UNIT-VI

Linear Block Codes

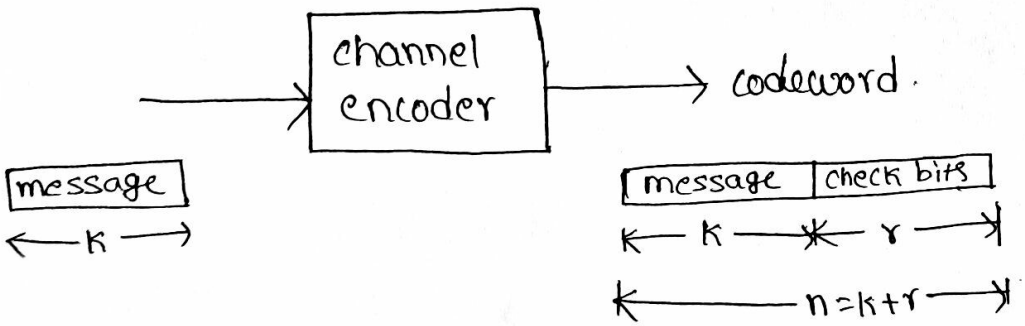
Error control methods are brodly classified into two types.

(i) Linear block codes

(ii) convolutional codes.

In linear block codes K bits of message is followed by r check bits.

check bits are used to verify the information at the receiver.



The output of channel encoder has n bits of information known as codeword.

The message bits are followed by check bits known as codewords. These codewords are known as systematic codewords.

The K message bits has 2^K distinct codewords. Those are linearly independent known as linear systematic code word.

Matrix Description:-

The encoding operation in linear block encoding scheme consists of two basic steps.

The information sequence is segmented into message blocks each block consisting of successive k information bits.

The encoder transforms each message block into a larger block of n bits according to some predetermine set of rules.

$$c_{k+1} = P_{11} d_1 + P_{21} d_2 + P_{31} d_3 + \cdots + P_{k1} d_k .$$

$$c_{k+2} = P_{12} d_1 + P_{22} d_2 + \cdots + P_{k2} d_k .$$

$$\vdots$$

$$c_n = P_{1,n-k} d_1 + P_{2,n-k} d_2 + \cdots + P_{k,n-k} d_k .$$

These are all modulo-2 addition operations.

$$C = (d_1, d_2 \cdots d_k) \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & P_{11} & P_{12} & \cdots & P_{1,n-k} \\ 0 & 1 & \cdots & 0 & P_{21} & P_{22} & \cdots & P_{2,n-k} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \\ 0 & 0 & \cdots & 1 & P_{k,1} & P_{k,2} & \cdots & P_{k,n-k} \end{bmatrix}}_{k \times n}$$

whole matrix order is $k \times n$.

P_{11}, P_{12} -- are polynomial coefficients, those are only 0's and 1's.

$$C = D G_1$$

C = channel matrix (code vector) (1 x n size)

D = input data (1 x k) (size).

G_1 = Generator matrix of k x n size.

→ The generator matrix for a 6x3 block code is given below find the code vectors of the code.

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

$k \times n = 3 \times 6$

$k = 3$

$n = 6.$

Sol:- code vector 1

$D = 0\ 0\ 0$

$$C = DG = \underset{1\times3}{[0\ 0\ 0]} \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]_{3\times6}$$

$$= [0\ 0\ 0\ 0\ 0\ 0]_{1\times6}.$$

code vector 2:-

$D = 0\ 0\ 1$

$$C = DG = [0\ 0\ 1] \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

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

code vector 3:-

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

$$c = DG_1 = [0\ 1\ 0] \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$= [0\ 1\ 0\ 1\ 0\ 1]_{1\times 6}$$

code vector 4:-

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

$$c = DG_1 = [0\ 1\ 1] \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$= [0\ 1\ 1\ 0\ 1\ 1]_{1\times 6}$$

code vector 5:-

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

$$c = DG_1 = [1\ 0\ 0] \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$c = [1\ 0\ 0\ 0\ 1\ 1]_{1\times 6}$$

code vector 6:-

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

$$c = DG_1 = [1\ 0\ 1] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$= [1\ 0\ 1\ 1\ 0\ 1]_{1\times 6}$$

code vector 7:-

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

$$c = DG_1 = [1\ 1\ 0] \cdot \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

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

code vector 8 :-

$$D = [1\ 1\ 1]$$

$$C = DG = [1\ 1\ 1] \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

$$= [1\ 1\ 1\ 0\ 0\ 0]_{1\times 6}.$$

$$= 0 =$$

Syndrome Decoding :-

parity check matrix, $H = [P^T : I_{n-k}]_{(n-k)\times n}$.

Generator matrix $G = [I_{k\times k} : P]$.

$C_1 H_1^T = 0$ then the codeword C_1 is generated from the generator matrix G_1.

$$\therefore \boxed{C H^T = 0}$$

for the error correction and detection we use syndrome vector i.e

$$\boxed{S = R H^T}$$

R = Received code vector.

$R = C + E$

C = original code vector

E = error vector.

$S = (C+E) H^T$

$= C H^T + E H^T$

$= 0 + E H^T$

$$\therefore \boxed{S = E H^T = R H^T}$$

S = syndrome vector.

* syndrome of a received vector is zero if R is a valid code vector.

* If errors occur in transmission then the syndrome 's' of the received vector is non-zero.

→ consider a 7x4 block code generated by

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]_{4\times 7}$$ find

$k = 4$
$n = 7$.

(i) parity check matrix

(ii) code vectors for a message 1011 and 1110 and 1111.

(iii) show that how a single error can be corrected.

Sol. (i) $n = 7$
$k = 4$.

$$\therefore [P^T] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}^T$$

$$H = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]_{n-k\times n}$$

(ii) Given generator matrix is

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]_{4\times 7}$$

(b) 1011

$$c_1 = [1\ 0\ 1\ 1] \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$c_1 = [1\ 0\ 1\ 1\ 0\ 0\ 1].$$

(b) 1110

$$c_2 = [1\ 1\ 1\ 0] \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

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

(c) 1111

$$c_3 = [1\ 1\ 1\ 1] \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$= [1\ 1\ 1\ 1\ 1\ 1\ 1].$$

(iii) Suppose if c_1 is transmitted

$$c_1 = 1011001.$$

if the received code word is.

$R = 1010001$ (There is an error in 4th position).

$$S = RH^T$$

$$= [1010001] \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ \boxed{0 & 1 & 1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$(2^{n-k}-1)$ → here $(2^{7-4}-1)$ → (2^3-1) combinations are there in H^T

∴ 3 check bits.

$S = [0\ 1\ 1]$.

verify with which row syndrome vector is identical in H^T

here syndrome vector is identical to 4^{th} row.

so in 4^{th} position we have the error.

∴ $R = 1011001$

For this code a single error i^{th} bit of 'c' could lead to a syndrome vector that would be identical to the i^{th} row of the H^T. Thus the single error can be corrected at the receiver by comparing syndrome vector with the rows of H^T. and correcting the i^{th} received bit if syndrome vector matches with i^{th} row of H^T.

→ $C_2 = 1110100$

$R = 1100100$.

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

$S = [1\ 0\ 1]$

syndrome vector is identical with 3^{rd} row

∴ error in 3^{rd} position.

$R_2 = [1\ 1\ 1\ 0\ 0\ 1\ 0]$

Cyclic codes

Cyclic codes are sub class of linear Block codes.

Advantages:-

- Encoding and Decoding calculations can be easily implemented by using simple shift registers and feed back connection.
- These codes have a fair amount of mathematical structure that makes it possible to design codes with useful error correcting properties.

Algebraic structure of cyclic codes:-

$$(V, K) \rightarrow C$$

$$V = V_0 \; V_1 \; --- \; V_{n-1}$$

$$V^1 = V_{n-1} \; V_0 \; V_1 \; ---- \; V_{n-2}$$

$$V^2 = V_{n-2} \; V_{n-1} \; V_0 \; V_1 \; --- \; V_{n-3}$$

$$\vdots$$

$$V^i = V_{n-i} \; V_{n-i+1} \; V_{n-i+2} \; --- \; V_{n-i-1}$$

if $V = 0\,0\,0\,1$

$V^1 = 1\,0\,0\,0$ (circular right shift)

$V^2 = 0\,1\,0\,0$

This property of cyclic codes allows us to treat the element of each code word as the coefficients of a polynomial of degree $n-i$.

$$v = v_0 + v_1 x + v_2 x^2 + ---- + v_{n-1} x^{n-1}.$$

either for addition or multiplication we should use modulo-2 operations.

Addition → XOR operation.

Multiplication → AND operation.

cyclic codes are basically two types

(i) Non-Systamatic cyclic code

(ii) systamatic cyclic code.

Non-systamatic cyclic code:-

If $g(x)$ is a polynomial of degree $(n-k)$ and is a factor of x^{n+1} then $g(x)$ generates nk cyclic code in which the code polynomial $V(x)$ for a data vector

$$D = D_0 \; D_1 \; D_2 \; \cdots \; D_{k-1}$$

$$D(x) = D_0 + D_1 x + D_2 x^2 + \cdots + D_{k-1} x^{k-1}$$

$$\boxed{V(x) = D(x)\, g(x)} \rightarrow \text{non-systamatic code vector.}$$

systamatic cyclic code:-

Given the generator polynomial $g(x)$ of a cyclic code the code can be put into a systamatic form as

$$V = [\underbrace{r_0 \; r_1 \; r_2 \; \cdots \; r_{n-k-1}}_{\text{check bits}} \; \underbrace{D_0 \; D_1 \; \cdots \; D_{k-1}}_{\text{message bits.}}]$$

$$r(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n-k-1} x^{n-k-1}$$

↳ parity check polynomial.

