BURST-ERROR-CORRECTING CONVOLUTIONAL CODES

MAJID REZAYAT

A MAJOR TECHNICAL REPORT

IN

THE DEPARTMENT

OF

ELECTRICAL ENGINEERING

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering at

Concordia University

Montreal, Quebec, Canada

February, 1979

ABSTRACT

BURST-ERROR-CORRECTING CONVOLUTIONAL CODES

Majid Rezayat

In this report, basic convolutional codes with rate $\frac{n_o-1}{n_o}$ and burst-error-correcting ability given by burst length $b \leq n_o$ are studied. The structure of block codes is compared with the structure of convolutional codes to understand their differences. Required bounds which must be followed by codes in order to have burst-error-correcting ability are given. Codes which are easy to implement and are good for use on real world communication channels are also studied.

Interleaving techniques, a powerful tool for constructing convolutional codes which can correct longer burst errors than basic codes or which have burst and random error correcting ability, are described. In order to make a fair comparison of these codes for a fixed burst-error-correcting ability, their minimum required guard space is considered in the report.

ACKNOWLEDGEMENT

I wish to express my gratitude to Dr. V.K. Bhargava for suggesting the topic and for his guidance throughout the course of this investigation. I would also like to thank Miss M. Hayes and Mr. W. Kotiuga for editing the manuscript and Miss J. Anderson for typing this report.

Finally, I wish to thank my wife, Elaheh, whose devotion gave me the time to provide this report.

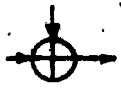
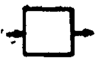
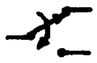
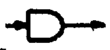
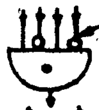
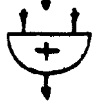
TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ARQ	Automatic Repeat Request
b	Burst length
B	Semi-infinite matrix with n_o columns
B_o	Matrix formed from the first N rows of matrix B
B_i	Semi-infinite matrix formed from matrix B by shifting all rows down i places
BPM	Berlekamp-Preparata-Massey
BSC	Binary symmetric channel
$C(f)$	Column f of matrix H_N
d_j^i	ith information digit in jth block
d_k	kth information digit
e	Semi-infinite error column vector
e_k^d	kth information digit error
e_i^j	jth error digit in ith block
e_N	Error column vector of dimension n
e_k^p	kth parity digit error
E	Error column vector
E^T	Transpose matrix of error column vector E

$f \varepsilon F$	The element f is contained in the set F
$F \subset K_b$	Every element of set F is in set K_b
$F \cap K_{n_o}$	Intersection of sets F and K_{n_o} (that is, the set of elements commonly contained in sets F and K_{n_o})
FEC	Forward error control
FM	Frequency modulation
FSK	Frequency shift keying
g	Guard space length
H	Parity check matrix
H_N	Matrix formed from the first N rows and first n columns of H
HF	High frequency (3-30 MHz)
i	Parity digits in each n_o-bit block
k_o	Information digits in each n_o-bit block
K_{n_o}	Set $\{1,2,\ldots,n_o\}$
$L(n_o)$	Length of the largest binary number among the first n_o odd binary numbers
m	Memory of convolutional code $= \frac{N}{i}$
n	Constraint length of convolutional code $= \frac{N}{i} n_o$

n_o	Block length
N	Number of first rows of matrix B, $(N \geq P_o)$
p	Probability of error
p_j	Parity digit in jth block and also jth parity digit
$p(t)$	Probability of error as a function of time
P_o	Number of nonzero rows of matrix B
P_j	Parity word of m digit $(j=1,\ldots,n_o)$
PSK	Phase-shift keying
r	Number of blocks
R	Code rate $= \frac{k_o}{n_o}$
RF	Radio frequency
S	Semi-infinite syndrome column vector
S^T	Transpose of matrix S
S_N	Syndrome column vector of dimension N
t	Number of random errors
T	Transform matrix
X	Transmitted data column vector
Y	Received data column vector
λ	Interleaving degree

Δ_0	Error density
Γ	Represents type B_1 or B_2 codes
	Symbol for exclusive OR logic
	Symbol for shift register
	Symbol for multiplexer
	Symbol for inverter
	Symbol for AND gate
	Symbol for OR gate

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In many digital communication systems, due to power and bandwidth constraints, it is impossible to rely on the improvement of conventional modems alone. Error correction codes can improve the performance of these systems by lowering the required signal-to-noise ratio at the expense of an increase in channel bandwidth.

A common misconception about error correcting codes is that they can be used to correct all errors of the system. This is certainly not true. Given a certain amount of signal redundancy one tries to correct the set of most probable errors. Although the correction procedure usually improves the reliability of the system, but by no means takes care of all errors. On the positive side it should be said that many codes exist to improve the reliability of the system drastically provided that sufficient redundancy is available. The effectiveness of coding is also strongly dependent on communication channels.

Communication systems are designed considering that the statistics of the channel are stationary. Therefore the noise introduced by the channel has Gaussian type distribution [1]. In the real world channel, statistics are not stationary. The error rate varies over a range of time and the distribution is complex [2].

Burst-error-correcting codes are being developed to improve the performance and reliability of digital communication systems in the presence of burst noise which usually occurs in real communication channels, like HF radios, troposcatter radios and telephone lines. Burst noise causes short-term channel interruptions which are not Gaussian. Hence they are not considered in the communications equipment system design. Interference in HF radio channel due to lightning, and microwave radio system blockages due to an aircraft flying too close to an antenna are examples of burst noises.

The basic idea of coding success is dependent on the matching of the channel statistics and the code capabilities. Therefore it is important to know the statistical properties of the channel in use to make sure it matches the class of code selected.

There are certain reasons for the increasing applications of coding systems: The rapid growth in satellite communications; the revolution in digital integrated circuits; the availability of inexpensive computers for system, algorithm, and hardware simulation; the increasing emphasis on the reliable transmission of digital data and of digitally coded analog signals; and most important inventions of effective decoding algorithms [3].

In this report, we will study the well known burst-error-correcting convolutional codes introduced by different authors and consider their advantages and disadvantages over each other. The presentation is organized as follows: Chapter 2 gives some definitions, background and analysis of burst error correcting convolutional codes.

In Chapter 3, some convolutional codes which are good for burst-error-correction are described. Chapter 4 considers codes which have ability to correct burst and random errors simultaneously and have practical use in communication systems. Finally Chapter 5 is the summary and conclusions.

CHAPTER 2

SOME DEFINITIONS, BACKGROUND AND ANALYSIS OF BURST ERROR CORRECTING CONVOLUTIONAL CODES

Wyner and Ash [4] have presented a method of convolutional code analysis using parity check matrices. Rodgers [5] has followed their method with some minor changes and assumptions. Their work is summarized in this Chapter since it has some common similarity with the construction of the codes described in this report. Also, they give a general knowledge and better understanding of burst-error-correcting convolutional codes.

Definitions of key words and terms will be stated when needed. The structure of block codes is also described on the basis of parity check matrices in order to clarify the different structure existing between the block codes and the convolutional codes.

2.1 Random and Burst Errors

The random-error is defined by the occurrence of errors that are uncorrelated; that is, each error occurrence is independent of the immediate past history of error patterns in a message.

The burst-error is defined [6], [7] by a clustering effect in a region of the data stream, thus showing a dependence on the past history of errors. In this region a minimum of two errors exist. The region begins with an error digit that is immediately preceded by a correct digit. A specified error density Δ_o must exist in the region, where Δ_0 is defined as the ratio of error digits to total digits in the burst region for maximum length burst. A burst always ends with an error digit that is immediately followed by a correct digit. The above definition of burst-error can be expressed in mathematical form as follows:

Let b be the length of a sequence of digits, w the number of errors in the sequence, and b_i the length up to the ith error. $b_1 = 1$ means that the first digit is an error, $b_w = b$ means that the last digit is an error, and $i/b_i \geq \Delta_0$ means that the error density up to the ith error is larger than or equal to the specified value Δ_0. A burst of length b with w errors is a sequence of b digits satisfying the following:

1) $b_1 = 1$

2) $b_w = b$

3) $i/b_i \geq \Delta_0$ for $1 \leq i \leq w$

and

4) $w+1/b_{w+1} < \Delta_0$

2.2 Real Channels

As indicated before, burst errors occur on real channels. How these errors occur is really the channel characteristic manifesting the complicated physical phenomena of propagation. In order to apply error correcting codes to the real channel, one has to first, know the channel characteristic and second, be able to analyze the effectiveness of the codes with the knowledge of the channel characteristic.

A general block diagram of a digital transmission system is illustrated in figure 2.1 in order to explain what a channel means to coding theorist and communication engineers.

For a coding theorist a channel is a black box containing modulator, physical channel and demodulator while for a communication engineer, the channel decomposes into the above three parts.

In coding theory, the channel most frequently assumed is the binary symmetric channel (BSC), as shown in figure 2.2a, in which each transmitted bit has the probability $p < 1/2$ of being incorrectly received and that this probability is independent for all received bits (p is the probability of error in the hard decision of demodulator). This is a poor channel model for most real channels where there will be periods of high signal strength that the demodulator's decisions based on the above model are relatively unreliable. In this case, the coding channel is actually a time varying BSC in which the error probability $p(t)$ changes with time t .

Kohlenberg and Forney [8] have given a useful division of such real channels under the term "dense-burst" channel, and "diffuse-burst" channels.

In a dense-burst channel, the time axis divides rather sharply into intervals of nearly perfect transmissions, say $p(t) \approx 10^{-6}$, and intervals of very poor transmissions, say $p(t) \approx 1/2$.

In a diffuse-burst channel, there are intervals of rather

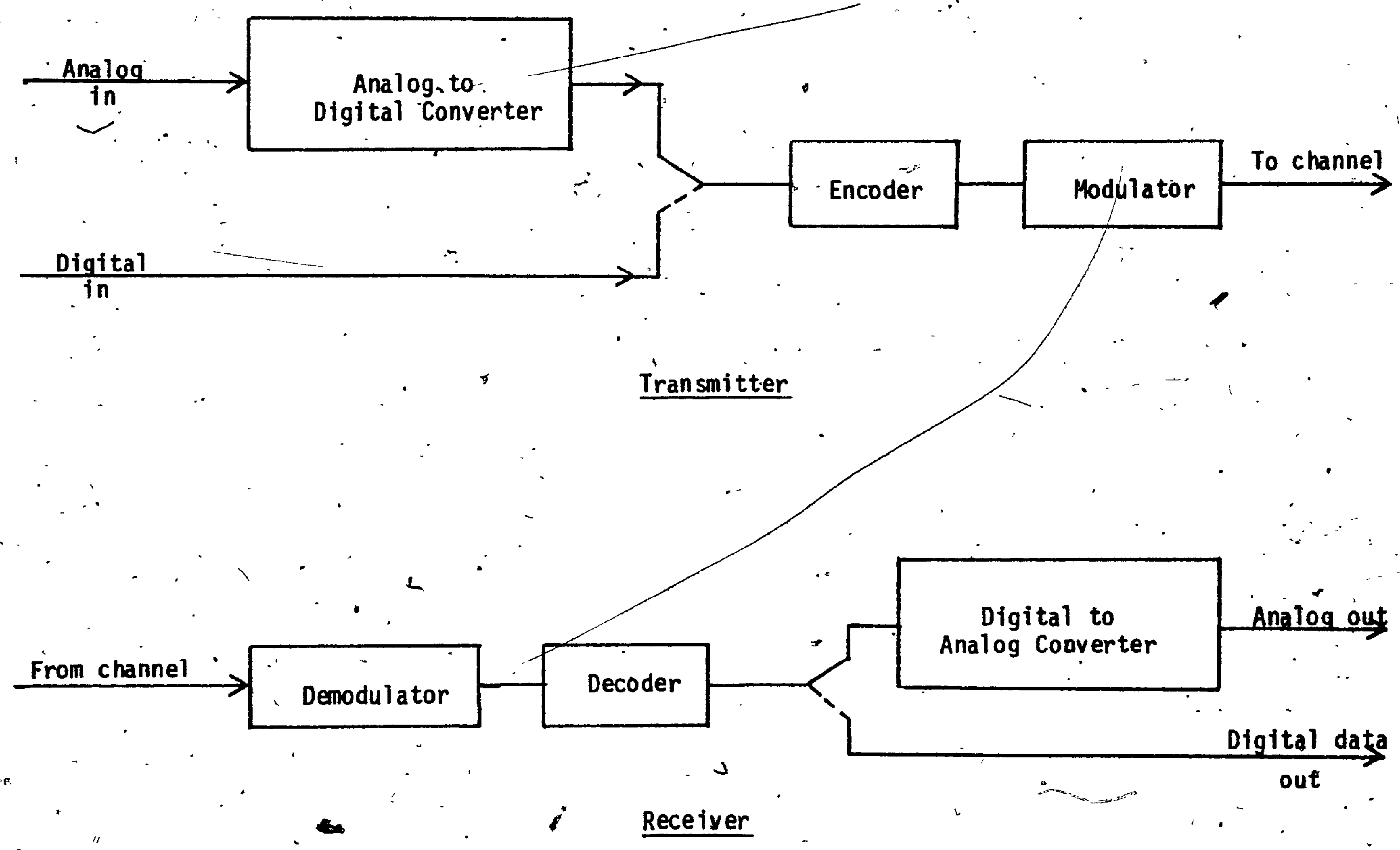


FIG. 2.1 Block Diagram of a Digital Transmission System

good transmission, say $p(t) \approx 10^{-4}$ and rather poor transmission, say $p(t) \approx 10^{-1}$ with a rather indistinct boundary between regions. Figure 2.2b illustrates these two types of real channels.

According to Kohlenberg and Forney a troposcatter system using FM modulation of several frequency division multiplexed data channels onto the RF carrier is a typical dense-burst channel and an HF data speed system with PSK or FSK modulation is a typical diffuse-burst channel.

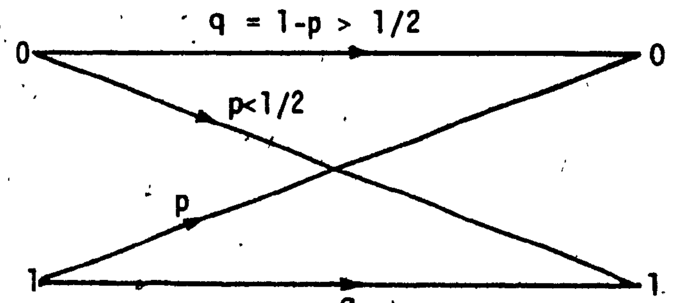
It is clear that these two classes of real channels will demand quite different coding systems for their effective use.

2.3 Error Control Techniques

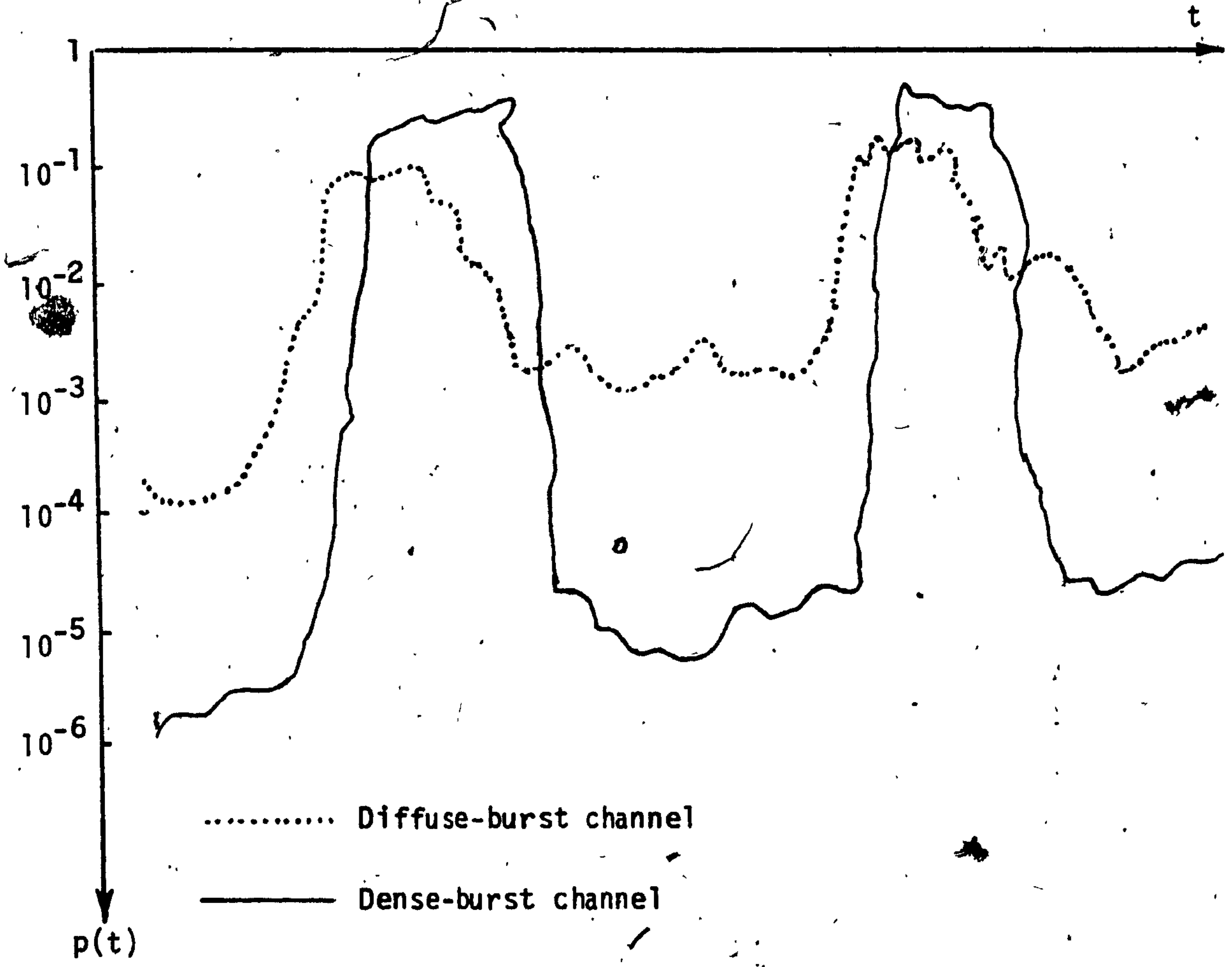
Error control coding techniques have been developing during the last two decades that it has become an essential part of the digital communication system design when an efficient and reliable data transmission is demanded.

Encoders and decoders are built for this reason using different algorithms for detecting and/or correcting errors. All of them have a common basis which is the addition of some extra digits (redundant or parity digits) to information digits by the encoder at the transmit point which in turn would be extracted by the decoder at the receive point. These extra digits are used for detecting and/or correcting data digits error.

In general, there are two types of error control techniques used for improving the efficiency and reliability of data transmission systems, called ARQ and FEC techniques [9].



(a)



(b)

FIG. 2.2 a) The Binary Symmetric Channel (BSC)

b) Idealized Dense-Burst and Diffuse-Burst Channels

2.3.1 Automatic-Repeat-Request Technique (ARQ)

The general function of ARQ system is shown in figure 2.3. Data are delivered from a source to a source encoder. The source encoder arranges data in blocks, buffers the blocks, attaches control and synchronization bits, and generally controls the network. Then, the data blocks are delivered to an encoder that adds the required extra redundant digits. The encoded block then will be modulated and transmitted over the channel. At the receive point, the received data will be demodulated and then delivered to the decoder. The decoder re-computes the redundant digits from the received data and compares them with the received redundant digits for each block. If there are no discrepancies, the data block will be delivered to the user through the source decoder and the source encoder will be informed by source decoder through a suitable feedback channel that the block has been correctly received. If discrepancies exist, again the source encoder will be informed of the situation and the block will be re-transmitted. Hence, erroneous data are delivered to the user when the decoder fails to detect the presence of errors. This technique is applicable when the feedback channel is available and the delay time between transmit and receive points is negligible.

2.3.2 Forward Error Control Technique (FEC)

This system can be depicted by removing the feedback channel from figure 2.3. The decoder is then a more complicated device. It attempts to determine the location of the errors from the pattern of discrepancies between the received and re-calculated redundant digits

2.3.1 Automatic-Repeat-Request Technique (ARQ)

The general function of ARQ system is shown in figure 2.3. Data are delivered from a source to a source encoder. The source encoder arranges data in blocks, buffers the blocks, attaches control and synchronization bits, and generally controls the network. Then, the data blocks are delivered to an encoder that adds the required extra redundant digits. The encoded block then will be modulated and transmitted over the channel. At the receive point, the received data will be demodulated and then delivered to the decoder. The decoder re-computes the redundant digits from the received data and compares them with the received redundant digits for each block. If there are no discrepancies, the data block will be delivered to the user through the source decoder and the source encoder will be informed by source decoder through a suitable feedback channel that the block has been correctly received. If discrepancies exist, again the source encoder will be informed of the situation and the block will be re-transmitted. Hence, erroneous data are delivered to the user when the decoder fails to detect the presence of errors. This technique is applicable when the feedback channel is available and the delay time between transmit and receive points is negligible.

2.3.2 Forward Error Control Technique (FEC)

This system can be depicted by removing the feedback channel from figure 2.3. The decoder is then a more complicated device. It attempts to determine the location of the errors from the pattern of discrepancies between the received and re-calculated redundant digits

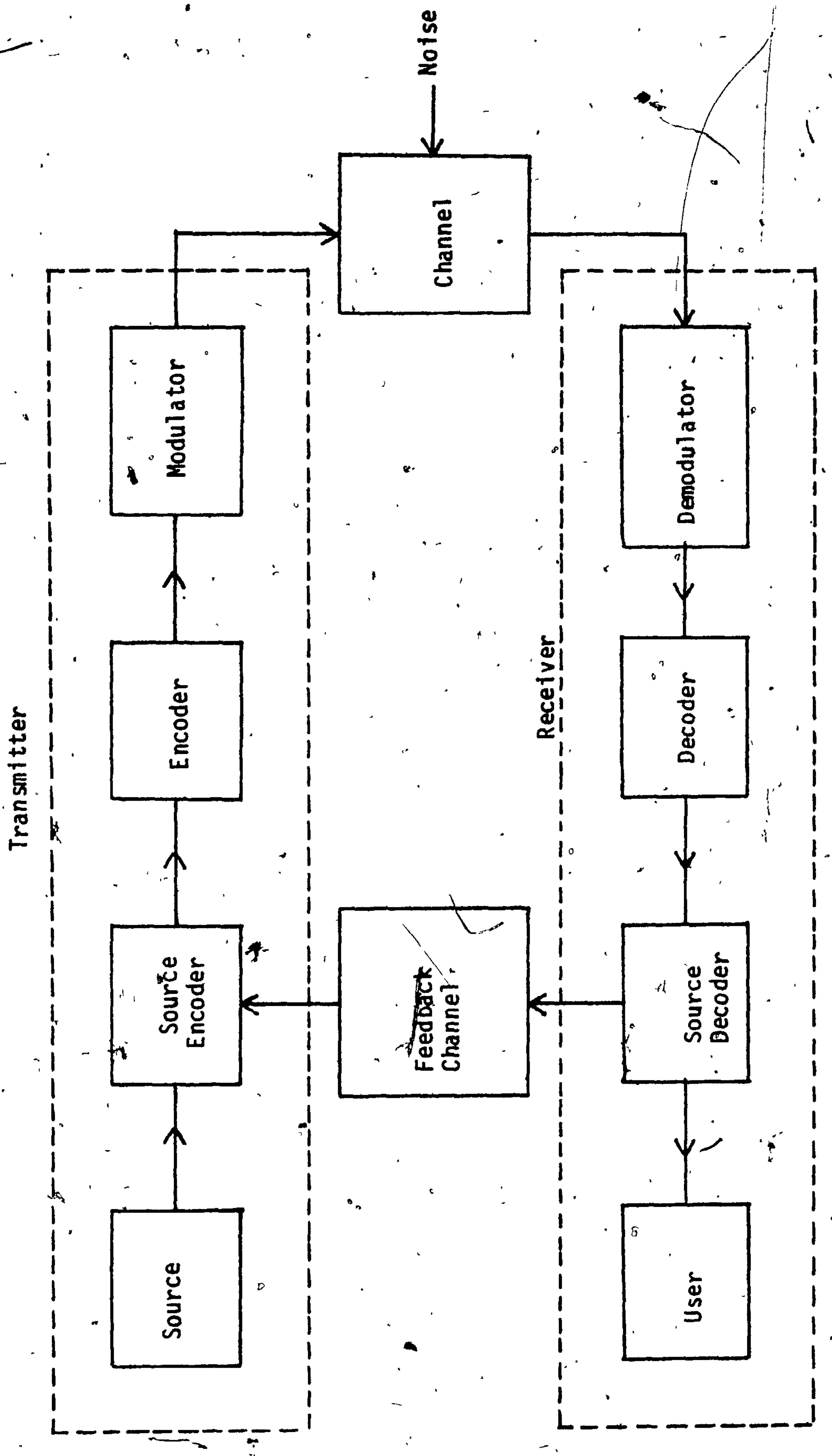


FIG. 2.3 ARQ System

and then correct them. Hence, erroneous data will be delivered to the user when the decoder cannot detect and correct errors.

There are two types of FEC codes, called block codes and convolutional or recurrent codes. For block codes there is no memory between code blocks. For conventional codes there is memory for each encoded bit. These codes are explained in detail later.

The FEC coding technique will have a potential use in digital data transmissions through satellite. Especially, when a geostationary satellite is used that there will be a large time delay, about 240 ms [10], from one ground station to another.

The codes which we have considered in this report are all of FEC type.

2.4 Classification of Burst-Error-Correcting Convolutional Codes

Burst error correcting codes in general may be classified as follows:

a) Codes which can correct burst errors.

b) Codes which can correct burst and random errors both together.

Codes which are good for burst and random error correction are usually constructed from codes good for random error correction by using an interleaving technique [11]. The principle behind this technique is to break up the burst error bits so that only one, or at most a few bits can be affected by a single burst of errors. Then the errors affecting each interleaved code are treated as if they were random, independent errors.

These codes are explained in detail later.

Burst error correcting convolutional codes in general are labelled as type B_1 or type B_2 codes [4].

2.4.1 Type B_1 Codes

These convolutional codes are able to correct all burst of length not greater than "b" digits of any "n" consecutive digits provided that "g" digits error free space which is known as guard space is available at each side of the burst. "n" is known as constraint length of convolutional codes and will become clearer later.

2.4.2 Type B_2 Codes

These codes are capable of correcting all burst of length "b" digits of any "n" consecutive digits provided that "g" blocks error free space is available at each side of the burst with the additional restriction on the burst that it be confined to "r" consecutive n_0-bit blocks, that is $b \leq r\, n_0$ digits. Type B_2 codes have been analyzed to a considerably greater extent than type B_1 codes.

2.5 Block Codes Structure

In block codes to each k_0 information digits at a time, n_0-k_0 parity digits are added by using a proper algorithm. These parity check digits do not depend on other previous or future k_0 information digits. The code is known as (n_0, k_0) code and code rate or efficiency is defined as $R = \frac{k_0}{n_0}$. Let X be a q-row column vector which represents transmitted sequence of information and parity digits and H be an m×q binary parity check matrix. Any sequence of code words comprising X

must satisfy the parity check equation. (The most recent transmitted bit is assumed to be the bottom row of X).

$$HX = 0 \pmod{2} \tag{2.1}$$

For a binary block code, the above equation has the form

$$HX = \left[\begin{array}{c:c:c:c:c} H_{11} & 0 & 0 & 0 & \cdots \\ \hdashline 0 & H_{22} & 0 & 0 & \cdots \\ \hdashline 0 & 0 & H_{33} & 0 & \\ \hdashline \vdots & \vdots & \vdots & & \ddots \end{array}\right] X = \underline{0} \tag{2.2}$$

where

$$H_{11} = H_{22} = H_{33} = \dots = H'$$

Each sub-matrix H' has n_o-k_o rows and n_o columns. Each block of n_o bits of X satisfy disjoint parity check equations. Hence each n_o-bit block can be encoded and decoded with no knowledge of future or past blocks of X. If X_j is a column vector of n_o transmitted bits, then we only need that each block of n_o transmitted bits, X_j ($j=1,\dots$) satisfy the equation:

$$H'X_j = 0 \tag{2.3}$$

In block codes, matrix H' is normally defined as the parity check matrix.

If the first k_o bits of each block code (or sub-block for convolutional codes) are unchanged data bits and the last n_o-k_o bits are must satisfy the parity check equation. (The most recent transmitted bit is assumed to be the bottom row of X).

$$HX = 0 \pmod 2 \tag{2.1}$$

For a binary block code, the above equation has the form

$$HX = \left[\begin{array}{c|c|c|c|c} H_{11} & 0 & 0 & 0 & \cdots \\ \hline 0 & H_{22} & 0 & 0 & \cdots \\ \hline 0 & 0 & H_{33} & 0 & \\ \hline & & & & \ddots \end{array}\right] X = 0 \tag{2.2}$$

where

$$H_{11} = H_{22} = H_{33} = \dots = H'$$

Each sub-matrix H' has $n_o - k_o$ rows and n_o columns. Each block of n_o bits of X satisfy disjoint parity check equations. Hence each n_o-bit block can be encoded and decoded with no knowledge of future or past blocks of X. If X_j is a column vector of n_o transmitted bits, then we only need that each block of n_o transmitted bits, X_j ($j=1,\ldots$) satisfy the equation:

$$H'X_j = 0 \tag{2.3}$$

In block codes, matrix H' is normally defined as the parity check matrix.

If the first k_o bits of each block code (or sub-block for convolutional codes) are unchanged data bits and the last $n_o - k_o$ bits are parity bits then the code is called a systematic encoded code. A transmitted stream of code bits is shown in figure 2.4.

2.6 Convolutional Codes Structure

An (mn_o, mk_o) convolutional code is defined by a set of parity check equations of the following form. Let B be a semi-infinite matrix with n_o columns and an infinite number of rows. B will be restricted to having only P_o finite number of nonzero rows confined to the first N rows of matrix B. Let B_i be a matrix formed from B shifting all of the rows of B down i places. Then the parity check matrix H of the code is:

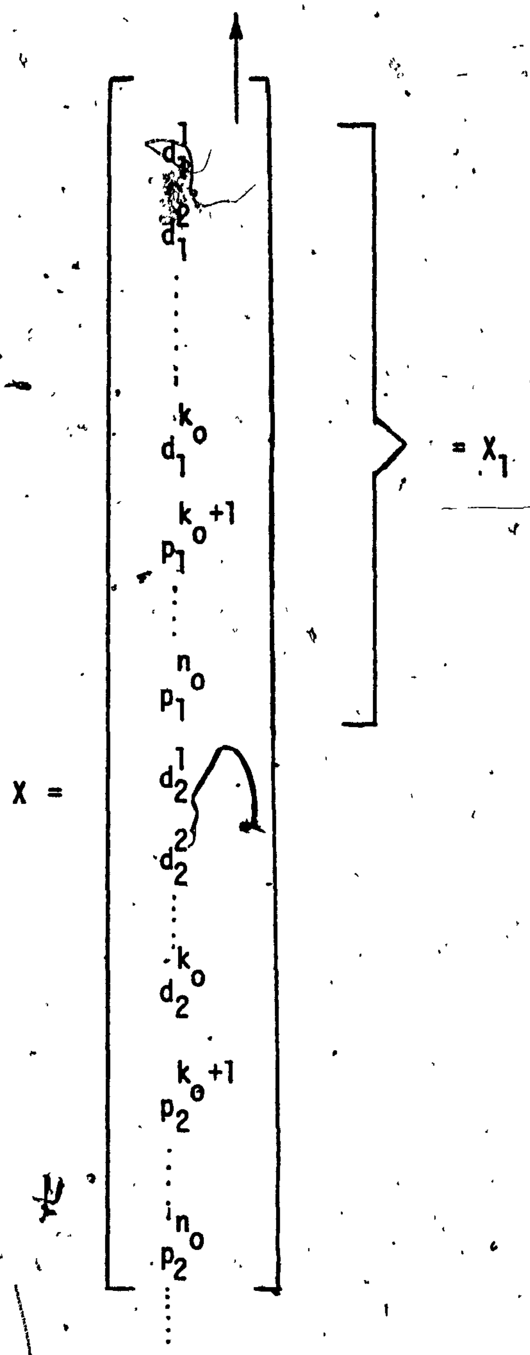
$$H = [B, B_i, B_{2i}, B_{3i} \dots] \tag{2.4}$$

where $i = n_o - k_o$ is a fixed positive integer. Matrix H, as a function of the matrix B, is schematically shown in figure 2.5 (B_o in this figure is a sub-matrix formed from the first N rows of matrix B). We notice that the parity equations are not disjoint as with block codes. The code words are the semi-infinite sequences defined by the column vector X such that $HX = 0 \pmod 2$.

2.6.1 Convolutional Encoding

The name convolutional encoding arises because the output of the encoder may be regarded as the convolution of the input streams of data digits and the response function of the encoder.

