# UNIT IV
# Linear Block Codes

## Introduction

Coding theory is concerned with the transmission of data across noisy channels and the recovery of corrupted messages. It has found widespread applications in electrical engineering, digital communication, mathematics and computer science. 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. Errors are introduced in the data when it passes through the channel. The channel noise interferes the signal. The signal power is reduced. 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.

## 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.Figure illustrates this concept.

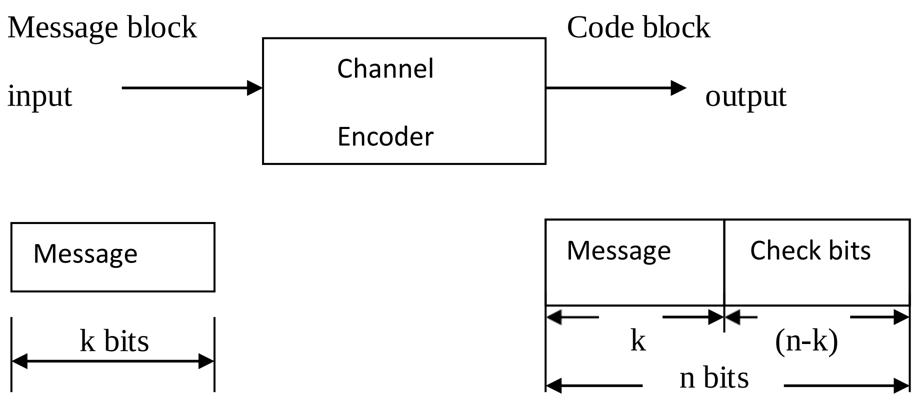



Figure: Functional block diagram of block coder

**Types are**

**Systematic codes:**

In the systematic block code, the message bits appear at the beginning of the code word. The message appears first and then check bits are transmitted in a block. This type of code is called systematic code.

**Nonsystematic codes:**

In the nonsystematic block code it is not possible to identify the message bits and check bits. They are mixed in the block.

Consider the binary codes and all the transmitted digits are binary.

**Linear Block Codes**

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 $m_1, m_2, m_3, \ldots m_k$ message bits and $c_1, c_2, c_3 \ldots c_q$ check bits. Then this code vector can be written as,

$$X=(m_1,m_2,m_3,\ldots m_k c_1,c_2,c_3\ldots c_q)$$

Here q=n-k

Whereq are the number of redundant bits added by the encoder.

Code vector can also be written as

$$X=(M/C)$$

Where M= k-bit message vector

C= q-bit check vector

The main aim of linear block code is to generate check bits and this check bits are mainly used for error detection and correction.

Example :

The (7, 4) linear code has the following matrix as a generator matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

If u = (1 1 0 1) is the message to be encoded, its corresponding code word would be

$$\begin{aligned} \mathbf{v} &= 1\cdot\mathbf{g}_0 + 1\cdot\mathbf{g}_1 + 0\cdot\mathbf{g}_2 + 1\cdot\mathbf{g}_3 \\ &= (1\,1\,0\,1\,0\,0\,0) + (0\,1\,1\,0\,1\,0\,0) + (1\,0\,1\,0\,0\,0\,1) \\ &= (0\,0\,0\,1\,1\,0\,1) \end{aligned}$$

A linear systematic (n, k) code is completely specified by ak × n matrix G of the following form

$$
\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} = \left[\begin{array}{ccccccc|ccccccc} p_{00} & p_{01} & \cdot & \cdot & \cdot & p_{0,n-k-1} & & 1 & 0 & 0 & . & . & . & 0 \\ p_{10} & p_{11} & \cdot & \cdot & \cdot & p_{1,n-k-1} & & 0 & 1 & 0 & . & . & . & 0 \\ p_{20} & p_{21} & \cdot & \cdot & \cdot & p_{2,n-k-1} & & 0 & 0 & 1 & . & . & . & 0 \\ & & & & & & & & & & & & & \\ & & & & & & & & & & & & & \\ & & & & & & & & & & & & & \\ p_{k-1,0} & p_{k-1,1} & \cdot & \cdot & \cdot & p_{k-1,n-k-1} & & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]
$$

P matrix | k × k identity matrix

where $p_{ij}$ = 0 or 1

Let u = ($u_0$, $u_1$, … , $u_{k-1}$) be the message to be encoded.The corresponding code word is

$$
\begin{aligned} v &= (v_0, v_1, v_2, \ldots, v_{n-1}) \\ &= (u_0, u_1, \ldots, u_{k\text{-}1}) \cdot G \end{aligned}
$$

The components of **v** are

$$
v_{n\text{-}k+i} = u_i \qquad \text{for } 0 \le i < k
$$

$$
v_j = u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k\text{-}1} p_{k\text{-}1,j} \quad \text{for } 0 \le j < n\text{-}k
$$

The n – k equations given by above equation are called parity-check equations of the code

**Example for Codeword**

The matrix **G** given by

$$
\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}
$$

Let **u** = ($u_0$, $u_1$, $u_2$, $u_3$) be the message to be encoded and **v** = ($v_0$, $v_1$, $v_2$, $v_3$, $v_4$, $v_5$,$v_6$) be the corresponding code word

Solution :

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (u_0, u_1, u_2, u_3) \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

By matrix multiplication, the digits of the code word v can be determined.

$$\begin{aligned} v_6 &= u_3 \\ v_5 &= u_2 \\ v_4 &= u_1 \\ v_3 &= u_0 \\ v_2 &= u_1 + u_2 + u_3 \\ v_1 &= u_0 + u_1 + u_2 \\ v_0 &= u_0 + u_2 + u_3 \end{aligned}$$

The code word corresponding to the message (1 0 1 1) is (1 0 0 1 0 1 1)

If the generator matrix of an (*n*, *k*) linear code is in systematic form, the parity-check matrix may take the following form

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

$$= \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 & p_{00} & p_{10} & . & . & . & p_{k-1,0} \\ 0 & 1 & 0 & . & . & . & 0 & p_{01} & p_{11} & . & . & . & p_{k-1,1} \\ 0 & 0 & 1 & . & . & . & 0 & p_{02} & p_{12} & . & . & . & p_{k-1,2} \\ . & & & & & & & & & & & & \\ . & & & & & & & & & & & & \\ . & & & & & & & & & & & & \\ 0 & 0 & 0 & . & . & . & 1 & p_{0,n-k-1} & p_{1,n-k-1} & . & . & . & p_{k-1,n-k-1} \end{bmatrix}$$

Encoding circuit for a linear systematic (n,k) code is shown below.

