(7,3) MAXIMUM-LENGTH BINARY CYCLIC CODE APPLIED TO SINGLE CHANNEL DIGITAL COMMUNICATION SYSTEM FOR ERROR CORRECTION

by

YOHANNES NEGASH

A thesis submitted in partial fulfilment

for the degree of M.Sc. in Electrical Engineering

in the Addis Ababa University

June 1996

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

(7,3) MAXIMUM - LENGTH BINARY CYCLE CODE APPLIED TO SINGLE CHANNEL DIGITAL COMMUNICATION SYSTEM FOR ERROR CORRECTION

BY
YOHANNES NEGASH

Approval of Board of Examiners:

Dr. Girma Mullisa
Chairman, Department Graduate Committee

Dr.-Ing. Ketema Alemu
Advisor

Dr.-Ing. Kissig
Examiner

Dr. P. Haferkorn
Examiner

TABLE OF CONTENTS

ACKNOWLEDGEMENT

This paper has been long in making, and many people are responsible for the outcome. In particular, I would like to thank my advisor Dr.-Ing. Ketema Alemu (Associate Professor) for his constant unfailing advice through out my studies in the graduate school.

I would like to thank the graduate school for giving me support and the Department of Electrical Engineering for allowing to use the computer laboratory. Last but not least, special thanks to my family, friends, and Eng'r Elias Alemu who have given their support both materially and financially through out my studies in the graduate school.

ABSTRACT

This paper presents the design and hardware implementation of the (7,3) maximum-length binary cyclic code applied to a single channel communication system. In hardware implementation of the system a PC with a data acquisition board with time sharing for interfacing the analog signals. A 12-bit digital output of the PC is divided into blocks of 3-bits for processing by the channel encoder

The implementation of the system is carried out using shift registers and logic gates. A sinusoidal input waveform is applied to the system input and a circuit designed with a combination of D-type flip-flops and logic gates is used to introduce the effects of a single-random-error and a double-adjacent-burst error to observe the performance of the system. The code resulted in good performance in correcting a single-random-error and a double-adjacent-burst-error.

CHAPTER 1

Information Transmission and Error Control Principles

In today's communication technology, due to the emergence and fast expansion of large scale, high speed data networks for the exchange, processing, and storage of digital information there is an increasing need for efficient and reliable digital transmission and storage systems. A major concern in the design of these systems is the control of errors for reliable reproduction of data.

1.1. Digital Communication Systems

The basic functional elements of a one way digital communica-tion system are illustrated by the general block diagram shown in figure 1.1. In practical communication

system other functional blocks, such as equalizers, clock recovery networks etc., not shown in the figure exist.

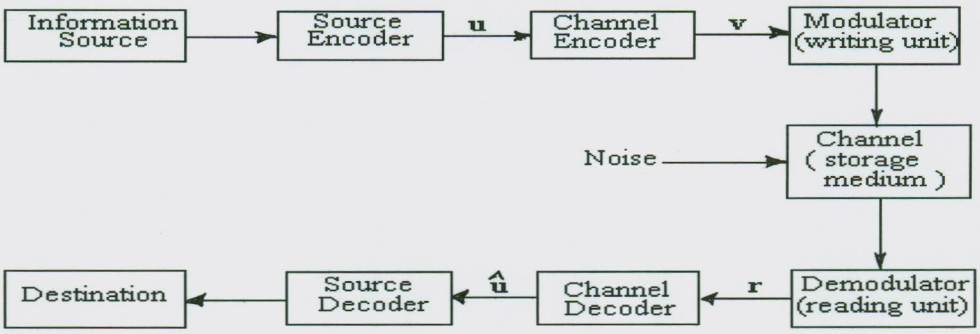


Figure 1.1. Digital communication using forward error control coding.

In the transmission or storage of digital information, data is transferred from an information source to a destination [or user].

The channel accepts electrical or electromagnetic signals and the output of the channel is a distorted version of the input signal due to the nonideal nature of the communication channel. The channel distortion and noise introduce errors in the information being transmitted and limits the rate at which information can be communicated from the source to the destination. The main function of the coder, the modulator, the demodulator, and the decoder is to combat the degrading effects of the channel on the signal and maximize the information rate and accuracy.

1.1.1. Information source

The information sources may be either analog or continuous; an analog source producing a time-continuous signal, while a discrete source producing sequences of discrete symbols. Typical examples of an analog source are speech signals, radar outputs,

and photographic scan data, etc., while discrete sources are such as computer data files and messages generated at teleprinter terminals.

The transformation of an analog source to a discrete information source takes place through the process of sampling and quantizing. Discrete information sources are characterized by the following parameters:

1. Source alphabet(symbols or letters)
2. Symbol rate
3. source alphabet probabilities
4. Probabilistic dependence of symbols in a sequence

From the above parameters, we can construct a probabilistic model of the information source and define the source entropy (***H***) and source information rate (***R***) in bits per symbol and bits per second, respectively.

An important parameter of a discrete source is its entropy. The entropy of a source, denoted by ***H***, refers to the average information content per symbol in a long message and is given the units of bits per symbol where bit is used as an abbreviation for binary digit. If we assume that all symbols occur with equal probabilities in a statistically independent sequence, and the total possible symbols are ***M*** then the source entropy is $\log_2 M$ bits per symbol.

The source information rate is defined as the product of the source entropy and the symbol rate and has the units of bits per second. The information rate denoted by ***R***, represents the minimum number of bits per second that will be needed on the average, to represent the information coming out of the discrete source. Alternatively, ***R*** represents the minimum average data rate needed to convey the information from the source to the destination. The purpose of a communication system is to deliver the source data to a user

at the rate $\boldsymbol{R}$ and at some required level of accuracy, which is stated as an upper limit on acceptable *probability of bit error* or *bit-error rate* in the delivered information. The over all efficiency of the communication system is directly related to the amount of signal energy needed to deliver each information bit with the required accuracy. This amount of energy is denoted by $\boldsymbol{E_b}$ and represents the required energy per information bit. The required signal power is therefore given by $\boldsymbol{S=E_bR}$

1.1.2. Source Encoder/Decoder

The *source encoder* transforms the source output into a sequence of binary digits called the information sequence **u**. In the case of a continuous source, this involves analog-to-digital (A/D) conversion. The source encoder is ideally designed so that (1) the number of bits per unit time required to represent the source output is minimized, and (2) the source output can be reconstructed from the information sequence **u** without ambiguity.

At the receiver, the source decoder transforms the *estimated sequence* $\hat{\mathbf{u}}$ which is an output of the channel decoder into an *estimate* of the source output and delivers this estimate to the *destination*. When the source is continuous, this involves digital-to-analog (D/A) conversion.

1.1.3. Transmission Channel

The term *transmission channel* includes all the operations required to prepare the baseband modulated waveforms for transmission through the physical channel, the transmission medium itself, and the receiving operations required to bring the signals to the point just prior to demodulation. In this way, any practical limitations or impairments in the equipments are included in the transmission channel.

Due to physical limitations, communication channels have only finite bandwidth, and the information bearing signal often suffers amplitude and phase distortion as it travels over the channel. In addition to the distortion, the signal power also decreases due to the attenuation of the channel. Furthermore, the signal is corrupted by noise. While some of the degrading effects of the channel can be removed or compensated for, the effects of noise cannot be completely removed.

Transmitted signal power is one of the means to minimize the effects of noise and provide a required level of accuracy. However, signal power cannot be increased due to established industry standards, and practical physical and economic limitations. An increase in signal power implies increase in size, weight, and cost of transmitting equipment. Particularly, applications utilizing the low regions of the radio spectrum require an enormous amount of energy to radiate usable signals, and therefore transmitted signal power is a main factor in the cost of the system. Even if the cost is acceptable, the signal power cannot be increased beyond a certain level due to the nonlinear effects that become dominant as the signal amplitude is increased. For this reason the signal-to-power ratio (***S/N***) which can be maintained at the output of a communication channel, is an important parameter of the system.

Other important parameters of the channel are the usable bandwidth, amplitude and phase response, and the statistical properties of the noise. Bandwidth is a parameter governing the achievable performance, since it limits the rate at which we can modulate waveforms in the channel. Noise in received signal constitutes the most prevalent factor limiting the performance of a communication system, since it limits the ability of the demodulator to reliably distinguish one modulated waveform from another, thereby

producing errors in the demodulator output. A common form of noise disturbance present in any communication system is the *additive white Gaussian noise* **(AWGN)**, such as the thermal noise which is always present in the electrical circuitry, for example, in the front end of the receiving equipment. Since *white Gaussian noise* is defined as a random process and has a power spectral density which is flat over the entire frequency range, errors tend to occur independently from one signalling interval to the other. Another type of noise affecting a communication system is an impulsive noise due to lightning discharges, and impulsive noise arising from transients in switching equipments and accidental circuit interruptions. Impulsive noise is characterized by relatively quite intervals punctuated by short periods of intens noise resulting in long error-free intervals interspersed with short *bursts* of errors. If the different characteristics of Gaussian noise channels and burst-error channels are taken into account quite different coding techniques are applied for each. But usually, the modulation and coding schemes are selected with a view toward the limitations imposed by the Gaussian background noise and then certain features of the coding implementation are adapted to the particular characteristics of the burst-error phenomena if they are concern to the application at hand.

1.1.4. Channel Encoder/Decoder

Digital channel coding is a practical method of realizing high transmission reliability and efficiency that otherwise be achieved by the use of signals of longer duration in the modulation/demodulation process. With digital coding, a relatively small set of analog signals, often two, is selected for transmission over the channel and the demodulator has the conceptually simple task of distinguishing between two different waveforms of known shapes. Error control is accomplished by by the channel coding operation that consists of

systematically adding extra bits (parity bits) to the output of the source coder. While these extra bits themselves convey no information, but make possible for the receiver to detect and/or correct some of the errors in the information bearing bits.

There are two methods of performing the channel coding operation. In the first method, called the block coding method, the encoder takes a block of k information bits from the source encoder and adds r error control bits (parity bits) . The number of error control bits added will depend on the value of k and the error control capabilities desired. In the second method, called the convolutional coding method the information bearing message stream is encoded in a continuous fashion by continuously interleaving information bits and error control bits. Both methods require storage and processing of binary data at the encoder and decoder.

The important parameters of a channel encoder are the method of coding, rate of efficiency of the coder (as measured by the ratio of data rate at input to the data rate at output), error control capabilities, and complexity of the encoder.

The channel decoder recovers the information bearing bits from the coded binary stream. Error detection and possible error correction is also performed by the channel decoder. The decoder operates either in a block mode or in a continuous sequential mode depending on the type of coding used in the system. For block coding, the decoder accepts consecutive blocks of n demodulator outputs and produces k decoded information symbols for each block. With convolutional coding, the decoder accepts a steady stream of demodulator output and operates over the current received symbols and on some number of previous symbols, producing b decoded outputs for each group of V received symbols. For both block and convolutional codes, the decoder attempts to make definite symbol

decisions, binary or *M*-ary in accordance with the code design. The complexity of the decoder and the time delay involved in the decoder are the important design parameters.

1.1.5. Modulator/Demodulator

The function of the modulator is to match the encoder output to the transmission channel. The modulator accepts binary or *M*-ary encoded symbols and produces waveforms appropriate to the physical transmission medium, which is always analog. The modulator accepts a bit stream as its input and converts it to an electrical waveform suitable for transmission over the communication channel. It can be effectively used to minimize the effects of channel noise, to match the frequency spectrum of the transmitted signal with channel characteristics, to provide the capability to multiplex many signals, and to overcome some equipment limitations.

The important parameters of the modulator are the types of the waveforms used, the duration of the waveforms, the power level, and the bandwidth used. The modulator accomplishes the task of minimizing the effects of the channel noise by the use of large signal power and bandwidth, and by the use of waveforms that last for longer durations. While the use of increasingly large signal power and bandwidth to combat the effects of noise is an obvious method, these parameters can not be increased indefinitely because of equipment and channel limitations, as system breakdown may occur.

At the receiving end of the communication link, the demodulator provides the interface between the transmission channel and the functions that compute and deliver estimates of the transmitted data to the user. Modulation is a reversible process, and the extraction of the message from the information bearing waveform produced by the modulator is accomplished by the demodulator. The demodulator operates on the

waveform received in each separate transmission symbol interval and produces a number or a set of that represent an estimate of a transmitted binary or *M*-ary symbol.

For a given type of modulation, the most important parameter of the demodulator is the method of demodulation. In the simplest cases, the modulator is designed to make a definite decision for each received symbol, that is 0 or 1 for binary transmission or one of 0,1,...,*M*-1 for *M*-ary transmission. Such cases are referred to as *hard-decision demodulation.*

All real transmission channels are, of course, analog channels that deliver waveforms that can in principle vary continuously over some range limited only by nonlinearity in the transmission medium and the receiving equipment. Thus the demodulator can be viewed as a form of waveform filtering followed by quantization, say to Q levels. The case of hard-decision binary demodulation thus requires quantization to Q=2 levels. If the output of the binary demodulator is quantized to $Q > 2$ levels, we refer to this as *soft-decision demodulation.*

1.2. Information Theory and Channel Capacity

Information sources can be classified in to two: analog (or continuous-valued) and discrete. Analog information sources, such as a microphone actuated by a voice signal,emit a continuous-amplitude, continuous-time electrical waveform. The output of a discrete information source such as a teletype consists of sequences of letters or symbols. Analog information sources can be transformed into discrete information sources through the process of sampling and quantizing.

1.2.1. Measure of Information

The output of a discrete information source is a message that consists of a sequence of symbols. The information content of one message drawn from a set of M equally likely messages is log M, where the logarithm base is arbitrary and depends on the basic unit of information. Based on this logarithmic measure of information we can define the *rate of an information source*. Assuming a binary source producing equally likely M-ary symbols, with the output symbols being independent from one symbol interval to the next, the rate of information source is $\log_2 M$ bits per symbol.

A measure of information for general sources can be provided with the application of the concept of *entropy* to an information source. Entropy provides an appropriate measure of the a priori uncertainty of any symbol or message to be produced by a discrete information source. Let, a source produces any one of M symbols, where the probabilities of occurrence are $p_1, p_2, ..., p_M$, with $p_1+p_2+...+p_M=1$. The entropy of the source is defined by

$$H(X) = -\sum_{i=1}^{M} p_i^t \log_2 (p_i^t) \qquad bits/symbol \qquad (1.1)$$

This definition in effect averages the logarithmic information measure with respect to the set of probabilities of the individual source symbols. It can be seen that for a set of equally likely symbols, the entropy of the source is simply $H = -\log p_i = \log M$. The entropy function H provides a measure of the average amount of information "produced" per symbol by the source.

1.2.2. Information Transmission Through a Channel

The entropy H of an information source is viewed as a measure of the prior uncertainty about the information produced by the source. Therefore, from the point of view of the information recipient, H represents the state of uncertainty before receiving information from the source.

For an M-ary discrete memoryless channel with a statistically independent input symbol sequence, the average amount of information per symbol going into the channel is given by the entropy of the input random variable X as

$$H(X) = -\sum_{i=1}^{M} p_i^t \log_2 p_i^t \qquad (1.2)$$

Similarly, the entropy of the output Y is defined as

$$H(Y) = -\sum_{i=1}^{M} p_i^r \log_2 (p_i^r) \ bits/symbol \qquad (1.3)$$

which represents the average number of bits per symbol needed to encode the output of the channel.

Given the entropy function to measure a priori uncertainty of the source information, we can use a corresponding function to measure the a posteriori uncertainty of the information after reception on a noisy channel, which is defined as the conditional entropy $H(X|Y)$, and is given by

$$(H|Y) = -\sum_{i=1}^{M}\sum_{j=1}^{M} P(X=i, Y=j)\log_2(P(X=i|Y=j)) \qquad (1.4)$$

which represents an *average measure of uncertainty* about X when Y is known.

For the case of error-free discrete transmission, the user receives uncorrupted source symbols, and the uncertainty about the source information vanishes as symbols are received. In the ideal case, the channel transfers information from the source to the user at an average rate of H bits per symbol, exactly the amount of uncertainty prior to transmission.

But in real cases transmission through the channel is not error-free, and thus after reception the user is left at some residual uncertainty concerning the exact identity of the transmitted information. Minimizing this residual uncertainty while making efficient use of signal energy is the essence of the communication system design problem. Consider an M-ary discrete memoryless channel accepting a statistically independent symbols at the rate of r_s symbols per second. The average rate at which information is going into the channel is given by

$$D_{in} = H(X)r_s \qquad \text{bits per second} \qquad (1.5)$$

But, due to errors, it is difficult to reconstruct the input symbol sequence with certainty by working on the received sequence. Hence, it can be said that some information is lost due to the errors.

