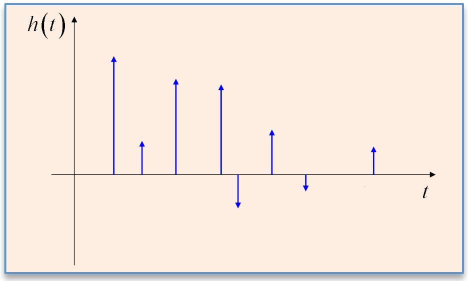
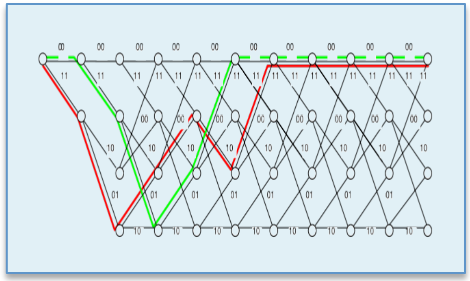
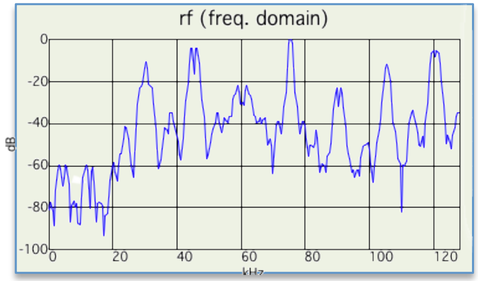
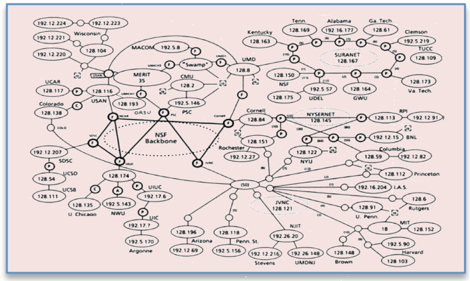








INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

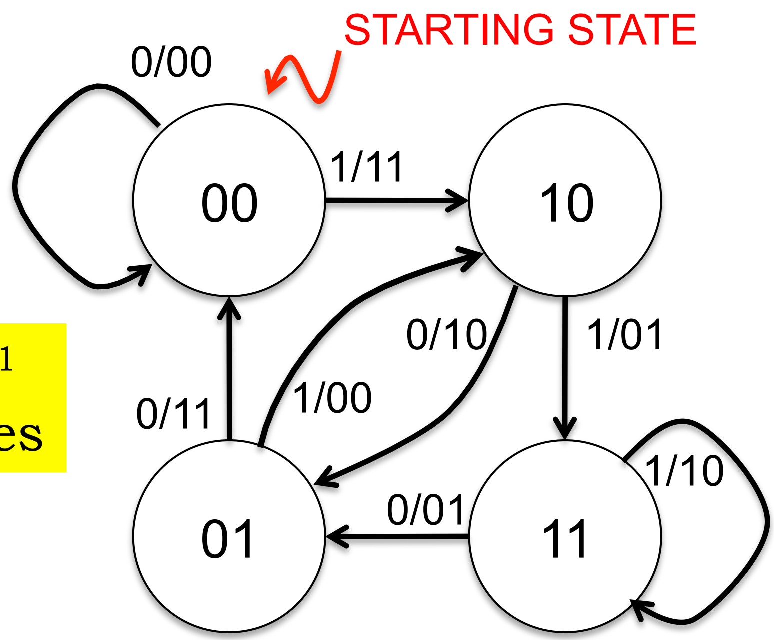
6.02 Fall 2012
Lecture #7

- Viterbi decoding of convolutional codes
  - Path and branch metrics
  - *Hard-decision* & *soft-decision* decoding
- Performance issues: decoder complexity, post-decoding BER, “free distance” concept

