Sistemas de Telecomunicações

7ª Aula

Universidade do Minho
Escola de Engenharia

Outline

- Basic coding techniques:
  - Error detection and Error correction;
  - Block codes;
  - Hamming codes;
  - Cyclic codes;
  - Convolutional codes.

Convolutional Codes

José Cabral
Departamento de Electrónica Industrial
Escola de Engenharia
Universidade do Minho

Introduction

- In this type of coding the encoder output is not in block form
- Encoded sequence is generated from an input information sequence
- The redundancy in the encoded sequence is used by the corresponding decoder to infer the message sequence by performing error correction

Introduction

- Unlike in block coding, the n elements that form the encoded segment do not depend only on the segment of k elements that are input at a given instant i
- They depend also on the previous segments input at instants $i - 1, i - 2, \ldots, i - K$
  - K is the memory of the encoder
- The higher the level of memory, the higher the complexity of the convolutional decoder, and the stronger the error correction capability of C_{conv}

Introduction

- A convolutional code with parameters n, k and K will be denoted as $C_{conv}(n, k, K)$
- They are constructed by using basic memory units, or delays, combined with adders and scalar multipliers
- These linear sequential circuits are also known as finite state sequential machines (FSSMs)
- The number of memory units, or delays, defines the level of memory of a given convolutional code $C_{conv}(n, k, K)$, determining also its error-correction capability

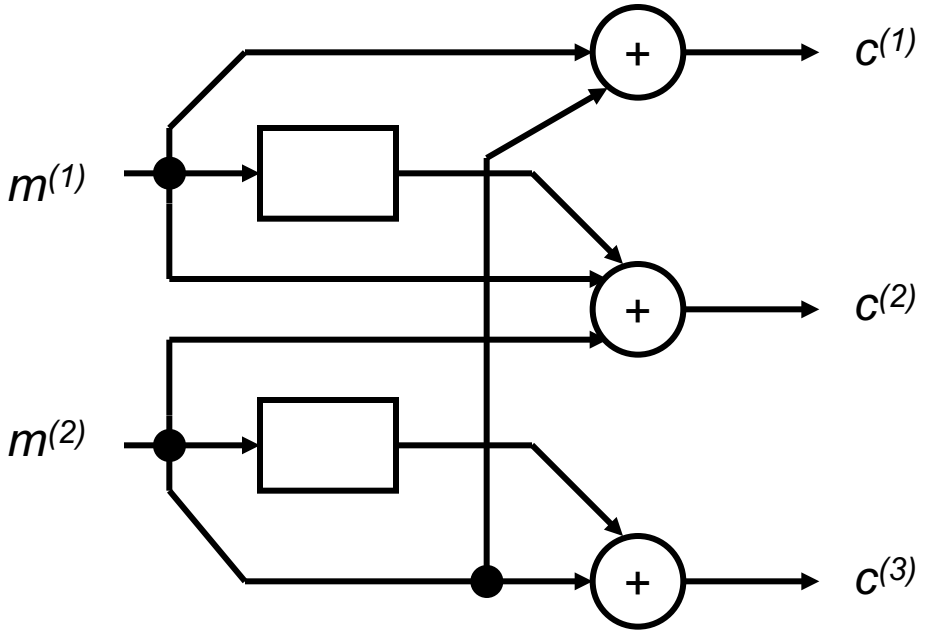
Introduction

- FSSM analysis is usually performed by means of a rational transfer function
  - $G(D) = P(D)/Q(D)$
- of polynomial expressions in the D domain, called the delay domain, where message and code sequences adopt the polynomial form $M(D)$ and $C(D)$, respectively
- For multiple input – multiple output FSSMs, the relationship between the message sequences and the code sequences is described by $G(D)$

Convolutional Codes and Encoders

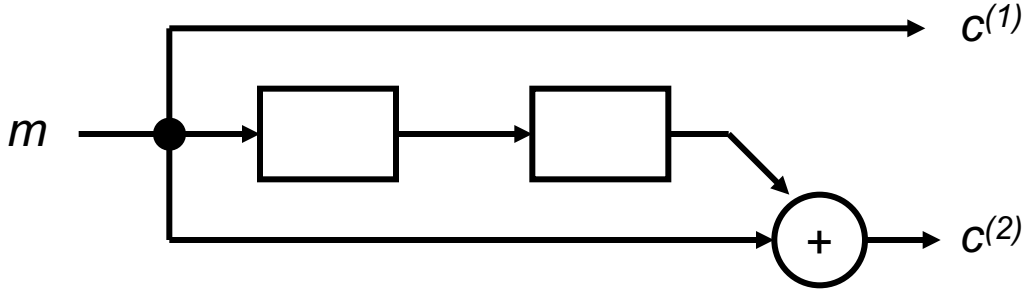
- A convolutional encoder takes a *k*-tuple m_i of message elements as the input
- Generates the *n*-tuple c_i of coded elements as the output at a given instant *I*
- Coded elements depends not only on the input *k*-tuple m_i of the message at instant *i* but also on previous *k*-tuples, m_j present at instants $j < i$

Examples of Codes and Encoders



$n = 3$
$k = 2$
$K = 1$

$R_c = k/n = 2/3$

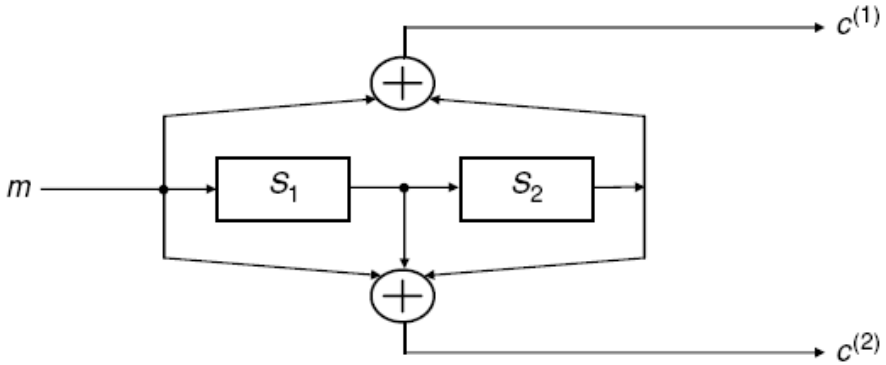


$n = 2$
$k = 1$
$K = 2$

$R_c = ½$

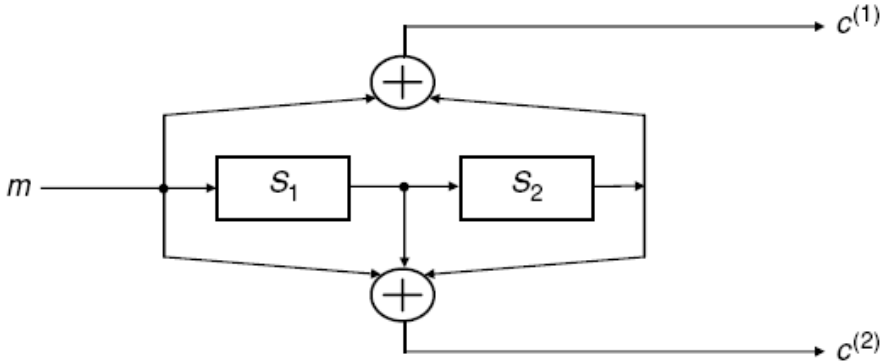
Systematic encoder – WHY?

Convolutional Codes and Encoders



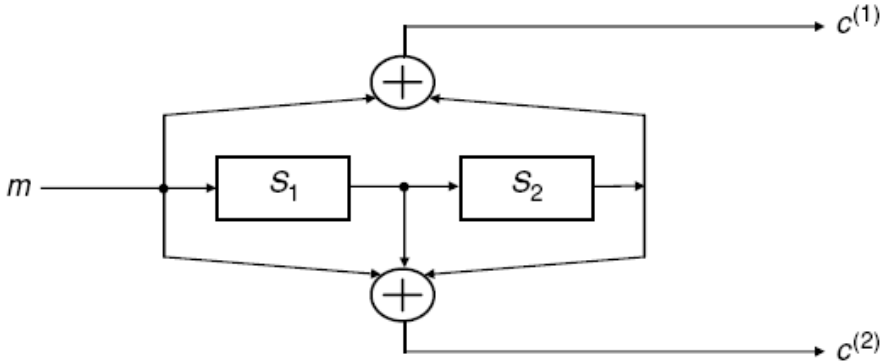
- Input sequence:
  - $m_i = (m_0, m_1, m_2, \ldots)$
- Output sequences:
  - $c^{(1)} = (c^{(1)}_0, c^{(1)}_1, c^{(1)}_2, \ldots)$
  - $c^{(2)} = (c^{(2)}_0, c^{(2)}_1, c^{(2)}_2, \ldots)$

Outputs and Impulse response



- The 2 output sequences can be obtained as the convolution between the input sequence and the 2 impulse responses of the encoder defined for each of its outputs
- Impulse responses can be obtained by applying the unit impulse input sequence m = (1, 0, 0, ...) and observing the resulting outputs $c_i^{(1)}$ and $c_i^{(2)}$

