$$p^3+p^2+0p+1 \overline{)\, p^6} \quad \text{quotient: } p^3+p^2+p$$

$$p^6+p^5+0p^4+p^3$$
$$p^5+0p^4+p^3$$
$$p^5+p^4+0p^3+p^2$$
$$p^4+p^3+p^2$$
$$p^4+p^3+0p^2+p$$
$$p^2+p \quad \leftarrow \textit{Remainder}$$

Here $R_1(p) = p^2 + p$

Hence first row polynomial will be,

$$p^6 + R_1(p) = p^6 + p^2 + p \Rightarrow \boxed{p^6+0p^5+0p^4+0p^3+p^2+p+0}$$

b) To obtain polynomial for Row 2 $(t=2)$

With $t = 2$, equation (3.3.72) becomes,

$\frac{p^{n-2}}{G(p)} \Rightarrow \frac{p^5}{p^3+p^2+1}$, This division is given below :

$$p^3+p^2+0p+1 \overline{)\, p^5} \quad \text{quotient: } p^2+p+1$$

$$p^5+p^4+0p^3+p^2$$
$$p^4+0p^3+p^2$$
$$p^4+p^3+0p^2+p$$
$$p^3+p^2+p$$
$$p^3+p^2+0p+1$$
$$p+1 \quad \leftarrow \textit{Remainder}$$

Here $R_2(p) = p + 1$

Hence second row polynomial will be,

$$p^5 + R_2(p) = p^5 + p + 1 \Rightarrow \boxed{0p^6+p^5+0p^4+0p^3+0p^2+p+1}$$

c) To obtain polynomial for Row 3, $(t=3)$

With $t = 3$, equation (3.3.72) becomes,

$\frac{p^{n-3}}{G(p)} = \frac{p^4}{p^3+p^2+1}$, this division is given below :

$$p^3+p^2+0p+1 \overline{)\, p^4} \quad \text{quotient: } p+1$$

$$p^4+p^3+0p^2+p$$
$$p^3+0p^2+p$$
$$p^3+p^2+0p+1$$
$$p^2+p+1 \quad \leftarrow \textit{Remainder}$$

Here $R_3(p) = p^2 + p + 1$

Hence third row polynomial will be,

$$p^4 + R_3(p) = p^4 + p^2 + p + 1 \Rightarrow \boxed{0p^6+0p^5+p^4+0p^3+p^2+p+1}$$

d) To obtain polynomial for Row 4, $(t=4)$

With $t = 4$, equation (3.3.72) becomes,

$\frac{p^{n-4}}{G(p)} = \frac{p^3}{p^3+p^2+1}$, this division is given below :

$$p^3+p^2+0p+1 \overline{)\, p^3} \quad \text{quotient: } 1$$

$$p^3+p^2+0p+1$$
$$p^2+0p+1 \quad \leftarrow \textit{Remainder}$$

Here $R_4(p) = p^2 + 0p + 1$

Hence fourth row polynomial will be,

$$p^3 + R_4(p)\; p^3+p^2+0p+1 \Rightarrow \boxed{0p^6+0p^5+0p^4+p^3+p^2+0p+1}$$

(e) To obtain generator matrix from row polynomials

Let us write the four row polynomials obtained in part (a) to (d) together. i.e.,

$t = 1 \Rightarrow 1^{st}$ row polynomial $= p^6+0p^5+0p^4+0p^3+p^2+p+0$

$t = 2 \Rightarrow 2^{nd}$ row polynomial $= 0p^6+p^5+0p^4+0p^3+0p^2+p+1$

$t = 3 \Rightarrow 3^{rd}$ row polynomial $= 0p^6 + 0p^5 + p^4 + 0p^3 + p^2 + p + 1$

$t = 4 \Rightarrow 4^{th}$ row polynomial $= 0p^6 + 0p^5 + 0p^4 + p^3 + p^2 + 0p + 1$

Now let us convert above polynomials into a matrix.

$$G = \begin{array}{c} \\ Row\,1 \\ Row\,2 \\ Row\,3 \\ Row\,4 \end{array} \begin{array}{c} \begin{array}{cccccccc} p^6 & p^5 & p^4 & p^3 & & p^2 & p^1 & p^0 \end{array} \\ \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & : & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & : & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & : & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & : & 1 & 0 & 1 \end{array}\right] \end{array} \quad \ldots (3.3.73)$$

ii) To obtain parity check matrix :

a) To obtain P submatrix

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

Here $k = 4$ and $q = 3 \Rightarrow G = [I_{4\times 4} : P_{4\times 3}]$

Comparing above equation with equation (3.3.73),

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

b) To obtain H^T

We know that,

$$H^T = \begin{bmatrix} P_{k\times q} \\ \ldots\ldots \\ I_{q\times q} \end{bmatrix}_{n\times q}$$

Putting values in above equation,

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

iii) Important conclusion and decoding table :

We derived generator matrix to get H^T. Because rows of H^T represent syndromes and error patterns. The rows of H^T can be written directly from remainder polynomials and identity matrix. This is illustrated in table 3.3.7. It is the decoding table.

Sr. No.	Remainder polynomial $R_i(p)$	Syndrome	Rows of H^T or identity matrix	Error pattern (E)						
1	$p^2 + p$	1 1 0, i.e.	1st row of H^T	1	0	0	0	0	0	0
2	$p + 1$	0 1 1, i.e.	2nd row of H^T	0	1	0	0	0	0	0
3	$p^2 + p + 1$	1 1 1, i.e.	3rd row of H^T	0	0	1	0	0	0	0
4	$p^2 + 0p + 1$	1 0 1, i.e.	4th row of H^T	0	0	0	1	0	0	0
5	–	1 0 0, i.e.	1st row of $I_{3\times 3}$	0	0	0	0	1	0	0
6	–	0 1 0, i.e.	2nd row of $I_{3\times 3}$	0	0	0	0	0	1	0
7	–	0 0 1, i.e.	3rd row of $I_{3\times 3}$	0	0	0	0	0	0	1

Table 3.3.7 Decoding table for $G(p) = p^3 + p^2 + 1$

Note : Above table shows that decoding table can be written directly from remainder polynomials.

iv) To obtain transmitted vectors

1) To decode $Y = 1101101$

The received vector polynomial will be,

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

Syndrome can be obtained by equation (3.3.62) as,

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

Hence let us divide $Y(p)$ by $G(p) = p^3 + p^2 + 1$.

$$\begin{array}{r} p^3 \\ p^3 + p^2 + 0p + 1 \overline{)\ p^6 + p^5 + 0p^4 + p^3 + p^2 + 0p + 1} \\ \underline{p^6 + p^5 + 0p^4 + p^3} \\ p^2 + 0p + 1 \leftarrow Remainder \end{array}$$

Thus syndrome polynomial is, $S(p) = p^2 + 0p + 1$

$\therefore \quad S = 101$

Syndrome is nonzero. Hence received vector contains errors. The syndrome of $S = 101$ corresponds to an error pattern of (see table 3.3.7),

$$E = 0\ 0\ 0\ 1\ 0\ 0\ 0$$

Correct codevector is,

$$\begin{aligned} X &= Y \oplus E \\ &= (1101101) \oplus (0001000) \\ &= 1100101 \end{aligned}$$

2) To decode $Y = 0101000$

$\therefore \quad Y(p) = p^5 + 0p^4 + p^3$

Let us divide $Y(p)$ by $G(p)$ i.e.,

$$\begin{array}{r} p^2 + p \\ p^3 + p^2 + 0p + 1 \overline{)\ p^5 + 0p^4 + p^3} \\ \underline{p^5 + p^4 + 0p^3 + p^2} \\ p^4 + p^3 + p^2 \\ \underline{p^4 + p^3 + 0p^2 + p} \\ p^2 + p \quad \leftarrow \textit{Remainder} \end{array}$$

Here $S(p) = p^2 + p$

$S = 1\ 1\ 0$

From table 3.3.7, $S = 110$ corresponds to

$$E = 1\ 0\ 0\ 0\ 0\ 0\ 0$$

$\therefore$ Correct codevector will be,

$$\begin{aligned} X &= Y \oplus E \\ &= (0101000) \oplus (1000000) \\ &= 1101000 \end{aligned}$$

3) To decode $Y = 0001100$

$$Y(p) = p^3 + p^2$$

$$\begin{array}{r} 1 \\ p^3 + p^2 + 0p + 1 \overline{)\ p^3 + p^2} \\ \underline{p^3 + p^2 + 0p + 1} \\ 1 \quad \leftarrow \textit{Remainder} \end{array}$$

Here $S(p) = 1$, Hence $S = 001$

From table 3.3.7, $S = 001$ corresponds to an error vector of,

$$E = 0\ 0\ 0\ 0 \quad 0\ 0\ 1$$

$\therefore$ Correct codevector will be,

$$\begin{aligned} X &= Y \oplus E \\ &= (0001100) \oplus (0000001) \\ &= 0001101 \end{aligned}$$

Results : In this example the codevector are systematic. Hence message vector (M) and check bits can be separated as shown below.

Sr no	Correct code vector, x	Message vector M	Check bits C
1	1100101	1100	101
2	1101000	1101	000
3	0001101	0001	101

Table 3.3.8 : Results of Ex. 3.3.14

Maximum likelihood decision rule based decoding is discussed in next section.

Example 3.3.17 : *A binary message sequence 1001 is coded using a generator polynomial* $G(x) = x^3 + x + 1$. *Assuming a systematic cyclic coding is used, determine -*

i) Length of coded sequence.

ii) The transmitted codeword.

iii) Assuming 4^{th} *bit in the received word is in error, workout the syndrome and draw the hardware block diagram for the syndrome generator.*

Solution : Given data

Generator polynomial, $G(p) = p^3 + p + 1$

Message sequence $M = 1001$

i) To determine length of the coded sequence

The degree of the generator polynomial is $q = 3$. Hence there are 3 check bits in coded sequence. Length of message sequence is $k = 4$. Therefore total number of bits in coded sequence will be,

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

This is $(7, 4)$ cyclic code.

(ii) To determine transmitted codeword

The check bit polynomial is given as,

$$C(p) = rem\left[\frac{p^q M(p)}{G(p)}\right] \quad \text{from equation (3.3.14)}$$

a) To obtain $p^q M(p)$

The message sequence is, $M = 1001$

$$M(p) = p^3 + 0p^2 + 0p + 1$$

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

$$p^q M(p) = p^3 (p^3 + 0p^2 + 0p + 1)$$

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

a) To divide $p^q M(p)$ by $G(p)$

The division of $p^q M(p)$ by $G(p)$ is shown below :

$$\begin{array}{r} p^3 + p \\ p^3 + 0p^2 + p + 1 \overline{\big)\ p^6 + 0p^5 + 0p^4 + p^3} \\ \underline{p^6 + 0p^5 + p^4 + p^3} \\ p^4 + 0p^3 \\ \underline{p^4 + 0p^3 + p^2 + p} \\ p^2 + p \leftarrow \textit{Remainder} \end{array}$$

Thus the remainder is, $C(p) = p^2 + p$

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

c) To obtain the codeword

The systematic form of the codeword is given as,

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

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

This is the required transmitted codeword.

ii) To determine syndrome for 4th bit in error

Here we have to obtain the decoding table. The table shows error pattern and corresponding syndrome vector. For the generator polynomial of $G(p) = p^3 + p + 1$ we have obtained the decoding table in Ex. 3.3.10. It is table 3.3.6. From this table observe that for 4th bit in error (i.e. E = 0 0 0 1 0 0 0), the syndrome vector is, $S = 011$ i.e. 4th row of H^T.

Here note that there is no need to obtain complete decoding table just for one syndrome.

How to obtain particular sundrome directly ?

In table 3.3.6 observe that first four syndromes are basically four rows of P-submatrix (see equation 3.3.70). Now see equation 3.3.52 carefully. These four rows of P-submatrix are part of generator matirx (G). i.e.,

From equation 3.4.52, $G =$

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

I_k matrix — P-submatrix

→ Indicates error in 1st bit
→ Indicates error in 2nd bit
→ Indicates error in 3rd bit
→ Indicates error in 4th bit

Now i^{th} syndrome can be obtained by dividing p^{n-i} by $G(p)$. Here $i = 4$. Hence divide p^{n-4} by $G(p)$. We have $n = 7$. Hence $p^{n-4} = p^3$.

$$\begin{array}{r} 1 \\ p^3 + 0p^2 + p + 1 \overline{\big)\ p^3} \\ \underline{p^3 + 0p^2 + p + 1} \\ p + 1 \leftarrow \textit{Remainder} \end{array}$$

The remainder $R_4(p) = p + 1$ represents syndrome.

i.e. $S = 0\ 1\ 1$

Note that this syndrome is same as that given in decoding table 3.3.6.

Important note :

Here $k = 4$. Hence first four syndromes can be obtained by above method. There are total $n = 7$ syndromes.

The remaining $q = 3$ syndromes are simply rows of the I_q matrix. i.e.,

$$I_{3\times3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} \leftarrow \text{indicates error in } 5^{th} \text{ bit} \\ \leftarrow \text{indicates error in } 6^{th} \text{ bit} \\ \leftarrow \text{indicates error in } 7^{th} \text{ bit} \end{matrix}$$

Hardware diagram for syndrome generator

The generator polynomial is,

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

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

On comparing above two equations,

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

With these values Fig. 3.3.4 can be redrawn as follows :

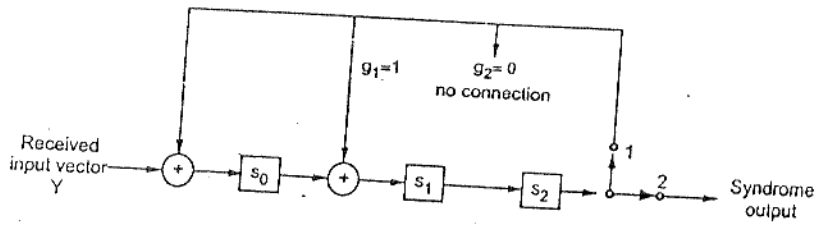


Fig. 3.3.13 Syndrome generator for (7,4) cyclic code for $G(p) = p^3 + p + 1$

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

Unsolved Examples

1. *A (15, 5) linear cyclic code has a generator polynomial*

   $G(p) = p^{10} + p^8 + p^5 + p^4 + p^2 + p + 1$

   i) Draw the block diagrams of encoder and syndrome calculator circuit for this code.

   ii) Find the code polynomial for the message polynomial

   $M(p) = p^4 + p^2 + 1$ *(in systematic form)*

   iii) Is $X(p) = p^{14} + p^8 + p^6 + p^4 + 1$ *a code polynomial ? If not, find the syndrome of X(p).*

   Ans. ii) $X(p) = p^{14} + p^{12} + p^{10} + p^9 + p^6 + p^2 + p + 1$

   iii) Syndrome $S(p) = p^9 + p^8 + p^7 + p^6 + p^3 + p$

2. *Using the generator polynomial* $g(x) = 1 + x^2 + x^3$, *generate systematic and non-systematic cyclic code words for the message vectors 1011 and 1001.*
3. *Sketch the encoder and syndrome calculator for the generator polynomial* $g(x) = 1 + x^2 + x^3$ *and obtain the syndrome for the received code word 1001011.*

3.4 Convolutional Codes

3.4.1 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. 3.4.1 shows a convolutional encoder.

