Forward Error Correction Coding

EE564: Digital Communication and Coding Systems

Keith M. Chugg
Spring 2017

USC University of Southern California

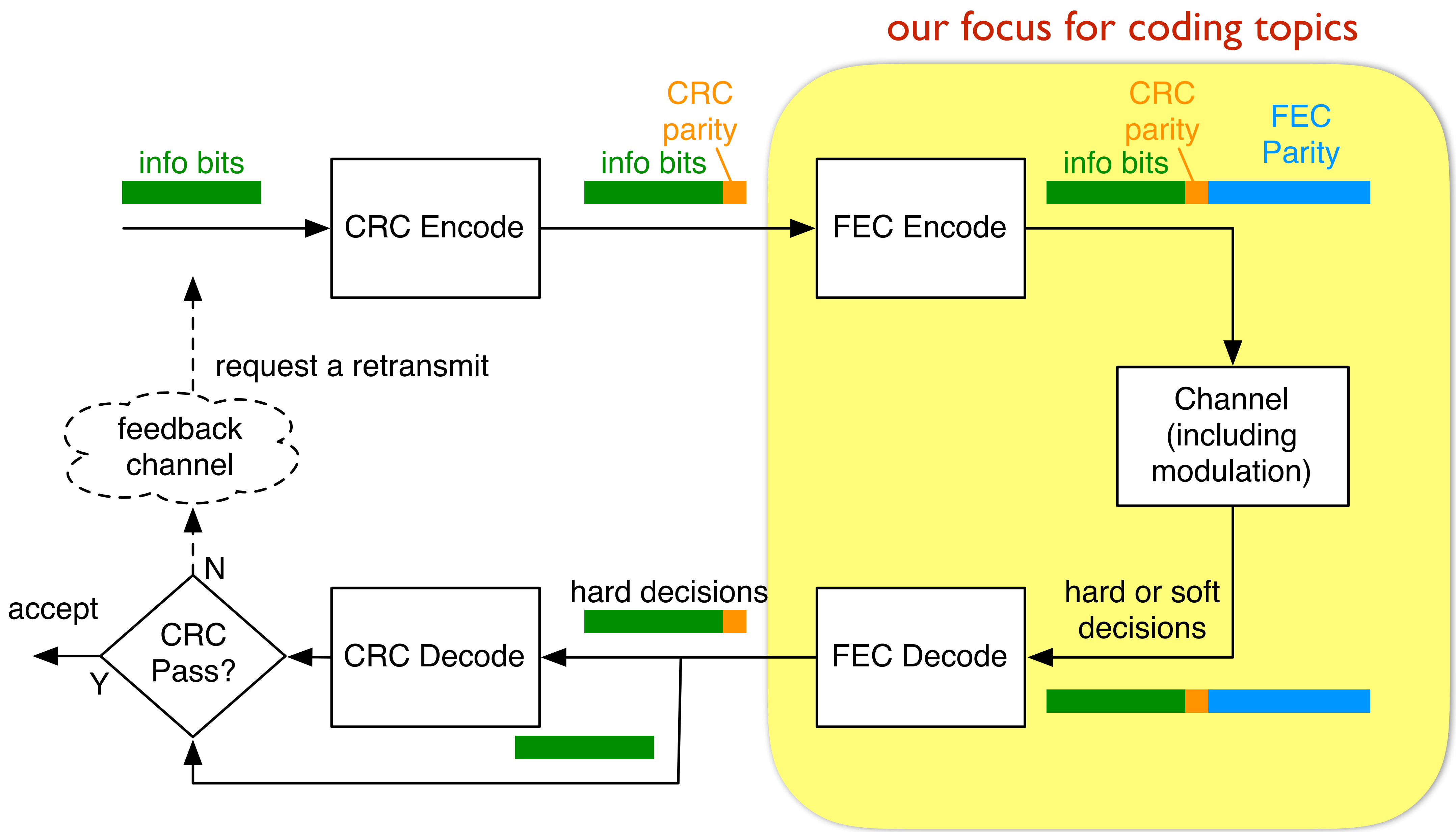
Course Topic (from Syllabus)

- Overview of Comm/Coding
- Signal representation and Random Processes
- Optimal demodulation and decoding
- Uncoded modulations, demod, performance
- Classical FEC
- Modern FEC
- Non-AWGN channels (intersymbol interference)
- Practical consideration (PAPR, synchronization, spectral masks, etc.)

Coding Topics

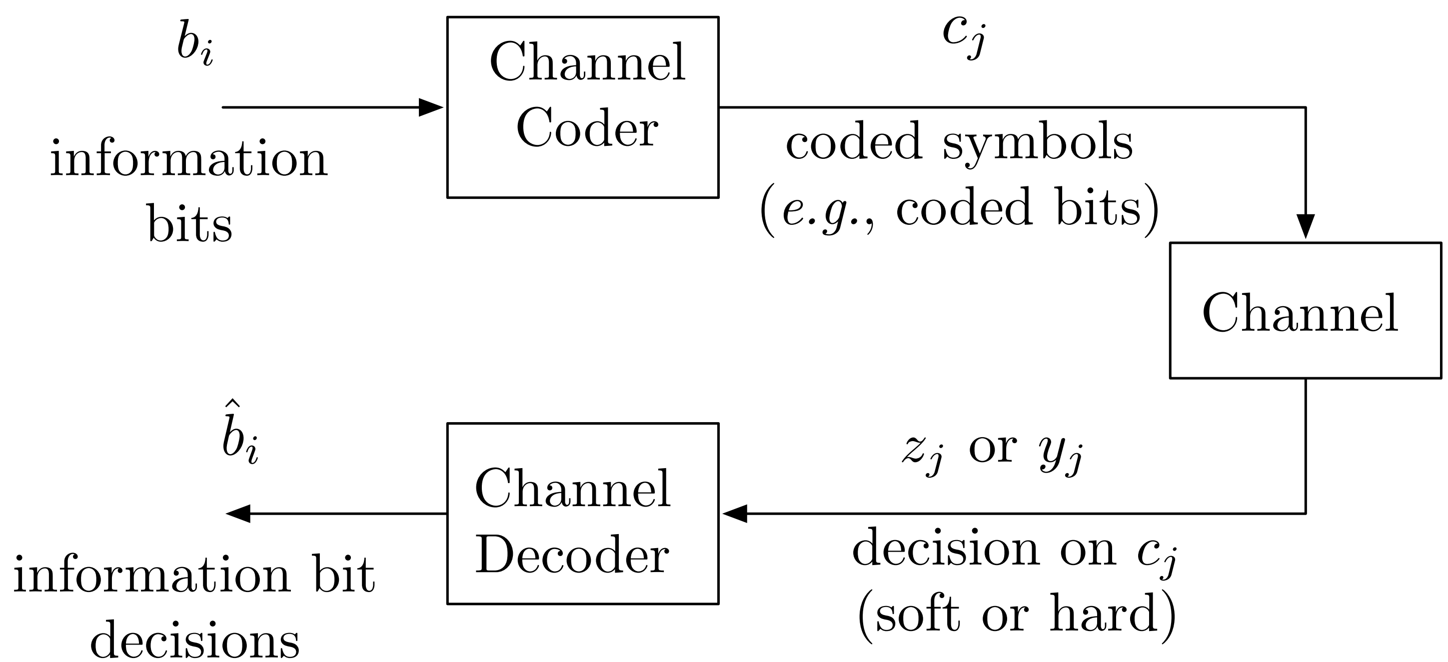
- Coding channel models
- Basics of code constructions
- Decoding rules — HIHO, SIHO, SISO
- Classical coding
- Modern Coding
- Performance limits
    - Capacity and finite block-size bounds)
    - Bounds for specific codes

Typical Use of Coding in Modern System



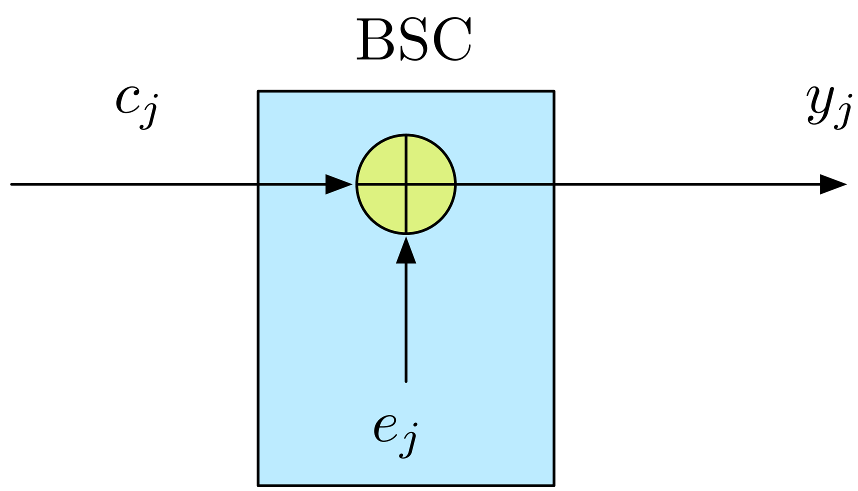
Hybrid ARQ (H-ARQ) System

Coding Channel Models



- Typically the coding channel is an abstraction of a more detailed model
  - e.g., it may encapsulate modulation/demod/demapping

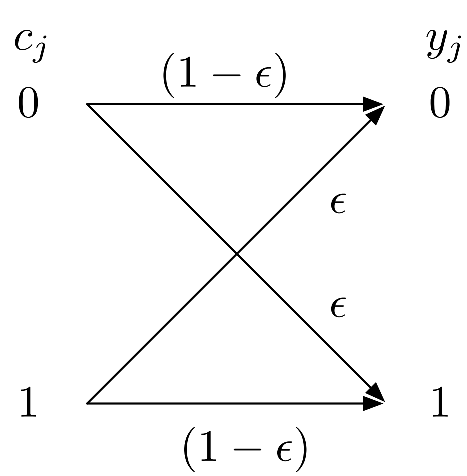
Binary Symmetric Channel



$$e_j(u) \sim \text{iid Bernoulli}(\epsilon)$$

all math is modulo 2

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

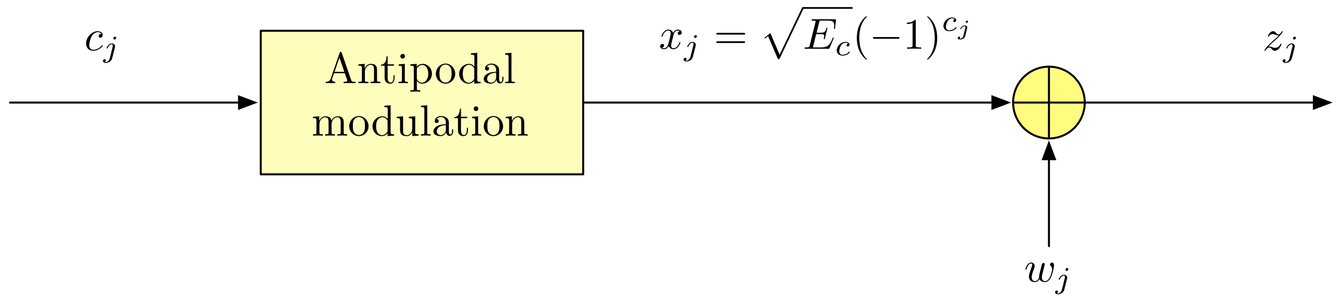


labels: $p_{y_j(u)|c_j(u)}(y_j|c_j)$

BSC is a special case the discrete memoryless channel (DMC) (non-binary)

DMCs are fully characterized by this type of transition diagram

BPSK-AWGN or BI-AWGN Channel



$$w_j(u) \sim \mathcal{N}(\cdot; 0; N_0/2)$$

$$f_{z_j(u)|c_j(u)}(z|c) = \mathcal{N}\left(z; \sqrt{E_c}(-1)^c; N_0/2\right)$$

BI-AWGN Channel is a special case of the modulation-constrained AWGN channel

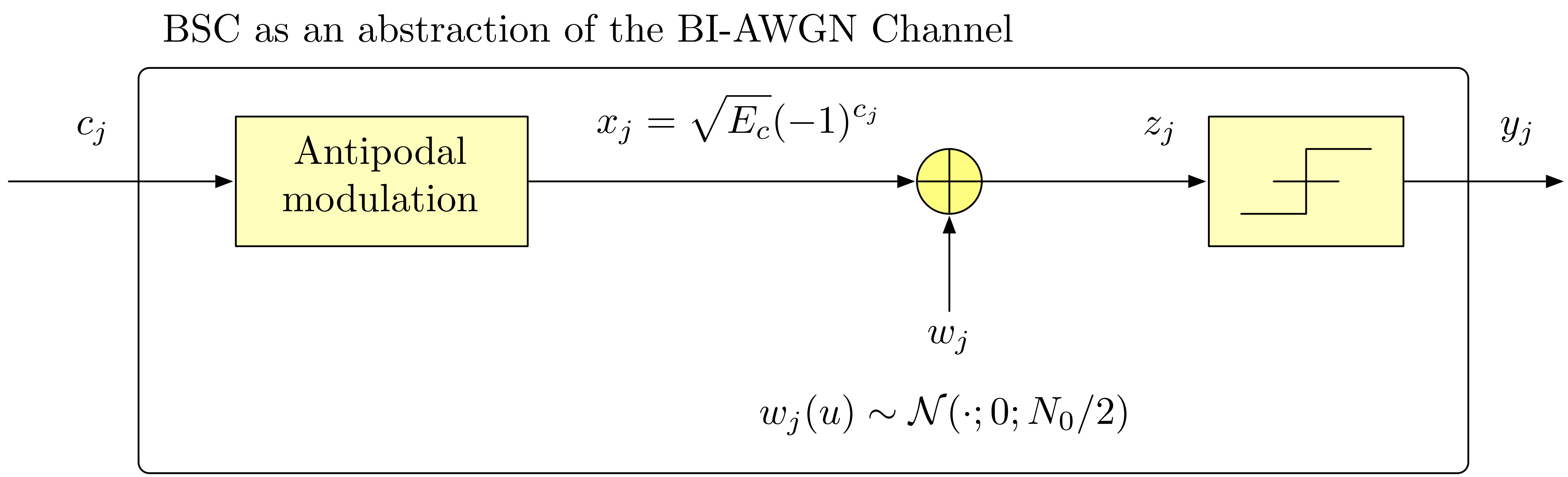
e.g., the 16-PSK constrained AWGN channel

$$\mathbf{z}_k(u) = \mathbf{x}(u) + \mathbf{w}(u)$$

$$\mathbf{w}(u) \sim \mathcal{N}_2\left(\cdot; \mathbf{0}; \frac{N_0}{2}\mathbf{I}\right)$$

$$\mathbf{x}(u) \in 16 \text{ PSK constellation}$$

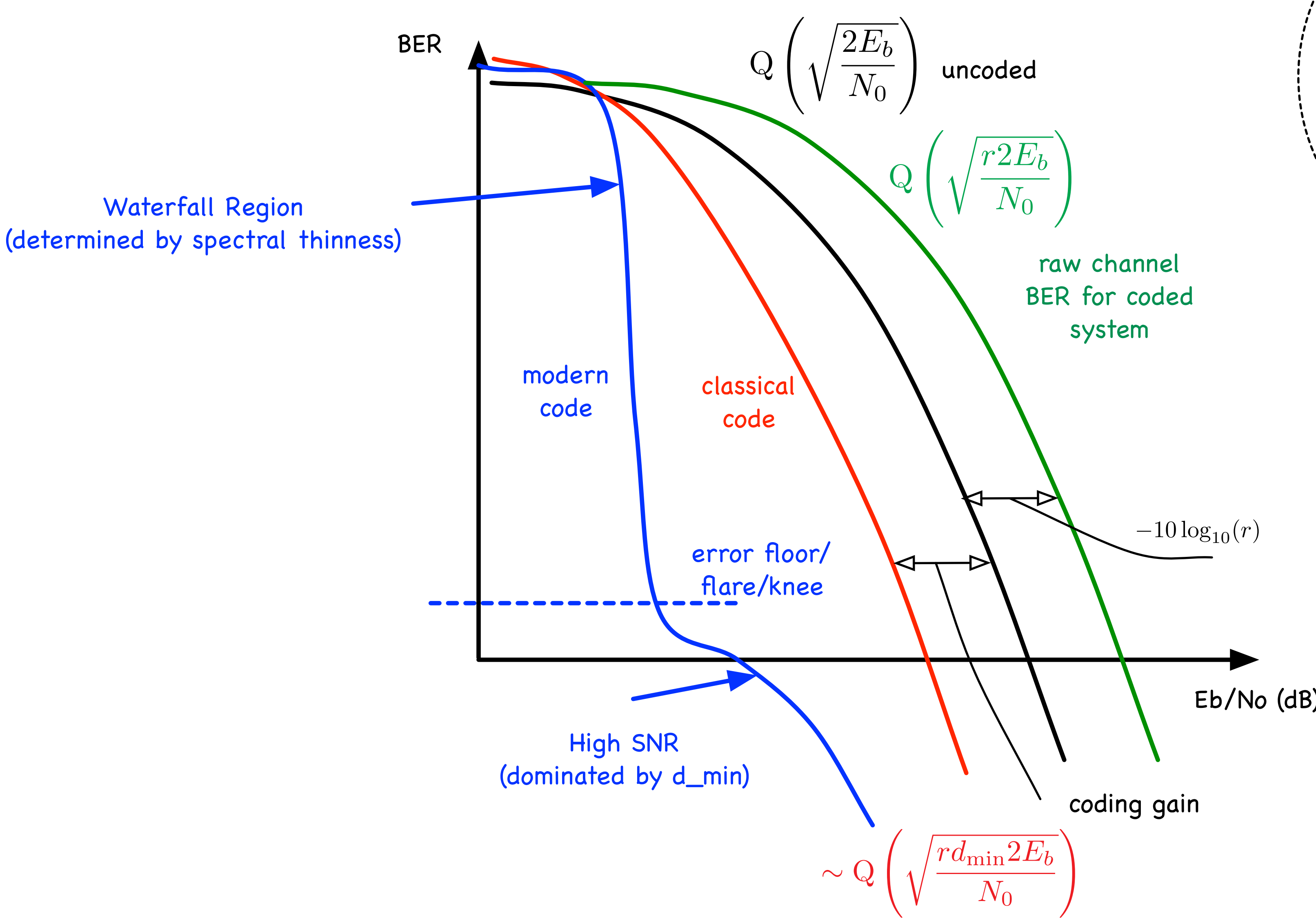
BSC as Abstraction of BI-AWGN Channel

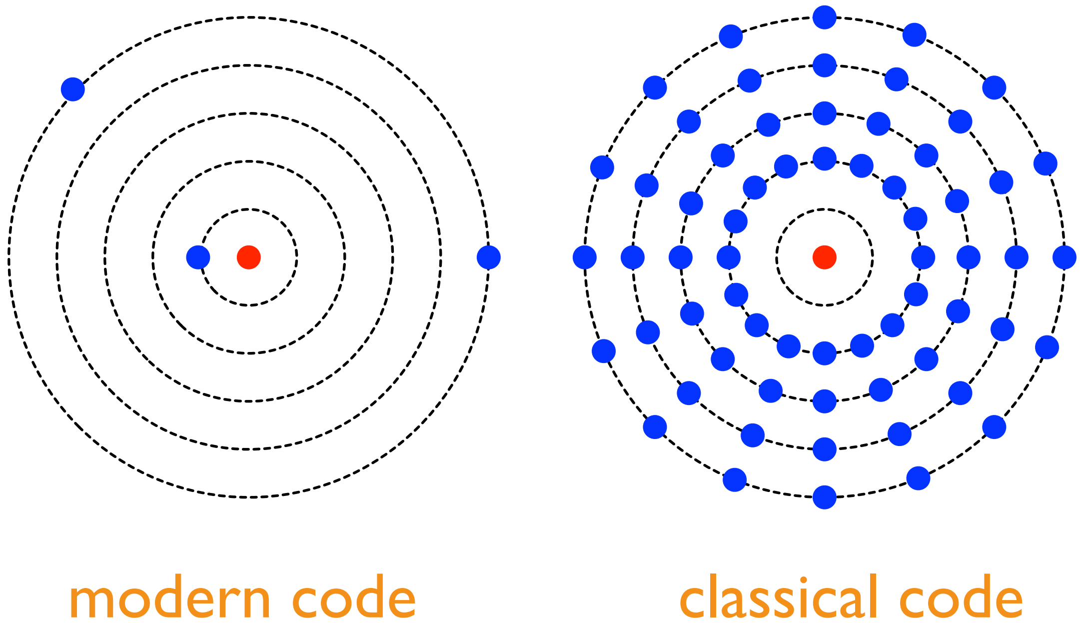


$$\epsilon = \mathrm{Q}\left(\sqrt{\frac{2E_c}{N_0}}\right) = \mathrm{Q}\left(\sqrt{\frac{r2E_b}{N_0}}\right)$$

raw channel error probability

Typical Performance on BI-AWGN





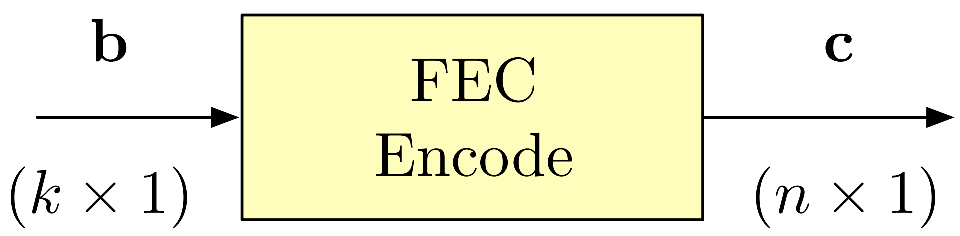
Coding Topics

- Coding channel models
- Basics of code constructions
- Decoding rules — HIHO, SIHO, SISO
- Classical coding
- Modern Coding
- Performance limits
    - Capacity and finite block-size bounds)
    - Bounds for specific codes

Code Constructions

- We are focused on **linear binary** codes
  - binary inputs, binary outputs
  - linear: sum of two codewords is also a codeword
- Linear (binary) block codes
- Linear (binary) convolutional codes
- Modern codes
  - Low Density Parity Check (LDPC) Codes
  - Concatenated convolutional codes - e.g., Turbo codes

Linear Block Codes



code rate: $r = k/n$

$$\mathbf{c}^{\mathrm{t}} = \mathbf{b}^{\mathrm{t}}\mathbf{G}$$

$\mathbf{G}$ $(k \times n)$ Generator Matrix

$$\mathbf{c} = \mathbf{G}^{\mathrm{t}}\mathbf{b}$$

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{k-1} \end{bmatrix} \qquad (k \times 1), \qquad b_i \in \mathcal{Z}_2 = \{0, 1\}$$

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} \qquad (n \times 1), \qquad c_j \in \mathcal{Z}_2 = \{0, 1\}$$

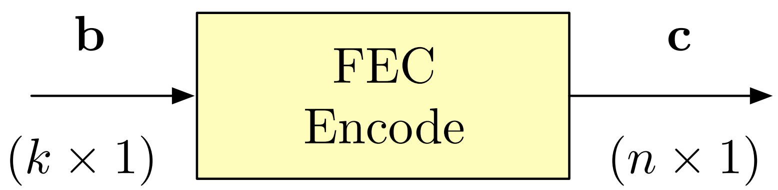
all math is modulo 2

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Coding Conventions/Notation

- (n,k) code - n, k almost universal notation
  - n = (output) block size
  - k = input/info block size
- row vectors are often used
  - (I use column vectors)
- Mod-2 arithmetic is not explicitly denoted
  - just a+b and (a+b)%2 is implied

Linear Block Codes - Generator Matrix



code rate: $r = k/n$

$$\mathbf{c}^{\mathrm{t}} = \mathbf{b}^{\mathrm{t}}\mathbf{G}$$

$$\mathbf{c} = \mathbf{G}^{\mathrm{t}}\mathbf{b}$$

$\mathbf{G}$ $(k \times n)$ Generator Matrix

$$\mathbf{G}^{\mathrm{t}} = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_{k-1} \end{bmatrix}$$

Only interested in full-rank **G** - no repeated codewords

$$\mathbf{c} = \sum_{i=0}^{k-1} b_i \mathbf{g}_i$$

columns of G-transpose are a basis and the info bits are the coefficients of codeword expansion in this basis

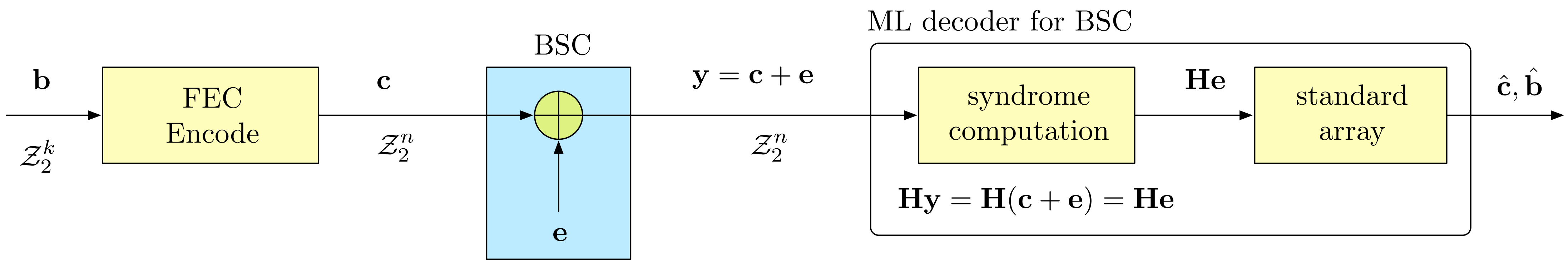
A linear block code is a linear subspace of the space of all (n x1) binary vectors

$$\mathcal{C} = \left\{ \mathbf{c} : \mathbf{c} = \mathbf{G}^{\mathrm{t}}\mathbf{b}, \mathbf{b} \in \mathcal{Z}_2^k \right\} \subset \mathcal{Z}_2^n$$

$$\dim(\mathcal{C}) = k$$

$$M = 2^k = \text{number of codewords}$$

Linear Block Codes - Parity Check Matrix



The *parity check matrix* **H** also characterizes the code

$$\mathbf{Hc} = \mathbf{0} \quad \Longleftrightarrow \quad \mathbf{c} \in \mathcal{C}$$

$$\mathbf{H} \text{ is } ((n-k) \times n)$$

$$\text{rank}(\mathbf{H}) = n - k$$

$$\mathbf{H}\mathbf{G}^{\mathrm{t}} = \mathbf{O}$$

the code as a constraint

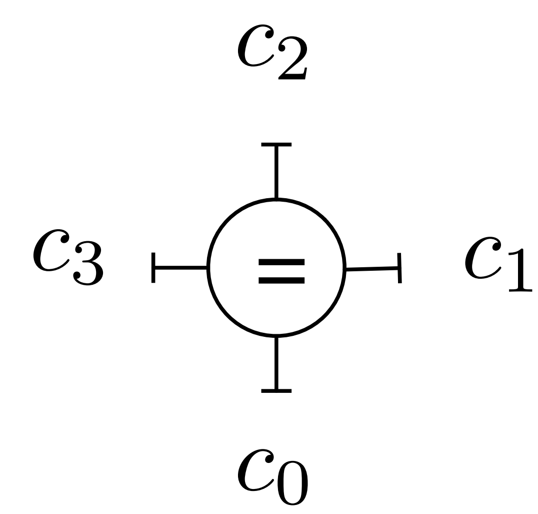
$$\mathcal{C} = \{\mathbf{c} : \mathbf{Hc} = \mathbf{0}\} \subset \mathcal{Z}_2^n$$

$$\dim(\mathcal{C}) = k$$

$$M = 2^k = \text{number of codewords}$$

Example: Repetition Code

Codewords for n = 4: 0000 1111



Number of codewords =2 , so k = 1

rate = 1/n (info bits per channel use)

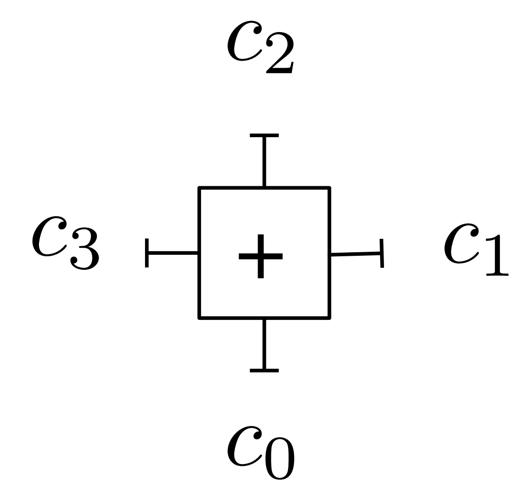
$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

In general, this is an (n, 1) code

Example: Single Parity Check Code

Codewords for n = 4: 0000 0101
0011 1001
1100 0110
1010 1111



Number of codewords = 8 , so k = 3 = n-1

rate = (n-1)/n (info bits per channel use)

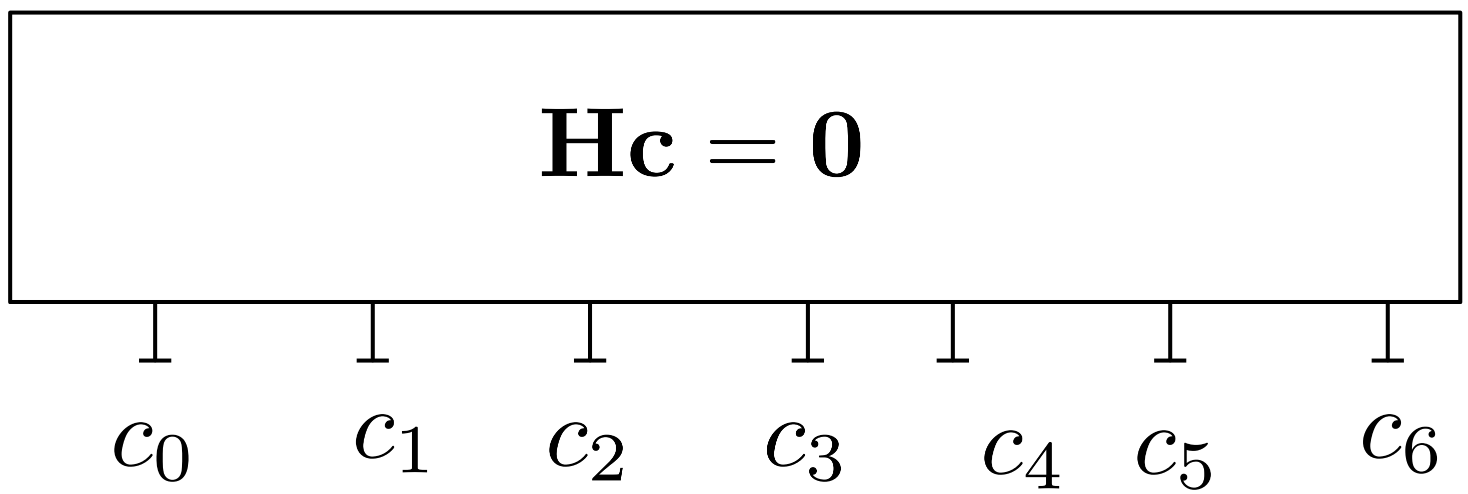
$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

In general, this is an (n, n-1) code

Example: (7,4) Hamming Code

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$



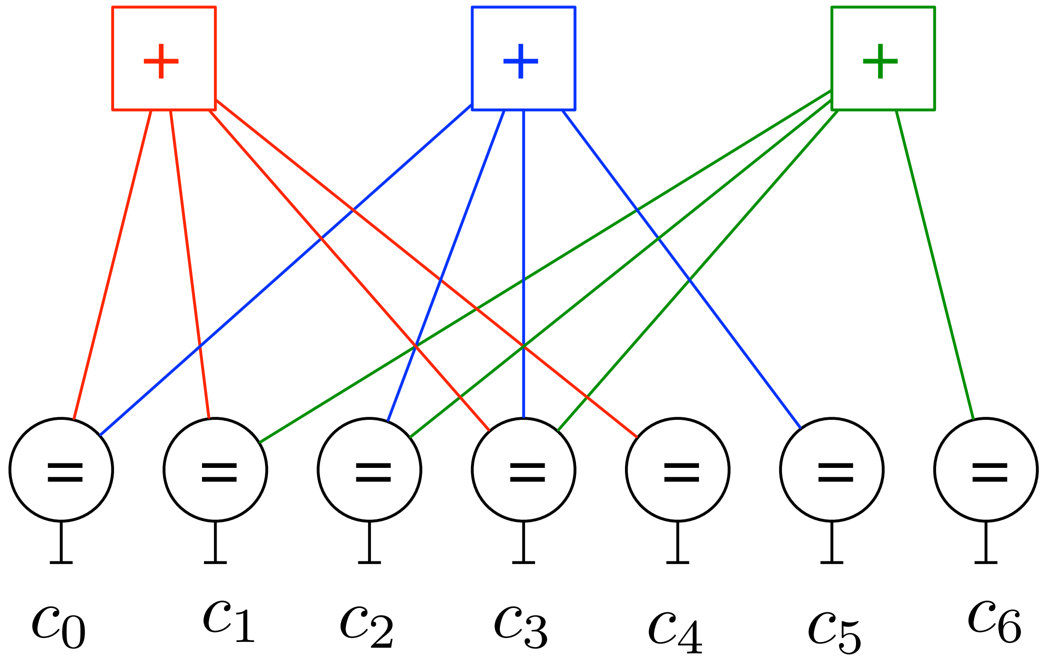
Linear Block Code ("Multiple Parity Check Code")

All three SPCs must be satisfied simultaneously

Example: (7,4) Hamming Code

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Parity Check Graph or Tanner Graph



All local constraints must be satisfied simultaneously

Example: Low Density Parity Check (LDPC) Code

Just a very large (multiple) parity check code with mostly 0s

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ 1 & 1 & \dots & 0 & 0 & 0 \end{bmatrix}$$

number of 1s = number of bits in first SPC

number of 1s = number of SPCs second code bit is involved in

A systematic way to build codes with very large block size

Example: (7,4) Hamming Code

$$\mathbf{H} = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

These **H** and **G** examples are in a specific format

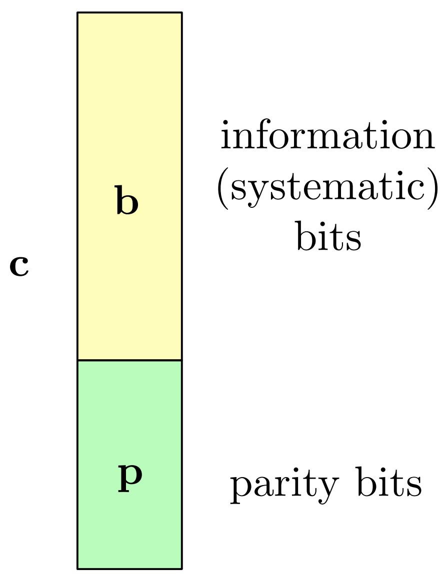
Relation Between Generator/Parity Check

$$\mathbf{Hc} = \mathbf{HG}^{\mathrm{t}}\mathbf{b} = \mathbf{0} \quad \forall\ \mathbf{b} \in \mathcal{Z}_2^k$$

$$\boxed{\mathbf{HG}^{\mathrm{t}} = \mathbf{O}}$$

All **H** and **G** for a given code must satisfy this

Systematic Code/Form



A systematic code is one is which the information bits appear explicitly in k of the coordinates of the codewords (typically the first k)

$$\mathbf{G} = \left[\begin{array}{c|c} \mathbf{I}_k & \mathbf{P} \end{array} \right]$$

$$\mathbf{H} = \left[\begin{array}{c|c} \mathbf{P}^{\mathrm{t}} & \mathbf{I}_{n-k} \end{array} \right]$$

systematic form for **G** and **H**

$$\mathbf{G}^{\mathrm{t}}\mathbf{b} = \begin{bmatrix} \mathbf{I}_k \\ \mathbf{P}^{\mathrm{t}} \end{bmatrix} \mathbf{b} = \begin{bmatrix} \mathbf{b} \\ \mathbf{P}^{\mathrm{t}}\mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{p} \end{bmatrix}$$

$$\mathbf{H}\mathbf{c} = \mathbf{H} \begin{bmatrix} \mathbf{b} \\ \mathbf{p} \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{P}^{\mathrm{t}} & \mathbf{I}_{n-k} \end{array} \right] \begin{bmatrix} \mathbf{b} \\ \mathbf{p} \end{bmatrix} = \left[\mathbf{p} + \mathbf{p} \right] = \mathbf{0}$$

