UNIT 5

ERROR CONTROL CODING

Channel coding theorem - Linear Block codes - Hamming codes - Cyclic codes - Convolutional codes – Vitterbi Decoder

5.1 INTRODUCTION

Errors are introduced in the data when it passes through the channel. The channel noise interferes the signal. The signal power is also reduced. In this chapter we will study various types of error detection and correction techniques.

5.1.1 Basics for Coding

• The transmission of the data over the channel depends upon two parameters. They are transmitted power and channel bandwidth. The power spectral density of channel noise and these two parameters determine signal to noise power ratio.

• The signal to noise power ratio determine the probability of error of the modulation scheme. For the given signal to noise ratio, the error probability can be reduced further by using coding techniques. The coding techniques also reduce signal to noise power ratio for fixed probability of error.

Fig. 5.1 shows the block diagram of the digital communication system which uses channel coding.

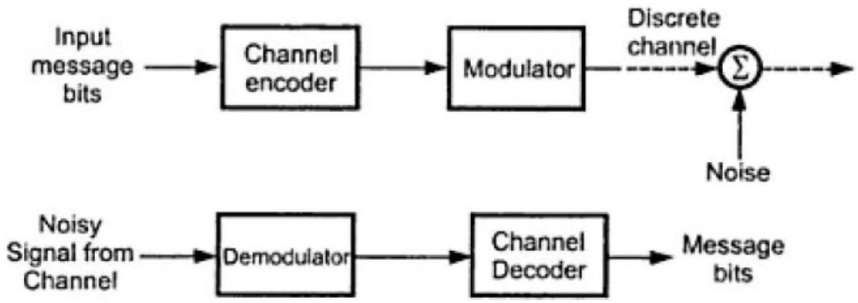


Fig. 5.1 Digital communication system with channel encoding

1. Channel encoder

The channel encoder adds extra bits (redundancy) to the message bits. The encoded signal is then transmitted over the noisy channel.

2. Channel decoder

• The channel decoder identifies the redundant bits and uses them to detect and correct the errors in the message bits if any.

• Thus the number of errors introduced due to channel noise is minimized by encoder and decoder. Due to the redundant bits, the overall data rate increases. Hence channel has to accommodate this increased data rate. The systems become slightly complex because of coding techniques.

5.1.2 Types of Codes

The codes are mainly classified as block codes and convolutional codes.

i) Block codes : These codes consists of 'n' number of bits in one block or codeword. This codeword consists of 'k' message bits and (n - k) redundant bits. Such block codes are called (n, k) block codes.

ii) Convolutional codes : The coding operation is discrete time convolution of input sequence with the impulse response of the encoder. The convolutional encoder accepts the message bits continuously and generates the encoded sequence continuously.

The codes can also be classified ns linear or nonlinear codes.

i) linear code: If the two codewords of the linear code are added by modulo-2 arithmetic, then it produces third codeword in the code. This is very important properly of the codes, since other codewords can be obtained by addition of existing codewords.

ii) Nonlinear code : Addition of the nonlinear codewords does not necessarily produce third codeword.

5.1.3 Discrete Memoryless Channels

• **Definition:** Consider the digital communication system of Fig. 5.1. Let the output of channel decoder depends only on the present transmitted signal, and it does not depend on any previous transmission. Then the modulator, discrete channel and demodulator of Fig. 5.1 can be combinely modeled as a discrete memoryless channel.

• We know that such channel is completely described by transition probabilities $P(y_j/x_i)$. Here . x_i is the input symbol to modulator from channel encoder. And y_j is the output of demodulator and input to the channel decoder. $P(y_j/x_i)$ represents the probability of receiving symbol y_j given that symbol x_i was transmitted.

5.1.4 Examples of Error Cont.rol Coding

• **Principle of redundancy**: Let us consider the error control coding scheme which transmits 000 to transmit symbol '0' and 111 to transmit symbol '1'. Here note that there are two redundant bits in every message (symbol) being transmitted. The decoder checks the received triplets and takes the decision in favour of majority of the bits. For example, if the triplet is 110, then there are two 1's. Hence decision is taken in favour of 1. Here note that there is certainly error introduced in the last bit. Similarly if the received triplet is 001 or 100 or 010, then the

decision is taken in favour of symbol '0'. The message symbol is received correctly if no more than one bit in each triplet is in error. If the message would have been transmitted without coding, then it is difficult to recover the original transmitted symbols. Thus the redundancy in the transmitted message reduces probability of error at the receiver.

• Important aspects of error control coding:

I. The redundancy bits in the message are called check bits. Errors can be detected and corrected with the help of these bits.

ii. It is not possible to detect and correct all the error in the message. Errors upto certain limit can only be detected and corrected.

iii. The check bits reduce the data rate through the channel.

5.1.5 Methods of Controlling Errors

There are two main methods used for error control coding: Forward acting error correction and Error detection with transmission.

i) Forward acting error correction

In this method, the errors lire detected and corrected by proper coding techniques at the receiver (decoder).The check bits or redundant bits are used by the receiver to detect and correct errors. The error detection and correction capability of the receiver depends upon number of redundant bits in the transmitted message. The forward acting error correction is faster. but over all probability of errors is higher. This is becausesome of the errors cannot be corrected.

ii) Error detection with retransmission

In this method, the decoder checks the input sequence. When it detects any error, it discards that part of the sequence and requests the transmitter for retransmission. The transmitter then again transmits the part of the sequence in which error was detected. Here note that, the decoder does not correct the errors. It just detects the errors and sends requests to transmitter. This method has lower probability of error, but it is slow.

5.1.6 Types of Errors

There are mainly two types of errors introduced during transmission on the data: random errors and burst errors.

i) Random errors: These errors are created due to white gaussian noise in the channel. The errors generated due to white gaussian noise in the particular interval does not affect the performance of the system in subsequent intervals. In other words, these errors are totally uncorrelated. Hence they are also called as random errors.

ii) Burst errors: These errors are generated due to impulsive noise in the channel. These impulse noises (bursts) are generated due to lightning and switching transients. These noise bursts affect several successive symbols. Such errors are called burst errors. The burst errors are dependent on each other in successive message intervals.

5.1.7 Important Terms Used In Error Control Coding

The terms which are regularly used in error control coding are.

Codeword: The encoded block of 'n' bits is called a codeword. It contains message bits and redundant bits.

Block length: The number of bits 'n' after coding is called the block length of the code.

Code rate: The ratio of message bits (k) and the encoder output bits (n) is called code rate. Code rate is defined by *'r'* i.e,

$$r = \frac{k}{n}$$

we find that $0 < r < 1$.

Channel data rate: It is the bit rate at the output of encoder. If the bit rate at the input of encoder is R_s then channel data rate will be,

$$channel\ data\ rate\ (R_0) = \frac{n}{k} R_s$$

Code vectors: An 'n' bit codeword can be visualized in an n-dimensional space as a vector whose elements or coordinates are the bits in the codeword. It is simpler to visualize the 3-bit codewords. Fig. 5.2 shows the 3-bit codevectors. There will be distinct '8' codewords (since number of codewords = 2^k). If we let bits b_o on x-axis, b_1 on y-axis and b_2 on z-axis, then the following table gives various points as code vectors in the 3-dimensional space.

Sr.No.	Bits of code vector		
	$b_2 = z$	$b_1 = y$	$b_0 = x$
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Table 5.1 Code vectors In 3-dlmenslonal space

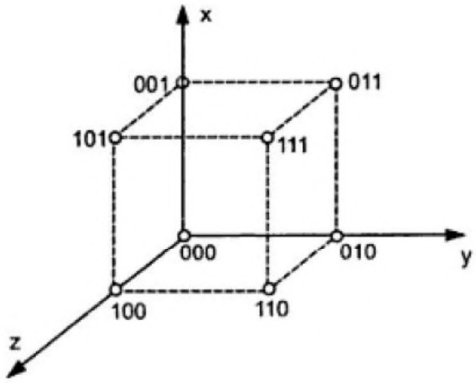


Fig. 5.2 Code vectors representing 30bltcodewords

Hamming distance: The hamming distance between the two code vectors is equal to the number of elements in which they differ. For example let X=(101)and Y=(110). The two code vectors differ in second and third bits. Therefore hamming distance between X and Y is 'two'. Hamming distance is denoted as *d (X,* Y) or simply' *d'.* i.e,

$$d(X,Y) = d = 2$$

Thus we observe from fig. 5.2 that the hamming distance between (100) and (011) is maximum i.e. 3. This is indicated by the vector diagram also.

Minimum distance (d_{min}): It is the smallest hamming distance between the valid code vectors.

Error detection is possible if the received vector is not equal to some other code vector. This shows that the transmission errors in the received code vector should be less than minimum distance d_{min}. The following table lists some of the requirements of error control capability of the code.

Sr.No.	Name of errors detected / corrected	Distance requirement
1	Detect upto 's' errors per word	$d_{min} \geq s + 1$
2	Correct upto 't' errors per word	$d_{min} \geq 2t + 1$
3	Correct upto 't' errors and detect $s > t$ errors per word	$d_{min} \geq t + s + 1$

Table 5.2 Error control capabilities

For the *(n, k)* block code, the minimum distance is given as,

$$d_{nin} \leq n - k + 1$$

Code efficiency: The code efficiency is the ratio of message bits in a block to the transmitted bit for that block by the encoder i.e.,

$$Code\ efficiency = \frac{message\ bits\ in\ a\ block}{transmitted\ bits\ for\ the\ black}$$

We know that for an (n, *k)* block code, there are *'k'* message bits and 'n' transmitted bits. Therefore code efficiency becomes,

$$Code\ efficiency = \frac{k}{n}$$

Comparing the above expression with the code rate (r), we find that,

$$Code\ efficiency = code\ rate = \frac{k}{n}$$

Weight of the code: The number of non-zero elements in the transmitted code vector is called vector weight. It is denoted by $w(X)$ where X is the code vector. For example if X= 0 1 110101, then weight of this code vector will be $w(X)$ =5.

5.2 THE CHANNEL CODING THEOREM

The **channel coding theorem** states that if a discrete memoryless channel has capacity C and a source generates information at a rate less than C, then there exists a coding technique such that the output of the source may be transmitted over the channel with an arbitrarily low probability of symbol error.

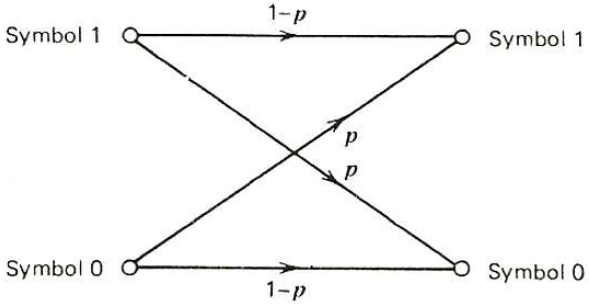
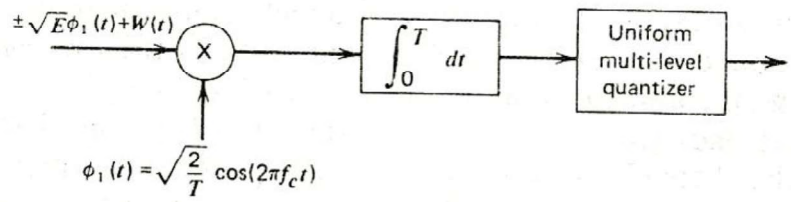
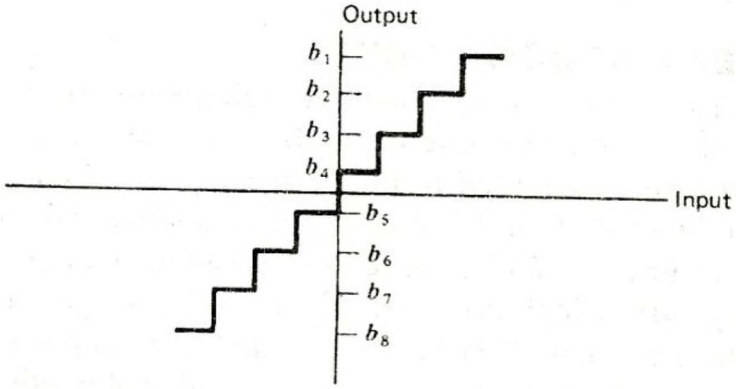


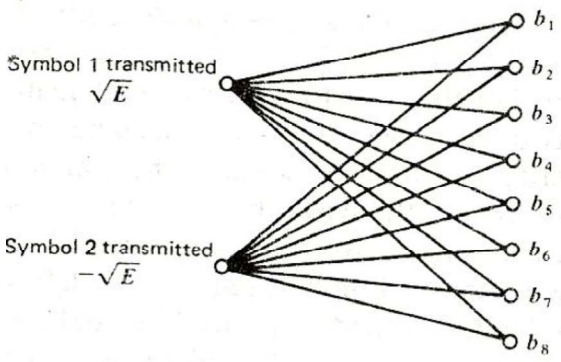
Figure 5.3 Transition distribution diagram of binary symmetric channel.



(a)



(b)



(c)

Figure 5.4 Binary input Q-ary output discrete memoryless channel *(a)* receiver, *b)* transfer characteristic of multilevel quantizer, *(c)* channel transition distribution. Parts *(b)* and (c) are illustrated for 8 levels.

For the special case of a binary symmetric channel, the theorem tells us that if the code rate *r* is less than the channel capacity C, then it is possible to find a code that achieves error-free transmission over the channel. Conversely, it is not possible to find such a code if the code rate *r* is greater than the channel capacity C.

The channel coding theorem thus specifies the channel capacity C as **fundamental limit** on the rate at which the transmission of reliable (error-free) messages can take place over a discrete memoryless channel. Thus the issue is not the signal-to-noise ratio but on how the channel input is encoded.

The disdvantage of the channel coding theorem is its nonconstructive nature. The theorem only asserts the **existence of good codes**. But, the theorem does not tell us how to find them. The

error-control coding techniques provide different methods of achieving this important system requirement.

5.3 LINEAR BLOCK CODES

Principle of block coding :

For the block of *k* message bits, (n- *k)* parity bits or check bits are added. Hence the total bits at the output of channel encoder are 'n'. Such codes are called (*n,k)* block codes. Fig.5.4 illustrates this concept.

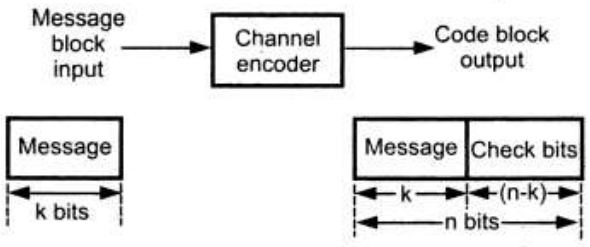


Fig. 5.5 Functional block diagram of block coder

Systematic codes : In the systematic block code, the message bits appear at the beginning of the code word. Thus as shown in Fig. 5.5, the message bits appear first and then check bits are transmitted in a block. This type of code is called systematic code.

Nonsystematic codes: In *Nonsystematic codes* it is not possible to identify message bits and check bits. They are mixed in the block.

In this section we will consider binary codes. That is all transmitted digits are binary.

Linear code: A code is 'linear' if the sum of any two code vectors produces another code vector. This shows that any code vector can be expressed as a linear combination of other code vectors. Consider that the particular code vector consists of $m_1, m_2, \ldots, m_k$ message bits and $c_1, c_2, c_3, \ldots, c_q$ check bits. Then this code vector can be written as,

$$X = (m_1, m_2, \ldots m_k\ c_1, c_2, \ldots c_q)$$

$$Here, \quad q = n - k$$

i.e. q are the number of redundant bits added by the encoder. The above code vector can also be written as,

$$X = (M|C)$$

$$Here, \quad M = k - bit\ message\ vector\ and$$

$$C = q - bit\ check\ vector$$

The check bits play the role of error detection and correction. The job of the linear block code is to generate those 'check bits'.

5.3.1 Matrix Description of Linear Blocks Codes

The code vector can be represented as,

$$X = MG \qquad --(1)$$

Here X = Code vector of *1 x n* size or *n* bits

M = Message *vector* of 1x *k* size or k bits

and G = Generator matrix of *k* x *n* size.

