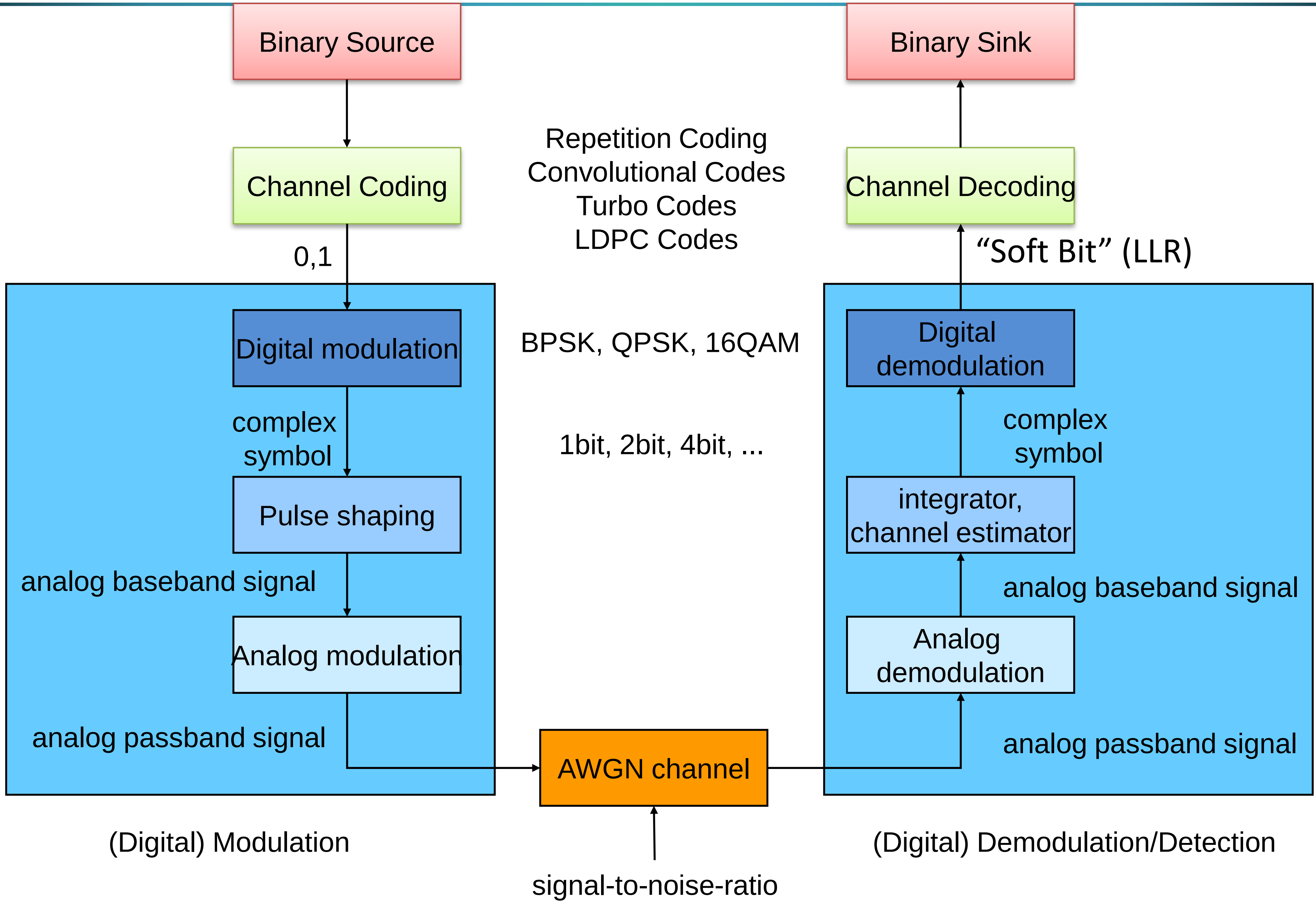
WLAN – Channel Coding

- Der WLAN-Standard spezifiziert CRC-Codes (FCS) im PHY Header zur Fehlererkennung
- Der WLAN-Standard spezifiziert Faltungscodes zur Fehlerkorrektur (FEC, Forward Error Correction)
  - die Dekodierung ist nicht spezifiziert
  - effizienter ist Soft-Decision Decoding (in Kapitel 3: Hard-Decision Decoding)
- Der WLAN-Standard spezifiziert ab dem 802.11n Standard (High Throughput PHY) optional auch LDPC Codes (Low Density Parity Check Codes)
  - nur Soft-Decision Decoding

Soft “Decision” Decoding

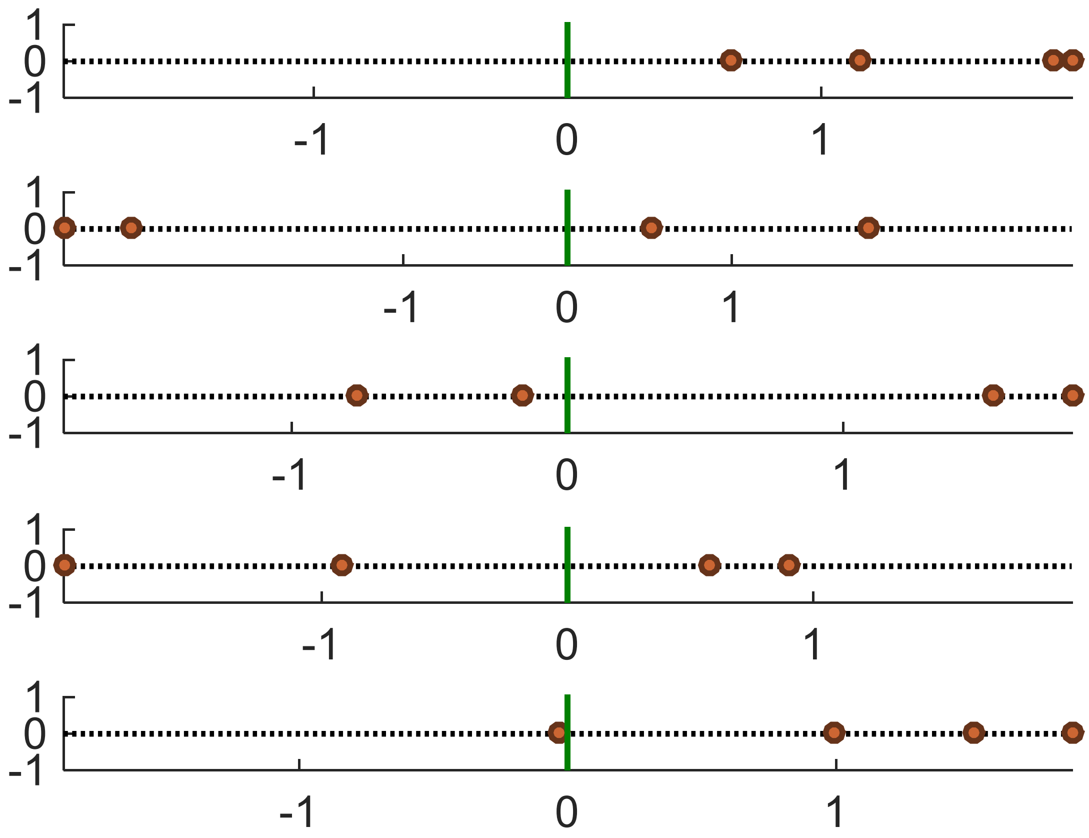


Hard vs. Soft Decision Decoding

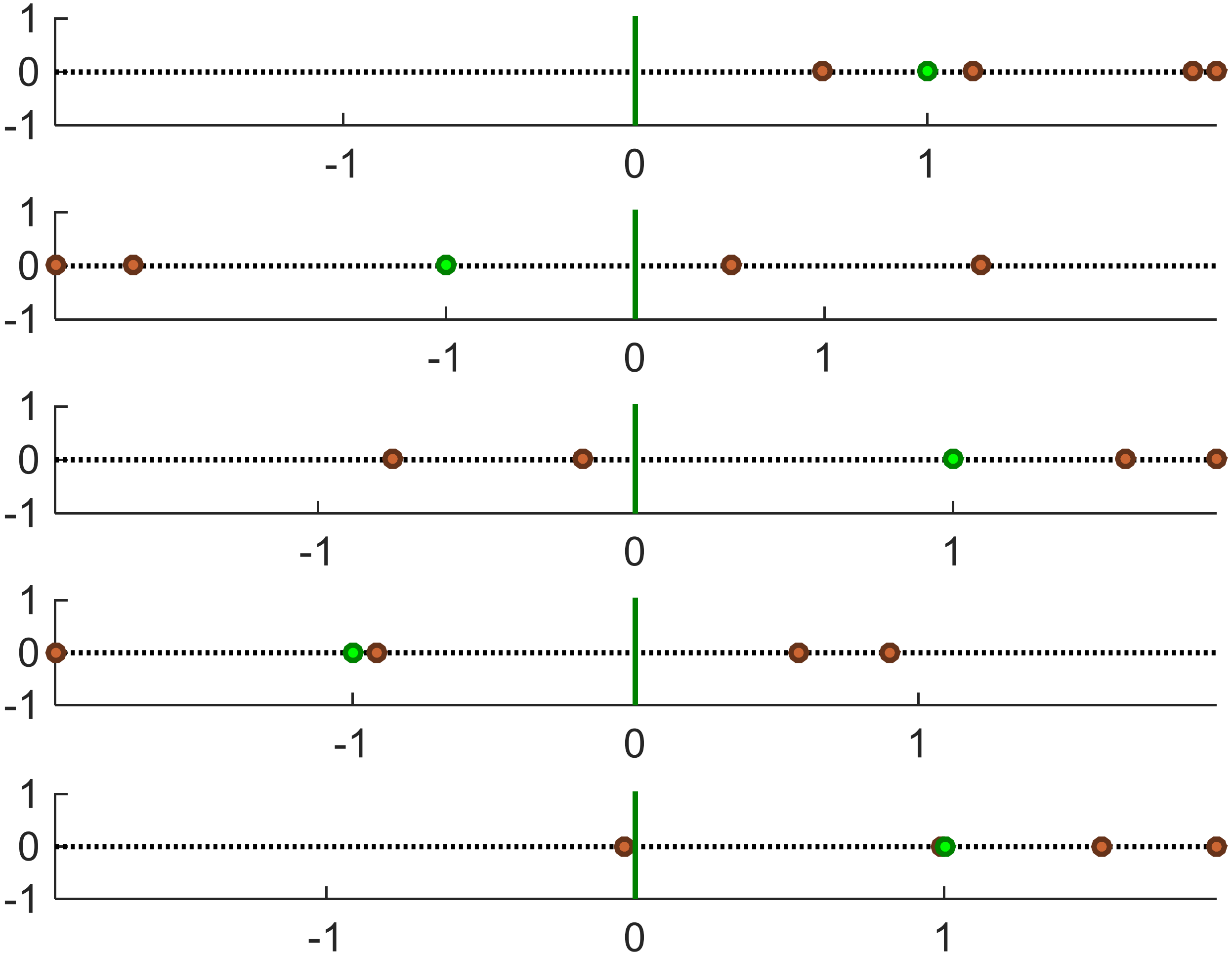
- Hard-decision decoding for (n,k)-code:
  - input: n binary values
  - output: k binary values
  - key code property:
    - minimal Hamming distance
    - number of correctable errors
- Soft Decision Decoding for (n,k)-code:
  - input: n quantized (integer) or unquantized (real) values
  - output: k binary values
  - key code property:
    - robustness against noise
    - bit error rate (BER) curve or frame error rate (FER) curve
      - shows BER/FER versus SNR/EsN0/EbN0 [dB]
      - minimum SNR that meets a target FER

Decoding: 1-4 Repetition Coding with BPSK

- Transmission of 5 Bits with an SNR of -6 dB
- Can you detect which bit was sent?

