Lecture 2
Error Detection & Correction

- **Types of Errors**
- **Detection**
- **Correction**

Basic concepts

★ Networks must be able to transfer data from one device to another with complete accuracy.

★ Data can be corrupted during transmission.

★ For reliable communication, errors must be detected and corrected.

★ **Error detection and correction** are implemented either at the **data link layer** or the **transport layer** of the OSI model.

Sources of errors

- – a)Electromagnetic distortion of signal: "noise" on the line.
- – b)Error in sampling pulse relative to neighbor pulse: "inter-symbol interference"
- – c)Energy coupling due to nearby links: "cross-talk".
- – d)The storage of information on the magnetic devices is prone to errors due to uneven magnetic surface, dust etc.

# Types of Errors
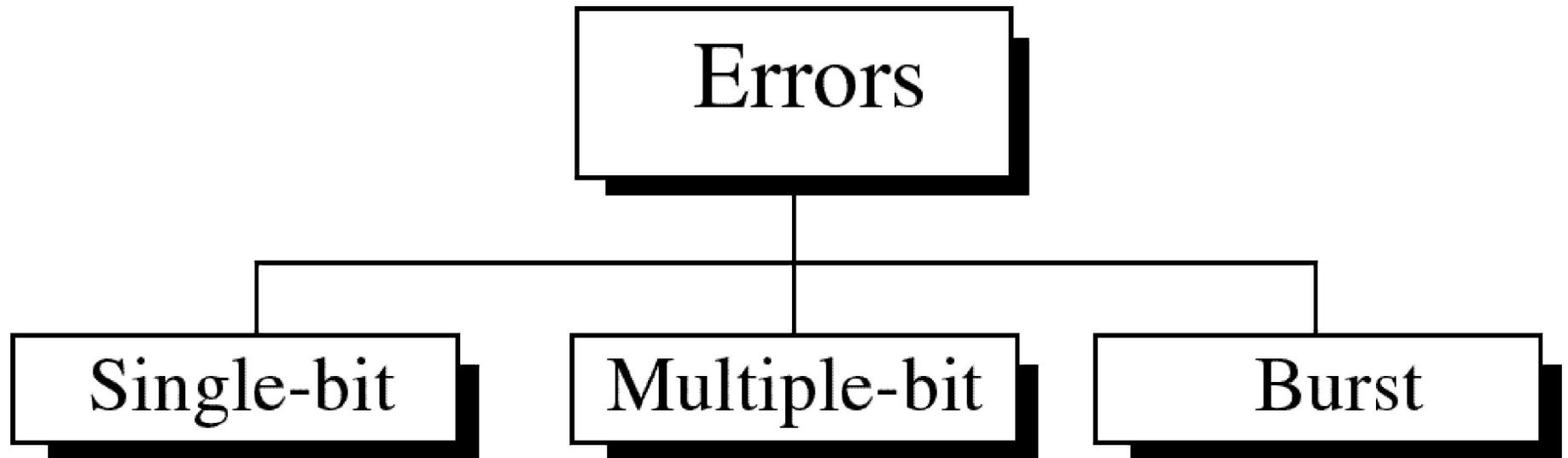
Single-bit error
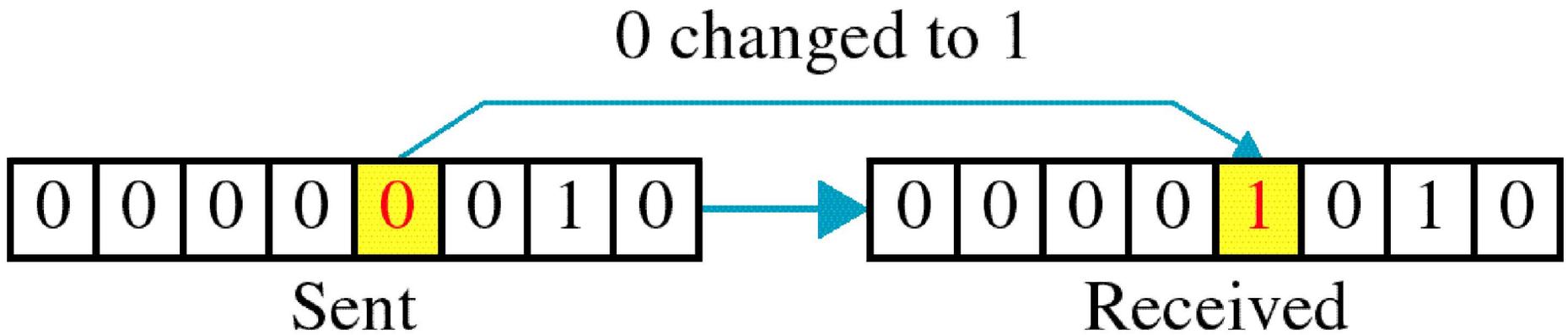
Single bit errors are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in parallel transmission.

Example:

★If data is sent at 1Mbps then each bit lasts only 1/1,000,000 sec. or 1 μs.

★For a single-bit error to occur, the noise must have a duration of only 1 μs, which is very rare.

Burst error

Sent

| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Burst error

| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

Received
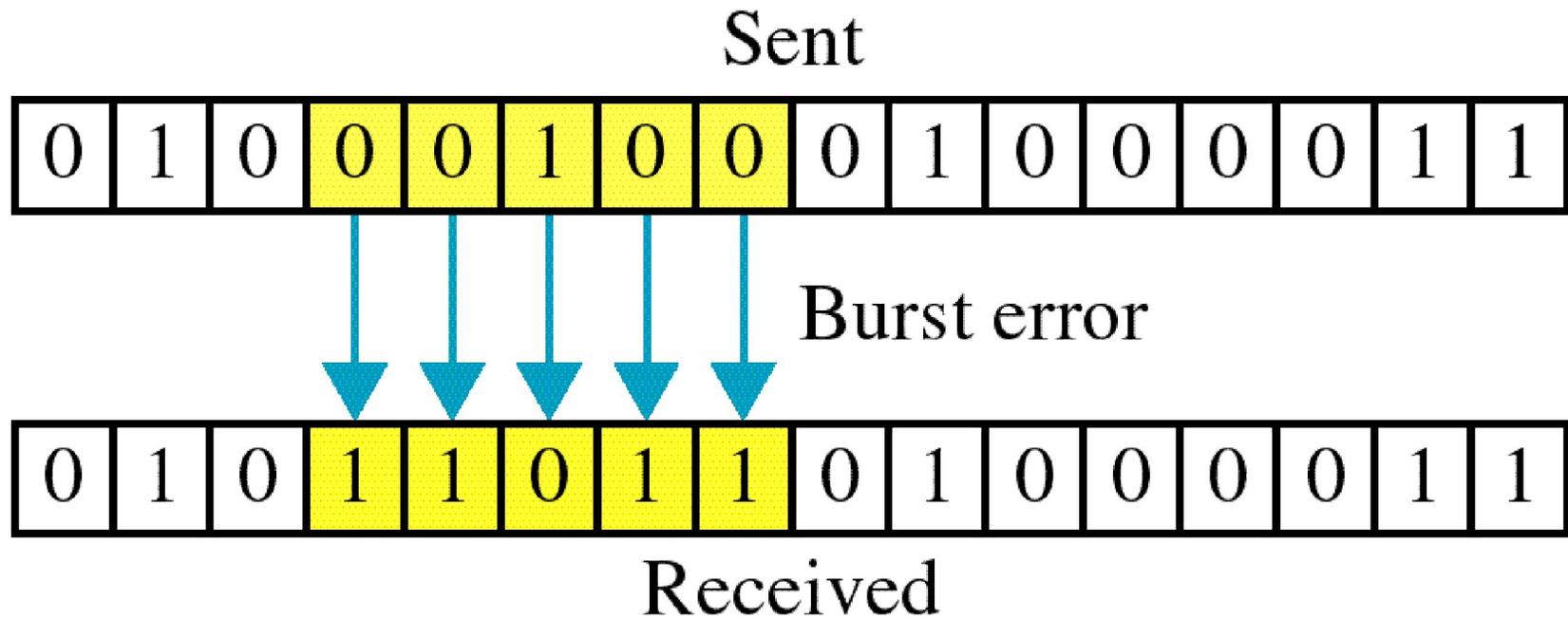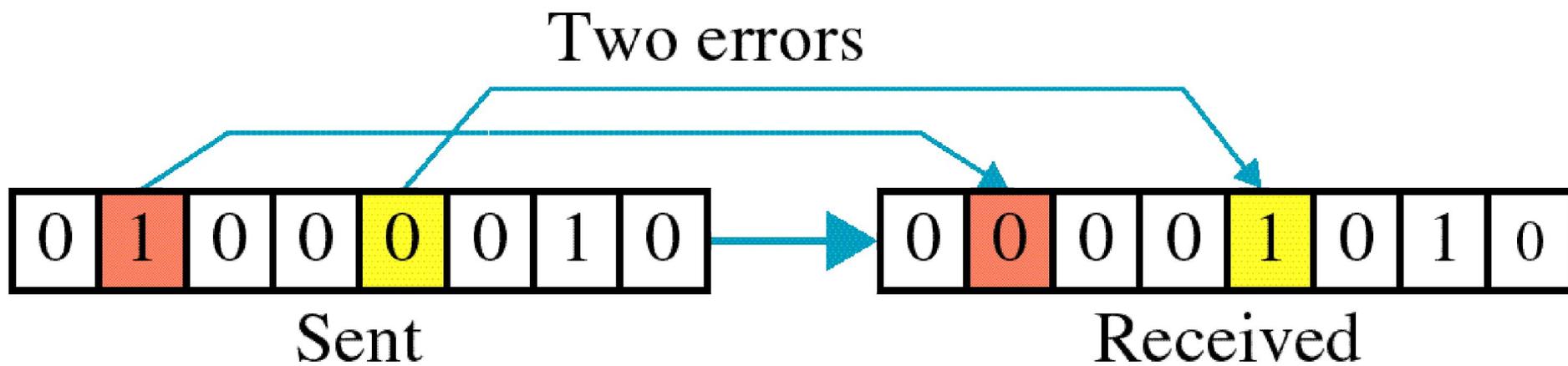
The term **burst error** means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Burst errors does not necessarily **mean that the errors occur in consecutive bits**, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

