STUDY AND ANALYSIS OF CHANNEL ENCODER FOR SPACE COMMUNICATION SYSTEMS

MAJOR PROJECT REPORT

Submitted by:

ROHIT BADOLA [R640209033]

SONAM MANDHOTRA [R640209039]

PUNEETH PRASHANTH [R640209013]

of

BACHELOR OF TECHNOLOGY

In

AVIONICS ENGINEERING

Under the Supervision of Asst. Prof. PAVAN KUMAR NANDURI

DEPARTMENT OF AEROSPACE ENGINEERING
COLLEGE OF ENGINEERING STUDIES (CoES)
UNIVERSITY OF PETROLEUM & ENERGY STUDIES
DEHRADUN- 248007, INDIA

APRIL/MAY 2013

Dedicated to

Our Guide Mr. Pavan Kumar Nanduri

THESIS CERTIFICATE

I hereby certify that the work which is being presented in the project report entitled **"Study and Analysis of Channel Encoder for Communication Systems"** in fulfillment of the requirements for the satisfactory performance for B.Tech Avionics Engineering, Major Project submitted in the Department of Aerospace Engineering, University of Petroleum and Energy Studies, Dehradun is an authentic record of my own work carried out during a period from July 2012 to April 2013.

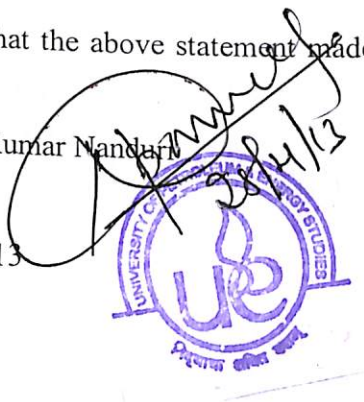
SUBMITTED BY:

Rohit Badola [R640209033]
Sonam Mandhotra [R640209039]
Puneeth Prashanth [R640209013]

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Asst. Prof. Pavan Kumar Nandurn
Guide

Date: 15 April 2013



ACKNOWLEDGEMENTS

It is with a sense of great satisfaction and pride that we are sitting down to pen out our major project thesis report. First and foremost we sincerely salute our esteemed Institution **University of Petroleum & Energy Studies, Dehradun** for giving this golden opportunity for fulfilling our warm dreams of becoming a bachelor.

The first and foremost person we would like to express our deep sense of gratitude and profound thanks to our guide **Asst. Prof. Pavan Kumar Nanduri,** Department of Aerospace Engineering, UPES Dehradun for his valuable advice, suggestions, insurmountable guidance which played a vital role in carrying out our project work successfully. Our thanks to **Prof. (Dr.) Om Prakash,** Head, Department of Aerospace Engineering, UPES, Dehradun for his valuable guidance towards our project.

We would like to thank **Asst. Prof. Sudhir Kumar Chaturvedi,** for his help & encouragement during the work. We also thank all faculty, staff, and students of Department of Aerospace Engineering and for rendering help during various stages of the project work. We would like to thanks our parent, our sisters/brothers and our friends for their support and encouragement for carrying out our project work in a well sincere manner.

Rohit Badola [R640209033]
Sonam Mandhotra [R640209039]
Puneeth Prasanth [R640209013]

ABSTRACT

Keywords: - Error Free Communication, Source coding, channel coding, Bit Error Rate, Signal to Noise ratio

Right from the invention of telephone, telecommunications field has witnessed path breaking inventions and discoveries which led way to the modern day technologies. Error Free Communication being the final frontier for the mankind, due to the advancements in space electronics (avionics), it's become feasible to design systems that can withstand the adverse weather conditions and perform well even in space. Advanced communication systems are designed to accomplish interstellar missions. In all the communication systems where data transmission is required, coding techniques are essential to make an efficient and reliable data transfer. There are two aspects of coding, source coding which is done to compress the data and channel coding which is done to make the data error free on the receiver end. The major aim of the project is to the study, analyze and evaluate the performance of channel encoder, which is currently finding their application in 3G communication and especially in deep space communications. A detailed study of all digital communication coding techniques (Forward Error Correction codes) is done and the relevant coding techniques which do have a low BER (Bit Error Rate) for the prescribed (low) SNRs are analyzed and their performance graphs are drawn using MATLAB & SIMULINK tools. This project prospects and explores the possibilities of analyzing the best coding techniques possible for error free communications.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER – 1
INTRODUCTION

1.1 Basics of Communication System

Over the past decade, the rapid expansion of digital communication technologies has been simply astounding. Internet, a word and concept once proverbial only till technologist and the scientific community, has now penetrated every aspect of people's daily life. It is quite difficult to find any individual in the modern society that has not been touched by new communication technologies ranging from Cellular phone to Bluetooth. With this it has become very important for every engineer to have a lucid understanding of communication system.

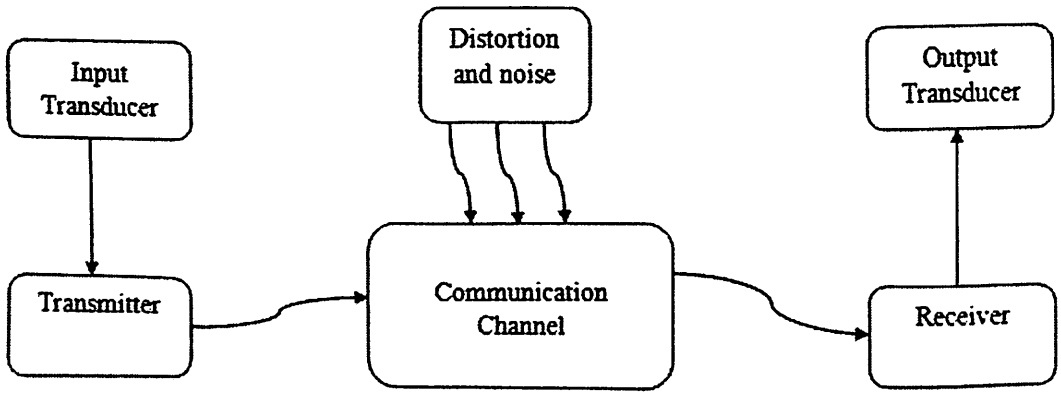


Figure 1.1: Basic Communication System

Communication System is an electrical system responsible for transfer of signal/ information from one point to another through a channel/ medium [7]. Figure 1.1 gives a brief functional block diagram of communication system. Information generated by the source can be of any form i.e. audio, picture or just a plain text in any language. The fundamental trait for source signal is that its occurrence should not be deterministic.

i. Transducer: It is a device used to convert the output of a source into an electrical signal which can then be suitable for transmission [1]. For example microphone is used to convert audio signal into electrical signal and camera is used to covert picture into electrical signal. Unlike the transmission end the function of transducer at the receiving end is to convert electrical signal into a form suitable for users.

ii. Transmitter: It is used to enhance the characteristics of the message signal in such way so as to match with the channel or transmission requirement. For example if the central

frequency of the radio station is 105 MHz, then the source signal should be transmitted at the same frequency to avoid the overlapping of signals with other frequency band allocated for other stations.

In general transmitter does this matching up of frequency by the process known as modulation. Modulation is changing of the amplitude or frequency or phase of a sinusoidal signal with respect to the amplitude of the message signal and hence correspondingly known as amplitude modulation (AM), frequency modulation (FM) and phase modulation (PM).

In addition to modulation, transmitter also performs other functions like filtering of information bearing signal, amplification of modulated signal and in case of wireless radiation of electrical signal into channel through antenna.

iii. Channel: It is a physical medium over which the transmitter transmits the signal or a physical medium over which the wave propagates for transmitter end to receiver end. For wireless transmission the medium is atmosphere of free space, while for telephone communication physical media are wire-lines or optical fiber cables. Whatever the medium may be but the essential part is transmitted signal will get tainted due to the presence of numerous noise effects present in the channel.

Channel noise can be broadly divided into two; additive noise and non-additive noise. Additive noises are those which are felt at the receiver end and keep on adding due to the receiver amplification [8]. These are thermal noise, man-made noise, atmospheric noise picked up by receiving antenna and interference from other users of channel is also a form of a noise. Non-additive noises are those which manifest it as the distance increases, this also called fading.

While doing mathematical modeling of a communication system all these constrains should be kept in mind, otherwise the system will fail to perform successfully.

iv. Receiver: The purpose of receiver is to receive the signal transmitted by the transmitting end. Transmitted signal is of the form, signal modulated, receiver's function is to demodulate the signal and to extract the information from the sinusoidal wave. As the demodulation is done in the presence of channel noise and distortion so it is apparent that received message will be a corrupted form of a transmitted signal [10]. The fidelity of the received signal depends on the type of modulation scheme used.

Apart from demodulation receiver also performs other tasks like amplification, filtering and noise separation.

1.2 Digital Communication System

Till now we have discussed about a communication system which has message signal as a continuous time varying one, such signals are often known as analog signal. Analog signals can be directly transmitted by modulating them and can be easily received by demodulating them. A communication system which uses this continuous time varying message signal is known as analog communication system.

Sometimes we use input analog signal and convert it to digital signal, then this digital signal is digitally modulated. There are many advantages of digital modulation most important is the fidelity of the message signal can be increased in digital form by adding few parity bits to it whereas in analog there is no such provision. Noise added in analog communication keeps on amplifying while amplifying the signal, whereas in digital communication we regenerate the signal on the timely basis due to which noise is completely eliminated from the regenerated signal [9]. With processing the signal digitally we have a provision to remove redundant bits and hence packing up more information in a given bandwidth.

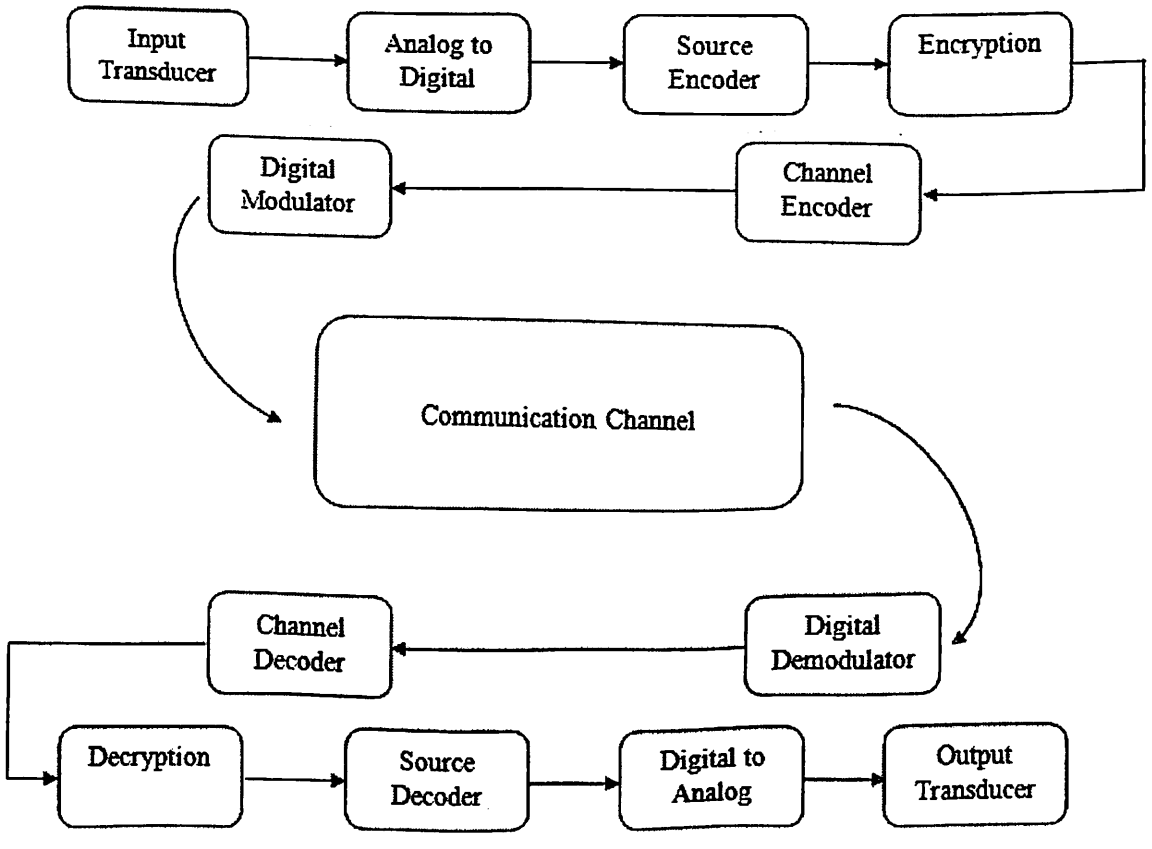


Figure 1.2: Digital Communication System

Figure 1.2 illustrates the functional diagram with basic elements of digital communication system. Few of these blocks have been discussed under 1.1, rest are explained as follows:

1. Input Transducer: Explained under 1.1
2. Analog to Digital: Output of input transducer is a time varying electrical signal which is changed to digital signal with two easy steps of sampling and quantization.
    - Sampling means to break a continuous signal into samples of not less than twice of its frequency (highest) [10]. An analog signal can be reconstructed by sending its samples rather than sending the complete signal.
    - Quantization is approximating or rounding off the sampled signal to its nearest quantized level [10].
3. Source Encoder: It is used to source encode the digital output of the A/D converter. It reduces the number of redundant bits present in the message signal and hence increases the quantity of message signal transfer for a given bandwidth.
4. Encryption: It is an act of coding a message signal in such a way that any unintended receiver cannot receive it or decode it [7]. It was initially used by spies and defense but has now become common to all to keep our data safe from pries as it is transmitted over internet.
5. Channel Encoder: The message signal coming out of the encryption is then channel coded. The purpose of channel encoder is to add some redundant bits to the message signal in a controlled manner such that by looking at these redundant bits (at the receiver end) we can remove the noise or error in the signal and can decode the signal correctly. Hence channel encoder increases the fidelity of the message signal by adding some redundant bits to it.
6. Digital Modulator: Signal coming out of channel modulator is a coded message signal in form of bits. Digital modulator acts as an interface between transmitter and communication channel, it maps the digital signal (0 and 1) by some sort of analogue signal which is then fed to antenna to transmit it to atmosphere.
   For example, bit 0 is mapped by an analogue signal s(t) and bit 1 is mapped by an analogue signal p(t).
7. Communication Channel: Discussed under 1.1 (pt iii).
8. Digital Demodulator: Its purpose is to process the received noise corrupted analog signal and to decide whether the transmitted signal was bit 0 or bit 1. Therefore digital demodulator demodulates the coded signal by making binary decision.
   Sometimes demodulator does not make a binary decision i.e. the received bit is neither bit 0 nor bit 1. Then we say that demodulator has inserted erasure in the demodulated data bits. Using the redundant bits added at the time of transmission is now used to fill the places where erasure has occurred.
9. Channel Decoder: It is used to decode the coded signal and tell whether the received signal is correct or not. This is done with the help of redundant bits which were added to it while channel encoding the signal. Today's channel decoder has the capability of both

error detection and error correction. Most example is Hamming Code (4, 7), which has message signal and 3 redundant bits added to it. This can detect error up to two bits and can correct error up to 1 bit.

10. Decryption: It is a process of decoding the signal so as to extract the actual information signal. Decryption can be done by authenticated receivers only.
11. Source Decoder: It adds the redundant bits (from the knowledge of source encoder used), removed at the time of source encoding back to the message signal. Output of Source decoder is nearly same as input of Source encoder in the transmitter section.
12. Digital to Analog: Output is given either in the form of analog signal e.g. sound, video, etc. or in the form of digital only, e.g. text or alphabets. So for analog output of signal, signal again needs to be changed to analog form, using digital to analog convertor.
13. Output Transducer: Discussed under 1.1

CHAPTER – 2
INFORMATION THEORY AND SOURCE CODING

Among all the means of communications discussed thus far none produces error free communication. We may be able to increase the accuracy in digital signals by reducing the error probability P_e. But it appears that as long as channel noise exists, our communication cannot be free from error. For example, in all the digital systems discussed thus far, P_e varies as e^{-kE_b} asymptotically. By increasing E_b, the energy per bit, we can reduce P_e to any desired level. Now the signal power is $S_i = E_b R_b$, where R_b is the bit rate. Hence, increasing E_b means either increasing the signal power S_i (for a given bit), decreasing the bit rate R_b (for a given power), or both. Because of physical limitations, however S_i cannot be increased beyond certain limit. Hence, to reduce P_e further, we must reduce R_b, the rate of transmission of information digits. Thus, the price to be paid for reducing P_e is a reduction in the transmission rate. To make P_e approach 0, R_b also approaches 0. Hence, it appears that in the presence of channel noise it is impossible to achieve the error-free communication. Thus thought communication engineers until the publication of Shannon's seminal paper in 1948. Shannon showed that for a given channel, as long as the rate of information digits per second to be transmitted is maintained within a certain limit determined by the physical channel(known as the channel capacity), it is possible to achieve error-free communication, that is, to attain $P_e \longrightarrow 0$, it is not necessary to make $R_b \longrightarrow 0$. Such a goal $(P_e \longrightarrow 0)$ can be attained by maintaining R_b below C, the channel capacity (per second). The gist of Shannon paper is that the presence of random disturbance in a channel does not, by itself, define any limit on transmission accuracy. Instead, it defines a limit on the information rate for which an arbitrarily small error probability $(P_e \longrightarrow 0)$ can be achieved.

