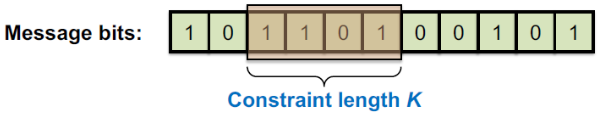
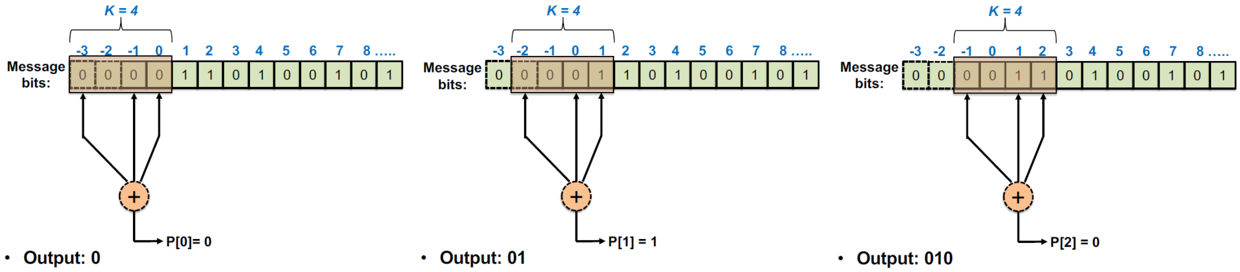
Example – Viterbi Decoding

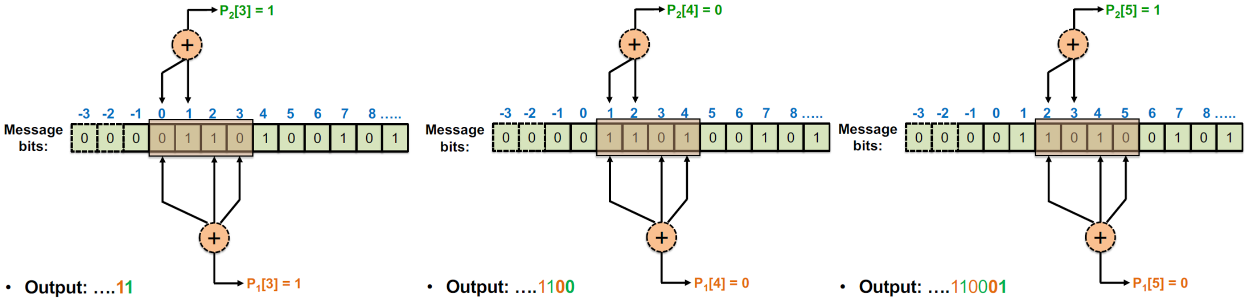
Tiniest info on convolutional encoding:
Instead of sending message bits, we send parity bits only with a fixed sliding window of length K (constraint length of the code). Each message bit is "spread across" K bits of the output parity bit sequence



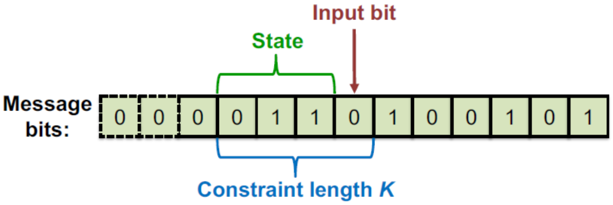
Sliding parity bit calculation:



Multiple parity bits (multiple generators):



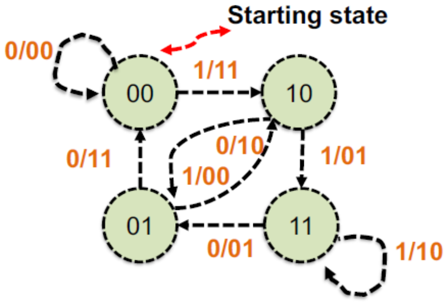
Input bit and $K-1$ bits of current state determine state on next clock cycle. There are 2^{K-1} possible states:

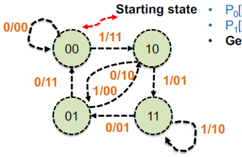


Code rate $= \frac{1}{\text{\# of generators}} \rightarrow$ more generators improves bit-error correction but decreases rate of the code.

