Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ERROR DETECTION FOR DATA COMMUNICATION SYSTEMS

by

John Biancamano

Thesis submitted to the Faculty of the Graduate School
of the New Jersey Institute of Technology in partial
fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering
1983

APPROVAL SHEET

Title of Thesis: ERROR DETECTION FOR DATA COMMUNICATION SYSTEMS

Name of Candidate: John Biancamano
Master of Science in Electrical Engineering

Thesis and Abstract Approved: _____ _____
Dr. Kenneth Sohn Date
Associate Professor
Electrical Engineering

Signature of other members of the thesis committee.

_____ _____
Date

_____ _____
Date

VITA

Name: John Biancamano

Degree and date to be conferred: M.S.E.E., 1983

Secondary education: Marist High School, June 1972

Collegiate institutions attended	Dates	Degree	Date of Degree
New Jersey Institute of Technology	72-76	B.S.Eng.Sc.	May 1976
New Jersey Institute of Technology	77-83	M.S.E.E.	May 1983

Major: Electrical Engineering

Minor: Computer Science

Positions held: Development Engineer
APPLIED COLOR SYSTEMS, INC.
P.O. Box 5800
Princeton, NJ 08540

Member of Technical Staff
RCA Astro-Electronics Division
P.O. Box 800
Princeton, NJ 08540

ABSTRACT

Title of Thesis: ERROR DETECTION FOR DATA COMMUNICATION SYSTEMS

John Biancamano, Master of Science in Electrical Engineering, 1983

Thesis directed by: Associate Professor Dr. Kenneth Sohn

A description of the problems encountered in the data communications field and the various solutions can be found in a number of diverse, and often theoretical sources. My intention in writing this thesis is to bring together, in a practical and understandable manner, the theory and the application of a method of error detection used extensively in data communication systems known as the Cyclic Redundancy Check (CRC).

To provide some background on the subject, a description of a data communication system is presented, and the possible sources of error are explored in some detail. Data transmission formats are described, and a comparison of various error detection schemes is presented so that the advantages of the CRC can be more readily understood.

The theory behind the CRC and its physical implementation is given, along with a detailed example showing the effectiveness of the CRC for error detection. Finally, the current state-of-the-art technology available for implementing the various error detection schemes is discussed, with particular emphasis on those technologies that perform the Cyclic Redundancy Check.

In memory of my mother,

Carmela,

And to my father,

John,

For all their love and support.

ACKNOWLEDGEMENT

I would like to thank Dr. Kenneth Sohn for all his help and guidance, and especially for his patience in allowing me to take as much time as I needed to complete this thesis. I would also like to express my sincerest thanks to Mary Ann for all the time and effort that she put forth to type this thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER I

INTRODUCTION

Reliable data communication is an absolute must in today's world of high-speed computers. Communication between one computer and another, or between the computer and its peripheral devices, such as video terminals, line printers, and storage devices, must be as error-free as possible. However, error-free data transmission in real-world communication systems does not exist. Errors, no matter how few, will always be present.

To deal with this problem, a good number of error detection techniques have been developed. They vary in both complexity of implementation and efficiency of detecting the possible errors.

This thesis does not attempt to cover all of the various techniques available. Instead, it will present an overview of the problem and a detailed solution using one of the more popular methods of error detection known as Cyclic Redundancy Checking.

CHAPTER II

OVERVIEW OF DATA COMMUNICATION SYSTEMS

A. Model of a Data Communication System

The term "data communications" refers to the process of transmitting digitally encoded information (data) from one device to another. The data communication system, in its most general form, consists of a data source (transmitter), a communication channel, and a destination (receiver). See Figure 1.1.

The transmitter usually contains several encoders which transform the data into a form acceptable to the communication channel and consistent with a pre-determined format. One of these encoders modifies the data before transmission so as to make possible the detection of errors when the data is received. The receiver contains several decoders for putting the data back into a form that is acceptable to the user, and for determining whether or not the data has been transmitted without error.

The communication channel is the path for data transmission between the source and the receiver. This channel may consist of a single wire, or a group of wires over which the information is sent, or it may be part of the radio frequency spectrum, where the information is transmitted in a manner similar to that used for transmitting radio and television signals.

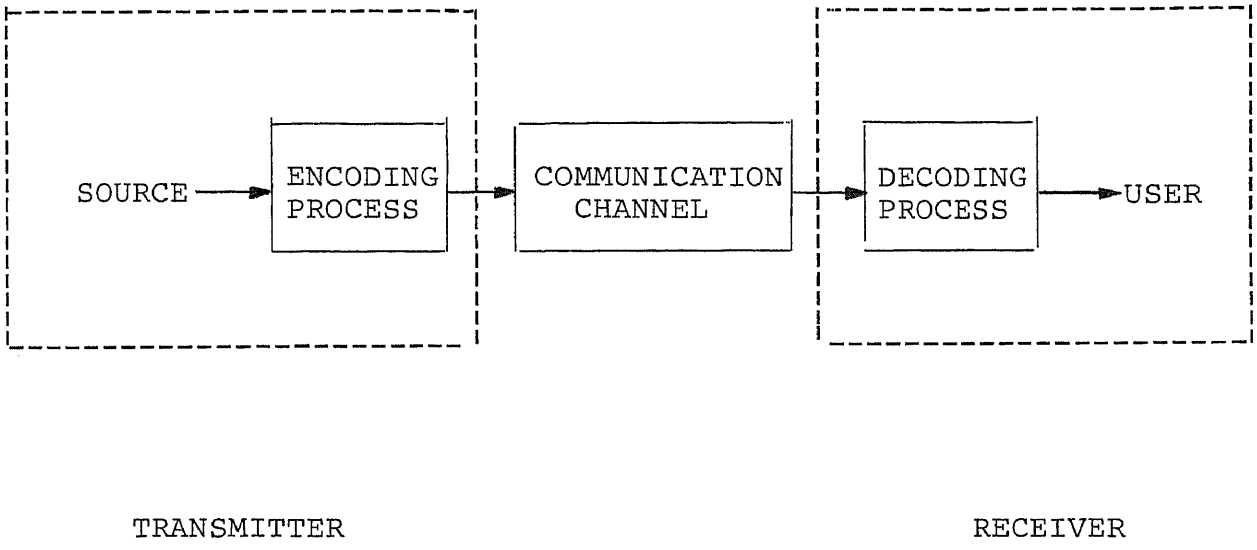


Figure 1.1 Block Diagram of a Data Communication System

Generally, a communication channel is characterized by:

1. the direction of transmission
2. the bandwidth (or capacity) of the channel medium.

There are basically three types of channels over which data can be transmitted. First, there is the simplex channel; it will allow data transmission in one direction only. An example of the use of a simplex line would be a remote instrument sending status data to a central station. The second type of channel is the half-duplex. It allows transmission in two directions, but only in one direction at a time. Half-duplex lines are sometimes used between a video terminal and a computer. The third type of channel is the full-duplex. This channel allows data transmission in both directions simultaneously. The most common example of a full-duplex channel is the telephone line.

The bandwidth of a channel will determine the maximum possible speed at which data can be transmitted. The larger the bandwidth, the higher the maximum allowable transmission speed. The capacity of a channel refers to the maximum number of data bits per second which can be transmitted. It is a function of both the channel bandwidth and the format of the data being transmitted.

A wide variety of communication channels are available to a user, and the major tradeoff in selecting one is the cost of the line versus the quality required. Naturally, it would be most advantageous to have a communication line that would allow high transmission speeds and introduce the least amount of errors into the data. However, these lines are expensive, so the user might have to settle for a line that is not of the highest quality, but one that costs less.

Telephone communication lines are available as private leased lines, as common-carrier leased lines, and through a public switching metwork. Both private and common-carrier leased lines are rented by the month and are available with or without conditioning. Conditioned lines are less susceptible to noise pick-up, a very common source of error in communication systems. Common carriers will also provide the user with some statistical information on the typical error rates and error patterns that are likely to occur on the lines they lease. Switched telephone lines, such as those used for Direct Distance Dialing (DDD), are payed for by the minute and are less costly than leased lines, but they are more susceptible to noise pick-up. They are, therefore, of poorer quality than leased lines and are not recommended for high speed data transmission.

