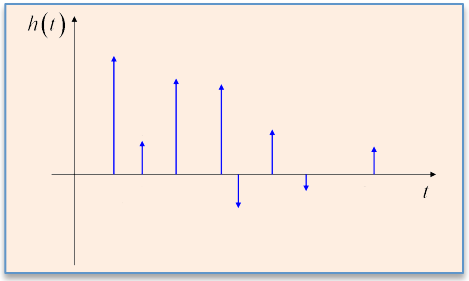
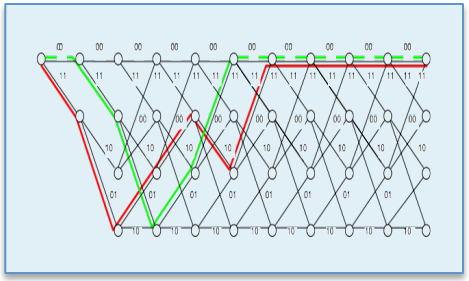
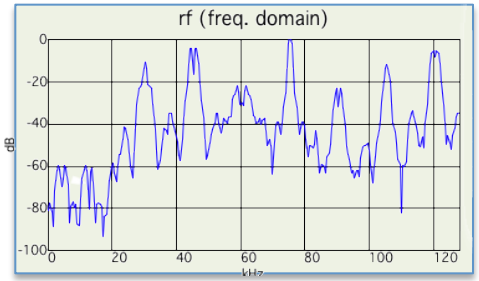
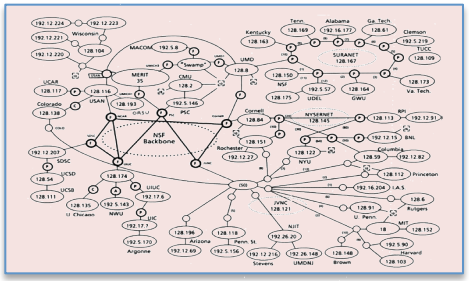








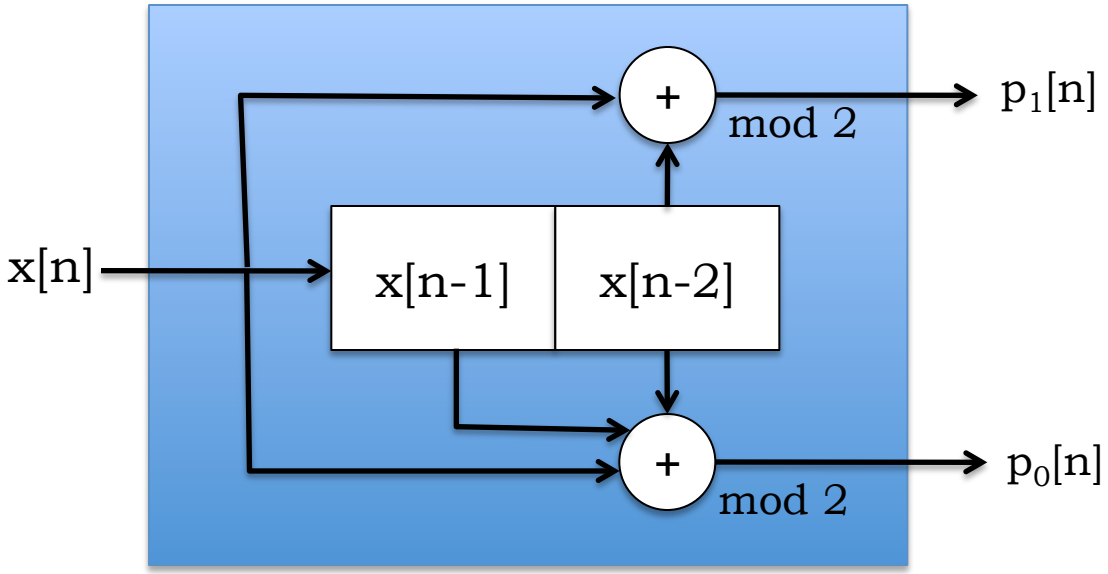
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2013
Lecture #7

