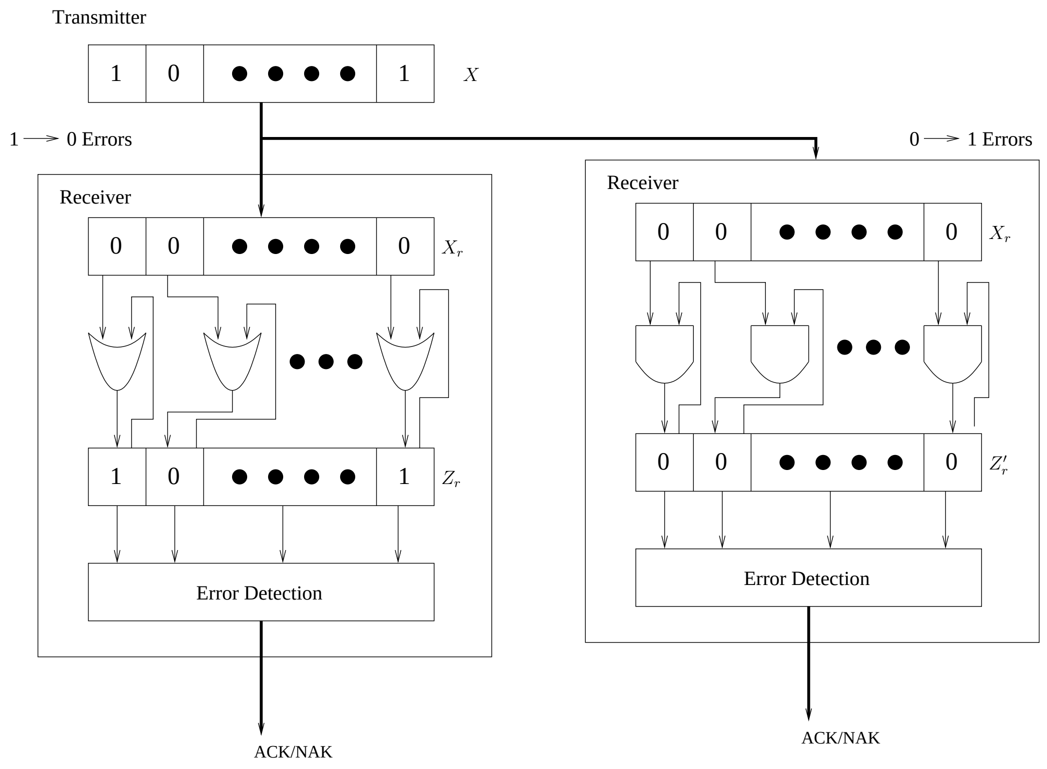


FIGURE 3.1: Code Combining for Unidirectional Errors

Theorem 3.0.2. *Let $\mathbb{C}$ be a t-UED code used with diversity combining operations given by equation (1). This scheme can correct up to $E_{max} = \lfloor \frac{t}{2} \rfloor$ unidirectional errors.*

Proof. At the receiver, the decoder must know whether the received word has $1 \to 0$ errors or $0 \to 1$ errors, if any, and then perform the OR or AND diversity combining.

Let $X \in \mathbb{C}$ be the transmitted codeword and Y be the received word. If the errors are of the $1 \to 0$ type and their number $e \leq \lfloor \frac{t}{2} \rfloor$, then X covers Y and $D_H(X, Y) = e$. Further there is no $X' \in \mathbb{C}$ such that Y covers X' and $D_H(Y, X') \leq \lfloor \frac{t}{2} \rfloor$. In order to prove this, let us assume that it does cover some $X' \in \mathbb{C}$ with $D_H(Y, X') \leq \lfloor \frac{t}{2} \rfloor$. Then, X will also cover X' and $D_H(X, X') \leq t$. But this contradicts the condition required for detecting t unidirectional errors.

Thus, if the number of unidirectional errors is less than or equal to $E_{max} = \lfloor \frac{t}{2} \rfloor$, the decoder can decide what type of errors, $1 \to 0$ or $0 \to 1$, have occurred in the received word. Then, these errors can be corrected using OR or AND diversity combining operations. □

Even though the above theorem shows the error correcting capabilities of any t-UED code, some simple schemes need to be developed in order to decide the type of errors within a received word, and then correct these errors. We now show how this decision can be taken for various unidirectional codes.

3.1 Determining Type of Error

3.1.1 Constant Weight Codes

These codes are also known as m-out-of-n codes. In these codes, each and every codeword has weight m, where $m \leq n$ and n is the size of the codeword. The number of codewords is $\binom{n}{m}$. These codes are capable of detecting any combination of unidirectional errors. Let X' be the received word and $w(X')$ be the weight of X'. The type of error can be identified as follows

$$\begin{aligned} w(X') > m &\Rightarrow 0 \rightarrow 1 \text{ error} \\ w(X') < m &\Rightarrow 1 \rightarrow 0 \text{ error} \end{aligned}$$

3.1.2 Berger Codes [5],[24]

These are *systematic* All Unidirectional Error Detecting(AUED) codes. The check value of the codeword is obtained by counting the number of 0's in the information part and expressing the value in binary. Let k be the number of information bits and r be the number of check bits. Then $r = \lceil \log_2(k+1) \rceil$. Let $X = IC$ be the transmitted word where I is the information part and C is the check part. The syndrome S from the received word $X' = I'C'$ is calculated as follows

$$S = I'_0 - \nu(C') \tag{3.2}$$

where $\nu(C')$ is the decimal value of the check and I'_0 is the number of 0's in I''s. The type of error can be identified as follows

$$\begin{aligned} I'_0 - \nu(C') = 0 &\Rightarrow \text{ no error} \\ I'_0 - \nu(C') > 0 &\Rightarrow 1 \rightarrow 0 \text{ error} \\ I'_0 - \nu(C') < 0 &\Rightarrow 0 \rightarrow 1 \text{ error} \end{aligned}$$

3.1.3 Borden Codes [10]

Borden codes are *optimal* t-Unidirectional Error Detecting (t-UED) *non-systematic* codes. The value t is the maximum number of errors that can be detected. For a code length n, all the codewords have weight, w where

$$w \equiv \left\lfloor \frac{n}{2} \right\rfloor \text{ mod } (t+1) \tag{3.3}$$

Example 3. *When $n = 20$ and $t = 4$,*

$$w \equiv 10 \ mod \ 5 \equiv 0 \ mod \ 5 \equiv 15 \ mod \ 5 \equiv 5 \ mod \ 5 \equiv 20 \ mod \ 5 \tag{3.4}$$

Thus, this code consists of 20 bit vectors with weights 0, 5, 10, 15 and 20. The number of codewords, n_c, is

$$n_c = \binom{20}{0} + \binom{20}{5} + \binom{20}{10} + \binom{20}{15} + \binom{20}{20} \tag{3.5}$$

Since this code is a 4 error detecting code, by Theorem 3.0.2 it can be seen that up to 2 errors can be corrected using diversity combining. Note that the weight difference between any pair of codewords is at least 5. For a vector with weight x where $x \in \{0, 5, 10, 15, 20\}$, if there are at most two (0 → 1) errors then the weight increases up to $x+2$. For a weight $x+5$ codeword there must be at least 3 (1 → 0) errors to get to $x+2$. Thus for any codeword if there are at most 2 (0 → 1) errors then the weight of the received word is unique. Similarly in case of (1 → 0) errors, the minimum weight possible is $x-2$ and there must be at least 3 (0 → 1) errors from $x-5$ to get to $x-2$. Thus, for a given weight, w_{re}, of the received word, there is a codeword with unique weight, w_c. If $w_c - w_{re} > 0$ then the error is of (1 → 0) type and if $w_c - w_{re} < 0$ it is of (0 → 1) type as long as the number of errors is ≤ 2.

3.2 Determining Error Type For Codes Proposed by Bose and Lin [9]

These are systematic t-UED codes. In the preceding codes deciding whether $0 \rightarrow 1$ or $1 \rightarrow 0$ errors have occurred in the received word is easy. However for the codes given in [9], even though making this decision is simple its derivation is not straight forward. We show how this can be done for codes constructed using Method 1 and Method 2 in [9].

3.2.1 Method 1

Code Construction

Let k be the number of information bits and r be the number of check bits. The length of the codeword is given by $n = k + r$. Let I_0 and I_1 be the number of 0's and 1's respectively in the information bits.

When $r = 2$ or 3, the check symbols are generated as follows. Count the number of 0's in the information part and take mod 2^r; i.e., when $r = 2$, CS $= I_0$ mod 4 and when $r = 3$, CS $= I_0$ mod 8. When $r = 2$ and $r = 3$, the code can detect up to 2 and 3 errors respectively. Thus, the maximum number of errors these codes can correct is $E_{max} = 1$.

For $r \geq 4$, let $(c_{r-1}, c_{r-2}, \ldots, c_0)$ be the r-bit check symbol. Divide the check bits into two parts with c_{r-1}, c_{r-2} in the first part and the remaining $(r-2)$ bits in the other part. The bits in the first part can be only 10 or 01, whereas the remaining $(r-2)$ bits can be any of the 2^{r-2} possible binary $(r-2)$ tuples. Thus, the number of check symbols is $2 \times 2^{r-2} = 2^{r-1}$. The check symbols are generated as follows.

Count the number of 0's in the information symbol and take modulo 2^{r-1}. Express this number as an $r-1$ bit binary number $(c_{r-2}c_{r-3} \ldots c_0)$. Then the check will be $(c_{r-1}c_{r-2} \ldots c_0) = (\overline{c_{r-2}}c_{r-2} \ldots c_0)$ i.e., the MSB of check c_{r-1} is the complement of c_{r-2}. Codes constructed using this method are capable of detecting up to $2^{r-2} + r - 2$ errors. By Theorem 3.0.2, the maximum number of errors that can be corrected using diversity combining is

$$E_{max} = \left\lfloor \frac{2^{r-2} + r - 2}{2} \right\rfloor \tag{3.6}$$

Determining Error Type

Let X be the transmitted word and let Y be the received word. Let I_0 and $I_0{}'$ be the number of 0's in the information parts of X and Y respectively. Let $\nu(C_0)$ and $\nu(C_0{}')$ be the decimal values of the check parts of X and Y respectively.

First the case when $r = 2$ or 3 is considered. Note that $E_{max} = 1$ in both of these cases. Syndrome S is calculated as follows

$$S = (I_0' - \nu(C')) \bmod 2^r \tag{3.7}$$

where $\nu(C)$ maps a binary vector to its decimal value. Since only one error is allowed, it can be either in the information part or in the check. For $1 \rightarrow 0$ error, the syndrome will be:

1. Error in the information part:

$$\begin{aligned} S &\equiv (I_0' - \nu(C')) \bmod 2^r \\ &\equiv ((I_0 + 1) - \nu(C)) \bmod 2^r \\ &\equiv 1 \bmod 2^r \end{aligned}$$

2. Error in the check part:

$$\begin{aligned} S &\equiv (I_0' - \nu(C')) \bmod 2^r \\ &\equiv (I_0 - \nu(C - 2^i)) \bmod 2^r \\ &\equiv 2^i \bmod 2^r \end{aligned}$$

where $i = \{0, \ldots, r-1\}$ for $r = 2, 3$.

Similarly for $0 \rightarrow 1$ error:

1. Error in the information part:

$$\begin{aligned} S &\equiv (I_0' - \nu(C')) \bmod 2^r \\ &\equiv ((I_0 - 1) - \nu(C)) \bmod 2^r \\ &\equiv -1 \bmod 2^r \equiv (2^r - 1) \bmod 2^r \end{aligned}$$

2. Error in the check part:

$$\begin{aligned} S &\equiv (I_0' - \nu(C')) \bmod 2^r \\ &\equiv (I_0 - \nu(C + 2^i)) \bmod 2^r \\ &\equiv -2^i \bmod 2^r \equiv (2^r - 2^i) \bmod 2^r \end{aligned}$$

where $i = \{0, \ldots, r-1\}$ for $r = 2, 3$.

Thus,

1. when $r = 2$,

$$\begin{aligned} S = 1 &\Rightarrow 1 \rightarrow 0 \text{ error} \\ S = 3 &\Rightarrow 0 \rightarrow 1 \text{ error} \\ S = 2 &\Rightarrow c_1' = 0 \Rightarrow 1 \rightarrow 0 \text{ error} \\ & c_1' = 1 \Rightarrow 0 \rightarrow 1 \text{ error} \end{aligned}$$

2. when $r = 3$,

$$\begin{aligned} S = \{1, 2\} &\Rightarrow 1 \rightarrow 0 \text{ error} \\ S = \{6, 7\} &\Rightarrow 0 \rightarrow 1 \text{ error} \\ S = 4 &\Rightarrow c_2' = 0 \Rightarrow 1 \rightarrow 0 \text{ error} \\ & \quad\; c_2' = 1 \Rightarrow 0 \rightarrow 1 \text{ error} \end{aligned}$$

Now we consider the case with $r \geq 4$. In some cases the type of error can be decided by the value of syndrome alone and in other cases by both the syndrome and the received check. When one of the two most significant bits of the check is in error, the error type can be easily decided. In the rest of the analysis, it is assumed that these two bits are not in error.

Again the syndrome S is calculated as

$$S \equiv (I_0' - \nu(C')) \bmod 2^{r-1} \tag{3.8}$$

The values of syndrome are:

1. $1 \rightarrow 0$ Errors.

   - Errors only in the information part: Since $S \equiv (I_0' - \nu(C')) \bmod 2^{r-1}$, S will be in the range , $1 \leq S \leq E_{max} = 2^{r-3} + \left\lfloor \frac{r-2}{2} \right\rfloor$
   - Errors only in the check: In this case $1 \leq S \leq (2^{r-2} - 1)$
   - Errors in both information and check parts: The maximum value of syndrome occurs when all the least significant $(r-2)$ bits of the check are in error and $E_{max} - (r-2)$ information bits are in error. Thus,

$$\begin{aligned} 2 \leq S &\leq (E_{max} - (r-2)) + (2^{r-2} - 1) \\ &= 3.2^{r-3} - \left\lceil \frac{r-2}{2} \right\rceil - 1 \end{aligned}$$

2. $0 \rightarrow 1$ Errors (Here the values of syndromes are the complement of the above cases).

   - Errors only in the information part: Since $S \equiv (I'_0 - \nu(C')) \bmod 2^{r-1}$, S will be in the range

   $$\begin{aligned} (2^{r-1}-1) \geq S &\geq 2^{r-1} - (2^{r-3} + \left\lfloor \frac{r-2}{2} \right\rfloor) \\ &= 3.2^{r-3} - \left\lfloor \frac{r-2}{2} \right\rfloor \end{aligned}$$

   - Errors only in the check: In this case $2^{r-2} + 1 \leq S \leq (2^{r-1} - 1)$

   - Errors in both information and check parts: The maximum value of syndrome occurs when all the least significant $(r-2)$ of the check are in error and $E_{max} - (r-2)$ information bits are in error. Thus,

   $$\begin{aligned} (2^{r-1}-2) \geq S &\geq 2^{r-1} - (E_{max} - 2) - (2^{r-2} + 1) \\ &= 2^{r-3} + \left\lceil \frac{r-2}{2} \right\rceil + 1 \end{aligned}$$

Let $S_{min} = 2^{r-3} + \lceil \frac{r-2}{2} \rceil$ and $S_{max} = 2^{r-1} - 2^{r-3} - \lceil \frac{r-2}{2} \rceil = 3.2^{r-3} - \lceil \frac{r-2}{2} \rceil$

It can be seen from the above calculation that for $0 \rightarrow 1$ errors the syndrome, S, cannot be less than or equal to S_{min} and for $1 \rightarrow 0$ errors S cannot be greater than or equal to S_{max}. Thus we have the following result

Theorem 3.2.1. *For $r \geq 4$ and S satisfying the following conditions, the type of error is given as*

1. $1 \leq S \leq S_{min} \Rightarrow$ *1 → 0 errors,*

2. $S_{max} \leq S \leq 2^{r-1} - 1 \Rightarrow$ *0 → 1 errors.*

When the syndrome is in the range $S_{min} + 1 \leq S \leq S_{max} - 1$, the type of error can be decided based on both the syndrome, S, and the received check.

Before giving the main results, we take an example and explain the theory behind the results. Suppose $r = 8$ and so the code can detect up to 35 errors. Then by Theorem 3.2.1 if the syndrome is between 1 to 35 the errors are of $1 \rightarrow 0$ type and if it is from 93 to 127 they are of $0 \rightarrow 1$ type. Now suppose $S = 36$. For $0 \rightarrow 1$ errors only two possible cases of error values can result in this value of S. They are 63 and 62 (i.e., the least 6 bits of the check must be 111 11x, where x is a don't care condition). In general for $S = 35 + l$ the number of possible values of the check error vector is as shown in Table 3.2.

l	1	2	3	4	5	6	7	8	9	10	11	12	13	…
# of possible error vectors(A)	2	4	4	6	8	8	8	10	12	12	14	16	16	…
Minimum Value of $\nu(C'_{r-2})$ $= 64 - A$	62	60	60	58	56	56	56	54	52	52	50	48	48	…

TABLE 3.2: Possible values of check error vector for $S = 35 + l$

In the above table, $\nu(C'_{r-2})$ represents the decimal value of the least $r - 2$ check bits. For example, when $l = 4$, the 6-bit error value can be 111111, 111110, 111101, 111100, 111011 and 111010 (i.e., the error vector must be one of the six highest 6 bit binary vectors). If the received check bit value is less than 111010 then the total number of errors must be greater than $E_{max} = 17$ to maintain syndrome. Therefore, the the error must be of $1 \rightarrow 0$ type if the received check bit value is less than 111010. Otherwise, it must be of $0 \rightarrow 1$ type. Thus, for a given l, if we can calculate the number of $0 \rightarrow 1$ error patterns that can result in this syndrome, we can decide the type of error. From the above table two important points can be observed.

1. When $l = 2^q - 1$, the number of error patterns is 2^q.

2. The number of times A remains the same for consecutive values of l follows the sequence $\varrho = (1, 2, 1, 3, 1, 2, 1, 4, 1, 2, \ldots)$.

There is a close relationship between the Binary Reflected Gray Code (BRGC) and the sequence ϱ. If we write down the sequence of bit transitions from one vector to the next in a BRGC listing, we get 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, 1, 2, 1, ... Thus the above sequence is same as the bit transition sequence of a BRGC.

Now the main results are given in the following two theorems.

Theorem 3.2.2. *Suppose* $S = (S_{min} + l) \bmod 2^{r-1}$, *where* $S_{min} + l \leq 2^{r-2}$ *i.e.,* $S_{min} < S \leq 2^{r-2}$. *Let* $C' = (c'_{r-1}c'_{r-2} \ldots c'_0)$ *be the received check bits and the decimal value of the least* $(r-2)$ *check bits is* $\nu(C'_{r-2})$. *For* $l = 1$, *the errors are of 0* $\rightarrow$ *1 type if* $\nu(C'_{r-2}) \geq 2^{r-2} - 2$, *otherwise they are of 1* $\rightarrow$ *0 type. For* $l > 1$ *such that the condition* $l > (2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$, *where* b_i*'s are chosen as the maximum values and* $b_1 > b_2 > \ldots > b_j$, *if* $\nu(C'_{r-2}) \geq 2^{r-2} - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 2)$ *then the errors are of 0* $\rightarrow$ *1 type. Otherwise the errors are of 1*$\rightarrow$ *0 type.*

In the above theorem, some explanation is required for what we mean by maximum b_j's. Suppose $l = 29$, even though $29 > (2^4 - 1) + (2^3 - 1)$ and $29 > (2^4 - 1) + (2^2 - 1) + (2^1 - 1)$ they do not satisfy the condition mentioned. The only set of values for b_j's that satisfy the conditions is $29 > (2^4 - 1) + (2^3 - 1) + (2^2 - 1) + (2^1 - 1)$. Since $(2^{b_1} - 1) > (2^{b_2} - 1) + (2^{b_3} - 1) + \ldots + (2^{b_j} - 1)$ for $b_1 > b_2 > \ldots > b_j$ there is a unique set of integers, $b_1, b_2, \ldots, b_j$ such that for a given l, $l > (2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$. Further, suppose $(2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$ and $(2^{d_1} - 1) + (2^{d_2} - 1) + \ldots + (2^{d_i} - 1)$ are two consecutive integers in a sequence, γ, where each integer can be represented in the above form ($\gamma = (1, 3, 4, 7, 8, 10, 11, 15, 16, 18, \ldots)$). Then

$$(2^{d_1} + 2^{d_2} + \ldots + 2^{d_i}) - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j}) = 2. \tag{3.9}$$

The following example explains Theorem 3.2.2.

Example 4. *Let $r = 8$. Suppose the syndrome $S = 56$. Since in this case $S_{min} = 35$, the value of l is 21. Where $21 = l > (2^4 - 1) + (2^2 - 1) + (2^1 - 1)$. Let the received check be 01001100. Since $(001100)_2 = 12 < 2^6 - (2^4 + 2^2 + 2^1) + 2 = 40$ the errors are of the $1 \rightarrow 0$ type. On the other hand, if the received check is 01011110 then the errors must be of the $0 \rightarrow 1$ type. This is because $(011110)_2 = 50 > 2^6 - (2^4 + 2^2 + 2^1 + 2) = 40$.*

Theorem 3.2.3. *Suppose $S = (S_{max} - l) \bmod 2^{r-1}$, where $S_{max} - l > 2^{r-2}$ i.e., $2^{r-2} < S < S_{max}$. Let $C' = (c'_{r-1} c'_{r-2} \ldots c'_0)$ be the received check bits and the decimal value of the least $(r-2)$ check bits be $\nu(C'_{r-2})$. For $l = 1$, the errors are of $1 \rightarrow 0$ type if $\nu(C'_{r-2}) < 2$, otherwise it is of $0 \rightarrow 1$ type. For $l > 1$ such that the condition $l > (2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$, where b_i's are chosen as the maximum values and $b_1 > b_2 > \ldots > b_j$, if $\nu(C'_{r-2}) < (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 2)$ then the errors are of $1 \rightarrow 0$ type. Otherwise the errors are of $0 \rightarrow 1$ type.*

In the rest of this section we will prove Theorem 3.2.2. The proof for Theorem

3.2.3 is similar to that of Theorem 3.2.2. The following lemmas are useful for proving the theorems.

Lemma 3.2.1. *Let x be an integer that can be represented using q bits and its weight is $w(x)$. Then the weight of the integer $x + 2^q$ will be $1 + w(x)$.*

Proof. Suppose $X = (x_{q-1}x_{q-2}\dots x_0), x_i \in \{0, 1\}$. Then, $2^q + X = (1x_{q-1}x_{q-2}\dots x_0)$. Thus, $w(2^q + X) = 1 + w(X)$. □

Lemma 3.2.2. *Suppose $S = (S_{min} + l) \bmod 2^{r-1}$, where $1 < l \leq 2^{r-2}$ and $l = (2^{b_1} - 1) + (2^{b_2} - 1) + \dots + (2^{b_j} - 1)$, where b_i's are such that $b_1 > b_2 > \dots > b_j$. Let $C' = (c'_{r-1}c'_{r-2}\dots c'_0)$ be the received check bits and the decimal value of the least $(r-2)$ check bits be $\nu(C'_{r-2})$. Then if $\nu(C'_{r-2}) \geq 2^{r-2} - (2^{b_1} + 2^{b_2} + \dots + 2^{b_j})$ then the errors are of 0 → 1 type. Otherwise, it is of 1 → 0 type.*

Proof. Suppose the errors are of $0 \rightarrow 1$ type. Let e_I be the number of errors in the information part and E be the check error vector i.e., $C' = C \oplus E$. Then, syndrome, S, will be

$$\begin{aligned} S = (S_{min} + l) &= (I'_0 - \nu(C')) \bmod 2^{r-1} \\ &= (I_0 - e_I - (\nu(C) + \nu(E))) \bmod 2^{r-1} \\ &= -(e_I + \nu(E)) \bmod 2^{r-1} \end{aligned}$$

Since $e_I + \nu(E) < 2^{r-1}$, we have

$$S_{min} + l = 2^{r-1} - (e_I + \nu(E)) \tag{3.10}$$

Thus $e_I = 2^{r-1} - S_{min} - l + \nu(E)$. Suppose $\nu(E) = 2^{r-2} - x$. Then

$$\begin{aligned} e_I &= 2^{r-1} - S_{min} - l - (2^{r-2} - x) \\ &= 2^{r-2} - 2^{r-3} - \left\lceil \frac{r-2}{2} \right\rceil - l + x \\ &= (E_{max} - l + x) - (r-2) \end{aligned}$$

where $E_{max} = 2^{r-3} + \left\lfloor \frac{r-2}{2} \right\rfloor$ is the error correcting capability of the code. Then, the total number of errors will be

$$e_I + w(2^{r-2} - x) = (E_{max} - l + x) - (r-2) + w(2^{r-2} - x) \qquad (3.11)$$

Note that, for the given value of l, the total number of errors must be $\leq E_{max}$. The following table shows, for a given value of the syndrome and the error vector value, the number of errors in the check, the number of errors in the information part and the total number of errors. The error vector values in the first column are in decreasing order starting with $2^{r-2} - 1$. Thus, the values in the third column, the number of errors in the information part, are in increasing order. The number of errors in the check part (column 2) can be obtained using Lemma 3.2.1. The values in column 4, the total number of errors, are monotonically increasing. From the table it can be seen that if the error vector value is $2^{r-2} - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j})$ then the total number of errors is $E_{max} - b_j < E_{max}$ since $b_j \geq 1$. Thus it is a valid combination. If the error vector value is $2^{r-2} - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 1)$ then the total number of errors is $E_{max} + 1 > E_{max}$. This value of check cannot, thus, be included. Therefore, the received check value must be $\geq 2^{r-2} - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j})$ for $0 \rightarrow 1$ errors if the total number of errors $\leq E_{max}$.

Value of Check Error	Number of Errors in Check	Number of Errors in Info	Total Number of Errors
$2^{r-2} - 1$	$r-2$	$(E_{max} - l + 1)$ $-(r-2)$	$(E_{max} - l + 1)$
$2^{r-2} - 2$	$r-3$	$(E_{max} - l + 2)$ $-(r-2)$	$(E_{max} - l + 1)$
$2^{r-2} - 3$	$r-3$	$(E_{max} - l + 3)$	$(E_{max} - l + 2)$

Continued on next page

Value of Check Error	Number of Errors in Check	Number of Errors in Info	Total Number of Errors
		$-(r-2)$	
$2^{r-2}-4$	$r-4$	$(E_{max}-l+4)$ $-(r-2)$	$(E_{max}-l+2)$
$2^{r-2}-5$	$r-3$	$(E_{max}-l+5)$ $-(r-2)$	$(E_{max}-l+4)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^{r-2}-2^{b_1}$	$r-2-b_1$	$(E_{max}-l+2^{b_1})$ $-(r-2)$	$(E_{max}-l+$ $2^{b_1}-b_1)$
$2^{r-2}-2^{b_1}-1$	$(r-3)$	$(E_{max}-l+2^{b_1}+1)$ $-(r-2)$	$(E_{max}-l+2^{b_1})$
$2^{r-2}-2^{b_1}-2$	$(r-4)$	$(E_{max}-l+2^{b_1}+2)$ $-(r-2)$	$(E_{max}-l+2^{b_1})$
$2^{r-2}-2^{b_1}-3$	$(r-4)$	$(E_{max}-l+2^{b_1}+3)$ $-(r-2)$	$(E_{max}-l+2^{b_1}+1)$
$2^{r-2}-2^{b_1}-4$	$(r-5)$	$(E_{max}-l+2^{b_1}+4)$ $-(r-2)$	$(E_{max}-l+2^{b_1}+1)$
$2^{r-2}-2^{b_1}-5$	$(r-4)$	$(E_{max}-l+2^{b_1}+5)$ $-(r-2)$	$(E_{max}-l+2^{b_1}+3)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^{r-2}-2^{b_1}-2^{b_2}$	$(r-b_2-3)$	$(E_{max}-l+2^{b_1}+2^{b_2})$ $-(r-2)$	$(E_{max}-l+2^{b_1}$ $+2^{b_2}-b_2-1)$
$2^{r-2}-2^{b_1}$	$(r-4)$	$(E_{max}-l+2^{b_1}+$	$(E_{max}-l+2^{b_1}$

Continued on next page

Value of Check Error	Number of Errors in Check	Number of Errors in Info	Total Number of Errors
$-2^{b_2} - 1$		$2^{b_2}) - (r-2)$	$+2^{b_2} - 1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^{r-2} - 2^{b_1}$ $-2^{b_2} - 2^{b_3}$	$(r - b_3 - 4)$	$(E_{max} - l + 2^{b_1}+$ $2^{b_2} + 2^{b_3})$ $-(r-2)$	$(E_{max} - l + 2^{b_1}$ $+2^{b_2} + 2^{b_3}-$ $b_3 - 2)$
$2^{r-2} - 2^{b_1}-$ $2^{b_2} - 2^{b_3} - 1$	$(r-5)$	$(E_{max} - l + 2^{b_1}$ $+2^{b_2} + 2^{b_3} + 1)$ $-(r-2)$	$(E_{max} - l + 2^{b_1}+$ $2^{b_2} + 2^{b_3} - 2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^{r-2} - 2^{b_1} - 2^{b_2}$ $-2^{b_3} - 2^{b_4}$	$(r - b_4 - 5)$	$(E_{max} - l + 2^{b_1}$ $+ \ldots + 2^{b_4})$ $-(r-2)$	$(E_{max} - l + 2^{b_1} + \ldots$ $+2^{b_4} - b_4 - 3)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^{r-2} - 2^{b_1}-$ $\ldots - 2^{b_{j-1}} - 1$	$(r-2)$ $-(j-1)$	$(E_{max} + 2^{b_1}+$ $\ldots + 2^{b_{j-1}} + 1$ $-(r-2)$	$(E_{max} - l + 2^{b_1}+$ $\ldots + 2^{b_j}) - (j-1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^{r-2} - 2^{b_1}-$ $\ldots - 2^{b_j}$	$(r-2)-$ $(j-1) - b_j$	$(E_{max} - l + 2^{b_1}+$ $\ldots + 2^{b_j}) - (r-2)$	$(E_{max} - l + 2^{b_1}+$ $\ldots + 2^{b_j} - b_j) - (j-1)$ $= (E_{max} - b_j) < E_{max}$
$2^{r-2} - 2^{b_1}-$ $\ldots - 2^{b_j} - 1$	$(r-2) - j$	$(E_{max} - l + 2^{b_1}+$ $\ldots + 2^{b_j} + 1)$	$(E_{max} - l + 2^{b_1}+$ $\ldots + 2^{b_j} + 1 - j)$

Continued on next page

Value of Check Error	Number of Errors in Check	Number of Errors in Info	Total Number of Errors
		$-(r-2)$	$= E_{max} + 1 > E_{max}$

TABLE 3.3: Valid Combinations of Errors when $l = 2^{b_1} + \ldots + 2^{b_j}$ for $0 \rightarrow 1$ errors

$\square$

Lemma 3.2.3. *Suppose* $(2^{b_1}-1)+(2^{b_2}-1)+\ldots+(2^{b_j}-1)$ *and* $(2^{d_1}-1)+(2^{d_2}-1)+\ldots+(2^{d_i}-1)$ *are two consecutive integers, where* $b_1 > b_2 > \ldots > b_j$ *and* $d_1 > d_2 > \ldots > d_i$*. Let* $l > (2^{b_1}-1)+(2^{b_2}-1)+\ldots(2^{b_j}-1) \geq 1$*. If* $E_{max}-l+2^{b_1}+2^{b_2}+\ldots+2^{b_j}+1-j \leq E_{max}$ *then* $E_{max} - l + 2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} - d_i - (i-1) \leq E_{max}$*.*

Proof. Since $E_{max} - l + 2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 1 - j \leq E_{max}$ if we can show that

$$E_{max}-l+2^{d_1}+2^{d_2}+\ldots+2^{d_i}-d_i-(i-1) \leq E_{max}-l+2^{b_1}+2^{b_2}+\ldots+2^{b_j}+1-j \quad (3.12)$$

then it will imply the result. We can simplify equation (3.12) as follows.

$$\begin{aligned} 2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} - d_i - (i-1) &\leq 2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 1 - j && (3.13) \\ \Rightarrow 2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} - d_i - (i-1) &\leq 2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} - 2 + 1 - j \\ \Rightarrow -d_i - (i-1) &\leq -1 - j \\ \Rightarrow d_i + (i-1) &\geq j + 1 && (3.14) \end{aligned}$$

Now we have to only show that equation (3.14) always holds to complete the proof. Let $V_1 = (2^{d_1} + 2^{d_2} + \ldots + 2^{d_i})$ and $V_2 = (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j})$. Note that b_i's and d_i's are bit indices with 1's in the binary representation of V_2 and V_1 respectively. Thus, j and i are the weights of V_2 and V_1 respectively.

