Revision of Lecture Fourteen

- **Convolutional code** $CC(n, k = 1, N)$ encoding:
  1. encoder circuit
  2. table of state transitions and output bits
  3. state-transition diagram, and trellis diagram

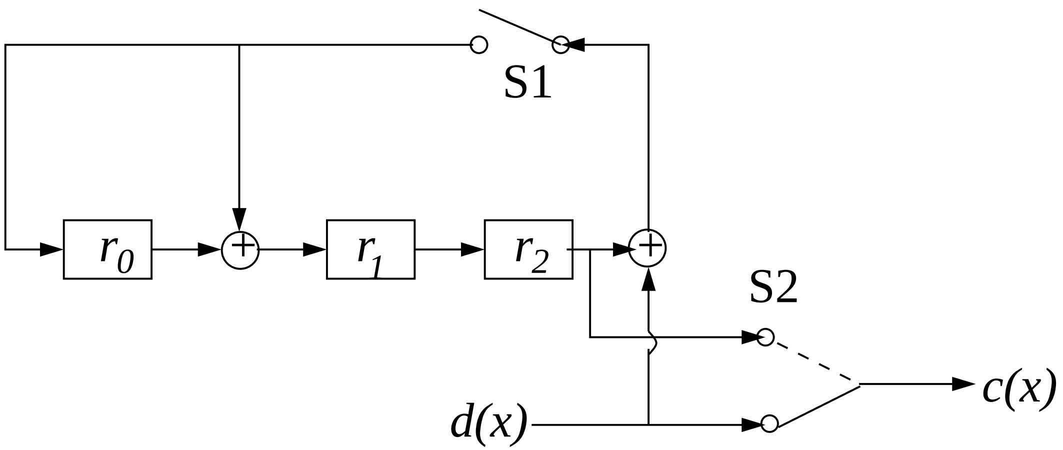
- Convolutional code $CC(n, k = 1, N)$ decoding:
  1. maximum likelihood sequence decoding principle
  2. trellis diagram based Viterbi decoding
  3. hard-input and hard-output decoding, soft-input and hard-output decoding

- This lecture focuses on a class of **linear block codes**, called BCH

Systematic BCH Codes

- $BCH(n, k, d_{\min})$: code rate $R = k/n$ with the minimum Hamming distance of the code $d_{\min}$. The block size is typically large and the smallest block size BCH is $BCH(7, 4, 3)$
- Example: $BCH(7, 4, 3)$ with $g(x) = 1 + x + x^3$

  Encoder circuit



- **Encoding process** for data $d(x) = 1 + x^2 + x^3$

input bits	shift index	shift register r_0	r_1	r_2	state no.	output bit
1 0 1 1	0	0	0	0	0	-
1 0 1	1	1	1	0	6	1
1 0	2	1	0	1	5	1
1	3	1	0	0	4	0
-	4	1	0	0	4	1
-	5	0	1	0	2	0
-	6	0	0	1	1	0
-	7	0	0	0	0	1

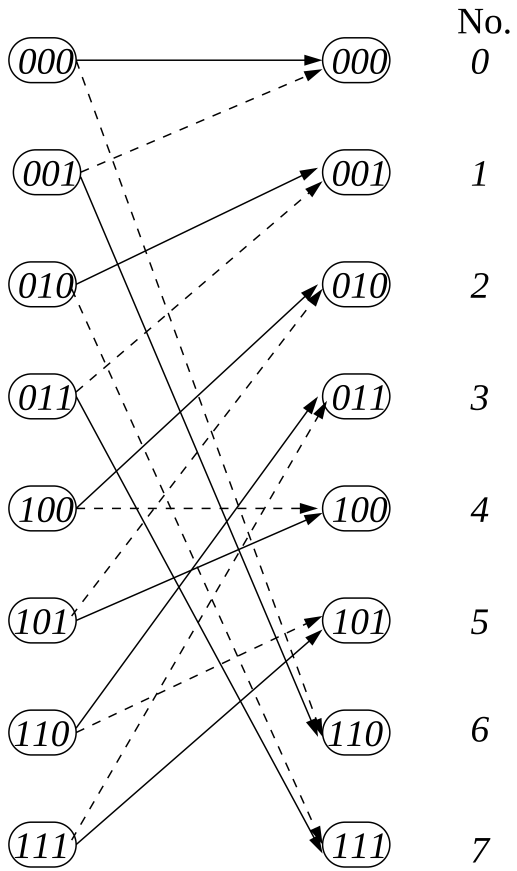
- Encoding has n stages, starts and ends at all zero state
- One output bit follows a clock pulse
- For first k shifts, output bit is input bit
- Next $n - k$ shifts, parity bits shifted to output
- Number of states 2^{n-k} ($2^{7-4} = 8$ in this example)