Most of the useful convolution codes either take blocks of



d_j^i stands for ith information digit in block j $(i=1,2, \ldots, k_o)$

p_j^k stands for kth parity digit in block j $(k=k_o+1, \ldots, n_o)$

FIG. 2.4 Transmitted Stream of Systematic Encoded Bits

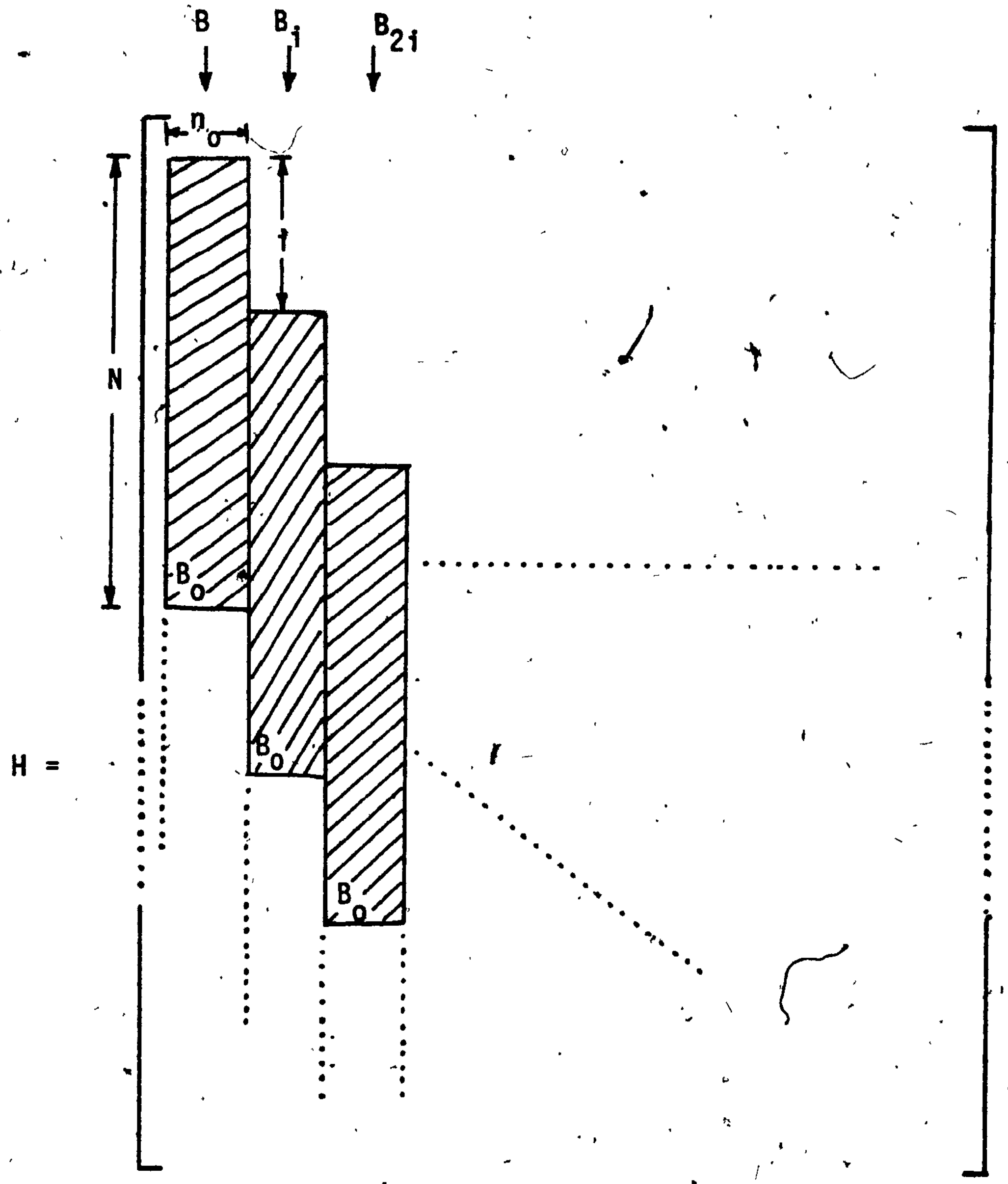


FIG. 2.5 Parity Matrix H as a Function of Matrix B (nonzero entries are in the shaded areas, B_o)

single digits into the encoder at a time and produce one or more check digits or take in blocks of several digits and produce one check digit each time. This means that code rates are 1/2, 1/3, 1/4, ... or 1/2, 2/3, 3/4 ... Codes with rates equal to 1/2 or greater are known as high rate codes and codes with rates smaller than 1/2 are known as low rate codes.

With consideration on k_0, n_0 and B it is possible to find matrix H for convolutional encoding and construct codes with different rates and degrees of usefulness.

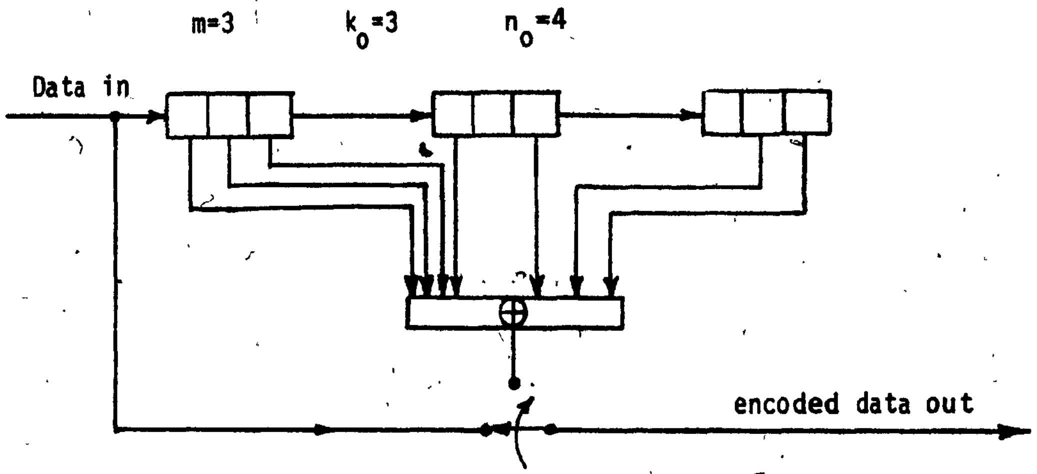
There are two kinds of encoders for convolutional codes [12], one is the k-stage encoder which accepts data in serial form and another one is the (n-k)-stage encoder which accepts the k_0 data bits at a time on k_0 parallel leads and emits n_0-k_0 parity bits on parallel leads. k-stage encoders require $k=mk_0$ shift register stages for implementation and (n-k)-stage encoders require v shift register stages where $v \leq (m-1)(n_0-k_0)$ and $m = \frac{N}{i}$, (N is the number of rows in matrix B_0 and $i = n_0-k_0$) . It may be noted that for (n-k)-stage encoders the required shift register stages are not equal to n-k stages. As an example encoders for (12,9) code are illustrated in figure 2.6. In figure 2.7, a three stage encoder and related data digits (input) and encoded data sequences (output) are shown knowing that information bits are shifted in one at a time. We shall pursue this example to develop various representation of convolutional codes and their properties.

It is traditional and instructive to exhibit a convolutional code by means of a tree diagram as shown in figure 2.8. At each junction in the tree, the upper path shows what happens if the next information single digits into the encoder at a time and produce one or more check digits or take in blocks of several digits and produce one check digit each time. This means that code rates are 1/2, 1/3, 1/4, ... or 1/2, 2/3, 3/4 ... Codes with rates equal to 1/2 or greater are known as high rate codes and codes with rates smaller than 1/2 are known as low rate codes.

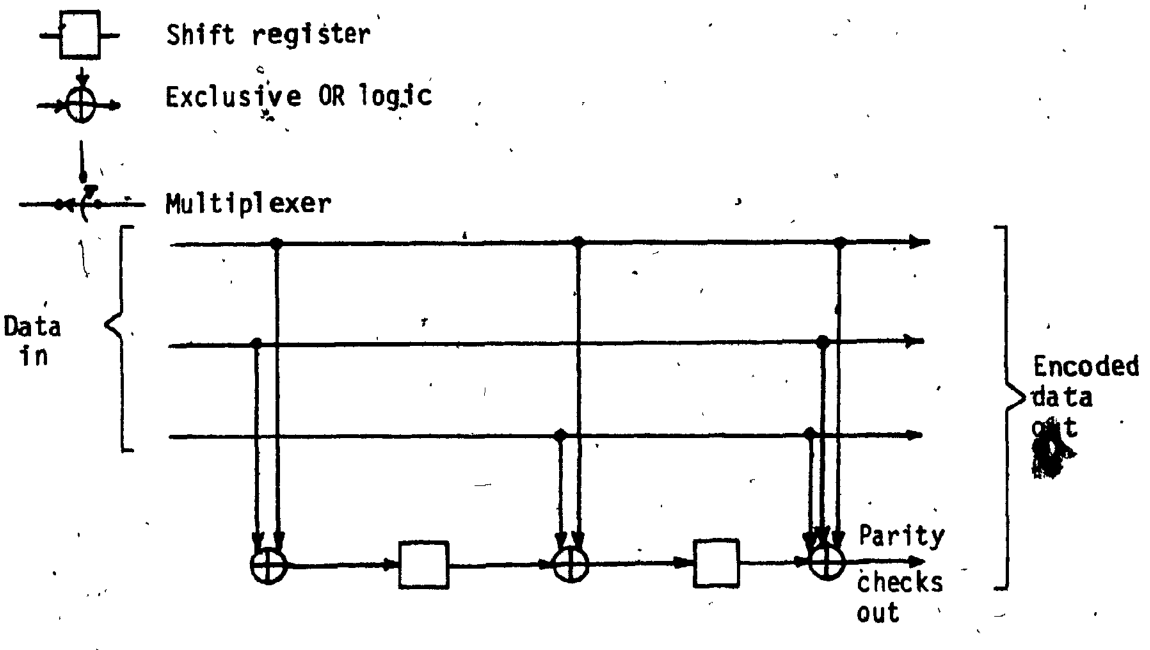
With consideration on k_o, n_o and B it is possible to find matrix H for convolutional encoding and construct codes with different rates and degrees of usefulness.

There are two kinds of encoders for convolutional codes [12], one is the k-stage encoder which accepts data in serial form and another one is the (n-k)-stage encoder which accepts the k_o data bits at a time on k_o parallel leads and emits n_o-k_o parity bits on parallel leads. k-stage encoders require $k=mk_o$ shift register stages for implementation and (n-k)-stage encoders require v shift register stages where $v \leq (m-1)(n_o-k_o)$ and $m = \frac{N}{i}$, (N is the number of rows in matrix B_o and $i = n_o-k_o$). It may be noted that for (n-k)-stage encoders the required shift register stages are not equal to n-k stages. As an example encoders for (12,9) code are illustrated in figure 2.6. In figure 2.7, a three stage encoder and related data digits (input) and encoded data sequences (output) are shown knowing that information bits are shifted in one at a time. We shall pursue this example to develop various representation of convolutional codes and their properties.

It is traditional and instructive to exhibit a convolutional code by means of a tree diagram as shown in figure 2.8. At each junction in the tree, the upper path shows what happens if the next information



a) 9-Stage Encoder



b) 2-Stage Encoder

FIG. 2.6 Encoders for (12,9) Code

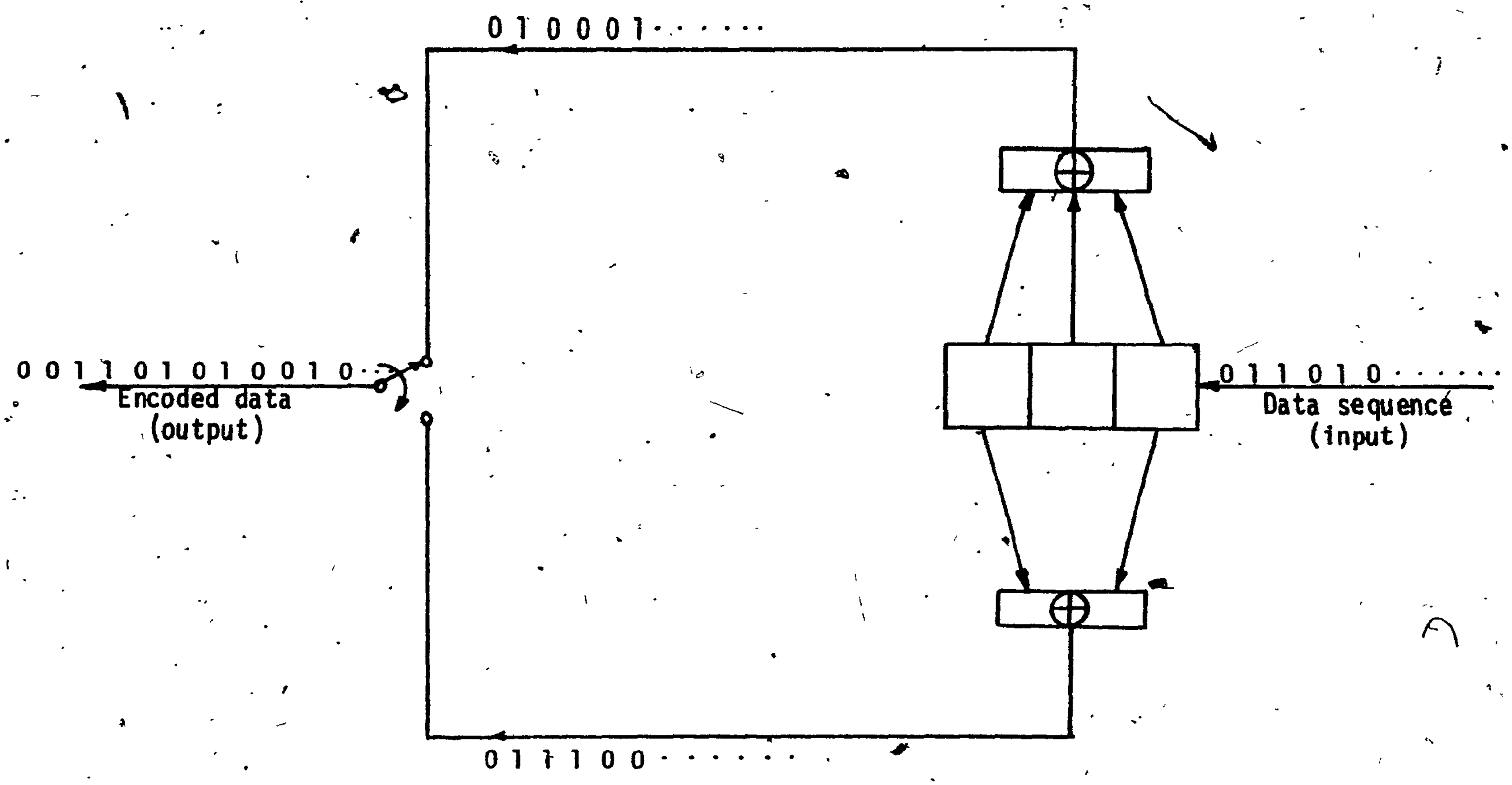


FIG. 2.7 3-Stage Convolutional Code Encoder

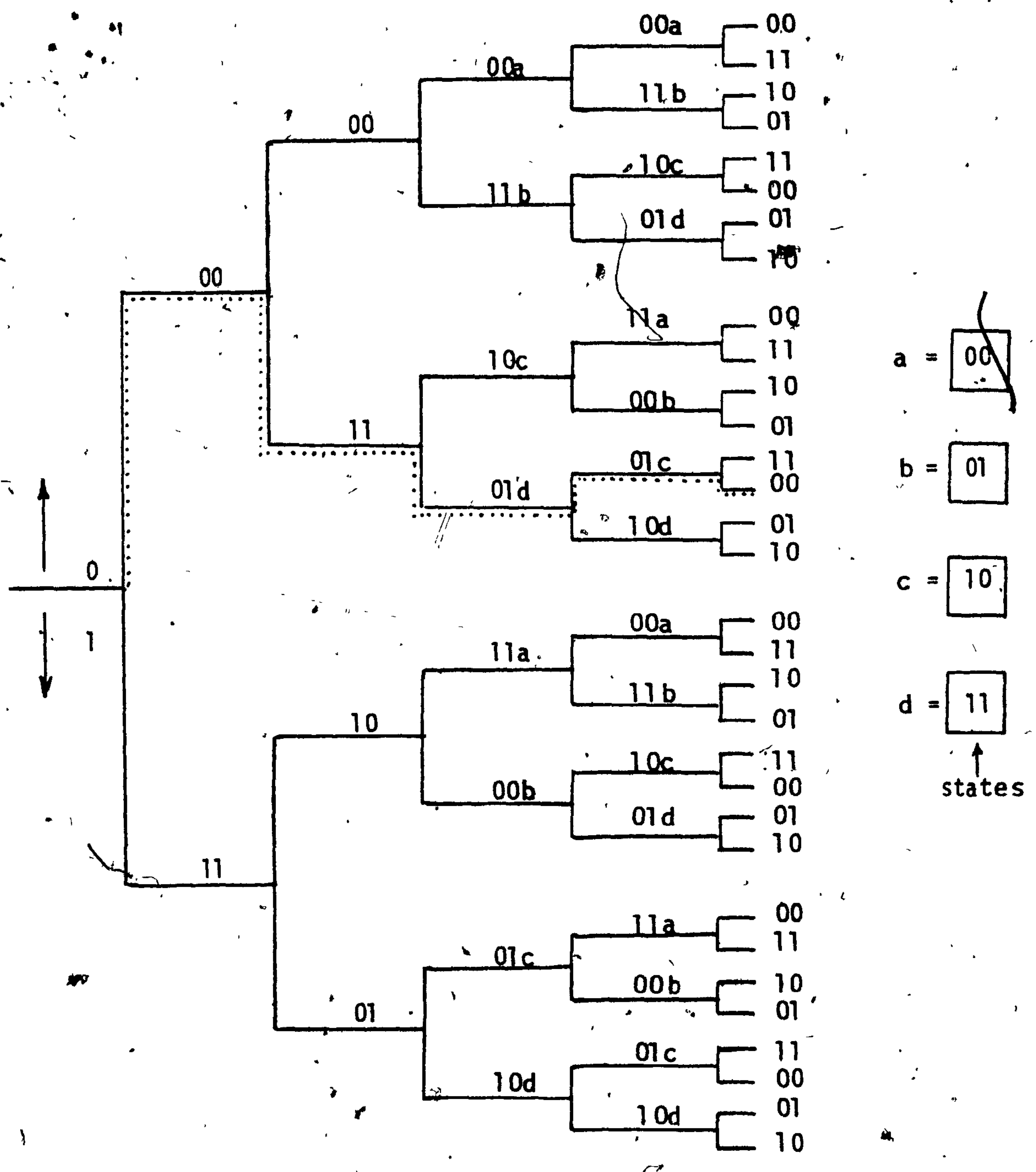


FIG. 2.8 Tree-Code Representation for Encoder of Fig. 2.7

bit is a "zero", the lower path displays the results of a "one". In this manner all the 32 possible outputs for the first 5 inputs are traced. The labelling on the graph indicates the output symbols (00, 01, ... etc.) and the contents of the first k-1 shift register stages (a=00, b=01, ... etc.). The contents of the first k-1 stages are known as the states, since subsequent symbols depend only on the state plus future input information bits. Obviously, there are 2^{k-1} states when information bits are shifted in one at a time. For the above example we have $2^{3-1} = 4$ states.

From the tree diagram we notice that after the first three branches, the structure becomes repetitive. In fact, we readily recognize that beyond the third branch, the code symbols on branches originating from the two nodes labelled "a" are identical; similarly for all the identically labelled pairs of nodes. This leads to redrawing the tree diagram as illustrated in figure 2.9 by joining the identical labelled nodes together. This has been called a trellis diagram [13], since a trellis is a tree-like structure with remerging branches. We should make note that code branches produced by a "zero" input bit are shown as solid lines and code branches produced by a "one" input bit are shown dashed. The trellis structure is used in Viterbi decoding scheme [14] of convolutional codes.

According to the repetitive structure of the trellis diagram, it is also possible to represent the code by the state diagram of figure 2.10. The states of the state diagram are labelled as the nodes of the trellis diagram. For the given example the nodes represent the previous two bits while the present bit is indicated by the

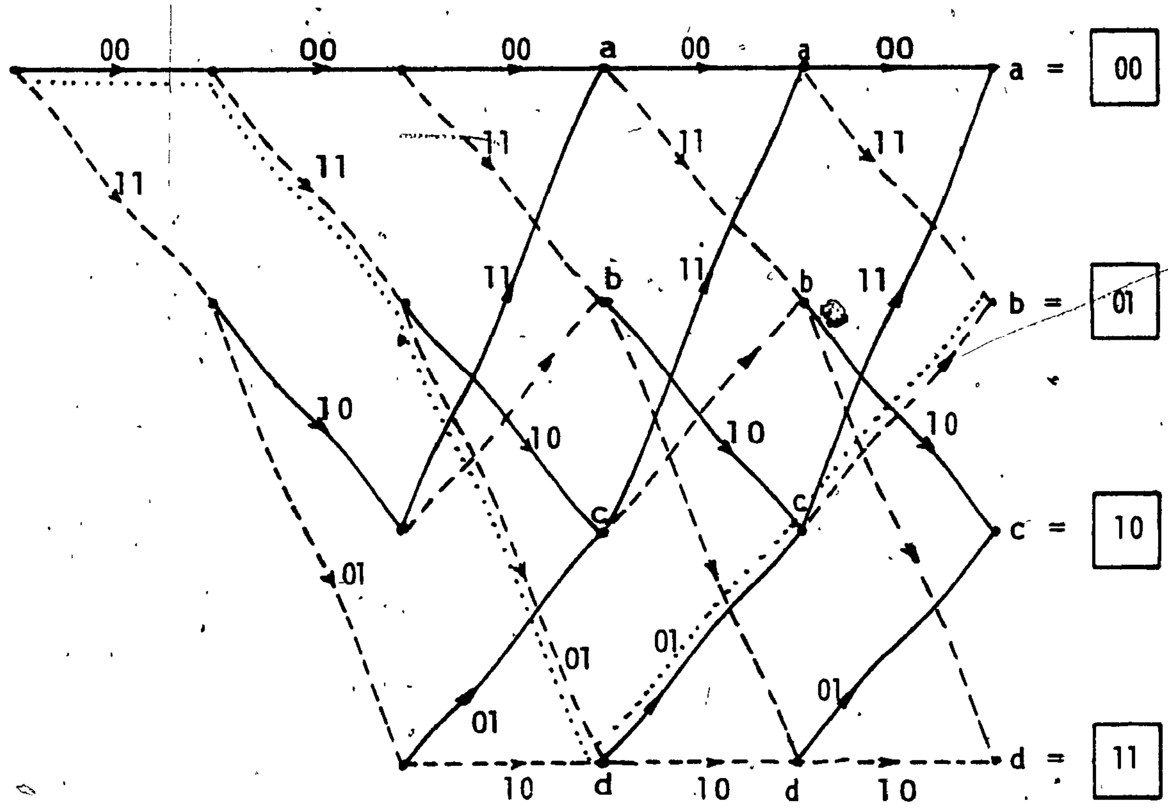


FIG. 2.9 Trellis-Code Representation for Encoder of Fig. 2.7

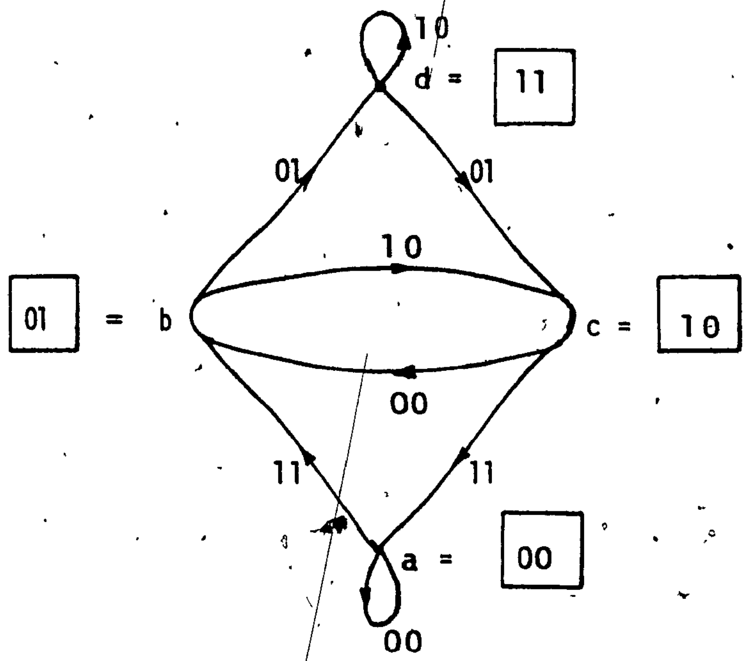


FIG. 2.10 State-Diagram Representation for Encoder of Fig. 2.7

transition branch; for example, if the encoder contains 101, that is represented in the diagram by the transition from state C=10 to state b=01 and the corresponding branch indicates the code symbol outputs "00". The state diagram is a very compact form of the encoder.

For the purpose of analysis, it is assumed the vector X is divided into sub-blocks of length n_0 and the first n_0-1 bits of each sub-block are data bits and the n_0th bit is a parity bit (systematic code). The first parity bit must satisfy the top equation of HX=0 . It is clear that there is a unique solution because there is one equation and one unknown. In order to make the solution easier and faster, Rodgers [5] minimized the required number of storage registers at both the encoder and decoder by making another assumption about the matrix B, that is, choosing the last i columns and top i rows of B as an identity matrix and remaining rows of the last i columns as zero. The H matrix for this code of rate $\frac{n_0-1}{n_0}$, (n_0 = 2, 3, ...) is shown schematically in figure 2.11. We notice that each parity bit is used in only one parity equation. The first parity bit is a function of the n_0-1 data bits in the first sub-block of X . The second parity bit is a function of the n_0-1 data bits in the first sub-block of X and the n_0-1 data bits in the second sub-block of X as shown by the second row of H . The Nth parity bit is a function of the n_0-1 data bits in the Nth sub-block and the $(N-1)(n_0-1)$ data bits of the N-1 sub-blocks, preceding the Nth sub-block (N is again the number of the last row of matrix B_0 with usual nonzero entries). The parity bit for sub-block Q greater than N(Q>N) is a function of only the data

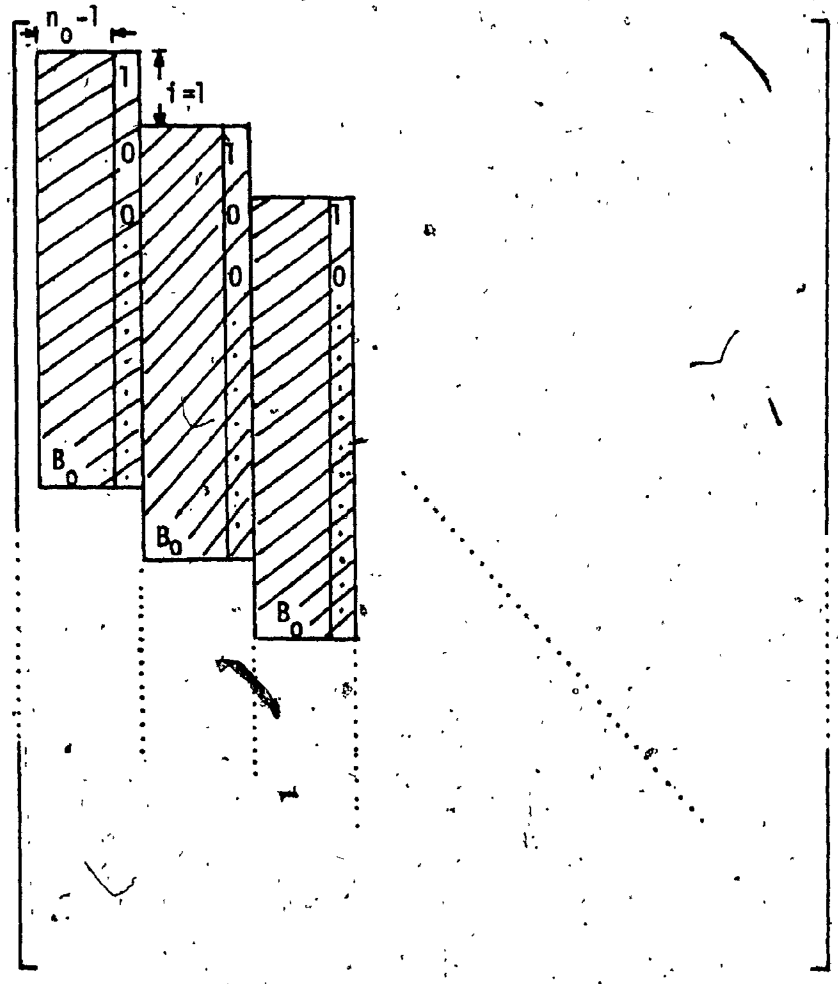


FIG. 2.11 Diagram of Matrix H to avoid Storing Parity Bits at the Decoder

bits in the N-1 sub-blocks preceding sub-block Q plus the data bits in the Qth sub-block. Therefore, the encoder needs only to store, at most, the data bits of the N-1 preceding sub-blocks. These bits plus the present n_0-1 data bits are sufficient to calculate the parity bit for the present sub-block. The number $n = \frac{N}{1} n_0$ bits spanned by the encoder is called the constraint length of the encoder.

2.6.2 Convolutional Decoding

Let us suppose the transmitted code word is vector X and the received code word is vector Y. If the communication channel is free of any noise disturbances we should have X=Y, but usually noise exists so we have

$$Y = X + e \pmod 2 \tag{2.5}$$

where e is a semi-infinite column vector representing errors due to noise. Now, if we re-encode the received code word exactly as we did at the transmit point, the result will be

$$HY = H(X+e) = He = S \tag{2.6}$$

"S" is called the syndrome and is a semi-infinite column vector which is the sum of columns of H corresponding to the bits in error. Its calculation is the basis for decoding.

Following Wyner and Ash [4], it is assumed that n_0 received bits are decoded at one time. However, decoding cannot wait until the entire syndrome is computed, as this could involve an infinite time delay. Therefore, the following assumption is made for decoding: to decode the first n_o bits, only the first N bits of the syndrome are used when N is, again, the number of the last row of sub-matrix B_o. The second set of n_o received bits are decoded on the basis of the first $N+i$ bits of the syndrome, etc. The first N bits of S are determined by the first N rows of H. Matrix H_N is defined as the first N rows and first $\frac{N}{i} n_o$ columns of H. S_N and e_N are the corresponding syndrome patterns and error patterns of dimension N and $\frac{N}{i} n_o$ respectively. Then

$$S_N = H_N \cdot e_N \tag{2.7}$$

Matrix H_N is shown schematically in figure 2.12 (i is assumed to be one).

Now we want to examine how S_N is formed when a burst error of length n_o or less occurs (most codes studied in this report are type B_1 or type B_2 with burst error correcting ability of $b \leq n_o$ called basic codes. Interleaving these basic codes will provide codes which can correct burst errors of length greater than n_o). For each syndrome bit calculated, n_o bits are received at the decoder. The error vector, e_N, as a function of time, typically appears as shown in figure 2.13. Each e_p, ($p=1,2 \ldots n_o$) can be either a "one" or a "zero".

The burst error patterns are the odd numbers from 1 to 2^{n_o}

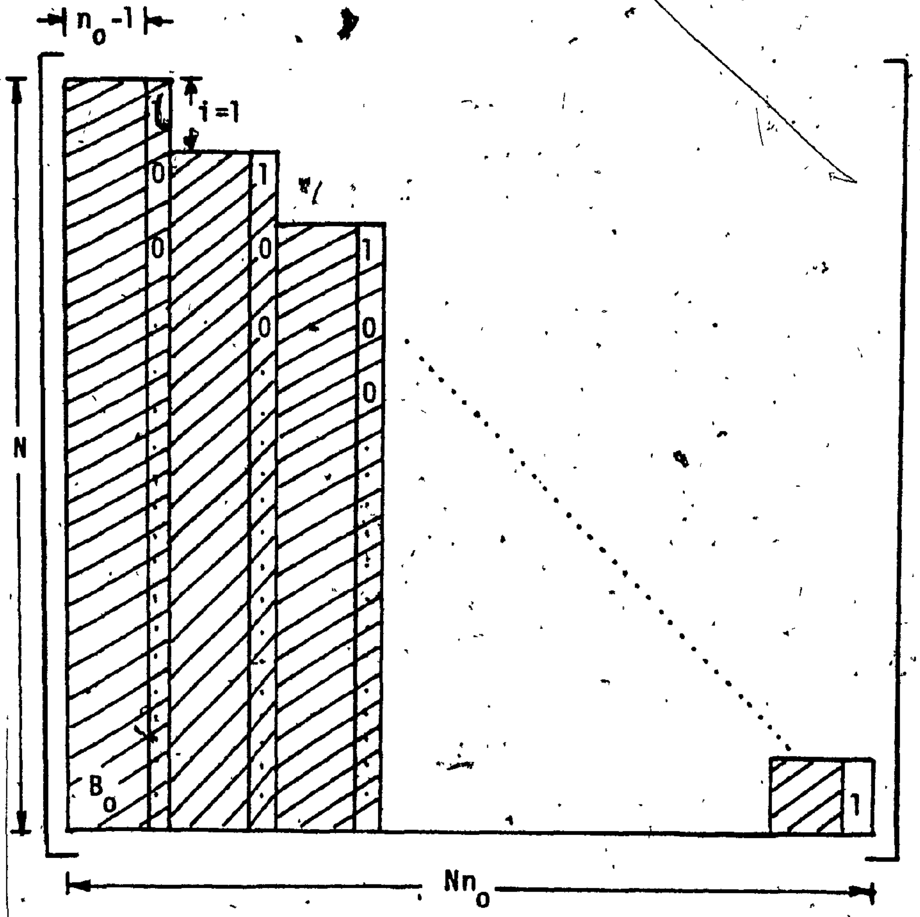


FIG. 2.12 Matrix H_N

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ e_1 \\ e_2 \\ \vdots \\ e_{n_0} \end{bmatrix} , \quad \begin{bmatrix} 0 \\ 0 \\ \vdots \\ e_1 \\ e_2 \\ \vdots \\ e_{n_0} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdots\cdots\cdots \begin{bmatrix} e_{j+1} \\ e_{j+2} \\ \vdots \\ e_{j+n_0} \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \quad \xrightarrow{\text{increasing time}}$$

$j=0$ or 1 or 2 or ... n_0-1

FIG. 2.13 Error Vector Appearances

written in binary (1, 11, 101, 1001, ...). It is clear that for a burst of length n_o there are 2^{n_o-2} burst patterns of length n_o and 2^{n_o-1} burst patterns of length n_o or less. For type B_1 codes the burst patterns of length n_o or less are $n_o 2^{n_o-1}$.

