Coding and error control

EP2950

KTH Technology
and Health

Outline

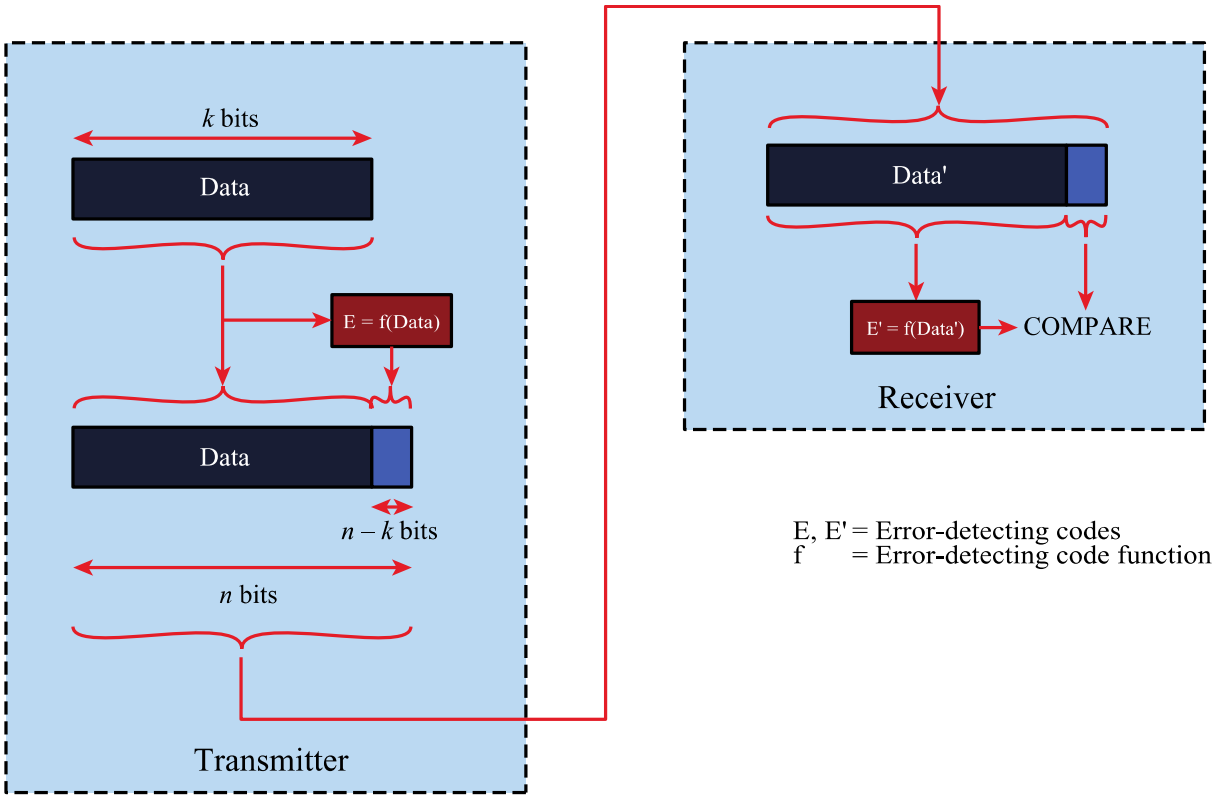
- ✓ Introduction
- ✓ Block error correction codes
- ✓ Convolutional codes

Introduction

- ✓ Link layer error control is less important in today's fixed networks
  - Low bit error ratio
  - High performance optical networks
  - Switched local area networks – no collision domain
  - Wired Ethernet
    - Error detection (CRC)
    - Leaves error control to upper layers (TCP)
    - "Over dimensioning"
  - QoS function on higher layers
- ✓ Wireless communication
  - Error control on the link layer is crucial
  - Path loss, multi-path fading, interference, topology etc

