PERFORMANCE ENHANCEMENT OF DIGITAL COMMUNICATION USING REED SOLOMON CODES

Yasir Hafeez Khan, Muhammad Riaz and Anas Bilal
School of Information Technology, University of Lahore, Islamabad Campus Islamabad, Pakistan.
yasirhkhan@yahoo.com, mriaz77@gmail.com, and chanasbilal@gmail.com

ABSTRACT: *In digital communication, when data is transmitted from one end to another, there is a probability that the received data may be corrupted by errors when received. The rate at which error occurs is called bit error rate. Errors are induced in transmission where noise or any sort of interference is present. These errors need to be corrected in order to have reliable transmission and this is done by the error control coding. A comparison of transmission of data with and without error control coding is presented. For error correction, Reed Solomon codes have been used. Complete simulation has been done in MATLAB. Experimental results show that the data transmitted using Reed Solomon codes provides a far better result as compared to the data transmitted without applying error correction code.*

Keywords: Error Control Coding(ECC),Automatic Repeat Request (ARQ) ,Forward Error Correction (FEC),Bit Error Rate (BER),Reed Solomon (RS), Additive White Gaussian Noise(AWGN) ,Galois Field (GF),Signal to Noise Ratio(SNR),Low Density Parity Check(LDPC).

1. INTRODUCTION

Error Control Coding has rapidly grown since past few decades. It is a common field applied everywhere, but only a few know what actually error control coding is. It is not wrong to quote that this proper knowledge about this field is a requirement for any person engaged in designing or inventing a communication system.

Error Control Coding comes under information theory which is a sub branch of applied mathematics. The main concepts for information theory were introduced by Claude Shannon. He described that if the information rate of a given code does not exceeds the capacity of the channel, on which the information is to be transmitted, then there exist a coding technique that allows us to send information with a low error rate [1]. According to this concept, it is possible to transmit information on a very low error rate through a noisy or unreliable channel.

Before the publication of Shannon's famous paper, it was a perception amongst scientists that noise induced errors in the data when the data is transmitted [2]. Shannon established the fact that noise only limits the rate of transmission not the error probability. He explained that there is a capacity factor for every channel in the world. This capacity factor is determined in bits per second. He further explained that error free transmission is very much possible provided the transmitted data, i.e. the transmission rate, is kept less than this capacity factor.

Shannon's work was just theoretical. He described the facts but didn't implement them. After the evolution of Shannon's facts, in 1950s communication scientists and researchers engaged in finding schemes and codes that would implement Shannon's facts that is they started finding ways to transmit data with minor chances of error. The research was not fruitful in its advance stages. Observing this, in 1960 the whole team involved in this research divided up in two groups. The people having command over algebra were in one group and the people having command over probability were in the other group. The first group was concerned in identifying with a specific class of codes which was known as block codes. The other group with probability scientists considered the encoding and decoding as a random process. As time passed the results were fruitful. Probability scientists succeeded in finding a class of codes known as convolution codes. After getting familiar with this class of codes, probability scientists were able to design very powerful decoders for convolution codes. The groups were reunited in 1970s and this resulted in introducing a number of powerful and efficient decoding algorithms were invented. The practical implementation of these decoders however took more time and finally in 1981, they were implemented.

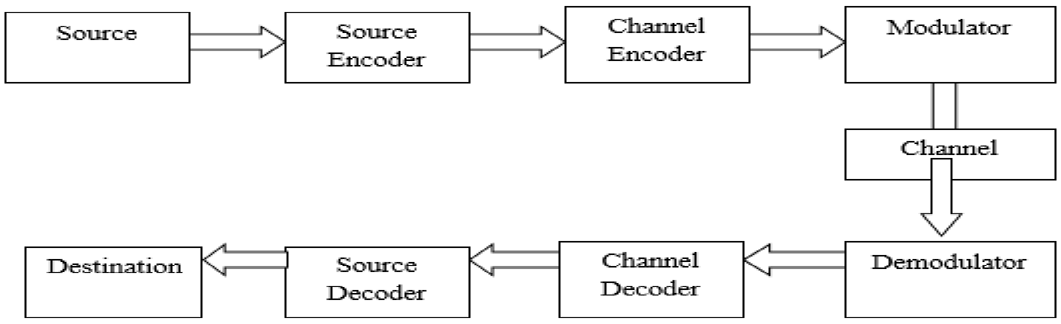

Figure 1: Digital Communication System [3]

The block diagram [3] shows the basic components of a digital communication system. Error control coding has numerous applications. There are a large number of fields where error control coding is considered to be playing a vital role in producing a reliable communication [4].

The main functions of error control coding are;

- Reduces the occurrence of undetected errors
- Reduces the overall cost of communications systems
- Secures the transmitted data
- Eliminates interference

Reducing the occurrence of errors was the main function of ECC. Today this is effectively done with the help of numerous error coding schemes. On the other hand cost of an overall communication system is reduced as error control coding prevents the repeated transmission of the same data and thus saves bandwidth which is the most precious component of any communication system. It also helps in making the transmission secure and reduces interference. Interference is a serious problem as spectrum becomes extensively packed with a number of users.

It is important to note that the facts and laws defined by Shannon depend upon whether the channel is power limited or bandwidth limited. For instance a satellite in deep space has extensive bandwidth, but limited power. So it comes under the category of power limited. On the other hand simple telephone channels have more power as compared to bandwidth allotted to them. So they come under the category of bandwidth limited.The introduction to error control coding by Shannon in 1948 was just theoretical. But today we find the successive implementation of Shannon ideas in almost every field of communication. It is due to the richness and importance of Shannon's theory that today we are able to communicate efficiently over long distances.

Without any doubt, reliable data transmission is impossible without using error correction schemes. The need of error control coding has increased with the innovations like digital cellular telephony and digital television, etc. It is absolutely correct to say that with the passage of time, this need would get even more pervasive[5].

Once an error is detected, there are two methods for moving further ahead. These methods are;

- Automatic Repeat Request (ARQ)
- Forward Error Correction (FEC)

1.1 Automatic Repeat Request

As the name itself suggests, in ARQ, the receiver after detecting the error demands the transmitter to resend the data. ARQ works on the concept of acknowledgements (ACKS) that are sent by the receiver exhibiting that the data has been received correctly. If the transmitter doesn't get an ACK for a particular message, it commences retransmission. ARQ is divided into three types of protocols;

- Stop and Wait
- Go Back N
- Selective Repeat

In Stop and Wait, a single frame is transmitted by the transmitter. After receiving acknowledgement for that particular frame, the transmitter transmits the next frame in the queue. If ACK is not received in a particular time period, the data is retransmitted. In Go Back N, the transmitter does not wait ACK for every single frame. When a particular frame undergoes error, the receiver discards that particular frame and all succeeding frames and regards them as out of order. A time comes when transmitter realizes that ACKS are not being received after a particular frame. So it goes back N frames and commences retransmission. In Selective Repeat, unlike Go Back N, out of order packets are not discarded and the timed out frames are retransmitted by the transmitter.

