



# EE 5654 - Digital Communications Spring 2005

---

Instructor: R. Michael Buehrer

Lecture #21 - Convolutional Codes: Representation  
and Encoding





# A Common Theme from Coding Theory

---

- The real issue is the complexity of the decoder.
  - For a binary code, we must match  $2^n$  possible received sequences with codewords
- Only a few practical decoding algorithms have been found:
  - Berlekamp-Massey algorithm for block codes
  - Viterbi algorithm (and similar techniques) for convolutional codes
- Code designers have focused on finding new codes that work with known algorithms



# Block Versus Convolutional Codes

---

- Block codes take  $k$  input bits and produce  $n$  output bits, where  $k$  and  $n$  are large
  - There is no data dependency between blocks
  - Useful for data communications (i.e., delay tolerant)
- Convolutional codes take a small number of input bits and produce a small number of output bits each time period
  - data passes through convolutional coder in a continuous stream
  - useful for low-latency communications
  - Note that convolutional codes can also be encoded in “frames” which are superficially similar to blocks



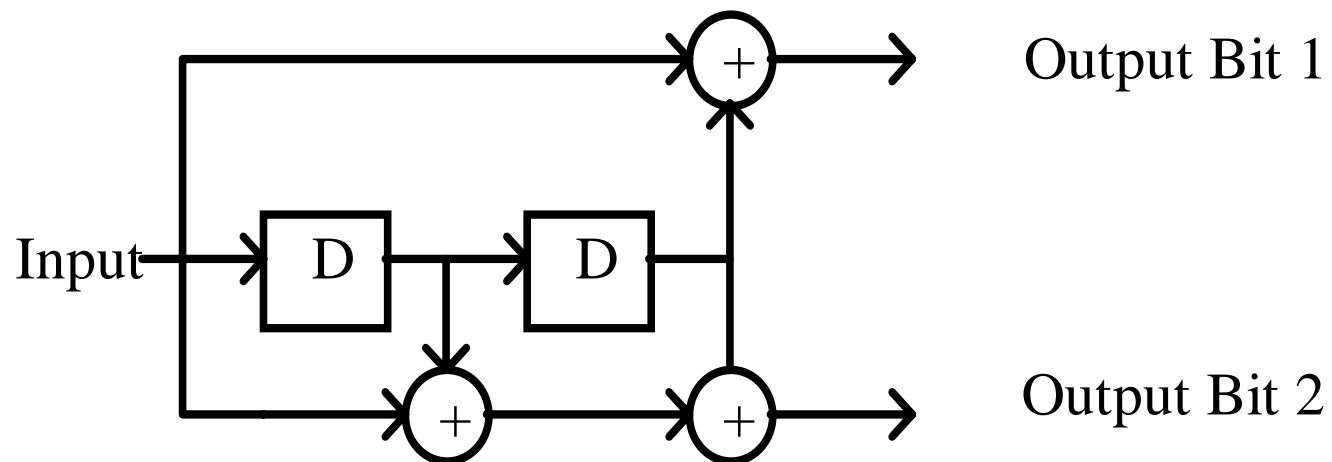
# Convolutional Codes

---

- $k$  bits are input,  $n$  bits are output
- Now  $k$  &  $n$  are very small (usually  $k=1-3$ ,  $n=2-6$ )
- Frequently, we will see that  $k=1$
- However, the output bits depend not only on current set of  $k$  input bits, **but also on past inputs.**
- The number of time slots on which output depends is called the "constraint length"  $K$ .
  - Can be thought of as the memory of the code
- Distance properties (and thus coding gain) increase with  $K$

