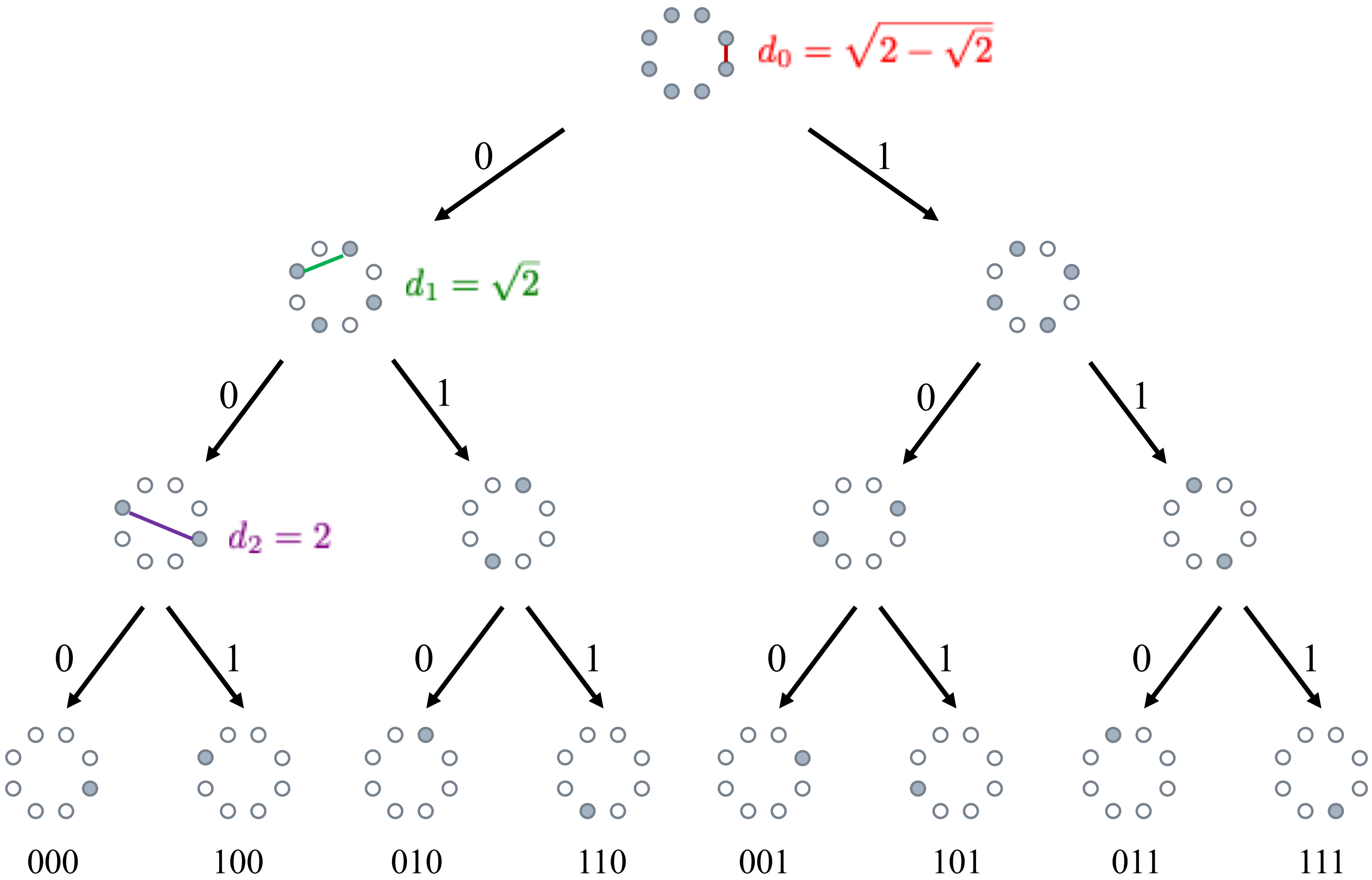
Part 8 Trellis Coded Modulation, Turbo Codes and LDPC Codes

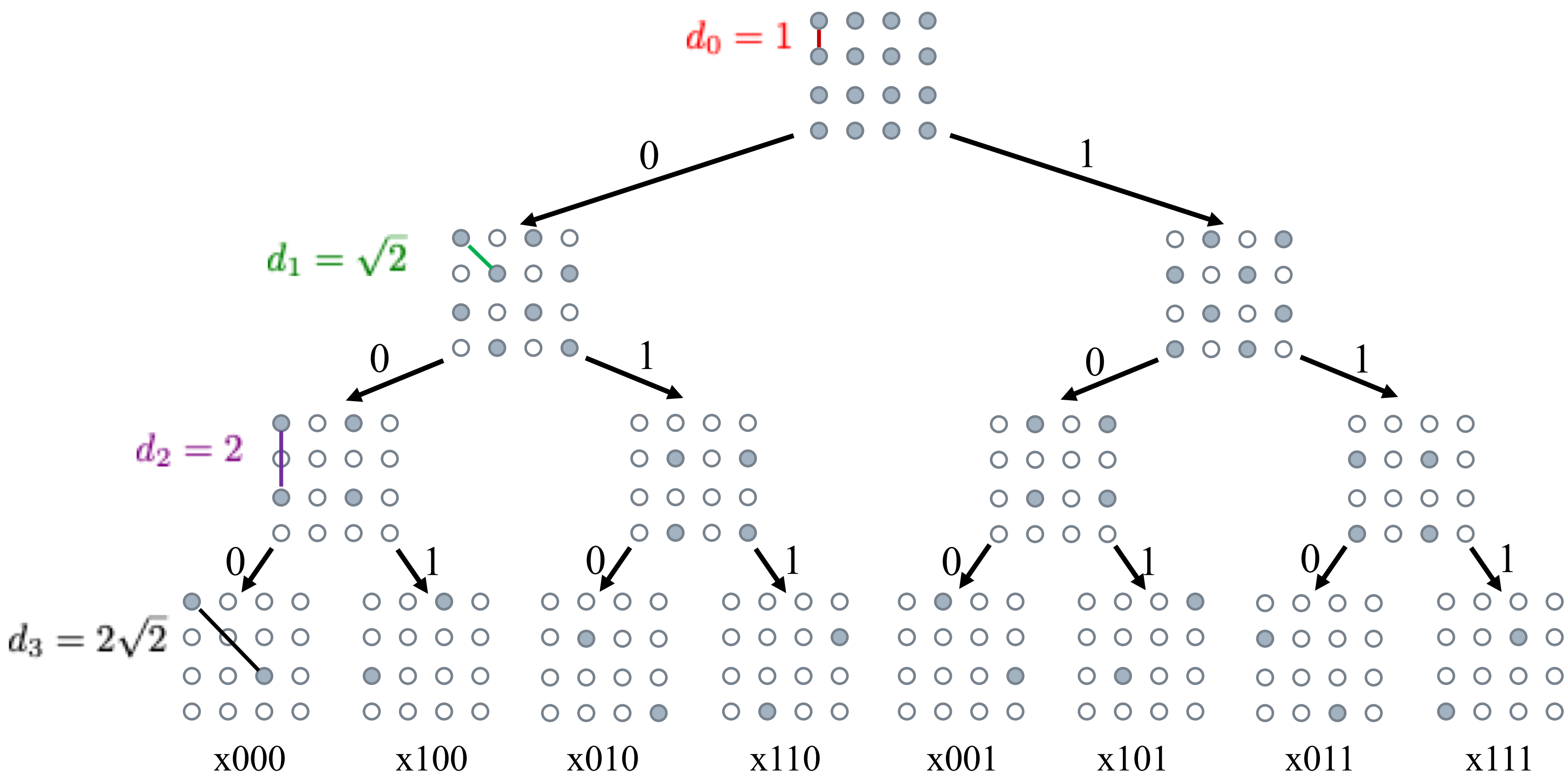
Trellis-Coded Modulation

- In the previous section, encoding is performed separately from modulation in the transmitter, and likewise for decoding and detection in the receiver.
- To attain more effective utilization of the available bandwidth and power, coding and modulation have to be treated as a single entity, e.g., trellis-coded modulation.
  - Instead of selecting codewords from "code bit domain", we choose codewords from "signal constellation domain".

Partitioning of 8-PSK constellation that shows $d_0 < d_1 < d_2$.



Partitioning of 16-QAM constellation that shows $d_0 < d_1 < d_2 < d_3$.



Trellis-Coded Modulation

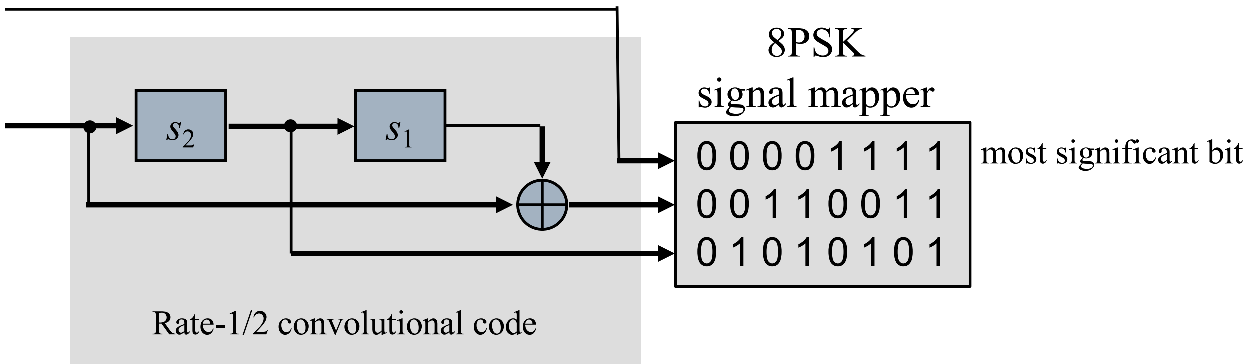
- Codeword versus code signal

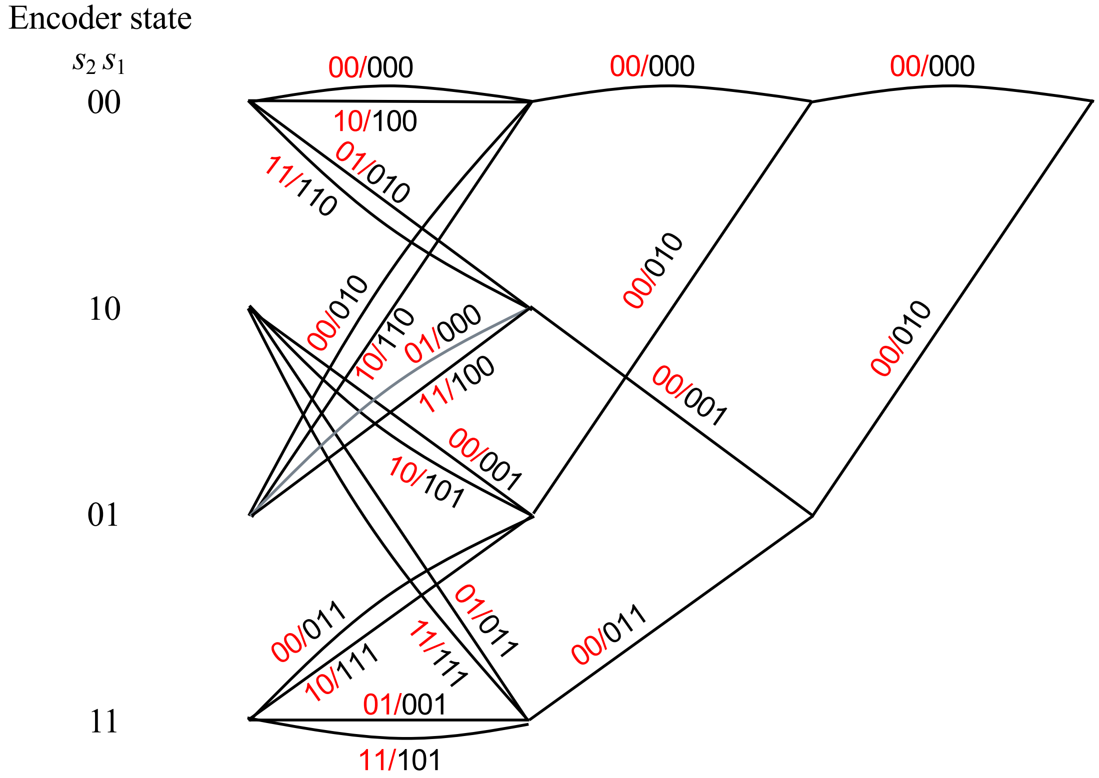
0000 0011 1100 1111	Select 4 out of 16 possibilities (The bit patterns are dependent temporally so that these bit patterns exhibit "error correcting capability".)
$0 \quad \pi$ $\frac{\pi}{2} \quad \frac{3\pi}{2}$ $\pi \quad 0$ $\frac{3\pi}{2} \quad \frac{\pi}{2}$	Select 4 out of 16 possibilities from QPSK constellation (The signal patterns are dependent temporally so these signal patterns exhibit "error correcting capability".)

Trellis-Coded Modulation

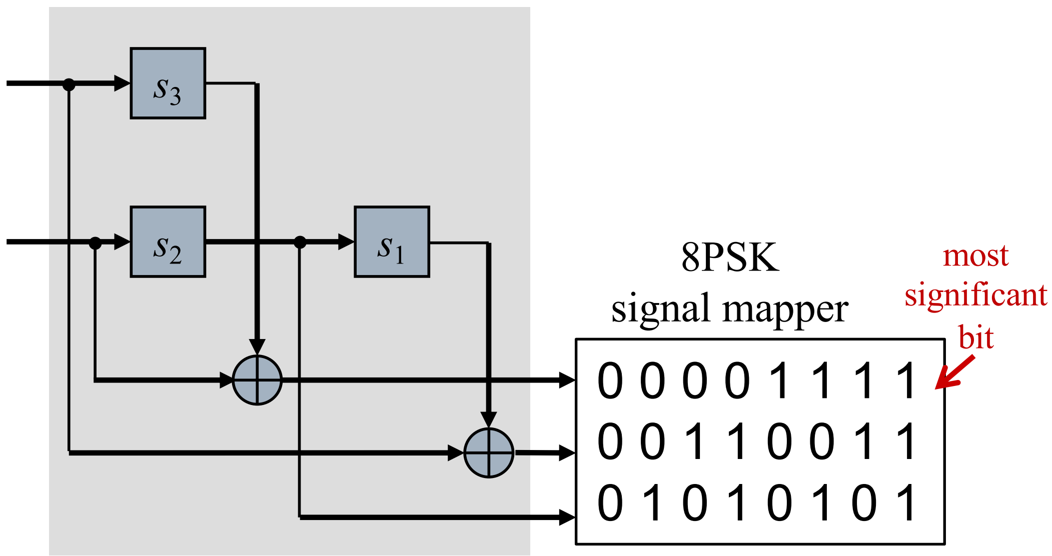
- Trellis codeword versus trellis code signal
  - The next code bit is a function of the current trellis state and some number of the previous information bits.
  - The next code signal is a function of the current trellis state and some number of the previous information signals.