$r(x)$ is reminder of $\left[\dfrac{x^{n-k}\, D(x)}{g(x)}\right]$

$$\boxed{V(x) = r(x) + x^{n-k}\, D(x)}$$

→ The generator polynomial of a 7X4 cyclic code is $g(x) = 1+x+x^3$ find the codewords of the following messages in systamatic and non-systamatic ways.

(i) 1010 (ii) 0000 (iii) 0101 (iv) 1101

Sol. (i) 1010

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

$D = 10\ 10 = 1\cdot x^0 + 0\cdot x^1 + 1\cdot x^2 + 0\cdot x^3.$

$= 1+x^2.$

non-systamatic cyclic code

$V(x) = D(x)\ g(x) = (1+x^2)(1+x+x^3)$

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

$= 1+x+x^2+x^5$

$\therefore V = [1\ 1\ 1\ 0\ 0\ 1\ 0]$ (we have to get 7 bits so place '0' at last).

systamatic cyclic code

$V(x) = r(x) + x^{n-k} D(x).$

$r(x)$ = reminder of $\left[\frac{x^{n-k} D(x)}{g(x)}\right]$

$x^{n-k} D(x) = x^{7-4} D(x) = x^3(1+x^2) = x^3+x^5.$

$\frac{x^{n-k} D(x)}{g(x)} = \frac{x^3+x^5}{1+x+x^3}.$

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

$$x^5+x^3+x^2$$

$$x^2$$

$\therefore r(x) = x^2 = 001.$

$$\therefore \quad V(x) = r(x) + x^{n-k} D(x)$$

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

$$= \underbrace{001}_{r} \underbrace{101}_{D}.$$

(ii) 0000

$D = 0000$

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

Non-systamatic code:-

$$V(x) = D(x)\, g(x) = 0 \cdot g(x) = 0.$$

$$V = [0\ 0\ 0\ 0\ 0\ 0\ 0]$$

systamatic code:-

$$x^{n-k} D(x) = x^3(0) = 0$$

$r(x)$ = reminder of $\dfrac{x^{n-k} D(x)}{g(x)}$

$$= \frac{0}{g(x)} = 0.$$

$$\therefore V(x) = r(x) + x^{n-k} D(x)$$

$$= 0+0$$

$$= 0$$

$\therefore V = [0\ 0\ 0\ 0\ 0\ 0\ 0].$

(iii) 0101

$D = 0101 \Rightarrow D(x) = x + x^3$.

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

Non-systematic code:-

$$V(x) = D(x)g(x) = (x+x^3)(1+x+x^3)$$
$$= x + x^2 + x^4 + x^4 + x^3 + x^6$$
$$= x + x^2 + x^3 + x^6$$

$\therefore V = [0111001]$

systematic-code :-

$$x^{n-k} D(x) = x^3(x+x^3) = x^4 + x^6$$

$$r(x) = \text{reminder}\left[\frac{x^{n-k} D(x)}{g(x)}\right] = \text{re}\left[\frac{x^4 + x^6}{1+x+x^3}\right]$$

$$x^3+x+1 \;)\; x^6 + x^4 \;(\; x^3$$
$$x^6 + x^4 + x^3$$
$$x^3$$

$\therefore r(x) = x^3 = 0001$.

$n - k = r \quad (7-4=3)$

$\therefore$ $n-k=3$ bits only so one can be neglected $r = 000$

$$\therefore V(x) = r(x) + x^{n-k} D(x)$$
$$= 0 + x^3(x+x^3)$$
$$= x^4 + x^6$$

$\therefore V = [0000101]$.

(iv) 1101

$D = 1101 = 1 + x + x^3$.

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

Non- systamatic code:-

$V(x) = D(x)\, g(x)$.

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

$= 1 + x + x^3 + x + x^2 + x^4 + x^3 + x^4 + x^6$.

$= 1 + x^2 + x^6$.

$\therefore V = [1\ 0\ 1\ 0\ 0\ 0\ 1]$

systamatic code:-

$x^{n-k} D(x) = x^3 (1 + x + x^3)$.

$= x^3 + x^4 + x^6$.

$r(x) = re\left[\frac{x^{n-k} D(x)}{g(x)}\right] = re\left[\frac{x^3 + x^4 + x^6}{1 + x + x^3}\right]$.

$x^3 + x + 1 \;)\; x^6 + x^4 + x^3 \;(\; x^3$

$x^6 + x^4 + x^3$

0

$r(x) = 0$

$\therefore V(x) = r(x) + x^{n-k} D(x)$

$= 0 + x^3 + x^4 + x^6$

$= x^3 + x^4 + x^6$.

$V = [0\ 0\ 0\ 1\ 1\ 0\ 1]$.

Error correction and detection capabilities of linear Block codes:

* **weight of the code:-**

Number of non-zero elements in that word.

Ex:- 1010100 → weight = 3.

* **Hamming distance:-**

Hamming distance between two vectors c_1 and c_2 is defined as number of components in which they differ.

* **Theorem 1:-**

The minimum distance of a linear block code is equals to the minimum weight of any non-zero word in the code.

codevectors			weight
0000	→	000000	→ 0
001	→	001110	→ 3
010	→	010101	→ 3
011	→	011011	→ 4
100	→	100011	→ 3
101	→	101101	→ 4
110	→	110110	→ 5
111	→	111000	→ 3

Minimum distance = 3.

Theorem 2:-

A linear Block code with a minimum distance "d_{min}" can correct up to $(d_{min}-1)/2$ errors and detect upto $(d_{min}-1)$ errors in each code word.

→ Hamming code is Single error correcting code.

→ To have all distinct rows in H^T, the condition to be satisfied is

$$2^r - 1 \geq n$$

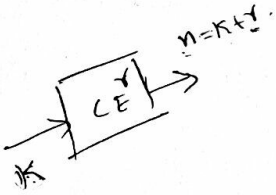
$$2^{n-k} - 1 \geq n.$$

$$2^{n-k} \geq 1+n$$

$$(n-k)\log_2 2 \geq \log(1+n)$$

$$n-k \geq \log_2(1+n)$$

$$\boxed{n \geq k + \log_2(1+n).}$$



(1) Design a block code with a minimum distance of 3 and a message block size of 8 bits?

Sol

$k = 8$.

$d_{min} = 3$.

$n \geq k + \log_2(1+n)$.

$12 \geq 8 + \log_2(1+12)$

$12 \geq 11.58$.

$\therefore r = n-k = 12-8 = 4$ bits.

$$H = \left[\begin{array}{c|c} P^T & I_{n-k} \end{array}\right]_{n \times (n-k)}.$$

$$H^T = \left[\begin{array}{c} 0011 \\ 0101 \\ 0110 \\ 0111 \\ 1001 \\ 1010 \\ 1011 \\ 1100 \\ \hline 1000 \\ 0100 \end{array}\right.$$

$$H = \left[\begin{array}{c|c} P & \begin{array}{c} 1000 \\ 0100 \\ 0010 \\ 0001 \end{array} \end{array}\right]$$

0000 x
0001 ✓
0010 x
0011
0100 x
0101
0110
0111
1000 x
1001
1010
1011
1100
1101
1110
1111

$$G_1 = \left[I_{k \times k} \mid P_{k \times (n-k)} \right]_{8 \times 12}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}_{8 \times 4}$$

Cyclic Redundancy Check (CRC):-

* This is a type of polynomial code in which a bit string is represented in the form of polynomials with coefficients of 0 and 1 only.

* For CRC code, the sender and receiver must agree upon a generator polynomial G(x).

* A codeword can be generated for a given dataword (message) polynomial M(x) with the help of long division. This Technique is more powerful than the parity check and checksum error detection.

* CRC is based on binary division. A sequence of redundant bits called CRC or CRC reminder is appended at the end of data unit such as byte. The resulting data unit after adding CRC reminder becomes exactly divisible by another predetermined binary number.

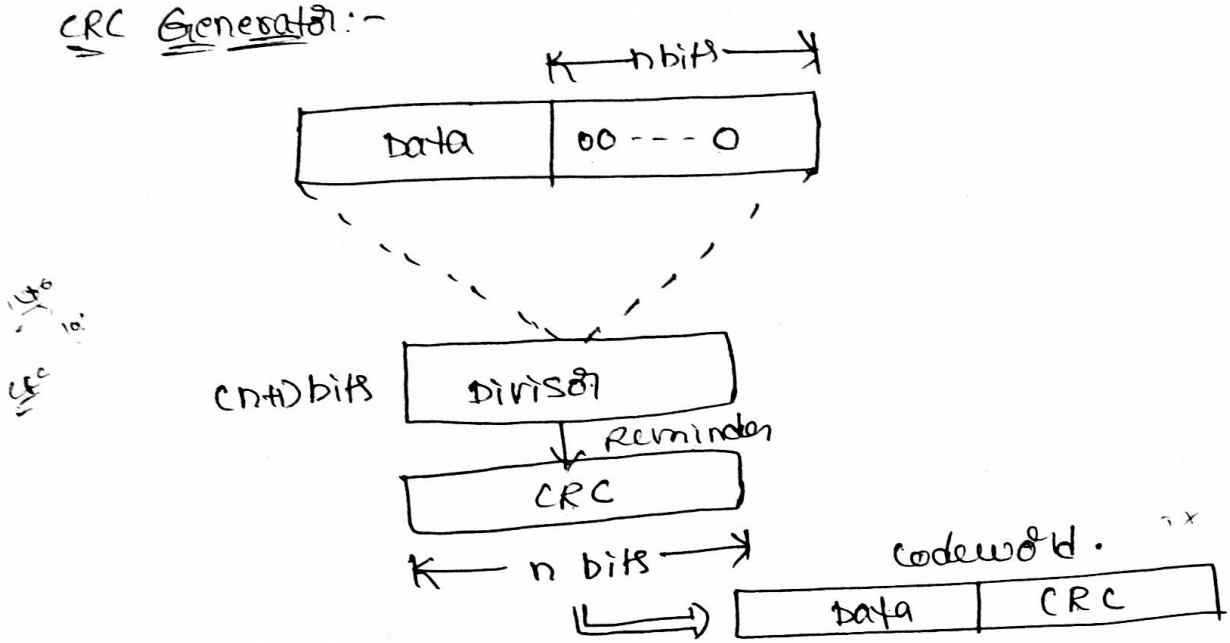
* At the receiver, this data unit is divided by the same binary number. There is no error if this division does not yield any reminder. But, a Non-Zero reminder indicates presence of errors in the received data unit. Such an erroneous data unit is rejected.

