UNIT. 8

CONVOLUTIONAL CODES

8.1 CONVOLUTIONAL CODES

The main difference between block codes discussed in previous units and the convolutional codes (refer section 5.6 for definitions) is the following.

In ***"block codes"***, a block of 'n' -digits generated by the encoder in a particular time-unit depends only on the block of 'k' input message digits within that time unit.

In ***"convolutional codes"***, a block of 'n' code digits generated by the encoder in a time unit depends on not only the block of 'k' message digits within that time unit, but also on the preceding (m – 1) blocks of message digits (m > 1). Usually the values of 'k' and 'n' will be small.

Like block codes, convolutional codes can be designed to either detect or correct errors. However, block codes are better suited for error detections and convolutional codes for error correction.

Encoding of convolutional codes can be accomplished using simple shift registers and several practical procedures have been developed for decoding.

ENCODER FOR CONVOLUTIONAL CODES :

A convolutional encoder, shown in its general form in figure 8.1, takes sequences of message digits and generates sequences of code digits. In any time unit, a message block consisting of k digits is fed into the encoder and the encoder generates a code block consisting of 'n' code digits (k < n). The n-digit code block depends not only on the k-digit message block of the same time unit, but also on the previous (m – 1) message blocks. The code generated by the above encoder is called as (n, k, m) convolutional code of constraint length "*nm*" digits and "rate efficiency k/n". The parameters of the encoder, n, k, and m are in general small integers and the encoder consists of shift registers and modulo-2 adders. The following example illustrates the concept.

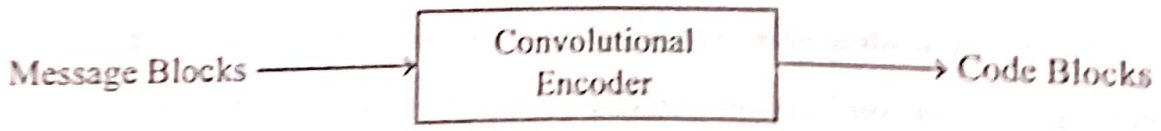


Fig. 8.1 : General convolutional encoder

Example 8.1 : An encoder for a (n, k, m) = (3, 1, 3) convolutional code is shown in figure 8.2. Explain the operation of the encoder and hence obtain the output of the encoder.

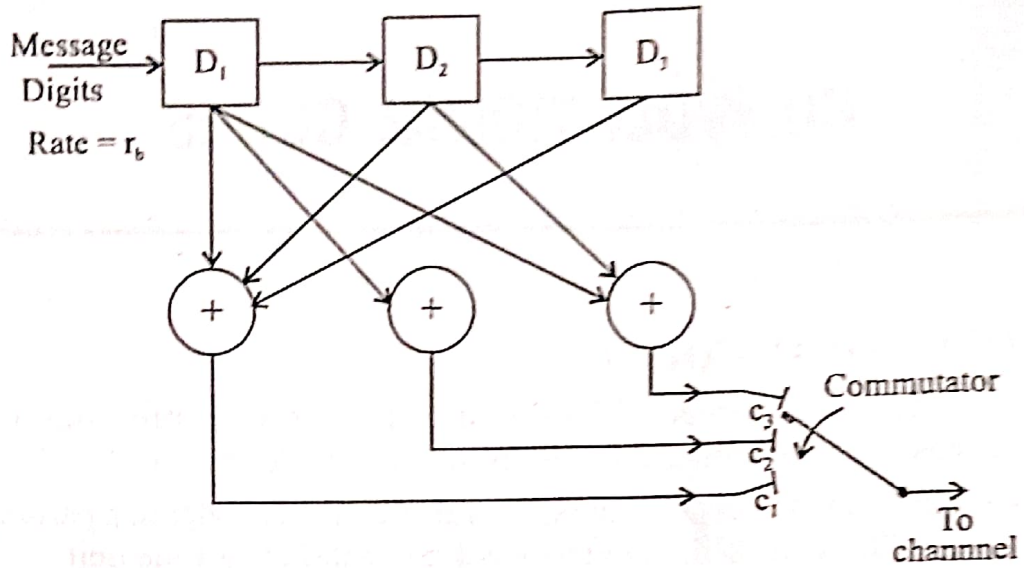


Fig. 8.2 : (3, 1, 3) convolutional encoder of example 8.1

Solution

Let us look into the operation of the given encoder for a message input of 1 0 1 1 0 (= d_1 d_2 d_3 d_4 d_5) .

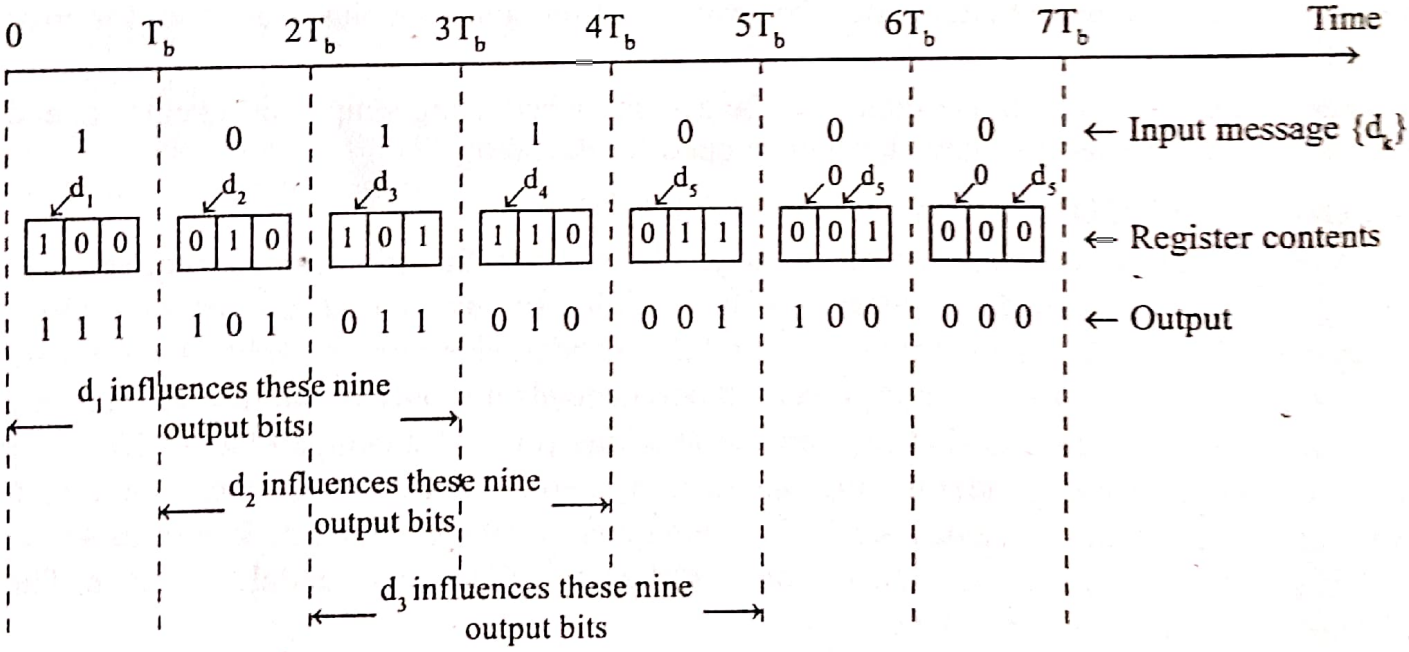


Fig. 8.3 : Encoding operation of the convolutional encoder of figure 4.1

Operation : Let the shift register be cleared initially. The first data bit 'd_1' is entered into the first flip-flop labelled D_1. The commutator samples the modulo-2 adder outputs c_1, c_2 and c_3. Thus a single message bit yields three output bits, The next message bit 'd_2' now enters D_1, while the contents of D_1 which was 'd_1' is shifted into D_2. Then, again the

commutator samples the three adder outputs. This process is repeated till the last message digit d_5 is shifted into the flip-flop D_3. In order to first move d_5 into D_2 and then into D_3, it is assumed that '0's are fed at the input. The details of the encoding operation is shown in figure 8.3 along with the output of the encoder.

In convolutional encoder, the message stream continuously runs through the encoder unlike in block coding schemes where the message stream is first divided into long blocks and then encoded. Thus the convolutional encoder requires very little buffering and storage hardware.

In a general (n, k, m) convolutional encoder, the following notations are used.

n = number of outputs = number of modulo-2 adders (normally)

k = number of input bits entering at any time

m = number of stages of shift register

= number of flip-flops

L = number of bits in the message sequence.

Constraint length = m × n digits

rate efficiency = k/n

8.2 ENCODING OF CONVOLUTIONAL CODES USING TIME-DOMAIN APPROACH

The concept of encoding of convolutional codes using time-domain approach can be clearly understood through an example.

Example 8.2 : Let us consider a (n, k, m → 2, 1, 3) convolutional encoder as shown in figure 8.4.

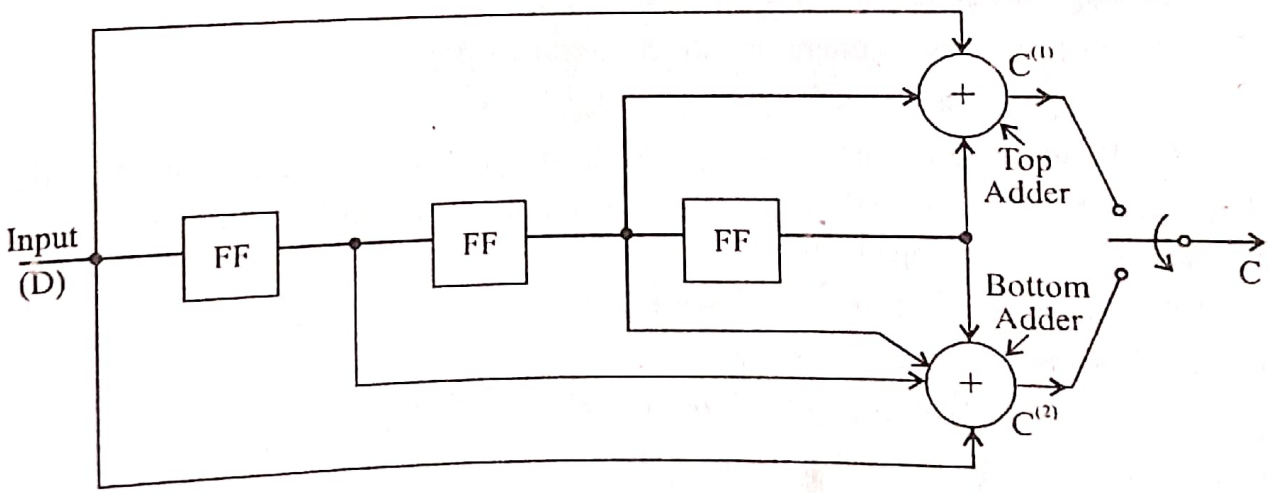


Fig. 8.4 : Convolutional encoder (n, k, m → 2, 1, 3) of example 8.2

The time-domain behaviour of a binary convolutional encoder may be defined in terms of a set of ***"n-impulse responses"***. The rate efficiency of the encoder of figure 8.4 is 1/2.

Therefore, we need to impulse responses to characterize its behaviour in time domain. [In general, we require "n" number of impulse responses].

I. Let the sequence $[g_1^{(j)}, g_2^{(j)}, g_3^{(j)}, \ldots g_{(m+1)}^{(j)}]$ denote the ***"impulse responses"*** also called ***"generator sequences"*** of the input-output path through "n" number of modulo-2 adders.

In the encoder of figure 8.4, there are two modulo-2 adders labelled top-adder and bottom adder. Hence there will be two generator sequences. Let $(d_1, d_2, \ldots d_L)$ represent the input message sequence that enters into the encoder, one bit at a time starting with d_1. Then the encoder generates two output sequences, denoted by $C^{(1)}$ and $C^{(2)}$, defined by the discrete convolutional sums, given by

$$C^{(1)} = [d] * g^{(1)} \qquad \ldots\ldots (8.1)$$

$$C^{(2)} = [d] * g^{(2)} \qquad \ldots\ldots (8.2)$$

From definition of discrete convolution, we have

$$C_l^{(j)} = \sum_{i=0}^{m} d_{l-i}\, g_{i+1}^{(j)} \qquad \ldots\ldots (8.3)$$

In the given encoder, j takes values 1 and 2 and i varies from 0 to m = 3.

Let the message sequence be $d_1\ d_2\ d_3\ d_4\ d_5 = 1\ 0\ 1\ 1\ 1$.

The output sequences are calculated as follows :

For j = 1 : From equation (8.3),

$$C_l^{(1)} = \sum_{i=0}^{3} d_{l-i}\, g_{i+1}^{(1)} \quad \text{where } d_{l-i} = 0 \text{ for all } l \le i$$

$$C_l^{(1)} = d_l\, g_1^{(1)} + d_{l-1}\, g_2^{(1)} + d_{l-2}\, g_3^{(1)} + d_{l-3}\, g_4^{(1)} \qquad \ldots\ldots (8.4)$$

The message sequence $d = d_1\ d_2\ d_3\ d_4\ d_5 = 1\ 0\ 1\ 1\ 1$.

The impulse response (generator sequence) for the top-adder is

$$g^{(1)} = g_1^{(1)}\ g_2^{(1)}\ g_3^{(1)}\ g_4^{(1)} = 1\ 0\ 1\ 1$$

[$g_2^{(1)} = 0$ since there is no connection from output of first flip-flop to the top adder].

The quantity "l" in equation (8.3) always varies from 1 to (L + m) in general where L = number of message sequence bits.

In the given encoder and message sequence, L = 5 and m = 3. ∴ 'l' varies from 1 to 8.