In the previous discussion the conditional entropy $H(X|Y)$ is defined as measure of the uncertainty of the input X given the output Y. For an ideal errorless channel, we have

no uncertainty about the input given the output and $H(X|Y)$ is equal to zero, that is, no information is lost.

Therefore, knowing that the conditional entropy is zero for the ideal case wherein no information is lost, $H(X|Y)$ can be used as an indication of the information lost in the channel due to errors. Hence the amount of information transmitted over a channel can be defined by the amount of information going into the channel minus the information lost in the channel due to errors. That is, the *average rate of information transmission* D_t may be defined as

$$D_t \triangleq [H(X) - H(X|Y)]r_s \qquad \text{bits/second} \qquad (1.6)$$

If we are given the probabilities defining the channel, the information transfer rate through the channel depend upon the probabilities of occurrence of the input symbols.

1.2.3. Channel Capacity

The capacity of a noisy (discrete, memoryless) channel is defined as the maximum possible rate of information trans-mission over the channel. The maximum rate of transmission occurs when the source is "matched" to the channel. Hence, the *channel capacity* C is defined as

$$\begin{aligned} C &\triangleq \max_{P(X)} \{D_t\} \\ &= \max_{P(X)} [H(X) - H(X|Y)]\, r_s \end{aligned} \qquad (1.7)$$

where the maximum is with respect to all possible information sources; that is the

maximum is taken with respect to all possible probability distributions for the discrete random variable X.

Thus given any channel that is defined by the input-to-output transition probabilities, one can determine the maximum achievable information transfer rate through the channel by performing the indicated maximization over all possible input distributions.

If we consider a communication channel between the output of modulator and the input of the demodulator, in which transmission of continuous signal rather than discrete symbols takes place, the capacity formula takes a different form. This analog channel can be considered as an band-limited Additive White Gaussian Noise channel with bandwidth B Hz and an input to the channel a random process $X_c(t)$, which consists of the collection of all the waveforms generated by the modulator.

The capacity of the channel can be found by maximizing the rate of information transmission with respect to the distribution of Xc(t). Therefore, for band-limited AWGN channel the *channel capacity* or the maximum information transfer rate is given by

$$C = B \log_2\left(1+\frac{S}{N}\right) \qquad bits/\text{sec} \qquad (1.8)$$

where B is the channel bandwidth, and $S|N$ is the ratio of the signal to noise power falling within the bandwidth. This is the Shannon's capacity formula for the band-limited continuous AWGN channel. The above remarkable result indicates that the ultimate performance limit caused by channel noise is not the accuracy with which communication can be achieved, but the rate at which information can be reliably transmitted.

1.3.4. Shannon's Channel Coding Theorem

Claude Shannon, in his channel coding theorem, demonstrated that for every channel having a channel capacity C, if an information transfer rate $R<C$ there exist codes of block length n and rate R achieving reliable communication and having probability of incorrect decoding $P(E)$ bounded by

$$P(E) \leq 2^{-nE_b(R)} \qquad (1.9)$$

where the exponent $E_b(R)$ is a positive function of R for $R<C$ and is determined entirely by the characteristics of the channel. The implication of the bound is that for any information rate less than C, the error probability can be made arbitrarily small by increasing the code block length n while holding the code rate constant. A similar bound can be written for convolutional codes, where n is replaced by k, the code constraint length.

Therefore, Shannon's results show that it is not necessary to transmit Gaussian noise waveforms in order to achieve capacity; rather, well-chosen codes can be used to produce the same result. For a coded communication system, sequences of information bits are mapped into long codeword by the error-control encoder then into long digital waveforms by the modulator. The demodulator and decoder then utilize all the received signal energy during the transmission of a codeword in the decision-making process.

1.3. Error Control Coding

Error control coding is concerned with methods of delivering information from a source to a destination with minimum of errors. It is a design technique. It has been used exten-

sively in a digital communication system because of its cost-effectiveness in achieving efficient, reliable digital transmission. Generally, error control coding can increase signal quality from problematic to acceptable levels.

Channel encoder and channel decoder are the functional blocks in a digital communication system that, by acting together, reduce the overall probability of error. The encoder divides the input message bits into blocks of k messages bits and replaces each k bit message block D with an n bit codeword C_w by adding n-k parity check bits to each message block. The decoder looks at the received version of the codeword C_w, which may occasionally contain errors, and attempts to decode the k message bits. While the check bits convey no new information to the receiver, they enable the decoder to detect and correct transmission errors thereby reduce the probability of error. The design of the encoder and the decoder consists of selecting rules for generating codeword from the message blocks and for extracting the message blocks from the received version of the codeword. The mapping rules for coding and decoding are to be chosen such that error control coding lowers the overall probability of error.

1.3.1. Types of Codes

Error control codes are often divided into two broad categories; *block codes* and *convolutional codes*. The encoder for a block code divides the information sequence into message blocks of k information bits each, represented by the binary k-tuple $\mathbf{u} = (u_1,u_2,...,u_k)$ called a *message*. Corresponding to the 2^k different possible messages, there are 2^k different possible code words at the encoder output. This set of 2^k code words of length n is called an (n,k) *block code*. The ratio $R=k/n$ is called the *code rate* and represents

the number of information bits entering the encoder per transmitted symbol. Encoders for block codes can be implemented using a combinational logic circuit.

The encoder for a convolutional code also accepts k-bit blocks of the information sequence **u** and produces an encoded sequence(code word) **v** of n-symbol blocks. But in convolutional coding, the symbols **u** and **v** are used to denote sequence of blocks rather than a single block. However, each encoded block depends not only on the corresponding k-bit message block at the same time unit, but also on m previous message blocks. Hence, the encoder has a *memory order* of m. The set of encoded sequence produced by a k-input, n-output encoder of memory order of m is called an (n,k,m) *convolutional code*. The ratio $R=k/n$ is called the *code rate*. Since the encoder contains memory, it must be implemented with a sequential logic circuit.

1.3.2. Types of Errors

Transmission errors in a digital communication system are caused by the communication channel. In general, two kinds of noise can be distinguished in a communication channel. The first kind, *Gaussian noise*, is a main concern in designing and evaluating modulators and demodulators. The transmission errors introduced by white Gaussian noise are referred to as *random errors*.

A second kind of noise common to a communication channel is called *impulse noise*. An impulse noise is characterized by long quiet intervals followed by high amplitude noise bursts. When an impulse noise occurs, it affects more than one symbol or bit, and there is usually a dependence of errors in successive transmitted symbols. Thus errors due to impulse noise occur in *bursts*.

While the codes devised for correcting random errors are called *random-error-correcting codes*, codes for correcting burst errors are called *burst-error-correcting codes*. some channels contain a combination of both random and burst errors. These are called compound channels and codes devised for correcting errors on these channels are called *burst-and -random-error-correcting codes*.

1.3.3. Error Control Schemes

Error control coding can be categorized as a *forward-error-correction* (FEC), *automatic-repeat-request* (ARQ), or as a combination of FEC and ARQ (*hybrid*), depending on the method of controlling errors at the receiver used by the channel decoder. The communication system depicted in Fig.1.1 shows a one way system employing a forward error control scheme. In this scheme a message is encoded, transmitted over the communication channel and received; then an attempt is made to decode and deliver whatever data is available. That attempt may be successful or unsuccessful, but in either case no further processing is done.

If system considerations permit, errors can be handled in an entirely different manner. In an automatic repeat request (ARQ) scheme, whenever the receiver detects an error in the transmitted message, it sends a retransmission request to the transmitter over a feedback channel. These requests are repeated until the message is received correctly. Although ARQ is known for its simplicity, it has a major shortcoming: the throughput efficiency may be highly dependent on channel conditions. Hence, at low SNR a successful transmission may involve a very long time delay, which is a behaviour unacceptable for delay-sensitive applications. One approach to reducing the time delay is the hybrid FEC/ARQ scheme. Here an ARQ scheme is used to obtain a desired error rate. FEC coding

is used to correct low-weight error patterns in each message, reducing the number of retransmission requests.

CHAPTER 2

Block Codes For Error Control

Block codes can be described as a parity-check code in which the *encoder* accepts k *information digits* from the information source and appends a set of r *parity-check digits*, which are derived from the information digits in accordance with a prescribed encoding rule. The encoding rule determines the mathematical structure of the code. The information and parity digits are transmitted as a block of $n=k+r$ digits over the communication channel. The code is referred to as an (n,k) *block code*, with the n-bit block called the *code block* or *code-word* and n the *block length* of the code.

2.1 Algebraic System for Coding

Groups, Fields, and Vector spaces are the three fundamental algebraic systems used to understand and provide the mathematical framework for the description of error-detecting and error-correcting codes. For instance, the description defining a linear block code as a set

of vectors in an n-dimensional vector space over a finite field could be related to the generator matrix and parity-check matrix for the code.

2.1.1. Groups

A *group* is a system of elements upon which one mathematical operation and its inverse, such as addition and subtraction or multiplication and division, are defined. If G is a set of elements, a *binary operation* * on G is a rule that assigns to each pair of elements a and b a uniquely defined third element $c=a*b$ in G. Defined formally a group G on which a binary operation * is defined must satisfy the following properties:

(i) The binary operation * is associative.

(ii) G contains an element e such that, for any a in G,

$$a * e = e * a = a.$$

This element e is called an *identity* element of G.

(iii) For any element a in G, there exists another element a' in G such that

$$a * a' = a' * a = e.$$

The element a' is called an inverse of a.

If $a * b = b * a$ for any two elements a and b, the group G is called a *com mutative* group.

The number of elements in a group is called the *order* of the group, and a group may be of *finite* or *infinite* order. A type of finite group that will be of interest is the set of integers 0,1,2,...,M-1 under *modulo-M addition*. Modulo-M is an operation that the ordinary sum of a set of integers is divided by M and the remainder is saved as a result. Conventionally the

elements of the group are referred to simply as the integers modulo-M. For instance, the integers modulo-2 represent symbols in a binary code, and the integers modulo-M can (for certain values of M) be used to represent symbols in codes constructed on M-ary alphabets.

Many groups are contained in larger groups. Let H be a non empty subset of G. The subset H is said to be a *subgroup* of G if H is closed under the group operation of G and satisfies all the conditions of a group. An important property that is of interest is that the number of elements in a finite group, its order, is a multiple of the order of each of its sub group.

2.1.2. Fields

The group concept can be extended to introduce another algebraic system, called a *field*, which is a set of elements in which addition, subtraction, multiplication, and division can be done without leaving the set. Stated more completely, the elements in a field F, must satisfy the following conditions:

(*i*) F is closed under the operations of addition and multiplications.

(*ii*) For each operation , the associative and commutative laws of ordinary arith metic hold, so that for any elements u, v, and w in F,

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

$$u+v = v+u$$

$$(u\cdot v)\cdot w = u\cdot(v\cdot w)$$

$$u\cdot v = v\cdot u$$

(*iii*) Connecting the two operations, the distributive law of ordinary arithmetic holds, so that

$$u\cdot(v+w) = u\cdot v+u\cdot w$$

(*iv*) F contains a unique additive identity element 0 and a unique multiplicative identity,

different from 0 and written 1, such that

$$u+0 = u$$

$$u \cdot 1 = u$$

for any element u in F. The two identity elements are the minimum elements that any field must contain.

(v) Each element u in the field has a unique additive inverse, denoted by $-u$, such that

$$u+(-u) = 0$$

and, for $u \neq 0$, a unique multiplicative inverse, denoted by u^{-1}, such that

$$u \cdot u^{-1} = 1$$

From the above, the inverse operations subtraction and division are defined by

$$u-v = u+(-v), \text{ for any } u,v, \text{ in } F$$

$$u \div v = u \cdot (v^{-1}), \; v \neq 0$$

where $-v$ and v^{-1} are the additive and multiplicative inverses, respectively, of v.

2.1.2.1. Finite Fields and Primitive Elements

A field having a finite number of elements is called a *finite field* or *Galois field* and is denoted by $GF(q)$, where q is the number of elements in the field. The number of elements in a finite field is called the *order* of the finite field. A finite field $GF(p^m)$ exists for any p^m, where p is a prime and m is an integer. The simplest example of a finite field is a *prime field*, $GF(p)$, where p is any prime number greater than 1 and the addition and multiplication operations are modulo-p. The relationship between $GF(p)$ and $GF(p^m)$ is such that $GF(p)$ is a *subfield* of $GF(p^m)$; that is, the elements of $GF(p)$ are a subset of the elements in $GF(p^m)$, the subset itself

having all the properties of a finite field. Equivalently, $GF(p^m)$ is called an *extension field* of $GF(p)$.

An important property of finite fields is that every finite field $GF(q)$ contains atleast one *primitive element*, called α, which has the property that the q-1 powers of α are the q-1 nonzero elements of the field. This means the nonzero field elements can be represented as $\alpha,\alpha^2,...,\alpha^{q-1}$. Therefore, the power of a primitive element generate all the nonzero elements of $GF(q)$. For an arbitrary nonzero element β in the field, the smallest positive integer n such that $\beta^n=1$ is called the *order of the element*. It follows that the order of a primitive element α is q-1.

2.1.2.2. Binary Fields and Primitive Polynomials

In section 2.1.2.1 it is mentioned that a finite field $GF(p^m)$ exists for any number p^m, where p is a prime and m is a positive integer. In general, a code can be constructed with symbols from any Galois field $GF(p^m)$, where p is a prime m is a positive integer p. However, codes constructed with symbols from the binary field $GF(2)$, or its extension $GF(2^m)$ are widely used in digital data transmission and storage systems because information in these systems is coded in binary form.

In a binary field, arithmetic is carried out using modulo-2 addition and modulo-2 multiplication. That is, the rules of ordinary arithmetic are used except that each element is its own additive inverse, and consequently addition and subtraction are the same operations. A polynomial $f(X)$ with *variable* X and coefficients from $GF(2)$ takes the form

$$f(X)=f_0+f_1X+f_2X^2+\cdots+f_nX^n,$$

where f_i=0 or 1 for $0 \le i \le n$. Polynomials over $GF(2)$ are added (or subtracted), multiplied, and divided in the usual way, but addition and multiplication of coefficients being done in $GF(2)$. Multiplying a polynomial $f(X)$ over $GF(2)$ with itself results in

$$f^2(X) = f_0^2 + (f_1 X)^2 + (f_2 X^2)^2 + \cdots + (f_n X^n)^2.$$

Since f_i=0 or 1, $f_i^2 = f_i$.

Hence,

$$f^2(X) = f_0 + f_1 X^2 + f_2 (X^2)^2 + \cdots + f_n (X^2)^n$$

For any $l \ge 0$, the above can be generalized as

$$[f(X)]^{2^l} = f(X^{2^l}). \qquad (2.1)$$

Let $\mathbf{g}(X) = g_0 + g_1 X + g_2 X^2 + \cdots + g_m X^m$ be an other polynomial over $GF(2)$. Dividing $f(X)$ by $\mathbf{g}(X)$, a quotient $q(X)$ over $GF(2)$ and remainder $\mathbf{r}(X)$ over $GF(2)$ will result such that

$$f(X) = q(X)\mathbf{g}(X) + \mathbf{r}(X) \qquad (2.2)$$

and the degree of $\mathbf{r}(X)$ is less than $g(X)$. If the remainder $\mathbf{r}(X)$ is identical to zero, then $f(X)$ is said to be divisible by $g(X)$ and $g(X)$ a factor of $f(X)$. A polynomial $p(X)$ over $GF(2)$ of degree m is called an *irreducible polynomial* (a polynomial that can not be factored) over $GF(2)$ if $p(X)$ is not divisible by any polynomial over $GF(2)$ of degree less than m but greater than zero.

An irreducible polynomial of degree m exists for any $m \ge 1$. Important property of irreducible polynomials is that any irreducible polynomial over $GF(2)$ of degree m divides $X^{2^{-1}} + 1$. An irreducible polynomial is called a *primitive polynomial* if the smallest positive integer n for which $p(X)$ divides $X^n + 1$ is $n = 2^m - 1$. For a given m, there may be more than one primitive polynomials of degree m.

