

# Coding Techniques

**Manjunatha. P**

[manjup.jnnce@gmail.com](mailto:manjup.jnnce@gmail.com)

**Professor**

**Dept. of ECE**

J.N.N. College of Engineering, Shimoga

June 28, 2013

# 1 Convolutional Encoding



- 1 Convolutional Encoding
- 2 Convolutional Encoder Representation



- 1 Convolutional Encoding
- 2 Convolutional Encoder Representation
- 3 Formulation of the Convolutional Decoding Problem



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain
- ⑦ Other Convolutional Decoding Algorithms:





- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain
- ⑦ Other Convolutional Decoding Algorithms:
  - Sequential Decoding:



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain
- ⑦ Other Convolutional Decoding Algorithms:
  - Sequential Decoding:
  - Feedback Decoding:



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain
- ⑦ Other Convolutional Decoding Algorithms:
  - Sequential Decoding:
  - Feedback Decoding:
- ⑧ Turbo Codes



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain
- ⑦ Other Convolutional Decoding Algorithms:
  - Sequential Decoding:
  - Feedback Decoding:
- ⑧ Turbo Codes
- ⑨ [1, 2, 3, 4, 5]



- ① Convolutional Encoding
- ② Convolutional Encoder Representation
- ③ Formulation of the Convolutional Decoding Problem
- ④ Properties of Convolutional Codes: Distance property of convolutional codes
- ⑤ Systematic and Nonsystematic Convolutional Codes
- ⑥ Performance Bounds for Convolutional Codes, Coding Gain
- ⑦ Other Convolutional Decoding Algorithms:
  - Sequential Decoding:
  - Feedback Decoding:
- ⑧ Turbo Codes
- ⑨ [1, 2, 3, 4, 5]



# Encoding of Convolutional Codes



- Convolutional codes are commonly specified by **three parameters**:  
( $n, k, K$ ) where
  - $n$  = number of outputs
  - $k$  = number of inputs
  - $K$  = number of memory registers
- The quantity  $k/n$  called **the code rate**, is a measure of the efficiency of the code commonly  $k$  and  $n$  parameters range from 1 to 8 and  $K$  from 2 to 10.
- The **constraint length  $L$**  represents the number of bits in the encoder memory that affect the generation of the  $n$  output bits.

$$\text{Constraint Length, } L = k(K - 1)$$



- A block diagram of a binary rate  $R=1/2$  nonsystematic feedforward convolutional encoder with memory order  $m=3$  ( $n,k,K$  i.e., 2,1,3) is as shown in Figure.
- The encoder consists of an  $K=3$ -stage shift register together with  $n=2$  modulo-2 adders and a multiplexer for serializing the encoder outputs.
- The mod-2 adder can be implemented as EXCLUSIVE-OR gate.
- Since mod-2 addition is a linear operation, the encoder is a linear feedforward shift register.
- All convolutional encoders can be implemented using a linear feedforward shift register of this type.

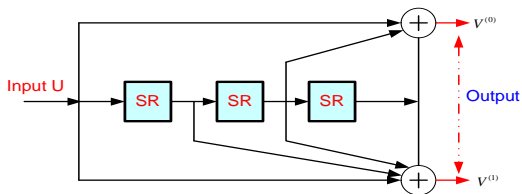


Figure: convolutional encoder (rate=1/2; K=3)





- Constraint length  $K=3$ ,  $k=1$  input,  $n=2$  modulo-2 adders i.e.,  $k/n=1/2$
- At each input bit, a bit is shifted to the leftmost stage and the bits in the registers are shifted one position to the right.
- Connection vector for the encoder is as follows:

$$g_1 = 111$$

$$g_2 = 101$$

- where a '1' in the  $i$ th position indicates the connection in the shift register, and '0' indicates no connection in the shift register and the modulo-2 adder.

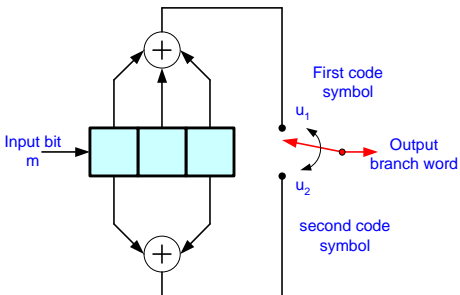


Figure: Convolutional encoder(rate=1/2,  $K=3$ )

- Consider a message vector  $m=1$   
0 1 are inputted one at time in  
the instants  $t_1, t_2$ , and  $t_3$
- $K-1=2$  zeros are inputted at  
times  $t_4$  and  $t_5$  to flush the  
register
- The output sequence of the  
encoder is 1 1 1 0 0 1 0 1 1

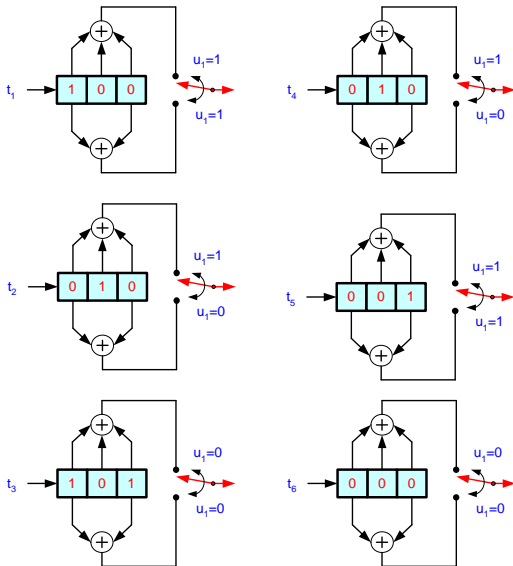


Figure: Convolutionally encoded message



## Impulse Response of the Encoder

- In impulse response a single bit at a time is applied to the encoder that moves through the encoder.

Register contents	Branch word	
	$u_1$	$u_2$
1 0 0	1	1
0 1 0	1	0
0 0 1	1	1

Input bit m	output					
1	1 1	1 0	1 1			
0		0 0	0 0	0 0		
1			1 1	1 0	1 1	
Modulo-2 sum	1 1	1 0	0 0	1 0	1 1	



## Encoding of Convolutional Codes: Polynomial Representation

- In any linear system, time-domain operations involving convolution can be replaced by more convenient **transform-domain operations** involving **polynomial multiplication**.
- Since a convolutional encoder is a linear system, each sequence in the encoding equations can be replaced by corresponding polynomial, and the **convolution operation** replaced by **polynomial multiplication**.
- In the polynomial representation of a binary sequence, the sequence itself is represented by the **coefficients** of the polynomial



- For the given encoder  $g_1(X)$  represents upper connection and  $g_2(X)$  represents lower connection.

$$\begin{aligned}g_1(X) &= 1 + X + X^2 \\g_2(X) &= 1 + X^2\end{aligned}$$

- The out put sequence is found as follows:

$$U(X) = m(X)g_1(X) \text{ interlaced with } m(X)g_2(X)$$

$$\begin{array}{r}m(X)g_1(X) = (1 + X^2)(1 + X + X^2) = 1 + X + X^3 + X^4 \\m(X)g_2(X) = (1 + X^2)(1 + X^2) = 1 + X^4 \\ \hline m(X)g_1(X) = 1 + X + 0X^2 + X^3 + X^4 \\m(X)g_2(X) = 1 + 0X + 0X^2 + 0X^3 + X^4 \\ \hline U(X) = (1,1) + (1,0)X + (0,0)X^2 + (1,0)X^3 + (1,1)X^4 \\ U(X) = (1\ 1) \ (1\ 0) \ (0,0) \ (1\ 0) \ (1\ 1)\end{array}$$

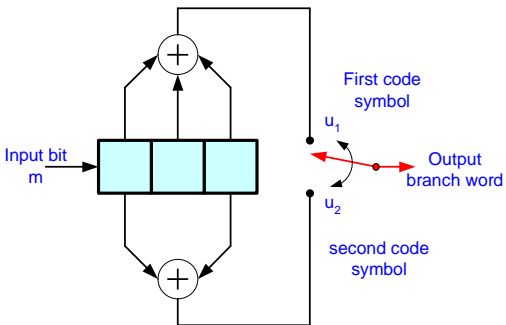
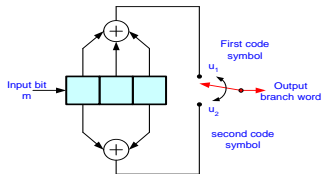


Figure: convolutional encoder(rate=1/2, K=3)

# State Representation and the State diagram

Table: State transition table (rate=1/2, K=3)

Input	Present State	Next State	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	10
1	10	11	01
0	11	01	01
1	11	11	10



- States represent possible contents of the rightmost K-1 register content.
- For this example there are only two transitions from each state corresponding to two possible input bits.
- Solid line denotes for input bit zero, and dashed line denotes for input bit one.

Tuple	State
00	a
10	b
01	c
11	d

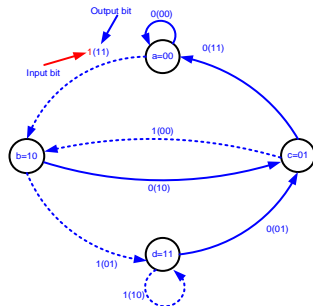


Figure: State diagram for rate=1/2 and K=3

Input bit	Register contents	State at time $t_i$	State at time $t_{i+1}$	Branch word time $t_i$	
				$u_1$	$u_2$
-	000	00	00	-	-
1	100	00	10	1	1
1	110	10	11	0	1
0	011	11	01	0	1
1	101	01	10	0	1
1	110	10	11	0	1
0	011	11	01	0	1
0	001	01	00	1	1

**Output sequence:**  $U = 11\ 01\ 01\ 00\ 01\ 01\ 11$



# The Tree diagram

- State diagram does not represent the time history to track encoder transition as a function of time.
- Tree diagram provides time history.
- Encoding is described by traversing from left to right.
- If the input bit is zero its branch word is found by moving to the next rightmost branch in the upward direction, and if the input bit is one its branch word is found by moving to the next rightmost branch in the downward direction
- Assuming initially the contents of register are zero, if the input bit is zero, its corresponding output is 00 otherwise if the first input bit is one its corresponding output is 11.
- The limitation of tree diagram is that the number of branches increases as a function of  $2^L$ , where L is the number of branch words.

Input	Present State	Next State	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	10
1	10	11	01
0	11	01	01
1	11	11	10

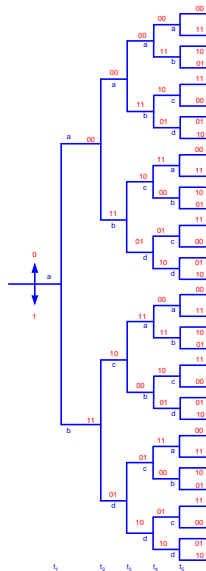


Figure: Tree representation

## The Trellis diagram

- Solid line denotes for input bit zero, and dashed line denotes for input bit one.
- Nodes represent the encoder states, 1 row represents state  $a=00$  subsequent rows correspond to state  $b=10$ ,  $c=01$  and  $d=11$ .
- At each unit of time, the trellis requires  $2^{K-1}$  nodes to represent the  $2^{K-1}$ .

