EC 721 Advanced Digital Communications
Spring 2008

Mohamed Essam Khedr

Department of Electronics and Communications

Error correcting codes

http://webmail.aast.edu/~khedr

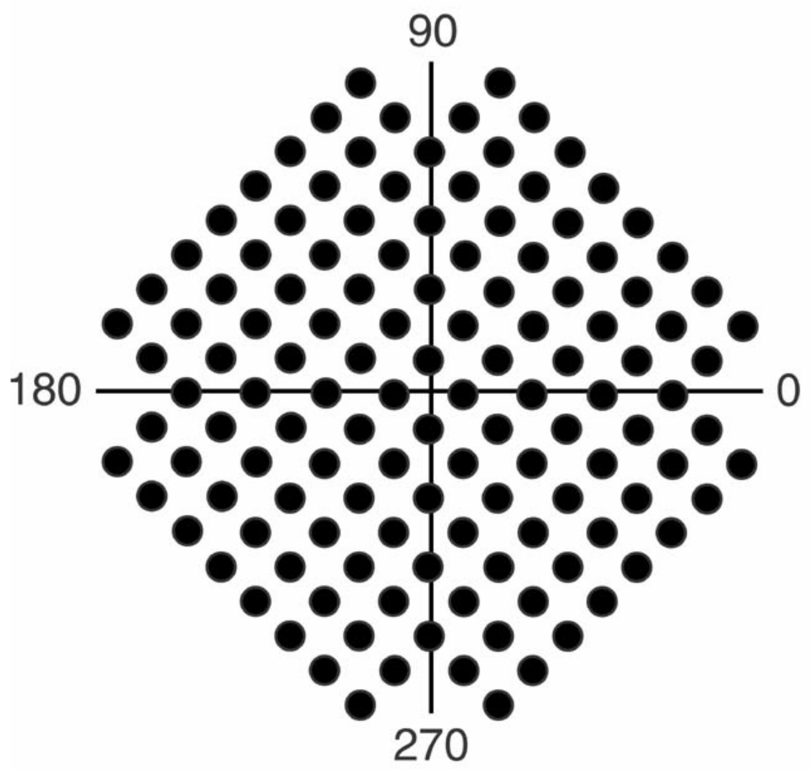
Syllabus

- Tentatively

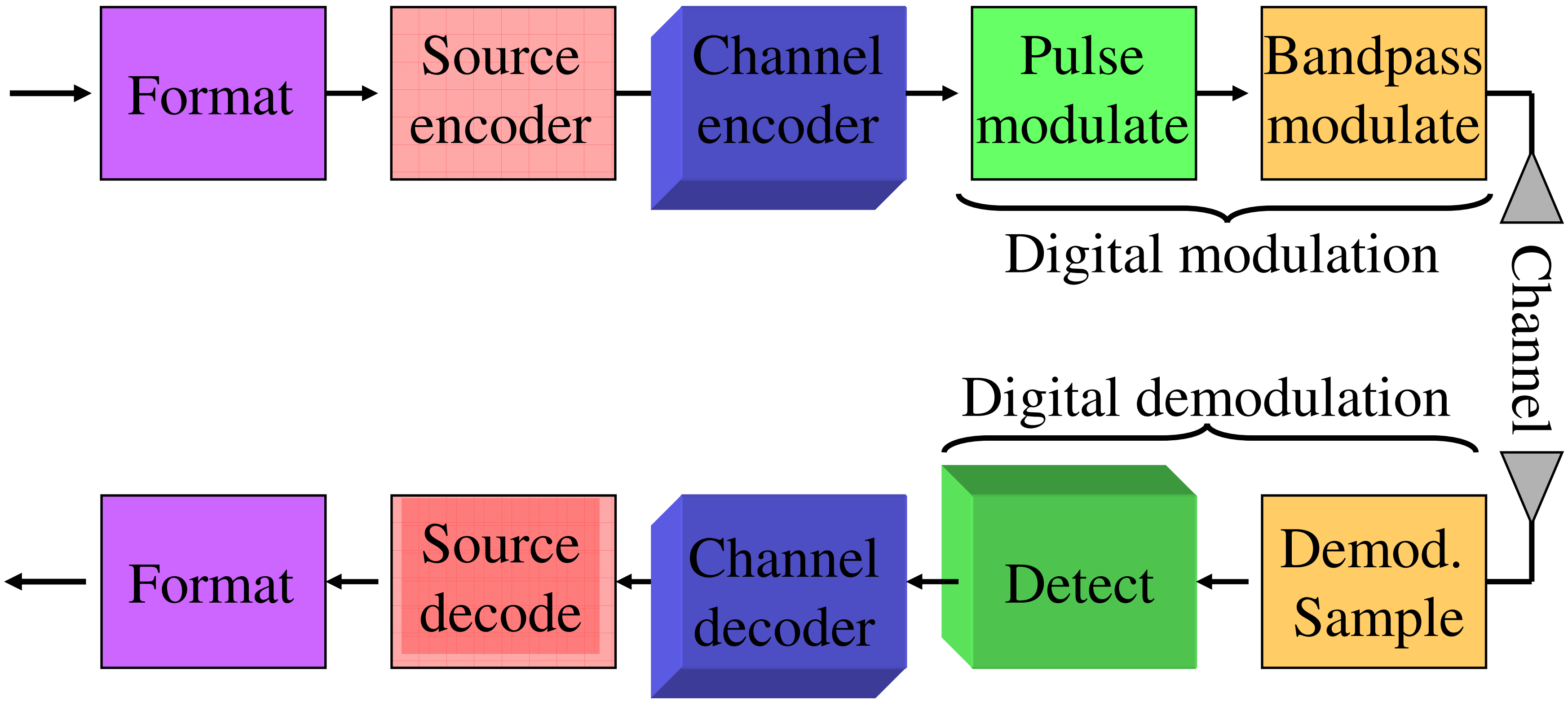
Week 1	Overview, Probabilities, Random variables
Week 2	Random Process, Optimum Detection
Week 3	Digital Signal Representation
Week 4	Signal space and probability of error
Week 5	Probability of error of M-ary techniques
Week 6	Coding theory
Week 7	Linear block codes
Week 8	Convolutional Codes
Week 9	
Week 10	
Week 11	
Week 12	
Week 13	
Week 14	
Week 15	

OOPS OOPS OOOOOOPS
Report and MATLAB ASSIGNMENT

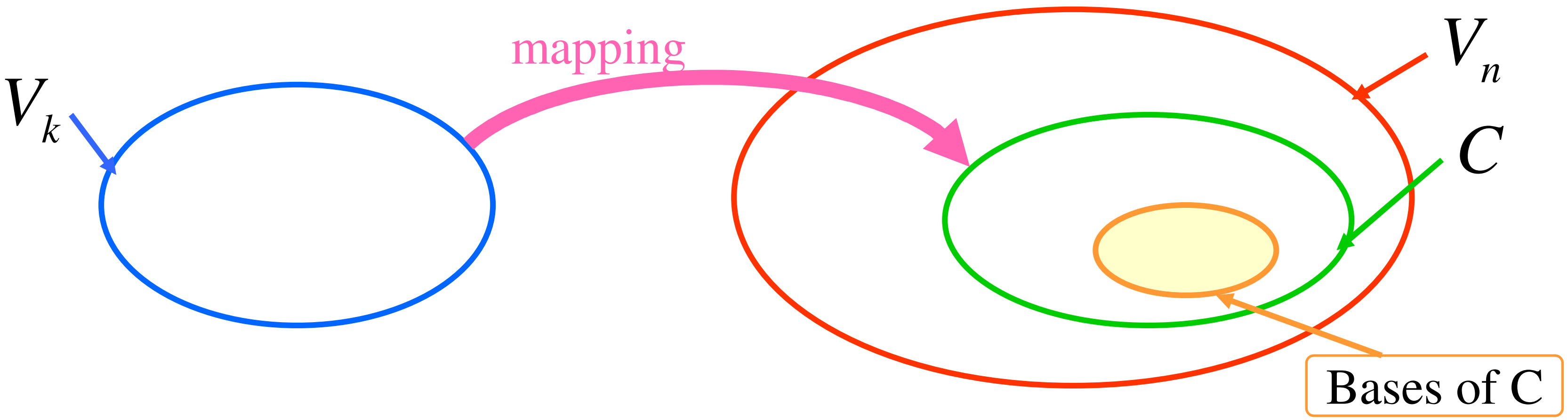
- Soft and hard decisions
- Puncturing
- Interleaving
  - Block
  - Convolutional
- P_{ec} and P_{euc} for V.32
  6 data bits. 1 parity.
  14.4 kbps. modem



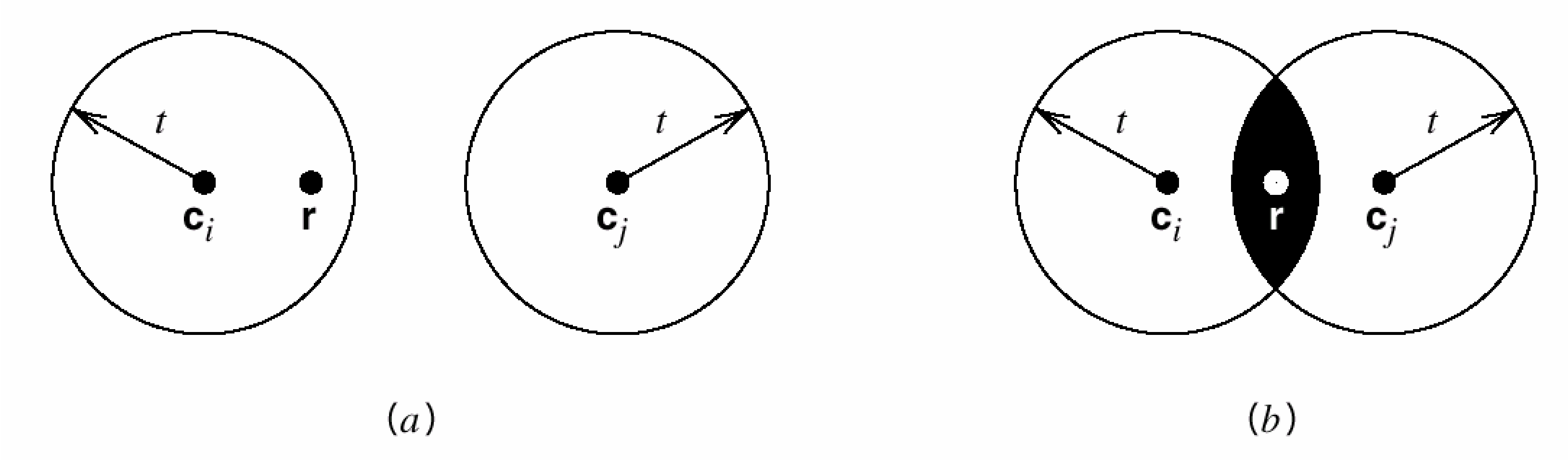
Block diagram of a DCS



Linear block codes – cont'd

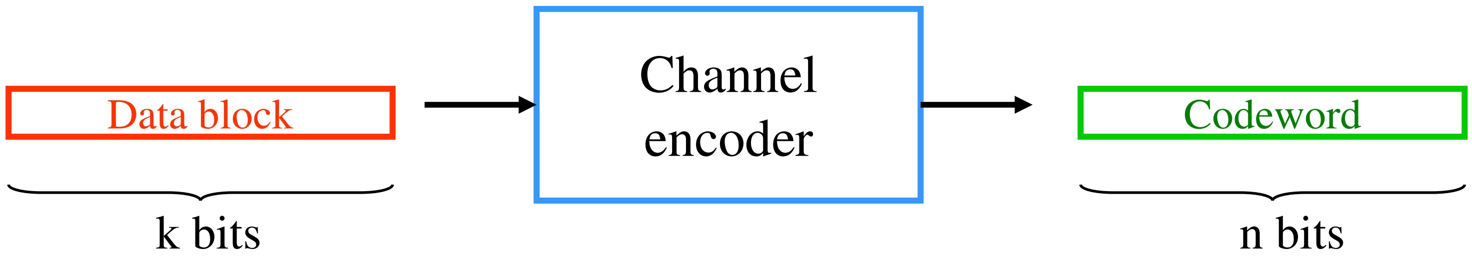


(*a*) Hamming distance $d(\mathbf{c}_i, \mathbf{c}_j) \geq 2t + 1$. (*b*) Hamming distance $d(\mathbf{c}_i, \mathbf{c}_j) < 2t$. The received vector is denoted by **r**.



Linear block codes – cont'd

- The information bit stream is chopped into blocks of k bits.
- Each block is encoded to a larger block of n bits.
- The coded bits are modulated and sent over channel.
- The reverse procedure is done at the receiver.



$n\text{-}k$ Redundant bits

$R_c = \frac{k}{n}$ Code rate

Linear block codes – cont'd

- The Hamming weight of vector **U**, denoted by w(**U**), is the number of non-zero elements in **U**.
- The Hamming distance between two vectors **U** and **V**, is the number of elements in which they differ.

$$d(\mathbf{U},\mathbf{V}) = w(\mathbf{U} \oplus \mathbf{V})$$

- The minimum distance of a block code is

$$d_{\min} = \min_{i \neq j} d(\mathbf{U}_i, \mathbf{U}_j) = \min_i w(\mathbf{U}_i)$$

Linear block codes – cont'd

- Error detection capability is given by

$$e = d_{\min} - 1$$

- Error correcting-capability **t** of a code, which is defined as the maximum number of guaranteed correctable errors per codeword, is

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

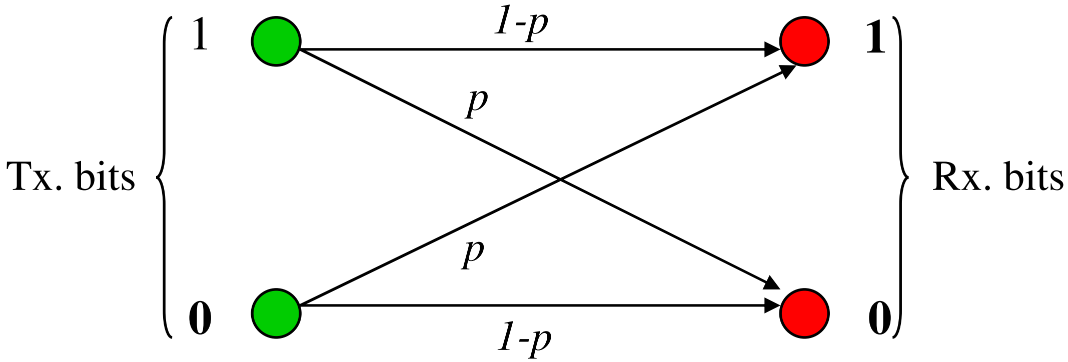
Linear block codes – cont'd

- For memory less channels, the probability that the decoder commits an erroneous decoding is $$P_M \leq \sum_{j=t+1}^{n} \binom{n}{j} p^j (1-p)^{n-j}$$
  - p is the transition probability or bit error probability over channel.
- The decoded bit error probability is

$$P_B \approx \frac{1}{n} \sum_{j=t+1}^{n} j \binom{n}{j} p^j (1-p)^{n-j}$$

Linear block codes – cont'd

- Discrete, memoryless, symmetric channel model

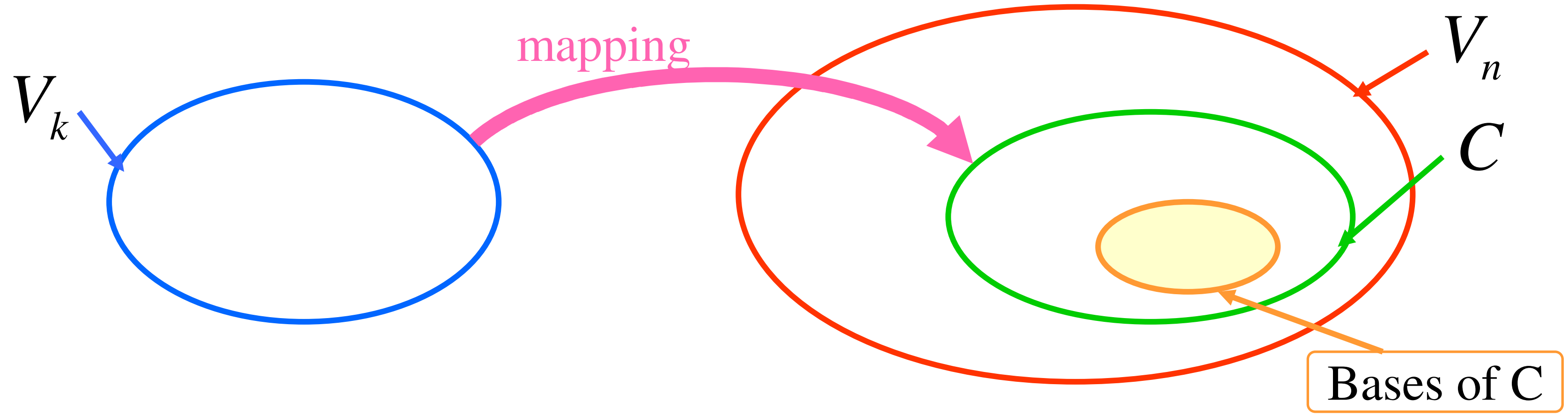


  - Note that for coded systems, the coded bits are modulated and transmitted over channel. For example, for M-PSK modulation on AWGN channels (M>2):

$$p \approx \frac{2}{\log_2 M} Q\left(\sqrt{\frac{2(\log_2 M)E_c}{N_0}} \sin\left(\frac{\pi}{M}\right)\right) = \frac{2}{\log_2 M} Q\left(\sqrt{\frac{2(\log_2 M)E_b R_c}{N_0}} \sin\left(\frac{\pi}{M}\right)\right)$$

where E_c is energy per coded bit, given by $E_c = R_c E_b$

Linear block codes –cont'd



- A matrix G is constructed by taking as its rows the vectors on the basis, $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_k\}$.

