

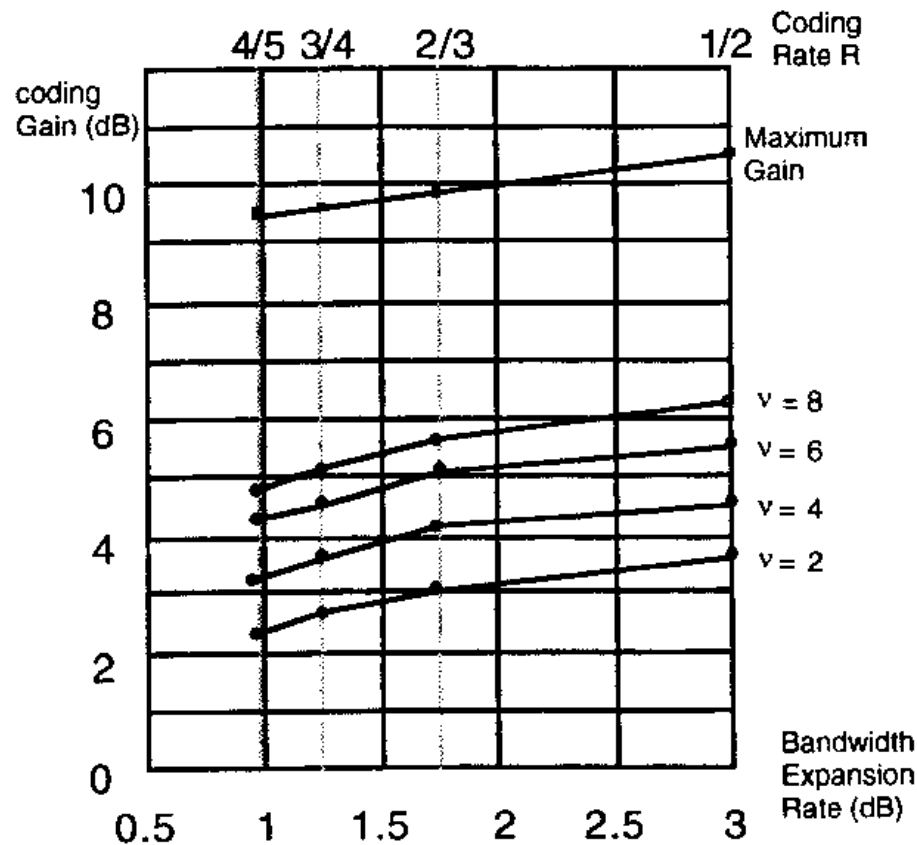
# EE 5654 - Digital Communications Spring 2005

---

Instructor: R. Michael Buehrer  
Lecture #25 - Turbo Codes



# Performance Gap for Convolutional Codes



Memory or constraint length

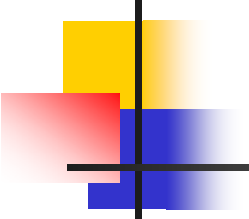
NSC = non-systematic convolutional

Source: Berrou and Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo Codes," IEEE Trans. On Comm., vol. 44, no. 10., pp. 1261-

1271 Oct 1996

Fig. 1. Coding gains at BER equal to  $10^{-6}$ , achievable with NSC codes (three-bit quantization, from [4], and maximum possible gains for 1/2, 2/3, 3/4, and 4/5 rates in a Gaussian channel, with quaternary phase shift keying (QPSK) modulation.

# Motivation for Turbo Codes

- 
- 
- Random codes are known to be good for sufficiently long block length. (known from information theory)
  - However, random codes have no structure and are hard to decode.
  - Most known codes with structure do not have as good performance as theory predicts is possible.
  - Goal: Develop codes with enough structure to decode them, but which “appear” random.



# Turbo Codes

---

- Proposed by team of French Researchers with interest in VLSI design
  - Berrou, Glavieux, and Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes" *ICC'93.*, pp. 1064-1070 October 1993.
  - Berrou, Glavieux, and Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes" *IEEE Trans. Comm.*, October 1996.
- *Parallel* contatenation of convolutional codes is used to give the codes structure so they can be decoded
- Psuedorandom interleaving is used to give the codes performance which approaches that for random coding