★**Burst error is most likely to happen in serial transmission** since the duration of noise is normally longer than the duration of a bit.

★The number of bits affected depends on the data rate and duration of noise.

Example:

➡If data is sent at rate = 1Kbps then a noise of 1/100 sec can affect 10 bits.(1/100*1000)

➡If same data is sent at rate = 1Mbps then a noise of 1/100 sec can affect 10,000 bits.(1/100*10^6)

ERROR DETECTING & ERROR CORRECTING CODES

To detect as well as correct these errors, various codes are present and these codes are:

(1) ERROR DETECTING CODES.

(2) ERROR CORRECTING CODES

The first type of code enables the device to detect errors which occur in the code of the bits and second type of code correct the errors automatically.

The various types of error detection and error correction techniques are closely related with the **HAMMING DISTANCE** between any two codes in the set of codes.

Error detection

Error detection means to decide whether the received data is correct or not without having a copy of the original message. In error detection technique, it can only determine whether error has occurred or not but it cannot correct it.

Error detection **uses the concept of redundancy**, **which means** adding extra bits for detecting errors at the destination.

Redundancy
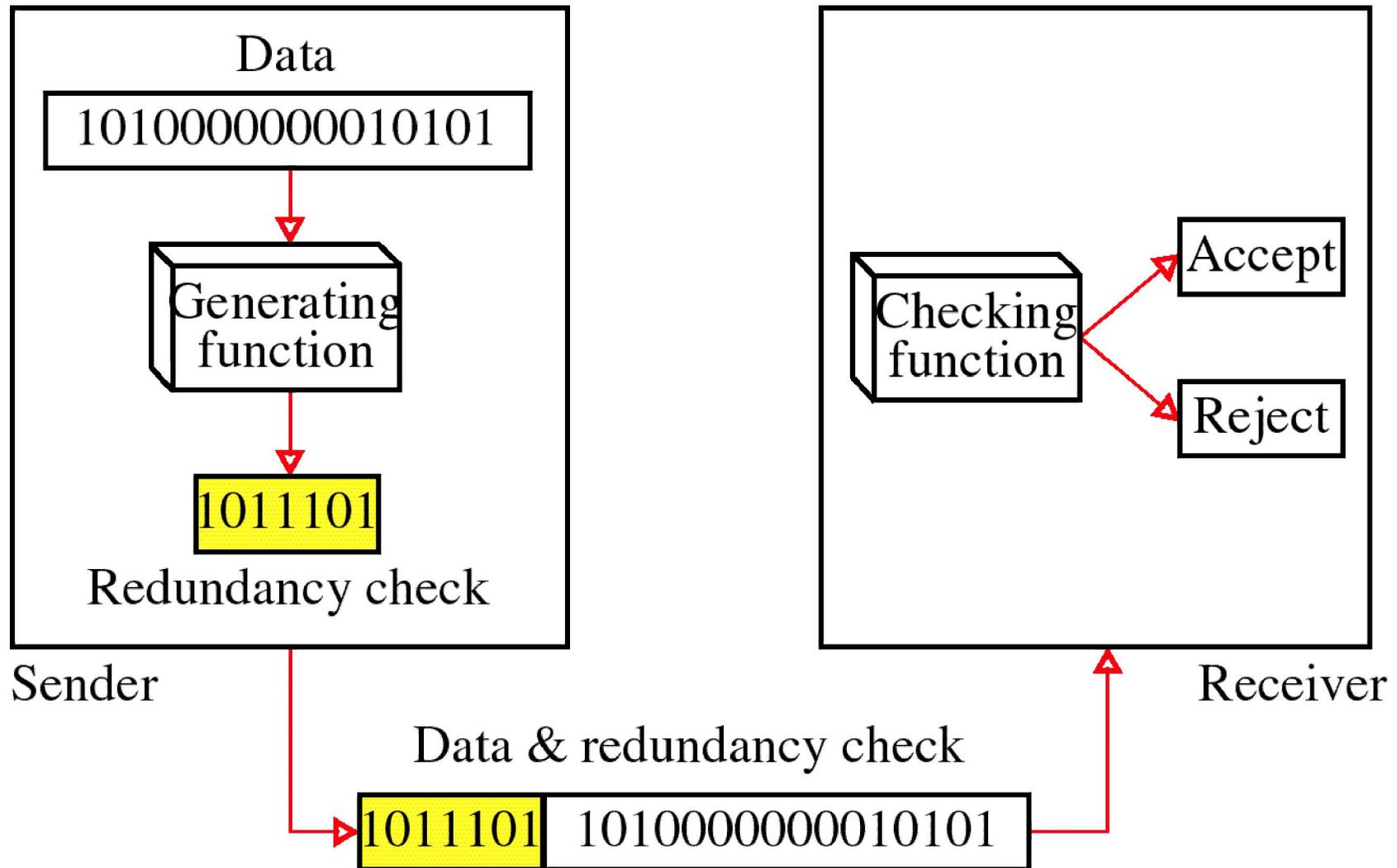
Four types of redundancy checks are used in data communications
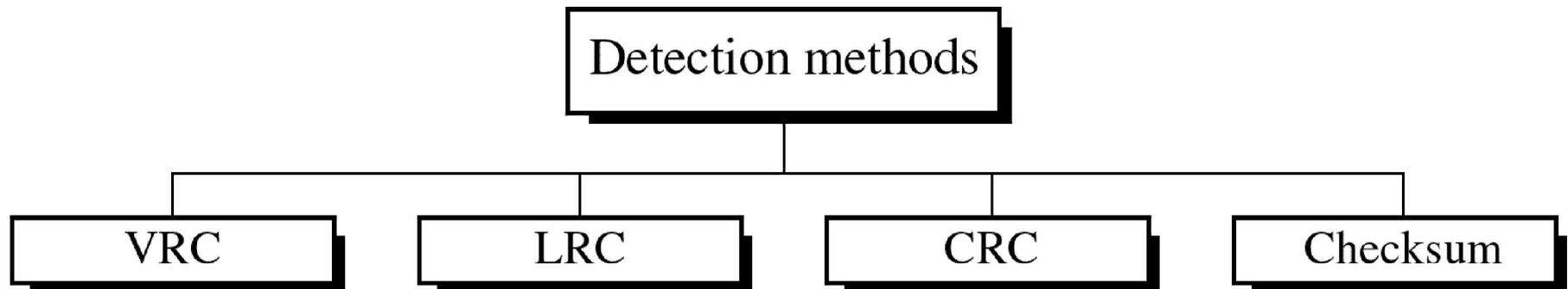
ERROR-DETECTING CODES

