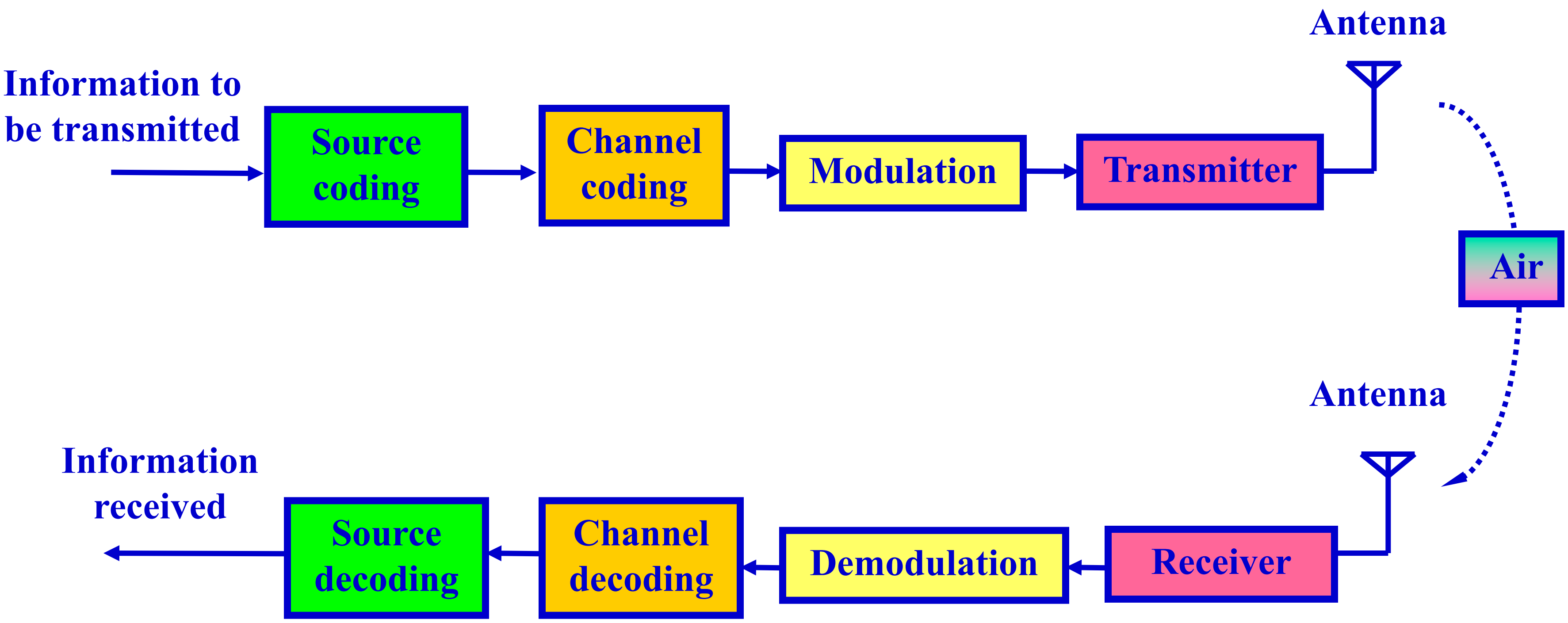
Chapter 4

Channel Coding and Error Control

Outline

- Introduction
- Block Codes
- Cyclic Codes
- CRC (Cyclic Redundancy Check)
- Convolutional Codes
- Interleaving
- Information Capacity Theorem
- Turbo Codes
- ARQ (Automatic Repeat Request)
  - Stop-and-wait ARQ
  - Go-back-N ARQ
  - Selective-repeat ARQ

Introduction



Forward Error Correction (FEC)

- The key idea of FEC is to transmit enough redundant data to allow receiver to recover from errors all by itself. No sender retransmission required
- A simple redundancy is to attach a parity bit
- 10101100
- The major categories of FEC codes are
  - Block codes
  - Cyclic codes
  - Reed-Solomon codes (Not covered here)
  - Convolutional codes, and
  - Turbo codes, etc.

FEC Code Principle

- Hamming distance(HD): the number of different bits for 2 n-bit binary sequences.
- e.g., v_1= 011011; v_2=110001; $d(v_1, v_2) = 3$
- We can correct 1-bit error or detect 2-bit error if the HD of any two codewords is larger than two.
  - Example:

Data bits	Codeword
00	00000
01	00111
10	11001
11	11110

Hamming Code Generation

- If we want to transmit *k* data bits, we can add *n-k* check bits to form an *n* bits Hamming code.
- Hamming check bits are inserted at the position of power of 2 i.e., positions 1, 2, 4, …, $2^{(n-k-1)}$
- The remaining *k* bits are data bits
- Each data position which has a value 1 is presented by a binary value equal to its position; thus if the 9th bit is 1 the corresponding value is 1001 (assume *check bits* = 4)
- All of the position values are then XORed together to produce the bits of the Hamming code
  - Example: The 8-bit data block is 00111001 and check bits = 4

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

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

V

Hamming Code Process

- Encoding: k data bits + $(n - k)$ check bits
- Decoding: compares received $(n - k)$ bits with calculated $(n - k)$ bits using XOR
  - Resulting $(n - k)$ bits called *syndrome word*
  - Syndrome range is between 0 and $2^{(n-k)} - 1$
  - Each bit of syndrome indicates a match (0) or conflict (1) in that bit position and hence

    $2^{(n-k)} - 1 >= k + (n - k) = n$

Hamming Code Characteristics

- We would like to generate a syndrome with the following characteristics:
  - If the syndrome contains all 0s, no error has been detected
  - If the syndrome contains only one bit set to 1, then an error has occurred in one of the check bits
  - If the syndrome contains more than one bit set to 1, then the numerical value of the syndrome indicates the position of the data bit in error

Table 8.1 Hamming Code Requirements

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

We can add one additional bit to detect double bits error.
The extra bit is a parity bit over the entire code block.

Linear Block Codes

- **Information is divided into blocks of length k**
- **r parity bits or check bits are added to each block (total length $n = k + r$)**
- **Code rate $R = k/n$**
- **An (n, k) block code is said to be linear if the vector sum of two codewords is a codeword**
- **Tradeoffs between**
  - **Efficiency**
  - **Reliability**
  - **Encoding/Decoding complexity**
- **All arithmetic is performed using Modulo 2 Addition**

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ \underline{0} & \underline{1} & \underline{0} & \underline{1} \\ 0 & 1 & 1 & 0 \end{array}$$

Linear Block Codes

- The uncoded k data bits be represented by the m vector:

$$m=(m_1, m_2, \ldots, m_k)$$

The corresponding codeword be represented by the n-bit c vector:

$$c=(c_1, c_2, \ldots c_k, c_{k+1}, \ldots, c_{n-1}, c_n)$$

- Each parity bit consists of weighted modulo 2 sum of the data bits represented by $\oplus$ symbol for Exclusive OR or modulo 2-addition

Linear Block Codes

k- data and $r = n-k$ redundant bits

$$\begin{cases} c_1 = m_1 \\ c_2 = m_2 \\ \ldots \\ c_k = m_k \\ c_{k+1} = m_1 p_{1(k+1)} \oplus m_2 p_{2(k+1)} \oplus \ldots \oplus m_k p_{k(k+1)} \\ \ldots \\ c_n = m_1 p_{1n} \oplus m_2 p_{2n} \oplus \ldots \oplus m_k p_{kn} \end{cases}$$

p_{ij} ($i = 1, 2, \ldots, k$; $j = k+1, k+2, \ldots, n$) is the binary weight of the particular data bit.