1.2 Forward Error Correction

FEC is the most used approach in error control coding. In this technique, the sender adds parity or redundant bits to the original data .On encountering an error, the correction is performed by the receiver itself using the parity bits. Forward error correction is also referred as channel coding. FEC is divided into two major categories;

- Convolution Codes
- Block Codes

1.2.1 Convolution Codes

In this class of codes, the output of the encoder is in the form of encoded sequence that is produced from an input information sequence which means a sequence of message is transformed into a sequence of code. It is important to note that these kinds of encoder possess a memory. The output of the encoder is not just dependent on the input given in that particular time instant, but it is dependent on previous inputs as well. The output of the encoder is given by 'n' at a particular 'k' input and 'm' previous inputs. Thus a convolution code is represented by $\mathrm{C_{conv}}\,(\mathrm{n,k,m})$ where 'm' is also referred as the encoder's memory and 'n' and 'k' are normally small integers with k<n. In order to achieve alow error probability, memory 'm' must be large.[6]

1.2.2 Block Codes

In the nomenclature of information and coding theory block codes is category of error correcting codes which works on the principle of encoding the data in the form of blocks. The block of the message is converted into a block of code. This class of codes includes Reed Solomon codes, Hamming codes etc. These codes are also lying under the class of linear codes and therefore are also referred as linear block codes. In block codes, the digital message to be transmitted is grouped into blocks of k bits. At the transmitting end, the encoder takes the complete block as an input and makes it a code word by adding parity bits to it. These parity bits, added by encoder, are also known as redundant bits. At the receiving end, the decoder works on the code word and removes the parity bits and obtains the original message transmitted by the sender. As the parity bits do not hold any message information, so they are discarded. All codes working on this principle are called block codes and are represented by$\mathrm{C_b}(\mathrm{n,k})$.

1.3 Reed Solomon Codes

Reed Solomon codes are error correcting codes proposed by Irving Reed and Gustave Solomon in 1960. This class of codes possesses great power and utility. RS codes find applicable in many fields, for instance, compact disc players, high speed modems, satellite communications etc.

A block of data is acquired by the RS encoder that adds redundant bits to the digital data before transmitting it. During transmission of the data, errors may arise due to certain reasons. At the receiving end, the RS decoder tries to spot the errors and then further tries to correct them in order to obtain the original data that has been transmitted by the sender side. It is important to note that the error correction capability of RS code depends upon code's characteristics , i.e. the parameters set up for the code.

The detail of the above mentioned parameters is described as the input of RS encoder is 'k' data symbols where each symbol is constructed of 'm' bits. When the encoder adds the parity symbols to the 'k' data symbol, the complete symbol code word (i.e. original data and the addition of parity bits) is represented by 'n'. The total number of parity symbols is n-k. In order to find the error correction capability t of RS code we make use of the formula 2t=n-k.

DATA	PARITY

<------------------------n----------------------------->

Suppose we intend to transmit a 'k' symbol message where 'k' is 223 and error correcting capability (t) is 16. Then n will be calculated as;

$n = k + 2t = 223 + 2(16)$=255

As the value of t is 16, this means that n=k+2t should assure reconstruction of the message at the receiving end if the total numbers of corrupted symbols are less than t. According to this fact, in the scenario described above, where k is 223 and n is 255, the code is capable of correcting 16 corrupted symbols of every 255 symbol packet[7].

Detailed description of RS codes is described in later sections of the report

.

A typical RS system can be shown as;

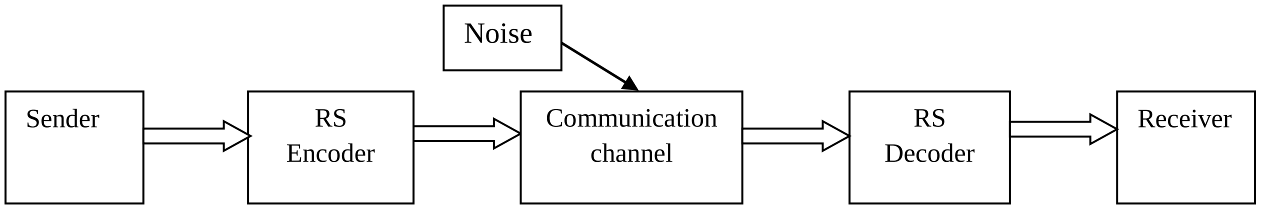

Figure 2: Typical Reed Solomon Architecture

2.Methodology

2.1 Galois Field

Galois field plays a very important role in encoding and decoding procedures of RS algorithm. It is named for a French mathematician Evariste Galois. Galois field is a finite field. For any prime number p, there is GF (p), i.e. Galois field that consists of precisely p elements. Galois Field is categorized into two types; binary field GF (2) and extended Galois Field GF (2^m)

Binary Galois Field

As the name suggest, binary Galois field contains 0 or 1. It is defined as GF (2) = {0, 1}. The addition of two numbers in the binary field will always result in the number related to the set. Similarly the multiplication will also result in the number related to the set. It uses the modulo 2 addition.

Table 1 below shows the addition in binary Galois field. It is evident that the result of addition always remains within the set.

Table 1: Addition for GF (2)

+	**0**	**1**
0	0	1
1	1	0

Table 2 below shows the multiplication in binary Galois field. Again we can see that multiplication result always remains within the set.

Table 2: Multiplication for GF (2)

*****	**0**	**1**
0	0	0
1	0	1

2.1.1 Polynomials in Binary Galois Field

When it comes to polynomials in binary Galois field, it is important to note that every binary number can be stand for a polynomial. A polynomial up to power n can be represented as;

$$f(x) = f_0 + f_1x + f_2x^2 + f_3x^3 + \cdots + f_nx^n$$

Suppose if we want to generate a polynomial for the binary number 101110001. In this case value of n is 8 which mean that the above equation goes up to 8. The multiplication is shown in the table below;

Table 3: Binary to Polynomial Conversion

1	0	1	1	1	0	0	0	1
x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8

The above multiplication results in the conversion of binary to polynomial which is given by;

$$101110001 \leftrightarrow 1 + 0 + x^2 + x^3 + x^4 + 0 + 0 + 0 + x^8$$

$$101110001 \leftrightarrow 1 + x^2 + x^3 + x^4 + x^8$$

2.1.2 Multiplication of Polynomials in Binary Galois Field

For multiplication of two polynomials consider any two polynomials. Suppose the first polynomial is $1 + x^2 + x^3 + x^4$ and another polynomial$x^3 + x^5$. Multiplication in finite fields uses AND logic operation. So multiplying these two polynomials results in;

$$(1 + x^2 + x^3 + x^4)*(x^3 + x^5) = x^3 + x^5 + x^5 + x^6 + x^7 + x^7 + x^8 + x^9$$

$$(1 + x^2 + x^3 + x^4)*(x^3 + x^5) = x^3 + x^6 + x^8 + x^9$$

Addition uses XOR logic operation so the even numbers of terms are canceled out.