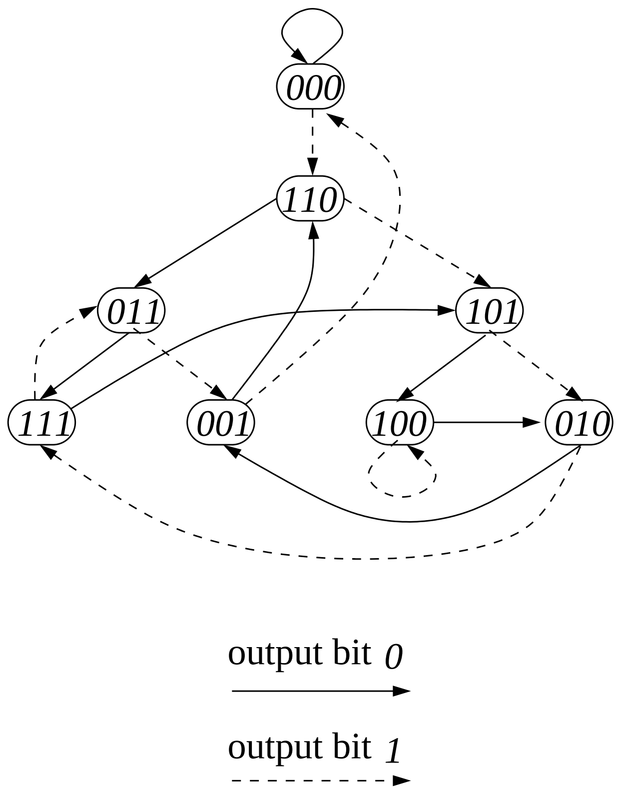
BCH(7,4,3) Encoder

- BCH(7,4,3) with $g(x) = 1 + x + x^3$

Table of state transitions with output bits, state transition diagram and state diagram

current state r_0	current state r_1	current state r_2	output bit	next state r_0	next state r_1	next state r_2
0	0	0	0	0	0	0
0	0	1	0	1	1	0
0	1	0	0	0	0	1
0	1	1	0	1	1	1
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	0	0	1	1
1	1	1	0	1	0	1
0	0	0	1	1	1	0
0	0	1	1	0	0	0
0	1	0	1	1	1	1
0	1	1	1	0	0	1
1	0	0	1	1	0	0
1	0	1	1	0	1	0
1	1	0	1	1	0	1
1	1	1	1	0	1	1





- Each row in Table of state transitions can be either in data bit shifting-out process or in parity bit shifting-out process

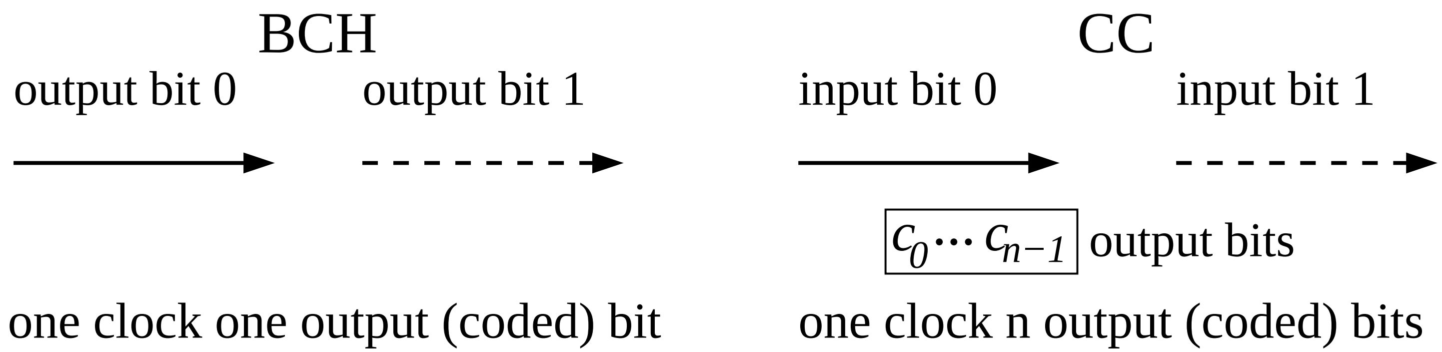
BCH Encoder (continue)

