IMPLEMENTATION OF REED SOLOMON ERROR CORRECTING CODES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in

Electronics and Communication Engineering

By

MOHIT AGRAWAL

ROLL NUMBER: 107EC025

Department of Electronics & Communication Engineering

National Institute of Technology

Rourkela

2010-2011

IMPLEMENTATION OF REED SOLOMON ERROR CORRECTING CODES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in

Electronics and Communication Engineering

Under the Guidance of

Prof. K K MAHAPATRA

BY

MOHIT AGRAWAL(107EC025)

Department of Electronics & Communication Engineering

Rourkela

2010-2011

CERTIFICATE

This is to certify that the project report titled "IMPLEMENTATION OF REED SOLOMON ERROR CORRECTING CODES" submitted by Mohit Agrawal (Roll No: 107EC025) in the partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Electronics and Communication Engineering during session 2007-2011 at National Institute of Technology, Rourkela (Deemed University) and is an authentic work carried out by them under my supervision and guidance.

Date :

Prof. K K MAHAPATRA
Department of ECE

NIT Rourkela

769008

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude and sincere thanks to our respected supervisor Prof. K K MOHAPATRA for his guidance, insight, and support he has provided throughout the course of this work.

I would also like to thank MR. AYASKANTA SWAIN, M.TECH, In Charge, VLSI LAB, NIT ROURKELA for his kind guidance and also his full time support.

I would like to thank all faculty members, staffs and research scholars of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their extreme help throughout course.

MOHIT AGRAWAL (107EC025)

CONTENTS

ABSTRACT

In the present world, communication has got many applications such as telephonic conversations etc. in which the messages are encoded into the communication channel and then decoding it at the receiver end. During the transfer of message, the data might get corrupted due to lots of disturbances in the communication channel. So it is necessary for the decoder tool to also have a function of correcting the error that might occur. Reed Solomon codes are type of burst error detecting codes which has got many applications due to its burst error detection and correction nature. My aim of the project is to implement this reed Solomon codes in a VHDL test bench waveform and also to analyse the error probability that is occurring during transmission.

To perform this check one can start with simulating reed Solomon codes in MATLAB and then going for simulation in XILINX writing the VHDL code. The encoder and decoder design of reed Solomon codes have got different algorithms. Based on your requirements you can use those algorithms. The difference between the algorithms is that of the computational calculations between them. The complexity of the code depends on the algorithm used**.** I will be using Linear Feedback Shift Register circuit for designing the encoder.

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Overview

Communication Engineering is been the vital field of engineering in last few decades Evolution of digital communication has made this field more interesting as well as challenging. All the advancements in the field of communication are to achieve two important goals namely reliability and efficiency. In most cases reliability is given the priority over efficiency though at certain cases one is compromised for the other. Reliability of communication has an impact even in our day to day life. For example message received on our mobile phone may become unreadable if some error occurs during the transmission, or a scratch in our DVD may make it unreadable. There are wide ranges of concern in the field of digital communication. Error control issues have been addressed in this thesis.

1.2 Errors and methods

Generally communication is understood as transmission and reception of data from one place to other at some distance. If we change the reference it can also include transmission and reception of data at the same place but at a different point of time, which means storage and retrieval of data. Hence storage is also a part of communication. In any system application we come across errors either in communication or in storage. Errors in transmission are mainly because of noise, electromagnetic interferences, cross talk, bandwidth limitation, etc. In case of storage, errors may occur because of increase in magnetic flux as in case of magnetic disc or it can be spurious change of bits because of electromagnetic interferences as in case of DRAM. Hence dealing with these errors when they occur is the matter of concern. The first step is to detect the error. And after the error gets detected there are two alternate approaches to proceed.

- Automatic repeat request (ARQ) : In this approach, the receiver first detects the error and then sends a signal to the transmitter to retransmit the signal. This can be done in two

ways: (i) continuous transmission mode (ii) wait for acknowledgement. In continuous transmission mode, the data is being sent by the transmitter continuously. Whenever receiver finds any error it sends a request for retransmission. However, the retransmission can either be selective repeat or go back N step type. As the name suggests, in selective repeat those data units containing error are only retransmitted. While in go back N type, retransmission of last N data unit occurs. Next, in wait for acknowledgement mode, acknowledgement is sent by the receiver after it correctly receives each message. Hence, when not sent, the retransmission is initiated by the transmitter.

- Forward error correction (FEC): In this approach, error is both detected and corrected at the receiver end. To enable the receiver to detect and correct the data, some redundant information is sent with the actual information by the transmitter.

After being introduced to both the approaches, one should choose whether which approach is to be used. Automatic repeat request is easier but if the error occurs much frequently, then retransmission at that frequency will particularly reduce the effective rate of data transmission. However, in some cases retransmission may not be feasible to us. In those cases, Forward Error correction would be more suitable. As Forward Error Correction involves additional information during transmission along with the actual data. It also reduces the effective data rate which is independent of rate of error. Hence, if error occurs less frequently then Automatic request approach is followed keeping in mind that retransmission is feasible.

Out of the various FEC's, Reed Solomon code is one. These are block error correcting codes with wide range of applications in the field of digital communications. These codes are used to correct errors in devices such as CD's, DVD,s etc.., wireless communications, many digital subscriber lines such as ADSL,HDSL etc…

They describe a systematic way of building codes that can detect and correct multiple errors. In a block code we have k individual information bits, r individual parity bits and a total of n (=k+r) bits. However, reed Solomon codes are organized in group of bits. This group of bits are referred to as symbols. So we can say, this code has n number of symbols. Each symbol comprises of m number of bits, where

$$n(max)=2^{m}-1$$

1.3 Scope of work

As stated in the last paragraph, RS codes are used for many applications. With the objective of developing high speed RS codes, the scope includes: Study of different error detection and correction methods, implementation of RS encoder and decoder in Hardware Description Language(HDL), building fully synchronous synthesizable logic core of RS encoder and decoder to meet the requirement of almost all the standards that employ RS codes, such as CCSDS, DVB, ETIS-BRAN, IEEE802.6, G.709, IESS-308, etc.

CHAPTER 2

BASICS OF COMMUNICATION AND CODING THEORY

2.1 General communication systems

Communication is the phenomenon of transmitting information. This transmission can either be made between two distinct places (for ex, a phone call) or between two points in time, for example the writing of this thesis so that it can be read later on. We shall restrict ourselves to the study of digital communication. That is, the transmission of messages that are sequences of symbols taken from a set called alphabet.

Digital communication has become predominant in today's world. It ranges from internet, storage disks, satellite communication to digital television and so on. Moreover, any analog system can be transformed into digital data by various sampling and signal transformation methods. Typical examples include encoding music in an mp3, numerical cameras, voice recognition and many others. A digital communication system, in all its generality is represented in figure 2.1.

- The information source outputs the data to be communicated. It produces messages to be transmitted to the receiving destination. When it is a digital source, these messages are sequences of symbols taken from a finite alphabet.
- The transmitter takes the source data as input and produces an associated signal suited for the channel. Transmitter aims to achieve one or more of the followings.
    - A maximum of information transmission per unit of time. This is directly connected to data compression techniques taking advantage of the statistical structure of the data.

- To ensure a reliable transmission across the noisy channel. In other words, to make it fault tolerant to errors introduced by the channel. This is typically done by adding structured redundancy in the message.
- To provide message confidentiality. This typically involves encryption which hides or scrambles the message so that unintended listeners cannot discern the real information content from the message.

- The physical channel is the medium used to transmit the signal from the source to the destination. Examples of channels conveying information is conveyed over space like telephone lines, fiber-optic lines, microwave radio channels. . . Information can also be conveyed between two distinct times like for example by writing data on a computer disk or a DVD and retrieving it later. As the signal propagates through the channel, or on its storage place, it may be corrupted. For example, the telephone lines suffer from parasitic currents, waves are subject to interference issues, a DVD can be scratched... But these perturbations are regrouped under the term of noise. The more noise, the more the signal is altered and the more it is di-cult to retrieve the information originally sent. Of course, there are many other reasons for errors like timing jitter, attenuation due to propagation, carrier offset... But all these perturbations lie beyond the scope of this thesis.

Figure 2.1 : General Communication System

- The receiver ordinarily performs the inverse operation done by the transmitter. It reconstructs the original message from the received signal.
- The destination is the system or person for whom the message is intended.

