Lecture 9

- Video Transcoding
- Channel Coding

Video Transcoding

- Video transcoding refers to the conversion of a compressed video stream into another compressed video stream and may provide the following functionalities:
  - Format conversion
  - Bit rate reduction
  - Spatial resolution reduction
  - Temporal resolution reduction
  - Error resilience

Video Transcoding (T to H)

- Video transcoding is applied in a digital video broadcast (DVB) scenario, where a source video stream from a DVB-T signal is transcoded to a target video stream suitable for DVB-H enabled devices, such as mobile phones.
- Since the transcoding has to be performed in real-time on a mobile device with limited memory and computational power, an appropriate trade-off between computational complexity and visual quality has to be achieved

Transcoding Architectures

- Existing transcoding methods can be roughly grouped into 3 different categories

  – Cascaded pixel domain transcoder
  – Open loop transcoder
  – Closed loop transcoder

Cascaded pixel domain transcoder

- Most straightforward approach is to cascade the decoder and encoder directly.
- The source video stream is fully decoded and then encoded to the target video stream considering the desired format, bit rate, frame rate and spatial resolution.
- Due to the encoding of the video stream, pixel domain transcoding is the most complex approach but also leads to the highest possible quality.

Open loop transcoder

- Least complex transcoding architecture.
- It translates the source bit-stream directly into the target bit-stream by applying:
  - variable length decoding to obtain the macroblock information,
  - Re-quantising the DCT coefficients to meet the target bitrate,
  - remapping the motion vectors and macroblock types,
  - Variable length encoding.
- Since this may introduce a mismatch between the prediction before and after the transcoding, open loop transcoding approaches are subject to error accumulation which is commonly referred to as drift error.

Closed loop transcoder

- Aims at eliminating the mismatch between the predictions by approximating the cascaded pixel domain transcoder.
- This is achieved through a single feedback loop which compensates the error introduced by the initial prediction.
- Closed loop architectures provide a good trade-off

Channel Coding
(Forward Error Correction)

Channel Coding

- Channel coding: signal processing according to the properties of transmission channel (medium): terrestrial, ground-to-satellite, fibre
- Two major types:
  - Convolutional coding:
    - binary coding, i.e., operating on 1 bit at a time
    - only requires error location + inversion of wrong bit
  - Block coding:
    - symbol coding, i.e., operating on blocks of tens or hundreds of bits at a time
    - more powerful, but requires error location + estimation of correct symbol)

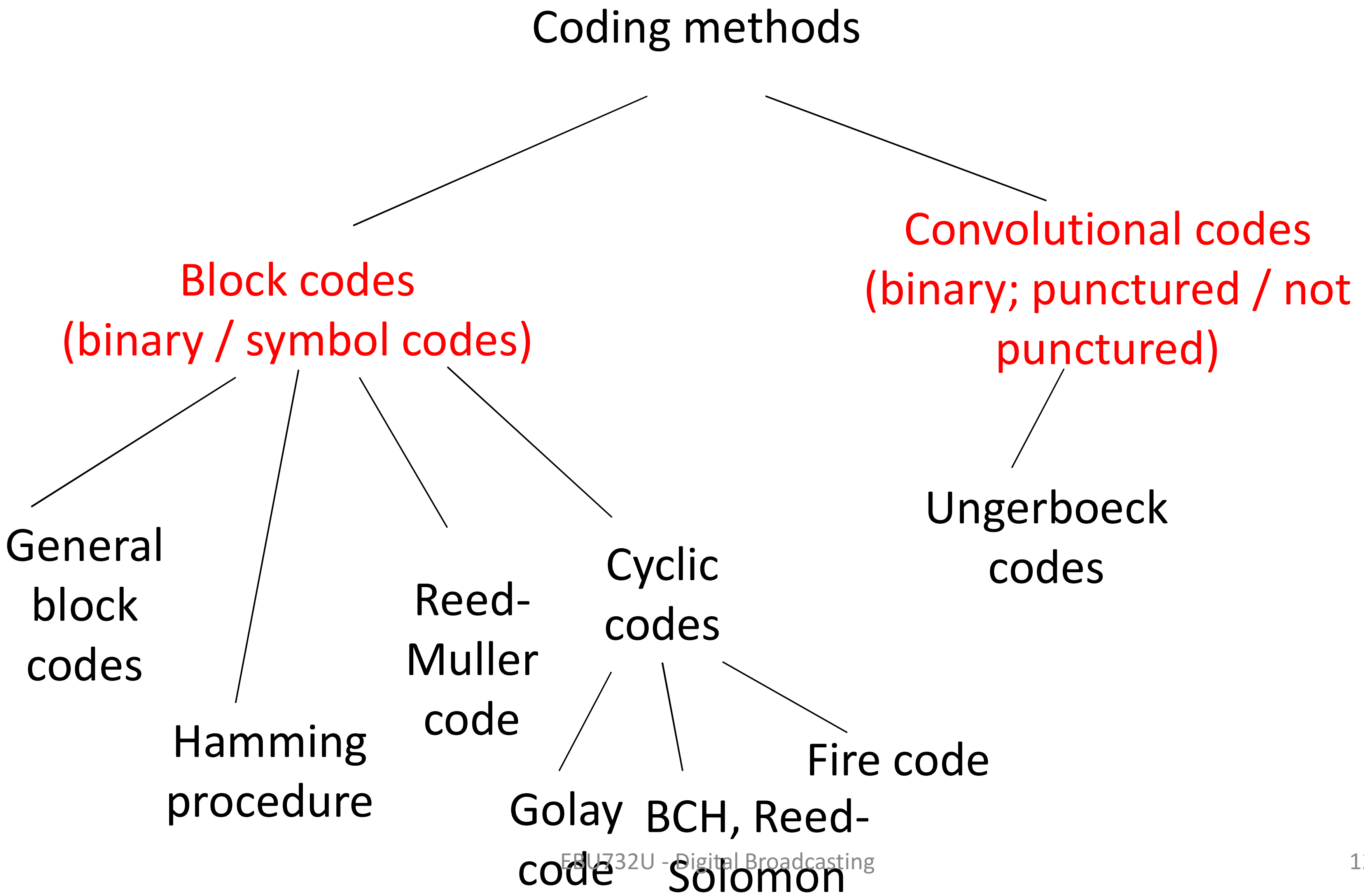
Coding: What and Why?

- Loss of information, especially due to additive white Gaussian noise (AWGN)
- Purpose of Forward Error Correction (FEC): to improve the capacity of a channel by designing and exploiting redundant information to the data that are transmitted through the channel.
  - Channel coding is the process of designing and adding this FEC information
  - Adding redundancy, i.e., no new payload info
  - Based on error correction schemes (ARQ, etc.)

Advances in Coding

- ARQ (Automatic Repeat request)
- Linear block codes
- Hamming codes
- **Convolutional coding**
- Block based channel coding
- Turbo coding
- Coded modulation
- Space-time block and trellis coding
- Turbo equalisation
- Low Density Parity Check (LDPC) coding

Code Classification



Block Codes vs. Convolutional Codes

- *Block codes*:
  - input bit stream divided in blocks of fixed length *m* (number of symbols); *m* correction symbols added on

- Achieved rate = n/m bits

Block Codes vs. Convolutional Codes

- *Convolutional codes*:
  - each bit is considered one by one
  - shift register stores *S* previous bits for processing

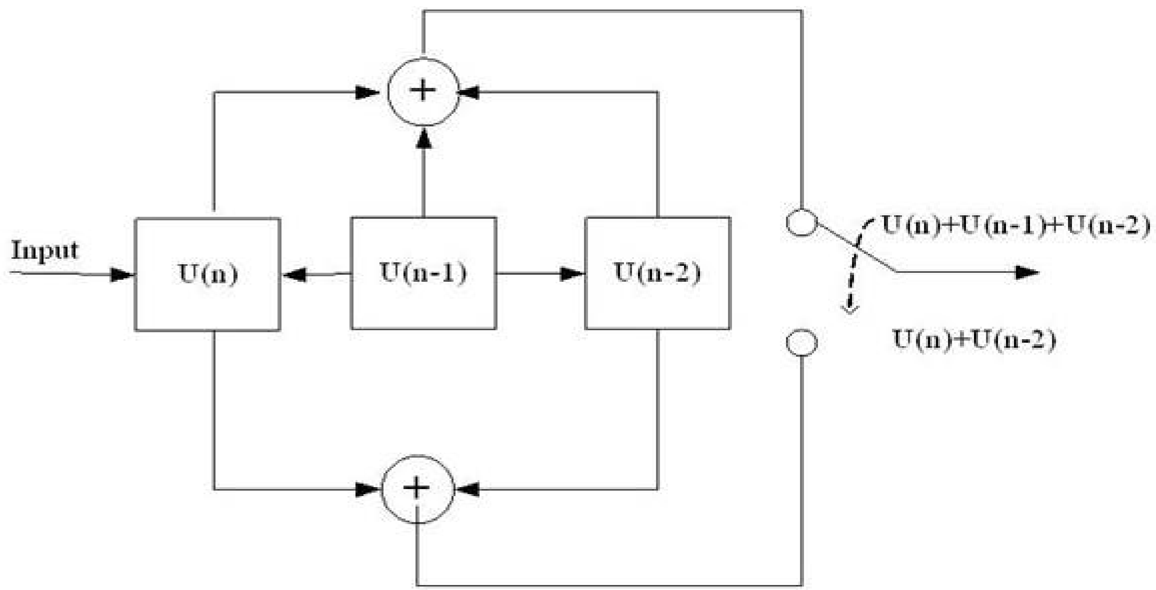


Figure 1: Shift Register in Convolutional code

Overview

- ARQ (telecom systems)
- Parity (digital circuit design)
- Linear block codes (telecom systems)
- Convolutional codes
- Viterbi decoding