B. Characteristics of Errors

Before proceeding with a discussion on errors, some definitions which will help clarify the remainder of this thesis are given below:

> Noise is any unwanted electrical signal that is present along with the transmitted data.
>
> Error, as used within the context of this thesis, is defined as any discrepancy between the actual value of a data bit and the theoretically correct value of that bit. An error is said to occur when a 0 is changed to a 2 or when a 1 is changed to a 0.
>
> Error rate is defined as the number of bits in error divided by the total number of bits transmitted.
>
> Error pattern refers to the number of errors occurring within a given number of data bits. That is, it specifies that x errors have occurred in a block of n data bits. It is not dependent upon the location of the errors within the block of data.

Errors generally occur for one of two reasons. First, there may be a part failure which causes the data to be changed. Second, and by far most common, errors occur because of noise pick-up on the communication channel. Communication lines run outside the computer and are exposed to much more severe environments than are the data paths found internal to the computer.

Errors do not normally occur one at a time. They are more likely to occur in bursts. That is, any noise on the communications line will most likely result in more than one bit of the received data being in error. To make the situation worse, errors are not uniformly distributed with respect to time. There may be times when a high number of errors are present, followed by periods where the error rate is quite low. This is very common when transmitting over a public switched network. On these lines, the error rate is usually higher during normal business hours than at other times of the day.

Noise spikes, or noise bursts, are present on a line for relatively long periods of time, and thus can cause serious problems if the errors are not detected. As an example, consider a burst of noise from a lightning strike. It can last for 0.01 seconds or more. During normal telephone conversations, noise of this duration might sound like a click or a crackle to the persons talking on the line, but they would still be able to understand what was said. However, if data was being

transmitted at 9600 bits per second, a noise burst of 0.01 seconds would "wipe out" 96 bits of data. Without error detection, a computer or a terminal could easily misinterpret this data as being correct, and would continue operating as if nothing was wrong.

C. Sources of Noise

Noise can be introduced into the communication systems either by faulty or improperly designed equipment, or by natural disturbances. There are many types of noise, some of which are inescapable, and some which present little or no problem to the system. Only the most severe types, the ones which are most detrimental to the communication system, are discussed here.

Impulse noise, or spikes, are the major source of errors in data communication systems. As mentioned previously, these spikes can last as long as 0.01 seconds or more. Impulse noise comes from a variety of sources including lightning flashes, telephone switching equipment, intermittent electrical connections, and voltage changes in lines or circuitry adjacent to the communication line.

Crosstalk occurs when some of the signal from one line is picked up by another line. Cross talk is most prominant in lines that run parallel to each other. The degree to which it degrades data on a line depends on certain factors. Cross talk will increase when the signal strength is increased, when the two lines are brought

closer together, and when higher frequencies are present in the signals.

Amplitude Noise is a sudden change in the level of a signal. Its effect is most noticeable on transmission systems using amplitude modulation techniques. It is generally caused by faults in the transmitting circuitry.

Line Outages are another major cause of data communication errors. They can result from mechanical failures, such as a break in the line, or from the loss of the carrier signal. Line failures can cause incomplete transmission as well as data errors.

Attenuation is the loss of a signal as it travels down the communication line. It is the result of power that is absorbed and lost from the line. As more power is absorbed by the communication line, the signal becomes weaker, and the more difficult it is for the receiver to correctly identify the data. To minimize this problem, repeaters (amplifiers) are spaced at regular intervals throughout the length of the telephone line.

Delay Distortion is evident when data is transmitted at two different frequencies. It occurs because, as data is travelling down a communication line, signals at some frequencies will be delayed more than those at other frequencies. The result is that data transmitted at one frequency will reach the receiver sooner than the data transmitted at the other frequency. An equalizer, whose

purpose is to produce a flat frequency response for transmission, will compensate for this effect.

CHAPTER III

DATA TRANSMISSION FORMATS

In all data transfers, there must be additional information sent along with the data so that the receiver can be alerted to the start of a transfer, and can determine when the end of the message has been reached. This "framing" of the data is needed for both asynchronous and synchronous transmissions. Asynchronous data transfers can take place at any time, as long as the receiver is ready to accept data. It is primarily used by slow speed terminals with transmission rates of up to 1200 bits per second.

Synchronous data transfers, as the name implies, can take place only when the transmitter and the receiver are synchronized in time. Synchronization implies that both devices are aware of the word boundaries within the message. That is, both the transmitter and the receiver can determine the beginning and the end of a word or group of words within a single transfer. Synchronous transmission is used for all high speed data transfers.

A. Data Communication Protocols

Data communication protocols are simply a set of rules that govern the transfer of information between devices. When data is transferred within a specific

computer (e.g., from memory to the CPU, or from the CPU to an I/O port), no protocols are necessary because the transfer takes place within the design constraints of the machine. However, when the data transfer is between two or more machines (devices), no longer can it be governed by internal design constraints of a single machine. It is for this reason that protocols are necessary. When two similar machines, or two completely different machines, can accept data in a format defined by a specific protocol, they can communicate accurately and reliably with one another.

While there are protocols which govern asynchronous data transfers, the major ones are generally associated with synchronous communication systems. Synchronous protocols may vary somewhat in their format details, but all can be said to perform the following functions:

1. Determine which of the two devices is the sender and which is the receiver.

2. Define the transmission format, i.e., establish which bits are for synchronization and error control, and which bits contain the actual message.

3. Define the method of encoding for error detection.

4. Define how the receiver will acknowledge a message or how, upon detection of an error, it will request for a retransmission of the message.

Figure 3.1 shows the envelope of a typical message protocol. A description of each of the block characters is given below:

Sync - there may be one or more synchronization characters to synchronize the receiver with the transmitter.

Header - contains the control information for the message, e.g., destination, priority, or message type.

Address - identification code for a specific device such that only that device will decode the message.

Message - the information (data) being transferred.

End of Message - a single character which lets the receiver know that the message has ended.

Block Check Character - one or more characters added to the message for error detection purposes.

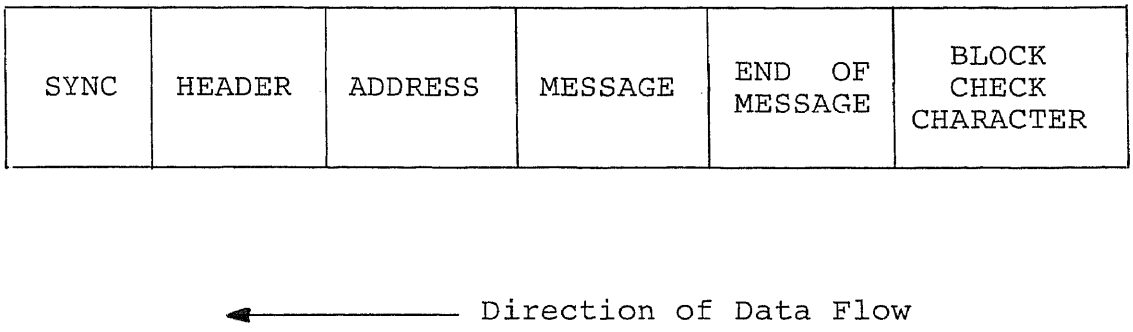


Figure 3.1. Typical Message Envelope

B. Retransmission-on-Error vs. Error Correction

When an error is detected by the receiving station, one of three things can happen. First, the error can be ignored. This is done in systems that use language text, such as for sending a telegram. If an error does occur, the correct message can generally be determined when the telegram is read. This method reduces transmission costs because no special error encoding or decoding circuitry is needed.

This is not a practical approach, however, for systems which rely heavily on their ability to transfer accurate and reliable data. The other two options, error correction and retransmission-on-error, are, therefore, the only methods used in data processing and control systems.

Both error correction and error detection with retransmission-on-error use encoding circuitry on the transmitting end of the system and decoding circuitry on the receiving end. Encoding for error detection involves adding extra bits (redundancy) to the message in order to permit the receiver to distinguish between an incorrect and a correct message. The encoding circuitry for both error detection schemes is simple and straightforward, with encoding for error correction being only slightly more complex. It is the complex

decoding circuitry for error correction which makes the scheme more difficult to implement and, therefore, less often used.

Encoding for error correction means that sufficient redundancy must be added to the message so that the receiver can not only detect the errors, but also locate their positions within the message in order to correct them. The amount of redundancy required varies with the number of errors to be detected and corrected. Some error-correcting codes require that a minimum number of error-free bits be present between bursts of errors. This is not a requirement which can always be guaranteed. The Hamming Code is an example of an error-correcting code.