2.2 Needs of Communication Systems

As was said in the previous section, the transmitter can have several roles together. To compress data, to secure data, to make it more reliable and lastly to transmit it as signals suited for the physical channel. Compressing data is also called source coding; it consists of mapping sequences of symbols in the original data stream to shorter ones. This is done based on the statistical distribution of the original data: the most frequent sequences are mapped to shorter ones while rare sequences are mapped to longer ones. By doing this, the resulting sequences are on average shorter, i.e. sequences with fewer symbols. On the opposite, in order to make the sequence of symbols robust to errors, redundancy is added to it. This is called channel encoding and consists of mapping shorter sequences to longer ones so that if a few symbols are corrupted the original data can nevertheless be found back. More often we need both of these. This seems contradictory since one reduces the number of sent symbols while the other increases it. However, it is not really. The source coding reduces the redundancy of unstructured data which would not provide protection if symbols were corrupted. For example, despite knowing that a message contains on average 99% of zeros, you cannot know which bits were corrupted when sending the message as it is. On the opposite, channel coding adds structured data to improve protection against such errors during the transmission. By taking the compressed message and repeating three times each bit, you can decode correctly up to one error per three bits introduced. One could wonder if a technique performing both in a single step could be more efficient than doing it sequentially. It turns out that performing source and channel coding sequentially tends to be optimal when the treated sequence length tends to infinity. This is known as the source-channel coding separation theorem and is one of the results of Shannon's ground breaking work. For finite sequence length, such joint encoding techniques are still a subject of research. Until now, we spoke only about symbols and about mapping sequences of symbols to other sequences of symbols with better properties. However, the physical channel does not, technically speaking, transmit symbols but signals (waves, voltage, etc...). We however assume a one-to-one mapping between symbols and signals which is done by a modulator to map a symbol to the corresponding signal and a demodulator mapping back a received signal to a received symbol, or information about the likelihood of each potential symbol. Notice that by

separating the source and channel coding, an encryption module can also be conveniently inserted between both. Putting all together, the obtained refined communication model is illustrated in figure 2.2 in the bottom of this page.

All three modules: compression, encryption, channel coding are of course optional and not necessarily included in a communication system. The part of interest for us is channel encoding and decoding. As a side note, but important, one consequence of source coding or encryption is that any of them tends to produce equal probability sequences of symbols. This argument will support an important assumption for the input of the channel encoder later on. Moreover, even if these modules are not present and nothing is known about the source's output, this is still the best assumption that can be made. The main part of interest for us is the channel encoder and decoder and it can now be isolated from the remaining system. Lastly, the modulator and demodulator constitute the glue between the transmitter, the physical channel and the receiver. The channel can be considered as the modulator, the physical channel and the demodulator together, providing an abstract model having as input and output symbols. However, the received signals, altered by noise, may not match any of the sent ones. So either it is mapped to other symbols in a bigger alphabet or some threshold decision must be used to decide which symbol of the original alphabet it should be. This is seen in more details in the section about channels

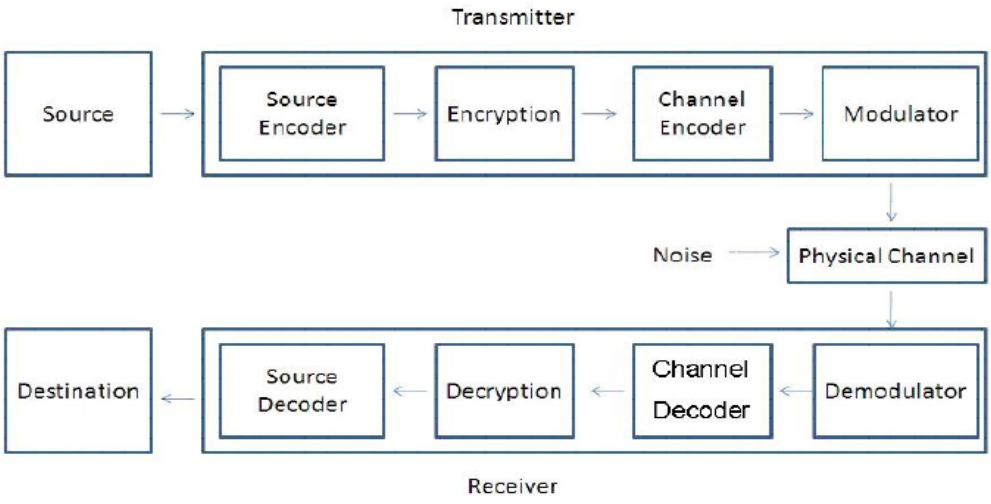

Fig 2.2 Needs of Communication Systems

2.3 Channel Encoder

The role of the encoder is to add structured redundancy to sequences of symbols. There are different ways of doing this but the biggest classes of encoders are by far block encoders. They work by cutting the data stream in blocks (of symbols) of fixed size and encoding these blocks one after another. Such a block, a finite sequence of symbols, is also known as a word.

Definition 2.1. A word W = (w_1,……,w_m) of length m over an alphabet A is an ordered sequence of m symbols taken from A.

The data block is called the message word. By hypothesis, it has a fixed length, say k. The encoder maps each such word to a longer one, called code word, also of fixed size, say n. A reasonable assumption is that both words, the message and the encoded code word, are defined over the same alphabet Aq. The encoder is thus a mapping from words of length k to words of length n where n > k. Words can be seen as points in a space over the alphabet. In this light, the encoder is a mapping of points in A^k_q to points in A^n_q . If the alphabet has q elements, then **qk** message words are mapped onto **qk** from **qn** possible words. The set of all code words is thus only a subset of An q and this set forms the code An q which is the image of the encoder function. On a closer look, it turns out that what is of interest is not the mapping, but the code C itself.

2.4 Basics of Coding

As we have discussed in previous chapter a code is the set of all the encoded words, the code words that an encoder can produce. That means when actual set of data is encoded it becomes a code.

Definition 2.2: A q_ary code C of length n is a set of code words in A_n, where A is an alphabet of q symbols. The size of the code, noted |C|, is the number of code words in the code.

Example 2.1: A ternary code of length 5 is the following set:

01201

00210

12012

20021

12120

10021

The ternary alphabet used is A = {0, 1, 2} and the code size is |C| = 6 (It contains 6 code words). Code words can be seen as vectors in the space A_n where the i^{th} symbol is the i^{th} coordinate. To compare words, the space A_n can be equipped with a convenient metric called the Hamming distance.

Definition 2.3: The Hamming distance between two words x; yϵA_n is the number of co-ordinates in which symbols differ.

$$d_H(x; y) = |i \mid x_i \neq y_i|$$

Example 2.2: d_H(ASHUTOSH; PARITOSH) = 4

$d_H(001010; 010010) = 2$

Minimal distance is a basic and important property of a code as it is directly related to the error correcting capability of the code. For nearest neighbor decoding, in order for the code to provide unambiguous decoding of the received code words up to e errors, it is necessary and sufficient that the minimal distance be at least d = 2e+1 . This follows directly from the triangle inequality since no received word w can lie at distance less than or equal to e to two code words if they are all at a distance d = 2e + 1.

CHAPTER 3

ERROR DETECTION AND CORRECTION METHODS

Being introduced to the concepts of coding it is easier to explain different error detection and correction schemes. Error correction may be avoided at certain cases where retransmission is feasible and effective. But error detection is a must in all cases. Hence first some important error detection schemes are discussed. Then classification of forward error correction is described followed by some important properties of Forward Error Correction codes.

3.1 Error Detection Schemes

- Repetition scheme -In repetition scheme the actual data is sent more than once. In receiver these repeated data are compared. Difference in these repeated data indicates error. Though this is the simplest way of detecting error it is not popular as it reduces the rate of actual message transmission.
- Parity scheme -In parity scheme all the data sets are assigned a particular parity i.e. either even or odd. In the receiver parity of received data is checked. If it does not satisfy the assigned parity, it is found to be in error. It is effective only for odd number of errors. It cannot detect even number of errors as even number of errors will leave the parity unchanged.
- Checksum Scheme -In this scheme a checksum is calculated in the transmitter and sent with the actual data. In receiver checksum is calculated and compared with the received checksum. A mismatch is an indication of error. If data and checksum both are received with error then the detection may not be possible.

- Cyclic Redundancy Check scheme - In this scheme the message is interpreted as polynomial and is divided by a generator polynomial. Then the reminder of the division is added to the actual message polynomial to form a code polynomial. This code polynomial is always divisible by the generator polynomial. This property is checked by the receiver. If failed to satisfy this property the received codeword is in error. It is complex but efficient error detection scheme.

