Chapter

8 ERROR CONTROL CODING

Introduction

The main aim of any communication schemes is to provide efficient and error-free data transmission system. In order to improve transmission efficiency, source coding techniques are implemented, whereas to reduce the probability of error, error control coding (channel coding) schemes are used.

8.1 The Source Coding

The process ofefficient representation of signal valuesgenerated by discrete source is called source encoding and is accomplished by source encoder. The simplest from of source coding is to assign a code word (combination of binary symbols "1" and "0")of fixed length to the signal values independent of their statistics (i.e. the probability of occurance) . Such kind of source coding is called fixed length coding (FLC) and normally is not very efficient form of source coding. For the source encoder to be efficient, knowledge of statistics of the source is very essential. In normal situations some source symbols are known to be more probable than others and this property may be exployed in the generation of a source code by assigning short code-words to frequent symbols, and long code-words to rare source symbols. We refer to such a source code as a variable-length code (VLC).

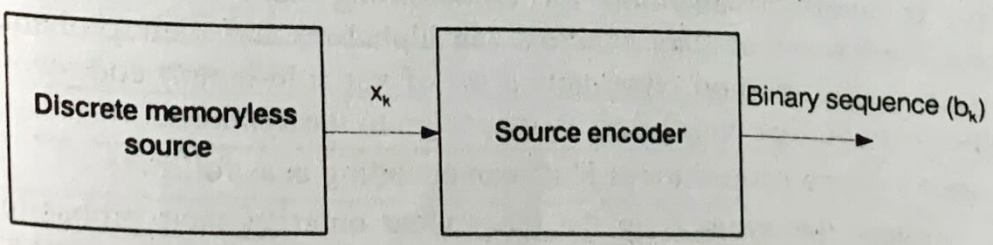

Fig.8.1 Source encoding

In Fig.8.1, the output x_k from the discrete memoryless source is converted into a stream of 0s and 1s, denoted by b_k, by the source encoder. We assume that the source has an alphabet with K different symbols, and that the k^{th} symbol x_k occurs with probability p_k, where, $k = 0, 1, \ldots K - 1$. Let the binary code-word assigned to symbol x_k by the encoder have length l_k, measured in bits (each "1" or "0" is called a *bi*nary digi*t* or bit). We define the average code-word length, L, of the source encoder as

$$\bar{L} = \sum_{k=0}^{K-1} p_k l_k \qquad (8.1)$$

The parameter $\bar{L}$ represents the average number of bits per symbol used in the source encoding process. Let L_{min} denote the theoretically achievable minimum value of $\bar{L}$. We then define the coding efficiency of the source encoder as

$$\eta = \frac{L_{min}}{\bar{L}} \tag{8.2}$$

From the "source coding theorem" for a discrete memoryless source of entropy $H(X)$ the average code-word length $\bar{L}$ for any source encoder is bounded as

$$\bar{L} \geq H(X) \tag{8.3}$$

Accordingly, the entropy $H(X)$ represents a fundamental limit on the average number of bits per source symbol, $\bar{L}$, necessary to represent a discrete memoryless source such that $\bar{L}$ can be made as small as, but no smaller than, the entropy $H(X)$. Thus with $L_{min} = H(X)$, we may rewrite the efficiency of a source encoder in terms of the entropy $H(X)$ as

$$\eta = \frac{H(X)}{\bar{L}} \tag{8.4}$$

8.1.1 Huffman Coding

In computer science and information theory, a Huffman code is an optimal code found using the algorithm proposedby David A. Huffman. Huffman coding is aneffectivemethod for compressing data with variable-length codes.Given a set of data symbols (an alphabet) and their probabilitiesof occurrence, the method constructs a set of variable-length code-wordswith the shortest average length and assigns them to the symbols.

The step-by-step procedure of Huffman encoding is as follows:

1. Arrange the symbols in the descending order of their probability, i.e., symbols with the highest probability at the top and the one with the lowest probability at the bottom. The two source symbols of lowest probability are assigned a 0 and 1.(This part of the step is referred as a splitting stage.)
2. Combine the probability of two symbols having the lowest probabilities to form what is regarded as a new source symbol with probability equal to the sum of the two original probabilities. (The list of source symbols, and therefore source statistics, is thereby reduced in size by one). The list is rearranged after the combination: this step is called reduction 1. As in step 1, assign 0 and 1 for the least highest and lowest probabilities. Repeat this procedure until only two ordered probabilities remain.
3. This procedure is repeated until we are left with a final list of the source statistics (symbols) of only two, for which 0 and a 1 are assigned.

The code for each (original) source symbol is found by working backward and tracing the sequence of 0s and 1s assigned to that symbol as well as its successors.

Example 8.1: ***Apply the Huffman coding procedure for the following message***

$$X = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$$

$$P = [0.4 \ 0.2 \ 0.12 \ 0.08 \ 0.08 \ 0.08 \ 0.04]$$

Solution:

Message	Probability	First Reduction	Second Reduction	Third Reduction	Fourth Reduction	Fifth Reduction
x_1	0.4	0.4	0.4	0.4	0.4	→0.6
x_2	0.2	0.2	0.2	→0.24	→0.36 1	0.4
x_3	0.12	0.12	→0.16	0.2 1	0.24 0	
x_4	0.08	→0.12	0.12 1	0.16 0		
x_5	0.08	0.08 1	0.12 0			
x_6	0.08 1	0.08 0				
x_7	0.04 0					

Code-word for symbol x_4 is.

$$c_4 = 1101$$

Similarly,

Code	Length
$c_1 = 0$	1
$c_2 = 111$	3
$c_3 = 101$	3
$c_4 = 1101$	4
$c_5 = 1100$	4
$c_6 = 1001$	4
$c_7 = 1000$	4

Then the average code word length is:

$$\bar{L} = \sum_{k=1}^{7} p_k l_k = (0.4\times1) + (0.2\times3) + (0.12\times3) + (0.08\times4) + (0.08\times4) + (0.08\times4) + (0.04\times4)$$

$$= 2.48 \text{ letters/message}$$

On the other hand the entropy of the source H(X) or the value of Lmin is defined by the term:

$$H(X) = -\sum_{k=1}^{7} p_k \log p_k$$

Which is now equal

$$H(X) = [(0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4)]$$
$$= 2.42 \text{ bits/message}$$

Therefore the efficiency of the encoder in the above example is:

$$\eta = \frac{H(X)}{\bar{L}} = \frac{2.42}{2.48} = 97.6\%$$

8.1.2 Shannon-Fano Coding

Claude E. Shannon and Robert M. Fanodeveloped highly efficient (but slightly lower than Huffman codes)and easily implementable coding procedure to generate a binary codes.