Convolutional Coding (CC)

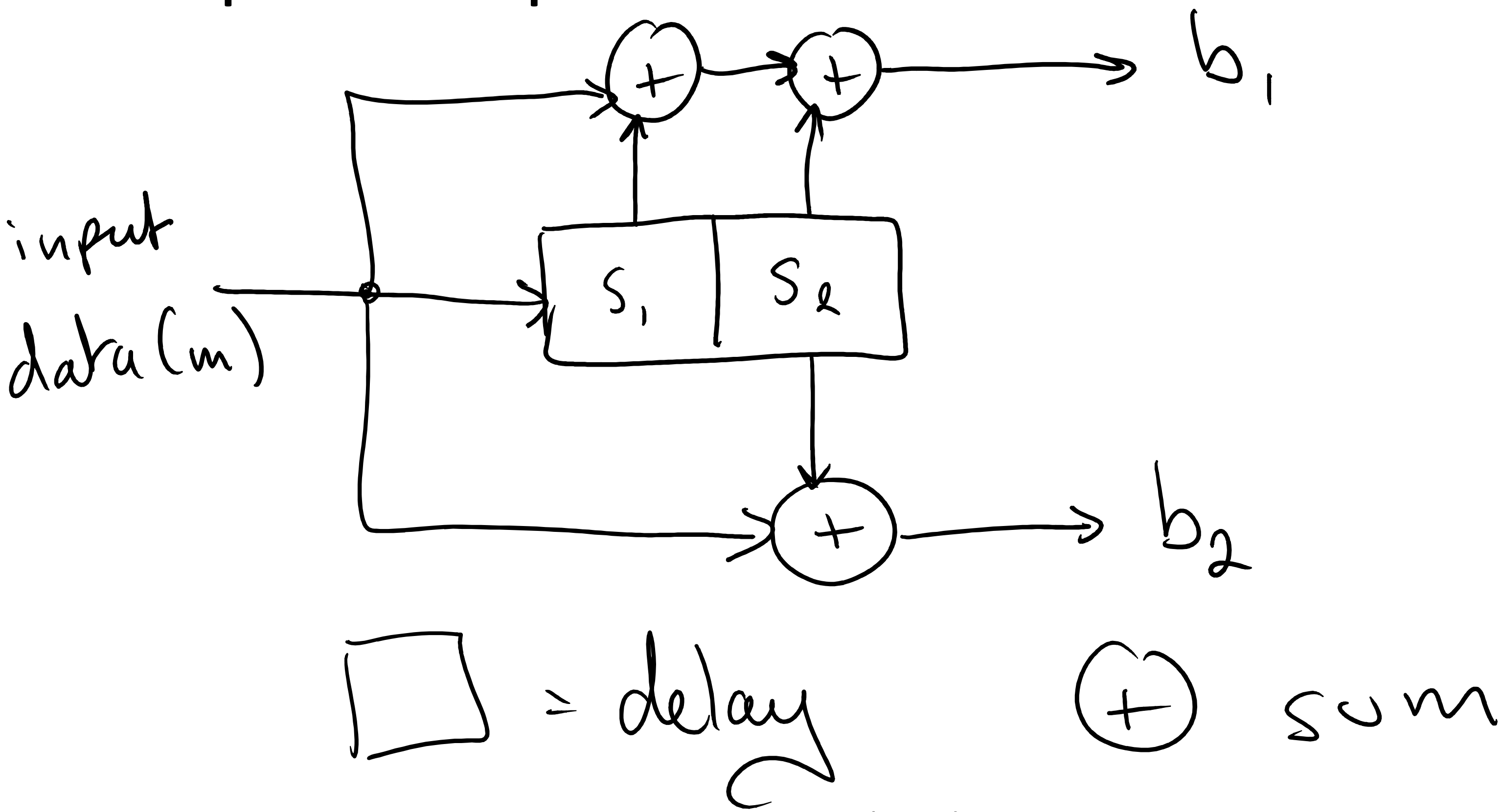
- Generate extra bits to protect data ($n > m$)
- Most robust way: generate one correction bit for every new data bit
  - this is also most expensive way, in terms of extra bits and lowering of throughput
- Has encoder and decoder

Representations

- Shift register
- Polynomial expansion – generator matrix
- State table & state diagram
- Trellis
- Tree
- Parity check matrix/syndrome

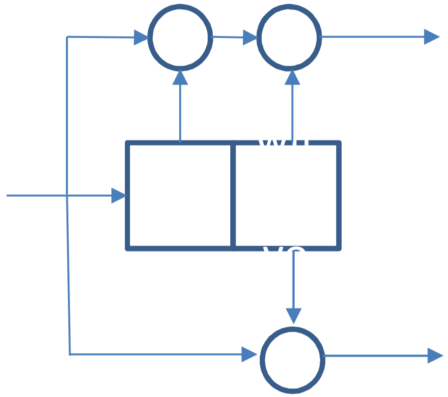
Shift Register Representation

- Many types of convolutional encoders exist
- Simple example:



Some Definitions

- Two main CC parameters :
  - coding rate *R*
  - constraint length *K*
  - E.g. "rate-½, (constraint) length-3 convolutional encoder"



m = number of input lines (in example: $m = 1$)

S = length of shift register (memory depth) = 2

$\Rightarrow$ memory size: $S \cdot m = 2$

K = number of inputs contributing to output of encoder

= constraint length = $(S+1)m = 3$

n = number of output lines = 2

$\Rightarrow$ coding rate: $R = m/n = ½$ (< 1)

NB: bandwidth expansion factor = $1/R$ (> 1)

convolutional coder *CC(n,m,K)* = *CC*(2,1,3) or also *CC*(1/2,3)

Representations

- Shift register
- Polynomial expansion – generator matrix
  (covered in telecoms module)
- State table & state diagram
- Trellis
- Tree
- Parity check matrix/syndrome

Polynomial Representation - Generator Matrix

$$[c]_{1\times n} = [d]_{1\times m}[G]_{m\times n}$$

c = output <u>c</u>ode word vector

d = input <u>d</u>ata word vector

G = <u>g</u>enerator matrix (*m* inputs, *n* outputs)

Representations

- Shift register
- Polynomial expansion – generator matrix
- State table & state diagram
- Trellis
- Tree
- Parity check matrix/syndrome

State Table - Example

Input (b_i)	s_1	s_2	Output (b_1)	Output (b_2)
0	0	0	0	0
1				
1				
1				
0				
1				
1				
0				
0				

State Table - Example

Input (b_i)	s_1	s_2	Output (b_1)	Output (b_2)
0	0	0	0	0
1	0			
1	1			
1	1			
0	1			
1	0			
1	1			
0	1			
0	0			

State Table - Example

Input (b_i)	s_1	s_2	Output (b_1)	Output (b_2)
0	0	0	0	0
1	0	0		
1	1	0		
1	1	1		
0	1	1		
1	0	1		
1	1	0		
0	1	1		
0	0	1		

State Table - Example

Input (b_i)	s_1	s_2	Output $b_1=mod_2(b_i+s_1+s_2)$	Output $b_2=mod_2(b_i+s_2)$
0	0	0	0	0
1	0	0	1	1
1	1	0		
1	1	1		
0	1	1		
1	0	1		
1	1	0		
0	1	1		
0	0	1		

State Table - Example

Input (b_i)	s_1	s_2	Output b_1=mod$_2$(b_i+s_1+s_2)	Output b_2=mod$_2$(b_i+s_2)
0	0	0	0	0
1	0	0	1	1
1	1	0	0	1
1	1	1	1	0
0	1	1	0	1
1	0	1	0	0
1	1	0	0	1
0	1	1	0	1
0	0	1	1	1