As already noted in equation (3.9)

$$V_1 = V_2 + 2 \tag{3.15}$$

Also, $d_1 > d_2 > \ldots > d_i$ and $b_1 > b_2 > \ldots > b_j$. If we consider a bit vector representing V_2, it can be seen that b_j is the least index with a 1 in it. First we will consider the case when $b_j > 1$.

$V_2 =$	2^{b_1}	2^{b_2}	$\ldots$	2^{b_j}	
$V_2 =$	b_1	b_2	$\ldots$	b_j	$0 \ldots 0$
	1	1	$\ldots$	1	
$V_1 =$	2^{d_1}	2^{d_2}	$\ldots$	2^{d_j}	2^1

TABLE 3.4: Adding 2 to V_2 when $b_j > 1$

Adding 2 to V_2 in this case increases the weight of the resultant vector, $w(V_1) = w(V_2) + 1$ and the least index with a 1 in V_1 is 1 as can be seen from table 3.4. Thus, $i = j + 1$ and $d_i = 1$. Therefore, the result follows.

Now consider the case where $b_j = 1$. Note that b_j cannot be less than 1. Let $x + 1$ be the next lowest index in V_2 which has a value 0. This means there are x 1's from b_j to $x + 1$.

Adding 2 to V_2 when $b_j = 1$ sets all bits $< (x + 1)$ to 0's and the bit $x + 1$ to 1 and hence the least index with a 1 in V_1 is $x + 1$ as can be seen from table 3.5. Thus, $i = (j + x - 1)$. Since $d_i > x$ the result follows. $\square$

Proof of Theorem 3.2.1: Suppose $(2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$ and $(2^{d_1} - 1) + (2^{d_2} - 1) + \ldots + (2^{d_i} - 1)$ are two consecutive integers that can be represented in this

b_1	b_2	$\ldots$		b_j
1	1	$\ldots$	$0\,\underbrace{111}_{x-1}$	1
2^{d_1}	2^{d_2}	2^{d_i}	$0\ldots0$	0

TABLE 3.5: Adding 2 to V_2 when $b_j = 1$

form, where $b_1 > b_2 > \ldots > b_j$ and $d_1 > d_2 > \ldots > d_i$. Then $(2^{d_1} + 2^{d_2} + \ldots + 2^{d_i}) - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j}) = 2$. When $l > (2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$, even with error vector value $2^{r-2} - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 2) = 2^{r-2} - (2^{d_1} + 2^{d_2} + \ldots + 2^{d_i})$, the total number of errors will be $E_{max} - l + 2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 1 - j \leq E_{max}$ and so from Lemma 3.2.3, $E_{max} - l + 2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} - d_i - (i-1) \leq E_{max}$. Now, if the error vector value is $2^{r-2} - (2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} + 1)$ then the total number of errors will be $E_{max} - l + 2^{d_1} + 2^{d_2} + \ldots + 2^{d_i} + 1 - i > E_{max}$ for $(2^{d_1} - 1) + (2^{d_2} - 1) + \ldots + (2^{d_i} - 1) \geq l > (2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$. This completes the proof.

3.2.2 Method 2

Code Construction

Let k be the number of information bits and r be the number of check bits. The length of the code is $n = k + r$ and I_0 and I_1 be the number of 0's and 1's respectively in the information part.

For $r \geq 5$, let $(c_{r-1}c_{r-2}\ldots c_0)$ be the r-bit check symbol. Divide the check bits into two parts with $(c_{r-1}c_{r-2}c_{r-3}c_{r-4})$ in the first part and the remaining $(r-4)$ in the other part. The bits in the first part take any one of the 2-out-of-4 vectors, namely, 0011, 0101, 0110, 1001, 1010, or 1100, whereas the last four bits take any one among the 2^{r-4} binary $(r-4)$ tuples. Thus, there are $6 \times 2^{r-4}$ distinct check symbols. Let $CS = I_0 \bmod (6 \times 2^{r-4})$. Thus, CS is $(r-1)$ bits long. The 3 most significant bits of CS can be $\{000, 001, 010, 011, 100, 101\}$ in binary or $\{0, 1, 2, 3, 4, 5\}$ in decimal. Now define any one-to-one function f from $\{0, 1, 2, 3, 4, 5\}$ to the 2-out-of-4 code. One simple function is $f(i) \leq f(j)$ for $i < j$ where $0 \leq i, j \leq 5$; i.e., $f(000) = 0011$, $f(001) = 0101$, $f(010) = 0110$, $f(011) = 1001$, $f(100) = 1010$ and $f(101) = 1100$. The concatenation of these 2-out-of-4 codes to the least significant $(r-4)$ bits gives us the check symbol CS.

Codes constructed using this method are capable of detecting up to $5.2^{r-4} + (r-4)$ unidirectional errors. By Theorem 3.0.2, the maximum number of errors that can be corrected using diversity combining is

$$E_{max} = \left\lfloor \frac{5.2^{r-4} + r - 4}{2} \right\rfloor \tag{3.16}$$

Determining Error Type

Let X be the transmitted word and let Y be the received word. Let I_0 and $I_0{}'$ be the number of 0's in the information parts of X and Y respectively. Let $\nu(C_0)$ and $\nu(C_0{}')$ be the decimal values of the check parts of X and Y respectively.

For $r \geq 5$, in some cases the type of error can be decided by the value of the syndrome alone and in other cases by both the syndrome and the received check. When any of the four most significant bits are in error the type of error can be easily deduced as shown in Section 3.1.1. This is because the four most significant bits form a 2-out-of-4 code. In the rest of the analysis, it is assumed that these four bits are not in error. The syndrome is calculated as

$$S \equiv (I_0' - \nu(C')) \bmod 6.2^{r-4} \tag{3.17}$$

The values of syndrome are:

1. $1 \rightarrow 0$ Errors.

   - Errors only in the information part: Since $S \equiv (I_0' - \nu(C')) \bmod 6.2^{r-4}$, S will be in the range , $1 \leq S \leq E_{max} = 5^{r-5} + \left\lfloor \frac{r-4}{2} \right\rfloor$
   - Errors only in the check: In this case $1 \leq S \leq (2^{r-4} - 1)$
   - Errors in both information and check parts: The maximum value of syndrome occurs when all the least significant $(r-4)$ bits of the check are in error and $E_{max} - (r-4)$ information bits are in error. Thus,

$$\begin{aligned} 2 \leq S &\leq (E_{max} - (r-4)) + (2^{r-4} - 1) \\ &= 7.2^{r-5} - \left\lceil \frac{r-4}{2} \right\rceil - 1 \end{aligned}$$

2. $0 \rightarrow 1$ Errors (Here the values of syndromes are complement of the above cases).

   - Errors only in the information part: Since $S \equiv (I_0' - \nu(C')) \bmod 6.2^{r-4}$, S will be in the range

$$\begin{aligned} (6.2^{r-4} - 1) \geq S &\geq 6.2^{r-4} - (5^{r-5} + \left\lfloor \frac{r-4}{2} \right\rfloor) \\ &= 7.2^{r-5} - \left\lfloor \frac{r-4}{2} \right\rfloor \end{aligned}$$

   - Errors only in the check: In this case $5.2^{r-4} + 1 \leq S \leq (6.2^{r-4} - 1)$
   - Errors in both information and check parts: The maximum value of syndrome occurs when all the least significant $(r-2)$ of the check are in error

and $E_{max} - (r-4)$ information bits are in error. Thus,

$$\begin{aligned}(6.2^{r-4} - 2) \geq S &\geq 6.2^{r-4} - (E_{max} - 4) - (2^{r-4} + 1) \\ &= 5.2^{r-4} + \left\lceil \frac{r-4}{2} \right\rceil + 1\end{aligned}$$

Let $S_{min} = 5.2^{r-5} + \lceil \frac{r-4}{2} \rceil$ and $S_{max} = 6.2^{r-4} - 5.2^{r-5} - \lceil \frac{r-4}{2} \rceil = 7.2^{r-5} - \lceil \frac{r-4}{2} \rceil$

It can be seen from the above calculation, for $0 \rightarrow 1$ errors the syndrome, S, cannot be less than or equal to S_{min} and for $1 \rightarrow 0$ errors S cannot be greater than or equal to S_{max}. Thus we have the following result

Theorem 3.2.4. *For $r \geq 5$ and S satisfying the following conditions, the type of error is given as*

1. $1 \leq S \leq S_{min} \Rightarrow$ *1 → 0 errors*

2. $S_{max} \leq S \leq 6.2^{r-4} - 1 \Rightarrow$ *0 → 1 errors*

When the syndrome is in the range $S_{min} + 1 \leq S \leq S_{max} - 1$, the type of error can be decided based on both the syndrome, S, and the received check.

We now state the main results of this section in the following two theorems. The proofs of these theorems are similar to those already proved in Section 3.2.1. The maximum constraint for the values of b_j is as already specified in Theorem 3.2.2.

Theorem 3.2.5. *Suppose $S = (S_{min} + l)$ mod 6.2^{r-4}, where $S_{min} + l \leq 6.2^{r-5}$ i.e., $S_{min} < S \leq 6.2^{r-5}$. Let $C' = (c'_{r-1}c'_{r-2} \ldots c'_0)$ be the received check bits and the decimal value of the least $(r-4)$ check bits be $\nu(C'_{r-4})$. For $l = 1$, the errors are of 0 → 1 type if $\nu(C'_{r-4}) \geq 2^{r-4} - 2$, otherwise it is of 1 → 0 type. For $l > 1$ and satisfying the condition $l > (2^{b_1} - 1) + (2^{b_2} - 1) + \ldots + (2^{b_j} - 1)$, where b_i's are chosen as the maximum values and $b_1 > b_2 > \ldots > b_j$, if $\nu(C'_{r-4}) \geq 2^{r-4} - (2^{b_1} + 2^{b_2} + \ldots + 2^{b_j} + 2)$ then the errors are of 0 → 1 type. Otherwise the errors are of 1→ 0 type.*

Example 5. *Let* $r = 10$. *By Theorem 3.2.4, if the syndrome is from 1 to 163, the errors are of 1* $\rightarrow$ *0 type and if it is from 221 to 383 they are of 0* $\rightarrow$ *1 type. Suppose the syndrome* $S = 184$. *Since in this case* $S_{min} = 163$, *the value of* l *is 21. Now* $21 = l > (2^4 - 1) + (2^2 - 1) + (2^1 - 1)$. *Let the received check be 0101001100. Since* $(001100)_2 = 12 < 2^6 - (2^4 + 2^2 + 2^1) + 2 = 40$ *the errors are of the 1* $\rightarrow$ *0 type. On the other hand, if the received check is 0101101110 then the errors must be of the 0* $\rightarrow$ *1 type. This is because* $(101110)_2 = 46 > 2^6 - (2^4 + 2^2 + 2^1 + 2) = 40$.

Theorem 3.2.6. *Suppose* $S = (S_{max} - l) \bmod 6.2^{r-4}$, *where* $S_{max} - l > 6.2^{r-5}$ *i.e.,* $6.2^{r-5} < S < S_{max}$. *Let* $C' = (c'_{r-1}c'_{r-2} \dots c'_0)$ *be the received check bits and the decimal value of the least* $(r-2)$ *check bits be* $\nu(C'_{r-4})$. *For* $l = 1$, *the errors are of 1* $\rightarrow$ *0 type if* $\nu(C'_{r-4}) < 2$, *otherwise it is of 0* $\rightarrow$ *1 type. For* $l > 1$ *and satisfying the condition* $l > (2^{b_1} - 1) + (2^{b_2} - 1) + \dots + (2^{b_j} - 1)$, *where* b_i*'s are chosen as the maximum values and* $b_1 > b_2 > \dots > b_j$, *if* $\nu(C'_{r-4}) < (2^{b_1} + 2^{b_2} + \dots + 2^{b_j} + 2)$ *then the errors are of 1* $\rightarrow$ *0 type. Otherwise the errors are of 0* $\rightarrow$ *1 type.*

CHAPTER 4

MIXED RADIX CODES

Let $K = k_n k_{n-1} \cdots k_1$ be an n-dimensional vector where k_i is the radix in dimension i and $k_i \geq 3$ for $1 \leq i \leq n$. Method 4.1 produces a cyclic Gray code if all k_i's are odd (or all are even). Method 2 gives a cyclic Gray code if at least one of the k_i's is even and a non-cyclic Gray code if all k_i's are odd.

Let $R = (r_n r_{n-1} \cdots r_1)$ be an n digit vector, where $0 \leq r_i \leq k_i - 1$ for $i = 1, 2, \cdots, n$. R is said to be in *mixed-radix notation*, and the corresponding integer value of R is given by

$$\begin{aligned} I(R) &= r_1 + r_2 k_1 + r_3 k_1 k_2 + \cdots + r_n k_1 k_2 \cdots k_{n-1} \\ &= r_1 + \sum_{i=2}^{n} \left(r_i \prod_{j=1}^{i-1} k_j \right) \end{aligned}$$

For example, if $K = 854$, then $I(442) = 2 + 4 \times 4 + 4 \times 5 \times 4 = 98$

4.0.3 Method 1

Assume that k_i is odd for $1 \leq i \leq n$, and that the dimensions are ordered such that $k_n \geq k_{n-1} \geq \cdots \geq k_1$. Also define

$$\overline{r_i} = \begin{cases} r_i, & \text{if } r_{i+1} \text{ is odd} \\ k_i - 1 - r_i, & \text{otherwise} \end{cases}$$

Arrange the n-dimensional vectors in a lexicographic order, i.e., $X = (x_n, x_{n-1}, \cdots, x_1) < Y = (y_n, y_{n-1}, \cdots, y_1)$ if $I(X) < I(Y)$. Then, f_1, which maps the vectors in mixed radix representation arranged in a lexicographic order to a cyclic Gray code, is defined as follows.

$$f_1(r_n, r_{n-1}, \cdots, r_1) = (g_n, g_{n-1}, \cdots, g_1)$$

where

$$\begin{aligned} g_n &= r_n, \text{and} \\ \text{for } 1 \leq i \leq n-1, \quad g_i &= \begin{cases} (r_i - r_{i+1}) \bmod k_i, & \text{if } r_{i+1} < k_i \\ \overline{r_i}, & \text{otherwise.} \end{cases} \end{aligned} \tag{4.1}$$

The inverse function for f_1 is

$$f_1^{-1}(g_n, g_{n-1}, \ldots, g_1) = (r_n, r_{n-1}, \ldots, r_1)$$

where

$$\begin{aligned} r_n &= g_n, \text{ and} \\ \text{for } n \geq i > 1, \quad r_{i-1} &= \begin{cases} (r_i + g_{i-1}) \bmod k_{i-1} & \text{if } r_i < k_{i-1} \\ \widetilde{r_{i-1}} & \text{otherwise.} \end{cases} \end{aligned} \tag{4.2}$$

In the above equation, $\widetilde{r_{i-1}}$ is defined as

$$\widetilde{r_{i-1}} = \begin{cases} k_{i-1} - 1 - g_{i-1} & \text{if } r_{i-1} \text{ is even} \\ g_{i-1} & \text{otherwise.} \end{cases}$$

Note

When all k_i's, $i = 1, 2, \cdots, n$, are even, a similar Gray code can be described. Again assume $k_n \geq k_{n-1} \geq \cdots \geq k_1$.

Define

$$\overline{r_i} = \begin{cases} r_i, & \text{if } r_{i+1} \text{ is even} \\ k_i - 1 - r_i, & \text{otherwise.} \end{cases}$$

Then

$$f_2(r_n, r_{n-1}, \cdots, r_1) = (g_n, g_{n-1}, \cdots, g_1)$$

where

$$\begin{aligned} g_n &= r_n, \text{and} \\ \text{for } 1 \le i \le n-1, \quad g_i &= \begin{cases} r_i - r_{i+1}, & \text{if } r_{i+1} < k_i \\ \overline{r_i}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.3}$$

The inverse function for f_2 can be defined as follows

$$f_2^{-1}(g_n, g_{n-1}, \ldots, g_1) = (r_n, r_n, \ldots, r_1) \tag{4.4}$$

where

$$\begin{aligned} r_n &= g_n, \text{ and} \\ \text{for } n \ge i > 1, \quad r_{i-1} &= \begin{cases} (r_i + g_{i-1}) \bmod k_{i-1} & \text{if } r_i < k_{i-1} \\ \widetilde{r_{i-1}} & \text{otherwise} \end{cases} \end{aligned} \tag{4.5}$$

In the above equation, $\widetilde{r_{i-1}}$ is defined as

$$\widetilde{r_{i-1}} = \begin{cases} k_{i-1} - 1 - g_{i-1} & \text{if } r_{i-1} \text{ is odd} \\ g_{i-1} & \text{otherwise} \end{cases}$$

Theorem 4.0.7. *Method 4.1 proposed above produces a Lee distance Gray code, when $k_i \ge 3$, for $i = 1, 2, \cdots, n$, and k_i's odd (or k_i's even).*

Proof. The proof is given when all k_i's, $i = n, n-1, \cdots, 1$ are odd. A similar proof can be obtained when they are all even.

- **Case 1:**

$$\begin{aligned} f_1(000\cdots0) &= 0000\cdots0 \\ f_1(k_n - 1, k_{n-1} - 1, \cdots, k_1 - 1) &= (k_n - 1, \overline{k_{n-1} - 1}, \overline{k_{n-2} - 1}, \cdots, \overline{k_1 - 1}) \end{aligned}$$

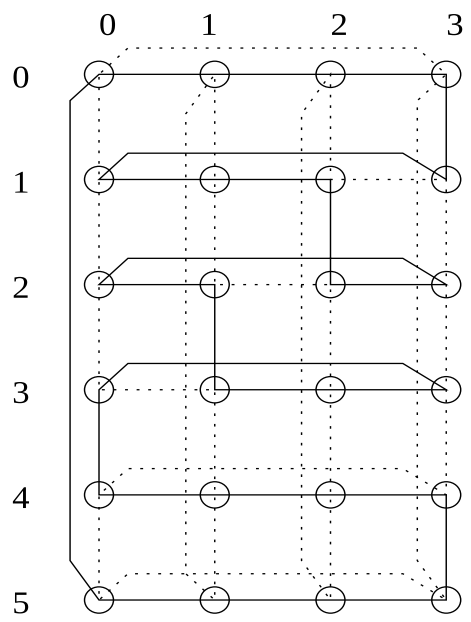


FIGURE 4.1: A Hamiltonian cycle in $C_6 \times C_4$ using Method 1

when $k_{i+1} > k_i$ for $i = 1, 2, \ldots, n-1$. Now for $i = n-1, n-2, \cdots, 1$, $\overline{k_i - 1} = 0$ because $(k_{i+1} - 1)$'s are all even. Further, if some $k_{i+1} = k_i$ it can be shown that $f_1(k_n - 1, k_{n-1} - 1, \cdots, k_1 - 1) = (k_n - 1, 0, 0, 0, \cdots, 0)$. Thus, the first and the last words are at a distance of 1.

- **Case 2:** Let X and Y be two consecutive numbers in the mixed radix representation and let m be the first index from left in which X and Y differ; i.e.,

$$\begin{aligned} X &= \{x_n x_{n-1} \cdots x_{m+1}\}^* \, x_m \, \{(k_{m-1} - 1)(k_{m-2} - 1) \cdots (k_1 - 1)\}^* \\ Y &= \{x_n x_{n-1} \cdots x_{m+1}\}^* \, (x_m + 1) \, \{00 \cdots 0\}^* \end{aligned}$$

where the segment marked by a $*$ may or may not exist depending on the value of m. Let $f_1(X) = a_n a_{n-1} \cdots a_1$ and $f_1(Y) = b_n b_{n-1} \cdots b_1$. $f_1(X)$ and $f_1(Y)$

are considered in three segments: (a) between dimensions n and $m+1$, (b) dimension m, and (c) between dimensions $m-1$ and 1. It is shown below that $a_i = b_i$ for $i \neq m$ and that $D_L(a_m, b_m) = 1$. This shows $D_L(f_1(X), f_1(Y)) = 1$.

(a) $[n \geq i \geq m+1]$: For this range, $x_i = y_i$ and so $a_i = b_i$.

(b) $[i = m]$:

case i $[x_{m+1} < k_m]$: In this case, $a_m = x_m - x_{m+1}$ and $b_m = x_m + 1 - x_{m+1}$, thus $D_L(a_m, b_m) = 1$.

case ii $[x_{m+1} \geq k_m]$: Here $a_m = \overline{x_m}$ and $b_m = \overline{x_m + 1}$. Further, if x_{m+1} is even,

$$\begin{aligned}
\overline{x_m} &= k_m - 1 - x_m \text{ and} \\
\overline{x_m + 1} &= k_m - 1 - (x_m + 1) \\
&= \overline{x_m} - 1
\end{aligned}$$

On the other hand, if x_{m+1} is odd, then $\overline{x_m} = x_m$ and $\overline{x_m + 1} = x_m + 1$. In both cases, $D_L(a_m, b_m) = D_L(\overline{x_m}, \overline{x_m + 1}) = 1$.

(c) $[m-1 \geq i \geq 1]$: Since each $k_i - 1$ is even and $k_i - 1 \geq k_{i-1} - 1$, it can be verified that $a_{m-2} = a_{m-3} = \cdots = a_1 = 0$. Further $b_{m-2} = b_{m-3} = \cdots = b_1 = 0$. Now, it will be shown that $a_{m-1} = b_{m-1}$ by considering the following three cases.

case i $[x_m + 1 < k_{m-1}]$: Then $a_{m-1} = k_{m-1} - 1 - x_m$ and $b_{m-1} = 0 - (x_m + 1) = k_{m-1} - x_m - 1$. So $a_{m-1} = b_{m-1}$.

case ii $[x_m + 1 = k_{m-1}$ and so $x_m = k_{m-1} - 1]$: $a_{m-1} = k_{m-1} - 1 - x_m = k_{m-1} - 1 - k_{m-1} - 1 = 0$; $b_{m-1} = 0$ since k_{m-1} is odd. Thus, in this case also $a_{m-1} = b_{m-1}$.

case iii $[x_m + 1 > k_{m-1}]$: If $x_m + 1$ is even (and so x_m is odd) then $a_{m-1} = k_{m-1} - 1$ and $b_{m-1} = \overline{0} = k_{m-1} - 1$; On the other hand, if x_{m+1} is odd (and so x_m is even) then $a_{m-1} = \overline{k_{m-1} - 1} = 0$ and $b_{m-1} = 0$. Thus in both cases $a_{m-1} = b_{m-1}$.

Further, it is easy to verify that f_1 is a one-to-one function. Therefore, f_1 produces a cyclic Gray code. □

As we have mentioned earlier, a Gray code over $k_n \times k_{n-1} \times \cdots \times k_1$ corresponds to an appropriate Hamiltonian cycle in the torus $T_{k_n} \times T_{k_{n-1}} \times \cdots \times T_{k_1}$. Figure 4.1 and Figure 4.2 show Hamiltonian cycles generated using Method 4.1 in $C_6 \times C_4$ and $C_9 \times C_7 \times C_3$, respectively.

4.0.4 Method 2

This method was first described in [14]. For completeness we describe the code's mapping function here. An efficient algorithm to generate this code is described in Chapter 5. Assume that at least one of the k_i's is even. Without loss of generality, assume that the dimensions are ordered so that if k_i is even and k_j is odd, then $i > j$. Let l be the index of the lowest even dimension. That is, the dimensions are ordered as follows.

$$\overbrace{k_n \cdots k_l}^{even} \; \overbrace{k_{l-1} \cdots k_1}^{odd}$$

Now, letting $\overline{r_i} = k_i - 1 - r_i$, and $r'_i = \sum_{j=i+1}^{l} r_j$, f_3 is defined as follows.

$$f_3(r_n, r_{n-1}, \cdots, r_1) = (g_n, g_{n-1}, \cdots, g_1),$$

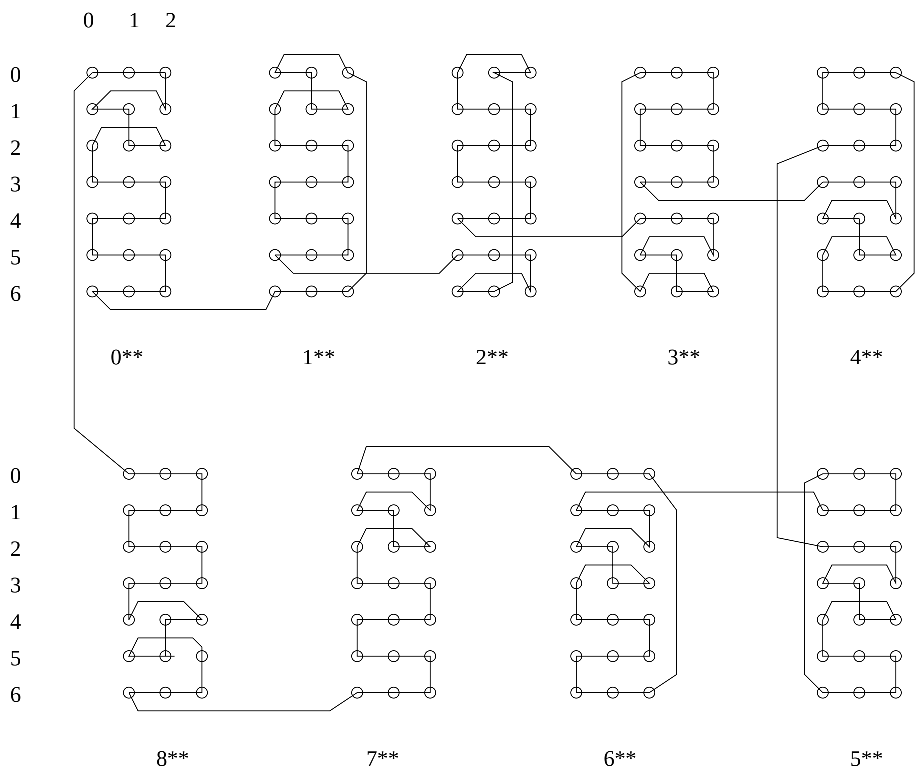


FIGURE 4.2: A Hamiltonian cycle in $C_9 \times C_7 \times C_3$ using Method 2

where

$$\begin{aligned} g_n &= r_n, \text{and} \\ g_i &= \begin{cases} r_i, & \text{if } r_{i+1} \text{ is even} \\ \overline{r_i}, & \text{otherwise} \end{cases} \quad \text{for } (n-1) \geq i \geq l \\ g_i &= \begin{cases} r_i, & \text{if } r_i' \text{ is even} \\ \overline{r_i}, & \text{otherwise} \end{cases} \quad \text{for } (l-1) \geq i \geq 1 \end{aligned} \tag{4.6}$$

The inverse function for f_3 can be defined as follows.

$$f_3^{-1}(g_n, g_{n-1}, \ldots, g_1) = (r_n, r_{n-1}, \ldots, r_1) \tag{4.7}$$

where

$$\begin{aligned} r_n &= g_n \\ \text{for } n-1 \geq i \geq 1;\ r_i &= \begin{cases} g_i & \text{if } \left(\sum_{j=i+1}^{n-1} g_j\right) \text{ is even} \\ k_i - 1 - g_i & \text{otherwise} \end{cases} \end{aligned}$$

4.1 Edge Disjoint Hamiltonian Cycles in Multi-dimensional Tori

In this section we show the construction of edge disjoint Hamiltonian cycles in toroidal networks when $n = 2$ and $n = 2^r$ for some $r \geq 0$. We use the Gray codes constructed using f_1 (or f_2) as the base cycle. The *torus* network is a popular parallel processor interconnection network topology. Some efficient communication algorithms can be designed based on edge disjoint Hamiltonian cycles (for example, all-to-all broadcast). Firstly, we introduce some concepts in graph theory to understand the constructions in this section.

