



# EE 5654 - Digital Communications Spring 2005

---

Instructor: R. Michael Buehrer  
Lecture #23 - Performance  
of Convolutional Codes





# Summary of Encoding and Decoding of Convolutional Codes

---

- Convolutional codes are encoded using a finite state machine.
- Optimal decoder for convolutional codes will find the path through the trellis which lies at the shortest distance to the received signal. (Maximum Likelihood sequence)
- Viterbi algorithm reduces the complexity of this search by finding the optimal path one stage at a time.
- The complexity of the Viterbi algorithm is proportional to the number of states
  - exponential relationship to constraint length



# Implementation of Viterbi Decoder

---

- Complexity is proportional to number of states  $2^{K-1}$ 
  - increases exponentially with constraint length  $K$ :
- Very well suited to parallel implementation
  - Each state has two transitions into it
  - Each state has two transitions out of it
  - Each node must compute two path metrics, add them to previous metric and compare
  - Much analysis as gone into optimizing implementations of this calculation



# Performance of Convolutional Codes

---

- When the decoder chooses a path through the trellis which diverges from the correct path, this is called an "error event"
- The probability that an error event begins during the current time interval is the "first-event error probability"

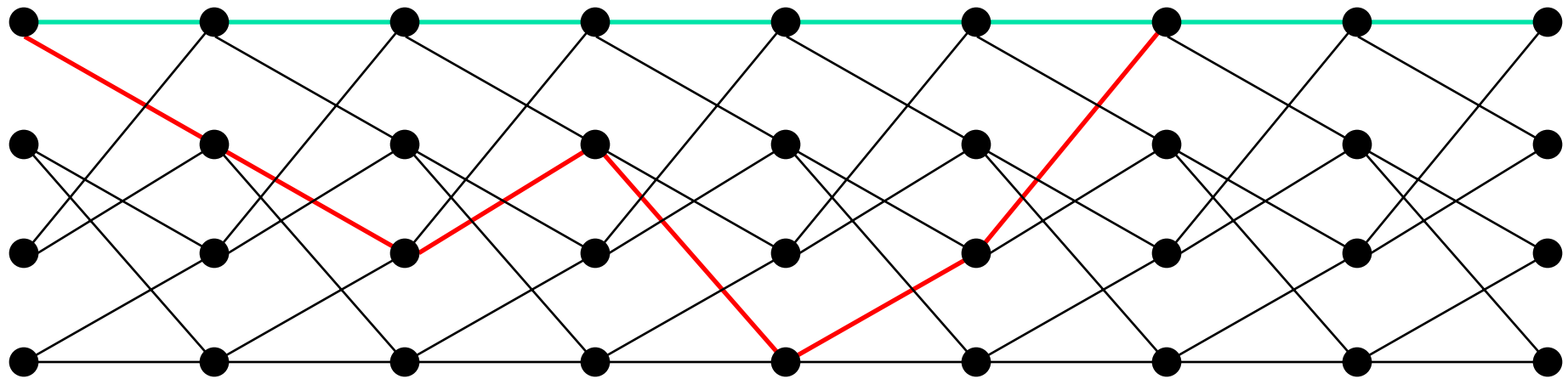
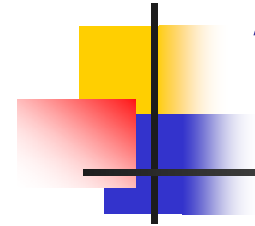
$$P_e$$

- The minimum Hamming distance separating any two distinct paths through the trellis is called the "free distance"

$$d_{\text{free}}$$

- Because of the linearity property of Convolutional codes, we can assume that any sequence was transmitted and the error probability will be identical. For the sake of convenience, we choose the all zero sequence to be the assumed transmitted sequence.

# An Error event



— Transmitted sequence  
— Decoded sequence

# Calculation of Error Event Probability

- What's the pairwise probability of choosing a path at distance  $d$  from the correct path?

$$P_2(d) = \begin{cases} \sum_{k=(d+1)/2}^d \binom{d}{k} p^k (1-p)^{d-k}, & d \text{ odd} \\ \sum_{k=\frac{d}{2}+1}^d \binom{d}{k} p^k (1-p)^{d-k} \\ \quad + \frac{1}{2} \binom{d}{d/2} p^{d/2} (1-p)^{d/2}, & d \text{ even} \end{cases}$$

Note that we are only interested in the bits where the two paths differ. Errors in the other paths will not have an impact on our decision

# Calculation of First Event Error Probability

- An upper (Union) bound:

$$P_e \leq \sum_{d=d_{\text{free}}}^{\infty} a_d \cdot P_2(d)$$

where  $a_d$  = # of paths at distance  $d$

- A good approximation:

$$P_e \approx a_{d_{\text{free}}} \cdot P_2(d_{\text{free}})$$

# Evaluating Error Probability

## Using the Transfer Function Bound

- Transfer function:  $T(D) = \sum_{d=d_{\text{free}}}^{\infty} a_d \cdot D^d$
- $T(D)$  is a polynomial in  $D$ . The coefficients of  $T(D)$  are the weights  $a_d$  which we need to determine.
- Using the Chernoff Bound, the transfer function leads to an easy to evaluate expression for  $P_e$ :

$$P_e \leq T(D) \Big|_{D=\sqrt{4p(1-p)}}$$

**Note: The Chernoff Bound provides a bound for the pairwise error probability. Specifically:**

$$P_2(d) \leq [4p(1-p)]^{d/2}$$

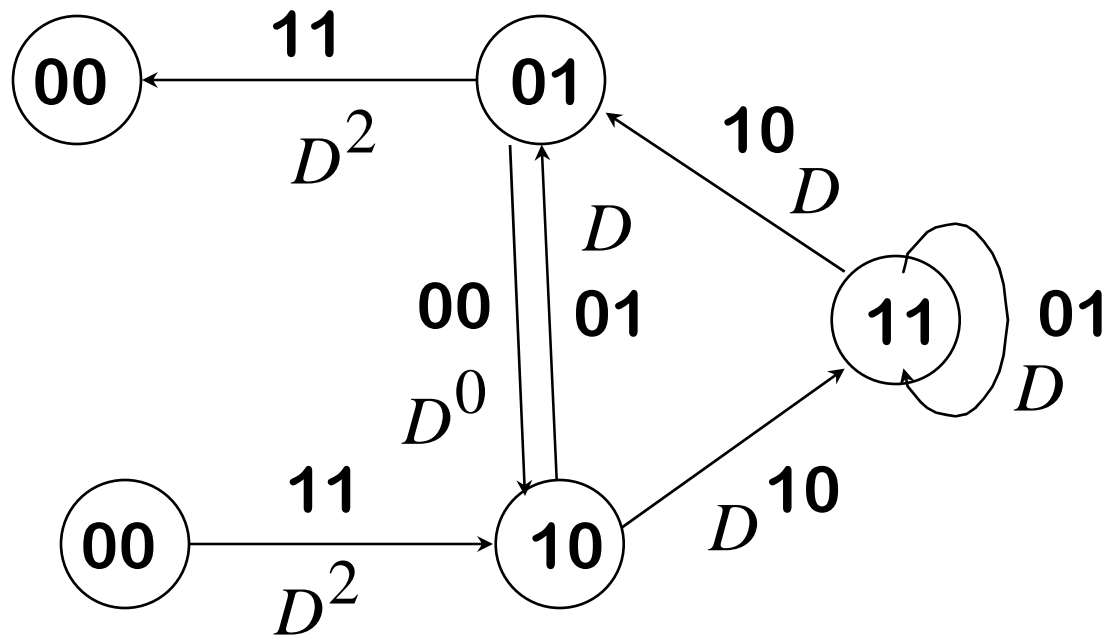


# Finding $T(D)$ from State Diagram

---

- Break the all 0's state in two, creating a starting state and a terminating state
- Re-label every output 1 as a  $D$
- $a_d$  is the number of distinct paths leading from the starting state to the terminating state while generating the function  $D^d$

# Example of State Diagram



$$T(D) = D^5 + 2D^6 + 4D^7 + \dots$$



# Performance Example for Convolutional Code

- Suppose we use a rate 1/2, constraint length 3 convolutional code with BPSK modulation. Find the value of  $E_b/N_0$  required for a first event error probability of  $P_e \leq 10^{-5}$