Example: $K = 3$, code rate $= \frac{1}{2}$, message x

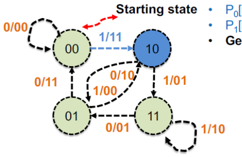
- There are $2^{K-1} = 4$ states
- States labeled with $(x[n-1], x[n-2])$
- Arcs labeled with $x[n]/p_0[n]p_1[n]$
- Generators: $g_0 = 111, g_1 = 101$
- $x = 101100$





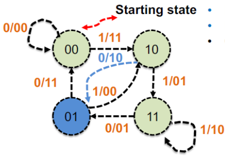
- P_0[n] = (1*x[n] + 1*x[n-1] + 1*x[n-2]) mod 2
- P_1[n] = (1*x[n] + 0*x[n-1] + 1*x[n-2]) mod 2
- **Generators**: g_0 = 111, g_1 = 101

- **msg** = 101100
- **Transmit**:



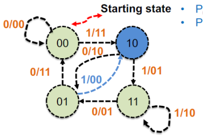
- P_0[n] = 1*1 + 1*0 + 1*0 mod 2
- P_1[n] = 1*1 + 0*0 + 1*0 mod 2
- **Generators**: g_0 = 111, g_1 = 101

- **msg** = **1**01100
- **Transmit**: **11**



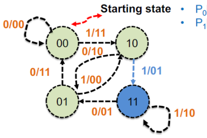
- P_0[n] = 1*0 + 1*1 + 1*0 mod 2
- P_1[n] = 1*0 + 0*1 + 1*0 mod 2
- **Generators**: g_0 = 111, g_1 = 101

- **msg** = 1**0**1100
- **Transmit**: 11 **10**



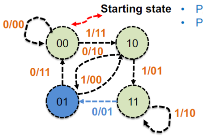
- P_0[n] = 1*1 + 1*0 + 1*1 mod 2
- P_1[n] = 1*1 + 0*0 + 1*1 mod 2

- **msg** = 10**1**100
- **Transmit**: 11 10 **00**



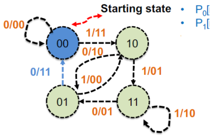
- P_0[n] = 1*1 + 1*1 + 1*0
- P_1[n] = 1*1 + 0*1 + 1*0

- **msg** = 101**1**00
- **Transmit**: 11 10 00 **01**



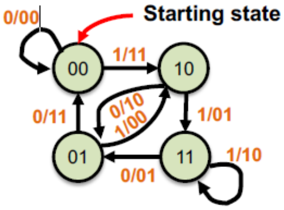
- P_0[n] = 1*0 + 1*1 + 1*1
- P_1[n] = 1*0 + 0*1 + 1*1

- **msg** = 1011**0**0
- **Transmit**: 11 10 00 01 **01**



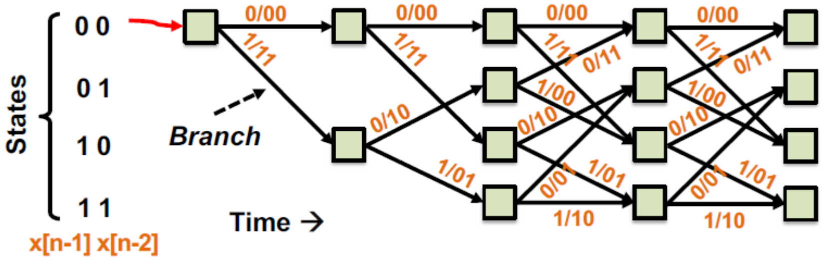
- P_0[n] = 1*0 + 1*0 + 1*1
- P_1[n] = 1*0 + 0*0 + 1*1

