
VITERBI DECODER PROCESSING

Viterbi References

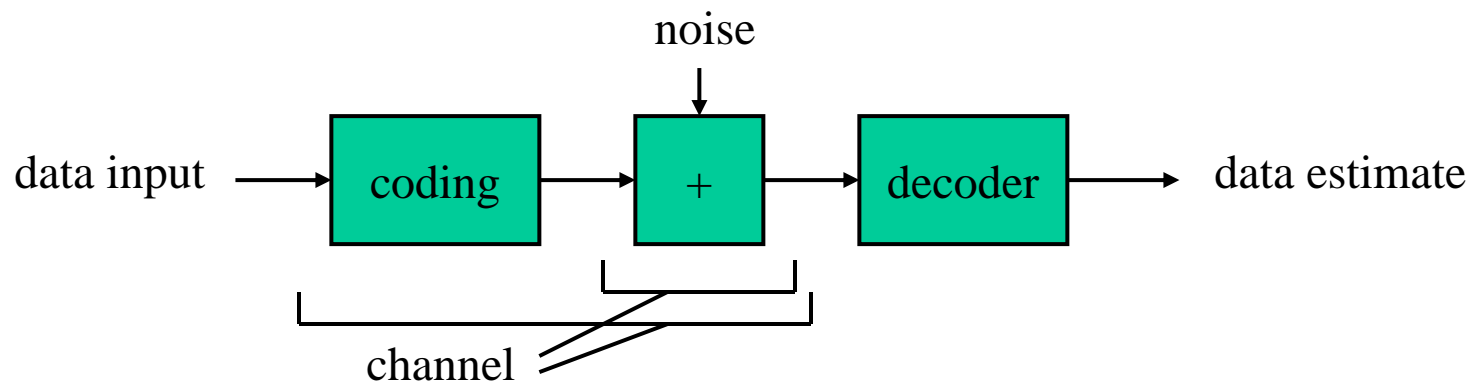
- Suggested references:
 - J. G. Proakis and M. Salehi, *Fundamentals of Communication Systems*, Prentice Hall, 2005. [coding and viterbi]
 - A. J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,” *IEEE Trans. on Information Theory*, April 1967.
 - G. D. Forney, “Maximum-Likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference,” *IEEE Trans. on Information Theory*, May 1972.
 - H.-L. Lou, “Implementing the Viterbi Algorithm,” *IEEE Signal Processing Magazine*, Sept. 1995.
 - P. J. Black, “Algorithms and Architectures for High Speed Viterbi Decoding,” Ph.D. dissertation, March 1993.

Viterbi Algorithm

- Widely used in communication systems to decode a data sequence that has been encoded by a “finite-state” process
 - Ex: ethernet receiver
 - Ex: hard disk read electronics
- A very common demanding DSP task
 - Ex: viterbi building blocks are in every serious DSP benchmarking suite
 - Ex: The Texas Instruments C64x DSP processor has a dedicated on-chip viterbi decoder. The viterbi decoder and a “turbo decoder” occupy about 8% of the chip’s area

Viterbi Algorithm

- Involves coding data, adding noise, and decoding
 - Deliberate encoding: convolutional or trellis codes
 - Unintentional encoding: intersymbol interference
- Output is an *estimate* of the original data
- Viterbi algorithm is optimal in the maximum likelihood sense—it finds the input that is most likely, given the observed channel output