Code vs. Encoder

$$\mathcal{C} = \left\{ \mathbf{c} : \mathbf{c} = \mathbf{G}^{\mathrm{t}}\mathbf{b}, \mathbf{b} \in \mathcal{Z}_2^k \right\} = \{ \mathbf{c} : \mathbf{H}\mathbf{c} = \mathbf{0} \}$$

A code is the linear space — think of this as the signal set

There are many generators for the same code

e.g., can do row operations on G without affecting row-space which is the code

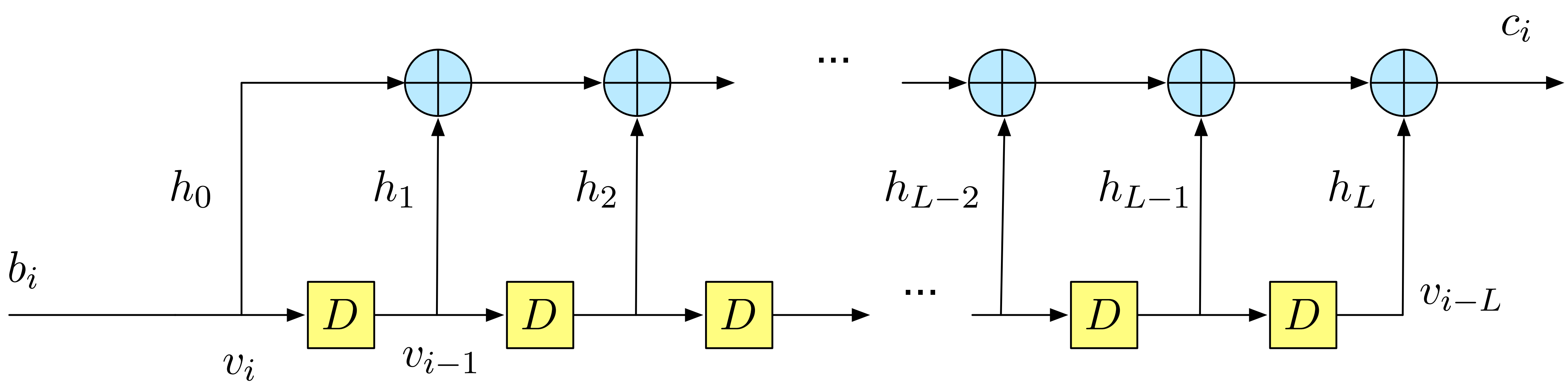
An encoder is the mapping from **b** to **c** — i.e., the generator matrix **G**

think of this as the bit-labeling of the signal set

If we do MAP codeword decoding, changing encoders will not affect the probability of codeword error, but may affect the probability of bit error

Linear Binary Convolutional Codes

non-recursive or feedforward convolutional encoder



$$v_i = b_i$$

$$c_i = h_0 v_i + h_1 v_{i-1} + h_2 v_{i-2} + \cdots h_L v_{i-L}$$

generator polynomial: $$G(D) = h_0 + h_1 D + h_2 D^2 \ldots + h_L D^L$$

state: $$s_i = (v_{i-1}, v_{i-2}, \ldots v_{i-L})$$

Models for Convolutional Codes

L = memory of the convolution code

K = (L+1) constraint length of the convolution code

$$\text{Number of states} = 2^L$$

Finite State Machine (FSM) Model

$$s_{i+1} = \text{next\_state}(b_i, s_i)$$

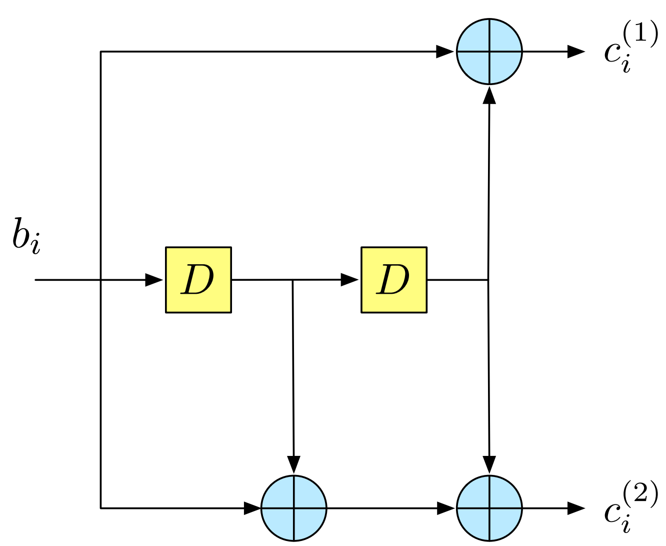
$$\mathbf{c}_i = \text{output}(b_i, s_i)$$

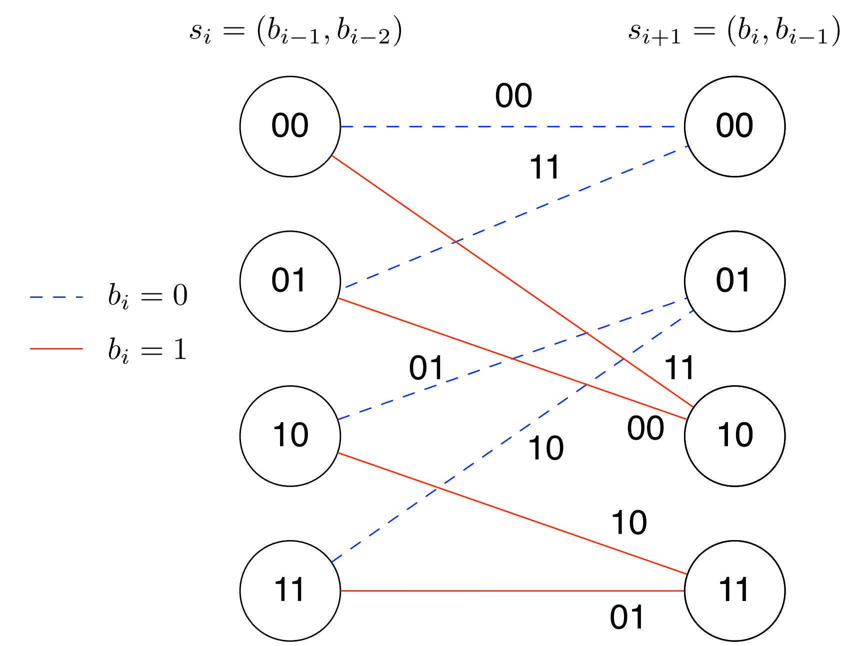
FSM model of Convolution Code (encoder) is given by any of the following:

- State transition table (above rules)
- State transition diagram
- Trellis diagram

Feedforward CC Example

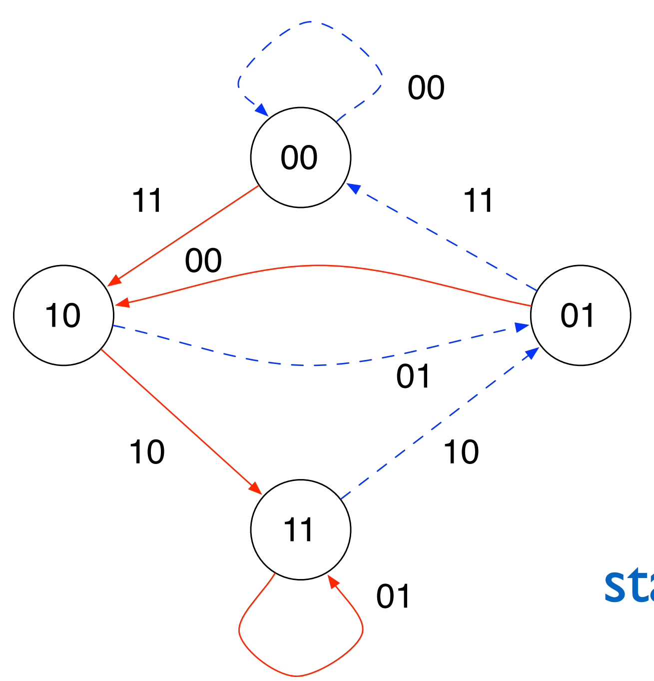
(encoder) block diagram





trellis diagram (one stage)

G1 = 5 = (101)
G2 = 7 = (111)

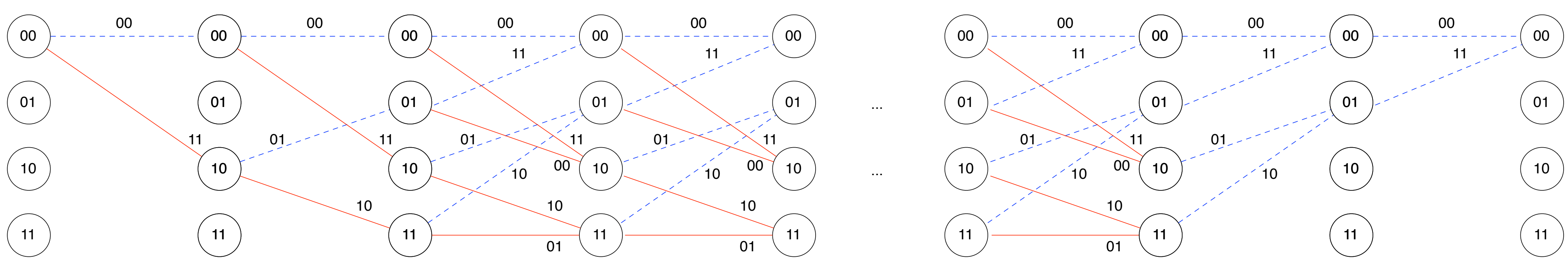


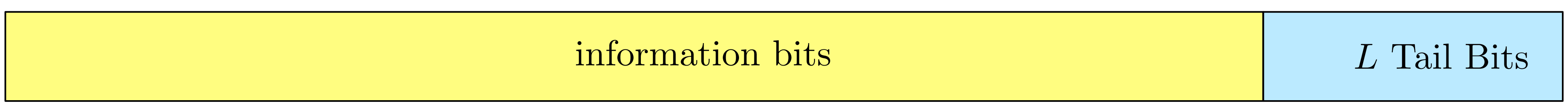
state transition diagram

Models for Convolutional Codes

trellis (typical usage)

all valid configurations of the code

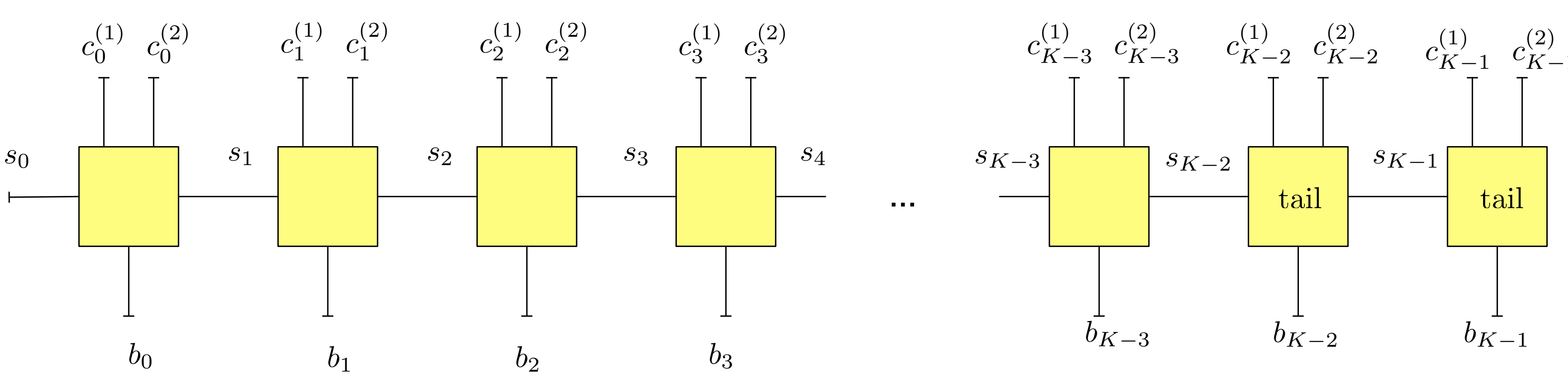




known initial state $s_0 = (00)$

Tail bits drive to zero final state

Graphical Model (Normal Graph)



Linear Binary Convolutional Codes

Non-recursive CCs are common in classical coding

GSM Cellular:	16 state, r=1/2	G1 = 23 = (10011) G2 = 35 = (11101)	d_free = 6
“Oldenwalder Code”:	64 state, r=1/2	G1 = 133 G2 = 171	d_free = 10

NASA’s Voyager mission, many satellite modems, Wi-Fi

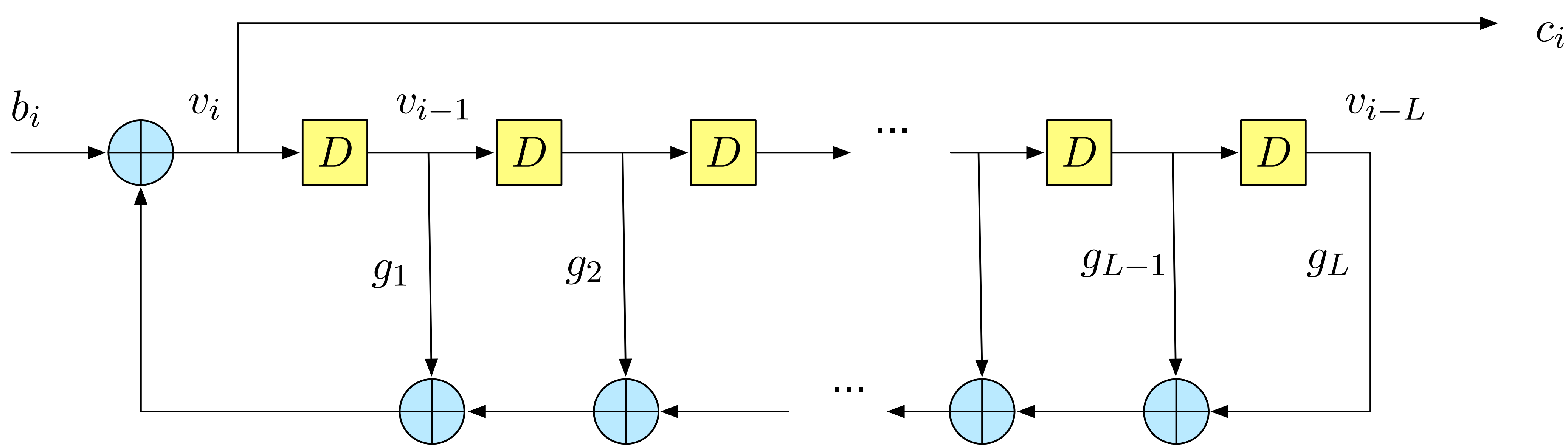
CDMA Cellular (IS-95):	256 state, r=1/2	G1 = 752 G2 = 561	d_free = 12

As L increase: decoder complexity increases, performance improves

see Benedetto, page 549 for list of best CCs

Linear Binary Convolutional Codes

recursive or feedback convolutional encoder



$$c_i = v_i = b_i + g_1 v_{i-1} + g_2 v_{i-2} + \cdots g_L v_{i-L}$$

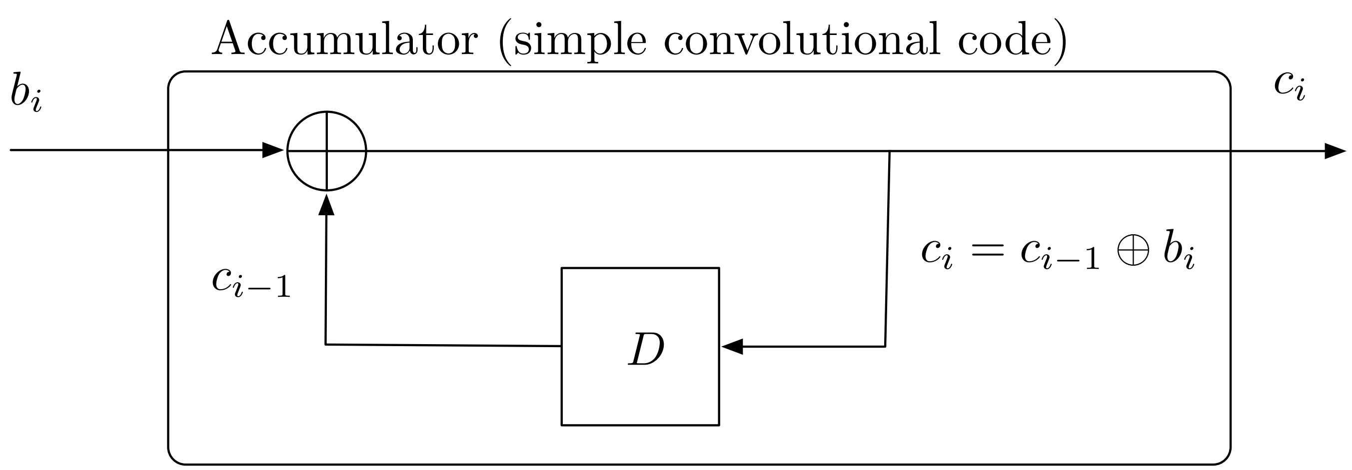
$$b_i = v_i + g_1 v_{i-1} + g_2 v_{i-2} + \cdots g_L v_{i-L}$$

generator polynomial:
$$G(D) = \frac{1}{1 + g_1 D + g_2 D^2 + \cdots g_L D^L}$$

state:
$$s_i = (v_{i-1}, v_{i-2}, \ldots v_{i-L})$$

Linear Binary Convolutional Codes

recursive or feedforward convolutional encoder

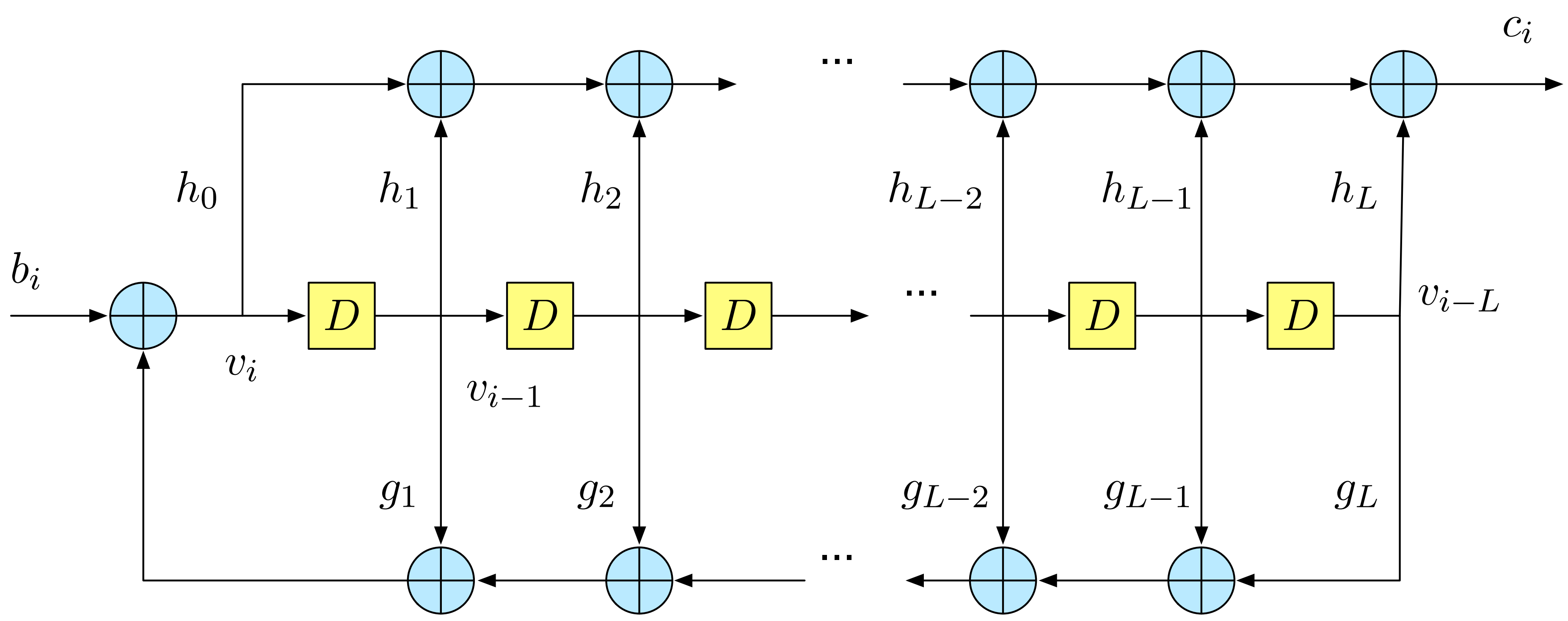


$$G(D) = \frac{1}{1+D}$$

example: accumulator (recall binary differential encoder)

Linear Binary Convolutional Codes

feedforward/feedback encoder (general case) - recursive if denominator != 1



$$v_i = b_i + g_1 v_{i-1} + g_2 v_{i-2} + \cdots g_L v_{i-L}$$

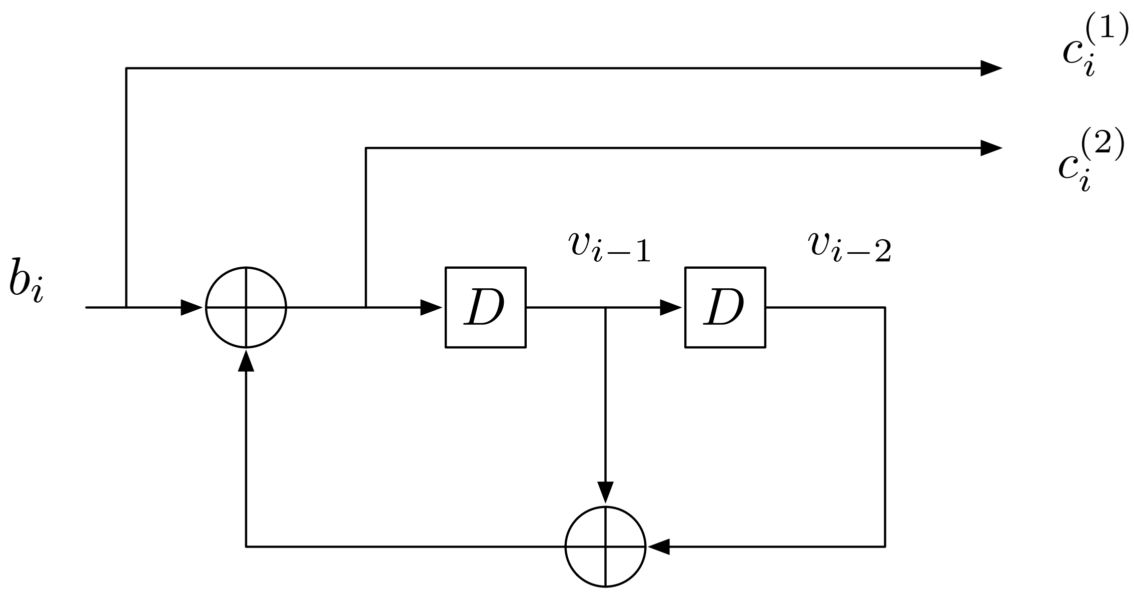
$$b_i = v_i + g_1 v_{i-1} + g_2 v_{i-2} + \cdots g_L v_{i-L}$$

$$c_i = h_0 v_i + h_1 v_{i-1} + h_2 v_{i-2} + \cdots h_L v_{i-L}$$

generator polynomial: $$G(D) = \frac{h_0 + h_1 D + \ldots h_L D^L}{1 + g_1 D + g_2 D^2 + \cdots g_L D^L}$$

state: $$s_i = (v_{i-1}, v_{i-2}, \ldots v_{i-L})$$

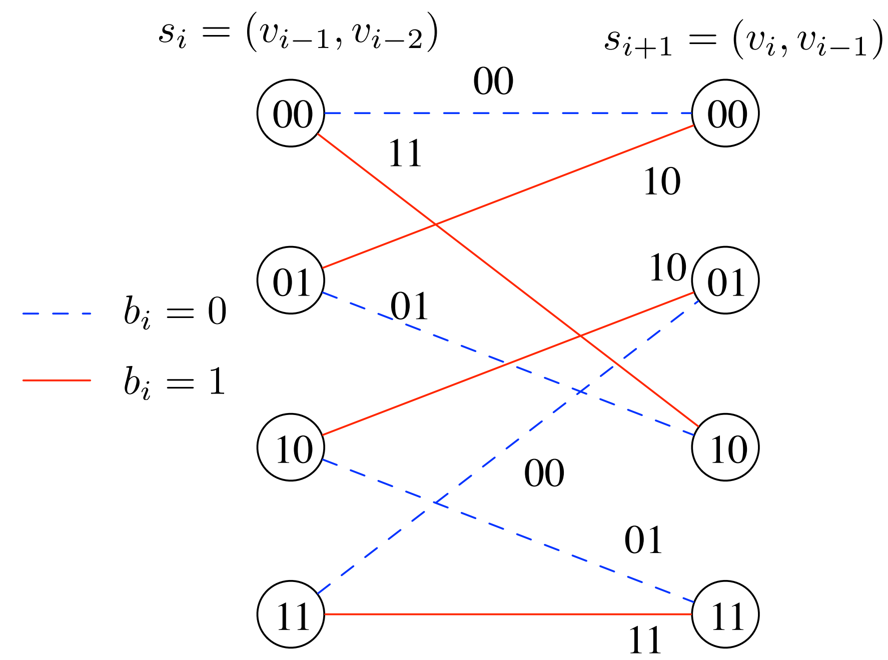
Models for Convolutional Codes



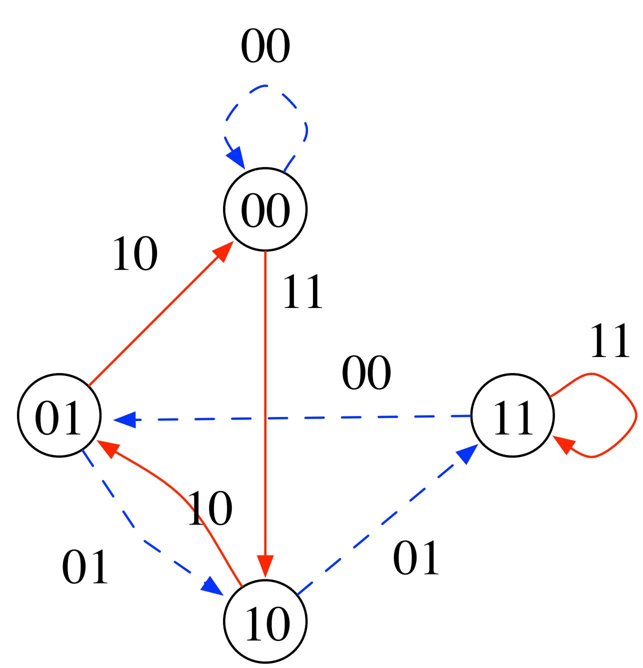
$$G_1(D) = 1$$

$$G_2(D) = \frac{1}{1 + D + D^2}$$

(encoder) block diagram



trellis diagram (one stage)

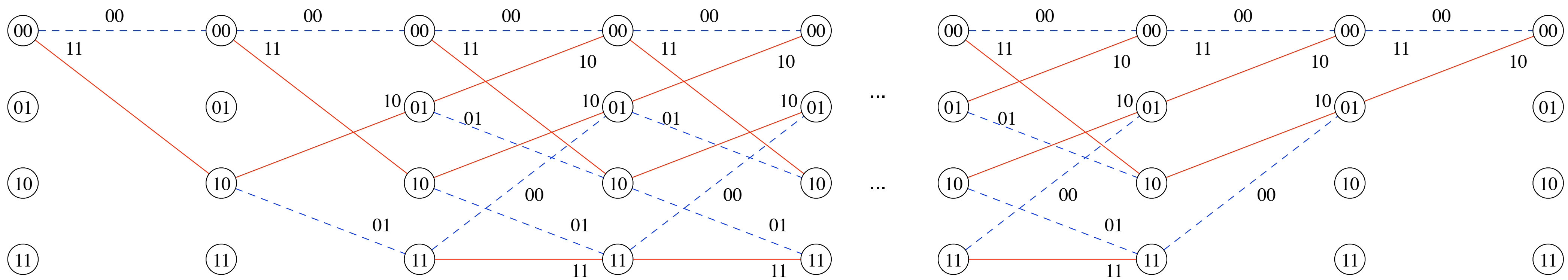


state transition diagram

Models for Convolutional Codes

trellis (typical usage)

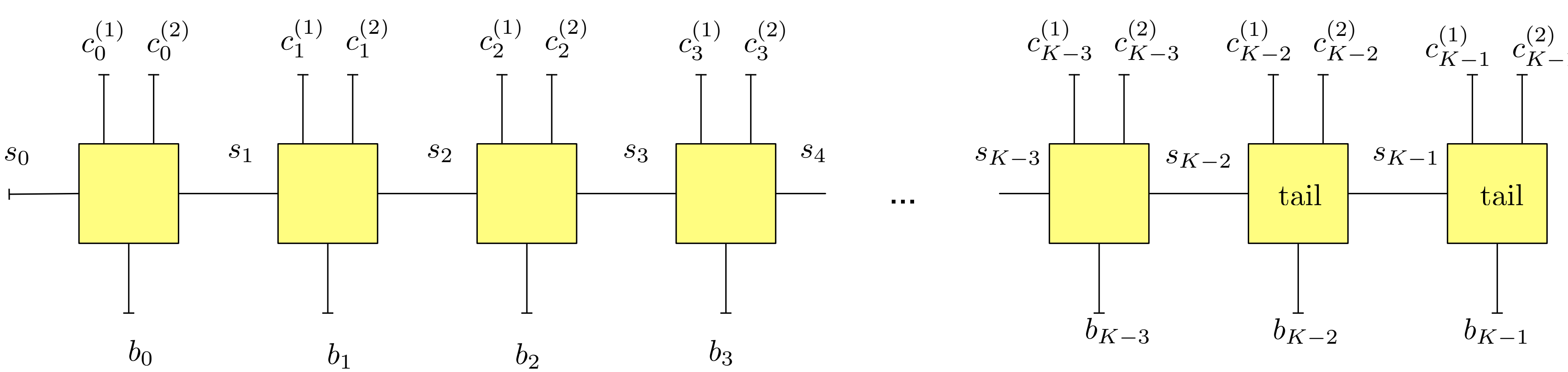
all valid configurations of the code



known initial state $s_0 = (00)$

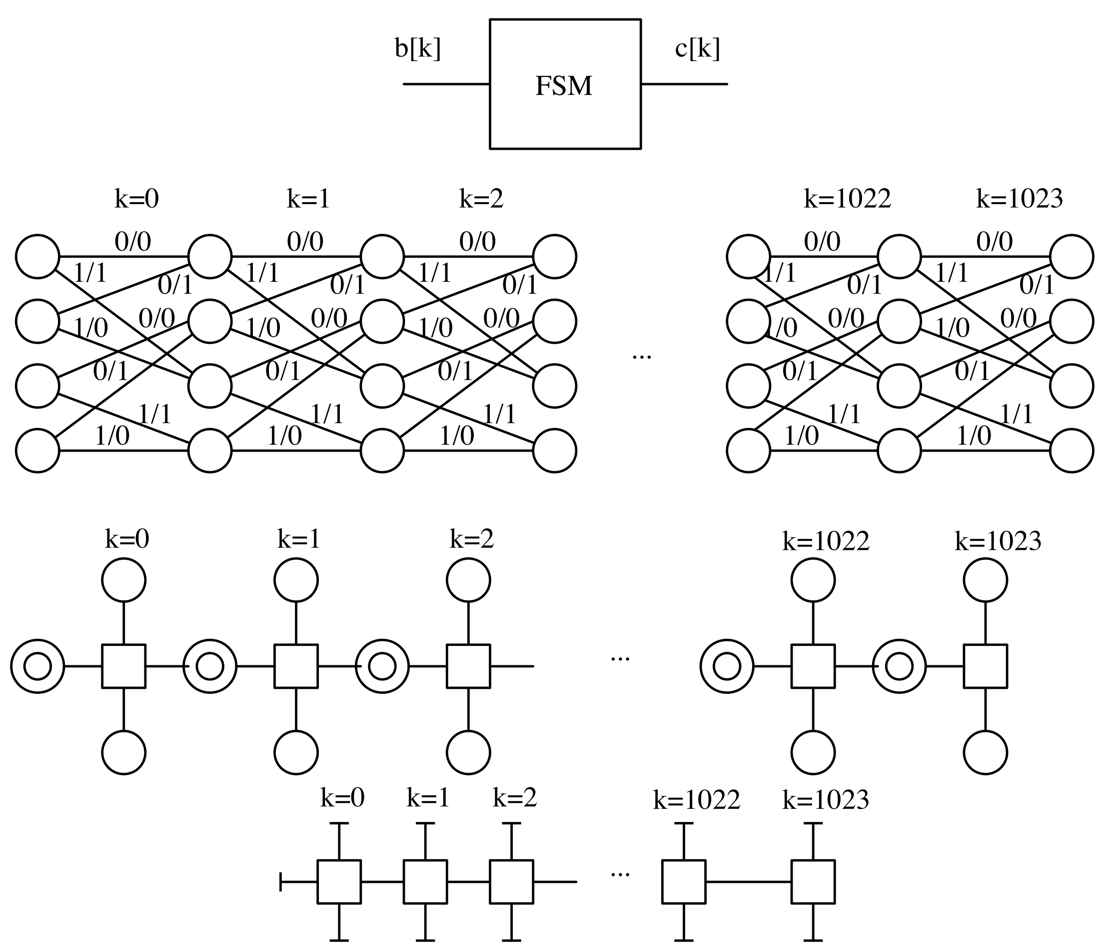
Tail bits drive to zero final state

Graphical Model (Normal Graph)