State Table - Example

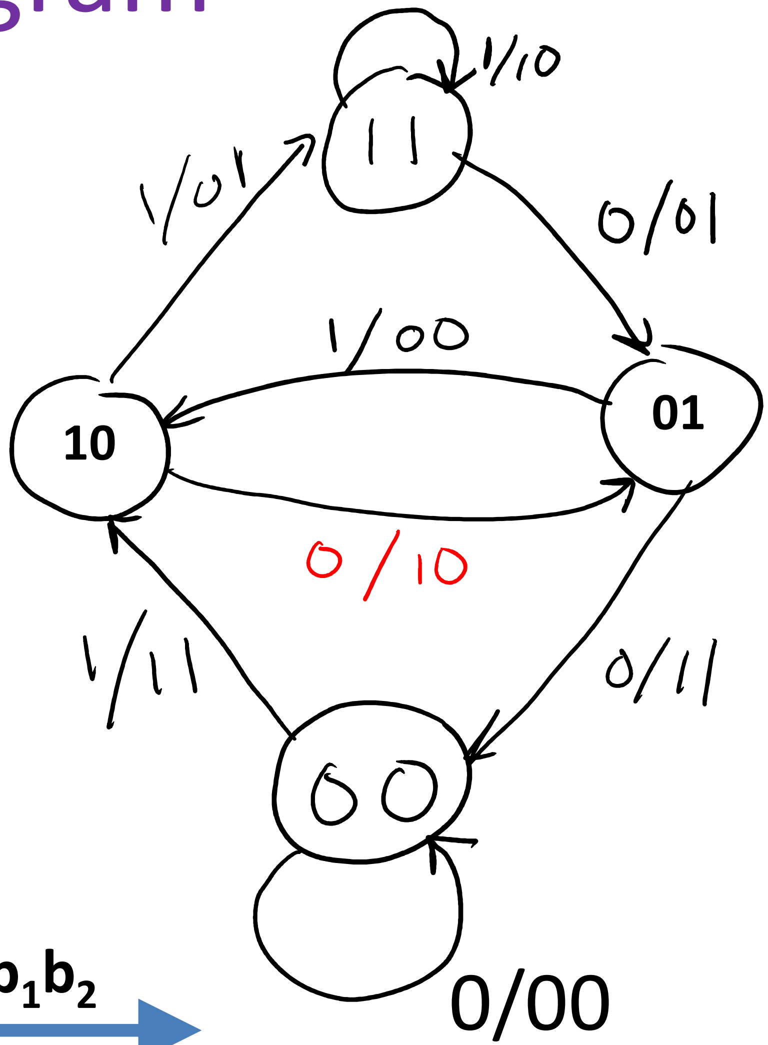
Input (b_i)	s_1	s_2	Output (b_1)	Output (b_2)
0	0	0	0	0
1	0	0	1	1
1	1	0	0	1
1	1	1	1	0
0	1	1	0	1
1	0	1	0	0
1	1	0	0	1
0	1	1	0	1
0	0	1	1	1

- 4 states: 00, 01, 10, 11
- Each state can transition to next state having received either a 1 or 0

Not all possible input/state combinations are included in table!

State Diagram

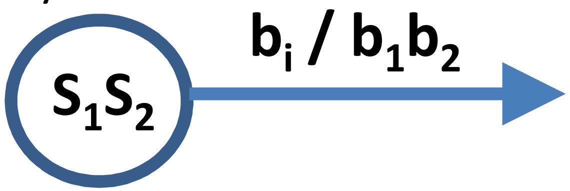
Input (b_i)	s_1	s_2	Output (b_1)	Output (b_2)
0	0	0	0	0
1	0	0	1	1
1	1	0	0	1
1	1	1	1	0
0	1	1	0	1
1	0	1	0	0
0	1	0	1	0
~~0~~	~~1~~	~~1~~	~~0~~	~~1~~
0	0	1	1	1



$$b_1 = \mathrm{mod}_2(b_i+s_1+s_2),$$

$$b_2 = \mathrm{mod}_2(b_i+s_2)$$

Key:



Forward Error Correction

- Starting from any state, there are only two legitimate transitions to a next state
- Finite State Machine (FSM): output depends not only on current input, but also on previous inputs
  - can be regarded as a form of prediction
  - e.g. if the current state is ‘01’ but the received bit stream gives the next two bits as ‘11’, one of these bits must be in error
- This CC(2,1,3) can correct single-bit errors
- Check: for a CC(R=1/2, K=k), each input has an effect on k consecutive pairs of output symbols
  - larger ‘spread’ k yields stronger power of error-correcting

Representations

- Shift register
- Polynomial expansion – generator matrix
- State diagram
- Trellis
- Tree
- Parity check matrix/syndrome

State Diagram Issues

Advantages:

-compact representation for long codes

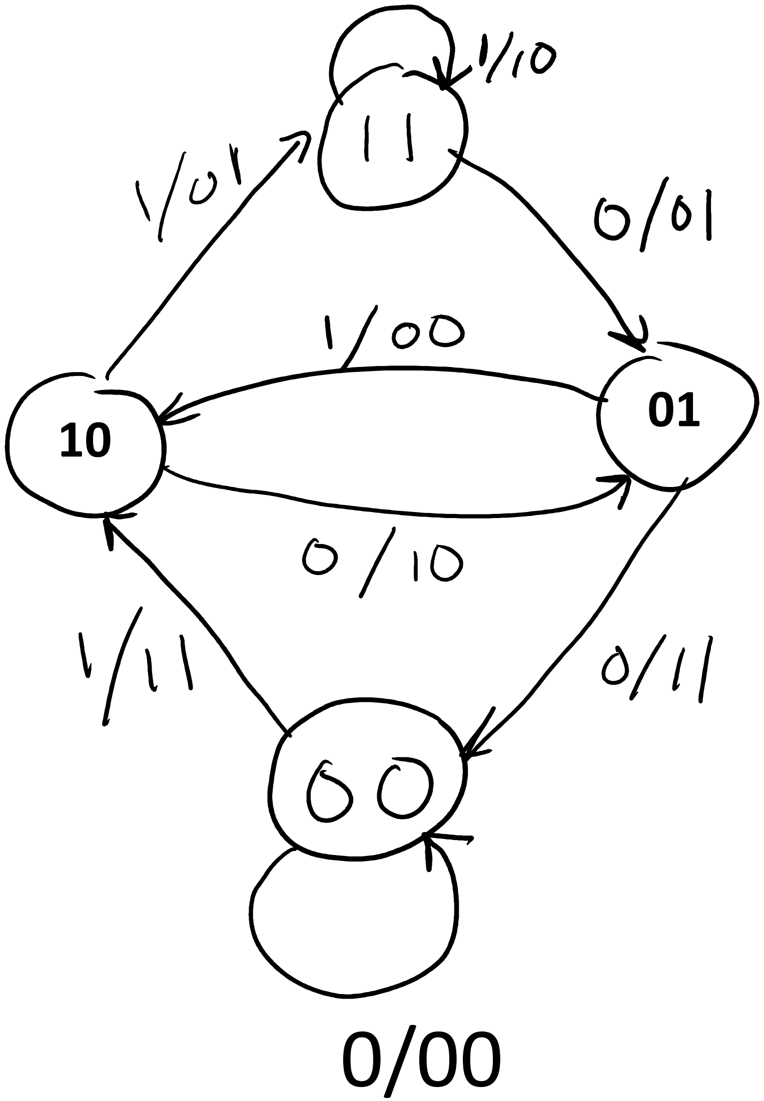
-all possible inputs/state/output combinations

-clear separation of i/s/o

Disadvantage:

-difficult to track states and outputs as a function of time

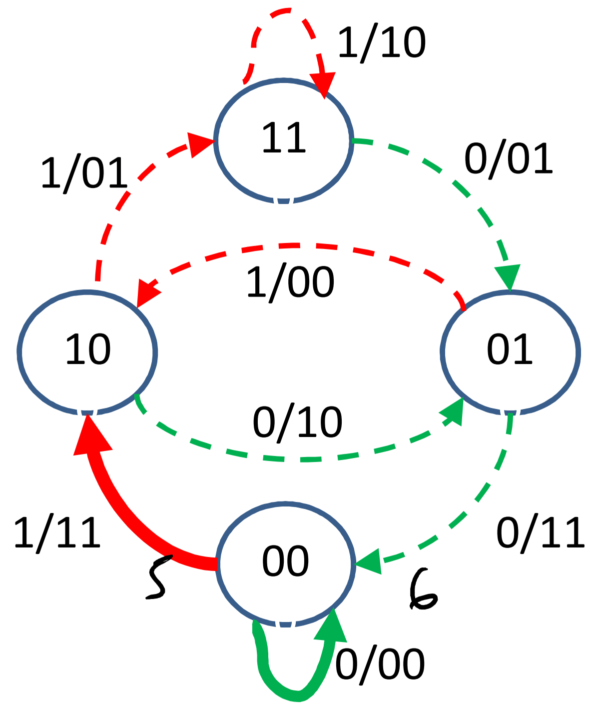
- this becomes an issue when comparing expected and actual outputs