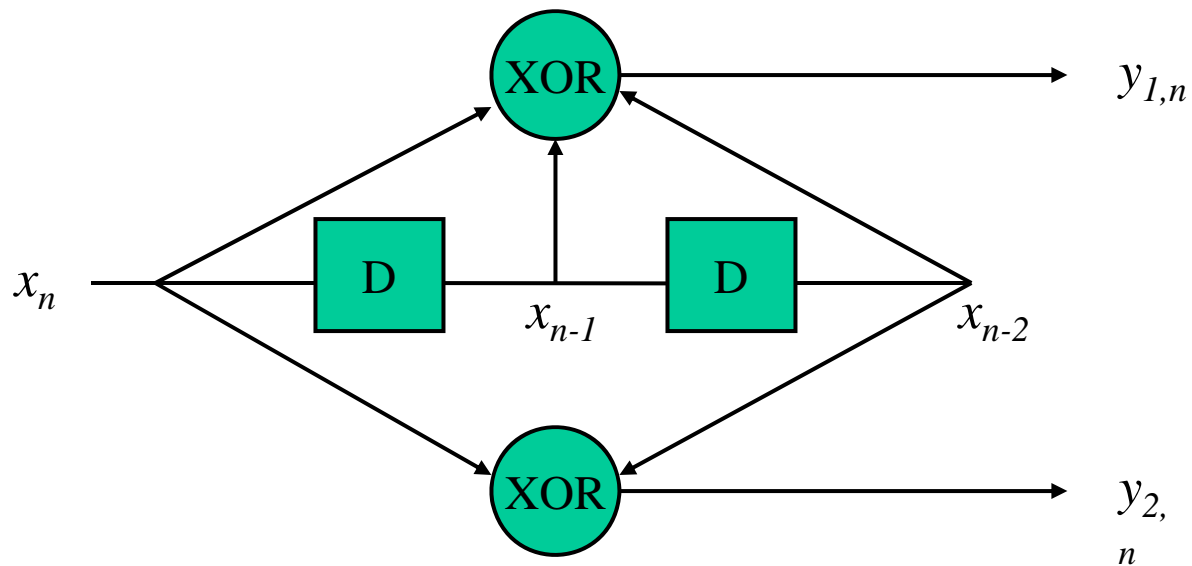


Convolutional Coding Example

- Encoding is a process which adds redundancy to data to reduce the probability of errors or increase the level of acceptable noise in the channel
- The constraint length (commonly called k) is a measure of the complexity of the code
 - Ex: $k = 7 \rightarrow 2^{k-1}$ states = 2^6 states = 64 states (in Wi-Fi)
 - Ex: $k = 9 \rightarrow 256$ states (a challenging design)
- Ex: a 4-state, Rate = $1/2$ convolutional encoder
 - input data x_n
 - encoded symbols y_1 and y_2
 - $y_{1,n} = x_n \text{ XOR } x_{n-1} \text{ XOR } x_{n-2}$
 - $y_{2,n} = x_n \text{ XOR } x_{n-2}$
 - Two data bits output by decoder for each input bit (rate = $1/2$)
 - “Double rate” output bits typically sent serially into one “channel”

Convolutional Coding Example

- Present state : $\underline{x_{n-2}} \quad \underline{x_{n-1}}$
- Next state: $\underline{x_{n-1}} \quad \underline{x_n \text{ (input)}}$



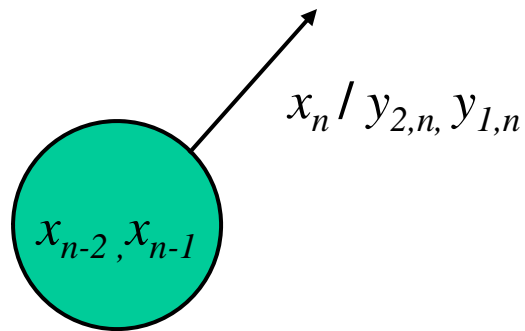
Convolutional Coding Example

- 8-entry truth table
 - inputs: three x 's
 - outputs: two y 's

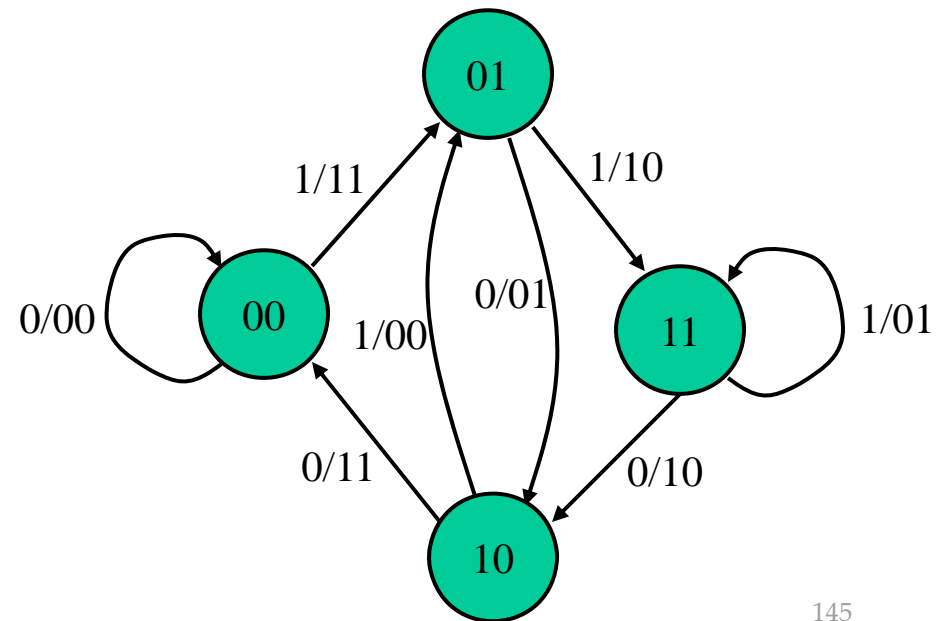
x_n, x_{n-1}, x_{n-2}	$y_{1,n}$	$y_{2,n}$
000	0	0
001	1	1
010	1	0
011	0	1
100	1	1
101	0	0
110	0	1
111	1	0