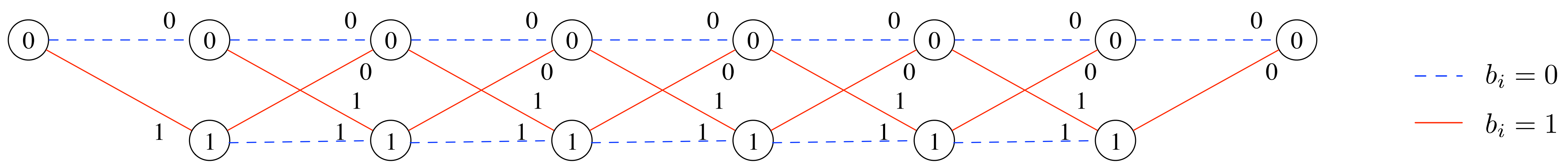


Models for Convolutional Codes

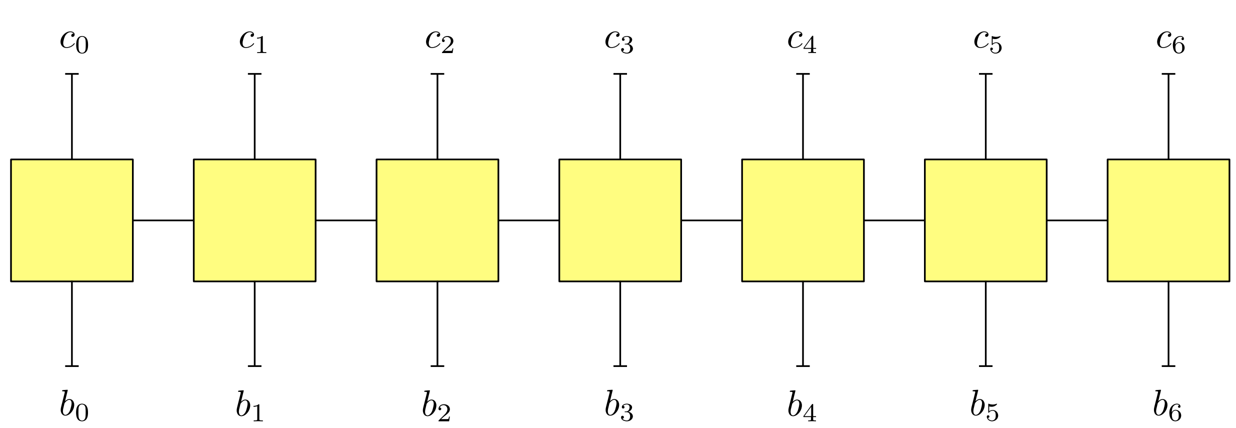
Model	Time (index)	Values
Block Diagram	implicit	implicit
Trellis	explicit	explicit
Graph	explicit	implicit



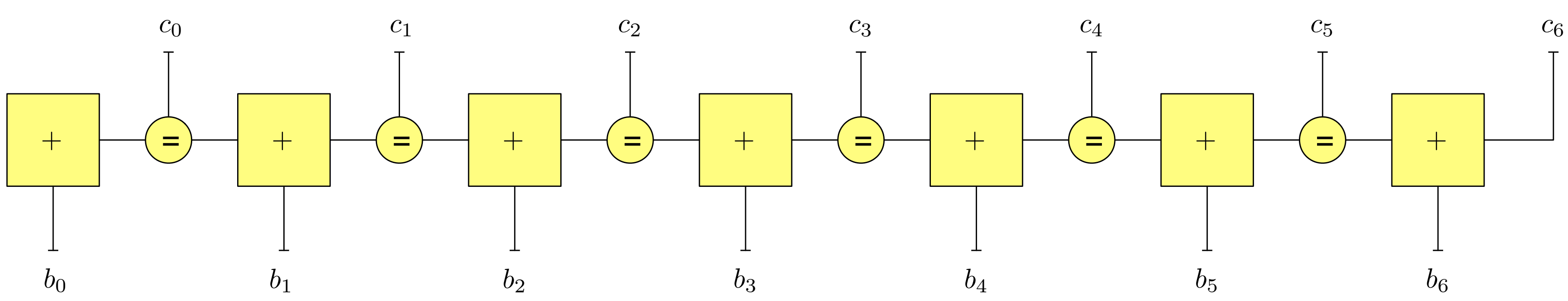
Accumulator Trellis & Graphical Model



$$c_i = c_{i-1} + b_i = s_i + b_i$$



trellis section are local codes



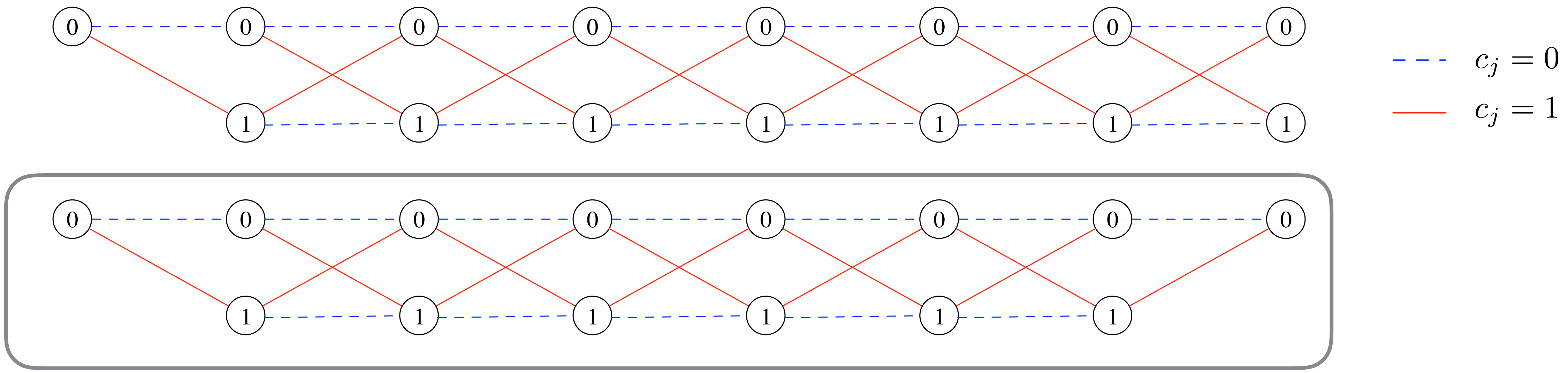
state is previous output

Parity Check Trellis For Linear Block Codes

(n,n-1) SPC

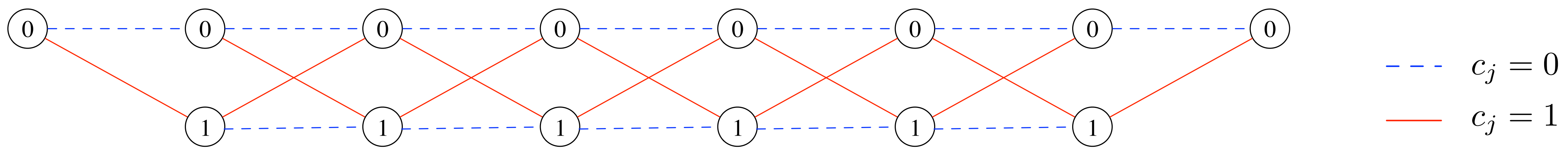
$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

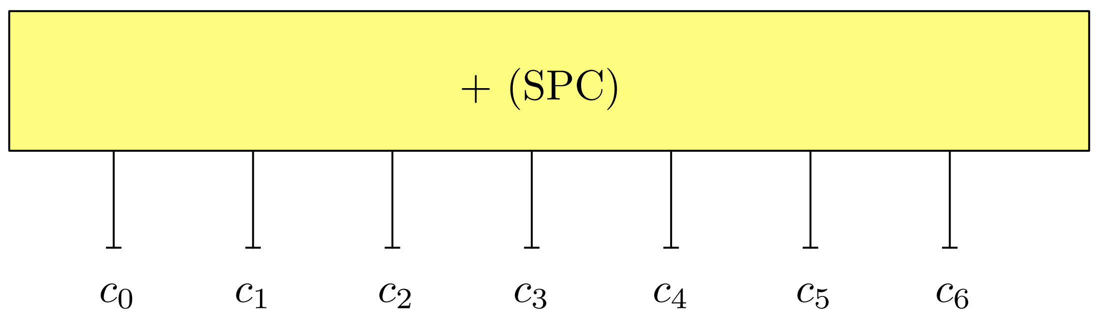
$$s_j = \sum_{m=0}^{j-1} c_j = s_{j-1} + c_j$$

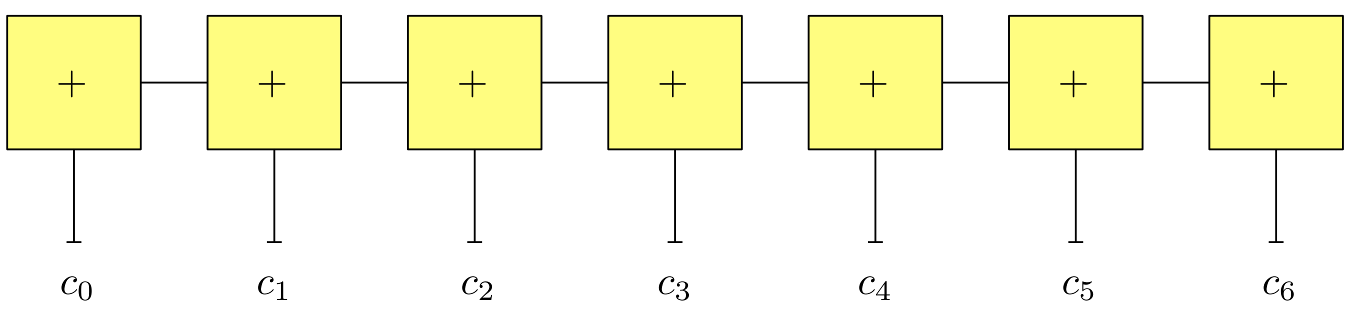


all valid codewords are paths in this trellis (total parity 0)

Parity Check Trellis For Linear Block Codes

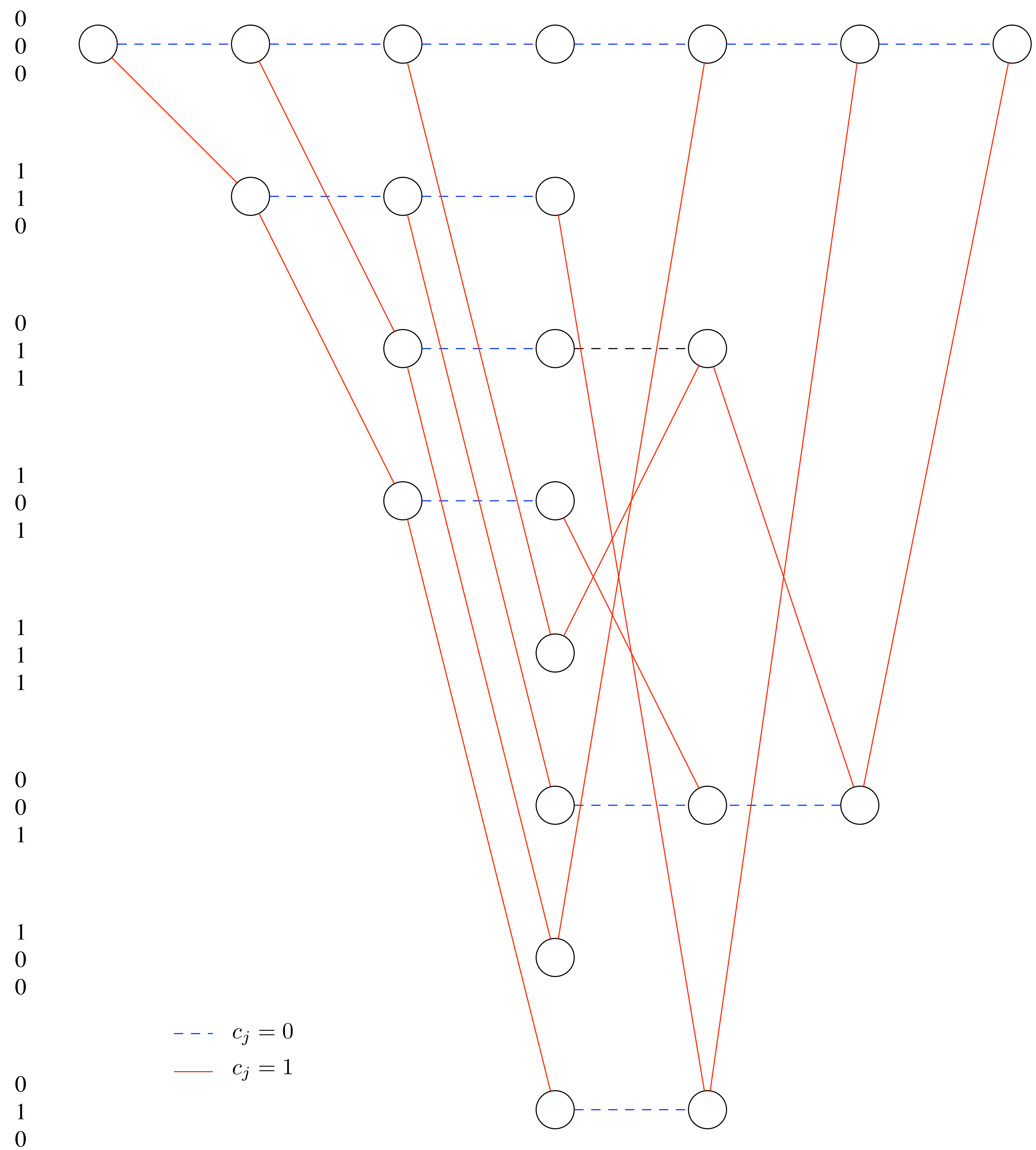






notice that this is very similar to the accumulator trellis (w/ no "output")

Parity Check Trellis For Linear Block Codes



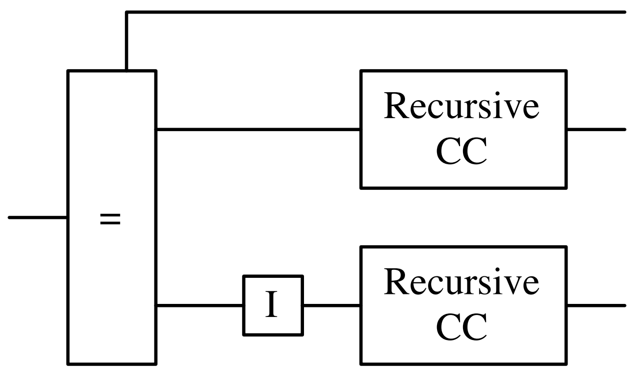
(6,3) code

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

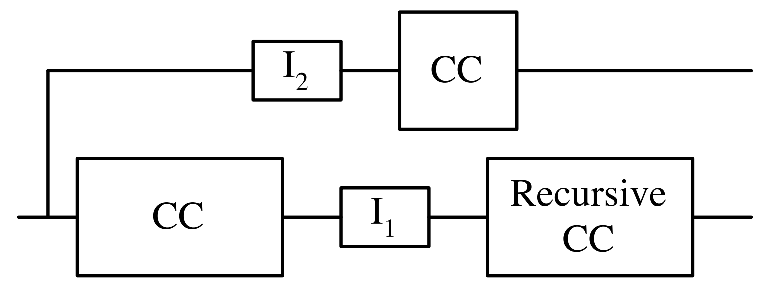
See the Parity Check Trellis handout

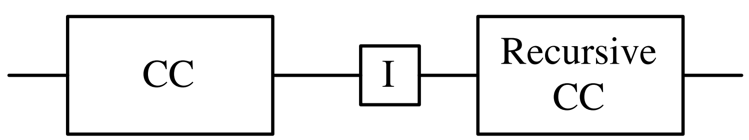
Modern Codes

Parallel Concatenated Convolutional Codes (PCCCs) or Turbo Codes

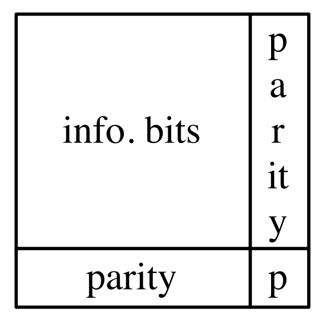


Hybrid Concatenated Convolutional Codes

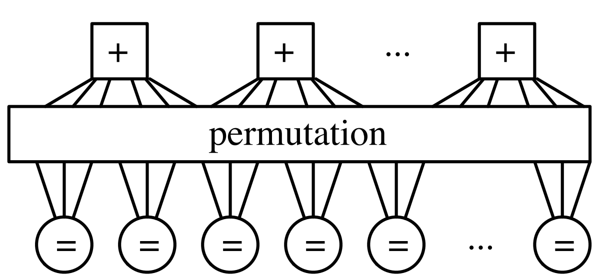




Serially Concatenated Convolutional Codes (SCCCs)



Product Codes



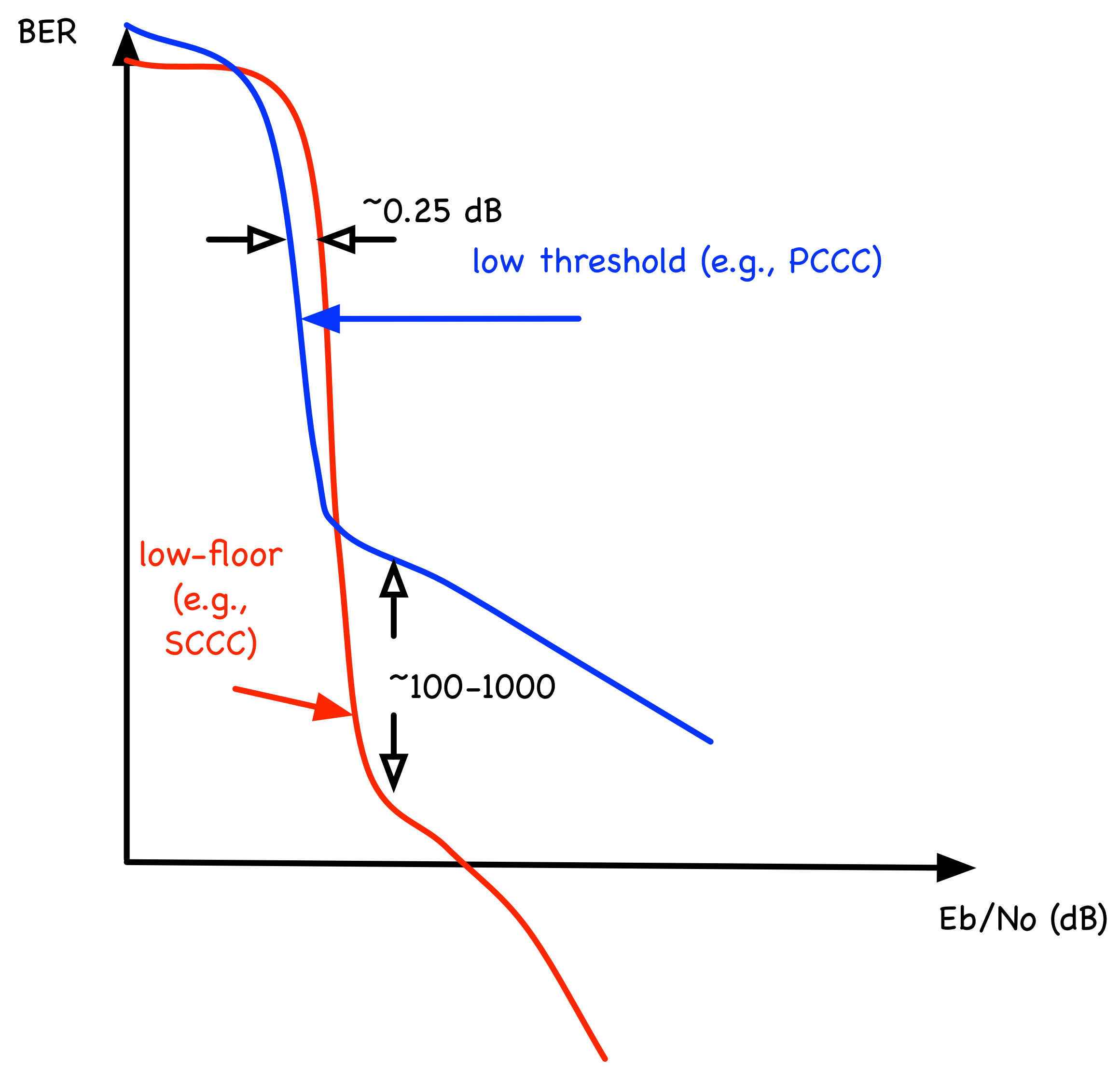
Low Density Parity Check (LDPC)

All are variations on a theme:

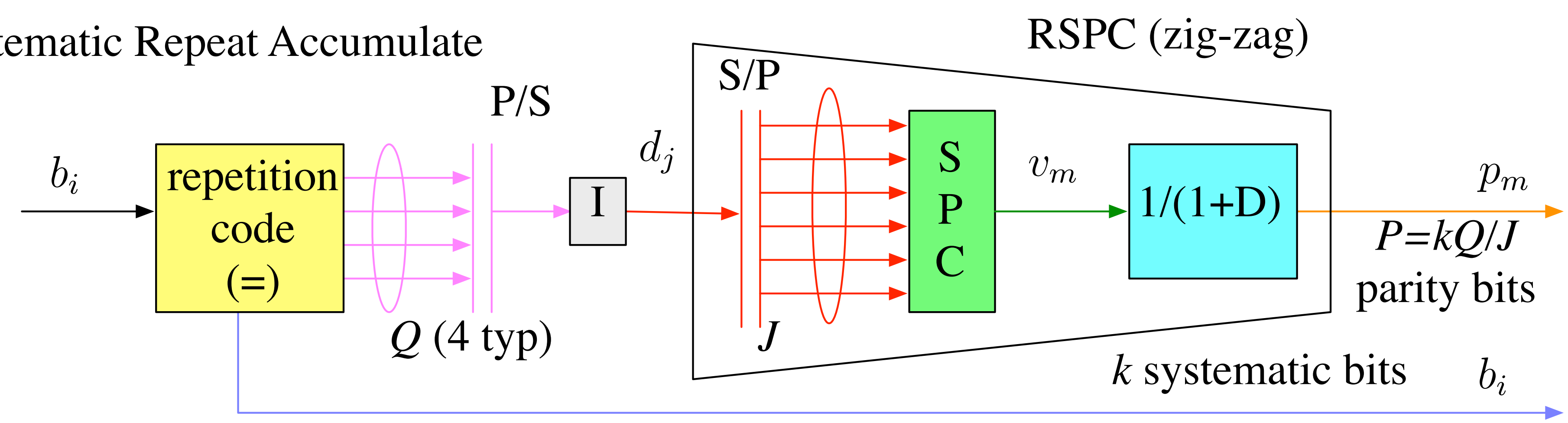
- Build big, global code from small local codes
- Local codes share common variables through permutations

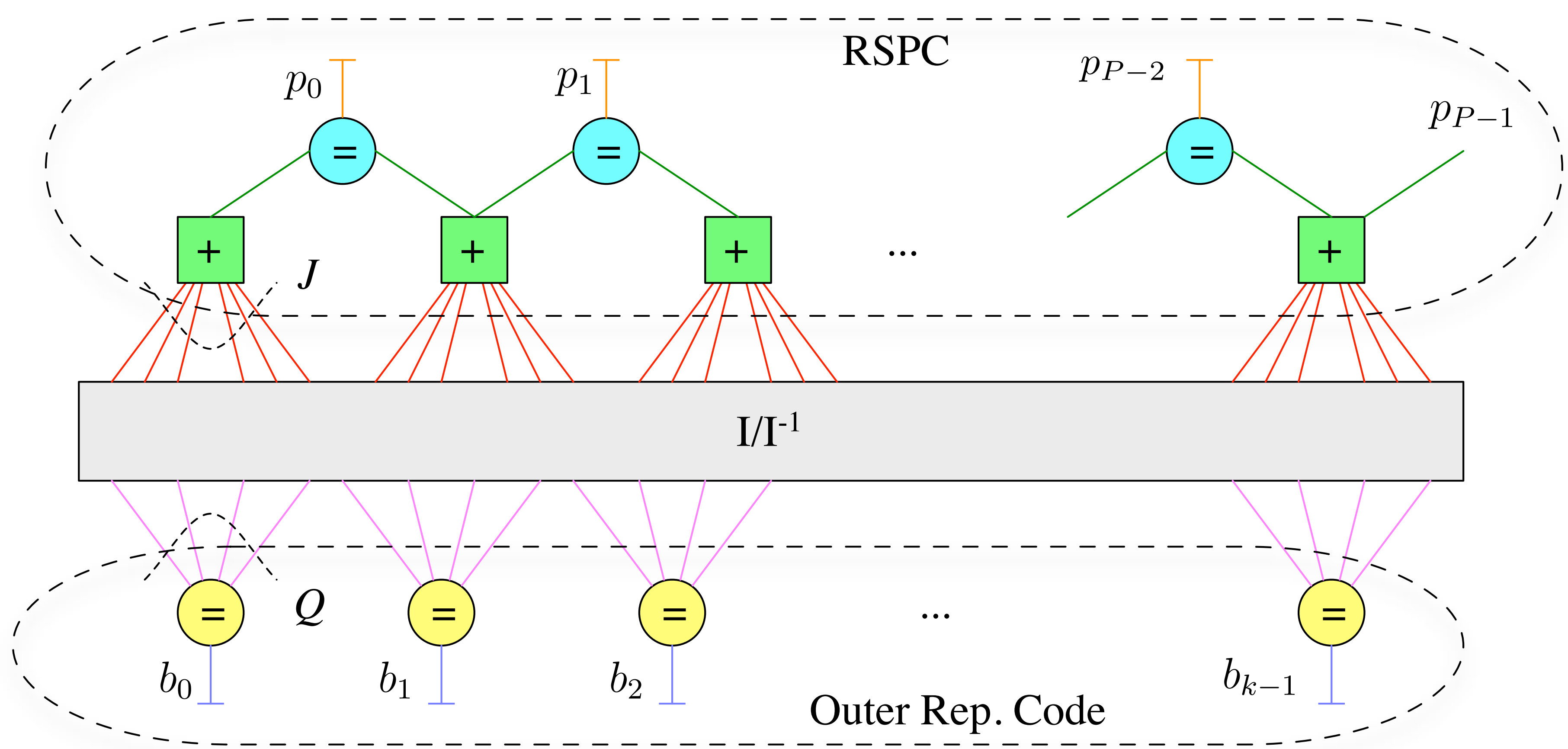
Modern Codes

Common performance trade-off



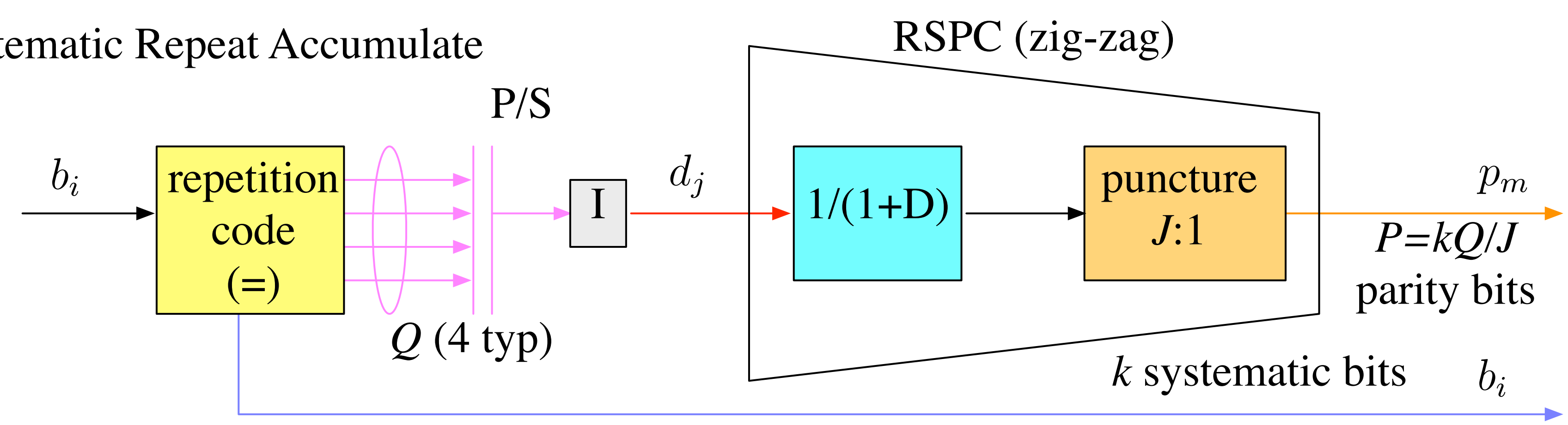
Modern Code Example: Systematic Repeat Accumulate

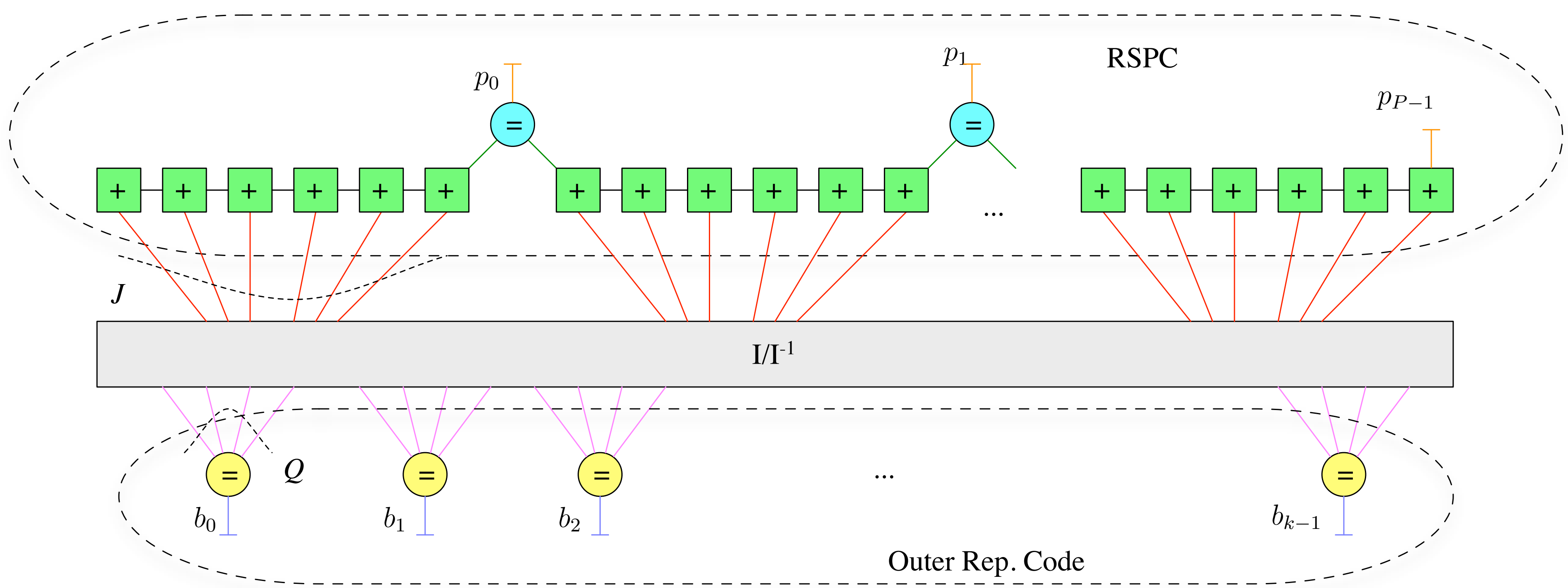


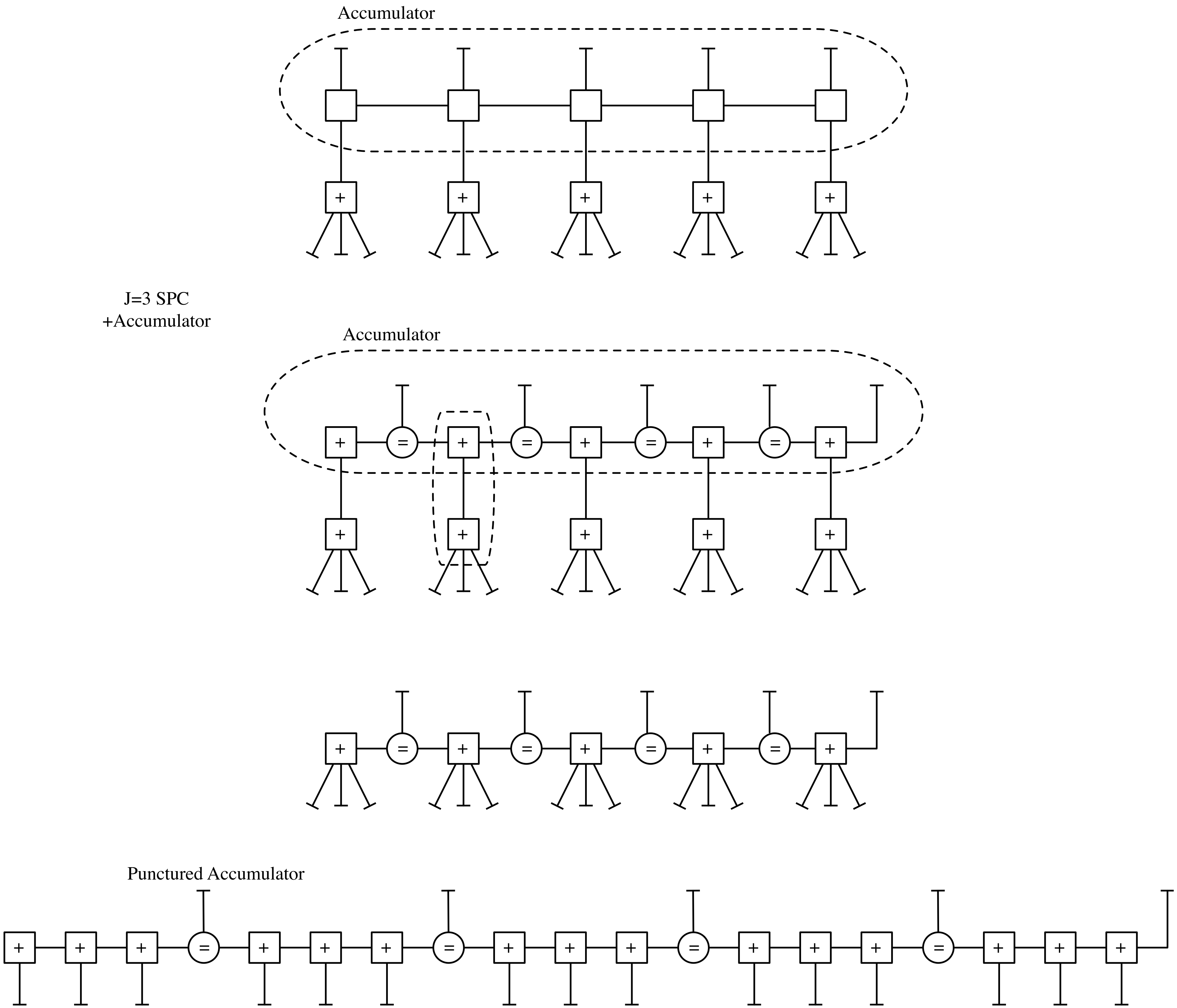


Modern Code Example: Systematic Repeat Accumulate

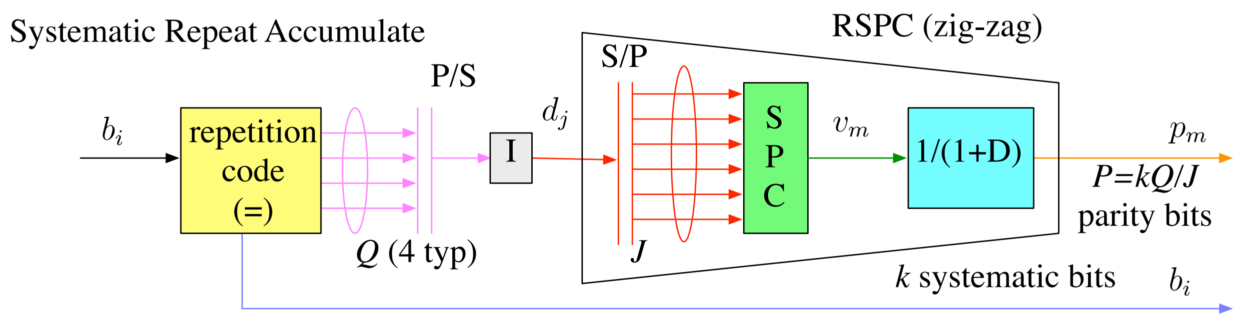
punctured accumulator model






Accumulator
J=3 SPC
+Accumulator
Accumulator
Punctured Accumulator

Modern Code Example: Systematic Repeat Accumulate



$$v_m = d_{mJ} + d_{mJ+1} + \cdots d_{mJ+(J-1)}$$

$$p_m = p_{m-1} + v_m$$

$$0 = p_m + p_{m-1} + (d_{mJ+1} + \cdots d_{mJ+(J-1)})$$

$$d_{I(i)} = b_i$$

$$\mathbf{Hc} = \left[\begin{array}{c|c} \mathbf{D} & \mathbf{S} \end{array} \right] \begin{bmatrix} \mathbf{b} \\ \mathbf{p} \end{bmatrix} = \mathbf{O}$$