Requirements of CRC:-

A CRC will be valid if and only if it satisfies the following requirements.

(i) It must have exactly one less bit than divisor.
(ii) Appending the CRC to the end of the data unit must result in the bit sequence which is exactly divisible by the divisor

CRC Generator:-



The procedure for CRC Generation:-

(i) We Append a string of n 0's to the data unit where n is 1 less than the Number of bits in the predecided divisor (n+1) bit long).

(ii) We divide the newly generated data unit in step (i) by the divisor. This is a binary division.

(iii) The reminder obtained after the division in step (ii) is the n bit CRC.

(iv) This CRC will replace the n 0's appended to the data unit in step (i), to get the codeword to be transmitted.

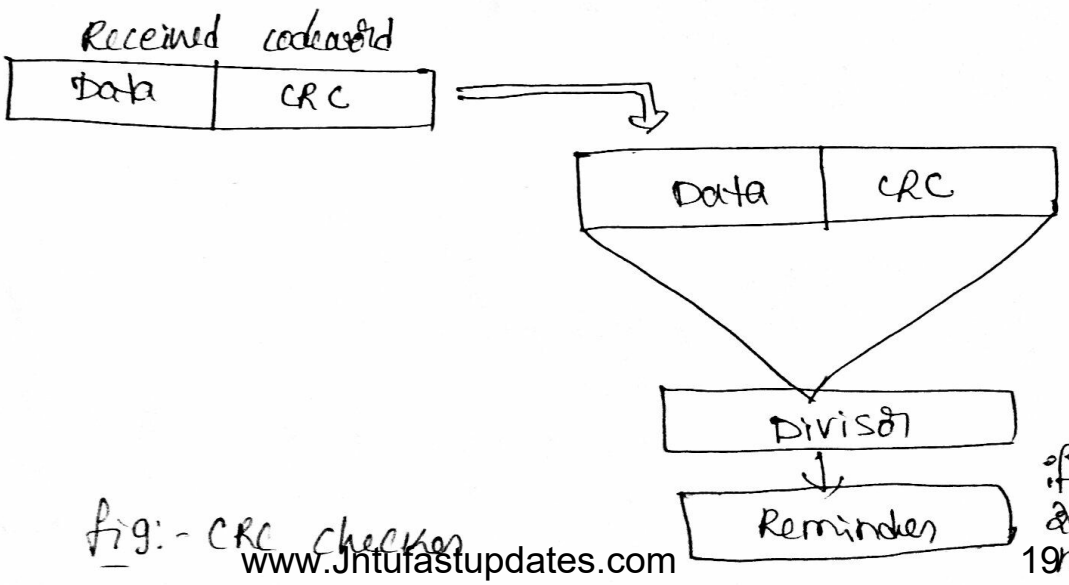


fig:- CRC checker

→ Generate the CRC code for the data word of 11001010. The divisor is 10101.

Sol:- Given

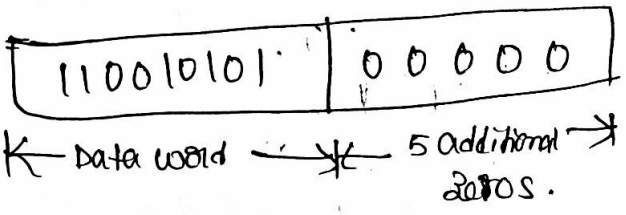
Data word: 11001010

Divisor: 10101

The Number of data bits = m = 9.

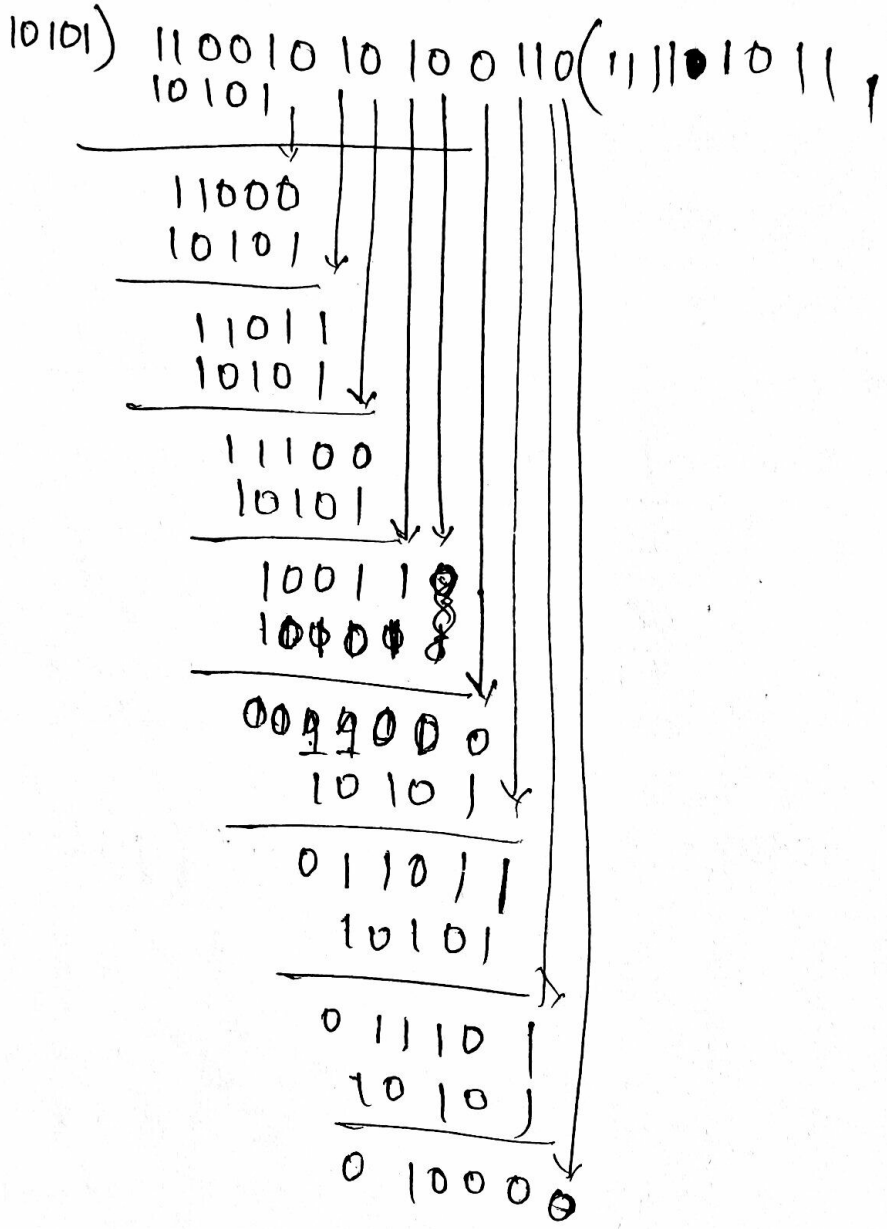
The Number of bits in the codeword = N.

Dividend = Dataword + Number of zeros.




```
10101) 110010100000 (11111001111
       10101
       -----
        01100
        10101
        -----
        01110
        10101
        -----
        01110
        10101
        -----
        01001
        10101
        -----
        00110
        10101
        -----
        01010
        10101
        -----
        01111
        10101
        -----
        10110
        10101
        -----
        00110
        10101
```

codeword = 1100101010011●.



CONVOLUTION CODES

There are many different error correcting codes. These code words can be classified into two categories as under:

(i) Block codes and

(ii) Convolutional codes.

The major difference between these codes is that the block codes do not need memory, whereas the convolutional codes do need memory.

Another way of classifying codes is:

(i) Linear codes, and

(ii) Non-linear codes.

Linear codes have an important property that any two code words of a linear code can be added in modulo-2 adder to produce a third code word in the code. On the other hand, non-linear codes do not exhibit such a property. All the practically used codes are linear codes.

Block codes:-

In block codes, the block of n bits generated by the encoder in a particular time slot depends only on the block of k message bits within that time slot.

Convolution codes:-

In the convolution codes, the block of n bits generated by the encoder in a time slot depends not only on the k message bits within that time slot, but also on the preceding 'L' blocks of the message bits. Generally, the values of k and n will be small.

* The convolutional codes can be designed to either detect or correct errors. However, because data is usually retransmitted in blocks, the block codes are more suitable for error detection and the convolutional codes are more suitable for error correction.