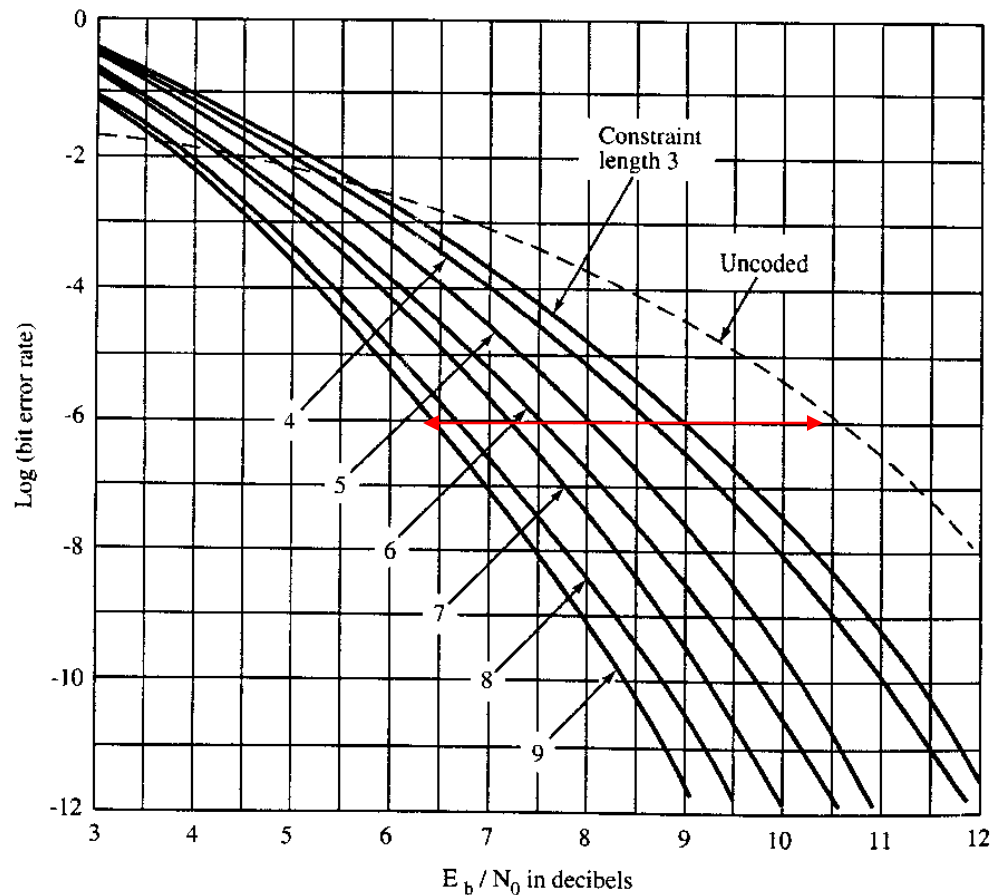
$$P_e \leq T(D) \Big|_{D=\sqrt{4p(1-p)}} \approx D^5 + 2D^6 + 4D^7 = 10^{-5}$$

$$\Rightarrow D = 0.096$$

$$\Rightarrow p = 0.0023 \quad (\text{from } D = \sqrt{4p(1-p)})$$

$$\Rightarrow E_b / N_0 = 9\text{dB} \quad (\text{from } p = Q(\sqrt{2rE_b/N_0}))$$

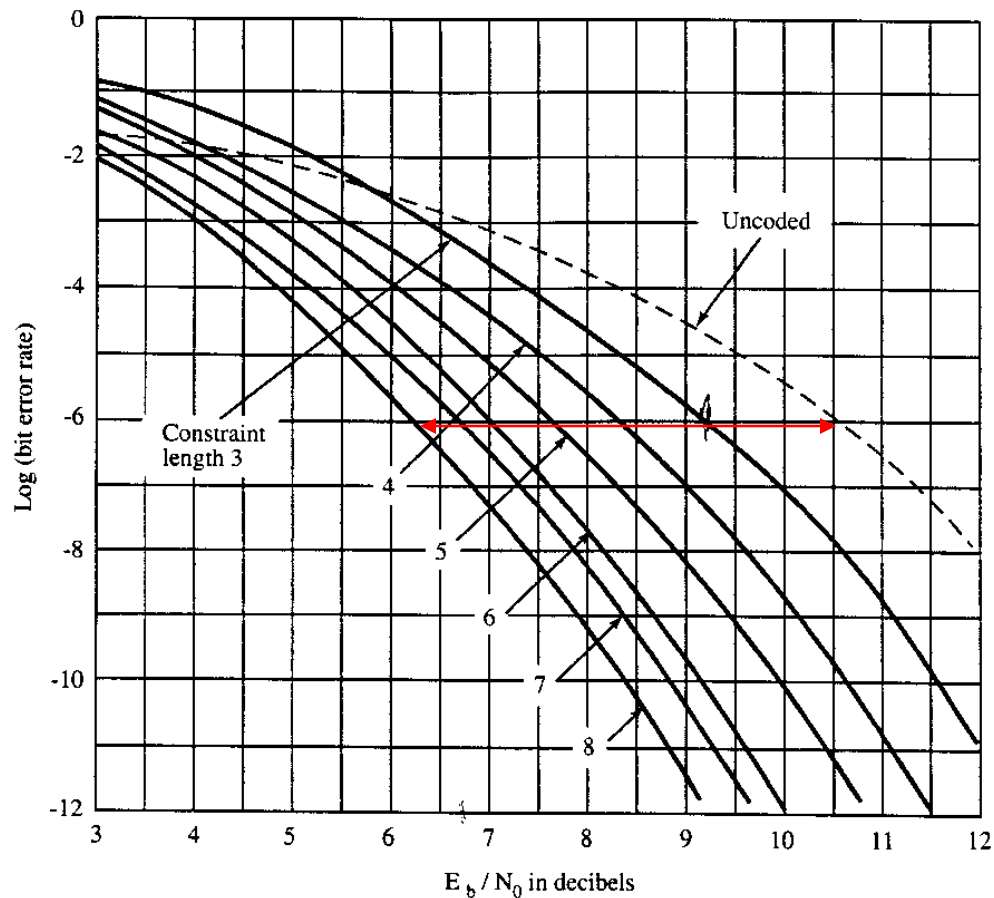
# Performance of $r=1/2$ Convolutional Codes with Hard Decisions



K	r	Gain (dB)
3	0.5	1.6
4	0.5	2.0
5	0.5	2.5
6	0.5	3.0
7	0.5	3.25
8	0.5	3.75
9	0.5	4.2

**FIGURE 6-19.** Bit error probability for rate  $1/2$  convolutional coding using hard decision channel.

# Performance of $r=1/3$ Convolutional Codes with Hard Decisions



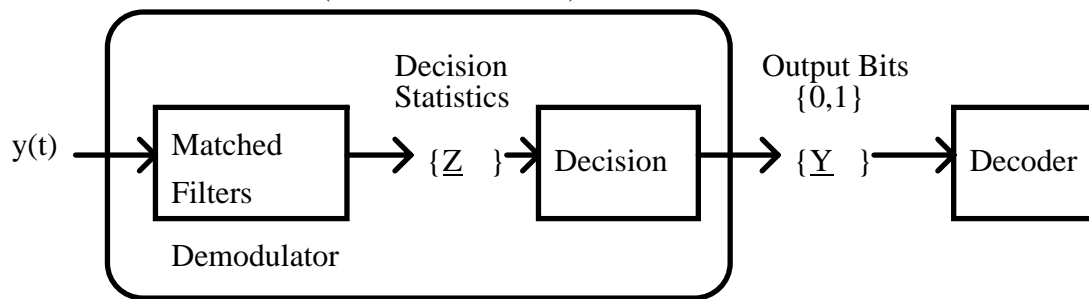
K	r	Gain (dB)
3	0.33	1.3
4	0.33	2.2
5	0.33	2.9
6	0.33	3.5
7	0.33	3.8
8	0.33	4.4

**FIGURE 6-21.** Bit error probability for rate  $1/3$  convolutional coding using hard decision channel.

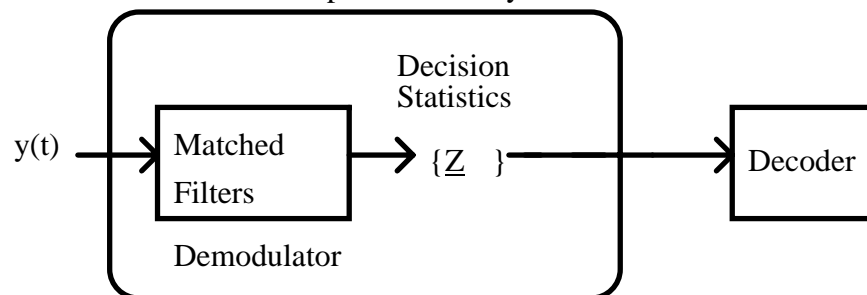
# Soft-Decision Decoding

- Basic Idea: Use unprocessed decision statistics rather than making "hard" decisions on individual bits.