Linear Block Codes

- **Linear Block Code**

The code vector c of the Linear Block Code is

$$c = m\ G,$$

where m is the message vector, G is the generator matrix.

$$G = [\mathbf{I}_k \mid P]_{k\times n},$$

where p_i = Remainder of $[x^{n-k+i-1}/g(x)]$ for i =1, 2, .., k, and I is unit or identity matrix.

- **At the receiving end, parity check matrix can be given as:**

$H = [P^T \mid \mathbf{I}_{n-k}]$, where P^T is the transpose of the matrix P.

Linear Block Codes: Example

Example: Find linear block code encoder G if code generator polynomial $g(x)=1+x+x^3$ for a (7, 4) code; n = total number of bits = 7, k = number of information bits = 4, r = number of parity bits = n - k = 3

Coefficients of $x^0x^1x^2$

$$p_1 = \mathrm{Re}\left[\frac{x^3}{1+x+x^3}\right] = 1+x \rightarrow [110]$$

$$p_2 = \mathrm{Re}\left[\frac{x^4}{1+x+x^3}\right] = x+x^2 \rightarrow [011]$$

$$p_3 = \mathrm{Re}\left[\frac{x^5}{1+x+x^3}\right] = 1+x+x^2 \rightarrow [111]$$

$$p_4 = \mathrm{Re}\left[\frac{x^6}{1+x+x^3}\right] = 1+x^2 \rightarrow [101]$$

$$G = \begin{bmatrix} 1000 | 110 \\ 0100 | 011 \\ 0010 | 111 \\ 0001 | 101 \end{bmatrix} = [I \mid P]$$

I is the identity matrix
P is the parity matrix

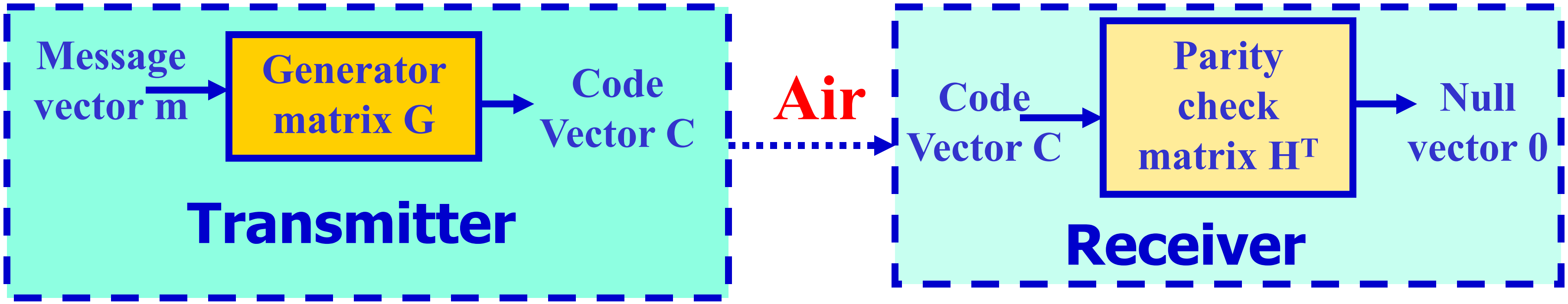
Linear Block Codes: Example

- **The Generator Polynomial can be used to determine the Generator Matrix *G* that allows determination of parity bits for a given data bits of *m* by multiplying as follows:**

$$c = m.G = \underbrace{[1011]}_{\text{Data}} \begin{bmatrix} 1000110 \\ 0100011 \\ 0010111 \\ 0001101 \end{bmatrix} = [\underbrace{1011}_{\text{Data}} \mid \underbrace{100}_{\text{Parity}}]$$

- **Other combinations of *m* can be used to determine all other possible codewords**

Block Codes: Linear Block Codes



Operations of the generator matrix and the parity check matrix

- **Consider a (7, 4) linear block code, given by *G* as**

$$G = \begin{bmatrix} 1000 \mid 110 \\ 0100 \mid 011 \\ 0010 \mid 111 \\ 0001 \mid 101 \end{bmatrix}$$

$$H^T = \begin{bmatrix} P \\ I_{n-k} \end{bmatrix} = \begin{bmatrix} 110 \\ 011 \\ 111 \\ 101 \\ \hline 100 \\ 010 \\ 001 \end{bmatrix}$$

For convenience, the code vector is expressed as

$c = \lfloor m \mid c_p \rfloor$ **Where** $c_p = mP$ **is an (*n*-*k*)-bit parity check vector**

Block Codes: Linear Block Codes

Received code vector $x = c \oplus e$, here e is an error vector, the matrix H^T has the property

$$cH^T = \begin{bmatrix} m \mid c_p \end{bmatrix} \begin{bmatrix} P \\ I_{n-k} \end{bmatrix}$$

$$= mP \oplus c_p = c_p \oplus c_p = 0$$

Block Codes: Linear Block Codes

- ✓ **The transpose of matrix H^T is** $H = \left[P^T \middle| I_{n-k}\right]$

- ✓ **I_{n-k} is a n-k by n-k unit matrix and P^T is the transpose of parity matrix P.**

- ✓ **H is called parity check matrix.**

- ✓ **Compute syndrome as $s = x\,H^T = (c \oplus e) * H^T$**

  $= cH^T \oplus eH^T = eH^T$

Linear Block Codes

- If s is 0 then message is correct else there are errors in it, from common known error patterns the correct message can be decoded.

  - For the (7, 4) linear block code, given by G as

$$G = \begin{bmatrix} 1000 \mid 111 \\ 0100 \mid 110 \\ 0010 \mid 101 \\ 0001 \mid 011 \end{bmatrix} \qquad H = \begin{bmatrix} 1110 \mid 100 \\ 1101 \mid 010 \\ 1011 \mid 001 \end{bmatrix}$$

- For $m = [1\ 0\ 1\ 1]$ and $c = mG = [1\ 0\ 1\ 1 \mid 0\ 0\ 1]$. If there is no error, the received vector $x = c$, and $s = cH^T = [0, 0, 0]$

Linear Block Codes

- **Let c suffer an error such that the received vector $x = c \oplus e$ = [1 0 1 1 0 0 1] $\oplus$ [0 0 1 0 0 0 0] =[1 0 0 1 0 0 1]**

Then, syndrome $s = xH^T$

$$= \begin{bmatrix} 1001 \mid 001 \end{bmatrix} \begin{bmatrix} 111 \\ 110 \\ 101 \\ 011 \\ -- \\ 100 \\ 010 \\ 001 \end{bmatrix} = [101] = (eH^T)$$

- **This indicates error position, giving the corrected vector as [1011001]**

Cyclic Codes

- It is a block code which uses a shift register to perform encoding and decoding the codeword with n bits is expressed as:

$$c(x) = c_1 x^{n-1} + c_2 x^{n-2} \ldots + c_k x^{n-k} \ldots + c_n$$

where each coefficient c_i ($i = 1, 2,.., n$) is either a 1 or 0

- The codeword can be expressed by the data polynomial $m(x)$ and the check polynomial $c_p(x)$ as

$$c(x) = m(x)\, x^{n-k} + c_p(x)$$

where $c_p(x)$ = remainder from dividing $m(x)\, x^{n-k}$ by generator $g(x)$

Cyclic Codes

- **Let $m(x)$ be the data block and $g(x)$ be the polynomial divisor, we have $x^{n-k}\, m(x)/g(x) = q(x) + c_p(x)/g(x)$**

