Data and Computer Communications

Error Correction

Syllabus

- Tentatively

Week 1	Overview
Week 2	Data Transmission
Week 3	Signal encoding techniques
Week 4	Error Detection
Week 5	**Error correction**
Week 6	Flow Control
Week 7	Error control
Week 8	HDLC
Week 9	Multiplexing
Week 10	Spread spectrum
Week 11	Wireless channel characteristics
Week 12	OFDM
Week 13	Packet switching
Week 14	Routing
Week 15	Revision

10.3 Correction

Retransmission

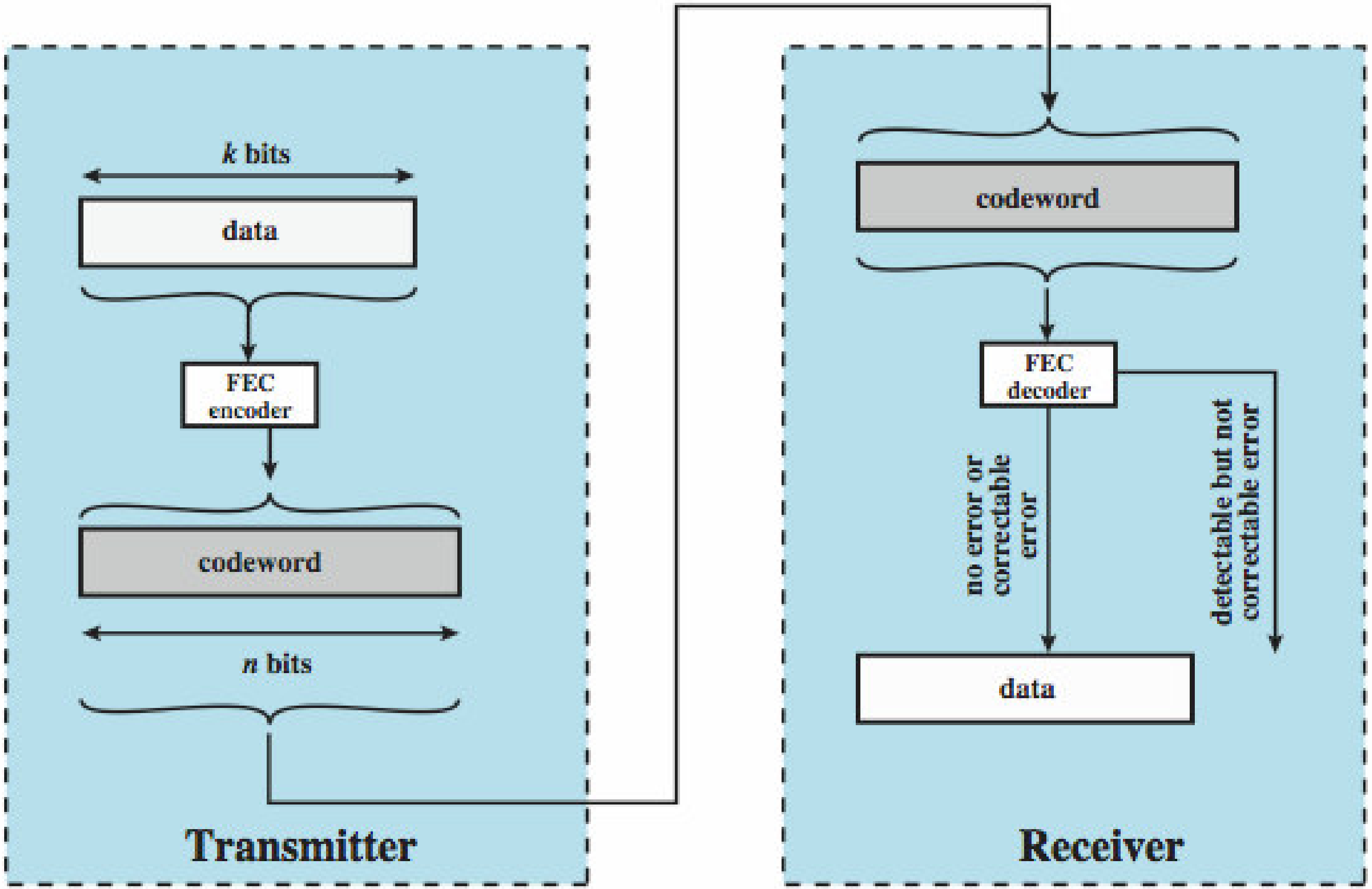
Forward Error Correction

Burst Error Correction

Error Correction

- correction of detected errors usually requires data block to be retransmitted
- not appropriate for wireless applications
  - bit error rate is high causing lots of retransmissions
  - when propagation delay long (satellite) compared with frame transmission time, resulting in retransmission of frame in error plus many subsequent frames
- instead need to correct errors on basis of bits received
- error correction provides this

Error Correction Process



How Error Correction Works

- adds redundancy to transmitted message
- can deduce original despite some errors
- eg. block error correction code
  - map *k* bit input onto an *n* bit codeword
  - each distinctly different
  - if get error assume codeword sent was closest to that received
- means have reduced effective data rate

Table 10.2 Data and redundancy bits

Number of data bits m	Number of redundancy bits r	Total bits m + r
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

10.14 *Positions of redundancy bits in Hamming code*

11	10	9	8	7	6	5	4	3	2	1
d	**d**	**d**	r_8	**d**	**d**	**d**	r_4	**d**	r_2	r_1

Redundancy bits calculation

r_1 will take care of these bits.

11		9		7		5		3		1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_2 will take care of these bits.

11	10			7	6			3	2	
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

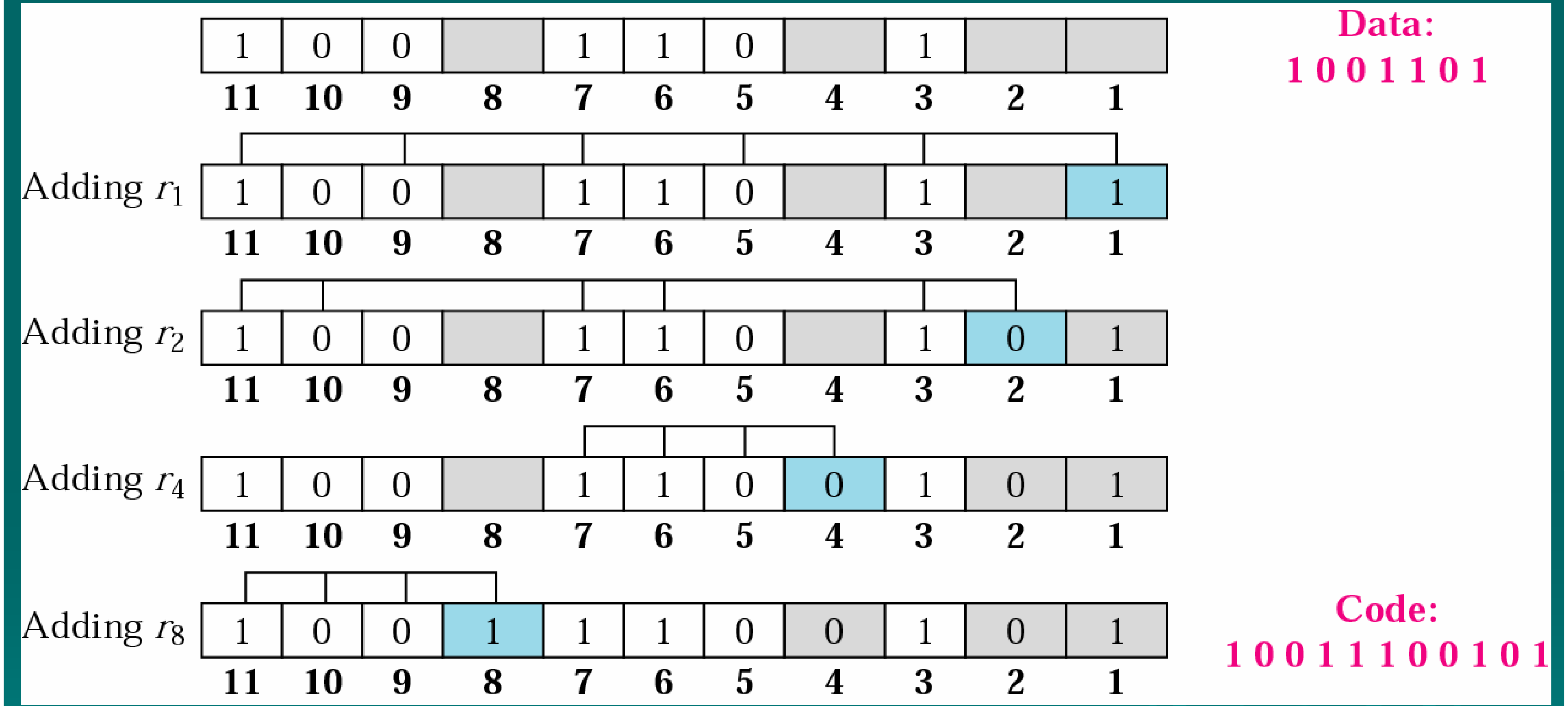
r_4 will take care of these bits.