Hard Decision Decoder (Cases 1 & 2 above)



Soft Decision Decoder operates directly on the decision statistics



**By using 'soft' decisions we maintain information about how likely the bit was correctly decoded**



# Simple Example of Soft-Decision Decoding

---

- Assume that BPSK modulation is used with rate (5,1) repetition code (i.e., there are two codewords (00000) and (11111) ).
- Suppose that the codeword  $\underline{c} = (11111)$  is sent, and the following five decision statistics are received:  $\underline{Z} = \{1.0, 1.0, -0.01, -0.02, -0.01\}$
- A hard decision decoder will look at the individual bits and decide that  $\underline{y} = (1\ 1\ 0\ 0\ 0)$  are the bits received. Since  $\underline{y}$  is nearer in Hamming distance to (00000) than (11111), the wrong codeword (00000) will be chosen.

# Simple Example of Soft-Decision Decoding (continued)

- A soft decision decoder will recognize that the first two bits {1.0, 1.0} were clearly received. The next three bits are not very strong (e.g., maybe the channel was in a deep fade) and therefore place less weight on them. A soft decision decoder will choose the correct codeword (11111).

**This occurs because soft decision decoding uses *Euclidean distance* as opposed to *Hamming distance*.**

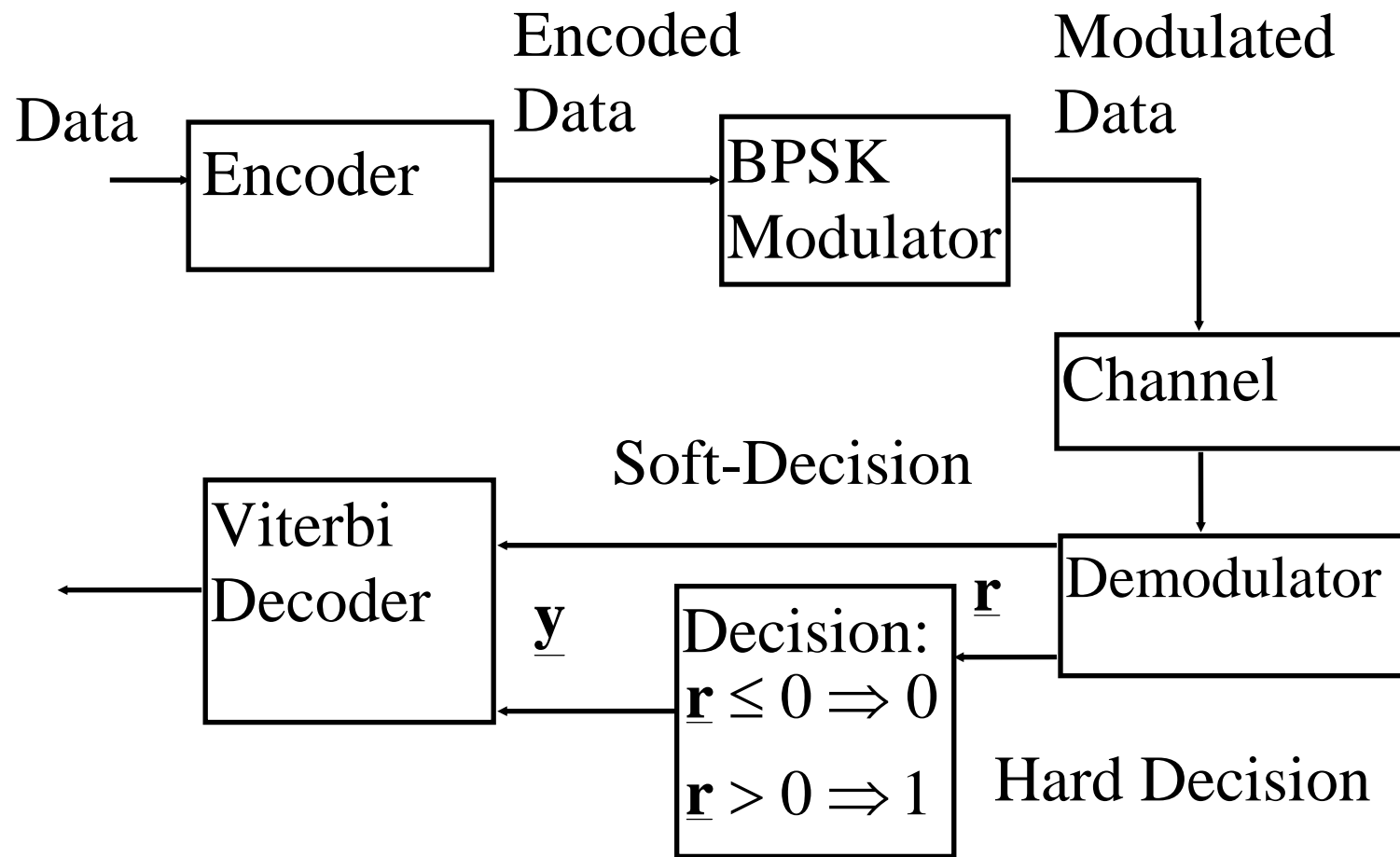
$$d_H(\underline{c}_0, \underline{z}) = 2$$

$$d_H(\underline{c}_1, \underline{z}) = 3$$

$$d_e(\underline{c}_0, \underline{z}) = 3.3$$

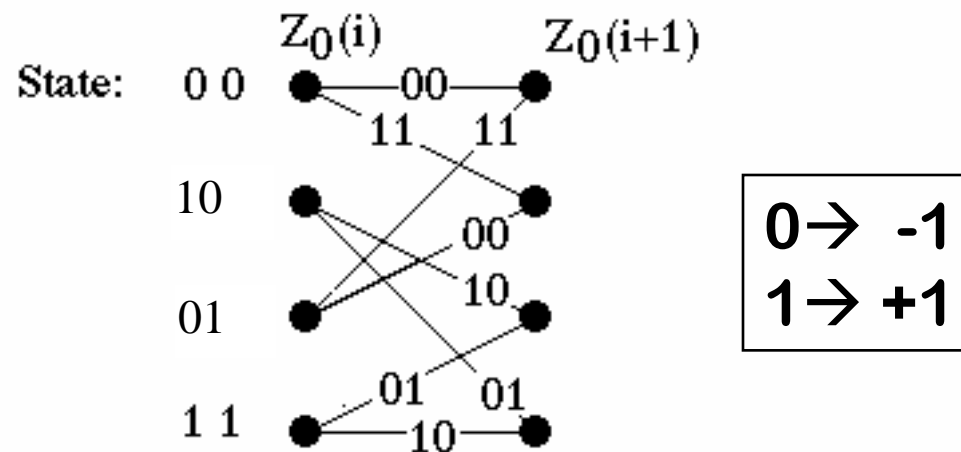
$$d_e(\underline{c}_1, \underline{z}) = 1.75$$