  The transmitted block is $c(x) = x^{n-k}\, m(x) + c_p(x) \rightarrow$

  $c(x)/g(x) = q(x)$
- **If there are no errors the division of $c(x)$ by $g(x)$ produces no remainder.**
- **If one or more bit errors, then the received block $c'(x)$ will be of the form $c'(x) = c(x) + e(x)$ and the error pattern is detected from known error syndromes $s(x) = e(x)/g(x)$**
- **The syndrome value $s(x)$ only depends on the error bits**
  - **To be able to correct all single and double bits errors the relationship is $(n + n(n\text{-}1)/2) \le (2^{n-k} - 1)$**

Cyclic Code: Example

- **Example : Find the codeword $c(x)$ if $m(x) = 1 + x + x^2$ and $g(x) = 1 + x + x^3$, for (7, 4) cyclic code**
- **We have n = total number of bits = 7, k = number of information bits = 4, r = number of parity bits = $n - k = 3$**

$$\because \quad c_p(x) = rem\left[\frac{m(x)x^{n-k}}{g(x)}\right]$$

$$= rem\left[\frac{x^5 + x^4 + x^3}{x^3 + x + 1}\right] = x$$

Then,

$$c(x) = m(x)x^{n-k} + c_p(x) = x + x^3 + x^4 + x^5$$

= 0111010

Cyclic Code: Example

- Example : Let $m(x) = 1 + x + x^2$ and $g(x) = 1 + x + x^3$, for (7, 4) cyclic code
- Assume $e(x)$ = 1000000. The received block $c'(x)$ = 1111010
- We have $s(x) = e(x)/g(x) = x^2 + 1$. Therefore, s = 101. According to Table 1(b), we have the error pattern 1000000
- Now, supposed the received block is 0111011, or $c'(x) = x^5 + x^4 + x^3 + x + 1$. Find $s(x)$ and the error pattern.

Table 1: A Single-Error-Correcting (7,4) Cyclic Code

(a) Table of valid codewords

Data Block	Codeword
0000	0000000
0001	0001011
0010	0010110
0011	0011101
0100	0100111
0101	0101100
0110	0110001
0111	0111010
1000	1000101
1001	1001110
1010	1010011
1011	1011000
1100	1100010
1101	1101001
1110	1110100
1111	1111111

(b) Table of syndromes for single-bit errors

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

Cyclic Redundancy Check (CRC)

- Cyclic Redundancy Code (CRC) is an error-checking code
- The transmitter appends an extra *n*-*k*-bit sequence to every frame called Frame Check Sequence (FCS). The FCS holds redundant information about the frame that helps the receivers detect errors in the frame
- Transmitter: For a *k*-bit block, transmitter generates an (*n*-*k*)-bit frame check sequence (FCS). Resulting frame of *n* bits is exactly divisible by predetermined number
- Receiver: Divides incoming frame by predetermined number. If no remainder, assumes no error

Cyclic Redundancy Check (CRC)

- **Generator polynomial is divided into the message polynomial, giving quotient and remainder, the coefficients of the remainder form the bits of final CRC**
- **Define:**

Q – The original frame (k bits) to be transmitted

F – The resulting frame check sequence (FCS) of n-k bits to be added to Q (usually n = 8, 16, 32)

J – The cascading of Q and F

P – The predefined CRC generating polynomial

The main idea in CRC algorithm is that the FCS is generated so that J should be exactly divisible by P

Cyclic Redundancy Check (CRC)

- **The CRC creation process is defined as follows:**
  - **Get the block of raw message**
  - **Left shift the raw message by *n-k* bits and then divide it by *P***
  - **Get the remainder *R* as FCS**
  - **Append the *R* to the raw message. The result *J* is the frame to be transmitted** $J = Q \cdot x^{n-k} + F\ (= R)$
  - **J should be exactly divisible by P**
- **Dividing $Q \cdot x^{n-k}$ by P gives $Q \cdot x^{n-k}/P = Q' + R/P$**
  - **Where *R* is the reminder**
  - **$J = Q \cdot x^{n-k} + R$. This value of J should yield a zero reminder for J/P**

Common CRC Codes

Code-Parity check bits	Generator polynomial $g(x)$
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
CRC-CCITT	$x^{16}+x^{12}+x^5+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

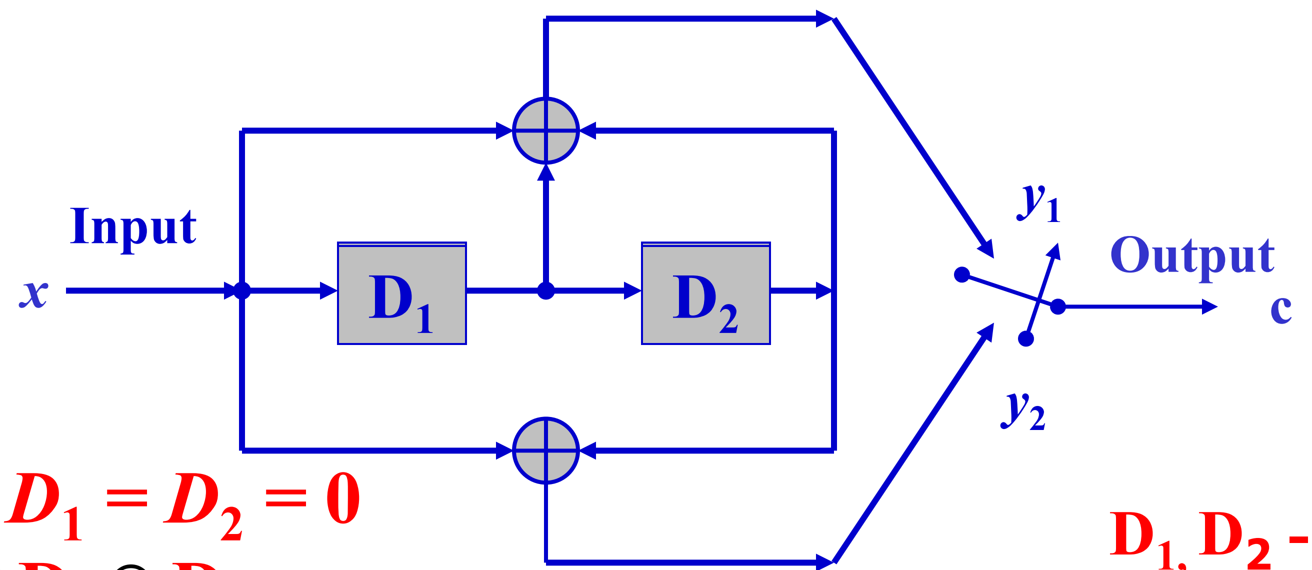
CRC Using Polynomials

- **All of the following errors are not divisible by a suitably chosen $g(x)$**
  - **All single-bit errors, if $g(x)$ has more than one nonzero term**
  - **All double-bit errors, if $g(x)$ has a factor with three terms**
  - **Any odd number of errors, as long as $g(x)$ contains a factor $(x+1)$**
  - **Any burst errors with length is less than or equal to $n - k$**
  - **A fraction of error bursts of length $n - k + 1$; the fraction equals $1 - 2^{-(n-k-1)}$**
  - **A fraction of error bursts of length greater than $n - k + 1$; the fraction equals $1 - 2^{-(n-k)}$**