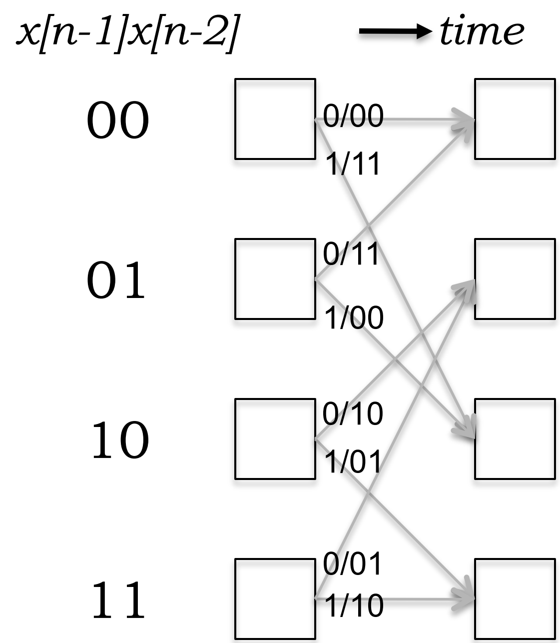
Convolutional Codes

- Coding review
- Decoding via Viterbi algorithm

Key Concept for Coding and Decoding: *Trellis*

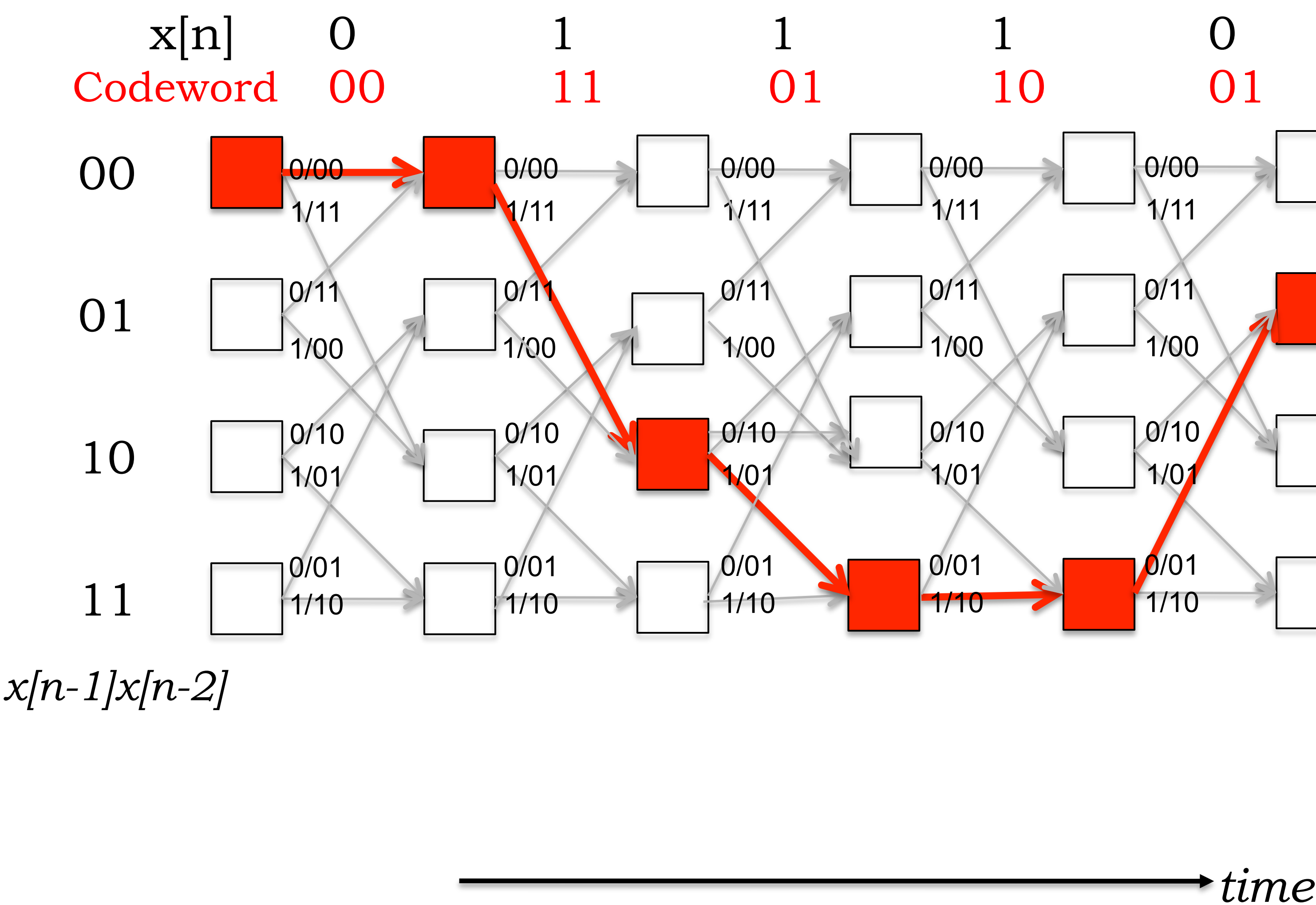


2^{K-1} states



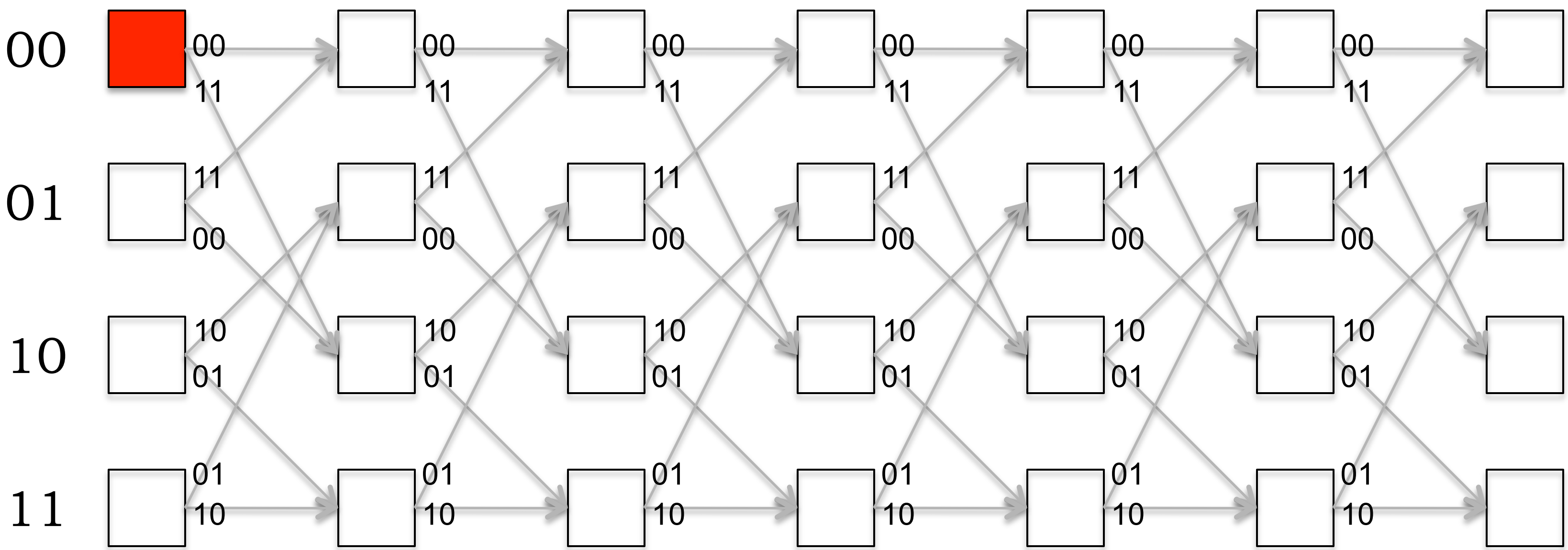
- Example: K=3, rate-½ convolutional code
  - $g_0 = 111$: $p_0[n] = 1*x[n] + 1*x[n-1] + 1*x[n-2]$
  - $g_1 = 101$: $p_1[n] = 1*x[n] + 0*x[n-1] + 1*x[n-2]$
- States labeled with *x[n-1] x[n-2]*
- Arcs labeled with $x[n]/p_0p_1$

Trellis View at Transmitter



Decoding:
Finding the Maximum-Likelihood (ML) Path

Rcvd: *0.1,0.1* *0.4,1.2* *0.2,0.99* *0.7,0.05* *0.11,1.05* *0.82,0.4*



Given the received voltages, the receiver must find the most-likely sequence of transmitter states, i.e., the path through the trellis that minimizes the "distance" between the received parity voltages and the voltages the transmitter would have sent had it followed that state sequence.

One solution: Viterbi decoding

Receiver

- For the code
  p0 = x[n]+x[n-1]+x[n-2]
  p1 = x[n]   +x[n-2]
- Received:
  000101100110
- Some errors have occurred…
- What's the 4-bit message?

Most likely: 0111

- i.e., message whose codeword is closest to rcvd bits

Msg	*Codeword*	*Received*	*Hamming distance*
0000	000000000000		5
0001	000000111011		-