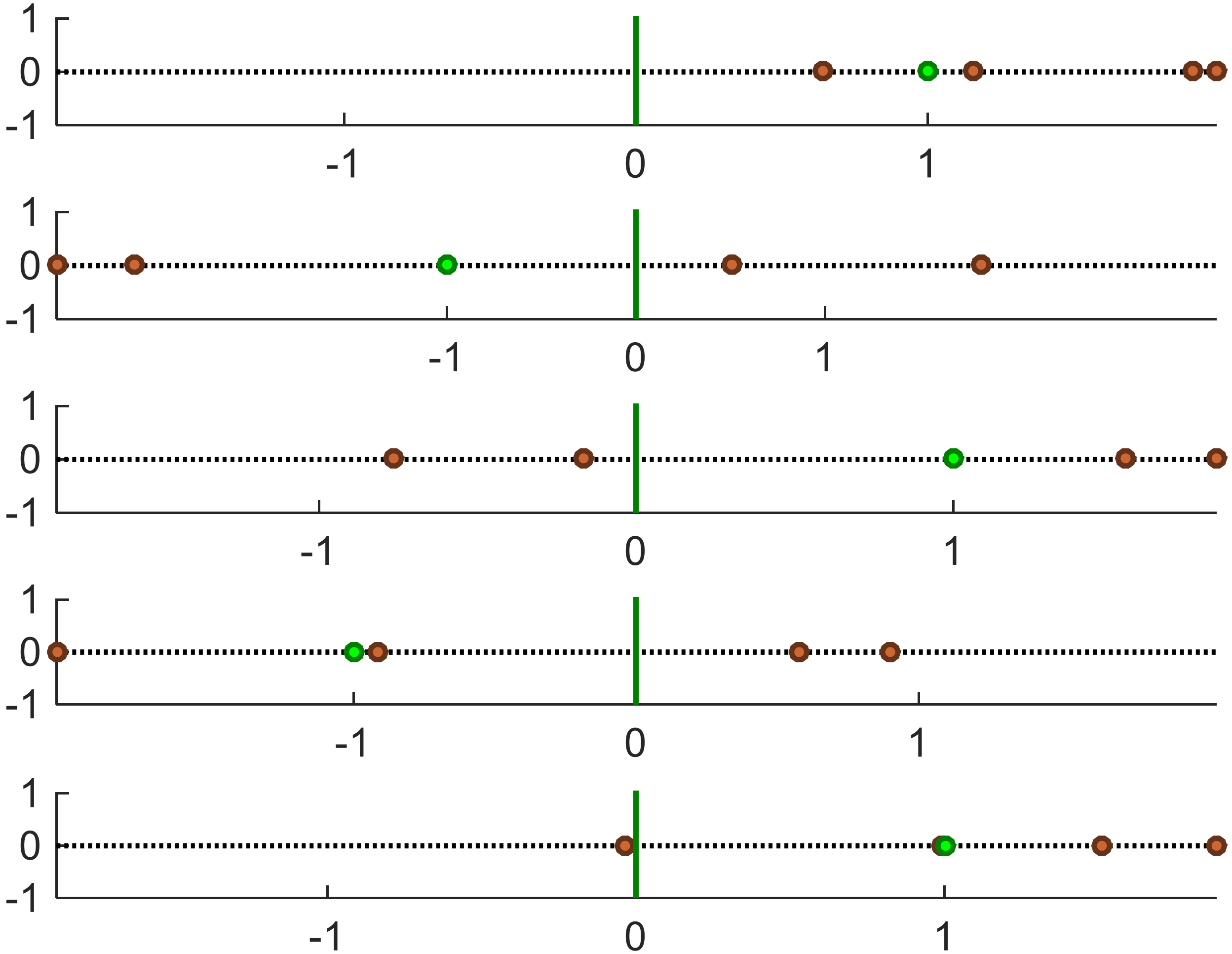


Decoding: 1-4 Repetition Coding with BPSK



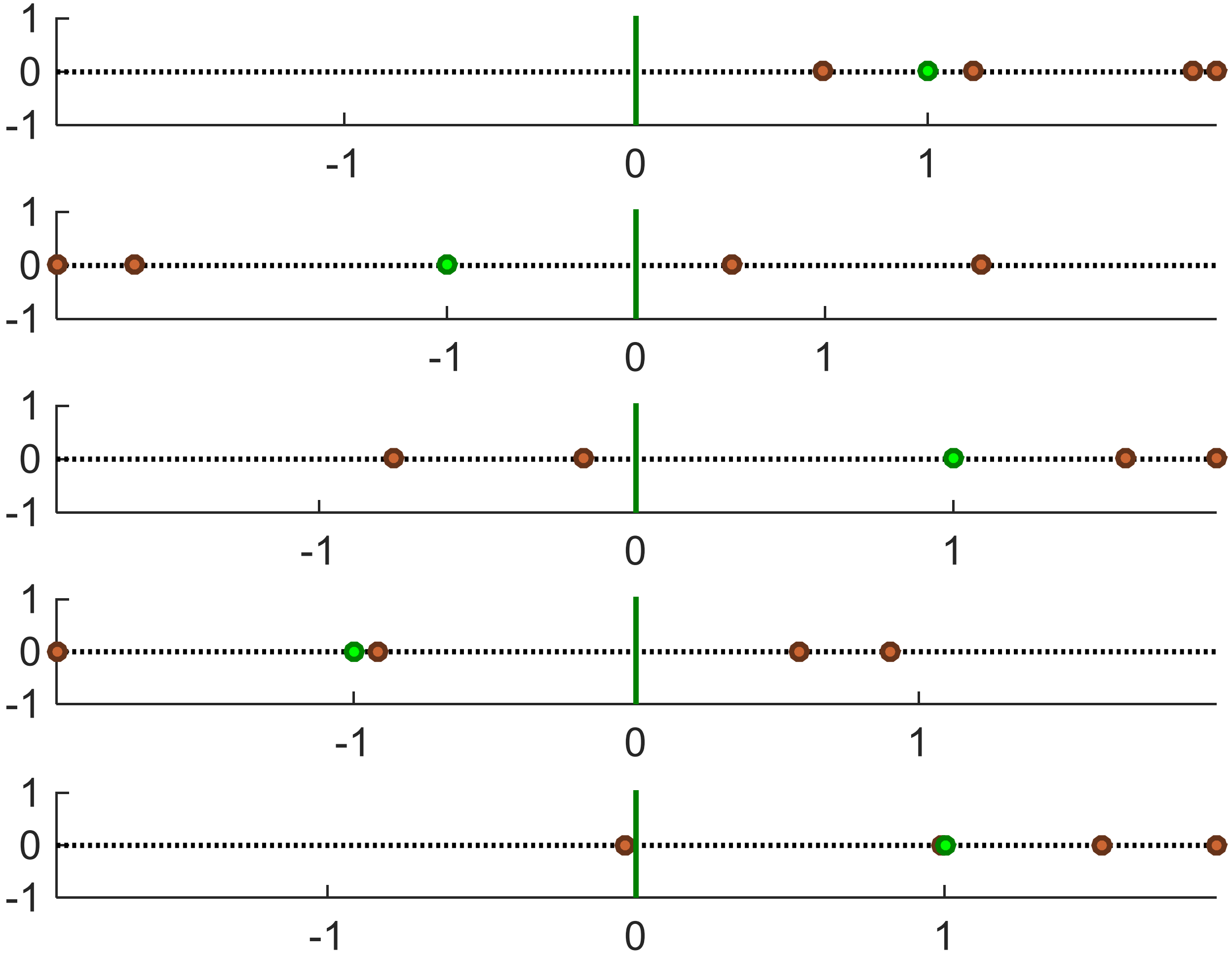
Hard Decision	Soft Decision

Decoding: 1-4 Repetition Coding with BPSK



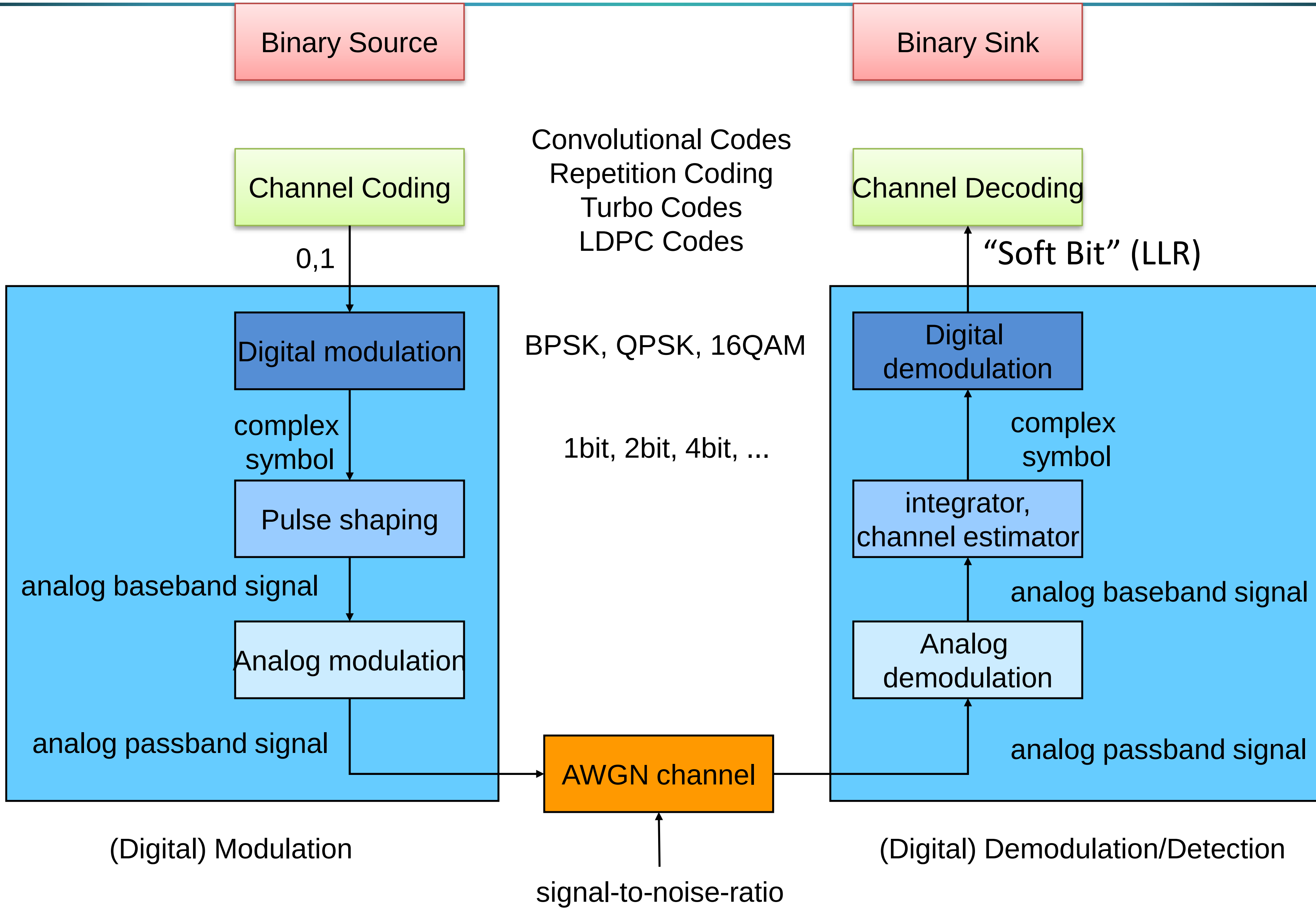
Hard Decision	Soft Decision
4:0 (strong one)	
2:2 (undecided)	
2:2 (undecided)	
2:2 (undecided)	
1:3 (weak one)	

Decoding: 1-4 Repetition Coding with BPSK



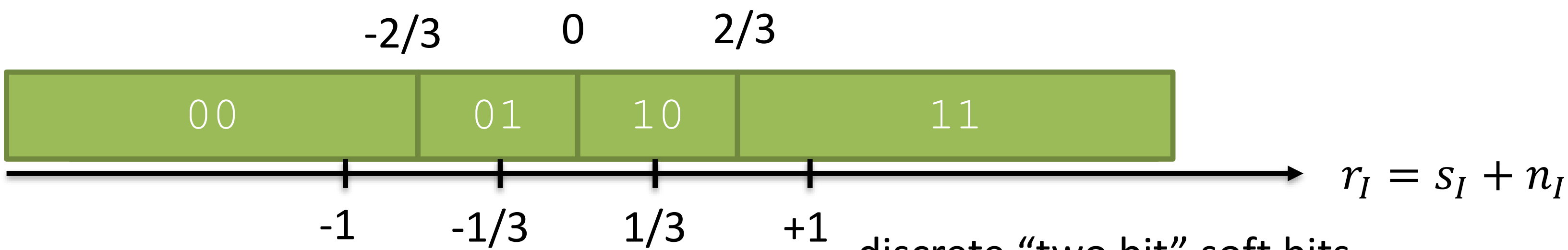
Hard Decision	Soft Decision
4:0 (strong one)	1.42 (one)
2:2 (undecided)	-0.84 (one)
2:2 (undecided)	0.61 (one)
2:2 (undecided)	-0.37 (zero)
1:3 (weak one)	1.1 (one)

Soft “Decision” Decoding



Soft (LLR) Demodulation

- The output of the demodulation process is a “soft” bit that may be either a real unquantized value or a binary value quantized to K values
  - e.g. a soft bit with K=4 values may represent the following intervals



discrete “two bit” soft bits

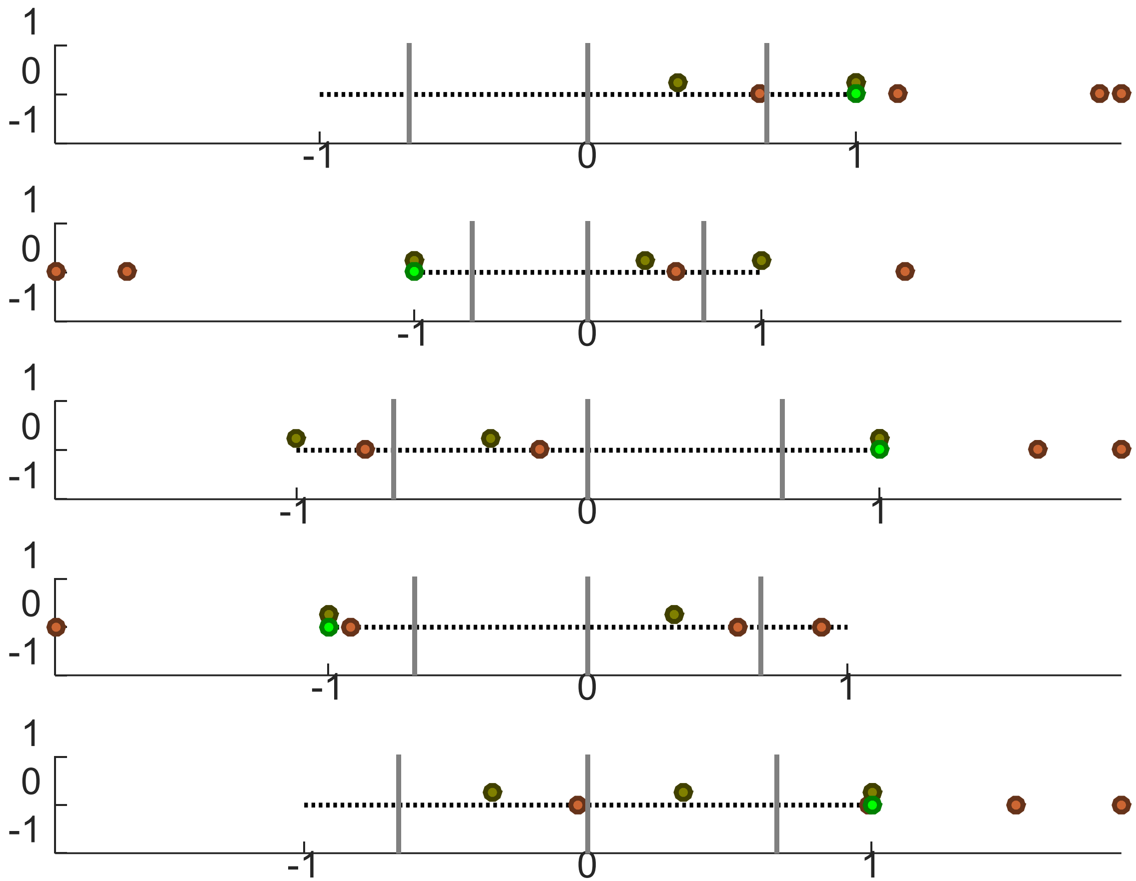
- Real-valued soft bits are calculated as log-likelihood ratios (LLR)

$$LLR(x) = -2 \cdot \ln \frac{\overbrace{\sum_{s=S_0} P\{x|s\}}^{probability\ that\ x\ was\ received\ when\ a\ "0"\ was\ sent}}{\underbrace{\sum_{s=S_1} P\{x|s\}}_{probability\ that\ x\ was\ received\ when\ a\ "1"\ was\ sent}}$$