# Example of Convolutional Code

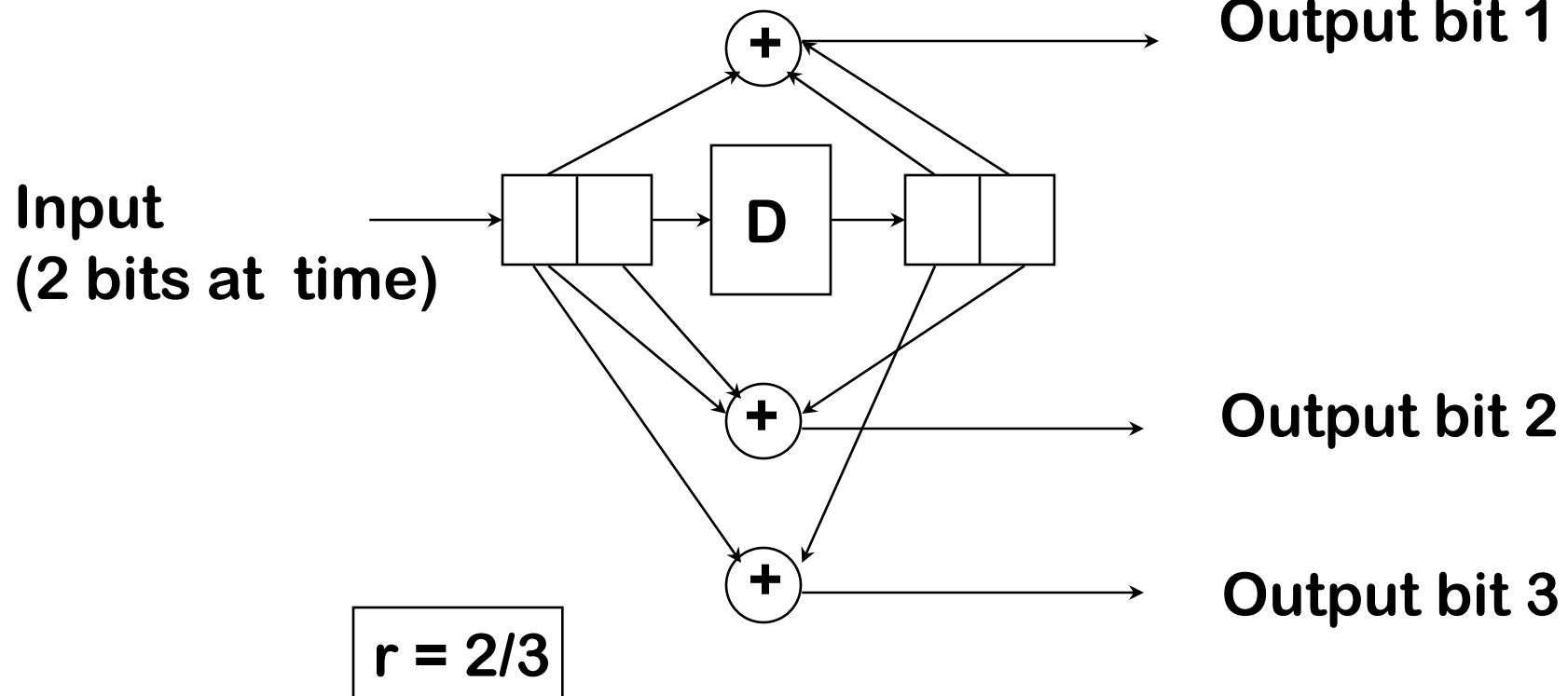
- $k=1, n=2, K=3$  convolutional code



$$r = \text{rate} = 1/2$$

# Example of Convolutional Code

- $k=2, n=3, K=2$  convolutional code





# Representation of a Convolutional Code

---

- There are in general four ways of representing a convolutional code
  - Encoder Block Diagram (two examples shown above)
  - Generator Representation
    - Represents the relationship between input (and stored) bits and output bits
  - Trellis Representation
  - State Diagram Representation



