Chapter 4 Convolutional Codes

4.11 Tailbiting Convolutional Codes

4.11.1 Tail-Biting Technique

4.11.2 . Encoding Procedure for Tailbiting Codes

4.11.3. Decoding Algorithms for Tailbiting Codes

4.12 Trellis Coded Modulation

4.12.1 TCM Encoder

4.12.2 Set-Partition and Distance Structure

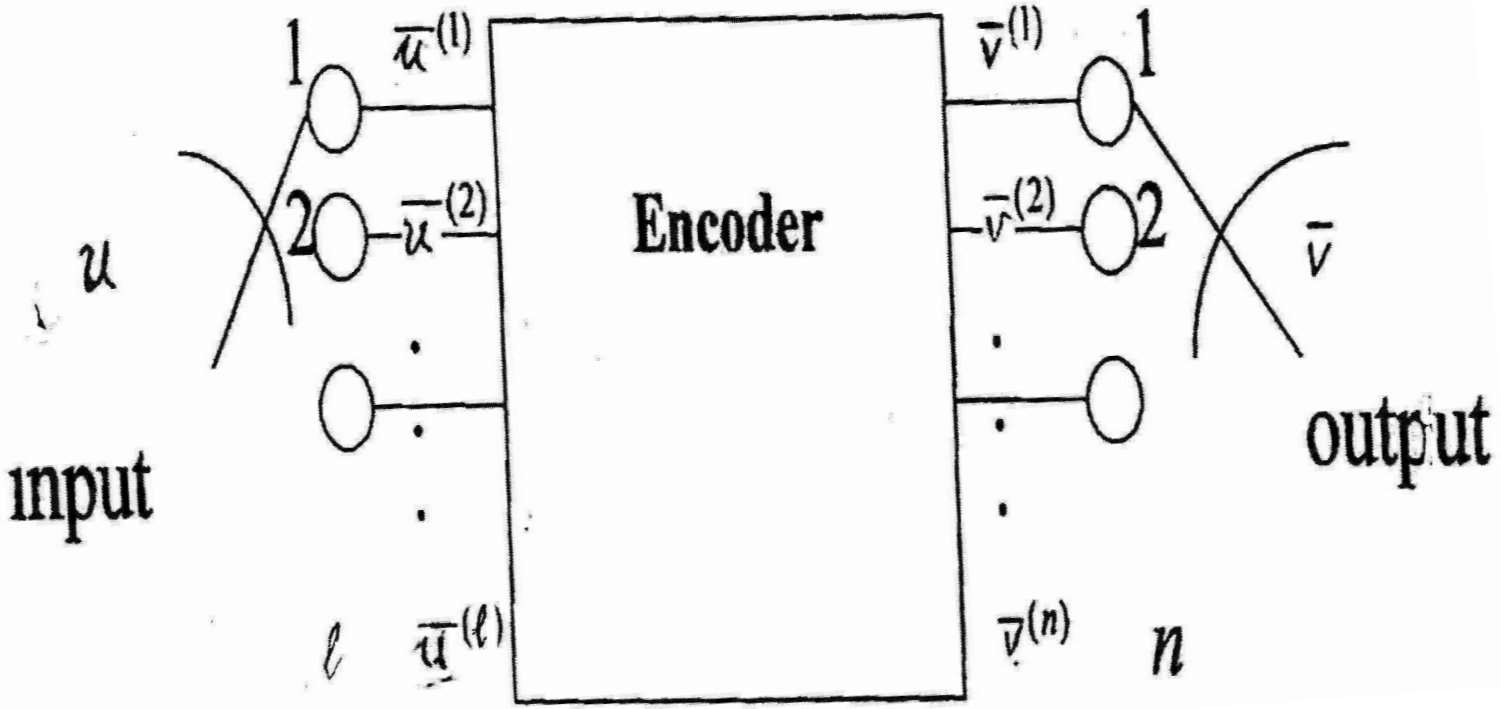
4.12.3 Decoding TCM

4.1 Introduction to Convolutional Codes

- **Convolutional codes were first discovered by P. Elias in 1955.**
- **The structure of convolutional codes is quite different from that of block codes.**

  During each unit of time, the input to a convolutional code encoder is also a *l*-bit message block and the corresponding output is also an *n*-bit coded block with $l < n$.
- **Each coded *n*-bit output block depends not only the corresponding *l*-bit input message block at the same time unit but also on the *M* previous message blocks.**
- **Thus the encoder has *l* input lines, *n* output lines and a memory of order *M* as shown in Figure 7.1 .**

Fig. 4. 1


1
2
$\bar{u}^{(1)}$
$\bar{u}^{(2)}$
$\bar{u}^{(l)}$
$\bar{u}$
input
Encoder
$\bar{v}^{(1)}$
$\bar{v}^{(2)}$
$\bar{v}^{(n)}$
1
2
$\bar{v}$
output
l
n

- **Each message (or information) sequence is encoded into a code sequence.**
- **The set of all possible code sequences produced by the encoder is called an (n,l,M) convolutional code.**

  The parameters, l and n , are normally small,

  say $1 \leq l \leq 8$ and $2 \leq n \leq 9$.

  The ratio $R = l/n$ is called the code rate.

  The parameter M is called the memory order of the code.
- **Note that the number of redundant (or parity) bits in each coded block is small.**

  However, more redundant bits are added by increasing the memory order M of the code while holding l and n fixed.

4.2 Encoder of (*n*,*l*,*M*) Convolutional Code

- For (*n*,*l*,*M*) code , the encoder has *l* inputs and *n* outputs as shown in Fig. 4.1

Time-Domain Representation

- At the *i*-th input terminal, the input message sequence is

$$\mathbf{u}^{(i)} = (u_0^{(i)}, \ldots, u_r^{(i)}, \ldots) \qquad \text{for } 1 \le i \le l \qquad (4.1)$$

- At the *j*-th output terminal, the output code sequence is

$$\mathbf{v}^{(j)} = (v_0^{(j)}, \ldots, v_r^{(j)}, \ldots) \qquad \text{for } 1 \le j \le n \qquad (4.2)$$

- **An (*n*,*k*,*M*) convolutional code is specified by $l \times n$ generator sequence:**

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_1^{(1)} & \mathbf{g}_2^{(1)} & \cdots & \mathbf{g}_l^{(1)} \\ \mathbf{g}_1^{(2)} & \mathbf{g}_2^{(2)} & \cdots & \mathbf{g}_l^{(2)} \\ \vdots & \vdots & & \vdots \\ \mathbf{g}_1^{(n)} & g_2^{(n)} & \cdots & \mathbf{g}_l^{(n)} \end{bmatrix} \qquad \textbf{(4.3)}$$

where $\mathbf{g}_i^{(j)} = [\, g_{i,0}^{(j)} \;\; g_{i1}^{(j)} \;\; g_{i2}^{(j)} \;\; \ldots]$

Define $\mathbf{G}_i^{(j)} = \begin{bmatrix} g_{i,0}^{(1)} & g_{i1}^{(1)} & g_{i2}^{(1)} & \cdots \\ g_{i,0}^{(2)} & g_{i1}^{(2)} & \cdots & \\ g_{i,0}^{(3)} & \ldots & & \end{bmatrix}$ **(4.4)**

and we obtain

$$\mathbf{v}^{(j)} = \sum_{i=1}^{l} \mathbf{u}^{(i)} * \mathbf{g}_i^{(j)} = \sum_{i=1}^{l} \mathbf{u}^{(i)} \mathbf{G}_i^{(j)} \qquad \textbf{(4.5)}$$

The *n* output code sequences are then given by

$$\mathbf{v}^{(1)} = u^{(1)} * g_1^{(1)} + u^{(2)} * g_2^{(1)} + \ldots + u^{(l)} * g_l^{(1)}$$

$$\mathbf{v}^{(2)} = u^{(1)} * g_1^{(2)} + u^{(2)} * g_2^{(2)} + \ldots + u^{(k)} * g_l^{(2)}$$

$$\vdots$$

$$\mathbf{v}^{(n)} = u^{(1)} * g_1^{(n)} + u^{(2)} * g_2^{(n)} + \ldots + u^{(k)} * g_l^{(n)}$$

$$\vdots$$

(4.6)

where $*$ denotes discrete convolution operation.

Encoder

- **The encoder of an (*n*,*l*,*M*) code consists of *l* shift-registers, each has at most *M* stages. The feedforward connections are based on the $l \times n$ generator sequences.**
- **The message bits stored in the *l* shift-registers together represent the state of the encoder.**

Example 4.1

Let n=3, l =2, and M =1. Consider the (3,2,1) convolutional code generated by the following generator sequences:

$$g_1^{(1)} = (1\ \ 1) \qquad g_1^{(2)} = (0\ \ 1) \qquad g_1^{(3)} = (1\ \ 1)$$

$$g_2^{(1)} = (0\ \ 1) \qquad g_2^{(2)} = (1\ \ 1) \qquad g_2^{(3)} = (1\ \ 0)$$

The encoder circuit is shown in Fig.4.2

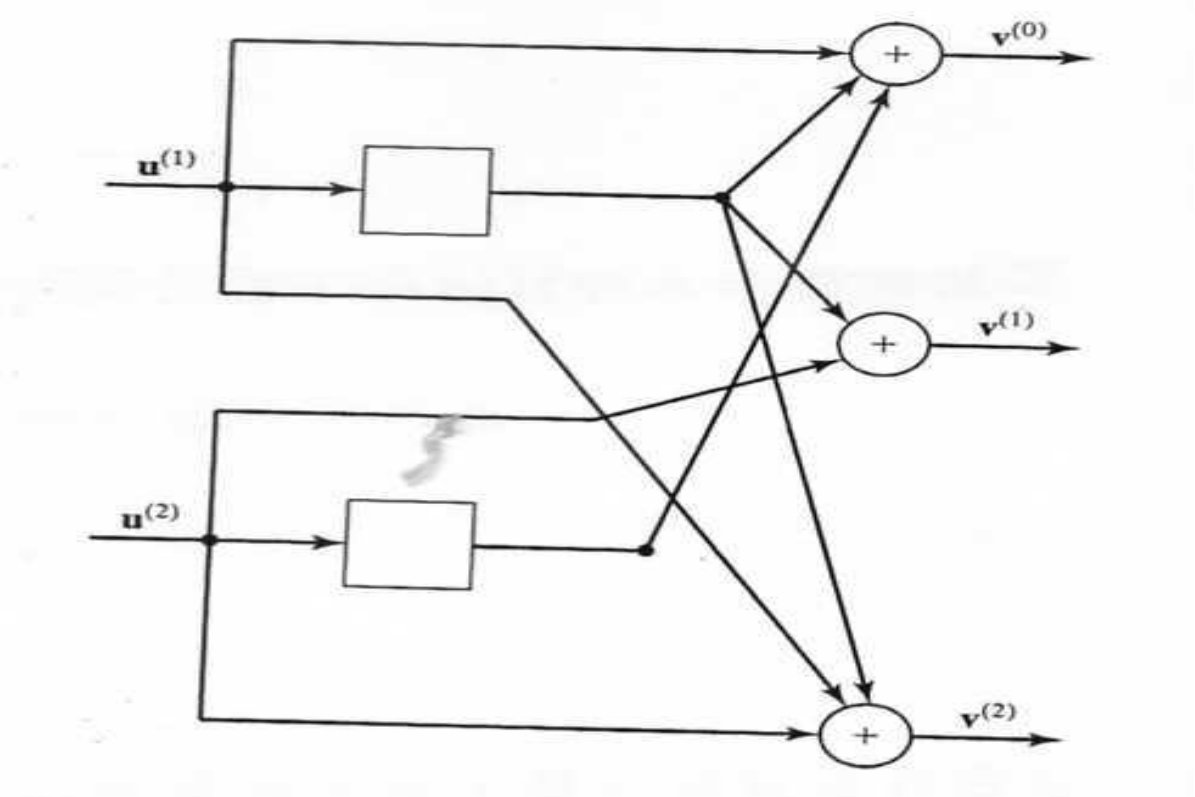
The encoder output of the j-th block are given by:

$$\mathbf{v}_k^{(1)} = u_k^{(1)} + u_{k-1}^{(1)} + u_{k-1}^{(2)}$$

$$\mathbf{v}_k^{(2)} = u_{k-1}^{(1)} + u_k^{(2)}$$

$$\mathbf{v}_k^{(3)} = u_k^{(1)} + u_k^{(2)} + u_{k-1}^{(1)} \qquad \textbf{(4.6)}$$

Fig.4.2 Encoder circuit for a (3,2,1) convolutional code



Transform –Domain Representation

- It is well known in the field of digital signal analysis the convolution operation in time-domain becomes multiplication in transform- domain, also called D- transform domain .
- The message sequence $u^{(i)} = (u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \ldots)$ can be represented in polynomial form

$$u^{(i)}(D) = u_0^{(i)} + u_1^{(i)} D + u_2^{(i)} D^2 + \ldots$$

where D represents a unit-delay .

$$u(D) = [u^{(1)}(D) \; u^{(2)}(D) \; \ldots \; u^{(l)}(D)$$

- The generator sequences can be represented by generator polynomials

$$g_i^{(j)}(D) = g_{i0}^{(j)} + g_{i1}^{(j)} D + g_{i2}^{(j)} D^2 + \ldots$$

- The encoder output

$$v(D) = [v^{(1)}(D) \; v^{(2)}(D) \; \ldots \; v^{(n)}(D)$$

In general , for a (n,k, M) convolutional code, the generator polynomials can be represented in matrix form as

$$G(D) = \begin{matrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \dots & g_1^{(n)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \dots & g_2^{(n)}(D) \\ \vdots & \vdots & & \vdots \\ g_l^{(1)}(D) & g_l^{(2)}(D) & \dots & g_l^{(n)}(D) \end{matrix} \qquad (4.7)$$

and then

$$v(D) = u(D)\, G(D) \qquad (4.8)$$

- **Example 4.2 : The (3,2,1) encoder is shown in Fig.7.2**

$$G(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

$$v(D) = \begin{pmatrix} u^{(1)} & u^{(2)} \end{pmatrix} \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

If $u^{(1)} = (101)$ $u^{(2)} = (110)$

we find $v^{(0)} = (1001)$

$v^{(1)} = (1001)$

$v^{(2)} = (0011)$

then $v = (110, 000, 001, 111)$

4.3 Systematic Form

- An (n, l , M) convolutional code is said to be systematic if the first output sequence is identical to the input message sequence , i.e.,

$$\mathbf{v}^{(1)} = \mathbf{u} \qquad (4.9)$$

Suppose the input message sequence is of finite length with L bits,

$$\mathbf{u}^{(i)} = (u_0^{(i)}, u_1^{(i)}, \ldots, u_{L-1}^{(i)}) \qquad (4.10)$$

It takes the last bit, u_{L-1} , M units of time to move out of the memory. Usually , M zeros are added to the message sequence to compute the last M output blocks and also clear the shift register . Thus ,each output sequence consists of $L+M$ bits,

$$\mathbf{v}^{(j)} = (v_0^{(j)}, v_1^{(j)}, \ldots, v_r^{(j)}, \ldots, v_{L+M-1}^{(j)})$$

for $j = 1, 2, \ldots, n$. (4.11)

The parameter $K = M + 1$ is called the constraint length of the code.

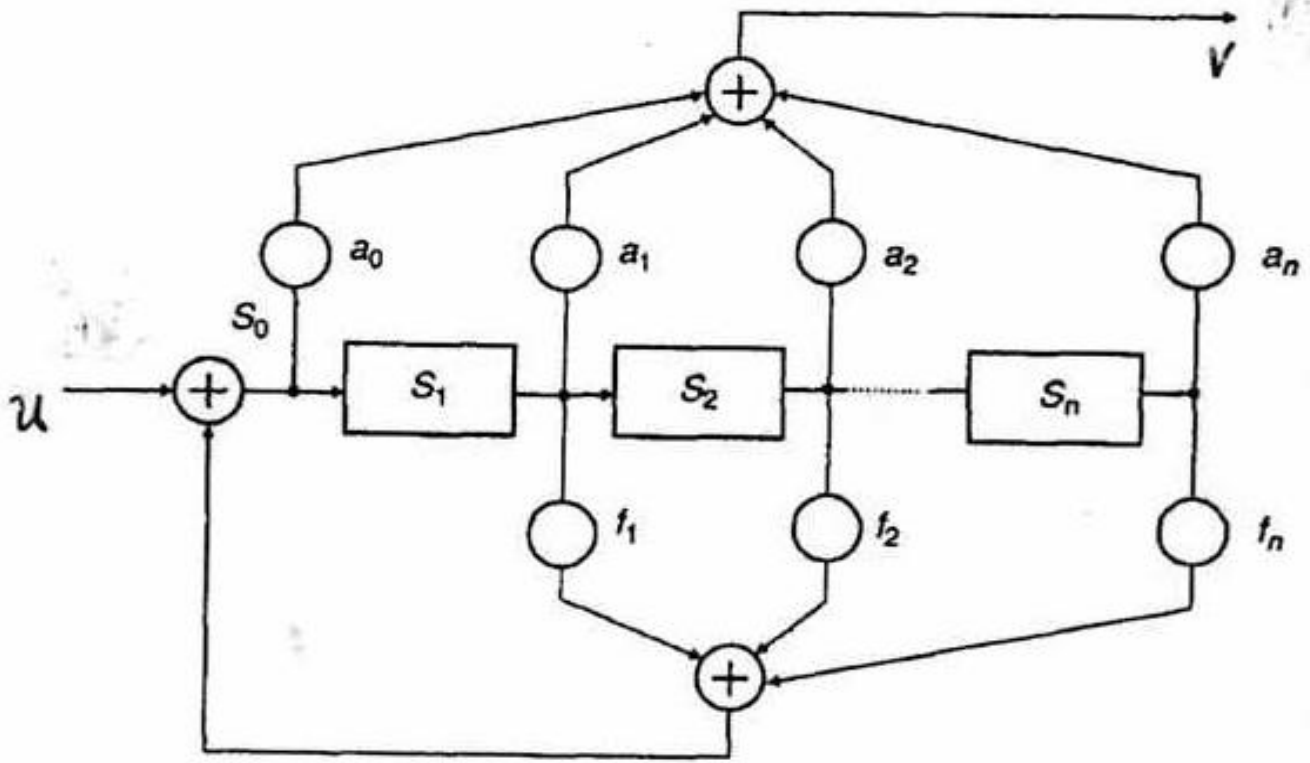
4.4 Non-recursive and Recursive Codes

- **Encoders for convolutional codes can be implemented by finite state sequential machines FSSMs) which fall into two general categories : recursive and non-recursive .**
- **Finite state sequential machines can be constructed by using basic memory units (such as shift registers), combined with adders and scalar multipliers.**
- **The following figure (Fig.4.3) shows the functional diagram of a general finite state sequential machine , expressed by the transfer function**

$$G(D) = (a_0 + a_1 D + a_2 D^2 + \ldots + a_n D^n) / (1 + f_1 D + f_2 D^2 + \ldots + f_n D^n)$$