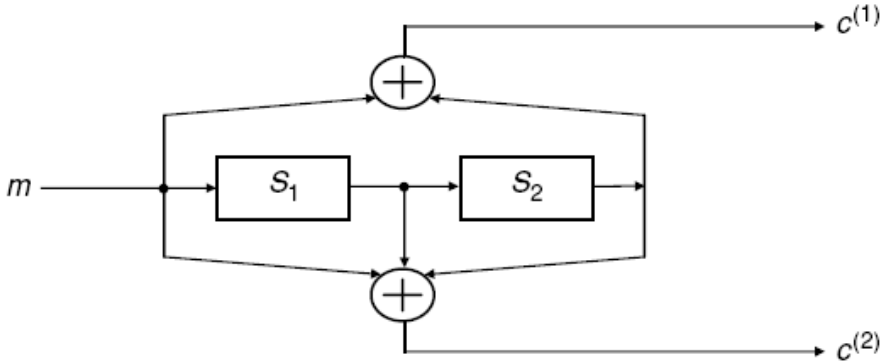
Outputs and Impulse response



Instant	$m\ S_1\ S_2$	$c^{(1)}$	$c^{(2)}$
t_1	1 0 0	1	1
t_2	0 1 0	0	1
t_3	0 0 1	1	1

Input response = 11 01 11

Outputs and Impulse response

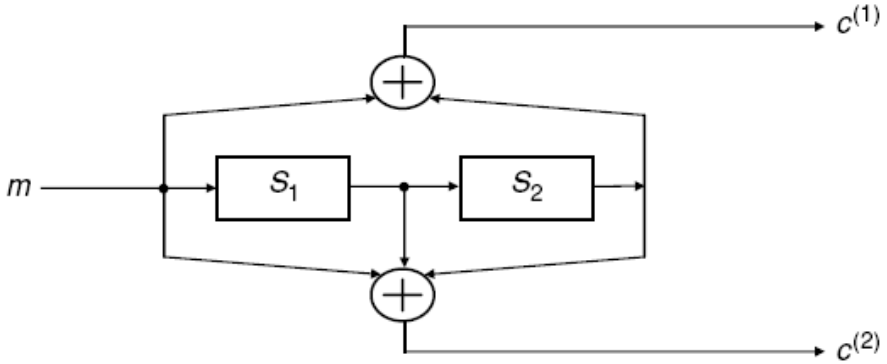


m = (1101)

Instant	Input	Output
t_1	1	11 01 11
t_2	1	11 01 11
t_3	0	00 00 00
t_4	1	11 01 11

Output: $c^{(1)}\, c^{(2)}$ = 11 10 10 00 01 11

Outputs and Impulse response



$\boldsymbol{g}^{(1)} = (1\ 0\ 1)$

$\boldsymbol{g}^{(2)} = (1\ 1\ 1)$

- The value $K + 1$ (3 for this example) is called the constraint length of the convolutional code C_{conv}
  - is the maximum number of time units that a given bit of the input sequence can influence the output sequence values
- If the input is the unit impulse, then:
  - $c^{(1)} = g^{(1)}$ and $c^{(2)} = g^{(2)}$

Impulse response

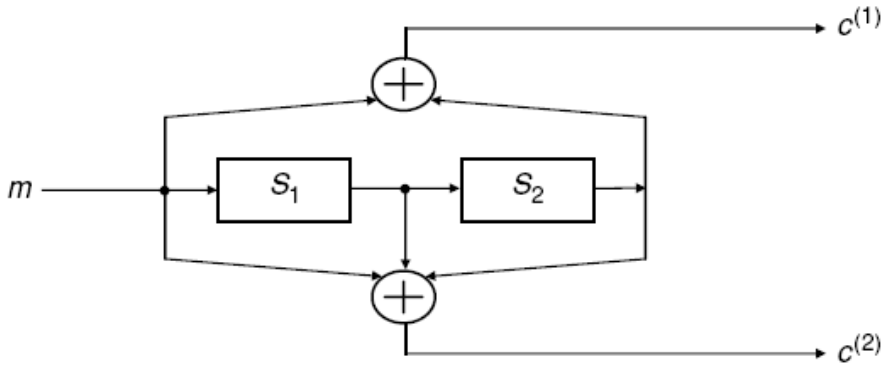
- These vectors describe the impulse responses of the FSSM and they are also a description of the connections of the structure of the FSSM
- When a given memory unit is connected to an output, the corresponding bit in the impulse response vector is ‘1’, whereas for an absent connection this bit is ‘0’
- The impulse responses are also known as the generator sequences of the C_{conv}

D-Transform Domain

- In the field of signals and their spectra, convolution in the time domain becomes multiplication in the spectral domain
- This suggests a better description of convolutional codes based on expressions given in the *D-transform domain*
- *D-transform domain* ⇔ Delay domain *D*
- The sequences adopt a polynomial form expressed in the variable D

D-Transform Domain

- Delay D can be interpreted as a shift parameter, and it plays the same role as the term Z^{-1} in the Z transform



$\boldsymbol{g}^{(1)} = (1\ 0\ 1)$

$\boldsymbol{g}^{(2)} = (1\ 1\ 1)$

- $C^{(1)}(D) = M(D).G^{(1)}(D) = M(D)(1 + D^2)$
- $C^{(2)}(D) = M(D).G^{(2)}(D) = M(D)(1 + D + D^2)$

D-Transform Domain

- For more general structures where there is more than 1 input and more than 1 output, the relationship between input *i* and output *j* is given by the corresponding transfer function $G_i^{(j)}(D)$
- In this input-to-output path the number of delays or memory units D is called the length of the register
- This number is equal to the degree of the corresponding generator polynomial for such a path

D-Transform Domain

- In a more general FSSM structure for which there are *k* inputs and *n* outputs, there will be *kn* transfer functions that can be arranged in matrix form as:

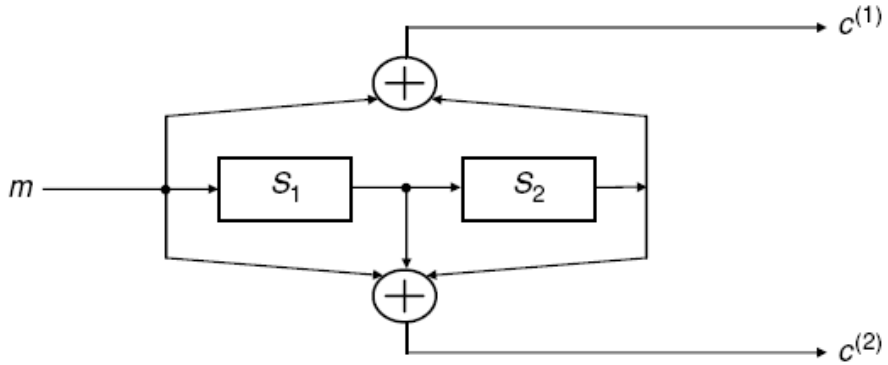
$$\boldsymbol{G}(D) = \begin{bmatrix} G_1^{(1)}(D) & G_1^{(2)}(D) & \cdots & G_1^{(n)}(D) \\ G_2^{(1)}(D) & G_2^{(2)}(D) & \cdots & G_2^{(n)}(D) \\ \vdots & \vdots & & \vdots \\ G_k^{(1)}(D) & G_k^{(2)}(D) & \cdots & G_k^{(n)}(D) \end{bmatrix}$$

D-Transform Domain

- A convolutional code $C_{conv}(n, k, K)$ produces an output sequence expressed in polynomial form as:
  - $\boldsymbol{C}(D) = \boldsymbol{M}(D)\boldsymbol{G}(D)$
- where:
  - $M(\mathrm{D}) = M^{(1)}(\mathrm{D}), M^{(2)}(\mathrm{D}), \ldots, M^{(k)}(\mathrm{D})$
- and:
  - $C(\mathrm{D}) = C^{(1)}(\mathrm{D}), C^{(2)}(\mathrm{D}), \ldots, C^{(n)}(\mathrm{D})$

Example

- For the convolutional code $C_{conv}(2, 1, 2)$



$\boldsymbol{g}^{(1)} = (1\ 0\ 1)$

$\boldsymbol{g}^{(2)} = (1\ 1\ 1)$

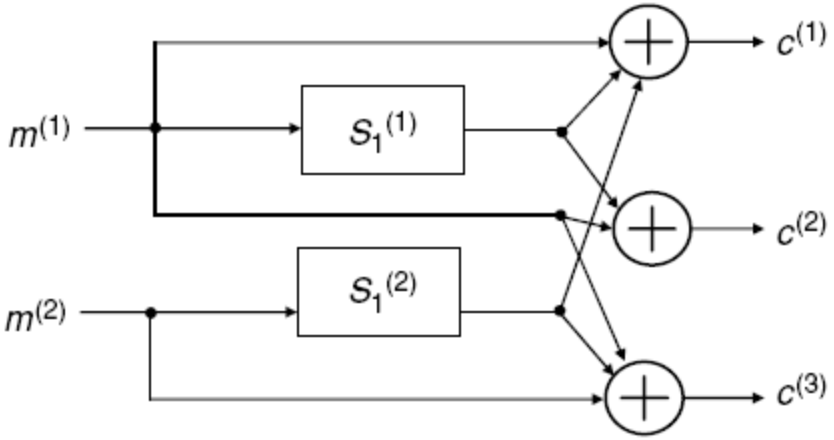
- Determine the polynomial expression of the output for the input sequence (100011)

Example

- The input sequence in polynomial form is
  - $M(D) = 1 + D^4 + D^5$
- The generator matrix is of the form
  - $G(D) = [1 + D^2 \quad 1 + D + D^2]$
- Then
  - $C(D) = (C^{(1)}(D)\ C^{(2)}(D)) = [1+D^4+D^5]\ [1+D^2 \quad 1+D+D^2]$
  - $C(D) = [1+D^2+D^4+D^5+D^6+D^7 \quad 1+D+D^2+D^4+D^7]$
  - $c^{(1)} = (10101111)$; $c^{(2)} = (11101001)$
- The output sequence is
  - $c = (11, 01, 11, 00, 11, 10, 10, 11)$

