Revision of Lecture Thirteen

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

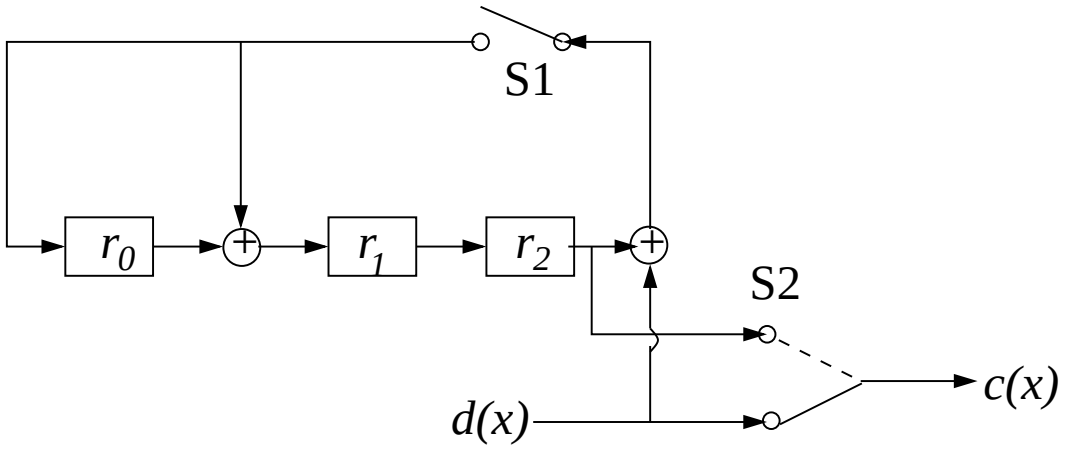
- $CC(n, k = 1, N)$ decoding: maximum likelihood sequence decoding, trellis diagram based Viterbi decoding, hard-input and hard-output decoding, soft-input and hard-output decoding

- 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

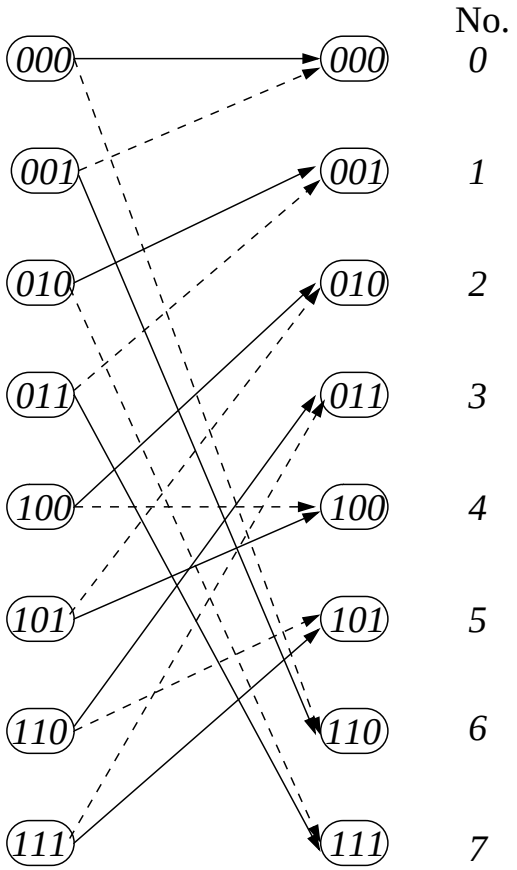
- Encoding has n stages, starts and ends at all zero state
- One output bit follows a clock pulse
- For first k shifts, output bit is input bit
- Next $n - k$ shifts, parity bits shifted to output
- Number of states 2^{n-k} ($2^{7-4} = 8$ in this example)