- Viterbi decoding of convolutional codes

Convolutional Coding – Shift Register View

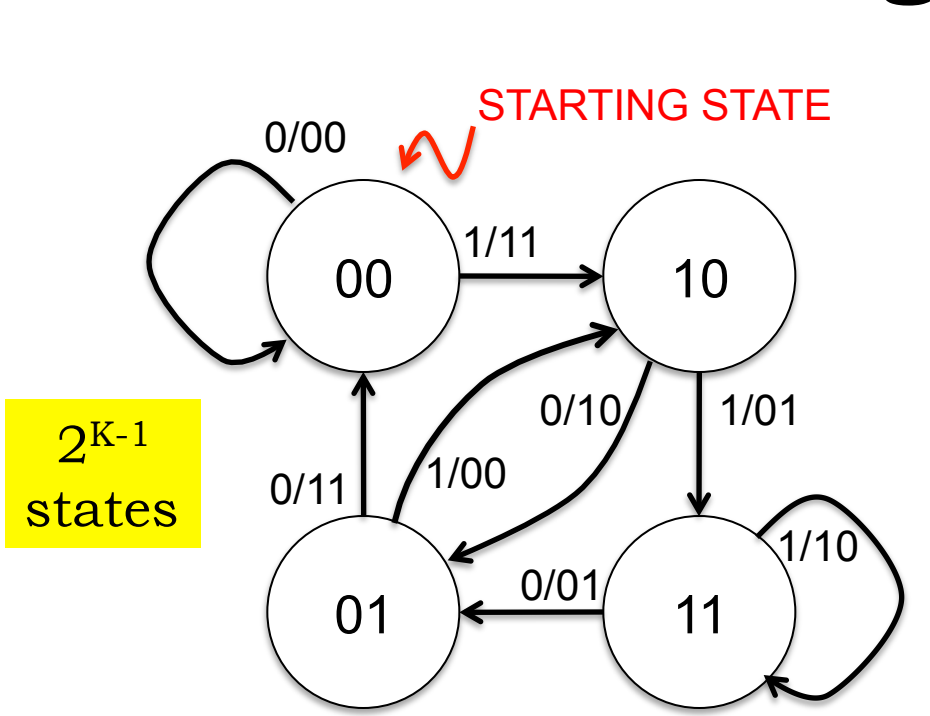


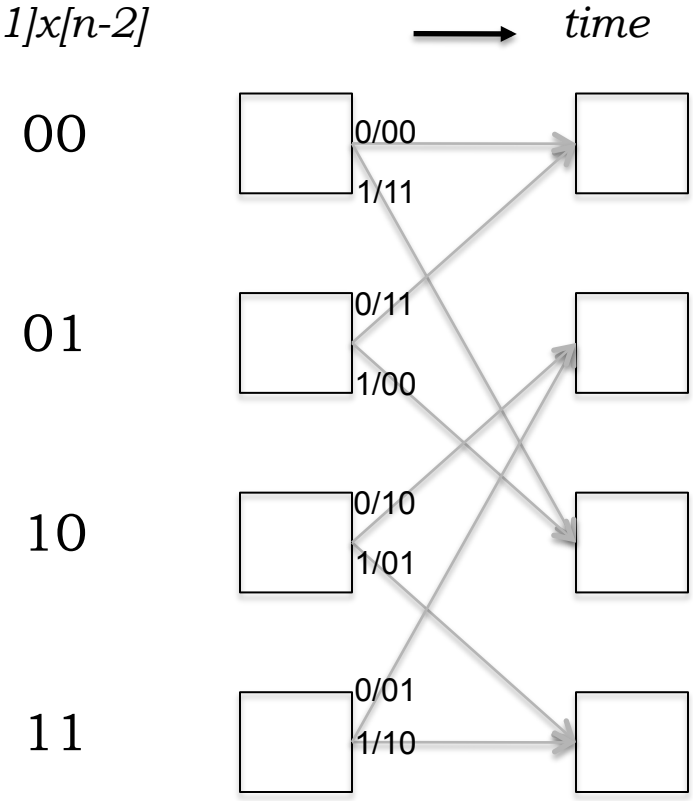
*The values in the registers define the **state** of the encoder*

Message bit in, parity bits out

- Input bits arrive one-at-a-time from the left
- The parity bits are computed from the incoming and *K*-1 previous message bits
- At the end of the bit interval, the saved message bits are *shifted right* by one, the oldest bit drops out, and the incoming bit moves into the left position

