1. Consider a systematic block code whose parity check equation are

$$P_1 = m_1 + m_2 + m_4$$
$$P_2 = m_1 + m_3 + m_4$$
$$P_3 = m_1 + m_2 + m_3$$
$$P_4 = m_2 + m_3 + m_4$$

where m_i is the message digits and P_i are the parity digits.

1. Find the generator matrix and the parity check matrix for this code.
2. How many errors can be detected and corrected?
3. If the received code word is 10101010, find the syndrome.

SOLUTION

I To obtain the generator matrix and parity check matrix

* 4 parity check equations
  - $\Rightarrow q = 4$
* message bits in equation $\Rightarrow$ m_1, m_2, m_3, m_4
  - $\Rightarrow k = 4$ $\Rightarrow n = k + q = 4 + 4 = 8$ bits.

$$C = mP \Rightarrow P = mP.$$

$$[P]_{1\times 4} = [m]_{1\times 4}[P]_{k\times q}$$

$$\Rightarrow [P]_{1\times 4} = [m]_{1\times 4}[P]_{4\times 4}.$$

$$[p_1\ p_2\ p_3\ p_4] = [m_1\ m_2\ m_3\ m_4]\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$$

$$= \begin{bmatrix} m_1p_{11} \oplus m_2p_{21} \oplus m_3p_{31} \oplus m_4p_{41} & m_1p_{12} \oplus m_2p_{22} \oplus m_3p_{32} \oplus m_4p_{42} & m_1p_{13} \oplus m_2p_{23} \oplus m_3p_{33} \oplus m_4p_{43} & m_1p_{14} \oplus m_2p_{24} \oplus m_3p_{34} \oplus m_4p_{44} \end{bmatrix}$$

$$\Rightarrow p_1 = m_1p_{11} \oplus m_2p_{21} \oplus m_3p_{31} \oplus m_4p_{41} \Leftrightarrow p_1 = m_1 + m_2 + m_4$$

$$p_2 = m_1p_{12} \oplus m_2p_{22} \oplus m_3p_{32} \oplus m_4p_{42} \Leftrightarrow p_2 = m_1 + m_3 + m_4$$

$$p_3 = p_{13}m_1 \oplus m_2p_{23} \oplus m_3p_{33} \oplus m_4p_{43} \Leftrightarrow p_3 = m_1 + m_2 + m_3$$

$$p_4 = m_1p_{14} \oplus m_2p_{24} \oplus m_3p_{34} \oplus m_4p_{44} \Leftrightarrow p_4 = m_2 + m_3 + m_4$$

Comparing the equations the parity matrix is given as

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

⇒ Generator Matrix

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

$$= [I_4 \vdots P_{4 \times 4}]$$

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

⇒ Parity check matrix

$$H = [P^T \vdots I_k]$$

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

II To obtain code vectors

Message				Parity				Code vector							
m_1	m_2	m_3	m_4	c_1	c_2	c_3	c_4	m_1	m_2	m_3	m_4	c_1	c_2	c_3	c_4
0	0	0	0												
0	0	0	1												
0	0	1	0												

$\Rightarrow d_{min} = 4$

Error detection $\boxed{d_{min} \geq s+1}$

$4 \geq s+1 \Rightarrow \boxed{s \leq 3} \Rightarrow$ 3 errors will be detected

Error correction $\boxed{d_{min} \geq 2t+1}$

$4 \geq 2t+1$

$2t \leq 3$

$t \leq \frac{3}{2} \Rightarrow \boxed{t \leq 1.5} \Rightarrow$ one error can be corrected

III To prepare decoding table

S.No.	Error vector (E)	Syndrome (S)
1.	0 0 0 0 0 0 0 0	0 0 0 0
2.	1 0 0 0 0 0 0 0	1 1 0 1
3.	0 1 0 0 0 0 0 0	1 0 1 1
4.	0 0 1 0 0 0 0 0	1 1 1 0
5.	0 0 0 1 0 0 0 0	0 1 1 1
6.	0 0 0 0 1 0 0 0	1 0 0 0
7.	0 0 0 0 0 1 0 0	0 1 0 0
8.	0 0 0 0 0 0 1 0	0 0 1 0
9.	0 0 0 0 0 0 0 1	0 0 0 1

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

$$S = YH^T$$

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

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

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

* Since this syndrome does not correspond to any row of H^T, there are more than 'one' error that cannot be corrected.

4.13.1 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. They are discussed next.

4.13.2 Properties of Cyclic Codes

As defined above, cyclic codes exhibit two fundamental properties :

1. Linearity and 2. Cyclic property.

4.13.2.1 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.

4.13.2.2 Cyclic Property

Every 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 :

$$X = \{x_{n-1}, x_{n-2}, \ldots\ldots, x_1, x_0\} \quad \ldots (4.13.1)$$

Here $x_{n-1}, x_{n-2}, \ldots, x_1, x_0$ 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, $X' = (x_{n-2}, x_{n-3}, \ldots, x_1, x_0, x_{n-1})$ ... (4.13.2)

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.,

$$X'' = (x_{n-3}, x_{n-4}, \ldots, x_1, x_0, x_{n-1}, x_{n-2}) \quad \ldots (4.13.3)$$

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

4.13.3 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., $$X(p) = x_{n-1}\, p^{n-1} + x_{n-2}\, p^{n-2} + \ldots + x_1\, p + x_0 \quad \ldots (4.13.4)$$

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

p is the arbitrary variable of the polynomial.

The power of 'p' represent 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.

4.13.3.1 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,

$$M(p) = m_{k-1}\, p^{k-1} + m_{k-2}\, p^{k-2} + \ldots + m_1\, p + m_0 \quad \ldots (4.13.5)$$

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

$$\boxed{X(p) = M(p)\, G(p)} \quad \ldots (4.13.6)$$

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,

$$G(p) = p^q + g_{q-1}\, p^{q-1} + \ldots + g_1\, p + 1 \quad \ldots (4.13.7)$$

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,

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

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

$$X_3(p) = M_3(p)\, G(p) \text{ and so on} \quad \ldots (4.13.8)$$

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

Ex. 4.13.1 *The generator polynomial of a (7, 4) cyclic code is* $G(p) = p^3 + p + 1$.

Find all the code vectors for the code in nonsystematic form. AU : May-16, Marks 8

Sol. : Here $n = 7$ and $k = 4$ therefore $q = n - k = 3$.

There will be total $2^k = 2^4 = 16$ message vectors of 7 bits each. Consider any message vector as,

$$M = (m_3\ m_2\ m_1\ m_0) = (0\ 1\ 0\ 1)$$

Then the message polynomial will be ($k = 4$ in equation 4.13.5),

$$M(p) = m_3\ p^3 + m_2\ p^2 + m_1 p + m_0$$

$$\therefore \quad M(p) = p^2 + 1 \qquad \ldots (1)$$

And given generator polynomial is,

$$G(p) = p^3 + p + 1 \qquad \ldots (2)$$

To obtain nonsystematic code vectors

The nonsystematic cyclic code is given by equation (4.13.6) as,

$$X(p) = M(p)\ G(p) = \left(p^2 + 1\right)\left(p^3 + p + 1\right)$$

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

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

$$= p^5 + p^2 + p + 1 \qquad (\text{Since } (1 \oplus 1)\, p^3 = 0p^3 = 0)$$

$$= 0p^6 + p^5 + 0p^4 + 0p^3 + p^2 + p + 1$$

Note that the degree of above polynomial is $n - 1 = 6$. The code vector corresponding to above polynomial is,

$$X = (x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0) = (0\ 1\ 0\ 0\ 1\ 1\ 1)$$

This is the code vector for message vector 0101. This code vector is non systematic cyclic code vector. Similarly other code vectors can be obtained using the same procedure. Table 4.13.1 lists the code vectors in nonsystematic form.

Sr.	Message bits				Nonsystematic code vectors						
No.	$M = m_3$	m_2	m_1	m_0	$X = x_6$	x_5	x_4	x_3	x_2	x_1	x_0
∴	0	0	0	0	0	0	0	0	0	0	0

2.	0	0	0	1	0	0	0	1	0	1	1
3.	0	0	1	0	0	0	1	0	1	1	0
4.	0	0	1	1	0	0	1	1	1	0	1
5.	0	1	0	0	0	1	0	1	1	0	0
6.	0	1	0	1	0	1	0	0	1	1	1
7.	0	1	1	0	0	1	1	1	0	1	0
8.	0	1	1	1	0	1	1	0	0	0	1
9.	1	0	0	0	1	0	1	1	0	0	0
10.	1	0	0	1	1	0	1	0	0	1	1
11.	1	0	1	0	1	0	0	1	1	1	0
12.	1	0	1	1	1	0	0	0	1	0	1
13.	1	1	0	0	1	1	1	0	1	0	0
14.	1	1	0	1	1	1	1	1	1	1	1
15.	1	1	1	0	1	1	0	0	0	1	0
16.	1	1	1	1	1	1	0	1	0	0	1

Table 4.13.1 Code vectors of a (7, 4) cyclic code for $G(p) = p^3 + p + 1$

To check whether cyclic property is satisfied :

Let us consider code vector X_9 which is given in above table as,

$$X_9 = (1011000)$$

Let us shift this code vector cyclically to left side by 1 bit position. Then we get,

$$X' = 0\ 1\ 1\ 0\ 0\ 0\ 1$$

From table, observe that

$$X' = X_8 = (0110001)$$

Thus cyclic shift of X_9 produces X_8. This can be varified for other code vectors also.

4.13.3.2 Generation of Code Vectors in Systematic Form

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

$$X = (k \text{ message bits} : (n-k) \text{ check bits}) \quad \ldots (4.13.9)$$

$$= \left(m_{k-1}\, m_{k-2} \ldots m_1\, m_0 : c_{q-1}\, c_{q-2} \ldots c_1\, c_0\right) \quad \ldots (4.13.10)$$

Here the check bits form a polynomial as,

$$C(p) = c_{q-1}\, p^{q-1} + c_{q-2}\, p^{q-2} + \ldots c_1\, p + c_0 \quad \ldots (4.13.11)$$

The check bit polynomial is obtained by,

LINEAR BLOCK CODE - TYPE II

* For every block code there is a $q \times n$ parity check matrix $[H]$. It is defined as

$$H = [P^T : I_q]_{q \times n}$$

Hamming codes

* Hamming codes are (n,k) linear block codes that 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 $d_{min} = 3$

Code rate $r = \frac{k}{n}$ $\Rightarrow r = 1 - \frac{q}{n}$

1. The Parity check matrix of a particular (7,4) linear block code is given by

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

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 and corrected?.

SOLUTION

Given :- $(n,k) = (7,4)$

$\Rightarrow n = 7$, $k = 4$, $q = 7-4 = 3$

I To determine P sub matrix

We know that

$$H = [P^T : I_q]_{q \times n}$$

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

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

II To Generate G matrix

We know that

$$G = \left[I_k \mid P_{k \times q} \right]$$

$k = 4 \Rightarrow$

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

III To Generate equations for check matrix

$$\boxed{C = mP}$$

$$[C]_{1 \times q} = [m]_{1 \times k} [P]_{k \times q}$$

$$[C]_{1 \times 3} = [m]_{1 \times 4} [P]_{4 \times 3}$$

$$\begin{bmatrix} C_0 & C_1 & C_2 \end{bmatrix} = \begin{bmatrix} m_0 & m_1 & m_2 & m_3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} m_0 \oplus m_1 \oplus m_2 & m_0 \oplus m_1 \oplus m_3 & m_0 \oplus m_2 \oplus m_3 \end{bmatrix}$$

$$\Rightarrow C_0 = m_0 \oplus m_1 \oplus m_2$$

$$C_1 = m_0 \oplus m_1 \oplus m_3$$

$$C_2 = m_0 \oplus m_2 \oplus m_3$$

IV To determine code vectors

Message				check Matrix			Code Vector	d
m_0	m_1	m_2	m_3	$C_0 = m_0 \oplus m_1 \oplus m_2$	$C_1 = m_0 \oplus m_1 \oplus m_3$	$C_2 = m_0 \oplus m_2 \oplus m_3$	$m_0 m_1 m_2 m_3 C_0 C_1 C_2$	w_m
0	0	0	0	0	0	0		0
0	0	0	1	0	1	1		3
0	0	1	0	1	0	1		3
0	0	1	1	1	1	0		4
0	1	0	0	1	1	0		3
0	1	0	1	1	0	1		4
0	1	1	0	0	1	1		4
0	1	1	1	0	0	0		3
1	0	0	0	1	1	1		4
1	0	0	1	1	0	0		3
1	0	1	0	0	1	0		3
1	0	1	1	0	0	1		4
1	1	0	0	0	0	1		3
1	1	0	1	0	1	0		4
1	1	1	0	1	0	0		4
1	1	1	1	1	1	1		7

IV Minimum distance between code vectors

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

$$\Rightarrow \boxed{d_{min} = 3}$$

V Error detection & correction capabilities

Error detection (s) $\Rightarrow \boxed{d_{min} \geq s+1}$ $\quad 3 \geq s+1 \Rightarrow \boxed{s \leq 2}$ 2 errors detected

Error correction (t) $\Rightarrow \boxed{d_{min} \geq 2t+1}$ $\quad 3 \geq 2t+1 \Rightarrow \boxed{t \leq 1}$ 1 error will be corrected.