* Encoding of the convolutional codes can be accomplished using simple shift register. The convolutional codes perform as well better than the block codes in many error control applications.

Encoding of convolutional codes:-

Convolution codes can be represented by (v, k, N). → code dimension

where v = number of modulo-2 adders.

k = no. of data bits.

N = N-stage shift registers.

The no. of coded bits "$n = v(k+N)$" and the

code rate $r = \frac{k}{n}$ bits/symbol.

A convolution-code is generated by combining the outputs of a K-stage shift register through the employment of v exclusive-OR logic summers. Such a coder is illustrated in the following figure for the case $k=4$ and $v=3$. Here m_1 through m_4 are 1-bit storage (memory) devices such as flip-flops.

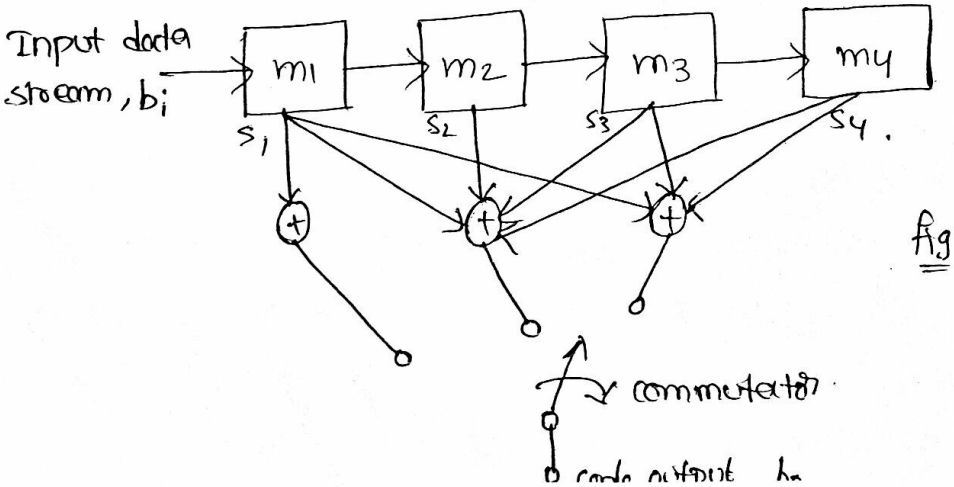


fig:- An example of convolution encoder

The outputs v_1, v_2 & v_3 of the adders in the above figure are

$$v_1 = s_1$$

$$v_2 = s_1 \oplus s_2 \oplus s_3 \oplus s_4$$

$$v_3 = s_1 \oplus s_3 \oplus s_4$$

The operation of the encoder proceeds as follows:

* We assome that initially the shift register is clear. The first bit of the input data stream is entered into m_1.

* During this message bit interval the commutator samples, in turn the adder outputs v_1, v_2 & v_3. Thus a single bit yields, in the present case, three encoded output bits. The encoder is therefore of rate 1/3.

* The next message bit then enters m_1, while the bit initially in m_1 transfers to m_2, and the commutator again samples all the 3 adder outputs.

* This process continues until eventually the last bit of the message has been entered into m_1.

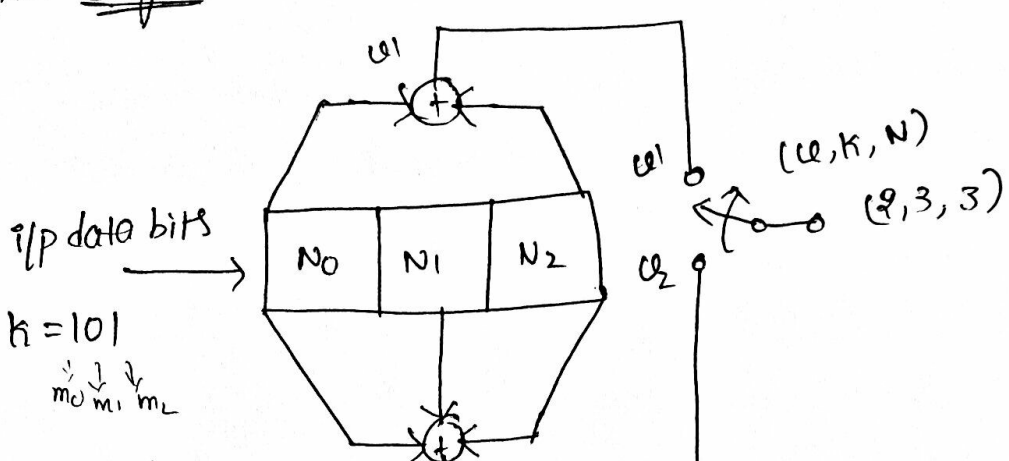
Thereafter, in order that every message bit may proceed entirely through the shift register, and hence be involved in the complete coding process, enough 0's are added to the message to transfer the last message bit through m_4, and, hence, out of the shift register. The shift register then finds itself in its initial "clear" condition.

The methods to encode convolution codes are

(1) connection oriented
(2) Time domain approach
(3) Transform domain approach
(4) Graphical representation
    (a) state diagram
    (b) Tree diagram
    (c) Trellis diagram

connection diagram:-

①

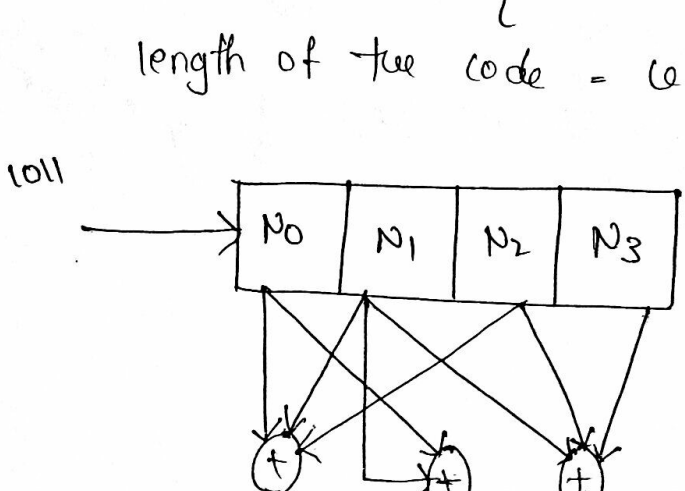


Time	i/p bits	N_0	N_1	N_2	G_1 $N_0 \oplus N_2$	G_2 $N_0 \oplus N_1 \oplus N_2$
-	-	0	0	0	-	-
1	1	1	0	0	1	1
2	0	0	1	0	0	1
3	1	1	0	1	0	0
4	0	0	1	0	0	1
5	0	0	0	1	1	1
6	0	0	0	0	0	0

coded output bit stream = {11 01 00 01 11 00}

length of the code = n(k+N) = 2(3+3) = 12.

②



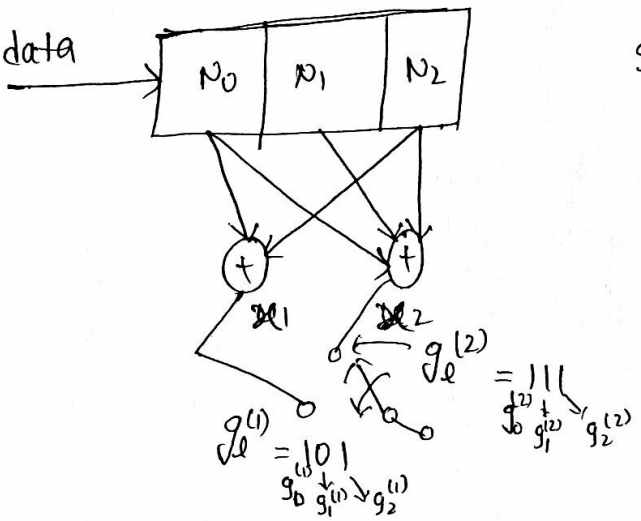
Time	input bits	N_0	N_1	N_2	N_3	v_1 $N_0 \oplus N_1 \oplus N_2$	v_2 $N_0 \oplus N_1$	v_3 $N_1 \oplus N_2 \oplus N_3$
–	⟶	0	0	0	0	–	–	–
1	1	1	0	0	0	1	1	0
2	1	1	1	0	0	1	1	1
3	0	0	1	1	0	0	1	1
4	1	1	0	1	1	0	0	0
5	0	0	1	0	1	0	1	0
6	0	0	0	1	0	1	0	0
7	0	0	0	0	1	0	0	1
8	0	0	0	0	0	0	0	0

code word $c = \{110\ 111\ 011\ 000\ 010\ 100\ 001\ 000\}$.

length of the code $= v(k+N) = 3(4+4) = 24$

Time domain approach:-

The time-domain behavour of a binary convolutional encoder may be defined in terms of n-impulse responses.



$g_i^{(1)}$ or $g_i^{(2)} = 1$ – if there is connection.

$g_i^{(1)}$ or $g_i^{(2)} = 0$ if there is no connection.

Let the impulse response of the adder generating x_1 in the above figure, be given by the sequence $\{g_0^{(1)}, g_1^{(1)} \ldots g_i^{(1)}\}$.

similarly, let the sequence $\{g_0^{(2)}, g_1^{(2)}, \ldots g_i^{(2)}\}$ denote the impulse response of the adder generating x_2.

These impulse responses are also called as "generator sequence" of the code.

Let $(m_0, m_1, m_2 \cdots)$ denote the message sequence entering the encoder one bit at a time (starting from m_0).

The encoder generates two output sequences by performing convolutions on the message sequence with the impulse responses.