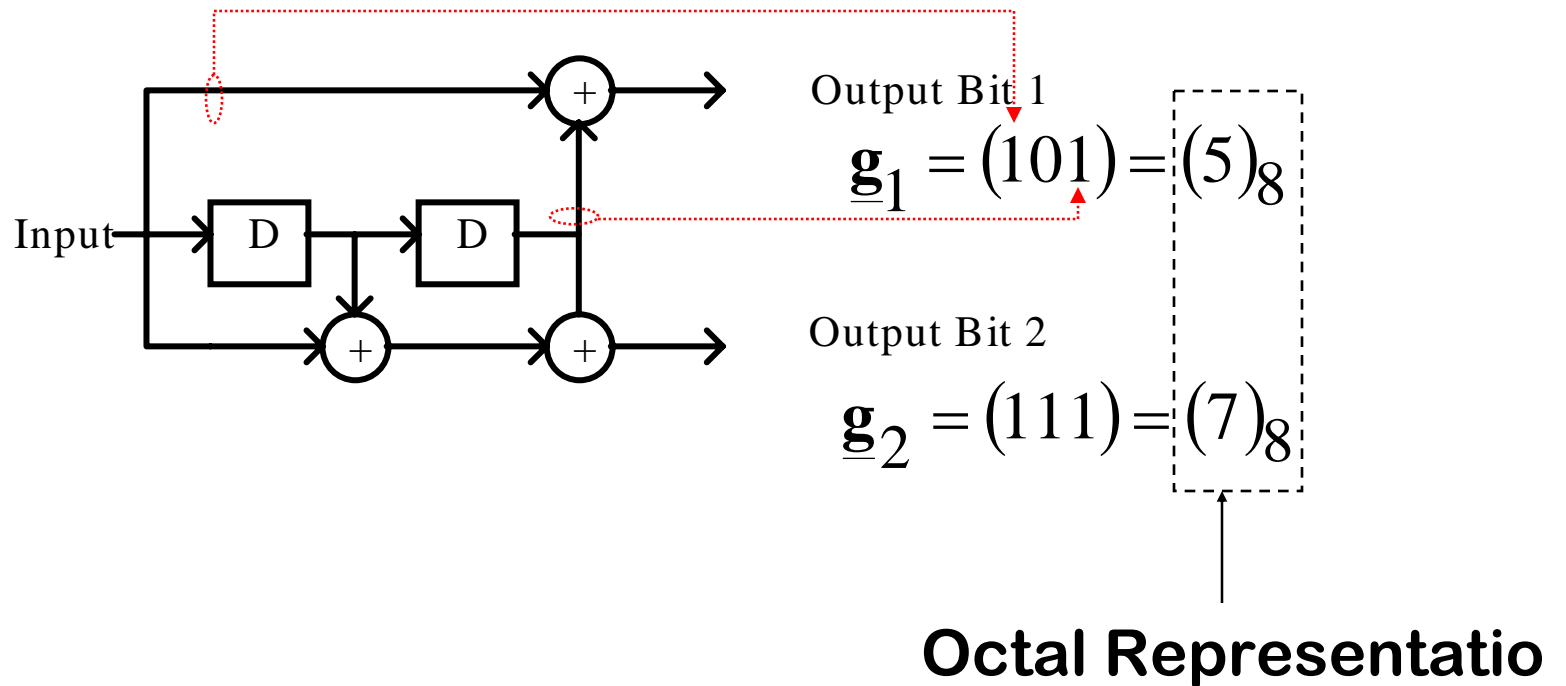
# Convolutional Code Generators

---

- There is one generator vector for each of the  $n$  output bits:
- The length of the generator vector for a rate  $r=k/n$  code with constraint length  $K$  is  $k*K$
- The bits in the generator vector from left to right represent the connections in the encoder circuit. A "1" represents a link from the shift register. A "0" represents no link.
- Encoder vectors are often given in octal representation