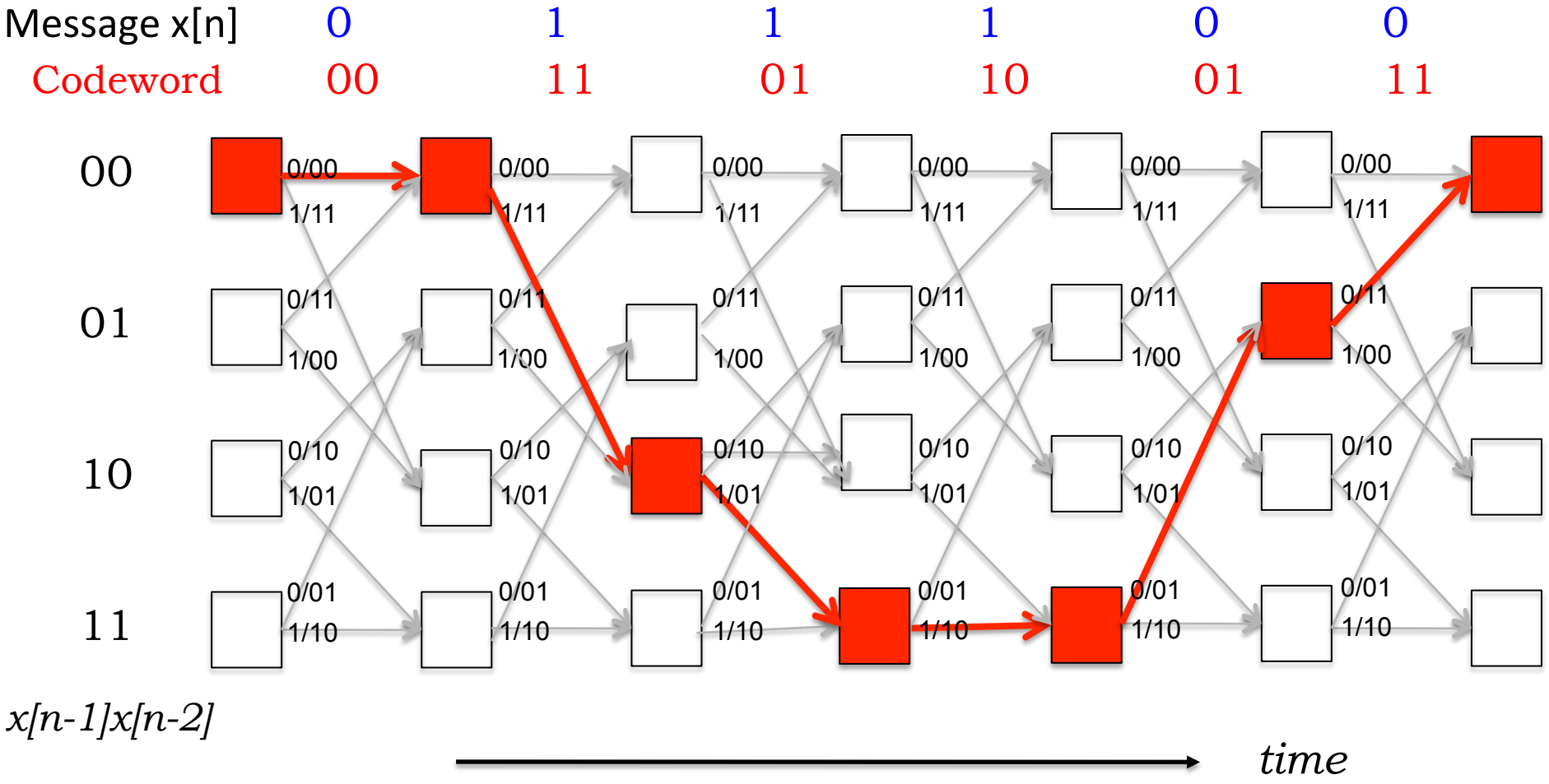
State-Transition Diagram → Trellis





- Example: *K*=3, rate-½ convolutional code
  - g_0 = 111: $p_0[n]$ = 1*x[n] + 1*x[n-1] + 1*x[n-2]
  - g_1 = 101: $p_1[n]$ = 1*x[n] + 0*x[n-1] + 1*x[n-2]
- States labeled with *x[n-1] x[n-2]*
- Arcs labeled with *x[n] / p_0 p_1* or just *p_0 p_1*

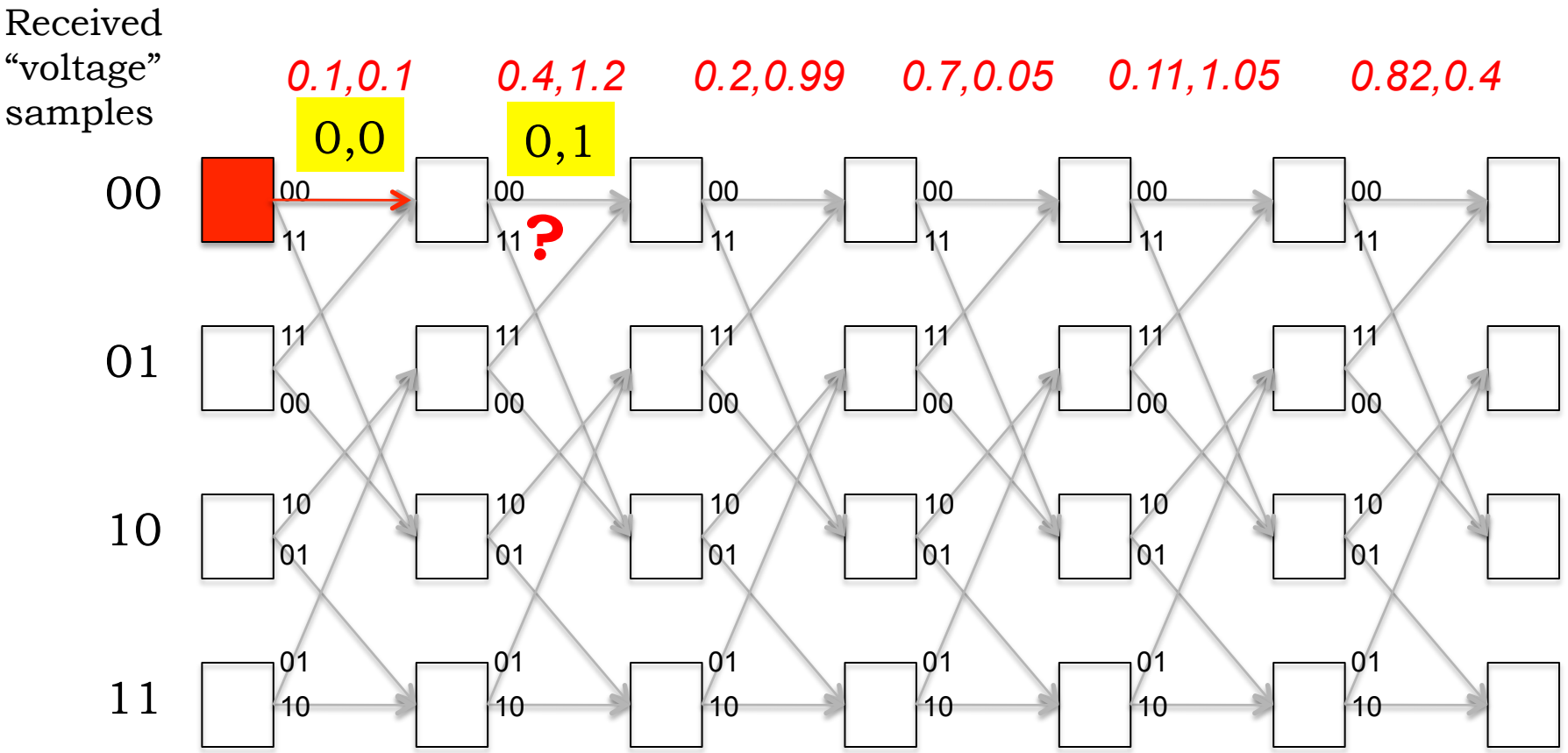
Trellis View at Transmitter



Generates a LINEAR code!