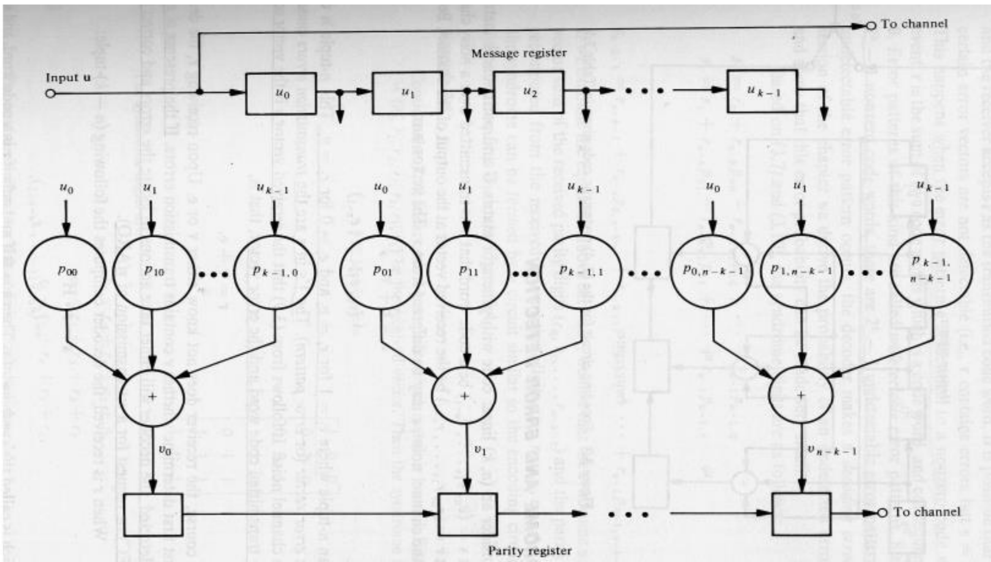



Figure: Encoding Circuit

For the block of k=4 message bits, (n-k) parity bits or check bits are added. Hence the total bits at the output of channel encoder are n=7. The encoding circuit for (7, 4) systematic code is shown below.

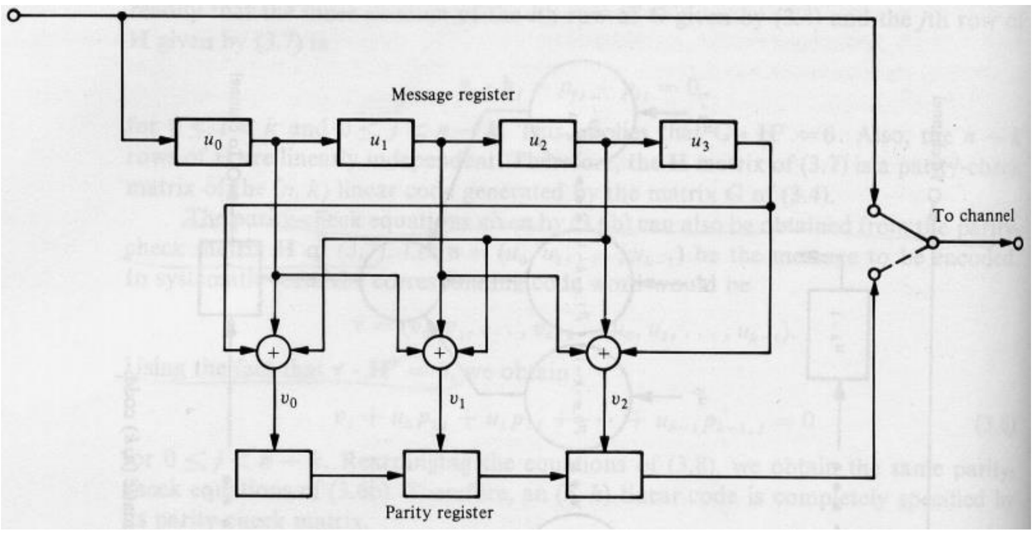
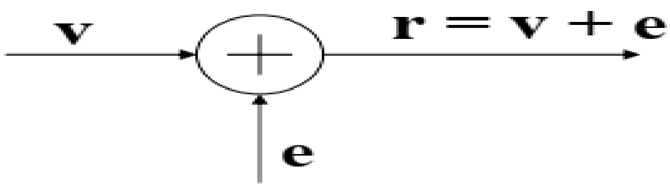



Figure: Encoding Circuit for (7,4) code

**Syndrome and Error Detection**

Let $v = (v_0, v_1, \ldots, v_{n-1})$ be a code word that was transmitted over a noisy channel. Let $r = (r_0, r_1, \ldots, r_{n-1})$ be the received vector at the outputof the channel




Where

$e = r + v = (e_0, e_1, \ldots, e_{n-1})$ is an n-tuple and the n-tuple 'e' is called the error vector (or error pattern).The condition is

$e_i = 1$ for $r_i \neq v_i$
$e_i = 0$ for $r_i = v_i$

Upon receiving r, the decoder must first determine whether r contains transmission errors. If the presence of errors is detected, the decoder will take actions to locate the errors, correct errors (FEC) and request for a retransmission of v.
When r is received, the decoder computes the following (n – k)-tuple.

$s = r \bullet HT$
$s = (s_0, s_1, \ldots, s_{n-k-1})$

where s is called the syndrome of r.

The syndrome is not a function of the transmitted codeword but a function of error pattern. So we can construct only a matrix of all possible error patterns with corresponding syndrome.

When s = 0, if and only if r is a code word and hence receiver accepts r as the transmitted code word. When s≠ 0, if and only if r is not a code word and hence the presence of errors has been detected. When the error pattern e is identical to a nonzero code word (i.e., r contain errors but s = r • HT = 0), error patterns of this kind are called undetectable error patterns. Since there are 2k – 1 non-zero code words, there are 2k – 1 undetectable error patterns. The syndrome digits are as follows:

$s_0 = r_0 + r_{n-k}\, p_{00} + r_{n-k}+1\ p_{10} + \cdots + r_{n-1}\, p_{k-1,0}$
$s1 = r_1 + r_{n-k}\, p_{01} + r_{n-k}+1\ p_{11} + \cdots + r_{n-1}\, p_{k-1,1}$

.