Fig.4.4

$$G(D) = \frac{C(D)}{M(D)} = \frac{a_0 + a_1 D + a_2 D^2 + \cdots + a_n D^n}{1 + f_1 D + f_2 D^2 + \cdots + f_n D^n}$$

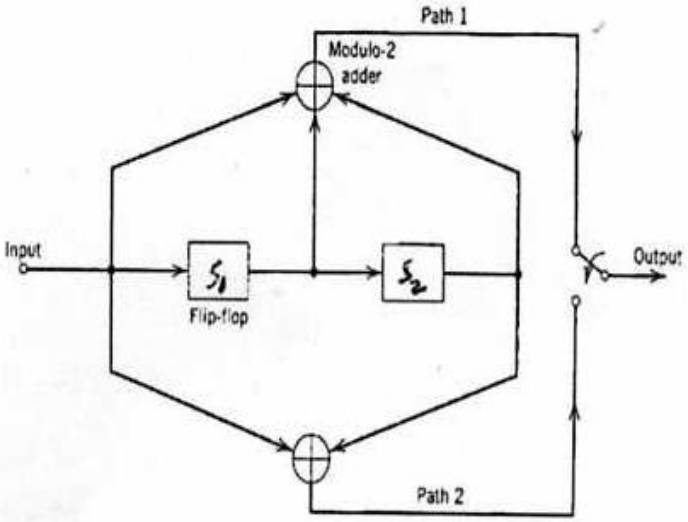


- **Fig.4.4(a) shows the structure of a non-recursive (2,1,2) convolutional code with generator matrix**

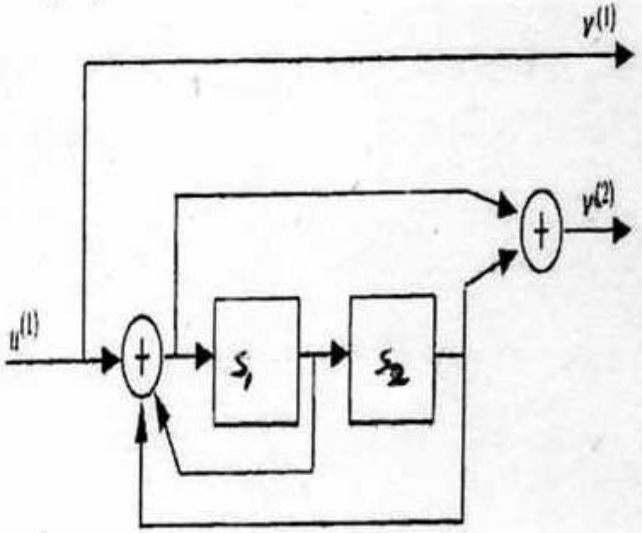
  $G(D)=[\ 1+D+D^2\ ,\ 1+D^2\ ]$, while Fig.4.4(b) shows the structure of the recursive convolutional code with generator matrix $G(D) = [\ 1\ \ (1+D^2\ ) / 1+D+D^2)\ ]$

Fig.4.4

(a)



(b)



4.5 State Diagram

- **Since the encoder is a linear sequential circuit, its behavior can be described by a state diagram.**

  Define the encoder state at the time k as the M-tuple,

  $$S_k = (s_{k1}, s_{k2}, \ldots, s_{kM})$$

  which consists of the M message bits stored in the shift register.

- **There are 2^M possible states. At any time instant, the encoder must be in one of these states.**
- **The encoder undergoes a state transition when a message bit u_k is shifted into the encoder register .**
- **The encoder state changes from S_k to a new state denoted as S_{k+1} .**

- **At each time unit, the output block depends on the input and the state,**

$$V_k = f(u_k, S_k) \qquad (4.12)$$

- **State Diagram (Pictorial Representation)**

Each state is represented by a vertex (or point) on a plane.

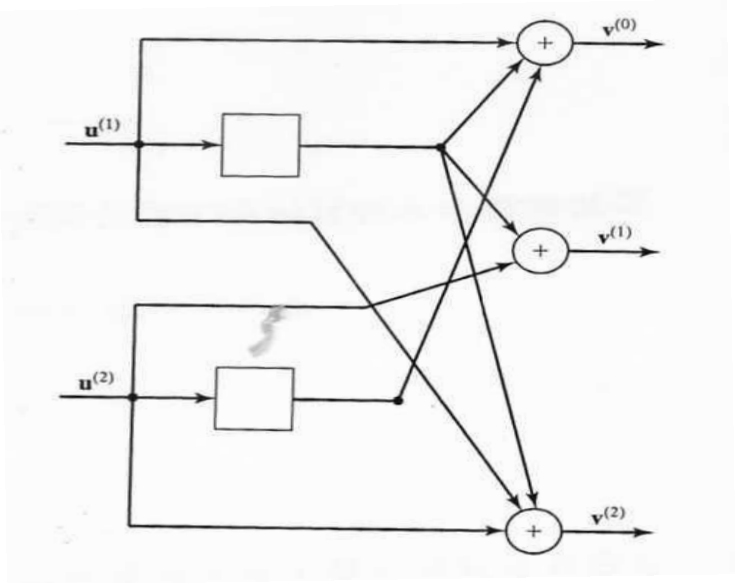
The transition from one state to another state is represented by a directed line (arc).

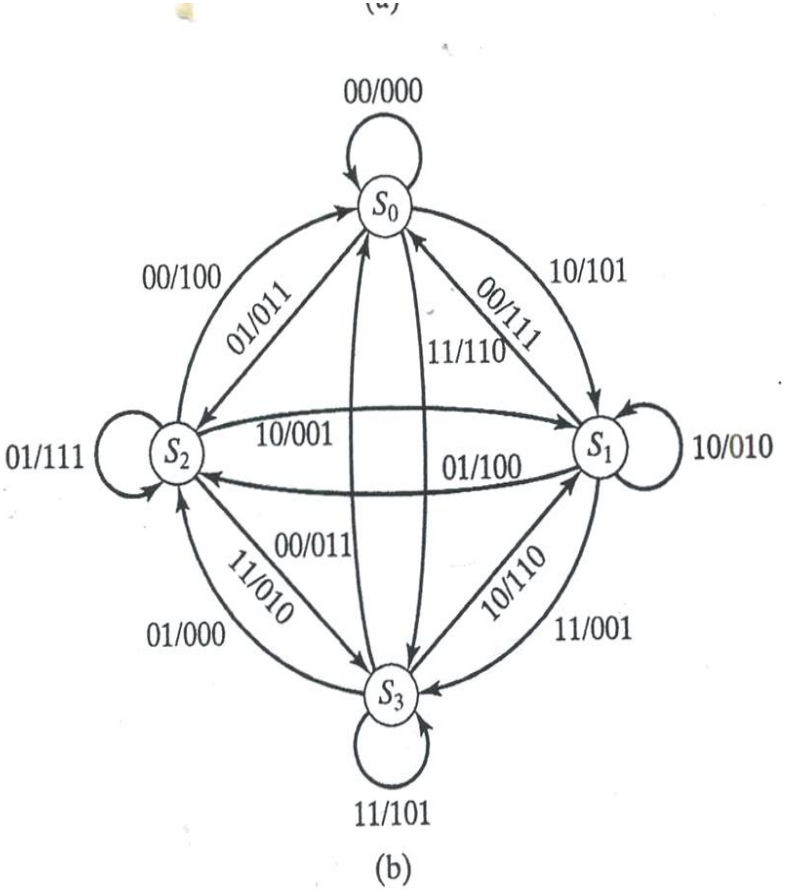
Each directed line is labeled with I/O (input/output) pair.

Example :

(a) (3,2,1) code

State $S =$ $(u_{k-1}^{(i)}, u_{k-1}^{(2)})$

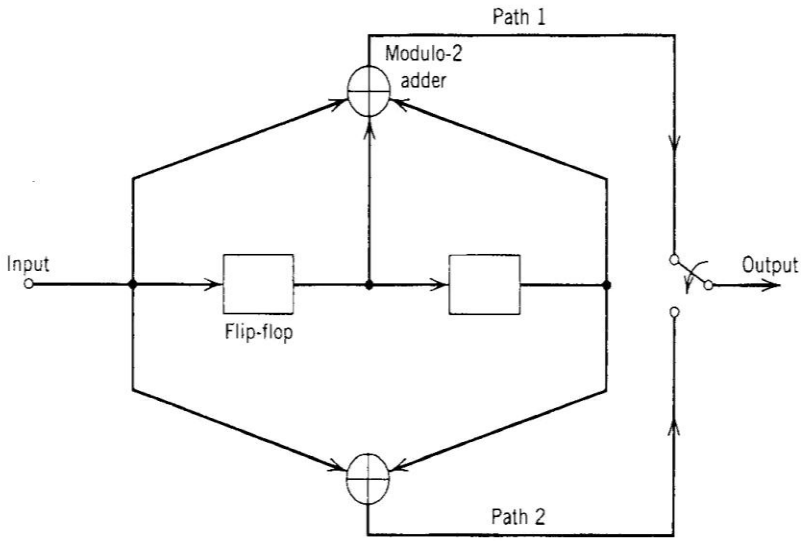


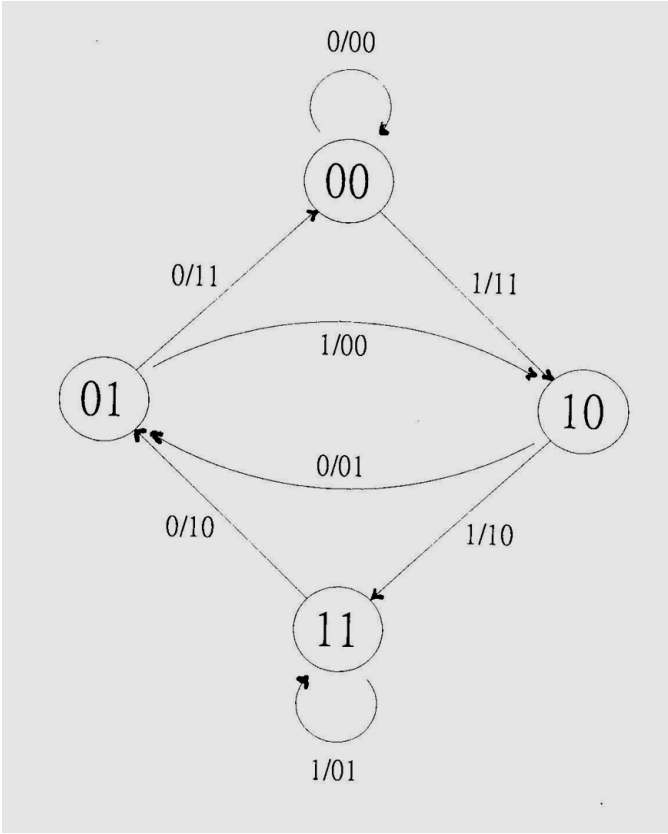


(b) State diagram of the (2,1,2) convolutional coder (Fig. 4.4(a)) is shown in Fig.4.5 . State S = (u_{k-1} u_{k-2})

Each state is one of the forms: (0 0), (0 1), (1 1), and (1 0).

Fig.4.5





4.6 Trellis Diagram

- **The state diagram can be expanded in time to display the state transition of a convolutional encoder in time. This expansion in time results in a trellis diagram.**

4.6.1 Construction of Trellis

For (n ,l ,M) code , normally the encoder starts from the all-zero state,

(0, 0, …, 0).

Every time, when a l-bit message is shifted into the encoder register, the number of state increases 2^l times until the number of states reaches 2^M.

At the time $k=M$, the encoder reaches the steady state.

Thus, for $k > M$, there are 2^l branches merging into a state in the trellis diagram at time k .

Example 4.4

(a) The (2,1,2) nonrecursive convolutional code with generator polynomial given by

$$g^1(D) = 1 + D + D^2 \quad , \quad g^2(D) = 1 + D^2$$

Its encoder and trellis diagram are shown in Figure 4.6

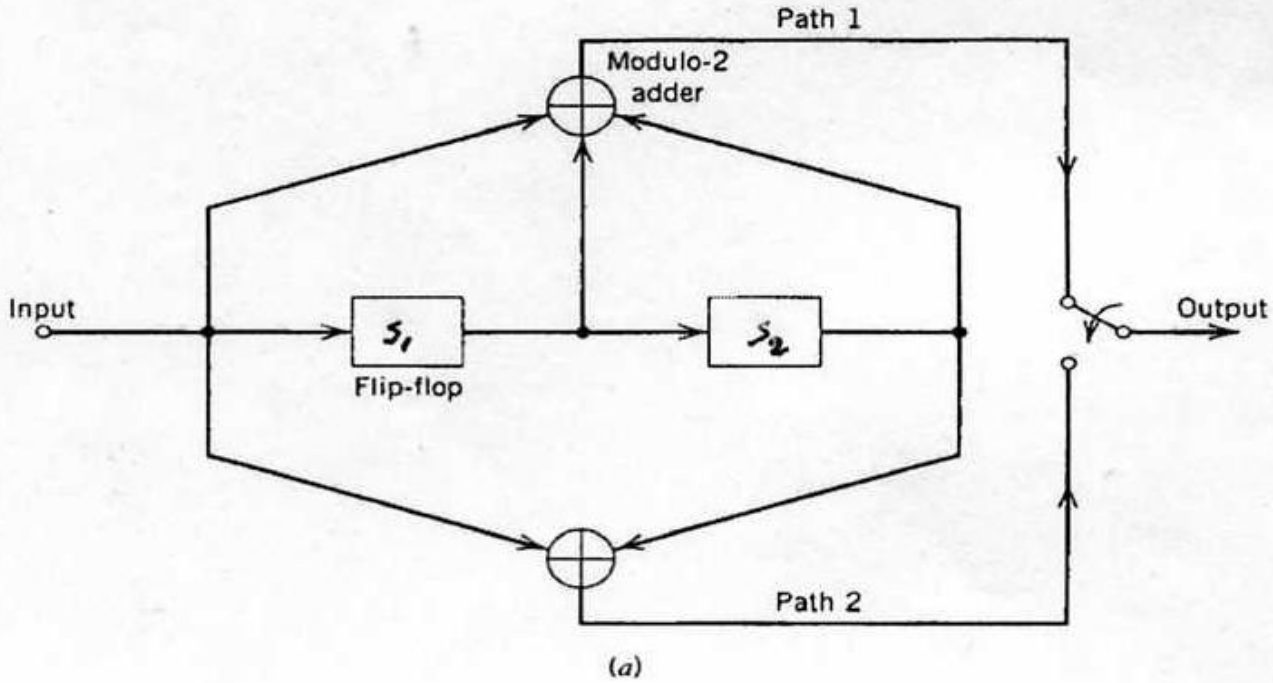
- **We see that there are two branches leaving each state, depending on the input symbol, $u_k = 0$ or $u_k = 1$.**
- **The upper branch corresponds to an input symbol $m_k = 1$, while the lower branch corresponds to an input symbol**

$$u_k = 0.$$

- **For $k > M = 2$, we see that there are two branches merging into a state.**
- **The encoding of a message sequence is equivalent to tracing a path through the trellis.**

Fig.4.6 (a) (2,1,2) convolutional code encoder

$$G(D) = [\, 1 + D + D^2 \quad 1 + D^2 \,]$$



(a)

4.5 State Diagram

- **Since the encoder is a linear sequential circuit, its behavior can be described by a state diagram.**

  Define the encoder state at the time k as the M-tuple,

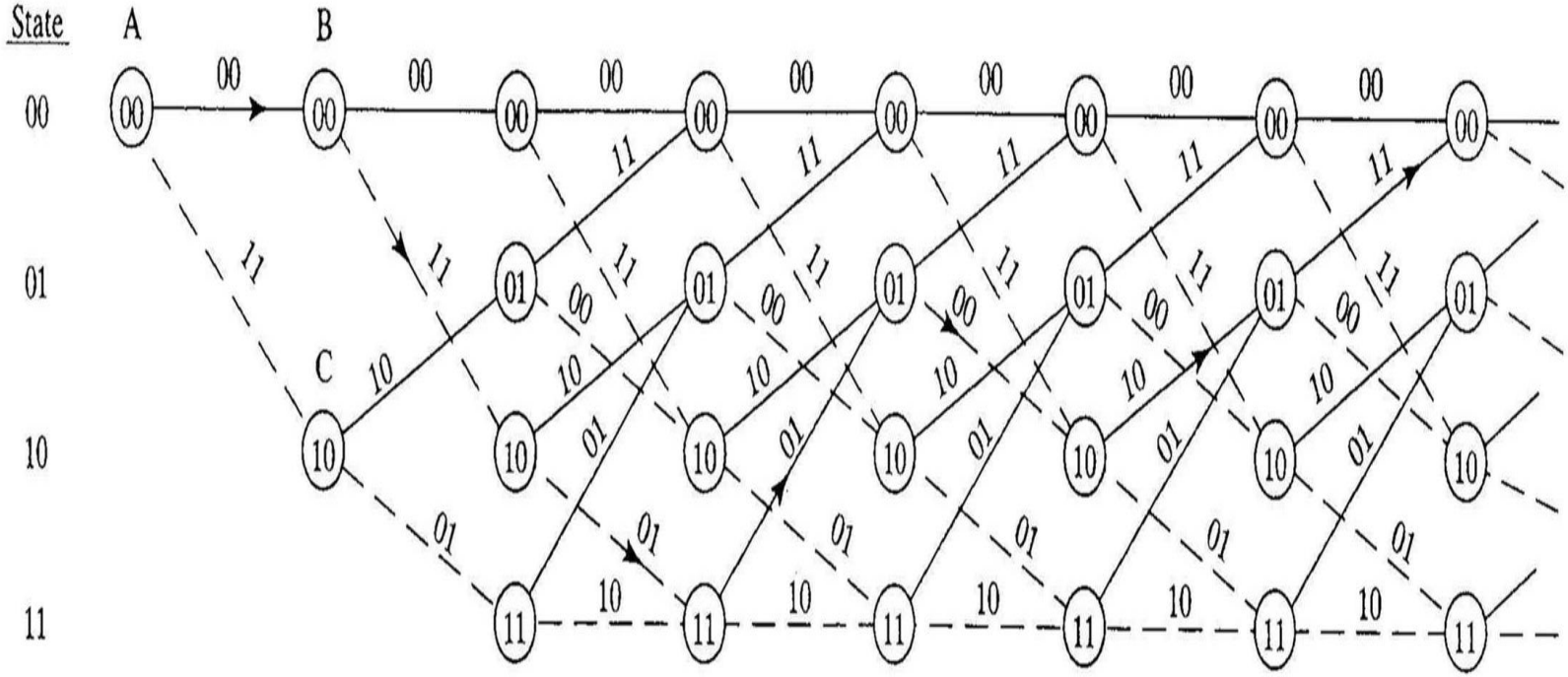
  $$S_k = (s_{k1}, s_{k2}, \ldots, s_{kM})$$

  which consists of the M message bits stored in the shift register.