Thus equation (1) above represents matrix form i.e.

$$[X]_{1\times n} = [M]_{1\times k}[G]_{k\times n}$$

The generator matrix depends upon the linear block code used. Generally it is represented as,

$$G = [I_k | P_{k\times q}]_{k\times n} \qquad --(2)$$

$$Here, \quad I_k = k \times k\ identity\ matrix\ and$$
$$P = k \times q\ submatrix$$

The check *vector* can be obtained as,

$$C = MP \qquad --(3)$$

Thus in the expanded form we can write above equation as,

$$[C_1, C_2, \ldots, C_q]_{1\times q} = [m_1, m_2, \ldots m_k]_{1\times k} \begin{bmatrix} P_{11} & P_{12} & \ldots & P_{1q} \\ P_{21} & P_{22} & \ldots & P_{2q} \\ \vdots & & & \\ \vdots & & & \\ P_{k1} & P_{k2} & \ldots & P_{kq} \end{bmatrix}_{k\times q} \qquad --(4)$$

By solving the above matrix equation, check vector can be obtained. i.e.

$$\left.\begin{aligned} C_1 &= m_1 P_{11} \oplus m_2 P_{21} \oplus m_3 P_{31} \oplus \ldots \oplus m_k P_{k1} \\ C_2 &= m_1 P_{12} \oplus m_2 P_{22} \oplus m_3 P_{32} \oplus \ldots \oplus m_k P_{k2} \\ C_3 &= m_1 P_{13} \oplus m_2 P_{23} \oplus m_3 P_{33} \oplus \ldots \oplus m_k P_{k3} \\ &\ldots \text{ and so on} \end{aligned}\right\}$$

Here note that all the additions are mod-2 additions

Example 5.1 The generator matrix for a (6, 3) block code is given below. Find all code vectors of this code.

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

Solution : The code vectors can be obtained through following steps:

i) Determine the P subrnatrix from generator matrix.

ii) Obtain equations for check bits using C =*MP*.

iii) Determine check bits for every message vector.

i) To obtain *P′* sub matrix:

From equation (2) we know that,

$$G = [I_k : P_{k\times q}]$$

Comparing this equation with the given matrix, we find that,

$$I_k = I_{3\times 3}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$P_{k\times q} = P_{3\times 3}\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

ii) To obtain the equations for check bits:

Here k =3, q = 3 and n= 6.

That is, the block size of the message vector is 3 bits. Hence there will be total 8 possible message vectors as shown below in the table.

Sr.No.	Bits of message vector in one block		
	m_1	m_2	m_3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

The P submatrix is given in the example as,

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

For the check bit vector, there will be three bits. They can be obtained using equation (3) or (4). i.e,

$$[C_1 C_2 C_3] = [m_1 m_2 m_3] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

From the above matrix multiplication we obtain,

$$\begin{aligned} C_1 &= (0 \times m_1) \oplus (m_2) \oplus (m_3) \\ C_2 &= (m_1) \oplus (0 \times m_2) \oplus (m_3) \\ C_3 &= (m_1) \oplus (m_2) \oplus (0 \times m_3) \end{aligned}$$

From the above three equations we obtain,

$$\left.\begin{aligned} C_1 &= m_2 \oplus m_3 \\ C_2 &= m_1 \oplus m_3 \\ C_3 &= m_1 \oplus m_2 \end{aligned}\right\}$$

The above three equations give check bits for each block of m_1, m_2, m_3 message bits.

iii) **To determine check bits and code vectors for every message vector:**

Consider the first block of $(m_1, m_2, m_3) = 000$ we have,

$$\begin{aligned} C_1 &= 0 \oplus 0 = 0 \\ C_2 &= 0 \oplus 0 = 0 \\ C_3 &= 0 \oplus 0 = 0 \qquad \text{i.e.} \quad (C_1, C_2, C_3) = 000 \end{aligned}$$

For second block of $(m_1, m_2, m_3) = 001$ we have,

$$\begin{aligned} C_1 &= 0 \oplus 1 = 1 \\ C_2 &= 0 \oplus 1 = 1 \\ C_3 &= 0 \oplus 0 = 0 \qquad \text{i.e.} \quad (C_1, C_2, C_3) = 110 \end{aligned}$$

The following Table 5.1 lists all the message bits, their check bits and code vectors calculated as above.

Sr. No.	Bits of message vector in one block			Check bits			Complete code vector					
	m_1	m_2	m_3	$C_1 = m_2 \oplus m_3$	$C_2 = m_1 \oplus m_3$	$C_3 = m_1 \oplus m_2$	m_1	m_2	m_3	C_1	C_2	C_3
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	1	1	0	0	0	1	1	1	0
3	0	1	0	1	0	1	0	1	0	1	0	1
4	0	1	1	0	1	1	0	1	1	0	1	1
5	1	0	0	0	1	1	1	0	0	0	1	1
6	1	0	1	1	0	1	1	0	1	1	0	1
7	1	1	0	1	1	0	1	1	0	1	1	0
8	1	1	1	0	0	0	1	1	1	0	0	0

Table 5.3 Code vectors of (6, 3) block code of example 5.1

Parity check matrix (H)

For every block code there is a q x n parity check matrix (H). It is defined as,

$$H = [P^T : I_q]_{q \times n} \qquad --(5)$$

Here p^T is the transpose of P submatrix. The P submatrix is defined as,

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1q} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2q} \\ P_{31} & P_{32} & P_{33} & \cdots & P_{3q} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ P_{k1} & P_{k2} & P_{k3} & \cdots & P_{kq} \end{bmatrix}_{k \times q}$$

The transpose of this submatrix become (by changing rows to columns),

$$P^T = \begin{bmatrix} P_{11} & P_{21} & P_{31} & \cdots & P_{k1} \\ P_{12} & P_{22} & P_{32} & \cdots & P_{k2} \\ P_{13} & P_{23} & P_{33} & \cdots & P_{k3} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ P_{1q} & P_{2q} & P_{3q} & \cdots & P_{kq} \end{bmatrix}_{q \times k}$$

With the above equation, we can write equation (5) as

$$H_{q \times n} = \begin{bmatrix} P_{11} & P_{21} & P_{31} & \cdots & P_{k1} & : & 1 & 0 & 0 & \cdots & 0 \\ P_{12} & P_{22} & P_{32} & \cdots & P_{k2} & : & 0 & 1 & 0 & \cdots & 0 \\ P_{13} & P_{23} & P_{33} & \cdots & P_{k3} & : & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & : & \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & : & \vdots & \vdots & \vdots & \cdots & \vdots \\ P_{1q} & P_{2q} & P_{3q} & \cdots & P_{kq} & : & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{q \times n} \quad \text{--(6)}$$

If we compare equation (2) and equation (5), we find that if generator matrix (G) is given, then parity check matrix (H) can be obtained and vice versa.

5.3.2 Hamming Codes

Hamming codes are (n, k) linear block codes.

Those codes satisfy the following conditions,

(1) Number of check bits $q \geq 3$

(2) Block length $n = 2^q - 1$

(3) Number of message bits $k = n - q$

(4) Minimum distance $d_{min} = 3$

We know that the code rate is given as,

$$r = \frac{k}{n}$$

$$= \frac{n-q}{n} \; for \; hamming \; code \; \; k = n - q$$

$$= 1 - \frac{q}{n}$$

Putting the value of $n = 2^q - 1$ we get,

$$r = 1 - \frac{q}{2^q - 1} \qquad --(7)$$

From the above equation we observe that $r \approx 1 \ \ if \ q \gg 1$.

Example 5.2: Prove that $GH^T = HG^T = 0$ for a systematic linear block code.

Solution: We know that,

$$[G]_{k\times n} = \left[I_{k\times k} | P_{k\times q}\right]_{k\times m}$$

$$and \quad [H]_{q\times n} = \left[P^T_{q\times k} | I_{q\times q}\right]_{q\times n}$$

Now consider the matrix product HG^T i.e.,

$$HG^T = \left[P^T_{q\times k} | I_{q\times q}\right].\left[\frac{I_{k\times k}}{P^T{}_{q\times k}}\right]$$

$$= [P^T \oplus P^T]_{q\times k} = 0$$

Similarly the identity $GH^T = 0$ can be proved. i.e.,

$$GH^T = \left[I_{k\times k} | P_{k\times q}\right].\left[\frac{P_{k\times q}}{I_{q\times q}}\right]$$

$$= [P \oplus P] = 0$$

5.3.3 Error Detection and Correction Capabilities of Hamming Codes

Since the minimum distance *(d* min) of Hamming code is 3, it can be used to detect double errors or correct single errors. This can also be obtained from the generalized Table 5.2.

For detecting double (2) $errors \Rightarrow d_{min} \geq 2 + 1 \ \ i.e. d_{min} \geq 3$

and for correcting upto one (1) $errors \Rightarrow d_{min} \geq 2\,(1) + 1 \ \ i.e. d_{min} \geq 3$

Example 5.3: The parity check matrix of a particular (7, 4) linear block code is given by

$$[H] = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad --(8)$$

i) Find the generator matrix (G)
ii) List all the code vectors
iii) What is the minimum distance between code vectors ?
iv) How many errors can be detected? How many errors can be corrected?

Solution: Here n = 7 and k = 4
∴ Number of check bits are n - k = 7 - 4 i.e, q = 3
Thus $n = 2^q - 1 = 2^3 - 1 = 7$
This shows that the given code is hamming code.
To determine the P submatrix :
The parity check matrix is of q x *n* size and is given by eqn (6). It can be written as, (with q = 3 and n = 7 and k = 4),

$$[H]_{3\times 7} = \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} & \dots & 1 & 0 & 0 \\ P_{12} & P_{22} & P_{32} & P_{42} & \dots & 0 & 1 & 0 \\ P_{13} & P_{23} & P_{33} & P_{43} & \dots & 0 & 0 & 1 \end{bmatrix}$$
$$= [P^T : I_3]$$

On comparing parity check matrices of equation (8) and equation (7) we get,

$$P^T = \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{12} & P_{22} & P_{32} & P_{42} \\ P_{13} & P_{23} & P_{33} & P_{43} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Therefore the P submatrix can be obtained as,

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \\ P_{41} & P_{42} & P_{43} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}_{4\times 3}$$

i) To obtain the generator matrix (G):

The generator matrix G is given as,

$$G = [I_k : P_{k\times q}]_{k\times n}$$

with *k=4, q=3* and n=7 the above equation becomes,

$$G = [I_4 : P_{4\times 3}]_{4\times 7}$$

Putting the identity matrix I_4 of size 4 x 4 and parity submatrix $P_{4x\,3}$ of size 4 x 3 as obtained in equation (8) we get,

$$G = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{I_{4\times 4}} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}}_{P_{4\times 3}}_{4\times 7}$$

This is the required generator matrix.

ii) To find all the codewords :

To obtain equations for check bits

The check bits can be obtained using equation ,

$$C = MP$$

In the more general form (with q =3, k = 4)

$$[C_1\ C_2\ C_3]_{1\times 3} = [m_1\ m_2\ m_3\ m_4]_{1\times 4}[P]_{4\times 3}$$

$$[C_1\ C_2\ C_3] = [m_1\ m_2\ m_3\ m_4]\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}_{4\times 3}$$

Solving the above equation with mod-2 addition we get,

$$C_1 = (1\times m_1) \oplus (1\times m_2) \oplus (1\times m_3) \oplus (0\times m_4)$$
$$C_2 = (1\times m_1) \oplus (1\times m_2) \oplus (0\times m_3) \oplus (1\times m_4)$$
$$C_3 = (1\times m_1) \oplus (0\times m_2) \oplus (1\times m_3) \oplus (1\times m_4)$$

Thus the above equations are,

$$\begin{aligned} C_1 &= m_1 \oplus m_2 \oplus m_3 \\ C_2 &= m_1 \oplus m_2 \oplus m_4 \\ C_3 &= m_1 \oplus m_3 \oplus m_4 \end{aligned} \quad \text{--(9)}$$

To determine the code vectors

Consider for example $(m_1\ m_2\ m_3\ m_4) = 1\ 0\ 1\ 1$ we get,

$$C_1 = 1 \oplus 0 \oplus 1 = 0$$
$$C_2 = 1 \oplus 0 \oplus 1 = 0$$
$$C_3 = 1 \oplus 1 \oplus 1 = 1$$

Thus for message vector of (1 0 1 1) the check bits are $(C_1\ C_2\ C_3)$ =001.

Therefore the systematic block code of the code vector (codeword) can be written as,

$$(m_1 m_2 m_3 m_4 C_1 C_2 C_3) = (1\ 0\ 1\ 1 : 0\ 0\ 1)$$

Using the same procedure as given above, we can obtain the other codewords or code vectors. Table 5.3 lists all the code vectors (codewords). Table also lists the weight of each codeword.

Sr. No.	Message vector M				Check bits (C) by eq. 4.2.20			Code vector or codeword X							Weight of code vector w(X)
	m_1	m_2	m_3	m_4	C_1	C_2	C_3	m_1	m_2	m_3	m_4	C_1	C_2	C_3	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	1	1	0	0	0	1	0	1	1	3
3	0	0	1	0	1	0	1	0	0	1	0	1	0	1	3
4	0	0	1	1	1	1	0	0	0	1	1	1	1	0	4
5	0	1	0	0	1	1	0	0	1	0	0	1	1	0	3
6	0	1	0	1	1	0	1	0	1	0	1	1	0	1	4
7	0	1	1	0	0	1	1	0	1	1	0	0	1	1	4
8	0	1	1	1	0	0	0	0	1	1	1	0	0	0	3
9	1	0	0	0	1	1	1	1	0	0	0	1	1	1	4
10	1	0	0	1	1	0	0	1	0	0	1	1	0	0	3
11	1	0	1	0	0	1	0	1	0	1	0	0	1	0	3
12	1	0	1	1	0	0	1	1	0	1	1	0	0	1	4
13	1	1	0	0	0	0	1	1	1	0	0	0	0	1	3
14	1	1	0	1	0	1	0	1	1	0	1	0	1	0	4
15	1	1	1	0	1	0	0	1	1	1	0	1	0	0	4
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7

Table 5.3 Code vectors of example 5.3

iii) Minimum distance between code vectors

The Table 5.3 lists $2^k = 2^4 = 16$ code vectors along with their weights. The smallest weight of any non-zero code vector is 3. We know that the minimum distance is $d_{min} = 3$. Therefore we can write:

The minimum distance of n linear block code is equal to the minimum weight of any non zero code vector i.e.

$$d_{min} = [w(X)]_{min}; X \neq (0\ 0 \ldots 0)$$

iv) Error detection and correction capabilities

Since $d_{min} = 3$,

$$d_{min} \geq s + 1$$
$$3 \geq s + 1$$

$$s \leq 2$$

Thus two errors will be detected.
and

$$d_{min} \geq 2t + 1$$
$$3 \geq 2t + 1$$
$$t \leq 1$$

Thus one error will be corrected.
The hamming code ($d_{min} = 3$) always two errors can be detected and single error can be corrected by its property.

Example 5.4: Write the generator matrix and parity check matrix of (7, 4) hamming code

Solution:
Following is the parity check matrix of (7, 4) hamming code.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & : & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & : & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & : & 0 & 0 & 1 \end{bmatrix}_{3x7}$$
$$= [P^T_{3\,x\,4} ;\ I_{3\,x\,3}]$$

$$here, \quad p^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}_{3x7} \qquad \therefore P = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}_{4x3}$$

$$And, \quad G = [I_k \ : \ P_{k\,x\,q}]_{k\,x\,n}$$
$$= \begin{bmatrix} 1 & 0 & 1 & : & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & : & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & : & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & : & 0 & 0 & 0 & 1 \end{bmatrix}$$

5.3.4 Encoder of (7, 4) Hamming Code

Fig. 4.2.2 shows the encoder of (7,.4) Hamming code.