2.1 Measure of Information

The amount of information in a message is proportional to the (minimum) time required to transmit the message. Consider transmission of alphabetic symbols in the English language using Morse code. This code is made up of various combinations of two symbols (such as a dash and dot in Morse code, or pulses of height A and $-A$ volts). Each letter is represented by a certain combination of these symbols, called the codewords, which has a certain length. Obviously, for efficient transmission, shorter codewords are assigned to the letters *e, t, a* and *o*, which occur more frequently. The longer codewords are assigned to letters *x, k, q* and *z*, which occur less frequently. Each letter may be considered to be a message. It is obvious that the letters that occur more frequently (with higher probability of occurrence) need a shorter time to transmit (shorter codewords) than those with smaller probability of occurrence. We shall now show that on the average, the time required to transmit a symbol (or a message) with probability of occurrence P is indeed proportional to $\log (1/P)$.

For sake of simplicity, let us begin with the case of binary messages m_1 and m_2, which are equally likely to occur. We may use binary digits to encode these messages, representing m_1 and m_2 by the digits **0** and **1**, respectively. Clearly, we must have a minimum of one binary digit (which can assume two values) to represent each of the two equally likely messages. Next, consider the case of the four equiprobable messages m_1, m_2, m_3, and m_4. If these messages are encoded in binary form, we need a minimum of two binary digits per message. Each binary digit can assume two values. Hence, a combination of two binary digits can form the four codewords **00, 01, 10, 11**, which can be assigned to the four equiprobable messages m_1, m_2, m_3, and m_4, respectively. It is clear that each of these four messages takes twice as much transmission time as that required by each of the two equiprobable messages and, hence, contains twice as much information. Similarly, we can encode any one of the eight equiprobable messages with a minimum of three binary digits. This is because three binary digits form eight distinct codewords which can be assigned to each of the eight messages. It can be seen that, in general, we need $\log_2 n$ binary digits to encode each of n equiprobable messages. Because all the messages are equiprobable, P, the probability of any one message occurring, is $1/n$. hence, to encode each message (with probability P), we need $\log_2(1/P)$ binary digits. Thus, from the engineering view point, the information I contained in a message with probability of occurrence P is proportional to $\log_2(1/P)$,

$$I = k \log_2 (1/P) \qquad \text{Eq.(2.1)}$$

Where k is a constant to be determined. The conclusion is that the information content of a message is proportional to the logarithm of the reciprocal of the probability of the message.

2.2 Sampling

Sampling is the reduction of a continuous signal to a discrete signal. A common example is the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal). A sample refers to a value or set of values at a point in time and/or space.

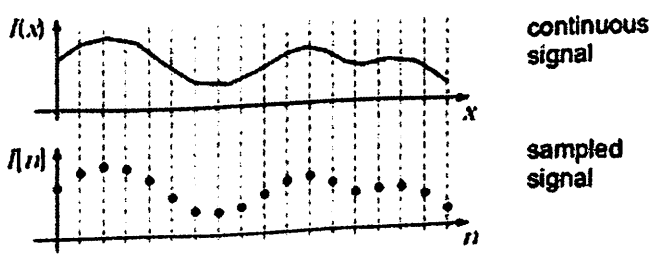


Figure 2.1: Sampling Process

Sampling can be done for functions varying in space, time, or any other dimension, and similar results are obtained in two or more dimensions. For functions that vary with time, let $s(t)$ be a continuous function (or "signal") to be sampled, and let sampling be performed by measuring the value of the continuous function every T seconds, which is called the sampling interval. Thus, the sampled function is given by the sequence: $s(nT)$, for integer values of n.

The sampling frequency or sampling rate f_s is defined as the number of samples obtained in one second (samples per second), thus $f_s = 1/T$.

Reconstructing a continuous function from samples is done by interpolation algorithms. The Whittaker–Shannon interpolation formula is mathematically equivalent to an ideal lowpass filter whose input is a sequence of Dirac delta functions that are modulated (multiplied) by the sample values. When the time interval between adjacent samples is a constant (T), the sequence of delta functions is called a Dirac comb. Mathematically, the modulated Dirac comb is equivalent to the product of the comb function with $s(t)$. That purely mathematical function is often loosely referred to as the sampled signal.

Most sampled signals are not simply stored and reconstructed. But the fidelity of a theoretical reconstruction is a customary measure of the effectiveness of sampling. That fidelity is reduced when $s(t)$ contains frequency components higher than $f_s/2$ Hz, which is known as the Nyquist frequency of the sampler. Therefore $s(t)$ is usually the output of alowpass filter, functionally known as an "anti-aliasing" filter. Without an anti-aliasing filter, frequencies higher than the Nyquist frequency will influence the samples in a way that is misinterpreted by the interpolation process.

Various types of distortion can occur at the time of reconstruction of signal, including:

- Aliasing. A precondition of the sampling theorem is that the signal be bandlimited. However, in practice, no time-limited signal can be bandlimited. Since signals of interest are almost always time-limited (e.g., at most spanning the lifetime of the sampling device in question), it follows that they are not bandlimited. However, by designing a sampler with an appropriate guard band, it is possible to obtain output that is as accurate as necessary.
- Integration effect or aperture effect. This results from the fact that the sample is obtained as a time average within a sampling region, rather than just being equal to the signal value at the sampling instant. The integration effect is readily noticeable in photography when the exposure is too long and creates a blur in the image. An ideal camera would have an exposure time of zero. In a capacitor-based sample and hold circuit, the integration effect is introduced because the capacitor cannot instantly change voltage thus requiring the sample to have non-zero width.
- Jitter or deviation from the precise sample timing intervals.
- Noise, including thermal sensor noise, analog circuit noise, etc.
- Slew rate limit error, caused by the inability of the ADC input value to change sufficiently rapidly.
- Quantization as a consequence of the finite precision of words that represent the converted values.

- Error due to other non-linear effects of the mapping of input voltage to converted output value (in addition to the effects of quantization).

2.3 Quantization

It is the process of mapping a large set of input values to a smaller set – such as rounding values to some unit of precision. A device or algorithmic function that performs quantization is called a quantizer. The error introduced by quantization is referred to as quantization error or round-off error. Quantization is involved to some degree in nearly all digital signal processing, as the process of representing a signal in digital form ordinarily involves rounding. Because quantization is a many-to-few mapping, it is an inherently non-linear and irreversible process (i.e., because the same output value is shared by multiple input values, it is impossible in general to recover the exact input value when given only the output value).

Scalar Quantization
The most common type of quantization is known as scalar quantization. Scalar quantization, typically denoted as $y = Q(x)$, is the process of using a quantization function $Q()$ to map a scalar (one-dimensional) input value x to a scalar output value y. Scalar quantization can be as simple and intuitive as rounding high-precision numbers to the nearest integer, or to the nearest multiple of some other unit of precision (such as rounding a large monetary amount to the nearest thousand dollars). Scalar quantization of continuous-valued input data that is performed by an electronic sensor is referred to as analog-to-digital conversion. Analog-to-digital conversion often also involves sampling the signal periodically in time (e.g., at 44.1 kHz for CD-quality audio signals).

2.4 Source Encoding

The minimum number of binary digits required to encode a message was shown to be equal to the source entropy $log(1/P)$ if all the message of the source are equiprobable (each message probability is P). We shall now generalize this result to the case of nonequiprobable message. We shall now show that the average number of binary digits per message required for encoding is given by $H(m)$ (in bits) for an arbitrary probability distribution of the messages.

Let a source m emit messages $m_1, m_2, \ldots, m_n$ with probability $P_1, P_2, \ldots, P_n$, respectively. Consider a sequence of $N \to \infty$. Lets k_i be the number of times messages m_i occur in the sequence. Then according to the relative frequency interpretation (or law of large number),

$$\lim_{N \to \infty} \frac{k_i}{N} = P_i \qquad \text{Eq.(2.2)}$$

Thus the message m_i occurs NP_i times in a sequence of N messages (provided $N \rightarrow \alpha$). Therefore, in a typical sequence of N messages, m_1 will occur NP_1 times, m_2 will occur NP_2 times,.....,m_n will occur NP_n times. All other composition are extremely unlikely to occur ($P \rightarrow 0$).Thus any typical sequence (where $N \rightarrow \alpha$) has the same proportion of the n messages, although in general the order will be different. We shall assume a memoryless source: that is, we assume that the message is emitted from the source independent of the previous messages. Consider now a typical sequence S_n of N messages from the source. Because of the n messages (of probability $P1,P2,...Pn$) occur $NP_1,NP_2,.....NP_n$ times, and because each message in independent, the probability of occurrence of a typical sequence S_N is given by

$$P(S_N) = (P_1)^{NP_1}(P_2)^{NP_2} \dots\dots\dots\dots (P_n)^{NP_n} \qquad \text{Eq.(2.3)}$$

Because all possible sequences of N messages from this source have the same composition, all the sequences (of N messages) are equiprobable, with probability $P(S_N)$. We can consider these long sequences as new messages (which are now equiprobable). To encode one such sequences we need L_N binary digits, where

$$L_N = log\,[1/P(S_N)] \quad \text{binary digits} \qquad \text{Eq.(2.4)}$$

$$L_N = N\sum_{i=1}^{n} P_i log\frac{1}{P_i} = NH(m) \quad \text{binary digits} \qquad \text{Eq.(2.5)}$$

Note that L_N is the length (number of binary digits) of the codeword required to encode N messages in sequence. Hence, L, the average numbers of digits required per message, is L_N/N and is given by

$$L = L_N/N = H(m) \quad \text{binary digits} \qquad \text{Eq.(2.6)}$$

Thus, by encoding N successive messages, it is possible to encode a sequence of source messages using, on the average, $H(m)$ binary digits per message, where $H(m)$ is the entropy of the source message (in bits). Moreover, one can show that $H(m)$ is indeed, on the average, the minimum number of digits required to encode this message source. It is impossible to find any uniquely decodable code whose average length is less than $H(m)$.

2.5 Huffman Coding

Huffman coding is a popular method for compressing data with variable-length codes. Given a set of data symbols (an alphabet) and their frequencies of occurrence (or, equivalently, their probabilities), the method constructs a set of variable-length codewords with the shortest average length and assigns them to the symbols. Huffman coding serves as the basis for several applications implemented on popular platforms. Some programs use just the Huffman method, while others use it as one step in a multistep compression process.

Lets us consider Huffman coding with the help of an e.g. we have 5 message symbols, w_1, w_2, w_3, w_4, w_5 with the probability of occurrence as 0.5, 0.2, 0.15, 0.1, 0.05 respectively.

Steps to be followed are discussed below and also represented in the adjacent figure.

- Arrange the symbols with descending order of probability.
- Add last two probabilities in the list.
- Now rearrange them again on the basis of new probability.
- Again add last two probabilities.
- Keep on repeating the above steps until you are left with only two probabilities.
- Now to the left two probabilities assign bit 0 to the probability with less weightage and bit 1 to the other probability.
- Carry on this bit to the two bits by whose addition you have achieved this probability.
- Again assign bit 0 and bit 1 to the two new probabilities (which have received the carry forward bits).
- The process will keep on going till you reached the original symbols or probability values.

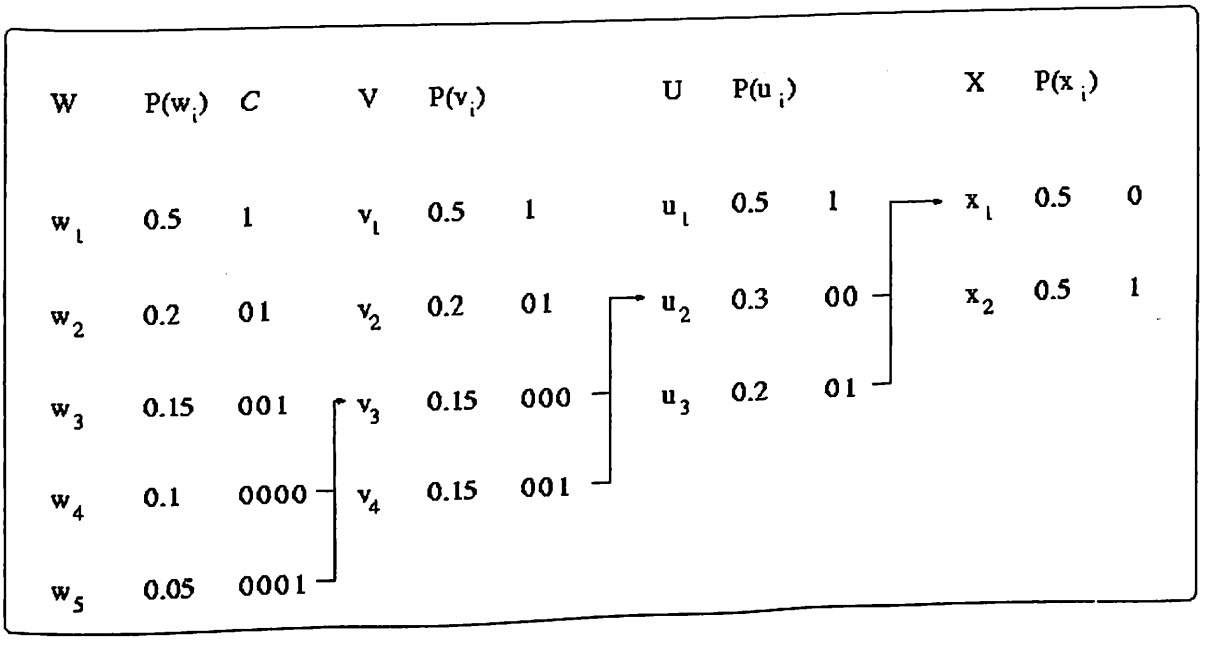


Figure 2.2: Huffman Coding

You will notice that all symbols are assigned with different number of bits. w_1 with 1 bit, w_2 with 2 bits, w_3 with 3 bits, w_4 with 4 bits, w_5 with 4 bits. Over all bits used are 14 instead of 15. Therefore packing of more data in a given bandwidth.

CHAPTER – 3
FORWARD ERROR CONTROL

Forward error correction code or Channel coding is a technique to add redundant bits (parity bits) to the message signal such that by looking at those redundant bits at the receiving end, errors can be detected and in some cases even corrected without retransmission [10]. There is no need to retransmit the data again if found corrupted but this is done at a cost of fixed, higher channel bandwidth. Therefore this technique is used where retransmission of data is very costly or impossible.

FEC is an integral part of receiver and is applied to a large bit stream. Many FEC coders have an ability to generate Bit Error Rate which can be given to FEC as a feedback to fine tune it.

FEC is accomplished by adding redundant bits to the message signal if the out after adding the redundant bits are same as the signal then it is known as systematic coding and if the code changes after coding then it is known as non – systematic coding. The most common example of FEC is Repetition code. In repetition code we add a no of bits to the message signal which same as message bits, depending on the bits added the potential of the code for number of error detection can be judged. For example 000, which has 0 as message signal and 00 as redundant bits can correct upto one bit error.

Forward Error Correction codes are divided into two major categories:

1. Block Codes

2. Convolution Codes

3.1 Block Codes

Block codes are a class of forward error correction codes (i.e. they detect and correct the errors) which takes input as a message block of bits and gives output as coded block of bits [10]. Let us assume that the output of an information source is a sequence of binary digits "0" or "1." In block coding, this binary information sequence is segmented into message blocks of fixed length; each message block, denoted by u, consists of k information digits. There are a total of 2k distinct messages. The encoder, according to certain rules, transforms each input message u into a binary n-tuple v with n> k. This binary n-tuple v is referred to as the code word (or code vector) of the message u, as shown in Figure 3.1.

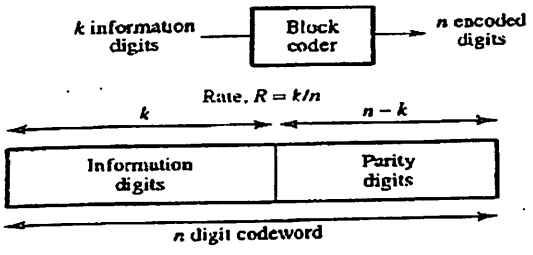


Figure 3.1: Block Code

Therefore, corresponding to the 2k possible messages, there are 2k code words. This set of 2k code words is called a block code. For a block code to be useful, the 2k code words must be distinct. Therefore, there should be a one-to-one correspondence between a message u and its code word v.

A block code of length n and 2k code words is called a linear (n, k) code if and only if its 2k code words form a k-dimensional subspace of the vector space of all the n-tuples over the field GF(2). In fact, a binary block code is linear if and only if the modulo-2 sum of two code words is also a code word. The block code given in Table 1 is a (7, 4) linear code. One can easily check that the sum of any two code words in this code is also a code word.

Table 3.1: Linear Block Code

$k = 4$ AND $n = 7$

Messages	Code words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

3.1.1 Minimum Distance of a Block Code

In this section an important parameter of a block code called the minimum distance is introduced. This parameter determines the random-error-detecting and random-error-correcting capabilities of a code. Let **v** = (v1, v2,. . ,vn - 1) be a binary ntuple. The Hamming weight (or simply weight) of **v**, denoted by w(v), is defined as the number of nonzero components of **v** [2]. For example, the Hamming weight of **v** = (1 0 0 1 0 1 1) is 4. Let **v** and **w** be two n-tuples. The Hamming distance (or simply distance) between **v** and **w**, denoted d(**v**, **w**), is defined as the number of places where they differ. For example, the Hamming distance between **v** = (1 0 0 1 0 1 1) and **w** = (0 1 0 0 0 1 1) is 3; they differ in the zeroth, first, and third places. It follows from the definition of Hamming distance and the definition of modulo-2 addition that the Hamming distance between two n-tuples, **v** and **w**, is equal to the Hamming weight of the sum of **v** and **w**, that is,

$$d(v,w) = w(v+w) \qquad \text{Eq.(3.1)}$$

For example, the Hamming distance between **v** = (1 0 0 1 0 1 1) and **w** = (1 1 1 0 0 1 0) is 4 and the weight of **v** + **w** = (0 1 1 1 0 0 1) is also 4.