2.1.2.3. Galois Field $GF(2^m)$, Construction and Its Properties

In constructing a representation for of the Galois field $GF(2^m)$ of 2^m elements($m>1$) from the binary field, first a degree-m primitive polynomial $p(X)$ over $GF(2)$ is selected and a primitive element α is found. Using the two elements 0 and 1 from $GF(2)$ and the primitive element α, the following set of elements are formed on which a multiplication operation $\cdot$ is defined:

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

Since $p(X)$ divides $X^{2^{-1}}+1$ and α is a root of $p(X)$, we have

$$X^{2^{-1}}+1 = q(X)p(X).$$

Substituting X with α, we obtain

$$\alpha^{2^{-1}} + 1 = q(\alpha)p(\alpha). \qquad (2.3)$$

Using the fact that $p(\alpha)=0$ and modulo-2 addition the above reduces to the following equality:

$$\alpha^{2^{-1}} + 1 = 1, \qquad (2.4)$$

and the set F becomes finite and contains the following elements as power representations for $GF(2^m)$:

$$F^*=\{0,1,\alpha,\alpha^2,...,\alpha^{2^{-2}}\}.$$

Elements in $GF(2^m)$ can also be represented as the p^m polynomials of degree m-1 or lower with coefficients in $GF(2)$. For $0\le i\le 2^m-1$, dividing the polynomial X^i by $p(X)$ we obtain the following:

$$X^i = q_i(X)p(X) + a_i(X),$$

Where $q_i(X)$ and $a_i(X)$ are the quotient and the remainder, respectively. The remainder $a_i(X)$ is a polynomial of degree m-1 or less over $GF(2)$ and is of the following form:

$$a_i(X)=a_{i0}+a_{i1}X+a_{i2}X^2+\cdots+a_{i,m-1}X^{m-1}.$$

Replacing X by α and using the equality that $q_i(\alpha)\cdot 0=0$, we obtain the following polynomial expression for α^i:

$$\alpha^i=a_i(\alpha)=a_{i0}+a_{i1}\alpha+a_{i2}\alpha^2+\cdots+a_{i,m-1}\alpha^{m-1}.$$

In this way, the 2^{m-1} nonzero elements, $\alpha^0,\alpha^1,\ldots,\alpha^{2^m-2}$ in F^*, are represented by *2^m - 1 distinct nonzero polynomials* of α over $GF(2)$ with degree m-1 or less.

The other representation for the field elements in $GF(2^m)$ is the vector representation. If $a_0+a_1\alpha+a_2\alpha^2+\cdots+a_{m-1}\alpha^{m-1}$ is the polynomial representation of a field element β, then β can be represented by an ordered sequence of m components, called an m-tuple as

$$(a_0,a_1,a_2,\ldots,a_{m-1})$$

where the m components are the m coefficients of the polynomial representation of β, which implies that there is a one-to-one correspondence between the polynomial representation of β and the m-tuple. The zero element 0 of $GF(2^m)$ is represented by the zero m-tuple (0,0,...,0). Addition of two components of $GF(2^m)$ is carried simply by adding modulo-2 the corresponding components of their m-tuple representations.

2.1.3. Vector Spaces and Their Matrix Representations

A *vector space* is one of the most important algebraic concepts used in the mathematical description of codes. The concept of a vector space V over a field F brings together a set of vectors and a set of field elements, called scalars, under the operations of addition and multiplications in a mathematical system very much like a system of geometric vectors, real numbers, and ordinary algebra. Vectors can be added(vector addition), and a scalar can multiply a vector(scalar multiplication). The addition of any two vectors **u** and **v** in V, written

as $\mathbf{u}+\mathbf{v}$, produces a vector that is also in V. The multiplication of a scalar a in F by a vector $\mathbf{v}$ in V, written as $a\mathbf{v}$, also yields a vector in V.

A type of vector space of special interest used in defining *error-control* codes is a vector space over finite fields in which the scalars represent code symbols and the vectors represent codewords. A vector space over $GF(2)$ is the most important vector space which plays a central role in coding theory. Consider an ordered sequence of n elements $u_0,u_1,...,u_{n-1}$ where each component u_i is an element from the binary field $GF(2)$ (i.e. u_i=0 or 1). This is called an *n-tuple over* $GF(2)$. Since there are two choices for each u_i, 2^n distinct n-tuples can be constructed and this set of n-tuples is denoted by V_n. The addition of two n-tuples is defined by the element by element addition as

$$\mathbf{v} + \mathbf{u}=(u_0+\upsilon_0,u_1+\upsilon_1,...,u_{n-1}+\upsilon_{n-1})$$

where each addition $u_i+\upsilon_i$ is performed in $GF(2)$, and $\mathbf{u}+\mathbf{v}$ is an n-tuple over $GF(2)$. The scalar multiplication of an n-tuple $\mathbf{v}$ in V_n by an element a from $GF(2)$ is defined as the element-by-element multiplication as

$$a\cdot(\upsilon_0,\upsilon_1,...,\upsilon_{n-1}) = (a\cdot\upsilon_0,a\cdot\upsilon_1,...,a\cdot\upsilon_{n-1})$$

where each multiplication $a\cdot\upsilon_i$ is carried out in $GF(2)$, and the resulting vector is an n-tuple in V_n. From the above it is possible to show that the addition and multiplication rules just given satisfy the distributive and associative laws, and therefore, the set V_n of all n-tuples over $GF(2)$ forms the vector space over $GF(2)$. The construction of a vector space of all n-tuples over a finite field F can be done in a similar way.

A subset S containing at least one vector in a vector space V over a field F is called a subspace of V if S itself has all the properties of a vector space with respect to the operations of addition and scalar multiplication in V. If we take a subset of vectors $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$ from V over

F, the set S of all vectors formed by a linear combinations of $\mathbf{v}_1,\mathbf{v}_2,\ \ ...,\mathbf{v}_k$ over F constitutes a vector space. Let $a_1,a_2,...,a_k$ be k scalars from F. The sum

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k$$

is called a linear combination of $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$. clearly, the sum of two linear combinations of $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$, and the product of a scalar c in F and a linear combination are also linear combinations of the k vectors $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$.

In general, if $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$ are vectors in a vector space V over a field F, then a *linear combinations* of $\mathbf{v}_1,\mathbf{v}_2,\ \ ...,\mathbf{v}_k$ is any sum of the form

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k$$

where each a_i is in the field F. The given set of vectors $\{\mathbf{v}_i\}$ is said to *span* a vector space V if any vector in V can be generated by a linear combination of the vectors in the set.

A set of k vectors $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$ is said to be *linearly independent* if no set of scalars $a_1,a_2,...,a_k$ (except all a_i=0) exists such that

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k = \mathbf{0}$$

where $\mathbf{0}$ denotes the zero vector. If the above equation is satisfied for at least one set of scalars not all equal to zero, the vectors $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_k$ are said to be *linearly dependent*. In any vector space there is at least one set of linearly independent vectors that spans the space. Any such set is said to be a *basis* (or *base*) of the vector space. All bases of a given vector space contain the same number of vectors; the number is called the *dimension* of the vector space.

As an example consider the vector space Vn of all n-tuples over $GF(2)$. Then form n n-tuples as:

$$\mathbf{e}_0 = (1,0,0,0,\ldots,0,0)$$
$$\mathbf{e}_1 = (0,1,0,0,\ldots,0,0)$$
$$\vdots$$
$$\mathbf{e}_{n-1} = (0,0,0,0,\ldots,0,1)$$

where the n-tuple $\mathbf{e}_i$ has only one nonzero component at ith position. Every n-tuple $(a_0,a_1,a_2,\ldots,a_{n-1})$ in V_n can be expressed as a linear combination of $\mathbf{e}_0,\mathbf{e}_1,\ldots,\mathbf{e}_{n-1}$ as:

$$(a_0,a_1,a_2,\ldots,a_{n-1}) = a_0\mathbf{e}_0+a_1\mathbf{e}_1+a_2\mathbf{e}_2+\cdots+a_{n-1}\mathbf{e}_{n-1}.$$

Therefore, $\mathbf{e}_0,\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}n$-1 span the vector space V_n of all n-tuple over $GF(2)$, and they are also linearly independent. Hence they form a bases for V_n and the dimension of V_n is n. If $k<n$ and $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k$ are k linearly independent vectors in V_n, then all the linear combinations of $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k$ of the form

$$\mathbf{u} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k$$

form a k-dimensional subspace S of V_n. Since each c_i has two possible values, 0 or 1, there are 2^k possible distinct linear combinations of $\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_k$. Thus S consists of 2^k vectors and is a k-dimensional subspace of V_n.

Let $\mathbf{u}=(u_0,u_1,\ldots,u_{n-1})$ and $\mathbf{v}=(v_0,v_1,\ldots,v_{n-1})$ be two n-tuples in V_n. The *inner product* (or *dot product*) of $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\mathbf{u}\cdot\mathbf{v} = u_0v_0 + u_1v_1 + \cdots + u_{n-1}v_{n-1}, \qquad (2.6)$$

where $u_i \cdot v_i$ and $u_i \cdot v_i + u_{i+1} \cdot v_{i+1}$ are carried in modulo-2 multiplication and addition. Hence, the inner product $\mathbf{u}\cdot\mathbf{v}$ is scalar in $GF(2)$. If $\mathbf{u}\cdot\mathbf{v}=0$, $\mathbf{u}$ and $\mathbf{v}$ are said to be *orthogonal* to each other.

Consider a k-dimensional subspace S of V_n and let S_d be the set of vectors in V_n such that, for any **u** in S and **v** in S_d, **u**·**v**=0. The set S_d contains at least the all- zero n-tuple 0=(0,0,...,0), since for any **u** in S, **0**·**u**=**0**. Thus S_d is nonempty. S_d forms a subspace of V_n, and is called the null (or dual) space of S. Conversely, S is also the null space of S_d. The dimension of the null space S_d is n-k (i.e., $\dim(S) + \dim(S_d) = n$).

It is convenient to write the linear combinations of basis vectors in a matrix notation. A $k \times n$ matrix over $GF(2)$ (or over any other field) is a rectangular array with k rows and n columns,

$$\boldsymbol{G} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,n-1} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix} \qquad (2.6)$$

where each entry g_{ij} with $0 \le i < k$ and $0 \le j < n$ is an element from the binary field $GF(2)$. It can be observed that each row of **G** is an n-tuple over $GF(2)$ and each column is a k-tuple over $GF(2)$. The matrix **G** can also be represented by its k rows $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ as

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \cdot \\ \cdot \\ \cdot \\ g_{k-1} \end{bmatrix} \qquad \text{f}$$

If the $k(k \leq n)$rows of **G** are linearly independent, then the 2^k linear combinations of these rows form a k-dimensional subspace of the vector space V_n of all the n-tuples over $GF(2)$. This subspace is called the row space of **G.** Performing elementary row operations on **G**, another matrix **G'** results which have the same row space.

Let S be the row space of a kxn matrix **G** over $GF(2)$ whose k rows $\mathbf{g}_0,\mathbf{g}_1,...,\mathbf{g}_{k-1}$ are linearly independent. Let S_d be the null space of S. Then the dimension of S_d is n-k. Let $\mathbf{h}_0\mathbf{h}_1$-$...,\mathbf{h}_{n-k-1}$ be n-k linearly independent vectors in S_d. An $(n-k)xn$ matrix **H** can be formed using $\mathbf{h}_0,\mathbf{h}_1,...,\mathbf{h}_{n-k-1}$ as rows as follows:

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \cdot \\ \cdot \\ \cdot \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,n-1} \\ h_{10} & h_{11} & \cdots & h_{1,n-1} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix} \qquad (2.7)$$

The row space of **H** S_d. Since each row $\mathbf{g}_i$ of **G** is a vector in S and each row $\mathbf{h}_j$ of **H** is a vector in S_d, the inner product of $\mathbf{g}_i$ and $\mathbf{h}_j$ must be zero (i.e., $\mathbf{g}_i \cdot \mathbf{h}_j = 0$). Since the row space S of **G** is the null space of the row space S_d of **H**, S is called the null (or dual) space of **H**.

2.2. Structure of Linear Block Codes

Assuming the output of an information source is a sequence of binary digits an (n,k) *linear block code* over a field $GF(2)$ is a k-dimensional vector subspace of the space of all n-tuples over $GF(2)$. Thus it can be stated that the set of all n-bit vectors formed by linear combinations over $GF(2)$ of k linearly independent basis vectors $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ is a binary (n,k) linear block code C. That is, a vector **v** is an (n,k) codeword in C if and only if it lies in the k-dimensional vector space spanned by $\{\mathbf{g}_i\}$.

2.2.1. Description of Linear Block Codes

The encoding operation in a linear block encoding scheme consists of two basic steps: (1) the information sequence is segmented into message blocks, each block consisting of k successive information bits; (2) the encoder transforms each message block into a larger block of n bits according to certain rules. These n-k additional bits are generated from linear combinations of the message bits, If the $\{\mathbf{g}_i\}$ are arranged as rows of a $k \times n$ matrix **G**, a code word **v** can be expressed as

$$\mathbf{v} = (u_0, u_1, \ldots, u_{k-1}) \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} = \mathbf{u} \cdot \mathbf{G} \qquad (2.8)$$

where **u** is a *k*-bit information vector and **G** is called the *generator matrix* of the code. The full set of codewords, referred to simply the *code*, is generated by letting **u** range through the set of all 2^k binary *k*-tuples. It follows from Eqn.2.8 that an (*n*,*k*) linear code is completely specified by the *k* rows of a generator matrix **G**. Therefore, the encoder only has to store the *k* rows of **G** and to form a linear combination of these *k* rows based on the input message $\mathbf{u}=(u_0, u_1,\ldots, u_{k-1})$.

A desirable property for a linear block code to possess is the *systematic structure* of the code words, where a code word is divided into two parts, the message part consisting of *k* unaltered information (or message) digits and the redundant checking part consisting of *n-k* parity-check digits, which are *linear sums* of the information digits. A linear block code with this structure is referred to as a *linear systematic block code*.

A linear systematic (*n*,*k*) code is completely specified by a *k*x*n* matrix **G** of the following form:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ . \\ . \\ . \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ & & & & & & & & \\ & & & & & & & & \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (2.9)$$

where p_{ij}=0 or 1. Denoting the *k*x*k* identity matrix by $\mathbf{I}_k$, then $\mathbf{G}=[\mathbf{P}\ \mathbf{I}_k]$. If the message to be encoded is $\mathbf{u}=(u_0,u_1,\ldots,u_{k-1})$, then the corresponding code word will be

$$V = (u_0, u_1, \ldots, u_{n-1}) \quad (2.10)$$

$$= (u_0, u_1, \ldots, u_{k-1}) \cdot G$$

It follows from Eqn.2.4 and Eqn.2.5 that the components of **v** are

$$v_{n-k-i} = u_i \qquad for\ 0 \le i < k \quad (2,11a)$$

and

$$V_j = u_0 p_{o,j} + u_1 P_{i,j} + \ldots + u_{k-1} p_{k-1,j} \quad (2.11b)$$

for $0 \le j < n-k$.

Eqn.(2.11a) shows that the rightmost digits of a codeword **v** are identical to the information digits $u_0,u_1,\ldots,u_{k-1}$ to be encoded and Eqn.(2.11b) shows that the leftmost *n-k* redundant digits are linear sums of the information digits. The *n-k* equations given by Eqn.(2.11b) are called parity-check equations of the code.

With every (n,k) linear block code, and its kxn matrix **G** with k linearly independent rows, there exists an $(n\text{-}k)$xn matrix **H**, called parity-check matrix of the code, with $n\text{-}k$ linearly independent rows such that any vector in the row space of **G** is orthogonal to the rows of **H** and any vector that is orthogonal to the rows of **H** is in the row space of **G**. Hence, an (n,k) linear block code generated by **G** can be described alternatively as follows: An n-tuple **v** is a code word in the code generated by **G** if and only if $\mathbf{v}.\mathbf{H}^T=0$. The 2^{n-k} linear combinations of the rows of matrix **H** form an $(n,n\text{-}k)$ linear code C_d. This code is the null space of the (n,k) linear code C generated by matrix **G**. C_d is called the dual code of C. Therefore a parity-check matrix for a linear code C is a generator matrix for its dual code C_d.

$$\boldsymbol{H} = [\boldsymbol{I}_{n-k} \quad \boldsymbol{P}^T]$$

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & p_{02} & p_{12} & \cdots & p_{k-1,2} \\ \cdot & & & & & & & & \\ \cdot & & & & & & & & \\ \cdot & & & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \quad (2.12)$$

If the generator matrix of an (n,k) linear code is in the systematic form of Eqn.(2.9), the parity check matrix takes the following form:

where $\mathbf{P}^T$ is the transpose of the matrix **P**. Let $\mathbf{h}_j$ be the jth row of **H**. It can be shown that the inner product of the ith row of **G** given by Eqn.(2.9) and the jth row of **H** given by Eqn.(2.12) is

$$\mathbf{g}_i \cdot \mathbf{h}_j = p_{ij} + p_{ij} = 0 \quad \text{for } 0 \le i < k \text{ and } 0 \le j < n\text{-}k,$$

which implies that $\mathbf{G}\cdot\mathbf{H}^T=0$ and the $n\text{-}k$ rows of **H** are linearly independent

The parity-check equations given by Eqn.(2.11b) can also be obtained from the parity-check matrix **H** of Eqn.(2.12). If $\mathbf{u}=(u_0,u_1,...,u_{k-1})$ is the message to be encoded, then the systematic form of the corresponding codeword would be

$$\mathbf{v}=(v_0,v_1,...,v_{n-k-1},u_0,u_1,...,u_{k-1})$$

Using the fact that $\mathbf{v}\cdot\mathbf{H}^T=0$, we obtain

$$v_j + u_0p_{0j} + u_1p_{1j} + \cdots + u_{k-1}p_{k-1,j} = 0 \qquad (2.13$$

For $0\le j<n-k$. Rearranging Equ.(2.13), we obtain the same parity-check equations of Eqn.(2.6b). Therefore, an (n,k) linear block code is completely specified by its parity-check matrix.