				7	6	5	4			
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

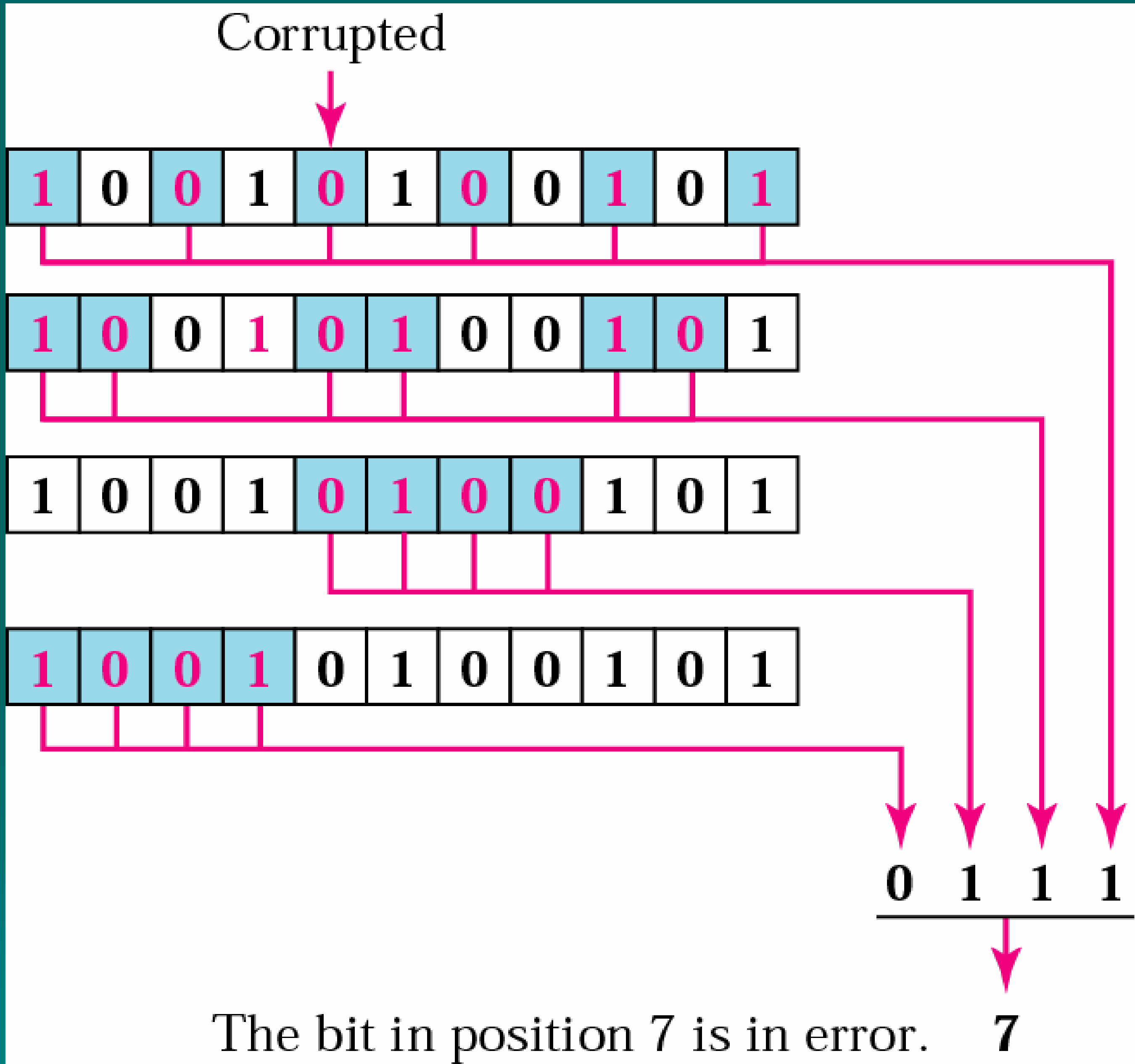
r_8 will take care of these bits.

11	10	9	8							
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

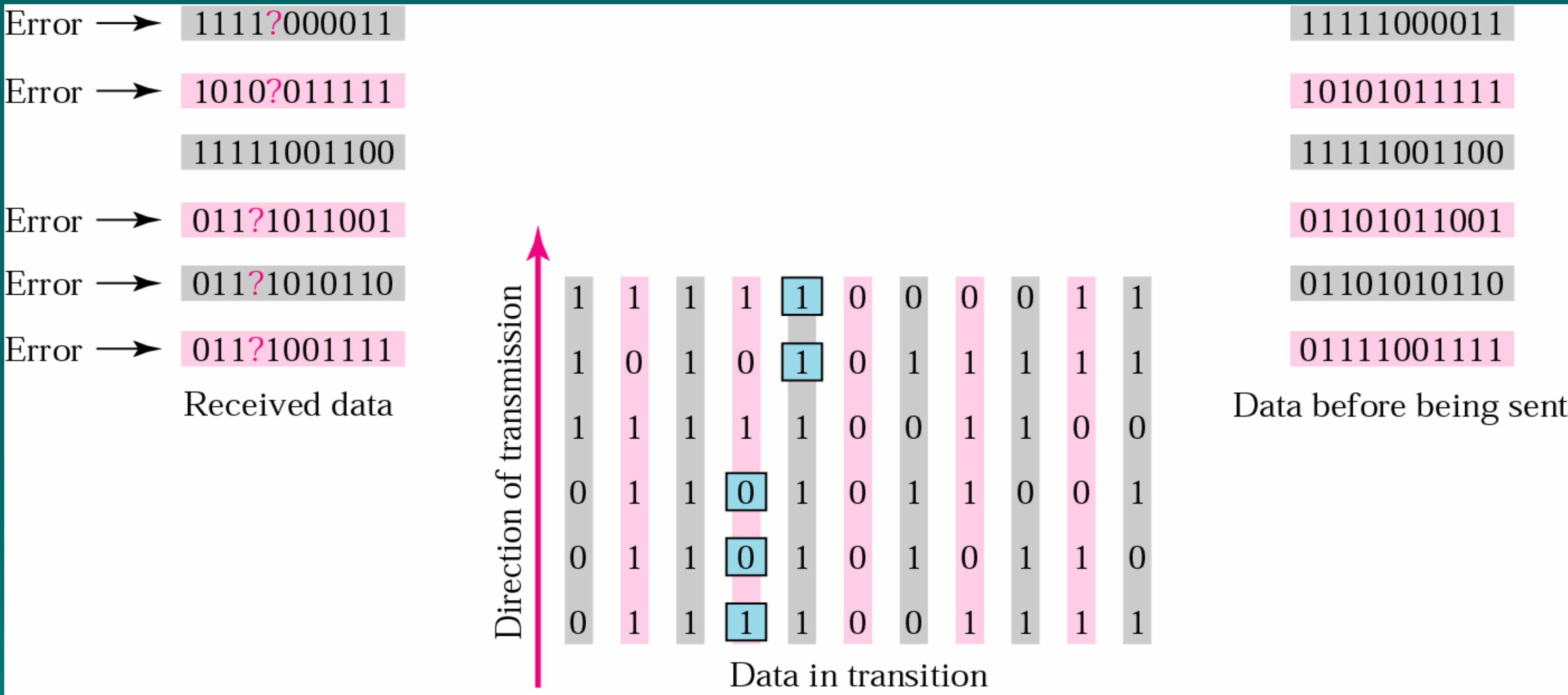
10.16 *Example of redundancy bit calculation*