TYPE III

SYNDROME DECODING

* When some errors are present in received vector Y, YH^T is non-zero.
* This means some errors are present in Y.
* The non-zero output of the product YH^T is called syndrome and it's used to detect the errors in Y.
* Syndrome 'S' is represented as

$$S = YH^T$$

$$[S]_{1xq} = [Y]_{1xn} [H^T]_{nxq}$$

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

$$S = EH^T$$

* This relationship shows that syndrome depends upon the error pattern only.
* It doesn't depend upon a particular message

1. The parity check matrix of a (7,4) hamming code is given as follows

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

Calculate the syndrome vector for single bit errors. Detect error in the

$Y = 10\phi 1110$

SOLUTION

GIVEN $(n,k) = (7,4)$

$n=7,\ k=4,\ q=n-k=7-4=3$

I To determine error pattern for single bit errors

* $q=3 \Rightarrow$ syndrome vector is 3 bits.
* No. of non-zero syndromes $= 2^q - 1 = 2^3 - 1 = 7$ non-zero syndromes

S.No.	Bit in error	single bit error pattern
1.	1	1 0 0 0 0 0 0
2.	2	0 1 0 0 0 0 0
3.	3	0 0 1 0 0 0 0
4.	4	0 0 0 1 0 0 0
5.	5	0 0 0 0 1 0 0
6.	6	0 0 0 0 0 1 0
7.	7	0 0 0 0 0 0 1

II Calculation of syndromes

* Syndrome vector is given by

$$S = EH^T$$

$$[S]_{1\times q} = [E]_{1\times n} \; H^T$$

$$[S_0 S_1 S_2] = [e_0\ e_1\ e_2\ e_3\ e_4\ e_5\ e_6] \begin{bmatrix} 1&0&1\\1&1&1\\1&1&0\\0&1&1\\1&0&0\\0&1&0\\0&0&1 \end{bmatrix}$$

syndrome for I error pattern

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

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

Note:-
syndrome vectors are rows of H^T

S.No.	Error vector	S: Syndrome
1.	0000000	000
2.	1000000	101
3.	0100000	111
4.	0010000	110
5.	0001000	011
6.	0000100	100
7.	0000010	010
8.	0000001	001

syndrome

000
101
111
110
011
100
010
001

III Error correction using syndrome

Algorithm :-

1. Calculate syndrome $S = YH^T$
2. Check the row of H^T which is same as 'S'.
   b [syndrome of error pattern]
3. For the p^{th} row of H^T, p^{th} bit is in error.
4. Obtain correct vector by $X = Y \oplus E$

I To obtain syndrome vector of received code vector

GIVEN $Y = 10\phi 1110$.

$$S = YH^T$$

$$= [10\phi 1110] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

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

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

$$= [1 1 0]$$

⇒ compare with syndrome table.

* 3rd bit in error.

⇒ ~~$S = EH^T$~~ $X = Y \oplus E$

$$= [1011110] \oplus [0010000]$$

$$= [1001110]$$

4.13.3.2 Generation of Code Vectors in Systematic Form

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

$$X = (k \text{ message bits} : (n-k) \text{ check bits}) \quad \ldots (4.13.9)$$

$$= \left(m_{k-1}\, m_{k-2} \ldots m_1\, m_0 : c_{q-1}\, c_{q-2} \ldots c_1\, c_0\right) \quad \ldots (4.13.10)$$

Here the check bits form a polynomial as,

$$C(p) = c_{q-1}\, p^{q-1} + c_{q-2}\, p^{q-2} + \ldots c_1\, p + c_0 \quad \ldots (4.13.11)$$

The check bit polynomial is obtained by,

$$C(p) = rem\left[\frac{p^q\ M(p)}{G(p)}\right] \qquad \text{... (4.13.12)}$$

Above equation means -

i) Multiply message polynomial by p^q.

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

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

Ex. 4.13.2 *The generator polynomial of a (7, 4) cyclic code is $G(p) = p^3 + p + 1$. Find all the code vectors for the code in systematic form.*

Sol. : Here $n = 7$ and $k = 4$ therefore $q = n - k = 3$.

There will be total $2^k = 2^4 = 16$ message vectors of 7 bits each. Consider any message vector as,

$$M = (m_3\ m_2\ m_1\ m_0) = (0\ 1\ 0\ 1)$$

Then the message polynomial will be ($k = 4$ in equation 4.13.5),

$$M(p) = m_3\ p^3 + m_2\ p^2 + m_1 p + m_0$$

$$\therefore \quad M(p) = p^2 + 1 \qquad \text{... (1)}$$

And given generator polynomial is,

$$G(p) = p^3 + p + 1 \qquad \text{... (2)}$$

To obtain $p^q\ M(p)$:

Since $q = 3$, $p^q\ M(p)$ will be,

$$p^q\ M(p) = p^3\ M(p) = p^3(p^2 + 1)$$

$$= p^5 + q^3 \qquad \text{(For message vector of (101)}$$

$$= p^5 + 0p^4 + p^3 + 0p^2 + 0p + 0$$

And $\quad G(p) = p^3 + p + 1 = p^3 + 0p^2 + p + 1$

To perform the division $\dfrac{p^q\ M(p)}{G(p)}$:

We now have $p^q\ M(p)$ and $G(p)$. Now let's perform the division to find remainder of this division.

$$\begin{array}{r} p^2+0+0 \quad \leftarrow \text{Quotient} \\ p^3+0p^2+p+1 \overline{)\, p^5+0p^4+p^3+0p^2+0p+0} \\ p^5+0p^4+p^3+p^2 \\ \oplus \quad \oplus \quad \oplus \quad \oplus \\ \hline 0 \quad +0 \quad +0 \quad +\underbrace{p^2+0p+0}_{\text{Remainder}} \end{array}$$

Mod-2 addition

Thus the remainder polynomial is p^2+0p+0 in the division of $p^3\ M(p)$ by $G(p)$. Therefore equation (4.13.12) can be written as,

$$C(p) = rem\left[\frac{p^3\ M(p)}{G(p)}\right] = p^2+0p+0$$

With $q=3$ the polynomial $C(p)$ from equation (4.13.11) is,

$$C(p) = c_2p^2 + c_1p + c_0$$

Thus $$c_2p^2 + c_1p + c_0 = p^2+0p+0$$

Therefore the check bits are

$$C = (c_2c_1c_0) = (1\ 0\ 0)$$

The code vector is written in system form as given by equation (4.13.10) i.e.,

$$X = \left(m_{k-1}m_{k-2}\cdots m_1m_0 : c_{q-1}c_{q-2}\cdots c_1c_0\right)$$

In this example $$X = (m_3m_2m_1m_0 : c_2c_1c_0) = (0\ 1\ 0\ 1 : 1\ 0\ 0)$$

This is the required cyclic code vectors in systematic form. The other code vectors can be obtained using the same procedure.

Table 4.13.2 lists all the systematic cyclic codes.

Sr. No.	Message bits $M=$	m_3	m_2	m_1	m_0	Systematic code vectors $X=$	m_3	m_2	m_1	m_0	c_2	c_1	c_0
1.		0	0	0	0		0	0	0	0	0	0	0
2.		0	0	0	1		0	0	0	1	0	1	1
3.		0	0	1	0		0	0	1	0	1	1	0
4.		0	0	1	1		0	0	1	1	1	0	1
5.		0	1	0	0		0	1	0	0	1	1	1
6.		0	1	0	1		0	1	0	1	1	0	0
7.		0	1	1	0		0	1	1	0	0	0	1

8.	0	1	1	1	0	1	1	1	0	1	0
9.	1	0	0	0	1	0	0	0	1	0	1
10.	1	0	0	1	1	0	0	1	1	1	0
11.	1	0	1	0	1	0	1	0	0	1	1
12.	1	0	1	1	1	0	1	1	0	0	0
13.	1	1	0	0	1	1	0	0	0	1	0
14.	1	1	0	1	1	1	0	1	0	0	1
15.	1	1	1	0	1	1	1	0	1	0	0
16.	1	1	1	1	1	1	1	1	1	1	1

Table 4.13.2 Code vectors of a (7, 4) cyclic code for $G(p) = p^3 + p + 1$

We have obtained nonsystematic code vectors for the same generating polynomial in Example 4.13.1. They are listed in Table 4.13.1.

4.13.4 Encoding using an $(n - k)$ Bit Shift Register

In this section we will discuss the encoders for systematic cyclic codes. Fig. 4.13.2 shows the block diagram of a generalized (n, k) cyclic code. The symbols used to draw encoders are shown in Fig. 4.13.1.

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

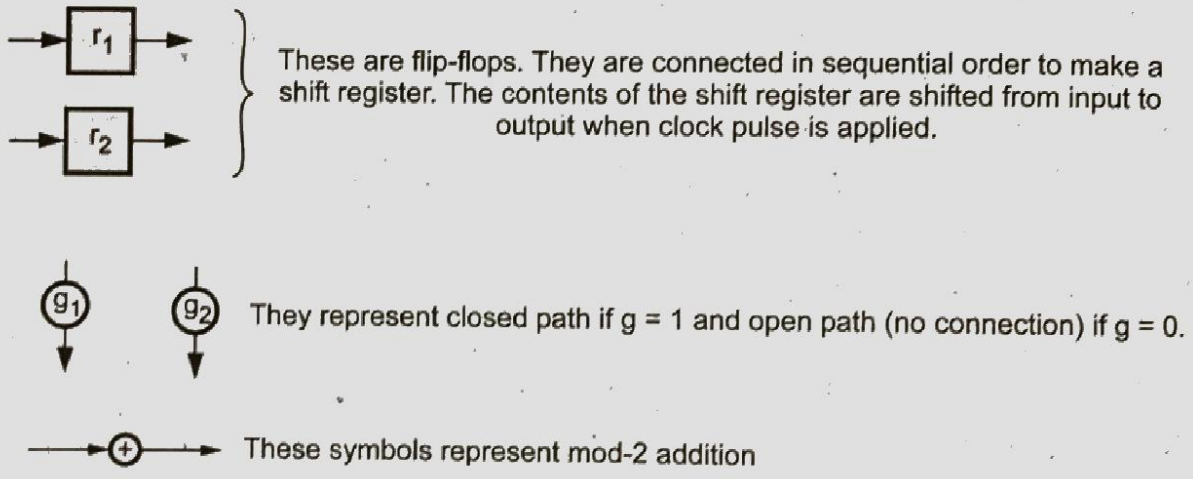


Fig. 4.13.1 Various symbols used in encoder

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.

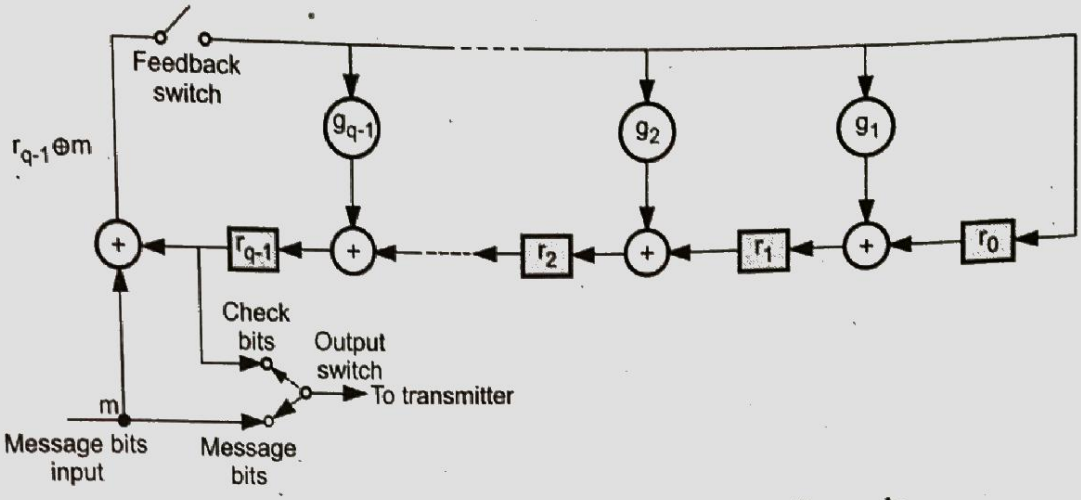


Fig. 4.13.2 Encoder for systematic (*n*, *k*) cyclic code

Ex. 4.13.3 *Design the encoder for the (7, 4) cyclic code generated by* $G(p) = p^3 + p + 1$ *and verify its operation for the received vector 1100.* **AU : Dec.-15, Marks 8**

Sol. : The generator polynomial is,

$$G(p) = p^3 + 0\,p^2 + p + 1$$

and

$$G(p) = p^3 + g_2\,p^2 + g_1\,p + 1$$

On comparison of the two equation we obtain,

$$g_1 = 1 \quad \text{and} \quad g_2 = 0$$

and

$$q = n - k = 7 - 4 = 3$$

With these values the block diagram of Fig. 4.13.2 will be as shown in Fig. 4.13.3 below.

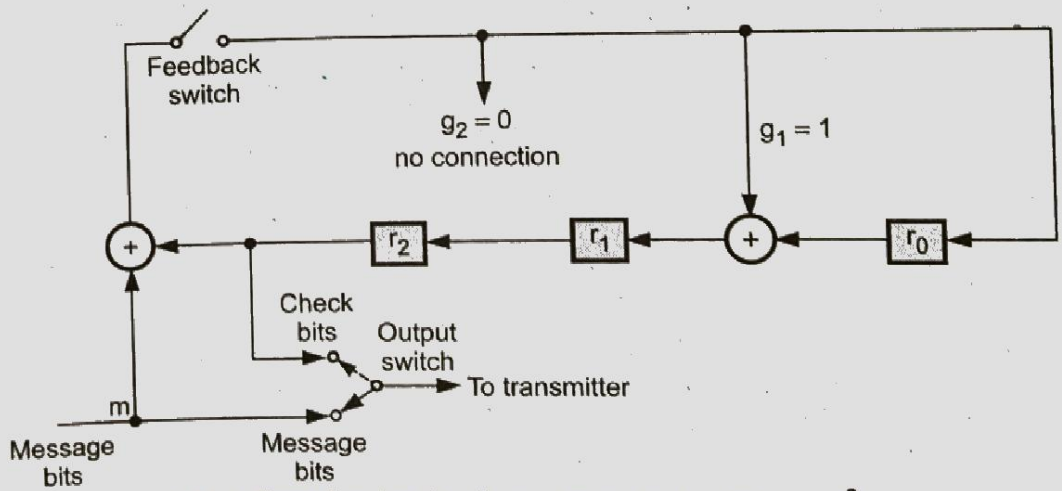


Fig. 4.13.3 Encoder for (7, 4) cyclic code for $G(p) = p^3 + p + 1$

Since $q=3$, there are '3' flip-flops in shift register to hold check bits c_1 c_2 and c_0. Since $g_2=0$, its link is not connected. $g_1=1$, hence its link is connected. Now let's verify the operation of this encoder for message vector $M=(m_3\ m_2\ m_1\ m_0)=(1100)$. Table 4.13.3 shows the contents of shift registers before and after shifts. (See Table 4.13.3 on next page.)