$s_{n-k-1} = r_{n-k-1} + r_{n-k}\, p_{0,n-k-1} + r_{n-k+1}\, p_{1,n-k-1} + \cdots + r_{n-1}\, p_{k-1,n-k-1}$

The syndrome s is the vector sum of the received parity digits $(r_0, r_1, \ldots, r_{n-k-1})$ and the parity-check digits recomputed from the received information digits $(r_{n-k}, r_{n-k+1}, \ldots, r_{n-1})$.

The below figure shows the syndrome circuit for a linear systematic (n, k) code.

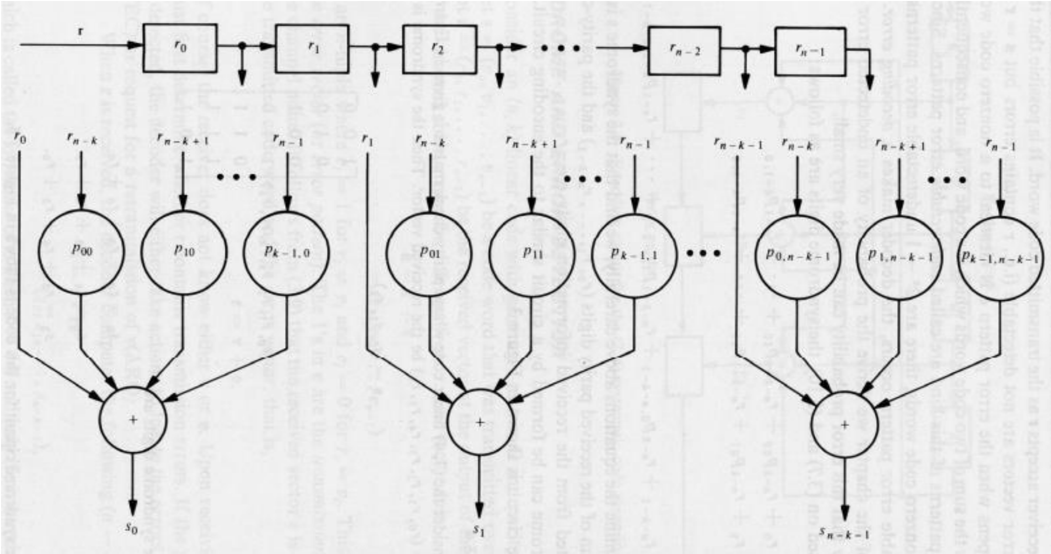



Figure: Syndrome Circuit

**Error detection and error correction capabilities of linear block codes:**

If the minimum distance of a block code C is $d_{min}$, any two distinct code vector of C differ in at least $d_{min}$ places. A block code with minimum distance $d_{min}$ is capable of detecting all the error pattern of $d_{min}$– 1 or fewer errors.

However, it cannot detect all the error pattern of $d_{min}$ errors because there exists at least one pair of code vectors that differ in $d_{min}$ places and there is an error pattern of $d_{min}$ errors that will carry one into the other. The random-error-detecting capability of a block code with minimum distance $d_{min}$ is $d_{min}$– 1.

An (n, k) linear code is capable of detecting 2n – 2k error patterns of length n Among the 2n – 1 possible non zero error patterns, there are 2k – 1 error patterns that are identical to the 2k – 1 non zero code words. If any of these 2k – 1 error patterns occurs, it alters the transmitted code word v into another code word w, thus w will be received and its syndrome is zero.

If an error pattern is not identical to a nonzero code word, the received vector r will not be a code word and the syndrome will not be zero.

**Hamming Codes:**

These codes and their variations have been widely used for error control in digital communication and data storage systems.

For any positive integer m ≥ 3, there exists a Hamming code with the following parameters:
Code length: n = 2m – 1
Number of information symbols: k = 2m – m – 1
Number of parity-check symbols: n – k = m
Error-correcting capability: t = 1(dmin= 3)

The parity-check matrix H of this code consists of all the non zero m-tuple as its columns (2m-1)

In systematic form, the columns of H are arranged in the following form

$$H = [I_m\, Q]$$

where $I_m$ is an m × m identity matrix

The sub matrix Q consists of 2m – m – 1 columns which are the m-tuples of weight 2 or more. The columns of Q may be arranged in any order without affecting the distance property and weight distribution of the code.

In systematic form, the generator matrix of the code is

$$G = [QT\ I2m–m–1]$$

where QT is the transpose of Q and I 2m–m–1 is an (2m – m – 1) ×(2m – m – 1) identity matrix.

Since the columns of H are nonzero and distinct, no two columns add to zero. Since H consists of all the nonzero m-tuples as its columns, the vector sum of any two columns, say $h_i$ and $h_j$, must also be a column in H, say $h_l h_i + h_j + h_l = 0$.The minimum distance of a Hamming code is exactly 3.

Using H' as a parity-check matrix, a shortened Hamming code can be obtained with the following parameters :

Code length: n = 2m – l – 1

Number of information symbols: k = 2m – m – l – 1

Number of parity-check symbols: n – k = m

Minimum distance : $d_{min} \geq 3$

When a single error occurs during the transmission of a code vector, the resultant syndrome is nonzero and it contains an odd number of 1’s (e x H’T corresponds to a column in H’).When double errors occurs, the syndrome is nonzero, but it contains even number of 1’s.

Decoding can be accomplished in the following manner:

i) If the syndrome s is zero, we assume that no error occurred

ii) If s is nonzero and it contains odd number of 1’s, assume that a single error occurred. The error pattern of a single error that corresponds to s is added to the received vector for error correction.

iii) If s is nonzero and it contains even number of 1’s, an uncorrectable error pattern has been detected.

**Problems:**

1.

The parity check bits of a (8,4) block code are generated by

$$c_0 = m_0 + m_1 + m_3$$
$$c_1 = m_0 + m_1 + m_2$$
$$c_2 = m_0 + m_2 + m_3$$
$$c_3 = m_1 + m_2 + m_3$$

where $m_1, m_2, m_3$ and $m_4$ are the message digits.

(a) Find the generator matrix and the parity check matrix for this code.

(b) Find the minimum weight of this code.

