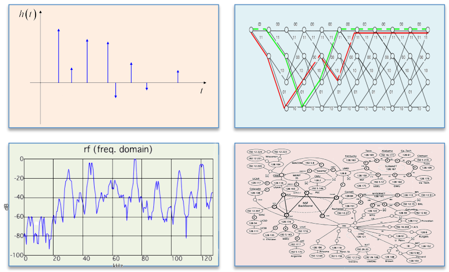


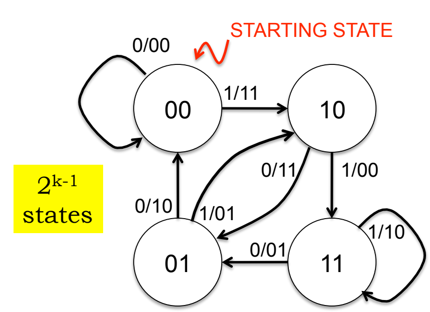
INTRODUCTION TO EECS II

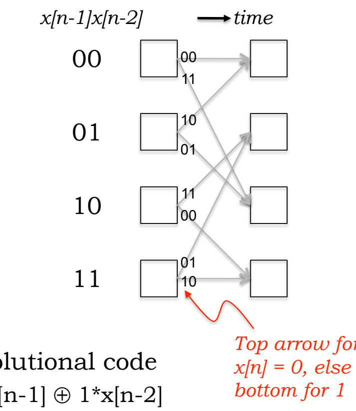
DIGITAL COMMUNICATION SYSTEMS

6.02 Spring 2009
Lecture #10

- state machines & trellises
- path and branch metrics
- Viterbi algorithm: add-compare-select
- hard decisions vs. soft decisions

State Machines & Trellises





- Example: k=3, rate ½ convolutional code
  - $G_0 = 111$: $p_0 = 1{*}x[n] \oplus 1{*}x[n-1] \oplus 1{*}x[n-2]$
  - $G_1 = 110$: $p_1 = 1{*}x[n] \oplus 1{*}x[n-1] \oplus 0{*}x[n-2]$
- States labeled with *x[n-1] x[n-2]*
- Arcs labeled with $x[n]/p_0p_1$

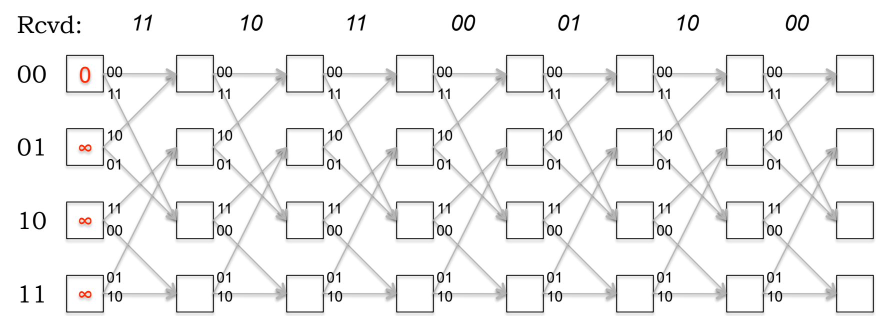
Example

- Using k=3, rate ½ code from earlier slides
- Received: 11101100011000
- Some errors have occurred…
- What's the 4-bit message?
- Look for message whose xmit bits are closest to rcvd bits

Msg	Xmit	Rcvd	d
0000	00000000000000		7
0001	00000011111000		8
0010	00001111100000		8
0011	00001101011000		4
0100	00111110000000		6
0101	00111101111000		5
0110	00110100100000		7
0111	00110010011000		6
1000	11111000000000	11101100011000	4
1001	11111011111000		5
1010	11110111100000		7
1011	11110100011000		2
1100	11000110000000		5
1101	11000101111000		4
1110	11001001100000		6
1111	11001010011000		3