PARITY

PARITY is the no. of ones present in the data whereas PARITY BIT is an extra bit (that may be 0 or 1) which may be added in the data for making the parity to be even or odd. The two different methods are used & they are:

- **EVEN PARITY METHOD.**

In this method, the value of parity bit is taken as 0 or 1 depending upon the total no. of 1's INCLUDING THE PARITY BIT to be even.

- **ODD PARITY METHOD.**

This method is similar to the even parity technique such that total no. of 1's INCLUDING THE PARITY BIT would be odd.

For detection of error an extra bit known as *parity bit* is attached to each code word to make the no. of ones in the code even(even parity) or odd(odd parity).

ERROR-DETECTING CODES

BCD code with the parity bit

BCD code with even & odd parities are shown in the table below

BCD code				BCD code with even parity					BCD code with odd parity				
D	C	B	A	P	D	C	B	A	P	D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0	0	1
0	0	1	0	1	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1	1	0	0	1	1
0	1	0	0	1	0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1	1	0	1	0	1
0	1	1	0	0	0	1	1	0	1	0	1	1	0
0	1	1	1	1	0	1	1	1	0	0	1	1	1
0	0	0	0	1	1	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	1	1	1	0	0	1

Error detection: Parity

- Add an extra bit to a code to ensure an even (odd) number of 1s
  - Every code word has an even (odd) number of 1s

- Detects all 1-bit errors

data strings

01 — 11
00 — 10

Parity Encoding:

White – Invalid (error)

011 — 111
010 — 110
000 — 100
001 — 101

Vertical Redundancy Check
VRC
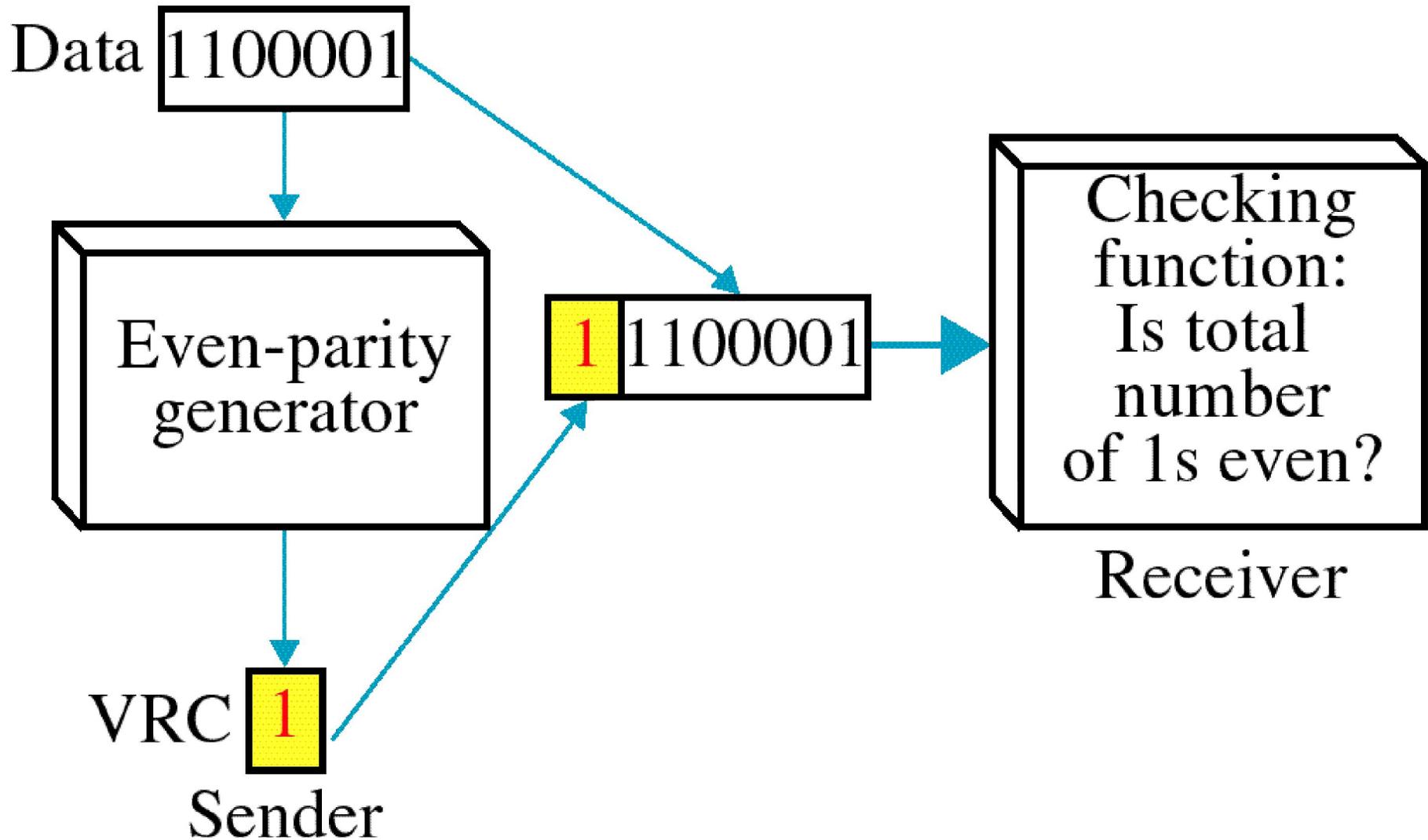
Performance

➔It can detect single bit error

➔It can detect burst errors only if the total number of errors is odd.

Longitudinal Redundancy Check
LRC

Direction of movement →

| 10101010	10101001	00111001	11011101	11100111
LRC		Data		
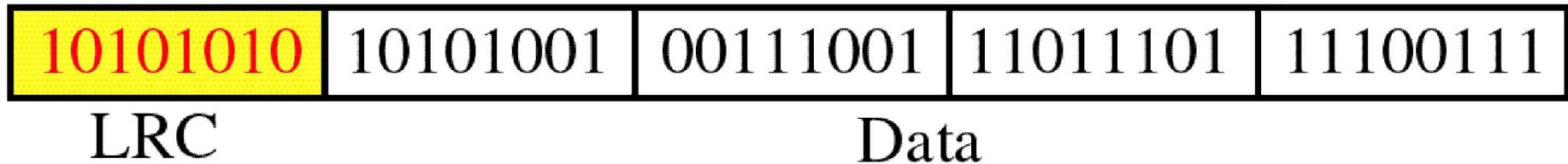
Performance

➔ LRC increases the likelihood of detecting burst errors.