(c) Find the error-detecting capabilities of this code.

(d) Show through an example that this code can detect three errors/codeword.

Solution

1(a) $\mathbf{c} = [c_0 \cdots c_8] = [b_0 \cdots b_3 m_0 \cdots m_3] = [m_0 \cdots m_3]\left[\begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{matrix} \vdots \mathbf{I}_4\right]$

$$\text{Therefore, } \mathbf{G} = \left[\begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{matrix} \vdots \mathbf{I}_4\right]$$

$$\text{and then } \mathbf{H} = \left[\mathbf{I}_4 \vdots \begin{matrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{matrix}\right]$$

(b)

| m | C |
|---|---|
| 0000 | 0000 0000 |
| 0001 | 1011 0001 |
| 0010 | 0111 0010 |
| 0011 | 1100 0011 |
| 0100 | 1101 0100 |
| 0101 | 0110 0101 |
| 0110 | 1010 0110 |
| 0111 | 0001 0111 |
| 1000 | 1110 1000 |
| 1001 | 0101 1001 |
| 1010 | 1001 1010 |
| 1011 | 0010 1011 |
| 1100 | 0011 1100 |
| 1101 | 1000 1101 |
| 1110 | 0100 1110 |
| 1111 | 1111 1111 |

Therefore, minimum weight = 4

(c) $d_{min}$ = minimum weight = 4

Therefore, error-detecting capability = $d_{min} - 1 = 3$

(d) Suppose the transmitted code be 00000000 and the received code be 11100000.

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = [11100000] \begin{bmatrix} & 1 & 1 & 0 & 1 \\ \mathbf{I}_4 \vdots & 1 & 1 & 1 & 0 \\ & 1 & 0 & 1 & 1 \\ & 0 & 1 & 1 & 1 \end{bmatrix}^T = [1110] \neq \mathbf{0}$$

## Binary Cyclic codes:

Cyclic codes are the sub class of linear block codes.

Cyclic codes can be in systematic or non systematic form.

### Definition:

A linear code is called a cyclic code if every cyclic shift of the code vector produces some other code vector.

### Properties of cyclic codes:

(i) Linearity (ii) Cyclic

**Linearity:** This property states that sum of any two code words is also a valid code word.

$$X_1+X_2=X_3$$

**Cyclic:** Every cyclic shift of valid code vector produces another valid code vector.

Consider an n-bit code vector

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

Here $x_{n-1}$, $x_{n-2}$ ….$x_1$, $x_0$ represent individual bits of the code vector 'X'.

If the above code vector is cyclically shifted to left side i.e., One cyclic shift of X gives,

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

Every bit is shifted to left by one position.

### Algebraic Structures of Cyclic Codes:

The code words can be represented by a polynomial. For example consider the n-bit code word X = $\{x_{n-1},x_{n-2},\ldots\ldots\ldots\ldots x_1,x_0\}$.

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

$$X(p)=x_{n-1}p^{n-1}+x_{n-2}p^{n-2}+.......................+x_1p+x_0$$

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

p- Arbitrary variable of the polynomial

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

$p^{n-1}$ – MSB

$p^0$ -- LSB

p -- Second bit from LSB side

Polynomial representation due to the following reasons

(i) These are algebraic codes, algebraic operations such as addition, multiplication, division, subtraction etc becomes very simple.
(ii) Positions of the bits are represented with help of powers of p in a polynomial.

**<u>Generation of code words in Non-systematic form:</u>**

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

$M(p)=m_{k-1}p^{k-1}+m_{k-2}p^{k-2}+......................+m_1p+m_0$

Let X(p) be the code word polynomial

X(p)=M(p)G(p)

G(p) is the generating polynomial of degree ‘q’

For (n,k) cyclic codes, q=n-k represent the number of parity bits.

The generating polynomial is given as

$G(p)= p^q+g_{q-1}p^{q-1}+.................+g_1p+1$

Where $g_{q-1}$, $g_{q-2}$, ..............................$g_1$ 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

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

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

$$X_3(p) = M_3(p)\, G(p)$$

**<u>Generation of Code vectors in systematic form:</u>**

$$X = (k \text{ message bits} : (n\text{-}k) \text{ check bits}) = (m_{k-1}, m_{k-2}, \ldots\ldots\ldots\ldots\ldots\ldots\, m_1, m_0 : c_{q-1}, c_{q-2}, \ldots\ldots\ldots\ldots\ldots\, c_1, c_0)$$

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

The check bit polynomial is obtained by

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

**<u>Generator and Parity Check Matrices of cyclic codes:</u>**

**<u>Non systematic form of generator matrix:</u>**

Since cyclic codes are sub class 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.

Let generator polynomial given by equation

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

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

$$p^i G(p) = p^{i+q} + g_{q-1}p^{i+q-1} \ldots\ldots\ldots\ldots\ldots\ldots + g_1 p^{i+1} + p^i \quad \text{and } i = (k-1), (k-2), \ldots\ldots\ldots\ldots\ldots\ldots 2, 1, 0$$

**<u>Systematic form of generator matrix:</u>**

Systematic form of generator matrix is given by

$$G = [I_k : P_{kxq}]_{kxn}$$

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

$$t^{th} \text{ row of } G = p^{n-t} + R_t(p)$$

Where t= 1, 2, 3 ............... k

Lets divide $p^{n-t}$ by a generator matrix G(p). Then we express the result of this division in terms of quotient and remainder i.e.,

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

Here remainder will be a polynomial of degree less than q, since the degree of G(p) is 'q'.