The bit sequence x_1 is given by,

$$x_1 = x_i^{(1)} = \sum_{l=0}^{N-1} g_l^{(1)} m_{i-l}$$

similarly, the other bit sequence x_2 is given by,

$$x_2 = x_i^{(2)} = \sum_{l=0}^{N-1} g_l^{(2)} m_{i-l}$$

Then these bit sequences are multiplexed with the help of the commutator switch to produce the following output

$$X = \{ x_0^{(1)} x_0^{(2)} \; x_1^{(1)} x_1^{(2)} \; x_2^{(1)} x_2^{(2)} \cdots \}$$

1011
m_0 m_1 m_2 m_3

where $x_1 = x_0^{(1)} x_1^{(1)} x_2^{(1)} \cdots$

and $x_2 = x_0^{(2)} x_1^{(2)} x_2^{(2)} \cdots$

101
$m_0\ m_1\ m_2$ m_3 m

→ input sequence is $m_0 m_1 m_2 = 101$

$N = 3$ (No. of shift registers).

$$g_l^{(1)} = 101$$

$$g_l^{(2)} = 111.$$

$$x_i^{(1)} = \sum_{l=0}^{N-1} g_l^{(1)} \cdot m_{i-l} = \sum_{l=0}^{2} g_l^{(1)} m_{i-l}$$

$$x_0^{(1)} = \sum_{l=0}^{2} g_l^{(1)} m_{0-l} = g_0^{(1)} m_0 + g_1^{(1)} m_{-1} + g_2^{(1)} m_{-2}$$

$$= (1).(1) + 0(0) + 1(0) = 1$$

$$x_1^{(1)} = \sum_{l=0}^{2} g_l^{(1)} m_{1-l} = g_0^{(1)} m_1 + g_1^{(1)} m_0 + g_2^{(1)} m_{-1}$$

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

$$= 0$$

$$x_2^{(1)} = \sum_{l=0}^{2} g_l^{(1)} m_{2-l} = g_0^{(1)} m_2 + g_1^{(1)} m_1 + g_2^{(1)} m_0$$

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

$$= 0 \quad \text{(XOR operation).}$$

$$x_3^{(1)} = \sum_{l=0}^{2} g_l^{(1)} m_{3-l} = g_0^{(1)} m_3 + g_1^{(1)} m_2 + g_2^{(1)} m_1$$

$$= 1(0) + 0(1) + 1(0) = 0.$$

$$x_4^{(1)} = \sum_{l=0}^{2} g_l^{(1)} m_{4-l} = g_0^{(1)} m_4 + g_1^{(1)} m_3 + g_2^{(1)} m_2$$

$$= 1(0) + 0(0) + (1)$$

$$= 1$$

→

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

$$x_0^{(2)} = g_0^{(2)} m_0 + g_1^{(2)} m_{-1} + g_2^{(2)} m_{-2}$$

$$= 1(1) = 1$$

$$x_1^{(2)} = g_0^{(2)} m_1 + g_1^{(2)} m_0 + g_2^{(2)} m_{-1}$$

$$= 1(0) + 1(1)$$

$$= 1$$

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

$$= 1(1) + 1(0) + 1(1)$$

$$= 0$$

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

$$= 1(0) + 1(0) = 1$$

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

$$= 1(0) + 1$$

$\therefore \{x_i\} = \{x_0^{(1)} x_0^{(2)} \; x_1^{(1)} x_1^{(2)} \; x_2^{(1)} x_2^{(2)} \; x_3^{(1)} x_3^{(2)} \; x_4^{(1)} x_4^{(2)}\}$

$= \{11 \; 01 \; 00 \; 01 \; 11 \; 00\}$.

⇒ The convolutional encoder has the following two generator sequences each of length 3.

$\{g_0^{(1)}, g_1^{(1)}, g_2^{(1)}\} = (1, 1, 1)$

and $\{g_0^{(2)}, g_1^{(2)}, g_2^{(2)}\} = (1, 0, 1)$. determine the encoded sequence for the following input message.

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

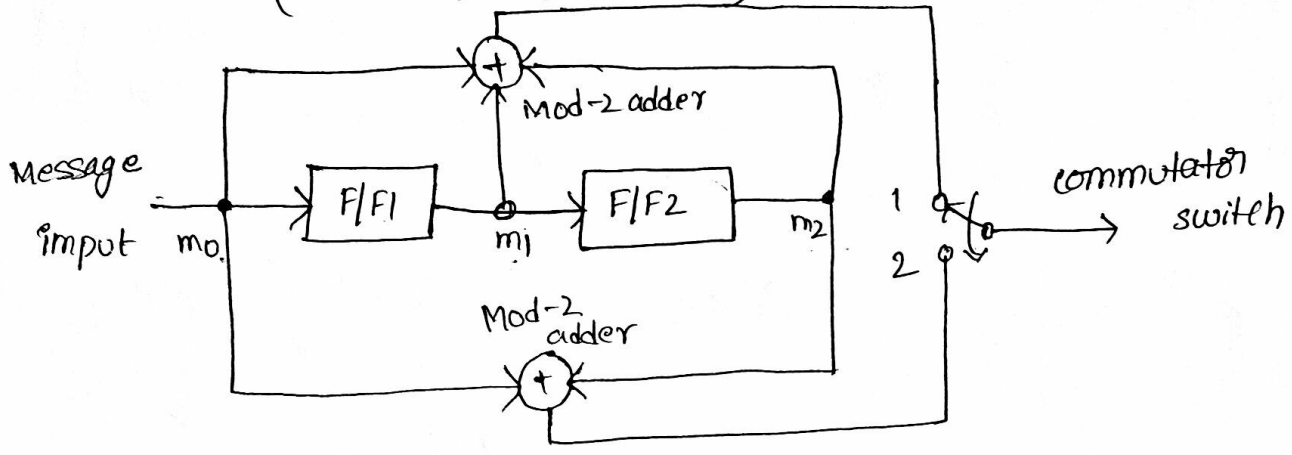
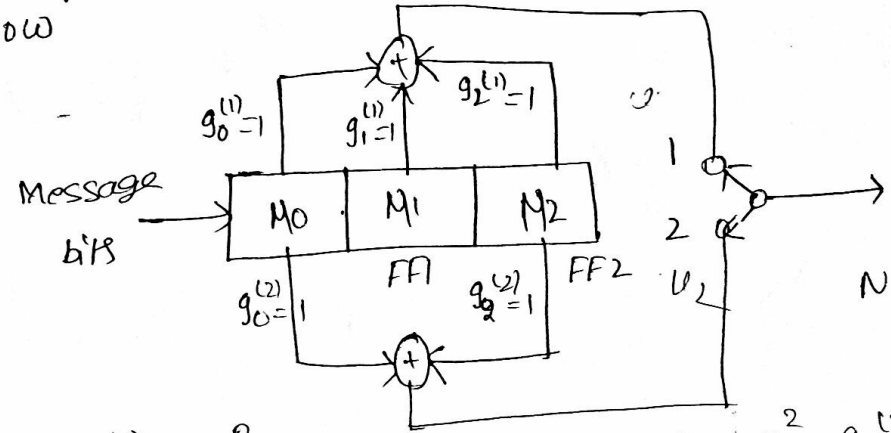


fig:- convolution encoder.

sol:- The given encoder can be drawn in standard form as shown below



$x_i^{(1)} = \sum_{l=0}^{2} g_l^{(1)} m_{i-l}$. $\quad x_i^{(2)} = \sum_{l=0}^{2} g_l^{(2)} m_{i-l}$.

Ans encoded sequence = $x_0^{(1)} x_0^{(2)} \; x_1^{(1)} x_1^{(2)} \; x_2^{(1)} x_2^{(2)} \; x_3^{(1)} x_3^{(2)} \; x_4^{(1)} x_4^{(2)} \; x_5^{(1)} x_5^{(2)} \; x_6^{(1)} x_6^{(2)}$

codeword = 11 01 00 01 11 00 11 length = $v(k+N) = 2(4+3) = 14$.

Transform Domain Approach:-

In this process, the first step is to replace each path in the encoder by a polynomial, whole coefficients are represented by the respective elements of the impulse response. for example $g_0^{(1)}=1$, $g_1^{(1)}=1$, $g_2^{(1)}=1$

∴ The input adder output path of the encoder can be expressed in terms of the polynomial as under.

$$g^{(1)}(D) = g_0^{(1)} + g_1^{(1)} D + g_2^{(1)} D^2$$

substituting the values we get

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

The General expression is given by

$$g^{(1)}(D) = g_0^{(1)} + g_1^{(1)} D + g_2^{(1)} D^2 + \cdots g_{N-1}^{(1)} D^{N-1}$$

Similarly $$g^{(2)}(D) = g_0^{(2)} + g_1^{(2)} D + g_2^{(2)} D^2 + \cdots g_{N-1}^{(2)} D^{N-1}.$$

where D = time delay of bits in impulse response.

The polynomials $g^{(1)}(D)$ & $g^{(2)}(D)$ are called as the "generator polynomials".