# Soft-Decision Decoding of Convolutional Codes



# Modification of the Viterbi Algorithm for Soft Decisions

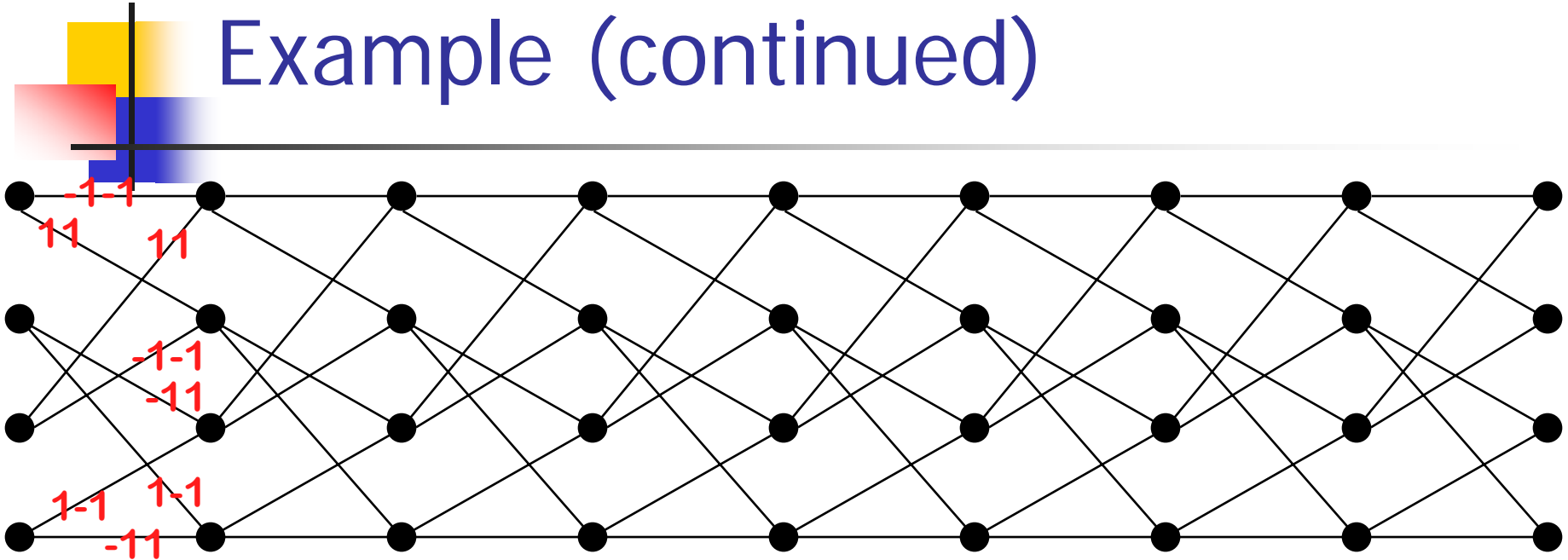
- Trellis is labeled transmitted code vectors (shown for BPSK here)



- Metric is now *squared* Euclidean distance
- Viterbi Algorithm operates just as before except:

$$d(\underline{\mathbf{y}}_i, \underline{\mathbf{c}}_i) = (r_{i,1} - c_{i,1})^2 + (r_{i,2} - c_{i,2})^2$$

# Viterbi Algorithm Decoding Example (continued)



$\underline{y} = (-1, 0.5 \quad -1, 1 \quad 2, -1 \quad -1, -2 \quad 1.5, -0.5 \quad 1, -1 \quad 2, 1)$

$$d([-1, 0.5], [-1, -1]) = (-1+1)^2 + (0.5+1)^2 = 2.25$$

$$d([-1, 0.5], [1, 1]) = (-1-1)^2 + (0.5-1)^2 = 4.25$$

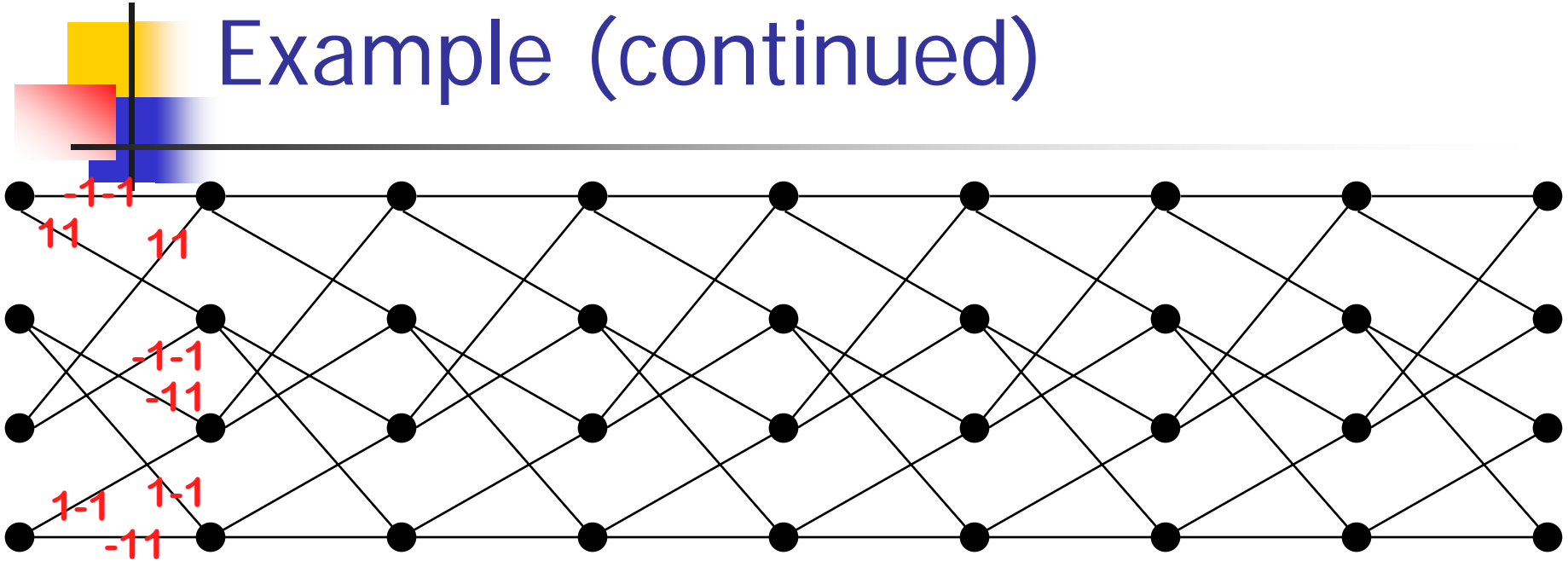
$$d([-1, 0.5], [-1, 1]) = (-1+1)^2 + (0.5-1)^2 = 0.25$$

$$d([-1, 0.5], [1, -1]) = (-1-1)^2 + (0.5+1)^2 = 6.25$$



**These branch  
distance metrics  
are now used to  
create the path  
metrics**

# Viterbi Algorithm Decoding Example (continued)



$\underline{y} = (-1, 0.5 \quad -1, 1 \quad 2, -1 \quad -1, -2 \quad 1.5, -0.5 \quad 1, -1 \quad 2, 1)$

$$\begin{aligned} d_{[-1,-1]} &= 2.25 \\ d_{[1,1]} &= 4.25 \\ d_{[-1,1]} &= 0.25 \\ d_{[1,-1]} &= 6.25 \end{aligned}$$

$$\begin{aligned} d_{[-1,-1]} &= 4 \\ d_{[1,1]} &= 4 \\ d_{[-1,1]} &= 0 \\ d_{[1,-1]} &= 8 \end{aligned}$$

$$\begin{aligned} d_{[-1,-1]} &= 9 \\ d_{[1,1]} &= 5 \\ d_{[-1,1]} &= 13 \\ d_{[1,-1]} &= 1 \end{aligned}$$