Minimum weight of nonzero word = 5

Decoding: Finding Max Likelihood Path



Given received "voltage" samples, find most-likely path through trellis, i.e., minimize "distance" between the received values and those produced along the trellis path.

"Hard decoding"

Simple-minded Decoding

- Enumerate all paths that start at the zero state, find the one that corresponds to a parity bit sequence that's closest in Hamming distance to the received (hard-decoded) sequence.

With L message bits, there are 2^L possible paths --- intractable for the kinds of large L that one would like to use (100's to 1000's).

The Viterbi  Algorithm

"The Viterbi algorithm (VA) is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite-state Markov process observed in memoryless noise. Many problems in areas such as digital communications can be cast in this form. This paper gives a tutorial exposition of the algorithm and of how it is implemented and analyzed. Applications to date are reviewed. Increasing use of the algorithm in a widening variety of areas is foreseen."

(Abstract of Forney's paper, Proc. IEEE, 1973)

Applications today:

"The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, speech synthesis, keyword spotting, computational linguistics, and bioinformatics. For example, in speech-to-text (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal."

(http://en.wikipedia.org/wiki/Viterbi_algorithm)

Viterbi Algorithm

- Dynamic programming algorithm, computes most likely message sequence leading up to every intermediate state, & associated cost

- Branch metric: BM(xmit,rcvd) for each branch of the trellis
  - proportional to negative log likelihood, i.e. negative log probability that we receive *rcvd*, given that *xmit* was sent
  - "Hard decision": use digitized bits, compute Hamming distance between xmit and rcvd (assumes $p < ½$ on BSC channel)
  - "Soft decision": use function of received voltages directly (e.g., sum of squares for iid Gaussian noise)

- Path metric: PM[s,i] for each state *s* of the 2^{K-1} transmitter states and bit time *i* , where $0 \le i < L-1$
  - PM[s,i] = smallest sum of BM(xmit, rcvd), minimized over all message sequences that place transmitter in state *s* at time *i*
  - PM[s,i+1] computed from PM[s,i] and the BM for outgoing branches at s,i

Hard Decisions

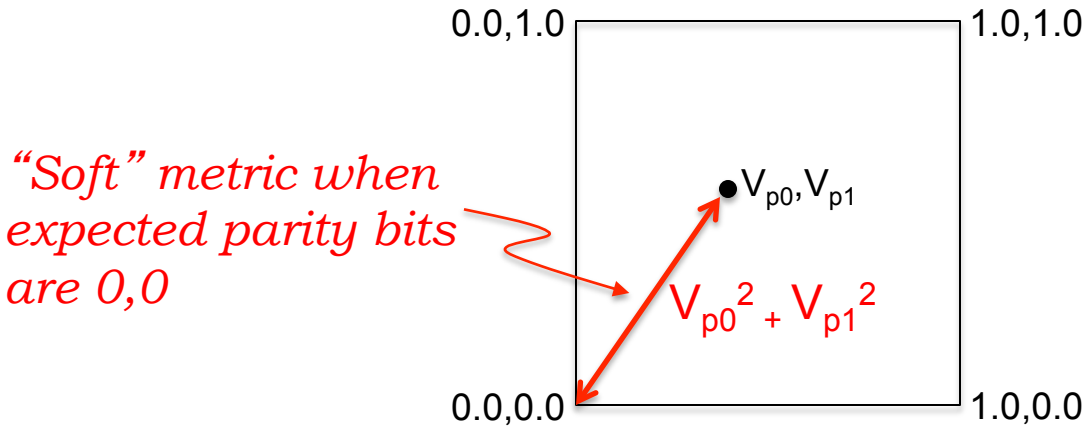
- As we receive each bit it is immediately digitized to "0" or "1" by comparing it against a threshold voltage
  - We lose the information about how "good" the bit is: a "1" at .9999V is treated the same as a "1" at .5001V

- The branch metric used in the Viterbi decoder under hard-decision decoding is the Hamming distance between the digitized received voltages and the expected parity bits

- Throwing away information is (almost) never a good idea when making decisions
  - Can we come up with a better branch metric that uses more information about the received voltages?

Soft-Decision Decoding

- In practice, the receiver gets a voltage level, V, for each received parity bit
  - Sender sends V0 or V1 volts; V in (-∞,∞) assuming additive Gaussian noise
- Idea: Pass received voltages to decoder **before** digitizing
- Define a "soft" branch metric as the square of the Euclidian distance between received voltages and expected voltages



- Soft-decision decoder chooses path that minimizes sum of the squares of the Euclidean distances between received and expected voltages
- Different BM & PM values, but otherwise the same algorithm