Trellis Diagram

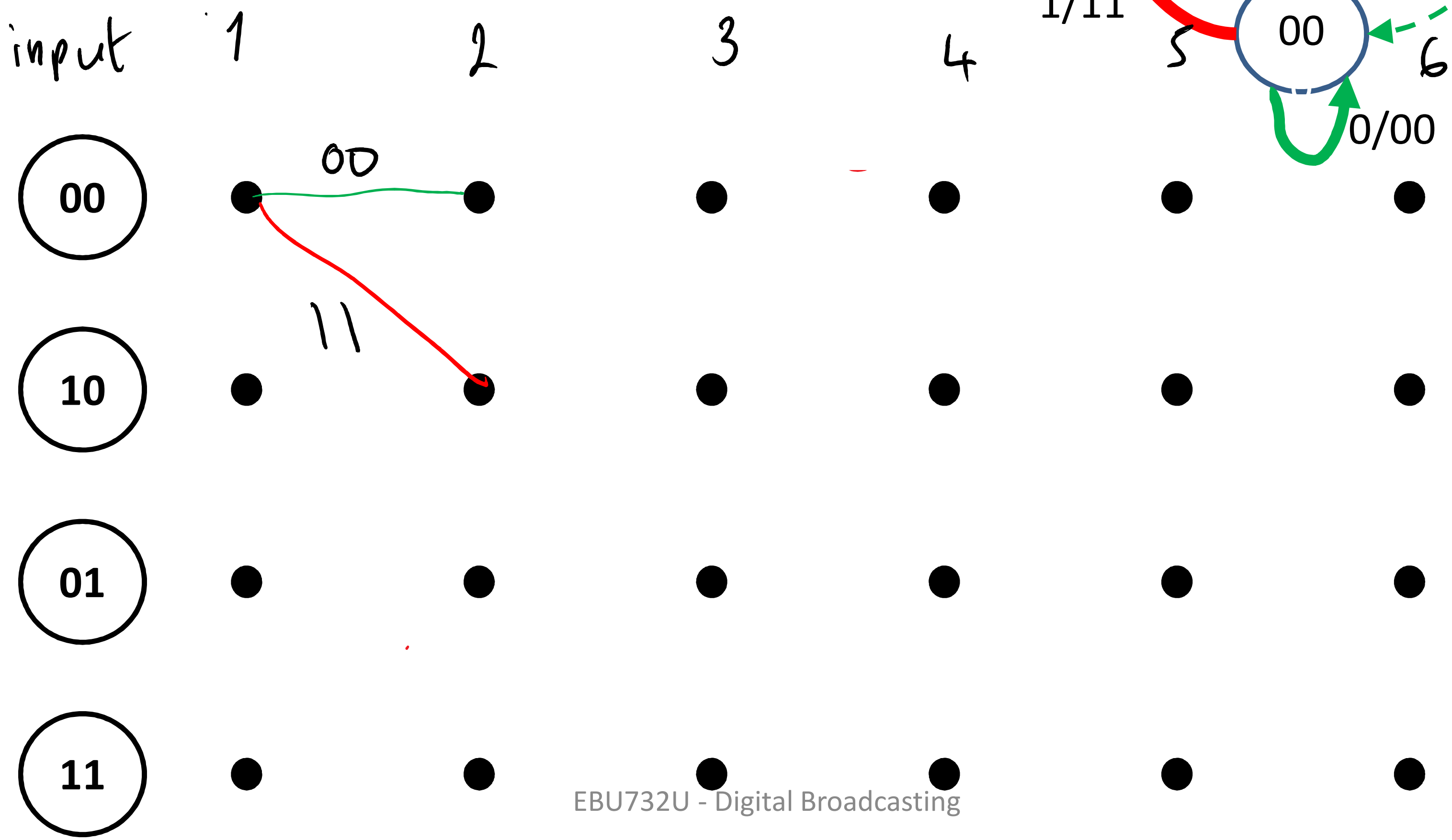
- Trellis diagram is an extension of state diagram which explicitly shows the passage of time
  - All the possible states are shown for each instant of time.
  - Time is indicated by a movement to the right.
  - The input data bits and output code bits are represented by a unique path through the trellis.
  - After the second stage, each node in the trellis has 2^k incoming paths and 2^k outgoing paths.

