Error Detection and Correction

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

What is data transmission?

- Data transmission refers to the process of transferring data between two or more digital devices.
- Data is transmitted from one device to another in analog or digital format.
- Basically, data transmission enables devices or components within devices to speak to each other.

How does data transmission work between digital devices?

- Data is transferred in the form of bits between two or more digital devices.
- There are two methods used to transmit data between digital devices:
  - serial transmission
  - parallel transmission.

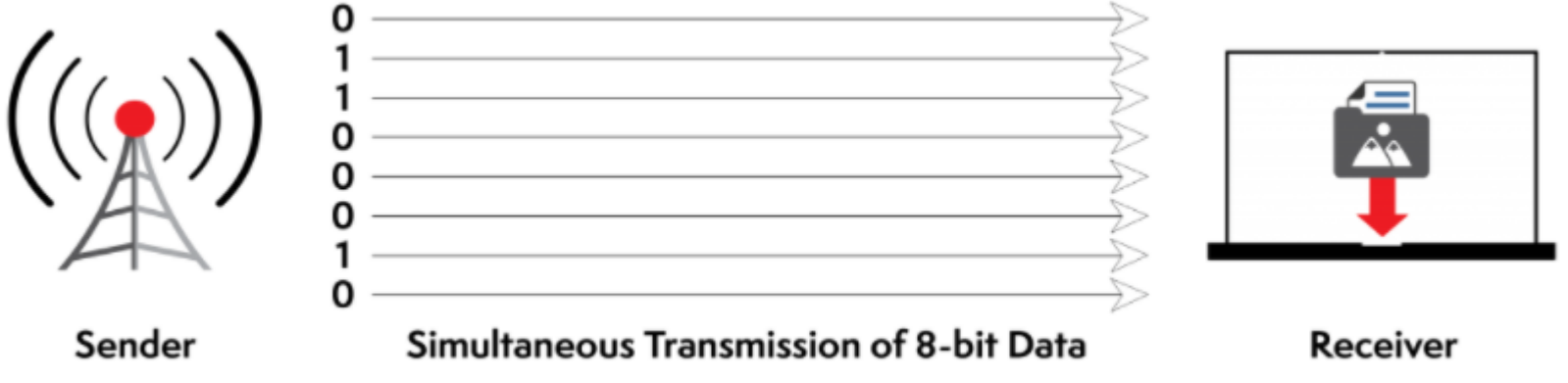
What is serial transmission?

Example of Serial Data Transmission

- When is serial transmission used to send data?
  - Serial transmission is normally used for long-distance data transfer.
  - It is also used in cases where the amount of data being sent is relatively small.
  - It ensures that data integrity is maintained as it transmits the data bits in a specific order, one after another.
- In this way, data bits are received in-sync with one another.

What is parallel transmission?

- When data is sent using parallel data transmission, multiple data bits are transmitted over multiple channels at the same time.
- This means that data can be sent much faster than using serial transmission methods.



Example of Parallel Data Transmission

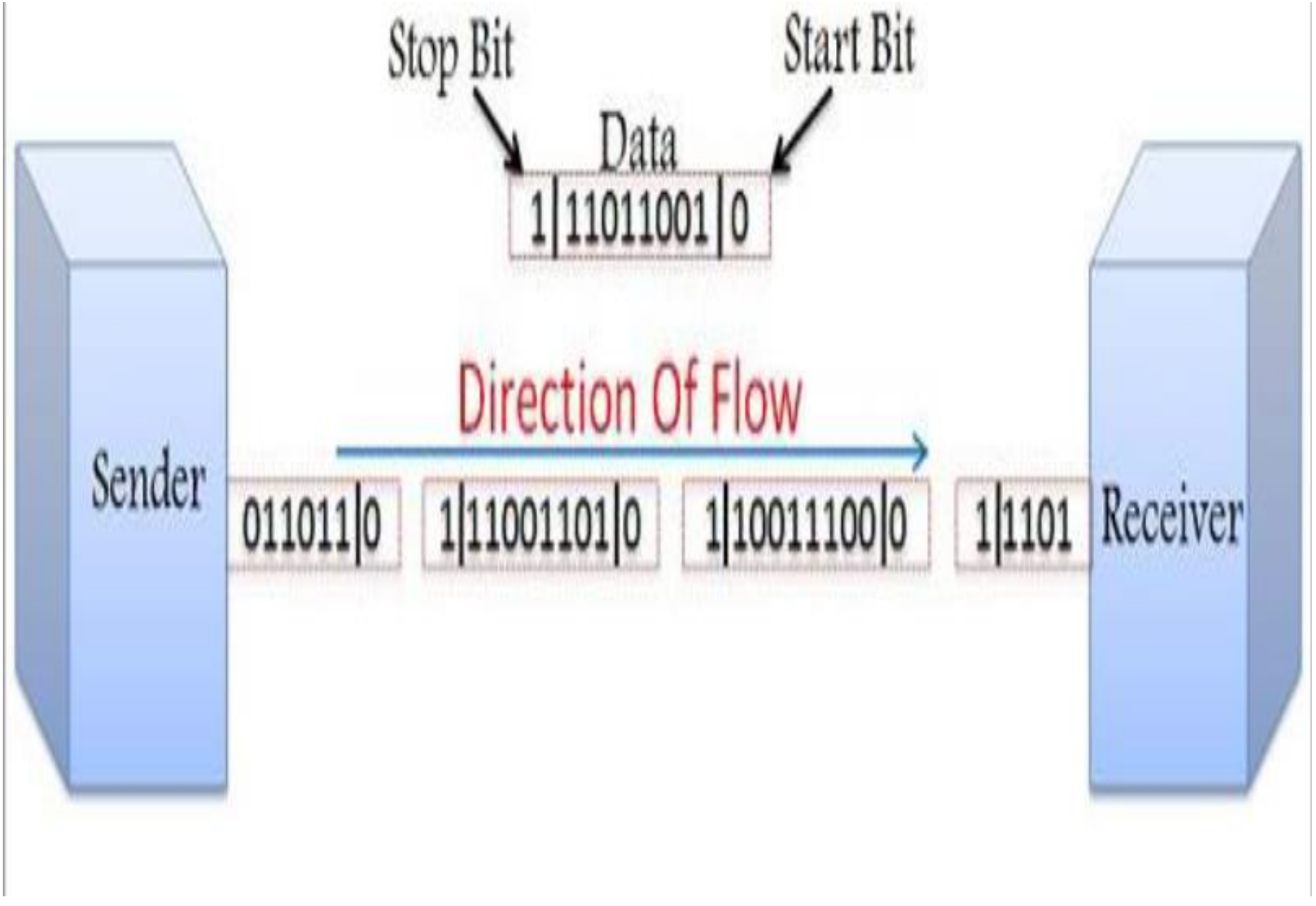
- The main advantages of parallel transmission over serial transmission are:
  - it is easier to program;
  - and data is sent faster.

- Disadvantage:
  - Although parallel transmission can transfer data faster, it requires more transmission channels than serial transmission.
  - This means that data bits can be out of sync, depending on transfer distance and how fast each bit loads.

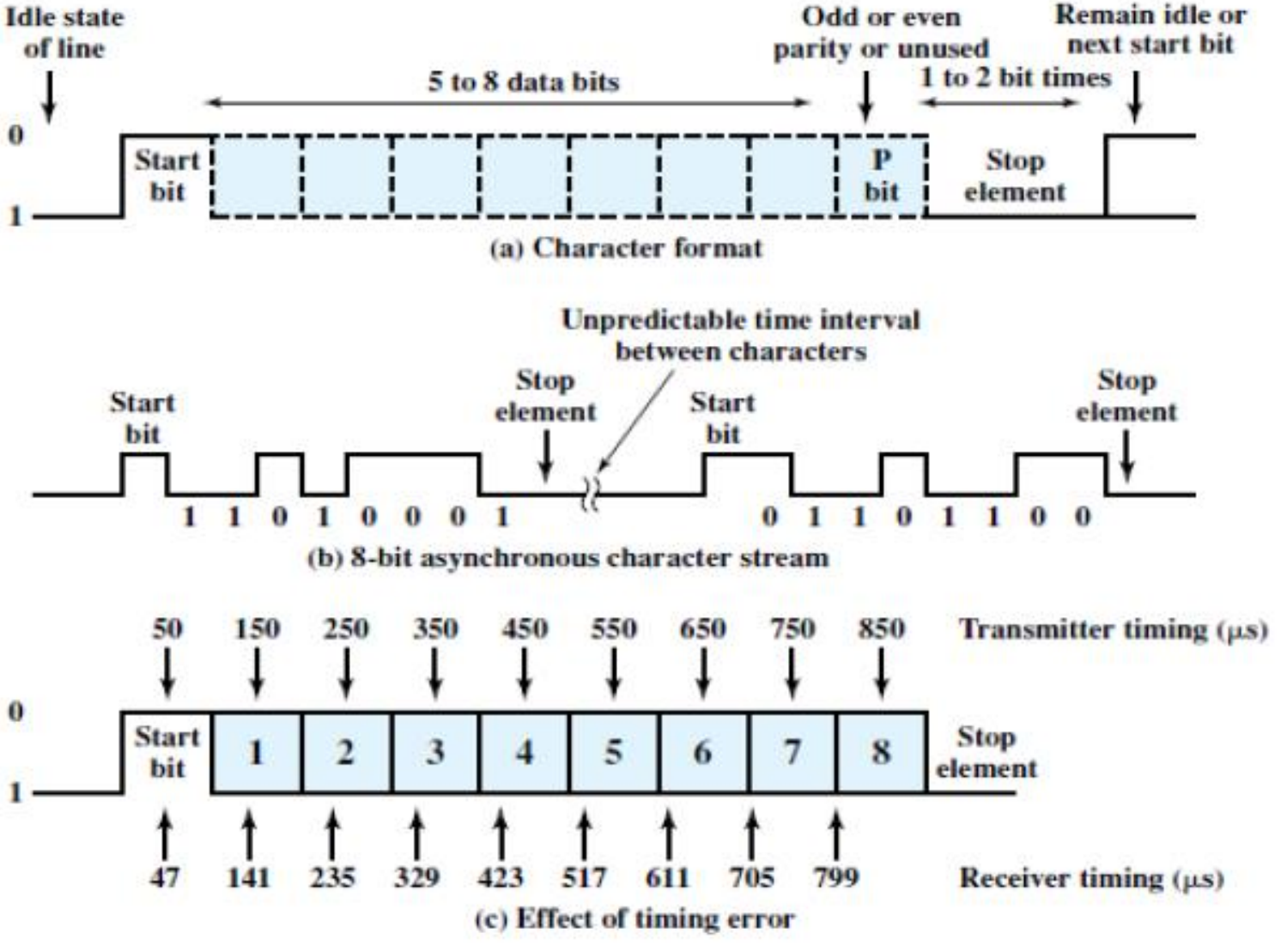
- Parallel transmission is used when:
  - a large amount of data is being sent;
  - the data being sent is time-sensitive;
  - and the data needs to be sent quickly.