Input message bit m	Register bit inputs before shift				Register bit outputs after shift	
	$r_2=r_2'$	$r_1=r_1'$	$r_0=r_0'$	$r_2'=r_1$	$r_1'=r_0\oplus r_2\oplus m$	$r_0'=r_2\oplus m$
–	0	0	0	0	0	0
1	0	0	0	0	$0\oplus0\oplus1=1$	$0\oplus1=1$
1	0	1	1	1	$1\oplus0\oplus1=0$	$0\oplus1=1$
0	1	0	1	0	$1\oplus1\oplus0=0$	$1\oplus0=1$
0	0	0	1	0	$1\oplus0\oplus0=1$	$0\oplus0=0$

Table 4.13.3 Shift register bits positions for input message M = (1100)

Shift clock	Message bit m	Shift register outputs			Feedback switch on / off	Output switch position	Transmitted bits
		r_2'	r_1'	r_0'			
1	1	0	1	1	on	message	1
2	1	1	0	1	on	message	1
3	0	0	0	1	on	message	0
4	0	0	1	0	on	message	0
5	–	0	1	0	off	check bits	0 (r_2')
6	–	1	0	0	off	check bits	1 (r_2')
7	–	0	0	0	off	check bits	0 (r_2')

$r_2'=r_1'$ $r_1'=r_0'$ $r_2'=0$

Table 4.13.4 Operation of (7, 4) cyclic code encoder

The Table 4.13.3 shows that at the end of last message bit the register bit outputs are $r_2' = 0, r_1' = 1$ and $r_0' = 0$. The feedback switch is opened and output switch is closed to check its position. The check bits are then shifted to the transmitter. The check bits are shifted as $c_2 = r_2', c_1 = r_1'$ and $c_0 = r_0'$. The Table 4.13.4 illustrates the shift operation of message and check bits. We know that the code vector is,

$$X = (m_3\ m_2\ m_1\ m_0\ c_2\ c_1\ c_0) = (1\ 1\ 0\ 0\ 0\ 1\ 0)$$

The Table 4.13.4 illustrates how the bits are transmitted when input message is (1100).

4.13.5 Syndrome Decoding for Cyclic Codes

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,

$$X = Y \oplus E \quad \text{(from equation 4.12.20)} \qquad \ldots (4.13.13)$$

or we can write the above equation as,

$$Y = X \oplus E \qquad \ldots (4.13.14)$$

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

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

$$Y(p) = X(p) + E(p) \qquad \ldots (4.13.15)$$

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

$$Y(p) = M(p)\,G(p) + E(p) \qquad \ldots (4.13.16)$$

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

$$\frac{Y(p)}{G(p)} = \text{Quotient} + \frac{Remainder}{G(p)} \qquad \ldots (4.13.17)$$

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

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

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 (4.13.17) becomes,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{R(p)}{G(p)} \qquad \ldots (4.13.18)$$

4.13.4 Encoding using an $(n - k)$ Bit Shift Register

In this section we will discuss the encoders for systematic cyclic codes. Fig. 4.13.2 shows the block diagram of a generalized (n, k) cyclic code. The symbols used to draw encoders are shown in Fig. 4.13.1.

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

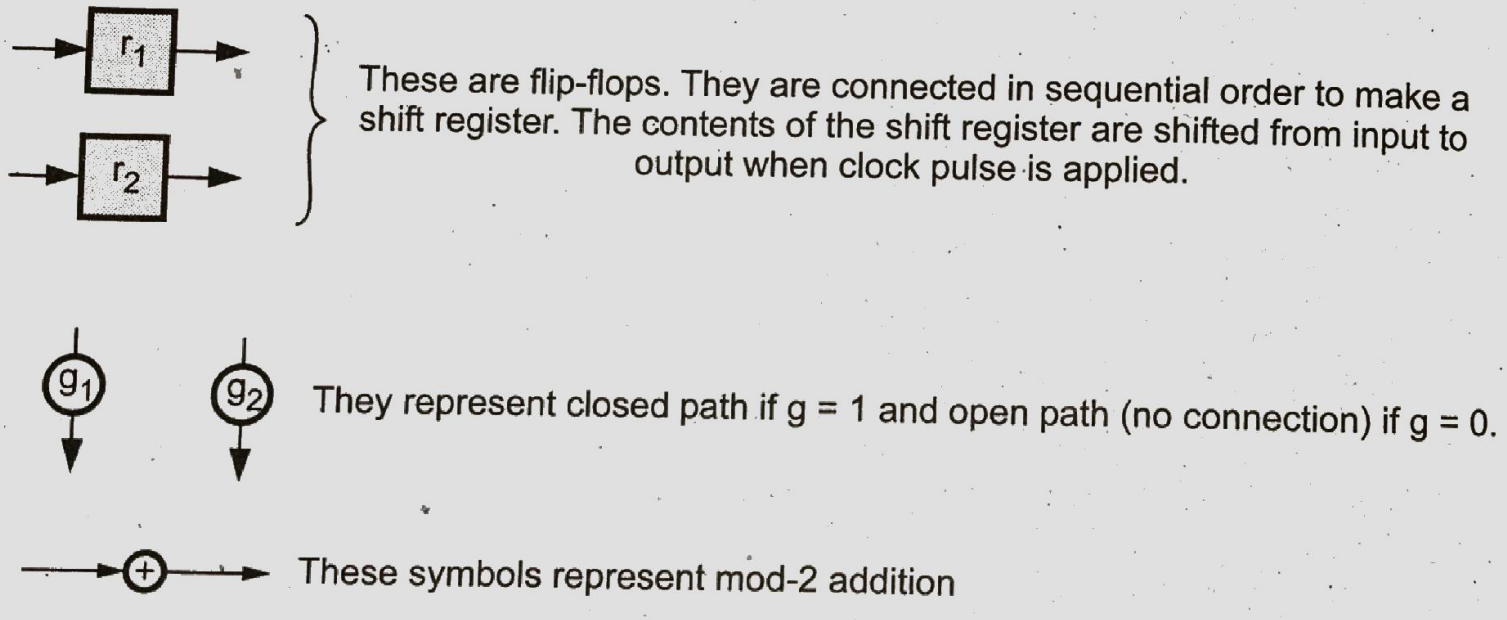


Fig. 4.13.1 Various symbols used in encoder

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.

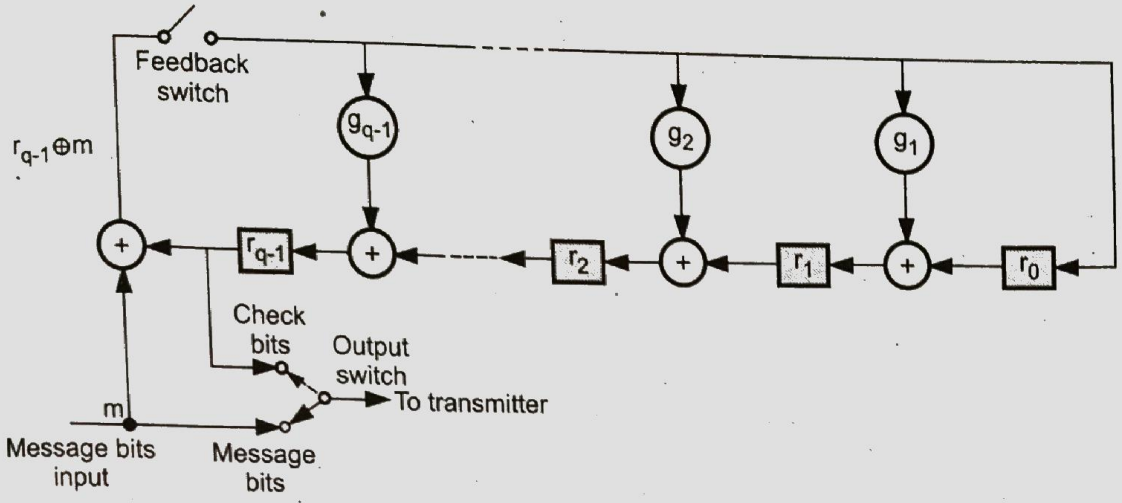


Fig. 4.13.2 Encoder for systematic (*n*, *k*) cyclic code

Ex. 4.13.3 *Design the encoder for the (7, 4) cyclic code generated by* $G(p) = p^3 + p + 1$ *and verify its operation for the received vector 1100.* **AU : Dec.-15, Marks 8**

Sol. : The generator polynomial is,

$$G(p) = p^3 + 0\,p^2 + p + 1$$

and $$G(p) = p^3 + g_2\,p^2 + g_1\,p + 1$$

On comparison of the two equation we obtain,

$$g_1 = 1 \quad \text{and} \quad g_2 = 0$$

and $$q = n - k = 7 - 4 = 3$$

With these values the block diagram of Fig. 4.13.2 will be as shown in Fig. 4.13.3 below.

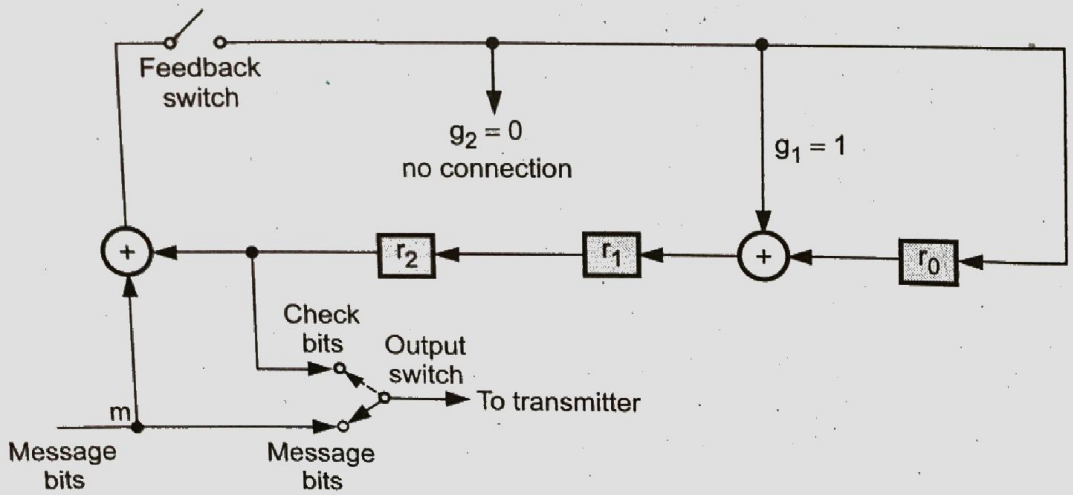


Fig. 4.13.3 Encoder for (7, 4) cyclic code for $G(p) = p^3 + p + 1$

Since $q=3$, there are '3' flip-flops in shift register to hold check bits c_1 c_2 and c_0. Since $g_2=0$, its link is not connected. $g_1=1$, hence its link is connected. Now let's verify the operation of this encoder for message vector $M=(m_3\ m_2\ m_1\ m_0)=(1100)$. Table 4.13.3 shows the contents of shift registers before and after shifts. (See Table 4.13.3 on next page.)

Input message bit m	Register bit inputs before shift				Register bit outputs after shift	
	$r_2=r_2'$	$r_1=r_1'$	$r_0=r_0'$	$r_2'=r_1$	$r_1'=r_0\oplus r_2\oplus m$	$r_0'=r_2\oplus m$
–	0	0	0	0	0	0
1	0	0	0	0	$0\oplus0\oplus1=1$	$0\oplus1=1$
1	0	1	1	1	$1\oplus0\oplus1=0$	$0\oplus1=1$
0	1	0	1	0	$1\oplus1\oplus0=0$	$1\oplus0=1$
0	0	0	1	0	$1\oplus0\oplus0=1$	$0\oplus0=0$

Table 4.13.3 Shift register bits positions for input message M = (1100)

Shift clock	Message bit m	Shift register outputs			Feedback switch on / off	Output switch position	Transmitted bits
		r_2'	r_1'	r_0'			
1	1	0	1	1	on	message	1
2	1	1	0	1	on	message	1
3	0	0	0	1	on	message	0
4	0	0	1	0	on	message	0
5	–	0	1	0	off	check bits	0 (r_2')
6	–	1	0	0	off	check bits	1 (r_2')
		$r_2'=r_1'$	$r_1'=r_0'$	$r_2'=0$			
7	–	0	0	0	off	check bits	0 (r_2')

Table 4.13.4 Operation of (7, 4) cyclic code encoder

The Table 4.13.3 shows that at the end of last message bit the register bit outputs are $r_2' = 0, r_1' = 1$ and $r_0' = 0$. The feedback switch is opened and output switch is closed to check bits position. The check bits are then shifted to the transmitter. The check bits are shifted as $c_2 = r_2', c_1 = r_1'$ and $c_0 = r_0'$. The Table 4.13.4 illustrates the shift operation of message and check bits. We know that the code vector is,

$$X = (m_3\ m_2\ m_1\ m_0\ c_2\ c_1\ c_0) = (1\ 1\ 0\ 0\ 0\ 1\ 0)$$

The Table 4.13.4 illustrates how the bits are transmitted when input message is (1100).

4.13.5 Syndrome Decoding for Cyclic Codes

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,

$$X = Y \oplus E \quad \text{(from equation 4.12.20)} \qquad \ldots (4.13.13)$$

or we can write the above equation as,

$$Y = X \oplus E \qquad \ldots. (4.13.14)$$

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

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

$$Y(p) = X(p) + E(p) \qquad \ldots (4.13.15)$$

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

$$Y(p) = M(p)\,G(p) + E(p) \qquad \ldots (4.13.16)$$

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

$$\frac{Y(p)}{G(p)} = \text{Quotient} + \frac{Remainder}{G(p)} \qquad \ldots (4.13.17)$$

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

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

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 (4.13.17) becomes,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{R(p)}{G(p)} \qquad \ldots (4.13.18)$$

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.

$$Y(p) = Q(p)G(p) + R(p) \quad \ldots (4.13.19)$$

On comparing equation 4.13.18 and above equation 4.13.19 we obtain,

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

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

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

$$\therefore \quad E(p) = [M(p) + Q(p)]\ G(p) + R(p) \quad \ldots (4.13.20)$$

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 equation (4.13.18) will be,

$$\frac{Y(p)}{G(p)} = Q(p) + \frac{S(p)}{G(p)} \quad \ldots (4.13.21)$$

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

$$\boxed{S(p) = rem\left[\frac{Y(p)}{G(p)}\right]} \quad \ldots (4.13.21(a))$$

4.13.5.1 Block Diagram of Syndrome Calculator

Fig. 4.13.4 shows the generalized block diagram of a syndrome calculator.