# Parallel Concatenated Codes

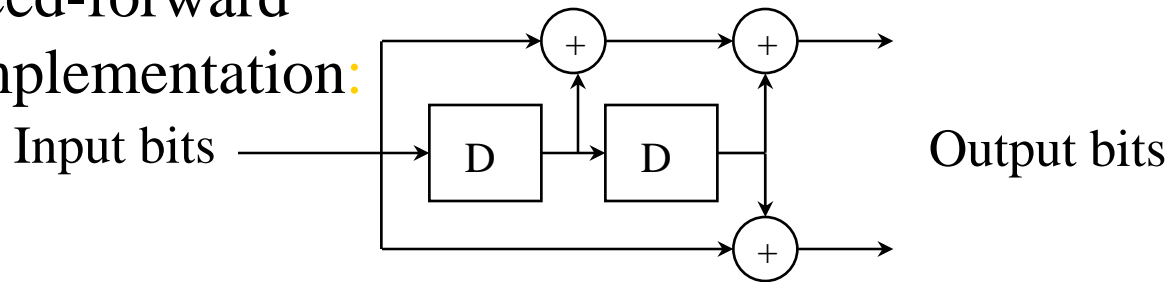
---

- Previously we discussed concatenated codes where the output of one encoder was fed into a second encoder. This is an example of *serial concatenated codes*.
- Instead of concatenating in serial, codes can also be concatenated in parallel.
- The original turbo code is a parallel concatenation of two *recursive systematic convolutional* (RSC) codes.
  - ***systematic***: one of the outputs is the input.
  - Convolutional codes can always be converted to a recursive systematic code without changing distance properties

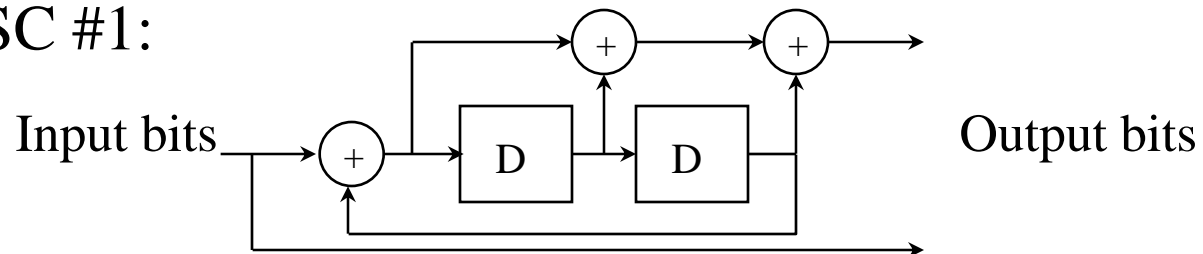
# Example RSC Code

Feed-forward

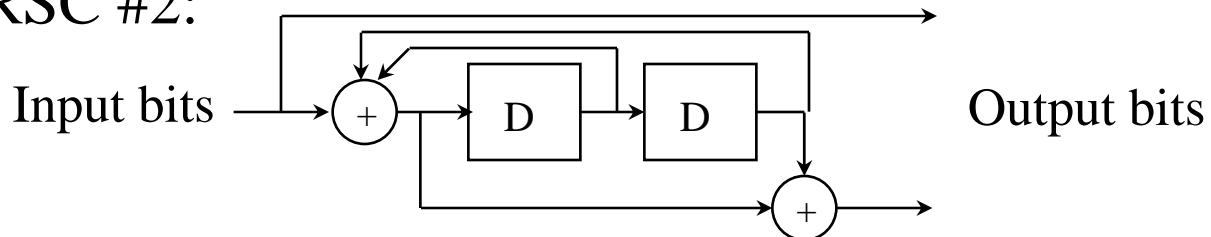
Implementation:



RSC #1:



RSC #2:





# Encoding of Turbo Codes

---

- Two Recursive Systematic Convolutional Codes are concatenated in parallel (rather than series).
- Each data bit generates two separate sets of check bits (one for each code).
- Data is interleaved according to a psuedo-random pattern prior to generating second set of output bits so that the two sets of check bits are independently generated.
- It may be possible to decode even if some of the check bits are dropped (this is called puncturing).