Asynchronous Transmission

- Bits are sent on a character-by-character basis
- Independent clocks at sender & receiver (but matching reasonably)
- Clock sync: receiver resync's its clock every character
- Block sync: each character is "bracketed" by start & stop bits
- Advantage: simple & cheap, usable up to ~20 kbps
- Disadvantage: not very efficient


Stop Bit
Start Bit
Data
1|11011001|0
Direction Of Flow
Sender
011011|0
1|11001101|0
1|10011100|0
1|1101
Receiver

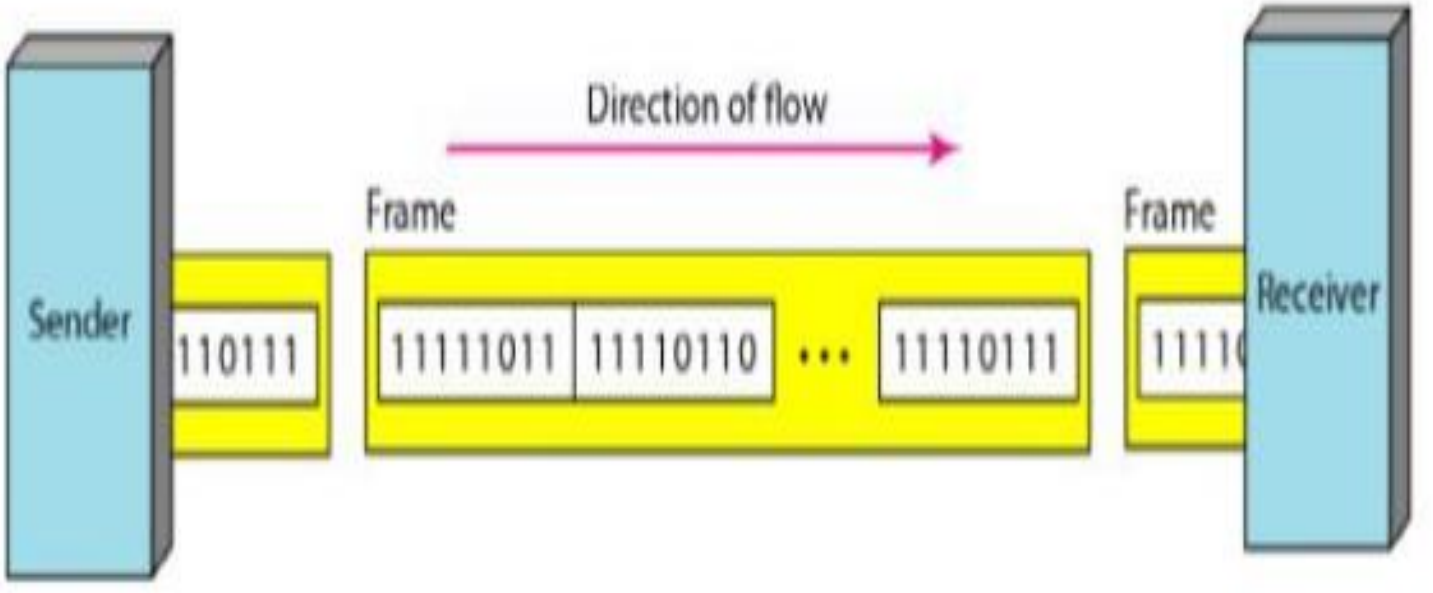
Asynchronous Transmission



Synchronous Transmission

- Sender & receiver have identical clocks
  - Separate clock line to carry clock signal from sender
  - "Self-clocking" transmission scheme
    - Biphase encoding (e.g. Manchester) ← digital sig.
    - Using carrier signal ← Analog sig.
  - Block sync:
    - Special preamble & postamble bit patterns used to indicate start/end of a block (frame)

Synchronous Transmission



Synchronous Transmission

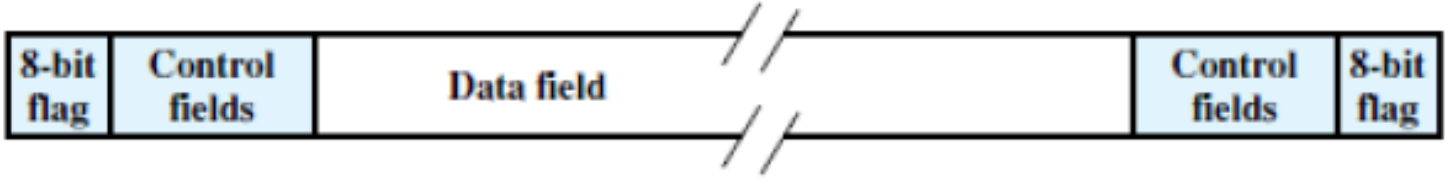


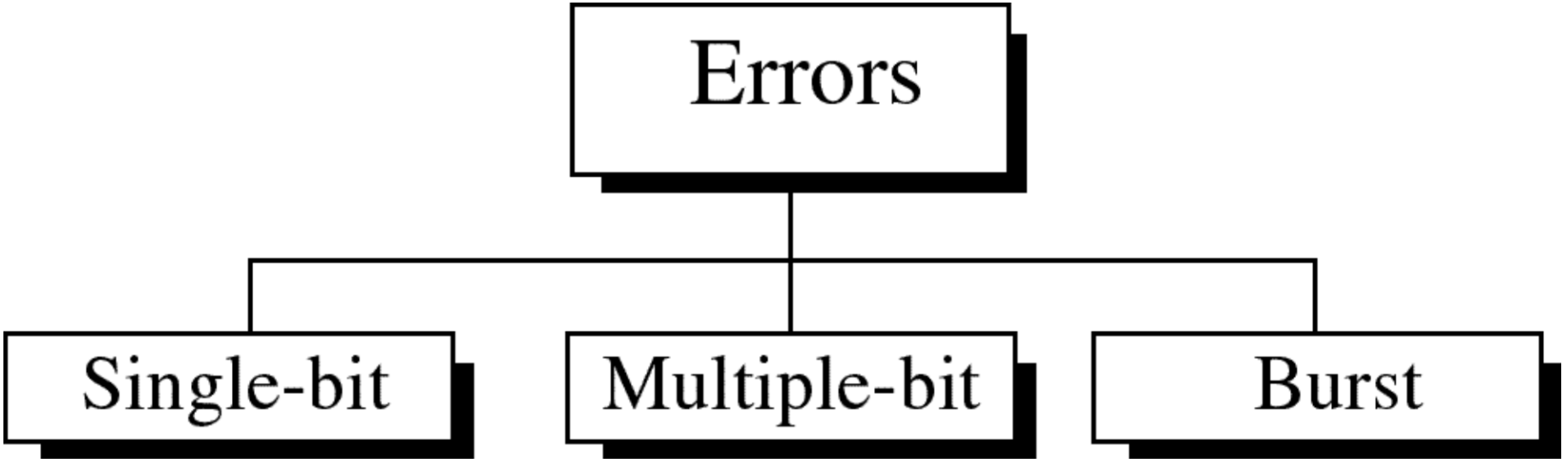
Figure 6.2 Synchronous Frame Format

Basic concepts

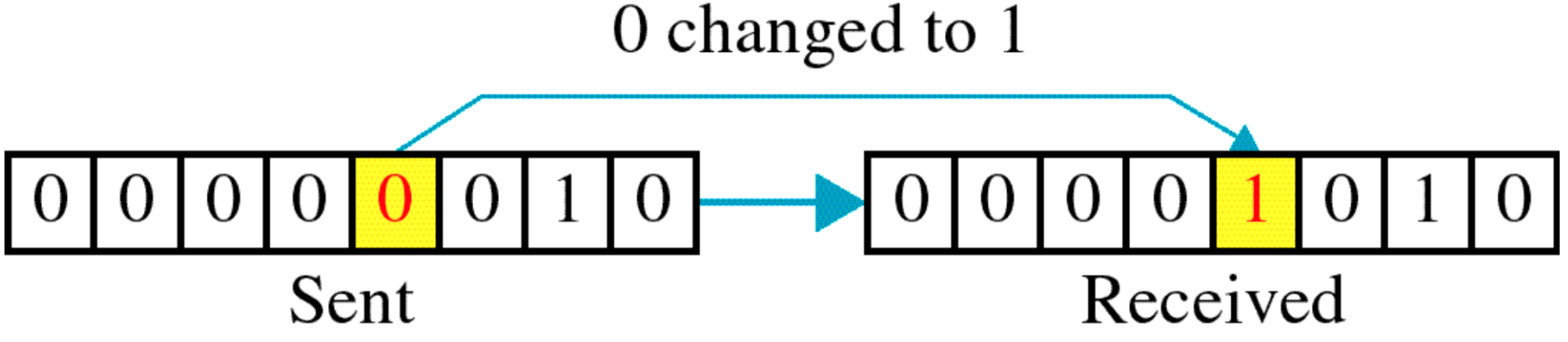
- Networks must be able to transfer data from one device to another with complete accuracy.
- Data can be corrupted during transmission.
- For reliable communication, errors must be detected and corrected.