Convolutional Coding Example

- Each state can transition to only 2 other states
 - These two states are reached by either $x_n = 0$ or 1
- Next state found by: x_{n-1} x_n (input)

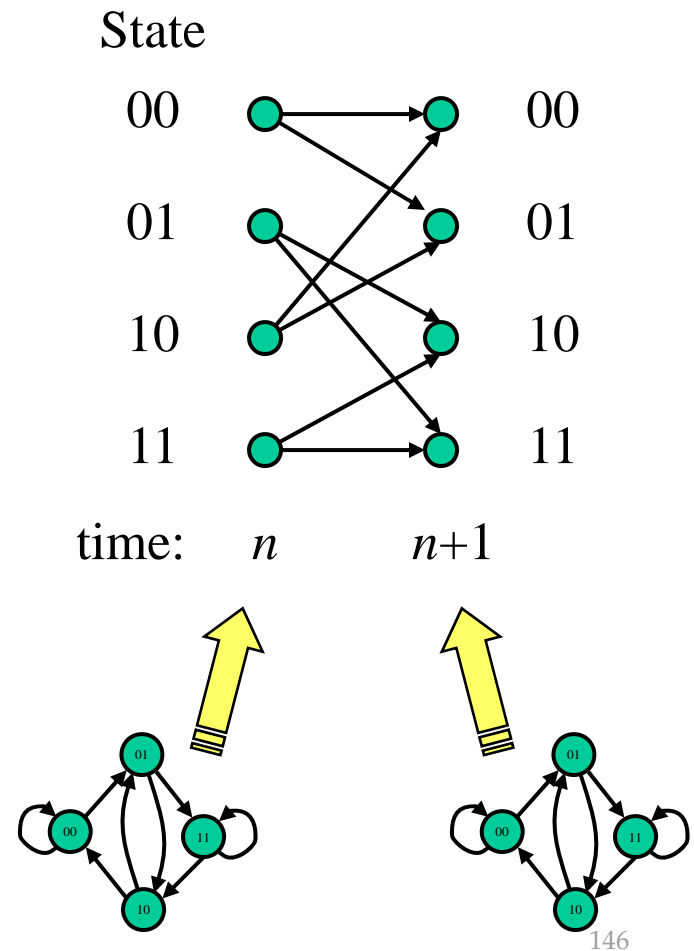


Notation used in State Graph



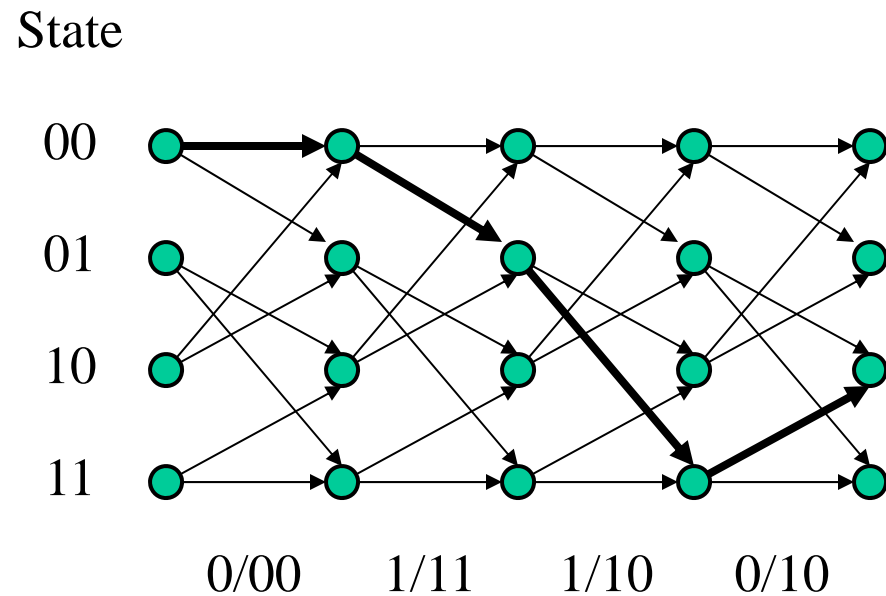
Trellis

- The *trellis* is a time indexed version of the state diagram
- Each state transition (input bit into the encoder) corresponds to a forward step in the trellis