Block error correction codes

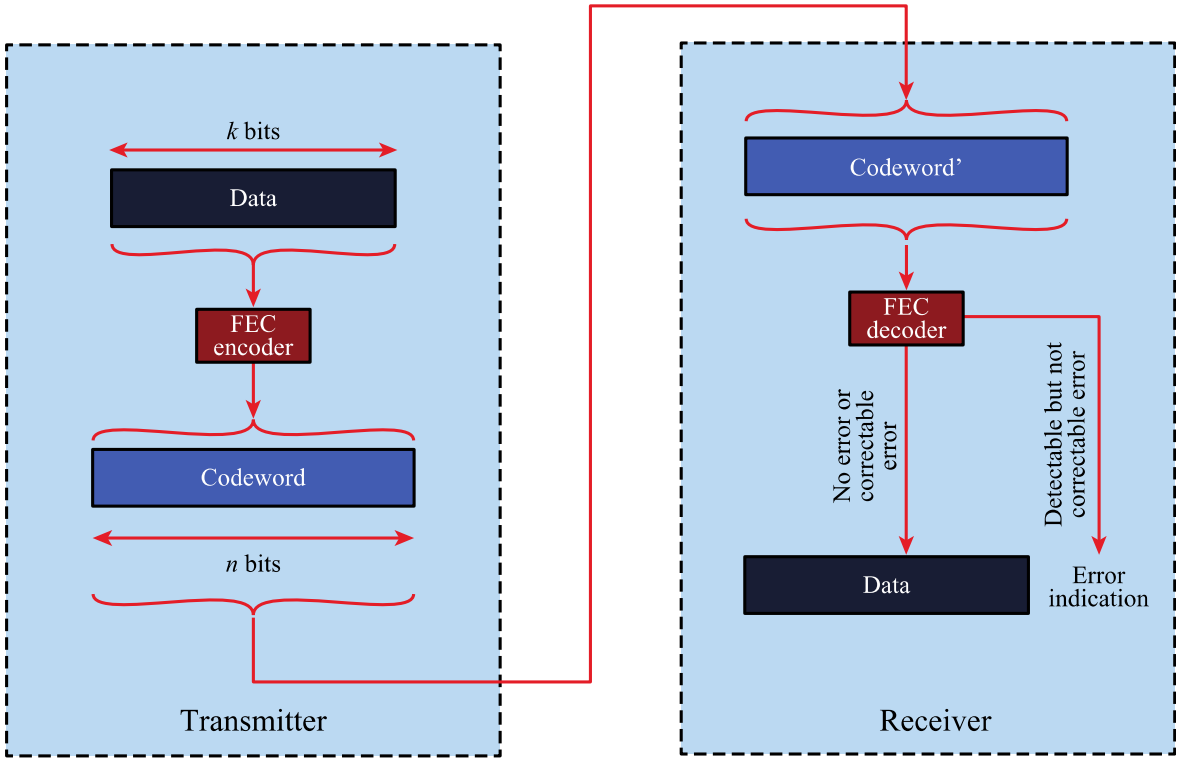
- ✓ Source encoding - minimize the number of bits to send
- ✓ Channel encoding – add bits - redundancy $(n - k)$



- ✓ Parity bit (odd/even number of ones/zeros)
  - Unable to detect an even number of bit errors
- ✓ Block parity
- ✓ Cyclic redundancy check (CRC)
  - CRC-32 in the Ethernet frame trailer
    - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + \ldots + 1$;
    - 100000100110000010001110110110111
  - Modulo 2 arithmetic (XOR)
  - Polynomials ($X^4 + X^3 + X^2 + 1 \rightarrow 11101$)
  - Digital logics (XOR gates and shift registers)
- ✓ Hamming code (linear error correcting block code)
- ✓ Cyclic codes
  - Sub class of linear codes generated and decoded as shift registers
- ✓ Convolutional codes

Forward error correction

- ✓ *k*-bit data block is mapped into *n*-bit codeword, $(n - k)$ bits added
- ✓ An (n,k) block code has 2^k valid codewords