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \vdots \\ \mathbf{V}_k \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & & \ddots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kn} \end{bmatrix}$$

Linear block codes – cont'd

- Encoding in (n,k) block code

$$\mathbf{U} = \mathbf{mG}$$

$$(u_1, u_2, \ldots, u_n) = (m_1, m_2, \ldots, m_k) \cdot \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_k \end{bmatrix}$$

$$(u_1, u_2, \ldots, u_n) = m_1 \cdot \mathbf{V}_1 + m_2 \cdot \mathbf{V}_2 + \ldots + m_2 \cdot \mathbf{V}_k$$

  - The rows of G, are linearly independent.

Linear block codes – cont'd

- Example: Block code (6,3)

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Message vector	Codeword
000	000000
100	110100
010	011010
110	101110
001	101001
101	011101
011	110011
111	000111

Linear block codes – cont'd

- Systematic block code (n,k)
  - For a systematic code, the first (or last) k elements in the codeword are information bits.

$$\mathbf{G} = [\mathbf{P} \;\vdots\; \mathbf{I}_k]$$

$$\mathbf{I}_k = k \times k \text{ identity matrix}$$

$$\mathbf{P}_k = k \times (n-k) \quad \text{matrix}$$

$$\mathbf{U} = (u_1, u_2, \ldots, u_n) = (\underbrace{p_1, p_2, \ldots, p_{n-k}}_{\text{parity bits}}, \underbrace{m_1, m_2, \ldots, m_k}_{\text{message bits}})$$

Linear block codes – cont'd

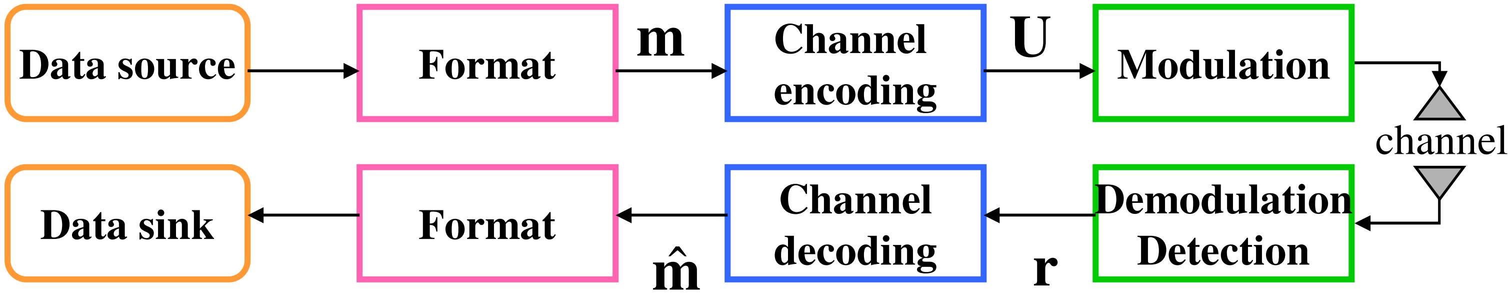
- For any linear code we can find an matrix $\mathbf{H}_{(n-k)\times n}$, which its rows are orthogonal to rows of $\mathbf{G}$:

$$\mathbf{G}\mathbf{H}^T = \mathbf{0}$$

- **H** is called the parity check matrix and its rows are linearly independent.
- For systematic linear block codes:

$$\mathbf{H} = [\mathbf{I}_{n-k} \;\vdots\; \mathbf{P}^T]$$

Linear block codes – cont'd



$$\mathbf{r} = \mathbf{U} + \mathbf{e}$$

$\mathbf{r} = (r_1, r_2, ..., r_n)$ received codeword or vector

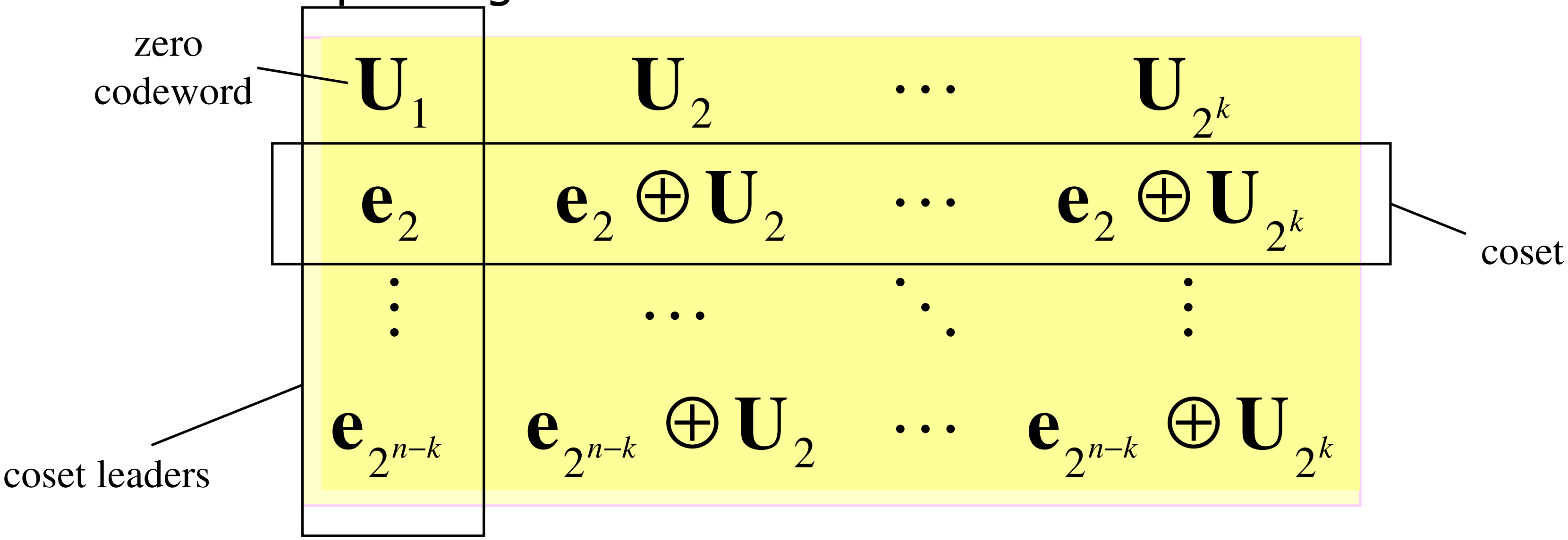
$\mathbf{e} = (e_1, e_2, ..., e_n)$ error pattern or vector

- Syndrome testing:
  - **S** is syndrome of **r**, corresponding to the error pattern **e**.

$$\mathbf{S} = \mathbf{r}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

Linear block codes – cont'd

- Standard array
  1. For row $i = 2,3,\ldots,2^{n-k}$, find a vector in V_n of minimum weight which is not already listed in the array.
  2. Call this pattern $\mathbf{e}_i$ and form the $i:\mathrm{th}$ row as the corresponding coset



$$\begin{matrix} \mathbf{U}_1 & \mathbf{U}_2 & \cdots & \mathbf{U}_{2^k} \\ \mathbf{e}_2 & \mathbf{e}_2 \oplus \mathbf{U}_2 & \cdots & \mathbf{e}_2 \oplus \mathbf{U}_{2^k} \\ \vdots & \cdots & \ddots & \vdots \\ \mathbf{e}_{2^{n-k}} & \mathbf{e}_{2^{n-k}} \oplus \mathbf{U}_2 & \cdots & \mathbf{e}_{2^{n-k}} \oplus \mathbf{U}_{2^k} \end{matrix}$$

Linear block codes – cont'd

- Standard array and syndrome table decoding
  1. Calculate $\mathbf{S} = \mathbf{r}\mathbf{H}^T$
  2. Find the coset leader, $\hat{\mathbf{e}} = \mathbf{e}_i$, corresponding to $\mathbf{S}$.
  3. Calculate $\hat{\mathbf{U}} = \mathbf{r} + \hat{\mathbf{e}}$ and corresponding $\hat{\mathbf{m}}$.

- Note that $\hat{\mathbf{U}} = \mathbf{r} + \hat{\mathbf{e}} = (\mathbf{U} + \mathbf{e}) + \hat{\mathbf{e}} = \mathbf{U} + (\mathbf{e} + \hat{\mathbf{e}})$
  - If $\hat{\mathbf{e}} = \mathbf{e}$, error is corrected.
  - If $\hat{\mathbf{e}} \neq \mathbf{e}$, undetectable decoding error occurs.

Linear block codes – cont’d

- Example: Standard array for the (6,3) code

codewords

000000	110100	011010	101110	101001	011101	110011	000111
000001	110101	011011	101111	101000	011100	110010	000110
000010	110110	011000	101100	101011	011111	110001	000101
000100	110000	011100	101010	101101	011010	110111	000110
001000	111100	⋮			⋮		⋮
010000	100100						
100000	010100				⋮		
010001	100101		…			…	010110

coset

Coset leaders

Linear block codes – cont'd

Error pattern	Syndrome
000000	000
000001	101
000010	011
000100	110
001000	001
010000	010
100000	100
010001	111

$\mathbf{U} = (101110)$ transmitted.

$\mathbf{r} = (001110)$ is received.

- The syndrome of $\mathbf{r}$ is computed:

  $\mathbf{S} = \mathbf{r}\mathbf{H}^T = (001110)\mathbf{H}^T = (100)$

- Error pattern corresponding to this syndrome is

  $\hat{\mathbf{e}} = (100000)$

- The corrected vector is estimated

  $\hat{\mathbf{U}} = \mathbf{r} + \hat{\mathbf{e}} = (001110) + (100000) = (101110)$

Hamming codes

- Hamming codes
  - Hamming codes are a subclass of linear block codes and belong to the category of *perfect codes*.
  - Hamming codes are expressed as a function of a single integer $m \geq 2$.

Code length :	$n = 2^m - 1$
Number of information bits :	$k = 2^m - m - 1$
Number of parity bits :	$n\text{-}k = m$
Error correction capability :	$t = 1$

  - The columns of the parity-check matrix, **H**, consist of all non-zero binary m-tuples.

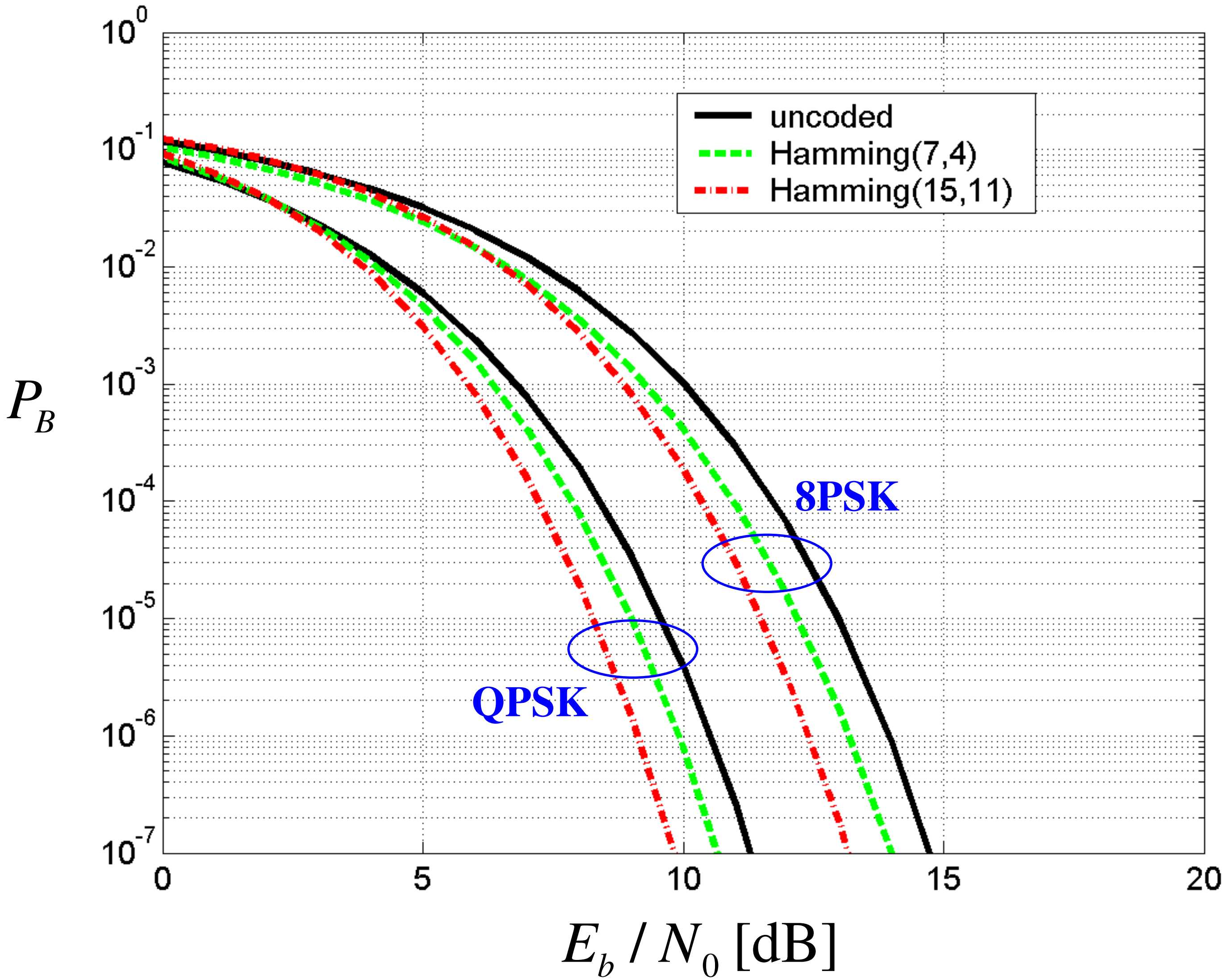
Hamming codes

- Example: Systematic Hamming code (7,4)

$$\mathbf{H} = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right] = [\mathbf{I}_{3\times 3} \;\vdots\; \mathbf{P}^T]$$

$$\mathbf{G} = \left[\begin{array}{ccc|cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] = [\mathbf{P} \;\vdots\; \mathbf{I}_{4\times 4}]$$