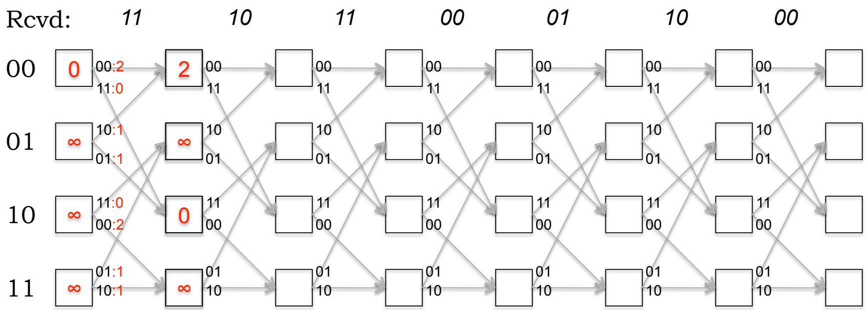
Most likely: 1011

Finding the Most-likely Path



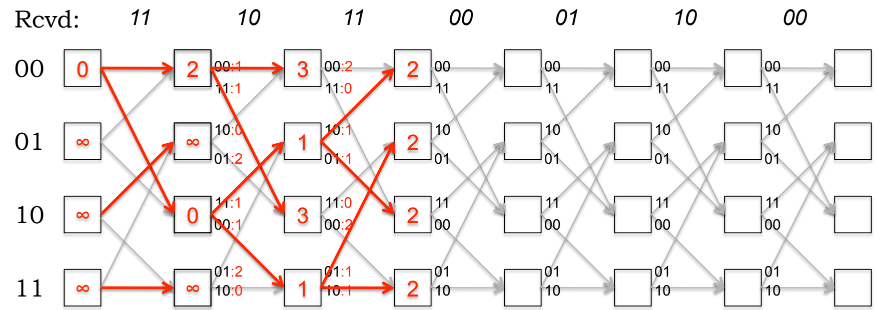
- Path metric: number of errors on most-likely path to given state (min of all paths leading to state)
- Branch metric: for each arrow, the Hamming distance between received parity and expected parity

Viterbi Algorithm



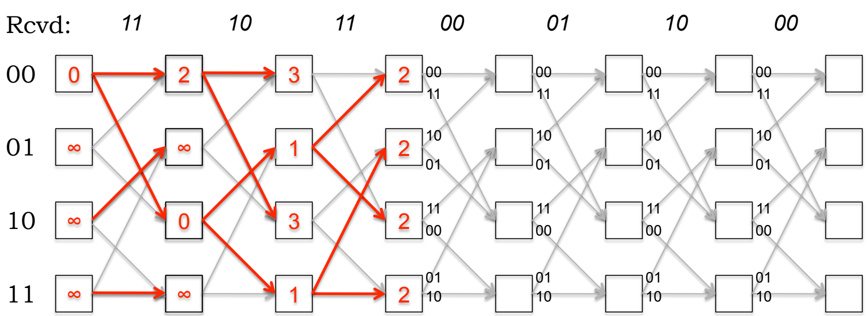
- Compute branch metrics for next set of parity bits
- Compute path metric for next column
  - *add* branch metric to path metric for old state
  - *compare* sums for paths arriving at new state
  - *select* path with smallest value (fewest errors, most likely)

Example (cont'd.)



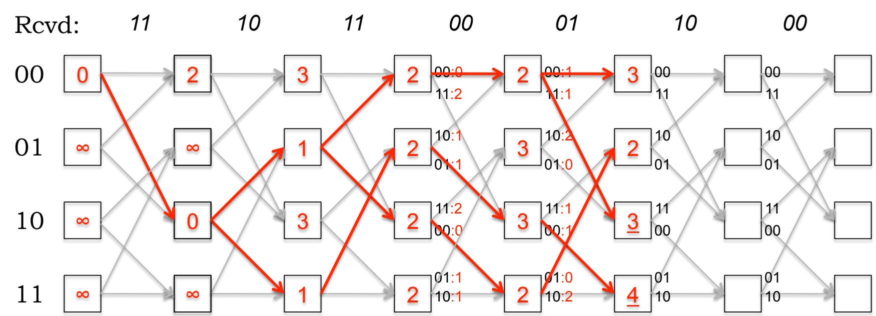
- After receiving 3 pairs of parity bits we can see that all ending states are equally likely
- Power of convolutional code: use future information to constrain choices about most likely events in the past

Survivor Paths



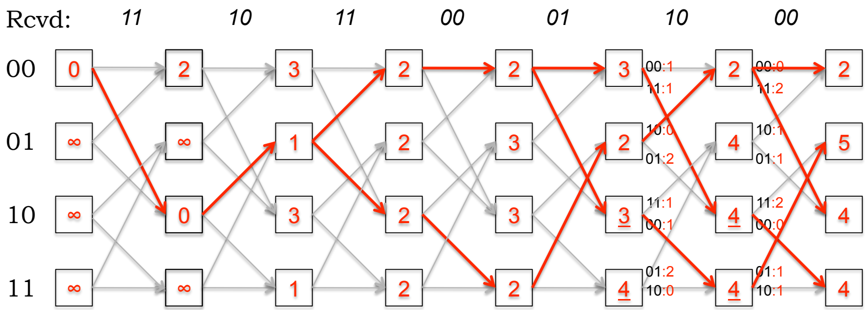
- Notice that some paths don't continue past a certain state
  - Will not participate in finding most-likely path: eliminate
  - Remaining paths are called *survivor paths*
  - When there's only one path: we've got a message bit!