How to decode

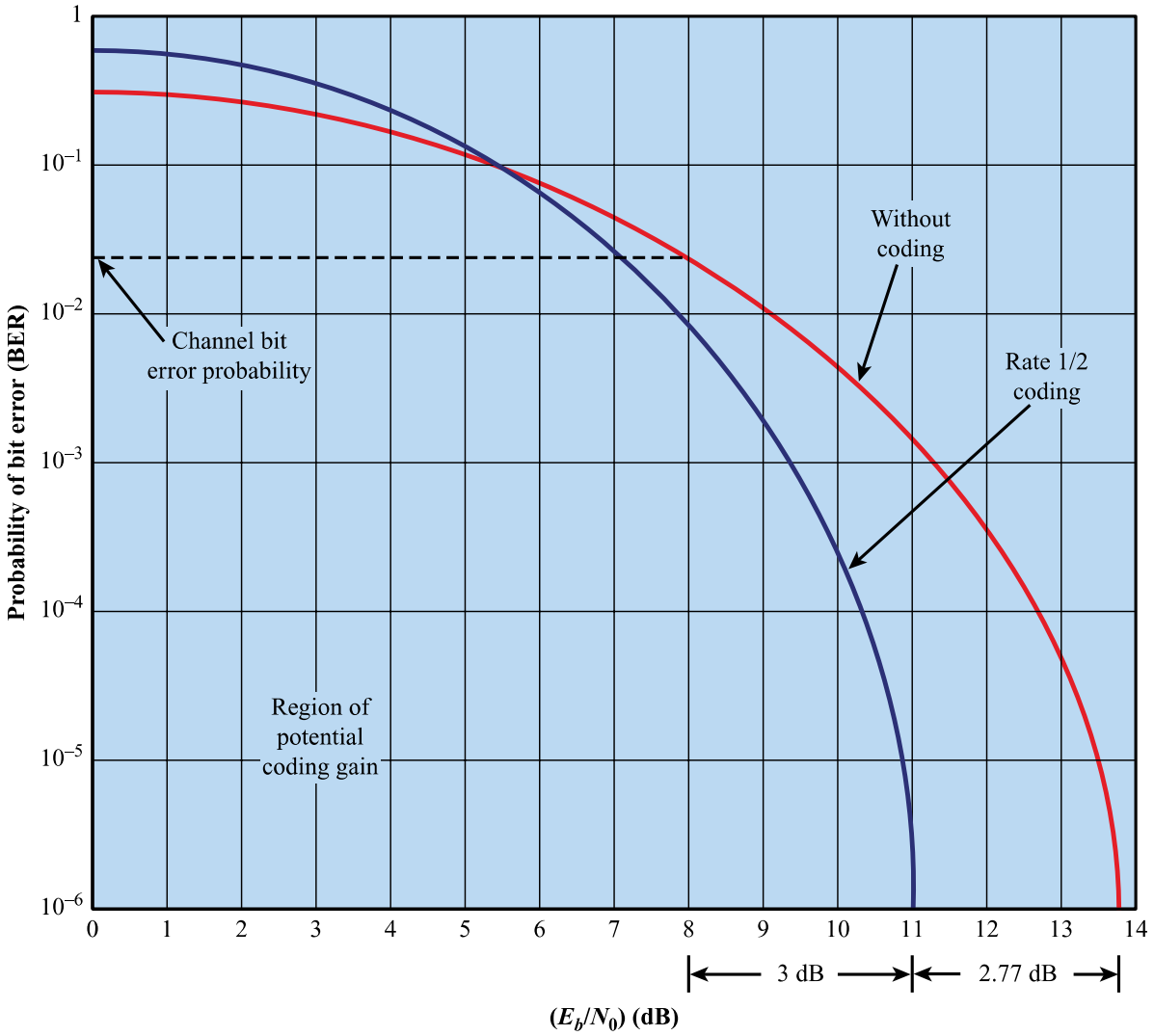
- ✓ Code table (code book)
- ✓ Received: 00100
- ✓ Not valid, error is detected
- ✓ Correction
  - One bit away from 00000
  - Two bits away from 00111
  - Three bits away from 11110
  - Four bits away from 11110
- ✓ Most likely 00000 was sent, assume data was 00
- ✓ Received: 01100
  - Two bits from 00000
  - Two bits from 11110
  - No other codes closer
  - Cannot decode. Only know bit errors are detected

Data block	Codeword
00	00000
01	00111
10	11001
11	11110

Hamming codes

- ✓ Code rate k/n, redundancy $(n-k)/k$
- ✓ Hamming distance (d) between w_1=011011; w_2=110001
  - $d(w_1, w_2)$; w_1 XOR w_2 = 101010 gives d=3
- ✓ Minimum distance between w_1, w_2, …, w_s, where $s=2^k$
  - $d_{min} = \min[d(w_i,w_j)]$, i≠j, the smallest d for all possible pairs of codewords
- ✓ To correct t bit errors requires $d_{min} \geq 2t+1$
  - $d_{min} \geq s+1$ means that s errors can be detected
- ✓ Main idea - the codewords must be "sufficiently different"
  - Let $d(w_1, w_2) = 2$, which means they are different in two bit positions. A codeword w_{error} with one bit error is received. The distance between both codewords will be the same. An error is detected but not corrected. If $d=3$, w_{error} would be closer to one of w_1 and w_2.

- ✓ Coding gain - the reduction in E_b/N_0 for a given BER



Hamming codes

- ✓ Redundancy (check or parity bits): $m = n - k$
- ✓ Minimum distance $d_{\min} = 3$
- ✓ Relation between n and k:
  - ✓ Block length: $n = 2^{n-k} - 1$
  - ✓ Number of check bits $m = n - k$: $2^{n-k} - 1 \geq n$
  - ✓ Number of data bits: $k = n - m = 2^m - 1 - m$
- ✓ Single-error correction (SEC)
- ✓ Single-error correction with double-error detection (SEC-DED)
  - An extra parity bit is needed

- ✓ Sender side
  - ✓ $(n - k)$ parity bits are generated (XOR) based on the k data bits and placed in certain bit positions
- ✓ Receiver side
  - ✓ A syndrome word identifies a bit error position using XOR between
    - The $(n - k)$ parity bits generated by the receiver
    - The received $(n - k)$ parity bits
- ✓ Requirement
  - $2^{n-k} - 1 \geq n$
  - The check (parity) bits must at least cover the range of the codeword (identify all bit positions)

- ✓ Design a parity bit pattern that covers all data bits
- ✓ Example (7,4) Hamming code
  - Check bit $C_1 = D_1 + D_2 + D_4$
  - Check bit $C_2 = D_1 + D_3 + D_4$
  - Check bit $C_3 = D_2 + D_3 + D_4$
- ✓ Receiver generates the same check bits and performs XOR with the received check bits → syndrome word
  - $S_1 = C_1 + D_1 + D_2 + D_4$
  - $S_2 = C_2 + D_1 + D_3 + D_4$
  - $S_3 = C_3 + D_2 + D_3 + D_4$
  - Syndrome words ($S_3S_2S_1$): 000 - no error, 001 – C_1 error, 010 – C_2 error, 011 – D_1 error, 100 – C_3 error, 110 means D_3 error, 111 means D_4 error
- ✓ Example 10.7 in Stallings shows (12,8) code example in detail.
  - ✓ Notice how bit positions are used