For $l = 1$, equation (8.4) becomes

$$C_1^{(1)} = d_1\, g_1^{(1)} + 0 + 0 + 0 = (1)\,(1) = 1$$

For $l = 2$,

$$C_2^{(1)} = d_2\, g_1^{(1)} + d_1\, g_2^{(1)} + 0 + 0$$
$$= (0)\,(1) + (1)\,(0) = 0$$

For $l = 3$,

$$C_3^{(1)} = d_3\, g_1^{(1)} + d_2\, g_2^{(1)} + d\, g_3^{(1)}$$
$$= (1)\,(1) + (0)\,(0) + (1)\,(1) = 1 + 1 = 0$$

For $l = 4$, $C_4^{(1)} = d_4\, g_1^{(1)} + d_3\, g_2^{(1)} + d_2\, g_3^{(1)} + d_1\, g_4^{(1)}$

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

For $l = 5$, $C_5^{(1)} = d_5\, g_1^{(1)} + d_4\, g_2^{(1)} + d_3\, g_3^{(1)} + d_2\, g_4^{(1)}$

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

for $l = 6$, $C_6^{(1)} = d_6\, g_1^{(1)} + d_5\, g_2^{(1)} + d_4\, g_3^{(1)} + d_3\, g_4^{(1)}$

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

For $l = 7$, $C_7^{(1)} = d_7\, g_1^{(1)} + d_6\, g_2^{(1)} + d_5\, g_3^{(1)} + d_4\, g_4^{(1)}$

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

For $l = 8$, $C_8^{(1)} = d_8\, g_1^{(1)} + d_7\, g_2^{(1)} + d_6\, g_3^{(1)} + d_5\, g_4^{(1)}$

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

$\therefore$ For top adder, the output sequence,

$$C^{(1)} = C_1^{(1)}\, C_2^{(1)}\, C_3^{(1)}\, C_4^{(1)}\, C_5^{(1)}\, C_6^{(1)}\, C_7^{(1)}\, C_8^{(1)} = 1\,0\,0\,0\,0\,0\,0\,1$$

For j = 2 : From equation (8.3),

$$C_l^{(2)} = \sum_{i=0}^{3} d_{l-i}\, g_{i+1}^{(2)}$$

$$C_l^{(2)} = d_l\, g_1^{(2)} + d_{l-1}\, g_2^{(2)} + d_{l-2}\, g_3^{(2)} + d_{l-3}\, g_4^{(2)} \qquad \text{..... (8.5)}$$

The message sequence d = 1 0 1 1 1 and the generator sequence for the bottom adder is

$$g^{(2)} = g_1^{(2)}\, g_2^{(2)}\, g_3^{(2)}\, g_4^{(2)} = 1\,1\,1\,1$$

The quantity 'l' again varies from 1 to 8.

For $l = 1$, equation (8.5) becomes

$C_1^{(2)} = d_1\, g_1^{(2)} + 0 + 0 + 0 = (1)(1) = 1$

For $l = 2$, $C_2^{(2)} = d_2\, g_1^{(2)} + d_1\, g_2^{(2)} + 0 + 0$

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

For $l = 3$, $C_3^{(2)} = d_3\, g_1^{(2)} + d_2\, g_2^{(2)} + d_1\, g_3^{(2)} + 0$

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

For $l = 4$, $C_4^{(2)} = d_4\, g_1^{(2)} + d_3\, g_2^{(2)} + d_2\, g_3^{(2)} + d_1\, g_4^{(2)}$

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

For $l = 5$, $C_5^{(2)} = d_5\, g_1^{(2)} + d_4\, g_2^{(2)} + d_3\, g_3^{(2)} + d_2\, g_4^{(2)}$

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

for $l = 6$, $C_6^{(2)} = d_6\, g_1^{(2)} + d_5\, g_2^{(2)} + d_4\, g_3^{(2)} + d_3\, g_4^{(2)}$

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

For $l = 7$, $\quad C_7^{(2)} = d_7 g_1^{(2)} + d_6 g_2^{(2)} + d_5 g_3^{(2)} + d_4 g_4^{(2)}$

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

For $l = 8$, $\quad C_8^{(2)} = d_8 g_1^{(2)} + d_7 g_2^{(2)} + d_6 g_3^{(2)} + d_5 g_4^{(2)}$

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

∴ For bottom adder, the output sequence,

$$C^{(2)} = C_1^{(2)}\, C_2^{(2)}\, C_3^{(2)}\, C_4^{(2)}\, C_5^{(2)}\, C_6^{(2)}\, C_7^{(2)}\, C_8^{(2)} = 1\,1\,0\,1\,1\,1\,0\,1$$

After encoding, the two output sequences are multiplexed into a single sequence called ***"code-word"*** for transmission over the channel. The ***"code-word"*** at the output of the convolutional encoder is given by

$$C = [C_1^{(1)}\, C_1^{(2)},\ C_2^{(1)}\, C_2^{(2)},\ C_3^{(1)}\, C_3^{(2)},\ \ldots\ldots\ C_{L+m}^{(1)}\, C_{L+m}^{(2)}]$$

in general for two modulo-2 adders.

In the given encoder.

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

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

∴ The encoder output is

$$C = [11, 01, 00, 01, 01, 01, 00, 11]$$

II METHOD (MATRIX METHOD) :

The generator sequences $g_1^{(1)}\, g_2^{(1)}\, g_3^{(1)} \ldots\ g_{m+1}^{(1)}$ for the top adder and $g_1^{(2)}\, g_2^{(2)}\, g_3^{(2)} \ldots\ldots\ g_{m+1}^{(2)}$ for the bottom adder, can be interlaced and arranged in a matrix form with the number of rows equal to the number of digits in the message sequence = L rows and number of columns = n (L + m). Such a matrix of order [L] × [n (L + m)] is called ***"generator matrix"*** of the convolutional encoder.

In general, for a two modulo-2 adder convolutional encoder, the generator matrix G is given by

$$G = \begin{bmatrix} g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & g_3^{(1)} g_3^{(2)} & \cdots & g_{m+1}^{(1)} g_{m+1}^{(2)} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 \quad 0 & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \cdots & g_m^{(1)} g_m^{(2)} & g_{m+1}^{(1)} g_{m+1}^{(2)} & 0 & 0 & & \cdots & 0 & 0 \\ 0 \quad 0 & 0 \quad 0 & g_1^{(1)} g_1^{(2)} & \cdots & \cdots\cdots & g_m^{(1)} g_m^{(2)} & g_{m+1}^{(1)} g_{m+1}^{(2)} & & & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots & & & & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots & & & & g_{m+1}^{(1)} g_{m+1}^{(2)} \end{bmatrix}$$

In the 2nd row, the number of '0's is equal to the number of modulo-2 adders. Since the generator matrix G has n(L + m) number of columns, the encoder output will have n(L + m) number of bits given by

$$C = dG \qquad \ldots\ldots (8.6)$$

In the given encoder of figure 8.4

$$d = 1\,0\,1\,1\,1$$

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

$\therefore$ The generator matrix has L = 5 rows and n(L + m) = 2 (5 + 3) = 16 columns given by

$$G = \begin{bmatrix} 11 & 01 & 11 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 01 & 11 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 01 & 11 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 01 & 11 & 11 \end{bmatrix}$$

The encoder output is given by equation (8.6) as

$$C = dG$$

$$= [1\,0\,1\,1\,1] \begin{bmatrix} 11 & 01 & 11 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 01 & 11 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 01 & 11 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 01 & 11 & 11 \end{bmatrix}$$

$\therefore$ C = [11, 01 , 00, 01, 01, 01, 00, 11] which is same as before.

8.3 ENCODING OF CONVOLUTIONAL CODES USING TRANSFORM-DOMAIN APPROACH

From the study of linear filter theory, we know that the convolution integral, which describes the linear filtering operation in the time domain, is replaced by the multiplication of Fourier transforms in the frequency domain. Since a convolutional encoder is a LTI finite state machine, we may simplify computation of the adder outputs by applying an appropriate transformation. Let the impulse response of each path in the encoder be replaced by a polynomial whose coefficients are represented by the respective elements of the impulse response. Thus for "j" number of modulo-2 adders [where "j" varies from 1 to n]. We define the generator polynomial

$$g^{(j)}(x) = g_1^{(j)} + g_2^{(j)} x + g_3^{(j)} x^2 + \ldots + g_{m+1}^{(j)} x^m \qquad \ldots\ldots (8.7)$$

The corresponding output of each of the adders is then given by

$$C^{(j)}(x) = d(x)\, g^{(j)}(x) \qquad \ldots\ldots (8.8)$$

After getting the polynomials at the output of each of the adders, the final encoder output polynomial is obtained from

$$C(x) = C^{(1)}(x^n) + x\, C^{(2)}(x^n) + x^2\, C^{(3)}(x^n) + \ldots + x^{n-1}\, C^{(n)}(x^n) \qquad \ldots (8.9)$$

Example 8.3 : Obtain the output of the convolutional encoder of figure 8.4 using transform domain approach.

Solution

The generator sequence for the top adder of figure 8.4 is given by

$$g^{(1)} = [g_1^{(1)}\, g_2^{(1)}\, g_3^{(1)}\, g_4^{(1)}] = [1\ 0\ 1\ 1]$$

The generator polynomial corresponding to the top adder is given by equation (8.7) as

$$g^{(1)}(x) = g_1^{(1)} + g_2^{(1)} x + g_3^{(1)} x^2 + g_4^{(1)} x^3$$
$$= 1 + 0 + x^2 + x^3 = 1 + x^2 + x^3$$

The generator sequence corresponding to the bottom adder of figure 8.4 is given by

$$g^{(2)} = [g_1^{(2)}\, g_2^{(2)}\, g_3^{(2)}\, g_4^{(2)}] = [1\ 1\ 1\ 1]$$

∴ The corresponding generator polynomial is

$$g^{(2)}(x) = g_1^{(2)} + g_2^{(2)} x + g_3^{(2)} x^2 + g_4^{(2)} x^3$$
$$= 1 + x + x^2 + x^3$$

∴ The top-adder output polynomial is given by equation (8.8) as

$$C^{(1)}(x) = d(x)\, g^{(1)}(x)$$

We have, for the message d = 1 0 1 1 1, the message polynomial given by

$$d(x) = 1 + x^2 + x^3 + x^4$$

$$\therefore \quad C^{(1)}(x) = (1 + x^2 + x^3 + x^4)(1 + x^2 + x^3)$$
$$= 1 + x^2 + x^3 + x^4$$
$$+ x^2 \quad + x^4 + x^5 + x^6$$
$$+ x^3 \quad + x^5 + x^6 + x^7$$

$$\therefore \quad C^{(1)}(x) = 1 + x^7$$

Similarly the bottom adder output polynomial is given by equation (8.8), with j = 2, as

$$C^{(2)}(x) = d(x)\, g^{(2)}(x)$$
$$= (1 + x^2 + x^3 + x^4)(1 + x + x^2 + x^3)$$
$$= 1 + x + x^2 + x^3$$
$$+ x^2 + x^3 + x^4 + x^5$$
$$+ x^3 + x^4 + x^5 + x^6$$
$$+ x^4 + x^5 + x^6 + x^7$$

$$C^{(2)}(x) = 1 + x + x^3 + x^4 + x^5 + x^7$$

The final encoder output polynomial is given by equation (8.9), with n = number of modulo-2 adders = 2.

$$\therefore \quad C(x) = C^{(1)}(x^2) + x\, C^{(2)}(x^2) \qquad \text{..... (8.10)}$$

We have $C^{(1)}(x) = 1 + x^7$

$$\therefore \quad C^{(1)}(x^2) = 1 + x^{14} \qquad \text{..... (8.11)}$$

and $C^{(2)}(x) = 1 + x + x^3 + x^4 + x^5 + x^7$

$$\therefore \quad C^{(2)}(x^2) = 1 + x^2 + x^6 + x^8 + x^{10} + x^{14} \qquad \text{..... (8.12)}$$

Using equations (8.11) and (8.12) in (8.10), we get

$$C(x) = 1 + x^{14} + x(1 + x^2 + x^6 + x^8 + x^{10} + x^{14})$$
$$= 1 + x + x^3 + x^7 + x^9 + x^{11} + x^{14} + x^{15}$$

The code-word corresponding to this polynomial is

$$C = [11, 01, 00, 01, 01, 01, 00, 11]$$

which is same as obtained using time-domain approach.

Example 8.4 : For the convolutional encoder shown in figure 8.5, the information sequence is d = 1 0 0 1 1. Find the output sequence using the following two approaches.

(a) Time-domain approach.

(b) Transform-domain approach. ***[VI Sem EC/TE, July/Aug. 2004, Q.8]***

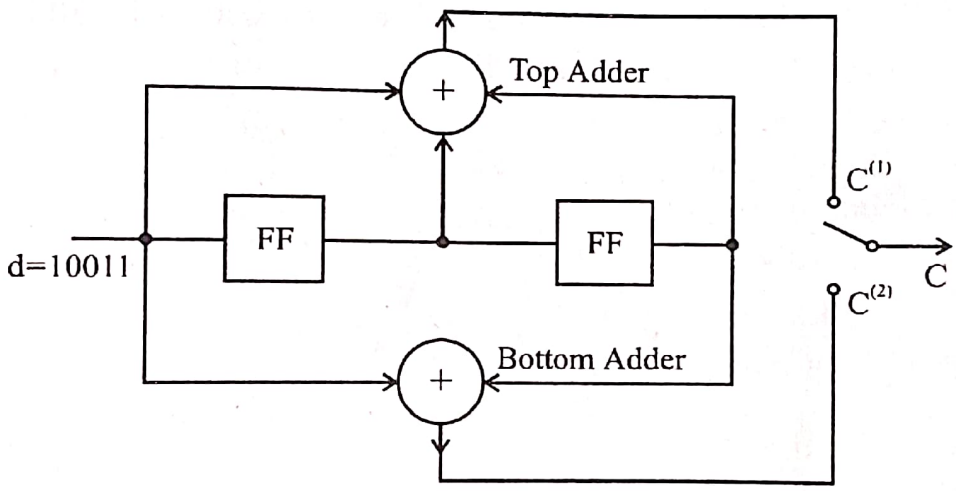


Fig. 8.5 : (2, 1, 2) convolutional encoder of example 8.4

Solution

(i) Time domain approach :

Let us use the matrix method for finding the output of the encoder.