Error detection with retransmission-on-error is, by far, the simplest and least expensive method for ensuring reliable data transmission. The encoding and decoding circuits for this scheme are, in some cases, identical. All of the major synchronous protocols use retransmission-on-error because it is such a straight-forward and easily implemented method for error control.

Figure 3.2 shows a flow diagram detailing a typical exchange between the transmitter and the receiver.

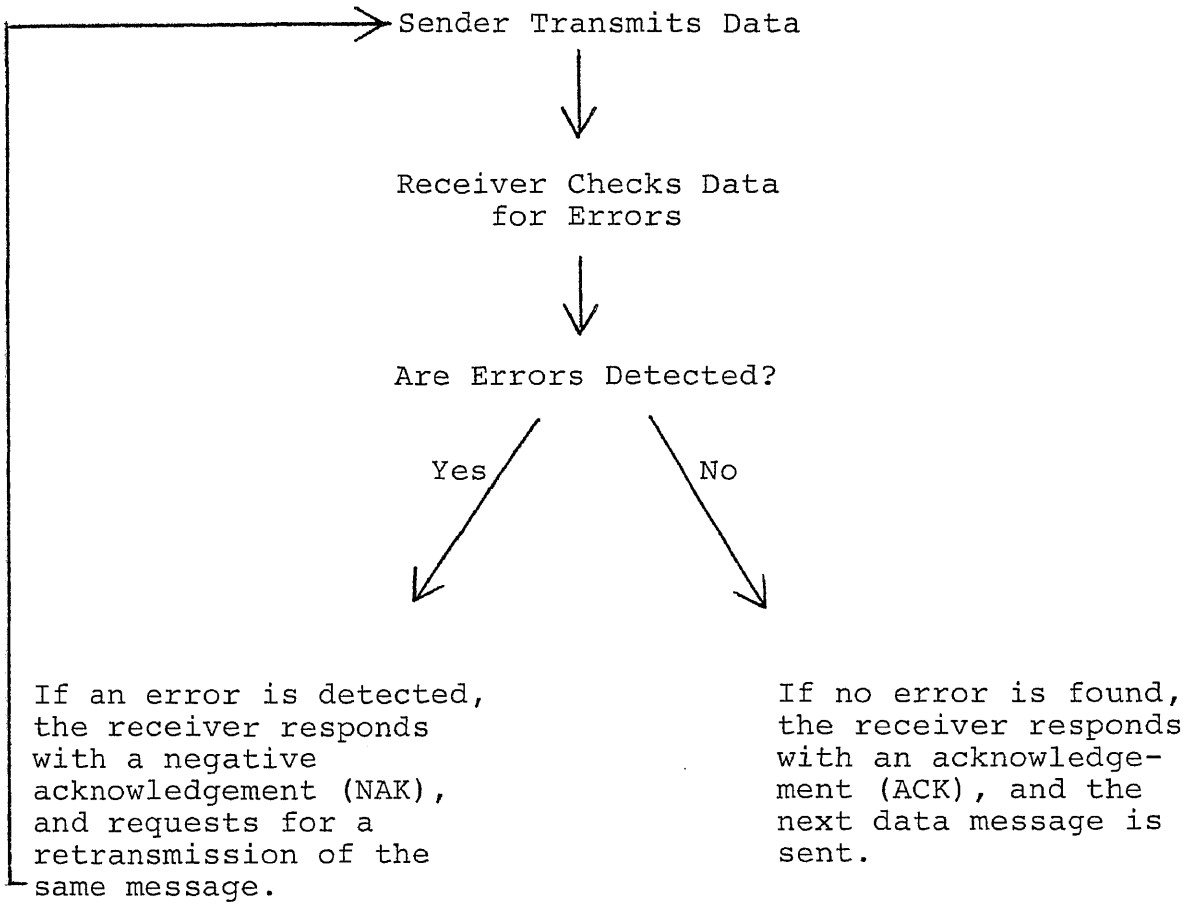


Figure 3.2 Flow Diagram For Retransmission-on-Error Format

If an error is detected, the receiver will request a retransmission of the last message until it is received without error, or until the message has been transmitted a pre-determined number of times and an error is still detected. At this point the receiver will send an error signal to the transmitter, indicating that the operation cannot continue because an error has occurred.

CHAPTER IV

ERROR DETECTION

In an practical data communication systems, there may occasionally be times when an error goes undetected. Since errors occur in bursts rather than in uniform, predictable patterns, the decision by the receiver as to whether or not an error has occurred is more of a statistical decision than an absolute one. It is really a "best guess" by the receiver on the basis of the available information, and it is not infallible. With a well designed error-detecting code, the probability of a wrong decision can be much smaller than the probability of the original message being received without error.

A. Parity

In order for the receiver to be able to detect an error, there needs to be some information sent along with the message that will tell it what constitutes a correct message. The simplest way to accomplish this is to add one extra bit to the data such that the entire data word will have an even or an odd number of 1's. This extra bit is called a parity bit. Odd parity

means that a parity bit is added to a word so that the total number of 1's in the word will be odd. For example, if the word 01101001 is to be transmitted, the number of 1's in the word is 4 (even). Therefore, the parity bit will have to be a 1, bringing the total number of 1's to 5 (odd). Now the transmitted word would be 011010011, with the parity bit added to the end of the original word. If the number of 1's in the original word was odd, then the parity bit would be 0. A similar technique exists for even parity, where the total number of 1's in the word must be even.

The parity bit is added by the encoding process in the transmitter and is checked by the decoding (or detecting) process in the receiver. While it is very easy to implement this scheme, it has a serious drawback. It can only detect one- (or three-, five-, seven-, etc.) bit errors. It will not be able to detect two- (or four-, six-, eight-, etc.) bit errors. This is true for both odd and even parity. If there was an odd number of 1's in the original word, a two-bit error would mean that there are still an odd number of 1's in the word, so it would look to the receiver as though no error had occurred.

This type of parity, where the check is performed on a single character (word), is known as Vertical Redundancy Checking (VRC). Every character in a block of data is transmitted with a parity bit. It is well suited for serial transmission and is generally used only for short data transfers.

B. Longitudinal Redundancy Check

A second type of parity checking is known as Longitudinal Redundancy Checking (LRC). Here, a special check character is transmitted at the end of a block of data such that each bit of the check character checks the parity of the corresponding bits of the message characters. For example, assume that the following message characters are to be transmitted: 1011011, 1101000, 0101001, 0011011. The check character is determined as shown below:

1011011	Message Character	1
1101000		2
0101001		3
0011011		4
1111110	Check Character	

The check character is computed such that each bit in the character will give its column an odd number of 1's.

This scheme will detect only single errors within a column. No double errors will be detected. Sometimes a combination of VRC and LRC will be used for more complete error detection. However, this is a rather inefficient method because a large number of check bits and check characters are needed when large blocks of data are being transmitted.

C. Efficiency of Error-Detection Techniques

The efficiency of an error detection scheme is defined as the number of data bits divided by the total number of bits (including data and check bits) transmitted. The more check bits that are sent along with the data, the more protection there is against an error going undetected. But as more check bits are sent, the throughput of useful data is reduced.

For a given error detection system, and a fixed number of data bits, as the number of detectable errors is increased, the efficiency of the system is decreased. There is a limit, however, to the maximum efficiency that a system can have. With short message blocks there will be less likelihood of an error and less of a need for retransmission. But short message blocks are inefficient. If longer, more efficient message blocks are used, retransmission will have to be performed

more often. There will always be this tradeoff with systems using retransmission-on-error.

D. Cyclic Redundancy Check

An error detection scheme that is more effective and more efficient than either VRC or LRC at detecting large bursts of errors is known as the Cyclic Redundancy Check (CRC). CRC is part of a family of error-detecting codes known as cyclic codes. Cyclic codes have the property that a cyclic shift of the code word, either left or right, will yield another code word. (The definition of cyclic codes is given here only for clairty,since cyclic codes, in general, will not be discussed.)