The step-by-step procedure of Shannon-Fano encoding is as follows:

1. Arrange the symbols in the descending order of their probability. i.e., symbols with the highest probability at the top and the one with the lowest probability at the bottom.
2. Partition the set into two sets that are as close to equiprobable as possible. and assign 0s to the upper set and 1s to the lower set.
3. Continue step 2 in each sub group until further partition is not possible.

Example 8.2: Apply the Shannon-Fano coding procedure for the following message

$$X = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$$
$$P = [0.4 \ 0.2 \ 0.12 \ 0.08 \ 0.08 \ 0.08 \ 0.04]$$

Solution:

These symbols can be coded by partitioning the table in two ways.

A. $X_1 = [x_1 \ x_2]$ $X_2 = [x_3 \ x_4 \ x_5 \ x_6 \ x_7]$

Symbols	Probabilities	Step 1	Step 2	Step 3	Step 4	Code length	Code
x_1	0.4	0	0			2	00
x_2	0.2	0	1			2	01
x_3	0.12	1	0	0		3	100
x_4	0.08	1	0	1		3	101
x_5	0.08	1	1	0		3	110
x_6	0.08	1	1	1	0	4	1110
x_7	0.04	1	1	1	1	4	1111

$$\bar{L} = \sum_{k=1}^{7} p_k l_k = (0.4\times 2) + (0.2\times 2) + (0.12\times 3) + (0.08\times 3) + (0.08\times 3) + (0.08\times 4) + (0.04\times 4)$$
$$= 2.52 \text{ letters/message}$$

And the efficiency is:

$$\eta = \frac{H(X)}{\bar{L}} = \frac{2.42}{2.52} = 96.03\%$$

B. $X_1 = [x_1]$ $X_2 = [x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$

Symbols	Probabilities	Step 1	Step 2	Step 3	Step 4	Code length	Code
x_1	0.4	0				1	00
x_2	0.2	1	0	0		3	100
x_3	0.12	1	0	1		3	101
x_4	0.08	1	1	1	0	4	1100
x_5	0.08	1	1	0	1	4	1101
x_6	0.08	1	1	1	0	4	1110
x_7	0.04	1	1	1	1	4	1111

$$\bar{L} = \sum_{k=1}^{7} p_k l_k = (0.4\times 1) + (0.2\times 3) + (0.12\times 3) + (0.08\times 4) + (0.0843) + (0.08\times 4) + (0.04\times 4)$$
$$= 2.48 \text{ letters/message}$$

Thus, it can be seen that the second method is better as it gives a lower value of $\bar{L}$.

Now,

$$H(X) = -\sum_{k=1}^{7} p_k \log p_k = [(0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4) + (0.4\log 0.4)]$$
$$= 2.42 \text{ bits/message}$$

Hence, efficiency of second method is

$$\eta = \frac{H(X)}{\bar{L}} = \frac{2.42}{2.48} = 97.6\%$$

Thus, it can be seen that the second method is better as it gives a lower value of $\bar{L}$.

8.2 Error Control Coding

The probability of error for a particular signaling scheme is a function of signal-to-noise ratio at the receiver input and the information rate. Due to fixed noise power density $N_o/2$ (Watts/Hz) for a particular environment and limitation on maximum signal power and bandwidth of the channel, it is often not possible to arrive at a signaling scheme which will yield an acceptable probability of error for a given application. The only practical alternative for reducing the probability of error is the use of error-control coding.

There are, in general two methods of error detection and/or correction.

1. Automatic Repeat Request (ARQ),
2. Forward Error Correction (FEC).

In ARQ system, when an error is detected by the receiver, it sends a request to the transmitter for repeat of transmission, after which the transmitter retransmits. The ARQ method needs duplex arrangement as apart from the conventional transmitter to receiver signal, the request signal is to travel from receiver to transmitter. Since time delays are involved in ARQ method due to request and repeat signals, it is not suitable for use in real-time systems.

In FEC, a mechanism of automatic error correction in the form of error-correction code (error-control code) is employed and hence retransmission of data is not necessary. The simplex channel is sufficient for FEC and therefore can be used in real-time systems. Obviously, FEC method is more popular than ARQ method.

In FEC, the channel encoder systematically adds digits to the transmitted message digits. Although these additional digits convey no new information, they make it possible for the channel decoder to detect, and correct errors in the information bearing digits. The overall probability of error is reduced due to error detection and/or correction.

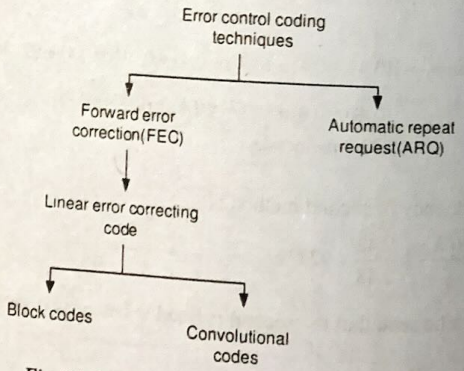

Fig.8.2 Types of error control coding techniques.