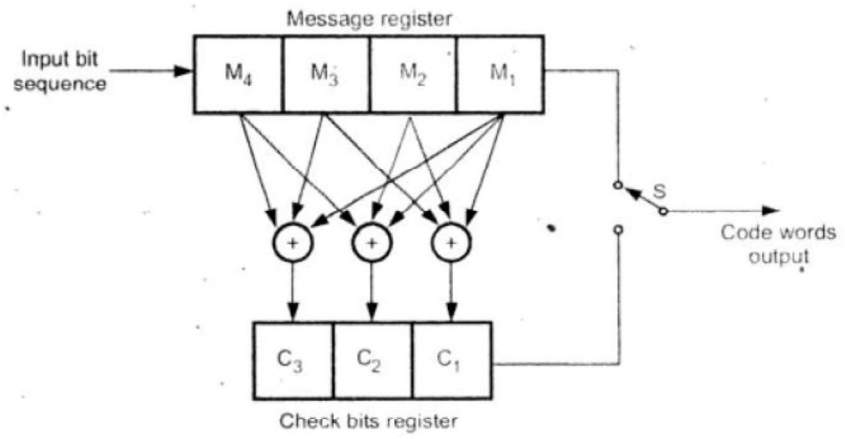


Fig. 5.6 Encoder for (7, 4) hamming code or (7, 4) linear block code

This encoder is implemented for generator matrix of the example 5.2. The lower register contains check bits c_1, c_2 *and* c_3. These bits are obtained from the message bits by mod-2 additions. These additions are performed according to equation (9). The mod-2 addition operation is nothing but exclusive-OR operation.

The switch 'S' is connected to message register first and all message bits are transmitted. The switch is then connected to the check bit register and check bits are transmitted. This forms a block of '7' bits. The input bits are then taken for next block.

5.3.5 Syndrome Decoding

It is a method used to correct errors in linear block coding. Let the transmitted code vector be $'X'$ and corresponding received code vector be represented by $'Y'$. Then we can write,

$$X = Y, \qquad if\ there\ are\ no\ transmission\ errors$$

$$and\ \ X \neq Y, \qquad if\ there\ are\ errore\ created\ during\ transmission$$

The decoder detects or corrects those errors in Y by using the stored bit pattern in the decoder about the code. For larger block length, more and more bits are required to be stored in the decoder. This increases the memory requirement and adds to the complexity and cost of the system. To avoid these problems, syndrome decoding is used in linear block codes.

We know that with every (n, k) linear block code, there exists a parity check matrix (h) of size q x n. It is defined as,

$$H = [p^T : I_q]_{\text{q x n}}$$

The transpose of the above matrix can be obtained by interchanging the rows and the columns, i.e.,

$$H^T = \begin{bmatrix} P \\ \dots \\ I_q \end{bmatrix}_{\text{n x q}}$$

Here P is the submatrix of size k x q and I_q is the identity matrix of size q x q.

Property

The transpose of parity check matrix (H^T) has very important as follows,

$$XH^T = (0\ \ 0\ \ 0\ \ \dots\ \ 0) \qquad --(10)$$

$$or \quad [H]_{1\,\text{x}\,\text{n}}\,[H^T]_{\text{n x q}} = (0\ \ 0\ \ 0\ \ \dots\ \ 0)_{1\,\text{x}\,q}$$

This is true for all code vectors.

For example consider the parity check matrix and code vectors obtained in example 5.3. The parity check matrix is given by eqn (8). The transpose of this matrix can be obtained as follows,

$$H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7 \times 3} \quad (n=7 \text{ and } q=3)$$

Table 5.2 lists all the code vectors for this parity check matrix. Consider the third code vector in this table.

$$X = (0\,0\,1\,0\,1\,0\,1)$$

Now let's apply the property of equation (10),

$$XH^T = [0010101]_{1\times 7} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7 \times 3}$$

Solving the above two matrices with the rules of mod-2 addition (Exclusive-OR operation) we get,

$$XH^T = (0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1)$$

$$= (0 \quad 0 \quad 0)$$

This proves the property. It can be proved for other code vectors also. Thus X belongs to the valid code vector at the transmitter. At the receiver, the received code vector is Y. Then we can write,

$$YH^T = (0\,0 \ldots 0), \qquad if\ X = Y\ i.e.\ no\ errors\ or\ Y\ is\ valid\ code\ vector$$

$$YH^T = Non-zero, \quad if\ X \neq Y\ i.e.some\ errors$$

Definition of syndrome (S)

When some errors are present in received vector Y, then it will not be from valid code vectors and it will not satisfy the property of equation (10). This shows that whenever YH^T is non-zero, some errors are present in Y. The non-zero output of the product YH^T is called syndrome and it is used to detect the errors in Y. Syndrome is represented by 'S' and can be written as,

$$S = YH^T \qquad --(11)$$
$$[S]_{1\times q} = [Y]_{1\times n}[H^T]_{n\times q}$$

Detecting error with the help of syndrome and error vector (E)

The non-zero elements of 'S' represent error in the output. When all elements of 'S' are zero, the two cases are possible.

i) 0 error in the output and Y = X
ii) Y is some other valid code vector other than X. This means the transmission errors are undetectable.

Let's consider on n-bit error vector E. Let this vector represent the position of transmission errors in Y. For example consider,

$$X = \begin{pmatrix} 1\ 0\ 1\ 1\ 0 \\ \uparrow \quad \uparrow \end{pmatrix} \qquad be\ a\ transmitted\ vector$$
$$and\ \ Y = \begin{pmatrix} 1\ 0\ 0\ 1\ 0 \\ \uparrow \quad \uparrow \end{pmatrix} \qquad be\ a\ received\ vector$$
$$Then, \quad E = (0\ 0\ 1\ 0\ 1) \qquad represents\ the\ error\ vector$$

The non-zero entries represent errors in Y.

Using the mod-2 addition rules we can write,

$$Y = X \oplus E \qquad --(12)$$
$$= (1 \oplus 0 \quad 0 \oplus 0 \quad 1 \oplus 1 \quad 1 \oplus 0 \quad 0 \oplus 1)$$

Bit by bit mod-2 addition

$$= (1\ 0\ 0\ 1\ 1)$$

or we can write,

$$X = Y \oplus E$$
$$= (1 \oplus 0 \quad 0 \oplus 0 \quad 0 \oplus 1 \quad 1 \oplus 0 \quad 1 \oplus 1)$$
$$= (1\ 0\ 1\ 1\ 0)$$

Relationship between syndrome vector (S) and error vector (E)

From equation (11) we know that syndrome vector is given as,

$$S = YH^T$$

Putting the value of $Y = (X \oplus E)H^T$. From eqn (12)

$$S = (X \oplus E)H^T$$

$$= XH^T \oplus EH^T$$

From the property of eqn (10) we know that $XH^T = 0$, then above equation will be,

$$S = EH^T \qquad --(13)$$

This relation shows that syndrome depends upon the error pattern only. It does not depend upon a particular message. Syndrome vector 'S' is of size 1 x q. Thus q bits of syndrome can only represent 2^q syndrome vectors. Each syndrome vector corresponds to a particular error pattern.

Example 5.5: The parity check matrix of a (7, 4). Hamming code is given as follows,

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

Calculate the syndrome vector for single bit errors.

Solution: This is a (7,4) linear block code

This is $n = 7 \ and \ k = 4$

$$q = n - k = 3$$

i) To determine error pattern for single bit errors

We know that syndrome vector is a q bit vector. For this example syndrome will be a 3 bit vector. Therefore there will be $2^3 - 1 = 7$ non zero syndromes. This shows that '7' single bit error patterns will be represented by these '7' non-zero syndromes. Error vector E is a n bit vector representing error pattern. For this example E is '7' bit vector. Following Table 5.4 shows the single error patterns in a 7 bit error vector. (Note that only single bit error patterns are shown)

S.No	Bit in error	Bits of Error vector (E), Non-zero bits shows error						
1.	1st	1	0	0	0	0	0	0
2.	2nd	0	1	0	0	0	0	0
3.	3rd	0	0	1	0	0	0	0
4.	4th	0	0	0	1	0	0	0
5.	5th	0	0	0	0	1	0	0
6.	6th	0	0	0	0	0	1	0
7.	7th	0	0	0	0	0	0	1

Table 5.4 Single error pattern of 7 bit error vector

ii) Calculation of syndromes

From equation (13) the syndrome vector is given as,

$$S = EH^T$$

s is q bit, E is n bit and H^T is n x q bits size. For this example $n=7$, $q=3$ we can write above equation as,

$$[S]_{1\times3} = [E]_{1\times7}[H^T]_{7\times3}$$

From the given parity check matrix H, we can obtain its transpose (H^T) by interchanging rows to columns, i.e,

$$H^T = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Syndrome for first bit in error

Let's calculate syndrome for first bit error vector i.e.

$$S = EH^T = [1000000]\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= (1\oplus0\oplus0\oplus0\oplus0\oplus0\oplus0 \quad 0\oplus0\oplus0\oplus0\oplus0\oplus0\oplus0 \quad 1\oplus0\oplus0\oplus0\oplus0\oplus0\oplus0)$$

$$= [1 \quad 0 \quad 1]$$

This is the syndrome vector for first bit in error.

Syndrome for second bit in error

Let's calculate syndrome for 2^{nd} bit in error. In Table 5.3 the error vector for error in 2^{nd} bit is given. We can, write

$$S = EH^T = [0100000]\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= (0\oplus1\oplus0\oplus0\oplus0\oplus0\oplus0 \quad 0\oplus1\oplus0\oplus0\oplus0\oplus0\oplus0 \quad 0\oplus1\oplus0\oplus0\oplus0\oplus0\oplus0)$$

$$= [1 \quad 1 \quad 1]$$

This is the syndrome vector for second bit in error.

Syndrome vectors are rows of H^T

The Table 5.5 lists the error vector with single bit error and corresponding syndromes. Other syndromes can be calculated using the same procedure as above. The table also lists the syndrome for no error vector i.e. $E = (0000000)$. Observe that the corresponding syndrome is S = (0 0 0).

The following table shows that error in the first bit corresponds to a syndrome vector of S = (101).This syndrome vector is same as the first row of H^T. The syndrome vector (S= 111) for error in second bit is same as second row of H^T. This is same for remaining syndromes.

Sr. No	Error vector 'E' showing single bit error patterns							Syndrome Vector 'S'			
1	0	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	1	0	1	← 1st row of H^T
3	0	1	0	0	0	0	0	1	1	1	← 2nd row of H^T
4	0	0	1	0	0	0	0	1	1	0	← 3rd row of H^T
5	0	0	0	1	0	0	0	0	1	1	← 4th row of H^T
6	0	0	0	0	1	0	0	1	0	0	← 5th row of H^T
7	0	0	0	0	0	1	0	0	1	0	← 6th row of H^T
8	0	0	0	0	0	0	1	0	0	1	← 7th row of H^T

Table 5.5 Syndromes for (7, 4) Hamming code of single bit error

5.3.5.1 Error Correction using Syndrome Vector

Let's see how single bit errors can be corrected using syndrome decoding. We will see this for (7, 4) block code. Let the transmitted code vector be,

$$X = (1\,0\,0\,1\,1\,1\,0)$$

Let there be error created in the 3^{rd} bit in the received code vector Y. Then Y will be

$Y = (1\,0\,\boxed{1}\,1\,1\,1\,0)$ boxed bit shows it is in error.

Now error correction can be done by adopting following steps:

i) Calculate the syndrome $S = YH^T$
ii) Check the row of H^T which is same as 'S'.
iii) For p^{th} row of H^T, p^{th} bit is in error. Hence write corresponding error vector (E).
iv) Obtain correct vector by $X = Y \oplus E$

Above procedure is illustrated next:

i) To obtain syndrome vector (5)

Let's use the parity check matrix and syndrome vectors of example 5.2 for this illustration. The receiver calculates $S = YH^T$ i.e.

$$S = YH^T = \underbrace{[1\ 0\ 1\ 1\ 1\ 1\ 0]}_{Y} \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{H^T}$$

$$= (1\oplus 0\oplus 1\oplus 0\oplus 1\oplus 0\oplus 0 \quad 0\oplus 0\oplus 1\oplus 1\oplus 0\oplus 1\oplus 0 \quad 1\oplus 0\oplus 0\oplus 1\oplus 0\oplus 0\oplus 0)$$

$$= [1\ \ 1\ \ 0]$$

From equation (11) and equation (13) we can write

$$S = YH^T = EH^T \qquad here \quad S = YH^T = EH^T = (1\ 1\ 0)$$

ii) To determine row of H^T which is same as 'S' and (iii) To determine 'E'

On comparing this syndrome with H^T, we observe that (S= 1 1 0) is the 3^{rd}, row of H^T. *From* Table 5.4 we can obtain the error pattern corresponding to this syndrome as,

$$E = (001\ 0\ 000)$$

This shows that there is an error in the third bit of Y. We have also verified that, if syndrome vector is equal to 3^{rd} row of H^T, then third bit of Y is in error.

iv) To obtain correct vector

The correct vector can-be obtained from equation (12) as,

$$X = Y \oplus E$$

i.e. $X = [1\ 0\ 1\ 1\ 1\ 1\ 0]\oplus[0\ 0\ 1\ 0\ 0\ 0\ 0]$

$= (1\ 0\ 0\ 1\ 1\ 1\ 0)$ which is same as transmitted code vector

Thus a single bit errors can be corrected using syndrome decoding.

If double error occurs in Y,

Let's see the case of double error in Y. Consider the same message vector X .i.e.,

$$X = 1001110$$

Let's consider that error is present in 3^{rd} and 4^{th} bits. Then Y will be

$Y = 1\ 0\ \boxed{1}\ \boxed{0}\ 1\ 1\ 0$ boxed bits are in error.

Then $S = YH^T$ gives,

$$S = YH^T = [1\ 0\ 1\ 0\ 1\ 1\ 0] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = 101$$

From Table 5.4 we observe that the syndrome S = 101 corresponds to an error pattern of E =1000000.This shows that there is an error in the first bit. Thus the error detection and correction goes wrong. The probability of occurrence of multiple errors is less compared to single errors. To correct multiple errors, extended hamming codes are used. In these codes one more extra bit is provided to correct double errors.

We know that for (n, *k)* block code, there are $2^q - 1$ distinct non-zero syndromes. There are $nC_1 = n$ single error patterns. nC_2 double error patterns, . nC_3 triple error patterns and so on. Therefore to correct 't' errors per word the following relation should be satisfied,

$$2^q - 1 \geq nC_1 + nC_2 + nC_3 + \cdots + nC_t \qquad --(14)$$

5.3.6 Hamming Bound

We can write equation (14) as,

$$2^q \geq 1 + nC_1 + nC_2 + nC_3 + \cdots + nC_t$$

$$\geq \sum_{i=0}^{t} nC_i$$

We know that q =n - k. Then the above equation becomes,

$$2^{n-k} \geq \sum_{i=0}^{t} nC_i$$

By taking logarithm to base 2 on both sides we get,

$$n - k \geq \log_2 \sum_{i=0}^{t} nC_i$$

Dividing both sides by n we get,

$$1 - \frac{k}{n} \geq \frac{1}{n} \log_2 \sum_{i=0}^{t} nC_i$$

Since code rate $r = \frac{k}{n}$ the above equation will be,

$$1 - r \geq \frac{1}{n} \log_2 \sum_{i=0}^{t} nC_i$$

This equation relates code rate' r' with the error correction capability of .t' errors per code vector in a block of $'n'$ bits. We know that the error correcting capability of the code (i.e.. t' errors per code vector) is related to the minimum distance. This minimum distance is also called hamming distance. Equation 4.2.34 gives the relation between code rate, number of errors to be corrected and number of bits jn a block. This equation is also called hamming bound.

5.3.7 Syndrome Decoder for (n, k) Block Code

Fig. 5.5 shows the block diagram of a syndrome decoder for linear block code to correct errors. The received n-bit vector $'Y$ is stored in an II-bit register. From this vector a syndrome is calculated using,

$$S = YH^T$$

Thus H^T is stored in the syndrome calculator. The q-bit syndrome vector is then applied to a look up table of error patterns. Depending upon the particular syndrome an error pattern is selected. This error pattern is added (mod - 2 addition) to the vector Y. The output is thus,

$$Y \oplus E = X$$

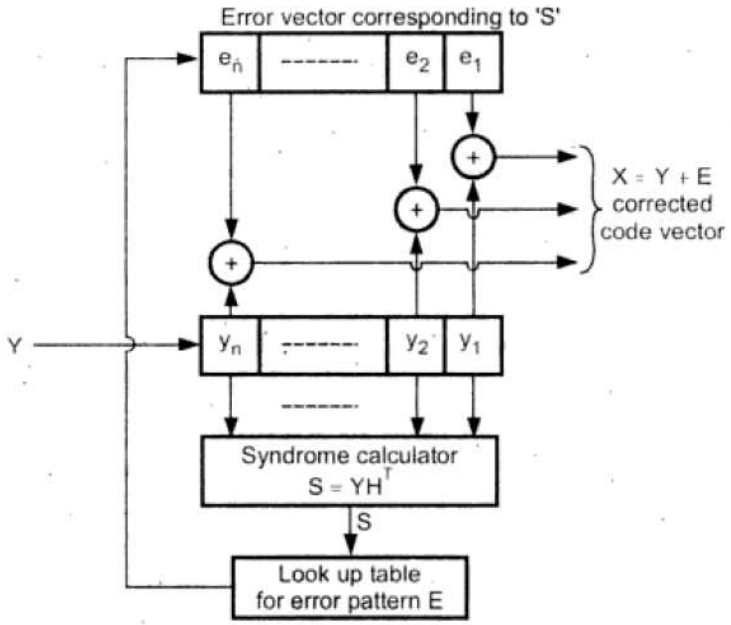


Fig. 5.5 Syndrome decoder for linear block code

The block diagram shown above can correct only single errors in the above vectors.