In Fig. 4.13.4 observe that there are 'q' stage shift register to generate 'q' bit syndrome vector.

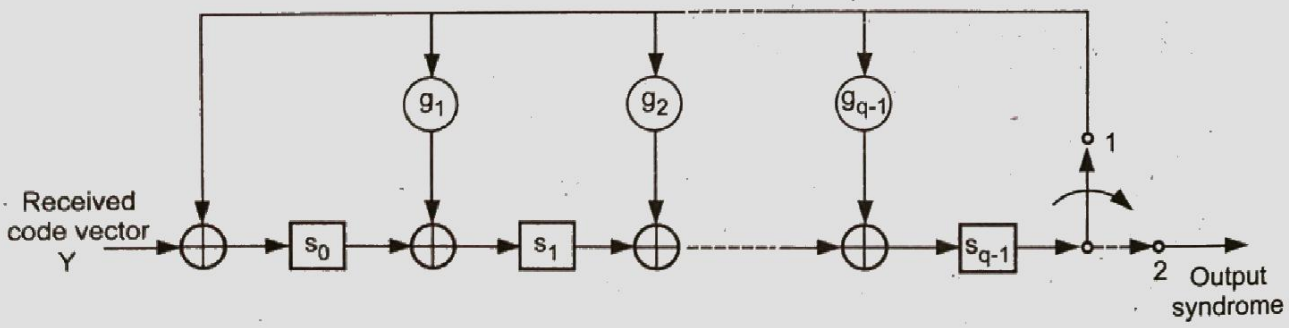


Fig. 4.13.4 Computation of syndrome for an (n, k) cyclic code

The operations as follows -

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 $S = (s_{q-1}, s_{q-2}, \ldots s_1\ s_0)$

Ex. 4.13.4 *Design a syndrome calculator for a (7, 4) cyclic Hamming code generated by the polynomial* $G(p) = p^3 + p + 1$. *Calculate the syndrome for* $Y = (1\ 0\ 0\ 1\ 1\ 0\ 1)$

AU : Dec.-15, Marks 8

Sol. : For the given code $n = 7, k = 4, q = n - k = 7 - 4 = 3$

The given generator polynomial is,

$$G(p) = p^3 + 0\,p^2 + p + 1$$

and $G(p) = p^3 + g_2\,p^2 + g_1\,p + 1$ generalized equation.

On comparison of the above two equations we obtain,

$$g_1 = 1 \text{ and } g_2 = 0$$

With these values the block diagram of a syndrome calculator for (7, 4) cyclic code will be as shown in Fig. 4.13.5.

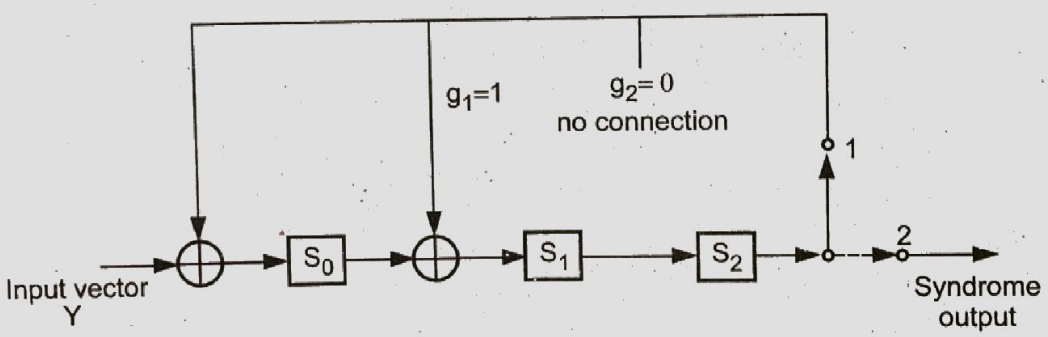


Fig. 4.13.5 Block diagram of a syndrome calculator for (7, 4) cyclic code with $G(p) = p^3 + p + 1$

Operation and explanation

The switch is kept in position 1 until all the '7' bits of received vector Y are shifted into the shift register. The flip-flops of the shift register contain syndrome vector when all bits of 'Y' are shifted. The switch is then closed to position 2 and clock pulses are applied to shift register. This gives syndrome vector at the output. The following table illustrates the operation of this syndrome calculator for received vector Y = (1 0 0 1 1 0 1). The table shows the contents of flip-flops with every shift.

Review Questions

1. *What are cyclic codes ? Why they are called sub class of block codes ?*
2. *Explain how cyclic codes are generated from the generating polynomials.*
3. *Explain how generator and parity check matrices are obtained for cyclic codes.*
4. *Explain the encoding and decoding methods for cyclic codes giving proper block diagrams.*
5. *Giving block diagram explain the operation of syndrome calculator for cyclic codes.*
6. *Explain what are BCH codes.*
7. *Explain the following*
   i) Golay codes ii) Shortened codes iii) Burst error correcting codes.

4.12 Convolutional Codes

AU : Dec.-10, 14, 16, May-15

4.12.1 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. 4.12.1 shows a convolutional encoder.

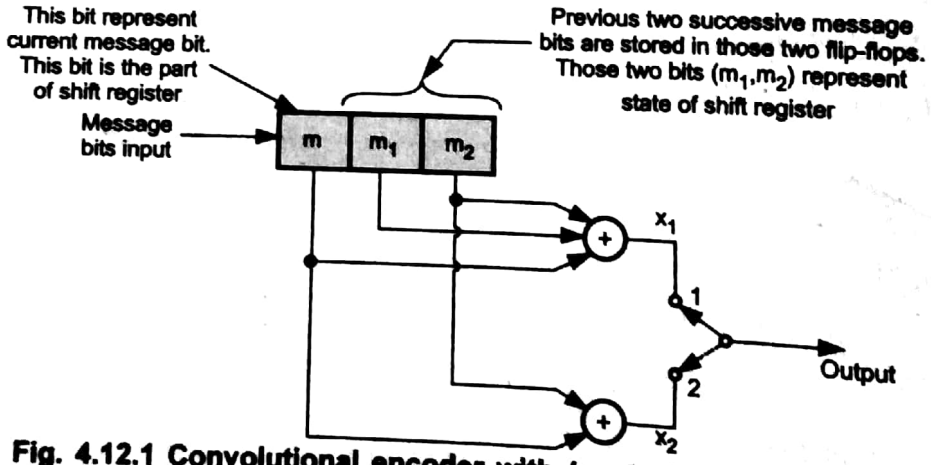


Fig. 4.12.1 Convolutional encoder with $k = 3$, $k = 1$ and $n = 2$

The above convolutional encoder operates as follows.

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,

$$x_1 = m \oplus m_1 \oplus m_2 \quad \ldots (4.12.1)$$

and

$$x_2 = m \oplus m_2 \quad \ldots (4.12.2)$$

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 4.12.1 and equation 4.12.2). The output switch then samples x_1 then x_2. Thus the output bit stream for successive input bits will be,

$$X = x_1x_2x_1x_2x_1x_2 \ldots \text{ and so on} \quad \ldots (4.12.3)$$

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, $n = 2$

4.12.1.1 Code Rate of Convolutional Encoder

The code rate of this encoder is,

$$r = \frac{k}{n} = \frac{1}{2} \quad \ldots (4.12.4)$$

In the encoder of Fig. 4.12.1, observe that whenever a particular message bit enters a shift register, it remains in the shift register for three shifts i.e.,

First shift →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, 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.

4.12.1.2 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. 4.12.1 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.

4.12.1.3 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. 4.12.1 has the dimension of (2, 1).

4.12.2 Time Domain Approach to Analysis of Convolutional Encoder

Let the sequence $\{g_0^{(1)}, g_1^{(1)}, g_2^{(1)} \ldots\ldots g_m^{(1)}\}$ denote the impulse response of the adder which generates x_1 in Fig. 4.12.1 Similarly, Let the sequence $\{g_0^{(2)}, g_1^{(2)}, g_2^{(2)} \ldots\ldots g_m^{(2)}\}$ denote the impulse response of the adder which generates x_2 in Fig. 4.12.1. These impulse responses are also called *generator sequences* of the code.

Let the incoming message sequence be $\{m_0, m_1, m_2 \ldots\ldots\}$. 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,

$$x_1 = x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l} \qquad i = 0, 1, 2 \ldots\ldots\ldots \quad (4.12.6)$$

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

$$x_2 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l} \qquad i = 0, 1, 2 \ldots\ldots \quad \ldots (4.12.7)$$

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

As shown in the Fig. 4.12.1, the two sequences x_1 and x_2 are multiplexed by the switch. Hence the output sequence is given as,

$$\{x_i\} = \left\{x_0^{(1)}\ x_0^{(2)}\ x_1^{(1)}\ x_1^{(2)}\ x_2^{(1)}\ x_2^{(2)}\ x_3^{(1)}\ x_3^{(2)} \ldots\ldots\right\} \quad \ldots (4.12.8)$$

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

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

Observe that bits from above two sequences are multiplexed in equation (4.12.8) The sequence $\{x_i\}$ is the output of the convolutional encoder.

Example for Understanding

Ex. 4.12.1 : *For the convolutional encoder of Fig. 4.12.2 determine the following :*
i) Dimension of the 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}

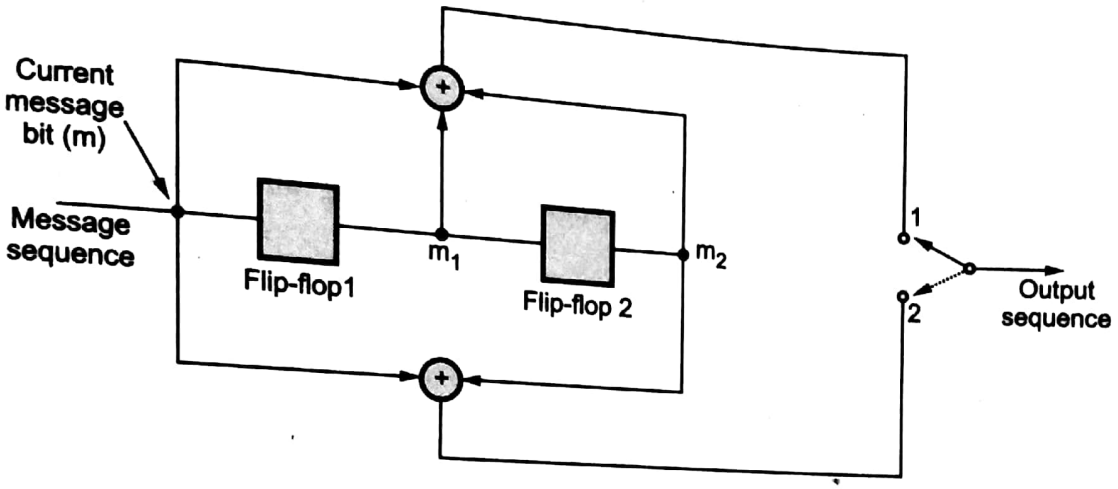


Fig. 4.12.2 Convolutional encoder of example 4.12.1

Sol. : In the Fig. 4.12.2 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. 4.12.3.

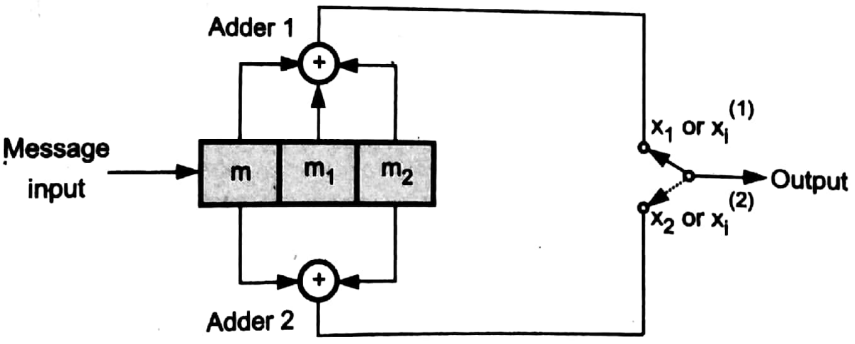


Fig. 4.12.3 Convolutional encoder of Fig. 4.12.1 redrawn alternately

Observe that the above encoder is exactly similar to that of Fig. 4.12.1.

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,

$$\text{Dimension} = (n, k) = (2, 1)$$

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 sequences :

In Fig. 4.12.3 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,

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

Here $g_0^{(1)}$ = 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 i.e. $x_i^{(2)}$ is generated by addition of first and last bits. Hence its generating sequence is given as,

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

Here $g_0^{(2)}$ = 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 (4.12.6) we can write,

$$x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l} \quad \ldots (3)$$

with $i=0$ above equation becomes,

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

$$\therefore \quad x_0^{(1)} = g_0^{(1)} m_0 = 1 \times 1 = 1, \quad \text{Here } g_0^{(1)} = 1 \text{ and } m_0 = 1$$

$i=1$ in equation (3), $x_1^{(1)} = g_0^{(1)} m_1 \oplus g_1^{(1)} m_0$

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

Here note that additions are mod-2 type.