- There are only two legitimate state transitions for each state, depending on the output bit; similarly, each state has two merging paths

- State diagram can be used to encode data without the need to use the shift register circuit, e.g. data 1011 (rightmost enters the encoder first):

$$000 \xrightarrow{1} 110 \xrightarrow{1} 101 \xrightarrow{0} 100 \xrightarrow{1} 100 \xrightarrow{0} 010 \xrightarrow{0} 001 \xrightarrow{1} 000$$

- Some **notation differences** between BCH and CC:



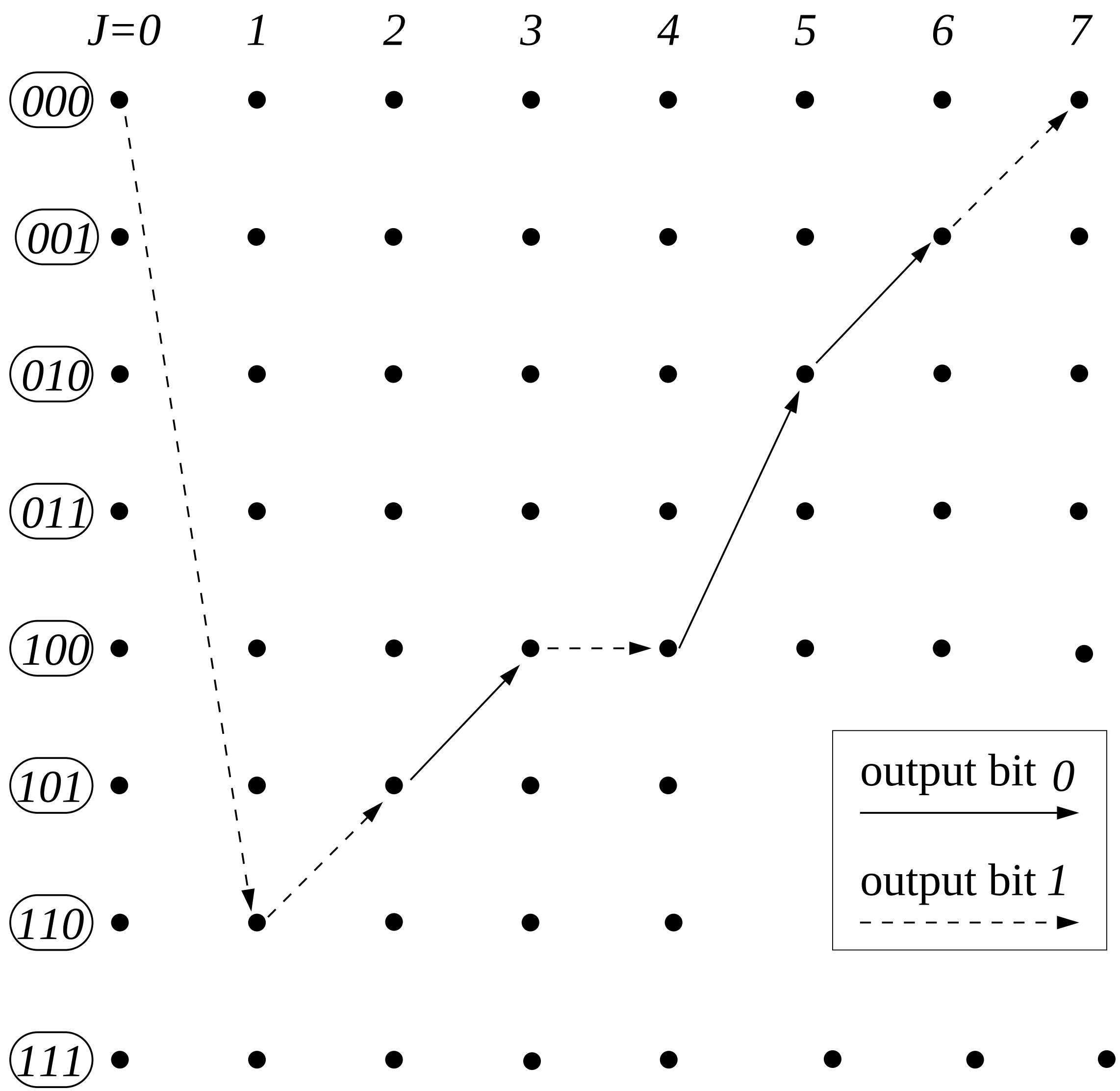
BCH Encoder: Trellis Diagram

- **Trellis diagram** shows the history of state transitions with output (code) bits

- It always starts from zero state and end at zero state after n clocks

- BCH(7,4,3) with $g(x) = 1 + x + x^3$: Encoding for data 1011 (rightmost bit enters the encoder first)

Difference with CC: "arrow" indicates output bit while in CC it indicates input bit