# Encoder for Basic Rate 1/3 Turbo Code

Data Bit

Systematic Bit

RSC code #1

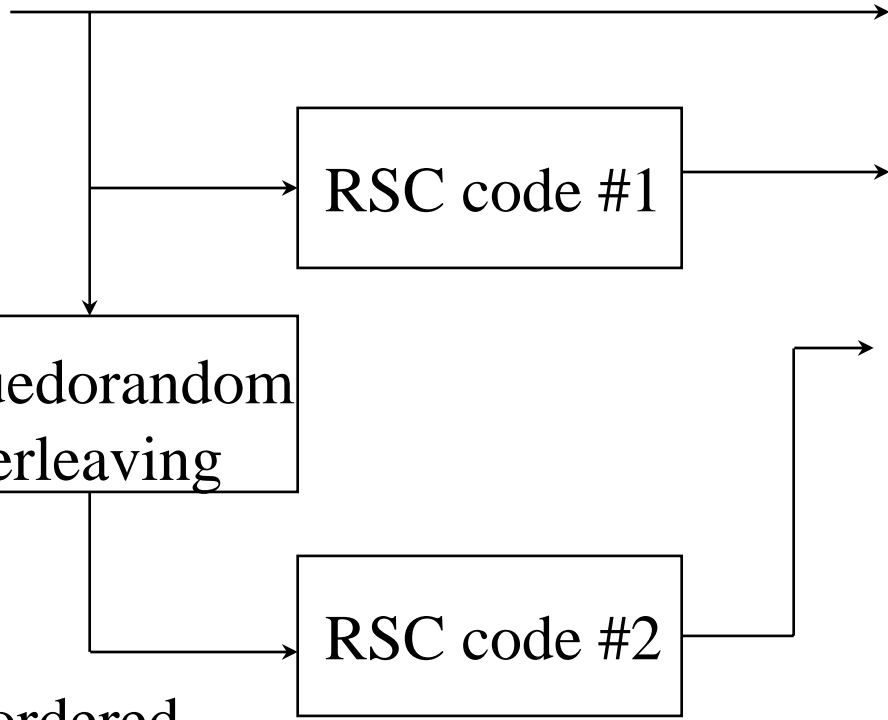
Redundant Bits

Pseudorandom  
Interleaving

RSC code #2

Reordered  
Data Bits

**Note: The systematic bits from the RSC coders are redundant and thus dropped.**





# Pseudo-random Interleaving

---

- The coding dilemma:
  - Shannon showed that large block-length random codes achieve channel capacity.
  - However, codes must have structure that permits decoding with reasonable complexity.
  - Codes with structure don't perform as well as random codes.
  - "Almost all codes are good, except those that we can think of."
- Solution:
  - Make the code appear random, while maintaining enough structure to permit decoding.
  - This is the purpose of the pseudo-random interleaver.
  - Turbo codes possess random-like properties.
  - However, since the interleaving pattern is known, decoding is possible.