Based on the equations of Eqn.(2.11a) and Eqn. (2.11b), the encoding circuit for an (n,k) linear systematic code can be implemented as shown in Fig. 2.1.

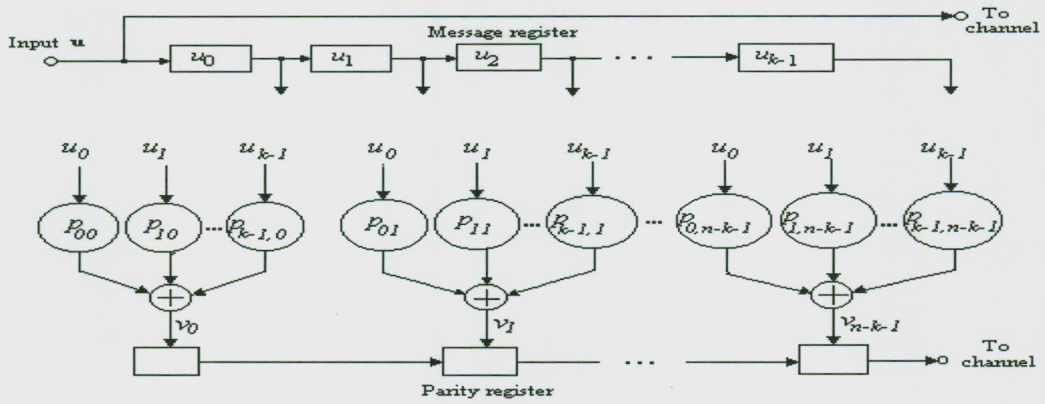


Fig. 2.1 Encoding circuit for a linear systematic (n,k) code.

In Fig. 2.2 the symbol →□→ is used to denote the flip-flops that make up a shift register, -⊕- a modulo-2 adder, and a closed path if $p_{ij}=1$ and an open path if $p_{ij}=0$. During the encoding operation, the message $\mathbf{u}=(u0,u1,...,u_{k-1})$ to be encoded is shifted into the message reg-

ister and simultaneously into the channel. As soon as the entire message has entered the message register, the *n-k* parity-check digits are formed at the outputs of the modulo-2 adders. The parity-check digits are then serialized and shifted in to the channel. It can be seen from the figure that, unless the code has some form of structure to utilize, the complexity of the encoding circuit is linearly proportional to the block length of the code.

2.2.2. Error Detection and Error-Correction

While the generator matrix is used in the encoding operation, the parity-check matrix is used in the decoding operation. For an (n,k) linear code with generator matrix **G** and parity-check matrix **H**, let **v** be a code word that was transmitted over a noisy channel and **r** be the received vector at the output of the channel. Then **r** is given by the vector sum of the code word **v** and an error vector **e**, that is,

$$\mathbf{r} = \mathbf{v} + \mathbf{e} \tag{2.14}$$

The decoder does not know **v** and **e**; its function is to decode **v** from **r**. The decoder does the decoding operation by computing the $(n-k)$-tuple **S** defined as:

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T \tag{2.15}$$

s is called the *syndrome* of **r**. Thus the syndrome of the received vector is zero if **r** is a valid code word, or $\mathbf{s} \neq 0$ if **r** is not a code word and the receiver assumes the transmitted code word is in error. When the error pattern is identical to a nonzero code word, the errors are not detectable (i.e., **r** contains errors but $\mathbf{s}=\mathbf{r} \cdot \mathbf{H}^T = 0$) and the decoder makes a decoding error. Error patterns of this type are called *undetectable error patterns*. Since there are 2^k-1 nonzero code words, there are 2^k-1 undetectable error patterns

The syndrome **s** is simply the vector sum of the received parity digits and the parity-check digits recomputed from the received information digits. But actually the syndrome **s** depends only on the error pattern **e**, and not on the transmitted code word **v**. Since **r** is the vector sum of **v** and **e**, it follows from Eqn.(2.15) that

$$\mathbf{s} = \mathbf{r}\cdot\mathbf{H}^T = (\mathbf{v} + \mathbf{e})\cdot\mathbf{H}^T = \mathbf{v}\cdot\mathbf{H}^T + \mathbf{e}\cdot\mathbf{H}_T$$

However, $\mathbf{v}\cdot\mathbf{H}^T=0$. Consequently, the following relation between the syndrome and the error pattern is obtained:

$$\mathbf{s} = \mathbf{e}\cdot\mathbf{H}^T \qquad (2.16)$$

If H is expressed in its systematic form as given by Eqn.(2.12), multiplying out $\mathbf{e}\cdot\mathbf{H}^T$ yields the following linear relationship between the syndrome digits and the error digits:

$$\begin{aligned} s0 &= e0 + e_{n-k}p00 + e_{n-k+1}p10 + \ldots + e_{n-1}p_{k-1,0} \\ s1 &= e1 + e_{n-k}p01 + e_{n-k+1}p11 + \ldots + e_{n-1}p_{k-1,1} \\ &\vdots \\ s_{n-k-1} &= e_{n-k-1} + e_{n-k}p_{0,n-k-1} + e_{n-k+1}p_{1,n-k-1} + \ldots + e_{k-1,n-k-1}. \end{aligned} \qquad (2.17)$$

Since the syndrome vector provides information about the error digits, it can be used for error correction. Therefore, any error correction scheme is a method of solving the $n-k$ linear equations of Eqn.(2.17) for the error digits and taking the vector sum $\mathbf{r}+\mathbf{e}$ as the actual transmitted code word. But the $n-k$ linear equations of Eqn.(2.17) have 2^k solutions, i.e., 2^k error patterns result in the same syndrome. Since the true error pattern **e** is just one of them, the decoder has to determine the true error vector from a set of 2^k candidates. To minimize the probability of decoding error, the most probable error pattern that satisfies the Eqn.(2.17) is chosen as the true error pattern. If the channel is a BSC, the most probable error pattern is the one that has the smallest number of nonzero digits

In describing the error-correction power of linear block codes there are three important concepts to be defined; the *Hamming distance* between two vectors, the *Hamming weight* of

a code vector, and the *minimum distance* of a code. The *Hamming weight* of a vector is defined as the number of nonzero elements in the vector. The *Hamming distance* between two vectors **v** and **w**, denoted by $d(\mathbf{v},\mathbf{w})$, is defined as the number of positions they differ. The Hamming distance is a metric function that satisfies the triangle inequality. If **v**,**w**, and **x** are three *n*-tuples, then

$$d(\mathbf{v},\mathbf{w}) + d(\mathbf{w},\mathbf{x}) \geq d(\mathbf{v},\mathbf{x}) \qquad (2.18)$$

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

$$d(\mathbf{v},\mathbf{w}) = W(\mathbf{v}+\mathbf{w}). \qquad (2.19)$$

The *minimum distance* d_{min} of a linear block code *C* is the smallest of the Hamming distances between pairs of different code words in the code. That is,

$$d_{\min} = \min\{d(\mathbf{v},\mathbf{w}): \mathbf{v},\mathbf{w} \in \mathbf{C}, \mathbf{v} \neq \mathbf{w}\} \qquad (2.20)$$

It follows that the minimum distance is the smallest Hamming weight of the sum of any two different code words in *C*. Since the sum of a pair of code words is another code word, the minimum distance of a linear code is the minimum Hamming weight of the nonzero code words. Then, from Eqn.(2.20)

$$\begin{aligned} d_{\min} &= \min\{ W(\mathbf{v}+\mathbf{w}): \mathbf{v},\mathbf{w} \in C, \mathbf{v} \neq \mathbf{w} \} \\ &= \min\{ W(\mathbf{x}) \ : \mathbf{x} \in C, \ \mathbf{x} \neq 0\} \qquad (2.21) \\ &\triangleq W_{\min} \end{aligned}$$

where $W_{\min} \triangleq \{ W(\mathbf{x}): \mathbf{x} \in C, \mathbf{x} \neq 0 \}$ is the minimum weight of the linear code *C*. Therefore, for a linear block code, to determine the minimum distance of the code is equivalent to determine its minimum weight.

There exist also a key relationship between the minimum distance of a linear block code and its parity-check matrix **H**. A linear block code C can be defined as the set of all n-tuples for which $\mathbf{v}\cdot\mathbf{H}^T=0$. This can be restated by saying that a code word in C is a vector having ones in positions such that the corresponding columns of **H** sum to the zero vector **0**. Therefore, the minimum weight of any code word in C is the minimum number of distinct columns of **H** that sum to **0**. Since the minimum weight of a code word is equal to the minimum Hamming distance between any pair of different code words, the minimum number of columns of **H** summing to **0** also defines the minimum distance of the code.

The *minimum distance* $d_{\min}$ of a linear block code is the key property of a code that determines its capability to detect and correct errors. When a code vector **v** is transmitted over a noisy channel, an error pattern of l errors will result in a received vector **r** which differs from the transmitted vector **v** in l places (i.e., $d(\mathbf{v},\mathbf{r}) = l$). Since any two distinct code vectors of a code C differ in at least dmin places, no error pattern of $d_{\min}$-1 or fewer errors can change one code vector into another. Therefore, any error pattern of $d_{\min}$-1 or fewer errors will result in a received vector **r** that is not a code word in C, and the receiver detects the presence of errors. But the receiver cannot detect all the error patterns of $d_{\min}$ errors because there exists at least one pair of code vectors that differ in dmin places and there is an error pattern $d_{\min}$ errors that will carry one into the other. errors. Hence, the *random-error-detecting* capability of a block code with minimum distance dmin is $d_{\min}$-1.

Even though a block code with minimum distance $d_{\min}$ guarantees detecting all the error patterns of $d_{\min}$-1 or fewer errors, it is also capable of detecting a large fraction of error patterns with $d_{\min}$ or more errors. In fact, an (n,k) linear code is capable of detecting 2^n-2^k error patterns of length n. This can be shown as follows. Among the 2^n-1 possible nonzero error pat-

terns, there are 2^k-1 error patterns that are identical to the 2^k-1 nonzero codeword. If any of these 2^k-1 error patterns occurs, it alters the transmitted codeword **v** into another codeword **w**. Thus, **w** will be received and its syndrome is zero. In these case the decoder accepts **w** as the transmitted codeword and thus commits an incorrect decoding. Therefore, there are 2^k-1 undetectable error patterns. If an error pattern is not identical to a nonzero code word, the received vector **r** will not be a code word and the syndrome will not be zero. In these case error will be detected. There are exactly 2^n-2^k error patterns that are not identical to the codeword of an (n,k) linear code. These 2^n-2^k error patterns are detectable error patterns. For large n, 2^k-1 is in general much smaller than 2^n, and only a small fraction of error patterns pass through the decoder without being detected. Let A_i be the number of code vectors of weight i in an (n,k) linear block code C. The number A_0, A_1, ..., A_n are called the weight distribution of C, and are used to compute the probability that the decoder fails to detect the presence of errors, if the code C is used only for error detection on a BSC. Let $P_u(E)$ denote the probability of undetected error. Since an undetected error occurs only when the error pattern is identical to a non-zero code vector of C,

$$P_u(E) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} \qquad (2.22)$$

where p is the transition probability of the BSC. If the minimum distance of C is $d_{\min}$, then A_1 to $A_{d\min-1}$ are zero.

If a block code with minimum distance d_{min} is used for random-error correction, one would like to know how many errors that the code is able to correct. Let t be a positive integer such that

$$2t + 1 \leq d_{min} \leq 2t + 2 \qquad (2.23)$$

A block code with minimum distance dmin guarantees correcting all error patterns of $t=\lfloor (d_{min}-1)/2 \rfloor$ or fewer errors, where $\lfloor (d_{min}-1)/2 \rfloor$ denotes the largest integer no greater than $(d_{min}-1)/2$. The parameter $t=(d_{min}-1)/2$ is called the *random-error-correcting capability* of the code. The code is referred as a t-error-correcting code.

A block code with random-error-correcting capability t is usually capable of correcting many error patterns of $t+1$ or more errors. For a t-error-correcting (n,k) linear code, it is capable of correcting a total of 2^{n-k} error patterns, including those with t or fewer errors. If a t-error-correcting block code is used strictly for error correction on a BSC with transition probability p, the probability that the decoder commits an erroneous decoding is upper bounded by

$$P(E) \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \qquad (2.24)$$

In practice, a code is often used for correcting λ or fewer errors and simultaneously detecting l ($l > \lambda$) 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 d_{min} of the code should be at least $\lambda+l+1$. That means the random-error-detecting and

random-error-correcting capabilities of a block code are determined by the code's minimum distance.

2.2.3. Standard Array and Table-Lookup Decoding

Suppose that an (n,k) linear code C is used for error correction and $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}_{2k}$ be the code vectors of C. Let $\mathbf{r}$ be the received vector. $\mathbf{r}$ can be any one of 2^n n-tuples over $GF(2)$, and the decoder has the task of associating $\mathbf{r}$ with one of 2^k n-tuples that are valid code words. The decoder can perform this task by partitioning the 2^n n-tuples into 2^k disjoint subsets D_1, D_2, ..., D_{2k} such that each subset contains only one code vector $\mathbf{v}_i$. Then if $\mathbf{r}\in D_i$, $\mathbf{r}$ is decoded into $\mathbf{v}_i$. Correct decoding results if $\mathbf{v}_i+\mathbf{e}\in D_i$.

The method of partitioning the set of 2^n n-tuples is shown in the table below. The code vectors $\mathbf{v}_1,\mathbf{v}_2,...,\mathbf{v}^{2k}$ are placed in the first row, with the code vector of all zeros appearing in the leftmost position. The first element in the second row D_2 is any one of the (2^n-2^k) n-tuples not appearing in the first row. once $\mathbf{e}_2$ is chosen, the second row is completed by adding $\mathbf{e}_2$ to the code words as shown in the table. Having completed the second row, an unused n-tuple $\mathbf{e}_3$ is chosen to begin the third row and the sum of $\mathbf{v}_i+\mathbf{e}_3(i=1,2, ...,2^k)$ are placed in the third row. The process is continued until all of 2^n n-tuples are used. The resulting array is called the *standard array* for the code and consists of 2^k columns that are disjoint. Each column has 2^{n-k} n-tuples with the top most n-tuple as a code vector.The jth column is the partition D_j that will be used for decoding as described at the beginning of this section. The rows of the standard array are called *co-sets* and the first element in each row is called a *co-set leader*. If the error pattern caused by the channel coincides with a co-set leader, then the received vector is correctly decoded. On the other hand, if the error pattern is not a co-set leader, then an incorrect

decoding will result. Thus the co-set leaders are called *correctable error patterns.*

Table 2.1. A Standard array for an (n,k) linear block code.

$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	...	$\mathbf{v}_{2^k}$
e_2	v_2+e_2	v_3+e_2	...	$v_{2k}+e_2$
e_3	v_2+e_3	v_2+e_3	...	$v_{2k}+e_3$
.	.	.	...	.
.	.	.		.

To minimize the probability of incorrect decoding, the 2^{n-k} *co-set leaders are chosen to be the error patterns that are most likely to occur for a given channel.* For a channel in which only random errors are occurring, an error pattern of smaller weight is more probable than an error pattern of larger weight. Hence, in constructing the standard array, the co-set leader should be chosen as the vector with minimum weight from the remaining available vectors. As a result, the decoding based on the standard array is the minimum distance decoding(i.e., the maximum likelihood decoding).

For an (n,k) linear code C with minimum distance d_{min} all the n-tuples of weight $t=\lfloor (d_{min}-1)/2 \rfloor$ or less can be used as coset leaders of a standard array of C. If all the n-tuples of weight t or less are used as coset leaders, there is atleast one n-tuple of weight $t+1$ that cannot be used as coset leader. In conclusion an (n,k) linear code with minimum distance d_{min} is capable of correcting all the error patterns of $\lfloor (d_{min}-1)/2 \rfloor$ or fewer errors, but it is not capable of correcting all the error patterns of weight $t+1$.

An important property of a standard array that can be used to simplify the decoding process of an (n,k) linear code with parity-check matrix **H**, is that, all the 2^k n-tuples of a co-set

have the same syndrome and syndromes for different co-sets are different. The syndrome of an n-tuple is an $(n-k)$-tuple and there are 2^{n-k} distinct $(n-k)$-tuples. There is a one-to-one correspondence between a co-set leader (a correctable error pattern) and a syndrome.