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

Types of Errors



Single-bit error

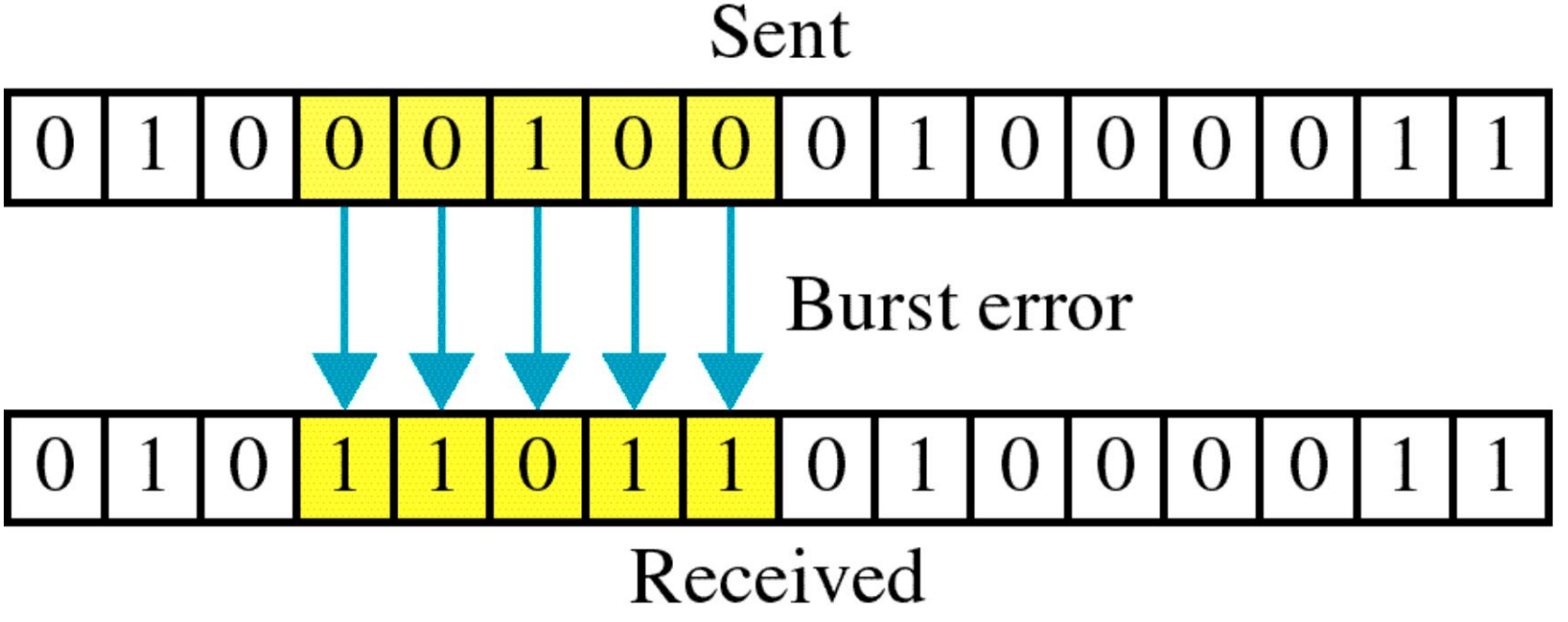


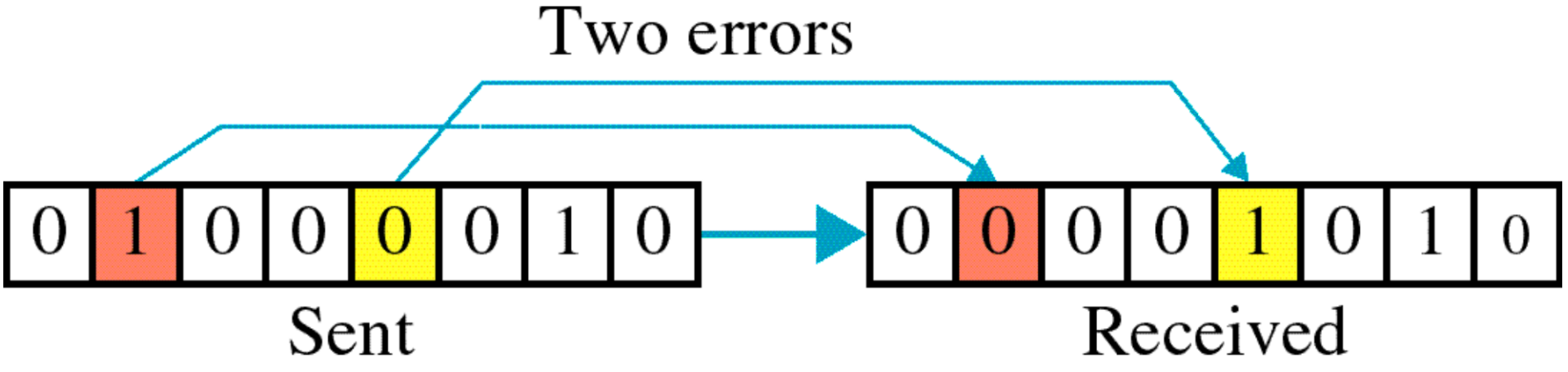
- **Single bit errors** are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare.
- However this kind of errors can happen in parallel transmission.

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


Two errors
0 1 0 0 0 0 1 0
Sent
0 0 0 0 1 0 1 0
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 detection

Error detection means to decide whether the received data is correct or not without having a copy of the original message.

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

- Assume that data are transmitted as one or more contiguous sequences of bits called frames.
- We define these probabilities with respect to errors in transmitted frames:
  - P_b-Probability that a bit is received in error; also known as the bit error rate(BER)
  - P_1-Probability that a frame arrives with no bit errors
  - P_2-Probability that, with an error-detecting algorithm in use, a frame arrives with one or more undetected errors
  - P_3-Probability that, with an error-detecting algorithm in use, a frame arrives with one or more detected bit errors but no undetected bit errors

To express the remaining probabilities, assume the probability that any bit is in error P_b is constant and independent for each bit.

Then we have

$$P_1 = (1 - P_b)^F$$

$$P_2 = 1 - P_1$$

Where F is the number of bits per frame.

In words,

- The probability that a frame arrives with no bit errors decreases when the probability of a single bit error increases.
- The probability that a frame arrives with no bit errors decreases with increasing frame length.
- The longer the frame, the more bits it has and the higher the probability that one of these is in error.