- ✓ Hamming code (12,8) transmitted block

Table 8.2 Layout of Data Bits and Check Bits (page 1 of 2)

(a) Transmitted block

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1
Transmitted Block	0	0	1	1	0	1	0	0	1	1	1	1
Codes			1010	1001		0111				0011		

(b) Check bit calculation prior to transmission

Position	Code
10	1010
9	1001
7	0111
3	0011
XOR = C8 C4 C2 C1	0111

- ✓ Hamming code (12,8) received block

Table 8.2 Layout of Data Bits and Check Bits (page 2 of 2)

(c) Received block

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1
Received Block	0	0	1	1	0	1	1	0	1	1	1	1
Codes			1010	1001		0111	0110			0011		

(d) Check bit calculation after reception

Position	Code
Hamming	0111
10	1010
9	1001
7	0111
6	0110
3	0011
XOR = syndrome	0110

- Check (parity bits)
  - C1 in bit position 1 - 0001
  - C2 in bit position 2 - 0010
  - C3 in bit position 4 - 0100
  - C4 in bit position 8 - 1000
  - C1 is parity bit for all bit positions xxx1 (bit positions 1, 3, 5, 7, 9, 11)
  - C2 is parity bit for all bit positions xx1x (bit positions 2, 3, 6, 7, 10, 11)
  - C3 is parity bit for all bit positions x1xx (bit positions 4, 5, 6, 7, 12)
  - C4 is parity bit for all bit positions 1xxx (bit positions 8, 9, 10, 11, 12)
- Apply even parity
  - XOR of the bit positions for the 1-set bits gives the parity bits C4C3C2C1 (see Table 10.2b in Stallings)
- Receiver calculates the syndrome word
  - XOR of bit positions for the 1-set bits and the check bits (Table 10.2d) - filter out possible bit error positions
  - Syndrome word identifies the position of a bit error

Cyclic codes

- ✓ Can be implement as linear feedback shift register
- ✓ Examples: BCH codes, Reed-Solomon
- ✓ Procedure
  - ✓ D(x) *k* bit data block
  - ✓ P(x) code generating polynomial
  - ✓ T(x) *n* bits sent
  - ✓ Code bit generation and transmitted message
    - ✓ $X^{n-k}D(x)/P(x)=Q(x)+C(x)/P(x)$
    - ✓ $T(x)=X^{n-k}D(x)+C(x)$
  - ✓ Z(x) received message, E(x) errors, S(x) syndrome
    - ✓ $Z(x)=T(x)+E(x)$
    - ✓ $Z(x)/P(x)=B(x)+S(x)/P(x)$

- ✓ Example 10.8. (7,4) code with $P(x)=x^3+x^2+1$ (1101) and $D=x^3+x$ (1010)
- ✓ $X^{n-k}D(x)/P(x)$ gives remainder $C(x)=1$ (001) and $T(x)= x^6+x^4+1$ (1010001)
- ✓ Table of error pattern and syndrome can be calculated
- ✓ Received message Z(x) divided by P(x) gives syndrome and error pattern which identifies the position of the bit error

(a) Table of valid codewords

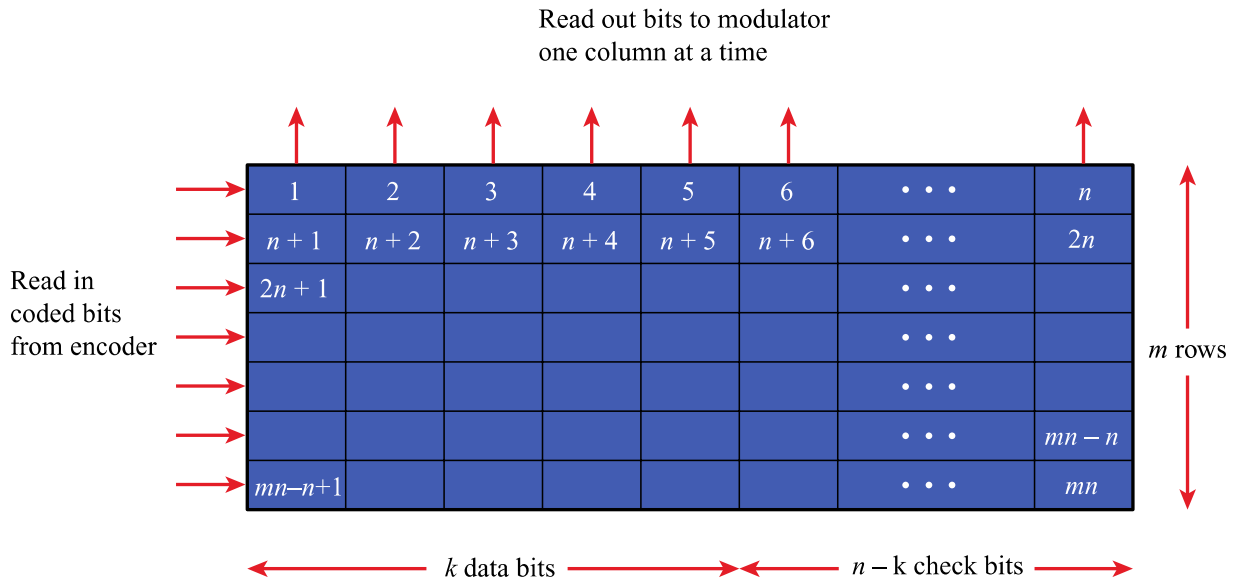
Data Block	Codeword
0000	0000000
0001	0001101
0010	0010111
0011	0011010
0100	0100011
0101	0101110
0110	0110100
0111	0111001
1000	1000110
1001	1001011
1010	1010001
1011	1011100
1100	1100101
1101	1101000
1110	1110010
1111	1111111