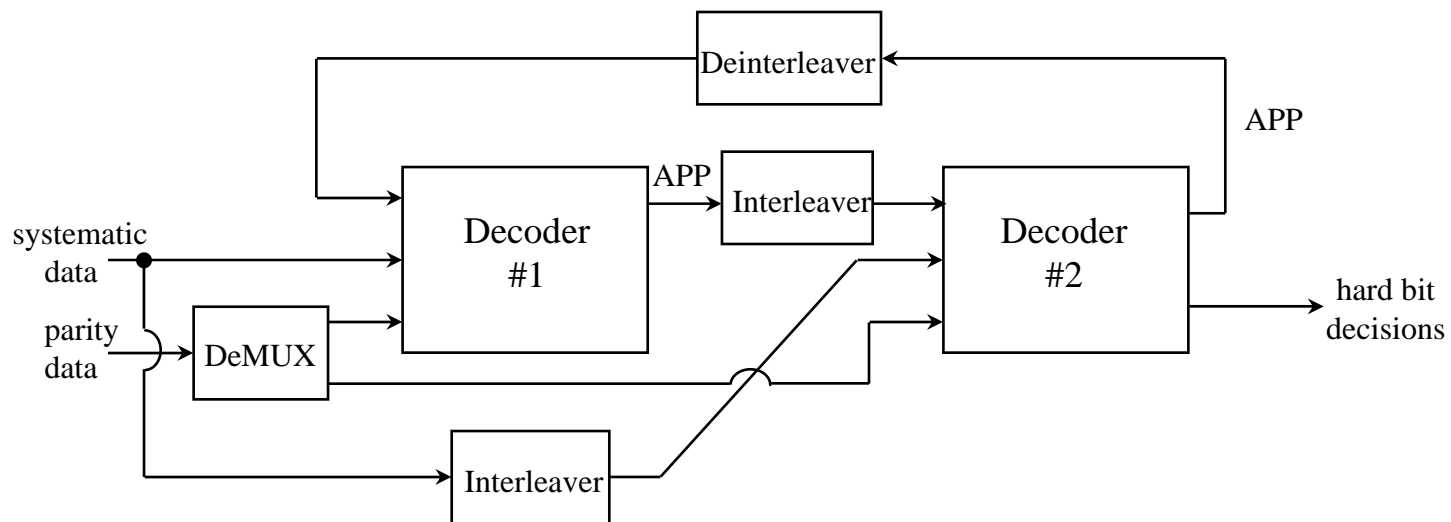


# Iterative Decoding

---

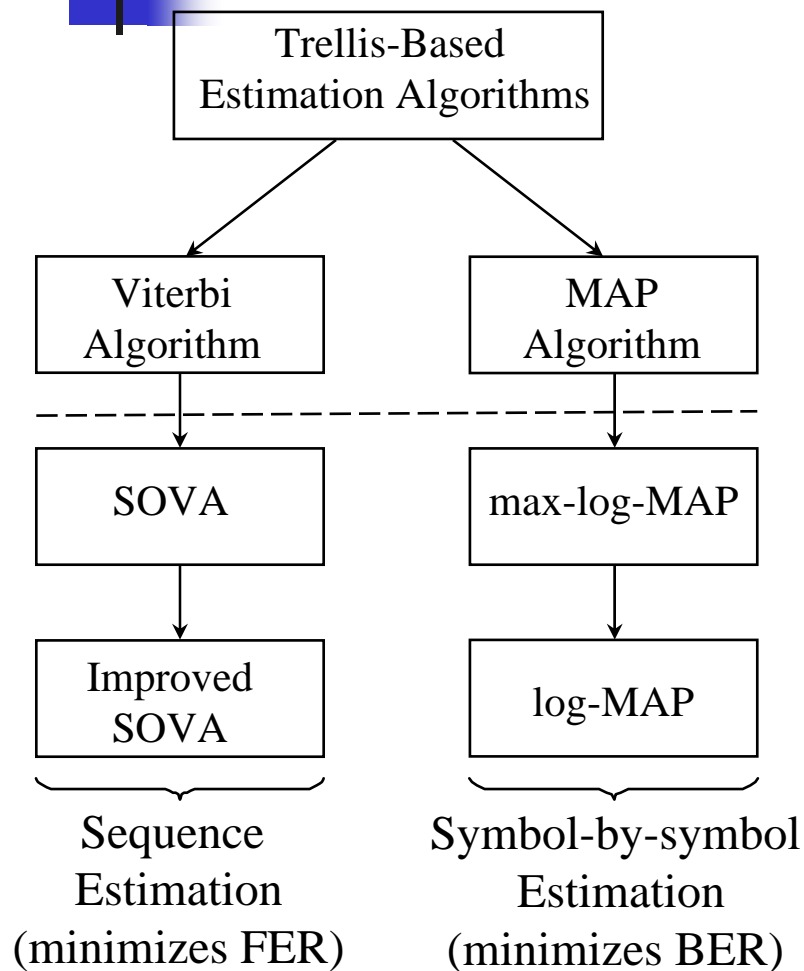
- Due to the random inter-leaver Maximum Likelihood decoding of these codes is incredibly complex. However, a sub-optimum iterative method works well.
- There is one decoder for each elementary encoder.
- Each decoder estimates the *a posteriori probability* (APP) of each data bit.
- The APP's are used as *a priori* information by the other decoder.
  - In other words the output of the previous decoder tells the current decoder how likely certain bits are.
- Decoding continues for a set number of iterations.
  - Performance generally improves from iteration to iteration, but follows a law of diminishing returns.

# Iterative Decoding



- **APP = a posteriori probabilities**
- **Instead of producing estimates of the data bits, the decoders should produce information which says how likely each bit is.**
- **At the end of the last iteration, we can make final decisions on the bits**
- **Because of the resemblance of this feedback between the decoders to a turbo engine, this is called *turbo decoding***

# Classes of Decoding Algorithms



- Requirements:
  - Accept Soft-Inputs in the form of *a priori* probabilities or log-likelihood ratios.
  - Produce APP estimates of the data.
  - “Soft-Input Soft-Output”
- MAP
  - (symbol-by-symbol) Maximum A Posteriori
- SOVA
  - Soft Output Viterbi Algorithm



# Decision Metrics

---

- The Viterbi Algorithm can operate using any additive metric
- We have seen two different metrics
  - Hamming distance - measures difference between sequences
  - Euclidean distance - measures difference between signals in Gaussian noise
- We will want to use an even more general metric:
  - Log-likelihood ratios



# Log-likelihood Ratios

---

- Let  $d$  be a bit that we want to determine the value of

- Then 
$$L(d) = \ln \frac{\Pr[d = 1]}{\Pr[d = 0]}$$

is the ***log-likelihood ratio (LLR)***

- Interpretation
  - $\Pr[d=1]=0.5 \rightarrow L(d) = 0$
  - $\Pr[d=1]=1 \rightarrow L(d)$  approaches infinity
  - $\Pr[d=1]=0 \rightarrow L(d)$  approaches minus infinity



# Relationship to Probability

---

- We can convert from a LLR directly to probability

$$L(d) = \ln \frac{\Pr[d = 1]}{\Pr[d = 0]} \Rightarrow e^{L(d)} = \frac{\Pr[d = 1]}{\Pr[d = 0]}$$

