DELHI TECHNOLOGICAL UNIVERSITY

MAJOR PROJECT-2

Design and Implementation of Golay Codes

Submitted for partial fulfilment of award of

MASTER OF TECHNOLOGY
DEGREE
IN
VLSI DESIGN & EMBEDDED SYSTEMS ENGINEERING

Under the guidance of

Dr Rajiv Kapoor (Professor)

Submitted By:

ANJALI GUPTA

2k14/VLS/02

CONTENTS

CERTIFICATE

I hereby certify that the work entitled **“Design and Implementation of Golay codes”** has been carried out by Ms. Anjali Gupta (2K14/VLS/02) in the partial fulfillment for the award of degree **Master of Technology** in **VLSI Design & Embedded Systems Engineering** submitted in the Department of Electronics and Communication Engineering, Delhi Technological University, Delhi, in the period of June, 2015 under my guidance. The said work has not been submitted anywhere else for the award of any other degree or diploma.

Date: 15.06.16 **Dr. Rajiv Kapoor**

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of major project undertaken during M.Tech second year. I own special debt of gratitude to Dr. Rajiv Kapoor (Professor), Department of Electronics & Communication, Delhi Technological University, Delhi for her constant support and guidance throughout the course of my work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me. It is only his cognizant efforts that my endeavors have seen light of the day.

I will also like the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during my work. Last but not the least, I acknowledge my friends for their contribution in the completion of the work.

ANJALI GUPTA
(2k14/VLS/02)

ABSTRACT

In recent years demand is increasing day by day for digital transmission of data and storage systems. VLSI technology and digital processing techniques are also acting as a supporting factor for this demand. In digital communication it is highly required that it must be fully reliable, as a single error can corrupt the data or can cause complete shutdown of the system, e.g. in online transaction of money. So digital communication systems must be fault tolerant as well as capable of detecting errors and recover it. The easiest way to detect a single error is a parity checksum method which can be implemented with the help of XOR gates. But with the advent of VLSI technology has entered in an era of Nano scale design where number of components on the single chip ranges from 1, 00, 000 to 10, 00, 000. As the number of components increase on the chip, the complexity increases due to large number of interconnections. So, for such cases parity check sum method is not a suitable technique to be used for error detection. Hence, some new coding techniques are required for data transmission.

Various types of Error Control coding schemes like BCH codes, Reed- Solomon codes, Turbo codes, Convolutional codes, LDPC codes are implemented now a days. One of these techniques is Golay encoding and decoding scheme which is used as basic motivation for the purpose of this thesis. Also advanced Golay codes are designed so that Golay codes can be made more reliable and sophisticated.

List of Figures

List of Tables

1. Introduction to Error coding in Digital Communication

1.1 Error Coding

In recent years demand is increasing day by day for digital transmission of data and storage systems. VLSI technology and digital processing techniques are also acting as a supporting factor for this demand. In digital communication it is highly required that it must be fully reliable, as a single error can corrupt the data or can cause complete shutdown of the system, e.g. in online transaction of money. So digital communication systems must be fault tolerant as well as capable of detecting errors and recover it. The easiest way to detect a single error is a parity checksum method which can be implemented with the help of XOR gates. But with the advent of VLSI technology has entered in an era of Nano scale design where number of components on the single chip ranges from 1, 00, 000 to 10, 00, 000. As the number of components increase on the chip, the complexity increases due to large number of interconnections. So, for such cases parity check sum method is not a suitable technique to be used for error detection. Hence, some new coding techniques are required for data transmission. Various types of Error Control coding schemes like BCH codes, Reed-Solomon codes, Turbo codes, Convolutional codes, LDPC codes are implemented now a days. One of these techniques is Golay encoding and decoding scheme which is used as basic motivation for the purpose of this thesis.

1.1.1 What is coding?

It is a technique used to prevent the unauthorized use of data by rearrangement of the symbols. This process is also known as ENCRYPTION of data. It is used in devices that scramble a digital data sequence to avoid the occurrence of long strings of 0s and 1s. It is a method for error detection and then correction. To form a code for error detection and correction, i.e. to encode the data some additional bits termed as redundant bits are added in the message data stream. These bits are termed as redundant bits [1]. In general terms, coding is a mutual assignment between a set of messages to be transmitted and a set of codewords that are used for transmitting these messages [2]. Encoding a data ensures the privacy of a confidential data which is to be protected by unauthorized sources as well as promises a reliable and efficient communication. Encoding also makes sure regarding the compromise between the channel capacity and the data that is to be sent over the channel. The example for encoding the messages is represented by the Table 1.1 shown below.

Table 1.1 Coding: a codeword for each message

Messages	Codewords
S_1	001
S_2	010
S_3	0111

1.1.2 Basic Elements of Digital Communication System

A digital communication system is a way of sharing information from one user to another user or can be said that from one party to another with the help of a sequence of symbols from a finite alphabet either in binary form or octal form or hexadecimal form etc. to represent the information [1]. The basic block diagram for the digital communication system is shown in fig. 1.1.

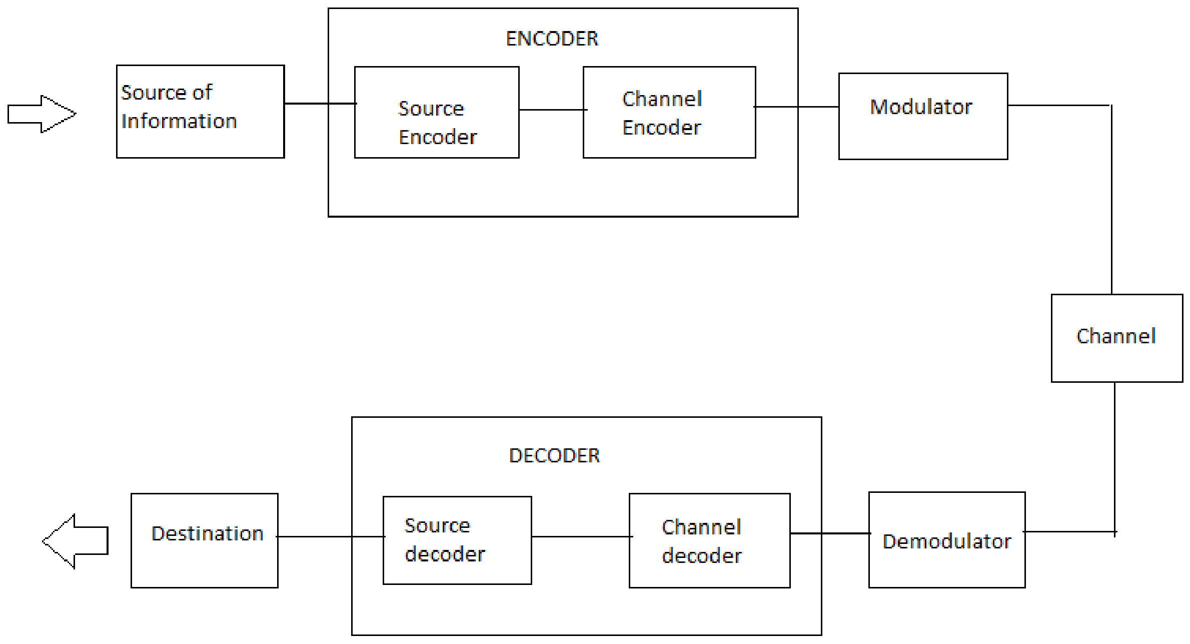


Fig 1.1 Block Diagram of Digital Communication System

Source of Information: The basic source of information can be a speech, audio or video signal which is to be transmitted over a channel.

Encoder: In the block diagram, two types of encoders are used. The main purpose of channel encoder is to perform error correction so that a reliable communication can take place. This purpose can be achieved by using any efficient encoding technique like hamming codes, Golay codes etc. whereas the purpose of source encoder is to make the source information rate R approach the channel capacity.

Modulator: It is the basic process in which one of the parameters of high frequency signal (amplitude, frequency or phase) is varied with respect to the amplitude of message or baseband signal. This is basically used to convert a wideband signal into narrowband signal and also to ensure a long distance communication. Various digital modulation techniques that are used now days are Pulse Code Modulation (PCM), Delta Modulation, Adaptive delta modulation, DPCM, various shift keying techniques like FSK, MSK etc. [3].

Channel: Channel is a medium through which a message signal travels from source to destination. This channel can be a coaxial cable; transmission lines; fiber optic cable or even atmosphere can act as a channel for a signal. Channel should be noise free to ensure the least corruption in data during transmission. To ensure this various regenerative repeaters are used in channel which amplify the signal level to a desired strength and degrade the effect of noise.

Decoder: Decoder is used to perform the exact reverse operation of encoder. Decoder extracts the original message signal from the received data sent be channel by separating the redundant bits and finally providing the data to final destination.

Destination: The received message is finally displayed or stored in destination.

1.2 Basic keyword Definitions

- **Word:** A sequence of symbols is said to be a word.

- **Code:** A code is a set of code words. These set are termed as code words.

- **Hamming Weight:** Hamming weight of a codeword is equal to the number of non-zero elements in the codeword. This is denoted by symbol w(c).

- **Hamming Distance**: The number of places by which two codewords differ is termed as hamming distance. The hamming distance between the two codewords can be represented as d $(\boldsymbol{c_1}, \boldsymbol{c_2})$. It can also be said that d $(\boldsymbol{c_1}, \boldsymbol{c_2})$ = w $(\boldsymbol{c_1} - \boldsymbol{c_2})$.

Example 1.1 Calculate the hamming weight for a code word given as 100101.

Solution: As the code word consists of 3 one's, so the hamming weight is equal to 3.

Example 1.2 Calculate the hamming distance between code word A= 1010101 and B = 1100001.

Solution:

A= 1 0 1 0 1 0 1

B= 1 1 0 0 0 0 1

In the given code words A and B the positions by which these two vectors differ are three. So the hamming distance is equal to 3.

1.3 Parameters which decide the performance of coding theory

a) Number of redundant bits added

Number of redundant bits that are added to form the error detection and error correction codes is a measure of efficiency of the coding theory. There is always a trade-off between number of redundant bits added to the information data bits and the rate at which information is sent over the link.

$$R=K/N \tag{1.1}$$

In equation (1.1) R is the rate at which data is transmitted, K is the no of message bits and N is the total no of bits that are transmitted i.e. the sum of message bits and parity bits.

b) Bit error rate(BER)

It is the no. of bit errors per unit time. Bit error ratio is defined as the number of bit errors divided by the total number of transferred bits during a studied time interval. It is a unit less performance measure, often expressed as a percentage.

$$\text{Bit error rate}=\frac{Number\ of\ bit\ errors}{Total\ time} \tag{1.2}$$

This factor is affected by transmission channel noise, interference, distortion, bit synchronization problems, attenuation, wireless multipath fading, etc. It can be improved by using choosing strong signal strength. The formula for bit error rate is shown in the equation 1.2.

c) Signal-Noise ratio(S/N Ratio)

It is the measure that compares the desired signal level to the level of background noise. It is defined as the ratio of signal power to the noise power often expressed in decibels. Equation 1.3 represents it.

$$\text{SNR}= \frac{Power\ of\ The\ desired\ signal}{Power\ of\ the\ noise\ signal} \qquad (1.3)$$

d) LUTs used in implementation

LUT refers to look up tables which are used in configurable logic blocks of FPGA to implement a function. This parameter is basically used to check how effectively the LUTs are used by a code in encoding and decoding.

e) Net Delay

It represents the total time taken by circuit to give the output after application of input. So basically it is the parameter which deals with the speed of data transfer. Less the speed, more will be the data transfer rate of a medium.

2. Galois fields

2.1 Introduction

In this chapter basic concept of groups, finite fields, vector spaces and arithmetic operators used in the finite fields are introduced. The definitions and main results related to finite field theory are presented along with the derivation of extension fields. Also, advanced encoders and decoders are presented which use arithmetic operators for their operation. These encoders and decoders need less hardware for implementation and also speed up the encoding or decoding by taking less LUT cycles or clocks. Error control codes are dependent on powerful and elegant algebraic tools called finite fields.

2.2 Set

An arbitrary collection of elements or objects without any predefined operations between these elements is termed to be a set. This set may be finite or infinite. The number of elements present in a set is called Cardinality [1].

2.3 Groups

A set of elements G on which binary operation '.' can be applied is said to be a group. When this operation is applied on any two elements of the group, the generated result is also an element of G. The cardinality of a group is called as order of the group [2].

2.3.1 Properties of Groups [2]

For G to be a group, following conditions must be satisfied.

1. Associativity: For every u, v, w in G

$$(u . v) . w = u . (v . w).$$

2. Identity: There exists x in G such that

$$u . x = x . u = u$$

   where u is in G.

3. Commutativity: For every u,v in G

$$u . v = v . u = G$$

4. Inverse: For all u € G, there exits an unique element u^{-1} € G such that

$$uu^{-1} = x$$

2.4 Rings

A ring R is a collection of elements with two binary operations, one is addition '+' and other is multiplication '.'. A ring satisfies the following properties [1, 2]:

1. Commutivity: R forms a commutative group under addition operator '+'. The additive element is considered to be '0'. Also for all u, v € R,

$$u \,.\, v = v \,.\, u$$

2. Associative: For all u, v, w € R,

$$(u \,.\, v) \,.\, w = u \,.\, (v \,.\, w)$$

3. Distributive: For all u, v, w € R,

$$u \,.\, (v + w) = (u \,.\, v) + (u \,.\, w)$$

4. Identity: For u€ R

$$u \,.\, 1 = 1.\, u = u$$

2.5 Field

A field is a set of elements which allows addition, subtraction, multiplication and division of field elements and always obtain another element as a result within the set. If a field contain finite number of elements then it is termed as finite field elements. For example a field of prime numbers is a kind of infinite field whereas a field of first 10 natural numbers is a finite field.

2.5.1 Field definitions and basic features of finite field [7]

A field F is a non-empty set of elements with two operators, one is addition and other is multiplication, denoted '+' and '*' respectively. For F to be a field following conditions must be satisfied [4, 6]:

1. Closure: For every u, v in F the closure is represented as

$$y = u + v; \qquad z = u * v;$$

where $y, z \in F$.

2. Associative: For every u, v, w in F

$$u + (v + w) = (u + v) + w \quad \text{and} \quad u * (v * w) = (u * v) * w.$$

3. Identity: For every u in F

$$0 + u = u + 0 = u \qquad \text{and} \qquad u * 1 = 1 * u = u$$

4. Inverse: If u is in F, there exist elements v and w in F such that

$$u + v = 0 \qquad\qquad u * w = 1.$$

Element v is called the additive inverse, $v = (-u)$, element w is called the multiplicative inverse, $w = u^{-1}$ $(u \neq 0)$.

Multiplicative inverse u^{-1} enables the use of division. This is because for $u, v, w \in F$, $w = v/u$ is defined as $w = v * u^{-1}$. In the same manner additive inverse $(-u)$ allows subtraction. In this case for $u, v, w \in F$, $w = v - u$ is defined as $w = v + (-u)$.

5. Commutative: For every u, v in F

$$u + v = v + u \qquad\qquad u * v = v * u.$$

6. Distributive: For every u, v, w in F

$$(u + v) * w = u * w + v * w.$$

2.5.2 Galois Fields

Galois field (named in the honour of Evariste Galois) is a field with finite number of elements [2]. A field is a set of elements which allows addition, subtraction, multiplication and division of field elements and always obtain another element as a result within the set. If a field contain finite number of elements then it is termed as finite field. For example a field of prime numbers is a kind of infinite field whereas a field of first 10 natural numbers is a finite field. Galois fields are used to construct binary codes.

For a set of integers (1,2,3,……p-1) where p is a prime number along with modulo p addition as well as multiplication, a field is formed which can be represented as GF(p) where GF represents the Galois field. Using this GF(p), the extension field GF(q) can be generated where $q = p^m$. Table 2.1 and 2.2 represents the modulo 3 addition and multiplication.

Table 2.1

Modulo 3 addition in Galois field

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 2.2

Modulo 3 multiplication in Galois field

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

To generate an extension field a primitive polynomial is required. Few definitions are required for its detailed study.

Definition 1: If a polynomial f(x) in GF(q) can't be factored into lower degree polynomials that exist in GF(q) then that polynomial is termed as **irreducible polynomial**. For an example $x^2 + 1$ is irreducible in GF(3).

Definition 2: An irreducible polynomial f(x) of degree m is termed as **primitive polynomial** if $n = p^m - 1$ is the smallest positive integer 'n' for which f(x) divides $x^n - 1$.

Using the above definitions any irreducible polynomial can be represented in the form of primitive elements as shown in table 2.3.

Table 2.3

Polynomial and Vector Representation of GF(8) for p(x)= $x^3 + x + 1$

Expression	Polynomial	Vector
0	0	000
1	1	100
α	α	010
α^2	α^2	001
α^3	$1+\alpha$	110
α^4	$\alpha+\alpha^2$	011
α^5	$1+\alpha+\alpha^2$	111
α^6	$1+\alpha^2$	101
α^7	1	100

If β_1, β_2, β_3,…………………… β_{q-1} denote the non-zero field elements of GF(q) then x^{q-1} -1 can be represented as the product of all these factors which are field elements as shown in equ. (2.1) shown below.

$$x^{q-1}\text{-}1\text{=}(x\text{-}\beta_1)(x\text{-}\beta_2)(x\text{-}\beta_3)\ldots\ldots(x\text{-}\beta_{q-1}) \qquad (2.1)$$

So using this approach x^n-1 can be written as shown in equ. 2.2.

$$x^n - 1 = f_1(x)\ f_2(x)\ f_3(x)\ldots\ldots\ldots\ f_p(x) \qquad (2.2)$$

These basic rules of Galois fields stated above are used for the generator polynomial of Golay codes.

Example 2.1 Let us consider an example for q=5. Since q =5, a prime number modulo arithmetic will work. Consider an element '2'.

$$2^0 = 1\ (mod\ 5) = 1$$

$$2^1 = 2\ (mod\ 5) = 2$$

$$2^2 = 4\ (mod\ 5) = 4$$

$$2^3 = 8\ (mod\ 5) = 3$$

So, we can see that all the elements of GF(5) i.e. (1, 2, 3, 4, 5) can be represented as powers of 2. Therefore 2 is a primitive element of GF(5).

Now let us consider an element '3'.

$$3^0 = 1\ (mod\ 5) = 1$$
$$3^1 = 3\ (mod\ 5) = 3$$
$$3^2 = 9\ (mod\ 5) = 4$$
$$3^3 = 27\ (mod\ 5) = 2$$

So, 3 can also be taken as a primitive element for GF(5).

Next let us consider an element ‘4’.

$$4^0 = 1\ (mod\ 5) = 1$$
$$4^1 = 4\ (mod\ 5) = 4$$
$$4^2 = 16\ (mod\ 5) = 1$$
$$4^3 = 64\ (mod\ 5) = 4$$

In this case we find that the field elements (2, 3) can’t be represented as a power of element ‘4’. So, it can’t be considered as the primitive element for GF(5).

2.6 Vector Spaces

Let S be a set of elements termed as scalars whereas V be a set of vector elements. If a scalar u € S and v € V then w = u.v results in a vector space over V if the conditions given below are satisfied [1]:

1. Commutative: V follows the commutativity under addition operation.
2. Closure: For any element u € S and v € V

u.v = w € V.

3. Distributive: V follows the distributive law as specified below:

u . (w + v) = u . w + u . v

4. Associative: For all t, u € S and all v € V then

(t .u) . v = t. (u . v)

5. Identity: For all v € V, v . 1 = v.

2.6.1 Spanning set of a vector

A collection of vectors J = $\{v_1, v_2, v_3, \dots \dots \dots \dots v_n\}$, the linear combination of which consists of all vectors in the vector space V, is termed as Spanning set of vector V.

2.6.2 Dimension of a Vector Space

Dimension of a vector space V is equal to the number of elements in the vector space. So if the vector space has ‘n’ number of elements then the dim(V) = n.

2.6.3 Dual Spaces of Vector Spaces [8]

Let T be a n-dimensional subspace of a vector space V and $T^{\perp}$ is the set of all the vectors v in V such that all u € T and all v € $T^{\perp}$, then if u .v= 0, then $T^{\perp}$ is said to be the dual space of T. This dual space $T^{\perp}$ of vector subspace T is said to be a vector subspace of V.

2.6.4 The Dimension Theorem of Vector Spaces

Let T be a subspace of vector V with finite dimensions and let $T^{\perp}$ be the dual space for T. then the dimension theorem states that when the dimension of T is added with the dimension of $T^{\perp}$, it gives the dimension of vector V.

$$\dim(T) + \dim(T^{\perp}) = \dim(V) \qquad (2.3)$$

Equation 2.3 given above represents the dimension theorem.

2.7 Minimal Polynomial

The smallest degree polynomial with coefficients in the base field GF(q) that has a zero in the extension field GF(q^m) is called minimal polynomial of β_j [2].

If f(x) is minimal polynomial of β, then it will also be the polynomial of elements in the set $\{\beta^1, \beta^q, \beta^{q^2}, \dots\dots\dots\dots\dots\dots, \beta^{q^{s-1}}\}$, where s is the smallest integer such that $\beta^{q^{s-1}}$ = β. This set of elements is called as set of conjugates. The elements in the set of conjugates are all zeros of f(x). So the minimal polynomial can be represented as in equation 2.4.

$$f(x) = (x-\beta)(x-\beta^q)\dots\dots\dots\dots\dots\dots\dots(x-\beta^{q^{s-1}}) \qquad (2.4)$$

Example 2.2 Let us consider a field F_8.Now we will calculate the minimal polynomial for this field. Use the table 2.3 for the polynomials of respective expression.