Convolutional Codes

- **Most widely used channel code**
- **Encoding of information stream rather than information blocks**
- **Decoding is mostly performed by the Viterbi Algorithm (not covered here)**
- **The output constraint length K for a convolution code is defined as $K = M + 1$ where M is the maximum number of stages in any shift register**
- **The code rate r is defined as $r = k/n$ where k is the number of parallel information bits and n is the the number of parallel output encoded bits at one time interval**
- **A convolution code encoder with n=2 and k=1 or code rate r = 1/2 is shown next**

Convolutional Codes: ($n = 2$, $k = 1$, $M = 2$) Encoder



Initial: $D_1 = D_2 = 0$

$y_1 = x \oplus D_1 \oplus D_2$

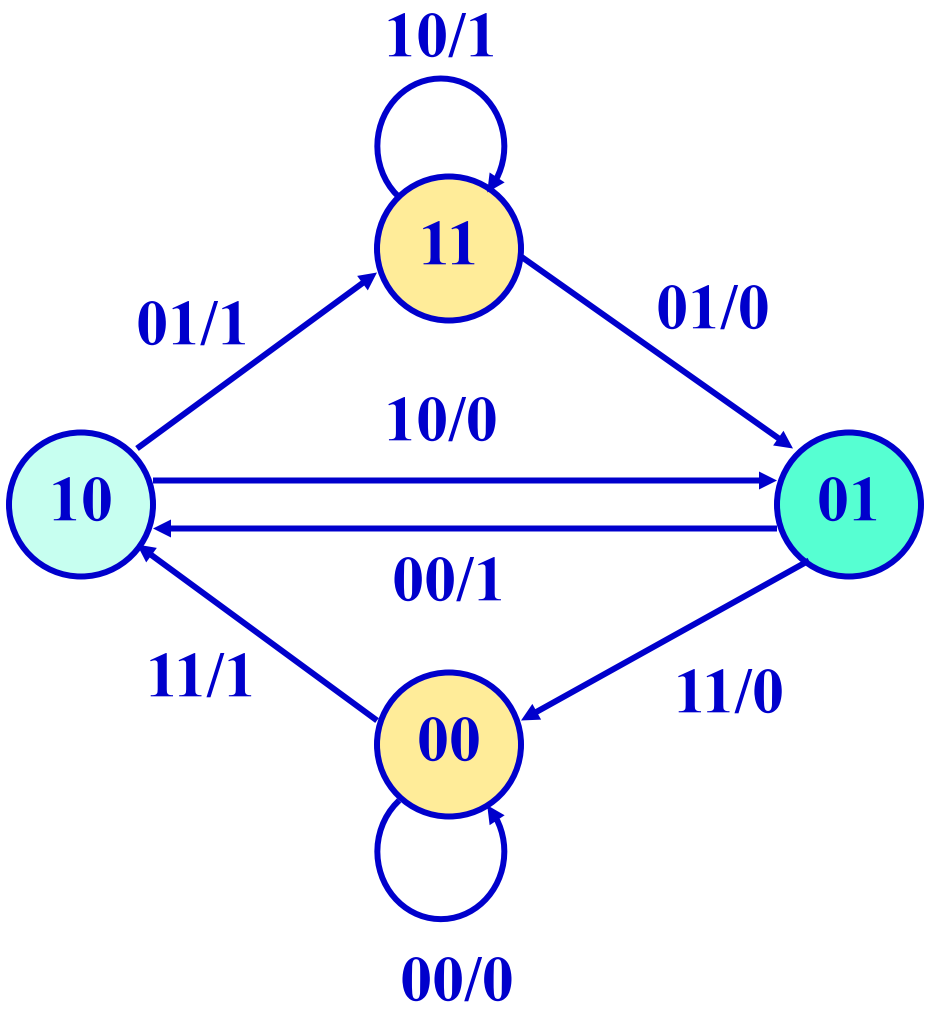
$y_2 = x \oplus D_2$

D_1, D_2 - Registers

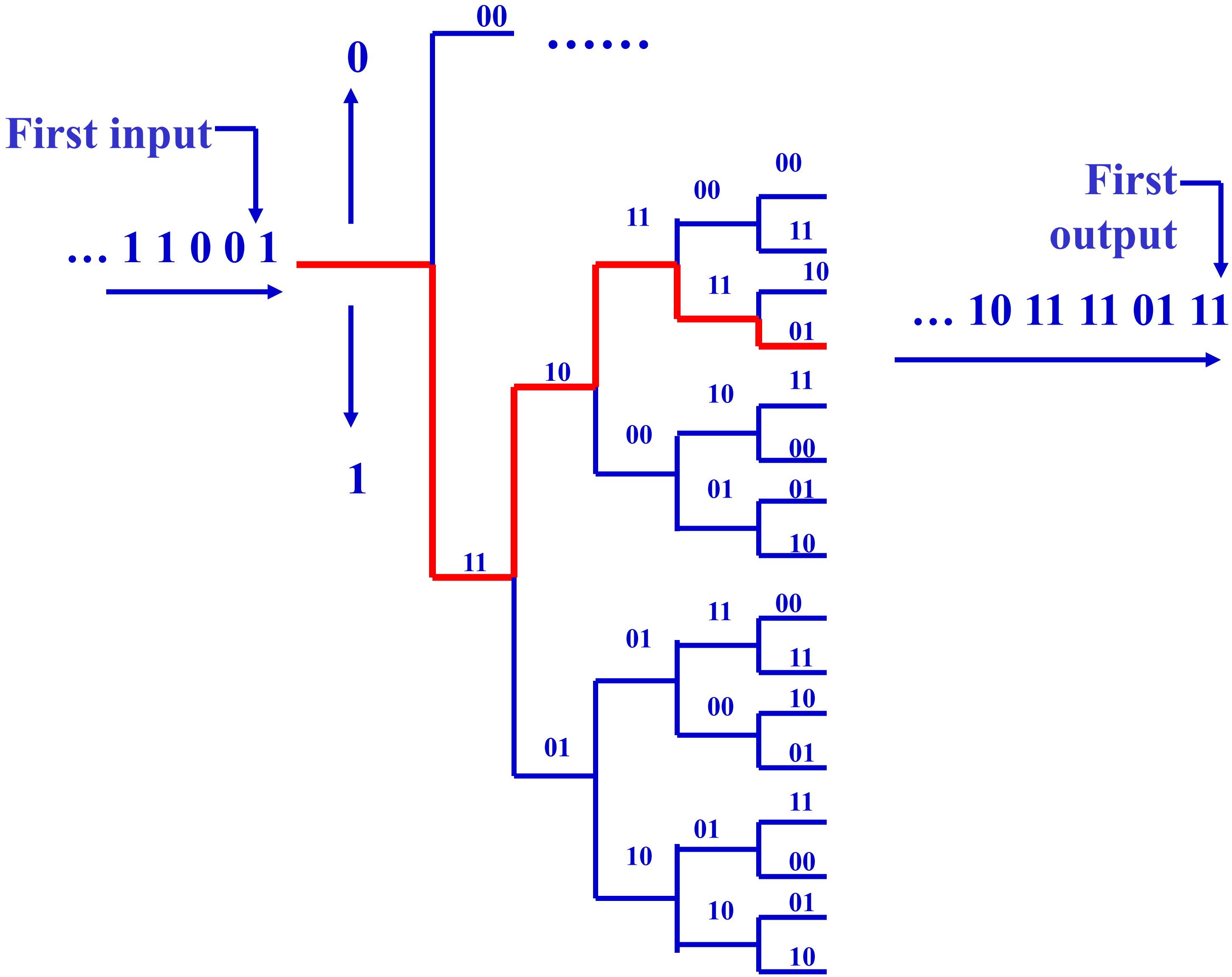
Input x:	1	1	1	0	0	0	…
Output y_1, y_2:	11	01	10	01	11	00	…

