Department of Electrical Engineering
University of Arkansas

ELEG 5693 Wireless Communications
Ch 4. Coding

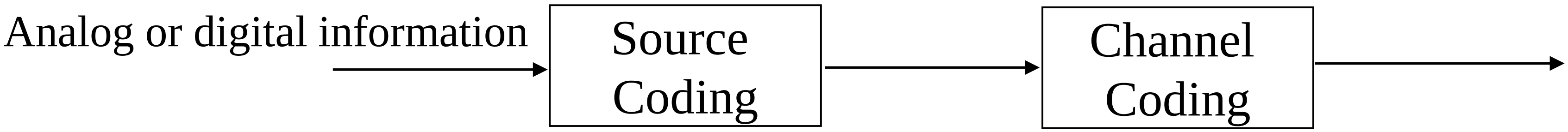
Dr. Jingxian Wu
wuj@uark.edu

OUTLINE

- **Introduction**
- **Source Coding**
- **Channel Coding: Convolutional Code**
- **Interleaving**

INTRODUCTION

- **Coding**
  - Source coding
    - Convert analog information into digital representation
    - Reduce the redundancy in the digital signal (compression)
  - Channel coding
    - Protect the information from channel distortions by adding redundancy.
    - Cyclic Redundancy Check (CRC), Linear Block Code, Convolutional Code (CC), Turbo Code, Low Density Parity Check (LDPC), etc.
- **Coding can only be used in digital communication systems.**



UNIVERSITY OF ARKANSAS

OUTLINE

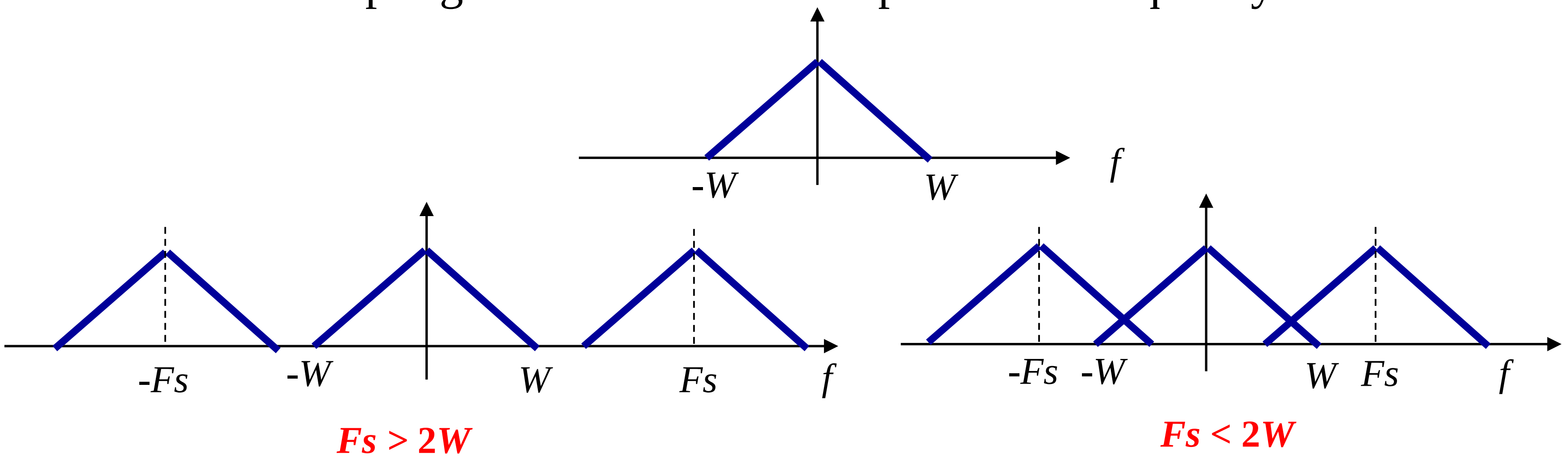
- Introduction
- **Source Coding**
- Channel Coding: Linear Block Code
- Channel Coding: Convolutional Code
- Interleaving

SOURCE CODE

- **Why source code?**
  - Convert analog signal into digital signal
    - Sampling and quantization, speech coding.
  - Reduce redundancy in digital signal representation (compression)
    - To save bandwidth ➔ improve bandwidth efficiency.
    - E.g. Winzip
- **Source code can be classified into two categories**
  - Lossless source code
    - No information is lost during compression
    - The original information can be perfectly recovered from the compressed information.
  - Source code with loss
    - Some information is lost during compression
    - The original information cannot be perfectly recovered after compression.
    - Analog to digital conversion, JPEG, MPEG.

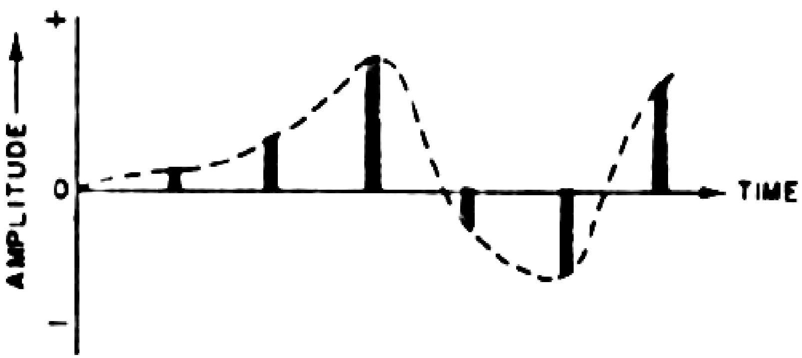
SOURCE CODE: SAMPLING

- **Sampling and quantization**
  - Convert analog information bearing signal to digital signal without significant loss of information.
- **Sampling theorem**
  - A band-limited signal with highest frequency W Hertz can be completely recovered from its samples if the sampling rate F_s is higher than $2W$ Hertz.
    - Sampling in time-domain ➔ repetition in frequency domain.

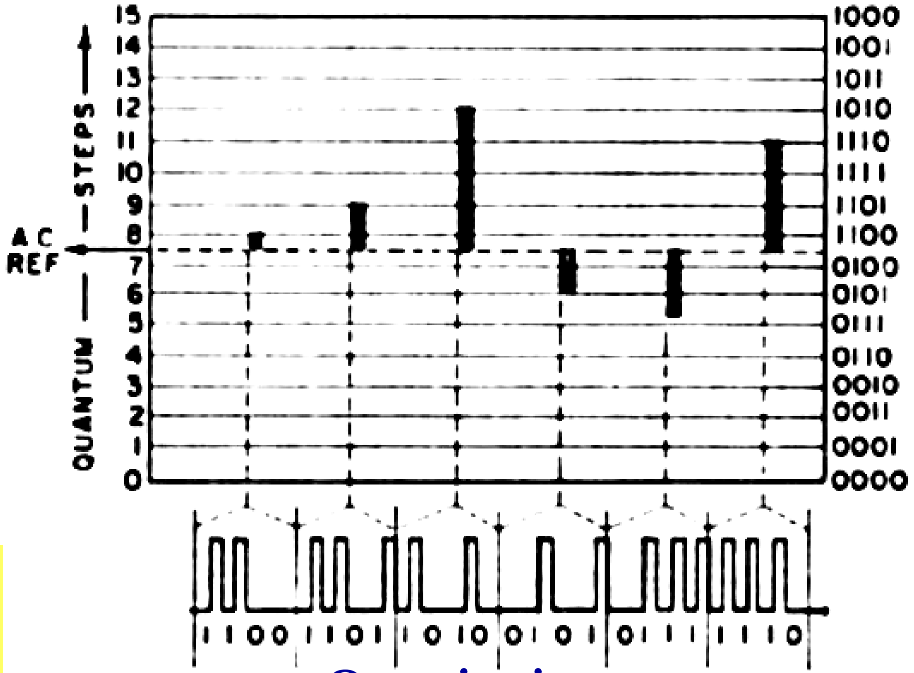


SOURCE CODE: PCM

- **Pulse Code Modulation (PCM): sampling and quantization**
  - Sampling: 8000 Samples/second (125 us/sample)
    - Bandwidth of speech signal in telephone system: 4KHz.
  - Quantization: represent each sample by 8-bit sequence (256 discrete levels)
  - Data rate: 8000 x 8 = 64 kbps



Sampling



Quantization

64kbps used in wired telephone system:
too high for wireless communication!

SOURCE CODE: ENTROPY

- **There is redundancy in the representation of information.**
  - Wirxlxss Commuxication (21 characters)
  - For efficient information transmission, we want reduce the redundancy
  - Given random data sequence, what is the maximum redundancy in the sequence?
    - OR: what is the minimum # of bits that can be used to represent the original data without loss of information at receiver?
    - OR: what is the maximum compression rate?
- **Entropy**
  - Entropy: the minimum # of bits required to represent one symbol from an information source

$$H = \sum_{k=0}^{K} p_k \log_2(1/p_k) \quad \text{(bit/sym)}$$

  - K: total # of possible symbols (e.g. 26 English characters)
  - p_k : the probability that the k-th character is generated by the source.

SOURCE CODE: ENTROPY

- **E.g. 1: two symbols: ‘0’, ‘1’**
  - p0 = 0.5, p1 = 0.5
  - H =
  - If there are 100 binary symbols, it can be represented with bits.
- **E.g. 2: two symbols: ‘0’, ‘1’**
  - p0 = 0.9, p1 = 0.1
  - H =
  - If there are 100 symbols, they can be represented with bits.
- **E.g. 3: two symbols: ‘0’, ‘1’**
  - p0 = 0, p1 = 1
  - H =
  - If there are 100 symbols, they can be represented with bits.
- **E.g. 4: the entropy of English is 1.1 ~ 1.6 bits/character**
  - 26 characters log2(26) = 4.7 bits
  - Approximately 2/3 are redundant
  - If there are 100 English characters, they can be represented by 160 bits
    - Against: 470 bits

SOURCE CODE: SPEECH CODING

- **Speech coding**
  - Convert analog speech signal into digital signal.
  - To reduce the bit rate *R* as much as possible.
    - R ↓ ➔ BW ↓ ➔ more users in limited spectrum
    - R ↓ ➔ voice quality ↓ (in general)
  - Tradeoff between bandwidth efficiency and voice quality.