Another example

- For the encoder of the $C_{conv}(3, 2, 1)$



- Expressions for the generator polynomials in the D domain are:
  - $G_1^{(1)}(D) = 1+D$; $G_1^{(2)}(D) = 1+D$; $G_1^{(3)}(D) = 1$
  - $G_2^{(1)}(D) = D$; $G_2^{(2)}(D) = 0$; $G_2^{(3)}(D) = 1+D$

Another example

- Thus, and if the input vector is for instance equal to $m^{(1)}$ = (1 0 1) and $m^{(2)}$ = (0 1 1)
- Then

$$M^{(1)}(D) = 1 + D^2 \qquad M^{(2)}(D) = D + D^2$$

$$\begin{aligned} C^{(1)}(D) &= M^{(1)}(D)G_1^{(1)}(D) + M^{(2)}(D)G_2^{(1)}(D) \\ &= (1 + D^2)(1 + D) + (D + D^2)D \\ &= 1 + D \end{aligned}$$

$$\begin{aligned} C^{(2)}(D) &= M^{(1)}(D)G_1^{(2)}(D) + M^{(2)}(D)G_2^{(2)}(D) \\ &= (1 + D^2)(1 + D) + (D + D^2)0 \\ &= 1 + D + D^2 + D^3 \end{aligned}$$

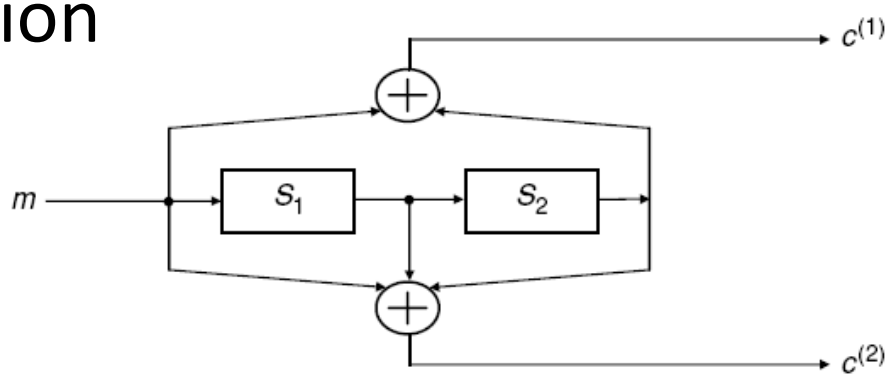
$$\begin{aligned} C^{(3)}(D) &= M^{(1)}(D)G_1^{(3)}(D) + M^{(2)}(D)G_2^{(3)}(D) \\ &= (1 + D^2)(1) + (D + D^2)(1 + D) \\ &= 1 + D + D^2 + D^3 \end{aligned}$$

c = (111, 111, 011, 011)

Convolutional Encoder Representations

- There are different forms to describe a convolutional encoder:
  - Representation of Connections
  - State Diagram Representation
  - Trellis Representation
  - State Transfer Function

This example of encoder will be used ->



Representation of Connections

- One form of describing a $C_{conv}(n, k, K)$ is by means of a vector description of the connections that the FSSM has
- Directly described by the vector representation of the generator polynomials $g^{(1)}$ and $g^{(2)}$ that correspond to the upper and lower branches of the FSSM
- In this description, a '1' means that there is connection, and a '0' means that the corresponding register is not connected

Representation of Connections

- For a given input sequence, this code description can provide the corresponding output sequence. This can be seen by implementing a table

$\boldsymbol{g}^{(1)} = (1\ 0\ 1)$

$\boldsymbol{g}^{(2)} = (1\ 1\ 1)$

$m = (100011)$

$c = (110111001110101 1)$

Input m_i	State at t_i	State at t_{i+1}	$c^{(1)}$	$c^{(2)}$
–	0 0	0 0	–	–
1	0 0	1 0	1	1
0	1 0	0 1	0	1
0	0 1	0 0	1	1
0	0 0	0 0	0	0
1	0 0	1 0	1	1
1	1 0	1 1	1	0
0	1 1	0 1	1	0
0	0 1	0 0	1	1
0	0 0	0 0	0	0

Convolutional Encoder Representations

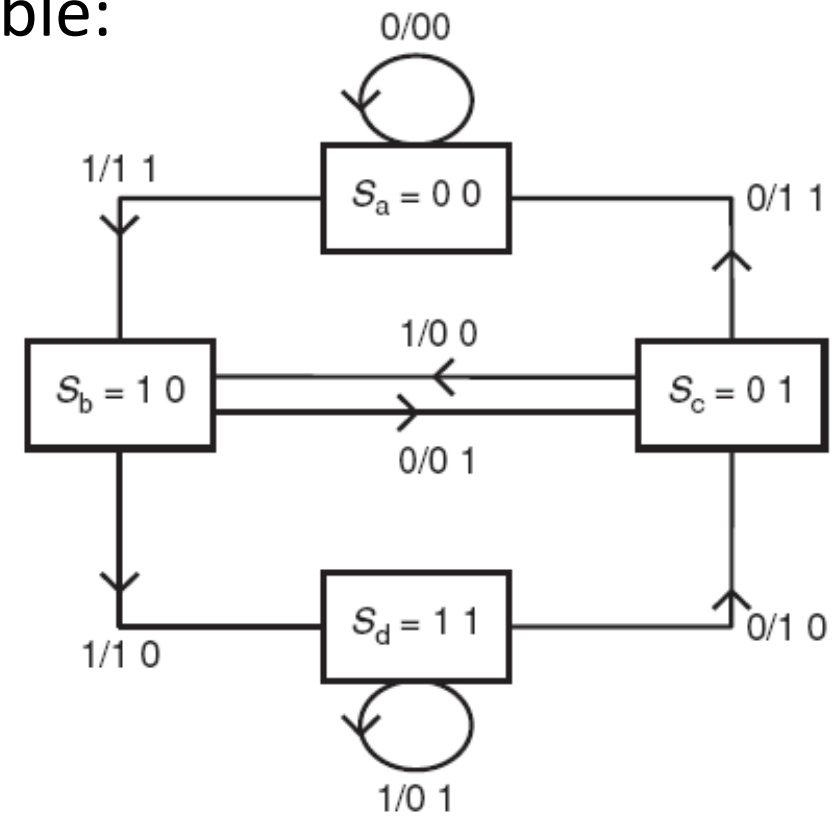
- Another table is a useful tool for constructing the corresponding state diagram

Input m_i	State at t_i	State at t_{i+1}	$c^{(1)}$	$c^{(2)}$
–	0 0	0 0	–	–
0	0 0	0 0	0	0
1	0 0	1 0	1	1
0	0 1	0 0	1	1
1	0 1	1 0	0	0
0	1 0	0 1	0	1
1	1 0	1 1	1	0
0	1 1	0 1	1	0
1	1 1	1 1	0	1

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

State Diagram Representation

- The state diagram representation is obtained by the previous table:



State Diagram Representation

- There are only two transitions emerging from a given state
- If the FSSM is in a given state, it is not possible to go to any other state in an arbitrary manner
- This sort of memory will be useful in determining that some transitions are not allowed in the decoded sequence, thus assisting the decisions required for error correction

Trellis Representation

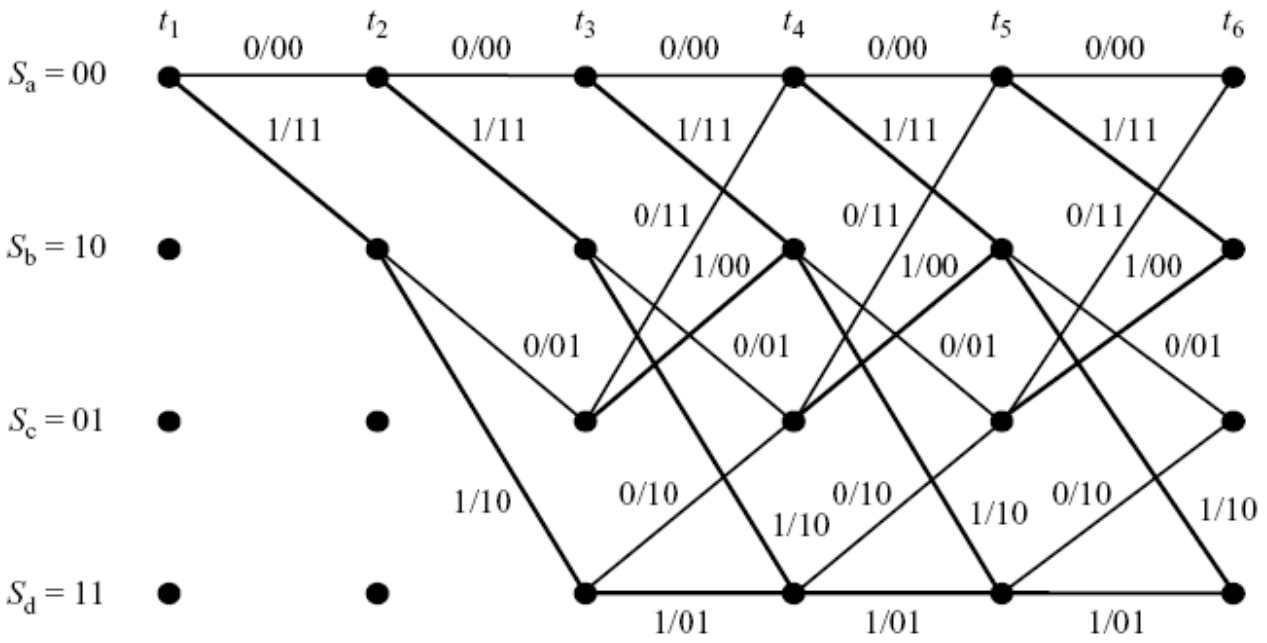
- A way of representing systems based on FSSMs is the tree diagram (*see references*)
- This representation is useful to indicate the start of the state sequence, but however the repetitive structure of state evolution is not clearly presented
- One of the interesting characteristics of the state evolution of a convolutional code is precisely that after K + 1 initial transitions, the state structure becomes repetitive, where K + 1 is the *constraint length of the code*