0010	000011101100		-
0011	000011010111		-
0100	001110110000		-
0101	001110001011		-
0110	001101011100		-
0111	001101100111	**000101100110**	2
1000	111011000000		-
1001	111011111011		-
1010	111000101100		-
1011	111000010111		-
1100	110101110000		-
1101	110101001011		-
1110	110110011100		-
1111	110110100111		-

Initial and final state: 00

Viterbi Algorithm

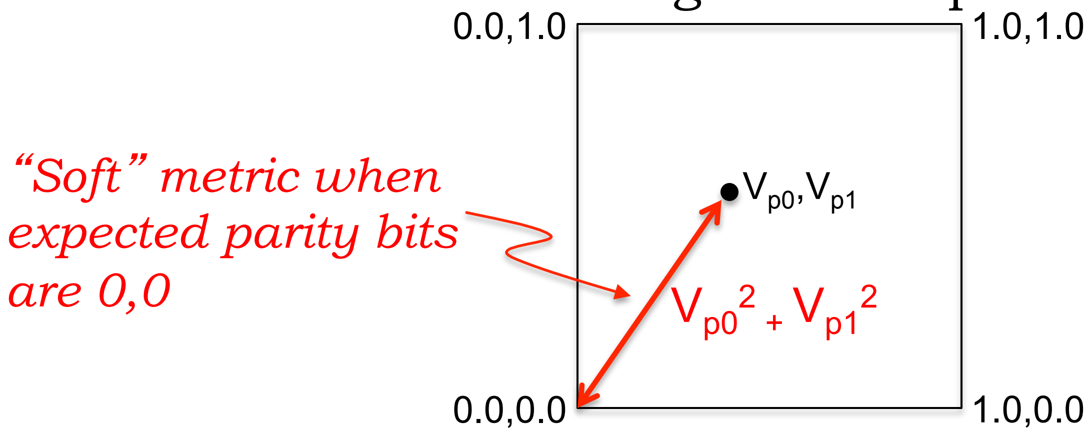
- Want: Most likely message sequence
- Have: (possibly corrupted) received parity sequences
- Viterbi algorithm for a given K and r:
  - Works incrementally to compute most likely message sequence
  - Uses two metrics
- Branch metric: BM(xmit,rcvd) proportional to negative log likelihood, i.e. negative log probability that we receive *rcvd*, given that *xmit* was sent.
  - "Hard decision": use digitized bits, compute Hamming distance between xmit and rcvd. Smaller distance is more likely if BER < 1/2
  - "Soft decision": use function of received voltages directly
- Path metric: PM[s,i] for each state *s* of the 2^{K-1} transmitter states and bit time *i* , where $0 \le i < L = \text{len(message)}$.
  - PM[s,i] = smallest sum of BM(xmit, rcvd) over all message sequences *m* that place transmitter in state *s* at time *i*
  - PM[s,i+1] computed from PM[s,i] and $p_0[i],\ldots,p_{r-1}[i]$