Solution: For a F_q field first of all calculate x $(x^{q-1} + 1)$. So here we get $x^8 + x$. Now using equation 2.9

$x^8 + x = \prod_j x - \beta_j$ = x(x+1)(x+α)(x+α^2)(x+α^3)(x+α^4) (x+α^5) (x+α^6)

Now using the property of conjugates we know that the polynomial for β, β^2, β^4 will be same. So (1, 2, 4) forms a cyclotomic-coset. The minimal polynomial for this coset will be given as:

(x+α)(x+α^2)(x+α^4) = $(x^2 + x\alpha^2 + x\alpha + \alpha^3)$(x+α^4)

$=x^3 + x^2\alpha^4 + x^2\alpha^2 + x\alpha^6 + x^2\alpha + \alpha^5 x + \alpha^3 x + \alpha^7$

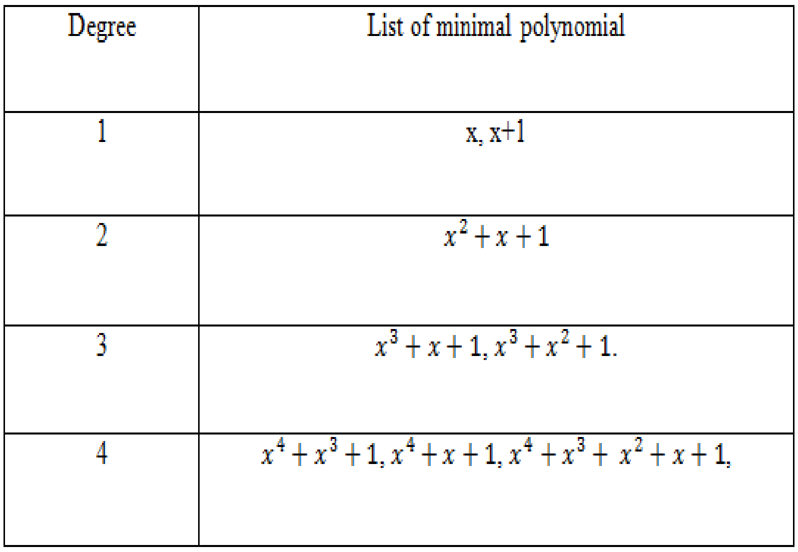
$=x^3 + x^2(\alpha^2 + \alpha) + x^2\alpha^2 + x(1 + \alpha^2) + x^2\alpha + x(1 + \alpha + \alpha^2) + x(1 + \alpha) + 1$

$=x^3 + x + 1$

So the minimal polynomial for cyclotomic coset is $x^3 + x + 1$. This can be easily proved by putting the values of $x=\alpha/\alpha^2/\alpha^4$ and we get a zero value, i.e these are the factors of this minimal polynomial.

A table for certain minimal polynomials is provided below:

Table 2.4 List of minimal polynomial



Degree	List of minimal polynomial
1	x, x+1
2	$x^2 + x + 1$
3	$x^3 + x + 1$, $x^3 + x^2 + 1$.
4	$x^4 + x^3 + 1$, $x^4 + x + 1$, $x^4 + x^3 + x^2 + x + 1$,

Similarly (3, 5, 6) forms a cyclotomic coset and provides a polynomial $x^3 + x^2 + 1$.

3. Block Codes

In this chapter we move to the practical world of error control systems by leaving the atmosphere of abstract algebra. A very basic code Block Code is introduced in this chapter. These codes use a controlled amount of redundancy for encoding the data and also provide the ability to receiver for the decoding of the transmitted codeword.

In the first section very basic introduction to Block codes is provided. In second section Properties of Linear Block Codes with their encoding strategies are introduced. Third section deals with the Syndrome error detection in the Block codes.

3.1 Introduction to Block Codes

A block code is a set of fixed length code words. This fixed length of any code word is termed as block length which is denoted by ‘n’. A block code of size ‘M’ defined over an alphabet with q- symbols is a set of M q-ary sequences where each sequence is of length ‘n’.

If q=2, then the symbols are termed as bits whereas the code is said to be a binary code. General expression for the relation between the M and q is given in equation 3.1.

$$M=q^{k} \tag{3.1}$$

where k is number of bits in the message data.

Generally a block code is represented by (n, k) in which the redundant bits are given by (n-k).

3.1.1 Code Rate for Block codes

The code rate for (n, k) block code is defined as the ration of k and n and this ratio depicts the fraction of the code word that consists of the information symbols. The maximum value of code rate is ‘1’ when k=n. As the code rate approaches to the value ‘1’ more efficient will be the code. The formula for code rate is given in equation 3.2.

$$R=k/n \tag{3.2}$$

3.1.2 Minimum Distance in Block codes

The minimum distance in two code words of a block code represent how strong a code is in terms of the error correct ability, i.e. how many errors in a code can be corrected.

Definition 3.1: The minimum distance in block codes is defined as the minimum hamming distance between all distinct pairs of code words in a code.

Theorem 3.1: A code with minimum distance d_m can detect all error patterns that has weight less than or equal to $d_m - 1$.

Theorem 3.2: A code with minimum distance d_m can correct all error patterns that has weight less than or equal to $(d_m - 1)/2$.

Example 3.1 Encode a message bit string 01101110001011 where the two bits binary number is represented as shown below:

Uncoded bits	Code words
00	00000
01	10010
10	11110
11	11111

Solution: Step 1. First of all, calculate the value of n and k. Here k=2 and n=5 i.e. a 2-bit message is converted into a 5-bit code word having 3 redundant bits.

Step 2.Now split the given message into groups of 2 bits each. So the message can be split as shown below.

01 10 11 10 00 10 11

Step 3. Now replace each group with the corresponding code word given in the question statement and hence the coded data is

10010 11110 11111 11110 00000 11110 11111

So the final encoded message will be 10010111101111111110000001111011111.

3.2 Linear Block Codes [14]

Definition 3.2: A block code (n, k) is said to be a linear block code if all the 2^k codewords forms a k-dimension vector subspace, of the vector space V of all the vectors of length n with components in the field GF(2).

3.2.1 Properties of Linear Block Codes

A linear code has the following properties:

- If two code words of a block code are summed up, then the resultant code word is also a part of the code.
- The minimum hamming distance between two code words of the linear block code is basically the minimum weight of any two non-zero code words.

3.2.2 Generator Matrix G

Generator matrix is used to encode a message of length k into a codeword of size n. Let us consider that Input matrix V is of order 1×k whereas the order of the output encoded matrix U is 1×n, in that case the order of generator matrix G must be k×n to satisfy the equation 3.3.

$$U=VG \tag{3.3}$$

The generator matrix for the linear block codes is shown in equation 3.4.

$$G=\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots\cdots\cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots\cdots\cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots\cdots\cdots & g_{k-1,n-1} \end{bmatrix} \tag{3.4}$$

$$[u_0 \quad u_1 \quad \cdots\cdots \quad u_{n-1}]=$$

$$[m_0 \quad m_1 \quad \cdots\cdots \quad m_{k-1}] \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots\cdots\cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots\cdots\cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots\cdots\cdots & g_{k-1,n-1} \end{bmatrix} \tag{3.5}$$

Equation 3.5 represents the encoding of data in matrix form which can also be written as given in equation 3.6.

$$[u_0 \quad u_1 \quad \cdots\cdots \quad u_{n-1}]=[m_0 \quad m_1 \quad \cdots\cdots \quad m_{k-1}] \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

$$=m_0g_0 \oplus m_1g_1 \oplus \cdots\cdots \oplus m_{k-1}g_{k-1} \tag{3.6}$$

Example 3.2 Given the Generator Matrix G $=\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$. Find the encoded result for the four packets of message, [0 0], [0 1], [1 0], [1, 1].

Solution:

U=VG

For the first message input i.e. [0 0]

u_1 = [0 0] $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ = [0 0 0]

For the 2nd message input i.e. [0 1]

u_2 = [0 1] $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ = [0 1 1]

For the 3rd message input i.e. [1 0]

u_3 = [1 0] $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ = [1 0 1]

For the 4th message input i.e. [1 1]

u_1 = [1 1] $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ = [1 1 0]

The final code for the given problem is {000, 011, 101, 110}.

3.2.3 Parity Check matrix

A matrix which provides an easiest method to decide whether there is an error in the received codeword or not, is called parity check matrix. This matrix is represented by H and is given in equation 3.7.

$$\mathrm{H}=\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots\cdots\cdots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \cdots\cdots\cdots & h_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots\cdots\cdots & h_{n-k-1,n-1} \end{bmatrix} \tag{3.7}$$

The order for parity check matrix is (n-k) × n.

For a valid codeword the following equation must satisfy

$$\mathrm{U}H^T = 0 \tag{3.8}$$

In equation 3.8 H^T represents the transpose of the parity check matrix.

$$\mathrm{VG}H^T = 0 \tag{3.9}$$

From equation 3.3 U=VG, so equation 3.8 can be written as shown in equation 3.9.

Equation 3.9 can also be summarized to equation 3.10 in case if Generator matrix is given and a parity check matrix is to be generated.

$$\mathrm{G}H^T = 0 \tag{3.10}$$

For systematic linear block codes Generator matrix is given as G= [I|P], where 'I' is a k×k identity matrix. Using equation 3.10 Parity check matrix can be calculated as

$$\mathrm{H}= [-\, P^T|I] = 0 \tag{3.11}$$

where 'I' in equation 3.11 is a (n-k) × (n-k) matrix.

Theorem 3.3: A vector U is a valid codeword iff UH^T = 0.

Example 3.3 Calculate the parity check matrix H for a linear block code (7, 4) if the Generator matrix G is given as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Solution:

From the given Generator matrix, matrix P is

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

So the parity check matrix can be calculating by taking the transpose of this parity check matrix and then concatenating it with identity matrix,

$$H= \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3.3 Syndrome Error Detection [3]

When a codeword is transmitted over a noisy channel, there may be the chances to receive an erroneous codeword on the receiver side. This received vector is different from the corresponding transmitted codeword.

Let the received vector be

Y = $(y_0, y_1, \dots \dots \dots . y_{n-1})$

If there is any error in the received codeword then that error pattern can be represented as

E= $(e_0 \, , e_{1\,,} \dots \dots \dots \dots \dots \dots . e_{n-1})$

The equation relating the received vector to the transmitted codeword is given by equation 3.12 and 3.13.

$$E= Y\oplus U$$

$$U=Y\oplus E \qquad (3.12)$$

$$Y=U\oplus E \qquad (3.13)$$

From equation 3.8 it is known that

$$UH^T = 0$$

So, an error detection technique is used to detect the errors using a Syndrome vector which is represented using S= $(s_0 \, , s_{1\,,} \dots \dots \dots \dots \dots \dots . s_{n-k-1})$.

This syndrome is calculated by multiplying the received vector with the transpose of parity check matrix as shown in equation 3.14.

$$S=YH^T \quad (3.14)$$

Using Equation 3.13 in equation 3.14

$$S= (U\oplus E)H^T= UH^T\oplus EH^T= EH^T \quad (3.15)$$

UH^T is equal to zero in equation 3.15 as shown in equation 3.8.

3.3.1 Properties of Syndrome Error Detection

- If the error pattern is a zero vector, then syndrome vector will also be a zero vector. This represents a valid code word.
- If the syndrome vector consists of at least one non-zero component, that means it is detecting the errors in the received vector.

3.3.2 Drawback of Syndrome error detection [1]

There is a possibility that the syndrome vector can't detect the errors even in presence of error. This is because syndrome can be a zero vector even if there are errors in the received code word. It happens in the case when the error pattern is equal to a code word i.e. the number and positions of the errors lead to a vector which is also a code word. So, in such a case syndrome method is not suitable for error detection. Such errors are called as undetected error patterns and these errors are not within the error correction ability of the code. This situation happens when S= $EH^T = 0$ and it is possible when E=U, i.e. when the error pattern is equal to the code word. There can be $2^k - 1$ such error patterns that can't be detected.

4. Cyclic Codes

Cyclic codes were first introduced in various technical notes and reports by E. Prange at the Air Force Cambridge Research Labs during 1957-1959. These cyclic codes led to the development of BCH codes, Reed-Solomen codes and Golay codes. These codes are considered to be an important class of linear block codes. Cyclic codes also possess an advantage of being implemented using high speed sequential logic or shift register based encoders and decoders. In this chapter general theory of cyclic codes is provided.

In the first section basic introduction to cyclic codes is provided which includes Polynomial representation, systematic representation of cyclic codes and various matrices used for encoding purpose. Second section deals with the encoding mechanism of the cyclic codes. Third section consists of decoding strategy used for the cyclic codes.

4.1 Introduction

Definition 4.1: An (n,k) linear code is said to be cyclic when every cyclic shift applied to the codeword results in a codeword existing in the code. Let U is an n-tuple code vector which can be represented as, as shown in equation 4.1.

$$U = [u_0\ u_1 \dots \dots \dots \dots \dots \dots u_{n-1}] \qquad (4.1)$$

If this codeword is cyclic shifted by j number of bits then it can be written as U^j and the code vector can be written as shown in equation 4.2

$$U^j = [u_{n-j}\ u_{n-j+1} \dots \dots \dots u_{n-1}\ u_0 \dots \dots \dots u_{n-j-1}] \qquad (4.2)$$

Example 4.1: U = [1 0 1 1]

U^2 = [1 1 1 0]

Theorem 4.1

Iff the code polynomials in U form an ideal as GF(q)[x]/($x^n - 1$) then U is said to be a q-ary linear cyclic code.

4.1.1 Polynomial Representation of Cyclic codes [4]

Cyclic codes can be represented in the form of a polynomial as shown in equation 4.3, U(x) being the code polynomial of code vector U.

$$U(x) = u_{n-1}x^{n-1} + u_{n-2}x^{n-2} + \dots\dots\dots\dots u_1x^1 + u_0 \qquad (4.3)$$

The polynomial representation for the cylic shifted code vector is represented as $U^j(x)$ as shown in equation 4.4.

$$U^j(x) = u_{n-j-1}x^{n-1} + u_{n-j-2}x^{n-2} + \dots\dots\dots\dots u_1x^{j+1} + u_0x^j + u_{n-1}x^{j-1} + u_{n-2}x^{j-2} + \dots\dots u_{n-j+1}x^1 + u_{n-j} \qquad (4.4)$$

4.1.2 Cyclic codes in Systematic Form

Multiplying the equation 4.4 with x^j results in equation 4.5 and 4.6

$$x^j U(x) = u_{n-1}x^{n+j-1} + u_{n-2}x^{n+j-2} + \dots\dots \dots\dots u_{n-j}x^n + u_{n-j-1}x^{n-1} + u_{n-j-2}x^{n-2} + \dots\dots\dots\dots u_1x^{j+1} + u_0x^j \qquad (4.5)$$

$$x^j U(x) = -u_{n-1}x^{j-1} - u_{n-2}x^{j-2} - \dots\dots\dots\dots - u_{n-j+1}x^1 + u_{n-j} + u_{n-1}x^{n+j-1} + u_{n-2}x^{n+j-2} + \dots\dots\dots + u_{n-j+1}x^{n+1} + u_{n-j}x^n + u_{n-j-1}x^{n-1} + u_{n-j-2}x^{n-2} + \dots\dots\dots\dots u_1x^{j+1} + u_0x^j + u_{n-1}x^{j-1} + u_{n-2}x^{j-2} + \dots\dots u_{n-j+1}x^1 + u_{n-j} \qquad (4.6)$$

$$x^j U(x) = q(x)(x^n - 1) + U^j(x) \qquad (4.7)$$

where $q(x) = u_{n-1}x^{j-1} + u_{n-2}x^{j-2} + \dots\dots u_{n-j+1}x^1 + u_{n-j}$

Equation 4.7 represents the final equation that will represent the nature of cyclic codes.

4.1.3 Matrix Representation of Cyclic codes

For all the code polynomials in U, there exists a unique polynomial G(x), termed as generator polynomial with minimal degree less than 'n'. The generator polynomial G(x) of U is a factor of $x^n - 1$ in GF(q)[x]. Let us consider a generator polynomial G(x) which is written as in equation 4.8

$$G(x) = g_{n-k}x^{n-k} + g_{n-k-1}x^{n-k-1} + \dots\dots\dots\dots g_1x^1 + g_0 \qquad (4.8)$$

The code polynomial U(x) is given by

$$U(x) = V(x)\,G(x) \qquad (4.9)$$

Where V(x) is input vector set.

Using equation 3.9 the code polynomial can be written as in equation 4.10

$$U(x) = (v_{k-1}x^{k-1} + v_{k-2}x^{k-2} + \dots\dots\dots\dots + v_0)\,G(x) \qquad (4.10)$$

The generator matrix can be represented in the form of matrix as in equation 4.11

$$G=\begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & \cdots & 0 & 0 \\ & \vdots & & & & \vdots & & \\ 0 & 0 & \cdots & g_0 & g_1 & \cdots & g_{n-k} & 0 \\ 0 & 0 & \cdots & 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} \end{bmatrix} \qquad (4.11)$$

These k row vectors as shown in the matrix are linearly independent vectors and we can represent the final output as

$$U = v_0 g_0 + v_1 g_1 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots v_{k-1} g_{k-1} \qquad (4.12)$$

Equation 4.12 suggests the k-dimensional space for the n-tuple vector space over GF(q).

The parity check matrix for the cyclic codes denoted as ‘H’ can be represented as in 4.13.

$$H=\begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 & 0 \\ 0 & h_k & \cdots & h_1 & h_0 & \cdots & 0 & 0 \\ & \vdots & & & & \vdots & & \\ 0 & 0 & \cdots & h_k & h_{k-1} & \cdots & h_0 & 0 \\ 0 & 0 & \cdots & 0 & h_k & \cdots & h_1 & h_0 \end{bmatrix} \qquad (4.13)$$

Parity check matrix also acts as generator matrix for the dual code of code V.

4.2 Encoding of Cyclic codes

4.2.1 System Algorithm for encoding of Cyclic codes [1]

Step 1. Multiply the message polynomial V(x) by x^{n-k}.

Let us consider the message polynomial V(x) represented as

$$V(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots\ldots\ldots. v_1x^1 + v_0$$

Step 2. Divide the result of step 1 by G(x) i.e. the generator polynomial.

Dividing x^{n-k} V(x) by G(x), we get

x^{n-k} V(x) = A(x)G(x) + B(x)

$$A(x)G(x) = x^{n-k}\, V(x) - B(x) \qquad (4.14)$$

A(x) is the quotient and B(x) is remainder in equation 4.14. This B(x) can be written in polynomial form as shown in equation 4.15.

$$b_{n-k-1}x^{n-1} + b_{n-k-2}x^{n-2} + \ldots\ldots\ldots. b_1x^1 + b_0 \qquad (4.15)$$

Step 3. Calculate the code polynomial.

A(x)G(x) can also be termed as Code Polynomial U(x). So the final expression for the code in terms of message polynomial is shown in equation 4.16

$$U(x) = x^{n-k}\, V(x) - B(x) \qquad (4.16)$$

Example 4.2 A (7, 4) binary cyclic code generated by G(x) $=x^3 + x^2 + 1$. Messge Polynomial V(x) is V(x) $=x^3 + x^2 + x$. Find code polynomial U(x).

Solution: U(x) = x^{n-k} V(x) – B(x)

Here n=7 and k=4. So n-k =3.

x^{n-k} V(x) $=x^3(x^3 + x^2 + x) = x^6 + x^5 + x^4$

Dividing $x^6 + x^5 + x^4$ by G(x) gives B(x) =x.

So U(x) = $x^6 + x^5 + x^4 + x$

4.2.2 Implementation of Cyclic encoder using Shift Registers [4]

For the systematic encoding of the cyclic codes, the steps are stated in section 4.2.1. These steps can be performed using shift registers, XOR gates and a multiplexer.