Input	Present State	Next State	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	10
1	10	11	01
0	11	01	01
1	11	11	10

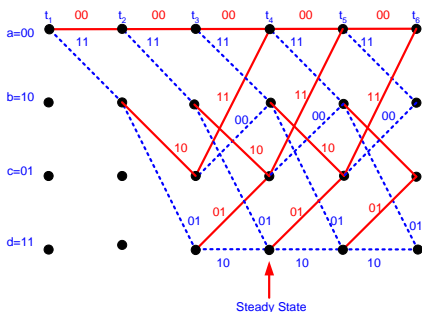


Figure: Trellis diagram for rate=1/2 and K=3



## Encoding of Convolutional Codes: Transform Domain

- In any linear system, time-domain operations involving convolution can be replaced by more convenient **transform-domain operations** involving **polynomial multiplication**.
- Since a convolutional encoder is a linear system, each sequence in the encoding equations can be replaced by corresponding polynomial, and the **convolution operation** replaced by **polynomial multiplication**.
- In the polynomial representation of a binary sequence, the sequence itself is represented by the **coefficients** of the polynomial



- For example, for a  $(2, 1, m)$  code, the encoding equations become

$$V^{(0)}(D) = u(D)g^{(0)}(D)$$

$$V^{(1)}(D) = u(D)g^{(1)}(D)$$

- where  $u(D) = u_0 + u_1(D) + u_2(D^2) \dots$
- The encoded sequences are

$$V^{(0)}(D) = v_0^{(0)} + v_1^{(0)}D + v_2^{(0)}D^2 + \dots$$

$$V^{(1)}(D) = v_0^{(1)} + v_1^{(1)}D + v_2^{(1)}D^2 + \dots$$

- The generator polynomials of the code are

$$g^{(0)}(D) = g_0^{(0)} + g_1^{(0)}D + \dots + g_m^{(0)}D^m$$

$$g^{(1)}(D) = g_0^{(1)} + g_1^{(1)}D + \dots + g_m^{(1)}D^m$$

$$V(D) = [v^{(0)}(D), v^{(1)}(D)]$$

- After multiplexing, the code word become

$$V(D) = v^{(0)}(D^2) + Dv^{(1)}(D^2)$$

- D is a delay operator, and the power of D denoting the number of time units a bit is delayed with respect to the initial bit.
- The general formula after multiplexing (n=numbr of output):

$$V(D) = v^{(0)}(D^n) + Dv^{(1)}(D^n) + \dots + D^{(n-1)}v^{(n-1)}(D^n)$$



## Example 11.1

- For  $(2,1,3)$  convolutional code,  $g^{(0)} = [1 \ 0 \ 1 \ 1]$  and  $g^{(1)} = [1 \ 1 \ 1 \ 1]$  the generator polynomials are

$$g^{(0)}(D) = 1 + D^2 + D^3$$

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

- For the information sequence  $u(D) = 1 + D^2 + D^3 + D^4$ , then

$$v^{(0)}(D) = (1 + D^2 + D^3 + D^4)(1 + D^2 + D^3)$$

$$v^{(0)}(D) = (1 + D^2 + D^3 + D^4 + D^2 + D^4 + D^5 + D^6 + D^3 + D^5 + D^6 + D^7)$$

$$v^{(0)}(D) = (1 + D^2 + D^2 + D^3 + D^3 + D^4 + D^4 + D^5 + D^5 + D^6 + D^6 + D^7)$$

$$v^{(0)}(D) = (1 + D^7)$$

$$v^{(1)}(D) = (1 + D^2 + D^3 + D^4)(1 + D + D^2 + D^3)$$

$$\begin{aligned} v^{(1)}(D) &= (1 + D^2 + D^3 + D^4 + D + D^3 + D^4 + D^5 + D^2 + D^4 + D^5 + D^6 \\ &= +D^3 + D^5 + D^6 + D^7) \end{aligned}$$

$$v^{(1)}(D) = 1 + D + D^3 + D^4 + D^5 + D^7$$

- and the code word is

$$v(D) = [1 + D^7, 1 + D + D^3 + D^4 + D^5 + D^7].$$



## Example 11.1

- and the code word is

$$v(D) = [1 + D^7, 1 + D + D^3 + D^4 + D^5 + D^7].$$

- After multiplexing, the code word become

$$v(D) = v^{(0)}(D^2) + Dv^{(1)}(D^2)$$

$$v^{(0)}(D^2) = (1 + D^{14})$$

$$v^{(1)}(D^2) = Dv^{(1)}(D^2) = D(1 + D^2 + D^6 + D^8 + D^{10} + D^{14})$$

$$v^{(1)}(D^2) = Dv^{(1)}(D^2) = D + D^3 + D^7 + D^9 + D^{11} + D^{15}$$

$$v(D) = v^{(0)}(D^2) + Dv^{(1)}(D^2)$$

$$v(D) = 1 + D + D^3 + D^7 + D^9 + D^{11} + D^{14} + D^{15}$$

- The result is the same as convolution and matrix multiplication.



## Example 11.2

- For (3,2,3) convolutional encoder

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

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

$$\begin{bmatrix} 1 + D & D & 1 + D \\ D & 1 & 1 \end{bmatrix}$$

- For the information sequence  $u^{(1)}(D) = 1 + D^2$ ,  $u^{(2)}(D) = 1 + D$ , the encoding equations give the codeword

$$V(D) = [v^{(0)}(D), v^{(1)}(D), v^{(2)}(D)]$$

$$V(D) = [1 + D^2, 1 + D] \begin{bmatrix} 1 + D & D & 1 + D \\ D & 1 & 1 \end{bmatrix}$$

$$\begin{aligned} V(D) &= [(1 + D^2).(1 + D) + (1 + D)D, (1 + D^2).D + (1 + D)1, \\ &= (1 + D^2).(1 + D) + (1 + D)1] \end{aligned}$$

$$V(D) = [1 + D^3, 1 + D^3, D^2 + D^3]$$

- After multiplexing, the code word become

$$V(D) = v^{(0)}(D^n) + Dv^{(1)}(D^n) + \dots + D^{(n-1)}v^{(n-1)}(D^n)$$

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

$$v(D) = (1 + D^9) + D(1 + D^9) + D^2(D^6 + D^9)$$

$$v(D) = 1 + D + D^8 + D^9 + D^{10} + D^{11}$$



# Decoding of Convolutional Codes





There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

- 1 **Maximum likely-hood decoding**



There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

- 1 **Maximum likely-hood decoding**  
Viterbi decoding



There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

- 1 Maximum likely-hood decoding  
Viterbi decoding
- 2 Sequential Decoding



There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

① **Maximum likely-hood decoding**

Viterbi decoding

② **Sequential Decoding**

i) Stack Algorithm



There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

① **Maximum likely-hood decoding**

Viterbi decoding

② **Sequential Decoding**

i) Stack Algorithm

ii) Fano Algorithm



There are several different approaches to decoding of convolutional codes. These are grouped in two basic categories.

① **Maximum likely-hood decoding**

Viterbi decoding

② **Sequential Decoding**

i) Stack Algorithm

ii) Fano Algorithm



# The Viterbi Decoding Algorithm



- In 1967, Viterbi introduced a decoding algorithm for convolutional codes which has since become known as Viterbi algorithm.
- Later, Omura showed that the Viterbi algorithm was equivalent to finding the shortest path through a weighted graph.
- Forney recognized that it was in fact a maximum likelihood decoding algorithm for convolutional codes; that is, the decoder output selected is always the code word that gives the largest value of the log-likelihood function.





- In order to understand Viterbi's decoding algorithm, expand the state diagram of the encoder in time (i.e., to represent each time unit with a separate state diagram).
- Consider (3, 1, 2) code with  $G(D) = [1 + D, 1 + D^2, 1 + D + D^2]$  and an information sequence of length  $h=5$ .
- The trellis diagram contains  $h+m+1$  time units or levels, and are labeled from 0 to  $h+m$ .
- Assuming that the encoder always starts in state  $S_0$  and returns to state  $S_0$ , the first  $m$  time units correspond to the encoder's departure from state  $S_0$ , and the last  $m$  time units correspond to the encoder's return to state  $S_0$ .



- Not all states can be reached in the first  $m$  or the last  $m$  time units. However, in the center portion of the trellis, all states are possible, and each time unit contains a replica of the state diagram.
- There are two branches leaving and entering each state.
- The upper branch leaving each state at time unit  $i$  represents the input  $u_i = 1$ , while the lower branch represents  $u_i = 0$ .
- Each branch is labeled with the  $n$  corresponding outputs  $v_i$ , and each of the  $2^h$  code words of length  $N = n(h + m)$  is represented by a unique path through the trellis.
- For example, the code word corresponding to the information sequence  $u = (1\ 1\ 1\ 0\ 1)$  is shown highlighted in Figure.



- In the general case of an  $(n, k, m)$  code and an information sequence of length  $kh$ , there are  $2^k$  branches leaving and entering each state, and  $2^{kh}$  distinct paths through the trellis corresponding to the  $2^{kh}$  code words.
- Now assume that an information sequence  $u = (u_0, \dots, u_{h-1})$  of length  $kh$  is encoded into a code word  $v = (v_0, v_1, \dots, v_{h+m-1})$  of length  $N = n(h + m)$ , and that a sequence  $r = (r_0, r_1, \dots, r_{h+m-1})$  is received over a discrete memoryless channel (DMC).
- Alternatively, these sequences can be written as  $u = (u_0, \dots, u_{kh-1})$ ,  $v = (v_0, v_1, \dots, v_{N-1})$ ,  $r = (r_0, r_1, \dots, r_{N-1})$ , where the subscripts now simply represent the ordering of the symbols in each sequence.



- As a general rule of detection, the decoder must produce an estimate of the code word  $v$  based on the received sequence  $r$ .
- A maximum likelihood decoder (MLD) for a DMC chooses as the code word  $v$  which maximizes the log-likelihood function  $\log P(r|v)$ .
- Since for a DMC

$$P(r|v) = \prod_{i=0}^{h+m-1} P(r_i|v_i) = \prod_{i=0}^{N-1} P(r_i|v_i) \quad (12.2)$$

- It follows that

$$\log P(r|v) = \sum_{i=0}^{h+m-1} \log P(r_i|v_i) = \sum_{i=0}^{N-1} \log P(r_i|v_i) \quad (12.3)$$

- where  $P(r_i|v_i)$  is a channel transition probability.
- This is a minimum error probability decoding rule when all code words are equally likely



- The log-likelihood function  $\log P(r|v)$  is called the metric associated with the path  $v$ , and is denoted  $M(r|v)$ .
- The terms  $\log P(r_i|v_i)$  in the sum of Equation (12.3) are called branch metrics, and are denoted  $M(r_i|v_i)$ , whereas the terms  $\log P(r_i|v_i)$  are called bit metrics, and are denoted  $M(r_i|v_i)$ .
- The path metric  $M(r|v)$  can be written as

$$M(r|v) = \sum_{i=0}^{h+m-1} M(r_i|v_i) = \sum_{i=0}^{N-1} M(r_i|v_i)$$

- The decision made by the log-likelihood function is called the soft-decision.
- If the channel is added with AWGN, soft-decision decoding leads to finding the path with minimum Euclidean distance.



- A partial path metric for the first  $j$  branches of a path can now be expressed as
- The following algorithm, when applied to the received sequence  $r$  from a DMC, finds the path through the trellis with the largest metric (i.e., the maximum likelihood path).
- The algorithm processes  $r$  in an iterative manner.
- At each step, it compares the metrics of all paths entering each state, and stores the path with the largest metric, called the survivor, together with its metric.

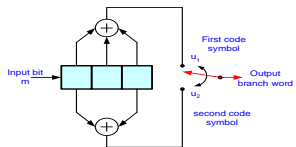


## The Viterbi Algorithm

- 1 **Step 1.** Beginning at time unit  $j = m$ , compute the partial metric for the single path entering each state. Store the path (the survivor) and its metric for each state.
- 2 **Step 2.** Increase  $j$  by 1. Compute the partial metric for all the paths entering a state by adding the branch metric entering that state to the metric of the connecting survivor at the preceding time unit. For each state, store the path with the largest metric (the survivor), together with its metric, and eliminate all other paths.
- 3 **Step 3.** If  $j < h + m$ , repeat step 2. Otherwise, stop.