10.17 *Error detection using Hamming code*



10.18 *Burst error correction example*



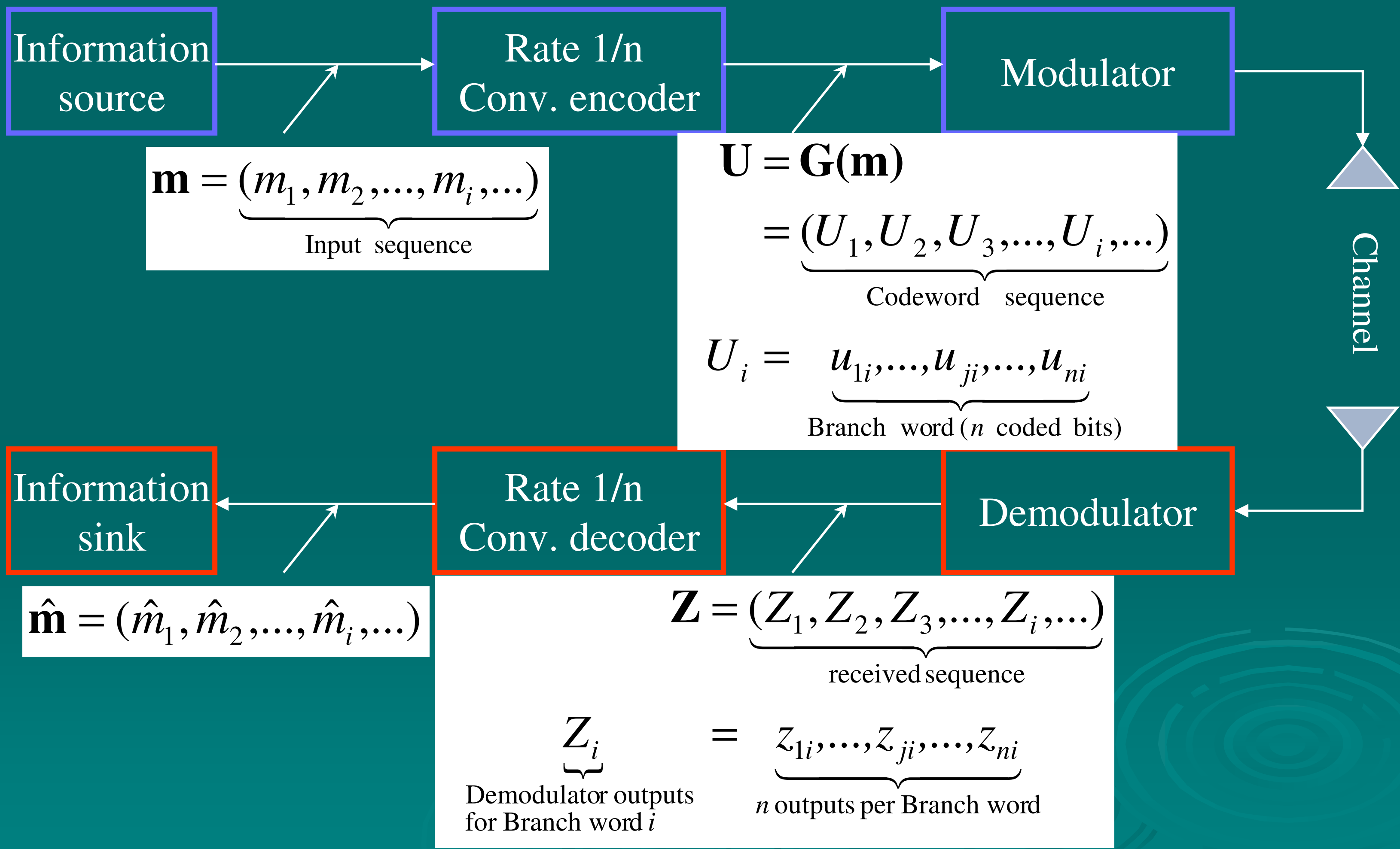
Convolutional codes

- Convolutional codes offer an approach to error control coding substantially different from that of block codes.
  - A convolutional encoder:
    - encodes the entire data stream, into a single codeword.
    - does not need to segment the data stream into blocks of fixed size (*Convolutional codes are often forced to block structure by periodic truncation*).
    - is a machine with memory.
- This fundamental difference in approach imparts a different nature to the design and evaluation of the code.
  - Block codes are based on algebraic/combinatorial techniques.
  - Convolutional codes are based on construction techniques.

Convolutional codes-cont'd

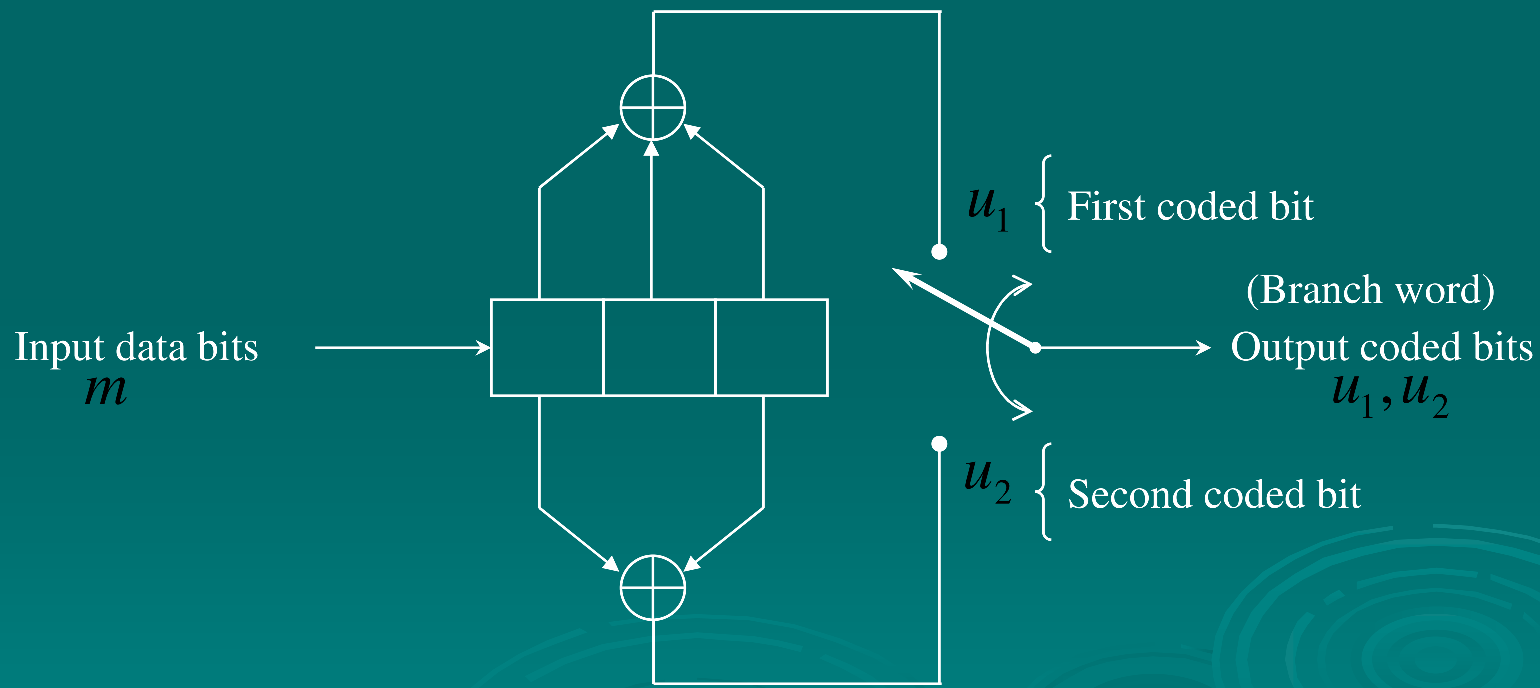
- A Convolutional code is specified by three parameters (n,k,K) or $(k/n,K)$ where
  - $R_c = k/n$ is the coding rate, determining the number of data bits per coded bit.
    - In practice, usually $k=1$ is chosen and we assume that from now on.
  - K is the constraint length of the encoder a where the encoder has $K-1$ memory elements.
    - There is different definitions in literatures for constraint length.