2.1.3 Addition of Polynomials in Binary Galois Field

For addition of two polynomials consider any two polynomials. Suppose three polynomials $1 + x^2 + x^3 + x^4$, $1 + x^3 + x^5$ and $1 + x^2$. Now adding these three polynomials results in;

$$(1 + x^2 + x^3 + x^4)+(1 + x^3 + x^5)+(1 + x^2) = 1 + x^2 + x^3 + x^4 + 1 + x^3 + x^5 + 1 + x^2$$

$$(1 + x^2 + x^3 + x^4)+(1 + x^3 + x^5)+(1 + x^2) = 1 + x^4 + x^5$$

2.1.4 Extended Galois Field

There is also a possibility of extending GF (p) up to p^m elements, where m is a positive integer. This extension results in GF (p^m). As the field is extended, there is addition of some exclusive elements other then 0 and 1. These elements are represented by α. Now power of α can represent each non zero in the extended field GF (2^m). F, an infinite field, is constituted by initiating with elements {0, 1, α}. Additional to this more elements are generated by multiplying α with every last entry[8]. This gives;

$$F = \{0, 1, \alpha, \alpha^2, \dots, \alpha^j, \dots\} = \{0,\ \alpha^0, \alpha^1\alpha^2, \dots, \alpha^j, \dots\}$$

In order to acquire finite set of elements of GF (2^m) from above mentioned expression, there is a compulsion of imposing a condition on F so that it only holds 2^m elements which should be closed under multiplication. This condition characterized by;

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

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

By making use of the above mentioned polynomial constraint, any field element that possess power greater than or equivalent to $2^m - 1$, can be trimmed down to powerless then the factor $2^m - 1$. This is shown by;

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

Now using the above mentioned expression, we can construct a finite sequence from an infinite sequence. We represent this newly constructed finite sequence with F* and is given by;

$$F *= \left\{0, 1, \alpha, \alpha^2, \dots, \alpha^{(2^m-2)}, \alpha^{(2^m-1)}, \alpha^{(2^m)}, \dots\right\}$$

$$F *= \{0,\ \alpha^0, \alpha^1\alpha^2 \dots, \alpha^{(2^m-2)}, \alpha^0, \alpha^1\alpha^2, \dots\}$$

Thus now the elements for the field GF (2^m) will be given by;

$GF(2^m) = \{0,\ \alpha^0, \alpha^1\alpha^2 \dots, \alpha^{(2^m-2)}\}$

As we know that the extended field is not restricted to just two numbers so we consider an example GF (4) = {0, 1, 2, 3}. Table 4 shows the addition of this example.

Table 4: Addition for GF (4)

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Table 5: Multiplication for GF (4)

+	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Table 5 shows the multiplication for the same example.

We can also represent GF (4) in binary by performing a conversion. The above example can be performed again using binary conversion. Table 6 shows addition in binary representation and Table 7 shows the multiplication in binary representation.

Table 6: Addition in Binary Representation

+	**00**	**01**	**10**	**11**
00	00	01	10	11
01	01	00	11	01
10	10	11	00	01
11	11	10	01	00

Table 7: Multiplication in Binary Representation

*****	**00**	**01**	**10**	**11**
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

We also need the polynomials representing the binary conversions in Table 6 and Table 7. These addition and multiplication of polynomials are shown in Table 8 and Table 9 respectively.

Table 8: Addition in Polynomial Representation

+	**0**	**1**	**x**	**x+1**
0	0	1	x	x+1
1	1	0	x+1	x
x	x	x+1	0	1
x+1	x+1	x	1	0

Table 9: Multiplication in Polynomial Representation

*****	**0**	**1**	**x**	**x+1**
0	0	0	0	0
1	0	1	x	x+1
x	0	x	x+1	1
x+1	0	x+1	1	x

2.1.5 Primitive Polynomials

Primitive polynomial characterizes the finite fields GF (2^m) which are required for Reed Solomon codes. There are conditions needed for a polynomial to be primitive. A polynomial 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. It is important to note that their division results into a non zero quotient and a zero reminder. Table 10 lists some important primitive polynomials.

Table 10: Primitive Polynomials [16]

m		m	
3	$1+X+X^3$	14	$1+X+X^6+X^{10}+X^{14}$
4	$1+X+X^4$	15	$1+X+X^{15}$
5	$1+X^2+X^5$	16	$1+X+X^3+X^{12}+X^{16}$
6	$1+X+X^6$	17	$1+X^3+X^{17}$
7	$1+X^3+X^7$	18	$1+X^7+X^{18}$
8	$1+X^2+X^3+X^4+X^8$	19	$1+X+X^2+X^5+X^{19}$
9	$1+X^4+X^9$	20	$1+X^3+X^{20}$
10	$1+X^3+X^{10}$	21	$1+X^2+X^{21}$
11	$1+X^2+X^{11}$	22	$1+X+X^{22}$
12	$1+X+X^4+X^6+X^{12}$	23	$1+X^5+X^{23}$
13	$1+X+X^3+X^4+X^{13}$	24	$1+X+X^2+X^7+X^{24}$

3.1 Reed Solomon Encoding

Reed Solomon codes consist of symbols which are made of m bit sequence. The following equation articulates conventional form of Reed Solomon (RS) error correction scheme.

$(n, k) = (2^m - 1, 2^{\,m} - 1 - t)$

. The polynomial generation in RS algorithm is given by;

$$g(x) = g_0 + g_1X + g_2X^2 + \cdots + g_{2t-1}X^{2t-1} + X^{2t}$$

The degree of polynomial generator and the number of parity symbols both are equivalent. RS codes are basically a subset of Bose Chaudhuri Hochquenghem (BCH) codes. The earlier mentioned correlation between polynomial generator and the number of parity symbols is the same for BCH codes as well. Given that the degree of polynomial generator is 2t, it is essential that there must be 2t powers of α which are also the roots of the polynomial. The roots of g(X) are referred as α, α^2 up to α^{2t}. However it is not compulsory to initiate the root α as initiating with any power of α is promising[9]. Taking an example of (7,3) error correcting Reed Solomon code. The according to the above mentioned equations the polynomial generator in terms of its roots would be given by;

$$g(X) = (X-\alpha)(X-\alpha^2)(X-\alpha^3)(X-\alpha^4)$$
$$= (X^2 - (\alpha+\alpha^2)X + \alpha^3)(X^2 - (\alpha^3+\alpha^4)X + \alpha^7)$$
$$= (X^2 - \alpha^4X + \alpha^3)(X^2 - \alpha^6X + \alpha^0)$$
$$= X^4 - (\alpha^4+\alpha^6)X^3 + (\alpha^3+\alpha^{10}+\alpha^0)X^2 - (\alpha^4+\alpha^9)X + \alpha^3$$
$$= X^4 - \alpha^3X^3 + \alpha^0X^2 - \alpha^1X + \alpha^3$$