The syndrome S_N , as a function of time, is the combination of n_o or less columns of H_N (maximum burst length is equal to n_o digits). The first possible nonzero S_N is a combination of the rightmost n_o columns of H_N . The next S_N includes at least one column from columns Nn_o-2n_o to Nn_o-n_o plus possibly one to n_o-1 columns from the last n_o columns of H_N , etc. Finally, the syndrome includes at least one column from the first n_o columns and possibly 1 to n_o-1 columns from columns n_o+1 to $2n_o-1$. This last syndrome is referred to as the desired syndrome and is used to correct erroneous bits. The previous syndromes are referred to as undesired syndromes and must be distinct from the desired syndromes (all syndromes which include at least one column of the first n_o columns of H_N are desired syndromes). In general, all undesired syndromes for all bursts of length n_o or less must differ from all desired syndromes for bursts of length n_o or less. Also, all desired syndromes must be distinct in order to uniquely decode and correct erroneous bits.

The first n_o columns of H_N are called block "0" . Columns n_o+1 through $2n_o$ are called block "1" , etc. When the syndrome, S_N, contains at least one column from block "0" , the error is called a block "0" burst. Block "0" bursts are important because the decoder must make an immediate decision as to which bits are to be corrected (otherwise, the received error bits will leave the decoder without any correction). If the burst is not a block "0" burst, the decoder is not to correct any bits. Block "0" burst are used extensively in burst correction decoder schemes [15]. Any received error pattern, e_N, will eventually produce a block "0" burst. When this happens, the decoder must correct all errors which cause columns of block "0" to be in S_N. One thing of importance to note is the requirement of minimum "g" error free digits on either side of each n_0-bit burst errors known as guard space. This guarantees that each syndrome vector will be related to the corresponding n_0-bit burst errors. Otherwise, the decoder makes a number of erroneous decisions which are called error propagation. Error propagation is a function of both the code and decoder. For a decoder which resets syndromes to zero after each data bit correction there is no serious problem of error propagation.

2.7 Necessary and Sufficient Conditions on H_N for Burst Error Correction

With the following assumptions, and the theorem due to Wyner and Ash [4], we can find necessary and sufficient conditions on H_N.

Assumptions:

1) The parity check matrix H is of the form

$$H = [B, B_i, B_{2i} \ldots]$$

where B is a semi-infinite matrix with n_0 columns and B_{hi} is formed from B by shifting the rows of sub-matrix B down hi places. The first i rows of H are assumed to be linearly independent.

2) The first block of n_0 digits of received data, called zeroth block, is decoded on the basis of the first N digits of the syndrome, where N is a fixed positive integer (chosen so that "i" divides N). In general, the first "h" blocks are decoded on the basis of the first $N+(h-1)i$ digits of the syndrome $(h=1,2 \ldots)$.

3) The matrix H_N consists of the first N rows and $(\frac{N}{i})n_0$ columns of H. Alternately

$$H_N = [B_0, TB_0, T^2B_0 \ldots, T^{(\frac{N}{i})-1} B_0] \qquad (2.8)$$

where B_0 consists of the first N rows of B. Matrix T is shown in figure 2.14 (in figure 2.12, the matrix H_N is a special case of the above assumption where i is supposed to be one, and the last $N-1$ digits of column n_0 of sub-matrix B_0 are supposed to be zero).

<u>Theorem 1</u>: The matrix H_N defines a type Γ burst error correcting code $(\Gamma = B_1, B_2)$ with $n = (\frac{N}{i})n_0$, the number of columns in H_N, if and only if the following condition is satisfied.

Let $Z_1 = \sum_{f \in F} C(f)$, $Z_2 = \sum_{j \in J} C(j)$ be type Γ correctable linear combinations of columns of H_N; then $Z_1 = Z_2$ implies that $F \cap K_{n_0} = J \cap K_{n_0}$ (where $K_{n_0} = \{1,2, \ldots, n_0\}$ and "$\cap$" denotes set intersection). In other words, correctable error patterns which disagree in the zeroth block cannot yield the same truncated syndrome, S_N.

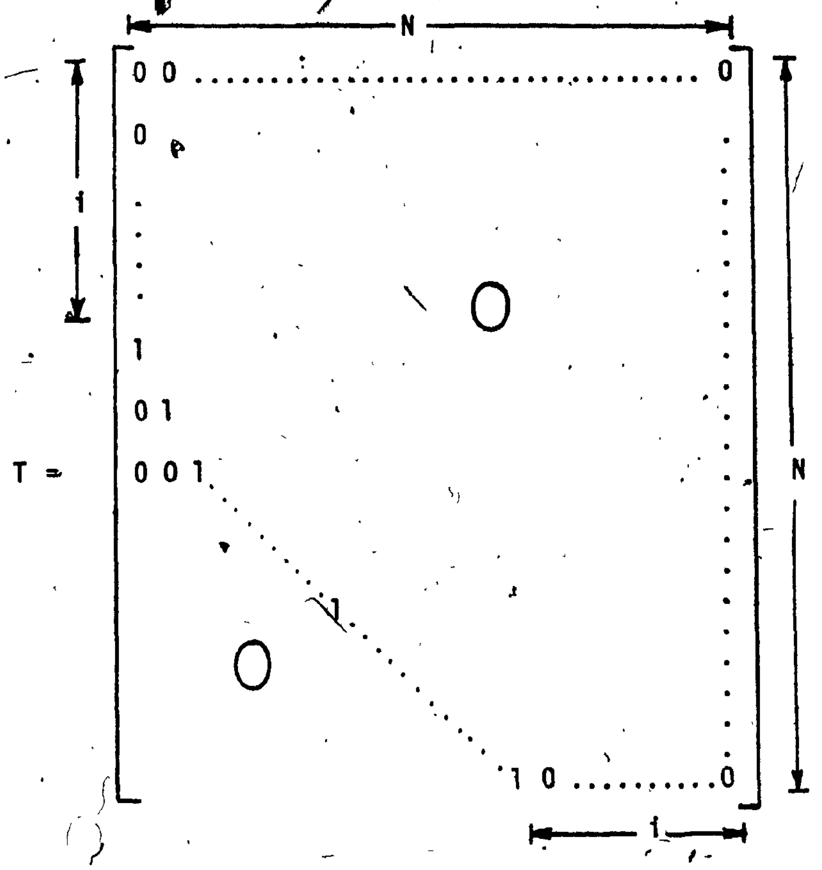


FIG. 2.14 Transform Matrix "T"

Proof:

a) Necessity - Assume we have constructed a type Γ code with [illegible]$(\frac{N}{1})n_0$. Suppose $Z_1 = Z_2$ and $F \cap K_{n_0} \neq J \cap K_{n_0}$. Say $F \cap K_{n_0}$ is not empty. Then Z_1 is the truncated syndrome S_N corresponding to a type Γ correctable error including digits in the zeroth block (i.e., those digits which belong to $F \cap K_{n_0}$). Since $Z_1 = Z_2$ and $F \cap K_{n_0} \neq J \cap K_{n_0}$, Z_1 is also the truncated syndrome corresponding to a type Γ correctable error, but with a different set of errors in the zeroth block. Thus, by looking at S_N , we cannot decode block "0" . This contradicts assumption 2 above. Consequently, $F \cap K_{n_0} = J \cap K_{n_0}$ if $Z_1 = Z_2$.

b) Sufficiency - Assume we have an H_N which satisfies the condition. If a type Γ correctable error occurs including errors in block "0" , then by hypothesis we can determine the errors in block "0" by looking at S_N (i.e., the hypothesis says that any $S_N = \sum_{f \in F} C(f)$ has a unique $F \cap K_{n_0}$) . Thus by examining S_N we can determine exactly which digits in block "0" are in error; subtract the columns of H corresponding to those errors (in block "0") from S (thus negating their effect on S), and proceed to decode block "1" as we did block "0" , etc.

Remark: $Z = \sum_{f \in F} C(f)$ is type Γ correctable if and only if F corresponds to a burst of length less than or equal to b .

2.8 Bounds on Burst-Error-Correcting Convolutional Codes

On a channel on which burst errors occur frequently, it is highly desirable to minimize the required guard space of a code for a fixed burst-error correcting ability "b" , since channel errors during this period may cause incorrect decoding.

With the definitions given for type B_1 and type B_2 codes, we easily recognize that for any (n,k) convolutional code the required guard space is $g = n-1$ digits if the code is of type B_1 , and $g = \frac{n}{n_0} -1$ blocks if the code is of type B_2 .

Minimization of "g" implies finding the minimum possible value of $n = \frac{N}{i} \cdot n_0$ for a fixed block length (n_0) , redundancy $(i = n_0-k_0)$ and burst length (b) .

2.8.1 <u>Wyner and Ash Bound</u>

Wyner and Ash [4] have found a lower bound on N for type B_2 codes. This bound is given as:

$$N \geq 2b - (r-1)i \tag{2.9}$$

Consequently

$$n \geq \frac{2n_0 b}{i} - (r-1)n_0 \tag{2.10}$$

Codes satisfying (2.10) with equality are called optimal codes. The proof of (2.10) is given in Appendix A.

2.8.2 <u>Gallager Bound</u>

The Gallager [16] bound is more general than the Wyner and Ash bound and states that in order to have a burst error correcting capability "b" for any type of coding (block, convolution or other) it is necessary to have

$$g/b \geq \frac{1+R}{1-R} \qquad (g \geq b\ , \quad R > 0) \tag{2.11}$$

where R is the code rate.

Massey [11] has obtained similar results and concludes that

$$g/b \geq \frac{R}{1-R} \tag{2.12}$$

Massey refers to his result as a bound on "almost all" burst-correction and to Gallager's bound as a bound on "complete" burst-correction. The above results were not published formally since the authors had been attributing it to each other. The proof of Gallager's bound is given in Appendix B.

2.9 Historical Notes and References

Elias [17] in 1955 was the first to propose convolutional codes for correction of random errors. Hagelbarger [18] in 1959 was the first to apply convolutional codes to burst-error correction. He also gave a simple decoder for implementing them. Kilmer [19], [20] in 1960 worked on both burst and random error correcting codes and laid the foundations for the basic work of Wyner and Ash [4]. Wyner and Ash in 1963 obtained a bound on minimum guard space between error bursts and provided a precise mathematical framework for algebraic convolutional codes. Berlekamp [20] in 1964 constructed a very useful class of burst-error-correcting codes. Preparata [21] did similar work independently of Berlekamp. Massey [15] in 1965 succeeded in finding a simple decoder for the Berlekamp-Preparata codes. These codes are known as Berlekamp-Preparata-Massey codes (BPM Codes). Massey and Kohlenberg [23], [24], [8] in 1964 introduced the diffuse code, which is good for both burst and random error correction. Gallager [25], [16] in 1965 discovered a code which is again good for both burst and random error correction. He also gave a bound for burst-error-correcting codes which is general and applicable to all kinds of codes. In 1968, Iwadare [26] discovered two new classes of codes; one having a shorter guard space than the BPM code. Hsu [27] in 1970 introduced a type B_1 code which requires shorter guard space and fewer shift register stages for implementation than Iwadare codes. Sullivan [28] in 1970 generalized Gallager's scheme. Freguson [29] in 1970 proposed a $\frac{1}{2}$ rate diffuse code with respect to the Massey-Kohlenbarg Code. In 1972, Mandelbaum [30] discovered some optimal type B_1 convolutional codes. Rodgers [5] in 1977 invented a class of codes which requires a guard space less than the best equivalent, non-interleaved Iwadare and Hsu codes. He also found some optimal type B_1 convolutional codes.

CHAPTER 3

BURST ERROR CORRECTING CONVOLUTIONAL CODES

In this Chapter we will study the well-known burst-error correcting convolutional codes which have the capability of correcting burst errors followed by "g" error free digits (guard space) for each burst occurrance (i.e. no error is allowed in guard space).

3.1 Hagelbarger Codes

The Hagelbarger code [18], [31] is a $(mn_0, m(n_0-1))$ code which has type B_1 burst error correcting capability. We follow Hagelbarger's example for a high redundant code (one parity digit for every data digit) which can correct sequences of errors up to $b = 6$ digits in length when there are at least $g = 19$ correct digits between the error-bursts. Figure 3.1 shows a block diagram of the encoder. The parity-digit generator is a modulo 2 adder (exclusive OR gate) for digits in positions "1" and "4" of the shift register and the multiplexer at the output end of encoder transmits this parity digit before the data digit in position "7" is shifted into the channel. Every data digit is subject to a parity check twice during its passage through the shift register. When the digits are transmitted over the channel the separation between any data digit and a parity digit related to it is greater than 6 digits; hence, burst errors of length 6 or less digits cannot cause any pair of such digits to be in error together. Table 3.1 shows sequences of the encoding process (all stages of the shift register have "0" digits at the beginning of encoding).

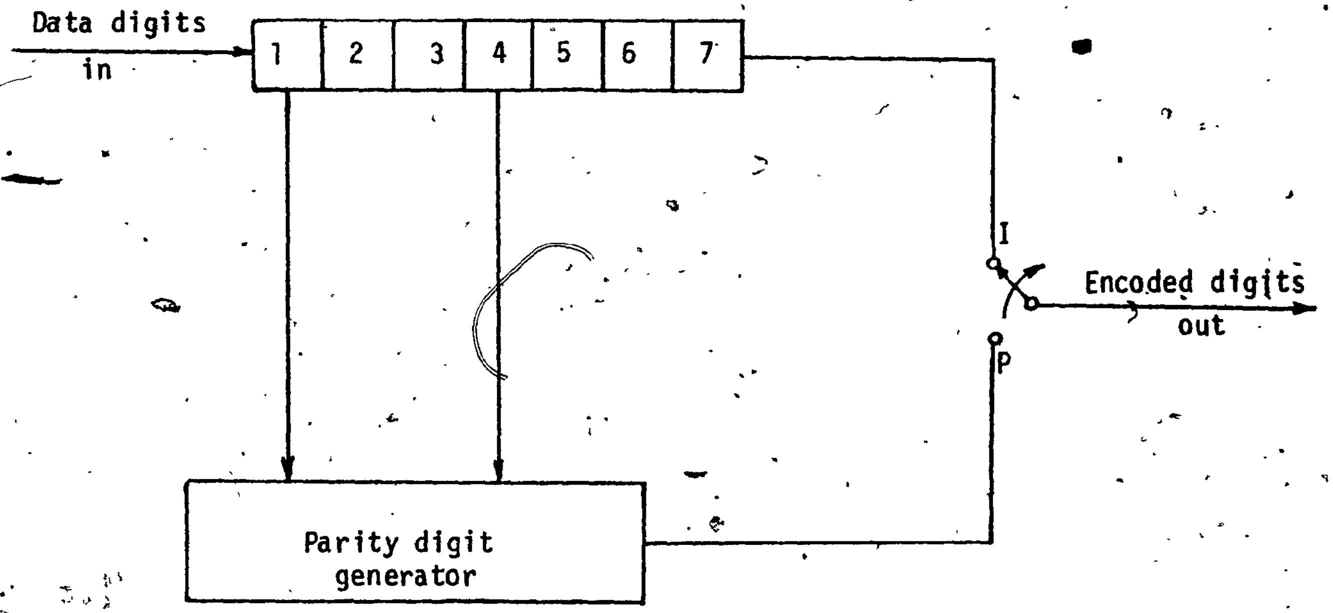


FIG. 3.1 Hagelbarger Encoder for 6-digit Burst Error Correction

TABLE 3.1

SEQUENCES OF THE HAGELBARGER ENCODING PROCESS

Time	Data Digit	Shift Register Content	Parity Digit	Encoded Data Stream
0	-	0000000	-	-
1	0	0000000	0	-
2	1	1000000	1	00
3	1	1100000	1	0100
4	0	0110000	0	010100
5	1	1011000	0	00010100
6	0	0101100	1	0000010100
7	1	1010110	1	010000010100
8	0	0101011	1	$\overset{**}{0}10100000\overset{*}{0}10\overset{*}{0}0$
↓	↓	↓	↓	

* Parity digits related to the first data digit

** First data digit in encoded data stream.

The principle used in decoding received digits is to examine the two parity digits covering a data digit simultaneously. That is, the received encoded stream will be switched either to the data shift register or to the parity check shift register in appropriate sequences and the data digits will be re-encoded in order to constitute new parity digits for comparison with the previous related parity digits. Figure 3.2 illustrates the decoder for 6-digit burst error correction.

Considering table 3.1 and figure 3.2 we notice that when the first data digit is in the fourth stage of data shift register, the first parity digit related to this data digit is in the tenth stage of parity shift register. The second parity digit related to this data digit is in the seventh stage of parity shift register. Stages 4 to 7 of the data shift register resemble encoding of the first data digit exactly as when this data digit was in the first stage of data shift register of the encoder (first parity digit, Fig. 3.1). Stages 1 to 4 of the data shift register of the decoder resemble encoding of the first data digit exactly as when this data was in the fourth stage of shift register of the encoder (second parity digit, Fig. 3.1).

Now if the outputs of both parity checks R and S are "1", we conclude that the data digit in the fourth stage of data shift register of decoder is in error. At this point the AND gate will operate, allowing corrected data to be sent out and to be stored in the fifth stage of data shift register. This is clear because when the data digit in the fourth stage of shift register is in error, the digits in the first stage, the seventh stage, and the tenth stage of parity

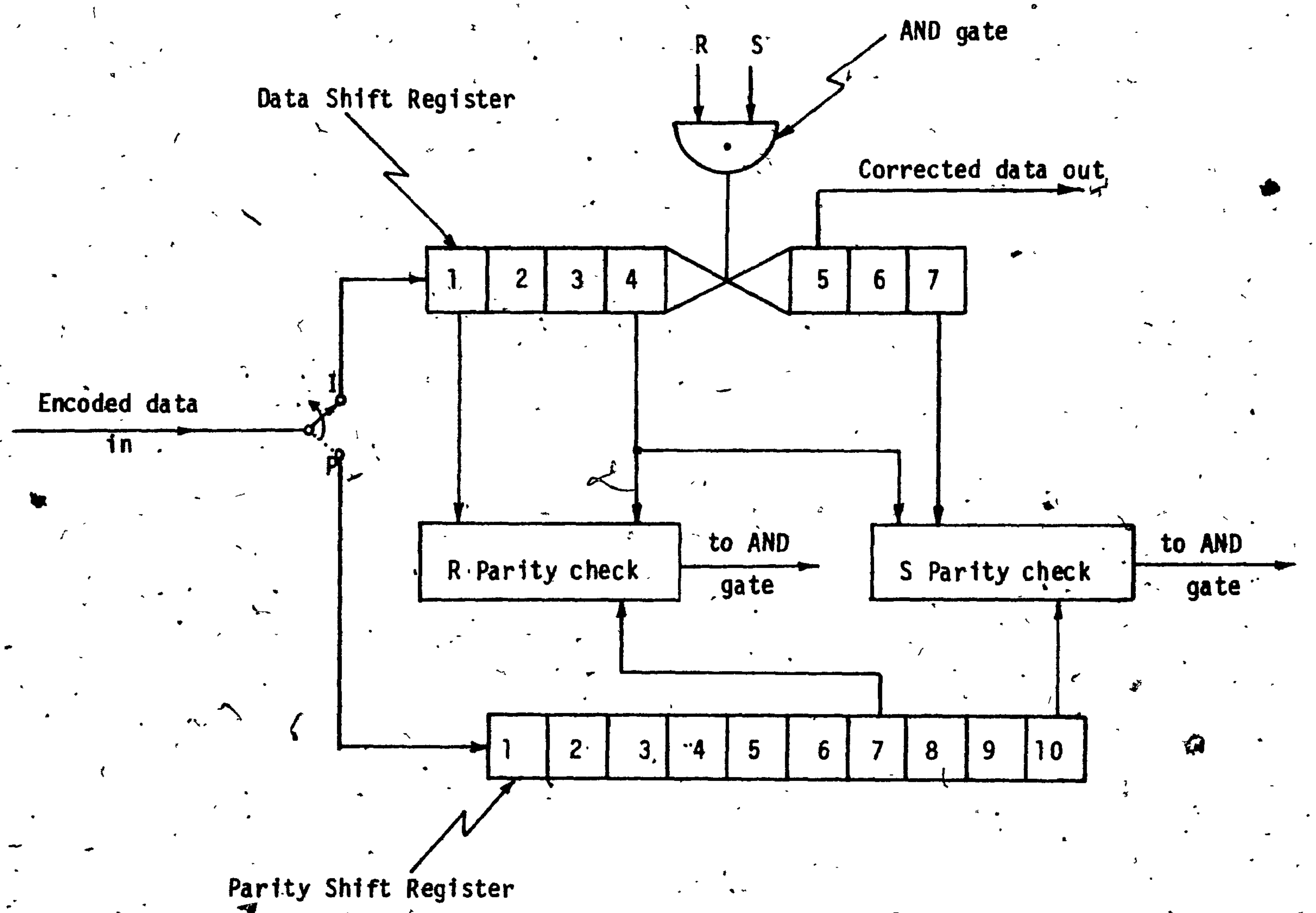


FIG. 3.2 Hagelbarger Decoder for 6-Digit Burst Error Correction

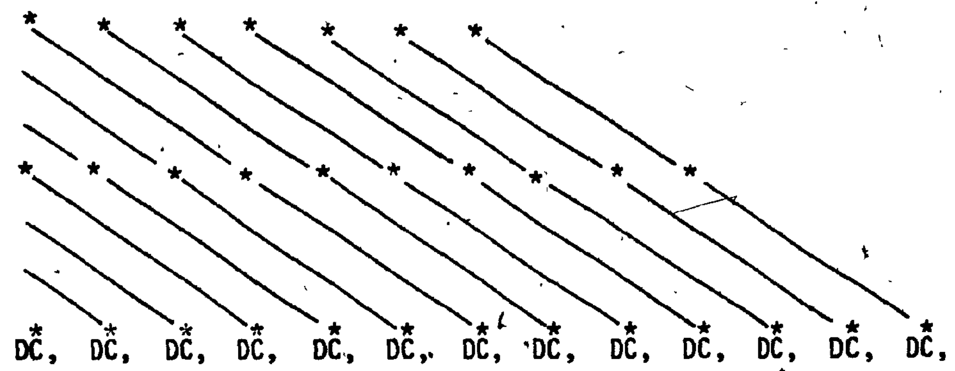
shift register cannot be in error since they are at least 6 digits apart from the digit in the fourth stage of data shift register. The digit in the seventh stage of data shift register is always a correct digit. Thus, R and S are both "1" because of their common connection to the fourth stage of data shift register which contains error data digit. In the case in which either S or R is "1" we conclude that either one of the parity digits or another information digit in the first stage of data shift register is in error. We do not have any interest in correcting either parity digits. If the data digit is in error, it will be corrected when it leaves the fourth stage of the data shift register. This decoder corrects data digits which are confined to six or less error-burst of the encoded data stream when at least 19 error free digits are available around each burst. Table 3.2 shows typical values of burst length, data shift register length, parity shift register length and guard space length of this type of Hagelbarger code.

In general, Hagelbarger has encoded the data digits so that there is one parity digit in each n_0-bit block. Since the block must have at least one data digit, the shortest block length is $n_0 = 2$ (this value is used in the example above). The data digits are loaded into the data digit positions in the order in which they are received. The parity digit is determined by a parity relation applied once for each block. This parity relation extends over a selected set of the digits in m consecutive blocks. Figure 3.3 shows a portion of the encoded data from the above example. The data and parity digits are indicated by D's and C's ; the blocks are marked off with commas and the parity relation is shown by lines having $\star$'s over the digits

TABLE 3.2

TYPICAL VALUES FOR HIGH REDUNDANT HAGELBARGER CODES

Burst Length	Data Shift Register Length	Parity Shift Register Length	Guard Space Length
2K	2K+1	3K+1	6K+1
4	5	7	13
6	7	10	19
8	9	13	25
10	11	16	31
.	.	.	.
.	.	.	.
.	.	.	.


$\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$, $\overset{*}{D}\overset{*}{C}$,

$$\begin{array}{l|ccccccc|} P_1 = P_D & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ P_2 = P_C & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

FIG. 3.3 Portion of Encoded Data Using Hagelbarger Encoder

in a given parity group. Note that m for this example is seven; that is, any one of the parity groups extends over seven blocks. Every parity group has three digits in it, two data and one parity; thus, each data digit is in two parity groups and each parity digit is in only one parity group.

We will denote which digits enter into the parity relation by n_0 binary words of m digits each. We call these the parity words and label them $P_1, P_2, \ldots, P_{n_0}$. Consider m consecutive blocks. We form P_1 by observing the first position of each block; if the digit is in this parity group we write "1", otherwise we write "0". P_2 depends on the second positions of these m blocks, and so on. In figure 3.3 we have shown one parity group by the arrow and written the parity words so that the digits fall under the corresponding blocks. Thus, P_1 has "1"'s in the first and fourth blocks and P_2 has a "1" only in the seventh block (the numbering of digits and blocks here and in figure 3.3 is from left to right). With this notation, there is a simple correspondence between the "1"'s in the parity words and the connections to the stages of shift register of the encoder and decoder.

At the decoder we have a circuit for checking the parity relation imposed by the encoder. This circuit gives an output once for each block received. If the parity check fails, the output is a "1"; otherwise, it is a "0". When a burst of errors does occur, the check circuit will give a pattern of "1"'s and "0"'s. This pattern which is known as syndrome will be used to identify the burst. It is obvious that syndromes are those binary words having "1"'s on both ends (odd numbers). The first problem is to choose the parity relation in such a fashion that each burst of length "b" or less has a distinct syndrome. A further problem is to choose the parity such that, given a distinct syndrome, the correction of the burst which caused it is easily mechanized. That is, we want a systematic scheme for correcting a burst, given the corresponding syndrome which is much better than having a table of all possible syndromes with corresponding bursts. The procedure for calculating the syndrome corresponding to a given burst is mentioned below in which digits with subscript are digits in error

$$\cdots\; Dc,\ DC,\ DC_3,\ D_2C,\ D_1C_1,\ D_0C,\ DC,\ DC,\ \ldots\ldots$$

$$\begin{array}{lcccccccccc}
P_D^0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & & & \\
P_C^1 & & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & \\
P_D^1 & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & & \\
P_D^2 & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \\
P_C^3 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
\text{Syndrome:} & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1
\end{array}$$

We number the blocks, starting with "0" for the first block containing an error, and continue the numbering far enough to include all blocks having errors in the burst under consideration (the direction of numbering should be such that increasing numbers represent digits received at later times). Then we write the parity word for the error in block number "0". Under this we write the parity words for any other errors in this block. Now we write the parity words for the other errors, shifting each one (in the direction of increasing time) the number of places equal to the block number in which the error occurs. The syndrome is the sum modulo two of these parity words. Note that there is never more than a single "1" in any column.

By spreading out the parity words with "0"'s so as to avoid interactions between errors in a burst, Hagelbarger has introduced a new code called lower redundancy code. For this code the redundancy can be made as low as is desired while maintaining a relatively simple correcting mechanism. Again, for this code we have one parity digit in each n_0-bit block.

Suppose we desire a code with a redundancy of one-quarter good for bursts of length four or less. We choose the following parity words (the digits are spaced to emphasize the method of forming and the choice of parity words will become clear later).

P_A	1 1 1	0 0 0	0 0 0	0
P_B	0 0 0	1 0 1	0 0 0	0
P_C	0 0 0	0 0 0	1 1 0	0
P_D	0 0 0	0 0 0	0 0 0	1

Note that placing the code 100 to the extreme right allows us to shorten the parity words by dropping the last two columns, all of which were "0"'s . In assigning the parity words, the groups of "1"'s should be arranged to go from the upper left to the lower right as shown above. That is, the order of the groups of "1"'s in the parity words should agree with the order of the digits in the block structure. If this is not done, then extra columns of "0"'s must be inserted in the array of parity words, which means the addition of more shift register stages in the encoder and decoder. The difficulty is shown as follows: If the parity words are

$$\begin{array}{lllll} P_A & 1\,1\,1 & 0\,0\,0 & 0\,0\,0 & 0 \\ P_B & 0\,0\,0 & 1\,0\,1 & 0\,0\,0 & 0 \end{array}$$

and the burst is

$$\ldots\ldots A\ \underbrace{B_1\ C\ D\ A_0}\ B\ \ldots\ldots\ldots$$

then the syndrome will be

$$\begin{array}{ll} P_A^0 & 1\,1\,1\,0\,0\,0\,0\,0\,0\,0 \\ P_B^1 & \quad 0\,0\,0\,1\,0\,1\,0\,0\,0\,0 \\ \hline \end{array}$$

Syndrome: $1\,1\,1\,0\,1\,0\,1$ which is acceptable, but if we choose parity words such as

$$\begin{array}{lllll} P_A & 0\,0\,0 & 1\,1\,1 & 0\,0\,0 & 0 \\ P_B & 1\,0\,1 & 0\,0\,0 & 0\,0\,0 & 0 \end{array}$$

and have the same burst as above, then the syndrome will be

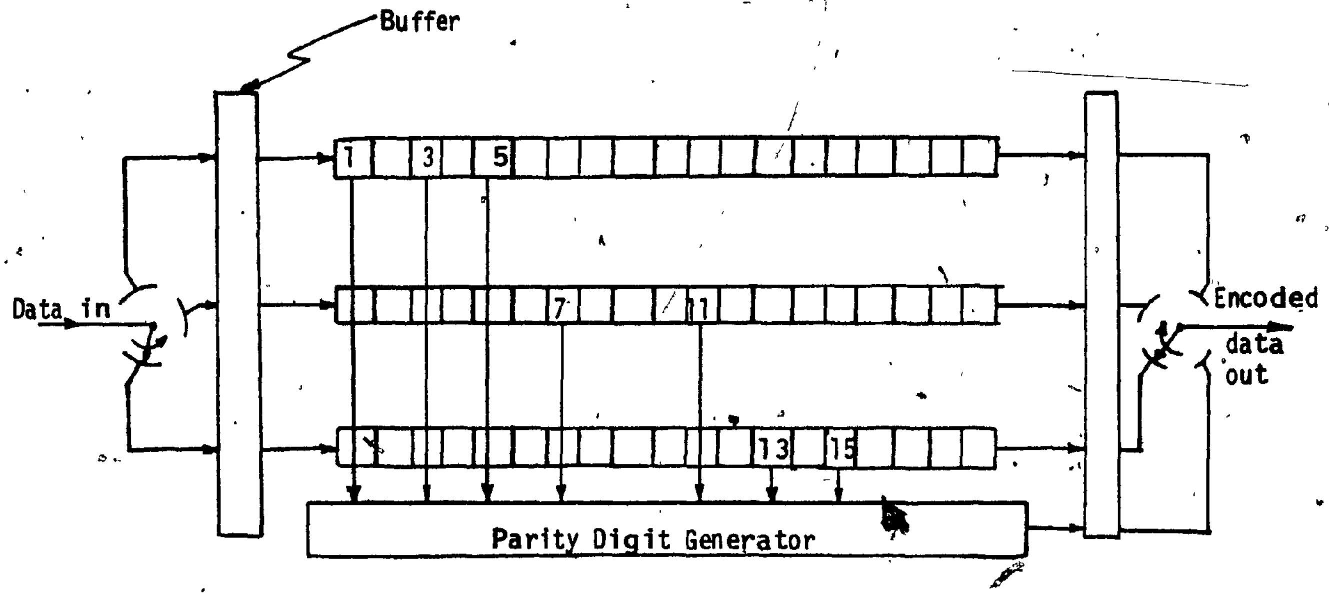
$$\begin{array}{lllll}
P_A^0 & 0\,0\,0 & 1\,1\,1 & 0\,0\,0 & 0 \\
P_B^1 & \;\;1\,0 & 1\,0\,0 & 0\,0\,0 & 0 \\
\hline
\text{Syndrome:} & \;\;1\,0 & 0\,1\,1 & &
\end{array}$$

which is not good.