- Hamming distance Based Check scheme -This scheme is basically parity based scheme but here parity of different combination of bits are checked for parity. It can detect double errors and can correct single errors.

- Polarity scheme - In this scheme the actual message along with its inversion format. In receiver it is checked whether two sets are inverse of each other. If not it is an indication of error. It is not that popular as the code occupies double the bandwidth for the actual message. Moreover if corresponding bits in the data and is inverse are in error then it will not be able to detect the error.

3.2 Classification of Forward Error Correction Codes

There are several ways of classifying the forward error correction codes as per different characteristics.

- Linear Vs Non linear- Linear codes are those in which the sum of any two valid code words is also a valid code word. In case of non linear code the above statement is not always true.
- Cyclic Vs Non-Cyclic- Cyclic code word are those in which shifting of any valid code word is also a valid code word. In case of non-circular code word the above statement is not always true.
- Systematic Vs Nonsystematic- Systematic codes are those in which the actual information appears unaltered in the encoded data and redundant bits are added for

detection and correction of error. In nonsystematic code the actual message does not appear in its original form in the code rather there exists one mapping method from the data word to code word and vice versa.

- Block Vs convolutional- The block codes are those in which one block of message is transformed into on block of code. In this case no memory is required. In case of convolutional code a sequence of message is converted into a sequence of code. Hence encoder requires memory as present code is combination of present and past message.
- Binary Vs Non binary- Binary codes are those in which error detection and correction is done on binary information i.e. on bits. Hence after the error is located, correction means only flipping the bit found in error. In Non binary code error detection and corrections are done on symbols, symbols may be binary though. Hence both the error location and magnitude is required to correct the symbol in error.

3.3 Properties of Forward Error Correction Codes (FEC's)

Four basic properties on basis of which a particular Forward Error Correction code can be selected are described here.

- Coding gain: Expressed in dB the difference between E_b/N_0 needed to achieve a given Bit Error Probability with and without encoding.
- Coding rate: It is the ratio of the number of message bits transmitted to the number of total bits transmitted (k/n).
- Power penalty: It is a result of sending a large constellation which include the extra parity or check bits of the error correcting code. Power penalty = (2B -1)/ (2b -1).
- Coding Complexity: It is the complexity involved in encoding and decoding. It increases design time as well as latency.

3.4 Comparison between some Forward Error Detection Techniques

After getting familiarized with classification and properties of forward error correction codes some of the error detection and correction codes are explained and compared in this section.

- Hamming Code - In a hamming encoder parity bits are inserted into the message bits. These parity bits are decided so as to impose a fixed parity on different combinations of data and parity bits. In decoder those combinations are checked for that fixed parity. Accordingly decoder parity bits are set. Binary equivalent of this combination decides the location of the error. Then that particular bit is flipped to correct the data. Hamming code is a single error correction code. Double errors can be detected if no correction is attempted.

- Berger Code - Berger code is a unidirectional error detection code. It means it can only detect error either '1' flipped to '0' or '0' flipped to '1' but not both in a single code. If designed for detecting errors with '1' flipped to '0' then binary equivalent of number of 0s in the message is sent along with the message. Similarly when design for detecting '0' flipped to '1' error binary equivalent of the number of 1s in the message are sent along with the message. Decoder compares the number of 0s or 1s as per the design with the binary equivalent received. Mismatch between the two indicates the error. It can be used where error is expected to be unidirectional.

- Constant weight code -In this code a valid code word always have a constant weight. It means number of 1s in a valid code word is fixed. Hence any variation in this is an indication of error. It is simple but not efficient way of encoding as multiple errors can cancel out each other.

- M out of N code - In an M out of N encoder message is mapped to a N bit code word having M number of 1s in it. The N-M bits of message are appended with additional M number of bits which are used to adjust the number of 1s in the code. If the

message consists of no 1s in it then all the M bits are set to '1'. It is also not an efficient code in terms of coding rate.

- Erasure code - Erasure means error when its location is known in advance from previous experience. Erasure code is able to correct such errors. In this type of code the decoder circuit does not need an error locator as it is already known. Hence only error magnitude is calculated by the decoder to correct the erasure.

- Low Density Parity check code - Low density parity check code is a linear block code. The message block is transformed into a code block by multiplying it with a transform matrix. Low density in the name implies low density of the transform matrix. That means number of 1s in the transform matrix is less. It is the best code as far as the coding gain is concerned but encoder and decoder design is complex. Mainly used in Digital Video Broadcasting.

- Turbo Code - It is a convolutional code. Encoding is simple convolutional encoding. It is defined by (n, k, l) turbo code where n is the number of input bits, k is the number of output bits and l is the memory of the encoder. Decoding is done in two stages. First one is soft decoding stage then a hard decoding stage. It has very good error correcting capability i.e. coding gain. The main drawback is that it has low coding rate and high latency. Hence it is not suitable for many applications. But in case of satellite communication as the latency due to the distance itself is so high this additional latency is negligible. Hence it is used mainly in satellite communication.

- Reed Solomon Code - Reed Solomon code is a linear cyclic systematic non-binary block code. In the encoder Redundant symbols are generated using a generator polynomial and appended to the message symbols. In decoder error location and magnitude are calculated using the same generator polynomial. Then the correction is applied on the received code. Reed Solomon code has less coding gain as compared to LDPC and turbo codes. But it has very high coding rate and low complexity. Hence it is suitable for many applications including storage and transmission.

CHAPTER 4

MATHEMATICAL THEORMS

This chapter is to get a thorough knowledge of the groups and the fields used in mathematics that will also be used further for RS encoding & decoding.

4.1 Groups

The most fundamental algebraic structure is a group. Following definitions elaborate it all.

Definition 4.1: A group (G; *) is a set G with a binary operation * : G*G→ G satisfying the following 3 axioms: [4]

Associativity: For all $a; b; c \epsilon G : (a * b * c) = (a * b) * c = a * (b * c)$.

Identity element: There is an element $e \epsilon G$ such that for all $a \epsilon G : e * a = a * e = a$.

Inverse element: For each $a \epsilon G$, there is an element $b \epsilon G$ such that $a * b = b * a = e$, where e is the identity element.

As is stated, a group is simply a set of elements having a neutral and inverses, noted a^{-1} or -a depending on the situation. A group is said to be commutative, or Abelian, if for $a; b \epsilon G$ we have $ab = ba$.

Example 4.1: The set of even integers, noted 2Z is a commutative group under addition. On the opposite, the set of odd integers is not a group at all since the sum of two odd integers does not

lie in the set of odd integers. The set of invertible square matrices R^{nxn} forms a non-commutative group under multiplication where the neutral element is the identity matrix.

Let S be an ordered set. Consider G: the set of bijections $\alpha: S \rightarrow S$. That is, the elements of G are permutations of the ordered set S. The set G is a group under composition. The inverse is simply the permutation which maps S to its original state and the neutral is the permutation which does not change anything.

It is straightforward to show that the cancellation laws hold in groups: $ab = ac \Rightarrow b = c$. Since it is sufficient to pre multiply both sides by a^{-1} to obtain the equality. Moreover the cancellation laws implies the following theorem.

Theorem 4.1: The map $x \rightarrow ax : G \rightarrow G$ is a bijection. [4]

Proof: To prove that the map is bijective, we will prove that it is both injective and surjective. The cancellation law directly implies that it is injective since $ax_1 = ax_2 \Rightarrow x_1 = x_2$. To show surjectivity, let us show that for every $b \in G$ there exists a value for $x \in G$ so that $ax = b$, this is simply $x = a^{-1}b$. By taking a subset of elements satisfying the properties of a group, we obtain a subgroup.

Definition 4.2: A subgroup (S; *) of (G; *) satisfies the following axioms:

$S \subset G$ and $S \neq \varnothing$

if $a; b \in S$ then $a * b \in S$

if $a \in S$ then $a^{-1} \in S$

So that(S; *) is a group by itself with the same neutral element.

Example 4.2: The set of even integers 2Z is a subgroup of the integers Z. [4]

By performing the operation (which can be the addition, the multiplication,...) on a subgroup by an element of a group, we obtain a coset.

Definition 4.3: Let (H; *; e) be a subgroup of (G; *; e) both commutative. A coset of H in G is a set of the form

$aH = \{ ah \mid h \in H\} = Ha$ for some fixed $a \in G$.