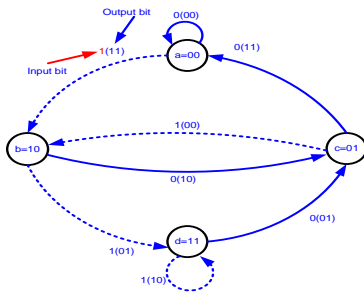
$$G(X) = [1 + X + X^2, 1 + D^2]$$



Input	Present State	Next State	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	10
1	10	11	01
0	11	01	01
1	11	11	10

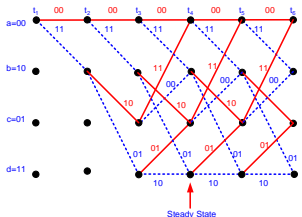
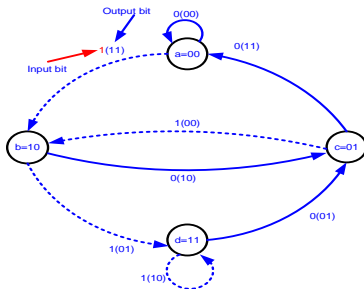
Figure: convolutional encoder of  $R=1/2$  and  $K=3$

Tuple	State
00	a
01	c
10	b
11	d

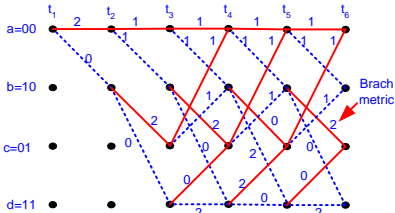


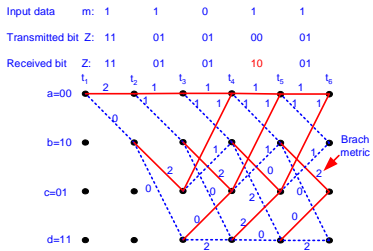
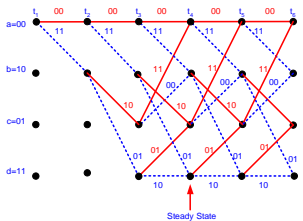


- The trellis diagram is redrawn by labeling each branch with the Hamming distance between received code symbol and the branch word.
- Figure shows a message sequence  $m$  the corresponding codeword sequence  $U$  and a noise corrupted received sequence  $Z$ .
- The branch words of the Trellis are known a priori to both encoder and decoder.
- At time  $t_1$  the received code is 11, state  $00 \Rightarrow 00$  transition with an output of 00 and is compared with received code and its Hamming distance is of 2 and is marked on that branch.
- Similarly at time  $t_1$  the received code is 11, state  $00 \Rightarrow 10$  transition with an output of 11 and is compared with received code and its Hamming distance is of 0 and is marked on that branch.

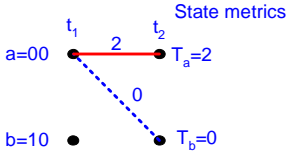


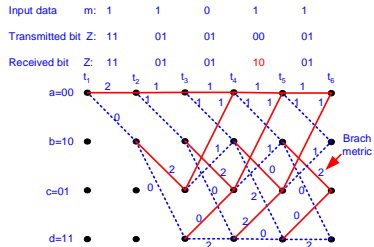
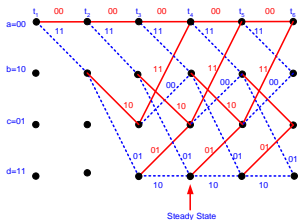
Input data  $m$ : 1 1 0 1 1  
 Transmitted bit  $Z$ : 11 01 01 00 01  
 Received bit  $Z$ : 11 01 01 10 01



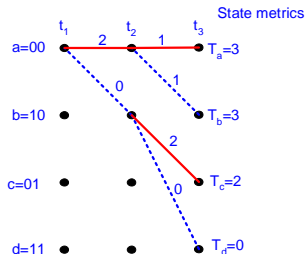
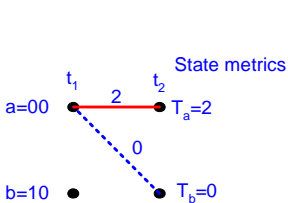


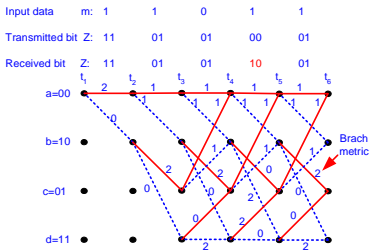
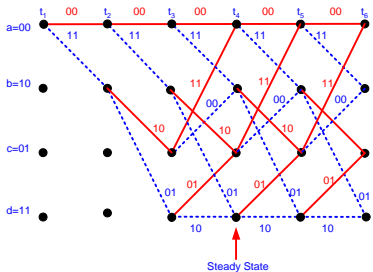
- At each time there are  $2^{K-1}$  states, and at each state two paths are entering (two paths leaving).
- Computing the metrics for the two paths entering each state and eliminating one of them and this is done for each of  $2^{K-1}$  states at time  $t_i$ ; then the decoder moves to time  $t_{i+1}$  and repeats the process.





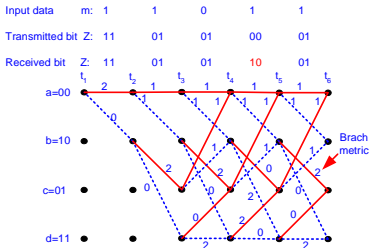
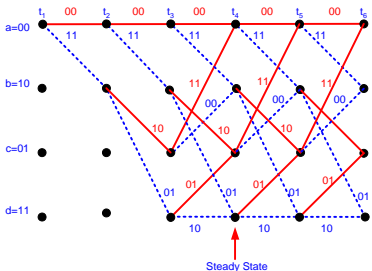
- At each time there are  $2^{K-1}$  states, and at each state two paths are entering (two paths leaving).
- Computing the metrics for the two paths entering each state and eliminating one of them and this is done for each of  $2^{K-1}$  states at time  $t_i$ ; then the decoder moves to time  $t_{i+1}$  and repeats the process.



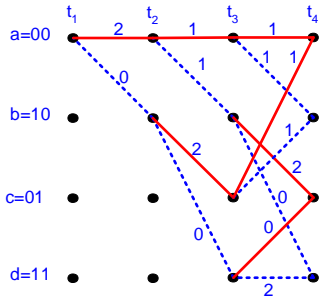


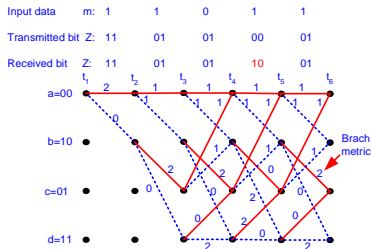
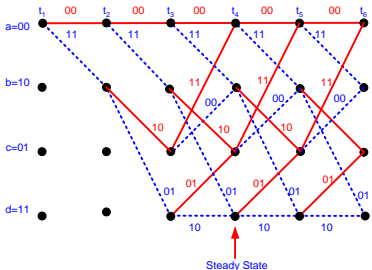
At time  $t_3$  two paths diverging from each state. As a result two paths entering each state time  $t_4$ .



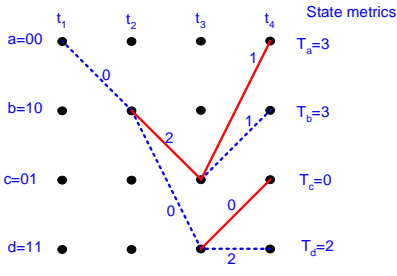
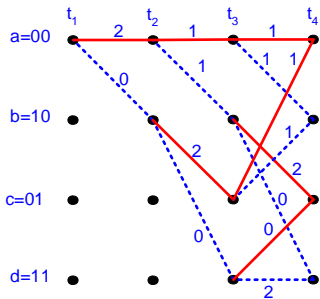


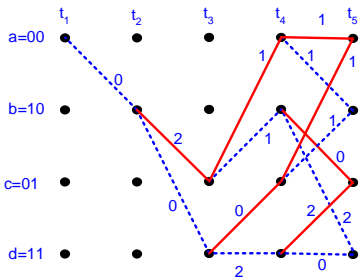
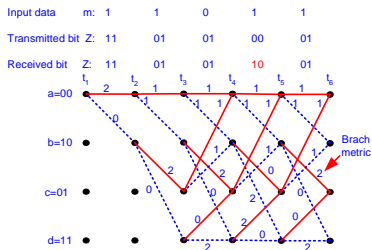
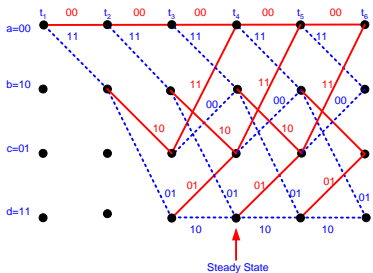
- At time  $t_3$  two paths diverging from each state. As a result two paths entering each state time time  $t_4$ .
- Larger cumulative path metric entering each state can be eliminated. In case if there are two paths having same cumulative path metric, then one path is selected arbitrarily.

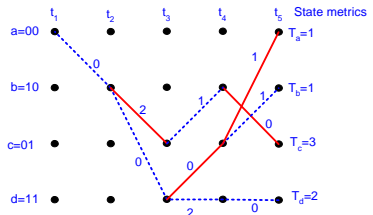
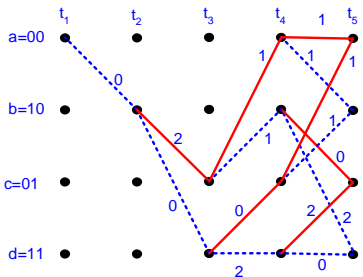
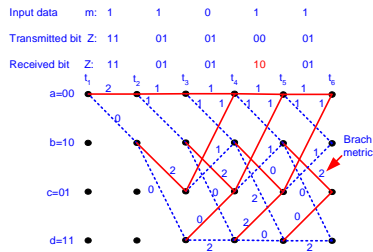
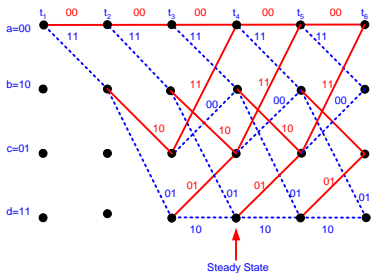




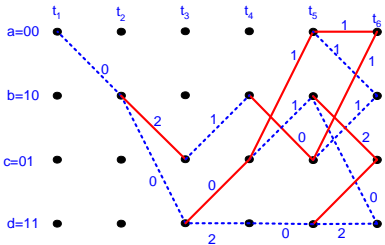
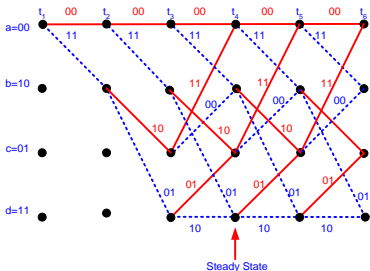
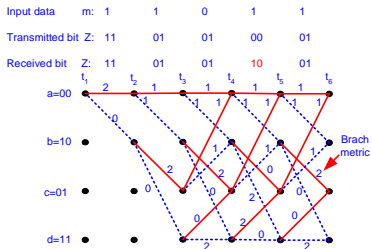
- At time  $t_3$  two paths diverging from each state. As a result two paths entering each state time time  $t_4$ .
- Larger cumulative path metric entering each state can be eliminated. In case if there are two paths having same cumulative path metric, then one path is selected arbitrarily.