Example of the block codes



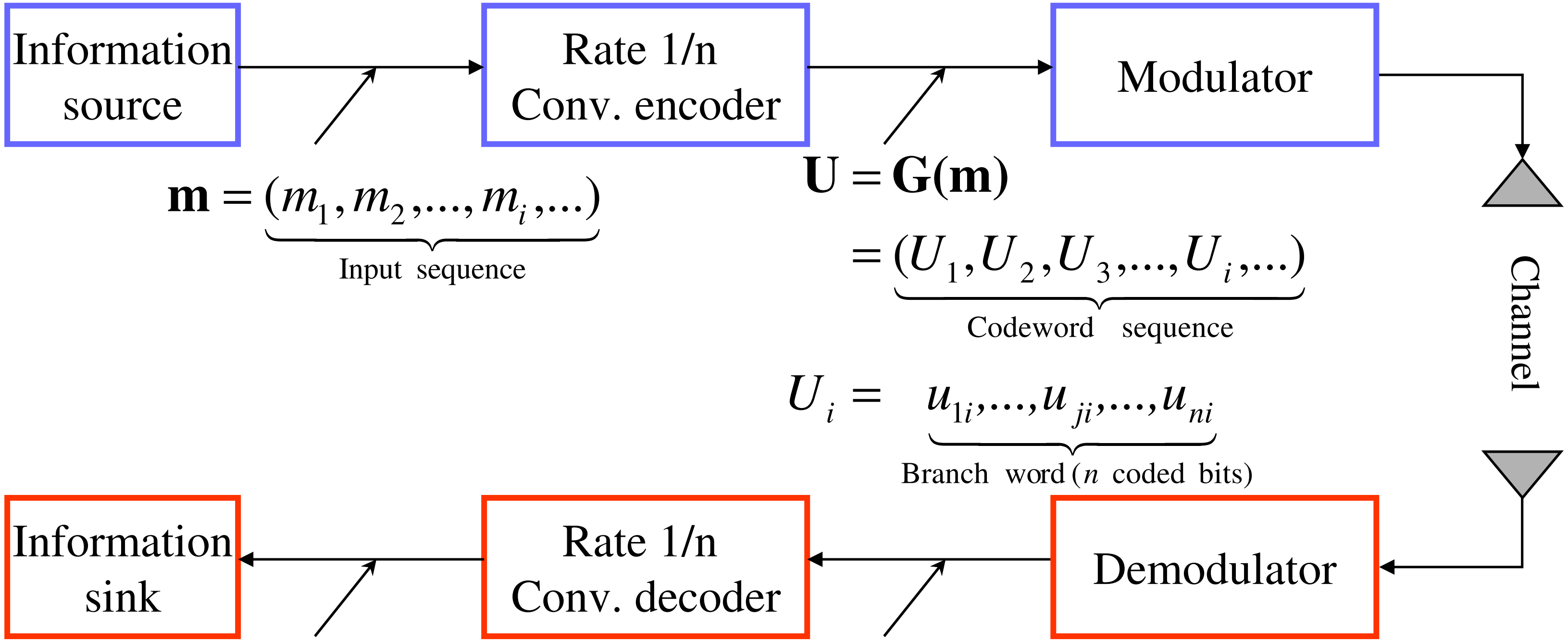
Convolutional codes

- 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 $K\text{-}1$ memory elements.
    - There is different definitions in literatures for constraint length.

Block diagram of the DCS



$$\mathbf{m} = \underbrace{(m_1, m_2, \ldots, m_i, \ldots)}_{\text{Input sequence}}$$

$$\mathbf{U} = \mathbf{G}(\mathbf{m}) = \underbrace{(U_1, U_2, U_3, \ldots, U_i, \ldots)}_{\text{Codeword sequence}}$$

$$U_i = \underbrace{u_{1i}, \ldots, u_{ji}, \ldots, u_{ni}}_{\text{Branch word } (n \text{ coded bits})}$$

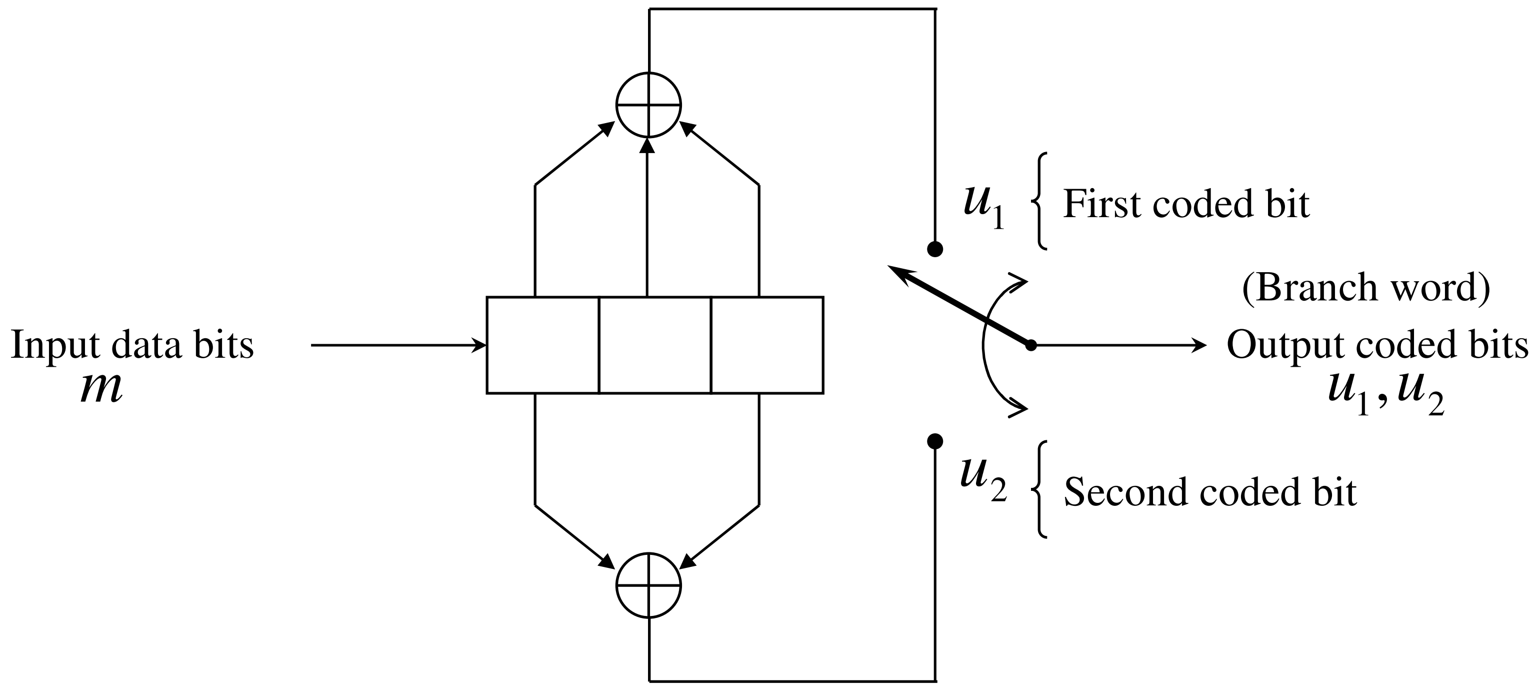
$$\hat{\mathbf{m}} = (\hat{m}_1, \hat{m}_2, \ldots, \hat{m}_i, \ldots)$$

$$\mathbf{Z} = \underbrace{(Z_1, Z_2, Z_3, \ldots, Z_i, \ldots)}_{\text{received sequence}}$$

$$\underbrace{Z_i}_{\substack{\text{Demodulator outputs} \\ \text{for Branch word } i}} = \underbrace{z_{1i}, \ldots, z_{ji}, \ldots, z_{ni}}_{n \text{ outputs per Branch word}}$$

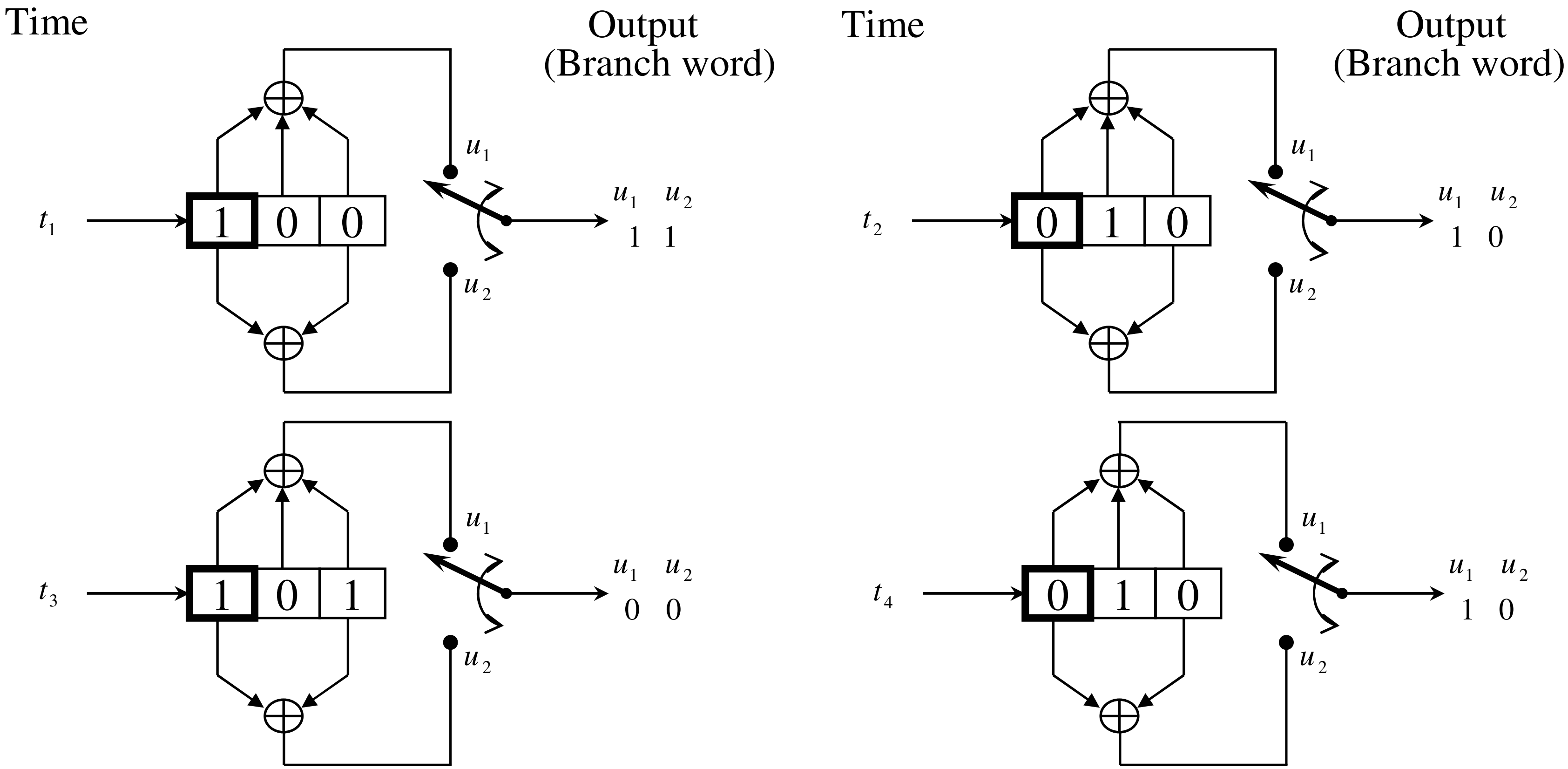
A Rate ½ Convolutional encoder

- 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.

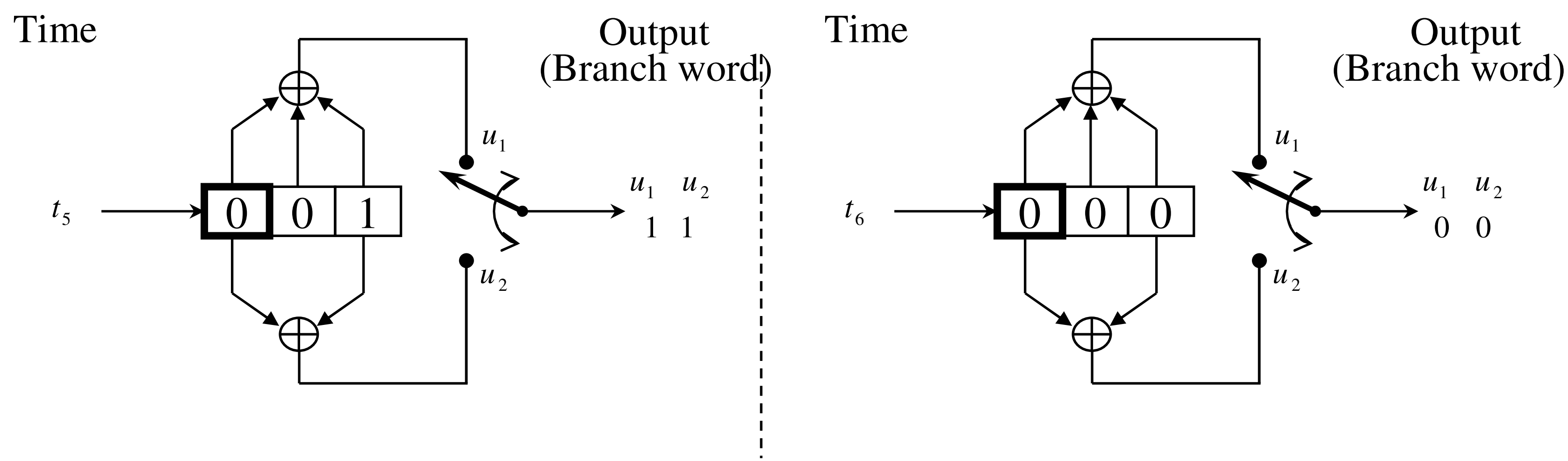


A Rate ½ Convolutional encoder

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



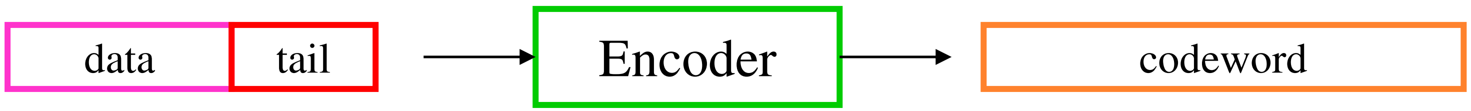
A Rate ½ Convolutional encoder



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

Effective code rate

- Initialize the memory before encoding the first bit (all-zero)
- Clear out the memory after encoding the last bit (all-zero)
  - Hence, a tail of zero-bits is appended to data bits.