When generating the CRC character (the Block Check Character in the protocol format), the entire serial block of data (the message) is treated, not as a string of 1's and 0's, but as a binary polynomial where the 1's and 0's are the coefficients of that polynomial, M(X), known as the message polynomial. For instance, if 10110101 was the message being transmitted, it would be treated as the polynomial $(1)X^7+(0)X^6+(1)X^5+(1)X^4+(0)X^3+(1)X^2+(0)X+1$, or simply $X^7+X^5+X^4+X^2+1$. The highest power of X is attached to the most significant bit (MSB) of the message.

M(X) is then prescaled and divided, using modulo-2 arithmetic (see Appendix A), by a fixed polynomial, G(X), known as the generator polynomial. The division will yield a quotient, Q(X), and a remainder, R(X). R(X) is the CRC character and it is appended to the end of the message before transmission. The prescaling of M(X) is done to insure that the degree of M(X) is always greater than the degree of G(X), so that the remainder, R(X), is always different from the message itself.

The process is carried out as follows:

1. Multiply (prescale) M(X) by $X^{(n-k)}$,

   where: n is the total number of bits being transmitted

   k is the number of information (message) bits

   (n-k) is the number of bits of the CRC character. (It is also the degree of G(X).)

$$X^{(n-k)}M(X)=Q(X)G(X)+R(X) \qquad \text{(Eq. 4.1)}$$

2. Divide $X^{(n-k)}M(X)$ by G(X) to determine R(X).

$$\frac{X^{(n-k)}M(X)}{G(X)}=Q(X)+\frac{R(X)}{G(X)} \qquad \text{(Eq. 4.2)}$$

3. Add R(X) to $X^{(n-k)}M(X)$ to form the code message polynomial, F(X).

From Eq. 4.1: $X^{(n-k)}M(X)-R(X)=Q(X)G(X)$

In modulo-2 arithmetic, R(X)=-R(X), therefore, $X^{(n-k)}M(X)-R(X)=X^{(n-k)}M(X)+R(X)$

$F(X)=X^{(n-k)}M(X)+R(X)=Q(X)G(X)$ (Eq. 4.3)

The simple example below will illustrate the concept:

Message polynomial, $M(X)=X^3+1$ (1001)

Generator polynomial, $G(X)=X^3+X+1$ (1011)

Number of information bits, k=4

Number of CRC bits, n-k=3

Step 1. $X^{(n-k)}M(X)=X^3(X^3+1)$

$=X^6+X^3$ (1001000)

The effect of the prescaling operation is to shift M(X) to the left by (n-k) positions (3 in this case), and fill in 0's at the right.

Step 2. $\frac{X^{(n-k)}M(X)}{G(X)} = \frac{X^6+X^3}{X^3+X+1}$

$$
\begin{array}{r|l}
 & X^3+X \quad \longleftarrow \text{the quotient} \\
X^3+X+1 & X^6+X^3 \qquad\qquad Q(X), \text{ is discarded} \\
 & \underline{X^6+X^4+X^3} \\
 & \quad X^4 \\
 & \quad \underline{X^4+X^2+X} \\
\text{the remainder, } R(X) \longrightarrow & X^2+X
\end{array}
$$

$R(X)=X^2+X=110$

Step 3. Since M(X) was prescaled, appending the remainder to the end of the message is accomplished by simple addition.

$$F(X)=X^{(n-k)}M(X)+R(X)$$

$$=X^6+X^3+X^2+X$$

$$=\underbrace{1001}_{M(X)}\underbrace{110}_{R(X)}$$

F(X) is now the coded message that is transmitted. The receiver takes F(X) and divides it by the same generator polynomial, G(X). Since F(X) is exactly divisible by G(X), then, if there is no error in the transmission, the division will produce a zero remainder.

From Eq. 4.3: $F(X)=Q(X)G(X)$

therefore, $\frac{F(X)}{G(X)}=Q(X),\ R(X)=0$ (Eq. 4.4)

$$
\begin{array}{r}
 & X^3+X \\
\hline
X^3+X+1 \;\big|\; X^6+X^3+X^2+X \\
 \underline{X^6+X^4+X^3} \\
 X^4+X^2+X \\
 \underline{X^4+X^2+X} \\
 0 \quad \longleftarrow \text{remainder} = 0
\end{array}
$$

In theory, G(X) can be any polynomial. However, standard polynomials are used for implementing the CRC which have proven to be the most effective at detecting errors.

E. Capability of Error Detection Using CRC

Consider a message transmitted as 1011010001101111 and received as 1010011010001111. Errors have occurred in bit positions 4, 7, 9, 10, and 11 counting from the left. E(X), the error polynomial is given by $(X^{12}+X^9+X^7+X^6+X^5)$ or $X^5(X^7+X^4+X^2+X+1)$. The error pattern is 8 bits long. The multiplication of the error pattern by X^a results in a shifting of the pattern within the message.

The received message, P(X), can be looked at as being the error polynomial, E(X), added modulo-2 to the code message polynomial, F(X).

$$\begin{aligned} P(X) &= F(X)+E(X) \\ &= F(X)+X^aE(X) \\ &= Q(X)G(X)+X^aE(X) \end{aligned} \qquad \text{(Eq. 4.5)}$$

It can be seen from Eq. 4.5 that for an error to go undetected, E(X) must be exactly divisible by G(X). If not, the division will produce a non-zero remainder and an error will have been detected.

If the degree of E(X), taken to be m, is less than the degree of G(X), which is (n-k), then the error will have to be detected. If m=(n-k), then the error will go undetected only if E(X)=G(X). If m > (n-k), the error can go undetected only if E(X) is divisible by G(X).

The result is that any error pattern of (n-k) or fewer adjacent bits will always be detected. In addition, a large number of errors occurring in more than (n-k) bits will also be detected.

CHAPTER V

RELATIONSHIP BETWEEN CRC POLYNOMIAL MATHEMATICS AND DIGITAL SWITCHING CIRCUITS

One of the nicest features of the Cyclic Redundancy Check is the ease with which it can be implemented in a practical circuit. In this chapter, the relationship between modulo-2 multiplication and division, and digital finite-state switching circuits will be explored.

A. Basic Building Blocks

Digital switching circuits are made up of three basic elements: an adder, a storage register, and a constant multiplier. These elements are shown functionally and schematically in Figure 5.1. The first of the three is the adder, whose output is equal to the sum of the two inputs, with no carry digit generated. An exclusive-OR gate is used to perform modulo-2 addition (see Appendix A). The second element is the storage register or memory device. The output value of the storage register is equal input value delayed by one unit of time. In digital circuits, this delay is accomplished by using a flip-flop (one stage of a shift register) that is clocked by a signal from a timing or synchronizing circuit. The clock pulse represents

one unit of time. The value of the output after the clock pulse is the same as that of the input before the pulse occurred. A number of these flip-flops will be cascaded together to make a shift register which will hold the results of the mathematical operations. The third element in the switching circuit is the constant multiplier. The output of this element is equal to the input multiplied by a constant. In the case of binary numbers, this constant is either 1 or 0. Multiplication by 1 is achieved by a physical connection between the input and the output. Multiplication by 0 implies that no connection exists.

B. Polynomial Transmission Format

Since the information is being transmitted serially, only one bit can be processed at a time. When implementing the CRC, the information is treated as a polynomial with only the coefficients being transmitted. In order for the division of the message polynomial to be carried out correctly, the high-order coefficients of the dividend (the message polynomial) must be processed first. For this to take place, the information must be encoded and transmitted with the most significant bits coming first.