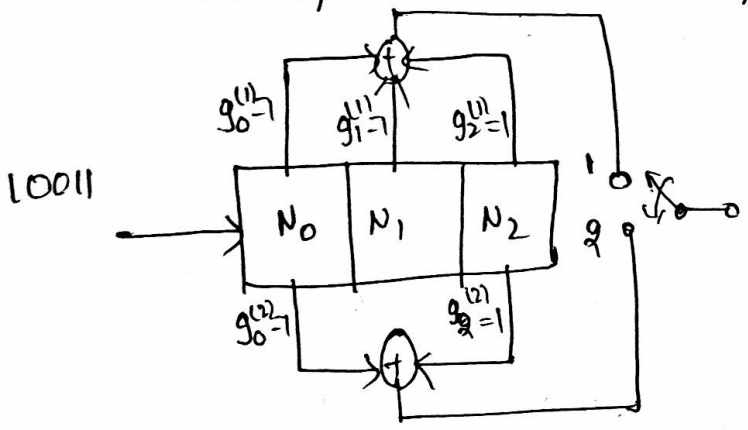
From the generator polynomial, we can obtain the codword polynomials.

$$x^{(1)}(D) = g^{(1)}(D)\, m(D)$$

$$x^{(2)}(D) = g^{(2)}(D)\, m(D).$$

if $m = 1101$ then $m(D) = 1+D+D^3$.

→ Determine the codeword for the cyclic encoder of the figure for the message signal (10011), using transform domain approach the impulse response of the input top adder output path is (1,1,1) and that of the input bottom adder output path is (1,0,1).



Sol:- Given $g_0^{(1)}=1$, $g_1^{(1)}=1$, $g_2^{(1)}=1$ & $g_0^{(2)}=1$, $g_1^{(2)}=0$, $g_2^{(2)}=1$.

$m = 10011$

$\therefore g^{(1)}(D) = g_0^{(1)} + g_1^{(1)} D + g_2^{(1)} D^2$

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

$g^{(2)}(D) = g_0^{(2)} + g_1^{(2)} D + g_2^{(2)} D^2$

$= 1 + D^2$

$m(D) = 1 + 0 \cdot D + 0 \cdot D^2 + 1 \cdot D^3 + 1 \cdot D^4$

$= 1 + D^3 + D^4$

$\Rightarrow x^{(1)}(D) = g^{(1)}(D)\, m(D) = (1+D+D^2)(1+D^3+D^4)$

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

$= 1 + D + D^2 + D^3 + D^6$.

corresponding code sequence is (1 1 1 1 0 0 1)

$$x^{(1)}(D) = g^{(1)}(D)\, m(D)$$

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

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

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

corresponding code sequence is (1011111)

final codeword at the output of the encoder is obtained by multiplexing the two code sequences.

codeword = 11 10 11 11 01 01 11

* It may be observed that we get the same result in both time domain & transform domain approaches, but, the computation using transform domain approach demands less effort than the time-domain approach.

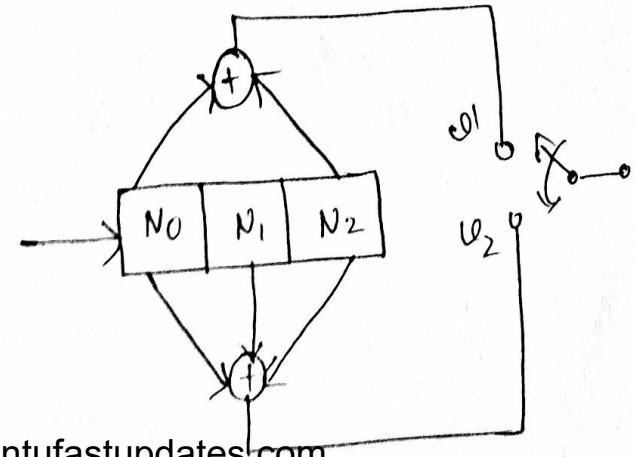
Graphical Representation of convolutional codes:-

For the convolutional encoding, there are three different graphical representations that are widely used. They are related to each other.

(1) State diagram
(2) Tree diagram
(3) Trellis diagram

State table

a = 00
c = 01
b = 10
d = 11

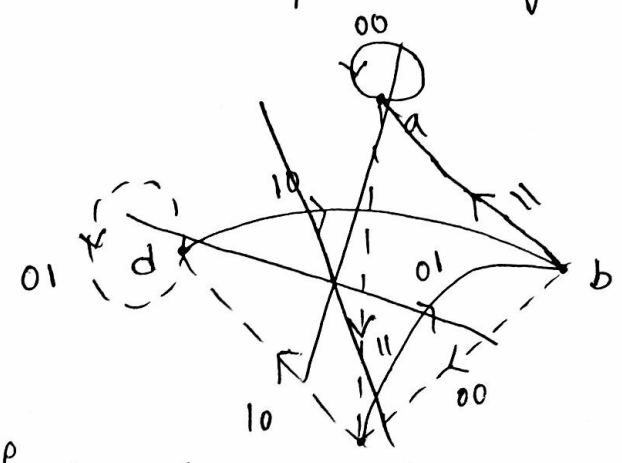


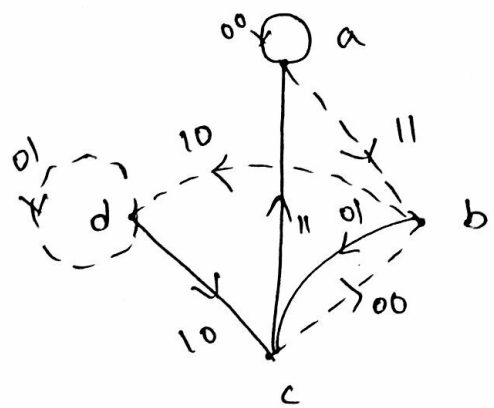
present state		Next bit	outputs		Next state	
			v_1	v_2		
N_1	N_2	N_0	$N_0 \oplus N_2$	$N_0 \oplus N_1 \oplus N_2$	N_1	N_2
0	0 (a)	0	0	0	0	0 (a)
0	0 (a)	1	1	1	1	0 (b)
0	1 (c)	0	1	1	0	0 (a)
0	1 (c)	1	0	0	1	0 (b)
1	0 (b)	0	0	1	0	1 (c)
1	0 (b)	1	1	0	1	1 (d)
1	1 (d)	0	1	0	0	1 (c)
1	1 (d)	1	0	1	1	1 (d)

State Diagram:-

'0' is represented by solid line ——,

'1' is represented by dotted line '- - - -'.

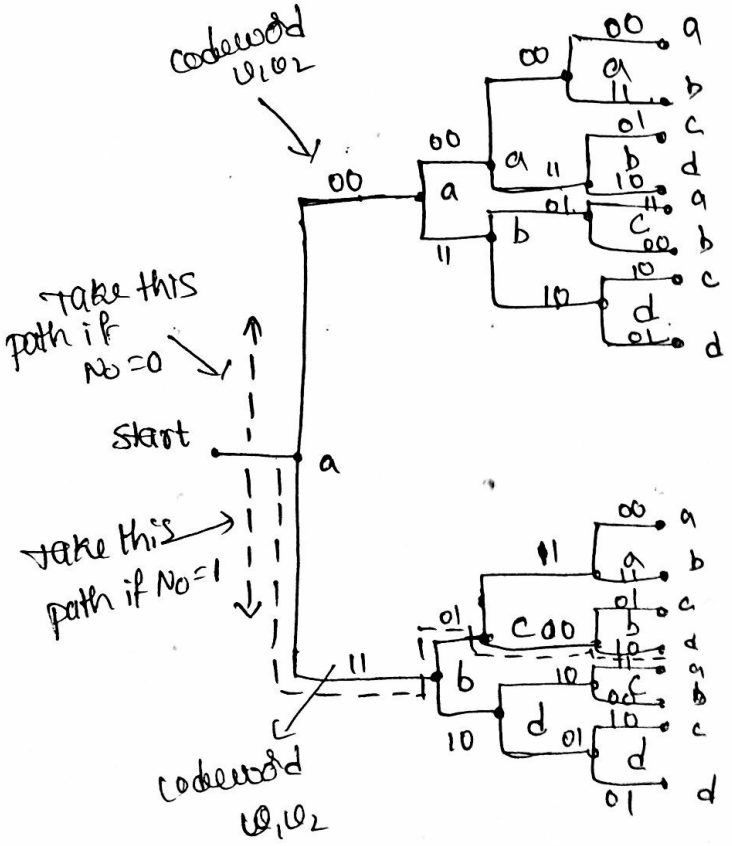




- If present state is 'a' and next bit is '0' then next state is also 'a' with output 00 [(v_1, v_2) from the above table] is represented by a solid line with output represented on it.
- If present state is 'a' and next bit is '1' then next state is 'b' with output 11 represented by dotted lines.
- Similarly the remaining states are represented as mentioned above.

Tree diagram:-

* Code Tree begins at a branch point on node 'a' which represents the initial state.
* If $N_0 = 0$ we should take the upper branch from node 'a' to obtain the output '00' and the next state "00" (a)
* If $N_0 = 1$, then we should take the lower branch from 'a' to obtain the output '11' and next state '01' (b).
* The code tree progresses in this manner for each new message bit. The nodes are labelled with letters a, b, c and d denoting the current state N_1N_2 and we go up or down from a node depending on the value of N_0.



if m = 1011

then codeword is.

11 01 00 10

fig:- Code Tree.

Trellis Diagram:-

More compact graphical representation which popularly known as code trellis. Here, the nodes on the left denote the four possible current states and the nodes on the right are the resulting next state.