Hard Decisions

- As we receive each bit it's 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



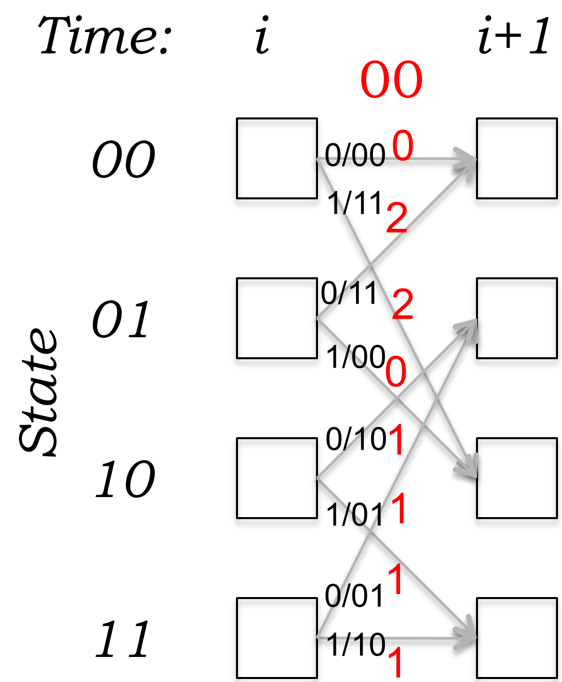
- Soft-decision decoder chooses path that minimizes sum of the squares of the Euclidean distances between received and expected voltages
  - Different BM & PM values, but otherwise the same algorithm

Viterbi Algorithm with Hard Decisions

- Branch metrics measure the contribution to negative log likelihood by comparing received parity bits to possible transmitted parity bits computed from possible messages.
- Path metric PM[s,i] proportional to negative log likelihood of transmitter being in state s at time *i*, assuming the mostly likely message of length i that leaves the transmitter in state s.
- Most likely message? The one that produces the smallest PM [s,N].
- 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 (as opposed to exponentially in message length for simple-minded enumeration).

Hard-decision Branch Metric

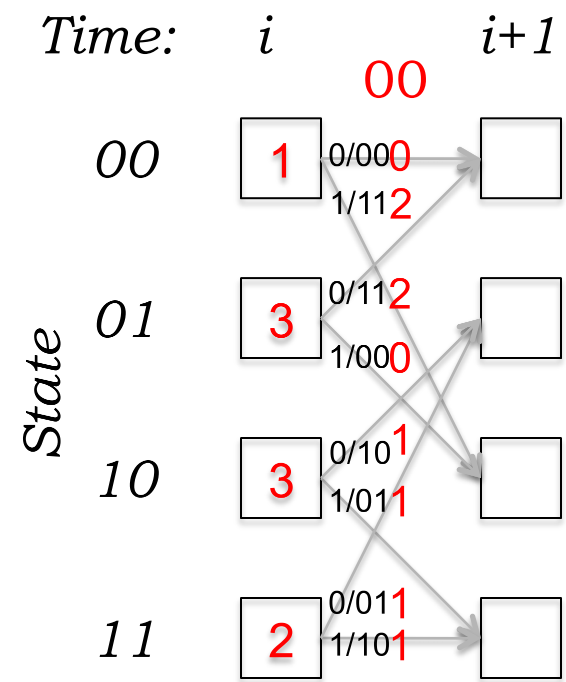
- BM = Hamming distance between expected parity bits and received parity bits
- Compute BM for each transition arc in trellis
  - Example: received parity = 00
  - BM(00,00) = 0
    BM(01,00) = 1
    BM(10,00) = 1
    BM(11,00) = 2
- Will be used in computing PM[s,i+1] from PM[s,i].



Computing PM[s,i+1]

Starting point: we've computed PM[s,i], shown graphically as label in trellis box for each state at time *i*.

Example: PM[00,*i*] = 1 means there was 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* for the transmitter at time *i*?