For the top adder in figure 8.5, the generator sequence is given by

$$g^{(1)} = [g_1^{(1)}\ g_2^{(1)}\ g_3^{(1)}] = [1\ 1\ 1]$$

Similarly for the bottom adder, the generator sequence

$$g^{(2)} = [g_1^{(2)}\ g_2^{(2)}\ g_3^{(2)}] = [1\ 0\ 1]$$

The generator matrix is of the order $L \times [n(L + m)] = 5 \times [2(5 + 2)] = 5 \times 14 \rightarrow 5$ rows and 14 columns.

$$\therefore \quad [G] = \begin{bmatrix} 11 & 10 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 10 & 11 \end{bmatrix}$$

The output of the convolutional encoder is given by

$$[C] = [d]\,[G]$$

$$= [1\,0\,0\,1\,1] \begin{bmatrix} 11 & 10 & 11 & 00 & 00 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 00 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 00 & 11 & 10 & 11 \end{bmatrix}$$

$$= [11, 10, 11, 11, 01, 01, 11]$$

(ii) Transform domain approach :

The message $[d] = [1\,0\,0\,1\,1]$

$$\therefore \quad d(x) = 1 + x^3 + x^4$$

The generator polynomials are given by

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

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

$\therefore$ The output polynomial of the adders are obtained from equation (8.8) given by

$$C^{(j)}(x) = d(x)\, g^{(j)}(x)$$

For the top adder (j = 1)

$$\begin{aligned} C^{(1)}(x) &= d(x)\, g^{(1)}(x) \\ &= (1 + x^3 + x^4)(1 + x + x^2) \\ &= 1 + x + x^2 + x^3 + x^4 + x^5 \\ &\qquad + x^4 + x^5 + x^6 \\ &= 1 + x + x^2 + x^3 + x^6 \end{aligned}$$

For the bottom adder (j = 2)

$$\begin{aligned} C^{(2)}(x) &= d(x)\, g^{(2)}(x) \\ &= (1 + x^3 + x^4)(1 + x^2) \\ &= 1 + x^3 + x^4 + x^2 + x^5 + x^6 \\ &= 1 + x^2 + x^3 + x^4 + x^5 + x^6 \end{aligned}$$

The output of the encoder is given by equation (8.9) with n = 2 as

$$\therefore \quad C(x) = C^{(1)}(x^2) + x\,C^{(2)}(x^2)$$
$$= 1 + x^2 + x^4 + x^6 + x^{12} + x\,(1 + x^4 + x^6 + x^8 + x^{10} + x^{12})$$
$$= 1 + x + x^2 + x^4 + x^5 + x^6 + x^7 + x^9 + x^{11} + x^{12} + x^{13}$$

$\therefore$ The output of the encoder is [11, 10, 11, 11, 01, 01, 11] which is same as before.

Example 8.5 : Consider the (3, 1, 2) convolutional code with $g^{(1)} = (1\ 1\ 0)$, $g^{(2)} = (1\ 0\ 1)$ and $g^{(3)} = (1\ 1\ 1)$.

(i) Draw the encoder block diagram.

(ii) Find the generator matrix.

(iii) Find the code-word corresponding to the information sequence (1 1 1 0 1) using time-domain and transform-domain approach. ***[VI Sem EC/TE, Jan/Feb. 2003, Q.8(a)]***

Solution

(i) The encoder has the parameters n = 3, k = 1, m = 2. i.e., it has 3 modulo-2 adders and 2 flip-flops. Figure 8.6 shows the complete encoder.

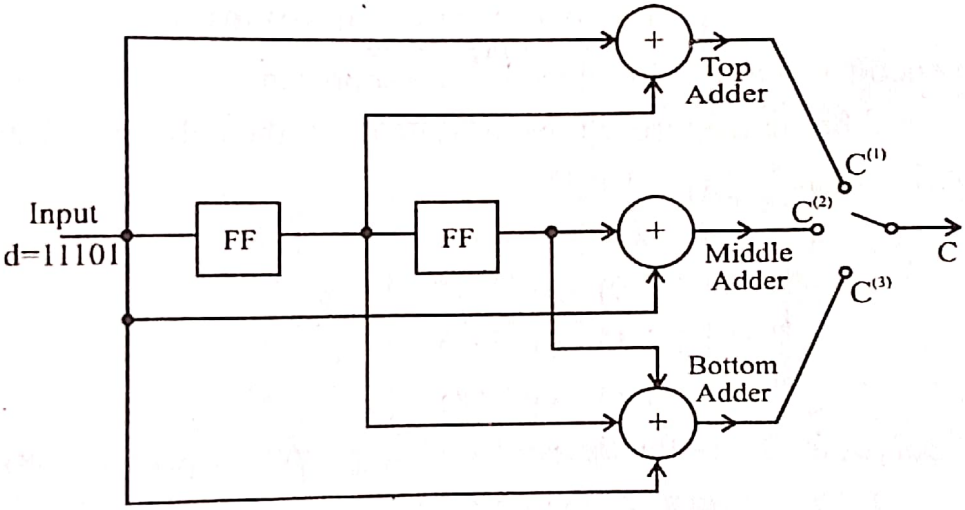


Fig. 8.6 : (3, 1, 2) convolutional encoder of example 8.5

(ii) In general, for a 3 modulo-2 adders convolutional encoder, the generator matrix G is given by

$$G = \begin{bmatrix} g_1^{(1)} g_1^{(2)} g_1^{(3)} & g_2^{(1)} g_2^{(2)} g_2^{(3)} & g_3^{(1)} g_3^{(2)} g_3^{(3)} & \cdots & g_{m+1}^{(1)} g_{m+1}^{(2)} g_{m+1}^{(3)} & 0 & 0 & 0 & \cdots & 000 \\ 0\ 0\ 0 & g_1^{(1)} g_1^{(2)} g_1^{(3)} & g_2^{(1)} g_2^{(2)} g_2^{(3)} & \cdots & g_m^{(1)} g_m^{(2)} g_m^{(3)} & g_{m+1}^{(1)} g_{m+1}^{(2)} g_{m+1}^{(3)} & & & \cdots & 000 \\ 0\ 0\ 0 & 0\ 0\ 0 & g_1^{(1)} g_1^{(2)} g_1^{(3)} & \vdots & \vdots & \vdots & & & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \vdots & \vdots \end{bmatrix}$$

[G] matrix has L rows and n(L + m) columns. In the encoder given, L = 5, n = 3, m = 2. Therefoire [G] has 5 rows and (3) (5 + 2) = 21 columns given by,

$$[G] = \begin{bmatrix} 111 & 101 & 011 & 000 & 000 & 000 & 000 \\ 000 & 111 & 101 & 011 & 000 & 000 & 000 \\ 000 & 000 & 111 & 101 & 011 & 000 & 000 \\ 000 & 000 & 000 & 111 & 101 & 011 & 000 \\ 000 & 000 & 000 & 000 & 111 & 101 & 011 \end{bmatrix}$$

(iii) The output of the encoder is given by

$$[C] = [d]\,[G]$$

$$\therefore \quad [C] = [1\,1\,1\,0\,1] \begin{bmatrix} 111 & 101 & 011 & 000 & 000 & 000 & 000 \\ 000 & 111 & 101 & 011 & 000 & 000 & 000 \\ 000 & 000 & 111 & 101 & 011 & 000 & 000 \\ 000 & 000 & 000 & 111 & 101 & 011 & 000 \\ 000 & 000 & 000 & 000 & 111 & 101 & 011 \end{bmatrix}$$

$$= [111, 010, 001, 110, 100, 101, 011]$$

This is the encoder output using time domain approach.

Using transform domain approach, let us write down the polynomials as given below :

We have $[d] = [1\,1\,1\,0\,1]$

$$\therefore \quad d(x) = 1 + x + x^2 + x^4$$

$$g^{(1)} = (1\,1\,0) \rightarrow g^{(1)}(x) = 1 + x$$

$$g^{(2)} = (1\,0\,1) \rightarrow g^{(2)}(x) = 1 + x^2$$

$$g^{(3)} = (1\,1\,1) \rightarrow g^{(3)}(x) = 1 + x + x^2$$

The output polynomials for the three adders are given by equation (8.8) as

$$C^{(j)}(x) = d(x)\, g^{(j)}(x) \text{ for } j = 1, 2, 3.$$

For the top adder ($j = 1$)

$$\begin{aligned} C^{(1)}(x) &= d(x)\, g^{(1)}(x) \\ &= (1 + x + x^2 + x^4)(1 + x) \\ &= 1 + x + x^2 + x^4 + x + x^2 + x^3 + x^5 \\ &= 1 + x^3 + x^4 + x^5 \end{aligned}$$

For the middle-adder ($j = 2$)

$$\begin{aligned} C^{(2)}(x) &= d(x)\, g^{(2)}(x) \\ &= (1 + x + x^2 + x^4)(1 + x^2) \\ &= 1 + x + x^2 + x^4 + x^2 + x^3 + x^4 + x^6 \\ &= 1 + x + x^3 + x^6 \end{aligned}$$

For the bottom-adder (j = 3)

$$\begin{aligned} C^{(3)}(x) &= d(x)\, g^{(3)}(x) \\ &= (1 + x + x^2 + x^4)(1 + x + x^2) \\ &= 1 + x + x^2 \\ &\quad + x + x^2 + x^3 \\ &\quad + x^2 + x^3 + x^4 \\ &\quad + x^4 + x^5 + x^6 \\ &= 1 + x^2 + x^5 + x^6 \end{aligned}$$

The output of the encoder is given by equation (8.9) with n = 3 as

$$\begin{aligned} C(x) &= C^{(1)}(x^3) + x\, C^{(2)}(x^3) + x^2\, C^{(3)}(x^3) \\ &= 1 + x^9 + x^{12} + x^{15} + x(1 + x^3 + x^9 + x^{18}) + x^2(1 + x^6 + x^{15} + x^{18}) \\ &= 1 + x + x^2 + x^4 + x^8 + x^9 + x^{10} + x^{12} + x^{15} + x^{17} + x^{19} + x^{20} \end{aligned}$$

∴ The output of the encoder is [111, 010, 001, 110, 100, 101, 011] which is same as before.

Example 8.6 : For the convolutional encoder shown in figure 8.7.

(i) Find the impulse response and hence calculate the output produced by the information sequence 10111.

(ii) Write the generator polynomials of the encoder and recompute the output for the input of (i) and compare with that of (i).

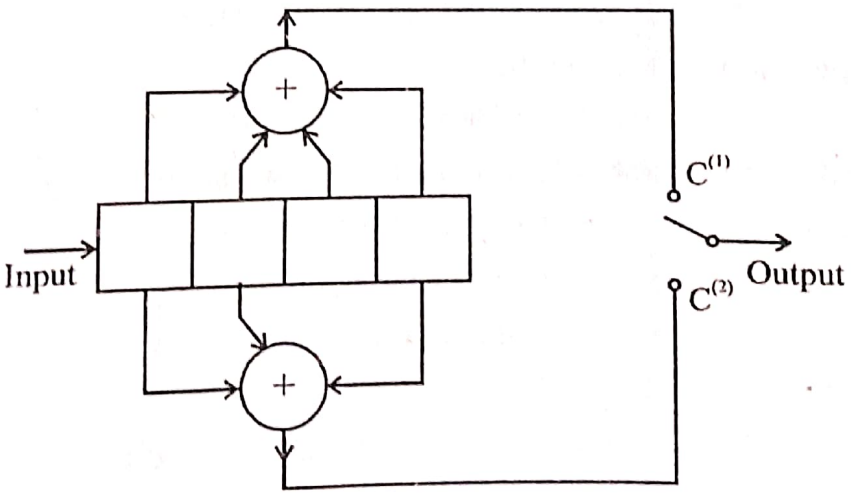


Fig. 8.7 : A (2, 1, 4) convolutional encoder of example 8.6

Solution

(i) Since there is no connection from the input to the two adders, the generator sequences (impulse responses) are given by

$$g^{(1)} = [\,0\ 1\ 1\ 1\ 1\,] \text{ for the top-adder.}$$

$$g^{(2)} = [0\ 1\ 1\ 0\ 1] \text{ for the bottom adder.}$$
$$d = [1\ 0\ 1\ 1\ 1] \text{ is the message input.}$$

The parameters of the encoder are L = 5, n = 2, k = 1, m = 4.

$\therefore$ Number of rows in [G] = L = 5

Number of columns in [G] = n (L + m)

= 2 (5 + 4) = 18

The [G] matrix is then constructed as below [the general form is given in example 8.1].

$$[G] = \begin{bmatrix} 00 & 11 & 11 & 10 & 11 & 00 & 00 & 00 & 00 \\ 00 & 00 & 11 & 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 00 & 00 & 11 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 00 & 00 & 11 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 00 & 00 & 11 & 11 & 10 & 11 \end{bmatrix}$$

The encoder output is given by [C] = [d] [G]

$$\therefore \quad [C] = [1\ 0\ 1\ 1\ 1] \begin{bmatrix} 00 & 11 & 11 & 10 & 11 & 00 & 00 & 00 & 00 \\ 00 & 00 & 11 & 11 & 10 & 11 & 00 & 00 & 00 \\ 00 & 00 & 00 & 11 & 11 & 10 & 11 & 00 & 00 \\ 00 & 00 & 00 & 00 & 11 & 11 & 10 & 11 & 00 \\ 00 & 00 & 00 & 00 & 00 & 11 & 11 & 10 & 11 \end{bmatrix}$$

$$= [00, 11, 11, 01, 11, 10, 10, 01, 11]$$

(ii) The message polynomial is given by

$$d(x) = 1 + x^2 + x^3 + x^4$$

The corresponding generator polynomials for the two adders are given by

$$g^{(1)}(x) = x + x^2 + x^3 + x^4 \text{ for the top adder}$$
$$g^{(2)}(x) = x + x^2 + x^4 \text{ for the bottom adder.}$$

The output polynomial for the top adder is

$$\begin{aligned} C^{(1)}(x) &= d(x)\, g^{(1)}(x) \\ &= (1 + x^2 + x^3 + x^4)(x + x^2 + x^3 + x^4) \\ &= x + x^2 + x^3 + x^4 \\ &\quad + x^3 + x^4 + x^5 + x^6 \\ &\quad + x^4 + x^5 + x^6 + x^7 \\ &\quad + x^5 + x^6 + x^7 + x^8 \\ &= x + x^2 + x^4 + x^5 + x^6 + x^8 \end{aligned}$$

The output polynomial for the bottom adder is

$$
\begin{aligned}
C^{(2)}(x) &= d(x)\, g^{(2)}(x) \\
&= (1 + x^2 + x^3 + x^4)(x + x^2 + x^4) \\
&= x + x^2 + \quad + x^4 \\
&\qquad + x^3 + x^4 \qquad + x^6 \\
&\qquad\quad + x^4 + x^5 \qquad + x^7 \\
&\qquad\qquad + x^5 + x^6 \qquad + x^8 \\
&= x + x^2 + x^3 + x^4 + x^7 + x^8
\end{aligned}
$$

The output of the encoder is given by equation (8.9) with n =2 as

$$
\begin{aligned}
C(x) &= C^{(1)}(x^2) + x\, C^{(2)}(x^2) \\
&= x^2 + x^4 + x^8 + x^{10} + x^{12} + x^{16} + x\,(x^2 + x^4 + x^6 + x^8 + x^{14} + x^{16}) \\
&= x^2 + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{12} + x^{15} + x^{16} + x^{17}
\end{aligned}
$$

$\therefore$ The output of the encoder is [00, 11, 11, 01, 11, 10, 10, 01, 11] which is same as obtained in (i).