(b) Table of syndromes for single-bit errors

Error pattern E	Syndrome S
0000001	001
0000010	010
0000100	100
0001000	101
0010000	111
0100000	011
1000000	110

Interleaving

- ✓ An additional method to deal with long bursts of bit errors
- ✓ Read and write data from memory in different order
- ✓ Store data in blocks (rows) of n bits (bit 1, 2, …, n, n+1, …)
- ✓ Send data column by column (bit 1, n+1, 2n+1, …)

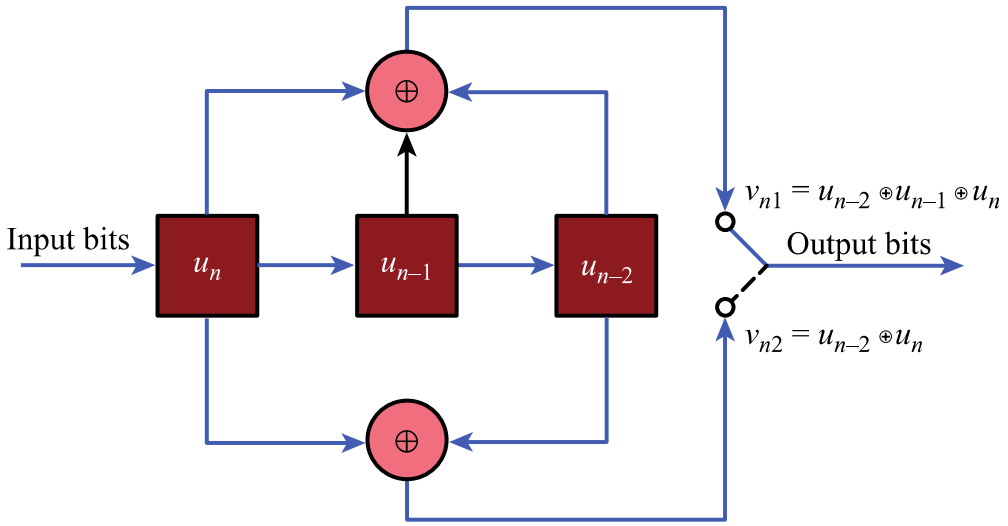


Note: The numbers in the matrix indicate the order in which bits are read in.
Interleaver output sequence: 1, n + 1, 2n + 1, …

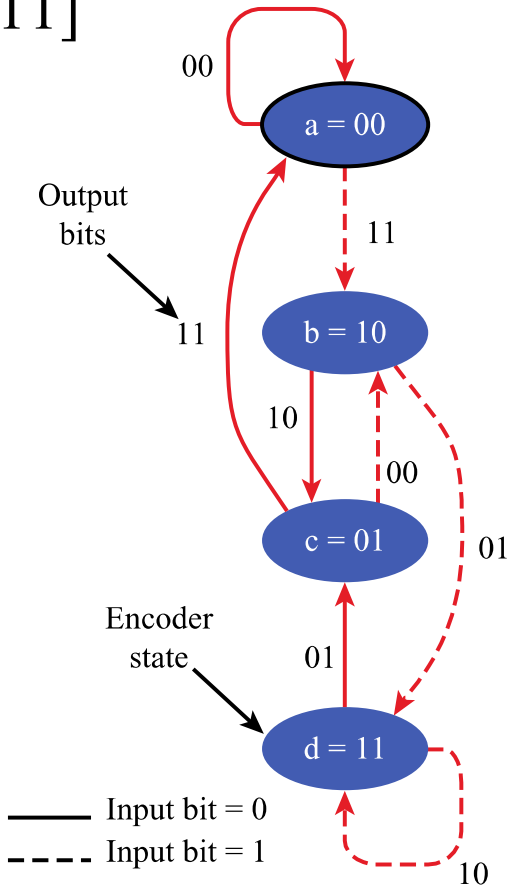
Convolutional codes

- ✓ Add redundant bits in a continuous stream
- ✓ Finite state machine with memory cells
- ✓ Minimize bit errors in noisy channels (not for long error bursts)
- ✓ (n, k, K) code
  - Input of k bits data block
  - Output of n bits for each k bits
  - Constraint factor K
  - k/n coding rate
- The n bit output determined by k input and K-1 previous values kept in memory cells
- ✓ (n, k, K) = (2,1,3)
  - Two output bits, one input bit , two memory cells, ½ rate