Given a block code C, one can compute the Hamming distance between any two distinct code words. The minimum distance of C, denoted dmin, is defined as

$$d_{min} = \min \{d(v,w) : v, w \in C, v \neq w\} \qquad \text{Eq.(3.2)}$$

If C is a linear block code, the sum of two vectors is also a code vector [2]. It follows that the Hamming distance between two code vectors in C is equal to the Hamming weight of a third code vector in C. Then it follows that

$$\begin{aligned} d_{min} &= \min \{w(v + w): v, w \in C, v \neq w\} \\ &= \min \{w(x) : x \in C, x \neq 0\} \\ &\approx w_{min} \end{aligned} \qquad \text{Eq.(3.3)}$$

The parameter w_{min} $\{w(x): x \in C, x \neq 0\}$ is called the minimum weight of the linear code C. Summarizing the result above, we have the following:

"The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words" [2] therefore, for a linear block code, to determine the minimum distance of the code is equivalent to determining its minimum weight. The (7, 4) code given in Table 1 has minimum weight 3; thus, its minimum distance is 3.

3.1.2 Error Correcting and Error Detecting Capability

If a block code C with minimum distance dmin is used for random-error correction, one would like to know how many errors that the code is able to correct. The minimum distance dmin is either odd or even. A block code with minimum distance dmin guarantees correcting all the error patterns of t = [(dmin — 1)/2] or fewer errors, where [(dmin — 1)/2] denotes the largest integer no greater than (dmin — 1)/2. The parameter t = [(dmin — 1)/2] is called the random-error-correcting capability of the code. The code is referred to as a t-error-correcting code. The (7, 4) code given in Table 1 has minimum distance 3 and thus t = 1. It is capable of correcting any error pattern of single error over a block of seven digits. A block code with random-error-correcting capability t is usually capable of correcting many error patterns of t + 1 or more errors [6].

For a t error-correcting (n, k) linear code, it is capable of correcting a total 2n-k error patterns, including those with t or fewer errors . In practice, a code is often used for correcting λ or fewer errors and simultaneously detecting L (L > λ) or fewer errors. That is, when λ or fewer errors occur, the code is capable of correcting them; when more than λ but fewer than L + 1 errors occur, the code is capable of detecting their presence without making a decoding error. For this purpose, the minimum distance dmin of the code is at least λ + L + 1 .Thus, a block code with dmin = 10 is capable of correcting three or fewer errors and simultaneously detecting six or fewer errors. From the discussion above, we see that random-error-detecting and random-error-correcting capabilities of a block code are determined by the code's minimum distance. Clearly, for given n and k, one would like to construct a block code with minimum distance as large as possible, in addition to the implementation considerations.

3.2 Hamming Code

Hamming codes are a family of linear error-correcting codes that generalize the Hamming (7, 4) code. Hamming codes can detect up to two and correct up to one bit errors [2]. By contrast, the simple parity code cannot correct errors, and can detect only an odd number of errors. Hamming codes are special in that they are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance 3. In mathematical terms, Hamming codes are a class of binary linear codes. For each integer $r \geq 2$ there is a code with block length $n= 2^r-1$ message length $k= 2^r-r-1$. Hence the rate of Hamming codes is $R= k/n = 1 - r/ (2^r-1)$, which is highest possible for codes with distance 3 and block length $2^r - 1$. The parity-check matrix of a Hamming code is constructed by listing all columns of length r that are pair-wise linearly independent.

Because of the simplicity of Hamming codes, they are widely used in computer memory (ECC memory). In this context, one often uses an extended Hamming code with one extra parity bit. Extended Hamming codes achieve a distance of 4, which allows the decoder to distinguish between the situation in which at most one bit error occurred and the situation in which two bit errors occurred. In this sense, extended Hamming codes are single-error correcting and double-error detecting, and often referred to as SECDED.

Hamming Code (4, 7) - It encodes 4 data bits into 7 bits by adding three parity bits. Hamming (7, 4) can detect and correct single-bit errors. With the addition of an overall parity bit, it can also detect (but not correct) double-bit errors [8].

Construction of G and H:

The matrix $G = (I_k | - A^T)$ is called a (Canonical) generator matrix of a linear (n, k) code and $H = (A | I_{n-k})$ is called a parity-check matrix. This is the construction of G and H in standard (or systematic) form. Regardless of form, G and H for linear block codes must satisfy

$$HG^T = 0, \text{ an all-zeros matrix.}$$

Since (7, 4, 3) = (n,k,d)= $[2^m - 1, 2^m - 1 - m, m]$. The parity-check matrix H of a Hamming code is constructed by listing all columns of length m that are pair-wise independent.

Thus H is a matrix whose left side is all of the nonzero n-tuples where order of the n-tuples in the columns of matrix does not matter. The right hand side is just the (n-k)-identity matrix. So G can be obtained from H by taking the transpose of the left hand side of H with the identity k-identity matrix on the left hand side of G.

The code generator matrix G and the parity-check matrix H are:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}_{(4,7)}$$

and

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}_{(3,7)}$$

Finally, these matrices can be mutated into equivalent non-systematic codes by the following operations. Column permutations (swapping columns), Elementary row operations (replacing a row with a linear combination of rows)

Hamming Codes with Addition Bit Parity - The Hamming (7,4) can easily be extended to an (8,4) code by adding an extra parity bit on top of the (7,4) encoded word. This can be summed up with the revised matrices:

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}_{(4,8)}$$

and

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}_{(4,8)}$$

Note that H is not in standard form. To obtain G, elementary row operations can be used to obtain an equivalent matrix to H in systematic form:

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{(4,8)}$$

For example, the first row in this matrix is the sum of the second and third rows of H in non-systematic form. Using the systematic construction for Hamming codes from above, the matrix a is apparent and the systematic form of G is written as

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}_{(4,8)}$$

The non-systematic form of G can be row reduced (using elementary row operations) to match this matrix. The addition of the fourth row effectively computes the sum of all the codeword bits (data and parity) as the fourth parity bit.

Hamming is also known for developing the concept of a "distance". The distance is defined as the minimum number of bits that would have to flip in order to go from one codeword (error free word) to another [3]. All error correcting mechanisms correct to the nearest codeword to the received string. Therefore the maximum number of errors a code can reliably fix is less than half of the distance between the closest two codewords. Any more mistakes and it could potentially decode incorrectly. Hamming proves that this is the most efficient use of parity checks for single detection, correction and double detection since he maximizes the distance between codewords. Hamming codes have one distinct problem. They are relatively inefficient when sending small amounts of data, but they get increasingly inaccurate as the number of bits increases. They can only correctly locate one flipped bit for each codeword regardless of its length [10]. Therefore you almost always have to encode strings of length n with about n parity checks in order to ensure accuracy of information.

3.3 Bose Chaudary Hocquenghem Code

Bose Chaudary Hocquenghem Code or BCH codes form a class of cyclic error-correcting codes that are constructed using finite fields [6]. One of the key features of BCH codes is that during code design, there is a precise control over the number of symbol errors correctable by the code. In particular, it is possible to design binary BCH codes that can correct multiple bit errors. Another advantage of BCH codes is the ease with which they can be decoded, namely, via an algebraic method known as syndrome decoding. This simplifies the design of the decoder for

these codes, using small low-power electronic hardware. BCH codes are used in applications like satellite communications, compact disc players, DVDs,

For any positive integer i, let $m_i(x)$ the minimal polynomial of α^i. The generator polynomial of the BCH code is defined as the least common multiple $g(x) = lcm(m_1(x),.....,m_{d-1}(x)$. It can be seen that $g(x)$ is a polynomial with coefficients in. $GF\ (q)$ and divides x^n-1. Therefore, the polynomial code defined by $g(x)$ is a cyclic code.

For example, let $q = 2$ and $m= 4$(therefore $n = 15$). Consider different values of d.There is a primitive root $\alpha \in GF(16)$ satisfying $\alpha^4+\alpha+1 = 0$, its minimal polynomial over $GF(2)$ is $m(x)=x^4+x+1$. Note that is $GF\ (2^4)$, the equation $(a+b)^2 = a^2+ ab+ ab +b^2= a^2+b^2$ holds, and therefore $m_1(\alpha^2) = m_1(\alpha)^2= 0$. Thus α^2 is a root of $m_1(x)$ and $m_2(x)= m_1(x)= x^4+x+1$.

To compute $m_1(x)$, notice that, by repeated application of above equation, we have the following linear relations,

- $1= 0\alpha^3 + 0\alpha^2 + 0\alpha+ 1$ Eq.(3.4a)
- $\alpha^3= 1\ \alpha^3+ 0\alpha^2+0\alpha+1$ Eq.(3.4b)
- $\alpha^6= 1\ \alpha^3+ 1\alpha^2+0\alpha+0$ Eq.(3.4c)
- $\alpha^9= 1\ \alpha^3+ 0\alpha^2+1\alpha+0$ Eq.(3.4d)
- $\alpha^{12}= 1\ \alpha^3+1\ \alpha^2+1\alpha+1$ Eq.(3.4e)

Five right hand sides of length four must be linearly dependent, and indeed we find a linear dependency $\alpha^{12}+ \alpha^9+ \alpha^6+\alpha^3+1 =0$.
Since there is no smaller degree dependency, the minimal polynomial of α^3 is

$$m_3(x) = x^4+ x^3+x^2+ x+1 \quad \text{Eq.(3.5a)}$$

$$m_4(x) =m_2(x) =m_1(x)= x^4+x+1 \quad \text{Eq.(3.5b)}$$

$$m_5(x) = x^2+x+1 \quad \text{Eq.(3.5c)}$$

$$m_6(x) = m_3(x) = x^4+ x^3+x^2+ x+1 \quad \text{Eq.(3.5d)}$$

$$m_7(x) = x^4+ x^3+1. \quad \text{Eq.(3.5e)}$$

The BCH code with d= 1,2,3 has generator polynomial $g(x)= m_1(x)= x^4+ x+1$.

It has minimal Hamming distance at least 3 and corrects up to 1 error. Since the generator polynomial is of degree 4, this code has 11 data bits and 4 checksum bits. The BCH code with $d= 4,5$ has generator polynomial

$$g(x) = lcm(m_1(x), m_3(x)) = (x^4+ x+1)(x^4+ x^3+x^2+ x+1) =x^8+ x^7+ x^6+ x^4+1. \quad \text{Eq.(3.6)}$$

It has minimal Hamming distance at least 5 and corrects up to 2 errors. Since the generator polynomial is of degree 8, this code has 7 data bits and 8 checksum bits.

3.4 Reed Solomon Code

Reed–Solomon (RS) codes are non-binary cyclic error-correcting codes [1]. They described a systematic way of building codes that could detect and correct multiple random symbol errors. By adding t check symbols to the data, an RS code can detect any combination of up to t erroneous symbols, or correct up to $\lfloor t/2 \rfloor$ symbols. As an erasure code, it can correct up to t known erasures, or it can detect and correct combinations of errors and erasures. Furthermore, RS codes are suitable as multiple-burst bit-error correcting codes, since a sequence of $b + 1$ consecutive bit errors can affect at most two symbols of size b. The choice of t is up to the designer of the code, and may be selected within wide limits.

In Reed–Solomon coding, source symbols are viewed as coefficients of a polynomial $p(x)$ over a finite field. The original idea was to create n code symbols from k source symbols by oversampling $p(x)$ at $n > k$ distinct points, transmit the sampled points, and use interpolation techniques at the receiver to recover the original message. That is not how RS codes are used today. Instead, RS codes are viewed as cyclic BCH codes, where encoding symbols are derived from the coefficients of a polynomial constructed by multiplying $p(x)$ with a cyclic generator polynomial [3]. This gives rise to efficient decoding algorithms (described below).

Reed–Solomon codes have since found important applications from deep-space communication to consumer electronics [1]. They are prominently used in consumer electronics such as CDs, DVDs, Blue-ray Discs, in data transmission technologies such as DSL and WiMAX, in broadcast systems such as DVB and ATSC, and in computer applications such as RAID 6 systems.

Format of Reed-Solomon Codes- The basic building block of Reed-Solomon codes is a symbol composed of m binary bits, where m can be any natural number greater than 2. For a given m, the length of all the Reed-Solomon codes composed of m-bit symbols is $2^m - 1$.

For example, for 8-bit symbols, the length of the Reed-Solomon codes is $2^8 - 1 = 255$.

A complete Reed-Solomon code consists of two parts: the data part and the parity part. For a Reed-Solomon code of n symbols, the first k symbols is the data part, which is the information to be protected against corruption, and the following (n-k) symbols is the parity part, which is calculated based on the data part. Such a Reed-Solomon code is referred to as an (n, k) Reed-Solomon code, or RS (n,k) code. The number of parity symbols is (n-k), usually an even number represented as 2t. A Reed-Solomon code with 2t parity symbols has the capability of correcting up to t error symbols. When the length of a Reed-Solomon code needs to be less than $2^m - 1$, zero padding is used to make the length of the code $2^m - 1$. After encoding, the padded zeros is removed to form a so called shortened Reed-Solomon code. For example, for an 8-bit Reed-Solomon code, we have 100 data bytes to be protected against up to 8 errors. Then, we will need 16 parity bytes. The total length of the code will be 116 bytes, which is less than 255. To calculate the 16 parity bytes, we need to pad 139 zeros to the data bytes and then encode the 239 total bytes. After encoding, the padded zeros will be removed and only the original data bytes

and the calculated parity bytes will be transmitted or stored. When decoding the code, the removed padding zeros will be added to the code first and then do the decoding.

Encoding of Reed-Solomon Codes- A Reed-Solomon code is defined by its generator polynomial. For a t error correcting Reed-Solomon code, its generator polynomial is

$$g(X) = (X + a^{m_0})(X + a^{m_0+1})(X + a^{m_0+2})\ldots(X + a^{m_0+2t-1}) = g_0 + g_1X + g_2X^2 + \ldots + g_{2t-1}X^{2t-1} + X^{2t},$$

where a, an m-bit binary symbol, is the primitive element of the finite field $GF(2^m)$, and m_0 is a pre-set number, usually 0 or 1. To encode a message sequence, the message polynomial is first constructed as

$$\mathbf{u}(X) = u_0 + u_1X + u_2X^2 + \ldots + u_{k-1}X^{k-1},$$

where $k = n - 2t$. The parity polynomial is calculated as the remainder of $X^{2t}.a(X)/g(X)$, which is represented as

$$\mathbf{v}(X) = v_0 + v_1X + v_2X^2 + \ldots + v_{2t-1}X^{2t-1}.$$

The parity sequence is the coefficients of the parity polynomial. The code is constructed as the data sequence followed by the parity sequence. The final code polynomial is

$$\mathbf{t}(X) = X^{2t}\,\mathbf{u}(X) + \mathbf{v}(X). \qquad \text{Eq.(3.7)}$$

Reed-Solomon codes are widely used in digital communication systems and digital information storage systems. Digital communication systems that use Reed-Solomon codes for error protection include wireless communication systems, broad band communication systems, digital video broadcasting systems, and space and deep space communication systems. Digital information storage systems that use Reed-Solomon codes for error protection include CD storage systems, DVD storage systems, and hard disc storage systems. One recent development is that Reed-Solomon codes start to find its use in dynamic memory protection applications.

CHAPTER – 4
SYSTEM DESIGN AND ANALYSIS

4.1 Overview of System Design

In this section we are intended to discuss the step wise designing of a communication system and analyzing the behavior of both uncoded and coded signals with the help of matlab simulation. We will be looking at behavior of both the uncoded signal and the coded signal, individually, in a communication channel. Signals are coded by using 'block codes' and a difference in performance is examined between 'Hamming code', 'BCH code' and 'Reed Solomon code'.

Given below is the theoretical guideline on whose basis the further communication system is designed thus it is very important to understand the functioning of the system and function of each of its block-set.