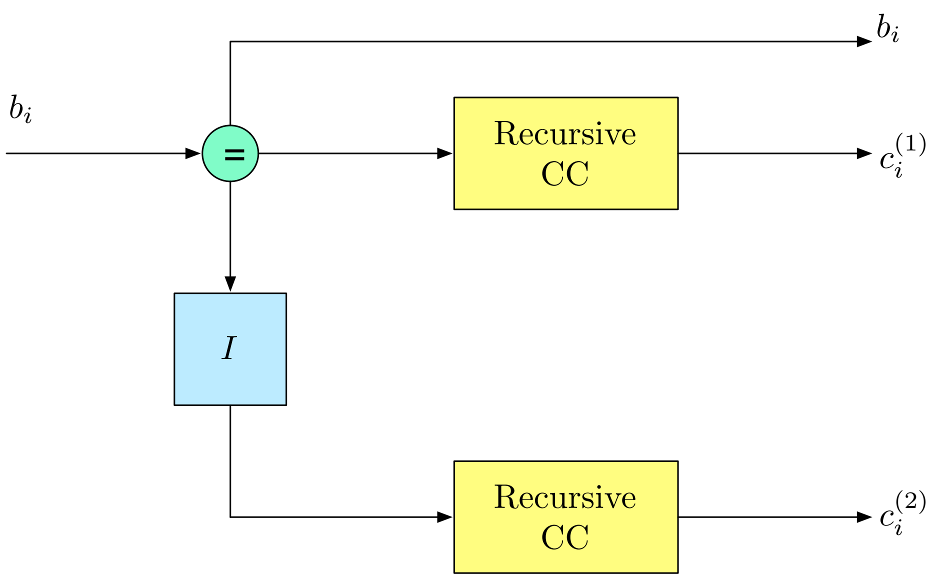
$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & \cdots & 0 & 1 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 1 \end{bmatrix}$$

J 1's per row

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 1 \\ \vdots & & & \ddots & & \vdots \\ 0 & 1 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

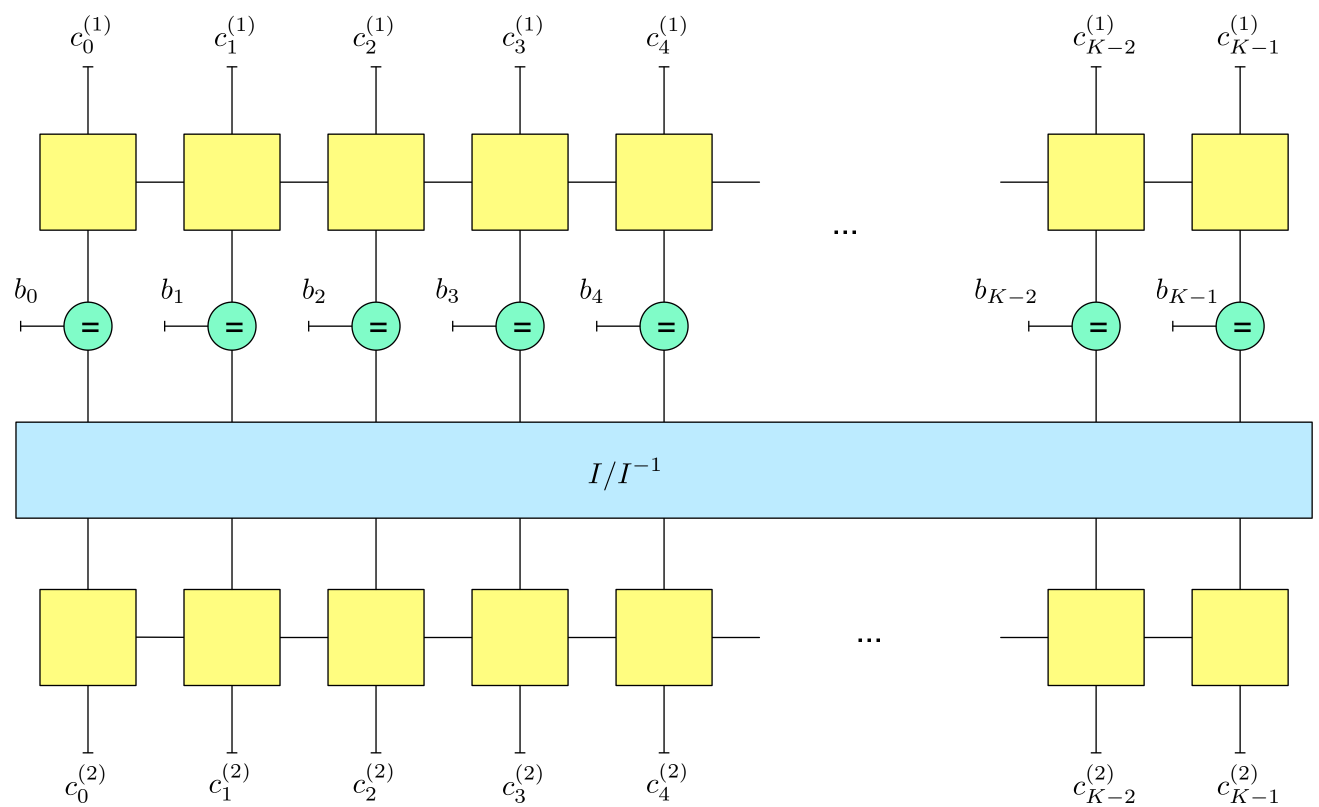
Q 1's per column

Modern Code Example: PCCCC

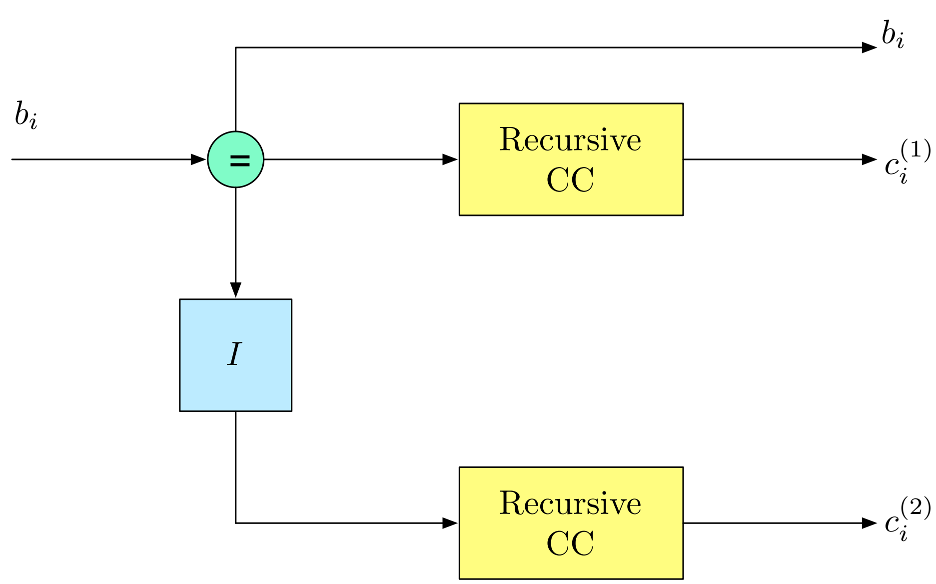


Encoder Block Diagram

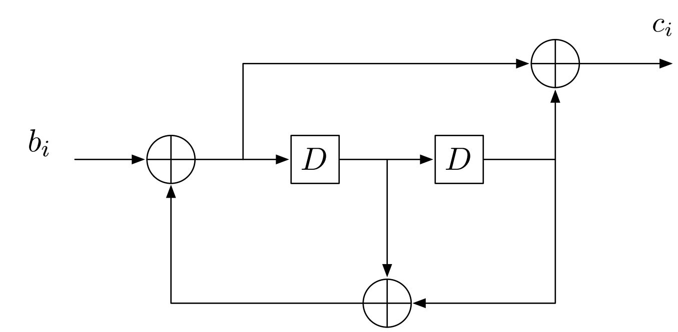
Graphical Model



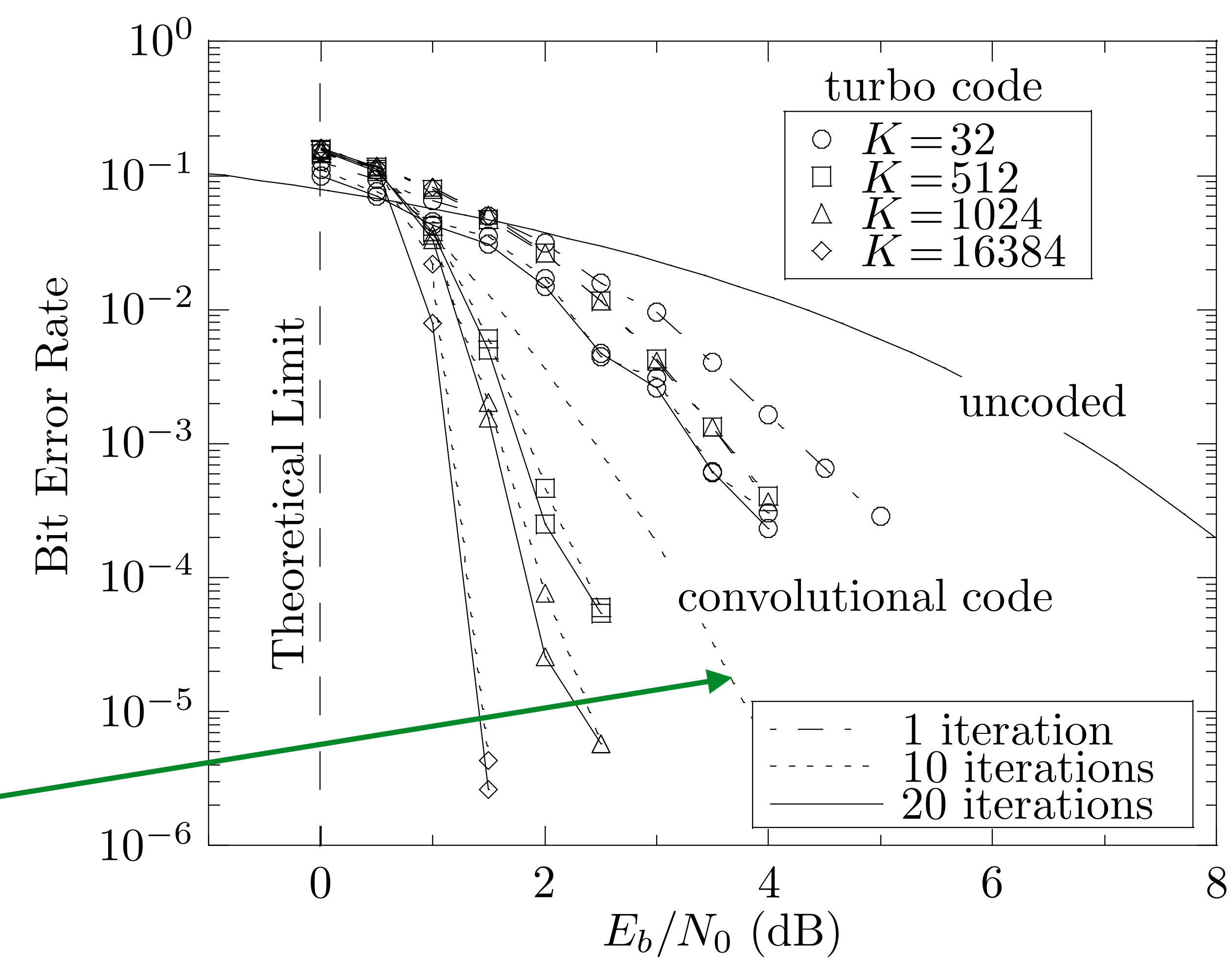
Modern Code Example: PCCCC



Encoder Block Diagram

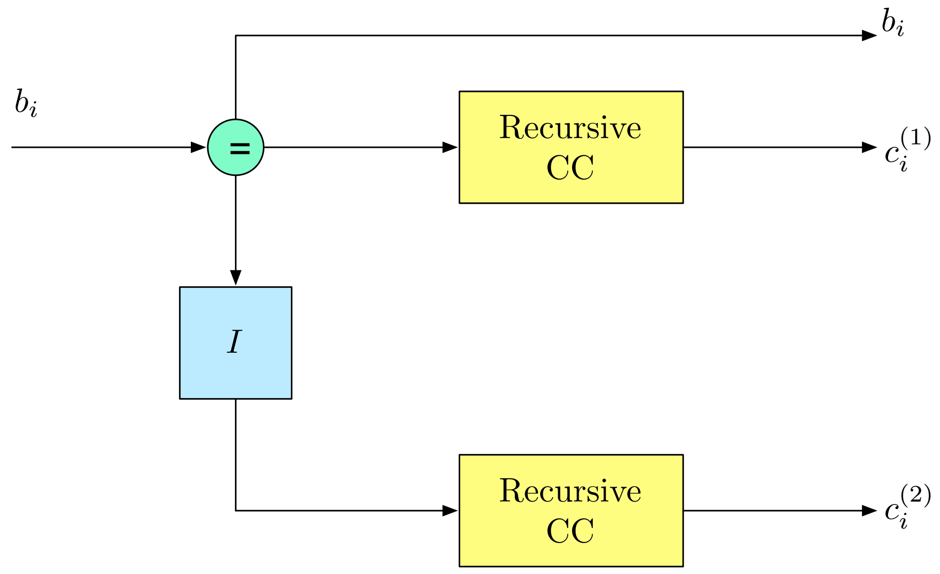


$$G(D) = \frac{1 + D^2}{1 + D + D^2}$$

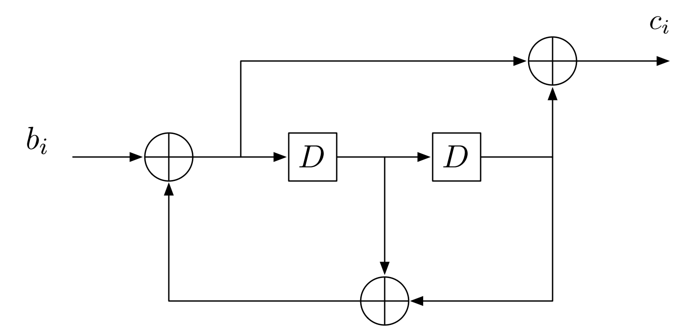


128-state CC

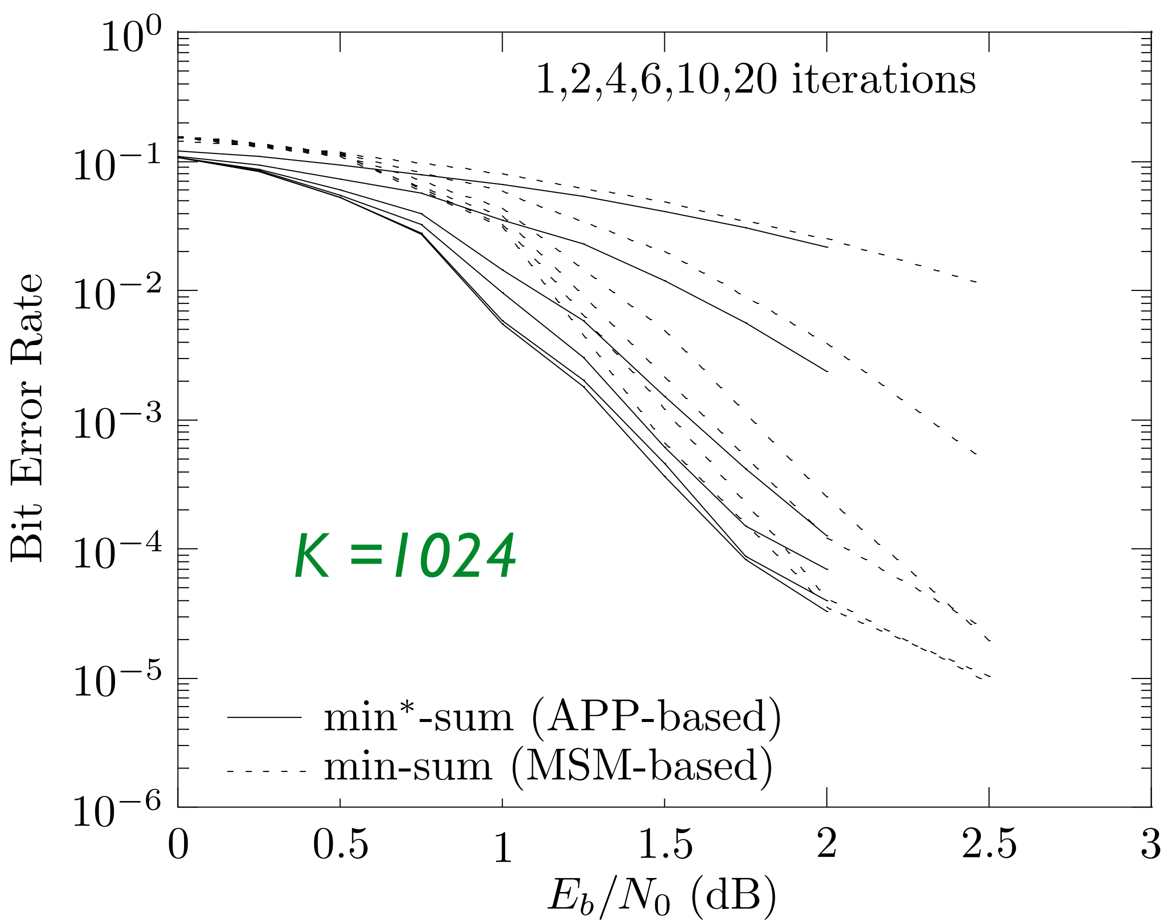
Modern Code Example: PCCCC



Encoder Block Diagram



$$G(D) = \frac{1 + D^2}{1 + D + D^2}$$



Coding Topics

- Coding channel models
- Basics of code constructions
- Decoding rules — HIHO, SIHO, SISO
- Classical coding
- Modern Coding
- Performance limits
  - Capacity and finite block-size bounds)
  - Bounds for specific codes

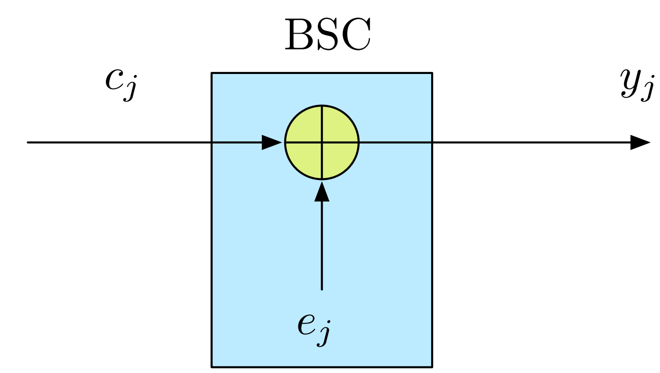
Decoding: Hard-in/Hard-out

MAP codeword decoding over the BSC

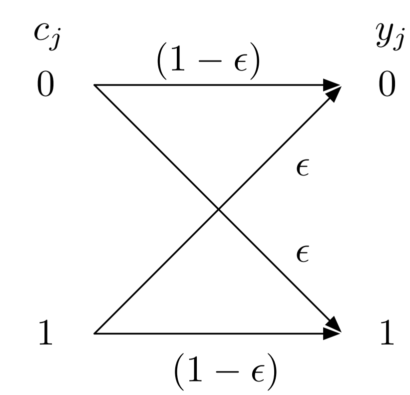
Assuming all inputs bits are iid, Bernoulli(1/2):

$$p_{\mathbf{y}(u)|\mathbf{c}(u)}(\mathbf{y}|\mathbf{c}) = \prod_{j=0}^{n-1} p_{y_j(u)|c_j(u)}(y_j|c_j) = \epsilon^{d_H(\mathbf{y},\mathbf{c})}(1-\epsilon)^{n-d_H(\mathbf{y},\mathbf{c})}$$

$$-\ln\left[p_{\mathbf{y}(u)|\mathbf{c}(u)}(\mathbf{y}|\mathbf{c})\right] \equiv d_H(\mathbf{y},\mathbf{c}) \ln\left[\frac{1-\epsilon}{\epsilon}\right]$$



$e_j(u) \sim$ iid Bernoulli(ϵ)



labels: $p_{y_j(u)|c_j(u)}(y_j|c_j)$

ML CW Decoding =
Minimum Hamming Distance Decoding

$$\boxed{\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathcal{C}} d_H(\mathbf{y},\mathbf{c})}$$

Minimum Distance of Linear Code

$$d_{\min} = \arg \min_{\mathbf{c} \neq \tilde{\mathbf{c}} \in \mathcal{C}} d_H(\mathbf{c}, \tilde{\mathbf{c}})$$

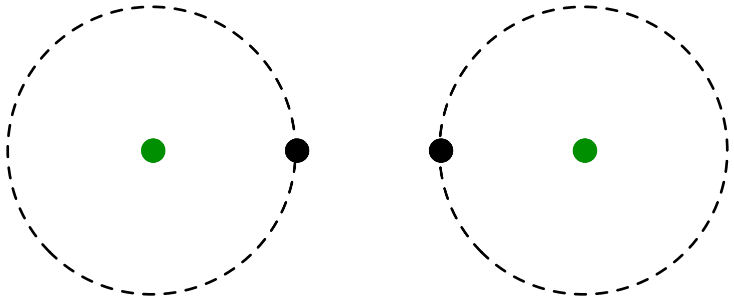
$$= \arg \min_{\mathbf{c} \neq \tilde{\mathbf{c}} \in \mathcal{C}} d_H(\mathbf{0}, \mathbf{c} + \tilde{\mathbf{c}})$$

$$= \arg \min_{\mathbf{c} \neq \mathbf{0} \in \mathcal{C}} d_H(\mathbf{0}, \mathbf{c})$$

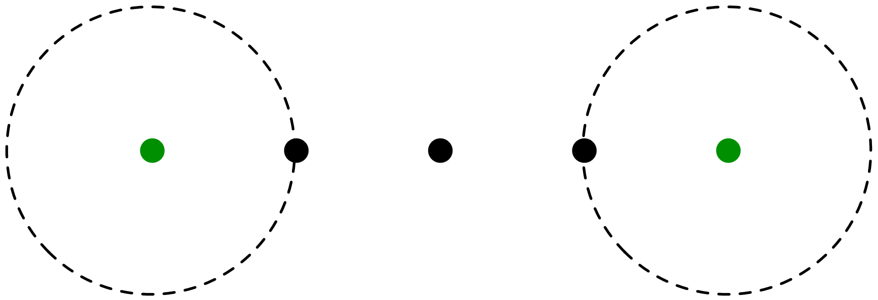
linear code: sum of codewords is a codeword

Minimum (Hamming) distance or minimum (Hamming) weight of the code

$d_{\min} = 3$

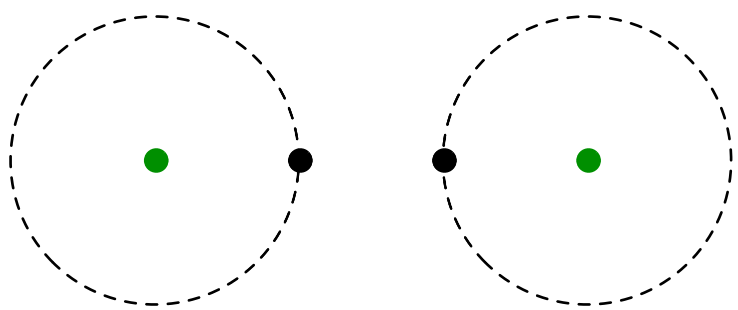


$d_{\min} = 4$



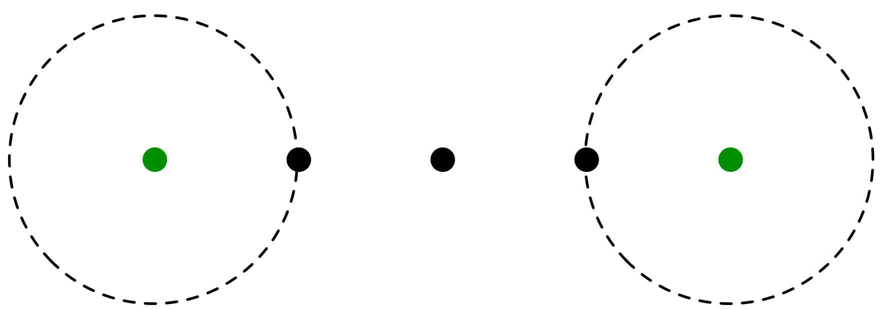
Error Correction Capability of Linear Code

$d_{\min} = 3$



Can correct all errors
of weight 0 or 1

$d_{\min} = 4$



Can correct all errors
of weight 0 or 1

Error correction capability of code

$$t_c = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

can correct all error patterns of weight t_c or smaller

Decoding: Hard-in/Hard-out

minimum Hamming distance decoding via the standard array

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

	Coset leader								Syndrome
Codewords	000000	100110	010011	001111	110101	101001	011100	111010	$(000)^T$
	000001	100111	010010	001110	110100	101000	011101	111011	$(001)^T$
	000010	100100	010001	001101	110111	101011	011110	111000	$(010)^T$
Elements of a coset	000100	100010	010111	001011	110001	101101	011000	111110	$(100)^T$
	001000	101110	011011	000111	111101	100001	010100	110010	$(111)^T$
	010000	110110	000011	011111	100101	111001	001100	101010	$(011)^T$
	100000	000110	110011	101111	010101	001001	111100	011010	$(110)^T$
	000101	100011	010110	001010	110000	101100	011001	111111	$(101)^T$

coset of code for each syndrome is code + coset leader

coset leader = min weight element of coset

Figure 4: Standard array for [6,3,3] code. (Kumars Notes)

Minimum Hamming Distance Decoding via Syndromes

1. Priori to decoding, for each of the 2^{n-k} cosets, store the minimum weight element. This is the coset leader: $\mathbf{l}(\mathbf{s})$.

2. When $\mathbf{y}$ is received, compute the syndrome $\mathbf{s} = \mathbf{H}\mathbf{y}$.

3. The minimum Hamming distance decision is: $\hat{\mathbf{c}} = \mathbf{y} + \mathbf{l}(\mathbf{s})$.

The standard array also includes all possible 2^n binary vectors arranged in cosets so that when a given n-tuple is received, it decodes to the codeword above it in the zero-coset.

Coset leader

codeword decision = y+coset leader

Syndrome

Codewords

Elements of a coset

000000	100110	010011	001111	110101	101001	011100	111010	$(000)^T$
000001	100111	010010	001110	110100	101000	011101	111011	$(001)^T$
000010	100100	010001	001101	110111	101011	011110	111000	$(010)^T$
000100	100010	010111	001011	110001	101101	011000	111110	$(100)^T$
001000	101110	011011	000111	111101	100001	010100	110010	$(111)^T$
010000	110110	000011	011111	100101	111001	001100	101010	$(011)^T$
100000	000110	110011	101111	010101	001001	111100	011010	$(110)^T$
000101	100011	010110	001010	110000	101100	011001	111111	$(101)^T$

s = Hy

coset leader = min weight element of coset

y

Figure 4: Standard array for [6,3,3] code. (Kumars Notes)

Interpreting the Standard Array

Note that the coset leaders are all of the correctable error patterns

All weight t_c and below vectors must be coset leaders!

Typically, will have some coset leaders with weight t_c+1 which means that the code can correct some patterns of weight t_c+1

	Coset leader								Syndrome
Codewords	000000	100110	010011	001111	110101	101001	011100	111010	$(000)^T$
	000001	100111	010010	001110	110100	101000	011101	111011	$(001)^T$
	000010	100100	010001	001101	110111	101011	011110	111000	$(010)^T$
Elements of a coset	000100	100010	010111	001011	110001	101101	011000	111110	$(100)^T$
	001000	101110	011011	000111	111101	100001	010100	110010	$(111)^T$
	010000	110110	000011	011111	100101	111001	001100	101010	$(011)^T$
	100000	000110	110011	101111	010101	001001	111100	011010	$(110)^T$
coset leader = min weight element of coset	000101	100011	010110	001010	110000	101100	011001	111111	$(101)^T$

Figure 4: Standard array for [6,3,3] code. (Kumars Notes)

Performance of HIHO Decoding on BSC

Since all weight t_c and lower error patterns are correctable:

$$1 - P_{CW} = 1 - \textrm{P}\textsc{r}\left\{\hat{\mathbf{c}}(u) \neq \mathbf{c}(u)\right\}$$

$$\geq \textrm{P}\textsc{r}\left\{w_H(\mathbf{e}(u)) \leq t_c\right\}$$

$$= \sum_{w=0}^{t_c} \binom{n}{w} \epsilon^w (1-\epsilon)^{n-w}$$

$$\boxed{\begin{aligned} P_{CW} &\leq 1 - \sum_{w=0}^{t_c} \binom{n}{w} \epsilon^w (1-\epsilon)^{n-w} \\ &= \sum_{w=t_c+1}^{n} \binom{n}{w} \epsilon^w (1-\epsilon)^{n-w} \end{aligned}}$$

$$\approx \binom{n}{t_c+1} \epsilon^{(t_c+1)} (1-\epsilon)^{n-(t_c+1)}$$ small epsilon

Performance of HIHO Decoding on BSC

If you have the coset leaders:

$$P_{CW} = \textsc{Pr}\left\{\mathbf{e}(u) \neq \text{a coset leader}\right\}$$

$$= 1 - \textsc{Pr}\left\{\mathbf{e}(u) \text{is a coset leader}\right\}$$

For example (6,3,3) code:

Coset leader
000000
000001
000010
000100
001000
010000
100000
000101

$$P_{CW} = 1 - \left[(1-\epsilon)^6 + 6\epsilon(1-\epsilon)^5 + \epsilon^2(1-\epsilon)^4\right]$$

Note that the bound yields:

$$P_{CW} \leq 1 - \left[(1-\epsilon)^6 + 6\epsilon(1-\epsilon)^5\right]$$

Interpreting the Standard Array

The number of coset leaders: 2^{n-k}

Coset leaders with weight <= t_c: $\sum_{w=0}^{t_c} \binom{n}{w}$

$$\boxed{\sum_{w=0}^{t_c} \binom{n}{w} \leq 2^{n-k}}$$

This is a bound on d_min — Sphere packing or Hamming bound

$$\sum_{w=0}^{t_c} \binom{n}{w} = 2^{n-k}$$

Possible?