The FEC or linear error correcting code is further sub divided into Block codes and Convolutional codes. In block codes, r check bits are derived from the block of k information bits. The structure of block codes is such that the information bits are followed by the check bits or vice versa. These check bits are used to verify the information bits at the receiver side.

In convolutional codes, the check bits are continuously interleaved with message bits. The check bits verify information bits not only in the block immediately preceding them, but in other blocks as well. The main difference between block codes and convolutional codes is that in the block code, a block of n digits generated by the encoder in a particular time unit depends only on the block of k input message digits within that time unit; whereas, in the convolutional code, a block of n code digits generated by the encoder in a time unit depends not only on the block of k message digits within that time unit, but also on the preceding $(N-1)$ blocks of messages $(N>1)$.

8.2.1 Basic Terminology used in Error Control Coding Theory

1. **Code word**

   The code word is the bit encoded block of bits. It contains message bits and may contain parity or redundant bits.

2. **Code rate or code efficiency**

   The code rate or code efficiency (rc) is defined as the ratio of the number of message bits ( ) to the total number of bits ( ) in a code word.

$$r_c = \frac{k}{n} \tag{8.5}$$

3. **Code vectors**

   We can visualize an n-bit code-word to be present in an n-dimensional space. The coordinates or elements of this code vector are the bits present in the code word. Fig.8.3, which shows the 3-bit code vectors. Table 8.1 enlists the 8 (i.e.,2^3) possible combinations of the 3-bit code word. We can assume bits x_0 to be on X-axis, bits x_1 to be on Y-axis and bits x_2 to be a Z-axis.

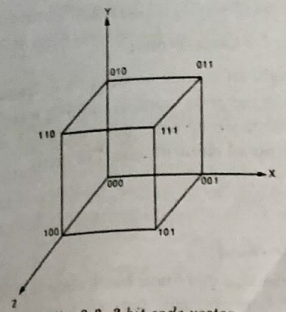

Fig.8.3. 3 bit code vector

Table 8.1 Combinations of 3 bit code word.

S No	Code word		
	$x_2 = Z$	$x_1 = Y$	$x_0 = 0$
0	0	0	0
1	0	0	1
2	0	1	0
:	:	:	:
:	:	:	:
7	1	1	1

4. **Hamming distance**

The Hamming distance is the number by which a code word differ from other code word of the same code vector.For example, let us consider two code vectors (or code words) having the same number of elements.

Code word 1: 11010100

Code word 2: 01011110

These two code words differ in the bit positions in bold. The count of such positions is three. Therefore the Hamming distance (HD) is 3. Alternatively, the HD can be found out by bit by bit X-ORing the two code words and counting the number of '1's in the result.

$$\begin{array}{r} 11010100 \\ \oplus 01011110 \\ \hline 10001010 \end{array}$$

Where, $\oplus$ is the X-OR operation. Thus,

Hamming distance =1+0+0+0+1+0+1+0=3

5. **Hamming weight of a Code Word:**

The Hamming weight of a code word x is defined as the number of non-zero elements in the code word. Hamming weight of a code vector (code word) is the distance between the code word and an all zero code vector (A code having all elements equal to zero).

For Code word: 11010100

Hamming weight is 4.

6. **Minimum Distance (d_{min})**

The minimum distance d_{min} of a linear block code is defined as the smallest Hamming distance between any pair of code vectors in the code.

Therefore, the minimum distance is same as the smallest Hamming weight of difference between any pair of code vectors. It can be proved that the minimum distance of a linear block code is the smallest Hamming weight of the non-zero code vectors in the code.

7. **Minimum distance and the error detection and correction capabilities**

The error detection is always possible when the number of transmission errors in a code word is less than the minimum distance d_{min} because then the erroneous word will not be a valid code word. But when the number of errors equals or exceeds d_{min}, the erroneous code word will correspond to another valid code word and errors cannot be detected. The error detection and correction capabilities of a coding technique (or the code vector) depend on the minimum distance d_{min} as given below.

a. For detecting upto s errors per word, $d_{min} \geq (s + 1)$ must be satisfied.

b. For correcting upto t errors per word, $d_{min} \geq (2t + 1)$ must be satisfied.

c. For correcting upto t errors and detecting $s > t$ errors per word, $d_{min} \geq (t + s + 1)$ must be satisfied.

Example 8.3

Consider a 3-bit code word. The combinations of codes with HD=2 are 000,101,110 and 011

If these code words are used then any single error in any of the code words can easily be detected. i.e., if for 101 the first bit has error i.e., it is received as 001, then as this combination now is not a valid code, it is then decided that there is an error (error detection), but you cannot correct it as error in 2nd bit of code 01^{c}1=001 will also yield the same received sequence as error in 1st bit of 101 i.e., 001.

Now if the code word with HD=3 (000 & 111) are used then we can detect and correct single error in each code word.

8.2.2 Block Code

In block codes, a block of k message bits is encoded into a block of n bits ($n > k$), by adding 'n-k' check bits to the block of message bits (Fig. 8.4). The check bits are derived from the message bits. The n-bit block of a channel encoder output is called a code word and the codes in which the message bit block appear at the beginning or the end of a code word, are called systematic codes.

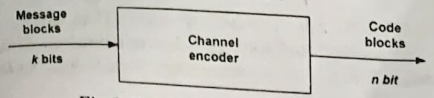

Fig.8.4 Block diagram of block code

The simplest possible block code is a parity check codes. In this coding scheme, when the total number of 1's in the code word is even, it is an even parity check code, and when the check bit is such that the total number of 1's in the code word is odd, it is an odd parity check code.

The following example explains the parity check code

Message	Code for even parity		Code for odd parity	
	Message	Check bit	Message	Check bit
010011	010011	1	010011	0
101110	101110	0	101110	1

If a single error occurs in a received message, it can be immediately detected as it changes the number of 1's from even to odd or from odd to even, although the position of the erroneous bit cannot be determined. Thus, with this code, through a single error can be detected, but it cannot be corrected.