Trellis Representation

- State diagram clearly shows the repetitive structure of the state evolution of a convolutional code, but it is not clear for describing the initial evolution
- A representation that clearly describes these two different parts of the state structure of a given convolutional code is the so-called trellis diagram
- This trellis diagram is a state versus time instant representation
- There are two branches emerging from and arriving at a given state, which correspond to transitions produced by the two possible inputs to the FSSM

Trellis Representation

- There are 2^K possible states in this diagram
- The state structure becomes repetitive after time instant t_4



Convolutional Codes in Systematic Form

- In a systematic code, message information can be seen and directly extracted from the encoded information
- In the case of a convolutional code, the transfer function for a systematic convolutional code is of the form:

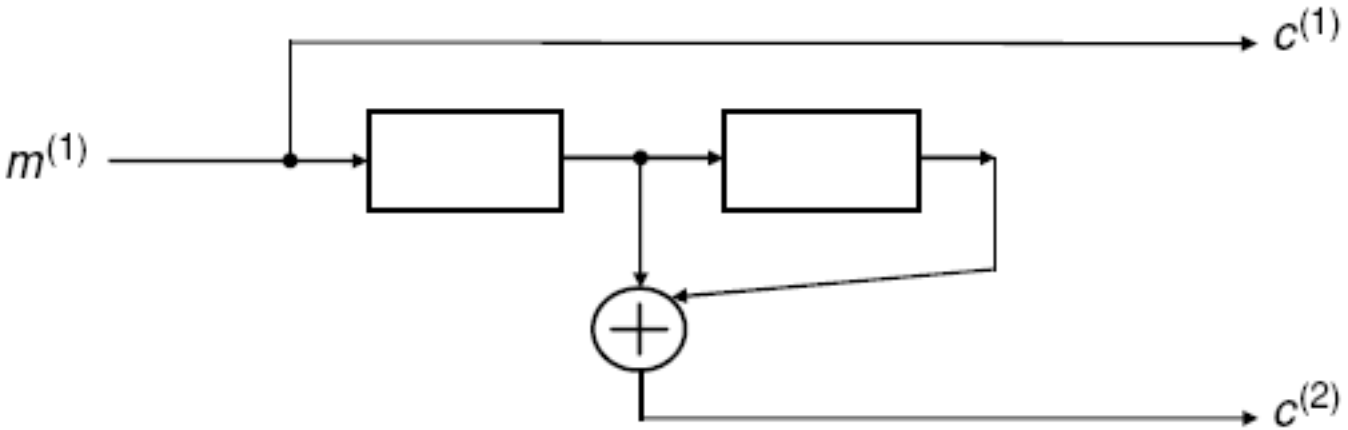
$$c^{(i)} = m^{(i)}, \quad i = 1, 2, \ldots, k$$

$$g_i^{(j)} = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

$$\boldsymbol{G}(D) = \begin{bmatrix} 1 & 0 & \cdots & 0 & G_1^{(k+1)}(D) & G_1^{(k+2)}(D) & \cdots & G_1^{(n)}(D) \\ 0 & 1 & \cdots & 0 & G_2^{(k+1)}(D) & G_2^{(k+2)}(D) & \cdots & G_2^{(n)}(D) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & G_k^{(k+1)}(D) & G_k^{(k+2)}(D) & \cdots & G_k^{(n)}(D) \end{bmatrix}$$

Convolutional Codes in Systematic Form

- Example: Determine the transfer function of the systematic convolutional code as shown bellow, and then obtain the code sequence for the input sequence m = (1101):



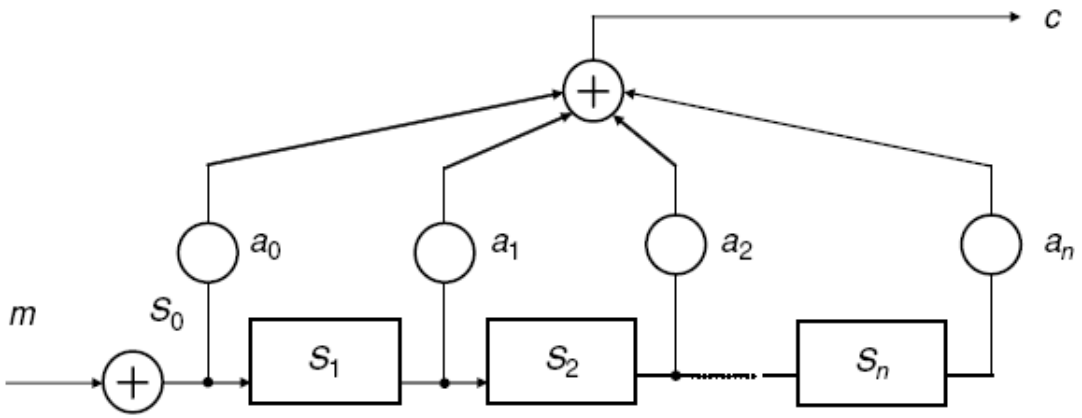
Convolutional Codes in Systematic Form

- The transfer function is:
  - $G(D) = [1 \quad D + D^2]$
- The message sequence in polynomial form is:
  - $M(D) = 1 + D + D^3$
- The code sequence in polynomial form is:
  - $C^{(1)}(D) = M(D)G^{(1)}(D) = 1 + D + D^3$
  - $C^{(2)}(D) = M(D)G^{(2)}(D) = (1 + D + D^3)(D + D^2) = D + D^3 + D^4 + D^5$
- Then
  - c = (10, 11, 00, 11, 01, 01)

In the case of systematic convolutional codes, there is no need to have an inverse transfer function decoder to obtain the input sequence, because this is directly read from the code sequence.

Finite Impulse Response FSSMs

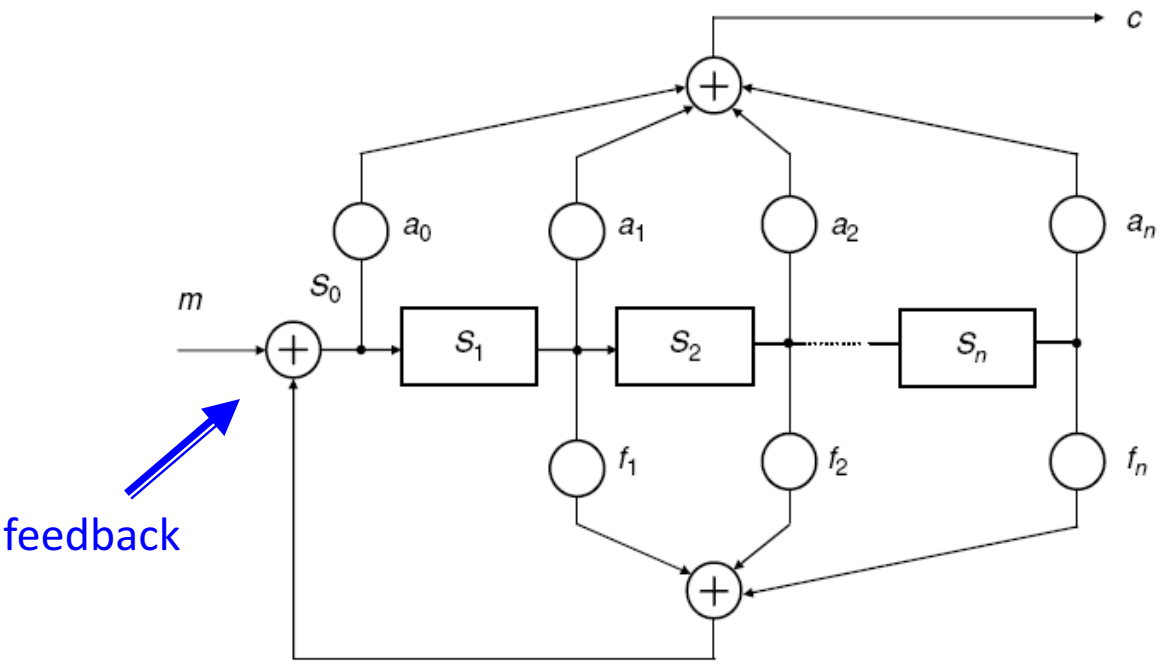
- The FIR general structure and the Transfer Function are:
  - $a_i = \{0 , 1\}$



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

Infinite Impulse Response FSSM

- The IIR general structure and the Transfer Function are:
  - $a_i = \{0 , 1\}$
  - $f_i = \{0 , 1\}$