- Example of trellis-coded modulation
  - 4-state Ungerboeck 8-PSK code
    - Code rate = 2 bits/symbol

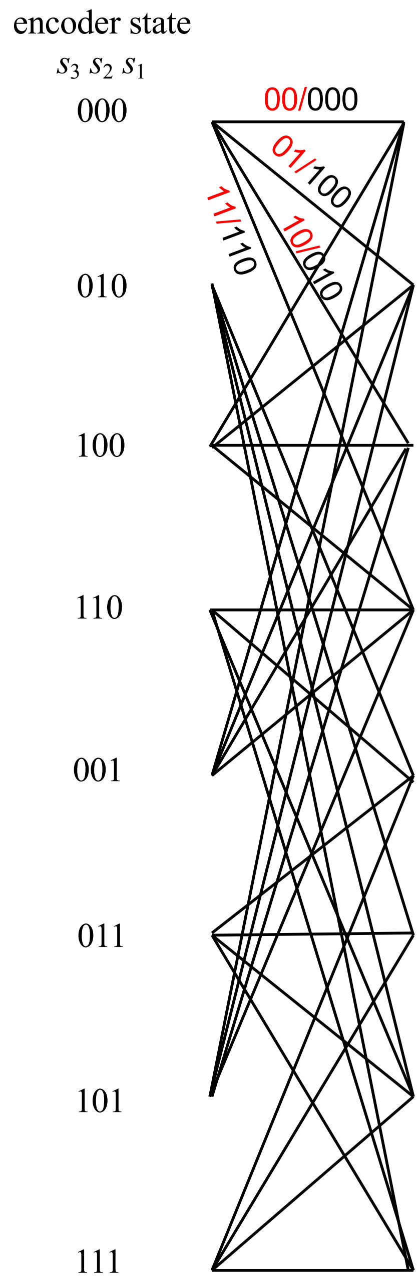




- Example of trellis-coded modulation
  - 8-state Ungerboeck 8-PSK code
    - Code rate = 2 bits/symbol



Rate-2/3 convolutional code



- Soft decision decoding (can be analyzed via an equivalent binary-input additive white Gaussian noise channel)
  - The error rate of Ungerboeck codes (particularly at high SNR) is dominated by the "two codewords (i.e., signal words)" whose pairwise Euclidean distance is equal to d_{free}. (This d_{free} represents Euclidean distance, not the Hamming distance defined previously.)

$$\Rightarrow \quad \text{Equivalently } x_j = s_{j,m} + w_j \text{ for } j = 1, \ldots, N$$

$$\text{where } \|\boldsymbol{s}_0 - \boldsymbol{s}_1\|^2 = \sum_{j=1}^{N} (s_{j,0} - s_{j,1})^2 = d_{\text{free}}^2$$

$$\Rightarrow \quad \hat{m} = \arg\max \{P(\boldsymbol{x} | \boldsymbol{s}_0), P(\boldsymbol{x} | \boldsymbol{s}_1)\}$$

$$\Rightarrow \quad \hat{m} = \arg\max \left\{ \prod_{j=1}^{N} e^{-(x_j - s_{j,0})^2/2\sigma^2}, \prod_{j=1}^{N} e^{-(x_j - s_{j,1})^2/2\sigma^2} \right\}$$

$$\Rightarrow \quad \|\boldsymbol{x}-\boldsymbol{s}_0\|^2 \underset{\boldsymbol{s}_1}{\overset{\boldsymbol{s}_0}{\lessgtr}} \|\boldsymbol{x}-\boldsymbol{s}_1\|^2 \qquad \boldsymbol{x}\begin{cases}\mathcal{N}(\boldsymbol{s}_0,\sigma^2\mathbb{I}) & \boldsymbol{s}_0 \text{ transmitted}\\ \mathcal{N}(\boldsymbol{s}_1,\sigma^2\mathbb{I}) & \boldsymbol{s}_1 \text{ transmitted}\end{cases}$$

- Based on the decision rule $\|\boldsymbol{x}-\boldsymbol{s}_0\|^2 \underset{\boldsymbol{s}_1}{\overset{\boldsymbol{s}_0}{\lessgtr}} \|\boldsymbol{x}-\boldsymbol{s}_1\|^2$

$$\begin{aligned}
&\text{Dominant pairwise error}\\
&= P(\boldsymbol{s}_0 \text{ transmitted})\, P\left(\|\boldsymbol{x}-\boldsymbol{s}_0\|^2 > \|\boldsymbol{x}-\boldsymbol{s}_1\|^2 \,|\boldsymbol{s}_0 \text{ transmitted}\right)\\
&\quad + P(\boldsymbol{s}_1 \text{ transmitted})\, P\left(\|\boldsymbol{x}-\boldsymbol{s}_0\|^2 < \|\boldsymbol{x}-\boldsymbol{s}_1\|^2 \,|\boldsymbol{s}_1 \text{ transmitted}\right)\\
&= P\left(\|\boldsymbol{x}-\boldsymbol{s}_0\|^2 > \|\boldsymbol{x}-\boldsymbol{s}_1\|^2 \,|\boldsymbol{s}_0 \text{ transmitted}\right)\\
&= P\left(\|\boldsymbol{w}\|^2 > \|\boldsymbol{w}+\boldsymbol{s}_0-\boldsymbol{s}_1\|^2\right), \text{ where } \boldsymbol{x}=\boldsymbol{s}_0+\boldsymbol{w}\\
&= P\left(\langle \boldsymbol{w}, \boldsymbol{s}_0-\boldsymbol{s}_1\rangle < -\frac{1}{2}\|\boldsymbol{s}_0-\boldsymbol{s}_1\|^2\right)\\
&= \Phi\left(\frac{-\frac{1}{2}\|\boldsymbol{s}_0-\boldsymbol{s}_1\|^2 - 0}{\|\boldsymbol{s}_0-\boldsymbol{s}_1\|\sigma}\right) = \Phi\left(-\frac{d_{\text{free}}}{2\sigma}\right) \le \exp\{-d_{\text{free}}^2/(4N_0)\}
\end{aligned}$$

$$\langle \boldsymbol{w}, \boldsymbol{s}_0-\boldsymbol{s}_1\rangle \sim \mathcal{N}(0, \|\boldsymbol{s}_0-\boldsymbol{s}_1\|^2\sigma^2)$$

$$\Phi(-x) = \frac{1}{2}\text{erfc}\left(\frac{x}{\sqrt{2}}\right) \le \frac{1}{x\sqrt{2\pi}}e^{-x^2/2} \le e^{-x^2/2} \text{ for } x > 1/\sqrt{2\pi}$$

$$\sigma^2 = N_0/2$$

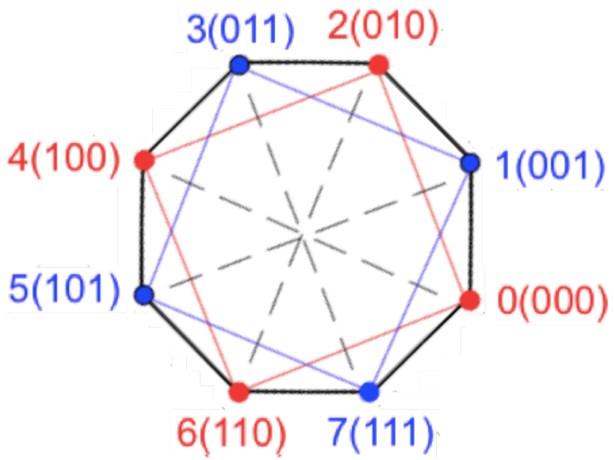
- Asymptotic coding gain (here, asymptotic = at high SNR) G_a
  - The performance gain due to coding (i.e., the performance gain of a coded system against an uncoded system)

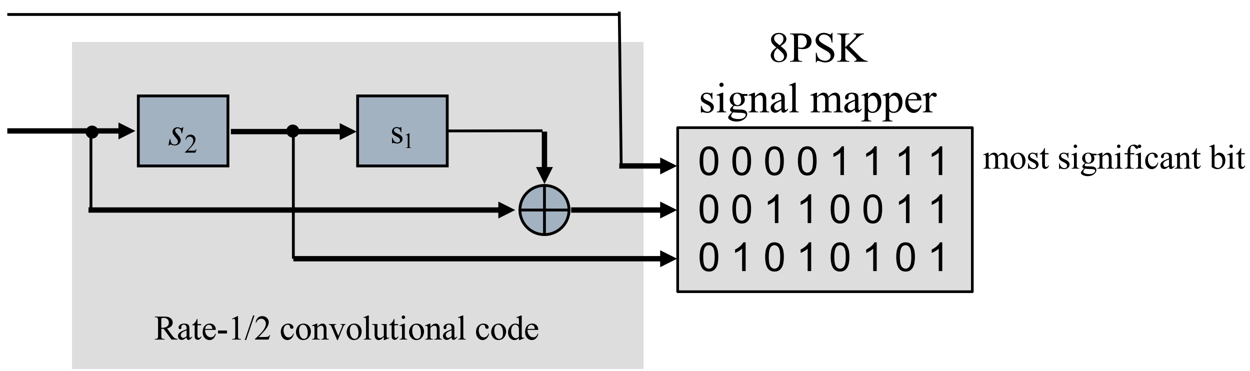
$$\text{Uncoded} \;\; \exp\left\{-d_{\text{ref}}^2/(4N_0)\right\}$$

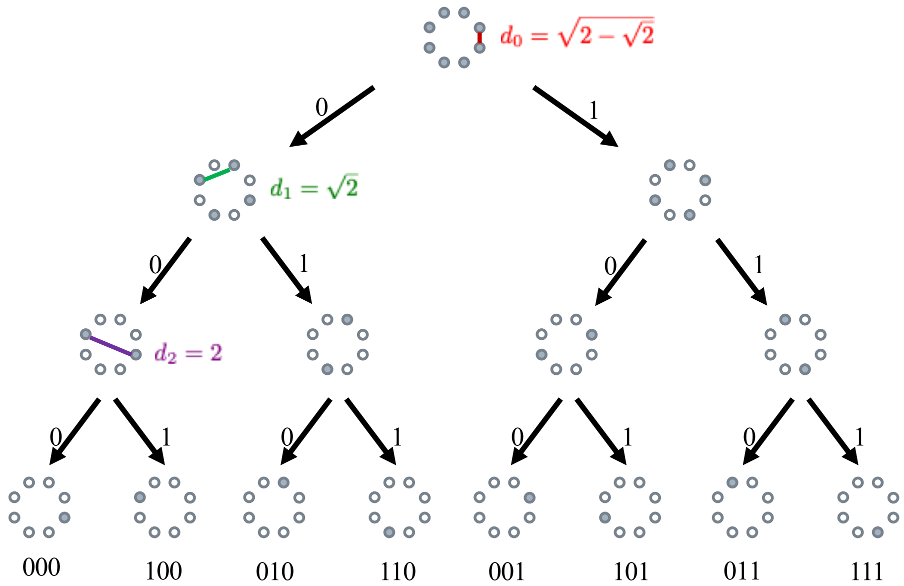
$$\text{Coded system} \;\; \exp\left\{-d_{\text{free}}^2/(4N_0)\right\}$$

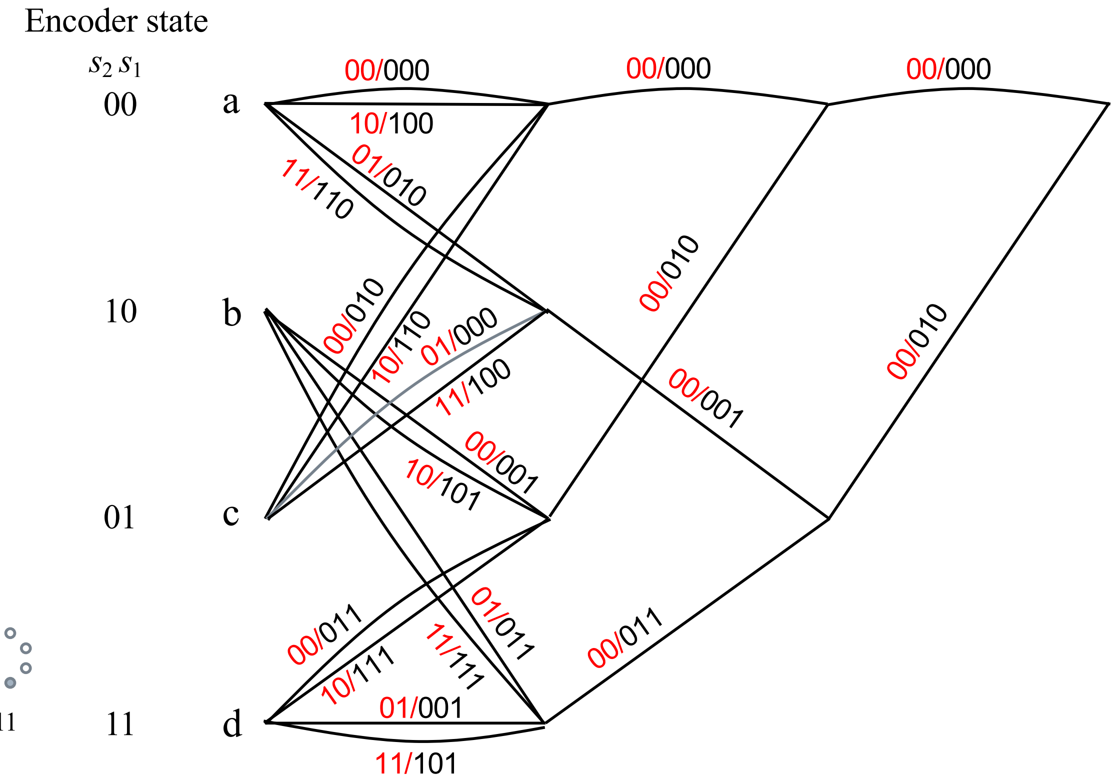
$$G_a = 10\log_{10}\left(\frac{d_{\text{free}}^2}{d_{\text{ref}}^2}\right)$$

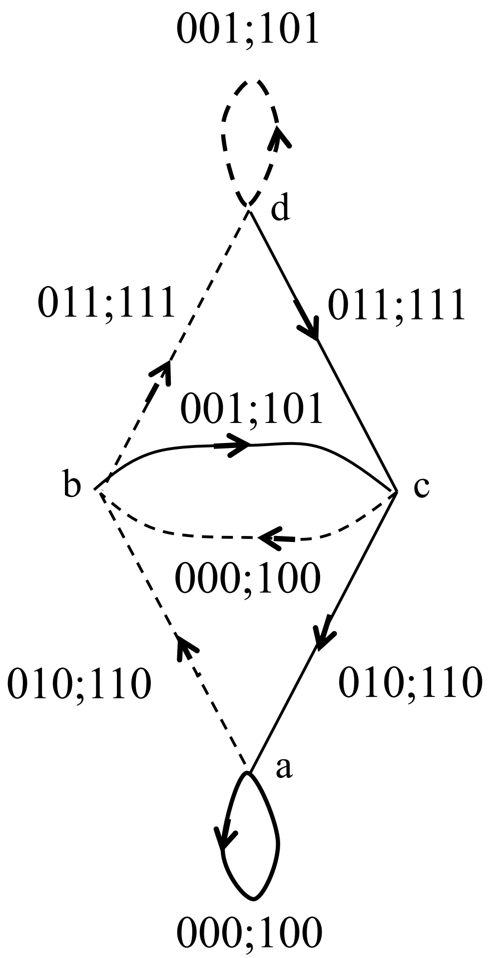
- 4-state Ungerboeck code
  - Its code rate is 2 bits/symbol; hence, it should be compared with uncoded QPSK.