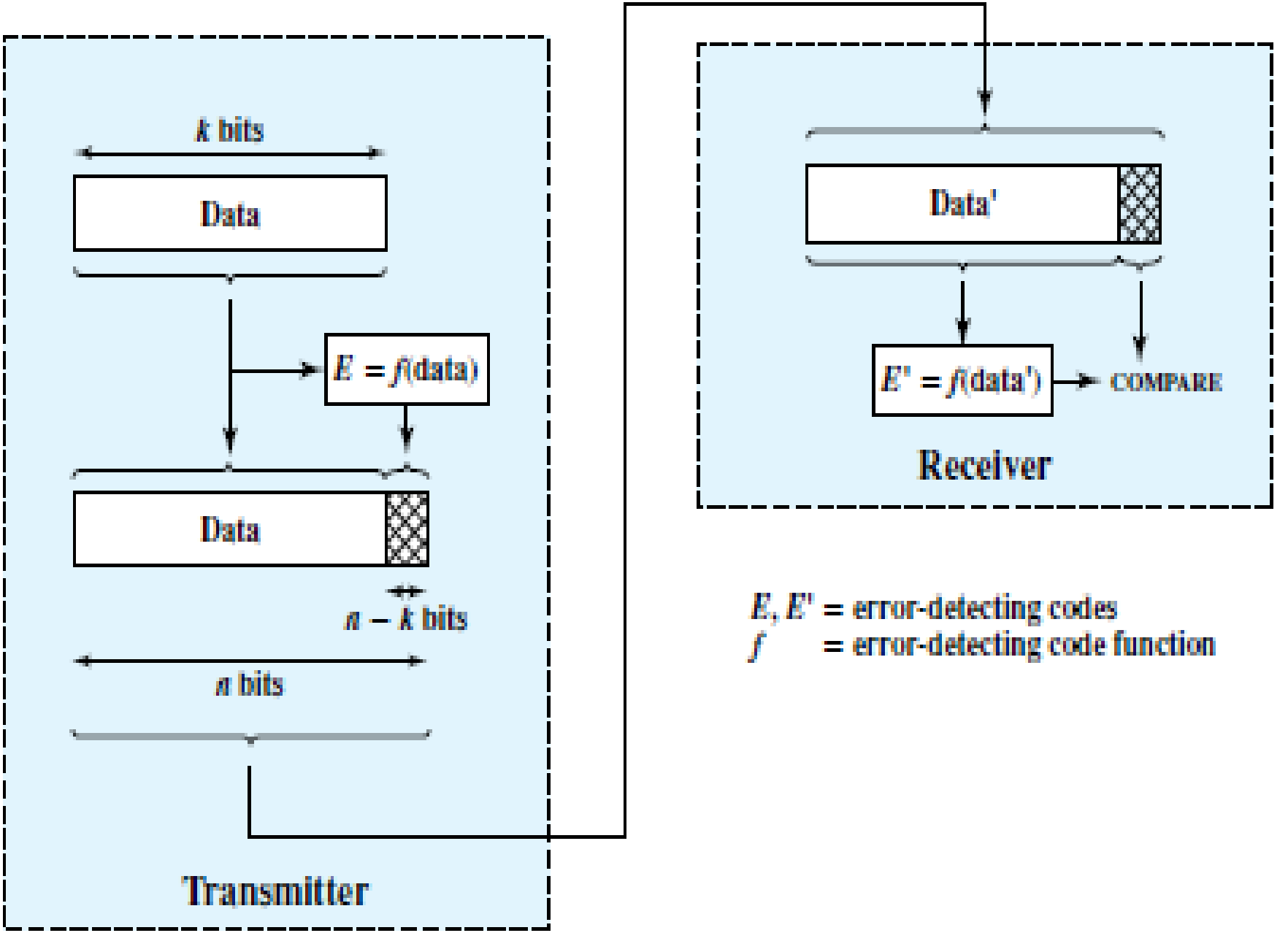
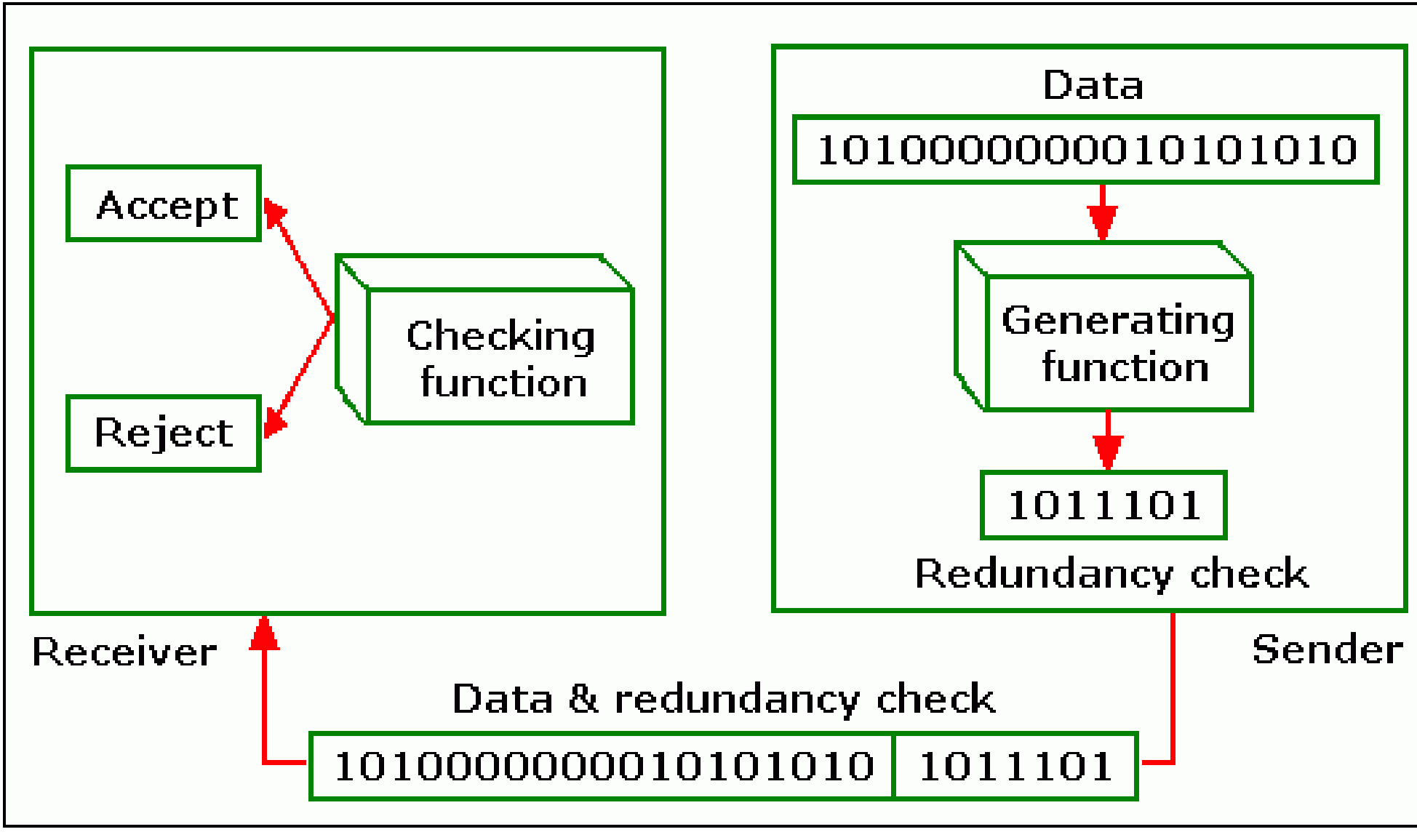
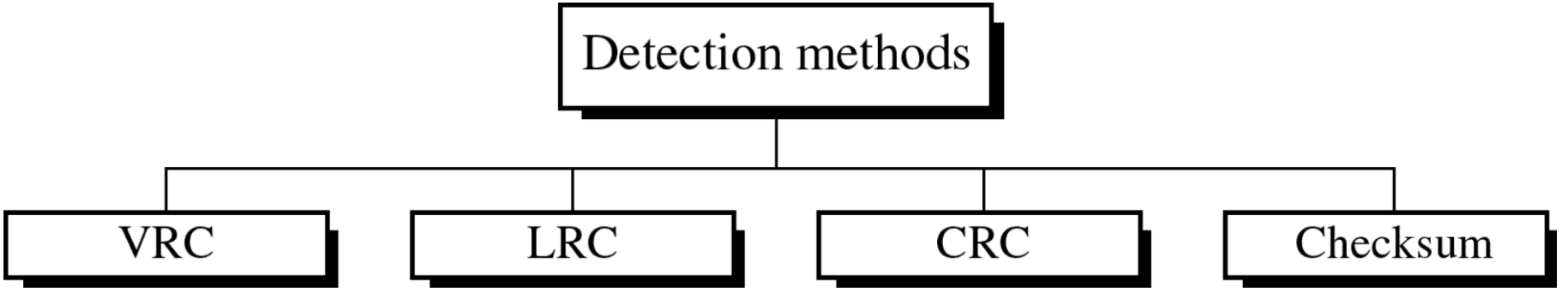


Figure 6.3 Error Detection Process

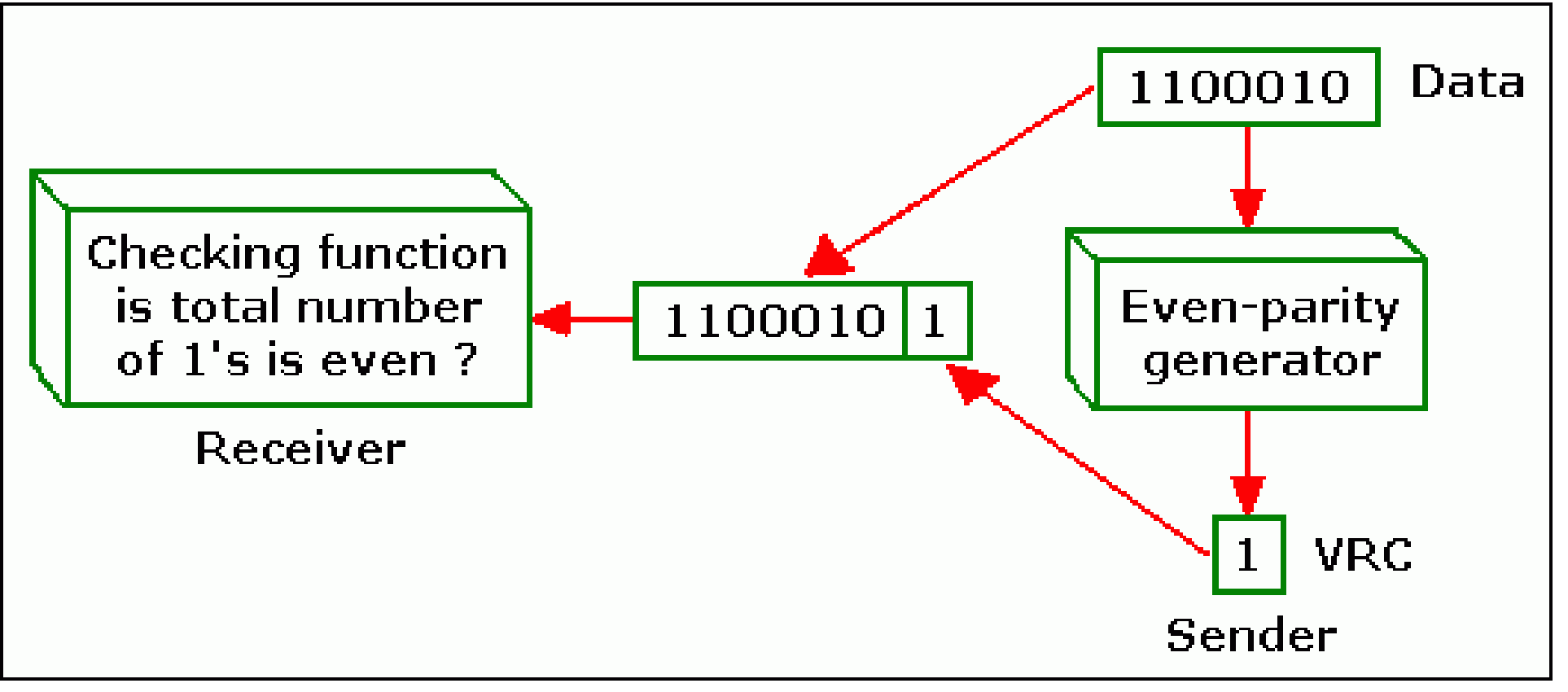
Error detecting code(Check bits/Redundancy check)