Yes: called a "perfect code" (rare)

only 3 known perfect binary codes

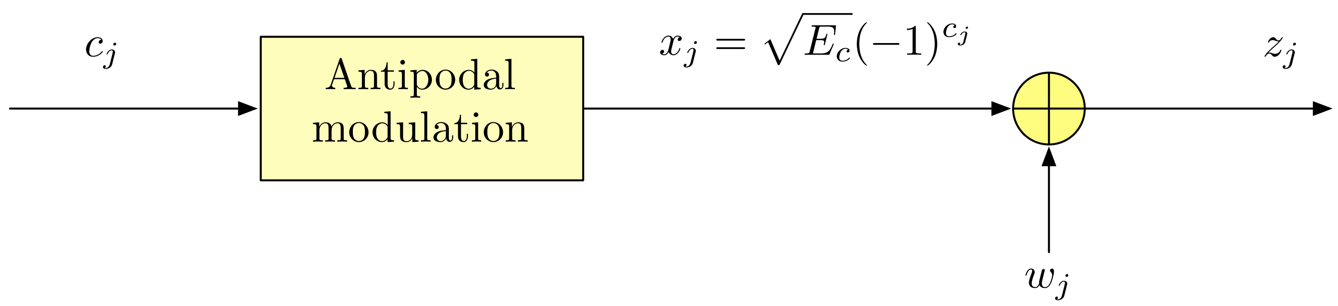
Hamming code is perfect

(see page 470 of Benedetto for the standard Array for the (7,4,3) Hamming code)

(n,1,n) repetition code is perfect for n odd

(23,12,7) Golay code is perfect

Decoding: Soft-in/Hard-out



$$f_{z_j(u)|c_j(u)}(z|c) = \mathcal{N}\left(z; \sqrt{E_c}(-1)^c; N_0/2\right)$$

$$w_j(u) \sim \mathcal{N}(\cdot; 0; N_0/2)$$

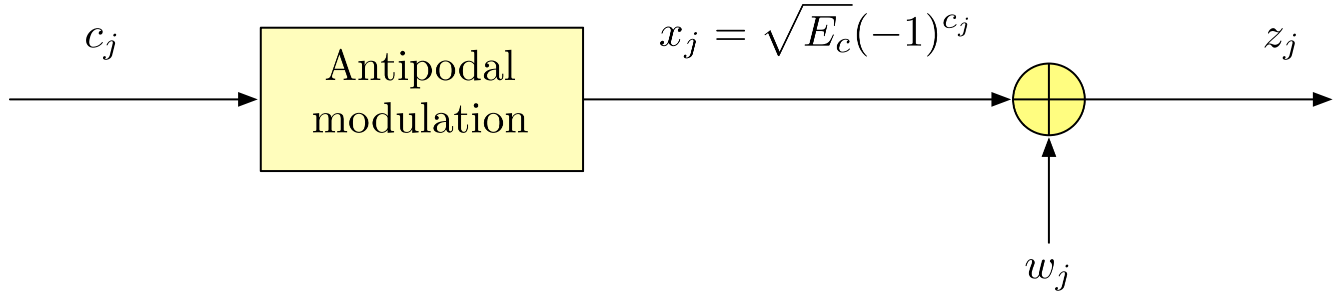
$$f_{\mathbf{z}(u)|\mathbf{c}(u)}(\mathbf{z}|\mathbf{c}) = \prod_{j=0}^{n-1} f_{z_j(u)|c_j(u)}(y_j|c_j)$$

$$-\ln\left[f_{\mathbf{z}(u)|\mathbf{c}(u)}(\mathbf{z}|\mathbf{c})\right] \equiv \frac{1}{N_0}\|\mathbf{z} - \mathbf{x}(\mathbf{c})\|^2$$

ML CW Decoding =
Minimum Euclidean Distance Decoding

$$\boxed{\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{z} - \mathbf{x}(\mathbf{c})\|^2}$$

SIHO Decoding Performance (BI-AWGN)



$$w_j(u) \sim \mathcal{N}(\cdot; 0; N_0/2)$$

$$f_{z_j(u)|c_j(u)}(z|c) = \mathcal{N}\left(z; \sqrt{E_c}(-1)^c; N_0/2\right)$$

$$P(\mathcal{E}|\mathbf{c}) \leq \sum_{\tilde{\mathbf{c}} \neq \mathbf{c} \in \mathcal{C}} P_{PW}(\mathbf{c}, \tilde{\mathbf{c}})$$

$$P_{PW}(\mathbf{c}, \tilde{\mathbf{c}}) = \mathrm{Q}\left(\sqrt{\frac{\|\mathbf{x}(\mathbf{c}) - \mathbf{x}(\tilde{\mathbf{c}})\|^2}{2N_0}}\right)$$

$$= \mathrm{Q}\left(\sqrt{\frac{d_H(\mathbf{c}, \tilde{\mathbf{c}}) 4E_c}{2N_0}}\right)$$

$$= \mathrm{Q}\left(\sqrt{d_H(\mathbf{c}, \tilde{\mathbf{c}}) r \frac{2E_b}{N_0}}\right)$$

For a linear code, the CW error probability the same conditioned on any codeword — i.e., can condition on zero CW

SIHO Decoding Performance (BI-AWGN)

$$\boxed{\mathrm{Q}\left(\sqrt{d_{\min} r \frac{2E_b}{N_0}}\right) \leq P_{CW} \leq \sum_{d \geq d_{\min}} A_d \mathrm{Q}\left(\sqrt{d r \frac{2E_b}{N_0}}\right)}$$

$$A_d = \text{ number of codewords with weight } d$$

weight distribution of the code

SIHO Decoding Performance (BI-AWGN)

$$
\begin{aligned}
P_b &= P_{b|\mathbf{0}} \\
&= \sum_{\mathbf{b}\neq\mathbf{0}} \frac{w_H(\mathbf{b})}{k} \mathrm{P_R}\left\{\hat{\mathbf{c}}(u) = \mathbf{G}^{\mathrm{t}}\mathbf{b} | \mathbf{c}(u) = \mathbf{0}\right\} \\
&\leq \sum_{\mathbf{b}\neq\mathbf{0}} \frac{w_H(\mathbf{b})}{k} P_{PW}(\mathbf{G}^{\mathrm{t}}\mathbf{b}, \mathbf{0}) \\
&= \sum_{\mathbf{b}\neq\mathbf{0}} \frac{w_H(\mathbf{b})}{k} \mathrm{Q}\left(\sqrt{d_H(\mathbf{G}^{\mathrm{t}}\mathbf{b}, \mathbf{0}) r \frac{2E_b}{N_0}}\right)
\end{aligned}
$$

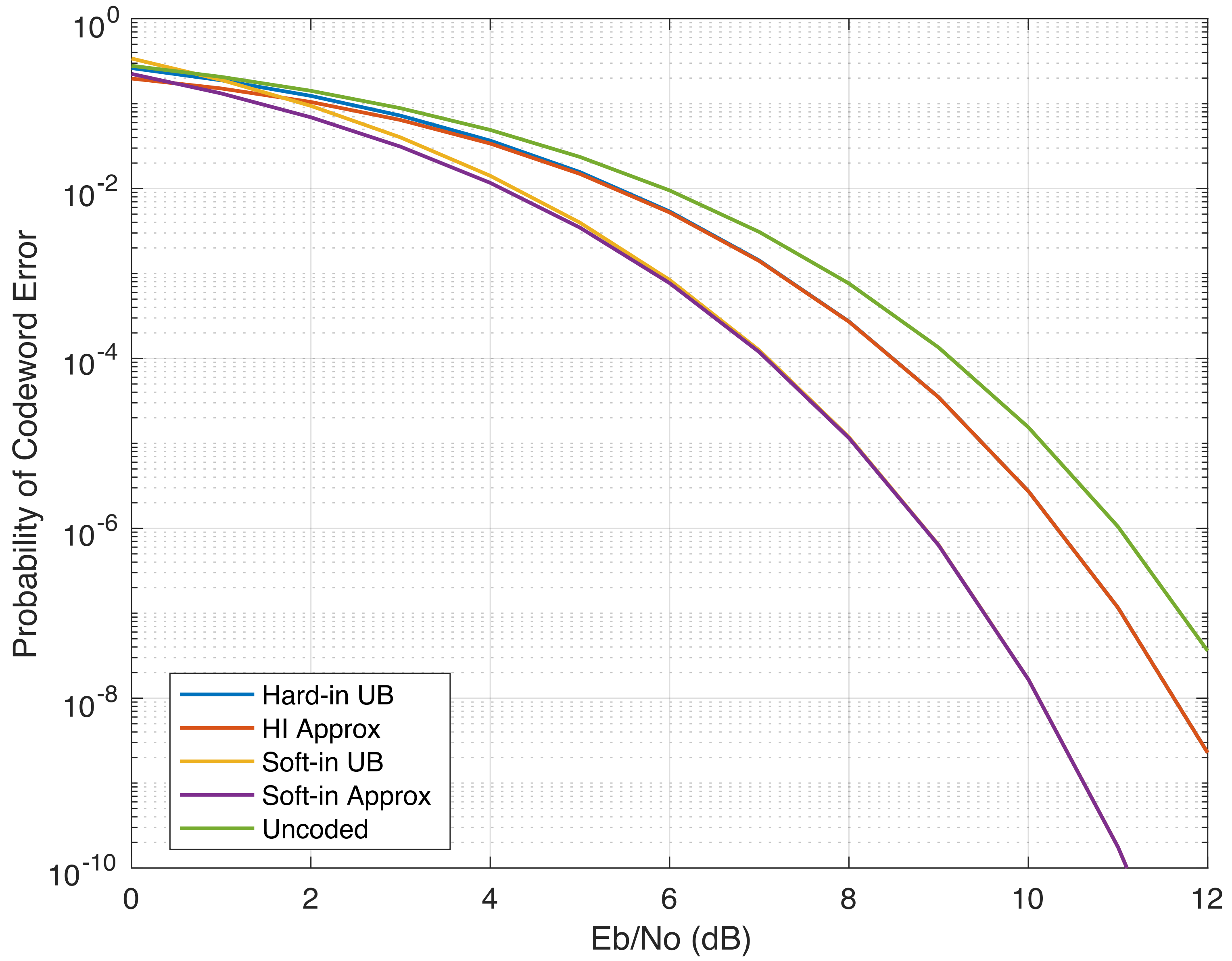
$$
\boxed{\frac{1}{k}\mathrm{Q}\left(\sqrt{d_{\min} r \frac{2E_b}{N_0}}\right) \leq P_b \leq \sum_{d \geq d_{\min}} K_d \mathrm{Q}\left(\sqrt{d r \frac{2E_b}{N_0}}\right)}
$$

$$
K_d = \sum_{w=1}^{k} \frac{w}{k} B_{w,d}
$$

$$
B_{w,d} = \text{ number of configurations with input weight } w \text{ and output weight } d
$$

Input/output weight distribution of the code

HIHO and SIHO Decoding Example



this is for the (7,4,3) Hamming Code

Other Bounds on Minimum Distance

Singleton Bound: $d_{\min} \leq (n-k)+1$

Mostly useful for non-binary codes — (non-binary) codes that achieve this bound are calls **Maximum Distance Separable (MDS)**.

Reed-Solomon codes are (non-binary) MDS codes. If you receive any k symbols of an MDS code, you can decode on erasure channel

Plotkin Bound: $d_{\min} \leq d_{\text{ave}}$

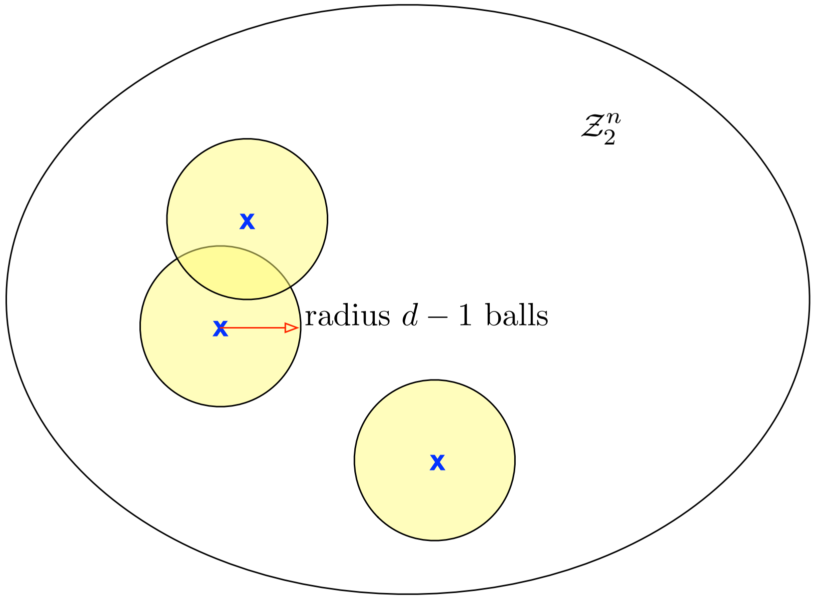
$d_{\min} < n/2:$ $\quad 2(d_{\min}-1) - \log_2(d_{\min}) \leq (n-k)$

$d_{\min} \geq n/2:$ $\quad d_{\min} \leq \dfrac{n2^{k-1}}{2^k - 1}$

For binary codes, the Hamming bound is usually tightest. Plotkin is tightest for very low rate codes

“Existence” Bounds on Minimum Distance

Suppose we build a code by randomly selecting a points, making sure that no two points are closer than d in Hamming distance?



Gilbert-Varshamov Bound

GV-1:

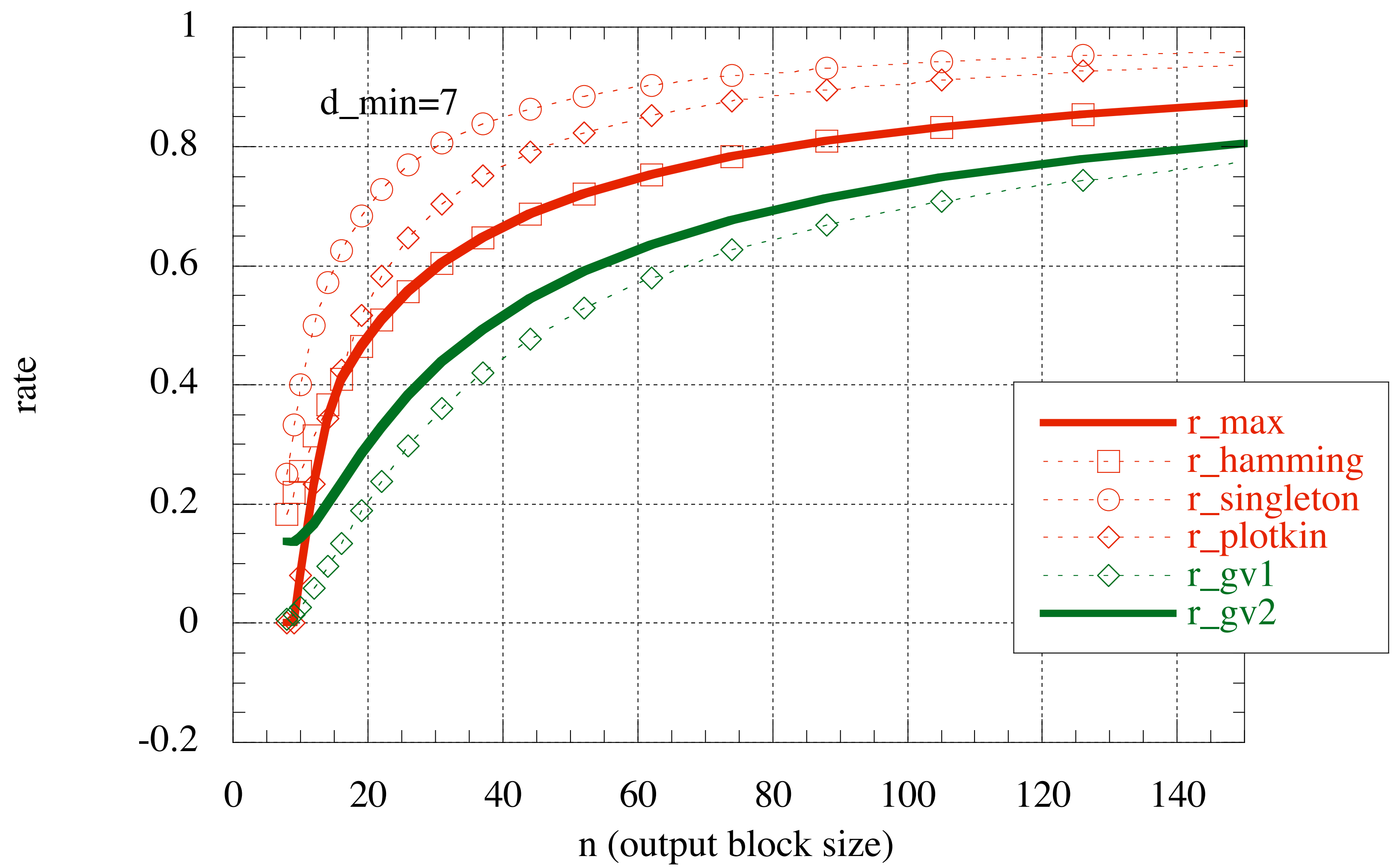
$$2^k \sum_{i=0}^{d-1} \binom{n}{i} < 2^n$$

If (n,k,d) satisfy the G-V bound, then there exists a code with these parameters

GV-2:

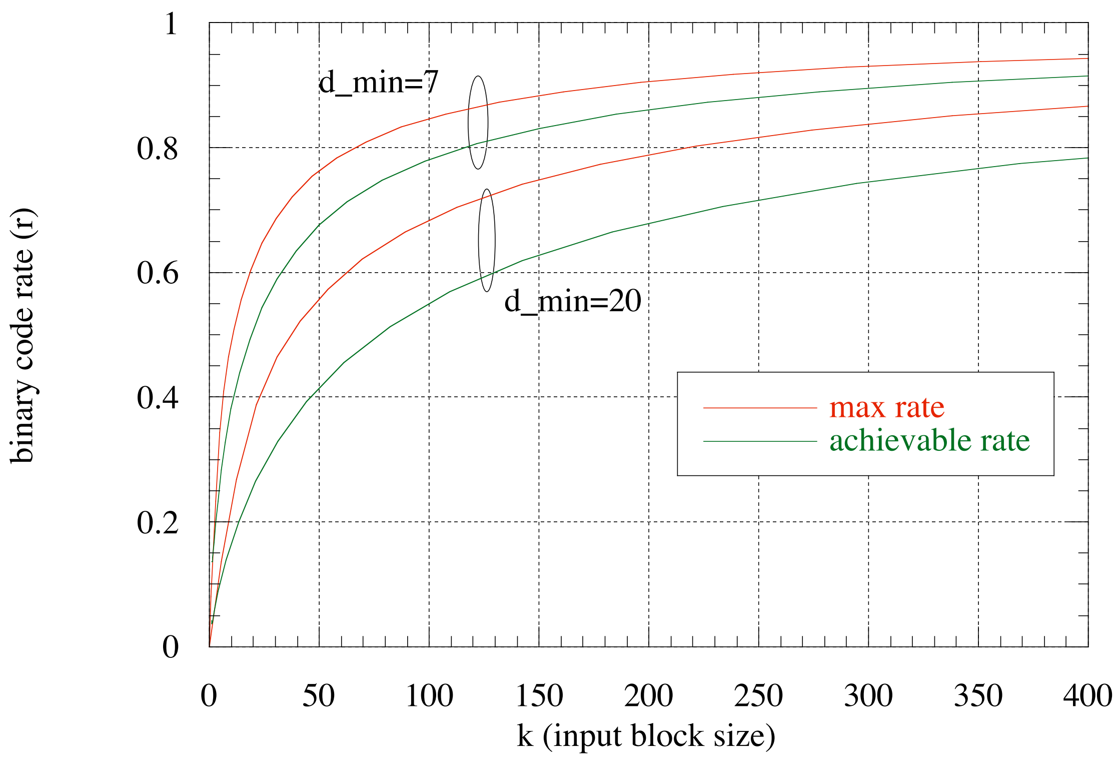
$$2^k \sum_{i=0}^{d-2} \binom{n-1}{i} < 2^n$$

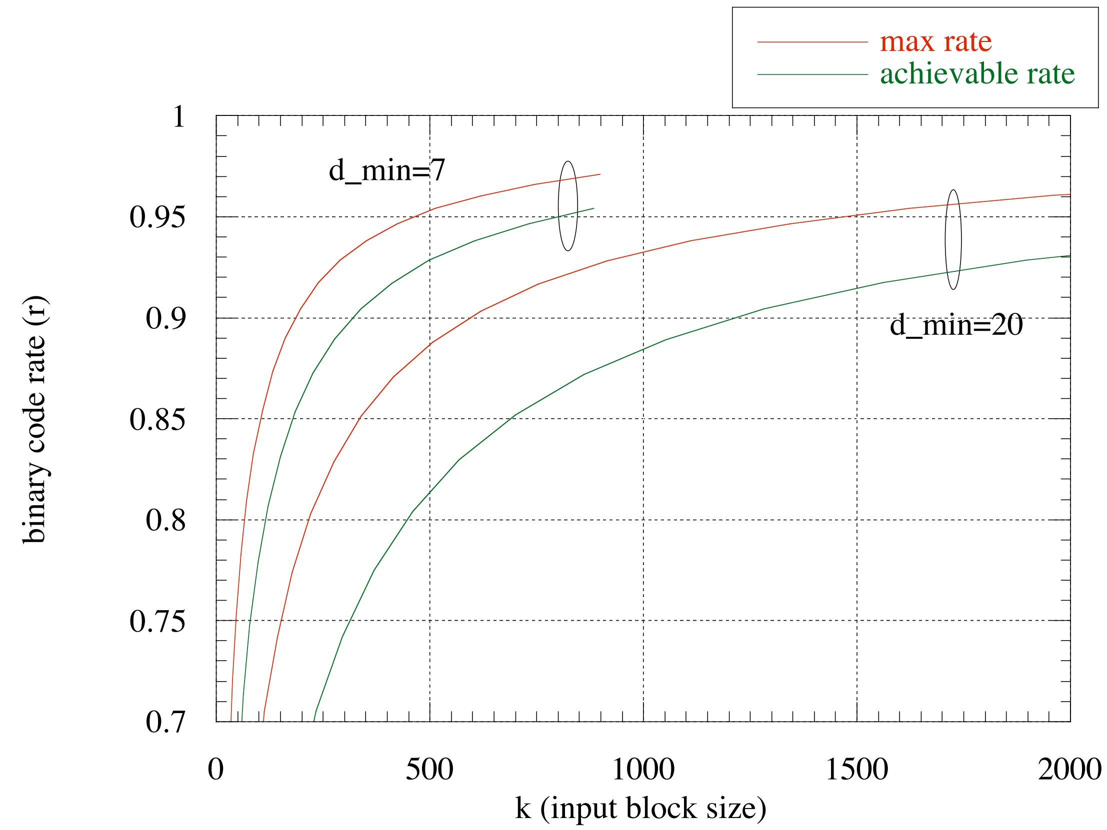
Bounds on Minimum Distance

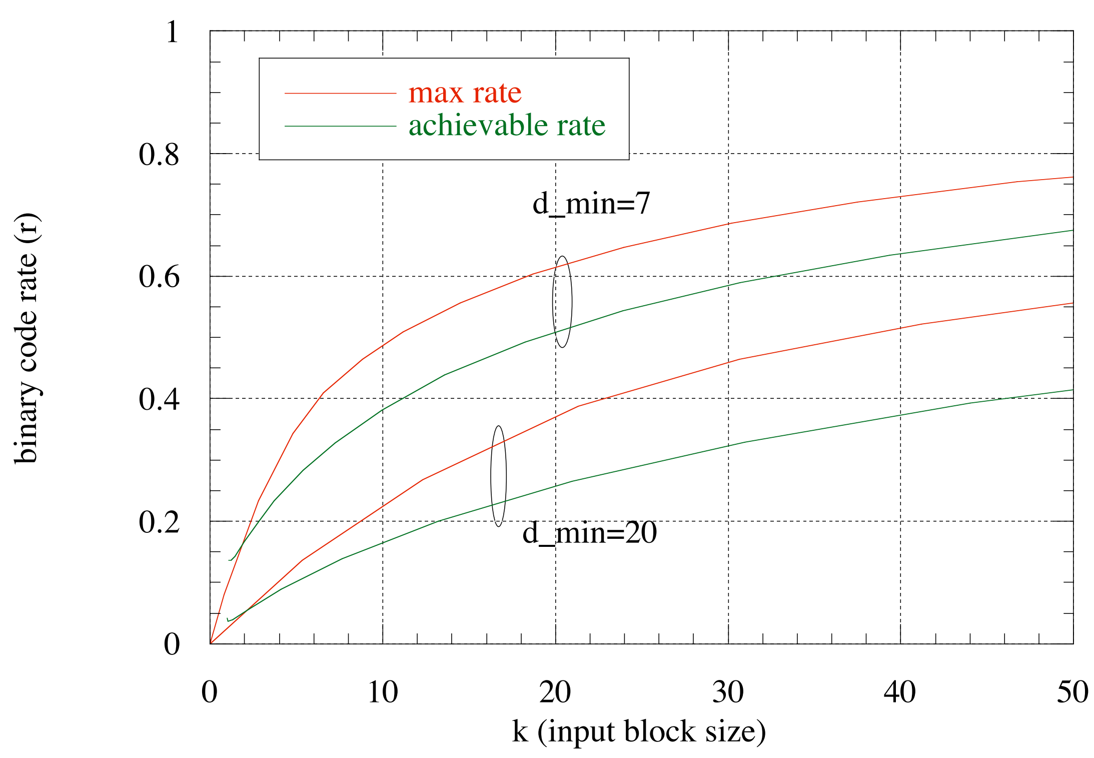


d_min = 7 codes exist with rate between the solid green and red curves

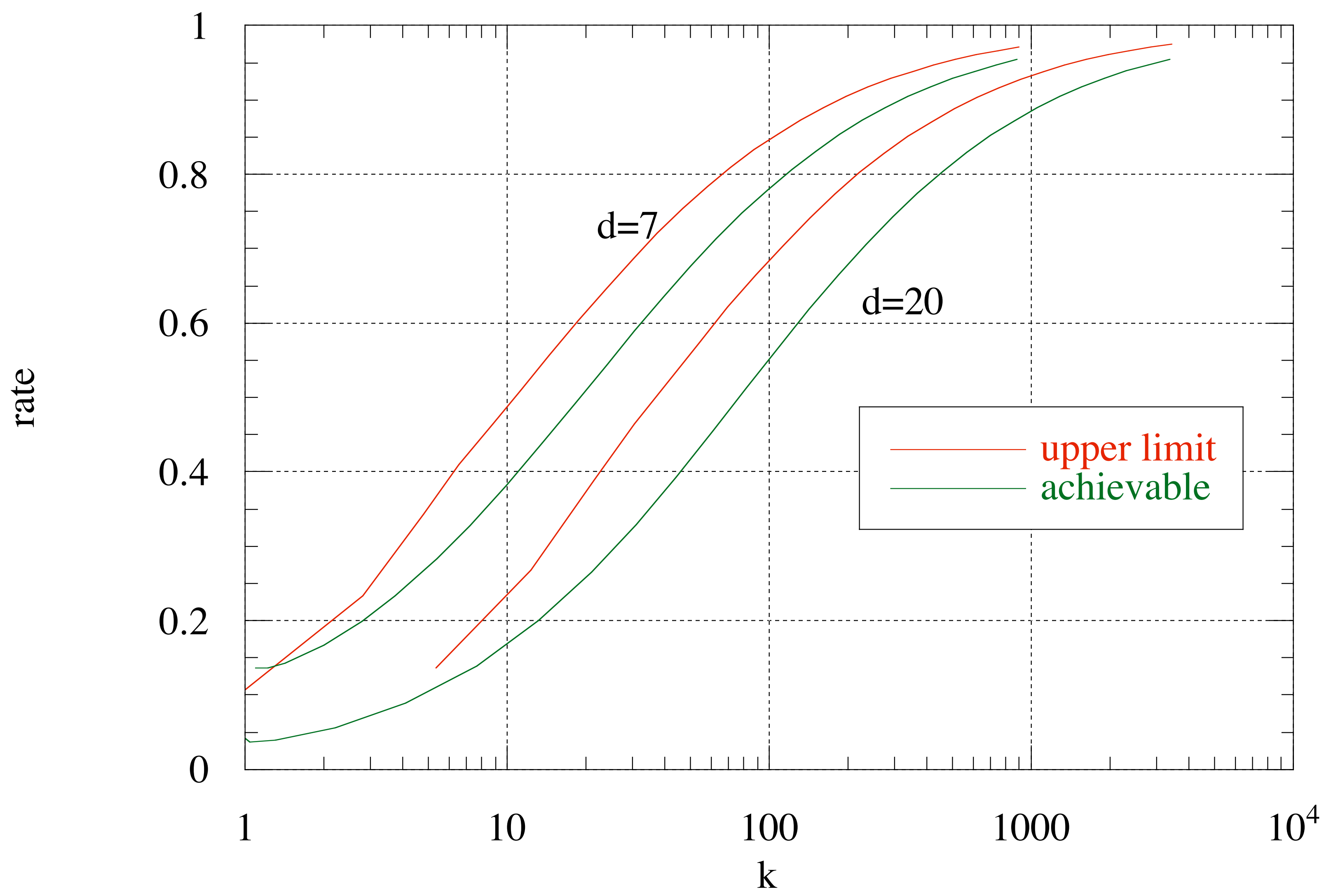
Bounds on Minimum Distance







Bounds on Minimum Distance



Hamming Family of Codes

This is a family of perfect, single error correcting block codes

$$
\begin{aligned}
m &= n - k \\
n &= 2^m - 1 \\
k &= 2^m - 1 - m \\
d_{\min} &= 3
\end{aligned}
$$

m = 2: (3,1,3) — aka repetition code

m = 3: (7,4,3)

m = 4: (15,11,3)

Note: the rate increases with block size

Construction: the parity check matrix has all non-zero (m x 1) binary vector

$$
\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}
$$

Reed-Mueller Family of Codes

$$\mathrm{RM}(r,m) \implies (n = 2^m, k_{r,m}, d_{\min} = 2^{m-r}) \qquad 0 \le r \le m$$

$$k_{r,m} = \sum_{j=0}^{r} \binom{m}{j}$$

r is called the **order** of the RM code

The $|u|u+v|$ construction suggests the following tableau of RM codes:

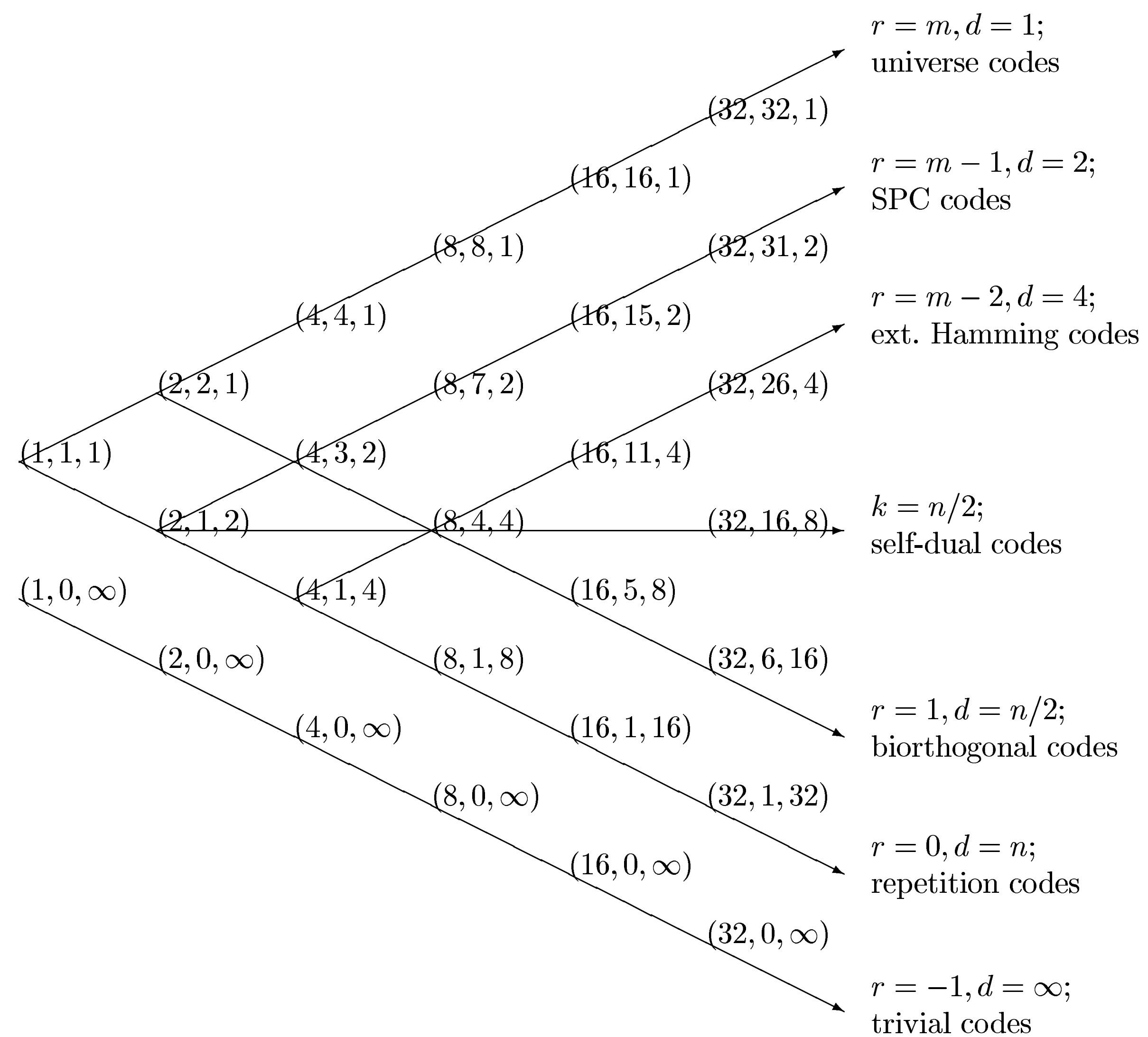


Figure 2. Tableau of Reed-Muller codes.

Forney's notes, 6.4

Reed-Mueller Family of Codes

Construction: many constructions. Here is on based on Hadamard matrices

$$\mathbf{U}_0 = 1$$

$$\mathbf{U}_1 = \begin{bmatrix} \mathbf{U}_0 & \mathbf{U}_0 \\ \mathbf{U}_0 & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{U}_2 = \begin{bmatrix} \mathbf{U}_2 & \mathbf{U}_2 \\ \mathbf{U}_2 & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$U_m \sim 2^m \times 2^m$$

$$\mathbf{U}_i = \begin{bmatrix} \mathbf{U}_{i-1} & \mathbf{U}_{i-1} \\ \mathbf{U}_{i-1} & \mathbf{0} \end{bmatrix}$$

$RM(r, m)$ has generator comprising all rows of U_m with weight 2^{m-r} or greater

Reed-Mueller Family of Codes

Construction: example RM(1,3) code which is (8,4,4) code

$RM(r,m)$ has generator comprising all rows of U_m with weight 2^{m-r} or greater

$r = 1, m = 3$ $\qquad 2^{m-r} = 4$

$$\mathbf{U}_3 = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dual Codes

Original Code:

$$\mathcal{C} : (n, k, d)$$

$$\text{Generator} : \mathbf{G}, (k \times n)$$

$$\text{Parity Check} : \mathbf{H}, (n - k \times n)$$

Dual Code:

$$\mathcal{C}^{\perp} : (n, k^{\perp} = n - k, d^{\perp})$$

$$\text{Generator} : \mathbf{G}^{\perp} = \mathbf{H}, (k^{\perp} \times n)$$

$$\text{Parity Check} : \mathbf{H}^{\perp} = \mathbf{G}, (n - k^{\perp} \times n)$$

It is possible to be self-dual — i.e., the the generator **G** is a valid parity check matrix **H**!