3(011)
2(010)
4(100)
1(001)
5(101)
0(000)
6(110)
7(111)

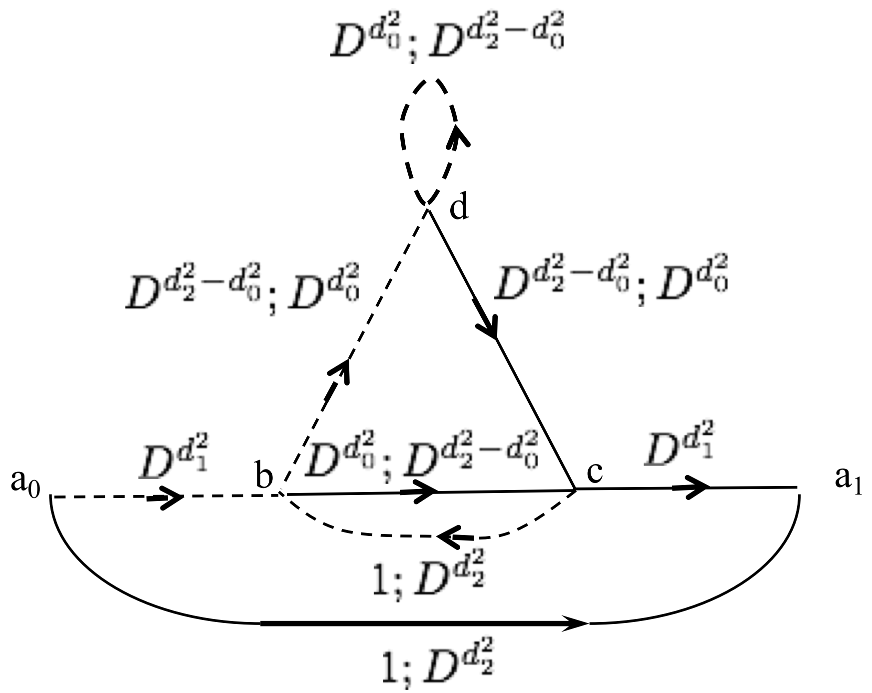

8PSK
signal mapper
s_2
s_1
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
most significant bit
Rate-1/2 convolutional code

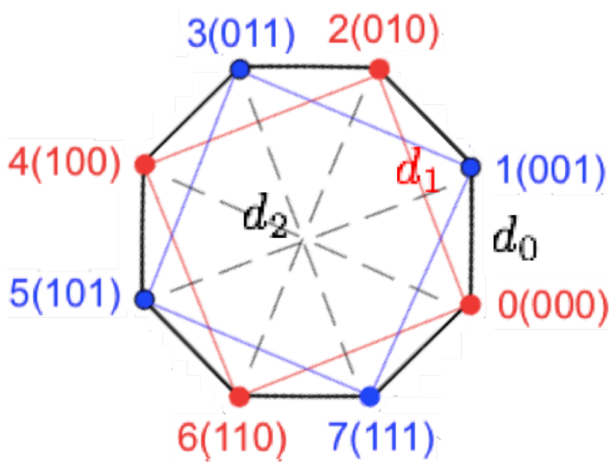

$d_0 = \sqrt{2-\sqrt{2}}$
$d_1 = \sqrt{2}$
$d_2 = 2$
0
1
000
100
010
110
001
101
011
111


Encoder state
$s_2 s_1$
00
a
10
b
01
c
11
d
00/000
10/100
01/010
11/110
00/010
10/110
01/000
11/100
00/001
10/101
00/011
10/111
01/011
11/111
01/001
11/101



Solid line : 00;10
Dashed line : 01;11



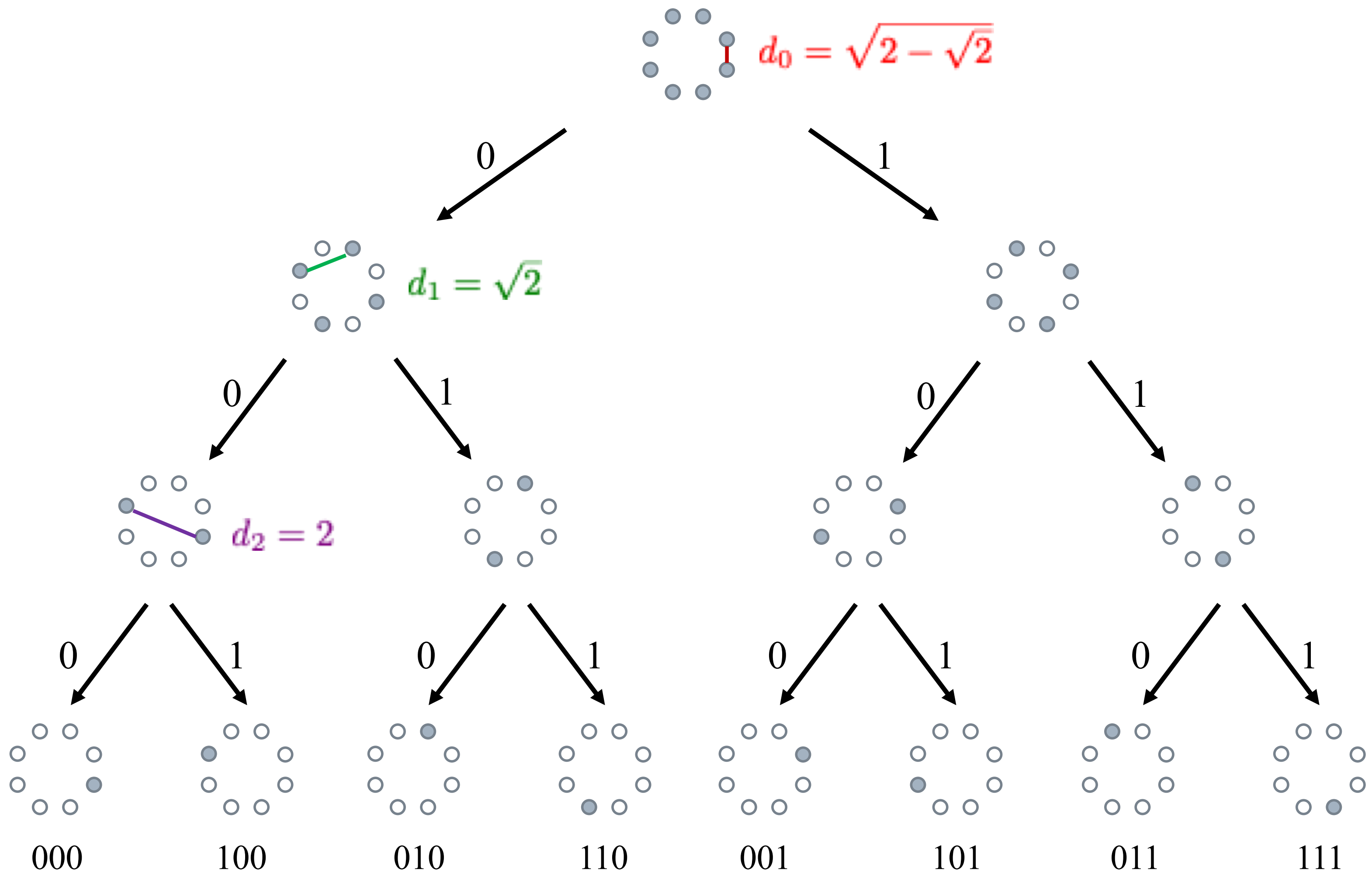


Hence, $d_{\text{free}} = d_2$.

See the example in the next slide with three input signals plus two zero tail signals.

Signals	Code signals	Distance square to "all-zero" signals
00 00 00	**000 000 000 000 000**	0
00 00 10	000 000 100 000 000	d_2^2
00 00 01	000 000 010 001 010	$2d_1^2 + d_0^2$
00 00 11	000 000 110 001 010	$2d_1^2 + d_0^2$
00 10 00	000 100 000 000 000	d_2^2
00 10 10	000 100 100 000 000	$2d_2^2$
00 10 01	000 100 010 001 010	$d_2^2 + 2d_1^2 + d_0^2$
00 10 11	000 100 110 001 010	$d_2^2 + 2d_1^2 + d_0^2$
00 01 00	000 010 001 010 000	$2d_1^2 + d_0^2$
00 01 10	000 010 101 010 000	$2d_1^2 + (d_2^2 - d_0^2)$
00 01 01	000 010 011 011 010	$2d_1^2 + 2(d_2^2 - d_0^2)$
00 01 11	000 010 111 011 010	$2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
00 11 00	000 110 001 010 000	$2d_1^2 + d_0^2$
00 11 10	000 110 101 010 000	$2d_1^2 + (d_2^2 - d_0^2)$
00 11 01	000 110 011 011 010	$2d_1^2 + 2(d_2^2 - d_0^2)$
00 11 11	000 110 111 011 010	$2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
10 00 00	100 000 000 000 000	d_2^2
10 00 10	100 000 100 000 000	$2d_2^2$
10 00 01	100 000 010 001 010	$d_2^2 + 2d_1^2 + d_0^2$
10 00 11	100 000 110 001 010	$d_2^2 + 2d_1^2 + d_0^2$
10 10 00	100 100 000 000 000	$2d_2^2$
10 10 10	100 100 100 000 000	$3d_2^2$
10 10 01	100 100 010 001 010	$2d_2^2 + 2d_1^2 + d_0^2$
10 10 11	100 100 110 001 010	$2d_2^2 + 2d_1^2 + d_0^2$
10 01 00	100 010 001 001 000	$d_2^2 + d_1^2 + 2d_0^2$
10 01 10	100 010 101 001 000	$d_2^2 + d_1^2 + (d_2^2 - d_0^2) + d_0^2$
10 01 01	100 010 011 011 010	$d_2^2 + 2d_1^2 + 2(d_2^2 - d_0^2)$
10 01 11	100 010 111 011 010	$d_2^2 + 2d_1^2 + d_0^2 + (d_2^2 - d_0^2)$
10 11 00	100 110 001 010 000	$d_2^2 + 2d_1^2 + d_0^2$
10 11 10	100 110 101 001 000	$d_2^2 + d_1^2 + (d_2^2 - d_0^2) + d_0^2$
10 11 01	100 110 011 011 010	$d_2^2 + d_1^2 + 2(d_2^2 - d_0^2) + d_1^2$
10 11 11	100 110 111 011 010	$d_2^2 + 2d_1^2 + d_0^2 + (d_2^2 - d_0^2)$

Signals	Code signals	Distance square to "all-zero" signals
01 00 00	010 001 010 000 000	$2d_1^2 + d_0^2$
01 00 10	010 001 110 000 000	$2d_1^2 + d_0^2$
01 00 01	010 001 000 001 010	$2d_1^2 + 2d_0^2$
01 00 11	010 001 100 001 010	$d_2^2 + 2d_1^2 + 2d_0^2$
01 10 00	010 101 010 000 000	$2d_1^2 + (d_2^2 - d_0^2)$
01 10 10	010 101 110 000 000	$2d_1^2 + (d_2^2 - d_0^2)$
01 10 01	010 101 000 001 010	$2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
01 10 11	010 101 100 001 010	$d_2^2 + 2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
01 01 00	010 011 011 001 000	$d_1^2 + 2(d_2^2 - d_0^2) + d_0^2$
01 01 10	010 011 111 001 000	$d_1^2 + (d_2^2 - d_0^2) + 2d_0^2$
01 01 01	010 011 001 011 010	$2d_1^2 + 2(d_2^2 - d_0^2) + d_0^2$
01 01 11	010 011 101 011 010	$2d_1^2 + 3(d_2^2 - d_0^2)$
01 11 00	010 111 011 010 000	$2d_1^2 + d_0^2 + (d_2^2 - d_0^2)$
01 11 10	010 111 111 010 000	$2d_1^2 + 2d_0^2$
01 11 01	010 111 001 011 010	$2d_1^2 + 2d_0^2 + (d_2^2 - d_0^2)$
01 11 11	010 111 101 011 010	$2d_1^2 + d_0^2 + 2(d_2^2 - d_0^2)$
11 00 00	110 001 010 000 000	$2d_1^2 + d_0^2$
11 00 10	110 001 110 000 000	$2d_1^2 + d_0^2$
11 00 01	110 001 000 001 010	$2d_1^2 + 2d_0^2$
11 00 11	110 001 100 001 010	$d_2^2 + 2d_1^2 + 2d_0^2$
11 10 00	110 101 010 000 000	$2d_1^2 + (d_2^2 - d_0^2)$
11 10 10	110 101 110 000 000	$2d_1^2 + (d_2^2 - d_0^2)$
11 10 01	110 101 000 001 010	$2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
11 10 11	110 101 100 001 010	$d_2^2 + 2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
11 01 00	110 011 011 010 000	$2d_1^2 + 2(d_2^2 - d_0^2)$
11 01 10	110 011 111 010 000	$2d_1^2 + (d_2^2 - d_0^2) + d_0^2$
11 01 01	110 011 001 011 010	$2d_1^2 + 2(d_2^2 - d_0^2) + d_0^2$
11 01 11	110 011 101 011 010	$2d_1^2 + 3(d_2^2 - d_0^2)$
11 11 00	110 111 011 001 010	$2d_1^2 + 2d_0^2 + (d_2^2 - d_0^2)$
11 11 10	110 111 111 001 010	$2d_1^2 + 3d_0^2$
11 11 01	110 111 001 011 010	$2d_1^2 + 2d_0^2 + (d_2^2 - d_0^2)$
11 11 11	110 111 101 011 010	$2d_1^2 + d_0^2 + 2(d_2^2 - d_0^2)$



$$\begin{aligned} d_{\text{free}} &= d_2 = 2 \\ d_{\text{ref}} &= d_1 = \sqrt{2} \end{aligned} \quad \Rightarrow G_a = 10\log_{10}\left(\frac{d^2_{\text{free}}}{d^2_{\text{ref}}}\right) = 3 \text{ dB}$$

- Final note
  - Asymptotic coding gain of Ungerboeck codes increases as the number of states grows.

Number of states	4	8	16	32	64	128	256	512
Coding gain (dB)	3	3.6	4.1	4.6	4.8	5	5.4	5.7

References

[1] Gottfried Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Trans, Inf. Theory*, vol. IT-28, no. 1, pp. 55-67, Jan. 1982.

[2] -----, “Trellis-coded modulation with redundant signal sets part I: Introduction,” *IEEE Comm. Magazine*, vol. 25, no. 2, pp. 5-11, Feb. 1987.

[3] -----, “Trellis-coded modulation with redundant signal sets part II: State of the art,” *IEEE Comm. Magazine*, vol. 25, no. 2, pp. 12-21, Feb. 1987.