Trellis

- Given a starting state, every possible input sequence corresponds to a unique path through the trellis
- Example:
 - Assume start in state 00
 - Inputs $x_n = 0, 1, 1, 0$
 - Outputs =
{0,0},
{1,1},
{1,0},
{1,0}



Viterbi Decoding

- The Viterbi decoder calculates a semi-brute-force estimate of the likelihood for each path through the trellis
- Key point: Once the estimates for all states in a step/iteration of the trellis have been calculated, the probabilities for all previous steps/iterations can be discarded; only the most likely entry to a state must be remembered
- It comprises four basic steps:
 1. Calculate the trellis
 - a) Weight the trellis branches by calculating branch metrics
 - b) Compute the minimum weight path to time $n+1$ in terms of the minimum weight path to time n . Retain step decisions. Uses *add-compare-select (ACS)* algorithm.
 2. Find the last state of the minimum weight path.
 3. Find the entire minimum weight path. Also called survivor path decode or *traceback*.
 4. Reorder bits into correct forward ordering.

1. Calculating the Trellis

- Each branch is assigned a weight, called a *branch metric*
- The branch metric is a measure of the likelihood of the transition given the noisy observations
- Likelihood of a transition is given by an appropriate measure of the “distance” between an ideal encoder output and the actual received signal
- The overall goal is to find the minimum weight path (often called *shortest path*) through the trellis