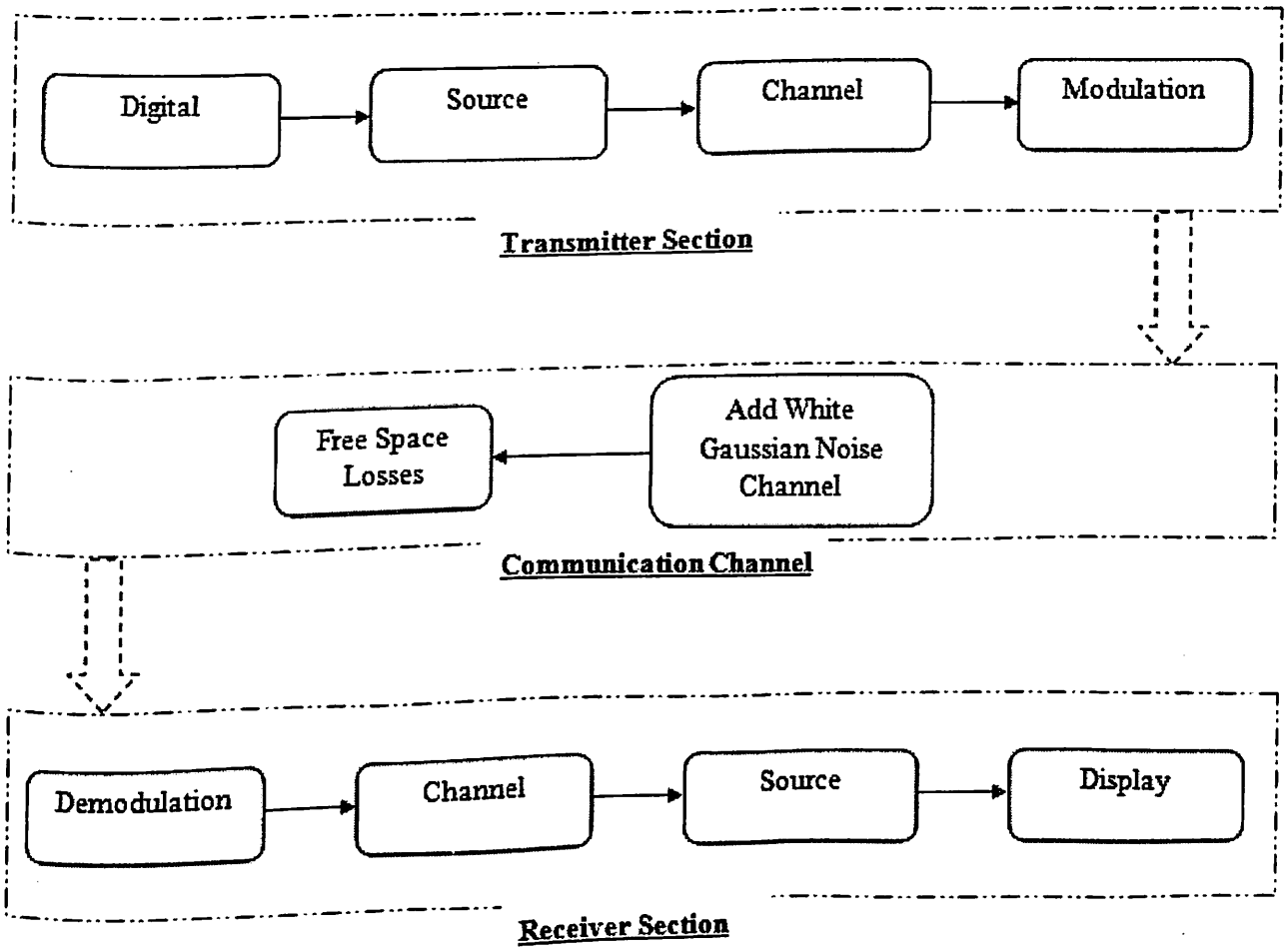


Figure 4.1: Model Overview

i. Digital Information: - Information will be given to the system in the digital form. You can also feed an analogue (can be real time) signal and then change it to digital using analog-to-digital convertor.

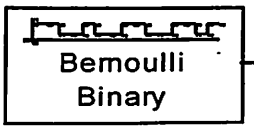


Figure 4.2: Bernoulli Binary Generator

We are 'Bernoulli binary generator' as our digital information source. It produces binary data using 'Bernoulli Distribution'.

Input parameters: - It has a provision to set the probability (let p) of occurrence of 'zero' in the sequence, therefore probability of occurrence of 'one' becomes 1- p (where p is any number from 0 to 1).

A provision to give 'initial seed' whose function is to generate an initial pseudo random code.

Output parameter: - Output signal can be of any of these three forms sample-based one-dimensional array, a sample-based row or column vector, or a frame-based matrix.

ii. Source Encoder: - It is used to decrease the redundancy of the bits and thus allowing more information to transfer through a given channel.

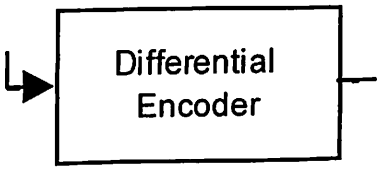


Figure 4.3: Differential Encoder

Differential Encoder is used as a Source encoder. It encodes the binary input signal within a channel and gives the output as the logical difference between the previous output element and the current input element.

iii. Channel Encoder: - It does exactly the opposite of source encoder. It adds redundant bits to the information signal such that by looking at those redundant bits on the receiver side we can detect and correct the signal.
We will be using three different channel encoders as discussed below:

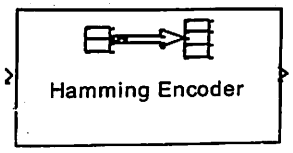


Figure 4.4: Hamming Encoder

a) Hamming Encoder: - It creates a Hamming code with message length 'K' and code length 'N'. The number N must have the form 2^M-1, where M is an integer greater than or equal to 3. Then K equals N-M.
The block takes column vector of length K as input and gives column vector of length N as output by adding redundant bits to it.

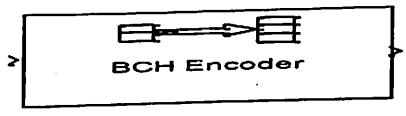


Figure 4.5: BCH Encoder

b) BCH Encoder: - The BCH Encoder block creates a BCH code with message length K and codeword length N. There is a provision to specify both N and K directly in the dialog box.
Block inducts column vector of length K as input and consider it as a message word to be encoded. It gives the output as a column vector of length N. N must be of the form 2^M-1, where $3 \leq M \leq 16$.

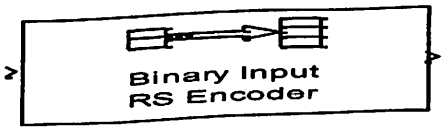


Figure 4.6: Binary Input RS Encoder

c) Binary Input RS Encoder: - This block creates a Reed-Solomon code with message length, K, and codeword length, N. N and K can directly be fed in the dialog box. The symbols for the code are binary sequences of length M, corresponding to elements of the Galois field GF (2^M), where the first bit in each sequence is the most significant bit. The difference N-K must be an even integer.

iv. Modulation: - The digital signal then modulated to a high frequency analogue signal which can now be radiated to atmosphere for propagation.

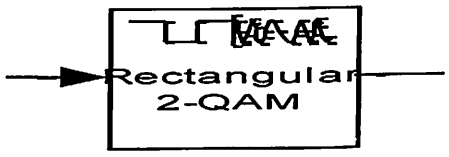


Figure 4.7: Rectangular QAM Modulator Baseband

We are using Rectangular QAM Modulator Baseband. This block modulates using M-ary quadrature amplitude modulation with a constellation on a rectangular lattice. The output is a baseband representation of the modulated signal. This block accepts a scalar or column vector input signal. When you set the Input type parameter to Integer, the block accepts integer values between 0 and M-1. M represents the M-ary number block parameter. When you set the Input type parameter to Bit, the block accepts binary-valued inputs that represent integers. The block collects binary-valued signals into groups of $K = \log_2(M)$ bits where K represents the number of bits per symbol. The input vector length must be an integer multiple of K. In this configuration, the block accepts a group of K bits and maps that group onto a symbol at the block output. The block outputs one modulated symbol for each group of K bits.

v. Add White Gaussian Noise Channel: - This block is intended to add noise to the communication channel as there is no noise free channel or communication.

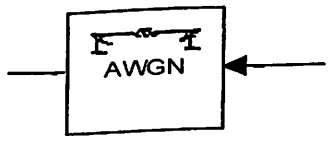


Figure 4.8: Add White Gaussian Noise

The AWGN Channel block adds white Gaussian noise to a real or complex input signal. When the input signal is real, this block adds real Gaussian noise and produces a real output signal. When the input signal is complex, this block adds complex Gaussian noise and produces a complex output signal. This block inherits its sample time from the input signal. This block uses the DSP System Toolbox Random Source block to generate the noise. Random numbers are generated using the Ziggurat method. The Initial seed parameter in this block initializes the noise generator. Initial seed can be either a scalar or a vector whose length matches the number of channels in the input signal.

vi. Free Space Losses: - This block is added to introduce the free space losses to the communication channel as per the fact that some amount of signal will be lost into the free space, not all of the transmitted signal will be captured by receiving antenna.

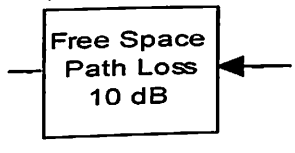


Figure 4.9: Free Space Path Losses

vii. Demodulator: - Digital signal is recovered back from the high frequency analogue signal such that other operations can be performed over it.

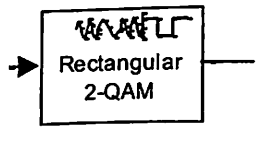


Figure 4.10: Rectangular QAM Demodulator Baseband

Here we are using 'Rectangular QAM Demodulator Baseband'. This block demodulates a signal that was modulated using quadrature amplitude modulation with a constellation on a rectangular lattice. The signal constellation has M points, where M is the M-ary number parameter. M must have the form 2^K for some positive integer K. The block scales the signal constellation based on how you set the Normalization method parameter. This block accepts a scalar or column vector input signal.

viii. Channel Decoder: - Redundant bits which were added to the information during channel encoding are removed here. It has a capability of detecting and correcting the errors by reading the redundant bits.

We have used three different decoding techniques each for the consecutive encoder. These are discussed as follows:

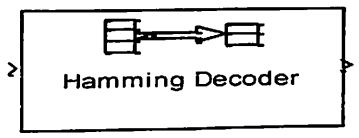


Figure 4.11: Hamming Decoder

a) Hamming Decoder:- This block decodes the hamming code word vector generated by the Hamming Encoder. For proper decoding, the parameters of decoding block should match as with that of encoding block. Message length of Hamming Code is K and codeword length N, where N is of the form 2^M-1, where M is greater than equal to3. Also K is equal to N-M.

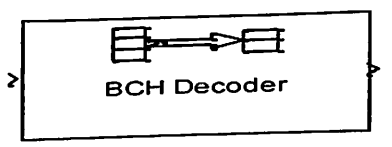


Figure 4.12: BCH Decoder

b) BCH Decoder: - This block decodes the BCH code word vector generated by the BCH Encoder. For proper decoding, the parameters of decoding block should match as with that of encoding block. This block accepts a column vector input signal with an integer multiple of N elements. Each group of N input elements represents one codeword to be decoded.

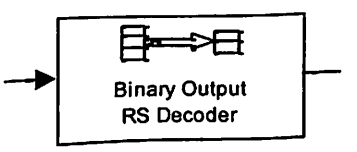


Figure 4.13: Binary Output RS Decoder

c) Binary Output RS Decoder: - This block decodes the Reed Solomon code word vector generated by the Reed Solomon Encoder. For proper decoding, the parameters of decoding block should match as with that of encoding block. The Reed-Solomon code

has message length, K, and codeword length, N. You specify both N and K directly in the dialog box. The symbols for the code are binary sequences of length M, corresponding to elements of the Galois field GF (2^M), where the first bit in each sequence is the most significant bit.

ix. Source Decoder: - We have used Differential Decoder as Source Decoder.

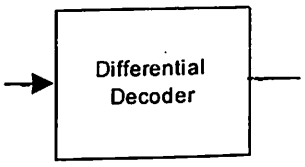


Figure 4.14: Differential Decoder

This block decodes the binary input signal within a channel and gives the output as the logical difference between the previous output element and the current input element.

x. Display: - This is an output section to examine the behavior of output. We have used three different types of displays with different objectives as follows:

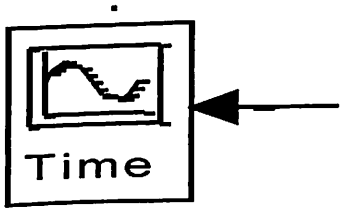


Figure 4.15: Vector Scope

Vector Scope: - The Vector Scope block is a comprehensive display tool similar to a digital oscilloscope. The block can display time-domain, frequency-domain, or user-defined signals. You can use the Vector Scope block to plot consecutive time samples from a vector, or to plot vectors containing data such as filter coefficients or spectral magnitudes.

We have used this to display the input information generated by Bernoulli Binary Generator.

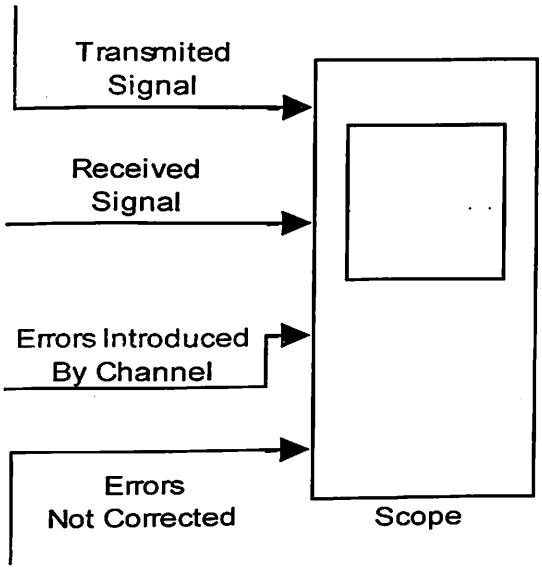


Figure 4.16: Scope

Scope: - This block allows you to see the input as per the simulation time. There is a provision to see multiple inputs at a same time and analyze them comparatively. You can also edit the time axis (x-axis) or the amplitude axis as per your feasibility.

This scope is used to do the comparative study of four different signals, those are, transmitted signal, received signal, errors introduced by channel and errors which are still not corrected by the decoder.

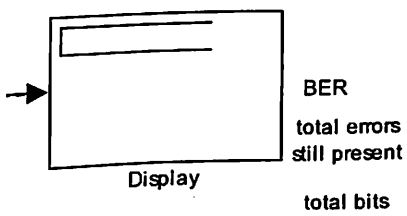


Figure 4.17: Display

Display: - This block display any set of parameters given to it in the following format of short, long, short_e, long_e, bank, hex (stored integer), binary (stored integer), decimal (stored integer), octal (stored integer).

This block is used to display the Bit Error Rate, Errors Not Corrected and Total Number of Bits Checked.

xi. Others: - There are few other block sets which are not mentioned above figure of 'Model Overview' but are still equally important. These are given as following :

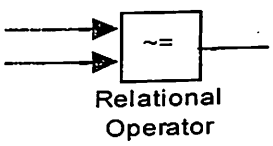


Figure 4.18: Relational Operator

Relational Operator: - This block compares the two inputs given to it using relational operator which you specify. Above port denotes first input while the lower port is the second input.

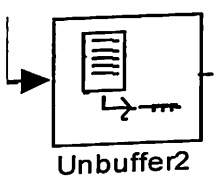


Figure 4.19: Unbuffer

Unbuffer: - The Unbuffer block unbuffers an M_i-by-N input into a 1-by-N output. That is, inputs are unbuffered row-wise so that each matrix row becomes an independent time-sample in the output. The rate at which the block receives inputs is generally less than the rate at which the block produces outputs.

4.2 Channel Noise Model

4.2.1 Selection of Blocks

To make a Channel Noise model first you need to drag all of its sub component blocks to a new model window such as follows:

a) Type 'commstartup' to change default Simulink settings for communication system models.
b) Now type 'simulink' to open Simulink Library Browser.
c) From the 'File' menu in the window click 'new' and then 'model'. This opens a new model window.
d) Drag the following blocks from the Simulink Library Browser to the new model window:
   - Bernoulli Binary Block Generator, from the Random Sources Sub-library of Comm Sources library.
   - Differential Encoder and Differential Decoder, from the Source Coding library.
   - Rectangular QAM Baseband Modulator and Rectangular QAM Baseband Demodulator, from the Digital Baseband Modulation sub-library of the Modulation library.
   - Free Space Path Loss, from the RF Impairments library.
   - AWGN Channel, from Channels library.
   - Error Rate Calculation, from Comm Sinks library.
   - Display and Scope, from the Sink library of Simulink blockset.
   - Vector Scope, from the Signal Processing Sink library of DSP System Toolbox.
   - Relational Operator, from Logic and Bit Operations library of Simulink blockset.
   - Unbuffer, from Buffers sub-library of Signal Management from DSP System Toolbox.

4.2.2 Setting Parameters

a) Double click on Bernoulli Binary Block Generator and enable Frame Based Output, set Samples per frame as 4.

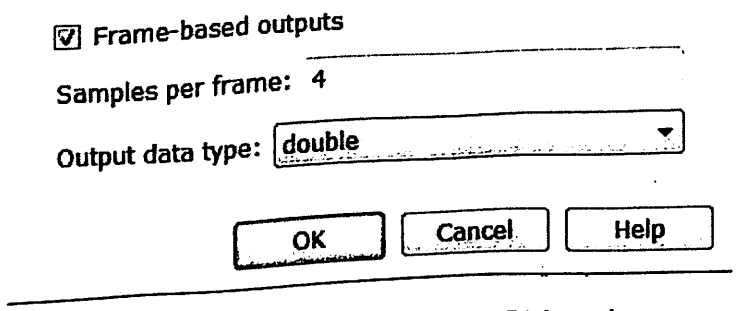


Figure 4.20: Bernoulli Binary Block Generator Dialogue box

b) Double click on Free Space Path loss block, set the loss as 200.

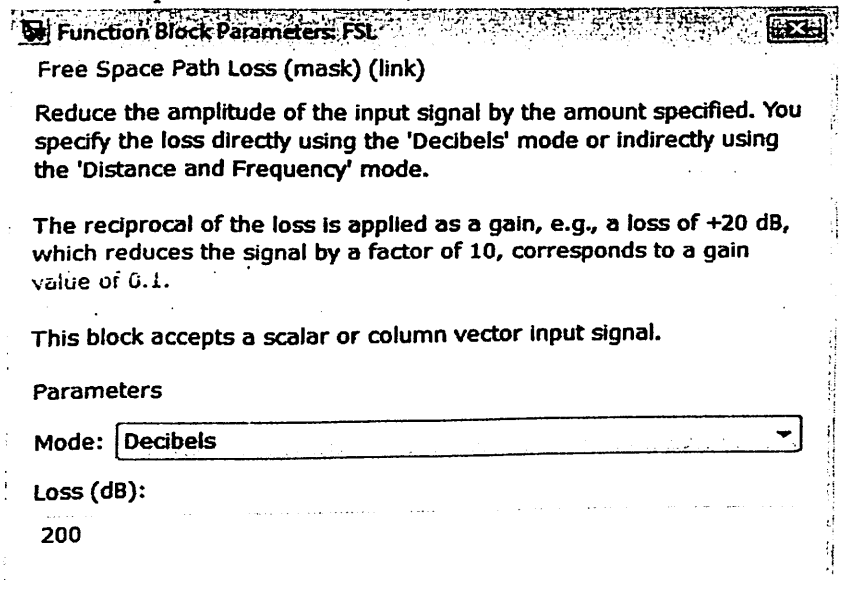


Figure 4.21: Free Space Path Loss Dialogue box

c) Double click on AWGN, set initial seed as 67 and Eb/N_o as 5.