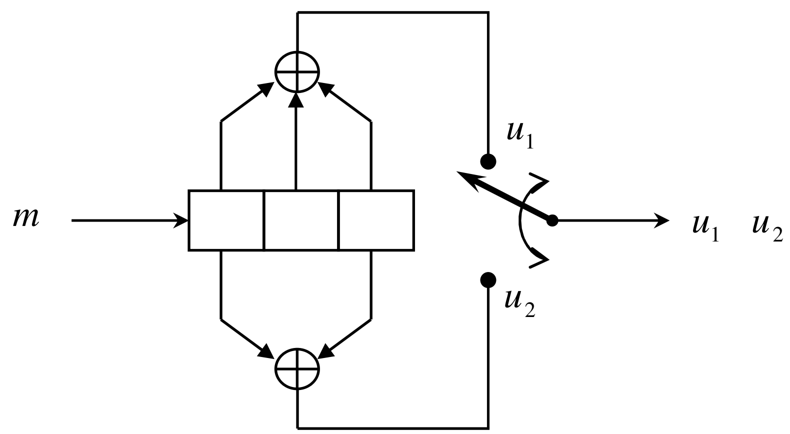
- Effective code rate :
  - L is the number of data bits and $k=1$ is assumed:

$$R_{eff} = \frac{L}{n(L+K-1)} < R_c$$

Encoder representation

- Vector representation:
  - We define n binary vector with K elements (one vector for each modulo-2 adder). The i:th element in each vector, is "1" if the i:th stage in the shift register is connected to the corresponding modulo-2 adder, and "0" otherwise.
    - Example:

$$\mathbf{g}_1 = (111)$$
$$\mathbf{g}_2 = (101)$$



Encoder representation – cont'd

- Impulse response representation:
  - The response of encoder to a single "one" bit that goes through it.
    - Example:

Register contents	Branch word u_1	u_2
100	1	1
010	1	0
001	1	1

$$\text{Input sequence}: \quad 1 \quad 0 \quad 0$$
$$\text{Output sequence}: \quad 11 \quad 10 \quad 11$$

Input **m**	Output				
1	11	10	11		
0		00	00	00	
1			11	10	11
Modulo-2 sum:	11	10	00	10	11

Encoder representation – cont'd

- Polynomial representation:
  - We define n generator polynomials, one for each modulo-2 adder. Each polynomial is of degree *K-1* or less and describes the connection of the shift registers to the corresponding modulo-2 adder.
    - Example:

$$\mathbf{g}_1(X) = g_0^{(1)} + g_1^{(1)}.X + g_2^{(1)}.X^2 = 1 + X + X^2$$

$$\mathbf{g}_2(X) = g_0^{(2)} + g_1^{(2)}.X + g_2^{(2)}.X^2 = 1 + X^2$$

The output sequence is found as follows:

$$\mathbf{U}(X) = \mathbf{m}(X)\mathbf{g}_1(X) \text{ interlaced with } \mathbf{m}(X)\mathbf{g}_2(X)$$

Encoder representation –cont'd

In more details:

$$\mathbf{m}(X)\mathbf{g}_1(X) = (1+X^2)(1+X+X^2) = 1+X+X^3+X^4$$

$$\mathbf{m}(X)\mathbf{g}_2(X) = (1+X^2)(1+X^2) = 1+X^4$$

$$\mathbf{m}(X)\mathbf{g}_1(X) = 1+X+0.X^2+X^3+X^4$$

$$\mathbf{m}(X)\mathbf{g}_2(X) = 1+0.X+0.X^2+0.X^3+X^4$$

$$\mathbf{U}(X) = (1,1)+(1,0)X+(0,0)X^2+(1,0)X^3+(1,1)X^4$$

$$\mathbf{U} = 11 \quad 10 \quad 00 \quad 10 \quad 11$$

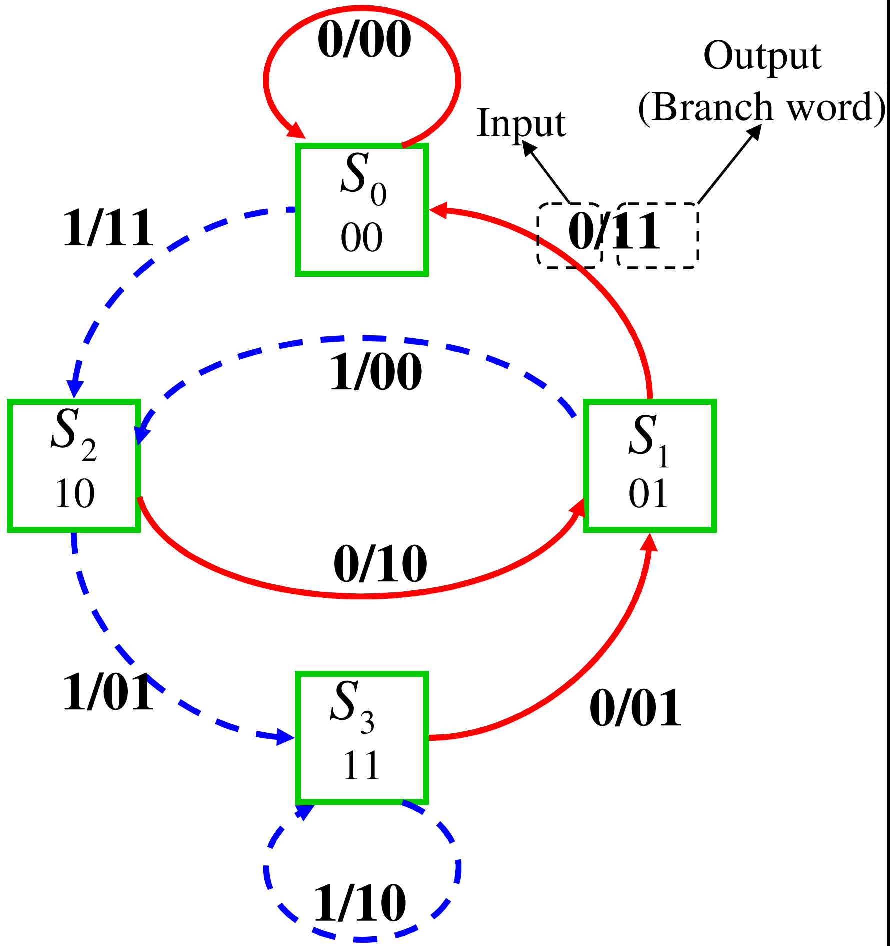
State diagram

- 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

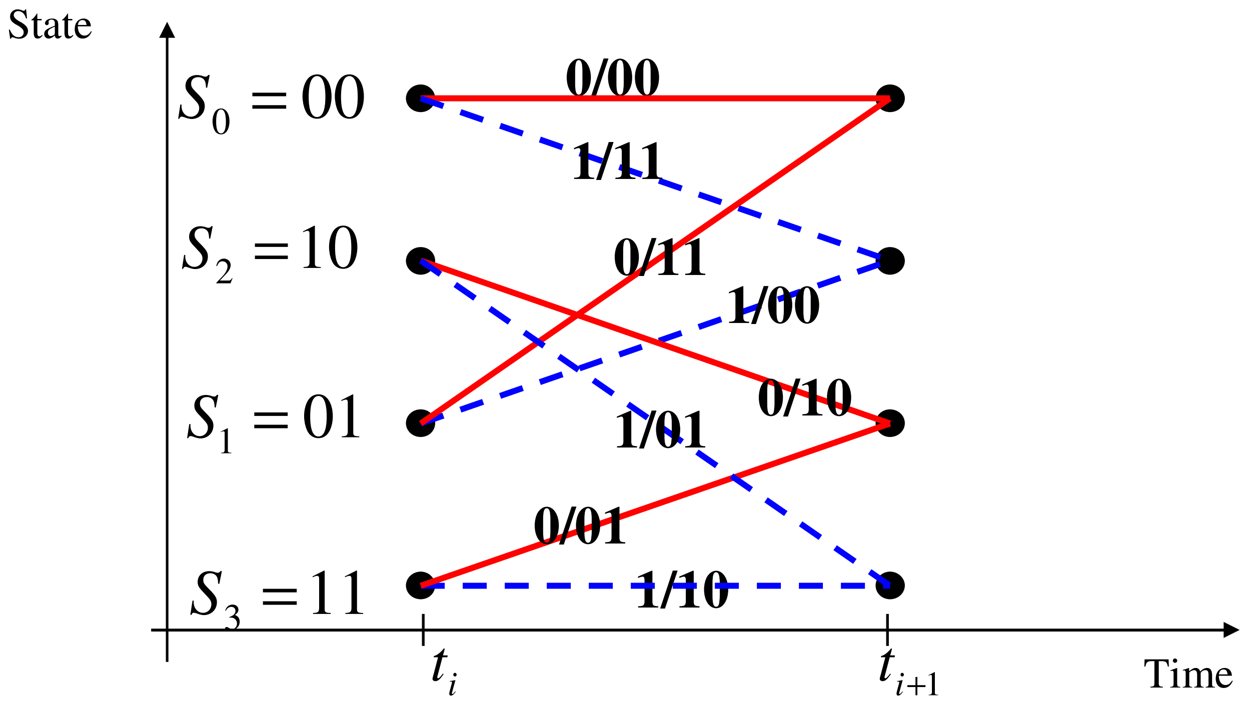
State diagram – cont'd



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

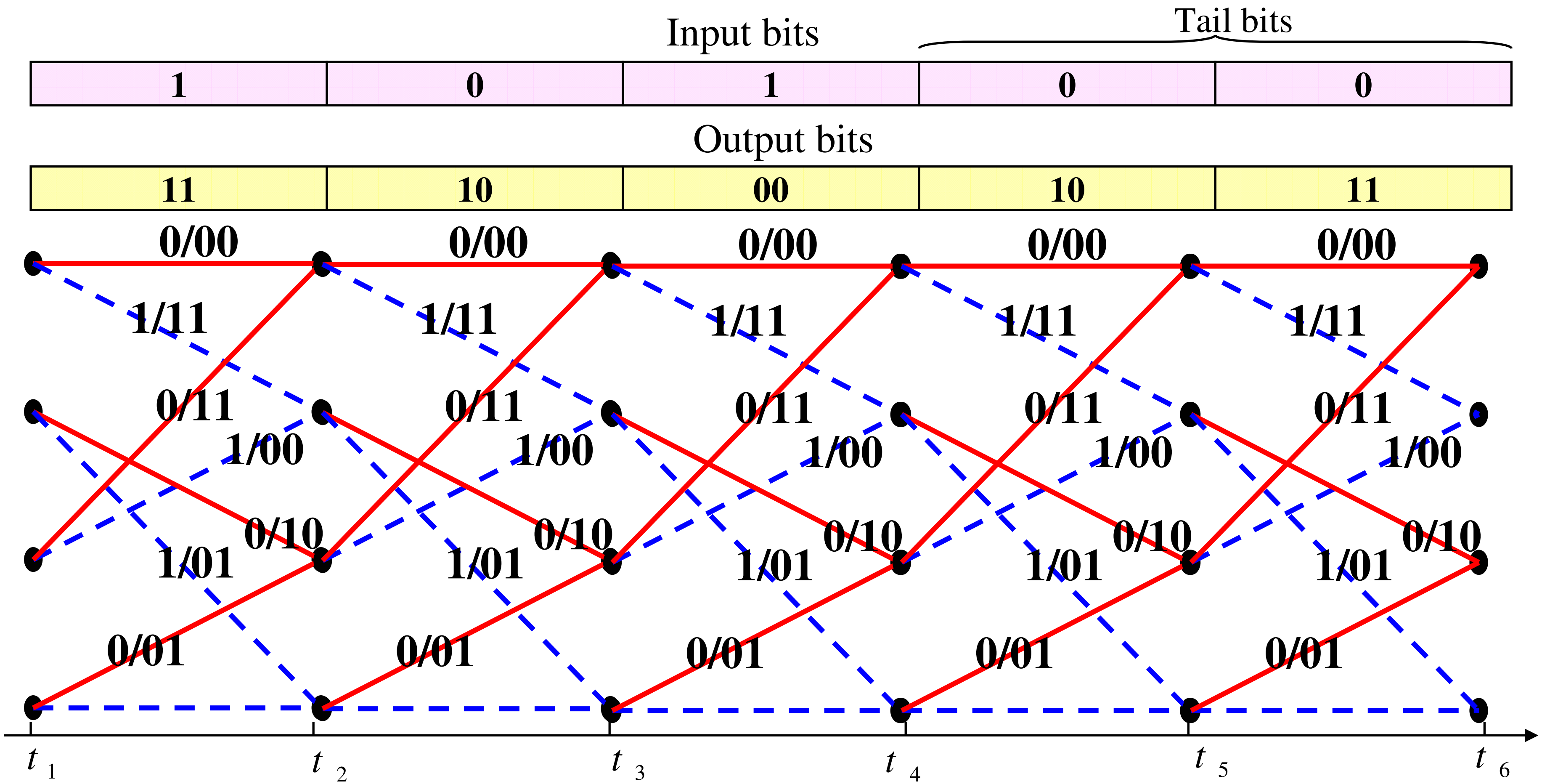
Trellis – cont'd

- 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

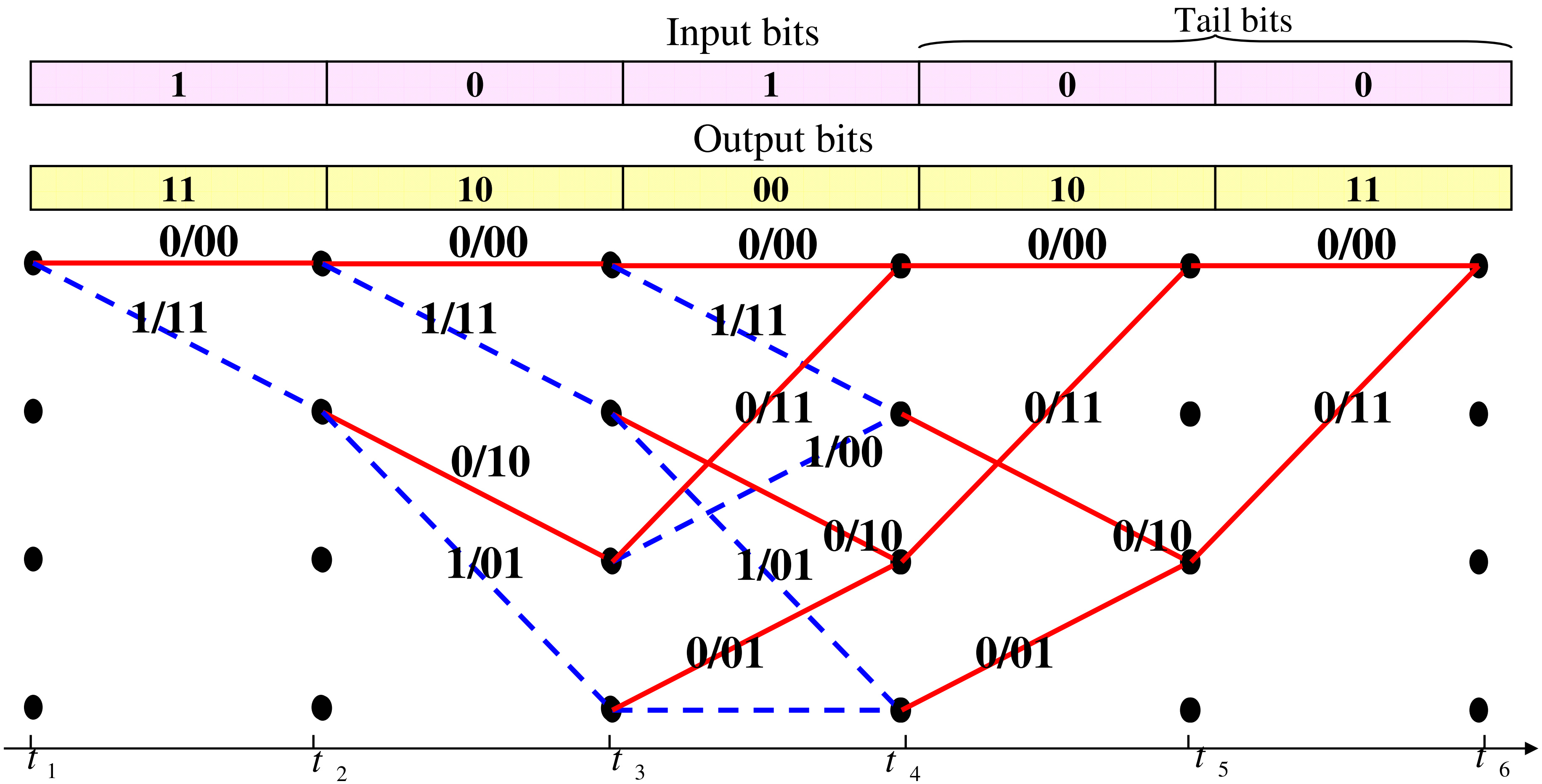


Trellis –cont'd

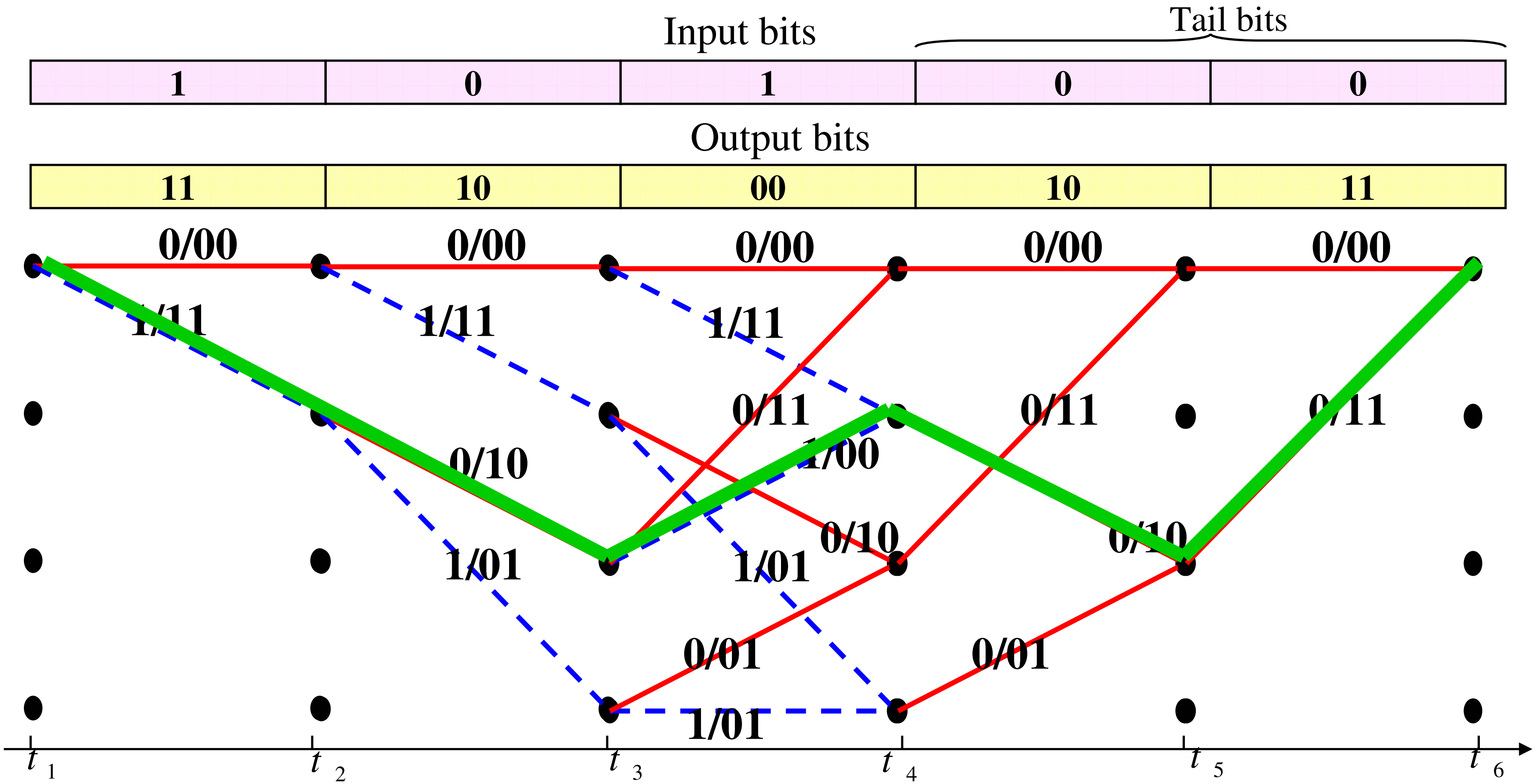
- A trellis diagram for the example code



Trellis – cont'd



Trellis of an example ½ Conv. code



Soft and hard decision decoding

- In hard decision:
  - The demodulator makes a firm or hard decision whether one or zero is transmitted and provides no other information for the decoder such that how reliable the decision is.

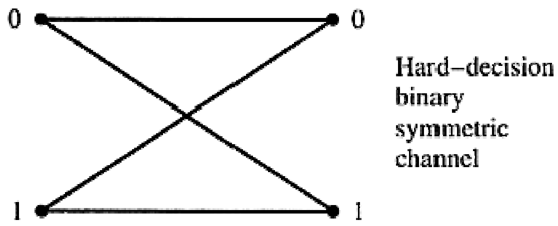
- In Soft decision:
  - The demodulator provides the decoder with some side information together with the decision. The side information provides the decoder with a measure of confidence for the decision.

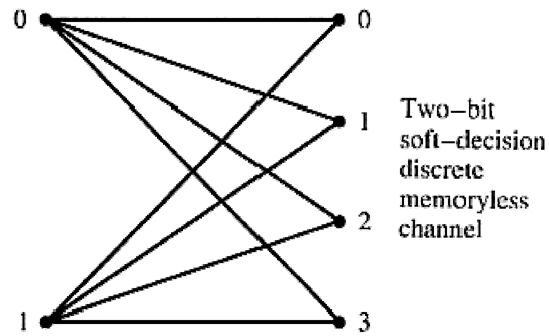
Soft and hard decoding

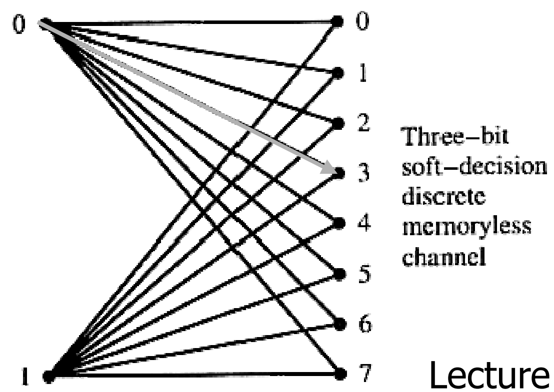
- Regardless whether the channel outputs hard or soft decisions the decoding rule remains the same: maximize the probability

$$\ln p(\mathbf{y},\mathbf{x}_m) = \sum_{j=0}^{\infty} \ln p(y_j \mid x_{mj})$$

- However, in soft decoding **decision region energies** must be accounted for, and hence Euclidean metric d_E, rather that Hamming metric d_{free} is used





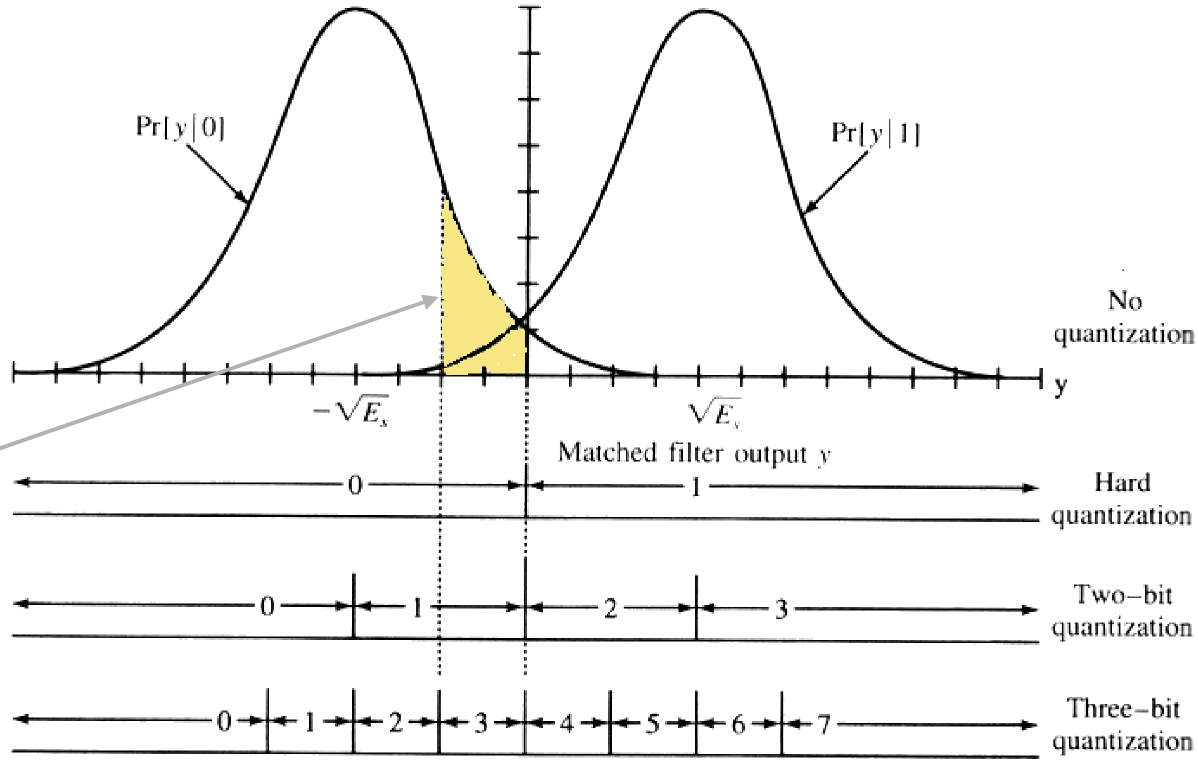


Transition for Pr[3|0] is indicated by the arrow

Decision regions

- Coding can be realized by soft-decoding or hard-decoding principle
- For soft-decoding reliability (measured by bit-energy) of decision region must be known
- Example: decoding BPSK-signal: Matched filter output is a continuos number. In AWGN matched filter output is Gaussian
- For soft-decoding several decision region partitions are used

Transition probability for Pr[3|0], e.g. prob. that transmitted '0' falls into region no: 3



Soft and hard decision decoding …

- ML soft-decisions decoding rule:
  - Choose the path in the trellis with minimum Euclidean distance from the received sequence

- ML hard-decisions decoding rule:
  - Choose the path in the trellis with minimum Hamming distance from the received sequence

The Viterbi algorithm

- The Viterbi algorithm performs Maximum likelihood decoding.

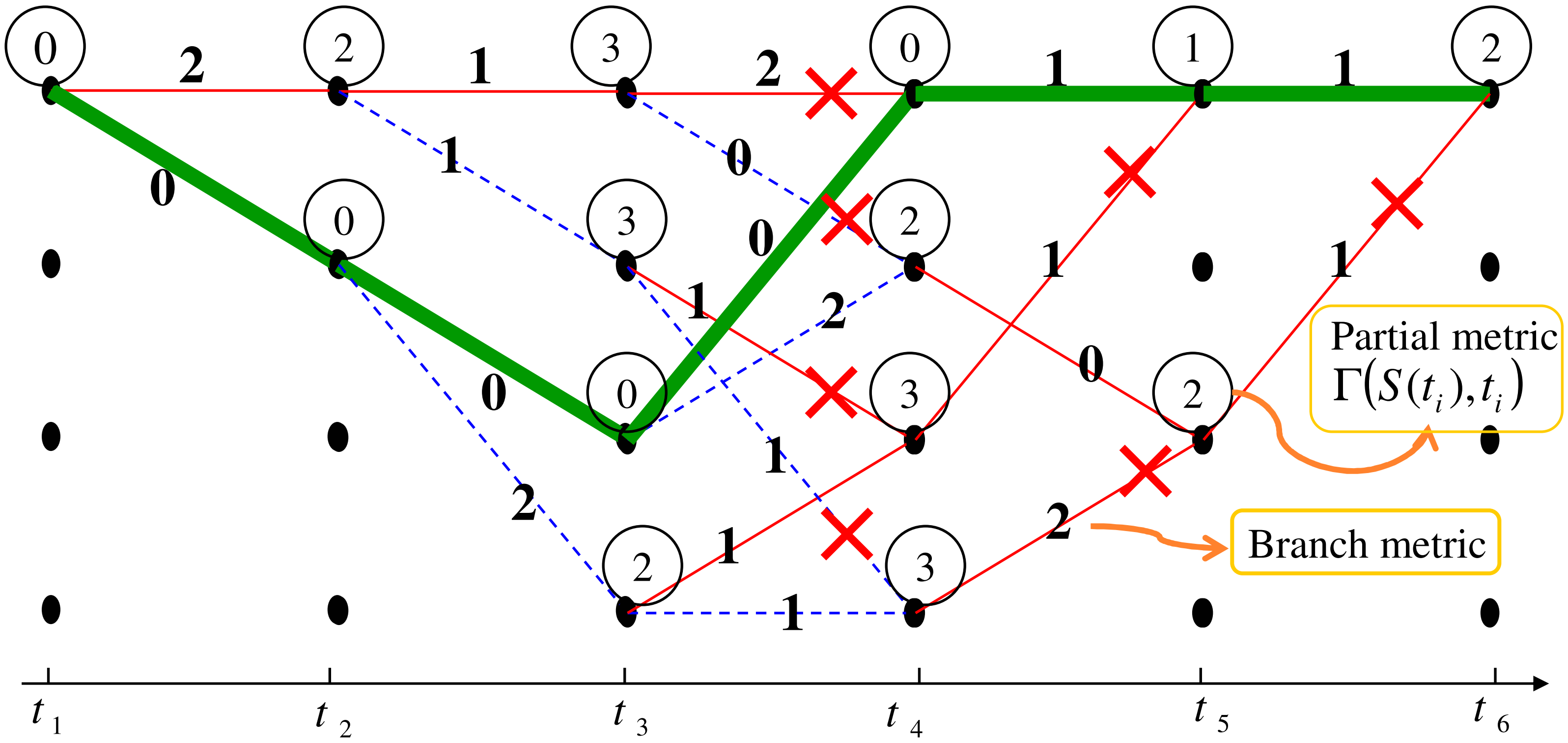
- It finds a path through trellis with the largest metric (maximum correlation or minimum distance).
  - At each step in the trellis, it compares the partial metric of all paths entering each state, and keeps only the path with the largest metric, called the survivor, together with its metric.

Example of hard-decision Viterbi decoding

$\mathbf{Z} = (11 \quad 10 \quad 11 \quad 10 \quad 01)$

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

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



Interleaving

- Convolutional codes are suitable for memoryless channels with random error events.

- Some errors have bursty nature:
  - Statistical dependence among successive error events (time-correlation) due to the channel memory.
    - Like errors in multipath fading channels in wireless communications, errors due to the switching noise, …

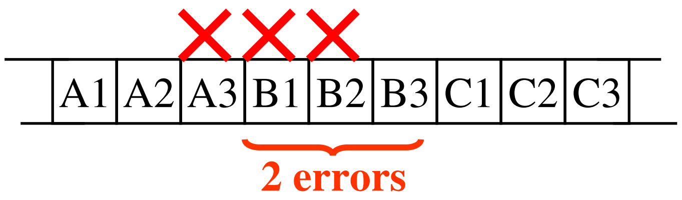
- “Interleaving” makes the channel looks like as a memoryless channel at the decoder.