Complexity of VA

- At any given time there are 2^{K-1} most-likely messages we're tracking

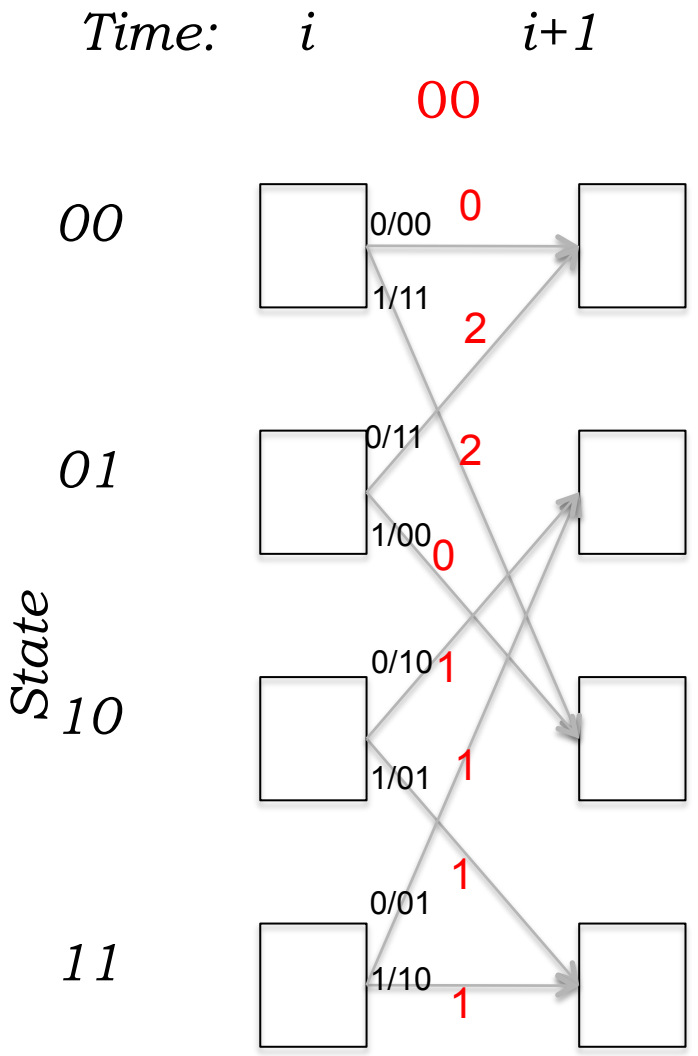
→ time complexity of algorithm grows exponentially with constraint length K, but only linearly with message length L

(as opposed to exponentially in L for simple enumeration)

(We saw K=15, for example, on Cassini spaceprobe)

Hard-decision Branch Metric

- BM = Hamming distance between expected parity bits and received parity bits
- Compute BM for each transition arc in trellis
  - Example: received parity = 00
  - BM(00,00) = 0
    BM(01,00) = 1
    BM(10,00) = 1
    BM(11,00) = 2
- Will be used in computing PM[s,i+1] from PM[s,i].



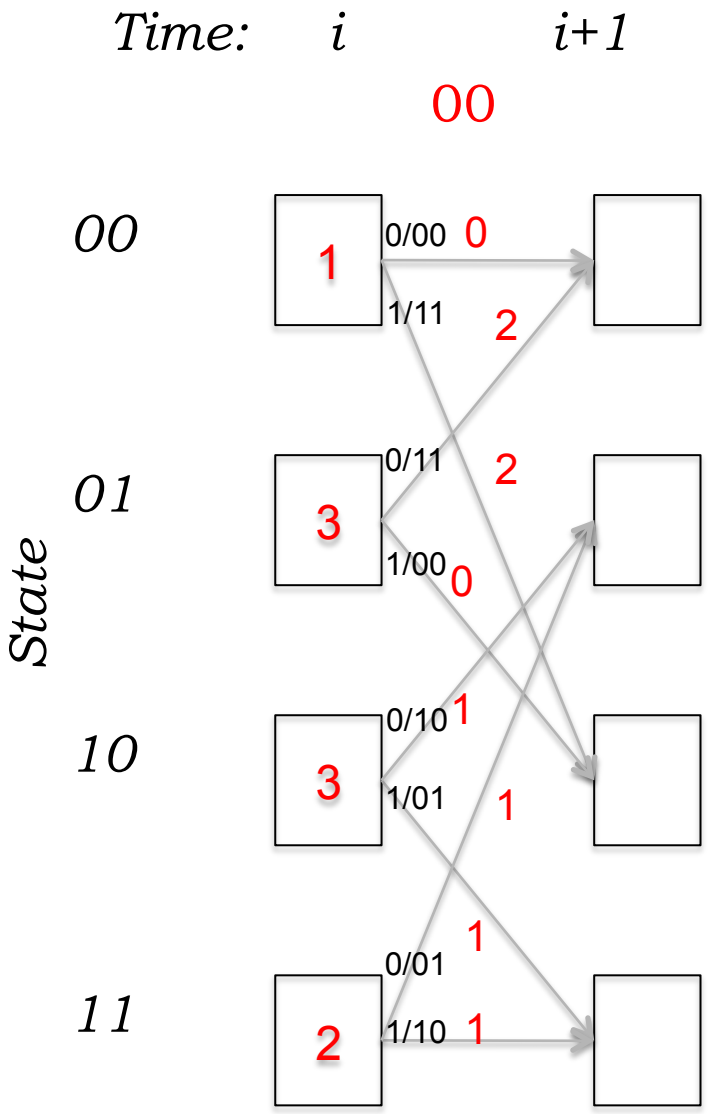
Computing PM[s,i+1]

Starting point: All PM[s,i] known, label in trellis box for each state at time i.

Example: PM[00,i] = 1 means 1 bit error detected when comparing received parity bits to what would have been transmitted when sending the most likely message, considering all messages that place the transmitter in state 00 at time i.