SOURCE CODE: SPEECH CODING

- **Two basic speech coding schemes**
  - Waveform coders:
    - Strive to reproduce time or frequency domain signal waveform as precisely as possible.
    - Source independent
    - Moderate complexity and data rates (30 ~ 50 kbps)
    - e.g. PCM. Usually used in wired telephone system.
  - Vocoders (also called “source code”)
    - Analyze & extract key parameters using *a priori* knowledge of speech characteristics
      - Extract speech model parameters
      - Synthesize voice in Rx using model parameters
    - Signal specific parameters ➔ depends on user and is less robust
    - Produces very low data rates (~ 5–15 kbps)
      - Very complex & computationally intensive
    - Cellular & PCS applications where minimizing user BW → more users supported in finite spectrum → more $$

SOURCE CODE: SPEECH CODING

- **Vocoders**
  - Model the speech generation process of vocal tract of human
  - Parameters of speech
    - Voice pitch ➔ difficult to extract, usually < 300 Hz
    - Pole frequencies ➔ resonant frequencies of vocal tract
      - Centered around: 500, 1500, 2500, 3500
    - Pole amplitudes ➔ relative strength at different pole frequencies
    - Speech type ➔ voiced or unvoiced
      - Voiced: “m”, “n”, “v” ➔ voice chord vibrations
      - Unvoiced: “f”, “s” ➔ air flow through constriction
  - These parameters are transmitted by the sender
    - Rx uses these parameters to synthesize the human speech
  - Very complicated, but low data rate: 5 ~ 13 kbps
  - Source dependent
    - Suitable for human speech, but not suitable to other sound (e.g. music)

SOURCE CODE: SPEECH CODING

- **Linear Predictive Coders (LPC)**
  - A kind of time-domain vocoder
    - Extract the time domain parameters of signal.
    - Transmit the parameters of the signal instead of the actual waveform.
  - Linear predictive: predict the future value based on current values.
    - Time domain speech waveform: $\mathbf{x} = [x_1, x_2, \cdots, x_N]$
    - The current values and future values are correlated!

$$\hat{x}_{N+1} = \sum_{n=1}^{N} a_n x_n = \mathbf{a}^T \mathbf{x}$$

$$\mathbf{a} = [a_1, a_2, \cdots, a_N]^T \qquad \mathbf{x} = [x_1, x_2, \cdots, x_N]^T$$

  - The coefficients **a** are calculated at Tx based on the statistical properties of **x**.
    - **x** is a random process.
  - Instead of transmitting **x**, transmit **a**!
- **How to calculate the coefficients? – choose a to minimize the mean square error (MSE).**

OUTLINE

- Introduction
- Source Coding
- **Channel Coding: Convolutional Code**
- Interleaving

CHANNEL CODING: OVERVIEW

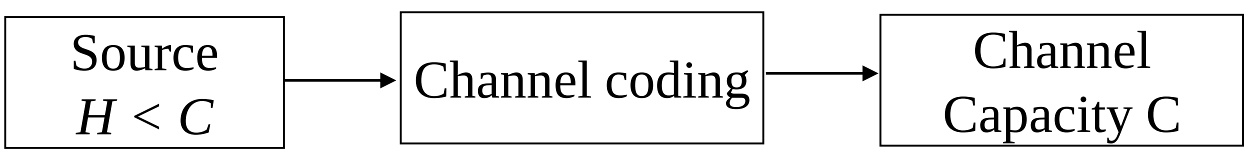
- **Channel coding**
  - Protect the transmitted information by adding redundancy.
  - E.g. repetition code:
    - '0': '000'
    - '1': '111'
- **Error detection**
  - Include only enough redundant information such that the Rx can detect an error by looking at the Rx data.
    - E.g. repeat '1' 2 times. Tx (1 1), Rx (0 1) ➔ Receiver knows there is an error, but couldn't guess what is transmitted
    - Send back Negative Acknowledgement (automatic-repeat request: ARQ)
- **Error correction**
  - Include enough redundant information such that the Rx can recover the original information by looking at the Rx data.
    - E.g. repeat '1' 3 times. Tx (1 1 1), Rx (0 1 1) ➔ Receiver will guess that (1 1 1) is transmitted ➔ detect '1'
    - Majority decision rule ➔ minimize the probability of error.

CHANNEL CODING: CHANNEL CAPACITY

- For an AWGN channel with bandwidth B, the maximum data rate that can be supported by the channel is

$$C(\text{bps}) = B\log_2(1+SNR)$$

- Shannon's coding theorem
  - For a channel with capacity C bps and an information source generates information at a rate less than C, then there exists a channel coding technique such that the output of the source can be transmitted by the channel with arbitrarily low error rate.



CHANNEL CODING: LBC

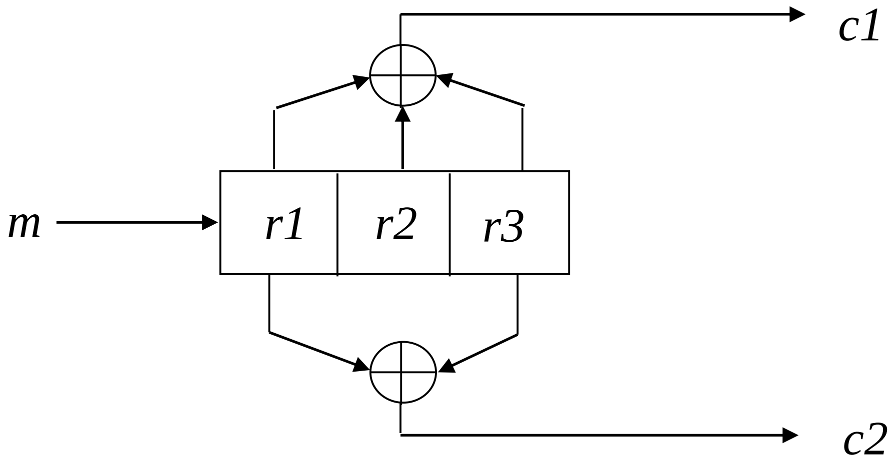
- **Linear block code (LBC)**
  - Every *k* bits of information corresponds to a codeword of length *n* bits
    - E.g. repetitionon code 1-bit of information, 3-bit codeword
  - n > k: there are (*n*-*k*) bits of redundancy
  - The code is called: (*n*, *k*) linear block code
  - Definition: code rate = (information block length)/(codeword length)
    - $r = k/n$
    - Measures the efficiency of the code (1-*r*: the percentage of redundancy)
    - E.g.: (3, 1) repetition code: r = 1/3. (2, 1) repetition code: r = 1/2.

CONVOLUTIONAL CODING

- **What is convolutional code (CC)**
  - n-bit codeword depends on not only current input, but also previous input
    - The encoder has memory
  - linear block code: n-bit codeword determined uniquely by the k-bit information
  - Compared to linear block code
    - Can achieve larger $d_{\min}$ with higher coding rate.
      - Better power efficiency with larger bandwidth efficiency
    - More complex than LBC
- **Parameters**
  - (n, k, K)
  - Every k-bit input leads to n-bit output
    - Coding rate $r = k/n$
  - K: constraint length (related to memory depth of the encoder)

CC: ENCODER

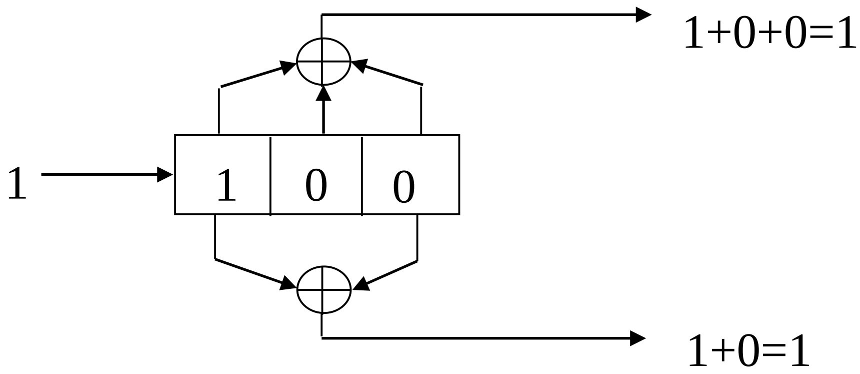
- **Register representation**



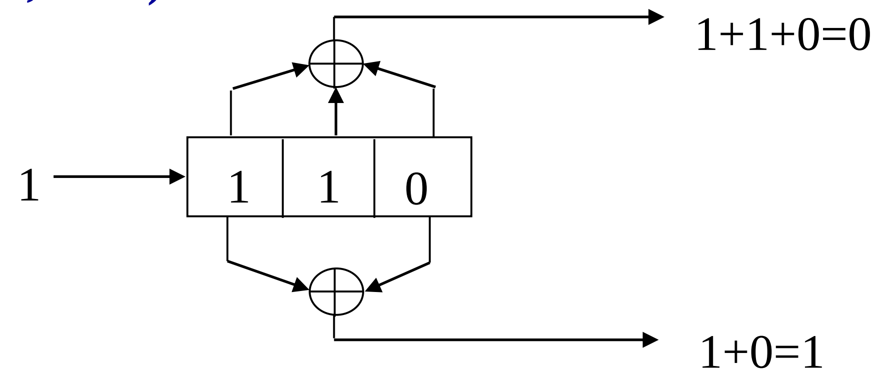
- *m*: current 1-bit input.
- r1, r2, r3: contents in shift register. Depends on current input and previous input
  - r1 = m: current input
  - r2, r3: previous inputs
- *c1* and *c2*: 2-bit output. Depends on r1, r2, r3
  - c1 = r1 + r2 + r3
  - c2 = r1 + r3
- (n = 2, k = 1, K = 3)

CC: ENCODER

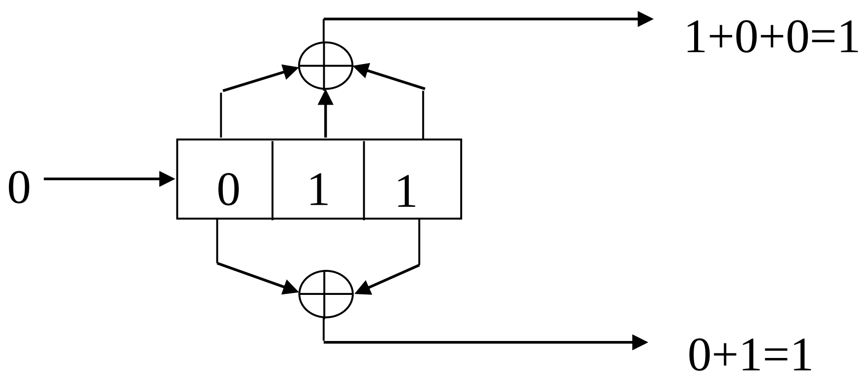
- **Example: m = [1 1 0 0], initially (r1=0, r2=0, r3=0)**



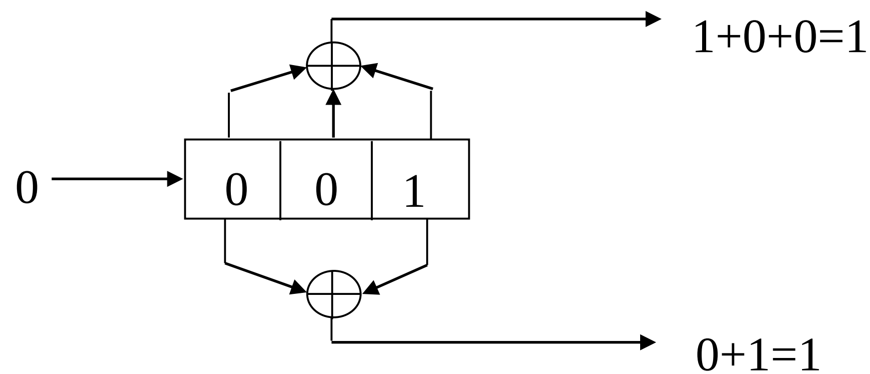
first input bit: 1



second input bit: 1
(shift 1 bit to the right)



third input bit: 1
(shift 1 bit to the right)



fourth input bit: 1
(shift 1 bit to the right)