The degree of quotient will depend upon value of t

Lets represent Remainder = $R_t(p)$

Quotient = $Q_t(p)$

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

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

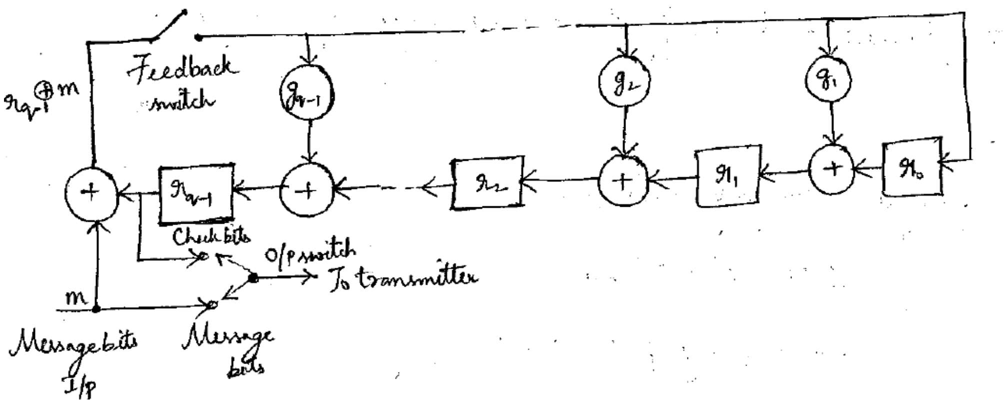
And t= 1,2,.................... k

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

Represents $t^{th}$ row of systematic generator matrix

**Parity check matrix** **$H = [P^T : I_q]_{qxn}$**

**<u>Encoding using an (n-k) Bit Shift Register:</u>**




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

The block diagram performs the division operation and generates the remainder. Remainder is stored in the shift register after all message bits are shifted out.

**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.

Lets represent the received code vector by Y.

If 'E' represents the error vector then the correct code vector can be obtained as

**X=Y+E or Y=X+E**

In the polynomial form we can write above equation as

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

X(p) = M(p)G(p)

Y(p)= M(p)G(p) + E(p)

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

If Y(p)=X(p)

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

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

Y(p)=Q(p)G(p) + R(p)

Clearly R(p) will be the polynomial of degree less than or equal to q-1

Y (p) =Q (p) G (p) +R (p)

M(p)G(p)+E(p)=Q(p)G(p)+R(p)

E(p)=M(p)G(p)+Q(p)G(p)+ R(p)

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

This equation shows that for a fixed message vector and generator polynomial, an error pattern or error vector 'E' depends on remainder R.

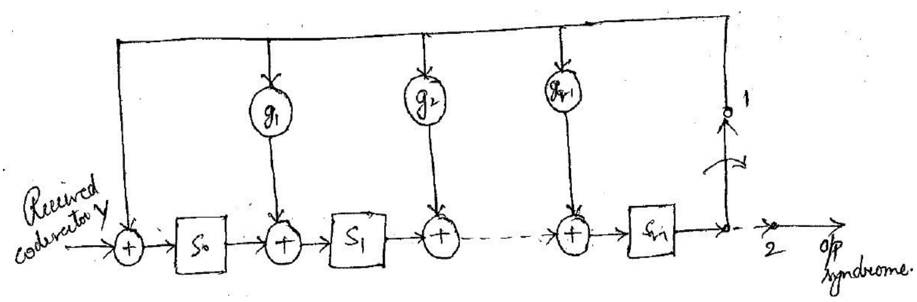
For every remainder 'R' there will be specific error vector. Therefore we can call the remainder vector 'R' as syndrome vector 'S', or R(p)=S(p). Therefore

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

Thus Syndrome vector is obtained by dividing received vector Y (p) by G (p) i.e.,

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

**Block Diagram of Syndrome Calculator:**



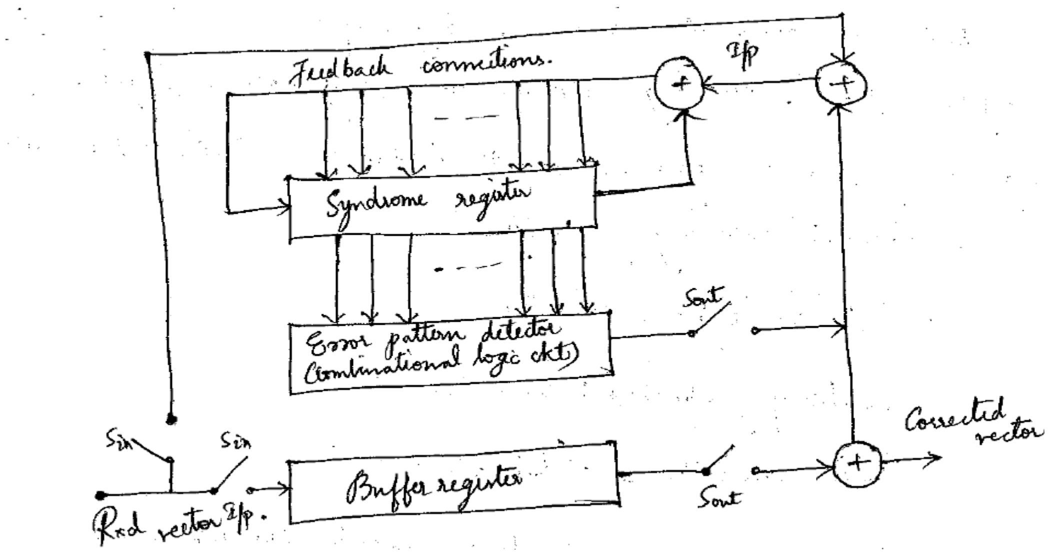

There are 'q' stage shift register to generate 'q' bit syndrome vector. Initially all the shift register contents are zero & the switch is closed in position 1.

The received vector Y is shifted bit by bit into the shift register. The contents of flip flops keep changing according to input bits of Y and values of g1,g2 etc.

After all the bits of Y are shifted, the 'q' flip flops of shift register contain the q bit syndrome vector. The switch is then closed to position 2 & clocks are applied to shift register. The output is a syndrome vector S= ($S_{q-1}$, $S_{q-2}$ ….$S_1$, $S_0$)

**Decoder of Cyclic Codes:**

Once the syndrome is calculated, then an error pattern is detected for that particular syndrome. When the error vector is added to the received code vector Y, then it gives corrected code vector at the output.




The switch named Sout is opened and Sin is closed. The bits of the received vector Y are shifted into the buffer register as well as they are shifted in to the syndrome calculator. When all the n bits of the received vector Y are shifted into the buffer register and Syndrome calculator the syndrome register holds a syndrome vector.

Syndrome vector is given to the error pattern detector. A particular syndrome detects a specific error pattern.

Sin is opened and Sout is closed. Shifts are then applied to the flip flop of buffer registers, error register, and syndrome register.

The error pattern is then added bit by bit to the received vector. The output is the corrected error free vector.