Block diagram of the DCS



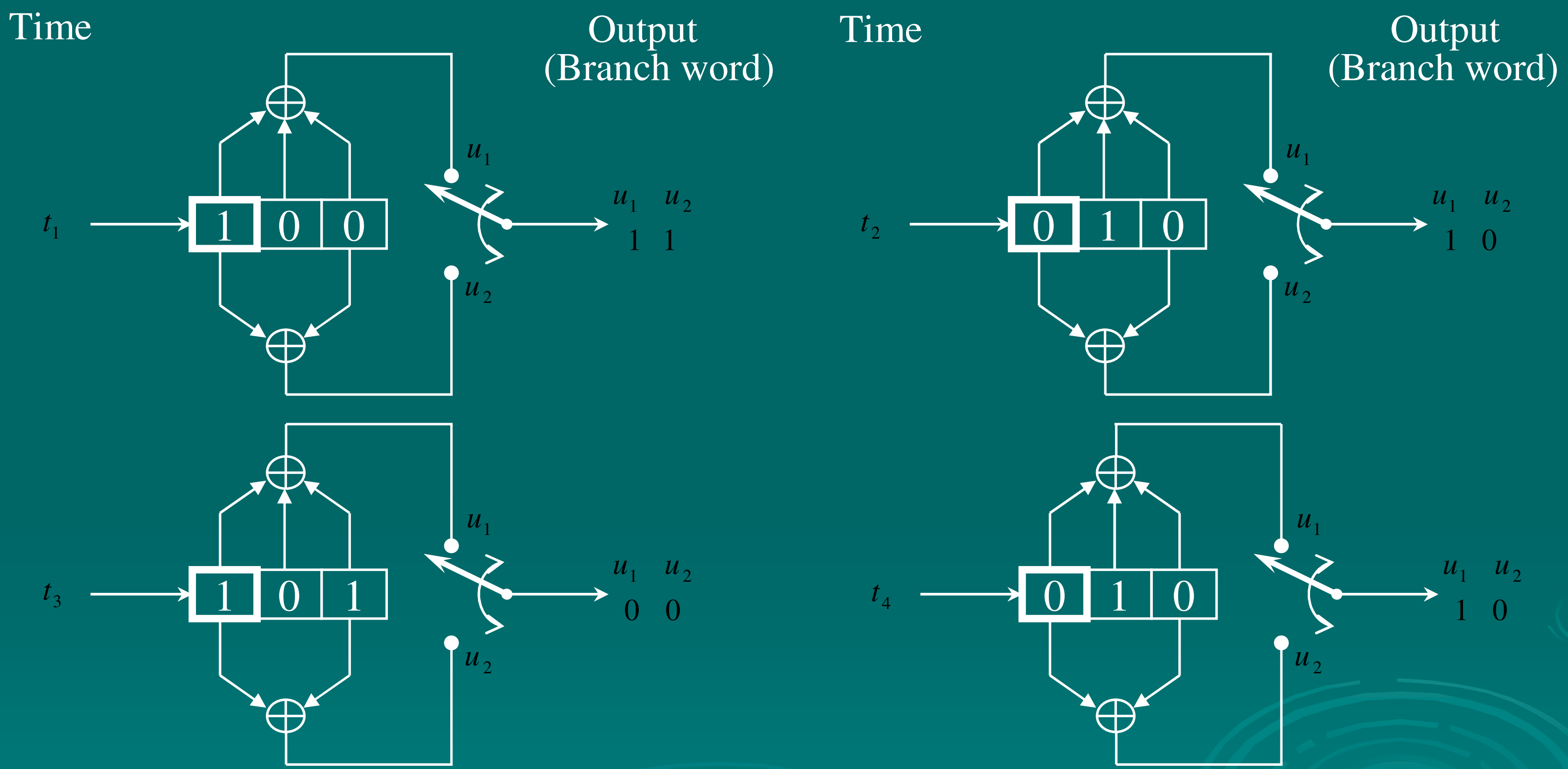
A Rate ½ Convolutional encoder

- Convolutional encoder (rate ½, K=3)
  - 3 shift-registers where the first one takes the incoming data bit and the rest, form the memory of the encoder.

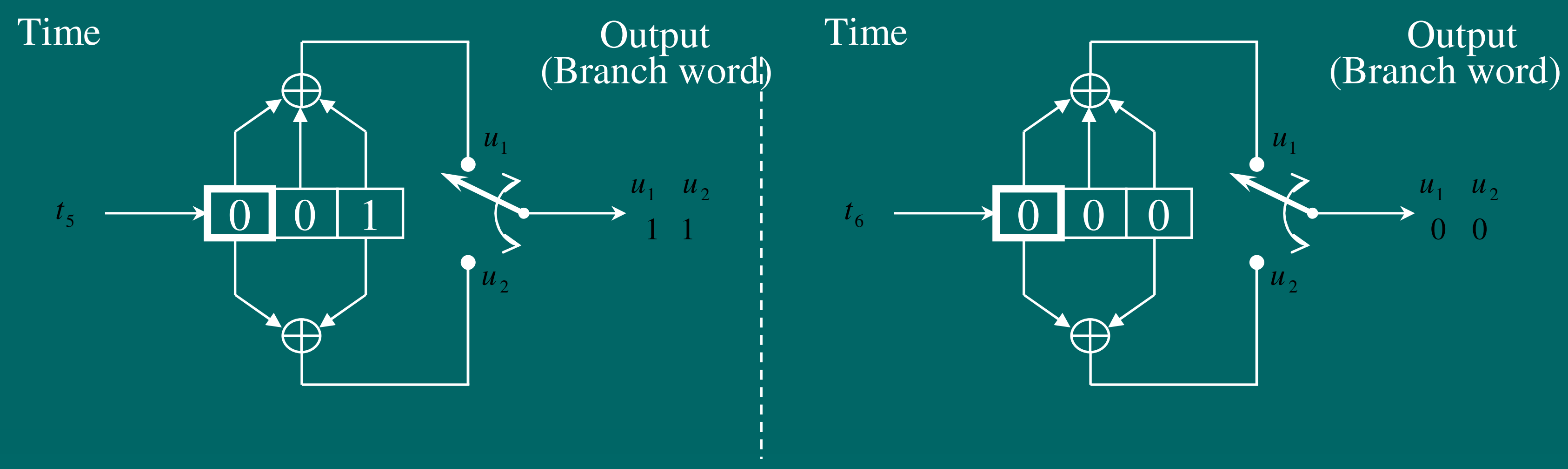


A Rate ½ Convolutional encoder

Message sequence: $\mathbf{m} = (101)$



A Rate ½ Convolutional encoder



$\mathbf{m} = (101)$ → Encoder → $\mathbf{U} = (11 \quad 10 \quad 00 \quad 10 \quad 11)$

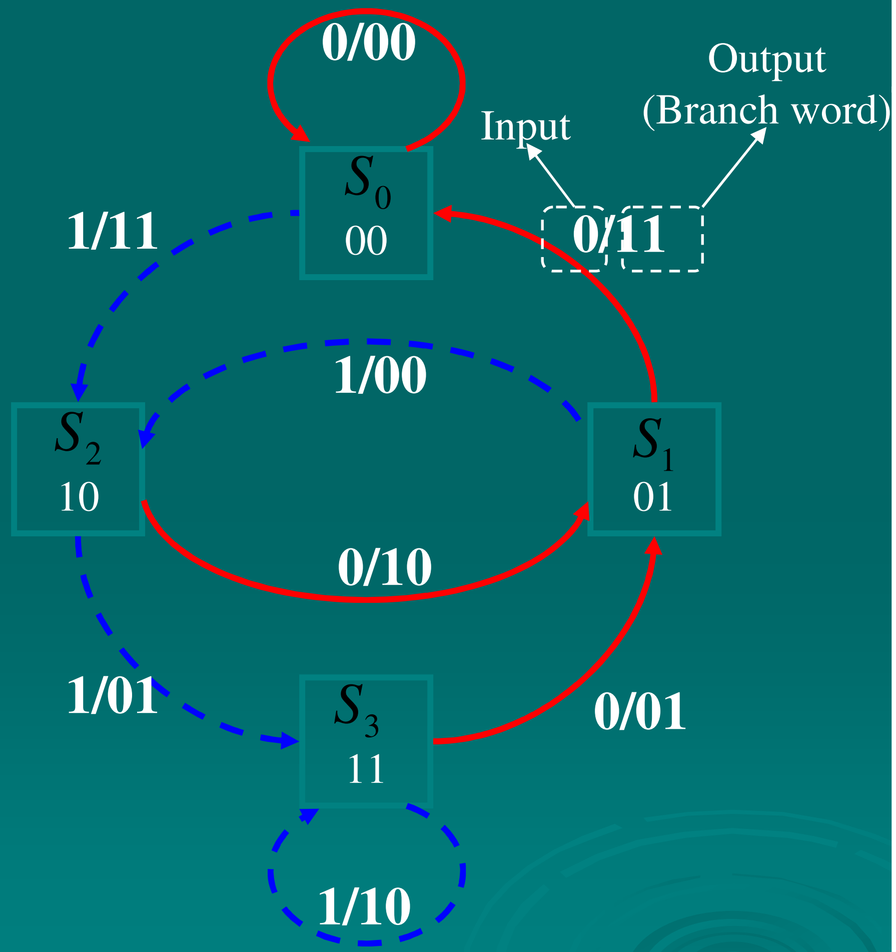
State diagram

- A finite-state machine only encounters a finite number of states.
- State of a machine: the smallest amount of information that, together with a current input to the machine, can predict the output of the machine.
- In a Convolutional encoder, the state is represented by the content of the memory.
- Hence, there are 2^{K-1} states.

State diagram – cont'd

- A state diagram is a way to represent the encoder.
- A state diagram contains all the states and all possible transitions between them.
- Only two transitions initiating from a state
- Only two transitions ending up in a state