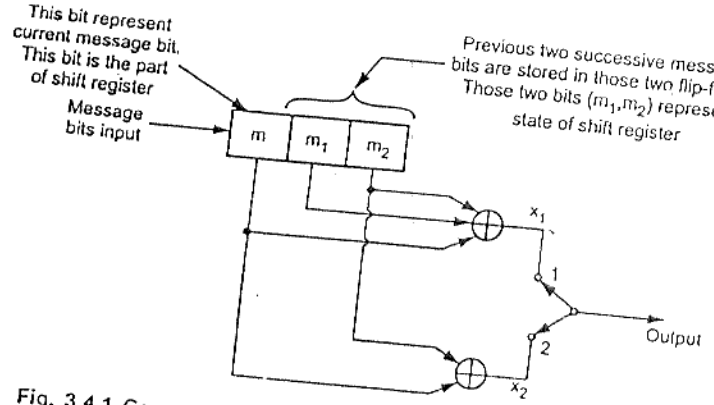


Fig. 3.4.1 Convolutional encoder with K = 3, k = 1 and n = 2

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,

$$x_1 = m \oplus m_1 \oplus m_2 \quad \ldots (3.4.1)$$

and

$$x_2 = m \oplus m_2 \quad \ldots (3.4.2)$$

The output switch first samples x_1 and then x_2. The shift register then shifts contents of m_1 to m_2 and contents of m to m_1. Next input bit is then taken and stored in m. Again x_1 and x_2 are generated according to this new combination of m, m_1 and m_2 (equation (3.4.1) and equation (3.4.2)). The output switch then samples x_1 then x_2. Thus the output bit stream for successive input bits will be,

$$X = x_1x_2x_1x_2x_1x_2 \ldots \text{ and so on} \quad \ldots (3.4.3)$$

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

Number of message bits, k = 1

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

3.4.1.1 Code Rate of Convolutional Encoder

The code rate of this encoder is,

$$r = \frac{k}{n} = \frac{1}{2} \quad \ldots (3.4.4)$$

In the encoder of Fig. 3.4.1, observe that whenever a particular message bit enters a shift register, it remains in the shift register for three shifts i.e.,

First shift → 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 overv[illegible]. 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.

3.4.1.2 Constraint Length (K)

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

For the encoder of Fig. 3.4.1 constraint length K = 3 bits. This is because in this encoder, a single message bit influences encoder output for three successive shifts. At the fourth shift, the message bit is lost and it has no effect on the output.

3.4.1.3 Dimension of the Code

The dimension of the code is given by n and k. We know that 'k' is the number of message bits taken at a time by the encoder. And 'n' is the encoded output bits for one message bits. Hence the dimension of the code is (n, k). And such encoder is called (n, k) convolutional encoder. For example, the encoder of Fig. 3.4.1 has the dimension of (2, 1).

3.4.2 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. 3.4.1 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. 3.4.1. These impulse responses are also called *generator sequences* of the code.

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

$$x_{1,i} = x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l} \quad i = 0, 1, 2 \ldots\ldots \quad \ldots (3.4.5)$$

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

$$x_2 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} m_{i-l} \quad i = 0, 1, 2 \ldots\ldots \quad \ldots (3.4.6)$$

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

As shown in the Fig. 3.4.1, the two sequences x_1 and x_2 are multiplexed by the switch. Hence the output sequence is given as,

$$\{x_i\} = \left\{ x_0^{(1)} x_0^{(2)} x_1^{(1)} x_1^{(2)} x_2^{(1)} x_2^{(2)} x_3^{(1)} x_3^{(2)} \ldots\ldots \right\} \quad \ldots (3.4.7)$$

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

---

* Some authors define constraint length as number of output bits influenced by a single message bit i.e.

Constraint length (k) = (n × M) bits ... (3.4.8)

where n = number of encoded output bits for every input bit

and M = number of storage elements in the shift register

For the encoder of Fig. 3.5.1 Constraint length = 2 × 3 = 6 bits.

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

Observe that bits from above two sequences are multiplexed in equation (3.4.7) The sequence $\{x_i\}$ is the output of the convolutional encoder.

Example 3.4.1 : *For the convolutional encoder of Fig. 3.4.2 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}

(Nov./Dec.-2003, 8 Marks)

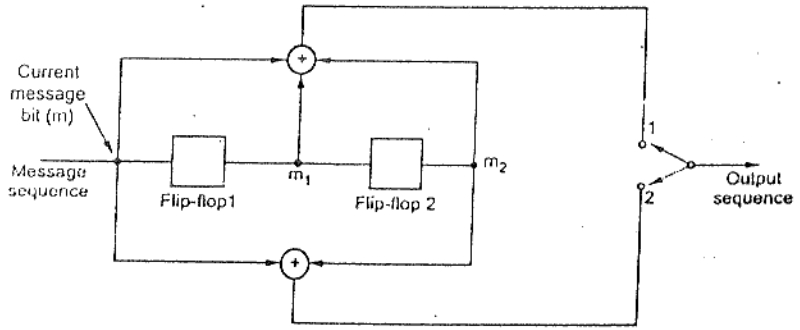


Fig. 3.4.2 Convolutional encoder of Ex. 3.4.1

Solution : In the Fig. 3.4.2 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. 3.4.3.

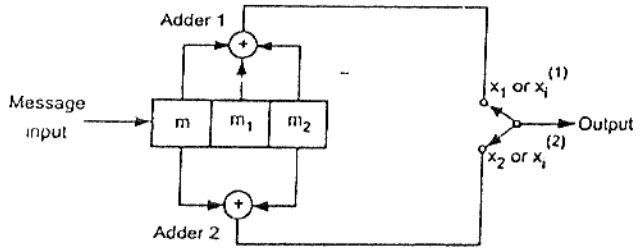


Fig. 3.4.3 Convolutional encoder of Fig. 3.4.1 redrawn alternately

Observe that the above encoder is exactly similar to that of Fig. 3.4.1.

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,

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. 3.4.3 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

$$g_i^{(1)} = \{1\ 1\ 1\} \qquad \ldots (3.4.9)$$

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

x_2 i.e. $x_i^{(2)}$ is generated by additi

sequence is given as,

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

Here $g_0^{(2)} = 1$ represents co

$g_1^{(2)} = 0$ rep.esents tha

$g_2^{(2)} = 1$ represents conr

The above sequences are also called impı

To obtain output sequence

The given message sequence is,

$$m = (m_0\ m_1\ m_2\ m_3\ m_4) :$$

Thus the output of ad

To obtain output due to adder 1

Then from equation (3.4.6) we can write,

$$x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} m_{i-l} \quad \ldots (3.4.11)$$

with $i = 0$ above equation becomes,

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

$$x_0^{(1)} = g_0^{(1)} m_0$$

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

$i = 1$ in equation (3.4.11), $x_1^{(1)} = g_0^{(1)} m_1 \oplus g_1^{(1)} m_0$

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

Here note that additions are mod-2 type.

$i = 2$ in equation (3.4.11), $x_2^{(1)} = g_0^{(1)} m_2 + g_1^{(1)} m_1 + g_2^{(1)} m_0$

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

$i = 3$ in equation (3.4.11), $x_3^{(1)} = g_0^{(1)} m_3 \oplus g_1^{(1)} m_2 \oplus g_2^{(1)} m_1$

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

$$= 1$$

$i = 4$ in equation (3.4.11), $x_4^{(1)} = g_0^{(1)} m_4 \oplus g_1^{(1)} m_3 \oplus g_2^{(1)} m_2$

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

$i = 5$ in equation (3.4.11), $x_5^{(1)} = g_0^{(1)} m_5 \oplus g_1^{(1)} m_4 \oplus g_2^{(1)} m_3$

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

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

$$= 0$$

uation (3.4.11), $x_6^{(1)} = g_0^{(1)} m_6 \oplus g_1^{(1)} m_5 \oplus g_2^{(1)} m_4$

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

$$= 1 \times 1$$

$$= 1$$

der 1 is,

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

To obtain output due to adder 2

Similarly from equation (3.4.7),

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

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

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

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

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

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

$$= 0$$

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

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

$$= 1$$

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

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

$$= 1$$

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

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

$$= 1$$

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

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

$$= 1$$

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

$$= 1 \times 1$$

$$= 1$$

Thus the sequence x_2 is,

$$x_2 = x_i^{(2)} = \{1\ 0\ 1\ 1\ 1\ 1\ 1\} \quad \ldots (3.4.12)$$

To obtain multiplexed sequence of x_1 and x_2 as per equation (3.4.8)

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

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

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

3.4.3 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.,

$$g^{(1)}(p) = g_0^{(1)} + g_1^{(1)}p + g_2^{(1)}p^2 + \ldots\ldots + g_M^{(1)}p^M \quad \ldots (3.4.13)$$

Similarly,

$$g^{(2)}(p) = g_0^{(2)} + g_1^{(2)}p + g_2^{(2)}p^2 + \ldots\ldots + g_M^{(2)}p^M \quad \ldots (3.4.14)$$

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

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

$$m(p) = m_0 + m_1p + m_2p^2 + \ldots\ldots + m_{L-1}p^{L-1} \quad \ldots (3.4.15)$$

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

$$\left.\begin{aligned} x^{(1)}(p) &= g^{(1)}(p) \cdot m(p) \\ x^{(2)}(p) &= g^{(2)}(p) \cdot m(p) \end{aligned}\right\} \quad \ldots (3.4.16)$$

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

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

Example 3.4.2 : *Repeat part (V) of example 3.4.1 using transform domain calculations (polynomial-multiplications).*

Solution : **a) To obtain generating polynomial for adder-1 :**

The first generating sequence is given by equation (3.4.9) i.e.,

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

Hence its polynomial can be obtained (equation 3.4.13) as follows :

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

$$= 1 + p + p^2 \quad \ldots (3.4.17)$$

b) To obtain generating polynomial for adder-2

The second generating sequence is given by equation (3.4.10). i.e.,

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

Hence its polynomial can be obtained (equation 3.4.14) as follows :

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

$$= 1 + p^2 \quad (3.4.18)$$

c) To obtain message polynomial

The message sequence is,

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

Hence its polynomial can be obtained (equation 3.4.15) as,

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

$$= 1 + p^3 + p^4 \quad \ldots (3.4.19)$$

d) To determine the output due to adder-1

Now $x^{(1)}(p)$ can be obtained from equation (3.4.16) 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.

3.4.4 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. 3.4.1. It is reproduced below for convenience.

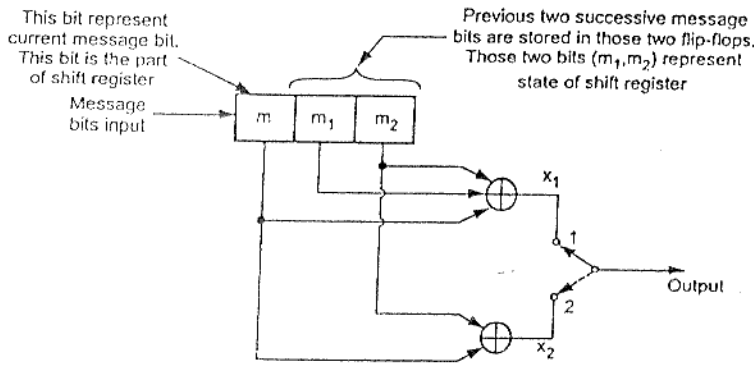


Fig. 3.4.4 Convolutional encoder with k = 1 and n = 2

3.4.4.1 States of the Encoder

In above figure 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 3.4.1.

Let the initial values of bits stored in m_1 and m_2 be zero. That is $m_1m_2 = 00$ 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 3.4.1 States of the encoder of Fig. 3.4.4

3.4.4.2 Development of the Code Tree

Let us consider the development of code tree 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.

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.

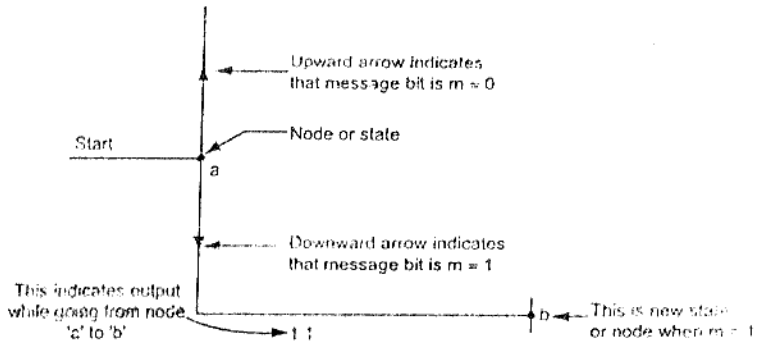


Fig. 3.4.5 Code tree from node 'a' to 'b'

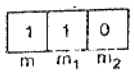
Thus the new state of encoder is $m_1 m_2 = 01$ or 'b' and output transmitted are $x_1x_2 = 11$. This shows that if encoder is in state 'a' and if input is $m = 1$ then the next state is 'b' and outputs are $x_1x_2 = 11$. The first row of Table 3.4.2 illustrates this operation.