Moving from least order to highest order and converting negative signs to positive, as +1 = –1 in binary field, we come up with the following equation for g(x);

$$g(X) = \alpha^3 + \alpha^1X + \alpha^0X^2 + \alpha^3X^3 + X^4$$

3.1.1 Systematic Encoding

Since RS codes are cyclic, so encoding in systematic form is similar to binary encoding process. RS coding is called systematic because the original data is not changed but only parity bits are added to it. A message polynomial, referred as m(X), is shifted into k stages of codeword in the most right register. After that an addition of parity polynomial, referred as p(X), is done by placing it into n-k stages in the most left register. Now m(X) is multiplied by X $^{n-k}$ which results in manipulating m(X) algebraically in such a way that it is shifted right with respect to n-k positions. Now division between X $^{n-k}$ m(X) and g(X) is done which gives;

$$X^{\,n-k}\, m(X) = q(X)g(X) + p(X)$$

In the above equation q(X) is quotient polynomial and p(X) is remainder polynomials. U(X) represents the codeword polynomial and is given by;

$$U(X) = p(X) + X^{\,n-k}\, m(X)$$

The procedure mentioned above could be more elaborated by encoding the following symbol messages.

$$\underbrace{010}_{\alpha^1}\ \underbrace{110}_{\alpha^3}\ \underbrace{111}_{\alpha^5}$$

For RS code (7,3) with the generator polynomial equation described earlier, multiplying the message polynomial α^1+ α^3 X+ α^5 X^2 by X $^{n-k}$= X4 results in α^1X^4+ α^3X^5+ α^5 X^6. Now the next step is division, as described earlier, which takes place between the shifted message polynomial with the polynomial generator α^3+ α^1 X+ α^0 X^2+ α^3 X^3+ X^4.

Table 11: Rules for Polynomial Addition

	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	0	α^3	α^6	α^1	α^5	α^4	α^2
α^1	α^3	0	α^4	α^0	α^2	α^6	α^5
α^2	α^6	α^4	0	α^5	α^1	α^3	α^0
α^3	α^1	α^0	α^5	0	α^6	α^2	α^4
α^4	α^5	α^2	α^1	α^6	0	α^0	α^3
α^5	α^4	α^6	α^3	α^2	α^0	0	α^1
α^6	α^2	α^5	α^0	α^4	α^3	α^1	0

Table 12: Rules for Polynomial Addition

	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^1	α^1	α^2	α^3	α^4	α^5	α^6	α^0
α^2	α^2	α^3	α^4	α^5	α^6	α^0	α^1
α^3	α^3	α^4	α^5	α^6	α^0	α^1	α^2
α^4	α^4	α^5	α^6	α^0	α^1	α^2	α^3
α^5	α^5	α^6	α^0	α^1	α^2	α^3	α^4
α^6	α^6	α^0	α^1	α^2	α^3	α^4	α^5

It is important to note that the polynomial division is more complex as compared to its binary equivalent. This is because the required operations have to follow certain rules which are presented in Table 11 and Table 12. The division results in the following remainder or parity polynomial.

$p(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3$

Then, the resulting polynomial can be written as follows:

$U(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6$

In order to encode a three symbol sequence in systematic shape in accordance with (7,3) Reed Solomon codes illustrated by polynomial generator g(X), a linear feedback shift register (LFSR) is required. As described earlier, the LFSR also works on the principle of moving from least to highest order. In this encoding procedure, the (7, 3) Reed Solomon nonzero code words consists of 2^m-1 = 7 symbols where each symbol consists of m= 3 bits. It is important to note that procedure to be done here is non binary, so that each stage in the shift register holds a 3-bit symbol.

LFSR consists of two switches referred as switch 1 and switch 2. Switch 1 is shut down during first k clock cycles which permit the shifting of message symbols to n-k stage shift register. Switch 2 is positioned descendent to permit instantaneous transport of the message symbols directly to an output register. Once the kth message symbol is shifted to output register, switch 2 is positioned ascendant and switch 1 is opened. Now parity symbols are transferred to the output register by n-k clock cycles as a result the shift register becomes under empty state. At this stage the total numbers of clock cycles are equal to n and the output register is the codeword polynomial p(X)+X^{n-k} m(X). Using the same symbol sequence 010, 110, 111 where the right most symbols is the early earliest symbol, and the rightmost bit is the first bit.

Once the third clock cycle is completed, there are four parity symbols in the register that are α^0, α^2, α^4, and α^6. Now switch 1 is opened and switch 2 turned in ascendant position and parity symbols are transferred to output. At this instant the output codeword i.e. U(X) is given by;

$$U(X) = \sum_{n=0}^{6} u_n X^n$$

Using the above mentioned formula output codeword comes to be;

$$U(X) = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6$$

$$U(X) = (100) + (001)X + (011)X^2 + (101)X^3 + (010)X^4 + (110)X^5 + (111)X^6$$

As discussed earlier that validating contents of the register in polynomial case is complex as compared to binary case. Rules, showed in Table 1 and Table 2, are to be applied here.

It is compulsory that roots of g(X) must be the roots of the codeword generated by it. This is because a suitable codeword is given by;

$$U(X) = m(X)\, g(X) \qquad [16]$$

As a result a random codeword must yield zero value when it is evaluated at any root of polynomial generator. Keeping this fact in mind the earlier mentioned equation $U(X) = (100) + (001)X + (011)X^2 + (101)X^3 + (010)X^4 + (110)X^5 + (111)X^6$ is simplified by using the following expression;

$$U(\alpha) = U(\alpha^2) = U(\alpha^3) = U(\alpha^4) = 0$$

By above mentioned equation each single term of U(X) will give following results;

$$U(\alpha) = \alpha^0 + \alpha^3 + \alpha^6 + \alpha^9 + \alpha^5 + \alpha^8 + \alpha^{11}$$

$$= \alpha^0 + \alpha^3 + \alpha^6 + \alpha^2 + \alpha^5 + \alpha^1 + \alpha^4$$

$$= \alpha^1 + \alpha^0 + \alpha^6 + \alpha^4 = \alpha^3 + \alpha^3 = 0$$

$$U(\alpha^2) = \alpha^0 + \alpha^4 + \alpha^8 + \alpha^{12} + \alpha^9 + \alpha^{13} + \alpha^{17}$$

$$= \alpha^0 + \alpha^4 + \alpha^1 + \alpha^5 + \alpha^2 + \alpha^6 + \alpha^3$$

$$= \alpha^5 + \alpha^6 + \alpha^0 + \alpha^3 = \alpha^1 + \alpha^1 = 0$$

$$U(\alpha^3) = \alpha^0 + \alpha^5 + \alpha^{10} + \alpha^{15} + \alpha^{13} + \alpha^{18} + \alpha^{23}$$

$$= \alpha^0 + \alpha^5 + \alpha^3 + \alpha^1 + \alpha^6 + \alpha^4 + \alpha^2$$

$$= \alpha^4 + \alpha^0 + \alpha^3 + \alpha^2 = \alpha^5 + \alpha^5 = 0$$

$$U(\alpha^4) = \alpha^0 + \alpha^6 + \alpha^{12} + \alpha^{18} + \alpha^{17} + \alpha^{23} + \alpha^{29}$$

$$= \alpha^0 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^1$$

$$= \alpha^2 + \alpha^0 + \alpha^5 + \alpha^1 = \alpha^6 + \alpha^6 = 0$$

The above calculations exhibits that a codeword calculated at any root of generator polynomial necessarily gives a zero value.