Four types of redundancy checks are used in data communications



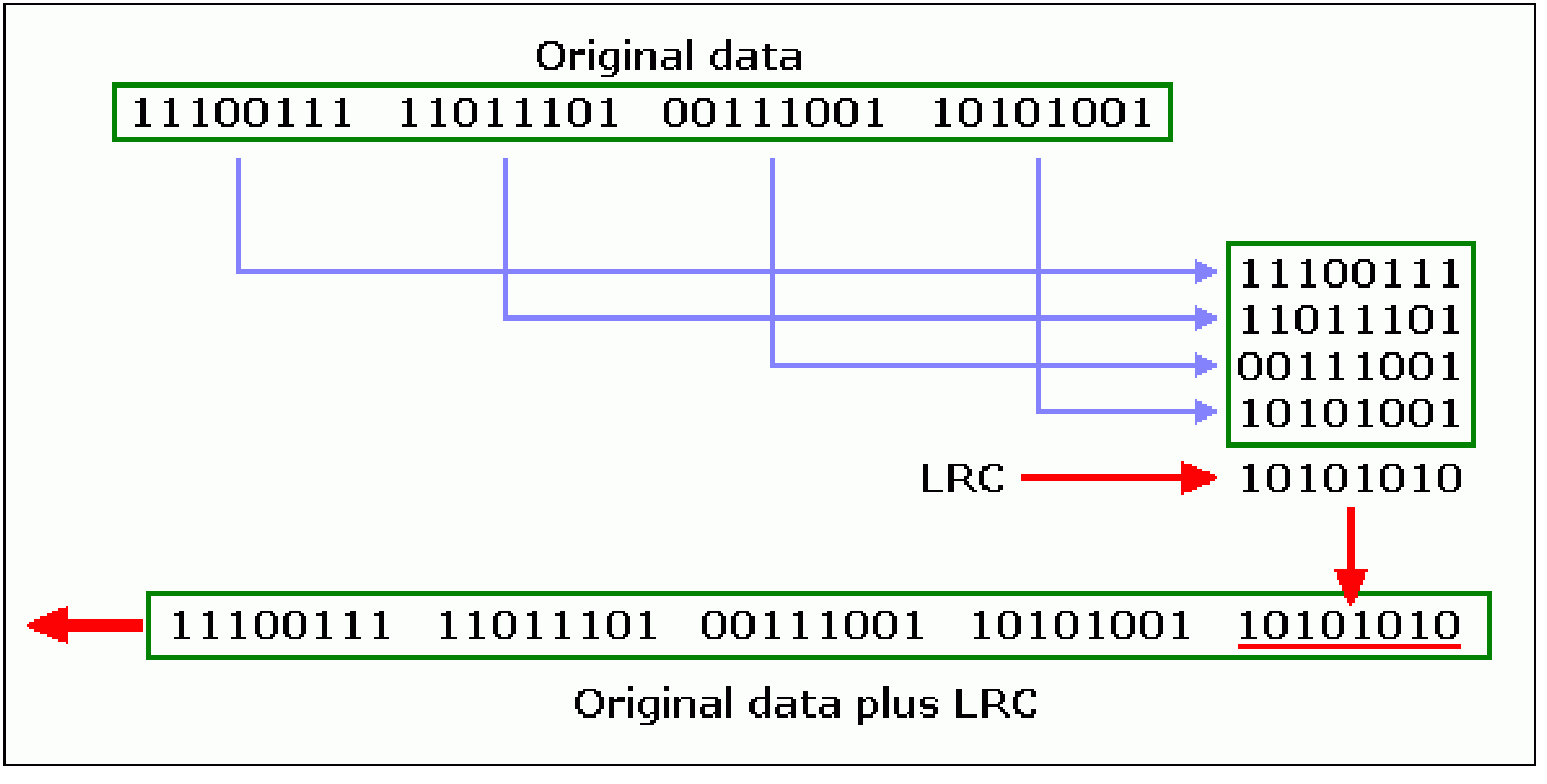
Vertical Redundancy Check
VRC



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



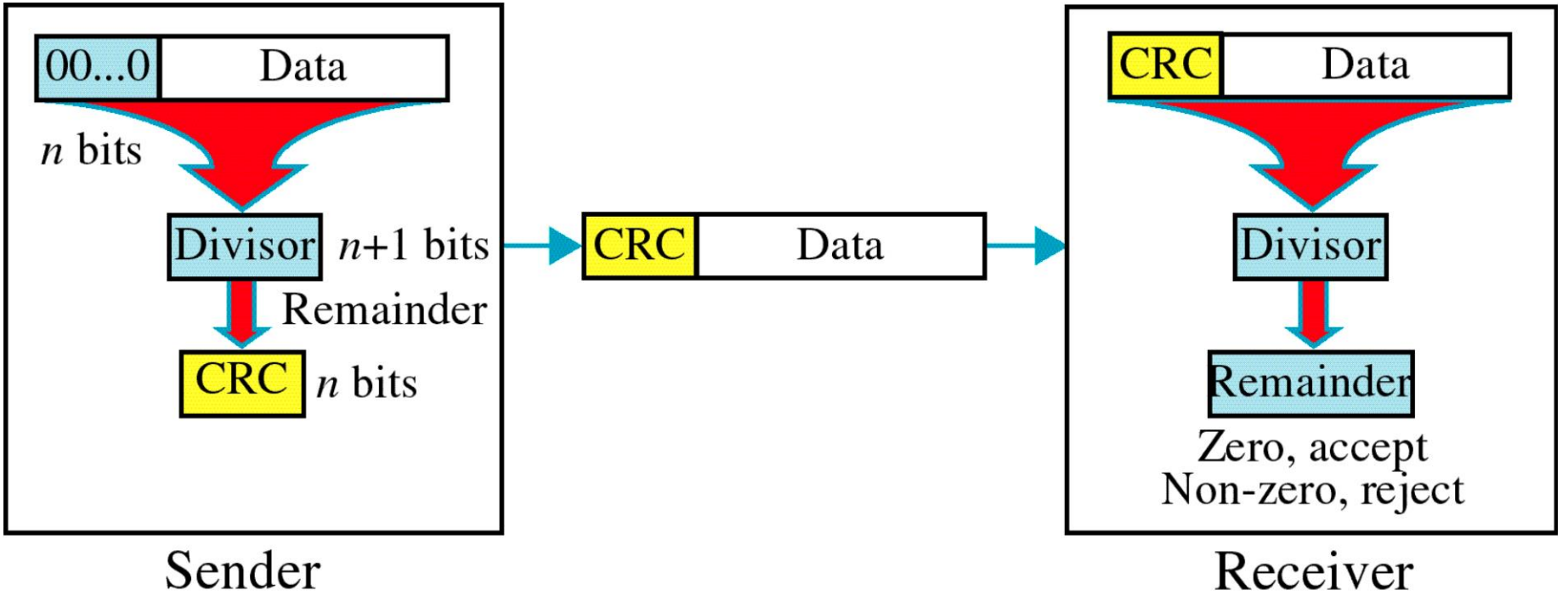
Performance

- 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.

Two Dimensional Parity Check
VRC and LRC

- VRC-Each Character Block
- LRC – Entire Block of codes

Cyclic Redundancy Check
CRC



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.

T = n-bit frame to be transmitted

D = k-bit block of data, or message, the first k bits of T

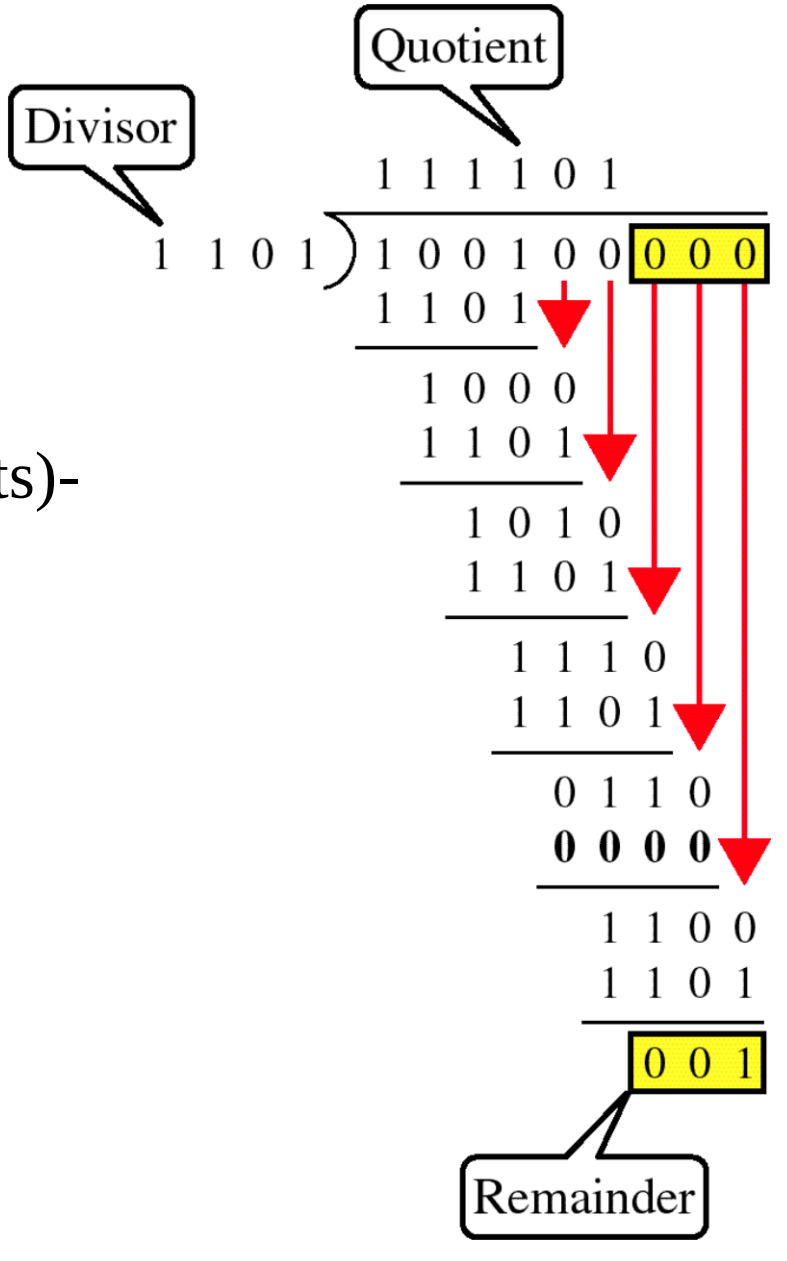
F = $(n - k)$-bit FCS, the last $(n - k)$ bits of T