The polynomial

$$F(X)=F_nX^n+F_{n-1}X^{n-1}+\ldots+F_1X+F_0$$

would be processed serially with F_n being the first bit, F_{n-1} being the second bit, delayed by one clock pulse, then F_{n-2}, delayed by two clock pulses, and so forth, until F_0 is

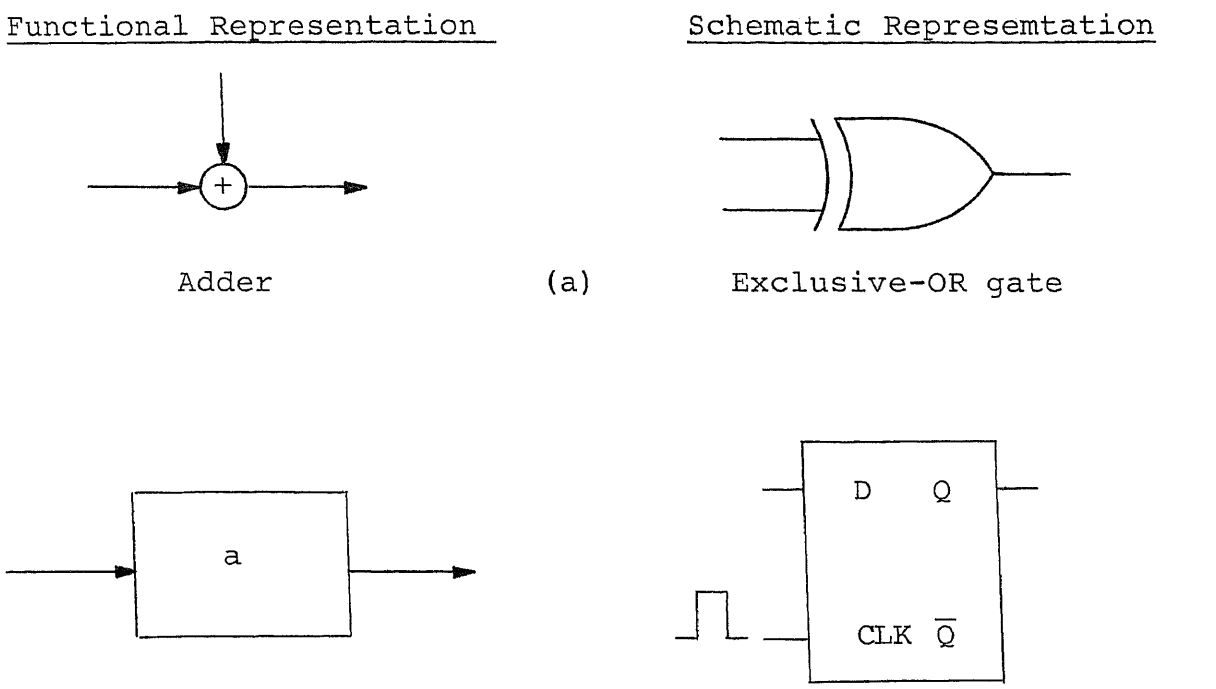


Figure 5.1 Basic Building Blocks for Digital Switching Circuits

processed, delayed by n clock pulses.

C. Multiplication of Polynomials

Given any two polynomials:

$$a(X)=a_nX^n+a_{n-1}X^{n-1}+\ldots+a_1X+a_0$$

$$b(X)=b_mX^m+b_{m-1}X^{m-1}+\ldots+b_1X+b_0$$

The product, $a(X)b(X)$, is

$$\begin{aligned}a(X)b(X)=&a_nb_mX^{n+m}+(a_{n-1}b_m+a_nb_{m-1})X^{n+m-1}\\&+(a_{n-2}b_m+a_{n-1}b_{m-1}+a_nb_{m-2})X^{n+m-2}\\&+\ldots+(a_0b_2+a_1b_1+a_2b_0)X^2\\&+(a_0b_1+a_1b_0)X+a_0b_0\end{aligned}\qquad \text{(Eq 5.1)}$$

If $a(X)$ is considered to be any input polynomial and $b(X)$ is a fixed polynomial, then the circuit in Figure 5.2 illustrates how any polynomial $a(X)$ can be multiplied by a fixed polynomial $b(X)$.

Initially, all registers are set to 0. The coefficients of each product term will be stored in the registers as they appear at the input. Besides the data inputs, all registers have a clock input which allows the coefficients to be shifted towards the output. For clarity, the clock inputs are not shown in the figure.

The first coefficient to appear at the input is a_n. The output is then equal to a_nb_m and all the registers still

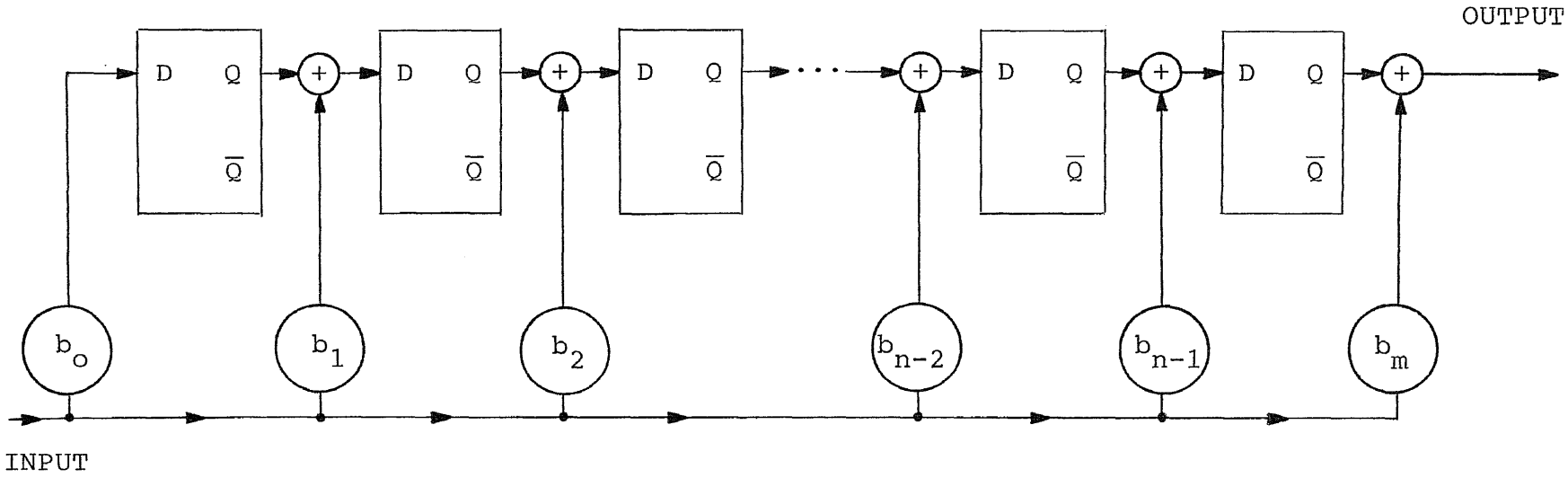


Figure 5.2 Multiplication Circuit For Polynomials

contains 0's. Notice that $a_n b_m$ is the coefficient of the first term in Eq. 5.1. After the first clock pulse, the registers contain, from left to right, $a_n b_0$, $a_n b_1$, $a_n b_2$,..., $a_n b_{m-2}$, $a_n b_{m-1}$, and the next coefficient, a_{n-1}, is at the input. The output is now $a_n b_{m-1}+a_{n-1} b_m$. This is the second coefficient in Eq. 5.1. After the second clock pulse, the registers contain $a_{n-1} b_0$, $a_n b_0+a_{n-1} b_1$, $a_n b_1+a_{n-1} b_2$,..., $a_n b_{m-2}+a_{n-1} b_{m-1}$, and the third coefficient, a_{n-2}, is at the input. The output is now $a_n b_{m-2}+a_{n-1} b_{m-1}+a_{n-2} b_m$, the third coefficient is Eq. 5.1.

The multiplication continues in this manner. After n+m-1 clock pulses, the registers contain 0,0,0...,0, $a_0 b_0$, $a_0 b_1+a_1 b_0$. Since the input is 0, the output is equal to $a_0 b_1+a_1 b_0$. This is the second to the last coefficient in Eq. 5.1. After the final clock pulse, all the registers contain 0's except the last one, which contains $a_0 b_0$. The input is still 0, so the output is $a_0 b_0$, the last coefficient in Eq. 5.1. The multiplication, after n+m clock pulses, is complete, and each of the coefficients has been shifted out of the circuit, with the higher-order coefficients coming out first.

D. Division of Polynomials

A circuit for dividing any polynomial by a fixed polynomial is shown in Figure 5.3. As with the multiplication circuit, all of the registers in the division circuit have clock inputs which are omitted from the figure for clarity.