Q: What's the most likely state s given measurement up to time i?

A: state 00 (smallest PM[s,i])



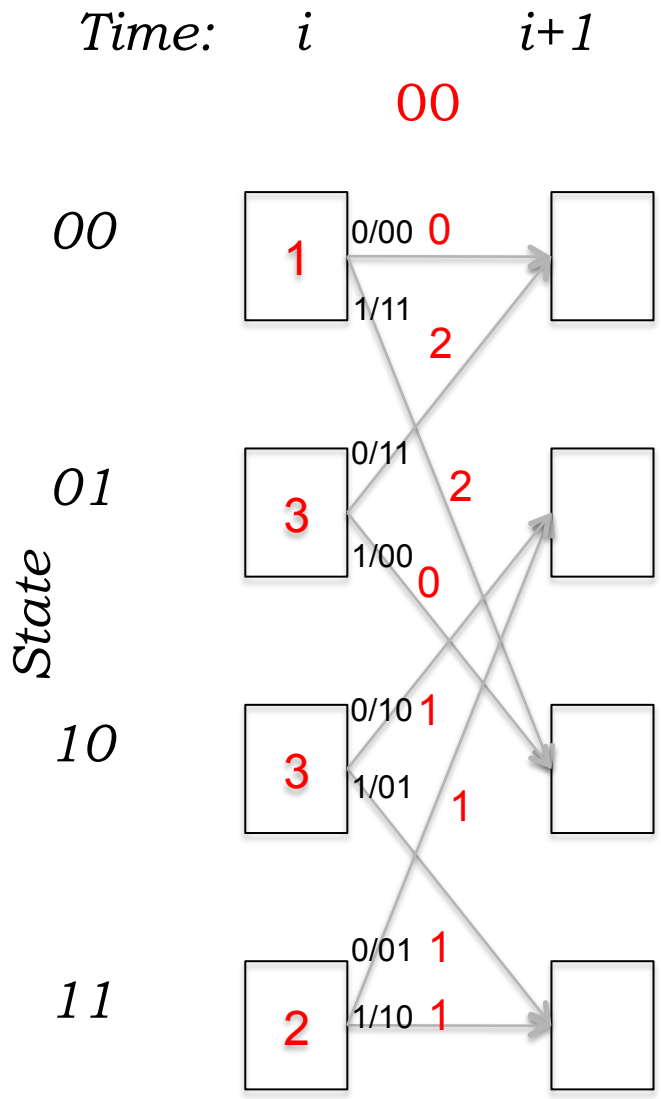
Computing PM[s,i+1] cont'd.

Q: If the trellis is in state s at time i+1, what states could it have been in at time i?

A: For each state s, there are two predecessor states α and β in the trellis diagram

Example: for state 01, α=10 and β=11.

Any message sequence that leaves the trellis in state s at time i+1 must have left the trellis in state α or state β at time i.



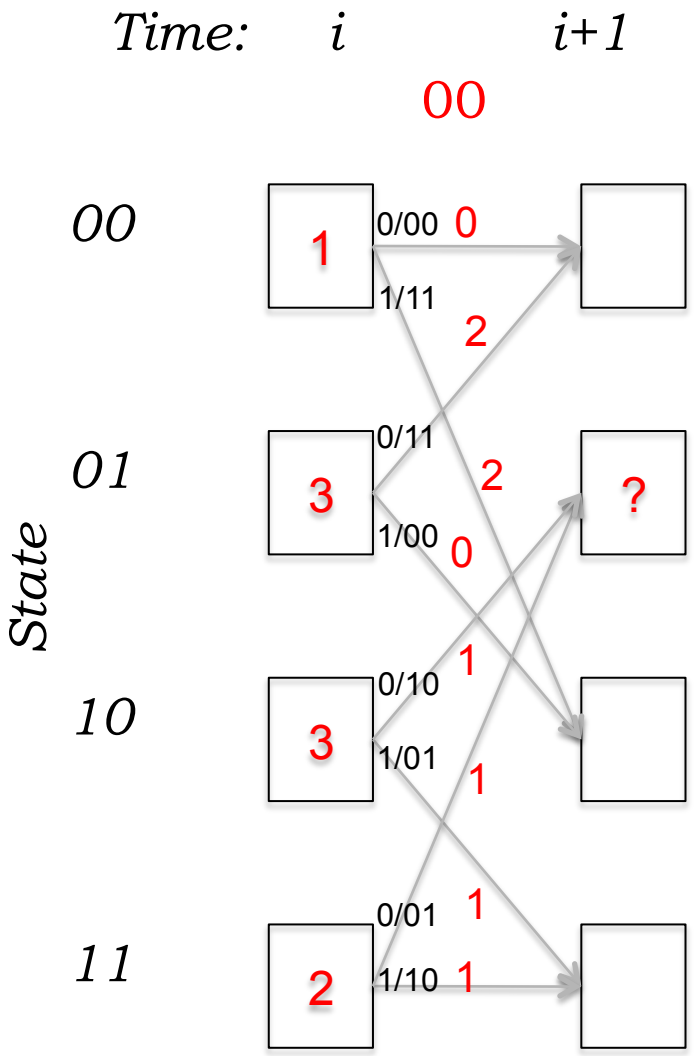
Computing PM[s,i+1] cont'd.

e.g., Which is the more likely path into state 01 at time i+1?