3.2 Reed Solomon Decoding

In the start (7, 3) Reed Solomon code was used to encode a message which ultimately leads to generation of a codeword polynomial. As the transmission takes place this codeword is altered and as a result errors are induced at receiver's end. Keeping in mind the fact that errors induced during the transmission should correspond to the code's error correction capacity. According to this particular case, where codeword consists of seven symbols, error e(X) can be given as follows;

$$e(X) = \sum_{n=0}^{6} e_n X^n$$

Assuming a double symbol error we can elaborate the above equation as;

$$e(X) = 0 + 0X + 0X^2 + \alpha^2 X^3 + \alpha^5 X^4 + 0X^5 + 0X^6$$

$$e(X) = (000) + (000)X + (000)X^2 + (001)X^3 + (111)X^4 + (000)X^5 + (000)X^6$$

In simple words we can assume from the above equation that a data symbol and a parity symbol have been corrupted. To elaborate more, data symbol is altered with 3 bit error and parity symbol is altered with 1 bit error where parity symbol and data symbol are represented by α^2 and α^5 respectively. Now another polynomial is induced here and is named as corrupted codeword polynomial represented by r(X) which is actually the sum of error pattern e(X) and transmitted codeword polynomial U(X). Simply represented as;

$r(X) = U(X) + e(X)$

As the above equation states that in order to get r(X) we have to add e(X) to U(X), so adding both expressions give;

$$r(X) = (100) + (001)X + (011)X^2 + (100)X^3 + (101)X^4 + (110)X^5 + (111)X^6$$

$$= \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^0 X^3 + \alpha^6 X^4 + \alpha^3 X^5 + \alpha^5 X^6$$

Observing the equations we reveal that there are four unknowns. This means that in order to solve it, we need four equations. It is important to note that the four so called unknowns are two error location and two error values.

3.3 Calculation of Syndrome

Now the concept of syndrome plays its role. Basically syndrome results from the parity check that is executed on corrupted codeword (r) in order to be certain about it validity. If r is a valid, i.e. it is a member of valid code word set, then syndrome value (S) must be equal to zero. On the other hand, if r is not a member of valid code word set, the value of S must be non zero. Basically syndrome (S) consists of n-k symbols i.e. Si have a set of values from i=1 up to n-k. So there will be four symbols in each single syndrome vector in our case i.e. (7,3) Reed Solomon code. The syndrome calculation can be done by the fact that;

$U(X) = m(X)g(X)$

The above expression, we can conclude that each U(X) is a multiple of g(X) which is the generator polynomial. Keeping the fact that U(X) is a multiple of g(X) it is compulsory that roots of both polynomial expressions, U(X) and g(X), must be same. As discussed earlier that r(X) = U(X) + e(X) so we should get a zero value for a valid code word when r(x) is evaluated at every single root of g(X). If a non-zero value is obtained, then it is verified that it is not a valid code word. Syndrome is computed by;

$$S_i = r(X)|_{X=\alpha^i} = r(\alpha^i) \qquad where \quad i = 1, \dots, n-k$$

Using the above expression, syndrome computation is done as follows;

$$S_1 = r(\alpha) = \alpha^0 + \alpha^3 + \alpha^6 + \alpha^3 + \alpha^{10} + \alpha^8 + \alpha^{11}$$

$$= \alpha^0 + \alpha^3 + \alpha^6 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^4 = \alpha^3$$

$$S_2 = r(\alpha^2) = \alpha^0 + \alpha^4 + \alpha^8 + \alpha^6 + \alpha^{14} + \alpha^{13} + \alpha^{17}$$

$$= \alpha^0 + \alpha^4 + \alpha^1 + \alpha^6 + \alpha^0 + \alpha^6 + \alpha^3 = \alpha^5$$

$$S_3 = r(\alpha^3) = \alpha^0 + \alpha^5 + \alpha^{10} + \alpha^9 + \alpha^{18} + \alpha^{18} + \alpha^{23}$$

$$= \alpha^0 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^4 + \alpha^4 + \alpha^2 = \alpha^6$$

$$S_4 = r(\alpha^4) = \alpha^0 + \alpha^6 + \alpha^{12} + \alpha^{12} + \alpha^{22} + \alpha^{23} + \alpha^{29}$$

$$= \alpha^0 + \alpha^6 + \alpha^5 + \alpha^5 + \alpha^1 + \alpha^2 + \alpha^1 = 0$$

The above calculation suggests that the message received holds errors since all syndrome values are not equal to zero.

The calculation done above can also be verified by another procedure. This procedure is to relate roots of g(X) with the error polynomial e(X). The term 'relate', used above, means that evaluating e(X) at the roots of g(X). This evaluation must give the same syndrome results when r(X) is evaluated at the roots of g(X). This calculation is done as follows;

$$S_i = r(X)|_{X=\alpha^i} = r(\alpha^i) \qquad where \quad i = 1, \dots, n-k$$

$$S_i = [U(X) + e(X)]|_{X=\alpha^i} = U(\alpha^i) + e(\alpha^i)$$

$$S_i = r(\alpha^i) = U(\alpha^i) + e(\alpha^i) = 0 + e(\alpha^i)$$

$$e(X) = \alpha^2 X^3 + \alpha^5 X^4$$

Using the above expression, syndrome computation is done as follows;

$$S_1 = e(\alpha^1) = \alpha^5 + \alpha^9 = \alpha^5 + \alpha^2 = \alpha^3$$

$$S_2 = e(\alpha^2) = \alpha^8 + \alpha^{13} = \alpha^1 + \alpha^6 = \alpha^5$$

$$S_3 = e(\alpha^3) = \alpha^{11} + \alpha^{17} = \alpha^4 + \alpha^3 = \alpha^6$$

$$S_4 = e(\alpha^4) = \alpha^{14} + \alpha^{21} = \alpha^0 + \alpha^0 = 0$$

Notice that the above syndrome values obtained by evaluating e(X) at the roots of g(X) are the same as the syndrome values obtained when r(X) was evaluated at the roots of g(X).

3.4 Error Location and Error Values

As the syndrome is computed, we know have to find error location and error values. First we will systematically find locations of the errors. Once locations have been found out, we will go for the error values.