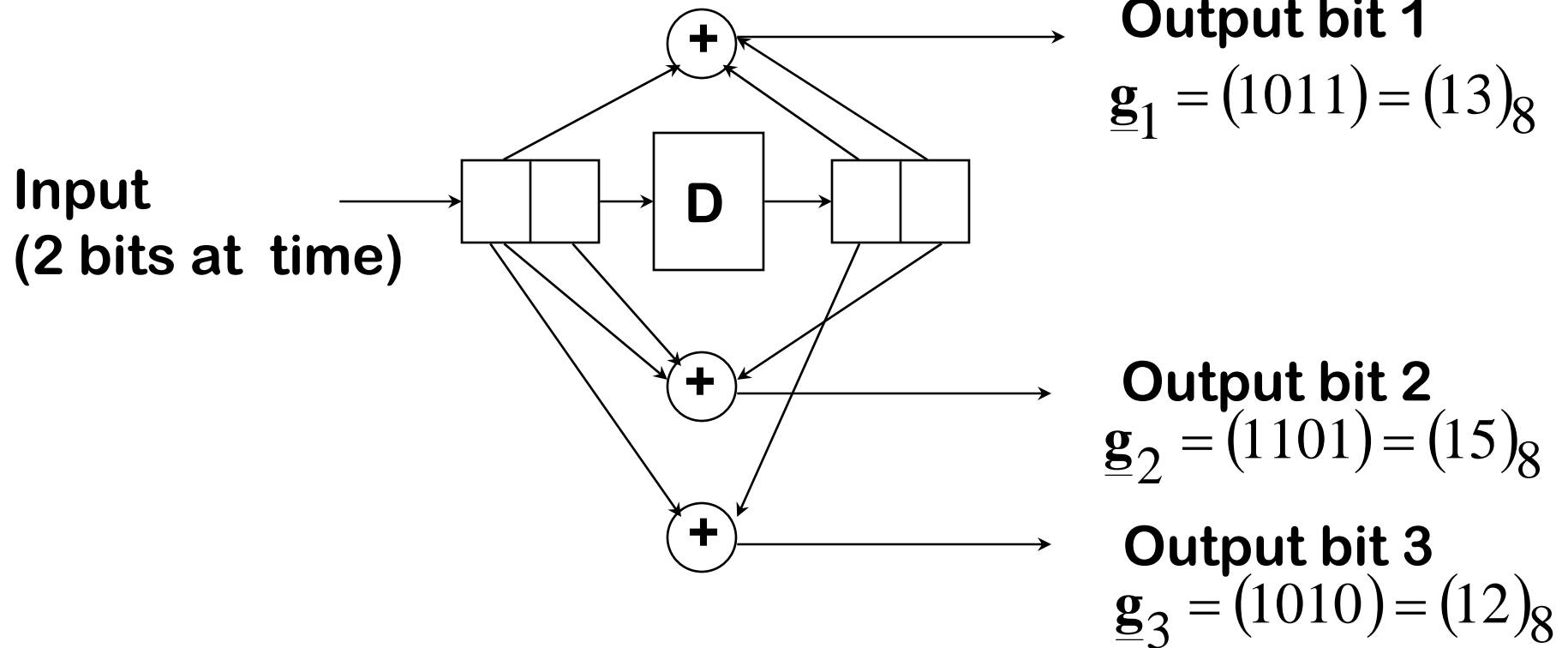
# Example of Convolutional Code

- $k=1, n=2, K=3$  convolutional code



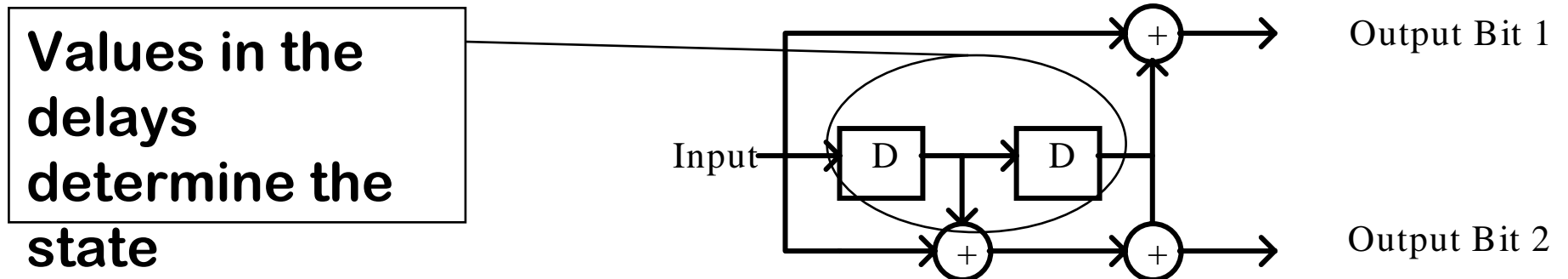
# Example of Convolutional Code

- $k=2, n=3, K=2$  convolutional code



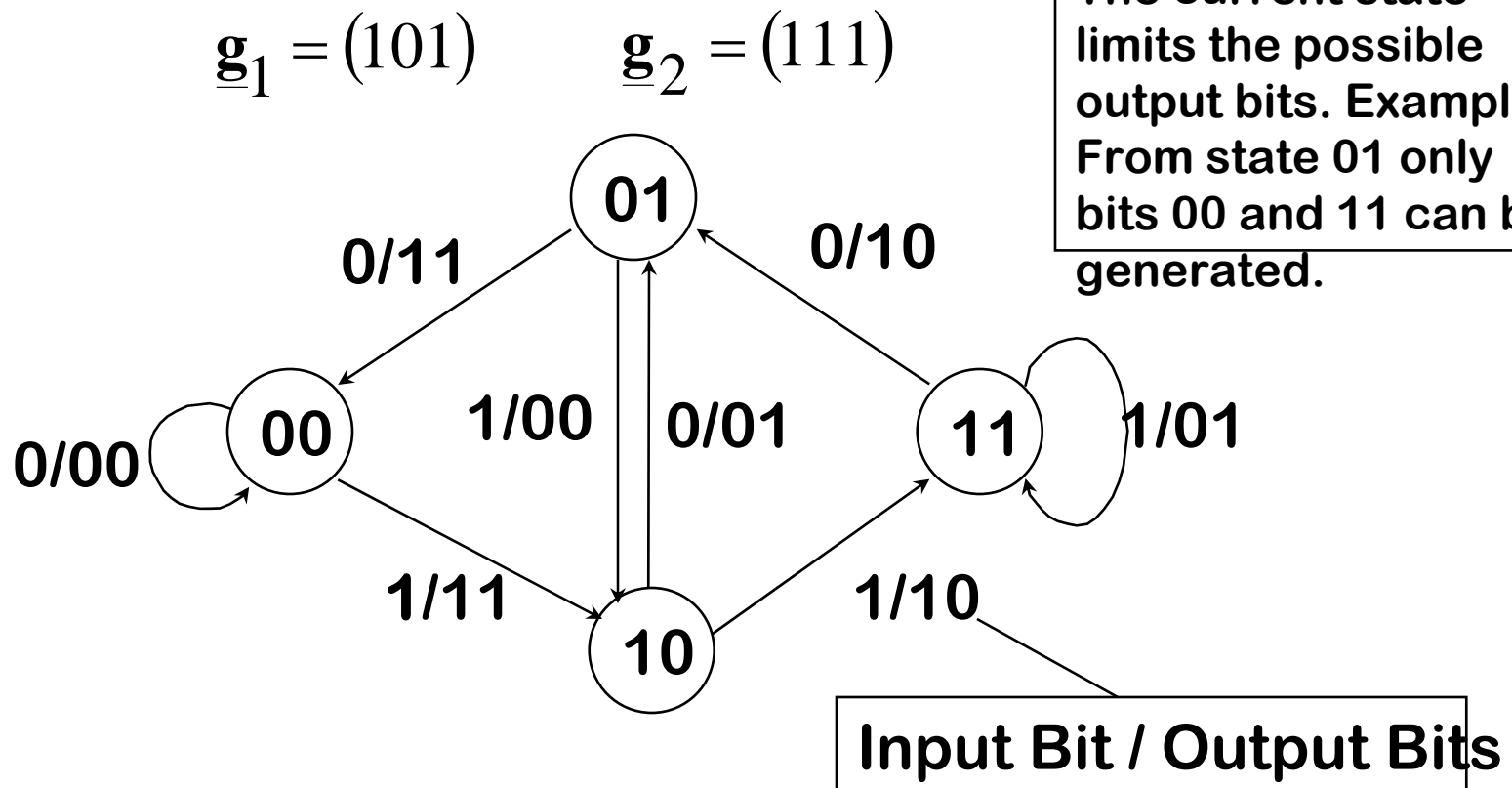
# State Diagram Representation of Convolutional Codes

- Contents of shift registers make up "state" of code:
  - Most recent input is most significant bit of state.
  - Oldest input is least significant bit of state.
  - (this convention is sometimes reverse)
- Arcs connecting states represent allowable transitions
  - Arcs are labeled with output bits transmitted during transition