Example 8.7 : For the (3, 2, 1) encoder shown in figure 8.8, find code-word C for input sequences of $d^{(1)} = 101$ and $d^{(2)} = 110$ using

(a) Time-domain approach (using generator matrix)

(b) Transform domain approach by constructing transfer function matrix.

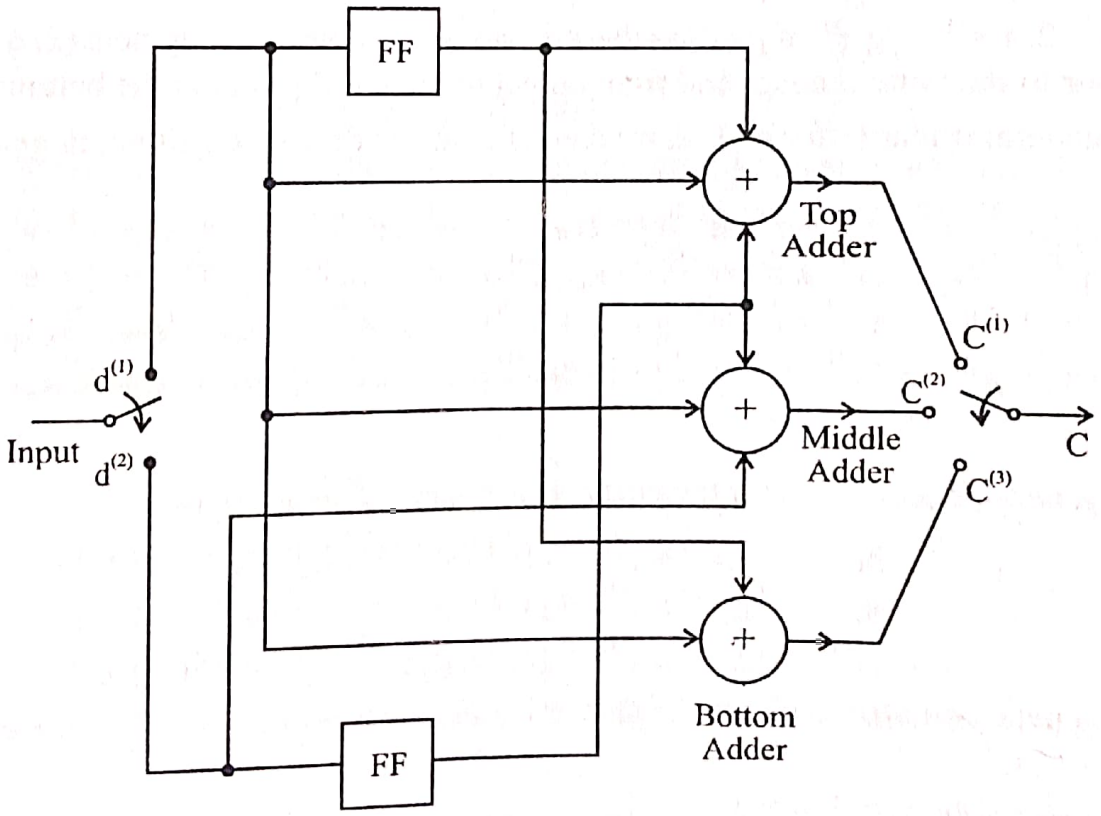


Fig. 8.8 : A (3, 2, 1) convolutional encoder of example 8.7

Solution

(a) The encoder shown in figure 8.8 consists of two m = 1 stage shift registers, with n = 3 modulo-2 adders and two commutators, one at the input and the other at the output. The number of input bits entering through input commutator is k =2.

For k = 2, in general, the input message sequences can be written as

$$d^{(1)} = [d_1^{(1)}\, d_2^{(1)}\, d_3^{(1)} \ldots\ldots]$$

$$\text{and} \quad d^{(2)} = [d_1^{(2)}\, d_2^{(2)}\, d_3^{(2)} \ldots\ldots]$$

The generator sequences corresponding to "i" number of segments in the input commutator and "j" number of segments in the output commutator is given by

$$g_i^{(j)} = [g_{i,1}^{(j)}\, g_{i,2}^{(j)} \ldots\ldots g_{i,m+1}^{(j)}]$$

There are 2 segments in the input commutator and 3 in output commutator. Therefore 'i' takes values 1 and 2 and j, 1, 2 and 3.

For i = 1, j = 1, $g_1^{(1)} = [1, 1]$ as there is connection from segment (1) of the input commutator as well as from the output of top flip-flop to the top adder.

For i = 1, j = 2, $g_1^{(2)} = [1, 0]$. There is no connection from the output of top flip-flop to the middle adder and hence 0.

For i = 1, j = 3, $g_1^{(3)} = [1, 1]$

For i = 2, j = 1, $g_2^{(1)} = [0, 1]$. There is no connection from segment (2) of the input commutator to the top adder and hence '0'.

For i = 2, j = 2, $g_2^{(2)} = [1, 1]$

For i = 2, j = 3, $g_2^{(3)} = [0, 0]$ as there are no connections from segment (2) of the input commutator to the bottom adder and from output of bottom flip-flop to the bottom adder.

The generator matrix for a (3, 2, m) convolutional code can be written, in general, as

$$[G] = \begin{bmatrix} g_{11}^{(1)} g_{11}^{(2)} g_{11}^{(3)} & g_{12}^{(1)} g_{12}^{(2)} g_{12}^{(3)} & \cdots & g_{1,m+1}^{(1)} g_{1,m+1}^{(2)} g_{1,m+1}^{(3)} & 0 & 0 & 0 & \cdots & 0 \\ g_{21}^{(1)} g_{21}^{(2)} g_{21}^{(3)} & g_{22}^{(1)} g_{22}^{(2)} g_{22}^{(3)} & \cdots & g_{2,m+1}^{(1)} g_{2,m+1}^{(2)} g_{2,m+1}^{(3)} & 0 & 0 & 0 & \cdots & 0 \\ 0\ 0\ 0 & g_{11}^{(1)} g_{11}^{(2)} g_{11}^{(3)} & \cdots & g_{1,m}^{(1)} g_{1,m}^{(2)} g_{1,m}^{(3)} & g_{1,m+1}^{(1)} g_{1,m+1}^{(2)} g_{1,m+1}^{(3)} & & & \cdots & 0 \\ 0\ 0\ 0 & g_{21}^{(1)} g_{21}^{(2)} g_{21}^{(3)} & \cdots & g_{2,m}^{(1)} g_{2,m}^{(2)} g_{2,m}^{(3)} & g_{2,m+1}^{(1)} g_{2,m+1}^{(2)} g_{2,m+1}^{(3)} & & & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & & & \vdots \end{bmatrix}$$

The generator sequences for the encoder of figure 8.8 are given by

$$g_1^{(1)} = [g_{11}^{(1)}\, g_{12}^{(1)}] = [1, 1],\ g_2^{(1)} = [g_{21}^{(1)}\, g_{22}^{(1)}] = [0, 1]$$

$$g_1^{(2)} = [g_{11}^{(2)}\, g_{12}^{(2)}] = [1, 0],\ g_2^{(2)} = [g_{21}^{(2)}\, g_{22}^{(2)}] = [1, 1]$$

$$g_1^{(3)} = [g_{11}^{(3)}\, g_{12}^{(3)}] = [1, 1],\ g_2^{(3)} = [g_{21}^{(3)}\, g_{22}^{(3)}] = [0, 0]$$

The generator matrix will now have L + L = 2L number of rows and n (L + m) number of columns.

In the example, L = 3, n = 3, m = 1

∴ [G] will have 6 rows and 12 columns given by

$$[G] = \begin{bmatrix} 111 & 101 & 000 & 000 \\ 010 & 110 & 000 & 000 \\ 000 & 111 & 101 & 000 \\ 000 & 010 & 110 & 000 \\ 000 & 000 & 111 & 101 \\ 000 & 000 & 010 & 110 \end{bmatrix}$$

∴ The output of the given convolutional encoder is given by

$$[C] = [d]\,[G]$$

where $[d] = [d_1^{(1)}\ d_1^{(2)}\ d_2^{(1)}\ d_2^{(2)}\ d_3^{(1)}\ d_3^{(2)}]$

It is given that $d^{(1)} = [d_1^{(1)}\ d_2^{(1)}\ d_3^{(1)}] = [1\ 0\ 1]$

$d^{(2)} = [d_1^{(2)}\ d_2^{(2)}\ d_3^{(2)}] = [1\ 1\ 0]$

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

$$\therefore\ [C] = [1\ 1\ 0\ 1\ 1\ 0]\begin{bmatrix} 111 & 101 & 000 & 000 \\ 010 & 110 & 000 & 000 \\ 000 & 111 & 101 & 000 \\ 000 & 010 & 110 & 000 \\ 000 & 000 & 111 & 101 \\ 000 & 000 & 010 & 110 \end{bmatrix}$$

$$\therefore\ \mathbf{[C] = [101, 001, 001, 101]}$$

(b) The generator polynomials corresponding to generator sequences are given by

$g_1^{(1)} = [1, 1] \rightarrow g_1^{(1)}(x) = 1 + x, \quad g_2^{(1)} = [0, 1] \rightarrow g_2^{(1)}(x) = x$

$g_1^{(2)} = [1, 0] \rightarrow g_1^{(2)}(x) = 1, \quad g_2^{(2)} = [1, 1] \rightarrow g_2^{(2)}(x) = 1 + x,$

$g_1^{(3)} = [1, 1] \rightarrow g_1^{(3)}(x) = 1 + x, \quad g_2^{(3)} = [0, 0] \rightarrow g_2^{(3)}(x) = 0$

∴ The generator polynomial matrix also called "TRANSFER FUNCTION MATRIX" is then given by

$$G(x) = \begin{bmatrix} g_1^{(1)}(x) & g_1^{(2)}(x) & g_1^{(3)}(x) \\ g_2^{(1)}(x) & g_2^{(2)}(x) & g_2^{(3)}(x) \end{bmatrix}$$

$$\therefore\ G(x) = \begin{bmatrix} 1+x & 1 & 1+x \\ x & 1+x & 0 \end{bmatrix}$$

The information sequence polynomials or message polynomials are given by

$d^{(1)} = [1\ 0\ 1] \rightarrow d^{(1)}(x) = 1 + x^2$

$d^{(2)} = [1\ 1\ 0] \rightarrow d^{(2)}(x) = 1 + x$

The code polynomials at the output of the adders are then given by

$$C(x) = [C^{(1)}(x) \quad C^{(2)}(x) \quad C^{(3)}(x)] = d(x)\ G(x)$$

$$= [(1+x^2)\ (1+x)] \begin{bmatrix} 1+x & 1 & 1+x \\ x & 1+x & 0 \end{bmatrix}$$

$$= \left[(1+x^2)(1+x)+(1+x)(x) \quad (1+x^2)+(1+x)^2 \quad (1+x^2)(1+x)\right]$$

$$= \left[1+x^2+x+x^3+x+x^2 \quad 1+x^2+1+x^2 \quad 1+x^2+x+x^3\right]$$

$$\therefore \quad C(x) = \left[1+x^3 \quad 0 \quad 1+x+x^2+x^3\right]$$

$$\therefore \quad C^{(1)}(x) = 1+x^3$$

$$C^{(2)}(x) = 0$$

$$C^{(3)}(x) = 1+x+x^2+x^3$$

The output code-word is given by equation (8.9) with n = 3 as

$$C(x) = C^{(1)}(x^3) + x\,C^{(2)}(x^3) + x^2\,C^{(3)}(x^3)$$

$$= 1+x^9+(x)\,(0)+x^2\,(1+x^3+x^6+x^9)$$

$$= 1+x^2+x^5+x^8+x^9+x^{11}$$

∴ The code-word at the output of the encoder is

$$C = [101, 001, 001, 101] \text{ which is same as before.}$$

8.4 STATE DIAGRAMS AND CODE-TREE OF CONVOLUTIONAL CODES

The state of an encoder is defined as its shift register contents. For an (n, k, m) code, with k > 1, the ith shift register contains k_i previous information bits. Defining $K \triangleq \sum_{i=1}^{k} K_i$ as the total encoder memory (m – represents the memory order which is the maximum length of any shift register), the encoder state at time unit "l", when the encoder inputs are $[d_l^{(1)}, d_l^{(2)}, \ldots d_l^{(k)}]$ is the binary k-tuple of inputs $[d_{l-1}^{(1)}, d_{l-2}^{(1)}, \ldots\ d_{l-k}^{(1)}; d_{l-1}^{(2)}, d_{l-2}^{(2)}, \ldots\ d_{l-k}^{(2)}; \ldots\ d_{l-1}^{(k)}, d_{l-2}^{(k)}, \ldots\ d_{l-k}^{(k)}]$ and there are a total of 2^k different possible states.

For a (n, 1, m) encoder, $K = K_1 = m$ and the encoder state at time unit 'l' is simply

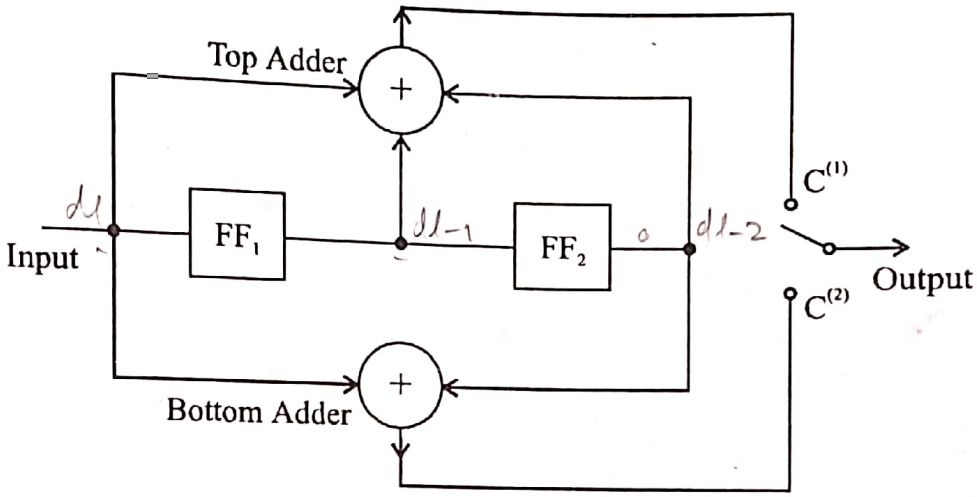
$$[d_{l-1}, d_{l-2}, \ldots, d_{l-m}]$$

Each new block of k-inputs causes a transition to a new state. Hence there are 2^k - branches leaving each state one each corresponding to the input block. For a (n, 1, m) code, there are only two branches leaving each state. On the state diagram, each branch is labelled with the k-inputs causing the transition and the n-corresponding outputs usually in parenthesis.