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

State Transfer Function for FIR FSSMs

- The state transfer function, for the FSSM bellow, with a slightly different notation, showing the variables involved in the discrete time domain is:

$$m(k) = s_0(k)$$

$$c^{(1)}(k) = s_0(k) + s_2(k) = s_0(k) + s_0(k-2)$$

$$s_1(k) = s_0(k-1)$$

$$s_2(k) = s_0(k-2)$$

$$S_1(D) = DS_0(D) = DM(D)$$

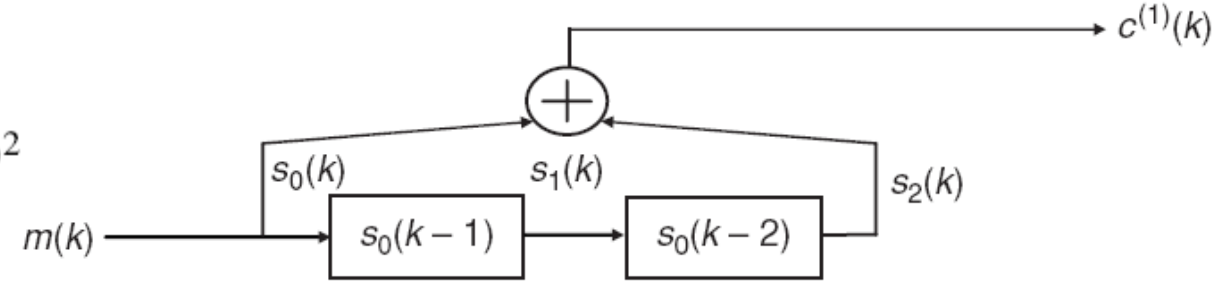
$$S_2(D) = D^2 S_0(D) = D^2 M(D)$$

- In the D domain:

$$C^{(1)}(D) = S_0(D) + D^2 S_0(D) = (1 + D^2)S_0(D) = (1 + D^2)M(D)$$

$$S_0(D) = M(D)$$

$$G(D) = \frac{C^{(1)}(D)}{M(D)} = 1 + D^2$$



State Transfer Function for FIR FSSMs

- The state transfer function can be used to determine the evolution of the states of the corresponding FSSM with respect to a particular input sequence
- In the case of the unit impulse input sequence, $M(D) = 1$, this state transfer function describes the state sequence as sequences in the D domain
- When the FSSM is a FIR FSSM, the impulse response additionally describes the shortest state sequence

State Transfer Function for FIR FSSMs

- For the previous example, the state of the FSSM is defined by the pair (S_1(D) S_2(D))

$$G(D) = \frac{C^{(1)}(D)}{M(D)} = 1 + D^2$$ -> transfer function

$$\boldsymbol{S}(D) = \left[S_0(D)/M(D) \;\; S_1(D)/M(D) \;\; S_2(D)/M(D) \right] = \left[1 \;\; D \;\; D^2 \right]$$ -> state transfer function

$$\left[S_0(D) \;\; S_1(D) \;\; S_2(D) \right] = \left[1 \;\; D \;\; D^2 \right] \bullet M(D)$$

$$\left[S_0(D) \;\; S_1(D) \;\; S_2(D) \right] = \left[1 \;\; D \;\; D^2 \right] \bullet 1 = \left[1 \;\; D \;\; D^2 \right]$$

-> state impulse response

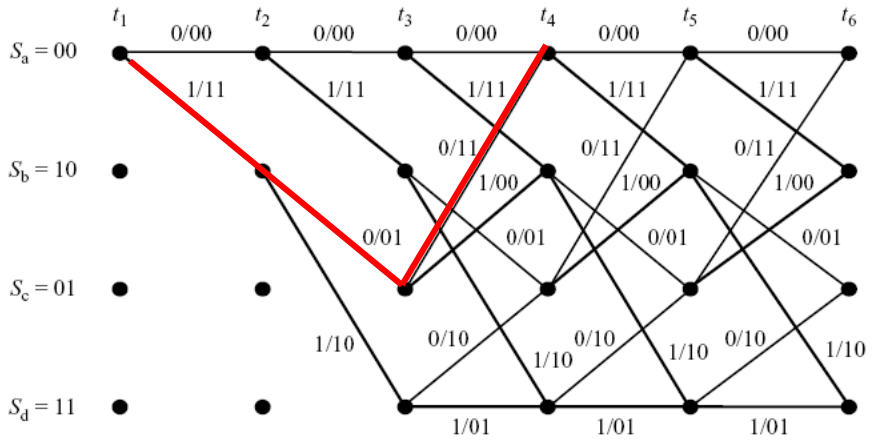
$$S_0 = (1, 0, 0, 0, 0, 0, \ldots)$$

$$S_1 = (0, 1, 0, 0, 0, 0, \ldots)$$

$$S_2 = (0, 0, 1, 0, 0, 0, \ldots)$$

State Transfer Function for FIR FSSMs

- The state transitions vector for the impulse response is:
  - $(S_1\ S_2) = (00, 10, 01, 00, 00, \ldots)$
    - describes the shortest state transition of the FSSM
    - the shortest sequence can be seen in the corresponding trellis diagram ($S_aS_bS_cS_a$)



State Transfer Function for IIR FSSMs

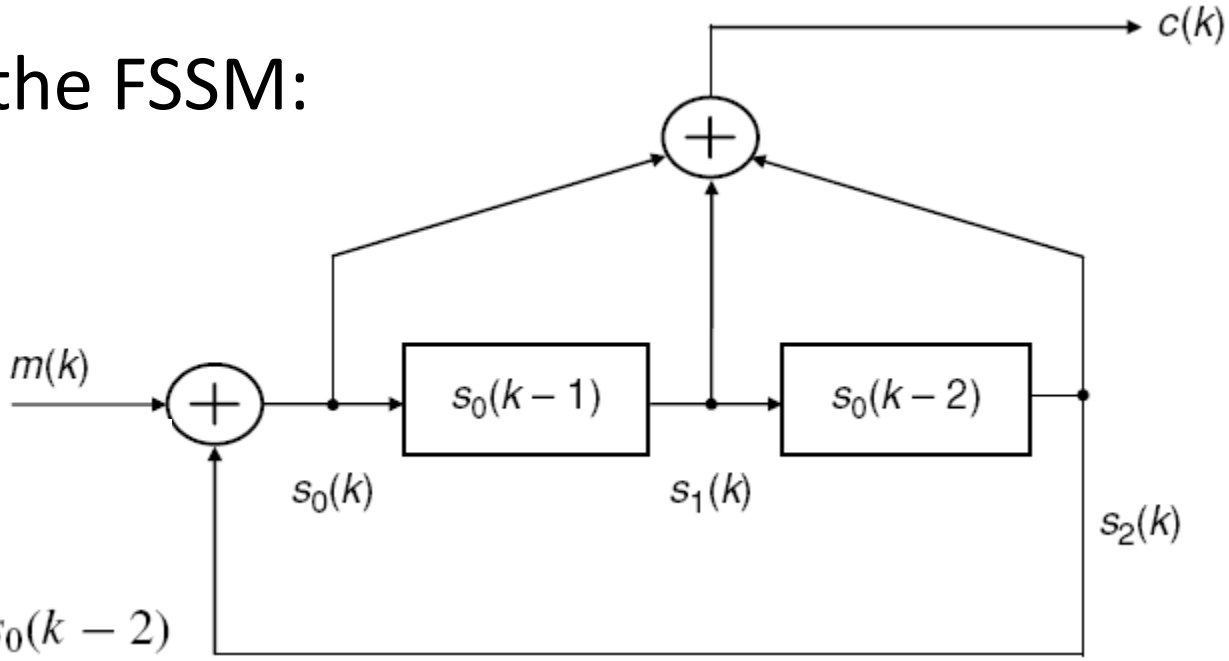
- Let us consider the FSSM:

$$s_0(k) = m(k) + s_2(k)$$

$$s_1(k) = s_0(k-1)$$

$$s_2(k) = s_0(k-2)$$

$$c(k) = s_0(k) + s_0(k-1) + s_0(k-2)$$



- A similar analysis to that presented for FIR FSSMs can be made

State Transfer Function for IIR FSSMs

- In the D domain:

$$S_1(D) = D\,S_0(D)$$

$$S_2(D) = D^2 S_0(D)$$

$$S_0(D) = M(D) + S_2(D) = M(D) + D^2 S_0(D)$$

$$S_0(D) + D^2 S_0(D) = M(D)$$

$$S_0(D) = \frac{M(D)}{1 + D^2}$$

$$C(D) = S_0(D) + D\,S_0(D) + D^2 S_0(D) = (1 + D + D^2)S_0(D) = (1 + D + D^2)\frac{M(D)}{1 + D^2}$$

$$G(D) = \frac{C(D)}{M(D)} = \frac{1 + D + D^2}{1 + D^2}$$ -> transfer function

$$S(D) = \left[\frac{S_0(D)}{M(D)} \; \frac{S_1(D)}{M(D)} \; \frac{S_2(D)}{M(D)} \right] = \left[\frac{1}{1 + D^2} \; \frac{D}{1 + D^2} \; \frac{D^2}{1 + D^2} \right]$$ -> state transfer function

State Transfer Function for IIR FSSMs

- The impulse response is now infinite, and it does not correspond to the shortest sequence of the FSSM
- The state transfer function can be used to identify which is the input for generating the shortest sequence
  - In this particular case, if the input is $M(D) = 1 + D^2$, the corresponding state sequence is:

$$[S_0(D) \quad S_1(D) \quad S_2(D)] = \left[\frac{1}{1+D^2} \quad \frac{D}{1+D^2} \quad \frac{D^2}{1+D^2}\right] \bullet M(D)$$

$$= \left[\frac{1}{1+D^2} \quad \frac{D}{1+D^2} \quad \frac{D^2}{1+D^2}\right] \bullet (1+D^2)$$

$$= [1 \quad D \quad D^2]$$

    - which is the same as the shortest sequence of the FIR FSSM, shown in previous example

Relationship Between the Systematic and the Non-Systematic Forms

- The transfer function description of an FSSM encoder can be used to obtain the equivalent systematic form of a given non-systematic encoder
- This conversion method consists of converting the transfer function of a non-systematic form into an expression of systematic form by means of matrix operations
- Example: Determine the equivalent systematic version of the convolutional encoder generated by the transfer function: $G(D) = G_{ns}(D) = [1+D^2 \;\; 1+D+D^2]$