$$\Rightarrow e^{L(d)} = \frac{\Pr[d = 1]}{1 - \Pr[d = 1]}$$

$$\Rightarrow \Pr[d = 1] = \frac{e^{L(D)}}{1 + e^{L(D)}}$$

$$\Rightarrow \Pr[d = 0] = \frac{1}{1 + e^{L(D)}}$$

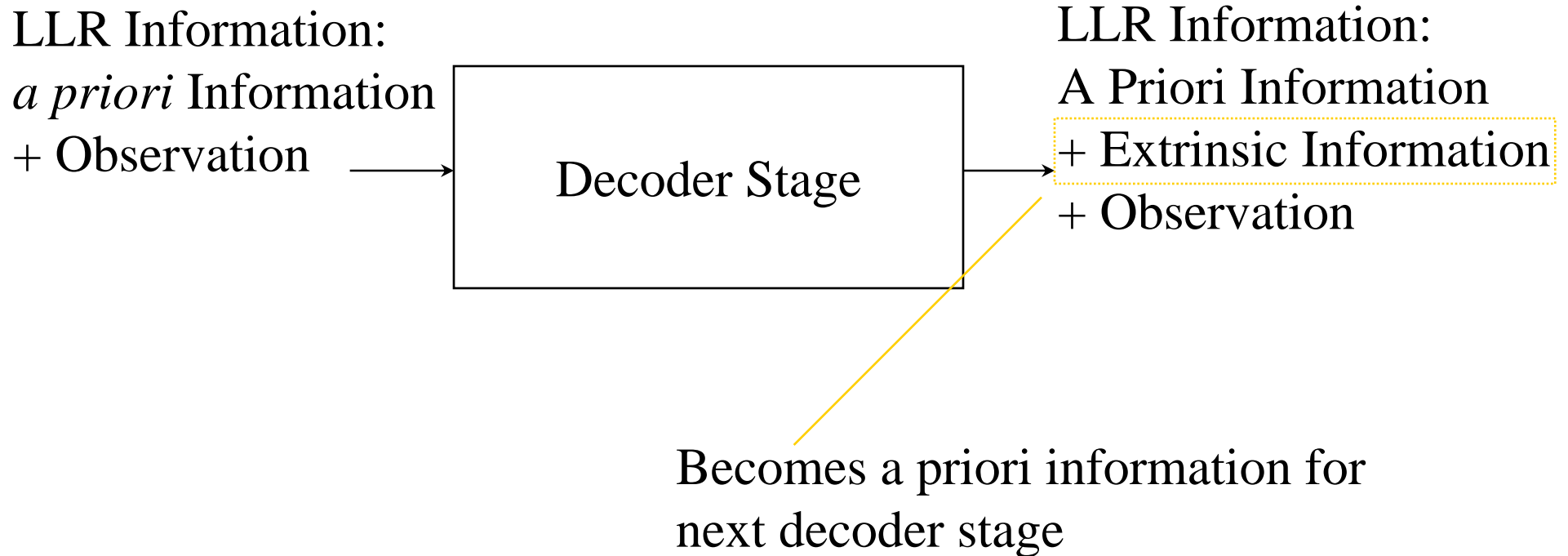


# Setup of Turbo Decoding Problem

---

- The decoders output LLR's for each bit in the sequence.
- These LLR's can be broken down into three components
  - *a priori* information
  - observed systematic data
  - value added by the decoder
- The output added by the decoder is called ***extrinsic*** information
  - Two pieces of information at the input: observed systematic channel data and *a priori probabilities*
  - Output contains three pieces of information: observed symbols, *a priori probabilities* and extrinsic information
- The extrinsic information will be the *a priori* information for the next stage of decoding