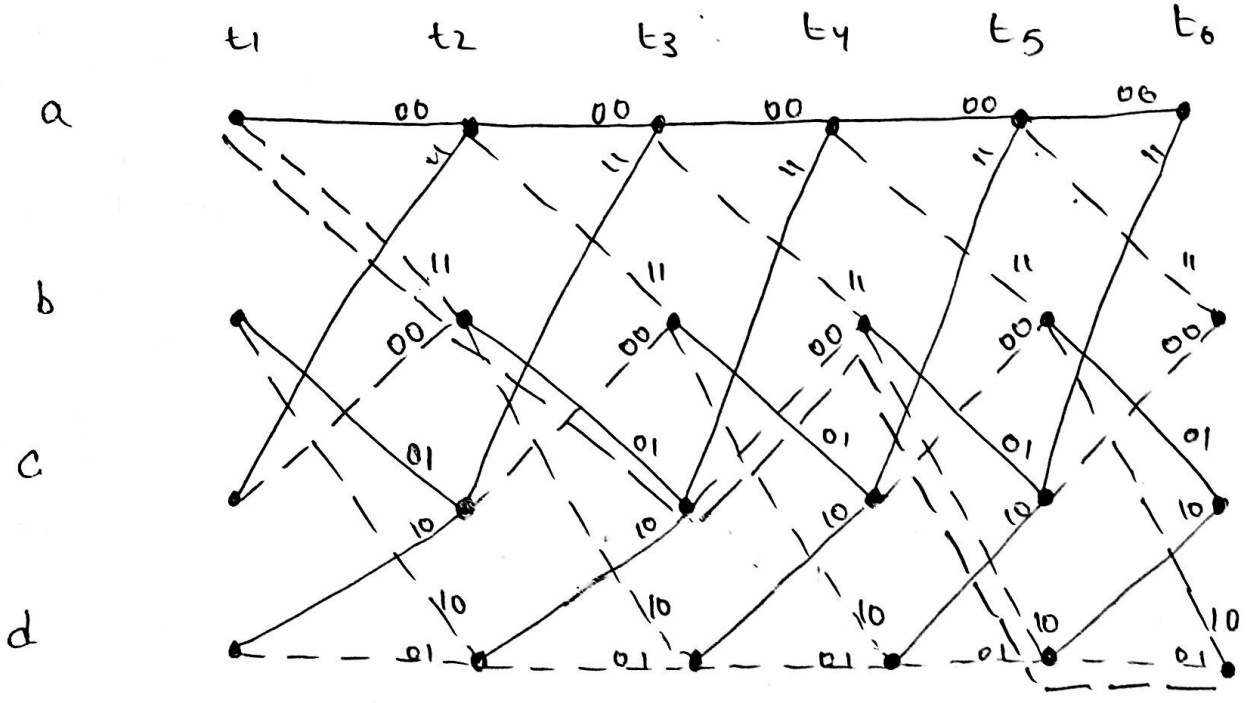
A solid line represents the state transition or branch for $m_0 = 0$ and the dotted line represents the branch for $m_0 = 1$. each branch is labelled with the resulting output bits $c_1 c_2$.

0 '——'

1 '- - -'.

States – horizontal

Time – vertical.



If $m = 10111$

convolution code = 11 01 00 10 01

Viterbi algorithm

To explain the viterbi algorithm for decoding of convolutional codes, it is necessary to define important terms.

Metric:-

It is defined as the hamming distance of each branch of each surviving path from the corresponding branch of y (received signal). The metric is defined by assuming that 0's and 1's have the same transmission error probability.

Surviving path:-

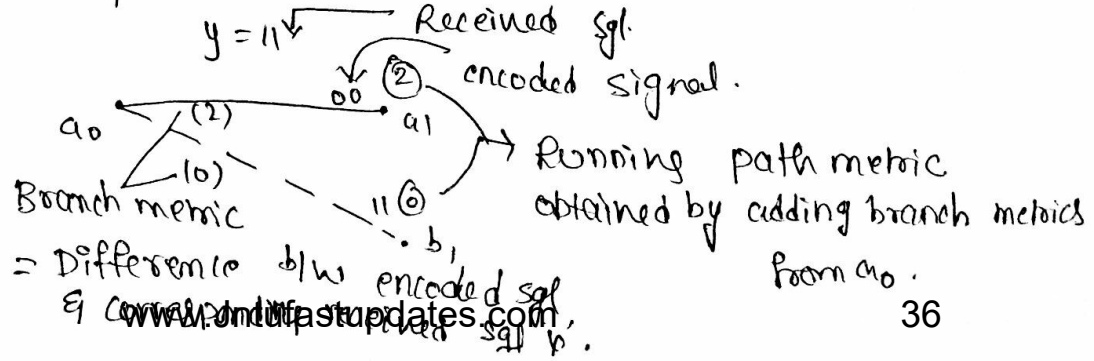
The surviving path is defined as the path of the decoded signal with minimum metric. By summing the branch metrics we get the path metric.

Example

Let us assume the received signal y = 11 01 11

At the current state 'a', the next state can be either a or b. depending on the message bit $m_0 = 0$ or 1. In the following example a_0 represents the initial state and a_1 and b_1 represent the next possible states.

The solid line represents the branch for $m_0 = 0$ and the dotted line represents the branch for $m_0 = 1$.



The number in the brackets, written below the branches represent the branch metric. For the branch a_0 to a_1, encoded signal is 00 and received signal is 11, hence branch metric is (2). where as for the path a_0b_1, the encoded signal and received signal both are 11, hence the branch metric is 00.

The encircled numbers at the right hand end of each branch represents the running path metric which is obtained by summing the branch metrices from a_0. The running path metric for the branch $a_0a_1 = 2$ and for the branch $a_0b_1 = 0$.

When the next part of input bits i.e y=01 are received at the nodes a_1 and b_1, then four possible branches emerge from these two nodes. The next four possible states are a_2, b_2, c_2 and d_2. The viterbi algorithm for all received bits is as shown in following figure.

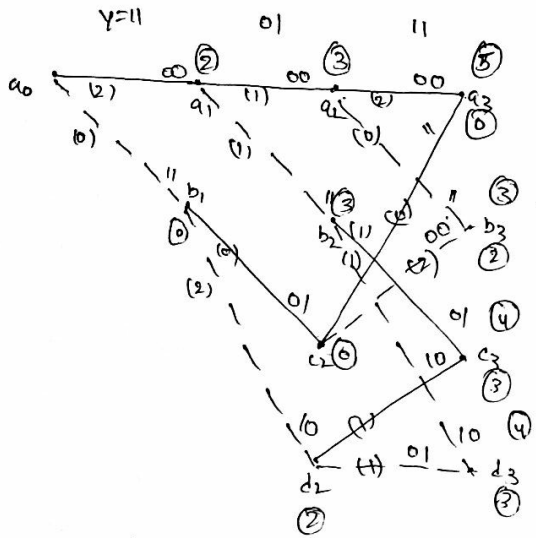


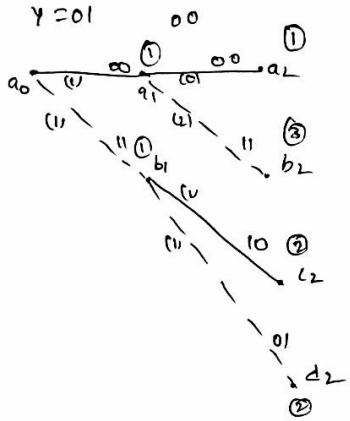
fig:- paths and their path metrices for the viterbi algorithm.

$\Rightarrow$ M = 00000

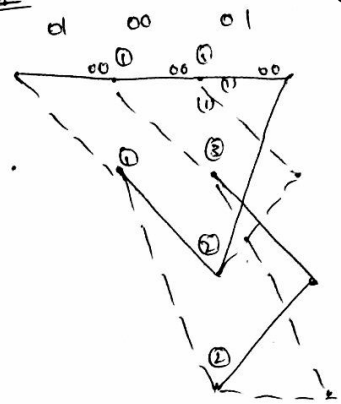
Tx = 00 00 00 00 00

Rx = 01 00 01 00 00

step 1:-



step 2:-



VII

Viterbi Algorithm:-

To explain the viterbi algorithm for the decoding of convolutional codes, it is necessary to define certain important terms

Metric:-

It is defined as the hamming distance of each branch of each surviving path from the corresponding branch of y (received signal). The metric is defined by assuming that 0's and 1's have the same transmission error probability.

Surviving path:-

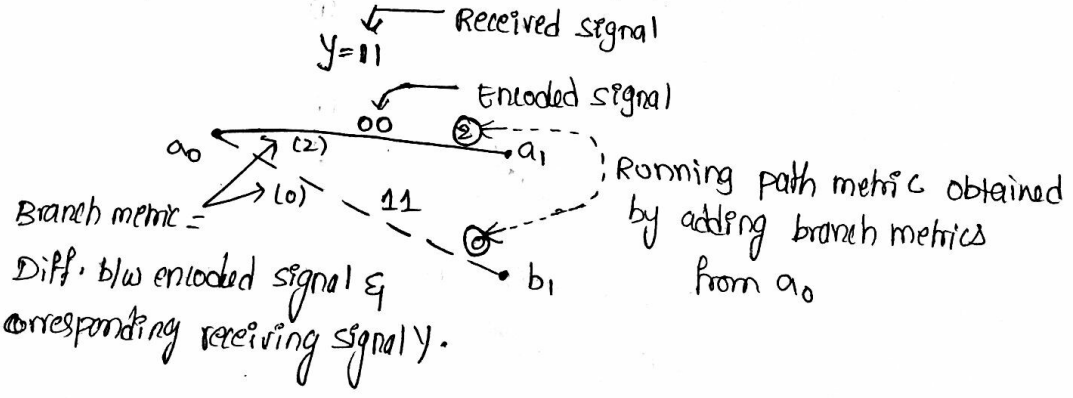
The surviving path is defined as the path of the decoded signal with minimum metric. By summing the branch metrices we get the path metric.