# Unit-5

# Convolution codes

## 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.

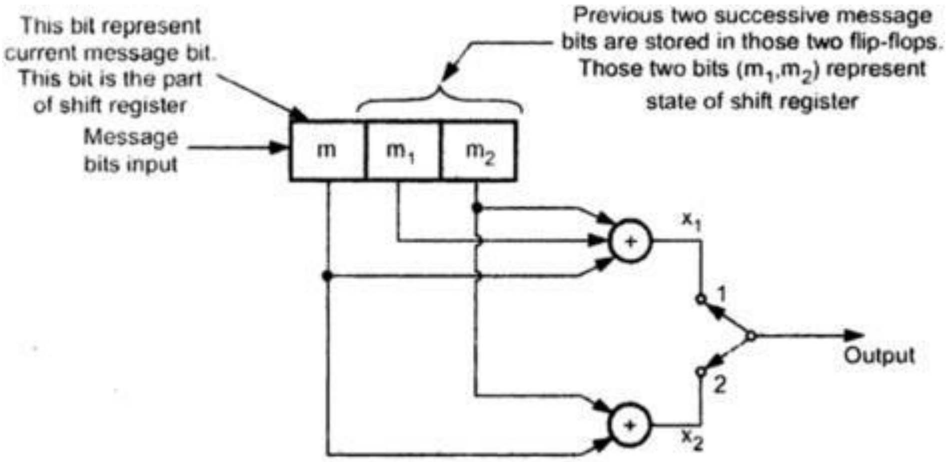



**Fig. 4.4.1 Convolutional encoder with k = 3, k = 1 and n = 2**

**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,

$$x_1 = m \oplus m_1 \oplus m_2 \qquad \dots (4.4.1)$$

and

$$x_2 = m \oplus m_2 \qquad \dots (4.4.2)$$

The output switch first samples $x_1$ and then $x_2$. The shift register then shifts contents of $m_1$ to $m_2$ 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 4.4.1 and equation 4.4.2). The output switch then samples $x_1$ then $x_2$. Thus the output bit stream for successive input bits will be,

$$X = x_1x_2x_1x_2x_1x_2 \dots \text{ and so on} \qquad \dots (4.4.3)$$

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

#### 4.4.1.1 Code Rate of Convolutional Encoder

The code rate of this encoder is,

$$r = \frac{k}{n} = \frac{1}{2} \quad \dots (4.4.4)$$

In the encoder of Fig. 4.4.1, observe that whenever a particular message bit enters a shift register, it remains in the shift register for three shifts i.e.,

First shift $\rightarrow$ Message bit is entered in position 'm'.

Second shift $\rightarrow$ Message bit is shifted in position $m_1$.

Third shift $\rightarrow$ 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 $m$, $m_1$, $m_2$. 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.

#### 4.4.1.2 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. 4.4.1 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.

#### 4.4.1.3 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. 4.4.1 has the dimension of (2, 1).

### 4.4.2 Time Domain Approach to Analysis of Convolutional Encoder

Let the sequence $\{g_0^{(1)}, g_1^{(1)}, g_2^{(1)} ...... g_m^{(1)}\}$ denote the impulse response of the adder which generates $x_1$ in Fig. 4.4.1 Similarly, Let the sequence $\{g_0^{(2)}, g_1^{(2)}, g_2^{(2)} ...... g_m^{(2)}\}$ denote the impulse response of the adder which generates $x_2$ in Fig. 4.4.1. 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,

$$\boxed{x_1 = x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l}} \qquad i = 0, 1, 2 ....... \qquad ... (4.4.6)$$

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

$$x_2 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l} \qquad i = 0, 1, 2 ....... \qquad ... (4.4.7)$$

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

As shown in the Fig. 4.4.1, the two sequences $x_1$ and $x_2$ are multiplexed by the switch. Hence the output sequence is given as,

$$\{x_i\} = \left\{ x_0^{(1)}\ x_0^{(2)}\ x_1^{(1)}\ x_1^{(2)}\ x_2^{(1)}\ x_2^{(2)}\ x_3^{(1)}\ x_3^{(2)} ...... \right\} \qquad ... (4.4.8)$$

$$v_1 = x_i^{(1)} = \left\{ x_0^{(1)}\ x_1^{(1)}\ x_2^{(1)}\ x_3^{(1)} ........ \right\}$$

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

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.

## 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.,

$$g^{(1)}(p) = g_0^{(1)} + g_1^{(1)}p + g_2^{(1)}p^2 + \ldots\ldots + g_M^{(1)}\, p^M \quad \ldots (4.4.13)$$

$$g^{(2)}(p) = g_0^{(2)} + g_1^{(2)}p + g_2^{(2)}p^2 + \ldots\ldots + g_M^{(2)}\, p^M \quad \ldots (4.4.14)$$

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.

Similarly we can write the polynomial for message polynomial i.e.,

$$m(p) = m_0 + m_1 p + m_2 p^2 + \ldots\ldots\ldots + m_{L-1} p^{L-1} \quad \ldots (4.4.15)$$

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

$$\boxed{\begin{aligned} x^{(1)}(p) &= g^{(1)}(p) \cdot m(p) \\ x^{(2)}(p) &= g^{(2)}(p) \cdot m(p) \end{aligned}} \quad \ldots (4.4.16)$$

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

## 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. 4.4.1. It is reproduced below for convenience.

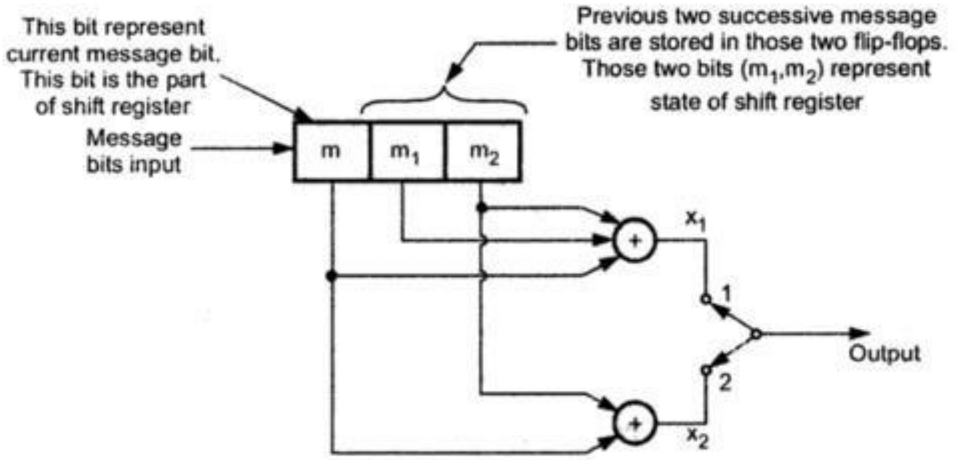



**Fig. 4.4.4 Convolutional encoder with k = 1 and n = 2**

### States of the Encoder

In Fig. 4.4.4 the previous two successive message 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 4.4.1.