  - S_1 are all symbol constellations where the decoded bit is “1” and S_0 are all symbol constellations where the decoded bit is “0”
  - with BPSK: $LLR(x) = -4 \cdot SNR \cdot r_I$

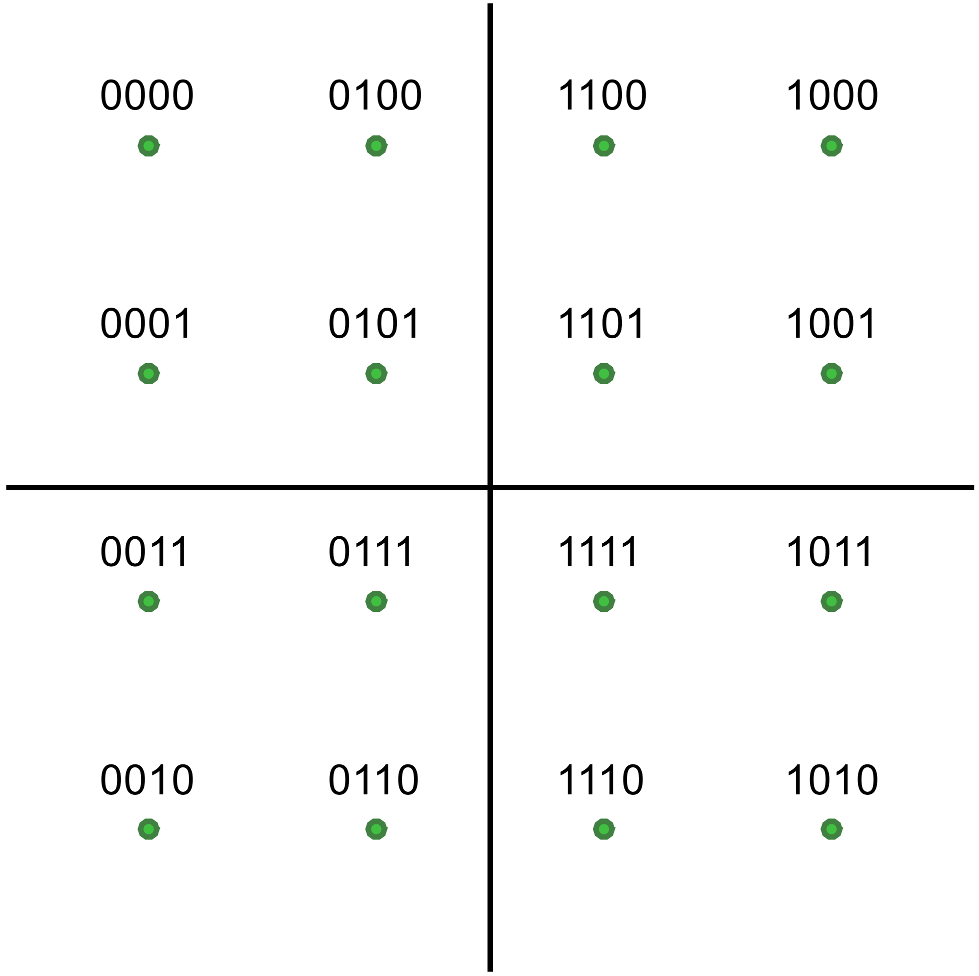
Quantized Soft-Demodulation: 1-4 Repetition Coding with BPSK

- Two-bit-quantization with 4 values: $-1, -1/3, 1/3, 1$
- Quantized values are shown as dark green spots

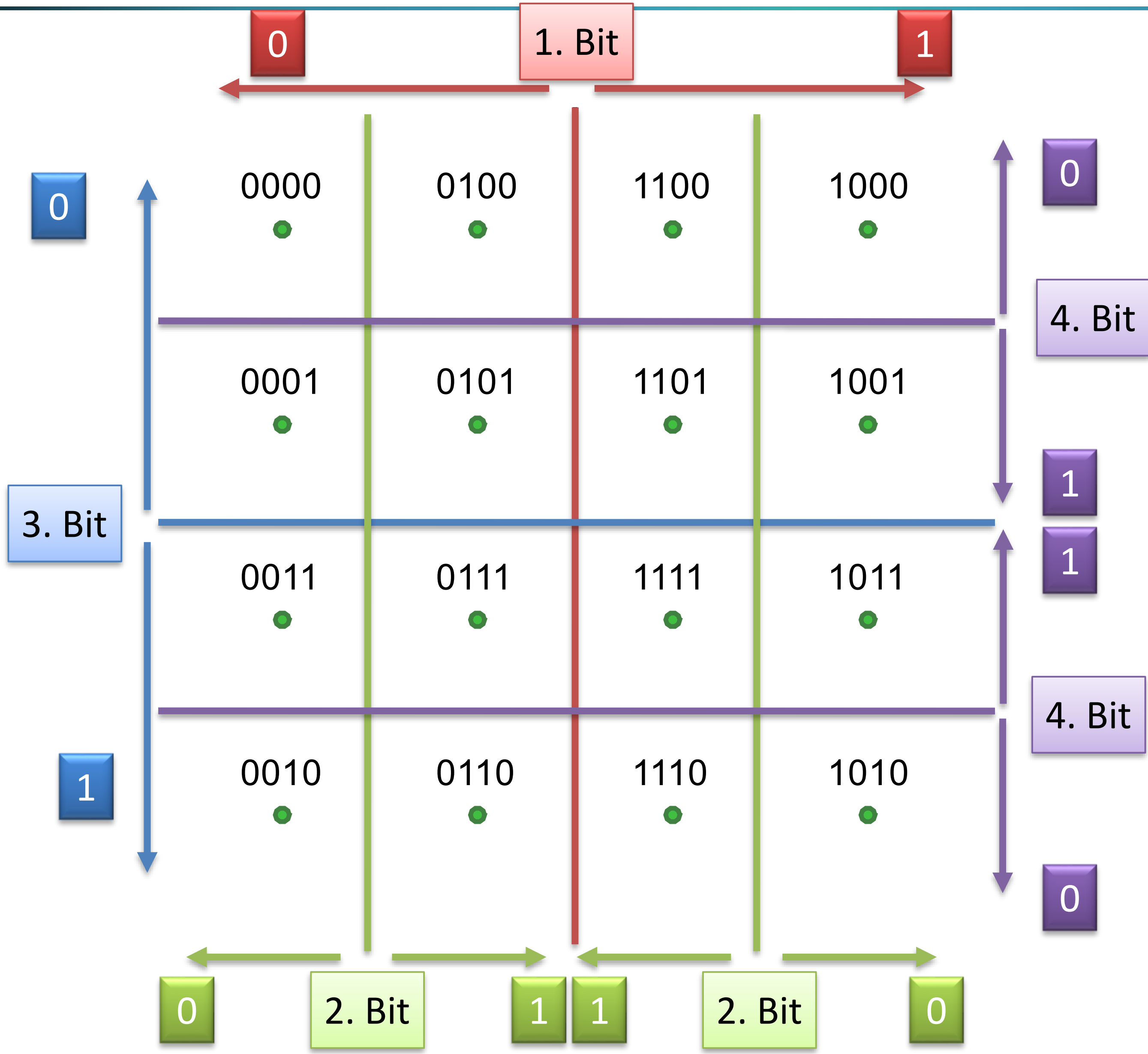


Example: Gray Coding

- Gray coding ensures that neighbored symbols differ only in a single bit



Soft Demodulation – 16 QAM with Gray coding



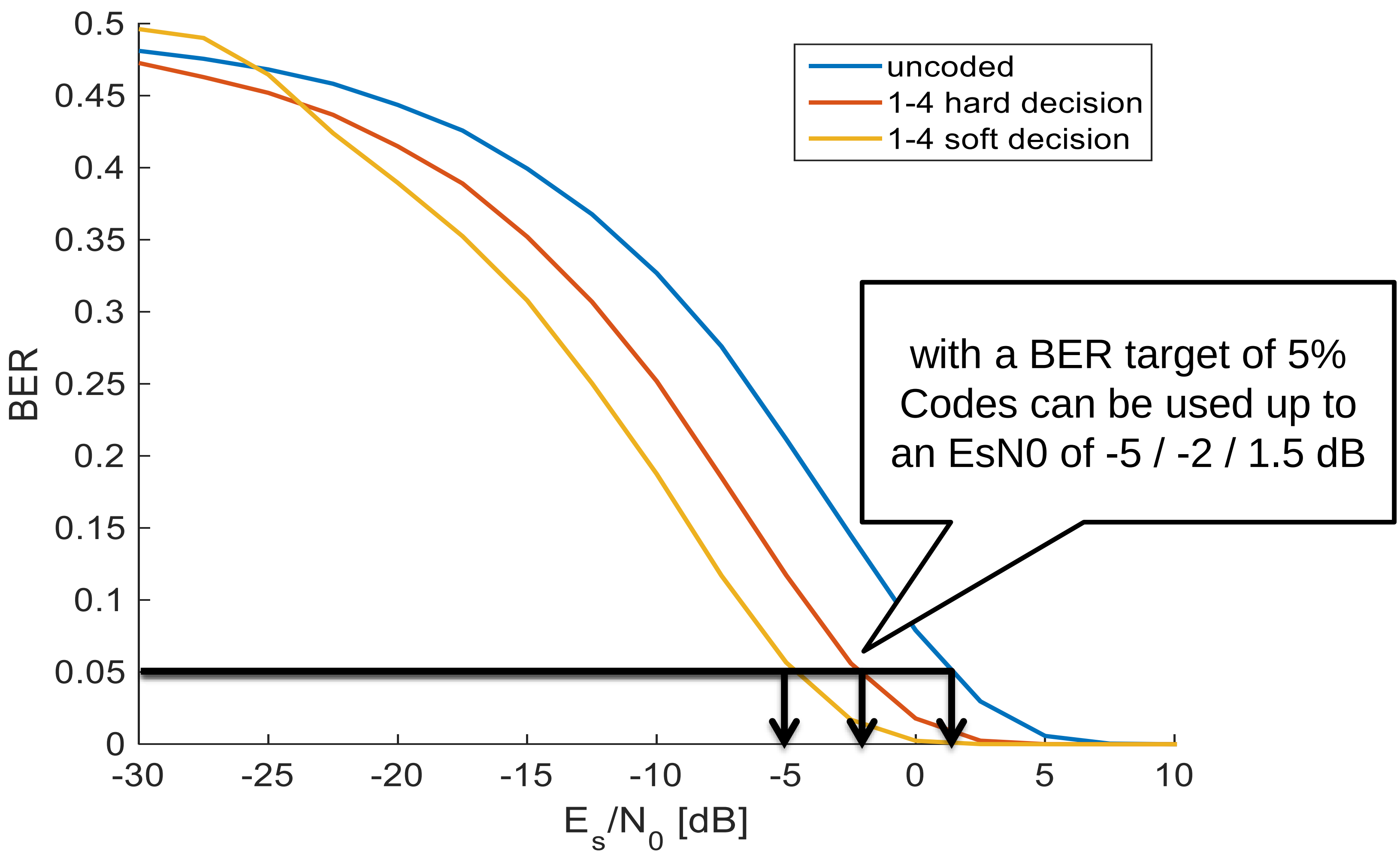
$LLR(x_1)=real(x)$

$LLR(x_2)=-2*abs(real(x))+1$

$LLR(x_3)=-imag(x)$

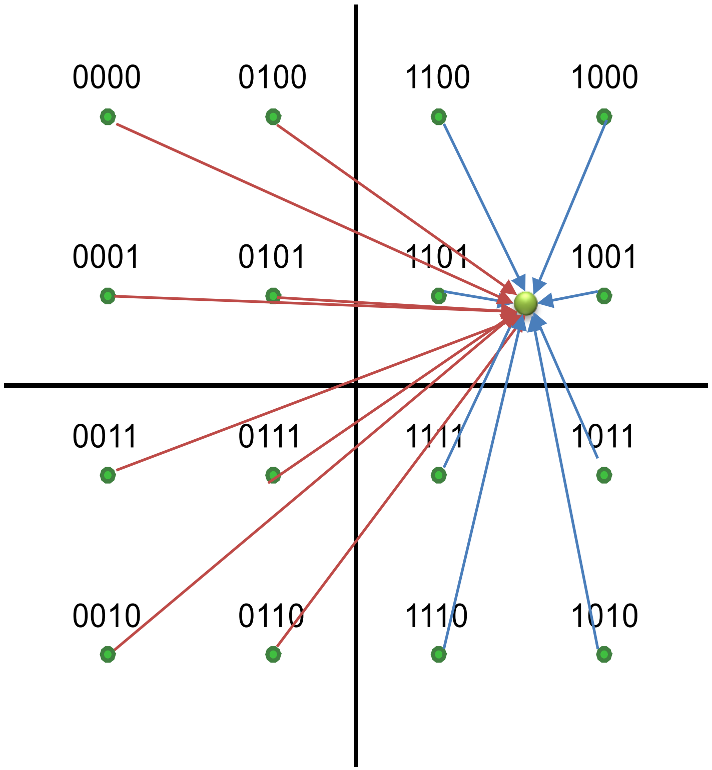
$LLR(x_4)=-2*abs(imag(x))+1$

BER curve for Soft-Decision Repetition Decoding



Soft Decision Decoding - Theory

- Soft (LLR) Demodulation for ith bit in a symbol:
  - channels characteristics (SNR) gives us the likelihood that a symbol r is received when a symbol s was sent $P\{r|s\}$
  - the likelihood that $s_i = 0$ was sent is the sum of the likelihoods of all symbols $s \epsilon S_{i0}$ with the ith bit s_i being “0”
  - the LLR (log-likelihood-ratio) is defined as two-times the negative logarithm of the ratio for receiving a “0” to receiving a “1”