Input x:	1	0	1	0	0	0	…
Output y_1, y_2:	11	10	00	10	11	00	…

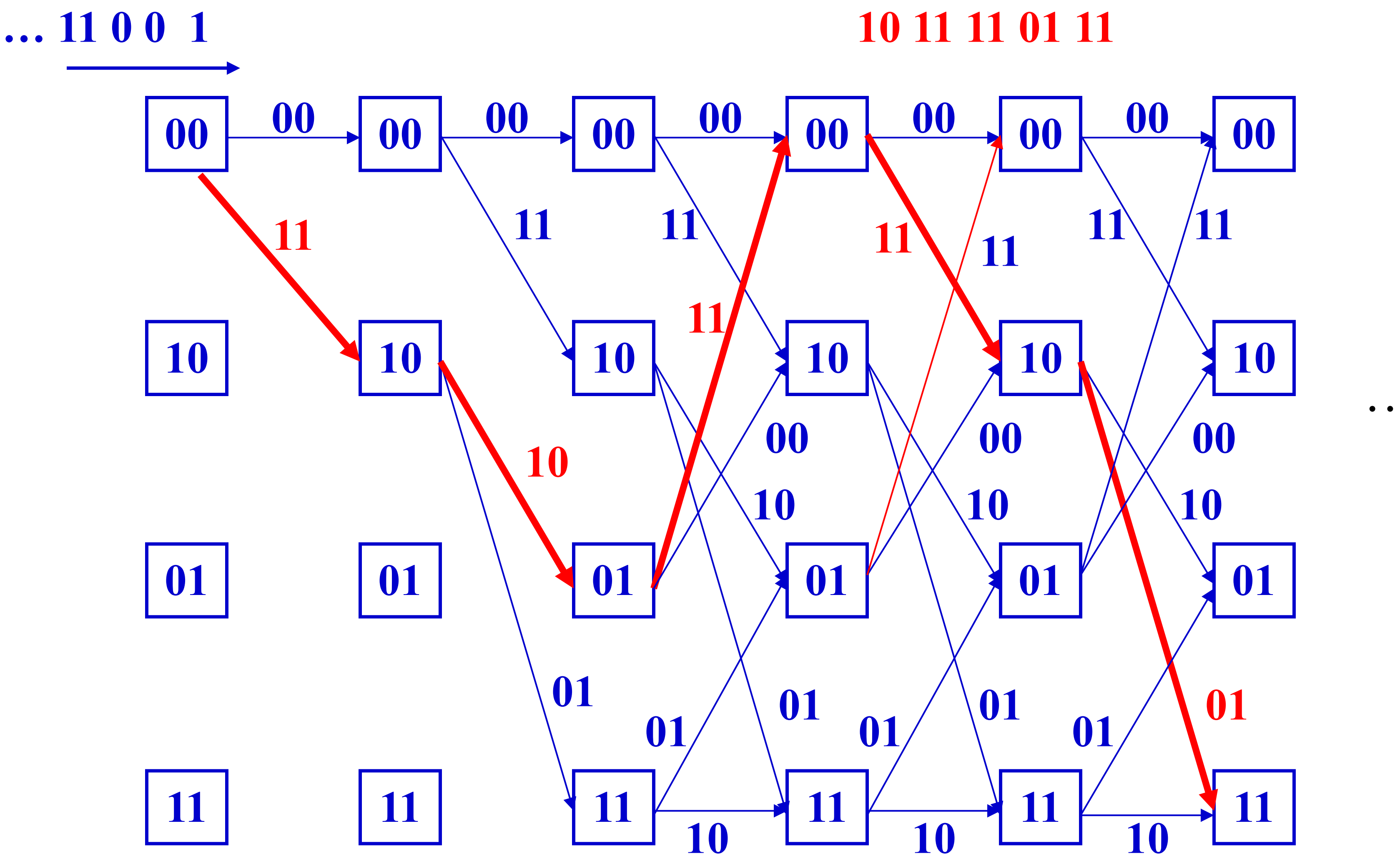
State Diagram



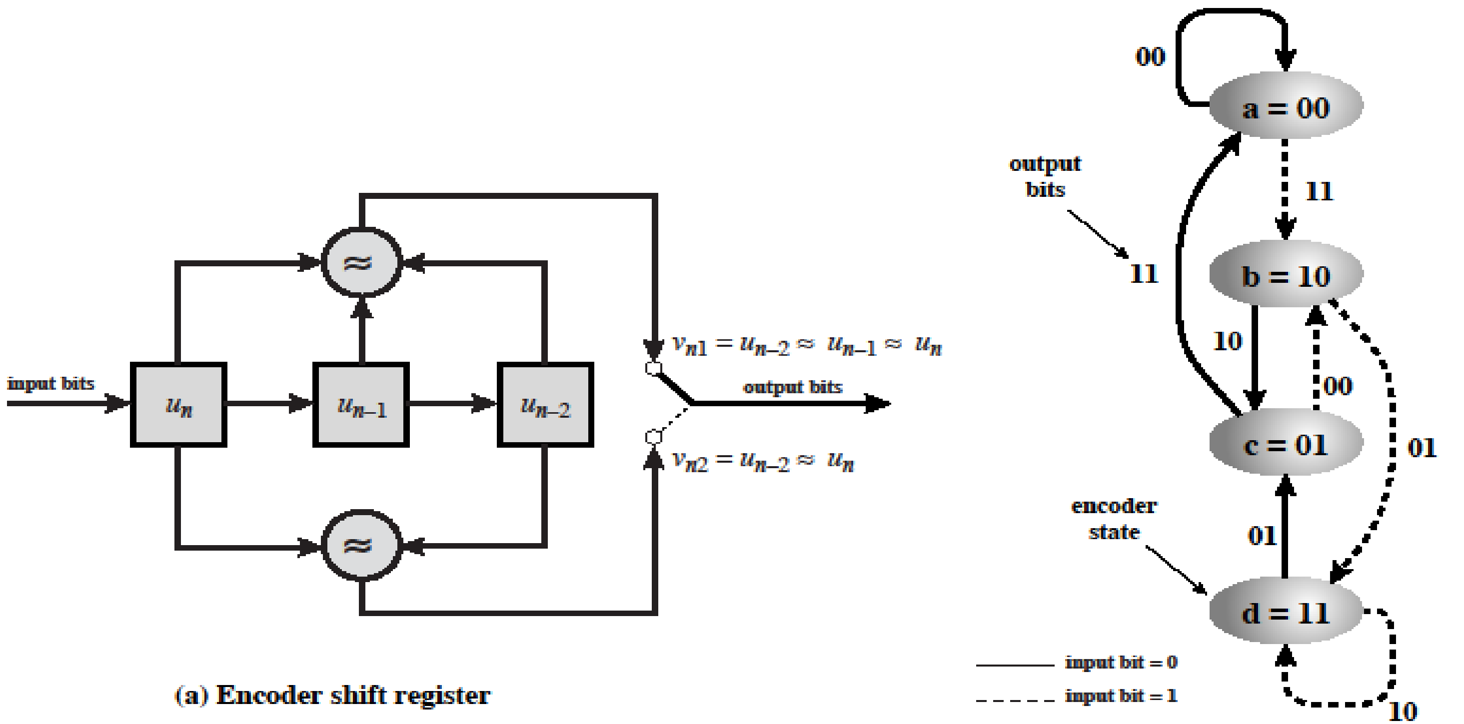
Tree Diagram



Trellis



Convolutional Code (n = 2, k = 1, M= 3)