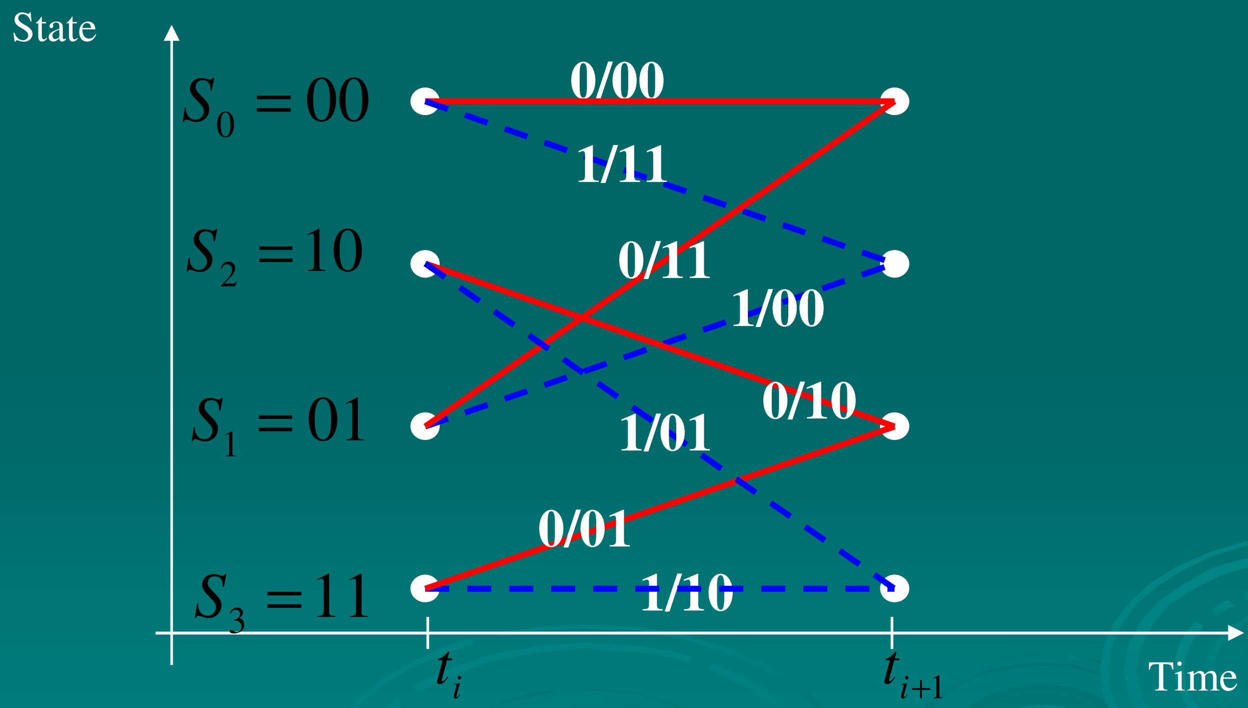
State diagram – cont'd



Current state	input	Next state	output
S_0 00	0	S_0	00
	1	S_2	11
S_1 01	0	S_0	11
	1	S_2	00
S_2 10	0	S_1	10
	1	S_3	01
S_3 11	0	S_1	01
	1	S_3	10

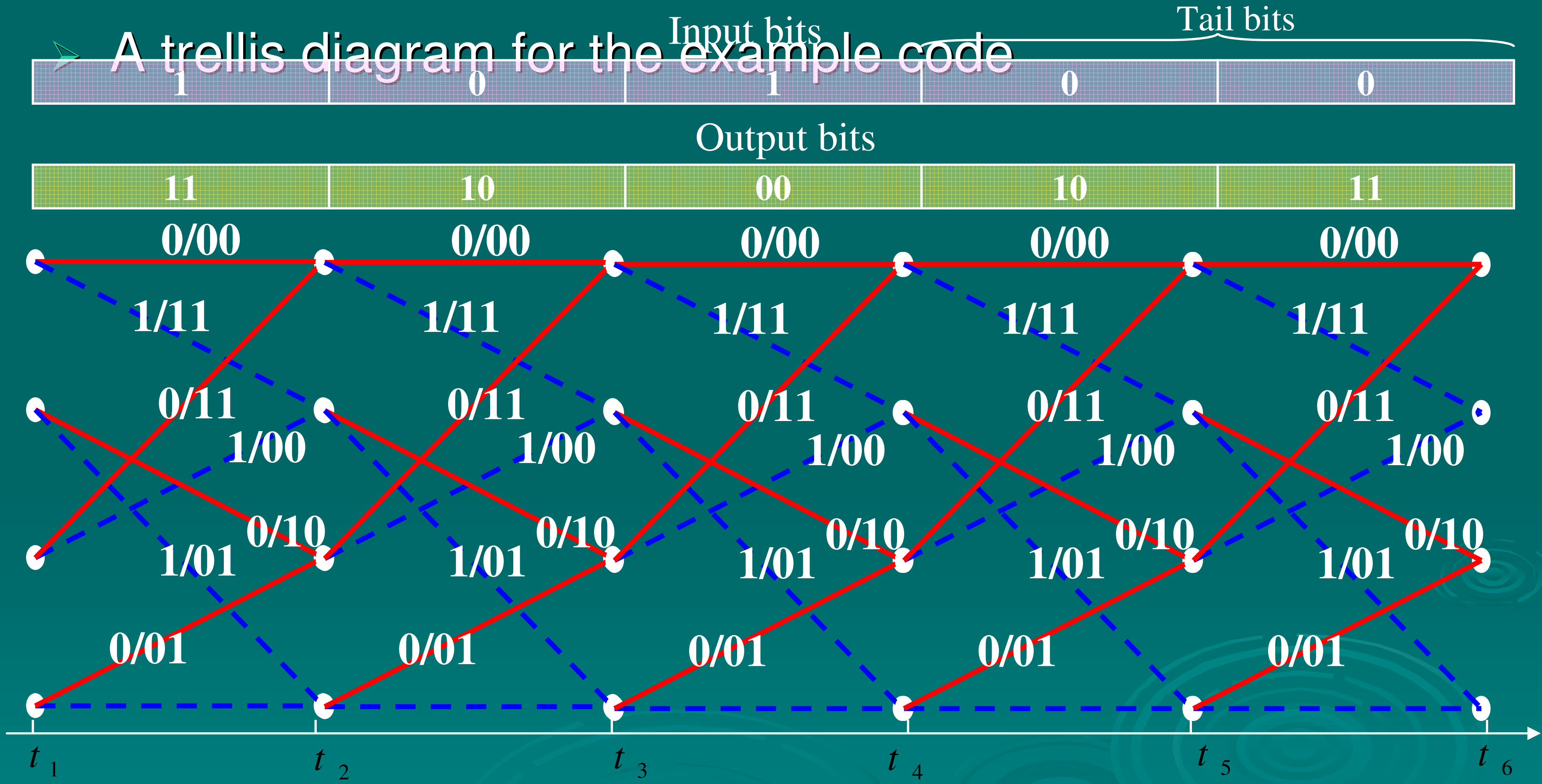
Trellis – cont'd

- Trellis diagram is an extension of the state diagram that shows the passage of time.
  - Example of a section of trellis for the rate ½ code