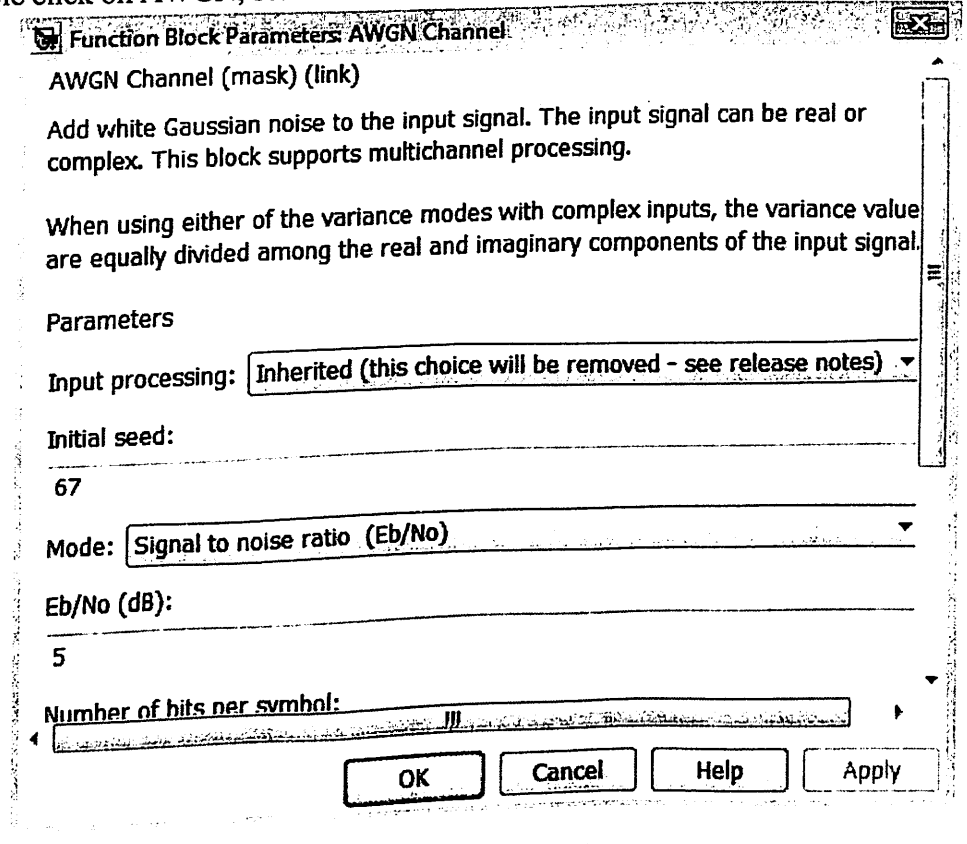


Figure 4.22: AWGN Dialogue box

d) Double click on Rectangular QAM Modulator Baseband and set M – ary number as 2.

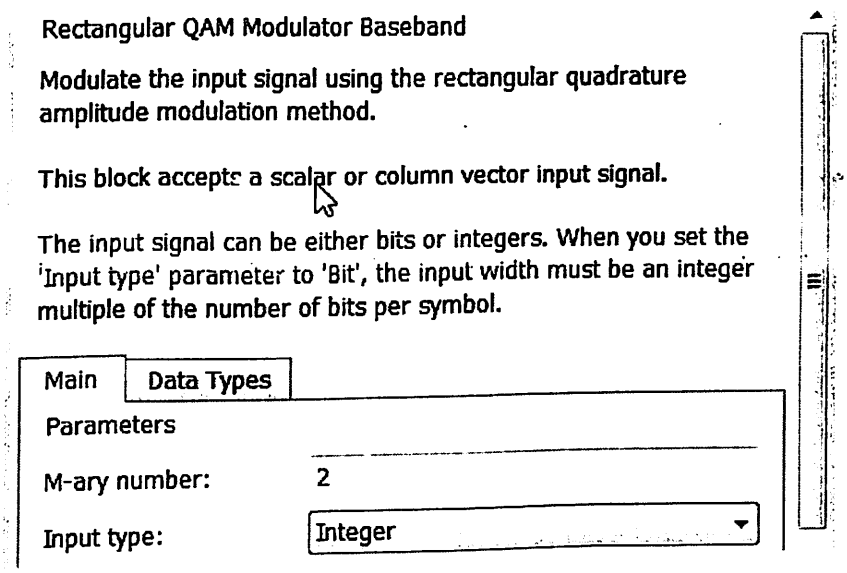


Figure 4.23: Modulator Dialogue box

e) Double click on Rectangular QAM Demodulator Baseband, set M – ary number as 2, output type as Bit and Decision type as Hard decision.

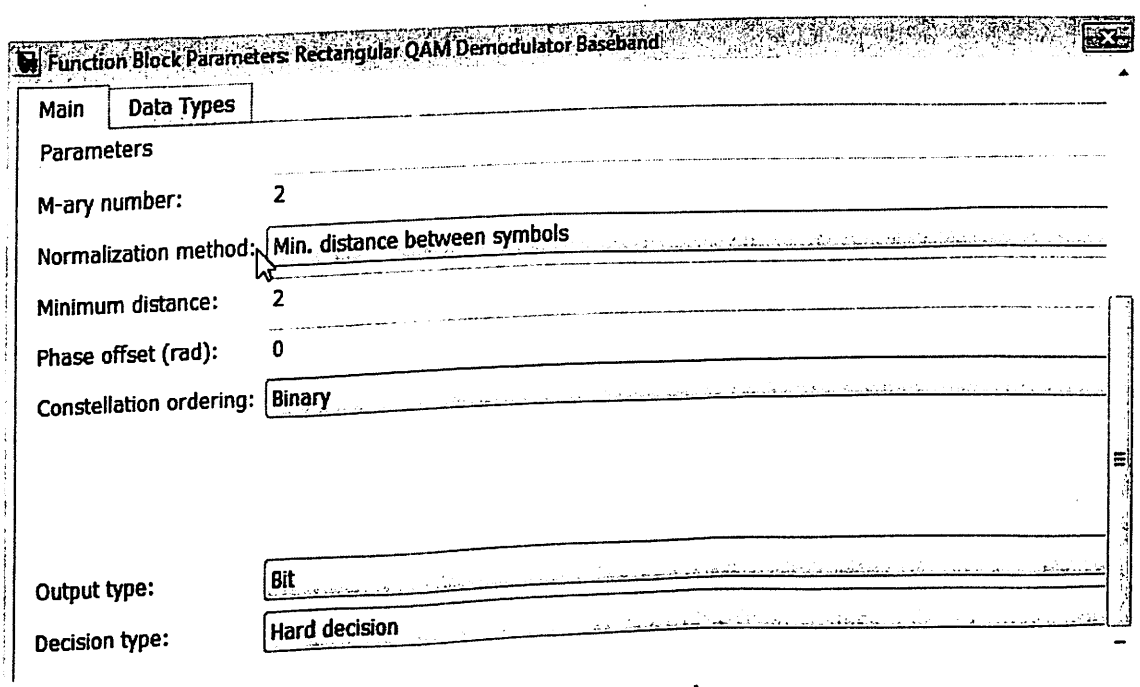


Figure 4.24: Demodulator Dialogue box

f) Double click on Vector scope, change Input domain as Time, Time display span should be changed to 30. Under the column of Axis Properties Y-max should be set as 1.5 while Y-min as – 0.5.

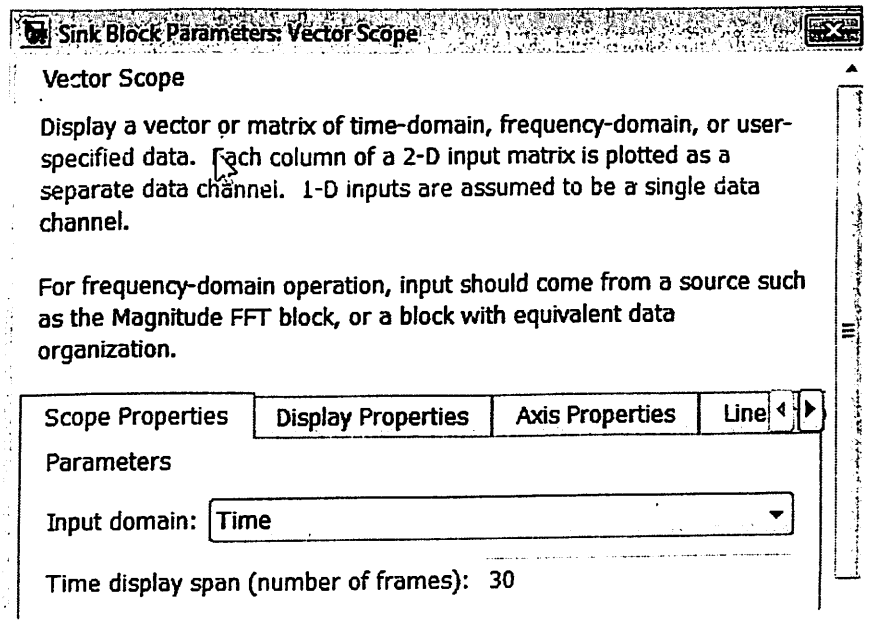


Figure 4.25: Vector Scope Dialogue box

g) Double click on Scope, set Number of axes as 4and Time range as1000.

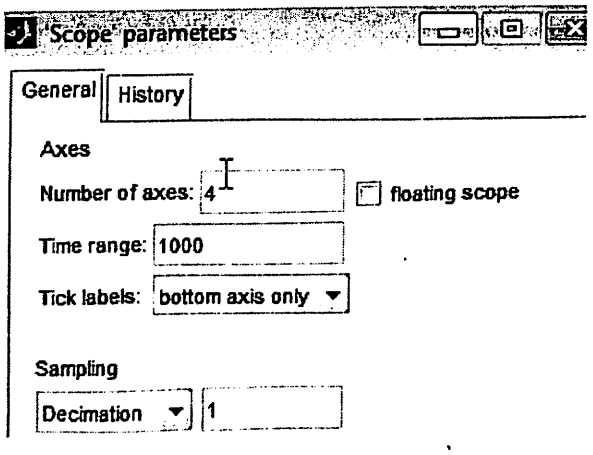


Figure 4.26: Scope Dialogue box

4.2.3 Connecting the Blocks

Next step is to connect the blocks logically so as to give meaning to the system. Connect the blocks as shown in the **FIGURE**. The following points should be taken care of while connecting the blocks, those are:

- Keep the blocks enough distant apart so as to avoid overlapping and confusion.
- Label the blocks so as to get a better understanding.
- Enable 'signal dimension' to get a clear view of the number of Bits at a given period of time at any section of the system. It is done by the following steps:
  Right click >> More >> Format >> Signal dimensions.
- Make sure to connect the output of the Differential Decoder to the receiver of the Error Rate Calculation block.
- Utmost care is to be given while making a branch line (as in the given figure from Bernoulli Binary Generator Block to Error Rate Calculation Block).
- Enlarge the Display block so as to view all three parameters given to it.

4.2.4 Running the Model

To run a Simulink design model enter the stop time as 10000 sec in the space provided in the toolbar. Click on start button, simulation will stop automatically after 10000 sec. There is a provision to pause or stop the program before the completion of the given time.

Now to save the model, select 'save' from the 'file menu' with an appropriate file name.

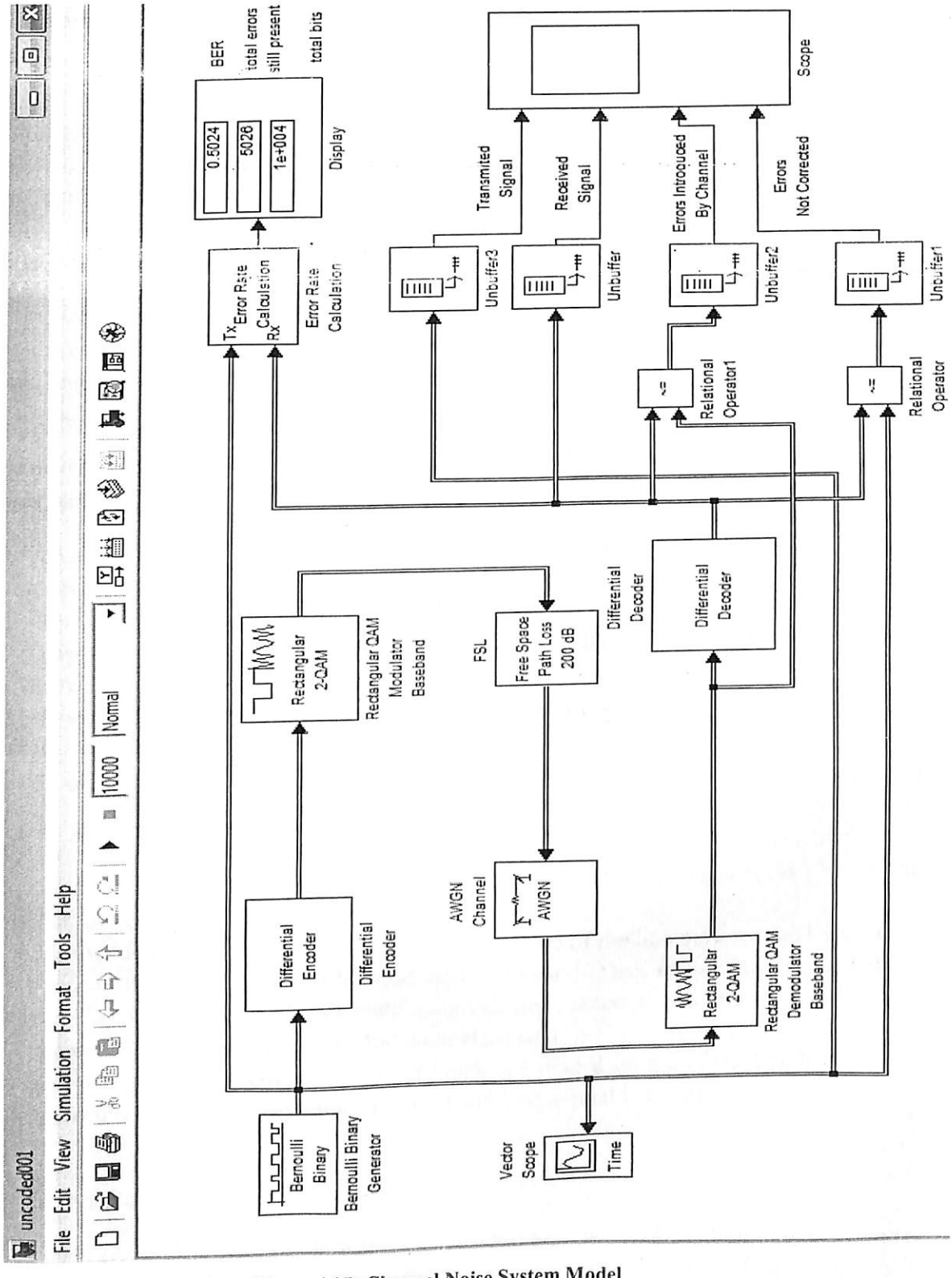


Figure 4.27: Channel Noise System Model

4.3 Hamming Code Model

4.3.1 Building of Hamming Code Model

Above model was a channel noise model in which channel noise is also present with the information at the receiver. As it is an uncoded model so the errors which were generated during the transmission of the signal cannot be detected at receiver end, and hence cannot be corrected.

This model explains how a coded system can correct the errors. Here hamming code technique is used to encode the signal. For this purpose a Hamming Encoder block needs to be added in the above model just after the Differential Encoder, such that the binary signals coming out of the source encoder can be channel encoded before sending it for modulation. Similarly another block of Hamming Decoder will be added just after the Rectangular QAM Demodulator Baseband block to decode the Hamming signal and get back the original signal.

To update the above Channel Noise Model to the Hamming Code Model, use the following steps enlisted below:

a) From the 'File' menu in the window's toolbar, click 'new' and then 'model'. This opens a new model window.
b) Copy Channel Noise Model and paste it in this new window.
c) Drag the following two Communication Blocksets blocks from the Simulink Library Browser to the new model window:
    - Hamming Encoder block, from the Block sub-library of the Error Detection and Correction Library.
    - Hamming Decoder block, from the Block sub-library of the Error Detection and Correction library.
d) Click the right border of the model and drag it to right to widen the model window.
e) Similarly move Rectangular QAM Modulator Baseband to the right hand side by the click and drag method.
f) Click the Hamming Encoder block and drag it on top of the line between the Differential Encoder and Rectangular QAM Modulator Baseband. Then release the mouse button. The Hamming Encoder block should automatically connect to the line from Differential Encoder and Rectangular QAM Modulator Baseband.
g) Similarly, click the Hamming Decoder block and drag it on top of the line between the Rectangular QAM Demodulator Baseband and Differential Encoder.

4.3.2 Setting Parameters

We have used [7, 4] Hamming Code, which encodes message word of length 4 into a Hamming code of length 7. As a result, the block converts frame of size 4 to the frame of size 7. The code

can correct one error in each transmitted codeword, if more than one error occurs, Hamming Decoder might decode incorrectly.

Therefore for an [n, k] code, n = 7 and k = 4.

As we had already enabled the Frame-based output and had also set Samples per frame as 4 in the Bernoulli Binary Block Generator, so no need to modify it.

Double click on Hamming Encoder and set the Codeword length N as 7. Do the same thing for Hamming Decoder block.