The last column of this table shows the code tree diagram. The code tree diagram starts at node or state 'a'. The diagram is reproduced as shown in Fig. 3.4.5.

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

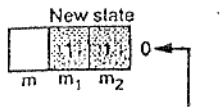
Now let the second message bit be 1. The contents of shift register with this input will be as shown below.



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

$x_2 = 1 \oplus 0 = 1$

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 3.4.2 in second row. The last column of the table shows the code tree for those first and second input bits.

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

Similarly 3rd row of the Table 3.4.2 (Refer table on next page) illustrated the operation of encoder for 3rd input message bit as m = 0. Now observe in the code tree of last column. Since input bit is m = 0, the path of the tree is shown by upward arrow towards node (or state) 'c'. That is the next state is 'c' (i.e. 10) and output is $x_1x_2 = 01$.

Complete code tree for convolutional encoder

Fig. 3.4.6 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. 3.4.6, you will find that the branch pattern begins to repeat after third bit. This is shown in figure. The repetition starts after 3rd bit, since particular message bit is stored in the shift registers of the encoder for three shifts. If the length of the shift register is increased by one bit, then the pattern of code tree will repeat after fourth message bit. (Please refer Fig. 3.4.6)

3.4.4.3 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. 3.4.7 shows code trellis diagram.

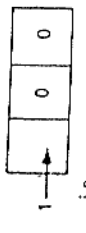
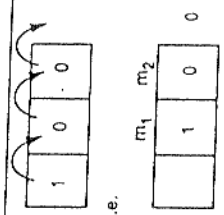
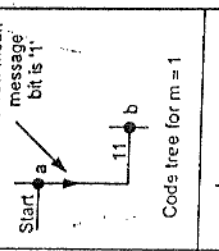
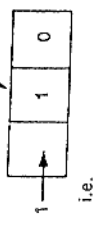
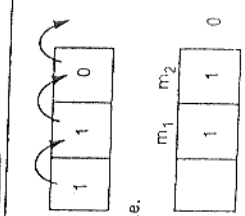
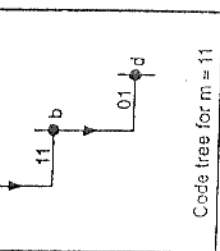
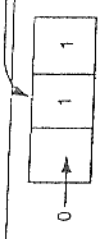
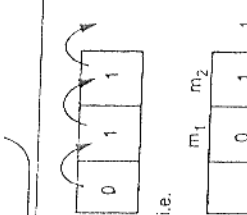
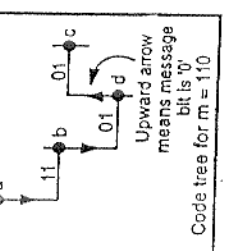
Sr. No.	Input message bit M	Status of shift register after entry of m	Calculation of outputs x_1 and x_2	Status of shift register after transmission of o/p and shift right by one bit	New state of encoder m_2 m_1	Transmitted outputs x_1 x_2	Code Tree diagram Downward arrow indicated input is 1
1	1		$x_1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

Table 3.4.2 Analysis of convolutional encoder of Fig. 3.4.4

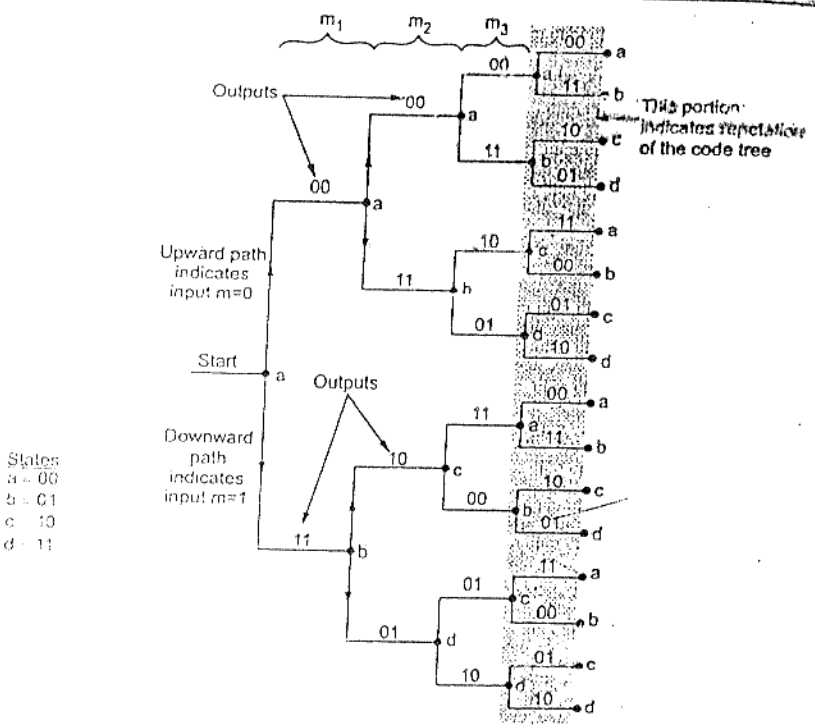


Fig. 3.4.6 Code tree for convolutional encoder of Fig. 3.4.5

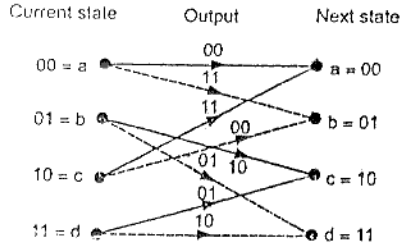


Fig. 3.4.7 Code trellis of convolutional encoder of Fig. 3.4.1

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.

3.4.4 State Diagram

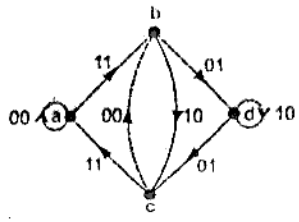


Fig. 3.4.8 State diagram for convolutional encoder of Fig. 3.4.1

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

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

Comparison between code tree and trellis diagram :

Table 3.4.3 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 from current to next states.
2	Code tree is lengthy way of representing coding process.	Code trellis diagram is shorter or compact way of representing coding process
3	Decoding is very simple using code tree.	Decoding is little complex using trellis diagram.
4	Code tree repeats after number of stages used in the encoder.	Trellis diagram repeats in every state. In steady state, trellis diagram has only stage
5	Code tree is complex to implement in programming.	Trellis diagram is simpler to implement in programming.

Table 3.4.3 Comparison between code tree and trellis diagram

3.4.5 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.

3.4.5.1 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. 3.4.1. For this encoder we have obtained code trellis in Fig. 3.4.7. 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 $(x_1 x_2)$ outputs. These outputs are received at the decoder and represented by Y. Thus Y given above represents the outputs for three successive message bits. Assume that the decoder is at state a_0. Now look at the code trellis diagram of Fig. 3.4.7 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.3.4.9. 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

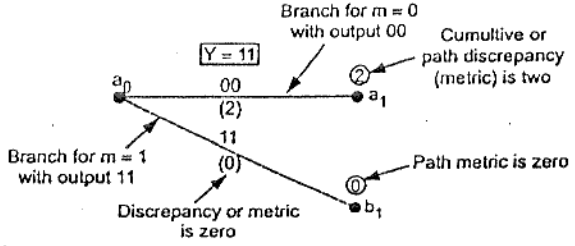


Fig. 3.4.9 Viterbi decoder results for first message bit

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

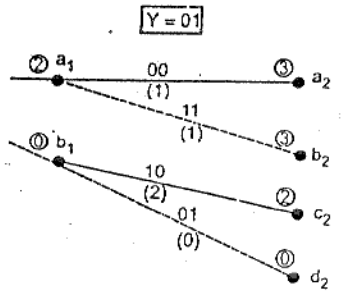


Fig. 3.4.10 Viterbi decoder results for second message bit

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. 3.4.10 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.

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

Fig. 3.4.11 shows the trellis diagram for all the six bits of Y.

Fig. 3.4.11 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. 3.4.11, the paths marked with × (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.

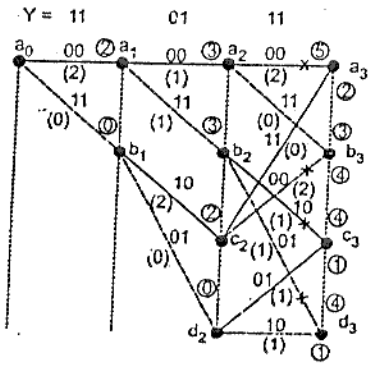


Fig. 3.4.11 Paths and their metrics for viterbi decoding

d) Further explanation of viterbi decoding for 12 message bits

Fig. 3.4.12 shows the continuation of Fig. 3.4.11 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.

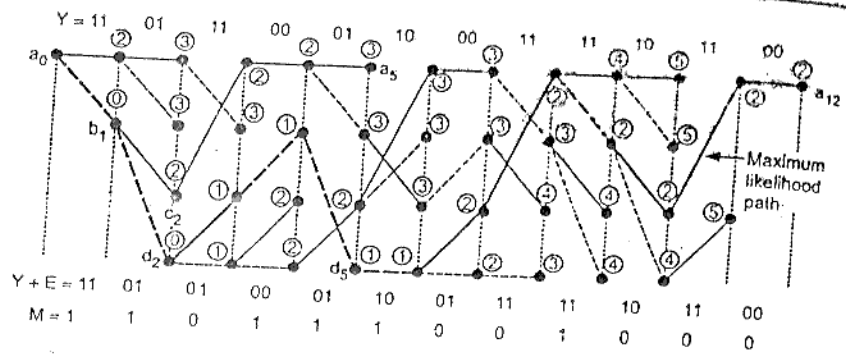


Fig. 3.4.12 Viterbi decoding

Only one path of particular node is kept which is having lower metric. In case if there are two paths have 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 survivity paths for four nodes

$$\text{Surviving paths} = 2^{(K-1)k} \qquad \ldots (3.4.20)$$

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

For the encoder for Fig. 3.4.1 $K = 3$ 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. 3.4.10. 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.

3.4.5.2 Sequential Decoding for Convolutional Codes

Sequential decoding uses metric divergence effect Fig. 3.4.13 (a) shows the code trellis for the convolutional encoder of Fig. 3.4.1. 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. 3.4.13 (a), the path for first three nodes is $a_0 b_1 d_2$ since its metric is the lowest.

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. 3.4.13 (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,

$$\text{Running metric} = jn\alpha \qquad \ldots (3.4.21)$$

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 $(jn\alpha + \Delta)$. Here Δ is the should be above $jn\alpha$ at j^{th} node. Fig. 3.4.13 (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 '2' 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. 3.4.13 (b) the value of $\alpha = 1/16$, for encoder of Fig. 3.4.1 we know that $n = 2$. Let's calculate $jn\alpha$ at 8^{th} node.

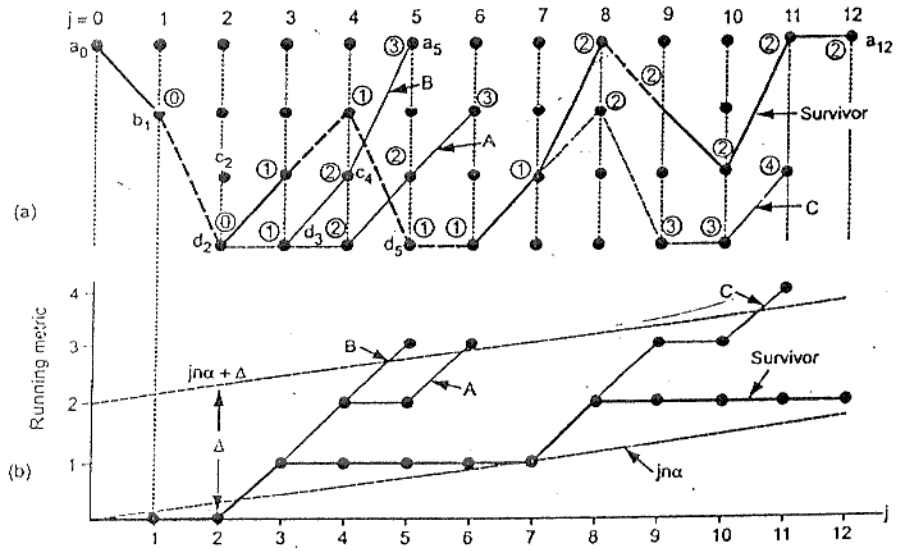


Fig. 3.4.13 Sequential decoding

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.

3.4.5.3 Free Distance and Coding Gain

For the block and cyclic codes we have seen that the error correction or detection power depends upon the minimum distance between the code vectors. A convolutional encoder does not divide the output encoded signal into different code vector, but complete transmitted sequence can be considered as a single code vector. Let X represent the transmitted sequence. We know that minimum distance between the code vector is equal to minimum weight of the code vector. Therefore a free

distance is equal to the minimum distance between the code vectors. Since minimum distance is equal to minimum weight of the code vector we can write,

Free distance (d_f) = Minimum distance between code vectors

= Minimum weight of the code vectors

i.e. $d_f = [w(X)]_{min}$ and X is non zero ... (3.4.22)

Here $[w(X)]_{min}$ is the minimum weight of code vector. For convolutional coding free distance (d_f) represents the error control power.

Coding Gain :

Coding gain is used as a basis of comparison for different coding methods. To achieve the same bit error rate the coding gain is defined as,

$$A = \frac{\left(\frac{E_b}{N_0}\right) Encoded}{\left(\frac{E_b}{N_0}\right) coded} \quad \ldots (3.4.23)$$

For convolutional coding the coding gain is given as,

$$A = \frac{r\, d_f}{2} \quad \ldots (3.4.24)$$

here 'r' is the code rate and

d_f is the free distance.

3.4.6 Probability of Errors for Soft and Hard Decision Decoding

In this subsection we will study the error rate performance of the viterbi algorithm for the additive white gaussian noise channel with soft decision and hard decision decoding.