1. Linear Block Code

In a (n,k) linear block codes k is the number of message (data) bits, n is the number of codeword bits and n-k is the number of check (parity) bits. These check bits are computed from the message bits, according to the predetermined encoding rule.

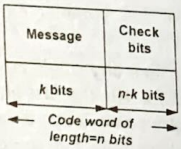

Fig.8.5 Structure of the codeword for a linear block code

If each of the 2^k code-words of a systematic code can be expressed as linear combinations of 2^k linearly independent code-vectors, the code is called linear systematic block code.

Generation of Linear block code

The n-k additional check bits are generated from linear combinations of the 'k' message bits. The encoding operations can be described with the help of matrices.

Let a message block be a row vector

$$D = [d_1 d_2 ... d_k] \tag{8.6}$$

Where each message bit can be a 0 or a 1. Thus, we have 2^k distinct message blocks (i.e., the total possible combination of k-bit block). Each message block is transformed into a code word C of length n bits.

$$C = [C_1 C_2 ... C_n] \tag{8.7}$$

After encoding, there will also be 2^k distinct code words. It may be noted that there is one unique code word for each distinct message block. This set of 2^k code words, also known as code vectors, is called an (n, k) block code. The rate efficiency of this is $\frac{k}{n}$.

Systematic Linear Block Code

In a systematic linear block code, the first k bits of the code bits are the message bits, i.e.,

$$c_i = d_i, i = 1,2,....k \tag{8.8}$$

The last $n - k$ bits $(i = k + 1,...,n)$ in the code word are check bits generated from k message bit according to some predetermined rule:

$$\left.\begin{aligned} C_{k+1} &= p_{11}d_1 \oplus p_{21}d_2 \oplus .... \oplus p_{k,1}d_k \\ C_{k+2} &= p_{12}d_1 \oplus p_{22}d_2 \oplus .... \oplus p_{k,2}d_k \\ &\vdots \\ C_n &= p_{1,n-k}d_1 \oplus p_{2,n-k}d_2 \oplus .... \oplus p_{k,n-k}d_k \end{aligned}\right\} \tag{8.9}$$

The coefficients $p_{i,j}$ in the above equation are 0's and 1's so that C_k's are always 0's and 1's.The addition are modulo-2 additions. Eq.(8.8) and Eq.(8.9) can be combined to give a matrix Eq.(8.10):

$$[C_1 C_2 .....C_n] = [d_1 d_2 .....d_k] \begin{bmatrix} 1000 & ... & 0 & \vdots & p_{11} & p_{12} & \cdots & p_{1,n-k} \\ 0100 & ... & 0 & \vdots & p_{21} & p_{22} & \cdots & p_{2,n-k} \\ 0010 & ... & 0 & \vdots & p_{31} & p_{32} & \cdots & p_{3,n-k} \\ ... & ... & ... & ... & ... & ... & ... & ... \\ 0000 & ... & 1 & \vdots & p_{k,1} & p_{k,2} & \cdots & p_{k,n-k} \end{bmatrix}_{k \times n} \tag{8.10}$$

or

$$C = DG \tag{8.11}$$

Where G is the $k \times n$ matrix on the RHS of Eq.(8.11) is called the generator matrix of the code and is used in encoding operation. It has the form

$$G = [I_k : P]_{k \times n} \tag{8.12}$$

Where I_k is the identity matrix of the order k and P is an arbitrary coefficient $k \times (n - k)$ matrix. Matrix P completely defines the (n, k) block code. Proper design of a P matrix is an important step in the design of an (n, k) block code as the code generated by G achieves certain desirable properties such as the ease of implementation, ability to correct, high rate efficiency, etc. based on the P matrix.

Parity check matrix

A parity check matrix H, used to verify the validity of code word generated by the generator matrix G, is associated with each (n, k) block code

$$H = \left[P^T : I_{n-k}\right]_{n-k \text{ by } n} \tag{8.13}$$

$$H = \left[\begin{array}{cccc:cccccc} p_{11} & p_{21} & \cdots & p_{k,1} & 1 & 0 & 0 & \cdots & 0 \\ p_{12} & p_{22} & \cdots & p_{k,2} & 0 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{1,n-k} & p_{2,n-k} & \cdots & p_{k,n-k} & 0 & 0 & 0 & \cdots & 1 \end{array}\right] \tag{8.14}$$

Where P^T is the transpose of matrix P. The parity check matrix will be used to verify whether a code word C is generated by the matrix G. The rule for verification is:

C is a code word in the (n, k) block code generated by G if and only if,

$$CH^T = [0 \;\; 0 \;\; \cdots \;\; 0] \tag{8.15}$$

Where H^T is the transpose of H and is also given by

$$H^T = \left[\frac{P}{I_{n-k}}\right] \tag{8.16}$$

The parity check matrix H is used in the decoding (error detection and /or correction) operation as follows:

Syndrome Testing

Consider a linear (n, k) block code with a generator matrix $G = [I_k : p]$ and a parity check matrix $H = [P^T : I_{n-k}]$. Let C and R be the transmitted and received code vectors, respectively, in a noisy communication channel. The received vector R, corrupted by the noise in the channel, is the sum of the transmitted code vector C, and an error vector E i.e.,

$$R = C \oplus E \tag{8.17}$$

The function of the channel decoder is to retrieve C from R with minimum error, and then the source decoder maps the message block D from C. The channel decoder does the decoding operation by determining an $(n - k)$ syndrome vector S.

$$S = RH^T \tag{8.18}$$

Eq.(8.18) can be rewritten as

$$S = (C \oplus E)H^T \tag{8.19}$$

$$S = CH^T \oplus EH^T \tag{8.20}$$

$$S = EH^T \tag{8.21}$$

since $CH^T = [0 \;\; 0 ....0]$

Thus, the syndrome is all zero if R is a valid code vector (i.e. when $R = C$ and $E = 0$). If error occurs in transmission, the syndrome S of the received vector is non-zero. Moreover, the sequence of S can be used to locate the bit error position in the received block and correct it.