Using these one-to-one correspondence relationship, a decoding table could be formed, which is much simpler to use than a standard array. The table consists of 2^{n-k} coset leaders (the correctable error patterns) and their corresponding syndromes. This table is either stored or wired in the receiver. The decoding of a received vector consists of three steps:

step1: Compute the syndrome of $\mathbf{r}$, $\mathbf{r}\cdot\mathbf{H}^T$.

step2: Locate the coset leader $\mathbf{e}_l$ whose syndrome is equal to $\mathbf{r}\cdot\mathbf{H}^T$. Then $\mathbf{e}_l$ is assumed to be the error pattern caused by the channel.

step3: Decode the received vector r into the code vector $\mathbf{v} = \mathbf{r}+\mathbf{e}_l$.

The above decoding scheme is called syndrome decoding or table-lookup decoding. In principle, table-lookup decoding can be applied to any (n,k) linear code. It results in minimum decoding delay and minimum error probability. However,, for large $n-k$, the implementation of this decoding scheme becomes impractical, and either a large storage or a complicated logic circuitry is needed.

There are several decoding schemes which are variations of the table-lookup decoding, each requiring additional properties in code other than the linear structure whichwill be discussed in the next section. The table-lookup decoding of an (n,k) linear code may be implemented by regarding the decoding table as a truth table of n switching functions:

$$e_0 = f_0(s_0, s_1, \ldots, s_{n-k-1}),$$

$$e_1 = f_1(s_0, s_1, \ldots, s_{n-k-1}), \qquad (2.25)$$

$$e_{n-1} = f_{n-1}(s_0, s_1, \ldots, s_{n-k-1}).$$

where $s_0, s_1, \ldots, s_{n-k-1}$ are the syndrome digits, which sre regarded as switching variables, and $e_0, e_1, \ldots, e_{n-1}$ are the estimated error digits. When these n switching functions are derived and simplified, a combinational logic circuit with the n-k syndrome digits as inputs and the estimated error digits as outputs can be realized. The general decoder for an (n,k) linear code based on the table-lookup scheme is shown in Fig.2.3.

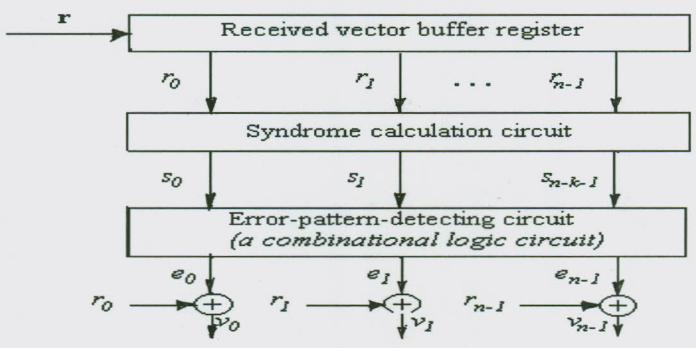


Figure 2.3 General decoder for a linear block code.

2.3. Binary Cyclic Codes

Binary Cyclic codes form a subclass of linear block codes described in the previous section. Cyclic codes attractive for two reasons. First, encoding and syndrome calculations can be easily implemented using simple shift registers with feedback connections. Second, cyclic codes have considerable mathematical structure that makes it possible to find various practical methods for decoding them.

2.3.1. Description of Binary Cyclic Codes

A binary code is said to be cyclic if the following two properties hold:

1. The code is linear, that is, the bit-by-bit addition of two code words in $GF(2)$ is again a code word.

2. Any cyclic ("end around") shift of a code word is also a code word.

The first property means that cyclic codes can be described as parity-check codes, as are all the simple codes. The second property simply means that if $\mathbf{v}=(\upsilon_0,\upsilon_1,\upsilon_2,\ldots,\upsilon_{n-1})$ is a code word, then so are all cyclic shifts, that is, $(\upsilon_{n-1},\upsilon_0,\upsilon_1,\upsilon_2,\ldots,\upsilon_{n-2})$, $(\upsilon_{n-2},\upsilon_{n-1},\upsilon_0,\upsilon_1,\ldots,\upsilon_{n-3})$, and so forth are code words.

Since a cyclic code C is a linear code, it can be described as the set of all n-tuples generated by the matrix equation $\mathbf{v}=\mathbf{u}\cdot\mathbf{G}$,where $\mathbf{u}$ is any length-k binary vector and $\mathbf{G}$ is a kxn binary matrix called the generator matrix. If $\mathbf{g}(\mathbf{X})$ is a polynomial of minimum degree from the polynomial representation of the rows of $\mathbf{G}$, it can be used to form all the code words in the code C. $\mathbf{g}(X)$ is called the *generator polynomial* of the code C.

For an arbitrary (n,k) cyclic code, the following properties hold:

1. $\mathbf{g}(X)$ has the degree $n-k=r$.

2. $\mathbf{g}(X)$ must be of the form

$$g(X) = X^r + \sum_{i=1}^{r-1} g_i X^i + 1, \qquad g_i = 0 \text{ or } 1$$

3. Every code word is a multiple of the generator polynomial.

As a consequence of property 3, encoding can be implemented as polynomial multiplication of a k-bit [degree-(k-1)] information polynomial by the generator polynomial. But this leads to a nonsystematic code. However encoding can be thought of as a division operation. For example, suppose we shift the degree-(k-1) polynomial $\mathbf{u}(X)$ left r places and divide by $\mathbf{g}(X)$. The remainder of that division when added to the $\mathbf{X}^r\mathbf{u}(\mathbf{X})$ is certainly divisible by $\mathbf{g}(X)$ and is therefore a code word. Specifically,

$$\mathbf{v}(X) = X^r\mathbf{u}(X) + [X^r\mathbf{u}(X) \bmod \mathbf{g}(X)]$$

This produces a systematic code structure. However, when describing the structure and properties of cyclic codes it is usual to visualize the code polynomials as the product of information polynomials with the generator polynomial.

2.3.2. Algebraic Structure of Binary Cyclic Codes

Cyclic codes can be described by using the algebra of polynomials. The reason for the emphasis of polynomial characterization of cyclic codes is first due to the insight gained into the structure of these codes by using the theory of polynomial algebra, and second the relative advantage of the polynomial characterization in devising efficient encoding and decoding algorithms.

If an n-tuple $\mathbf{v}=(v_0, v_1, ..., v_{n-1})$ is cyclically shifted one place to the right, the resulting n-tuple would be,

$$\mathbf{v}^{(1)}=(v_{n-1}, v_0,...,v_{n-2}).$$

Cyclically shifting $\mathbf{v}$ i places to the right results in an n-tuple

$$\mathbf{v}^{(i)}=(v_{n-i},v_{n-i+1},...,v_{n-1},v_0,v_1,...,v_{n-i-1}).$$

Clearly, cyclically shifting **v** i places to the right is equivalent to cyclically shifting **v** n-i places to the left.

The algebraic properties of a cyclic code are developed by treating the components of a code vector $\mathbf{v}=(v_0,v_1,...,v_{n-1})$ as the coefficients of a polynomial of degree

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

The correspondence between the vector **v** and the polynomial $\mathbf{v}(X)$ is one-to-one, and $\mathbf{v}(X)$ is called the code polynomial of **v**. The polynomial representation of the code vector $\mathbf{v}^{(i)}$ is

$$\mathrm{v}^{(i)}(X)=v_{n-i}+v_{n-i+1}X+\cdots+v_{n-1}X^{i-1} +v_0X^i+v_1X^{i+1}+\cdots+v_{n-i-1}X^{n-1}.$$

It can be shown that $\mathbf{v}^{(i)}(X)$ is the remainder resulting from dividing $X^i\mathbf{v}(X)$ by X^{n+1}, that is

$$X^i\mathbf{v}(X)= \mathbf{q}(X)(X^n+1)+\mathbf{v}^{(i)}(X), \qquad (2.26)$$

where $\mathbf{q}(X)=v_{n-i}+v_{n-i+1}X+\cdots+v_{n-1}X^{i-1}$. This shows that the code polynomial $\mathbf{v}^{(i)}(X)$ is the remainder resulting from dividing the polynomial $X^i\mathbf{v}(X)$ by X^n+1.

There exists a number of algebraic properties of a cyclic code which make the simple implementation of encoding and syndrome computation:

1. The nonzero code polynomial of minimum degree in a cyclic code C is unique.
2. Let $\mathbf{g}(X)=g_0+g_1X+\cdots+g_{r-1}X^{r-1}+X^r$ be the nonzero code polynomial of minimum degree in an (n,k) cyclic code C, then the constant g_0 must be equal to 1.
3. Let $\mathbf{g}(X)=1+g_1X+\cdots+g_{r-1}X^{r-1}+X^r$ be the nonzero code polynomial of minimum degree in and (n,k) cyclic code C. A binary polynomial of degree n-1 or less is a code polynomial if and only if it is a multiple of $\mathbf{g}(X)$.
4. In an (n,k) cyclic code, there exists one and only one code polynomial of degree n-k.
5. The generator polynomial $\mathbf{g}(X)$ of an (n,k) cyclic code is a factor of X^{n+1}.
6. If $\mathbf{g}(X)$ is a polynomial of degree n-k and is a factor of X^{n+1}, then $\mathbf{g}(X)$ generates an

(n,k) cyclic code.

The second property simply states that the nonzero code polynomial of minimum degree in an (n,k) cyclic code C is of the form:

$$\mathbf{g}(X)=1+g_1X+g_2X^2+\cdots+g_{r-1}X^{r-1}+X^r. \tag{2.27}$$

Consider the polynomials $X\mathbf{g}(X),X^2\mathbf{g}(X),\ldots,X^{n-r-1}\mathbf{g}(X)$, which have degrees $r+1,r+2,\ldots,n-1$, respectively. It follows from Eqn.(2.26) that $X\mathbf{g}(X) = \mathbf{g}^{(1)}(X)$, $X^2\mathbf{g}(X) = \mathbf{g}^{(2)}(X),\ldots,X^{n-r-1}\mathbf{g}(X)=-\mathbf{g}^{(n-r-1)}(X)$, that is they are cyclic shifts of the code polynomial $\mathbf{g}(X)$, and, therefore, they are code polynomials in C. Since C is linear, a linear combination of $\mathbf{g}(X),X\mathbf{g}(X),\ldots,X^{n-r-1}\mathbf{g}(X)$,

$$\begin{aligned}\mathbf{v}(X) &= u_0\mathbf{g}(X)+u_1X\mathbf{g}(X)+\cdots+u_{n-r-1}X^{n-r-1}\mathbf{g}(X)\\ &= (u_0+u_1X+\cdots+u_{n-r-1}X^{n-r-1})\mathbf{g}(X),\end{aligned} \tag{2.28}$$

is also a code polynomial where $u_i=0$ or 1. The third property characterizes this important relationship of a cyclic code. To see what property 4 states; we know that the number of binary polynomials of degree n-1 or less that are multiples of $\mathbf{g}(X)$ is 2^{n-r}. It follows from property 3 that these polynomials form all the (n,k) cyclic code C. Since there are 2^k code polynomials in C, then 2^{n-r} must be equal to 2^k. As a result, $r=n-k$(i.e., the degree of $\mathbf{g}(X)$ is $n-r$). Hence, the nonzero code polynomial of minimum degree in an (n,k) cyclic code is of the form

$$\mathbf{g}(X)=1+g_1X+g_2X^2+\cdots+g_{n-k-1}X^{n-k-1}+X^{n-k}. \tag{2.29}$$

Consequently, every code polynomial $\mathbf{v}(X)$ in an (n,k) cyclic code can be expressed in the form

$$\begin{aligned}\mathbf{v}(X) &= \mathbf{u}(X)\mathbf{g}(X)\\ &= (u_0+u_1X+\cdots+u_{k-1}X^{k-1})\mathbf{g}(X)\end{aligned} \tag{2.30}$$

If the coefficients of $\mathbf{u}(X)$, $u_0,u_1,\ldots,u_{k-1}$, are the k information digits to be encoded, $\mathbf{v}(X)$ is the corresponding code polynomial. Hence, the encoding can be achieved by multiplying the

message $\mathbf{u}(X)$ by $\mathbf{g}(X)$. Therefore, an (n,k) cyclic code is completely specified by its nonzero code polynomial of minimum degree, $\mathbf{g}(X)$. The polynomial $\mathbf{g}(X)$ is called the *generator polynomial* of the code. The degree of $\mathbf{g}(X)$ is equal to the number of parity-check digits of the code.

Property 6 states that any factor of X^n+1 with degree $n-k$ generates an (n,k) cyclic code, the code can be put in systematic form(i.e., the rightmost k-digits of each code vector are the unaltered information digits and the leftmost $n-k$ digits are parity-check digits). If the message to be encoded is $\mathbf{u}=(u_0,u_1,...,u_{k-1})$, then the code vector in systematic form would be

$$(b_0,b_1,...,b_{n-k-1},u_0,u_1,...,u_{k-1}).$$

The $n-k$ parity-check digits are simply the coefficients of the remainder resulting from dividing the message polynomial $X^{n-k}\mathbf{u}(X)$ by the generator polynomial $\mathbf{g}(X)$. In connection to cyclic codes, the first $n-k$ symbols, the coefficients of $1,X,...,X^{n-k-1}$, are taken as parity-check digits and the last k symbols, the coefficients of $X^{n-k},X^{n-k+1},...,X^{n-1}$ are taken as the information digits. In general encoding in systematic form consists of three steps:

1. Premultiply the message $\mathbf{u}(X)$ by X^{n-k}
2. Obtain the remainder $\mathbf{b}(X)$ (the parity-check digits) from dividing $X^{n-k}\mathbf{u}(X)$ by the generator polynomial $\mathbf{g}(X)$.
3. Combine $\mathbf{b}(X)$ and $X^{n-k}\mathbf{u}(X)$ to obtain the code polynomial $\mathbf{b}(X)+X^{n-k}\mathbf{u}(X)$.

2.3.3. Generator and Parity-Check Matrices of Cyclic codes

Let C be an (n,k) cyclic code with generator polynomial $\mathbf{g}(X)=g_0+g_1X+...+g_{n-k}X^{n-k}$. Since the k code polynomials generate the code C, the corresponding k n-tuples can be used as rows of an kxn matrix to form the generator matrix for C as:

where $g_0=g_{n-k}=1$. In general, **G** is not in systematic form. However it can be put in systematic form with elementary row operations.

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k} \end{bmatrix} \qquad (2.31)$$

The generator polynomial $\mathbf{g}(X)$ is a factor of X^n+1,i.e.,

$$X^n+1 = \mathbf{g}(X)\mathbf{h}(X) \qquad (2.32)$$

where the polynomial $\mathbf{h}(X)$ has degree k and is of the form

$$\mathbf{h}(X) = h_0 + h_1X + \cdots + h_kX^k$$

with $h_0=h_k=1$. The parity-check matrix can be constructed from the polynomial $\mathbf{h}(X)$. Since the reciprocal of $\mathbf{h}(X)$, which is defined as

$$X^k\mathbf{h}(X^{-1}) = h_k + h_{k-1}X + h_{k-2}X^2 + \cdots + h_0X^k, \qquad (2.33)$$

is also a factor of X^n-1 it generates an $(n,n-k)$ cyclic code with the following $(n-k)$xn matrix

$$H = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & 0 & 0 & \cdots \\ 0 & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & 0 & \cdots \\ 0 & 0 & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & \cdots \\ \vdots & & & & & & & & \\ 0 & 0 & \cdots & 0 & h_k & h_{k-1} & h_{k-2} & \cdots & \cdots h_0 \end{bmatrix} \qquad (2.34)$$

as the generator matrix. Since any code vector **v** in C is orthogonal to every row of **H**, H is a parity-check matrix of the cyclic code of C, and the row space of **H** is the dual code of C.