- **There are 2^M possible states. At any time instant, the encoder must be in one of these states.**
- **The encoder undergoes a state transition when a message bit u_l is shifted into the encoder register .**
- **The encoder state changes to a new state denoted as S_l**

Fig.4.6(b) Trellis Diagram for (2,1,2) code

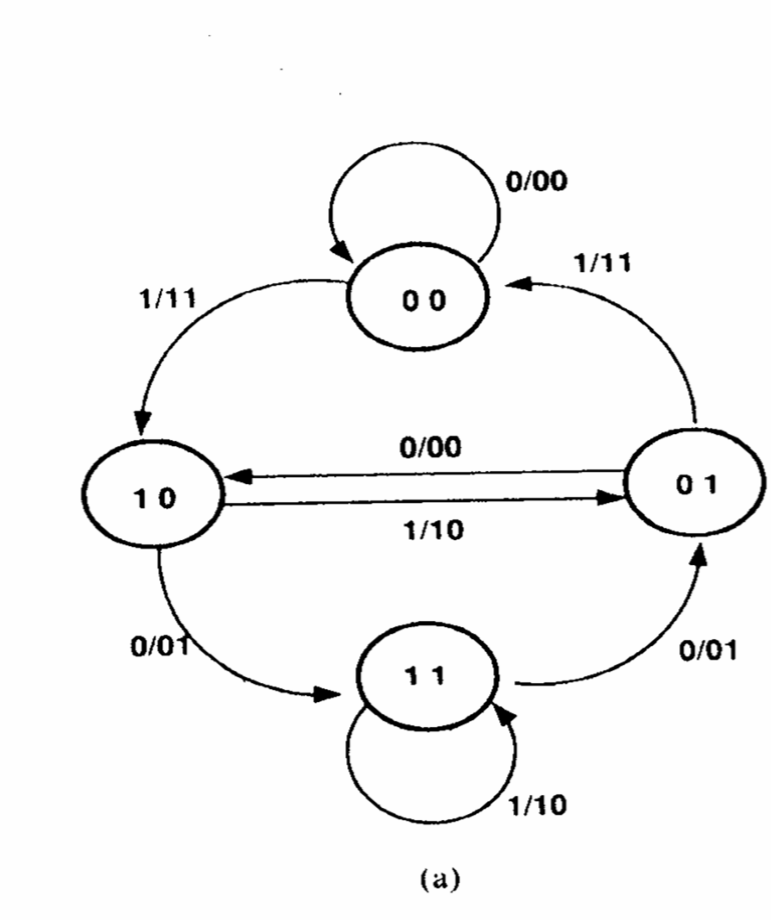
Note that a dashed line is for an for input 1 and a solid line is for an input 0

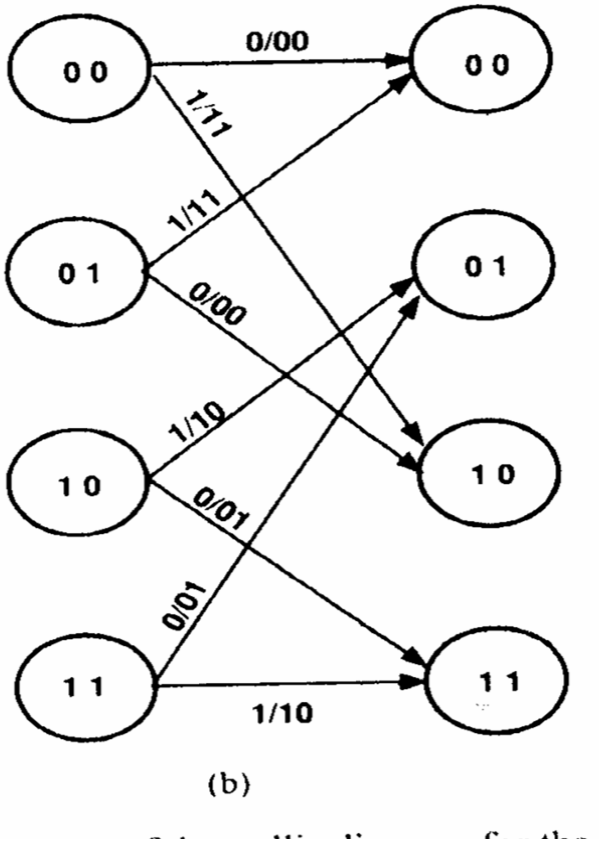


Information sequence	0	1	1	0	1	0	0
Codeword sequence	00	11	01	01	00	10	11

(b) RSC

$$G(D) = [\ 1 \quad (1+D^2)\ /\ 1+D+D^2)\ ]$$



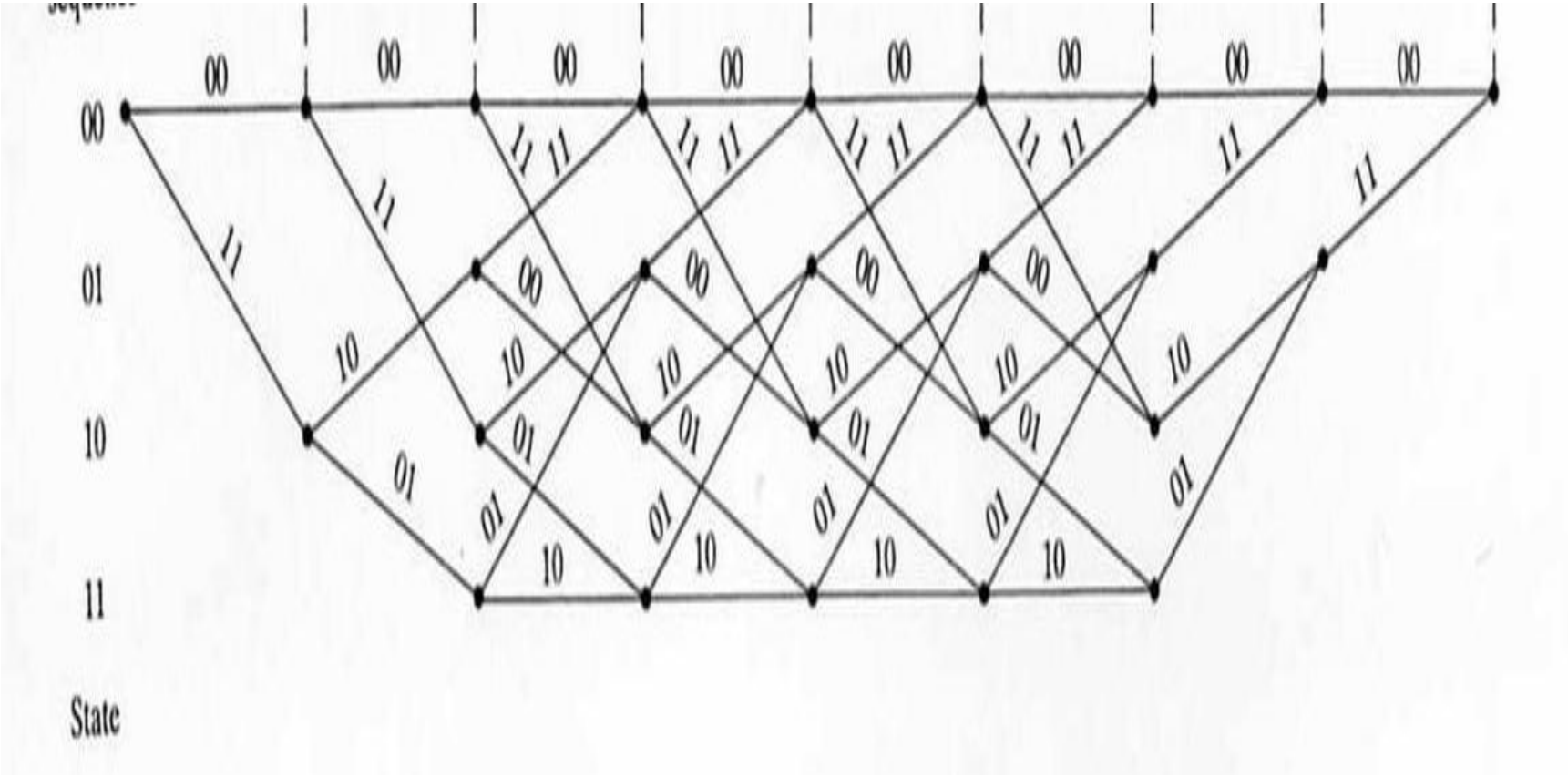


4.6.2 Termination of a Trellis

- Suppose the message sequence is of L bits long,

  $\mathbf{u} = (u_0, u_2, \ldots, u_{L-1})$

- When the entire sequence has been encoded, the encoder must return to the starting state. This can be done by appending M zeros to the message sequence u .
- When the M-th "0" is shifted into the register, the encoder is back to the all-zero state, (0, 0, …, 0).
- At this instant, the trellis converges into a single vertex.
- During the termination process, the number of states is reduced by half as each "0" is shifted into the encoder register.


00
01
10
11
State
00 00 00 00 00 00 00 00
11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11
11 00 00 00 00 00 00 10
10 10 10 10 10 10 10 01
01 01 01 01 01 01 01 10
10 10 10 10 10 10 10

4.7 Minimum Free Distance

- **The most important distance measure for convolutional codes is the minimum free distance, denoted d_{free}.**
- **The minimum free distance of a convolutional code is simply the minimum Hamming distance between any two code sequences in the code.**
- **It is also the minimum weight of all the code sequences, which are produced by the nonzero message sequences.**
- **The minimum free distance of the (2,1,2) convolutional code given in Example 4.4 is 5 , i.e., $d_{free} = 5$.**

The Most Widely Used Convolutional Codes

- **The most widely used convolutional code is (2,1,6) Odenwalter code generate by the following generator sequence,**

$$g^{(1)}(D) = 1 + D + D^3 + D^4 + D^6$$

$$g^{2}(D) = 1 + D^3 + D^4 + D^5 + D^6$$

This code has d_{free}=10.

- **With hard-decision decoding, it provides a 3.98 dB coding gain over the uncoded BPSK modulation system.**

With soft-decision decoding, the coding gain is 6.98dB.

Remarks :

Note that good convolutional codes have as large a free distance as possible; at high signal-to- noise ratios these codes are optimum. Some non-systematic convolutional codes have a superior distance structure. Therefore non-systematic codes are sometimes preferred over systematic codes .

Summary :

- **A (*n.l,M*) convolutional code can be represented by :**
  1. **Encoder block diagram using digital circuits (shift registers , adders , etc.)**
  2. **Generator polynomials, $g^{(i)}(D)$.**
  3. **State diagram**
  4. **Trellis diagram**

  Note : constraint length K = *M+1*

  number of states = 2^M

4.8. Maximum Likelihood Decoding of Convolutional Codes

4.8.1 Maximum Likelihood Decoding

- For a convolutional code, each code sequence is a path in the trellis diagram of the code.

- Suppose each message sequence consists of L message blocks of k bits each, $\mathbf{u} = (u_0, u_2, \ldots, u_{L-1})$

  Then each code sequence c is a path of $L+M$ branches long in the trellis diagram.

- Suppose a code sequence is transmitted , $\mathbf{v} = (v_0, v_1, \ldots, v_{L+M-1})$

  where the j-th branch (or code block) $v_j = (v_j^{(1)}, v_l^{(2)}, \ldots, v_l^{(n)})$

- Let $\mathbf{r} = (r_0, r_1, \ldots, r_{L+M-1})$

  be the received sequence where $\mathbf{r}_j$ *is* the j-th received block. .

- **MLD: Find the path through the trellis diagram such that the conditional probability, $P(\mathbf{r} \mid \mathbf{v})$ is the largest.**
- **For a binary input, Q-ary output discrete memoryless channel (DMC), v is a binary sequence and r is a Q-ary sequence.**
- **The conditional probability $P(\mathbf{r} \mid \mathbf{v})$ can be computed as follows:**

$$P(\mathbf{r} \mid \mathbf{v}) = \prod_{k=0}^{L+m-1} P(r_k \mid v_k) \qquad (4.15)$$

where $P(r_k \mid v_k)$ is the branch conditional probability.

- **The branch conditional probability is given by**

$$P(r_k \mid v_k) = \prod_{i=0}^{n-1} P(r_k^{(i)} \mid v_k^{(i)}) \qquad (4.16)$$

where $P(r_k^{(i)} \mid v_k^{(i)})$ is the channel transition probability.

- **Define the log-likelihood function of a path v as follows:**

$$M(\mathbf{r} \mid \mathbf{v}) = \log P(\mathbf{r} \mid \mathbf{v}) \qquad (4.17)$$

which is called the metric of path v .

- From (4.15) and (4.17), we have

$$M(\mathbf{r} \mid \mathbf{v}) = \sum_{k=0}^{L+m-1} \log P(r_k \mid v_k) = \sum_{k=0}^{L+m-1} M(r_k \mid v_k) \quad (4.18)$$

where $M(r_k \mid v_k) = \log P(r_k \mid v_k)$ (4.19)

is called the **branch metric**.

- From (4.16) and (4.19), we have the branch metric

$$M(r_k \mid v_k) = \sum_{i=0}^{n-1} \log P(r_k^{(i)} \mid v_k^{(i)}) \quad (4.20)$$

where $M(r_k^{(i)} \mid v_k^{(i)}) = \log P(r_k^{(i)} \mid v_k^{(i)})$ (4.21)

is called the **bit metric**.

- MLD: Find the path $\mathbf{v}$ in the trellis diagram such that $M(\mathbf{r} \mid \mathbf{v})$ is maximized. Then $\mathbf{v}$ is the estimate of the transmitted code sequence, $\bar{C}$.
- For the first j branches of a path through the trellis, the partial path metric is

$$M([r \mid v]_j) = \sum_{k=0}^{j-1} M(r_k \mid v_k) \quad (4.22)$$

4.8.2 ML Decoding for Binary Symmetric Channel

- For a BSC (Q=2) with transition probability $p < \frac{1}{2}$, the log-likelihood function becomes

$$\log P(\mathbf{r} \mid \mathbf{v}) = d(\mathbf{r},\mathbf{v}) \log [p/(1-p)] + (L+m)\, n \log (1-p) \qquad (4.23)$$

where $d(\mathbf{r},\mathbf{v})$ is the Hamming distance between r and v .

Since $\log [p/(1-p)] < 0$ and $(L+m)\, n \log (1-p)$ is a constant for all code sequences , $\log P(\mathbf{r} \mid \mathbf{v})$ is maximized if and only if $d(\mathbf{r},\mathbf{v})$ is minimized.

- MLD: The received sequence r is decoded into the code sequence v if $d(\mathbf{r},\mathbf{v})$ is minimized.

4.8.3 The Viterbi Decoding Algorithm

- **The Viterbi algorithm performs maximum likelihood decoding but reduces the computational complexity by taking advantage of the special structure of the code trellis.**
- **It was first introduced by A. Viterbi in 1967 and was first recognized by D. Forney in 1973 that it is a MLD algorithm for convolutional code.**
- **The Viterbi algoritm, when applied to the received sequence r from a discrete memoryless channel (DMC)**

finds the path through the trellis with the largest metric. The algorithm processes r in a recursive manner. At each time unit , it adds 2^l branches metrics to each previously stored path metric, it compares metric of all 2^l paths entering each state , and it selects the path with the largest metric, called the survivor. The survivor atb each state is then stored along with its metric.

The Viterbi Algorithm

- **Step 1. Starting at the level $k = M$ in the trellis, compute the partial metric for the single path entering each m-th order node. Store the path (the survivor) and its metric for each node.**
- **Step 2.**

  Increasing k by 1.

  Compute the partial metric for all the paths entering a (k+1)-th order node by adding the branch metric entering that node to the metric of the connecting survivor at a previous k-th order node.

  For each (k+1)-th node, store the path with the largest metric (the survivor) , together with its metric, and eliminate all the other paths.
- **Step 3. If $k < L+M$, repeat Step2. Otherwise, stop.**

Example 4.5

- **Consider the (2,1,2) convolutional code given in Example xx whose trellis diagram is shown in Fig.4.7.**

  Suppose the code is used for a BSC. In this case, we may use the Hamming distance as the path metric. The survivor at each node is the path with the smallest Hamming distance from the received sequence.