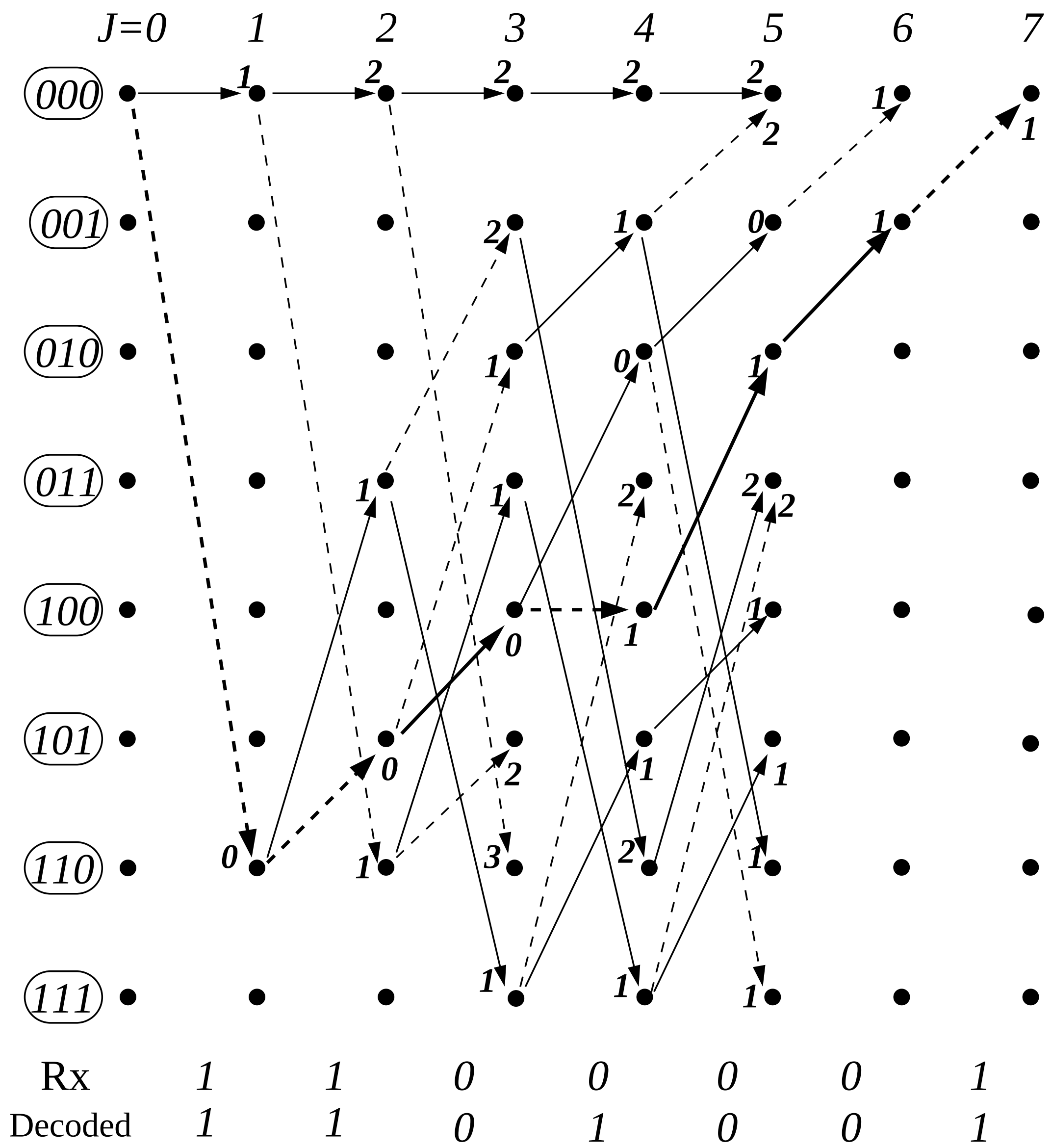


Hard-Input Hard-Output Viterbi Decoding

- Same $BCH(7,4,3)$ with transmitted sequence 1101001 and received sequence 1100001 (the leftmost bit at the leftmost position of trellis)
- Unlike CC, BCH trellis starts always ends at zero state after n stages

  For this code $n = 7$, and at stage 6, there is no need to consider state transitions for states 100, 101, 110 and 111, as corresponding paths will not end at zero state at stage 7
- Usual VA decoding rules apply

  For example, if decoding is correct, the winning path metric is the number of transmission errors caused by the channel