➔ If two bits in one data units are damaged and two bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

VRC and LRC
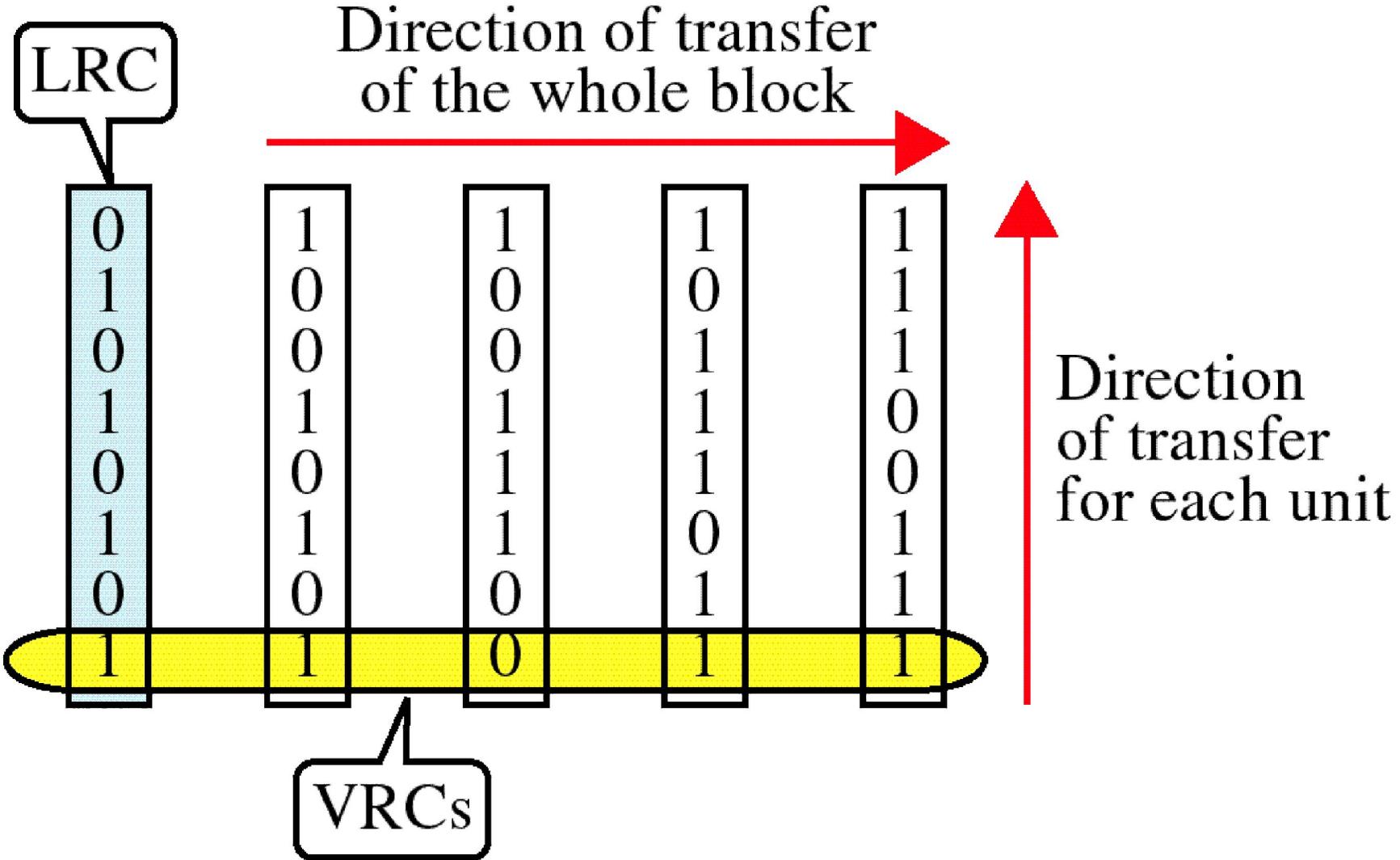
Cyclic Redundancy Check
CRC

Sender

Receiver
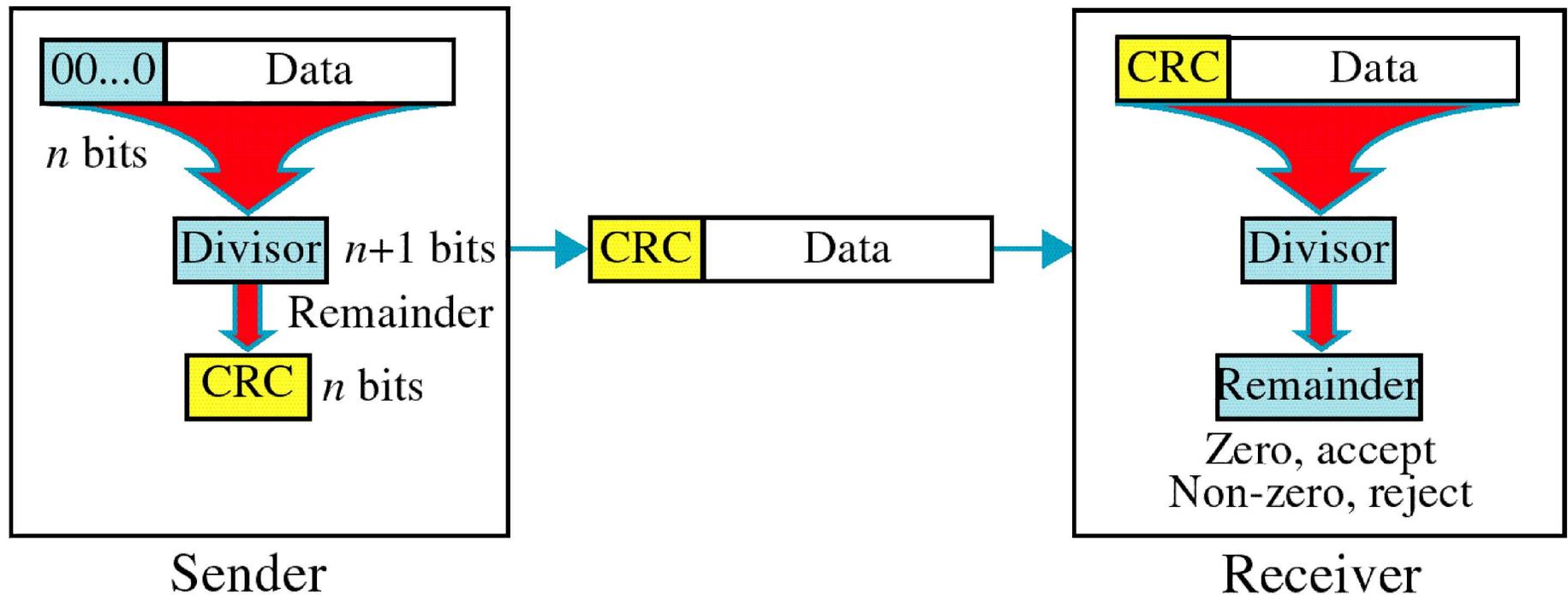
Cyclic Redundancy Check

- Given a k-bit frame or message, the transmitter generates an n-bit sequence, known as a *frame check sequence (FCS)*, so that the resulting frame, consisting of (k+n) bits, is exactly divisible by some predetermined number.

- The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error.