Trellis Diagram: States



Input symbol = 1

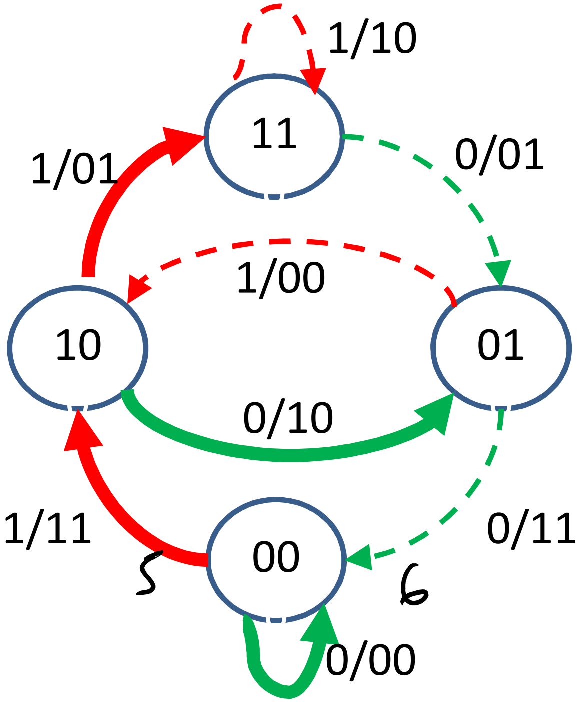
input symbol = 0

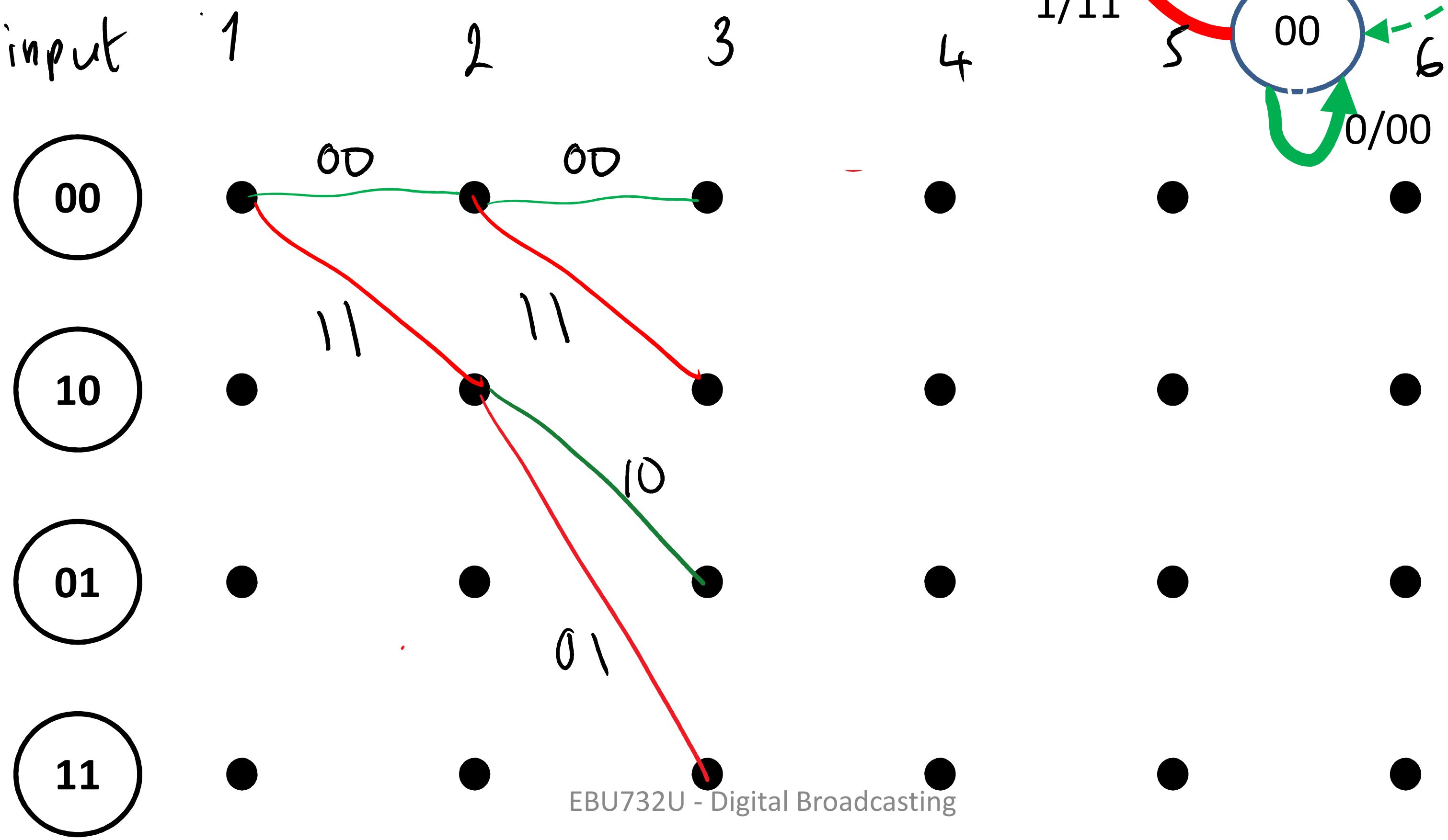


Trellis Diagram: States

Input symbol = 1

input symbol = 0

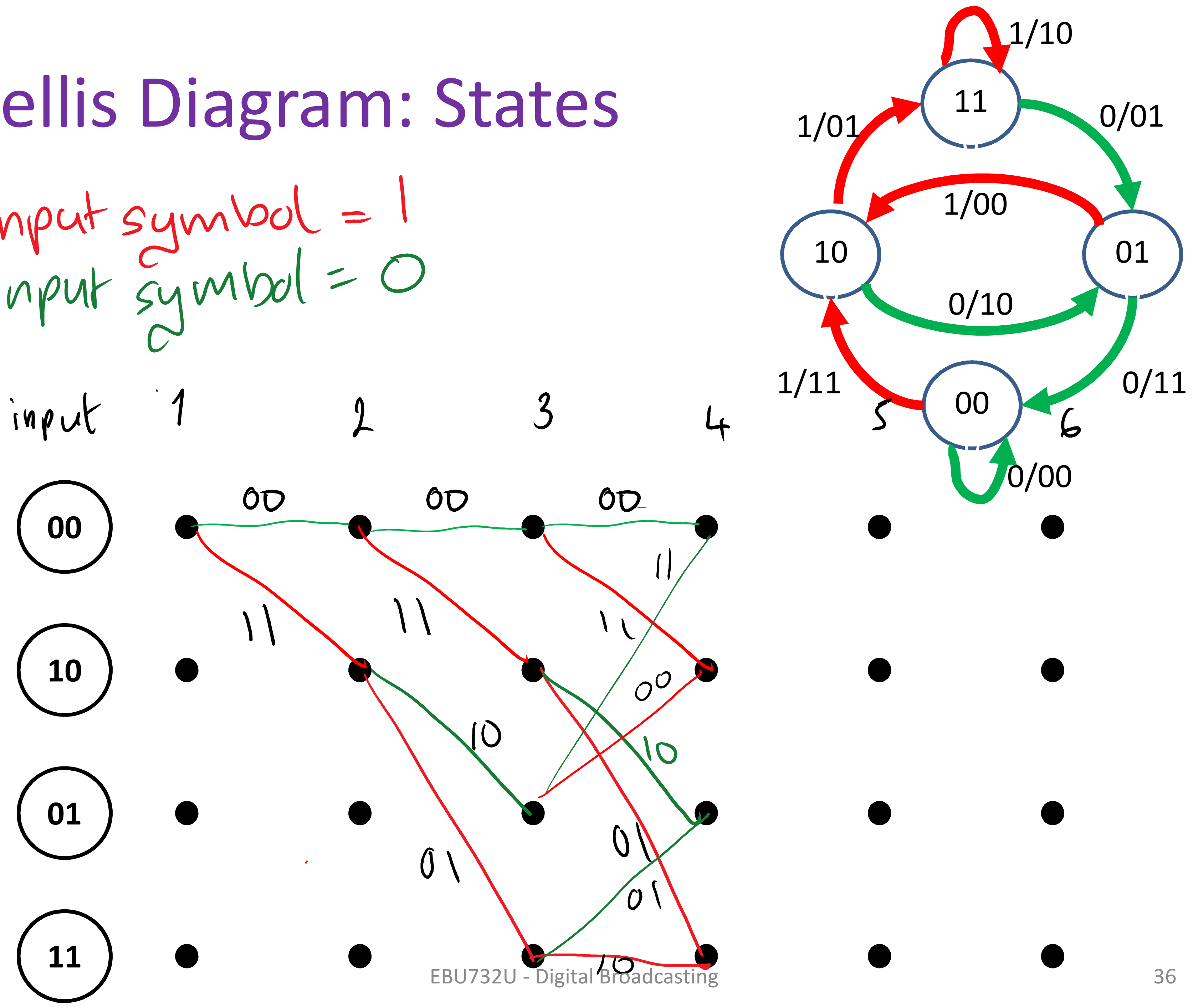




Trellis Diagram: States

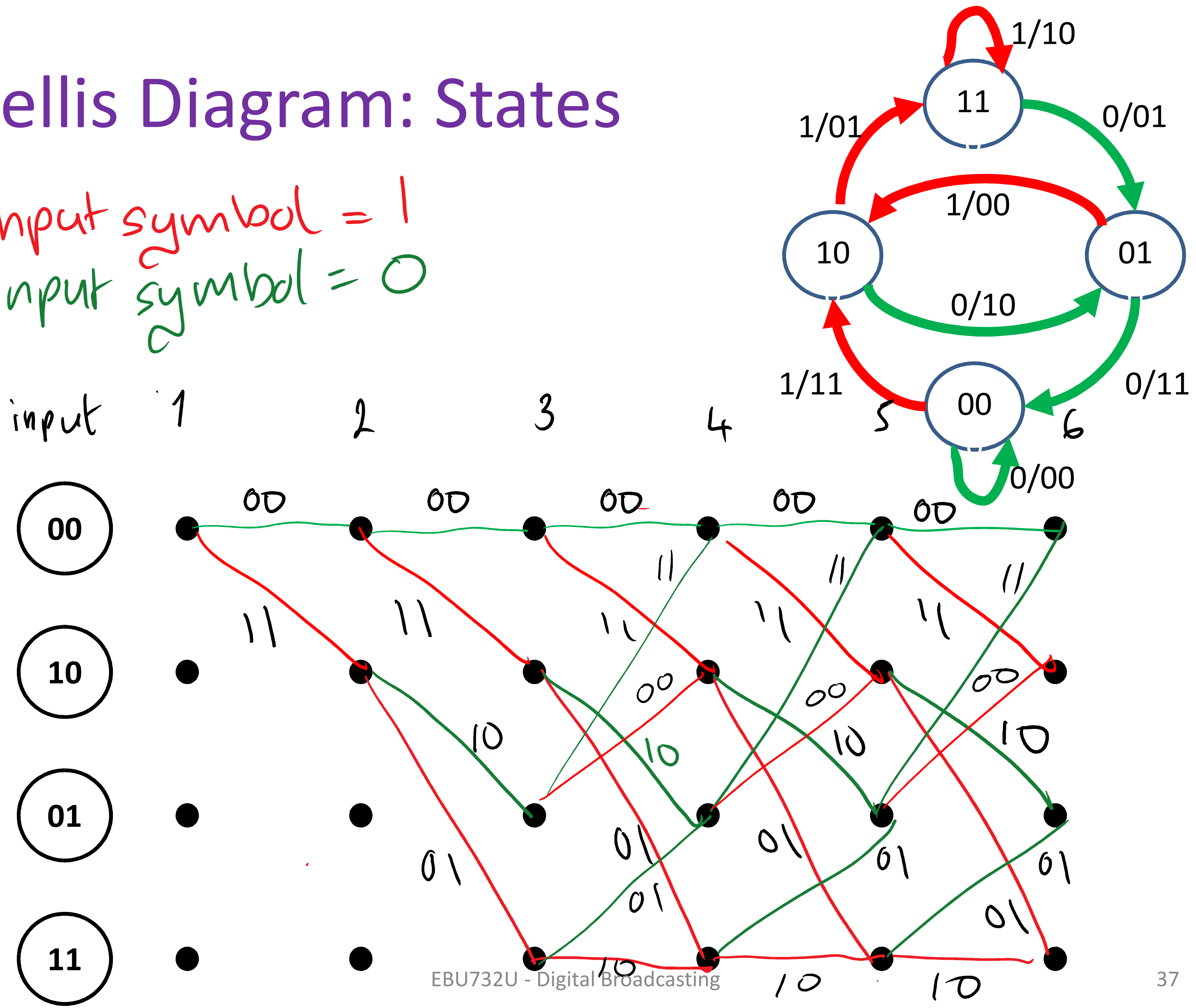
— Input symbol = 1

— input symbol = 0



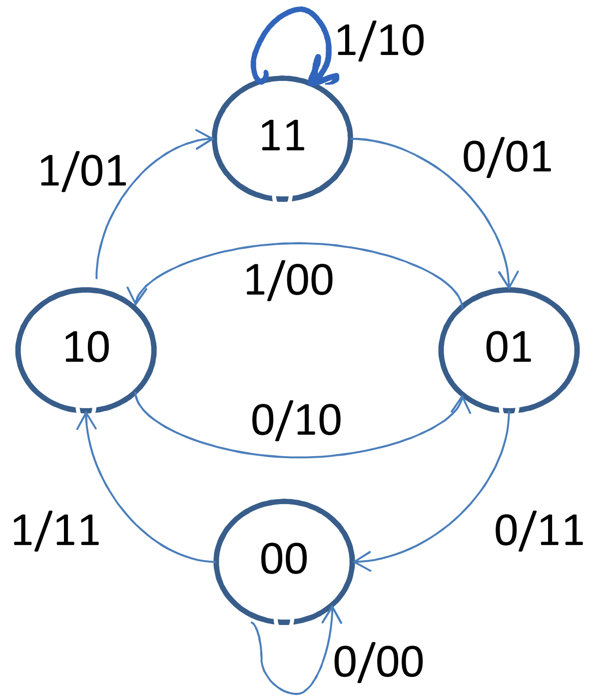
Trellis Diagram: States

— Input symbol = 1
— input symbol = 0

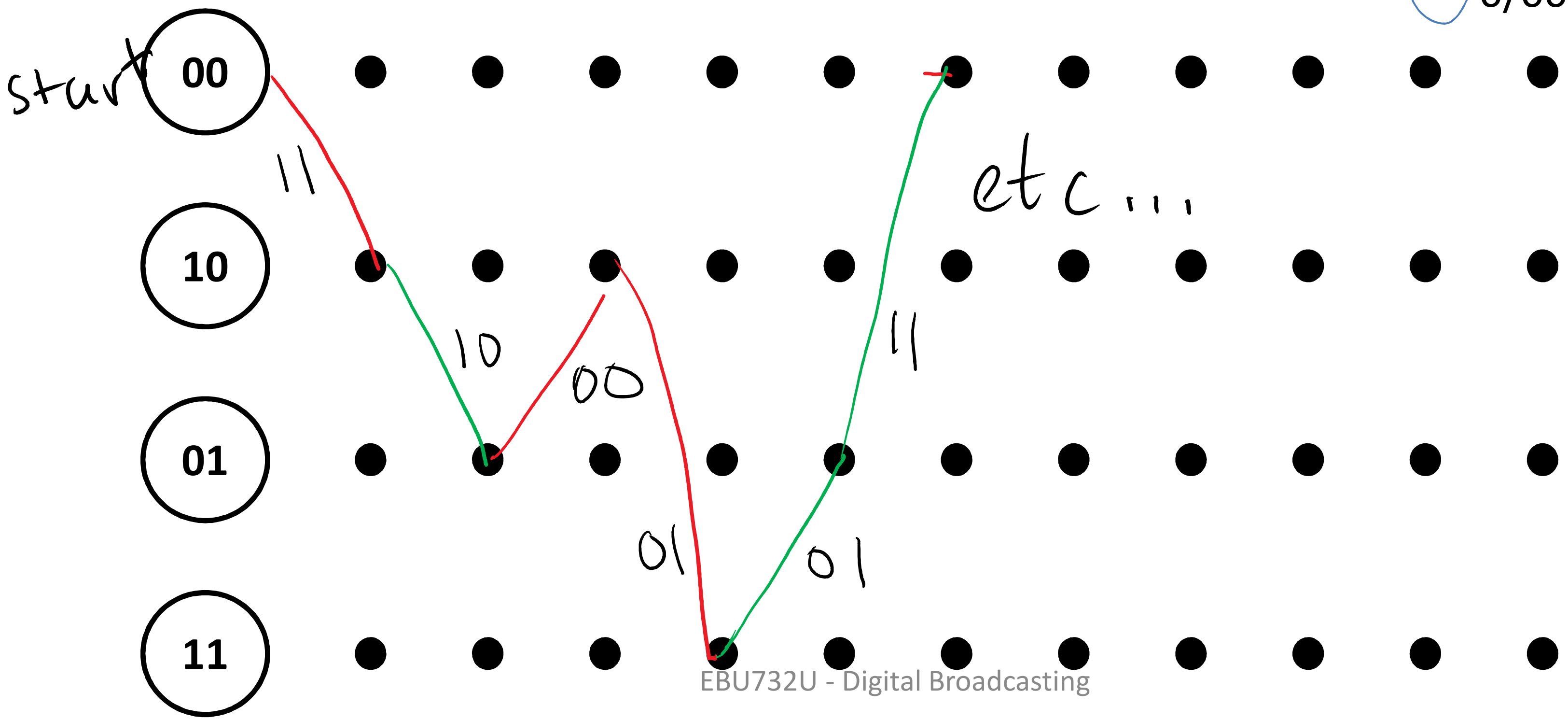


Trellis: Output Path

Based on <u>known</u> <u>input</u> sequence, follow jumps from state to state in the state diagram to complete the trellis path:



e.g. input sequence = 101100 etc.:



Input = 1 0 1 1 0 0 0