(a) Encoder shift register

(b) Encoder state diagram

Figure 8.9 Convolutional Encoder with $(n, k, K) = (2, 1, 3)$

Viterbi Algorithm

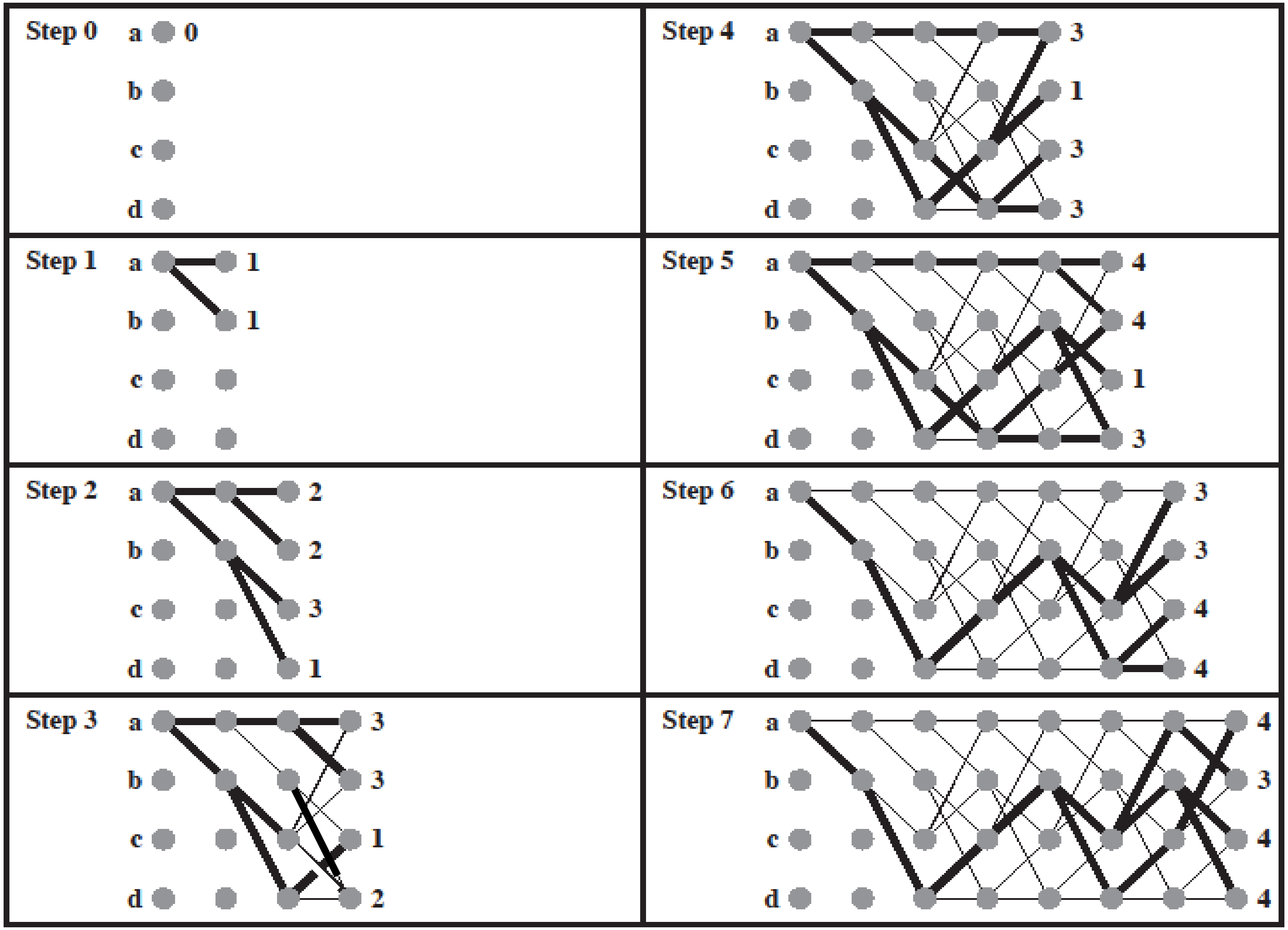


Figure 8.12 Viterbi Algorithm for w = 10010100101100… with decoding window $b = 7$

Interleaving

Input Data a1, a2, a3, a4, a5, a6, a7, a8, a9, …

Write

Interleaving

Read

a1,	a2,	a3,	a4
a5,	a6,	a7,	a8
a9,	a10,	a11,	a12
a13,	a14,	a15,	a16

Transmitting Data a1, a5, a9, a13, a2, a6, a10, a14, a3, …

Through Air

Received Data a1, a5, a9, a13, a2, a6, a10, a14, a3, …

Read

De-Interleaving

Write

a1,	a2,	a3,	a4
a5,	a6,	a7,	a8
a9,	a10,	a11,	a12
a13,	a14,	a15,	a16

Output Data a1, a2, a3, a4, a5, a6, a7, a8, a9, …

Interleaving (Example)