- ✓ $2^{k\times(K-1)}$ different states of the memory cells (u_{n-1}, u_{n-2})
- ✓ (2,1,3) gives $2^{1\times(3-1)} = 4$ states – [00,10,01,11]
- ✓ v_{n1}= mod2(u_{n-2}+u_{n-1}+u_n), G_{p1}=[111]
- ✓ v_{n2}= mod2(u_{n-2}+u_n), G_{p2}=[101]



(a) Encoder shift register



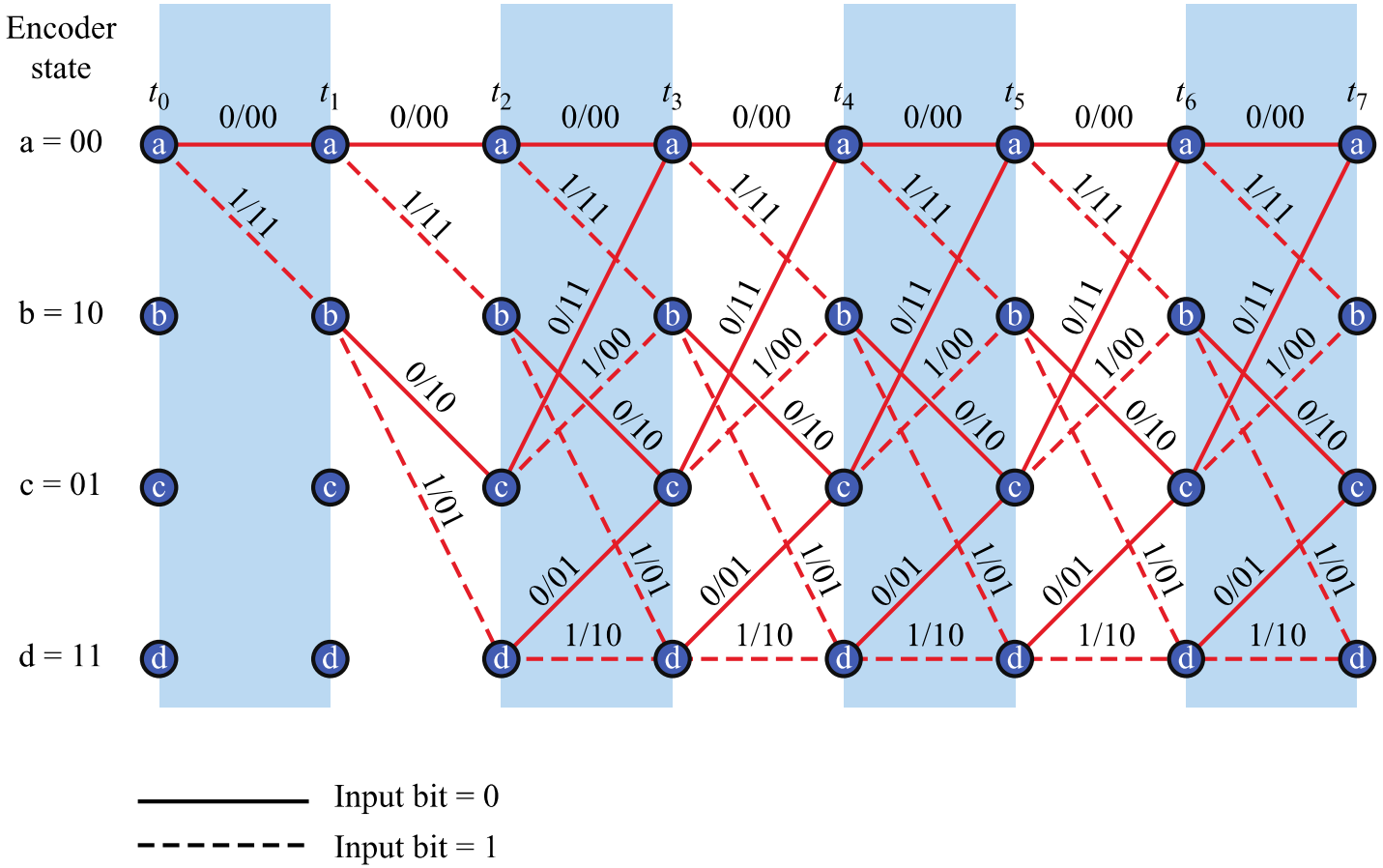
(b) Encoder state diagram

- ✓ Table representation for $(n, k, K) = (2, 1, 3)$
  - States and transitions for input bit = 0 or 1
  - Output from encoder
  - Generator polynomials G_{p1}=[111] and G_{p2}=[101]