The procedure for writing the state diagram consists of 3 parts. In part-1, the different possible states are identified and state table is constructed. In part-2, the state transition table is constructed showing clearly the present state, next state, input bits and the corresponding

output bits. In the last part, the state diagram is drawn showing all the states of the flip-flops, interconnection between them for various input combinations and the corresponding outputs. Using the state diagram, the code-tree can easily be drawn as shown in the following example.

Example 8.8 : Consider the binary convolutional encoder shown in figure 8.9. Draw the state table, state transition table, state diagram and the corresponding code tree. Using the code tree, find the encoded sequence for the message (1 0 1 1 1). Verify the output sequence so obtained using transform domain approach.



(2, 1, 2)
(n, k, m)

Fig. 8.9 : A (2, 1, 2) convolutional encoder of example 8.8

Solution

Part-1 : State Table

There are 2 flip-flops in the shift register and hence there are $2^2 = 4$ states. Let the states be S_0, S_1, S_2 and S_3 represented as shown in table 8.1.

State	S_0	S_1	S_2	S_3
Binary Description	00	10	01	11

Table 8.1 : State table for example 8.8

Part-2 : State Transition Table

From figure 8.9, the outputs of the two modulo-2 adders (shown as top and bottom adder) $C^{(1)}$ and $C^{(2)}$ can be represented as output sequences given by

$$C^{(1)} = d_l + d_{l-1} + d_{l-2} \text{ for the top-adder}$$

$$C^{(2)} = d_l + d_{l-2} \text{ for the bottom-adder.}$$

The state transition table can be constructed as shown in table 8.2.

Present state	Binary Description	Input	Next state	Binary Description	d_l	d_{l-1}	d_{l-2}	Output $C^{(1)}$	Output $C^{(2)}$
S_0	00	0	S_0	00	0	0	0	0	0
		1	S_1	10	1	0	0	1	1
S_1	10	0	S_2	01	0	1	0	1	0
		1	S_3	11	1	1	0	0	1
S_2	01	0	S_0	00	0	0	1	1	1
		1	S_1	10	1	0	1	0	0
S_3	11	0	S_2	01	0	1	1	0	1
		1	S_3	11	1	1	1	1	0

Table 8.2 : State transition table for example 8.8

Construction of State Transition Table

Let the start state of the flip-flops be the cleared state '00' $\rightarrow S_0$. With an input of '0', the shift register remains at '00' $\rightarrow S_0$ state. The two outputs of the adders will be $C^{(1)}\ C^{(2)} = 00$.

With an input of '1', the first flip-flop FF_1 changes state from '0' to '1' and the contents of FF_1 (which was '0') gets shifted to FF_2. The new contents of the shift register are '10' $\rightarrow S_1$. This transition is caused by a '1' input and therefore $d_l\ d_{l-1}\ d_{l-2} = 1\ 0\ 0$. The corresponding output bits are $C^{(1)}\ C^{(2)} = 1\ 1$. These conditions are entered in the tabular column of table 8.2.

For the next state $10 \rightarrow S_1$, the contents of the flip-flops are analysed and entered in the table. This process is continued for all the states and the table is completed as shown.

Part - 3 : State Diagram

In the state diagram of figure 8.10, the four states are represented by small circles labelled S_0, S_1, S_2 and S_3. Consider the state S_0. With '0' input, the shift register remains at same state S_0 with output sequence of 00. This is shown as a loop starting from S_0 and ending at S_0 with an arrow mark and being labelled as 0(00) $\rightarrow$ the '0' outside the bracket shows the input '0' and '00' within the brackets are the output sequences. With '1' input, the shift register moves to state "10" $\rightarrow S_1$. This is shown as a line starting from S_0 and ending at S_1 with an arrow mark pointing towards S_1 and being labelled as 1 (11) $\rightarrow$ the '1' outside the bracket is the input and '11' inside the bracket are the corresponding outputs. In this way, the compete state diagram is drawn as shown in figure 8.10.

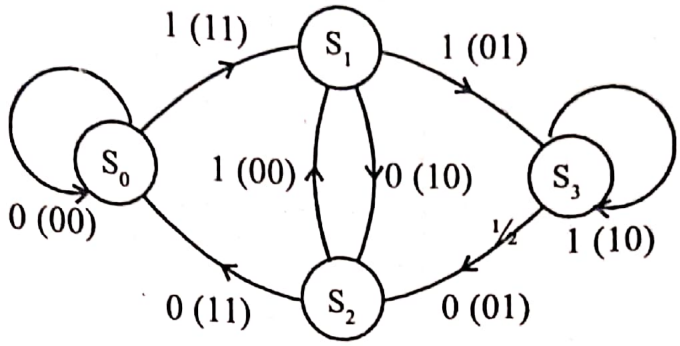


Fig. 8.10 : State diagram for example 8.8

Part - 4 : Code Tree

The state diagram of figure 8.10 can be re-drawn as a ***"code-tree"*** as shown in figure 8.11. The rules followed while constructing the code tree is as follows.

If the input is a '0', then the upper path is followed and if the input is a '1', then lower path is followed. The vertical line represents a ***"node"*** and horizontal line is called a ***"branch"***. The output code-words $[C^{(1)}\ C^{(2)}]$ for each input bit are shown on the branches. The encoder output for any information sequence can be traced through the code-tree paths.

For an information sequence of 1 0 1 1 1, we can traverse through the code-tree in a path shown by dotted lines. While doing so, we pass through the nodes $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_1 \rightarrow S_3 \rightarrow S_3$ producing the outputs (11, 10, 00, 01, 10). This is not the complete code sequence since the total code-length is n (L + m) = 2 (5 + 2) = 14 bits. The number of flip-flops in the encoder = m = 2 and hence we require two more bits to clear and reset the encoder. Hence, to obtain the complete code sequence corresponding to an information sequence of length kL, the tree graph has to be extended, in general, by n (m – 1) time units and this extended part is called ***"tail of the tree"***. Thus the extended tree diagram for the (2, 1, 2) encoder, for the information sequence 10111 is shown in figure 8.11. In order to get the tail of the tree, it is assumed that the inputs are '0's. From the code-tree, the complete information sequence is (11, 10, 00, 01, 10, 01, 11).

The same output sequence can also be obtained by traversing through the state diagram of figure 8.10. The outputs are given inside the parenthesis and the input outside. For an information sequence of 10111, we can now traverse through the state diagram through the states from $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_1 \rightarrow S_3 \rightarrow S_3 \rightarrow S_2 \rightarrow S_0$ getting an output sequence (11, 10, 00, 01, 10, 01, 11) which is same as obtained from code-tree. Also note that the last two input bits are again '0's to get the complete sequence.

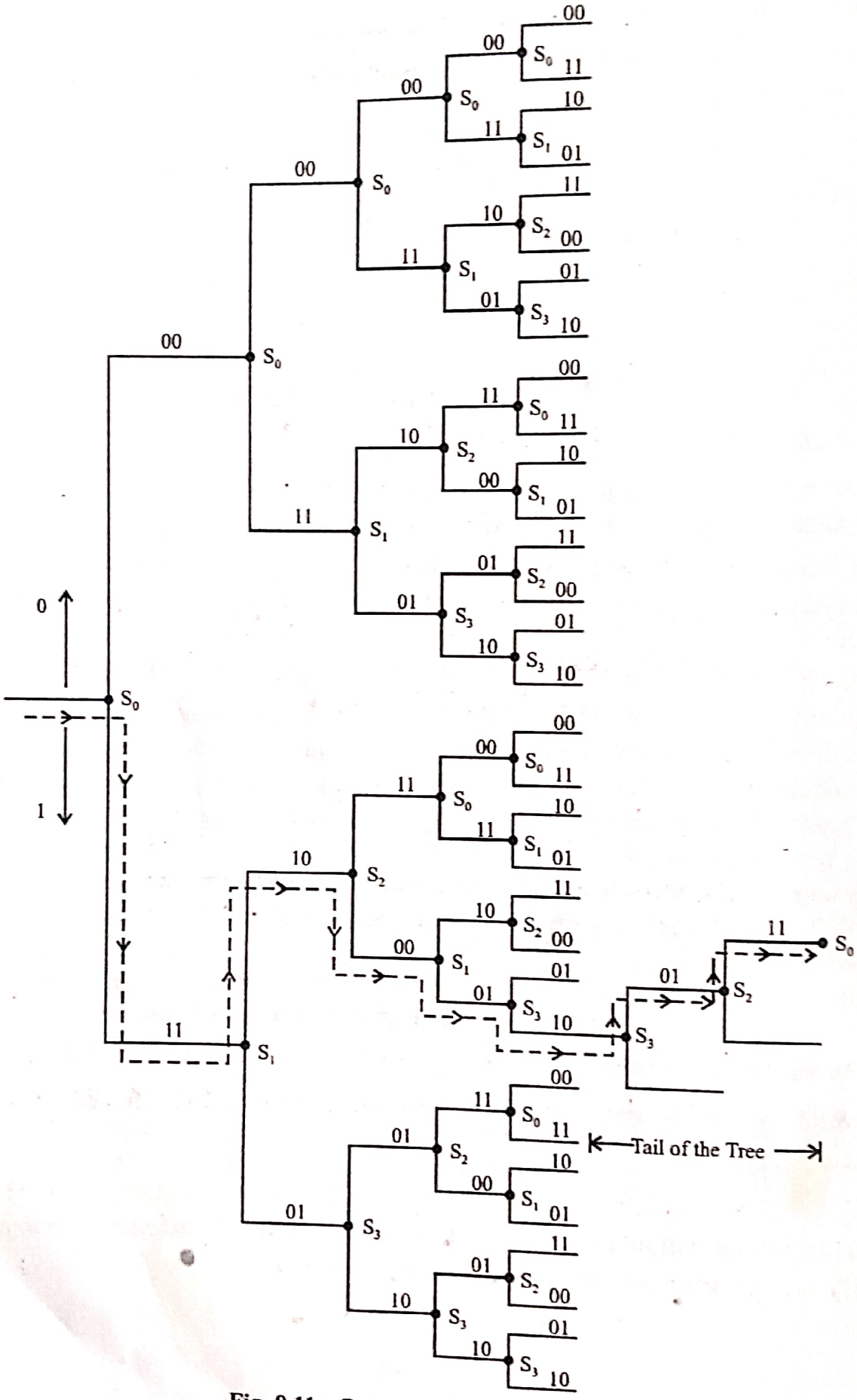


Fig. 8.11 : Code-tree for example 8.8

Verification through Transform Domain Approach :

The generator sequences and the corresponding generator polynomials are given by

$g^{(1)} = [1\ 1\ 1] \therefore g^{(1)}(x) = 1 + x + x^2$

$g^{(2)} = [1\ 0\ 1] \therefore g^{(2)}(x) = 1 + x^2$

The message $d = [1\ 0\ 1\ 1\ 1]$

$\therefore \quad d(x) = 1 + x^2 + x^3 + x^4$

For the top-adder

$$\begin{aligned} C^{(1)}(x) &= d(x)\, g^{(1)}(x) \\ &= (1 + x^2 + x^3 + x^4)(1 + x + x^2) \\ &= 1 + \quad + x^2 + x^3 + x^4 \\ &\quad + x + \quad + x^3 + x^4 + x^5 \\ &\quad + x^2 \quad + x^4 + x^5 + x^6 \\ &= 1 + x + x^4 + x^6 \end{aligned}$$

For the bottom-adder

$$\begin{aligned} C^{(2)}(x) &= d(x)\, g^{(2)}(x) \\ &= (1 + x^2 + x^3 + x^4)(1 + x^2) \\ &= 1 + x^2 + x^3 + x^4 \\ &\quad + x^2 \quad + x^4 + x^5 + x^6 \\ &= 1 + x^3 + x^5 + x^6 \end{aligned}$$

The output of the encoder is given by equation (8.9) with n = 2 as

$$\begin{aligned} C(x) &= C^{(1)}(x^2) + x\, C^{(2)}(x^2) \\ &= 1 + x^2 + x^8 + x^{12} + x\,[1 + x^6 + x^{10} + x^{12}] \\ &= 1 + x + x^2 + x^7 + x^8 + x^{11} + x^{12} + x^{13} \end{aligned}$$

$\therefore$ The output code sequence is $C \rightarrow [11, 10, 00, 01, 10, 01, 11]$ which is same as before.

Example 8.9 : Consider the convolutional encoder shown in figure 8.12. The code is systematic.

(i) Draw the state diagram

(ii) Draw the code tree.

(iii) Find the encoder output produced by the message sequence 10111.

(iv) Verify the output using time-domain approach (matrix method).

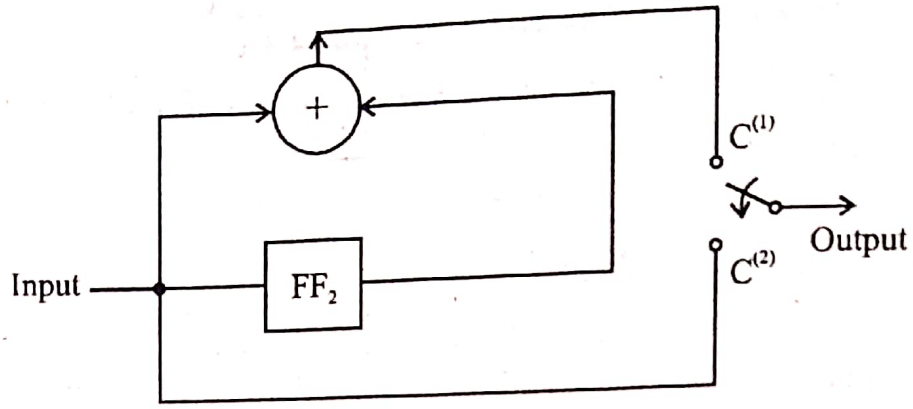


Fig. 8.12 : A (2, 1, 1) convolutional encoder of example 8.9

Solution

Part - 1:

Since there is only one flip-flop, we have only 2 states. Let the two states be represented by $S_0 \rightarrow 0$ and $S_1 \rightarrow 1$.