To form the generator matrix in systematic form X^{n-k+i} is divided by the generator polynomial $\mathbf{g}(X)$ for i=0,1,...,k-1 and as

$$X^{n-k+i} = \mathbf{a}_i(X)\mathbf{g}(X) + \mathbf{b}_i(X) \tag{2.35}$$

where $\mathbf{b}_i(X)$ is the remainder with the form,

$$\mathbf{b}_i(X) = b_{i0} + b_{i1}X + \cdots + b_{i,n-k-1}X^{n-k-1}.$$

Since $\mathbf{b}_i(X) + X^{n-k+i}$ for i=0,1,...,k-1 are multiples of $\mathbf{g}(X)$, they are code polynomials. Arranging these k code polynomials as rows of a kxn matrix results in the generator matrix of C in systematic form as shown below,

$$\boldsymbol{G} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \cdots & b_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ b_{10} & b_{11} & b_{12} & \cdots & b_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ b_{20} & b_{21} & b_{22} & \cdots & b_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & & & & & & \vdots \\ b_{k-1,0} & b_{k-1,1} & b_{k-1,2} & \cdots & b_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2.36}$$

The corresponding parity-check matrix for C is

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & b_{00} & b_{10} & b_{20} & \cdots & b_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & b_{01} & b_{11} & b_{21} & \cdots & b_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & b_{02} & b_{12} & b_{22} & \cdots & b_{k-1,2} \\ \vdots & & & & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & b_{0,n-k-1} & b_{1,n-k-1} & b_{2,n-k-1} & \cdots & b_{k-1,n-k-1} \end{bmatrix} \tag{2.37}$$

2.3.4. Encoding Cyclic Codes Using Shift Registers

The encoding of an (n,k) cyclic code in systematic form described in Section 2.3.2 involve the division of $X^{n-k}\mathbf{u}(X)$ by the generator polynomial $\mathbf{g}(X)$ to obtain the parity-check digits. This division can be accomplished with a division circuit which is a linear (n,k)-stage shift register with feedback connections based on the generator polynomial

$\mathbf{g}(X) = 1+g_1X + g_2 X^2+\cdots+g_{n-k-1} X^2 + X_{n-k}$ as shown in Fig.2.3. The encoding operation

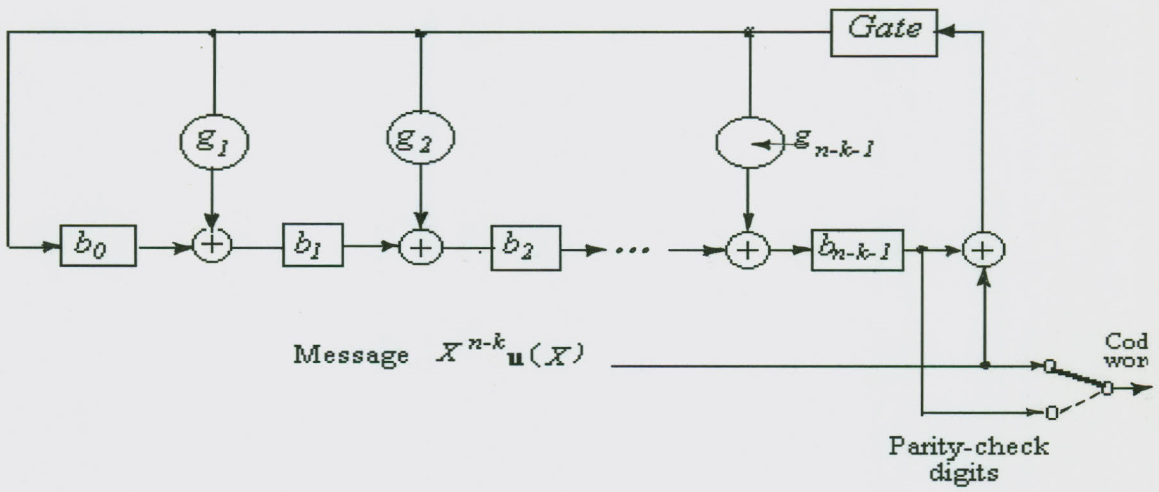


Figure 2.3 Encoder circuit for an (n,k) cyclic code with generator polynomial $g(X)=1+g_1X+g_2X^2+\cdots+g_{n-k-1}\ x^{n-k-1}+x^{n-k}$

consists of the following steps:

1. With the gate turned on the k information digits $u_0, u_1, u_2, \ldots, u_{k-1}$ are shifted into the circuit and the com munication channel simultaneously. As soon as the k information digits have entered the circuit, the register contains the *n-k* parity-check digits.
2. With the gate turned off , the contents of the shift register are shifted into the channel. These *n-k* parity-check digits, together with the k information digits,

form a complete code vector.

2.3.5. Decoding Binary Cyclic Codes

Decoding of cyclic codes consists of the same steps as decoding linear codes: syndrome computation, association of the syndrome to an error pattern, and error correction. Syndrome computation for cyclic codes can be performed with a division circuit. The cyclic structure of a cyclic code permits to decode a received vector serially, i.e., the received digits decoded one at a time and with the same circuitry. The error correction step is simply adding (modulo-2) the error pattern to the received vector using a single EXCLUSIVE-OR gate.

2.3.5.1. Syndrome Computation, Error Detection, and Error Correction

The first step of the decoding procedure involves re-encoding the received information digits to obtain a new parity sequence. The modulo-2 difference between these parity sequence and the original parity sequence is the syndrome. If no errors have occurred, the parity bits computed will be identical to those actually received, and the syndrome bits will be zero. If the syndrome bits are not zero, errors have been detected.

The syndrome for a linear systematic code (n,k) the syndrome is the remainder resulting from dividing the received vector polynomial $\mathbf{r}(X)$ by the generator polynomial $\mathbf{g}(X)$ as:

$$\mathbf{r}(X) = a(X)\mathbf{g}(X) + s(X) \qquad (2.38)$$

where $s(X)$ is of degree n-k-1 or less. The n-k coefficients of $s(X)$ form the syndrome vector.

Because of the cyclic structure of the code, the syndrome $s(X)$ has the property that the remainder $\mathbf{s}^{(i)}(X)$ resulting from dividing $X^i\mathbf{s}(X)$ by the generator polynomial $\mathbf{g}(X)$ is the

syndrome of $^{r(i)}(X)$, which is the *i*th cyclic shift of r(X). This is a useful property in decoding of cyclic codes. A division circuit for the computation of syndrome is shown in Fig. ., which is

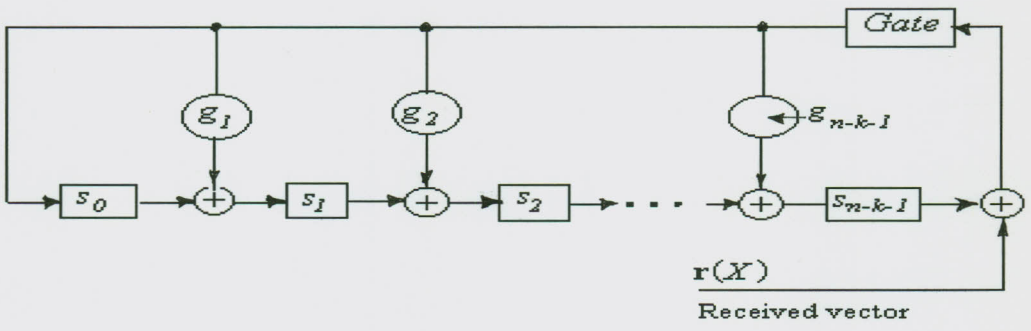


Figure 2.4 An (n,k)-stage syndrome circuit with input from the right end.

identical to the (n-k)-stage encoding circuit:

Shifting the received vector **r**(X) from the right end is equivalent to premultiplying **r**(X) by X^{n-k}. Therefore at the end of shifting the entire **r**(X) the contents in the register form the syndrome $s^{(n-k)}(X)$ of $\mathbf{r}^{(n-k)}(X)$, which is the (n-k)th cyclic shift of **r**(X). $s^{(n-k)}(X)$ is the remainder resulting from dividing $\mathbf{r}^{(n-k)}(X)$ by **g**(X).

The received polynomial can be put as the sum of the transmitted code word and the error pattern as:

$$\mathbf{r}(X) = \mathbf{v}(X) + e(X) \qquad (2.39)$$

Since **v**(x) is a multiple of the generator polynomial **g**(X), Eqn. (2.38) and Eqn.(2.39) can be combined to derive the relationship between the error pattern and the syndrome as:

$$e(X) = [a(X) + b(X)]\mathbf{g}(X) + s(X) \qquad (2,40)$$

where $b(X)\mathbf{g}(X)=\mathbf{v}(X)$.This shows that the syndrome is actually the remainder resulting from dividing the error pattern by the generator polynomial. From Eqn. (2.40) it follows that $s(X)$ becomes zero if and only if the error pattern $e(X)=0$ or it is identical to a code vector, a case in which the error pattern becomes undetectable. If the syndrome is not zero the error detection circuit can be constructed with an OR gate with the syndrome digits as inputs, whose output is "1" when $s(X)\neq 0$ i.e., when the presence of errors has been detected.

For error correction the syndrome is processed further. Since the syndrome contains information about the error pattern the decoder has to estimate $e(X)$ based on the syndrome $s(X)$. If the error pattern is a co-set leader in the standard array and table-lookup decoding is used, the error pattern can be correctly determined from the syndrome and added modulo-2 to the received vector to determine the transmitted code vector.

As soon as the syndrome has been computed, the decoding circuit checks whether the syndrome corresponds to a correctable error pattern with an error at the highest order position X^{n-1}(i.e., $e_{n-1}=1$). If the syndrome does not correspond to an error pattern with en-1=1, the received polynomial (stored in a buffer register) and the syndrome register are cyclically shifted once simultaneously. But if the first received digit is an erroneous digit, the correction is carried out by taking the modulo-2 sum of e_{n-1} and r_{n-1}. The decoding circuit proceeds to decode the received digits, and whenever an error is detected and corrected its effect on the syndrome is removed. The decoding stops after a total of n shifts and if the error is a correctable error pattern the contents of the syndrome register will be zero at the end of the decoding operation.

A general decoder for an (n,k) cyclic code is shown in Fig. 3.5. It consists three major parts: a syndrome register, an error-pattern detector, and a buffer register to hold the received

vector. The decoder is called Meggitt decoder, which applies to any cyclic code depending on the error pattern detection circuit

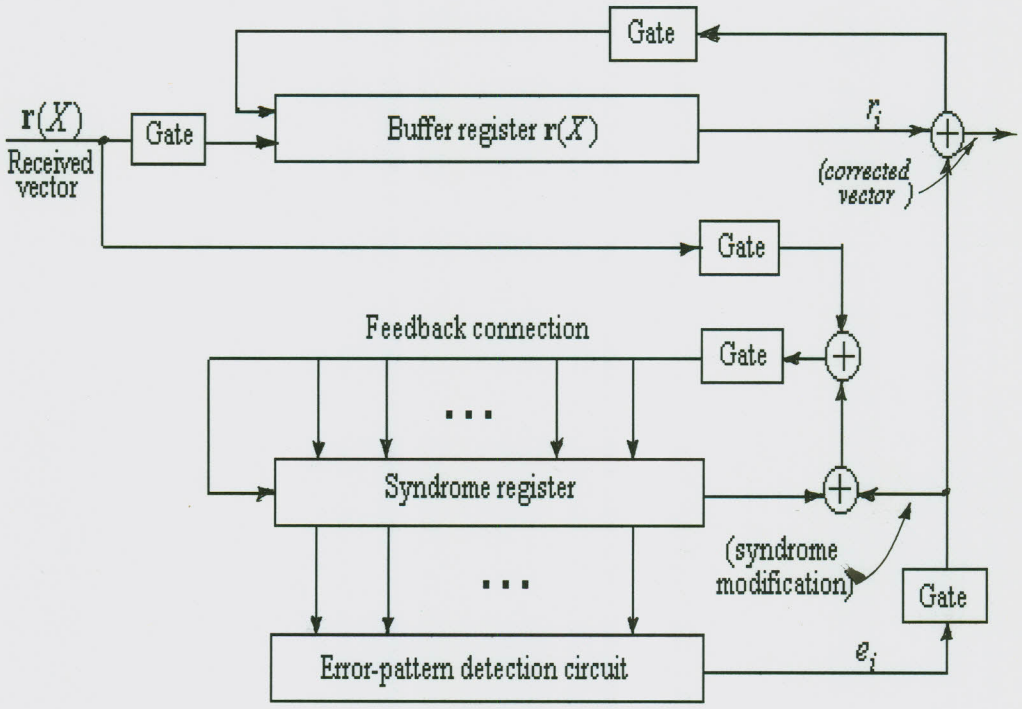


Figure 2.5 General cyclic code decoder with the received polynomial $\mathbf{r}(X)$ shifted into the syndrome register from the right end.

In Fig. 2.5 above the received polynomial $\mathbf{r}(X)$ is shifted into the syndrome register from the right end for computing the syndrome. When $\mathbf{r}(X)$ has been shifted into the syndrome register, the register contains $s^{(n-k)}(X)$, which is the syndrome of $\mathbf{r}^{(n-k)}(X)$. If $s^{(n-k)}(X)$ corresponds to an error pattern $e(X)$ with en-1=1, the highest order digit r^{n-1} is erroneous, and must be corrected. In $\mathbf{r}^{(n-k)}(X)$, the digit rn-1 is at the location X^{n-k-1}. When rn-1 is corrected, the error effect must be removed from $s^{(n-k)}(X)$. The effect of an error at the location X^{n-1} on the syndrome can be removed by feeding the error digit into the syndrome register from the right end through an *EXCLUSIVE-OR* gate.

2.3.5.2. Error Trapping Decoding

Error-trapping decoding is a practical variation of Meggit decoding which uses a very simple combinational logic circuit in the error pattern detection and correction circuit for decoding of single-error correcting codes, some short double-error-correcting codes, and burst-error-correcting cyclic codes.

To practically implement the Meggitt decoder the correctable error patterns need to be identified. For an (n,k) cyclic code with generator polynomial $\mathbf{g}(X)$, suppose a code vector $\mathbf{v}(X)$ is transmitted and is corrupted by an error pattern $\mathrm{e}(X)$. If the errors are confined to the n-k high order positions, $X^k, X^{k+1}, \ldots, X^{n-1}$ of $\mathbf{r}(X)$ [i.e., $\mathbf{e}(X)=e_k X^k+e_{k+1}X^{k+1}+\ldots+e_{n-1}X^{n-1}$] and $\mathbf{r}(X)$ is cyclically shifted n-k times, the errors will be confined to n-k low-order parity positions, $X^0, X^1, \ldots, X^{n-k-1}$ of $\mathbf{r}(X)$ with the corresponding error pattern being

$$\mathbf{e}^{(n-k)}(X)=e_k+e_{k+1}X+\ldots+e_{n-1}X^{n-k-1}.$$

since the syndrome $\mathbf{s}^{(n-k)}(X)$ of $\mathbf{r}^{(n-k)}(X)$ is equal to the remainder resulting from dividing $\mathbf{e}^{(n-k)}(X)$ by $\mathbf{g}(X)$ and since the degree of $\mathbf{e}^{(n-k)}(X)$ is less than n-k, the following equality is holds true:

$$\mathbf{s}^{(n-k)}(X)=\mathbf{e}^{(n-k)}(X)=e_k+e_{k+1}X+\ldots+e_{n-1}X^{n-k-1}.$$

Multiplying $\mathbf{s}^{(n-k)}(X)$ by X, the following results:

$$X\mathbf{s}^{(n-k)}(X) = \mathbf{e}(X)$$

$$= e_k X^k+e_{k+1}X^{k+1}+\ldots+e_{n-1}X^{n-1}.$$

That is, if the errors are confined to the n-k high-order positions of the received polynomial $\mathbf{r}(X)$, the error pattern $\mathbf{e}(X)$ is identical to $X^k\mathbf{s}^{(n-k)}(X)$, where $\mathbf{s}^{(n-k)}(X)$ is the syndrome of $\mathbf{r}^{(n-k)}(X)$,

the $(n\text{-}k)$th cyclic shift of $\mathbf{r}(X)$. When this happens simply $\mathbf{s}^{(n-k)}(X)$ is computed and $X^k\mathbf{s}^{(n-k)}(X)$ is added to $\mathbf{r}(X)$, and the resulting vector will be the transmitted code vector.

If the errors are not confined to the $(n\text{-}k)$ high-order positions but to $n\text{-}k$ consecutive positions, say $X^i, X^{i+1}, \ldots, X^{(n-k)+i-1}$, of $\mathbf{r}(X)$. If $\mathbf{r}(X)$ is cyclically shifted $n\text{-}i$ times to the right, errors will be confined to the $n\text{-}k$ low-order position of $\mathbf{r}^{(n-i)}(X)$ and the error pattern will be identical to $X^i\mathbf{s}^{(n-i)}(X)$ where $\mathbf{s}^{(n-i)}(X)$ is the syndrome of $\mathbf{r}^{(n-i)}(X)$.

Assuming the received vector $\mathbf{r}(X)$ is shifted into the syndrome register from the right end, after the entire $\mathbf{r}(X)$ has been shifted into the syndrome register the contents of the syndrome register will form the syndrome $\mathbf{s}^{(n-k)}(X)$ of $\mathbf{r}^{(n-k)}(X)$. If the errors are confined to the $n\text{-}k$ high order positions then they are identical to $\mathbf{s}^{(n-k)}(X)$. However, if the errors are confined to $n\text{-}k$ consecutive positions (including end-around) other than the $n\text{-}k$ high-order positions of $\mathbf{r}(X)$, after shifting $\mathbf{r}(X)$ into the syndrome register, the syndrome register must be shifted certain number of times before its contents are identical to the error digits.