Binary Division
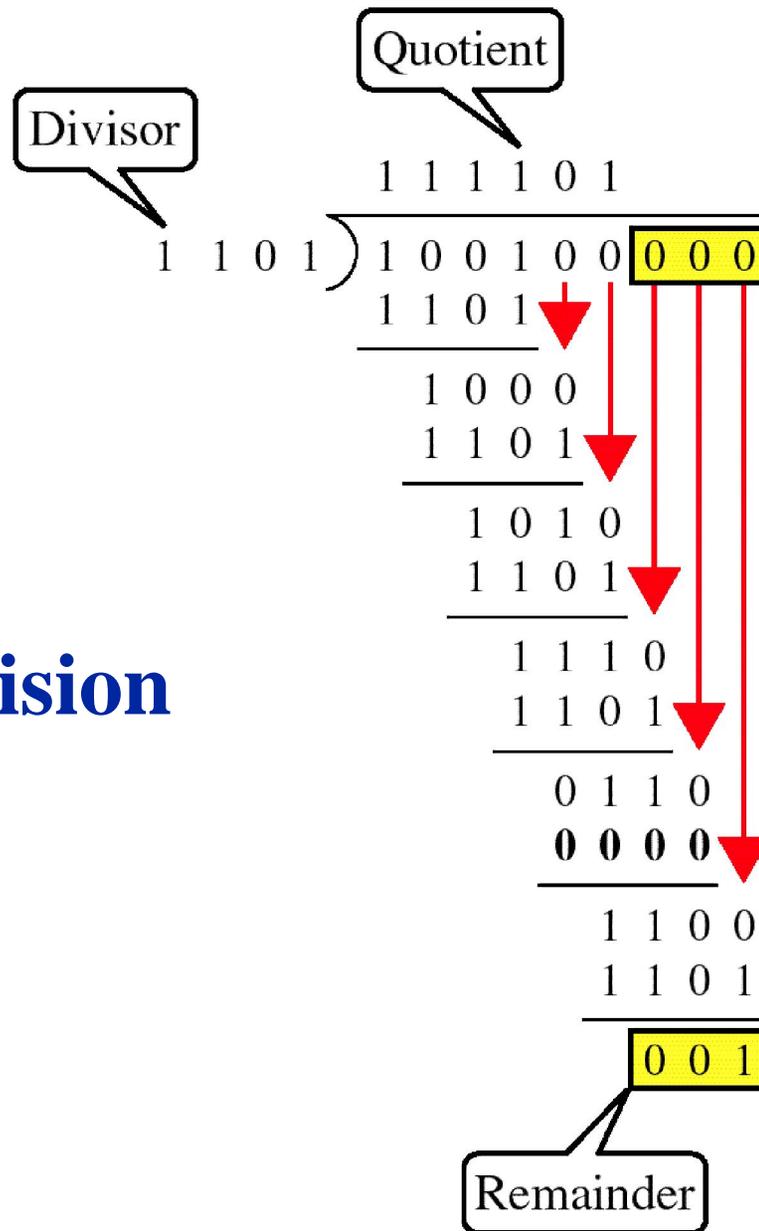
Polynomial

$$x^7 + x^5 + x^2 + x + 1$$

Polynomial and Divisor

Polynomial

$$x^7 + x^5 + x^2 + x + 1$$

x^6 x^4 x^3

1 0 1 0 0 1 1 1
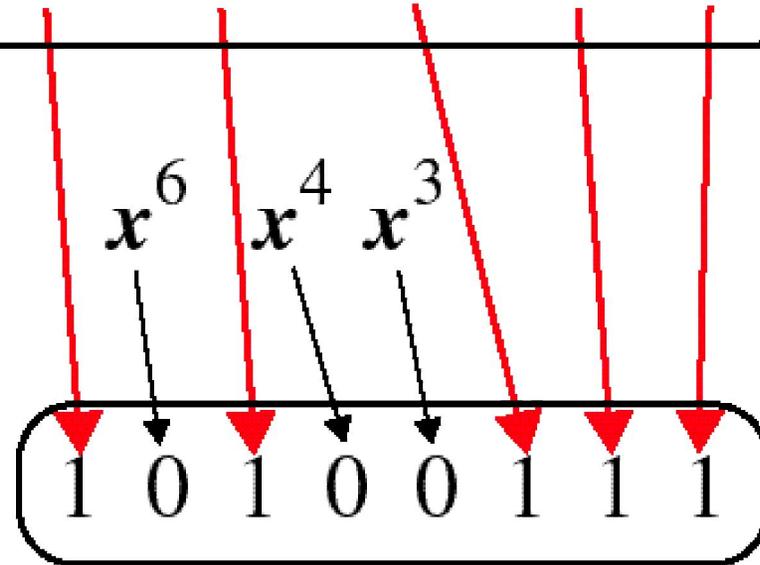
Divisor

Standard Polynomials

CRC-12

$$x^{12} + x^{11} + x^3 + x + 1$$

CRC-16

$$x^{16} + x^{15} + x^2 + 1$$

CRC-ITU

$$x^{16} + x^{12} + x^5 + 1$$

CRC-32

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

CHECKSUM METHOD

The PARITY METHOD will not work when even no. of changes occur because the even error will not change the oddness & evenness of the total no. of 1's in the code.

Thus, to avoid this **CHECKSUM METHOD** is used as each data to be transmitted is added to the previous one & sum is retained at the sending end. At the end of transmission, the total sum known as checksum is sent and is checked at the receiving point.

Checksum
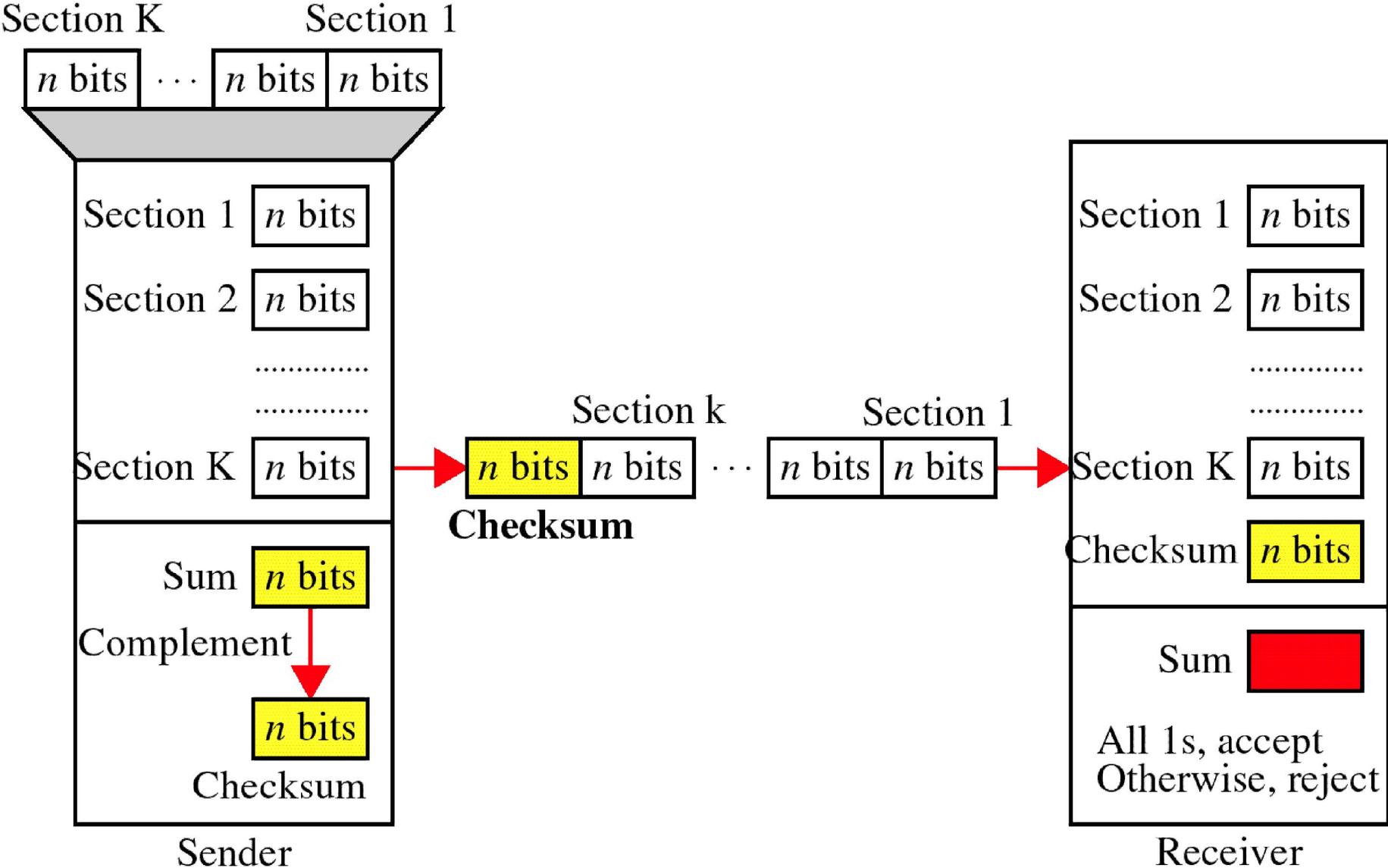
At the sender

➲ The unit is divided into k sections, each of n bits.