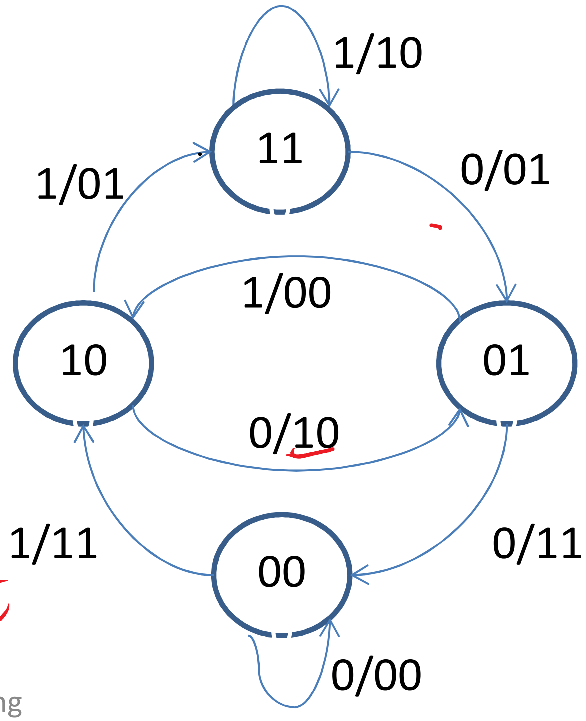
Coded bit sequence = 11 10 00 01 01 11 00

Received = 10 10 10 01 01 11 00
(assumed; given)

error

Always start in state 00

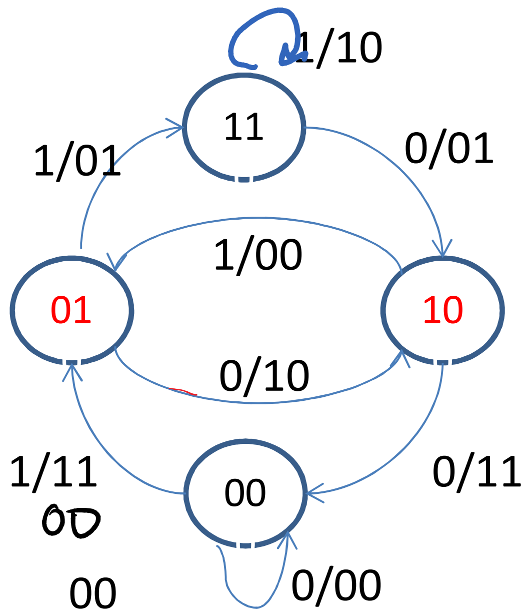
If we know there is an error, we can correct it



Decoding of Convolutional Codes

- Problem statement: input sequence unknown at receiver
- Two main CC decoding techniques:
  - Sequential decoding:
    - +: good performance for high-K codes
    - -: variable decoding time
  - Viterbi decoding:
    - +: fixed decoding time
    - -: computational time grows **exponentially** with K
      - Limited to K=7 or 9 in practice
    - e.g. for R=1/2: signal-to-noise ratio E_b/N can be reduced by 5 dB when using convolutional coding + Viterbi decoding