Thus shifting of the syndrome register until its contents are identical to the error digits

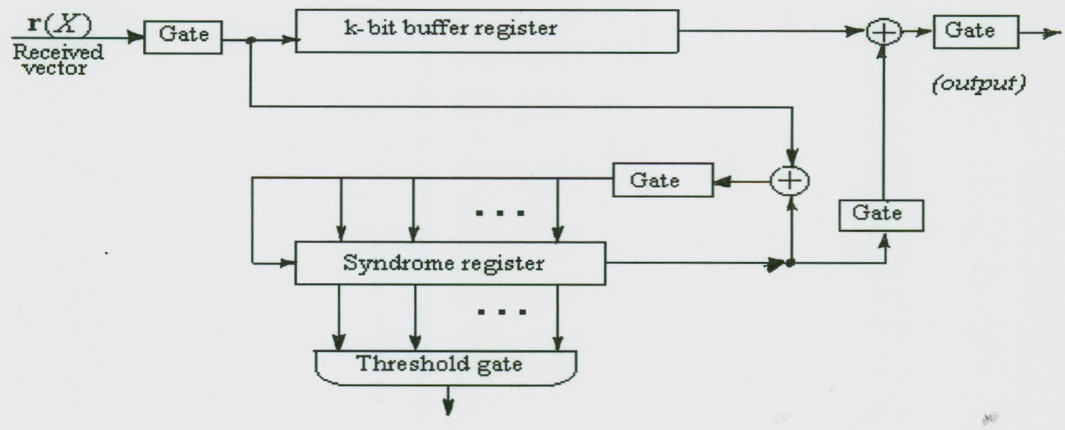


Figure 2.6 Error-trapping decoder.

is called *error trapping*. Knowing that errors are confined to the *n-k* consecutive positions of **r**(X) and detecting when the errors are trapped in the syndrome register, error correction can be accomplished by simply adding the contents of the syndrome register to the received digits at the *n-k* proper positions.

Based on the above concept an error-trapping decoder can be implemented as shown in Fig.2.6 For a *t*-error-correcting cyclic code the errors are trapped in the syndrome register only when the weight of the syndrome becomes *t* or less.

CHAPTER 3

Design of a Single Channel Communication System With Error Correction

3.1. (7,3) Maximum-Length Binary Cyclic Code

Cyclic codes are not in general limited for detection of burst errors, but they are also most effective for burst error correction and detection. An *l-burst-error-correcting* code must have atleast $2l$ parity-check digits, i.e.,

$$n\text{-}k \geq 2l.$$

(3.1) The above implies that, for a given n and k, the burst-error-correcting capability of an (n,k) code is upper bounded by $\lfloor (n\text{-}k)/2 \rfloor$. This upper bound is called the Reiger bound, and codes that meet the Reiger bound are said to be *optimal*. The *burst -correcting* efficiency of a code is measured by the ratio,

$$z = [2l/(n\text{-}k)].$$

(3.2) An optimal code has a *burst-correcting efficiency* of 1. The decoder for an *l-burst-error-correcting* cyclic code can be implemented using error trapping technique.

The (7,3) maximum-length cyclic code is an optimal code with a capability of correcting a single-random-error and a burst error of length 2. A maximum-length cyclic code, for any integer m≥3, is defined with the following parameters:

Block length: $n=2^m-1$

Number of information digits: $k=m$

Minimum distance: $d=2^{m-1}$

The generator polynomial of this code is

$$\mathbf{g}(X) = (X^n+1)/\mathbf{p}(X) \tag{3.3}$$

where $\mathbf{p}(X)$ is a primitive polynomial of degree m. This code consists of the all-zero code vector and 2^m-1 code vectors of weight 2^{m-1}. The dual code of the maximum length code is a $(2^m-1, 2^m-m-1)$cyclic code generated by the reciprocal of the parity polynomial $\mathbf{p}(X)$,

$$\mathbf{p}^*(X) = X^m\mathbf{p}(X^{-1}). \tag{3.4}$$

Since $\mathbf{p}^*(X)$ is also a primitive polynomial of degree m, the dual code is thus a Hamming code.

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Therefore, based on Eqn. (3.3) the generator polynomial of the (7,3) cyclic code can be can be derived from the primitive polynomial $\mathbf{p}(X)= 1+X+X^3$ as:

$$\mathbf{g}(X) = (X^n+1)/\mathbf{p}(X)$$

$$= 1+X^2+X^3+X^4$$

The generator and parity-check matrices for the (7,3) maximum-length code are shown in Eqn. 3.5 and Eqn. 3.6 below respectively:

Based on the generator polynomial for each 3-bit message the encoder generates the corresponding 7-tuple code word as shown in Table 3.1 below

Table 3.1 3-bit message and their corresponding code words

3-bit message	***Code word***
000	0000000
100	1011100
010	1110010
110	0101110
001	0111001
101	1100101
011	1001011
111	0010111

The decoding table for the single random error and double adjacent burst error correction is shown in Table 2.2 below:

Table 2.2 Decoding table for the (7,3) cyclic code.

Error pattern	***Syndrome***			
	S_0	S_1	S_2	S_3
$\mathbf{e}(X) = X^6$	1	0	0	1
$\mathbf{e}(X) = X^5$	1	0	1	0

Error pattern	*Syndrome*			
	S_0	S_1	S_2	S_3
$\mathbf{e}(X) = X^4$	1	1	0	0
$\mathbf{e}(X) = X^3$	1	1	0	1
$\mathbf{e}(X) = X^2$	1	1	1	1
$\mathbf{e}(X) = X$	1	0	1	1
$\mathbf{e}(X) = 1$	1	1	1	0
$\mathbf{e}(X) = X^6+X^5$	0	0	1	1
$\mathbf{e}(X) = X^5+X^4$	0	1	1	0
$\mathbf{e}(X) = X^4+X^3$	0	0	0	1
$\mathbf{e}(X) = X^3+X^2$	0	0	1	0
$\mathbf{e}(X) = X^2+X$	0	1	0	0
$\mathbf{e}(X) = X+1$	0	1	0	1
$\mathbf{e}(X) = X^6+1$	0	1	1	1

The sundrome values shown in the table are the correctable error patterns or the co-set leaders for single random error and double adjacent burst errors. The design of the decoder is based on these error patterns.

3.2. Layout of the Single Channel System

The design of the system is based on the generalized block diagram shown in Fig.3.1. The system is one-way, that is communication commences in only one direction. The system layout is shown in Fig.3.2., and the timing diagram in Fig.3.3. A buffer is used to store the information transmitted to the receiver during the signal processing activities performed by the encoder at the transmitter side and to form the digital message block received prior to its delivery to its final point. In this particular case the input buffer is used to store the output of the analog-to-digital convertor for processing by the decoder. The output buffer is used as a

temporary storage before digital-analog-conversion is performed.

3.2.1. Source Encoder and Source Decoder

The system uses time sharing an IBM PC/AT computer equipped with a PCL-812 *data aquisition board* for source encoding and source decoding. The data aquisition board is equipped with a 12-bit resolution successive approximation analog-to-digital convertor and a 12-bit monolithic-multiplying digital-to-analog convertor. A computer program was written to initialize the data aquisition board and to initiate an analoge-to-digital conversion and digital-to-analoge conversion. In addition to this, the program was used for the generation and monitoring of control signals for real-time processing of the source information.

3.2.2. Channel Encoder

The channel encoder was constructed based on the generator polynomial of the (7,3) cyclic code. It generates a code word in systematic form. In each cycle it takes a three-bit message block from the input buffer register and transmittes a 7-bit length code word, that is appending a 4 parity-check digits, and waits for a control signal (signifying the completion of the decoding process and initialized to receive the next code word) from the decoder.

3.2.3. Channel Decoder

The channel decoder was constructed based on the error-trapping decoding principle described in section, with the received message fed to the syndrome circuit from the right end. Since the code is capable of correcting a random error as well as a burst error, the error correction circuit was designed for correction of a single random error and a burst error of length 2. The circuit was constructed from *D*-type flip-flops for the received sequence buffer

register and for the computation of the syndrome digits. The error correction circuit was constructed using simple four-input NAND gate and is based on the decoding table for the (7,3) code shown in Table 2.2.

3.3. System Operation

The system operation begins by running the program for the initialization and initiation of the A/D and D/A and monitoring of the control signals between the PC and the external hardware system. The overall operation follows the following steps:

Step 1: The program initializes the data aquisition board for the software controlled A/D trigger, program controlled data transfer, and the output port of the board which is connected to the input buffer of the transmitter circuit.

Step 2: The program triggers A/D conversion, and at the completion of the conversion the A/D convertor generates a DRDY signal signifying the completion of the conversion. The program puts the data to the output port and simultaneously generates a control signal signifying that the data is readily on the output port of the data aquisition board.

Step 3: The control circuit of the transmitter generates a system reset signal, initializes the external hardware and enables the storage of the digital information into the input buffer register.

Step 4: The transmitter control circuit clocks the first three message bits into the encoder circuit and simultaneously into the communication channel. After three clock periods the control circuit disables the clocking of the input buffer and shifts the contents of the encoder shift register into the communication channel forming a code word.

Step 5: The receiver shifts the received sequence into the decoder shift register for temporary storage for processing and simultaneously into the syndrome circuit.

Step 6: The error pattern detection circuit tests the syndrome value. If the syndrome value is **s**= (1,0,0,1) then it corresponds to an error pattern of **e**= (0,0,0,0,0,0,1), a random error, and the error detection circuit generates a correction bit and corrects the the received bit at the higher order position of X^6. The storage register is cyclically shifted once to the right with the syndrome register shifted simultaneously to decode the received digit at the X^5 position. The syndrome value at the X^3 position is also corrected at the same time. If the syndrome value is **s**= (0,0,1,1) then it corresponds to an error pattern of **e**= (0,0,0,0,0,1,1), a double adjacent burst error at the position of X^5 and X^6, and the error detection circuit generates a correction bit and correctes the error at the X^6 position. The storage register is shifted cyclically shifted to the right once with the syndrome register shifted simultaneously to decode correct the error at the X^5 position. If the syndrome value does not correspond to one of them then the storage register and the syndrome register are shifted simultaneously and the syndrome values are tested for a correctable error pattern. This step continues for a period of three clock cycles to decode the first three received bits. While the received digits are clocked and corrected out of the storage register of the decoder, they are simultaneously clocked in to the output buffer of the receiver.

Step 7: After the end of the 7 clock periods the decoder generates a control signal to the Transmitter send the next code word.

Step 8: Step 4 to Step 7 are repeated until all the 12 information bits in the transmitter buffer are transmitted.

Step 9: After all the 12 information digits are transmitted and entered the receiver output buffer the decoder circuit generates a data ready signal to the PC interfaced to the external hardware.

Step 10:The program inputs the data from the digital input port of the data aquisition card and triggers the D/A convertor. The program immediately after the triggering of the D/A convertor it triggers the A/D convertor for the next sample.

The cycle continues until a key is pressed by the user to inturrept the process.

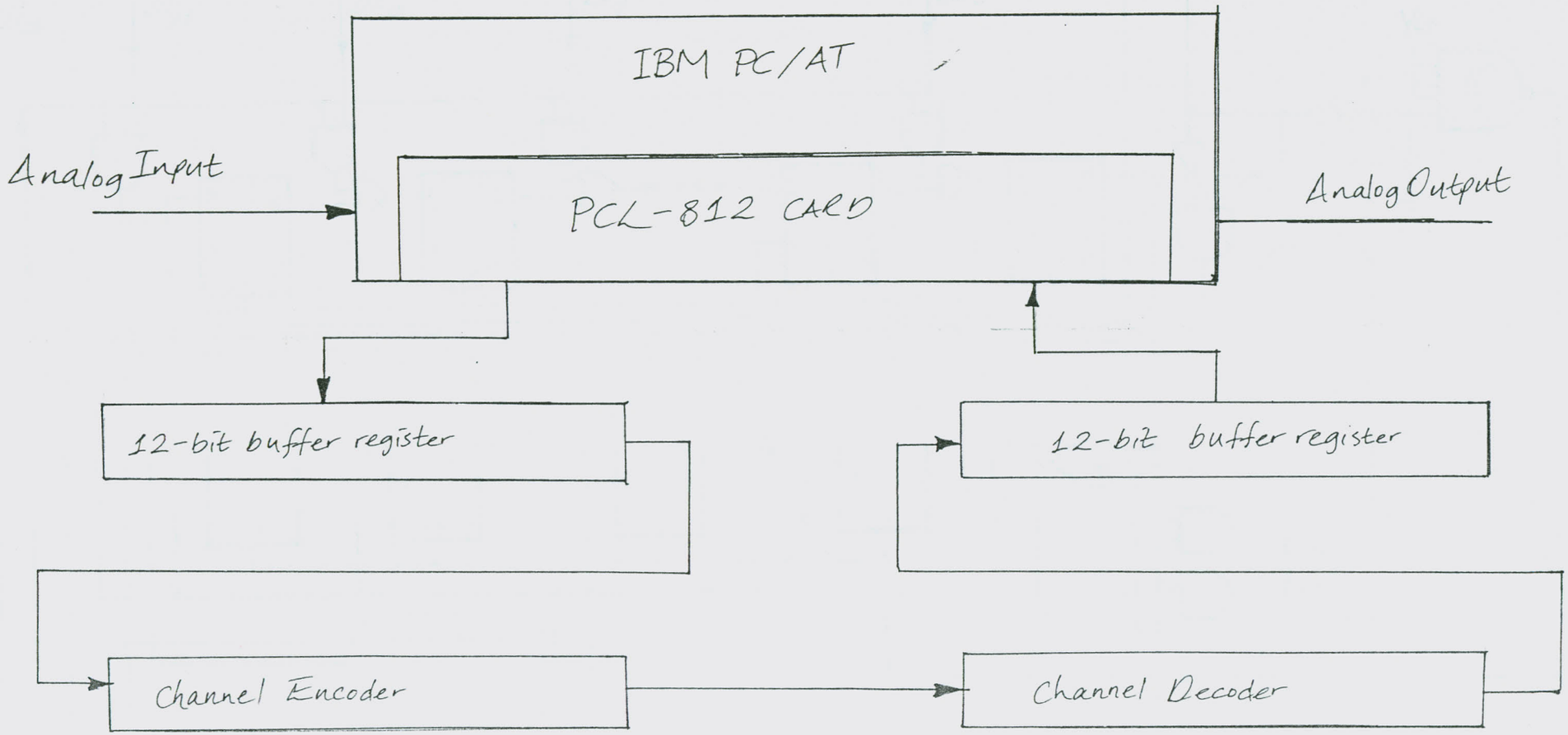


Figure 3.1 Block Diagram for Hardware Implementation of (7,3) Cyclic Code.