Turbo Codes

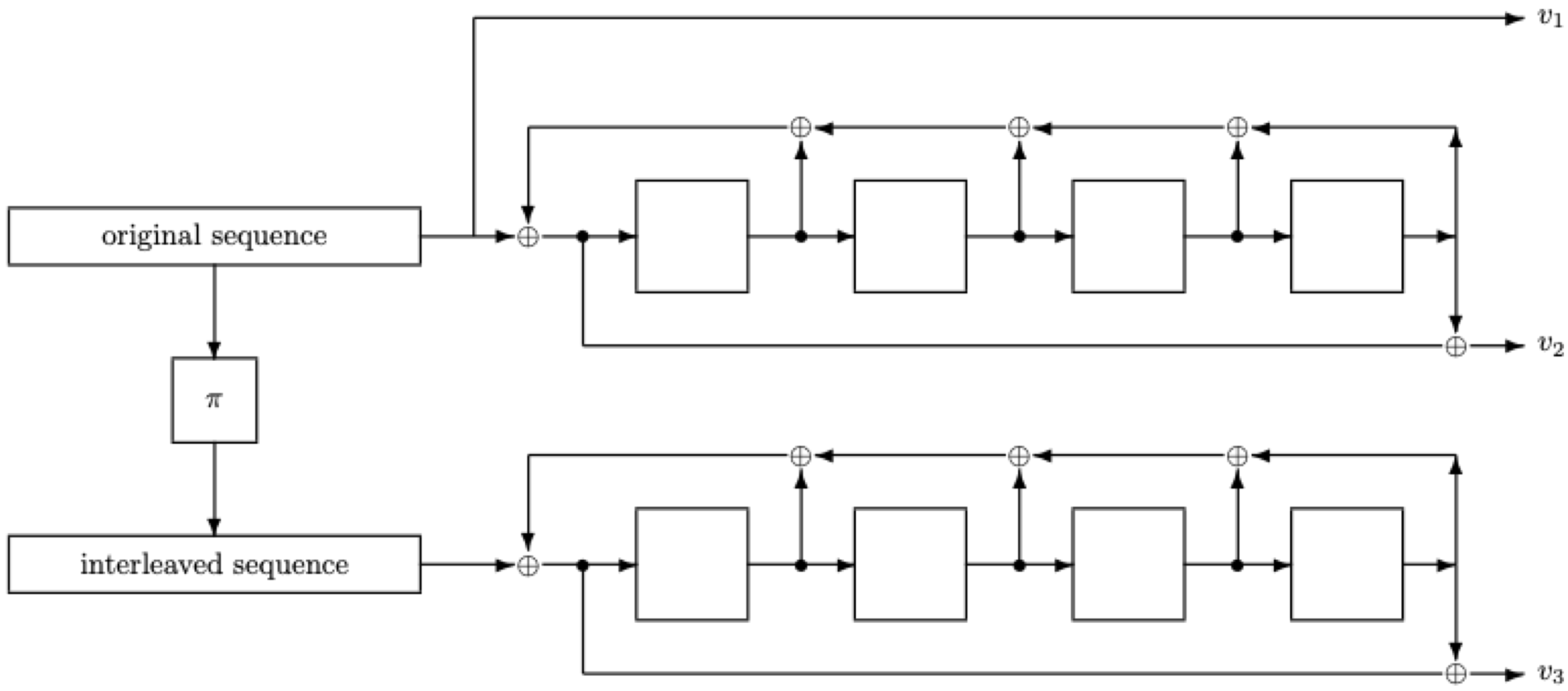
- The birth of turbo coding
  - Year: 1993
  - Authors: Berrou, Glavieux and Thitimajshima
  - Paper Title: Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes
  - Place: International Conference on Communications (ICC'93) in Geneva

Turbo Codes

- Autobiography of the inventors
  - Claude Berrou
    - Born in France in 1951
    - Received the electrical engineering degree from the Institut National Polytechnique, Grenoble, France, in 1975
    - Joined France Telecom University in 1978
  - Alain Glavieux
    - Born in France in 1949
    - Received the engineering degree from the Ecole Nationale Superieure des Telecommunications, Paris, France, in 1978
    - Joined France Telecom University in 1979
  - P. Thitimajshima
    - Received Ph.D. degree in 1993

Turbo Codes

- Structure of the turbo code encoder

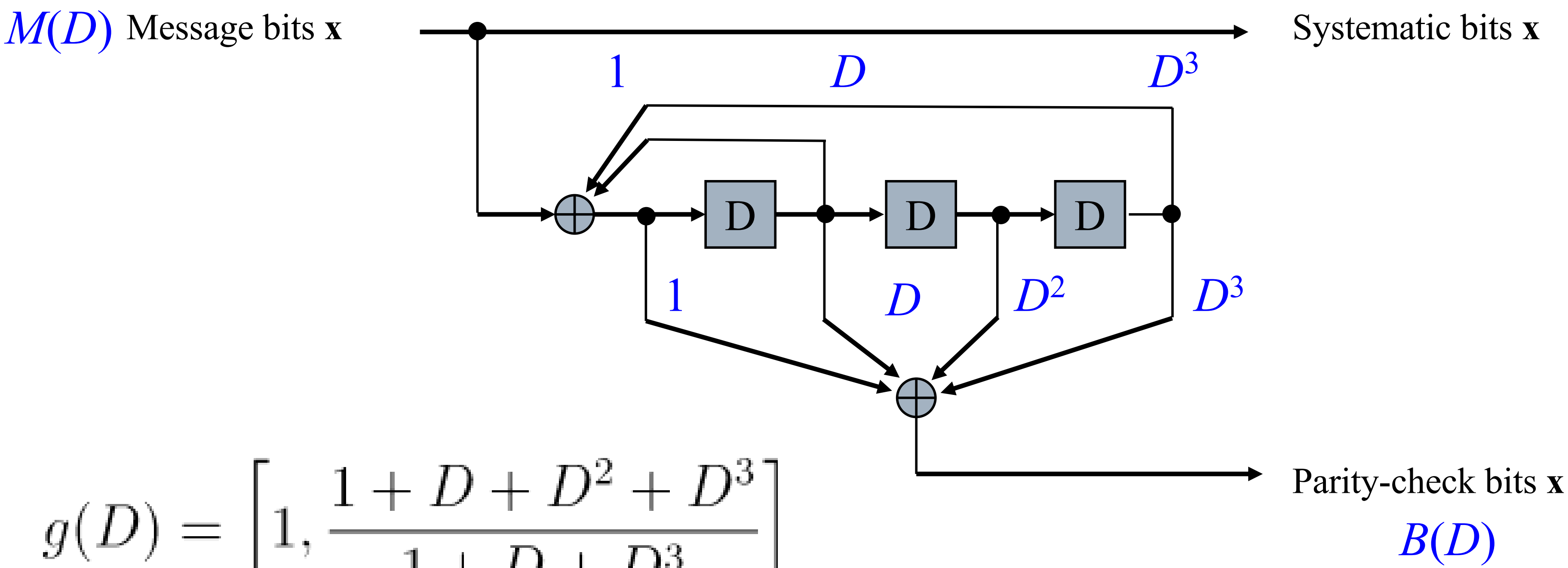


Turbo Codes

- Basic considerations
  - Add an interleaver to tie together distant bits.
  - Use *recursive* systematic convolutional (RSC) codes to make the internal state depend on the past outputs.
  - Use recursive *systematic* convolutional (RSC) codes to make the turbo-like iterative decoding possible.
    - RSC code may suffer *catastrophic error propagation* (one single output error produces an infinite number of parity errors).
  - Use *short constraint-length* RSC codes to reduce to decoding burden in each decoding iteration.

Turbo Codes

- Example: Eight-state RSC (constituent) encoder



$$g(D) = \left[1, \frac{1 + D + D^2 + D^3}{1 + D + D^3}\right]$$

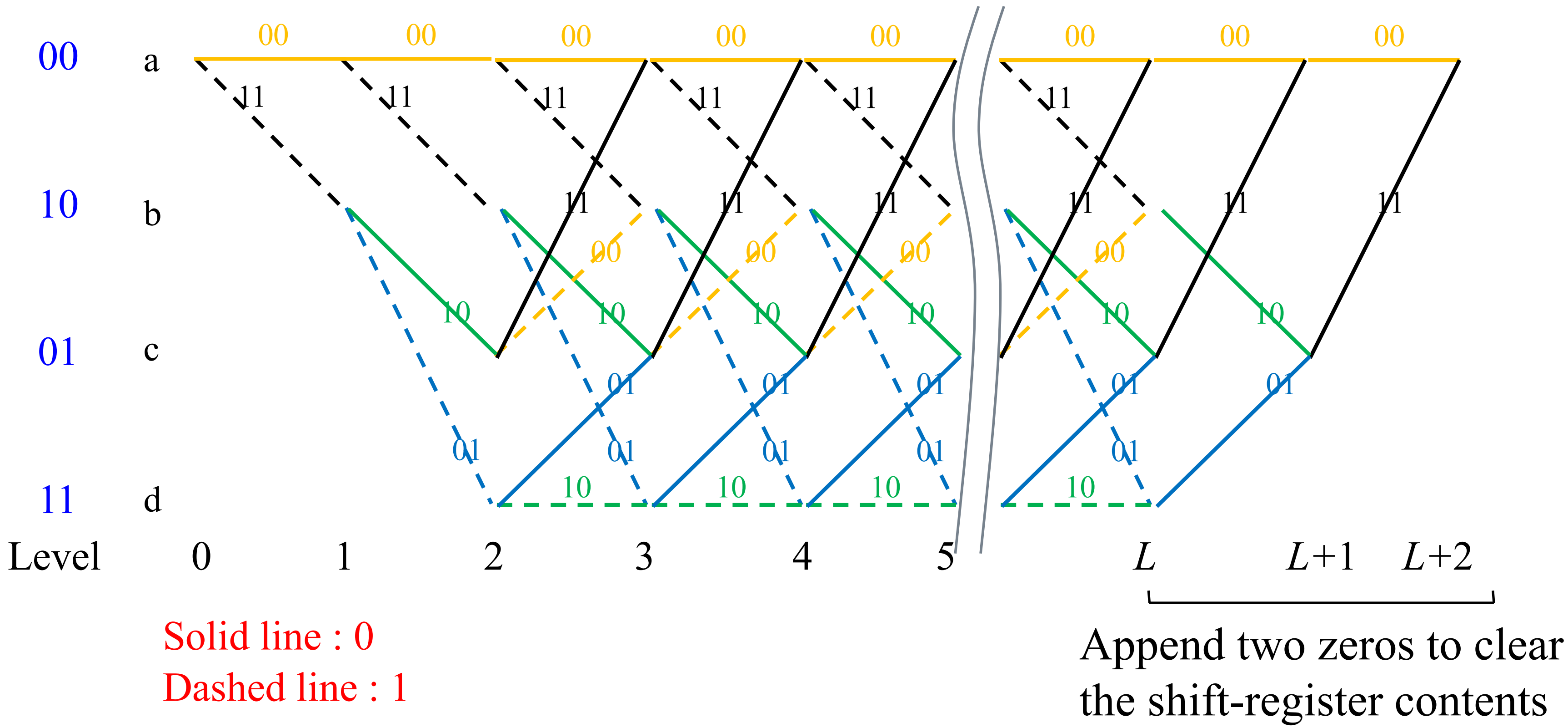
Turbo Codes

$$\frac{B(D)}{M(D)} = \frac{1 + D + D^2 + D^3}{1 + D + D^3}$$

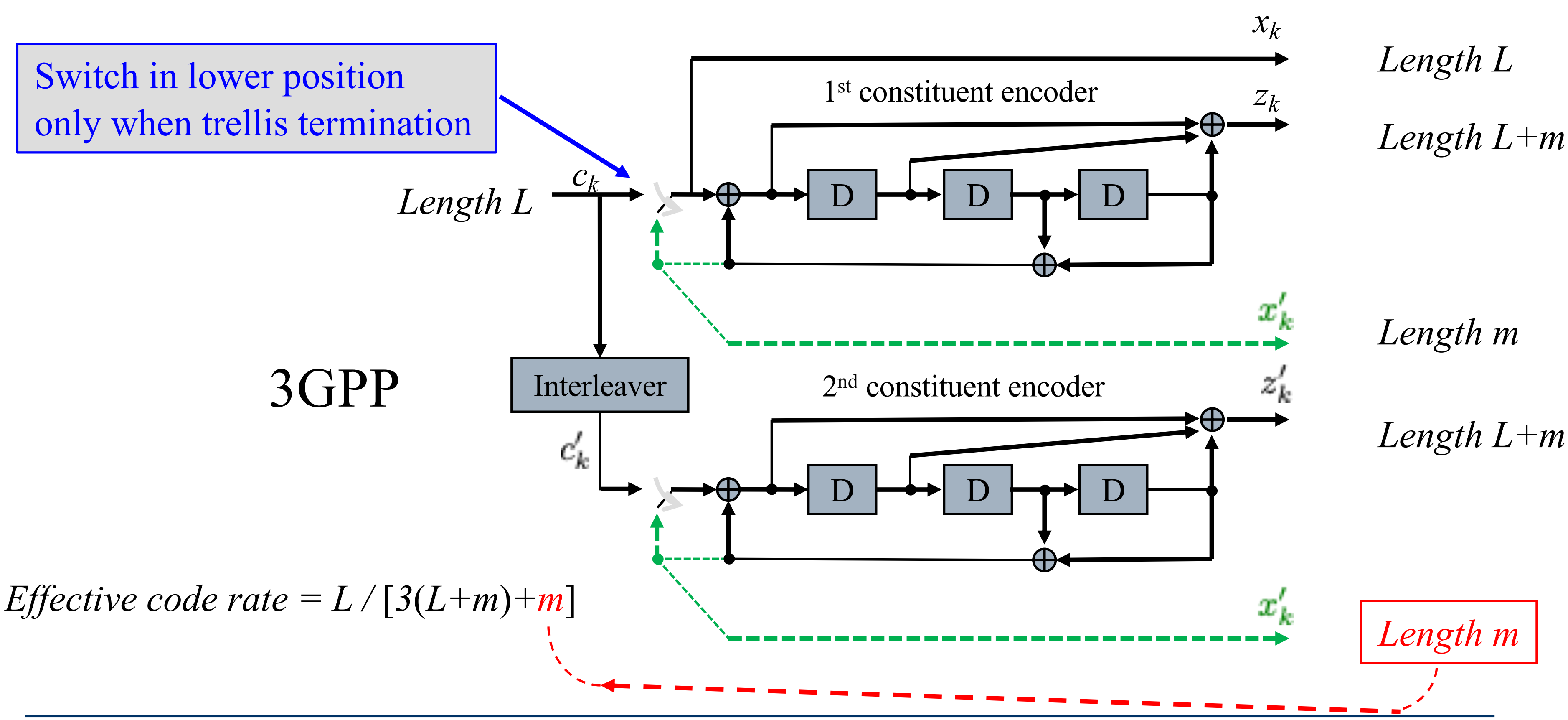
$$\Rightarrow b_i = m_i + m_{i-1} + m_{i-2} + m_{i-3} - b_{i-1} - b_{i-3}$$