Example:-

Let us assume the received signal y = 11 01 11

for the current state a, the next state can be either a & b depending on the message bit $m_0 = 0$ & 1. in the following diagram a_0 represents the initial state and a_1 and b_1 represent the next possible states.

The solid line represents the branch for $m_0 = 0$ and the dotted line represents the branch for $m_0 = 1$.



The number in the brackets, written below the branches represent the branch metric. For the branch a_0 to a_1, encoded signal is 00 and received signal is 11, hence the branch metric is (2) where as for the path a_0b_1, the encoded signal and received signal both are 11, hence the branch metric is (0).

The encircled numbers at the right hand end of each branch represents the running path metric which is obtained by summing the branch metrices from a_0. The running path metric for the branch $a_0a_1 = 2$ and for the branch $a_0b_1 = 0$.

When the next part of input bits i.e y=01 are received at the nodes a_1 and b_1, then four possible branches emerge from these two nodes. The next four possible states are a_2, b_2, c_2 and d_2. The Viterbi algorithm for all received bits is as shown in following figure.

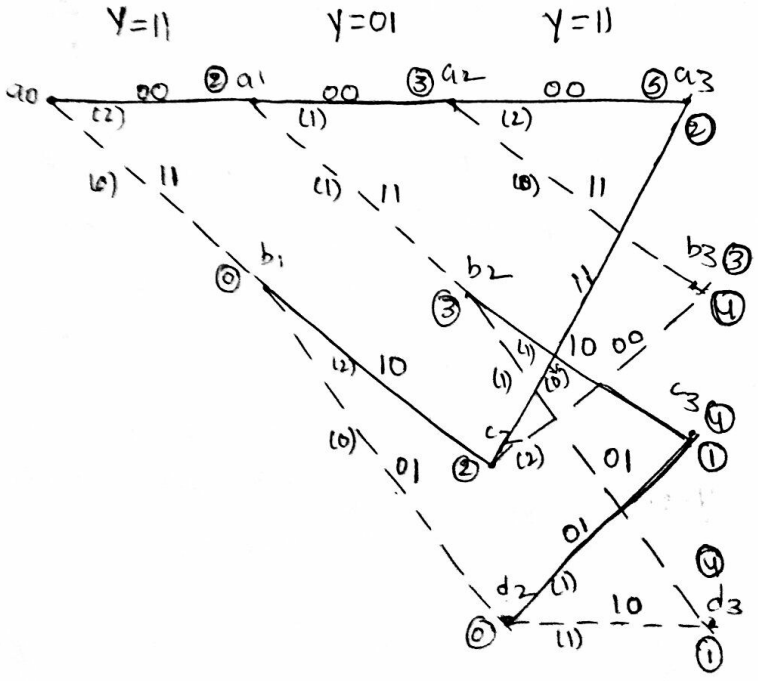


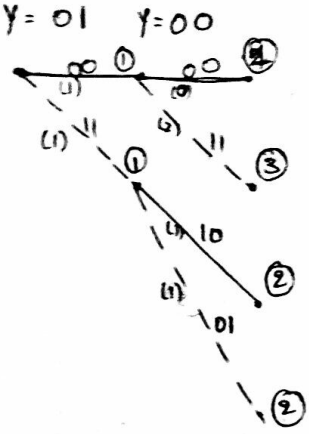
Fig: paths and their path metrices for the Viterbi algorithm.

→ M = 00000

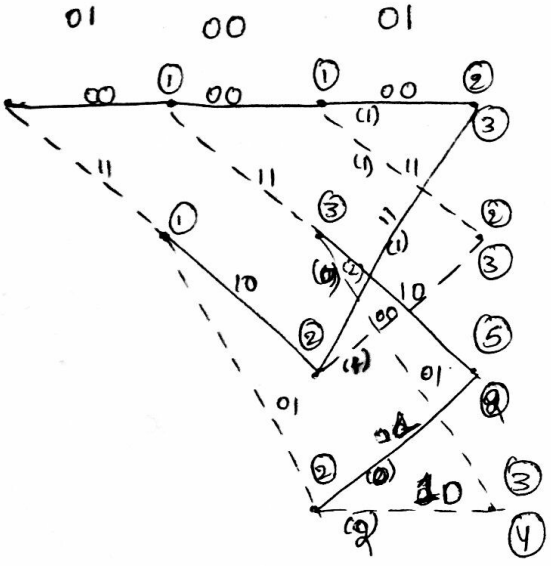
Tx = 00 00 00 00 00

Px = 01 00 01 00 00

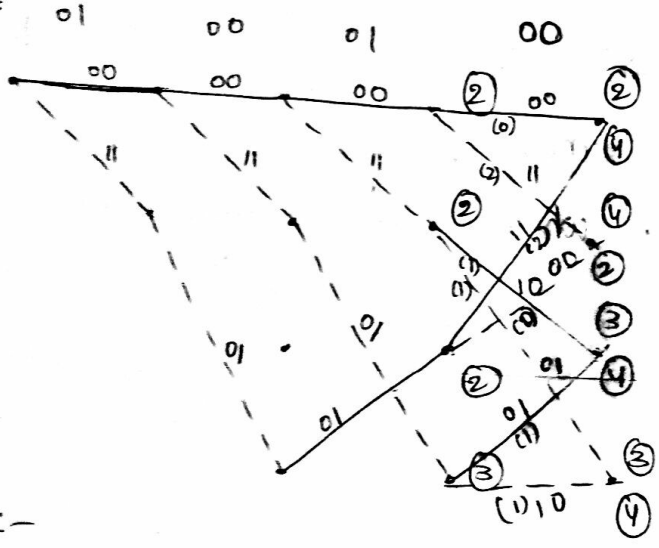
Step 1:-



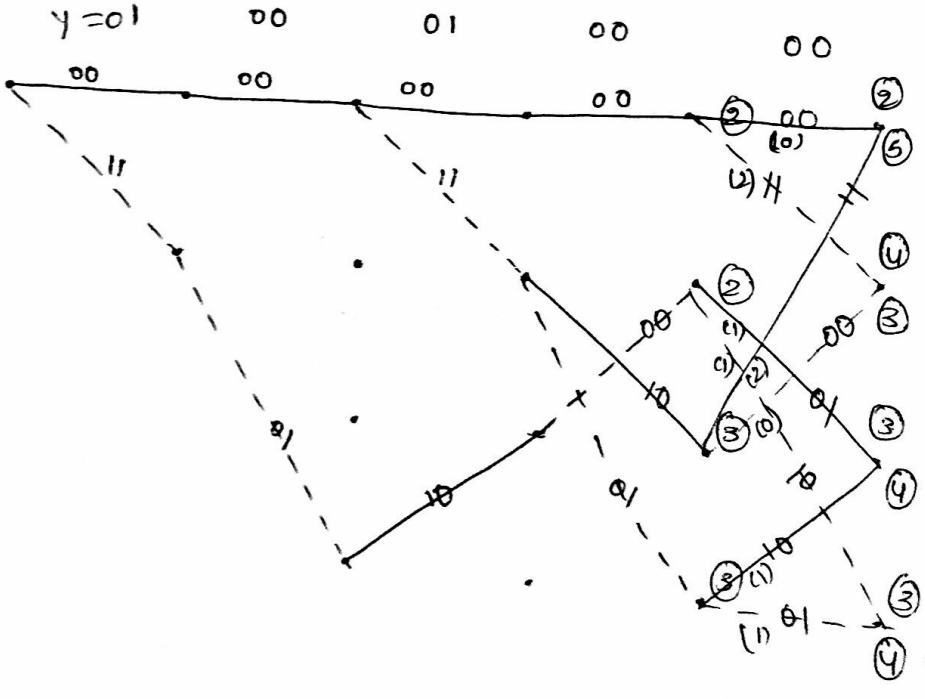
Step 2:

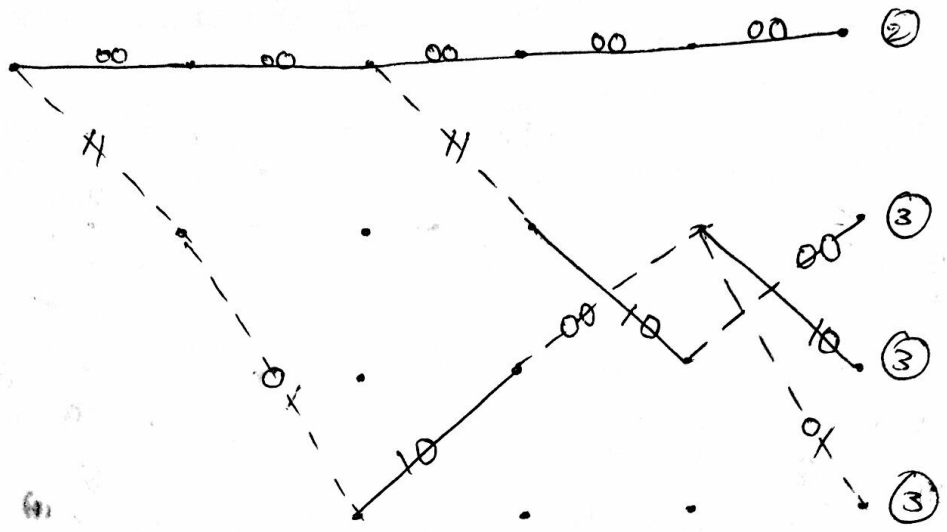


Step 3:-



tep 4:-





In the above figure we see that the all zero path has the smallest metric. This clearly shows that the all zeros sequence is the maximum likely hood choice of the viterbi decoding algorithm. which agrees exactly with transmi' sequence.