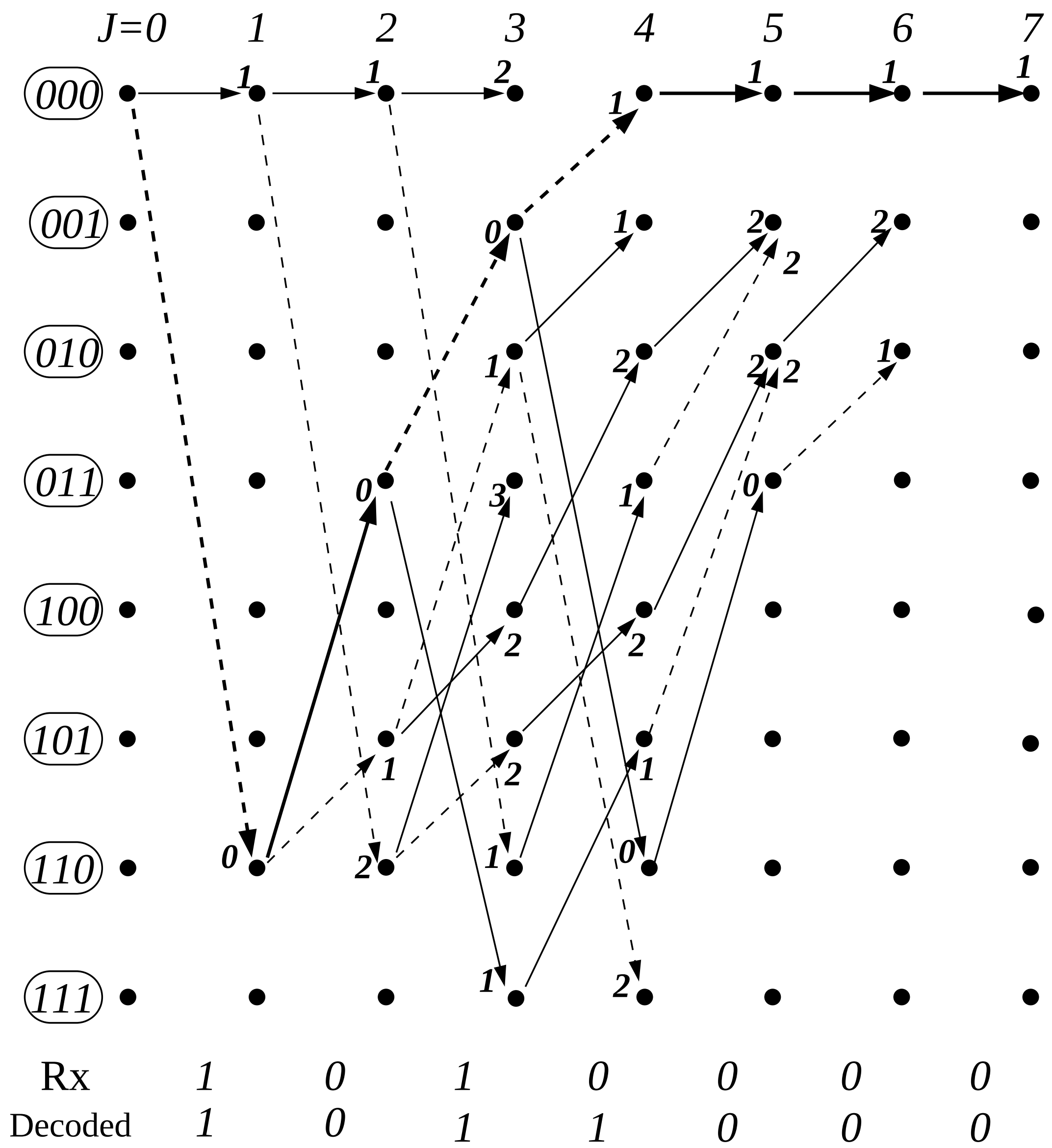
HIHO Viterbi Decoding: Another Example

- The same $BCH(7,4,3)$ but with transmitted sequence 0000000 and received sequence 1010000 (the leftmost bit at the leftmost position of trellis)