Burst error

Transmitting Data

0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,…

Read

De-Interleaving

Write

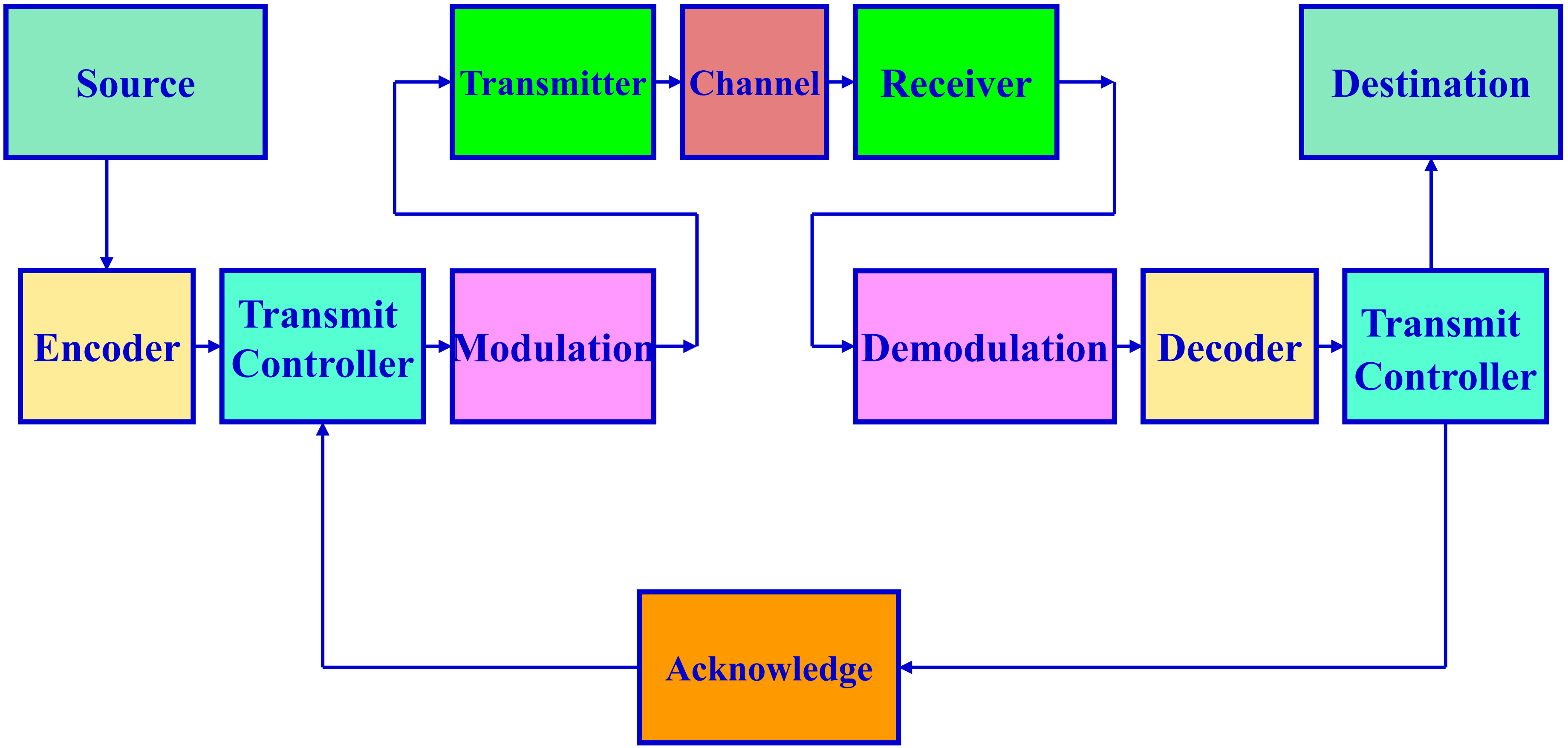
0,	1,	0,	0
0,	1,	0,	0
0,	1,	0,	0
1,	0,	0,	0

Output Data

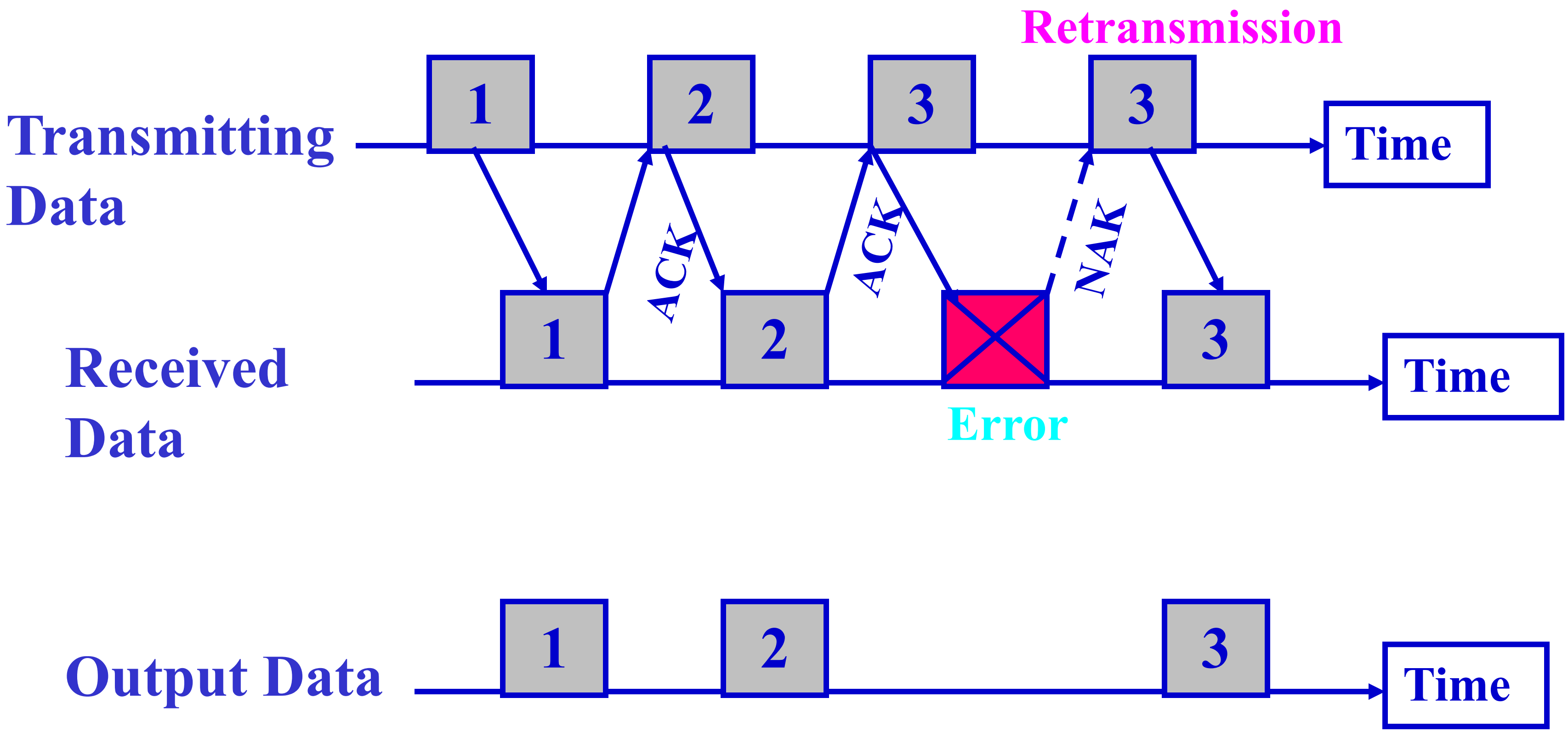
0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, …

Discrete errors

Automatic Repeat Request (ARQ)