$$LLR(r_i) = -2 \cdot \ln \frac{\overbrace{\sum_{s \in S_{i0}} P\{r|s\}}^{probability\ that\ r\ is\ received\ if\ s_i=0\ is\ sent}}{\underbrace{\sum_{s \in S_{i1}} P\{r|s\}}_{probability\ that\ r\ is\ received\ if\ s_i=1\ is\ sent}}$$

Soft Decision Decoding - Theory

- Soft Decision Decoding:
  - maximum likelihood decoding: find the code word that maximizes the likelihood of the received signal / the received soft bits

$$x^* = \arg\max_{x \epsilon X} P\{y|x\}$$

Large search space:
$|X| = 2^k$

y: received signal
x^*: decoded signal
X: set of all data words
$P\{y|x\}$: probability that y is received when data word x was encoded and sent

- AWGN channels
  - produce independent bit errors
  - metric for best code word is minimum distance to soft bits

$$x^* = \arg\min_{x \epsilon X} \sum_{i=1}^{n} |C_i(x) - y_i|$$

$C_i(x)$: ith bit of code word of data word x
n: length of code word

Hard Decision Viterbi Decoding

- Objective:
  - find the code word that is closest to the received bit sequence
  - find the path through the trellis that is closest to the received bit sequence
- Algorithm:
  - states: $s = 0,\ldots,2^M - 1$
  - path metric of state s at step j: $J(j,s)$
  - initialization of path metric: $J(0,s) = \begin{cases} 0 & \text{, for } s = 0 \\ \infty & \text{, for } s \neq 0 \end{cases}$
  - at step j for every state s:
    - best predecessor to state s: $q(s) = \arg\min_{q} \left\{ J(j-1,q) + H(r_j,q,s) \right\}$

$$H(r_j,q,s) = \begin{cases} \text{Hamming Distance}\left(r_j, \mathit{output}(q \to s)\right) & \text{if } \exists q \to s \\ \infty & \mathit{else} \end{cases}$$

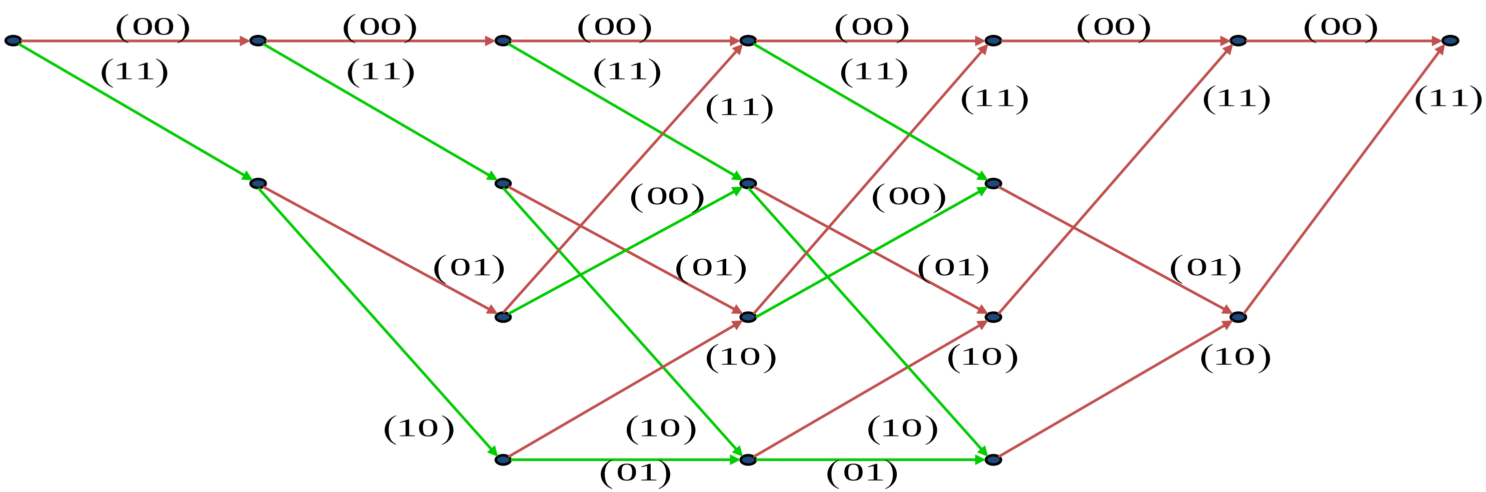
    - path metric for state s: $J(j,s) = J(j-1,q(s)) + H(r_j,q(s),s)$
    - best path (code word) to state s: $c(j,s) = \left[c(j-1,s)\,\mathit{input}(q \to s)\right]$

Andrew J. Viterbi

Hard Decision Viterbi Decoding Example

- Input: 1101 Output: 11 10 10 00 01 11

  Received bit sequence: 11 11 10 01 01 11

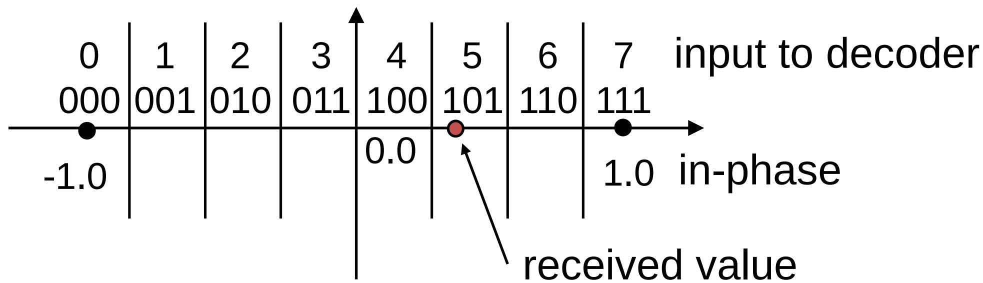


J(j,s)/c(j,s)		11	11	10	01	01	11
(00)	0	2/0	4/00	2/100	2/1100	3/11000	2/110100
(10)	∞	0/1	2/01	2/101	2/1101	∞	∞
(01)	∞	∞	1/10	1/110	2/1010	2/11010	∞
(11)	∞	∞	1/11	2/011	2/0111	∞	∞

Stage by stage the low-cost path to every state is found. In each stage every state can be reached by two states of the previous stage. The costs for the path via a previous state are the costs to reach the previous state plus the transition costs to the next state. The costs for the cheapest path are the cost for the state in the current stage. Due to the appended zeros there is only a single state, the all-zeros state, in the last stage and the path with minimal costs from the all-zeros state at the beginning to the all-zeros state at the end corresponds to the decoded bit sequence.

Soft Decision Viterbi Decoding

- in the example:
  - input to decoder: bit sequence
  - path metric: number of different bits
  - variant: hard decision decoding
- soft decision decoding:
  - input to decoder: real numbers/quantized values
  - example: BPSK



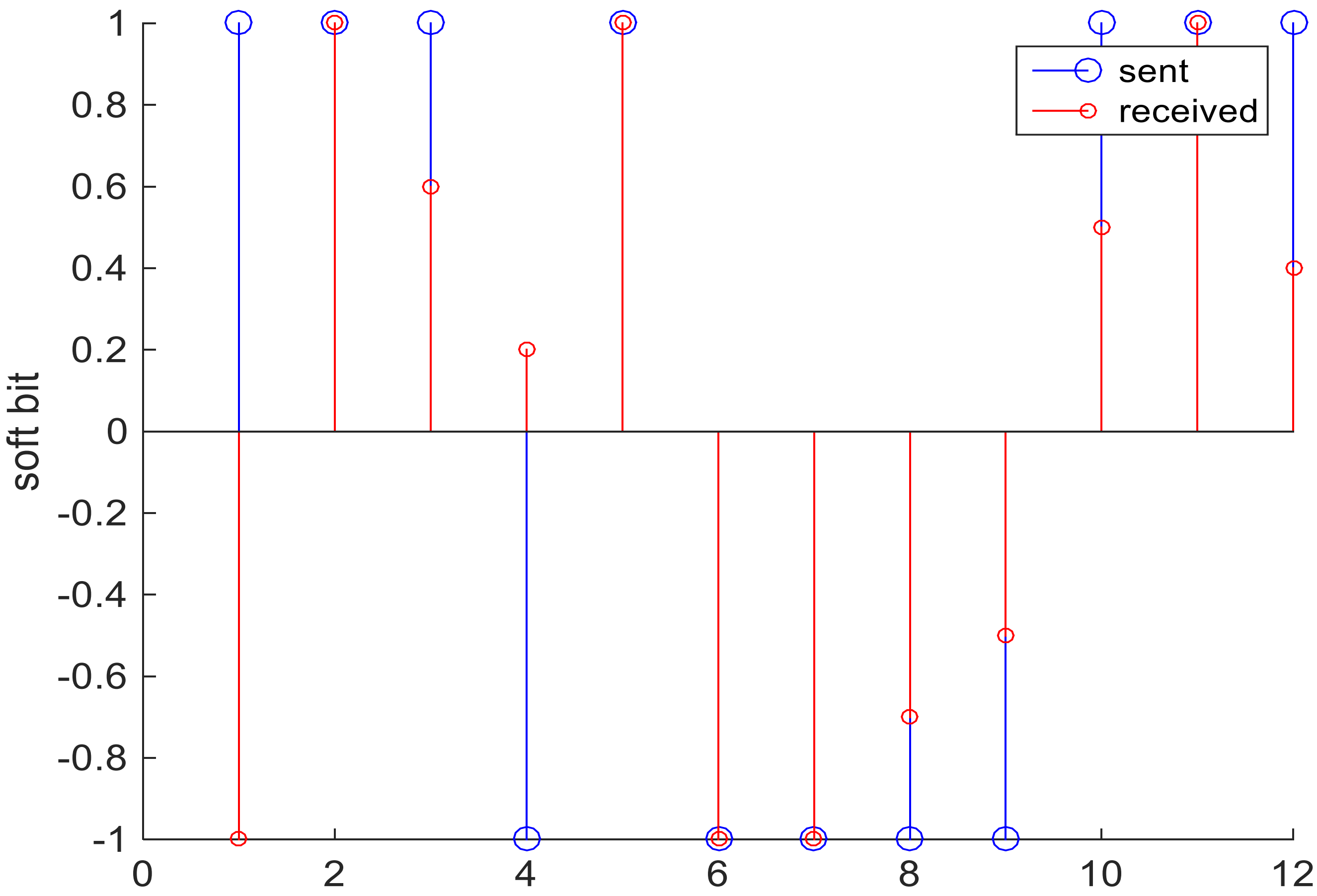
- path metric

$$H\left(r_j, q, s\right) = \begin{cases} \sum_{i=1}^{n} \left| r_{j,i} - output\left(q \to s, i\right) \right| & \text{if } \exists q \to s \\ \infty & else \end{cases}$$

input sequence to decoder: 45

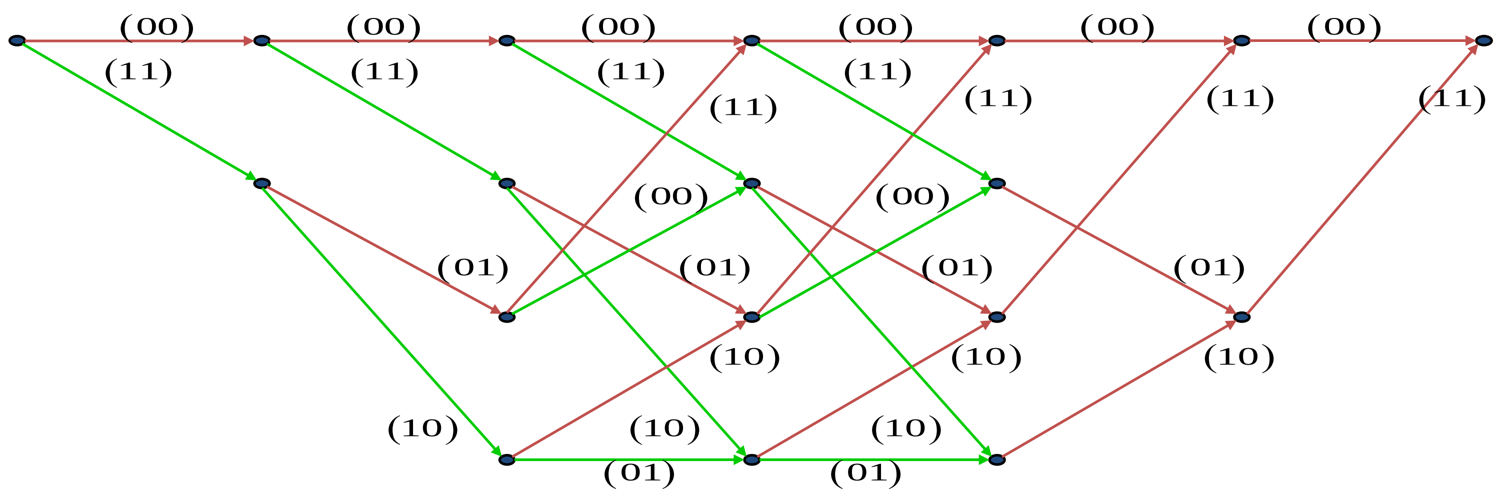
trellis output:	metric:
00	9
07	6
70	8
77	5