- For this code, minimum Hamming distance $d_{\min} = 3$ and hard-input decoding can only correct upto 1 bit error

  But this example has two bit errors
- Hence this is erroneous decoding

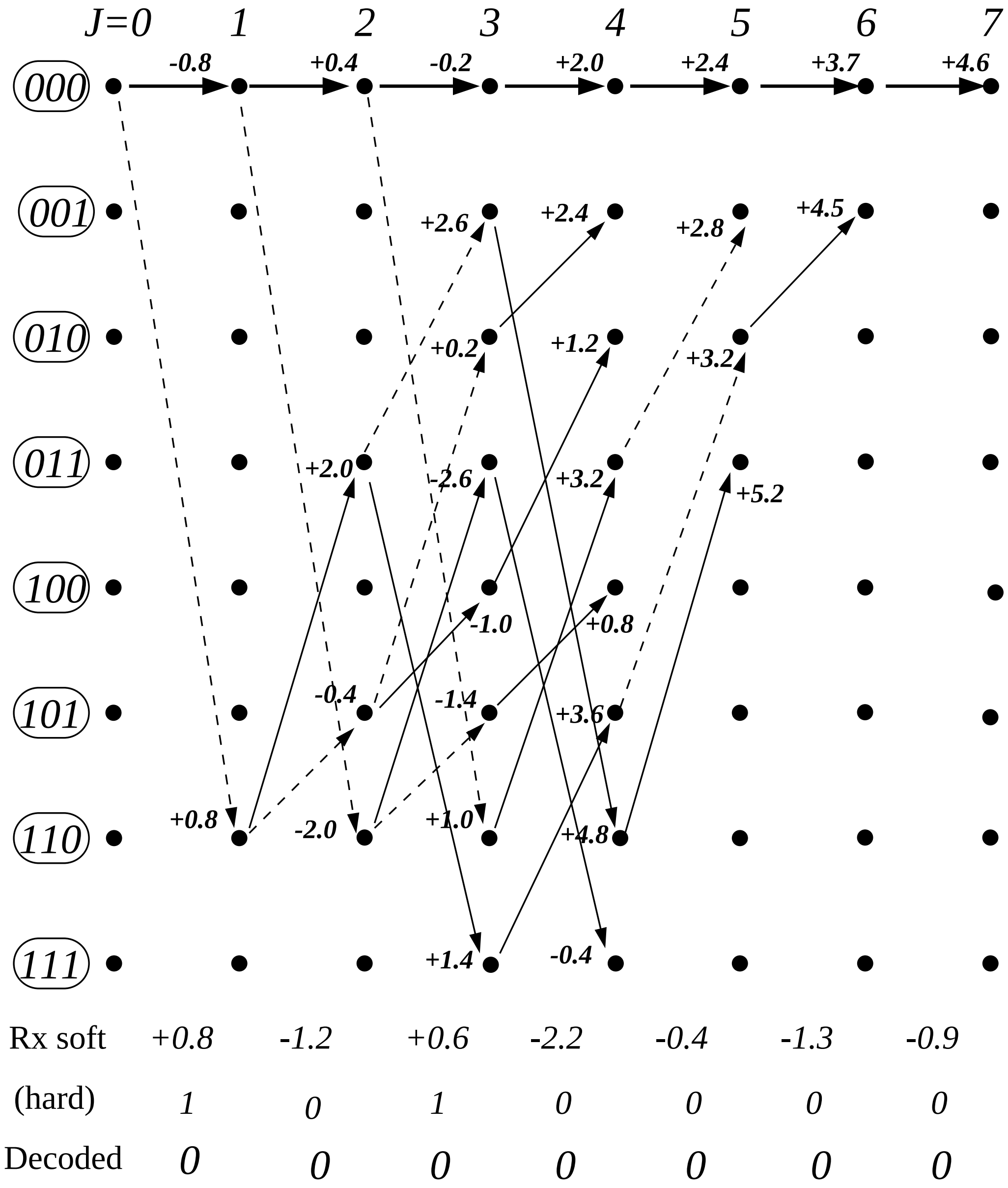
  Note how decoding actually make thing worst