Example: (8,4,4) RM code on previous slide

Weight Enumerating Function

$$A_d = \text{number of codewords with weight } d$$

$$A(D) = \sum_{d=0}^{n} A_d D^d \quad \text{(weight enumerating function)}$$

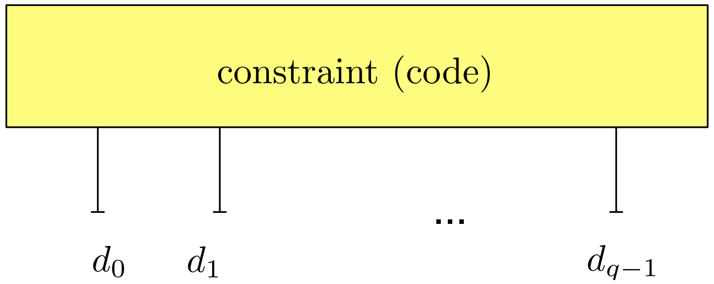
Example: (7,4,3) Hamming Code: $A(D) = 1 + 7D^3 + 7D^4 + D^7$

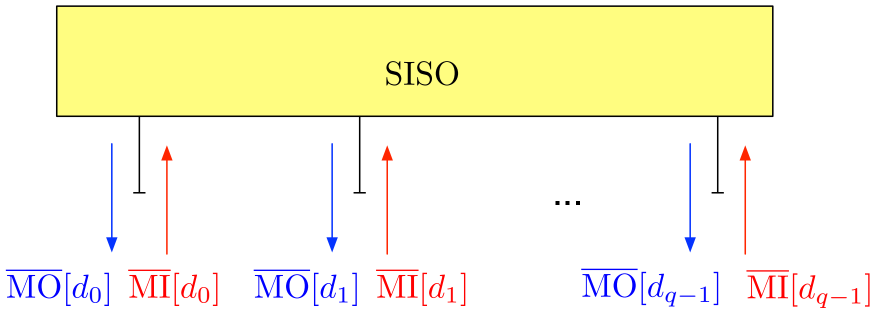
MacWilliams Identity:

$$A_{\text{dual}}(D) = 2^{-k}(1+D)^n A\left(\frac{1-D}{1+D}\right)$$

The WEF of the dual code is determined from the original code

Decoding: Soft-in/Soft-out





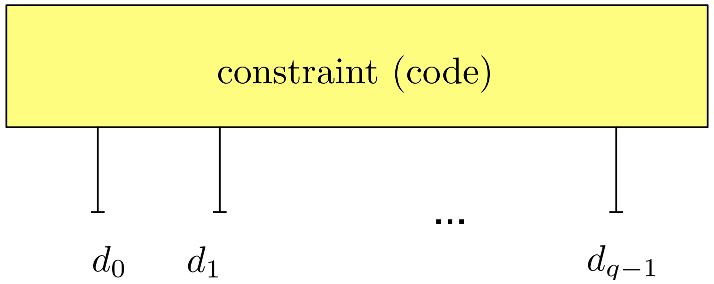
1. **Combine** incoming marginal metrics to get configuration metrics for all valid configurations

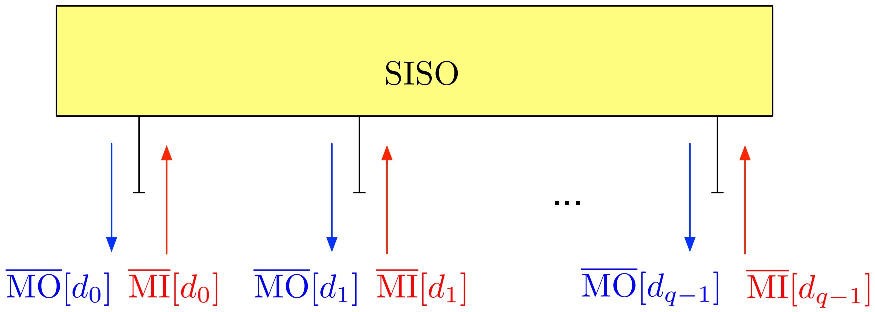
$$\overline{\mathrm{M}}[\text{config} = m] = \sum_j \overline{\mathrm{MI}}[d_j(m)]$$

2. **Marginalize** configuration metrics to get outgoing marginal metrics

$$\overline{\mathrm{MO}}[d_j] = \left(\min_{m:d_j=1} \overline{\mathrm{M}}[\text{config} = m] - \min_{m:d_j=0} \overline{\mathrm{M}}[\text{config} = m] \right) - \overline{\mathrm{MI}}[d_j]$$

Decoding: Soft-in/Soft-out



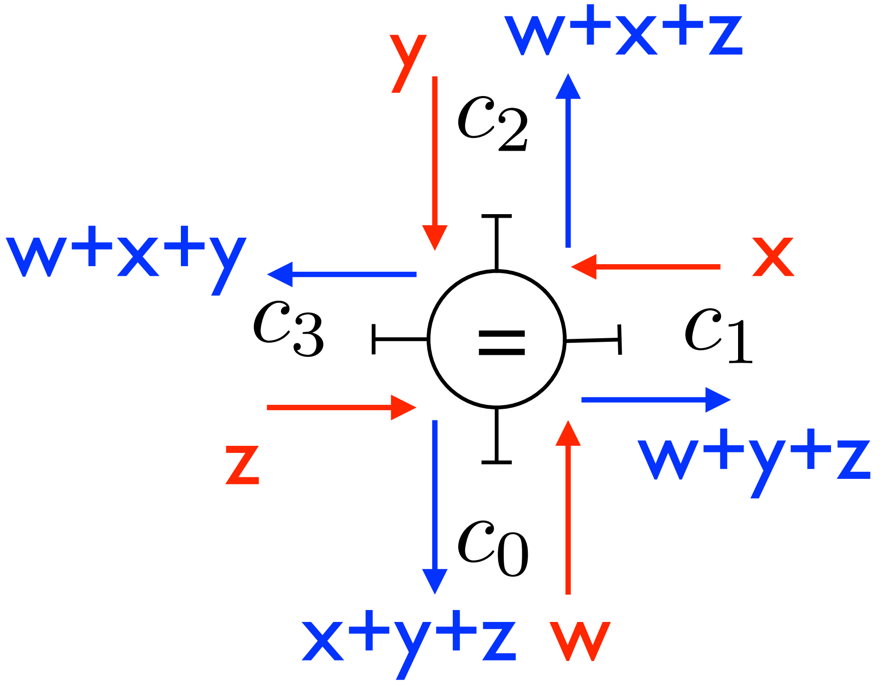


see SISO summary handout and
633_SISO.xlsx

Example: Repetition Code SISO

Special case of degree 4

	config	config metric
m=0:	0000	0
m=1:	1111	w+x+y+z



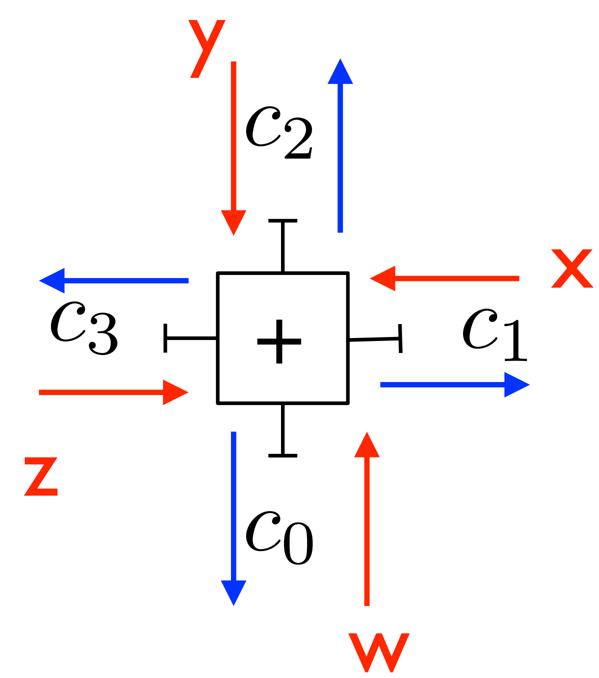
Note that there is no marginalizing in this case
min-sum and min*-sum are same

Example: SPC SISO

Consider degree 4:

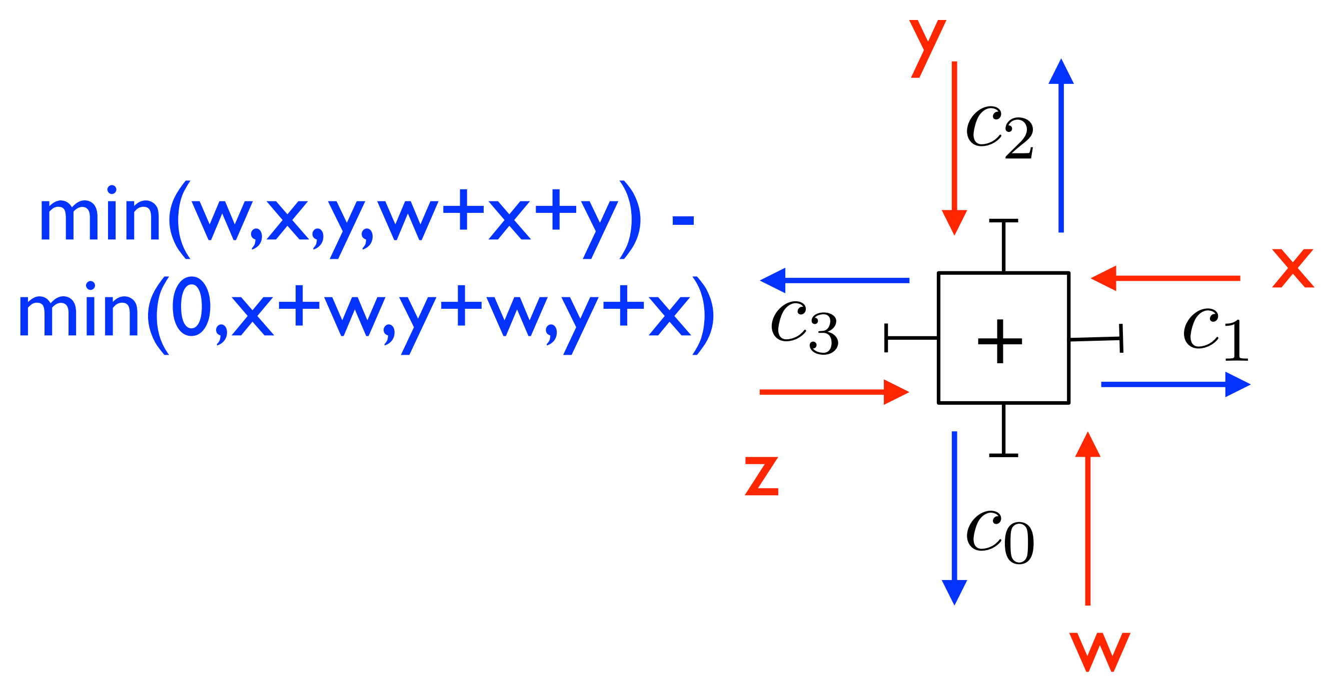
	config (3,2,1,0)	config metric
m=0:	0000	0
m=1:	0011	x+w
	0101	y+w
	0110	y+x
	1001	z+w
	1010	z+x
	1100	z+y
	1111	z+y+x+w

min(w,x,y,w+x+y) -
min(0,x+w,y+w,y+x)



for min*-sum, change
min to min*

Example: min-sum SPC SISO



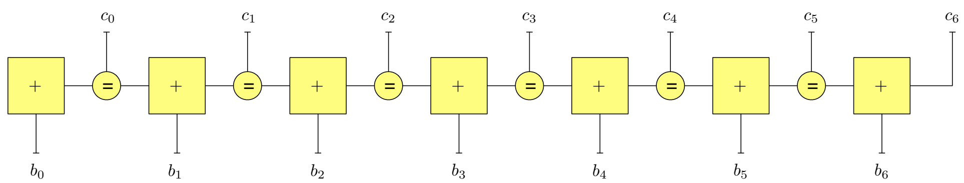
$$\min(w,x,y,w+x+y) - \min(0,x+w,y+w,y+x) = [\min(|w|,|x|,|y|)]\,\mathrm{sgn}(w)\mathrm{sgn}(x)\mathrm{sgn}(y)$$

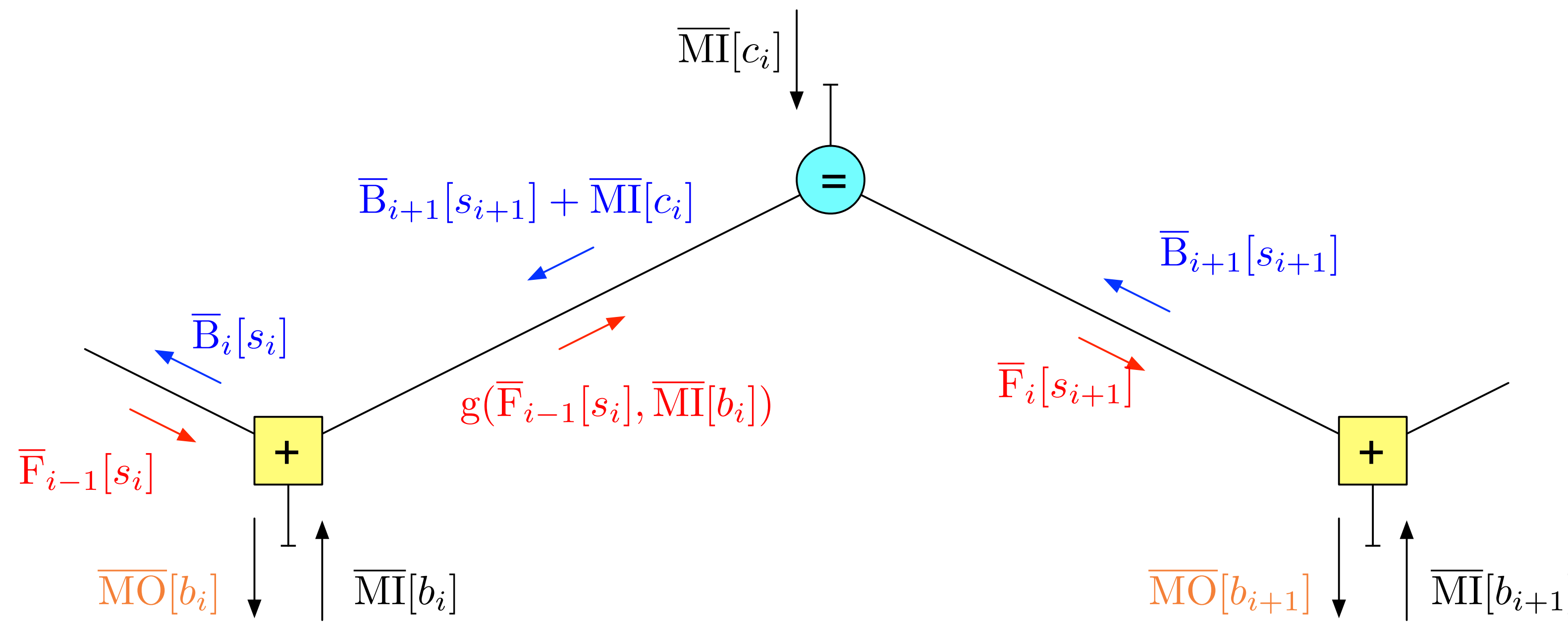
This is valid for min-sum only (cannot change mins to min*)

(example of a non-semi-ring property/algorithm)

"min-mag/sign-product" shortcut for SPC min-sum SISO

Example: Accumulator SISO





$$g(x,y) = \min(x,y) - \min(0, x+y)$$

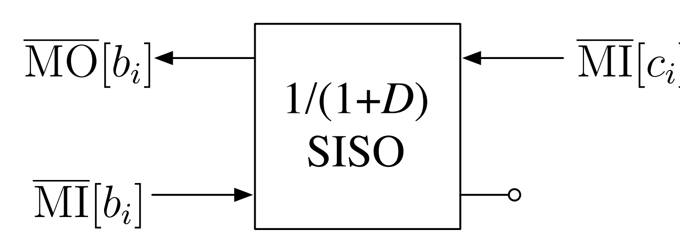
$$= \min(|x|, |y|)\mathrm{sgn}(x)\mathrm{sgn}(y)$$

$$g^*(x,y) = \min{}^*(x,y) - \min{}^*(0, x+y)$$

Forward Recursion: $\overline{\mathrm{F}}_i[s_{i+1}] = \overline{\mathrm{MI}}[c_i] + \mathrm{g}(\overline{\mathrm{F}}_{i-1}[s_i], \overline{\mathrm{MI}}[b_i])$

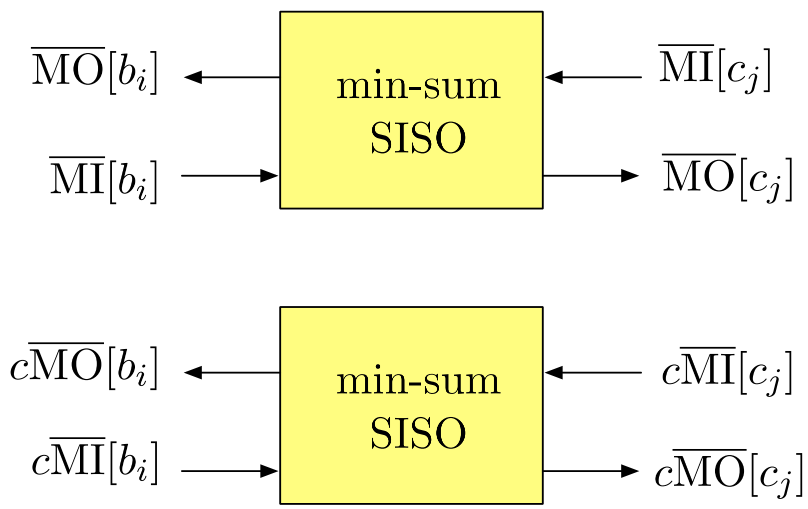
Backward Recursion: $\overline{\mathrm{B}}_i[s_i] = \mathrm{g}(\overline{\mathrm{B}}_{i+1}[s_{i+1}] + \overline{\mathrm{MI}}[c_i], \overline{\mathrm{MI}}[b_i])$

Completion on input: $\overline{\mathrm{MO}}[b_i] = \mathrm{g}(\overline{\mathrm{B}}_{i+1}[s_{i+1}] + \overline{\mathrm{MI}}[c_i], \overline{\mathrm{F}}_{i-1}[s_i])$

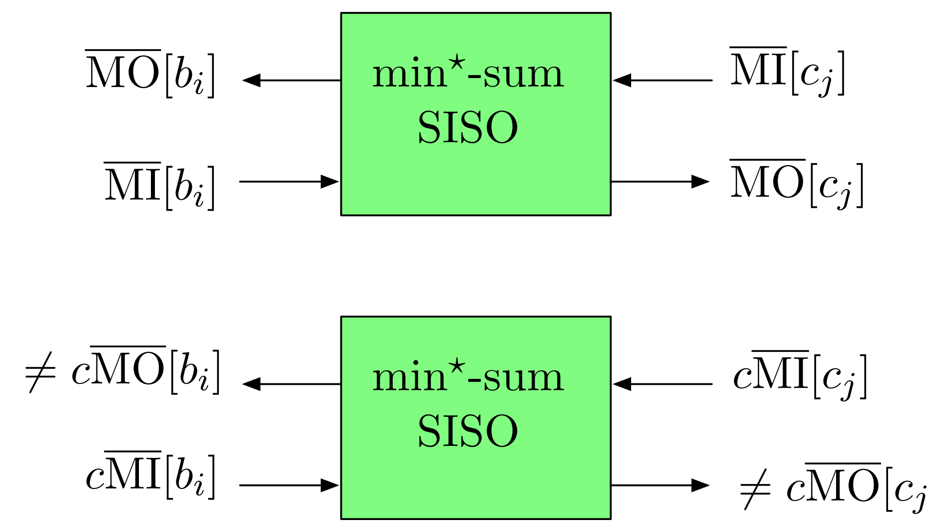


Special case of the Forward-Backward Algorithm

min-sum vs min*-sum



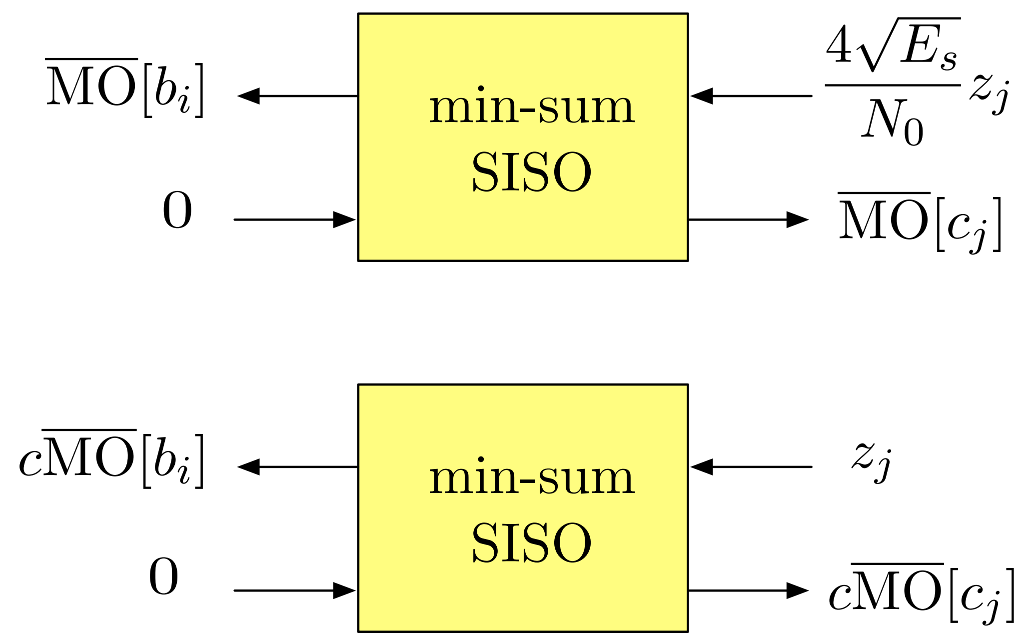
$$\min(cx, cy) = c\min(x, y) \quad (c > 0)$$



$$\min{}^\star(cx, cy) \neq c\min{}^\star(x, y)$$

This is a non-semi-ring property that holds for min-sum

min-sum vs min*-sum



min-sum processing does not require knowledge of Es or No when the inputs are iid uniform

Viterbi Algorithm & FBA

Model: FSM in memoryless noise (e.g., AWGN)

$$z_i(u) = x_i(b_i, s_i) + w_i(u)$$

Sequence/Configuration APP — recursive computation

$$f(\mathbf{z}_0^{I-1}|\mathbf{b}_0^{I-1}, s_0)p(\mathbf{b}_0^{I-1}, s_0) = p(s_0)\prod_{i=0}^{I-1} f(z_i|b_i, s_i)p(b_i)$$

(State) Transition Metrics

$$\mathrm{M}[\mathbf{t}_0^{I-1}] = -\ln[p(s_0)] + \sum_{i=0}^{I-1} \mathrm{M}_i[t_i]$$

$$t_i = (b_i, s_i)$$

Viterbi Algorithm & FBA

$$
\begin{aligned}
f(\mathbf{z}_0^{I-1}|\mathbf{b}_0^{I-1}, s_0) &= f(z_{I-1}|\mathbf{z}_0^{I-2}, \mathbf{b}_0^{I-1}, s_0) f(\mathbf{z}_0^{I-2}|\mathbf{b}_0^{I-1}, s_0) \\
&= f(z_{I-1}|\mathbf{b}_0^{I-1}, s_0) f(\mathbf{z}_0^{I-2}|\mathbf{b}_0^{I-2}, s_0) \\
&= f(z_{I-1}|b_i, s_i) f(\mathbf{z}_0^{I-2}|\mathbf{b}_0^{I-2}, s_0) \\
&= \prod_{i=0}^{I-1} f(z_i|b_i, s_i)
\end{aligned}
$$

In the tail this becomes: $p(b_i|s_i)$

$$
\begin{aligned}
p(\mathbf{b}_0^{I-1}, s_0) &= p(b_{I-1}|\mathbf{b}_0^{I-2}, s_0) p(\mathbf{b}_0^{I-2}, s_0) \\
&= p(b_{I-1}) p(\mathbf{b}_0^{I-2}, s_0) \\
&= p(s_0) \prod_{i=0}^{I-1} p(b_i)
\end{aligned}
$$

$$\mathrm{M}_i[t_i] = \mathrm{MI}[x_i(t_i)] + \mathrm{MI}[b_i(t_i)]$$

$$\mathrm{MI}[x_i(t_i)] = -\ln(f(z_i|x_i(t_i)))$$

$$\mathrm{MI}[b_i(t_i)] = -\ln[p(b_i)]$$

Viterbi Algorithm

Forward State Metric Recursion

$$\begin{aligned}
\mathrm{MSM}_0^i[s_{i+1}] &= \min_{\mathbf{t}_0^i : s_{i+1}} \left[\sum_{j=0}^{i} \mathrm{M}_j[t_j] \right] \\
&= \min_{\mathbf{t}_0^i : s_{i+1}} \left[\mathrm{M}_i[t_i] + \sum_{j=0}^{i-1} \mathrm{M}_j[t_j] \right] \\
&= \min_{t_i : s_{i+1}} \left[\mathrm{M}_i[t_i] + \min_{\mathbf{t}_0^{i-1} : s_{i+1}} \sum_{j=0}^{i-1} \mathrm{M}_j[t_j] \right] \\
&= \min_{t_i : s_{i+1}} \left(\mathrm{M}_i[t_i] + \mathrm{MSM}_0^{i-1}[s_i] \right)
\end{aligned}$$

$$\boxed{\mathrm{F}_i[s_{i+1}] = \min_{t_i : s_{i+1}} \left(\mathrm{M}_i[t_i] + \mathrm{F}_{i-1}[s_i] \right)}$$

Viterbi Algorithm

Forward State Metric Recursion

+

Survivor Path Storage (non-semi-ring)

+

Survivor Traceback and Decode

Forward-Backward Algorithm

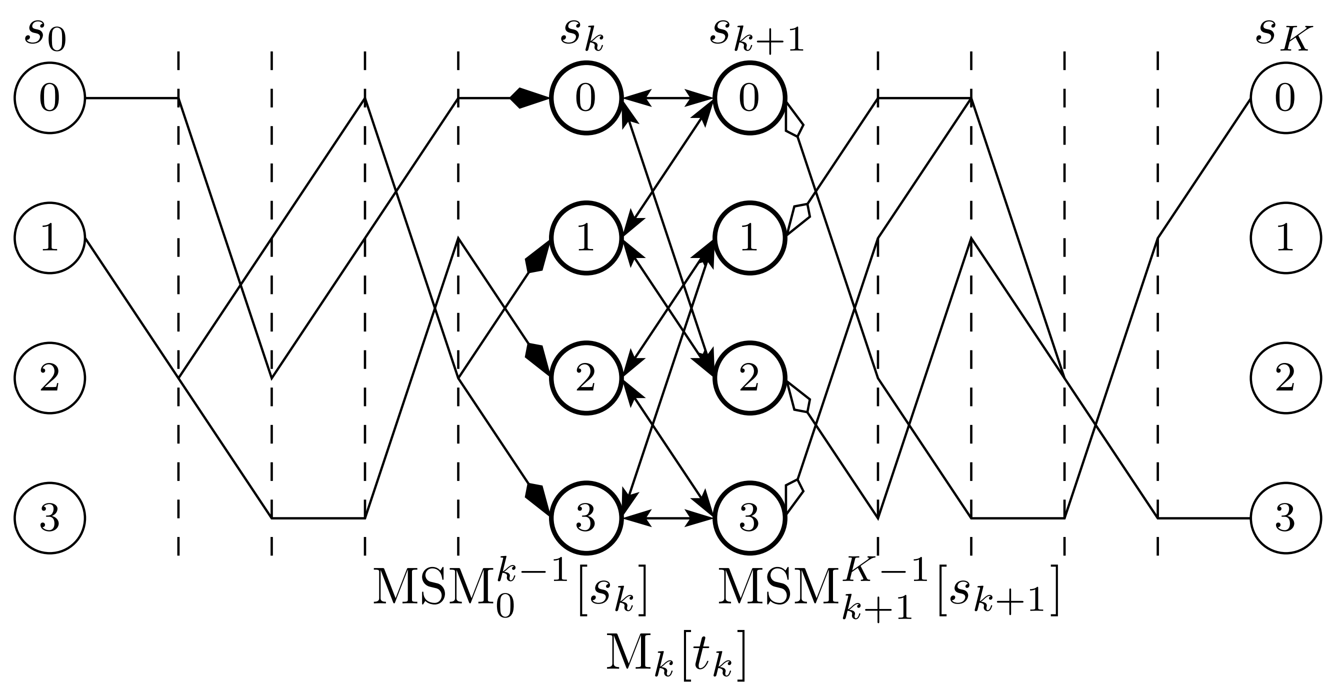


Figure 1.13. The MSM for a given transition may be computed by summing the transition metric and the forward and backward state metrics.

$$\mathrm{MSM}_0^{K-1}[t_k] = \min_{\mathbf{t}_0^{K-1}:t_k} \sum_{i=0}^{K-1} \mathrm{M}_i[t_i] \tag{1.66a}$$

$$= \min_{\mathbf{t}_0^{K-1}:t_k} \left[\sum_{i=0}^{k-1} \mathrm{M}_i[t_i] + \mathrm{M}_k[t_k] + \sum_{i=k+1}^{K-1} \mathrm{M}_i[t_i] \right] \tag{1.66b}$$

Forward-Backward Algorithm

$$\mathrm{M}_i[t_i] = \mathrm{MI}[c_i(t_i)] + \mathrm{MI}[b_i(t_i)] \qquad i = 0, 1, \ldots I-1 \qquad \text{Metric Computation}$$

$$\mathrm{F}_i[s_{i+1}] = \min_{t_i : s_{i+1}} \left(\mathrm{F}_{i-1}[s_i] + \mathrm{M}_i[t_i]\right) \qquad i = 0, \ldots I-2 \qquad \text{Forward Recursion}$$

$$\mathrm{B}_i[s_i] = \min_{t_i : s_i} \left(\mathrm{M}_i[t_i] + \mathrm{B}_{i+1}[s_{i+1}]\right) \qquad i = I-2, I-3, \ldots 1 \qquad \text{Backward Recursion}$$

$$\begin{aligned} \overline{\mathrm{MO}}[b_i] = &\min_{t_i : b_i = 1} \left(\mathrm{F}_{i-1}[s_i] + \mathrm{M}_i[t_i] + \mathrm{B}_{i+1}[s_{i+1}]\right) \\ &- \min_{t_i : b_i = 0} \left(\mathrm{F}_{i-1}[s_i] + \mathrm{M}_i[t_i] + \mathrm{B}_{i+1}[s_{i+1}]\right) - \overline{\mathrm{MI}}[b_i] \qquad i = 0, \ldots I-1 \qquad \text{Completion} \end{aligned}$$

$$\begin{aligned} \overline{\mathrm{MO}}[c_i] = &\min_{t_i : c_i = 1} \left(\mathrm{F}_{i-1}[s_i] + \mathrm{M}_i[t_i] + \mathrm{B}_{i+1}[s_{i+1}]\right) \\ &- \min_{t_i : c_i = 0} \left(\mathrm{F}_{i-1}[s_i] + \mathrm{M}_i[t_i] + \mathrm{B}_{i+1}[s_{i+1}]\right) - \overline{\mathrm{MI}}[b_i] \qquad i = 0, \ldots I-1 \qquad \text{Completion} \end{aligned}$$

Forward-Backward Algorithm

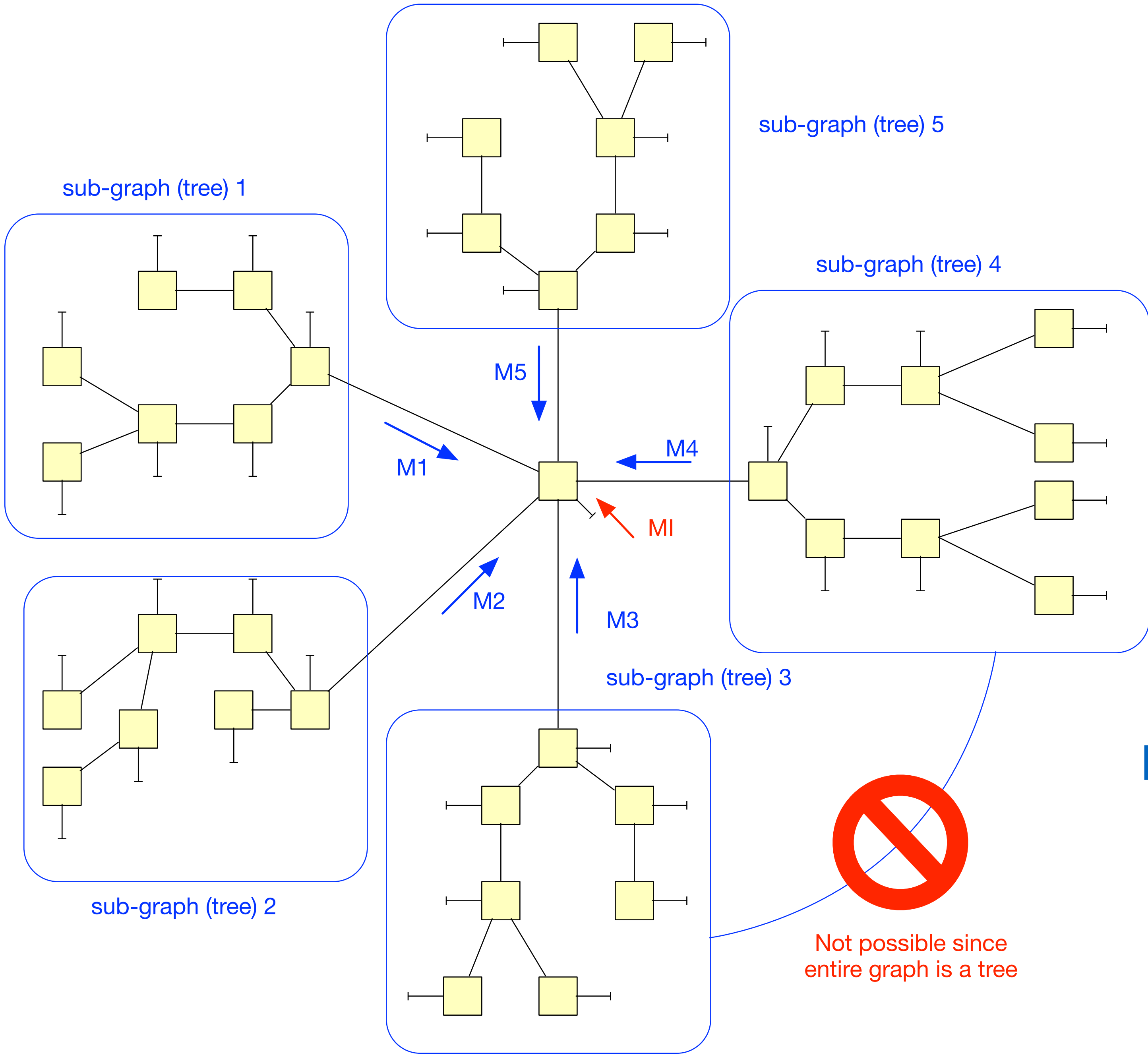
$$
\begin{aligned}
p(\mathbf{z}_0^{K-1}, t_k) &= p(\mathbf{z}_{k+1}^{K-1}|t_k)p(\mathbf{z}_0^k, t_k) && (1.69a)\\
&= p(\mathbf{z}_{k+1}^{K-1}|t_k)p(z_k, a_k|s_k)p(\mathbf{z}_0^{k-1}, s_k) && (1.69b)\\
&= [p(\mathbf{z}_0^{k-1}, s_k)][p(z_k|x_k(t_k))p(a_k)][p(\mathbf{z}_{k+1}^{K-1}|s_{k+1})] && (1.69c)
\end{aligned}
$$

$$
p(\mathbf{z}_0^k, s_{k+1}) = \sum_{t_k:s_{k+1}} \left[p(\mathbf{z}_0^{k-1}, s_k)p(z_k|x_k(t_k))p(a_k) \right] \qquad (1.70a)
$$

$$
p(\mathbf{z}_k^{K-1}|s_k) = \sum_{t_k:s_k} \left[p(\mathbf{z}_{k+1}^{K-1}|s_{k+1})p(z_k|x_k(t_k))p(a_k) \right] \qquad (1.70b)
$$

Sum-product version via probability manipulations

Why Optimal For Trees?