# Example of State Diagram Representation of Convolutional Codes

- $k=1, n=2, K=3$  convolutional code





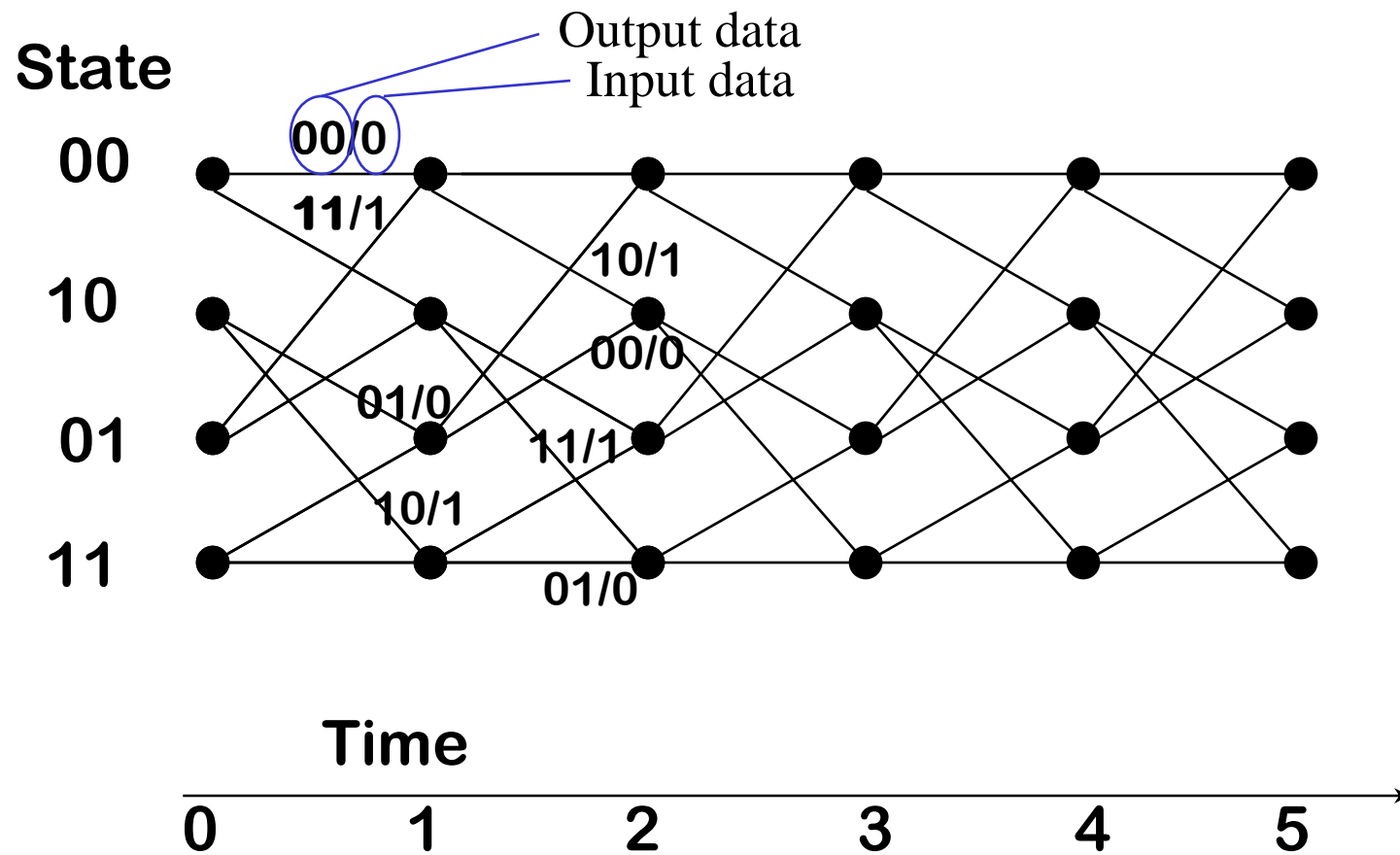
# Trellis Representation of Convolutional Code

---

- State diagram is “unfolded” as a function of time
- Time indicated by movement towards right
- Contents of shift registers make up "state" of code:
  - Most recent input is most significant bit of state.
  - Oldest input is least significant bit of state.
- Allowable transitions are denoted by connections between states
  - Transitions may be labeled with transmitted bits
  - Allowable transitions determine which output bits are possible from a given state