Maximum likelihood decoding for linear block codes:

We know that there are 2^q different syndromes. These syndromes can only represent $2^q - 1$ error patterns. But in an n-bit vector, there can be 2^n error patterns. Hence syndrome does not

uniquely represent error vector (E). With the help of syndrome we can correct only $2^q - 1$ error patterns and remaining patterns are uncorrectable. Single errors are more common than double and higher errors. Therefore single error patterns are most likely compared to double and higher error patterns. Therefore syndrome decoding corrects single errors which are most likely. Hence syndrome decoding is called **maximum likelihood decoding.** The maximum likelihood decoding selects the code vector that has the smallest hamming distance from received vector. Such code vector is obtained by

X= Y + E, Here Y is received vector

and 'E' is the most likely error pattern. This error pattern is selected based on calculated syndrome. The maximum likelihood decoding minimizes the word error probability.

5.4 CYCLIC CODES

Cyclic codes are the sub class of linear block codes. Cyclic codes can be in systematic or nonsystematic form. In systematic form, check bits are calculated separately and the code vector is X= (M:C) form. Here 'M' represents message bits and 'C' represents check bits.

5.4.1 Definition of Cyclic Code

A linear code is called cyclic code if every cyclic shift of the code vector produces some other code vector. This definition includes two fundamental properties of cyclic codes.

5.4.2 Properties of Cyclic Codes

As defined above, cyclic codes exhibit two fundamental properties :

1. Linearity and 2. Cyclic property

5.4.2.1 Linearity Property

This property states that sum of any two codewords is also a valid codeword. For example let X_1 and X_2 are two codewords. Then,

$$X_3 = X_1 \oplus X_2$$

Here X_3 is also a valid codeword. This property shows that cyclic code is also a linear code.

5.4.2.2 Cyclic Property

Very cyclic shift of the valid code vector produces another valid code vector. Because of this property, the name 'cyclic' is given. Consider an n-bit code vector as shown below:

$$X = \{x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0\}$$

Here $x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0$ etc. represent individual bits of the code vector 'X'. Let us shift the above code vector cyclically to left side. i.e.,

$$one\ cyclic\ shift\ of\ X\ gives, \qquad X' = (x_{n-2}, x_{n-3}, \ldots\ldots, x_1, x_0, x_{n-1})$$

Here observe that every bit is shifted to left by one position. Previously x_{n-1} was MSB but after left cyclic shift it is at LSB position. Here the new code vector is X' and it is valid code vector. One more cyclic shift yields another code vector X''. i.e.,

$$X'' = (x_{n-3}, x_{n-4}, \ldots. x_1, x_0, x_{n-1}, x_{n-2})$$

Here observe that x_{n-3} is now at MSB position and x_{n-2} is at LSB position.

5.4.3 Algebraic Structures of Cyclic Codes

The codewords can be represented by a polynomial.

For example, consider the n-bit codeword,

$$X = (x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0)$$

This codeword can be represented by a polynomial of degree less than or equal to (n-1). i.e.,

$$X(p) = x_{n-1}p^{n-1}, x_{n-2}p^{n-2}, \ldots\ldots, x_1 p, x_0$$

Here $X(p)$ is the polynomial of degree (n-1)

P is the arbitrary variable of the polynomial

The power of 'p' represents the positions of the codeword bits. i.e.,

p^{n-1} represents MSB

p^0 represents LSB

p^1 represents second bit from LSB side.

Why to represent codewords by a polynomial?

Polynomial representation is used due to following reasons:

i) These are algebraic codes. Hence algebraic operations such as addition, multiplication, division, subtraction etc. becomes very simple.

ii) Positions of the bits are represented with the help of powers of p in a polynomial.

5.4.3.1 Generation of Code vectors in Nonsystematic Form

Let $M = \{m_{k-1}, m_{k-2}, \ldots m_1, m_0\}$ be 'k' bits of message vector. Then it can be represented by the polynomial as,

$$M(p) = m_{k-1}\, p^{k-1} + m_{k-2}\, p^{k-2} + \cdots + m_1 p = m_0$$

Let $X(p)$ represent the codeword polynomial. It is given as,

$$\boxed{X(p) = M(p)G(p)}$$

Here $G(p)$ is the generating polynomial of degree 'q'. For an *(n,k)* cyclic code, q = n - k represent the number of parity bits. The generating polynomial is given as,

$$G(p) = p^q + g_{q-1}p^{q-1} + \cdots + g_1 p + 1$$

Here $g_{q-1}, g_{q-2}, \ldots g_1$ are the parity bits.

If M_1 M_2. M_3 etc are the other message vectors, then the corresponding code vectors can be calculated as

$$X_1(p) = M_1(p)G(p)$$

$$X_2(p) = M_2(p)G(p)$$

$$X_3(p) = M_3(p)G(p) \ \ and\ so\ on$$

All the above code vectors $X_1, X_2, X_3 \ldots$ are in nonsystematic form and they satisfy cyclic property. Note the generator polynomial $G(p)$ remains the same for all code vectors.

5.4.3.2 Generation of Code vectors in Systematic Form

Now let us study systematic cyclic codes. The systematic form of the block code is,

$$X = (k\ message\ bits : (n-k) check\ bits)$$

$$= (m_{k-1}m_{k-2} \ldots m_1 m_0 : c_{q-1}c_{q-2} \ldots c_1 c_0)$$

Here the check bits form a polynomial as,

$$C(p) = c_{q-1}p^{q-1}c_{q-2}p^{q-2} \ldots c_1 p c_0$$

The check bit polynomial is obtained by

$$\boxed{C(p) = rem\left[\frac{p^q M(p)}{G(p)}\right]}$$

Above equation means -

i) Multiply message polynomial by p^d.

ii) Divide $p^q M(p)$ by generator polynomial.

iii) Remainder of the division is $C(p)$.

5.4.4 Generator and Parity Check Matrices of Cyclic Codes

5.4.4.1 Nonsystematic Form of Generator Matrix

Since cyclic codes are subclass of linear block codes, generator and parity check matrices can also be defined for cyclic codes. The generator matrix has the size of k x n. That means there are $'k'$ rows and 'n' columns. Let the generator matrix $G(p)$ be given by,

$$G(p) = p^q + g_{q-1}p^{q-1} + \cdots + g_1 p + 1$$

Multiply both the sides of this polynomial by p^i i.e.,

$$\boxed{p^i G(p) = p^{i+q} + g_{q-1}p^{i+q-1} + \cdots + g_1 p^{i+1} + p^i}$$

The above equation gives the polynomials for the rows of generating polynomials.

5.4.4.2 Systematic Form of Generator Matrix

The systematic form of generator matrix is given by,

$$G = \left[I_k : P_{k\times q}\right]_{k\times n}$$

The t^{th} row of this matrix will be represented in the polynomial form as,

$$\boxed{t^{th}\ row\ of\ G = p^{n-t} + R_t(p)} \qquad where\ t = 1,2,3,\ldots k$$

Let's divide p^{n-t} by a generator matrix $G(p)$. Then we can express the result of this division in terms of quotient and remainder. i.e.,

$$\frac{p^{n-t}}{G(p)} = Quotient + \frac{Remainder}{G(p)} \qquad --(1)$$

Here remainder will be a polynomial of degree less than $'q'$, since degree of $G(p)$ is $'q'$. The degree of quotient will depend upon value of t

Let's represent Remainder = $R_t(p)$

and Quotient = $Q_t(p)$

then equation (1) will be,

$$\frac{p^{n-t}}{G(p)} = Q_t(p) + \frac{R_t(p)}{G(p)}$$

$$i.e.\quad p^{n-t} = Q_t(p)G(p) \oplus R_t(p) \quad and \quad t = 1,2,\ldots t$$

We know that if $z = y \oplus t$, then $z \oplus y = t$ or $z \oplus t = y$. That is mod-2 addition and subtraction yields same results. Then we can write above equation as,

$$p^{n-t} \oplus R_t(p) = Q_t(p)G(p)$$

The above equation represents t^{th} row of systematic generator matrix.

5.4.4.3 Parity Check Matrix

Once the generator matrix in systematic form is obtained then parity check matrix can be obtained as per the procedure discussed in last section.

5.4.5 Encoding using an (n - k) Bit Shift Register

Fig. 5.7 shows the block diagram of a generalized (n, k) cyclic code. The symbols used to draw encoders are shown in Fig. 5.6.

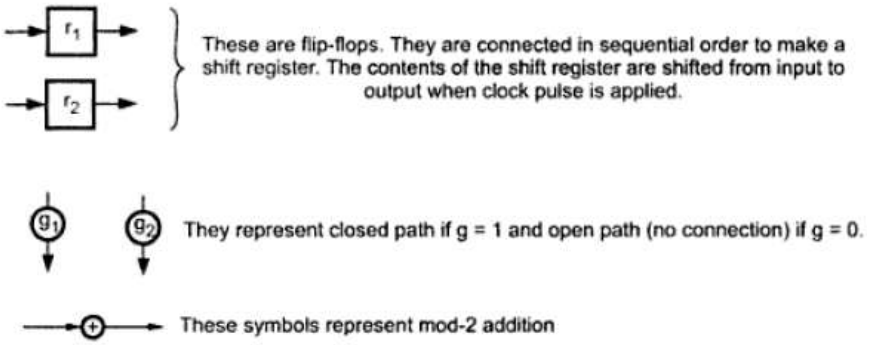


Fig. 5.6 Various symbols used in encoder

Operation: The feedback switch is first closed. The output switch is connected to message input. All the shift registers are initialized to all zero state. The *k* message bits are shifted to the transmitter as well as shifted into the registers.

After the shift of *.k′* message bits the registers contain *' q′* check bits. The feedback switch is now opened and output switch is connected to check bits position. With the every shift, the check bits are then shifted to the transmitter.

Here we observe that the block diagram performs the division operation and generates the remainder (i.e. check bits). This remainder is stored in the shift register after all message bits are shifted out.

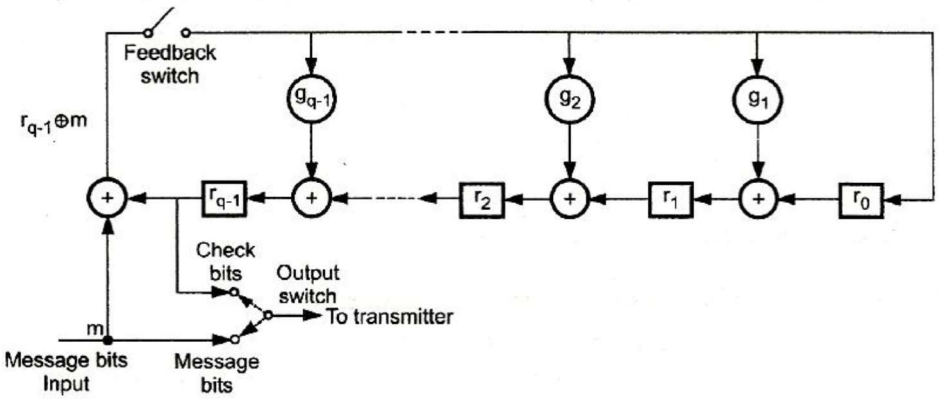


Fig 5.7 Encoder for Systematic (n,k) cyclic code

5.4.6 Syndrome Decoding, Error Detection and Error Correction

In cyclic codes also during transmission some errors may occur. Syndrome decoding can be used to correct those errors. Let's represent the received code vector by Y. If' 'E' represents an error vector then the correct code vector can be obtained as,

$$X = Y \oplus E$$

or we can write the above equation as,

$$Y = X \oplus E$$

We can write the above equation since it is rnod-2addition.

In the polynomial form we can write the above equation as,

$$Y(p) = X(p) + E(p)$$

$$since, \quad X(p) = M(p)G(p) \quad the\ above\ equation\ will\ be,$$

$$Y(p) = M(p)G(p) + E(p) \qquad --(1)$$

Let the received polynomial $Y(p)$ be divided by $G(p)$ i.e.

$$\frac{Y(p)}{G(p)} = Quotient + \frac{Remainder}{G(p)} \qquad --(2)$$

In the above equation if $Y(p) = X(p)$ i.e. if it does not contain any error then,

$$\frac{X(p)}{G(p)} = Quotient + \frac{Remainder}{G(p)}$$

Since $X(p) = M(p)G(p)$, quotient will be equal to $M(p)$ and remainder will be zero. This shows that if there is no error, then remainder will be zero. Here $G(p)$ is factor of code vector polynomial. Let's represent quotient by $Q(p)$ and remainder by $R(p)$ then equation (2) becomes,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{R(p)}{G(p)}$$

Clearly $R(p)$ will be the polynomial of degree less than or equal to $q - 1$. Multiply both sides of above equation by G *(p)* i.e,

$$Y(p) = Q(p)G(p) + R(p) \qquad --(3)$$

On comparing equation (1) and above equation (3) we obtain,

$$M(p)G(p) \oplus E(p) = E(p)G(p) \oplus R(p)$$

$$\therefore \quad E(p) = M(p)G(p) \oplus Q(p)G(p) \oplus R(p)$$

The above equation has all mod-2 additions. Therefore subtraction and addition is same.

$$\therefore \quad E(p) = [M(p) + Q(p)]G(p) + R(p)$$

This equation shows that for a fixed message vector and generator polynomial, an error pattern or error vector' E depends on remainder R. For every remainder' R' there will be specific error vector. Therefore we can call the remainder vector *'R'* as syndrome vector '5', or R *(p)* =5 *(p).* Therefore equation (3.3.59) will be,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{S(p)}{G(p)}$$

Thus the syndrome vector is obtained by dividing received vector $Y(p)$ by $G\ (p)$, i.e,

$$\boxed{S(p) = rem\left[\frac{Y(p)}{G(p)}\right]}$$

5.4.6.1 Block Diagram of Syndrome Calculator

Fig. 5.8 shows the generalized block diagram of a syndrome calculator.

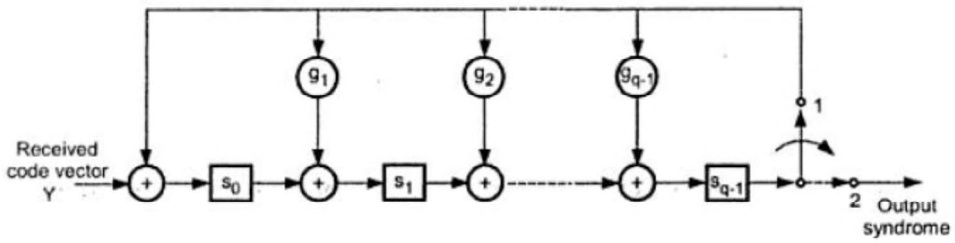


Fig. 5.8 Computation of syndrome for an (n, k) cyclic code

In above figure there are 'q' stage shift register to generate' q' bit syndrome vector.

Operation: Initially all the shift register contents are zero and the switch is closed in position 1. The received vector Y is shifted bit by bit into the shift register. The contents of flip flops keep on changing according to input bits of Y and values of g_1, g_2 etc. After all the bits of Y are shifted, the 'q' flip-flops of shift register contains the q - bit syndrome vector. The switch is then closed to position 2 and clocks are applied to the shift register. The output is a syndrome vector $S = (s_{q-1}, s_{q-2}, \ldots s_1, s_0)$

5.4.7 Decoder for Cyclic Codes

Once the syndrome is calculated, then an error pattern is detected for that particular syndrome. When this error vector is added to the received vector Y, then it gives corrected code vector at the output. This decoding operation can be performed by the scheme shown in Fig. 5.9.