$i=2$ in equation (3), $x_2^{(1)} = g_0^{(1)} m_2 + g_1^{(1)} m_1 + g_2^{(1)} m_0$

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

$i=3$ in equation (3), $x_3^{(1)} = g_0^{(1)} m_3 \oplus g_1^{(1)} m_2 \oplus g_2^{(1)} m_1$

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

$i=4$ in equation (3), $x_4^{(1)} = g_0^{(1)} m_4 \oplus g_1^{(1)} m_3 \oplus g_2^{(1)} m_2$

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

$i=5$ in equation (3), $x_5^{(1)} = g_0^{(1)} m_5 \oplus g_1^{(1)} m_4 \oplus g_2^{(1)} m_3$

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

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

$i=6$ in equation (3), $x_6^{(1)} = g_0^{(1)} m_6 \oplus g_1^{(1)} m_5 \oplus g_2^{(1)} m_4$

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

$$= 1 \times 1 = 1$$

Thus the output of adder 1 is,

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

To obtain output due to adder 2

Similarly from equation (4.12.7),

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

And $m_{i-l} = 0$ for all $l > i$.

with $i=0$ in above equation we get,

$$x_0^{(2)} = g_0^{(2)} m_0 = (1\times1) = 1 \quad \text{Here } g_0^{(2)} = 1 \text{ and } m_0 = 1$$

With $i=1$, $x_1^{(2)} = g_0^{(2)} m_1 \oplus g_1^{(1)} m_0 = (1\times0) \oplus (0\times1) = 0$

With $i=2$, $x_2^{(2)} = g_0^{(2)} m_2 \oplus g_1^{(2)} m_1 \oplus g_2^{(2)} m_0 = (1\times0) \oplus (0\times0) \oplus (1\times1) = 1$

With $i=3$, $x_3^{(2)} = g_0^{(2)} m_3 \oplus g_1^{(2)} m_2 \oplus g_2^{(2)} m_1 = (1\times1) \oplus (0\times0) \oplus (1\times0) = 1$

With $i=4$, $x_4^{(2)} = g_0^{(2)} m_4 \oplus g_1^{(2)} m_3 \oplus g_2^{(2)} m_2 = (1\times1) \oplus (0\times1) \oplus (1\times0) = 1$

With $i=5$, $x_5^{(2)} = g_1^{(2)} m_4 \oplus g_2^{(2)} m_3 = (0\times1) \oplus (1\times1) = 1$

With $i=6$, $x_6^{(2)} = g_2^{(2)} m_4 = 1\times1 = 1$

Thus the sequence x_2 is,

$$x_2 = x_i^{(2)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\} \quad \text{... (4)}$$

To obtain multiplexed sequence of x_1 and x_2 as per equation (4.12.8)

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

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

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

4.12.3 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.,

$$g^{(1)}(p) = g_0^{(1)} + g_1^{(1)} p + g_2^{(1)} p^2 + \ldots\ldots + g_M^{(1)} p^M \quad \text{... (4.12.9)}$$

Similarly,

$$g^{(2)}(p) = g_0^{(2)} + g_1^{(2)} p + g_2^{(2)} p^2 + \ldots\ldots + g_M^{(2)} p^M \quad \ldots (4.12.10)$$

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.,

$$m(p) = m_0 + m_1 p + m_2 p^2 + \ldots\ldots + m_{L-1} p^{L-1} \quad \ldots (4.12.11)$$

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

$$\boxed{\begin{aligned} x^{(1)}(p) &= g^{(1)}(p) \cdot m(p) \\ x^{(2)}(p) &= g^{(2)}(p) \cdot m(p) \end{aligned}} \quad \ldots (4.12.12)$$

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.

Ex. 4.12.2 : *Repeat part (V) of example 4.12.1 using transform domain calculations (polynomial multiplications).*

Sol. : a) To obtain generating polynomial for adder-1 :

The first generating sequence is given by equation (1) of example 4.12.1 i.e.,

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

Hence its polynomial can be obtained equation (4.12.9) as follows,

$$\begin{aligned} g^{(1)}(p) &= 1 + 1 \times p + 1 \times p^2 \\ &= 1 + p + p^2 \end{aligned} \quad \ldots (1)$$

b) To obtain generating polynomial for adder-2 :

The second generating sequence is given by equation (2) of example 4.12.1. i.e.,

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

Hence its polynomial can be obtained equation (4.12.10) as follows,

$$\begin{aligned} g^{(2)}(p) &= 1 + 0 \times p + 1 \times p^2 \\ &= 1 + p^2 \end{aligned} \quad \ldots (2)$$

c) To obtain message polynomial :

The message sequence is,

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

Hence its polynomial can be obtained equation (4.12.11) as,

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

$$= 1+p^3+p^4 \quad \ldots (3)$$

d) To determine the output due to adder-1 :

Now $x^{(1)}(p)$ can be obtained from equation (4.12.12) i.e.

$$x^{(1)}(p) = g^{(1)}(p)\cdot m(p) = (1+p+p^2)(1+p^3+p^4) = 1+p+p^2+p^3+p^6$$

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,

$$x^{(2)}(p) = g^{(2)}(p)\cdot m(p) = (1+p^2)(1+p^3+p^4) = 1+p^2+p^3+p^4+p^5+p^6$$

Thus the output sequence $x_i^{(2)}$ 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.

4.12.4 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. 4.12.1. It is reproduced below for convenience.

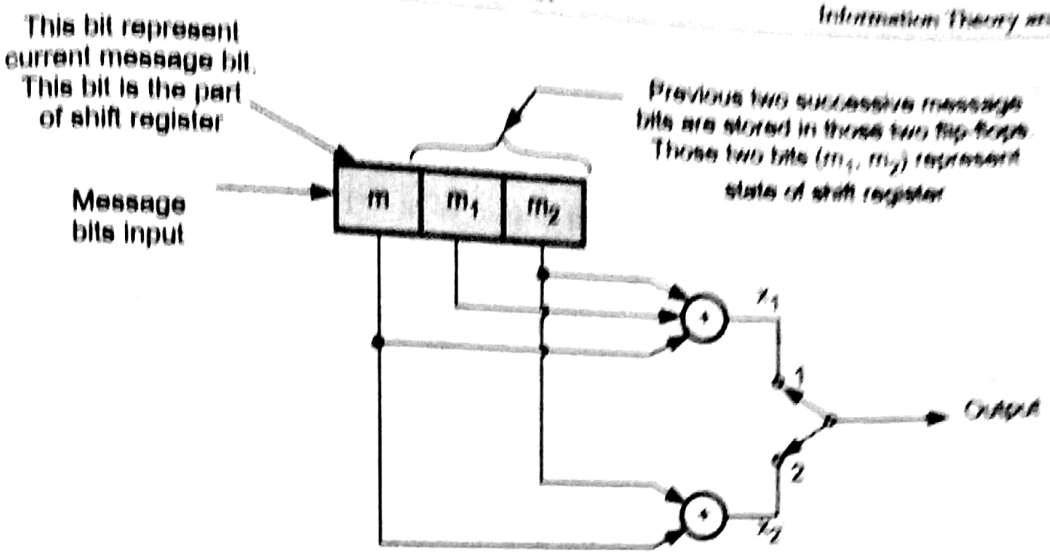


Fig. 4.12.4 Convolutional encoder with $k = 1$ and $n = 2$

4.12.4.1 States of the Encoder

In Fig. 4.12.4 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 4.12.1.

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

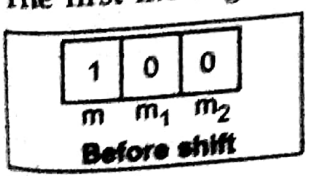
Table 4.12.1 States of the encoder of Fig. 4.12.4

4.12.4.2 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.

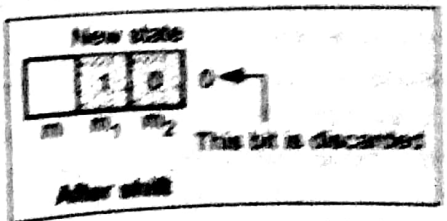
1) When $m=1$ i.e. first bit

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



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

$x_2 = 1 \oplus 0 = 1$



The values of x_1x_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_1x_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_1x_2 =11. The first row of Table 4.12.2 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. 4.12.5.

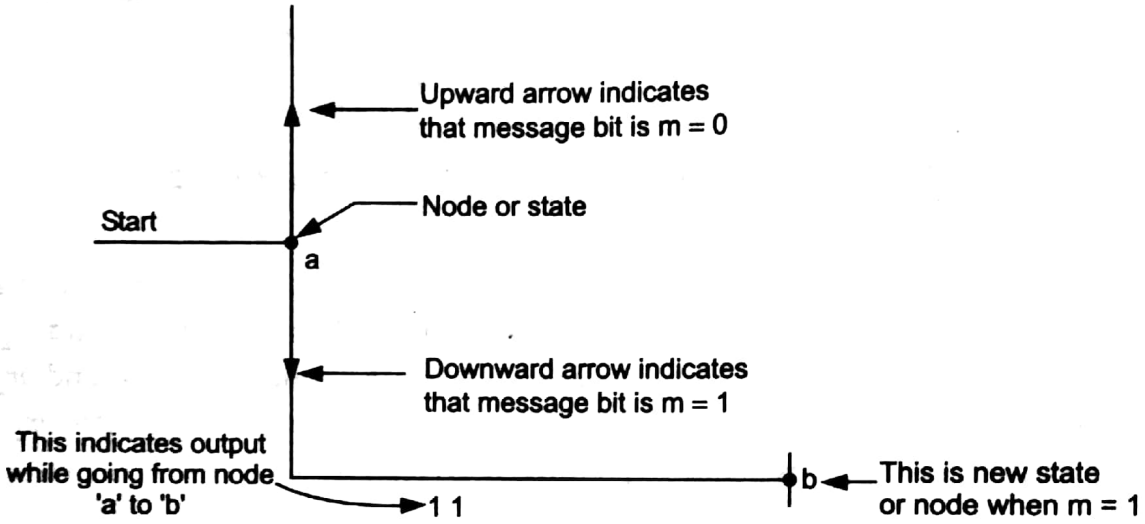
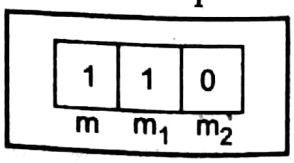


Fig. 4.12.5 Code tree from node 'a' to 'b'

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

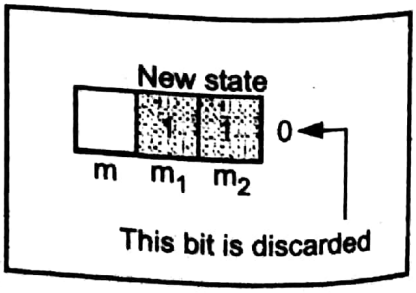
Now let the second message bit be 1. The contents of shift register with this input will be as shown below.



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

$x_2 = 1 \oplus 0 = 1$

These values of x_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 4.12.2 in second row. The last column of the table shows the code tree for those first and second input bits.

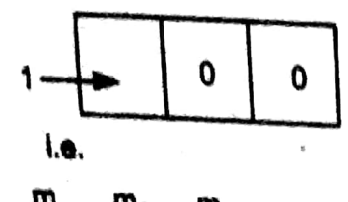
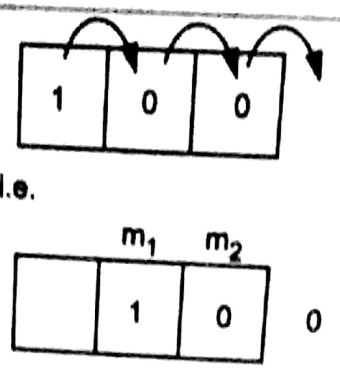
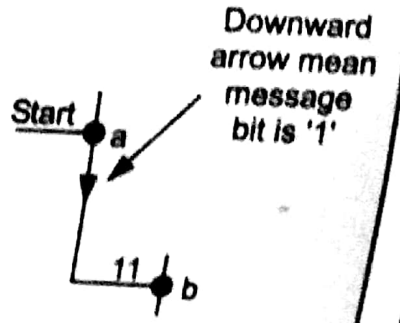
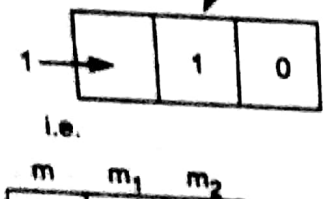
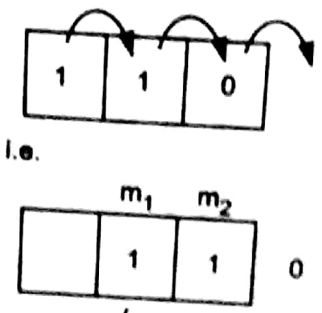
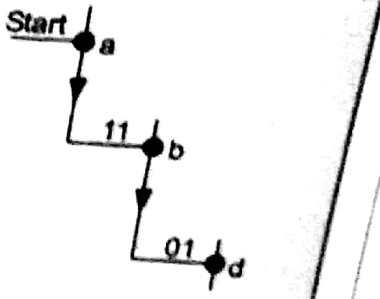
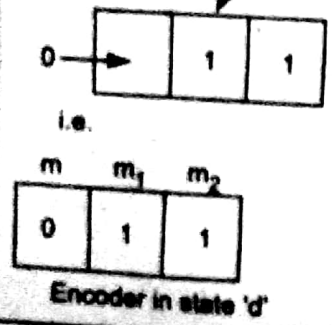
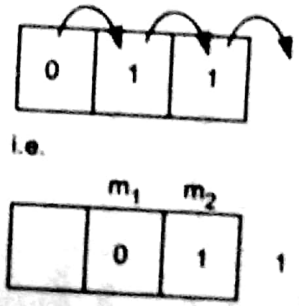
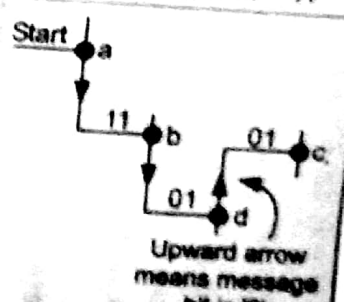
Bit No.	Input message bit m	Status of shift register after entry of m	Calculation of outputs x_1 and x_2	Status of shift register after transmission of o/p and shift right by one bit	New state of encoder $m_2\ m_1$	Transmitted outputs $x_1\ x_2$	Code tree diagram downward arrow indicated input is 1
1.	1		$x_1 = 1 \oplus 0 \oplus 0 = 1$ $x_2 = 1 \oplus 0 = 1$		01, i.e. b	11	 Code tree for m = 1
2.	1		$x_1 = 1 \oplus 1 \oplus 0 = 0$ $x_2 = 1 \oplus 0 = 1$		11, i.e. d	01	 Code tree for m = 11
3.	0		$x_1 = 0 \oplus 1 \oplus 1 = 0$ $x_2 = 0 \oplus 1 = 1$		10, i.e. c	01	 Code tree for m = 110

Table 4.12.2 Analysis of convolutional encoder of Fig. 4.12.4

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

Similarly 3rd row of the Table 4.12.2 (Refer table on previous page) illustrates 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 $x_1x_2 = 01$.