☐ Remark 1: Zero tail bits

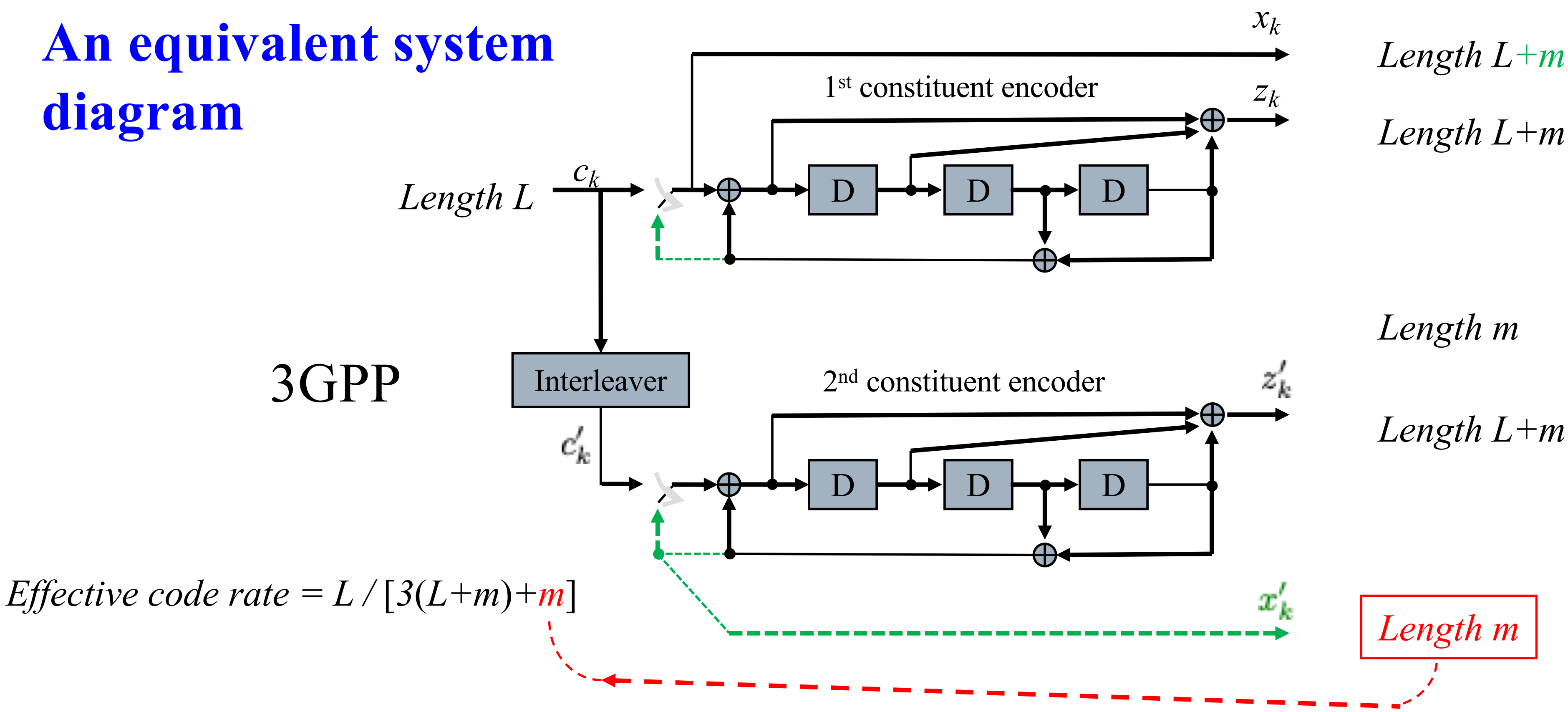
- With the pseudo-random interleaver, the zero tail bits for the first encoder may not appear to be the tail bits of the second encoder.



- Remark 1: Zero tail bits (continue)
  - With a careful design, dual clearing of the two encoder register contents can be achieved, which results in considerable performance improvement at medium to high SNRs.



An equivalent system diagram



- Remark 2: Punctured convolution code.
  - Each (2, 1) constituent encoder generates $L+m$ parity-check bits. With two constituent encoders, the system transmits $3L+4m$ bits, which reduces the code rate to approximately 1/3.

- Remark 2: Punctured convolution code (continue)
  - To improve the code rate, one can "puncture" half of the parity-check bits generated by each constituent encoder.
  - With two constituent encoders, the system transmits L information bits and approximately $L = (L/2) * 2$ parity-check bits, which reduces the code rate to around 1/2.
  - At the decoder side, since we exactly know that "no transmission" is performed in those punctured positions, we can directly "nullify" (i.e., make them zero) the corresponding received scalars.
  - For example,

x_1 and x_2 are information bits.

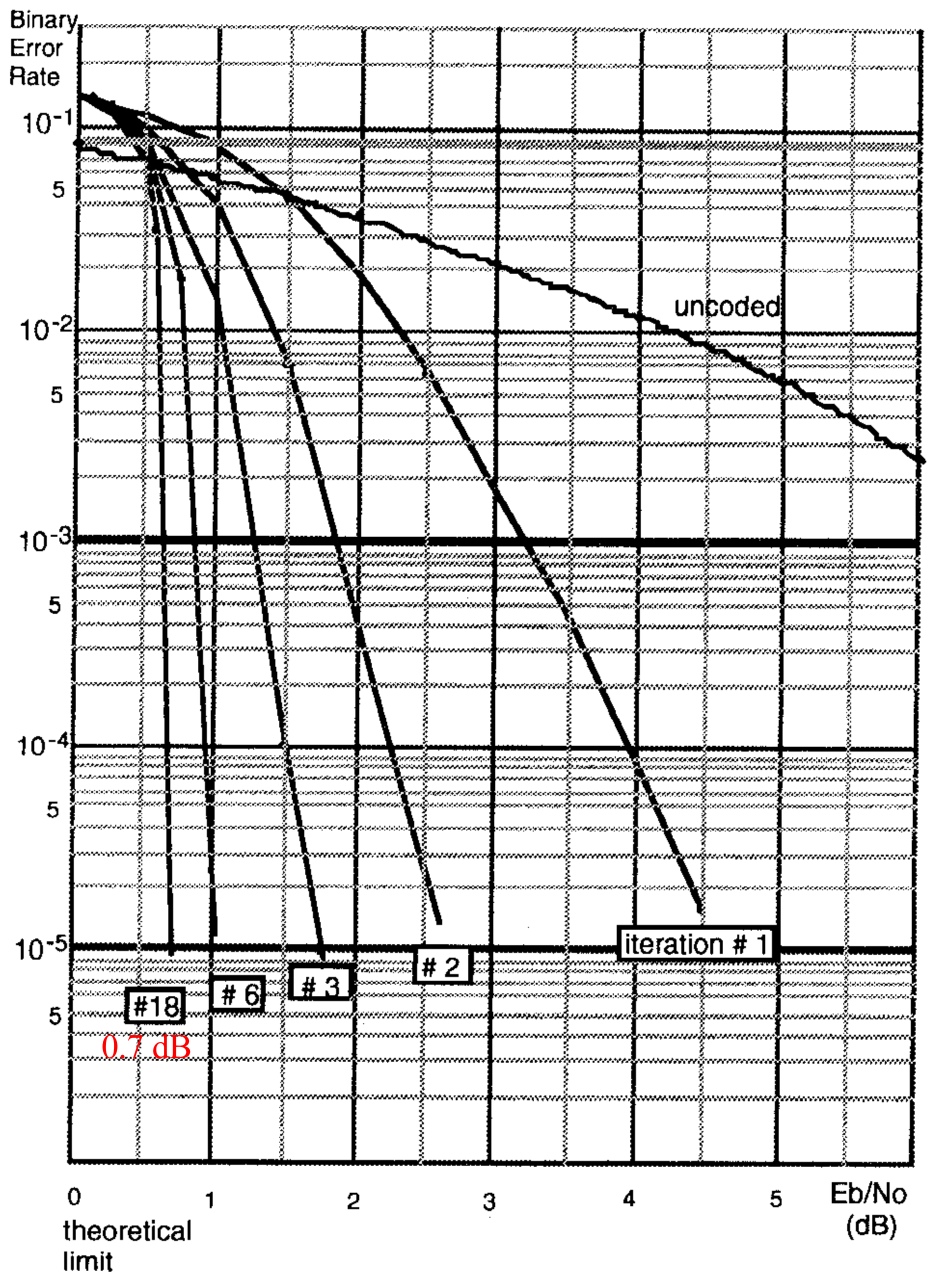
$$[r_1\ r_2\ r_3\ r_5] = [x_1\ x_2\ x_3\ x_5] + [w_1\ w_2\ w_3\ w_5]$$

Parity-check bits x_4 and x_6 are punctured.

Decoder decodes x_1 and x_2 based on $[r_1\ r_2\ r_3\ 0\ r_5\ 0]$.

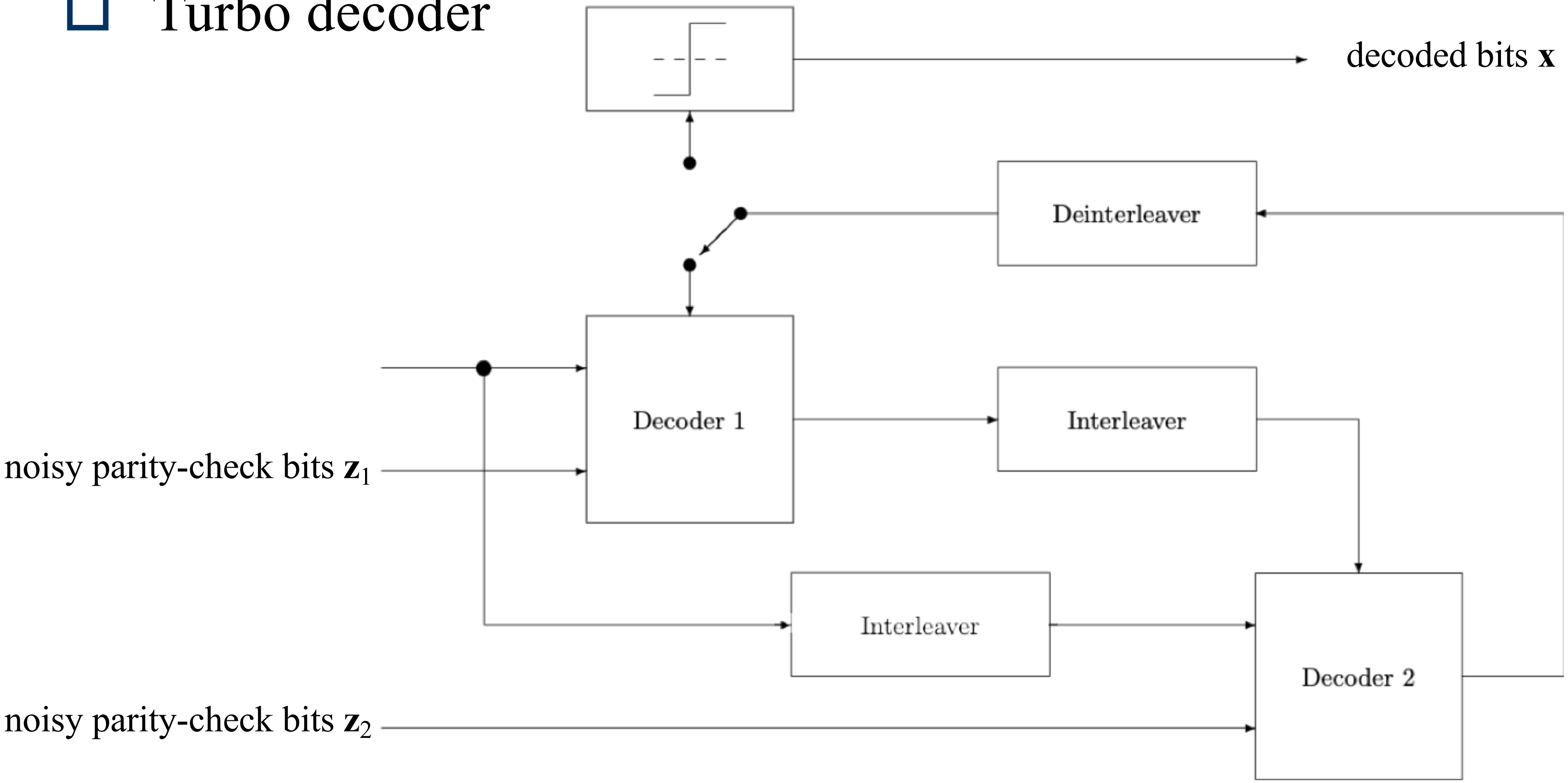
Turbo Codes

- Performance of Turbo codes
  - BER given by turbo coding with generators (37, 21) with punctuation and memory m=4, Berrou-Glavieux interleaver with size 256×256 and iterative MAP decoder (See Fig. 5 in the ICC'93 paper by Berrou *et. al*).
  - See Slide IDC6-77 for the Shannon limit 0.186 dB for (2,1) code over the binary-input AWGN channels.



Turbo Codes

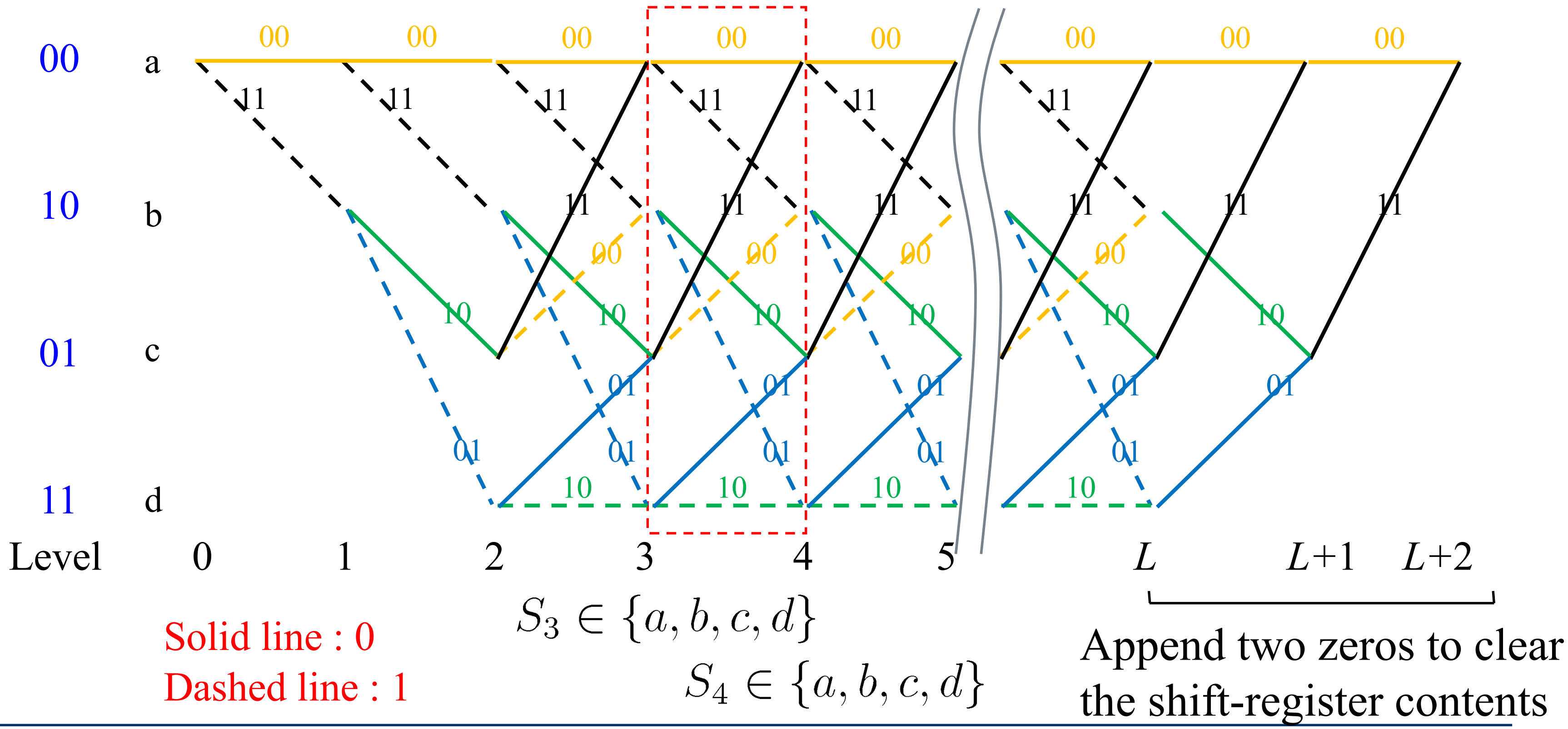
- Turbo decoder



- Turbo component decoder (BCJR algorithm or log-MAP algorithm)
  - Use to decode a code whose present state and present output are a function of the past state and current input bit.