Part - 2: State Transition Table

From figure 8.12, the outputs of the encoder are given by

$$C^{(1)} = d_l + d_{l-1}$$

$$\text{and } C^{(2)} = d_l$$

The state transition table can be constructed as shown in table 8.3.

Present state	Binary Description	Input	Next state	Binary Description	d_l	d_{l-1}	Output $C^{(1)}$	Output $C^{(2)}$
S_0	0	0	S_0	0	0	0	0	0
		1	S_1	1	1	0	1	1
S_1	1	0	S_0	0	0	1	1	0
		1	S_1	1	1	1	0	1

Table 8.3 : State transition table for example 8.9

Part - 3: State diagram :

There are only two states S_0 and S_1 and hence the state diagram appears to be very simple as shown in figure 8.13.

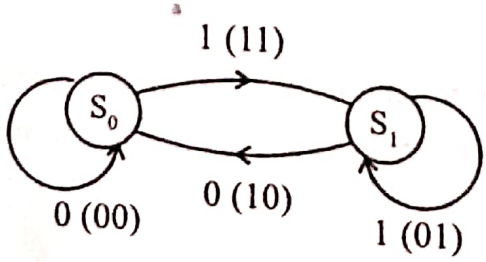


Fig. 8.13 : State diagram for example 8.9

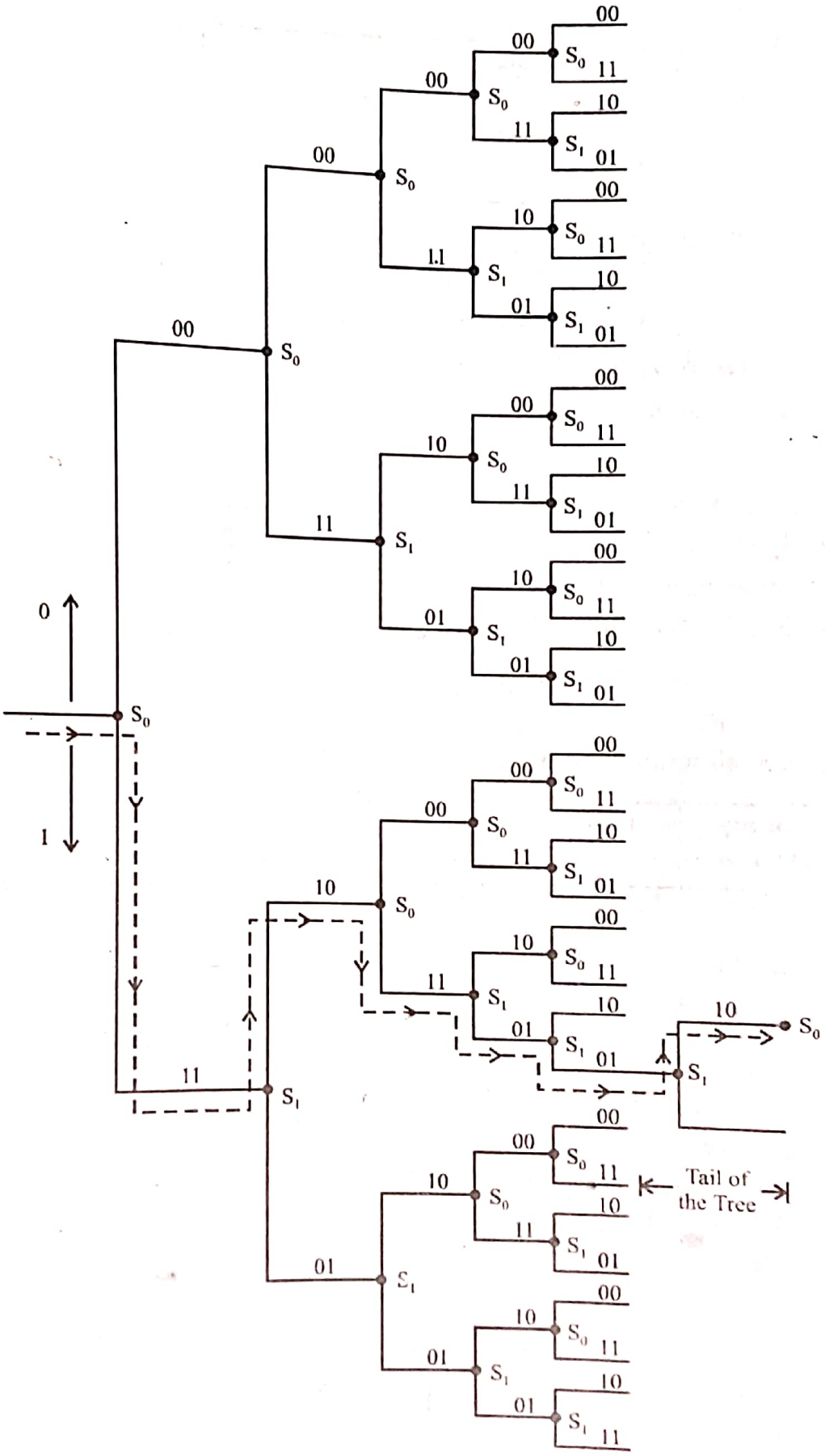


Fig. 8.14 : Code-tree for example 8.9

For the given message sequence of 10111, we can traverse through the state diagram of figure 8.13 starting from state $S_0 \rightarrow S_1 \rightarrow S_0 \rightarrow S_1 \rightarrow S_1 \rightarrow S_1 \rightarrow S_0$ producing the output sequence (11, 10, 11, 01, 01, 10). [Again note that the last pair of output is obtained by feeding a '0' at the input].

Part - 4: Code Tree :

The code-tree corresponding to the state diagram of figure 8.13 can be drawn as shown in figure 8.14. For the given message sequence of 10111, we can traverse through the code-tree of figure 8.14 in a path shown by broken lines producing an output sequence (11, 10, 11, 01, 01, 10, 00) which is same as before.

Verification through time-domain approach :

The generator sequences are given by

$$g^{(1)} = [1, 1]$$

$$\text{and} \quad g^{(2)} = [1, 0]$$

The generator matrix has L = 5 rows and n (L + m) = 2 (5 + 1) = 12 columns given by

$$[G] = \begin{bmatrix} 11 & 10 & 00 & 00 & 00 & 00 \\ 00 & 11 & 10 & 00 & 00 & 00 \\ 00 & 00 & 11 & 10 & 00 & 00 \\ 00 & 00 & 00 & 11 & 10 & 00 \\ 00 & 00 & 00 & 00 & 11 & 10 \end{bmatrix}$$

The encoder output is given by C = d G

$$\therefore \quad C = [1\,0\,1\,1\,1] \begin{bmatrix} 11 & 10 & 00 & 00 & 00 & 00 \\ 00 & 11 & 10 & 00 & 00 & 00 \\ 00 & 00 & 11 & 10 & 00 & 00 \\ 00 & 00 & 00 & 11 & 10 & 00 \\ 00 & 00 & 00 & 00 & 11 & 10 \end{bmatrix}$$

$$= [11, 10, 11, 01, 01, 10] \text{ which is same as before.}$$

Example 8.10 : Figure 8.15 shows a (2, 1, 3) convolutional encoder

(i) Draw the state diagram

(ii) Draw the code tree.

(iii) Find the encoder output produced by the message sequence 11101 by traversing through the code-tree.

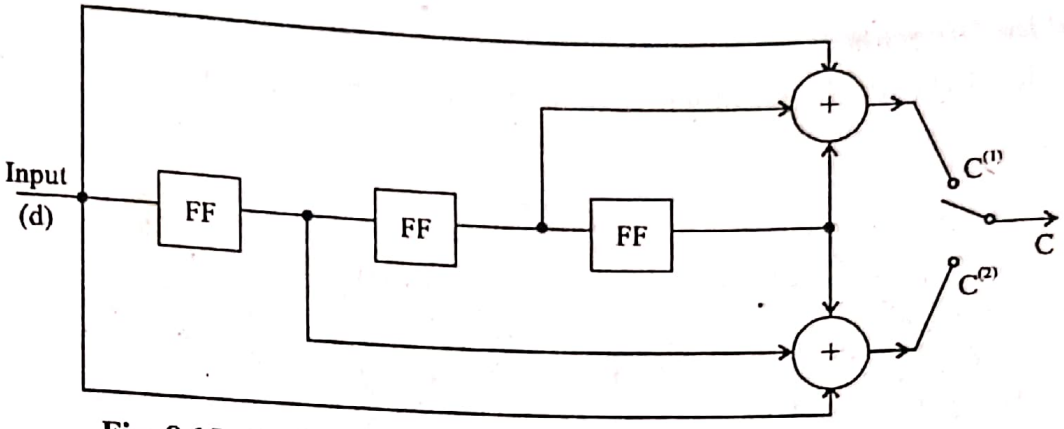


Fig. 8.15 : A (2, 1, 3) convolutional encoder of example 8.10

Solution

Part - 1: State Table :

There are 3 flip-flops in the shift register and hence there are $2^3 = 8$ states. Let the states S_0 to S_7 be represented as shown in table 8.4.

State	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Binary Description	000	100	010	110	001	101	011	111

Table 8.4 : State table for example 8.10

Part - 2 : State Transition Table :

Looking at the encoder of figure 8.15, the outputs are given by

$$C^{(1)} = d_l + d_{l-2} + d_{l-3} \quad \text{and} \quad C^{(2)} = d_l + d_{l-1} + d_{l-3}$$

The state transmission table can now be constructed as shown in table 8.5.

Present state	Binary Description	Input	Next state	Binary Description	d_l	d_{l-1}	d_{l-2}	d_{l-3}	Output $C^{(1)}$	Output $C^{(2)}$
S_0	000 0	0	S_0	000	0	0	0	0	0	0
		1	S_1	100	1	0	0	0	1	1
S_1	100 4	0	S_2	010	0	1	0	0	0	1
		1	S_3	110	1	1	0	0	1	0
S_2	010 2	0	S_4	001	0	0	1	0	1	0
		1	S_5	101	1	0	1	0	0	1
S_3	110 6	0	S_6	011	0	1	1	0	1	1
		1	S_7	111	1	1	1	0	0	0
S_4	001 1	0	S_0	000	0	0	0	1	1	1
		1	S_1	100	1	0	0	1	0	0
S_5	101 5	0	S_2	010	0	1	0	1	1	0
		1	S_3	110	1	1	0	1	0	1
S_6	011 3	0	S_4	001	0	0	1	1	0	1
		1	S_5	101	1	0	1	1	1	0
S_7	111 7	0	S_6	011	0	1	1	1	0	0
		1	S_7	111	1	1	1	1	1	1

Table 8.5 : State transition table for example 8.10

Part - 3: State Diagram :

Figure 8.16 shows the complete state diagram comprising the eight states S_0 to S_7.

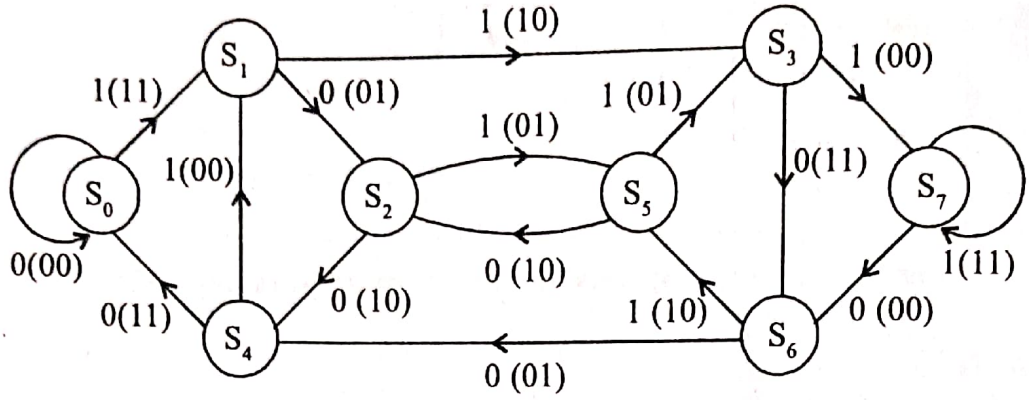


Fig. 8.16 : State diagram for example 8.10

Part - 4: Code Tree :

The code-tree corresponding to the state diagram of figure 8.16 can be drawn as shown in figure 8.17. For the given message sequence of 11101 we can traverse through the code-tree of figure 8.17 in a path shown by broken lines producing an output sequence [11, 10, 00, 00, 10, 10, 10, 11].

8.5 TRELLIS DIAGRAM

In the previous section, we have seen that the structural properties of a convolutional encoder are represented in graphical form by using either code tree or the state diagram. The state diagram is not sufficient to understand the implicacies of a convolutional encoder in detail. On the other hand, the code-tree becomes enormously large when the number of input bits keep on coming. In fact, the number of branches of a code-tree grows in geometric progression with every input bit. Thus, the state diagram of a convolutional encoder is too small and code-tree is too large. Therefore, we would like to have a diagram which contains all the information of a convolutional encoder which is neither too small (as state diagram) nor too large (as code-tree). Fortunately, such a diagram exists called as "***TRELLIS***". A trellis is a tree-like structure with remerging branches. A trellis is more instructive than a code-tree in that it brings out explicitely the fact that the convolutional encoder is a finite-state machine. Trellis diagram is more helpful in decoding the received sequence than either state diagram or code-tree.