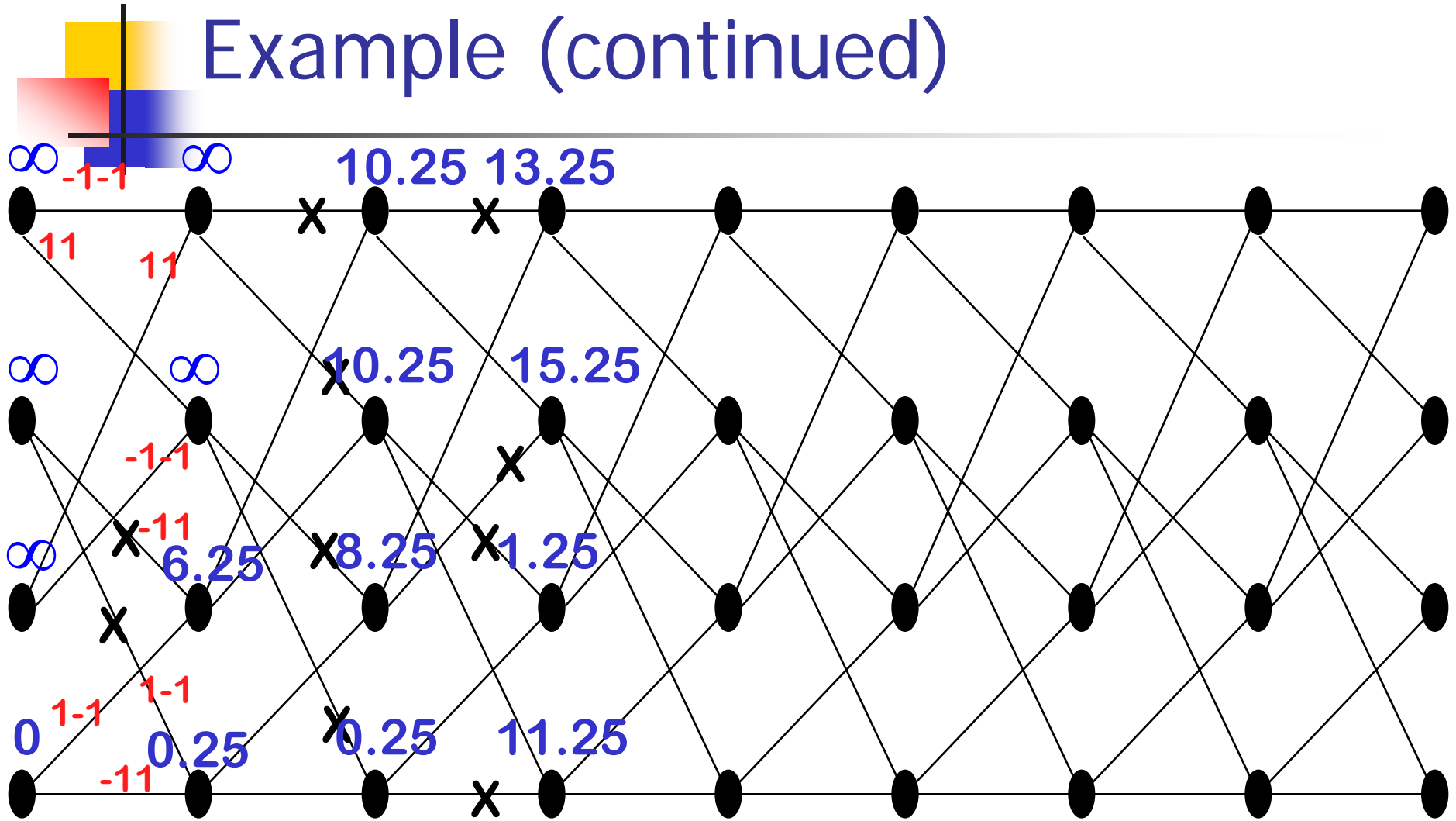
$$\begin{aligned} d_{[-1,-1]} &= 1 \\ d_{[1,1]} &= 13 \\ d_{[-1,1]} &= 9 \\ d_{[1,-1]} &= 5 \end{aligned}$$

$$\begin{aligned} d_{[-1,-1]} &= 6.5 \\ d_{[1,1]} &= 2.5 \\ d_{[-1,1]} &= 8.5 \\ d_{[1,-1]} &= 0.5 \end{aligned}$$

$$\begin{aligned} d_{[-1,-1]} &= 4 \\ d_{[1,1]} &= 4 \\ d_{[-1,1]} &= 8 \\ d_{[1,-1]} &= 8 \end{aligned}$$

$$\begin{aligned} d_{[-1,-1]} &= 13 \\ d_{[1,1]} &= 1 \\ d_{[-1,1]} &= 9 \\ d_{[1,-1]} &= 5 \end{aligned}$$

# Viterbi Algorithm Decoding Example (continued)



$\underline{y} = (-1, 0.5 \quad -1, 1 \quad 2, -1 \quad -1, -2 \quad 1.5, -0.5 \quad 1, -1 \quad 2, 1)$

# Performance of Soft-Decision Decoding

- Pairwise Error Probability:

$$P_2(d) = Q\left(\frac{d_e}{\sqrt{2N_0}}\right) = Q\left(\sqrt{\frac{2E_b r d_H}{N_0}}\right)$$

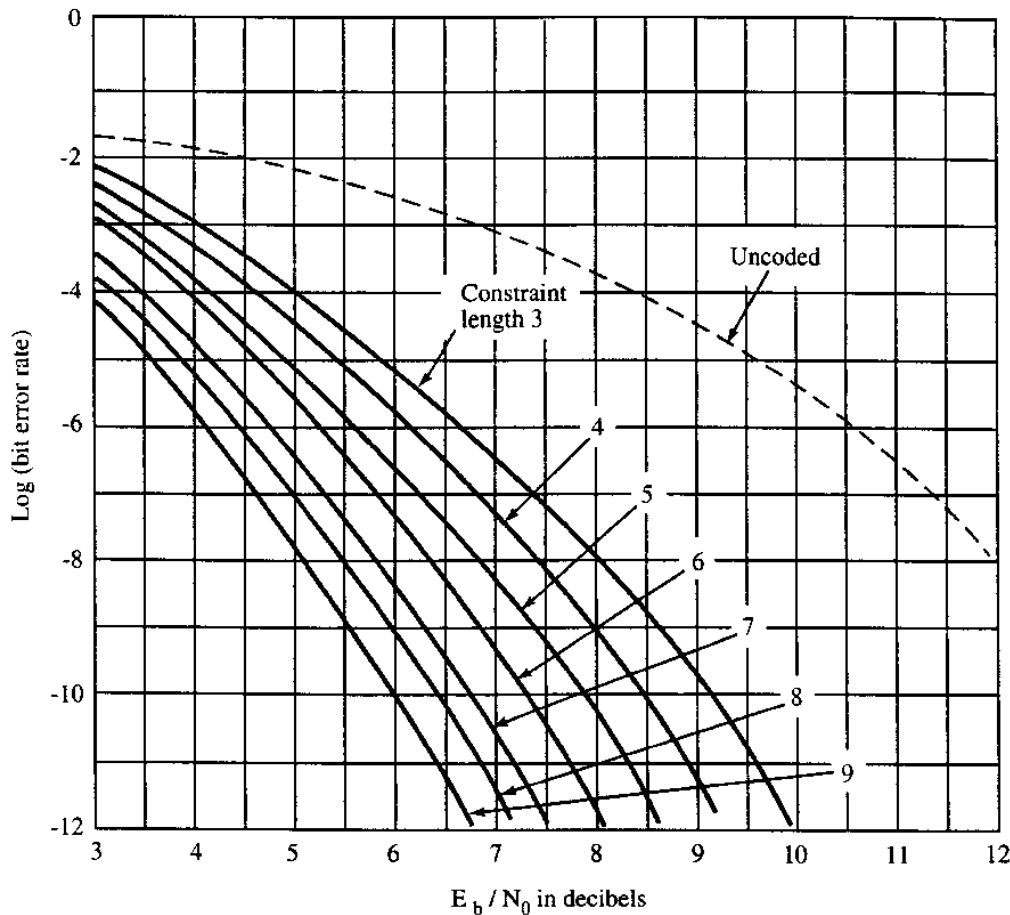
where  $d_e^2 = 4E_b r d_H$  (for BPSK)

- Error Event Probability:

$$P_e \leq \sum_{d_H=d_{\text{free}}}^{\infty} a_d \cdot P_2(d) \approx a_{d_{\text{free}}} \cdot P_2(d_{\text{free}})$$