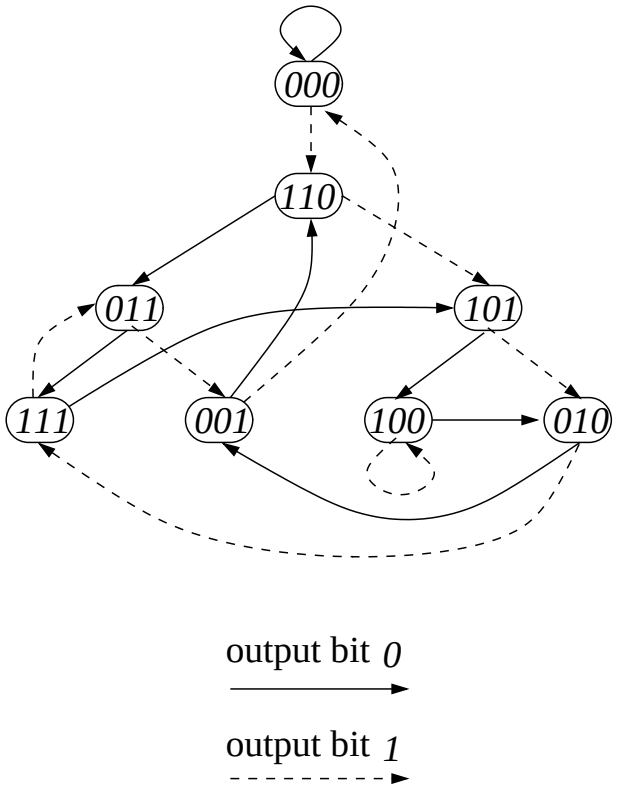
BCH(7,4,3) Encoder

- 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			output	next state		
r_0	r_1	r_2	bit	r_0	r_1	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





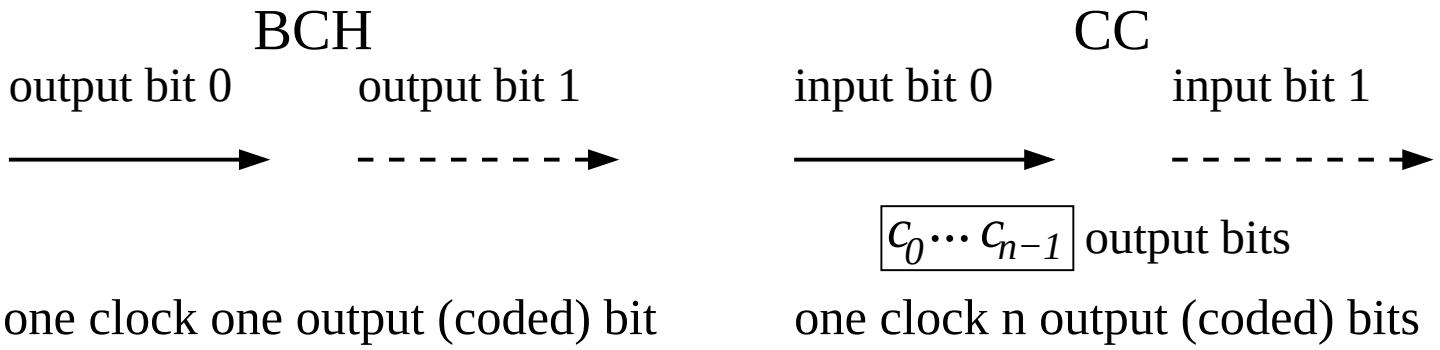
BCH Encoder (continue)

- 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):