Relationship Between the Systematic and the Non-Systematic Forms

- The transfer function should adopt the form bellow to correspond to a systematic convolutional encoder

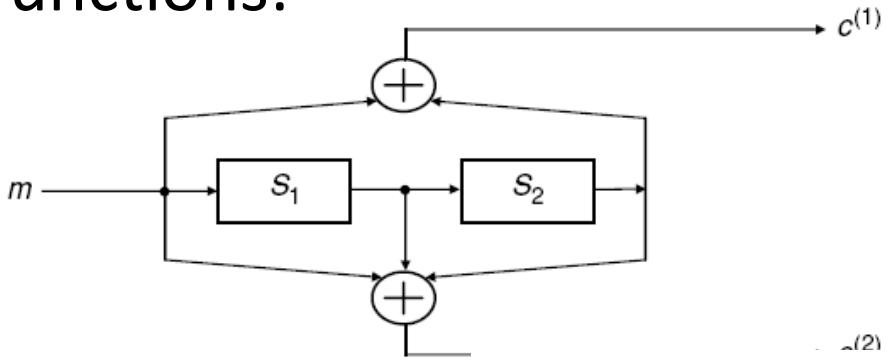
$$G(D) = \begin{bmatrix} 1 & 0 & \cdots & 0 & G_1^{(k+1)}(D) & G_1^{(k+2)}(D) & \cdots & G_1^{(n)}(D) \\ 0 & 1 & \cdots & 0 & G_2^{(k+1)}(D) & G_2^{(k+2)}(D) & \cdots & G_2^{(n)}(D) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & G_k^{(k+1)}(D) & G_k^{(k+2)}(D) & \cdots & G_k^{(n)}(D) \end{bmatrix}$$

- In this case the procedure is quite simple, because it only consists of dividing both polynomials of the transfer function by the polynomial $1 + D^2$

$$G_S(D) = \begin{bmatrix} 1 & \dfrac{1+D+D^2}{1+D^2} \end{bmatrix}$$

Relationship Between the Systematic and the Non-Systematic Forms

- A non-systematic convolutional code encoded with FIR transfer functions has an equivalent systematic convolutional code encoded with IIR transfer functions:

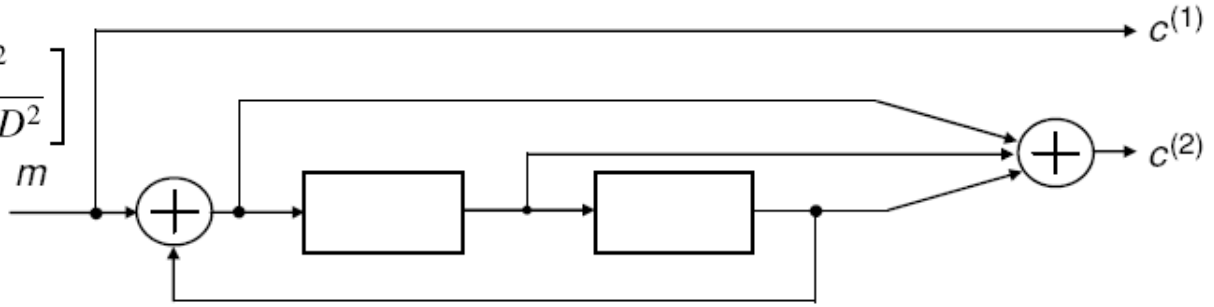


$S(D) = [1 \quad 1+D^2 \quad 1+D+D^2]$

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

$$S(D) = \left[\frac{1}{1+D^2} \quad \frac{D}{1+D^2} \quad \frac{D^2}{1+D^2} \right]$$

$$G(D) = \left[1 \quad \frac{1+D+D^2}{1+D^2} \right]$$

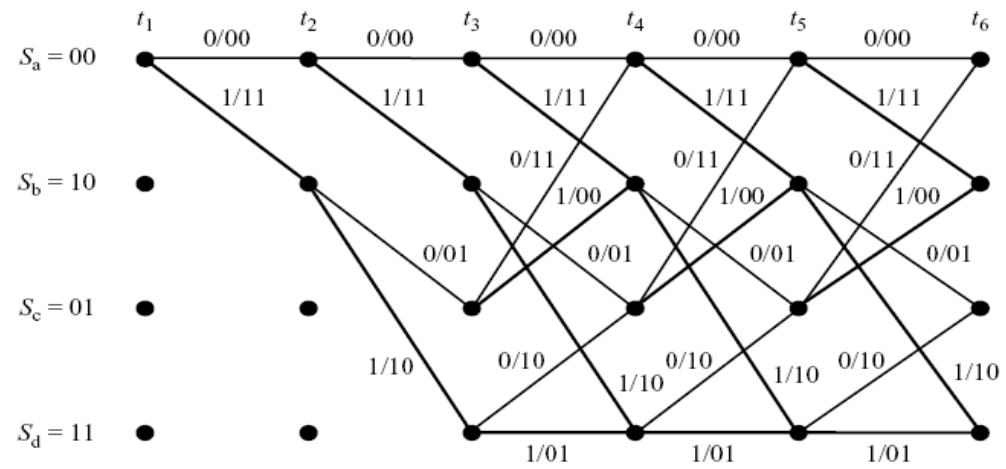


Relationship Between the Systematic and the Non-Systematic Forms

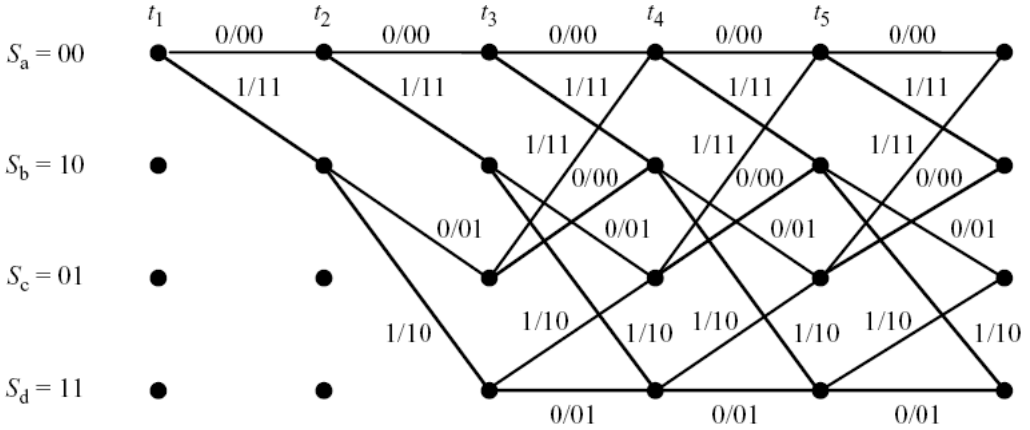
- It can be verified that transitions in the trellis have the same output assignments as the trellis of convolutional encoder in systematic form
- The difference between the systematic and the non-systematic forms of the same convolutional code is in the way the input is assigned a given output
- As in the case of block codes, the systematic convolutional encoder generates the same code as its corresponding non-systematic encoder, but with different input–output assignments

Relationship Between the Systematic and the Non-Systematic Forms

- Trellis



Systematic



Relationship Between the Systematic and the Non-Systematic Forms

- The shortest state sequence using IIR transfer functions, does not correspond to the unit impulse input
  - in this case generates an infinite output or state sequence
- The input that produces the shortest state sequence can be obtained by inspection of the corresponding state transfer function
  - $M(D) = 1 + D^2$
  - (S1 S2) = (00, 10, 01, 00, 00, . . .)
    - that is, the shortest sequence in the trellis

Relationship Between the Systematic and the Non-Systematic Forms

- The corresponding output sequence is:

$$C^{(1)}(D) = 1M(D) = 1 + D^2$$

$$C^{(2)}(D) = \frac{1 + D + D^2}{1 + D^2} M(D) = 1 + D + D^2$$

- which is an output of weight 5

Distance Properties of Convolutional Codes

- One of the most significant parameters of an error-correcting or error-detecting code is the minimum distance of the code, normally evaluated as the minimum value of the distance that exists between any two code vectors of the code
- When the code is linear, it is sufficient to determine the distance between any code vector and the all-zero vector
- As seen for block codes, the minimum distance can be interpreted as the minimum-weight error pattern that converts a given code vector into another code vector in the same code

Distance Properties of Convolutional Codes

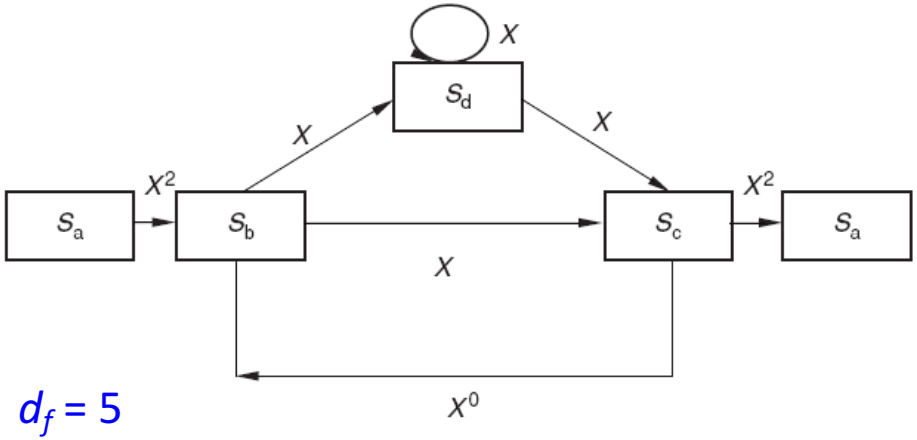
- In the case of convolutional codes, this becomes the number of errors that convert a given code sequence into another valid code sequence
- Since almost all convolutional codes of practical use are linear, the minimum distance of the code can be determined by finding the code sequence of minimum weight
- The above analysis implies a search for the minimum number of errors that convert the all-zero sequence into another code sequence