CC: ENCODER

- **State**
  - Every input depends on current input r1=m, and previous inputs (r2, r3)
  - State: (r2, r3): state a: 00; state b: 10; state c: 01; state d: 11
- **Input-output table representation**

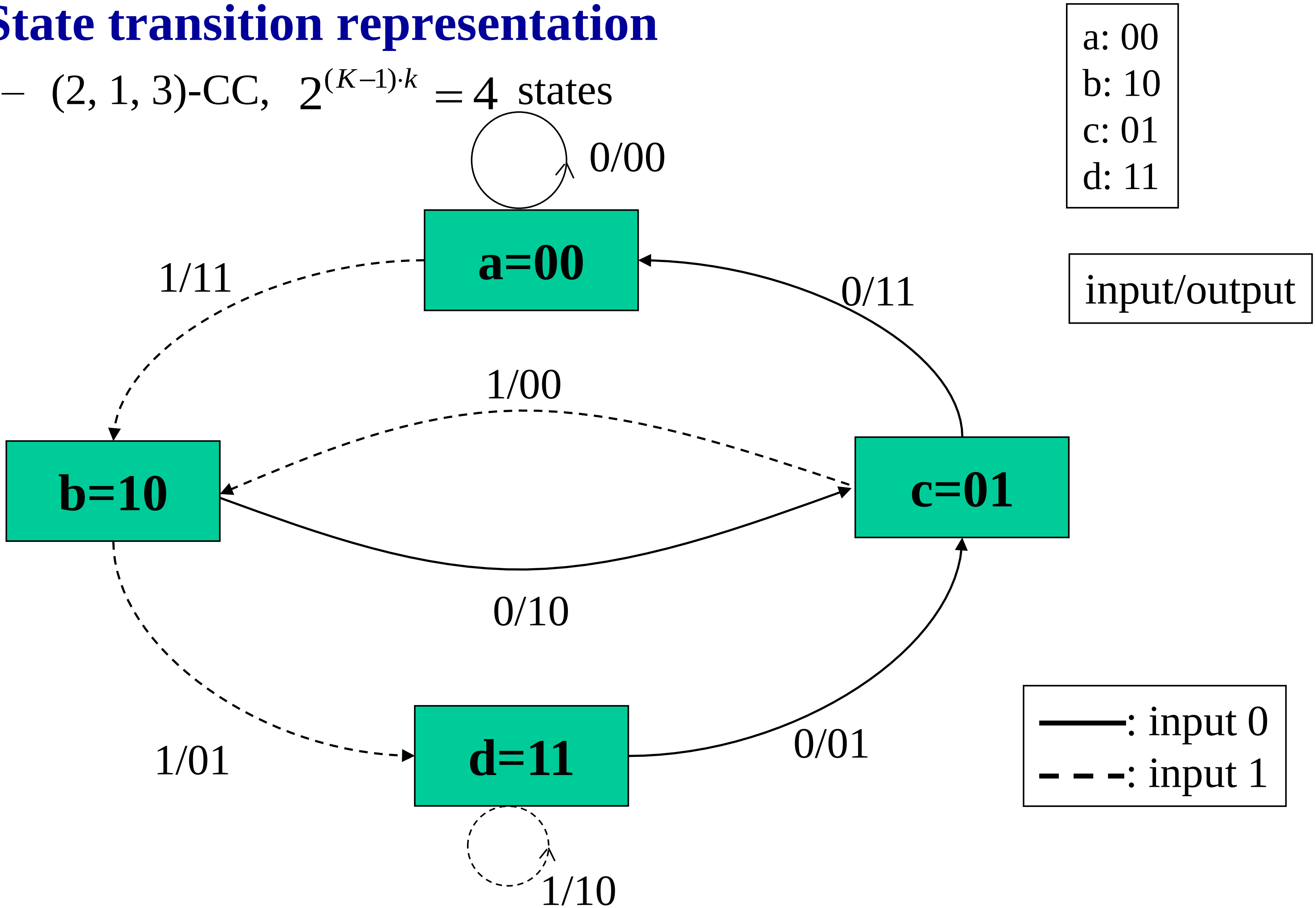
Input	Current state (r2, r3) now	(r1, r2, r3)	Next state (r2, r3) after shift	Output (c1, c2)
0	a (0 0)	0 0 0	a (0 0)	(0, 0)
0	c (0 1)	0 0 1	a (0 0)	(1, 1)
0	b (1 0)			
0	d (1 1)			
1	a (0 0)			
1	c (0 1)			
1	b (1 0)			
1	d (1 1)			

CC: ENCODER

- **(n, k, K)**
  - n = 2: 2-bit of output
  - k = 1: 1-bit of input
  - K = 3: output depends on current input, and two previous inputs
- **State:**
  - Every output depends not only current input, but also $(K-1) = 2$ previous inputs ➔ the encoder remembers the previous $(K-1)$ inputs ➔ the encoder has a memory depth of $(K-1)$
  - For a particular 1-bit of input, the encoder might be in one of 4 possible states ➔ there are four possible outputs
  - Output depends on:
    - 1. current input
    - 2. state of the encoder
  - # of states: $2^{(K-1)\cdot k}$

CC: ENCODER

- **State transition representation**
  - (2, 1, 3)-CC, $2^{(K-1)\cdot k} = 4$ states



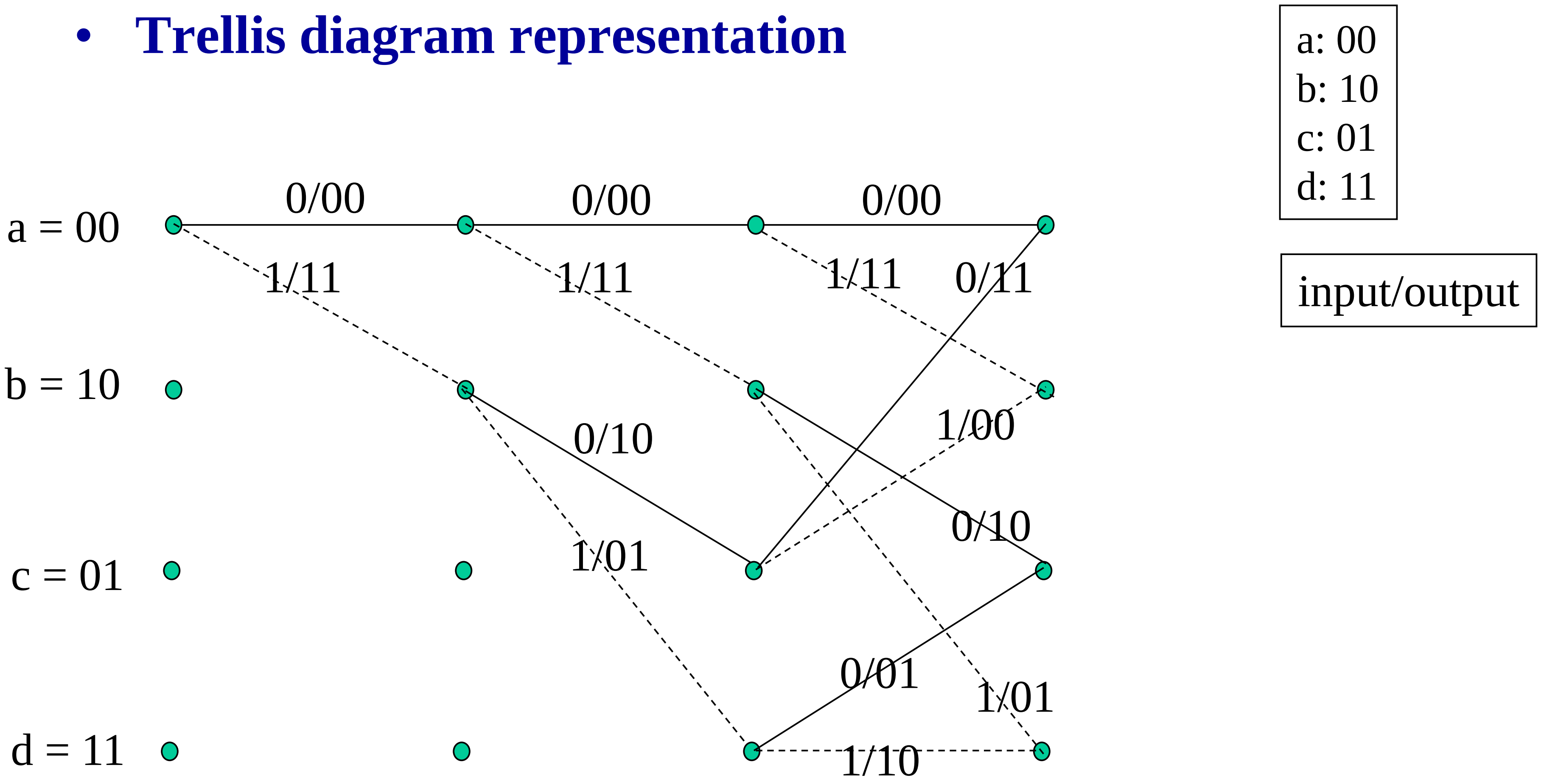
UNIVERSITY OF ARKANSAS

CC: ENCODER

- **Example: encoding by using state transition diagram**
  - Initial state: a
  - Input bits: (1 1 0 0 0 1 0)
  - 1st bit: (state = a, input = 1) ➔ (next state = b, output = 11)
  - 2nd bit: (state = b, input = 1) ➔ (next state = d, output = 01)
  - 3rd bit: (state = d, input = 0) ➔ (next state = c, output = 01)
  - 4th bit: (state = c, input = 0) ➔ (next state = a, output = 11)
  - 5th bit: (state = a, input = 0) ➔
  - 6th bit:
  - 7th bit:

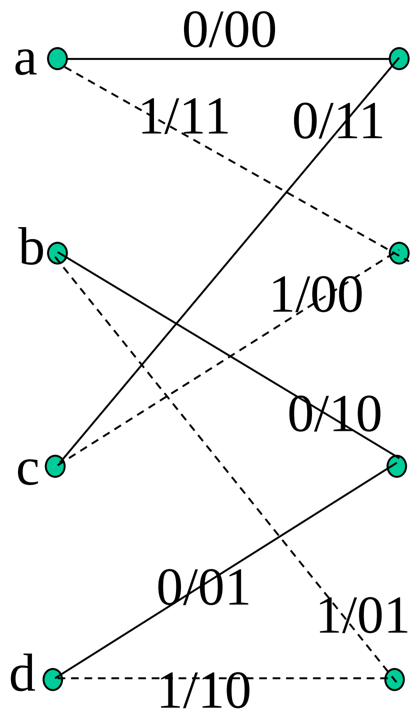
CC: ENCODER

- **Trellis diagram representation**



CC: ENCODER

- **Example:**
  - initial state: a
  - Input bits: (1 1 0 0 0 1 0)
  - 1st bit: (state = a, input = 1) ➔ (next state = b, output = 11)
  - 2nd bit: (state = b, input = 1) ➔
  - 3rd bit:
  - 4th bit:
  - 5th bit:
  - 6th bit:
  - 7th bit:

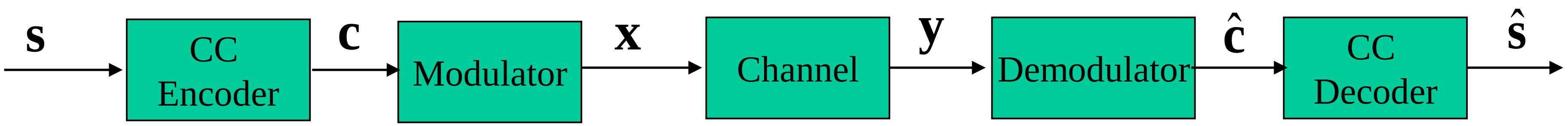


CC: ENCODER

- **Four alternative encoder representations:**
  - – Register representation
    - Usually used by hardware implementation of CC encoder
  - – Input-output table representation
  - – State transition diagram representation
  - – Trellis diagram representation
    - Mainly used by decoder for decoding
  - – The 4 alternative representations are equivalent
    - Given one representation, we can easily find out the other three representations.