3.4.6.1 Probability of Error with Soft Decision Decoding

Let us consider that the all zero sequence is transmitted and we calculate the probability of error for detector deciding in favour of another sequence. Let the coded binary digits for the j^{th} branch of the convolutional code be represented as c_{jm}, m=1,2, ...n. The input to the viterbi decoder be the sequence, r_{jm}, m = 1, 2, ...n and j = 1, 2, ... Here r_{jm} is given as,

$$r_{jm} = \sqrt{E_c}\,(2c_{jm} - 1) + n_{jm} \quad \ldots (3.4.25)$$

Here c_{jm} represent the transmitted bits. j indicates the j^{th} branch and m indicates m^{th} bit in that branch. E_c is the transmitted signal energy for each code bit and n_{jm} is the additive noise.

The viterbi soft decision decoder calculates also branch metrics by following relation :

$$\mu_i^{(i)} = \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1) \quad \ldots (3.4.26)$$

Hence i represents the 1^{th} path and $\mu_{jm}^{(i)}$ is the metric of j^{th} branch in i^{th} path. The viterbi decoder then obtains the path metrics as,

$$CM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)}$$

$$= \sum_{j=1}^{B} \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1) \quad \ldots (3.4.27)$$

Here i is the i^{th} path and B is the branches in that path.

The convolutional code does not have a fixed length. Hence we will derive its performance from the probability of error for the sequences that merge with the all zero sequence for the first time at a given node in the trellis.

The probability of error when another path merges with all zero path is equivalent to probability that metric of this path exceeds the metric of all zero path for the first time. i.e.,

$$P_2(d) = P(CM^{(1)} \geq CM^{(0)})$$

$$= P(CM^{(1)} - CM^{(0)})$$

putting for $CM^{(1)}$ and $CM^{(0)}$ in above equation from equation (3.4.27) we get,

$$P_2(d) = P\left\{2\sum_{j=1}^{B}\sum_{m=1}^{n} r_{jm}(c_{jm}^{(1)} - c_{jm}^{(0)})\right\} \quad \ldots (3.4.28)$$

Here the difference $c_{jm}^{(1)} - c_{jm}^{(0)}$ is equal to 1 at the bit positions which are in error because of incorrect path with respect to all zero path. Thus the above equation is the probability of such non zero bits at 'd' positions in the incorrect path. Hence above equation can be written in the simplified form as,

$$P_2(d) = P\left\{\sum_{l=1}^{d} r_l' \geq 0\right\} \quad \ldots (3.4.29)$$

Here $\{r_l'\}$ are the set of [illegible] bits at 'd' positions. $\{r_l'\}$ have gaussian distribution with $-\sqrt{E_c}$ mean and $\frac{1}{2}N_0$ variance. This is because these bits are basically error bits. Hence the above probability equation can be written as,

$$P_2(d) = Q\left\{\sqrt{\frac{2E_c}{N_0}d}\right\} \quad \ldots (3.4.30)$$

This equation gives probability of error in the pairwise comparison of the two paths which differs in 'd' bits. Since $\gamma_b = \frac{E_b}{N_0}$ and $R_c = \frac{E_c}{E_b}$, the above equation can be written as,

$$P_2(d) = Q(\sqrt{2\gamma_b R_c d}) \quad \ldots (3.4.31)$$

This is the probability of path of distance 'd' from the all zero path. Actually there will be many such paths with different distances and they merge with all zero path at given node B. The first event error probability can be obtained by summing the error probabilities over all the possible path distances. Hence the upper bound on first event error probability will be,

$$P_e \leq \sum_{d=d_{free}}^{\infty} a_d P_2(d) \leq \sum_{d=d_{free}}^{\infty} a_d Q(\sqrt{2\gamma_b R_c d}) \quad \ldots (3.4.32)$$

Here a_d is the number of paths of distance 'd' from the all zero path which merge with all zero path for the first time.

Hence the above expression is called first event error probability. This first event error probability provides the measure of the performance of convolutional code. The bit error probability is more useful performance measure. The bit error probability is upper bounded by first event error probability.

3.4.6.2 Probability of Error with Hard Decision Decoding

The binary symmetric channel uses hard decision decoding. Here let us consider the performance of viterbi algorithm for hard decision decoding.

Consider that the all zero path is transmitted, and the path which is selected has distance 'd' from the all zero path. With hard decision decoding, the probability that incorrect path is selected is given as,

$$P_2(d) = \sum_{k=\frac{(d+1)}{2}}^{d} \binom{d}{k} p^k (1-p)^{d-k} \quad \ldots (3.4.33)$$

Here p is the probability of error in the binary symmetric channel. The union bound on the first event error probability that all the possible paths merge with all zero path at a given node is,

$$P_e < \sum_{d=d_{free}}^{\infty} a_d P_2(d) \quad \ldots (3.4.34)$$

Here $\{a_d\}$ represents the number of paths corresponding to the set of distances d. The bit error probability can be more useful than the first event error probability. The union bound given by equation (3.4.34) is the upper bound on bit error probability.

Comparison between hard decision decoding and soft decision decoding :

We studied hard decision decoding and soft decision decoding for block codes and convolutional codes. Following table 3.4.4 lists the comparison of them.

Sr. No.	Parameter	Hard decision decoding	Soft decision decoding
1	Principle	The decoder operates on hard decisions made by the demodulator. This decoding is called hard decision decoding.	The demodulator output is quantized into more than two levels. Hence decoder operates on soft decisions made by demodulator. It is called soft decision decoding.
2	Preferred type of channel	Binary symmetric channel.	Gaussian channel.
3	Correct / incorrect decisions	The decisions can be labeled as correct or incorrect.	The decisions cannot be directly labeled as correct or incorrect.
4	Probability of error	Symbol error probability can be calculated directly.	Likelihood of symbol types can be expressed. Hence conditional probabilities are normally written.
5	Complexity of decoders	Implementation of decoders is simple.	Implementation of decoders is complex.
6	Preferred code	Block codes and convolutional codes use hard decision decoding.	Convolutional codes use soft decision decoding.

Table 3.4.4 Comparison between hard decision decoding and soft decision decoding

Example 3.4.3 : *The figure below depicts a rate 1/2, constraint length $N = 2$ convolutional code encoder. Sketch the code tree for the same.*

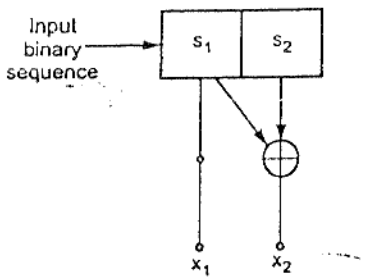


Fig. 3.4.14 Convolutional encoder

Solution : a) Define states of encoder

The constraint length of the given convolutional encoder is $K = 2$. Its rate is $\frac{1}{2}$ means for single message bit input, two bits x_1 and x_2 are encoded at the output. 's_1' represents the input message bit and s_2 stores the previous message bit. Since only one previous message bit is stored, this encoder can have states depending upon this stored message bit. Let's represent,

$s_2 = 0$ state 'a'

and $s_2 = 1$ state 'b'

b) Outputs of encoder

Let's assume that the contents of s_1 and s_2 are zero initially. From encoder of Fig. 3.4.14 it is clear that outputs x_1 and x_2 are given as,

$$\left.\begin{aligned} x_1 &= s_2 \\ x_2 &= s_2 \oplus s_2 \end{aligned}\right\} \quad \ldots (3.4.35)$$

c) Prepare state diagram

Before drawing code tree, we will first prepare the state diagram for this encoder. State diagram represents the compact version of code tree. Table 3.4.5 shows the present and next states corresponding to different inputs of message signal. First row of the table shows that if present state of encoder (which is defined by s_2) is 'a' and if input s_1 (i.e. m) = 0, then outputs $x_1\, x_2 = 00$ and next state of the encoder will be 'a'. Second row of the table shows that in present state of encoder is 'a' and input is 1 then contents of $s_1\, s_2 = 10$. Then the output $x_1\, x_2 = 11$ and next state of encoder is 'b'. Similarly other two rows represent how encoder operates if it's present state is 'b'. (Please refer Table 3.4.5 on next page.)

Based on the result of Table 3.4.5, we can draw the state diagram as shown in Fig. 3.4.15.

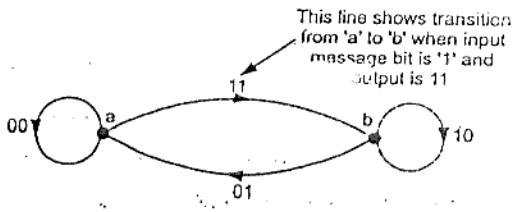


Fig. 3.4.15 State diagram for the encoder of Fig. 3.4.14. Continuous line represent input as '0' and dotted line represents input as '1'

Input message bit m	Status of shift register	Present state	Output x_1 x_2	Next state after transmitting outputs	Next state
0	s_1 s_2 : 0 0	a	$x_1 = 0$ $x_2 = 0$	0 0	a
	i.e. s_1 s_2 : 0 0			i.e. s_1 s_2 : 0 0 — 0	
1	s_1 s_2 : 1 0	a	$x_1 = 1$ $x_2 = 1$	1 0	b
	i.e. s_1 s_2 : 1 0			i.e. s_1 s_2 : 1 0 — 0	
1	s_1 s_2 : 1 1	b	$x_1 = 1$ $x_2 = 0$	1 1	b
	i.e. s_1 s_2 : 1 1			i.e. s_1 s_2 : 1 0 — 1	
0	s_1 s_2 : 0 1	b	$x_1 = 0$ $x_2 = 1$	0 1	a
	i.e. s_1 s_2 : 0 1			i.e. s_1 s_2 : 0 0 — 1	

Table 3.4.5 Operation of encoder of Fig. 3.4.14

As shown in state diagram, above encoder remains in state '*a*' if input is zero. In the above diagram continuous lines are used when input message bit is '0' and dotted lines are used when message bit is '1'. The arrow on the line shows the transition

towards next state. The numbers marked on line represent outputs x_1 x_2 during that transition.

(d) To obtain code tree

The code tree is derived from the state diagram is shown in Fig. 3.4.15. Observe that the branch pattern of the code tree repeats after two successive message bits. This is because any message bit remains stored in the encoder register for two successive shifts.

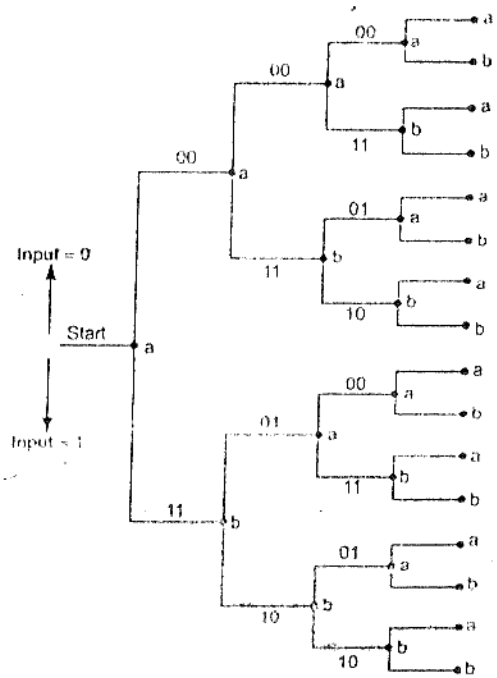


Fig. 3.4.16 Code tree for encoder of Fig. 3.4.14

Example 3.4.4 : *For the convolutional encoder arrangement shown in Fig. 3.4.17, draw the state diagram and hence trellis diagram. Determine output digit sequence for the data digits 1 1 0 1 0 1 0 0. What are the dimensions of the code (n, k) and constraint length ? Use viterbi's algorithm to decode the sequence, 100 110 111 101 001 101 001 010.*

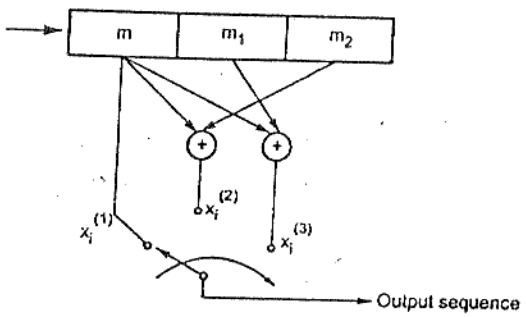


Fig. 3.4.17 Encoder of Ex. 3.4.4

Solution : **i) To obtain dimension of the code :** Observe that one message bit is taken at a time in the encoder of Fig. 3.4.17. Hence k = 1. There are three output bits for every message bit. Hence n = 3. Therefore the dimension of the code is (n, k) = (3, 1).

ii) Constraint length

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

Constraint length K = 3 bits.

iii) To obtain code trellis and state diagram

Let the states of the encoder be as given in Table 3.4.1. Fig. 3.4.18 shows the code trellis of the given encoder.

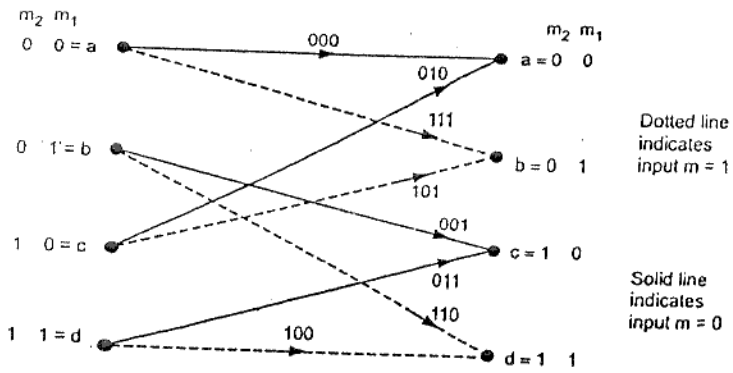


Fig. 3.4.18 Code trellis of encoder of Fig. 3.4.17

The nodes in the code trellis can be combined to form state diagram as shown in Fig. 3.4.19.