- **The message length L = 6.**

  received sequence : 10 10 10 11 11 10 01 11

  (answer : decoded sequence : 11 10 11 11 01 10 01 11)

Fig.4.7 (a) Truncated trellis diagram

$G(D) = (1+D+D^2 \quad 1+D^2)$

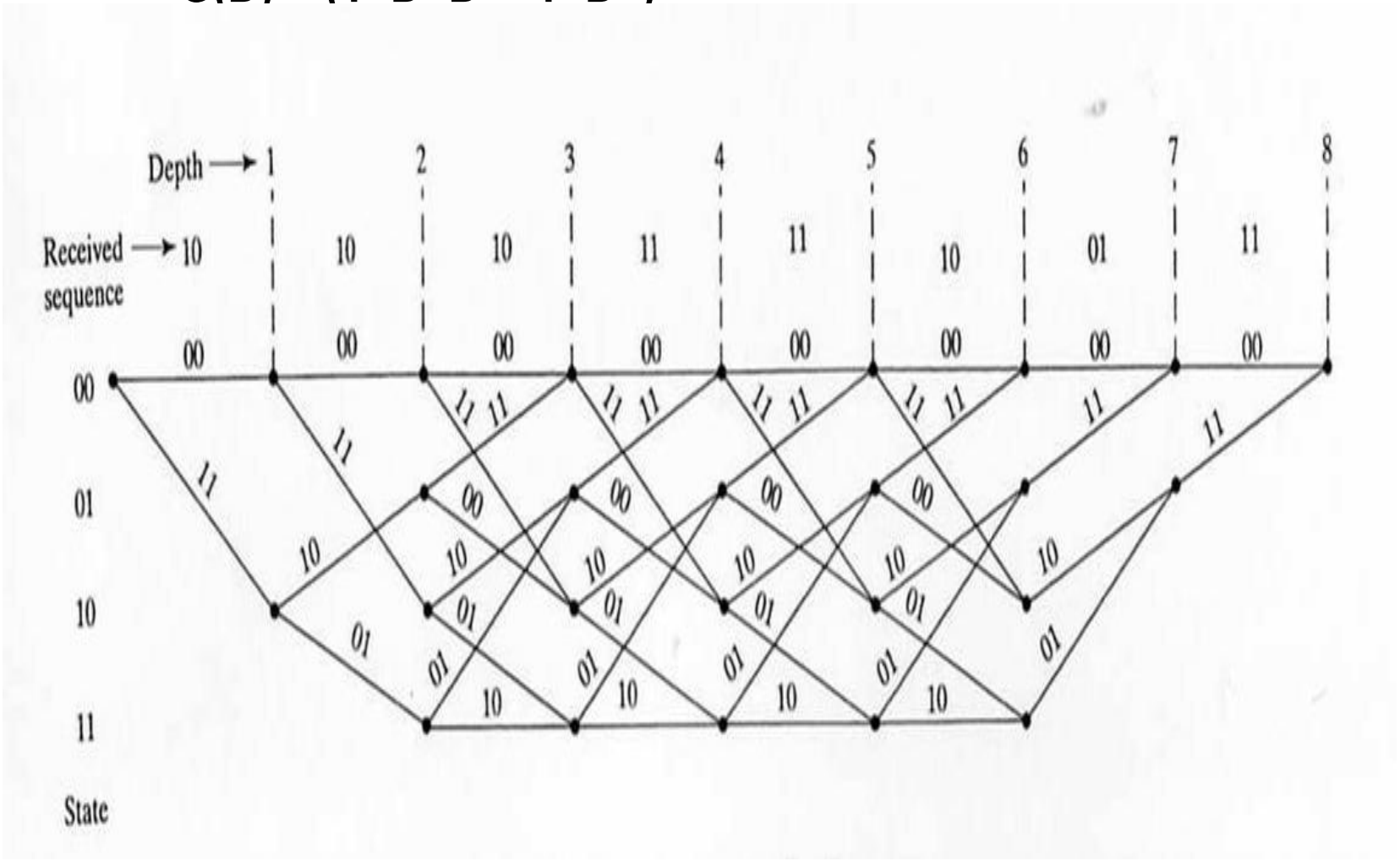


Fig.4.7 (b)

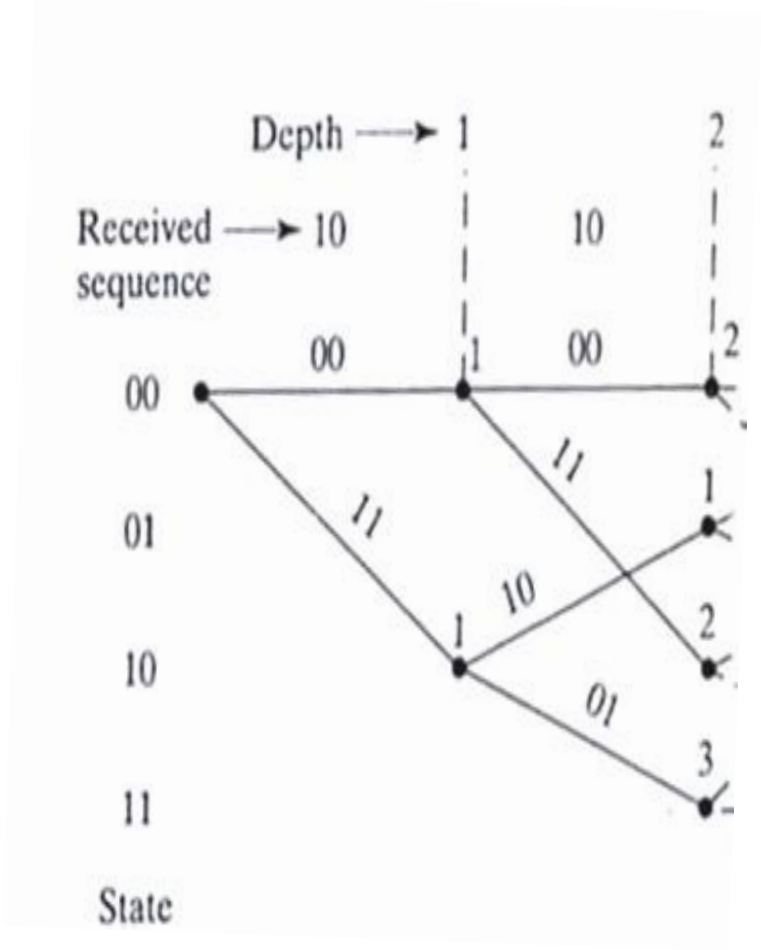


Fig.4.7 (c)

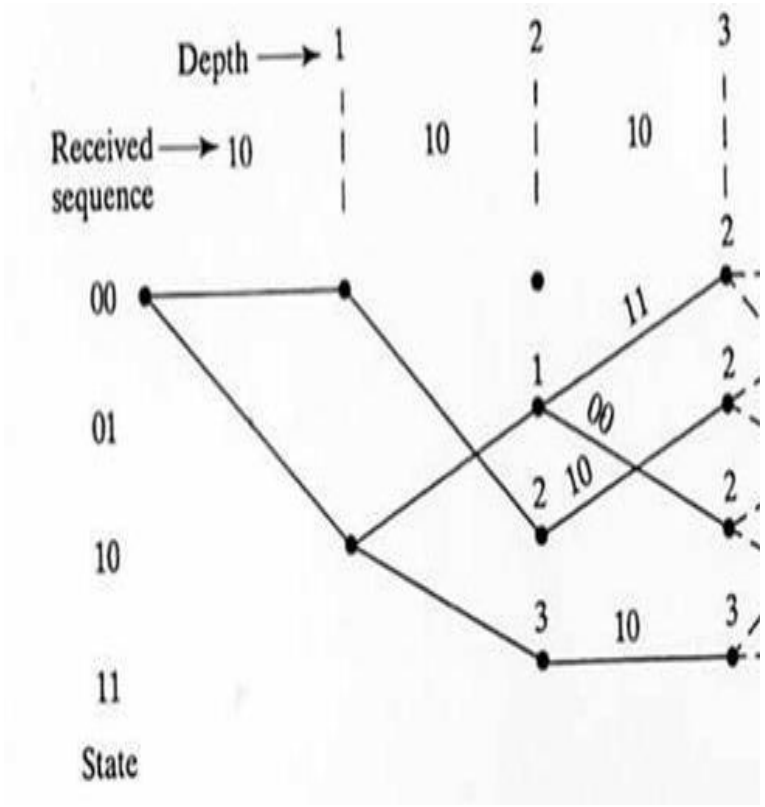


Fig.4.7(d)

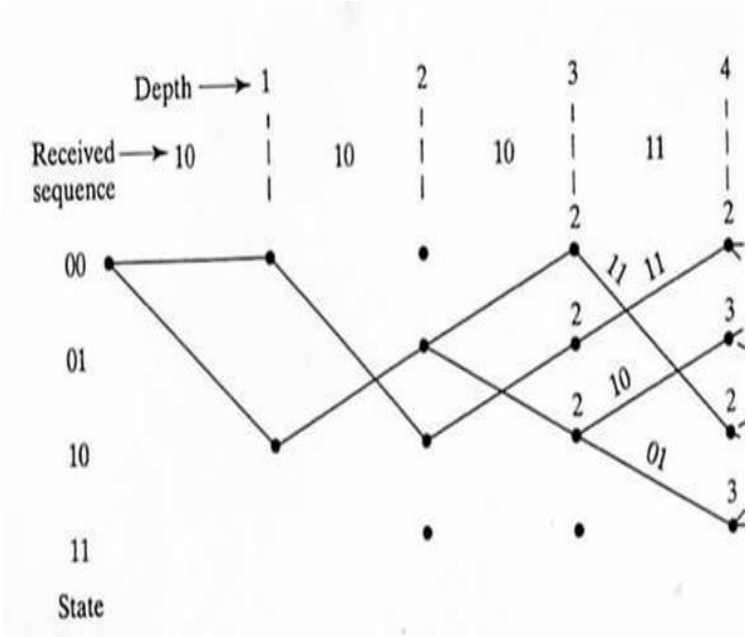


Fig.4.7(e)

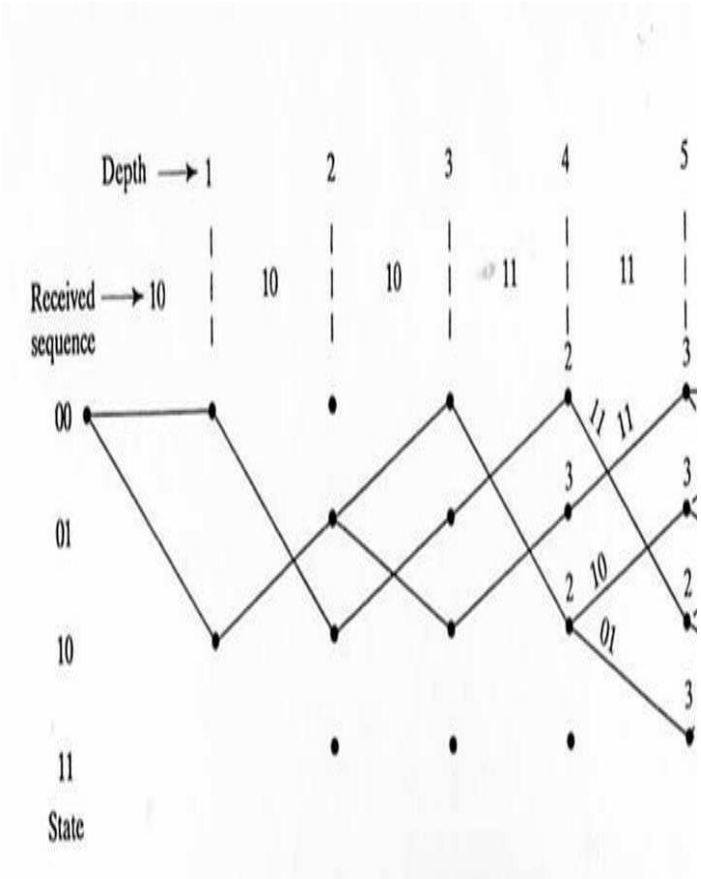


Fig. 4.7(f)

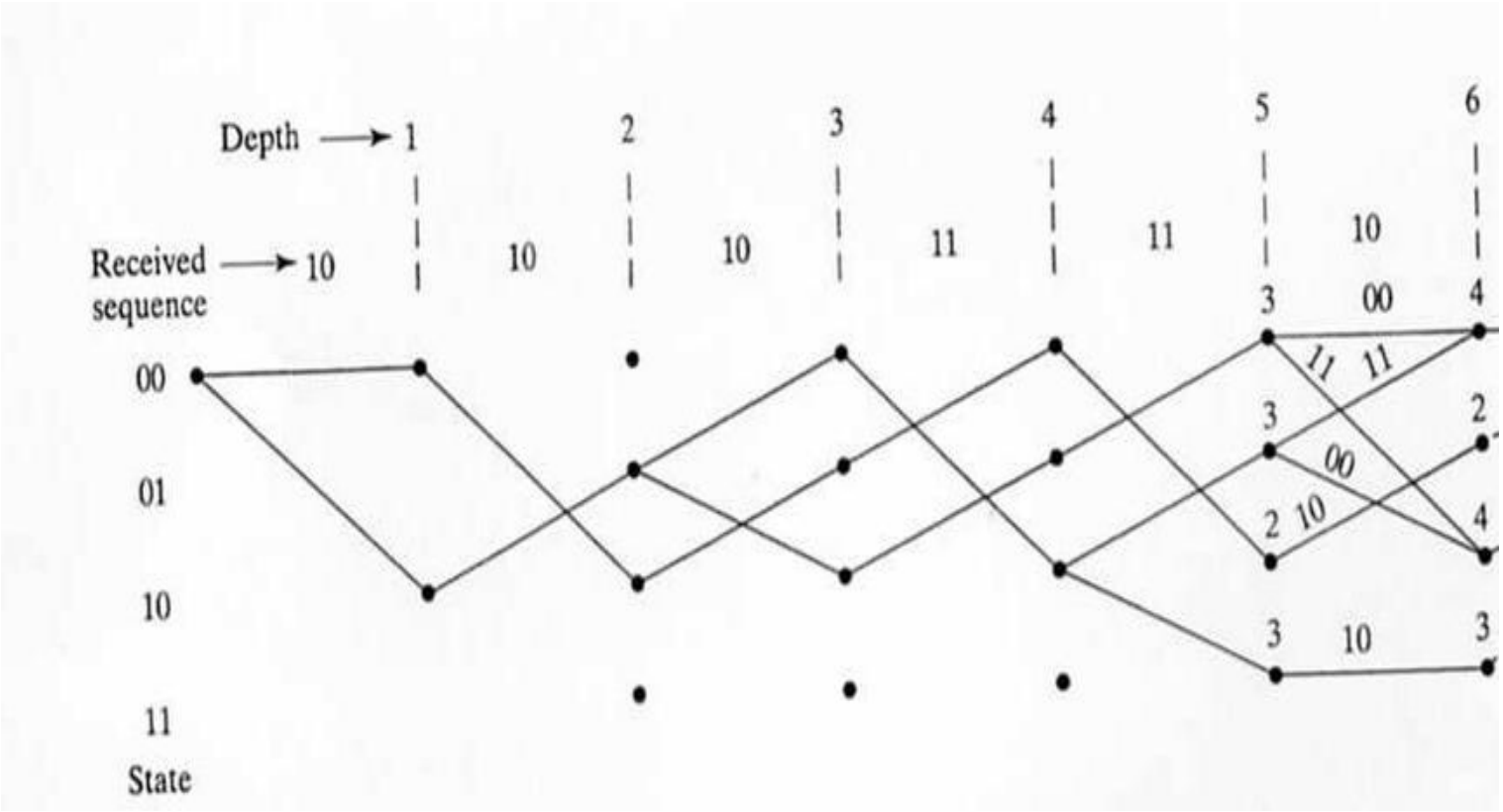

Depth → 1 2 3 4 5 6
Received sequence → 10
10 10 11 11 10
3 00 4
11 11
3 2
00
2 10 4
3 10 3
00
01
10
11
State

Fig.4.7(g)

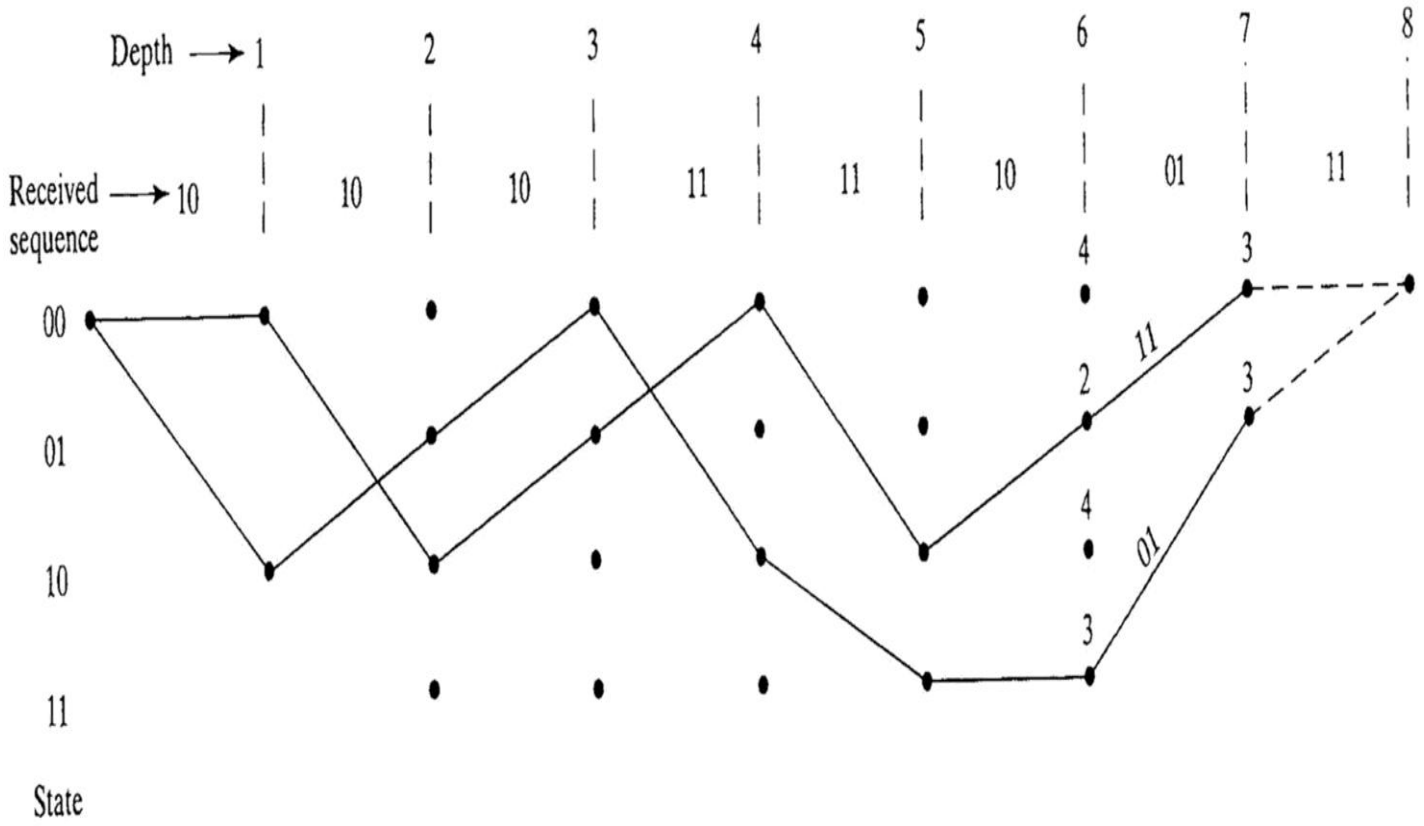
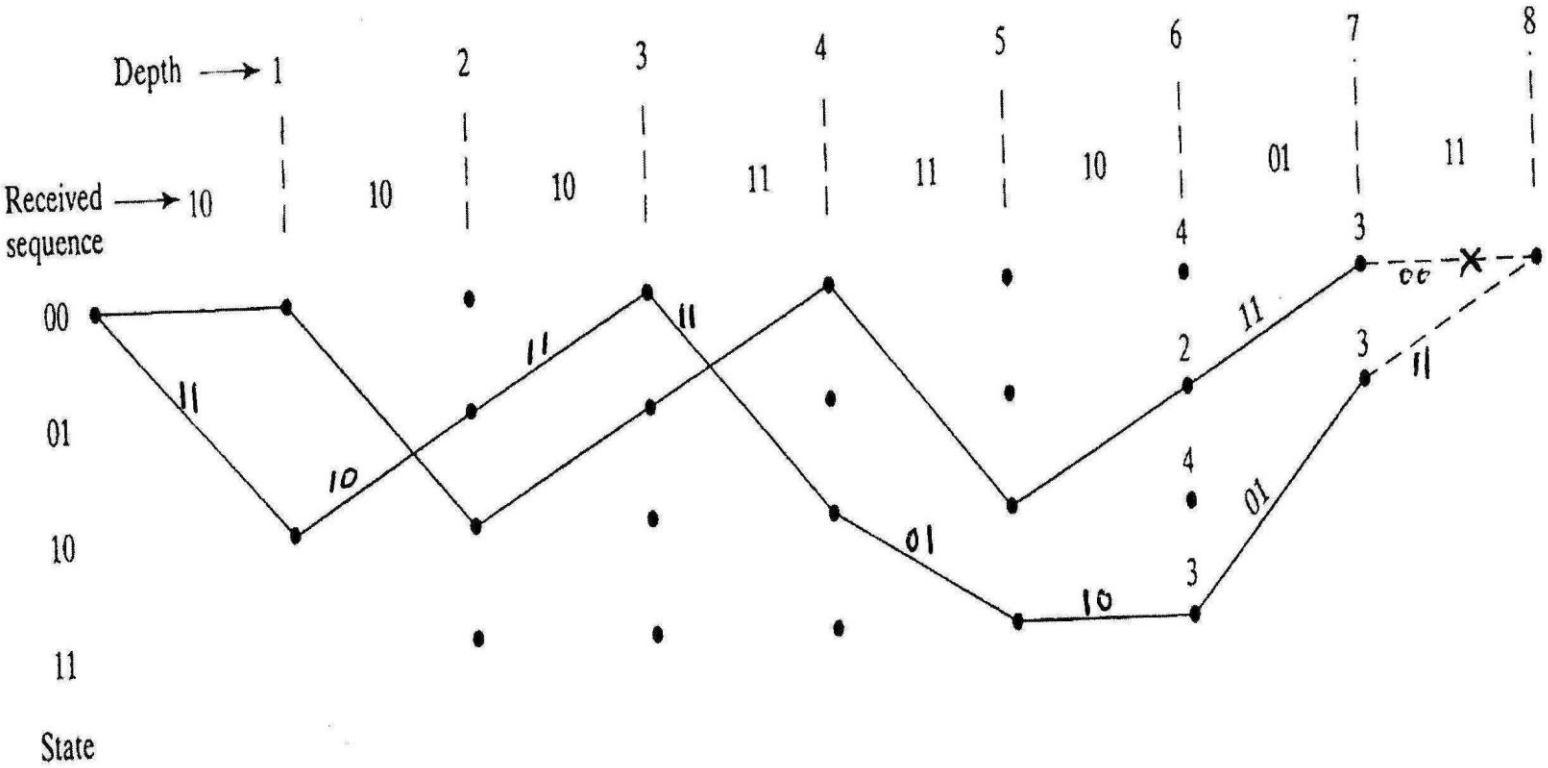