- Soft Decisions typically result in 2-3 dB more coding gain

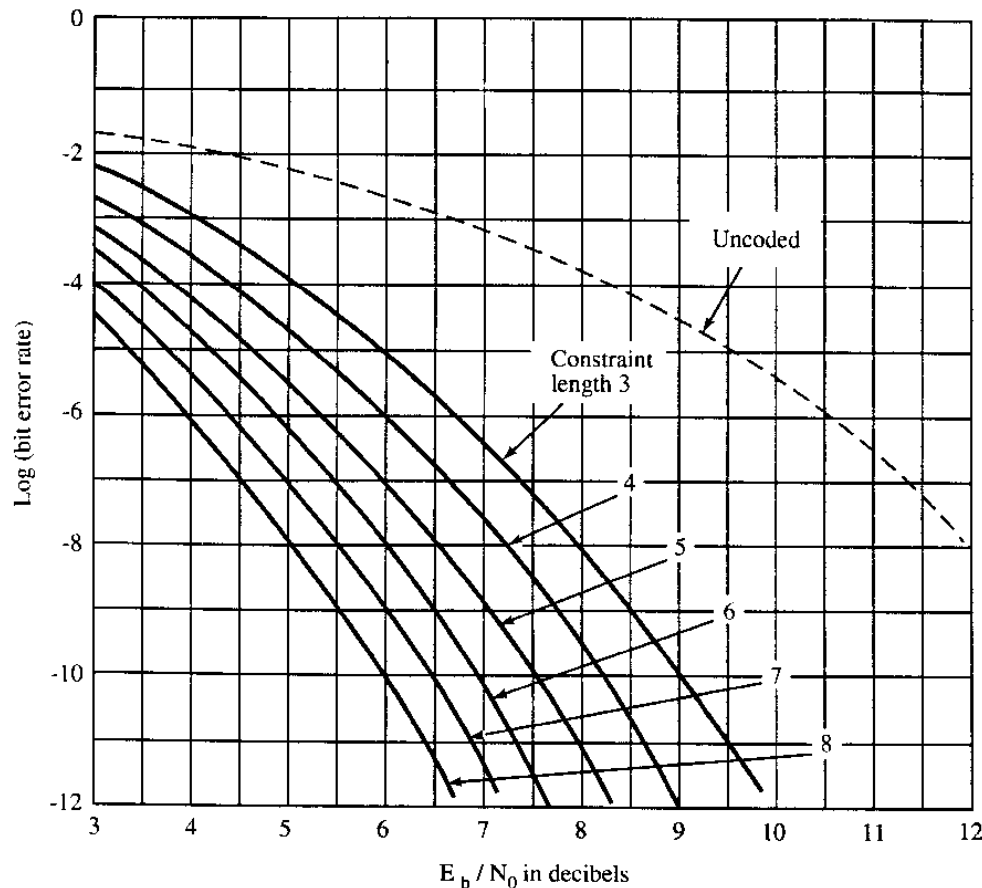
# Performance of $r=1/2$ Convolutional Codes with Soft Decisions



K	r	Gain (dB)
3	0.5	4.0
4	0.5	4.4
5	0.5	5.0
6	0.5	5.3
7	0.5	5.8
8	0.5	6.2
9	0.5	6.6

**FIGURE 6-20.** Bit error probability for rate  $1/2$  convolutional coding using soft decision channel.

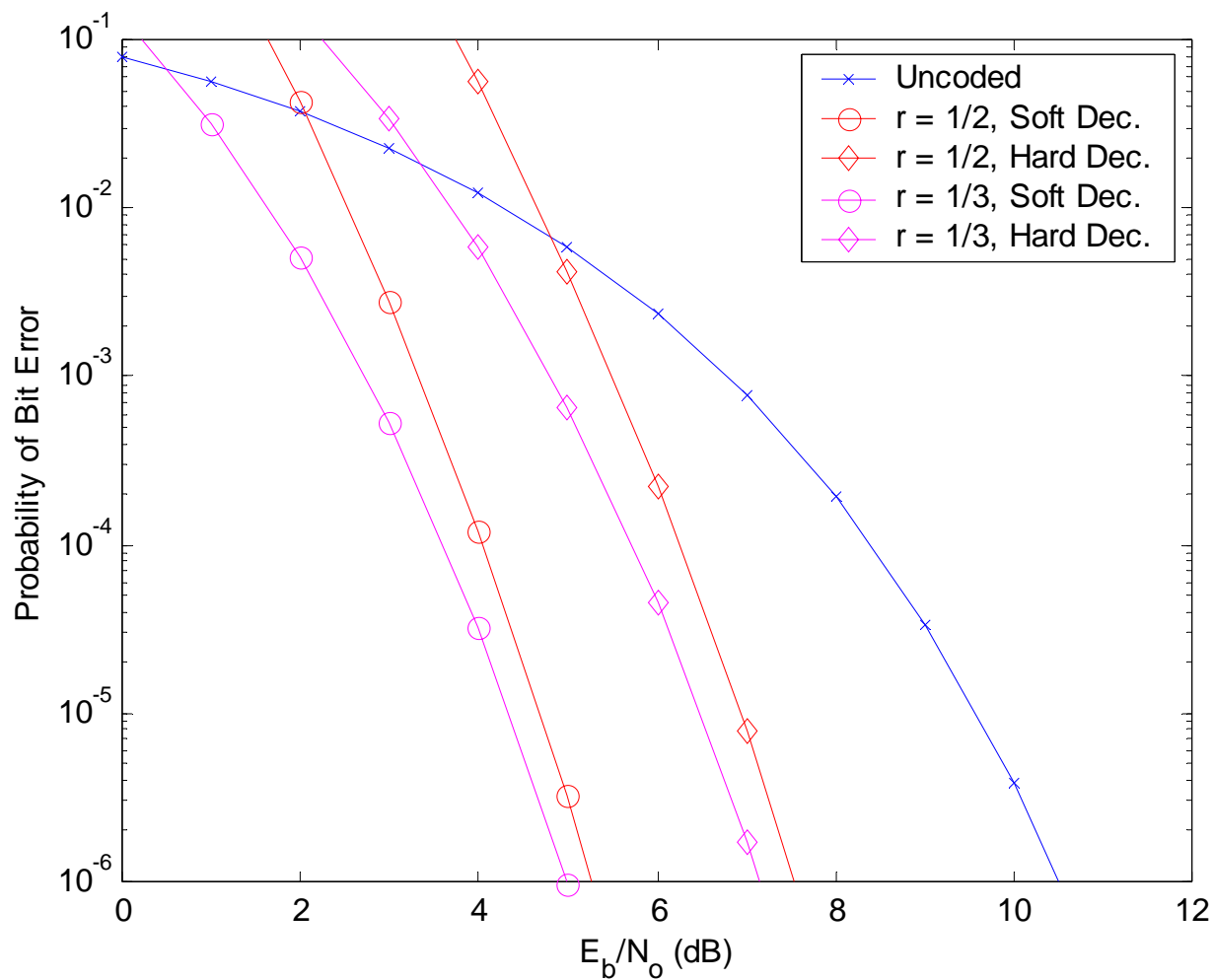
# Performance of $r=1/3$ Convolutional Codes with Soft Decisions



K	r	Gain (dB)
3	0.33	3.9
4	0.33	4.5
5	0.33	5.3
6	0.33	5.8
7	0.33	6.2
8	0.33	6.6

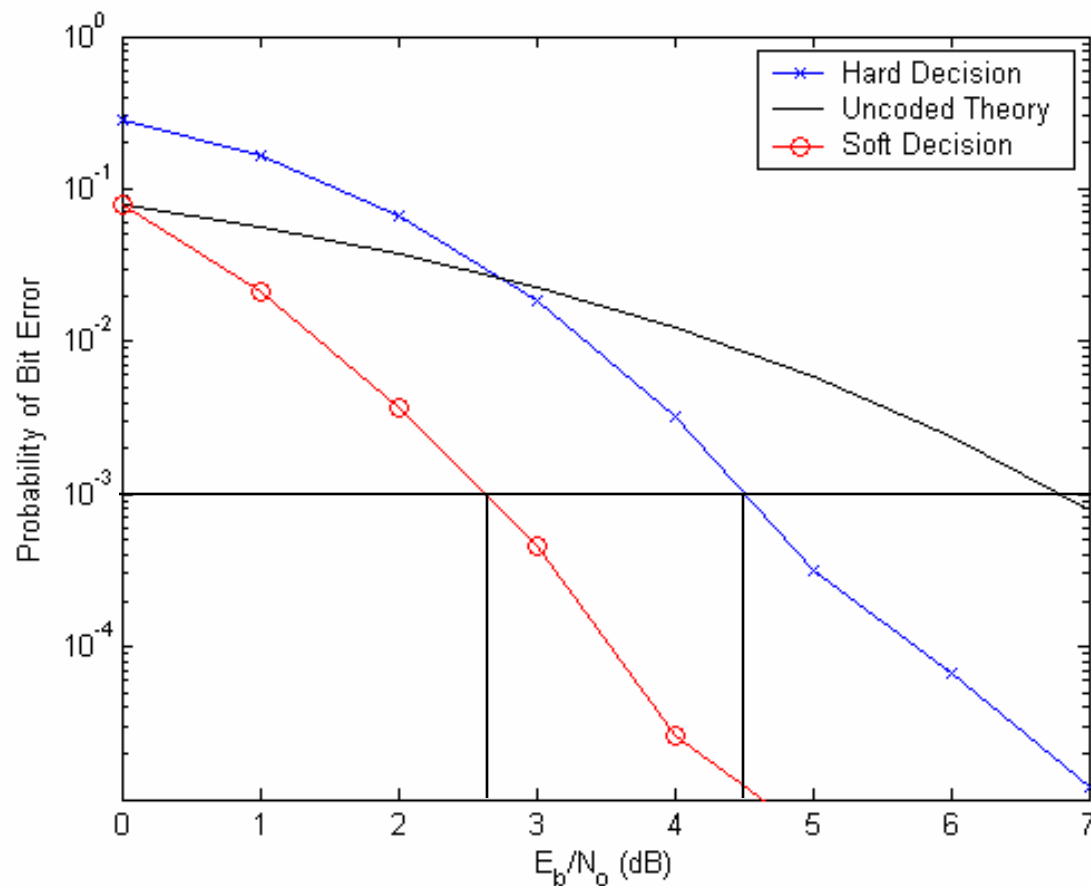
**FIGURE 6-22.** Bit error probability for rate  $1/3$  convolutional coding using soft decision channel.