# Example of Trellis Diagram

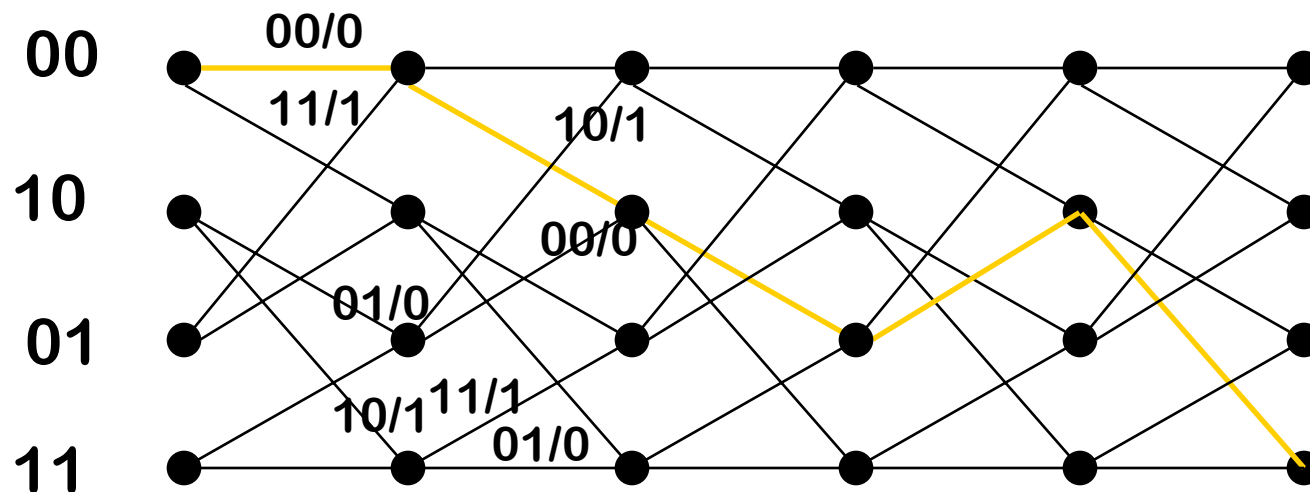
- $k=1, n=2, K=3$  convolutional code



# Encoding Example Using Trellis Representation

- $k=1, n=2, K=3$  convolutional code

State



- We begin in state 00:
- Input Data: 0      1      0      1      1      0      0
- Output:      0 0      1 1      0 1      0 0      1 0      1 0      1 1

# Distance Structure of a Convolutional Code

- The Hamming Distance between any two distinct code sequences  $\underline{c}_1 \in C$  and  $\underline{c}_2 \in C$  is the number of bits in which they differ:

$$d_H(\underline{c}_1, \underline{c}_2) = \sum_{i=-\infty}^{\infty} c_{1,i} \oplus c_{2,i}$$

- The minimum free Hamming distance  $d_{\text{free}}$  of a convolutional code is the smallest Hamming distance separating any two distinct code sequences (or paths through the trellis):

$$d_{\text{free}} = \min_{i \neq j} \{d_H(\underline{c}_i, \underline{c}_j)\}$$



# Search for good codes

---

- We would like convolutional codes with large free distance
  - Must avoid “catastrophic codes”
- Generators for best convolutional codes are generally found via computer search
  - Search is constrained to codes with regular structure
  - Search is simplified because any permutation of identical generators is equivalent
  - Search is simplified because of linearity.

# Best Rate 1/2 Codes

| $K$ | Generators (in octal) |     | $d_{\text{free}}$ |
|-----|-----------------------|-----|-------------------|
| 3   | 5                     | 7   | 5                 |
| 4   | 15                    | 17  | 6                 |
| 5   | 23                    | 35  | 7                 |
| 6   | 53                    | 75  | 8                 |
| 7   | 133                   | 171 | 10                |
| 8   | 247                   | 371 | 10                |
| 9   | 561                   | 753 | 12                |

Free distance increases with constraint length

# Best Rate 1/3 Codes

| $K$ | Generators (In Octal) |     |     | $d_{\text{free}}$ |
|-----|-----------------------|-----|-----|-------------------|
| 3   | 5                     | 7   | 7   | 8                 |
| 4   | 13                    | 15  | 17  | 10                |
| 5   | 25                    | 33  | 37  | 12                |
| 6   | 47                    | 53  | 75  | 13                |
| 7   | 133                   | 145 | 171 | 15                |
| 8   | 225                   | 331 | 367 | 16                |
| 9   | 557                   | 663 | 711 | 18                |

Free distance increases with constraint length

# Best Rate 2/3 Codes

| $K$ | Generators (In Octal) |     |     | $d_{\text{free}}$ |
|-----|-----------------------|-----|-----|-------------------|
| 2   | 17                    | 06  | 15  | 4                 |
| 3   | 27                    | 75  | 72  | 6                 |
| 4   | 236                   | 155 | 337 | 7                 |