Depth → 1 2 3 4 5 6 7 8
Received sequence → 10 10 10 11 11 10 01 11
00
01
10
11
State
4
3
2
4
3
11
01

Fig.4.7(h)


Depth → 1 2 3 4 5 6 7 8
Received sequence → 10 10 10 11 11 10 01 11
00
01
10
11
State
11
10
11
11
01
10
01
11
00
11
4
2
4
3
3
3

4.8.4 Modifications of Viterbi Algorithm : Truncation

- **For very large L, this is practically impossible, and some trade-offs must be made.**
- **One approach to this problem is to truncate the path memory of the decoder by storing only the most recent q blocks of message bits for each survivor, where $q << L$.**
- **After the first q blocks of the received sequence have been processed by the decoder, the decoder memory is full.**
- **As soon as the next received block is processed, a decoding decision must be made on the first block of l message bits, since they can no longer be stored in the decoder memory.**

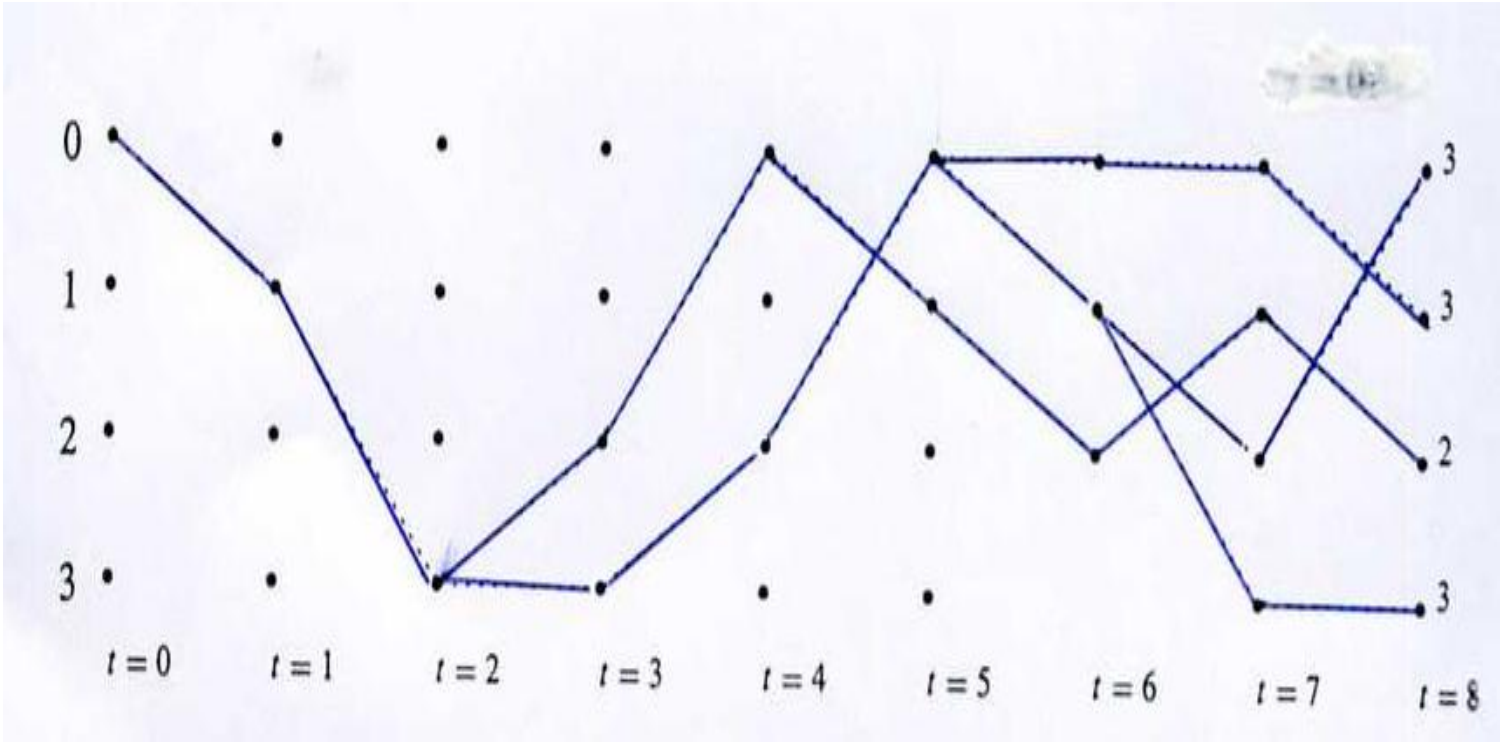
- **The optimum strategy to make this decision is to select the survivor with the best metric, and the first block of k message bits of this survivor is chosen as the decoded message block and released to the user.**

- **After the first decoding decision is made, subsequent decoding decisions are made in the same manner for each new received block processed.**
- **Note that decoding decisions made in this way are no longer maximum likelihood, but can be almost as good if q is large enough.**

- **Experience and analysis have shown that if *r* is in the order of 5 times of the encoder memory *M* or more, with probability approaching "1", all the 2*M* survivors stem from the same branch *r* levels back as shown in Figure 4.8 .**
  Hence there is no ambiguity in making decoding decision. The parameter *q* is called the decoding span (or depth).

Figure 4.8

Decoding decision with a finite path memory *q*



4.9 Coding Gain

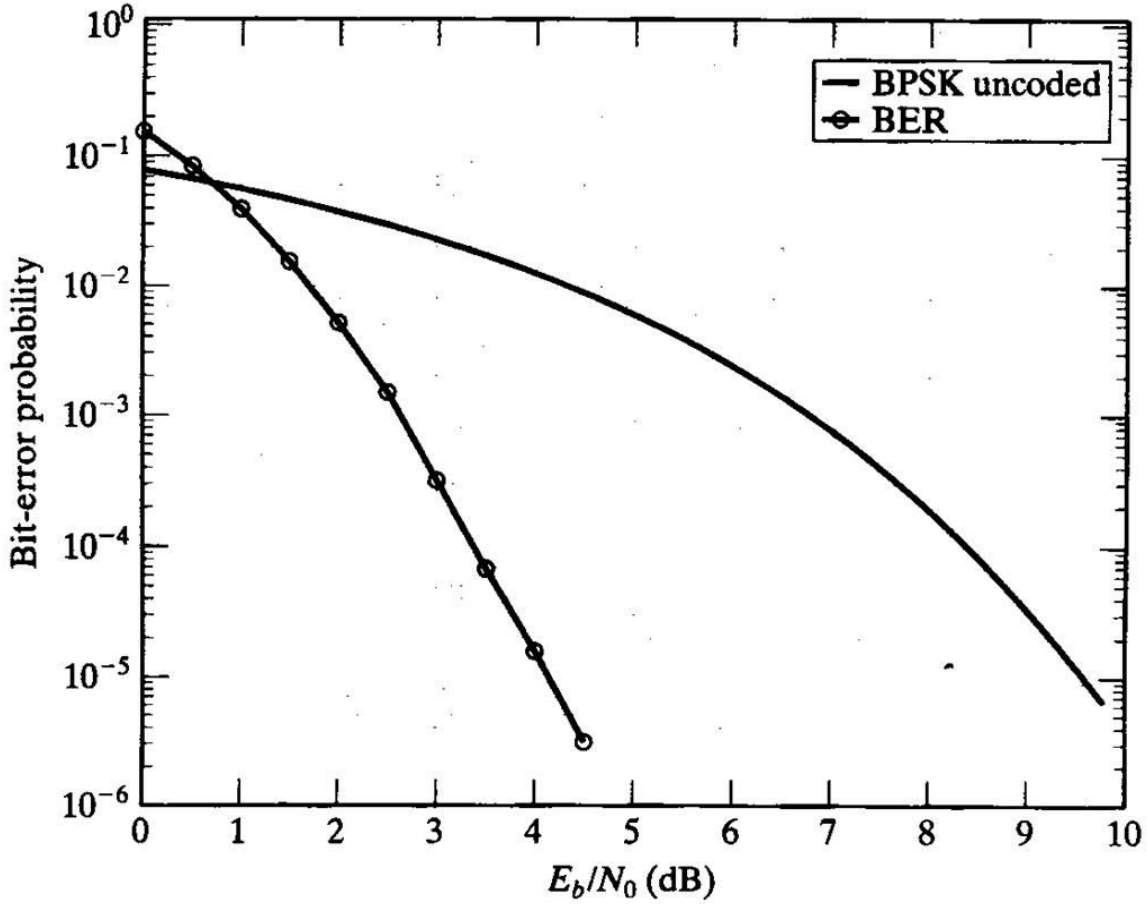
- **Coding gain is defined as the reduction in the require E_b/N_0 (usually expressed in decibels) to achieve a specified error probability of a coded system over an uncodeed system with the same modulation and channel characteristic , as illustrated in Fig.4.9.**
- **For an uncoded coherent BPSK system with an AWGN channel, the bit-error rate simply the transition probability,**

$$P_b(\mathrm{E}) = Q\left(\sqrt{(2E_b/No)}\right)$$

- **For large E_b/N_0, this error rate (without coding) is approximated by**

$$P_b(\mathrm{E}) \fallingdotseq 0.282 \exp\left(-E_b/N_0\right) \qquad (4.24)$$

Fig.4.9 Illustration of coding gain



4.10 Punctured Convolutional Codes

- **In many bandwidth-limited application s , high-rate or low-redundancy convolutional codes are desirable . However, the Viterbi decoder for these codes is often quite complicated . For the (*n,l.M*) binary convolutional code , the complexity of the Viterbi decoding is proportional to $2^{l M}$.**
- **Puncturing is a technique used to make a *k/n* rate code from a "basic" rate 1/2 code. It is reached by deletion of some bits in the encoder output. Bits are deleted according to *puncturing matrix*.**
- **This has the same effect as encoding with an error-correction code with a higher rate, or less redundancy. However, with puncturing the same decoder can be used regardless of how many bits have been punctured, thus puncturing considerably increases the flexibility of the system without significantly increasing its complexity**

- **A pre-defined pattern of puncturing is used in an encoder. Then, the inverse operation, known as depuncturing, is implemented by the decoder. Puncturing is often used with the Viterbi Algorithm in coding systems.**

4.10.1 Rate R= (*n-1*)/ *n* Punctured Convolutional Code

- **We are to make a code with rate 2/3 using the 4-state ,**

  rate $R = 1/2$ **mother code generated by the (2,1,2) non-systematic feedforward convolutional encoder with the generator matrix**

  $$G(D) = [1 + D + D^2 \quad 1 + D^2]$$

  This code has free distance $d_{free} = 5$ **.**

  we should take a basic encoder output and transmit every second bit from the first branch and every bit from the second one, that is, deleting the first bit on every other branch of the trellis, as shown in Fig, 4.10

Example :Puncturing Pattern of 2/3 and 4/5 codes

- **In each of the above case, the puncturing pattern is indicated using a 2 x *T* binary matrix P , where *T* is the puncturing period. The first row of P indicates the bits to be deleted from the first encode sequence , and the second row of P indicates the bits to be deleted from the second encoded sequence. In the matrix P , a 0 indicates a bit to be deleted , and a 1 indicated a bit to be transmitted.**
- **The puncturing patterns in Fig. 4.10 and Fig.4.11 are given by**

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

Fig. (a)

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

Fig. (b)

Fig.4.10 Trellis diagram of R= 2/3 ,M=2 code produced by puncturing *R= 1/2* , *M* =2 convolutional code

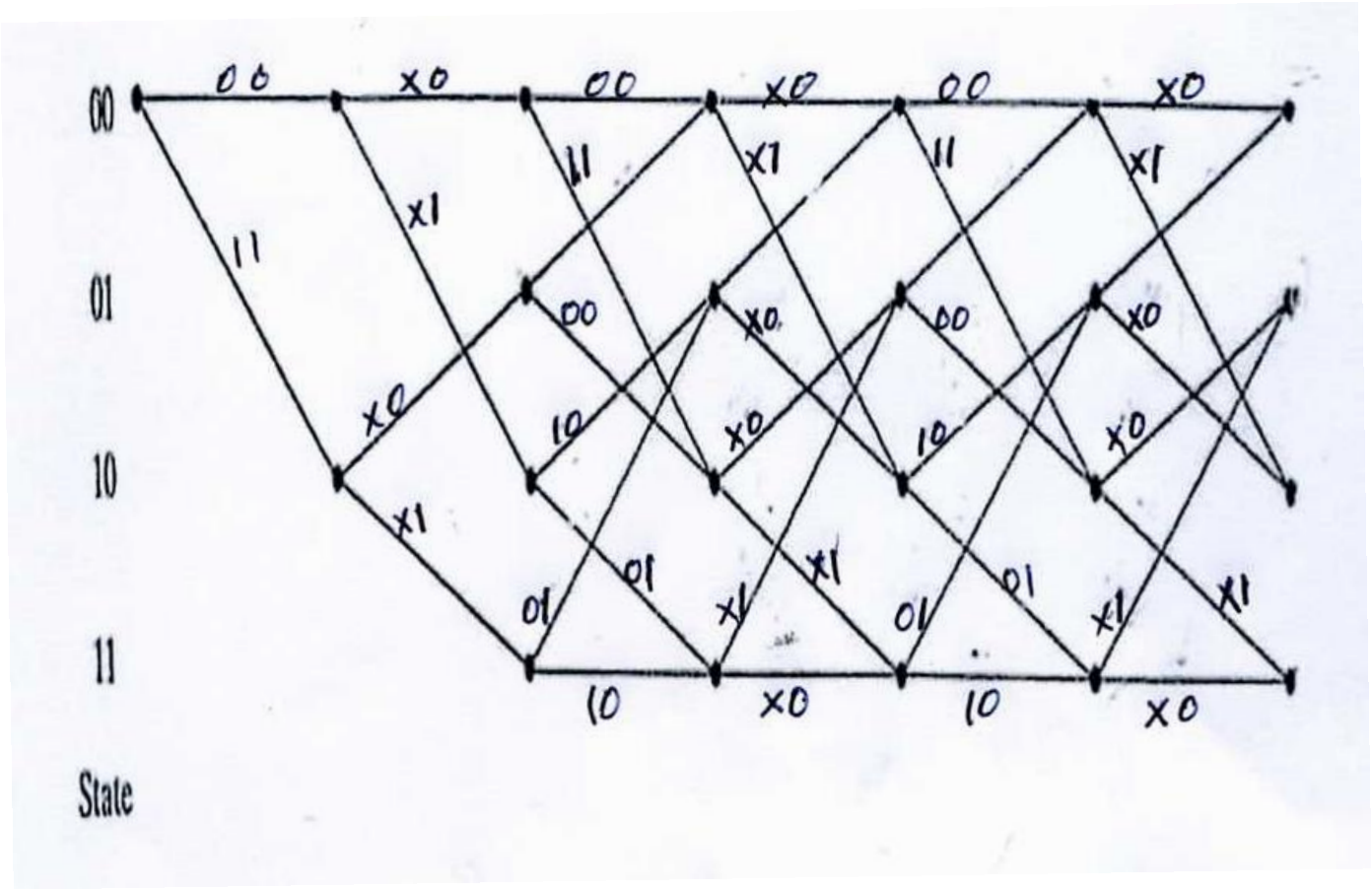
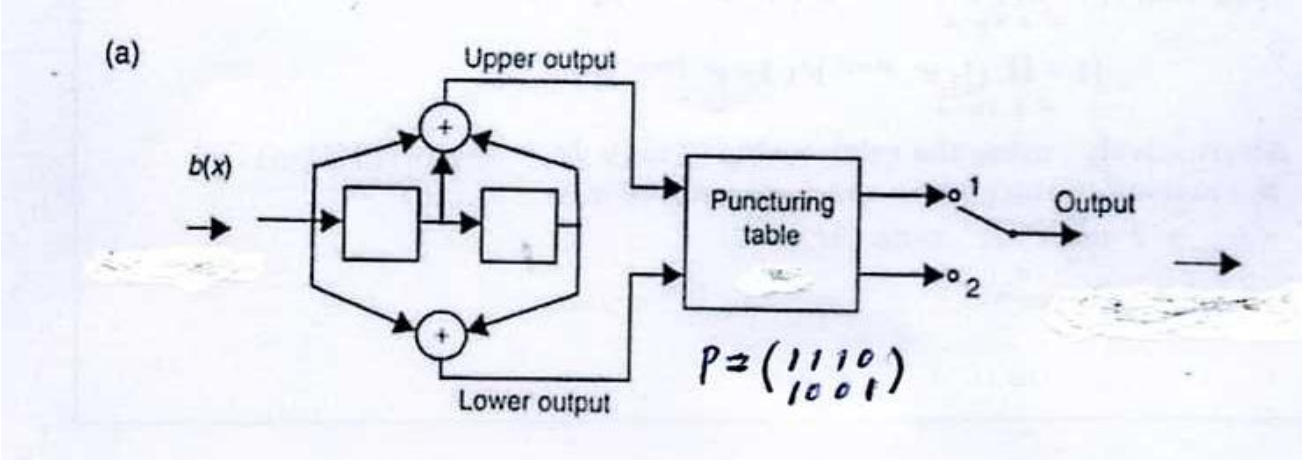
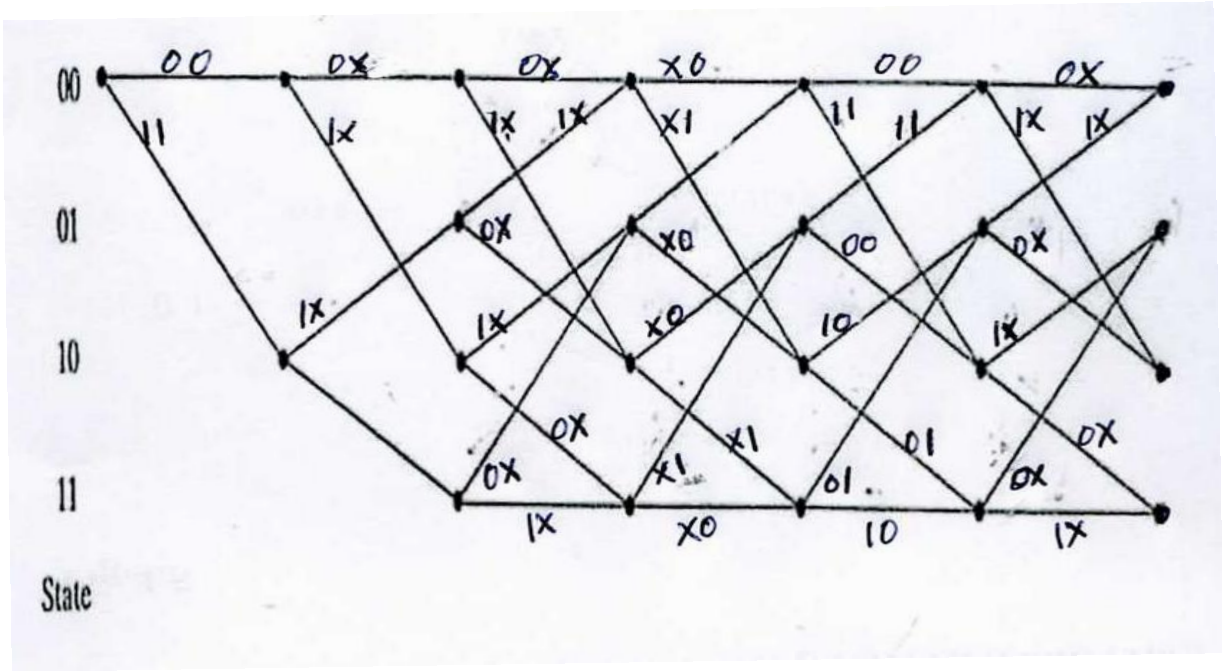