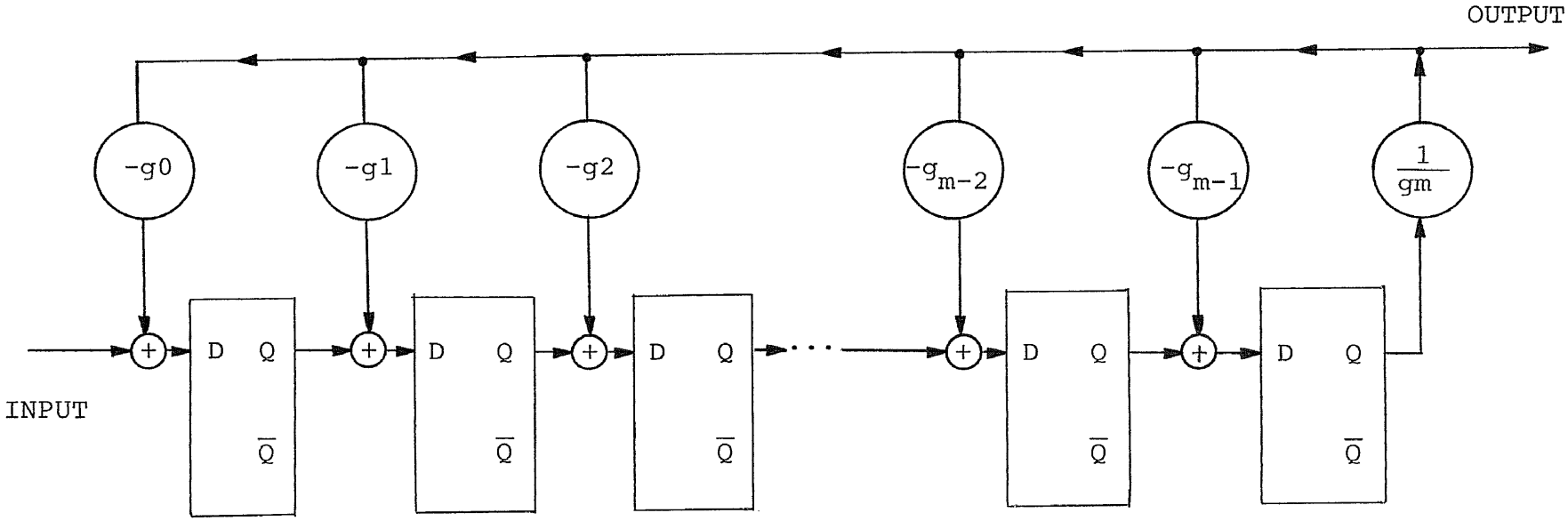


Figure 5.3 Division Circuit for Polynomials

Given:

$$f(X)=f_nX^n+f_{n-1}X^{n-1}+\ldots+f_1X+f_0$$

as the input polynomial, and

$$g(X)=g_mX^m+g_{m-1}X^{m-1}+\ldots+g_1X+g_0$$

as the fixed polynomial

Initially, each stage of the shift register is set to 0. The coefficients of the input polynomial are shifted serially into the registers on each clock pulse, with f_n being first, followed by f_{n-1}, f_{n-2}, etc. After m clock pulses, f_n is stored in the last register, and the first coefficient of the quotient, f_n/g_m, appears at the output. For every coefficient, q_k, of the quotient, the polynomial $q_kg(X)$ must be subtracted from the dividend. This is accomplished via the feedback connections to the adders. After a total of n+1 clock pulses, all the coefficients of the quotient have been shifted out of the circuit, and the remainder is left in the registers.

This operation will now be illustrated with an example, using the polynomials, $f(x) = X^6+X^3$, and, $g(X)=X^3+X+1$, from Chapter IV. The circuit for performing the division is shown in Figure 5.4. The division is performed below, and compared with the step-by-step operation of the circuit as shown in Table 5.1.

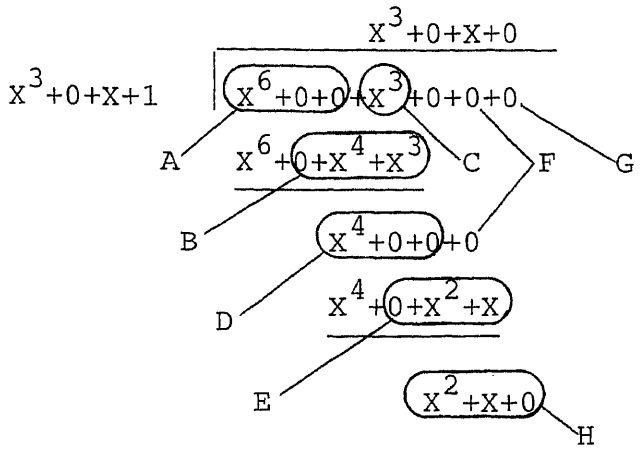


Table 5.1 Step-by-Step Operation of the Division Circuit

Number of Clock Pulse	Input Before Clock Pulse	Contents of Shift Register After The Clock Pulse	Output After The Clock Pulse
Start	-	000	0
1	1	100	0
2	0	010	0
3	0	001	1
4	1	010	0
5	0	001	1
6	0	110	0
7	0	011	0

After the first three clock pulses, the contents of the shift register contain the coefficients of the polynomial marked A in the division operation, with the last register containing the higher-order term. The leading coefficient, therefore, is the first coefficient of the quotient and the output after the third clock pulse. The feedback, although

not shown in Table 5.1, matches the coefficients of the polynomial marked B, and the next input coefficient corresponds to C, the term which is brought down during the division process. After the fifth clock pulse, the shift register contains the coefficients of the polynomial marked D and the feedback matches the coefficients of E. F is the next to the last input. G is the last input, and after the seventh clock pulse, the shift register contains the remainder, the coefficients of the polynomial marked H.

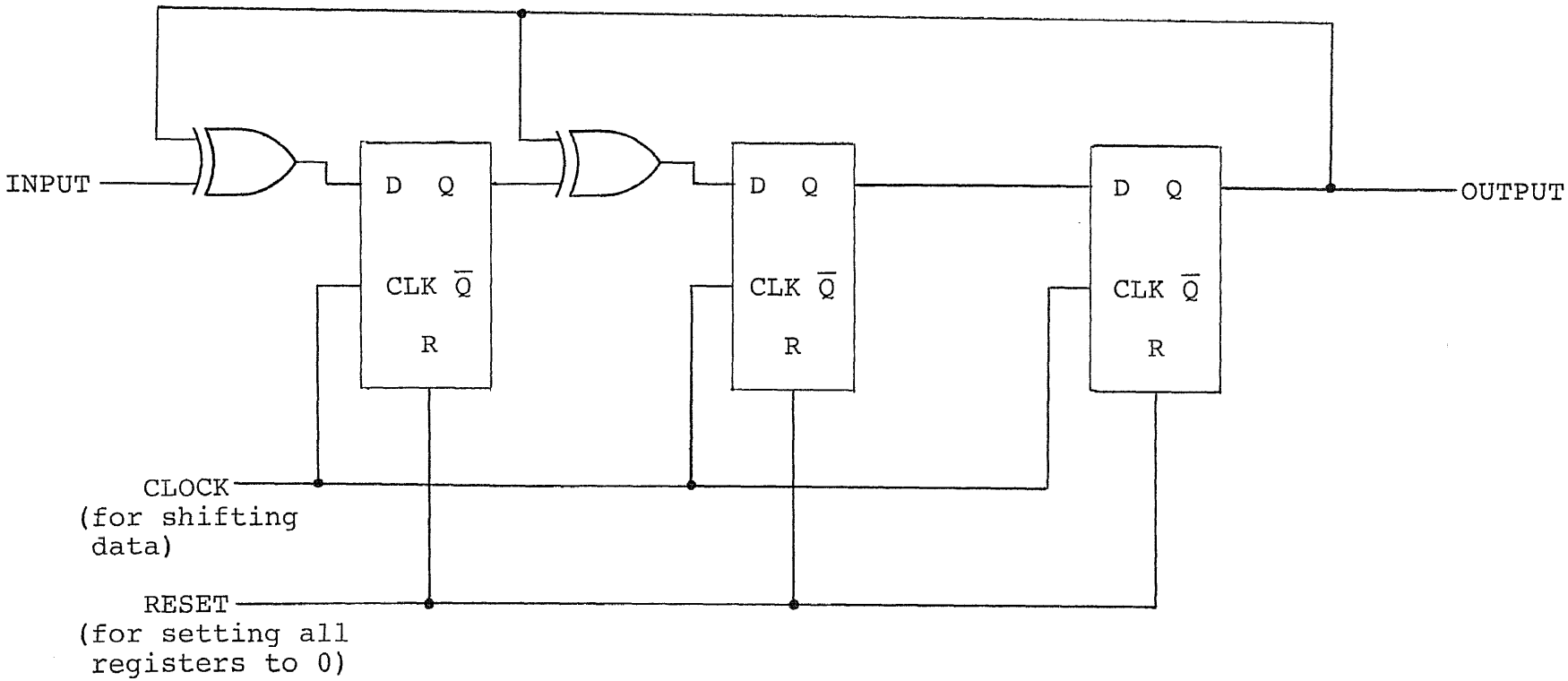


Figure 5.4. Circuit for Dividing By X^3+X+1

CHAPTER VI

DETAILED EXAMPLE OF THE USE OF CRC FOR ERROR DETECTION

If it was desired to implement an error detection scheme using the Cyclic Redundancy Check to detect all bursts of errors up to 8 bits in length, the CRC character would have to be 8 bits long. The generator polynomial, G(X), for such a scheme might be: X^8+X^4+X+1.