Increasing  
free  
distance  
with  
constraint  
length



# Summary of Convolutional Codes

---

- Convolutional Codes are useful for real-time applications because they can be continuously encoded and decoded
  - Frame based encoding and decoding is also possible
- We can represent convolutional codes as generators, block diagrams, state diagrams, and trellis diagrams
- We want to design convolutional codes to maximize free distance while maintaining non-catastrophic performance



# Decoding of Convolutional Codes

---

- Let  $C_m$  be the set of allowable code sequences of length  $m$ .
  - Not all sequences in  $\{0,1\}^m$  are allowable code sequences!
- Each code sequence  $\underline{c} \in C_m$  can be represented by a unique path through the trellis diagram.
- What is the probability that the code sequence  $\underline{c}$  is sent and the binary sequence  $\underline{y}$  is received?

$$\Pr(\underline{y}|\underline{c}) = p^{d_H(\underline{y},\underline{c})} \cdot (1-p)^{m-d_H(\underline{y},\underline{c})}$$

where  $p$  is the probability of bit error resulting from modulation scheme



# Decoding Rule for Convolutional Codes

---

- Maximum Likelihood Decoding Rule:

$$\max_{\underline{c} \in C_m} \{\Pr(\underline{y}|\underline{c})\} \Rightarrow \min_{\underline{c} \in C_m} \{d_H(\underline{y}, \underline{c})\}$$

- Choose the code sequence through the trellis which has the smallest Hamming distance to the received sequence!
- Note that this is identical to the rule for Maximum Likelihood demodulation
  - Simply replaces  $k$ -dimension vector which represents the symbol with  $m$ -dimensional vector representing the received sequence of bits



# The Viterbi Algorithm

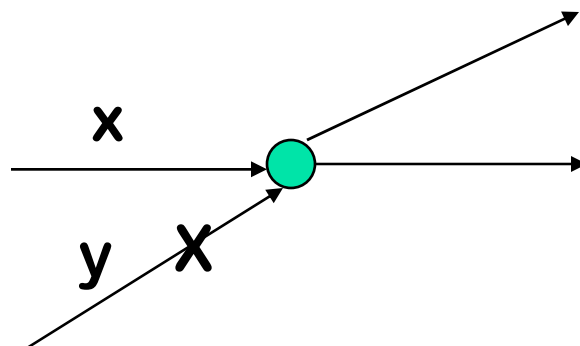
---

- The Viterbi Algorithm (Viterbi, 1967) is a clever way of implementing Maximum Likelihood decoding.
  - Computer Scientists will recognize the Viterbi Algorithm as an example of a CS technique called "Dynamic Programming."
- Reference: G. D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE*, 1973.
- Chips are available from many manufacturers which implement the Viterbi Algorithm for  $K < 10$ .
- Can be used for either hard or soft decision decoding
  - We consider hard decision decoding initially

# Basic Idea of Viterbi Algorithm

- There are  $2^{rm}$  code sequences in  $C_m$
- This number of sequences approaches infinity as  $m$  becomes large
- Instead of searching through all possible sequences to determine the closest vector, we find the best code sequence "one stage at a time"
- Consider two possible code sequences (paths in the trellis)  $\mathbf{x}$  and  $\mathbf{y}$  which merge at a particular node:

Two paths  $\mathbf{x}$  and  $\mathbf{y}$  in the trellis represent valid possible code vectors



If the distance between  $\mathbf{x}$  and the received vector  $\mathbf{r}$  (up until this node) is smaller than the distance between  $\mathbf{y}$  and  $\mathbf{r}$ ,  $\mathbf{y}$  cannot be part of the shortest path



# Analogy

---

- Suppose I wanted to determine the fastest driving route between San Francisco and Miami. To determine this, I sent 500 people driving the exact same speed from San Francisco on 500 different predetermined routes.
- Suppose along the way two people (Fred and Alice) arrive in Dallas in the midst of their trips.
- Comparing notes, Fred sees that his cumulative driving time has been 20 hours and while Alice's cumulative driving time has been 19.5 hours.
- Further they note that the remainder of their respective paths are identical.
- Can Fred possibly have the shortest driving route ?
  - If not, perhaps we can stop Fred now and help save the ozone layer.



# Implementation of Viterbi Decoder

---

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