Complete code tree for convolutional encoder

Fig. 4.12.6 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.

If you carefully observe the code tree of Fig. 4.12.6, you will find that the branch pattern begins to repeat after third bit. 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.

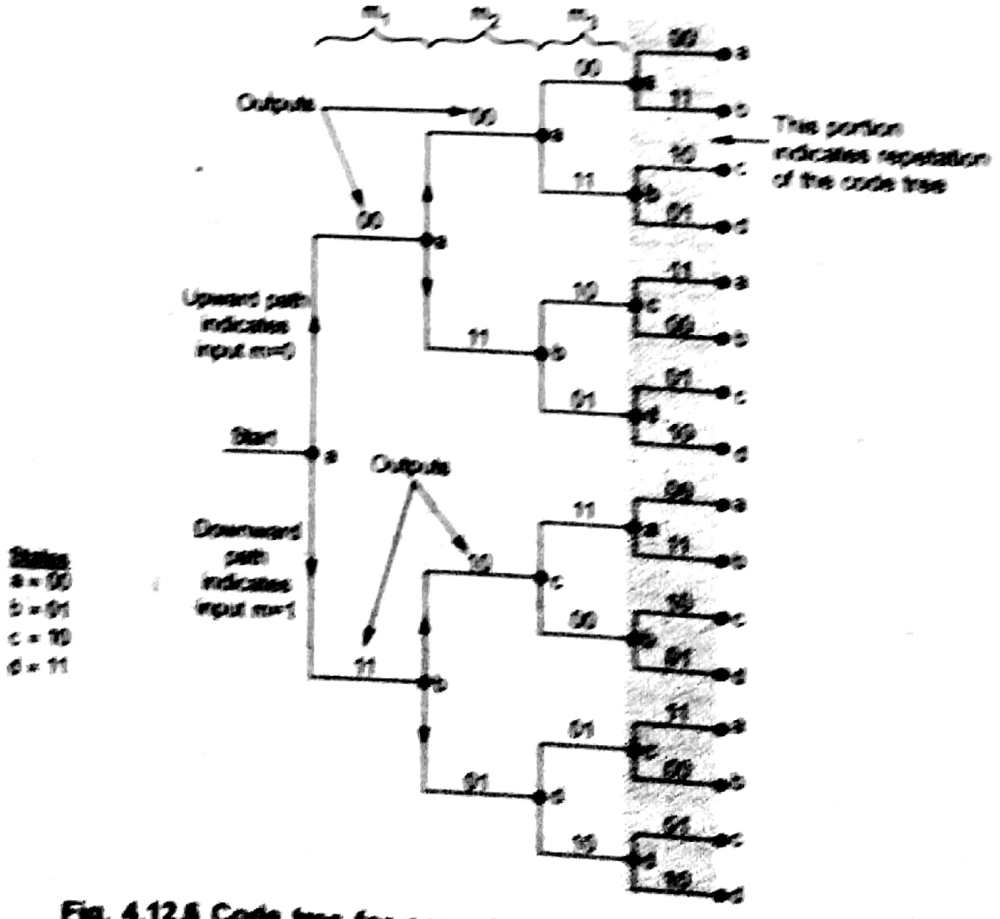


Fig. 4.12.6 Code tree for convolutional encoder of Fig. 4.12.5

4.12.4.3 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, an unique diagram for such transitions. Fig. 4.12.7 shows code trellis diagram.

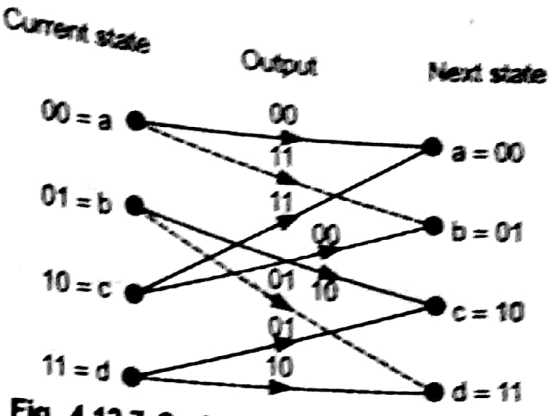


Fig. 4.12.7 Code trellis of convolutional encoder of Fig. 4.12.1

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 line 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.

4.12.4.4 State Diagram

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

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 $x_1x_2 = 11$

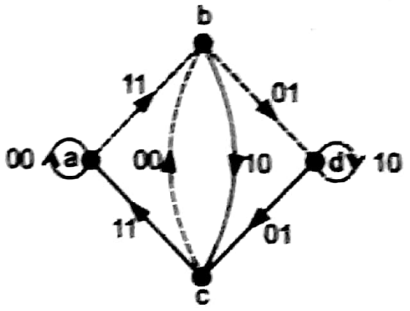


Fig. 4.12.8 State diagram for convolutional encoder of Fig. 4.12.1

Comparison between code tree and trellis diagram :

Table 4.12.3 shows the comparison between code tree and trellis diagram as a graphic structures 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 between connect to next states.
2.	Code tree is lengthy way of representing coding process.	Code trellis diagram is shorter or compact way of representing
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 4.12.3 Comparison between code tree and trellis diagram

4.12.5 Decoding Methods of Convolutional Codes

These methods are used for decoding of convolutional codes. They are viterbi algorithm, sequential decoding and feedback decoding. Let's consider them in details in subsequent sections.

4.12.5.1 Viterbi 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 as such. 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. 4.12.1. For this encoder we have obtained code trellis in Fig. 4.12.7. 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_1x_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

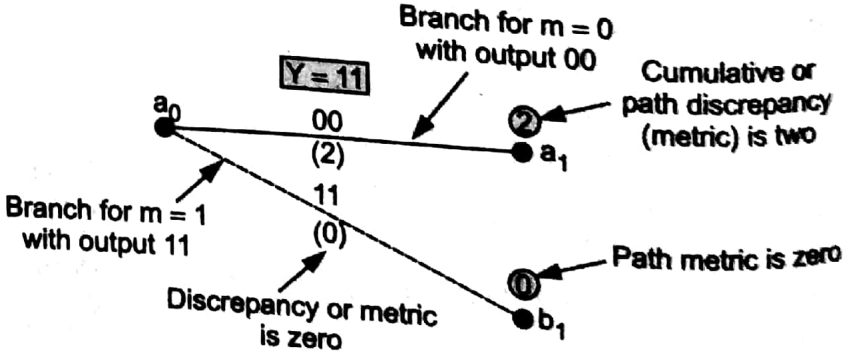


Fig. 4.12.9 Viterbi decoder results for first message bit

at state a_0. Now look at the code trellis diagram of Fig. 4.12.7 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. 4.12.9. 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.

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. 4.12.10 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.

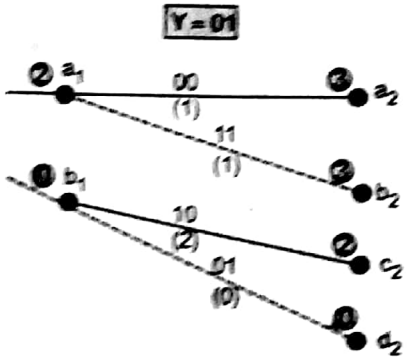


Fig. 4.12.10 Viterbi decoder results for second message bit

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

Fig. 4.12.11 shows the trellis diagram for all the six bits of Y.

Fig. 4.12.11 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 $a_0 a_1 a_2 a_3$ with metric 5. The other path is $a_0 b_1 c_2 a_3$ 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. As shown in Fig. 4.12.11, the paths marked with $\times$ (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.

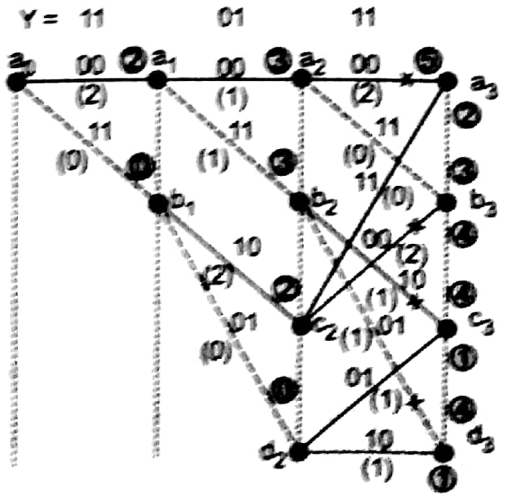


Fig. 4.12.11 Paths and their metrics for viterbi decoding

d) Further explanation of viterbi decoding for 12 message bits

Fig. 4.12.12 shows the continuation of Fig. 4.12.11 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.

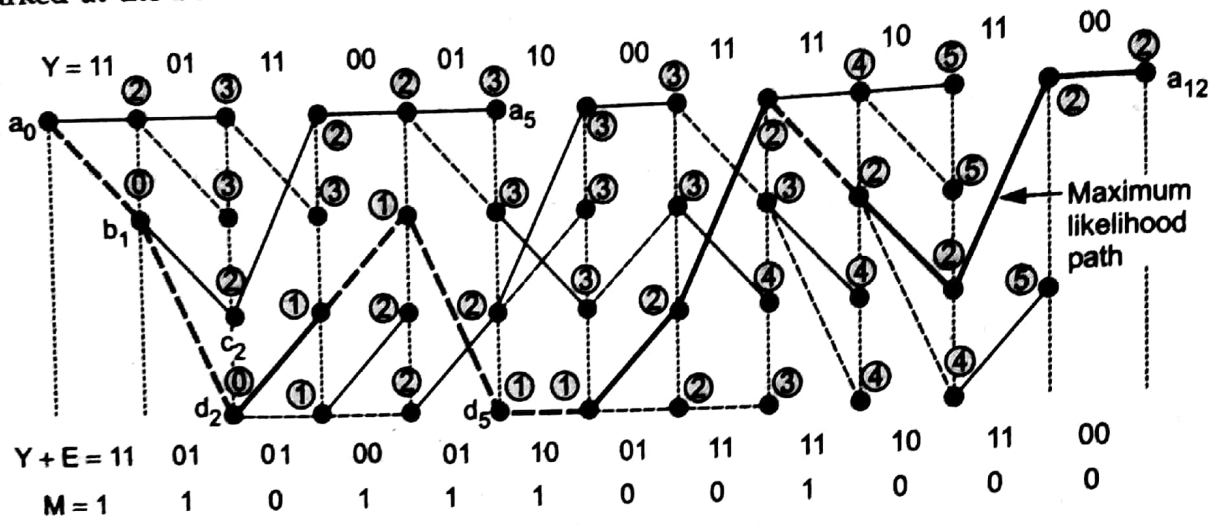


Fig. 4.12.12 Viterbi decoding

Only one path of particular node is kept which is having lower metric. In case if there are two paths having same metric, then any one 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.

This is the complete explanation of viterbi decoding. The method of decoding used in viterbi decoding is called *maximum likelihood decoding.*

e) Surviving paths

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

$$\text{Surviving paths} = 2^{(K-1)k} \quad \ldots (4.12.13)$$

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

For the encoder for Fig. 4.12.1 $K=3$ and $k=1$

$$\therefore \quad \text{Surviving paths} = 2^{(3-1)\times 1} = 4$$

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 (in present example four) 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 in Fig. 4.12.10. 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.

4.12.5.2 Sequential Decoding for Convolutional Codes

Sequential decoding uses metric divergence effect. Fig. 4.12.13 (a) shows the code trellis for the convolutional encoder of Fig. 4.12.1. The same code trellis we have seen in the last subsection. Following are the important points about sequential decoding.

1) The decoding starts at a_0. It follows the single path by taking the branch with smallest metric. For example as shown in Fig. 4.12.13 (a), the path for first three nodes is $a_0 b_1 d_2$ since its metric is the lowest.

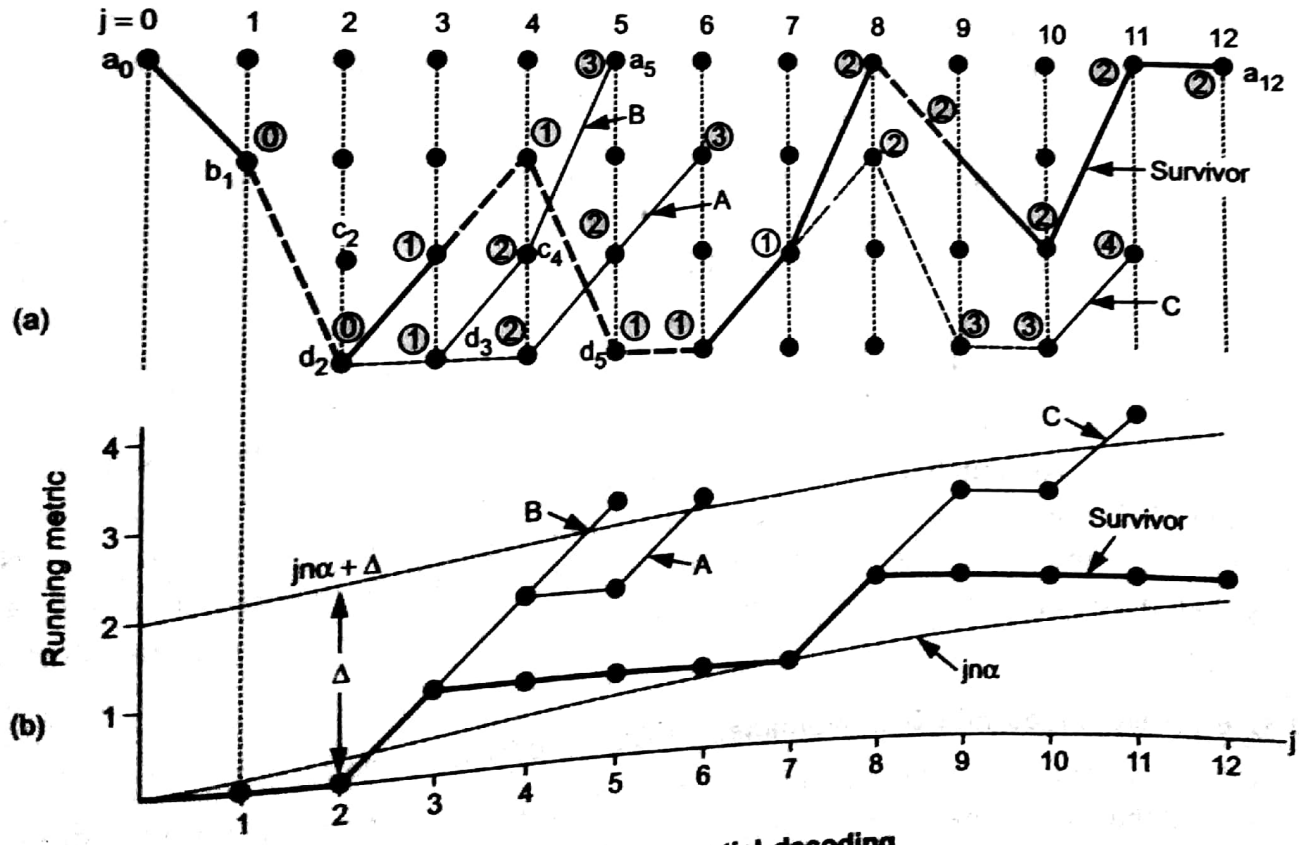


Fig. 4.12.13 Sequential decoding

2) If there are two or more branches from the same node with same metric, then decoder selects any one branch and continues decoding.