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

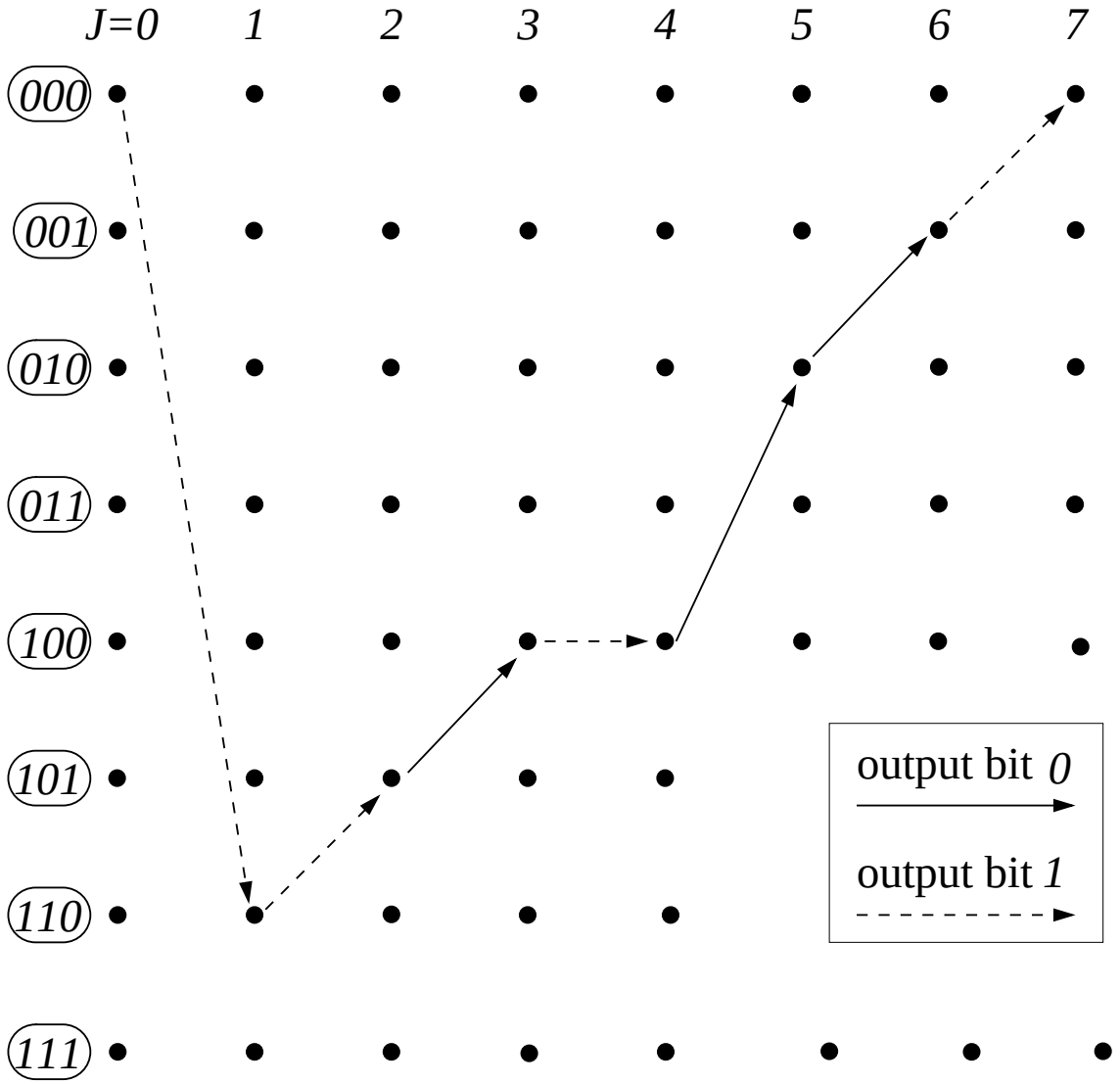
- Some **notation differences** between BCH and CC:



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

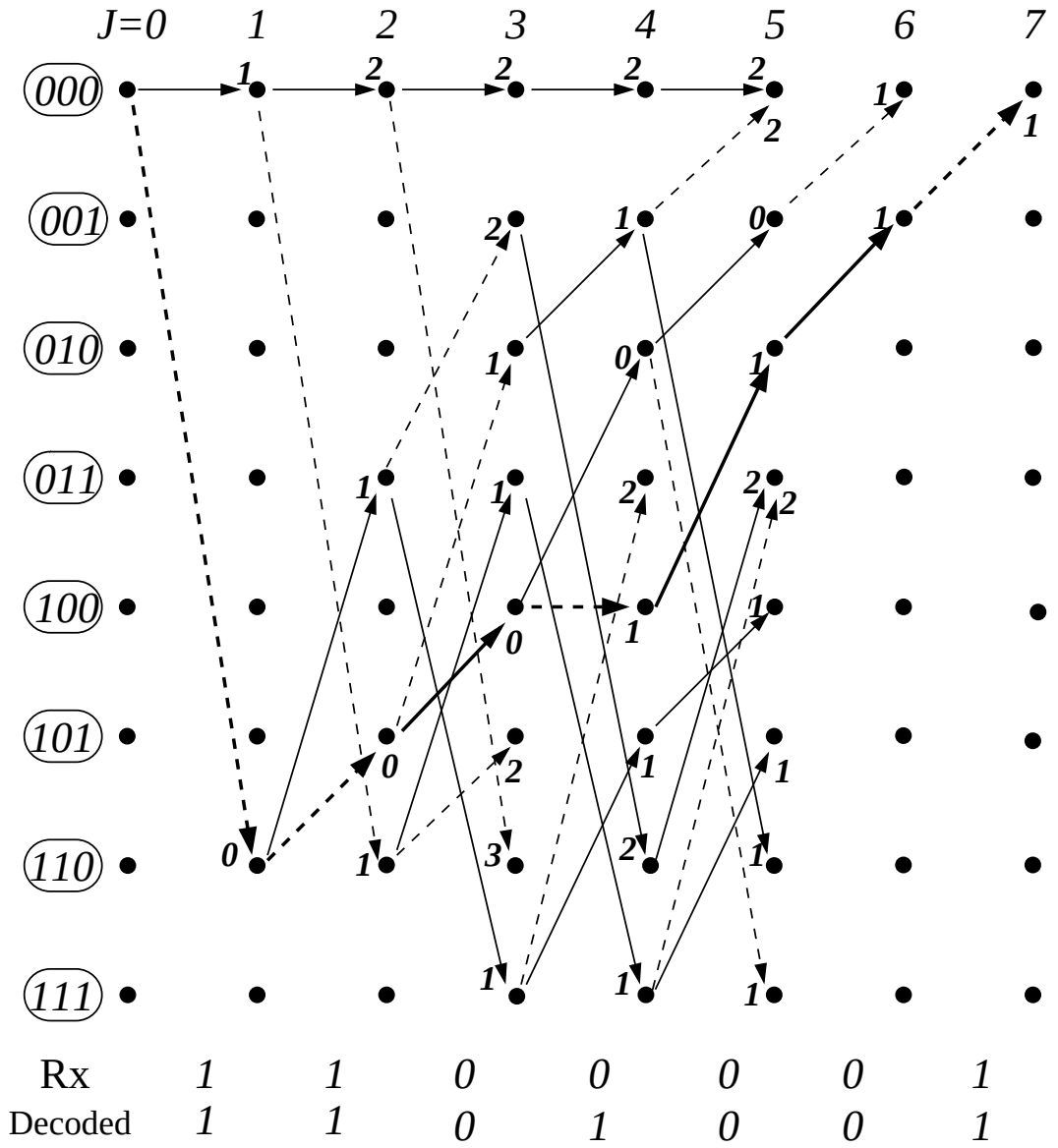


Hard-Input Hard-Output Viterbi Decoding