Example 4.3: Let us take as group the integers Z under addition. Then 5Z is a subgroup of it and 5Z + 1 is a coset. Notice however that a coset is generally not a group. Indeed, it is easy to see that if a =2 H, then the coset aH has no neutral element and is therefore not a group. Despite of this, cosets have some nice properties, illustrated in the following theorem. [4]

Theorem 4.2: Let H be a subgroup of G. If C is a coset of H and $a \in C$ then C = aH Cosets form a partition on G When G is finite, all cosets have the same number of elements. [4]

Proof: To show the first point, let a 2 C = cH so that a = ch for some h. By multiplying by h^{-1}, we have $c = ah^{-1}$. On one hand, any element $x \in C$ can be expressed as $x = ch' = ah^{-1}h' \in aH$; thus C c aH. On the other hand, any element $y \in aH$ can be expressed as $y = ah'' = chh'' \in C$ and thus aH c C. Combining both, for any $a \in C$ we have C = aH. To show that the cosets form a partition on G, we must show that no element can lie in two distinct cosets.If an element a lies in C and C0, then C = aH = C' and thus the sets are either equal or disjoint. Lastly, that the cosets have the same number of elements as H follows directly from the fact that $x \rightarrow ax$ is a bijection on G.

Example 4.4: Let us take 3Z, the multiples of 3, as subgroup of the integers Z. The cosets are:

{….;-6;-3; 0; 3; 6; ….}

{….. ; -5;-2; 1; 4; 7; …..}

{…;-4;-1; 2; 5; 8; ….} [4]

and the properties in the previous theorem are easily checked. Despite most examples were applied to numbers, it should be kept in mind that these numbers form a particular instance of the problem. In particular, we will see in the next chapter on linear codes that these form a group.

4.2 Fields

Fields are mathematical structures behaving with the same rules as usual arithmetic and close to our everyday intuition. A field is a set where operations like addition, subtraction, multiplication or division subject to the laws of commutativity, distributivity, associativity and other "usual" properties.

Definition 4.5: A field is a set F or F with two operations + and . such that: [4]

(F,.+) is a commutative group;
(F*,.) where F* = F \ {0}, is a commutative group;
the distributive law holds.

Notice that a field is a ring, by definition, with the additional property that every element has an inverse under multiplication. Some common examples of fields are R, C and Q. Indeed, every axiom is straightforward to verify. However, the set of integers Z is not a field. Indeed for (Z*, .) to be a group, any of its element should have an inverse under multiplication. This is clearly not the case since no integers have an inverse in this set except -1 and 1.

Moreover, it should be noted that no flat assumptions are made about operations like subtraction or division. These two can respectively be seen as undoing the operation, i.e. a + (-b) and $a.b^{-1}$. Other familiar properties are not assumed by default but easily proved. A field is finite when it contains a finite number of elements, referred to as the size of a field and F_q denotes a field of size q. This notation turns out to be unambiguous because all fields of the same size are isomorphic (identical via a renaming of elements). One of the most common finite fields used in coding theory is the binary field encountered before. The addition and multiplication of this field are illustrated in table 5.2 and correspond to XOR and AND binary operations.

There can be fields with more elements. Let us take as an example Z=5Zwhich is the ring of the integers modulo 5. The field axioms can easily be verified. Thus the ring Z=5Z is also a field whose addition and multiplication tables are illustrated in table 5.4.

This leads us to the question whether all such rings are also fields or not. It can be observed that although the ring of integers modulo 5 is a field, all such rings are not. For example, consider Z=6Z. It does not form a group under multiplication since the elements 2, 3 and 4 have no

multiplicative inverses. For 2 and 4 it is straightforward to see that the product of it with any number results in an even integer and they have therefore no inverse. For 3, it is easy to see as well. By multiplying 3 by any odd integer is equivalent to 3 and by an even integer results in 0.

Theorem4.3: Z= nZ is a field if and only if n is prime. [4]

However, they are by no means the only finite fields. They are the simplest and sufficient to illustrate and understand how arithmetic in fields can be done. The key thing to remember is that because of the field axioms, elements can be unambiguously added, subtracted, multiplied and divided, in opposition to previous algebraic structures covered. By taking the alphabet as a field, say F, then F_n is a vector space (which is itself a group). This vector space structure is practical and will help us further explaining finite field and Reed Solomon Code.

4.2.1 Glaois Fields

In order to understand the encoding & decoding principles of Reed Solomon code, one should have a thorough knowledge of finite fields known as Glaois fields. For any prime number p there exists a glaois field GF(p) which contains exactly p elements. It is quite possible to extend GF(p) to an extension of p^m elements making it GF(p^m) where m is a non zero positive integer.

Besides the numbers 0 and 1, there are additional unique elements in the extension field that will be represented with a new symbol α. Each nonzero element in GF(2^m) can be represented by a power of α. An infinite set of elements, *F*, is formed by starting with the elements {0, 1, α}, and generating additional elements by progressively multiplying the last entry by α, which yields the following:

$$F = \{0,\ 1,\ \alpha,\ \alpha 2,\ \cdots,\ \alpha\ j,\ \cdots\} = \{0,\ \alpha 0,\ \alpha 1,\ \alpha 2,\ \cdots,\ \alpha\ j,\ \cdots\}\ [1]$$

To obtain the *finite* set of elements of GF(2^m) from *F*, a condition must be imposed on *F* so that it may contain only 2*m* elements and is closed under multiplication. The condition that closes the

set of field elements under multiplication is characterized by the irreducible polynomial shown below:

$$(\alpha^{(2m-1)}) + 1 = 0$$

or

$$\alpha^{(2m-1)} = 1 = \alpha^{0} \qquad [1]$$

Each of the 2^m elements of the finite field, GF(2^m), can be represented as a distinct polynomial of degree m - 1 or less. The degree of a polynomial is the value of its highest-order exponent. We denote each of the nonzero elements of GF(2^m) as a polynomial, a_i (X), where at least one of the m coefficients of a_i (X) is nonzero. Consider the case of m = 3, where the finite field is denoted GF(2^3). Figure 7 shows the mapping (developed later) of the seven elements $\{\alpha^i\}$ and the zero element, in terms of the basis elements $\{X_0, X_1, X_2\}$. Since we know that that $\alpha_0 = \alpha_7$, there are seven nonzero elements or a total of eight elements in this field. Each row in the Figure 7 mapping comprises a sequence of binary values representing the coefficients $a_{i,0}$, $a_{i,1}$, and $a_{i,2}$. One of the benefits of using extension field elements $\{\alpha^i\}$ in place of binary elements is the compact notation that facilitates the mathematical representation of non binary encoding and decoding processes. Addition of two elements of the finite field is then defined as the modulo-2 sum of each of the polynomial coefficients of like powers.[1]

$$\alpha^i + \alpha^j = (a_{i,0} + a_{j,0}) + (a_{i,1} + a_{j,1})X + \ldots + (a_{i,m-1} + a_{j,m-1})X_{m-1} \qquad [1]$$

4.2.2 Primitive Polynomial

A class of polynomials called primitive polynomials is of interest because such functions define the finite fields GF(2^m) that in turn are needed to define R-S codes. The following condition is necessary and sufficient to guarantee that a polynomial is primitive. An irreducible polynomial $f(X)$ of degree m is said to be primitive if the smallest positive integer n for which $f(X)$ divides X^n+1 is $n = 2m - 1$. Note that the statement A divides B means that A divided into B yields a

nonzero quotient and a zero remainder. Polynomials will usually be shown low order to high order. Sometimes, it is convenient to follow the reverse format.