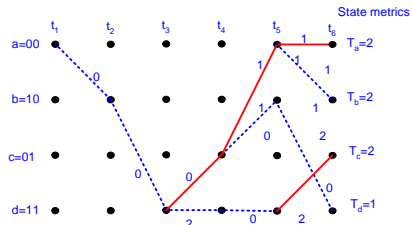
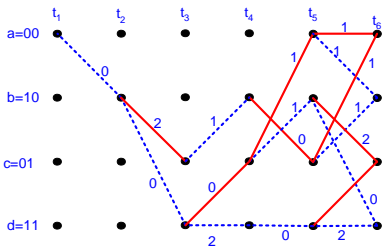
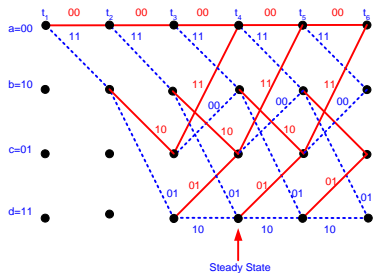
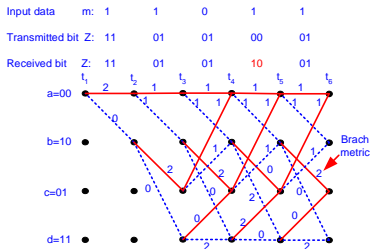












Proakis IV Edition-8-26

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

Table: State transition table for the (3,1,2)encoder

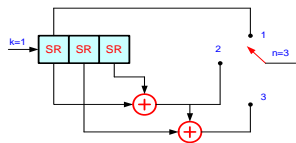
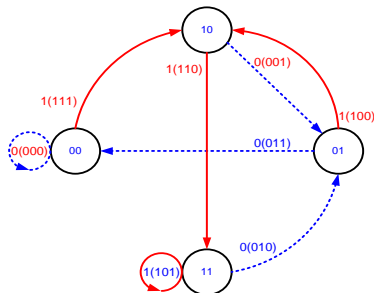
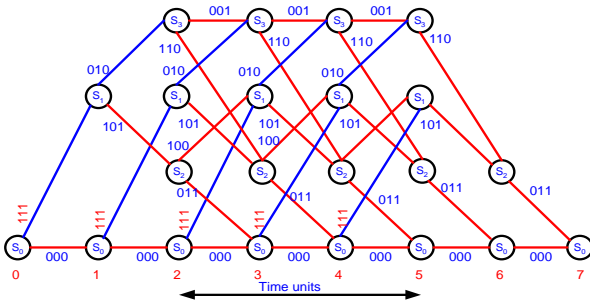
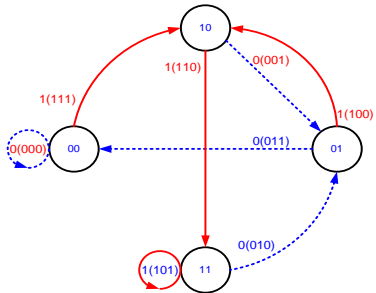


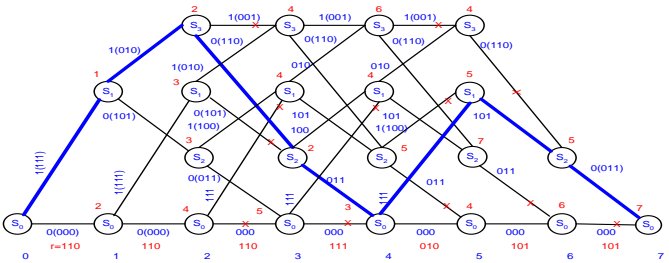
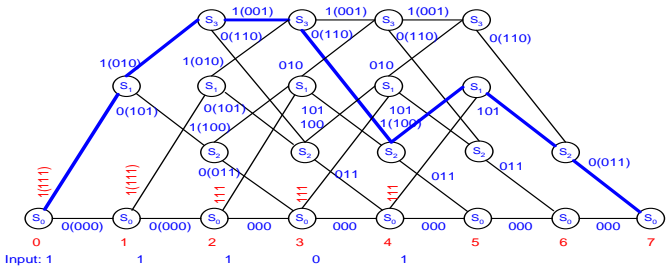
Figure: R=1/3 convolutional encoder with memory m=2

Input	Present State	Next State	Output
0	00	00	000
1	00	10	111
0	01	00	011
1	01	10	100
0	10	01	001
1	10	11	110
0	11	01	010
1	11	11	101

Tuple	State
00	$s_0$
01	$s_2$
10	$s_1$
11	$s_3$







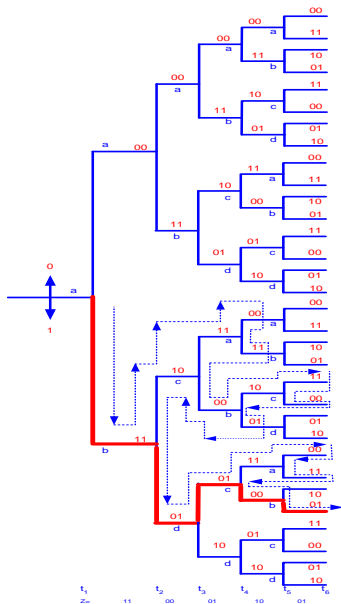
# Sequential Decoding: The Fano Decoding Algorithm



- The decoder starts at the origin node with the threshold  $T=0$  and the metric value  $M=0$ .
- It looks forward to the best of the  $2^k$  succeeding nodes, i.e., with largest metric.
- If  $M_f$  is the metric of the forward node being examined and if  $M_f \geq T$  then the decoder moves to this node.
- It checks for the end of tree has been reached, otherwise threshold tightening is performed if the node is examined for the first time i.e.,  $T$  is increased by the largest multiple of a threshold increment  $\Delta$  so that new threshold does not exceed the current metric.
- If the node has been examined previously, no threshold tightening is performed.
- Then the decoder again looks forward the the best succeeding node.
- If  $M_f < T$ , the decoder looks backward to the preceding node.
- If  $M_b$  is the metric of the backward node being examined, and if  $M_b \leq T$ , then the  $T$  is lowered by  $\Delta$  and the look forward to the best node step is repeated.
- If  $M_b \geq T$ , the decoder moves back to the preceding node.

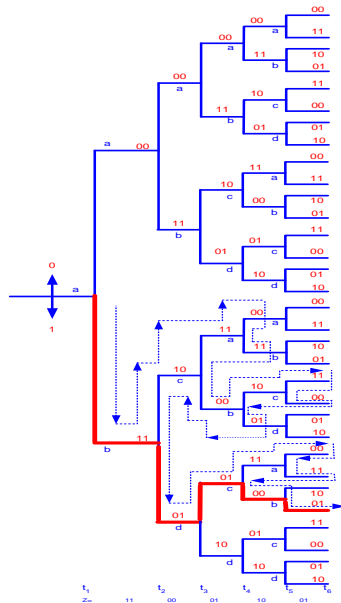


- 1 At time  $t_1$  receives symbol 11 decoder moves downward and decodes as bit 1.





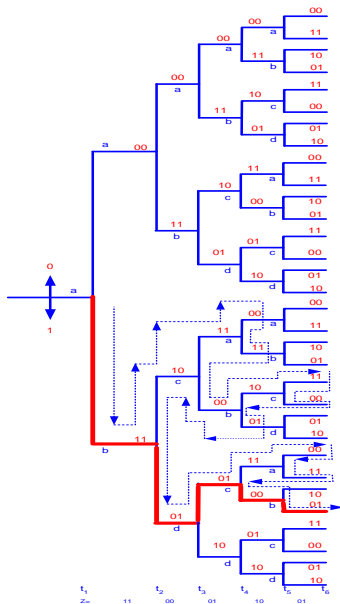
- 1 At time  $t_1$  receives symbol 11 decoder moves downward and decodes as bit 1.
- 2 At time  $t_2$  receives symbol 00 there are two branches 10 and 01 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 1.





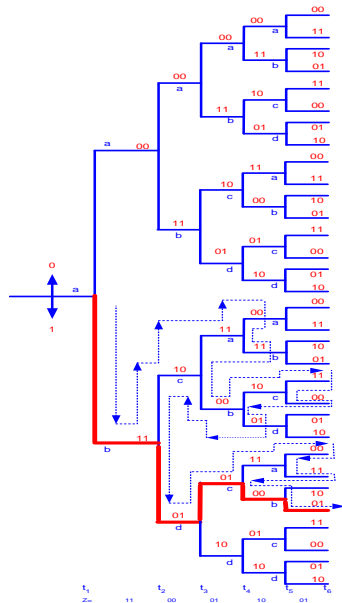


- 1 At time  $t_1$  receives symbol 11 decoder moves downward and decodes as bit 1.
- 2 At time  $t_2$  receives symbol 00 there are two branches 10 and 01 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 1.
- 3 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 2.
- 4 At time  $t_4$  receives symbol 10 there are two branches 00 and 11 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 3.
- 5 Count 3 is turnaround criterion decoder backs out tries alternate path by decrementing count by i. e., count=2

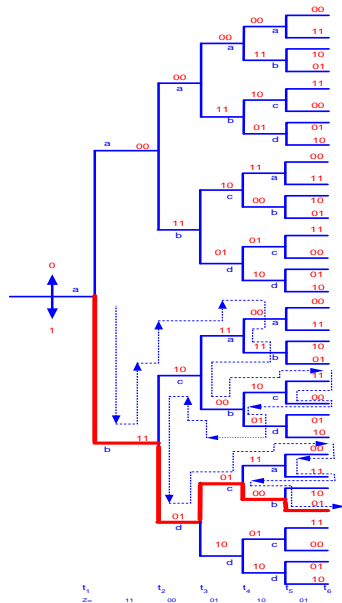




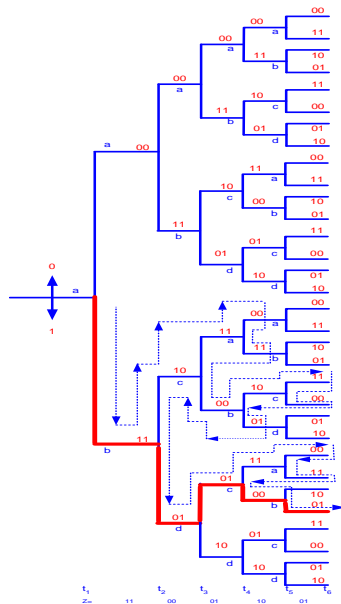
- 1 At time  $t_1$  receives symbol 11 decoder moves downward and decodes as bit 1.
- 2 At time  $t_2$  receives symbol 00 there are two branches 10 and 01 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 1.
- 3 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 2.
- 4 At time  $t_4$  receives symbol 10 there are two branches 00 and 11 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 3.
- 5 Count 3 is turnaround criterion decoder backs out tries alternate path by decrementing count by i. e., count=2
- 6 At time  $t_4$  receives symbol 10 and moves 11 branch decoder moves downward and decodes as bit 0 again it is disagreement with count = 3.
- 7 Count 3 is turnaround criterion all the alternative paths have traversed decoder backs out tries alternate path by decrementing count i. e., count=1 moves to the node at  $t_3$  level.



- 8 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 and untried one is 00 decoder moves downward and decodes as bit 1 with disagreement count = 2.

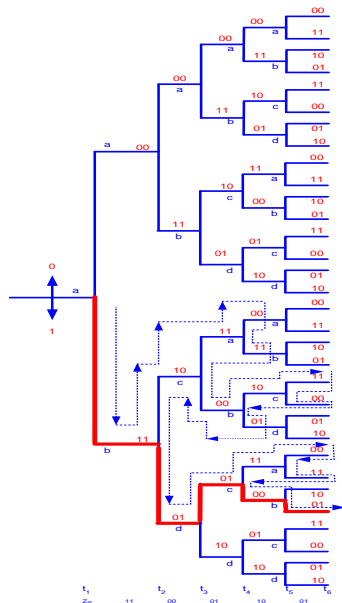


- 8 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 and untried one is 00 decoder moves downward and decodes as bit 1 with disagreement count = 2.
- 9 At time  $t_4$  receives symbol 10 and moves 10 branch decoder moves upward and decodes as bit 0 with agreement with count = 2.