4.1.1 Cross Product of Graphs

Given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, define the *cross product* of G_1 and G_2, denoted by $G_1 \times G_2$, as the graph $G = (V, E)$, where

$$\begin{aligned} V &= \{(u, v) | u \in V_1, v \in V_2\}, \text{, and} \\ E &= \{((u_1, v_1), (u_2, v_2)) | (u_1, u_2 \in E_1 \text{ and } v_1 = v_2), \\ &\quad \text{or } (u_1 = u_2 \text{ and } (v_1, v_2) \in E_2)\} \end{aligned}$$

A cycle of length k is denoted by C_k, and each node in C_k is labeled with a radix k number, $0, 1, \ldots, k-1$. There is an edge between vertices u and v *iff* $D_L(u, v) = 1$.

The k-ary n-cube (C_k^n) and a n-dimensional torus ($T_{k_n,k_{n-1},\ldots,k_1}$) can alternatively be defined as the product of cycles as follows:

$$\begin{aligned} C_k^n &= \underbrace{C_k \times C_k \times \ldots \times C_k}_{n \text{ times}} = \times_{i=1}^{n} C_k \\ T_{k_n,k_{n-1},\ldots,k_1} &= C_{k_n} \times C_{k_{n-1}} \times \ldots \times C_{k_1} \end{aligned}$$

The above definition demonstrates a useful topological property of a C_k^n: a C_k^n can be recursively defined in terms of smaller k-ary cubes.

$$C_k^n = \begin{cases} C_k, & \text{if } n = 1 \\ C_k \times C_k^{n-1}, & \text{if } n > 1. \end{cases}$$

4.1.2 Edge Disjoint Hamiltonian Cycles when $n = 2$

Theorem 4.1.1. *In a 2-dimensional toroidal network, the edges not used by the Hamiltonian cycle corresponding to the Gray Code generated using f_1 (or f_2) forms another Hamiltonian cycle.*

Proof. The second Hamiltonian cycle is given by the following sequence of edges and these edges are not in Gray code generated by f_1 (or f_2).

Column Number	Nodes
0	$(0,0),(1,0),(2,0),\ldots,(k_1-1,0)$
1	$(k_1-1,1),(k_1,1),\ldots,(k_2-1,1),(0,1),(1,1),\ldots,(k_1-2,1)$
2	$(k_1-2,2),(k_1-1,2),\ldots,(k_2-1,2),(0,2),(1,2),$
2	$\ldots,(k_1-3,2)$
$\vdots$	$\vdots$
i	$(k_1-i,i),(k_1-i+1,i),\ldots,(k_2-1,i),(0,i),(1,i),$
i	$\ldots,(k_1-i-1,i)$
$\vdots$	$\vdots$
(k_1-2)	$(2,k_1-2),(3,k_1-2),\ldots,(k_2-1,k_1-2),$
(k_1-2)	$(0,k_1-2),(1,k_1-2)$
(k_1-1)	$(1,k_1-1),(2,k_1-1),\ldots,(k_1-1,k_1-1)$
(k_1-1) and 0	$(k_1,k_1-1),(k_1,0),(k_1+1,0),(k_1+1,k_1-1),\ldots,$
	$(k_2-1,0),(k_2-1,k_1-1),(0,k_1-1),(0,0)$

In the above table each consecutive pair of nodes represents an edge. Each row in the table represents the order in which nodes in the respective column are visited. The last node of a row i will be adjacent to the first node in row $i+1$. In the last row the the edges alternate between dimension 0 and k_1-1. Also, the last node of the cycle is $(0,0)$. $\square$

A mapping function, f', which maps vectors in a lexicographic order, (x_2,x_1), to their corresponding codewords, (g_2,g_1) can be defined for the second cycle described above, as follows.

1. When $0 \leq x_2.k_1 + x_1 \leq k_1 - 1$:

$$f'(x_2,x_1) = (g_2,g_1)$$

where

$$\begin{aligned} g_2 &= x_1 \\ g_1 &= x_2 = 0 \end{aligned}$$

2. When $k_1 \le x_2.k_1 + x_1 < (k_1.k_2 + 2(k_2 - k_1))$:

$$\text{Let } p = (x_2.k_1 + x_1 + k_1 - 1)$$

Let (y_2, y_1) be such that, $p = y_2.k_2 + y_1$ where $0 \le y_1 \le k_2 - 1$. Then,

$$\begin{cases} g_1 = & y_2 \\ g_2 = & (y_1 - y_2 + k_1) \ \mathbf{mod} \ k_2 \end{cases}$$

3. When $(k_1.k_2 + 2(k_2 - k_1)) \le x_2.k_1 + x_1 \le (k_2.k_1 - 1))$:

Let $b = (k_1.k_2 + 2(k_2 - k_1))$ mod 4, then:

$$\begin{cases} g_1 = 0 & \text{if } (x_2.k_1 + x_1) \ \mathbf{mod} \ 4 = b \text{ or } b + 1 \\ g_1 = k_1 - 1 & \text{if } (x_2.k_1 + x_1) \ \mathbf{mod} \ 4 = b + 2 \text{ or } b + 3 \end{cases}$$

and

$$g_2 = \left(\left\lceil \frac{d}{2} \right\rceil + k_1 \right) \ \mathbf{mod} \ k_2$$

where,

$$d = (x_2.k_1 + x_1 - b)$$

4.1.3 Edge Disjoint Hamiltonian Cycles when $n = 2^r$

First, we explain how to obtain 4 edge disjoint Hamiltonian cycles from a 4 dimensional torus. Let

$$T_{k_4 \times k_3 \times k_2 \times k_1} = C_{k_4} \times C_{k_3} \times C_{k_2} \times C_{k_1} \tag{4.8}$$

Here k_i's are all odd (or all even).

$C_{k_4} \times C_{k_3}$ is a two dimensional torus and so it can be decomposed into two edge disjoint Hamiltonian cycles H_1 and H_2 of length $k_4 \times k_3$ as described in Theorem 4.1.1. These two Hamiltonian cycles give two Gray codes and so if any two vectors are adjacent in one Gray code they are not adjacent in the other. We can write

$$C_{k_4} \times C_{k_3} = H^1_{k_4 \times k_3} + H^2_{k_4 \times k_3} \tag{4.9}$$

Similarly we can write

$$C_{k_2} \times C_{k_1} = H^1_{k_2 \times k_1} + H^2_{k_2 \times k_1} \tag{4.10}$$

where $H^1_{k_2 \times k_1}$ and $H^1_{k_2 \times k_1}$ are two edge disjoint cycles of length $k_2 \times k_1$ obtained from $C_{k_2} \times C_{k_1}$ using Theorem 4.1.1. Consider the two graphs $G_1 = H^1_{k_4 \times k_3} \times H^1_{k_2 \times k_1}$ and $G_2 = H^2_{k_4 \times k_3} \times H^2_{k_2 \times k_1}$. Both G_1 and G_2 have the same number of nodes and the node labels are the same. The row labels of G_1 (and G_2) are the Gray codes corresponding to $H^1_{k_4 \times k_3}$ (and $H^2_{k_4 \times k_3}$) respectively. Similarly the column labels of G_1 (and G_2) are from $H^1_{k_2 \times k_1}$ (and $H^2_{k_2 \times k_1}$) respectively. We will show that G_1 and G_2 are edge disjoint.

Suppose $(X = (x_4, x_3, x_2, x_1), Y = (y_4, y_3, y_2, y_1)$) is an edge in G_1. If X and Y differ in one of the least significant two digits then (X, Y) is a row edge. This row edge cannot be a row edge in G_2. This is so because the vectors (x_2, x_1) and (y_2, y_1) are adjacent in the Gray code corresponding to $H^1_{k_2 \times k_1}$ and hence they cannot be adjacent in the Gray code corresponding to $H^2_{k_2 \times k_1}$. Similarly if X and Y differ in one of the two most significant digits then this forms a column edge in G_1. Again, using an argument similar to that for a row edge it can be seen that if (x_4, x_3) and

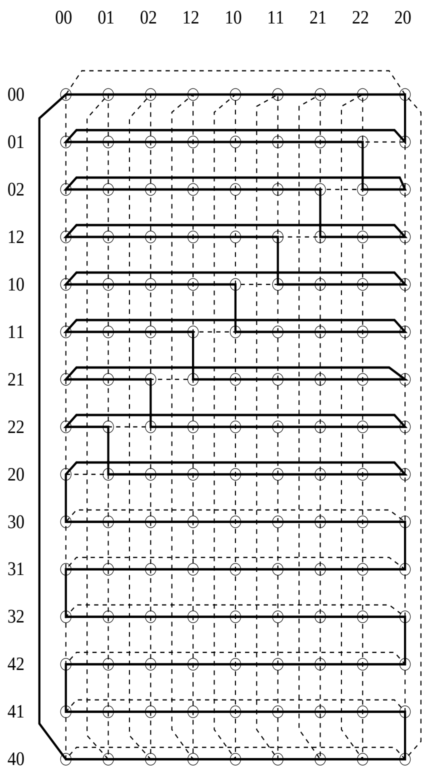
(y_4, y_3) are adjacent in G_1 they cannot be a column edge in G_2. Finally from each of $H^1_{k_4\times k_3} \times H^1_{k_2\times k_1}$ and $H^2_{k_4\times k_3} \times H^2_{k_2\times k_1}$ we can obtain two edge disjoint Hamiltonian cycles. These can be written in terms of graph decomposition as follows.

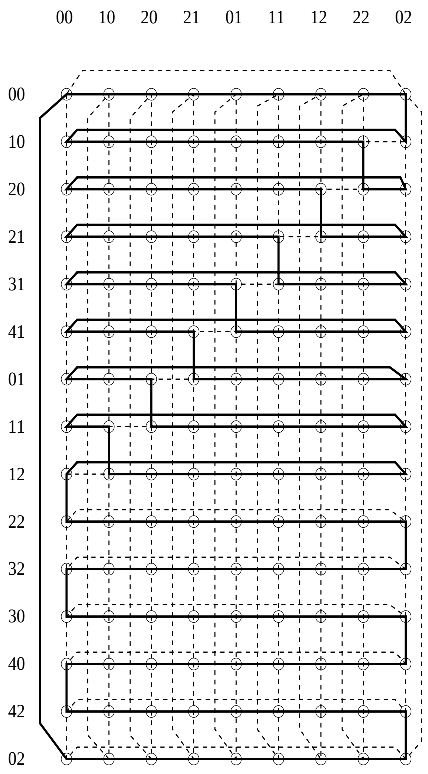
$$\begin{aligned}
T_{k_4\times k_3\times k_2\times k_1} &= C_{k_4} \times C_{k_3} \times C_{k_2} \times C_{k_1} && (4.11)\\
&= (C_{k_4} \times C_{k_3}) \times (C_{k_2} \times C_{k_1}) && (4.12)\\
&= (H^1_{k_4\times k_3} + H^2_{k_4\times k_3}) \times (H^1_{k_2\times k_1} + H^2_{k_2\times k_1}) && (4.13)\\
&= (H^1_{k_4\times k_3} \times H^1_{k_2\times k_1}) + (H^2_{k_4\times k_3} \times H^2_{k_2\times k_1}) && (4.14)\\
&= (H^1_{k_4\times k_3\times k_2\times k_1} + H^3_{k_4\times k_3\times k_2\times k_1}) && (4.15)\\
&+ (H^2_{k_4\times k_3\times k_2\times k_1} + H^4_{k_4\times k_3\times k_2\times k_1}) && (4.16)
\end{aligned}$$

Example 6. *Consider a $T_{5\times3\times3\times3}$ toroidal network. Four edge disjoint cycles, H^1, H^2, H^3 and H^4 can be generated by decomposing the graph into two 2-dimensional toroidal networks, ($T^1_{15\times9}$ and $T^2_{15\times9}$), of size 15×9. These two toroidal networks, $T^1_{15\times9}$ and $T^2_{15\times9}$, are edge disjoint. Now, functions f_1 and f' can be applied to these $T_{15\times9}$ graphs to obtain 2 edge disjoint cycles from each graph respectively. These cycles are shown in Figures 4.3(a) and 4.3(b) respectively. The graph in Figure 4.3(a) is obtained by the cross product $H^1_{5\times3} \times H^1_{3\times3}$. The graph in Figure 4.3(b) is obtained by the cross product $H^2_{5\times3} \times H^2_{3\times3}$ where H^1 and H^2 are the Gray codes obtained by using f_1 and f' in the 5×3 and 3×3 graphs.*

In this case, the n edge disjoint cycles can be recursively constructed as described in the following theorem.

Theorem 4.1.2. *It is possible to construct n edge disjoint Hamiltonian cycles for an n-dimensional torus network $T_{k_n,k_{n-1},\ldots,k_1}$ where $n = 2^r$ and $k_i \geq 3$, for $1 \leq i \leq n$.*





(a) H^1(Solid Lines) and H^3 (Dotted Lines)

(b) H^2 (Solid Lines) and H^4(Dotted Lines)

FIGURE 4.3: Four Edge Disjoint Hamiltonian Cycles in a $5 \times 3 \times 3 \times 3$ Torus

Proof. The following proof does not depend on the parities of k_i's and can thus be applied to Gray codes constructed using either f_1 or f_2. The proof is by induction on n.

- **Base:** When $n = 2$, this theorem is reduced to Theorem 4.1.1

- **Induction Hypothesis:** Assume that there are n edge disjoint Hamiltonian cycles in $T_{k_n,k_{n-1},\ldots,k_1}$ say, $H^1_{k_n\times k_{n-1}\times\ldots\times k_1}$, $H^2_{k_n\times k_{n-1}\times\ldots\times k_1}, \ldots, H^n_{k_n\times k_{n-1}\times\ldots\times k_1}$ for $n = 2^r$, i.e., $T_{k_n,k_{n-1},\ldots,k_1} = H^1_{k_n\times k_{n-1}\times\ldots\times k_1} + H^2_{k_n\times k_{n-1}\times\ldots\times k_1} + \ldots + H^n_{k_n\times k_{n-1}\times\ldots\times k_1}$.

Here, $G_1 + G_2$ indicates the union of two edge disjoint graphs, G_1 and G_2, where the graphs have the same set of nodes.

- **Induction Step:** Now, consider the case for $n' = 2n = 2^{r+1}$. A $T_{k_{2n},k_{2n-1},\ldots,k_1}$ can be decomposed as

$$\begin{aligned}
T_{k_{2n},k_{2n-1},\ldots,k_1} &= C_{k_{2n}} \times C_{k_{2n-1}}, \ldots, C_{k_1} \\
&= (C_{k_{2n}} \times C_{k_{2n-1}} \times \ldots \times C_{k_{n+1}}) \times (C_{k_n} \times C_{k_{n-1}} \times \ldots \times C_{k_1}) \\
&= (H^1_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} + H^2_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} + \ldots \\
&+ H^n_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}}) \times \\
&\quad (H^1_{k_n\times k_{n-1}\times\ldots\times k_1} + H^2_{k_n\times k_{n-1}\times\ldots\times k_1} + \ldots + H^n_{k_n\times k_{n-1}\times\ldots\times k_1}) \\
&= (H^1_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} \times H^1_{k_n\times k_{n-1}\times\ldots\times k_1}) \\
&+ (H^2_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} \times H^2_{k_n\times k_{n-1}\times\ldots\times k_1}) + \\
&\quad \ldots + (H^n_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} \times H^n_{k_n\times k_{n-1}\times\ldots\times k_1}) \\
&= (H^1_{k_{2n}\times k_{2n-1}\times\ldots\times k_1} + H^{n+1}_{k_{2n}\times k_{2n-1}\times\ldots\times k_1}) \\
&+ (H^2_{k_{2n}\times k_{2n-1}\times\ldots\times k_1} + H^{n+2}_{k_{2n}\times k_{2n-1}\times\ldots\times k_1}) + \\
&\quad \ldots + (H^n_{k_{2n}\times k_{2n-1}\times\ldots\times k_1} + H^{2n}_{k_{2n}\times k_{2n-1}\times\ldots\times k_1})
\end{aligned}$$

Now, $(H^i_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} \times H^i_{k_n\times k_{n-1}\times\ldots\times k_1})$ is a two dimensional torus of size $(k_{2n} \times k_{2n-1} \times \ldots \times k_{n+1}) \times (k_n \times k_{n-1} \times \ldots \times k_1)$ and this is edge disjoint from $(H^j_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} \times H^j_{k_n\times k_{n-1}\times\ldots\times k_1})$ for $i \neq j$. From each $(H^i_{k_{2n}\times k_{2n-1}\times\ldots\times k_{n+1}} \times H^i_{k_n\times k_{n-1}\times\ldots\times k_1})$, two edge disjoint Hamiltonian cycles can be constructed using Theorem 4.1.1. Thus, $2n$ disjoint Hamiltonian cycles can be constructed for $T_{k_{2n},k_{2n-1},\ldots,k_1}$.

$\square$

CHAPTER 5

ALGORITHMS FOR GENERATING GRAY CODES

In this chapter we derive efficient algorithms for generating the Gray codes defined by f_1, f_2 and f_3. All these algorithms are efficient in the the sense that the time taken to transform one codeword to the next is bound by a constant. The following algorithms extend the idea of efficiently generating the *transition sequence* of the respective code and then adding the logic to correctly derive the value of the digit that changes. As it turns out the transition sequences of all the three codes are of the same structure, though the logic required to derive the value of the digit that changes is different.

5.0.4 Transition Sequence

Let a Gray Code listing of all the $k_n.k_{n-1}.\dots.k_1$ vectors over $Z_{k_n}.Z_{k_{n-1}}.\dots.Z_{k_1}$ be represented as $C = (c_1, c_2, \dots, c_{k_n.k_{n-1}.\dots.k_1})$. The transition sequence of C is defined as $\Gamma_{k_{n-1},k_{n-2},\dots,k_1} = (\gamma_1, \gamma_2, \dots, \gamma_{k_n.k_{n-1}.\dots.k_1-1})$, where γ_i is the digit that changes in C between g_i and g_{i+1}, for $1 \leq i < k_n.k_{n-1}.\dots.k_1$.

Example 7. *Table 5.1 shows the Gray codes constructed using f_1, f_2 and f_3 in $Z_{5\times3}$, $Z_{6\times4}$ and $Z_{6\times3}$ respectively and their corresponding transition sequences. The transition sequences are of the same structure.*

For two digits, the transition sequences of the Gray codes defined using f_1, f_2 and f_3 are of the form.

$$\Gamma_{k_2,k_1} = ((1)^{k_1-1}2)^{k_2-1}(1)^{k_1-1}$$

i	$C_{5\times 3}(f_1)$	$\Gamma_{5\times 3}$
1	00	1
2	01	1
3	02	2
4	12	1
5	10	1
6	11	2
7	21	1
8	22	1
9	20	2
10	30	1
11	31	1
12	32	2
13	42	1
14	41	1
15	40	

i	$C_{6\times 4}(f_2)$	$\Gamma_{6\times 4}$
1	00	1
2	01	1
3	02	1
4	03	2
5	13	1
6	10	1
7	11	1
8	12	2
9	22	1
10	23	1
11	20	1
12	21	2
13	31	1
14	32	1
15	33	1
16	30	2
17	40	1
18	41	1
19	42	1
20	43	2
21	53	1
22	52	1
23	51	1
24	50	

i	$C_{6\times 3}(f_3)$	$\Gamma_{6\times 3}$
1	00	1
2	01	1
3	02	2
4	12	1
5	11	1
6	10	2
7	20	1
8	21	1
9	22	2
10	32	1
11	31	1
12	30	2
13	40	1
14	41	1
15	42	2
16	52	1
17	51	1
18	50	

TABLE 5.1: Transition sequences of Gray codes defined using f_1, f_2 and f_3

In the above equation, the exponential term is the number of times the pattern occurs repetitively. Let $\Gamma_{k_n,k_{n-1},\ldots,k_1}$ be the generalized transition sequence of a mixed radix Gray code constructed using any of the functions defined in this paper. This sequence can be recursively defined as follows.

$$\Gamma_{k_n,k_{n-1},\ldots,k_1} = ((\Gamma_{k_{n-1},k_{n-2},\ldots,k_1})\ n)^{k_n-1}\ \Gamma_{k_{n-1},k_{n-2},\ldots,k_1} \tag{5.1}$$

$$\Gamma_{k_1} = (1)^{k_1-1} \tag{5.2}$$

We are interested in efficiently generating the codewords of the Gray codes successively from one vector to the next. It is sufficient to be able to generate $\Gamma_{k_n,k_{n-1},\ldots,k_1}$ efficiently and this can be done using a stack as follows: the stack initially contains $(k_1 - 1)$ 1's, $(k_2 - 1)$ 2's, ..., $(k_n - 1)$ n's with the 1's on top. The algorithm pops off the top element i, and then, when $i > 1$, $(k_{i-1} - 1)$ copies of $(i-1)$'s, $(k_{i-2} - 1)$ copies of $(i - 2)$'s, ..., $(k_1 - 1)$ copies of 1's are pushed onto the stack. The sequence of the values popped in each iteration forms the transition sequence of the code. Notice that the stack operates in a highly restricted manner since when $j > 1$ is placed in the stack we know a priori that $j - 1, \ldots, 1$ elements will be placed above it.

To efficiently implement this functionality we use an array $(\tau_n, \tau_{n-1}, \ldots, \tau_0)$ as the stack as follows [7]: τ_0 points to the top of the stack and for $j \geq 1$, the value of τ_j is the element below j on the stack if j is on the stack. If j is not on the stack, then the value of τ_j cannot affect the computation and its value can be reset to $j + 1$, since we know that when $j + 1$ is put on the stack, the element above it must be j. Moreover, since the elements $i - 1, i - 2, \ldots, 1$ will be put on the stack when i is removed, we need only to set $\tau_{i-1} \leftarrow \tau_i$, assuming that τ_j was reset when j was removed for all j, $i - 2 \leq j \leq 1$. Finally, if $i \neq 1$, the elements $i - 1, \ldots, 1$ are put on the stack using the assignment $\tau_0 \leftarrow 1$.

Note that the array, $\tau = (\tau_n, \tau_{n-1}, \ldots, \tau_1)$, has only one copy of each digit.

We avoid storing multiple copies of digit indices by using a counter array, $c = (c_n, c_{n-1}, \ldots, c_1)$. Each element c_i in the counter array keeps track of the number of times i has been generated. Once a bit index j has been generated $k_j - 1$ times, c_j is reset to 0.

Consider the following code fragment. This is precisely the part of code that generates the transition sequence and is same for all codes shown in this paper. To prove that it generates the correct transition sequence, we need to show that the sequence of values of the variable i is $\Gamma_{k_n,k_{n-1},\ldots,k_1}$.

```
while $\tau_0 < q+1$ do
    $i \leftarrow \tau_0$
    output $i$
    $\tau_0 \leftarrow 1$
    if $(c_i \geq (k_i - 2))$
        then
            $\tau_{i-1} \leftarrow \tau_i$
            $\tau_i \leftarrow i+1$
            $c_i \leftarrow 0$
        else
            $c_i \leftarrow c_i + 1$
                                                        (5.3)
```

Consider the above **while** loop when started with $\tau_j = \alpha_j$, $0 \leq j \leq q$, where $\alpha_j = j+1$ for $0 \leq j < q$, $\alpha_q \geq q+1$, and $c_j = 0$, $0 \leq j \leq q$. The counter array, c_i, for $1 \leq i \leq n$, maintains a count of the number of times the corresponding digit has occurred in the sequence of transition indices generated so far. The radix array, k_i, $0 \leq i \leq n$, represents the radices of the corresponding digits.

Thus, we are done if we demonstrate that the above loop produces the sequence $\Gamma_{k_q,k_{q-1},\ldots,k_1}$ stopping with $\tau_j = j+1$ for $1 \leq j \leq q$, $\tau_0 = \alpha_q \geq q+1$ and $c_j = 0$, $0 \leq j \leq q$. When $q = 1$, the value of $c_1 < k_1 - 2$, τ_0 is reset to 1 (**else** part of the **if**), and the stack state is not changed. When $c_1 = k_1 - 2$, $(k_1 - 1)$ 1's have been inserted and the **then** part of **if** part is executed with $i = 1$ as follows.

$$\begin{aligned} &\tau_0 \leftarrow (\tau_1 = 2) \\ &\tau_1 \leftarrow 1 + 1 \\ &c_1 \leftarrow 0 \end{aligned} \tag{5.4}$$

When the above statements are executed the algorithm terminates leaving $\tau_1 = 2$, $\tau_0 = \alpha_q \geq 2$ and $c_1 = 0$, obviously satisfying the above conditions and Γ_{k_1} has been generated.

Suppose the hypothesis is true for some $q = n - 1$. Consider what happens when $q = n$. Since the hypothesis for $q = n - 1$ is also satisfied, we know by induction that $\Gamma_{k_{n-1},k_{n-2},\ldots,k_1}$ is produced leaving $\tau_j = j + 1$ for $1 \leq j \leq n - 1$, $\tau_0 = \alpha_{n-1} = n$ and $c_j = 0$ for $0 \leq j \leq (n - 1)$. The next iteration produces n and since $c_n = 0 \leq k_n - 2$ the else part is executed leaving the stack at $\tau_j = j + 1$ for $0 \leq j \leq n - 1$ and c_n is incremented to 1. Then, again by induction, we know that $\Gamma_{k_{n-1},k_{n-2},\ldots,k_1}$ is produced. The next iteration again produces n and this process repeats until n has been produced $k_n - 1$ times. Now $c_n = k_n - 2$, the **then** part of the **if** is executed leaving $\tau_j = j + 1$ for $0 \leq j < n - 1$ and sets $\tau_{n-1} = \alpha_n = n + 1$ with $c_j = 0$ for $0 \leq j \leq n - 1$. Then by induction $\Gamma_{k_{n-1},k_{n-2},\ldots,k_1}$ is produced for the last time. After this, $\tau_0 = \alpha_n \geq n + 1$ leaving $\tau_j = j + 1$ for $1 \leq j \leq n - 1$ and $c_j = 0$ for $0 \leq j \leq n$ and hence

$$\Gamma_{k_n,k_{n-1},\ldots,k_1} = ((\Gamma_{k_{n-1},k_{n-2},\ldots,k_1})n)^{k_n - 1}\,\Gamma_{k_{n-1},k_{n-2},\ldots,k_1}$$

has been produced.

We use an array $G = (g_n, g_{n-1}, \ldots, g_1)$ to store the current vector. In each iteration the value of g_i, where i is the digit to be updated in that iteration, has to be either incremented or decremented as defined in Equation 4.1. To implement this functionality, we use a parity array $(p_n, p_{n-1}, \ldots, p_1)$ to decide whether a digit has to be incremented or decremented and another array $(\pi_n, \pi_{n-1}, \ldots, \pi_1)$ to decide when the corresponding update function ($(r_i - r_{i+1}) \bmod k_i$ or $\overline{r_i}$) has to be applied. Next we show the algorithms for generating the Gray codes.

Algorithm 1. *Algorithm for generating the Gray code defined by* f_1 *:*

```
for $j = 0$ to $n+1$ do
    $g_j \leftarrow 0$
    $\tau_j \leftarrow j + 1$
    $c_i \leftarrow 0$
    $p_i \leftarrow 0$
    $\pi_i \leftarrow 0$
$i \leftarrow 0$
while $i < n+1$ do
    output $g_n, \ldots, g_1$
    $i \leftarrow \tau_0$
    if $(\pi_i < k_i)$
        then {$g_i \leftarrow (g_i + 1)$ mod $k_i$
        else if $(p_i = 1)$
            then {$g_i \leftarrow (g_i + 1)$
            else {$g_i \leftarrow (g_i - 1)$
    $\tau_0 \leftarrow 1$
    if $(c_i \geq (k_i - 2))$
        then
            $\tau_{i-1} \leftarrow \tau_i$
            $\tau_i \leftarrow i + 1$
            $c_i \leftarrow 0$
            $p_i \leftarrow \overline{p_i}$
            $\pi_i \leftarrow \pi_i + 1$
            if $((\pi_i = k_{i+1})$ and $(i < n))$ then
                $\pi_i \leftarrow 0$
                $p_i \leftarrow \overline{p_i}$
        else
            $c_i \leftarrow c_i + 1$
```

Proof of Correctness

To prove that the algorithm is correct we need to show that it generates the correct transition sequence of the Gray code in variable i and the logic for setting the value of g_i, for $n \geq i \geq 1$, sets the correct values so as to generate the code. As already proved, i contains the correct value of the digit that has to change in each iteration.