P = pattern of $n - k + 1$ bits; this is the predetermined divisor

We would like T/P to have no remainder. It should be clear that

$$T = 2^{n-k}D + F$$

Modulo 2 arithmetic



Data:100100 (6 bits) –(k-bits)
Generated Pattern (P): 1101 (4 bits)-(n-bits)
Redundant bits (F) : ? (n-1)
(3 bits)

Transmitted :100100001 (n bits)
CRC :001 (n-k bits)

T = n-bit frame to be transmitted

D = k-bit block of data, or message, the first k bits of T

F = $(n - k)$-bit FCS, the last $(n - k)$ bits of T

P = pattern of $n - k + 1$ bits; this is the predetermined divisor

We would like T/P to have no remainder. It should be clear that

$$T = 2^{n-k}D + F$$

That is, by multiplying D by 2^{n-k}, we have in effect shifted it to the left by $n - k$ bits and padded out the result with zeroes. Adding F yields the concatenation of D and F, which is T. We want T to be exactly divisible by P. Suppose that we divide $2^{n-k}D$ by P:

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P} \tag{6.1}$$

There is a quotient and a remainder. Because division is modulo 2, the remainder is always at least one bit shorter than the divisor. We will use this remainder as our FCS. Then

$$T = 2^{n-k}D + R \tag{6.2}$$

Does this R satisfy our condition that T/P have no remainder? To see that it does, consider

$$\frac{T}{P} = \frac{2^{n-k}D + R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P}$$

Substituting Equation (6.1), we have

$$\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P}$$

However, any binary number added to itself modulo 2 yields zero. Thus

$$\frac{T}{P} = Q + \frac{R + R}{P} = Q$$

There is no remainder, and therefore T is exactly divisible by P.

Thus, the errors in an *n*-bit frame can be represented by an *n*-bit field with 1s in each error position.

The resulting frame T_r can be expressed as

$$T_r = T \oplus E$$

where

T = transmitted frame
E = error pattern with 1s in positions where errors occur
T_r = received frame
$\oplus$ = bitwise exclusive-OR(XOR)

If there is an error(E != 0) the receiver will fail to detect the error if and only if T_r is divisible by P, which is equivalent to E divisible by P.

Polynomial

A second way of viewing the CRC process is to express all values as polynomials in a dummy variable X, with binary coefficients.

The coefficients correspond to the bits in the binary number.

Arithmetic operations are again modulo 2

P = 11001 $P(X) = X^4 + X^3 + 1$

D = 110011 $D(X) = X^5 + X^4 + X + 1$

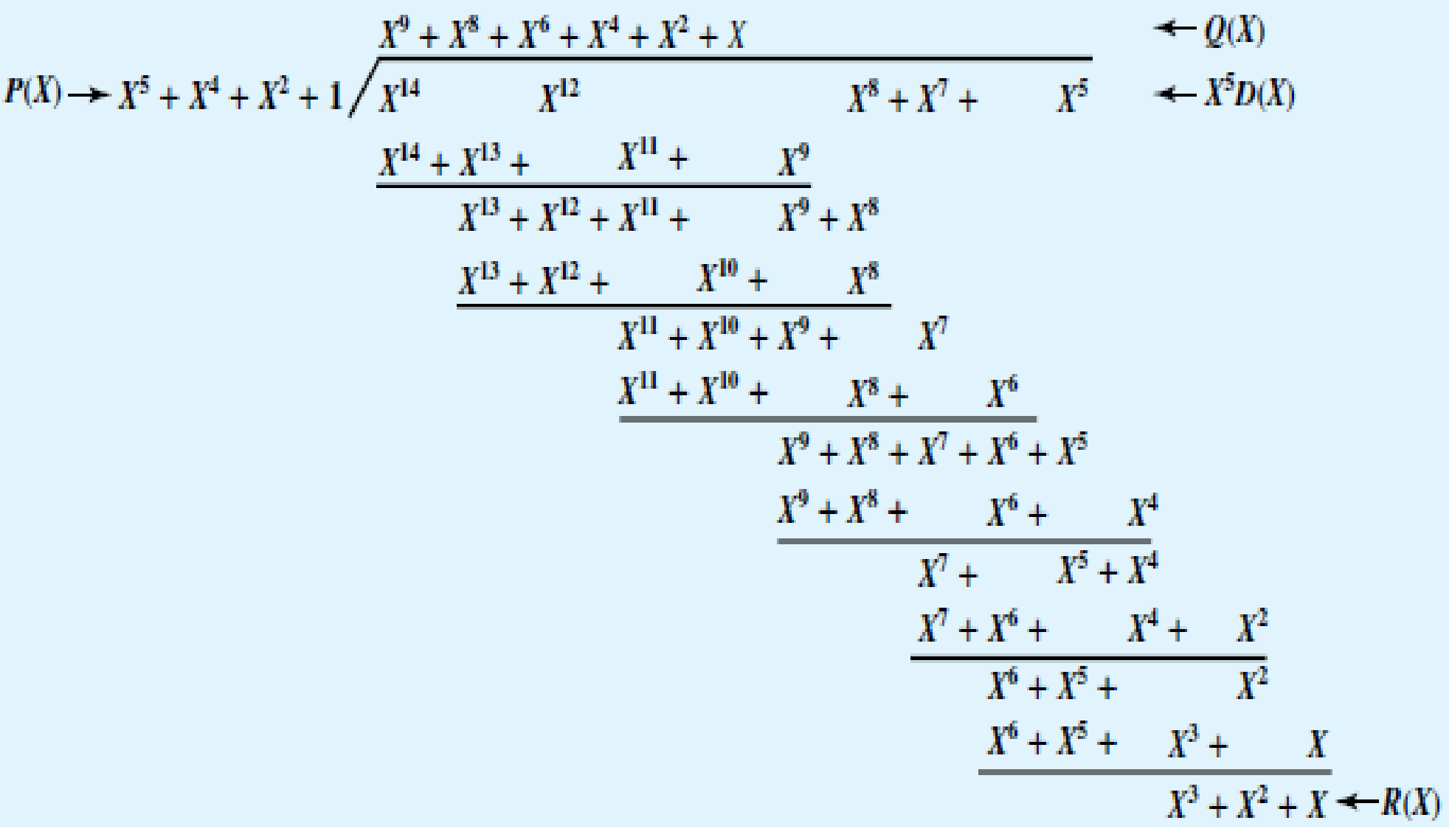
The CRC process can now be described as

$$\frac{X^{n-k}D(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$$

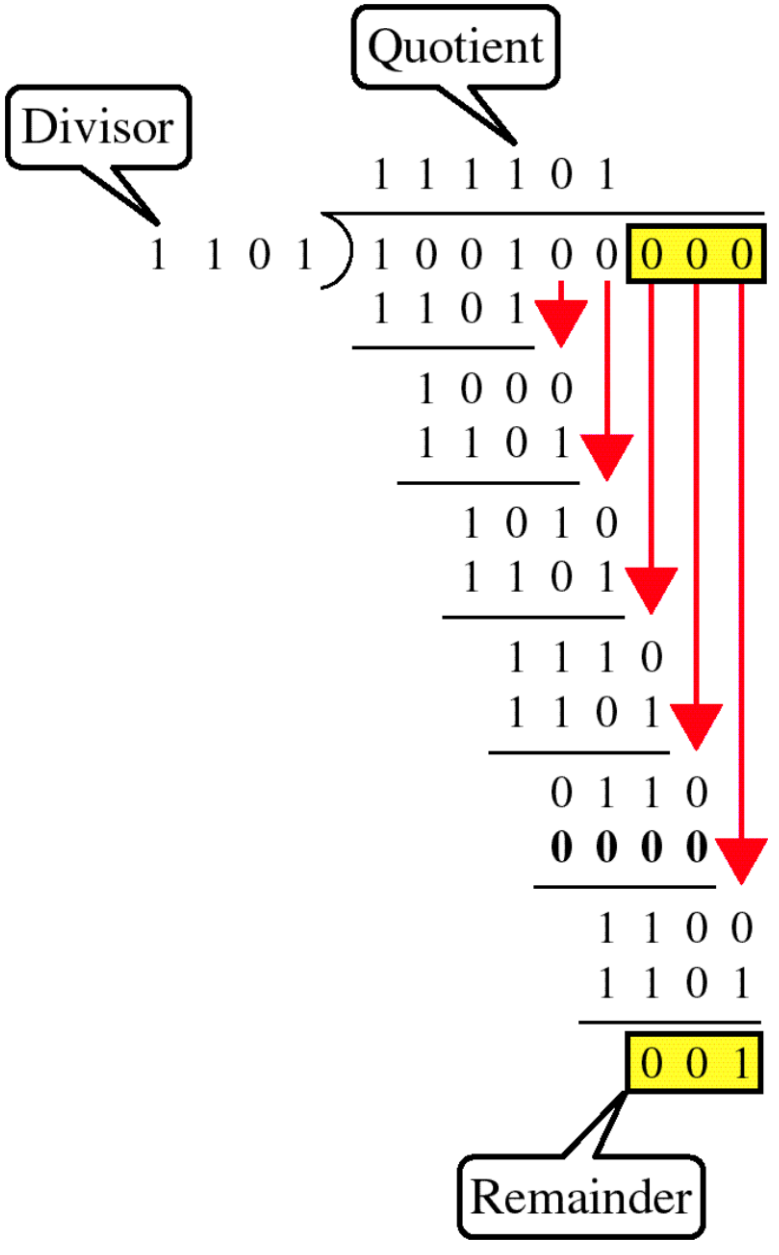
$$T(X) = X^{n-k}D(X) + R(X)$$

- D(X)=1010001101
- P(X)=110101

Polynomial and Divisor



Polynomial??