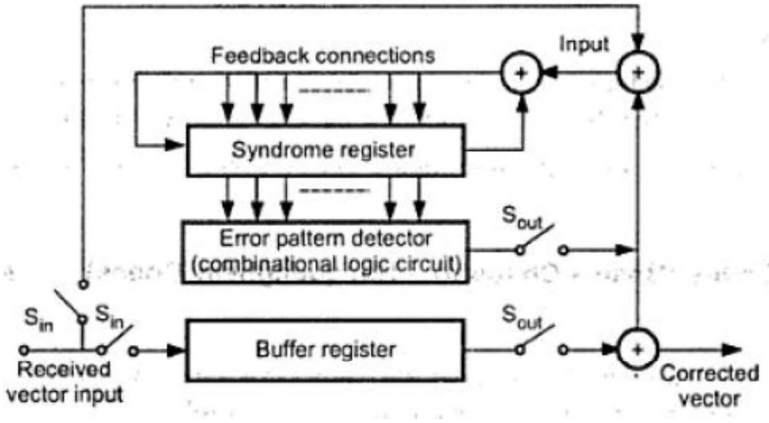


Fig. 5.9 Generalized block diagram of decoder for cyclic codes

Operation of the decoder

The switches named S_{out} are opened and S_{in} are closed. The bits of the received vector Y are shifted into the buffer register as well as they are shifted into the syndrome calculator. When all the 'n' bits of the received vector Y are shifted into buffer register and syndrome calculator the syndrome register holds a syndrome vector. The syndrome vector is given to the error pattern detector. A particular syndrome detects a specific error pattern. The switches S_{in} are opened and S_{out} are closed. The shifts are then applied to the flip-flops of buffers register, error register (which holds error pattern) and syndrome register. The error pattern is then added bit by bit to the received vector (which is stored in buffer register). The output is the **corrected error free vector.**

5.4.8 Advantages and Disadvantages of Cyclic Codes

As we have seen that cyclic codes are the subclass of linear block codes, they have some advantages over non-cyclic block codes as given below.

Advantages :

1) The error correcting and decoding methods of cyclic codes are simpler and easy to implement. These methods eliminate the storage needed for lookup table decoding. Therefore the codes become powerful and efficient.

2) The encoders and decoders for cyclic codes are simpler compared to non-cyclic codes.

3) Cyclic codes also detect error burst that span many successive bits.

4) Cyclic codes have well defined mathematical structure. Hence very efficient decoding schemes are possible.

In spite of these advantages cyclic codes also have some disadvantages.

Disadvantages :

1) The error detection in cyclic codes is simpler but error correction is little complicated since the combinational logic circuits in error detector are complex.

To avoid such complex circuits some special cyclic codes are used which are discussed next.

5.4.9 BCH Codes (Bose - Chaudhri - Hocquenghem Codes)

Features

- BCH codes are most extensive and powerful error correcting cyclic codes. The decoding of BCH codes is comparatively simpler.
- For any positive integer m and t (where $t < (2^m - 1) / 2$) there exists a BCH code with following parameters

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

  Number of parity check bits : $n - k \leq mt$

  Minimum distance : $d_{min} \geq 2t + 1$

- Each BCH code can detect and correct upto 't' random errors.
- BCH codes have flexibility in selection of block length and code rate.
- BCH codes are the best codes for block lengths upto few hundred bits.

The decoding schemes of BCH codes can be implemented on digital computer. Because of software implementation of decoding schemes they are flexible compared to hardware implementation of other schemes.

5.4.10 Reed-Soloman (RS) Codes

Features

- These are nonbinary BCH codes. The encoder of RS codes operates on multiple bits simultaneously.
- The (n, k) RS code takes the groups of m-bit symbols of the incoming binary data stream. It takes such 'k' number of symbols in one block. Then the encoder adds (n - k) redundant symbols to form the codeword of 'n' symbols,
- Thus there are 'n' symbols in the codeword. Each symbol contains 'm' number of bits. Normally m = 8 is taken.
- The 't' error correcting RS code has the following parameters:

  Block length: n= $2^m - 1$ symbols
  Message size: k symbols
  Parity check size: n - k = 2t symbols
  Minimum distance: d_{min} = 2t + 1 symbols.

- Here observe that the minimum distance is greater than the number of parity symbols. Hence this code is maximum distance separable code.
- These codes provide wide range of code rates. Efficient decoding techniques are available with RS codes.

5.4.11 Burst Error Correcting Codes

The above codes detect and correct errors occurring independently at different bit positions. Burst errors occur as a cluster of errors. Cyclic and shortened codes can be used to detect these burst errors.

The burst of length q is defined as the vectors whose nonzero components are limited to 'q' consecutive digit positions with nonzero first and last digits. For example the vector $x = [0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]$ has the burst of length 9. The q – burst error correcting code is capable of correcting the bursts length q or less. The following theorem gives the number of parity bits required by burst error correcting code:

The q-burst error correcting code must have at least 2q parity check digits i.e.,

$$n - k \geq 2q$$

Thus we can say that the (n, k) burst error correcting code can correct the bursts *n-k* of length up $\frac{n-k}{2}$. This becomes the upper bound on the burst error correcting capability of (n, k) code i.e.,

$$q \leq \frac{n-k}{2}$$

The burst error correcting efficiency is denoted by z. It is given as,

$$z = \frac{2q}{n-k}$$

To detect the burst of length d, then the check bits must be,

$$n - k \geq d$$

Thus the check bits must be at least equal to d.

5.4.12 Cyclic Redundancy Check (CRC) Codes

Definition: A cyclic code which is used for **error detection** purpose only is called cyclic redundancy check (CRC) code.

Characteristics

Why CRC codes ?

Cyclic codes are very much suitable for error detection because of two reasons :

i) Many combinations likely errors can be detected with the help of cyclic codes.

ii) Implementation of encoding and error detection circuits is practically possible.

Error detection capabilities of binary (n, k) CRC codes

The CRC codes are capable of detecting,

i) All error bursts of length (n - *k)* or less.

ii) Fraction of error bursts of length equal to (n- k + 1).

iii) Fraction of error bursts of length greater than (n - k +1).

iv) All error combinations of $(d_{min} - 1)$ or less.

v) If generator polynomial $G(p)$ has even number of coefficients, than all error patterns with odd number of errors can also be detected.

Commonly used CRC codes

Three commonly used CRC codes are given below:

CRC-12: $G(p) = 1 + p + p^2 + p^{11} + p^{12},\ with\ n - k = 12$

CRC-16: $G(p) = 1 + p^2 + p^{15} + p^{16},\ with\ n - k = 16$

CRC-ITU: $G(p) = 1 + p^5 + p^{12} + p^{16},\ with\ n - k = 16$

All the above codes contain $1 + p$ as a prime factor. CRC-12 code is used for 6-bit characters. CRC-16and CRC-ITU are used for 8-bit characters.

Applications:

1) CRC codes are used mainly in ARQ systems for error detection.

2) They are also used in digital subscriber lines.

5.4.13 Maximum length codes

Features

- The maximum length codes has $m \geq 3$ and it has following parameters :

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

  Number of message bits: $k = m$

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

- The maximum length codes are generated by the polynomial which is given as,

$$G(p) = \frac{p^n + 1}{H(p)}$$

Here $H(p)$ is the primitive polynomial of degree 'm'

- The polynomial $H(p)$ defines feedback connections of an encoder and $G(p)$ defines one period of maximal length code
- Maximum length codes are dual of hamming codes

5.5 CONVOLUTIONAL CODES

5.5.1 Definition of Convolutional Coding

A convolutional coding is done by combining the fixed number of input bits. The input bits are stored in the fixed length shift register and they are combined with the help of mod-2 adders. This operation is equivalent to binary convolution and hence it is called **convolutional coding**. This concept is illustrated with the help of simple example given below.

Fig. 5.10 shows a convolutional encoder.

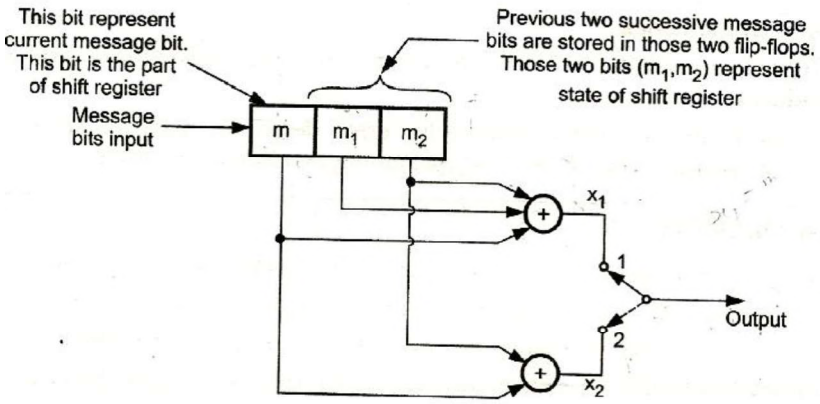


Fig. 5.10 Convolutional encoder with k = 3, k =1 and n = 2

Operation:

Whenever the message bit is shifted to position *'m'*, the new values of x_1 *and* x_2 are generated depending upon m, m_1 *and* m_2. m_1 *and* m_2 store the previous two message bits. The current bit is present in *m*. Thus we can write,

$$x_1 = m_1 \oplus m_1 \oplus m_2 \qquad --(1)$$

$$and \qquad x_2 = m \oplus m_1 \qquad --(2)$$

The output switch first samples x_1 *and then* x_2. The shift register then shifts contents of m_1 *to* $m2$ and contents of m *to* m_1 Next input bit is then taken and stored in m. Again

x_1 *and* x_2 are generated according to this new combination of m, m_1 *and* m_2 (equation (1) and equation (2)). The output switch then samples x_1 *then* x_2. Thus the output bit stream for successive input bits will be,

$$X = x_1x_2x_1x_2x_1x_2 \ldots \ \textit{and so on}$$

Here note that for every input message bit two encoded output bits x_1 *and* x_2 are transmitted. In other words, for a single message bit, the encoded code word is two bits i.e. for this convolutional encoder,

Number of message bits, k. = 1

Number of encoded output bits for one message bit, n = 2

Code Rate of Convolutional Encoder

The code rate of this encoder is,

$$r = \frac{k}{n} = \frac{1}{2}$$

In the encoder of Fig.5.10, observe that whenever a particular message bit enters n shift register, it remains in the shift register for three shifts i.e.,

First shift → Message bit is entered in position 'm'.

Second shift → Message bit is shifted In position $'m_1'$

Third shift → Message bit is shifted in position $'m_2'$

And at the fourth shift the message bit is discarded or simply lost by overwriting. We know that x_1 *and* x_2 are combinations of m_1, m_2, m_3. Since a single message bit remains in m during first shift, in m_1 during second shift and in m_2 during third shift; it influences output x_1 *and* $x_2$for 'three' successive shifts.

Constraint Length (K):

The constraint length of a convolution code is defined as the number of shifts over which a single message bit can influence-the encoder output. It is expressed in terms of message bits. . .' .

For the encoder of Fig. 5.10 constraint length K = 3 bits. This is because in this encoder, a single message bit influences encoder output for three successive shifts. At the fourth shift, the message bit is lost and it has no effect on the output.

Dimension of the Code

The dimension of the code is given by n and k, We know that 'k' is the number of message bits taken at a time by the encoder. And 'n' is the encoded output bits for one message bits. Hence the dimension of the code is (n, k). And such encoder is called. (n, k) convolutional encoder, For example, the encoder of Fig. 5.10 has the dimension of (2, 1).

5.5.2 Time Domain Approach to Analysis of Convolutional Encoder

Let the sequence $\left\{g_0^{(1)}, g_1^{(1)}, g_2^{(1)}, \ldots\ g_m^{(1)}\right\}$, denote the impulse response of the adder which generates x_1 .an F'ig. 5.10 Similarly, Let the sequence $\left\{g_0^{(2)}, g_1^{(2)}, g_2^{(2)}, \ldots\ g_m^{(2)}\right\}$, denote the impulse response of the adder which generates x_2 in Fig. 5.10. These impulse responses are also called **generator sequences** of the code.

Let the incoming message sequence be m_0, m_1, m_2 .... The encoder generates the two output sequences x_1 and x_2 These are obtained by convolving the generator sequences with the message sequence. Hence the name convolutional code is given.

The sequence x_1 is given as,

$$\boxed{x_1 = x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l}} \qquad i = 0,1,2,\ldots \qquad --(1)$$

Here m_{i-l}= 0 for all $l > i$. Similarly the sequence x_2 is given as,

$$\boxed{x_2 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l}} \qquad i = 0,1,2,\ldots \qquad --(2)$$

Note: All additions in above equations are as per mod-2 addition rules.

As shown in the Fig. 5.10. the two sequences x_1 *and* x_2 are multiplexed by the switch. Hence the output sequence is given as,

$$\{x_i\} = \left\{x_0^{(1)} x_0^{(1)} x_1^{(1)} x_1^{(2)} x_2^{(1)} x_2^{(2)} x_3^{(1)} x_3^{(2)} \ldots \ldots.\right\} \qquad --(3)$$

$$here, \qquad v_1 = x_i^{(1)} = \left\{x_0^{(1)} x_1^{(1)} x_2^{(1)} x_3^{(1)} \ldots\right\}$$

$$and \quad v_2 = x_i^{(2)} = \left\{x_0^{(2)} x_1^{(2)} x_2^{(2)} x_3^{(2)} \ldots\right\}$$

Observe that bits from above two sequences are multiplexed in equation (4.4.8). The sequence $\{x_i\}$ is the output of the convolutional encoder.

Example 5.6 : For the convolutional encoder of Fig. 5.11 determine the following:

i. Dimension of tire code
ii. Code rate
iii. Constraint length
iv. Generating sequences (impulse responses)
v. Output sequence for message sequence of m = {1 0 0 1 1}

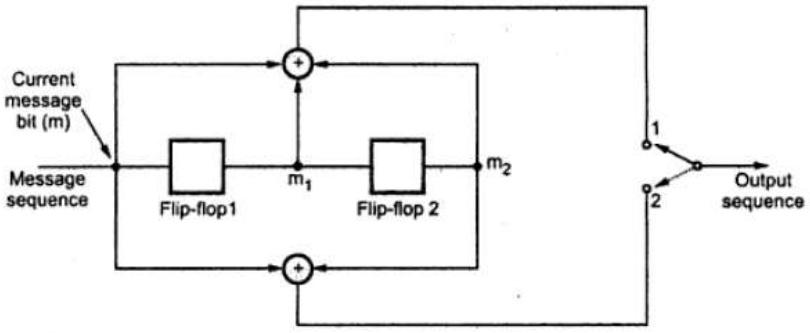


Fig 5.11 Convolutional encoder of example 5.6

Solution:

In the Fig. 5.11 observe that input of flip-flop 1 is the current message bit (m). The output of flip-flop 1 is the previous message bit i.e. m_1. The output of flip flop 2 is previous to previous message bit i.e. m_2. Hence above diagram can be redrawn as shown in Fig. 5.12

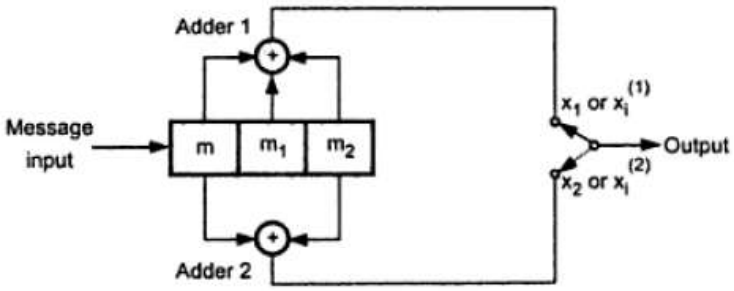


Fig. 5.12 Convolutional encoder of Fig. 5.10 redrawn alternately

Observe that the above encoder is exactly similar to that of Fig. 5.10

i) Dimension of the code

Observe that encoder takes one message bit at a time. Hence k = 1. It generates two bits for every message bit. Hence n = 2. Hence,

Dimension = (n, k) = (2, 1)

ii) Code rate

Code rate is given as,

$$r = \frac{k}{n} = \frac{1}{2}$$

iii) Constraint length

Here note that every message bit, affects output bits for three successive shifts. Hence,

Constraint length K = 3 bits

iv) Generating sequence.