3.5 Error Location

Assume that there are v in the code word at the locations represented by $X^{j_1}, X^{j_2}, X^{j_v}$.. Assuming this, the error polynomial will be defined as;

$e(X) = e_{j_1} X^{j_1} + e_{j_2} X^{j_2} + \dots + e_{j_v} X^{j_v}$

In the above expression, indices 1, 2 up to v correspond to 1st, 2nd up to vth errors. On the other hand, the index j corresponds to error location. Now there is need to determine every single error value and its location in order to correct the code. Error value is represented by e_{j1} and its corresponding error location is represented as X^{j1}. $\beta_l = \alpha^{j1}$ is an error locator number to be used here. Now the syndrome symbols are acquired by replacing α^i into r(X). This would be given as follows;

$$S_1 = r(\alpha) = e_{j_1}\beta_1 + e_{j_2}\beta_2 + \dots + e_{j_v}\beta_v$$

$$S_2 = r(\alpha^2) = e_{j_1}\beta_1^2 + e_{j_2}\beta_2^2 + \dots + e_{j_v}\beta_v^2$$

$$\vdots \; S_{2t} = r(\alpha^{2t}) = e_{j_1}\beta_1^{2t} + e_{j_2}\beta_2^{2t} + \dots + e_{j_v}\beta_v^{2t}$$

As discussed earlier that as the syndrome gives a non zero value, it is evident that error is present. Now it is required to find the error location. Here another term is introduced which is known as error locator polynomial and is represented by

The roots of error locator polynomial come out to be $1/\beta_1, 1/\beta_2 \dots 1/\beta_v$. Now a system of equations can be formed involving syndrome and error locator polynomial. This is done using modelling techniques and the system is givenby;

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{t-1} & S_t \\ S_2 & S_3 & S_4 & \cdots & S_t & S_{t+1} \\ & & & \bullet & & \\ & & & \bullet & & \\ & & & \bullet & & \\ S_{t-1} & S_t & S_{t+1} & \cdots & S_{2t-3} & S_{2t-2} \\ S_t & S_{t+1} & S_{t+2} & \cdots & S_{2t-2} & S_{2t-1} \end{bmatrix} \begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \bullet \\ \bullet \\ \bullet \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ \bullet \\ \bullet \\ \bullet \\ -S_{2t-1} \\ -S_{2t} \end{bmatrix}$$

Assuming the (7,3) Reed Solomon code we can express the model as;

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix}$$

$$\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^6 \\ 0 \end{bmatrix}$$

Now in order to find coefficients we have to take inverse for the above system. The inverse of a matrix is given by;

$$\text{Inv}\,[\text{A}] = \frac{\text{cofactor}\,[\text{A}]}{\det[\text{A}]}$$

Now finding the determinant $\det[\text{A}]$ to be used in above expression;

$$\det \begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} = \alpha^9 + \alpha^{10} = \alpha^2 + \alpha^3 = \alpha^5$$

Finding the cofactor to be used in inverse calculation;

$$\text{cofactor} \begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} = \begin{bmatrix} \alpha^6 & \alpha^5 \\ \alpha^5 & \alpha^3 \end{bmatrix}$$

Finally, using determinant and cofactor, inverse would be given by;

$$\text{Inv} \begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} = \frac{\begin{bmatrix} \alpha^6 & \alpha^5 \\ \alpha^5 & \alpha^3 \end{bmatrix}}{\alpha^5} = \alpha^{-5} \begin{bmatrix} \alpha^6 & \alpha^5 \\ \alpha^5 & \alpha^3 \end{bmatrix}$$

$$\text{Inv} \begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} = \begin{bmatrix} \alpha^1 & \alpha^0 \\ \alpha^0 & \alpha^5 \end{bmatrix}$$

The above calculated inverse could be verified by a simple matrix calculation law that multiplication of a matrix with its inverse results in an identity matrix of the same size. So in order to confirm our result;

$$\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} \begin{bmatrix} \alpha^1 & \alpha^0 \\ \alpha^0 & \alpha^5 \end{bmatrix} = \begin{bmatrix} \alpha^4 + \alpha^5 & \alpha^3 + \alpha^{10} \\ \alpha^6 + \alpha^6 & \alpha^5 + \alpha^{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

As the multiplication results in a 2×2 identity matrix, hence it is verified that inverse is accurately calculated.

Now refreshing the earlier discussed equation;

$$\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^6 \\ 0 \end{bmatrix}$$

Finding the error locations started by simplifying the coefficients of the σ(X).

σ(X). This would be given as follow;

$$\sigma(X) = (1 + \beta_1 X)(1 + \beta_2 X) \dots (1 + \beta_V X)$$

$$= 1 + \sigma_1 X + \sigma_2 X^2 + \cdots + \sigma_v X^v$$

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^1 & \alpha^0 \\ \alpha^0 & \alpha^5 \end{bmatrix} \begin{bmatrix} \alpha^6 \\ 0 \end{bmatrix} = \begin{bmatrix} \alpha^7 \\ \alpha^6 \end{bmatrix} = \begin{bmatrix} \alpha^0 \\ \alpha^6 \end{bmatrix}$$

Where σ(X) is represented as;

$$\sigma(X) = \alpha^0 + \sigma_1 X + \sigma_2 X^2 = \alpha^0 + \alpha^6 X + \alpha^0 X^2$$

As we know that the roots of σ(X) are the reciprocals of the error locations. From this fact it is evident that if roots are found out, we will get to know error location. An important point to be noted here is that σ(X) = 0 indicates an error. Keeping these points in mind, following calculations are computed to locate an error;

$$\sigma(\alpha^0) = \alpha^0 + \alpha^6 + \alpha^0 + \alpha^6 \neq 0$$
$$\sigma(\alpha^1) = \alpha^0 + \alpha^7 + \alpha^2 + \alpha^2 \neq 0$$
$$\sigma(\alpha^2) = \alpha^0 + \alpha^8 + \alpha^4 + \alpha^6 \neq 0$$
$$\sigma(\alpha^3) = \alpha^0 + \alpha^9 + \alpha^6 = 0$$
$$\sigma(\alpha^4) = \alpha^0 + \alpha^{10} + \alpha^8 = 0$$
$$\sigma(\alpha^5) = \alpha^0 + \alpha^{11} + \alpha^{10} = \alpha^2 \neq 0$$
$$\sigma(\alpha^6) = \alpha^0 + \alpha^{12} + \alpha^{12} = \alpha^0 \neq 0$$

In the above calculated expressions, $\sigma(\alpha^3) = \sigma(\alpha^4) = 0$. This means that these both locations indicate an error. To be more elaborate, we can say that one root exits at $1/\beta l = \alpha^3$. Thus, $\beta l = 1/\alpha^3 = \alpha^4$ and another root exits at $1/\beta l' = \alpha^4$. Thus, $\beta l' = 1/\alpha^4 = \alpha^3$, where l and l′ refer to the 1st, 2nd up to vth error. As we have concluded that there are two symbol errors, so e(x) can be written as;

$$e(X) = e_{j_1} X^{j_1} + e_{j_2} X^{j_2}$$

Concluding the above explanation we can say that we have come across two errors at α^3 and α^4.