Selection of H^T

First k rows of H^T (i.e., the matrix P) should he chosen to be distinct. Moreover, none of the them can be all 0's, since it corresponds to no-error condition. Also, none of them can be any row of I_{n-k} (i.e., with a single 1 in it), since the last $n - k$ row s of H^T constitute the identity matrix I_{n-k} (each of which have a single 1 in it). Thus, the matrix P should be so chosen that all the rows are distinct and consist at least two 1's in them.

Example 8.4 Suppose a linear systematic block code uses the following generator matrix G for encoding:

$$G = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

Then the parity check matrix H for it is:

$$H = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

Now the code word for a block of (1011) message bits generated by G is:

$$C = DG = 1011001$$

And if the code word is received without any error, the syndrome S will be:

$$S = C H^T = 000$$

Now suppose the third bit of C has been received with error, i.e. the received code word 'R' now is '1001001', and the error code vector E is '0010000' such that:

$$R = C \oplus E$$

With the error in the received code word, the syndrome 'S' will be:

$$S = EH^T = 101$$

This combination of the syndrome S matches with the third row of HT, indicating the bit error position in the received code vector R.

2. Cyclic code

Cyclic codes are a subclass of linear block codes. The cyclic codes offer the following benefits.

i. Simple shift registers can be used for encoding and syndrome calculations.

ii. The mathematical structure of these codes allow the design of codes having useful error-correcting properties, thus suitable for situation which requires constructing of higher order error correcting codes.

A binary code is said to be cyclic code if it exhibits two fundamental properties:

1. Linearity property: The sum of any two code words in the code is also a code word(All cycle codes are essentially linear block codes) of the same code vector.
2. Cyclic property: any cyclic shift of a code word in the code is also a codeword of the same code vector.

In cyclic codes, the code words of a code vector are simply the cyclic shifts of one another. In other words, cyclic shift of a code word will result in another code word of the same code vector.

For example, if $c = (c_0, c_1, c_2, \ldots, c_{n-1})$ is a codeword, then $(c_1, c_2, c_3, \ldots, c_{n-1}, c_0)$, $(c_2, c_3, \ldots, c_{n-1}, c_0, c_1)$ and so on are also code words.

Cyclic codes can easily described in the polynomial form. This property is extremely useful in the analysis and implementation of these codes. The code vector c in Eq. (8.24) can be expressed as the $(n-1)$ degree polynomial

$$c(x) = c_0 + c_1 x + c_2 x^2 + \ldots\ldots\ldots + c_{n-1} x^{n-1} \qquad (8.24)$$

The coefficient of the polynomial is either 0 or 1, and they obey the following rules:

0+0=0	0×0=0
0+1=1+0=1	0×1=1×0=0
1+1=0	1×1=1

The code polynomial $c^{(i)}(x)$ for the cyclic shifted code vector $c^{(i)}$ in Eq.(8.23) is

$$c^{(i)}(x) = c_i x^i + c_{i+1} x^{i+1} + \ldots + c_{n-1} x^{n-1} + c_0 + \ldots + c_{i-1} x^{i-1}) \qquad (8.25)$$

(i) Systematic cyclic code

A (n, k) code word can be represented by a code polynomial

$$c(x) = c_0 + c_1 x + c_2 x^2 + \ldots\ldots\ldots + c_{n-1} x^{n-1} \qquad (8.26)$$

The generator polynomial $g(x)$ (of degree $n - k$) and data polynomial $d(x)$ (of degree $k - 1$) of a code can be represented as

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots\ldots\ldots + g_{n-k-1} x^{n-k-1} + x^{n-k} \qquad (8.27)$$

$$d(x) = d_0 + d_1 x^1 + \ldots\ldots\ldots + d_{k-1} \qquad (8.28)$$

The code polynomial can be thus generated multiplying generator and data polinimial via Eq.(8.29)

$$c(x) = d(x) g(x) \qquad (8.29)$$

Note that there are 2^k distinct polynomials (or code words). For cyclic code, a code word after cyclic shift is still a code word. Thus generated code words are non-systematic in form.

(ii) Systematic Cyclic Code

The codes generated by simple cyclic generation matrix are not systematic. In a systematic code, the first k digits are the data bits, and the last $m = n - k$ digits are the parity check bits. Systematic codes are a special case of general codes.

In systematic code generation, the relation between the code polynomial $c(x)$ and data polynomial $d(x)$ is given by

$$c(x) = x^{n-k} d(x) + \rho(x) \qquad (8.30)$$

Where, $\rho(x)$ is the remainder from dividing $x^{n-k} d(x)$ by $g(x)$, i.e.,

$$\rho(x) = \text{Rem} \frac{x^{n-k} d(x)}{g(x)} \qquad (8.31)$$

Syndrome Decoding for cyclic code

Every valid code polynomial $c(x)$ is a multiple of $g(x)$. In other words, $c(x)$ is divisible by $g(x)$. When an error occurs during the transmission, the received word polynomial $r(x)$ will not be a multiple of $g(x)$. Thus,

$$\frac{r(x)}{g(x)} = q(x) + \frac{s(x)}{g(x)} \tag{8.32}$$

$$r(x) = q(x) \cdot g(x) + s(x) \tag{8.33}$$

Where $s(x)$ is the syndrome polynomial of degree $n - k - 1$ or less. The $q(x)$ represents the quotient polynomial.