Set of transitions corresponding to symbol 0 : $\mathcal{B}_{3,4}(0) = \{(a,a),(b,c),(c,a),(d,c)\}$

Set of transitions corresponding to symbol 1 : $\mathcal{B}_{3,4}(1) = \{(a,b),(b,d),(c,b),(d,d)\}$



- It minimizes the bit error directly rather than word error.

$$P(m_4 = 0|\boldsymbol{r}) \quad = \quad P((S_3, S_4) \in \mathcal{B}_{3,4}(0)|\boldsymbol{r})$$

Left-hand side = The probability of the 4th message bit = 0, given that the receiver receives $\boldsymbol{r}$.

Right-hand side = The probability of the encoder going through state S_3 and state S_4 in $B_{3,4}(0)$, given that the receiver receives $\boldsymbol{r}$.

$$\Rightarrow l(4) \quad = \quad \log \frac{\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(1)} P(S_3, S_4|\boldsymbol{r})}{\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(0)} P(S_3, S_4|\boldsymbol{r})}$$

$$= \quad \log \frac{\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(1)} P(S_3, S_4, \boldsymbol{r})}{\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(0)} P(S_3, S_4, \boldsymbol{r})}$$

My derivation is based on the original work and is different from the textbook.

$$
\begin{aligned}
P(S_3, S_4, \boldsymbol{r}) &= P(S_3, S_4, r_1^6, r_7^8, r_9^N) \\
&= P(r_9^N | S_3, S_4, r_1^6, r_7^8) P(S_3, S_4, r_1^6, r_7^8) \\
&= \underbrace{P(r_9^N | S_4)}_{\beta(S_4)} P(S_3, S_4, r_1^6, r_7^8)
\end{aligned}
$$

$(S_3, r_1^6, r_7^8) \to S_4 \to r_9^N$ forms a Markov chain.

$$
\begin{aligned}
P(S_3, S_4, r_1^6, r_7^8) &= P(S_3, r_1^6) P(S_4, r_7^8 | S_3, r_1^6) \\
&= \underbrace{P(S_3, r_1^6)}_{\alpha(S_3)} \underbrace{P(S_4, r_7^8 | S_3)}_{\gamma(S_3, S_4)}
\end{aligned}
$$

In the notations of α (past), β (future) and γ (now), we ignore the received vector $\boldsymbol{r}$.

$$
\Rightarrow l(4) = \log \frac{\displaystyle\sum_{(S_3, S_4) \in \mathcal{B}_{3,4}(1)} \alpha(S_3)\beta(S_4)\gamma(S_3, S_4)}{\displaystyle\sum_{(S_3, S_4) \in \mathcal{B}_{3,4}(0)} \alpha(S_3)\beta(S_4)\gamma(S_3, S_4)}
$$

$$
\begin{aligned}
\alpha(S_3) &= P(S_3, r_1^6) \\
&= \sum_{S_2 \in \{a,b,c,d\}} P(S_2, S_3, r_1^4, r_5^6) \\
&= \sum_{S_2 \in \{a,b,c,d\}} P(S_2, r_1^4) P(S_3, r_5^6 | S_2, r_1^4) \\
&= \sum_{S_2 \in \{a,b,c,d\}} P(S_2, r_1^4) P(S_3, r_5^6 | S_2) \\
&= \sum_{S_2 \in \{a,b,c,d\}} \alpha(S_2) \gamma(S_2, S_3)
\end{aligned}
$$

Initial value $\alpha(S_0) = P(S_0, r_1^0) = P(S_0) = \begin{cases} 1, & S_0 = a; \\ 0, & S_0 = b; \\ 0, & S_0 = c; \\ 0, & S_0 = d \end{cases}$

$$
\begin{aligned}
\beta(S_4) &= P(r_9^N|S_4) \\
&= \sum_{S_5\in\{a,b,c,d\}} P(S_5, r_9^{10}, r_{11}^N|S_4) \\
&= \sum_{S_5\in\{a,b,c,d\}} P(r_{11}^N|S_4, S_5, r_9^{10})P(S_5, r_9^{10}|S_4) \\
&= \sum_{S_5\in\{a,b,c,d\}} P(r_{11}^N|S_5)P(S_5, r_9^{10}|S_4) \\
&= \sum_{S_5\in\{a,b,c,d\}} \beta(S_5)\gamma(S_4, S_5)
\end{aligned}
$$

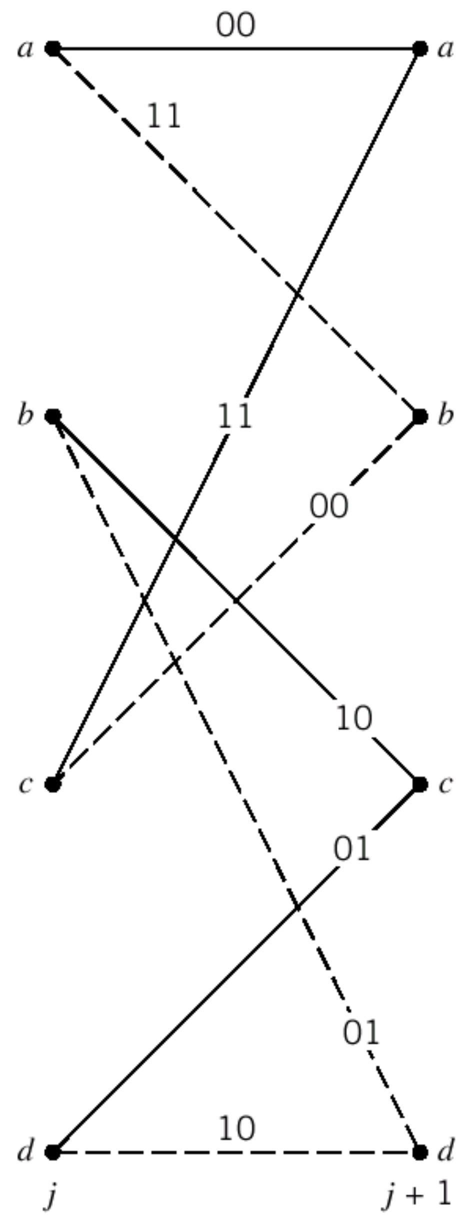
Initial value

$$
\beta(S_{L+m}) = \begin{cases} 1, & S_{L+m} = a; \\ 0, & S_{L+m} = b; \\ 0, & S_{L+m} = c; \\ 0, & S_{L+m} = d \end{cases}
$$

Note that the turbo codes use systematic code; hence, one of the code bit should be the same as the message bit

$$
\begin{aligned}
\gamma(S_3, S_4) &= P(S_4, r_7^8|S_3) \\
&= P(r_7^8|S_3, S_4)P(S_4|S_3) \\
&= \underbrace{P(r_7^8|x_7^8(S_3, S_4))}_{\text{channel}}\underbrace{P(S_4|S_3)}_{\text{prior}} = \underbrace{P(r_7^8|m_4, x_8(S_3, S_4))}_{\text{channel}}\underbrace{P(S_4|S_3)}_{\text{prior}} \\
&= \underbrace{P(r_7|m_4)}_{\text{systematic}}\underbrace{P(r_8|x_8(S_3, S_4))}_{\text{parity}}\underbrace{P(S_4|S_3)}_{\text{prior}}
\end{aligned}
$$

$$P(S_4|S_3) = \begin{cases} P(a|a) &=& P(m_4=0) \\ P(b|a) &=& P(m_4=1) \\ P(c|a) &=& 0 \\ P(d|a) &=& 0 \\ P(a|b) &=& 0 \\ P(b|b) &=& 0 \\ P(c|b) &=& P(m_4=0) \\ P(d|b) &=& P(m_4=1) \\ P(a|c) &=& P(m_4=0) \\ P(b|c) &=& P(m_4=1) \\ P(c|c) &=& 0 \\ P(d|c) &=& 0 \\ P(a|d) &=& 0 \\ P(b|d) &=& 0 \\ P(c|d) &=& P(m_4=0) \\ P(d|d) &=& P(m_4=1) \end{cases}$$



Let $\tilde{\gamma}(S_3, S_4) = \underbrace{P(r_8|x_8(S_3, S_4))}_{\text{parity}}$

$$\Rightarrow l(4) = \log \frac{\displaystyle\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(1)} \alpha(S_3)\beta(S_4)\tilde{\gamma}(S_3,S_4)P(S_4|S_3)\,P(r_7|m_4)}{\displaystyle\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(0)} \alpha(S_3)\beta(S_4)\tilde{\gamma}(S_3,S_4)P(S_4|S_3)\,P(r_7|m_4)}$$

$$= \log \frac{\displaystyle\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(1)} \alpha(S_3)\beta(S_4)\tilde{\gamma}(S_3,S_4)P(m_4=1)P(r_7|1)}{\displaystyle\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(0)} \alpha(S_3)\beta(S_4)\tilde{\gamma}(S_3,S_4)P(m_4=0)P(r_7|0)}$$

$$= \underbrace{\log\frac{P(m_4=1)}{P(m_4=0)}}_{a\ priori\ l_{\text{in}}} + \underbrace{\log\frac{P(r_7|1)}{P(r_7|0)}}_{\text{systematic}} + \underbrace{\log \frac{\displaystyle\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(1)} \alpha(S_3)\beta(S_4)\tilde{\gamma}(S_3,S_4)}{\displaystyle\sum_{(S_3,S_4)\in\mathcal{B}_{3,4}(0)} \alpha(S_3)\beta(S_4)\tilde{\gamma}(S_3,S_4)}}_{\text{extrinsic } l_{\text{ex}}}$$

(intrinsic)

Step 1: With $l_{\text{in}}(m_j)$ known, we can compute

$$P(m_j = 0) = \frac{1}{1 + \exp\{l_{\text{in}}(m_j)\}} \text{ and } P(m_j = 1) = \frac{\exp\{l_{\text{in}}(m_j)\}}{1 + \exp\{l_{\text{in}}(m_j)\}}$$

for each $1 \le j \le L$. We can in turn compute $\gamma(S_j, S_{j+1})$ for all $(S_j, S_{j+1}) \in \{a, b, c, d\}^2$ and for each $0 \le j \le j + m$.

Step 2: With γ available, we can recursively compute α and β in the forward and backward fashions, respectively.

Step 3: With α, β and γ ready, we can compute

$$l(j) = \log \frac{P(m_j = 1|\boldsymbol{r})}{P(m_j = 0|\boldsymbol{r})} = \log \frac{\sum_{(S_{j-1}, S_j) \in \mathcal{B}_{j-1,j}(1)} \alpha(S_{j-1})\beta(S_j)\gamma(S_{j-1}, S_j)}{\sum_{(S_{j-1}, S_j) \in \mathcal{B}_{j-1,j}(0)} \alpha(S_{j-1})\beta(S_j)\gamma(S_{j-1}, S_j)}$$

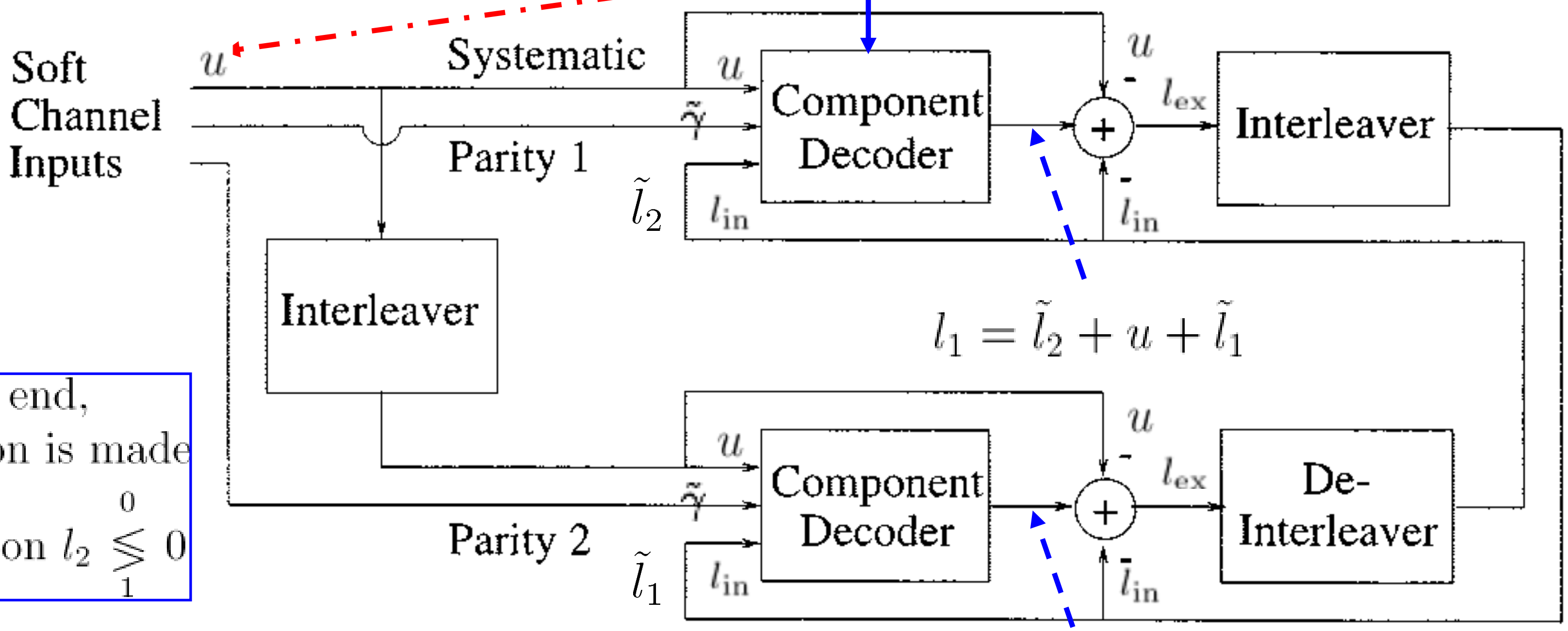
Step 4: Calculate $l_{\text{ex}}(j) = l(j) - \log \dfrac{P(r_{2j-1}|1)}{P(r_{2j-1}|0)} - l_{\text{in}}(j)$.