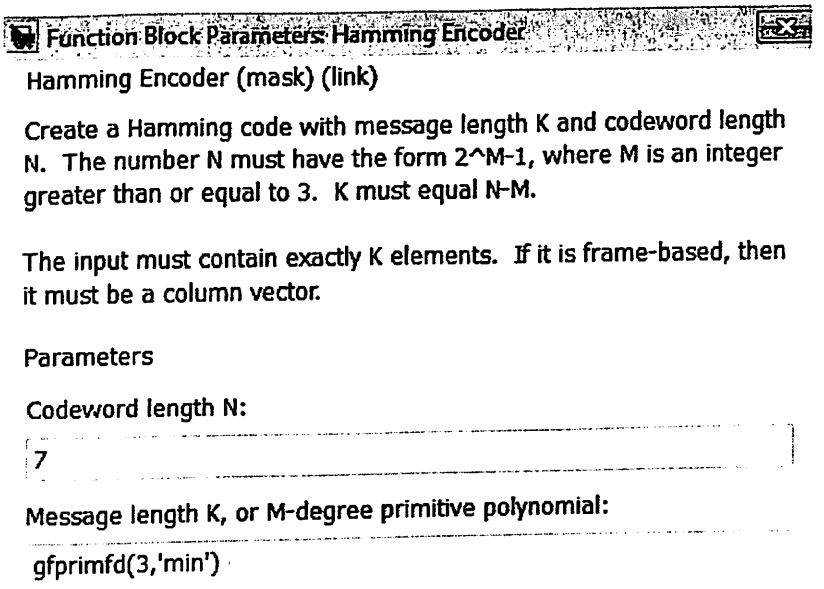


Figure 4.28: Hamming Encoder Dialogue Box

Run the model as explained in Section 4.2.4 and compare the output of the Hamming Code Model with that of Channel Noise Model, observe the difference in number of uncorrected errors.

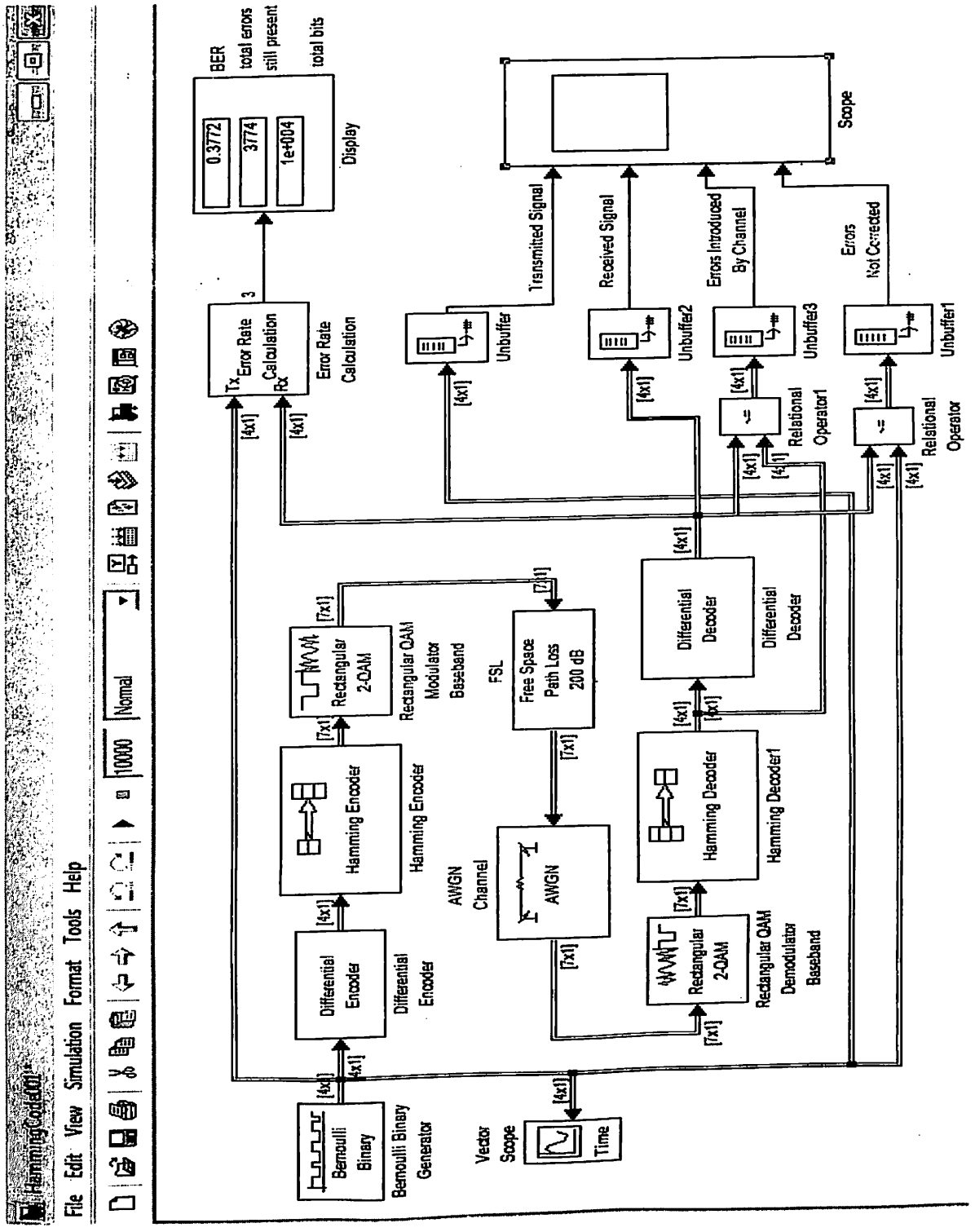


Figure 4.29: Hamming Code Model

4.4 BCH Code Model

4.4.1 Building of BCH Model

Under this section we will be examining the number of errors corrected using BCH Encoding technique. For this purpose we will again be taking the above described Channel Noise Model and modify it, so as to correct the number of errors generated by channel noise. Add BCH Encoder block to this model just after the Differential Encoder, such that the binary signals coming out of the source encoder can be channel encoded before sending it for modulation. Similarly another block of BCH Decoder will be added just after the Rectangular QAM Demodulator Baseband block to decode the BCH signal and get back the original signal.

To update the above Channel Noise Model to the BCH Code Model, use the following steps enlisted below:

a) From the 'File' menu in the window's toolbar, click 'new' and then 'model'. This opens a new model window.
b) Copy Channel Noise Model and paste it in this new window.
c) Drag the following two Communication Blocksets blocks from the Simulink Library Browser to the new model window:
    - BCH Encoder block, from the Block sub-library of the Error Detection and Correction Library.
    - BCH Decoder block, from the Block sub-library of the Error Detection and Correction library.
d) Click the right border of the model and drag it to right to widen the model window.
e) Similarly move Rectangular QAM Modulator Baseband to the right hand side by the click and drag method.
f) Click the BCH Encoder block and drag it on top of the line between the Differential Encoder and Rectangular QAM Modulator Baseband. Then release the mouse button. The BCH Encoder block should automatically connect to the line from Differential Encoder and Rectangular QAM Modulator Baseband.
g) Similarly, click the BCH Decoder block and drag it on top of the line between the Rectangular QAM Demodulator Baseband and Differential Encoder.

4.4.2 Setting Parameters

We have used [15, 11] BCH Code, which encodes message word of length 11 into a BCH code of length 15. As a result, the block converts frame of size 11 to the frame of size 15. The code can correct one error in each transmitted codeword, if more than one error occurs, BCH Decoder might decode incorrectly.

Therefore for an [n, k] code, n = 15 and k = 11.

As the size of the frame to the input of BCH Encoder is 11 so we need to set Samples per frame as 11 in Bernoulli Binary Generator.

Double click on BCH Encoder and set the Codeword length N as 15 and Message length, k as 11. Do the same thing for BCH Decoder block.

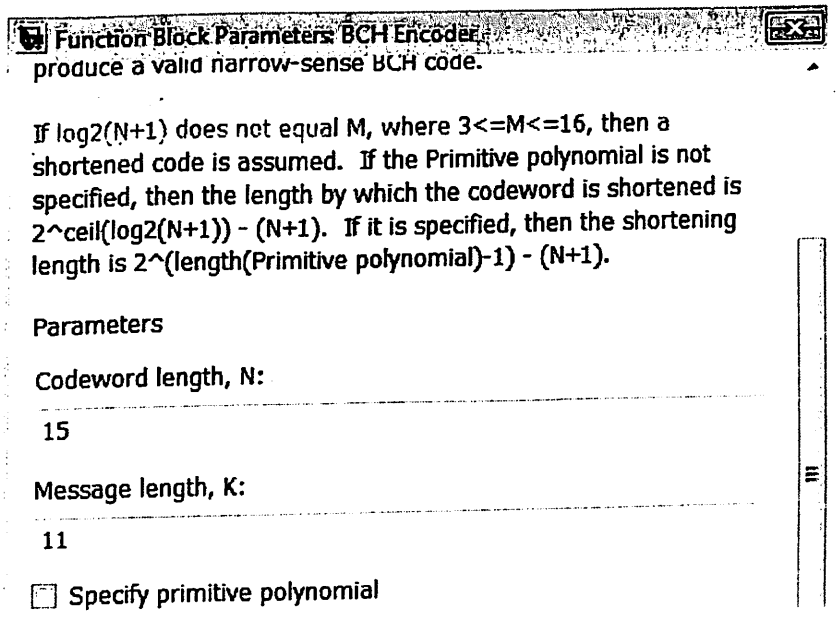


Figure 4.30: BCH Decoder Dialogue Box

Run the model as explained in Section 4.2.4 and compare the output of the BCH Code model with that of Hamming Code model and Channel Noise model, observe the difference in number of uncorrected errors.

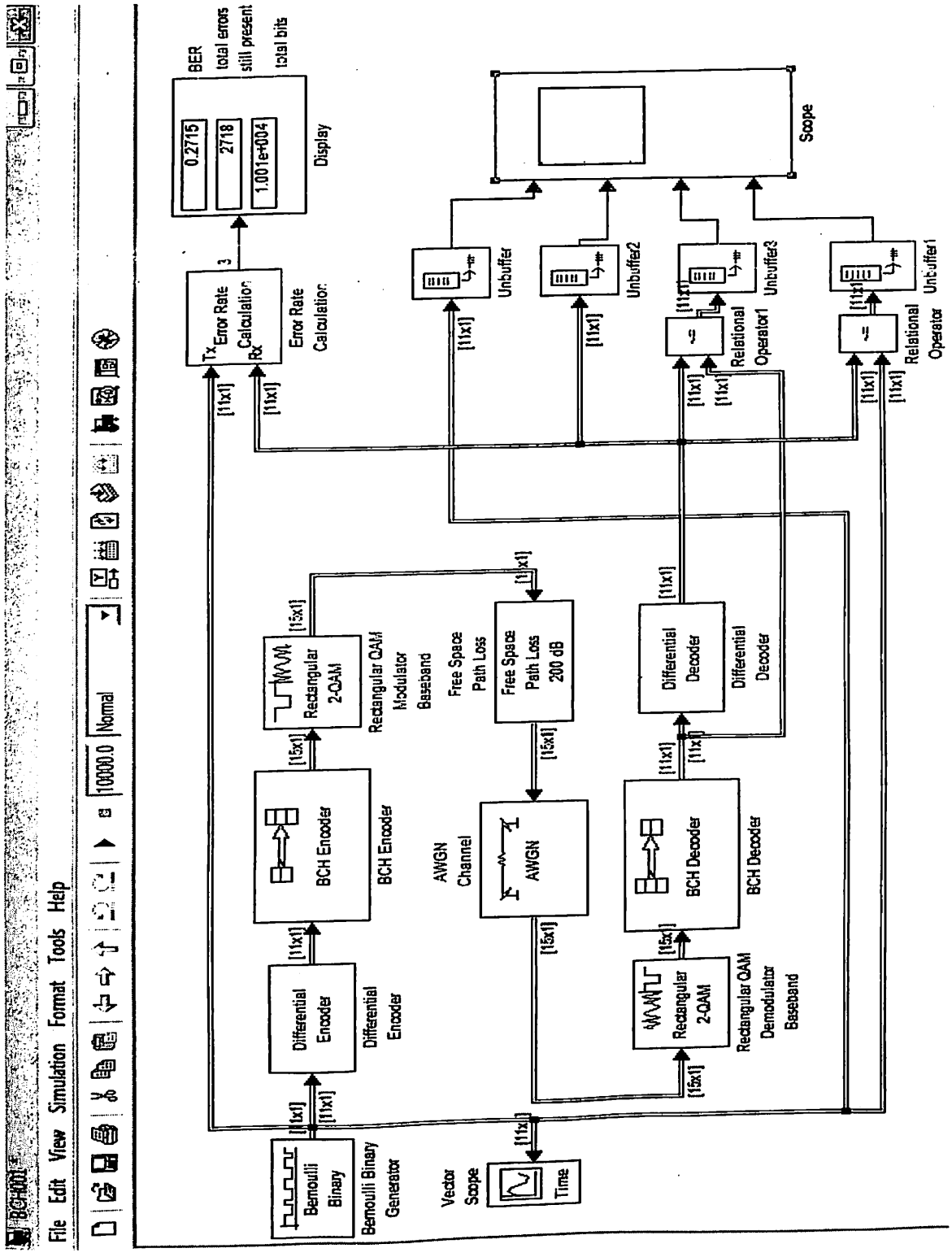


Figure 4.31: BCH Code Model

4.5 Reed Solomon Code Model

4.5.1 Building of RS Model

This section enlightens the process of removing errors incorporated by the channel to the message signal. For this purpose we will be considering Reed Solomon (RS) Encoder. As we already have a Noise Channel (discussed earlier) whose capacity of noise is already known to us, update this model by adding RS Encoder block just after the Differential Encoder, such that the binary signals coming out of the source encoder can be channel encoded before sending it for modulation. Similarly another block of RS Decoder will be added just after the Rectangular QAM Demodulator Baseband block to decode the RS signal and get back the original signal.

Up-gradation of Channel Noise Model to RS Model can easily be done by using the following steps:

a) From the 'File' menu in the window's toolbar, click 'new' and then 'model'. This opens a new model window.
b) Copy Channel Noise Model and paste it in this new window.
c) Drag the following two Communication Blocksets blocks from the Simulink Library Browser to the new model window:
    - RS Encoder block, from the Block sub-library of the Error Detection and Correction Library.
    - RS Decoder block, from the Block sub-library of the Error Detection and Correction library.
d) Click the right border of the model and drag it to right to widen the model window.
e) Similarly move Rectangular QAM Modulator Baseband to the right hand side by the click and drag method.
f) Click the RS Encoder block and drag it on top of the line between the Differential Encoder and Rectangular QAM Modulator Baseband. Then release the mouse button. The RS Encoder block should automatically connect to the line from Differential Encoder and Rectangular QAM Modulator Baseband.

Similarly, click the RS Decoder block and drag it on top of the line between the Rectangular QAM Demodulator Baseband and Differential Encoder.

4.5.2 Setting Parameters

We have used [7, 3] RS Code, which encodes message word of length 3 into a RS code of length 7. As a result, the block converts frame of size 3 to the frame of size 7. The code can correct one error in each transmitted codeword, if more than one error occurs, RS Decoder might decode incorrectly.

Therefore for an [n, k] code, n = 7 and k = 3.

RS Encoder block supports m-bit symbols instead of bits. Each entry must be an integer between 0 and 2^m-1. As we are using Binary Input RS Encoder so the input that we give to this block is in bits form but there is a provision within a block to convert these bits to integer, do the processing and then again convert it back to bits, therefore the output that we get is in integer form. Due to this reason the Samples per frame in the Bernoulli Binary Generator is set to 9.

Double click on RS Encoder and set the Codeword length N as 7 and Message length, k as 3. Do the same for RS Decoder block.

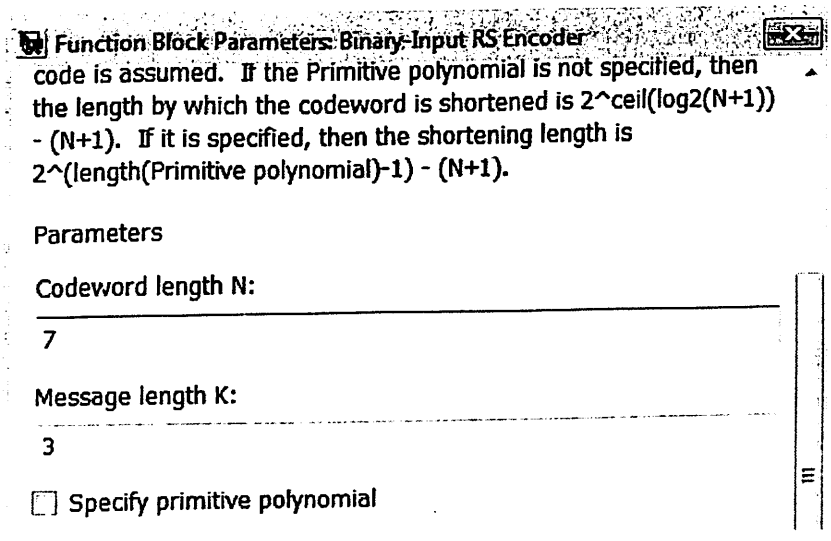


Figure 4.32: RS Encoder Dialogue Block

Run the model as explained in Section 4.2.4. We can easily note the number of errors corrected as the amount of channel noise in the Channel Noise Model was known to us before hand. We can also compare the output of RS Code with the output of BCH Code model and with that of Hamming Code model and can easily make a decision of which technique is better amongst the three.