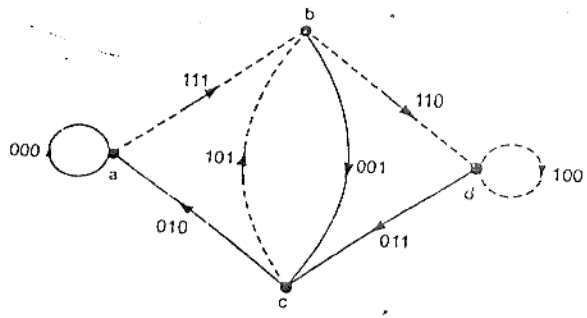


Fig. 3.4.19 State diagram of the encoder of Fig. 3.4.17

iv) To determine output sequence

a) Determine generator polynomials

The generating sequence can be written for $x_i^{(1)}$ from Fig. 3.4.17 as,

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

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

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

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

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

Hence the corresponding generating polynomials can be written as,

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

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

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

b) Determine message polynomials

The given message sequence is,

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

Hence the message polynomial will be,

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

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

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

$$\begin{aligned} x_i^{(1)} &= g^{(1)}(p) \cdot m(p) \\ &= 1\,(1+p+p^3+p^5) \\ &= 1+p+p^3+p^5 \end{aligned}$$

Hence the corresponding sequence will be,

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

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

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

$$\begin{aligned} x_i^{(2)} &= g^{(2)}(p) \cdot m(p) \\ &= (1+p^2)\,(1+p+p^3+p^5) \\ &= 1+p+p^2+p^7 \end{aligned}$$

Hence the corresponding sequence will be,

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

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

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

$$\begin{aligned} x_i^{(3)} &= g^{(3)}(p) \cdot m(p) \\ &= (1+p) \cdot (1+p+p^3+p^5) \\ &= 1+p^2+p^3+p^4+p^5+p^6 \end{aligned}$$

Hence the corresponding sequence is,

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

f) To multiplex three output sequences

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

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

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

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

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

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

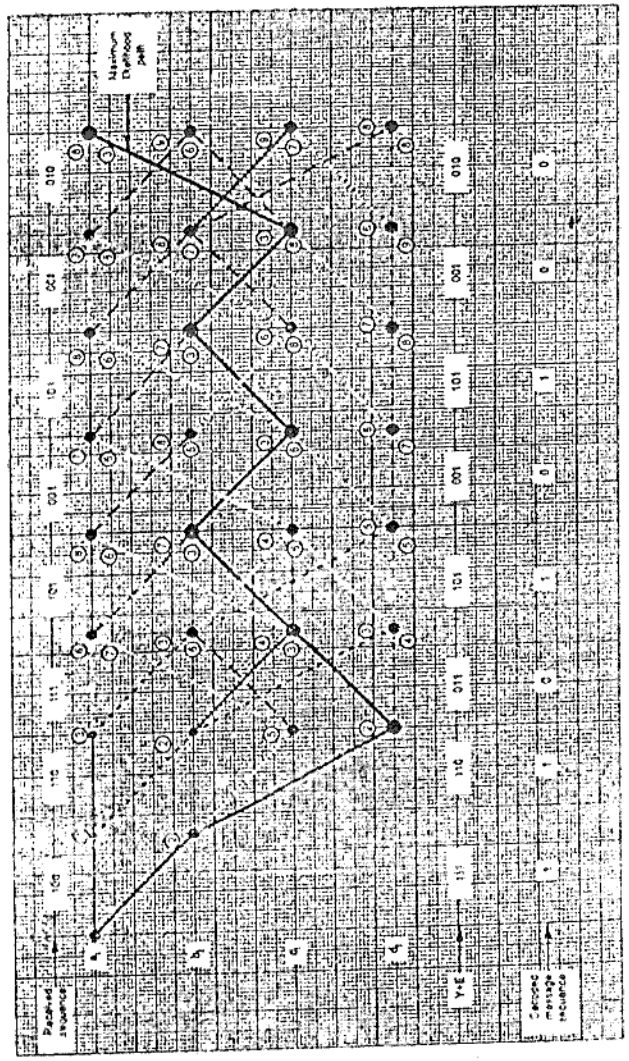


Fig. 3.4.20 Viterbi decoding for example 3.4.4

v) Viterbi algorithm to decode the given sequence

Fig. 3.4.20 (See Fig. on previous page) shows the diagram based on viterbi decoding. It shows received sequence at the top. The decoded (Y + E) sequence and decoded message sequence is shown at the bottom.

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

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

Example 3.4.5 : *A convolutional encoder has single shift register with two stages three modulo-2 adders and an output multiplexer. The following generator sequences are combined by the multiplexer to produce the encoder output -*

$g_1 = (1,0,1)$; $g_2 = (1,1,0)$; $g_3 = (1,1,1)$

i) Draw block digram of the encoder.

ii) For the message sequence (10011), determine encoded sequence.

If above hardware is enhanced by increasing number of stages in shift register and number of mod-2 adders respectively, what is the effect on

a) Generated ouput sequence

b) Periodicity of the codetree.

Solution : i) To obtain block diagram of the encoder

The shift register is two stage but every output g_1, g_2 and g_3 combines three inputs. Fig. 3.4.21 shows the encoder.

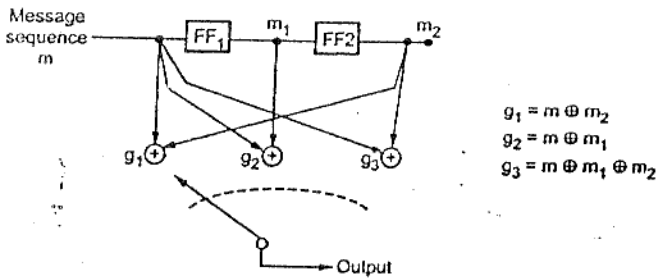


Fig. 3.4.21 Block diagram of the convolutional of Ex. 3.4.5

ii) To obtain output sequence for m = (1 0 0 1 1)

a) Obtain generator polynomials

The polynomials of g_1, g_2 and g_3 can be written as,

$$g_1 = (101) \Rightarrow g_1(p) = 1 + p^2$$
$$g_2 = (110) \Rightarrow g_2(p) = 1 + p$$
$$g_3 = (111) \Rightarrow g_3(p) = 1 + p + p^2$$

b) Obtain message polynomial

The message polynomial becomes,

$$m = (10011) \Rightarrow m(p) = 1 + p^3 + p^4$$

c) Output sequence due to g_1

Output of sequence g_1 is given as,

$$x_1(p) = g_1(p)m(p) = (1 + p^2)(1 + p^3 + p^4)$$
$$= 1 + p^2 + p^3 + p^4 + p^5 + p^6$$

Hence $x_1 = (1011111)$

d) Output sequence due to g_2

Similarly output of g_2 is given as,

$$x_2(p) = g_2(p)m(p) = (1 + p)(1 + p^3 + p^4)$$
$$= 1 + p + p^3 + p^5$$

Hence $x_2 = (1\ 1\ 0\ 1\ 0\ 1)$

e) Output sequence due to g_3

Output of g_3 is given as,

$$x_3(p) = g_3(p)m(p) = (1 + p + p^2)(1 + p^3 + p^4)$$
$$= 1 + p + p^2 + p^3 + p^6$$

Hence $x_3 = (1111001)$

f) Multiplexing the sequences due to g_1, g_2 and g_3

The multiplexer will multiplex the bits of x_1, x_2 and x_3 as follows :

Output sequence = (1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1)

Note that x_2 contains only 6 output digits. Hence its 7th output digit is assumed zero in above multiplexed sequence.

If hardware is enhanced by adding shift registers and adders

a) Effect on generated output sequence

For each input message bit, three output bits are generated (See Fig. 3.4.21). There are three mod-2 adders in the encoder. Therefore three output bits are generated. *If mod-2 adders are increased, then output bits for every message bit are increased.* Therefore length of the coded sequence increases.

b) Effect on periodicity of code tree

Periodicity of codetree is related to number of stages in the shift register. In Fig. 3.4.21, observe that three message bits are present at any time in shift register. Hence code tree will be periodic after fourth message bit. If number of stages are increased, then period of code tree also increases.

3.4.7 Transfer Function of the Convolutional Code

The state diagram of the convolutional code gives information about distance properties and error rate performance. Consider the state diagram shown in Fig 3.4.22. We will use this state diagram to obtain the distance properties of the convolutional code.

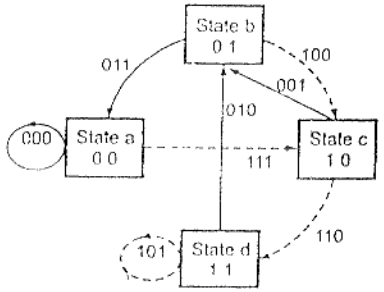


Fig. 3.4.22 The state diagram of rate $\frac{1}{3}$, k = 3 convolutional code

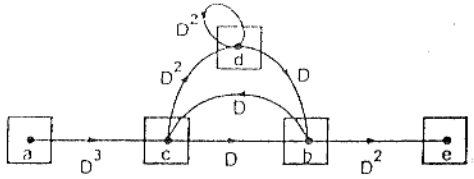


Fig. 3.4.23 State diagram of Fig. 3.4.22 with distance labels on the branches

Let us label the branches of this state diagram as D^0, D^1, D^2 or D^3. Here the exponent of D represents the number of 1s in the output corresponding to that branch. Thus the exponent of D is equivalent to hamming distance of the output with respect to all zero output. Such reorganized state diagrams with labels D, D^2 and D^3 is shown in Fig. 3.4.23.

The self loop at node 'a' has all zero outputs, i.e. $D^0 = 1$. Hence it is not shown in Fig. 3.4.23. This self loop with all zero output does not contribute to the distance properties of the code sequence relative to all zero code sequence. The node 'a' is split into two nodes. One node is called 'a' only and it is like input node and the other node is 'e' and it is like output node of the state diagram (See Fig. 3.4.23).

Along the branches between the nodes, D with proper exponents are written. For example there is one branch from 'a' to 'c' with output 111. Hence D^3 is written along this branch. Similarly the branch from 'b' to 'a' has output 011 in Fig. 3.4.22. This is shown as branch from 'b' to 'e' (output node 'e', which is obtained due to splitting 'a') with label D^2 as shown in Fig. 3.4.23.

Similarly all other branches are labeled in Fig. 3.4.23. There are five nodes in Fig. 3.4.23. The four state equations can be written for state diagram of Fig. 3.4.23 as follows :

$$\left.\begin{aligned} X_c &= D^3 X_a + D X_b \\ X_b &= D X_c + D X_d \\ X_d &= D^2 X_c + D^2 X_d \\ X_e &= D^2 X_b \end{aligned}\right\} \quad \ldots (3.4.36)$$

The state equation for the node is written from incident branches upon that node. For example node 'c' has incident branches from nodes 'a' and 'b'. The transfer function of the code is defined as,

$$T(D) = \frac{X_e}{X_a} \quad \ldots (3.4.37)$$

on solving the state equations in equation (3.4.36) we can write above transfer function as,

$$T(D) = \frac{D^6}{1 - 2D^2}$$

$$= D^6 + 2D^8 + 4D^{10} + 8D^{12} \quad \ldots (3.4.38)$$

The first term of the transfer function is D^6. It means there is single path of hamming distance d = 6 between node 'a' and 'e'. As shown in Fig. 3.4.23 this path is acbe. The second term in above equation is $2D^8$. It means there are two paths of distance d = 8 between nodes a to e in Fig. 3.4.23. These two paths are acdbe and acbcbe. Actually the path from node a to e means the path starting from node 'a' and comming back to node 'a' only. There is all zero output starting and ending at node 'a'. The distances of the various paths obtained above are the hamming distances from the all zero output path on node 'a'.

Thus the distance properties of the convolutional code can be obtained from the transfer function. The minimum distance of the code is called as minimum free distance. It is denoted by d_{free}. In this example $d_{free} = 6$.

3.4.8 Distance Properties of Binary Convolutional Codes

Consider the convolutional code of constraint length K and rate $\frac{1}{n}$. Then the minimum free distance for this code is given by the standard result as,

$$d_{free} \leq \min_{l \geq 1} \left\lfloor \frac{2^{l-1}}{2^l - 1}(K + l - 1)n \right\rfloor \quad \ldots (3.4.39)$$

Here the symbol $\lfloor x \rfloor$ means largest integer contained in x. Table 3.4.6 lists the minimum free distance and its upper bound for rate $\frac{1}{2}$ code at various constraint lengths.

The generators are also tabulated for these constraint lengths. This code is optimal in the sense that it has largest possible d_{free} for the given rate and the constraint length.

Constraint length K	Generators in octal		d_{free}	Upper bound on d_{free}
3	5	7	5	5
4	15	17	6	6
5	23	35	7	8
6	53	75	8	8
7	133	171	10	10
8	247	371	10	11
9	561	753	12	12
10	1,167	1,545,	12	13
11	2,335	3,661	14	14
12	4,335	5,723	15	15
13	10,533	17,661	16	16
14	21,675	27,123	16	17

Table 3.4.6 Rate $\frac{1}{2}$, maximum free distance code

3.4.9 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.

Example 3.4.6 : *A rate 1/3 convolution encoder has generating vectors as* $g_1 = (100)$, $g_2 = (111)$ *and* $g_3 = (101)$.

i) Sketch the encoder configuration.

ii) Draw the codetree, state diagram and trellis diagram.

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

Solution : 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. 3.4.24 shows the diagram of encoder based on above discussion.

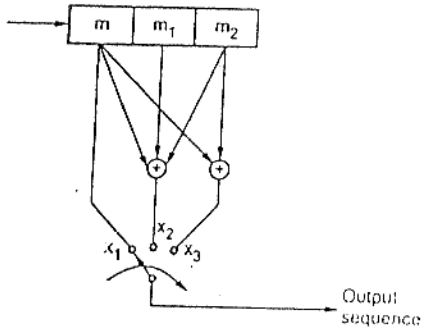


Fig. 3.4.24 Encoder of Ex. 3.4.6

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 3.4.7 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 3.4.7 State transition calculations.

How next stage is written ?

Consider the following diagram.