Example: Soft Bits after Demodulation



Soft Decoding Example (unquantized)

Sent	+1	+1	+1	-1	+1	-1	-1	-1	-1	+1	+1	+1
Received	-1.0	+1.0	0.6	0.2	1.0	-1.0	-1.0	-0.7	-0.5	0.5	1.0	0.2



Encoding:

00 → (-1,-1)

01 → (-1,+1)

10 → (+1,-1)

11 → (+1,+1)

	Sent	**+1**	**+1**	**+1**	**-1**	**+1**	**-1**	**-1**	**-1**	**-1**	**+1**	**+1**	**+1**
	Received	-1.0	+1.0	0.6	0.2	1.0	-1.0	-1.0	-0.7	-0.5	0.5	1.0	0.2
States	0: 00												
	1: 10												
	2: 01												
	3: 11												
	Back												

Soft Decoding Example

Sent	+1	+1	+1	-1	+1	-1	-1	-1	-1	+1	+1	+1
Received	-1.0	+1.0	0.6	0.2	1.0	-1.0	-1.0	-0.7	-0.5	0.5	1.0	0.2

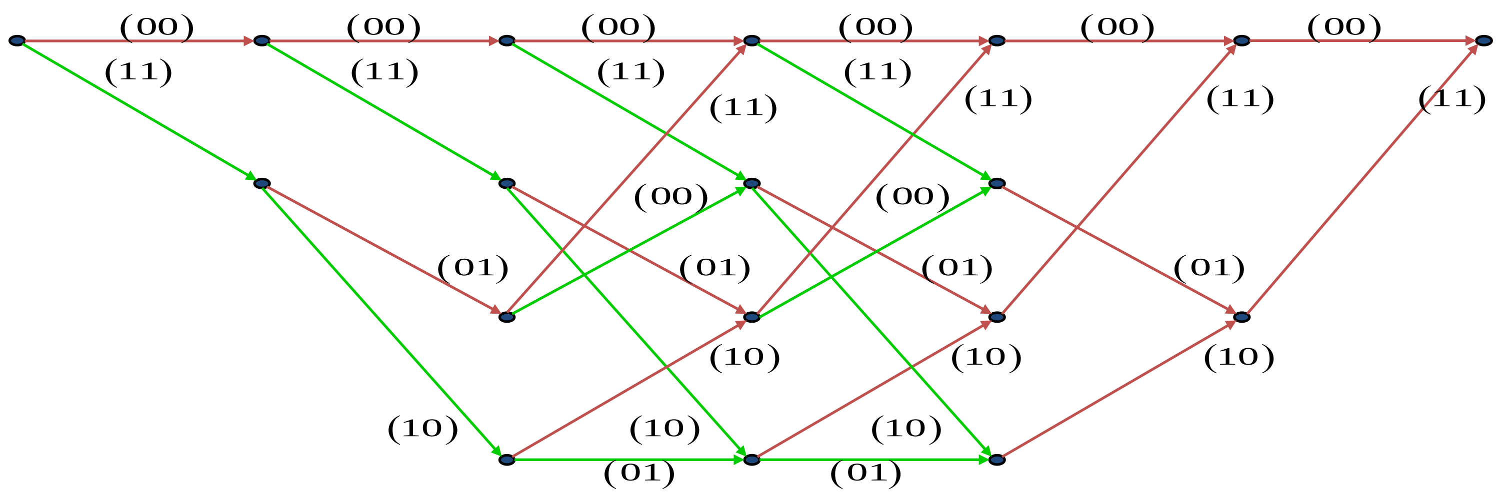
Encoding:

00 ➔(-1,-1)

01 ➔(-1,+1)

10 ➔(+1,-1)

11 ➔(+1,+1)

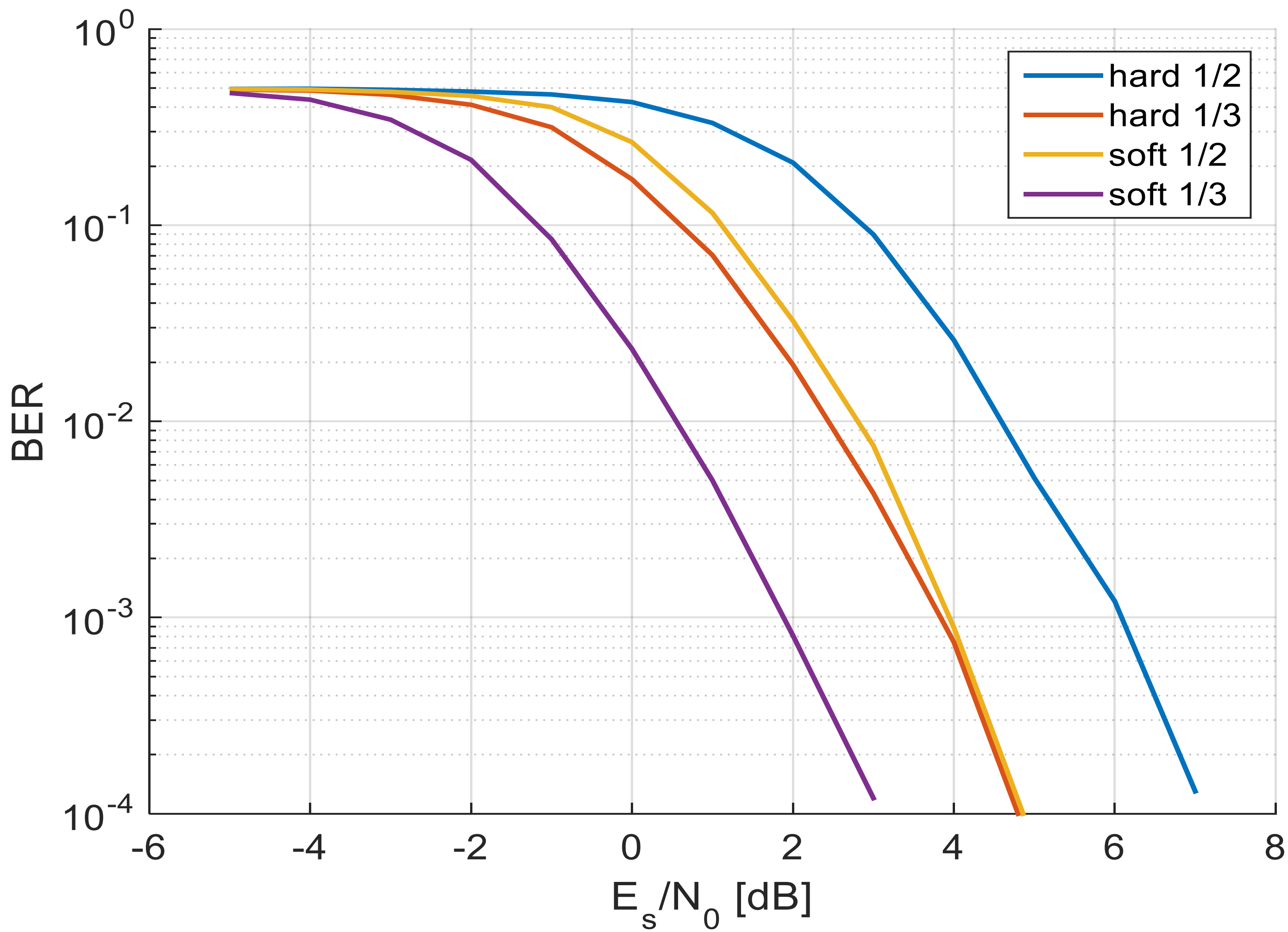


	Sent	+1	+1	+1	-1	+1	-1	-1	-1	-1	+1	+1	+1
	Received	-1.0	+1.0	0.6	0.2	1.0	-1.0	-1.0	-0.7	-0.5	0.5	1.0	0.2
States	0: 00	0:2		0:4.8		0:6.8	2:6.4	0:6.7	2:7.3	0:8.7	2:7.5	0:10.7	2:5.7
	1: 10	0:2		0:3.2		0:6.8	2:6.4	0:10.1	2:3.9				
	2: 01			1:4.4		1:7.2	3:3.6	1:8.1	3:5.5	1:4.9	3:7.9		
	3: 11			1:3.6		1:3.2	3:7.6	1:8.7	1:4.9				
	Back	0->1 (**1**,2.0)		1->3 (**1**,1.6)		3->2 (**0**,0.0)		2->1 (**1**,0.3)		1->2 (0,1.0)		2->0 (0, 0.8)	

Notation: [Previous State:Transistion Cost]   [Previous State-> Next State (Bit, Transition Cost)]

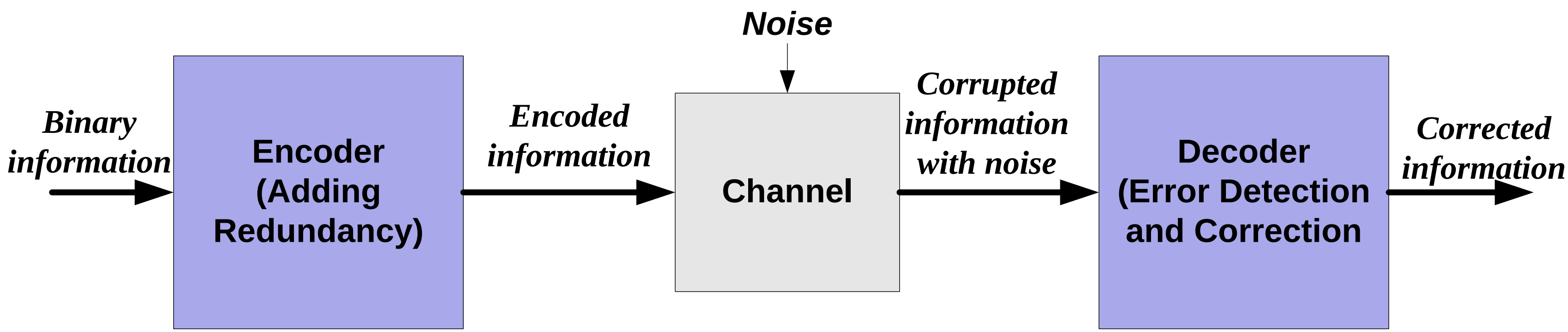
Performance Gain by Soft Decision Decoding – QPSK, AWGN