# Examples



- $r = 1/2$ ,  $K=7$  conv. Code with hard and soft decisions
- $r = 1/3$  conv. Code with hard and soft decisions
- Decreasing rate from  $1/2$  to  $1/3$  provides  $\sim 0.5$ dB gain
- Soft decision decoding provides  $\sim 2$ dB gain

# Examples



- $r = 1/6$  ,  $K=8$  conv. Code with hard and soft decisions
- Soft decision decoding provides  $\sim 2$ dB gain at a BER of  $10^{-3}$

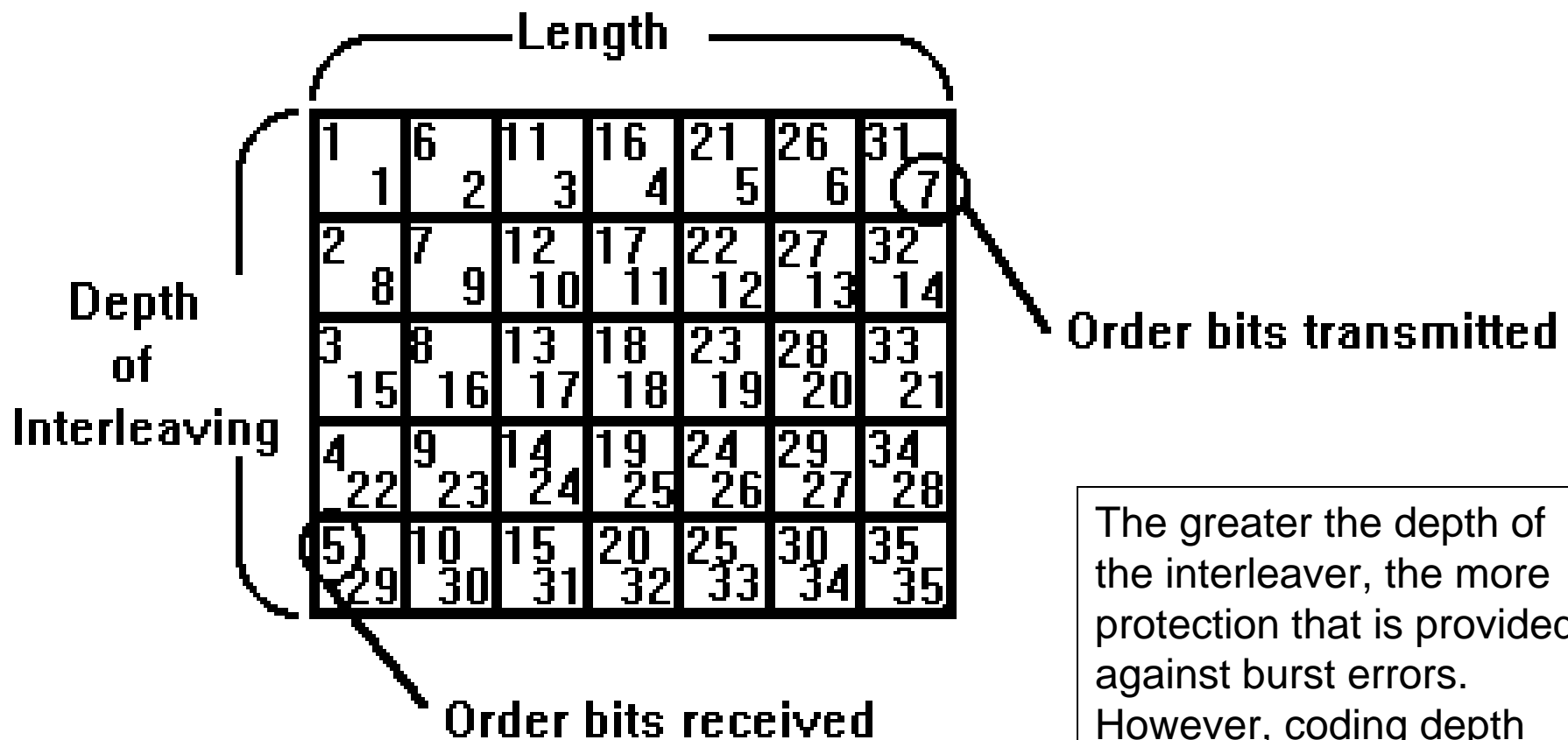


# Interleaving

---

- The vast majority of Forward Error Correction (FEC) codes are designed for an AWGN channel which exhibits no memory.
- Thus, they do not handle bursts of errors well.
- Burst errors are common in jamming scenarios (esp. pulse jammers) as well as in time-varying (i.e., fading) channels
- Interleavers randomize the order of bits going into the channel. Bursts of errors are then spread out after deinterleaving.
- Ideally, this provides the decoder with random, independent errors.

# Block Interleaver



The greater the depth of the interleaver, the more protection that is provided against burst errors. However, coding depth also introduces delay.

# Interleaver - Example

- Bits going into the interleaver

$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$	$b_{10}$	$b_{11}$	$b_{12}$	$b_{13}$
-------	-------	-------	-------	-------	-------	-------	-------	-------	----------	----------	----------	----------

- Coming out of the interleaver

$b_1$	$b_6$	$b_{11}$	$b_{16}$	$b_{21}$	$b_{31}$	$b_2$	$b_7$	$b_{12}$	$b_{17}$	$b_{22}$	$b_{27}$	$b_{32}$
-------	-------	----------	----------	----------	----------	-------	-------	----------	----------	----------	----------	----------

- After demodulator (errors due to pulse interference)

$b_1$	$b_6$	$b_{11}$	$b_{16}$	$b_{21}$	$b_{31}$	$b_2$	$b_7$	$b_{12}$	$b_{17}$	$b_{22}$	$b_{27}$	$b_{32}$
-------	-------	----------	----------	----------	----------	-------	-------	----------	----------	----------	----------	----------

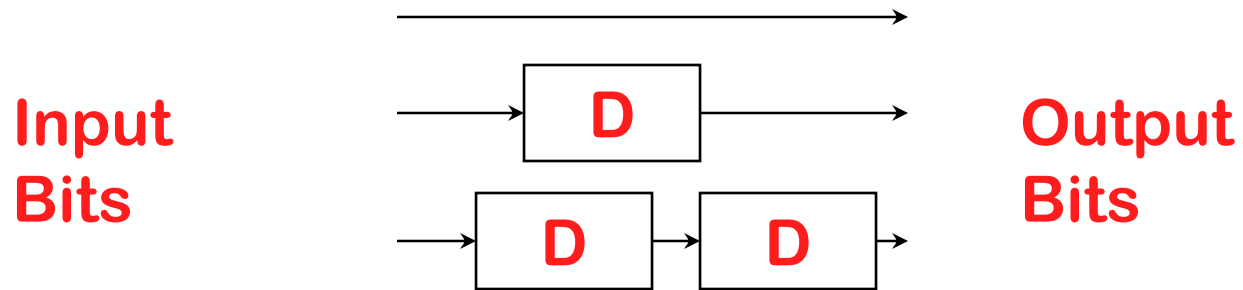
Error Burst

- After de-interleaving – errors dispersed

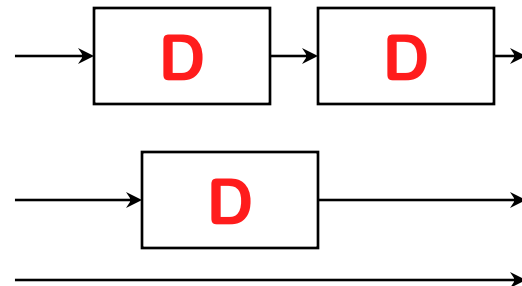
$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$	$b_{10}$	$b_{11}$	$b_{12}$	$b_{13}$
-------	-------	-------	-------	-------	-------	-------	-------	-------	----------	----------	----------	----------