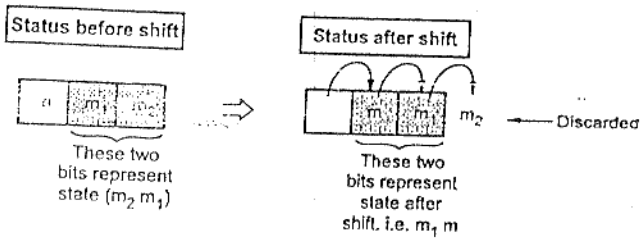
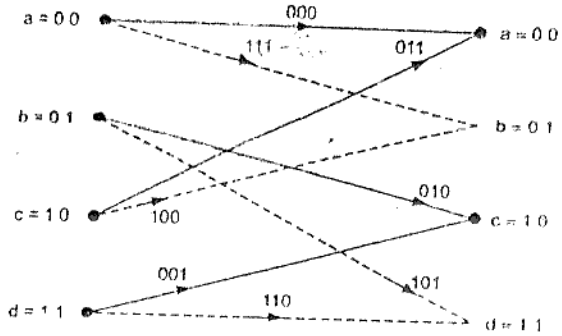


Fig. 3.4.25 Present and next states

As shown in above figure, current state is $m_2\ m_1$. When the bits are shifted, then next state becomes $m_1\ m$. Table 3.4.7 shows current and next states according to this concept.

A trellis diagram is shown in Fig. 3.4.26 based on table 3.4.7.



Dotted line indicates input m = 1

Solid line indicates input m = 0

Fig. 3.4.26 Trellis diagram for encoder of Fig. 3.4.24

b) To obtain state diagram

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

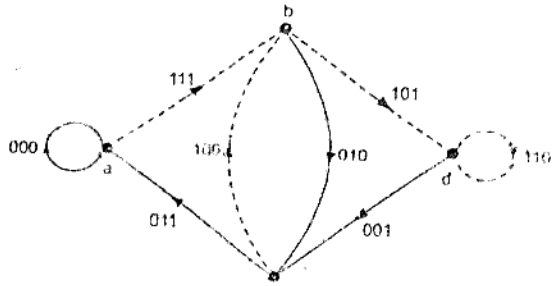


Fig. 3.4.27 State diagram of encoder of Fig. 3.4.24

c) To obtain code tree

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

1. Begin with node 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. 3.4.28.

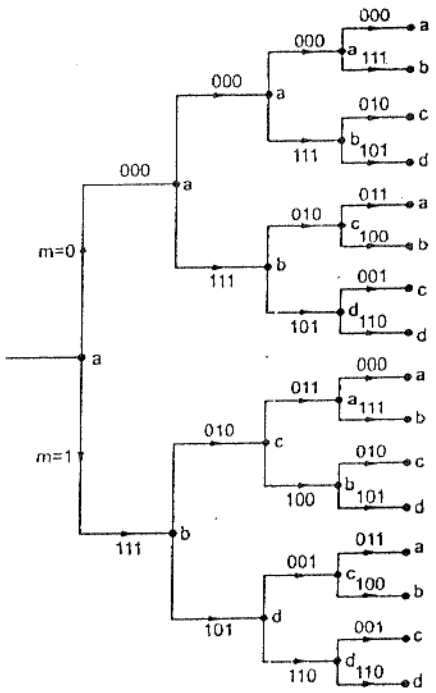


Fig. 3.4.28 Code tree of encoder of Fig. 3.4.24

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,

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

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

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

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

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

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

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

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

(b) To determine the message polynomial

The message sequence is given as,

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

$$\therefore \quad m(p) = 1 + 0p + p^2 + p^3 + 0p^4 \quad \Rightarrow \quad \boxed{m(p) = 1 + p^2 + p^3}$$

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,

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

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

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

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

Hence the corresponding sequence is,

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

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

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

$$x_i^{(3)} = g_i^{(3)}(p)\ m(p)$$
$$= (1+p^2)(1+p^2+p^3)$$
$$= 1+p^2+p^3+p^2+p^4+p^5$$
$$= 1+0p^2+p^3+p^4+p^5$$

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 3.4.7 : *A rate 1/3 convolutional coder with constraint length of '3' uses the generating vectors*

$g_1 = (100)$, $g_2 = (101)$ *and* $g_3 = (111)$

i) Sketch encoder configuration and prepare the logic table.

ii) Draw the state diagram for the coder.

iii) Determine the d_{free} *distance of the coder.*

Solution : To determine dimensions of the code :

This is rate 1/3 code. We know that,

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

Encoder configuration and logic table :

1. Encoder configuration :

The generator sequences given in this example are similar to those in Ex. 3.4.6 (previous example). Only g_2 and g_3 are exchanged. Hence encoder will be same as Fig. 3.4.24 with g_2 and g_3 exchanged.

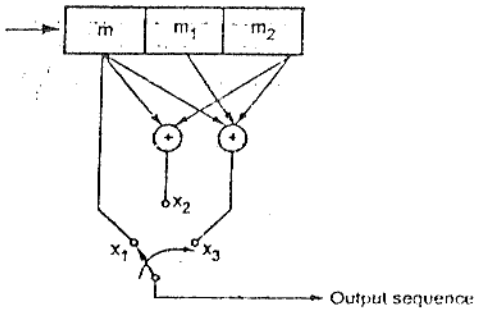


Fig. 3.4.29 Encoder of ex. 3.4.7

In above figure observe that,

$g_1 = (1\ 0\ 0)$, hence $x_1 = m$

$g_2 = (1\ 0\ 1)$, hence $x_2 = m \oplus m_2$

$g_3 = (1\ 1\ 1)$, hence $x_3 = m \oplus m_1 \oplus m_2$

2. Logic table :

Table 3.4.8 shows the state transition calculations. The states are 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'

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

Table 3.4.8 Logic table of encoder of Fig. 3.4.29

In above table note that current state is written as $m_2\ m_1$. When shift takes place, then m_2 is discarded. m_1 is shifted in place of m_2 and m is shifted in place of m_1. Hence next state is represented by $m_1\ m$. This is illustated in Fig. 3.4.25.

ii) State diagram :

With the help of table 3.4.8 state diagram can be drawn. Fig. 3.4.30 shows the state diagram.

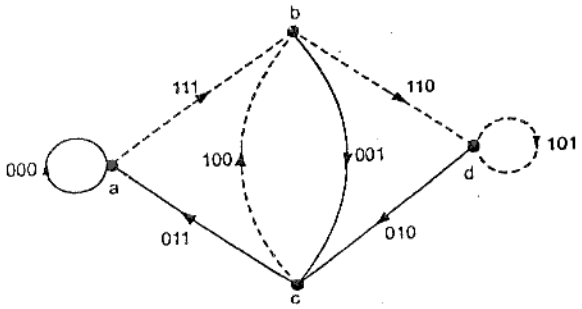


Fig. 3.4.30 State diagram of encoder of Fig. 3.4.29

iii) To determine the d_{free} of the coder

The distance d_{free} can be obtained through following steps :

a) Split the state diagram with input note as 000 output.

b) Write state equations for all nodes.

c) Determine transfer function.

d) Lowest order of transfer function is d_{free}.

a) To split the state diagram into signal flow graph

In the state diagram of Fig. 3.4.30, node 'a' generates 000 output when it returns back to itself. Hence we will split the node a into two nodes a : input node and e : output node. The signal flow graph is then prepared as shown in Fig. 3.4.31

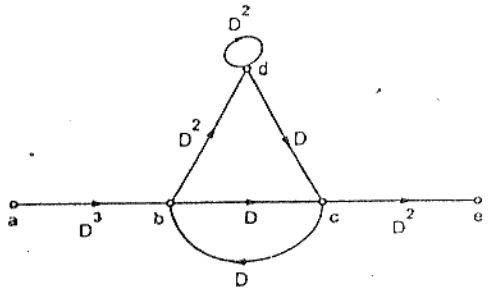


Fig. 3.4.31 Signal flow graph of Fig. 3.4.30 (Node a is split to 'a' and 'e')

The outputs are marked as D, D^2, D^3 in above figure. For example the output from 'a' to 'b' is 111. It is marked as D^3. The output from 'b' to 'c' is 001. It is marked as D.

Rule for writing output in terms of D, D^2 and D^3 :

We have considered the reference node with output 000. From 'a' to 'b', the output is 111. This output differs with 000 in 3 positions. Hence D^3 is written on branch from 'a' to 'b'. This rule is repeated for all outputs. This is explained in sec. 3.4.7 also.

b) To write the state equations for all nodes

$$X_b = D^3 X_a + D X_c \qquad \text{... (3.4.40)}$$

This is the output from node 'a' to 'b'

This is the output from node 'c' to 'b'

These equations can be written on the same lines as equation 3.4.36. Consider node 'b'. It's equation becomes,

Thus in above equation, the outputs due to branches *incident* on 'b' are considered. Similarly at node 'c' branches from 'b' and 'd' are incident. Hence its equation becomes,

$$X_c = DX_b + DX_d \qquad \text{... (3.4.41)}$$

Similarly at node 'd', one branch is incident from itself and other is incident from 'b'. i.e.,

$$X_d = D^2 X_b + D^2 X_d \quad \ldots (3.4.42)$$

At node 'e' only one branch is incident from 'c'. i.e.,

$$X_e = D^2 X_c \quad \ldots (3.4.43)$$

c) To determine transfer function

To determine transfer function, we have to solve the state equations obtained above. Eliminating X_b from equation (3.4.40) and (3.4.41) we get,

$$(1-D^2)X_c - DX_d = D^4 X_a \quad \ldots (3.4.44)$$

Eliminating X_b from equation (3.4.31) and equation (3.4.32) we get,

$$-DX_c + X_d = 0$$

Eliminating X_d from above equation and equation (3.3.34) we get,

$$(1-2D^2)X_c = D^4 X_a$$

$$X_c = \frac{D^4}{1-2D^2} X_a$$

Putting this value of X_c in equation 3.4.33,

$$X_e = D^2 \frac{D^4}{1-2D^2} X_a$$

$$\frac{X_c}{X_a} = \frac{D^6}{1-2D^2}$$

Transfer function of the code is given as,

$$T(D) = \frac{X_e}{X_a}$$

$$= \frac{D^6}{1-2D^2}$$

d) To determine lowest order of 'D' in $T(D)$

Let us determine the polynomial of $T(D)$. i.e.,

$$1-2D^2 \;\overline{)\; D^6}$$

$$\begin{array}{l} D^6 + 2D^8 + 4D^{10} + 8D^{12} \\ \underline{D^6 - 2D^8} \\ \quad 2D^8 \\ \quad \underline{2D^8 - 4D^{10}} \\ \quad\quad 4D^{10} \\ \quad\quad \underline{4D^{10} - 8D^{12}} \\ \quad\quad\quad 8D^{12} \\ \quad\quad\quad \underline{8D^{12} - 16D^{14}} \\ \quad\quad\quad\quad 16D^{14} \quad \ldots\ and\ so\ on \end{array}$$

Thus,

$$T(D) = \frac{D^6}{1-2D^2} = D^6 + 2D^8 + 4D^{10} + 8D^{12} + \ldots$$

Above equation shows that first term is D^6. This means, there is single path of distance '6' between node 'a' and 'e'. This path is abce in Fig. 3.4.31. Second term in $T(D)$ is $2D^8$. This means there are two paths of distance 8 and so on.

Free distance (d_{free}) :

The free distance is given by lowest order of the term in $T(D)$. Here it is 6. Hence,

$$d_{free} = 6$$

d_{free} can be obtained by inspection of signal flow graph

Here we derived $T(D)$, then determine d_{free}. We know that d_{free} is the minimum distance path between nodes 'a' and 'e'. By looking at Fig. 3.4.31 we can say that the minimum distance path is a-b-c-e. Along this path the distances are D^3, D and D^2. They correspond to distances of 3, 1 ans 2 respectively. Hence the distance between 'a' and 'e' will be 3 + 1 + 2 = 6. Other paths will have definitely more distances.

Hence,

$$d_{free} = 6$$

Example 3.4.8 : *Determine the state diagram for the convolutional encoder shown in Fig. 3.4.32. Draw the trellis diagram through the first set of steady state transitions. On the second trellis diagram, show the termination of trellis to all zero state.*

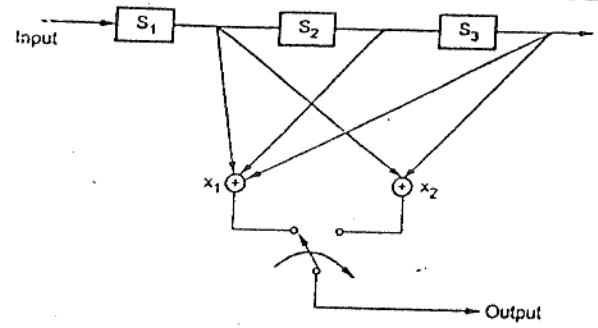


Fig. 3.4.32 Convolutional encoder of Ex. 3.4.8

Solution : **(i) To determine dimension of the code :**

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

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

ii) To obtain the state diagram :

First, let us define the states of the encoder.

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

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

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

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

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

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

Table 3.4.9 : State transition table.

Based on above table, the state diagram can be prepared easily. It is shown below in Fig. 3.4.33.

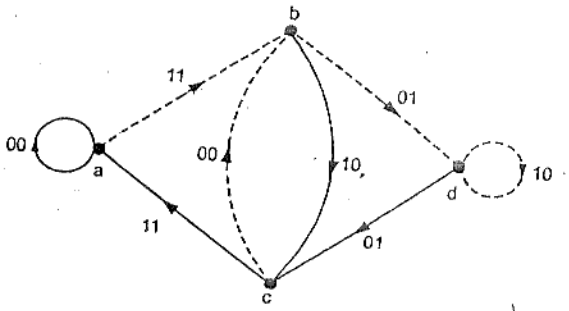


Fig. 3.4.33 State diagram of convolutional encoder of Fig. 3.4.32

iii) To obtain trellis diagram for steady state :

From table 3.4.9, the code trellis diagram can be prepared. It is steady state diagram. It is shown below.