Viterbi Decoding

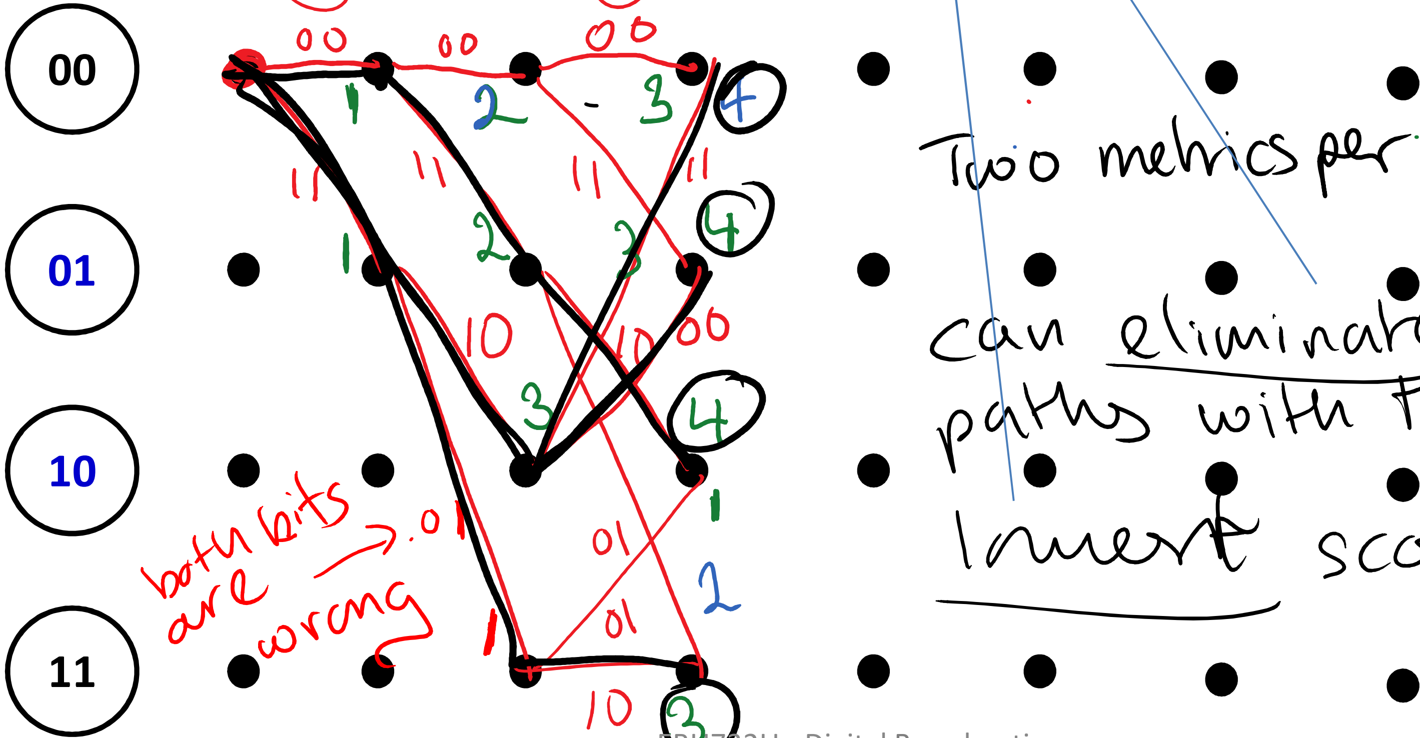


Metric: cumulative number of correct bits

Correct sequence 11 10 00 01 01 11

error

Received: 10 10 10 01 01 11 00

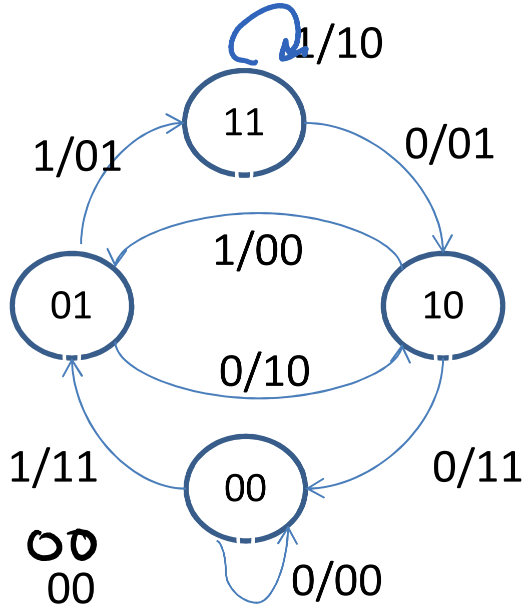


Two metrics per state.

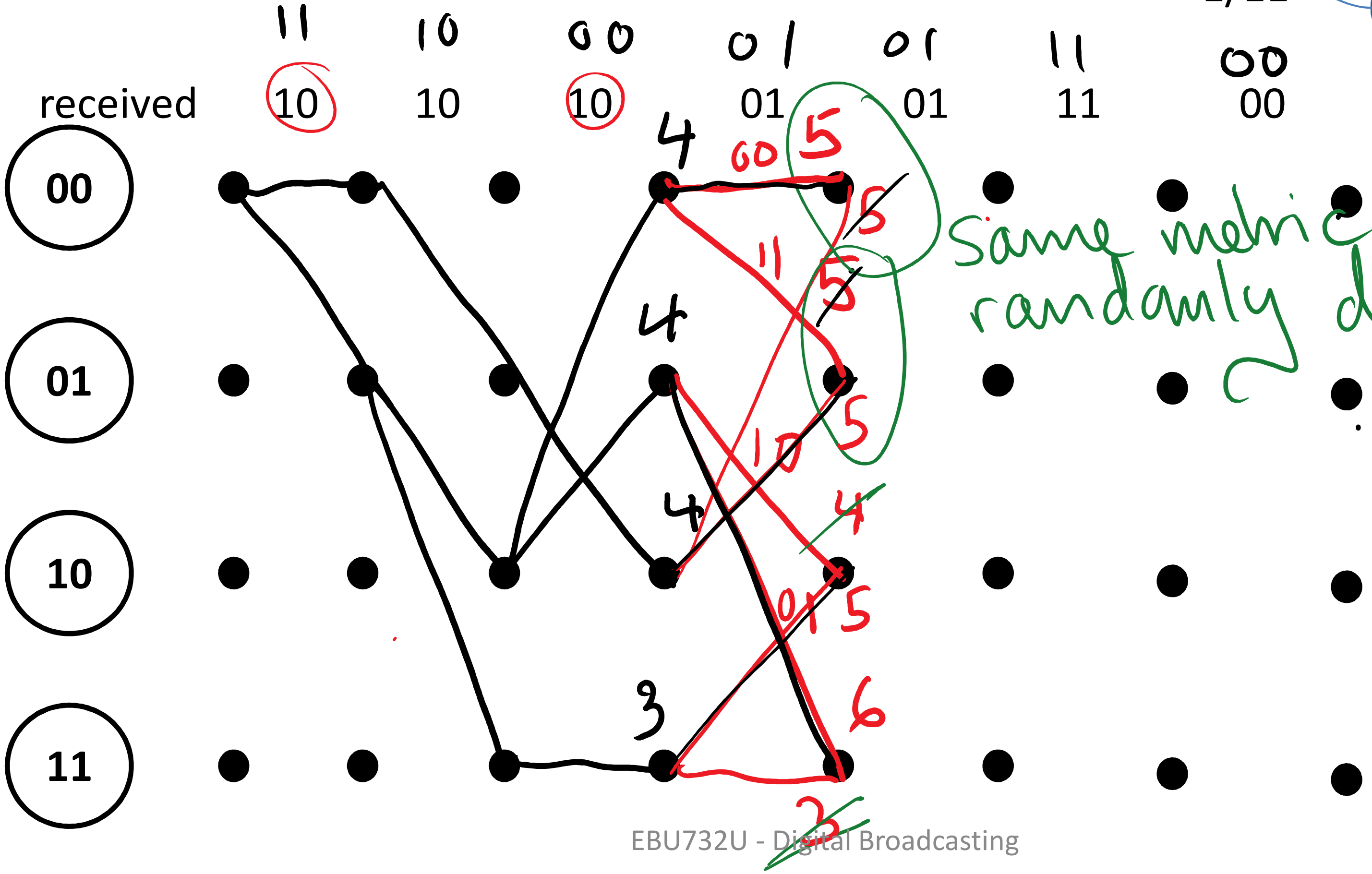
can eliminate the paths with the lowest scores

NB: Reimers: different state convention: state = S_2S_1 !! same trellis

Viterbi Decoding



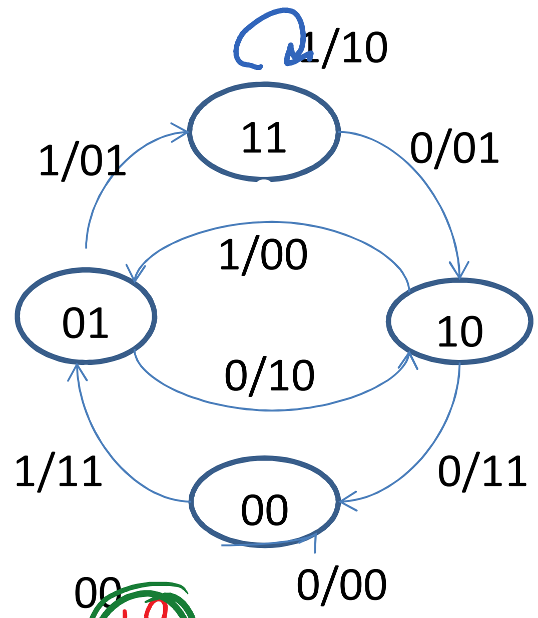
- Path with highest weight (correct # received bits) is the most likely to have occurred so far
  - $\Rightarrow$ eliminate the other arriving path weight
- Paths with <u>equal</u> weights: ambiguous decoding