  If decoding is incorrect, the winning path metric is not number of transmission errors caused by the channel



Soft-Input Hard-Input Viterbi Decoding

- The same $BCH(7,4,3)$ with the transmitted sequence 0000000 and the received soft sequence $+0.8, -1.2, +0.6, -2.2, -0.4, -1.3, -0.9$ (Received hard sequence would be 1010000, with the leftmost bit at the leftmost position of trellis)
- Usual soft-input Viterbi decoding rules apply, e.g. if trellis branch output bit is $+1$ and received soft output bit is $+0.8$, it has metric $+0.8$, while for trellis branch of output bit -1 it has metric -0.8
- Previously, HIHO Viterbi algorithm produced erroneous decoding
- Note with soft-input decoder is able to correct two bit errors



MAP and Soft Output Viterbi Decoding

- We have examined HIHO and SIHO Viterbi decoding schemes, and there are SISO schemes for iterative decoding
- **Maximum a posterior probability** decoding is naturally SISO, input log likelihood ratios and output log likelihood ratios
  - MAP decoding algorithm is more powerful than soft output Viterbi decoding at cost of higher complexity
  - Reference: L. Hanzo, T.H. Liew and B.L. Yeap, *Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission Over fading Channels*. Wiley, 2002
- We briefly discuss **soft output Viterbi algorithm**
  - Transmitted codeword $\boldsymbol{x}_k = [x_{k,0}\ x_{k,1} \cdots x_{k,n-1}]$, received codeword $\boldsymbol{y}_k = [y_{k,0}\ y_{k,1} \cdots y_{k,n-1}]$, and n is number of bits in each codeword ($n = 1$ for BCH)
  - Given transmitted $x_k \in \{\pm\}$, receiver output

$$y_k = a x_k + \varepsilon_k$$

    ε_k: AWGN with $\mathrm{E}\{|\varepsilon_k|^2\} = 2\sigma^2$, a: channel fading amplitude
  - Channel reliability value L_c depends on SNR and channel fading amplitude

$$L_c = 4a\frac{E_b}{2\sigma^2}$$

    E_b: transmitted energy per bit, and for AWGN channel $a = 1$

Soft-Output Viterbi Algorithm

- Two modifications to classical Viterbi algorithm
  1. Path metrics take account of *a priori* information when selecting ML path from trellis
  2. Provide soft output in form of *a posterior* LLR $L(b_k|\boldsymbol{y})$ for each decoded bit

1. Consider state sequence $\boldsymbol{s}_k^s$: states along surviving path at state $S_k = s$ of stage k in trellis
   - If path $\boldsymbol{s}_k^s$ at stage k has path $\boldsymbol{s}_{k-1}^{\grave{s}}$ at its first $k-1$ transitions, path metric $M\left(\boldsymbol{s}_k^s\right)$ is

$$M\left(\boldsymbol{s}_k^s\right) = M\left(\boldsymbol{s}_{k-1}^{\grave{s}}\right) + \ln\left(\gamma_k\left(\grave{s}, s\right)\right)$$

   - $\gamma_k\left(\grave{s}, s\right)$ is branch transition probability from $S_{k-1} = \grave{s}$ to $S_k = s$, and

$$\ln\left(\gamma_k\left(\grave{s}, s\right)\right) = \frac{1}{2} b_k L(b_k) + \frac{L_c}{2} \sum_{l=0}^{n-1} y_{k,l} x_{k,l}$$