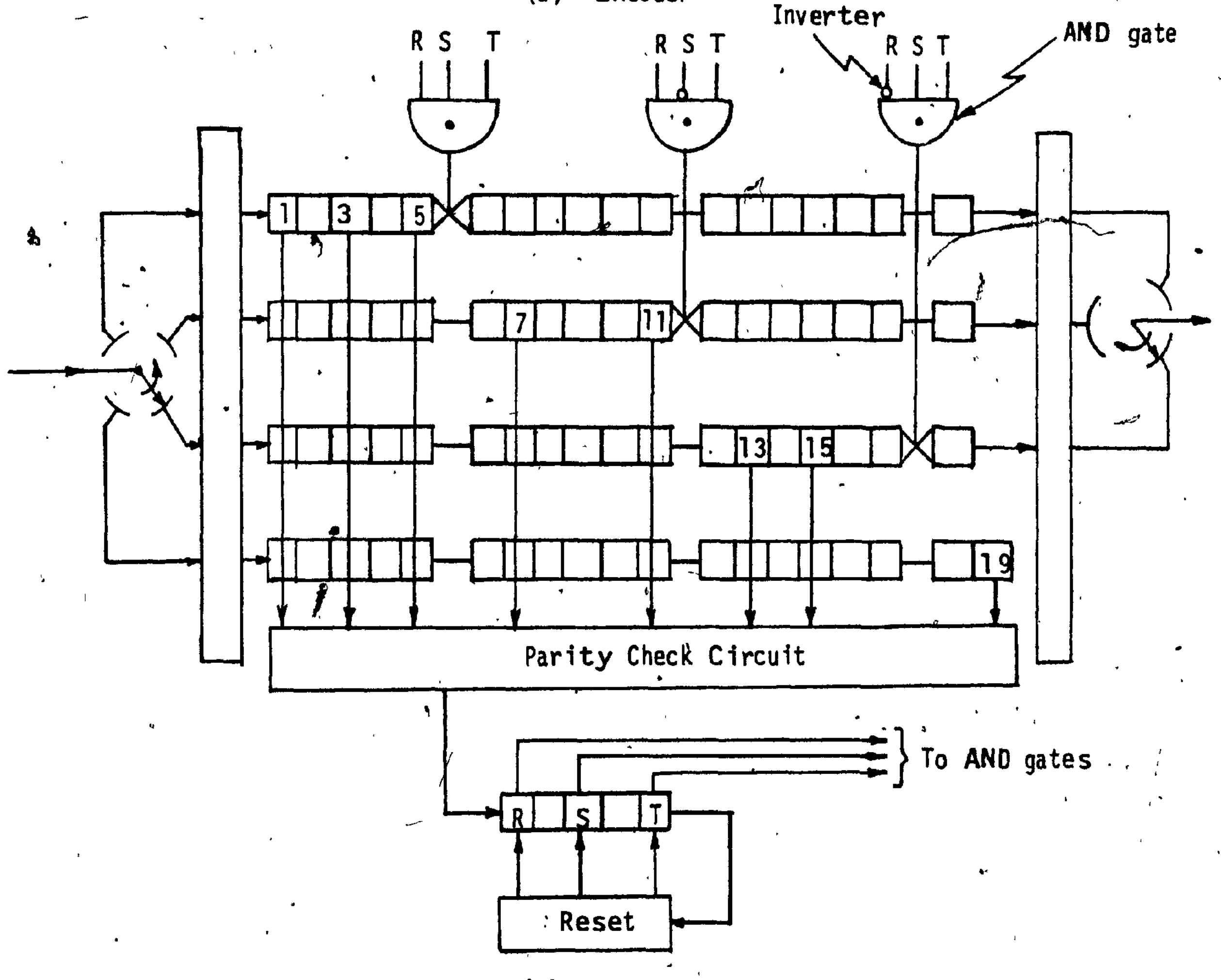
(the burst A_1 B C D A B_0..is not allowed since the above code is for bursts of length four or less). If we want to make a code for the same redundancy good for bursts of length of eight or less, we form the parity words by inserting "0"'s between each of the digits of the above code. Thus we will have

$$\begin{array}{lllll}
P_A & 1\,0\,1\,0\,1 & 0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0 & 0\,0 \\
P_B & 0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,1 & 0\,0\,0\,0\,0\,0 & 0\,0 \\
P_C & 0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0 & 0\,1\,0\,1\,0\,0 & 0\,0 \\
P_D & 0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0 & 0\,0\,0\,0\,0\,0 & 0\,1
\end{array}$$

The related encoder and decoder for this code are illustrated in figure 3.4. Note the correspondence between the parity words and the shift registers. The top register comes from P_A , the second from P_B , and so on. This system can correct error bursts of up to eight digits in length when at least 91 clear digits are available on either side of each burst. The input digits are switched sequentially to three data shift registers of 19 stages each. Parity digits are generated on positions 1, 3 and 5 of the first register, 7 and 11 of the second register, and 13 and 15



(a) Encoder



(b) Decoder

FIG. 3.4 Hagelbarger a) encoder, b) decoder for $1/n_0 = 1/4$

of the third register. The parity digits and the data digits are re-multiplexed into a time sequence before being transmitted over the channel. Buffers are used to store digits temporarily to allow all the shift registers to be stepped together.

At the decoder, the data digits are routed into three registers corresponding to the ones they occupied at the encoder. The parity digits are sent into a separate register (synchronization must be maintained so that the digits enter into proper registers). The parity relationships are checked and, if a check fails, a "1" is put into the syndrome register which is stepped at the same rate as the other registers. There is no delay in feeding digits to this register. Thus the digit in position R at any time depends on the parity which is being carried out at this time.

Suppose an error digit enters into the first data register. Accordingly, the parity checks fail when this digit leaves positions 1, 3 and 5 and cause "1"'s to be sent into the syndrome register. When the error data leaves position 5, the output of positions R,S and T of the syndrome will all be "1"'s. At this moment the AND gate will operate and the error digit will be corrected. Errors in the second data register cause R and T to be "1"'s and S to be "0" . The error data will be corrected when it leaves position 11 of the second data shift register. Errors in the third data register cause R and S to be "1"'s and T to be "0" . The error data will be corrected when it leaves position 17 of the third data shift register. Notice that the correction for this case will be carried out when the error data leaves position 17 and not position 15. The reason for this is that after each correction the syndrome will be reset to zero through the reset circuit. This circuit activates when a "1" leaves position T of the syndrome. If the correction is to be carried out when the error data leaves position 15, then the reset circuit will not operate and we will have error propagation. In general, a procedure for making a code of redundancy $1/n_0$ is as follows:

Take the first n_0 odd binary numbers and let L be the number of digits in the largest one. Then form each of these numbers to a L-digit word by adding "0"'s to the right end. As well as a square array with n_0 rows and n_0 columns is formed. The entries in this array are L-digit words that will be placed along the main diagonal with the word having a single "1" going in the lower right corner. The order of the other words on the diagonal is arbitrary. All remaining words will be filled with "0"'s . The new array will be a $n_0 \times n_0$ (L) array. We also strike out the last L-1 columns from the right so as to have a $n_0' \times [(n_0-1)L+1]$ array. The rows of this array are the parity words of the desired code. This code corrects all bursts of length n_0 or less. In order to make this code good for bursts of length Kn_0 or less, K-1 "0"'s are ~~added between~~ each adjacent pair of digits of the above parity words. If the odd binary numbers are not increased to L digits as above then for some allowable bursts the resulting syndromes would be incorrectly interpreted by the decoder. The above method is regular enough to enable us to find formulas for the shift register stages and guard space.

Definitions:

n_o, b, K, $L(n_o)$ are positive integers. The block length is n_o with one parity digit; hence the redundancy is $1/n_o$. This code can be used against burst of length "b" or less, where $b=Kn_o$. $L(n_o)$ is the smallest integer such that:

$$L(n_o) \geq 1 + \log_2 n_o$$

Thus

n_o	2	3	4	5	6	7	8	9	10
$L(n_o)$	2	3	3	4	4	4	4	5	5

The encoder will have n_o-1 shift registers, each of length $(\frac{b}{n_o})(n_o-1)L(n_o) + 1$. This is true because each parity word has $n_oL(n_o) - L(n_o) + 1$ digits in original form. When adding $K-1$ zero digits between each pair of original parity word, the new parity word will have:

$$(K-1)[n_oL(n_o)-L(n_o)+1-1] + n_oL(n_o)-L(n_o)+1$$

or $K(n_o-1)L(n_o) + 1$ digits. When substituting for K the $K = \frac{b}{n_o}$ the above formula is found. Since there are n_o-1 shift registers in the encoder, the total amount of shift register stages is:

$$\frac{b}{n_o}(n_o-1)^2 L(n_o) + n_o-1$$

The decoder has $b(n_o-1)L(n_o) + 2n_o + (\frac{b}{n_o})L(n_o)-b$ stages.

This is obvious since the decoder is comprised of the following circuits:

1) A circuit similar to the encoder but with a difference of $(n_0-1)(K-1)$ stages of shift register less than the related encoder. This ~~is~~ equal to

$$(n_0-1)^2(\frac{b}{n_0})L(n_0) + n_0-1-(n_0-1)(K-1) \text{ stages.}$$

2) A parity check register with the number of shift register stages equal to a parity word

$$(\frac{b}{n_0})(n_0-1)\, L(n_0) + 1$$

3) A syndrome register having the number of shift register stages equal to

$$(L(n_0)-1)\,(K-1) + L(n_0)$$

or $$L(n_0)K-K+1$$

By summing the quantities indicated in 1, 2 and 3 above, the formula for calculating the decoder's shift register stages, is found. The required guard space is

$$n_0bL(n_0) + n_0-b-1$$

Again this is clear since it is required that

$$n_0[\frac{b}{n_0}L(n_0)(n_0-1) + 1]$$

or $\quad bL(n_o)(n_o-1) + n_o$

error free digits enter the decoder to refill all the stages of shift registers except the syndrome register. Also, an extra $n_o[L(n_o)K-K+1-1]$ or $bL(n_o)-b$ error-free digits must enter the decoder to refill the syndrome register. The addition of these two required error-free digits minus one will give the required guard space as indicated above.

Table 3.3 shows some typical values of burst length, data shift register length, parity shift register length and guard space length of lower redundancy Hagelbarger codes.

3.2 Berlekamp-Preparata-Massey (BPM) Code

The basic BPM code [21], [22], [15], [12], [32] is a $(2n_o(n_o), 2n_o(n_o-1))$ convolutional code with rate $R = \frac{n_o-1}{n_o}$. It has type B_2 burst error correcting ability when $g = 2n_o^2 - n_o$ error free digits are available on either side of the burst (in general, for type B_2 codes the burst errors must be confined to "r" consecutive blocks of n_o-bit each, for basic codes of this type $(r=1)$ burst errors are confined to one block). The parity check matrix of this code is of the form:

$$H_N = [B_o, B_1, B_2 \ldots B_i \ldots B_{2n_o-1}] \tag{3.1}$$

where B_i is "i" down-shifted truncated version of B_o. Since this is a type B_2 code, $b=n_o$ bit errors cannot be confined to two consecutive blocks, such as the "0"th block and the first block. An n-tuple error pattern which has errors of ("1"'s) in the "0"th

TABLE 3.3

TYPICAL VALUES FOR LOW REDUNDANT HAGELBARGER CODES

Block Length	Code Rate	Burst Length	Shift Register Stages		Guard Space
n_o	R	b	Encoder	Decoder	g
2	1/2	2	3	8	7
		4	5	12	13
		6	7	16	19
		8	9	20	25
		10	11	24	31
3	2/3	3	14	24	26
		6	26	42	50
		9	38	60	74
4	3/4	4	30	43	47
		8	57	78	91
5	4/5	5	68	89	99
		10	132	168	194

block and "i"th block can be shown as

$$E^T = [E_0^T \, 00 \ldots \; E_i^T \, 00 \ldots \; 0]$$

where $E_0 \neq 0$. From previous discussion, if B_0 can be chosen such that $S = H_N E \neq 0$ for all choices of $E_0 \neq 0$, E_i and i , then the code has the capability of correcting $b \leq n_0$- digit burst-errors. For this to happen we must have:

$$[B_0 \; B_i] \begin{bmatrix} E_0 \\ E_i \end{bmatrix} \neq 0 \qquad 1 \leq i \leq 2n_0-1 \qquad (3.2)$$

Now consider two cases:

Case 1: When $n_0 \leq i \leq 2n_0-1$, the upper right quadrant of $[B_0 B_i]$ is the all-zero matrix of order n_0 . Choosing the upper half of B_0 as any nonsingular matrix guarantees that Eq. 3.2 holds (if the upper part of matrix B_0 is singular, it is still possible for some of E_0 error patterns to follow Eq. 3.2, but there is no guarantee that this will happen. This can be clarified by supposing that the upper part of B_0 is $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and E_0 is $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$. For this case $S \neq 0$ but for E_0 as $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ the upper part of $S = 0$ and lower part of S depends on the lower part of B_0 and E_i) . To simplify the decoding, the upper part of matrix B_0 is chosen to be the identity matrix of order n_0 , denoted I_{n_0} .

Case 2: When $1 \leq i \leq n_0-1$, the equation 3.2 holds provided that the matrices $[B_0B_1]$, $[B_0B_2]$, ..., $[B_0,B_{n_0-1}]$ are simultaneously nonsingular. Elementary row operations can transform $[B_0B_i]$ to the form:

$$[B_0B_i] \equiv \left[\begin{array}{c:c} I_{n_0} & X_i \\ \hdashline 0 & Y_i \end{array}\right] \qquad (3.3)$$

and so $[B_0B_i]$ is nonsingular if and only if the n_0- by -n_0 matrix Y_i is nonsingular.

Berlekamp [21] has initially chosen the lower half of B_0 as

$$\begin{bmatrix} 0 & \ldots\ldots & & & 0 & 0 & 0 \\ 0 & \ldots\ldots & & & 0 & 0 & A \\ 0 & \ldots\ldots & & & 0 & C & B \\ 0 & \ldots\ldots & & 0 & F & E & D \\ 0 & \ldots & 0 & J & I & H & G \\ \vdots & & & & & & \vdots \\ 0 & & & & & & \end{bmatrix}$$

in which blank spaces in the matrix are zeros. The letters represent binary variables that will be specified to make Eq. 3.2 hold. Matrix Y_i can be constructed in terms of these binary variables. As an example for $n_0=3$ we follow the Berlekamp method and find Y_1 and Y_2 relating to $[B_0B_1]$ and $[B_0B_2]$ respectively. We have:

$$B_o = \begin{bmatrix} 1&0&0\\ 0&1&0\\ 0&0&1\\ 0&0&0\\ 0&0&A\\ 0&C&B \end{bmatrix}, \quad [B_oB_1] = \begin{bmatrix} 1&0&0&0&0&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&0\\ 0&0&0&0&0&1\\ 0&0&A&0&0&0\\ 0&C&B&0&0&A \end{bmatrix}$$

Now we apply elementary transformation to matrix $[B_oB_1]$ (we change column 3 with column 5, column 2 with column 4). The result is:

$$[B_oB_1] \equiv \begin{bmatrix} 1&0&0&0&0&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&0\\ 0&0&0&0&0&1\\ 0&0&0&0&A&0\\ 0&0&0&C&B&A \end{bmatrix} \equiv \left[\begin{array}{c:c} I_{n_o} & X_1 \\ \hdashline O & Y_1 \end{array}\right]$$

where $Y_1 = \begin{bmatrix} 0&0&1\\ 0&A&0\\ C&B&A \end{bmatrix}$. For $[B_oB_2]$ we have:

$$[B_oB_2] = \begin{bmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&1&0&0\\ 0&0&0&0&1&0\\ 0&0&A&0&0&1\\ 0&C&B&0&0&0 \end{bmatrix}$$

Applying elementary transformation (changing column 3 with column 4, multiplying row 2 by $-c$ and adding the result to row 6) we will find:

$$[B_oB_2] \equiv \begin{bmatrix} 1\,0\,0 & 0\,0\,0 \\ 0\,1\,0 & 0\,0\,0 \\ 0\,0\,1 & 1\,0\,0 \\ 0\,0\,0 & 0\,1\,0 \\ 0\,0\,0 & A\,0\,1 \\ 0\,0\,0 & B\,0\,0 \end{bmatrix} \equiv \left[\begin{array}{c:c} I_{n_o} & X_2 \\ \hdashline O & Y_2 \end{array}\right]$$

where $Y_2 = \begin{bmatrix} 0 & 1 & 0 \\ A & 0 & 1 \\ B & 0 & 0 \end{bmatrix}$

It is possible to choose the binary variables A, B and C in such a way as to make both matrices Y_1 and Y_2 nonsingular. Choose the elements A, B and C successively in alphabetical order in such a way as to make all the square submatrices, whose upper right corner coincides with the upper right corner of Y_1 and Y_2, nonsingular. From matrix Y_1 we conclude that submatrix $\begin{bmatrix} 0 & 1 \\ A & 0 \end{bmatrix}$ is nonsingular when $A = 1$. Matrix Y_2 will be nonsingular when $B=1$ (A has already been replaced with "1"). Matrix Y_1 is nonsingular when $C=1$ (A and B have already been replaced with "1"'s).

In general the matrix B_o for BPM code is specified for all values of n_o as follows:

$$B_o = \begin{bmatrix} 0 & \cdots & \overset{I_{n_o}}{\cdots} & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & 1 \\ \vdots & \emptyset & 0 & 1 & 1 & 1 \\ \vdots & 0 & 1 & 0 & 1 & 1 \\ \vdots & \cdot^{\cdot^{\cdot}} & O & \vdots & \vdots & \vdots \\ 1 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix} \qquad (3.4)$$

Note that with this choice for matrix B_0 the BPM codes are systematic, but the parity digit in each block is transmitted before the data digits of the block. The usual encoding and syndrome calculating circuits must be modified to take this difference into account. As an example a 5-stage encoder for (18,12) BPM code is illustrated in figure 3.5.

The basic BPM codes can be decoded as follows. Suppose that a burst has occurred in the "0"th block. Since the upper half of the matrix B_0 was chosen to be the identity matrix, the first n_0 bits of the syndrome are identical to the error pattern that was added to the "0"th block. The only additional information required to decode a block is to check whether the burst occurred in the "0"th block or elsewhere. If the burst occurred in the "0"th block, the second half of the syndrome must be:

$$S_2 = B_{o2} E_o \tag{3.5}$$

where E_o denotes the burst, B_{o2} denotes the lower half of the matrix B_o and S_2 is the second half of the syndrome. In other words, since the first half of the syndrome S_1 is identical to E_o, if

$$S_2 + B_{o2}S_1 = 0 \tag{3.6}$$

the burst occurred in the "0"th block; otherwise it occurred elsewhere. Eq. 3.6 represents n_o linear equations involving the syndrome digits. A single multiple-input modulo-2 adder is all that is required to implement each equation. One n_o-input OR gate is sufficient to determine whether or not all of the equations are satisfied. As an example consider the

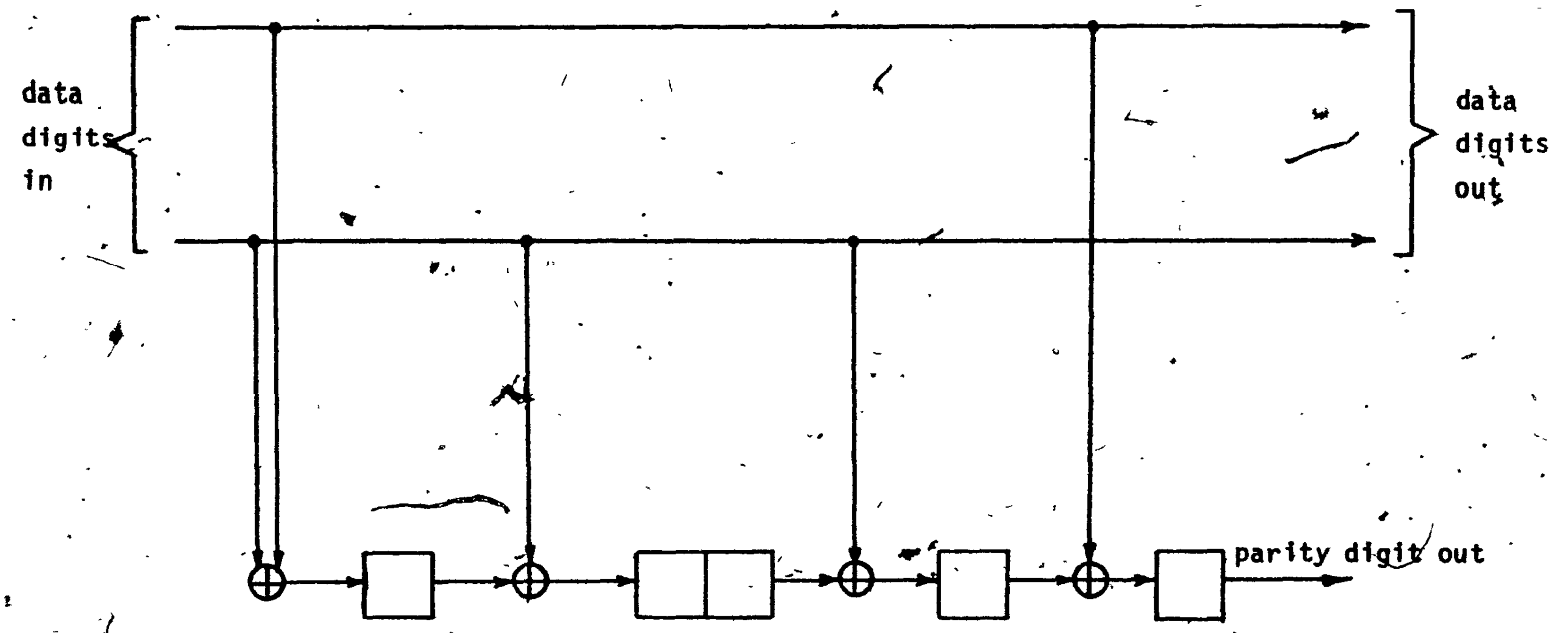


FIG. 3.5 5-Stage Encoder for (18, 12) BPM Code

(18,12) BPM code with the assumption that the burst occurred in the "0"th block. The related linear equations with respect to Eq. 3.6 are:

$$-\begin{bmatrix} S_{21} \\ S_{22} \\ S_{23} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} S_{11} \\ S_{12} \\ S_{13} \end{bmatrix} = 0 \qquad (3.7)$$

Solving (3.7), we find:

$$S_{21} = 0 , \quad S_{22} = S_{13} \text{ and } S_{23} = S_{12} + S_{13} \qquad (3.8)$$

where S_{ij} represents the jth digit of the ith half of the syndrome.

The decoder for this code is shown in figure 3.6. It functions as follows. The syndrome is calculated as usual and stored in the 6-stage syndrome register. If the relationships in Eq. 3.8 hold, the output of the inverter is a "1" , otherwise it is a "0" . If it is a "1" and if all the errors affecting the digits in the decoder are confined to a single 3-bit block, these errors occurred in the "0"th block. Then proper AND gates will operate and the correction will be carried out. Errors in the other blocks having the proper guard space will be corrected as "0"th block. After each correction the relating syndrome will be set to zero; consequently, errors cannot propagate.

3.3 Iwadare Codes

The basic Iwadare code [26], [32], [33], is a $(mn_0, m(n_0-1))$ convolutional code with rate $R = \frac{n_0-1}{n_0}$ and has type B_1 burst-error

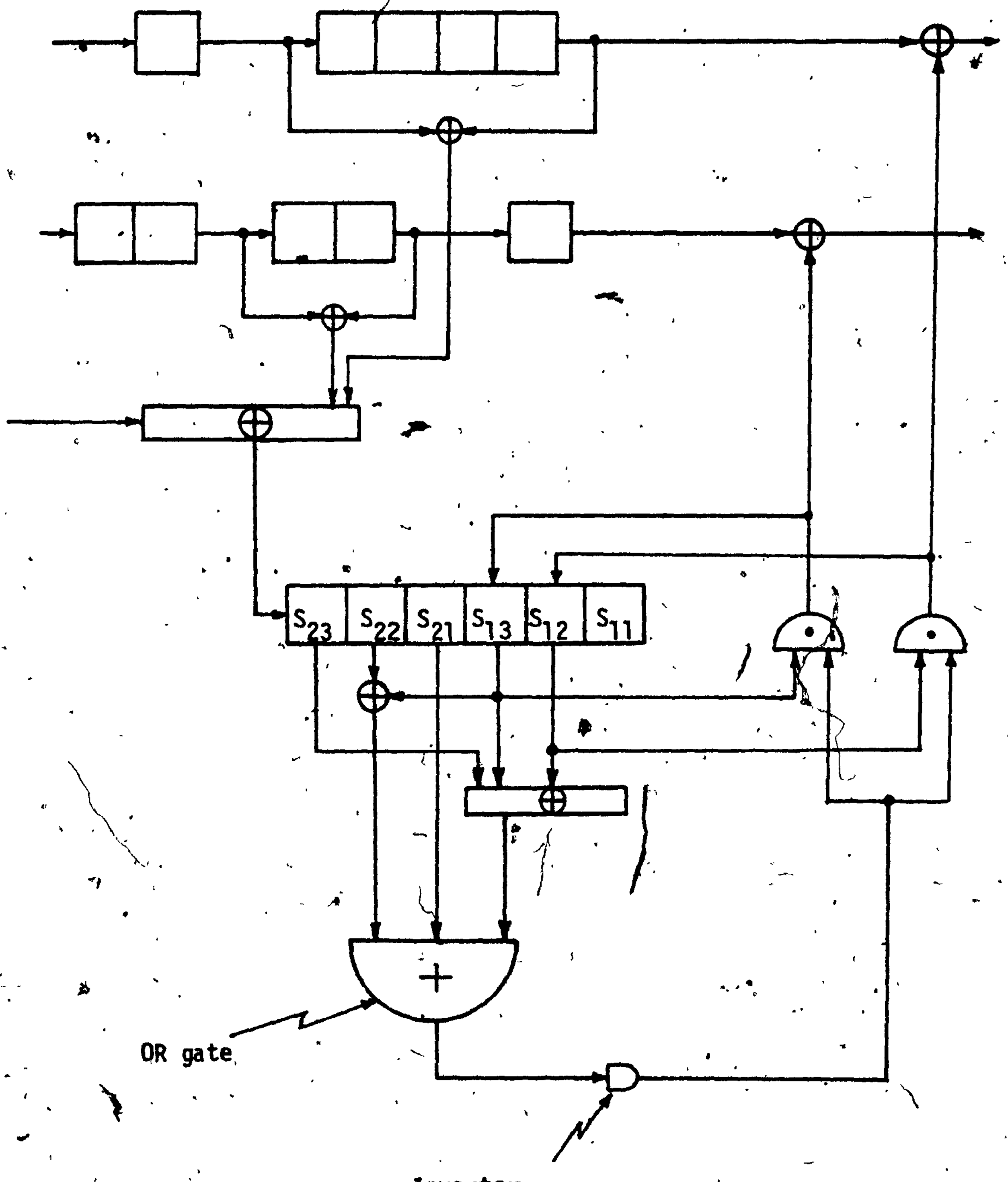


FIG. 3.6 Decoder for (18,12) BPM Code

correcting ability, $b \leq n_0$, when $g = mn_0-1$ error free digits are available on either side of the burst (a burst can be any combination of error digits in one or two consecutive blocks when they are confined to n_0-bit errors). The general parity-check matrix of this code is of the form

$$H_N = [B_0 \; B_i \; B_{2i} \; \ldots \;] \tag{3.9}$$

where B_i is $i = n_0-k_0$ down-shifted truncated version of B_0 . For basic code $i=1$, Iwadare has found two different B_0 matrices. Accordingly they are named first class and second class Iwadare codes.

For the first class code, the B_0 matrix and m are given as below:

$$B_0 = \begin{array}{c l} \overbrace{\hphantom{000\cdots\cdots0000}}^{n_0} & \\ \left[\begin{array}{c} 000\cdots\cdots0001\\ 000\cdots\cdots0010\\ 000\cdots\cdots0000\\ 000\cdots\cdots0100\\ \vdots \quad \iddots \quad \vdots\\ 010\cdots\cdots0000\\ 000\cdots\cdots0000\\ 100\cdots\cdots0000\\ 000\cdots\cdots0000\\ \vdots \qquad\qquad \vdots\\ 000\cdots\cdots0000\\ 000\cdots\cdots0010\\ 000\cdots\cdots0100\\ \vdots \quad \iddots \quad \vdots\\ 0010\cdots\cdots0000\\ 010\cdots\cdots0000\\ 100\cdots\cdots0000 \end{array}\right] & \begin{array}{l} \\ \\ \\ 2n_0-2 \\ \\ \\ \\ \\ \\ n_0 \\ \\ \\ \\ n_0-1 \\ \\ \\ \\ \end{array} \end{array} \tag{3.10}$$

$$m = 2(2n_0-1)-1 \qquad (3.11)$$

The rule for constructing matrix B_0 of Eq. 3.10 is as follows:

1) The jth columns $(j=1,2, \ldots, n_0-1)$ have only two nonzero entries at the $(2n_0-2j)$th row and the $(4n_0-2-j)$th row.

2) The n_0th column has only one nonzero entry at the first row.

For the second class code, the B_0 matrix and m are as follows:

$$B_0 = \begin{array}{c}\overbrace{\hphantom{0\cdots\cdots\cdots0001}}^{n_0}\\ \left[\begin{array}{c} 0\cdots\cdots\cdots0001 \\ 0\cdots\cdots\cdots0000 \\ \vdots \qquad\qquad \vdots\;\vdots \\ 0\cdots\cdots\cdots0000 \\ 0\cdots\cdots\cdots0010 \\ 0\cdots\cdots\cdots0010 \\ 0\cdots\cdots\cdots0100 \\ 0\cdots\cdots\cdots0000 \\ 0\cdots\cdots\cdots0100 \\ \\ 01\cdots\cdots\cdots0000 \\ 10\cdots\cdots\cdots0000 \\ 00\cdots\cdots\cdots0000 \\ \vdots\;\vdots \qquad\qquad \vdots \\ 00\cdots\cdots\cdots0000 \\ 10\cdots\cdots\cdots0000 \end{array}\right]\end{array} \begin{array}{l} \\ \left.\begin{array}{c}\\ \\ \\ \\ \end{array}\right\} n_0 \\ \left.\begin{array}{c}\\ \\ \end{array}\right\} 2 \\ \left.\begin{array}{c}\\ \\ \\ \end{array}\right\} 3 \\ \\ \\ \left.\begin{array}{c}\\ \\ \\ \\ \\ \end{array}\right\} n_0 \end{array} \qquad (3.12)$$

$$m = \frac{n_o(n_o-1)}{2} + (2n_o-1) \qquad (3.13)$$

The rule for constructing matrix B_o of Eq. 3.12 is given below:

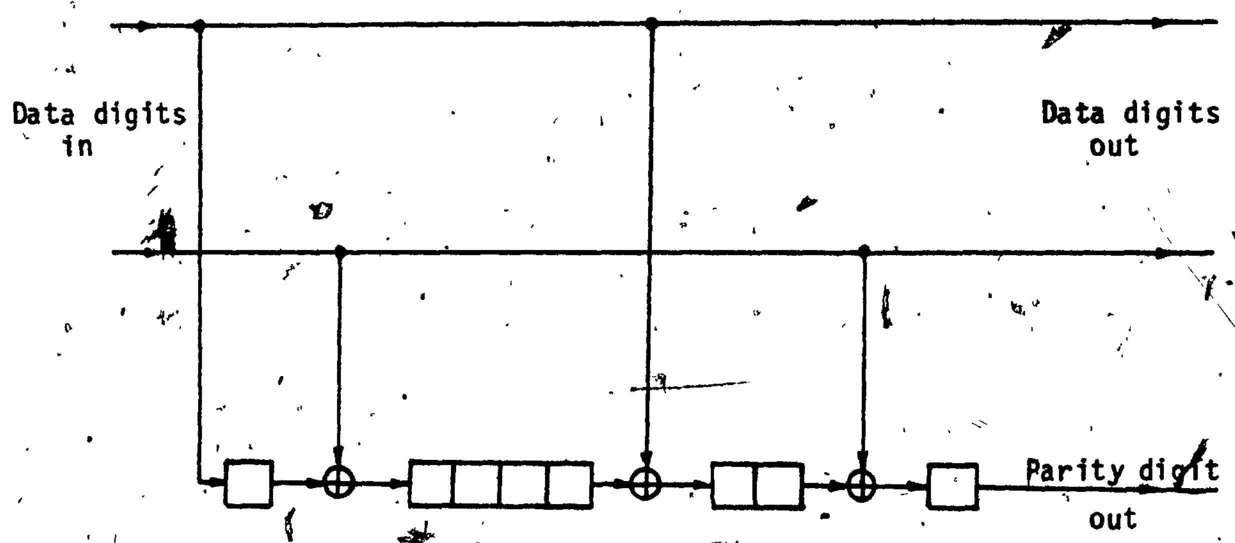
1) The jth columns $(j=1,2, \ldots, n_o-1)$ have only nonzero entries at the $n_o + [\frac{(n_o-j)(n_o-j+1)}{2}]$th row and $n_o + [\frac{(n_o-j)(n_o-j+1)}{2}] + (n_o-j)$th row.

2) The n_oth column has only one nonzero entry at the first row.

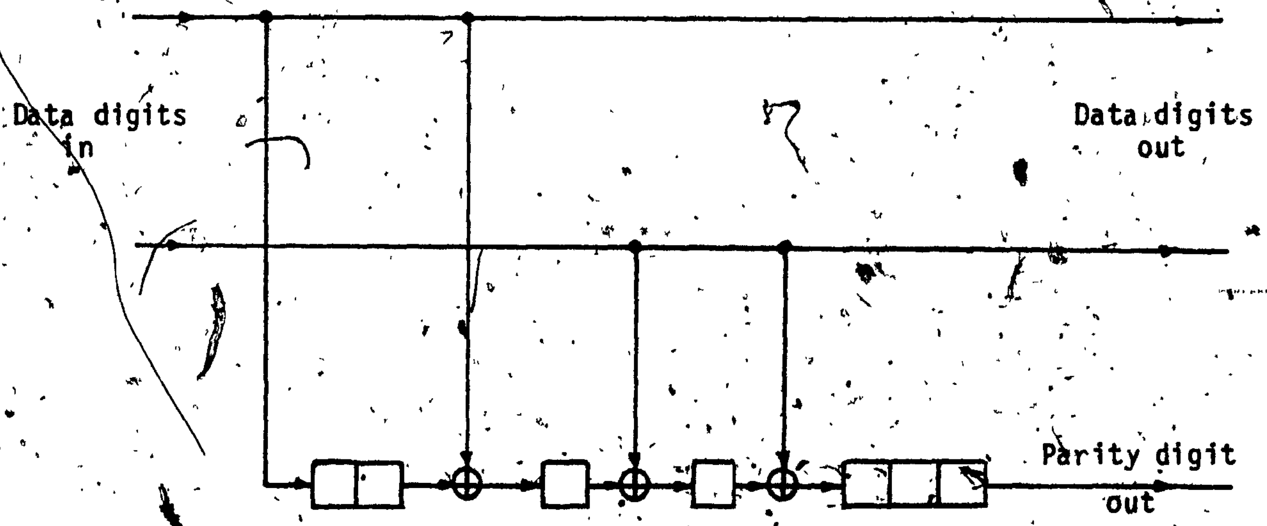
When we know the matrix B_o, we can easily construct an (n-k)-stage encoder for encoding data digits. We will notice later that the connection of the shift registers is determined by the position of nonzero entries in each column of B_o. Figure 3.7 illustrates the encoders for both Iwadare codes when $n_o=3$. The B_o matrices are also shown below:

$$B_o = \begin{bmatrix} 0&0&1\\ 0&1&0\\ 0&0&0\\ 1&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&0\\ 0&1&0\\ 1&0&0 \end{bmatrix} \text{(First class)}, \quad B_o = \begin{bmatrix} 0&0&1\\ 0&0&0\\ 0&0&0\\ 0&1&0\\ 0&1&0\\ 1&0&0\\ 0&0&0\\ 1&0&0 \end{bmatrix} \text{(Second class)}$$

The procedure for decoding the Iwadare codes differs fundamentally from all decoding techniques discussed previously. That is,



(a)



(b)

FIG. 3.7 Iwadare Encoders; a) for (27,18) first class code
b) for (24,16) second class code

the n_o-1 data digits in each block are decoded at different times. In particular, the (n_o-1)th digit is decoded after n_o+2 blocks are received, the (n_o-2)th digit after n_o+2+3 blocks ... the first digit after $n_o+2+3+\ldots+n_o$ blocks. Thus, decoding is accomplished in n_o-1 steps.