Quotient
Divisor
1 1 1 1 0 1
1 1 0 1) 1 0 0 1 0 0 0 0 0
1 1 0 1
1 0 0 0
1 1 0 1
1 0 1 0
1 1 0 1
1 1 1 0
1 1 0 1
0 1 1 0
0 0 0 0
1 1 0 0
1 1 0 1
0 0 1
Remainder

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

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

- An error E(X) will only be undetectable if it is divisible by P(X).

- It can be shown [PETE 61, RAMA88] that all of the following errors are not divisible by a suitably chosen P(X) and hence are detectable:
  - All single-bit errors, if $P(X)$ has more than one nonzero term
  - All double-bit errors, as long as $P(X)$ is a special type of polynomial, called a primitive polynomial, with maximum exponent L, and the frame length is less than $2^L - 1$.
  - Any odd number of errors, as long as $P(X)$ contains a factor (X+1)

- Any burst error for which the length of the burst is less than or equal to n-k; that is, less than or equal to the length of the FCS
- A fraction of error bursts of length n-k, the fraction equals $1-2^{-(n-k-1)}$.
- A fraction of error bursts of length n-k+1 the fraction equals $1-2^{-(n-k)}$

Digital Logic

- The CRC process can be represented by, and indeed implemented as, a dividing circuit consisting of XOR gates and a shift register.
- The shift register is a string of 1-bit storage devices.
- Each device has an output line, which indicates the value currently stored, and an input line.
- At discrete time instants, known as clock times, the value in the storage device is replaced by the value indicated by its input line.
- The entire register is clocked simultaneously, causing a 1-bit shift along the entire register.

The circuit is implemented as follows:

1. The register contains $n - k$ bits, equal to the length of the FCS.
2. There are up to $n - k$ XOR gates.
3. The presence or absence of a gate corresponds to the presence or absence of a term in the divisor polynomial, $P(X)$, excluding the terms 1 and X^{n-k}.

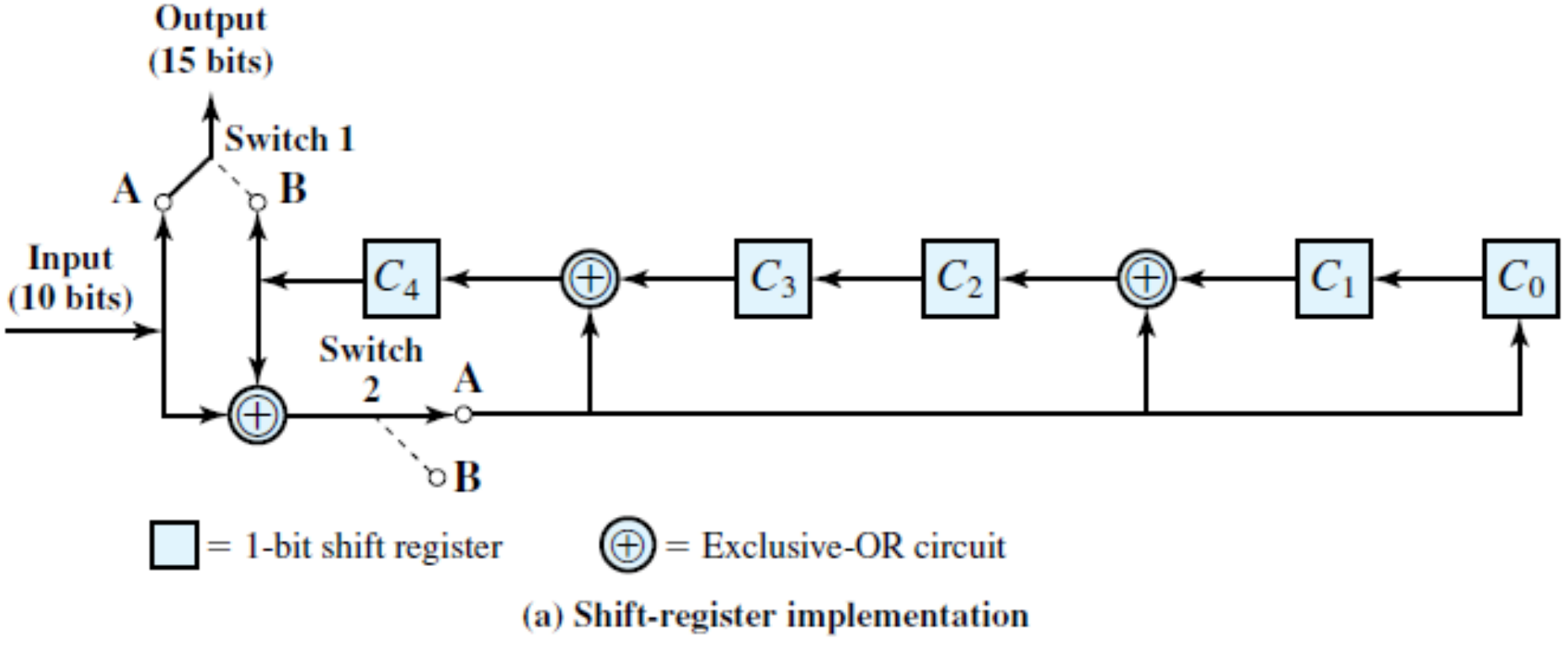
- Draw the circuit with shift registers and find the CRC.
- Data:1010
- Pattern :1011

- Draw the circuit with shift registers for dividing the polynomial.

- P(X):$X^5+X^4+X^2+1$
- D(X): $X^9+X^7+X^3+X^2+1$

P(X):$X^5+X^4+X^2+1$

D(X): $X^9+X^7+X^3+X^2+1$



(a) Shift-register implementation

(a) Shift-register implementation

	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3 \oplus I$	$C_4 \oplus C_1 \oplus I$	$C_4 \oplus I$	I = input	
Initial	0	0	0	0	0	1	1	1	1	
Step 1	1	0	1	0	1	1	1	1	0	
Step 2	1	1	1	1	1	1	1	0	1	
Step 3	1	1	1	1	0	0	0	1	0	
Step 4	0	1	0	0	1	1	0	0	0	Message
Step 5	1	0	0	1	0	1	0	1	0	be sent
Step 6	1	0	0	0	1	0	0	0	1	
Step 7	0	0	0	1	0	1	0	1	1	
Step 8	1	0	0	0	1	1	1	1	0	
Step 9	1	0	1	1	1	0	1	0	1	
Step 10	0	1	1	1	0					

(b) Example with input of 1010001101

Figure 6.5 Circuit with Shift Registers for Dividing by the Polynomial $X^5 + X^4 + X^2 + 1$