It has to be shown that Algorithm 1 also sets the values of g_i correctly in each iteration. Note that k_i is *odd*, for $n \leq i \leq 1$, and the code chooses between reflection and rotation of digits to obtain g_i depending on the value of r_{i+1} and k_i. The array, $G = (g_n, g_{n-1}, \ldots, g_1)$, holds the current codeword. The array, $\Pi = (\pi_n, \pi_{n-1}, \ldots, \pi_0)$ holds the lexicographical value of the digit $i + 1$, for $n < i \leq 1$, for calculating the value of g_i. This value is used as defined in Equation 4.1 to decide when the respective operation has to be applied to the i^{th} digit. Also, a parity array $P = (p_n, p_{n-1}, \ldots, p_1)$, is defined to handle the reflective case. When $p_i = 1$, g_i is in a sequence of values in ascending order; otherwise it is in a sequence of values in descending order.

For $n - 1 \geq i \geq 1$, consider the **if** statements in Equations 5.5 and 5.6. These occur in the code in the order they are listed and only the relevant portions of the code are shown.

$$\textbf{if } (\pi_i < k_i) \begin{cases} \textbf{then} \, \{g_i \leftarrow (g_i + 1) \textbf{ mod } k_i \\ \textbf{else if } (p_i = 1) \begin{cases} \textbf{then} \, \{g_i \leftarrow (g_i + 1) \\ \textbf{else } \, \{g_i \leftarrow (g_i - 1) \end{cases} \end{cases} \tag{5.5}$$

The above **if** statement, in Equation 5.5, uses the lexicographical value, r_{i+1} stored as π_i to choose the type of operation to be applied to g_i. If $\pi_i < k_i$ then, by equation 4.1, $g_i = (r_i - r_{i+1}) \bmod k_i$. This effectively translates to an addition 1 to $g_i \bmod k_i$. On the other hand when $\pi_i \geq k_i$, we use the parity bit to decide whether g_i occurs in lexicographic order or is reflected. This proves that the correct function is applied to g_i provided that π_i has the correct value for i in each iteration.

$$\textbf{if } (c_i \geq (k_i - 2)) \left\{ \begin{array}{l} \textbf{then} \left\{ \begin{array}{l} p_i \leftarrow \overline{p_i} \\ \pi_i \leftarrow \pi_i + 1 \\ \textbf{if } ((\pi_i = k_{i+1}) \textbf{ and } (i < n)) \textbf{ then} \left\{ \begin{array}{l} \pi_i \leftarrow 0 \\ p_i \leftarrow \overline{p_i} \end{array} \right. \end{array} \right. \\ \\ \textbf{else} \left\{ c_i \leftarrow c_i + 1 \right. \end{array} \right. \tag{5.6}$$

The next part of the proof shows that π_i and p_i are correctly set by the second **if** statement. Let b_i, for $n > i > 1$, be a block of vectors such that $g_n, g_{n-1}, \ldots, g_{i+1}$ are the same. The number of blocks of b_i in a single block of b_{i+1} is equal to k_{i+1}. When ordered lexicographically, these blocks are labeled starting with 0 to $k_{i+1} - 1$. As can be seen from Equation 5.6, π_i is incremented each time $k_{i-1} \times k_{i-2}, \ldots, k_1$ vectors have been generated and is reset to 0 when $\pi = k_{i+1}$. Thus, π_i holds the correct index of the current block, b_i, in a single block of b_{i+1}.

Since all k_i's are odd, we know that $(k_{i+1} - k_i)$ is *even*. The first k_i blocks use rotation operation for g_i , this corresponds to $\pi_i < k_i$. The next $(k_{i+1} - k_i)$ blocks are reflected. This implies that for the first value of g_{i+2}, when the first block of reflected vectors is generated, $p_i = 1$. But for the next value of g_{i+2} it starts with $p_i = 0$ (as shown in Example 8). But we need a consistent value of p_i to know if g_i is to be incremented or decremented. Thus, we need to complement p_i one more time to ensure that this is so. This is implemented in the inner **if** in Equation 5.6. Thus the values of π_i and p_i are correctly updated. Thus, this combination of **if** statements set the values of g_i's correctly.

When $i = n$, the **else** part of the first **if** is always executed because $\pi_n = 0$. Since n appears $k_n - 1$ times in the transition sequence, the values of g_n are $(0, 1, \ldots, k_n - 1)$. It can thus be seen that g_n is also set correctly. This completes the proof.

Example 8. *When $k_i = 3$ and $k_{i+1} = 7$, consider the Gray code listing and the corresponding value of p_i between two consecutive b_{i+1} blocks. Columns 1 and 2 represent the blocks of vectors for consecutive values of g_{i+2}*

π_i	g_i	p_i	π_i	g_i	p_i
0	$f_R(\pi_i)$	0	0	$f_R(\pi_i)$	1
1	$f_R(\pi_i)$	1	1	$f_R(\pi_i)$	0
2	$f_R(\pi_i)$	0	2	$f_R(\pi_i)$	1
3	$f_{Re}(\pi_i)\downarrow$	1	3	$f_{Re}(\pi_i)\downarrow$	0
4	$f_{Re}(\pi_i)\uparrow$	0	4	$f_{Re}(\pi_i)\uparrow$	1
5	$f_{Re}(\pi_i)\downarrow$	1	5	$f_{Re}(\pi_i)\downarrow$	0
6	$f_{Re}(\pi_i)\uparrow$	0	6	$f_{Re}(\pi_i)\uparrow$	1
	$g_{i+2} = \nu$			$g_{i+2} = \nu \pm 1$	

(5.7)

where f_R refers to the rotation part of f_1 and f_{Re} refers to reflection. As can be seen from the above table, the relation between value p_i and whether g_i is in ascending order (denoted by $\downarrow$), is not consistent ($p_i = 1$ implies ascending order in the first column ($g_{i+2} = \nu$) whereas $p_i = 1$ implies descending order in the second ($g_{i+2} = \nu \pm 1$)). Also, rotation is applied when $\pi_i < k_i$ and reflection is applied otherwise.

Algorithm 2. *Algorithm for generating the Gray code defined by f_2 :*

```
for $j = 0$ to $n+1$ do
    $g_j \leftarrow 0$
    $\tau_j \leftarrow j+1$
    $c_i \leftarrow 0$
    $p_i \leftarrow 0$
    $\pi_i \leftarrow 0$

$i \leftarrow 0$
while $i < n+1$ do
    output $g_n, \ldots, g_1$
    $i \leftarrow \tau_0$
    if $(\pi_i < k_i)$
        then {$g_i \leftarrow (g_i + 1)$ mod $k_i$
        else if $(p_i = 0)$
            then {$g_i \leftarrow (g_i + 1)$
            else {$g_i \leftarrow (g_i - 1)$
    $\tau_0 \leftarrow 1$
    if $(c_i \geq (k_i - 2))$
        then
            $\tau_{i-1} \leftarrow \tau_i$
            $\tau_i \leftarrow i+1$
            $c_i \leftarrow 0$
            $p_i \leftarrow \overline{p_i}$
            $\pi_i \leftarrow \pi_i + 1$
            if $((\pi_i = k_{i+1})$ and $(i < n))$
                then $\pi_i \leftarrow 0$
        else
            $c_i \leftarrow c_i + 1$
```

Proof of Correctness

Since all k_i's are even the extra complementation operation for p_i is not required. Otherwise the proof of this algorithm is similar to the one shown above. Also, when

$p_i = 0$, g_i is in a sequence of digits in ascending order. Otherwise it is in a sequence of digits in descending order.

CHAPTER 6

BINARY GRAY CODE

6.1 Lee Distance Gray Codes

Let $R = (r_n r_{n-1} \cdots r_1)$ be an n digit vector, where $0 \leq r_i \leq k-1$ for all $i = 1, 2, \cdots, n$. For example, if $k = 4$, then $I(301) = 3 \times 4 \times 4 + 0 \times 4 + 1 = 49$. Arrange all the n digit vectors of radix k in lexicographic order, i.e., $X = (x_n, x_{n-1}, \cdots, x_1) < Y = (y_n, y_{n-1}, \cdots, y_1)$ iff $I(X) < I(Y)$. Then, f_l, which maps the vectors arranged in a lexicographic order to a cyclic Gray code, is defined as follows [12].

$$f_l(r_n, r_{n-1}, \cdots, r_1) = (g_n, g_{n-1}, \cdots, g_1)$$

where

$$\begin{aligned} g_n &= r_n, \text{and} \\ \text{for } 1 \leq i \leq n-1, \quad g_i &= (r_i - r_{i+1}) \bmod k \end{aligned} \tag{6.1}$$

The inverse function for f_l is

$$f_l^{-1}(g_n, g_{n-1}, \ldots, g_1) = (r_n, r_{n-1}, \ldots, r_1)$$

where

$$\begin{aligned} r_n &= g_n, \text{ and} \\ \text{for } n > i \geq 1, \quad r_i &= \left(\sum_{j=i}^{n} g_j \right) \bmod k \end{aligned} \tag{6.2}$$

The n-digit k-ary Lee distance Gray code, $\mathcal{L}_k(n)$, can also be recursively defined as given in [39]. The k-ary Gray code of 1 digit is given by

$$\mathcal{L}_k(1) = \mathcal{L}_k^0(1) = [0, 1, \ldots, k-1]^T \tag{6.3}$$

where T denotes transpose. Let

$$\mathcal{L}_k^i(1) = [k-i, k-i+1, \ldots, k-i-1]^T \tag{6.4}$$

be obtained by cyclically shifting the elements of $\mathcal{L}_k(1)$ by i places. Note that $\mathcal{L}_k^i(1)$ is a k-ary Gray code whose first entry is $k-i$. Now we can define a $k \times 1$ column vector with each entry equal to j as

$$j_k = [j, j, \ldots, j]^T \tag{6.5}$$

Then $\mathcal{L}_k(2)$ is defined by

$$\mathcal{L}_k(2) = \begin{bmatrix} 0_k & \mathcal{L}_k^0(1) \\ 1_k & \mathcal{L}_k^1(1) \\ \vdots & \vdots \\ (k-1)_k & \mathcal{L}_k^{k-1}(1) \end{bmatrix} \tag{6.6}$$

To extend this definition for an n-digit, k-ary Gray code starting with the all zero vector, we need the following notation:

$$\mathcal{L}_k(n) = \mathcal{L}_k^0(n) = [l_0(n), \ldots, l_{k^n-1}(n)]^T \tag{6.7}$$

where $l_i(n)$ is the $(i+1)$th n digit vector in $\mathcal{L}_k(n)$ and $l_0(n)$ is the all 0 n digit vector in the same listing, and

$$\mathcal{L}_k^m(n) = [l_0^m(n), \ldots, l_i^m(n), \ldots, l_{k^n-1}^m(n)]^T \tag{6.8}$$

where

$$l_i^m(n) = l_i(n) + m.l_{k^n-1}(n) \mod k \tag{6.9}$$

the product of a n digit vector with an integer $m \leq k-1$, is to be considered component wise multiplication of two elements of $\mathcal{Z}_k$ modulo k.

In terms of the notation, $\mathcal{L}_k(n)$ can be obtained from $\mathcal{L}_k(n-1)$ as follows.

$$\mathcal{L}_k(n) = \begin{bmatrix} 0_{k^{n-1}} & \mathcal{L}_k^0(n-1) \\ 1_{k^{n-1}} & \mathcal{L}_k^1(n-1) \\ \vdots & \vdots \\ k-1_{k^{n-1}} & \mathcal{L}_k^{k-1}(n-1) \end{bmatrix} \tag{6.10}$$

when $n > 1$ then $\mathcal{L}_k(n)$ has k blocks of length k^{n-1} each. $\mathcal{L}_k^m(n)$ is an arrangement obtained by cyclically shifting blocks by m and by simultaneously giving m cyclic shifts within each block.

The Lee distance gray code of radix 4 for $n \geq 3$, can then be defined as follows using Equation (6.10).

$$\mathcal{L}_4(n) = \begin{bmatrix} 0_{4^{n-1}} & \mathcal{L}_4^0(n-1) \\ 1_{4^{n-1}} & \mathcal{L}_4^1(n-1) \\ \vdots & \vdots \\ 3_{4^{n-1}} & \mathcal{L}_4^{k-1}(n-1) \end{bmatrix} \tag{6.11}$$

The n digit Lee distance Gray code over $\mathcal{Z}_4$ under the mapping $\mathcal{F} : \mathcal{Z}_4 \rightarrow \mathcal{Z}_2^2$ where $\mathcal{F}(0) \rightarrow 00$, $\mathcal{F}(1) \rightarrow 01$, $\mathcal{F}(2) \rightarrow 11$ and $\mathcal{F}(3) \rightarrow 10$ gives a binary Gray code of length $2n$. Under this mapping Lee distance between any two codewords is equal to the Hamming distance between them.

Example 9. *Table 6.1 shows the 2 digit Lee distance Gray code constructed using Equation (6.1) in column 1 and the corresponding binary Gray code obtained by applying $\mathcal{F}$ in column 2. Column 3 lists the 4 bit BRG code which is obviously different from the binary code obtained from $\mathcal{F}$.*

Let the set of binary vectors obtained by applying $\mathcal{F}$ from $\mathcal{L}_4(n)$ be denoted by $\mathcal{G}_B(N)$, where $N = 2n$, can be defined recursively, using Equation (6.12), as follows.

$$\mathcal{G}_B(N+2) = \begin{bmatrix} 00_{2^{n-2}} & \mathcal{G}_B^{(0)}(N) \\ 01_{2^{n-2}} & \mathcal{G}_B^{(1)}(N) \\ 11_{2^{n-2}} & \mathcal{G}_B^{(2)}(N) \\ 10_{2^{n-2}} & \mathcal{G}_B^{(3)}(N) \end{bmatrix} \tag{6.12}$$

where $\mathcal{G}_B^{(i)}(N)$ is obtained by first constructing $\mathcal{L}_4^{(i)}(N/2)$ code as defined in Equation (6.11) and then mapping the digits of the code under $\mathcal{F}$ to the corresponding binary vectors. The base case of this recursion occurs when $N = 2$ and is defined as

$$\mathcal{G}_B(2) = \begin{bmatrix} 00 \\ 01 \\ 11 \\ 10 \end{bmatrix} \tag{6.13}$$

6.2 Main Results

6.2.1 Constant Weight Vectors

Let the listing of vectors in $\mathcal{G}_B(N)$ which contains the same number of 1's, w, be $\mathcal{G}_B(N, w)$. In this listing, vectors of the same weight w are listed in the order they appear in $\mathcal{G}_B(N)$. It can be shown that consecutive vectors in $\mathcal{G}_B(N, w)$ differ exactly in two bits. This is a property that the $\mathcal{G}_B(N)$ code shares with the binary reflected Gray code [7]. To simplify the proof, it is helpful to define $\mathcal{G}_B(N, w)$ in a recursive manner analogous to the recursive definition of $\mathcal{G}_B(N)$: for $N > w > 0$ and N is even.

When $w \geq 2$,

$$\mathcal{G}_B(N,w) = \begin{bmatrix} 00 & \mathcal{G}_B^{(0)}(N-2,w) \\ 01 & \mathcal{G}_B^{(1)}(N-2,w-1) \\ 11 & \mathcal{G}_B^{(2)}(N-2,w-2) \\ 10 & \mathcal{G}_B^{(3)}(N-2,w-1) \end{bmatrix} \tag{6.14}$$

when $w = 1$,

$$\mathcal{G}_B(N,1) = \begin{bmatrix} 00 & \mathcal{G}_B^{(0)}(N-2,1) \\ 01 & \mathcal{G}_B^{(1)}(N-2,0) \\ 10 & \mathcal{G}_B^{(3)}(N-2,0) \end{bmatrix} \tag{6.15}$$

and $\mathcal{G}_B(N,0) = 0^N$, $\mathcal{G}_B(N,N) = 1^N$. Here, $\mathcal{G}_B^{(i)}(N,w)$ represents the set of all weight w vectors of length N, listed in the order they appear in $\mathcal{G}_B^{(i)}(N)$.

Theorem 6.2.1. *Successive codewords in $\mathcal{G}_B(N,w)$ differ in exactly 2 bits*

Proof. The proof is by induction on N. The values of N are even and start with 2 as already defined in Equation (6.11). When $N = 2$ and w can be either 0, 1, or 2. When $w = 2$ or $w = 0$ the theorem holds trivially. When $w = 1$, the two possible vectors are 01 and 10 in that order. Thus the theorem holds for $N = 2$.

Suppose it is true for some $N > 2$ and w, $0 \leq w \leq N$. It must be shown that the theorem holds for $N+2$ and w, $0 \leq w \leq N+2$. If $w = 0$ or $w = N+2$ it is trivially true since $\mathcal{G}_B(N+2)$ contains only one codeword with no 1-bits and only one codeword with $N+2$ 1-bits. For $1 \leq w \leq N$, the theorem holds by the inductive hypothesis if successive codewords are both in $00\mathcal{G}_B(N)$, $01\mathcal{G}_B^{(1)}(N)$, $11\mathcal{G}_B^{(2)}(N)$ or $10\mathcal{G}_B^{(3)}(N)$. Thus the only case left is to show that the last codewords with w 1-bits in one group of $\mathcal{G}_B(N)$ differs in exactly two bits from the first codeword with the subsequent group when $1 \leq w \leq N+1$. From Equation (6.10) it can be seen that four such cases arise where this has to be proved.

We can define these sets of the last and first vectors of the respective groups as follows: $B1 \rightarrow \{$ last vector of $00\mathcal{G}_B(N-2,w)$ and first vector of $01\mathcal{G}_B^{(1)}(N-2,w-1)$ $\}$, $B2 \rightarrow$ $\{$last vector of $01\mathcal{G}_B^{(1)}(N-2,w-1)$ and first vector of $11\mathcal{G}_B^{(2)}(N-2,w-2)\}$, $B3 \rightarrow$ $\{$last vector of $11\mathcal{G}_B^{(2)}(N-2,w-2)$ and first vector of $10\mathcal{G}_B^{(3)}(N-2,w-1)\}$ and $B4 \rightarrow \{$last vector of $10\mathcal{G}_B^{(3)}(N-2,w-1)$ and first vector of $00\mathcal{G}_B^{(0)}(N-2,w)\}$ are considered as separate cases in that order to complete the proof.

1. When $w = 1$.

   In this case, only vectors with most significant bit (msb) two bits set to 00, 01 or 10 are possible.

   (a) Consider the last vector in $00\mathcal{G}_B(N)$ and the first in $01\mathcal{G}_B^{(1)}(N)$. They are

   $$0\underline{0}\ \underline{1}0\ 0^{N-2} \tag{6.16}$$

   $$0\underline{1}\ \underline{0}0\ 0^{N-2} \tag{6.17}$$

   From the above equations it is seen that these vectors differ by exactly 2 bits.

   (b) Consider the last vector of $01\mathcal{G}_B^{(1)}(N)$ and the first vector of $10\mathcal{G}_B^{(3)}(N)$.

   $$\underline{01}\ 00\ 0^{N-2} \tag{6.18}$$

   $$\underline{10}\ 00\ 0^{N-2} \tag{6.19}$$

   (c) Consider the last vector of $10\mathcal{G}_B^{(3)}(N)$ and the first vector of $00\mathcal{G}_B^{(0)}(N)$.

   $$\underline{1}0\ 0^{N-2}\ 0\underline{0} \tag{6.20}$$

   $$\underline{0}0\ 0^{N-2}\ 0\underline{1} \tag{6.21}$$

   From the above equations it is seen that these vectors differ by exactly 2 bits.This completes the proof when $w = 1$.

2. When $2 \leq w \leq N+1$.

   (a) Vectors from $B1$

      i. When $2 \leq w < N$

         - *w is odd*: The last vector of the group $00\mathcal{G}_B(N)$ is of the form

$$\begin{aligned} & 00\ 10\ 0^{N-w-1}\ 1^{w-1} \\ = \ & 0\underline{0}\ 10\ 0^{N-w-1}\ \underline{1}1\ 1^{w-3} \end{aligned} \tag{6.22}$$

           The first vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$0\underline{1}\ 10\ 0^{N-w-1}\ \underline{0}1\ 1^{w-3} \tag{6.23}$$

           From equations (6.22) and (6.23) it can be seen that vectors differ in exactly 2 bits.

         - *w is even*: The last vector of the group $00\mathcal{G}_B(N)$ is of the form

$$0\underline{0}\ 10\ 0^{N-w-2}\ \underline{1}0\ 1^{w-2} \tag{6.24}$$

           The first vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$\begin{aligned} & 0\underline{1}\ 10\ 0^{N-w}\ 1^{w-2} \\ = \ & 0\underline{1}\ 10\ 0^{N-w-2}\ \underline{0}0\ 1^{w-2} \end{aligned} \tag{6.25}$$

           From equations (6.24) and (6.25) it can be seen that vectors differ in exactly 2 bits.

      ii. when $w = N$

         The last vector of the group $00\mathcal{G}_B(N)$ is of the form

$$0\underline{0}\ 1\underline{1}\ 1^{N-2} \tag{6.26}$$

The first vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$0\underline{1}\ 1\underline{0}\ 1^{N-2} \tag{6.27}$$

From equations (6.26) and (6.27) it can be seen that vectors differ in exactly 2 bits.

(b) Vectors from $B2$

i. When $3 < w \leq N+1$

- *w is odd*: The last vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$\begin{aligned} & 01\ 11\ 0^{N-w+1}\ 1^{w-3} \\ = \ & \underline{0}1\ 11\ 0^{N-w+1}\ \underline{1}1\ 1^{w-5} \end{aligned} \tag{6.28}$$

The first vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$\underline{1}1\ 11\ 0^{N-w+1}\ \underline{0}1\ 1^{w-5} \tag{6.29}$$

From equations (6.28) and (6.29) it can be seen that vectors differ in exactly 2 bits.

- *w is even*: The last vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$\underline{0}1\ 11\ 0^{N-w}\ \underline{1}0\ 1^{w-4} \tag{6.30}$$

The first vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$\begin{aligned} & 11\ 11\ 0^{N-w+2}\ 1^{w-4} \\ = \ & \underline{1}1\ 11\ 0^{N-w}\ \underline{0}0\ 1^{w-4} \end{aligned} \tag{6.31}$$

From equations (6.30) and (6.31) it can be seen that vectors differ in exactly 2 bits.

ii. when $w = 2$

The last vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$\underline{0}1\ \underline{1}0\ 0^{N-2} \tag{6.32}$$

The first vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$\underline{1}1\ \underline{0}0\ 0^{N-2} \tag{6.33}$$

From equations (6.32) and (6.33) it can be seen that vectors differ in exactly 2 bits.

iii. when $w = 3$

The last vector of the group $01\mathcal{G}_B^{(1)}(N)$ is of the form

$$\underline{0}1\ 1\underline{1}\ 0^{N-2} \tag{6.34}$$

The first vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$\underline{1}1\ 1\underline{0}\ 0^{N-2} \tag{6.35}$$

From equations (6.34) and (6.35) it can be seen that vectors differ in exactly 2 bits.

(c) Vectors from $B3$

i. When $3 \leq w \leq N$

- *w is odd*: The last vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$\begin{aligned} & 11\ 01\ 0^{N-w+1}\ 1^{w-3} \\ = \; & 1\underline{1}\ 01\ 0^{N-w-1}\ 0\underline{0}\ 1^{w-3} \end{aligned} \tag{6.36}$$

The first vector of the group $10\mathcal{G}_B^{(3)}(N)$ is of the form

$$1\underline{0}\ 01\ 0^{N-w-1}\ 0\underline{1}\ 1^{w-3} \tag{6.37}$$

From equations (6.36) and (6.37) it can be seen that vectors differ in exactly 2 bits.

- *w is even*: The last vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$1\underline{1}\ 01\ 0^{N-w}\ 1\underline{0}\ 1^{w-4} \tag{6.38}$$

The first vector of the group $10\mathcal{G}_B^{(3)}(N)$ is of the form

$$\begin{aligned} & 10\ 01\ 0^{N-w}\ 1^{w-2} \\ = \ & 1\underline{0}\ 01\ 0^{N-w}\ 1\underline{1}\ 1^{w-4} \end{aligned} \tag{6.39}$$

From equations (6.38) and (6.39) it can be seen that vectors differ in exactly 2 bits.

ii. when $w = 2$

The last vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$1\underline{1}\ 0\underline{0}\ 0^{N-2} \tag{6.40}$$

The first vector of the group $10\mathcal{G}_B^{(3)}(N)$ is of the form

$$1\underline{0}\ 0\underline{1}\ 0^{N-2} \tag{6.41}$$

From equations (6.40) and (6.41) it can be seen that vectors differ in exactly 2 bits.

iii. when $w = N + 1$

The last vector of the group $11\mathcal{G}_B^{(2)}(N)$ is of the form

$$1\underline{1}\ \underline{0}1\ 1^{N-2} \tag{6.42}$$

The first vector of the group $10\mathcal{G}_B^{(3)}(N)$ is of the form

$$1\underline{0}\ \underline{1}1\ 1^{N-2} \tag{6.43}$$

From equations (6.42) and (6.43) it can be seen that vectors differ in exactly 2 bits.

(d) Vectors from $B4$

i. When $2 \leq w \leq N$

- *w is odd:* The last vector of the group $10\mathcal{G}_B^{(3)}(N)$ is of the form

$$\begin{aligned} & 10\ 0^{N-w-1}\ 00\ 1^{w-1} \\ = \ & \underline{1}0\ 0^{N-w-1}\ 0\underline{0}\ 1^{w-1} \end{aligned} \tag{6.44}$$

The first vector of the group $00\mathcal{G}_B^{(0)}(N)$ is of the form

$$\underline{0}0\ 0^{N-w-1}\ 0\underline{1}\ 1^{w-1} \tag{6.45}$$

From equations (6.44) and (6.45) it can be seen that vectors differ in exactly 2 bits.

- *w is even*: The last vector of the group $10\mathcal{G}_B^{(3)}(N)$ is of the form

$$\underline{1}0\ 0^{N-w}\ 1\underline{0}\ 1^{w-2} \tag{6.46}$$

The first vector of the group $00\mathcal{G}_B^{(0)}(N)$ is of the form

$$\begin{aligned} & 0^{N-w+2}\ 1^{w} \\ = \ & \underline{0}0\ 0^{N-w}\ 1\underline{1}\ 1^{w-2} \end{aligned} \tag{6.47}$$

From equations (6.46) and (6.47) it can be seen that vectors differ in exactly 2 bits.

$\square$

6.2.2 New Classes of Binary Gray Codes

In this section, we define the new class of Gray codes, $\mathcal{G}_k$, where $k \geq 4$ and k is a power of 2. The construction of $\mathcal{G}_k$ for any radix k satisfying the afore mentioned properties is as follows.

Firstly, we define a mapping function, $\mathcal{F}_k$, which maps the digits in a radix k number system to its corresponding $\log_2(k)$-bit binary BRGC vector. $\mathcal{F}_k$ can be defined as follows.

$$\begin{aligned} \mathcal{F}_k : \{0, 1, 2, \ldots, k-1\} &\rightarrow \mathcal{Z}_2^{\log_2(k)} \\ i &\mapsto i \oplus \left\lfloor \frac{i}{2} \right\rfloor \end{aligned} \tag{6.48}$$

where i is the binary vector representation of a radix k number and $\oplus$ is the bit-by-bit exclusive-OR operation.

Next, we construct the n digit Lee distance Gray code using Equation 6.1. Now we replace the corresponding digits in the Gray code with their respective binary vectors using $\mathcal{F}_k$. Now we have to show that resulting $\log_2(k).n$ bit listing of vectors forms a binary Gray code.

Theorem 6.2.2. *Given a value $k, k \geq 4$ and $k = 2^r$ for some $r \geq 2$ and any n-digit Lee distance Gray code over k. If we replace the digits of $\mathcal{Z}_k$ under $\mathcal{F}_k$ the resultant set of vectors form a binary Gray code of length* $\log_2(k).n$ *bits.*

Proof. It is based on the following fact:

$$\text{for all } X, Y \in Z_k^n, D_L(X, Y) = 1 \Longrightarrow D_H(X, Y) = 1 \tag{6.49}$$

Let i be the position where X differs from Y, and x_i and y_i the corresponding components of X and Y respectively. Then

$$D_L(X,Y) = 1 \implies W_L(X-Y) = |x_i - y_i| \tag{6.50}$$

$$\implies x_i = y_i \pm 1 \bmod k. \tag{6.51}$$

The rightmost $\implies$ follows by analyzing the two possible cases which can happen in the equation

$$|x-y| = \min\{(x-y) \bmod k, k - [(x-y) \bmod k]\} = 1, \tag{6.52}$$

with $x, y \in Z_k = \{0, 1, 2, ..., k-1\}$. In fact, if $|x-y| = (x-y) \bmod k$ then

$$|x-y| = (x-y) \bmod k = 1 \implies x = y + 1 \bmod k. \tag{6.53}$$

If instead, $|x-y| = k - [(x-y) \bmod k]$ then

$$|x-y| = k - [(x-y) \bmod k] = 1 \tag{6.54}$$

$$\implies (x-y) \bmod k = k-1 \tag{6.55}$$

$$\implies x = y - 1 \bmod k. \tag{6.56}$$

Now, let $\mathcal{F}_k(X)$ and $\mathcal{F}_k(Y)$ be the images of X and Y through the function $\mathcal{F}_k$ defined in Equation (6.48). Since

- $\mathcal{F}_k$ defines a cyclic binary Gray code and
- $x_j = y_j$ if $j \neq i$ and $x_j = y_j \pm 1 \bmod k$ if $j = i$,

it follows,

$$D_H(\mathcal{F}_k(X), \mathcal{F}_k(Y)) = D_H(\mathcal{F}_k(x_i), \mathcal{F}_k(y_i)) \tag{6.57}$$

$$= D_H(\mathcal{F}_k(y_i \pm 1 \bmod k), \mathcal{F}_k(y_i)) \tag{6.58}$$

$$= 1 \tag{6.59}$$

$\square$

$\mathcal{L}_4(3)$	$\mathcal{G}_B(6)$	BRG(6)
000	00 00 00	00 00 00
001	00 00 01	00 00 01
002	00 00 11	00 00 11
003	00 10 10	00 00 10
013	00 01 10	00 01 10
010	00 01 00	00 01 11
011	00 01 01	00 01 01
012	00 01 11	00 01 00
022	00 11 11	00 11 00
023	00 11 10	00 11 01
020	00 11 00	00 11 11
021	00 11 01	00 11 10
031	00 10 01	00 10 10
032	00 10 11	00 10 11
033	00 10 10	00 10 01
030	00 10 00	00 10 00
130	01 10 00	01 10 00
$\vdots$	$\vdots$	$\vdots$
301	10 00 01	10 00 10
302	10 00 11	10 00 11
303	10 00 10	10 00 01
300	10 00 00	10 00 00

TABLE 6.1: Binary Gray code obtained from $\mathcal{F}$

CHAPTER 7

EDGE DISJOINT HAMILTONIAN CYCLES

So far, we have described how to generate 2 edge disjoint Hamiltonian cycles in a $T_{k_2 \times k_1}$. We also showed how this approach can be used to decompose torus networks with $n = 2^r$ (Chapter 4). In this chapter we describe the approach we propose to generate edge disjoint Hamiltonian cycles when the number of dimensions, n, is greater than 2 and $n \neq 2^r$ for any r. In the remaining analysis it is assumed that the radices are either all odd or all even and they are arranged so that $k_n \geq k_{n-1} \geq \ldots \geq k_1$.