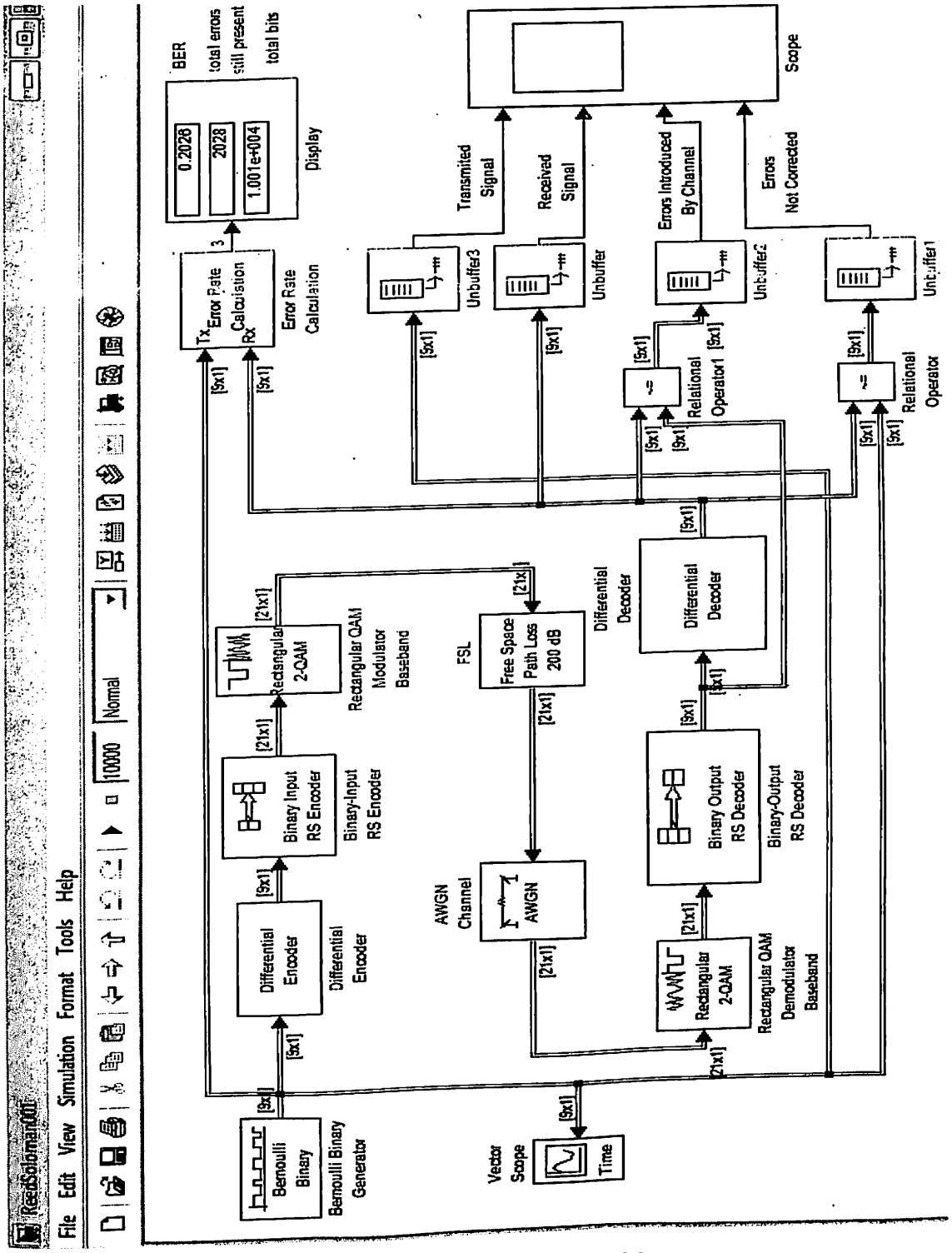


Figure 4.33: Reed Solomon Code Model

CHAPTER - 5
RESULTS

We have tried to keep all the affecting factors like input or E_b/N_o constant while dealing with all the above mentioned four cases so as to get a clear idea of which coding technique is best amongst the three.

Shown below is the input signal/ message signal (i.e. output from Bernoulli Binary Bit Generator) which is same for all the four cases.

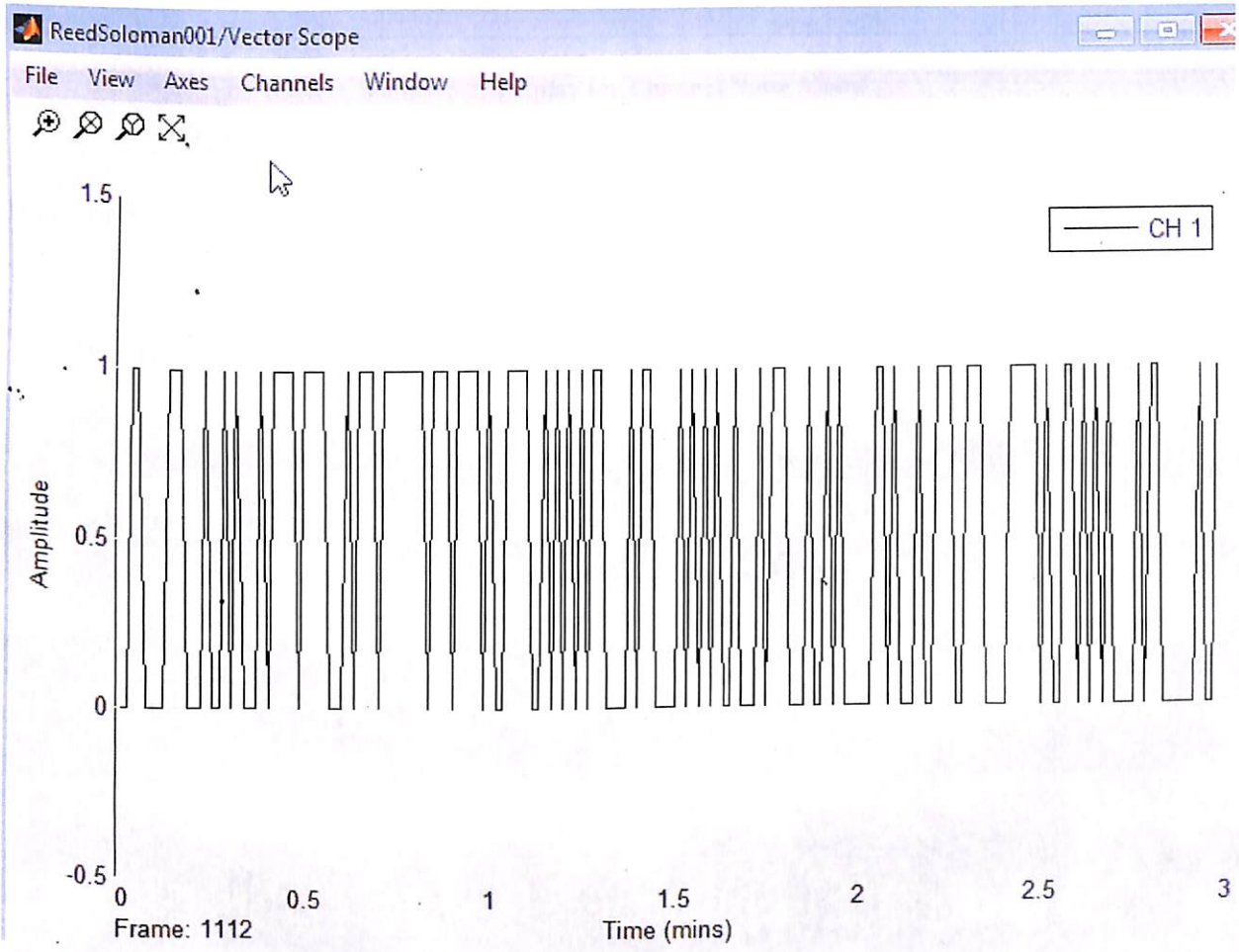


Figure 5.1: Running Output of Vector Scope

Next, we will be looking at the outputs of the various design models. We have two types of output, one is of Display which gives numerical data about BER, Total Errors Still Present and Total Bits checked. Other output is of Scope which is a graphical output of four different parameters, those are, Transmitted Signal, Received Signal, Errors Introduced by Channel and Errors Not Corrected.

5.1 Results of Channel Noise Model

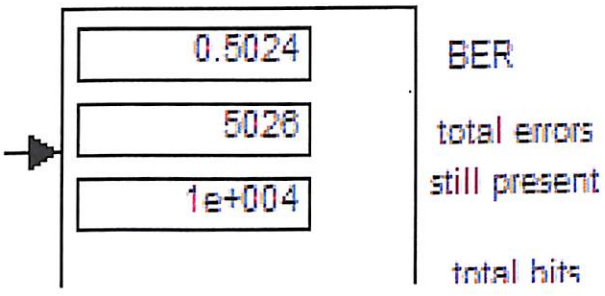


Figure 5.2: Display for Channel Noise Model

Bit Error Rate	=	0.5024
Total Errors Still Present	=	5026
Total Bits Tested	=	10000

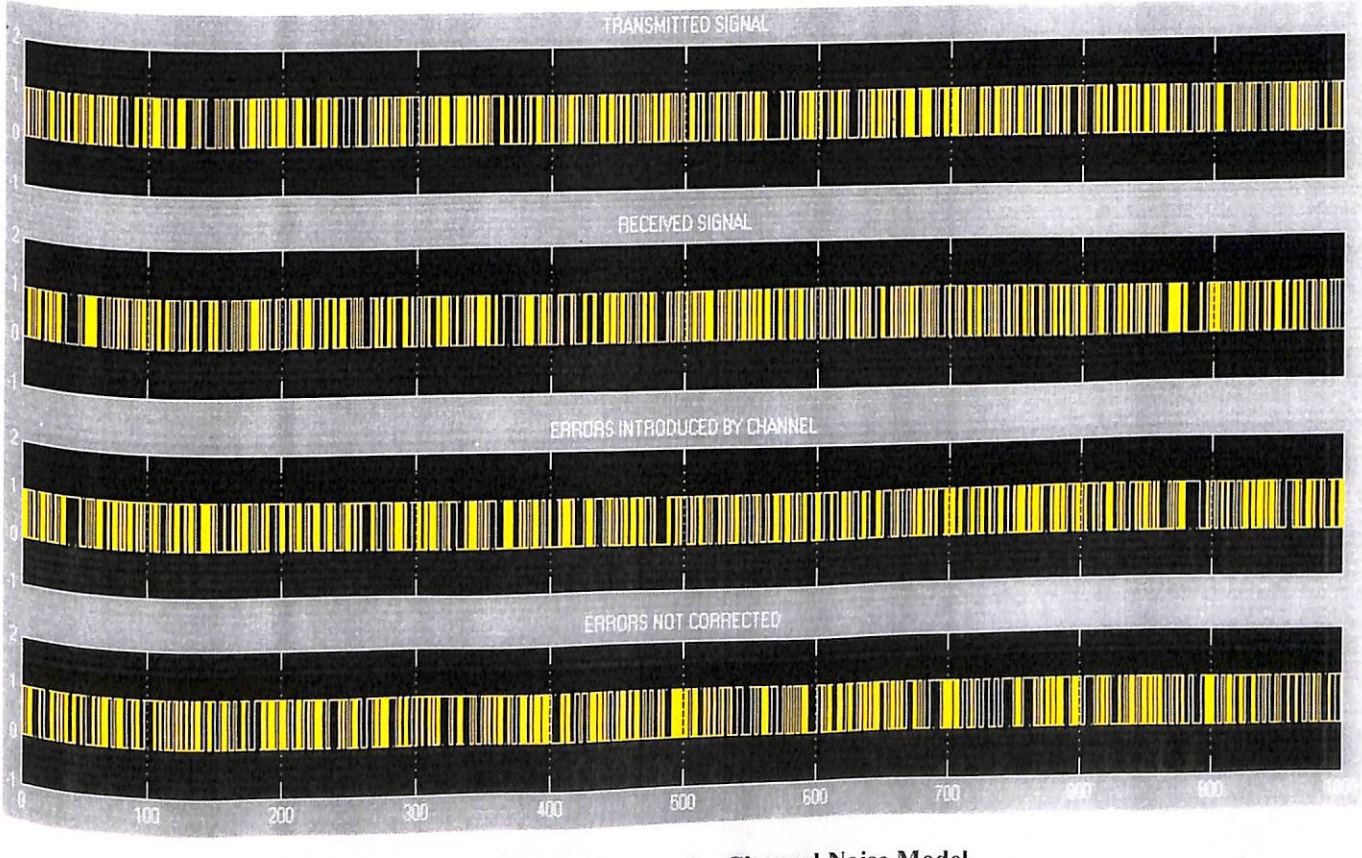


Figure 5.3: Scope Output for Channel Noise Model

5.2 Results of Hamming Code Model

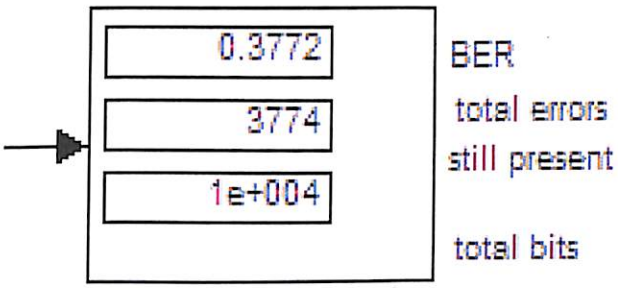


Figure 5.4: Display for Hamming Code Model

Bit Error Rate	=	0.3772
Total Errors Still Present	=	3774
Total Bits Tested	=	10000

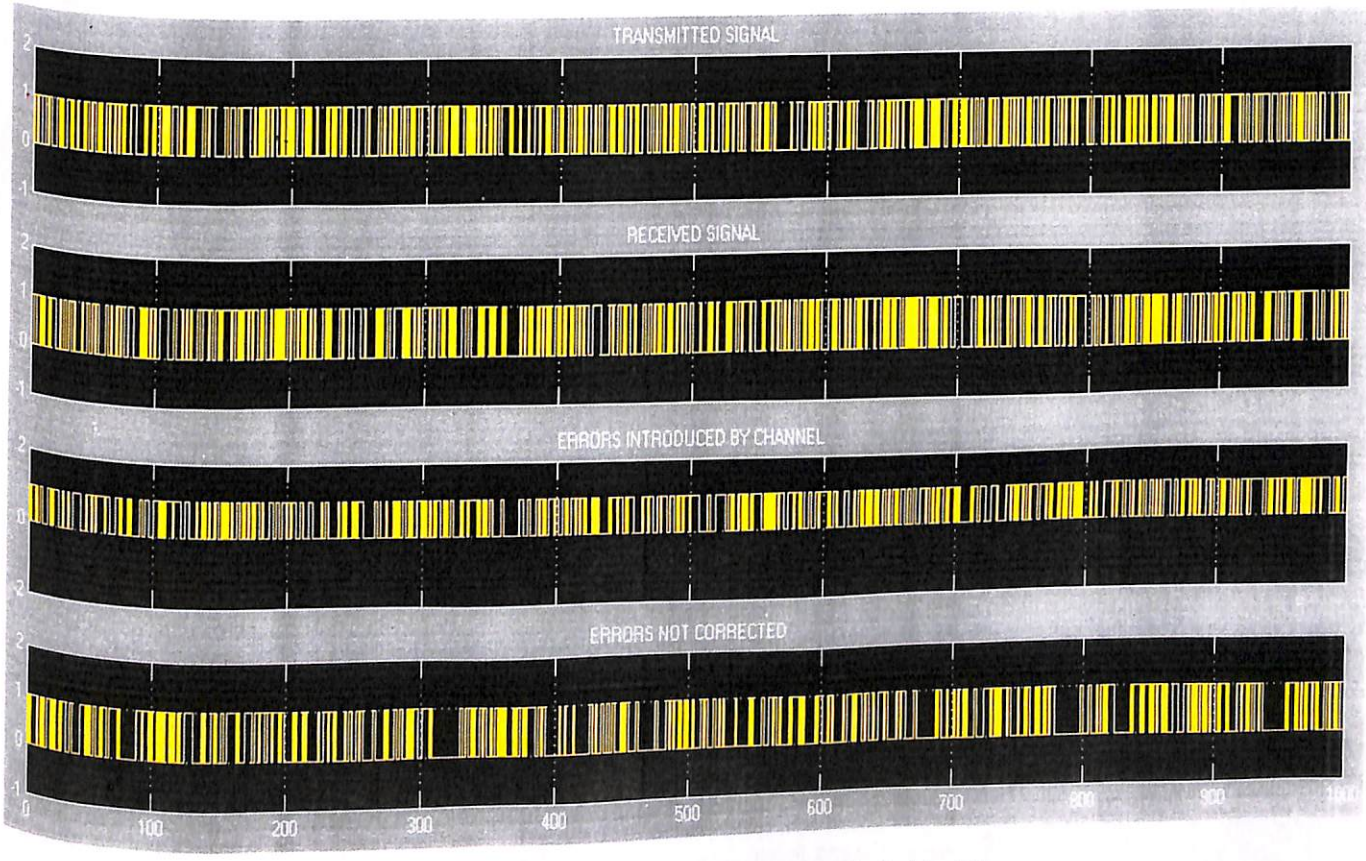


Figure 5.5: Scope Output for Hamming Code Model

5.3 Results of BCH Code Model

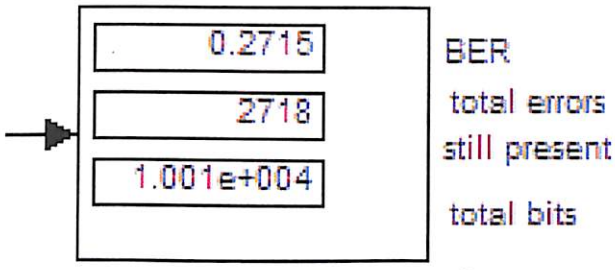


Figure 5.6: Display for BCH Code Model

Bits Error Rate	=	0.2715
Total Errors Still Present	=	2718
Total Bits Tested	=	10000