Fig.4.11 *R = 4/5 code obtained by puncturing (2,1,2) code*





4.10.2 Rate-Compatible Punctured Convolutional (RCPC) Codes

- In applications where it is necessary to support two or more different code rates, it is sometimes convenient to make use of rate-compatible punctured convolutional (RCPC) codes.
- An RCPC code is a set of two or more convolutional codes punctured from the same mother code in such a way that the codewords of a higher-rate code can be obtained from the codewords of a lower-rate code simply by deleting additional bits.
- In other words, the set of puncturing patterns must be such that the P matrix of a higher-rate code is obtained from the P matrix of a lower-rate code by simply changing some of the 1's to 0's .
- .

- **An RCPC code then has the property that all the codes in the set have the same encoder and decoder**

Example : Puncturing (2,1.2) convolutional code

$$P(1)=\begin{pmatrix}1&1&1&0\\1&0&0&1\end{pmatrix}\quad P(2)=\begin{pmatrix}1&1&1&0\\1&1&0&1\end{pmatrix},\quad P(3)=\begin{pmatrix}1&1&1&1\\1&1&0&1\end{pmatrix}\quad P(4)=\begin{pmatrix}1&1&1&1\\1&1&1&1\end{pmatrix}$$

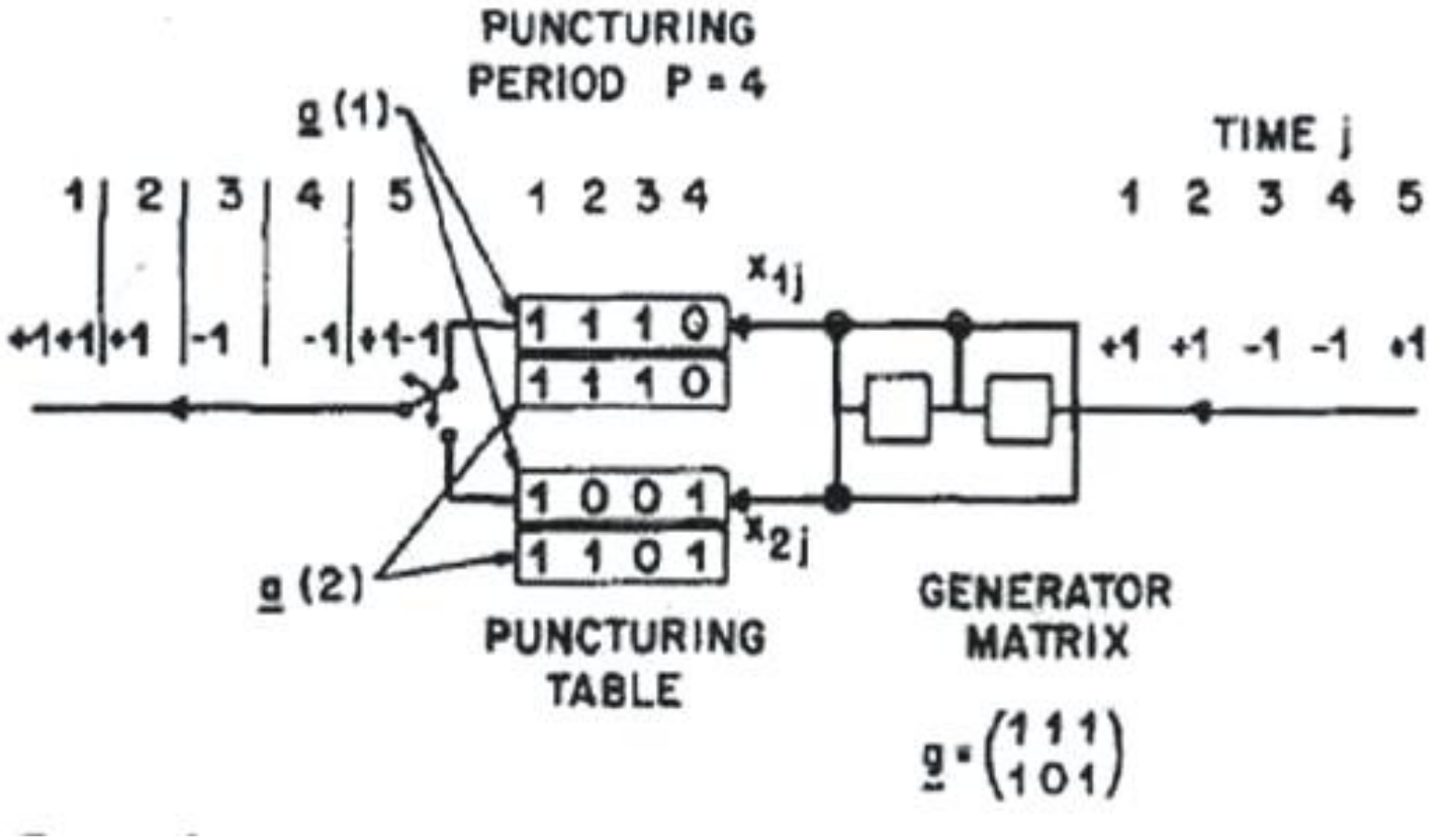
4/5 4/6 4/7 4/8

64

Example of RCPC codes

Rate 1/2 , M =2 convolutional codes

puncturing period P =4



4.11 Tail-Biting Convolutional Codes

4.11.1 Tail-Biting

4.11.2 Encoding Procedure for Tailbiting Codes

4.11.3. Decoding Algorithms for Tailbiting Codes

4.11.4. Wrap-around Viterbi Algorithm (WAVA)

References

1. H.H. Ma and J.K. Wolf , " On tailbiting convolutional codes", IEEE Trans. Commun., vol. 34 , pp. 104-111 , Feb.1986.
2. R.V. Cox and C.E. Sundberg ," An efficient adaptive circular Viterbi algorithm for decoding generalized tailbiting convolutional codes ," IEEE Trans. Veh. Technol. Vol.43, pp. 57-68 , Feb.1994.
3. R.Y.Shao, S.Lin and P.C.Fossorier , " Two decoding algorithms for tailbiting codes," IEEE Trans. Commun. Vol.51, , pp. 1658-1665 ,Oct.. 2003.

4.11 Tail-Biting Convolutional Codes

4.11.1 Tail-Biting Technique

- **A tail-biting convolutional code is obtained by terminating the encoder output sequence after the last information block in the input sequence.**
- **In other words, no " tail " of input blocks is used to force the encoder to the all-zero state. In this case, the input sequence has length *hk* , the output sequence has length *hn* , and the rate of the resulting tail-biting convolutional code is**

$$R = hl / hn = l / n$$

There is no rate loss associated with this trellis termination technique.

- **The tail-biting convolutional codes have been adopted as the forward error correcting code for data and/or overhead channels in many wireless communications , such as EDGE, WiMAX and LTE.**

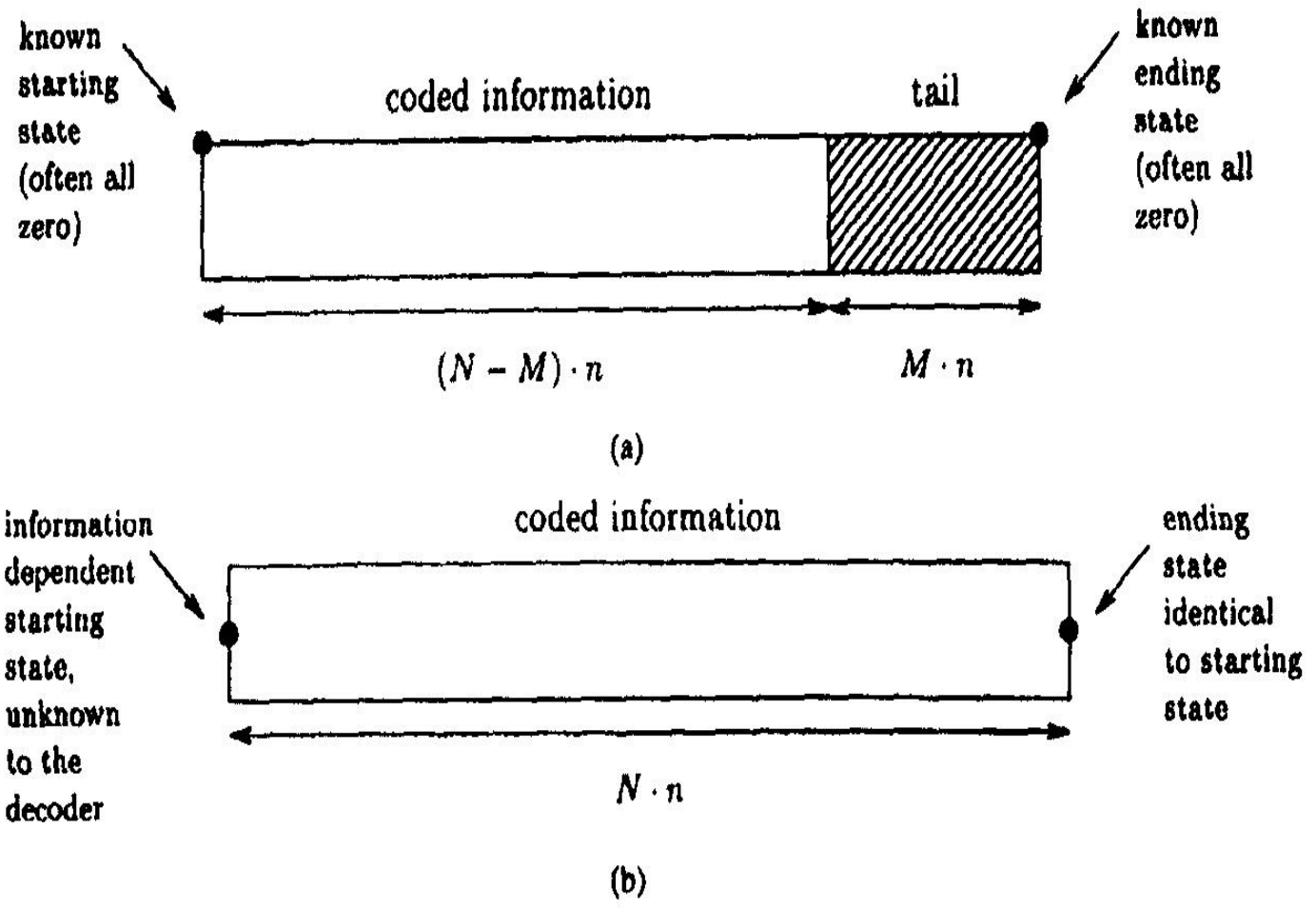
- **For the tail-biting case, the allowable codewords are those that start and end in the same state. Thus , one approach to implement a ML decoder for a tail-biting code , as suggested by Ma and Wolf, is to run M parallel Viterbi algorithms , where M is the number of states in the trellis structure of the convolutional code. Each Viterbi algorithm has a different postulate for the starting and ending states.**

  The Viterbi algorithm that produces the globally best metric gives the ML estimate of the transmitted bits.

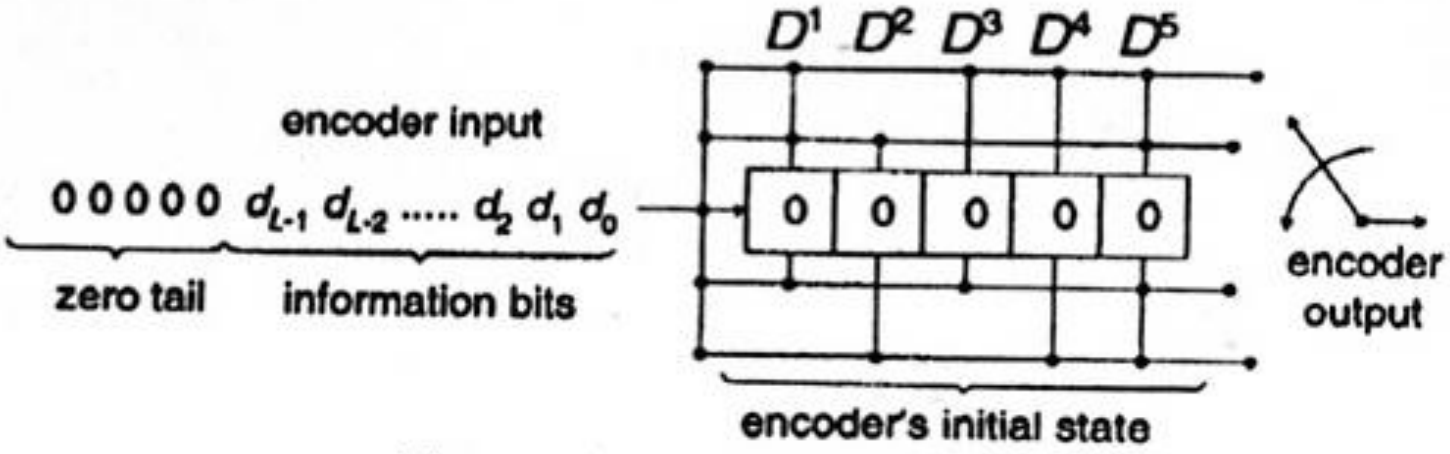
  The obvious disadvantage of this method is that the decoding method for tail-biting codes is M times as complex as the decoding algorithm for the code with tail bits .

4.11.2 . Encoding Procedure for Tailbiting Codes

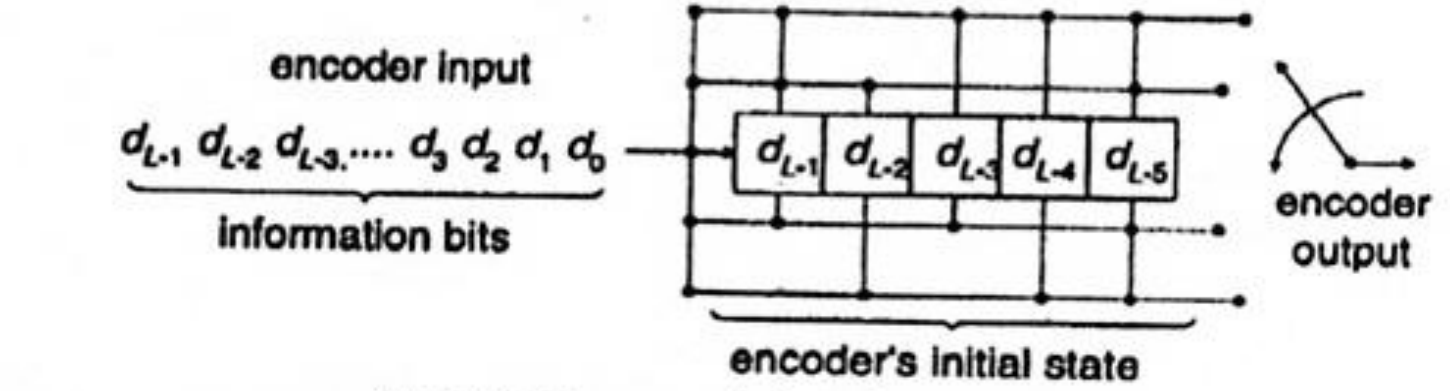
- **In the tailbiting convolutional coding system, we encode and decode a block of N trellis sections without a known tail, thus keeping the effective rate of transmission equal to the code rate. This is done by letting the encoder start and end in the same (for the decoder unknown) state , as illustrated in the following figure.**
- **The encoding procedure to achieve this is straightforward. In its simplest form , the encoder starts encoding in the state given by the last M information bits of message. This will also be the ending state. The decoder now has to make a decision about the unknown information and starting (= ending) state.**



(a) Block of coded information and known tail. Rate $k/n \cdot (N - M)/N$. (b) Block of coded information and unknown tail. Rate k/n.