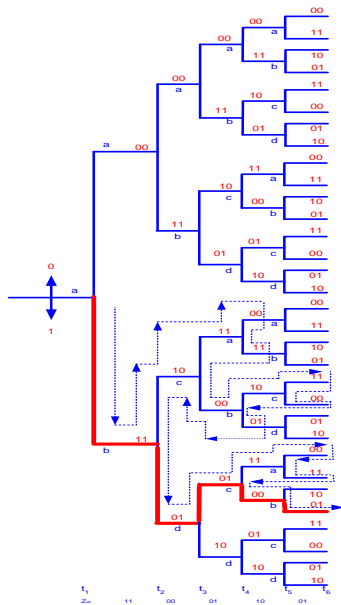




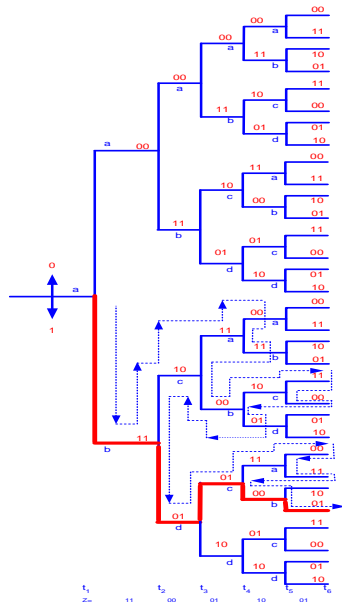
- 8 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 and untried one is 00 decoder moves downward and decodes as bit 1 with disagreement count = 2.
- 9 At time  $t_4$  receives symbol 10 and moves 10 branch decoder moves upward and decodes as bit 0 with agreement with count = 2.
- 10 At time  $t_5$  receives symbol 01 there are two branches 11 and 00 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 3.



- 8 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 and untried one is 00 decoder moves downward and decodes as bit 1 with disagreement count = 2.
- 9 At time  $t_4$  receives symbol 10 and moves 10 branch decoder moves upward and decodes as bit 0 with agreement with count = 2.
- 10 At time  $t_5$  receives symbol 01 there are two branches 11 and 00 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 3.
- 11 At this count decoder back up and reset the counter=2 time  $t_5$  receives symbol 01 there are two branches 11 and 00 and tries alternate path as 00 with disagreement of 1 and again count = 3.

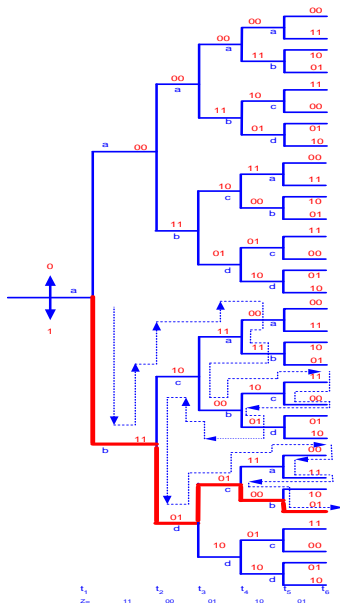


- 8 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 and untried one is 00 decoder moves downward and decodes as bit 1 with disagreement count = 2.
- 9 At time  $t_4$  receives symbol 10 and moves 10 branch decoder moves upward and decodes as bit 0 with agreement with count = 2.
- 10 At time  $t_5$  receives symbol 01 there are two branches 11 and 00 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 3.
- 11 At this count decoder back up and reset the counter=2 time  $t_5$  receives symbol 01 there are two branches 11 and 00 and tries alternate path as 00 with disagreement of 1 and again count = 3.
- 12 The decoder backs out of this path and sets count=2, all the alternate paths have traversed at  $t_5$  level, so decoder returns to node at  $t_4$  and resets count=1





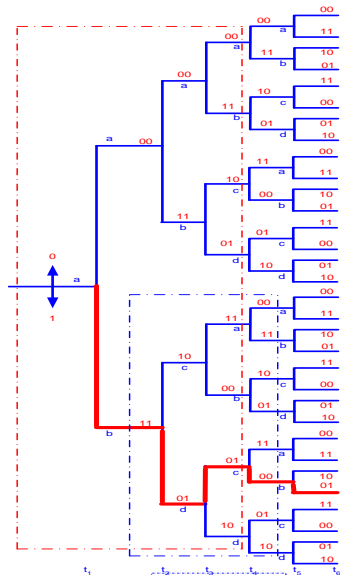
- 8 At time  $t_3$  receives symbol 01 there are two branches 11 and 00 and untried one is 00 decoder moves downward and decodes as bit 1 with disagreement count = 2.
- 9 At time  $t_4$  receives symbol 10 and moves 10 branch decoder moves upward and decodes as bit 0 with agreement with count = 2.
- 10 At time  $t_5$  receives symbol 01 there are two branches 11 and 00 decoder moves upward arbitrarily and decodes as bit 0 with disagreement count = 3.
- 11 At this count decoder back up and reset the counter=2 time  $t_5$  receives symbol 01 there are two branches 11 and 00 and tries alternate path as 00 with disagreement of 1 and again count = 3.
- 12 The decoder backs out of this path and sets count=2, all the alternate paths have traversed at  $t_5$  level, so decoder returns to node at  $t_4$  and resets count=1
- 13 At  $t_4$  with data as 10 and path with 01 makes count=3 decoder back to the time  $t_2$  node
- 14 At time  $t_2$  receives symbol 00 and now follows the branch word 01 with disagreement of 1 with count = 1.



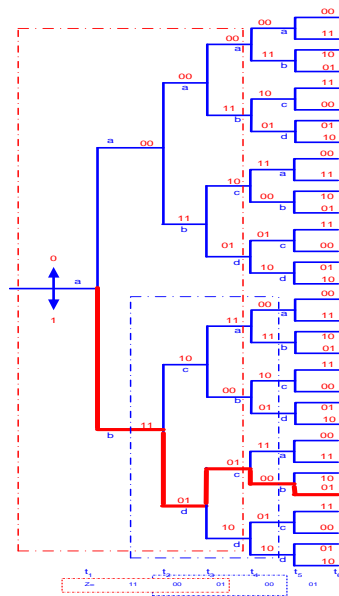
# Feedback Decoding



- Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.

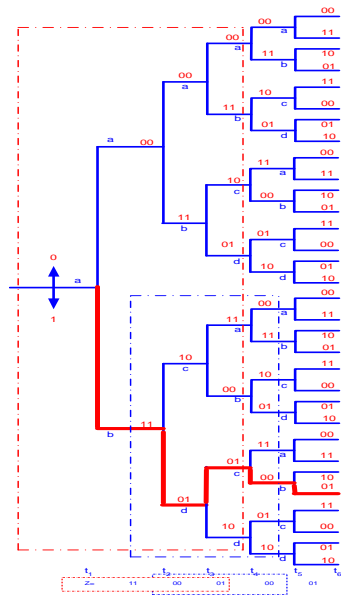


- Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.

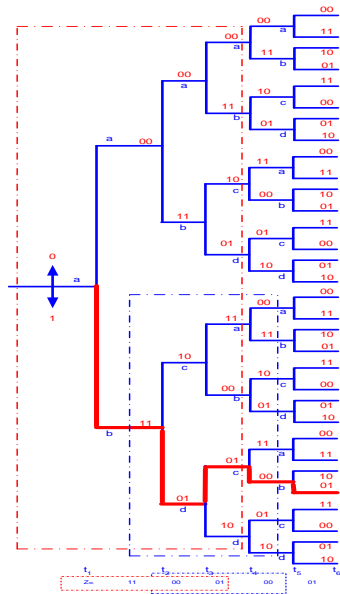




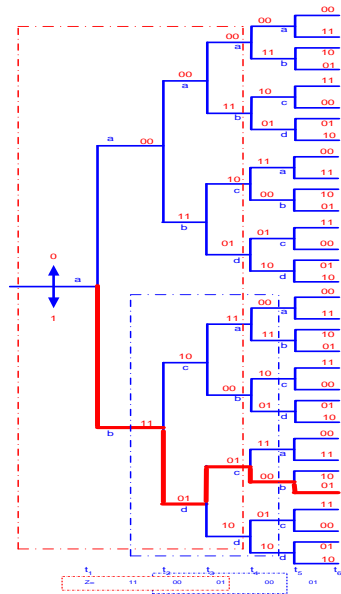
- Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- Upper-half metrics: 3,3,6,4



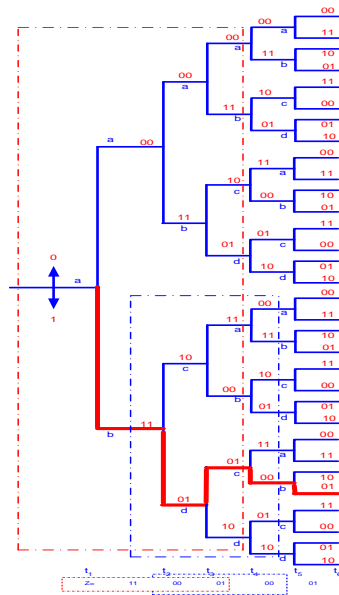
- 1 Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- 2 Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- 3 Upper-half metrics: 3,3,6,4
- 4 Lower-half metrics: 2,2,1,3



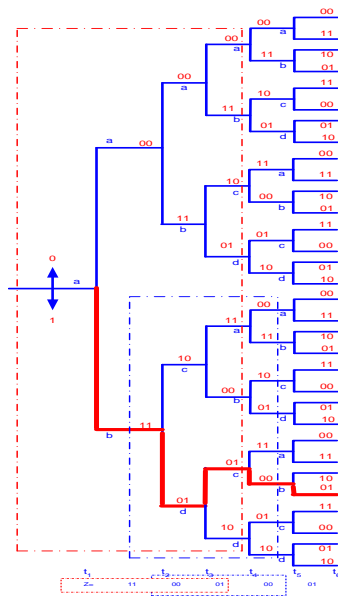
- 1 Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- 2 Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- 3 Upper-half metrics: 3,3,6,4
- 4 Lower-half metrics: 2,2,1,3
- 5 The minimum metric is contained in the lower part of the tree therefore the first coded bit is one.



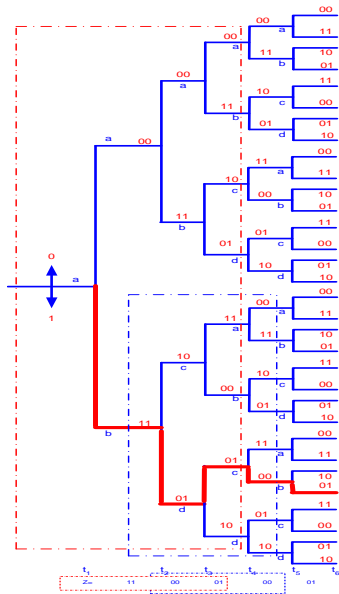
- Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- Upper-half metrics: 3,3,6,4
- Lower-half metrics: 2,2,1,3
- The minimum metric is contained in the lower part of the tree therefore the first coded bit is one.
- The next step is to extend the lower part of the tree.



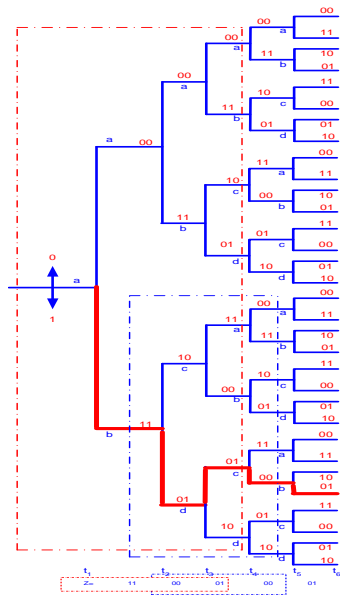
- 1 Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- 2 Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- 3 Upper-half metrics: 3,3,6,4
- 4 Lower-half metrics: 2,2,1,3
- 5 The minimum metric is contained in the lower part of the tree therefore the first coded bit is one.
- 6 The next step is to extend the lower part of the tree.
- 7 In the next step slide over two code symbols.