The decoders are composed of $(n_o-1)(m-1)$ stage encoders as well as a decoding algorithm circuit. Again, we consider $n_o=3$ and explain the principle of decoding for first class of Iwadare code. Suppose that errors are in the "0"th block of encoded data digits; that is

$$E^T = e_o^1\ e_o^2\ e_o^3\ 00 \ldots 0 \tag{3.14}$$

The syndrome sequence which comes into the decoding algorithm according to (3.14) is

$$S^T = e_o^3\ e_o^2\ 0\ e_o^1\ 000\ e_o^2\ e_o^1 \tag{3.15}$$

Notice that errors on data digits, e_o^2 and e_o^1, are repeated twice in S^T. When the second e_o^2 comes into the circuit, as shown in figure 3.8a, the pattern $(e_o^2\ XXXXX\ e_o^2)$ where X is either "0" or "1" satisfies the input to the AND gate number 2. Thus, the output o[illegible] AND gate corrects the second error data digit and at the same [illegible] is fed back to the syndrome for syndrome resetting. One time unit later e_o^1 comes into the circuit and the pattern $(e_o^1\ XXXX\ e_o^1)$ satisfies the AND gate number 1 which in turn corrects the first error data digit. This is followed by immediate syndrome resetting, thus decoding is completed.

If a burst starts from the second data sequence, then:

$$E^T = e_0^2\ e_0^3\ e_1^T\ 00 \ldots 0 \qquad (3.16)$$

and $$S^T = e_0^3\ e_0^2\ 00\ e_1^1\ 00\ e_0^2\ 0\ e_1^1\ 0 \ldots \qquad (3.17)$$

The pattern $(e_1^1\ XXXX\ e_1^1)$ appears just one digit shifted after the pattern $(e_0^2\ XXXXX\ e_0^2)$ and will be decoded as before without any erroneous operation. For the second class of Iwadare code , if the burst occurs in the "0"th block, the syndrome sequence will be

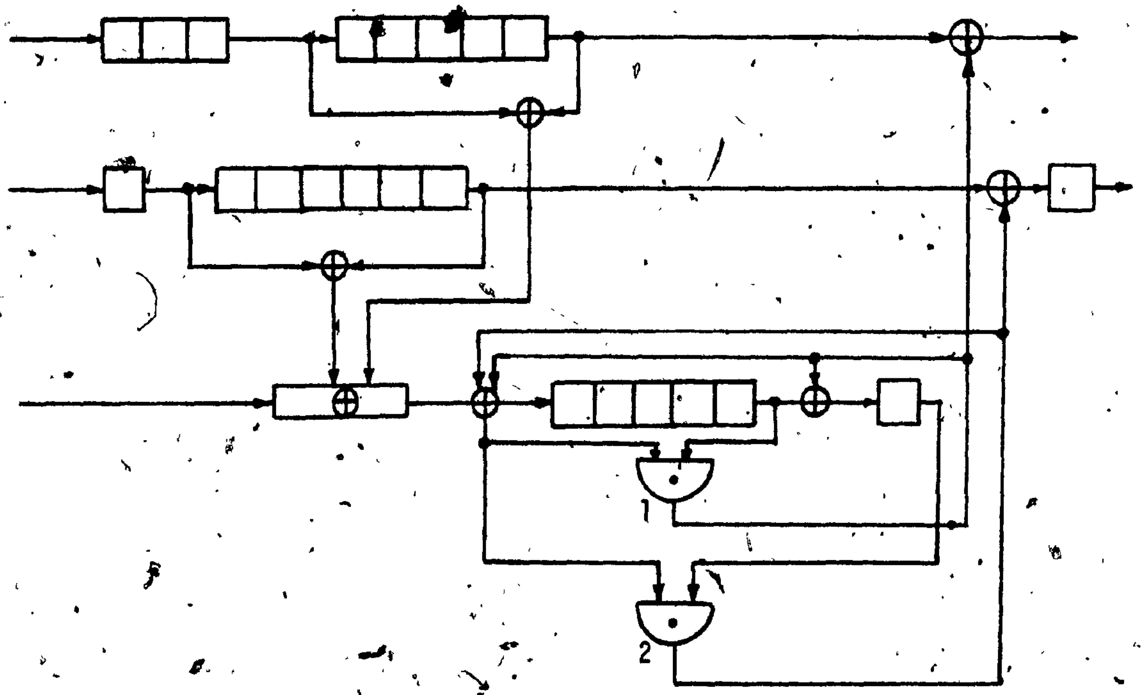
$$S^T = e_0^3\ 00\ e_0^2\ e_0^2\ e_0^1\ 0\ e_0^1\ 00 \ldots \qquad (3.18)$$

The second error data digit will be corrected by the pattern $(e_0^2\ e_0^2)$ and the first error data by the pattern $(e_0^1 X e_0^1)$ ..

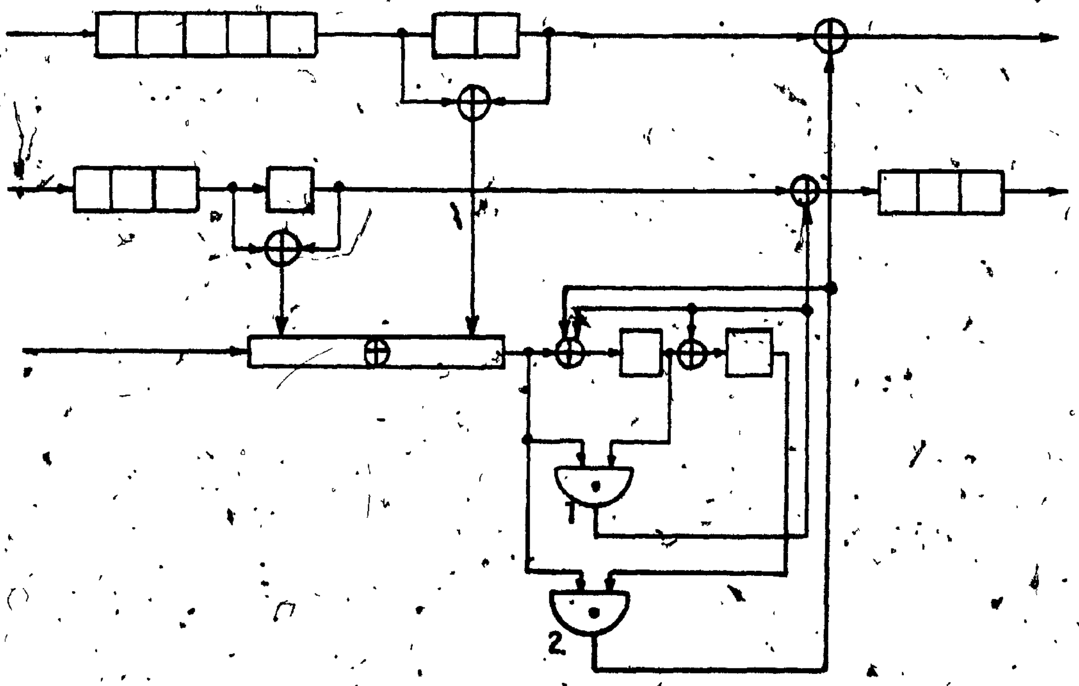
Decoders for the first and second class Iwadare codes are illustrated in figure 3.8. Some typical values for guard space, encoder and decoder shift register stages, code rates and block lengths are given in table 3.4 for first class codes and in table 3.5 for second class Iwadare codes.

3.4 Hsu Code

The basic Hsu code [27] is a $(mn_0, m(n_0-1))$ convolutional code with rate $R = \frac{n_0-1}{n_0}$ and Type B_1 burst error correcting ability, $b \le n_0$, when $g = mn_0-1$ error free digits are available on either side of the burst. For this code $m = 3n_0$ and matrix B_0 is given as follows:



(a)



(b)

FIG. 3.8 Iwadare Decoder a) for (27,18) first class code
b) for (24,16) second class code

TABLE 3.4

TYPICAL VALUES FOR THE FIRST CLASS IWADARE CODE

Block Length n_o	Code Rate R	Shift Register Stages		Guard Space g^*
		Encoder E^*	Decoder D^*	
2	1/2	4	7	9
3	2/3	8	22	26
4	3/4	12	45	51
5	4/5	16	76	84

$^*g = 4n_o^2 - 3n_o - 1$

$E = 4n_o - 4$

$D = 4n_o^2 - 5n_o + 1$

TABLE 3.5

TYPICAL VALUES FOR THE SECOND CLASS IWADARE CODE

Block Length n_o	Code Rate R	Shift Register Stages		Guard Space g^*
		Encoder E^*	Decoder D^*	
2	1/2	3	4	7
3	2/3	7	16	23
4	3/4	12	39	51
5	4/5	18	76	94

$$^*g = \frac{n_o^3 + 3n_o^2 - 2n_o - 2}{2}$$

$$E = \frac{n_o^2 + 3n_o - 4}{2}$$

$$D = \frac{n_o^3 + 2n_o^2 - 5n_o + 2}{2}$$

$$B_o = \begin{array}{c} \overset{\longleftarrow \; n_o \; \longrightarrow}{} \\ \left[\begin{array}{cccccc} 0 & 0 & \cdots & \cdots & 0 & 1 \\ \vdots & 0 & \cdots & 0 & 1 & 0 \\ \vdots & & \mathbf{O} & & 0 & 0 \\ \vdots & & & \mathbf{O} & \vdots & \vdots \\ 0 & 1 & & & \vdots & \vdots \\ 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \\ 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 \\ \vdots & & & & & \vdots \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \end{array}\right] \end{array} \begin{array}{l} \\ \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} n_o+1 \\ \\ \\ \\ \\ \\ \\ \end{array} \qquad (3.19)$$

The parity check matrix, H_N , for this code is the same as Eq. (3.9) which was given before. Again, when we know matrix B_o and i we can easily construct matrix H_N and encode the data digits. As an example, we construct matrix B_o for $n_o = 3$ and illustrate the appropriate encoding circuit for (27, 18) Hsu code. The encoder is shown in figure 3.9.

$$B_o = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

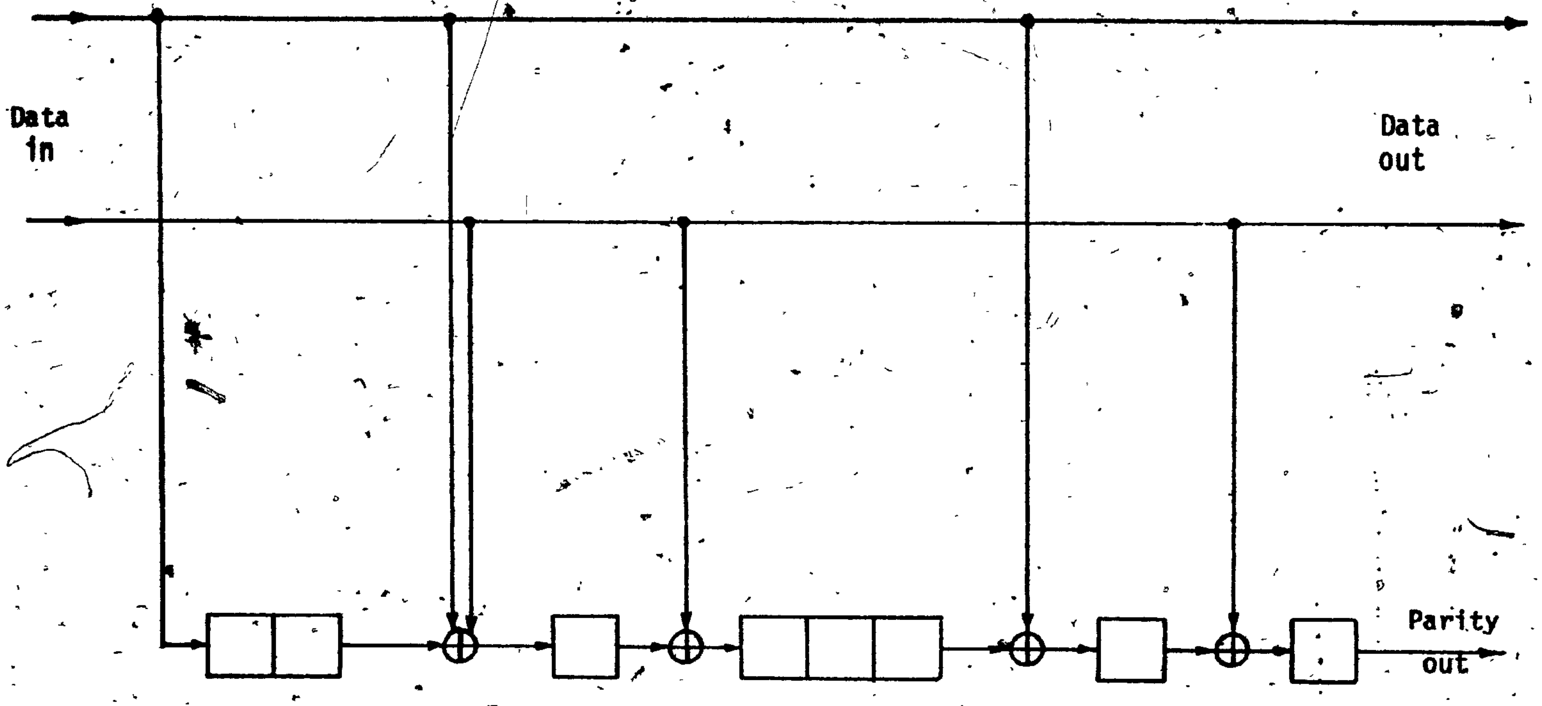


FIG. 3.9 8-Stage Encoder for (27,18) Hsu Code

The decoding procedure is similar to the Iwadare method, so it will not be mentioned in detail again. Rather, we will just consider the syndrome pattern and decoder circuit for $n_o = 3$.

If the error is in the "0"th block, we have:

$$E^T = (e_o^1 \, e_o^2 \, e_o^3 \, 00 \ldots 0) \tag{3.20}$$

Thus the syndrome will be:

$$S^T = (e_o^3 \, e_o^2 \, e_o^1 \, 0 \, e_o^3 \, 0 \, e_o^2 \, 0 \, e_o^1 \, 00 \ldots 0) \tag{3.21}$$

Notice that each error digit e_o^i, $1 \leq i \leq 3$, is repeated twice and can be detected according to table 3.6, where X is arbitrary.

TABLE 3.6

SYNDROME PATTERN OF HSU CODE

Error Digit	Syndrome Pattern	Gate Signal
e_o^1	$\underbrace{e_o^1 \text{ XXXXX } e_o^1}_{6}$	$q_1 = 1$
e_o^2	$\underbrace{e_o^2 \text{ XXXX } e_o^2}_{5}$	$q_2 = 1$
e_o^3	$\underbrace{e_o^3 \text{ XXX } e_o^3}_{4}$	$q_3 = 1$

From table 3.6 it is clear that for each e_o^i, $1 \leq i \leq 3$, a different syndrome pattern:

$$(e_0^i\,\underbrace{XXX\ \quad \dots X\, e_0^i}_{7-i})$$

exists. Therefore if $e_0^i = 1$, then e_0^i can be detected. If e_1^i occurs instead of e_0^i, the same syndrome pattern for e_1^i exists, i.e.,:

$$(e_1^i\,\underbrace{XXX\ \quad \dots X\, e_1^i}_{7-i})$$

except that it occurs one unit of time later. If e_0^i is detected, a gate signal, q_i, is generated triggering the error correction and syndrome-resetting. Only one gate signal should be generated each time. Therefore, an inhibiting rule is provided to enfore that if $q_i = 1$, then $q_j = 0$ for all $j>i$. Figure 3.10 shows the decoder for (27, 18) Hsu code.

Comparing this code with the first class of Iwadare code [26], one notices that for the same burst error correcting ability, $b \leq n_0$; and block length, n_0, when $n_0 \geq 5$ Hsu code has advantages over Iwadare code, that is, it requires less guard space and fewer shift register stages. Table 3.7 shows this comparison in which S_I-S_H stands for the difference in the total number of encoder and decoder shift register stages and g_I-g_H stands for the difference in guard space. Subscripts "I" and "H" refer to Iwadare and Hsu, respectively.

3.5 Rodgers Code

The basic Rodgers code [5] is a $(mn_0, m(n_0-1))$ convolutional code with rate $R = \frac{n_0-1}{n_0}$ and type B_1 burst-error correcting ability,

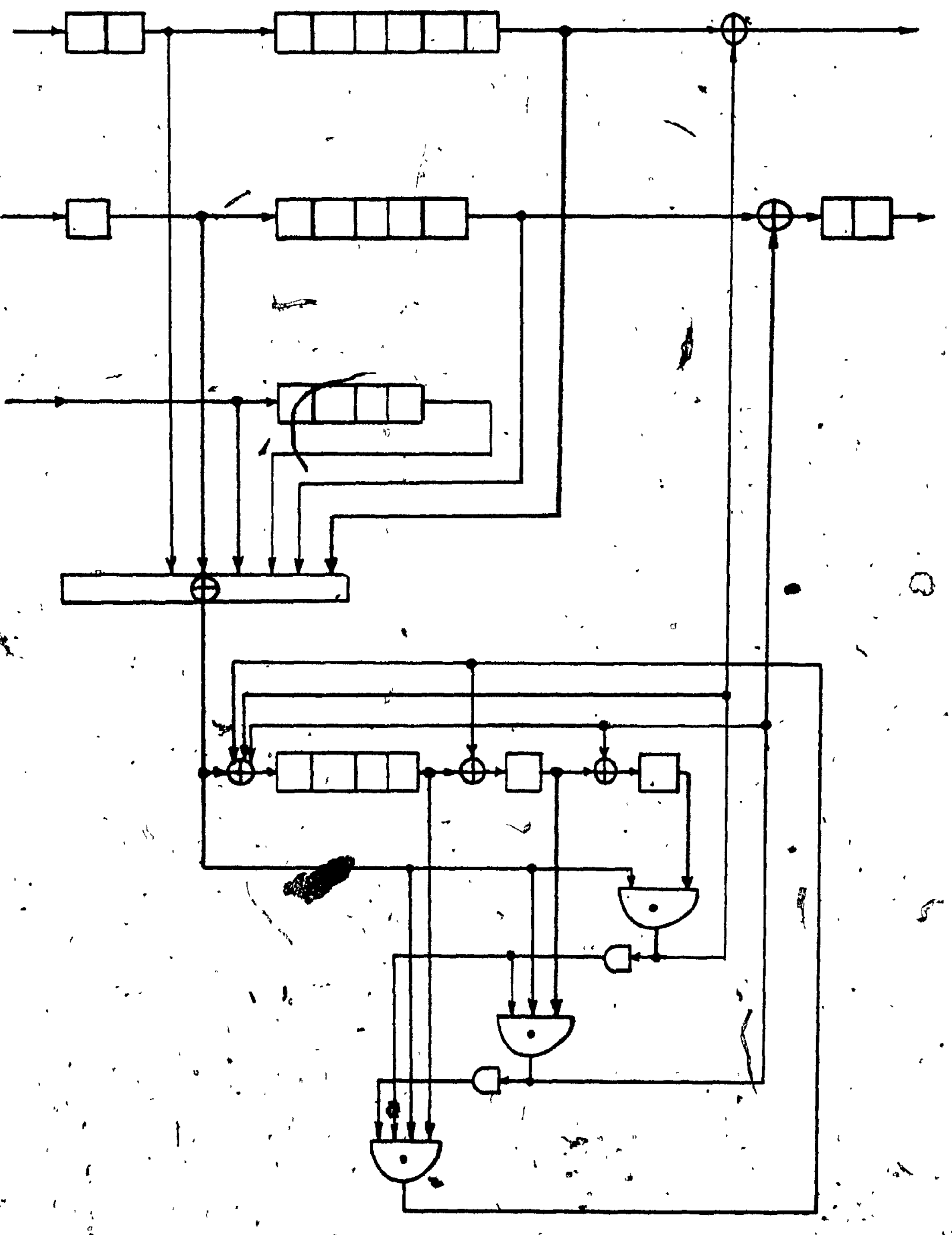


FIG. 3.10 Decoder for (27,18) Hsu Code

TABLE 3.7

COMPARISON BETWEEN HSU AND IWADARE CODES, S_I-S_H AND G_I-G_H VS. BLOCK LENGTH n_o

n_o	S_I-S_H *	g_I-g_H *
3	$-4 \therefore S_H > S_I$	0
4	0	4
5	6	10
10	66	70
15	176	180
20	336	340

* $S_I-S_H = n_o(n_o-3)-4$

$g_I-g_H = n_o(n_o-3)$

$b \leq n_o$ when $g = 3n_o^2 - 2n_o-1$ error free digits are available on either side of the burst. The matrix B_o for this code is given below:

$$B_o = \begin{array}{c|l}
\multicolumn{1}{c}{\overleftrightarrow{\quad n_o \quad}} & \\
\left[\begin{array}{ccccccc}
0 & 0 & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & \cdots & 0 & 0 & 1 & 0 \\
0 & 0 & \cdots & 0 & 1 & 0 & 0 \\
\vdots & & & \iddots & & \vdots & \vdots \\
0 & 1 & 0 & \cdots & 0 & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 1 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 1 & 0 & 0 \\
\vdots & & & \iddots & & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & 0
\end{array}\right] &
\begin{array}{l}
\left.\begin{array}{c}\\ \\ \\ \\ \\ \end{array}\right\} n_o \\
\} 1 \\
\left.\begin{array}{c}\\ \\ \\ \\ \\ \\ \end{array}\right\} 2(n_o-2) \\
\} 1
\end{array} \Bigg\} 3n_o-2
\end{array}$$

For this code $m = 3n_o-2$ and $i=1$.

The matrix B_o is constructed as follows:

The top n_o rows are the mirror image of the identity matrix of order n_o . Row n_o+1 is a repeat of row n_o . The next $2(n_o-2)$ rows are formed in sets of 2 rows each, the top rows of each set are to the power of 2 from 2^1 to 2^{n_o-2} written in binary form. The lower row of each set is the all-zero row. The bottom row of B_o is comprised of zeros. As an example, when $n_o = 2$ the matrix B_o is equal to

$$B_0 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

This code was found by Berlekamp [21] and is known as a minimum constraint length code $(n = 2n_0^2)$. Appendix C lists the B_0 matrices of codes with minimum constraint length.

Now let us consider $n_0 = 3$ and find the B_0 matrix. We will then illustrate the encoder for (21, 14) Rodgers code, accordingly. The encoder is shown in figure 3.11.

$$B_0 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The decoder circuit is based on the analysis mentioned in Chapter 2. That is, desired syndromes which are distinct from each other are chosen and undesired syndromes are rejected. Then each distinct desired syndrome pattern is used to correct the related error digit. This will be done for the error digits in the "0"th block simultaneously each time. The effect of the "0"th block errors on the syndrome will be removed after each correction in order to prevent error propagation. The errors in the other blocks will be corrected as "0"th block provided

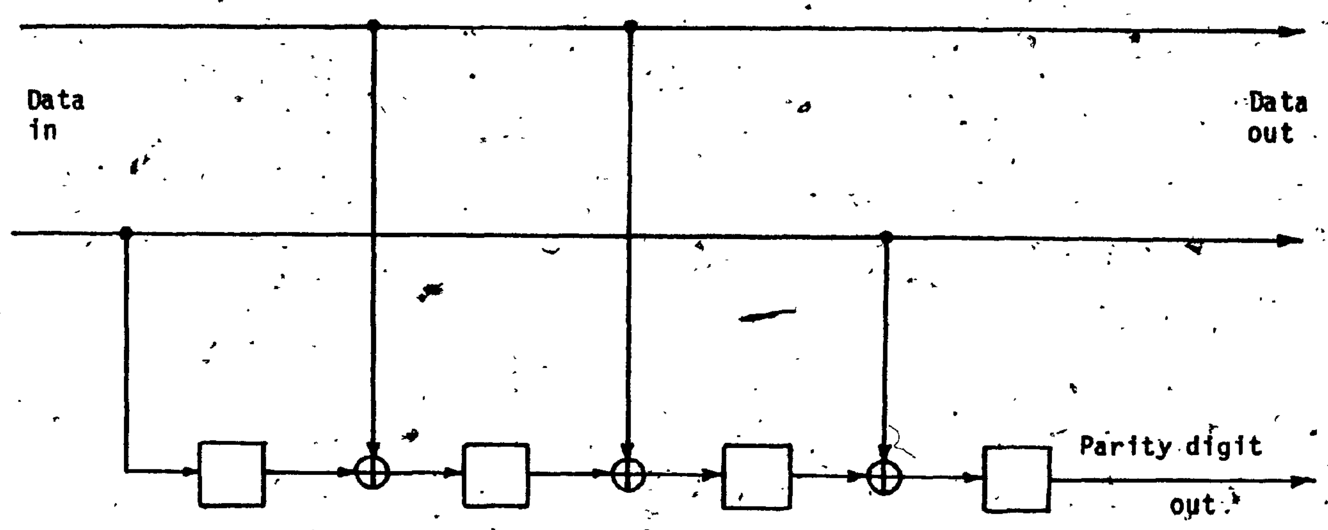


FIG. 3.11 4-Stage Encoder for (21,14) Rodgers Code

that the required guard space around each burst is available.

The simplest way to understand Rodgers' decoding circuit is by folloiwng an example, again, for (21, 14) Rodgers code. The code is of the type B_1 so we have $n_o 2^{n_o - 1}$ desired syndromes. For $n_o = 3$ this is equal to 12 . The syndromes for different error patterns are given below (data digits are represented as d_j^i where i is the ith data digit and j is the jth block. p_j is the parity digit in the jth block and comes after the data digits of each block). We have:

$S_1^T = [1\ 1\ 1\ 1\ 1\ 0\ 0]$ when d_o^1, d_o^2 and p_o are all in error
$S_2^T = [0\ 1\ 1\ 1\ 1\ 0\ 0]$ when d_o^1 and d_o^2 are both in error
$S_3^T = [1\ 0\ 1\ 1\ 0\ 0\ 0]$ when d_o^1 and p_o are both in error
$S_4^T = [0\ 0\ 1\ 1\ 0\ 0\ 0]$ when d_o^1 is in error
$S_5^T = [0\ 1\ 0\ 0\ 1\ 0\ 0]$ when d_o^2 is in error
$S_6^T = [1\ 0\ 0\ 0\ 0\ 0\ 0]$ when p_o is in error
$S_7^T = [1\ 1\ 0\ 0\ 1\ 0\ 0]$ when d_o^2 and p_o are both in error
$S_8^T = [1\ 1\ 0\ 1\ 0\ 0\ 0]$ when d_o^2 , p_o and d_1^1 are all in error
$S_9^T = [0\ 1\ 0\ 1\ 0\ 0\ 0]$ when d_o^2 and d_1^1 are both in error
$S_{10}^T = [1\ 0\ 0\ 1\ 1\ 0\ 0]$ when p_o and d_1^1 are both in error
$S_{11}^T = [1\ 0\ 1\ 1\ 1\ 1\ 0]$ when p_o , d_1^1 and d_1^2 are all in error
$S_{12}^T = [1\ 0\ 1\ 0\ 0\ 1\ 0]$ when p_o and d_1^2 are both in error

Now we group those syndromes that have d_o^1 indicated as an error in all of them. Thus we will have:

$$\begin{array}{cccc}
S_1 & S_2 & S_3 & S_4 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}$$

Notice that rows 3 and 4 of these syndromes are all "1"'s and rows 6 and 7 are all "0"'s . When there is a "1" in row 2, there is also a "1" in row 5 and if there is a "0" in row 2, there is also a "0" in row 5. With this information we can design the required logic circuit for detecting the error in d_o^1 . This is shown in figure 3.12a.

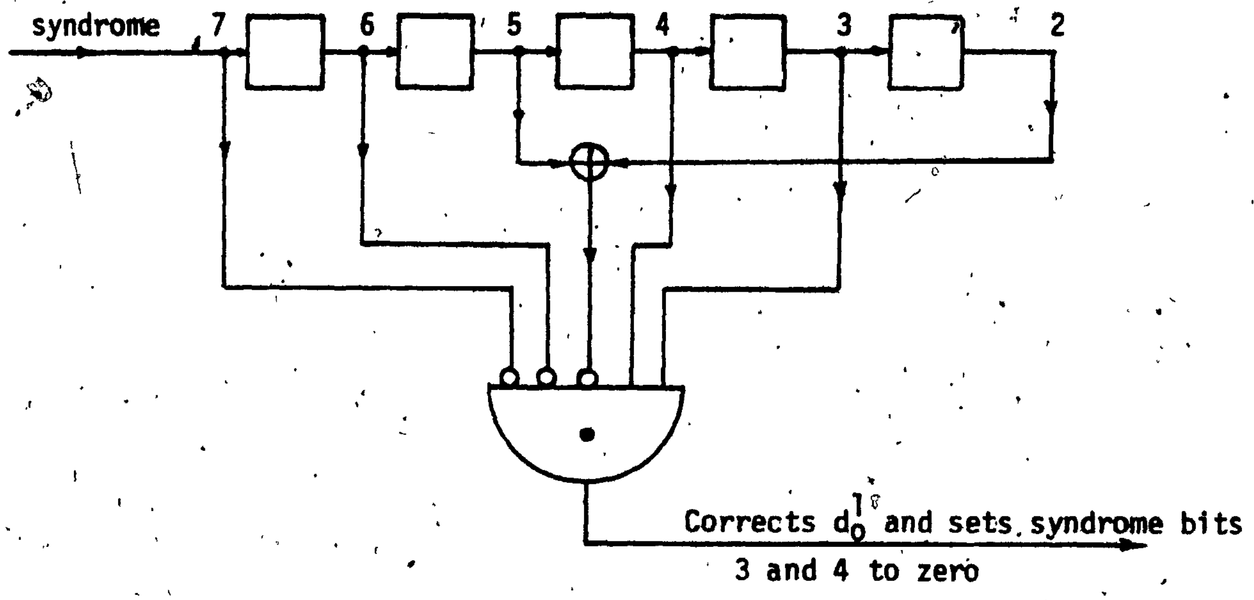
Now grouping those syndromes in which d_o^2 is in error in all of them, we obtain

$$\begin{array}{cccccc}
S_1 & S_2 & S_5 & S_7 & S_8 & S_9 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

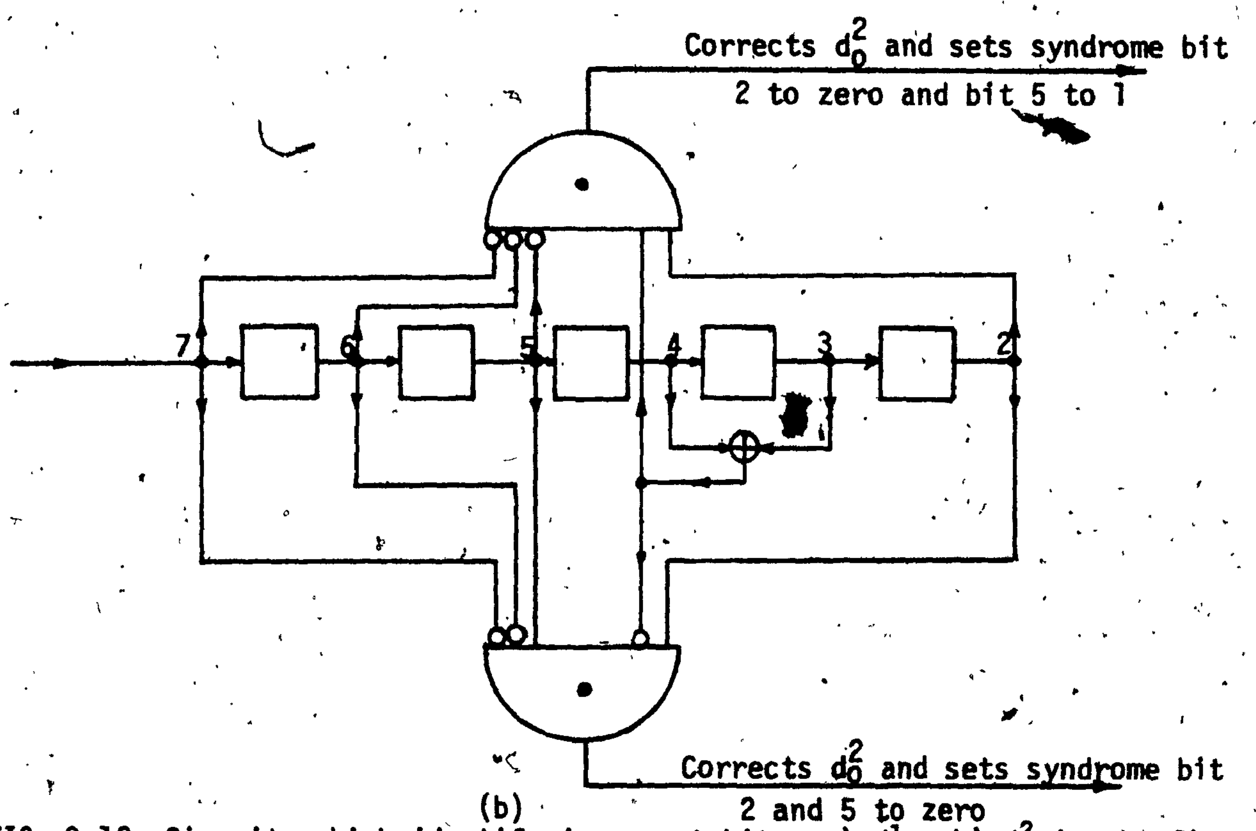
Notice that row 2 of all syndromes are "1"'s . Rows 6 and 7 all are "0"'s . Row 5 is "1" when the digits in rows 3 and 4 of each individual syndrome are identical and is "0" when these digits are different. Based on this information the logic circuit for detecting the error digit in d_0^2 is drawn in figure 3.12b.