If $e(x)$ is the error polynomial, then

$$r(x) = c(x) + e(x) \tag{8.34}$$

Remembering that $c(x)$ is a multiple of $g(x)$,

$$s(x) = \text{Rem}\frac{r(x)}{g(x)} \tag{8.35}$$

$$s(x) = \text{Rem}\frac{c(x) + e(x)}{g(x)} \tag{8.36}$$

$$s(x) = \text{Rem}\frac{e(x)}{g(x)} \tag{8.37}$$

Thus, the syndrome polynomial s(x) is all zero if no errors are present in the received code. Non-zero values of s (x) will indicated the error position in the received code.

Example 8.5: ***Let the generator polynomial for a (7,4) cyclic code is given by,***

$g(x)=1+x+x^3$

Determine all the systematic and non-systematic code vectors for message code word.

D=(d_1 d_2 d_3 d_4)=(1 0 1 0)

Solution:

Non-systematic code vector

The message polynomial is given by,

$$d(x) = d_0 + d_1 x^1 + \ldots\ldots\ldots + d_{k-1}$$

Substituting the value of (d_1 d_2 d_3 d_4) we get

$d(x)=1+x^2$

The non-systematic cyclic code is given by

$$c(x) = d(x)g(x)$$

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

$$c(x) = 1 + x + x^2 + (1 \oplus 1)x^3 + x^5$$

We know that, $(1 \oplus 1)=0$

Therefore,

$$c(x) = 1 + x + x^2 + x^5 = 1 + 1x + 1x^2 + 0x^3 + 0x^4 + 1x^5 + 0x^6$$

Note that the degree of the code word polynomial is 6 i.e., (n-1). The code word polynomial is given by,

C=(1 1 1 0 0 1 0) and this is not in the systematic form.

Systematic code vector

To obtain the systematic code vector we know that

$$c(x) = x^{n-k}d(x) + \rho(x)$$

Now, computing the above Eq. in parts

$$x^{n-k}d(x) = x^3(1 + x^2) = x^3 + x^5 \tag{1}$$

and,

$$\rho(x) = \text{Rem}\frac{x^{n-k}d(x)}{g(x)}$$

$$\text{Rem}\frac{x^3 = x^5}{1 + x + x^3} = x^2 \tag{2}$$

Combining 1 and 2 we get

$$c(x) = (x^3 + x^5) \oplus x^2 = x^2 + x^3 + x^5$$

Therefore, the code vector is

$$c(x) = 0 + 0x^1 + 1x^2 + 1x^3 + 0x^4 + 1x^5 + 0x^6$$

$$C = (0\ 0\ 1 : 1\ 0\ 1\ 0)$$

This is the code vector obtained in systematic form. The first 3 bits are check bits and the later 4 bits are the message bits.

Error Detection using syndrome polynomial

The transmitted code vector is C = (0 0 1: 1 0 1 0) thus, the corresponding code polynomial is represented as

$$c(x) = x^2 + x^3 + x^5$$

for a generator polynomial

$$g(x) = 1 + x + x^3$$

Suppose the received code is as R = (0 0 1: 0 0 1 0) thus the corresponding code polynomial at the receiver is

$$r(x) = x^2 + x^5$$

Now computing the syndrome, we have

$$s(x) = \text{Rem}\frac{r(x)}{g(x)}$$

$$\text{Rem}\frac{r(x)}{g(x)} = \text{Rem}\frac{x^2 + x^5}{1 + x + x^3} = x^3$$

Thus the syndrome s(x) is not zero and is equal to x^3, stating that there is an error in the 4th bit of the received code.

Note: *Polynomial division*

$$1 + x + x^3 \;\overline{)\; 1x^2 + 0x^3 + 1x^5}$$

$$\underline{1x^2 \oplus + 1x^3 \oplus + 1x^5 \oplus}$$

$$0x^2 + 1x^3 + 0x^5$$

$\oplus$ is an *X-OR* operation (modulo-2)

8.2.3 Convolutional Code

Unlike block codes, where n code digits generated by the encoder in any particular time unit depends only on the block of k input data digits within that time unit, in convolutional code the block of n code digits generated by the encoder in a particular time unit depends not only on the block of k message digits within that time unit but also on the data digits within a previous span of time units. For convolutional codes, k and n are usually small. Convolutional codes can be used for correcting random errors, burst errors, or both. Encoding and decoding are easily implemented by using shift registers.

Terminology used in convolutional code

1. **Constraint length (N)**

The constraint length of a convolutional code, expressed in terms of coded symbols n, is defined as the number of output code symbols that are influenced by a single message bit. In an encoder with M-stage shift register, the memory of the encoder equals M message bits, and M + 1 shifts are required for a message bit to enter the shift register and finally come out. Hence, the constraint length of the encoder is $N = (M + 1)$.

2. **Code Dimension**

The code dimension of a convolutional code depends on n, k, M. Here k represents the number of message bits taken at a time by the encoder, n is the number of encoded bits per message bit and M is the encoder's memory i.e., number of shift registers. The code dimension is therefore represented by (n, k, M). The operation of a convolutional encoder is bea illustrated through the example.

Fig.8.6 shows a convolutional encoder with k = 1, n = 2 and N = 3 . Hence the code rate of this encoder is n/k i.e.,1/2, the length of bits per message bit is 2 and the constraint length is 3 (i.e. the number of shift registers is 2).