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

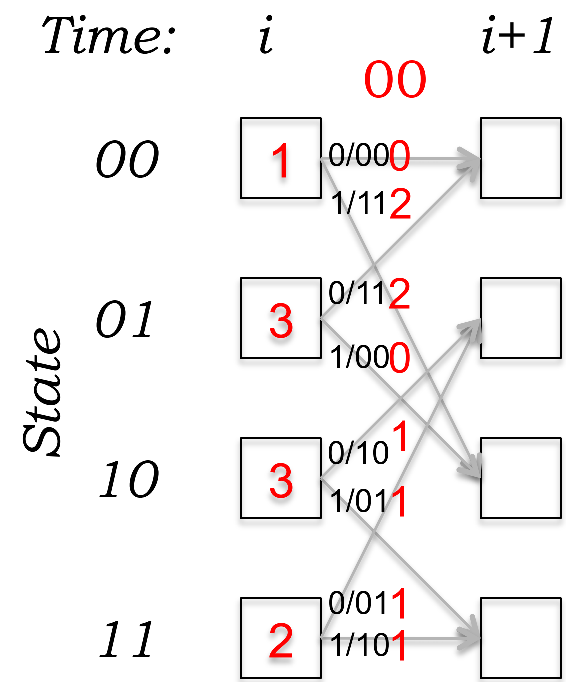
Computing PM[s,i+1] cont'd.

Q: If the transmitter is in state s at time i+1, what state(s) 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 transmitter in state s at time i+1 must have left the transmitter in state α or state β at time i.



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

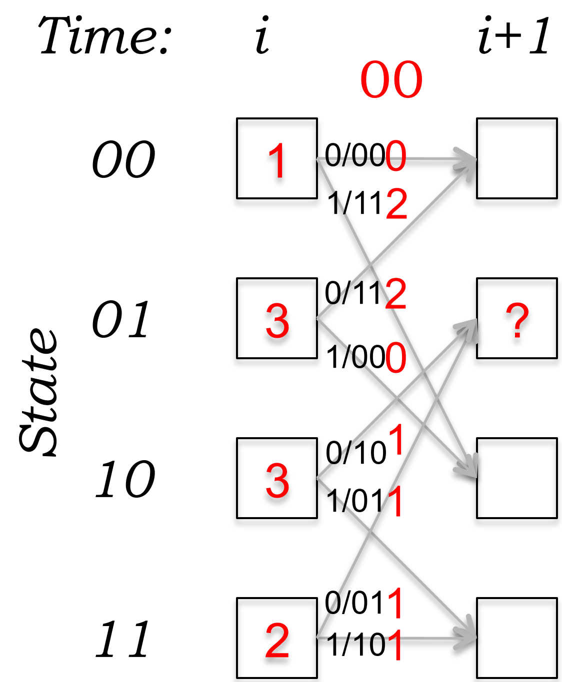
Example cont' d: to arrive in state 01 at time i+1, either

1)The transmitter was in state 10 at time i and the i^{th} message bit was a 0. If that' s the case, the transmitter sent 10 as the parity bits and there was 1 bit error since we received 00. Total bit errors = PM[10,i] + 1 = 4

OR

2)The transmitter was in state 11 at time i and the i^{th} message bit was a 0. If that' s the case, the transmitter sent 01 as the parity bits and there was 1 bit error since we received 00. Total bit errors = PM[11,i] + 1 = 3

Which is more likely?



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

Formalizing the computation:

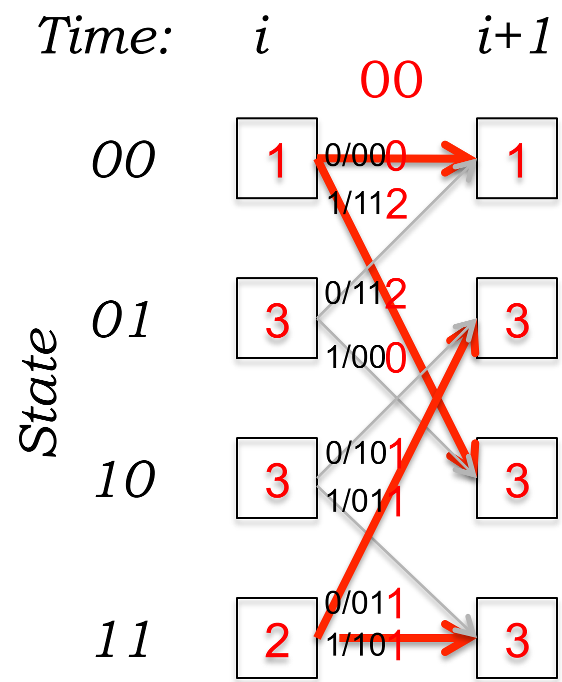
$$PM[s,i+1] = \min(PM[\alpha,i] + BM[\alpha \to s], PM[\beta,i] + BM[\beta \to s])$$