Example (cont'd.)



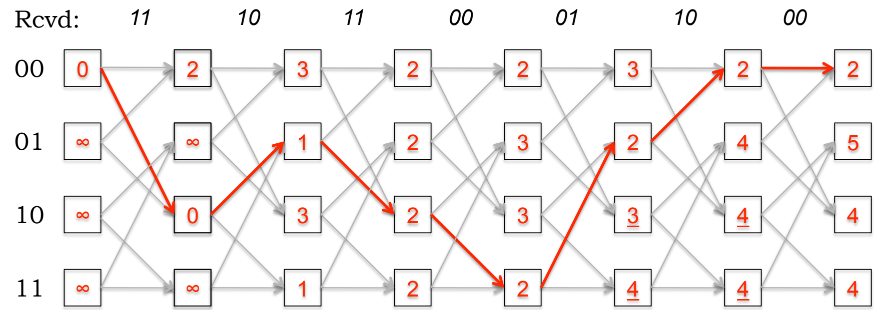
- When there are "ties" (sum of metrics are the same)
  - Make an arbitrary choice about incoming path
  - If state is not on most-likely path: choice doesn't matter
  - If state is on most-likely path: choice may matter and error correction has failed *(mark state with underline to tell)*

Example (cont'd.)



- When we reach end of received parity bits
  - Each state's path metric indicates how many errors have happened on most-likely path to state
  - Most-likely final state has smallest path metric
  - Ties means end of message uncertain (but survivor paths may merge to a unique path earlier in message)

Traceback



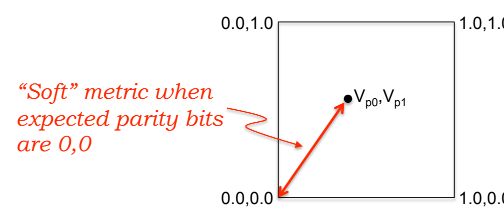
- Use most-likely path to determine message bits
  - Trace back through path: message in reverse order
  - Message bit determined by high-order bit of each state (remember that came from message bit when encoding)
  - Message in example: 1011000 (w/ 2 transmission errors)

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 is the Hamming distance between the digitized received voltages and the expected parity bits
  - This is called *hard-decision Viterbi decoding*
- 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?

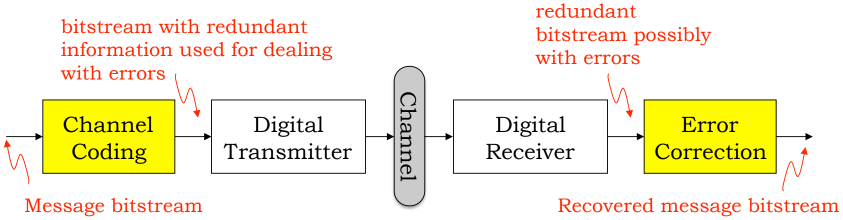
Soft Decisions

- Let's limit the received voltage range to [0.0,1.0]
  - $V_{eff} = \max(0.0, \min(1.0, V_{received}))$
  - Voltages outside this range are "good" 0's or 1's
- Define our "soft" branch metric as the Euclidian distance between received V_{eff} and expected voltages



- Soft-decision decoder chooses path that minimizes sum of Euclidian distances between received and expected voltages
  - Different branch metric but otherwise the same recipe

Channel Coding Summary



- Add redundant info to allow error detection/correction
  - CRC to detect error-transmission (our safety net for catching undiscovered or uncorrectable errors)
  - Block codes: multiple parity bits, RC codes: oversampled polynomials
  - Convolutional codes: continuous streams of parity bits
- Error correction
  - Block codes: use parity errors to triangulate which bits are in error
  - RS codes: use subsets of bits to vote for message, majority rules!
  - Convolutional codes: use Viterbi algorithm to find most-likely message