7.1 Edge Disjoint Hamiltonian cycles when $n > 3$

Assume that there is a way to generate 3 edge disjoint Hamiltonian cycles in a $T_{k_3 \times k_2 \times k_1}$ graph . For higher dimensions, we first recursively decompose the given torus into tori of smaller dimensions. The base case of this recursion occurs when $n = 2$ or 3. We have already shown how we can obtain two edge disjoint Hamiltonian cycles in a 2 dimensional torus. Later in this section we show how 3 edge disjoint Hamiltonian cycles can be obtained in a 3 dimensional torus. In the rest of this section, we show how we decompose higher dimensional tori when the number of dimensions is even or odd. For the inductive hypothesis we assume that for a given n we can obtain $\lceil \frac{n}{2} \rceil$ edge disjoint Hamiltonian cycles from the $\lceil \frac{n}{2} \rceil$ dimensional torus.

7.1.1 *When* $n = 2m$ *and* $n > 3$

In this case $2m$ edge disjoint Hamiltonian cycles can be generated as follows.

$$\begin{aligned}
C_{k_{2m}\times k_{2m-1}\times\ldots\times k_1} &= (C_{k_{2m}} \times C_{k_{2m-1}} \times \ldots \times C_{k_{m+1}}) \\
&\quad \times(C_{k_m} \times C_{k_{m-1}} \times \ldots \times C_{k_1}) \\
&\quad \text{[Partition into two } m \text{ dimensional tori]} \\
&= (H'^{(1)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} + H'^{(2)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} \\
&\quad + \ldots + H'^{(m)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}}) \times (H'^{(1)}_{k_m\times k_{m-1}\times\ldots\times k_1} + \\
&\quad H'^{(2)}_{k_m\times k_{m-1}\times\ldots\times k_1} + \ldots + H'^{(m)}_{k_m\times k_{m-1}\times\ldots\times k_1}) \\
&\quad \text{[By Inductive Hypothesis obtain } m \\
&\quad \text{edge disjoint cycles in each torus]} \\
&= (H'^{(1)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} \times H'^{(1)}_{k_m\times k_{m-1}\times\ldots\times k_1}) \\
&\quad +(H'^{(2)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} \times H'^{(2)}_{k_m\times k_{m-1}\times\ldots\times k_1}) \\
&\quad + \ldots + (H'^{(m)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} \times H'^{(m)}_{k_m\times k_{m-1}\times\ldots\times k_1}) \\
&\quad \text{[Combining pairs of cycles, one from each group to} \\
&\quad \text{get } m \text{ 2 dimensional edge disjoint tori]} \\
&= (H^{1}_{k_{2m}\times k_{2m-1}\times\ldots\times k_1} + H^{(m+1)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_1}) \\
&\quad +(H^{2}_{k_{2m}\times k_{2m-1}\times\ldots\times k_1} + H^{(m+2)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_1}) \\
&\quad + \ldots + (H^{m}_{k_{2m}\times k_{2m-1}\times\ldots\times k_1} + H^{(2m)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_1}) \\
&\quad \text{[obtain } 2m \text{ edge disjoint cycles 2 from each} \\
&\quad \text{2 dimensional tori(Theorem 4.1.1)]}
\end{aligned}$$

Each $(H'^{(i)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} \times H'^{(i)}_{k_m\times k_{m-1}\times\ldots\times k_1})$ is a 2 dimensional torus with $(k_{2m} \times k_{2m-1} \times \ldots \times k_{m+1})$ nodes in one dimension and $(k_m \times k_{2m-1} \times \ldots \times k_1)$ nodes in the other. We can obtain 2 Hamiltonian cycles of length $k_{2m} \times k_{2m-1} \times \ldots \times k_1$ by the construction described in Chapter 4. Thus $2m$ edge disjoint Hamiltonian cycles can be obtained from the m components.

For example, when $k = 6$ we can obtain 6 edge disjoint Hamiltonian cycles as follows.

$$\begin{aligned}
T_{k_6 \times k_5 \times \ldots \times k_1} &= (C_{k_6} \times C_{k_5} \times C_{k_4}) \times (C_{k_3} \times C_{k_2} \times C_{k_1}) \\
&= (H'^{(1)}_{k_6 \times k_5 \times k_4} + H'^{(2)}_{k_6 \times k_5 \times k_4} + H'^{(3)}_{k_6 \times k_5 \times k_4}) \\
&\quad \times (H'^{(1)}_{k_3 \times k_2 \times k_1} + H'^{(2)}_{k_3 \times k_2 \times k_1} + H'^{(3)}_{k_3 \times k_2 \times k_1}) \\
&= (H'^{(1)}_{k_6 \times k_5 \times k_4} \times H'^{(1)}_{k_3 \times k_2 \times k_1}) + (H'^{(2)}_{k_6 \times k_5 \times k_4} \times H'^{(2)}_{k_3 \times k_2 \times k_1}) \\
&\quad + (H'^{(3)}_{k_6 \times k_5 \times k_4} \times H'^{(3)}_{k_3 \times k_2 \times k_1}) \\
&= (H^0_{k_6 \times k_5 \times \ldots \times k_1} + H^3_{k_6 \times k_5 \times \ldots \times k_1}) + (H^1_{k_6 \times k_5 \times \ldots \times k_1} + H^4_{k_6 \times k_5 \times \ldots \times k_1}) \\
&+ (H^2_{k_6 \times k_5 \times \ldots \times k_1} + H^5_{k_6 \times k_5 \times \ldots \times k_1})
\end{aligned}$$

7.1.2 When $n = 2m + 1$ *and* $n > 3$

In this case $2m + 1$ edge disjoint Hamiltonian cycles can be generated as follows.

$$\begin{aligned}
C_{k_{2m+1}\times k_{2m}\times\ldots\times k_1} &= (C_{k_{2m+1}}\times C_{k_{2m}}\times C_{k_{2m-1}}\times\ldots\times C_{k_{m+1}})\\
&\quad \times(C_{k_m}\times C_{k_{m-1}}\times\ldots\times C_{k_1})\\
&\quad \text{[Partition into one } m+1\\
&\quad \text{and one } m \text{ dimensional tori]}\\
&= (H'^{(1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}} + H'^{(2)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}}\\
&\quad +\ldots+H'^{(m+1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}})\times(H'^{(1)}_{k_m\times k_{m-1}\times\ldots\times k_1}+\\
&\quad H'^{(2)}_{k_m\times k_{m-1}\times\ldots\times k_1}+\ldots+H'^{(m)}_{k_m\times k_{m-1}\times\ldots\times k_1})\\
&\quad \text{[By Inductive Hypothesis obtain } m+1\\
&\quad \text{edge disjoint cycles in one torus and } m \text{ in the other]}\\
&= ((H'^{(1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}} + H'^{(m+1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}})\times H'^{(1)}_{k_m\times\ldots\times k_1})\\
&\quad +(H'^{(2)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}}\times H'^{(2)}_{k_m\times k_{m-1}\times\ldots\times k_1})\\
&\quad +\ldots+(H'^{(m)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}}\times H'^{(m)}_{k_m\times k_{m-1}\times\ldots\times k_1})\\
&\quad \text{[Combining pairs of cycles, one from each group to}\\
&\quad \text{get one 3 dimensional torus and } m-1\\
&\quad \text{ 2 dimensional edge disjoint tori]}\\
&= (H^1_{k_{2m+1}\times k_{2m}\times\ldots\times k_1}+H^{(m+1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_1}\\
&\quad +H^{(2m+1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_1})\\
&\quad +(H^2_{k_{2m+1}\times k_{2m}\times\ldots\times k_1}+H^{(m+2)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_1})\\
&\quad +\ldots+(H^m_{k_{2m+1}\times k_{2m}\times\ldots\times k_1}+H^{(2m)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_1})\\
&\quad \text{[obtain 3 edge disjoint cycles from first group and}\\
&\quad 2m-2 \text{ edge disjoint cycles 2 from each}\\
&\quad \text{2 dimensional tori (Theorem 4.1.1)]}
\end{aligned}$$

The first component, $((H'^{(1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}} + H'^{(m+1)}_{k_{2m+1}\times k_{2m}\times\ldots\times k_{m+1}}) \times H'^{(1)}_{k_m\times k_{m-1}\times\ldots\times k_1})$, is a 3 dimensional torus. We would like to provide a construction to obtain 3 edge disjoint Hamiltonian cycles from this component. The remaining $(H'^{(i)}_{k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1}} \times H'^{(i)}_{k_m\times k_{m-1}\times\ldots\times k_1})$ are 2 dimensional torus networks with $(k_{2m}\times k_{2m-1}\times\ldots\times k_{m+1})$ nodes in one dimension and $k_m\times k_{m-1}\times\ldots\times k_1$ nodes in the other. We obtain 2 Hamiltonian cycles of length $k_{2m}\times k_{2m-1}\times\ldots\times k_1$ from each component by the construction described in Chapter 4. Thus $2m+1$ edge disjoint Hamiltonian cycles can be obtained as $2(m-1)$ from the 2 dimensional tori and 3 from the first component. For example, the five cycles in $T_{k_5\times k_4\times\ldots\times k_1}$ can be obtained as follows

$$\begin{aligned}
T_{k_5\times k_4\times\ldots\times k_1} &= (C_{k_5}\times C_{k_4}\times C_{k_3})\times(C_{k_2}\times C_{k_1})\\
&= (H'^{(1)}_{k_5\times k_4\times k_3} + H'^{(2)}_{k_5\times k_4\times k_3} + H'^{(3)}_{k_5\times k_4\times k_3})\times(H'^{(1)}_{k_2\times k_1} + H'^{(2)}_{k_2\times k_1})\\
&= ((H'^{(1)}_{k_5\times k_4\times k_3} + H'^{(2)}_{k_5\times k_4\times k_3})\times H'^{(1)}_{k_2\times k_1}) + (H'^{(3)}_{k_5\times k_4\times k_3}\times H'^{(2)}_{k_2\times k_1})\\
&= (H^{0}_{k_5\times k_4\times\ldots\times k_1} + H^{3}_{k_5\times k_4\times\ldots\times k_1} + H^{4}_{k_5\times k_4\times\ldots\times k_1} +\\
&+ (H^{1}_{k_5\times k_4\times\ldots\times k_1} + H^{2}_{k_5\times k_4\times\ldots\times k_1})
\end{aligned}$$

Thus, from the above constructions we can see how we can decompose higher dimensional torus networks recursively in to those with 2 or 3 dimensions. We have already shown how we can construct 2 edge disjoint Hamiltonian cycles in a 2 dimensional torus. Now, if we have a method to construct 3 edge disjoint Hamiltonian cycles in a 3 dimensional torus then we can use this construction, along with that of the 2 dimensional case, to obtain edge disjoint Hamiltonian cycles in networks with arbitrary number of dimensions. This construction is not straightforward and in the next section we briefly show the approach we plan to use to solve this problem.

7.2 Edge Disjoint Hamiltonian cycles when $n = 3$

The problem of finding three edge disjoint Hamiltonian cycles in a $C_{k_3} \times C_{k_2} \times C_{k_1}$ can be solved as follows. Firstly, we can decompose a $T_{k_3 \times k_2 \times k_1}$ graph as follows.

$$\begin{aligned}
T_{k_3 \times k_2 \times k_1} &= C_{k_3} \times C_{k_2} \times C_{k_1} && (7.1)\\
&= (C_{k_3} \times C_{k_2}) \times C_{k_1} && (7.2)\\
&= (H^1_{k_3 \times k_2} + H^2_{k_3 \times k_2}) \times C_{k_1} && (7.3)\\
&= (H^1_{k_3 \times k_2 \times k_1} + H^2_{k_3 \times k_2 \times k_1} + H^3_{k_3 \times k_2 \times k_1}) && (7.4)
\end{aligned}$$

In the above, $H^1_{k_3 \times k_2}$ and $H^2_{k_3 \times k_2}$ are two Gray codes in $Z_{k_3 \times k_2}$ obtained using the construction for the two dimensional case as described above. That is, $H^1_{k_3 \times k_2}$ and $H^2_{k_3 \times k_2}$ are two edge disjoint Hamiltonian cycles in $T_{k_3 \times k_2}$ graph (Equation 7.3).
The cross product of the $T_{k_3 \times k_2}$ graph with C_{k_1} results in a graph with cardinality $(k_3 \times k_2) \times k_1$. Consider the cross product of $H^1_{k_3 \times k_2}$, which is equivalent to $C_{k_3 \times k_2}$, with C_{k_1} (Equation 7.3). We can obtain two edge disjoint Hamiltonian cycles over the $T_{k_3 \times k_2 \times k_1}$ graph by considering one dimension as having $k_3 \times k_2$ nodes labeled using $H^1_{k_3 \times k_2}$ and the other by labels of C_{k_1}. In this graph we can obtain two edge disjoint Hamiltonian cycles using the construction for the two dimensional case (Figures 7.1 (a) and (b)). The remaining edges in $T_{k_3 \times k_2 \times k_1}$, $H^3_{k_3 \times k_2 \times k_1}$, unfortunately, do not result in a Hamiltonian cycle (Figure 7.1 (c)). Also between them, the three cycles partition all the edges in $T_{k_3 \times k_2 \times k_1}$.
In order to generate three edge disjoint Hamiltonian cycles, some edges in $H^1_{k_3 \times k_2 \times k_1}$ or $H^2_{k_3 \times k_2 \times k_1}$ must be exchanged with those in $H^3_{k_3 \times k_2 \times k_1}$. The edges that will be removed from $H^1_{k_3 \times k_2 \times k_1}$ or $H^2_{k_3 \times k_2 \times k_1}$ are called *exchange* edges. However, selecting proper exchange edges is not straightforward; note that the Hamiltonian property of $H^1_{k_3 \times k_2 \times k_1}$ or $H^2_{k_3 \times k_2 \times k_1}$ must be maintained even after the exchange of edges is done. We propose to use some horizontal edges in the $H^1_{k_3 \times k_2 \times k_1}$ graph in (Figure 7.1 (a))

as the exchange edges to make $H^3_{k_3\times k_2\times k_1}$ Hamiltonian. The problem now, is finding the proper set horizontal edges to exchange so that we can obtain three edge disjoint Hamiltonian cycles for any k_3, k_2, k_1 (Equation 7.4).

Example 10. *Consider the problem of finding 3 edge disjoint Hamiltonian cycles in a $T_{5\times5\times3}$ network. First, we consider the two most significant dimensions. We apply the construction of the 2-dimensional case to obtain to edge disjoint cycles, $H^1_{5\times3}$ and $H^3_{5\times3}$, of length 15 (as in Equation (7.3)). Next, we consider $H^1_{5\times3}\times C_3$ and apply the construction of the 2-dimensional case to obtain two edge disjoint Hamilton cycles in $T_{5\times3\times3}$. Figure 7.1(a) shows these cycles in the left component and the remaining edges in $T_{5\times3\times3}$ (i.e., those not used by $H^1_{5\times3\times3}$ and $H^2_{5\times3\times3}$) are shown in the right component.*

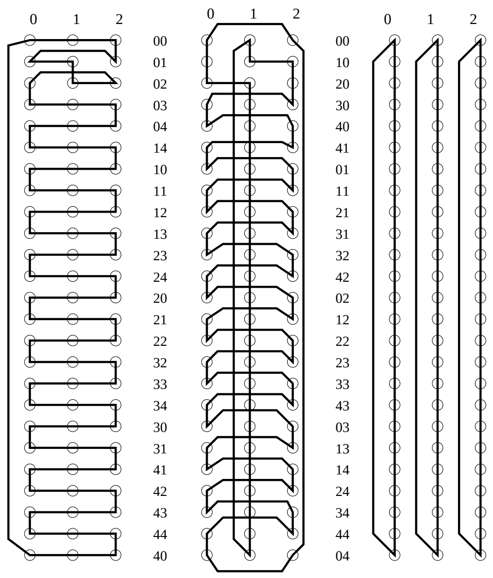
Note that the first two cycles in Figure 7.1(a) are Hamiltonian but the third cycle is not. This cycle can be made Hamiltonian by exchanging edges from this cycle with those in the first cycle.

Now if we exchange edges (000,001), (100,101), (101,102) and (201,202) from cycle 1 with edges (000,100), (001,101),(101,201), and (202,102) we see that the resultant graph has three edge disjoint Hamiltonian cycles (Figure 7.1(b)).

Even though it was easy in this case, the selection of exchange edges which will result in three edge disjoint Hamiltonian cycles is not straight forward and this is the problem we are interested in solving. In general we need to pick $2(k_1-1)$ exchange edges.

7.2.1 Choosing the Exchange Edges

In this section, we show how to choose the exchange edges. In particular, we pick horizontal edges to exchange from $H^1_{k_3\times k_2\times k_1}$ with some vertical edges from $H^3_{k_3\times k_2\times k_1}$. We need to exchange $2(k_1-1)$ edges and we think of this as being accomplished in


(a) 2 edge disjoint Hamiltonian cycles and a partial cycle

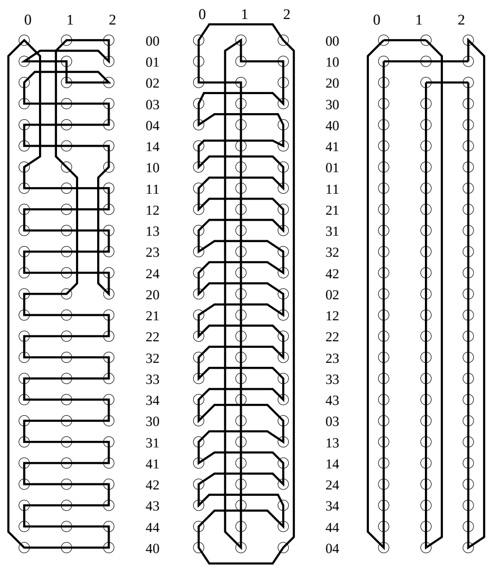

(b) 3 edge disjoint Hamiltonian cycles

FIGURE 7.1: 3 Edge Disjoint Hamiltonian Cycles in a $5 \times 3 \times 3$ Torus

$(k_1 - 1)$ steps where in each step a pair of two edges is exchanged between these graphs. As it turns out we need to consider the cases where k_is are all odd or all even separately. Before, we show the set of edges that need to be exchanged we briefly mention some of the properties of the $T_{k_3 \times k_2 \times k_1}$ graph that would help in understanding the rationale behind the choice of exchange edges.
Firstly, we observe that the labeling of one dimension of the $H^1_{k_3 \times k_2 \times k_1}$ graph consists of a Gray code $G^1_{k_3 \times k_2}$ and in $H^3_{k_3 \times k_2 \times k_1}$ using the other edge disjoint Hamiltonian cycle $G^2_{k_3 \times k_2}$. Thus, two nodes which are adjacent in $H^1_{k_3 \times k_2 \times k_1}$ are not in $H^3_{k_3 \times k_2 \times k_1}$. Now, consider the sequence of words $\psi = \{00, 10, 20, \ldots, (k_3 - 2)0, (k_3 - 1)0\}$. Note that this sequence of nodes appear at consecutive positions in the Gray code $G^2_{k_3 \times k_2}$ and are hence not adjacent to each other in $G^1_{k_3 \times k_2}$. We will now show some important properties of ψ.

Lemma 7.2.1. *In the Gray code $G^1_{k_3 \times k_2}$, if we number the listing starting from 0 to $(k_3 \times k_2) - 1$. The difference in position of occurrences of any pair of consecutive words in ψ is even if k_is are all odd.*

Proof. By the definition of the two dimensional Gray code, $G^1_{k_3 \times k_2}$, it can be seen that the least significant digit is shifted upwards by one position between two blocks of vectors with the same most significant digit. Also, each of the k_2 values occur exactly once for a given value of the MSB digit. This implies that it takes $k_2 + 1$ positions for the LSB digit to repeat. Since, k_2 is odd the result follows. □

The following lemma follows from the previous one, as a corollary.

Lemma 7.2.2. *In the Gray code $G^1_{k_3 \times k_2}$, if we number the listing starting from 0 to $k_3 \times k_2 - 1$. The difference in position of occurrences of any pair of consecutive words in ψ is odd if k_is are all even.*

Choosing the right set of exchange edges is not straightforward. In general, we are interested in exchanging $k_1 - 1$ pairs of edges and we view this as a process accomplished in $k_1 - 1$ steps. At the end of each step we aim to preserve the following property: $H^1_{k_3 \times k_2 \times k_1}$ is still a Hamiltonian cycle over $k_3 \times k_2 \times k_1$ and exactly $k_3 \times k_2$ nodes are added to $H^3_{k_3 \times k_2 \times k_1}$.

Firstly, we observe from the construction of, $H^1_{k_3 \times k_2 \times k_1}$, the values in the least significant digit follow a specific pattern of change. For some values of (v_3, v_2), of the radices k_3 and k_2, the LSB digit is a cyclic shift and for the remainder for a given (v_3, v_2) the digit values are either in ascending or descending order alternatively. In this case if for a given (v_3, v_2) the LSB digits are in ascending order, it is denoted as $(v_3, v_2) \uparrow$ and when they are in descending order as $(k_3, k_2) \downarrow$

When $I(v_3, v_2) \geq k_1$, the LSB digits alternatively increase and decrease starting with the first sequence in ascending order. When $I(v_3, v_2) < k_1$, then digit are cyclically shifted upward starting with an ascending order listing. Obviously, for $(v_3, v_2) = (0, 0)$ the LSB digits are in ascending order.

Lemma 7.2.3. *The sequence of LSB digit values for* $(v_3, v_2) = (1, 0)$ *is always in descending order in* $H^1_{k_3 \times k_2 \times k_1}$.

Proof. This proof applies to any k irrespective of whether it is even or odd. Consider the following table

k_3, k_2	k_1
00	$\{0, 1, \ldots, k_1 - 1\}$
01	$\{k_1 - 1, 0, 1, \ldots, k_1 - 2\}$
$\vdots$	$\vdots$
$0(k_1 - 1)$	$\{1, 2, 3, \ldots, k_1 - 1, 0\}$
$1(k_1 - 1)$	$\{0, 1, 2, \ldots, k_1 - 1\}(\uparrow)$
10	$\{k_1 - 1, k_1 - 2, \ldots, 1, 0\}(\downarrow)$
$\vdots$	$\vdots$

(7.5)

which lists the $H^1_{k_3 \times k_2 \times k_1}$ cycle starting with node (000). In the last two lines the pattern of change for digits k_3, k_2 itself is due to the fact that they are labeled from $H^1_{k_3 \times k_2}$ and the proof follows. $\square$

The following two lemmas follows from Lemmas 7.2.1 and 7.2.2.

Lemma 7.2.4. *When k_is are odd, the order in which the LSB digit changes in $H^1_{k_3 \times k_2 \times k_1}$ for the nodes in τ, is as follows:* $\tau = \{(00)\uparrow, (10)\downarrow, (20)\downarrow, \ldots, (k_3-1, 0)\downarrow\}$

Lemma 7.2.5. *When k_is are even, the order in which the LSB digit changes in $H^1_{k_3 \times k_2 \times k_1}$ for the nodes in τ, is as follows:* $\tau = \{(00)\uparrow, (10)\downarrow, (20)\uparrow, \ldots, (k_3-2, 0)\uparrow, (k_3 - 1, 0)\downarrow\}$

Exchange Edges when k_is are odd

As already mentioned we need to exchange $k_1 - 1$ pairs of edges to make $H^3_{k_3 \times k_2 \times k_1}$ to make it a Hamiltonian cycle. Here k_1 is odd and the number of pairs to be exchanged is even.

The set of edges that need to be exchanged between $H^1_{k_3\times k_2\times k_1}$ and $H^3_{k_3\times k_2\times k_1}$ graphs where k is odd is as follows.

When $k_1 = 3$,

$$\varphi = \left\{\begin{array}{ll} [000,001] & [000,100] \\ [100,101] & [001,101] \\ [101,102] & [101,201] \\ [201,202] & [202,102] \end{array}\right\} \tag{7.6}$$

In the above equation edges are represented by pairs of nodes they connect as $[N_1, N_2]$ where Ns are the node labels. The edges in the first column belong to $H^1_{k_3\times k_2\times k_1}$ and those in the second belong to $H^3_{k_3\times k_2\times k_1}$. The edge in the first column is exchanged with that in the second.

Theorem 7.2.1. *For $k_1 = 3$ if we pick as exchange edges those that belong to φ, $H^1_{k_3\times k_2\times k_1}$ is still Hamiltonian and $H^3_{k_3\times k_2\times k_1}$ is also turned into a Hamiltonian cycle.*

Proof. From Figure 7.2 (a) and (b) it is clear that the set of exchange edges result in 3 Hamiltonian cycles after the exchange in a $C_{k_3} \times C_{k_2} \times C_3$ torus graph.

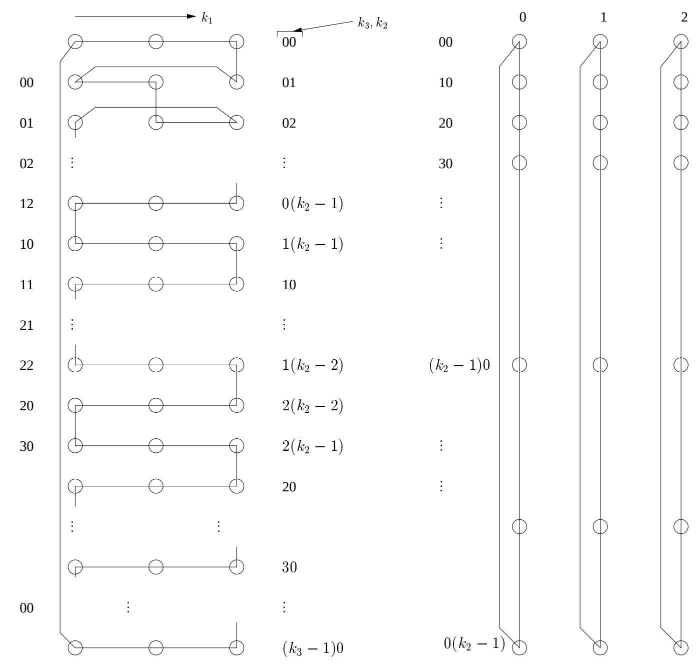
□

For $k_1 > 3$, firstly we define a set of exchange edges ζ as follows

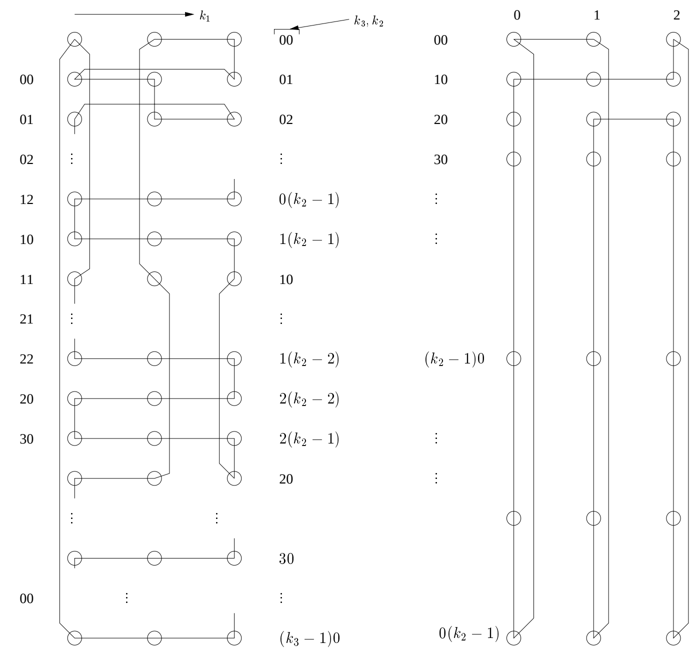
$$\zeta = \begin{array}{ll} \{[(k_1-j)0j,(k_1-j)0(j+1)] & [(k_1-j)0j,(k_1-j+1)0j]\,, \\ ([(k_1-j+1)0j,(k_1-j+1)0(j+1)] & [(k_1-j)0(j+1),(k_1-j+1)0(j+1)]\} \end{array} \tag{7.7}$$

for all j where, $2 \le j \le k_1 - 2$. Here the edges in the first column belong to

We now briefly show the intuition behind choosing ζ the way we have. From Lemma 7.2.4 we see that the for nodes of the form $x0$ the LSB digit is always in the decreasing order. This means that the direction of visit is from $(k_1 - 1)$ to 0 (Figure 7.3). For



(a) before exchange



(b) after exchange

FIGURE 7.2: $H^1_{k_3 \times k_2 \times 3}$ and $H^3_{k_3 \times k_2 \times 3}$ graphs

some $x0$ and $(x+1)0$ in $H^1_{k_3\times k_2\times k_1}$, which are not adjacent. We know that these nodes are adjacent in $H^3_{k_3\times k_2\times k_1}$. Consider what happens when we pick a pair horizontal edges of the form

$$[(x0q, x0(q+1)), ((x+1)0q, (x+1)0(q+1))] \tag{7.8}$$

from $H^1_{k_3\times k_2\times k_1}$ and exchange it with a pair of the edges of the form

$$[(x0q, (x+1)0q), (x0(q+1), (x+1)0(q+1))] \tag{7.9}$$

from $H^3_{k_3\times k_2\times k_1}$. Before this exchange the order of visit of these nodes in $H^1_{k_3\times k_2\times k_1}$ is (Figure 7.3)

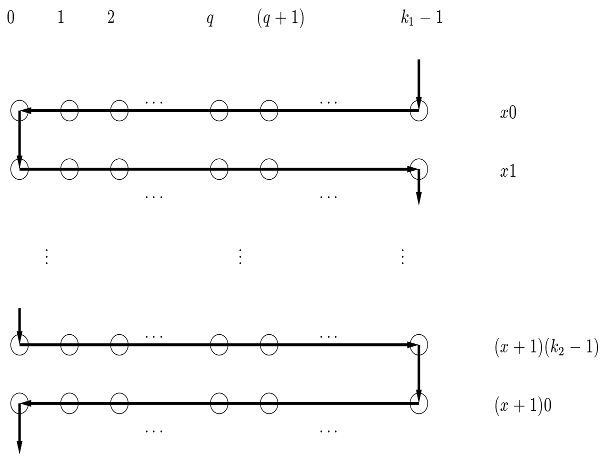


FIGURE 7.3: $H^1_{k_3\times k_2\times k_1}$ before exchange

$$x0(q+1), x0q, x0(q-1), \ldots, x00 \tag{7.10}$$

$$x10, x11, x12, \ldots, x1(k_1-1) \tag{7.11}$$

$$\vdots \tag{7.12}$$

$$x(k_2-1)0, x(k_2-1)1, x(k_2-1)2, \ldots, x(k_2-1)(k_1-1) \tag{7.13}$$

$$(x+1)0(k_1-1), (x+1)0(k_1-2), \ldots, (x+1)0(q+1), (x+1)0(q), \ldots \tag{7.14}$$

.