Interleaving …

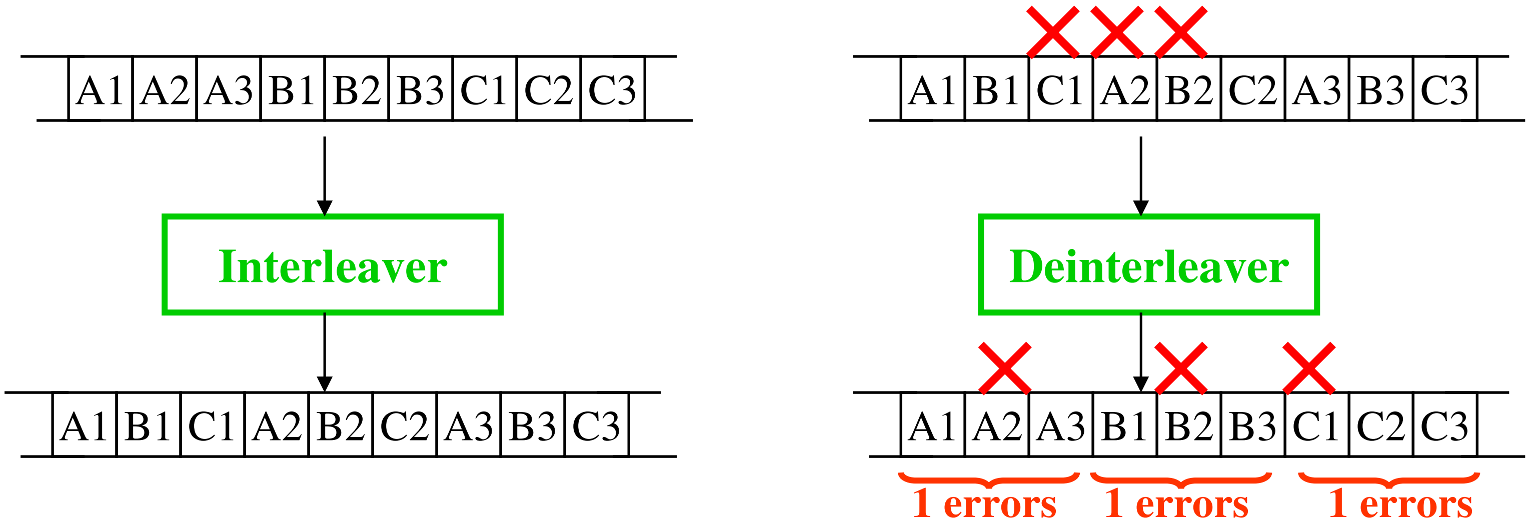
- Interleaving is done by spreading the coded symbols in time (interleaving) before transmission.
- The reverse in done at the receiver by deinterleaving the received sequence.
- "Interleaving" makes bursty errors look like random. Hence, Conv. codes can be used.
- Types of interleaving:
  - Block interleaving
  - Convolutional or cross interleaving

Interleaving …

- Consider a code with t=1 and 3 coded bits.
- A burst error of length 3 can not be corrected.

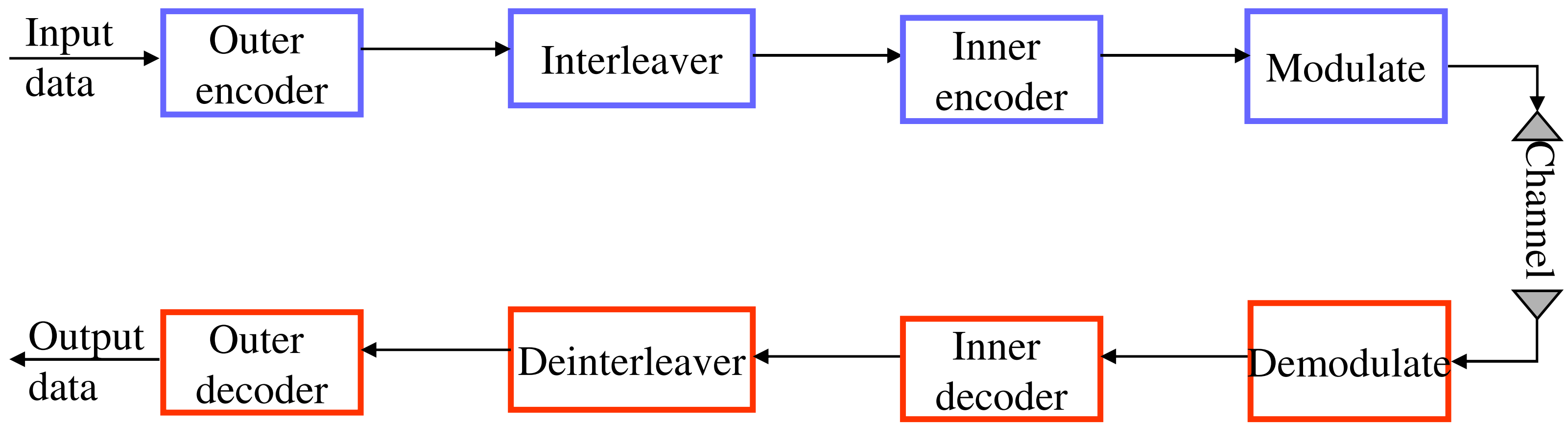


- Let us use a block interleaver 3X3



Concatenated codes

- A concatenated code uses two levels on coding, an inner code and an outer code (higher rate).
  - Popular concatenated codes: Convolutional codes with Viterbi decoding as the inner code and Reed-Solomon codes as the outer code
- The purpose is to reduce the overall complexity, yet achieving the required error performance.



Optimum decoding

- If the input sequence messages are equally likely, the optimum decoder which minimizes the probability of error is the *Maximum likelihood* decoder.

- ML decoder, selects a codeword among all the possible codewords which maximizes the likelihood function $p(\mathbf{Z} | \mathbf{U}^{(m')})$ where $\mathbf{Z}$ is the received sequence and $\mathbf{U}^{(m')}$ is one of the possible codewords:

2^L codewords to search!!!

> ➢ML decoding rule:
>
> $$\text{Choose } \mathbf{U}^{(m')} \text{ if } p(\mathbf{Z} | \mathbf{U}^{(m')}) = \max_{\text{over all } \mathbf{U}^{(m)}} p(\mathbf{Z} | \mathbf{U}^{(m)})$$

The Viterbi algorithm

- The Viterbi algorithm performs Maximum likelihood decoding.
- It find a path through trellis with the largest metric (maximum correlation or minimum distance).
  - It processes the demodulator outputs in an iterative manner.
  - At each step in the trellis, it compares the metric of all paths entering each state, and keeps only the path with the largest metric, called the survivor, together with its metric.
  - It proceeds in the trellis by eliminating the least likely paths.
- It reduces the decoding complexity to $L2^{K-1}$!

The Viterbi algorithm - cont'd

- Viterbi algorithm:

A. Do the following set up:

- For a data block of L bits, form the trellis. The trellis has $L+K-1$ sections or levels and starts at time t_1 and ends up at time t_{L+K} .
- Label all the branches in the trellis with their corresponding branch metric.
- For each state in the trellis at the time t_i which is denoted by $S(t_i) \in \{0,1,...,2^{K-1}\}$, define a parameter $\Gamma\left(S(t_i), t_i\right)$

B. Then, do the following:

The Viterbi algorithm - cont'd

1. Set $\Gamma(0,t_1)=0$ and $i=2.$
2. At time t_i , compute the partial path metrics for all the paths entering each state.
3. Set $\Gamma\left(S(t_i),t_i\right)$ equal to the best partial path metric entering each state at time t_i.

   Keep the survivor path and delete the dead paths from the trellis.
4. If $i<L+K$, increase i by 1 and return to step 2.

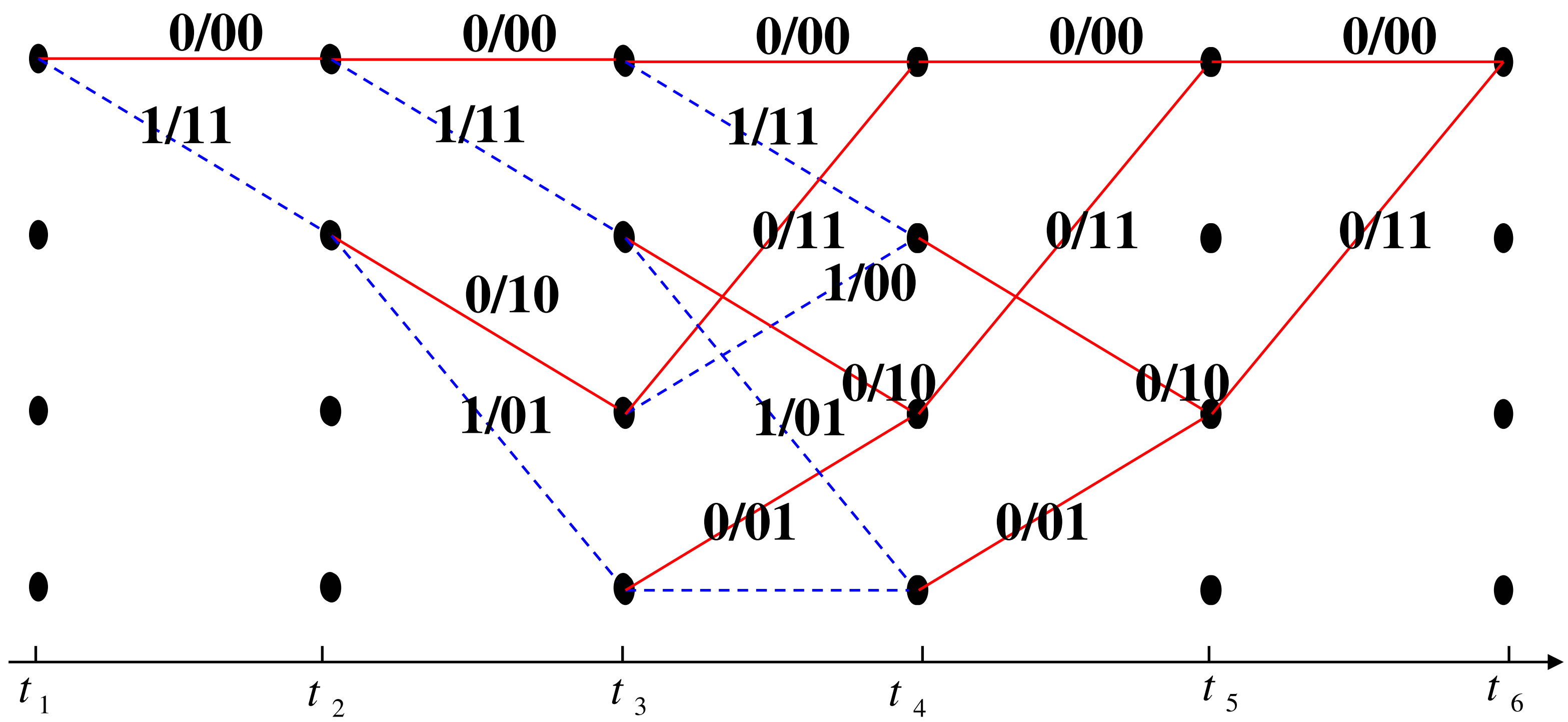
C. Start at state zero at time t_{L+K} . Follow the surviving branches backwards through the trellis. The path thus defined is unique and correspond to the ML codeword.

Example of Hard decision Viterbi decoding

$\mathbf{m} = (101)$

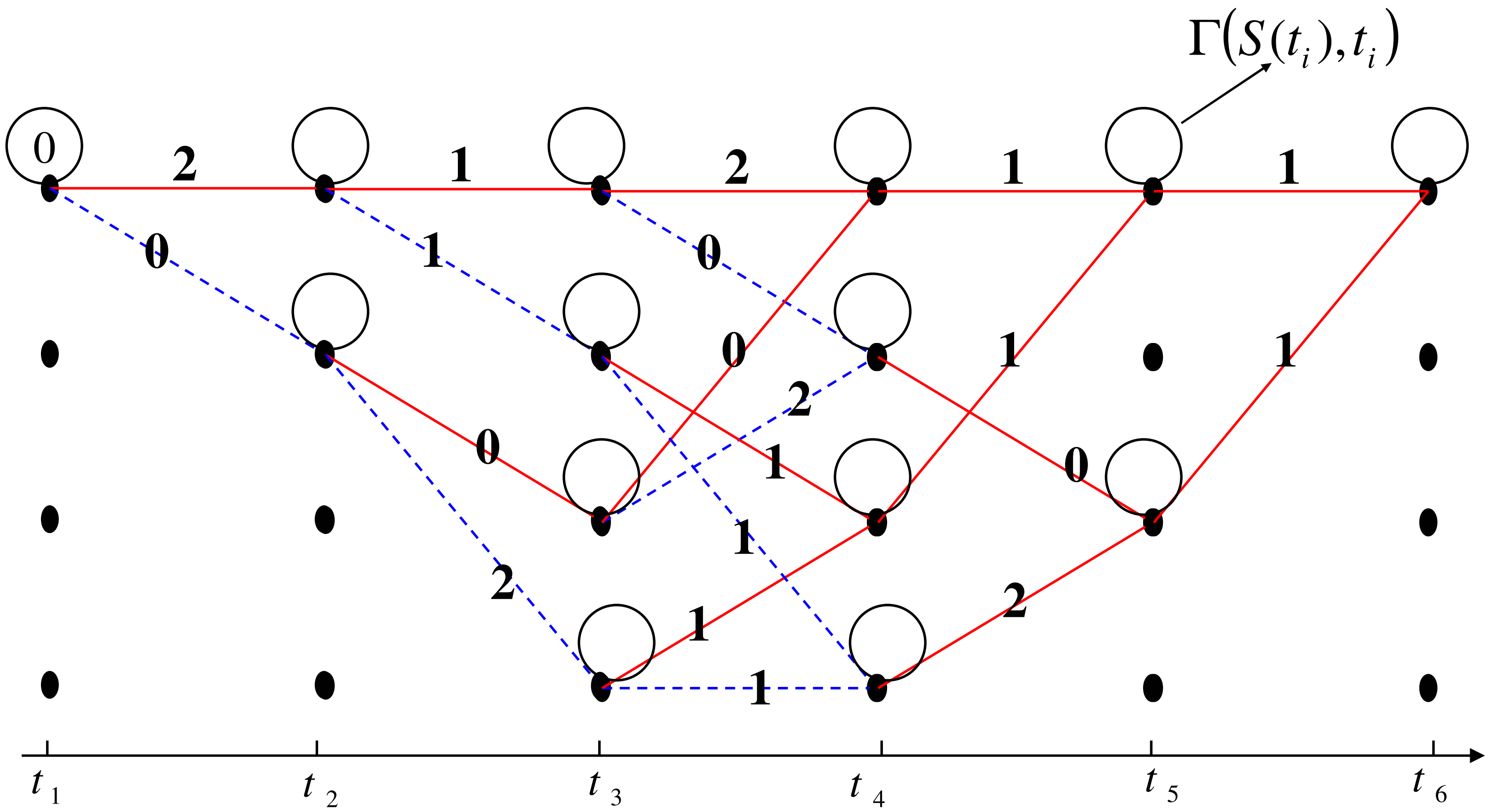
$\mathbf{U} = (11 \quad 10 \quad 00 \quad 10 \quad 11)$