PM[01,i+1]

= min{PM[10,i] + 1 , PM[11,i] + 1}

= min(3+1 , 2+1) = 3



Computing PM[s,i+1] cont'd.

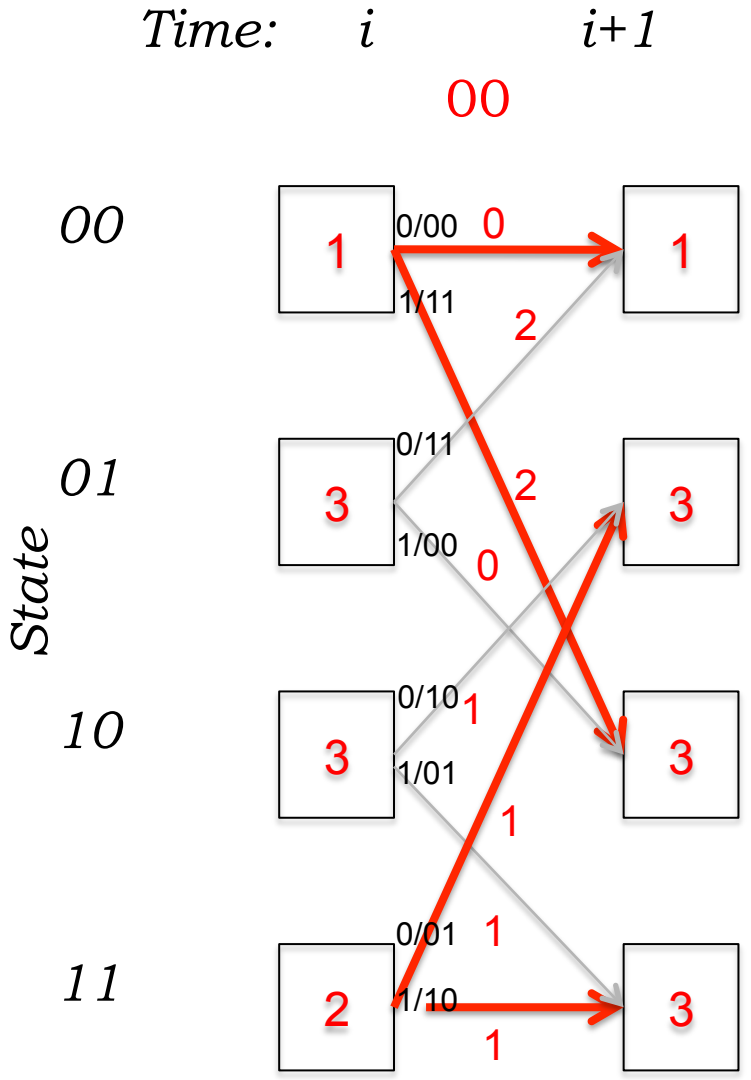
Formalizing the computation:

PM[s,i+1] = min(PM[α,i] + BM[α→s],
PM[β,i] + BM[β→s])

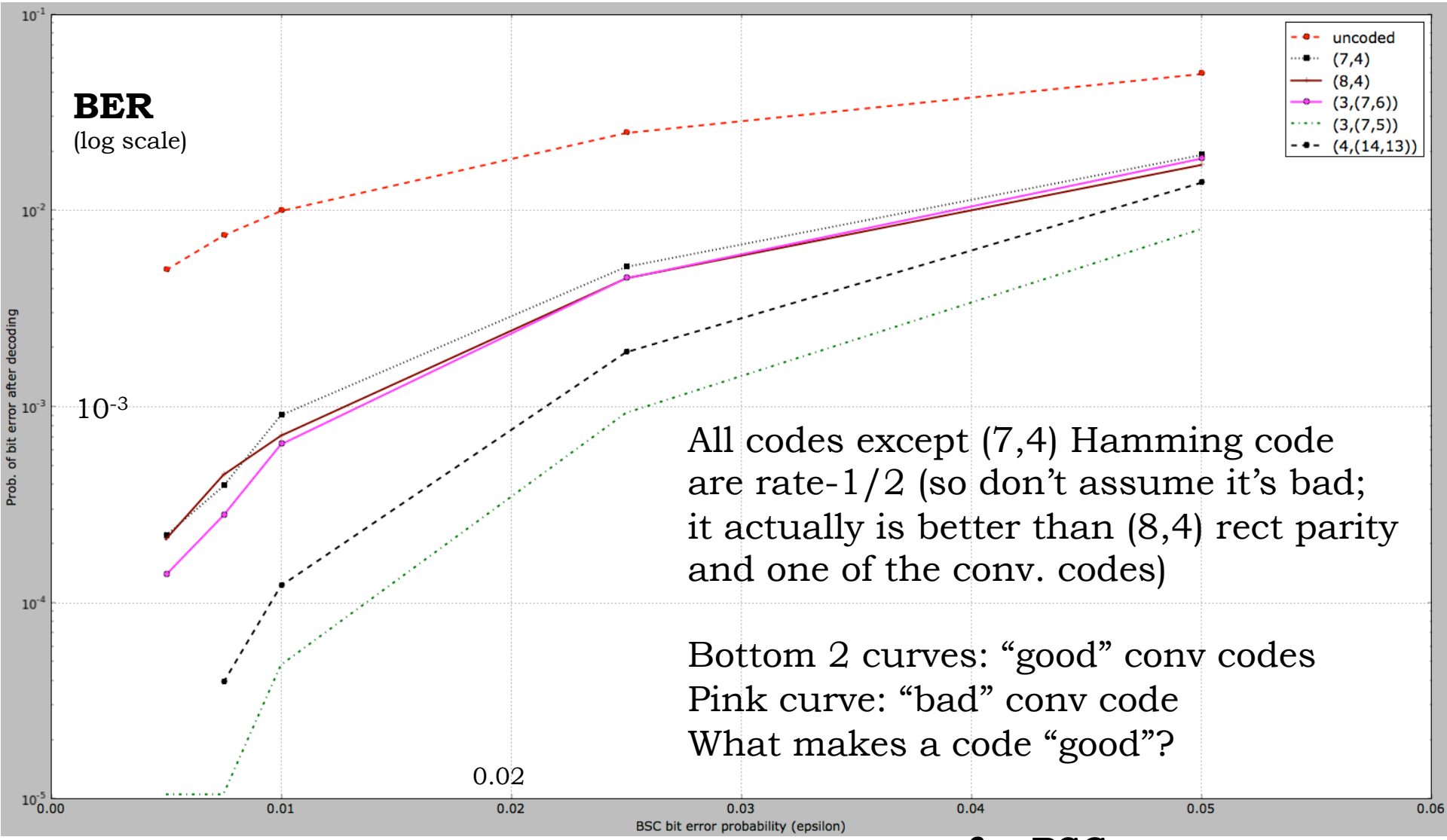
- Remember which arc was min;