➲ All sections are added together using one's complement to get the sum.

➲ The sum is complemented and becomes the checksum.

➲ The checksum is sent with the data

At the receiver

- The unit is divided into k sections, each of n bits.

- All sections are added together using one's complement to get the sum.

- The sum is complemented.

- If the result is zero, the data are accepted: otherwise, they are rejected.

Performance

➡ The checksum detects all errors involving an odd number of bits.

➡ It detects most errors involving an even number of bits.

➡ If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged, the sums of those columns will not change and the receiver will not detect a problem.

Error Correction

It can be handled in two ways:

1) receiver can have the sender retransmit the entire data unit.

2) The receiver can use an error-correcting code, which automatically corrects certain errors.

Single-bit error correction

To correct an error, the receiver reverses the value of the altered bit. To do so, it must know which bit is in error.

Number of redundancy bits needed

- Let data bits $= m$

- Redundancy bits $= r$

$\therefore$ Total message sent $= m + r$

The value of r must satisfy the following relation:

$$2^r \geq m + r + 1$$

Error Correction

Data (m) bits

Redundancy (r) bits

Total $m + r$ bits
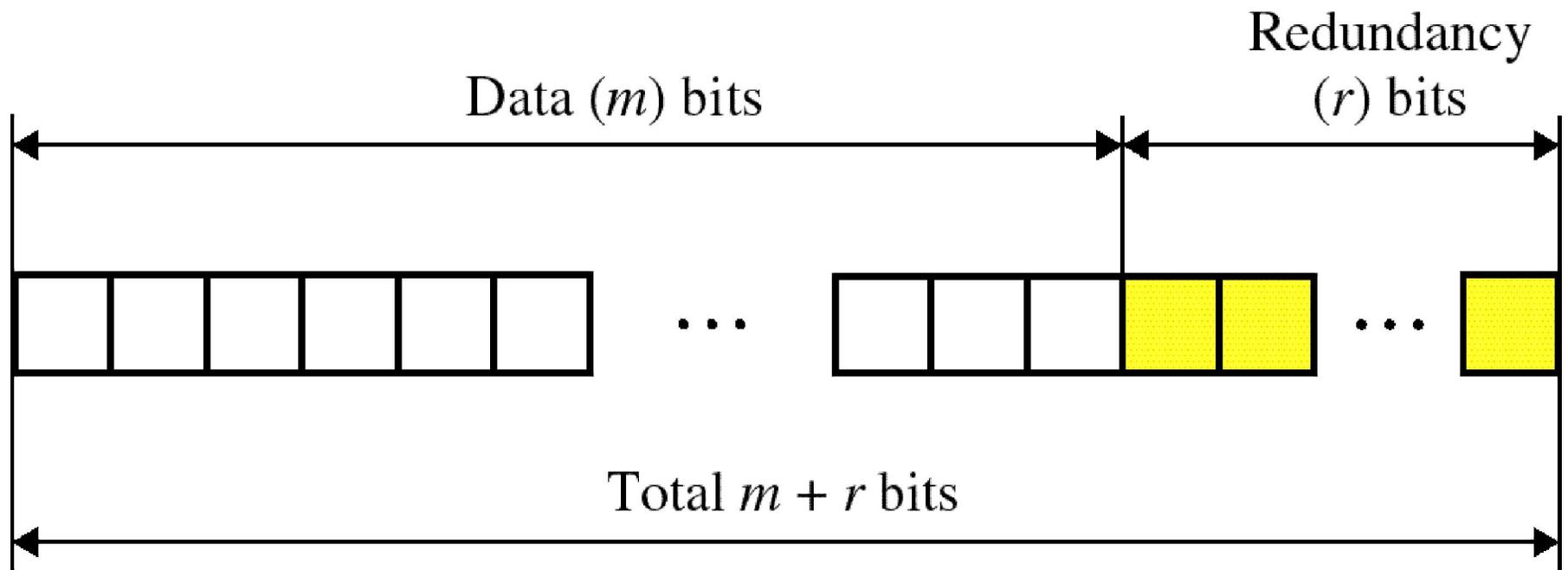
Hamming Codes

HAMMING CODE :

is constructed by adding a no. of parity bits to each group of n-bit information, or message in such a way so as to be able to locate the bit position in which error occurs.

Values (0 or 1) are assigned to the parity bits so as to make the hamming code have either even parity or odd parity & when an error occurs, the position no. will take on the value assigned to the location of the erroneous bit.

Hamming Code
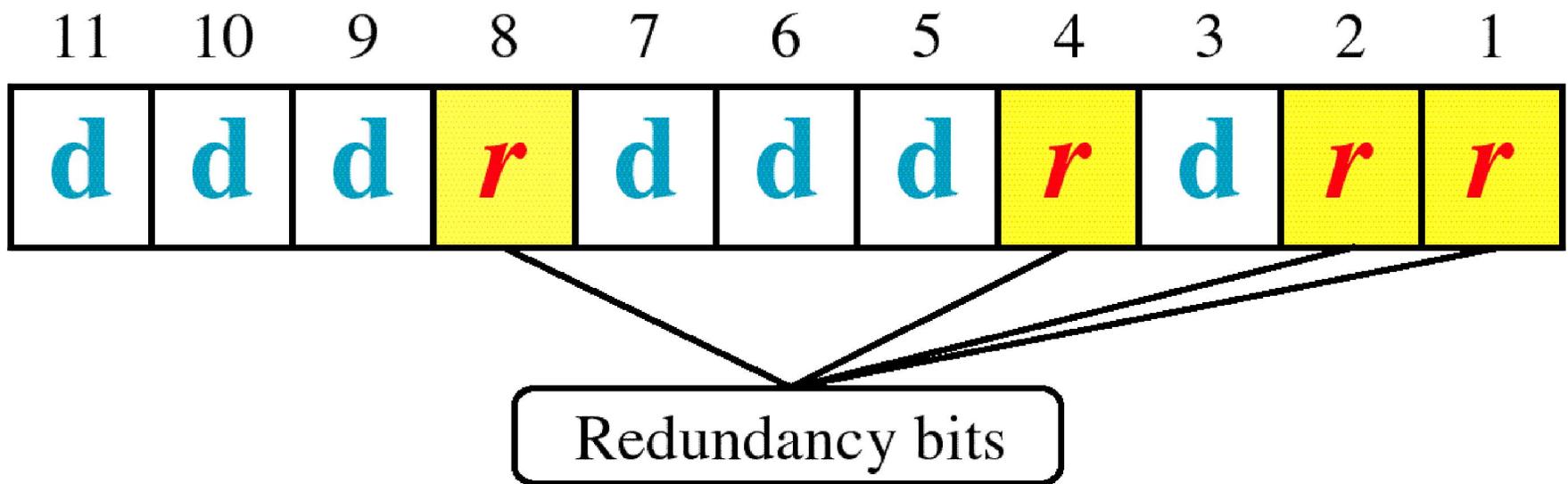
Hamming Codes

Hamming Codes

    Construction for 1-bit error-correcting codes

    Minimal number of check bits required

Construction

    number bits from 1 upward

    powers of 2 are check bits

   all others are data bits

   Check bit j is XOR of all bits k such that (j AND k) = j

Example:

   4 bits of data, 3 check bits

Hamming Codes: Example

1	2	3	4	5	6	7
C_1	C_2	D_3	C_4	D_5	D_6	D_7

$$C1 = D3 \oplus D5 \oplus D7$$

$$C2 = D3 \oplus D6 \oplus D7$$

$$C4 = D5 \oplus D6 \oplus D7$$

Hamming Codes: Example

- Say the data is 0100.
- Then D3=0, D5=1, D6=0, D7=0.
- Therefore C1=1, C2=1, C4=0
  ➔ transmit 1001100
- Suppose we receive 1011100.
- Verify checkbits:
  - C1 ⊕ D3 ⊕ D5 ⊕ D7 = 1
    - Position 1
  - C2 ⊕ D3 ⊕ D6 ⊕ D7 = 1
    - Position 2
  - C4 ⊕ D5 ⊕ D6 ⊕ D7 = 0
    - Position 4

C1	C2	D3	C4	D5	D6	D7

C1 = D3 ⊕ D5 ⊕ D7
C2 = D3 ⊕ D6 ⊕ D7
C4 = D5 ⊕ D6 ⊕ D7

Result:
We know that there's an error, in position 1+2, so bit #3 should be flipped

Hamming Codes: Example

For example

A hamming code <u>0110001</u> is being received. Find the correct code which is being transmitted.