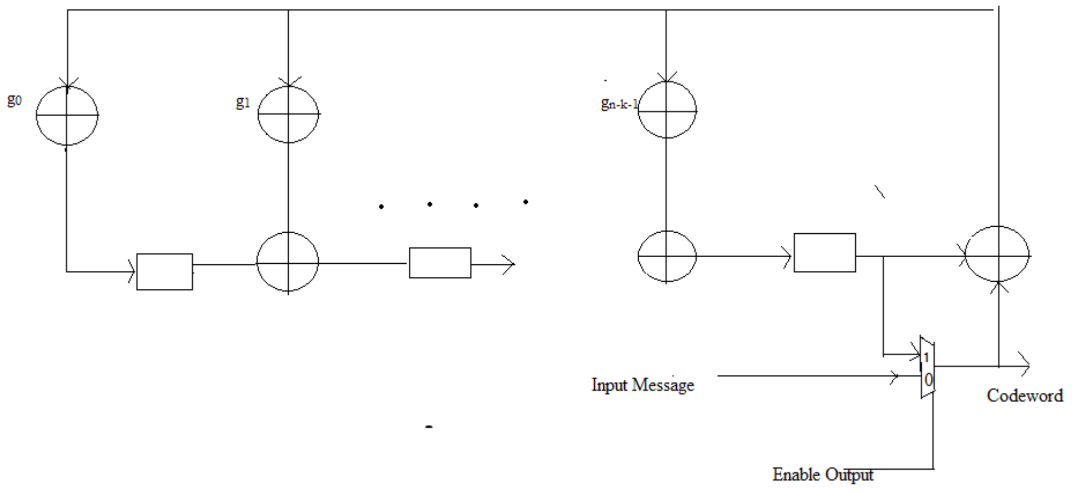


Figure 4.1 Encoding of Cyclic codes

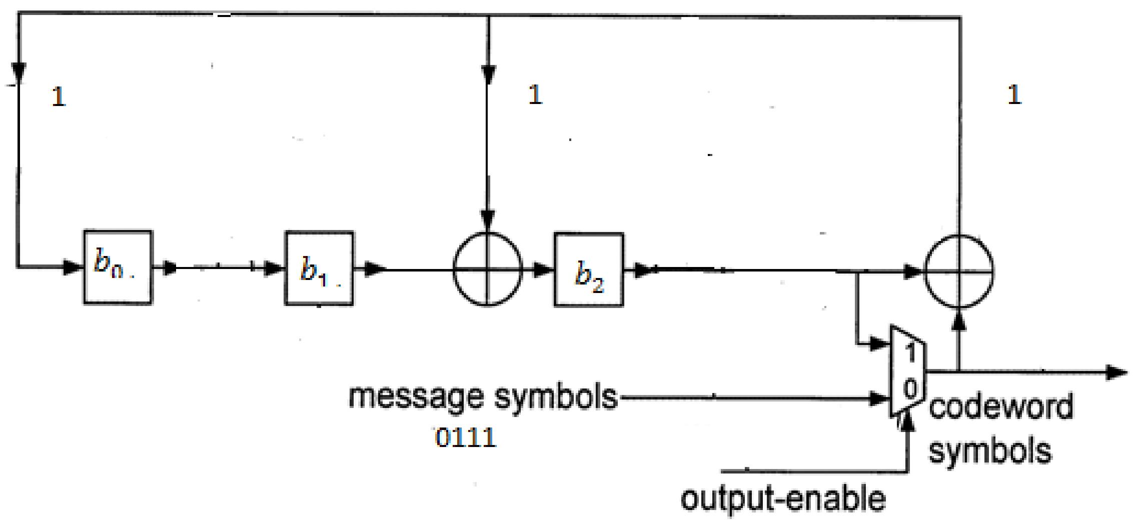
1. First of all a select signal Output enable in multiplexer is used to select whether to load the message bits to calculate the remainder or to enable the output.

2. Initially output-enable is set to ‘0’, and hence message bits $v_{n-1}, v_{n-2}, \ldots\ldots\ldots v_1, v_0$ are transmitted and shifted in the same order into the circuit. This shifting operation is performed using shift registers. This circuit calculates x^{n-k} V(x) and x^{n-k} V(x)/G(x) simultaneously. After ‘k’ shifts the coefficients of remainder i.e. $b_0, b_1, \ldots\ldots\ldots\ldots\ldots b_{n-k-1}$ are stored in the respective registers.

3. Now output-enable is set to value’1’ to transmit the coefficients $b_{n-k-1}, b_{n-k-2}, \ldots\ldots\ldots b_0$ after shifting them and finally provides the desired code word.

Example 4.3 Consider G(x) =$x^3 + x^2 + 1$. Messge Polynomial V(x) =$x^3 + x^2 + x$. Implement the encoder using shift registers, XOR gates and multiplexer.

Solution: The circuit for the desired encoder is implemented below.



Following Table shows the results for all the calculation:

Shift	Input	$b_0 = \text{input} + b_2$	$b_1 = b_0$	$b_2 = \text{Input} + b_1 + b_2$
1	1	1	0	1
2	1	0	1	0
3	1	1	0	0
4	0	0	1	0

After 4 shifts B(x) = x.

U(x) = x^{n-k} V(x) – B(x)

x^{n-k} V(x) =$x^3(x^3 + x^2 + x) = x^6 + x^5 + x^4$

So U(x) = $x^6 + x^5 + x^4$ +x

Final code word will be [0100111].

4.3 Decoding of Cyclic codes [5]

The syndrome vector for an (n, k) systematic cyclic code can be represented using a polynomial S(x) of maximum degree n-k-1.

$$S(x) = [S_0\ S_1 \ldots\ldots\ldots\ldots\ldots\ldots.S_{n-k-1}] \qquad (4.17)$$

Syndrome vector can be calculated by summing a parity polynomial computed from the information part of R(x) i.e. the received signal and the received parity polynomial.

Let us consider U(x) is the code polynomial and E(x) is the error polynomial for the systematic cyclic code. The degree of E(x) is less than or equal to n-1.

$$U(x) = U_s(x) - B(x)$$

Let error polynomial is split into two parts one is $E_s(x)$ of degree n-1(equation 4.18) and other part is $E_c(x)$ of degree n-k-1(equation 4.19).

$$E(x) = E_s(x) + E_c(x)$$

$$E_s(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \dots\dots\dots\dots\dots e_{n-k}x^{n-k} \qquad (4.18)$$

$$E_c(x) = e_{n-k-1}x^{n-k-1} + e_{n-k-2}x^{n-k-2} + \dots\dots\dots\dots\dots e_0 \qquad (4.19)$$

The received word from a transmission channel is given by the sum of code word polynomial and the error polynomial if the channel is considered to be noisy or channel is

$$\begin{aligned} R(x) &= U(x) + E(x) \\ &= U_s(x) - B(x) + E_s(x) + E_c(x) \\ &= U_s'(x) + B'(x) \qquad (4.20) \end{aligned}$$

$$U_s'(x) = U_s(x) + E_s(x)$$

$$B'(x) = -B(x) + E_c(x)$$

The degree for $U_s'(x)$ is n-1 whereas the degree for $B'(x)$ is n-k-1.

$$R(x) = K(x)G(x) + S(x)$$

Where K(x) is the quotient and S(x) is the remainder.

Now dividing R(x) by G(x) results in equation 4.21

$$\frac{U_s'(x)}{G(x)} + \frac{B'(x)}{G(x)} = K(x) + \frac{S(x)}{G(x)} \qquad (4.21)$$

Equation 4.21 can be transformed in equation 4.22 as following

$$\text{Rem}\left[\frac{U_s'(x)}{G(x)}\right] + \text{Rem}\left[\frac{B'(x)}{G(x)}\right] = \text{Rem}\left[\frac{S(x)}{G(x)}\right] \qquad (4.22)$$

As the degree of $B'(x)$ and S(x) is less as compared to the degree of G(x), equation 4.22 can be stated as in 4.23.

$$S(x) = \text{Rem}\left[\frac{U_s'(x)}{G(x)}\right] + B'(x) \qquad (4.23)$$

S(x) is the syndrome polynomial which can be stated as

$$s_{n-k-1}x^{n-k-1} + s_{n-k-2}x^{n-k-2} + \ldots\ldots\ldots + s_2x^2 + s_1x^1 + s_0 \qquad (4.24)$$

Now as we know that

$$R(x) = K(x)G(x) + S(x)$$

$$R(x) = U(x) + E(x)$$

Equating these two equation results in

$K(x)G(x) + S(x) = U(x) + E(x)$

As the codeword U(x) is a multiple of Generator Polynomial G(x) we can write

$$Q(x)G(x) + S(x) = A(x)G(x) + E(x)$$

$$E(x) = [Q(x) - A(x)]\, G(x) + S(x) \qquad (4.25)$$

Theorem 4.2: For a received polynomial if S(x) is considered as the syndrome polynomial then for the jth cyclic shift of R(x) termed as $R^j(x)$, the syndrome polynomial will be $S^j(x)$, i.e. the syndrome polynomial after j number of cyclic shifts and $S^j(x)$ is given by the remainder of $\frac{x^j\, S(x)}{G(x)}$.

4.3.1 Implementation of Cyclic decoder using Shift Registers

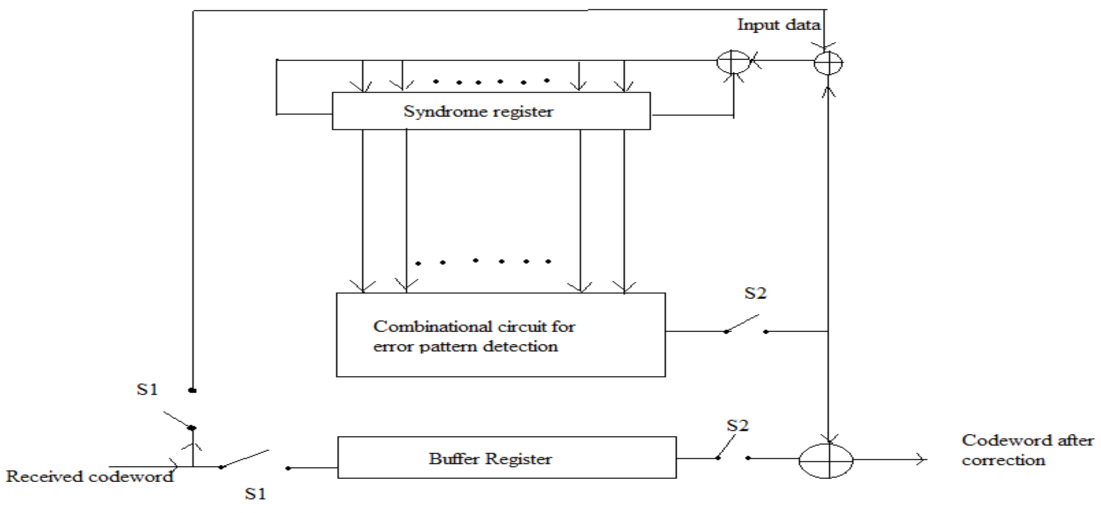

Input data
Syndrome register
Combinational circuit for error pattern detection
S2
S1
S2
Buffer Register
Codeword after correction
Received codeword
S1

Figure 4.2 Implementation of Cyclic Decoder.

1. When switch S1 is closed and S2 is open, the bits of received codeword are shifted into buffer register and also in syndrome register. The syndrome register consists of

the coefficients of syndrome polynomial S(x) after n shifts. Open the switch S_{IN} after the computation of syndrome coefficients.

2. Now close S2 switch whereas S1 is kept open. The bits of syndrome register are loaded into the error pattern detector to check whether the error is correctable or not. If highest order position i.e. x^{n-1} doesn't has any error, i.e. $e_{n-1} = 0$, then the contents of syndrome register and buffer register are shifted cyclically to the right. Then r_{n-2} of R(x) becomes first digit of $R^1(x)$ and in this way all bits are tested.

$$r_1(x) = x^{n-1} + r(x)$$

$$= (r_{n-1} \oplus e_{n-1})\, x^{n-1} + r_{n-2}x^{n-2} + \ldots + r_1 x + r_0$$
$$= r_{n-1}{}' x^{n-1} + r_{n-2}x^{n-2} + \ldots + r_1 x + r_0 \qquad (4.26)$$

Where $r_{n-1}{}' = (r_{n-1} \oplus e_{n-1})$

3. The effect of error digit e_{n-1} can be removed by using equation 4.27.

$$s_1(x) = e_{n-1}\, x^{n-1} + S(x) \qquad (4.27)$$

4. After that syndrome register and buffer register both are cylindrically shifted to the right once and it results in equation 4.28.

$$S_1^1\,(x) = \text{Rem}\,\{x\frac{s_1(x)}{G(x)}\}$$
$$= \text{Rem}\,\{[x^n + xS(x)]/G(x)\}$$
$$= \text{Rem}\{x\, S(x)/G(x)\} + 1$$
$$= S^1\, x + 1 \qquad (4.28)$$

5. Repeat these steps to finally decode the code word to get the message data.

4.4 Conclusion

Cyclic codes possess an advantage of being implemented using high speed sequential logic or shift register based encoders and decoders. These discussed properties of cyclic codes became the motivation for the generalization of BCH codes and Reed-Solomon codes which are explained in the next chapter. These codes are also considered to be an important class of linear block codes.

5. Golay Codes

The Golay code was first introduced by M. J. E. Golay in [19] for error detection and correction in a transmitted message. Golay codes are known for their vital role in deep space network programme of JPL-NASA [20], ultrasonic imaging [21], radiolocation [22], etc. These codes are known for their application in the Voyager aircraft I and II launched by United States for the purpose of space exploration. In this chapter the various encoding and decoding mechanisms for the Golay codes are discussed.

5.1 Introduction

The binary Golay code is generally represented as (n, k, d) where ‘n’ denotes the number of transmitted bits,’ k’ represents the number of message bits and ‘d’ represents the minimum binary distance between two codewords. For an example G_{23} Golay code is represented as (23, 12, 7) where the message to be transmitted is of 12 bits, the received codeword will be of 23 bits and the minimum hamming distance between the codewords is 7.

The extended Golay code is generated by the additional parity bit introduced in the codeword. It is denoted by G_{24} and represented as (24, 12, 8). So it has 24 bits in final codeword with 8- bit hamming distance.

5.2 Encoding of Golay Codes

For the construction of Golay codes a generator matrix is required. This generator matrix can be constructed using Hadamard matrices. These matrices further use either Sylvester construction or Paley construction methodology to finally calculate a generator matrix. Before going to the in depth study of construction of Golay codes first an overview of Hadamard matrices, Sylvester construction and Paley construction is given below.

5.2.1 Hadamard Matrices

Hadamard Matrices are introduced by J. Hadamard in [23] in 1893.The main feature with these matrices is the maximum possible value of determinant lies in range [-1, 1] for a given order of m*m. In Hadamard matrices the inner product of any two distinct rows is zero i.e. these rows are said to be orthogonal.

So it can be stated from (5.1) that the product of Hadamard matrix H along with its transposed matrix H^T gives an identity matrix I multiplied by a scalar 'm' where n is the order of the matrix.

$$[H][H^T] = m\,[I] \tag{5.1}$$

Theorem 5.1 For an 'm' order Hadamard matrix. m is 1, 2,4, 8, 12, or it can be a multiple of four [1].

Hadamard matrices can be constructed by two techniques, Sylvester construction and Paley construction.

5.2.1.1 Sylvester Construction Methodology

It states that for a Hadamard matrix of order m i.e. H_m, H_{2m} can be calculated as shown in equ. 5.2.

$$H_{2m} = \begin{bmatrix} H_m & H_m \\ H_m & -H_m \end{bmatrix} \tag{5.2}$$

Let us suppose that $H_1 = [1]$. Then to calculate the H_4 the steps are to be taken as follows:

1. First calculate $H_{2.}$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

2. Then calculate $H_{4.}$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

In this way Hadamard matrix using Sylvester Construction can be calculated for H_8, H_{16} and so on.

5.2.1.2 Quadratic residues and Paley Construction Methodology

The technique of Paley construction was introduced by R. E. A. C. Paley in [24] in 1933. This technique is based on the quadratic residues which are basically squares of non-zero values modulo x; where x is any prime number. For an example let us say for prime number x, the elements of the set includes {0, 1, 2, 3,… ,x-1}. The squares of these elements can be represented as {0, 1, 4, 9,…$(x-1)^2$} and then the final residues can be calculated by modulo x. Say x = 23. So the set can be represented as {0, 1, 2, 3, …, 22} and the set for squares of non-zero element is {1, 4, 9, 16, 25,……., 484}. Now if the residues are calculated for this set using MOD 23 these residues will be {1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18}. The main properties of quadratic residues are given below:

1. All the squares repeat the first (x-1)/2 squares but in reverse order. For example for x =23 the quadratic residues are {1, 4, 9, 16, 2, 13, 3, 18, 12, 8, 6, 6, 8, 12, 18, 3, 13, 2, 16, 9, 4, 1}.
2. (x-1)/2 quadratic residues modulo x exist for x being a prime number.
3. If two quadratic residues are multiplied then the resultant will be a quadratic residue.
4. If two quadratic non-residues are multiplied then the resultant will be a quadratic residue.
5. If one quadratic residue and one quadratic non-residue are multiplied then the resultant will be a quadratic non-residue.

Now according to the rules of Legendre symbol [25] introduced by Adrien- Marie- Legendre in 1798, the elements for Jacobsthal matrix R_{ab} can be calculated using equ. (5.3).

$$R_{ab} = X(b-a) \qquad (5.3)$$

where X(y) is the Legendre symbol.

X(y) = 1; if y is a quadratic residue for modulo x

X(y) = 0; if y is a multiple of x.

X(y) = -1; if y is a quadratic non- residue for modulo x.

Using the above principals stated above now let us move to the construction of Binary Golay codes G_{23} and extended binary Golay code G_{24}.

5.2.2 Generator matrix for G_{24}

As the no of message bits in G_{24} Golay code is 12, so the prime number used for the generation of Jacobsthal matrix is 11. So the quadratic residues will be {1, 3, 4, 5, 9}. Now using equation (5.3) the Jacobsthal matrix R_{11} is calculated using the following steps:

1. Calculate the first row for a=0 in R_{11}.

   0 1 -1 1 1 1 -1 -1 -1 1 -1

2. Calculate all the remaining rows of R_{11} by performing the cyclic shift operation on the first row. The resultant matrix is shown on the next page.

$$R_{11} = \begin{bmatrix} 0 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ -1 & 0 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & -1 & 0 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 0 & 1 & -1 & 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & 0 & 1 & -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 0 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & -1 & 0 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & 0 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 0 & 1 & -1 \\ -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 0 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 0 \end{bmatrix}$$

3. To convert the above Jacobsthal matrix into the Paley Hadamard matrix H_{12}, the formula used is shown below in equation 5.4.

$$H_{12} = \begin{bmatrix} 1 & \mathbf{1} \\ \mathbf{1}^{T} & R_{11} - I_{11} \end{bmatrix} \qquad (5.4)$$

$$H_{12} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \end{bmatrix}$$

where $\mathbf{1}^{\mathbf{T}}$ represents the transpose of row with all 1's.

4. Now to convert the Paley Hadamard matrix into binary Hadamard matrices B_{12} the

process is to convert 1’s to zeros and -1’s to 1’s.

$$B_{12} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1
\end{bmatrix}$$

5. Final Generator matrix is calculated as shown in equation (5.5).

$$G_{24} = \begin{bmatrix} \mathbf{1}^{T} & I_{11} & \mathbf{0}^{T} & B_{11} \\ 0 & \mathbf{0} & 1 & \mathbf{1} \end{bmatrix} \qquad (5.5)$$

So the final generator matrix for extended Golay code is shown in figure 5.1.

$$\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}$$

Fig. 5.1 Generator Matrix for Extended Golay code

After calculating the generator matrix for the extended Golay code, the codeword can be calculated using equation 5.6 shown below

$$U=MG \qquad (5.6)$$

where M is the input message of order 1*12, G is the generator matrix of order 12*24 and U is the codeword of order 1*24.

For generalization let us consider that

M= $m_{k-1}x^{k-1}$ +$m_{k-2}x^{k-2}$ +...........+m_0

$$[u_0 \quad u_1 \quad \dots\dots \quad u_{n-1}] = [m_0 \quad m_1 \quad \dots\dots\dots \quad m_{k-1}] \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

$$= m_0 g_0 \oplus m_1 g_1 \oplus \dots\dots\dots \oplus m_{k-1} g_{k-1}$$

5.3 Generator Polynomial for the Golay code

For Golay code (23, 12, 7) the possible generator polynomials can be:

1. $x^{11} + x^9 + x^7 + x^6 + x^5 + x^1 + 1$
2. $x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + x^1$

Both of the above polynomials are irreducible polynomial over the galois field GF(2). So there must exist a primitive element α € GF(2^{11}) such that G(α) = 0. The elements of GF(2^{11}) can be represented as in equation (5.7):

$$GF(2^{11}) = b_0 + b_1\alpha^1 + b_2\alpha^2 + \dots\dots\dots\dots\dots b_{10}\alpha^{10} \qquad (5.7)$$

This α is a primitive twenty-third root of unity in galois field GF(2^{11}). So it can be stated that generator polynomial G(x) generates a cyclic BCH block code with code word length 23 and this code is called as Golay Code.

Since the minimum distance of Golay code is 7, then by using the equation (5.8), it can be shown that Golay codes can correct three errors at most.

$$d \geq 2t+1 \qquad (5.8)$$

The code word U(x) for a Golay code can be represented in the form of coefficients u_j of the polynomial as shown in equation (5.9).

$$U(x) = \sum_{j=0}^{22} u_j x^j \qquad (5.9)$$

Here u_j € GF(2) and x is considered to be an intermediate.

The generator Polynomial G(x) for a Golay code is given by equation 5.10 [26].

$$g(x) = \sum_{k=0}^{10} (x - \alpha^{2k}) \qquad (5.10)$$
$$= x^{11} + x^9 + x^7 + x^6 + x^5 + x^1 + 1$$

There can be only two generator polynomials for the generation of binary Golay codes G_{23}.

5.4 Efficient Encoding technique for Golay codes

5.4.1 Algorithm for hardware efficient Encoder [27]

1. First of all a generator polynomial G(x) is chosen for calculation of check bits.
2. After that to the right of M(x) i.e. the message input, eleven zeros are appended to evaluate the polynomial P(x) which is further used in long division process along with generator polynomial.
3. After the long division process the remainder bits are used as check bits except the MSB (Most Significant Bit). These check bits are appended with the message bits and it gives encoded Golay Codeword (23, 12, 7).