Example : Verify whether the polynomial is primitive or not. f(X) = $1 + X + X^4$ [1]

We can verify whether this degree $m = 4$ polynomial is primitive by determining whether it divides $X^n + 1 = X^{(2m-1)} = X^{15} + 1$, but does not divide $X^n + 1$, for values of n in the range of $1 \leq n < 15$. It is easy to verify that $1 + X + X^4$ divides $X^{15} + 1$, and after repeated computations it can be verified that $1 + X + X\ 4$ will not divide X^{n+1} for any n in the range of $1 \leq n < 15$. Therefore, $1 + X + X^4$ is a primitive polynomial.

m	
3	$1+X+X^3$
4	$1+X+X^4$
5	$1+X^2+X^5$
6	$1+X+X^6$
7	$1+X^3+X^7$
8	$1+X^2+X^3+X^4+X^8$
9	$1+X^4+X^9$
10	$1+X^3+X^{10}$
11	$1+X^2+X^{11}$
12	$1+X+X^4+X^6+X^{12}$
13	$1+X+X^3+X^4+X^{13}$

m	
14	$1+X+X^6+X^{10}+X^{14}$
15	$1+X+X^{15}$
16	$1+X+X^{3+}X^{12}+X^{16}$
17	$1+X^3+X^{17}$
18	$1+X^7+X^{18}$
19	$1+X+X^{2+}X^5+X^{19}$
20	$1+X^3+X^{20}$
21	$1+X^2+X^{21}$
22	$1+X+X^{22}$
23	$1+X^5+X^{23}$
24	$1+X+X^{2+}X^7+X^{24}$

TAB 4.1 Some primitive polynomials [1]

CHAPTER 5

REED SOLOMON CODES

5.1 Overview

Reed Solomon codes are non binary cyclic error correcting codes. They describe a systematic way of building codes that can detect and correct multiple errors. In a block code we have k individual information bits, r individual parity bits and a total of n (=k+r) bits. Rather reed Solomon codes are organized in group of bits. These group of bits are referred to as symbols. So we can say this code has n number of symbols. This symbol comprises of m number of bits, where

$$n=2^m-1$$

These n symbols form a code word. Out of these n symbols k are information symbols where n-k are parity symbols. These codes are used to transfer data between any two medium. There are possibilities that error may occur due to any disturbance in the traverse channel. So we need to have a design tool specified for encoding and then decoding after correcting signals in the receiver side. Reed Solomon codes can be the tool discussed above. The error correcting capability of the code is defined as t, where

$$t= [(n-k)/2]$$

where [x] represents the greater integer function. The code can correct only half of the number of parity symbols because for every error one parity symbol is used to locate the error and the other is used to correct it. The advantage behind reed Solomon code is that it can correct burst errors. However if erasures are present then only one parity symbol is used to correct error as the location of error is already known. The code rate is defined as R_c, where

$$R_c= k/n$$

5.2 Reed Solomon Codes: Properties

The properties of Reed-Solomon codes make them especially well-suited to applications where errors occur in bursts. This is because

- It does not matter to the code how many bits in a symbol are in error—if multiple bits in a symbol are corrupted it only counts as a single error. Alternatively, if a data stream is not characterized by error bursts or drop-outs but by random single bit errors, a Reed-Solomon code is usually a poor choice. More effective codes are available for this case.
- Designers are not required to use the "natural" sizes of Reed-Solomon code blocks. A technique known as "shortening" can produce a smaller code of any desired size from a larger code. For example, the widely used (255,251) code can be converted to a (160,128) and not transmitting them. At the decoder, the same portion of the block is loaded locally with binary zeroes.[10]
- Its capability to correct both burst errors and erasures makes it the best choice for the designer to use it as the encoding and decoding tool.

5.3 Reed Solomon Encoding

Reed Solomon encoding can be done in many ways. Some of them being

- Encoding by polynomial division
- Encoding in the frequency domain
- Encoding using Cauchy cell matrix method

However, the basic encoding principle is going to be the same. First of all, the information symbols are being transferred to the output using generator polynomial or Cauchy cell matrix and then the parity bits are added to these information symbols.

5.4 Reed Solomon Decoding

RS decoding is done in three levels. First one being, syndrome calculation that tells us whether an error has occurred during the transmission of data. The second step includes error location which tells us where the error is present, and the third one is the error evaluation which corrects the error. However, the decoder has the capability of correcting t errors where n=k+2t.
When a code word is decoded, there are three possible outcomes:

- If $2s + r < 2t$ (s errors, r erasures) then the original transmitted code word will always be recovered.[2]
- The decoder will detect that it cannot recover the original code word and indicate this fact.[2]
- The decoder can incorrectly decode and recover an incorrect code word without any indication.[2]

5.5 Reed Solomon Error Probability

Reed Solomon codes are mainly used for burst error correction. However the code has its own error correcting capability. So, the error probability plays a crucial role in saving our time detecting and correcting the error. Let us assume that the code can correct 4 error symbols in an (255,251) RS code. A maximum of 32 bits of error can be corrected. So, if we can calculate the bit error rate properly and then manage the syndrome calculation part as if the decoder calculates more than 32 bits of error, then send a signal that decoder cannot correct. Therefore plotting the bit error probability (P) against the SNR will help. We can have a range of SNR for which the error can be corrected. However the range will include many parts like the percentage of probability that the signal will get detected. Figure 5.1 shows the plot between bit error probability and SNR. The code is for a random of about 255 symbols where each symbol contains 8 bits being transmitted. These 255 symbols form a code word. And there are about 500

such code words. However the range estimation for different capability of correcting errors can be calculated.

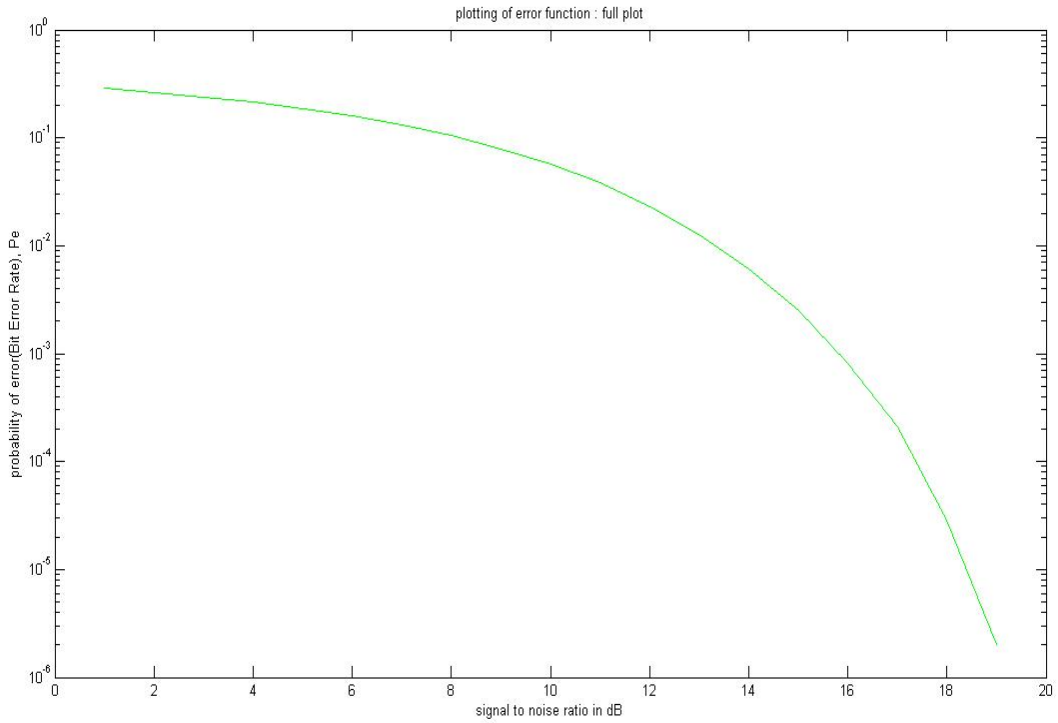

Figure 5.1: Graph between SNR and Bit error rate (BER)

To analyse the error probability graph, 255 symbols corresponds to m=8, therefore each symbol will contain 8 bits. And for different error correcting capability different range of SNR can be deduced. So, for different error correcting capability the number of error bits can be found out from the graph. For example for 2t=4; t=2 16 error bits can maximum be corrected. So a range of SNR can be found out using the graph. Similarly for 2t=8; t=4 at maximum 32 bits can be corrected. This range of SNR would be quite bigger than the last range. This result helps us to find out whether the decoder can correct the received signal or not which saves our lot of time and effort.

CHAPTER 6

REED SOLOMON ENCODER AND DECODER

In this chapter, the design of reed Solomon encoder and decoder in MATLAB and VHDL will be discussed. In MATLAB a (15,7) RS code is designed and then the output is taken. And in VHDL a (255,251) has been designed.