Example:

$$PM[01,i+1] = \min(PM[10,i] + 1, PM[11,i] + 1) = \min(3+1, 2+1) = 3$$

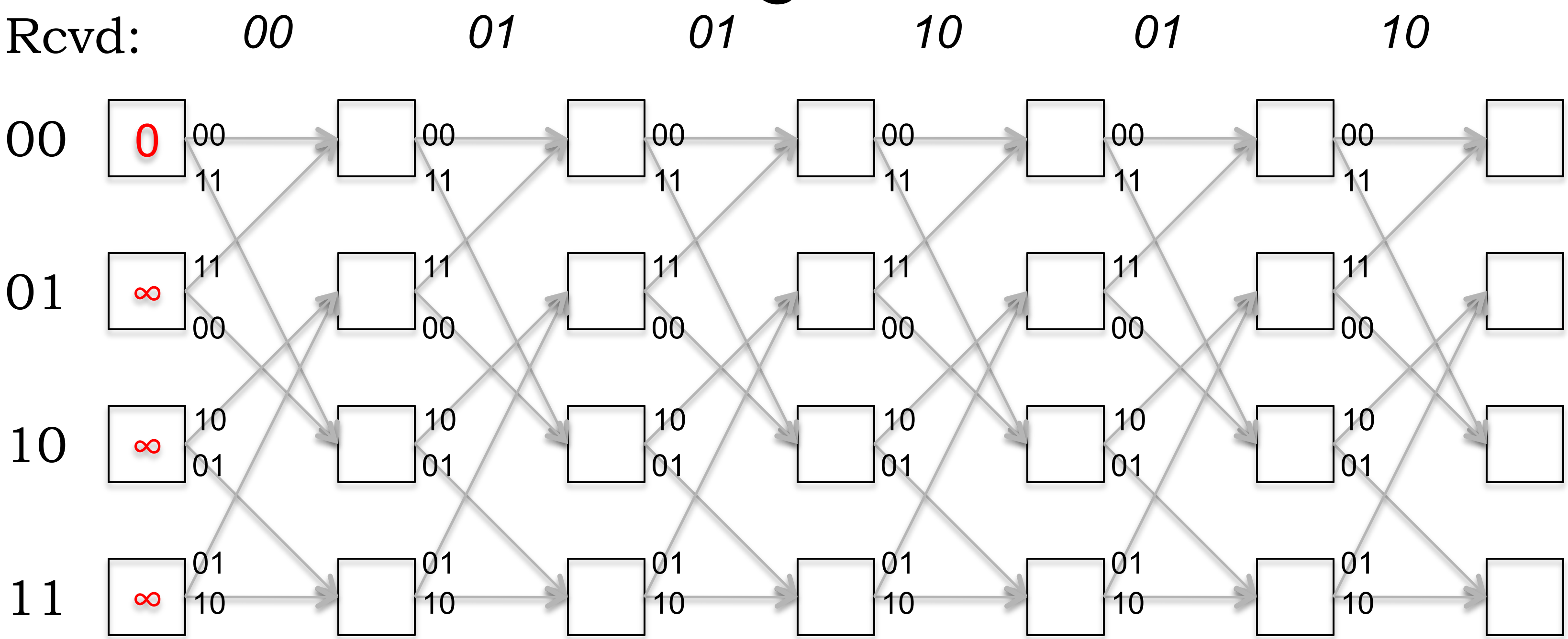
Notes:

1) Remember which arc was min; saved arcs will form a path through trellis
2) If both arcs have same sum, break tie arbitrarily (e.g., when computing PM[11,i+1])