Since we are not interested in correcting parity bits, the circuit for detecting them is not considered here.

The complete decoder circuit for (21, 14) Rodgers code is illustrated in figure 3.13.



(a)


Corrects d_0^2 and sets syndrome bit 2 to zero and bit 5 to 1
7
6
5
4
3
2
Corrects d_0^2 and sets syndrome bit 2 and 5 to zero

(b)

FIG. 3.12 Circuits which identify incorrect bits; a) d_0^1 ; b) d_0^2 in the first sub-block of the error burst

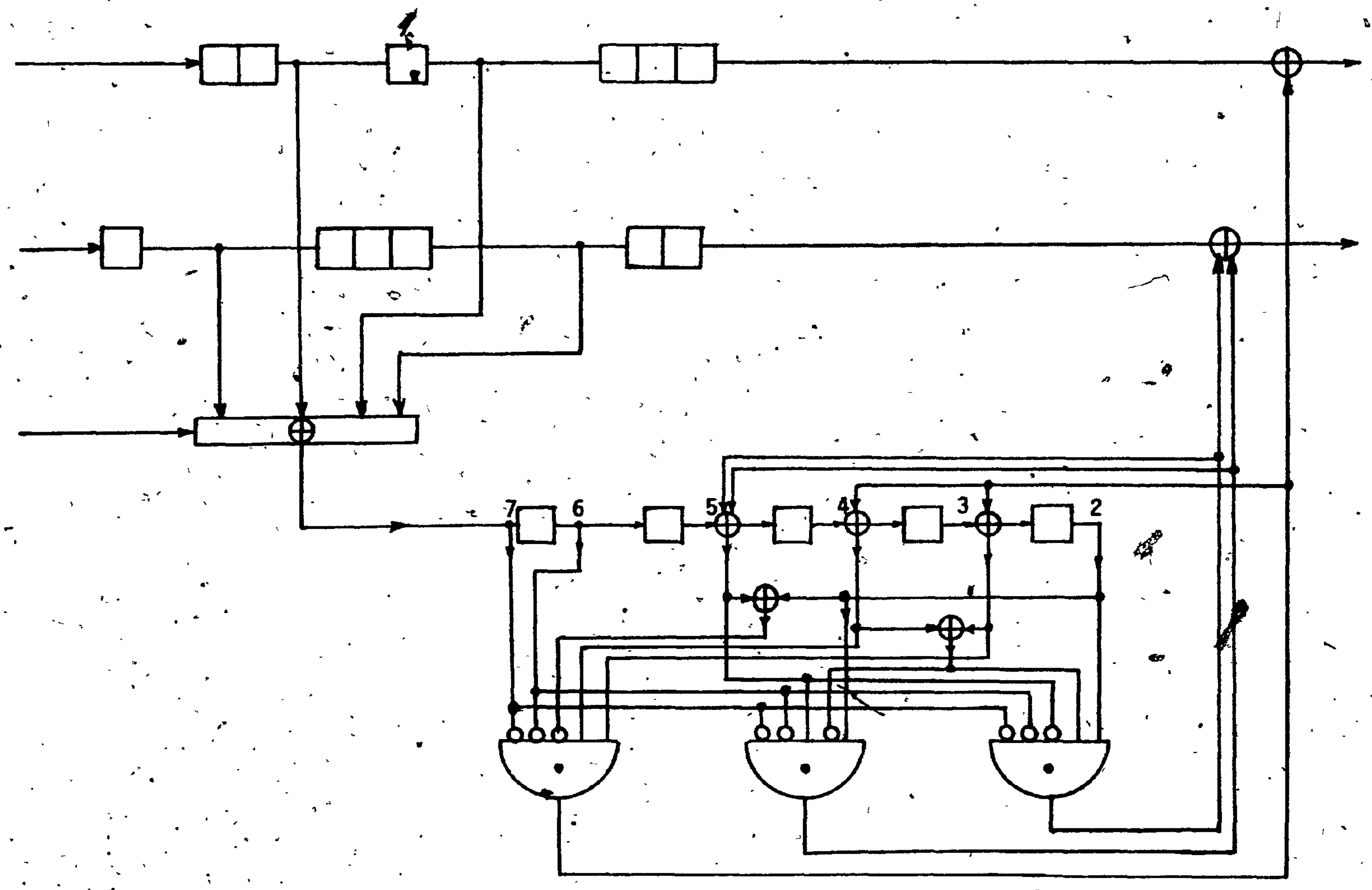


FIG. 3.13 Decoder for (21, 14) Rodgers Code

CHAPTER 4

RANDOM AND BURST-ERROR-CORRECTING CONVOLUTIONAL CODES

In real communication channels we usually interface with two kinds of error patterns. These are burst errors and random errors. The codes described in this chapter have the capability of combating random and burst errors provided that the required conditions for each case exist.

4.1 Massey-Kohlenberg Diffuse Code

The Massey-Kohlenberg code [23], [24], [8] is a $(2(3x+2)$, $3x+2)$ convolutional code with rate $R = 1/2$ which is derived from a $t=2$ random error correcting Massey code [34]. It is capable of correcting all burst-error patterns not more than $2x$ digits long $(x \geq 1$ information digits and x parity digits) given an error free guard space of $g=n-t = 6x+2$ digits on either side of the burst.

The encoder, shown in figure 4.1, is a $(3x+1)$-stage shift register. Information digits are fed into the channel (there is no information delay in the encoding process) and the shift register simultaneously, and a parity digit is formed by summing (modulo 2) the contents of three register locations and the input (locations 1, $x+2$, and $2x+2$). For the case in which the information digit d_{3x+2} enters into the channel and shift register, the related parity digit is:

$$p_{3x+2} = d_1 + d_{x+2} + d_{2x+2} + d_{3x+2} \qquad (4.1)$$

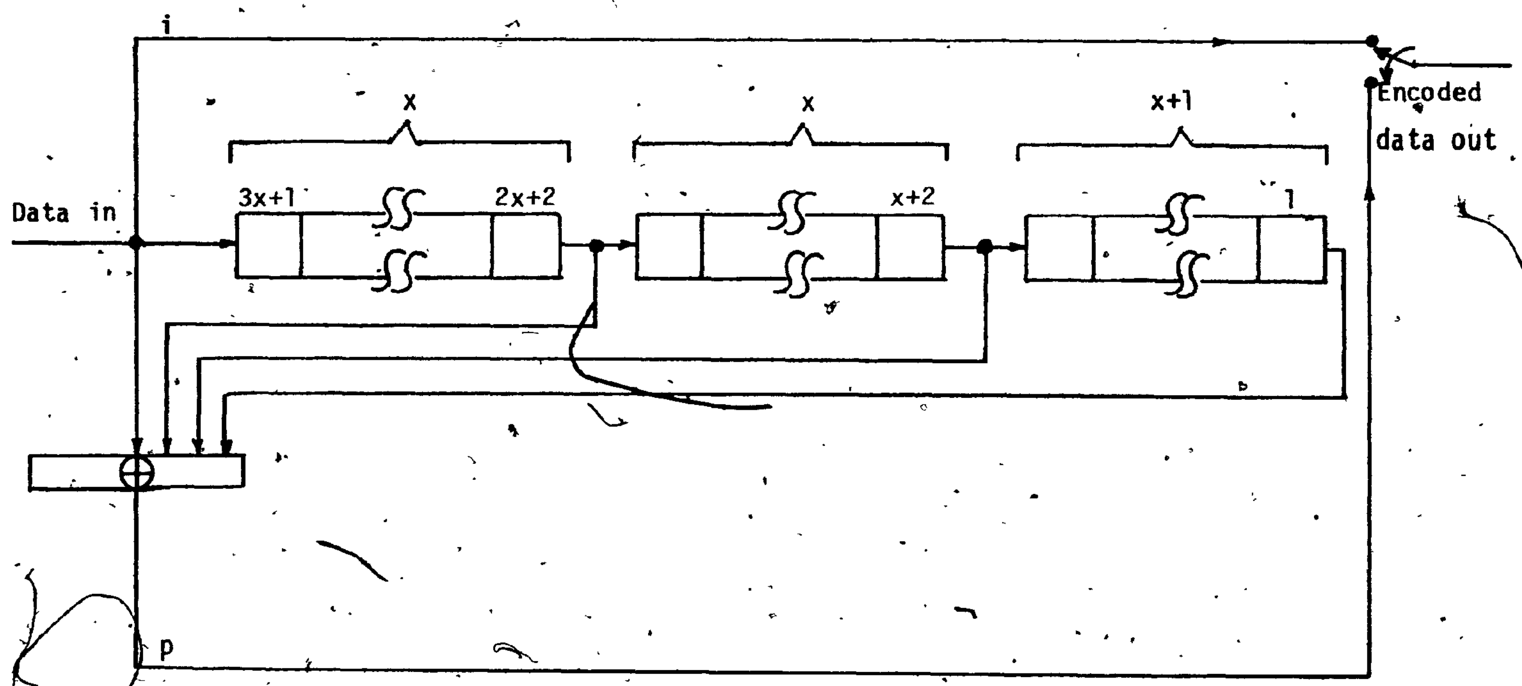


FIG. 4.1 Encoder for the Massey-Kohlenberg Diffuse Code

This will be fed into the channel after information digit d_{3x+2} . In the next coding cycle, digit d_{3x+3} enters the register, all other digits shift one place and the related parity digit is:

$$p_{3x+3} = d_2 + d_{x+3} + d_{2x+3} + d_{3x+3}$$

which, again, will be fed into the channel after information digit d_{3x+3} . The encoding operation continues in this manner with no division of the information or parity into an independent block structure.

The decoder, shown in figure 4.2, contains a replica of the encoder into which the received information digits are fed. These received digits d'_k are primed to indicate that they may not be equal to the original digits d_k , due to possible transmission error. As before, we calculate the syndrome. When the data digit d'_{3x+2} and the parity digit p'_{3x+2} enter the decoder, the syndrome S_{3x+2} is given by:

$$S_{3x+2} = d'_1 + d'_{x+2} + d'_{2x+2} + d'_{3x+2} + p'_{3x+2} \tag{4.2}$$

where $d'_k = d_k + e^d_k$

$$p'_k = p_k + e^p_k \tag{4.3}$$

e^d_k is the information error digit and e^p_k is the parity error digit. They are "0" or "1" depending on whether the corresponding received bit is correct or incorrect. Substituting Eqs. 4.3 into Eq. 4.2 and using Eq. 4.1 we find that:

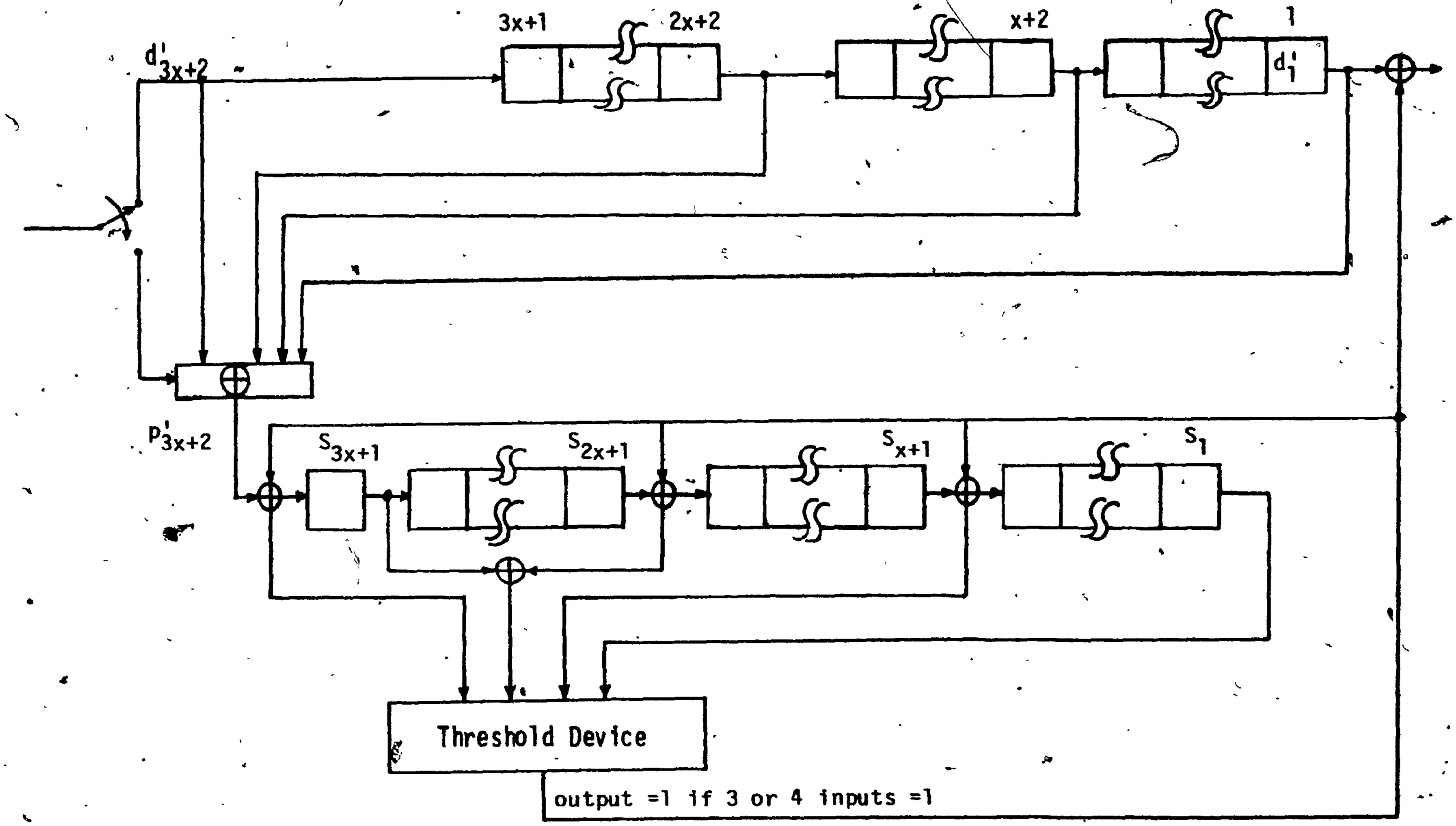


FIG. 4.2 Decoder for the Massey-Kohlenberg Diffuse Code

$$S_{3x+2} = e_1^d + e_{x+2}^d + e_{2x+2}^d + e_{3x+2}^d + e_{3x+2}^p \qquad (4.4)$$

This means that the values of the syndromes depend only on the errors in the information and parity bits from which they are formed and not on the actual values of the transmitted bits themselves. These syndromes are stored in a shift register of the same length as that of the information register.

At the instant shown in figure 4.2 the function of the decoder is to decide whether or not d_1' is correct and, if it is incorrect to change it. To explain the decoding procedure we shall initially assume that there have been no previous transmission errors, so that errors in "d" , or "p" , may be called first errors. With this assumption, we can compute the values of the specific "S" shown in the figure and used in decoding. The result will be:

$S_1 = e_1^d + e_1^p$ when d_1' and p_1' enter into the decoder,

$S_{x+1} = e_1^d + e_{x+1}^d + e_{x+1}^p$ when d_{x+1}' and d_{x+1}' enter into the decoder,

$S_{2x+1} = e_1^d + e_{x+1}^d + e_{2x+1}^d + e_{2x+1}^p$ when d_{2x+1}' and p_{2x+1}' enter into the decoder,

$S_{3x+1} = e_{x+1}^d + e_{2x+1}^d + e_{3x+1}^d + e_{3x+1}^p$ when d_{3x+1}' and p_{3x+1}' enter into the decoder,

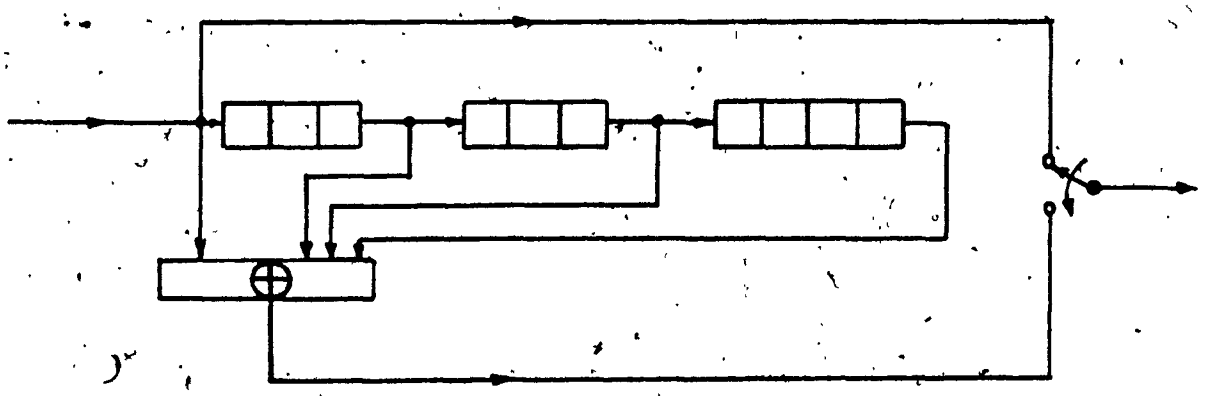
and $S_{3x+2} = e_1^d + e_{x+2}^d + e_{2x+2}^d + e_{3x+2}^d + e_{3x+2}^p$ when d_{3x+2}' and p_{3x+2}' enter into the decoder. From the above equations we notice that if we constitute the sum of S_{2x+1} and S_{3x+1} (modulo 2) , we will have four equations in which the term e_1^d appears, while no other bit error term appears more than once. Such equations are called "orthogonal" on e_1^d . It is this property of orthogonality that permits their solution for e_1^d if not more than two errors are present in the bits entering the four equations. These equations are rewritten below:

$$\begin{aligned} S_1 &= e_1^d + e_1^p \\ S_{x+1} &= e_1^d + e_{x+1}^d + e_{x+1}^p \\ S_{2x+1} + S_{3x+1} &= e_1^d + e_{2x+1}^p + e_{3x+1}^d + e_{3x+1}^p \\ S_{3x+2} &= e_1^d + e_{x+2}^d + e_{2x+2}^d + e_{3x+2}^d + e_{3x+2}^p \end{aligned} \tag{4.5}$$

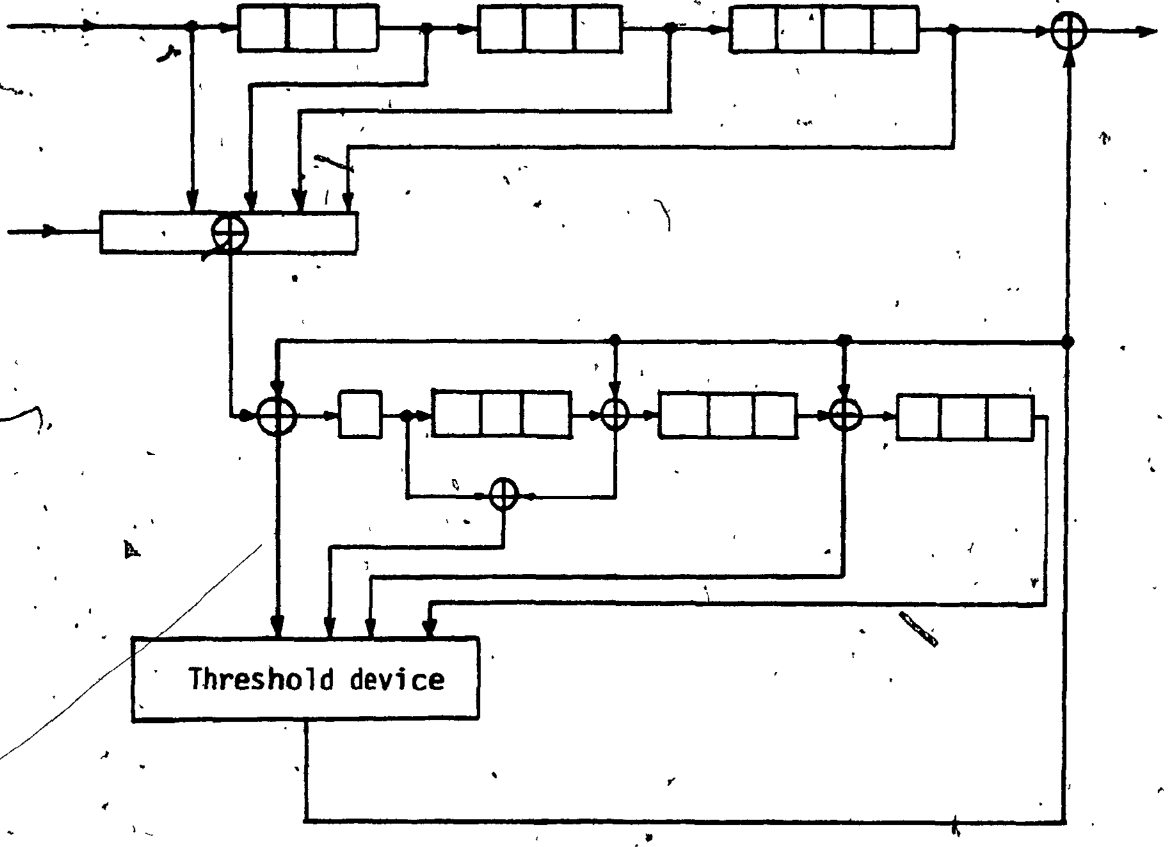
Since the decoder at the instant shown is only interested in the value of e_1^d, solving for it is the decoding operation.

The solution is implemented by the logic indicated in the figure as the threshold device, that is, if more than two of the four expressions forming (4.5) have the value 1, we decide that $e_1^d = 1$ and make a correction in d_1'. This decision is correct if at most, two of the eleven different error bits in (4.5) are in error. Otherwise, the decoding may succeed or fail depending upon the specific error pattern. When the threshold circuit has produced an output, meaning that the received d_1' is wrong, it not only changes d_1' but also complements the values of syndromes, S_{3x+2}, S_{2x+1} and S_{x+1}. These are the three syndromes that contain e_1^d and that will be involved in later decoding decisions. If the decoding decision is correct, the effect of complementing these three bits is to remove the effect of the incorrect bit d_1' from the decoder. Thus, when the decoder goes on to consider later bits, its state is precisely what it would have been if d_1' had not been wrong. In this sense, each correction is a correction of a first error and this justifies the above assumption that if d_1' or p_1' were wrong, they were first errors.

Now we explain the burst-correcting capability of this code. Suppose an error burst occurs, but that it does not span more than 2x channel digits (x parity and x information digits). Examination of figure (4.5) shows that as the burst moves through the decoder, it can just cause two error bits in S_1 and one error bit in the three other syndrome equations (the other bits are separated from each other by at least x digits). Thus the burst errors can be detected and totally corrected in the absence of other errors. This means that there shall be no errors in the decoder when the burst first enters, and that none will enter until it leaves. Referring to figure 4.2, notice that when we have 2x-digit burst errors in the channel, x of them will be information digits. When these x digits reach to the right most shift register stages of the decoder, the other $3x+1-x = 2x+1$ shift register stages of the decoder should have error free digits. The same amount of error free parity digits should have been entered into the syndrome shift register, so the total error free digits are $4x+2$ when the first digit of burst error is in the stage of leaving the decoder. In order that the last information digit of the burst, digit x , leaves the decoder correctly, 2x more error free digits should enter into the decoder, so that the total amount of $4x+2+2x = 6x+2$ error free digits are required after each burst of length 2x for correcting erroneous data digits. As an example, for x=3, the encoder and decoder for (22,11) Massey-Kohlenberg code are illustrated in figure 4.3.



(a)



(b)

FIG. 4.3 Massey-Kohlenberg; a) encoder, b) decoder for (22,11) Code

Lynon and Beaudet [35] recently studied the performance of different codes on aeronautical satellite data links. Their results indicates that the Massey-Kohlenberg code has the least implementation difficulty and is very well suited to these channels.

4.2 Gallager Code

The Gallager code [25], [16] is a $(mn_0, m(n_0-1))$ convolutional code with rate $R = \frac{n_0-1}{n_0}$ which is capable of correcting random errors and almost all burst errors using an adaptive decoding scheme which was devised by Gallager himself. For this code $m = L+M+k_1+1$ where L, M and k_1 are design parameters. They will become clear when we study encoder and decoder circuits. This code is capable of correcting "t" random errors (depending on the first k_1 shift register stages of the encoder and its tap configuration), and $b \leq n_0 L$ burst errors if $g = n_0(L+M+k_1)$ error free digits are available on either side of the burst. It is clear that "t" random error correcting is possible when "t" errors are confined to $n_0(k_1+1)$ consecutive digits. For ease of explanation we consider the Gallager code with rate $R = 1/2$.

The operation of the encoder is similar to that described for the Massey-Kohlenberg code. As each new information bit enters the encoder it is also simultaneously transmitted over the channel along with the corresponding parity bit. This parity bit is the sum of (modulo 2) the oldest bit and the new bits in the tapped positions at the left end of the register. Figure 4.4 illustrates a general encoder for the Gallager code with rate $R = 1/2$.

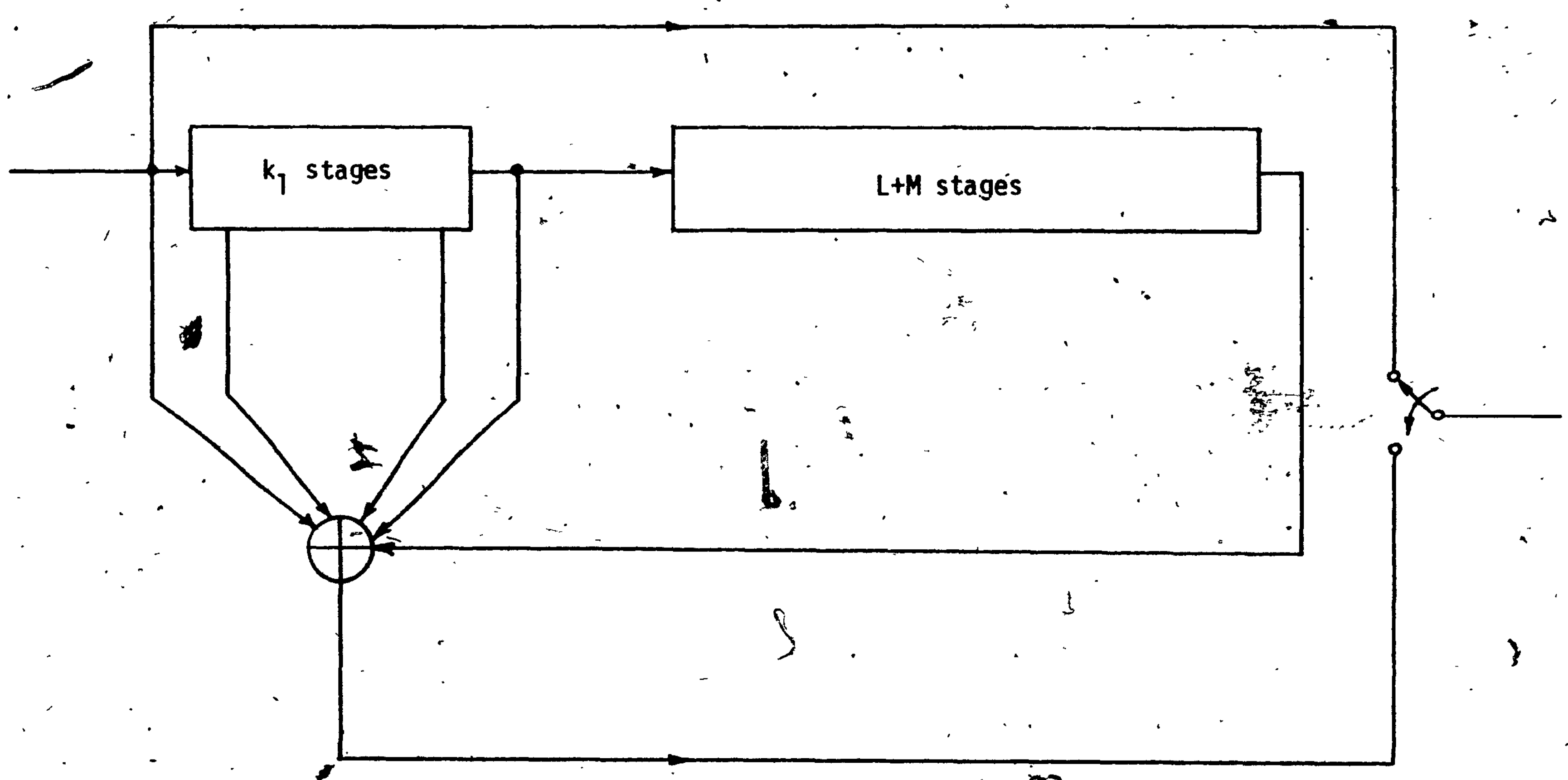


FIG. 4.4. A General Encoder for Gallager Code with rate 1/2

The operation of the decoder for the Gallager code with rate $R = 1/2$ is given in figure 4.5. The figure shows the decoder in its normal form in which the received parity digit as well as the present and past information digits are used to form syndromes. The values of the syndromes are used as decoding criterion.

The decoder can operate in two modes: random and burst. In either mode it uses the values of certain syndromes to decide whether to move ahead without change, to perform a correction, or to shift to the other mode.

In order to understand the functioning of the circuit in figure 4.5, suppose that all the shift register stages of the decoder including the syndrome register contain zero digits. When the first information and parity digits enter the decoder, the first syndrome digit will constitute and will be stored in the syndrome register. This information digit will be corrected (if received in error) when it leaves the L-stage shift register of the decoder, provided the syndrome digits in the k_1-stage shift register of the syndrome satisfy the criterion of random error correcting mode. Therefore, the correct information digits will enter to M-stage shift register of the decoder and will leave the decoder after M time units delay. For each error correction the related syndromes will be reset. This means that the values of the syndromes will be complemented in order to remove the effect of the corrected error from the syndromes which will be involved in later decoding decisions. If the criterion does not satisfy the random mode but can detect burst errors, a signal will turn off gate "2" and turn

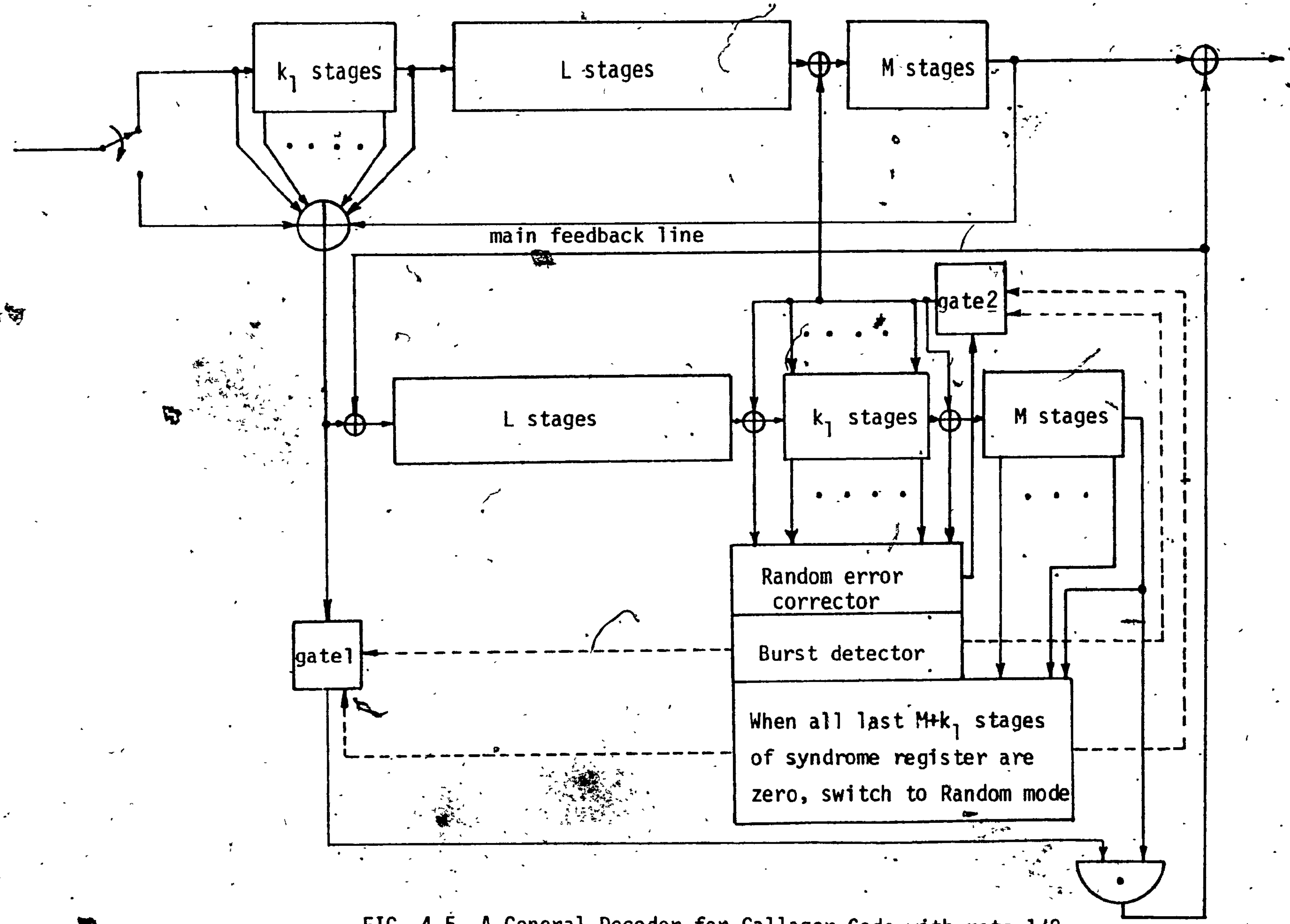


FIG. 4.5 A General Decoder for Gallager Code with rate 1/2

on gate "1", switching from random mode to burst mode. The incorrect information digits will enter into the M-stage shift register without any correction at this stage. Since the length of the burst should be $b \leq 2L$ then, at most, L information digits are in error and after M time units all of them will occupy at most the L right most shift register stages of the decoder. This implies that all digits in k_1+M left most shift register stages of the decoder must be error free and the new syndrome digit, from the error point of view, depends only on the information digit which leaves the M-stage shift register of the decoder. If this syndrome digit and the one which is leaving the M-stage shift register of the syndrome satisfy the input of the AND gate, then the error information digit will be corrected and the new syndrome will be reset to zero through the main feedback line. This process will continue until all corrected burst errors leave the decoder and all syndrome digits in the k_1-stage and M-stage syndrome shift registers become zero. At this time a signal will turn on gate "2" and turn off gate "1", indicating a switch from burst mode to random mode. From the above explanation we realize that the Gallager code is constructed from a basic random error correcting code which has a constraint length of $n' = n_0(k_1+1)$. The constraint length of the Gallager code is $n = n'+n_0(L+M) = n_0(L+M+k_1+1)$. Any basic convolutional code can be used as the Gallager code if the required design parameters are followed. In practice $L >> M+k_1$ and is normally quite large, say several hundred, while M is much smaller, say about twenty.