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 |

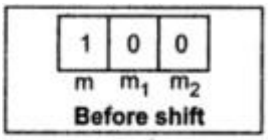
**Table 4.4.1 States of the encoder of Fig. 4.4.4**

### 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.

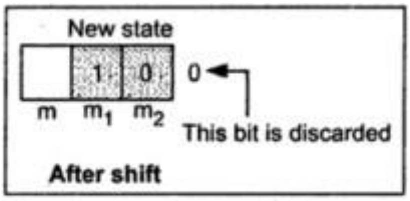
*1) When $m = 1$ i.e. first bit*

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




$x_1 = 1 \oplus 0 \oplus 0 = 1$

$x_2 = 1 \oplus 0 = 1$




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 4.4.2 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 reproduced as shown in Fig. 4.4.5.

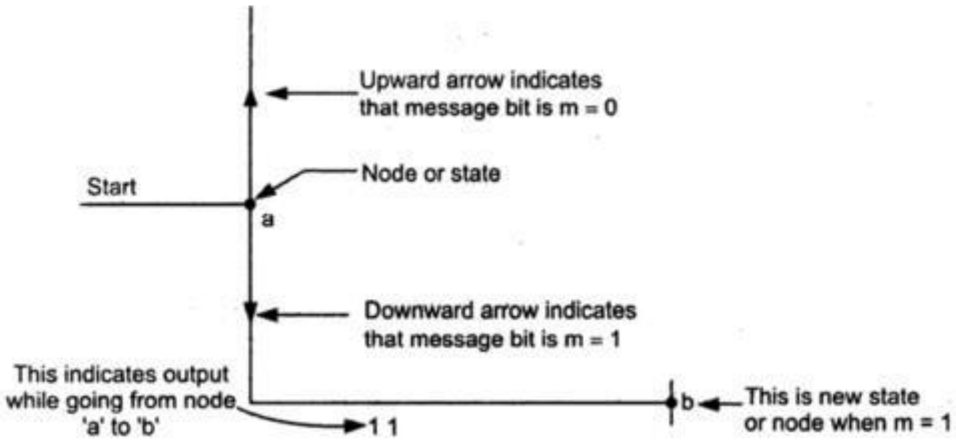
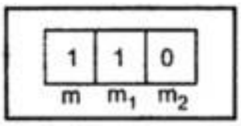



**Fig. 4.4.5 Code tree from node *'a'* to *'b'***

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*

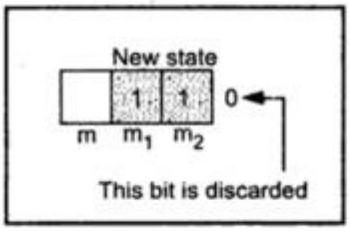
Now let the second message bit be 1. The contents of shift register with this input will be as shown below.




$x_1 = 1 \oplus 1 \oplus 0 = 0$

$x_2 = 1 \oplus 0 = 1$

These values of $x_1 x_2 = 01$ 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 $m_2 m_1 = 11$ or *'d'* and the outputs transmitted are $x_1 x_2 = 01$. Thus the encoder goes from state 'b' to state *'d'* if input is '1' and transmitted output $x_1 x_2 = 01$. This operation is illustrated by Table 4.4.2 in second row. The last column of the table shows the code tree for those first and second input bits.

**Example 4.4.8 :** *Determine the state diagram for the convolutional encoder shown in Fig. 4.4.32. Draw the trellis diagram through the first set of steady state transitions. On the second trellis diagram, show the termination of trellis to all zero state.*

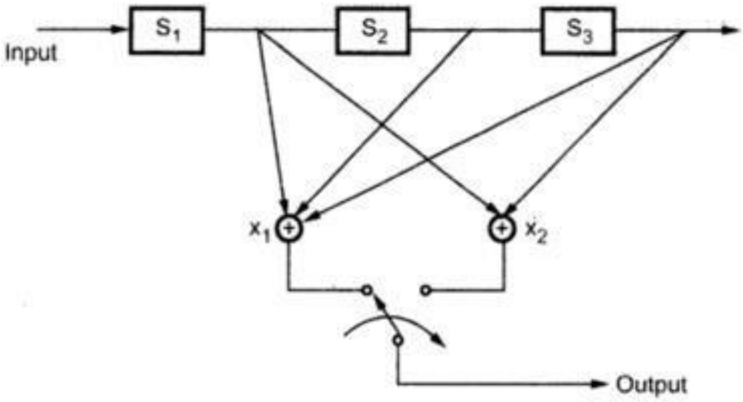



**Fig. 4.4.32 Convolutional encoder of example 4.4.8**

**Sol. : (i) To determine dimension of the code :**

For every message bit $(k = 1)$, two output bits $(n = 2)$ are generated. Hence this is rate $\frac{1}{2}$ code. Since there are three stages in the shift register, every message bit will affect output for three successive shifts. Hence constraint length, $K = 3$. Thus,

$$k = 1, \quad n = 2 \text{ and } K = 3$$

**ii) To obtain the state diagram :**

First, let us define the states of the encoder.