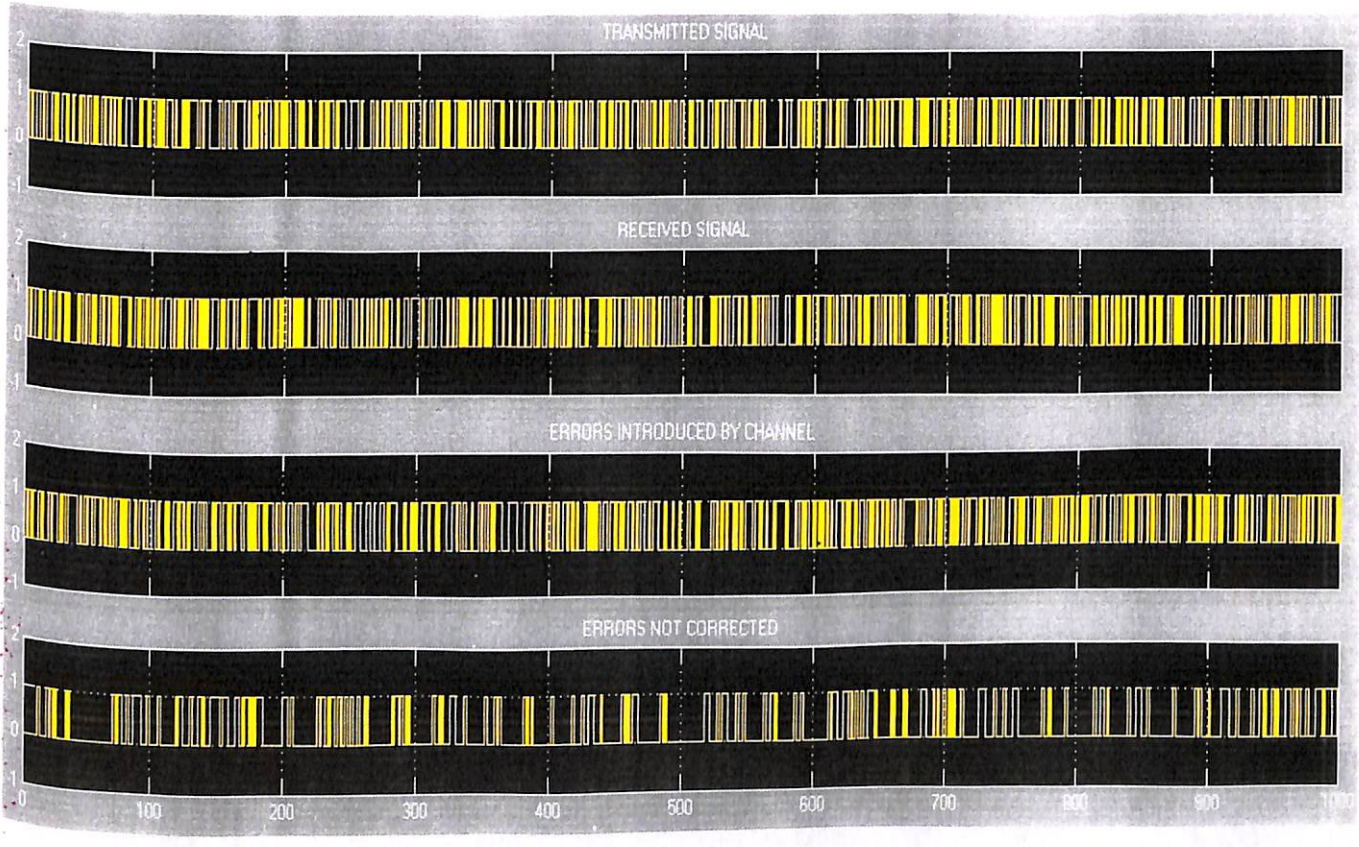


Figure 5.7: Scope Output for BCH Code Model

5.4 Results of Reed Solomon Code Model

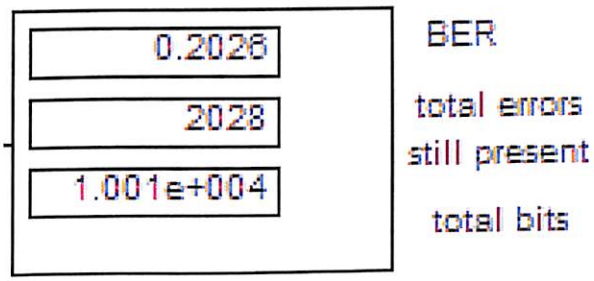


Figure 5.8: Display for RS Code Model

Bit Error Rate	=	0.2026
Total Errors Still Present	=	2028
Total Bits Tested	=	10000

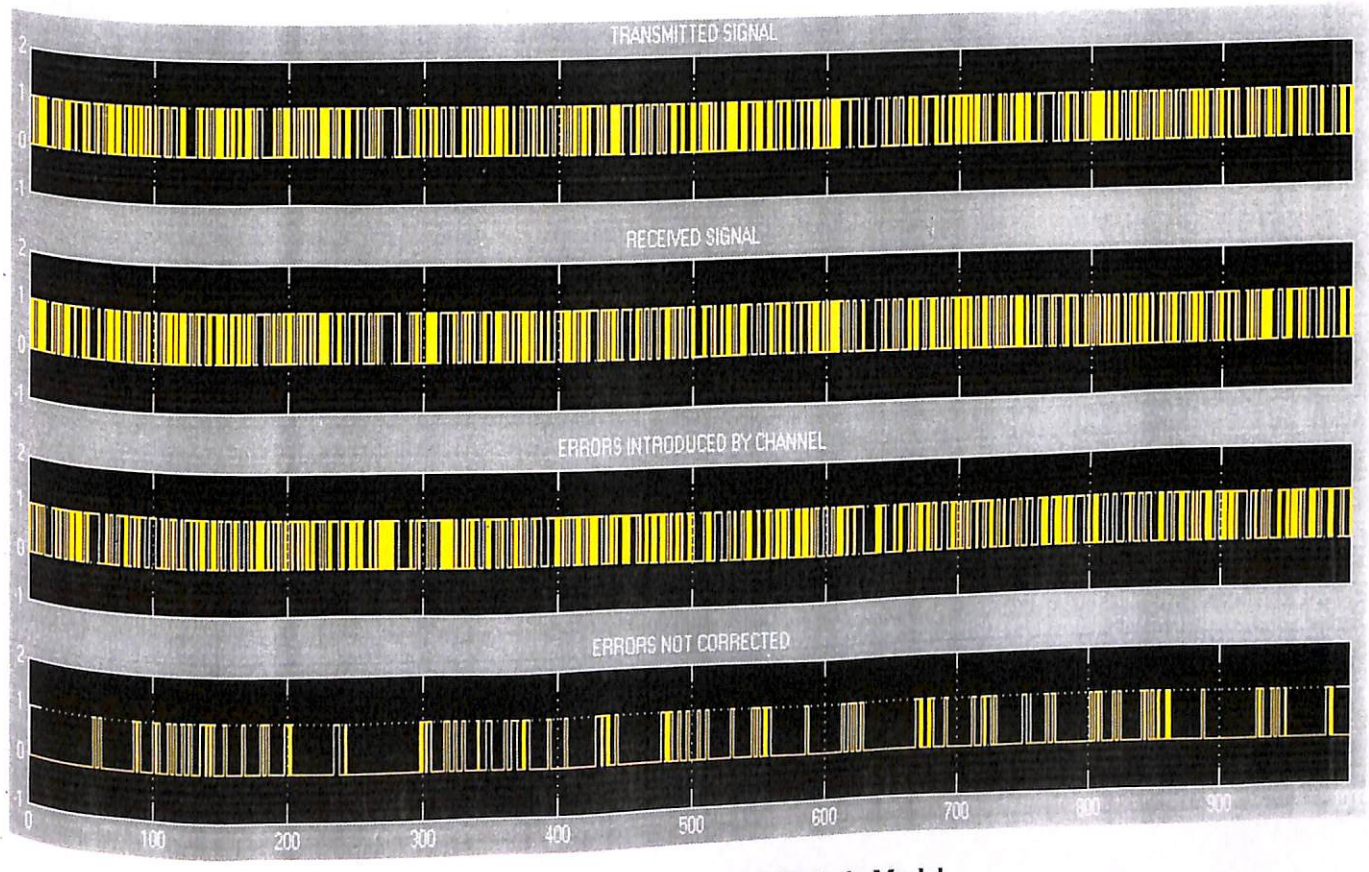


Figure 5.9: Scope Output for RS Code Model

CHAPTER - 6
CONCLUSION

Today, we are in the era of digital communication, from computer applications to mobile phone networks. Video Conferencing, e-commerce, advertisements are few other very important aspect covered under digital communication. Hence error free communication over a noisy channel is the need of the hour.

As examined earlier Forward Error Correction (FEC) technique allows the detection and correction of errors present in the message signal at the receiver end without retransmission of message over and over again. Even though bit-rate has to be compromised up to an extent but the final signal that we receive is error free or has minimal error.

The above result compares the error correcting efficiency of Hamming Code, BCH Code and Reed Solomon Code; shown below are the errors present in the message signal after their respective decoding.

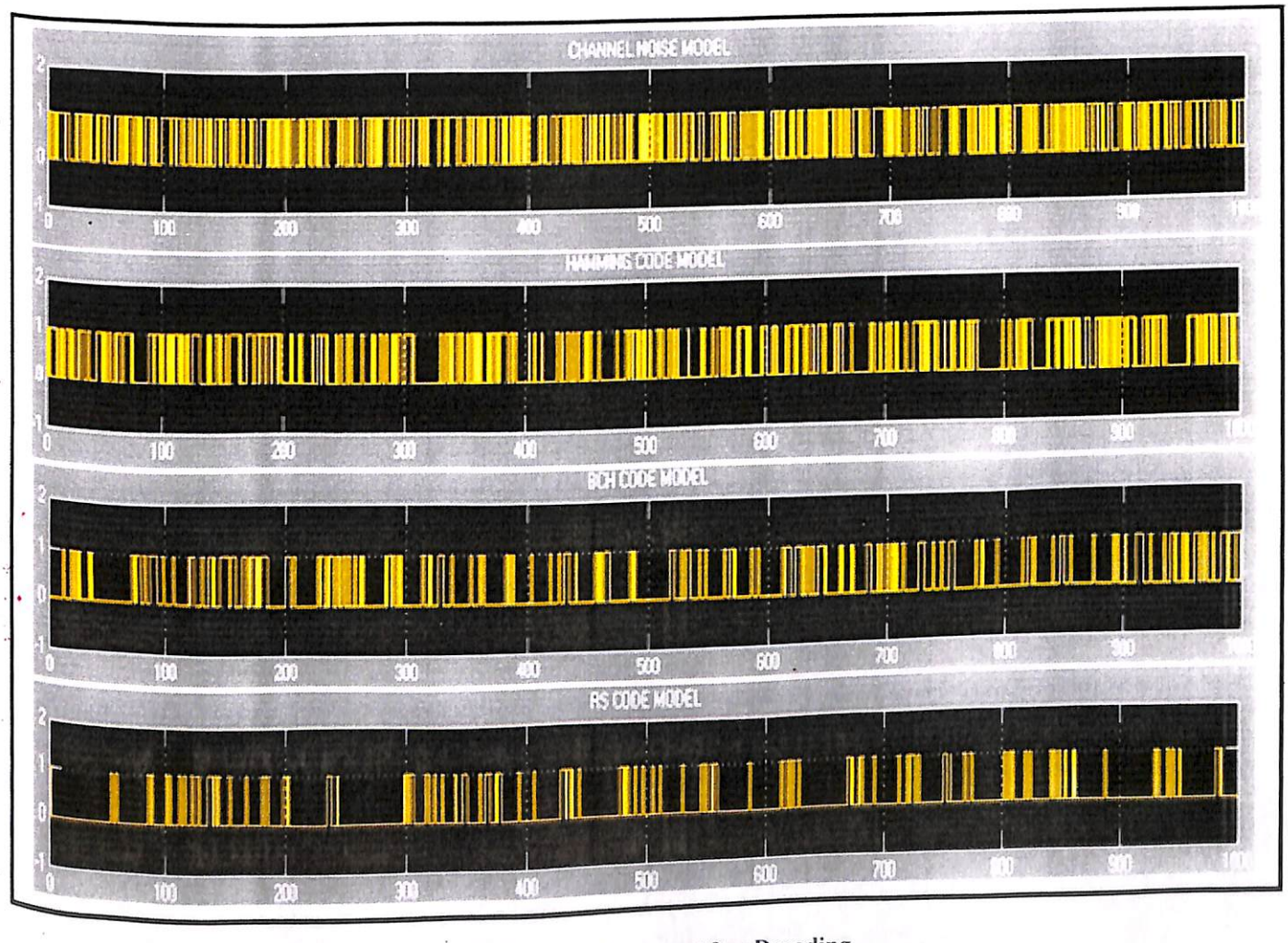


Figure 6.1: Errors Present After Decoding

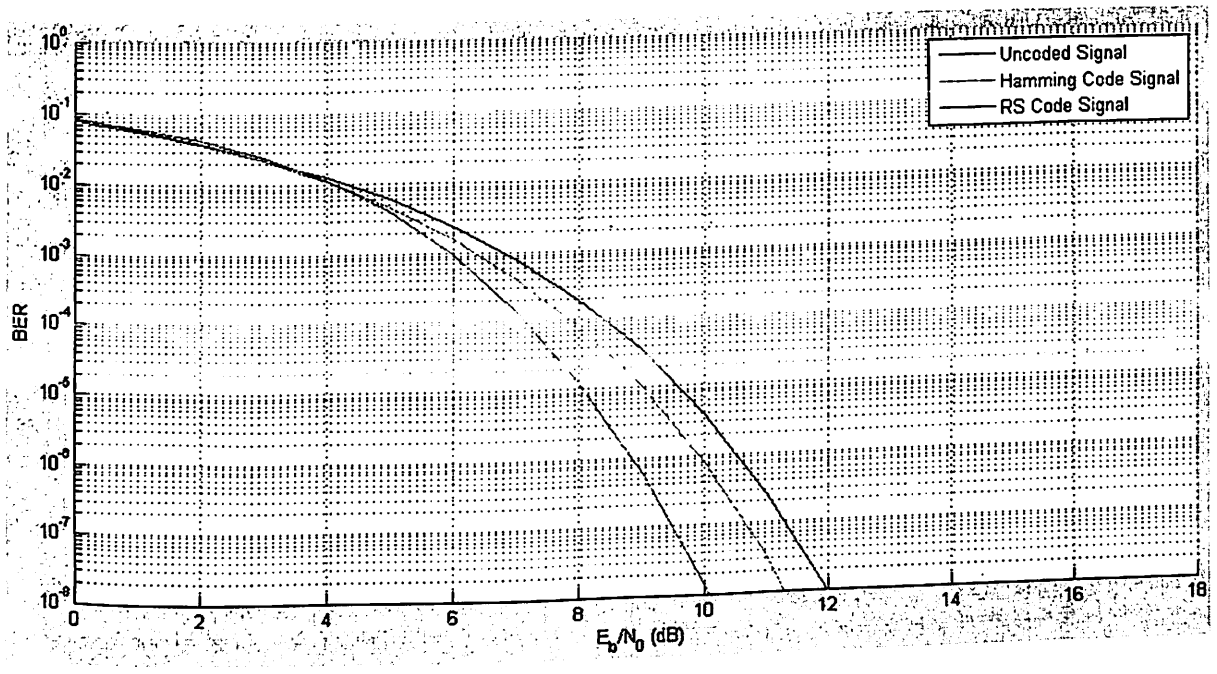


Figure 6.2: BER V/s Eb /No Plot

As seen from the above observations Reed Solomon Code has the best error detection and correction capability amongst the three, followed by BCH Code and Hamming Code.

7. REFERENCES

1. **Hamood, S, and I. Widad** (2010) The development and implementation of reed Solomon codes for OFDM using software – defined radio platform, *International Journal of Computer Science and Communication,* Vol. 1, No. 1, 129–136.
2. **Hamming, R.W.** (1950) Error Detecting and Error correcting Codes, *The Bell SystemTechnical Journal,J Soc, Indust. Appl. Math,* Vol. 26, No.2, 124-140.
3. **Reed, I.S. and G. Solomon** (1960) *Polynomial Codes Over Certain Finite Fields, The Communication Journal J Soc, Indust. Appl. Math.* Vol. 8, No. 2,330-347.
4. **Mishra, B.K., S. Kaulgud, and S. Save** (2012) Design of RS code using Simulink platform, *International Conference on Recent Trends in Technology (TCET),* India, June, 02-08.
5. **Wicker, S.B. and V.K. Bhargava** *Reed–Solomon Codes and Their Applications,* IEEE Press, New Jersey, 1994.
6. **S. Peter** *Error Control Coding: An Introduction.* Prentice Hall, Englewood Cliffs, New Jersey, 1991.
7. **Sklar, B.** *Digital Communications: Fundamentals and Applications,* Prentice Hall, Englewood Cliffs, New Jersey, 1988.
8. **Lin, S. and D.J. Costello** *Error Control Coding: Fundamentals and Applications,* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
9. **Sklar, B.** *Digital Communications Fundamentals and Applications, 2nd Ed.* Prentice Hall, Englewood Cliffs, New Jersey, 2001.
10. **Lathi, B.P, and D. Zhi** *Modern Digital and Analog Communication System: Error Correction Codes.* Oxford University Press, New York, 2010.