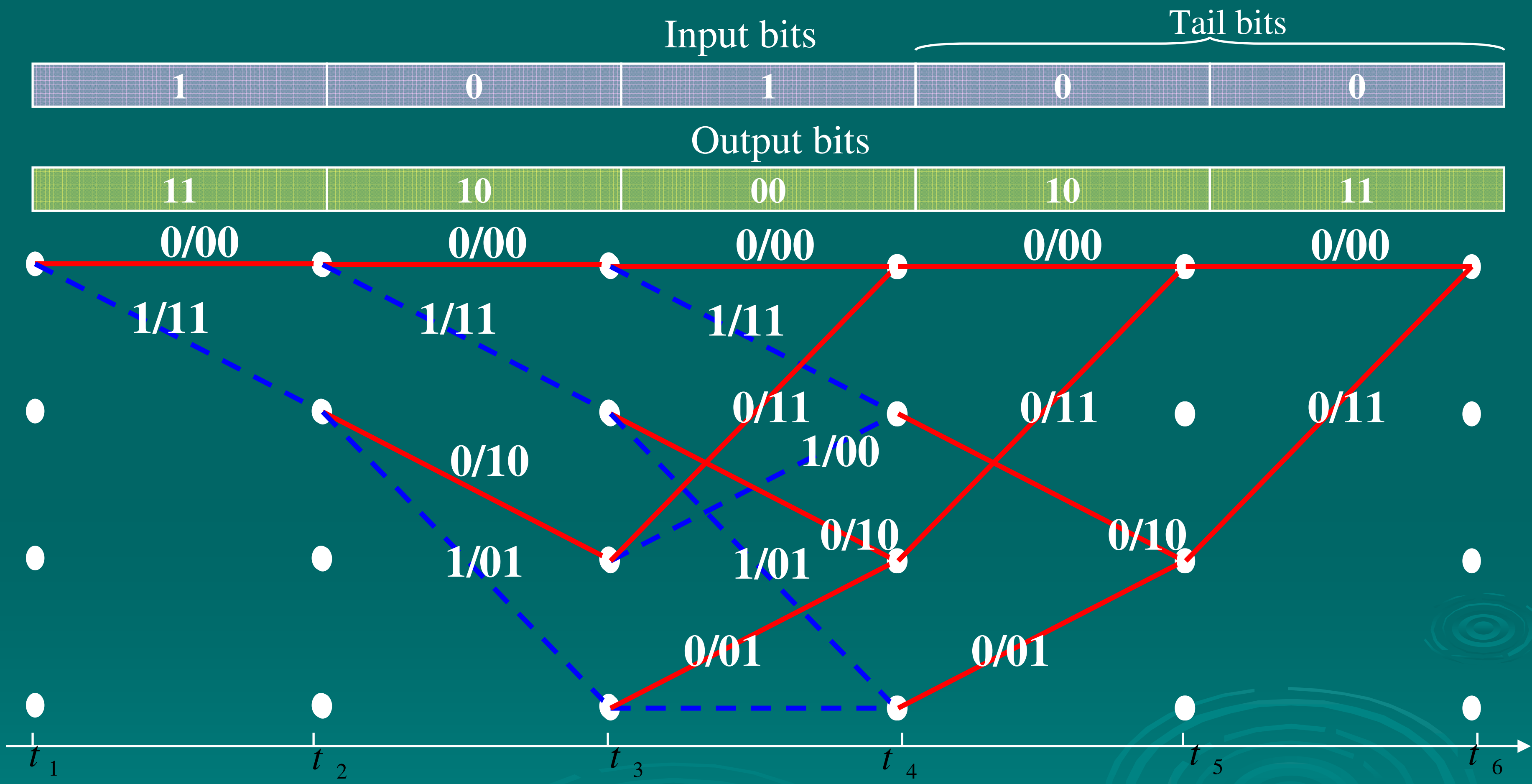


Trellis –cont’d

- A trellis diagram for the example code

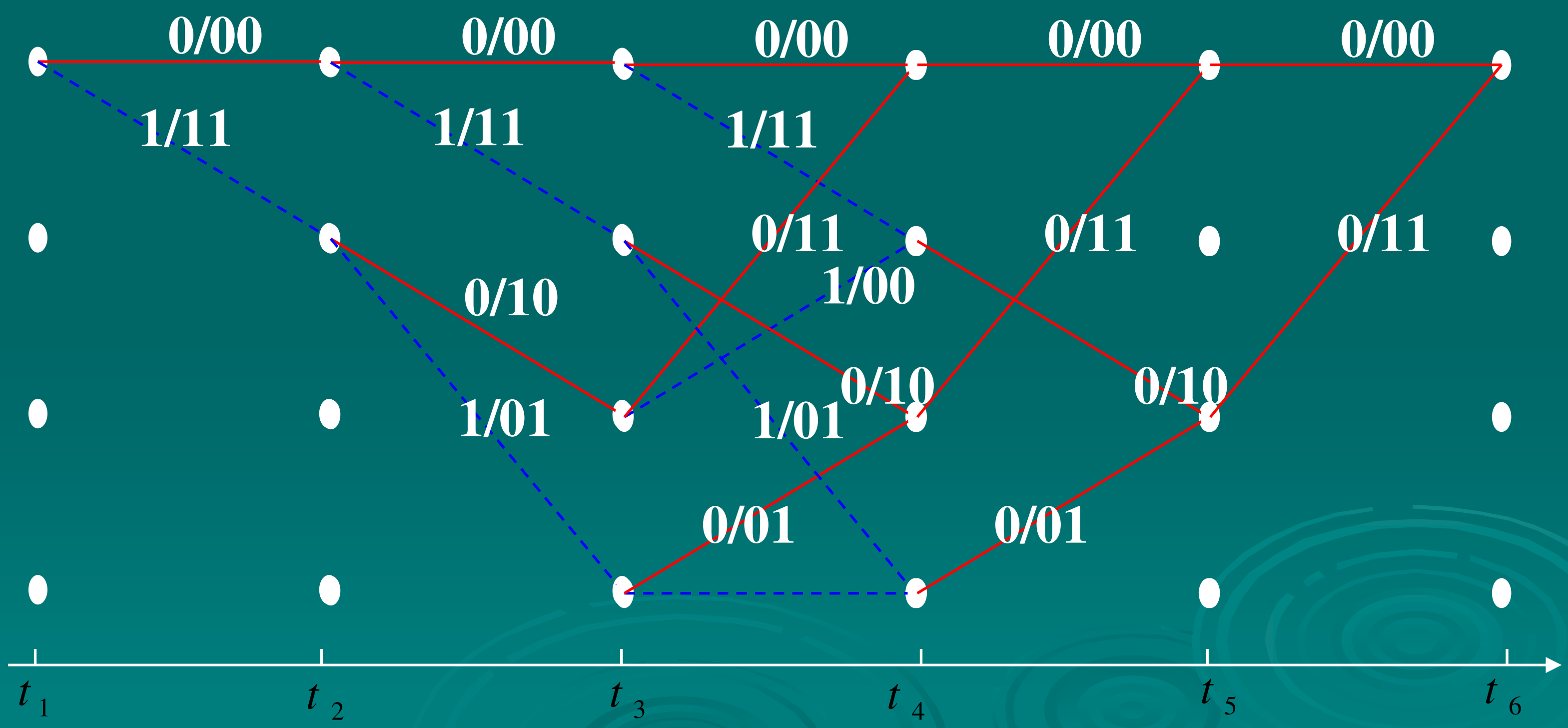


Trellis – cont'd



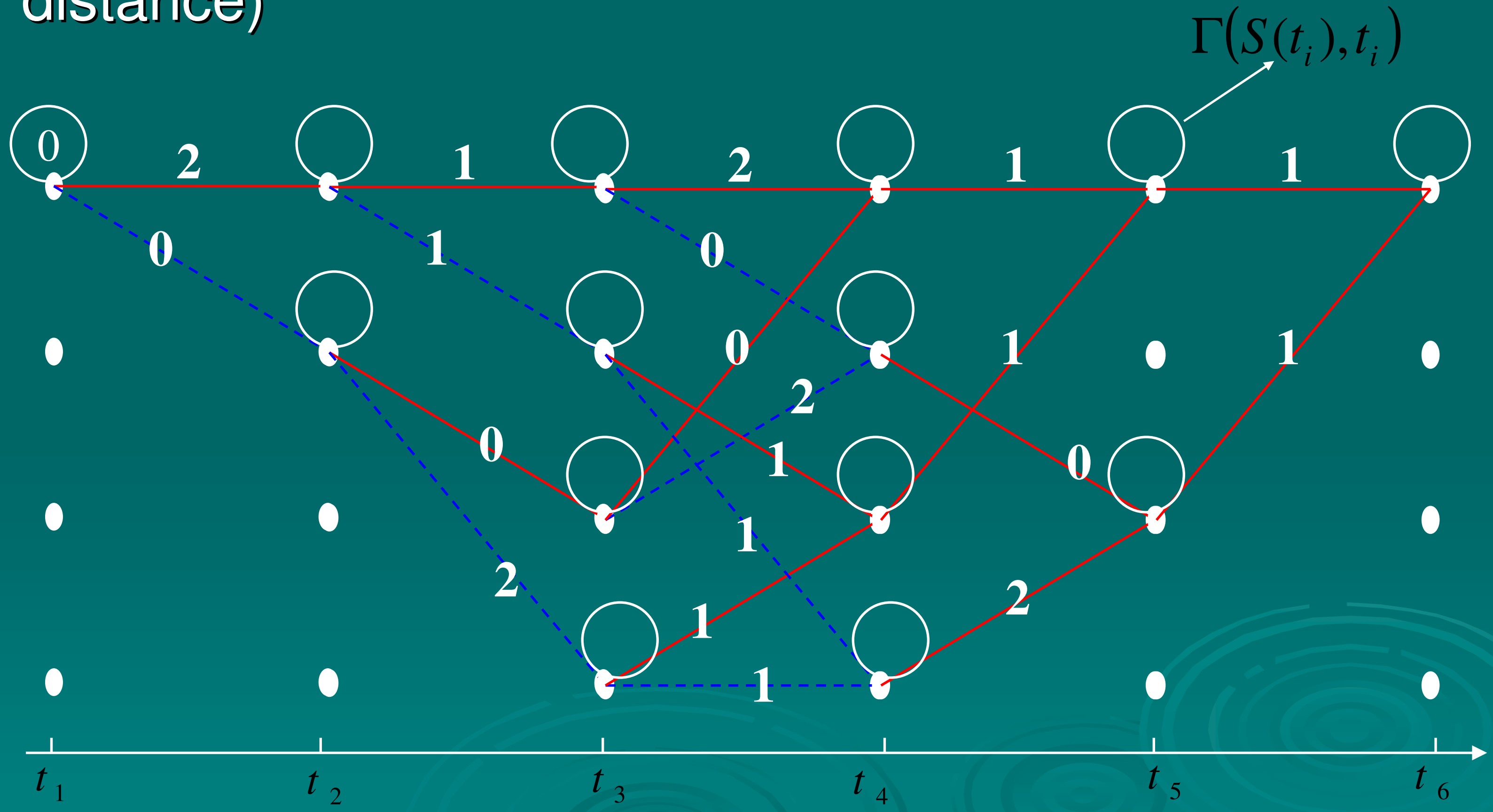
Example of Hard decision Viterbi decoding

$\mathbf{m} = (101)$
$\mathbf{U} = (11 \quad 10 \quad 00 \quad 10 \quad 11)$
$\mathbf{Z} = (11 \quad 10 \quad 11 \quad 10 \quad 01)$