# Convolutional Interleavers

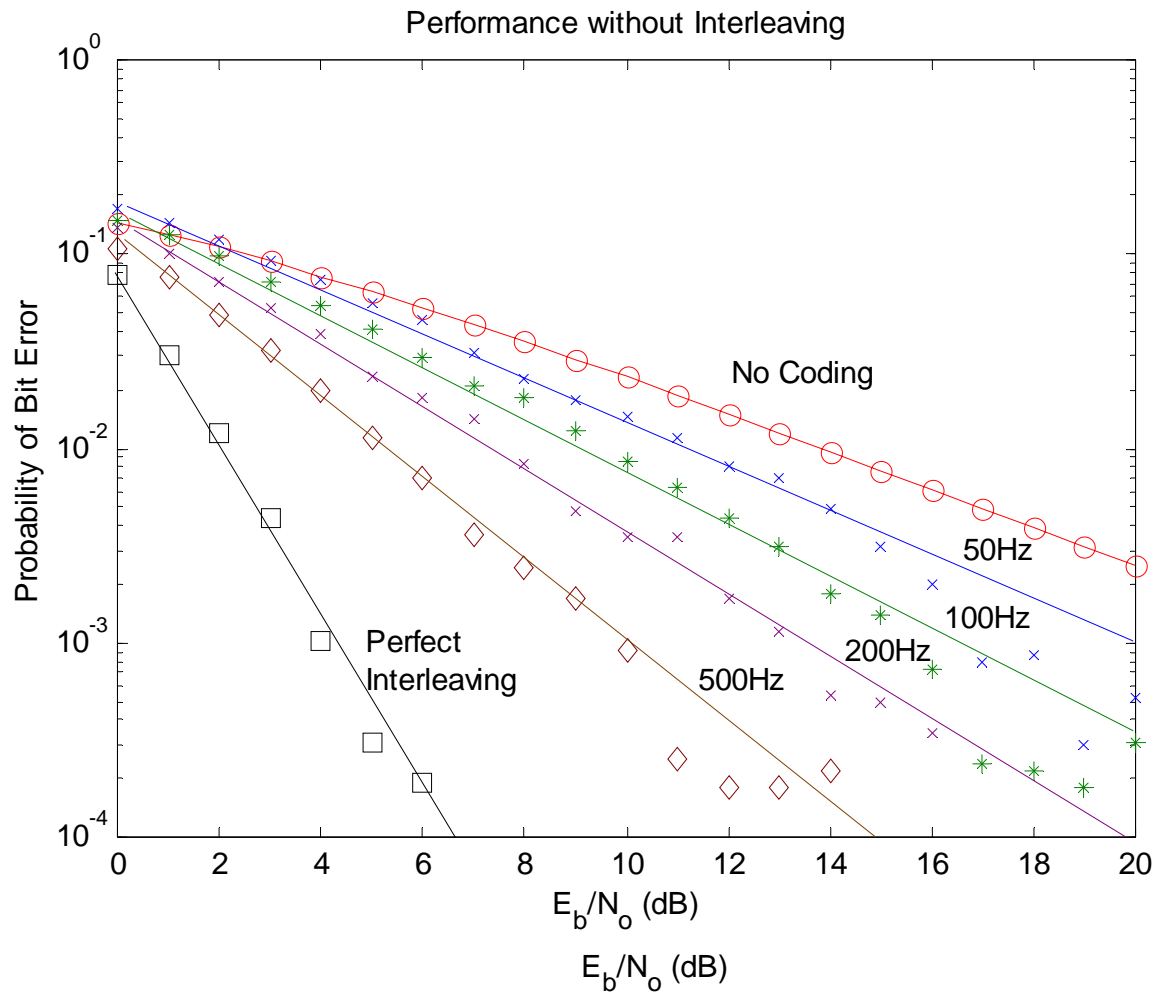
- Also called Cross-Interleaver



- Corresponding Deinterleaver

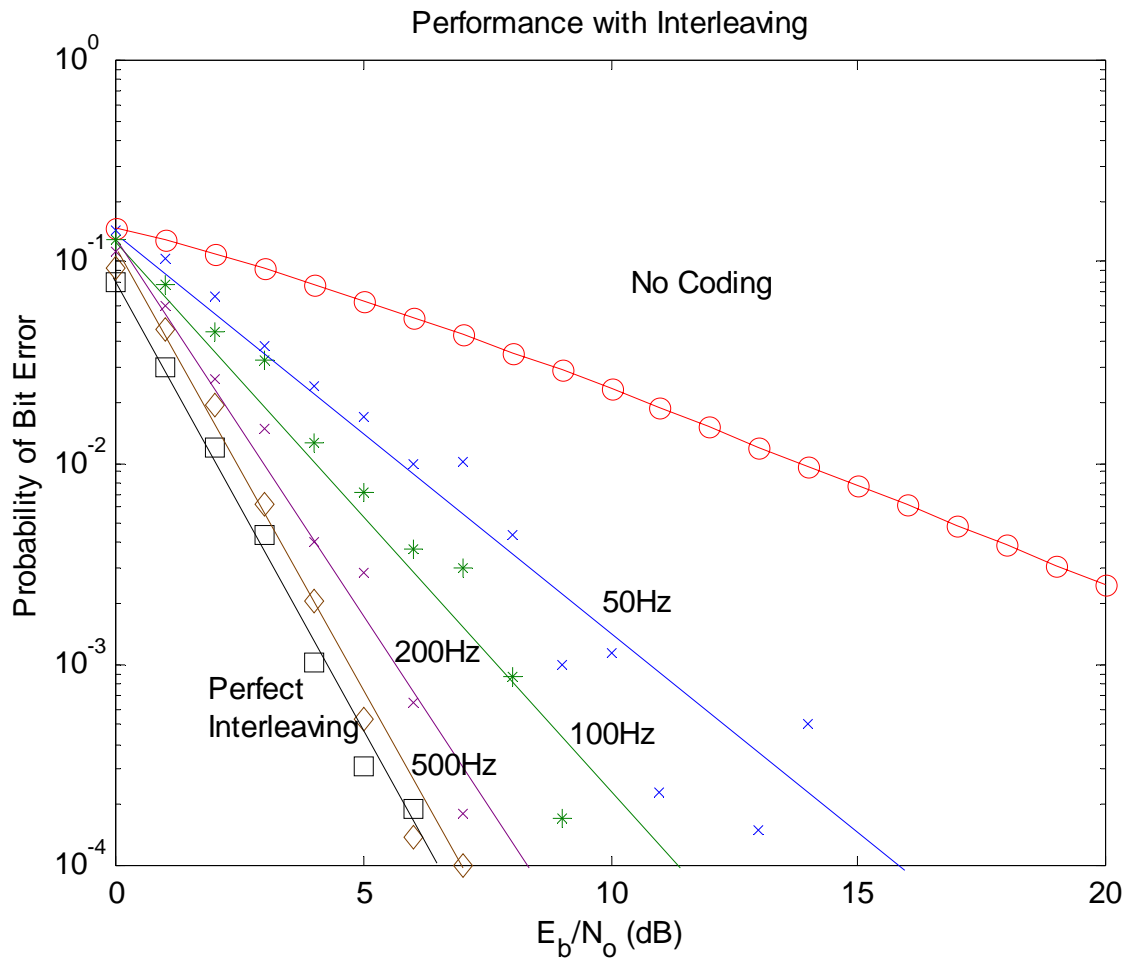


# Performance with Interleaving



- Simulated Performance
- Rayleigh Fading
- $R = \frac{1}{2}$  constraint length 7 convolutional code
- 20ms frames
- No interleaving
- 50Hz, 100Hz, 200Hz, 500Hz Maximum Doppler Spreads

# Performance with Interleaving



- Simulated Performance
- 1 Path
- $R = \frac{1}{2}$  constraint length 7 convolutional code
- 20ms frames
- Block interleaving
- 50Hz, 100Hz, 200Hz, 500Hz Maximum Doppler Spreads