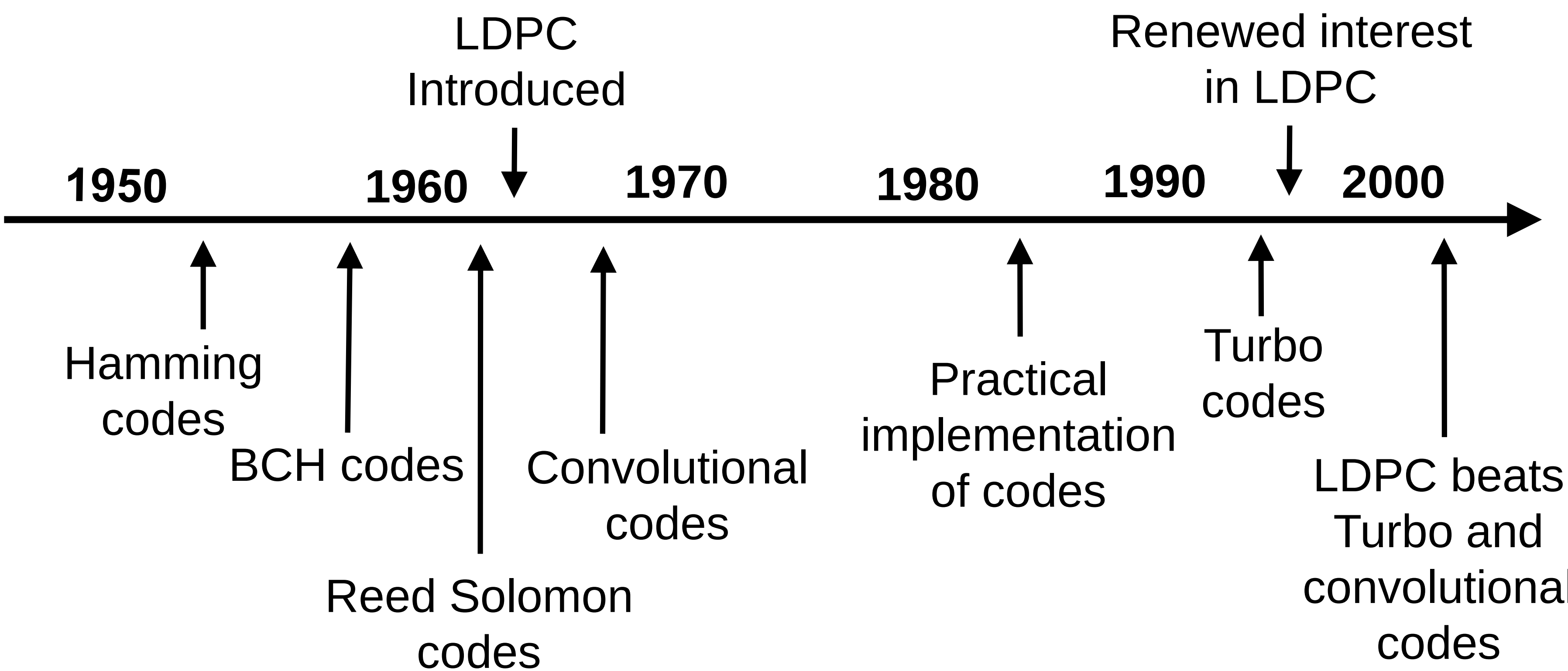
Simulation eines Frames mit 100000 Bits



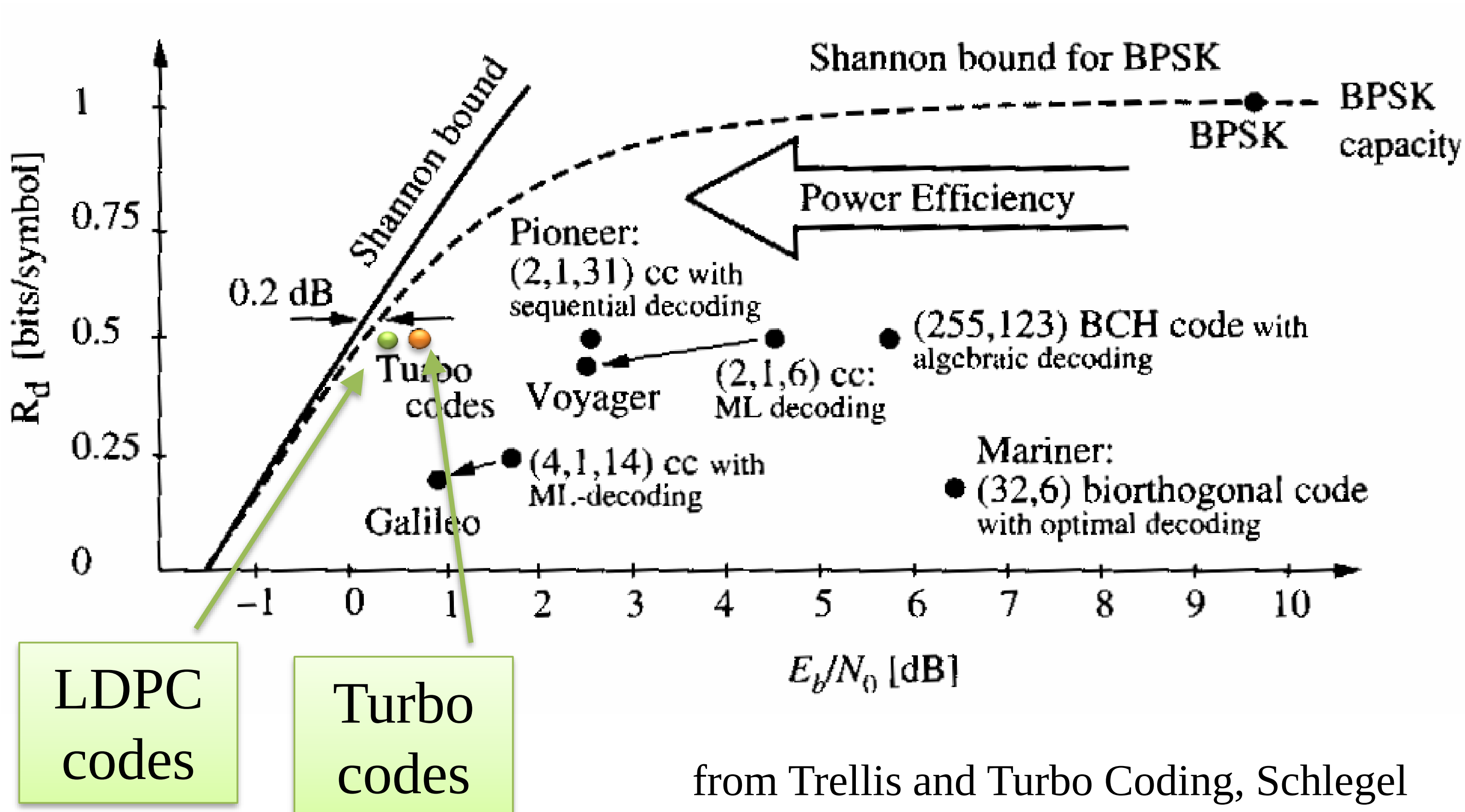
Chapter 5: IEEE 802.11 Wireless LAN

Overview


Noise
Binary information
Encoder (Adding Redundancy)
Encoded information
Channel
Corrupted information with noise
Decoder (Error Detection and Correction
Corrected information


LDPC Introduced
Renewed interest in LDPC
1950
1960
1970
1980
1990
2000
Hamming codes
BCH codes
Reed Solomon codes
Convolutional codes
Practical implementation of codes
Turbo codes
LDPC beats Turbo and convolutional codes

Evolution of Coding Technology

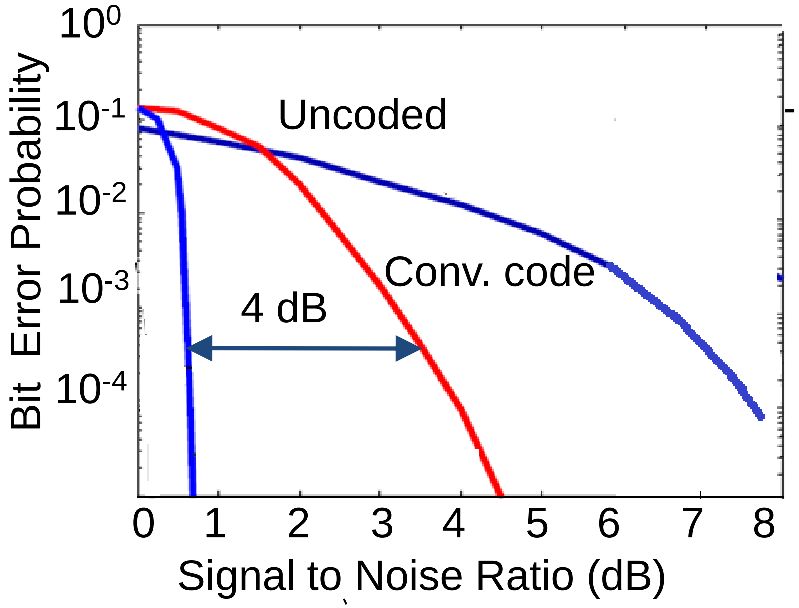


from Trellis and Turbo Coding, Schlegel and Perez, IEEE Press, 2004

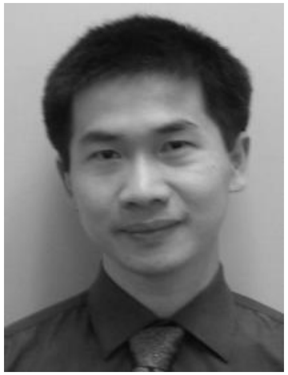
Application of LDPC Codes

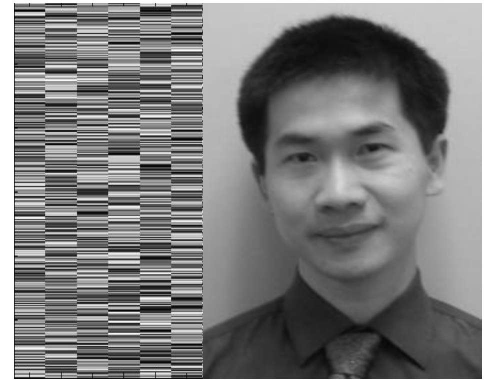
- Low Density Parity Check (LDPC) codes have superior error performance
  - 4 dB coding gain over convolutional codes

- Standards and applications
  - 10 Gigabit Ethernet (10GBASE-T)
  - Digital Video Broadcasting T2, DVB-C2)
  - Next-Gen Wired Home (G.hn)
  - WiMAX (802.16e)
  - WiFi (802.11n)
  - Hard disks
  - Deep-space satellite missions



Encoding Picture Example



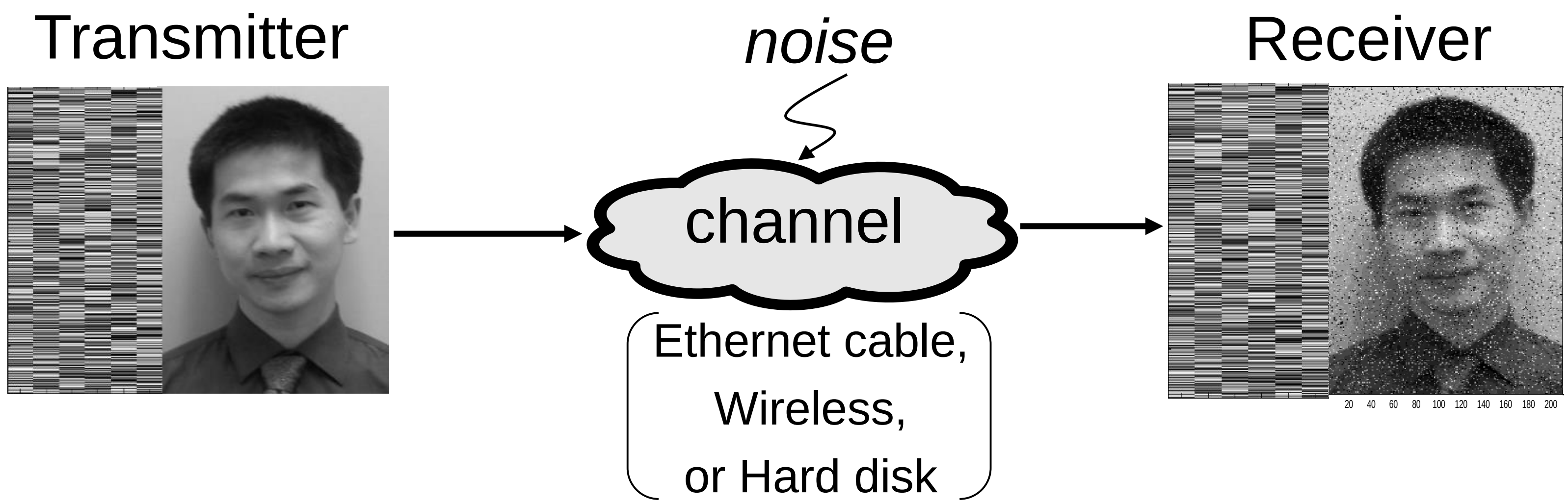


$$H=\begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots 1\,1\,0\,0\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\ldots 1\,1\,1\,0\,0\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0\ldots 0\,0\,1\,1\,0\,1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0\ldots 0\,0\,0\,1\,1\,0\,1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,1 \\ \ldots \qquad\qquad \ldots \end{bmatrix} \quad V=\begin{bmatrix} 1\,0\,1\,1\,1\,0\,1\,1\,0\,0\,0\,0\ \ldots\ 1\,1\,1\,1\,0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix}$$

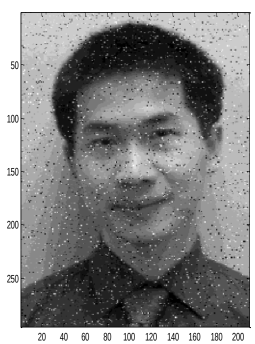
$$H.V_i^T=0$$

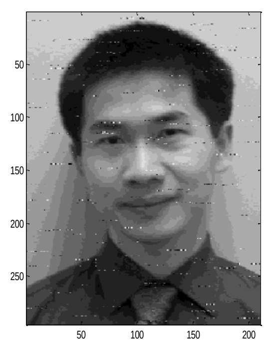
Binary multiplication called syndrome check

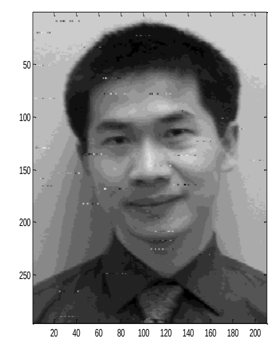
Decoding Picture Example



Iterative message passing decoding


Iteration 1


Iteration 5


Iteration 15


Iteration 16

LDPC Code

- LDPC code is based on a parity-check-matrix as we know from linear block codes
- A parity-check matrix is a set of linear equations

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

$$\begin{matrix} c_1 & + & c_2 & + & c_3 & + & c_5 & = & 0 \\ c_1 & + & c_3 & + & c_4 & + & c_6 & = & 0 \\ c_1 & + & c_2 & + & c_4 & + & c_7 & = & 0 \end{matrix}$$

Soft Decision Decoding with Parity Equations

- Parity-Check equation $$\boxed{c_1 + c_2 + c_3 + c_5 = 0}$$
- Mapping from $c_i \in \{0,1\}$ to $s_i \in \{1,-1\}$

$$\boxed{c_1 + c_2 + c_3 + c_5 = 0 \;\Leftrightarrow\; s_1 \cdot s_2 \cdot s_3 \cdot s_5 = 1}$$