3) From (2) above we know that if there are two branches from one nodes with equal metrics, then any one is selected at random. If the selected path is found to be unlikely with rapidly increasing merit, then decoder cancels that path and goes back to that node. It then selects other path emerging from that node. For example observe in the Fig. 4.12.13 (a) that two branches with same metric emerge from node d_2. One path is $d_2 d_3 c_4 a_5$ (or path marked 'B') with metric '3' at a_5. Therefore decoder drops this path and follows other path.

4) The decision about dropping a path is based on the expected value of running metric at a given node. Running metric at a particular j^{th} node is given as,

$$\text{Running metric} = j n \alpha \quad \ldots (4.12.14)$$

Where j is the node at which metric is to be calculated.

n is the number of encoded output bits for one message bit.

and α is the transmission error probability per bit.

The sequential decoder abandons a path whenever its running metric exceeds $(j n \alpha + \Delta)$. Here Δ is the should be above $j n \alpha$ at j^{th} node. Fig. 4.12.13 (b) shows the running metric at a particular node with respect to number of that node. The two dotted lines shows the range of threshold 'Δ' above $jn\alpha$ at a particular node. Observe that since metric of path 'B' exceeds the threshold at 5^{th} node, it is abandoned and decoder starts from node 'Z' again. Similarly path 'A' is also abandoned.

5) If the running metric of every path goes out of threshold limits then the value of threshold 'Δ' is increased and decoder tries back again. In Fig. 4.12.13 (b) the value of $\alpha = 1/16$, for encoder of Fig. 4.12.1 we know that $n = 2$. Let's calculate $jn\alpha$ at 8^{th} node.

At 8^{th} node $jn\alpha = 8 \times 2 \times 1/16 = 1$ The value of $\Delta = 2$. Therefore threshold will be,

Threshold = $jn\alpha + \Delta = 1 + 2 = 3$ at 8^{th} node. Similarly the threshold at other nodes can be calculated. The computations involved in sequential decoding are less than viterbi decoding. But the back tracking in sequential decoding is complex. The output error probability is more in case of sequential decoding. Both the viterbi decoding and sequential decoding methods can be implemented with the help of computer software efficiently.

4.12.6 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 some 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.

Ex. 4.12.3 : *A rate 1/3 convolution encoder has generating vectors as* $g_1 = (1\,0\,0)$, $g_2 = (1\,1\,1)$ *and* $g_3 = (1\,0\,1)$.

i) Sketch the encoder configuration.

ii) Draw the code tree, state diagram and trellis diagram.

iii) If input message sequence is 10110, determine the output sequence of the encoder.

Sol. : To determine dimension of the code :

This is rate 1/3 code. We know that

rate $= \frac{k}{n} = \frac{1}{3}$, therefore $k = 1$ and $n = 3$.

i) To sketch encoder configuration :

Here $k = 1$ and $n = 3$. This means each message bit generates three output bits. There will be three stage shift register. It will contain m, m_1 and m_2.

First output x_1 will be generated due to $g_1 = (100)$

Since $g_1 = (100)$, $x_1 = m$

Second output x_2 will be generated due to $g_2 = (111)$

Since $g_2 = (111)$, $x_2 = m \oplus m_1 \oplus m_2$

Third output x_3 will be generated due to $g_3 = (101)$

Since $g_3 = (101)$, $x_3 = m \oplus m_2$.

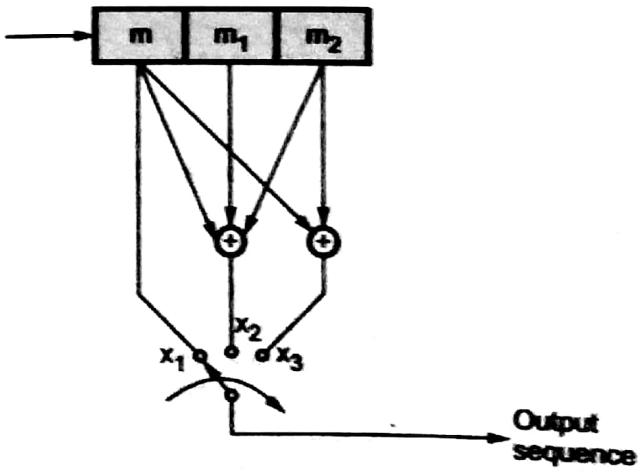


Fig. 4.12.14 Encoder

Fig. 4.12.14 shows the diagram of encoder based on above discussion.

ii) To draw code tree, state diagram and trellis diagram :

a) To obtain trellis diagram

The two bits $m_2\ m_1$ in the shift register will indicate the state of the encoder. Let these states be defined as follows :

$m_2\ m_1$ = 0 0 state '*a*'

$m_2\ m_1$ = 0 1 state '*b*'

$m_2\ m_1$ = 1 0 state '*c*'

$m_2\ m_1$ = 1 1 state '*d*'

Table 4.12.4 lists the state transition calculations.

Sr. No.	Current state $m_2 m_1$	Input m	Outputs $x_1 = m$ $x_2 = m \oplus m_1 \oplus m_2$ $x_3 = m \oplus m_2$			Next state $m_1 m$
1.	a = 0 0	0	0	0	0	0 0, i.e. a
		1	1	1	1	0 1, i.e. b
2.	b = 0 1	0	0	1	0	1 0, i.e. c
		1	1	0	1	1 1, i.e. d
3.	c = 1 0	0	0	1	1	0 0, i.e. a
		1	1	0	0	0 1, i.e. b
4.	d = 1 1	0	0	0	1	1 0, i.e. c
		1	1	1	0	1 1, i.e. d

Table 4.12.4 State transition calculations

How next stage is written ?

Consider the following diagram.

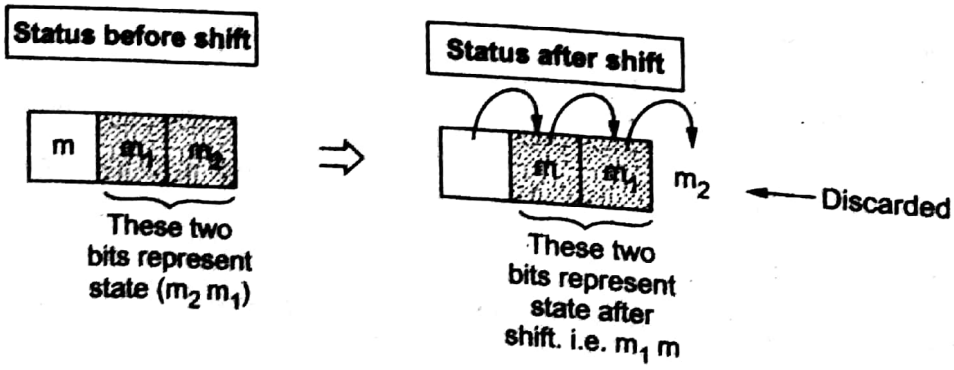
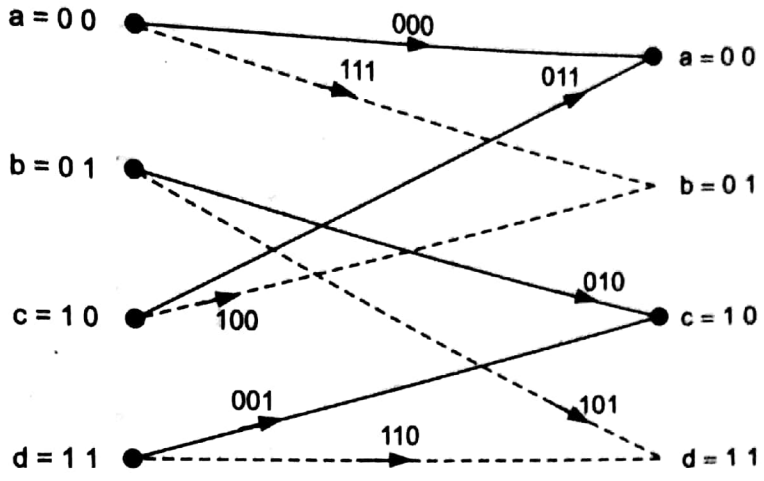


Fig. 4.12.15 Present and next states

As shown in Fig. 4.12.15, current state is $m_2\ m_1$. When the bits are shifted, then next state becomes $m_1\ m$. Table 4.12.4 shows current and next states according to this concept.

A trellis diagram is shown in Fig. 4.12.16 based on Table 4.12.4.



Dotted line indicates input m = 1

Solid line indicates input m = 0

Fig. 4.12.16 Trellis diagram for encoder of Fig. 4.12.14

b) To obtain state diagram

If we combine the modes in trellis diagram, then we will get state diagram. It is shown below.

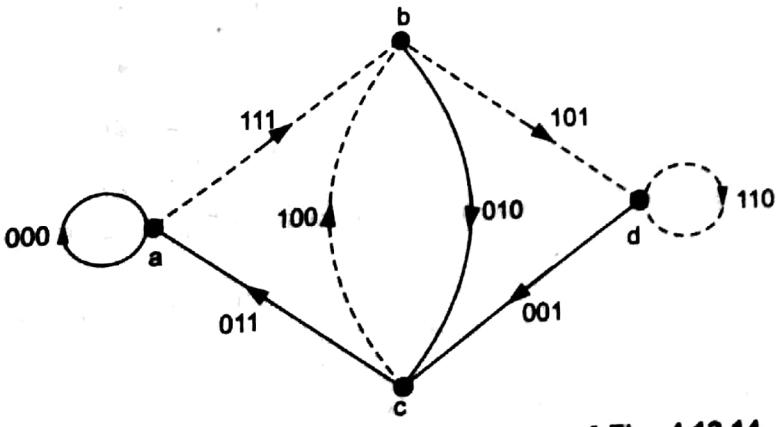


Fig. 4.12.17 State diagram of encoder of Fig. 4.12.14

c) To obtain code tree

Code tree can be developed with the help of state diagram. Following procedure should be performed :

1. Begin with any node (normally *a*).
2. Draw its next states for $m = 0$ and 1.
3. For every state determine next states for $m = 0$ and 1.
4. Repeat step 3 till code tree starts repeating.

Assumption : Upward movement in code tree indicates $m=0$.

Downward movement indicates $m=1$.

Based on above procedure, the code tree is developed as shown in Fig. 4.12.18.

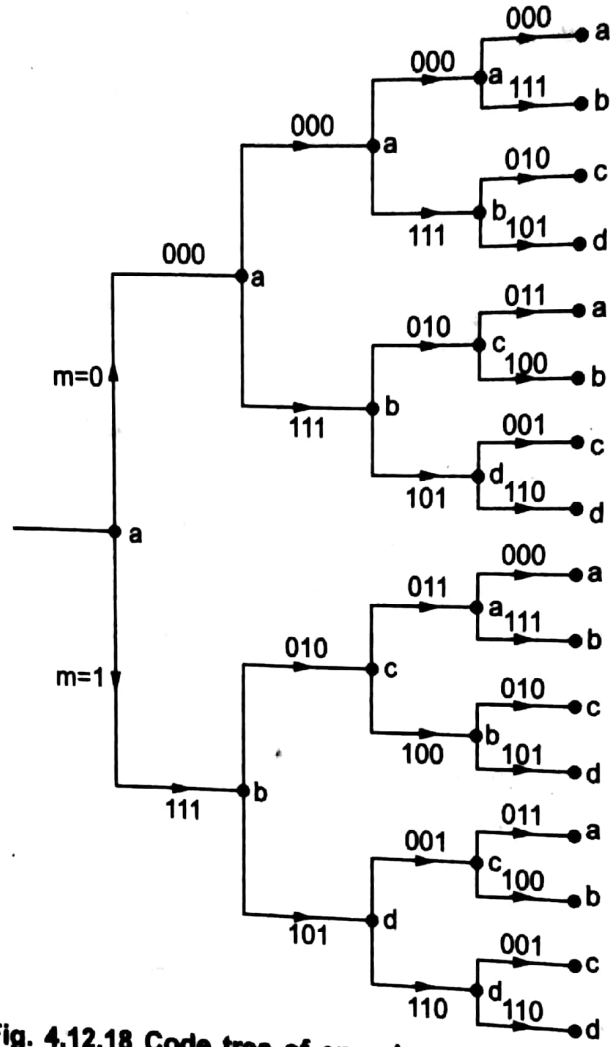


Fig. 4.12.18 Code tree of encoder of Fig. 4.12.14

In the above figure, observe that code tree repeats after third stage. This is because each input bit affects upto three output bits of every mod-2 adder.

iii) To obtain output sequence for m = 10110 :

a) To determine generator polynomials :