The Gallager code's outstanding virtue is that it requires a very short and adaptive guard space, roughly equal to the actual length of the burst it is correcting. This is a contrast to the other codes which require a guard space about three times the length of the maximum correctable burst. This can be clarified by using Gallager's lower bound Eq. 2.11, for a code with rate $R = \frac{n_0-1}{n_0}$. We find:

$$g/b \geq 2n_0-1 \text{ and since } n_0 \geq 2 \text{, this means } g/b \geq 3 \text{ .}$$

Now if we write the ratio g/b for the Gallager code which we explained above, we find:

$$g/b = \frac{n_0(L+M+k_1)}{n_0 L} = \frac{L+M+k_1}{L}$$

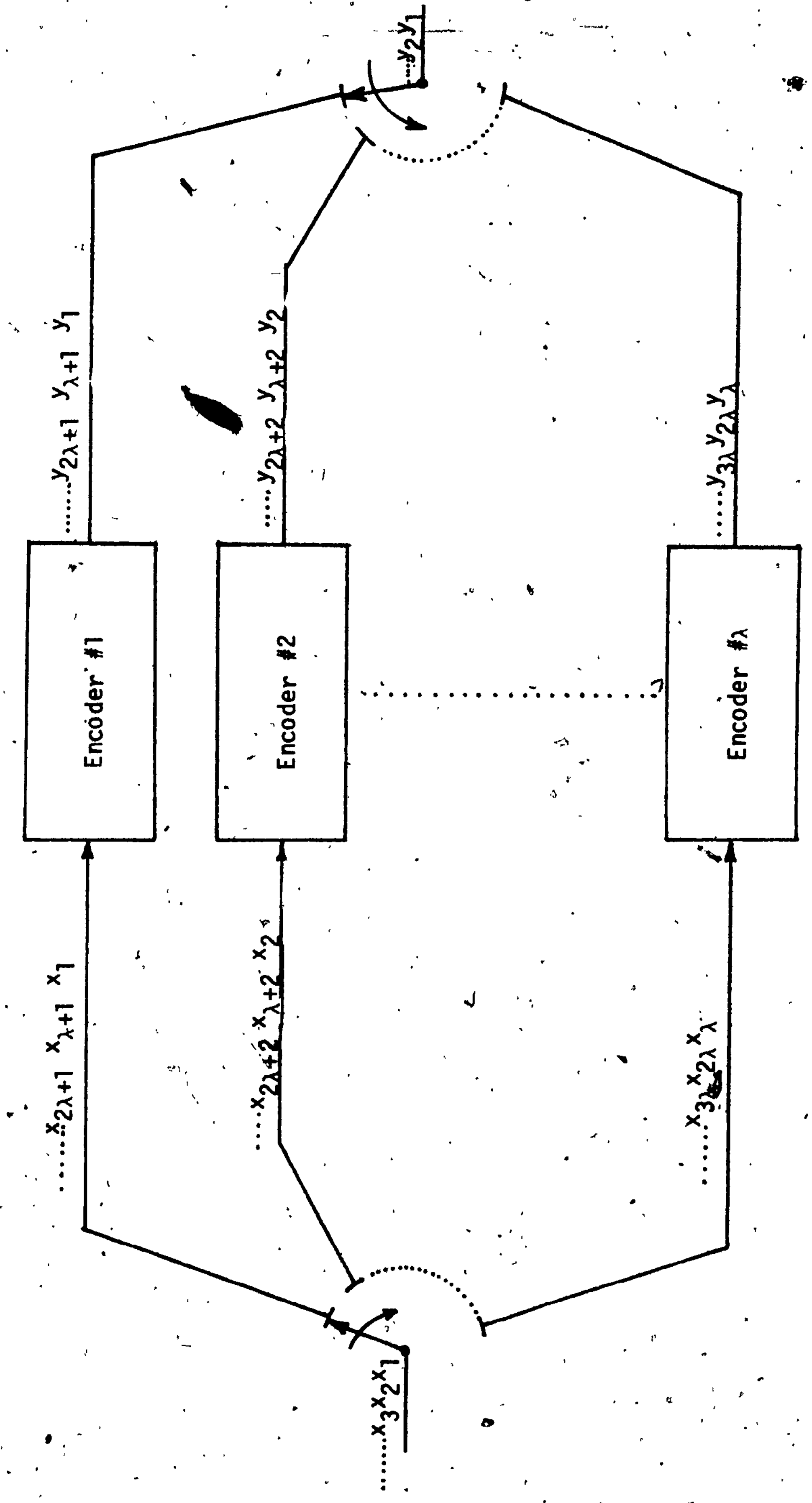
Since $L >> M+k_1$ we conclude that $g/b \approx 1$. This result indicates that the Gallager system will not correct every burst of less than its maximum designed length. Some bursts will begin with error patterns that the random error correcting part of the decoder cannot detect. For these the decoder will not enter the burst mode before the first bit of the burst reaches the last position of the M-stage shift register of the decoder. These bits will therefore not be corrected. Other bursts may be of such a nature that the decoder will leave the burst mode too early, and assuming the burst is over, other errors will be made. However, if suitable criteria are used for burst detection, the errors due to failure of detection will be very infrequent.

Another drawback of the Gallager code is its sensitivity to errors in the guard space. Sullivan [28] generalized Gallager's scheme to allow the presence of random errors in the guard space. This was achieved at a modest sacrifice in rate, but a significant increase in decoder complexity.

4.3 Interleaved Convolutional Codes

The idea of interleaving burst-correcting convolutional codes was first introduced by Hagelbarger [18]. It is understood that the basic idea behind all burst correcting convolutional codes is that the digits involved in the decoding of a particular digit are spread in time so that only one, or at most, a few can be affected by a single burst of errors. Interleaving technique is the simplest way of achieving this spreading. This can be done effectively by breaking the data stream into λ independent streams as shown in figure 4.6. Each of the λ data streams is then separately encoded and the encoded sequences are multiplexed together for transmission on the channel. At the channel output the received data stream is again separated into λ streams. Each stream is separately decoded and the decoded data is finally multiplexed together again. Consequently, any of the coding techniques previously discussed or any other random-error correcting convolutional codes can be used on a burst noise channel in conjunction with interleaving technique. The parameter λ is referred to as the interleaving degree. In practice one does not want to actually build λ separate encoders and decoders but seeks a trick to use one encoder and one decoder in such a way that its operation is equivalent to that of λ separate encoders and decoders.

Interleaving can be achieved by simply inserting λ-1 stage



$\cdots x_3 x_2 x_1$
$\cdots x_{2\lambda+1}\ x_{\lambda+1}\ x_1$
$\cdots x_{2\lambda+2}\ x_{\lambda+2}\ x_2$
$\cdots x_{3\lambda}\ x_{2\lambda}\ x_\lambda$
Encoder #1
Encoder #2
Encoder #λ
$\cdots y_{2\lambda+1}\ y_{\lambda+1}\ y_1$
$\cdots y_{2\lambda+2}\ y_{\lambda+2}\ y_2$
$\cdots y_{3\lambda}\ y_{2\lambda}\ y_\lambda$
$\cdots y_2 y_1$

FIG. 4.6 Interleaved Encoding

delay lines between stages of the original convolutional encoder. The resulting single encoder then generates the λ interleaved codes. This method is of special interest when the threshold decoding [34] technique is used for the decoding of basic codes since the same technique can be employed in the decoder resulting in a single (time-shared) decoder rather than λ decoders. This method is known as the block interleaving technique [32] in which n_0-bit blocks, separated by λ blocks, form an independent data stream. This technique is applicable when the burst error correcting capability of the basic code is $b \geq n_0$. The interleaved code will correct bursts of length $b\lambda$ or less when an error-free guard space of length $g\lambda$ is available on either side of the burst (g is the guard space of the basic code). As an example the basic (27, 18) Hsu code with burst error correcting ability $b=3$ is block interleaved to degree $\lambda=2$. This new code has burst error correcting ability of $b=6$. The corresponding encoder and decoder are shown in figure 4.7 and 4.8, respectively. The original encoder and decoder were shown in figure 3.9 and 3.10.

Another technique which is more flexible is known as bit interleaving technique [11], [33] which is a true interleaving technique. This method separates each digit by $\lambda-1$ intervening digits and is applicable to any basic code with any error correcting ability provided that n_0 divides $\lambda-1$. Figure 4.9 illustrates a general bit interleaved convolutional encoder and decoder where each stage of the shift register of the original encoder and decoder are replaced with λ stages shift

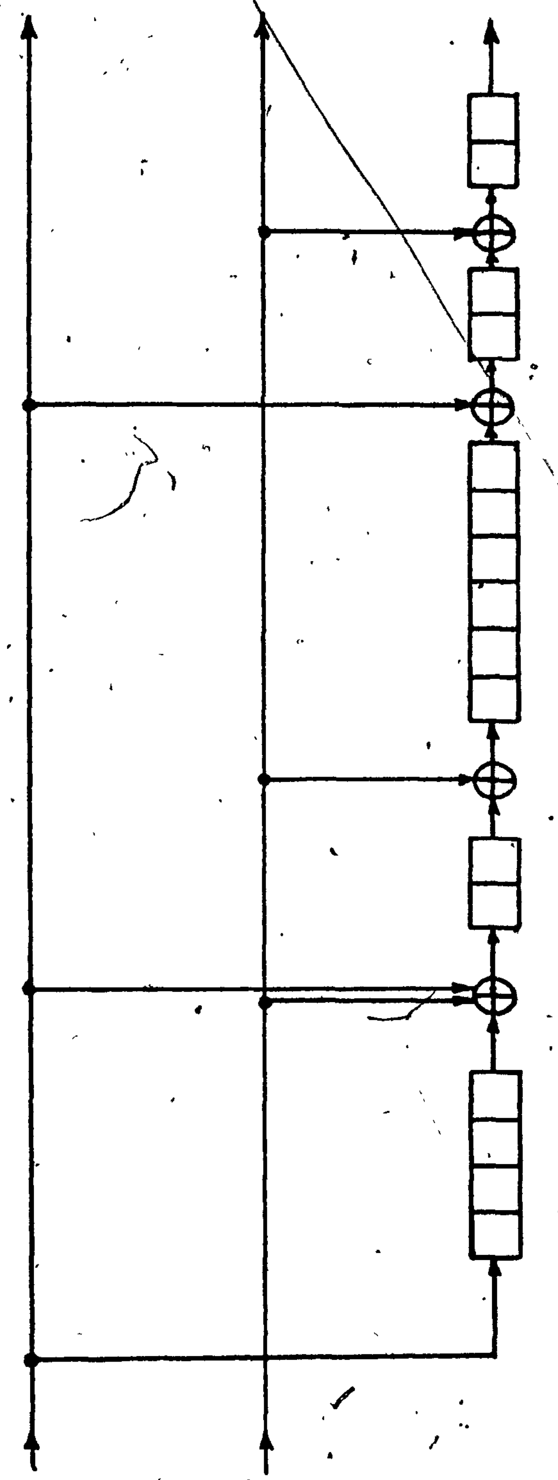


FIG. 4.7 Block Interleaved Encoder for Basic (27,18) Hsu Code, $\lambda=2$

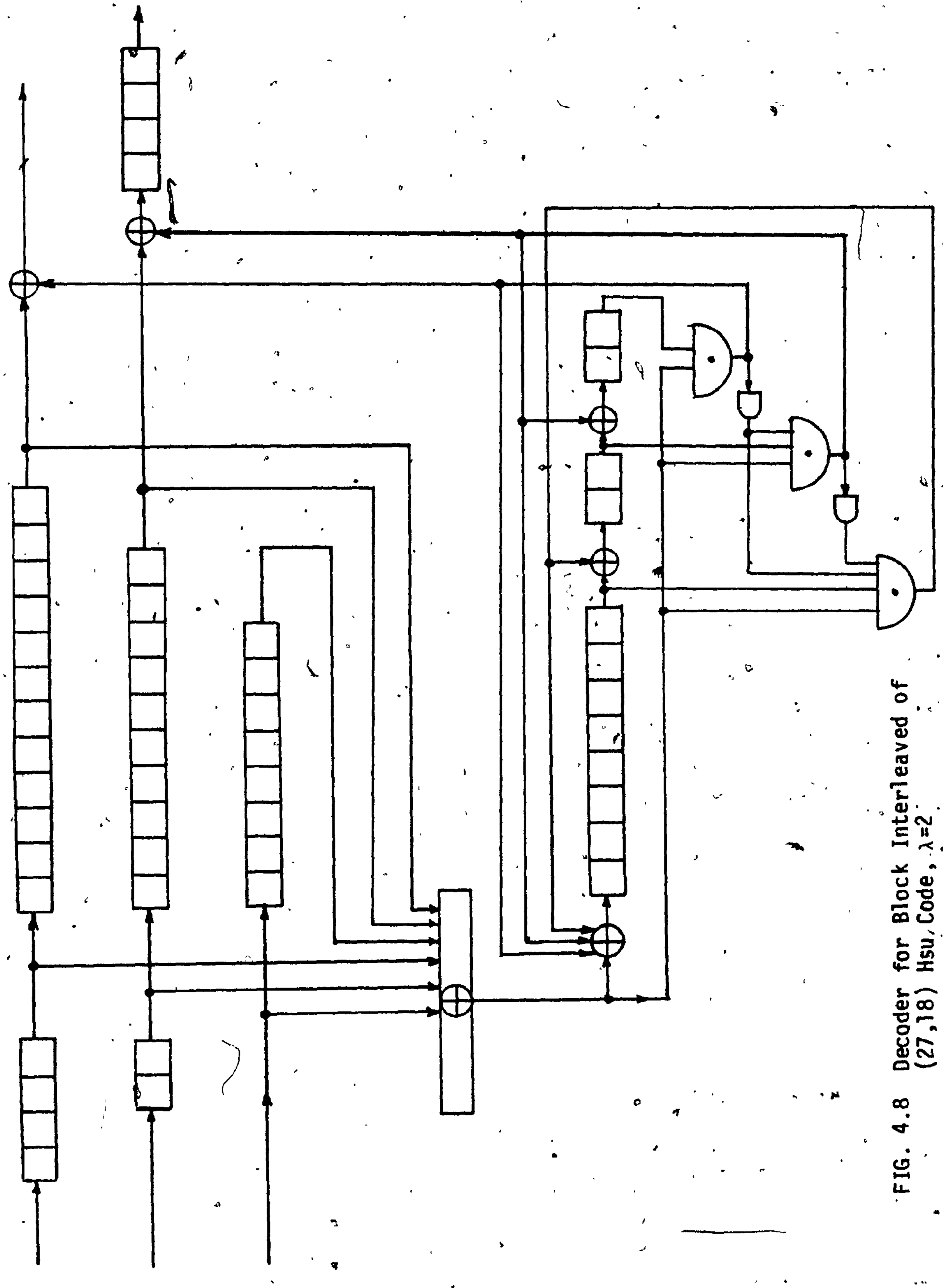
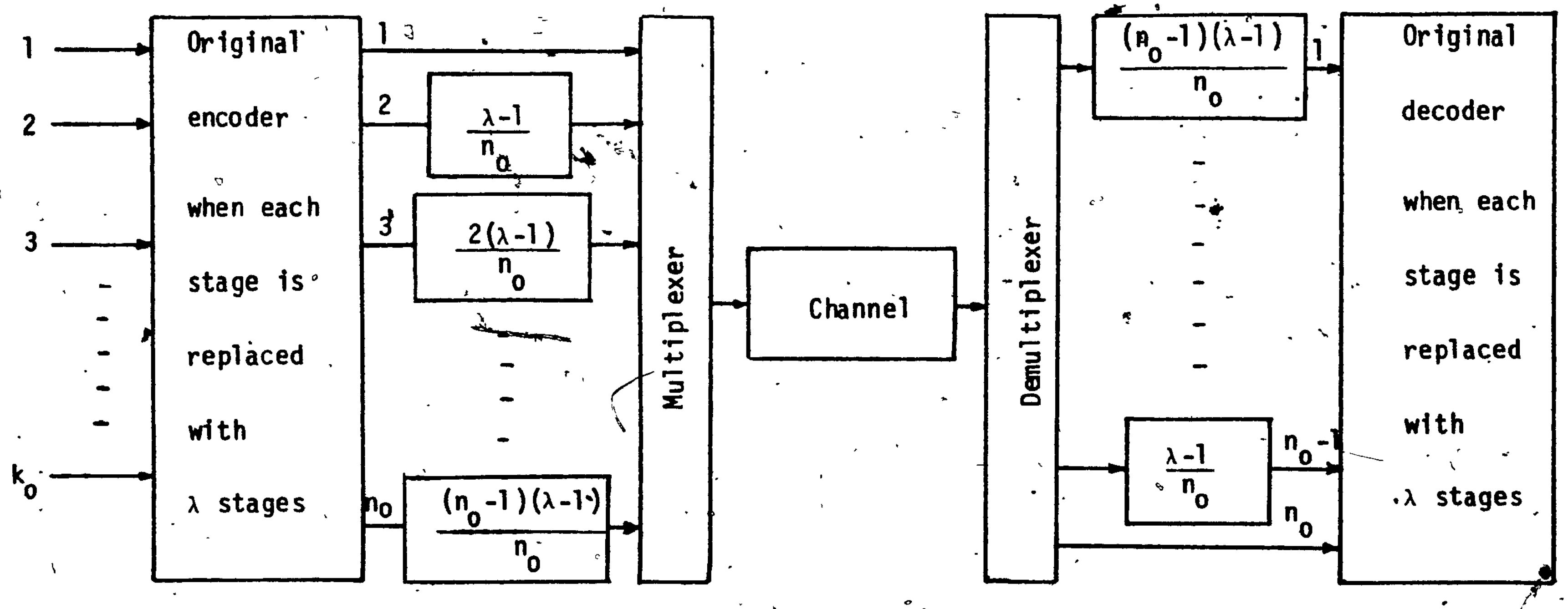


FIG. 4.8 Decoder for Block Interleaved of (27,18) Hsu Code, λ=2

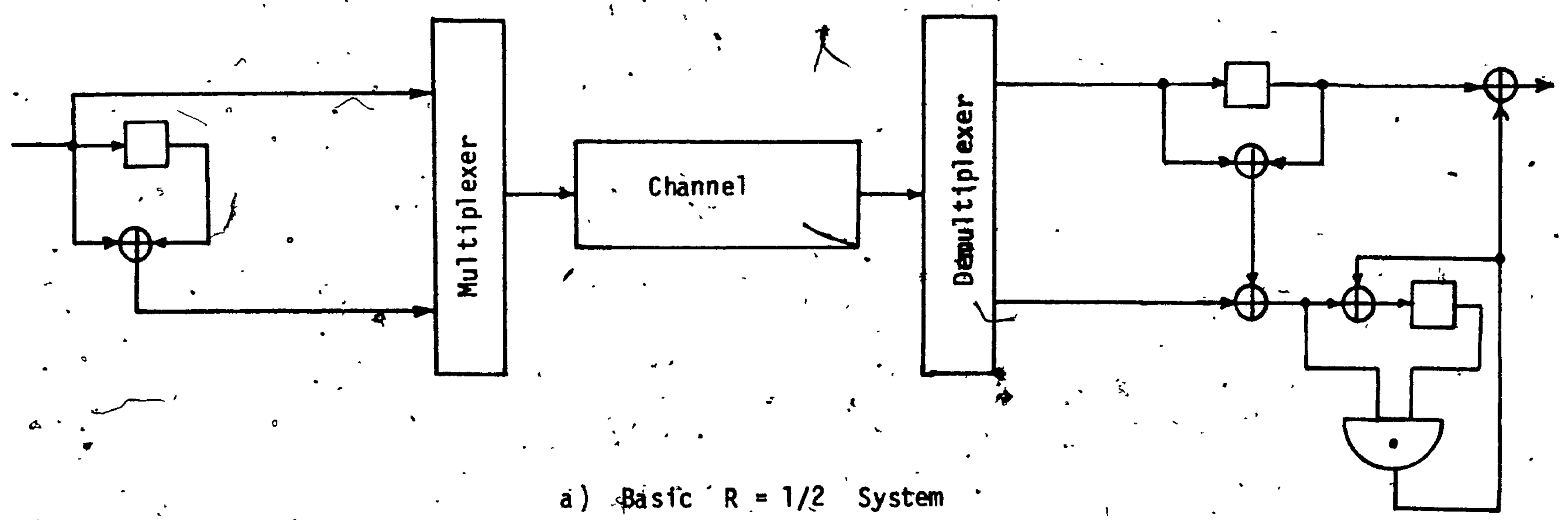

1
2
3
k_o
Original encoder when each stage is replaced with λ stages
1
2
3
n_o
$\frac{\lambda-1}{n_o}$
$\frac{2(\lambda-1)}{n_o}$
$\frac{(n_o-1)(\lambda-1)}{n_o}$
Multiplexer
Channel
Demultiplexer
$\frac{(n_o-1)(\lambda-1)}{n_o}$
$\frac{\lambda-1}{n_o}$
1
n_o-1
n_o
Original decoder when each stage is replaced with λ stages

Note: n_o must divide $\lambda-1$

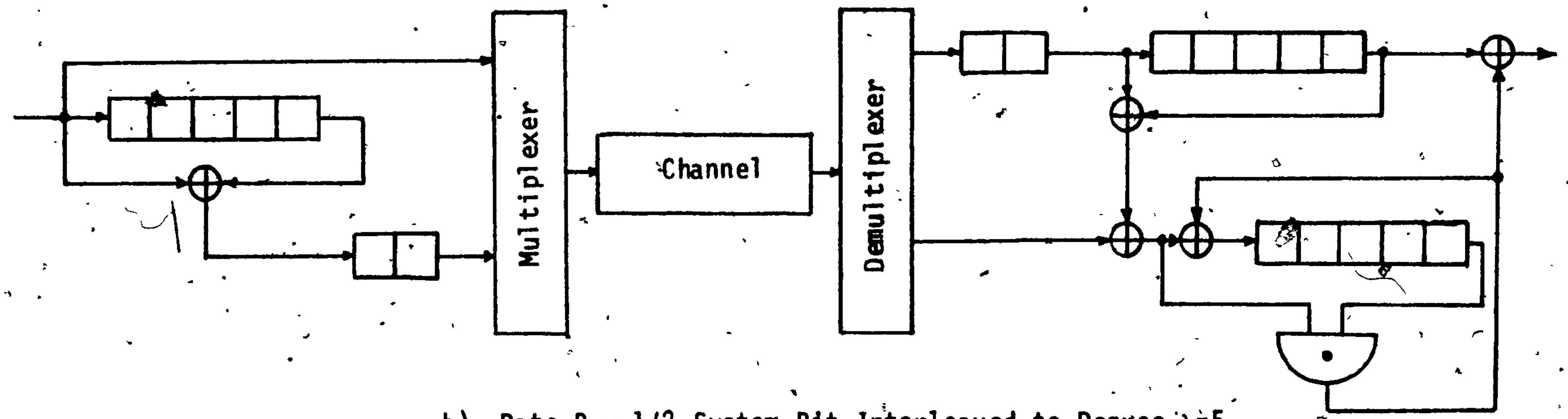
FIG. 4.9 Bit Interleaved Convolutional Coding System

register and the additional $\frac{(n_0-1)(\lambda-1)}{2}$ stages shift register are used after the output of the original encoder, known as interleaver. The same amount of stages shift register are used before the input of the original decoder, known as de-interleaver. The function of the interleaver is to ensure that the n_0 digits in each subblock formed by the encoder are separated by $\lambda-1$ intervening digits for transmission over the channel. The modification of the original encoder actually corresponds to λ distinct encoders whose subblocks are successively formed by the new single encoder. This ensures that there will also be $\lambda-1$ intervening digits between the last digit in a subblock and the first digit of the next subblock of the same one of the λ different convolutional codes as the digits are sent over the channel. As an example the encoder and decoder for a (4,2) convolutional code with rate $R = 1/2$ with error correcting ability of $t=1$, when $g=3$ error free digits are available on either side of each error, are shown in figure 4.10a. The corresponding bit interleaved convolutional encoder and decoder for $\lambda=5$ are illustrated in figure 4.10b. This code is capable of correcting burst errors of length 5 or less when a guard space of length 15 digits is available on either side of each burst.

The above mentioned techniques are perfectly acceptable techniques when threshold decoding is used but are not attractive for the more powerful Viterbi maximum likelihood decoding technique [14]. The reason of course is that the latter must store and constantly update a relatively large amount of information in the form of path metrics and hypothesized information sequences. If the decoder is faced with decoding several independent bit streams it must either store this



a) Basic R = 1/2 System



b) Rate R = 1/2 System Bit Interleaved to Degree λ=5

FIG. 4.10 Bit Interleaved Convolutional Coding System for (4,2) Code

information for each stream or have a sufficient speed advantage so that it can re-compute a considerable portion of this information as it moves from stream to stream. A more desirable system is to interleave so that only one encoded message need be provided to the decoder. This is easily done by inserting an external interleaver-deinterleaver between the encoder and decoder, respectively. A good reference regarding interleaves-deinterleavers (scramblers-unscramblers) can be found in a paper by Ramsy [36].

Goldberg, Moyes and Quigley's [37] experiment shows that rate 1/2 interleaved convolutional code with Viterbi decoding has better performance over troposcatter channels when compared with the Massey-Kohlenberg and Gallager codes. Their results are given in table 4.1 and the related diagram is shown in figure 4.11.

Burst-error-correcting codes using interleaved Viterbi decoding is currently used in most mobile military satellite communication systems for multipath immunity and error control and is predicted as possible in future technology (1980-90) for commercial application using current techniques with improved performance and at reduced cost [38].

TABLE 4.1

DECODED OUTPUTS BIT-ERROR-RATES (BER) AND RELATED IMPROVEMENT FACTORS (F) DUE TO CODING EFFECTS IN TROPOSCATTER SYSTEMS

Ch. Error Rate	Massey-Kohlenberg		Gallager		Interleaved, Viterbi (7-stage encoder)	
	Inf. Error Rate	F	Inf. Error Rate	F	Inf. Error Rate	F
2.2×10^{-6}	9.2×10^{-7}	2.4	2.9×10^{-6}	0.8	6.3×10^{-7}	3.5
9.6×10^{-6}	2.4×10^{-8}	399	1.9×10^{-7}	50.3	0.00	∞
1.1×10^{-5}	9.0×10^{-6}	1.3	8.6×10^{-6}	1.3	7.2×10^{-6}	1.6
1.8×10^{-5}	2.4×10^{-6}	7.3	9.2×10^{-6}	1.9	2.9×10^{-7}	61.6
2.4×10^{-5}	1.7×10^{-5}	1.3	1.9×10^{-5}	1.1	2.8×10^{-5}	0.8
3.6×10^{5}	3.8×10^{-6}	9.4	1.0×10^{-6}	35.1	0.00	∞
7.4×10^{-5}	1.7×10^{-5}	4.3	9.1×10^{-7}	81.2	0.00	∞
1.0×10^{-4}	3.8×10^{-6}	26.5	9.8×10^{-6}	10.2	1.6×10^{-6}	63.7

$$\text{Improvement Factor (F)} = \frac{\text{Channel Error Rate}}{\text{Output Information Error Rate}}$$

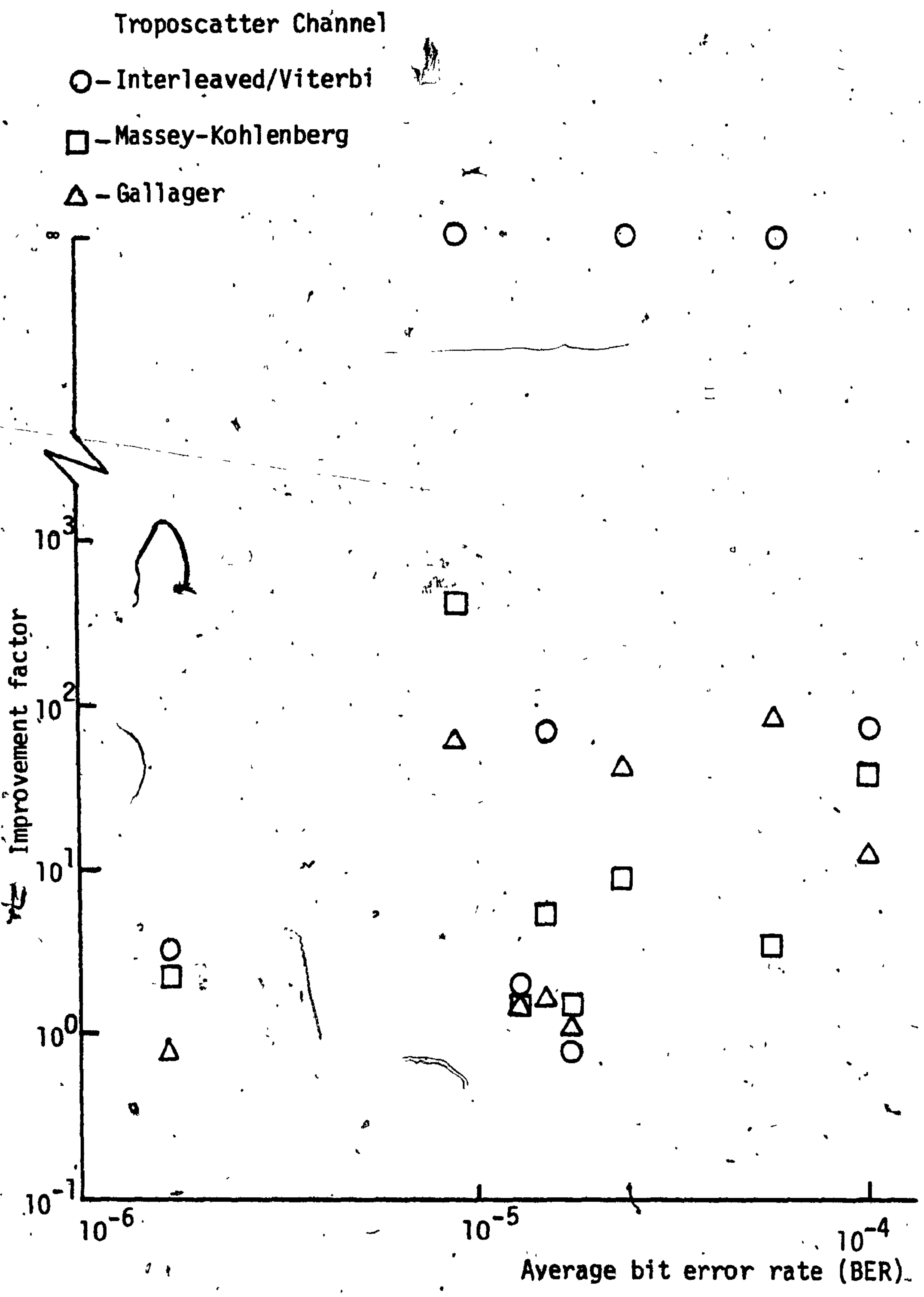


FIG. 4.11 Diagram of the results shown in Table 4.1

CHAPTER 5

SUMMARY AND CONCLUSIONS

In this report we have considered burst-error-correcting convolutional codes which are in use in digital communication systems. Some important definitions and mathematical background which are required to analyze these codes were given in chapter 2.