- 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



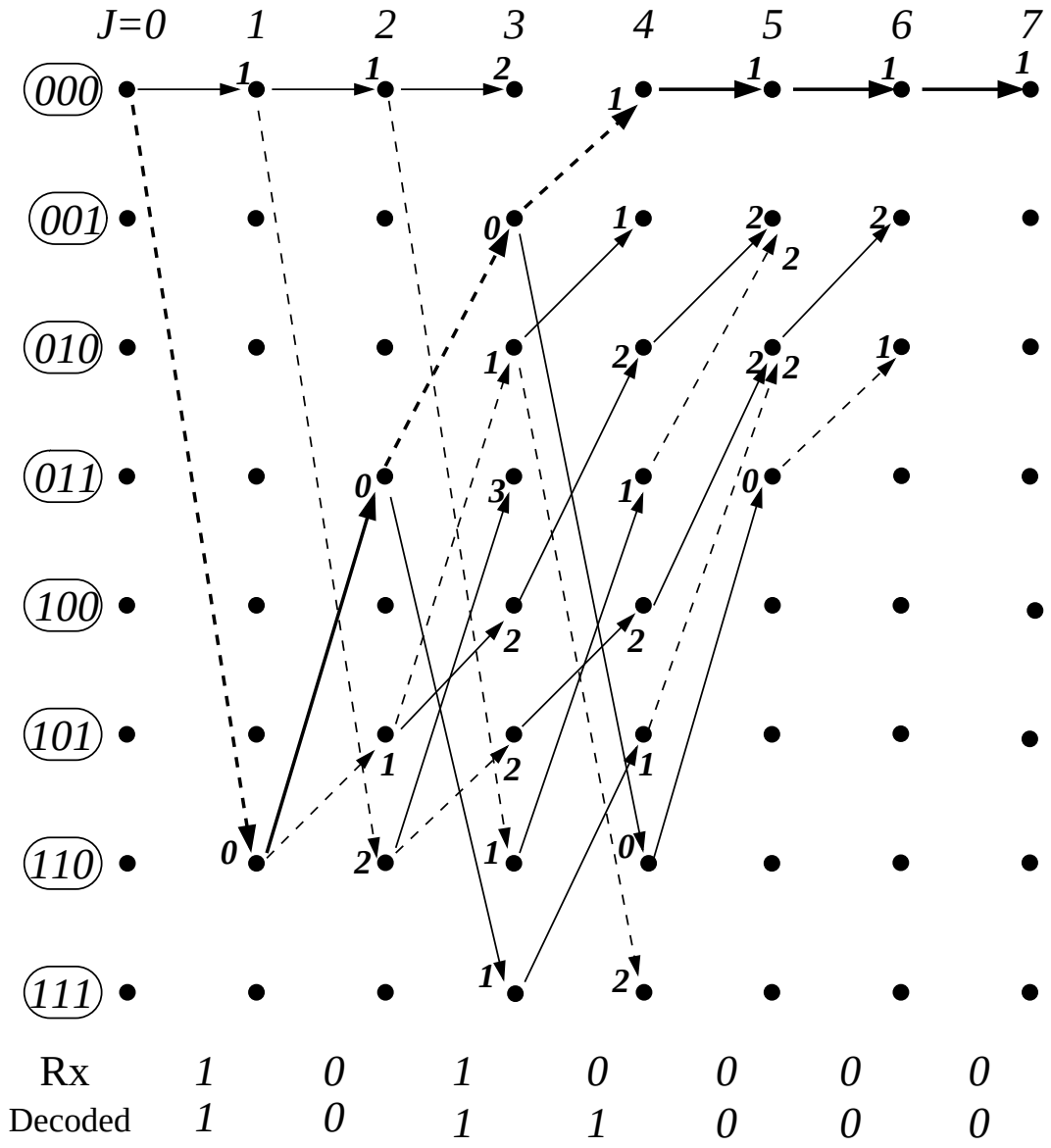
HIHO Viterbi Decoding: Another Example