The input data could be any reasonable number of bits long, however, for simplicity, it will be limited here to 12 bits, and might look as follows: 101100100011 ($X^{11}+X^9+X^8+X^5+X+1$).

To encode the data, the coefficients of the polynomial $X^{11}+X^9+X^8+X^5+X+1$ would be processed through the division circuit shown in Figure 6.1. The steps for mathematically encoding the data are detailed below, and the corresponding output of the division circuit is shown in Table 6.1. The division is performed here using 1's and 0's instead of the polynomials, as was done in Chapters IV and V.

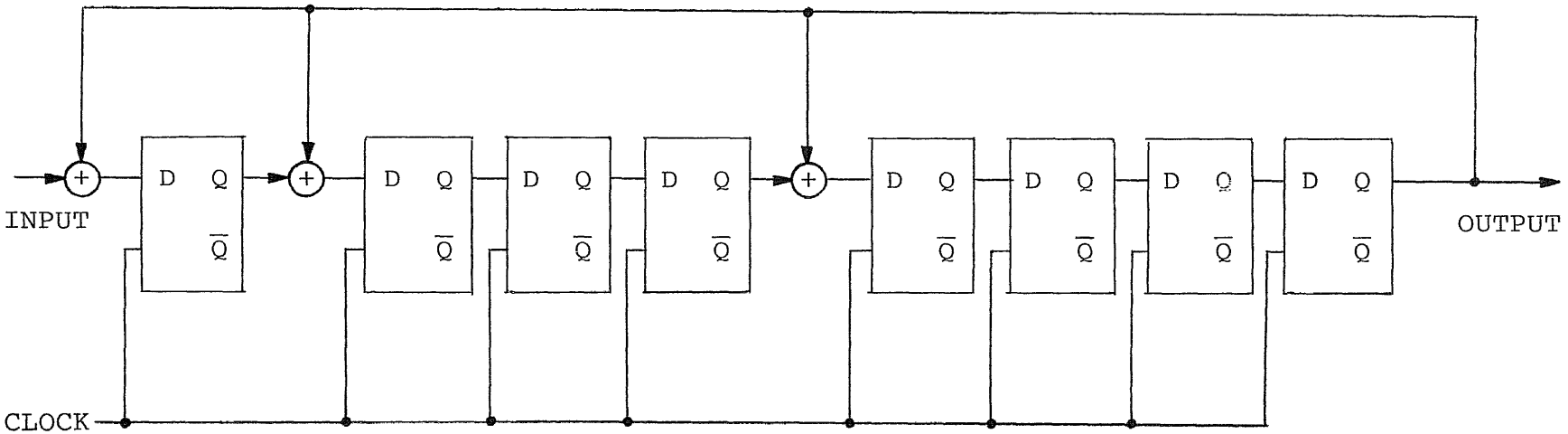


Figure 6.1. Division Circuit for Dividing by X^8+X^4+X+1

Given:

Message Polynomial, M(X)-101100100011 ($X^{11}+X^9+X^8+X^5+X+1$)

Generator Polynomial, G(X)=100010011 (X^8+X^4+X+1)

Number of information bits, k=12

Number of CRC bits, n-k=8

Step 1: $X^{n-k}M(X)=X^8(X^{11}+X^9+X^8+X^5+X+1)$

$=101100100011000000000$

(continued on next page)

Step 2: $\frac{X^{n-k}M(X)}{G(X)}$

```
                    101110000111
              ┌──────────────────
100010011 | 10110010001100000000
            100010011
              111011101
              100010011
               110011101
               100010011
                100011100
                100010011
                     111100000
                     100010011
                      111100110
                      100010011
                       111101010
                       100010011
remainder, R(X) ──────► 11111001
```

$$R(X)=X^7+X^6+X^5+X^4+X^3+1$$

Step 3: Code Message Polynomial, $F(X)=X^{n-k}M(X)+R(X)$

$$F(X)=\underbrace{10110010001}_{M(X)}\underbrace{111111001}_{R(X)}$$

F(X) is now transmitted.

Table 6.1 Division Circuit Output for Encoding of the Data

Number of Clock Pulse	Input Before Clock Pulse	Contents of Shift Register After The Clock Pulse	Output After The Clock Pulse
Start	-	00000000	0
1	1	10000000	0
2	0	01000000	0
3	1	10100000	0
4	1	11010000	0
5	0	01101000	0
6	0	00110100	0
7	1	10011010	0
8	0	01001101	1
9	0	11101110	0
10	0	01110111	1
11	1	01110011	1
12	1	01110001	1
13	0	11110000	0
14	0	01111000	0
15	0	00111100	0
16	0	00011110	0
17	0	00001111	1
18	0	11001111	1
19	0	10101111	1
20	0	10011111	1

CRC Character

At the receiving station, F(X) would be divided by G(X), using the same circuit that was used for encoding the data. If F(X) were received without error, the division would be as shown below. The corresponding circuit output is shown in Table 6.2.

$\frac{F(X)}{G(X)}$:

```
                          101110000111
                 ┌──────────────────────
100010011        │ 10110010001111111001
                   100010011
                   ─────────
                     111011101
                     100010011
                     ─────────
                      110011101
                      100010011
                      ─────────
                       100011101
                       100010011
                       ─────────
                            111011110
                            100010011
                            ─────────
                             110011010
                             100010011
                             ─────────
                              100010011
                              100010011
                              ─────────
remainder ──────────►          00000000
```

R(X)=0, therefore, no errors were detected

Table 6.2. Division Circuit Output For Message Received With No Errors

Number Of Clock Pulse	Input Before Clock Pulse	Contents Of the Shift Register After the Clock Pulse	Output After The Clock Pulse
Start	-	00000000	0
1	1	10000000	0
2	0	01000000	0
3	1	10100000	0
4	1	11010000	0
5	0	01101000	0
6	0	00110100	0
7	1	10011010	0
8	0	01001101	1
9	0	11101110	0
10	0	01110111	1
11	1	01110011	1
12	1	01110001	1
13	1	01110000	0
14	1	10111000	0
15	1	11011100	0
16	1	11101110	0
17	1	11110111	1
18	0	10110011	1
19	0	10010001	1
20	1	00000000	0

remainder

In an error occurred during transmission such that bits 2,3,4,7,and 9, counting from left to right, were changed, the code message would be received as: 11000000101111111001. This is an 8-bit burst error, E(X), and should be detected by the circuit, as shown below and in Table 6.3.

$\frac{F(X)+E(X)}{G(X)}$:

```
                    110011010011
              ┌──────────────────
100010011     │ 11000000101111111001
               100010011
                100100100
                100010011
                   110111111
                   100010011
                    101011001
                    100010011
                      100101011
                      100010011
                         111000100
                         100010011
                          110101111
                          100010011
remainder ──►              10111100
```

Since R(X) is not 0, an error has been detected, and a retransmission of the data will be requested.

Table 6.3. Division Circuit Output for Message Received With Errors

Number of Clock Pulse	Input Before Clock Pulse	Contents of the Shift Register After the Clock Pulse	Output After the Clock Pulse
Start	-	00000000	0
1	1	10000000	0
2	1	11000000	0
3	0	01100000	0
4	0	00110000	0
5	0	00011000	0
6	0	00001100	0
7	0	00000110	0
8	0	00000011	1
9	1	01001001	1
10	0	11101100	0
11	1	11110110	0
12	1	11111011	1
13	1	00110101	1
14	1	01010010	0
15	1	10101001	1
16	1	00011100	0
17	1	10001110	0
18	0	01000111	1
19	0	11101011	1
20	1	00111101	1

remainder

CHAPTER VII

STATE-OF-THE-ART TECHNOLOGY

Because of the ease with which it is implemented and its efficiency at detecting errors, CRC has become one of the most widely used error detection schemes in serial data communication systems. Standards have been established for communication systems using 6- and 8-bit characters within their data message protocols.