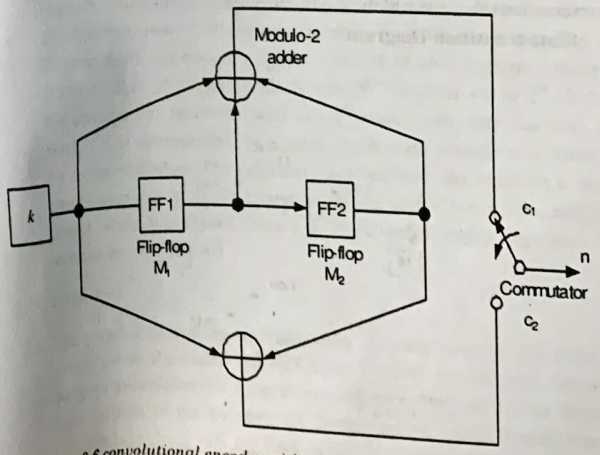

Fig. 8.6 convolutional encoder with constraint length N=3, rate=1/2

Encoder

The operation of convolutional encoder is explained with the help of Fig 8.6. In the figure the encoder consists of two flip-flops and two modulo-2 adder. The encoder takes a single bit input which is converted to two bit code word. The output of each of the modulo-2 adders are defined by the following equations

$$c_1 = k \oplus FF1 \oplus FF2 \tag{8.38}$$

$$c_2 = k \oplus FF2 \tag{8.39}$$

Operation

i. Initially suppose that all shift registers are clear.

ii. First message bit enters $FF1$ and during the bit period of this message bit the commutator samples all two outputs.

iii. Therefore a single input bit is converted into 2-bit code word.

iv. When the next message bit enter the $FF1$, its contents are shifted to $FF2$ and during the bit duration (T_b) of second message bit, the commutator samples c_1 and c_2 to generate another code block.

v. In this way the status of the first bit influence the two blocks of code words containing of a total of 6 bits (including the first set of code word generated immediately i.e., when at block k). Therefore the influence level (constraint length) is equal to $N = (M + 1)$.

Graphical representation for convolutional encoder

For the convolutional encoding there are three different graphical representations that are widely used which are.

1. **State transition Diagram**

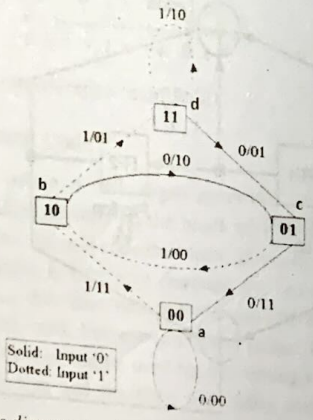

Fig.8.7 (a) State diagram for the (2,1,2) convolutional encoder of Fig.8.6.

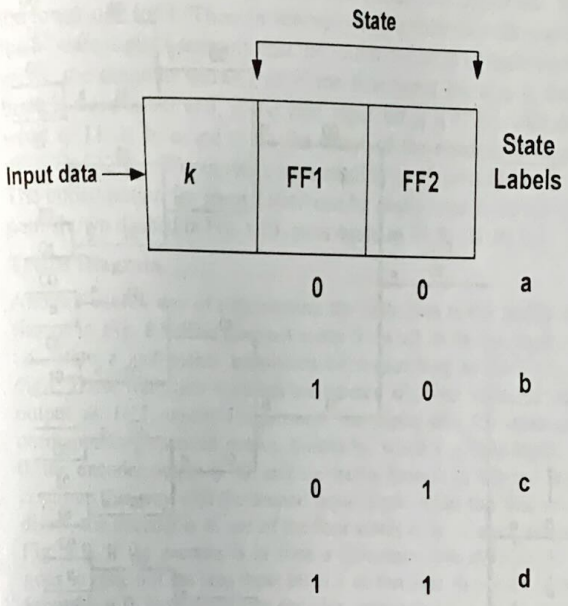

Fig.8.7 (b): State labels for shift register states

Fig.8.7(a) shows the state diagram that corresponds to the shift-register implementation of Fig.8.7(b). For convolutional codes, it is more convenient to illustrate encoders by their corresponding state diagrams. As the encoder consists out of M shift registers, there is a finite number of 2^M states. Each state $S^{(m)}$ with m = 0 to $2^M - 1$, hence the encoder can be considered as a 2^M-state finite state machine. Each state $S^{(m)}$ is represented by a node, each state transition is illustrated by a directed edge. The edges are labeled with the input bit k and the output symbols n, in the example shown above $k/c_1\ c_2$. Dashed edges depict state transitions where $k = 0$, solid edges represent state transitions where $k = 1$.

2. **Code tree**

The process of coding and decoding is made easy by the code tree, which shows the coded output for any possible sequence of data digits. At each successive bit time the encoding procedure can be described by transition in the tree branch describing an output branch word (c_1 c_2). The branching rule for finding a code word sequence is as follows.

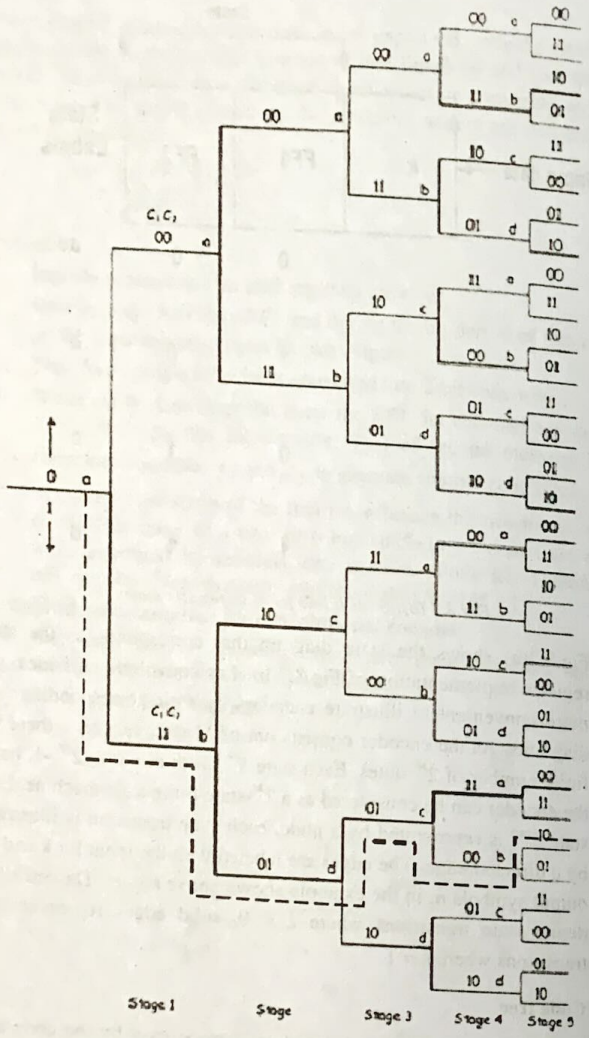

Fig.8.8 Code tree for (2,1,2) encoder

If the input is 0, its associated branch word is found by moving to the next right most branch in upward direction. If the input bit is 1, its branch word is found by moving to the next right most branch in the downward direction. At the terminal node of each of the two branches.