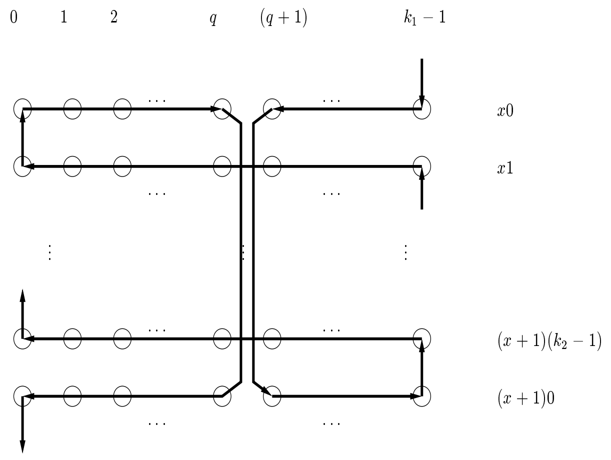


FIGURE 7.4: $H^1_{k_3 \times k_2 \times k_1}$ after exchange

After the exchange it is (Figure 7.4)

$$x0(q+1), \quad (x+1)0(q+1) \quad ,\dots,(x+1)0(k_1-1) \tag{7.15}$$

$$x(k_2-1)(k_1-1), \quad x(k_2-1)(k_1-2) \quad ,\dots,x(k_2-1)0 \tag{7.16}$$

$$\vdots \tag{7.17}$$

$$x1(k_2-1), \quad x1(k_1-2) \quad ,x1(k_1-3),\dots,x10 \tag{7.18}$$

$$x00, \quad x01 \quad ,\dots,x0q,(x+1)0q,\dots \tag{7.19}$$

It can also similarly seen that after the exchange we have added exactly one cycle of $k_3 \times k_2$ to the base cycle in $H^3_{k_3 \times k_2 \times k_1}$ (Figure 7.5).

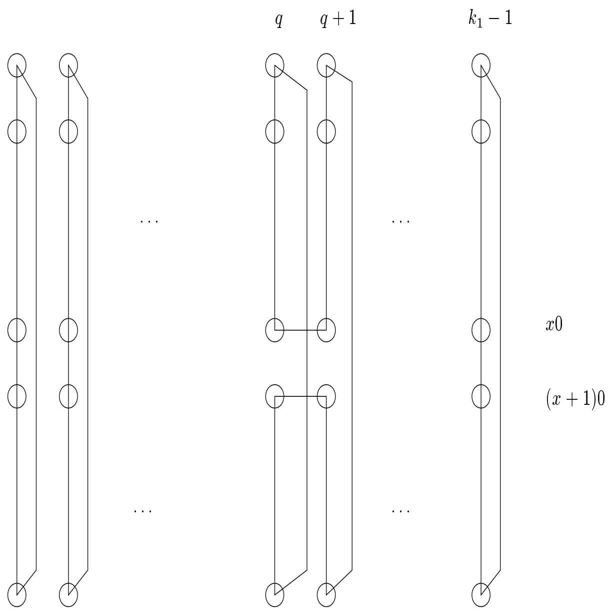


FIGURE 7.5: $H^3_{k_3 \times k_2 \times k_1}$ after exchange

Now, consider what happens when we exchange the following edges from $H^1_{k_3\times k_2\times k_1}$.

$$[(x0q, x0(q+1)), ((x+1)0q, (x+1)0(q+1))] \tag{7.20}$$

$$[(x0(q+1), (x)0(q+2)), ((x-1)0(q+1), (x-1)0(q+2))] \tag{7.21}$$

are exchanged with the following in $H^3_{k_3\times k_2\times k_1}$ (Figures 7.6 and 7.7).

$$[(x0q, (x+1)0q), (x0(q+1), (x+1)0(q+1))] \tag{7.22}$$

$$[(x0(q+1), (x-1)0(q+1)), (x0(q+2), (x11)0(q+2))] \tag{7.23}$$

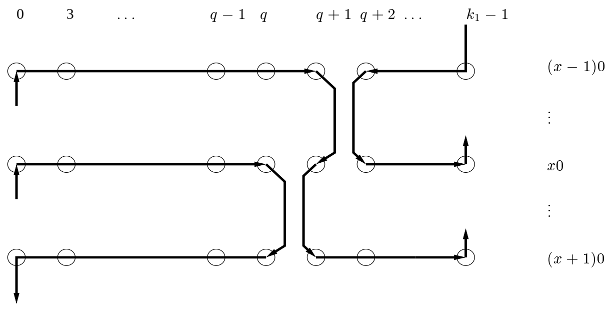


FIGURE 7.6: $H^1_{k_3\times k_2\times k_1}$ after exchange

Following in this manner, we can see how this choice of edges helps in making the $H^3_{k_3\times k_2\times k_1}$ cycle of size $k_3 \times k_2 \times (k_1-3)$ nodes while maintaining the Hamiltonian property of $H^1_{k_3\times k_2\times k_1}$. We will later see how the remaining two cycles can be added. Next, we prove that when we exchange those edges in ζ of $H^1_{k_3\times k_2\times k_1}$ with those in $H^3_{k_3\times k_2\times k_1}$. We first list the nodes starting from 000 for the cycle $H^1_{k_3\times k_2\times k_1}$ which is as follows.

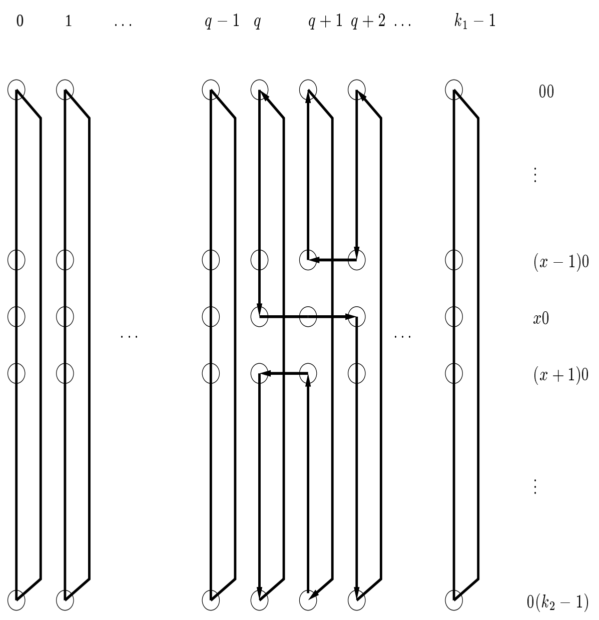


FIGURE 7.7: $H^3_{k_3 \times k_2 \times k_1}$ after exchange

k_3, k_2	k_1
00	$0, 1, 2, \ldots, k_1 - 1$
01	$k_1 - 1, 0, 1, \ldots, k_1 - 2$
$\vdots$	$\vdots$
$0(k_1 - 1)$	$1, 2, \ldots, (k_1 - 1), 0$
$0(k_1)$	$0, 1, 2, \ldots, (k_1 - 1)$
$0(k_1 + 1)$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
$0(k_2 - 1)$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$1(k_2 - 1)$	$0, 1, 2, \ldots, (k_1 - 1)$

Continued on next page

k_3, k_2	k_1
10	$(k_1-1), (k_1-2), \ldots, 0$
11	$0, 1, 2, \ldots, k_1-1$
$\vdots$	$\vdots$
20	$(k_1-1), (k_1-2), \ldots, 0$
$\vdots$	$\vdots$
$(k_3-1)0$	$(k_1-1), (k_1-2), \ldots, 0$

Note that the last node in each row is adjacent to the first node of the following row and this listing is a Hamiltonian cycle.

The nodes of $H^3_{k_3 \times k_2 \times k_1}$ can be listed as

k_3, k_2	k_1
$00, 10, 20, \ldots, (k_2-1)0, \ldots, 0(k_2-1)$	0
$00, 10, 20, \ldots, (k_2-1)0, \ldots, 0(k_2-1)$	1
$00, 10, 20, \ldots, (k_2-1)0, \ldots, 0(k_2-1)$	2
$\vdots$	$\vdots$
$00, 10, 20, \ldots, (k_2-1)0, \ldots, 0(k_2-1)$	(k_1-1)

(7.24)

In the above the listing each row constitutes a cycle with the first node in each row being adjacent to the last in the same row. Obviously the above listing is not Hamiltonian.

Theorem 7.2.2. *When the set of exchange edges as defined by, ζ, are exchanged the following invariants holds after exchange of each pair of edges*

- $H^1_{k_3 \times k_2 \times k_1}$ *remains a Hamiltonian cycle*
- *Exactly one cycle of length $k_3 \times k_2$ nodes is added to $H^3_{k_3 \times k_2 \times k_1}$.*

Proof. Here, we think of the process of exchange as a step-by-step process where we exchange a pair of edges from $H^1_{k_3\times k_2\times k_1}$ with a pair of edges in $H^3_{k_3\times k_2\times k_1}$ in each step. We consider as steps the different values of j and the pair of edges is given by those in ζ for the corresponding value of j. The proof, then, is by induction on j. Before the exchange we note that $H^1_{k_3\times k_2\times k_1}$ is a Hamiltonian cycle and $H^3_{k_3\times k_2\times k_1}$ contains k_1 disconnected cycles of length $k_3 \times k_2$. Thus, the state before exchange, in $H^1_{k_3\times k_2\times k_1}$ and $H^3_{k_3\times k_2\times k_1}$ are as in Equations 7.2.1 and 7.24 respectively.

Now consider the base case of the induction when $j = 2$. The edges picked for exchange from $H^1_{k_3\times k_2\times k_1}$ are $[(k_1-2)02, (k_1-2)03]$ and $[(k_1-1)02, (k_1-1)03]$. The edges removed from $H^3_{k_3\times k_2\times k_1}$ are $[(k_1-2)02, (k_1-1)02]$ and $[(k_1-2)03, (k_1-1)03]$. After removing these edges we add the edges removed from $H^1_{k_3\times k_2\times k_1}$ to $H^3_{k_3\times k_2\times k_1}$, we have

k_3, k_2	k_1
$00, 10, 20, \ldots, (k_2-1), \ldots, 0(k_2-1)$	0
$00, 10, 20, \ldots, (k_2-1), \ldots, 0(k_2-1)$	1
$00, 10, 20, \ldots, (k_1-2)0$	2
$(k_1-2)0, (k_1-3)0, \ldots, 00, 0(k_2-1), \ldots, (k_1-1)0$	3
$(k_1-1)0, k_10, \ldots, 0(k_2-1)$	2
$\vdots$	$\vdots$
$00, 10, 20, \ldots, (k_2-1), \ldots, 0(k_2-1)$	(k_1-1)

(7.25)

From the above equation, it can be seen that the cycle with $k_1 = 3$ has been added to the cycle with $k_1 = 2$. Now consider, what happens to $H^1_{k_3\times k_2\times k_1}$ after adding edges from $H^3_{k_3\times k_2\times k_1}$, we have

k_3, k_2	k_1
00	$0, 1, 2, \ldots, k_1 - 1$
01	$k_1 - 1, 0, 1, \ldots, k_1 - 2$
$\vdots$	$\vdots$
$0(k_1 - 1)$	$1, 2, \ldots, (k_1 - 1), 0$
$0(k_1)$	$0, 1, 2, \ldots, (k_1 - 1)$
$0(k_1 + 1)$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
$0(k_2 - 1)$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$1(k_2 - 1)$	$0, 1, 2, \ldots, (k_1 - 1)$
10	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
20	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - 2)0$	$(k_1 - 1), (k_1 - 2), \ldots, 3$
$(k_1 - 1)0$	$3, 4, \ldots, (k_1 - 1)$
$(k_1 - 1)(k_1 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - 2)1$	$k_1 - 1, k_1 - 2, \ldots, 0$
$(k_1 - 2)0$	$0, 1, 2$
$(k_1 - 1)0$	$2, 1, 0$
$\vdots$	$\vdots$
$(k_2 - 1)0$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$

Continued on next page

k_3, k_2	k_1
$(k_3 - 1)0$	$(k_1 - 1), (k_1 - 2), \ldots, 0$

From the above equation it can be seen that $H^1_{k_3 \times k_2 \times k_1}$ is still a Hamiltonian cycle. Assuming that the hypothesis holds up to some value of $j = \gamma - 1$, where $2 < \gamma < k_1 - 2$, we have to show that the induction hypothesis holds for $j = \gamma$.
The edges picked for exchange from $H^1_{k_3 \times k_2 \times k_1}$ are $[(k_1 - \gamma)0\gamma, (k_1 - \gamma)0(\gamma + 1)]$ and $[(k_1 - \gamma + 1)0\gamma, (k_1 - 1)0(\gamma + 1)]$. The edges removed from $H^3_{k_3 \times k_2 \times k_1}$ are $[(k_1 - \gamma)0\gamma, (k_1 - \gamma + 1)0\gamma]$ and $[(k_1 - \gamma)0(\gamma + 1), (k_1 - \gamma + 1)0(\gamma + 1)]$. After removing these edges we add the edges removed from $H^1_{k_3 \times k_2 \times k_1}$ to $H^3_{k_3 \times k_2 \times k_1}$, we need to consider the case where γ is even and odd separately. Firstly, when γ is odd, we have

k_3, k_2	k_1
$00, 10, 20, \ldots, (k_2 - 1), \ldots, 0(k_2 - 1)$	0
$00, 10, 20, \ldots, (k_2 - 1), \ldots, 0(k_2 - 1)$	1
$00, 10, 20, \ldots, (k_1 - 2)0$	2
$(k_1 - 2)0$	3
$(k_1 - 2)0, (k_1 - 1)0, \ldots, 0(k_2 - 1), \ldots, (k_1 - 4)0$	4
$\vdots$	$\vdots$
$(k_1 - \gamma - 1)0$	$\gamma - 2$
$(k_1 - \gamma - 1)0, \ldots, 0(k_2 - 1), 00, \ldots (k_1 - \gamma - 3)0$	$\gamma - 1$
$(k_1 - \gamma - 3)0, (k_1 - \gamma - 4), \ldots, 00, \ldots, (k_1 - \gamma - 2)0$	γ
$(k_1 - \gamma - 2)0$	$\gamma - 1$
$(k_1 - \gamma - 2)0, \ldots, (k_1 - \gamma - 3)0 \ldots 00, 0(k_2 - 1), \ldots (k_1 - \gamma - 4)0$	$\gamma - 2$
$\vdots$	$\vdots$
$(k_1 - 1)0, k_1 0, \ldots, 0(k_2 - 1)$	2

Continued on next page

k_3, k_2	k_1
$\vdots$	$\vdots$
$00, 10, 20, \dots, (k_2-1), \dots, 0(k_2-1)$	(k_1-1)

and in the case where γ is even, we have

k_3, k_2	k_1
$00, 10, 20, \dots, (k_2-1), \dots, 0(k_2-1)$	0
$00, 10, 20, \dots, (k_2-1), \dots, 0(k_2-1)$	1
$00, 10, 20, \dots, (k_1-2)0$	2
$(k_1-2)0$	3
$(k_1-2)0, (k_1-1)0, \dots, 0(k_2-1), \dots, (k_1-4)0$	4
$\vdots$	$\vdots$
$(k_1-\gamma-4)0, k_1-\gamma-3)0, \dots, 00, 0(k_2-1), \dots, (k_1-\gamma-2)0$	$\gamma-2$
$(k_1-\gamma-2)0$	$\gamma-1$
$(k_1-\gamma-2)0, (k_1-\gamma-2), \dots, 00, \dots, (k_1-\gamma-1)0$	γ
$(k_1-\gamma-1)0, (k_1-\gamma-2)0, \dots, 00, 0(k_2-1), \dots, (k_1-\gamma-3)0$	$\gamma-1$
$(k_1-\gamma-3)0$	$\gamma-2$
$\vdots$	$\vdots$
$(k_1-1)0, k_10, \dots, 0(k_2-1)$	2
$\vdots$	$\vdots$
$00, 10, 20, \dots, (k_2-1), \dots, 0(k_2-1)$	(k_1-1)

(7.26)

From both the cases shown above (Equations 7.2.1 and 7.26) we can see that one cycle with $k_3 \times k_2$ nodes with LSB digit equal to γ is added to the existing cycle. In the $H^1_{k_3\times k_2\times k_1}$ graph the addition of edges removed leads to

k_3, k_2	k_1
00	$0, 1, 2, \ldots, k_1 - 1$
01	$k_1 - 1, 0, 1, \ldots, k_1 - 2$
$\vdots$	$\vdots$
$0(k_1 - 1)$	$1, 2, \ldots, (k_1 - 1), 0$
$0(k_1)$	$0, 1, 2, \ldots, (k_1 - 1)$
$0(k_1 + 1)$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
$0(k_2 - 1)$	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$1(k_2 - 1)$	$0, 1, 2, \ldots, (k_1 - 1)$
10	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
20	$(k_1 - 1), (k_1 - 2), \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - \gamma - 1)0$	$k_1 - 1, k_1 - 2, \ldots, \gamma + 1$
$(k_1 - \gamma)0$	$k_1 - 1$
$(k_1 - \gamma)(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - \gamma - 1)0$	$0, 1, \ldots, \gamma$
$(k_1 - \gamma)0$	γ
$(k_1 - \gamma + 1)0$	$\gamma, \gamma + 1, \ldots, k_1 - 1$
$(k_1 - \gamma + 1)(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - \gamma)0$	$0, 1, \ldots, \gamma - 1$

Continued on next page

k_3, k_2	k_1
$(k_1 - \gamma + 1)0$	$\gamma - 1$
$\vdots$	$\vdots$
$(k_1 - 2)0$	3
$(k_1 - 1)0$	$3, 4, \ldots, k_1 - 1$
$(k_1 - 1)(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - 2)0$	$0, 1, 2$
$(k_1 - 1)0$	$2, 1, 0$
$(k_1)0$	$0, 1, 2, \ldots, k_1 - 1$
$\vdots$	$\vdots$
$(k_3 - 1)0$	$k_1 - 1, k_1 - 2, \ldots, 0$

From the above it can be seen that exactly one cycle is added to $H^3_{k_3 \times k_2 \times k_1}$ while maintaining the Hamiltonian property of $H^1_{k_3 \times k_2 \times k_1}$. □

The complete set of exchange edges when the radices are odd is then given by

$$\phi = \varphi + \zeta \tag{7.27}$$

where ϕ is the set of edges to be exchanged when $k > 3$ and $+$ indicates the set concatenation operation of the sets represented by φ and ζ.

To prove this we start with the state we have after the edges in ζ have been exchanged. The state of $H^1_{k_3 \times k_2 \times k_1}$ after this is as shown in Figure 7.8

The numerals I and II in Figure 7.8 represent the order of visit (between nodes in the same row) by the $H^1_{k_3 \times k_2 \times k_1}$ cycle starting with 000. The corresponding listing of nodes is as follows.

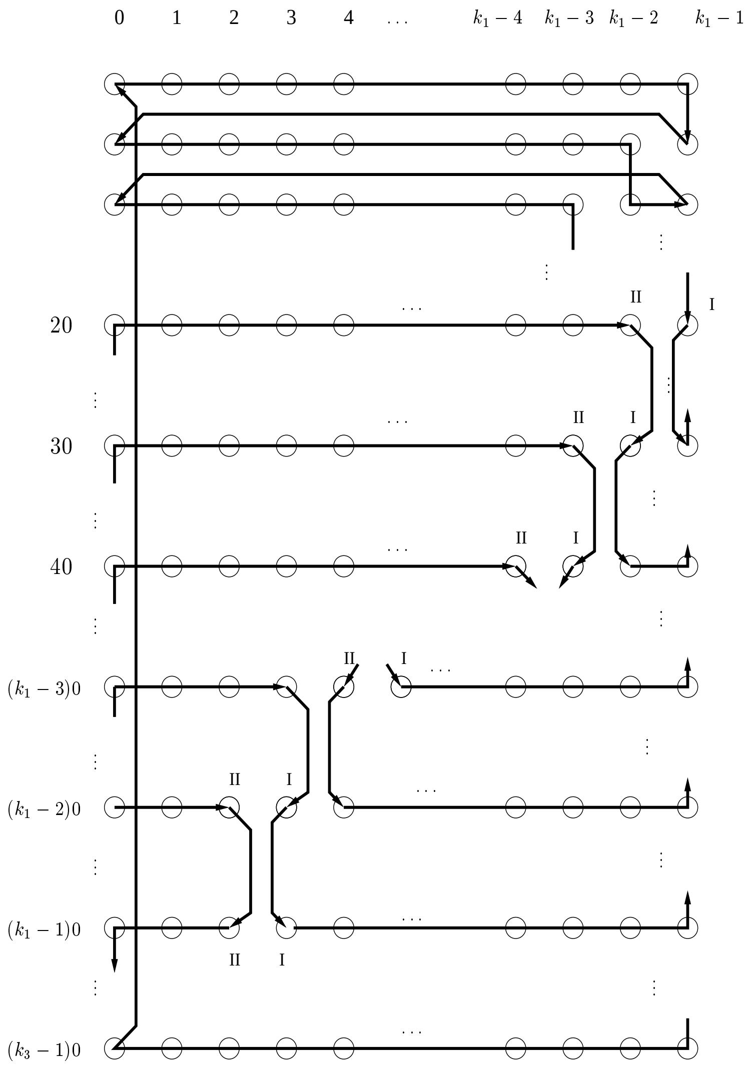


FIGURE 7.8: $H^1_{k_3\times k_2\times k_1}$ graph after exchange of edges in ζ

k_3, k_2	k_1
00	$0, 1, \ldots, k_1-1$
01	$k_1-1, 0, 1, \ldots, k_1-2$

Continued on next page

k_3, k_2	k_1
$\vdots$	$\vdots$
10	$k_1 - 1, 0, 1, \ldots, k_1 - 2$
$\vdots$	$\vdots$
$2(k_2 - 1)$	$0, 1, \ldots, k_1 - 1$
20	$(k_1 - 1)$
30	$(k_1 - 1)$
$3(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
20	$0, 1, \ldots, k_1 - 2$
30	$k_1 - 2$
$\vdots$	$\vdots$
$(k_1 - \gamma - 1)0$	$k_1 - 1, k_1 - 2, \ldots, \gamma + 1$
$(k_1 - \gamma)0$	$k_1 - 1$
$(k_1 - \gamma)(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - \gamma - 1)0$	$0, 1, \ldots, \gamma$
$(k_1 - \gamma)0$	γ
$(k_1 - \gamma + 1)0$	$\gamma, \gamma + 1, \ldots, k_1 - 1$
$(k_1 - \gamma + 1)(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1 - \gamma)0$	$0, 1, \ldots, \gamma - 1$
$(k_1 - \gamma + 1)0$	$\gamma - 1$
$\vdots$	$\vdots$
$(k_1 - 2)0$	3

Continued on next page

k_3, k_2	k_1
$(k_1-1)0$	$3, 4, \dots, k_1-1$
$(k_1-1)(k_2-1)$	$k_1-1, k_1-2, \dots, 0$
$\vdots$	$\vdots$
$(k_1-2)0$	$0, 1, 2$
$(k_1-1)0$	$2, 1, 0$
$(k_1)0$	$0, 1, 2, \dots, k_1-1$
$\vdots$	$\vdots$
$(k_3-1)0$	$k_1-1, k_1-2, \dots, 0$

The state of $H^3_{k_3 \times k_2 \times k_1}$ after the exchange is as shown in Figure 7.9

Note the we need to add the two cycle whose LSB digits are equal to 0 and 1 (Figure 7.9) to the existing cycle in order to make $H^3_{k_3 \times k_2 \times k_1}$ Hamiltonian. The corresponding listing of nodes is as follows.

k_3, k_2	k_1
$00, 10, 20, \dots, (k_2-1), \dots, 0(k_2-1)$	0
$00, 10, 20, \dots, (k_2-1), \dots, 0(k_2-1)$	1
$00, 10, 20, \dots, (k_1-2)0$	2
$(k_1-2)0$	3
$(k_1-2)0, (k_1-1)0, \dots, 0(k_2-1), \dots, (k_1-4)0$	4
$\vdots$	$\vdots$
$50, 60, \dots, 0(k_2-1), 00, 10, \dots, 30$	k_1-3
30	k_1-2
$30, 40, \dots, 0(k_2-1), 00, 10, 20$	k_1-1
$20, 10, 00, 0(k_2-1), \dots, 40$	k_1-2

Continued on next page

k_3, k_2	k_1
40	$k_1 - 3$
$\vdots$	$\vdots$
$(k_1 - 1)0, k_10, \ldots, 0(k_2 - 1)$	2

Now, we will consider what happens when exchange edges as picked from ϕ which belong to $H^3_{k_3 \times k_2 \times k_1}$ are added after edges which belong to $H^1_{k_3 \times k_2 \times k_1}$, as given in ϕ, are removed. This is illustrated in Figure 7.10.

It can be seen that the Hamiltonian property of $H^1_{k_3 \times k_2 \times k_1}$ is maintained. The corresponding listing on nodes that are visited is as follows.

k_3, k_2	k_1
00	0
10	0
11	$0, 1, 2, \ldots, k_1 - 1$
12	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
$2(k_2 - 1)$	$0, 1, \ldots, k_1 - 1$
20	$(k_1 - 1)$
30	$(k_1 - 1)$
$3(k_2 - 1)$	$k_1 - 1, k_1 - 2, \ldots, 0$
$\vdots$	$\vdots$
20	$0, 1$
10	1
00	$1, 2, \ldots, k_1 - 1$
01	$k_1 - 1, 0, 1, \ldots, k_1 - 2$

Continued on next page

k_3, k_2	k_1
$\vdots$	$\vdots$
10	$k_1-1, k_1-2, \ldots, 2$
20	$2, 3, \ldots, k_1-2$
30	k_1-2
$\vdots$	$\vdots$
$(k_1-\gamma-1)0$	$k_1-1, k_1-2, \ldots, \gamma+1$
$(k_1-\gamma)0$	k_1-1
$(k_1-\gamma)(k_2-1)$	$k_1-1, k_1-2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1-\gamma-1)0$	$0, 1, \ldots, \gamma$
$(k_1-\gamma)0$	γ
$(k_1-\gamma+1)0$	$\gamma, \gamma+1, \ldots, k_1-1$
$(k_1-\gamma+1)(k_2-1)$	$k_1-1, k_1-2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1-\gamma)0$	$0, 1, \ldots, \gamma-1$
$(k_1-\gamma+1)0$	$\gamma-1$
$\vdots$	$\vdots$
$(k_1-2)0$	3
$(k_1-1)0$	$3, 4, \ldots, k_1-1$
$(k_1-1)(k_2-1)$	$k_1-1, k_1-2, \ldots, 0$
$\vdots$	$\vdots$
$(k_1-2)0$	$0, 1, 2$
$(k_1-1)0$	$2, 1, 0$
$(k_1)0$	$0, 1, 2, \ldots, k_1-1$

Continued on next page

k_3, k_2	k_1
$\vdots$	$\vdots$
$(k_3-1)0$	$k_1-1, k_1-2, \ldots, 0$

Now consider the state of the $H^3_{k_3 \times k_2 \times k_1}$ shown in Figure 7.11. It can seen that it is now a Hamiltonian cycle.