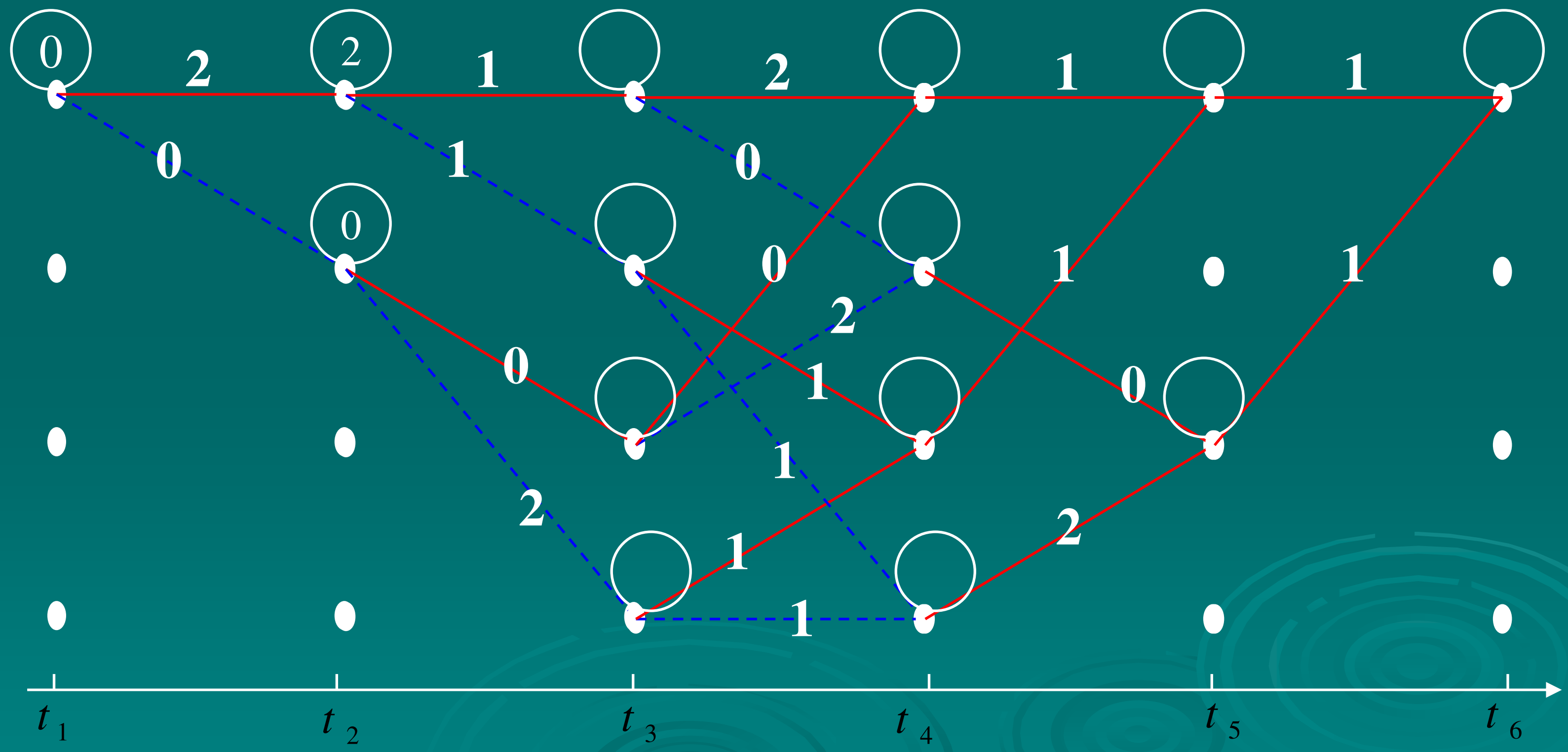
Example of Hard decision Viterbi decoding-cont'd

- Label al the branches with the branch metric (Hamming distance)



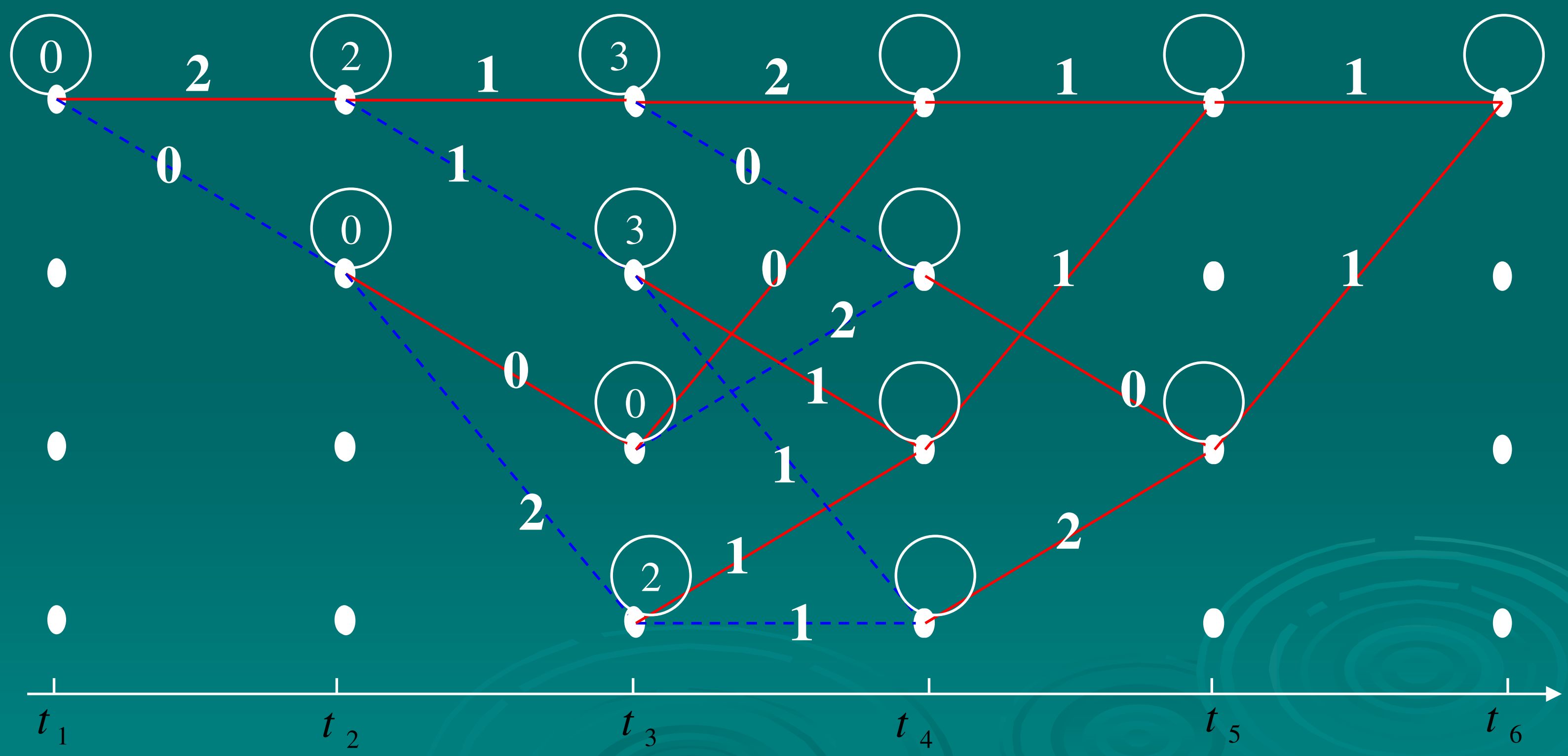
Example of Hard decision Viterbi decoding-cont'd

- i=2



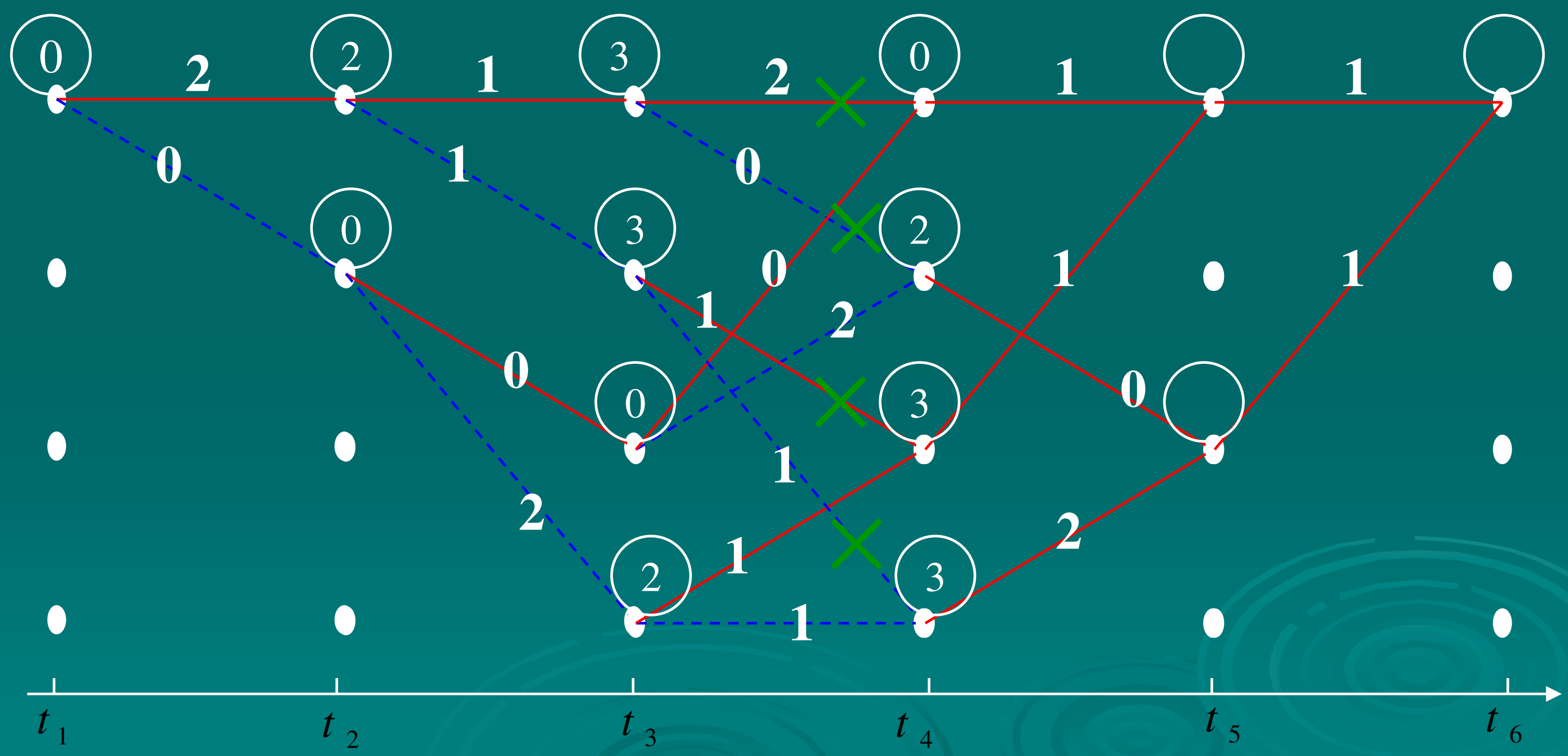
Example of Hard decision Viterbi decoding-cont'd