$\mathbf{Z} = (11 \quad 10 \quad 11 \quad 10 \quad 01)$



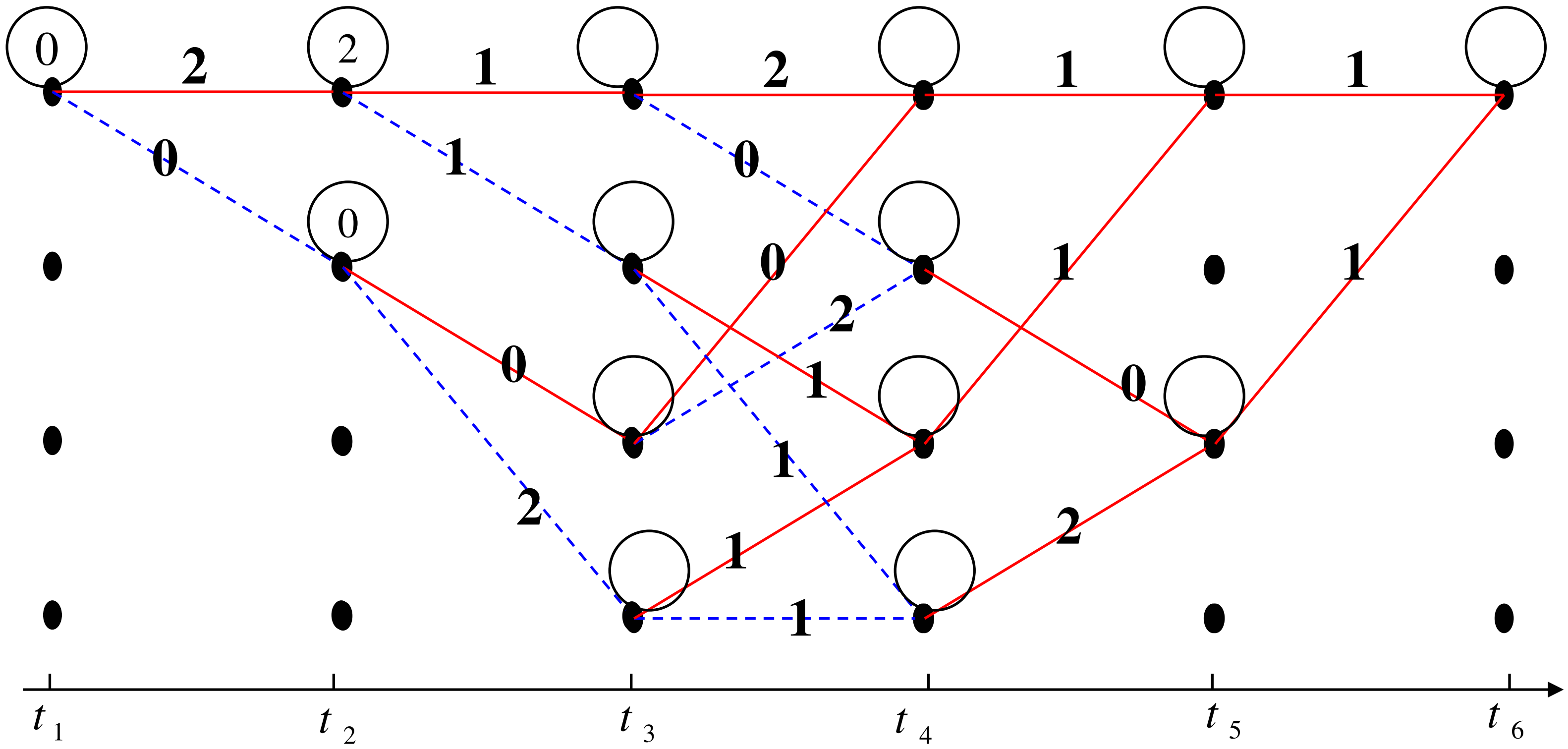
Example of Hard decision Viterbi decoding-cont'd

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



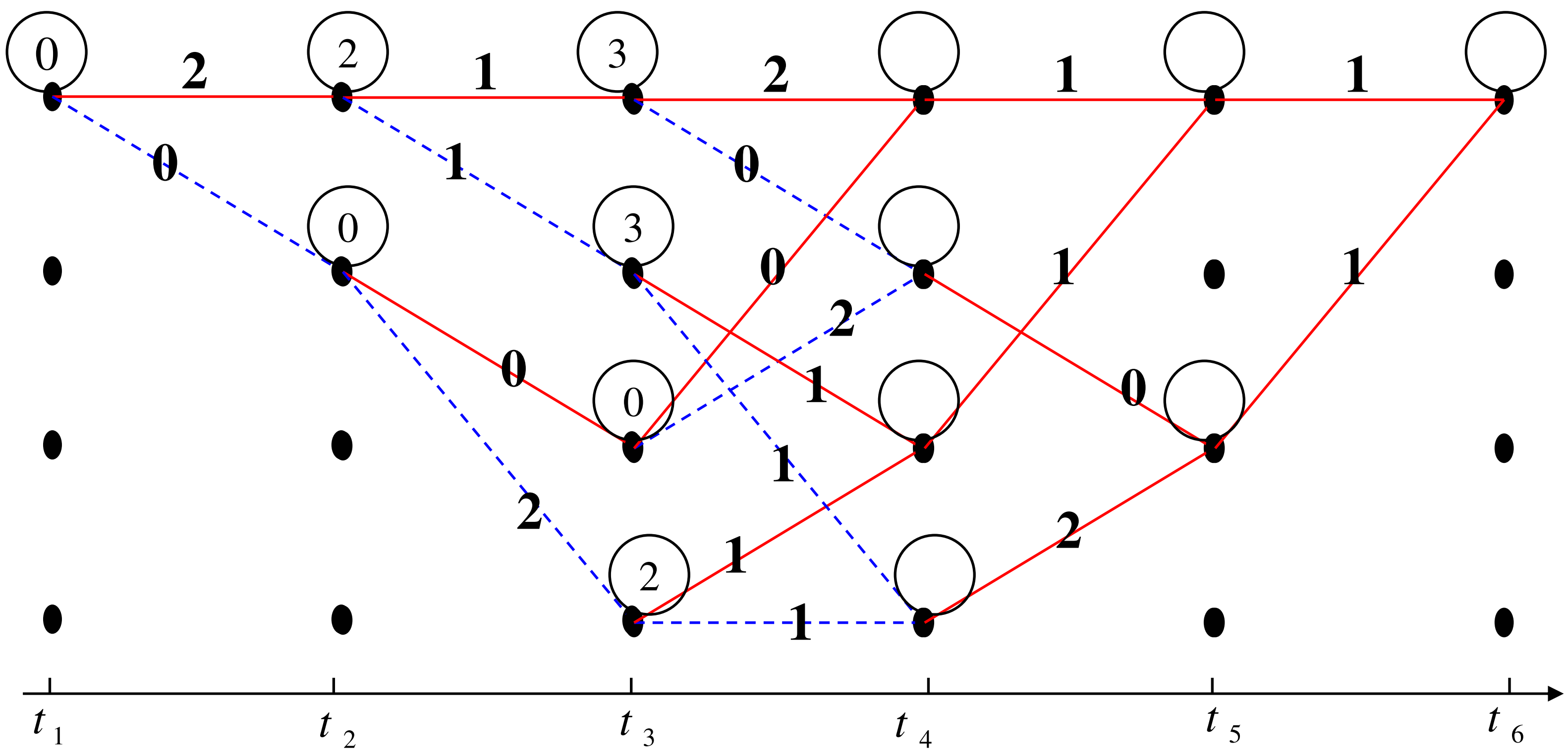
Example of Hard decision Viterbi decoding-cont'd

- i=2



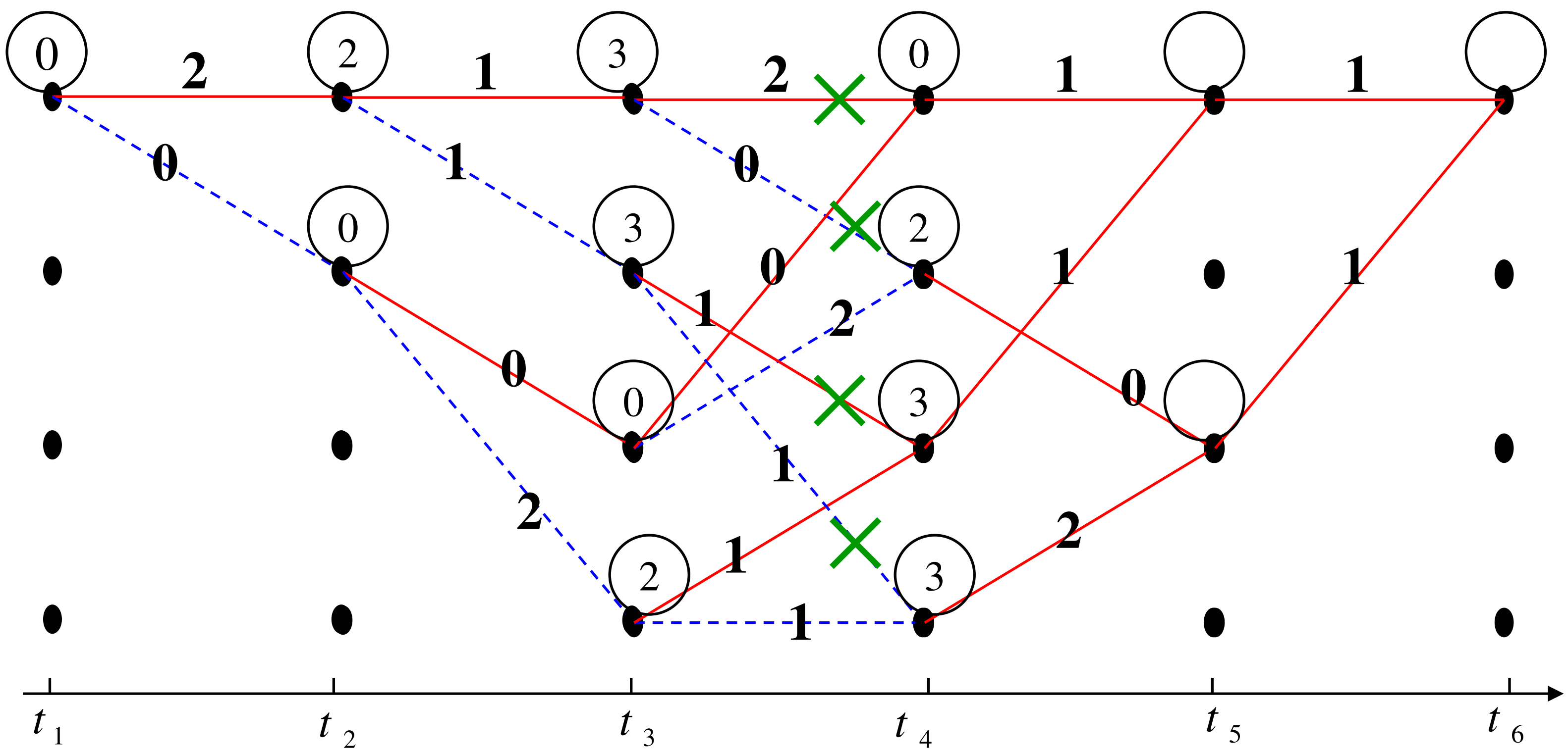
Example of Hard decision Viterbi decoding-cont'd

- i=3



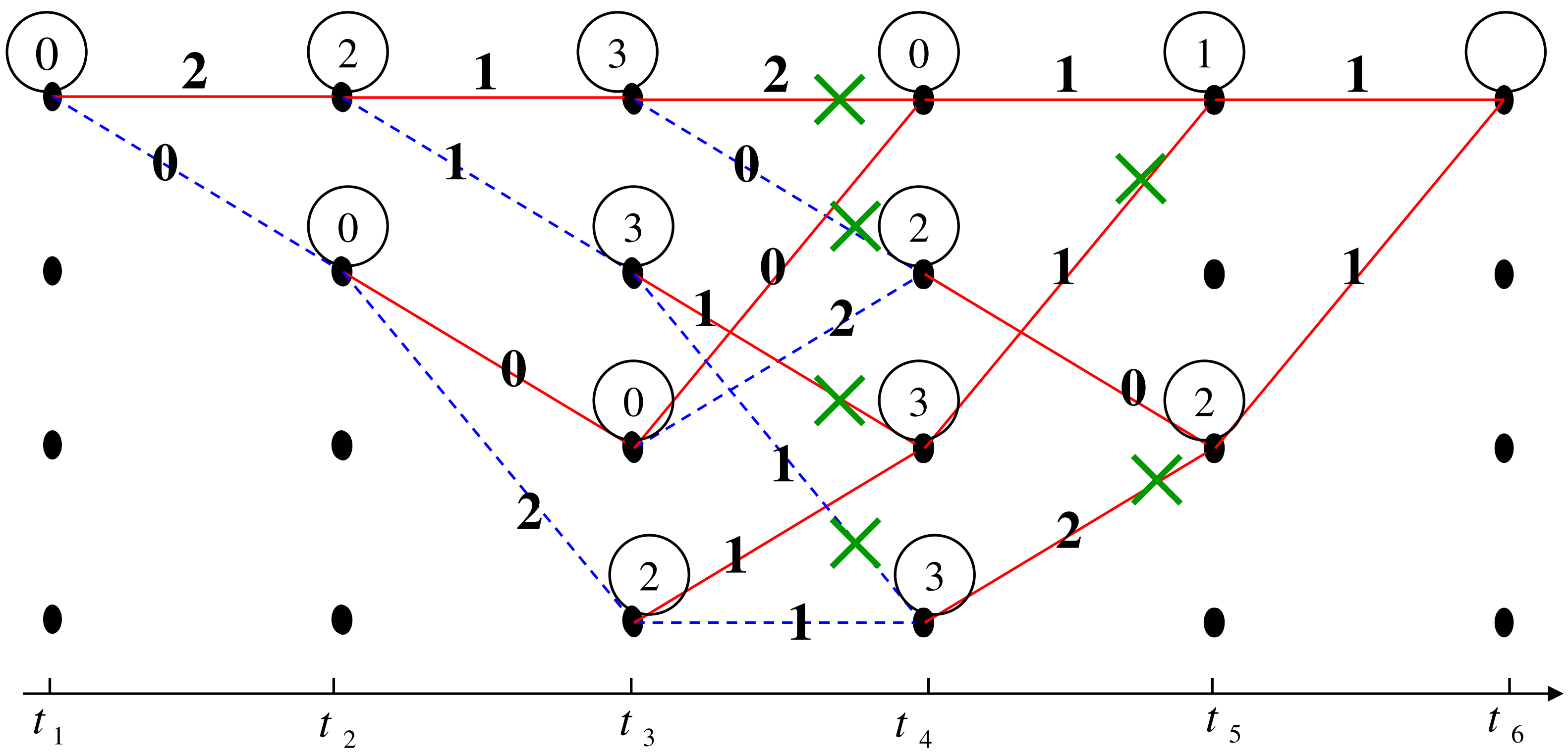
Example of Hard decision Viterbi decoding-cont'd

- i=4



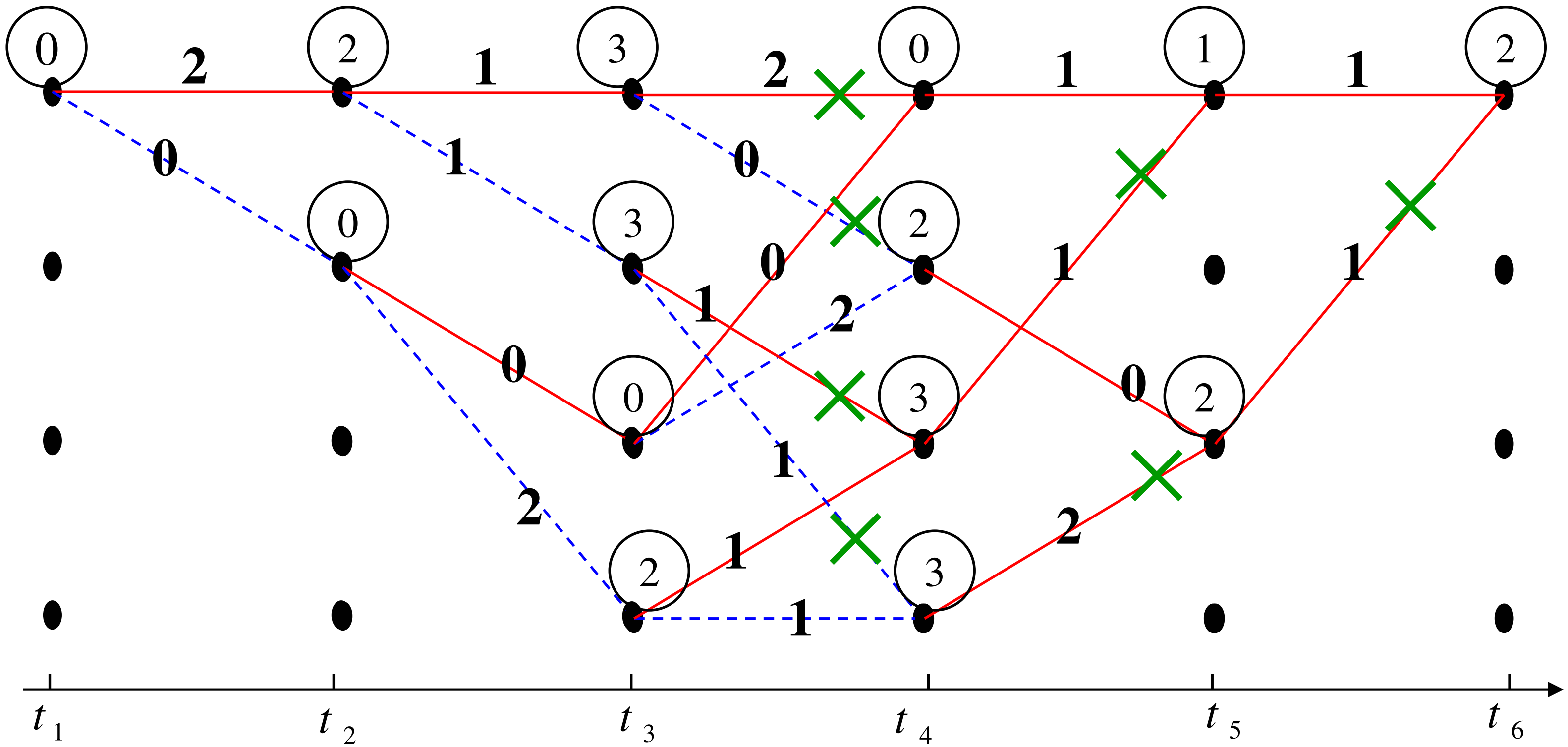
Example of Hard decision Viterbi decoding-cont'd