To convert the binary Golay codeword into extended Golay codeword (24, 12, 8)a parity bit is added to the binary Golay code. Parity bit 0 is appended in case the weight of binary golay code is even whereas for odd weight 1 is appended.

Example 5.1 Let us consider an example where M(x) = A27h and G(x) is AE3h. So P(x) in binary can be written as 1010 0010 0111 0000 0000 000.

```
              ______________________________________
101011100011 |
             |  1010 0010 0111 0000 0000 000
            _|  1010 1110 0011
                ______________________________
                0000 1100 0100 0000
                     1010 1110 0011
                _____________________
                     0110 1010 0011 0
                      101 0111 0001 1
                _________________________
                       011 1101 0010 10
                        10 1011 1000 11
                  __________________________
                         01 0110 1010 010
                           1 0101 1100 011
                   ________________________
                          0 0011 0110 0010 00
                                10 1011 1000 11
                     _________________________
                                01 1101 1010 110
                                  1 0101 1100 011
                       _________________________
                                  0  1000 0110 101
```

We get the check bits as 435h. So the encoded message is A27435h.

To convert this binary code into extended golay code a parity bit is appended. As in the coded message the weight of the message is 11 i.e. 11 ones are present, so parity bit '1' is included in it. Hence the final codeword becomes as (1010 0010 0111 1000 0110 101 1).

5.4.2 Hardware efficient architecture for generation binary Golay Code and Extended Golay code

The architecture for Binary Golay code and Extended Golay code are shown in the figure 5.2, 5.3 and 5.4.

For this architecture first a section which is generating binary Golay code is shown and then in the next section binary Golay code is converted to extended Golay code.

Generation of Binary Golay code

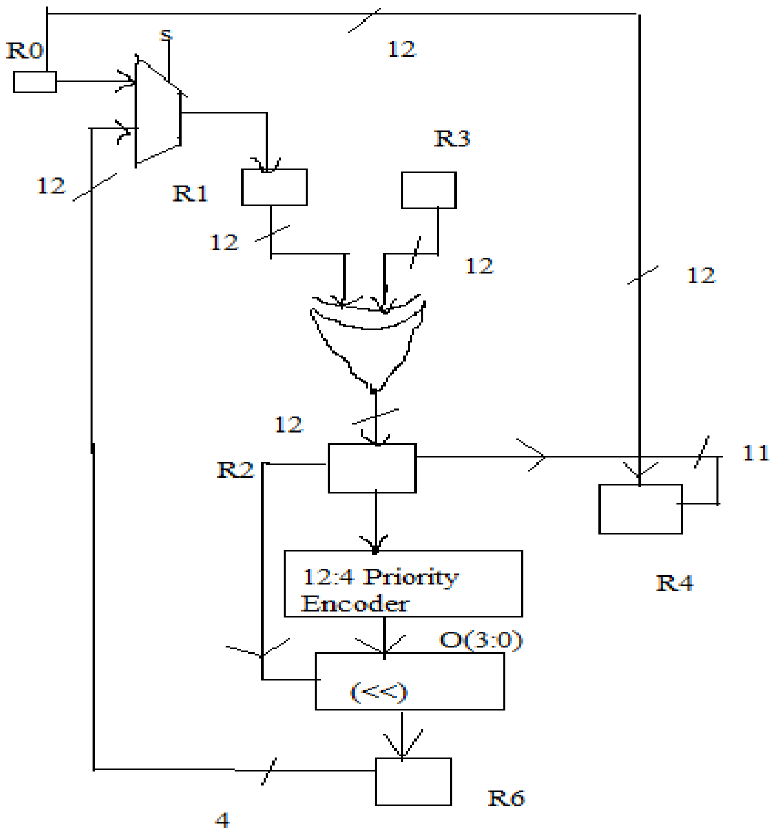


Figure 5.2 Architecture for generation of binary Golay code

For polynomial division simple binary XOR operation is performed on bits of generator polynomial and P(x) for modulo-2 subtraction. The output of XOR gate is stored in a register and then applied to a priority encoder to detect the leading zeros before first 1 occurs. Then P(x) is circular shifted by the output of the priority encoder using a circular shift register. A multiplexer with two inputs and one select line is used to choose between the original message and circularly shifted message. The select line used in the MUX is bit wise OR operation of priority encoder output.

A controlled subtractor is used to deal with loop. Initially, the subtractor is provided with ‘11’ as one input. This ‘11’ actually denotes the number of zeros that are appended to the message input for long division process. The input of subtractor is updated by using the contents of a register depicted as R3 in the diagram. . The other input that is given to the priority encoder is the output of priority encoder. After the final iteration is executed, the output of subtractor gets zero which is stored in register R3 and as the contents of R3 gets zero, R4 is loaded which represents the termination of the long division process providing the check bits.

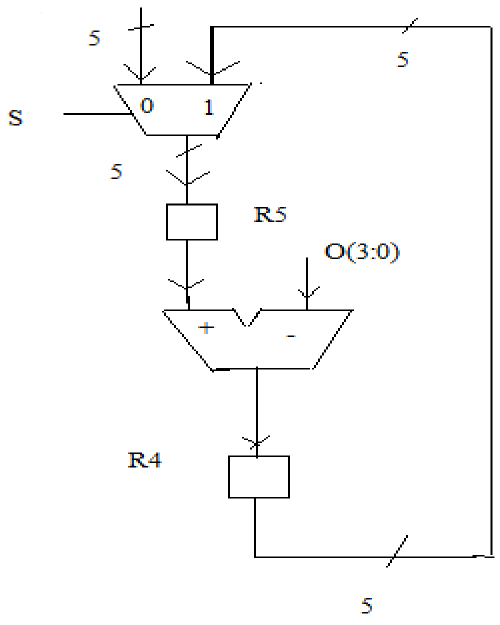


Figure 5.3 Decision Control Unit

A signal t is used to handle the loading of contents of register R2[10:0] after final iteration at R3. Basically

t= (!R4(4)&(|(R4)))

where |(R4) represents the bitwise OR operation performed on the contents of register R4.

As t gets zero R2 is mapped to the register R3 and we get the desired code word.

Generation Of Extended Golay Code:

In this case weight of the binary Golay code is stored in register R9. Register R40 contains content of R4 that are combined with zero whereas register R41 contains content of R4 combined with the value 1. Then a multiplexer is used as a data selector out of R40 and R41 contents.

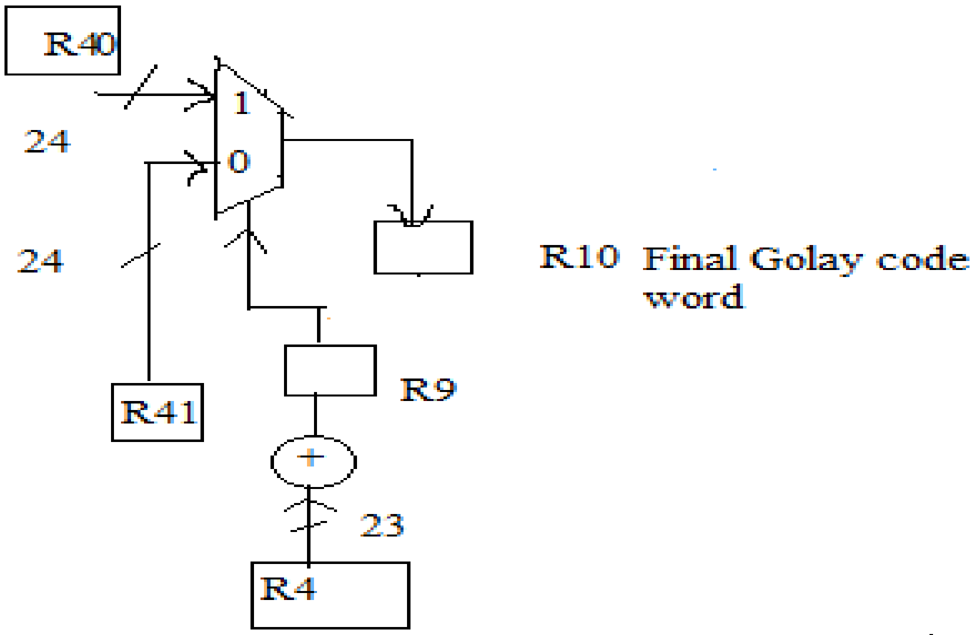


Figure 5.4 Architecture to convert Binary Golay code word to extended Golay code word

The LSB bit of register R9 controls the multiplexer as its select line.

5.5 Decoding of Golay Codes

5.5.1 IMLD Algorithm [28]

To decode the extended Golay code, IMLD algorithm is used. Let us consider that R is the received code word, G is the generator matrix and E is the error pattern. As there exists two parity check matrices [G|I] and [I|G], so two syndromes i.e.S_1 and S_2 are possible. The IMLD algorithm for decoding the golay codes is stated below:

1. First of all calculate the first syndrome vector which is given by S_1= R*H.
2. If weight of syndrome vector S_1 is less than equal to three then set the error vector as E= [S_1, 0] and go to step 8.
3. If weight of S_1+G_i is less than equal to two, where G_i is row I of generator matrix, then set the error vector as E= [S_1+G_i, E_i] and go to step 8.

4. Calculate the second syndrome vector S_2.
5. If weight of syndrome vector S_2 is less than equal to three then set the error vector as E= [0, S_2] and go to step 8.
6. If weight of S_2+G_i is less than equal to two, where G_i is row I of generator matrix, then set the error vector as E= [E_i,S_2+G_i] and go to step 8.
7. The error pattern can’t be detected, so retransmit the data.
8. Final decoded vector is R_1 = R+E.

This IMLD algorithm can find utmost detect three error patterns in the received code word.

5.6 Conclusion

Literature Surveys for different techniques used for encoding and decoding of Golay codes are conducted [26], [29]- [31] but no method is found which is suitable for detection and correction of 5 and 4 errors respectively. Although an efficient technique for designing of encoder and decoder is introduced in [27] for (G_{23}) and (G_{24}) but this technique is also suitable for only triple error- correction. So need of advanced Golay codes arises which can correct more number of errors. The detailed work is presented in the next chapter.

6. Advanced Golay Codes

As the Golay codes find a crucial role in the space applications, different antenna transmissions, laser excitation, ultrasonic imaging etc. there is a huge need to improve the codes so that they can be more efficient. The main prospective of this work is to design advanced Golay code (G_{47}) and (G_{48}) which can detect 5 errors and can correct up to 4 errors. Hadamard matrices are used for this purpose and along with these matrices concepts of Paley construction and Quadratic Residue codes are used.

6.1 Introduction

Advance Golay code $G_{47}$is represented as (47, 24, 10) where a message of 24 bits is coded into a codeword of 47 bits with the hamming distance between the codewords being 10.

As the hamming distance is 10, so using the hamming distance bound theorem in which $d \geq 2t+1$, it can be stated that these codes can correct up to four errors and can detect 5 errors in a sent message.

6.2 Generator Matrix for Advanced Golay codes

The generator matrix is a very essential component for the encoding of any cyclic codes. In the similar manner, the generator matrix for advanced Golay codes is required. The steps required for the required generator matrix are as follows:

1. For G_{48} code number of message bits is 24, so the prime number that will be used to calculate the quadratic residues will be 23. So the first step is to calculate the quadratic residues modulo 23. For modulo 23 the set is {0, 1, 2, 3, …, 22} and the set for squares of non- zero element is {1, 4, 9, 16, 25,……., 484}. Now if the residues are calculated for this set using MOD 23 these residues will be {1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18}.
2. Use the rules of Legendre symbol [25] to calculate the first row of Jacobsthal matrix as given in section 5.2.1.2. The first row is

   [0 1 1 1 1 - 1 – 1 1 - - 1 1 - - 1 – 1 - - - -]

   In this row -1's are represented only by '-' to reduce the ambiguity between 1's and -1's.
3. The Jacobsthal matrix R_{23} is calculated by performing the cyclic shift operations on the first row as these codes are cyclic codes. After the cyclic shift operations, the resultant matrix is shown on the next page.

$$\begin{bmatrix}
0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-\\
-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-\\
-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-\\
-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-\\
-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1\\
1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-\\
-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1\\
1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-&-\\
-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1&-\\
-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1&1\\
1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-&1\\
1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-&-\\
-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1&-\\
-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1&1\\
1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-&1\\
1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1&-\\
-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-&1\\
1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1&-\\
-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1&1\\
1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1&1\\
1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1&1\\
1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0&1\\
1&1&1&1&-&1&-&1&1&-&-&1&1&-&-&1&-&1&-&-&-&-&0
\end{bmatrix}$$

Fig. 6.1. Jacobsthal matrix R_{23}.

4. In the fourth step Jacobsthal matrix is converted into binary Hadamard matrix using the formula shown in equation 6.1.

$$H_{24} = \begin{bmatrix} 1 & \mathbf{1} \\ \mathbf{1^T} & R_{23} - I_{23} \end{bmatrix} \tag{6.1}$$

Here **1** represents a row of 23 number of 1's and $\mathbf{1^T}$ represents the column with 23 number of 1's. I_{23} is identity matrix of order 23*23, whereas R_{23} is Jacobsthal matrix**.** The resultant binary Hadamard matrix is shown in figure 6.2. In case of Hadamard matrix all the 1's of Jacobian matrix are converted into 0's and -1's are converted into 1's to get our final resultant Hadamard matrix.

5. In fifth step the binary Hadamard matrix is converted into the G_{24} matrix by using the equation shown in (6.2).

$$G_{24} = \begin{bmatrix} \mathbf{1^T} & I_{23} & \mathbf{0^T} & H_{23} \\ 0 & \mathbf{0} & 1 & \mathbf{1} \end{bmatrix} \tag{6.2}$$

H_{23} is obtained by removing the leftmost column and the topmost row of H_{24}.

$$\left[\begin{array}{cccccccccccccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1\\
0&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1\\
0&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1\\
0&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1\\
0&1&1&1&.1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0\\
0&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1\\
0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0\\
0&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1\\
0&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0&1\\
0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0&0\\
0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1&0\\
0&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1&1\\
0&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0&1\\
0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0&0\\
0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1&0\\
0&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0&1\\
0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1&0\\
0&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0&1\\
0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0&0\\
0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0&0\\
0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0&0\\
0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1&0\\
0&0&0&0&0&1&0&1&0&0&1&1&0&0&1&1&0&1&0&1&1&1&1&1
\end{array}\right]$$

Fig. 6.2. Hadamard Matrix A_{24}.

6. Final normalized generator matrix is represented in the form $[I_{24}, B_{24}]$ where B_{24} is

$$\left[\begin{array}{cccccccccccccccccccccccc}
0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0\\
1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1\\
1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1\\
1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1\\
1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1\\
1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1\\
1&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0\\
1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1\\
1&1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0\\
1&1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1\\
1&0&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1\\
1&0&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0\\
1&1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0\\
1&1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1\\
1&0&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1\\
1&0&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0\\
1&1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0\\
1&0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1\\
1&1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0\\
1&0&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1\\
1&0&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0\\
1&0&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0\\
1&0&1&1&1&1&1&0&1&0&1&1&0&0&1&1&0&0&1&0&1&0&0&0
\end{array}\right]$$

Fig. 6.3 B_{24} Matrix for Golay Code G_{48}.

shown in fig.6.3 and I_{24} is identity matrix of order 24.

6.3 Encoding of Advanced Golay codes

The input message of 24 bits can be converted into a codeword of 48 bits using the generator matrix [I_{24}, B_{24}]. The formula i.e. used for encoding mechanism is shown in equation 6.3.

$$U=MG \tag{6.3}$$

Where M is the message input of order 1*24 and U is the output codeword of order 1*48.

Let us consider a message polynomial [011010010000011011110000]. Now if this message polynomial is multiplied with generator polynomial [I_{24}, B_{24}] then the final codeword is represented as [011010000000011011110000110110010001000110000010].

6.4 Conclusion

So we have designed advanced Golay encoder for the purpose of better error detection, high SNR, better bit error ratio. The various simulation results are performed for both Golay encoders i.e. G_{24} and G_{48} in next chapter and hence compared.

7. Golay Code Synthesis System

In this chapter Golay encoder is implemented using VHDL platform. Using this tool various parameters required in the encoder are provided as input and it provides us a VHDL model for the encoder. The VHDL codes are written in a way that they can provide us basic gate level design as well as sophisticated FPGA chip can also be designed using its FPGA floorplan.

7.1 G_{24} Encoder

7.1.1 RTL Schematic

This file shows a schematic representation of the pre-optimized design in terms of generic symbols that are independent of the targeted Xilinx device, for example, in terms of adders, multipliers, counters, AND gates, and OR gates etc.The RTL Schematic for the Golay encoder is shown in figure 7.1. RTL Schematic represents the circuit as a block with the inputs and output.

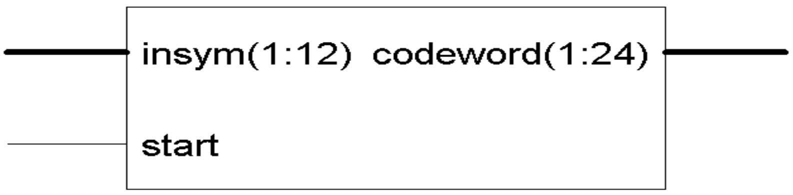


Fig 7.1 RTL Schematic for the G_{24} Encoder

Internal view of the RTL Schematic

The internal view of the RTL schematic provides the internal nets and data paths taken by input signal to reach to output as shown in figure 7.2. Also it can be seen that a component is instantiated again and again. This component is a multiplexer whose internal structure is shown in figure 7.3.