6.1MATLAB SIMULATION

Our main aim for simulating the same in MATLAB was to understand the phenomenon as how the signal is being transmitted, what would happen if the signal has some error and up to what extent, the decoder can detect and correct errors. In MATLAB, a random symbol of integers was taken as input. These random symbols were then encoded using RS encoder. And some random symbol noise was added to the input, these symbols were then received at the decoder end. Now at the decoder end, the decoder can correct up to 2t symbols. Our basic aim was to see if any error would have occurred in the parity symbols then what result comes out. The reed Solomon decoder actually corrects symbols up to 2t symbols from the n number of symbols. It never matters to him whether the errors are in parity symbols or the information symbols. After correcting the error, however the decoder takes the redundant bits out which were generated while encoding the symbols. An output with errors in parity symbols (first one parity symbol error and then two parity symbol error) will be shown in the results and discussions chapter. However an comparison between the decoded and received data and the encoded data structure will be shown in figures. This comparison will be shown in RS (255,251) code. The analysis was done using RS (15,7) because it is easy to produce an error in an parity symbol in a 15,7 code than in an 255,251 code. Figure shows the comparison between the encoded data and the message symbols which shows that 251 information symbols are the same as the old one while the 4 parity symbols are added in the encoded data. And the second Figure shows the comparison between the decoded data and the message symbols when the errors are corrected.

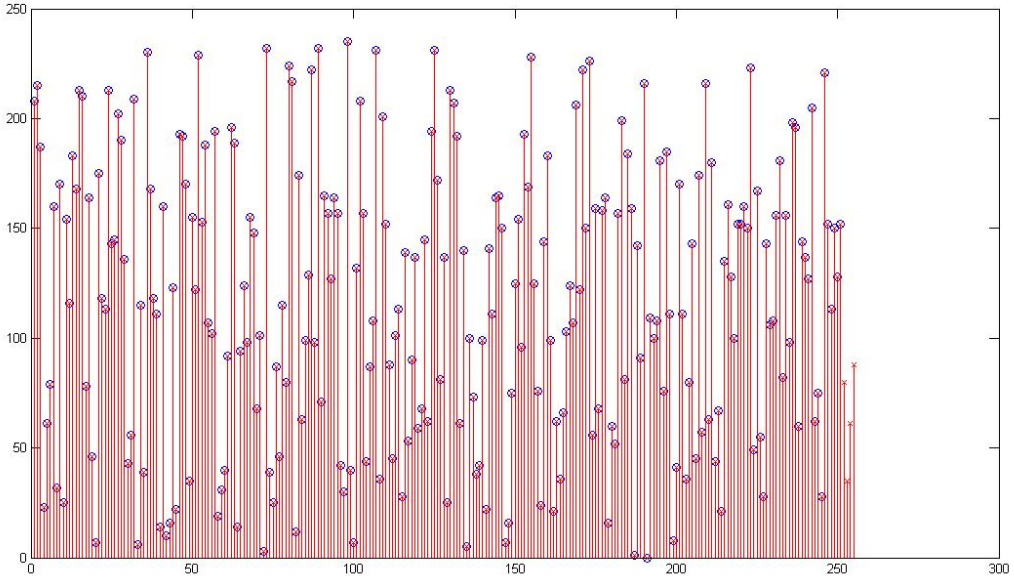

Fig 6.1 Comparison between Encoded data and original data

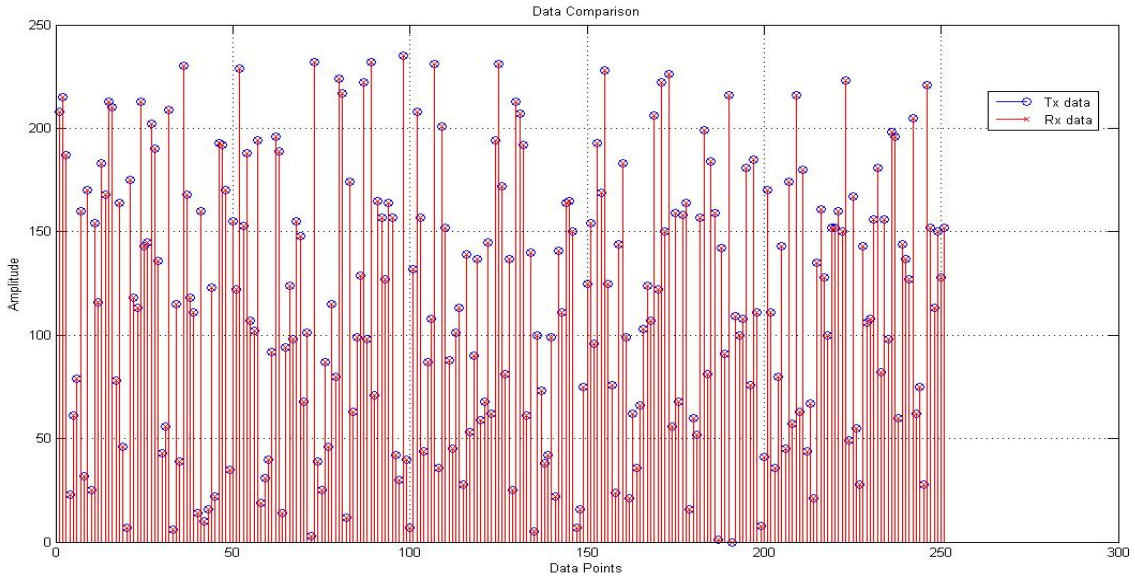

Fig 6.2 Comparison between decoded and input data

6.2Glaois Field Multiplier

It is already explained in chapter 6 that Galois _eld addition is basically modulo2 addition. Modulo2 addition is logical XOR operation. But multiplication in galois field is not that trivial. Multiplication in a particular field is not applicable to another field with different primitive polynomial. Hence implementation of the multiplier is the back bone of the design. In Our design GF (2^8) is used. And the primitive polynomial choosen is $x^8+x^4+x^3+x^2+1$. Binary representation is 100011101.

The basic difference between Galois field multiplication and general binary multiplication comes from the fact that the Galois field is finite. Hence the result of any operation should lie in the same field. In binary multiplication of two m bit data the result may go beyond m bits. But in case of Galois field as the field is defined as set of m bit symbols the result of any operation should be of m bit. Hence there are various ways of implementing it.

- Look up table approach : In this approach results of all the combinations is stored in look up table. Combination of multiplicand and multiplier can be used as address to find out the result of their multiplication. It looks to be very simple but for a larger field it takes more resources for implementation. Hence this approach is suitable for smaller fields only.
- Log Antilog approach: This approach is like using logarithm and Antilogarithm for multiplication. That means the multiplicand and multiplier can be converted into the power of the primitive element. Then these powers can be added. Corresponding element can be found out. More resources are required in this approach for larger field.
- Logic Implementation approach: In this approach logic implementation of the multiplier is done. Implementation can be either sequential or combinational.
    - Sequential approach is similar to binary multiplication. Each bit of multiplier is multiplied by the multiplicand and the partial results are shifted and added to find out the result. In addition additional effort is required as result should not exceed m bit. In the intermediate stages of multiplication whenever a '1' is carried out of the 8 bit an additional XOR operation is done. This XOR operation is derived

from the primitive polynomial. Problem with this approach is that it takes m clock cycles for multiplication in m bit Galois field. To work synchronously with other operations it should run with a clock 8 times faster than the clock used for other components.

- In Combinational approach AND and XOR gates are used to generate the result. Logic can be found out using classical truth table method. But this method is not suitable for larger field as the combinations become larger. Alternately logic can be derived using the general multiplication steps and primitive polynomial. The delay introduced by combinational logic is almost fixed. If the clock time period is higher than this delay then it can be used in a sequential circuit.