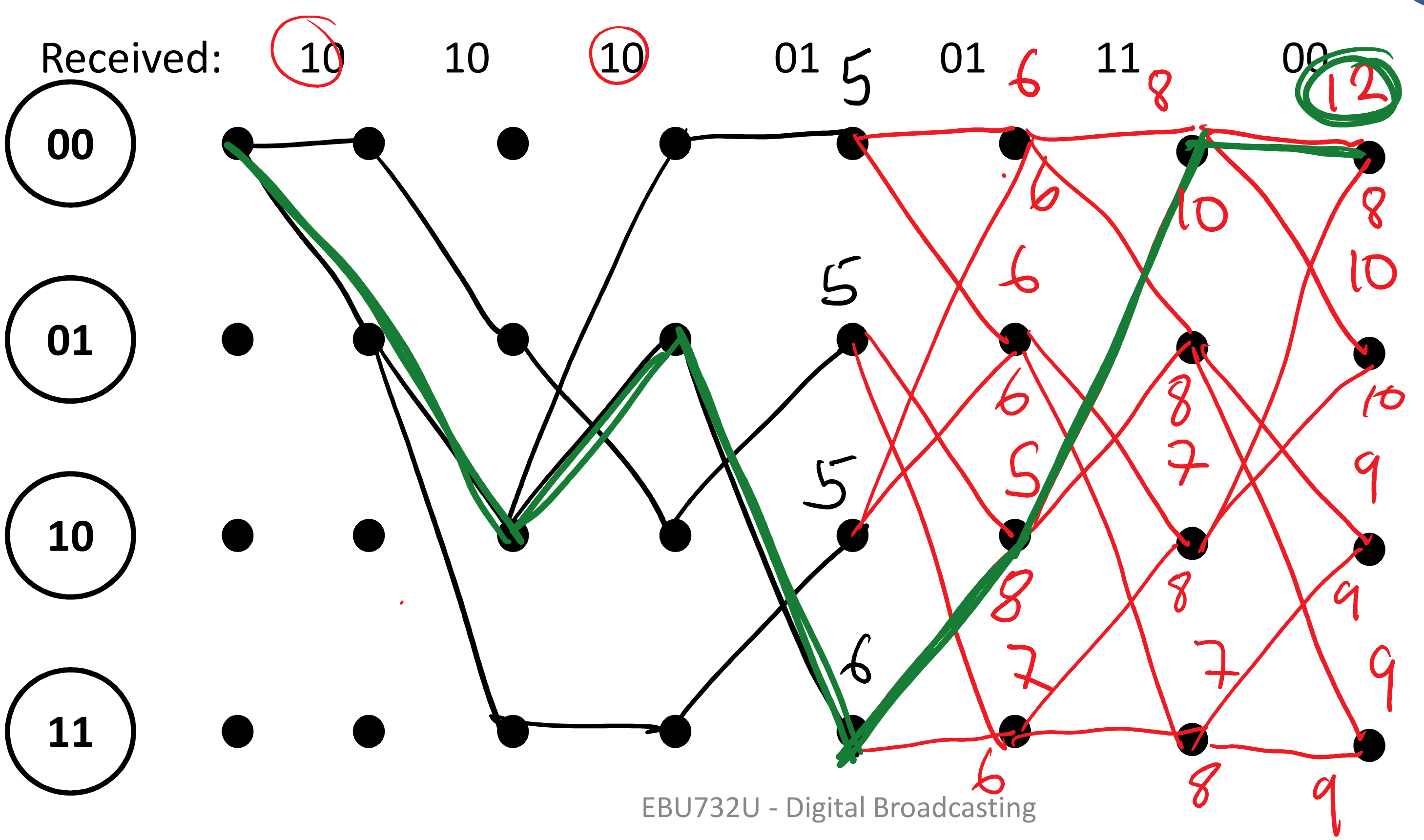


same metric so randomly delete one

Viterbi Decoding

highest path is the most probable





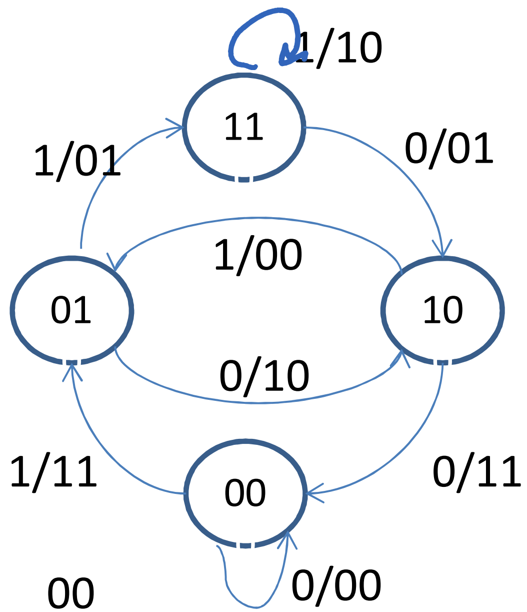
Number of errors that occurred
$= 7 \times 2 - 12$
$= 2$

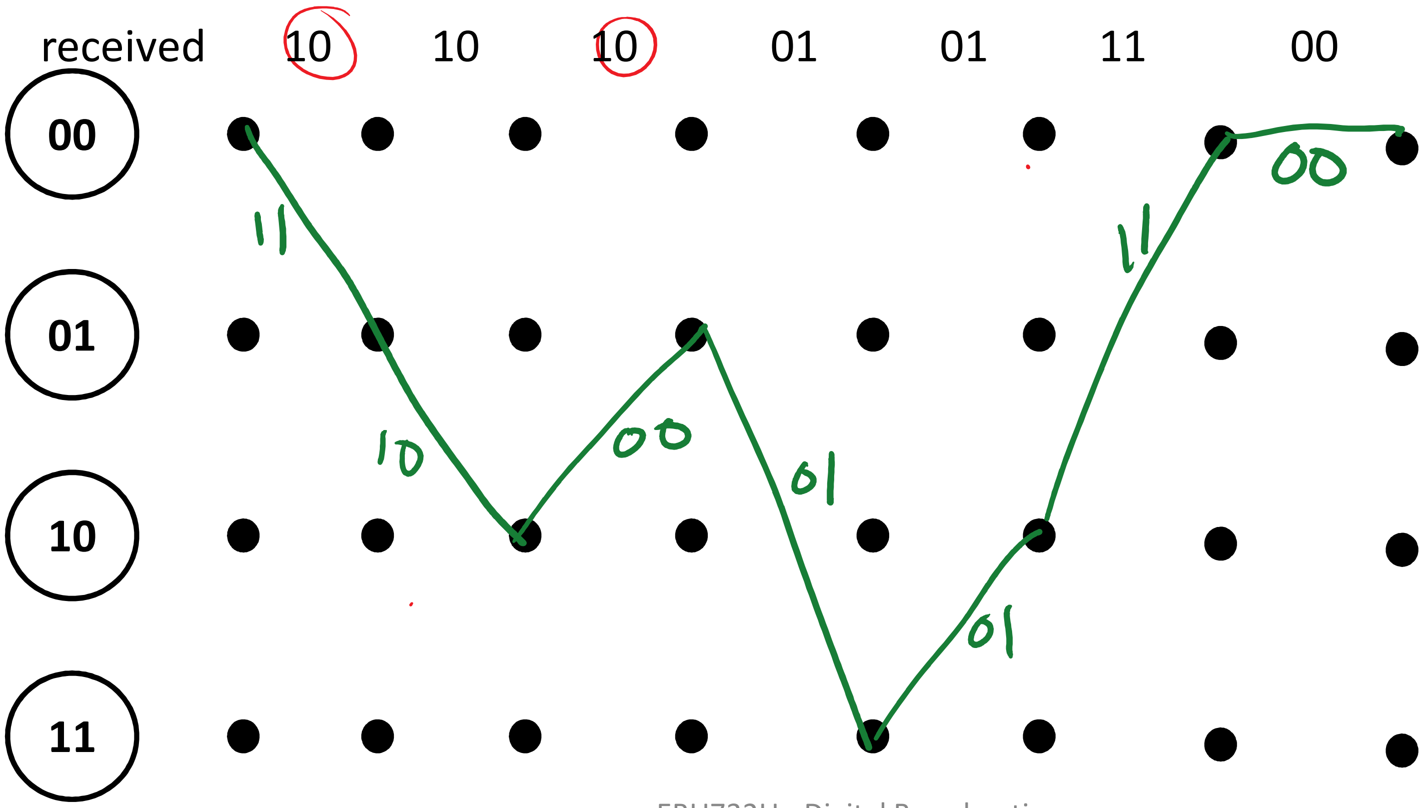
Viterbi Decoding

Predicted path

11 10 00 01 01 11 00

correct! (see slide 33)





Traceback

- Traceback = decoding the optimum path in the trellis in backward direction, to estimate the original input bits
  - Requires backward state transitions
- In principle, traceback requires the entire coded output sequence to be known before backward procedure can be started (updated weights)
- In practice: it can be shown: traceback depth = $5 \times K$ is sufficient to give good performance of (non-punctured) Viterbi decoder

Code Concatenation

- What? Serial chaining/encapsulation of two different encoding methods
  - Typically: outer = Reed-Solomon, inner = convolutional
- +: further improved error correction power
- -: slower rate of data transmission: $R_u/(R_1R_2)$