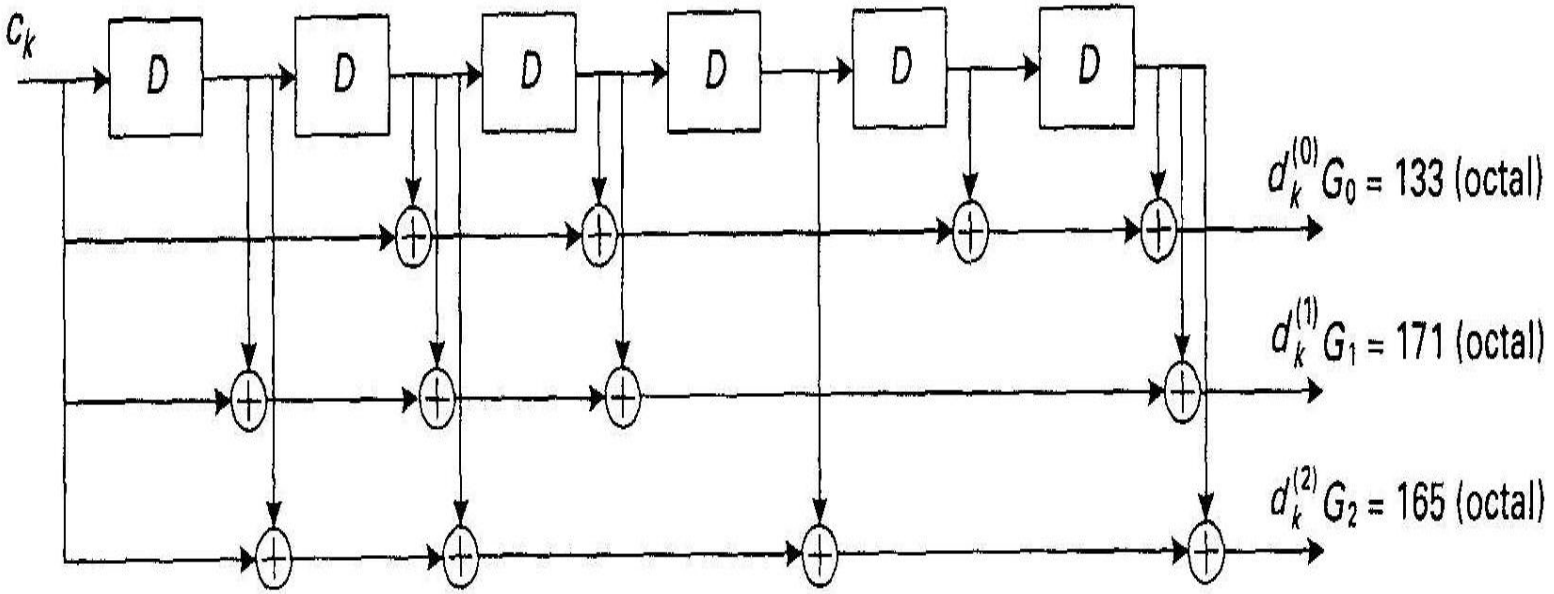


(a) zero-tail convolutional encoder

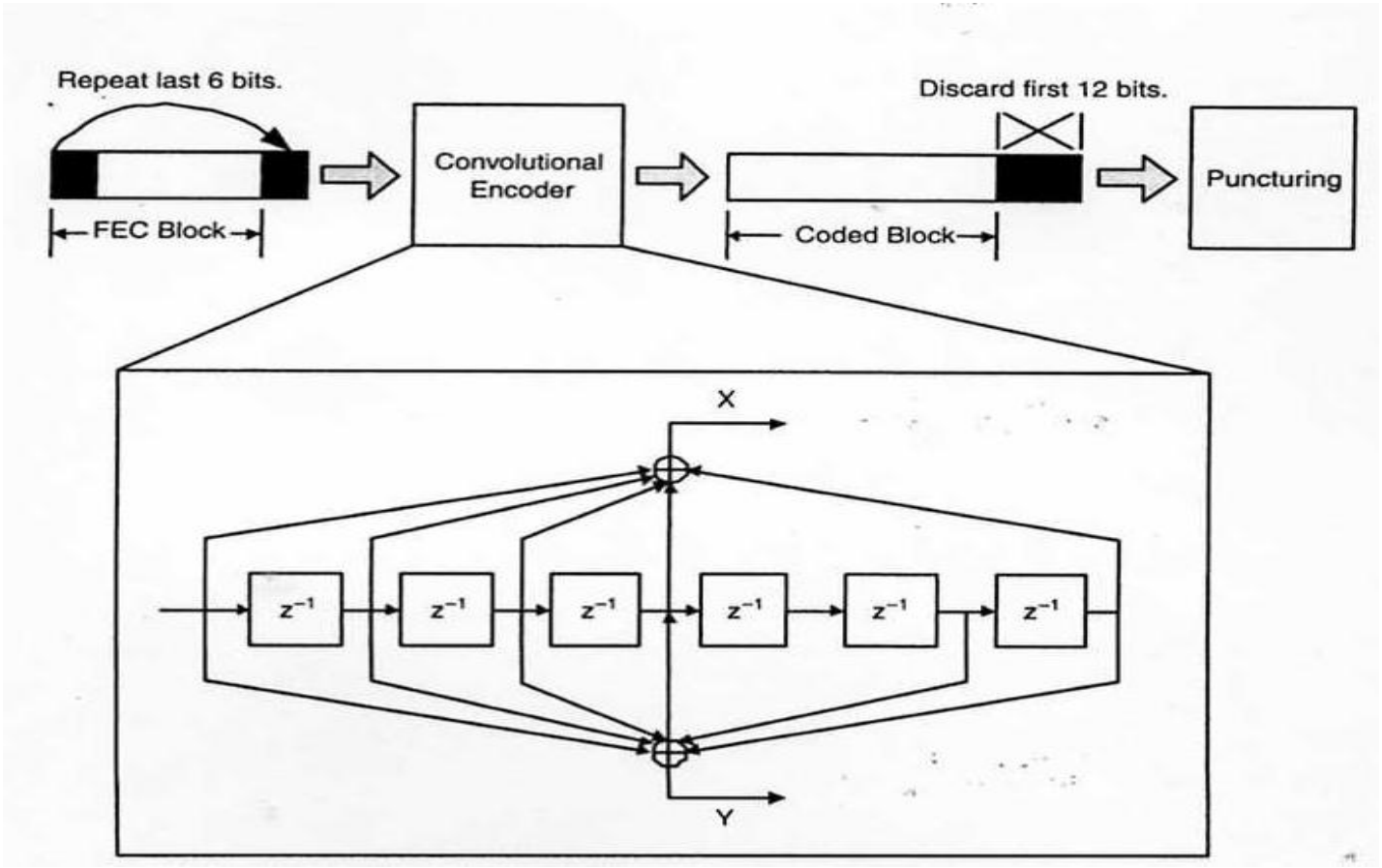


(b) tail-biting convolutional encoder

- **A rate 1/3 tailbiting convolutional code , constraint length K=7 . The initial values of the shift registers are set to equal to the last six information bits in the input stream. This code is defined for LTE .**



- **A tailbiting convolutional code for IEEE 802.16e-2005**



4.11.3. Decoding Algorithms for Tailbiting Codes

- **To perform optimum ML decoding for tail-biting convolutional codes, all Viterbi decoders with all possible 2^M different starting and ending states are performed exhaustively, the decoded path with the best metric is chosen.**
- **Several suboptimum algorithms which are more or less complex and more or less efficient, have been proposed in the literature.**

One of the suboptimum algorithm due to Ma and Wolf is the so-called two-step algorithm .The number of trials that the decoder uses grows exponentially in the code memory and the algorithm is only somewhat simpler than the optimum decoder.

- **In the following , we discuss a recently proposed decoding algorithm called Wrap-around Viterbi Algorithm by Shao, Lin and Fossorier (2003), denotes as WAVA .**

R.Y.Shao, S.Lin and P.C.Fossorier , " Two decoding algorithms for tailbiting codes," IEEE Trans. Commun. Vol.51, , pp. 1658-1665 ,Oct.2003.

4.12 Trellis Coded Modulation

- **Error correcting coding such as convolutional coding or block coding, leads to a coding gain at the cost of spectral efficiency Although this is attractive for power-limited applications , it is not desirable for band-limited applications.**
- **In 1982, Ungerboeck published a paper in which he proposed a technique to achieve the coding gain without bandwidth expansion or reducing data rate. The basic concept of this technique is to perform coding gain onto an expanded modulation signal set.**
- **That is, the number of signal points in the constellation used is larger (usually twice) than what is required for the modulation format of interest.**

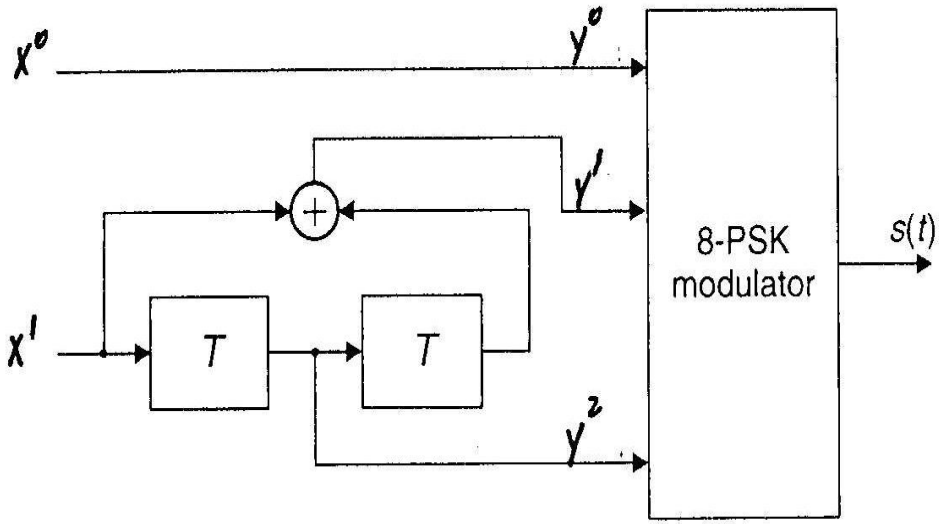
- **TCM has been generally accepted that modulation and coding should be combined in a single entity to provide good coding gain over Gaussian channels with no expansion in bandwidth.**

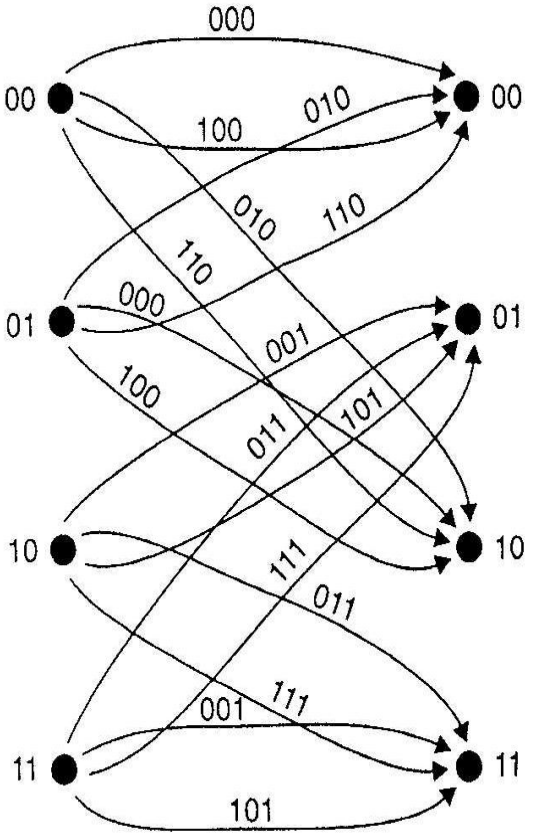
  There has been much progress in improving the reliability of digital communication over a Rayleigh fading channel by means of the trellis coded modulation .

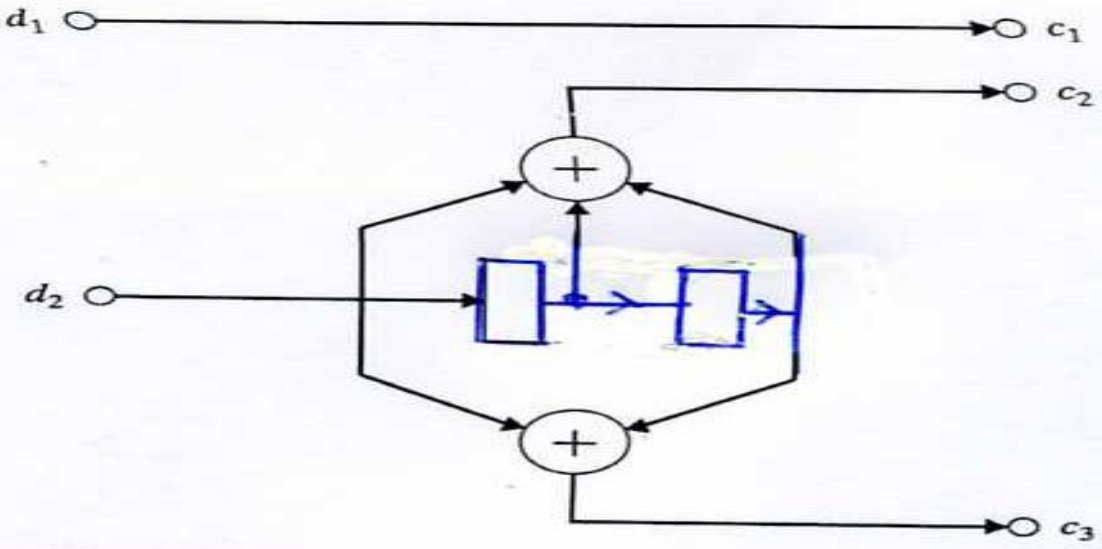
- **Ungerboeck presented an effective method , denoted as set partitioning, for mapping the code bits into signal points such that the minimum Euclidean distance is maximized.**
- **In this joint approach to coding and modulation , convolutional codes and MPSK (or QAM) are mostly employed. The combined convolutonal code and modulation is denoted as Trellis Coded Modulation (TCM). Decoding is performed by using a soft-decision Viterbi decoder .**
- **TCM experienced a fast transition from research to practical applications. In 1984,a generation of modems using TCM became available, achieving reliable transmission at speedof 14.4 kbits/s on private modems and 9.6 kbits/s on switshed –network modems.TCM was adopted in the ITU-T Rec. V.32/33/34 for data communications over the standard telephone network.**
- **Note that V.90 , the 56 kbits/s modem in common use today**
- **TCM is also used in IEEE 802.15.3**

4.12.1 TCM Encoder

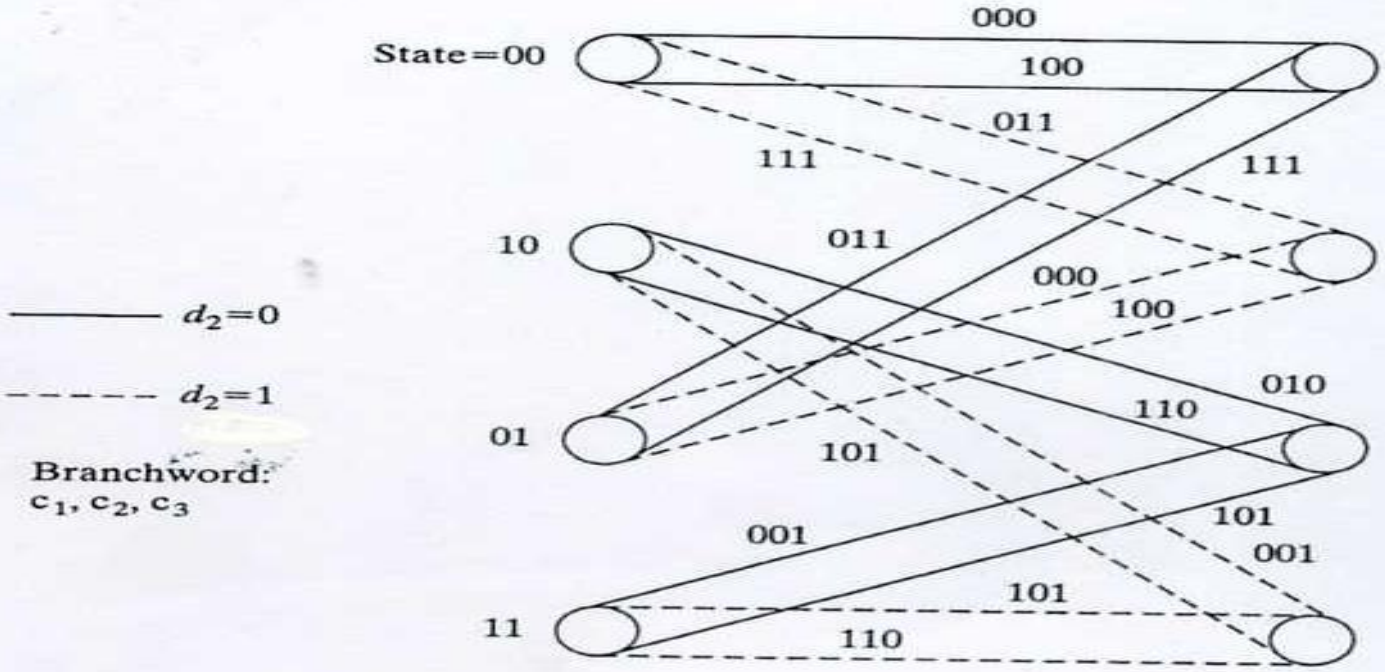
- **Fig. 4.Xa shows an example of TCM , a combination of (3,2 ,2) convolutional encoder and 8-PSK modulator. The corresponding trellis diagram is shown in Fig.4.xb**