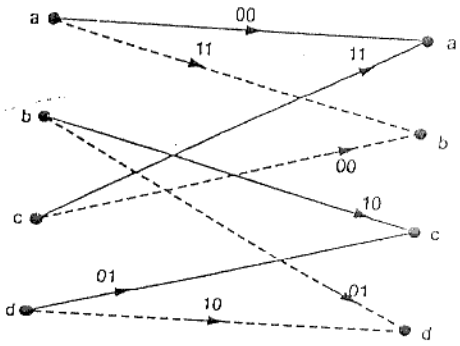


Fig. 3.4.34 Code trellis diagram for steady state

iv) Termination of trellis to all zero state :

Fig. 3.4.35 shows the trellis diagram for first four received set of symbols. It begins with state $a = 00$.

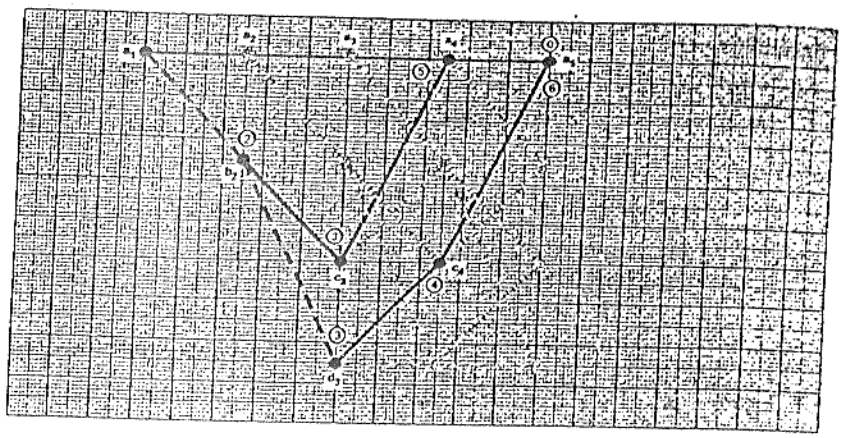


Fig. 3.4.35 Code trellis showing termination to all zero state

There is a path $a_1 - a_2 - a_3 - a_4 - a_5$ which passes through all zero states. Second path is $a_1 - b_1 - c_3 - a_4$. This path has a metric of 5 and it terminates on state a_4. Third path is, $a_1 - b_1 - d_3 - c_4 - a_5$ with metric 6. it terminates on state a_5. These are the shortest paths that terminate to all zero state. It shows that after reception of 4 message bits, the trellis can terminate to all zero state.

Example 3.4.9 : *An encoder shown in Fig. 3.4.36 generates an all zero sequence which is sent over a binary symmetric channel. The received sequence 0100100.... There are two errors in this sequence (at 2nd and 5th position). Show that this double error detection is possible with correction by application of viterbi algorithm.*

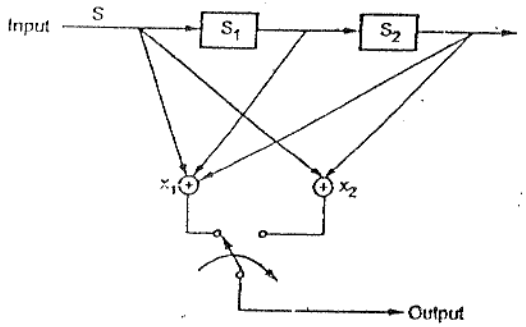


Fig. 3.4.36 Convolutional encoder of Ex. 3.4.9

Solution : i) To prepare code trellis :

First we will prepare the code trellis diagram. Let the states of the encoder be defined as follows :

$s_2\ s_1 = 00$, state 'a'
$s_2\ s_1 = 01$, state 'b'
$s_2\ s_1 = 10$, state 'c'
$s_2\ s_1 = 11$, state 'd'

A table is prepared that shows the state transitions, message input and output. This table is as follows :

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

Table 3.4.10 : State transition table for encoder of Fig. 3.4.36

Observe that above table is similar to table 3.4.9 in previous example. This is because the convolutional encoder of Fig. 3.4.36 and Fig. 3.4.32 are also similar. Note that the flip-flop for input is not shown in Fig. 3.4.36. But it doesnot affect the operation of the encoder. Hence code trellis diagram is similar to that of Fig. 3.4.34. It is shown below :

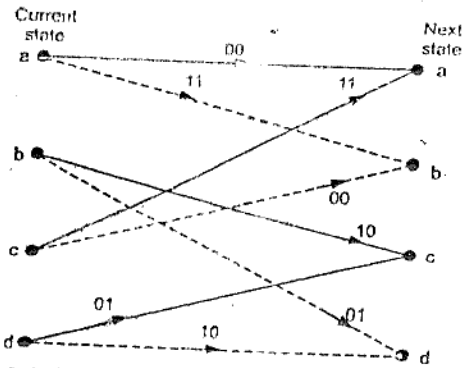


Fig. 3.4.37 Code trellis for encoder of Fig. 3.4.36

ii) To prove double error detection :

Based upon code trellis of above figure, the trellis diagram is shown for multiple stages. The diagram begins from node a_1. The outputs and metric are marked along each branch. The cumulative matric are also marked near every node. In above figure observe that the path,

$a_1 - a_2 - a_3 - a_4 - a_5 \Rightarrow$ metric (2) is maximum likelyhood path. This path has lowest metric, compared to all other paths. Hence it is maximum likelyhood path. Output due to this path is 00 00 00 00... This shows that two errors at 2^{nd} and 5^{th} bit positions are corrected by viterbi algorithm.

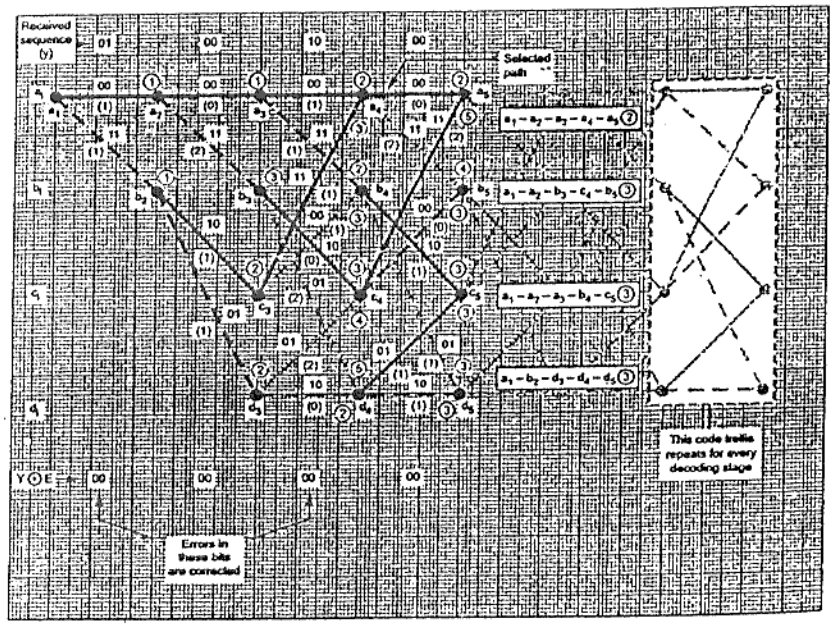


Fig. 3.4.38 Viterbi algorithm for detection of all zero sequence

Example 3.4.10 : *For the convolutional encoder with constraint length of 3 and rate 1/2 as shown in Fig. 3.4.39, draw the state diagram and trellis diagram. Is the generated code systematic ? By using viterbi algorithm, decode sequence 0100010000 ...*

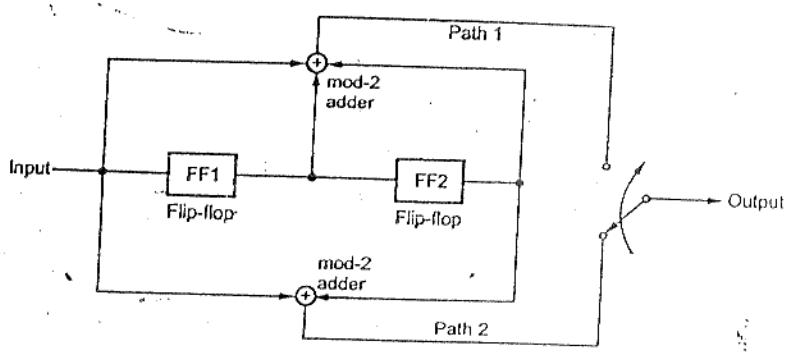


Fig. 3.4.39 Convolutional encoder of Ex. 3.4.10

Solution : **(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 below. The state of the encoder is represented by $m_2\, m_1$. Input is 'm'. Carefully observe that, above figure is similar to convolutional encoder of Fig. 3.4.4. In above figure 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.3.4.40, there are three stages in shift register which contain m, m_1 and m_2. Functionally both the encoders are same.

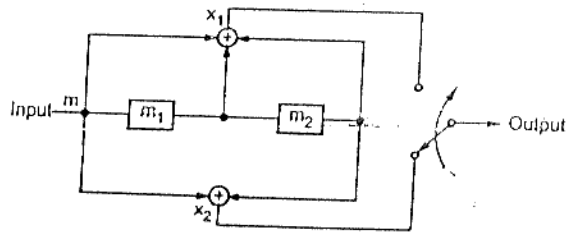


Fig. 3.4.40 Convolutional encoder of Fig. 3.4.39

Hence code trellis and state diagram of this encoder will be similar to those given in Fig. 3.4.7 and Fig. 3.4.8. They are given below :

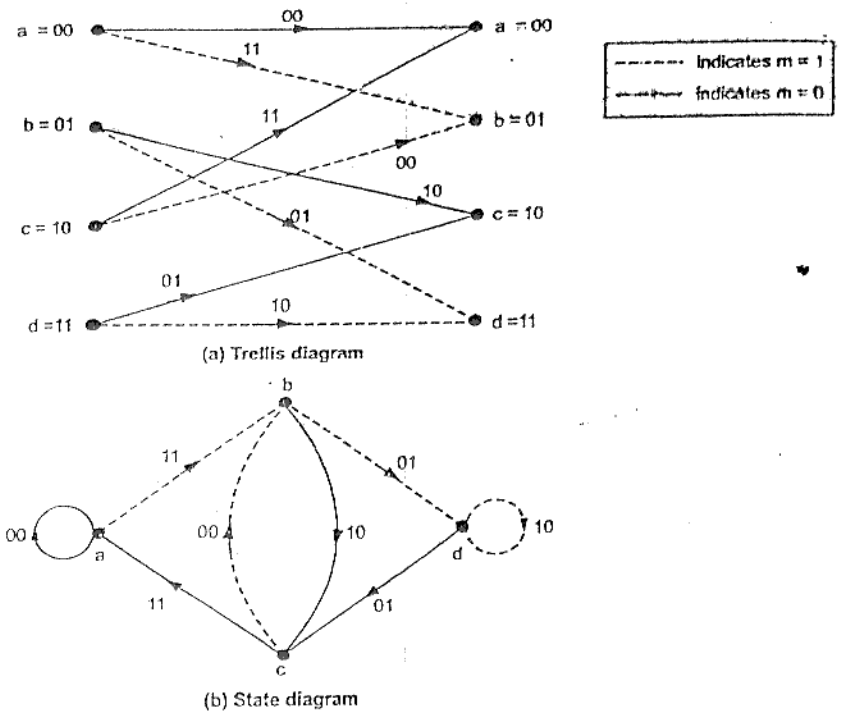


Fig. 3.4.41

ii) Whether the generated code 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.

iii) To decode 010001000 .....

A trellis diagram is shown in Fig. 3.4.42. 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. They are :

Please refer Fig. 3.4.42 on next page.

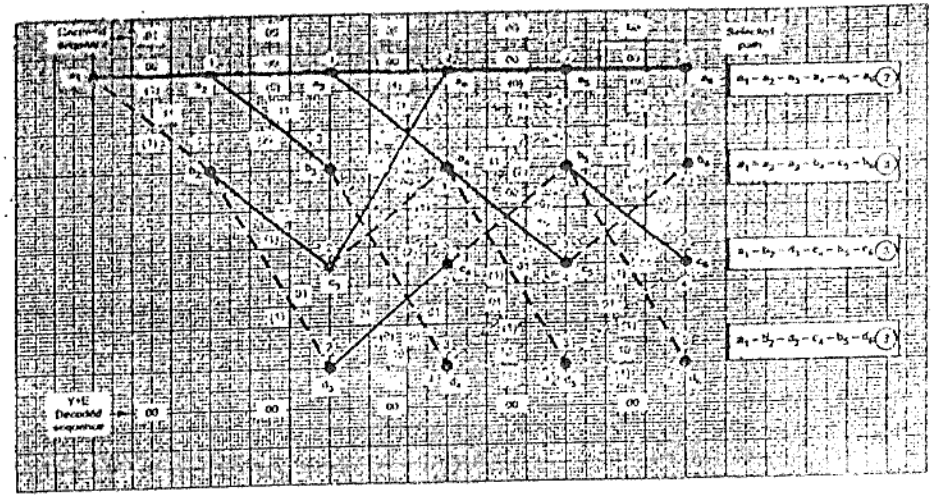


Fig. 3.4.42 Viterbi decoding

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. 3.4.42.

For path '1' the output is,

Output : 00 00 00 00 00 00

Example 3.4.11 : *For a convolutional encoder shown in Fig. 3.4.43, sketch the state diagram and trellis diagram. Determine the output data sequence for the input data sequence of 10110.*

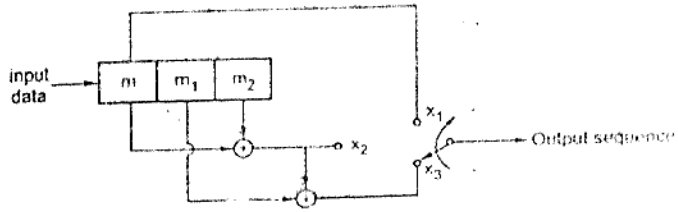


Fig. 3.4.43 Convolutional encoder of Ex. 3.4.11

Solution : i) To obtain dimensions of the code

In above figure observe that three bits are generated for every message bit. Hence $k=1$ and $n=3$. This is rate 1/3 encoder. There are three stages in the shift register. Hence constraint length is $K=3$.

ii) To prepare state diagram and trellis diagram

a) To prepare state transition table

Let the states of the encoder be defined as follows :

$m_2\ m_1 = 00$, i.e. state 'a'

$m_2\ m_1 = 01$, i.e. state 'b'

$m_2\ m_1 = 10$, i.e. state 'c'

$m_2\ m_1 = 11$, i.e. state 'd'

The outputs of the encoder can be represented as,

$$x_1 = m$$

$$x_2 = m \oplus m_2$$

$$x_3 = x_2 \oplus m_1$$

$$= [m \oplus m_2] \oplus m_1 \quad \text{Putting for } x_2$$

$$= m \oplus m_1 \oplus m_2$$

Table 3.4.11 shows the transitions between various states along with inputs and outputs.