saved arcs will generate path through trellis

- If both arcs have same sum, break tie arbitrarily (e.g., when computing PM[10,i+1])

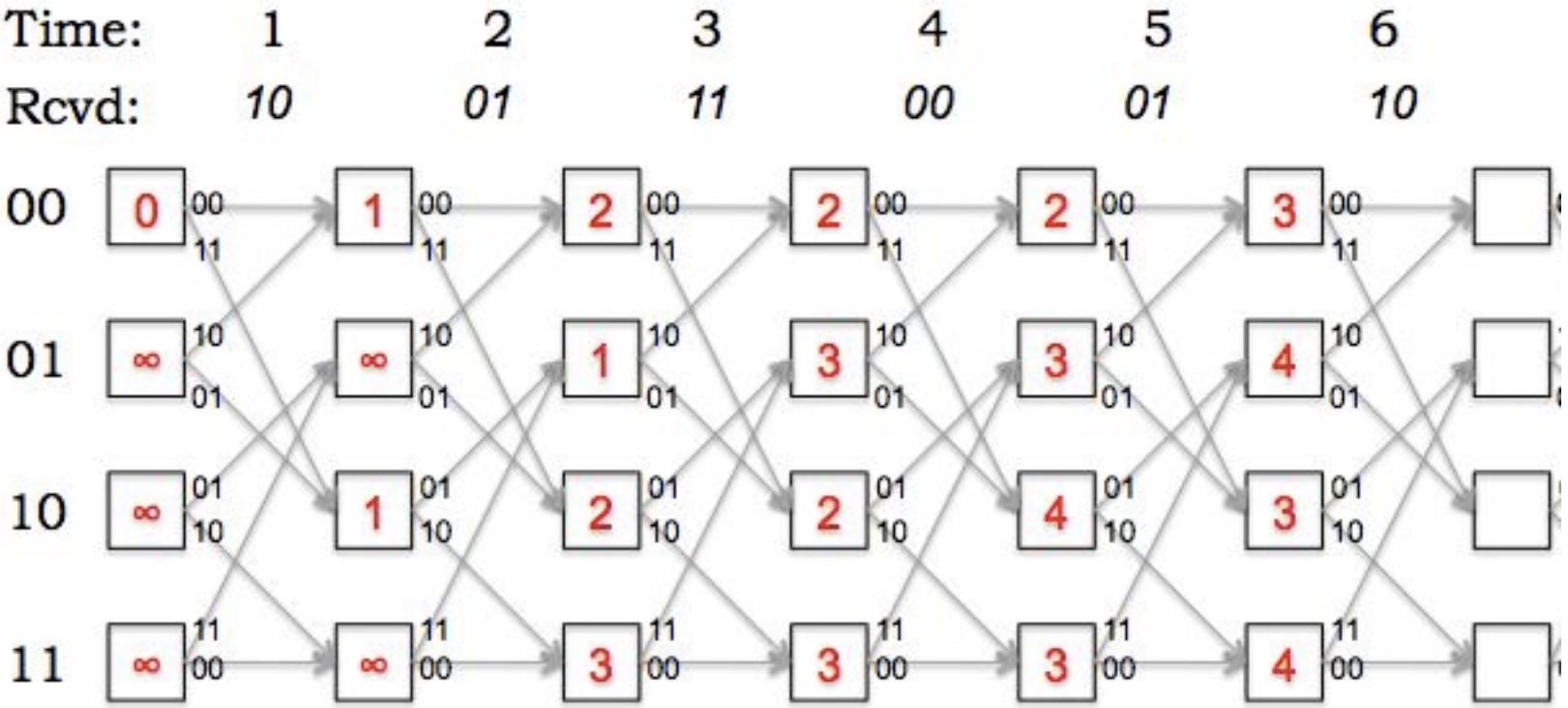


Post-decoding BER v. BSC error prob.



p for BSC (linear scale)

Spot Quiz Time…



1. What are the path metrics for the empty boxes?
2. What is the most-likely state after time step 6?
3. If the decoder had stopped after time step 2 and returned the most-likely message, what would the bits of the message be (careful about order!)?