SOLUTION:	001	010	011	100	101	110	111
	1	2	3	4	5	6	7
	P1	P2	D1	P3	D2	D3	D4
	0	1	1	0	0	0	1

$$P1 = 1, 3, 5, 7$$
$$P2 = 2, 3, 6, 7$$
$$P3 = 4, 5, 6, 7$$

Checksum Bits:

$$C1 = 0$$
$$C2 = 1$$
$$C3 = 1$$

In reverse order, 110.
Therefore, error has occurred at position 6.
Hence, the exact code transmitted is: <u>0110011</u>.

ERROR DETECTING & ERROR CORRECTING CODES
(Conclusion)

1. These codes help us to trace out the errors in the digital data and to correct that error.

2. There are many types of codes for error detection and correction e.g. parity bit is a kind of error detection code that can find only one bit error in the data.

3. Hamming code is an error detection and correction code.

4. These codes are used in detecting and correcting error in the transmitted digital data and it helps in the error free transmission of the data.

Hamming Code

r_1 will take care of these bits

1011   1001   0111   0101   0011   0001
11   **9**   **7**   **5**   **3**   **1**

| d | d | d | r_8 | d | d | d | r_4 | d | r_2 | r_1 |

r_2 will take care of these bits

10111010   01110110   0011 0010
11   **10**   **7**   **6**   **3**   **2**

| d | d | d | r_8 | d | d | d | r_4 | d | r_2 | r_1 |

Hamming Code

r_4 will take care of these bits

$$0111\ 0110\ 0101\ 0100$$
$$7 \qquad 6 \qquad 5 \qquad 4$$

d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_8 will take care of these bits

$$1011\ 1010\ 1001\ 1000$$
$$11 \quad 10 \quad 9 \qquad 8$$

d	d	d	r_8	d	d	d	r_4	d	r_2	r_1
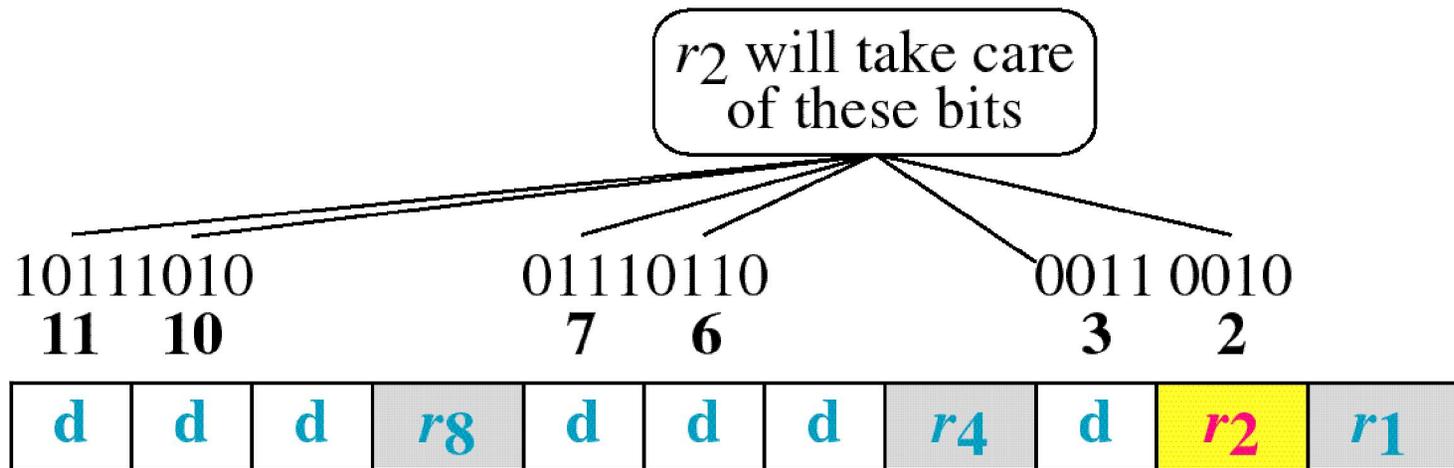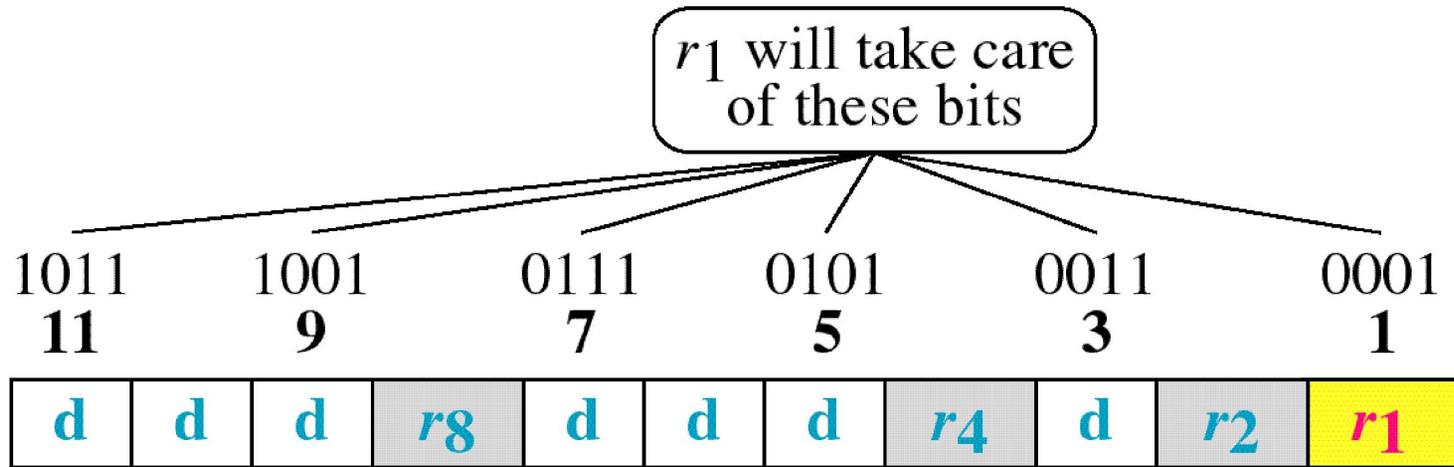
Example of Hamming Code