Error Value

Once the error location is found out, next part is to find error values. Which means now we have to find the error values e_{j1} corresponding to the locations $\beta_1 = \alpha^3$ and $\beta_2 = \alpha^4$ It is worth noting that j corresponds to the error location and l corresponds to lth error. In order to evaluate error values we will use the equations for syndrome which were calculated in the earlier section of decoding process. It is important to note that here we can make use any of the four equations.

$$S_1 = r(\alpha) = e_{j_1}\beta_1 + e_{j_2}\beta_2$$
$$S_2 = r(\alpha^2) = e_{j_1}\beta_1^2 + e_{j_2}\beta_2^2$$

Rewriting the above equations in matrix form:

$$\begin{bmatrix} \beta_1 & \beta_2 \\ \beta_1^2 & \beta_2^2 \end{bmatrix} \begin{bmatrix} e_2 \\ e_1 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

$$\begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^8 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \alpha^3 \\ \alpha^5 \end{bmatrix}$$

As we are to find out e1 and e2 so inverse would be calculated here using the same formula applied earlier i.e. finding cofactor and determinant and then dividing them to get inverse. This calculation is done as follow;

$$\text{Inv} \begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^1 \end{bmatrix} = \frac{\begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix}}{\alpha^3\alpha^1 - \alpha^6\alpha^4} = \frac{\begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix}}{\alpha^4 + \alpha^3}$$

$$\text{Inv} \begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^1 \end{bmatrix} = \alpha^{-6} \begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix} = \alpha^1 \begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix}$$

$$\text{Inv} \begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^1 \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha^5 \\ \alpha^0 & \alpha^4 \end{bmatrix}$$

Using the above calculated inverse error values come out to be;

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha^5 \\ \alpha^0 & \alpha^4 \end{bmatrix} \begin{bmatrix} \alpha^3 \\ \alpha^5 \end{bmatrix} = \begin{bmatrix} \alpha^5 + \alpha^{10} \\ \alpha^3 + \alpha^9 \end{bmatrix} = \begin{bmatrix} \alpha^2 \\ \alpha^5 \end{bmatrix}$$

After finding error location and error values we can deduce estimated error polynomial that is represented by$\hat{e}(X)$. This estimated error polynomial is given by;

$$\hat{e}(X) = e_1 X^{j_1} + e_2 X^{j_2} = \alpha^2 X^3 + \alpha^5 X^4$$

Applying the same calculation earlier used but replacing the term e(X) with ê(X) we have;

$$\widehat{U}(X) = r(X) + \hat{e}(X) = U(X) + e(X) + \hat{e}(X)$$

Where

$$r(X) = (100) + (001)X + (011)X^2 + (100)X^3 + (101)X^4 + (110)X^5 + (111)X^6$$

And

$$\hat{e}(X) = (000) + (000)X + (000)X^2 + (001)X^3 + (111)X^4 + (000)X^5 + (000)X^6$$

Adding both terms yields;

$$\widehat{U}(X) = (100) + (001)X + (011)X^2 + (101)X^3 + (010)X^4 + (110)X^5 + (111)X^6$$
$$= \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6$$

As the message symbol is composed of k=3 right most symbols so the message that we decoded from the above expression comes out to be;

$$\underbrace{010}_{\alpha^1}\ \underbrace{110}_{\alpha^3}\ \underbrace{111}_{\alpha^5}$$

The decoded message is precisely the same that was earlier selected during the encoding process.

4.Implementation

The design of a Reed Solomon encoder and decoder in MATLAB will be discussed in this chapter. RS (255,223) code is designed in MATLAB and output is taken and analysed. In RS (255,223) code, every code word constitutes of 255 code word bytes. Out of these 255 bytes, 223 bytes are the original data and the rest 32 are for parity.The main purpose of implementing in Reed Solomon codes in MATLAB is to understand the basis working of this technique and to understand that how signal transmission occurs. During transmission if error occurs, then what happen to transmitted signal, capability of decoder, to what level decoder can detect the errors and error correction capability of the decoder. That RS code will be able to correct up to 16 symbol errors out of 255 symbols automatically. Maximum code word length for a RS code is given by the formula;

$n = 2^m - 1$

As in our case m is 8 so,

$n = 2^8 - 1$ = 255 bytes

For error correction capability, we make use of the following formula;

$$t = \frac{n-k}{2}$$

As in our case n=255 and k=223 so,

$$t = \frac{255-223}{2} = 16$$

Once the parameters have been set, we generate transmission data bits applying the random function by using code length and input bits. Now we convert these data bits to data symbols using binary to decimal convertor which is a MATLAB built in function. At this point we have obtained our message symbols to be transmitted. Now we apply Galois field function to these message symbol that is necessary for RS encoding procedure. Then the converted message symbols which are served as an input to encoder.

After the modulation process in completed, we induce an SNR vector in order to calculate bit error rate on these values of SNR. We induce a noise signal using 'rand' function and the symbol of random noise is added to encoded data. This noise signal once added to the transmitted signal, we name it as received signal. This procedure gives a look that the data travelled on channel and when received at the other end, it has induced errors in itself due to noise and other processes.

Now the received signal with the noise is demodulated and then bits in errors are calculated. BER is found by using the formula;

$$\text{BER} = \frac{\text{Total Number of Bits in Error}}{\text{Total Number of Transmitted Bits}}$$

Now the data with the addition of noise itself becomes the input of RS decoder. RS decoder can remove the error from 2t symbols from the data given at its input.

The decoder function is applied to convert the original sequence by removing errors using the redundant or parity bits added by the encoder and we receive the same message signal which was transmitted in the beginning

5.RESULTS AND DISCUSSIONS

Now the results obtained is a two dimensional graph which plots the bit error rate as a function of signal to noise ratio. To be more elaborate we can say that the plot constitutes of bit error rate on y axis and Eb/No on x axis. Eb/No is actually energy per bit divided by power spectral density of noise. It is important to note that energy per bit is defined as the total amount of energy divided by the total number of bits. Bit error rate is expressed in power of tens and Eb/No is represented in decibels.

The considerable point here is that high bit error rate indicates that more bits are in error. This results in the performance degradation for any system. So in order to have efficient performance, bit error rate should be in certain limits.

Figure 3 shows the results for the simulation of transmission of data without using any type of coding. Figure 4 shows the results for the simulation of transmission using Reed Solomon codes. Now the comparison of both plots in Figure 5 clearly shows that the curve for RS coding is beneath the curve for without coding. This clearly shows that the bit error rate is higher when transmission is done without using coding schemes which ultimately means that more number of bits are in error. Whereas if we observe the curve for RS coding, we see that it has a lower bit error rate as compared to the upper curve which means that when we transmit data using RS codes, the amount of bits in error is considerably low.

Furthermore, we can also observe bit error rates for corresponding Eb/No values. For instance at 6 decibels, RS gives a bit error rate of $10^{-1.64}$ whereas at the same value of Eb/No the curve for without channel coding gives a bit error rate of approximately $10^{-0.84}$. This shows a significant difference between to bit error rates for a same Eb/No.