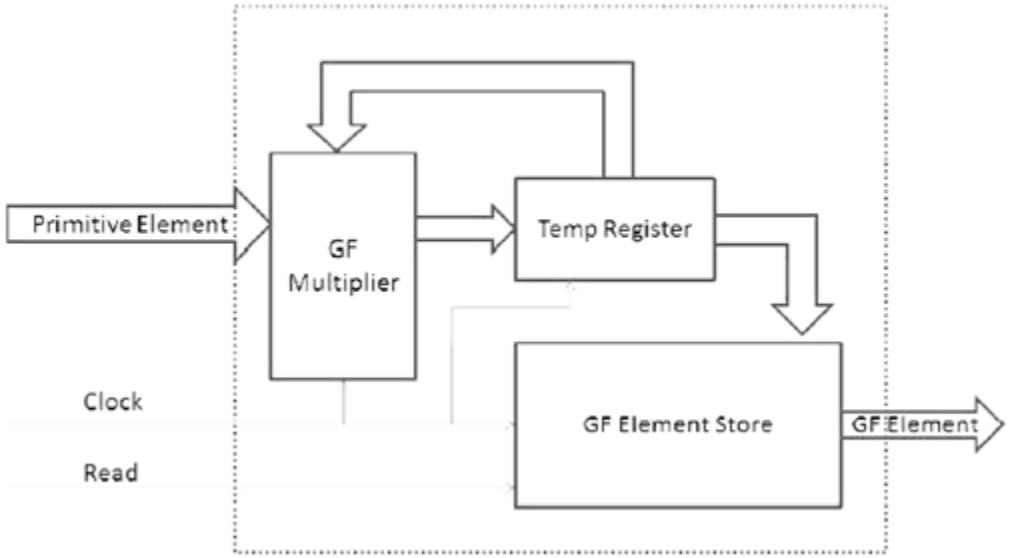

Fig 6.3 Block diagram of FEG

P	E	P	E	P	E	P	E	P	E	P	E	P	E	P	E
00	01	20	9D	40	5F	60	D9	80	85	A0	E6	C0	82	E0	12
01	02	21	27	41	BE	61	AF	81	17	A1	D1	C1	19	E1	24
02	04	22	4E	42	61	62	43	82	2E	A2	BF	C2	32	E2	48
03	08	23	9C	43	C2	63	86	83	5C	A3	63	C3	64	E3	90
04	10	24	25	44	99	64	11	84	B8	A4	C6	C4	C8	E4	3D
05	20	25	4A	45	2F	65	22	85	6D	A5	91	C5	8D	E5	7A
06	40	26	94	46	5E	66	44	86	DA	A6	3F	C6	07	E6	F4
07	80	27	35	47	BC	67	88	87	A9	A7	7E	C7	0E	E7	F5
08	1D	28	6A	48	65	68	0D	88	4F	A8	FC	C8	1C	E8	F7
09	3A	29	D4	49	CA	69	1A	89	9E	A9	E5	C9	38	E9	F3
0A	74	2A	B5	4A	89	6A	34	8A	21	AA	D7	CA	70	EA	FB
0B	E8	2B	77	4B	0F	6B	68	8B	42	AB	B3	CB	E0	EB	EB
0C	CD	2C	EE	4C	1E	6C	D0	8C	84	AC	7B	CC	DD	EC	CB
0D	87	2D	C1	4D	3C	6D	BD	8D	15	AD	F6	CD	A7	ED	8B
0E	13	2E	9F	4E	78	6E	67	8E	2A	AE	F1	CE	53	EE	0B
0F	26	2F	23	4F	F0	6F	CE	8F	54	AF	FF	CF	A6	EF	16
10	4C	30	46	50	FD	70	81	90	A8	B0	E3	D0	51	F0	2C
11	98	31	8C	51	E7	71	1F	91	4D	B1	DB	D1	A2	F1	58
12	2D	32	05	52	D3	72	3E	92	9A	B2	AB	D2	59	F2	B0
13	5A	33	0A	53	BB	73	7C	93	29	B3	4B	D3	B2	F3	7D
14	B4	34	14	54	6B	74	F8	94	52	B4	96	D4	79	F4	FA
15	75	35	28	55	D6	75	ED	95	A4	B5	31	D5	F2	F5	E9
16	EA	36	50	56	B1	76	C7	96	55	B6	62	D6	F9	F6	CF
17	C9	37	A0	57	7F	77	93	97	AA	B7	C4	D7	EF	F7	83
18	8F	38	5D	58	FE	78	3B	98	49	B8	95	D8	C3	F8	1B
19	03	39	BA	59	E1	79	76	99	92	B9	37	D9	9B	F9	36
1A	06	3A	69	5A	DF	7A	EC	9A	39	BA	6E	DA	2B	FA	6C
B	0C	3B	D2	5B	A3	7B	C5	9B	72	BB	DC	DB	56	FB	D8
1C	18	3C	B9	5C	5B	7C	97	9C	E4	BC	A5	DC	AC	FC	AD
1D	30	3D	6F	5D	B6	7D	33	9D	D5	BD	57	DD	45	FD	47
1E	60	3E	DE	5E	71	7E	66	9E	B7	BE	AE	DE	8A	FE	8E
1F	C0	3F	A1	5F	E2	7F	CC	9F	73	BF	41	DF	09		

TAB 6.1 GF(2^8) elements in terms of Power of primitive element

6.3 Encoder design

6.3.1 Architecture

The RS encoder will consist of 4 GF multipliers, 4 GF adders, 2 2:1 multiplexers, shift register as shown in figure. The circuit is known as the Linear Feedback Shift Register (LFSR) circuit.

Generator polynomial has been stored in generator polynomial register. The shift register has been initialized to all zeros. Message input is loaded sequentially one symbol at a time. As soon as message to be encoded appears on the input MUX1 switches to GF addition of input symbol and the highest degree coefficient of the shift register polynomial which has been set to zero initially. This in turn becomes the feedback. Feedback is multiplied with generator polynomial. The result is added to the shift register polynomial and shifted towards right once to update the Shift register polynomial. All these operation are synchronous. After 251 such cycle shift register polynomial contains the redundant symbols. At that time MUX2 passes these 4 redundant symbols to complete the encoding process.

6.3.2 Design Details

RS encoder is designed as per the architecture explained above. The encoder block is shown in figure on the following page.

Input Output description has been provided in table

The Encoder block is divided into two modules as described below.

1. Redundant generator: It is the main module in which the arithmetic operations have been done to generate the redundant symbols. The design has been done as explained in the architecture. It has been controlled by some control signals which have been derived by the control logic.

2. Control logic: The control logic has been used to generate the control signals for the redundant generator circuit. An additional provision has been given for bypassing any symbol. When the bypass signal is active whatever input given to the encoder will appear at the output without taking part in the encoding process. Along with the encoded data output encoder provides some status signal to indicate valid output, redundant symbol transfer and ready to accept the next block of input. It has also been responsible for deriving appropriate generator polynomial from the number of message symbols and number of redundant symbols taken as input.[8]

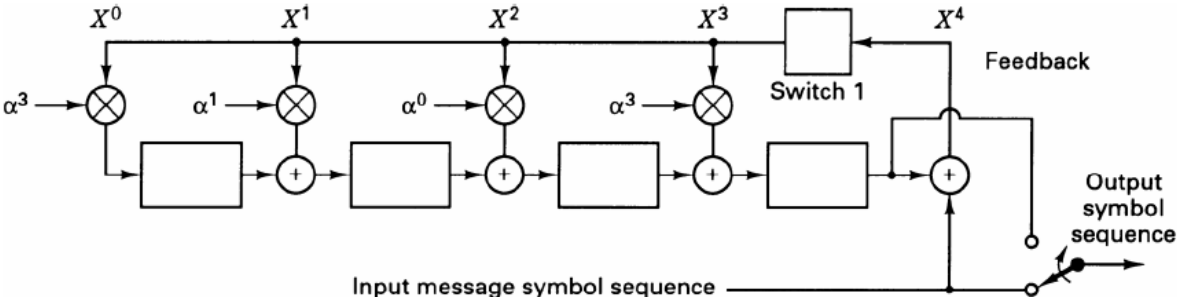

Fig 6.4 Block diagram of LFSR circuit [1]

SIGNAL	DESCRIPTION
RST	Reset all values
CLK	Clock signal
STR	Start execution of program
RD	Data read out strobe
D_IN	Data input
D_OUT	Data output
SNB	Impulse when encoding is finished

TAB 6.2 I/O ports of RS Encoder

Top level block diagram of the encoder is shown in figure on the following page RS(255,251) has been implemented and it can be further extended to various RS codes. D_IN signal has been provided to accept number of message symbols.