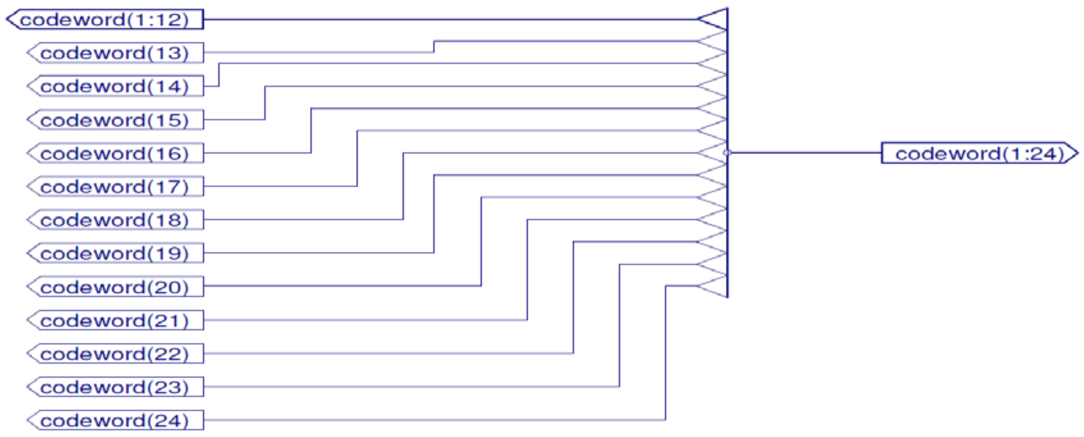


Fig. 7.2 Internal View of RTL Schematic of G_{24} Encoder

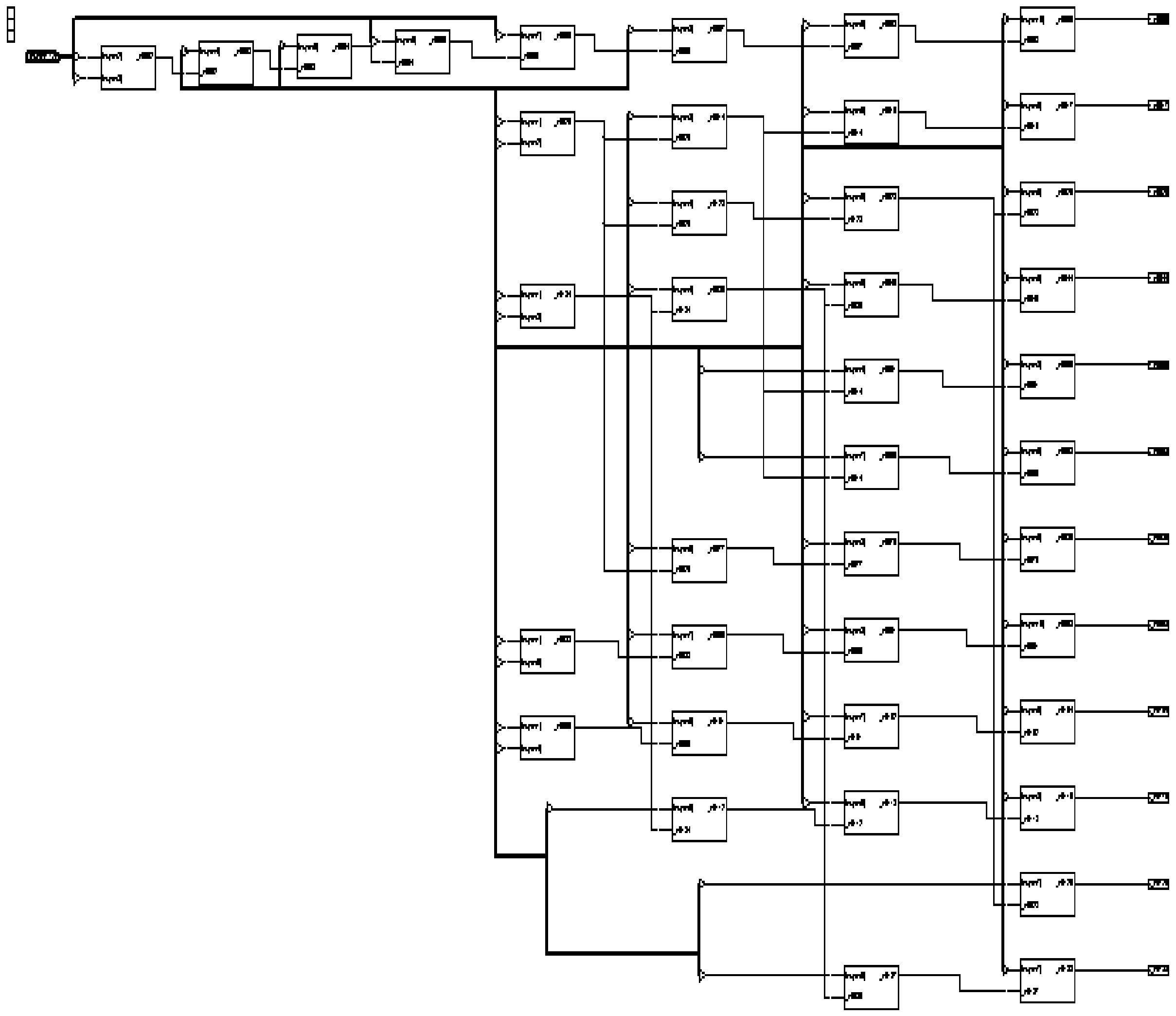


Fig. 7.2 Internal View of RTL Schematic of Golay Encoder

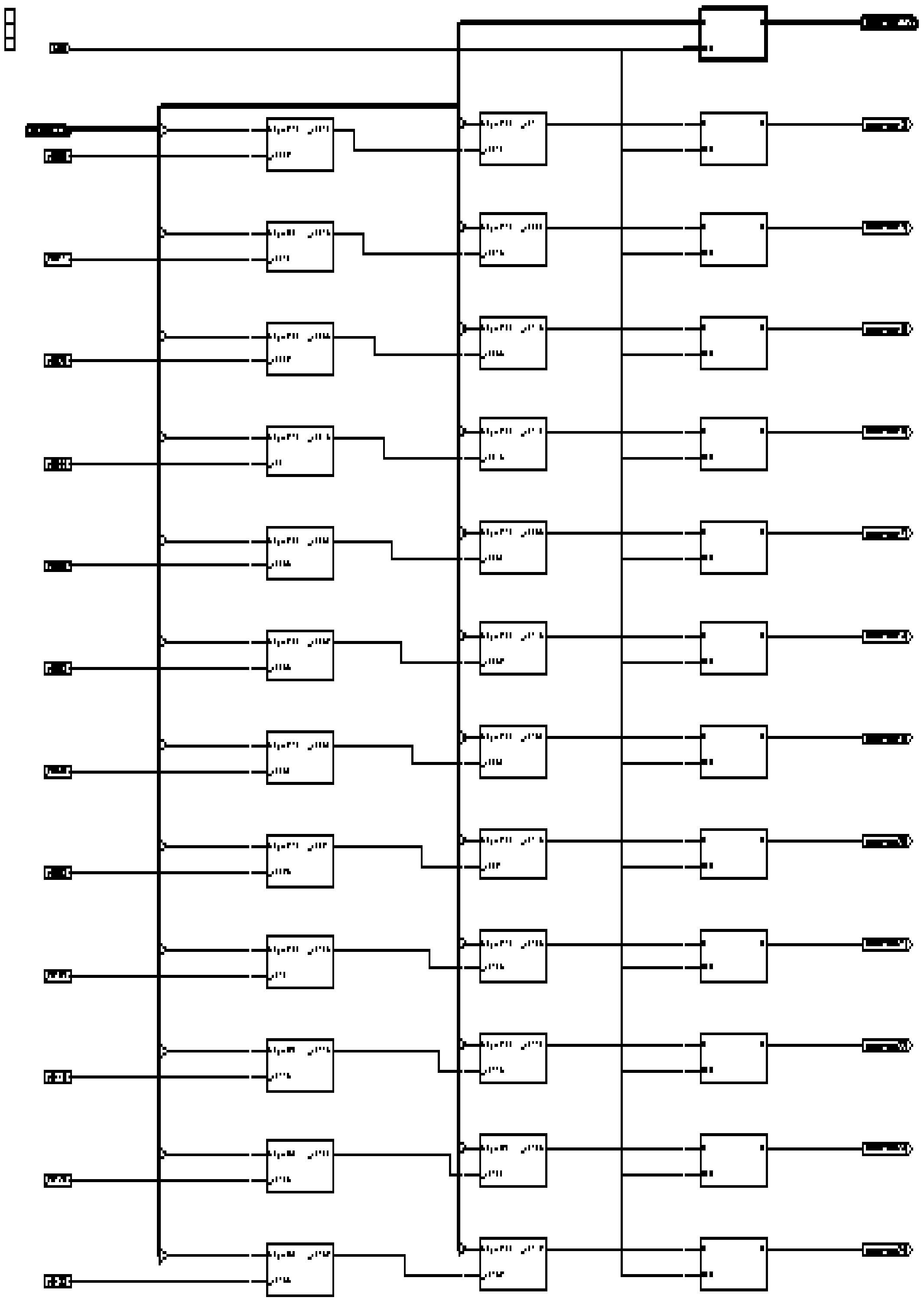


Fig. 7.2 Internal View of RTL Schematic of G_{24} Encoder

Internal view of the instance

In the internal view of the instance shown in figure 7.3 we can see that a multiplexer is instantiated again and again and the mux outputs the data based on the select signal.

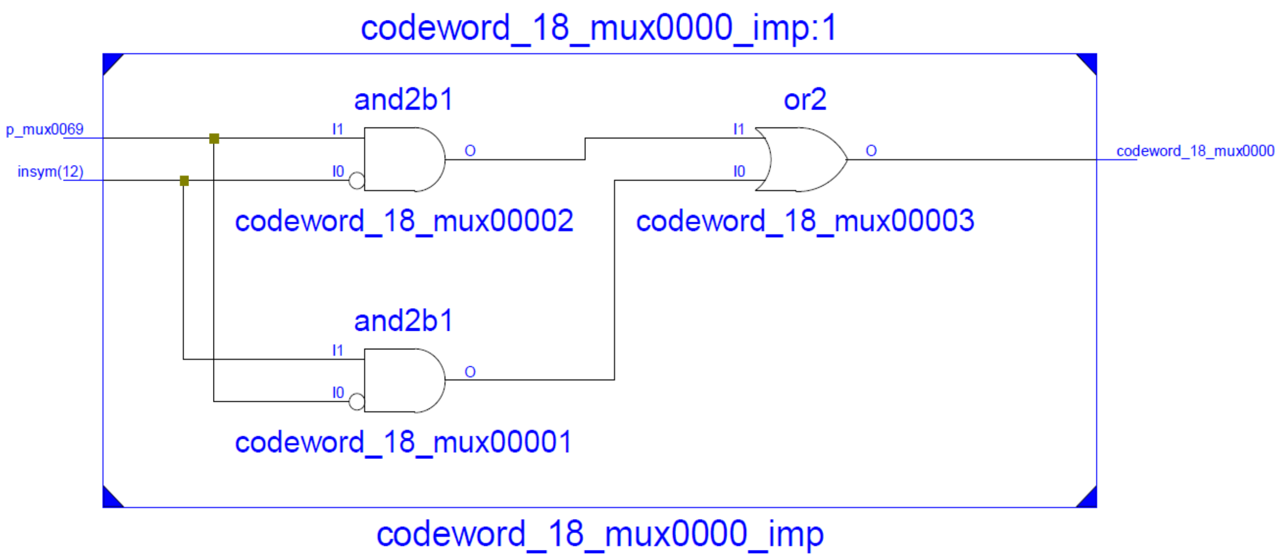


Fig 7.3 Internal view of the instance of RTL Schematic for G_{24} Encoder

7.1.2 Technology Schematic of Golay Encoder:

This file contains the detailed information of the exact elements used in FPGA chip. In technology schematic the required process is explained in the form of Flip-Flops, LUTs (Look Up Tables), input-output buffers which are an integrated part of Configurable Logic blocks in FPGA. The Technology schematic view for the Golay Encoder is shown in Figure 7.4 on the next pages.

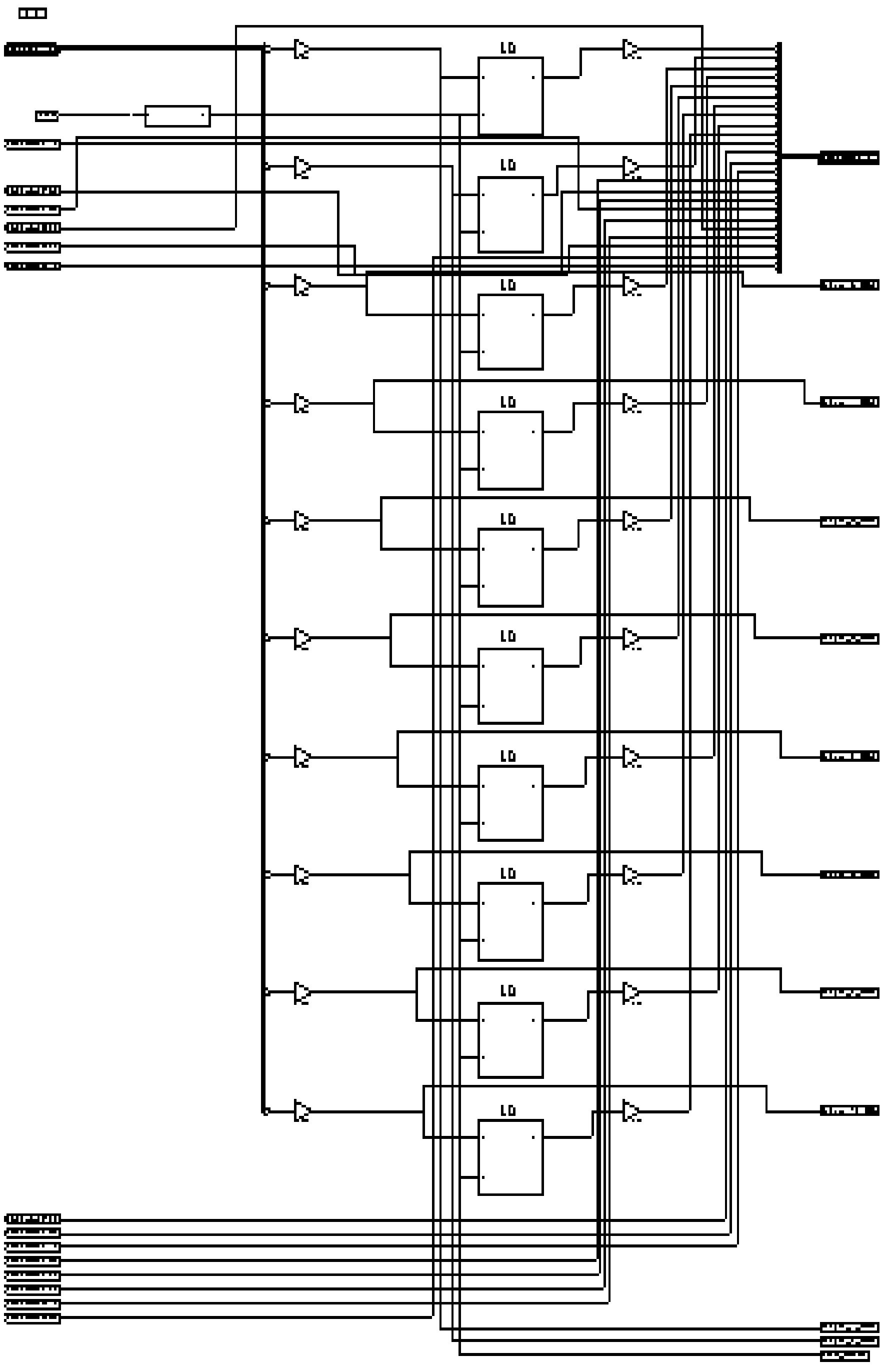


Figure 7.4 Technology Schematic of G_{24} Encoder

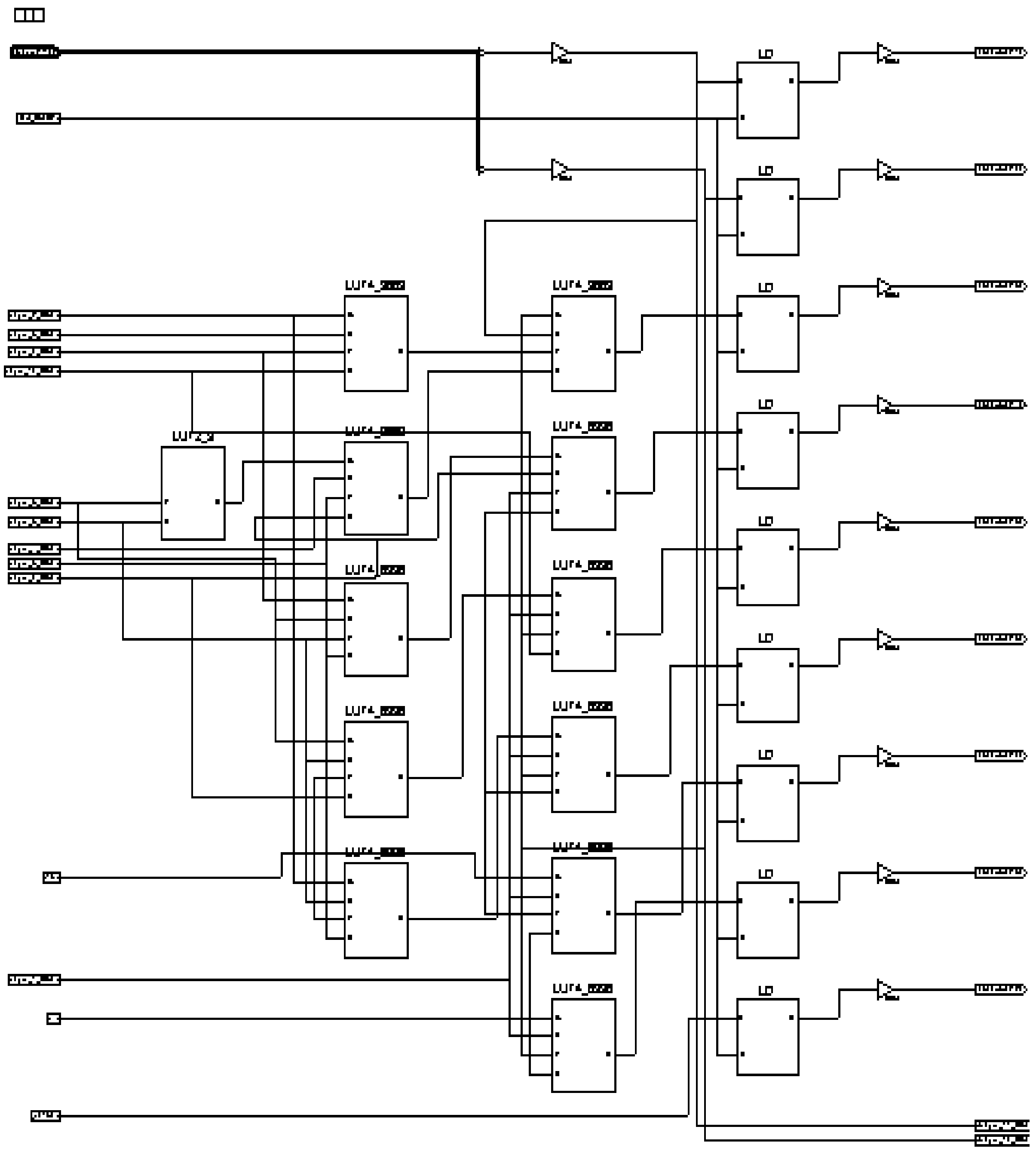


Fig 7.4 Technology Schematic of G_{24} Encoder

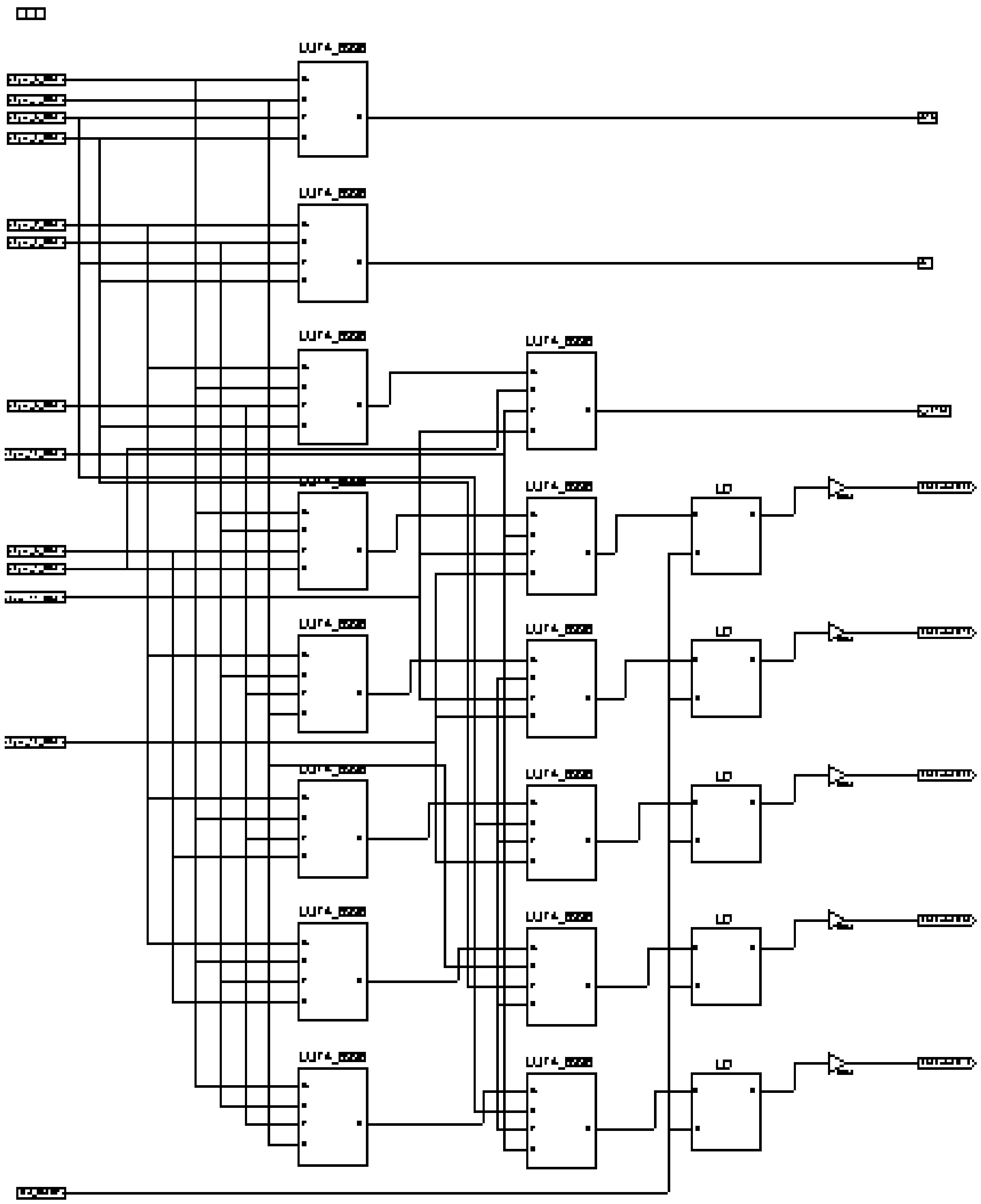


Fig 7.4 Technology Schematic of G_{24} Encoder

Schematic and Truth Table for LUT Block of Technology Schematic

The schematic for the LUT (Look up Table) is shown in figure 7.5 which represents the internal function of one Configurable logic Block in FPGA.