In Fig. 5.12 observe that x_1 i.e. $x_i^{(1)}$ is generated by adding all the three bits. Hence generating sequence $g_{(i)}^1$ is given as

$$g_i^{(1)} = [1\ 1\ 1]$$

$$here, \quad g_0^{(1)} = 1\ represents\ connection\ of\ bit\ m$$

$$g_1^{(1)} = 1\ represents\ connection\ of\ bit\ m_1$$

$$g_2^{(1)} = 1\ represents\ connection\ of\ bit\ m_2$$

x_2 is generated by addition of first and last bits. Hence its generating sequence is given as,

$$g_i^{(2)} = [1\ 0\ 1]$$

$$here, \quad g_0^{(2)} = 1\ represents\ connection\ of\ bit\ m$$

$$g_1^{(2)} = 0\ represents\ that\ m_1\ is\ not\ connected$$

$$g_2^{(2)} = 1\ represents\ connection\ of\ bit\ m_2$$

The above sequences are also called impulse responses.

v) To obtain output sequence

The given message sequence is,

$$m = (m_0\ m_1\ m_2\ m_3\ m_4) = (1\ 0\ 0\ 1\ 1)$$

To obtain output due* to *adder 1

Then from equation (1) we can write,

$$x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)}\, m_{i-l} \qquad --(4)$$

with $i = 0$ above equation becomes,

$$\therefore \quad x_0^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l}$$

$$\therefore \quad x_0^{(1)} = g_0^{(1)} m_0$$

$$= 1 \times 1 = 1, \qquad Here \ g_0^{(1)} = 1 \ and \ m_0 = 1$$

i=1 in eqn (4)

$$x_1^{(1)} = g_0^{(1)} m_1 \oplus g_1^{(1)} m_0$$

$$= (1 \times 0) \oplus (1 \times 1) = 1$$

Here note that additions are mod-2 type.

i=2 in eqn (4)

$$x_2^{(1)} = g_0^{(1)} m_2 \oplus g_1^{(1)} m_1 \oplus g_2^{(1)} m_0$$

$$= (1 \times 0) \oplus (1 \times 0) \oplus (1 \times 1) = 1$$

i=3 in eqn (4)

$$x_3^{(1)} = g_0^{(1)} m_3 \oplus g_1^{(1)} m_2 \oplus g_2^{(1)} m_1$$

$$= (1 \times 1) \oplus (1 \times 0) \oplus (1 \times 0) = 1$$

i =4 in equation (4),

$$x_4^{(1)} = g_0^{(1)} m_4 \oplus g_1^{(1)} m_3 \oplus g_2^{(1)} m_2$$

$$= (1 \times 1) \oplus (1 \times 1) \oplus (1 \times 0) = 0$$

i =5 in equation (4),

$$x_5^{(1)} = g_0^{(1)} m_5 \oplus g_1^{(1)} m_4 \oplus g_2^{(1)} m_3$$

$$= g_1^{(1)} m_4 \oplus g_2^{(1)} m_3 \quad since \ m_5 \ is \ not \ available$$

$$= (1 \times 1) \oplus (1 \times 1) = 0$$

i =6 in equation (4),

$$x_6^{(1)} = g_0^{(1)} m_6 \oplus g_1^{(1)} m_5 \oplus g_2^{(1)} m_4$$

$$= g_2^{(1)} m_4 \quad since \ m_6 \ and \ m_5 \ are \ not \ available$$

$$= (1 \times 1) = 1$$

Thus the output of adder 1 is,

$$\boxed{x_1 = x_i^{(1)} = \{1\ 1\ 1\ 1\ 0\ 0\}}$$

To obtain output due to adder 2

Similarly from equation (2),

$$x_1 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l}$$

$$and \quad m_{i-l} = 0 \quad for\ all\ l > i$$

With i = 0 in above eqn we get,

$$x_0^{(2)} = g_0^{(2)} m_0 = (1 \times 1) = 1 \qquad here\ g_0^{(2)} = 1\ and\ m_0 = 1$$

With i= 1

$$x_1^{(2)} = g_0^{(2)} m_1 \oplus g_1^{(1)} m_0$$

$$= (1 \times 0) \oplus (0 \times 1) = 0$$

With *i* = 2,

$$x_2^{(2)} = g_0^{(2)} m_2 \oplus g_1^{(2)} m_1 \oplus g_2^{(2)} m_0$$

$$= (1 \times 0) \oplus (0 \times 0) \oplus (1 \times 1) = 1$$

With; = 3,

$$x_3^{(2)} = g_0^{(2)} m_3 \oplus g_1^{(2)} m_2 \oplus g_2^{(2)} m_1$$

$$= (1 \times 1) \oplus (0 \times 0) \oplus (1 \times 0) = 1$$

With i=4,

$$x_4^{(2)} = g_0^{(2)} m_4 \oplus g_1^{(2)} m_3 \oplus g_2^{(2)} m_2$$

$$= (1 \times 1) \oplus (0 \times 1) \oplus (1 \times 0) = 1$$

With i=5,

$$x_5^{(2)} = g_1^{(2)} m_4 \oplus g_2^{(2)} m_3$$

$$= (0 \times 1) \oplus (1 \times 1) = 1$$

With i=6,

$$x_6^{(2)} = g_2^{(2)} m_4$$

$$= 1 \times 1 = 1$$

Thus the sequence x_2 is,

$$x_2 = x_i^{(2)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\}$$

To obtain multiplexed sequence of x_1 and x_2 as per equation (3)

The two sequences x_1 and x_2 are multiplexed to get the final output i.e.

$$x_i = x_0^{(1)} x_0^{(2)} x_1^{(1)} x_1^{(2)} x_2^{(1)} x_2^{(2)} x_3^{(1)} x_3^{(2)} x_4^{(1)} x_4^{(2)} x_5^{(1)} x_5^{(2)} x_6^{(1)} x_6^{(2)}$$

$$= \{1\ 1, 1\ 0, 1\ 1, 1\ 1, 0\ 1, 0\ 1, 1\ 1\}$$

5.5.3 Transform Domain Approach to Analysis of Convolutional Encoder

In the previous section we observed that the convolution of generating sequence and message sequence takes place. These calculations can be simplified by applying the transformations to the sequences. Let the impulse responses be represented by polynomials. i.e.

$$g^{(1)}(p) = g_0^{(1)} + g_1^{(1)} p + g_2^{(1)} p^2 + \cdots + g_M^{(1)} p^M$$

Similarly,

$$g^{(2)}(p) = g_0^{(2)} + g_1^{(2)} p + g_2^{(2)} p^2 + \cdots + g_M^{(2)} p^M$$

Thus the polynomials can be written for other generating sequences. The variable 'p' is unit delay operator in above equations. It represents the time delay of the bits in impulse response.

Simtlarly we can write the polynomial (or message polynomial i.e.

$$m(p) = m_0 + m_1 p + m_2 p^2 + \cdots + m_{L-1} p^{L-1}$$

Here L is the length of the message sequence. The convolution sums are converted to polynomial multiplications in the transform domain. i.e.,

$$\boxed{\begin{matrix} x^{(1)}(p) = g^{(1)}(p).m(p) \\ x^{(2)}(p) = g^{(2)}(p).m(p) \end{matrix}} \qquad --(5)$$

The above equations are the output polynomials of sequences $x_i^{(1)}$ *and* $x_i^{(2)}$.

Note : All additions in above equations are as per mod 2 addition rules.

Example 5.7: Repeat part (V) of example 5.6 using transform domain calculations (polynomial multiplications).

Solution:

a) To obtain generating polynomial for adder-1 :

The first generating sequence is given by,

$$g_i^{(1)} = \{1\ 1\ 1\}$$

Hence its polynomial can be obtained as follows:

$$g^{(1)}(p) = 1 + 1 \times p + 1 \times p^2$$

$$= 1 + p + p^2$$

b) To obtain generating polynomial for adder-2

The second generating sequence is given by,

$$g_i^{(2)} = \{1\ 0\ 1\}$$

Hence its polynomial can be obtained as follows:

$$g^{(2)}(p) = 1 + 0 \times p + 1 \times p^2$$

$$= 1 + p^2$$

c) To obtain message polynomial

The message sequence is,

$$m = (1\ 0\ 0\ 1\ 1)$$

Hence its polynomial can be obtained as,

$$m(p) = 1 + 0 \times p + 0 \times p^2 + 1 \times p^3 + 1 \times p^4$$

$$= 1 + p^3 + p^4$$

d) To determine the output due to adder-1

Now $x^{(1)}(p)$ can be obtained from equation (5). i.e.

$$x^{(1)}(p) = g^{(1)}(p).m(p)$$

$$= (1 + p + p^2)(1 + p^3 + p^4)$$

$$= 1 + p + p^2 + p^3 + p^6$$

The above polynomial can also be written as,

$$x^{(1)}(p) = 1 + (1 \times p) + (1 \times p^2) + (1 \times p^3) + (0 \times p^4) + (0 \times p^5) + (1 \times p^6)$$

Thus the output sequence $x_i^{(1)}$ is,

$$x_i^{(1)} = \{1\ 1\ 1\ 1\ 0\ 0\ 1\}$$

e) To determine the output due to adder·2

Similarly polynomial ,$x^{(2)}(p)$ can be obtained as,

$$x^{(2)}(p) = g^{(2)}(p).m(p)$$

$$= (1 + p^2)(1 + p^3 + p^4)$$

$$= 1 + p^2 + p^3 + p^4 + p^5 + p^6$$

Thus the output sequence $x_i^{(2)}(p)$ is,

$$x_i^{(2)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\}$$

f) To determine the multiplexed output sequence

The multiplexed output sequence will be as follows:

$$\{x_i\} = \{1\ 1, 1\ 0, 1\ 1, 1\ 1, 0\ 1, 0\ 1, 1\ 1\}$$

Here note that very few calculations are involved in transform domain.

5.5.4 Code Tree, Trellis and State Diagram for a Convolution Encoder

Now let's study the operation of the convolutional encoder with the help of code tree, trellis and state diagram. Consider again the convolutional encoder of Fig. 5.10. It is reproduced below,

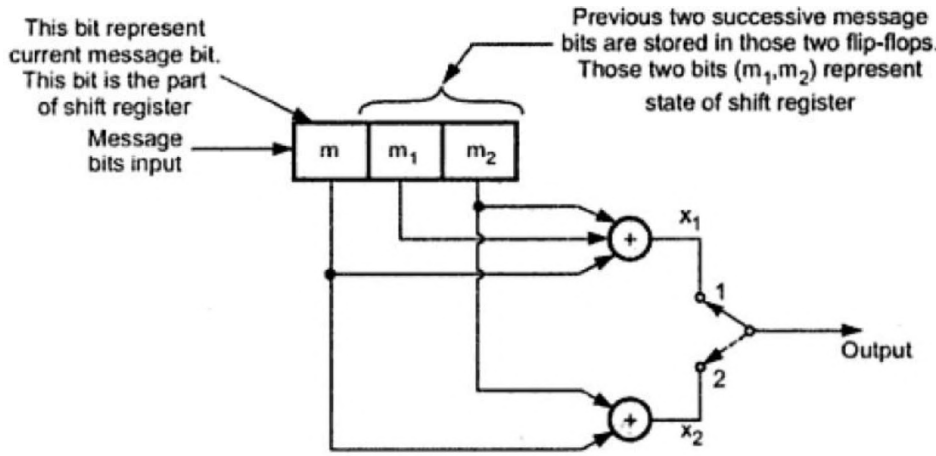


Fig. 5.13 Convolutional encoder with k = 1 and n =2

5.5.4.1 States of the Encoder

In Fig 5.13, the previous two successive bits m_1 *and* m_2 represents state. The input message bit m affects the 'state' of the encoder as well as outputs x_1 *and* x_2 during that state. Whenever new message bit is shifted to *'m'*, the contents of m_1 *and* m_2 define new state. And outputs x_1 *and* x_2 are also changed according to new state m_1, m_2 and message bit m. Let's define these states as shown in Table 5.6

Let the initial values of bits stored in $m_1 and\ m_2$ be zero. That is $m_1 m_2 = 00$ initially and the encoder is in state 'a'.

m_2	m_1	State of encoder
0	0	a
0	1	b
1	0	c
1	1	d

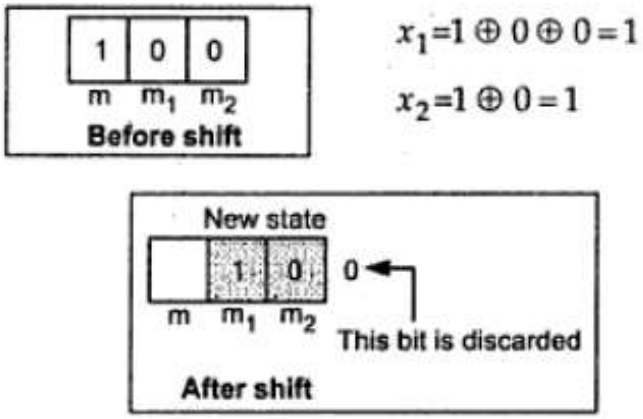
Table 5.6 States of the encoder of Fig. 5.13.

5.5.4.2 Development of the Code Tree

Let us consider the development of code free for the message sequence m = 110. Assume that $m_1 m_2 = 00$ initially.

1) When $m_1 = 1$ i.e. first bit

The first message input is m = 1. With this input x_1 and x_2 will be calculated as follows.



The values of $x_1 x_2 = 11$ are transmitted to the output and register contents are shifted to right by one bit position as shown.

Thus the new state of encoder is $m_2 m_1 = 01$ or *'b'* and output transmitted are $x_1 x_2 = 11$. This shows that if encoder is in state 'a' and if input is m = 1, then the next state is *'b'* and outputs are $x_1 x_2 = 11$. The first row of Table 5.7 illustrates this operation.

The last column of this table shows the code tree diagram. The code tree diagram starts at node or state *'a'*. The diagram is shown in Fig. 5.14.

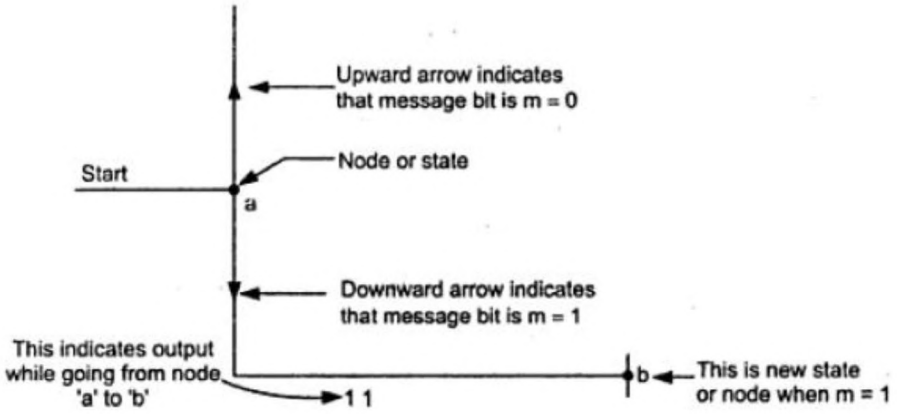
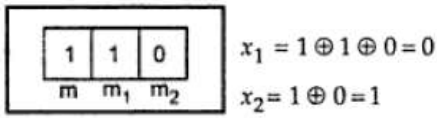


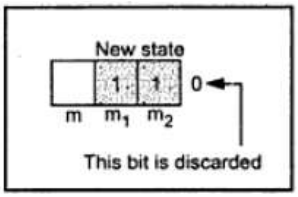
Fig. 5.14 Code tree from node 'a' to *'b'*

Observe that if $m = 1$ we go downward from node 'a'. Otherwise if $m = 0$, we go upward from node *'a'*. It can be verified that if $m = 0$ then next node (state) is 'a' only. Since $m = 1$ here we go downwards toward node *b* and output is 11 in this node (or state),

2) When m = 1 i.e. second bit

Now let the second message bit be 1. The contents of shift register with this input will be as shown below





These values of $x_1x_2 = 01$ are then transmitted to output and register contents are shifted to right by one bit. The next state formed is as shown.