In order to understand the construction of trellis diagram, let us consider the example of code-tree of example 8.8 given in fig. 8.11. Careful observation of code-tree of fig. 8.11 reveals the fact that it becomes repetitive after the first ***three*** branches. In fact, beyond the third branch, the two nodes labeled "S_0" are identical and so are all the other node-pairs that are identically labelled. This repetitive property may be established by examining the convolutional encoder of fig. 8.9. The encoder has memory of 2 bits. When the third message bit enters the encoder, the first message bit is shifted out of the register. Consequently, after

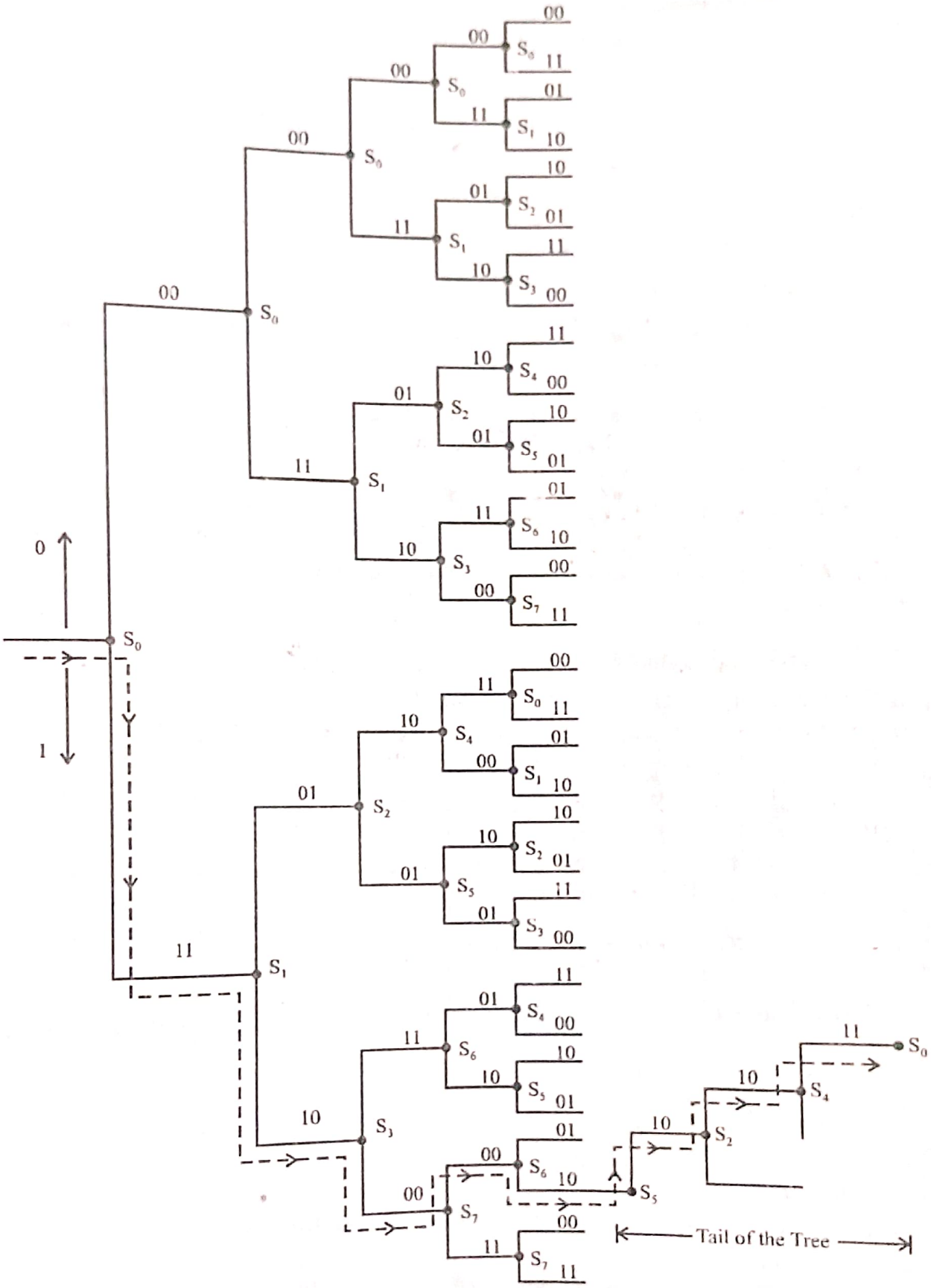


Fig. 8.17 : Code-tree for example 8.10

the third branch, the message sequences 100 d_3 d_4 ..... and 000 d_3 d_4 generate the same code symbols and the pair of nodes labelled "S_0" may be joined together. A similar reasoning applies to other pair of nodes.

We may thus collapse the code-tree of fig. 8.11 into a new form shown in fig. 8.18 which is called as trellis. The convention used in the construction of trellis is that a code branch produced by an input "0" is drawn as solid line, while a code branch produced by an input "1" is drawn as a broken line. As in the case of code-tree, each input message sequence corresponds to a specific path through the trellis. For example, we can observe from fig. 8.18 that the message sequence 10111 produces the encoded output sequence (11, 10, 00, 01, 10 with tail 01, 11), which agrees with the result obtained in example 8.8.

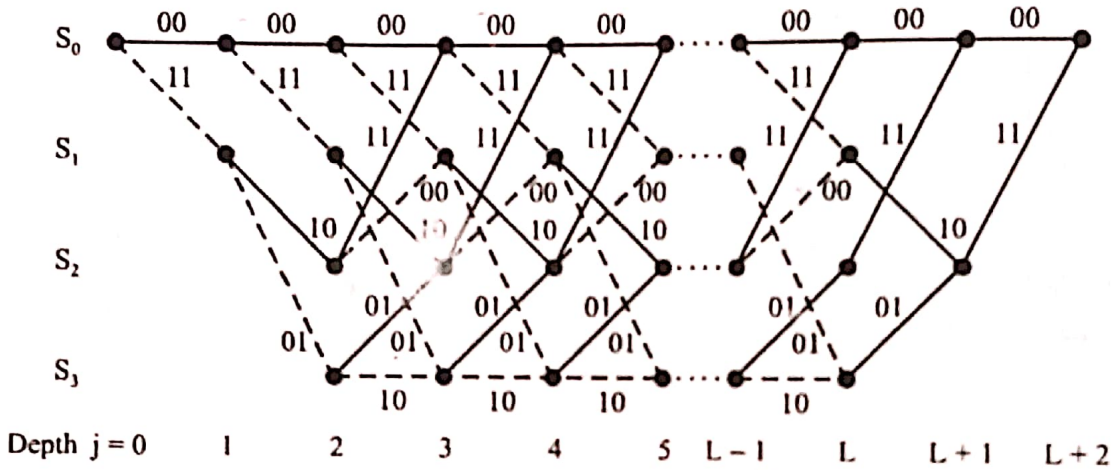


Fig. 8.18 : Trellis Diagram for the convolutional encoder of fig. 8.9

Referring to fig. 8.18, the levels of the trellis are labelled as j = 0, 1, 2, ... (L + m). The level "j" is also sometimes referred to as "***depth j***". The first "m" levels correspond to the encoder's departure from the initial state "S_0" and the last "m" levels correspond to the encoder's return to the state S_0. Clearly, not all the states can be reached in these two portions of the trellis. However, in the central portion of the trellis, for which the level j lies in the range $m \le j \le L$, all the states of the encoder are reachable. It may also be noted that the central portion of the trellis exhibits a fixed periodic structure.

MISCELLANEOUS EXAMPLES

Example 8.11 : Consider a (3, 1, 2) convolutional encoder with $g^{(1)} = (110)$, $g^{(2)} = (101)$ and $g^{(3)}(111)$

(i) Draw the encoder block diagram

(ii) Draw state table

(iii) Draw state transition table

(iv) Draw state diagram

(v) Find the encoder output by traversing through the state diagram for input message sequence of (11101)

(vi) Draw code trellis and obtain the output of the encoder for the same input sequence of (11101).

Solution:

(i) Fig. 8.19 shows the encoder block diagram.

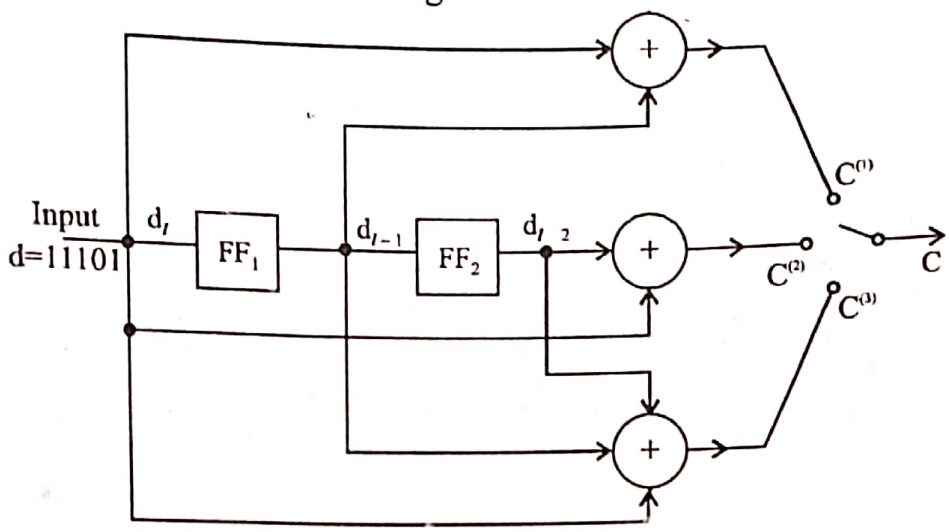


Fig. 8.19 : Convolutional encoder of example 8.11

(ii) **State table:** There are 2 flip-flops and hence there are $2^2 = 4$ states. Let these states be S_0, S_1, S_2 and S_3 as shown in table 8.6.

State	S_0	S_1	S_2	S_3
Binary Description	00	10	01	11

Table 8.6 : State table for example 8.11

(iii) **State transition table:** From fig. 8.19, the outputs of the three modulo-2 adders $C^{(1)}$, $C^{(2)}$ and $C^{(3)}$ can be represented as output sequences given by

$$C^{(1)} = d_l + d_{l-1}$$
$$C^{(2)} = d_l + d_{l-2}$$
$$C^{(3)} = d_l + d_{l-1} + d_{l-2}$$

The state transition table can be constructed as shown in table 8.7.

Present state	Binary Description	Input	Next state	Binary Description	d_l	d_{l-1}	d_{l-2}	Output $C^{(1)}$	$C^{(2)}$	$C^{(3)}$
S_0	00	0	S_0	00	0	0	0	0	0	0
		1	S_1	10	1	0	0	1	1	1
S_1	10	0	S_2	01	0	1	0	1	0	1
		1	S_3	11	1	1	0	0	1	0
S_2	01	0	S_0	00	0	0	1	0	1	1
		1	S_1	10	1	0	1	1	0	0
S_3	11	0	S_2	01	0	1	1	1	1	0
		1	S_3	11	1	1	1	0	0	1

Table 8.27 : State transition table for example 8.11

(iv) **State diagram:** Fig. 8.20 shows the state diagram consisting of four states S_0, S_1, S_3 and S_3. It also shows the state transitions based on the state transition conditions given in table 8.7 and the corresponding input and outputs in brackets.

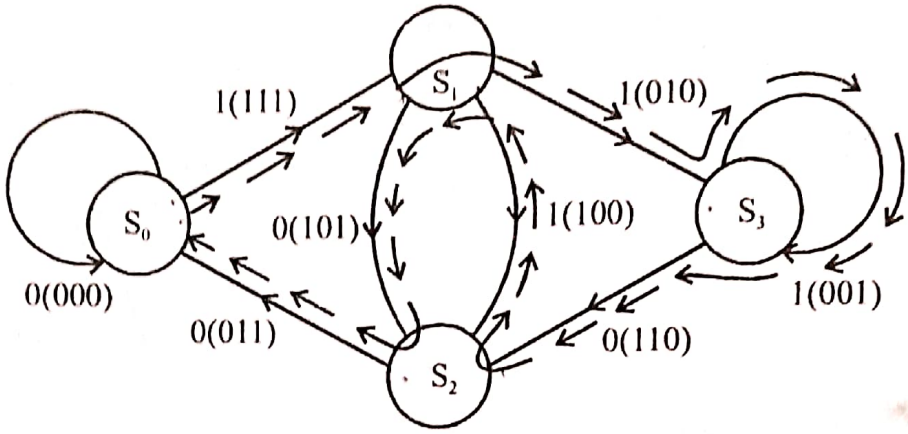


Fig. 8.20 : State diagram of example 8.11

(v) By traversing through the state diagram following the path $S_0 \rightarrow S_1 \rightarrow S_3 \rightarrow S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_2 \rightarrow S_0$ producing an output of

$$C \rightarrow [111, 010, 001, 110, 100, 101, 011]$$

(vi) **Code Trellis:** The code trellis diagram for example 8.11 is shown in fig. 8.21. The nodes on the left side denote the four possible current states and the nodes on the right denotes the next states after each transition. The solid line represents state transition from one of the left node to one of the right nodes when the date input is binary '0' and broken line when date input is binary '1'. The corresponding outputs are also shown.

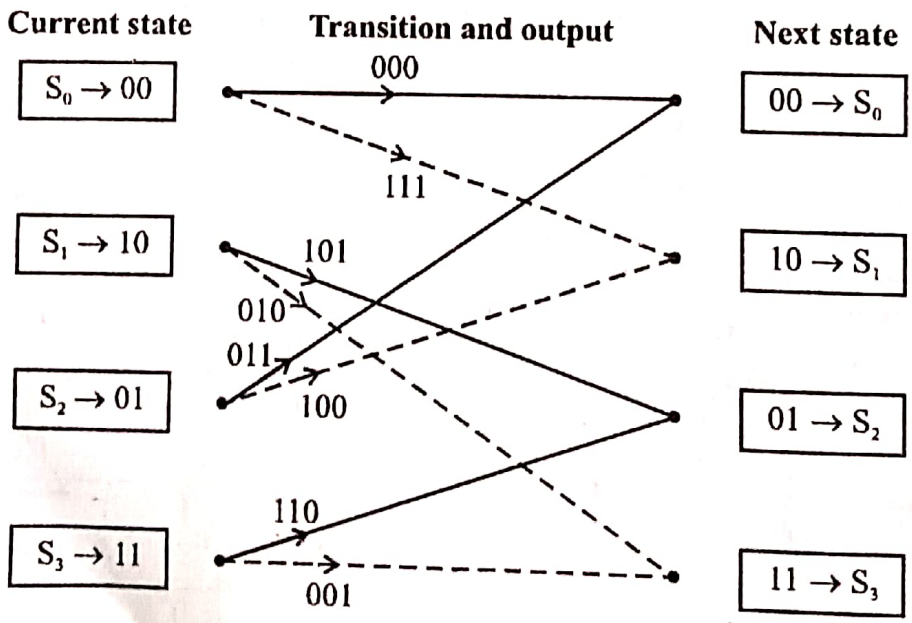


Fig. 8.21 : Code trellis for example 8.11 for single transition in a single time unit

From part (v) above, the transitions from one state to the other is given by the sequence

$$S_0 \to S_1 \to S_3 \to S_3 \to S_2 \to S_1 \to S_2 \to S_0$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$[111,\ 010,\ 001,\ 110,\ 100,\ 101,\ 011]$$

The transition from S_0 to S_1 produces an output 111, the transition from S_1 to S_3 produces 010 and so on. We observe that the output sequence so obtained is same as obtained in part (v).