# A typical decoder stage



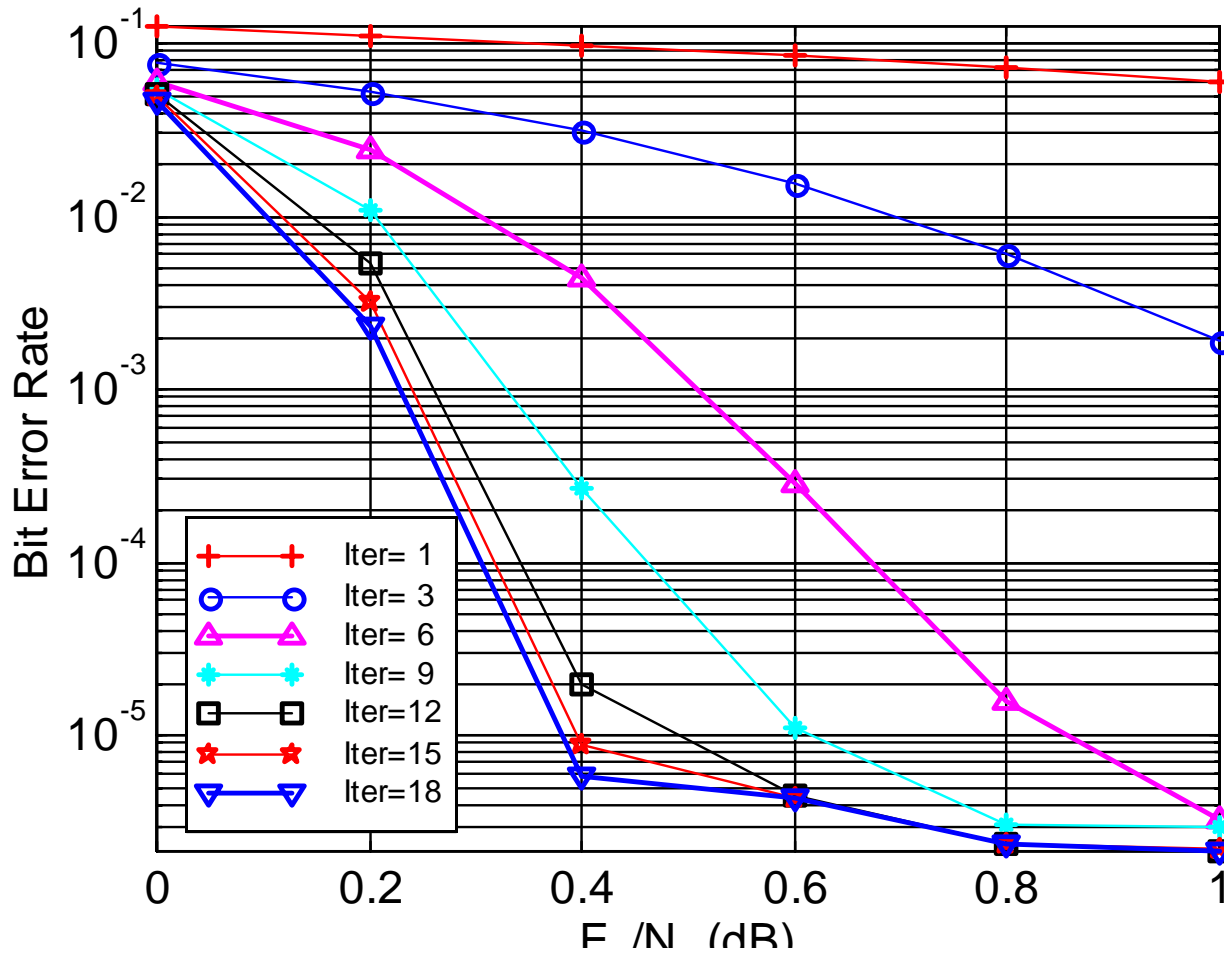


# Decoding Algorithm

---

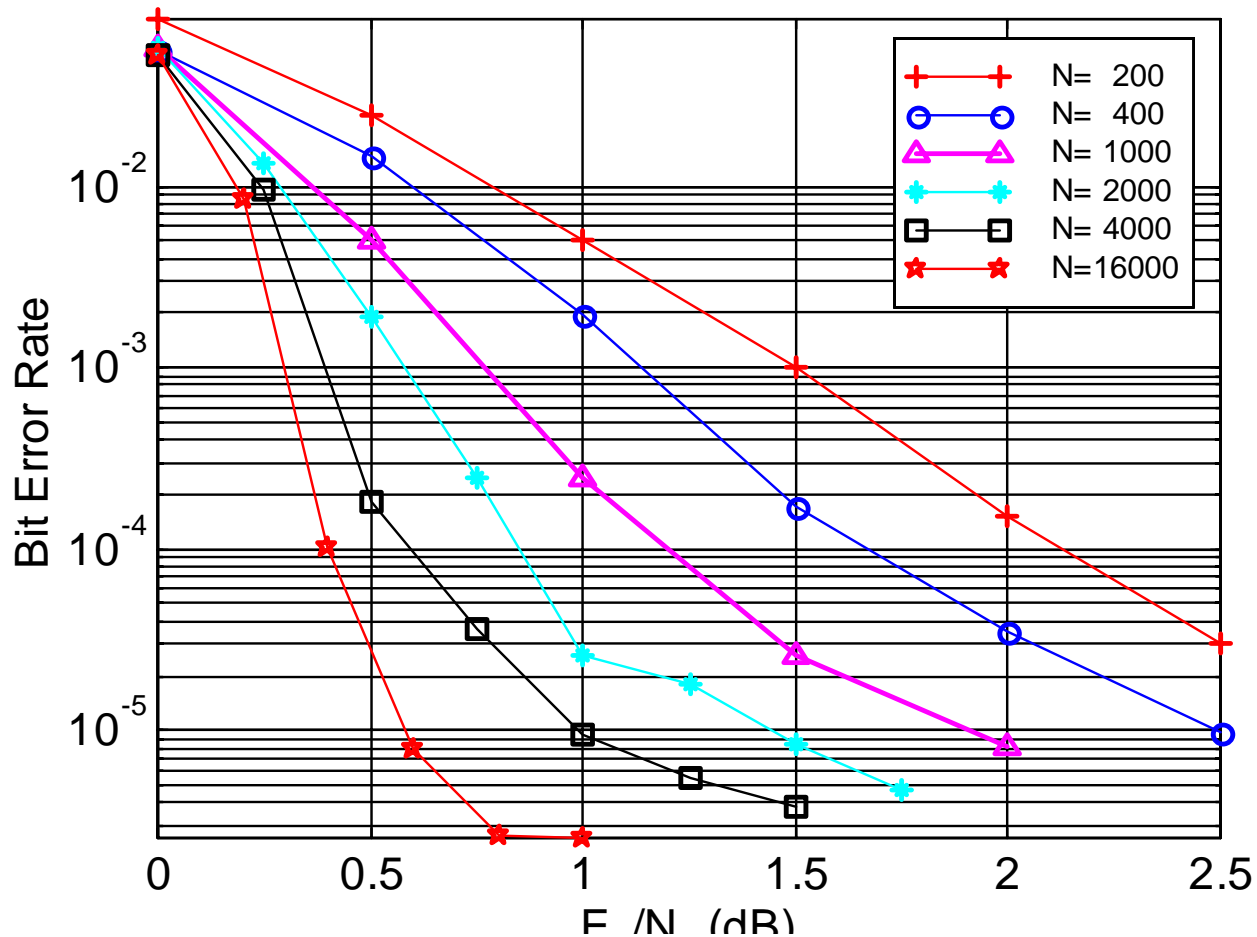
- The conventional Viterbi algorithm could use LLR as a metric (since LLRs add).
  - We would try to find the path which maximized the LLR metric.
- The conventional Viterbi decoding algorithm produces hard decision outputs
  - It cannot produce information for next decoding stage
- We need to find a decoding algorithm that accepts LLRs as input and produces them as output
  - ***Soft Input Soft Output*** (SISO)