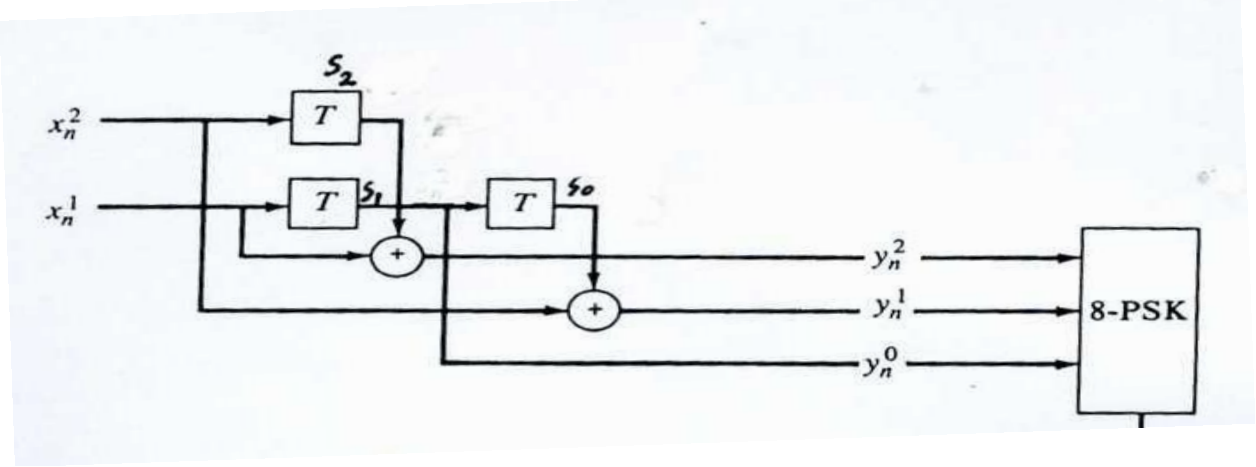




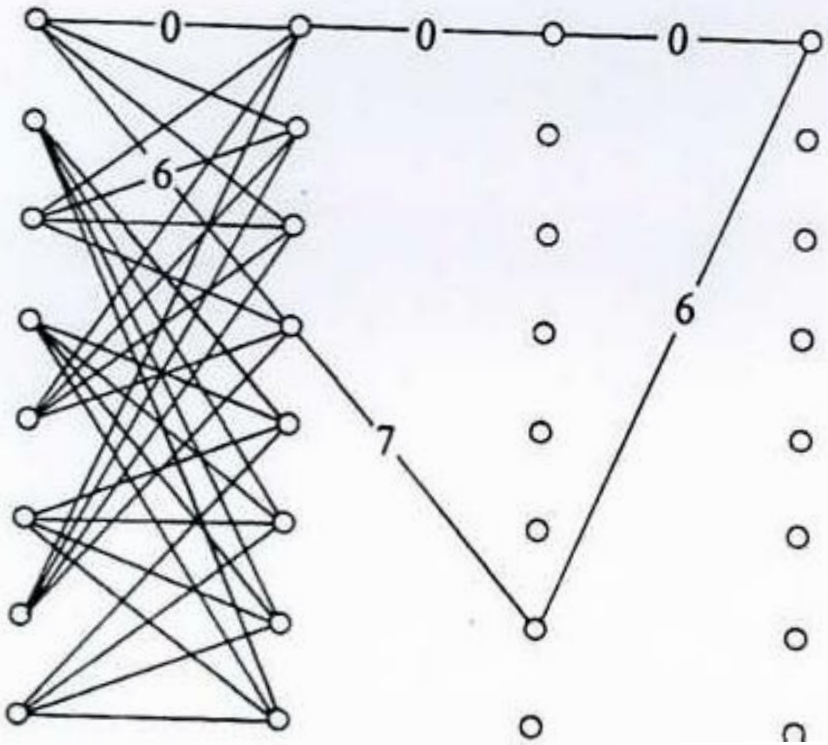
(a) Convolutional encoder

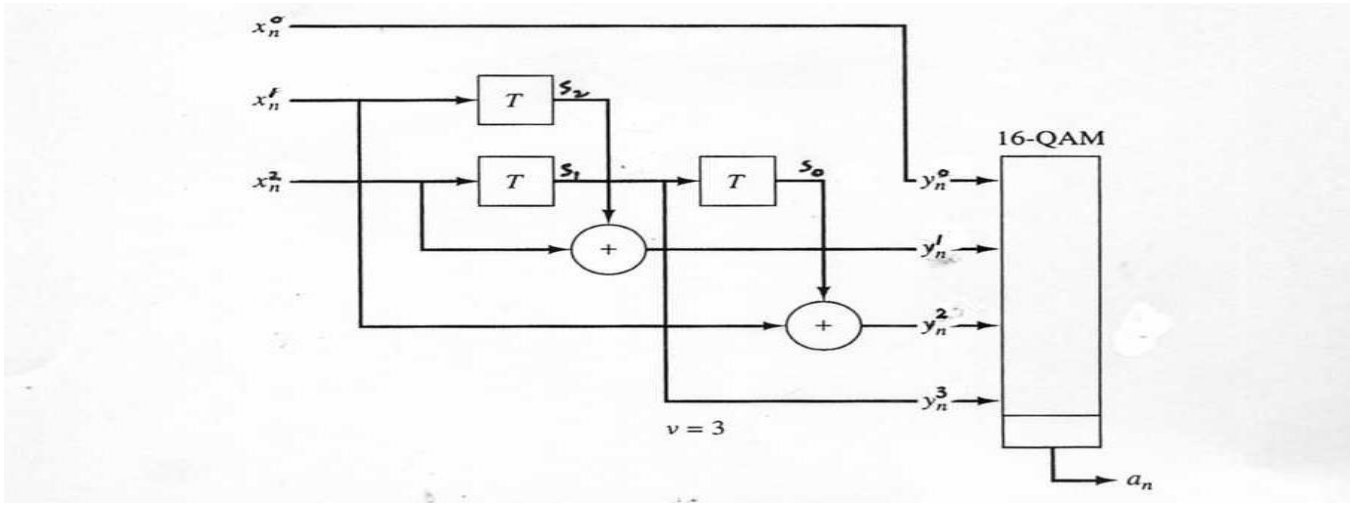


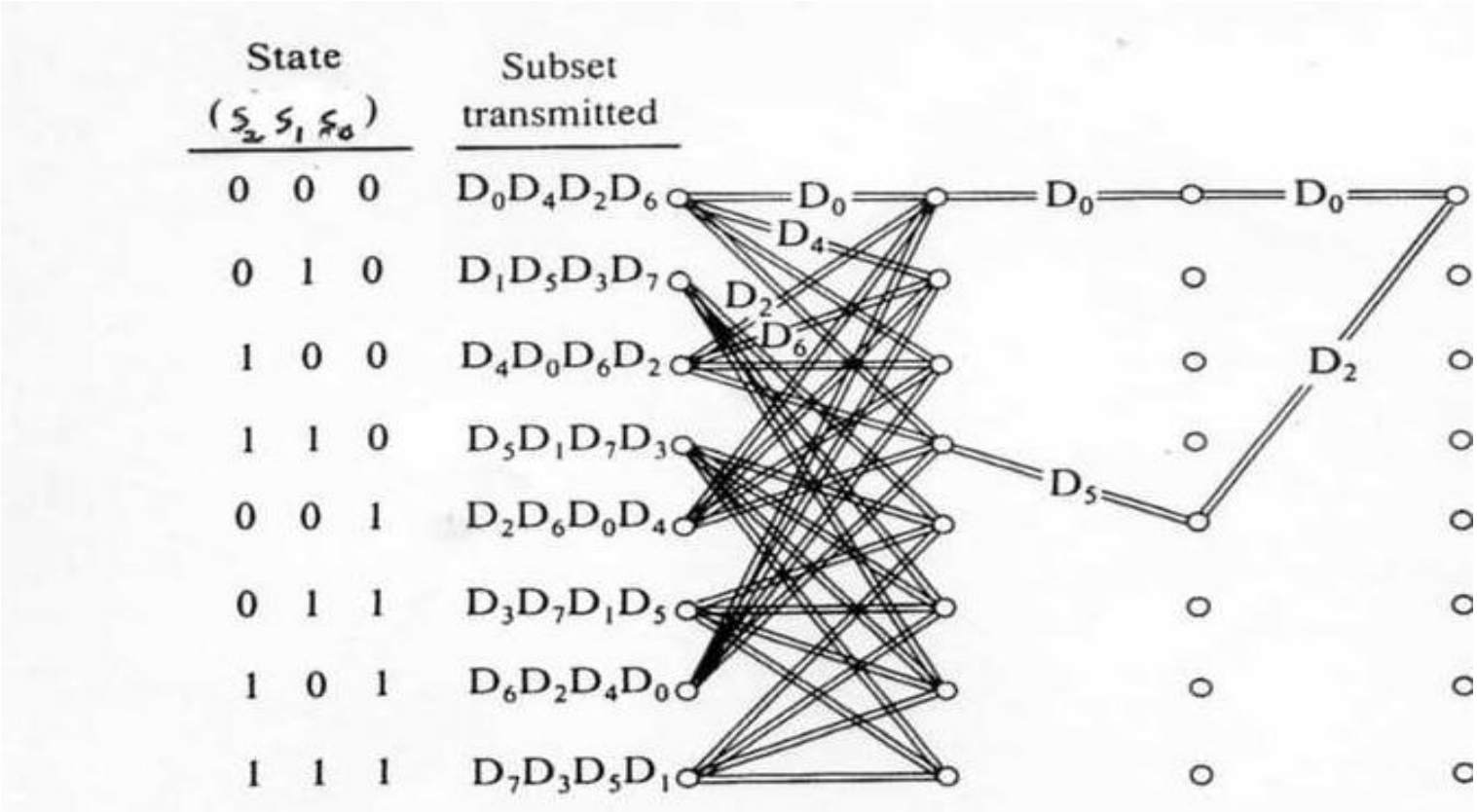
(b) Trellis diagram



State $S_2\ S_1\ S_0$	Subset transmitted
000	0426
010	1537
100	4062
110	5173
001	2604
011	3715
101	6240
111	7351




x_n^0
x_n^1
x_n^2
T
T
T
s_2
s_1
s_0
+
+
16-QAM
y_n^0
y_n^1
y_n^2
y_n^3
$\nu = 3$
a_n


State
$(s_2\ s_1\ s_0)$
Subset
transmitted
0 0 0
$D_0D_4D_2D_6$
0 1 0
$D_1D_5D_3D_7$
1 0 0
$D_4D_0D_6D_2$
1 1 0
$D_5D_1D_7D_3$
0 0 1
$D_2D_6D_0D_4$
0 1 1
$D_3D_7D_1D_5$
1 0 1
$D_6D_2D_4D_0$
1 1 1
$D_7D_3D_5D_1$
D_0
D_4
D_2
D_6
D_0
D_0
D_2
D_5

4.12.2 Set-Partition and Distance Structure

- **Consider the 8-PSK signal constellation (expanded constellation from QPSK) as shown in Fig.6xa , where each signal point is labeled with by a 3-tuple (y^0 , y^1 , y^2) .**

  The constellation points are partitioned into subsets . The points in each subset are far apart in Euclidean distance and are made to correspond to the uncoded bits .

  If the average signal power is chosen equal to r^2 , then the minimum distance between any two points is equal to

  $d_0 = 2\ r \sin(\pi/8) = 0.765\ r$.

  In the first partitioning , the eight points are subdivided into two subsets of four points each , such that the minimum distance between points becomes

  $d_1 = \sqrt{2}\ r$.

In the second level of partitioning , each of the two subsets is subdivided into two subsets of two points , such that the minimum distance becomes $d_2 = 2r$.

The partitioning process continues until only one point is left in the subset .

- **Fig.4.xx shows the set-partition diagram of a 16-point QAM constellation , where the average power of the QAM signal (amplitude-square) is chosen equal to 1.**
- **The assignment process can be viewed in terms of a trellis and proceeds according to the following set partitioning rules :**

1. **All parallel transitions in the trellis are assigned the maximum possible Euclidean distance.**
2. **All transition emanating or merging into a trellis state are assigned next to largest possible Euclidean distance.**
3. **All signals are used equally often .**

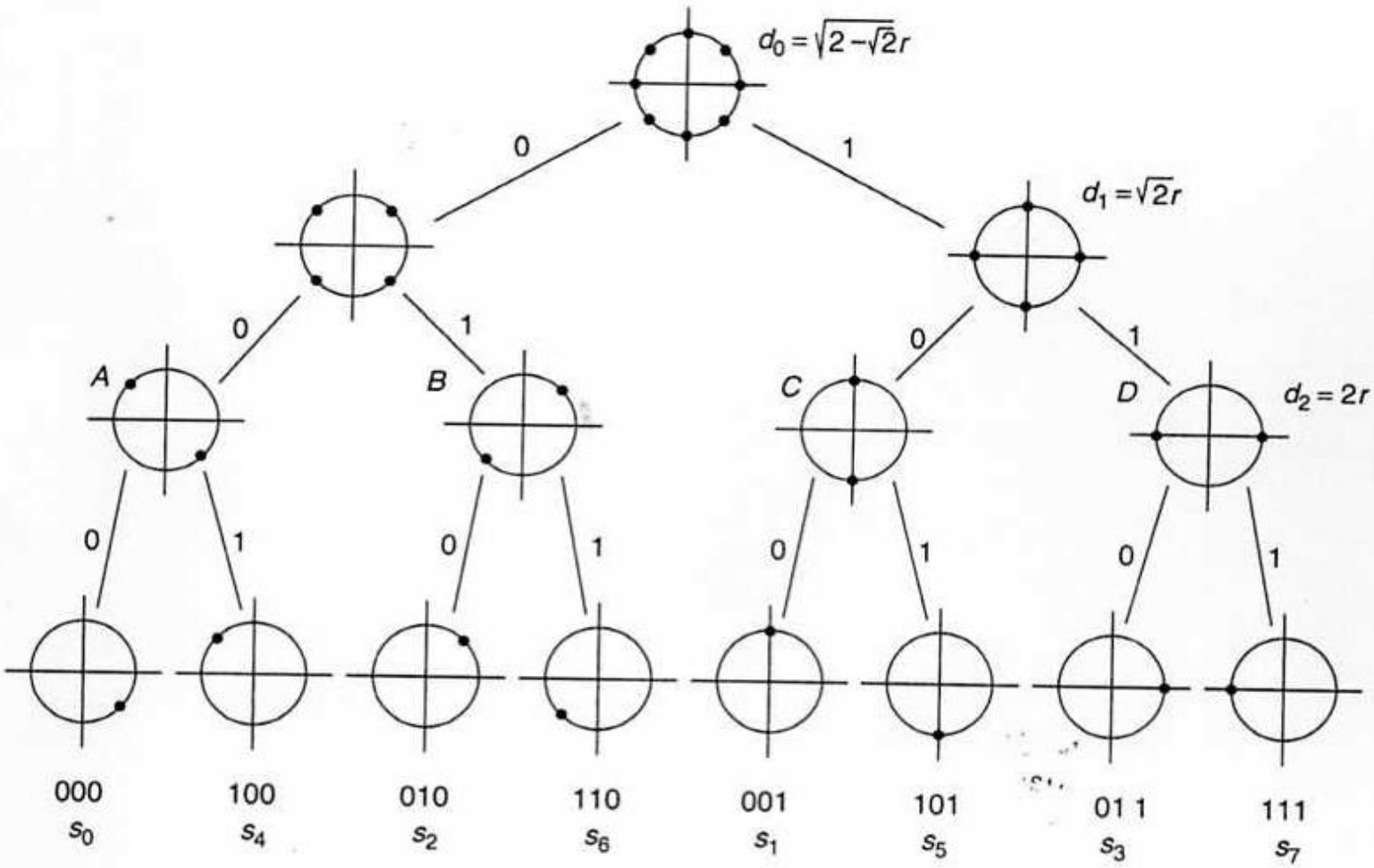


Figure 4.47 Example of set partitioning for 8-PSK modulation

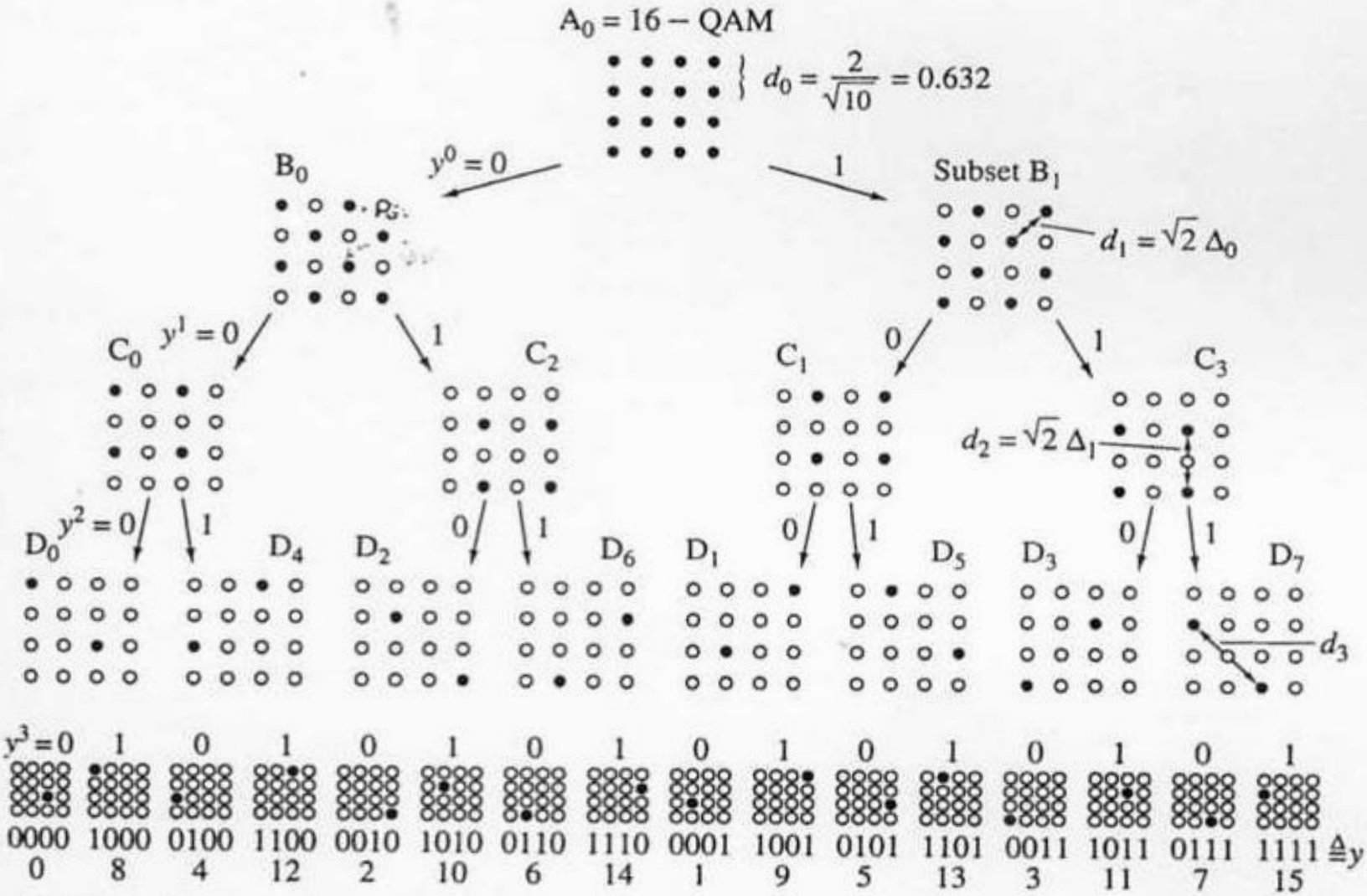


FIGURE 13.4.4 Mapping by set partitioning of the 16-point QAM constellation. From G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Trans. Inf. Theory*, © 1982 IEEE.

4.12.3 Decoding TCM

- **Due to the one-to-one correspondence between signal sequence (baseband) and paths traversing the trellis , maximum likelihood (ML) decoding consists of searching trellis path with the minimum Euclidean distance to the received signal sequence.**
- **If a sequence of length L is transmitted , and the sequence $y_0 , y_1 , \ldots, y_{L-1}$ is observed at the output of the AWGN channel , then the ML receiver look for the sequence $x_0 , x_1 , \ldots, x_{L-1}$ that minimizes $\sum_{i=0}^{L-1} (y_i - x_i)^2$**

  This is done by Viterbi algorithm .

References:

1.Ungerboeck,G.," Channel coding with multi-level/phase signals, " IEEE Trans. Inform., vol.28, no.1 ,pp 55-67, 1982.