Distance Properties of Convolutional Codes

- This can be seen in the corresponding trellis of the convolutional code as a sequence that emerges from the all-zero state, normally labelled S_a, and arrives back at the same state after a certain number of transitions
- Then the Hamming or minimum distance of the code can be determined by calculating the minimum weight among all the sequences that emerge from and arrive back at the all-zero state S_a after a finite number of transitions
- A tool for analysing the distance properties of a convolutional code is obtained by modifying the traditional state diagram, in such a way that the modified diagram starts and ends in the all-zero state S_a

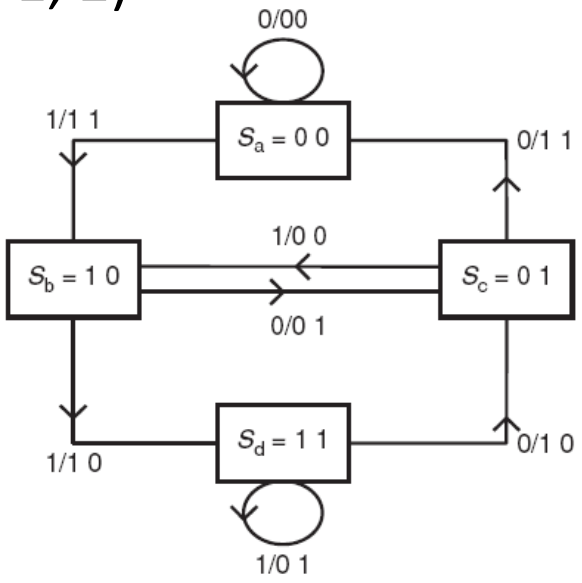
Distance Properties of Convolutional Codes

- Using the previous example of the $C_{conv}(2, 1, 2)$



$d_f = 5$

Modified state diagram



Original state diagram

  - In this modified state diagram branches emerging and arriving at the states are denoted by the term X^i, where i is the weight of the code sequence that corresponds to that branch

Minimum Free Distance of a Convolutional Code

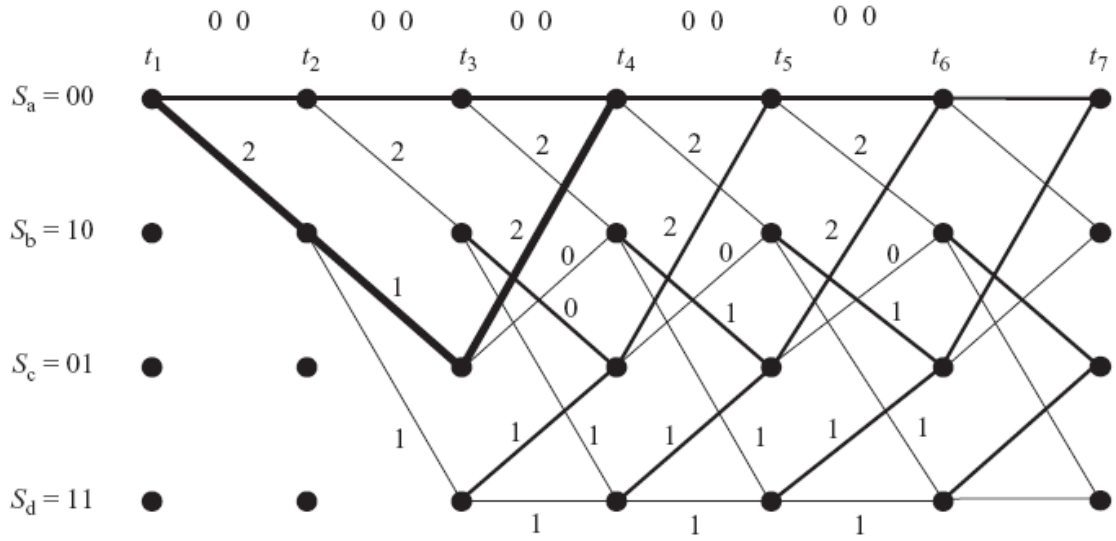
- The minimum free distance determines the properties of a convolutional code, and it is defined as:
  - $d_f = \min \{ d(c_i, c_j) : m_i \neq m_j \}$
  - where c_i , c_j are two code sequences that correspond to the message sequences $m_i \neq m_j$
- The minimum free distance is defined as the minimum distance between any two code sequences of the convolutional code
- Assuming that the convolutional code is linear, the calculation of the minimum distance between any two code sequences is the same as determining the weight of the sum of these two code sequences

Minimum Free Distance

- The calculation of the minimum distance between any two code sequences is the same as determining the weight of the sum of these two code sequences as:
  - $d_f = \min \{ w(c_i \oplus c_j) : m_i \neq m_j \} = \min \{ w(c) : m_i \neq 0 \}$
- This also implies that the minimum free distance of a convolutional code is the minimum weight calculated among all the code sequences that emerge from and return to the all-zero state, and that are not the all-zero sequences, c ≠ 0
- Example: Determine the minimum free distance of the convolutional code $C_{conv}(2, 1, 2)$ used in previous examples, by employing the above procedure, implemented over the corresponding trellis

Minimum Free Distance

- The sequence corresponding to the path described by the sequence of states $S_aS_bS_cS_a$, seen in bold in Figure, is the sequence of minimum weight, which is equal to 5



- There are other sequences like those described by the state sequences $S_aS_bS_cS_bS_cS_a$ and $S_aS_bS_dS_cS_a$ that both are of weight 6. The remaining paths are all of larger weight, and so the minimum free distance of this convolutional code is $d_f = 5$

Minimum Free Distance

- In the case of convolutional codes the distance between any two code sequences is not clearly determined, since the transmission is done not in blocks of information but as a continuous sequence of bits with a degree of memory
- However, when the all-zero sequence is transmitted, and this sequence is converted by the effect of the channel errors into another code sequence, it is clear that an undetectable error pattern has occurred
- This is the same as in the case of block codes, where the minimum Hamming distance can be considered as the minimum number of errors produced by the channel that have to occur in a transmitted code vector to convert it in another code vector

Minimum Free Distance

- The error-correction capability of the code is defined as the number *t* of errors that can be corrected, which is equal to:

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

- This error-correction capability is obtained when error events are separated by at least the constraint length of the code, measured in bits

Maximum Likelihood Detection

- For a given code sequence c generated by the encoder of a convolutional code, the channel noise converts this sequence into the received sequence s_r, which is essentially the code sequence *c* with errors produced in the transmission
- An optimal decoder is one that is able to compare the conditional probabilities $P(s_r/c')$ that the received sequence s_r corresponds to a possible code sequence c', and decide upon the code sequence with the highest conditional probability

Maximum Likelihood Detection

- The maximum likelihood criterion is given by:

$$P(s_r/c') = \max_{\text{all } c} P(s_r/c)$$

- It is in agreement with the intuitive idea of decoding by selecting the code sequence that is most alike the received sequence
- For a code sequence of length *L* bits, there are 2^{RcL} possible sequences, where *Rc* is the rate of the code
- The maximum likelihood decoder selects a sequence c', from the set of all possible sequences, which has the maximum similarity to the received sequence

Maximum Likelihood Detection

- If the channel is memoryless, and the noise is additive, white and Gaussian (AWGN), each symbol is independently affected by this noise
- For a convolutional code of rate 1/*n*, the probability of being alike to the received sequence is measured as:

$$P(s_r/c) = \prod_{i=1}^{\infty} P(s_{ri} / c_i) = \prod_{i=1}^{\infty} \prod_{j=1}^{n} P(s_{r,ji} / c_{ji})$$

- The decoding procedure consists of selecting a sequence that maximizes the probability function
- One algorithm that performs this procedure for convolutional codes is the Viterbi decoding algorithm

The Viterbi Algorithm

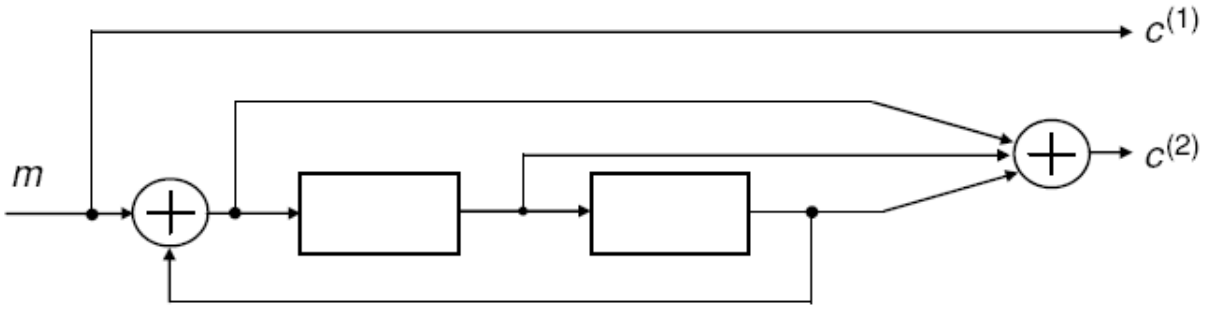
- The Viterbi algorithm (VA) performs maximum likelihood decoding. It is applied to the trellis of a convolutional code whose properties are conveniently used to implement this algorithm
- As explained above, one of the main problems that faces maximum likelihood decoding is the number of calculations that have to be done over all the possible code sequences
- The VA reduces this complexity of calculation by avoiding having to take into account all the possible sequences