$$\text{Observe } l(j) = l_{\text{in}}(j) + \log \frac{P(r_{2j-1}|1)}{P(r_{2j-1}|0)} + l_{\text{ex}}(j).$$

Intuition: Recursion between these two

(Slide IDC 8-33) Compute γ based on u, $\tilde{\gamma}$, l_{in}
(Slides IDC 8-32&33) Compute recursively α, β
(Slides IDC 8-36) Compute l (Step 3)

$$\begin{aligned} u_j &= \log \frac{P(r_{2j-1}|1)}{P(r_{2j-1}|0)} \\ &= \log \frac{\exp\left\{-\frac{(r_{2j-1}-(+1))^2}{2\sigma^2}\right\}}{\exp\left\{-\frac{(r_{2j-1}-(-1))^2}{2\sigma^2}\right\}} \\ &= \frac{2}{\sigma^2} r_{2j-1} \end{aligned}$$



$$l_1 = \tilde{l}_2 + u + \tilde{l}_1$$

In the end, decision is made based on $l_2 \underset{1}{\overset{0}{\lessgtr}} 0$

$$l_2 = \tilde{l}_1 + u + \tilde{l}_2$$

- Turbo coding, although quite impressive in performance, is designed based on an empirical intuition.
  - For example, Berrou and Glaviexu wrote in their 1996 T-COM paper that

> - *... for very low SNRs, the BER can sometimes increase during the iterative decoding process. In order to overcome this effect, the extrinsic information* $\tilde{l}_1$ *(resp.* $\tilde{l}_2$*) has been divided by* $(1+\theta|\tilde{l}_1|)$ *(resp.* $(1+\theta|\tilde{l}_2|)$*).* θ *acts as a stability factor and its value of* 0.15 *was adopted after several simulation tests at* $E_b/N_0 = 0.7$ *dB*.... [1, pp. 1270]

[1] Claude Berrou and Alain Glavieux, "Near optimal error correcting coding and decoding: Turbo-codes," *IEEE Trans. Comm.*, vol. 44, no. 10, pp. 1261-1271, Oct. 1996.

Low-Density Parity-Check (LDPC) Codes

- LDPC codes (also known as Gallager codes) are also iteratively decodable.
- Its advantages over turbo coding technique are
  - absence of low-weight codewords;
    - With a careless interleaver design, a turbo code may have low weight codewords, which is the main cause for error floor.
  - And iterative decoding with a lower complexity.

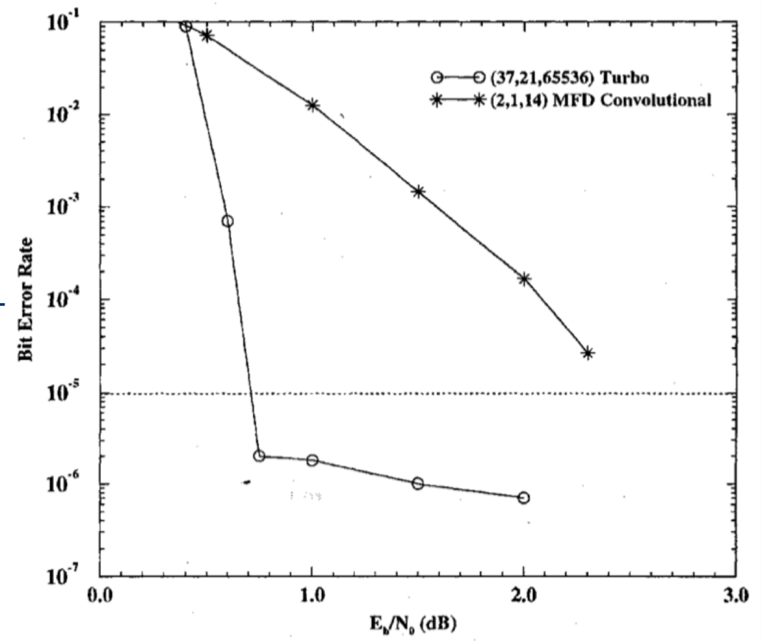
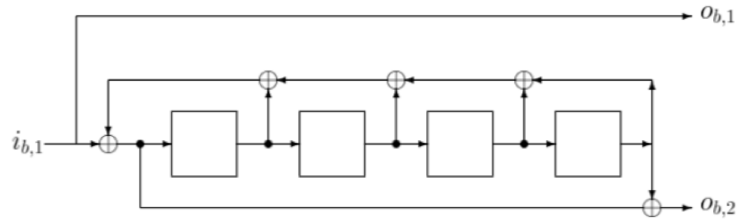


Fig. 1. Simulation results for a (37, 21, 65536) Turbo code and a (2, 1, 14) MFD convolutional code.

L. C. Perez, J. Seghers and D. J. Costello, "A Distance Spectrum Interpretation of Turbo Codes," *IEEE Trans. Info. Theory*, pp. 1698-1709, Nov. 1996.



$$G(D) = \begin{bmatrix} 1 & \dfrac{D^4+1}{D^4+D^3+D^2+D+1} \end{bmatrix} = \begin{bmatrix} 1 & \dfrac{(10,001)_2}{(11,111)_2} \end{bmatrix} = \begin{bmatrix} 1 & \dfrac{(2,1)_8}{(3,7)_8} \end{bmatrix}$$

Low-Density Parity-Check Codes

- In notation, a regular LDPC code (with parity-check matrix $\mathbf{H}_{(n-k)\times n}$) is usually denoted by three tuple (n, t_c, t_r).
  - n = block length
  - t_c = number of 1s in each column of n bits
  - t_r = number of 1s in each row of (n–k) bits with $t_r > t_c$
  - It is not necessary to specify k since

$$(\#\text{ of 1s}) = nt_c = (n-k)t_r \Rightarrow \frac{t_c}{t_r} = 1 - \frac{k}{n}$$

- Example: $(n, t_c, t_r) = (10, 3, 5)$.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}_{(10-4)\times 10}$$

$$\frac{k}{n} = 1 - \frac{t_c}{t_r} = 1 - \frac{3}{5} = \frac{2}{5}$$

Low-Density Parity-Check Codes

- How to find the generator matrix for a given parity-check matrix for systematic LDPC codes?

$$\boldsymbol{c} = [\boldsymbol{b} : \boldsymbol{m}]$$

$$\text{where } \begin{cases} \boldsymbol{b} \text{ are parity-check bits} \\ \boldsymbol{m} \text{ are message bits} \end{cases}$$

$$\mathbf{H}^T_{n\times(n-k)} = \begin{bmatrix}\mathbf{H}_1\\ \cdots \\ \mathbf{H}_2\end{bmatrix} \Rightarrow [\boldsymbol{b} : \boldsymbol{m}] \begin{bmatrix}\mathbf{H}_1\\ \cdots \\ \mathbf{H}_2\end{bmatrix} \Rightarrow \boldsymbol{b}\mathbf{H}_1 + \boldsymbol{m}\mathbf{H}_2 = \mathbf{0}$$

The generator matrix of a systematic code (including LDPC codes) must be of the shape

$$\mathbf{G}_{k\times n} = \left[\mathbf{P}_{k\times(n-k)} \;\vdots\; \mathbf{I}_k\right] \Rightarrow \boldsymbol{m}_{1\times k}\mathbf{P}_{k\times(n-k)} = \boldsymbol{b}_{1\times(n-k)}$$

This concludes to:

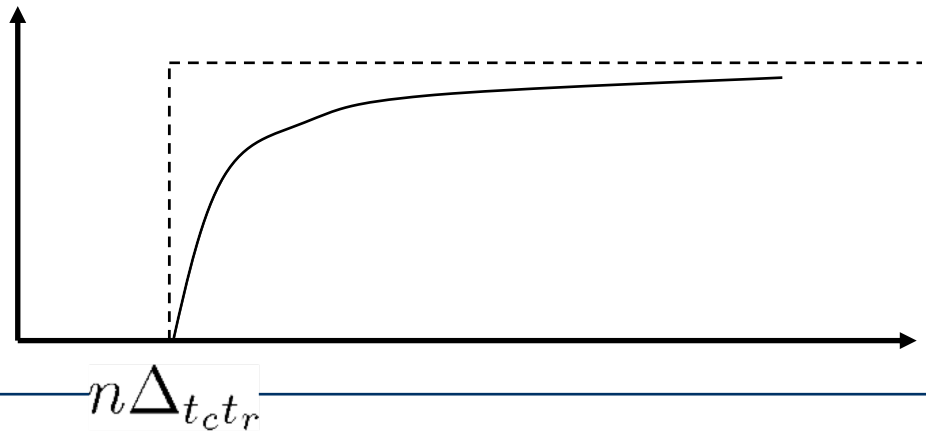
$$\boldsymbol{m}\mathbf{P}\mathbf{H}_1 + \boldsymbol{m}\mathbf{H}_2 = 0 \;\Rightarrow \mathbf{P} = \mathbf{H}_2\mathbf{H}_1^{-1} \;\Rightarrow \mathbf{G} = \left[\mathbf{H}_2\mathbf{H}_1^{-1} \;\vdots\; \mathbf{I}_k\right]$$

Low-Density Parity-Check Codes

- Remarks
  - Low-density parity-check code gets its name since the number of 1s in each row and column is small (low-density).
  - If the number of 1s in each row and also in each column is fixed, the LDPC code is said to be *regular*.
  - Under regularity, the inverse matrix of $\mathbf{H}_1$ may be difficult to make to exist.
  - Hence, some "manipulation" or even allowing some "irregularity" is sometimes necessary.

Low-Density Parity-Check Codes

- Minimum distance of LDPC codes
  - By uniformly selecting codeword pairs, the pairwise distance becomes a random variable, for which the cumulative distribution function (cdf) can be empirically plotted.
    - It is shown that this cdf can be overbounded by a unit step function as shown below.



Low-Density Parity-Check Codes

t_c	t_r	Code rate k/n	$\Delta_{t_c t_r}$
5	6	0.167	0.255
4	5	0.2	0.210
3	4	0.25	0.122
4	6	0.333	0.129
3	5	0.4	0.044
3	6	0.5	0.023

$$(\#\text{ of 1s}) = nt_c = (n-k)t_r \Rightarrow \frac{t_c}{t_r} = 1 - \frac{k}{n}$$

Low-density parity-check codes

- Probabilistic decoding of LDPC codes (Mackay and Neal, 1996)
  - In the form of belief propagation or message passing.
  - Forney's factor graph (Bipartite graph)

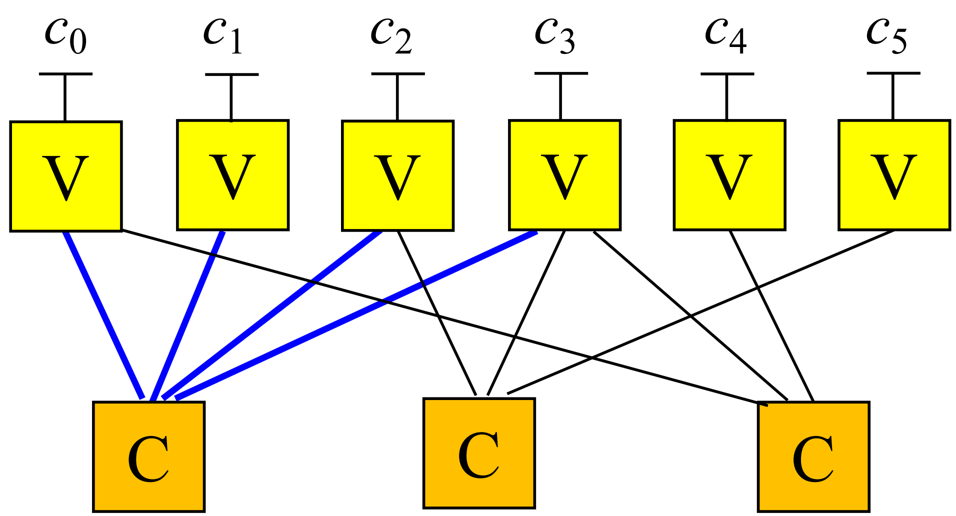
Have $(n-k)=(6-3)$ parity-check equations

$$[c_0, c_1, \ldots, c_5]\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}_{6\times 3} = \mathbf{0}_{1\times 3}$$

Need to solve 6 variables $[c_0, c_1, \ldots, c_5]$

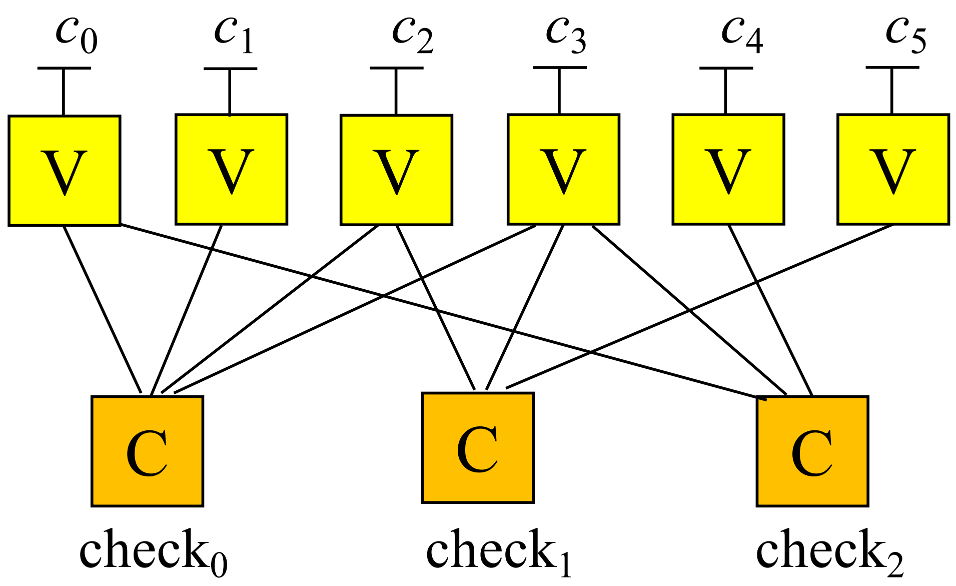
Variable nodes

Check nodes



Map {0, 1} to {1, -1} and thus change "xor" to "product".
Abuse the notation by retaining c_j to be the information in {1, -1}.
Hence, for each check node, we should have even number of -1.
In other words,

$$\begin{cases} \text{Check}_0 = c_0 c_1 c_2 c_3 = 1; \\ \text{Check}_1 = c_2 c_3 c_5 = 1; \\ \text{Check}_2 = c_0 c_3 c_4 = 1; \end{cases}$$



$\text{Bit}(0) = \{0, 2\}$, $\text{Bit}(1) = \{0\}$, $\text{Bit}(2) = \{0, 1\}$, $\text{Bit}(3) = \{0, 1, 2\}$, $\text{Bit}(4) = \{2\}$, $\text{Bit}(5) = \{1\}$
$\text{Check}(0) = \{0, 1, 2, 3\}$, $\text{Check}(1) = \{2, 3, 5\}$, $\text{Check}(2) = \{0, 3, 4\}$

$Q^x_{i,j}$ = probability that Check$_i$ = 1, with $c_j = x$ and the messages that it receives from the variable nodes connected to it.