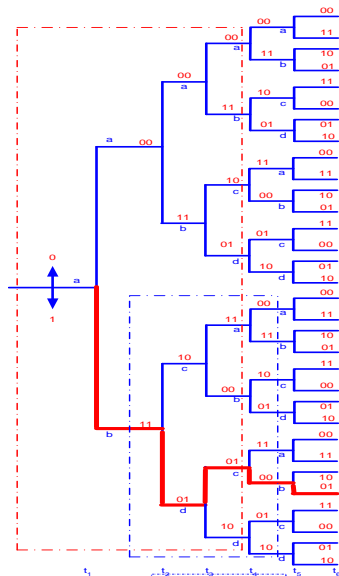
- 1 Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- 2 Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- 3 Upper-half metrics: 3,3,6,4
- 4 Lower-half metrics: 2,2,1,3
- 5 The minimum metric is contained in the lower part of the tree therefore the first coded bit is one.
- 6 The next step is to extend the lower part of the tree.
- 7 In the next step slide over two code symbols.
- 8 Upper-half metrics: 2,4,3,3



- 1 Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- 2 Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- 3 Upper-half metrics: 3,3,6,4
- 4 Lower-half metrics: 2,2,1,3
- 5 The minimum metric is contained in the lower part of the tree therefore the first coded bit is one.
- 6 The next step is to extend the lower part of the tree.
- 7 In the next step slide over two code symbols.
- 8 Upper-half metrics: 2,4,3,3
- 9 Lower-half metrics: 3,1,4,4



- 1 Beginning with the first branch the decoder computes  $2^L$  Hamming path metrics and decides the bit is zero if the minimum distance is in the upper part of the tree and decides the bit is one if the minimum distance is in the lower part of the tree.
- 2 Assuming the received sequence as  $Z=1\ 1\ 0\ 0\ 1\ 0\ 0\ 1$  and examining the eight paths from time  $t_1$  through  $t_3$  and computing the metrics of these eight paths.
- 3 Upper-half metrics: 3,3,6,4
- 4 Lower-half metrics: 2,2,1,3
- 5 The minimum metric is contained in the lower part of the tree therefore the first coded bit is one.
- 6 The next step is to extend the lower part of the tree.
- 7 In the next step slide over two code symbols.
- 8 Upper-half metrics: 2,4,3,3
- 9 Lower-half metrics: 3,1,4,4
- 10 The same procedure is continued until the entire message is decoded.





# Structural Properties of Convolutional Codes



- Graphically, there are three ways to represent convolution encoder, in which we can gain better understanding of its operation. These are:
  - 1 State Diagram
  - 2 Tree Diagram
  - 3 Trellis Diagram



- The state diagram is a graph of the possible states of the encoder and the possible transitions from one state to another.
- The diagram shows the possible state transitions.
- Each circle in the state diagram represents a state.
- At any one time, the encoder resides in one of these states.
- The lines to and from it shows the state transition that are possible as bits arrive.
- Assuming that the encoder is initially in state  $S_0$  (all-zero state), the code word corresponding to any given information sequence can be obtained by following the path through the state diagram and noting the corresponding outputs on the branch labels.
- Following the last nonzero information block, the encoder is return to state  $S_0$  by a sequence of  $m$  all-zero blocks appended to the information sequence.



- It looks like a tree structure with emerging branches hence it is called a trellis.
- A trellis is more instructive than a tree in that it brings out explicitly the fact that the associated convolutional encoder is finite state machine.
- The convention used in Figure to distinguish between input symbol 0 and 1 is as follows.
- A code branch produced by an input 0 is drawn as a solid line, whereas a code branch produced by an input 1 is drawn as a dashed line.
- Each input sequence corresponds to a specific path through the trellis.
- The trellis contains  $(L+K)$  levels, where  $L$  is the length of the incoming message sequence, and  $K$  is the constraint length of the code.
- The levels of the trellis are labeled as  $j=0, 1, 2L+K-1$ . The first  $(K-1)$  levels corresponds to the encoder's departure from the initial state  $a$ , and the last  $(K-1)$  levels corresponds to the encoder's return to the state  $a$ .
- Not all these state can be reached in these two portions of the trellis.
- After the initial transient, the trellis contains four nodes at each stage, corresponding to the four states.
- After the second stage, each node in the trellis has two incoming paths and two outgoing paths.
- Of the two outgoing paths, corresponds to the input bit 0 and the other corresponds input bit 1.



# Signal Flow Graph:

## Signal Flow Graph:

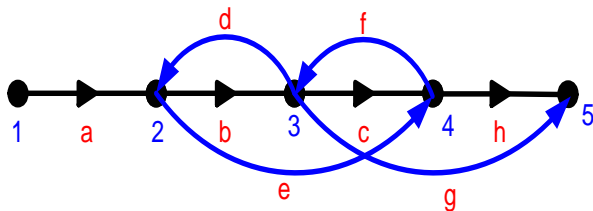
- Any system operation can be represented by a set of linear algebraic equations.
- The algebraic equations represent the relationship between system variables.
- A signal flow graph is a graphical representation of relationship between variables of a set of linear equations.

$$ax_1 + dx_3 = x_2$$

$$bx_2 + fx_4 = x_3$$

$$ex_2 + cx_3 = x_4$$

$$gx_3 + hx_4 = x_5$$



## Basic Terminologies:

- **Node:** Node represents the system variable.
- **Branch:** The line joining  $x_1$  and  $x_2$  forms the branch. Branch  $j$ - $k$  originates at node  $j$  and terminates upon node  $k$ , the direction from  $j$  to  $k$  being indicated by an arrowhead on the branch. Each branch  $j$ - $k$  has associated with it a quantity called the branch gain and each node  $j$  has an associated quantity called the node signal .
- **Source (Input) Node:** A source is a node having only outgoing branches.
- **Sink (output) Node:** A sink is a node having only incoming branches.
- **Path:** A path is any continuous succession of branches traversed in the indicated branch directions.
- **Forward path:** A forward path is a path from source to sink along which no node is encountered more than once .
- **Loop:** A loop is path which originates and terminates at the same node (a closed path without crossing the same point more than once)



- **Non-Touching Loops:** Loops are said to be non-touching if they do not pass through a common node. Or two loops are non-touching or non-interacting if they have no nodes in common.
- **Feedback Loop:** A feedback loop is a path that forms a closed cycle along which each node is encountered once per cycle.
- **Path Gain:** A path gain is the product of the branch gains along that path.
- **Loop gain:** The loop gain of a feedback loop is the product of the gains of the branches forming that loop. The gain of a flow graph is the signal appearing at the sink per unit signal applied at the source.
- To find the graph gain, first locate all possible sets of non-touching loops and write the algebraic sum of their gain products as the denominator of.





## Mason's Gain Formula:

- The general expression for graph gain may be written as

$$G = \frac{\sum_k G_k \Delta_k}{\Delta}$$

- where  $G_k$  = gain of the  $k$ th forward path
- $\Delta = 1 - \sum_m P_{m1} + \sum_m P_{m2} - \sum_m P_{m3} + \dots$
- $P_{mr}$  = Gain product of the  $m$ th possible combination of non touching loops
- $\Delta = 1 - \sum$  loop gains +  $\sum$  combination of two non touching loops -  $\sum$  combination of 3 non touching loops .....
- $\Delta_k$  = The value of  $\Delta$  for that part of the graph not touching the  $k$ th forward path



## The Transfer Function of a Convolutional Code:

- For every convolutional code, the transfer function gives information about the various paths through the trellis that start from the all-zero state and return to this state for the first time. According to the coding convention described before, any code word of a convolutional encoder corresponds to a path through the trellis that starts from the all-zero state and returns to the all-zero state.
- The performance of a convolutional code depends on the **distance properties of the code**. The **free distance** of convolutional code is defined as the minimum **Hamming distance** between any two code words in the code. A convolutional code with **free distance  $d_{\text{free}}$**  can correct  $t$  errors if and only if  **$d_{\text{free}}$  is greater than  $2t$** .
- **The free distance can be obtained from the state diagram of the convolutional encoder.**
- Any nonzero code sequence corresponds to a complete path beginning and ending at the all-zero state.
- **The state diagram is modified to a graph called as signal flow graph with a single input and a single output.**



- A signal flow graph consists of nodes and directed branches and it operates by the following rules:
  - 1 A branch multiplies the signal at its input node by the transmittance characterizing that branch.
  - 2 A node with incoming branches sums the signals produced by all of these branches.
  - 3 The signal at a node is applied equally to all the branches outgoing from that node.
  - 4 The transfer function of the graph is the ratio of the output of the signal to the input signal
- The branches of the graph is labeled as  $D^0 = 1, D^1, D^2, D^3$  where the exponent of D denotes the Hamming distance between the sequence of output bits corresponding to each branch and the sequence of output bits corresponding to the all-zero branch.



- There is one path which departs from all zeros at time  $t_1$  and merges with zeros at time  $t_4$  with a distance of 5.
- There are two paths of distance 6 one departs from all zeros at time  $t_1$  and merges with zeros at time  $t_5$  and another departs from all zeros at time  $t_1$  and merges with zeros at time  $t_6$ .

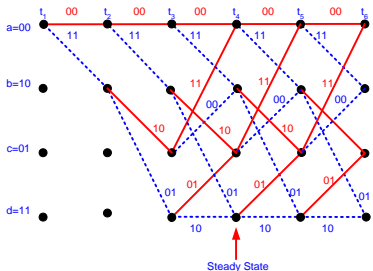


Figure: Trellis diagram

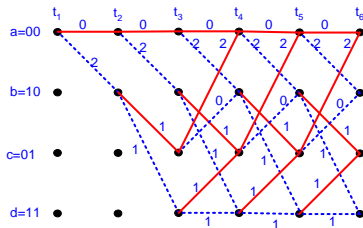


Figure: Trellis diagram showing distance from all zeros path



- Branches of the state diagram are labeled as  $D$ ,  $D^1$  or  $D^2$  where the  $D$  denotes the Hamming distance from the branch word of that branch to the all zeros branch.
- Node **a** is split into two nodes labeled as **a** and **e** one which represents the input and the other output of the state diagram.
- All paths originating at  $a = 00$  and terminating terminating at  $e = 00$

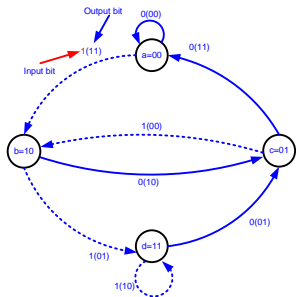


Figure: State diagram

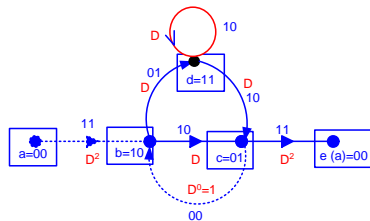


Figure: Signal flow graph

$$X_b = D^2 X_a + X_c$$

$$X_c = D X_b + D X_d$$

$$X_d = D X_b + D X_d$$

$$X_e = D^2 X_c$$



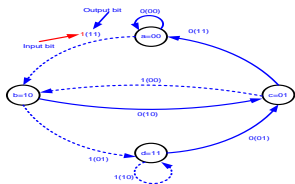


Figure: State diagram

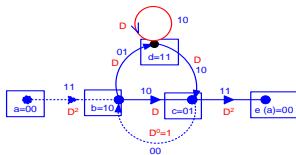


Figure: Signal flow graph

There are two forward paths

Gain of the forward path (abc) is:  $P_1 = (D^2 * D * D^2) = (D^5)$

Gain of the forward path (abdce) is:  $P_2 = (D^2 * D * D * D^2) = (D^6)$

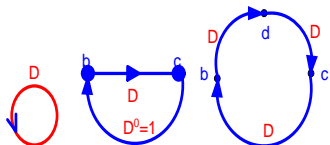


Figure: Loops in a signal flow graph

$$\Delta = 1 - (L_1 + L_2 + L_3) + (L_1 L_2) = 1 - (D + D + D^3) + (D^2)$$

$$\Delta = 1 - 2D - D^3 + D^3 = 1 - 2D$$

$$\Delta_1 = 1 - (L_1) = 1 - D \quad \Delta_2 = 1$$

$$G = \frac{\sum_k G_k \Delta_k}{\Delta} T(D) = \frac{P_1 \Delta_1 + P_2 \Delta_2}{\Delta}$$

$$= \frac{D^5(1 - D) + D^6(1)}{1 - 2D^2} = \frac{D^5 - D^6 + D^6}{1 - 2D} = \frac{D^5}{1 - 2D}$$

There are three loops in a graph:

Gain of the self loop at (d) is:  $L_1 = D$

Gain of the loop for (cb) is:  $L_2 = (D * 1) = D$

Gain of the loop for (cdb) is:  $L_3 = (D * D * D) = D^3$

$$T(D) = D^5 + 2D^6 + 4D^7 + \dots + 2^l D^{l+5}$$

[Use Binomial expansion:  $(1-x)^{-1} = 1+x+x^2+x^3+x^4+\dots$ ]

- There is one path at distance 5 from the all zeros path, that departs from the all zeros path at time  $t_1$  and merges with it at time  $t_4$
- Similarly there are two paths at distance 6 one which departs at time  $t_1$  and merges with it at time  $t_5$ , and other which departs at time  $t_1$  and merges with it at time  $t_6$  and so on.
- In general there are  $2^l$  paths of distance  $l + 5$  from the all zeros path where  $l = 0, 1, 2, \dots$
- The minimum distance in the set of all arbitrarily long paths that diverge and remerge, called the **minimum free distance** or simply the **free distance**. It is of 5 in this example.
- The error correcting capability of the code with the free distance  $d_f$  as

$$t = \left\lfloor \frac{d_f - 1}{2} \right\rfloor$$

$$t = \left\lfloor \frac{5 - 1}{2} \right\rfloor = 2$$

- The encoder can correct any two channel errors.