In chapter 3, we discussed some FEC convolutional codes which are good for burst-error-correction. These codes have rate $\frac{n_0-1}{n_0}$ and burst-error-correcting ability given by burst length $b \leq n_0$. Among these codes, Rodgers code has the least required guard space of $g = 3n_0^2-2n_0-1$ but it is difficult to implement. On the other hand, Hsu code has the least implementation difficulties but is inferior to Rodgers code when the comparison is based on the required guard space $(g = 3n_0^2-1)$.

The FEC convolutional codes with random and burst-error correction ability were studied in chapter 4. These codes have practical use and are applicable to real communication channels. For rate 1/2 in the Gallager code, we noticed that the required guard space is approximately equal to the burst length $g \approx b \leq n_0$ (which is the smallest possible guard space achievable). This is also smaller than the guard space given by the Wyner and Ash, and Gallager bounds. This is obvious because this code could not correct all bursts of length b. From the points of view of implementation and of less-sensitivity to errors in the guard space the Massey-Kohlenberg code seems more advantageous than the Gallager code. Lastly, the interleaving technique described in this chapter, makes it possible to use any existing convolutional code for burst-error-correction. An additional advantage of this technique is that it provides the existing basic codes the ability of correcting burst errors that are longer than those considered in chapter 3, provided that the bounds are met.

Due to advancement in LSI and microprocessor technology and their cost effectiveness as well as the demand for accurate high speed data transmission it seems that Interleaved Viterbi Convolutional Code will be used in future digital communication systems where there is a requirement for compensating the degradation of the system due to the occurrances of the burst errors.

APPENDIX "A"

PROOF OF WYNER AND ASH BOUND

Due to Wyner and Ash [4], the following assumptions and theorems are required to prove inequality 2.9 given in chapter 2.

Assumptions:

1) Type B_2 code with $i=1$ has been constructed where $b \leq rn_0$ is the burst length.

2) B' = the $(N+r-1)\times n_0$ matrix consisting of the first $N+r-1$ rows of B. Note that at least the last $r-1$ rows of B' are zero.

3) T' = the following $(N+r-1)\times(N+r-1)$ matrix:

$$T' = \begin{bmatrix} 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & & & & \vdots \\ 0 & 1 & 0 & & O & \vdots \\ \vdots & & \ddots & \ddots & & \vdots \\ \vdots & O & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & 0 \end{bmatrix}$$

4) $H'_N = [B', T'B', T'^2B' \ \ldots \ T'^{N+r-2}B']$

H'_N is given schematically in figure A.1. The fth column of H'_N is $C'(f)$.

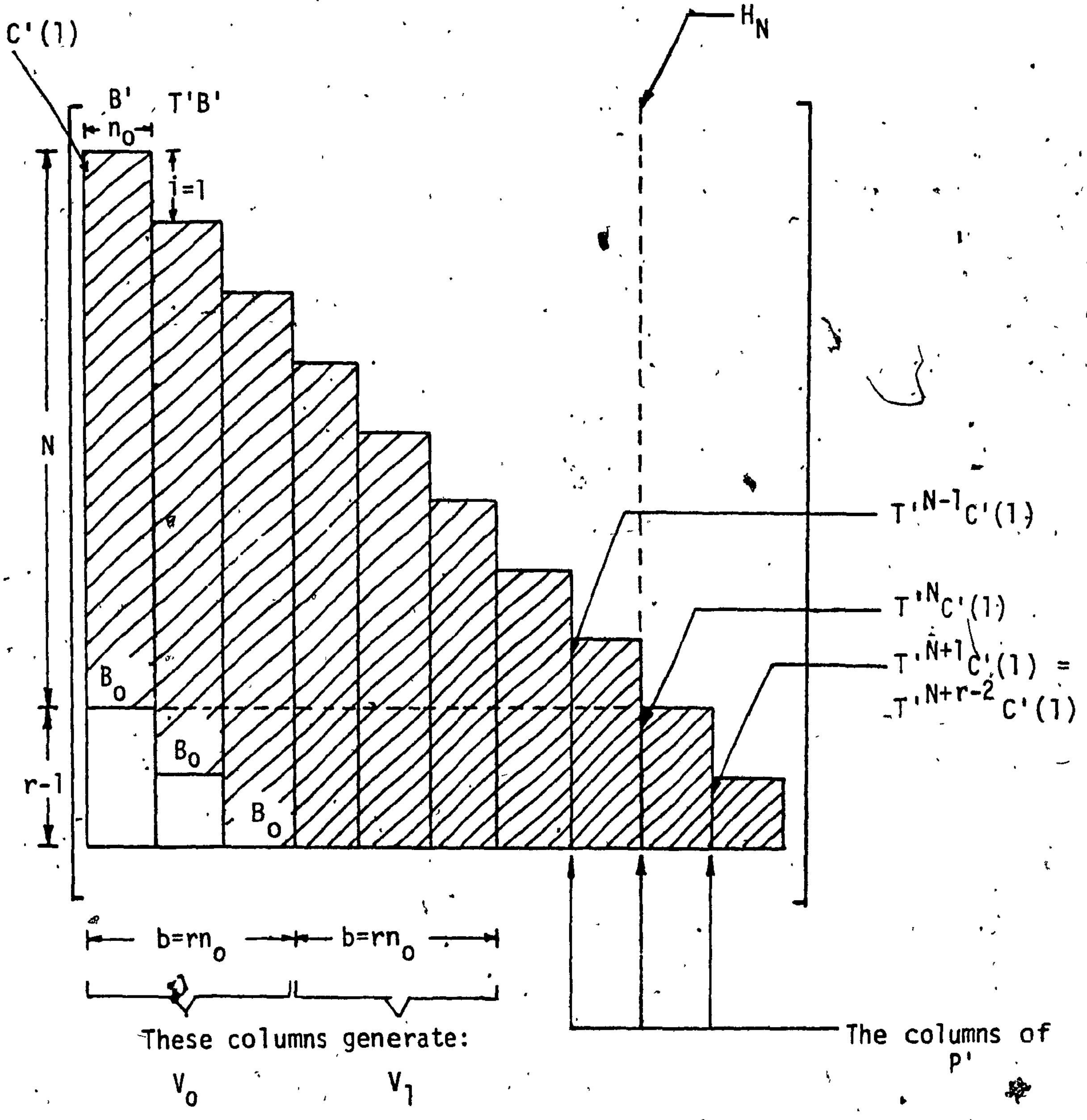


FIG. A.1 Schematic diagram of the matrix H_N' for r=3. The nonzero entries are within the shaded areas

5) V_o = the space of $(N+r-1)$-vectors spanned by the columns of B', $T'B'$ ... $T'^{r-1}B'$ = the first b columns of H_N' (linear independent vectors).

6) V_1 = the space of $(N+r-1)$-vectors spanned by the columns of $T'^{r}B'$, $T'^{r+1}B'$ ..., $T'^{2r+1}B'$ = the second set of r blocks of H_N'.

Remarks:

1) $[C'(f)]_N = C(f)$ if $f \leq Nn_o$ and $[C'(f)]_N = 0$ if $f > Nn_o$ ($C(f)$ is the fth column of H_N. It is also assumed that the top entry of $C(1)$ is "1").

2) If $Z_1' = \sum_{f\varepsilon F} C'(f)$, then:

$$[Z_1']_N = [\sum_{f\varepsilon F} C'(f)]_N = \sum_{f\varepsilon F} [(C'(f)]_N = \sum_{f\varepsilon F\cap K_{Nn_o}} C(f)$$

is a linear combination of columns of H_N. We now restate the necessary condition of theorem 1, mentioned in section 2.7 of chapter 2, in terms of the H_N' matrix.

<u>Theorem 2</u>: If $z_1' = \sum_{f\varepsilon F} c'(f)$ and $z_2' = \sum_{j\varepsilon J} c'(j)$ are type B_2 correctable linear combinations of columns of H_N', then:

$$z_1' = z_2' \text{ implies } F \cap K_b = J \cap K_b$$

Proof: The proof parallels that of theorem 1. If $z_1' = z_2'$ and $F \cap K_b \neq J \cap K_b$, then z_1' and z_2' form a truncated syndrome S_{N+r-1} corresponding to two different sets of errors in the first r blocks.

This contradicts assumption 2 in section 2.7. Hence $z_1' = z_2'$ implies $F \cap K_b = J \cap K_b$. Following a sequence of lemmas, proved by Wyner and Ash, along with theorem 2, will lead us to the lower bound on N.

Lemma 1: Any type B_2 code with $i=1$ must be such that $N>r$.

Proof: We assume the contrary, i.e., $N \leq r$ and consider the matrix P as follows:

$$P = [C(1), C(n_0+1), \ldots, C((N-1)n_0+1)]$$

The columns of P are the first columns of each of the N blocks of H_N. Matrix P has the following properties:

1) P is triangular since the top entry of $C(1)$ is "1". Hence, the columns of P are N independent N-vectors and thus span the vector space of N-vectors (over the modulo two field).

2) The columns of P are from the first "b" columns of H_N since $(N-1)n_0+1 \leq (r-1)n_0+1 = b-n_0+1 \leq b$. Any linear combination of columns of P is type B_2 correctable, since the columns are confined to r blocks. Since $C(2)$ is an N-vector. 1) yields

$$C(2) = \sum_{f \varepsilon F} C(f)$$

where the $C(f)$ are columns of P. By 2) F is type B_2 correctable. But $F \cap K_{n_0} = \{1\}$ and $\{2\} \cap K_{n_0} = \{2\}$. This contradicts the necessary condition of theorem 1. We conclude $N>r$.

Lemma 2: If z_1' is a nontrivial linear combination of the generators of

V_0 (i.e., of the first b columns of H_N'), then $[z_1']_{N-1} \neq 0$.

Proof: Assume the lemma is false so that $[z_1']_{N-1} = 0$. Consider the matrix:

$$P' = [T'^{N-1}C'(1), T'^{N}C'(1) \dots T'^{N+r-2}C'(1)]$$

The columns of P' are the first columns of the last r blocks of H_N'. It follows from the definition of T' that $[P']_{N-1} = 0$. Also, since we are assuming that the top entry of $C(1)$ is 1, hence $C'(1)$ has top entry "1". As well, the last r rows of P' are triangular; hence a nonsingular matrix. It follows that the columns of P' span the set of $(N+r-1)$-vectors whose first $N-1$ entries are zero. Thus, if $[z_1']_{N-1}=0$, we may write z_1' as a linear combination of columns of P', or equivalently:

$$z_1' = \sum_{f \varepsilon F} C'(f)$$

where F is confined to the last r blocks of H_N', i.e., $f \geq (N-1)n_0+1$ for all $f \varepsilon F$. Note that since the columns of P' are restricted to r consecutive blocks of H_N', F is type B_2 correctable. Now $z_1' \varepsilon V_0$ by hypothesis, and consequently:

$$z_1' = \sum_{j \varepsilon J} C'(j)$$

where J is an non-null subset of K_b and hence type B_2 correctable. Thus we will have:

$$\sum_{f \varepsilon F} C'(f) = \sum_{j \varepsilon J} C'(j)$$

where $J \cap K_b \neq \phi$. We claim that $F \cap K_b = \phi$, thus contradicting theorem 2 and establishing the lemma. To verify the assertion, we note that since $f \geq (N-1)n_0+1$ for all $f \varepsilon F$, lemma 1 yields $f > (r-1)n_0+1$. Since each f corresponds to the first column of a block, $f \geq rn_0+1 = b+1$ and thus $F \cap K_b = \phi$.

Q.E.D.

Corollary 1: Dim $V_0 = b$, where dim V_0 is the dimension of the vector space V_0.

Proof: If the b generators of V_0 are not independent, there is a nontrivial linear combination z_1' of these generators which is zero. Thus $[z_1']_{N-1} = 0$, contradicting lemma 2.

Corollary 2: Dim $V_1 = b$

Proof: Let T_0 be the linear transformation from V_0 into the space of all binary $(N+r-1)$-vectors defined by $T_0(z_1') = T'z_1'$, $(z_1' \varepsilon V_0)$. We may write $V_1 = T_0^r(V_0)$, i.e., V_1 is the set of all vectors $T_0^r(z_1')$, $z_1' \varepsilon V_0$. If we can show that T_0^r is nonsingular and thus dimension preserving, then $\dim V_1 = \dim V_0 = b$. It will suffice to show that the kernel of T_0^r contains only the zero vector. Suppose $z_1' \varepsilon$ kernel T_0^r, i.e., $z_1' \varepsilon V_0$ and $T_0^r(z_1') = (T')^r z_1' = 0$. Now it follows from the definition of T' that if $(T')^q z_2' = 0$, the top $N+r-1-q$ entries of z_2' are zero. Thus the top $N+r-1-r = N-1$ digits of z_1' must be zero, i.e. $[z_1']_{N-1} = 0$. But since $z_1' \varepsilon V_0$, $z_1' = 0$ by lemma 2, this concludes the proof.

Lemma 3: $V_0 \cap V_1 = \{0\}$

Proof: If $z_1' \varepsilon V_0$ we write $z_1' = \sum_{f \varepsilon F} C'(f)$ where F is a subset of K_b. If $z_1' \varepsilon V_1$ we write $z_1' = \sum_{j \varepsilon J} C'(j)$ where J is a subset of $K_{2b}-K_b$.

Thus $J \cap K_b = \phi$. Since F and J are type B_2 correctable, $F \cap K_b = \phi$ by theorem 2. Since $F \subset K_b$, $F=\phi$ and therefore $z_1' = 0$.

Theorem 3: Any type B_2 code with $i=1$ must be such that $N \geq 2b-r+1$.

Proof: Let V_0+V_1 be the sum of the subspaces V_0 and V_1, i.e.,

$$V_0+V_1 = \{\alpha z_1'+\beta z_2' : z_1' \epsilon V_0 , z_2' \epsilon V_1 , \alpha,\beta = 0 \text{ or } 1\}$$

A standard theorem of vector space theory states that:

$$\dim(V_0+V_1) = \dim V_0 + \dim V_1 - \dim (V_0 \cap V_1)$$

Since V_0+V_1 is a space of $(N+r-1)$-vectors, $\dim (V_0+V_1) \leq N+r-1$. Thus by lemma 3 and the corollaries to lemma 2 we have:

$$N+r-1 \geq b+b-0 = 2b$$

and the proof is complete. The following generalization of theorem 3 may be derived in a manner analogous to the above procedure.

Theorem 3': Any type B_2 code must be such that:

$$N \geq 2b-(r-1)i$$

consequently

$$n = \frac{N}{i} n_0 \geq \frac{2bn_0}{i} - (r-1)n_0$$

APPENDIX "B"

PROOF OF GALLAGER BOUND

Gallager's approach [16] for proving inequality 2.11, given in chapter 2, is as follows:

Accoridng to Gallager, we assume that there is an arbitrary but finite decoding delay of, say, L information digits. That is, by the time the "u"th information digit enters the decoder, $(u>L)$, at least $u-L$ information digits must be decoded. This condition can be translated into requiring that by the time "M" encoded digits have been received, at least $RM-L$ information digits must be decoded (R is code rate). We also assume that the code has burst-error-correcting capability "b" relative to a guard space "g" , $(g\geq b)$, and the number of received digits, M , is a multiple of $b+g$, i.e., $M = \ell(b+g)$. Two types of error sequences are supposed and shown in figure B.1. In each type, the error sequence is constrained to have zero values in the positions shown and may have arbitrary values in the positions marked "x" . Let X_1 and X_2 be encoded sequences corresponding to different choices of the first $RM-L$ information digits, and E_1 and E_2 be error sequences of the first and second type respectively. Since by assumption these error sequences cannot cause decoding errors, we have:

$$X_1+E_1 \neq X_2+E_2 \pmod 2 \qquad \text{(B.1)}$$

More generally, if E_1 and E_1' are error sequences of the first type and E_2 and E_2' are error sequences of the second type, we must have:

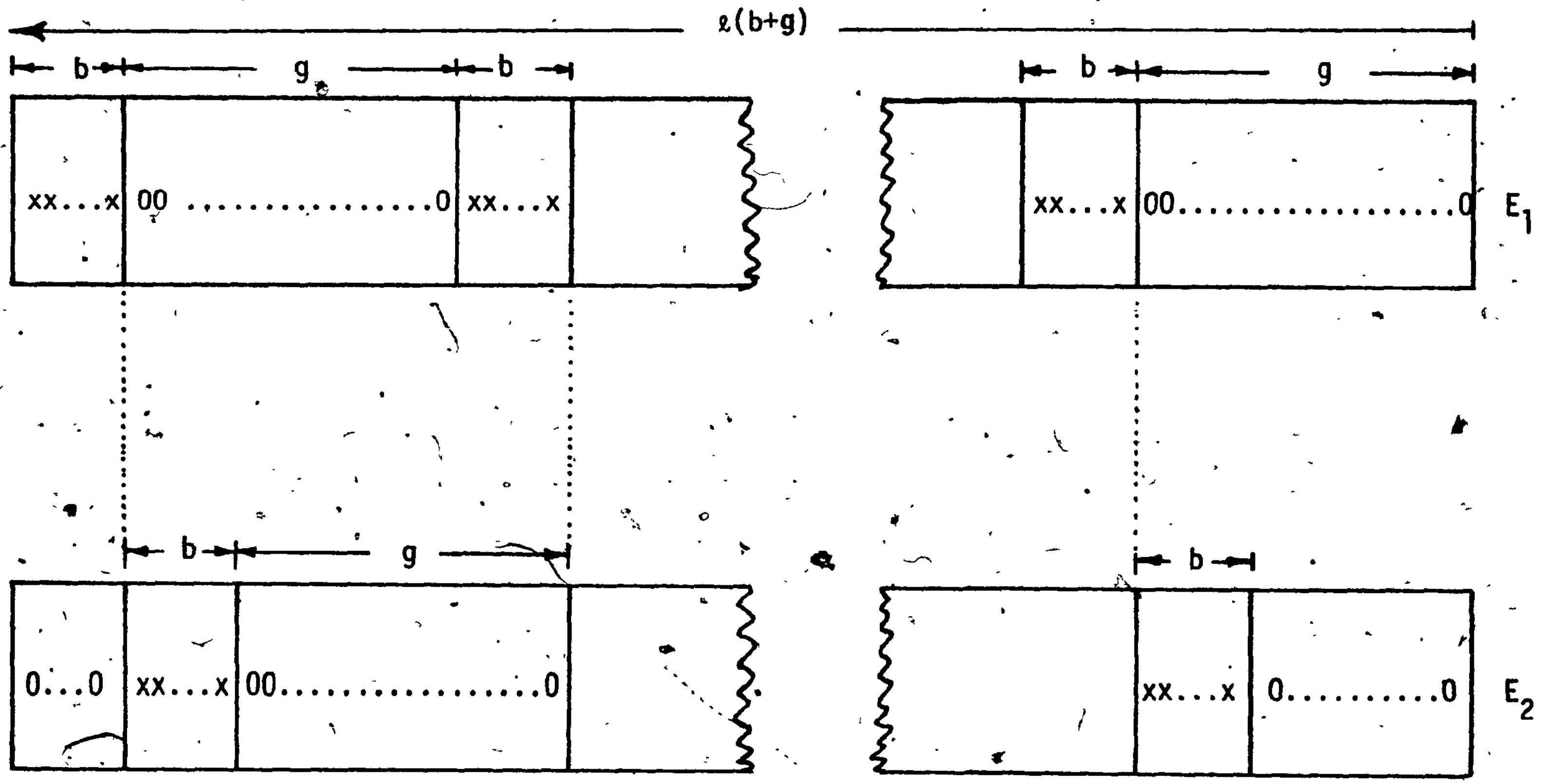


FIG. B.1 Two Types of Noise Sequences; x's Represent Arbitrary Binary Digits

$$X_1+E_1+E_2 \neq X_2+E_1'+E_2' \qquad (B\cdot 2)$$

To establish B.2 , we assume that B.2 is false for some choice of sequences and establish a contradiction. If the equality holds true in B.2 then we can add E_1' and E_2 to both sides of the equation, resulting in

$$X_1+E_1+E_1' = X_2+E_2+E_2' \qquad (B.3)$$

Since E_1+E_1' is an error sequence of the first type and E_2+E_2' is an error sequence of the second type, B·3 contradicts B.1 and thus B.2 is valid. Finally, we observe that if X_1 and X_2 are equal and correspond to the same first RM-L information digits but if either $E_1 \neq E_1'$ or $E_2 \neq E_2'$, then again the inequality holds true in B·2 .

In order to choose X_1, we can consider at least $2^{RM-L} = 2^{R\ell(g+b)-L}$ different ways, each corresponding to a different choice of the first RM-L information digits. Similarly we can consider $2^{\ell b}$ different ways of choosing E_1 and $2^{\ell b}$ different ways of choosing E_2 . From B·2 , each different choice of the triple (X_1, E_1, E_2) leads to a different sequence $X_1+E_1+E_2$. Since there are 2^M different received binary sequences of length M , we have the inequality:

$$2^{RM-L}\, 2^{\ell b}\, 2^{\ell b} \leq 2^M \qquad (B.4)$$

or $$\log 2^{RM-L+2\ell b} \leq \log 2^M \qquad (B.5)$$

or $$RM-L + 2\ell b \leq M \qquad (B\cdot 6)$$

We substitute $\ell(g+b)$ for M, thus

$$R(g+b) - \frac{L}{\ell} + 2b \leq g+b \tag{B.7}$$

Since L is fixed but B.7 must be satisfied for all $\ell \geq 1$, we can pass to the limit $\ell \to \infty$, obtaining

$$2b \leq (g+b)(1-R) \tag{B.8}$$

or

$$g/b \geq \frac{1+R}{1-R} \tag{B.9}$$

which is Gallager bound.

APPENDIX "C"

B_o MATRICES OF CONVOLUTIONAL CODES WITH MINIMUM CONSTRAINT LENGTH

This appendix contains all known codes of rate $\frac{n_o-1}{n_o}$ with constraint length $n = 2n_o^2$ having burst-error-correction ability of $b \leq n_o$. Three out of four codes of rate 1/2 and all codes of rate 2/3 were found by Berlekamp [21]. One code of rate 1/2 and 18 codes of rate 2/3 were found by Rodgers [5]. Two other codes of rate 3/4 were also found by Mandelbaum [30]. The B_o matrices of these codes are given below.

B_o Matrix for Rate 1/2 Codes

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}}_{\text{Berlekamp}} \qquad \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}}_{\text{Rodgers}}$$

<u>B_0 Matrix for Rate 2/3 Codes</u>

All found by Berlekamp

$$\begin{bmatrix}0&1&1\\0&1&0\\1&1&0\\0&0&0\\1&0&0\\1&0&0\end{bmatrix},\quad \begin{bmatrix}0&0&1\\1&1&0\\0&1&0\\1&0&0\\0&1&0\\0&1&0\end{bmatrix},\quad \begin{bmatrix}1&1&1\\1&0&0\\1&1&0\\1&0&0\\1&0&0\\0&1&0\end{bmatrix},\quad \begin{bmatrix}1&0&1\\1&0&0\\1&0&0\\0&1&0\\0&0&0\\0&1&0\end{bmatrix}$$

$$\begin{bmatrix}0&1&1\\1&1&0\\1&0&0\\1&1&0\\0&0&0\\0&1&0\end{bmatrix},\qquad \begin{bmatrix}1&0&1\\1&0&0\\0&1&0\\0&0&0\\1&0&1\\1&1&0\end{bmatrix}$$

<u>B_0 Matrix for Rate 3/4 Codes</u>

$$\underbrace{\begin{bmatrix}0&0&1&1\\0&1&1&0\\1&1&0&0\\0&1&0&0\\1&0&1&0\\1&1&0&0\\0&0&1&0\\1&0&0&0\end{bmatrix},\ \begin{bmatrix}0&0&1&1\\1&1&1&0\\0&1&1&0\\0&0&1&0\\1&1&1&0\\0&0&0&0\\1&1&1&0\\0&1&0&0\end{bmatrix}}_{\text{Mandelbaum}}\ \underbrace{\begin{bmatrix}1&1&1&1\\0&1&0&0\\1&1&0&0\\1&0&1&0\\0&1&0&0\\0&0&0&0\\0&0&1&0\\1&0&1&0\end{bmatrix},\ \begin{bmatrix}1&1&1&1\\0&0&1&0\\1&0&1&0\\1&1&0&0\\0&0&1&0\\0&0&0&0\\0&1&0&0\\1&1&0&0\end{bmatrix}}_{\substack{\text{Rodgers, also}\\ \text{the other next 16 codes}}}$$

$$\begin{bmatrix} 0&0&1&1\\ 1&1&1&0\\ 1&0&0&0\\ 1&1&0&0\\ 0&1&1&0\\ 1&0&0&0\\ 0&0&1&0\\ 0&1&0&0 \end{bmatrix}, \begin{bmatrix} 0&1&0&1\\ 0&0&1&0\\ 0&1&1&0\\ 0&0&0&0\\ 1&1&0&0\\ 1&0&0&0\\ 1&0&1&0\\ 1&0&1&0 \end{bmatrix}, \begin{bmatrix} 1&0&1&1\\ 1&0&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&1&1&0\\ 1&1&0&0\\ 1&1&0&0\\ 0&1&0&0 \end{bmatrix}, \begin{bmatrix} 1&1&0&1\\ 0&0&1&0\\ 1&0&1&0\\ 1&1&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&1&0&0\\ 1&1&0&0 \end{bmatrix}$$

$$\begin{bmatrix} 0&1&0&1\\ 1&1&0&0\\ 1&0&0&0\\ 0&1&1&0\\ 0&0&0&0\\ 0&0&1&0\\ 1&0&0&0\\ 1&1&1&0 \end{bmatrix}, \begin{bmatrix} 1&0&1&1\\ 1&0&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&1&1&0\\ 1&1&0&0\\ 0&1&0&0\\ 0&1&0&0 \end{bmatrix}, \begin{bmatrix} 0&0&0&1\\ 1&0&1&0\\ 0&0&1&0\\ 0&1&0&0\\ 0&1&0&0\\ 1&0&0&0\\ 1&1&1&0\\ 1&0&0&0 \end{bmatrix}, \begin{bmatrix} 1&1&0&1\\ 1&0&0&0\\ 1&1&1&0\\ 0&0&1&0\\ 0&1&1&0\\ 0&0&0&0\\ 0&1&1&0\\ 0&1&0&0 \end{bmatrix}$$

$$\begin{bmatrix} 0&1&1&1\\ 1&1&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 1&1&1&0\\ 1&1&0&0\\ 0&0&0&0\\ 1&0&0&0 \end{bmatrix}, \begin{bmatrix} 0&1&1&1\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 1&1&1&0\\ 1&0&0&0\\ 1&1&0&0\\ 1&1&0&0 \end{bmatrix}, \begin{bmatrix} 1&0&1&1\\ 0&1&0&0\\ 1&1&0&0\\ 0&1&1&0\\ 0&0&1&0\\ 1&0&0&0\\ 0&0&1&0\\ 1&0&1&0 \end{bmatrix}, \begin{bmatrix} 0&0&1&1\\ 1&0&1&0\\ 1&0&0&0\\ 1&1&1&0\\ 1&1&1&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&1&0&0 \end{bmatrix}$$

$$\begin{bmatrix} 0&0&1&1\\ 1&0&1&0\\ 1&0&0&0\\ 1&1&1&0\\ 0&0&0&0\\ 1&0&1&0\\ 0&1&0&0\\ 0&1&1&0 \end{bmatrix}, \begin{bmatrix} 0&0&0&1\\ 0&0&1&0\\ 1&1&0&0\\ 1&0&1&0\\ 1&0&0&0\\ 0&1&1&0\\ 0&1&1&0\\ 1&0&1&0 \end{bmatrix}, \begin{bmatrix} 0&0&1&1\\ 1&1&1&0\\ 1&0&0&0\\ 0&1&0&0\\ 0&0&1&0\\ 0&0&1&0\\ 1&1&0&0\\ 1&1&1&0 \end{bmatrix}, \begin{bmatrix} 0&0&1&1\\ 1&1&1&0\\ 0&1&1&0\\ 0&1&0&0\\ 1&0&1&0\\ 1&1&1&0\\ 0&0&0&0\\ 1&0&0&0 \end{bmatrix}$$

REFERENCES

1. G.R. Cooper and C.D. McGillem, "Probabilistic methods of signal and system analysis", Holt, Rinehart and Winston Inc., 1971.

2. B.D. Fritchman, "A binary channel characterization using partitioned Markov chains", IEEE Trans. Information Theory, Vol. IT-13, pp. 221-227, April 1967.

3. I.M. Jacobs, "Practical applications of coding", IEEE Trans. Information Theory, Vol. IT-20, pp. 305-310, May 1974.

4. A.D. Wyner and R.B. Ash, "Analysis of recurrent codes", IEEE Trans. Information Theory, Vol. IT-9, pp. 143-156, July 1963.

5. W.E. Rodgers, "Burst-error-correcting convolutional codes with short constraint length", PhD, dissertation, The Ohio State University, Columbus, OH, March 1977.

6. K. Brayer and O. Cardinale, "Evaluation of error correction block encoding for high-speed HF data", IEEE Trans. Commun. Technol., Vol. COM-15, pp. 371-382, June 1967.

7. S. Tsai, "Analytic method of evaluating error correcting codes on a real channel", ICC-72, pp.(15-1) - (15-6).

8. A. Kohlenberg and G.D. Forney, Jr., "Convolutional coding for channels with memory", IEEE Trans. Information Theory, Vol. IT-14, pp. 618-626, Sept. 1968.

9. G.D. Forney, Jr., "Burst-correcting codes for classic bursty channel", IEEE Trans. Commun. Technol., Vol. COM-19, pp. 772-781, Oct. 1971.

10. J.J. Spilker, Jr., "Digital communications by satellite", Prentice-Hall Inc., 1977.

11. J.L. Massey, "Coding for everyday channels", Guest Professor, Laboratory for Commun. Theory, Technical University of Denmark, 2800 Lyngby. On leave from Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, 46556, U.S.A., May 1972.

12. R.W. Lucky, J. Salz and E.J. Weldon, Jr., "Principle of data communication", McGraw Hill, 1968.

13. G.D. Forney, Jr., "Coding system design for advanced solar missions", submitted to NASA Ames Res. ctr. by Codex Corp., Watertown, Mass., Final Rep., contract NAS2-3637, Dec. 1967.

14. A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", IEEE Trans. Information Theory, Vol. IT-13, pp. 260-269, April 1967.

15. J.L. Massey, "Implementation of burst-correcting convolutional codes", IEEE Trans. Information Theory, Vol. IT-11, pp. 416-422, July 1965.

16. R.G. Gallager, "Information theory and reliable communication", John Wiley and Sons Inc., 1968.

17. P. Elias, "Coding for noisy channels", IRE Convention Record, Part 4, pp. 37-47, 1955.

18. D.W. Hagalbarger, "Recurrent codes: Easily mechanized, burst-connecting binary codes", Bell Syst. Tech. J., pp. 969-984, July 1959.

19. W.L. Kilmer, "Some results on best recurrent type binary-error-correcting codes", IRE Int. Convention Record, Part 4, pp. 135-147, 1960.

20. ______________, "Linear recurrent binary-error-correcting codes for memoryless channels", IRE Trans. Information Theory, Vol. IT-7, pp. 7-12, 1961.

21. E.R. Berlekamp, "Note on recurrent codes", IEEE Trans. Information Theory, Vol. IT-10, pp. 257-258, July 1964.

22. F.P. Preparata, "Systematic construction of optimal linear recurrent codes for burst error correction", Calcolo 2, pp. 1-7, 1964.

23. J.L. Massey, "Advances in threshold decoding-Advances in communications systems", Academic Press Inc., 1968.

24. A. Kohlenberg, "Random and burst error control", First IEEE Annual Communications Convention, Boulder, Colo., June 7-9, 1965.

25. R.G. Gallager, "Lower bounds on the tails of probability distributions", MIT Res. Lab. of Electronics, QPR77, pp. 277-291, 1965.

26. Y. Iwadare, "On type B_1 burst-error-correcting convolutional codes", IEEE Trans. Information Theory, Vol. IT-14, pp. 577-583, July 1968.

27. H.T. Hsu, "A note on burst-error-correcting recurrent codes", Int. J. Control, Vol. 13, pp. 465-472, 1971.

28. D.D. Sullivan, "A generalization of Gallager's adaptive error control scheme", IEEE Trans. Information Theory, Vol. IT-17, pp. 727-735, Nov. 1971.

29. M.J. Ferguson, "Diffuse threshold decodable rate $\frac{1}{2}$ convolutional codes", IEEE Trans. Information Theory, Vol. IT-17, pp. 171-180, March 1971.

30. D.M. Mandelbaum, "Some optimal type B_1 convolutional codes", IEEE Trans. Information Theory, Vol. IT-19, pp. 250-251, March 1973.

31. A.M. Rosi, "Information and communication theory", Van Nostrand Reinhold Co. Ltd., second edition, 1973.

32. W.W. Peterson and E.J. Weldon, "Error-correcting codes", MIT Press, second edition, 1972.

33. S. Lin, "An introduction to error-correcting codes", Prentice Hall, 1970.

34. J.L. Massey, "Threshold decoding", MIT Press, Cambridge, Mass., 1963.

35. R. Lyons and L. Beaudet, "Application of forward error correction over aeronautical satellite data links", Communications Research Centre, Dept. of Communications, Ottawa, CRC Report No.1314, April 1978.

36. J.L. Ramsey, "Realization of optimum interleavers", IEEE Trans. Information Theory, Vol. IT-16, pp. 338-345, May 1970.

37. B. Goldberg, E.D. Moyes and J.E. Quigley, "Interleaved Viterbi decoding applied to troposcatter channels", IEEE NTC-73, Vol..2, pp. (21c-1)-(21c-6).

38. H.L. Van Trees, E.V. Hoversten and T.P. McGarty, "Communication satellites: looking to the 1980", IEEE Spectrum, Vol. 14, No. 12, pp. 43-51, Dec. 1977.