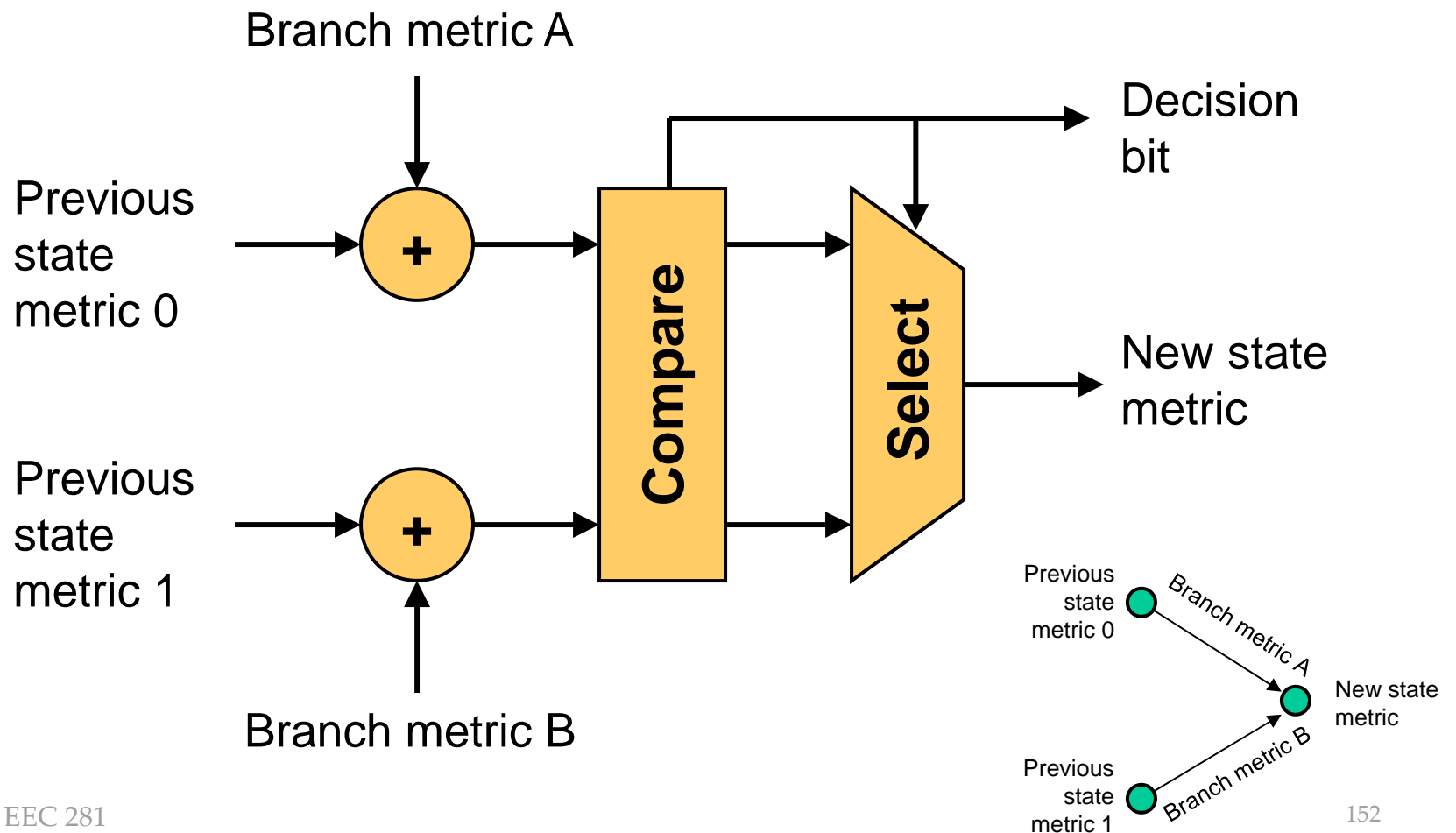
1a) Branch Metric Calculations

- More likely transitions receive *lower* weights
- Example:
 - Receive: {0.13, 0.89}
 - Branch with {0,1} outputs receives a lower weight (because it was the more likely transmitted pair) than the {1,0} branch
- A very simple implementation would add just the differences between received values, for example:
 - Starting state: 0,1 Notice from State Graph 0/01 and 1/10 are only options
 - Receive: {0.13, 0.89}
 - Branch {0,0} Impossible, not present on State Graph
 - Branch {0,1} Branch metric = $|0-0.13| + |1-0.89| = 0.24$
 - Branch {1,0} Branch metric = $|1-0.13| + |0-0.89| = 1.76$
 - Branch {1,1} Impossible, not present on State Graph
- Higher performance decoders would use more complex mapping functions between received values and the branch metric weights

1b) ACS

- Goal is to find the most likely entry path into a state
- ADD previous state metric to current branch metric
 - for first branch
 - for second branch
- COMPARE which incoming branch produces the lower metric (higher probability)
- SELECT the minimum and save the branch
- Fully parallel viterbi decoders use many ACS datapaths to calculate an entire trellis state update in one cycle

ACS Hardware

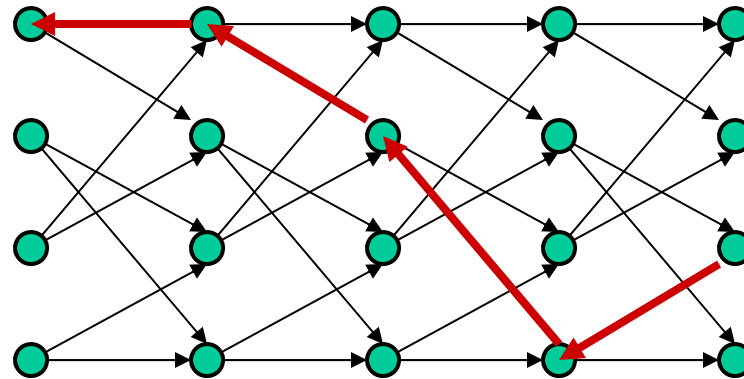


2. Finding The Most Likely Path

- After all data have been processed, we must find the most likely path through the trellis
- Done by searching through the final “column” of states and selecting the most likely (lowest weight)
- Operation done infrequently compared to other calculations

3. Traceback

- The goal is to read out the best estimate for all decoded bits
- The process begins at the most likely state and it follows this procedure for each state in the trellis
 - 1) Read stored decision bit from ACS
 - 2) Output this “decoded bit”
 - 3) Use the decision bit to calculate the previous state in the trellis
 - 4) <repeat>



4. Reorder Output Bits

- Bits output from the traceback operation are in reverse order (last bit out corresponds to first received)
- Need logic and memory to buffer and reverse order
- Fundamental to the algorithm, cannot be avoided
- Summary
 - Viterbi is data and memory intensive
 - *Forward* ACS
 - *Reverse* traceback
 - *Forward* re-ordering