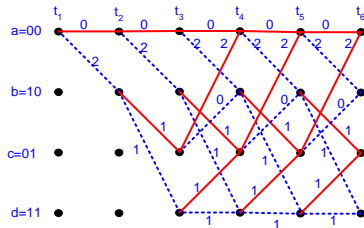


Figure: Trellis diagram showing distance from all zeros path



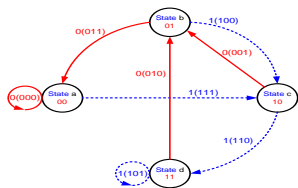


Figure: State diagram

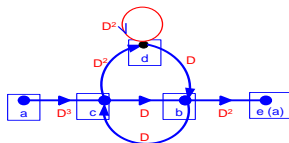


Figure: Signal flow graph

There are two forward paths

Gain of the forward path (acbe) is:  $P_1 = (D^3 * D * D^2) = (D^6)$

Gain of the forward path (acde) is:  $P_2 = (D^3 * D^2 * D * D^2) = (D^8)$

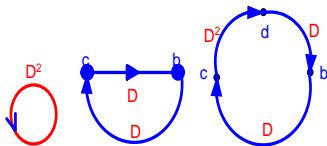


Figure: Loops in a signal flow graph

There are three loops in a graph:

Gain of the self loop at (d) is:  $L_1 = D^2$

Gain of the loop for (cb) is:  $L_2 = (D * D) = D^2$

Gain of the loop for (cdb) is:  $L_3 = (D * D^2 * D) = D^4$

$$\Delta = 1 - (L_1 + L_2 + L_3) + (L_1 L_2) = 1 - (D^2 + D^2 + D^4) + (D^4)$$

$$\Delta = 1 - 2D^2 - D^4 + D^4 = 1 - 2D^2$$

$$\Delta_1 = 1 - (L_1) = 1 - D^2 \quad \Delta_2 = 1$$

$$G = \frac{\sum_k G_k \Delta_k}{\Delta} T(D) = \frac{P_1 \Delta_1 + P_2 \Delta_2}{\Delta}$$

$$= \frac{D^6(1 - D^2) + D^8(1)}{1 - 2D^2} = \frac{D^6 - D^8 + D^8}{1 - 2D^2} = \frac{D^6}{1 - 2D^2}$$

$$T(D) = D^6 + 2D^8 + 4D^{10} + 8D^{12}$$

[Use Binomial expansion:  $(1-x)^{-1} = 1+x+x^2+x^3+x^4+\dots$ ]





- The weight distribution function of a code can be determined by considering the modified state diagram as a signal flow graph and applying Mason's gain formula to compute its generating function

$$A(X) = \sum_d A_d X^d$$

- where  $A_i$  is the number of code words of weight  $i$ .
- In a signal flow graph, a path connecting the initial state to the final state which does not go through any state twice is called a forward path.
- A closed path starting at any state and returning to that state without going through any other state twice is called a loop.



- Let  $C_i$  be the gain of the  $i$ th loop.
- A set of loops is nontouching if no state belongs to more than one loop in the set.
- Let  $\{i\}$  be the set of all loops,  $\{i, j\}$  be the set of all pairs of nontouching loops,  $\{i'', j'', l''\}$  be the set of all triples of nontouching loops, and so on.

$$\Delta = 1 - \sum_i C_i + \sum_{i'j'} C_{i'} C_{j'} - \sum_{i''j''l''} C_{i''} C_{j''} C_{l''} +$$

- where  $\sum_i C_i$  is the sum of the loop gains,  $\sum_{i'j'} C_{i'} C_{j'}$  is the product of the loop gains of two nontouching loops summed over all pairs of nontouching loops,  $\sum_{i''j''l''} C_{i''} C_{j''} C_{l''}$  is the product of the loop gains of three nontouching loops summed over all nontouching loops.

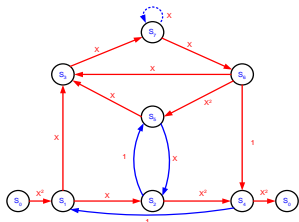


- And  $\Delta_i$  is defined exactly like  $\Delta$ , but only for that portion of the graph not touching the  $i$ th forward path; that is, all states along the  $i$ th forward path, together with all branches connected to these states, are removed from the graph when computing  $\Delta_i$ .
- Mason's formula for computing the generating function  $T(X)$  of a graph can now be stated as

$$A(X) = \frac{\sum_i F_i \Delta_i}{\Delta}$$

- where the sum in the numerator is over all forward paths and  $F_i$  is the gain of the  $i$ th forward path.





There are 11 loops in the state diagram.

Cycle 1:  $S_1 S_3 S_7 S_6 S_5 S_2 S_4 S_1 (C_1 = X^8)$

Cycle 2:  $S_1 S_3 S_7 S_6 S_4 S_1 (C_2 = X^3)$

Cycle 3:  $S_1 S_3 S_6 S_5 S_2 S_4 S_1 (C_3 = X^7)$

Cycle 4:  $S_1 S_3 S_6 S_4 S_1 (C_4 = X^2)$

Cycle 5:  $S_1 S_2 S_5 S_2 S_4 S_7 S_6 S_4 S_1 (C_5 = X^4)$

Cycle 6:  $S_1 S_2 S_3 S_6 S_4 S_1 (C_6 = X^3)$

Cycle 7:  $S_1 S_2 S_4 S_1 (C_7 = X^3)$

Cycle 8:  $S_2 S_5 S_2 (C_8 = X)$

Cycle 9:  $S_3 S_7 S_6 S_5 S_3 (C_9 = X^5)$

Cycle 10:  $S_3 S_6 S_5 S_3 (C_{10} = X^4)$

Cycle 11:  $S_7 S_7 (C_{11} = X)$

There are 10 pairs of nontouching loops:

- Cycle pair 1: (Cycle 2, Cycle 8):  $(C_2 C_8 = X^4)$
- Cycle pair 2: (Cycle 3, Cycle 11):  $(C_3 C_{11} = X^8)$
- Cycle pair 3: (Cycle 4, Cycle 8):  $(C_4 C_8 = X^3)$
- Cycle pair 4: (Cycle 4, Cycle 11):  $(C_4 C_{11} = X^3)$
- Cycle pair 5: (Cycle 6, Cycle 11):  $(C_6 C_{11} = X^4)$
- Cycle pair 6: (Cycle 7, Cycle 9):  $(C_7 C_9 = X^8)$
- Cycle pair 7: (Cycle 7, Cycle 10):  $(C_7 C_{10} = X^7)$
- Cycle pair 8: (Cycle 7, Cycle 11):  $(C_7 C_{11} = X^4)$
- Cycle pair 9: (Cycle 8, Cycle 11):  $(C_8 C_{11} = X^2)$
- Cycle pair 10: (Cycle 10, Cycle 11):  $(C_{10} C_{11} = X^5)$

There are two triples of nontouching loops:

- Cycle triple 1: (Cycle 4, Cycle 8, Cycle 11):  $(C_4 C_8 C_{11} = X^4)$
- Cycle triple 2: (Cycle 7, Cycle 10, Cycle 11):  $(C_7 C_{10} C_{11} = X^8)$

There are no other sets of nontouching loops.

$$\Delta = 1 - (X^8 + X^3 + X^7 + X^2 + X^4 + X^3 + X^3 + X^5 + X^4 + X) + (X^4 + X^8 + X^3 + X^4 + X^8 + X^7 + X^4 + X^2 + X^5) - (X^4 + X^8)$$



- Forward paths 1 and 5 touch all states in the graph, and hence the sub graph not touching these paths contains no states. Therefore

$$\Delta_1 = 1$$

- The subgraph not touching forward paths 3 and 6:

$$\Delta = 1 - X$$

- The subgraph not touching forward path 2:

$$\Delta_2 = 1 - X$$

- The subgraph not touching forward path 4:

$$\Delta_4 = 1 - (X + X) + (X^2) = 1 - 2X + X^2$$

- The subgraph not touching forward path 7:

$$\Delta_7 = 1 - (X + X^4 + X^5) + (X^5) = 1 - X - X^4$$

$$\begin{aligned} T(X) &= \frac{X^{12} \cdot 1 + X^7(1 - X) + X^{11}(1 - X) + X^6(1 - 2X + X^2) + X^8 \cdot 1 + X^7(1 - X) + X^7(1 - X - X^4)}{1 - 2X - X^3} \\ &= \frac{X^6 + X^7 - X^8}{1 - 2X - X^3} = X^6 + 3X^7 + 5X^8 + 11X^9 + 25X^{10} + \dots \end{aligned}$$



- There are 10 pairs of nontouching loops :

Cycle pair 1: (Cycle 2, Cycle 8):  $(C_2 C_8 = X^4)$

Cycle pair 2: (Cycle 3, Cycle 11):  $(C_3, C_{11} = X^8)$

Cycle pair 3: (Cycle 4, Cycle 8):  $(C_4 C_{11} = X^3)$

Cycle pair 4: (Cycle 4, Cycle 11):  $(C_4 C_{11} = X^3)$

Cycle pair 5: (Cycle 6, Cycle 11):  $(C_6 C_{11} = X^4)$

Cycle pair 6: (Cycle 7, Cycle 9):  $(C_7 C_9 = X^8)$

Cycle pair 7: (Cycle 7, Cycle 10):  $(C_7 C_{10} = X^7)$

Cycle pair 8: (Cycle 7, Cycle 11):  $(C_7 C_{11} = X^4)$

Cycle pair 9: (Cycle 8, Cycle 11):  $(C_8 C_{11} = X^2)$

Cycle pair 10: (Cycle 10, Cycle 11):  $(C_{10} C_{11} = X^5)$

- There are two triples of nontouching loops :

- Cycle triple 1: (Cycle 4, Cycle 8, Cycle 11):  $(C_4 C_8 C_{11} = X^4)$

- Cycle triple 2: (Cycle 7, Cycle 10, Cycle 11):  $(C_7 C_{10} C_{11} = X^8)$

- There are no other sets of nontouching loops. Therefore,

$$\Delta = 1 - (X^8 + X^3 + X^7 + X^2 + X^4 + X^3 + X^3 + X^5 + X^4 + X) + (X^4 + X^8 + X^3 + X^4 + X^8 + X^7 + X^4 + X^2 + X^5) - (X^4 + X^8)$$

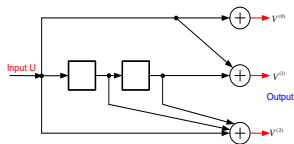
**Problem 9.41 Proakis** A convolutional code is described by

$$g_1 = [1 \ 0 \ 0] \quad g_2 = [1 \ 0 \ 1] \quad g_3 = [1 \ 1 \ 1]$$

- Draw the encoder corresponding.
- Draw the state-transition diagram.
- Draw the trellis diagram.
- Find the transfer function and the free distance.
- If the received sequence is  $r = (110, 110, 110, 111, 010, 101, 101)$ , using the Viterbi algorithm find the transmitted bit sequence.