CC: DECODER

- **Optimum hard decoding**



- **s**: length *m* information vector (sequence of 0's and 1's)
- **c**: length $n = m/r$ CC codeword (sequence of 0's and 1's)
- **x**: modulated symbols (modulation symbols)
- **y**: received symbols (distorted by channels)
- $\hat{\mathbf{c}}$: length $n = m/r$ demodulated information (sequence of 0's and 1's)
- $\hat{\mathbf{s}}$: decoded length *r* information vector (sequence of 0's and 1's).

- If the block length is *m*, there are 2^m possible codewords
  - Decoding: find the codeword that has the smallest Hamming distance with $\hat{\mathbf{c}}$

CC: DECODER

- **Example: block length m = 3, initial state a (0 0)**
  - Input (0 0 0) ➔ output (0 0 0 0 0 0)
  - Input (0 0 1) ➔ output (0 0 0 0 1 1)
  - Input (0 1 0) ➔ output (0 0 1 1 1 0)
  - Input (0 1 1) ➔ output (0 0 1 1 0 1)
  - Input (1 0 0) ➔ output (1 1 1 0 1 1)
  - Input (1 0 1) ➔ output (1 1 1 0 0 0)
  - Input (1 1 0) ➔ output (1 1 0 1 0 1)
  - Input (1 1 1) ➔ output (1 1 0 1 1 0)

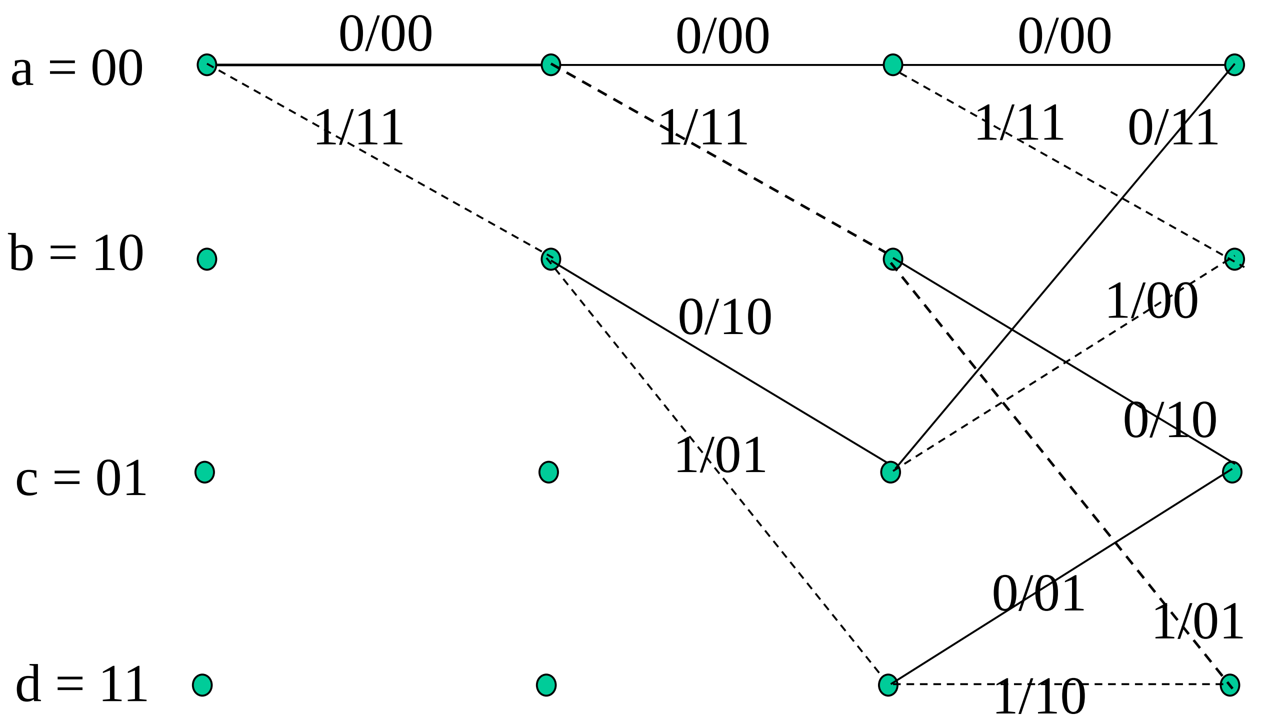
  - If the vector after demodulator is $\hat{\mathbf{c}} = [1\ 0\ 0\ 1\ 0\ 1]$
    - Distance with the 8 possible codewords
      - c1: 3, c2: 3, c3: 4, c4: , c5: , c6: , c7: , c8:
      - Winner:
      - Decoded information:

CC: DECODER

- **Optimum hard decoding: find the codeword with the smallest Hamming distance**
  - Exhaustive searching works fine when information block length is small
    - m = 3 ➔ 2^3 = 8 possible codewords
    - m = 8 ➔ 2^8 = 256 possible codewords
  - The computational complexity becomes inhibitively expensive when information block length is large
    - Typical information block length: 100
    - m = 20 ➔ $2^{20} = 1{,}048{,}576$
    - m = 100 ➔ $2^{100} = 1.27 \times 10^{30}$
  - When *m* is large, it's impossible to find the optimum codeword by exhaustive searching all the possible codewords!
  - The optimum hard decoding can be performed by exploiting the trellis structure of the encoder
    - Viterbi algorithm

CC: DECODER

- **Optimum decoding by using Viterbi algorithm**
  - The process of CC encoding is equivalent to find a path along the trellis transition diagram.
    - E.g. Input of CC encoder (1 1 0) ➔ codeword: (11 01 01)
  - Decoding: find out the path with the smallest Hamming distance with received codeword.



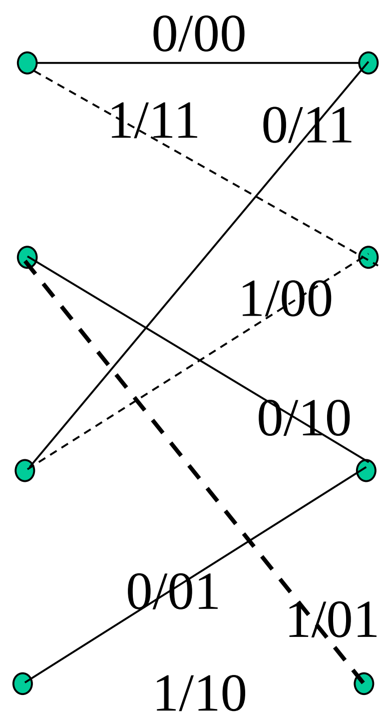
CC: DECODER

- **Optimum decoding by using Viterbi algorithm**
  - At each transition, for each ending state, find the branch minimizing the accumulated Hamming distance (survival branch)

$$d_k = d_{k-1} + d(\text{current output}, \text{branch output})$$

- d_{k-1}: accumulated Hamming distance from the previous transition
- $d(\text{current output}, \text{branch output})$: the Hamming distance of this branch

One survival branch for each ending state!