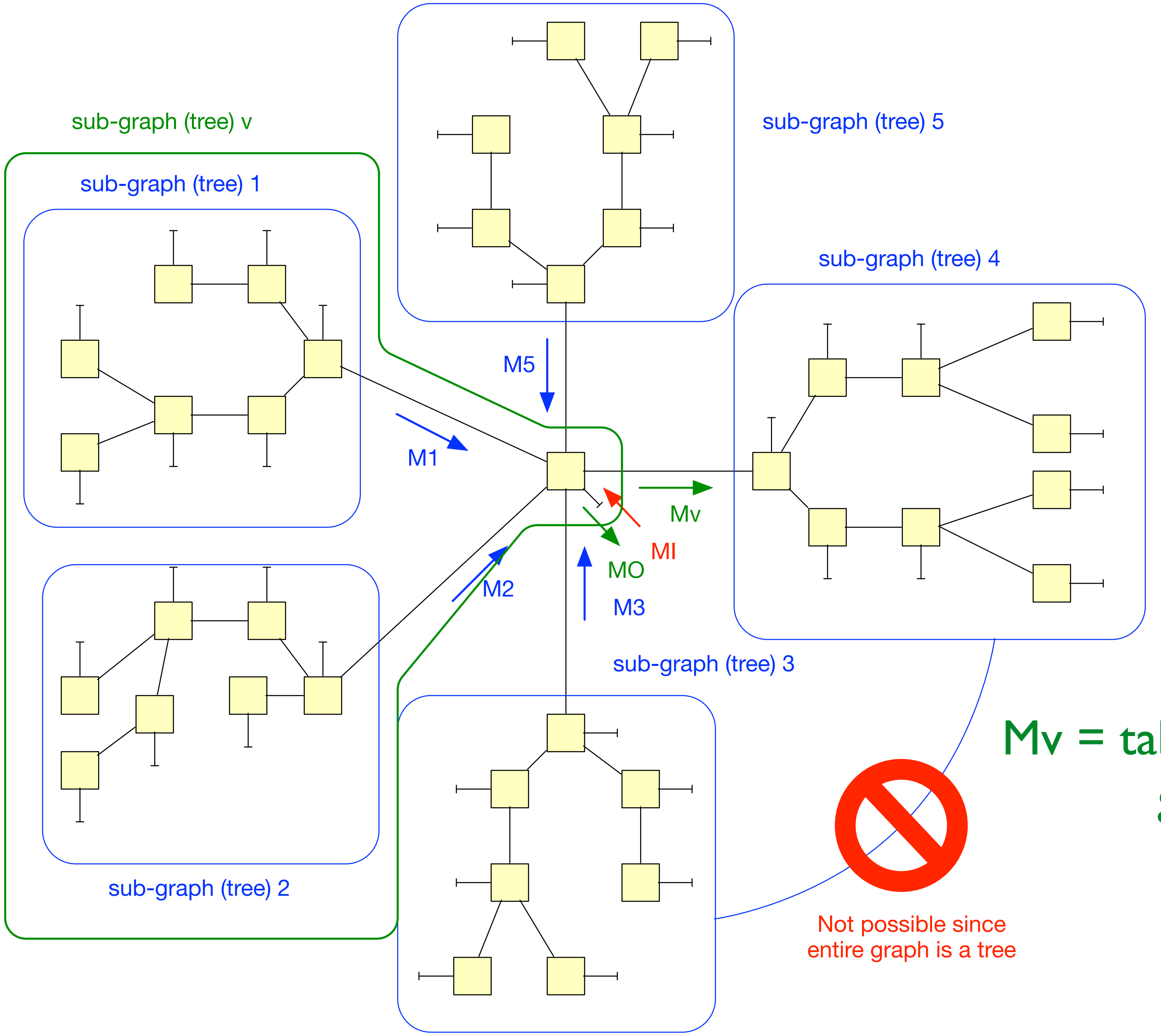


Mn = table of MSMs for that edge variable

= metric of best configuration of tree n, given that conditional value of the edge variable

Use Viterbi Algorithm to find minimum weight simple error pattern

Why Optimal For Trees?



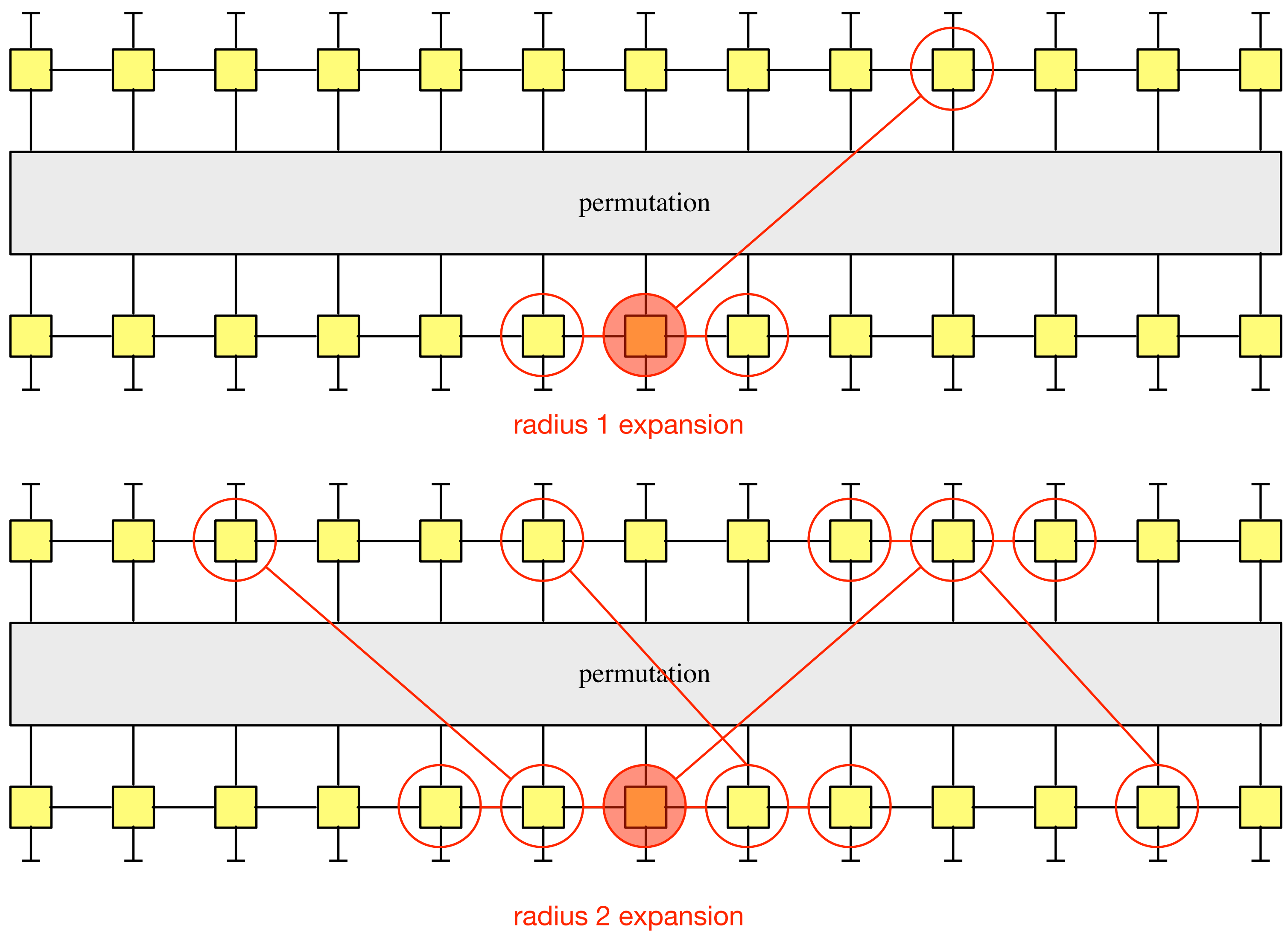
apply this reasoning to "grow" trees that have all of the required information for global optimality

direct generalization of the argument used for Viterbi Algorithm — e.g., partition problem into two problems (east and west)

Mv = table of MSMs for best configuration of tree v, given conditional edge variable value

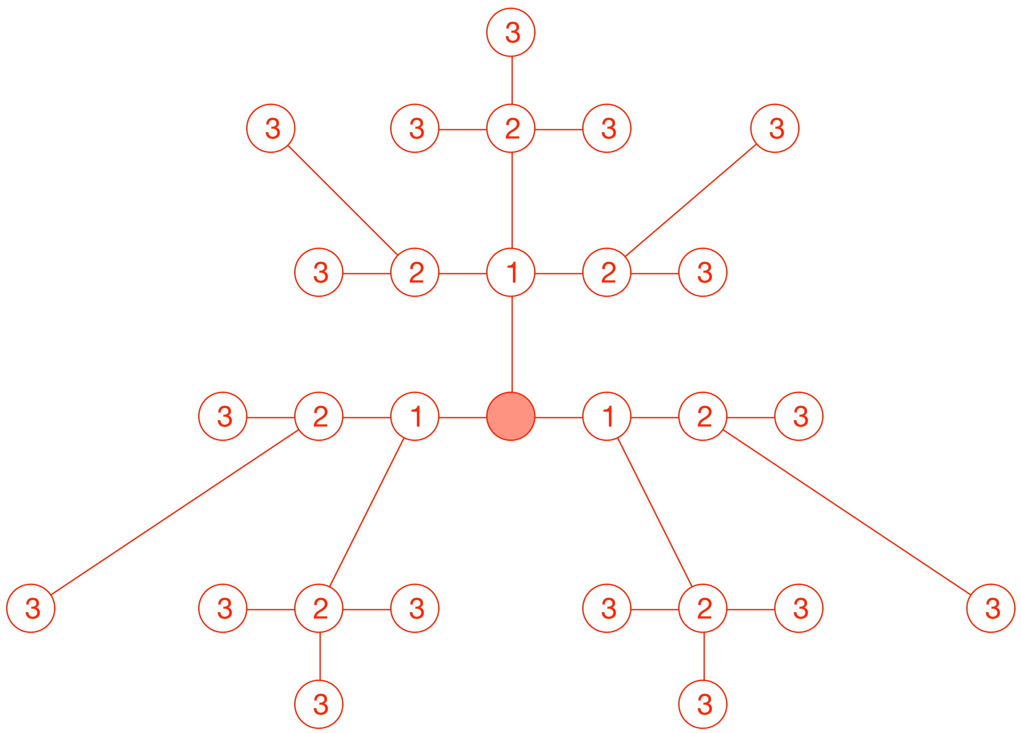
MO = globally optimal extrinsic soft information

Why Good Heuristic for Cyclic Graphs?



For an expansion by looking out r steps from a given node

Why Good Heuristic for Cyclic Graphs?



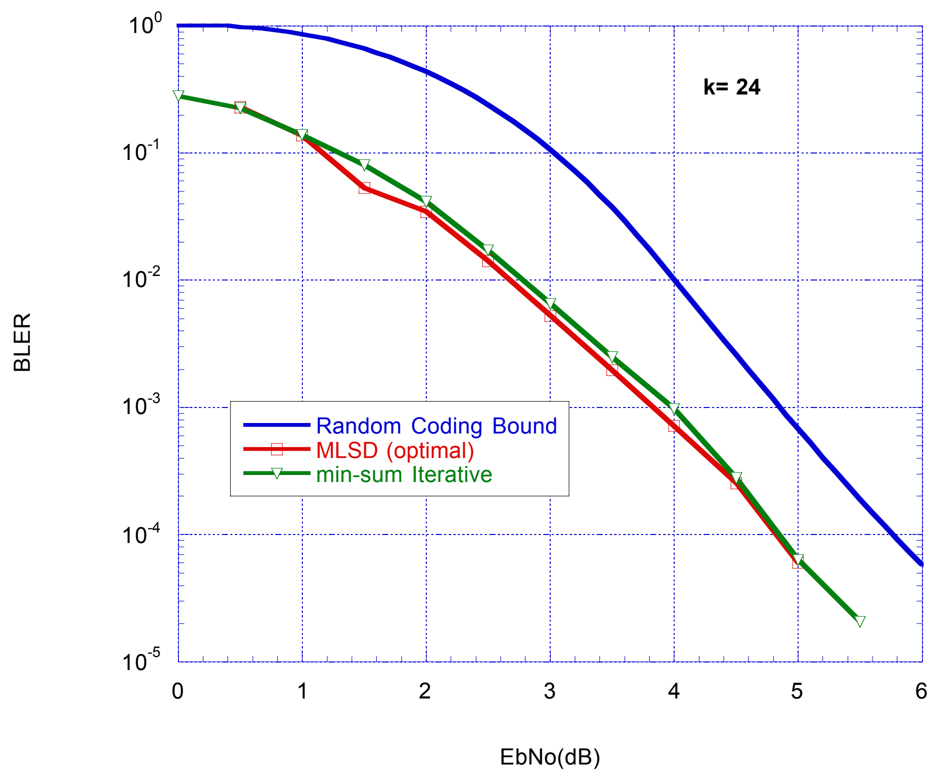
r = 3 expansion

Note: if radius r expansion is cycle free, then after r flooding activations, the central node can perform optimal decision based on all incoming messages within radius r

Conclusion: If the minimal cycle length is longer than the "survivor merging" radius of the graph, then standard message-passing should approximate optimal inference

In practice: Long cycles and random cycle structure is sought for near-optimal performance — intuition, do not want all (weak) echoes coming back to source at once

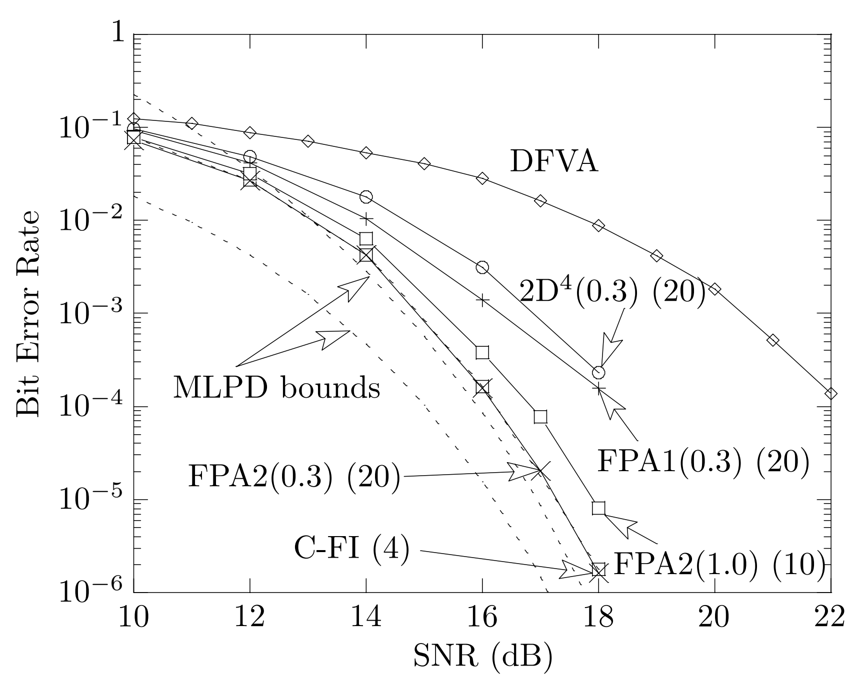
Example of Heuristic vs. Optimal



Input block size 24, 4 state PCCC (Turbo Code)

MLSD (optimal) decoder adopted from d_min paper:

R. Garello, F. Chiaraluce, P. Pierleoni, M. Scaloni, and S. Benedetto. On error floor and free distance of turbo code. In *Proc. International Conf. Communications*, pages 45–49, Helsinki, Finland, jun. 2001.



2-dimensional ISI problem - MLPD bounds are similar to our error probability bounds (Ch. 5 of my book)

Coding Topics

- Coding channel models
- Basics of code constructions
- Decoding rules — HIHO, SIHO, SISO
- Classical coding
- Modern Coding
- Performance limits
  - Capacity and finite block-size bounds)
  - Bounds for specific codes

Performance Limits

- Performance limits (information theory based bounds)
  - Infinite block length, zero error probability
    - Channel capacity
      - Modulation-unconstrained AWGN Channel
    - Symmetric Information Rate (SIR)
      - Modulation-constrained AWGN Channel
  - Finite block size, finite error probability
    - Sphere packing bound (SPB)
    - Random Coding Bound (RCB)
    - Pragmatic guideline

Channel Capacity

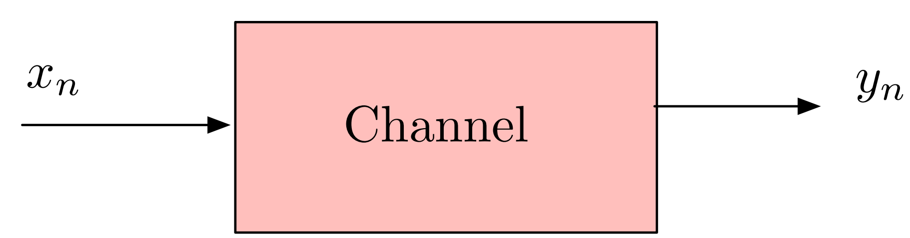
Mutual Information

$$\mathsf{I}(x(u);y(u)) = \sum_y \sum_x p_{x(u),y(u)}(x,y) \left[\log_2 \left(\frac{p_{x(u),y(u)}(x,y)}{p_{x(u)}(x) p_{y(u)}(y)} \right) \right]$$

$$\sum_y \sum_x p_{x(u),y(u)}(x,y) \left[\log_2 \left(\frac{1}{p_{y(u)}(y)} \right) - \log_2 \left(\frac{1}{p_{y(u)|x(u)}(y|x)} \right) \right]$$

Channel Capacity for Memoryless Channel

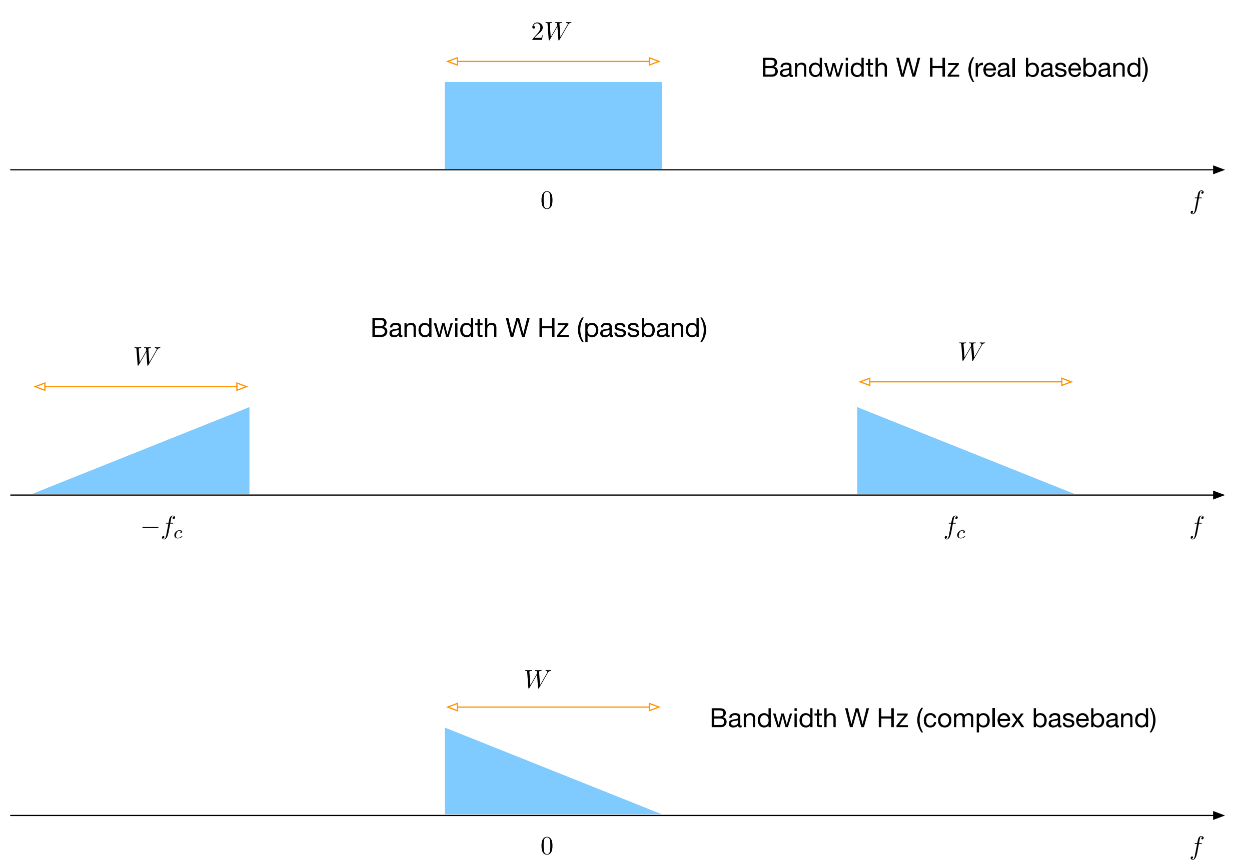
$$\max_{p_{x(u)}(\cdot)} \mathsf{I}(x(u);y(u))$$



$$P(\mathbf{y}|\mathbf{x}) = \prod_n P(y_n|x_n)$$

AWGN Channel Capacity

measuring bandwidth:



With this, we can get ~ 2WT dimensions in W Hz of bandwidth and T secs

$$\mathbf{z}_i(u) = \mathbf{x}_i(u) + \mathbf{w}_i(u) \qquad (D \times 1)$$

$$D = 2WT$$

$$\mathbf{w}_i(u) \sim \mathcal{N}_D(\cdot; 0; N_0/2\mathbf{I})$$

$$\mathbb{E}\left\{\|\mathbf{x}(u)\|^2\right\} \leq PT$$

memoryless channel

AWGN Channel Capacity

$$C_{\mathrm{AWGN}} = (2WT)\frac{1}{2}\log_2\left(1 + \frac{P}{N_0 W}\right) \quad \text{bits per } D \times 1 \text{ channel use}$$

$$= W\log_2\left(1 + \frac{P}{N_0 W}\right) \quad \text{bits per second}$$

Achieved when x is Gaussian!

$$\frac{C_{\mathrm{AWGN}}}{W} = \log_2\left(1 + \frac{P}{N_0 W}\right) \quad \text{bps/Hz}$$

$$\frac{C_{\mathrm{AWGN}}}{W} = \log_2\left(1 + \frac{P}{N_0 W}\right) \quad \text{bps/Hz}$$

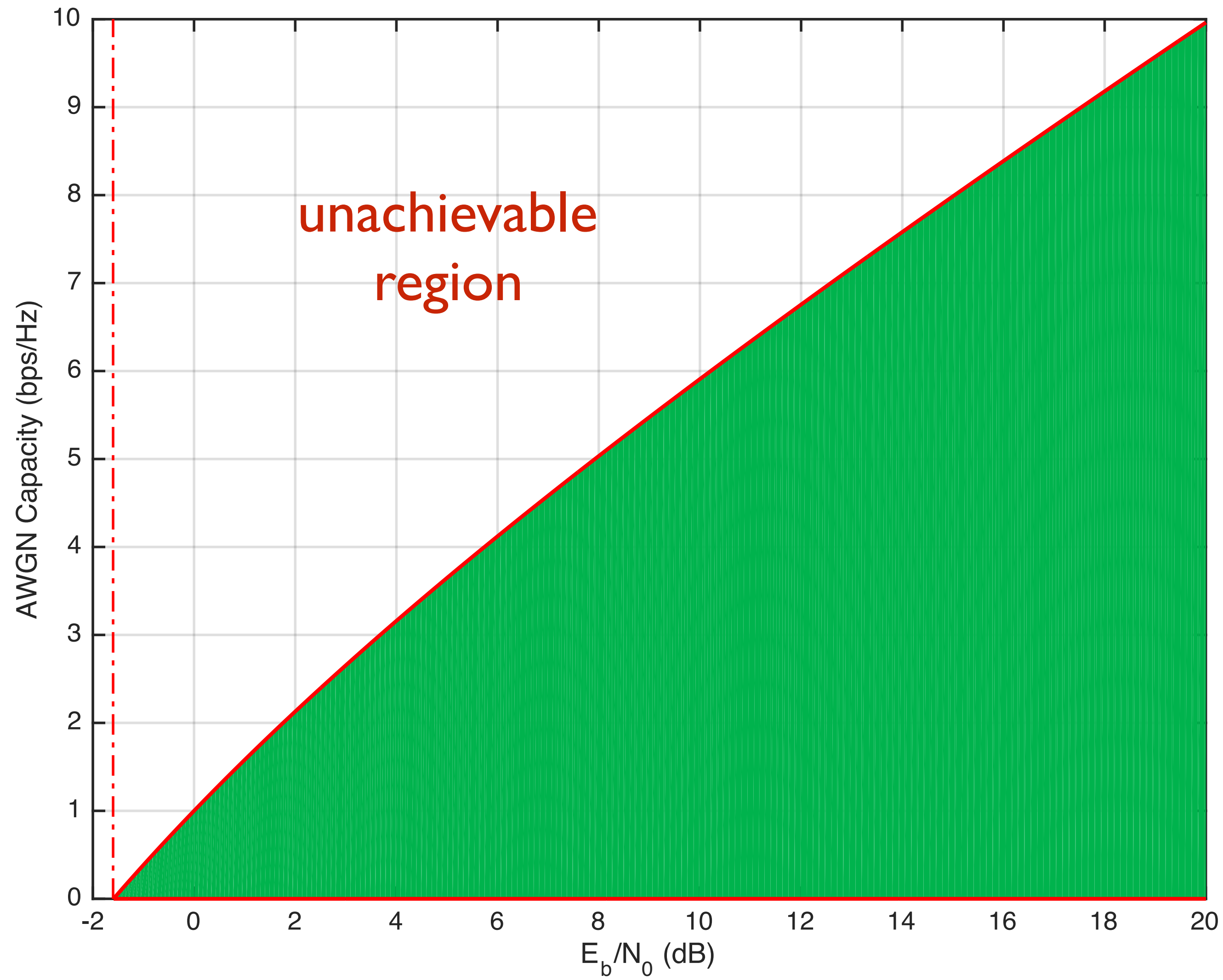
$$= \log_2\left(1 + \frac{E_b R_b}{N_0 W}\right)$$

Operating at capacity (Rb = C):

$$\boxed{\frac{C_{\mathrm{AWGN}}}{W} = \log_2\left(1 + \left[\frac{E_b}{N_0}\right]_{\min} \frac{C_{\mathrm{AWGN}}}{W}\right) \quad \text{bps/Hz}}$$

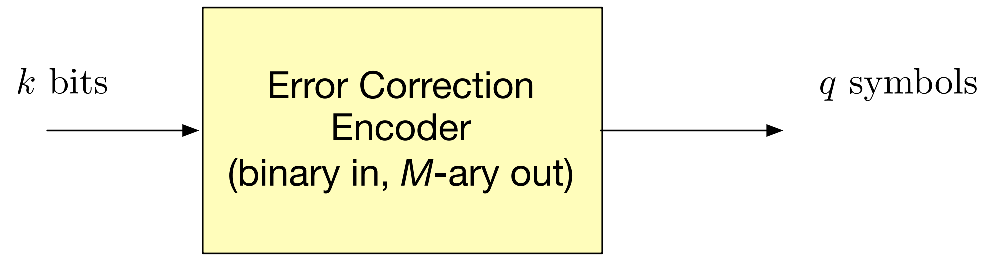
AWGN Capacity

$$\boxed{\left[\frac{E_b}{N_0}\right]_{\min} = \frac{2^{\eta_{\mathrm{bps/Hz}}} - 1}{\eta_{\mathrm{bps/Hz}}}}$$

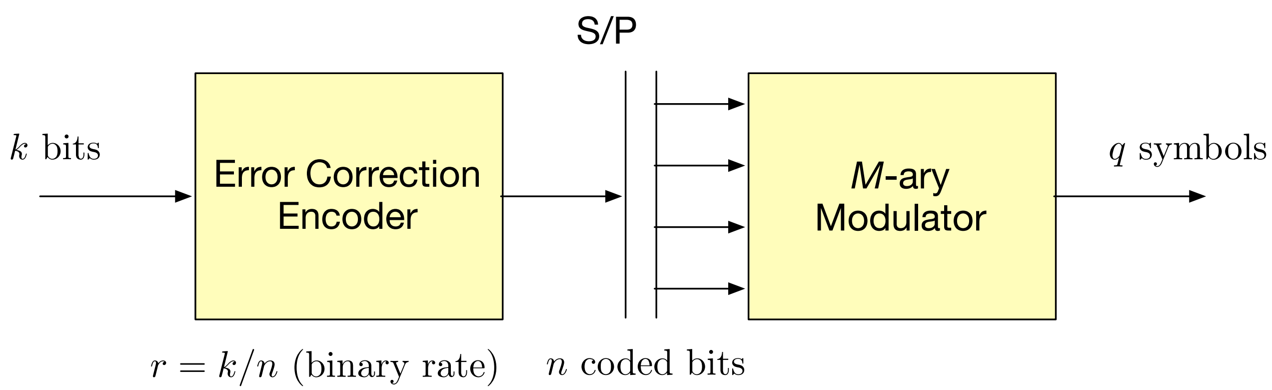


Eb/No = -1.6 dB is the smallest value of Eb/No for reliable communications on the AWGN channel

Computing Rates for Coded-Modulation



general case can be thought of at having two stages



$$q = \frac{n}{\log_2(M)}$$

$$\eta_{\mathrm{b/sym}} = k/q = r\log_2(M)$$

$$\eta_{\mathrm{b/2d}} = \frac{2}{D}\eta_{\mathrm{b/sym}} = \frac{2k}{Dq}$$

usually assumed in papers/textbooks

$$\boxed{\eta_{\mathrm{bps/Hz}} = \eta_{\mathrm{b/2d}} \qquad \text{(ideal)}}$$

$$\eta_{\mathrm{bps/Hz}} = \frac{\eta_{\mathrm{b/2d}}}{1+\beta} \qquad \text{(RRC)}$$

Modulation Constrained AWGN Capacity

Signal Model:

$$\mathbf{z}(u) = \sqrt{\frac{E_s}{N_0}}\mathbf{x}(u) + \mathbf{w}(u) \qquad (D \times 1) \tag{1}$$

$$\mathbb{E}\left\{\|\mathbf{x}(u)\|^2\right\} = \sum_{m=0}^{M-1} p_m \|\mathbf{s}_m\|^2 = 1 \tag{2}$$

$$\mathbb{E}\left\{\mathbf{w}(u)\mathbf{w}^{\mathrm{t}}(u)\right\} = \frac{1}{2}\mathbf{I} \tag{3}$$

$$p(\mathbf{z}|\mathbf{s}_m) = \frac{1}{\pi^{D/2}} \exp\left(-\left\|\mathbf{z} - \sqrt{\frac{E_s}{N_0}}\mathbf{s}_m\right\|^2\right) \tag{4}$$

Normalized so noise variance is 1 per real dimension

Modulation Constrained AWGN Capacity/SIR

Symmetric Information Rate (SIR)

$$
\begin{aligned}
\mathrm{I}(\mathbf{z}(u);\mathbf{x}(u)) &= \sum_{m=0}^{M-1} p_m \int_{R^D} p(\mathbf{z}|\mathbf{s}_m)\log_2\left(\frac{p(\mathbf{z}|\mathbf{s}_m)}{p(\mathbf{z})}\right) d\mathbf{z} &\text{(9a)}\\
&= \sum_{m=0}^{M-1} p_m \int_{R^D} p(\mathbf{z}|\mathbf{s}_m)\log_2\left(\frac{p(\mathbf{z}|\mathbf{s}_m)}{\sum_{n=0}^{M-1} p(\mathbf{z}|\mathbf{s}_n)p_n}\right) d\mathbf{z} &\text{(9b)}
\end{aligned}
$$

Capacity:

$$C = \max_{\mathbf{p}} \mathrm{I}(\mathbf{z}(u);\mathbf{x}(u))$$

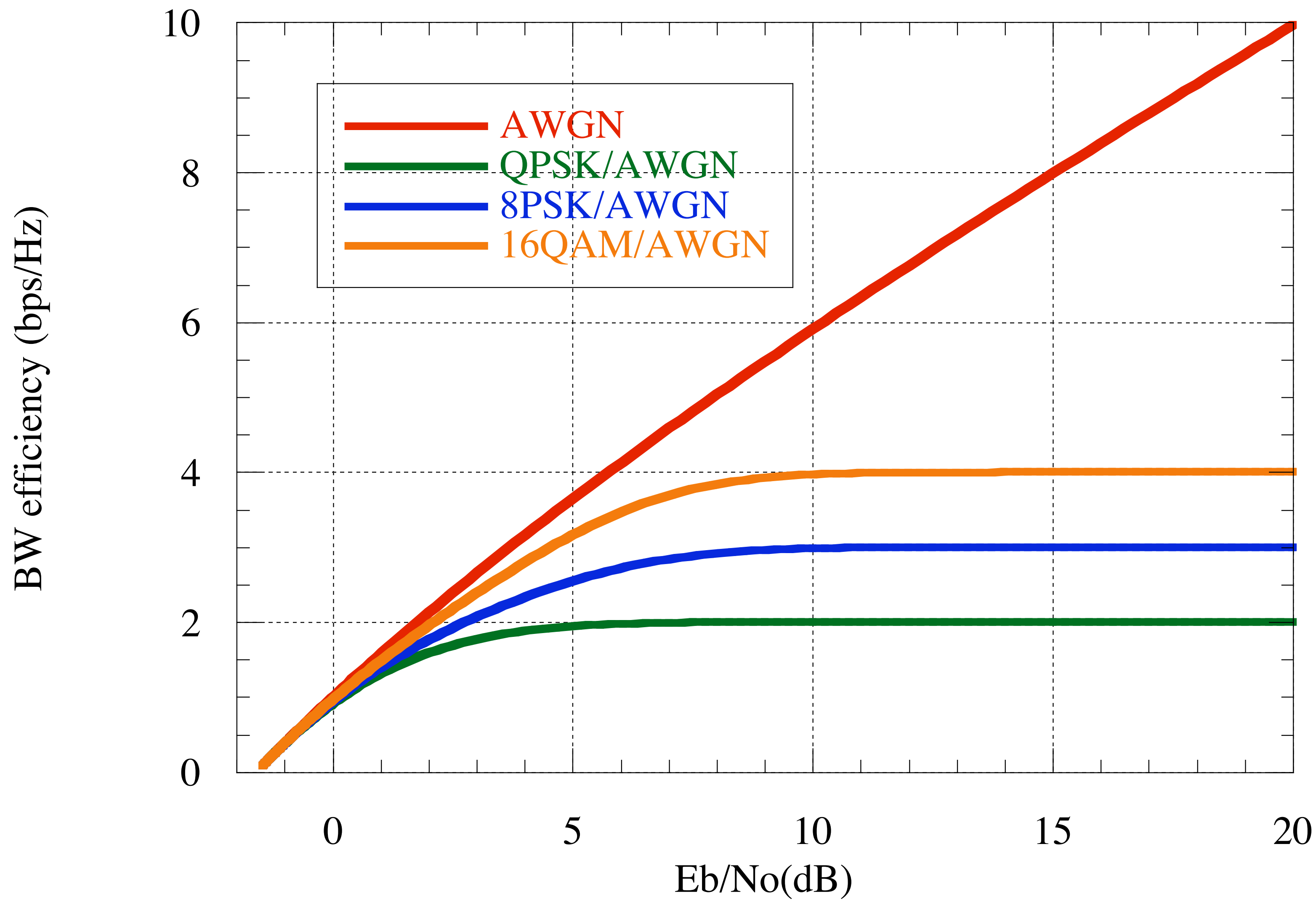
SIR <= Capacity

SIR is often used in place of Capacity for simplicity (not always clearly stated)