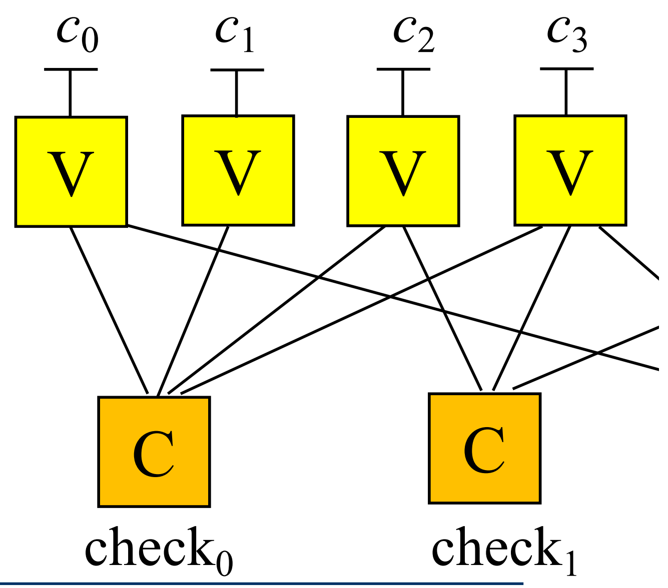
$$\Rightarrow E[c_1c_2c_3] = 1 \times Q^1_{0,0} + (-1) \times Q^{-1}_{0,0} = Q^1_{0,0} - Q^{-1}_{0,0}$$

$P^x_{i,j}$ = probability that $c_j = x$, given that the information derived via all checks connected to c_j except Check$_i$.

$$\Rightarrow E[c_1c_2c_3] = E[c_1]E[c_2]E[c_3] \quad \text{(Assume independence among } \{c_j\}.)$$
$$= (P^1_{0,1} - P^{-1}_{0,1})(P^1_{0,2} - P^{-1}_{0,2})(P^1_{0,3} - P^{-1}_{0,3})$$

$$\begin{cases} Q^1_{0,0} - Q^{-1}_{0,0} = \prod_{j \in \text{Check}(0)\setminus\{0\}} (P^1_{0,j} - P^{-1}_{0,j}) \\ Q^1_{0,0} + Q^{-1}_{0,0} = 1 \end{cases}$$

$$\Rightarrow \begin{cases} Q^1_{0,0} = \frac{1}{2}\left(1 + \prod_{j \in \text{Check}(0)\setminus\{0\}} (P^1_{0,j} - P^{-1}_{0,j})\right) \\ Q^{-1}_{0,0} = \frac{1}{2}\left(1 - \prod_{j \in \text{Check}(0)\setminus\{0\}} (P^1_{0,j} - P^{-1}_{0,j})\right) \end{cases}$$



- This summarizes to the so-called Horizontal step:

Horizontal step: Update $Q^1_{i,j} = \frac{1}{2}\left(1 + \prod_{k \in \text{Check}(i) \setminus \{j\}} (2P^1_{i,k} - 1)\right)$

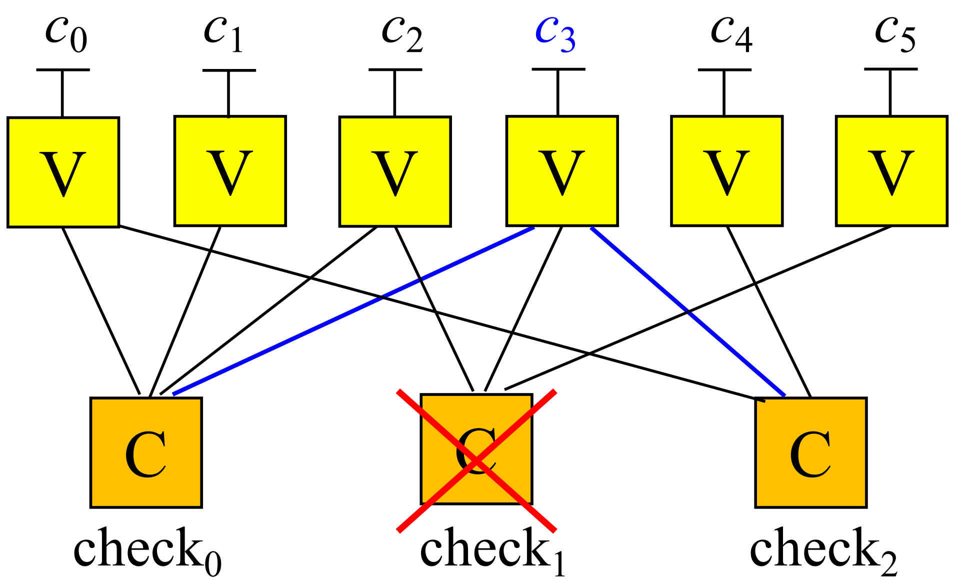
$$\begin{bmatrix} P^1_{0,0} & P^1_{1,0} & P^1_{2,0} \\ P^1_{0,1} & P^1_{1,1} & P^1_{2,1} \\ P^1_{0,2} & P^1_{1,2} & P^1_{2,2} \\ P^1_{0,3} & P^1_{1,3} & P^1_{2,3} \\ P^1_{0,4} & P^1_{1,4} & P^1_{2,4} \\ P^1_{0,5} & P^1_{1,5} & P^1_{2,5} \end{bmatrix} \Rightarrow \begin{bmatrix} Q^1_{0,0} & Q^1_{1,0} & Q^1_{2,0} \\ Q^1_{0,1} & Q^1_{1,1} & Q^1_{2,1} \\ Q^1_{0,2} & Q^1_{1,2} & Q^1_{2,2} \\ Q^1_{0,3} & Q^1_{1,3} & Q^1_{2,3} \\ Q^1_{0,4} & Q^1_{1,4} & Q^1_{2,4} \\ Q^1_{0,5} & Q^1_{1,5} & Q^1_{2,5} \end{bmatrix}$$

- Next we move on to the Vertical step:

$Q_{i,j}^x$ = probability that Check$_i$ = 1, with $c_j = x$ and the messages that it receives from the variable nodes connected to it.

$P_{i,j}^x$ = probability that $c_j = x$, given that the information derived via all checks connected to c_j except Check$_i$.

> $P_{1,3}^1$ should be proportional to $p_3^1 Q_{0,3}^1 Q_{2,3}^1$, and $P_{1,3}^{-1}$ should be proportional to $p_3^{-1} Q_{0,3}^{-1} Q_{2,3}^{-1}$, where p_j^1 and p_j^{-1} are the initial probabilities of $c_j = 1$ and $c_j = -1$, respectively.



Vertical step: Update

$$P^1_{i,j} = \frac{p^1_j \prod_{k\in \text{Bit}(j)\backslash\{i\}} Q^1_{k,j}}{p^1_j \prod_{k\in \text{Bit}(j)\backslash\{i\}} Q^1_{k,j} + (1-p^1_j) \prod_{k\in \text{Bit}(j)\backslash\{i\}} (1-Q^1_{k,j})}$$

$$\begin{bmatrix} Q^1_{0,0} & Q^1_{1,0} & Q^1_{2,0} \\ Q^1_{0,1} & Q^1_{1,1} & Q^1_{2,1} \\ Q^1_{0,2} & Q^1_{1,2} & Q^1_{2,2} \\ Q^1_{0,3} & Q^1_{1,3} & Q^1_{2,3} \\ Q^1_{0,4} & Q^1_{1,4} & Q^1_{2,4} \\ Q^1_{0,5} & Q^1_{1,5} & Q^1_{2,5} \end{bmatrix} \Rightarrow \begin{bmatrix} P^1_{0,0} & P^1_{1,0} & P^1_{2,0} \\ P^1_{0,1} & P^1_{1,1} & P^1_{2,1} \\ P^1_{0,2} & P^1_{1,2} & P^1_{2,2} \\ P^1_{0,3} & P^1_{1,3} & P^1_{2,3} \\ P^1_{0,4} & P^1_{1,4} & P^1_{2,4} \\ P^1_{0,5} & P^1_{1,5} & P^1_{2,5} \end{bmatrix}$$

$$P^1_j = \frac{p^1_j \prod_{k\in \text{Bit}(j)} Q^1_{k,j}}{p^1_j \prod_{k\in \text{Bit}(j)} Q^1_{k,j} + (1-p^1_j) \prod_{k\in \text{Bit}(j)} (1-Q^1_{k,j})} \quad \text{and} \quad p^1_j = P^1_j$$

Initialization:

$$P_{i,j}^1 = p_j^1$$

Horizontal step:

$$Q_{i,j}^1 = \frac{1}{2}\left(1 + \prod_{k \in \text{Check}(i)\setminus\{j\}} (2P_{i,k}^1 - 1)\right)$$

Vertical step:

$$P_{i,j}^1 = \frac{p_j^1 \prod_{k \in \text{Bit}(j)\setminus\{i\}} Q_{k,j}^1}{p_j^1 \prod_{k \in \text{Bit}(j)\setminus\{i\}} Q_{k,j}^1 + (1 - p_j^1) \prod_{k \in \text{Bit}(j)\setminus\{i\}} (1 - Q_{k,j}^1)}$$

recursion

Decision step:

$$P_j^1 = \frac{p_j^1 \prod_{k \in \text{Bit}(j)} Q_{k,j}^1}{(1 - p_j^1) \prod_{k \in \text{Bit}(j)} (1 - Q_{k,j}^1) + p_j^1 \prod_{k \in \text{Bit}(j)} Q_{k,j}^1} \mathop{\gtrless}_{0}^{1} 1 - P_j^1$$

Termination:

If $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$, the algorithm stops;
else $p_j^1 = P_j^1$ and go to Horizontal step.

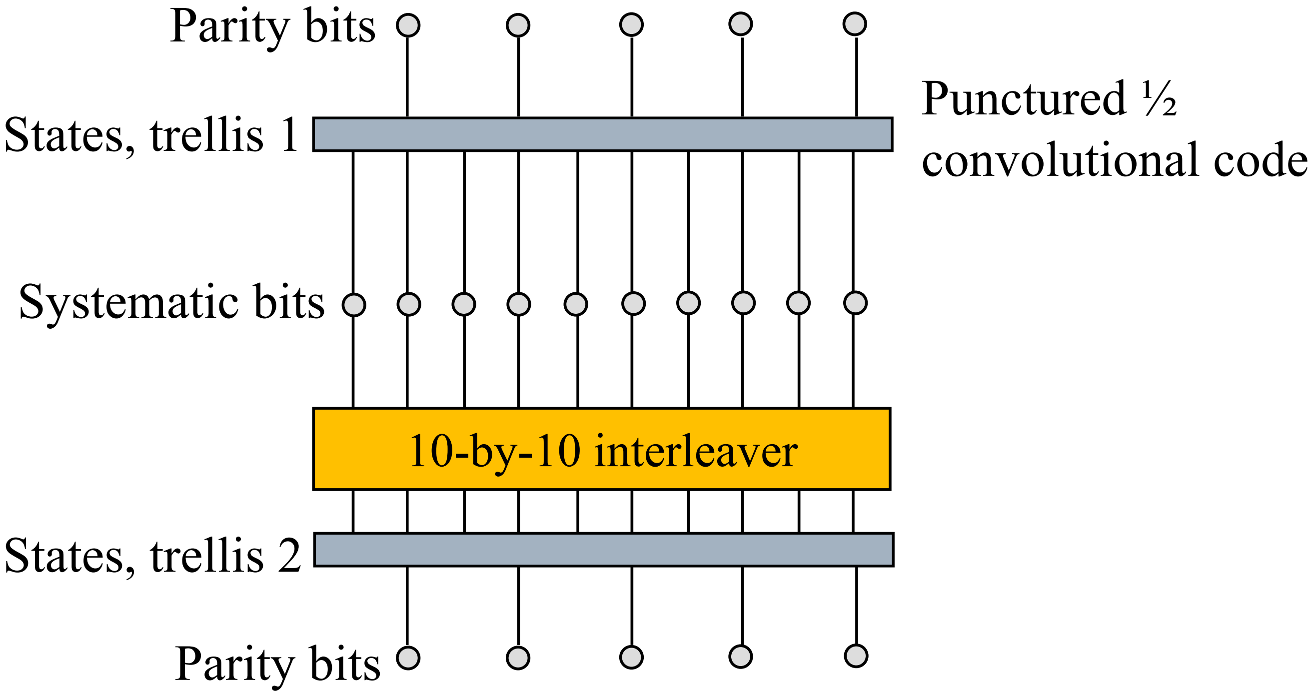
- Final remark
  - Regular LDPC codes do not appear to come as close to Shannon's limit as their turbo code counterparts.
  - Hence, irregular LDPC codes are more popular.
    - The number of 1s in each column may vary.
    - The number of 1s in each row may vary.

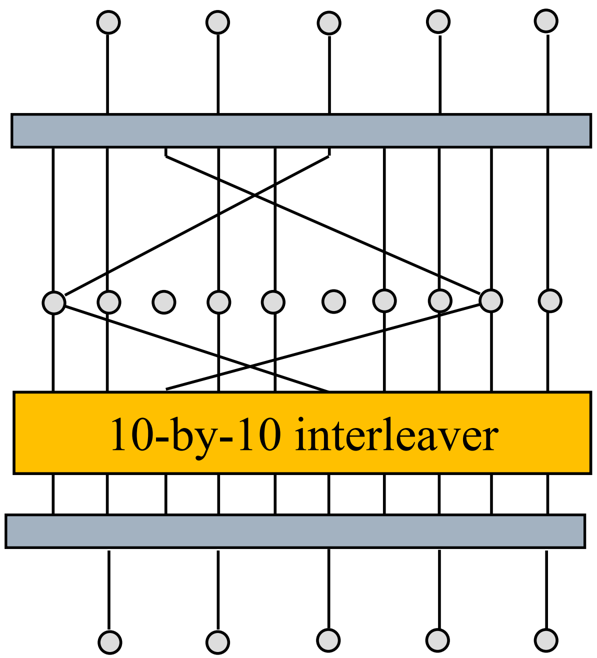
Irregular LDPC Codes

- The performance of turbo codes and LDPC codes can be further improved by "irregularity".
  - By "irregularity", we mean that each systematic bit is not used the same number of times.
  - For example, regular turbo code indicates that each systematic bit is used twice in the encoding process.

Regular (20, 10) turbo code

Irregular (18, 8) turbo code
(Bits 0 and 6 are used four times, while bits 1, 2, 3, 4, 5, 7 are used only twice.)





Irregular LDPC Codes

- Why irregularity gives better performance?
  - The codeword is "bit-wisely dependent".

    000000
    000111
    111000
    111111

  - If we give a much better estimation on certain positions, e.g., bit 0 and bit 4 in the above example, then the transmitted codeword may be more easily identified (via iterative decoding).

Irregular LDPC Codes

(See Figure 10.33 in textbook.) (Code rate = ½)