$s_3 s_2 = 0\,0,$ state 'a'

$s_3 s_2 = 0\,1,$ state 'b'

$s_3 s_2 = 1\,0,$ state 'c'

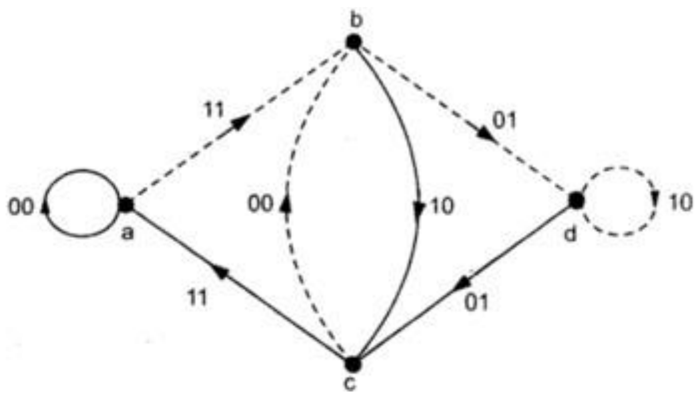
$s_3 s_2 = 1\,1,$ state 'd'

A table is prepared that lists state transitions, message input and outputs. The table is as follows :

| Sr. No. | Current state $s_3\ s_2$ | Input $s_1$ | Outputs $x_1 = s_1 \oplus s_2 \oplus s_3$ | $x_2 = s_1 \oplus s_3$ | Next state $s_2\ s_1$ |
|---|---|---|---|---|---|
| 1 | a = 0 0 | 0 | 0 | 0 | 0 0, i.e. a |
| | | 1 | 1 | 1 | 0 1, i.e. b |
| 2 | b = 0 1 | 0 | 1 | 0 | 1 0, i.e. c |
| | | 1 | 0 | 1 | 1 1, i.e. d |
| 3 | c = 1 0 | 0 | 1 | 1 | 0 0, i.e. a |
| | | 1 | 0 | 0 | 0 1, i.e. b |
| 4 | d = 1 1 | 0 | 0 | 1 | 1 0, i.e. c |
| | | 1 | 1 | 0 | 1 1, i.e. d |

**Table 4.4.8 : State transition table**

Based on above table, the state diagram can be prepared easily. It is shown below in Fig. 4.4.33.




**iii) To obtain trellis diagram for steady state :**

From Table 4.4.9, the code trellis diagram can be prepared. It is steady state diagram. It is shown below.

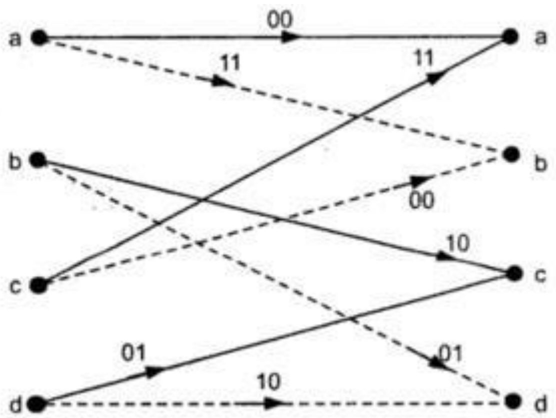



**Fig. 4.4.34 Code trellis diagram for steady state**

**Decoding methods of Convolution code:**

**1.Veterbi decoding**

**2.Sequential decoding**

**3.Feedback decoding**

Veterbi algorithm for decoding of convolution codes(maximam likelihood decoding):

Let represent the received signal by y.

Convolutional encoding operates continuously on input data

Hence there areno code vectorsand blocks such as.

**Metric**:it is the discrepancybetwen the received signal y and the decoding signal at particular node .this metric can be added over few nodes a particular path

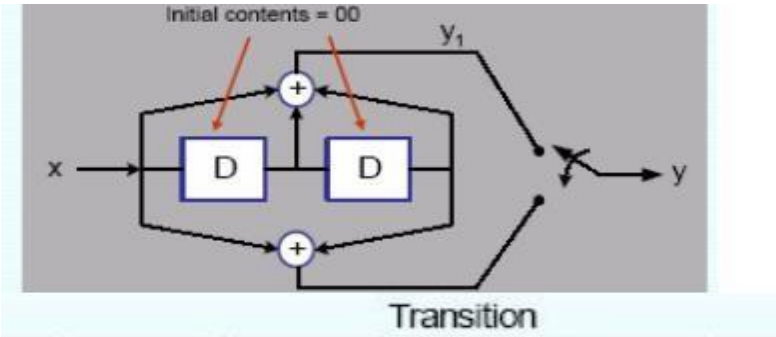
**Surviving path:** this is the path of the decoded signalwith minimum metric

In veterbi decoding ametric isassigned to each surviving path

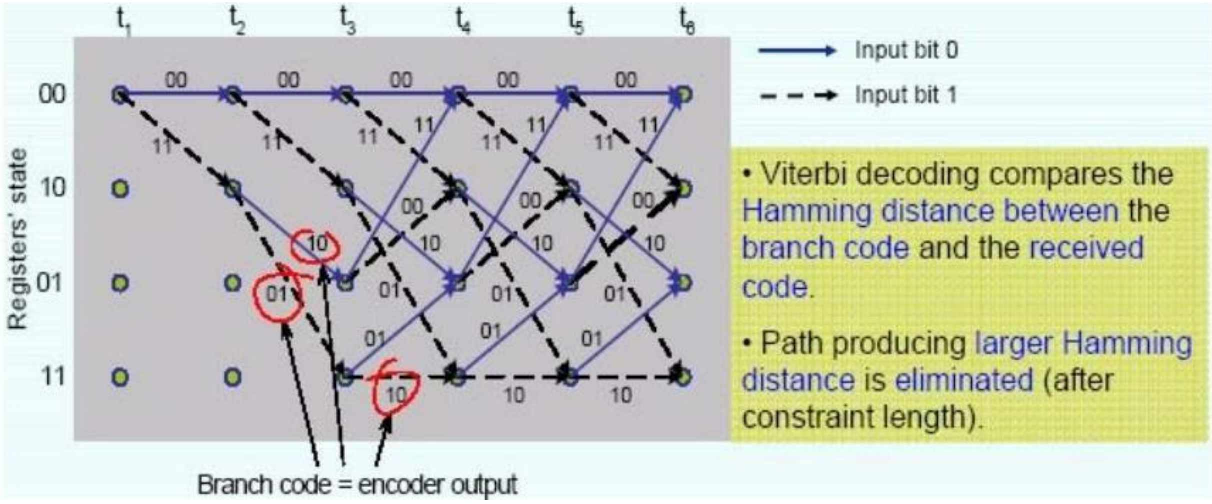
Metric of the particular is obtained by adding individual metric on the nodes along that path.

Y is decoded as the surviving path with smallest metric.

Example:




**Exe:**




*Input data* : m =1 1 0 1 1

*Codeword* : X = 11 01 01 00 01

*Received code* : Z = 11 01 01 10 01