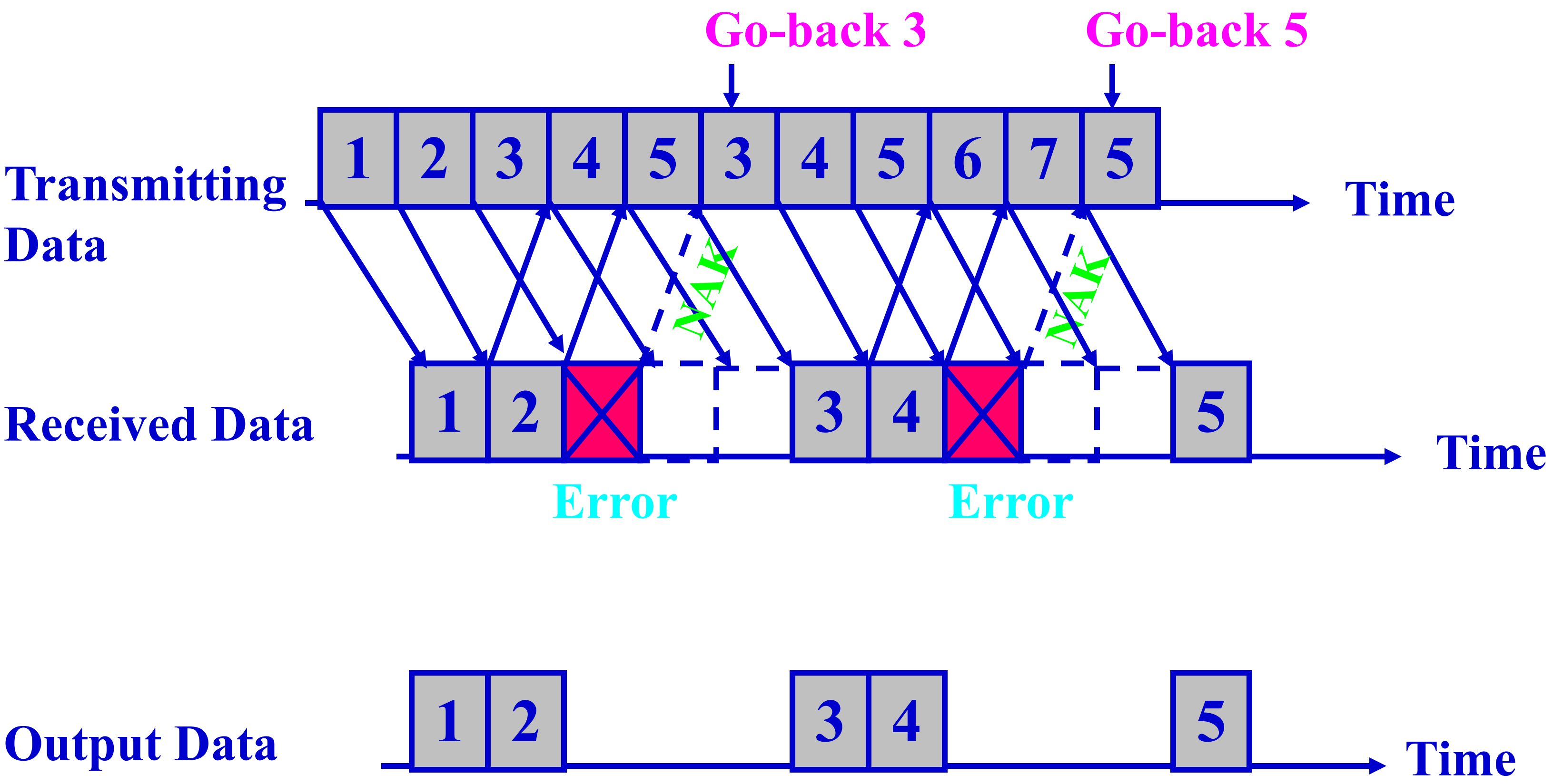
Stop-And-Wait ARQ (SAW ARQ)



ACK: Acknowledge

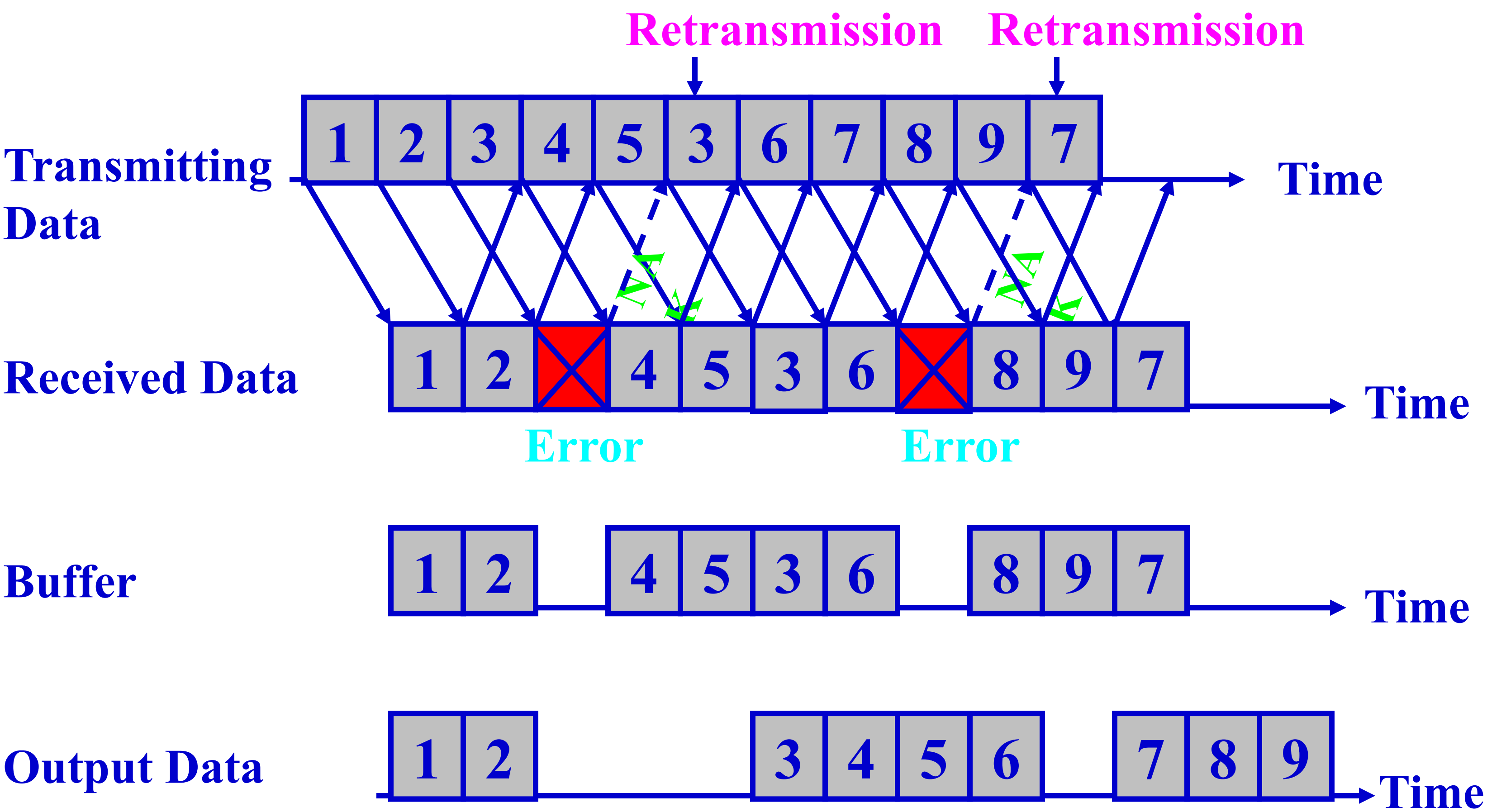
NAK: Negative ACK

Go-Back-N ARQ (GBN ARQ)



- ➢ Sender needs to buffer all the packets have not been acknowledged.
- ➢ Only a buffer of one packet size is needed at the receiver.

Selective-Repeat ARQ (SR ARQ)



- ➢ Receiver needs a large memory to buffer and reorder packets before passing to the upper layer.

Homework

- Problems: 4.2, 4.5, 4.8, 4.13, 4.23