One of the more popular of these standards for synchronous systems is known as CRC-16. The CRC check character is 16 bits long and it is used in protocols where the data words are 8 bits long. The generator polynomial for this standard is $X^{16}+X^{15}+X^{2}+1$. It can detect all bursts of errors up to 16 bits in length and, in addition, can detect about 99% of the error bursts longer than 16 bits.

The technology available today for implementing Cyclic Redundancy Checking can be grouped into the following categories:

1. Custom and Programmable Logic Arrays

   The arrays are LSI (Large Scale Integration) chips in which the user designs and implements

a single circuit for performing a specific CRC. All of the necessary shift registers and adders are provided on a single integrated circuit, and the user just has to specify the connections.

2. Error-Checking Circuits

These IC's are designed to implement various error detection schemes, including CRC. Some can work with asynchronous, as well as synchronous communication systems. Examples of this type of chip are the Fairchild 9401 CRC Generator/Checker, and the 68653/2653 Polynomial Generator Checker manufactured by Signetics and Motorola.

3. Specialized and General Purpose Protocol Controllers

These IC's implement selectable or specialized protocols on a serial data stream. That is, they enclose the data stream with a sync signal, a header, and a CRC field before transmission, and strip away these fields when the message is received. They also perform error checking on the message. An example of a general

purpose protocol controller is the 68652/2652 Multi-protocol communications controller also manufactured by Signetics and Motorola.

4. Board-Level Protocol Controllers

This last category is the most recent of all the technologies to appear in the marketplace. They are single-board controllers for implementing communications protocols, including error detection, and for interfacing any one of a number of manufacturers' devices to a standard communications line. One such controller board is the microcomputer-based Ethernet Protocol Module manufactured by Interlan, Inc. (Ethernet is one of the most popular local area networks, providing data communication links for computers and peripherals.)

CHAPTER VIII

CONCLUSION

The intent of this thesis was to describe the problems plaguing the data communications field, and to present one of a number of possible solutions. A general overview of a data communication system was presented, describing the physical make-up of a system, along with the nature and the sources of signal contamination (Chapter II). The rules for synchronous serial transmission were laid out (Chapter III), and the various techniques available for error detecting were put forth (Chapters III and IV).

The Cyclic Redundancy Check (CRC) was chosen because it is a viable solution to the troublesome task of providing accurate and reliable data transmission. It has, as of this writing, become one of the most widely implemented error-detection schemes, being used extensively, not only in data communication systems, but also in data storage systems, such as in floppy disk controllers for use with mini and microcomputers.

The relationship between the CRC polynomial mathematics and digital switching circuits was presented (Chapter V) in order to provide some background for the theory leading up to the actual circuit implementation. A detailed example of the use of the CRC for data encoding and error detecting was shown (Chapter VI)

so that its simplicity and effectiveness could be more easily understood. Finally, the state of the current technology was presented (Chapter VII), as it applies to the CRC.

The material presented in this thesis covers a span of about 25-30 years, from the time when the theory for polynomial (cyclic) codes was first introduced to the present, where, with the continuing refinement of the integrated circuit, the applications for error detection have evolved to a point that makes almost full and complete use of the theories.

APPENDIX A

MODULO-2 ARITHMETIC

In the manipulation of binary polynomials, all of the mathematics were performed using modulo-2 arithmetic. Any operation performed modulo-2 means that the operation is performed in normal arithmetic and the result is divided by 2, and only the remainder is used. In order to more easily grasp the theory and the application of the Cyclic Redundancy Check, the basic rules for modulo-2 (mod-2) arithmetic are presented here.

The truth table for the addition of two binary numbers, X and Y, is shown in Figure A.1. S represents the sum digit and C represents the carry digit.

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure A.1. Truth Table For Binary Addition

In mod-2 addition, the arithmetic is performed as it was for normal binary addition, but only the sum bit is used. The carry bit is ignored. The rules for mod-2 addition are:

$$0+0 = 0$$

$$0+1 = 1$$

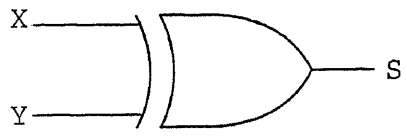
$$1+0 = 1$$

$$1+1 = 0$$

Modulo-2 addition is also known as logical addition because the sum bit can be generated by an exclusive-OR (XOR) gate. Figure A.2 shows the truth table for the exclusive-OR function and the logic symbol for the XOR gate.

X	Y	S
0	0	0
0	1	1
1	0	1
1	1	0

(a)



$S = X \oplus Y$

$\oplus$ is the symbol for the exclusive-OR function

(b)

Figure A.2. Exclusive-OR (a) Truth Table, and (b) Logic Symbol

The truth table for subtracting Y from X is shown in Figure A.3. D is the difference digit and B is the borrow.

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Figure A.3 Truth Table For Binary Substration

In mod-2 subtraction, only the difference bit is used and the borrow is ignored. The truth tables for mod-2 addition and subtraction are the same if the carry and the borrow bits, respectively, are disregarded. Therefore, mod-2 subtraction is the same as mod-2 addition.

The rules for mod-2 multiplication are:

$$0x0 = 0$$

$$0x1 = 0$$

$$1x0 = 0$$

$$1x1 = 1$$

Mod-2 multiplication is performed using the logical AND function.

The rules for modulo-2 division are the same as those for normal long division, except that all multiplications and subtractions are performed mod-2. Several examples of mod-2 division are shown in Chapters IV, V, and VI.

BIBLIOGRAPHY

Birkner, John, and Coli, Vincent, PAL Programmable Array Logic Handbook, Sunnyvale, Calif., Monolithic Memories, Inc., 1981, pp. 4-302, 4-303.

"Data Comm Async and Sync Transmission", Signetics Technical Brief 4004, September, 1981.

"Data Comm Message Protocol", Signetics Technical Brief 4005, September, 1981.

"Data Comm Protocols", Signetics Technical Brief 4000, September, 1981.

Fink, Donald G., ed., Electronics Engineers' Handbook, New York, N.Y., McGraw-Hill, Inc., 1975, pp. 23-25 to 23-36.

FitzGerald, Jerry, and Eason, Tom S., Fundamentals of Data Communications, New York, N.Y., John Wiley and Sons, Inc., 1978, pp. 113-124.

Goldberger, Alex, "A Designer's Review of Data Communications", a reprint from Computer Design, May, 1981.

Hamming, Richard W., Coding and Information Theory, Englewood Cliffs, N.J., Prentice-Hall, Inc., 1980, pp. 26-33.

Howell, Dave, ed., IC Master, Garden City, N.Y., Hearst Business Communications, Inc./UTP Division, 1982 ed., pp. 429, 430, 1104, 1314, 2061.

McLeod, Jonah, "IC's Rush to Fill Gaps in Data-Comm, Phone System", Electronics Design, Vol. 30, No. 5, March 4, 1982, pp. 63-74.

McNamara, John E., Technical Aspects of Data Communication, Bedford, Mass., Digital Press, 1977, pp. 145-158, 191-228.

Micom Systems, Inc., Data Communications for Minicomputer Users: A General Introduction, Chatsworth, C.A., 1980.

Peterson, W. Wesley, Error-Correcting Codes, Cambridge, Mass., The MIT Press, 1961, pp. 1-5, 107-161.

St. Amand, Joseph V., and Seifert, William, "Ethernet Communications Controller Does It All", Systems and Software, Vol. 1, No. 4, Dec., 1982, pp. 57-62.

Taub, Herbert, and Schilling, Donald, Digital Integrated Electronics, New York, N.Y., McGraw-Hill, Inc. 1977

Wakerly, John, Error-Detecting Codes, Self-Checking Circuits, and Applications, New York, N.Y., North-Holland, 1978 pp. 24-33.

Wiggert, Djimitri, Error-Control Coding and Applications, Dedham, Mass., Artech House, Inc., 1978, pp65-76.

Class Notes from CIS 651, "Data Communications", N.J.I.T.