we follow a similar procedure, corresponding to the second data digit. Hence, two branches initiate from each node, the upper one for 0 and the lower one for 1. This convention will be followed throughout until the k^{th} data digit. Assuming that the initial contents of the encoder is all zeros, the diagram shows that if the first input bit is a 0, the output branch word is 00 and, if the first input bit is a 1, the output branch word is 11. a, b, c and d are the states of the encoder. Hence, in all there are 32 (or 2^k) outputs corresponding to 2^k possible data vectors. The coded output for input 11010 can be easily read from this tree (the path shown dashed in Fig. 8.8). an is equal to 11 01 01 00 10.

3. Trellis Diagram

Another useful way of representing the code tree is the trellis diagram shown in Fig. 8.9. The diagram starts from all 0s in the shift register, i.e., state a and makes transitions corresponding to each input data digit. These transition branches are labeled with the value of input and output as 1/11, where 1 represents the input and 11 represents the corresponding encoded output. Similarly, when the first input digit is 0, the encoder output is 00, and the trellis branch is labeled 0/00. We continue this way with the second input digit. After the first two input digits, the encoder is in one of the four states a, b, c, or d, as shown in Fig. 8.9. If the encoder is in state a (previous two data digits 00), it goes to state b if the next input bit is 1 or remains in state a if the next input bit is 0. In so doing, the encoder output is 11 (a to b) or 00 (a to a).

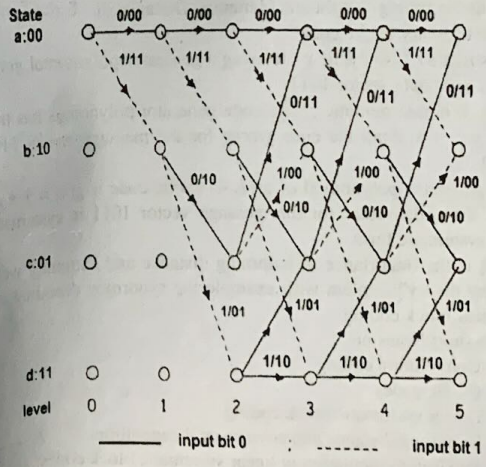

Fig.8.9 Trellis diagram for (2,1,2) encoder

Decoding Algorithms

Maximum-likelihood decoding (Viterbi's algorithm)

Among various decoding methods for convolutional codes, Viterbi's maximum likelihood algorithm is one of the best techniques for digital communications when reduction in computational complexity is the main importance. The decoder structure is relatively simple for short constraint length N, making decoding feasible at relatively high rates of up to 10 Gbit/s.

Consider the trellis diagram of Fig.8.9. We observe that at level 3, there are two paths entering any of the four nodes in the trellis. The algorithm operates by computing a metric for every possible path in the trellis. The metric for a particular path is defined as the Hamming distance between the coded sequence represented by that paths and the received sequence. Thus, for each node (state) in the trellis, the algorithm compares the two paths entering the node. The path with the lower metric (i.e., minimum hamming distance) is retained which is the survivor or active paths, and the other path is discarded. A difficulty that may arise in the application of the Viterbi algorithm is the possibility that two matrices are identical for paths entering a state. In such a situation any one of the two paths are chosen.

The Viterbi algorithm is a maximum-likelihood decoder, which is optimum for an AWGN channel is described as follows.

Previous Exam Questions

1. Define Hamming weight and Hamming Distance.
2. What is binary cyclic code?
3. Construct a (7, 4) cyclic code using a generator polynomial $g(x) = x^3 + x^2 + 1$ with data vector 1011.
4. A (7, 4) non-systematic cyclic code generator polynomial has the form $g(x) = 1 + x$. Find the code words for the message blocks 1101 and 0010.
5. The generator polynomial of a (7, 4) cyclic code is $g(x) = 1 + x + x^3$. Find the code vector for the message vector 1011 in systematic and non-systematic form.
6. What is the importance of hamming distance and hamming weight in coding theory? Explain with example the syndrome decoding method in linear block coding.
7. Write short Notes on:
   a. Convolution coding
   b. Cyclic codes
   c. Linear systematic block coding
   d. Hamming distance and error control capabilities
   e. Syndrome calculation in linear systematic block codes

✡✡✡

- Dinesh Kumar Sharma, *Manual on Co*
- A.V. Oppenheim and R.W.Schafe (Prentice-Hall, 1975).
- A.Papoulis, Probability, *Random Va* (McGraw-Hall, 1984)
- S.Haykin, *Digital Communications*, (W
- S.Haykin, *An Introduction to Analog* (Wiley, 1989)
- B.P.Lathi and Zhi Ding, *Modern Digi Systems*, (Oxford, 2010)
- R.B. Ash, *Information Theory* (Wiley,
- N.Abramson, *Information Theory and*
- T.C. Bartee, *Data Communications*, N W. Sams, 1985).
- Bell Telephone Laboratories, *Communications* (1970).
- R.W.Hamming, *Coding and Informatio*
- R.W.Hamming,"*Error Detecting and* Sustem Tech. J., 1950