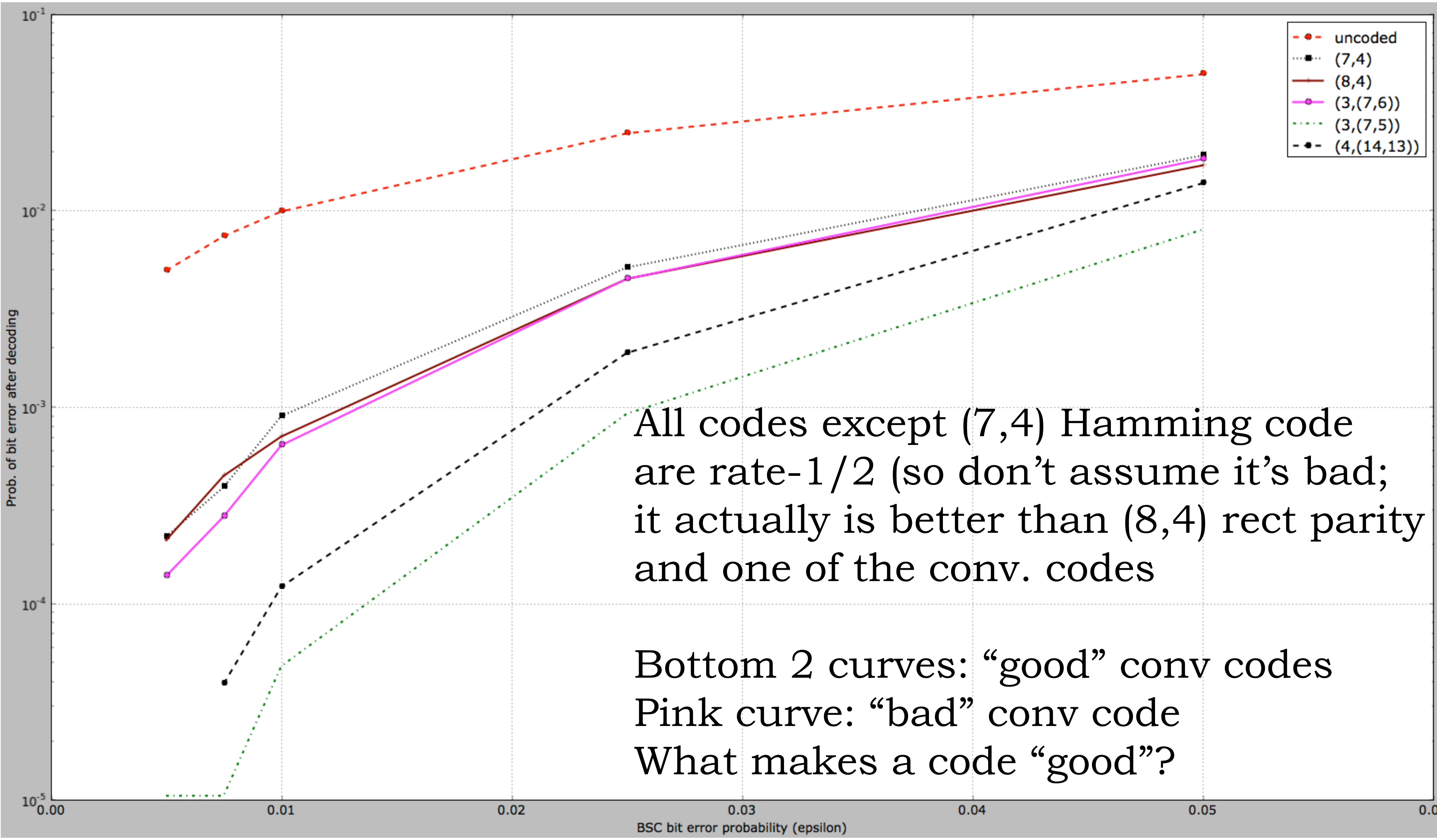
Hard-Decision Viterbi Decoding

A walk through the trellis



- Path metric: number of errors on maximum-likelihood path to given state (min of all paths leading to state)
- Branch metric: for each arrow, the Hamming distance between received parity and expected parity

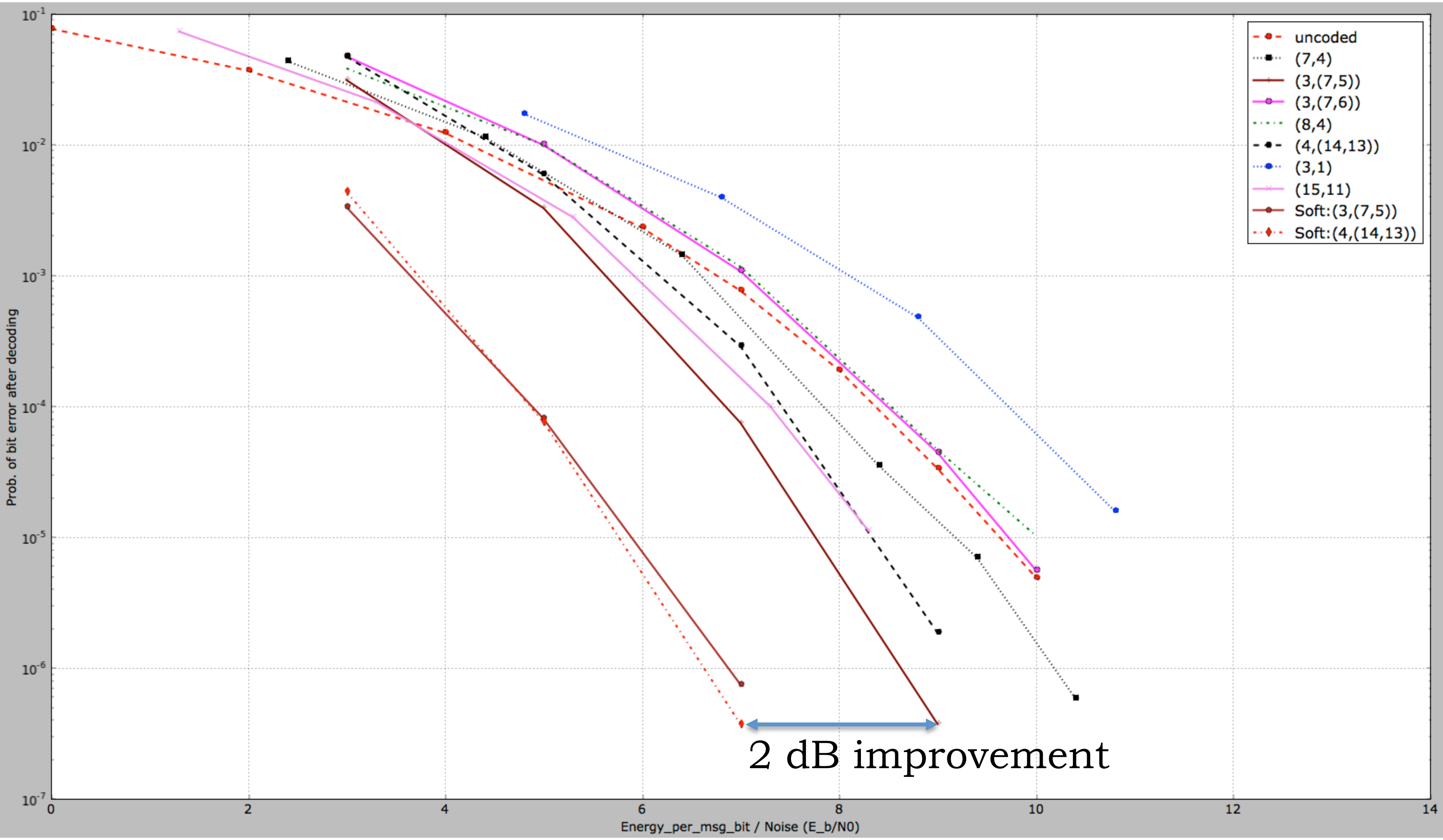
Post-decoding BER v. or BSC error prob.