- i=5



Example of Hard decision Viterbi decoding-cont'd

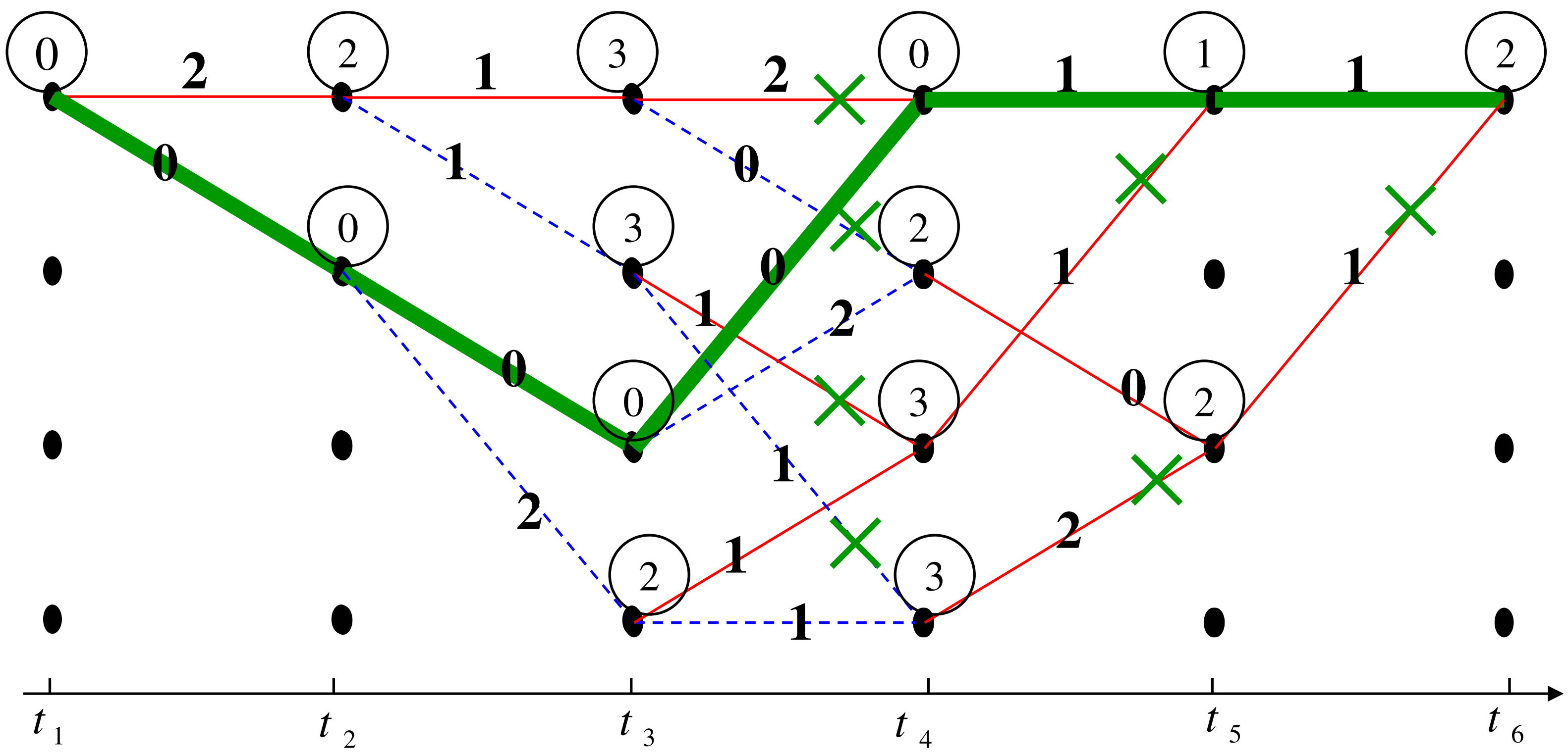
- i=6



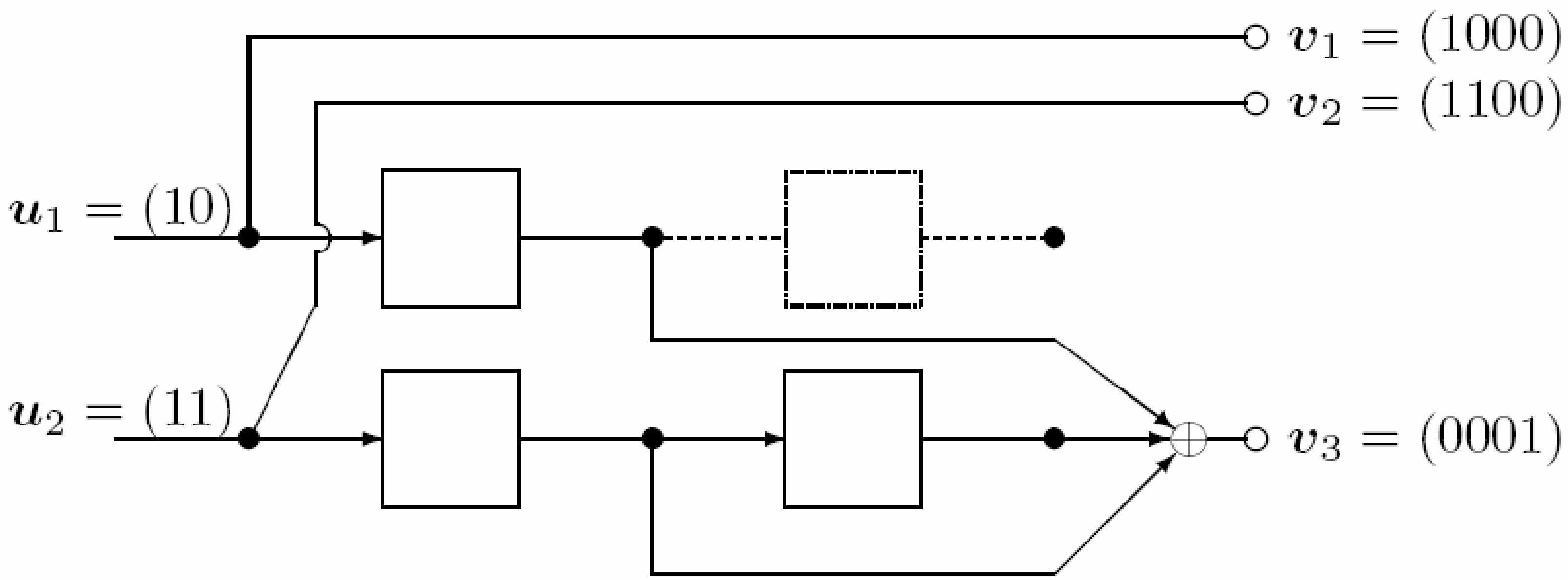
Example of Hard decision Viterbi decoding-cont'd

- Trace back and then:

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

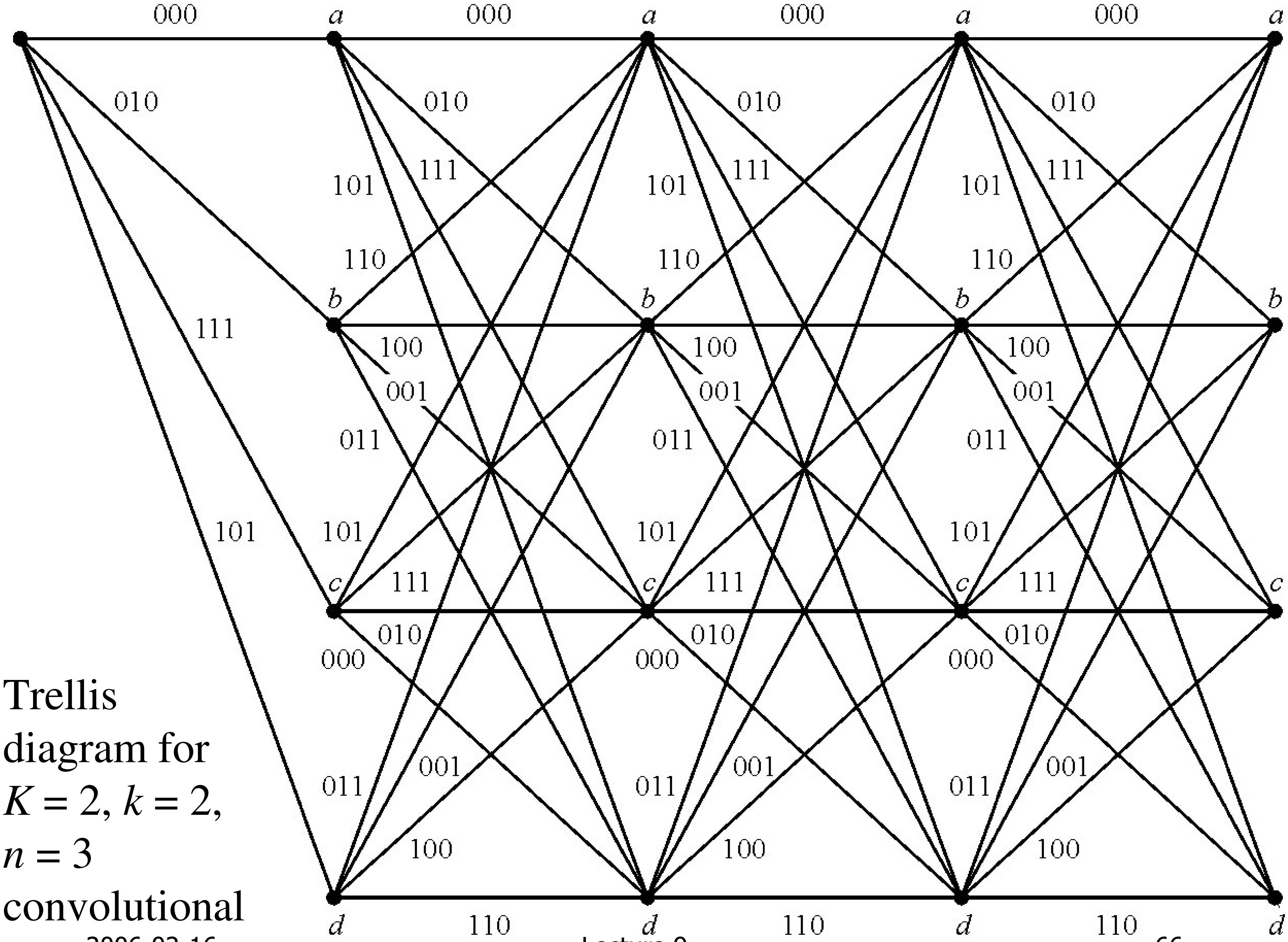


Encoder for the Binary $(3, 2, 2)$ Convolutional Code

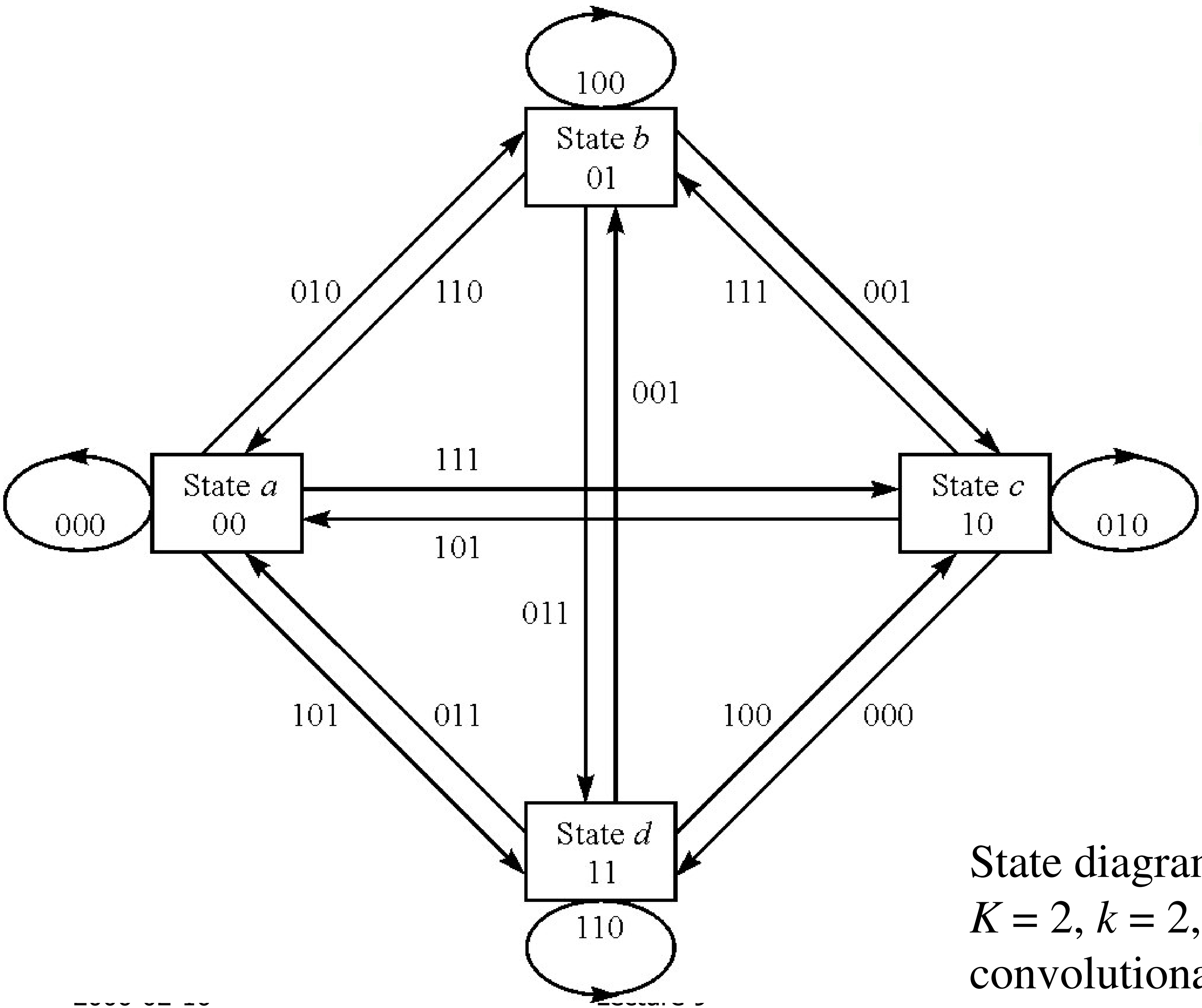


$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010\ 000\ 001)$



Trellis diagram for $K = 2$, $k = 2$, $n = 3$ convolutional code.



State diagram for $K = 2$, $k = 2$, $n = 3$ convolutional code.