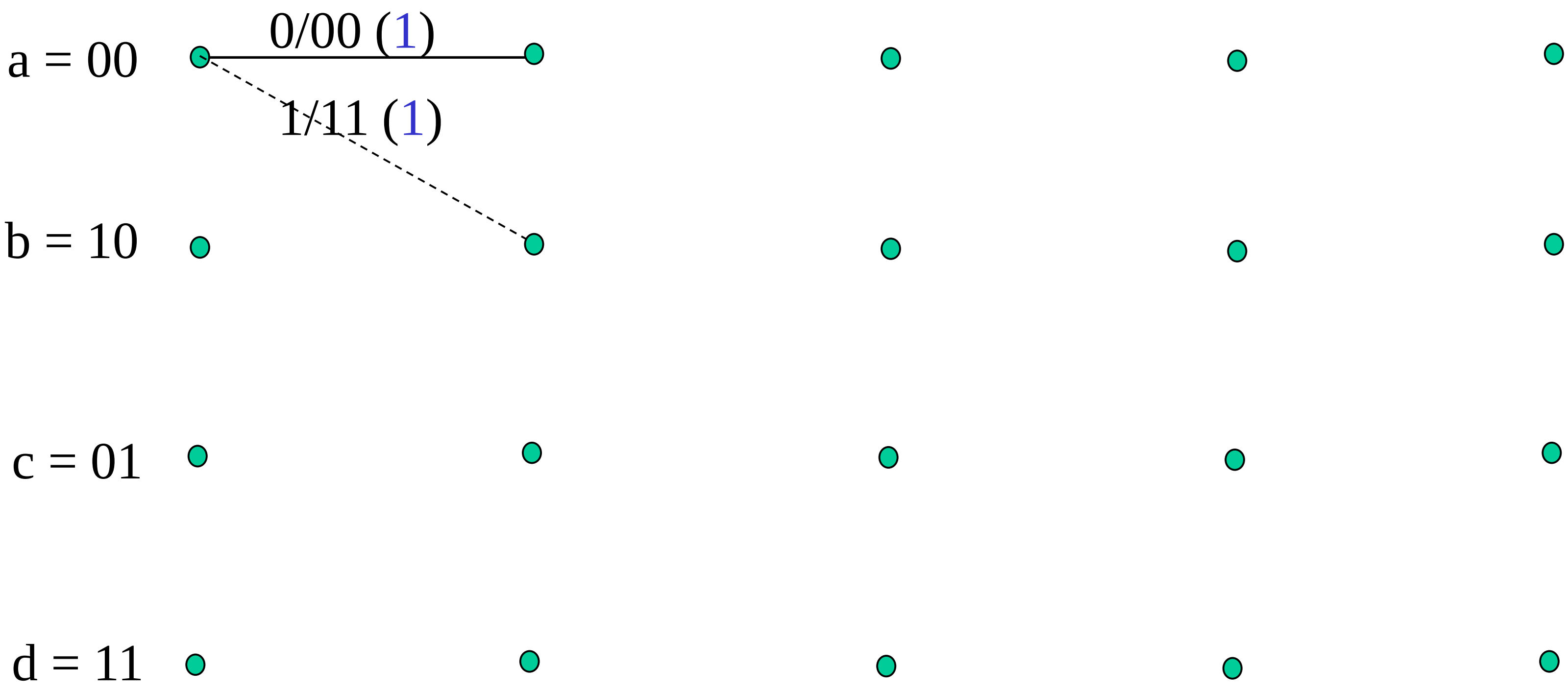


UNIVERSITY OF ARKANSAS

CC: DECODER

- **Viterbi algorithm: example (first transition)**

Output of demodulator: **10** **01** **01** **11**

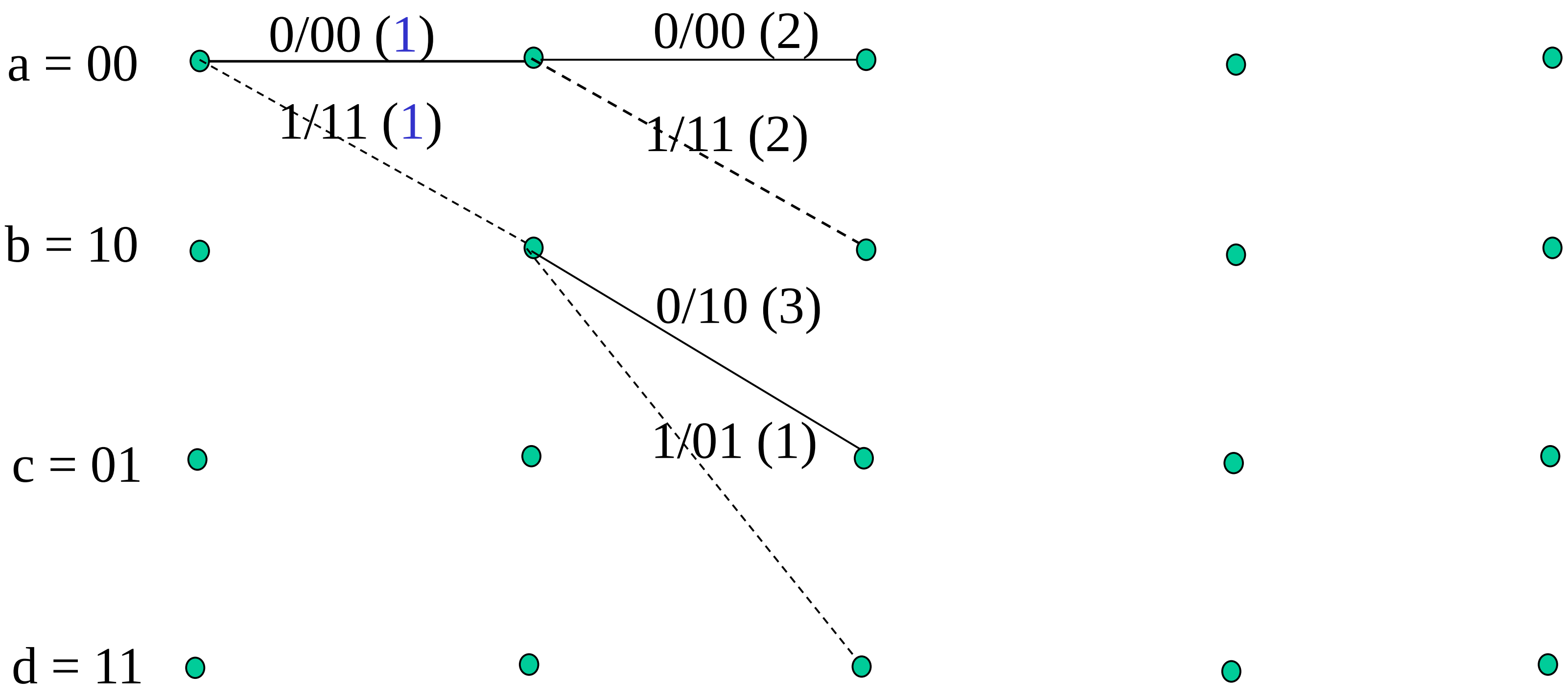


a: d1 = 0+d(10, 00) = 1 **b: d1 = 0+d(10, 11) = 1**

CC: DECODER

- **Viterbi algorithm: example (second transition)**

Output of demodulator: **10** **01** **01** **11**

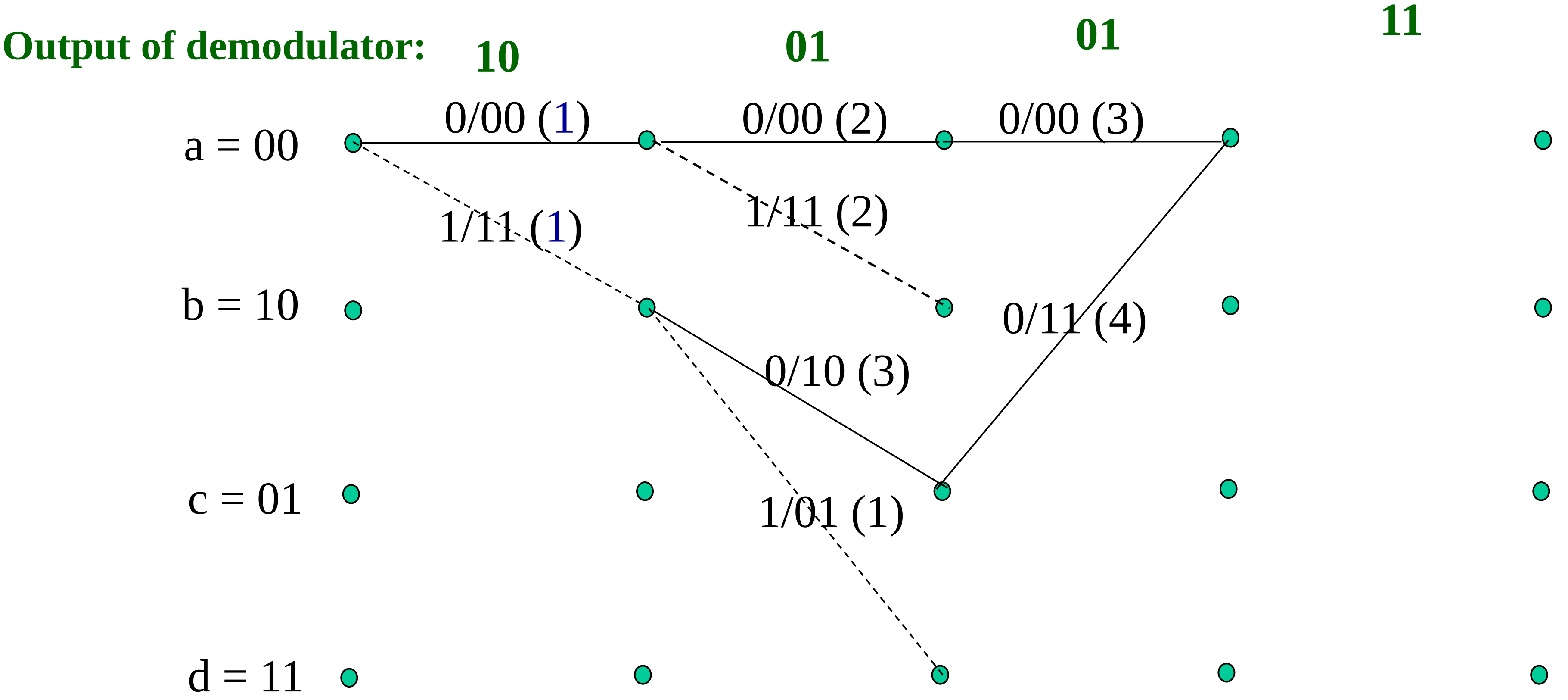


a: d2 = 1+d(01, 00) = 2 **b: d2 = 1+d(01, 11) = 2** **c: d2 = 1+d(01, 10) = 3** **d: d2 = 1+d(01, 01) = 1**