Thus the new state of the encoder is $m_2m_1 = 11$ or *'d'* and the outputs transmitted are $x_1x_2 = 01$. Thus the encoder goes from state 'b' to state *'d'* if input is '1' and transmitted output $x_1x_2 = 01$. This operation is illustrated by Table 5.7 in second row. The last column of the table shows the code tree for those first and second input bits,

3) When m = 0, i.e. 3rd bit

Similarly *3rd* row of the Table 5.7 illustrated the operation of encoder for 3rd input message bit as m = 0. Now observe in the code tree of last column. Since input bit is m = 0, the path of the tree is shown by upward arrow towards node (or state) 'c'. That is the next state is 'c' (i.e, 10) and output is $x_1x_2 = 01$.

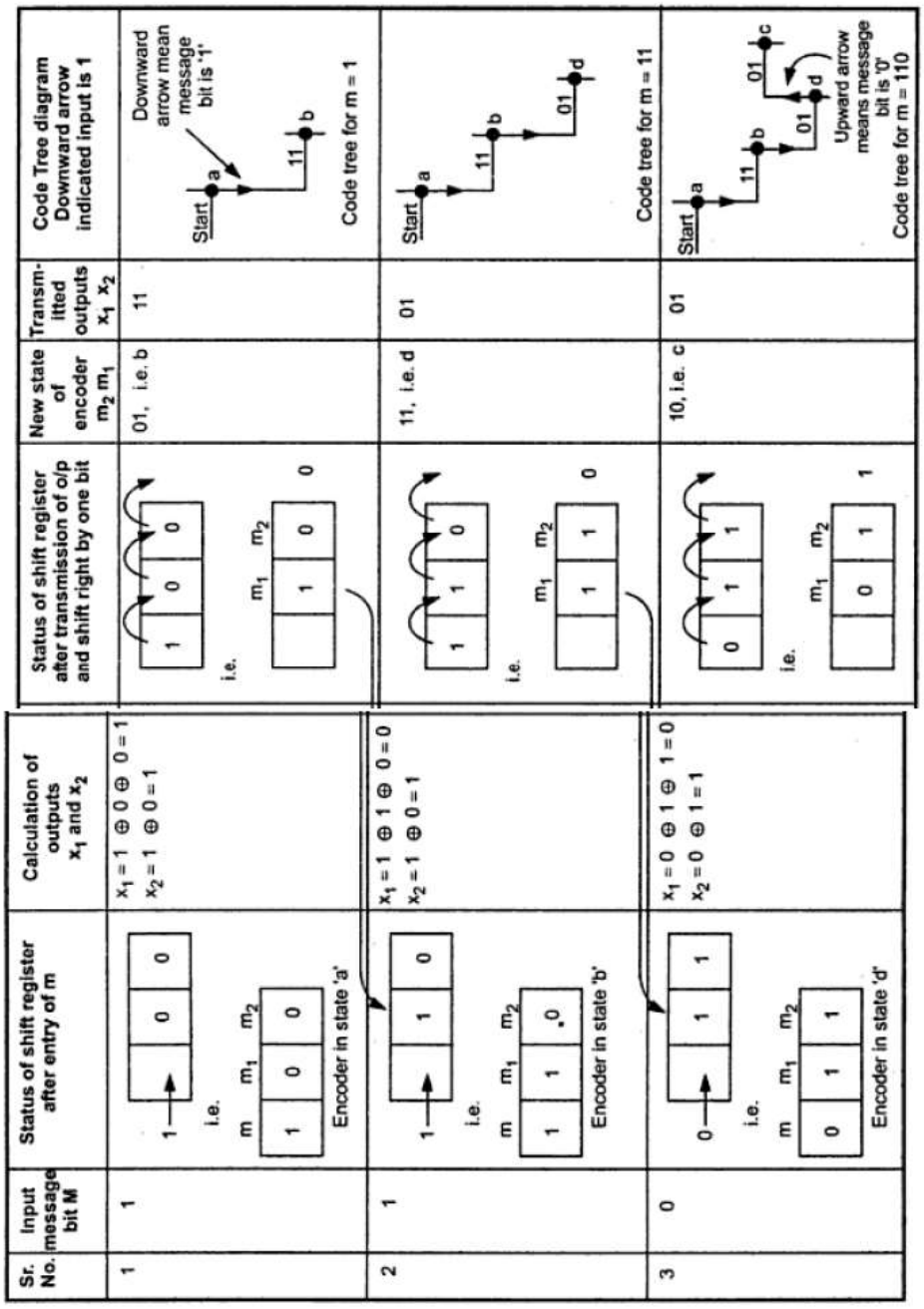


Table 5.7 Analysis of convolutional encoder of Fig 5.13

Complete code tree for convolutional encoder

Fig. 5.13 shows the code tree for this encoder. The code tree starts at node 'a'. If input message bit is '1' then path of the tree goes down towards node 'b' and output is 11. Otherwise if the input

is m = 0 at node 'a', then path of the tree goes upward towards node 'a' and output is 00. Similarly, depending upon the input message bit, the path of the tree goes upward or downward. The nodes are marked with their states a, b, c or d. On the path between two nodes the outputs are shown. We have verified the part of this code tree for first three message bits as 110.

In the code tree of Fig. 5.15, we find that the branch pattern begins to repeat after third bit. This is shown in figure. The repetition starts after 3rd bit, since particular message bit is stored in the shift registers of the encoder for three shifts. If the length of the shift register is increased by one bit, then the pattern of code tree will repeat after fourth message bit.

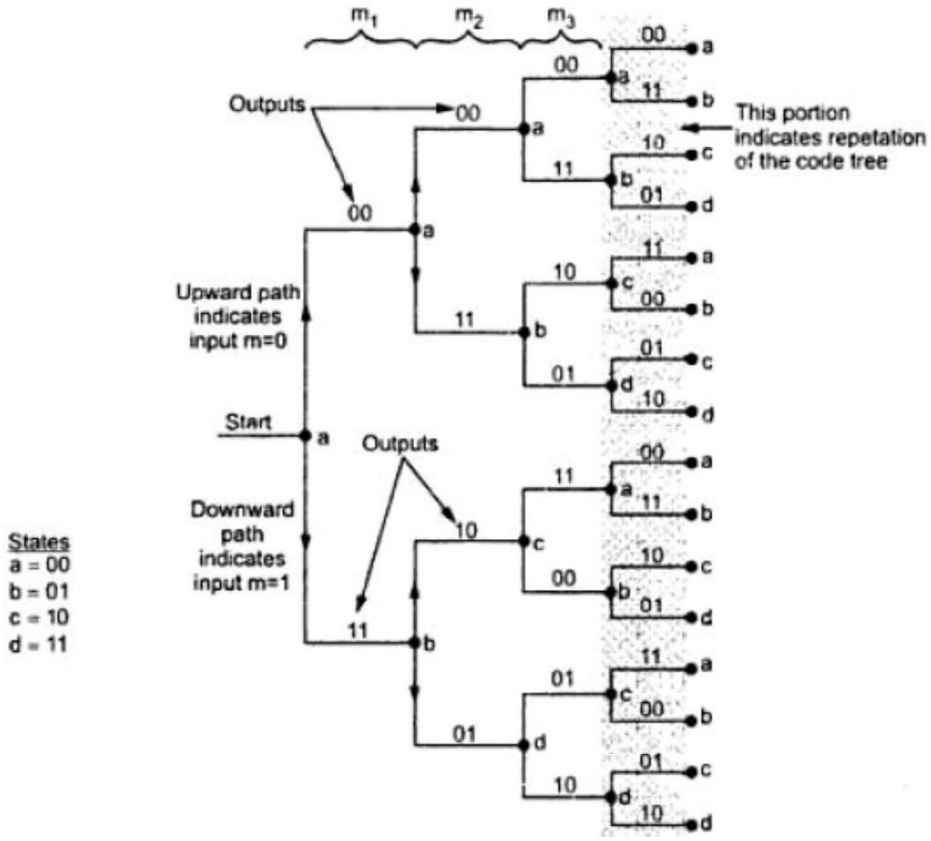


Fig. 5.15 Code tree for convolutional encoder of Fig 5.10

5.5.4.3 Code Trellis (Represents Steady State Transitions)

Code trellis is the more compact representation of the code tree. We know that in the code tree there are four states (or nodes). Every state goes to some other state depending upon the input code. Trellis represents the single and unique diagram for such transitions. Fig. 5.16 shows code trellis diagram.

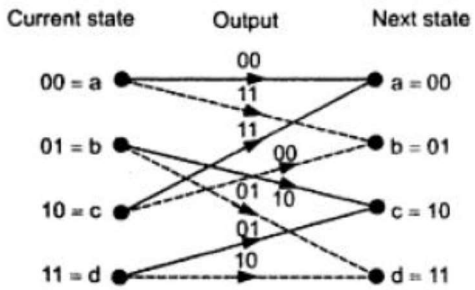


Fig. 5.16 Code trellis of convolutional encoder of Fig 5.10

The nodes on the left denote four possible current states and those on the right represent next state. The solid transition line represents input m = 0 and broken tine represents input $m = 1$. Along with each transition line the output x_1x_2 is represented during that transition. For example let the encoder be in current state of 'a'. If input m = 0, then next state will be 'a' with outputs $x_1x_2 = 11$. Thus code trellis is the compact representation of code tree.

5.5.4.4 State Diagram

If we combine the current and next states, then we obtain state diagram

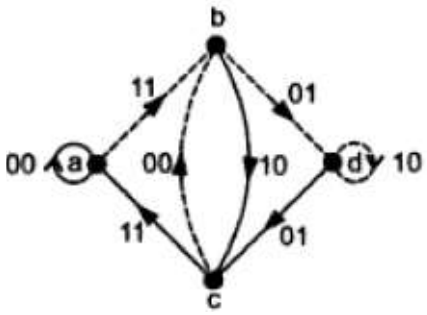


Fig. 5.17 State diagram for convolutional encoder of Fig 5.10

For example consider that the encoder is in state *'a'*. If input m = 0, then next state is same i.e. a (i.e. 00) with outputs x_1x_2= 00. This is shown by self loop at node 'a' in the state diagram. If input $m = 1$, then state diagram shows that next state is *'b'* with outputs x_1x_2= 11 .

Comparison between code tree and trellis diagram :

Table 5.8 shows the comparison between code tree and trellis diagram as a graphic structure to generate and decode convolutional code.

Sr. No.	Code tree	Trellis diagram
1	Code tree indicates flow of the coded signal along the nodes of the tree.	Trellis diagram indicates transitions from current to next states.
2	Code tree is lengthy way of representing coding process.	Code trellis diagram is shorter or compact way of representing coding process.
3	Decoding is very simple using code tree.	Decoding is little complex using trellis diagram.
4	Code tree repeats after number of stages used in the encoder.	Trellis diagram repeats in every state. In steady state, trellis diagram has only stage.
5	Code tree is complex to implement in programming.	Trellis diagram is simpler to implement in programming.

Table 5.8 Comparison between code tree and trellis diagram

5.5.5 Decoding Methods of Convolutional Codes

These methods are used for decoding of convolutional codes. They are viterbi algorithm, sequential decoding and feedback decoding.

5.5.5.1Viterbi Algorithm for Decoding of Convolutional Codes (Maximum Likelihood Decoding)

Let's represent the received signal by Y. Convolutional encoding operates continuously on input data. Hence there are no code vectors and blocks. Let's assume that the transmission error probability of symbols 1's and 0's is same. Let's define an integer variable metric as follows.

Metric:

It is the discrepancy between the received signal Y and the decoded signal at particular node. This metric can be added over few nodes for a particular path.

Surviving Path:

This is the path of the decoded signal with minimum metric.

In viterbi decoding a metric is assigned to each surviving path. (Metric of a particular path is obtained by adding individual metric on the nodes along that path). Y is decoded as the surviving path with smallest metric.

Consider the following example of viterbi decoding. Let the signal being received is encoded by the encoder of Fig. 5.10. Let the first six received bits be

$$Y = 11\ 01\ 11$$

***a) Decoding of first message bit for Y* = 11**

Note that for single bit input the encoder transmits two bits $(x_1 x_2)$ outputs. These outputs are received at the decoder and represented by *Y*. Thus *Y* given above represents the outputs for

three successive message bits. Assume that the decoder is at state a_0. Now look at the code trellis diagram of Fig. 5.16 for this encoder. It shows that if the current state is 'a', then next state will be 'a' or 'b'. This is shown in Fig. 5.18. Two branches are shown from a_0. One branch is at next node a_1 representing decoded signal as 00 and other branch is at b_1 representing decoded signal as 11.

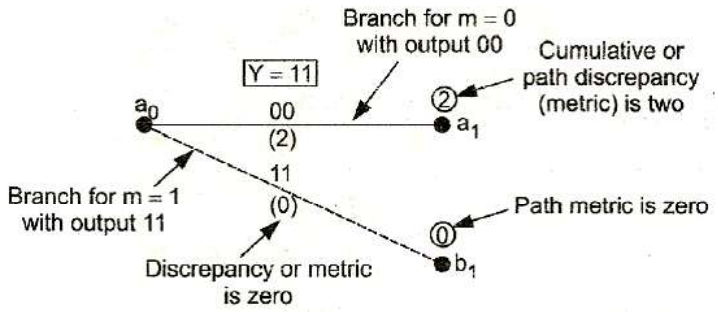


Fig. 5.18 Viterbi decoder results for first message bit

The branch from $a_0 b_1$ to represents decoded output as 11 which, is same as received signal at that node i.e. 11. Thus there is no discrepancy between received signal and decoded signal. Hence 'Metric' of that branch is zero. This metric is shown in brackets along that branch. The metric of branch from a_0 to a_1 is two. The encoded number near a node shows path metric reaching to the node.

***b) Decoding of second message bit for Y* = 01**

When the next part of bits $Y = 01$ is received at nodes a_1 and b_1, then from nodes a_1 and b_1 four possible next states a_2, b_2, c_2 *and* d_2 are possible. Fig. 5.19 shows all these branches, their decoded outputs and branch metrics corresponding to those decoded outputs. The encircled number near a_2, b_2, c_2 *and* d_2 indicate path metric emerging from a_0. For example the path metric of path a_0, a_1, a_2 is 'three'. The path metric of path $a_0 b_1 d_2$ is zero

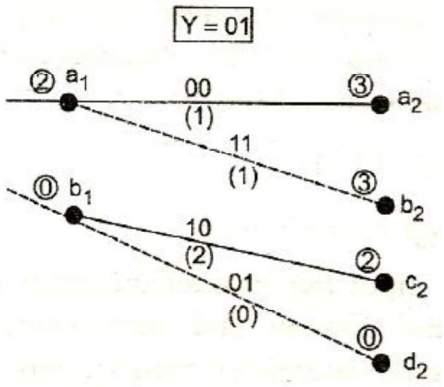


Fig. 5.19 Viterbi decoder results for second message bit

c) *Decoding of 3rd message bit for* Y = 11

Fig. 5.20 shows the trellis diagram for all the six bits of Y.