- 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

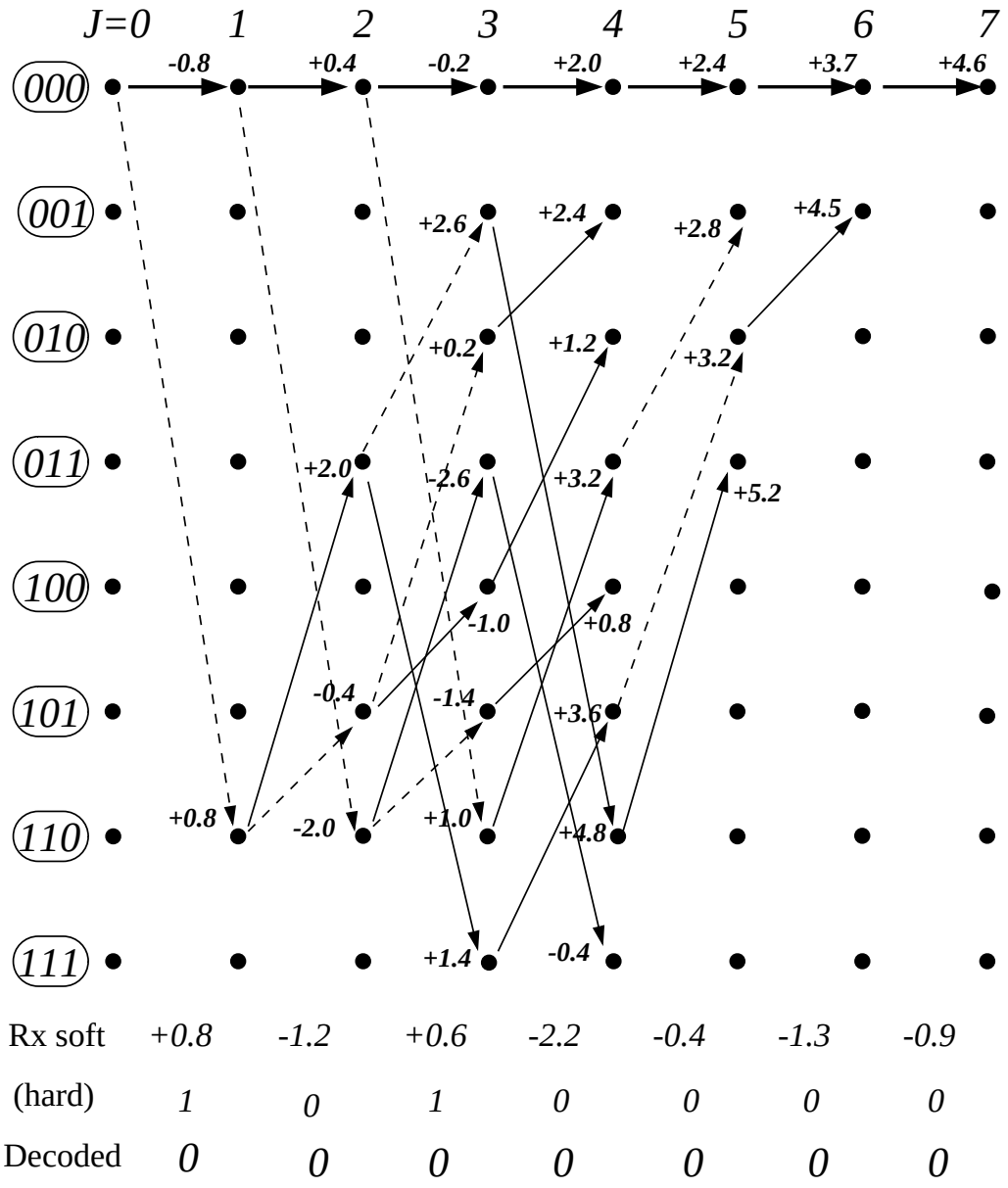
  Note how decoding actually make thing worst