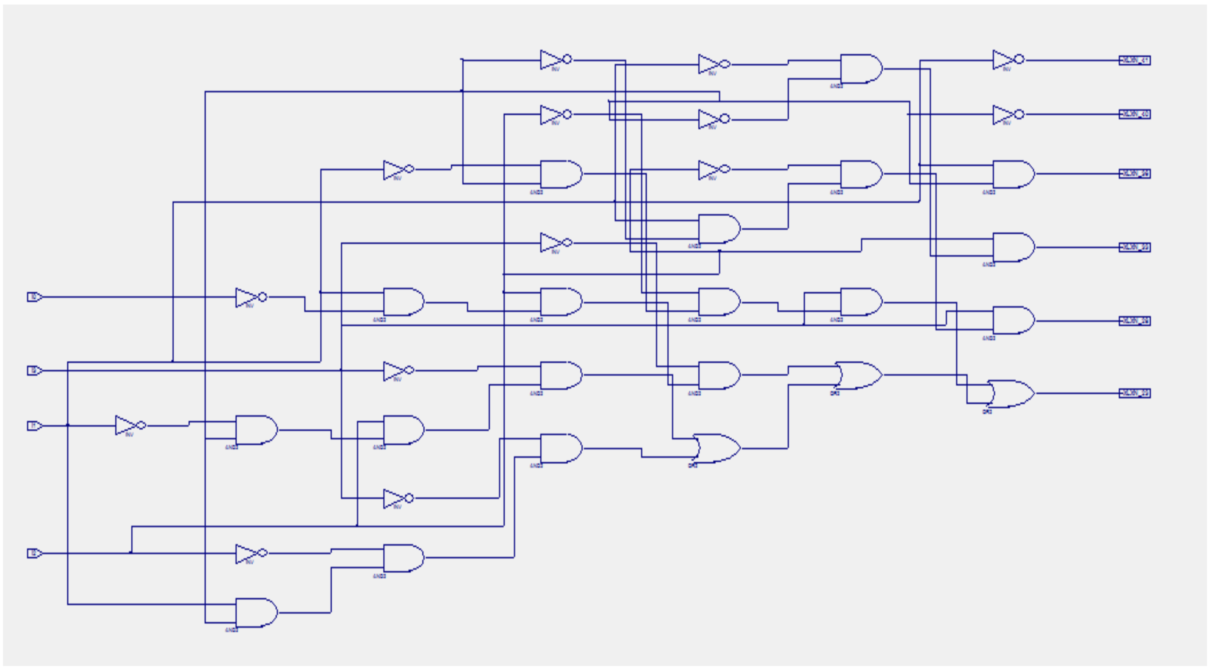
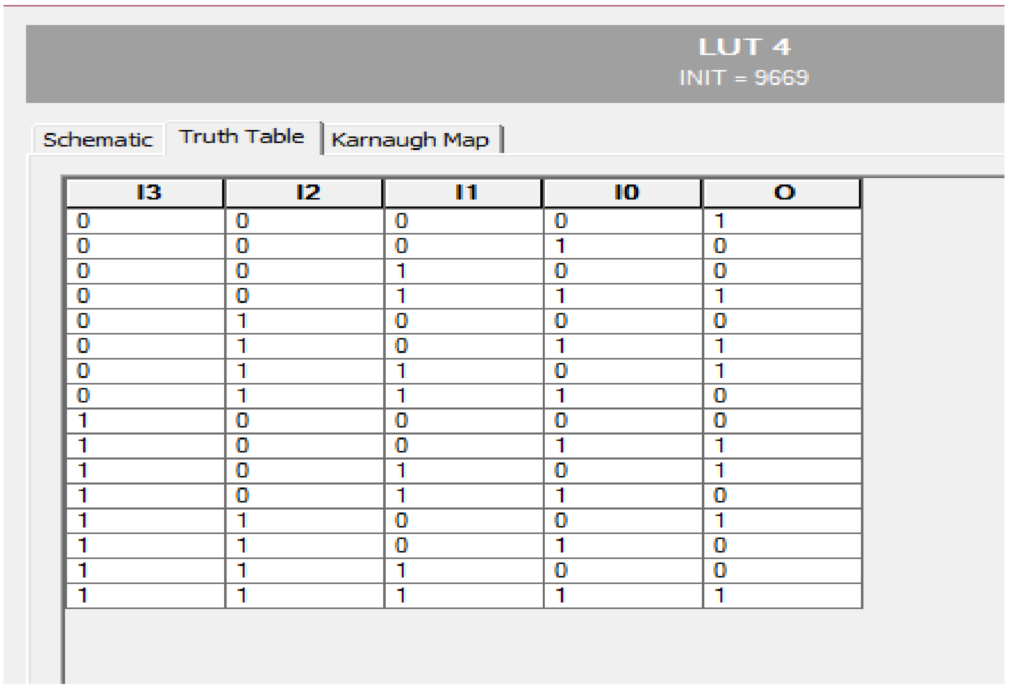


Fig 7.5 Schematic for LUT of G_{24} Encoder

The truth Table for LUT generated by the system is shown in figure 7.6.



LUT 4
INIT = 9669

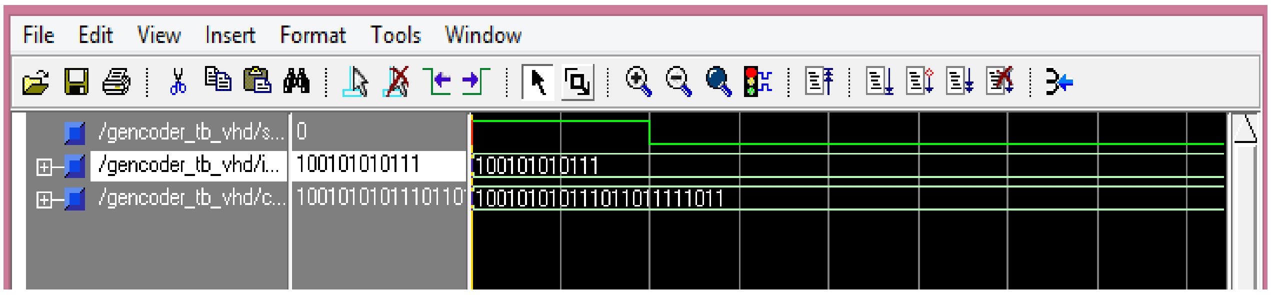
Schematic | Truth Table | Karnaugh Map

I3	I2	I1	I0	O
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Fig 7.6 Truth Table for LUT of G_{24} Encoder

7.1.3 Simulation Output for Encoder:

The testbench Output for the Golay encoder is shown in figure 7.7 while the other characteristics are discussed below in Table 7.1.



```
/gencoder_tb_vhd/s...  0
/gencoder_tb_vhd/i...  100101010111                100101010111
/gencoder_tb_vhd/c...  10010101011101101           100101010111011011111011
```

Fig 7.7 Simulation results for G_{24} Encoder

Table 7.1 Result for various parameters for G_{24} Encoder

Parameters	Value of Parameters
LUT utilization (%)	1
Latency (Clock Cycles)	24
Maximum Delay (ns)	0.972
Power Consumed(mW)	9.29
Net Skew(ns)	0.134
Memory used(MB)	258.852

7.2 G_{48} Encoder

7.2.1 RTL Schematic

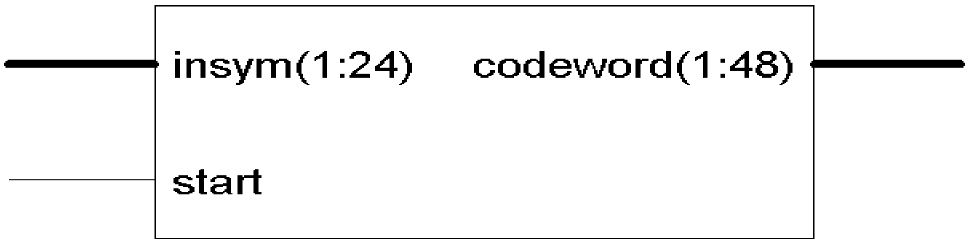


Fig 7.8 RTL Schematic of G_{48} encoder

This file shows a schematic representation of the pre-optimized design in terms of generic symbols that are independent of the targeted Xilinx device, for example, in terms of adders, multipliers, counters, AND gates, and OR gates etc.The RTL Schematic for the Golay encoder is shown in figure 7.8 which depicts there are 24 bit input message, a start to control the operation of encoder and the output has 48 bit codeword.

Internal view of RTL Schematic

The internal view of the RTL schematic provides the internal nets and data paths taken by input signal to reach to output as shown in figure 7.9. Also it can be seen that a component is instantiated again and again. This component is a multiplexer whose internal structure is shown in figure 7.10.

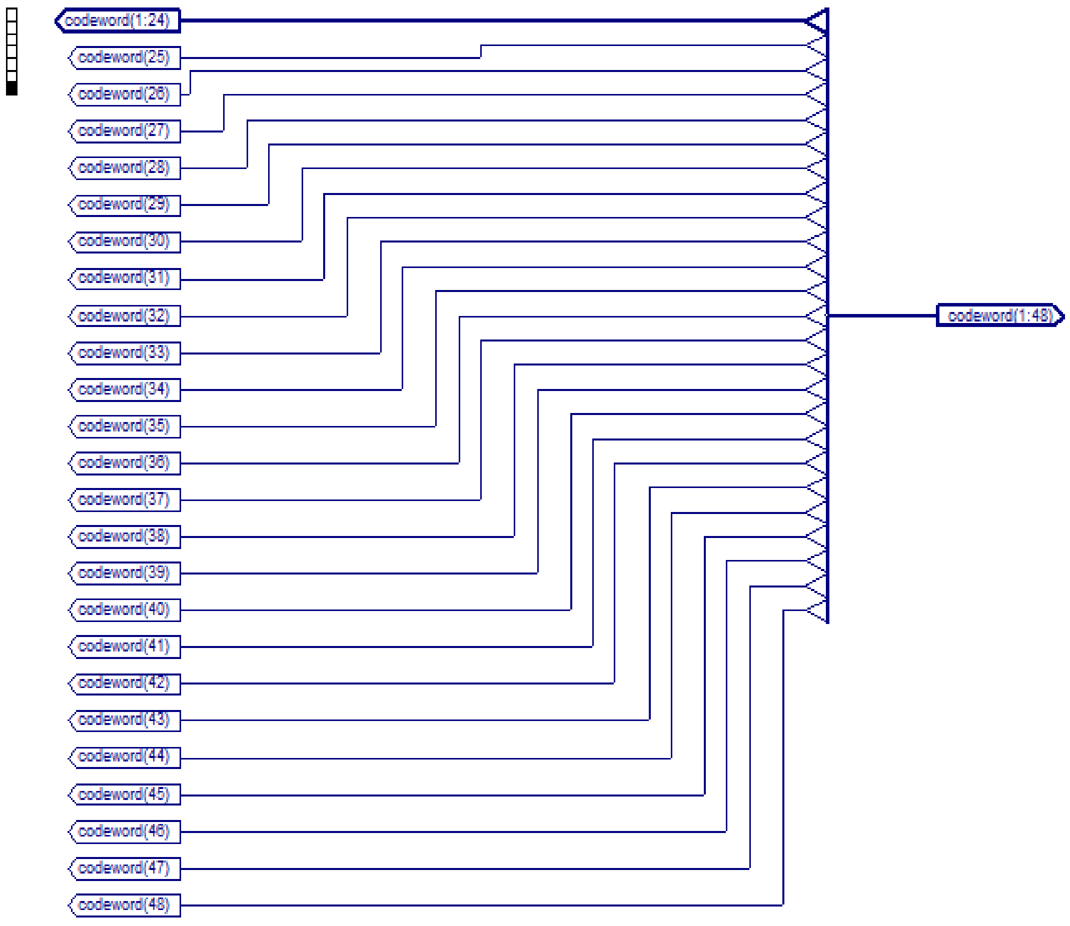


Fig. 7.9 Internal view of RTL Schematic of G_{48} encoder

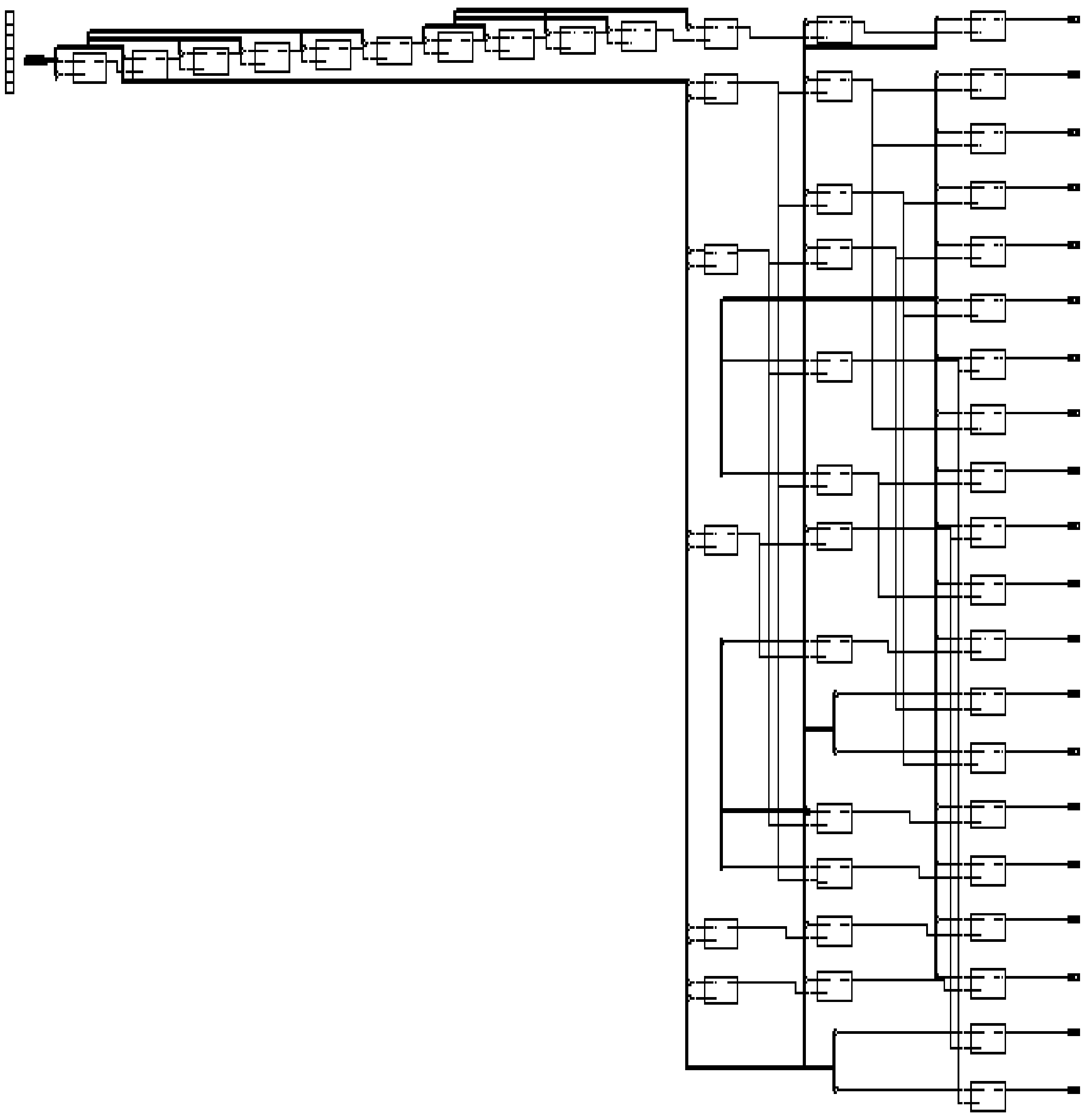


Fig. 7.9 Internal view of RTL Schematic of G_{48} encoder

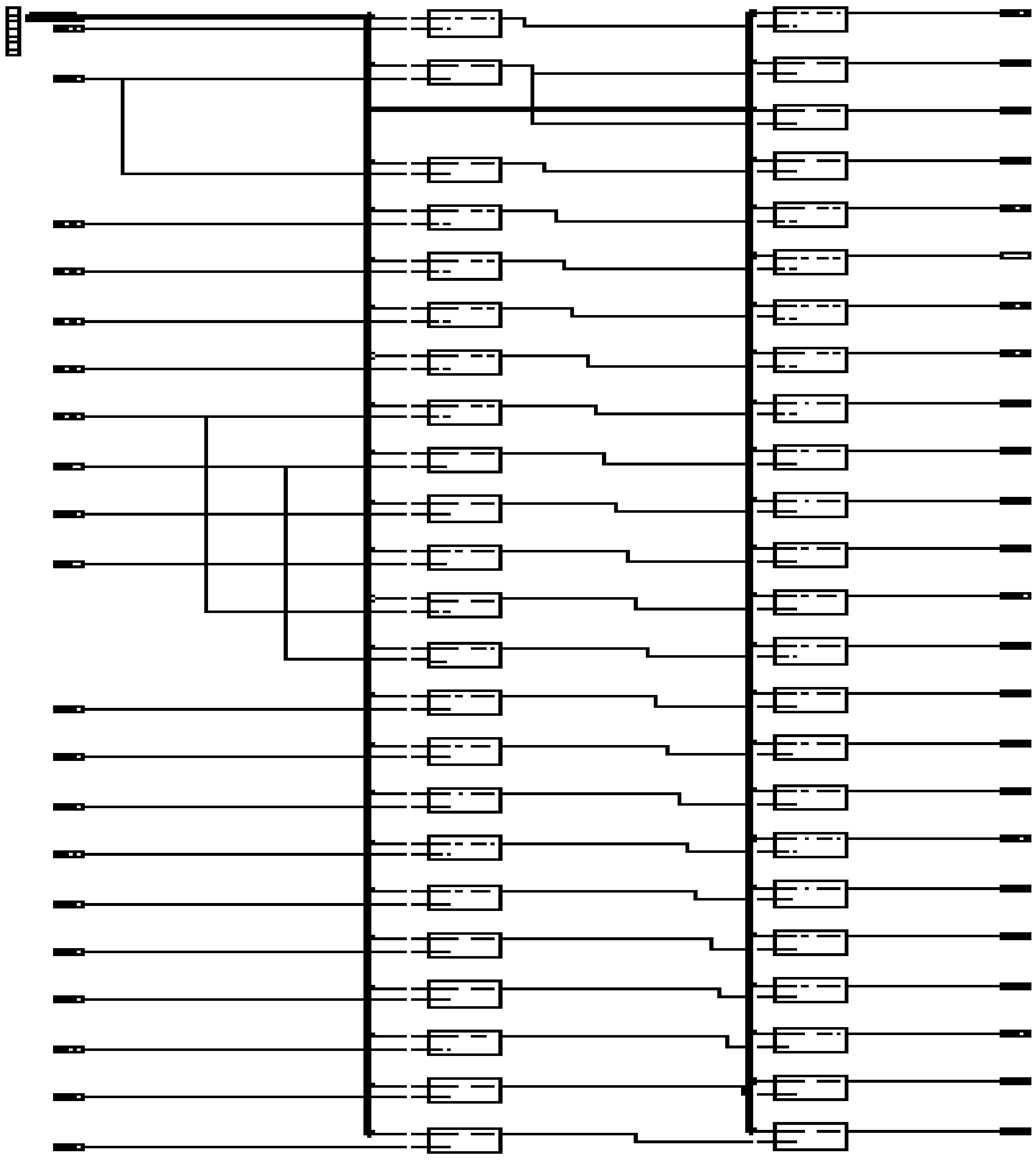


Fig. 7.9 Internal view of RTL Schematic of G_{48} encoder

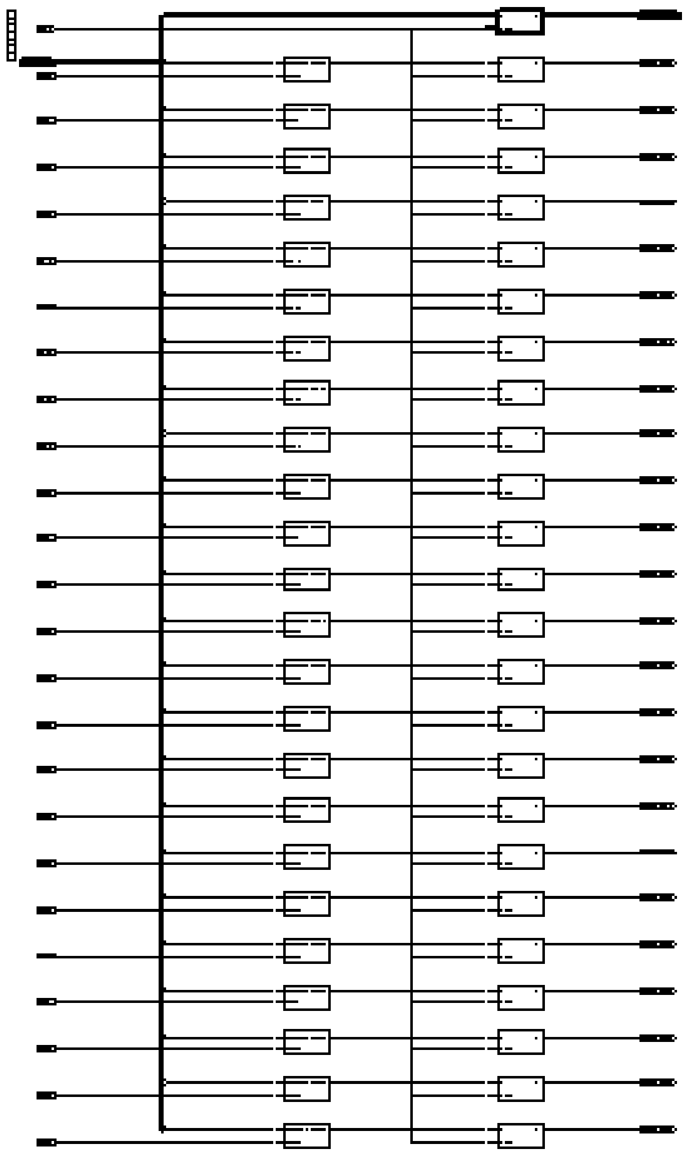


Fig. 7.9 Internal view of RTL Schematic of G_{48} encoder

Internal view of the instance

In the internal view of the instance shown in figure 7.10 we can see that a multiplexer is instantiated again and again and the mux outputs the data based on the select signal.