Example 8.12 : For the convolutional encoder shown in fig. 8.22

(a) Draw state table

(b) Draw state transition table

(c) Draw state diagram

(d) Find encoder output for the message sequence (10011) by traversing through the state diagram.

(e) Draw code trellis for one time unit and obtain the output for the same message sequence (10011) using the core trellis.

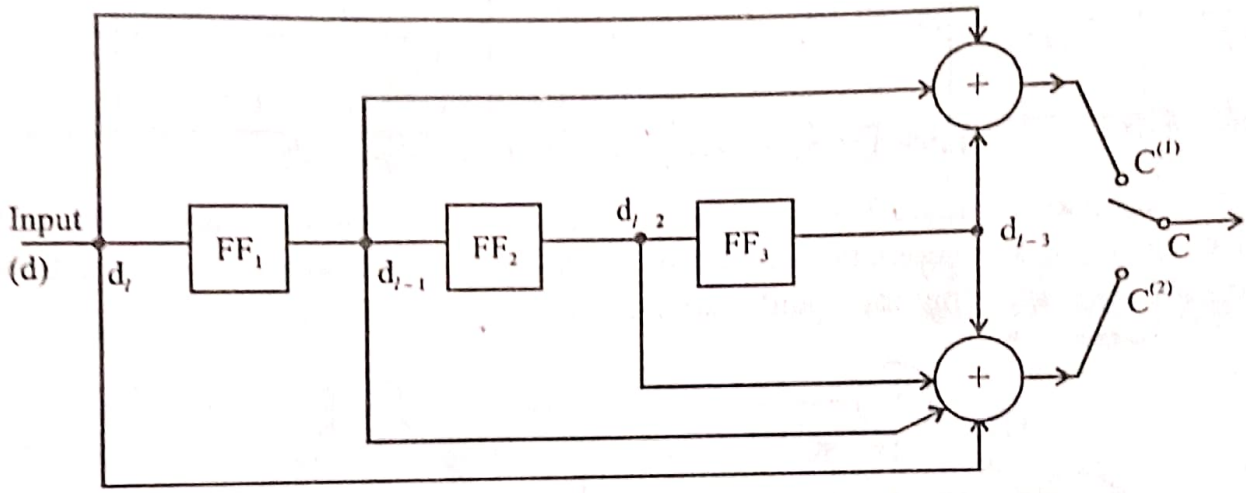


Fig. 8.22 : Convolutional encoder for example 8.12

Solution:

(a) **State table:** There are 3 flip-flops and hence there are $2^3 = 8$ states. Let the states be $S_0, S_1, \ldots\ldots S_7$. These states and their binary descriptions are as shown in table 8.8.

State	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Binary Description	000	100	010	110	001	101	011	111

Table 8.8 : State table for example 8.12

(b) **State transition table:** From fig. 8.22, the outputs of the two modulo-2 adders $C^{(1)}$ and $C^{(2)}$ can be represented as output sequences given by

$$C^{(1)} = d_l + d_{l-1} + d_{l-3}$$
$$C^{(2)} = d_l + d_{l-1} + d_{l-2} + d_{l-3}$$

The state transition table can be constructed as shown in table 8.9.

Present state	Binary Description	Input	Next state	Binary Description	d_l	d_{l-1}	d_{l-2}	d_{l-3}	Output $C^{(1)}$	Output $C^{(2)}$
S_0	000	0	S_0	000	0	0	0	0	0	0
		1	S_1	100	1	0	0	0	1	1
S_1	100	0	S_2	010	0	1	0	0	1	1
		1	S_3	110	1	1	0	0	0	0
S_2	010	0	S_4	001	0	0	1	0	0	1
		1	S_5	101	1	0	1	0	1	0
S_3	110	0	S_6	011	0	1	1	0	1	0
		1	S_7	111	1	1	1	0	0	1
S_4	001	0	S_0	000	0	0	0	1	1	1
		1	S_1	100	1	0	0	1	0	0
S_5	101	0	S_2	010	0	1	0	1	0	0
		1	S_3	110	1	1	0	1	1	1
S_6	011	0	S_4	001	0	0	1	1	1	0
		1	S_5	101	1	0	1	1	0	1
S_7	111	0	S_6	011	0	1	1	1	0	1
		1	S_7	111	1	1	1	1	1	0

Table 8.5 : State transition table for example 8.12

(c) **State diagram:** Fig. 8.23 shows the state diagram consisting of 8 states S_0, S_1,..... S_7. It also shows the state transitions based on the state transition conditions given in table 8.9 and the corresponding input and output.

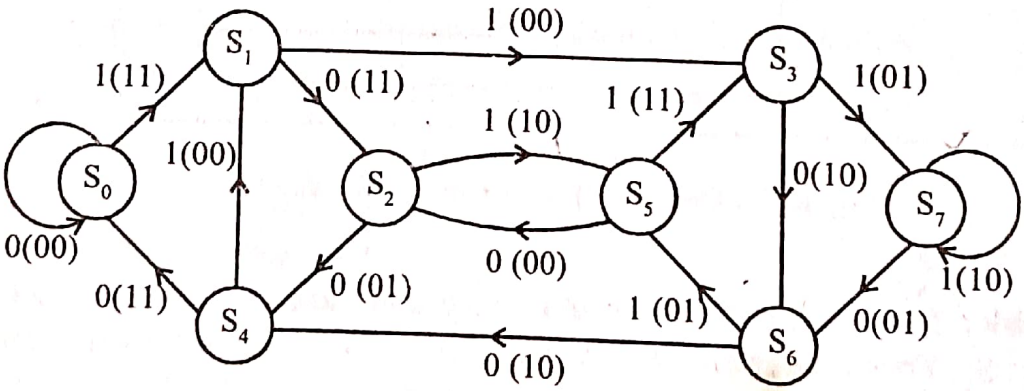


Fig. 8.23 : State diagram of example 8.12

(d) The encoder output is obtained by traversing through the state diagram of fig. 8.23 through the path $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_1 \rightarrow S_3 \rightarrow S_6 \rightarrow S_4 \rightarrow S_0$ corresponding to input sequence 10011 with additional three 0's. The output of the encoder is given by [11, 11, 01, 00, 00, 10, 10, 11].

(e) Figure 8.24 shows the code trellis consisting of eight current states and eight next states for one time unit. It also shows the transition from a left node to a right node for binary input '0' or '1'. Solid line shows transition for binary '0' input and broken line for binary '1' input. The corresponding outputs are also shown.

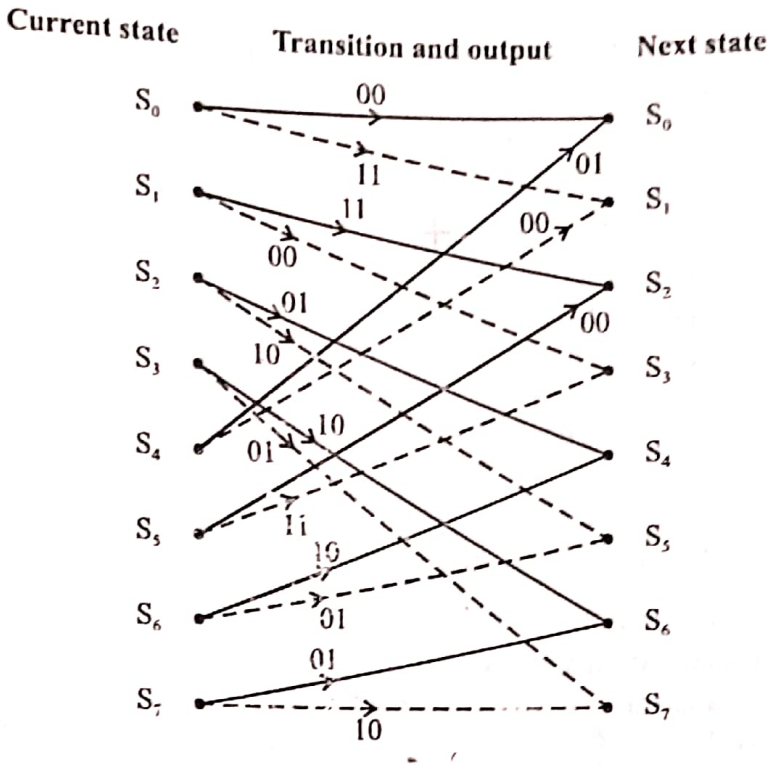


Fig. 8.24 : Code trellis for example 8.12

From fig. 8.24 above, the transition from one state to the other given by the sequence

$$\begin{array}{ccccccccccccccc} S_0 & \rightarrow & S_1 & \rightarrow & S_2 & \rightarrow & S_4 & \rightarrow & S_1 & \rightarrow & S_3 & \rightarrow & S_6 & \rightarrow & S_4 & \rightarrow & S_0 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\ [11, & & 11, & & 01, & & 00, & & 00, & & 10, & & 10, & & 01] & & \end{array}$$

is the corresponding output of the convolutional encoder and is same as obtained in part (d).

8.6 REVIEW QUESTIONS

1. What are convolutional codes? How is it different from block codes?
2. Explain encoding of convolutional codes using time-domain approach with an example.
3. Explain encoding of convolutional codes using transform-domain approach with an example.
4. What do you understand by state diagram of a convolutional encoder? Explain clearly.
5. What do you understand by code tree of a convolutional encoder? Explain clearly.
6. What do you understand by trellis diagram of a convolutional encoder? Explain clearly.

8.7 PROBLEMS

1. Consider a (3, 1, 2) convolutional code with $g^{(1)} = (011)$, $g^{(2)} = (110)$, $g^{(3)} = (101)$.
   (i) Draw the encoder block diagram
   (ii) Find the generator matrix.
   (iii) Find the code-vector corresponding to the information sequence d = 10001
   (iv) Find the transfer function matrix.
2. For the convolutional encoder given in figure P-1,
   (i) Find impulse responses and hence calculate the output produced by the information sequence 11011 using time-domain approach.

(ii) Write the generator polynomials of the encoder and recompute the output for the same sequence 11011 and compare with (i).

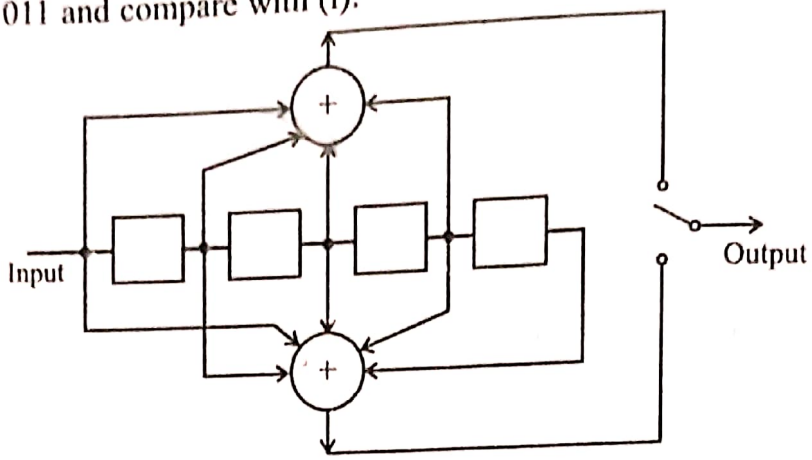


Fig. P-1 : Convolutional encoder for problem 2

3. Consider the encoder of problem 1.
   (i) Draw the state diagram of the encoder.
   (ii) Draw code-tree and tracing the path through the code-tree find the code-vector for a message of 1011.

4. For a (2, 1, 3) convolutional encoder, the generator sequences are $g^{(1)} = [1, 1, 0, 1]$ and $g^{(2)} = [0, 1, 1, 1]$

   Determine the encoder output produced by the message sequence 10011 using
   (i) Time domain approach based on discrete convolution
   (ii) Transform domain approach
   (iii) Construct state diagram and find the output for the same message sequence of 10011 by tracing the path through the state diagram.
   (iv) Construct the code-tree and find the output for the same message sequence of 10011 by tracing the path through the code tree.

5. Consider the convolutional encoder shown in figure P-5. The message bits are shifted into the encoder two bits at a time.
   (a) Find the constraint length and rate efficiency of the code.
   (b) Assume the initial contents of the registers to be zero and find the output code-vector for the input message sequence of (110101).

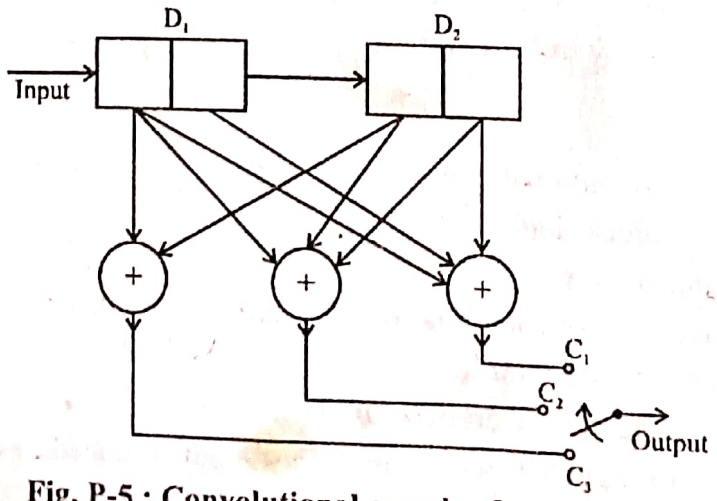


Fig. P-5 : Convolutional encoder for problem 5

ಓಂ ಶ್ರೀ ಕೃಷ್ಣಾರ್ಪಣಮಸ್ತು