  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 Decoding and Turbo Coding

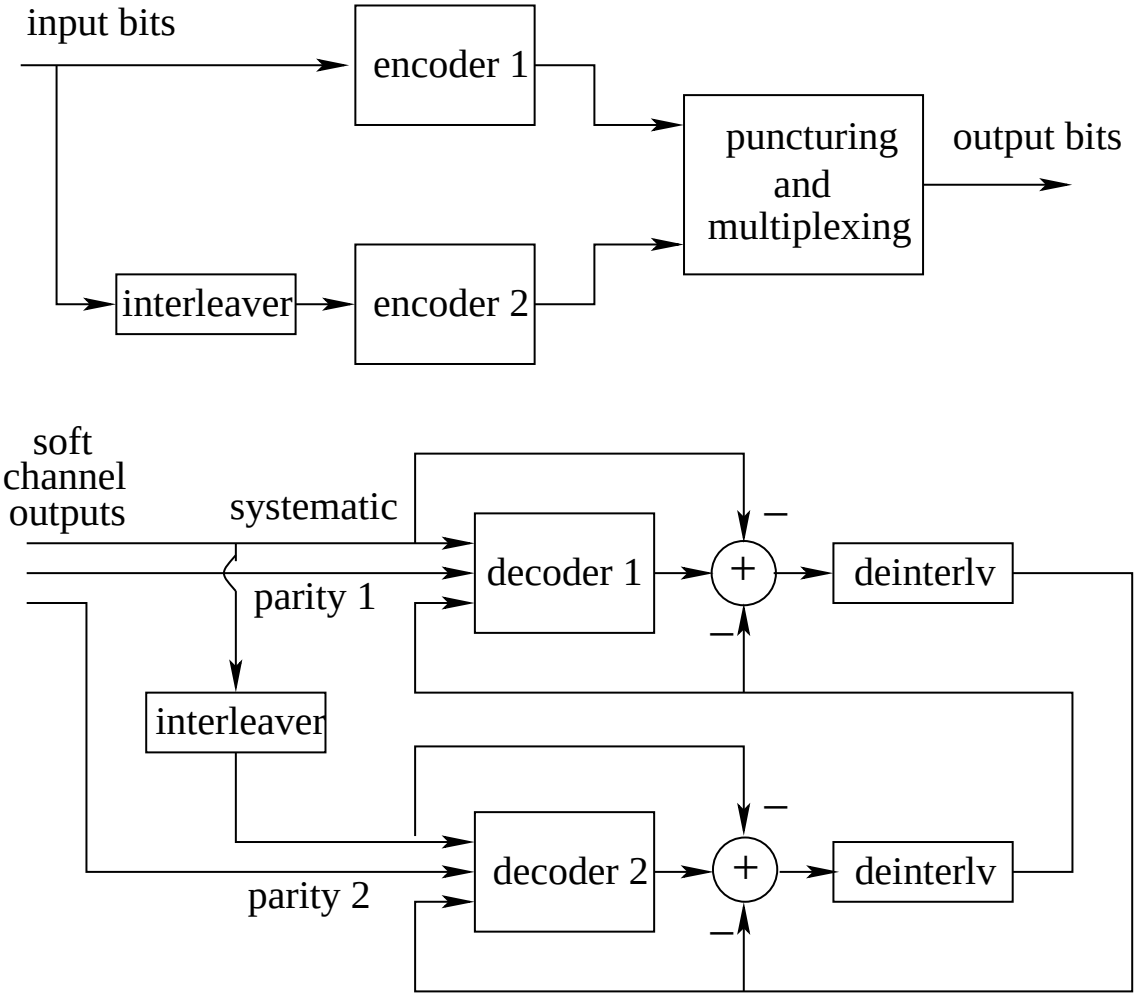
We have examined HIHO and SIHO Viterbi decoding schemes, and there are SISO schemes for iterative decoding

- Maximum a posterior probability decoding is more powerful than Viterbi decoding at cost of higher complexity

  MAP is naturally SISO, input log likelihood ratios and output log likelihood ratios

  Refer to an advance text book for further study if interested

- Concept of turbo coding
  - Turbo encoder (assuming systematic $CC(2, 1, 2)$):
  - Turbo decoder: an iterative decoding



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, very powerful, MAP and turbo coding concepts

Iterative or turbo principle has been generalised to variety of application, where **soft-information** plays a key role, e.g. Iterative detection / decoding, iterative time recovery / detection, iterative between three components, etc