UNIVERSITY OF ARKANSAS

CC: DECODER

- **Viterbi algorithm: example (third transition)**

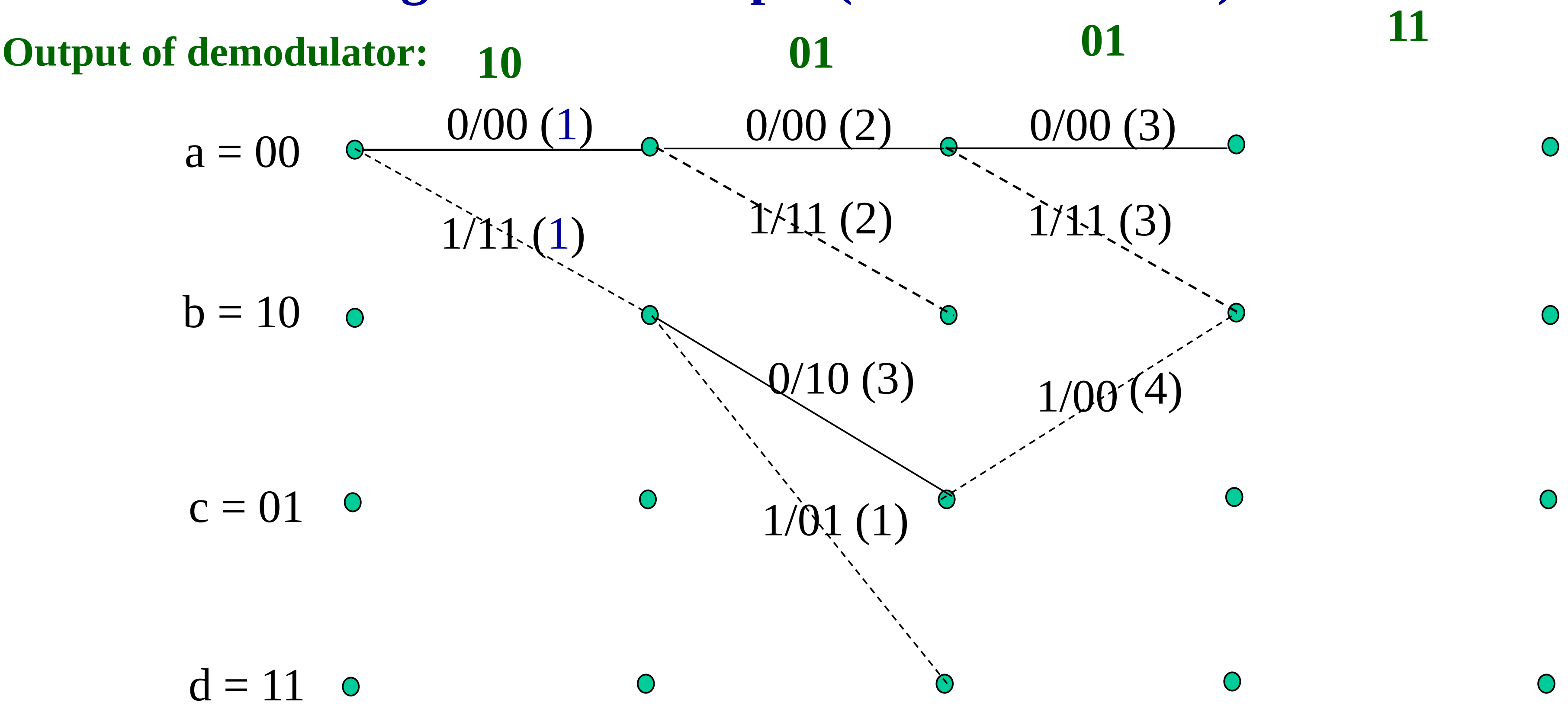


a: d3 = 2+d(01, 00) = 3

d3 = 3+d(01, 11) = 4

CC: DECODER

- **Viterbi algorithm: example (third transition)**

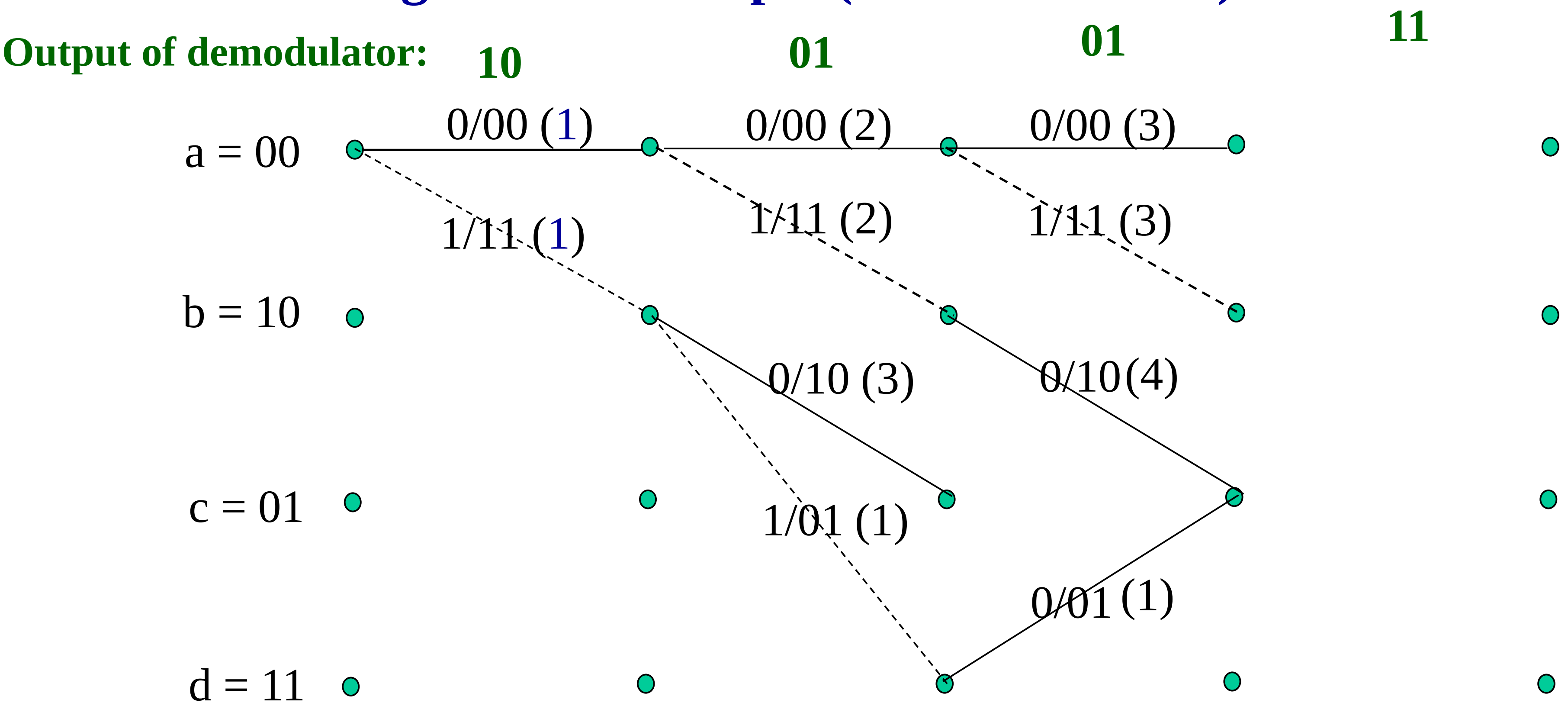


a: d3 = 2+d(01, 00) = 3 **b:** d3 = 2+d(01, 11) = 3

d3 = 3+d(01, 11) = 4 d3 = 3+d(01, 00) = 4

CC: DECODER

- **Viterbi algorithm: example (third transition)**



a: d3 = 2+d(01, 00) = 3

d3 = 3+d(01, 11) = 4

b: d3 = 2+d(01, 11) = 3

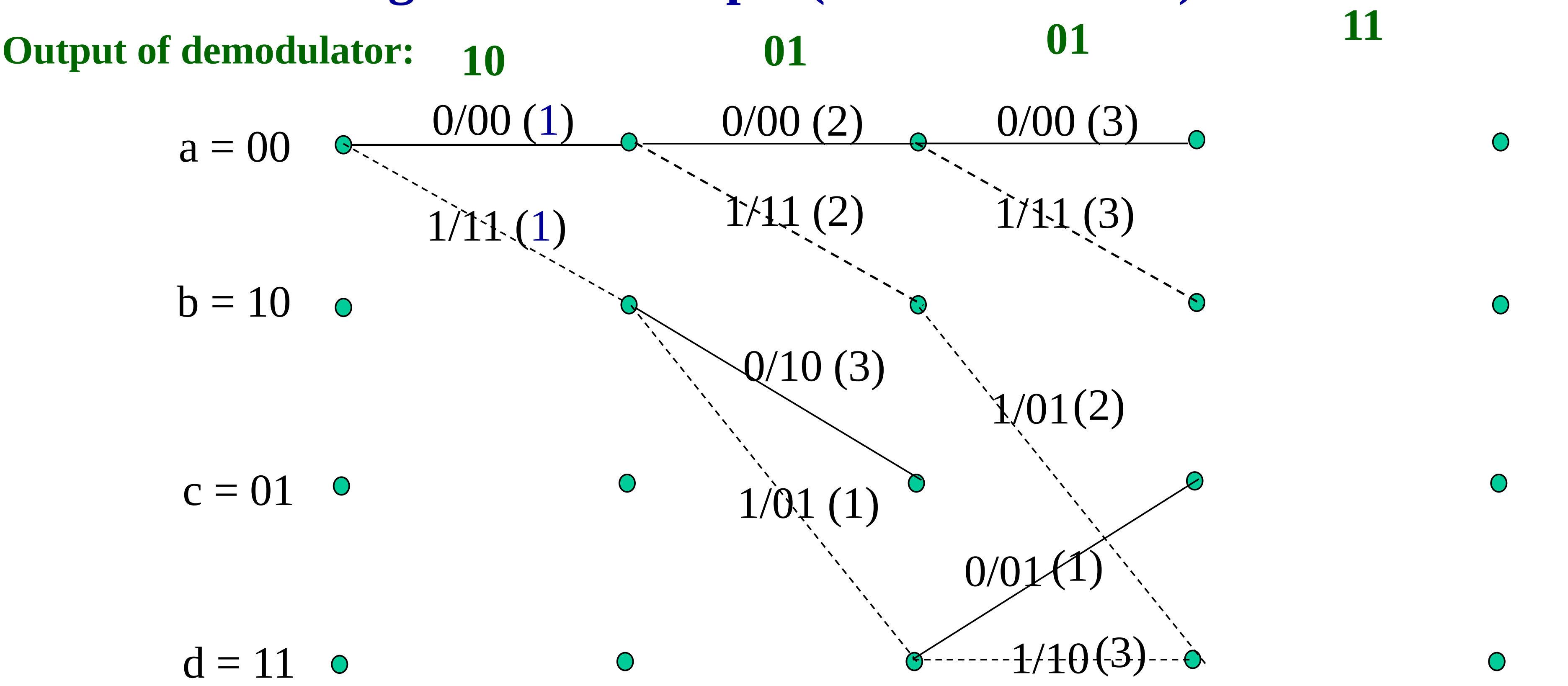
d3 = 3+d(01, 00) = 4

c: d3 = 2+d(01, 10) =

d3 = 1+d(01, 01) =

CC: DECODER

- **Viterbi algorithm: example (third transition)**



a: **d3 = 2+d(01, 00) = 3**

d3 = 3+d(01, 11) = 4

b: **d3 = 2+d(01, 11) = 3**