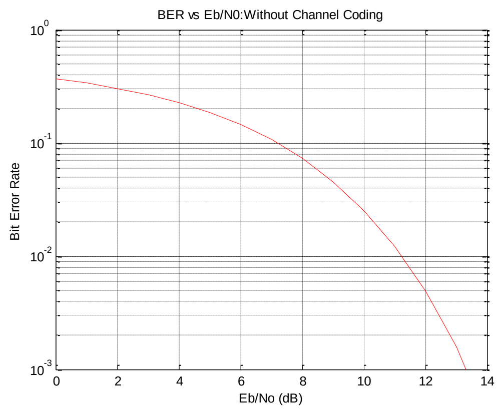

Figure 3: BER Vs Eb/No without Channel Coding

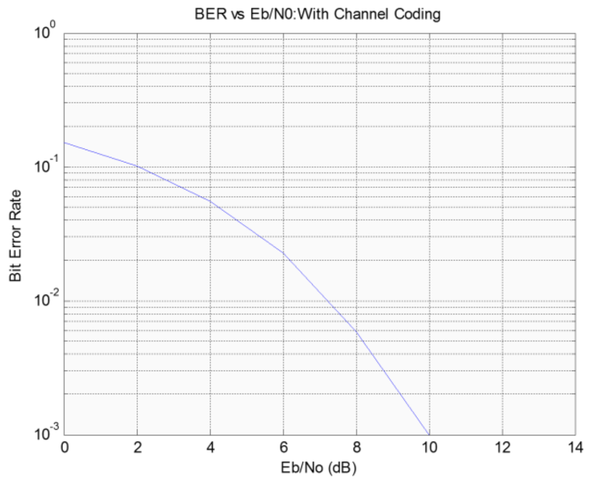

Figure 4: BER Vs Eb/No for Reed Solomon Coding

From Figure 5 we can also evaluate the coding gain. Coding gain is referred as the measure in difference between signal to noise ratio levels between the uncoded and coded systems require to achieve the same BER level. For instance, in or case the RS coded system has a bit error rate of 10^{-2} at approximately 7.2 decibels whereas the uncoded system has the same BER level at approximately 11.2 decibels. So the coding gain comes out to be 11.2db-7.2db=4db.

Bit error rate is a significant parameter in evaluating any system's performance. It evaluates a complete end to end performance for any system transmitting digital data from one side to another. The curve plotted for bit error rate against signal to noise ratio helps the communication engineers to choose the best alternative for transmission in terms of reliability. The major reasons for performance degradation are noise and alteration in propagation paths, in case of wireless, and in communication these both situations are regularly observed. So to overcome these situations we make use of error correction schemes which ensures us a very much reliable data transmission to a great extent as our results clearly depicts this point.

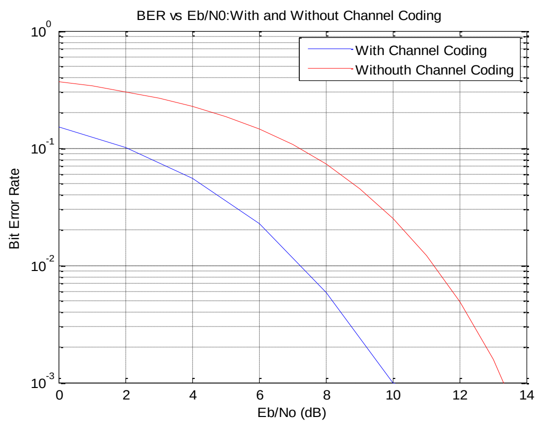

Figure 5: Comparison of Both Results

6.CONCLUSION

We presented a comparison of transmitting digital data with and without using error control coding using MATLAB. For ECC we implemented Reed Solomon (255,223) code which constructs 255 symbols of 8 bits each with the error correction capability of 16 with total. BER curves for both transmissions were obtained and compared. From the results, it is very obvious that the BER curve for Reed Solomon code is healthier as compared to the BER curve of data transmitted without coding. It was witnessed that for a distinct value of signal to noise ratio, the bits in error can be found if the total number of symbols in error is less than the error correction capability of code. We observed that RS codes are a dominant and effective class of block codes in the field of error control coding. In future, one can implement LDPC codes and can compare results given by RS code with the results of LDPC codes and can spot the difference between the two. One can also applydifferent modulation schemes and can deduce a comparison.

REFERENCES

[1] Claude Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379-423, October 1948.

[2] Jay M. Jacobsmeyer, "Introduction to Error Control Coding," Pericle Communications Company, 2004.

[3] Jorge Castineira Moreira and Patrick Guy Farrell, *Essintials of Error Control Coding*. Chichester, England: John Wiley & Sons Ltd., 2006.

[4] Daniel J. Costello, Hagenauer Joachim, Hideki Imai, and Stephen B. Wicker, "Applications of Error Control Coding," *IEEE Transactions On Information Theory*, vol. 44, pp. 2531-2560, October 1998.

[5] Daniel J. Costello and G. David Forney, "Channel Coding: The Road to Channel Capacity," *Proceedings of the IEEE*, vol. 95, pp. 1150-1117, June 2007.

[6] Qinghua Zhao, Pamela Cosman, and Lawrence B. Milstein, "Tradeoffs of Source Coding, Channel Coding and Spreading in CDMA Systems," Department of Elictrical and Computer Enginnering, University of California,.

[7] Mahesh Patel, "New Channel Coding Technique to Achieve The Ultimate Shannon Limit," in *National Conference on Recent Trends in Engineering and Technology*, 2011.

[8] Vahid Tarokh, Hamid Jafarkhani, and A. Robert Claderbank, "Space Time Block Coding for Wireless Communications: Performance Results," *IEEE Journal in Selected Areas In Communications*, vol. 17, March 1999.

[9] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, "Concurrent Error Detection in Reed–Solomon Encoders and Decoders," *IEEE Transactions on VLSI Systems*, vol. 15, pp. 842-846, July 2007.

[10] Joschi Brauchle and Ralf Koetter, "A Systematic Reed–Solomon Encoder with Arbitrary Parity Positions," in *IEEE*, 2009.

[11] William C. Cox, Jim A. Simpson, Carlo P. Domizioli, John F. Muth, and Brian L. Hughes, "An Underwater Optical Communication System Implementing Reed-Solomon Channel Coding," in OCEANS 2008, September 2008, pp. 1-6.

[12] Aqib. Al Azad, Minhazul Huq, and Iqbalur Rahman Rokon, "Efficient Hardware Implementation of Reed Solomon Encoder and Decoder in FPGA using Verilog," in International Conference on Advancements in Electronics and Power Engineering, Bangkok, 2011.

[13] Peter Trifonov, "Soft-decision decoding of polar codes with Reed-Solomon kernels," in Thirteenth International Workshop on Algebraic and Combinatorial Coding Theory, Bulgaria, 2012, pp. 317-322.

[14] N. Sireesha and V. Prasanth, "Iterative Multivariate Interpolation for Low Complexity Reed-Solomon Codes," International Journal of Modern Engineering Research, vol. 2, no. 5, pp. 3769-3772, September-October 2012.

[15] Parul Gaur, Deepak Gaur, and Aruna Tomar, "N-Byte Error Correcting and Detecting Code Using Reed Solomon and Cellular Automata Approach," International Journal of Research ion Engineering and Technology, vol. 1, no. 3, pp. 511-515, November 2012.

[16] Bernard Sklar, Digital Communications: Fundamentals and Applications, 2nd ed.: Prentice-Hall, 2001.