- Soft Decoding:
  - soft bits s_i are real values
  - we have 3 soft bits and want to estimate the 4th soft bit
    - sign (zero or one): product of all signs must be 1 so the sign of the missing soft bits must be equal to the product of the signs of the known bits
    - value (confidence of estimation): the sign of the estimated bits changes if the sign of any known bit changes so the value is the minimum absolute value of the known bits
    - example for s_2 $$\boxed{s_2 = \prod_{i \in \{1,3,5\}} sign(s_i) \cdot \min_{i \in \{1,3,5\}} |s_i|}$$

Soft Decision Decoding with Parity Equations

- Example:

s_1	s_2	s_3	s_4
0.8	x	0.4	-0.3

- Estimation of s_2:
  - sign: $sign(s_2) = (+1) \cdot (+1) \cdot (-1) = -1$
  - value: $|s_2| = min(0.8{,}0.4{,}0.3) = 0.3$
  - soft bit: $s_2 = -0.3$

- Bits:

s_1	s_2	s_3	s_4
0.8	-0.3	0.4	-0.3

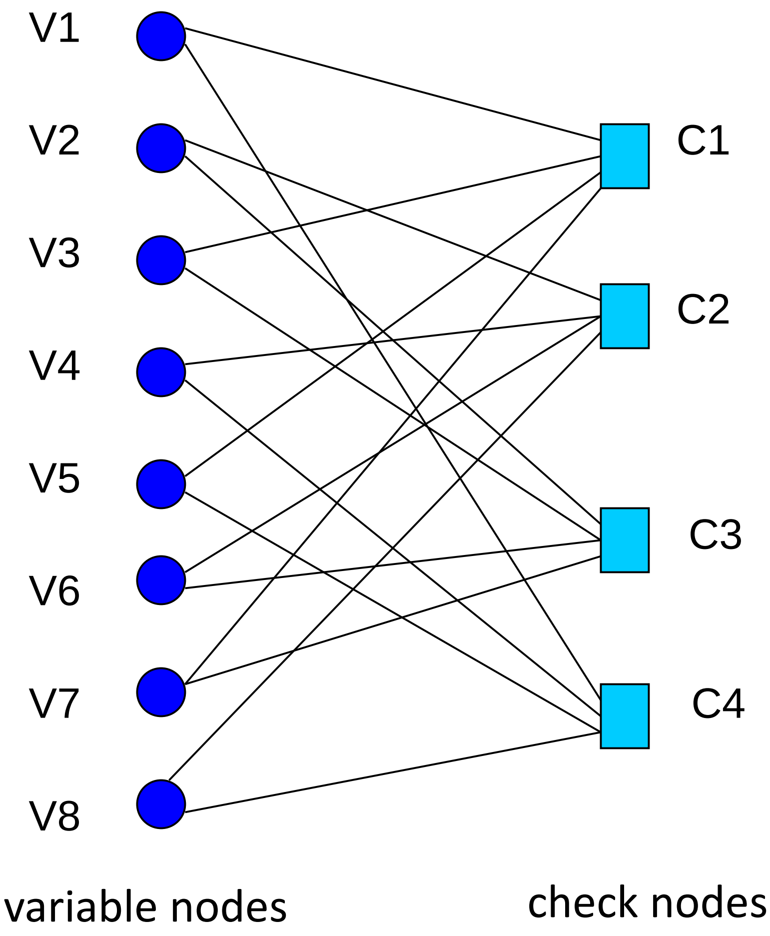
c_1	c_2	c_3	c_4
0	1	0	1

- Parity equation: $c_1 + c_2 + c_3 + c_5 = 0 + 1 + 0 + 1 = 0$

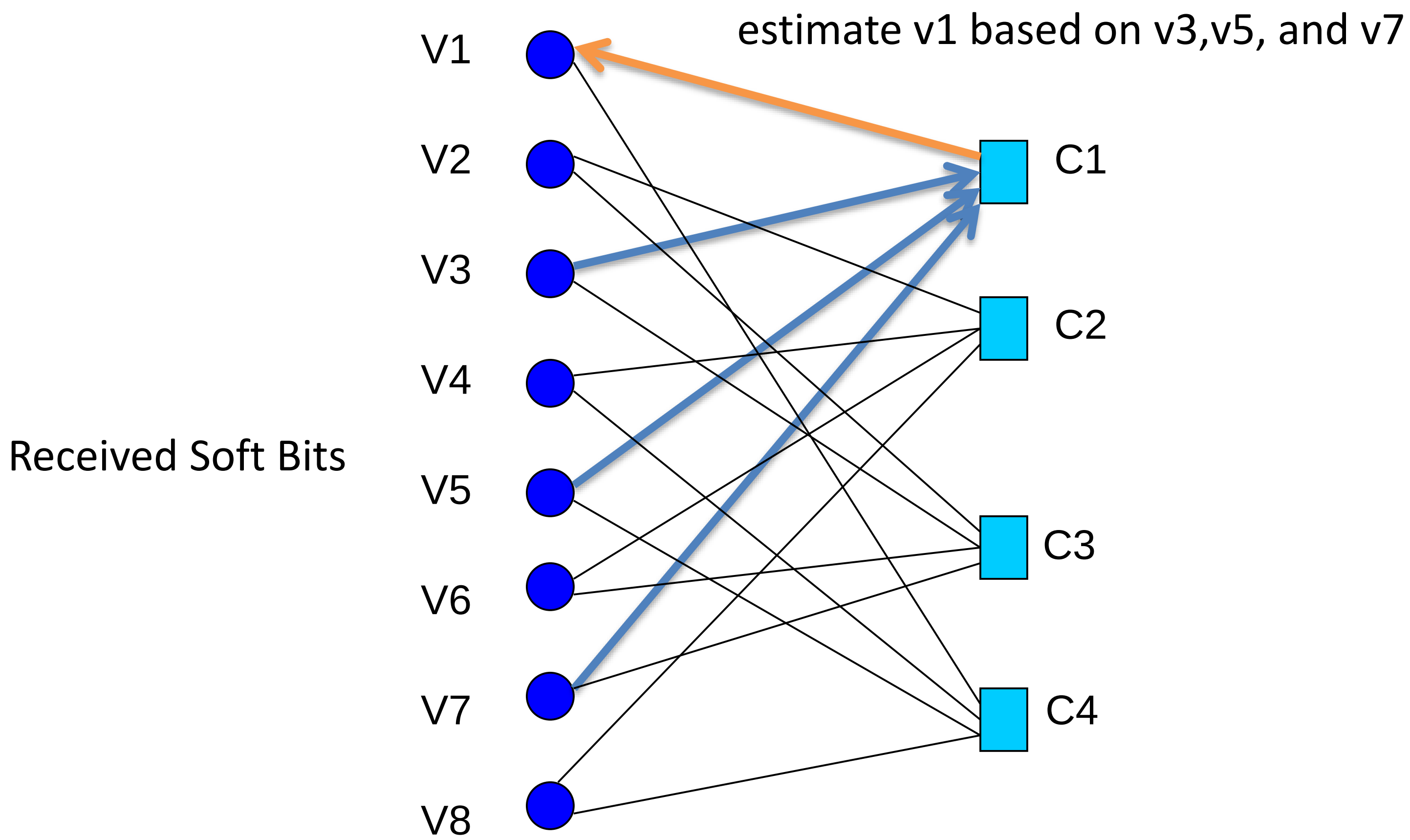
Representation by a Tanner Graph

Parity Check Matrix $\Longleftrightarrow$ Tanner Graph

$$
\begin{array}{c}
H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{array}{c} C1 \\ C2 \\ C3 \\ C4 \end{array} \\
\quad\;\; \begin{array}{cccccccc} V1 & V2 & V3 & V4 & V5 & V6 & V7 & V8 \end{array}
\end{array}
$$

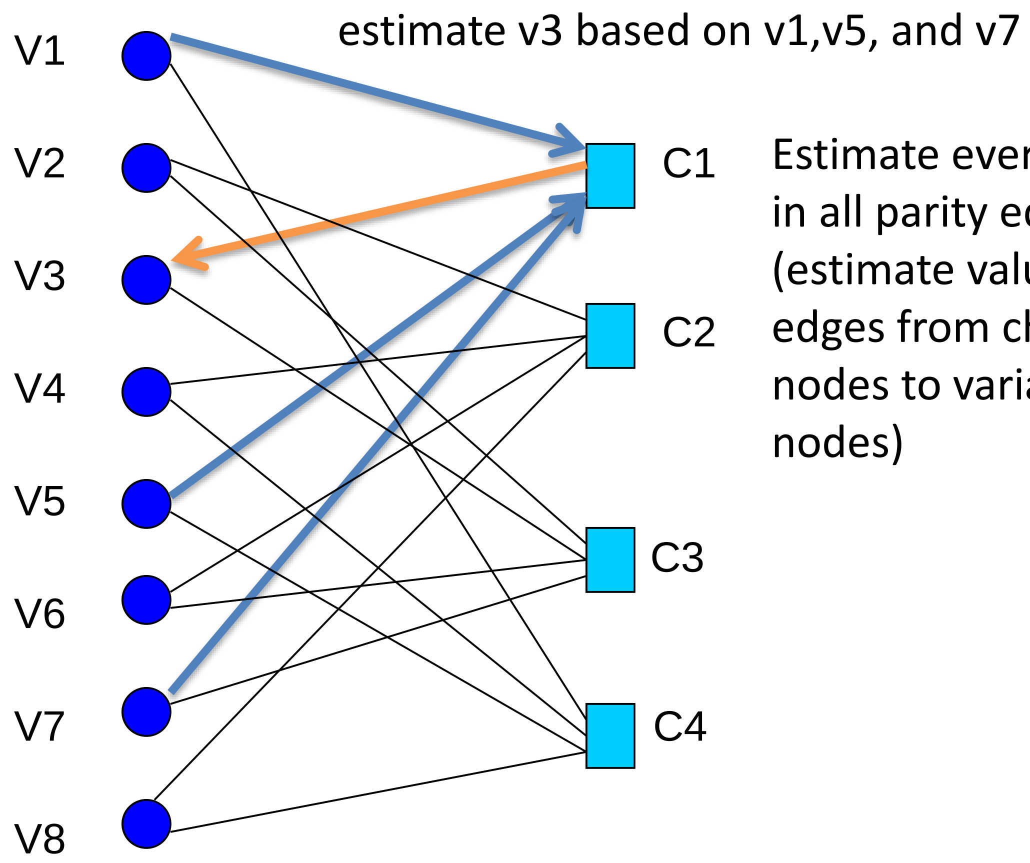


Representation by a Tanner Graph



Representation by a Tanner Graph

Received Soft Bits

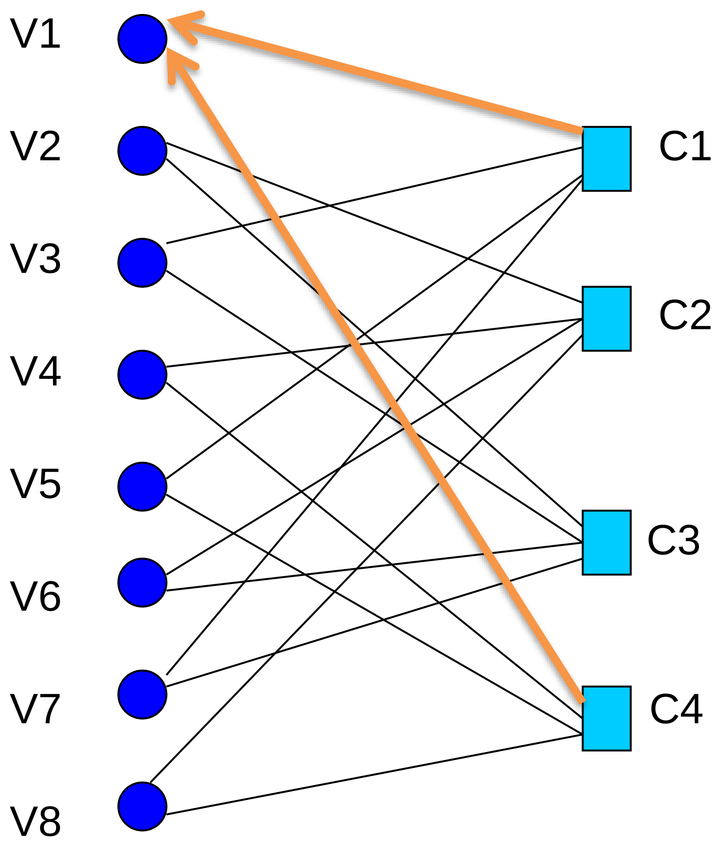


Estimate every soft bit in all parity equations (estimate values for all edges from check nodes to variables nodes)