For PSKs, SIR=C, for QAMS, SIR is strictly less than capacity

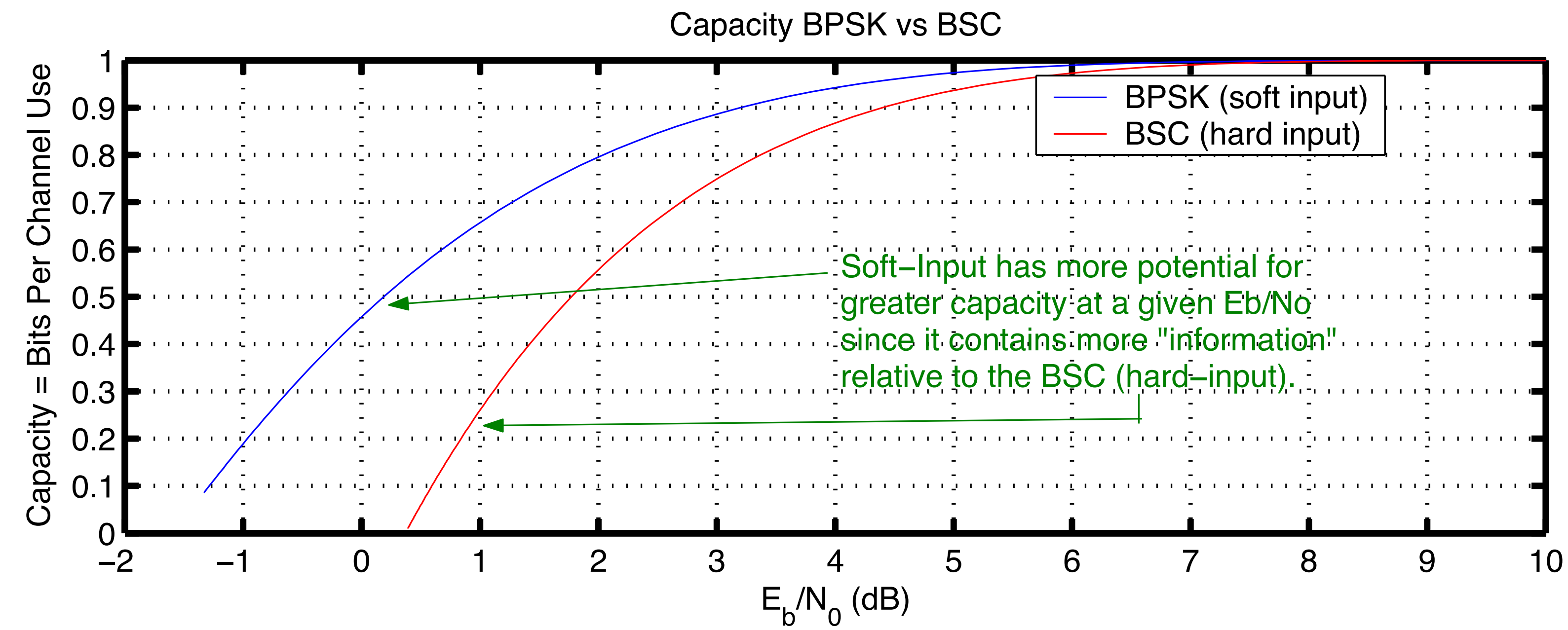
(difference is called “shaping gain”)

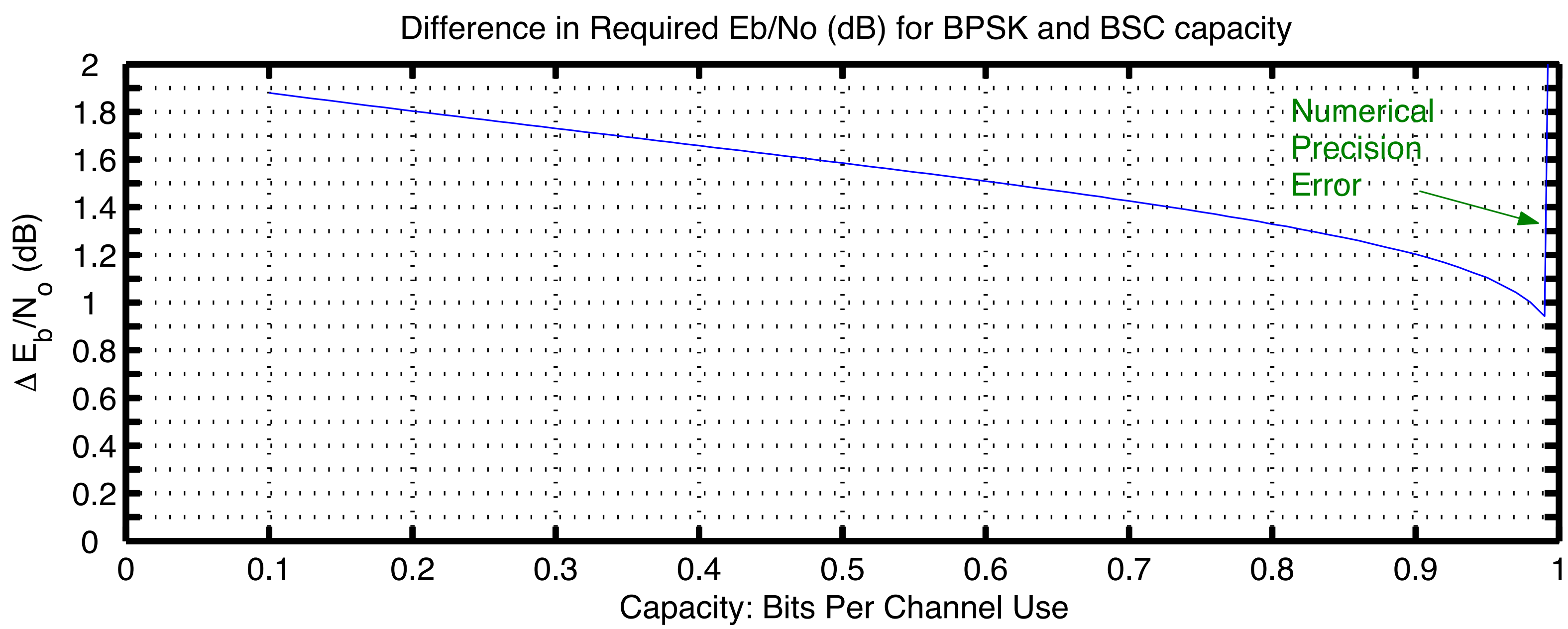
Modulation Constrained AWGN Capacity/SIR



SIR is computed via numerical integration

Modulation Constrained AWGN Capacity/SIR - example





Use information theory to predict soft-in vs hard-in coding gain (Problem 4.2)

Finite Block Size, Finite Error Probability Bounds

- **Sphere packing bound (SPB)**
  - Lower bound on P_cw for **any** code of a given rate and block size
- **Random Coding Bound (RCB)**
  - Upper bound on P_cw, averaged over all random codes
- **Common Features**
  - Both converge to capacity as block length goes to infinity
    - So they "sandwich" capacity
  - Both are challenging to evaluate numerically (SPB more so)
  - Both have optimizations over a-priori like capacity, so both have "symmetric" versions

Random Coding Bound

$$\bar{P}_{\text{cw}} \leq \exp\left(-qE_r(\eta_{\text{b/sym}})\right) \quad (17)$$

$$E_r(\eta_{\text{b/sym}}) = \max_{0\leq\rho\leq1} \max_{\mathbf{p}} \left[E_0(\rho, \mathbf{p}, \eta_{\text{b/sym}}) - \rho \ln(2)\eta_{\text{b/sym}}\right] \quad (18)$$

$$E_0(\rho, \mathbf{p}, \eta_{\text{b/sym}}) = \int_{R^D} \left[\sum_{m=0}^{M-1} p_m \left\{p(\mathbf{z}|\mathbf{s}_m)\right\}^{\frac{1}{1+\rho}}\right]^{1+\rho} d\mathbf{z} \quad (19)$$

Symmetric version uses p_m = 1/M

$$\bar{P}_{word} \leq e^{-k(E_b/N_0)} \left\{ \min_{0\leq\rho\leq1} 2^{\rho r+1} \int_0^\infty \frac{e^{\frac{-y^2}{2}}}{\sqrt{2\pi}} \cosh^{1+\rho}\left(\frac{y\sqrt{2r(E_b/N_0)}}{1+\rho}\right) dy \right\}^n$$

[3] R. Gallager, *Information Theory and Reliable Communication.* John Wiley & Sons, 1968.

Sphere Packing Bound

[8] S. Dolinar, D. Divsalar, and F. Pollara, "Code performance as a function of block size," tech. rep., JPL-TDA, May 1998. 42–133.

This report generates an approximation to the S-SPB for binary codes and BPSK.

I have found this generalizes to M-ary coded modulation

$$\left(\frac{E_b}{N_o}\right)_{\min} = \frac{2^{\eta_{\mathrm{bps/Hz}}} - 1}{\eta_{\mathrm{bps/Hz}}} \tag{35}$$

$$\Delta_{\mathrm{dB}} = \sqrt{\frac{20\eta_{\mathrm{b/2d}}\left(2^{\eta_{\mathrm{b/2d}}} + 1\right)\left[10\log_{10}(1/P_{\mathrm{CW}})\right]}{k\ln(10)\left(2^{\eta_{\mathrm{b/2d}}} - 1\right)}} \tag{36}$$

SPB approximation AWGN no modulation constraint

$$\left(\frac{E_b}{N_0}\right)_{\mathrm{min,SPB,\ (dB)}} \approx 10\log_{10}\left[\frac{2^{\eta_{\mathrm{b/2d}}} - 1}{\eta_{\mathrm{b/2d}}}\right] + \Delta_{\mathrm{dB}}$$

S-SPB Approximation for Modulation Constrained AWGN Channel

$$\left(\frac{E_b}{N_0}\right)_{\text{min,SIR-SPBA, (dB)}} \approx \left(\frac{E_b}{N_0}\right)_{\text{min,SIR, (dB)}} + \Delta_{\text{dB}} \tag{39}$$

$$\Delta_{\text{dB}} = \sqrt{\frac{20\eta_{\text{b/2d}}\left(2^{\eta_{\text{b/2d}}}+1\right)\left[10\log_{10}(1/P_{\text{CW}})\right]}{k\ln(10)\left(2^{\eta_{\text{b/2d}}}-1\right)}} \tag{36}$$

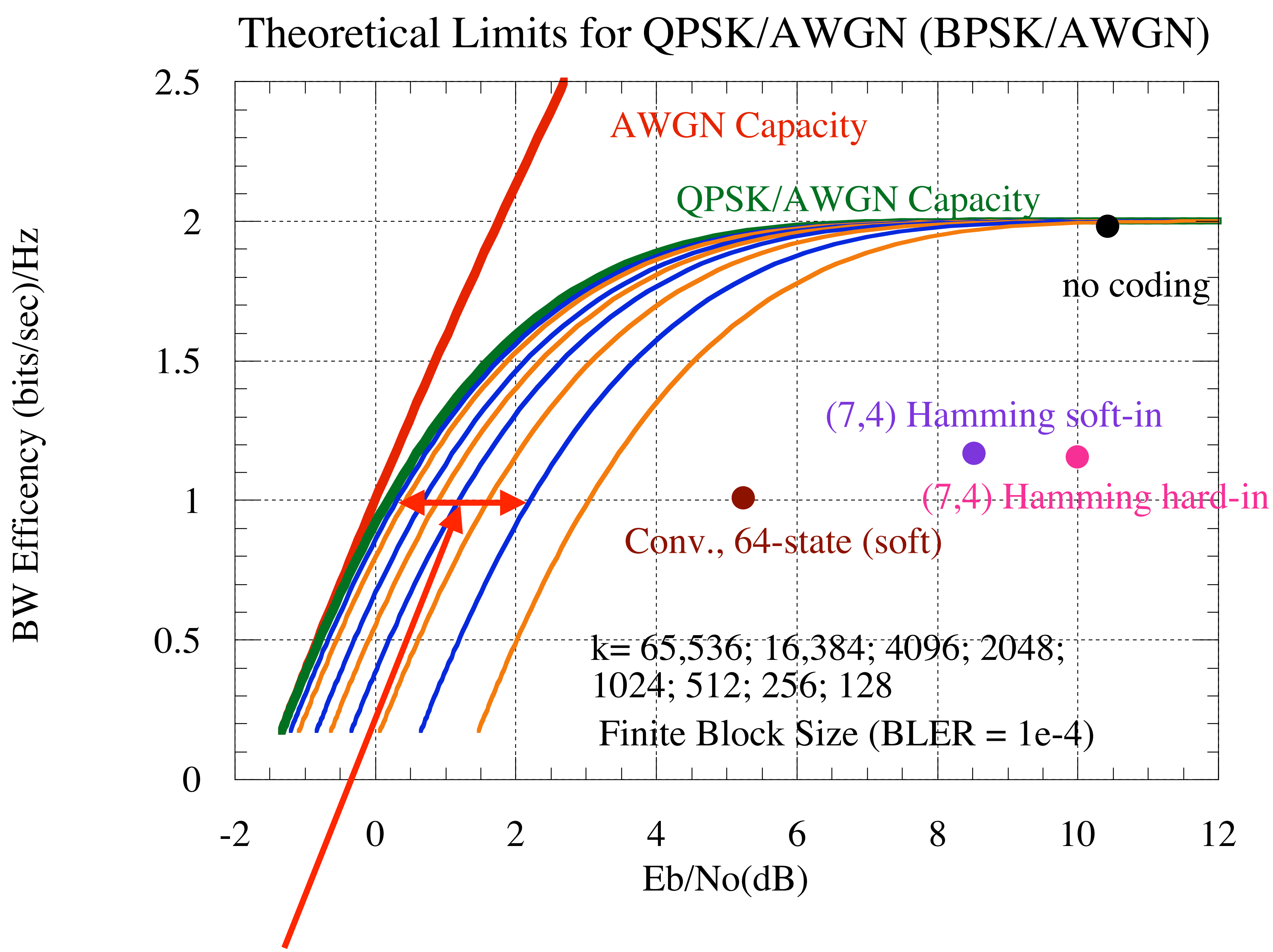
SPB approximation Modulation-Constrained AWGN Channel

$$\boxed{\left(\frac{E_b}{N_0}\right)_{\text{min,SIR-SPBA, (dB)}} \approx \left(\frac{E_b}{N_0}\right)_{\text{min,SIR, (dB)}} + \Delta_{\text{dB}}}$$

compute once via numerical integration

trivial computation

S-SPB Approximation for Modulation Constrained AWGN Channel



Δ_{dB} for BPSK and $k = 512, P_{CW} = 10^{-4}$

Note: Delta-dB is a weak function of eta

Pragmatic Guideline

- **S-SPB Approximation and S-RCB are very close to each other**
    - User k >~ 512, r <~ 8/9
    - Only need simple to compute S-SPB Approximation
- **Pragmatic Guideline**
    - Best modern code designs are about 0.5 dB from S-SPB Approximation
    - Hardware codecs should be within 1 dB of S-SPB Approximation

performance_limits_chugg.xls

S-RCB vs. S-SPB-Approximation

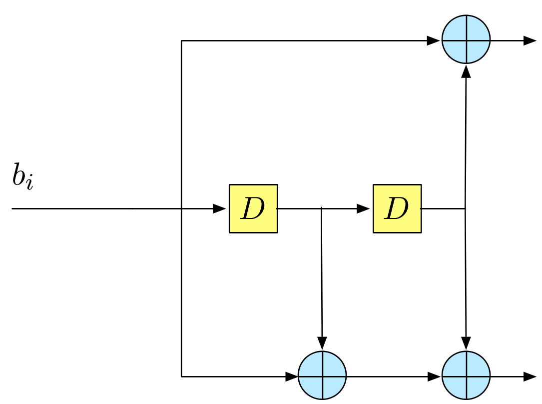
(add plot from limits.c)

These are very close (<~ 0.1 dB of Eb/No) for:
input block sizes

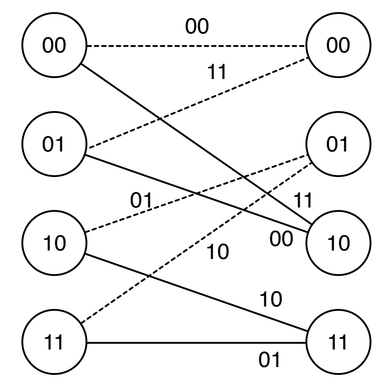
Performance Bounds for Convolutional Codes

covered on the PAD notes

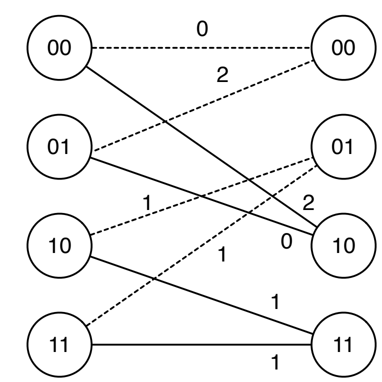
Example Free Distance Computation



toy 4-state code



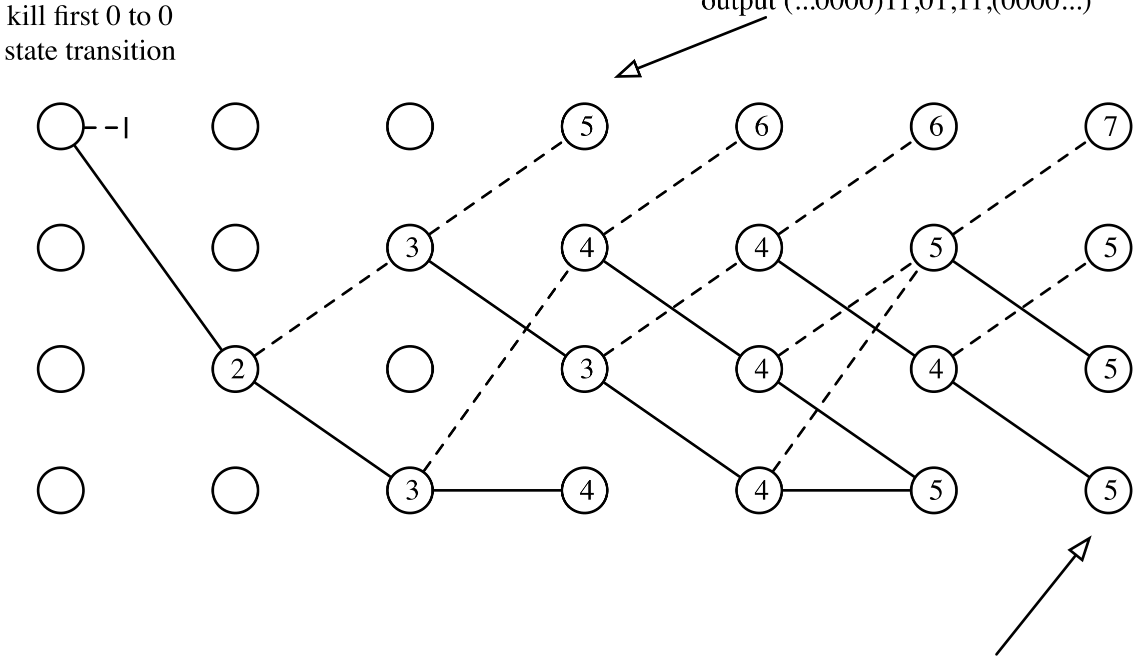
trellis



metrics = distance from zero path

Start in 0 state, kill first 0 to 0 state transition

d_free = 5:
input sequence (...00)100(00..),
output (...0000)11,01,11,(0000...)



Can terminate search because all survivors have weight at least 5 and a path that remerges with all 0 path with weight 5 has been found earlier

Use Viterbi Algorithm to find minimum weight simple error pattern

Uniform Interleaver Analysis (summary)

- Analyze union bound as block size tends toward infinity
  - Average over all possible interleaves (N!)
  - Determine trends in BER, BLER
  - Determine design rules

$$P_b \tilde{\leq} \sum_{d \geq d\mathrm{min}} K_d \mathrm{Q}\left(\sqrt{\frac{rdE_b}{2N_0}}\right)$$

$$K_d \sim C_d \left(\sum_{\alpha(d)} N^{\alpha(d)}\right)$$

$$P_b \sim N^{\alpha_{\max}}$$

$$P_{cw} \sim N^{\alpha_{\max}+1}$$

maximum exponent of N: $\alpha_{\max} = \max_d \alpha(d)$

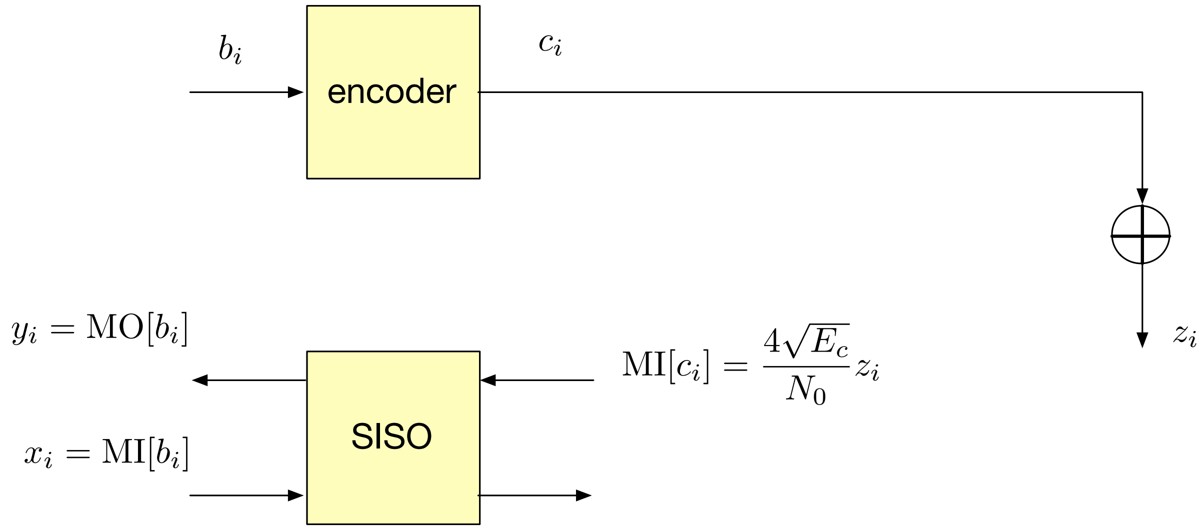
Uniform Interleaver Analysis (summary)

- PCCCs (w/ recursive encoders): $\alpha_{\max} = -1$
  - BER interleaver gain
  - No BLER interleaver gain
- SCCCs (w/ recursive inner code): $\alpha_{\max} = -\left\lfloor \frac{d_{\mathrm{o,min}} + 1}{2} \right\rfloor$
  - BER & BLER interleaver gain for do,min>=3

Some constructions naturally have better floor properties - eg, SCCCs have lower floors than PCCCs

Threshold Optimization & Irregular Designs

Idea: treat each SISO node as an amplifier of soft-information quality



For various values of E_c/N_0, plot the mutual information between y_i and b_i vs. the mutual information x_i and b_i

Generate negative log-likelihoods x_i using the symmetry condition & Gaussian model:

$$\sigma_{x_i}^2 = 2\mathbb{E}\left\{x_i(-1)^{b_i}\right\}$$

EXIT Charts

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 49, NO. 10, OCTOBER 2001 1727

Transactions Papers

Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes

Stephan ten Brink, *Member, IEEE*

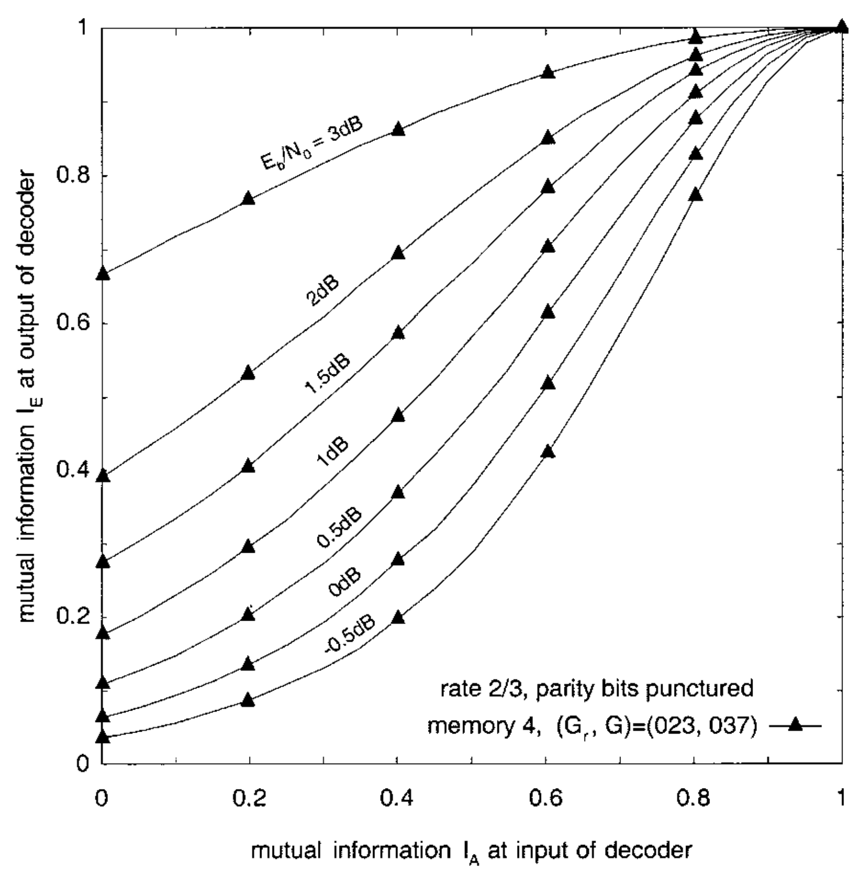


Fig. 2. Extrinsic information transfer characteristics of soft in/soft out decoder for rate 2/3 convolutional code; E_b/N_0 of channel observations serves as parameter to curves.

characterizing a single constituent convolutional code

EXIT Charts

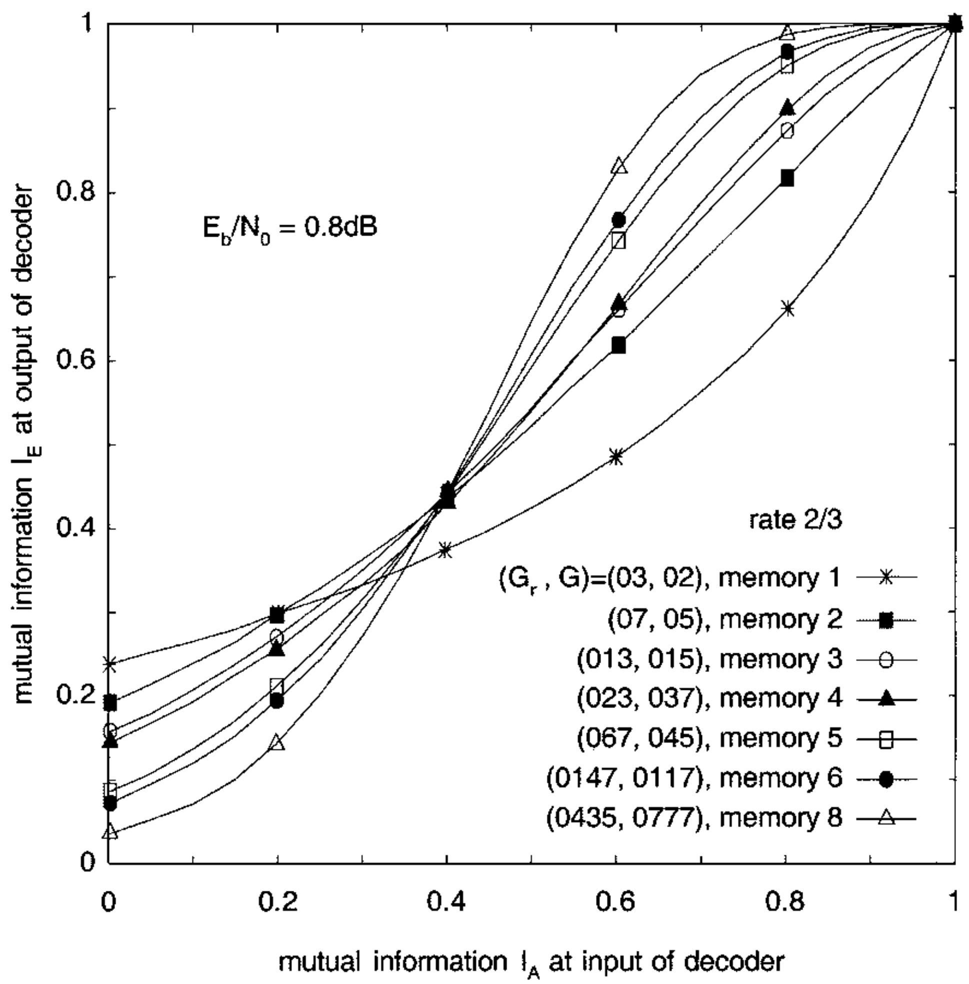


Fig. 3. Extrinsic information transfer characteristics of soft in/soft out decoder for rate 2/3 convolutional code, $E_b/N_0 = 0.8$ dB, different code memory.

varying code parameters affects these mutual information curves

EXIT Charts

EXIT chart for two fixed codes, above and below the threshold

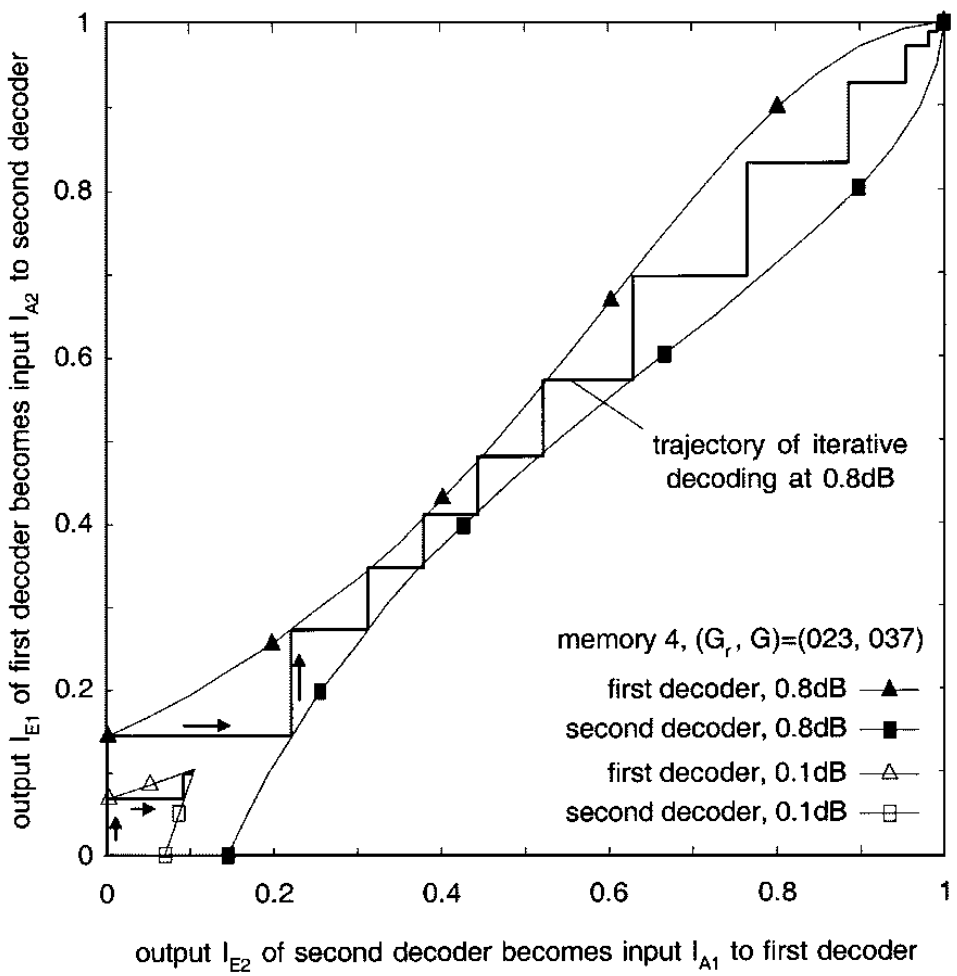


Fig. 5. Simulated trajectories of iterative decoding at $E_b/N_0 = 0.1$ dB and 0.8 dB (symmetric PCC rate 1/2, interleaver size 60 000 systematic bits).

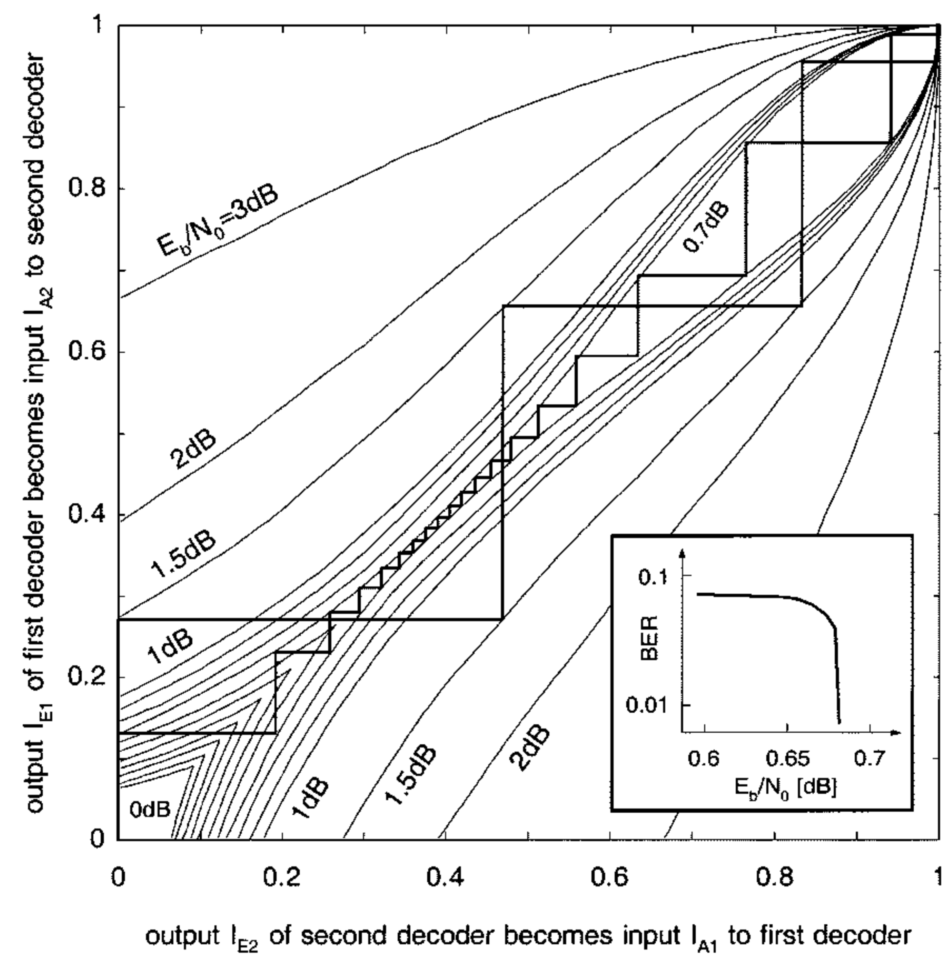


Fig. 6. EXIT chart with transfer characteristics for a set of E_b/N_0-values; two decoding trajectories at 0.7 dB and 1.5 dB (code parameters as in Fig. 5, PCC rate 1/2); interleaver size 10^6 bits.

Examples References

SNR Threshold Optimization

[4] S. ten Brink, "Convergence of iterative decoding," *IEE Electronics Letters*, pp. 1117–1119, June 1999.

[5] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commununication*, pp. 1727–1737, October 2001.

[6] T.J. Richardson and R.L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

[7] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 619–673, February 2001.

Uniform Interleaver Analysis

[10] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Trans. Information Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[11] D. Divsalar and F. Pollara, "Hybrid concatenated codes and iterative decoding," Tech. Rep., JPL-TDA, August 1997, 42–130.

ISI-AWGN Channel

with QASK Modulation

a post-matched filter model: $$z_k = f_k * x_k + w_k = \sum_{m=0}^{L} f_m x_{k-m} + w_k$$

FIR ISI in AWGN

Optimal processing is Viterbi Algorithm (hard-out) or FBA (soft-out)

Number of states is M^L — bad complexity scaling

OFDM (discrete multitone)