   $b_k L(b_k)$: *a priori* information
   - Two paths $\boldsymbol{s}_k^s$ and $\widetilde{\boldsymbol{s}}_k^s$ reaching state $S_k = s$ have metrics $M\left(\boldsymbol{s}_k^s\right)$, and $M\left(\widetilde{\boldsymbol{s}}_k^s\right)$ and $\boldsymbol{s}_k^s$ is survivor because of its higher metric
   - Then metric difference is LLR of correct decision at $S_k = s$

$$L(\text{correct decision at } S_k = s) = \Delta_k^s = M\left(\boldsymbol{s}_k^s\right) - M\left(\widetilde{\boldsymbol{s}}_k^s\right) \geq 0$$

2. At end of trellis, when winning ML path is identified, we need to finds LLRs giving reliability of the bit decisions along ML path

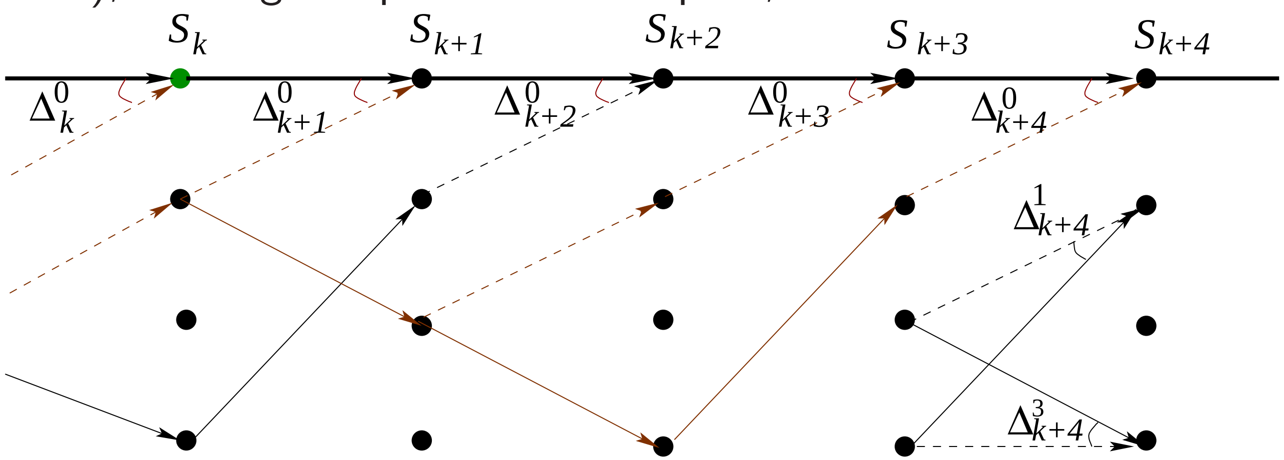
SOVA (continue)

2. When calculating LLR of bit b_k, SOVA must take account of probability that paths merging with ML path from stage k to stage $k+\delta$ were incorrectly discarded
   – δ may be set to five times of constraint length for convolutional code
   – and *a posterior* LLR

$$L(b_k|\boldsymbol{y}) \approx b_k \min_{\substack{i=k,\cdots,k+\delta \\ b_k \neq b_k^i}} \Delta_i^{s_i}$$

   – b_k: bit value given by ML path
   – b_k^i: value of this bit for the path which merged with ML path and was discarded at stage i

- Four states (0 to 3), winning ML path is all-zero path, $\delta = 4$



   – $L(b_k|\boldsymbol{r})$: -1 multiplied by the minimum of metric differences Δ_k^0, Δ_{k+1}^0, Δ_{k+3}^0 and Δ_{k+4}^0
   – Note Δ_{k+2}^0 is not considered, as $b_k^{k+2} = b_k = -1$

Summary

- $BCH(n, k, d_{\min})$: code rate $R = k/n$ and the minimum Hamming distance of the code $d_{\min}$
- $BCH(n, k, d_{\min})$ encoder: encoder circuit, table of state transitions and output bit, state-transition diagram, state diagram and trellis diagram
- $BCH(n, k, d_{\min})$ decoder: trellis diagram based Viterbi decoding, hard-input and hard-output decoding, soft-input and hard-output decoding
- Similarities and differences with convolutional codes
- Soft-input and soft-output decoding: for iterative decoding
  - Soft-output Viterbi decoding, for both BCH block codes and convolutional codes