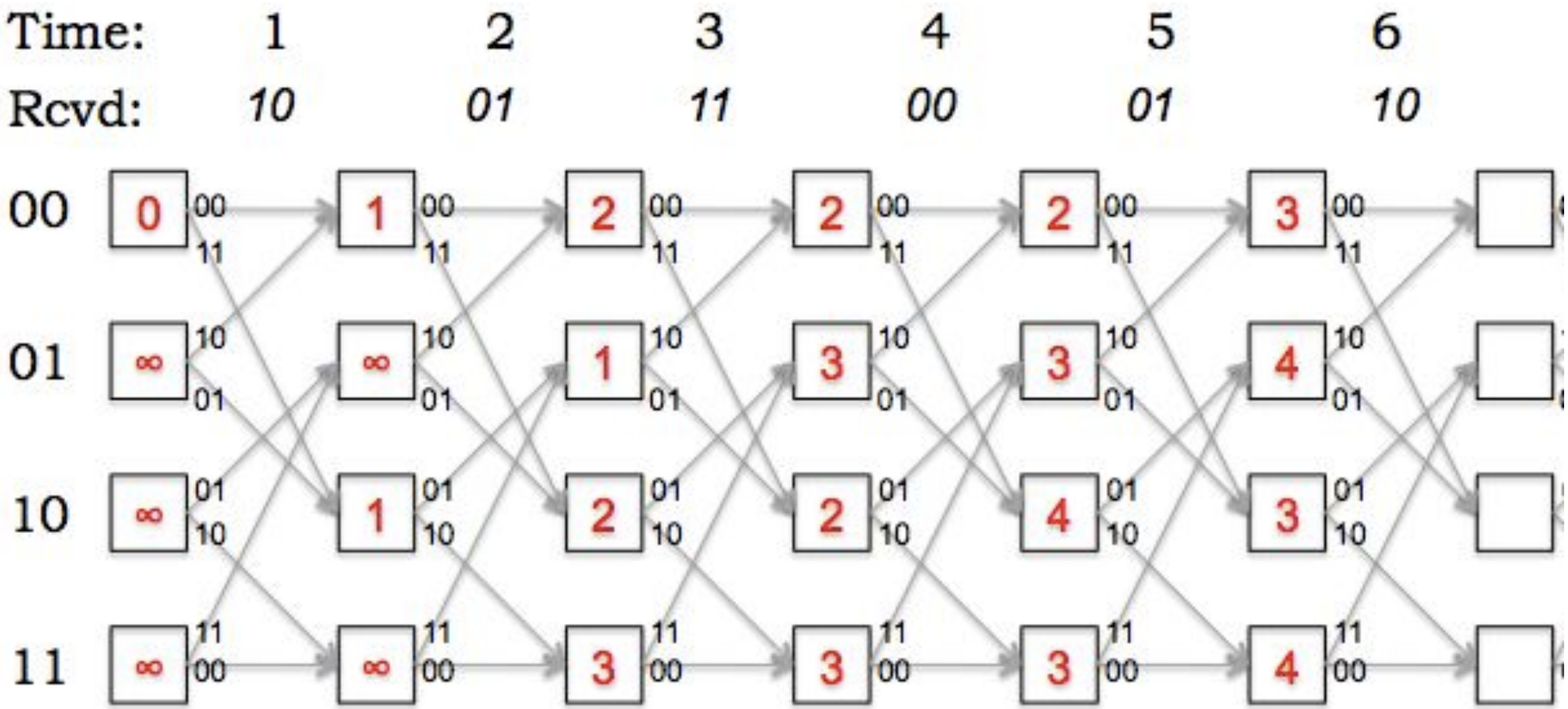
All codes except (7,4) Hamming code are rate-1/2 (so don't assume it's bad; it actually is better than (8,4) rect parity and one of the conv. codes

Bottom 2 curves: "good" conv codes
Pink curve: "bad" conv code
What makes a code "good"?

Soft Decoding Beats Hard Decoding



Spot Quiz Time…



1. What are the path metrics for the empty boxes (top to bottom order)?
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!)?