Current state	Next state if Input=0	Next state if Input=1	Output if Input=0	Output if Input=1
00	00	10	00	11
01	00	10	11	00
10	01	11	10	01
11	01	11	01	10

Trellis time-state diagram

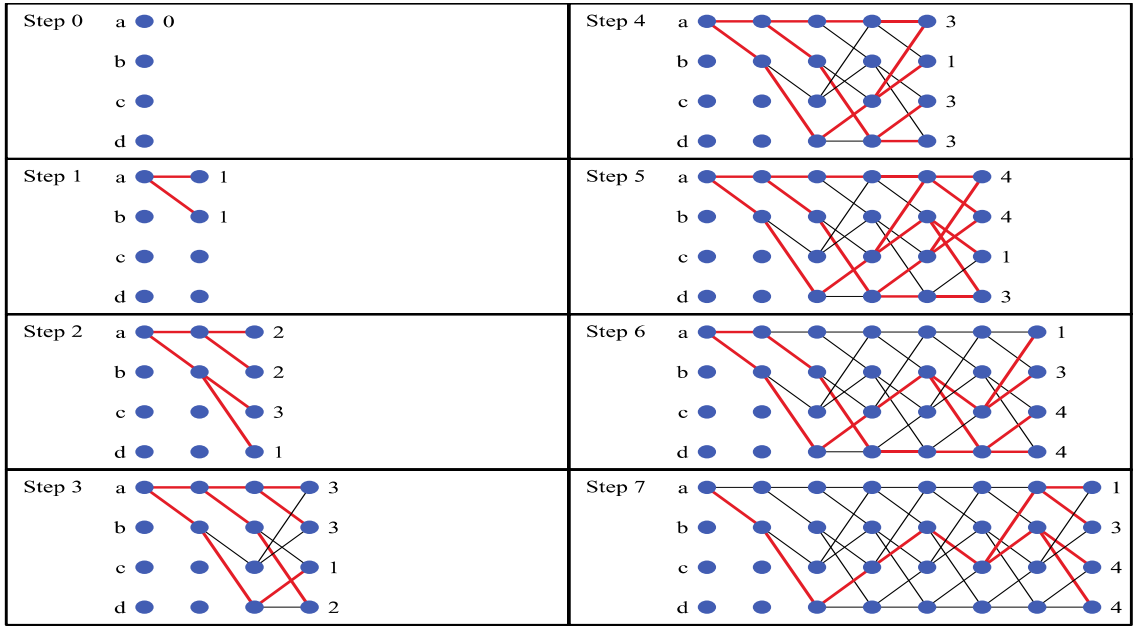
- Dotted line – input=1, solid line – input=0, output along the lines
- Input: 1011000 gives 1/11, 0/10, 1/00, 1/01, 0/01, 0/11, 0/00
- End with K-1 zeros as input to retain zeros in all cells – tail bits