**Solution:**

a.



b.

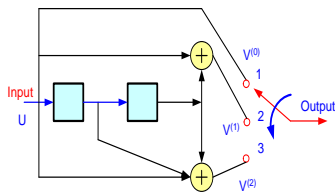
$$G = \begin{bmatrix} 111 & 101 & 011 & & & & & \\ & 111 & 101 & 011 & & & & \\ & & 111 & 101 & 011 & & & \\ & & & 111 & 101 & 011 & & \\ & & & & 111 & 101 & 011 & \end{bmatrix}$$

c. No of rows for  $G$  are 5. The code word corresponding to the information sequence  $u = (1 \ 1 \ 1 \ 0 \ 1)$  is:

$$V = \mathbf{UG} = (1 \ 1 \ 1 \ 0 \ 1) \begin{bmatrix} 111 & 101 & 011 & & & & & \\ & 111 & 101 & 011 & & & & \\ & & 111 & 101 & 011 & & & \\ & & & 111 & 101 & 011 & & \\ & & & & 111 & 101 & 011 & \end{bmatrix} = (111 \ 010 \ 001 \ 110 \ 100 \ 101 \ 011 \ 000)$$



Table: State transition table for the (3,1,2)encoder



Input	Present State	Next State	Output
0	00	00	000
1	00	10	111
0	01	00	011
1	01	10	100
0	10	01	001
1	10	11	110
0	11	01	010
1	11	11	101

Tuple	State
00	a
01	b
10	c
11	d

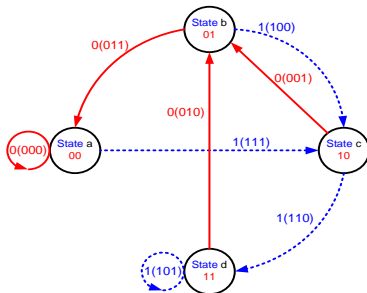


Figure: State diagram for rate 1/3 m=2 convolutional code (3,1,2)





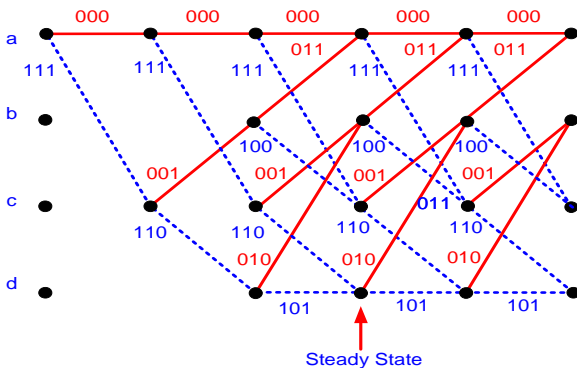
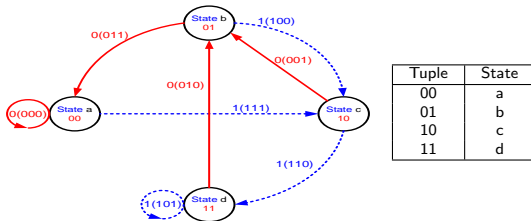


Figure: Trellis diagram for rate 1/3  $m=2$  convolutional code (3,1,2)

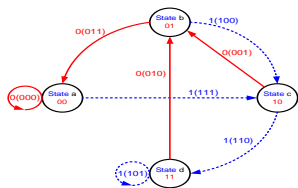


Figure: State diagram

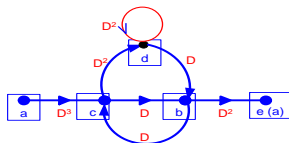
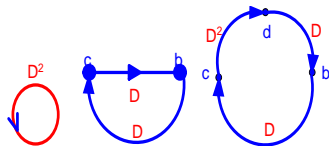


Figure: Signal flow graph

There are two forward paths

Gain of the forward path (acbe) is:  $P_1 = (D^3 * D * D^2) = (D^6)$

Gain of the forward path (acde) is:  $P_2 = (D^3 * D^2 * D * D^2) = (D^8)$



$$\Delta = 1 - (L_1 + L_2 + L_3) + (L_1 L_2) = 1 - (D^2 + D^2 + D^4) + (D^4)$$

$$\Delta = 1 - 2D^2 - D^4 + D^4 = 1 - 2D^2$$

$$\Delta_1 = 1 - (L_1) = 1 - D^2 \quad \Delta_2 = 1$$

$$G = \frac{\sum_k G_k \Delta_k}{\Delta} T(D) = \frac{P_1 \Delta_1 + P_2 \Delta_2}{\Delta}$$

$$= \frac{D^6(1 - D^2) + D^8(1)}{1 - 2D^2} = \frac{D^6 - D^8 + D^8}{1 - 2D^2} = \frac{D^6}{1 - 2D^2}$$

Figure: Loops in a signal flow graph

There are three loops in a graph:

Gain of the self loop at (d) is:  $L_1 = D^2$

Gain of the loop for (cb) is:  $L_2 = (D * D) = D^2$

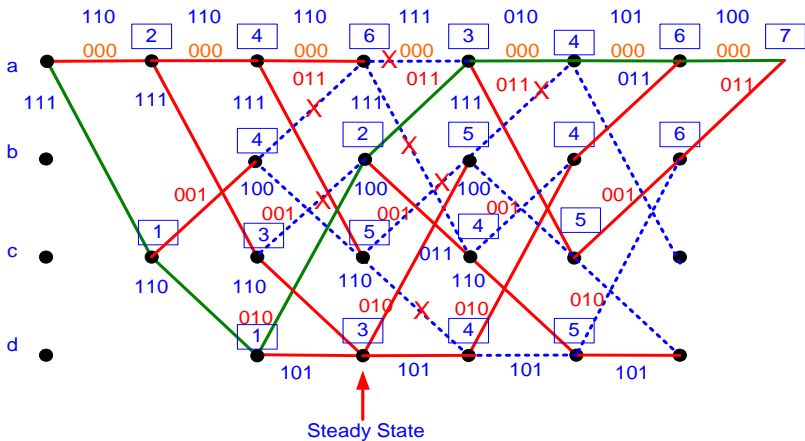
Gain of the loop for (cdb) is:  $L_3 = (D * D^2 * D) = D^4$

$$T(D) = D^6 + 2D^8 + 4D^{10} + 8D^{12}$$

Hence,  $d_{free} = 6$

[Use Binomial expansion:  $(1 - x)^{-1} = 1 + x + x^2 + x^3 + x^4 + \dots$ ]





The path traced by the Viterbi is (111, 110, 010, 011, 000, 000, 000) and the decoded sequence is (1 1 0 0 0 0 0)



## systematic and Non-systematic codes

Systematic convolution code for rate,  $1/2, K=3$  is as follows

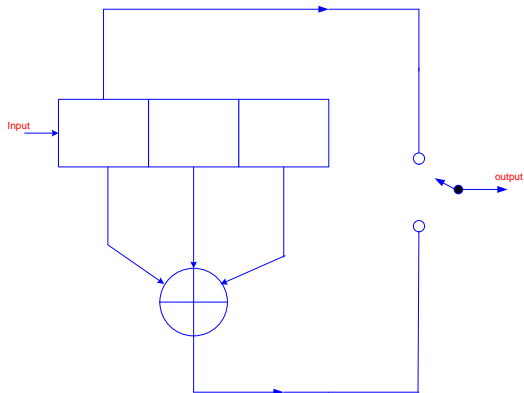


Figure: Systematic convolution code



- A systematic convolutional code is one in which the input  $k$ -tuple appears as part of the output branch  $n$ -tuple.
- As shown in the figure it has the binary rate  $1/2, K=3$  systematic encoder.
- $(n-k)$  inputs should be same as input.
- For Linear Block codes, any non systematic code transformed into a systematic code with the same block distance properties. but this is not in the case of convolution codes.
- The reason for this is the convolution codes depend largely on free distance;
- In general, reduces the maximum possible free distance for a given constraint length and rate.
- the following table shows the maximum free distance for the rate  $1/2$  systematic and nonsystematic codes for  $k=2$  through 8.



## compensation of symmetric and non-symmetric free distance rate 1/2

constraint length	Free distance systematic	Free distance non systematic
2	3	3
3	4	5
4	5	6
6	6	7
7	6	10
8	7	10



## catastrophic Error propagation in convolution codes

- A catastrophic error is an event where by a finite number of code symbol errors causes an infinite number of decoded data bit errors.
- Massey and sain has derived a sufficient and necessary condition for convolution codes to display catastrophic error propagation.
- The condition for the catastrophic error propagation is that have a common polynomial factor,for example it is illustrated in the following example of rate 1/2,K=3.with upper polynomial and lower polynomial as follows

$$g_1(X) = 1 + x$$

$$g_2(X) = 1 + x^2$$

the generator  $g_1$  and  $g_2$  have common polynomial factors  $1+X$  since

$$1 + X^2 = (1 + X) + (1 + X)$$



The following figure shows the catastrophic error propagation.

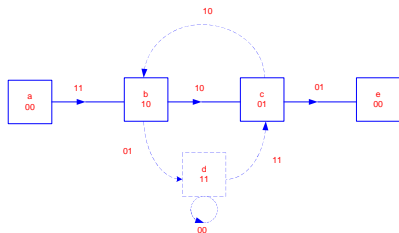
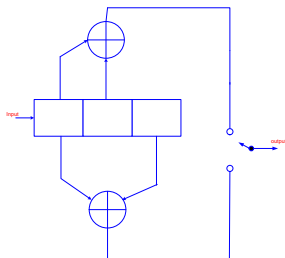


Figure: Catastrophic Error propagation





- From the state diagram for any rate code, catastrophic error occur if and only if any closed-loop path in the diagram has zero weight (zero distance from all the zero path)
- Assume that the all zero path is correct path and incorrect path is a,b,d,d,...d,c, has exactly 6 ones. so number of error occur is proportional to the number of times is self loop at the position d.

## coding Gain

The basic formula to compute the coding gain is given by

$$G(db) = \left(\frac{E_b}{N_0}\right)_u (db) - \left(\frac{E_b}{N_0}\right)_c$$

Where  $\left(\frac{E_b}{N_0}\right)_u$  and  $\left(\frac{E_b}{N_0}\right)_c$ ,  $\left(\frac{E_b}{N_0}\right)$  required for uncoded and coded respectively








- The following table list an upper bound for on the coding gains, compared to uncoded coherent BPSK for several maximum free distance convolutional codes with constraint length varying from 3 to 9 over a channel with hard decision coding.
- The coding gain is cannot increase indefinitely; it has an upper bound as shown in the table, and this is in decibel and given by

$$\text{coding gain} \leq \log_{10}(rd_f)$$

where  $r$  is the code rate and  $d_f$  is the free distance



# References

-  S. Lin and D. J. C. Jr., *Error Control Coding*, 2nd ed. Pearson / Prentice Hall, 2004.
-  R. Blahut, *Theory and Practice of Error Control Codes*, 2nd ed. Addison Wesley, 1984.
-  J. G. Proakis, *Digital communications*, 4th ed. Prentice Hall, 2001.
-  J. G. Proakis and M. Salehi, *Communication Systems Engineering*, 2nd ed. Prentice Hall, 2002.
-  S. Haykin, *Digital communications*, 2nd ed. Wiley, 1988.