# Parallel Concatenated Convolutional Codes: Number of Iterations



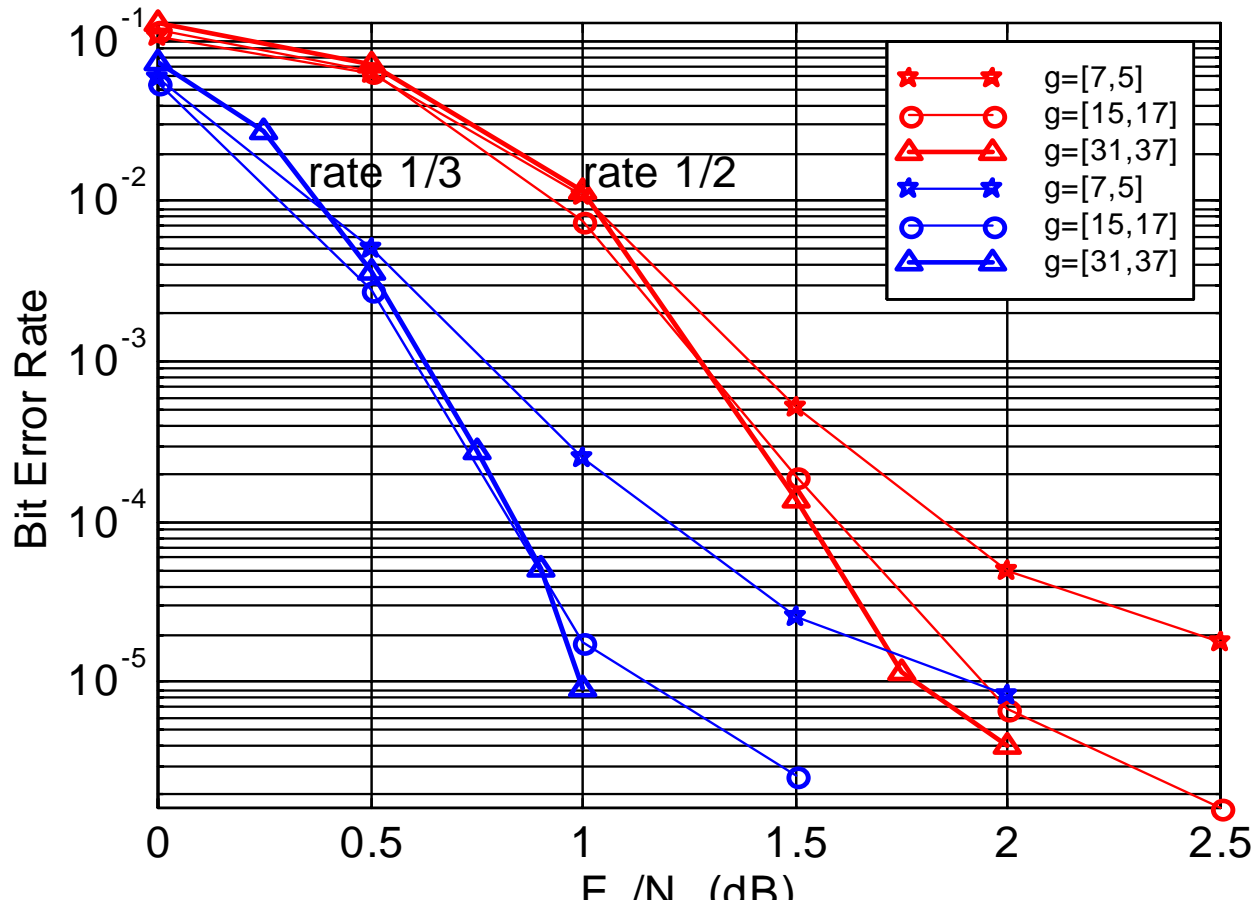
- PCCC
- $g=(7,5)_{\text{octal}}$
- Code rate 1/3
- $N=16000$
- Random interleaver.