The corresponding listing of nodes is as follows.

k_3, k_2	k_1
00	0
$00, 0(k_2-1), \ldots, 10$	1
$20, 30, \ldots, (k_1-2)0$	2
$(k_1-2)0$	3
$(k_1-2)0, (k_1-1)0, \ldots, 0(k_2-1), \ldots, (k_1-4)0$	4
$\vdots$	$\vdots$
$50, 60, \ldots, 0(k_2-1), 00, 10, \ldots, 30$	k_1-3
30	k_1-2
$30, 40, \ldots, 0(k_2-1), 00, 10, 20$	k_1-1
$20, 10, 00, 0(k_2-1), \ldots, 40$	k_1-2
40	k_1-3
$\vdots$	$\vdots$
$(k_1-3)0, (k_1-4)0, \ldots, 00, 0(k_2-1), \ldots, (k_1-1)0$	3
$(k_1-1)0k_10, \ldots, 0(k_2-1), 00, 10$	2
10	1
$10, 20, \ldots, 0(k_2-1)$	0

(7.28)

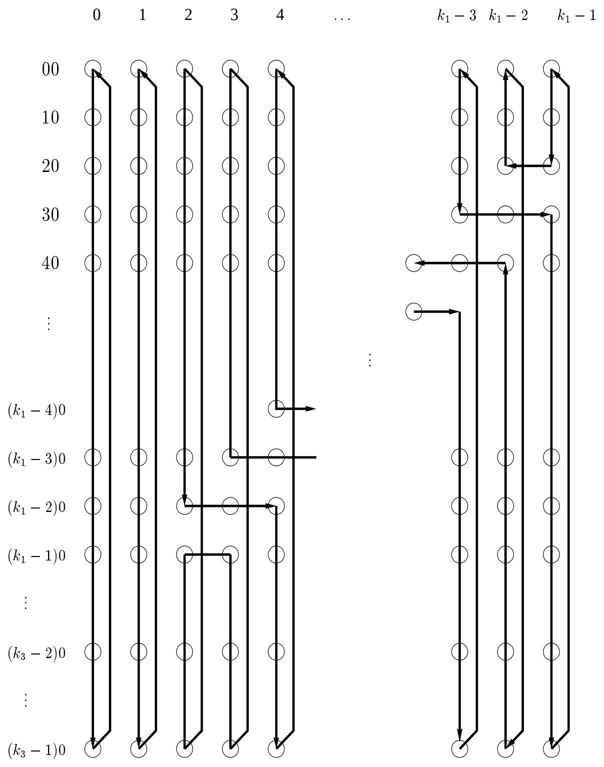


FIGURE 7.9: $H^3_{k_3\times k_2\times k_1}$ graph after exchange of edges in ζ

Exchange Edges when k_is are even

As already mentioned we need to exchange k_1-1 pairs of edges to make $H^3_{k_3\times k_2\times k_1}$ to make it a Hamiltonian cycle. Here k_1 is even the number of pairs to be exchanged is odd.

The set of edges that need to be exchanged between $H^1_{k_3\times k_2\times k_1}$ and $H^3_{k_3\times k_2\times k_1}$ graphs where k is odd is as follows.

When $k_1=2$,

$$\beta = \left\{ \begin{array}{cc} [010,011] & [010,110] \\ [110,111] & [011,111] \end{array} \right\} \tag{7.29}$$

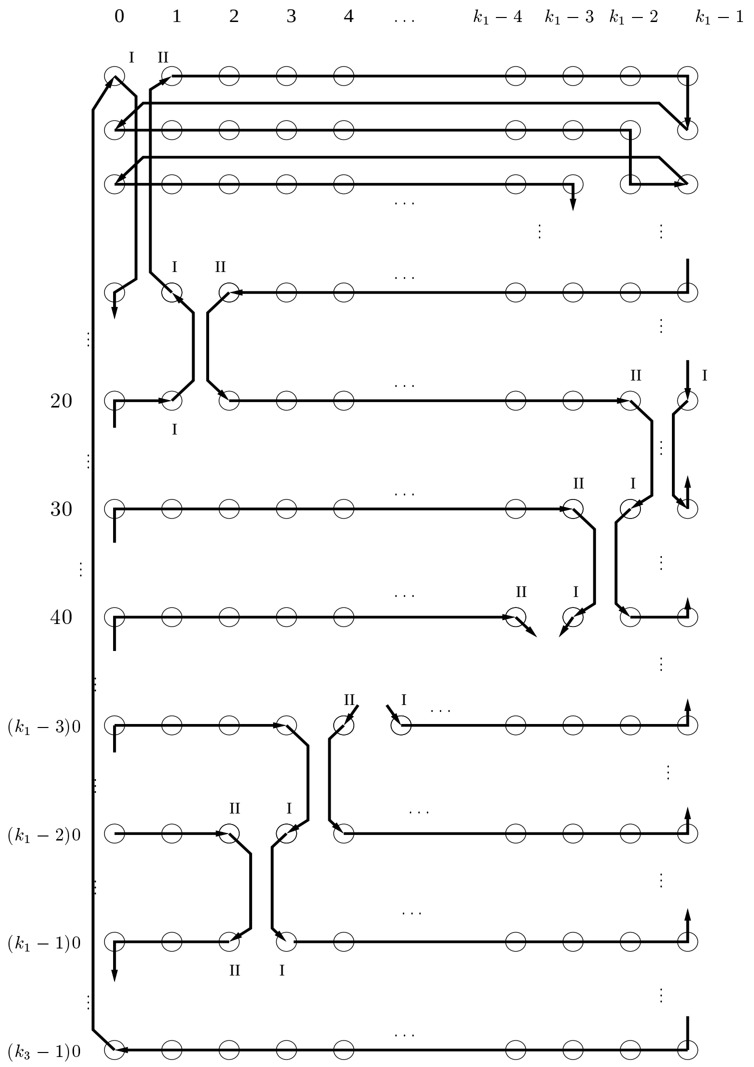


FIGURE 7.10: $H^1_{k_3\times k_2\times k_1}$ graph after exchange of edges in $\zeta + \phi$

The edges in the first column belong to $H^1_{k_3\times k_2\times k_1}$ and those in the second belong to $H^3_{k_3\times k_2\times k_1}$. For $k > 3$, firstly we define a set of exchange edges η as follows

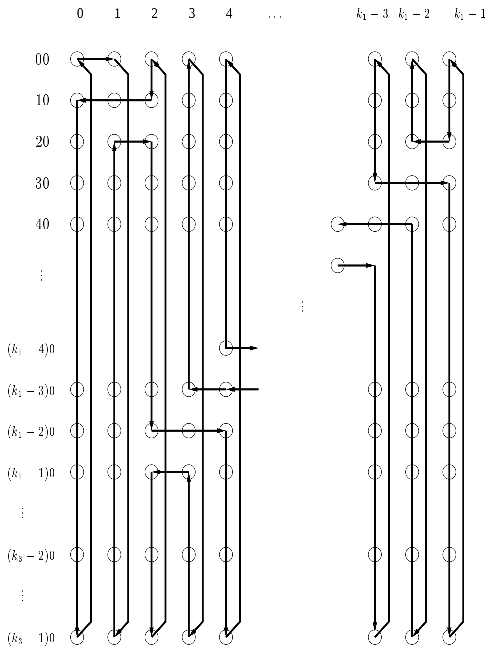


FIGURE 7.11: $H^3_{k_3\times k_2\times k_1}$ graph after exchange of edges in $\zeta+\phi$

When $k_1 \geq 4$,

$$\eta = \{ \begin{array}{cc} [(k_1-j)0j,(k_1-j)0(j+1)] & [(k_1-j)0j,(k_1-j+1)0j] \\ [(k_1-j+1)0j,(k_1-j+1)0(j+1)] & [(k_1-j)0(j+1),(k_1-j+1)0(j+1)] \end{array} \} \tag{7.30}$$

for all j where, $1 \leq j \leq k_1-2$. Here the edges in the first column belong to $H^1_{k_3\times k_2\times k_1}$ and those in the second belong to $H^3_{k_3\times k_2\times k_1}$.

The following theorem is similar to the one in the odd case. The proof is very similar to the one for the odd case. The difference is that we step through 2 values of j in each step of induction to utilize the fact that for every alternate pairs of nodes of the form $x0$ and $(x+2)0$ the direction is always decreasing.

Theorem 7.2.3. *When the set of exchange edges as defined by, η, are exchanged the* *following invariants holds after exchange of each pair of edges*

- $H^1_{k_3 \times k_2 \times k_1}$ *remains a Hamiltonian cycle*
- *Exactly one cycle of length* $k_3 \times k_2$ *nodes is added to* $H^3_{k_3 \times k_2 \times k_1}$.

Thus the complete set of exchange edges is given by

$$\phi = \beta + \eta \tag{7.31}$$

where $+$ indicates the set concatenation operation of the sets represented by φ and ζ. The proof of this is very similar to the one already discussed of the odd case.

CHAPTER 8

GRAY CODES OVER GAUSSIAN INTEGERS

Increasing the number of transistors per chip has led to the design of multiprocessors in a single die. These chips containing multiple processor cores are denoted on-chip multiprocessors (CMPs). Large scale systems such as Piranha [4] and IBM Power 4 [11] combine multiple CMPs to obtain higher performance. With current technology, on-chip networks have to be arranged in two dimensions. In [19], it is shown that Dense Gaussian graphs are a suitable topology for this application. In this chapter, we consider a labeling of nodes in this graph using Gaussian integers.

We show how a Gray code, under a distance metric D, over this labeling corresponds to a Hamiltonian cycle in this graph and also show the existence of two edge disjoint Hamiltonian cycles. This is important to allow efficient all-to-all broadcast and for fault-tolerant communication [13].

The rest of this chapter is organized as follows. First we introduce Gaussian graphs and specify various properties of the graph that we are interested in. We then show an example of edge disjoint Hamiltonian cycles in a G_{3+4i} graph and the corresponding Gray codes. The construction of the Hamiltonian cycles as Gray codes is then shown. Some simple algorithms for generating these codes are then presented.

8.1 Gaussian Graphs

Now, we focus our attention on the relationship between the previously mentioned (in Chapter 2) subfamily of circulant graphs of degree 4 and a new family of graphs whose nodes are labeled by a subset of Gaussian integers

The *Gaussian integers* $\mathbb{Z}[i]$ is the subset of complex number with integer real and imaginary parts, that is

$$\mathbb{Z}[i] = \{x + yi | x, y \in \mathbb{Z}\}$$

$\mathbb{Z}[i]$ is an Euclidean domain and the norm is defined as:

$$\begin{aligned} \mathcal{N}: \ & \mathbb{Z}[i] \to \mathbb{Z}^+ \\ & x + yi \mapsto x^2 + y^2 \end{aligned}$$

Then, for every $\alpha, \pi \in \mathbb{Z}[i]$ with $\pi \neq 0$ there exist $q, r \in \mathbb{Z}[i]$ such that $\alpha = q\pi + r$ with $\mathcal{N}(r) < \mathcal{N}(\pi)$. This means that there exists an Euclidean division algorithm for Gaussian integers analogous to that of rational integers.

Typically, we denote the set of remainders of the division by any integer $N \neq 0$ as $\mathbb{Z}_N$. This set is usually called as the integers modulo N. In an analogous way we can consider $\mathbb{Z}[i]_\pi$, i.e. the Gaussian integers modulo π. It is well known that the number of residue classes modulo a Gaussian integer $\pi \neq 0$ is equal to $\mathcal{N}(\pi)$ and various representations of these residue classes as points in an complex plane are given in [29].

A distance metric was proposed for graphs whose nodes were labeled with the residue classes of a Gaussian integer, π [18]. The Gaussian integer, π, can be of any value and is not necessarily of prime norm.

Definition 8.1.1. *[18] For $\alpha, \beta \in \mathbb{Z}[i]_\pi$, consider $\gamma = x + yi$ in the class of $\beta - \alpha$ with $|x| + |y|$ minimum. The distance D between α and β is*

$$D(\alpha, \beta) = |x| + |y|$$

Theorem 8.1.1. *[18] D defines a distance over the quotient ring $\mathbb{Z}[i]_\pi$.*

A new family of circulant graphs of degree four whose nodes are labeled by Gaussian integers and their adjacency is determined by the distance D as previously defined.

Definition 8.1.2. *[18] Given $\pi = a + bi$ with $gcd(a, b) = 1$. Define the graph $G_\pi = (V, E)$ where:*

1. *$V = \mathbb{Z}[i]_\pi$ is the node set, and*

2. *$E = \{(\alpha, \beta) \in V \times V | D(\alpha, \beta) = 1\}$ is the edge set.*

G_π is the Gaussian graph generated by π.

Theorem 8.1.2. *[18] Let $\pi = a + bi \in \mathbb{Z}[i]$ such that $gcd(a, b) = 1$. We have $C_{\mathcal{N}(\pi)}(a, b)$ and G_π are isomorphic graphs. The graph isomorphism is*

$$\begin{aligned} \Phi \quad : \quad & \mathbb{Z}_{\mathcal{N}(\pi)} \longrightarrow \mathbb{Z}[i]_\pi \\ & j \mapsto x + yi \mod \pi \end{aligned}$$

where $j \equiv ax + by \mod \mathcal{N}(\pi)$.

Proof. [18] We have to prove that Φ is a bijection that preserves the distances. Firstly, note that the set of solutions of the Diophantine equation

$$aX + bY = s(a^2 + b^2)$$

is $\{(X, Y) = (sa + bt, sb - at) | t \in \mathbb{Z}\}$.
Φ is a mapping. Let $j \in \mathbb{Z}$ such that $j \equiv ax + by \mod N$ and $h \equiv j \mod N$. We have to prove that $\Phi(j) \equiv \Phi(h) \mod \pi$. Suppose that $h \equiv ax' + by' \mod N$. Then, assuming the hypothesis, we have that $a(x-x')+b(y-y') \equiv 0 \mod N$, that is, there exists $s \in \mathbb{Z}$ verifying $a(x-x')+b(y-y') = sN$. Now, if $a(x-x')+b(y-y') = s(a^2+b^2)$

then $(x - x') + i(y - y') \equiv 0 \mod \pi$. We have that $(x - x') + i(y - y') = (sa + bt) + (sb - at)i = s(a + bi) + t(b - ai) = s(a + bi) - it(a + bi) = (s - ti)(a + bi)$. which concludes this part of the proof, since $s - ti \in \mathcal{Z}[i]$.

Φ is injective. We are going to prove that $\Phi(j) \equiv \Phi(h) \mod \pi$, with $j, h \in \mathbb{Z}_N$, implies that $j \equiv h \mod N$. Let $\Phi(j) = x + yi$, $\Phi(h) = x' + y'$. Then, $(x - x') + (y - y')i = \alpha\pi$ with $\alpha \in \mathcal{Z}[i]$. Let $\alpha = \alpha_1 + \alpha_2 i$. Thus, $(x - x') + (y - y')i = \alpha\pi = (\alpha_1 a - \alpha_2 b) + (\alpha_1 b + \alpha_2 a)i$, so we get

$$x - x' = \alpha_1 a - \alpha_2 b) \Rightarrow a(x - x') = \alpha_1 a^2 - \alpha_2 ab$$
$$y - y' = \alpha_1 b + \alpha_2 a) \Rightarrow b(y - y') = \alpha_1 b^2 - \alpha_2 ab$$

Now, $a(x - x') + b(y - y') = \alpha_1(a^2 + b^2)$, with $\alpha_1 \in \mathbb{Z}$, that is, $j \equiv h \mod N$. Also, it can easily proved that Φ is surjective. □

The proof of the above theorem is included for the sake of completeness. With the above definitions we see that we can label this graph using Gaussian integers. In the following section we give an example where we show a Gaussian graph satisfying the aforementioned properties and a set of edge disjoint Hamiltonian cycles.

Example 11. *Figure 8.1 shows the graph G_{3+4i} and how the nodes are labeled. The wrap-around edges are valid in the graph because the residue class of the difference in the labels of the nodes they connect is 1 under $\mathcal{D}$. The connection pattern of these graphs will be explained in detail later. The dotted and thick shaded edges in the graph correspond to the edge disjoint Hamiltonian cycles in this graph.*

As can be seen from the above example we can now define these edge disjoint Hamiltonian cycles as two independent Gray codes over the set of Gaussian integers $\mathbb{Z}[i]_\pi$.

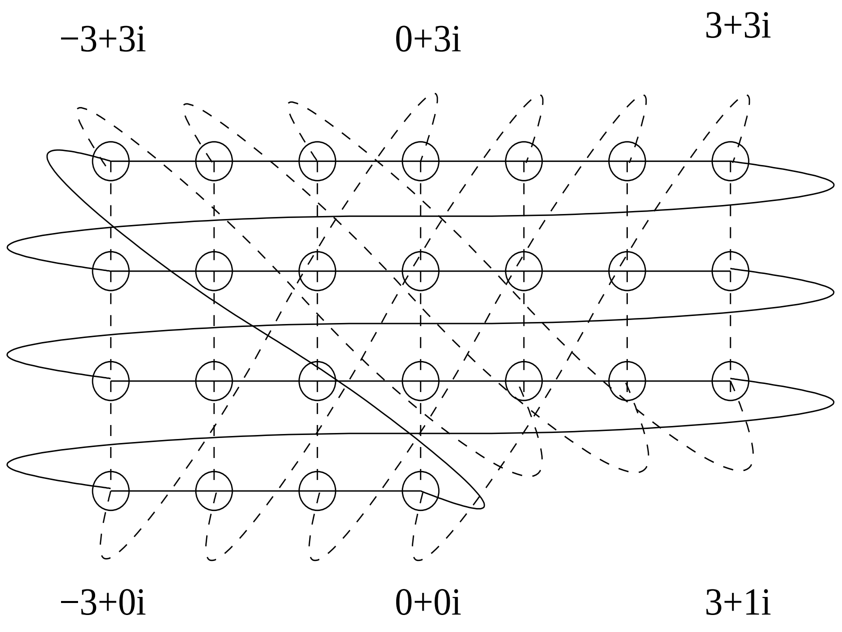


FIGURE 8.1: Edge Disjoint Hamiltonian cycles in G_{3+4i}

In this Gray listing, any two consecutive labels, $\alpha, \beta \in \mathbb{Z}[i]_\pi$, differ by distance 1 as defined using the distance metric D. Let the first Gray code (denoted by thick lines) be denoted as G^1_{3+4i} and the second by G^2_{3+4i}. Then these Gray codes are defined as shown in Table 8.1.

In this paper, we are only interested in Gaussian integers of the form $\gamma = a + ib$ where $\gcd(a, b) = 1$, $b > a \geq 1$. When $a = 0$, this graph is isomorphic to the two dimensional torus graph [12]. Hence we can use the construction in [13] to obtain two edge disjoint Hamiltonian cycles.

G^1_{3+4i}	G^2_{3+4i}
$0+0i$	$0+0i$
$-3+3i$	$0+1i$
$-2+3i$	$0+2i$
$-1+3i$	$0+3i$
$0+3i$	$-3+0i$
$1+3i$	$-3+1i$
$2+3i$	$-3+2i$
$3+3i$	$-3+3i$
$-3+2i$	$1+1i$
$-2+2i$	$1+2i$
$-1+2i$	$1+3i$
$0+2i$	$-2+0i$
$1+2i$	$-2+1i$
$2+2i$	$-2+2i$
$3+2i$	$-2+3i$
$-3+1i$	$2+1i$
$-2+1i$	$-2+2i$
$-1+1i$	$-2+3i$
$0+1i$	$-1+0i$
$1+1i$	$-1+1i$
$2+1i$	$-1+2i$
$3+1i$	$-1+3i$
$-3+0i$	$3+1i$
$-2+0i$	$3+2i$
$-1+0i$	$3+3i$
$0+1i$	$0+0i$

TABLE 8.1: Gray codes over G_{3+4i}

8.2 Structure of a Gaussian graph

In this section we will briefly introduce the representation used for residue classes of Gaussian integers and the connectivity patterns of the graph, $\mathcal{G}_\pi$ under the distance metric, $\mathcal{D}$.

We assume that $\pi = a + bi$ with $\gcd(a, b) = 1$ and $a \geq 0, b > a$. We are interested in labeling the nodes of the Gaussian graph, $\mathcal{G}_\pi$, using a complete residue system modulo π. There are different equivalent representations of the complete residue system modulo π [29]. In this paper we use a representation referred to as Utah representation [29].

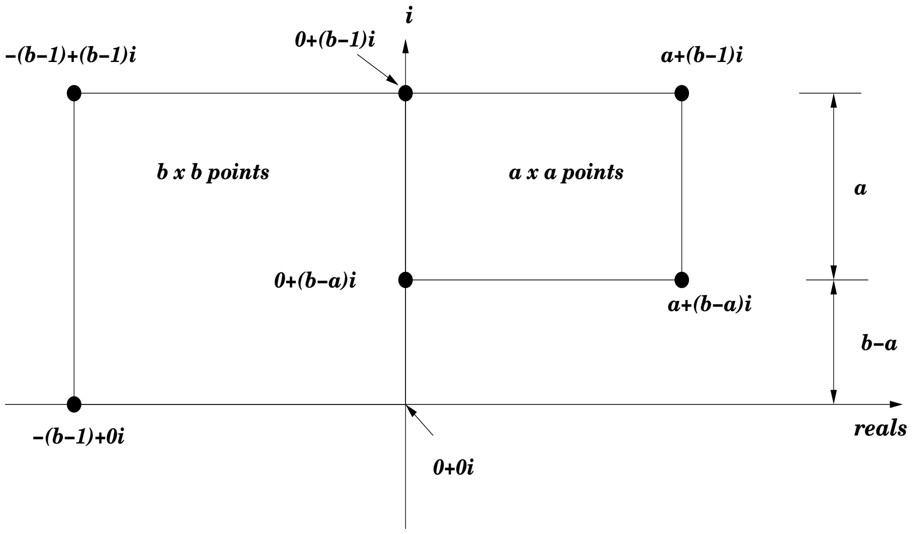


FIGURE 8.2: Residue classes in $G_{\pi=a+bi}$

Figure 8.2 shows the labeling scheme we propose for the $a^2 + b^2$ nodes of $\mathcal{G}_\pi$. $\mathcal{G}_\pi$ is a degree 4 regular graph. In general each node is connected to two nodes whose labels differ by $\pm i$ **mod** π and two nodes whose labels differ by ± 1 **mod** π. For the nodes which do not lie in the border of either squares the connection pattern is simple and each node connects to four nodes in its vicinity (Figure 8.1). For the nodes in the border two connections (one direction in real and one direction in imaginary) are simple and re to two nodes in the vicinity. The other two links are wrap around links, one each in the remaining directions in each dimension. The general pattern of connectivity for these links is as shown in Figures 8.3 and 8.4.

Definition 8.2.1. *A link in $\mathcal{G}_\pi$ is said lie in the real dimension if and only if the two nodes it connects, α_1 and α_2 satisfy the following relation.*

$$\alpha_1 - \alpha_2 \equiv \pm 1 \ \textbf{\textit{mod}}\ \pi$$

Definition 8.2.2. *A link in $\mathcal{G}_\pi$ is said lie in the imaginary dimension if and only if the two nodes it connects, α_1 and α_2 satisfy the following relation.*

$$\alpha_1 - \alpha_2 \equiv \pm i \ \textbf{\textit{mod}}\ \pi$$

From the above definitions it can be seen that the horizontal edges in $\mathcal{G}_\pi$ lie in the real dimension and vertical edges lie in the imaginary dimension. Figures 8.3 and 8.4 show the wrap-around edges in the real dimension and imaginary dimensions respectively. Note that each node in the border of $\mathcal{G}_\pi$ has one wrap-around link in Figure 8.3 and one in Figure 8.4 respectively. This shows the connectivity structure of $\mathcal{G}_\pi$. We show that these wrap-around links are adjacent under the distance metric, $\mathcal{D}$ and they lie in the *real* dimension. Lemmas 8.2.1 and 8.2.2 prove that nodes connected by wrap-around edges in Figure 8.3 are adjacent.

Lemma 8.2.1. *Let $\alpha_1 = a_1 + ib_1$ and $\alpha_1 = a_2 + ib_2$ be any two Gaussian integers in residue class of $\pi = a + ib$ where $(b-1) \geq b_1 \geq (b-a)$ and $b_1 - b_2 = (b-a)$ and*

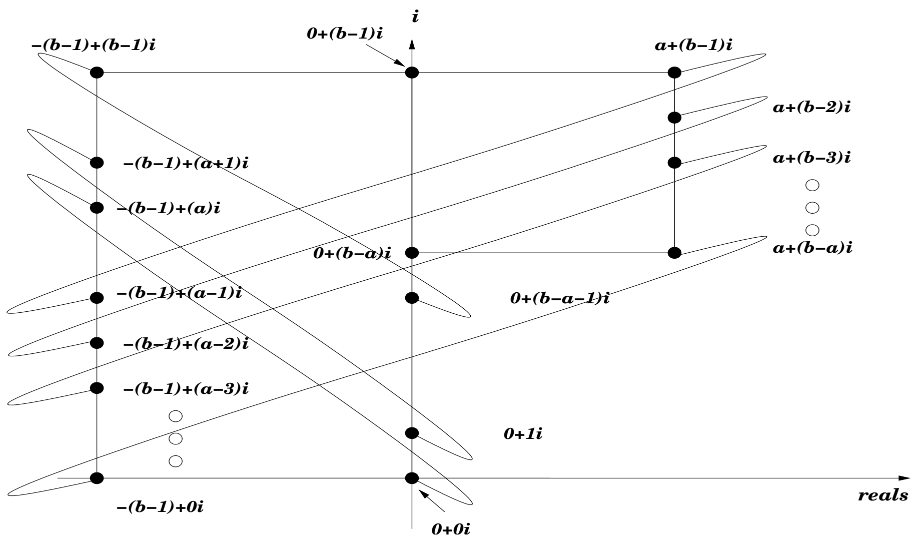


FIGURE 8.3: Wrap-around where $\alpha_1 - \alpha_2 \equiv 1$ **mod** π in $\mathcal{G}_{\pi=a+bi}$

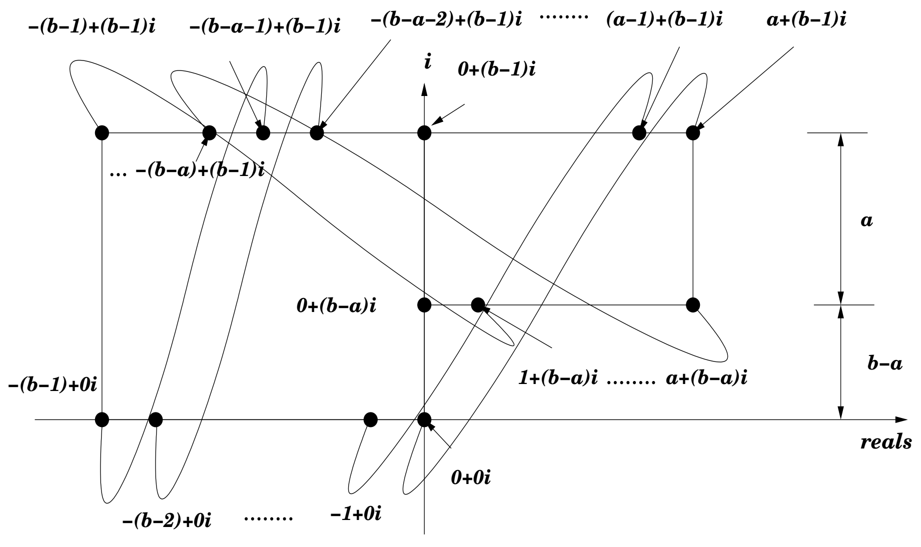


FIGURE 8.4: Wrap-around where $\alpha_1 - \alpha_2 \equiv i$ **mod** π in $\mathcal{G}_{\pi=a+bi}$

$a_1 = a$ and $a_2 = -(b-1)$. Then, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.

Proof. The Gaussian integers, α_1 and α_2, under the given assumptions can be expressed as follows:

$$\begin{aligned} \alpha_1 &= a + i(b-k) \\ \alpha_2 &= -(b-1) + i((b-k) - (b-a)) \\ &= -(b-1) + i(a-k) \end{aligned}$$

where $1 \leq k \leq a$. Now,

$$\alpha_1 - \alpha_2 = (a+b-1) + i(b-a) \tag{8.1}$$

To find the residue class of $\alpha_1 - \alpha_2$ we first find the quotient of $(\alpha_1 - \alpha_2)/\pi$ and then subtract it from $\alpha_1 - \alpha_2$ to find the remainder.

$$\begin{aligned} q &= \frac{\alpha_1 - \alpha_2}{\pi} = \frac{(a+b-1) + i(b-a)}{a+ib} && (8.2) \\ &= \left(\frac{(a+b-1) + i(b-a)}{a+ib}\right)\left(\frac{a-ib}{a-ib}\right) && (8.3) \\ q &= \left[\frac{a(a+b-1) + b(b-a)}{a^2+b^2}\right] - i\left[\frac{b(a+b-1) - a(b-a)}{a^2+b^2}\right] && (8.4) \end{aligned}$$

In Equation 8.4, [x] denotes rounding to the closest integer. Now we will show that the q is equal to $1 - i$.

First consider the real part of the Gaussian integer in Equation 8.4. Under the specified rounding operation, the equation will evaluate to 1 if and only if $\frac{1}{2}(a^2+b^2) \leq a(a+b-1) + b(b-a) \leq (a^2+b^2)$. We will now prove that this is the case.