The Viterbi Algorithm

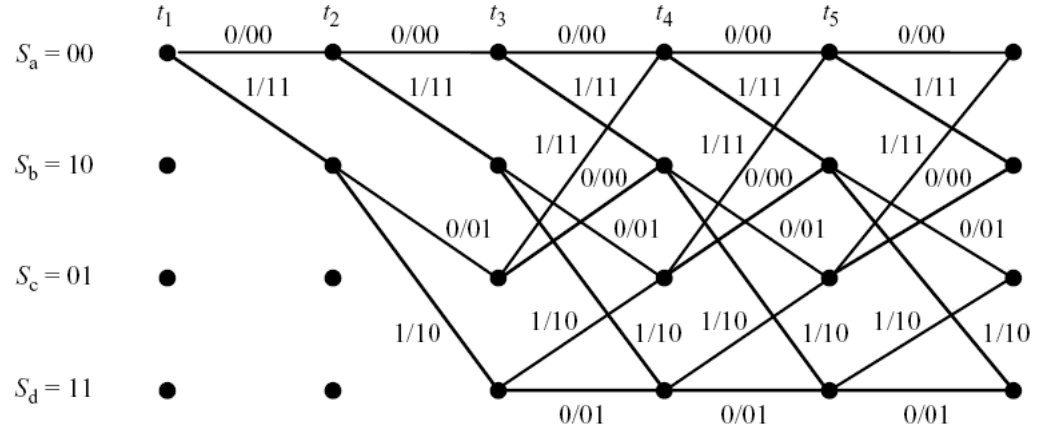
- The decoding procedure consists of calculating the cumulative distance between the received sequence at an instant t_i at a given state of the trellis, and each of all the code sequences that arrive at that state at that instant t_i
- This calculation is done for all states of the trellis, and for successive time instants, in order to look for the sequence with the minimum cumulative distance
- The sequence with the minimum cumulative distance is the same as the sequence with the highest probability of being alike to the received sequence if transmission is done over the AWGN channel

Example of VA

- Example: Apply the VA to the convolutional code, whose trellis and encoder are represented bellow, if the received sequence is s_r = 11 01 01 00 11 . . . .

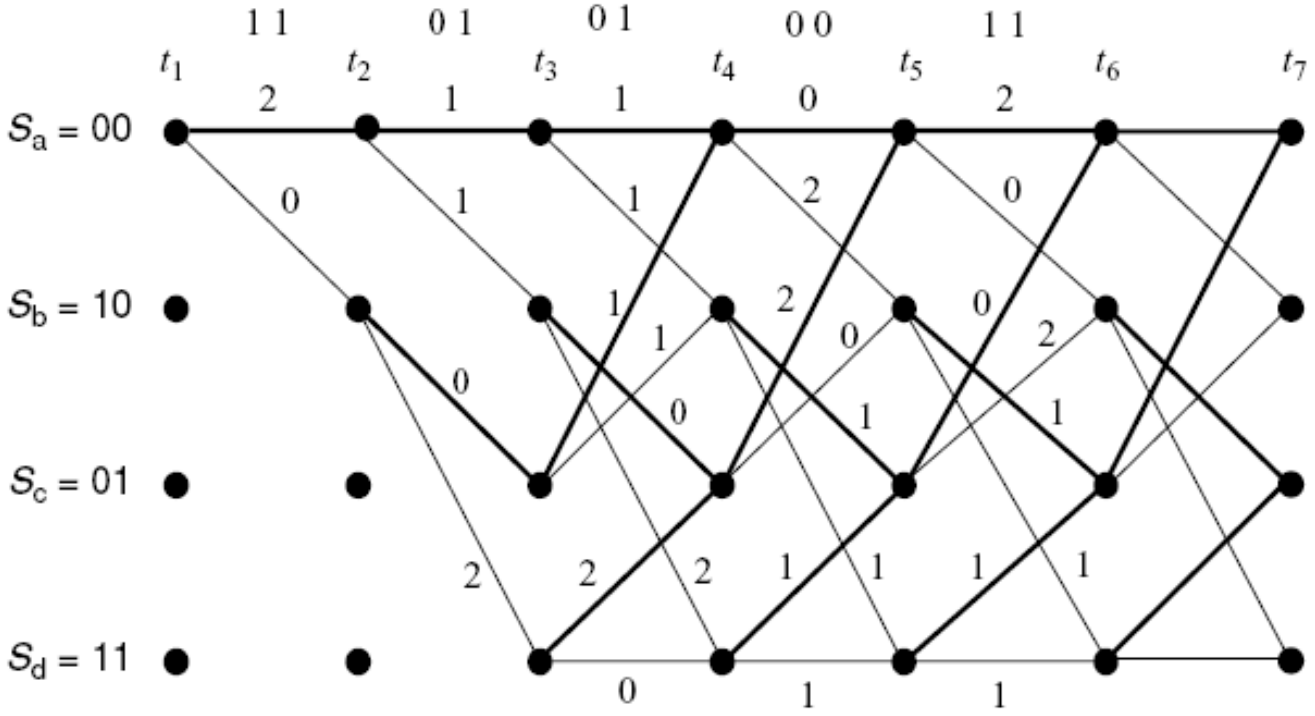


Message sequence	1 0 1 0 1
Code sequence	11 01 11 00 11
Received sequence	11 01 01 00 11



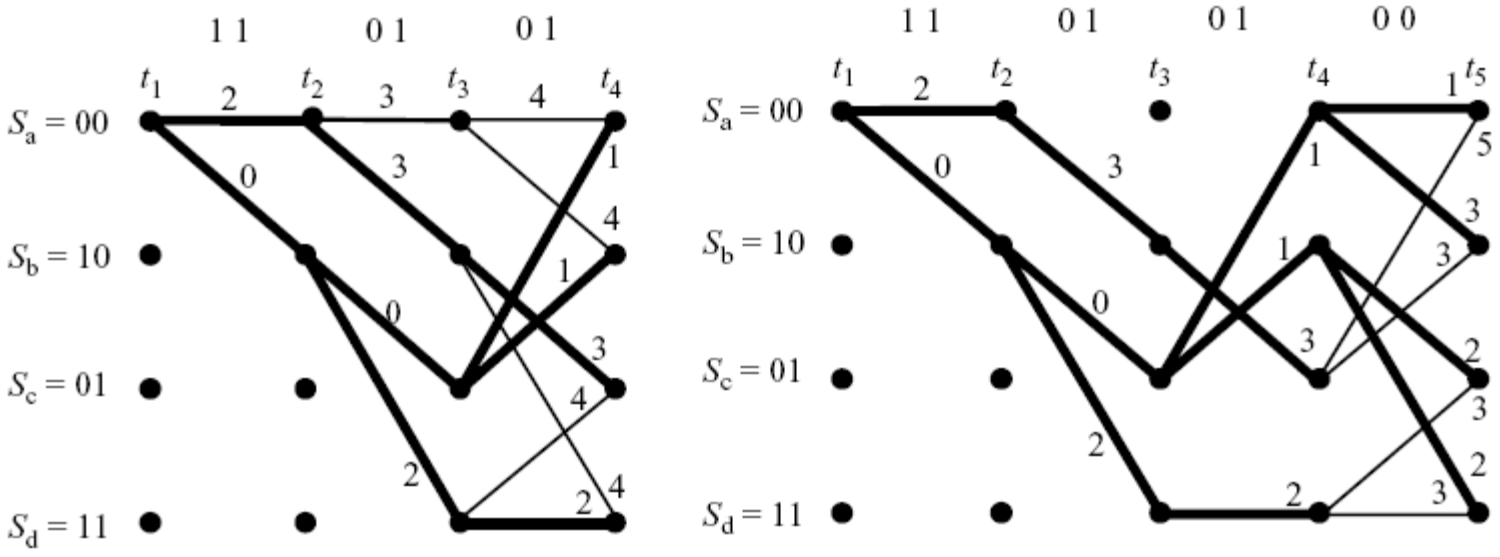
Example of VA

- The first step is to determine the Hamming distance between the received sequence and the outputs at the different states and time instants, over the trellis



Example of VA

- When two or more paths arrive at a given time instant and state of the trellis, only one of them would have the minimum cumulative distance, and should be selected from among the others as the survivor



The Viterbi Algorithm

- After a given number of time instants, the procedure is truncated, and the sequence with the minimum cumulative Hamming distance is selected as the decoded sequence
- J is defined as the decoding length, measured in time instants
- It can be heuristically shown that the error-correction capability of the convolutional code is maximized if J is approximately equal to five times the constraint length of the code; that is, $J \approx 5(K + 1)$
  - In the example: $J = 15$

The Viterbi Algorithm

- VA leads directly to the estimated message sequence, and performs error correction without the need of decoding table look-up, or of algebraic equation solution, as in the case of traditional syndrome decoding of block codes

- This fact makes convolutional codes particularly efficient in FEC systems, or in general, for those coding schemes where error correction is preferred over error detection

Reading Material

- *S. Lin and D. J. Costello. Error Control Coding. Prentice-Hall, Upper Saddle River, NJ, 2nd edition, 2004 (ISBN 0-13-017973-6).*
- *Sílvio A. Abrantes. Códigos Correctores de Erros em Comunicações Digitais, FEUP Edições, Porto, 2010 (ISBN 978-972-752-127-2).*
- *Simon Haykin. Communication Systems, McMaster Univ, 4th Edition, 2004. (ISBN: 978-0-471-17869-9).*