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

Table 3.4.11 State transition table

b) To obtain trellis diagram

Fig. 3.4.44 shows the trellis diagram based on above table. It indicates the transitions between current state and next states.

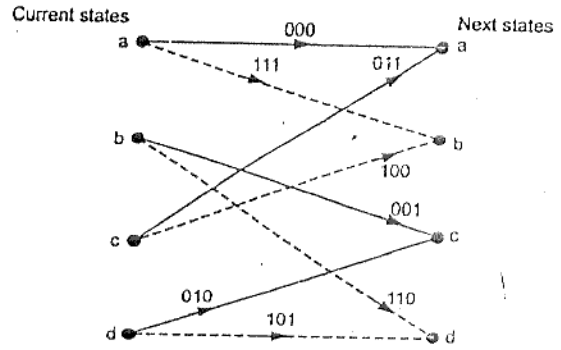


Fig. 3.4.44 Trellis diagram for the convolutional encoder of Fig. 3.4.42

c) To obtain state diagram

State diagram can be obtained by combining the current and next states in above figure. The state diagram is shown below :

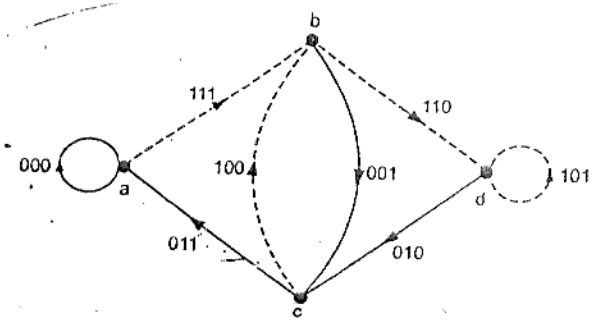


Fig. 3.4.45 State diagram of encoder of Fig. 3.4.43

iii) To obtain output for m = 10110

a) To obtain generator sequences and their polynomials

The three outputs, their corresponding generator sequences and polynomials are given in table 3.4.12.

Solution : i) To obtain dimensions of the code

In above figure observe that three bits are generated for every message bit. Hence $k = 1$ and $n = 3$. This is rate 1/3 encoder. There are three stages in the shift register. Hence constraint length is $K = 3$.

ii) To prepare state diagram and trellis diagram

a) To prepare state transition table

Let the states of the encoder be defined as follows :

$m_2\ m_1$ = 00, i.e. state 'a'

$m_2\ m_1$ = 01, i.e. state 'b'

$m_2\ m_1$ = 10, i.e. state 'c'

$m_2\ m_1$ = 11, i.e. state 'd'

The outputs of the encoder can be represented as,

$$x_1 = m$$

$$x_2 = m \oplus m_2$$

$$x_3 = x_2 \oplus m_1$$

$$= [m \oplus m_2] \oplus m_1 \quad \text{Putting for } x_2$$

$$= m \oplus m_1 \oplus m_2$$

Table 3.4.11 shows the transitions between various states along with inputs and outputs.

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

Table 3.4.11 State transition table

b) To obtain trellis diagram

Fig. 3.4.44 shows the trellis diagram based on above table. It indicates the transitions between current state and next states.

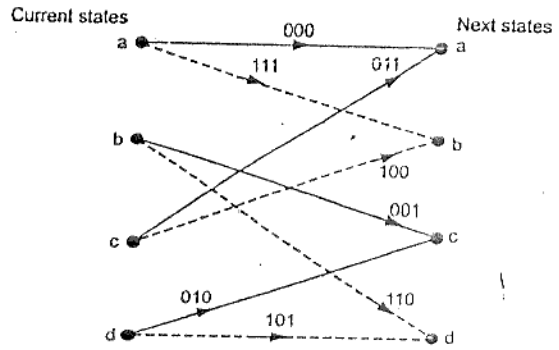


Fig. 3.4.44 Trellis diagram for the convolutional encoder of Fig. 3.4.42

c) To obtain state diagram

State diagram can be obtained by combining the current and next states in above figure. The state diagram is shown below :

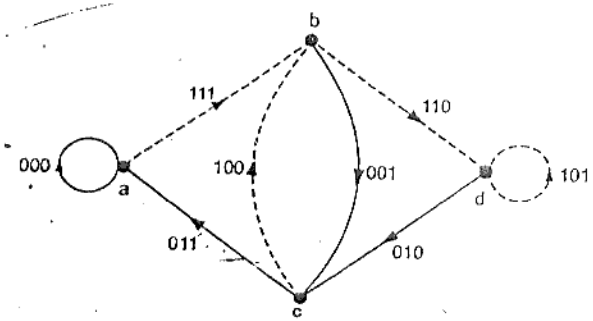


Fig. 3.4.45 State diagram of encoder of Fig. 3.4.43

iii) To obtain output for m = 10110

a) To obtain generator sequences and their polynomials

The three outputs, their corresponding generator sequences and polynomials are given in table 3.4.12.

Sr. No.	Output sequence	Output equation	Generator sequence	Generator polynomial
1	x_1	$x_1 = m$	1 0 0	$g_1(p) = 1$
2	x_2	$x_2 = m + m_2$	1 0 1	$g_2(p) = 1 + p^2$
3	x_3	$x_3 = m + m_1 + m_2$	1 1 1	$g_3(p) = 1 + p + p^2$

Table 3.4.12 Generating polynomials.

b) Determine message polynomial

The message sequence is given as,

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

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

c) To obtain output sequences x_1, x_2 and x_3

The sequence of x_1 can be obtained by multiplying $g_1(p)$ and $m(p)$. Similarly other sequences can be obtained. These calculations are listed in table 3.4.13.

Sr. No.	Output sequence	Output polynomial $x_i = g(p)\,m(p) = g(p)(1 + p^2 + p^3)$	Corresponding sequence
1	$x_i^{(1)}$	$x_i^{(1)} = g_1(p)m(p)$ $= 1(1 + p^2 + p^3)$ $= 1 + p^2 + p^3$ $= 1 + 0p + p^2 + p^3$	$x_i^{(1)} = \{1\ 0\ 1\ 1\}$
2	$x_i^{(2)}$	$x_i^{(2)} = g_2(p)m(p)$ $= (1 + p^2)(1 + p^2 + p^3)$ $= 1 + p^2 + p^3 + p^2 + p^4 + p^5$ $= 1 + 0p + 0p^2 + p^3 + p^4 + p^5$	$x_i^{(2)} = \{1\ 0\ 0\ 1\ 1\ 1\}$
3	$x_i^{(3)}$	$x_i^{(3)} = g_3(p)m(p)$ $= (1 + p + p^2)(1 + p^2 + p^3)$ $= 1 + p^2 + p^3 + p + p^3 + p^4 + p^2 + p^4 + p^5$ $= 1 + p + 0p^2 + 0p^3 + 0p^4 + p^5$	$x_i^{(3)} = \{1\ 1\ 0\ 0\ 0\ 1\}$

Table 3.4.13 : Output polynomials and sequences for $m(p) = 1 + p^2 + p^3$

To multiplex $x_i^{(1)}$, $x_i^{(2)}$ and $x_i^{(3)}$:

Final output sequence can be obtained by multiplexing the three sequences length of $x_i^{(2)}$ and $x_i^{(3)}$ is 6. And length of $x_i^{(1)}$ is 4. Hence two zeros must be appended to $x_i^{(1)}$ to make lengths of all the sequences equal. The sequences are given below :

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

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

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

Multiplexing above sequences, we get,

$$x_i = \{111\ \ 001\ \ 100\ \ 110\ \ 010\ \ 011\}$$

This is the output data sequence.

3.4.10 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 3.4.14 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. $x_i = \sum_{i=0}^{M} g_i m_{l-i}, i = 0, 1, 2, \ldots$
2	For a block of message bits, encoded block (code vector) is generated.	Each message bit is encoded separately. For every message bit, two or more encoded bits are generated.
3	Coding is block by block.	Coding is bit by bit.
4	Syndrome decoding is used for most likelyhood decoding.	Viterbi decoding is used for most likelyhood 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 codevectors.	Distance properties of the code can be studied from transfer function.
7	Generating polynomial and generator matrix are used to get codevectors.	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 3.4.14 : Comparison of linear block codes and convolutional codes

Review Questions

1. *What are convolutional codes ? How are they different from block codes ?*
2. *Giving block diagram, explain the operation of any convolutional encoder.*
3. *What is constraint length for convolutional encoders ?*
4. *What are code tree, code trellis and state diagrams for convolutional encoders?*

5. *Explain the viterbi algorithm and sequential decoding of convolutional codes.*
6. *Compare linear block codes, cyclic codes and convolutional codes by giving their advantages and disadvantages.*

Unsolved Examples

1. *A rate ½, K = 3, binary convolutional encoder is shown in Fig. 3.4.46.*
   a) Draw the tree diagram, trellis diagram and the state diagram for above encoder.

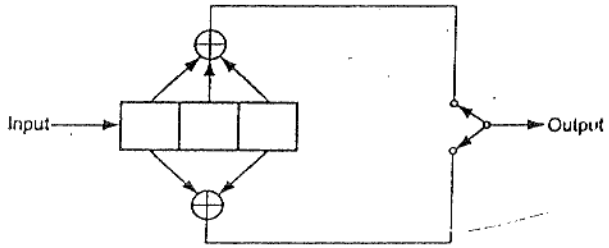


Fig. 3.4.46

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.

2. *For the convolutional encoder shown in Fig. 3.4.47, sketch the code tree.*

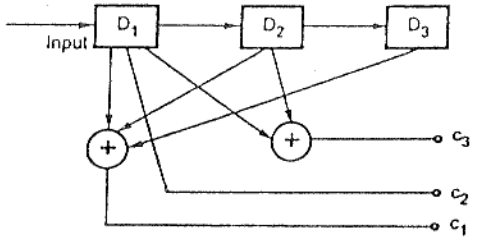


Fig. 3.4.47

3.5 Short Answered Questions

Q.1 What is hamming distance ? **[Nov.Dec.-2004, 2 Mark, Nov./Dec.-2003, 2 Marks]**

Ans. : The hamming distance between the two codevectors is equal to the number of elements in which they differ. For example, let the two codewards be,

$$X = (101) \text{ and } Y = (110)$$

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

Q.2 Define code efficiency ?

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

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

Q.3 What is meant by systematic and nonsystematic codes ?

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

Q.4 What is meant by linear code ?

Ans. : A code is linear if modulo-2 sum of any two codevectors produces another codevector. This means any code vector can be expressed as linear combination of other codevectors.

Q.5 What are the error detection and correction capabilities of Hamming codes ?

Ans. : The minimum distance (d_{min}) of Hamming codes is '3'. Hence it can be used to detect double errors or correct single errors. Hamming codes are basically linear block codes with $d_{min} = 3$.

Q.6 What is meant by cyclic code ?

Ans. : Cyclic codes are the subclass of linear block codes. They have the properly that a cyclic shift of one codeword produces another code word. For example consider the codeword.

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

Let us shift above codevector to left cyclically,

$$X^i = (x_{n-2}, x_{n-3}, \ldots x_0, x_1, x_{n-1})$$

Above codevector is also a valid codevector.

Q.7 How syndrome is calculated in Hamming codes and cyclic codes ?

Ans. : In Hamming codes the syndrome is calculated as,

$$S = YH^T$$

Here Y is the received and H^T is the transpose of parity check matrix.

In cyclic code, the syndrome vector polynomial is given as,

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

Here Y(p) is received vector polynomial and G(p) is generator polynomial.

Q.8 What is BCH code ?

Ans. : BCH codes are most extensive and powerful error correcting cyclic codes. The decoding of BCH codes is comparatively simpler. For any positive integer 'm' and 't' (where $t < 2^{m-1}$) there exists a BCH code with following parameters :

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

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

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

Q.9 What is RS code ?

Ans. : These are nonbinary BCH codes. The encoder for RS codes operate on multiple bits simultaneously. The (n,k) RS code takes the groups of m – bit symbols of the incomming binary data stream. It takes such 'k' number of symbols in one block. Then the encoder adds (n – k) redundant symbols to form the codeword of 'n' symbols.

RS code has :

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

Message size : k symbols

Parity check size : $n - k = 2t$ symbols

Minimum distance : $d_{min} = 2t + 1$ symbols

Q.10 What is the difference between block codes and convolutional codes ?

Ans. : Block codes take 'k' number of message bit simultaneously and form 'n'-bit code vector. This code vector is also called block. Convolutional code takes one message bit at a time and generates two or more encoded bits. Thus convolutional codes generate a string of encoded bits for input message string.

Q.11 Define constraint length in convolutional codes .

Ans. : Constraint length is the number of shifts over which the single message bit can influence the encoder output. It is expressed in terms of message bits.

Q.12 Define free distance and coding gain.

Ans. : Free distance is the minimum distance between code vectors. It is also equal to minimum weight of the code vectors.

Coding gain is used as a basis of comparison for different coding methods. To achieve the same bit error rate the coding gain is defined as,

$$A = \frac{\left(\frac{E_b}{N_o}\right)encoded}{\left(\frac{E_b}{N_o}\right)coded}$$

For convolutional coding, the coding gain is given as,

$$A = \frac{rd_f}{2}$$

Here 'r' is the code rate

and 'd_f' is the free distance.

❑❑❑