Assuming to the contrary that $a(a+b-1) + b(b-a) > (a^2+b^2)$ is true, we get

$$\Rightarrow a(a+b-1)+b(b-a) \quad > \quad (a^2+b^2) \tag{8.5}$$

$$\Rightarrow a^2+ab-a+b^2-ba \quad > \quad (a^2+b^2) \tag{8.6}$$

$$\Rightarrow -a > 1 \quad \Rightarrow \quad a < -1 \tag{8.7}$$

Thus from Equation 8.7 we see that our assumption is true only for $a < -1$. In our case we have $a \geq 1$. Therefore $a(a+b-1)+b(b-a) \leq (a^2+b^2)$.
To prove that $\frac{1}{2}(a^2+b^2) \leq a(a+b-1)+b(b-a)$ we start by assuming the contrary. Thus, we get

$$\Rightarrow a(a+b-1)+b(b-a) \quad < \quad \frac{1}{2}(a^2+b^2) \tag{8.8}$$

$$\Rightarrow 2a^2+2ab-2a+2b^2-2ba \quad < \quad (a^2+b^2) \tag{8.9}$$

$$\Rightarrow (a^2+b^2) < 2a \tag{8.10}$$

Clearly from Equation 8.10, this is again impossible when $a \geq 1$ and $b > a$. Therefore, the real part of q rounds to 1.
Following an approach similar to that for real part, we assume that $b(a+b-1) - a(b-a) > (a^2+b^2)$, we get

$$\Rightarrow b(a+b-1)-a(b-a) \quad > \quad (a^2+b^2) \tag{8.11}$$

$$\Rightarrow b^2+ab-b+a^2-ba \quad > \quad (a^2+b^2) \tag{8.12}$$

$$\Rightarrow -b > 1 \quad \Rightarrow \quad b < -1 \tag{8.13}$$

Again this is not true as we have $b > 1$.
Now to show that $b(a+b-1)-a(b-a) \geq \frac{1}{2}(a^2+b^2)$, we start by assuming that this is not true.

$$\Rightarrow b(a+b-1) - a(b-a) \quad < \quad \frac{1}{2}(a^2+b^2) \tag{8.14}$$

$$\Rightarrow 2b^2 + 2ab - 2b + 2a^2 - 2ba \quad < \quad (a^2+b^2) \tag{8.15}$$

$$\Rightarrow (a^2+b^2) < 2b \tag{8.16}$$

Equation 8.16 is not true for $b > 1$. Thus the imaginary part also rounds to 1. Thus, $q = 1 - i$. Now, let us find the residue class of $\alpha_1 - \alpha_2$.

$$\begin{aligned} r &= (a+b-1) + i(b-a) - [(a+ib)(1-i)] && (8.17)\\ &= (a+b-1) + i(b-a) - [(a+b) + i(b-a)] && (8.18)\\ &= -1 && (8.19) \end{aligned}$$

Now to see the minimality of this rounding technique, firstly observe that q remains same when rounding to next highest integer for Equation 8.4. If we are rounding to the next lowest integer we get $q = 0$ (in Equation 8.4).

$$r = (a+b-1) + i(b-a) \tag{8.20}$$

and so,

$$\begin{aligned} \mathcal{D}(\alpha_1, \alpha_2) &= |(a+b-1)| + |(b-a)| && (8.21)\\ &= 2b-1 && (8.22) \end{aligned}$$

Obviously this is greater than the distance when $r = 1$ for $b > 1$. □

Lemma 8.2.2. *Let $\alpha_1 = a_1 + ib_1$ and $\alpha_1 = a_1 + ib_1$ be any two Gaussian integers in the residue class of $\pi = a + ib$ where $0 \leq b_1 \leq (a-1)$ and $b_2 = b_1 + a$ and $a_1 = 0$ and $a_2 = -(b-1)$. Then, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.*

Proof. The Gaussian integers, α_1 and α_2, under the given assumptions can be expressed as follows

$$\begin{aligned}\alpha_1 &= 0 + ik \\ \alpha_2 &= -(b-1) + i(k+a)\end{aligned}$$

where $0 \leq k \leq (a-1)$. Now,

$$\alpha_1 - \alpha_2 = (b-1) - ia \tag{8.23}$$

To find the residue class of $\alpha_1 - \alpha_2$ we first find the quotient of $(\alpha_1 - \alpha_2)/\pi$ and then subtract it from $\alpha_1 - \alpha_2$ to find the remainder.

$$\begin{aligned} q = \frac{\alpha_1 - \alpha_2}{\pi} &= \frac{(b-1) - ia}{a + ib} \qquad (8.24) \\ &= \left(\frac{(b-1) - ia}{a + ib}\right)\left(\frac{a - ib}{a - ib}\right) \qquad (8.25) \\ q &= \left[\frac{a(b-1) - ab)}{a^2 + b^2}\right] - i\left[\frac{b(b-1) + a^2}{a^2 + b^2}\right] \qquad (8.26) \end{aligned}$$

In Equation 8.26, $[x]$ denotes rounding to the closest integer. Now we will show that the q is equal to $-i$.

First consider the real part of the Gaussian integer in Equation 8.26. Under the specified rounding operation, the equation will evaluate to 0 if and only if $a(b-1) - ab < \frac{1}{2}(a^2 + b^2)$. We will now prove that this is the case.

Assuming that $a(b-1) - ab \geq \frac{1}{2}(a^2 + b^2)$ is true, we get

$$\begin{aligned} \Rightarrow a(b-1) - ab &\geq (a^2 + b^2) \qquad (8.27) \\ \Rightarrow -a &\geq (a^2 + b^2) \qquad (8.28) \end{aligned}$$

Clearly Equation 8.28 is not true given $a \geq 1$. Therefore $a(b-1)+a^2 < \frac{1}{2}(a^2+b^2)$ and the real part of q under this rounding scheme will evaluate to 0.

Following an approach similar to one in Lemma 8.2.1, we assume that $b(b-1)+a^2 > (a^2+b^2)$. We get

$$\Rightarrow b(b-1)+a^2 > (a^2+b^2) \tag{8.29}$$

$$\Rightarrow b^2-b+a^2 > (a^2+b^2) \tag{8.30}$$

$$\Rightarrow -b > 1 \Rightarrow b < -1 \tag{8.31}$$

Thus, our assumption is not true as we have $b > 1$.

Now to show that $b(b-1)+a^2 \geq \frac{1}{2}(a^2+b^2)$, we start by assuming that this is not true.

$$\Rightarrow b(b-1)+a^2 < \frac{1}{2}(a^2+b^2) \tag{8.32}$$

$$\Rightarrow 2b^2-2b+2a^2 < (a^2+b^2) \tag{8.33}$$

$$\Rightarrow (a^2+b^2) < 2b \tag{8.34}$$

Equation 8.34 is not true for $b > 1$. Thus the imaginary part rounds to 1. Thus, $q = -i$. Now, let us find the residue class of $\alpha_1 - \alpha_2$.

$$r = (b-1) - ia - [(a+ib)(-i)] \tag{8.35}$$

$$= (b-1) - ia - [b-ia] \tag{8.36}$$

$$= -1 \tag{8.37}$$

Now we prove the minimality of this rounding scheme. Firstly if we apply rounding to next highest integer is used then, $q = 1-i$. For this value if we calculate the value of r, we get

$$\begin{aligned} r &= (b-1) - ia - [(a+b) + i(b-1)] \\ &= -3a + b + 1 \end{aligned}$$

and so,

$$\begin{aligned} \mathcal{D}(\alpha_1, \alpha_2) &= |-3a| + |b| + 1 \qquad (8.38) \\ &= 3a + b + 1 \qquad (8.39) \end{aligned}$$

Obviously this is greater than the distance when $r = 1$ for $b > 1$.

Similarly considering the case of rounding to next lowest integer we get $q = 0$ and $r = (b-1) - ai$ in this case. Thus distance in this case is

$$\begin{aligned} \mathcal{D}(\alpha_1, \alpha_2) &= |(b-1)| + |-a| \qquad (8.40) \\ &= a + b - 1 \qquad (8.41) \end{aligned}$$

Again this is greater than 1 when $a \geq 1$ and $b > a$. Thus rounding-to-closest-integer is the rounding scheme where $\mathcal{D}$ is minimized. $\square$

The following two Lemmas are for nodes connected by wrap-around links shown in Figure 8.4. The proofs of these Lemmas are similar to those of Lemmas 8.2.1 and 8.2.2 and are hence omitted. The interesting aspect of these wrap-around edges is that they lie in the *imaginary* dimension.

Lemma 8.2.3. *Let $\alpha_1 = a_1 + ib_1$ and $\alpha_1 = a_1 + ib_1$ be any two Gaussian integers in the residue class of $\gamma = a + ib$ where $-(b-1) \leq a_1 \leq 0$ and $a_1 - a_2 = a$ and $b_1 = 0$ and $a_2 = (b-1)$. Then, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.*

Lemma 8.2.4. *Let $\alpha_1 = a_1 + ib_1$ and $\alpha_1 = a_1 + ib_1$ be any two Gaussian integers in the residue class of $\gamma = a + ib$ where $0 < a_1 \leq a$ and $a_1 - a_2 = b$ and $b_1 = (b - a)$ and $a_2 = (b - 1)$. Then, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.*

8.3 Edge Disjoint Hamiltonian Cycles

In this section we show the construction of two edge disjoint Hamiltonian cycles in a Gaussian graph, $\mathcal{G}_\pi$, where the nodes are labeled using the residue classes modulo $\pi = a + ib$ where a and b are as already assumed.

As already defined, a Gray code, $\mathcal{A}_\pi$, is a listing of all residue classes modulo a Gaussian integer, $\pi = a+ib$, where $\gcd(a, b) = 1$ and $b > a \geq 1$. The special property of this listing is that for any pair of consecutive vectors, α and β, $\mathcal{D}(\alpha, \beta) = 1$. Under these conditions the first Gray code, $\mathcal{A}^1_\pi$, is defined as follows

$\mathcal{A}^1_\pi$			
$-(b-1)$	$+$	$0i$	
$-(b-2)$	$+$	$0i$	
$\vdots$	$\vdots$	$\vdots$	
(-1)	$+$	$0i$	
0	$+$	$0i$	
$-(b-1)$	$+$	$(a \textbf{ mod } b)i$	
$-(b-2)$	$+$	$(a \textbf{ mod } b)i$	
$\vdots$	$\vdots$	$\vdots$	
(-1)	$+$	$(a \textbf{ mod } b)i$	
0	$+$	$(a \textbf{ mod } b)i$	if $(a \textbf{ mod } b) < (b-a)$
1	$+$	$(a \textbf{ mod } b)i$	

Continued on next page

	$\mathcal{A}^1_\pi$		
$\vdots$	$\vdots$	$\vdots$	
a	$+$	$(a \textbf{ mod } b)i$	if $(a \textbf{ mod } b) \geq (b-a)$
$-(b-1)$	$+$	$(2a \textbf{ mod } b)i$	
$-(b-2)$	$+$	$(2a \textbf{ mod } b)i$	
$\vdots$	$\vdots$	$\vdots$	
(-1)	$+$	$(2a \textbf{ mod } b)i$	
0	$+$	$(2a \textbf{ mod } b)i$	if $(2a \textbf{ mod } b) < (b-a)$
1	$+$	$(2a \textbf{ mod } b)i$	
$\vdots$	$\vdots$	$\vdots$	
a	$+$	$(2a \textbf{ mod } b)i$	if $(2a \textbf{ mod } b) \geq (b-a)$
$\vdots$	$\vdots$	$\vdots$	
$-(b-1)$	$+$	$((b-1)a \textbf{ mod } b)i$	
$-(b-2)$	$+$	$((b-1)a \textbf{ mod } b)i$	
$\vdots$	$\vdots$	$\vdots$	
(-1)	$+$	$((b-1)a \textbf{ mod } b)i$	
0	$+$	$((b-1)a \textbf{ mod } b)i$	if $((b-1)a \textbf{ mod } b) < (b-a)$
1	$+$	$((b-1)a \textbf{ mod } b)i$	
$\vdots$	$\vdots$	$\vdots$	
a	$+$	$((b-1)a \textbf{ mod } b)i$	if $((b-1)a \textbf{ mod } b) \geq (b-a)$

Lemma 8.3.1. *Given $a \geq 0$ and $b > a$, where $gcd(a,b) = 1$. The least multiple of a divisible by b is ba. Also, $\{0a, 1a, \ldots, (b-2)a, (b-1)a\}$ are all unique modulo b.*

Theorem 8.3.1. *The Gray code defined by $\mathcal{A}^1_\pi$ forms a Hamiltonian cycle in the $\mathcal{G}_\pi$ graph.*

Proof. To show that $\mathcal{A}_\pi^1$ forms a Hamiltonian cycle we must show that each node in $\mathcal{G}_\pi$ is visited exactly once and all nodes are visited and between every two consecutive nodes, α_1, α_2, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.
By Lemma 8.3.1, we see that each row is visited exactly once. All node labels are visited exactly once within each row (Equation 8.3) . Thus, each node in a row is visited exactly once and each row is visited exactly once.
We will now show that the distance between any pair of consecutive nodes, α_1, α_2 in $\mathcal{A}_\pi^1$ is 1.

1. Firstly, we consider nodes that differ only in the real part. That is, let $\alpha_1 = \gamma_1 + i\sigma_1$ and $\alpha_2 = \gamma_2 + i\sigma_2$ be such that $\sigma_1 = \sigma_2$. In this case, $\beta = \alpha_1 - \alpha_2 = \pm 1$. Since $\beta = \pm 1$ is a unit in $\mathcal{Z}[i]_\pi$ this is also an equivalence class for β. For any two nodes $\alpha_1 \neq \alpha_2$, β is the equivalence class for which $\mathcal{D}$ is minimum. Thus, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.

2. Consider the case when both the real and imaginary parts are not equal. There are two sub cases we need to consider.

   (a) Let $\alpha_1 = \gamma_1 + i\sigma_1$ and $\alpha_2 = \gamma_2 + i\sigma_2$ be such that $\sigma_1 = (p-1)a \textbf{ mod } b$ and $\sigma_2 = pa \textbf{ mod } b$. If $\sigma_1 \geq (b-a)$ then $\sigma_1 > \sigma_2$ and $\gamma_1 = a$, $\gamma_2 = -(b-1)$. The two nodes thus satisfy the conditions of Lemma 8.2.1 and thus $\mathcal{D}(\alpha_1, \alpha_2) = 1$.

   (b) Let $\alpha_1 = \gamma_1 + i\sigma_1$ and $\alpha_2 = \gamma_2 + i\sigma_2$ be such that $\sigma_1 = (p-1)a \textbf{ mod } b$ and $\sigma_2 = pa \textbf{ mod } b$. If $\sigma_1 \leq (a-1)$ then $\sigma_1 < \sigma_2$ and $\gamma_1 = 0$ and $\gamma_2 = -(b-1)$. The two nodes thus satisfy the conditions for Lemma 8.2.2. Therefore, $\mathcal{D}(\alpha_1, \alpha_2) = 1$.

Also, the first and last nodes in $\mathcal{A}_\pi^1$ satisfy the conditions for Lemma 8.2.1 and hence are at a distance 1. This shows the cyclic property of $\mathcal{A}_\pi^1$. Note that links considered in case 1 correspond to those in a row. While those in case 2 correspond to links going from one row to another. This proves that G_π^1 forms a Hamiltonian cycle in $\mathcal{A}_\pi$.

□

Note that the wrap around edges and horizontal edges used by $\mathcal{A}_\pi^1$ lie in real dimension. No edges which lie in the imaginary dimension are used. The second Gray code, $\mathcal{A}_\pi^2$, can be defined as follows

	$\mathcal{A}_\pi^2$		
a	$+$	$(b-1)i$	
a	$+$	$(b-2)i$	
$\vdots$	$\vdots$	$\vdots$	
a	$+$	$(b-a+1)i$	
a	$+$	$(b-1)i$	
$\mathcal{M}(b)$	$+$	$(b-1)i$	
$\mathcal{M}(b)$	$+$	$(b-2)i$	
$\vdots$	$\vdots$	$\vdots$	
$\mathcal{M}(b)$	$+$	$(b-a+1)i$	
$\mathcal{M}(b)$	$+$	$(b-a)i$	if $\mathcal{M}(b) > 0$
$\mathcal{M}(b)$	$+$	$(a-1)i$	
$\vdots$	$\vdots$	$\vdots$	
$-\mathcal{M}(b)$	$+$	$(0)i$	if $\mathcal{M}(b) \leq 0$
$\mathcal{M}(2b)$	$+$	$(b-1)i$	

Continued on next page

	$\mathcal{A}_\pi^2$		
$\mathcal{M}(2b)$	$+$	$(b-2)i$	
$\vdots$	$\vdots$	$\vdots$	
$\mathcal{M}(2b)$	$+$	$(b-a+1)i$	
$\mathcal{M}(2b)$	$+$	$(b-a)i$	if $\mathcal{M}(2b) > 0$
$\mathcal{M}(2b)$	$+$	$(a-1)i$	
$\vdots$	$\vdots$	$\vdots$	
$\mathcal{M}(2b)$	$+$	$(0)i$	if $\mathcal{M}(2b) \leq 0$
$\vdots$	$\vdots$	$\vdots$	
$\mathcal{M}((a+b-1)b)$	$+$	$((b-1)i$	
$\mathcal{M}((a+b-1)b)$	$+$	$((b-2)i$	
$\vdots$	$\vdots$	$\vdots$	
$\mathcal{M}((a+b-1)b)$	$+$	$(b-a+1)i$	
$\mathcal{M}((a+b-1)b)$	$+$	$(b-a)i$	if $\mathcal{M}((a+b-1)b) > 0$
$\mathcal{M}((a+b-1)b)$	$+$	$(a-1)i$	
$\vdots$	$\vdots$	$\vdots$	
$\mathcal{M}((a+b-1)b)$	$+$	$(0)i$	if $(-1 \times \mathcal{M}((a+b-1)b)) \leq 0$

where $\mathcal{M}(p) = -1 \times [(p \textbf{ mod } (a+b)) - a]$.

Theorem 8.3.2. *The Gray code defined by $\mathcal{A}_\pi^2$ forms a Hamiltonian cycle in the $\mathcal{G}_\pi$ graph.*

The proof of Theorem 8.3.2 is similar to that of Theorem 8.3.1. Note that $\mathcal{A}_\pi^2$ uses only edges in the imaginary dimension.

Theorem 8.3.3. *The Hamiltonian cycles defined by Gray codes, $\mathcal{A}_\pi^1$ and $\mathcal{A}_\pi^2$ are edge disjoint.*

Proof. We prove this fact by contradiction. If $(A)^1_\pi$ and $\mathcal{A}^2_\pi$ are not edge disjoint, then there is a pair of consecutive nodes (i.e., an edge) in one listing which is also consecutive in the other. Let these nodes be $\zeta_1 = \alpha_1 + i\beta_1$ and $\zeta_2 = \alpha_2 + i\beta_2$ where $\zeta_1, \zeta_2 \in \mathbb{Z}[i]_\pi$. From Theorems 8.3.1 and 8.3.2 it can be seen that $(A)^1_\pi$ uses only edges in the real dimension, this implies that

$$\zeta_1 - \zeta_2 \equiv \pm 1 \mod \pi. \tag{8.42}$$

Also, $(A)^2_\pi$ uses only edges in the imaginary dimension, this implies that

$$\zeta_1 - \zeta_2 \equiv \pm i \mod \pi. \tag{8.43}$$

The above equations will have to then imply that

$$\pm i \equiv \eta = \zeta_1 - \zeta_2 \equiv \pm 1 \tag{8.44}$$

That is there exists a $\eta \in \mathcal{Z}[i]_\pi$ that is equivalent to both $\pm i$ and ± 1. This would then imply that $\pm 1 \equiv \pm i$. This is clearly a contradiction because $\pm 1, \pm i$ are units in $\mathcal{Z}[i]_\pi$ and are not equivalent. Therefore, under the given construction, $\mathcal{A}^1_\pi$ and $\mathcal{A}^2_\pi$ are edge disjoint. $\square$

We now have the nodes labels for the edge disjoint Hamiltonian cycles, $\mathcal{A}^1_\pi$ and $\mathcal{A}^1_\pi$, in $\mathcal{G}_{\pi=a+ib}$. Now, we can obtain the corresponding the node labels of the edge disjoint Hamiltonian cycles in the isomorphic $\mathcal{C}_N(j_1, j_2)$ graph, where $j_1 = a, j_2 = b$ and $\mathcal{N} = j_1^2 + j_1^2$ by finding a j such that $j \equiv ax + by$ **mod** $\mathcal{N}$ is satisfied for a node label $x + iy$ in either $\mathcal{A}^1_\pi$ or $\mathcal{A}^2_\pi$. This is a simple reverse mapping and is already defined in Theorem 8.1.2.

8.4 Algorithms to generate $\mathcal{A}^1_\pi$ and $\mathcal{A}^2_\pi$

In this section we present simple and efficient algorithms to visit each node of $\mathcal{G}_\pi$ using the Gray codes defined by $\mathcal{A}^1_\pi$ and $\mathcal{A}^2_\pi$. These algorithms are optimal in the sense that they take time proportional to a constant number of operations to transform one code word to the next. The correctness of the algorithms is easy to verify.

Algorithm 3. *Algorithm for generating Gray code defined by* $\mathcal{A}_\pi^1$

```
$t \leftarrow 0;$          $l \leftarrow 0$
$i \leftarrow -(b-1)$
$c \leftarrow 0$
while $c < b$ do
    $t \leftarrow (t+a)$ mod $b$
    if $t < (b-a)$
        then $l \leftarrow 0$
        else $l \leftarrow a$                                  (8.45)
    while $i \leq l$ do
        output $i, t$
        $i \leftarrow i+1$
    $c \leftarrow c+1$
    $i \leftarrow -(b-1)$
```

The algorithm uses t and i to store the imaginary and real parts of the node label that is currently visited. To calculate the running time, firstly we observe that to visit each node in a row takes constant time (Internal **while** loop). This is so because only one operation ($i \leftarrow i+1$) is needed. Next we observe that to set up for visiting the next row the algorithm takes exactly 4 steps (all statements excluding the internal **while**). Therefore, the time to go from one vector to the next is either 4 steps or 1 step. Thus it takes a constant time to transform one vector to the next.

Algorithm 4. *Algorithm for generating Gray code defined by* $\mathcal{A}_\pi^2$

```
$t \leftarrow 0;$    $l \leftarrow 0$
$i \leftarrow (b-1)$
$c \leftarrow 0$
while $c < (a+b)$ do
    $t \leftarrow (t+b)$ mod $(a+b)$
    if $t > 0$
        then $l \leftarrow (b-a)$
        else $l \leftarrow 0$
    while $i \leq l$ do
        output $t, i$
        $i \leftarrow i-1$
    $c \leftarrow c+1$
    $i \leftarrow (b-1)$
                                        (8.46)
```

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

In this proposal for research we have considered the design of new classes of Gray codes and applications of existing Gray codes for new applications. We have shown new constructions of Gray codes along with their applications. We have also shown how existing constructions can be applied to the solution of new problems.

In the construction for generating edge disjoint Hamiltonian cycles in toroidal networks, we assume that the radices are either odd or all are even. This is a slightly restrictive assumption. In the future we would investigate if the methods we have proposed here can be extended to networks with a mix of even and odd radices. Also, we would like to investigate further with extensions to dense gaussian network to if there are advantages in extending the topology to higher dimensions. We could then develop elegant methods to construct edge disjoint Hamiltonian cycles in this network.

BIBLIOGRAPHY

[1] S. Al-Bassam and B. Bose, "Asymmetric/Unidirectional Error Correcting and Detecting Codes", *IEEE Trans. Computers*, Vol. C-43, May 1994, pp. 590-597.

[2] D. J. Amalraj, N. Sundararajan, and G. Dhar, "A data structure based on Gray code encoding for graphics and image processing", *SPIE: International Society for Optical Engineering*, pages 65–76, 1990.

[3] M. Bae and B. Bose, "Lee distance Gray codes and edge disjoint Hamiltonian cycles in toroidal networks", 12^{th} *IEEE International Parallel Processing Symposium*, pages 365–370, May 2000.

[4] L. A. Barroso et al, "Piranha: A Scalable architecture Based on Single-Chip Multiprocessing", *International Symposium on Computer Architecture*, pp.282-293, 2000.

[5] J. M. Berger, "A note on Error-Detecting Codes for Asymmetric Channels" , *Information and Control*, vol. 4, March 1961, pp. 68-73.

[6] R. Beivede, E. Herrada, J. L. Balcázar and A. Arrubarrena. "Optimal Distance Networks of Low Degree for Parallel Computers", *IEEE Transactions on Computers*, Vol. C-40, No. 10, pp 1109-1124, 1991.

[7] J. R. Bitner, G. Ehrlich and E. M. Reingold, "Efficient Generation of the Binary Reflected Gray Code and its Applications" *Communications of the ACM*, Volume 9, Number 19, September 1976.

[8] M. Blaum, "Codes for Detecting and Correcting Unidirectional Errors", *IEEE Computer Society Press*, Los Alamitos, CA, 1993.

[9] B. Bose and D. J. Lin, "Systematic Unidirectional Error-Detecting Codes", *IEEE Trans. Computer*, vol. C-34, November 1985, pp. 63-69.

[10] J. Borden, "Optimal Asymmetric Error-Detecting Codes", *Information and Control*, vol. 53, April 1982, pp. 66-73.

[11] F. T. Boesch and J. Wang, "Piranha: Reliable Circulant Networks with Minimum Transmission Delay", IEEE Transactions on Circ. Systems, 32:pp.188-197, 1985

[12] B. Bose, R. Broeg, Y. Kwon, and Y. Ashir, "Lee Distance and Topological Properties of k-ary n-cubes", *IEEE Transactions on Computers*, 44(8):1021–1030, August 1995.

[13] M. Bae and B. Bose, "Edge Disjoint Hamiltonian Cycles in k-ary n-cubes and Hypercubes", *IEEE Transactions on Computers*, 10:1271–1284, 2003.

[14] R. Broeg, B. Bose and V. Lo, "Lee Distance, Gray codes and the torus", *Telecommunication Systems*, 10:21–32, 1998.

[15] C. C. Chang, H. Y. Chen, and C. Y. Chen, "Symbolic Gray codes as a data allocation scheme for two disc systems," *Computer Journal*, 35(3):299–305, 1992.

[16] M. Cohn, "Affine m-ary Gray codes", *Information and Control*, vol 6, pp 70–78, 1963.

[17] M. Chen and K. G. Shin, “Subcube allocation and task migration in hypercube machines”, *IEEE Transactions on Computers*, 39(9):1146–1155, 1990.

[18] C. Martinez, R. Beivede, J. Guitterez and E. Gabidulin, “On the Perfect t-Dominating Set Problem in Circulant Graphs and Codes Over Gaussian Integers”, *ISIT 2005*, Adelaide, 2005.

[19] C. Martinez,E. Vallejo, R. Beivede, C. Lzu and M. Moreto, “Dense Gaussian Networks: Suitable Topologies for On-chip Multiprocessors”, *International Journal of Parallel Programming*, Vol-34, No. 3, June 2006.

[20] P. Diaconis and S. Holmes, “Gray codes for randomization procedures”, *Statistics and Computing*, 4:287–302, 1994.

[21] D. E. Knuth, “The Art of Computer Programming, Volume 4, Fascicle 2: Generating all Tuples and Permutations”, *Addison Wesley*, 2005

[22] M. C. Er, “On Generating N-ary Reflected Gray codes”, *IEEE Transactions on Computers*, 33(8):739–741, 1984.

[23] C. Faloutsos, “Gray codes for partial match and range queries”, *IEEE Transactions on Software Engineering*, 14(10):1381–1393, 1988.

[24] C. V. Freiman. *Optimal Error Detecting Codes for Completely Asymmetric Binary Channels*, Information and Control, vol. 5, March 1962, pp. 66-71

[25] Martin Gardner, “Curious properties of the Gray code and how it can be used to solve puzzles,” *Scientific American*, 227(2):106–109, 1972.

[26] E. N. Gilbert, "Gray codes and paths on the n-cube", *Bell Systems Technical Journal*, 37:815–826, 1958.

[27] F. Gray, "Pulse Cose Communication", U.S. Patent 2632058, March 1953.

[28] F. Gray, "Pulse Code Communications", *U.S Patent 2632058*, March 1953.

[29] J. .H. Jordan and C. J. Potratz, "Complete Residue Systems in the Gaussian Integers", *Mathematics Magazine*, Vol-38, No. 1, Jan 1965, pp. 1-12.

[30] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, Second Edition, 2004.

[31] R. M. Losee, "A Gray code based ordering for documents on shelves: Classification for browsing and retrieval", *Journal of the American Society for Information Science*, 43(4):312–322, 1992.

[32] L. Tallini, S. Elmougy and B. Bose, *Analysis of Plain and Diversity Combining ARQ Hybrid Protocols over the m($\geq$2)-ary Asymmetric Channel*, *(*Submitted) IEEE Transactions on Information Theory.

[33] J. E. Ludman, "Gray code generation for MPSK signals", *IEEE Transactions on Communications*, 29:1519–1522, 1981.

[34] A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms for Computers and Calculators*, Academic Press, 1978.

[35] E. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms - Theory and Practise*, Prentice Hall, 1977.

[36] D. Richards, “Data compression and Graycode sorting,” *Information Processing Letters*, 22:210–205, 1986.

[37] J. Robinson and M. Cohn, “Counting sequences”, *IEEE Transactions on Computers*, C-30:17–23, 1981.

[38] C. Savage, “A Survey of Combinatorial Gray Codes,” *SIAM REV.*, 39(4):605–629, December 1997.

[39] B. D. Sharma and R. K. Khanna, “On m-ary Gray Codes”, *Information Sciences*, vol 15, pp 31–43, 1978.

[40] S. B. Wicker, *Error Control Systems for Digital Communications and Storage*, Prentice Hall, 1995.

[41] H. S. Wilf, *Combinatorial Algorithms: An Update*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1989.