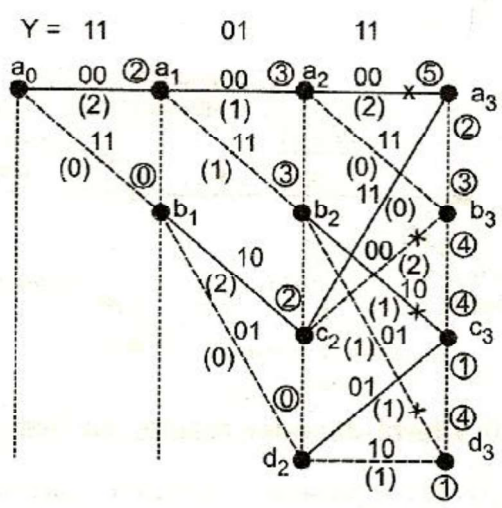


Fig. 5.20 Paths and their metrics for viterbi decoding

Fig. 5.20 shows the nodes with their path metrics on the right hand side at the end of sixth bit of Y. Thus two paths are common to node *'a'*. One path is $a_0a_1a_2a_3$ with metric 5. The other path is $a_0b_1c_2a_3$ with metric 2. Similarly there are two paths at other nodes also. According to viterbi decoding, only one path with lower metric should be retained at particular node. The paths marked with x (cross) are cancelled because they have higher metrics than other path coming to that particular node. These four paths with lower metrics are stored in the decoder and the decoding continues to next received bits.

d) Further explanation of viterbi decoding for* 12 *message bits

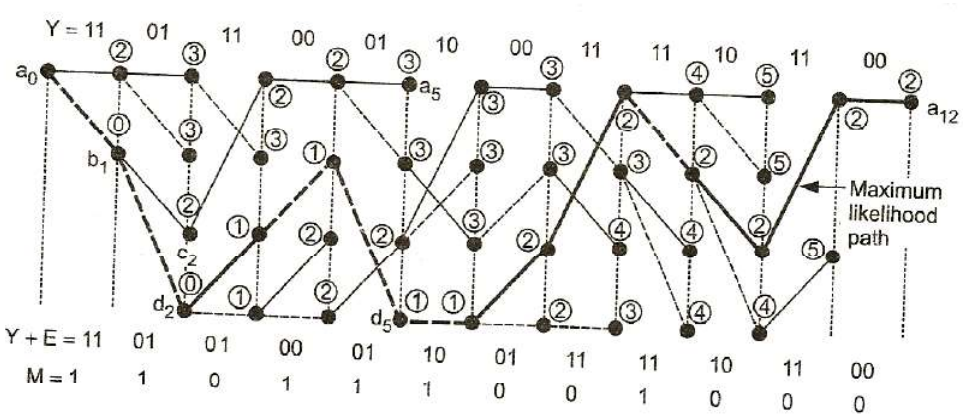


Fig. 5.21 Viterbi decoding

Fig. 5.21 shows the continuation of Fig. 5.20 for a message 12 bits. Observe that in this figure, received bits Y are marked at the top of the decoded value of output i.e. Y+E is marked at the bottom and decoded message signal is also marked

Only one path of particular node is kept which is having lower metric. In case if there are two paths having same metric, then anyone of them is continued. Observe that a node $'a_{12}'$ only one path arrives with metric two. This path is shown by a thick line. Since this path is lowest metric it is the surviving path and hence Y is decoded from this path. All the decoded values of output are taken from the outputs of this path. Whenever this path is broken it shows message bit m = 1 and if it is continuous, message bit m =0 between two nodes.

The method of decoding used in viterbi decoding is called **maximum likelihood decoding**.

e) Surviving paths

During decoding we find that a viterbi decoder has to store four surviving paths for four nodes.

$$Surviving\ paths\ =\ 2^{(K-1)k}$$

Here K is constraint length and k is number of message bits.

For the encoder for Fig. 5.10 K = 3 and k = 1

$$\therefore Surviving\ paths\ =\ 2^{(3-1)\,x\,1}\ =\ 4$$

Thus the viterbi decoder has to store four surviving paths always. If the number of message bits to be decoded are very large, then storage requirement is also large since the decoder has to store multiple paths. To avoid this problem metric diversion effect is used.

f) Metric Diversion Effect:

For the two surviving paths originating from the same node, the running metric of less likely path tends to increase more rapidly than the metric of other path within about $5(k-1)$ branches from the common node. This is called **metric divergence effect**. For example consider the two paths coming from node b_1. One path comes at a_5 and other path comes at d_5. The path at a_5 is less likely and hence its metric is more compared to the path' at d_5. Hence at node d_5 only the surviver path is selected and the message bits are decoded. The fresh paths are started from d_5. Because of this, the memory storage is reduced since complete path need not be stored.

Example 5.8: For the convolutional encoder arrangement shown in Fig. 5.22, draw the state diagram and hence trellis diagram. Determine output digit sequence for the data digits 1 1 0 1 0 1 0 0. What are the dimensions of the code (n, k) and constraint length ? Use viterbi's algorithm to decode the sequence, 100 110 111 101 001 101 001 010.

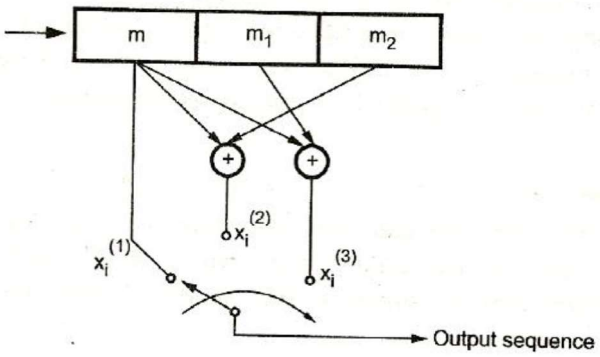


Fig. 5.22 Encoder of example 3.4.6

Solution:

i) To obtain dimension of the code:

Observe that one message bit is taken at a time in the encoder of Fig. 5.22. Hence $k = 1$. There are three output bits for every message bit. Hence $n = 3$. Therefore the dimension of the code is $(n, k) = (3, 1)$.

ii) Constraint length

Here note that every message bit affects three output bits. Hence,

$$Constraint\ length\ K\ =\ 3\ bits.$$

iii) To obtain code trellis and state diagram

Let the states of the encoder be as given in Table 5.6. Fig. 5.23 shows the code trellis of the given encoder

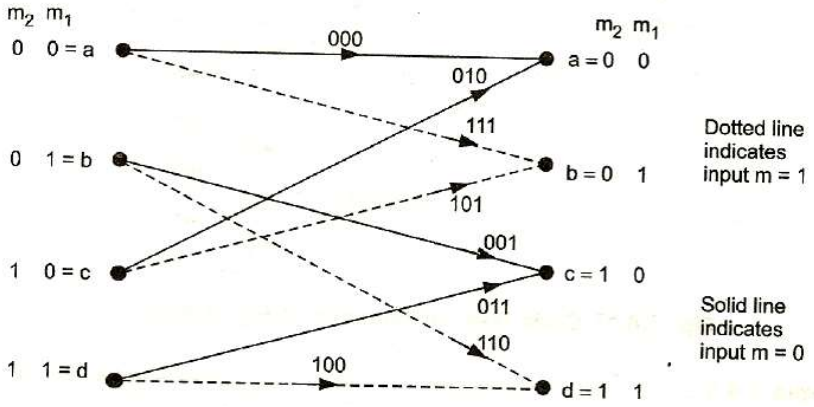


Fig. 5.23 Code trellis of encoder of Fig. 3.4.18

The nodes in the code trellis can be combined to form state diagram as shown in Fig. 5.24

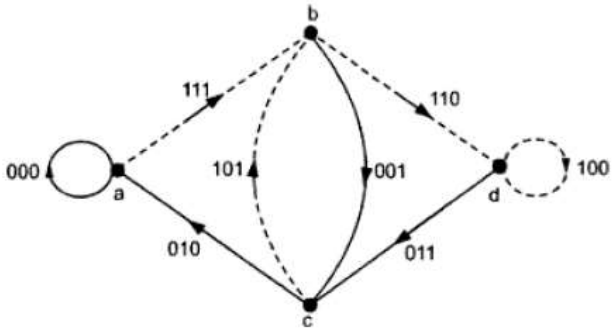


Fig. 5.24 State diagram of the encoder of Fig. 4.4.17

iv)To determine output sequence

a) Determine generator polynomials

The generating sequence can be written for $x_i^{(1)}$ from Fig. 5.22 as,

$$g_i^{(1)} = \{1, 0, 0\} \qquad since\ only\ m\ is\ connected$$

Similarly generating sequence for $x_i^{(2)}$ will be,

$$g_i^{(2)} = \{1, 0, 1\} \qquad since\ \ m\ and\ m_2\ are\ connected$$

And generating sequence for , $x_i^{(3)}$ will be,

$$g_i^{(3)} = \{1, 1, 0\} \qquad since\ \ m\ and\ m_1\ are\ connected$$

Hence the corresponding generating polynomials can be written as,

$$g^{(1)}(p) = 1$$

$$g^{(2)}(p) = 1 + p^2$$

$$g^{(3)}(p) = 1 + p$$

b) Determine message polynomials

The given message sequence is,

$$m = \{1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\}$$

Hence the message polynomial will be,

$$m(p) = 1 + p + p^3 + p^5$$

c) Obtain output for $\boldsymbol{g_i^{(1)}}$

The first sequence $x_i^{(1)}$ is given as,

$$x_i^{(1)} = g^{(1)}(p).m(p)$$

$$= 1(1 + p + p^3 + p^5)$$

$$= 1 + p + p^3 + p^5$$

Hence, the corresponding sequence will be,

$$\left\{x_i^{(1)}\right\} = \{1\ 1\ 0\ 1\ 0\ 1\}$$

d) ***Obtain output for*** $\boldsymbol{g_i^{(2)}}$

The first sequence $x_i^{(2)}$ is given as,

$$x_i^{(2)} = g^{(2)}(p).m(p)$$

$$= (1 + p^2)(1 + p + p^3 + p^5)$$

$$= 1 + p + p^2 + p^7$$

Hence the corresponding sequence will be,

$$x_i^{(2)} = \{1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\}$$

e) Obtain output for $\boldsymbol{g_i^{(3)}}$

The third sequence $x_i^{(3)}$ is given as,

$$x_i^{(3)} = g^{(3)}(p).m(p)$$

$$= (1 + p)(1 + p + p^3 + p^5)$$

$$= 1 + p^2 + p^3 + p^4 + p^5 + p^6$$

Hence the corresponding sequence is,

$$x_i^{(3)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\}$$

f) To multiplex three output sequences

The three sequences $x_i^{(1)}, x_i^{(2)} and\ x_i^{(3)}$ are made equal in length i.e. 8 bits. Hence zeros are appended in sequence $x_i^{(1)}\ and\ x_i^{(2)}$. These sequences are shown below:

$$x_i^{(1)} = \{1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\}$$

$$x_i^{(2)} = \{1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\}$$

$$x_i^{(3)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\}$$

The bits from above three sequences are multiplexed to give the output sequence i.e.

$$\{x_i\} = \{111\ \ 110\ \ 011\ \ 101\ \ 001\ \ 101\ \ 001\ \ 010\}$$

v) Viterbi algorithm to decode the given sequence

Fig. 5.25 (next page) shows the diagram based on viterbi decoding. It shows received sequence at the top. The decoded (Y + E) sequence and decoded message sequence is shown at the bottom.

The dark line shows maximum likelihood path. It has the lowest running metric, i.e. 3. At any point only four paths are retained. The decoded message sequence is,

$$m = \{110\ 1\ 0\ 100\}$$

5.5.6 Advantages and Disadvantages of Convolutional Codes

Convolutional codes can be designed to detect or correct the errors. Some convolutional codes available which are used to correct random errors and bursts. Convolutional codes have advantages over block codes

Advantages:

1. The decoding delay is small in convolutional codes since they operate on smaller blocks of data
2. The storage hardware required by convolutional decoder is less since the block sizes are smaller
3. Synchronization problem does not affect the performance of convolutional codes

Disadvantages:

1. Convolutional codes are difficult to analyze since their analysis is complex
2. Convolutional codes are not developed much as compared to block codes

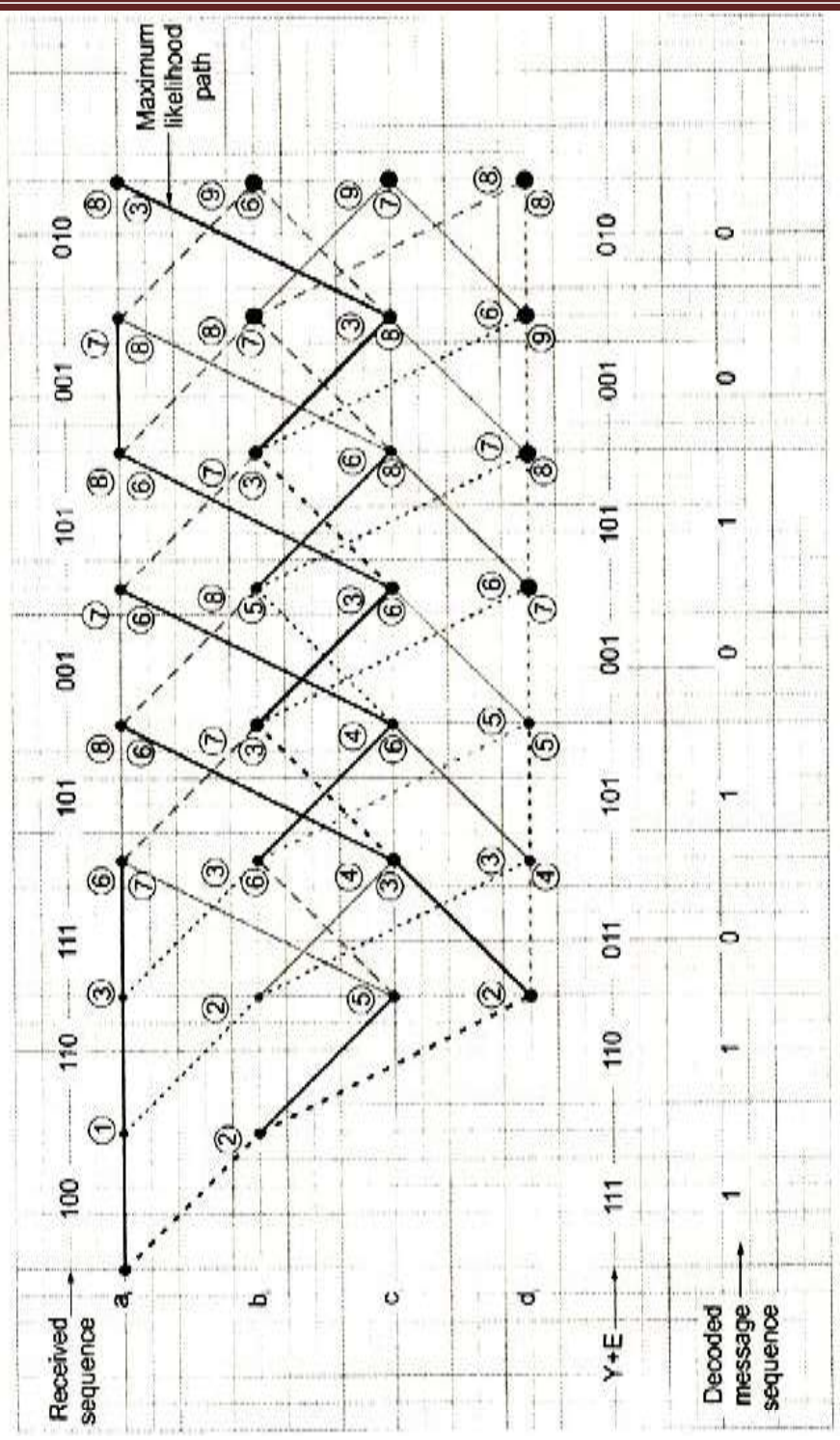


Fig. 5.25 Viterbi decoding for example 5.8

5.5.7 Comparison between Linear Block Codes and Convolutional Codes

The convolutional codes and block codes can be compared on the basis of their encoding methods, decoding methods, error correcting capabilities, complexity etc points.

Sr. No.	Linear block codes	Convolutional codes
1	Block codes are generated by, $X = MG$ or $X(p) = M(p)G(p)$	Convolutional codes are generated by convolution between message sequence and generating sequence. i.e. $x_i = \sum_{l=0}^{M} g_l m_{i-l}, i = 0, 1, 2, \ldots$
2	For a block of message bits, encoded block (code vector) is generated.	Each message bit is encoded separately. For every message bit, two or more encoded bits are generated.
3	Coding is block by block.	Coding is bit by bit.
4	Syndrome decoding is used for most likelihood decoding.	Viterbi decoding is used for most likelihood decoding.
5	Generator matrices, parity check matrices and syndrome vectors are used for analysis.	Code tree, code trellis and state diagrams are used for analysis.
6	Distance properties of the code can be studied from code vectors.	Distance properties of the code can be studied from transfer function.
7	Generating polynomial and generator matrix are used to get code vectors.	Generating sequences are used to get code vectors.
8	Error correction and detection capability depends upon minimum distance d_{min}.	Error correction and detection capability depends upon free distance d_{free}.

Table 5.9 Comparison of linear block codes and convolutional codes

2 MARKS

1. What is hamming distance ?

The hamming distance between the two code vectors is equal to the number of elements in which they differ. For example, let the two codewords be,

$$X = (101) \; and \; Y = (110)$$

These two codewords differ in second and third bits. Therefore the hamming distance between X and Y is two.

2. Define code efficiency?

The code efficiency is the ratio of message bits in a block to the transmitted bits for that block by the encoder i.e.,

$$code\ efficiency = \frac{message\ bits}{Transmitted\ bits} = \frac{k}{n}$$

3. What is meant by systematic and nonsystematic codes?

In a systematic block code, message bits appear first and then check bits. In the nonsystematic code, message and check bits cannot be identified in the code vector.