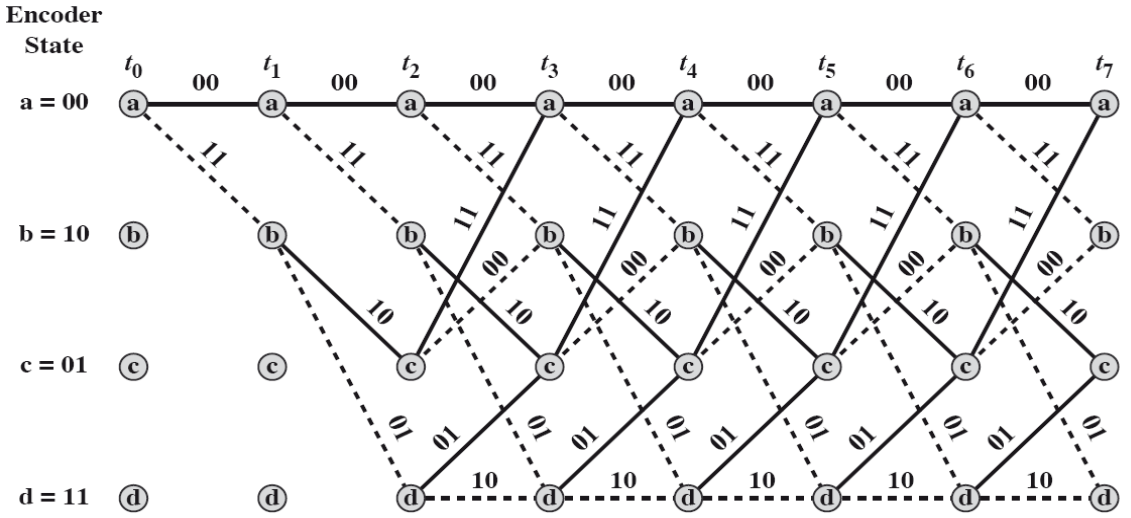


Viterbi decoding

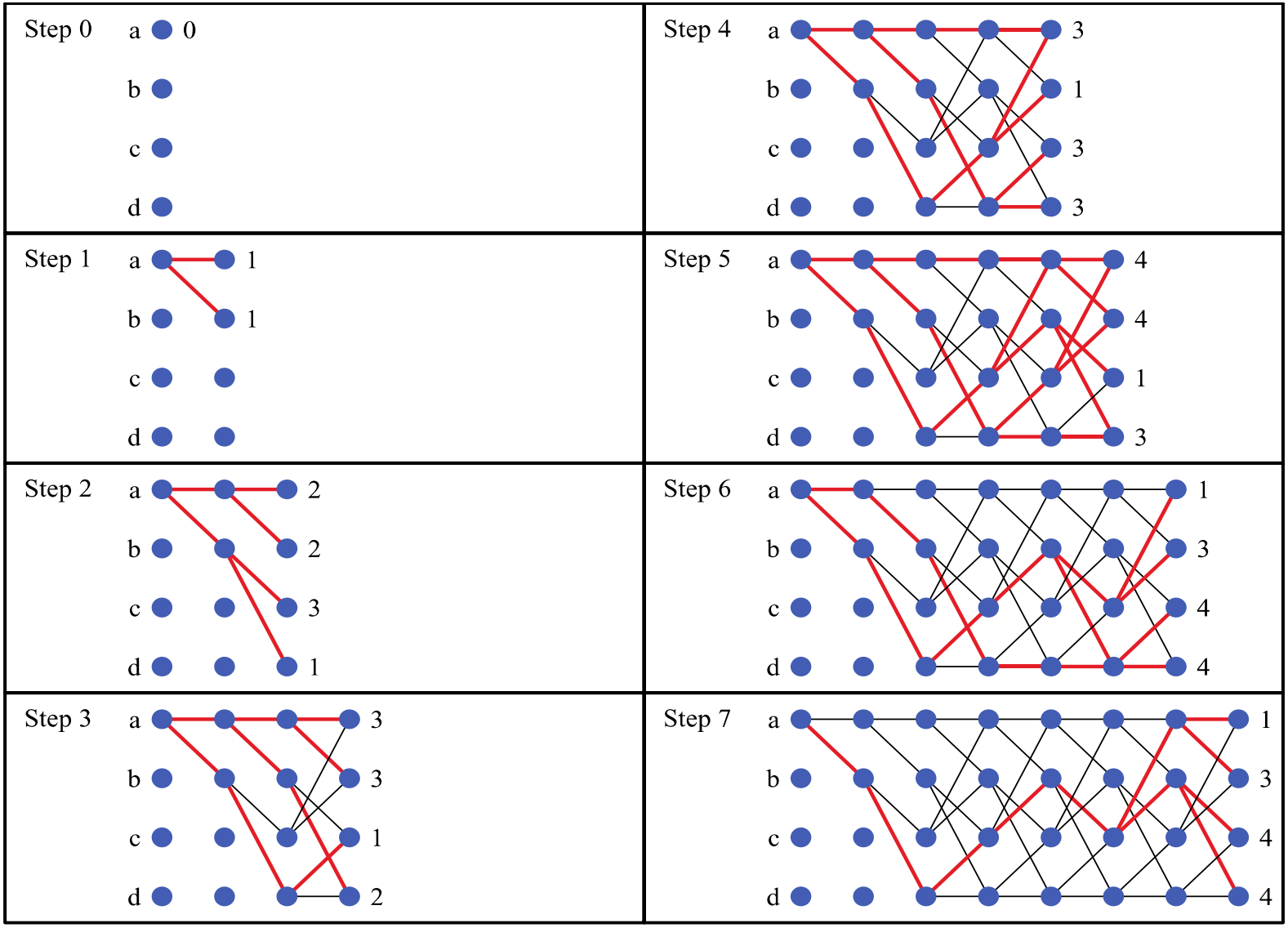
- ✓ Maximum likelihood decoding
  - Find the most likely sent codes - minimizes the errors
- ✓ Compare the received code with all possible sent codes
- ✓ Use the Hamming distance as an error metric
- ✓ Choose a path through the Trellis diagram that minimizes the Hamming distance
- ✓ Compare to the least-cost (or shortest-path) routing algorithms in previous courses

Step 1:
w=10
a: a→a; *dist*=1
b: a→b; *dist*=1
Step 2:
w=10 01
a: a→a→a; *dist*=1+1=2
b: a→a→b; *dist*=1+1=2
c: a→b→c; *dist*=1+2=3
d: a→b→d; *dist*=1+0=1
Step 3:
w=10 01 01
a: a→a→a→a; *dist*=1+1+1=3
b: a→a→a→b; *dist*=1+1+1=3
c: a→b→d→c; *dist*=1+0+0=1
d: a→a→b→d; *dist*=1+1+0=2





w=10 01 01 00 10 11 00



w=10 01 01 00 10 11 00

More on encoding

- ✓ Possible topics for short-paper
  - ✓ Not included in written exam
- ✓ Punctured convolutional codes
- ✓ Soft decision coding
- ✓ Turbo codes
- ✓ Hybrid Automatic Repeat Request (HARQ)