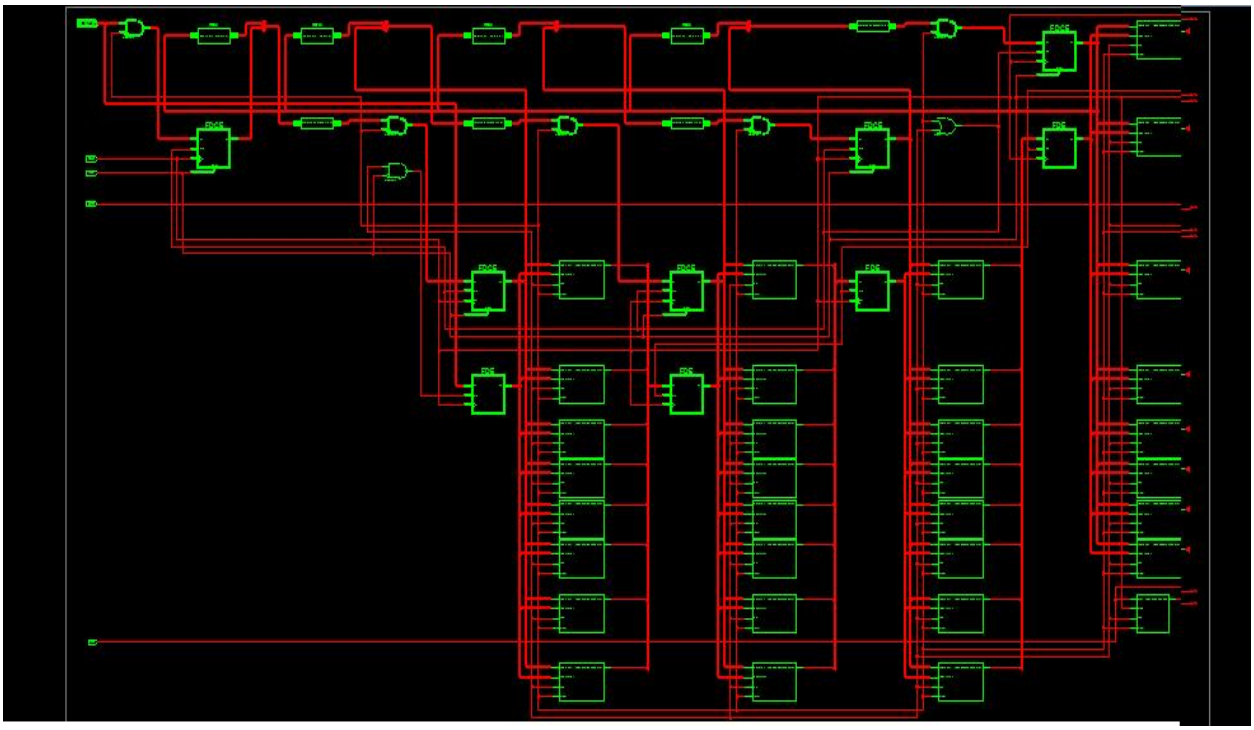

Fig 6.6 RTL schematic of RS encoder

CHAPTER 7

REULTS AND DISCUSSIONS

Reed Solomon Encoder have been synthesized on xilinx XCV2P30 FPGA. Details of the device utilization and timing summary for RS encoder has been given in table 11.1 on the following page and design statistics has been given below. Simulation Snapshot is shown in Fig. 7.1

Processor used: SPARTAN 3E

Number of clock cycles used: 255

Speed Grade: -4

Minimum period: 4.127ns (Maximum Frequency: 242.50MHz)

Minimum input arrival time before clock: 5.094 ns

Maximum output required time after clock: 5.138 ns

Maximum combinational path delay: No combinational path delay found.

The Reed Solomon Encoder has been simulated with different inputs and a clock frequency 100 MHz and the encoded data can been used in further simulation of decoder circuit.

Logic utilization	Used	Available	Utilization
Slice Flip Flops	172	9312	1%
4 Input LUTs	277	9312	2%
Input/ Output Ports	21		
Bonded IOBs	21	232	9%
GCLKs	1	24	4%

Tab 7.1 : Device utilization in FPGA in RS encoder

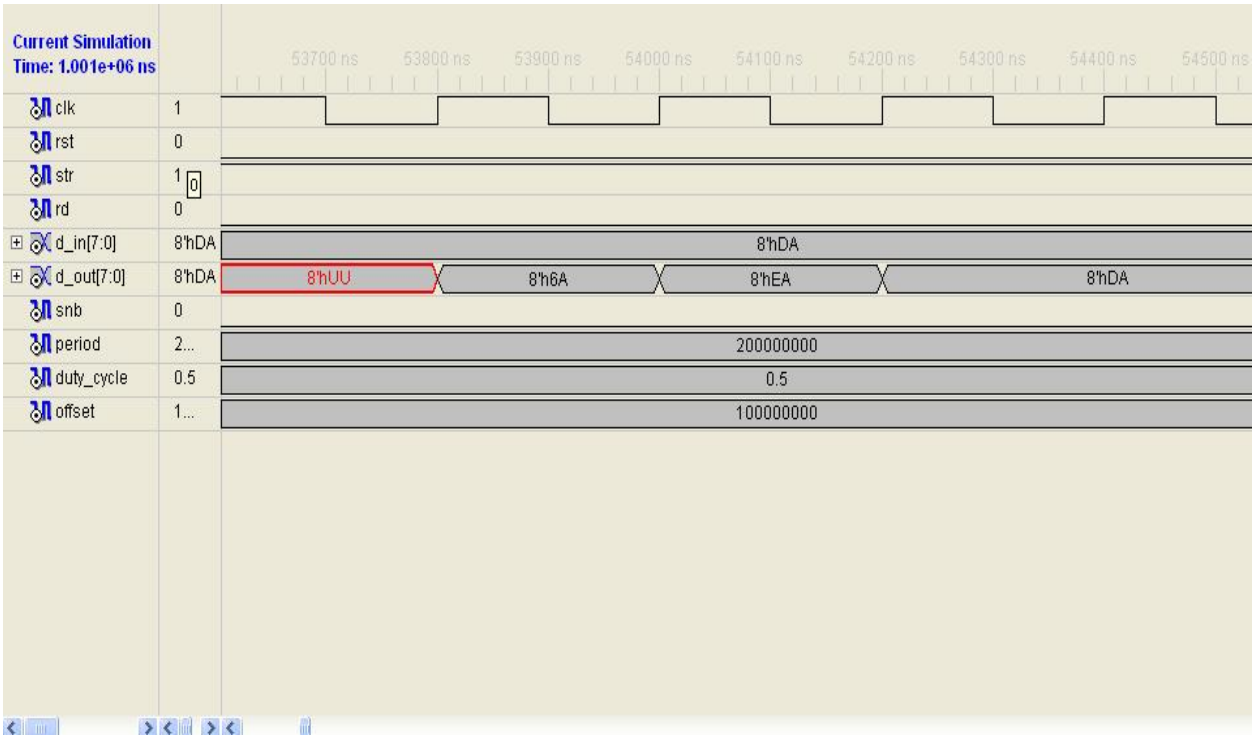

Fig 7.1 Simulation snapshot of RS encoder

CONCLUSION AND FUTURE WORK

Reed Solomon (255,251) code was simulated in MATLAB. It was found that if the error is in parity symbols even then the decoder is able to detect the output and it is of no matter to the decoder that in which symbols the error is present. The decoder first corrects the symbols and then removes the redundant parity symbols from the code word and produces the original input code word. In the error probability graph, it was seen that for a particular range of SNR, the number of error bits present can be found out using the bit error rate probability. So if the number of bits in error is beyond our range of our error correcting capability, then the signal can be ignored in the same place and an acknowledgement can be sent to send the signal again. However, if the number of bits in error is within the range then also we can estimate using those bits about how many symbols will be in error and then decide whether to try decode the transmitted signal or not. For different error correcting capabilities of RS code, the range of SNR goes on increasing as the error correcting capability increases. In VHDL, the RS encoder was designed and the timing diagrams and device utilization is stated in the results and discussions chapter. In future, one can design the RS decoder and then test this benchmark in a real time based example. One can also implement this in an FPGA.

References

[1] Sklar B, "Digital Communication: Fundamentals and Applications", Second Edition, Prentice-Hal, 2001.

[2] Martyn Riley and Iain Richardson' "An introduction to Reed Solomon codes: principles, architecture & implementation" , prentice-hall , 2001

[3] Proakis John G., Salehi M., "Communication System Engineering", Second Edition, Prentice Hall of India, 2005

[4] C. Shannon, "A Mathematical Theory of Communication", The Bell System Technical Journal, 1948

[5] Lee H., "High Speed VLSI Architecture for parallel Reed solomon Decoder", IEEE Transactions on VLSI systems, 2003.

[6] Wilhelm W., "A New Scalable VLSI Architecture for Reed-Solomon Decoders", IEEE Journal of Solid State Circuits, 1999

[7] Y. J. Jeong and W Burleson, "VLSI Array synthesis for polynomial GCD Computation and Application to Finite Field Division", IEEE Transactions on Circuits and Systems, 1994

[8] S. B. Wicker and V. K. Bhargava; "Reed Solomon Codes and Their Applications", Piscataway, NJ: IEEE Press, 1994.

[9] http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/reedsolomon

[10] www.wikipedia.org

[11]www.opencores.org

[12]www.google.com

[13]www.wordiq.com

[14]www.enc-slider.com