# Parallel Concatenated Convolutional Codes: Frame Size



- PCCC
- $g=(7,5)_{\text{octal}}$
- Code rate 1/3
- 10 iterations
- Random interleaver

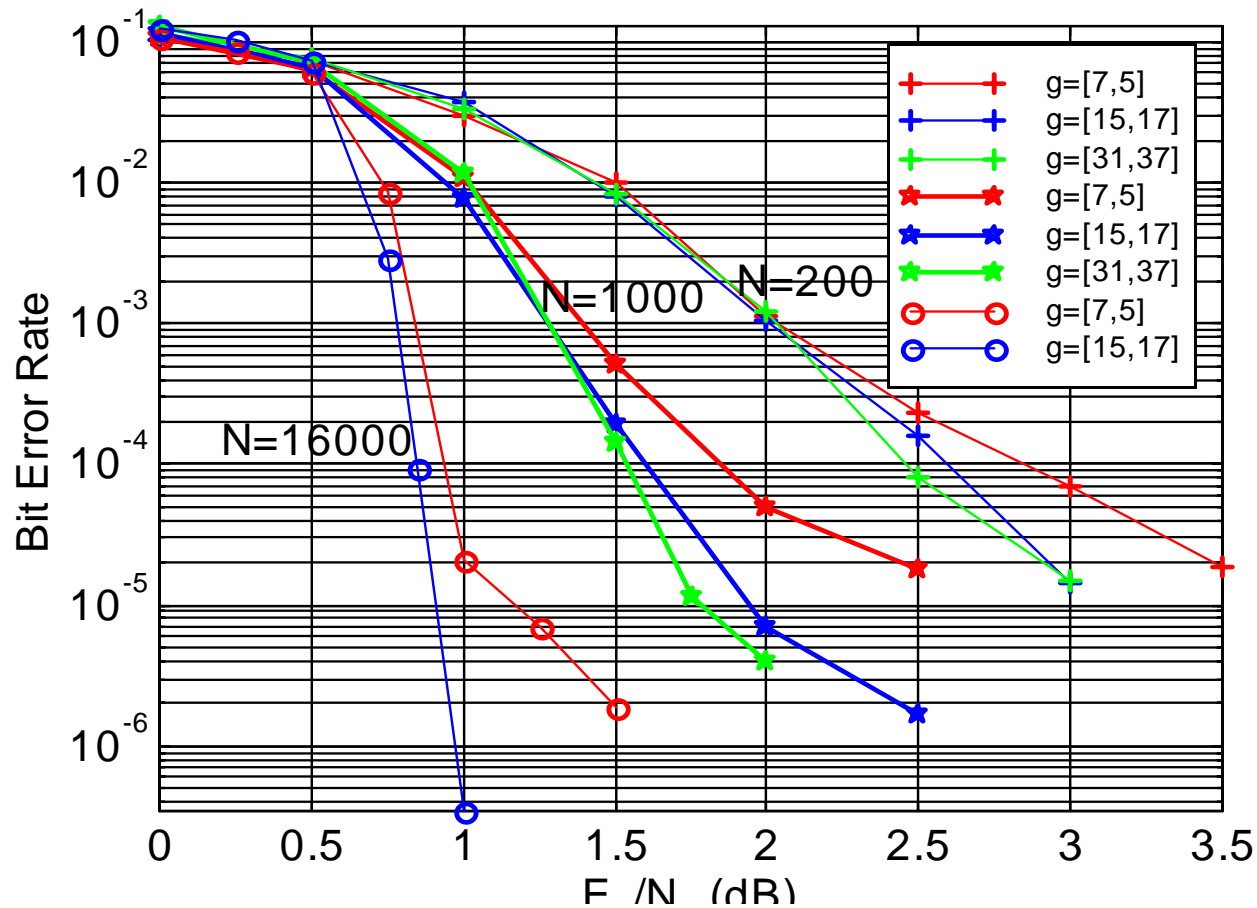
# Parallel Concatenated Convolutional Codes: Code Rate (with Various Code Generators)



- PCCC
- $N=1000$
- 10 iterations
- Random interleaver

0.7 dB more coding gain from rate 1/2 to rate 1/3

# Parallel Concatenated Convolutional Codes: Code Generator



- PCCC
- Code rate  $1/2$
- $K = 3,4,5$
- 10 iter. for  $N=200$  & 1000
- 18 iter. for  $N=16000$
- Random interleaver



# Comparisons – $10^{-4}$ BER

Code	Rate	$E_b/N_0$ Required for $10^{-4}$
Uncoded QPSK	1	8.4dB
K=4, Conv. Code, Hard Dec.	1/2	7.0dB
K=9, Conv. Code, Hard Dec.	1/2	5.25dB
K=4, Conv. Code, Soft Dec.	1/2	5.0dB
K=9, Conv. Code, Soft Dec.	1/2	2.8dB
K=3, 18 iter., N=16000, Turbo	1/2	0.8dB



# Implementation Challenges

---

- Complexity
  - iterative decoding algorithm more complex than Viterbi algorithm but shorter constraint lengths may be used
- Fixed Point DSP
  - Optimal decoding algorithm sensitive to numerical precision
  - simplified 'Soft Output Viterbi Algorithm' is less sensitive
- Adaptation
  - There are a rich set of performance tradeoffs between code rate, decoding latency, and complexity
- Fading Channels
  - Decoder requires estimate of instantaneous SNR
  - Performance is less well understood in fading

# Capacity