- **msg** = 10110**0**
- **Transmit**: 11 10 00 01 01 **11**

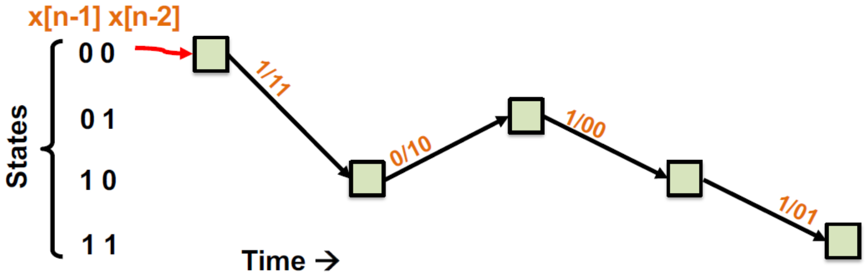


- **Vertically,** lists encoder **states**
- **Horizontally,** tracks **time steps**
- ***Branches*** connect states in successive time steps

Trellis:



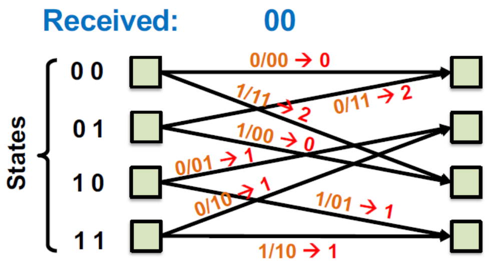
At the sender, transmitted bits trace a unique, single path of branches through the trellis – e.g. transmitted message $x = 1011$



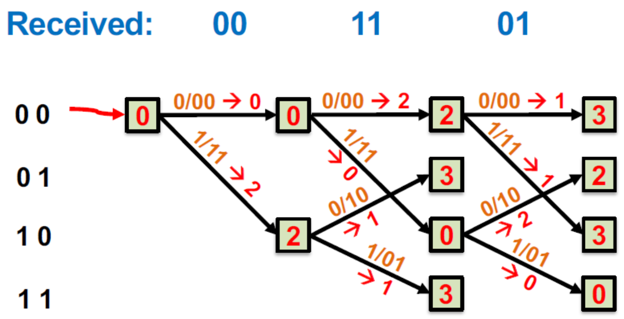
Two possibilities for computing the “length” of the path:

- **hard decision**: have possibly-corrupted encoded bits, after reception
- soft decision: have possibly-corrupted likelihoods of each bit, after reception

Hard decision branch metric: Hamming Distance between received and transmitted bits



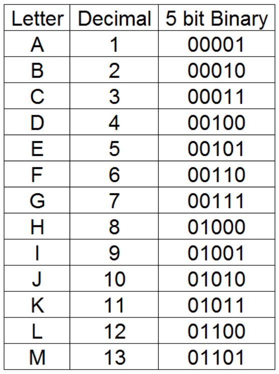
Winning branch has lower path metric (fewer bit errors):



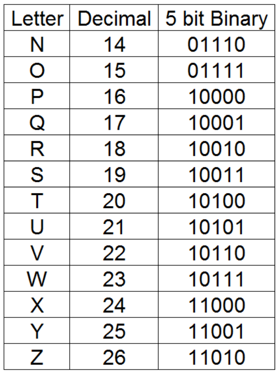
Assignment: For the following code: $K = 4, g_0 = 1101, g_1 = 1010, g_2 = 0110$, and a starting state 000, decode the following message:

$x = 000010101010010101010100110101101001011010100010100110101101011110011001011100110101011010010010001010110$

The 105 bits correspond to 35 “messages bits”, each of length 3 – the number of generators, i.e. the decoded message will have 35 bits. You can try to decipher the message using the following table (i.e., group the result by 5 bits and find the corresponding 7 letters)



Letter	Decimal	5 bit Binary
A	1	00001
B	2	00010
C	3	00011
D	4	00100
E	5	00101
F	6	00110
G	7	00111
H	8	01000
I	9	01001
J	10	01010
K	11	01011
L	12	01100
M	13	01101



Letter	Decimal	5 bit Binary
N	14	01110
O	15	01111
P	16	10000
Q	17	10001
R	18	10010
S	19	10011
T	20	10100
U	21	10101
V	22	10110
W	23	10111
X	24	11000
Y	25	11001
Z	26	11010