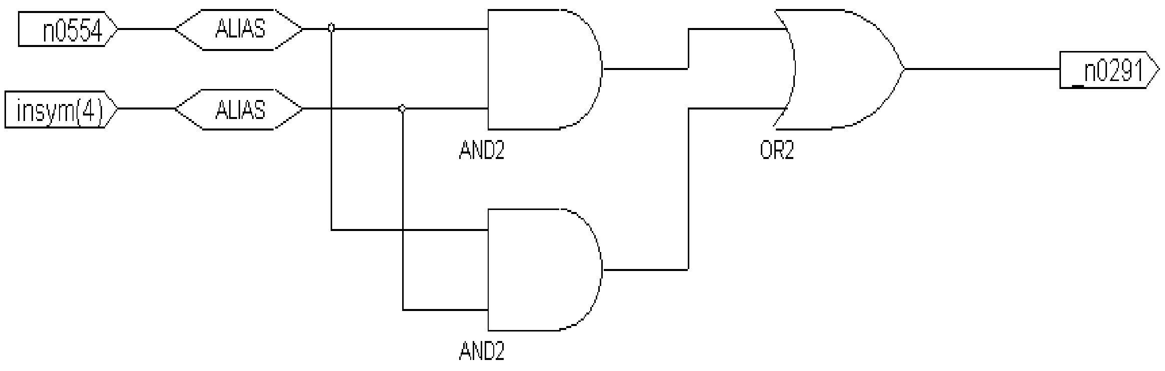


Fig 7.10 Internal view of the instance of RTL Schematic for G_{24} Encoder

7.2.2 Technology Schematic of Golay Encoder:

This file contains the detailed information of the exact elements used in FPGA chip. In technology schematic the required process is explained in the form of Flip-Flops, LUTs (Look Up Tables), input-output buffers which are an integrated part of Configurable Logic blocks in FPGA. The Technology schematic view for the Golay Encoder is shown in Figure 7.11 on the next pages.