Data: **1 0 0 1 1 0 1**
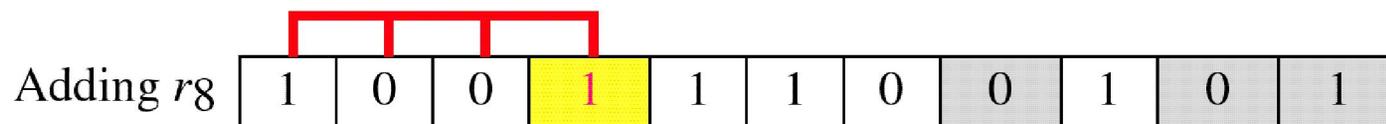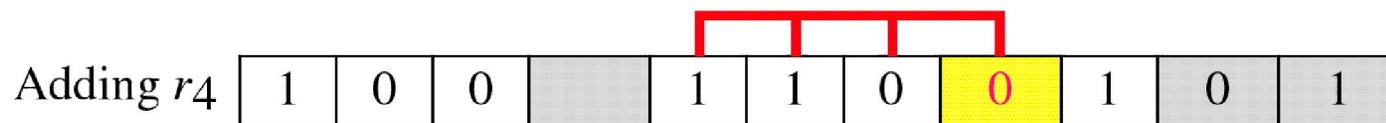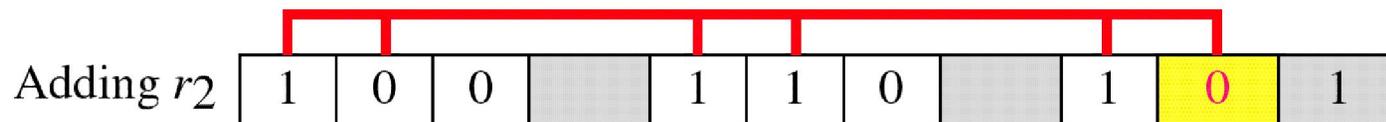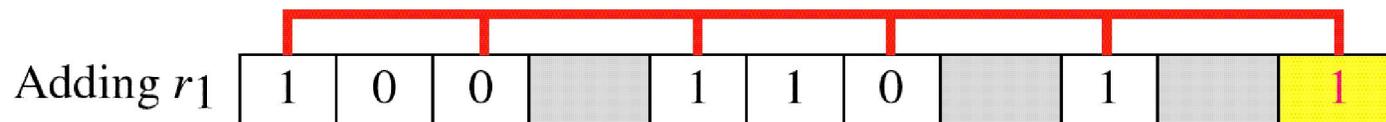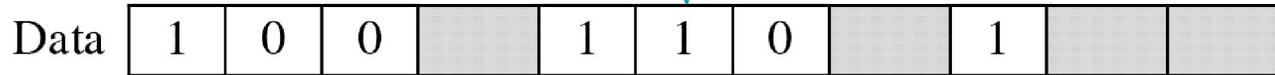
Code: **1 0 0 1 1 1 0 0 1 0 1**

Single-bit error

Sent

| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Received

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

Error

Error Detection

11 10 9 8 7 6 5 4 3 2 1

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

11 10 9 8 7 6 5 4 3 2 1

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

11 10 9 8 7 6 5 4 3 2 1

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

11 10 9 8 7 6 5 4 3 2 1

| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

0 1 1 1

7

The bit in position 7
is in error.
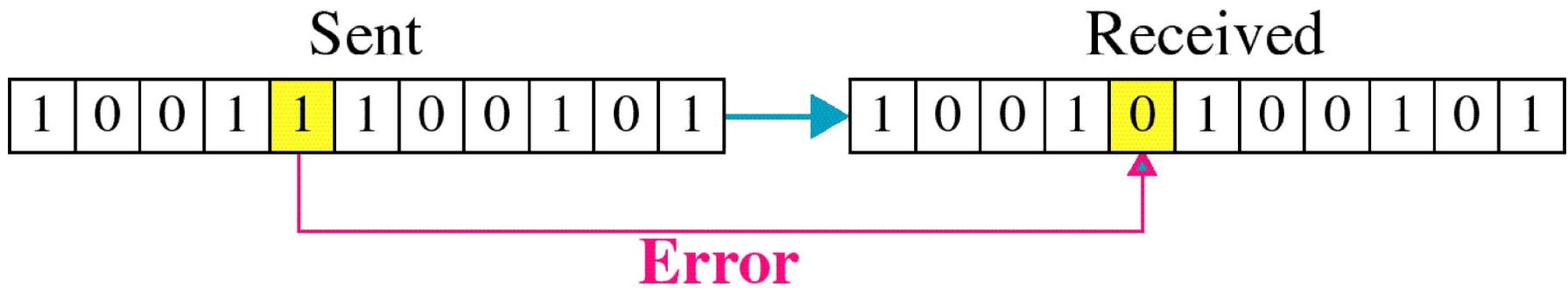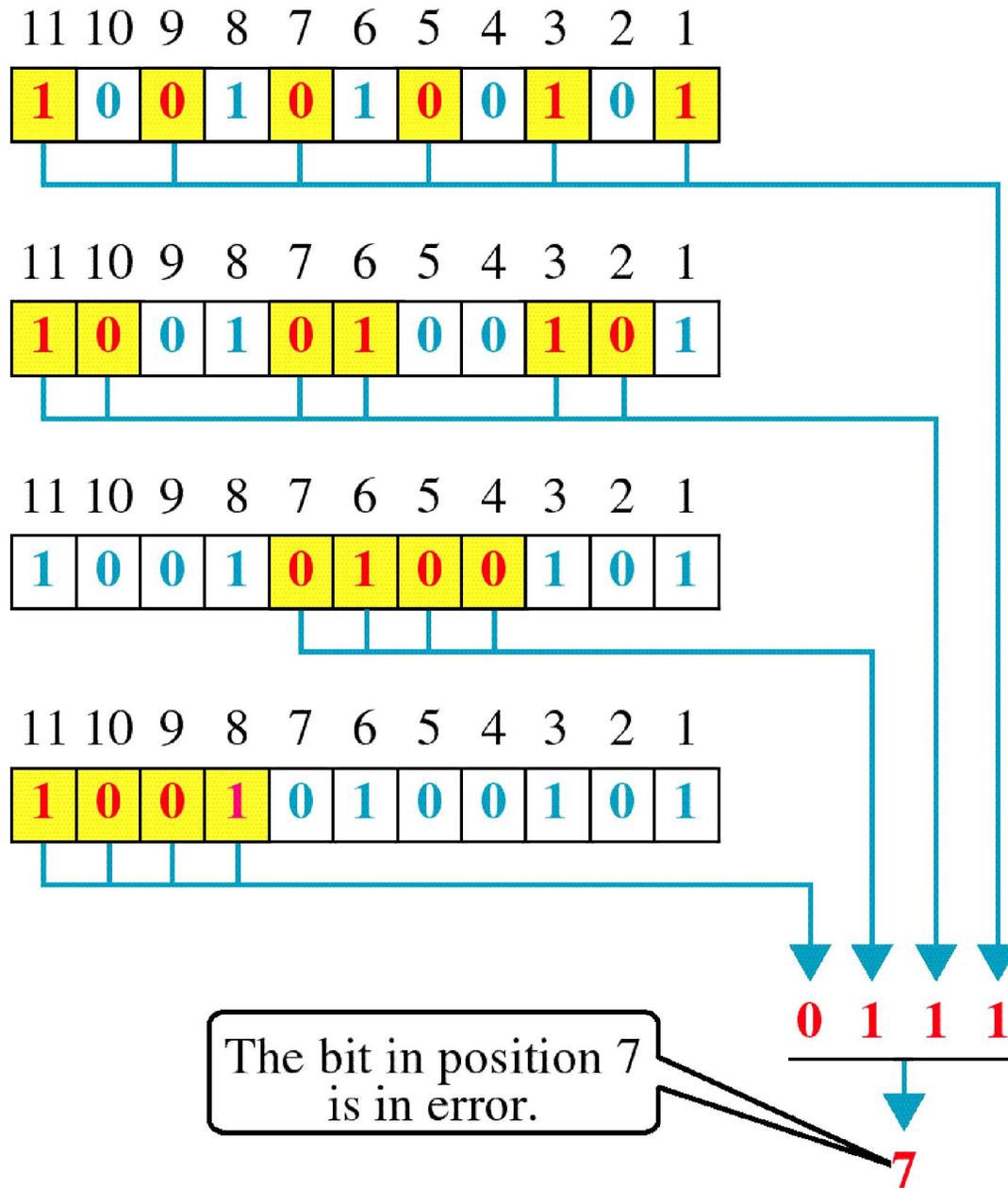
Assignment-2

- Generate the hamming code for data 111011001 with even parity ?