The generator polynomial can be written for $x_i^{(1)}$ as,

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

$$\therefore \quad g^{(1)}(p) = 1+0\,p+0\,p^2$$

$$\Rightarrow \quad \boxed{g^{(1)}p=1}$$

Similarly generating polynomial for $x_i^{(2)}$ can be written as,

$$g_i^{(2)} = \{111\}$$

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

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

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

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

$$\therefore \quad g^{(3)}(p) = 1 + 0\,p + p^2$$

$$\Rightarrow \boxed{g^{(3)}(p) = 1 + p^2}$$

b) To determine the message polynomial

The message sequence is given as,

$$m = 1\ 0\ 1\ 1\ 0$$

$$\therefore \quad m(p) = 1 + 0\,p + p^2 + p^3 + 0\,p^4$$

$$\Rightarrow \boxed{m(p) = 1 + p^2 + p^3}$$

c) To obtain output sequence for $g_i^{(1)}$

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

$$x_i^{(1)} = g_i^{(1)}(p)\ m(p) = 1\left(1 + p^2 + p^3\right) = 1 + p^2 + p^3$$

Hence the corresponding sequence is,

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

d) To obtain output sequence for $g_i^{(2)}$

The sequence $x_i^{(2)}$ can be obtained as,

$$x_i^{(2)} = g_i^{(2)}(p)\ m(p)$$

$$= \left(1 + p + p^2\right)\left(1 + p^2 + p^3\right) = 1 + p^2 + p^3 + p + p^3 + p^4 + p^2 + p^4 + p^5$$

$$= 1 + p + 0\,p^2 + 0\,p^3 + 0\,p^4 + p^5$$

Hence the corresponding sequence is,

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

e) To obtain output sequence for $g_i^{(3)}$

The sequence $x_i^{(3)}$ can be obtained as,

$$x_i^{(3)} = g_i^{(3)}(p)\, m(p) = \left(1+p^2\right)\left(1+p^2+p^3\right)$$

$$= 1+p^2+p^3+p^2+p^4+p^5 = 1+0\,p^2+p^3+p^4+p^5$$

Hence the corresponding output sequence is,

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

f) To multiplex three output sequences

The three sequences $x_i^{(1)}$, $x_i^{(2)}$ and $x_i^{(3)}$ are made in equal length, i.e. 6 bits. Hence zeros are appended to $x_i^{(1)}$. These sequences are as follows :

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

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

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

Bits from the above three sequences are multiplexed to give the output sequence. i.e.,

$$x_i = \{111\ \ 010\ \ 100\ \ 101\ \ 001\ \ 011\}$$

This is an output sequence of the encoder.

Example with Solution

Ex. 4.12.4 : *For the convolutional encoder with constraint length of 3 and rate 1/2 as shown in Fig. 4.12.19, draw the state diagram and trellis diagram. Is the generated code systematic ? By using viterbi algorithm, decode sequence 0100010000 ....*

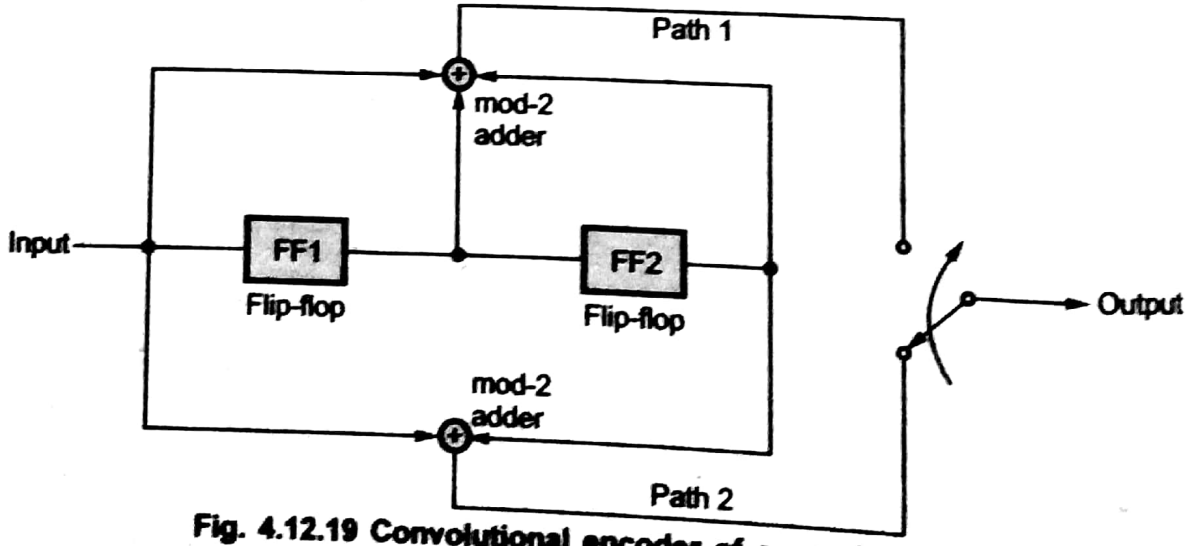


Fig. 4.12.19 Convolutional encoder of example 4.12.4

Sol. : i) To determine dimensions of the code :

Rate $= \frac{k}{n} = \frac{1}{2}$. Hence $k=1$ and $n=2$. For every message bit, there are two bits encoded at the output.

Constraint length $K=3$. Hence output is influenced by three shifts in the encoder.

ii) To obtain state diagram and trellis diagram :

Let us redraw the diagram of encoder as shown. The state of the encoder is represented by $m_2\, m_1$. Input is 'm'. Carefully observe that, Fig. 4.12.19 is similar to convolutional encoder of Fig. 4.12.4. In Fig. 4.12.19 two flip flops hold previous two inputs (i.e. $m_1\, m_2$). Third flip-flop is not shown, but input 'm' is used directly. In Fig. 4.12.4, there are three stages in shift register which contain m, m_1 and m_2. Functionally both the encoders are same.

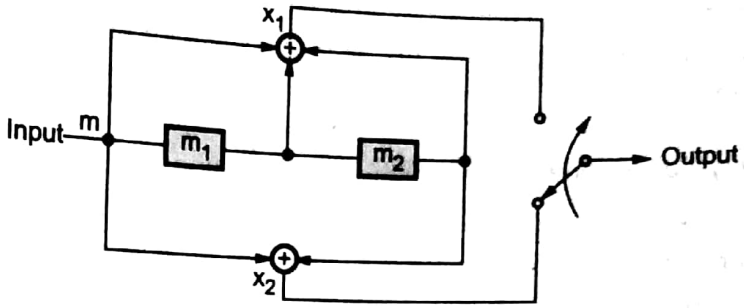
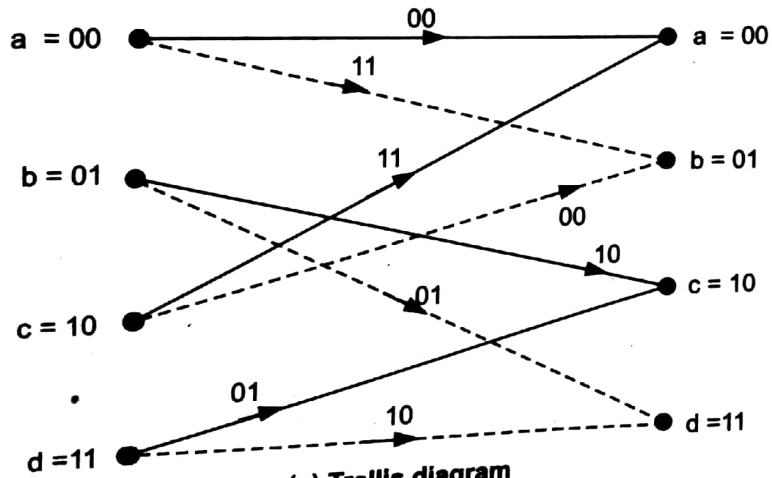
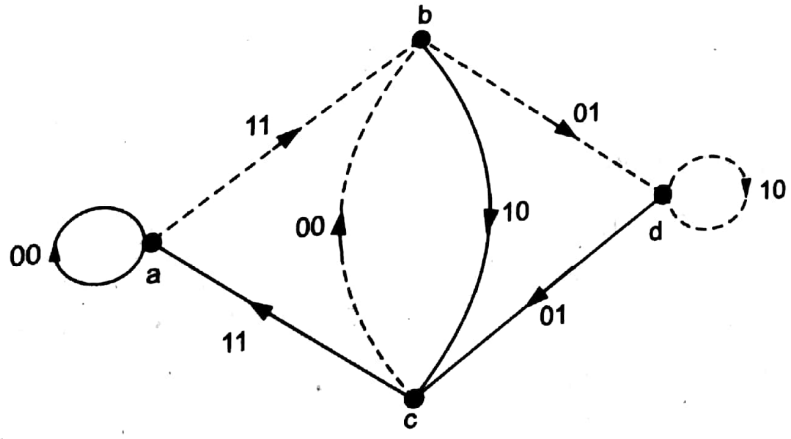


Fig. 4.12.20 Convolutional encoder of Fig. 4.12.19

Hence code trellis and state diagram of this encoder will be similar to those given in Fig. 4.12.7 and Fig. 4.12.8. They are given below :



(a) Trellis diagram



(b) State diagram

Fig. 4.12.21

iii) Whether the generated code is systematic ?

For the output code to be systematic, the message bit and check bits must be identified. But this is not possible in the output sequence of given encoder. Hence generated code is not systematic.

iv) To decode 0100010000 .....

A trellis diagram is shown in Fig. 4.12.22. In this figure observe that only survivor paths are shown dark. Running metrics are marked near every node. The path giving lower metric is retained at particular node. Thus at the nodes a_i, b_i, c_i, d_i only four paths are retained. These paths are evaluated at every stage of decoding. At the end, four survivor paths are written along with their metrics.

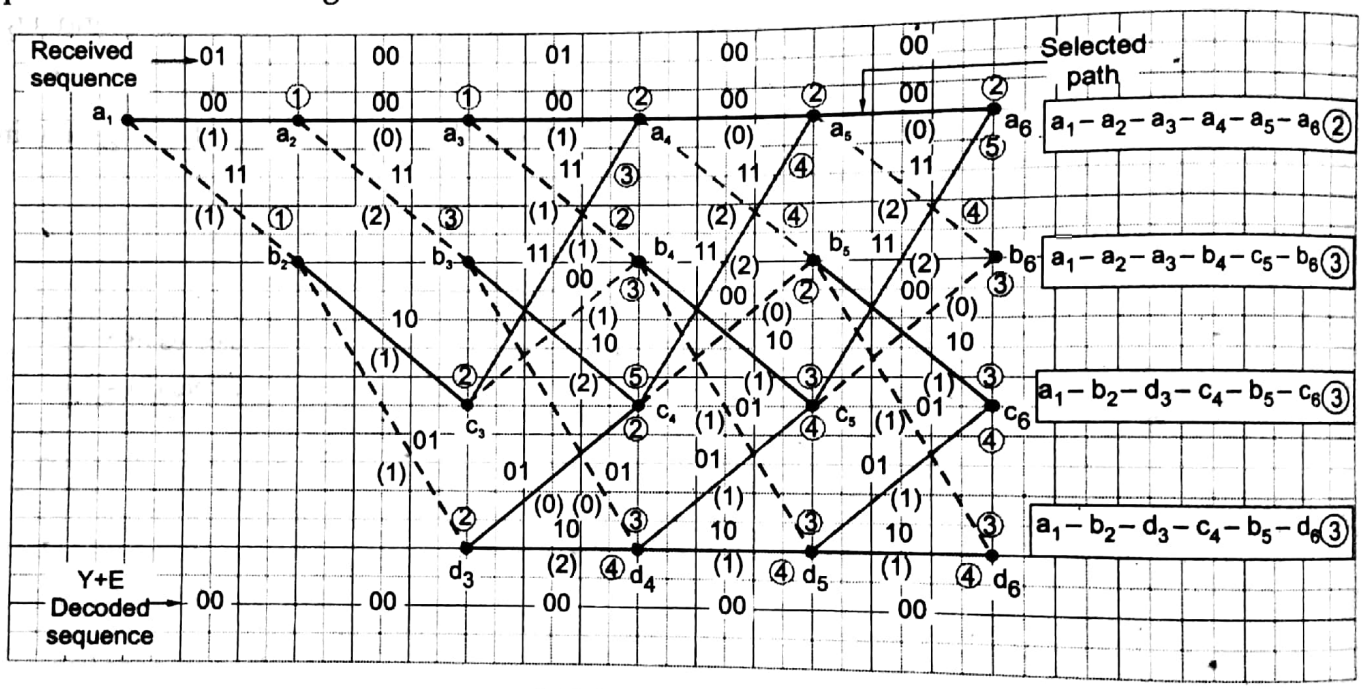


Fig. 4.12.22 Viterbi decoding

They are :

Path 1 : $a_1 - a_2 - a_3 - a_4 - a_5 - a_6$ with metric 2

Path 2 : $a_1 - a_2 - a_3 - b_4 - c_5 - b_6$ with metric 3

Path 3 : $a_1 - b_2 - d_3 - c_4 - b_5 - c_6$ with metric 3

Path 4 : $a_1 - b_2 - d_3 - c_4 - b_5 - d_6$ with metric 3

Out of these four paths, first path has lowest metric. Hence it must be considered for decoding the output sequence.

Path 1 : $a_1 - a_2 - a_3 - a_4 - a_5 - a_6$ is shown thick in Fig. 4.12.22.

For path '1' the output is,

Output : 00 00 00 00 00 00

4.12.7 Comparison between Linear Block Codes and Convolutional Codes

Till now we studied convolutional codes and block codes. They can be compared on the basis of their encoding methods, decoding methods, error correcting capabilities, complexity etc points. Table 4.12.5 lists the comparison.

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. $x_i = \sum_{l=0}^{M} g_l m_{i-l}, i = 0, 1, 2, ...$
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 4.12.5 Comparison of linear block codes and convolutional codes

Examples for Practice

Ex. 4.12.5 : *A rate ½, K = 3, binary convolutional encoder is shown in Fig. 4.12.23.*

a) Draw the tree diagram, trellis diagram and the state diagram for above encoder.

b) If the received signal at the decoder for eight message bits is,

$Y = (00\ 01\ 10\ 00\ 00\ 00\ 10\ 01)$

Trace the decision on a trellis or code tree diagram and find out message bit sequence.