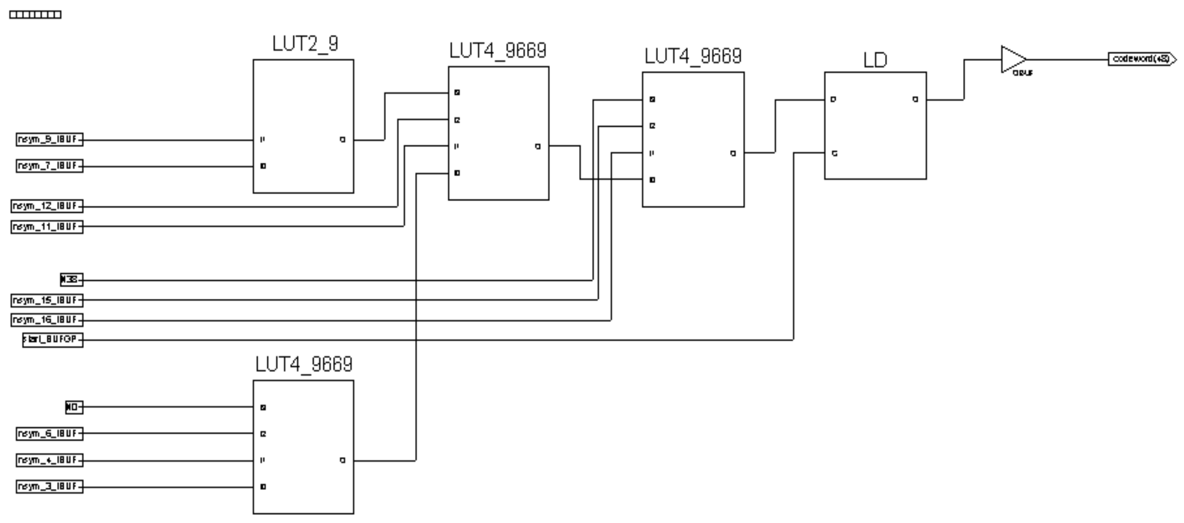


Fig. 7.11 Technology Schematic of G_{48} Encoder

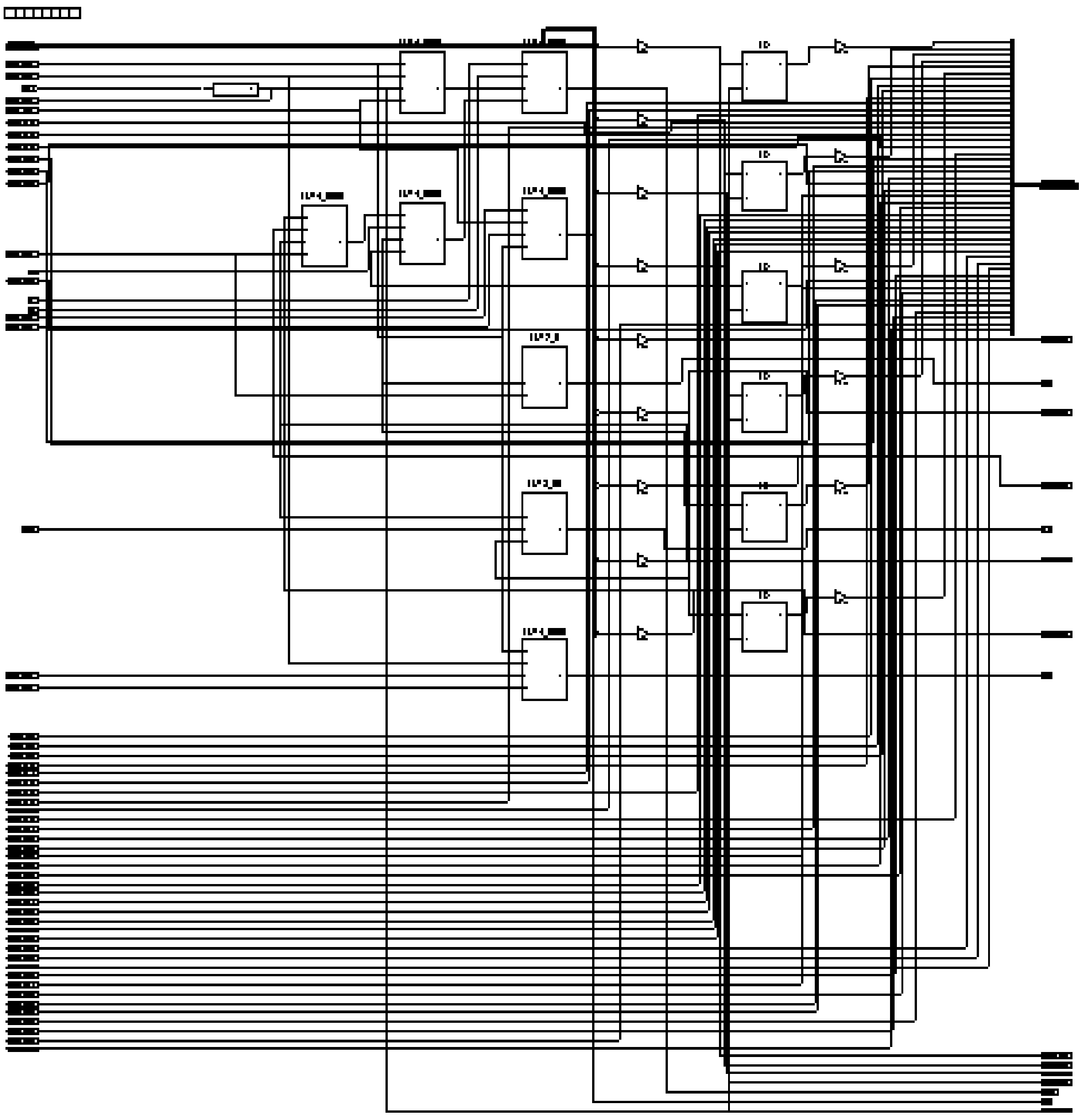


Fig. 7.11 Technology Schematic of G_{48} Encoder

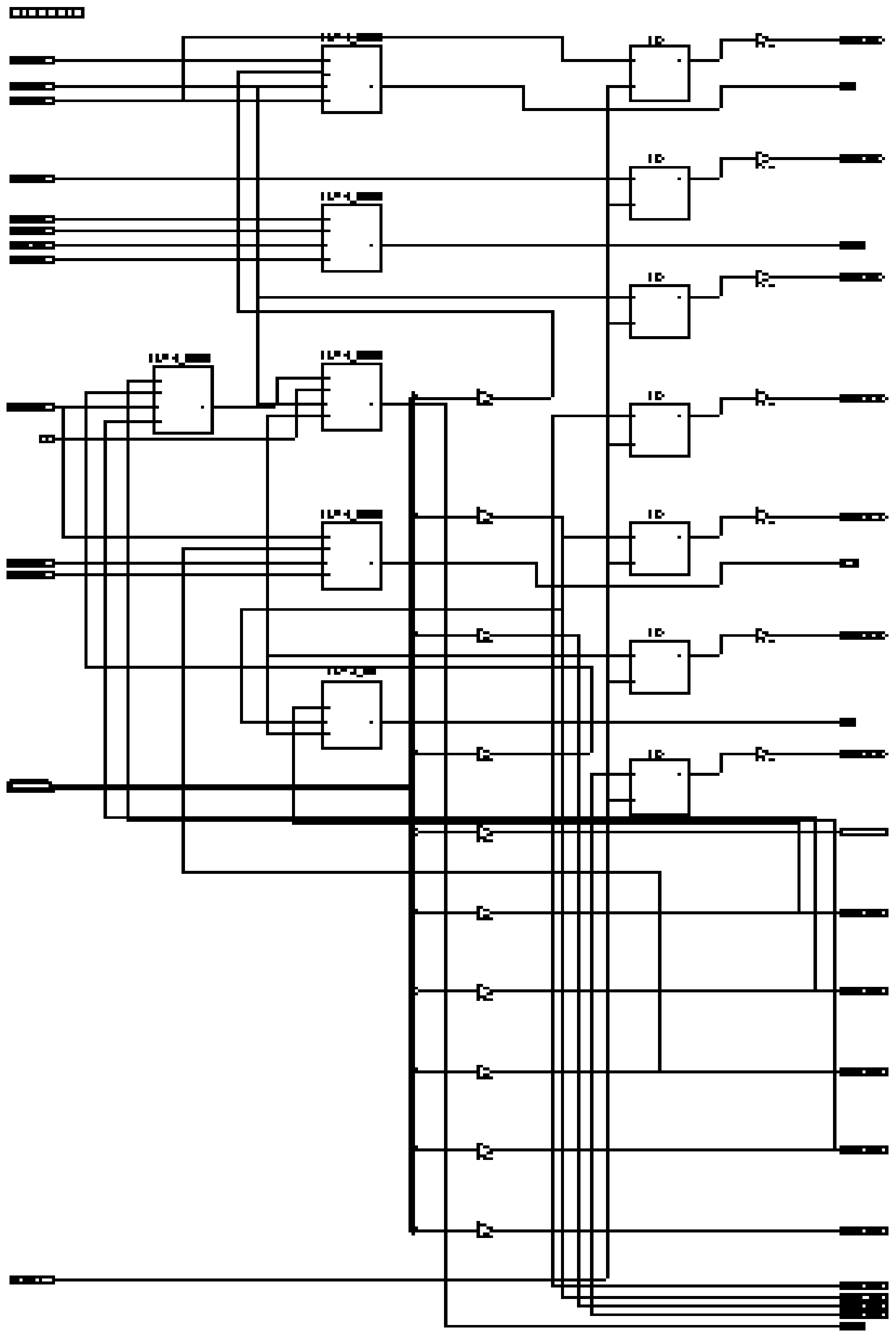


Fig. 7.11 Technology Schematic of G_{48} Encoder

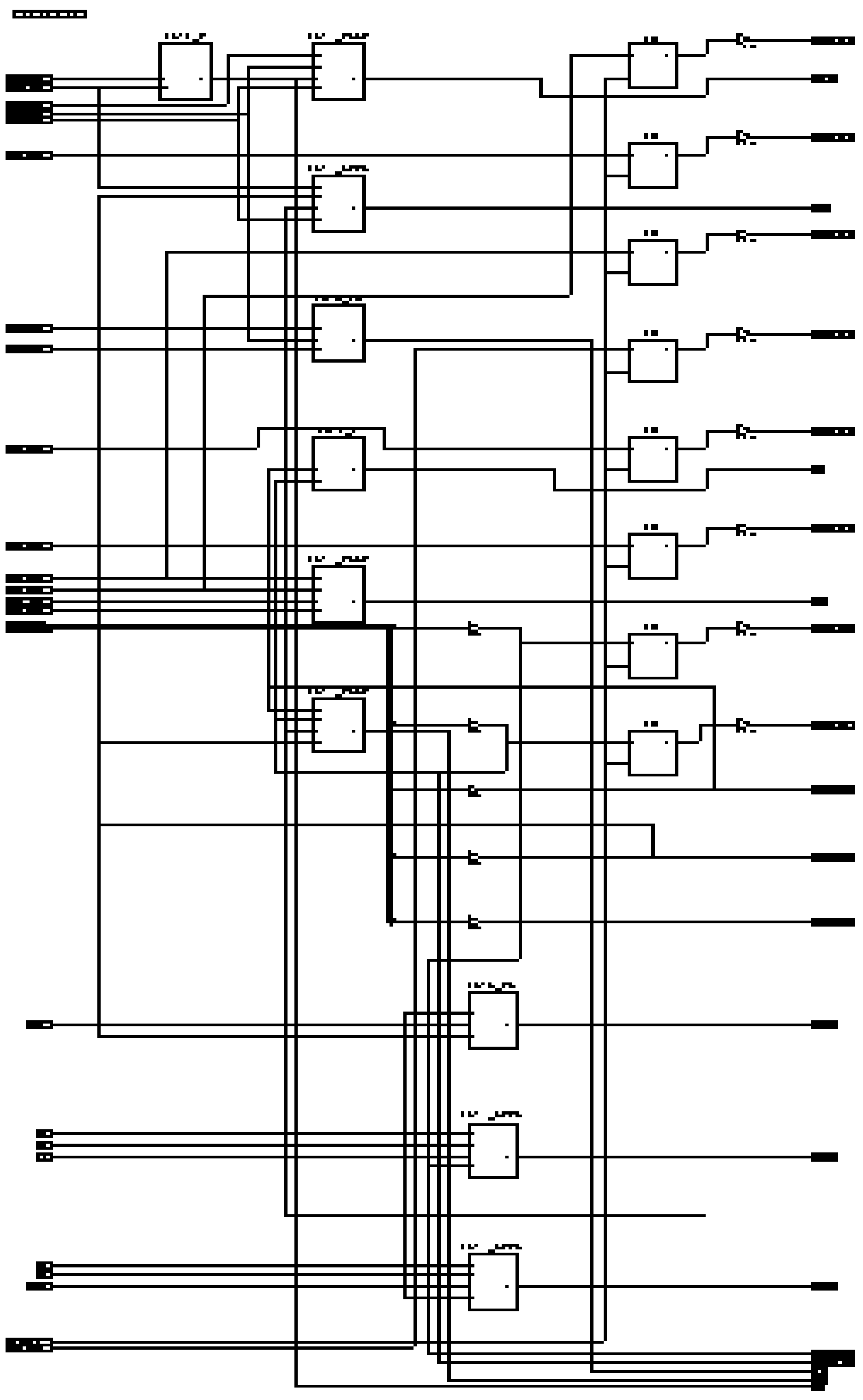


Fig. 7.11 Technology Schematic of G_{48} Encoder

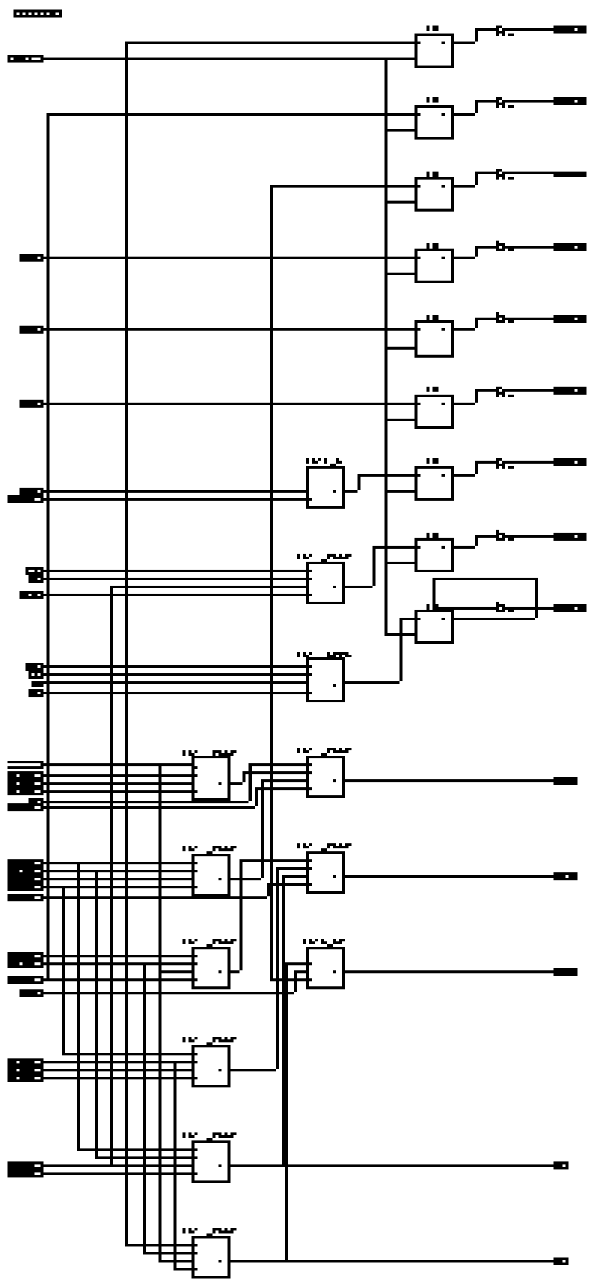


Fig. 7.11 Technology Schematic of G_{48} Encoder

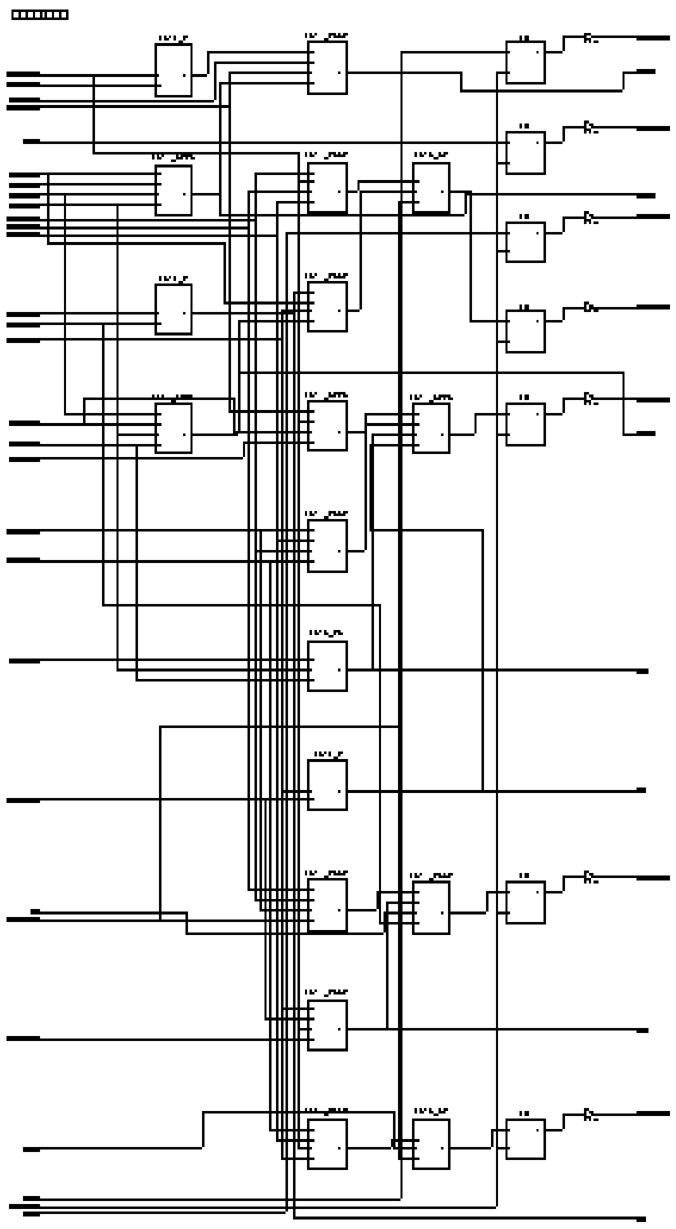


Fig. 7.11 Technology Schematic of G_{48} Encoder

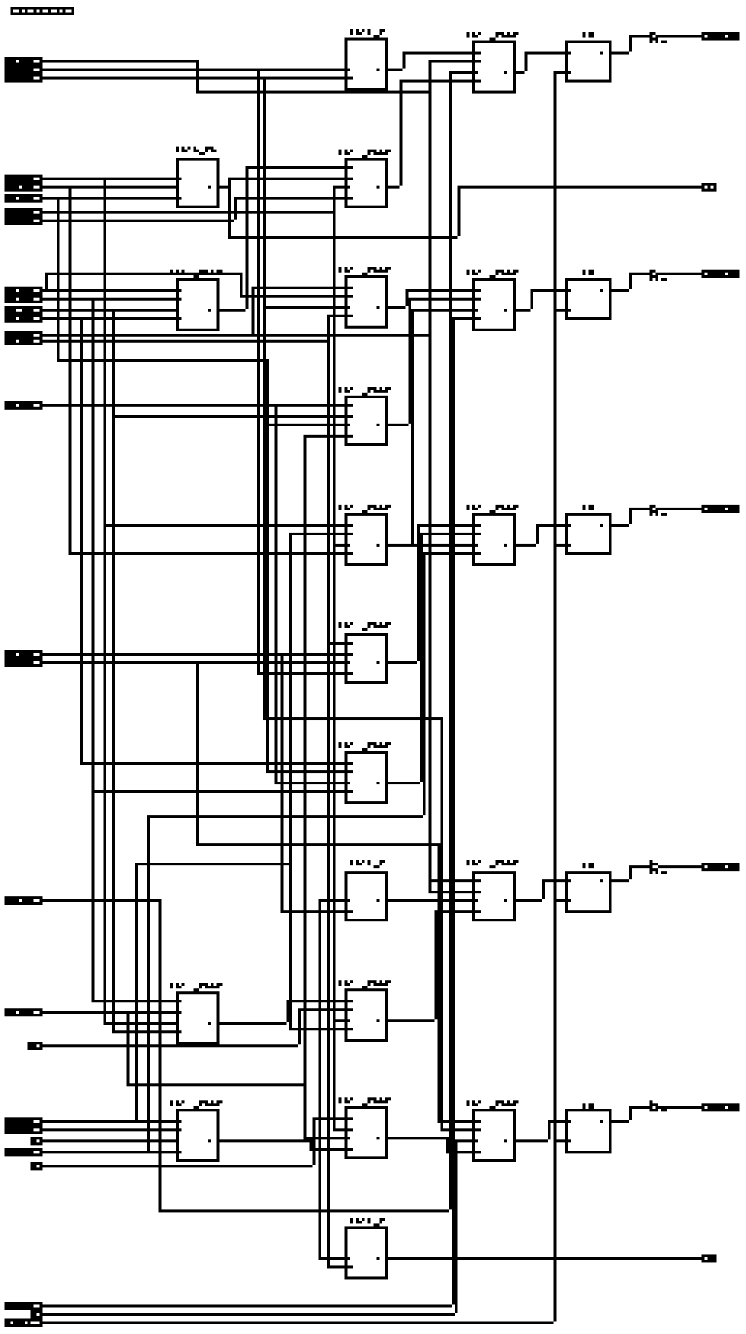


Fig. 7.11 Technology Schematic of G_{48} Encoder

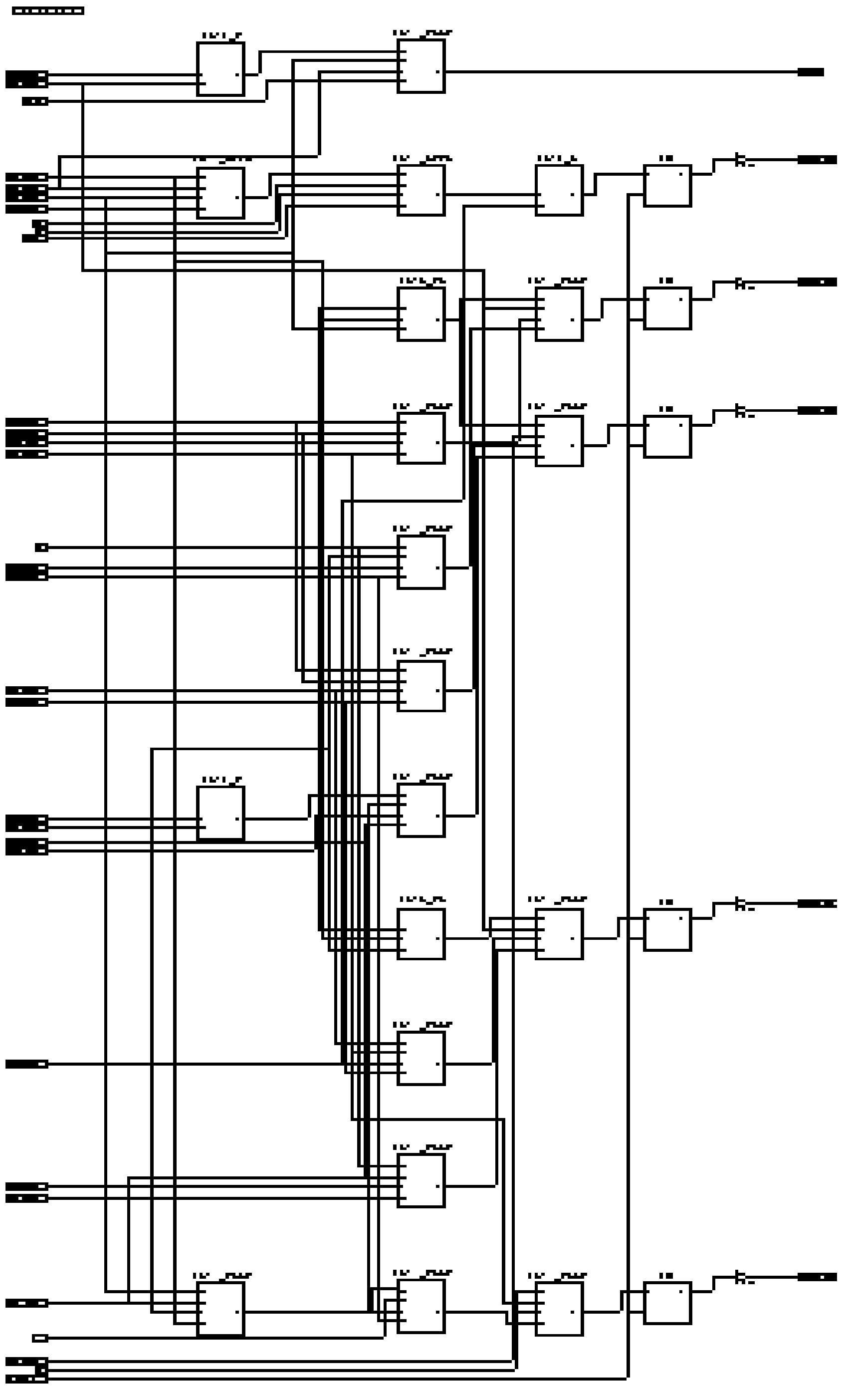


Fig. 7.11 Technology Schematic of G_{48} Encoder

Schematic and Truth Table for LUT Block of Technology Schematic

The schematic for the LUT (Look up Table) is shown in figure 7.12 which represents the internal function of one Configurable logic Block in FPGA.

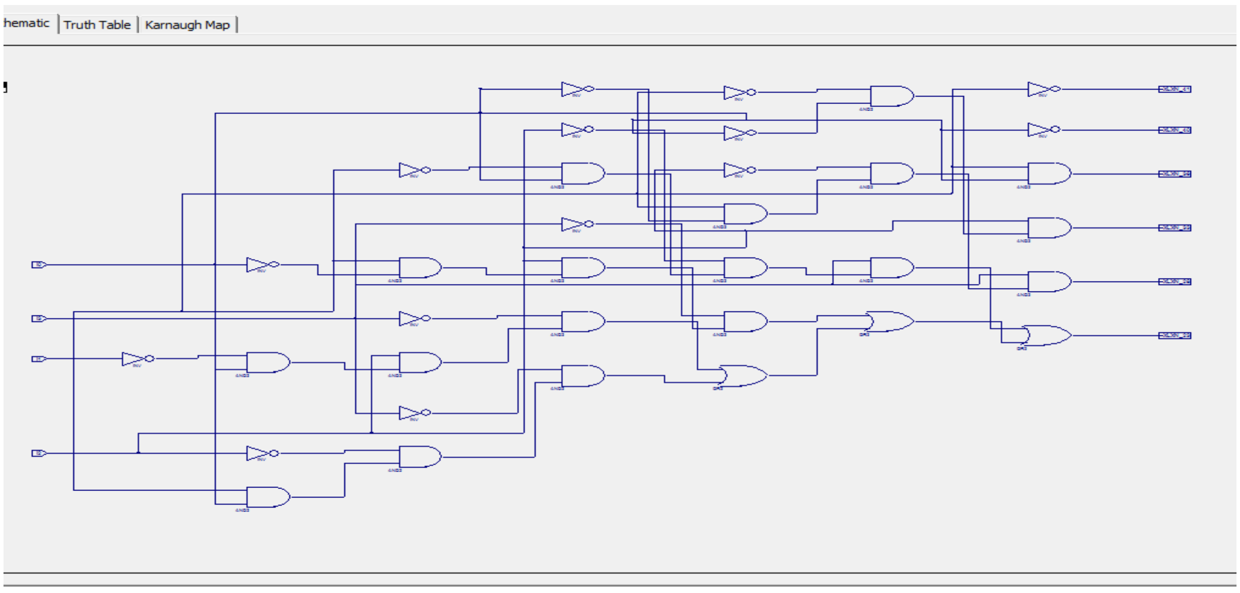
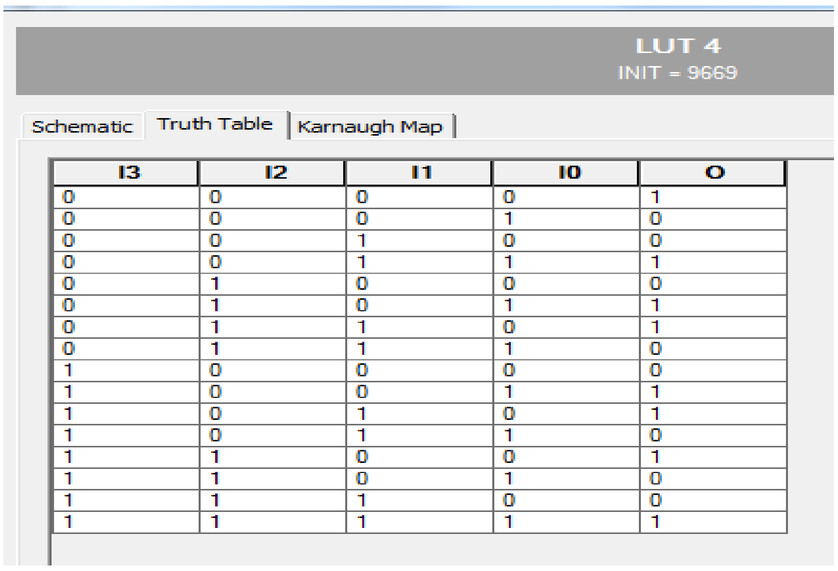


Fig 7.12 Schematic for LUT of G_{48} Encoder

The truth Table for LUT generated by the system is shown in figure 7.13.


Schematic
Truth Table
Karnaugh Map

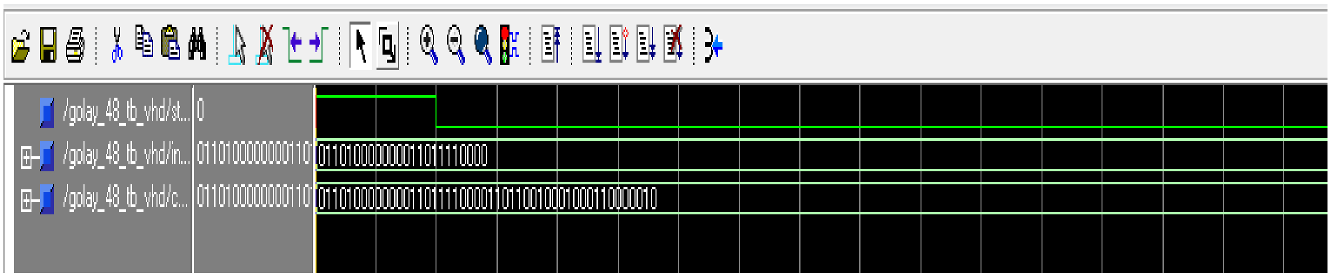
LUT 4
INIT = 9669

I3	I2	I1	I0	O
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Fig 7.13 Truth Table for LUT of G_{48} Encoder

7.2.3 Simulation Output for Encoder

The testbench Output for the Golay encoder is shown in figure 7.14 while the other characteristics are discussed below in Table 7.2.



```
/golay_48_tb_vhd/st...  0
/golay_48_tb_vhd/in...  0110100000000110  011010000000011011110000
/golay_48_tb_vhd/c...   0110100000000110  011010000000011011110000110110010001000110000010
```

Fig 7.14 Simulation results for G_{24} Encoder

Table 7.2 Result for various parameters for G_{48} Encoder

Parameters	Value of Parameters
LUT utilization (%)	25
Latency (Clock Cycles)	48
Maximum Delay (ns)	9.477
Power Consumed(mW)	10.05
Net Skew(ns)	0.134
Memory used(MB)	260.722

7.3 Conclusion

Hence binary Golay code G_{24} encoder as well as advanced Golay code G_{48} encoder is implemented using project navigator of Xilinx ISE.

Table 7.3 Comparison between G_{24} and G_{48} encoder

Parameters	G_{24}	G_{48}
LUT utilization (%)	1	25
Latency (Clock Cycles)	24	48
Maximum Delay (ns)	0.972	9.477
Power Consumed(mW)	9.29	10.05
Net Skew(ns)	0.134	0.134
Memory used(MB)	258.852	260.722

It is found that although there is a little bit increment in latency and memory used yet G_{48} codes are more sophisticated and reliable in terms of error correction. The final comparison is shown in table 7.3.

REFERENCES

1. Stephen B. Wicker, “Error Control Systems for Digital Communication and Storage”, Prentice Hall, 1995.
2. J. C. Moreira, P. G. Farrell, “Essentials of Error-Control Coding”, John Wiley and Sons, 2006.
3. B. P. Lathi, “Modern Digital and Analog Communication Systems”, Oxford University Press , 1998
4. L. H. Charles Lee, “Error Control Block Codes”, Artech House Publishers, New York, 2000.
5. R.E. Blahut, “Theory and practice of error-control codes”, Addison-Wesley, Reading, MA, 1983.
6. Shu Lin, Daniel J. Castello, “Error control coding, Fundamentals and applications”, Premtice-Hall, New Jersey, 1983.
7. I. Stewart, “Galois theory”, Chapman & Hall, London, 1973
8. S.T.J. Fenn, M.Benaissa, D. Taylor, “GF(2^m) Multiplication and division over the dual field”, IEEE Transaction on computers, vol. 45, no. 3, pp.319 - 327, March 1996.
9. J.L. Massey, J.K. Omura, “Computational method and apparatus for finite field arithmetic”, U.S. Patent Aplication, submitted 1981.
10. E. Mastrovito, “VLSI design for multiplications over finite fields GF(2^m)”, 6th Int. Conf. Applied Algebra, Algebraic Algorithms & Error-correcting codes, Rome, 1988.
11. S. Fenn, M. Benaissa, D. Taylor, “Improved algorithm for division over GF(2^m)”, Electronic Letters, Vol. 29, 4th March, 1993, pp. 469-470.
12. R.W. Hamming, “Error detecting and error correcting codes”, Bell Syst. tech. J., 29, pp. 147-160, April 1950
13. W.W. Peterson, E.J. Weldon, “Error correcting codes”, MIT Press, Cambridge, MA, 1972.
14. G. Caire, E. Biglieri, “Linear Block Codes over cyclic groups”, IEEE Transaction on Information Theory, vol. 41, no. 5, pp. 1246-1256, September 1995
15. M. D. Edwards, “Automatic Logic Synthesis Techniques for Digital Systems”, Macmillan 1992.
16. M. Damiani, J. Chih-Yuan Yang, G. De Micheli, “Optimisation of combinational logic circuit based on compatible gates”, IEEE Transaction in Computer-Aided Design of Integrated Circuit and Systems, vol. 14, no. 11, Nov. 1995.
17. Texas Instrument, “TTL - advanced low-power Schottky, advanced Schottky”, vol. 2, 1989.
18. A. Hocquenhem, “Codes corecteurs d’erreurs”, Chiffres, 2 pp. 147-156, 1959.
19. M. J. E. Golay, “Notes on digital coding”, Proc. IRE, Vol. 37, p. 657, jun. 1949.
20. M.-I. Weng and L. –N. Lee, “Weighted Erasure codec for the (24, 12) extended Golay code”, U. S. Patent 4 397 022, Aug. 2, 1983.
21. Trots, Y. Tasinkevych, and A. Nowicki, “Orthogonal Golay codes with local beam pattern correction in Ultrasonic Imaging”, IEEE Signal Processing Letters, Vol. 22, No. 10, Oct. 2015.
22. V. E. Bychkov, O. D. Mrachkovskiy, and V. I. Pravda, “Golay’s Codes Application Features in Radiolocation”, ISSN 0735-2727, Radioelectronics and Communications Systems, Vol. 51, No. 4, pp. 210–214, 2008.
23. J. Hadamard, “Rèsolution d’une question relative aux dèterminants”, Bull. Sci. Math. 17, 240-246, 1893.
24. R. E. A. C. Paley “On orthogonal Matrices”, Journal of Mathematics and Physics 12:311-320, 1933.
25. G. A. Jones and J. M. Jones, “The Legendre Symbol”

26. I. S. Reed, X. Yin, T. K. Troung, J. K. Holmes, “Decoding the (24, 12, 8) Golay code” IEE Proc. , Vol. 137, May 1990.
27. S. Sarangi, S. Banerjee, “Efficient Hardware Implementation of Encoder and Decoder”, IEEE Transaction on VLSI Systems, Vol. 23, No. 9, September 2015
28. A. Alimohammad, S. F. Fard, “FPGA-Based Bit Error Rate Performance Measurement of Wireless Systems”, IEEE Transaction on VLSI Systems, VOL.22, No. 7, July 2014
29. X.-H. Peng and P. G. Farrell, “On construction of the (24, 12, 8) Golay codes,” IEEE Trans. Inf. Theory, Vol. 52, No. 8, pp. 3669–3675, Aug. 2006.
30. B. Honary and G. Markarian, “New simple encoder and trellis decoder for Golay codes,” Electron. Lett., Vol. 29, No. 25, pp. 2170–2171, Dec. 1993.
31. J. Snyders and Y. Be’ery, “Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes,” IEEE Trans. Inf. Theory, Vol. 35, No. 5, pp. 963–975, Sep. 1989.
32. A. Neale and M. Sachdev, “A new SEC-DEC error correction code subclass for adjacent MBU tolerance in embedded memory,” IEEE Trans. Device Mater. Rel., vol. 13, no. 1, pp. 223–230, Mar. 2013.
33. Z. Ming, X. L. Yi, and L. H. Wei, “New SEC-DED-DAEC codes for multiple bit upsets mitigation in memory,” in Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst.-Chip, Oct. 2011, pp. 254–259.
34. A. Dutta and N. A. Touba, “Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code,” in Proc. IEEE VLSI Test Symposium, May 2007, pp. 349–354.
35. L.-J. Saiz-Adalid, P. Reviriego, P. Gil, S. Pontarelli, and J. A. Maestro, “MCU tolerance in SRAMs through low-redundancy triple adjacent error correction,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., to be published.
36. W. W. Peterson, “Cyclic codes for error detection,” Proceedings of the IRE, 1961.
37. X.-H. Peng and P. G. Farrell, “On construction of the (24, 12, 8) Golay codes,” IEEE Trans. Inf. Theory, Vol. 52, No. 8, pp. 3669–3675, Aug. 2006.
38. M.-H. Jing, Y.-C. Su, J.-H. Chen, Z.-H. Chen, and Y. Chang, “High-speed low-complexity Golay decoder based on syndrome weight determination,” in Proc. 7th Int. Conf. Inf., Commun., Signal Process. (ICICS), Dec. 2009, pp. 1–4.
39. S.-W. Wei, and C.-H. Wei, “On high-speed decoding of the (23, 12, 7) Golay code,” IEEE Transaction Information Theory, vol. 36, no. 3, pp. 692–695, May 1990.
40. P. Reviriego, S. Liu, L. Xiao, and J. A. Maestro, “An efficient Single and Double adjacent error correcting Parallel Decoder for the (24, 12) Extended Golay code”, IEEE Transactions VLSI systems, yet to be published.
41. P. Reviriego, J. A. Maestro, S. Baeg, S. Wen, and R. Wong, “Protection of memories suffering MCUs through the selection of the optimal interleaving distance,” IEEE Transaction Nuclear Science, vol. 57, no. 4, pp. 2124-2128, Aug. 2010.
42. P.J. Ashenden, “The designer’s guide to VHDL”, Morgan Kaufmann Publishers, Inc. 1996.
43. J. Bhaskar, “A VHDL Primer,” Prentice Hall.
44. Xilinx, “The programmable logic data book”, 1994.
45. F. Vahid, S. Narayan, D.D. Gajski, “SpecCharts: A VHDL front-end for embedded systems”, IEEE Transaction on CAD of circuits and systems, vol. 14, no. 6, June 1995, pp. 694-706
46. M.D. Edwards “Automatic logic synthesis techniques for digital systems,” Macmillan, 1992.
47. Exemplar Logic Inc. “Galileo, User manual”, Alameda, 1995.

48. E. Mastrovito, “VLSI design for multiplications over finite fields GF(2m)”, 6th Int. Conf. Applied Algebra, Algebraic Algorithms & Error-correcting codes, Rome, 1988.
49. E.R. Berlekamp, “Algebric Coding Theory,” McGraw-Hill, New York, 1968.
50. J. E. Meggitt, “Error correcting codes and their implementation,” IRE Transaction on Information Theory, vol. IT-7, pp. 232-244, October 1961.
51. R. B. J. Allenby, “Rings, Fields and Groups: An Introduction to Abstract Algebra,” Edward Arnold, London, 1983.