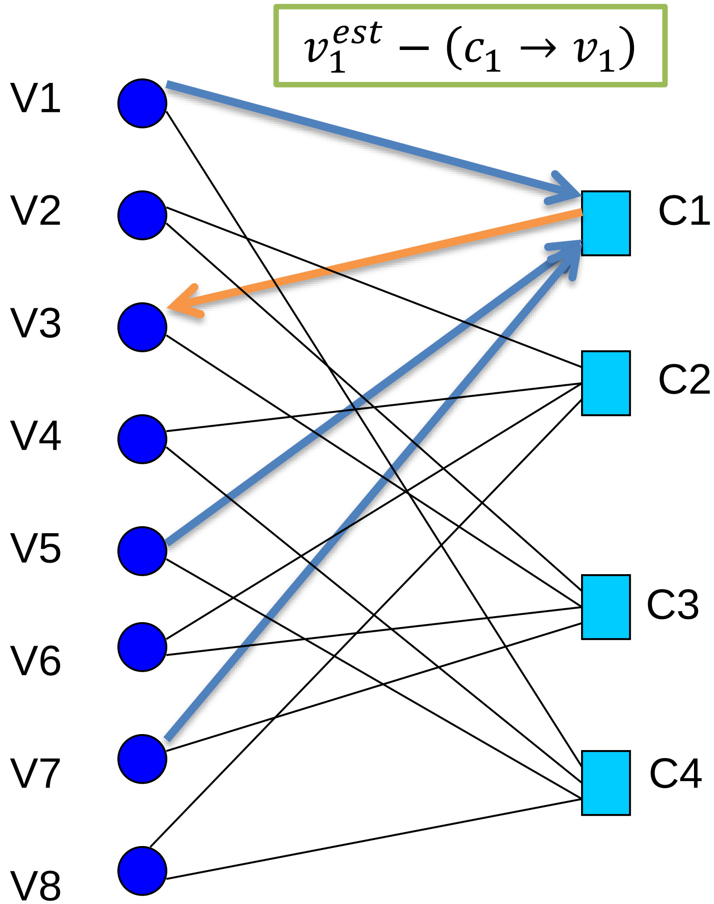
Representation by a Tanner Graph

$$v_1^{est} = v_1 + (c_1 \rightarrow v_1) + (c_4 \rightarrow v_1)$$

Update soft bit: correct the received bit according to the value estimated from the parity equation



Iterate



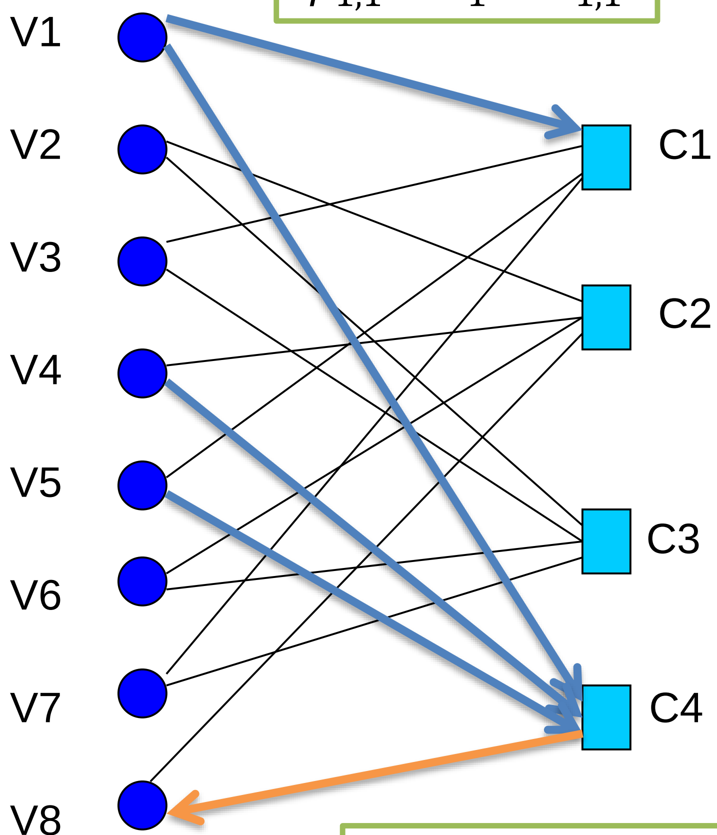
Estimate again every soft bit in all parity equations. Basis is the new estimated soft bit but **excluding** the estimate from the previous estimation.

Iterate

$$s_1 = v_1 + \sum_{i\in\{1,4\}} \alpha_{i,1}$$

$$\beta_{1,1} = s_1 - \alpha_{1,1}$$

Check parity equations based on sign of s_i and stop if all parity equations are fulfilled

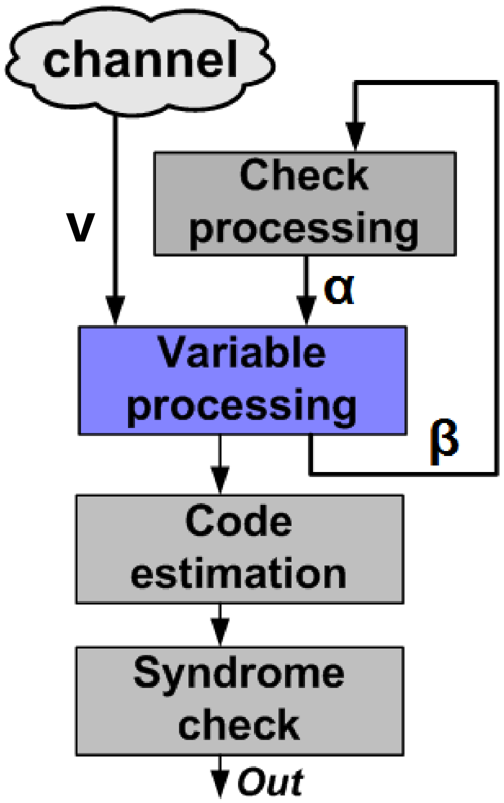


$$s_8 = v_8 + \sum_{i\in\{2,4\}} \alpha_{i,8}$$

$$\alpha_{4,8} = \prod_{i\in\{1,4,5\}} sign(\beta_{i,4}) \cdot \min_{i\in\{1,4,5\}} |\beta_{i,4}|$$

Algorithm

- receive soft bits v as output from soft demodulation
- continue until maximum number of iterations
  - compute new estimates of output bits
  - conduct syndrome check
  - break if syndrome check is successful



Example

- Parity Check Matrix: $H = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right]$

- Sent code word: $x = [0\ 0\ 1\ 1\ 0\ 1\ 1]$
- Received soft bits:
  $r = [-0.8, -0.1, -0.9, -1.2, +1.1, -0.7, -0.8]$
- Hard Decoding: $\mathrm{y} = [1\ 1\ 1\ 1\ 0\ 1\ 1]$
- Error Syndrome: $s = y \cdot H^T = [1\ 1\ 0]$
- Corrected codeword: $\mathrm{y} = [1\ 1\ 1\ 1\ \underline{1}\ 1\ 1]$

- Result of hard decision decoding is a wrong code word and payload
- Let's check how iterative soft decision decoding performs

Round 1: Initial Betas

H	1	0	0	1	1	1	0
	0	1	0	1	1	0	1
	0	0	1	1	0	1	1
v	-0,8	-0,1	-0,9	-1,2	1,1	-0,7	-0,8
beta	-0,8	0	0	-1,2	1,1	-0,7	0
	0	-0,1	0	-1,2	1,1	0	-0,8
	0	0	-0,9	-1,2	0	-0,7	-0,8
alpha	0,7	0	0	0,7	-0,7	0,8	0
	[illegible]	0,8	0	0,1	-0,1	0	0,1
	0	0	-0,7	-0,7	0	-0,8	-0,7
s	-0,1	0,7	-1,6	-1,1	0,3	-0,7	-1,4

$\beta = H.* v$

Round 1: Betas to Alphas

H	1	0	0	1	1	1	0
	0	1	0	1	1	0	1
	0	0	1	1	0	1	1
v	-0,8	-0,1	-0,9	-1,2	1,1	-0,7	-0,8
beta	-0,8	0	0	-1,2	1,1	-0,7	0
	0	-0,1	0	-1,2	1,1	0	-0,8
	0	0	-0,9	-1,2	0	-0,7	-0,8
alpha	0,7	0	0	0,7	-0,7	0,8	0
	0	0,8	0	0,1	-0,1	0	0,1
	0	0	-0,7	-0,7	0	-0,8	-0,7
s	-0,1	0,7	[illegible]	-1,1	0,3	-0,7	-1,4

$$sign(-1{,}2) * sign(1{,}1) * sign(-0{,}7) =' +'$$
$$min(1{,}2; 1{,}1; 0{,}7) = 0{,}7$$

Round 1: Betas to Alphas

H	1	0	0	1	1	1	0
	0	1	0	1	1	0	1
	0	0	1	1	0	1	1
v	-0,8	-0,1	-0,9	-1,2	1,1	-0,7	-0,8
beta	-0,8	0	0	-1,2	1,1	-0,7	0
	0	-0,1	0	-1,2	1,1	0	-0,8
	0	0	-0,9	-1,2	0	-0,7	-0,8
alpha	0,7	0	0	0,7	-0,7	0,8	0
	0	0,8	0	0,1	-0,1	0	0,1
	0	0	-0,7	-0,7	0	-0,8	-0,7
s	-0,1	0,7	-1,6	-1,1	0,3	-0,7	-1,4

$$sign(-0{,}9) * sign(-1{,}2) * sign(-0{,}8) = '-'$$
$$min(0{,}9; 1{,}2; 0{,}8) = 0{,}8$$

Round 1: Alphas to s

H	1	0	0	1	1	1	0
	0	1	0	1	1	0	1
	0	0	1	1	0	1	1
v	-0,8	-0,1	-0,9	-1,2	1,1	-0,7	-0,8
beta	-0,8	0	0	-1,2	1,1	-0,7	0
	0	-0,1	0	-1,2	1,1	0	-0,8
	0	0	-0,9	-1,2	0	-0,7	-0,8
alpha	0,7	0	0	0,7	-0,7	0,8	0
	0	0,8	0	0,1	-0,1	0	0,1
	0	0	-0,7	-0,7	0	-0,8	-0,7
s	-0,1	0,7	-1,6	-1,1	0,3	-0,7	-1,4

$$s = v + \sum \alpha$$

$$-1{,}2 + 0{,}7 + 0{,}1 - 0{,}7 = -1{,}1$$

Round 1: Bits and Error Check

H	1	0	0	1	1	1	0
	0	1	0	1	1	0	1
	0	0	1	1	0	1	1
v	-0,8	-0,1	-0,9	-1,2	1,1	-0,7	-0,8
beta	-0,8	0	0	-1,2	1,1	-0,7	0
	0	-0,1	0	-1,2	1,1	0	-0,8
	0	0	-0,9	-1,2	0	-0,7	-0,8
alpha	0,7	0	0	0,7	-0,7	0,8	0
	0	0,8	0	0,1	-0,1	0	0,1
	0	0	-0,7	-0,7	0	-0,8	-0,7
s	-0,1	0,7	-1,6	-1,1	0,3	-0,7	-1,4

bits	1	0	1	1	0	1	1

$$syndrome = bits * H^T$$

Syndrome	1	0	0

Round 2

H	1	0	0	1	1	1	0
	0	1	0	1	1	0	1
	0	0	1	1	0	1	1
v	-0,8	-0,1	-0,9	-1,2	1,1	-0,7	-0,8
beta	-0,8	0	0	-1,2	1,1	-0,7	0
	0	-0,1	0	-1,2	1,1	0	-0,8
	0	0	-0,9	-1,2	0	-0,7	-0,8
alpha	0,7	0	0	0,7	-0,7	0,8	0
	0	0,8	0	0,1	-0,1	0	0,1
	0	0	-0,7	-0,7	0	-0,8	-0,7
s	-0,1	0,7	-1,6	-1,1	0,3	-0,7	-1,4
beta	-0,8	0	0	-1,8	1	-1,5	0
	0	-0,1	0	-1,2	0,4	0	-1,5
	0	0	-0,9	-0,4	0	0,1	-0,7
alpha	1	0	0	0,8	-0,8	0,8	0
	0	0,4	0	0,1	-0,1	0	0,1
	0	0	0,1	0,1	0	-0,4	0,1
s	0,2	0,3	-0,8	-0,2	0,2	-0,3	-0,6

bits	0	0	1	1	0	1	1

Syndrome	0	0	0

Parity-Check-Matrix for WiFi (k=324, n=648)

(a) Coding rate R = 1/2.

0	-	-	-	0	0	-	-	0	-	-	0	1	0	-	-	-	-	-	-	-	-	-	-
22	0	-	-	17	-	0	0	12	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-
6	-	0	-	10	-	-	-	24	-	0	-	-	-	0	0	-	-	-	-	-	-	-	-
2	-	-	0	20	-	-	-	25	0	-	-	-	-	-	0	0	-	-	-	-	-	-	-
23	-	-	-	3	-	-	-	0	-	9	11	-	-	-	-	0	0	-	-	-	-	-	-
24	-	23	1	17	-	3	-	10	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-
25	-	-	-	8	-	-	-	7	18	-	-	0	-	-	-	-	-	0	0	-	-	-	-
13	24	-	-	0	-	8	-	6	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-
7	20	-	16	22	10	-	-	23	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-
11	-	-	-	19	-	-	-	13	-	3	17	-	-	-	-	-	-	-	-	-	0	0	-
25	-	8	-	23	18	-	14	9	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0

$P_{25}\ (Z = 27)$

$$P_0 = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}, P_1 = \begin{bmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0 \end{bmatrix}, P_5 = \begin{bmatrix} 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0 \end{bmatrix}$$

Figure 20-12—Examples of cyclic-permutation matrices with *Z*=8

Summary

- Decoding with soft bits is superior to decoding with hard bits
  - hard demodulation mean loss of information
- Convolutional codes
  - Viterbi soft-decision decoding
- LDPC codes:
  - iterative soft-decision decoding
  - soft bits are estimated independently for every parity-check equation (check node)
  - information from multiple parity-check-equations is aggregated for every soft bits
- Turbo codes:
  - high-end code in mobile communication (HSDPA, LTE)
  - iterative soft-in soft-out decoding
  - two soft-in soft-out convolutional decoders decode the information bits independently and exchange their interleaved results