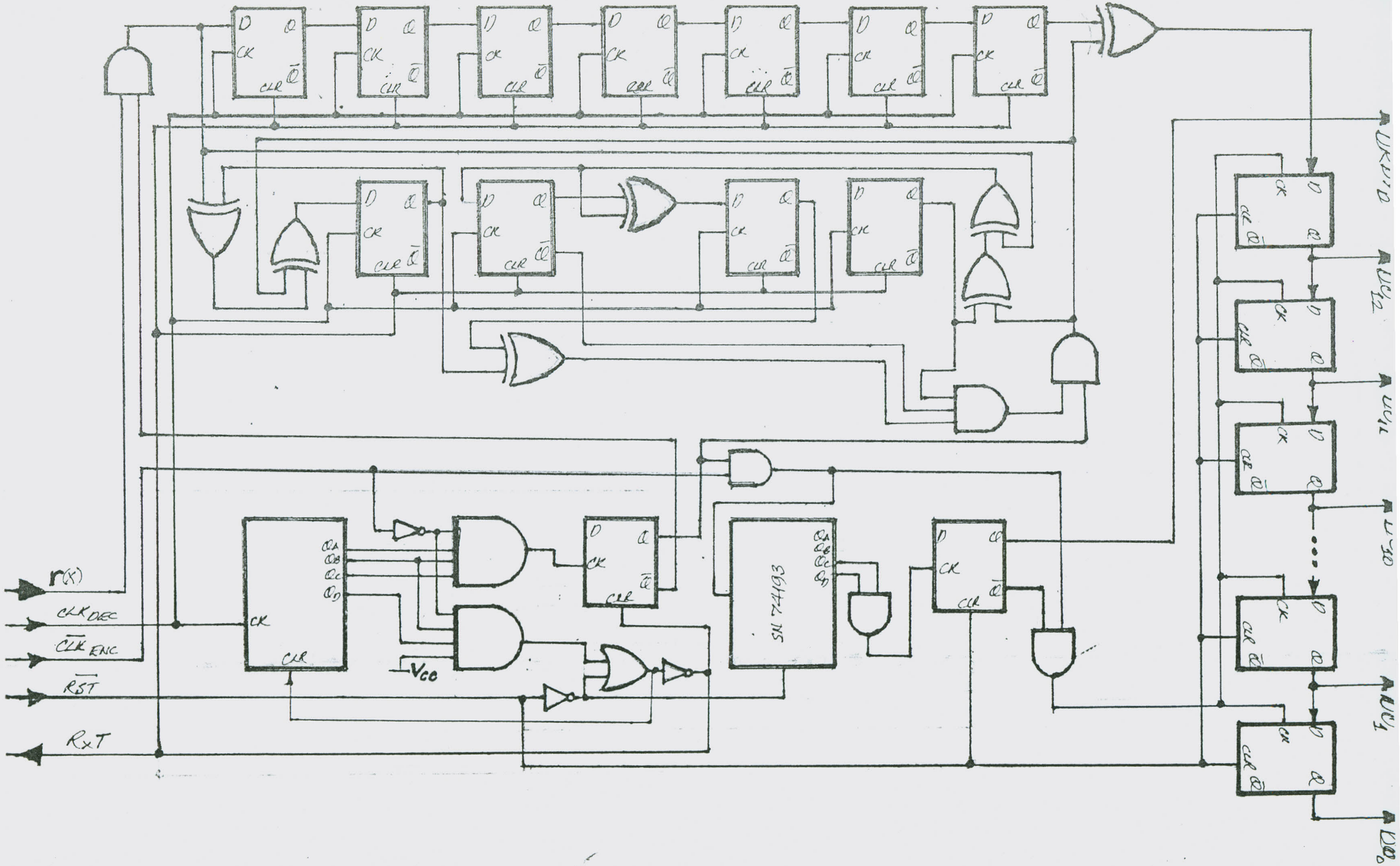


Figure 3.3 Channel Decoder for the (7,3) Cyclic Code

CHAPTER 4

Results of The Thesis Work

The results were observations of the output waveforms with correction and without correction for a sinusoidal input waveform. The error generator circuits for a single random error at the position of X^{n-1}, and for the double burst-error at the position of X^{n-1} and X^{n-2} are shown in Fig. 4.1.(a) and (b) respectively. The wave forms are for selected error-pattrns and are shown in Fig. 4.2.

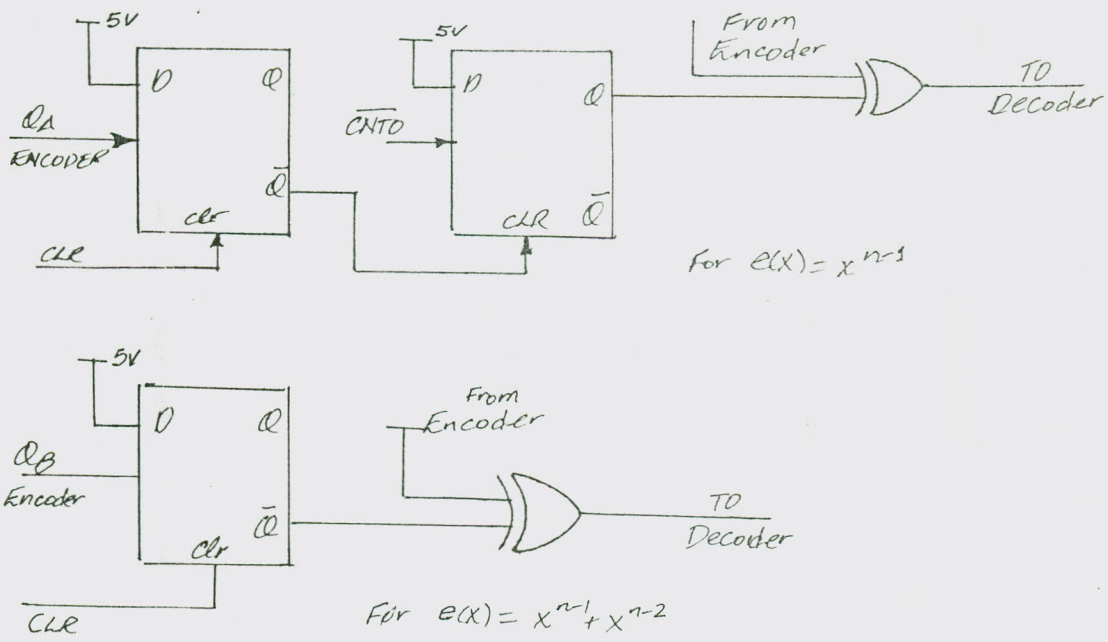


Figure 4.1 Error-generator circuits

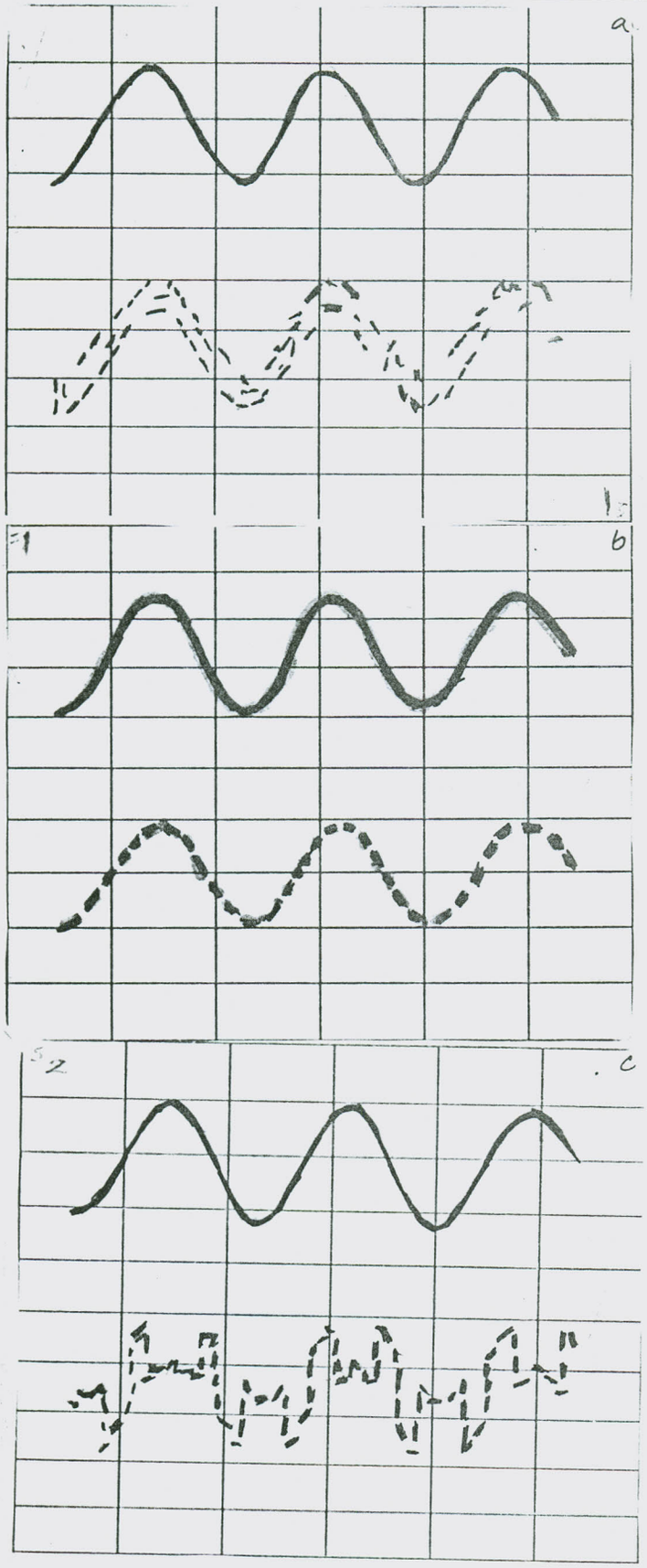


Figure 4.2 (a) uncorrected output with error at X^{n-1}; (b) corrected output with error at X^{n-1}

(c) uncorrected output with error at X^{n-2} position.

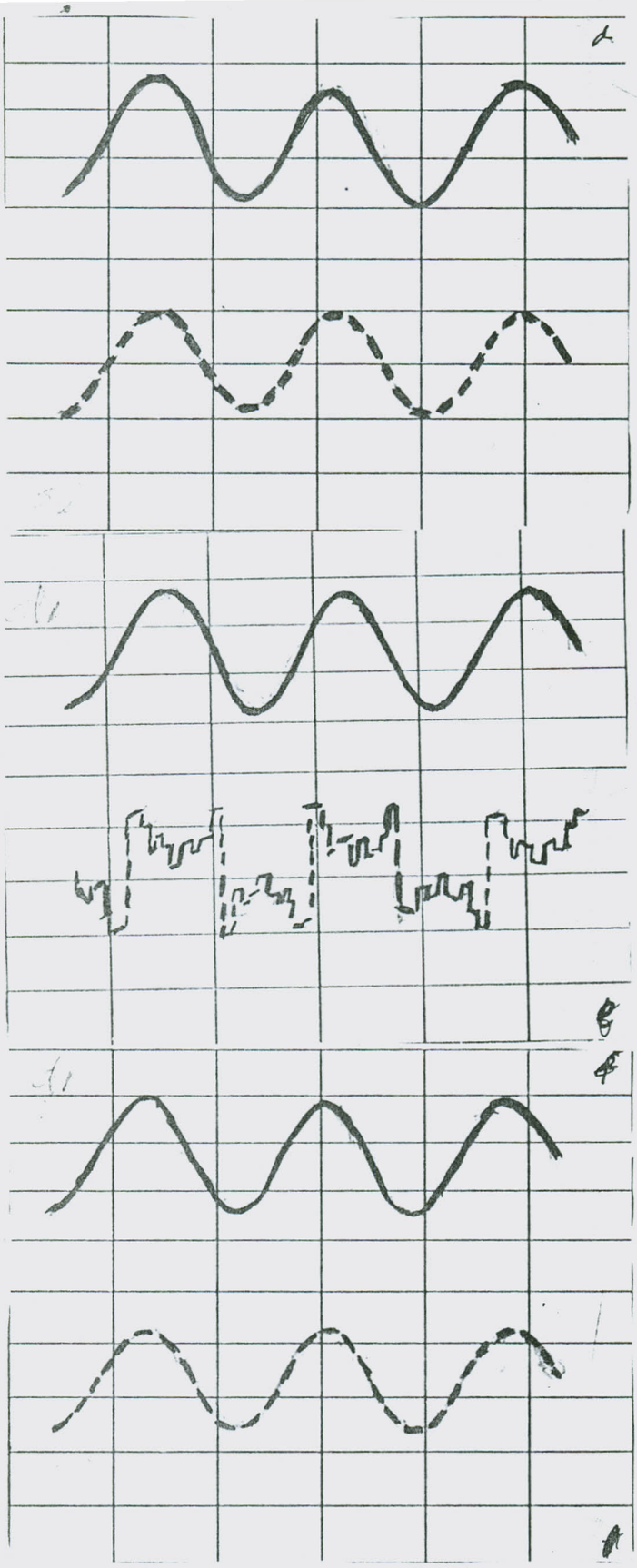


Figure 4.2 (d) corrected output with error at X^{n-2}, (e) uncorrected output error at $X^{n-1}+X^{n-2}$ (f) corrected output with error at $X^{n-1}+X^{n-2}$.

CHAPTER 5

Discussion

The results shown in Chapter 4 are sample results, i.e., samples taken for the single-random-error case and for the double-adjacent-burst-error. It was shown that the results are collected designing different error generator circuits for the different error patterns.

The hardware implementation was for a one way transmmission system and introduces a small decoding time since only the information bits are needed. The encoder and decoder are simple in construction and used flip-flops and logic gates. The performance of the system was different using error-correction and n0 error-correction, which showed that the (7,3) binary cyclic code corrects all the single-random-error patterns and double-adjacent-burst-error patterns.

The input frequency of the input signal could not be increased beyond a certain hundred of herth because of limitation on the sampling frequency. This happened since one data aquisition card was used for source encoding and source decoding to implement a real time operation of the hardware. The card's operation is also controlled by a program, and the

excution of each instruction of the program takes time introducing appreciable dely in completion of a single cycle, i.e., initiating A/D conversion, reading A/D data, writing the A/D data to the output port for processing by the external hardware, reading digital data from external hardware, writing the output data to initiate D/A conversion, and back to initiate a second A/D conversion.

In effect the sample time could not be decreased, therby prohibiting increase in input signal frequency. The PC used (an 80286 machine) was another reason which imposed a limit on the sampling frequency. Due to its low speed, particularily for this application which is time critical, it took "too much" time to execute a single instruction of the program. If it was not for the problem of the sampling frequency, the system could have performed well for speech signal as input

In applications where the system protocols require the transmission of blocks of data, such as TDMA sattelite communication system and Digital Radio Multi Access Subscriber Systems, block codes appear to be attractive. In practice, the selection of a coding scheme greatly depends on the data rate, the channel conditions, and implementation complexity. The application of VLSI technology will make it possible to realise such a coding system with few chips. In the area of mobile communication where burst error due to multipath fading are dominant the burst-error correcting cyclic code will find practical application. Mobile communication is evolving to a point where sattelite access is possible, allowing global personal communications. Since power limitations will be more strict in the future systems, error control coding will find importance in the future mobile sattelite communication systems.

APPENDIX

Pascal Program Driver for The PCL-812 Data Aquisition Board for Application to The (7,3) Maximum-Length Cyclic Code Hardware System.

```
Program    Hardware_Driver;
Uses       Crt;
var
        adhigh, adlow : byte;
        dahigh, dalow : byte;
BEGIN   { MAIN}
        port[$22B] := $01;
        port[$22A] := 0;
        While not keypressed do
            Begin
                port[$22E] :=0;            {initializing the high byte output port }
                port[$22C] := 0;           {trigger A/D conversion}
                repeat
                   adhigh := port[$225];
                until (adhigh AND $10) = 0;      { check DRDY of A/D signal}
                port[$22D] := port[$224];        {output A/D low byte}
                port[$22E] := adhigh OR $10;     { output A/D high byte}
                Repeat
                     Dahigh := port[$227];
```

```
            until ( dahigh AND $10);      {check Ready signal of the decoder}
            port[$224] := port[$226];     {input digital low byte}
            Port[$225] := dahigh;         {input digital high byte}
        End;
END.
```

BIBLIOGRAPHY

1. Lin, S. , and D.J.Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentic-Hall, Englewood Cliffs NJ, 1983.
2. Michelson, Arnold M., *Error Control Techniques for Digital Communication*, John Wiley, Massachusetts, U.S.A., 1982.
3. McEliece, Robert J, *The Theory of Information and Coding*, Addison-Wesley, Massachusetts, U.S.A., 1977.
4. Papoulis, Athanasios, *Probability, Random Varibles, and Stochastic Processes*, McGraw-Hill, Singapore, 1984.
5. *Sattelite Communication Technology*, KDD Publishing Inc., Tokyo, Japan, 1981.
6. Jakes, William C., Jr., *Microwave Mobile Communications*, John Wiley, Massachusetts, U.S.A., 1974.
7. Taub, Herbert, *Principles of Communication System*, McGraw-Hill, Singapore, 1986.
8. Shanmugam, K.Sam, *Digital and Analog Communication Systems*, John Wiley, NY, U.S.A., 1985.
9. Bellamy, John., *Digital Telephony*, John Wiley, Massachusetts, U.S.A., 1982.
10. Sanjay, K. Bose, *Digital Systems*, Wiley Eastern Ltd., India, 1986.
11. Vijay K. Bhargava, Qing Yang and David J. Peterson. Coding Theory and its Application in Communication Systems, *Defence Science Journal*, Vol 43, No 1, January 1993, pp 59-69.
12. Ian F. Blake. *A Perspective on Coding Theory*, Information Sciences **57-58,** 111-118,

(1991).

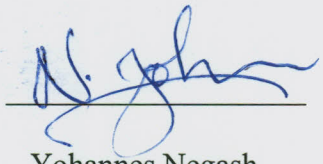
13. John T. Coffey and Rodney M. Goodman. Any Code of Which We Cannot Think Is Good. *IEEE Trans. Inform. Theory*. Vol. 36, No. 6, 1999.

14. Vijay K. Bhagava, Tho Le-Ngoc and Yousef R. Shayan. Error Control Coding for Mobile Radio System. *IEEE Proceedings*, Vol.4, 1989, pp535-537.

15. Mao-Chao Lin. Bounds on the Undetected Error Probabilities of Linear Codes for Both Error Correction and Detection. *IEEE Trans. Inform. Theory*, Vol. 36, No. 5, 1990, pp1139-1141.

DECLARATION

The thesis is my original work, has not been presented for a degree in any other university and that all sources of material used for the thesis have been duly acknowledged.



Yohannes Negash