d3 = 3+d(01, 00) = 4

c: **d3 = 2+d(01, 10) =**

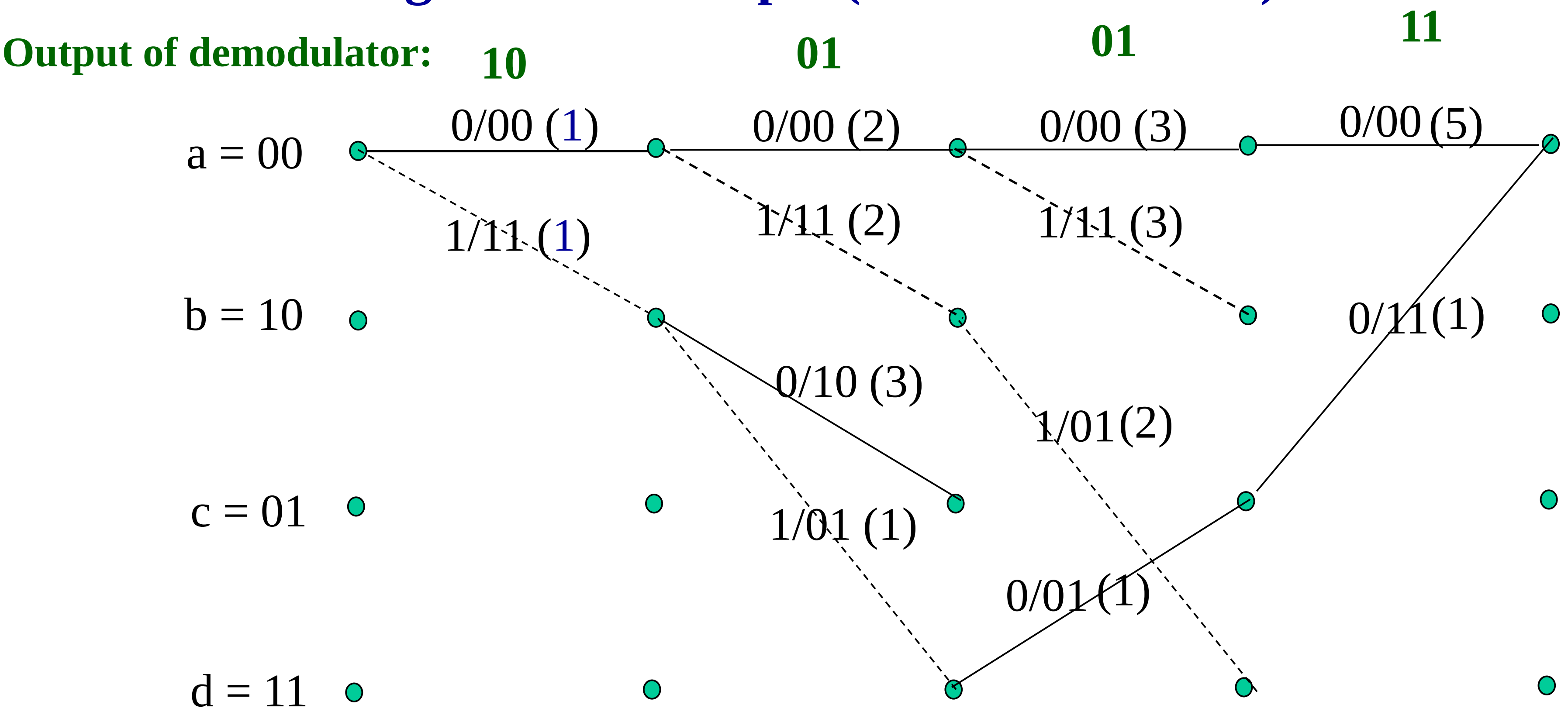
d3 = 1+d(01, 01) =

d: **d3 = 2+d(01, 01) =**

d3 = 1+d(01, 10) =

CC: DECODER

- **Viterbi algorithm: example (fourth transition)**



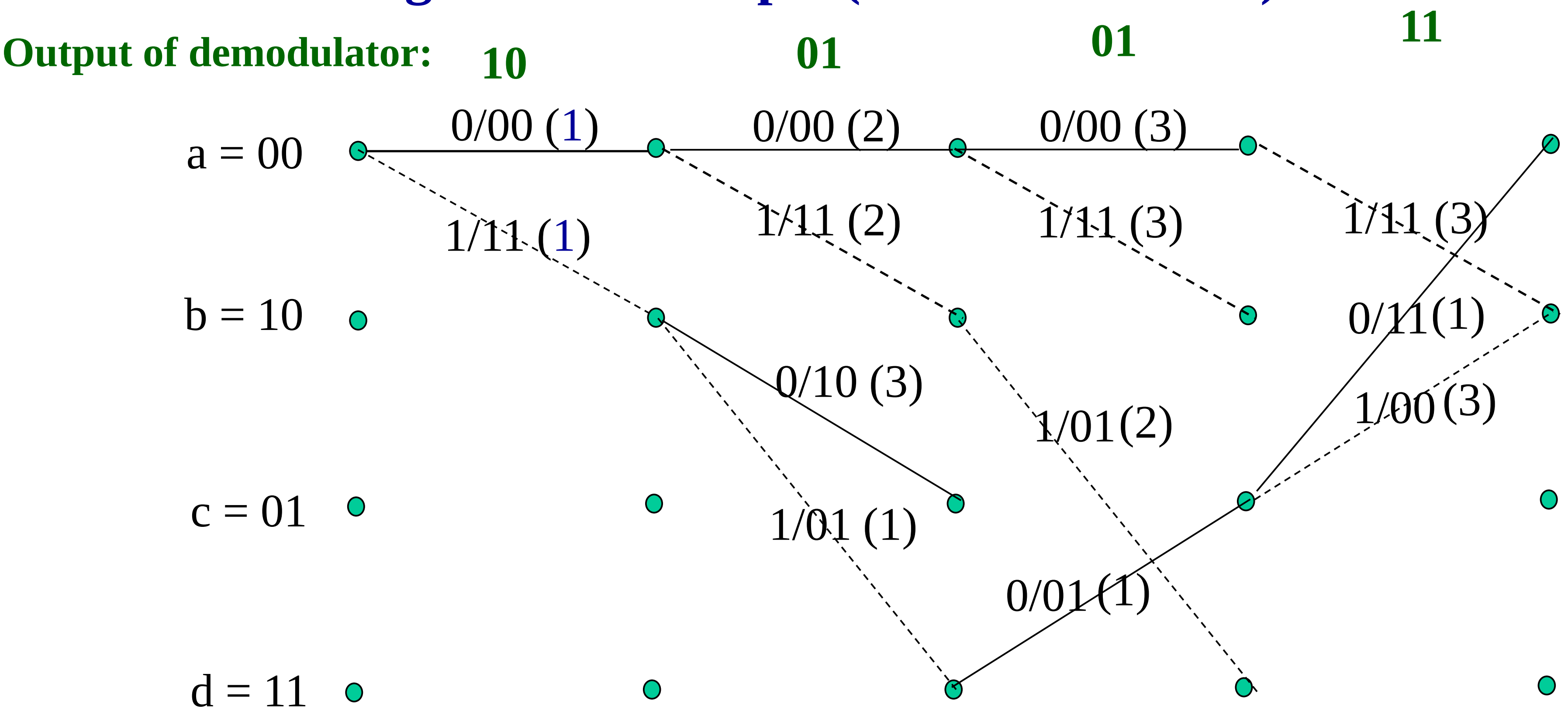
a: d4 = 3+d(11, 00) = 5

d4 = 1+d(11, 11) = 1

UNIVERSITY OF ARKANSAS

CC: DECODER

- **Viterbi algorithm: example (fourth transition)**

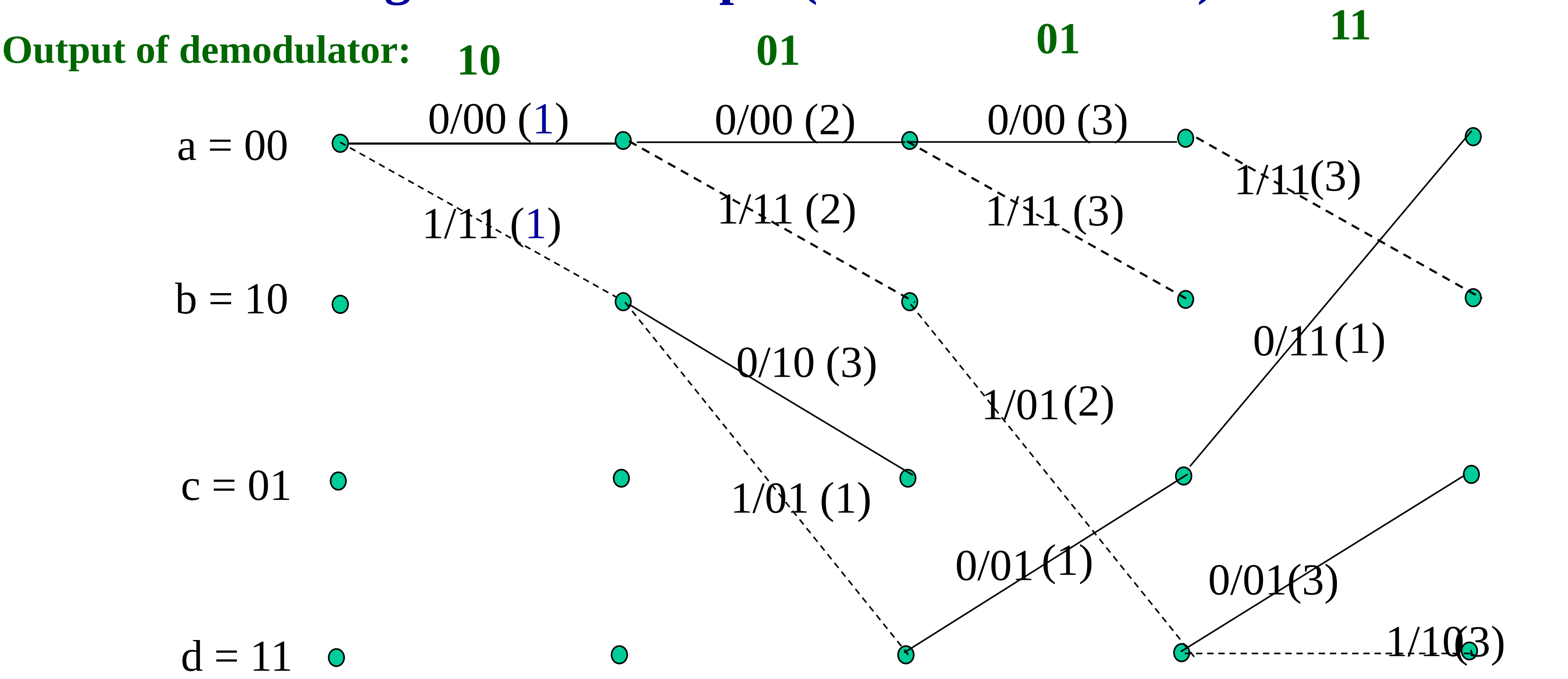


a: **d4 = 3+d(11, 00) = 5**   **b:** **d4 = 3+d(11, 11) = 3**

d4 = 1+d(11, 11) = 1   **d4 = 1+d(11, 00) = 3**

CC: DECODER

- **Viterbi algorithm: example (fourth transition)**



a: d4 = 3+d(11, 00) = 5
d4 = 1+d(11, 11) = 1

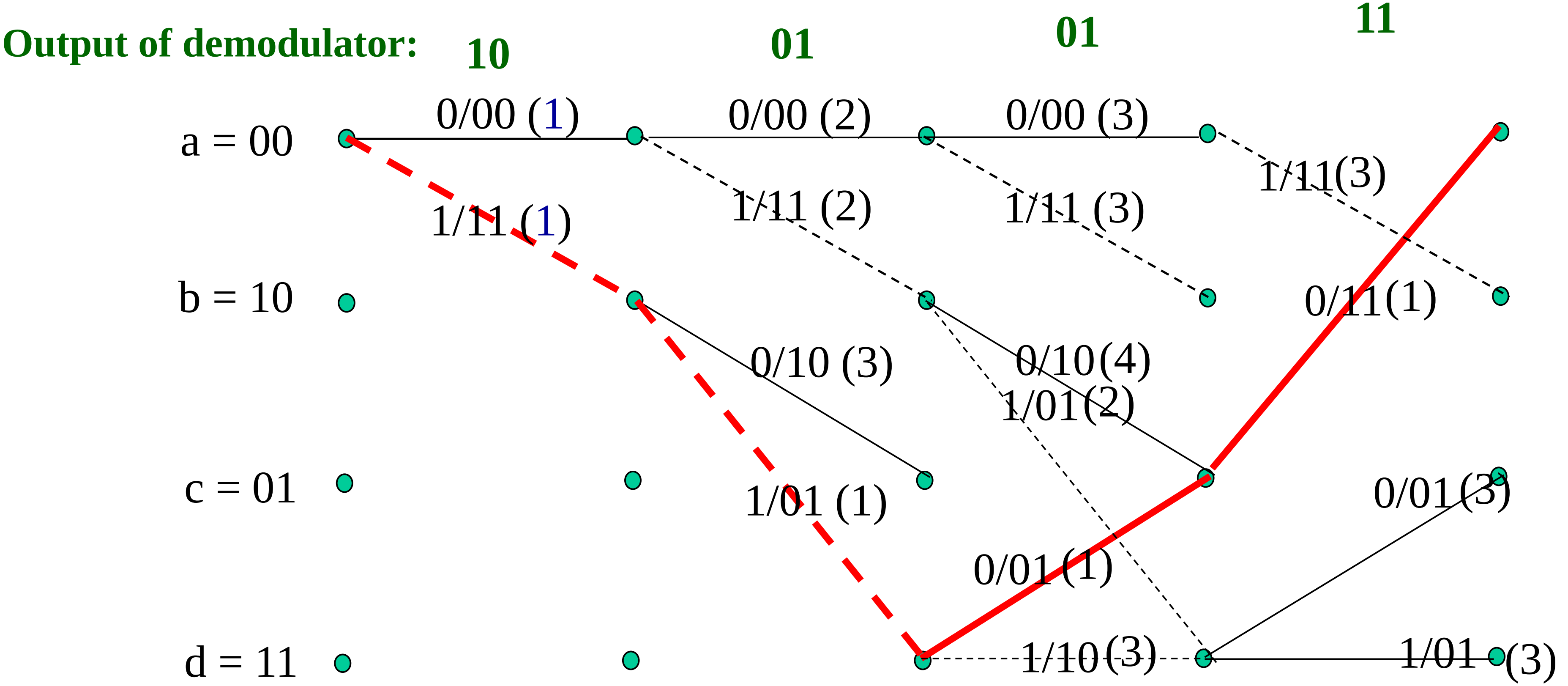
b: d4 = 3+d(11, 11) = 3
d4 = 1+d(11, 00) = 3