- i=3



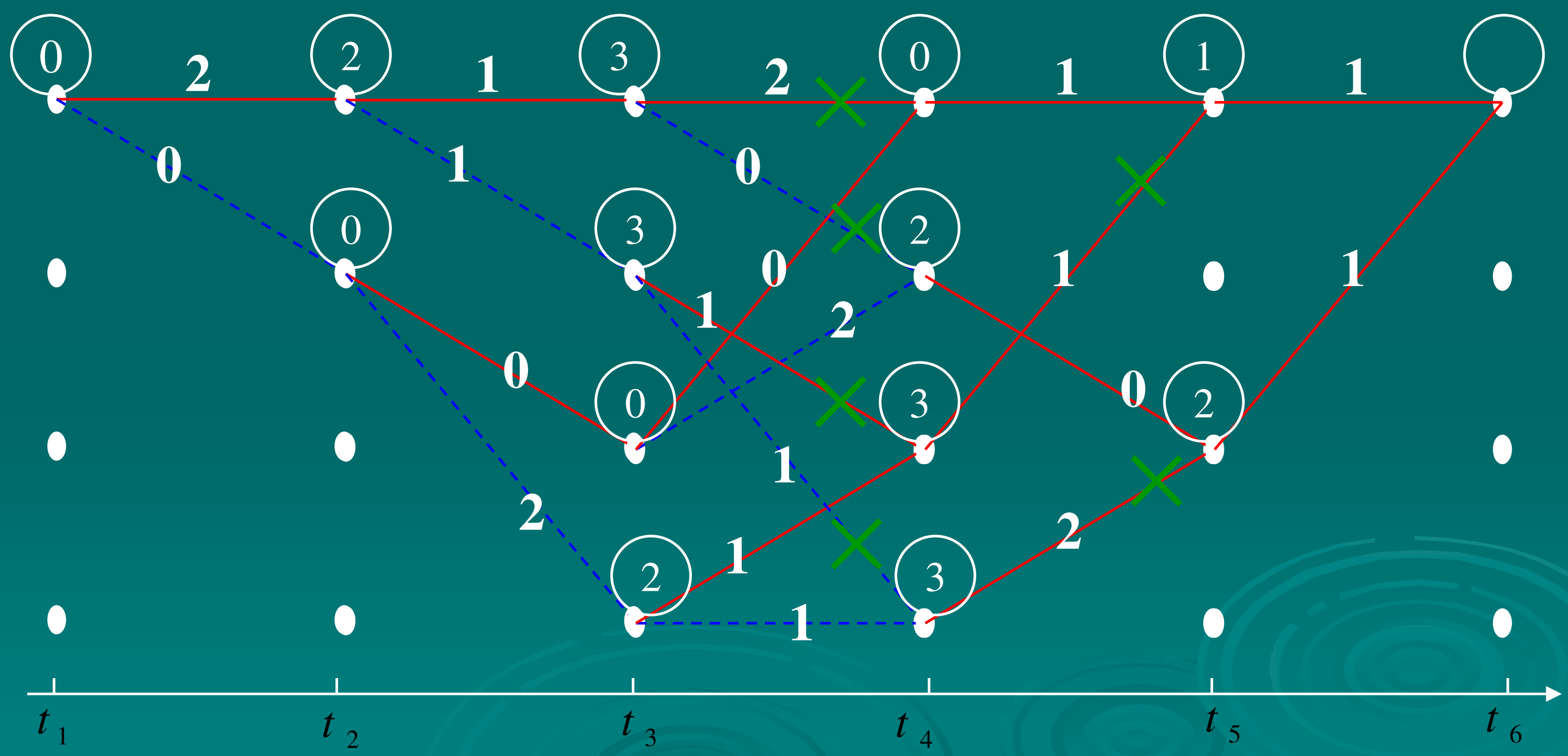
Example of Hard decision Viterbi decoding-cont'd

- i=4



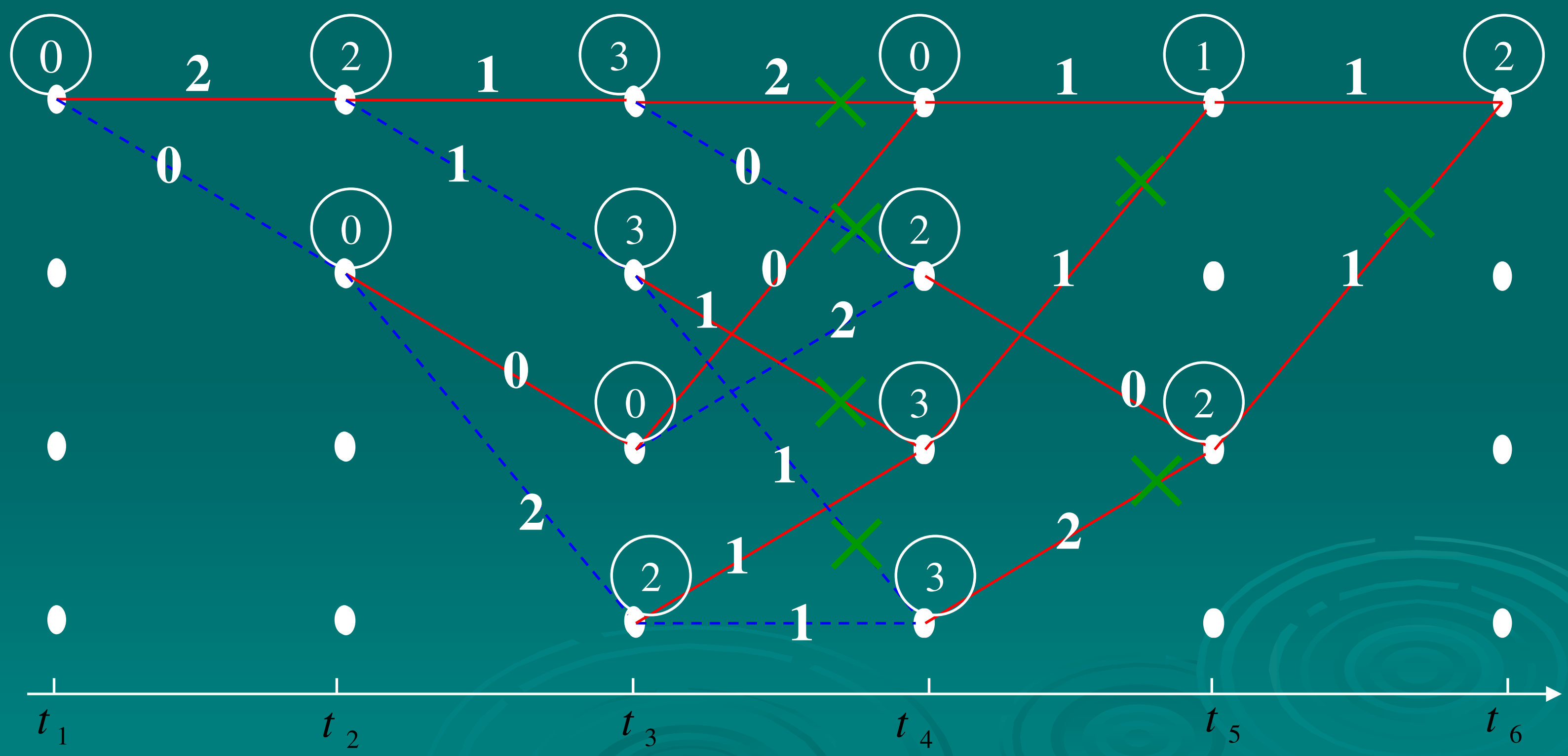
Example of Hard decision Viterbi decoding-cont'd

- i=5



Example of Hard decision Viterbi decoding-cont'd

- i=6



Example of Hard decision Viterbi decoding-cont'd

- Trace back and then:

$$\hat{\mathbf{m}} = (100)$$
$$\hat{\mathbf{U}} = (11 \quad 10 \quad 11 \quad 00 \quad 00)$$