c: d4 = 3+d(11, 10) = 4
d4 = 2+d(11, 01) = 3

d: d4 = 3+d(11, 01) = 4
d4 = 2+d(11, 10) = 3

CC: DECODER

- **Viterbi algorithm: example: survival path**



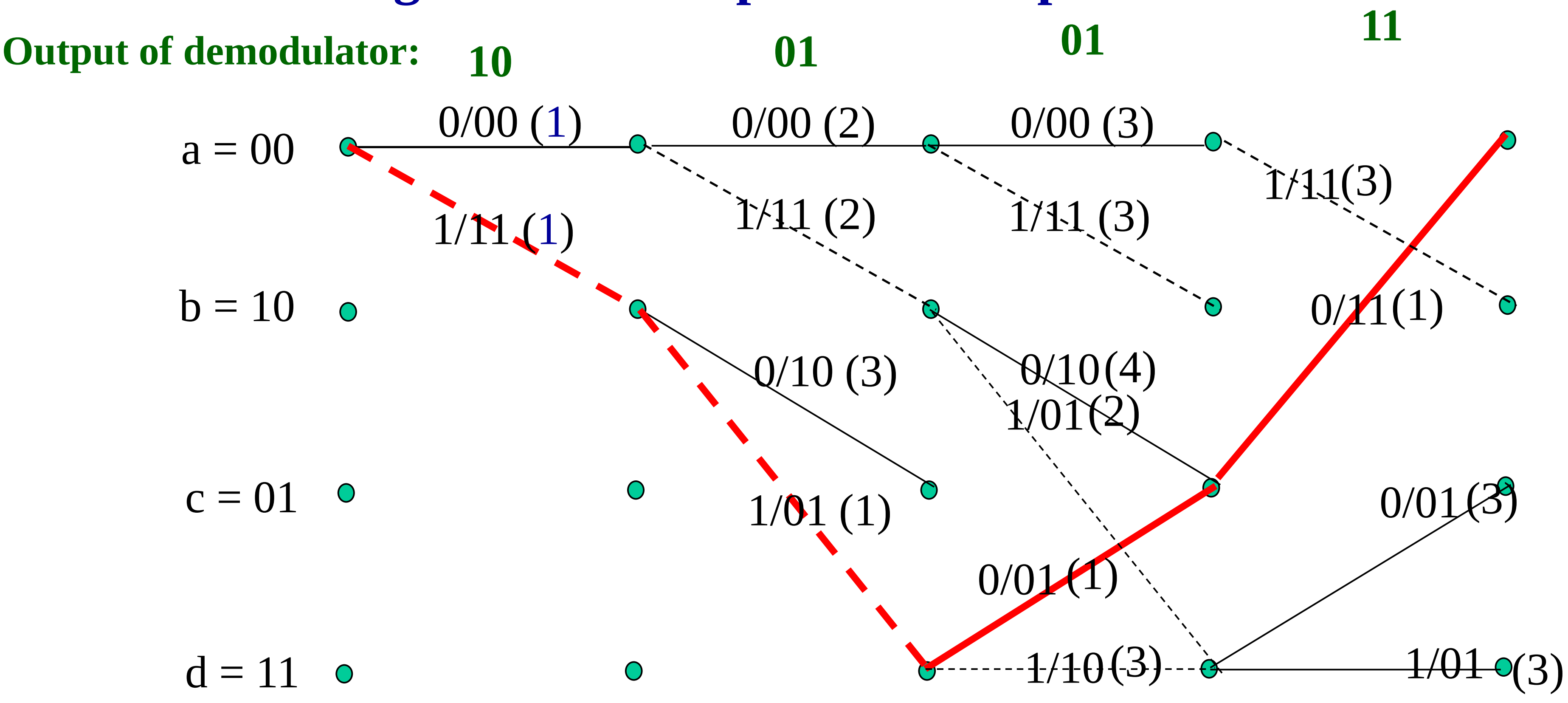
a: d4 = 3+d(11, 00) = 5 **b: d4 = 3+d(11, 11) = 3** **c: d4 = 3+d(11, 10) = 4** **d: d4 = 3+d(11, 01) = 4**

d4 = 1+d(11, 11) = 1 **d4 = 1+d(11, 00) = 3** **d4 = 3+d(11, 01) = 4** **d4 = 3+d(11, 10) = 4**

UNIVERSITY OF ARKANSAS

CC: DECODER

- **Viterbi algorithm: example: survival path**



Output of decoder: 1 1 0 0

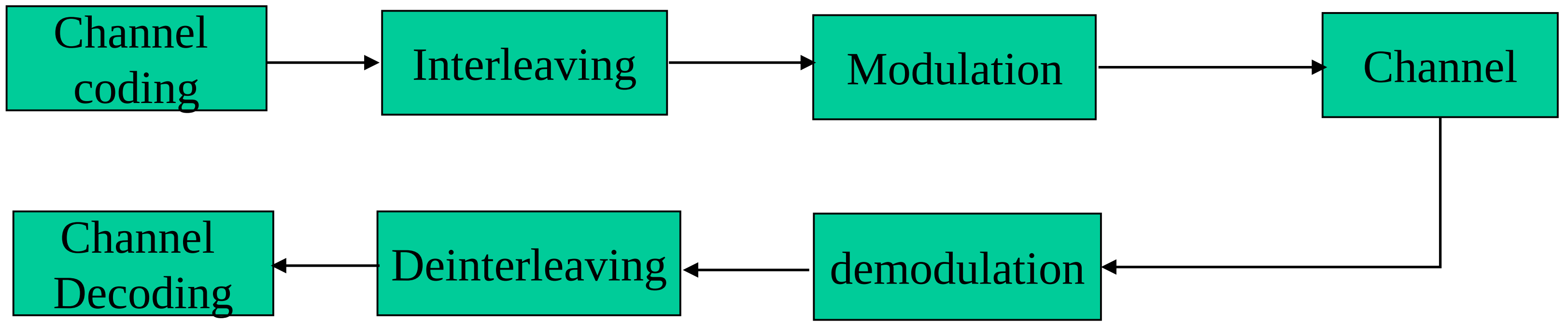
OUTLINE

- Introduction
- Source Coding
- Channel Coding: Linear Block Code
- Channel Coding: Convolutional Code
- **Interleaving**

INTERLEAVING

- **Motivation**
  - In fading channel, error usually occurs in burst:
    - a sequence of consecutive error bits.
  - Some channel coding scheme can only correct a few bits of error in one block
    - E.g. (5, 2) linear block code can correct only 1 bit error in one block.
  - What if the number of error bits in one block exceeds the correction ability of the channel code?
  - Solution: interleaving!
- **Interleaving**
  - Spread out bits of one codeword in time such they are undergoing independent fading.
  - Two kinds of interleaver:
    - block interleaving
    - Convolutional interleaving.

INTERLEAVING



- **Block interleaving**
  - Spread the encoded data into a rectangular matrix of *m* row and *n* columns.
  - The matrix is filled row-wise
    - $b_1 \to (1,1), b_2 \to (1,2), \cdots, b_n \to (1,n)$
      $b_{n+1} \to (2,1), \cdots\cdots\cdots, b_{2n} \to (2,n)$
  - The matrix is readout column-wise

INTERLEAVING

- **Block interleaving**
  - E.g. before interleaving: [b1, b2, b3, b4, b5, …, b11, b12]

b1	b2	b3	b4
b5	b6	b7	b8
b9	b10	b11	b12

  - After interleaving
    - [b1, b5, b9, b2, b6, b10, b3, b7, b11, b4, b8, b12]
    - The bits are transmitted in this order
    - One codeword is spread out in time
- **Block deinterleaving**
  - At the receiver, the deinterlever performs the reverse operation
    - Fill the matrix with the received bits column wise
    - Read out the matrix row wise.

INTERLEAVING

- **How could interleaving help?**
  - E.g. if 3-bit burst error happens during transmission
    - [b1, b5, b9, b2, b6, b10, b3, b7, b11, b4, b8, b12]
  - After deinterleaving
    - [b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12]
      - the 3-bit error is spread out to three codewords!
  - The system can correct 3-bit error !

b1	b2	b3	b4
b5	b6	b7	b8
b9	b10	b11	b12

INTERLEAVING

- **Block interleaving**
  - The burst of *m* consecutive errors results in isolated 1-bit error at the output of the deinterleaver
  - The number of rows *m* is called the interleaver depth
    - The correction capability of channel coding is multiplied by *m* times!
  - Cost:
    - longer delay:
      - decoding cannot be performed until all *mn* bits are received. (delay improved by a factor of *m*)
      - Tradeoff between power efficiency and delay
    - Larger memory
      - *mn* bits
      - (without interleaver: n bits)

SUMMARY

- **Source coding**
  - Why? Convert analog information into digital; reduce redundancy.
  - Lossless source code, source code with loss.
  - PCM, Entropy, speech coding (waveform, vocoder)
- **Channel coding**
  - Add redundancy to protect information (error correction, error detection) ➔ tradeoff between bandwidth efficiency and power efficiency
  - Channel capacity
- **Linear block code**
  - Generation matrix, parity check matrix, syndrome decoding
  - Coding rate, Hamming distance, Minimum Hamming distance
- **Convolutional code**
  - Four representations (shift register, input-output table, state transition, trellis)
  - Decoding: Viterbi algorithm
- **Interleaving**
  - Spread out the bits of one codeword in time
  - Block interleaver.
  - Tradeoff between power efficiency and processing delay.