Tutorial 6

- D= 10111001
- P= 11001

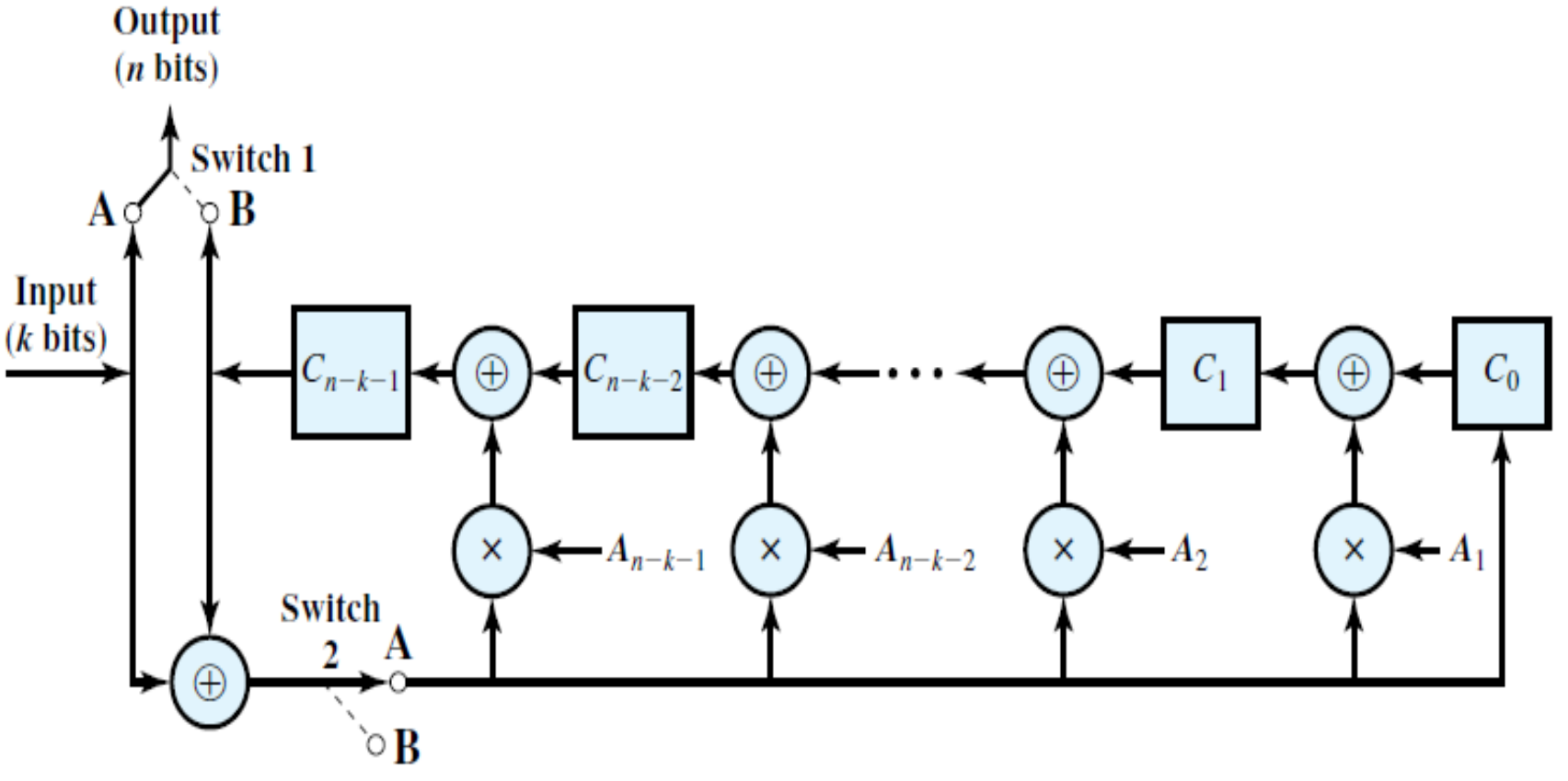
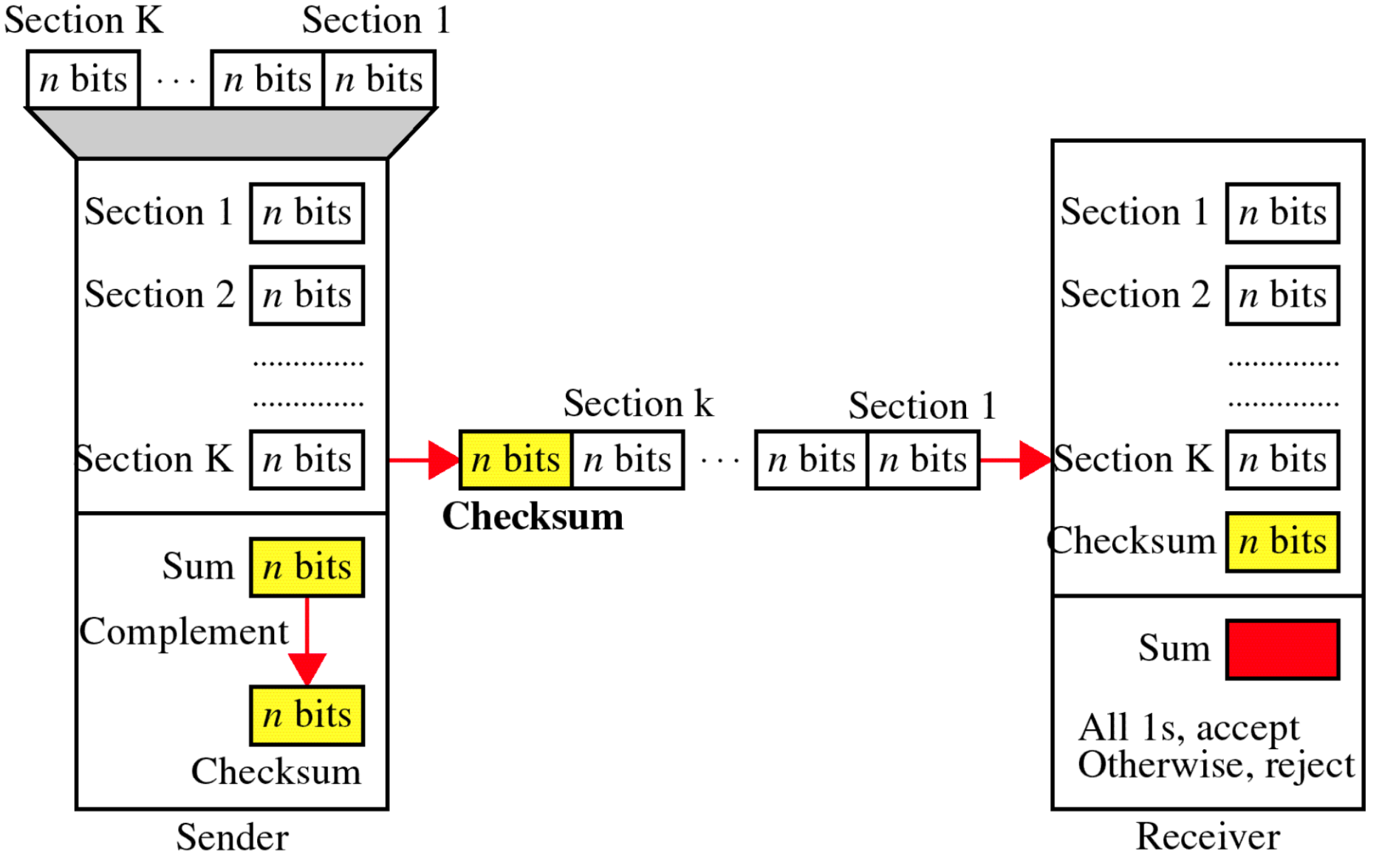


Figure 6.6 General CRC Architecture to Implement Divisor $(1 + A_1X + A_2X^2 + \cdots + A_{n-1}X^{n-k-1} + X^{n-k})$

Tutorial 6

- A wants to send a message "**HI**" to B. The divisor used is 11001. What is the codeword that is sent to B? The Hexadecimal equivalent of H is 0x48 (110000) and I is 0x49 (110001). (Use polynomials)
- "1100 0011 0001 1101" is received by B. B checks for error using hardware implementation. What is the conclusion? (Divisor=11001)

Checksum


Section K
Section 1
n bits
n bits
n bits
Section 1
n bits
Section 2
n bits
Section K
n bits
Sum
n bits
Complement
n bits
Checksum
Sender
Section k
Section 1
n bits
n bits
n bits
n bits
Checksum
Section 1
n bits
Section 2
n bits
Section K
n bits
Checksum
n bits
Sum
All 1s, accept
Otherwise, reject
Receiver

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.

- Find the checksum at sender aand receiver for the following sequence.
  - 10110011 10101011 01011010 11010101

Tutorial 6

- What does "**cyclic**" signify in CRC?
- CRC versus Checksum versus Parity check?
- VRC versus LRC?

- Calculate (and verify) the checksum of “CHECKSUM”.

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

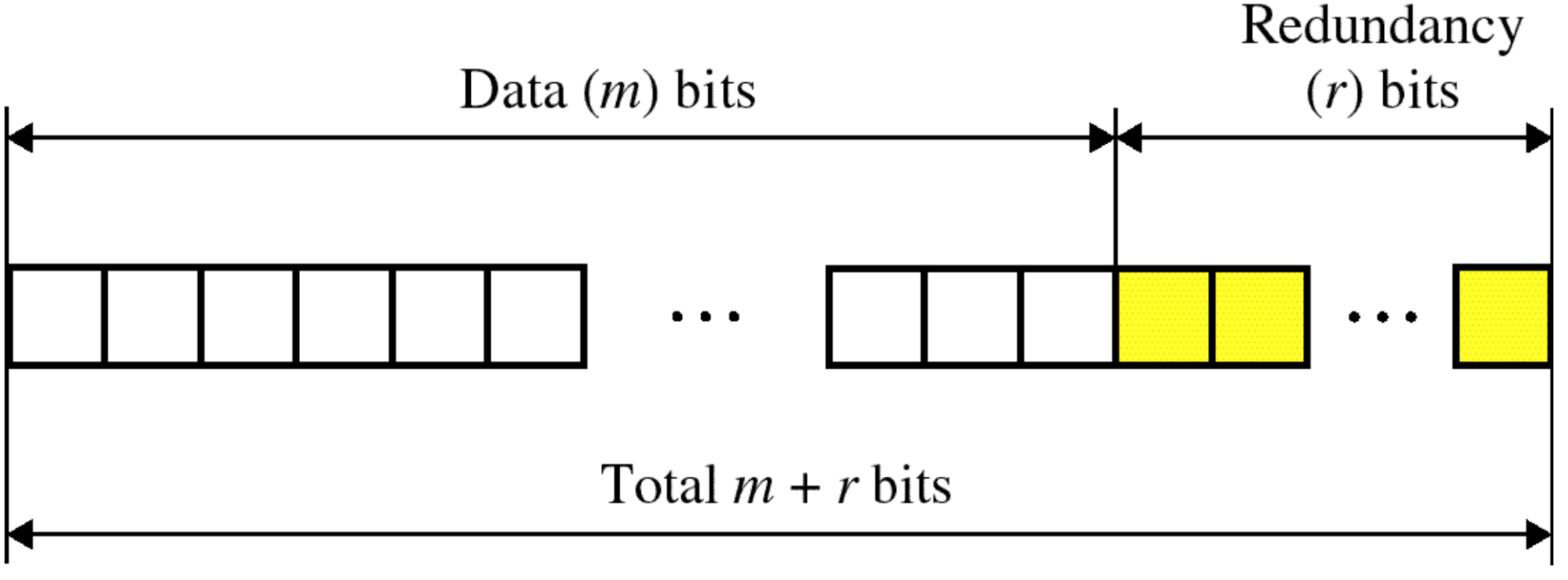
- Let data bits = m
- Redundancy bits = r

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

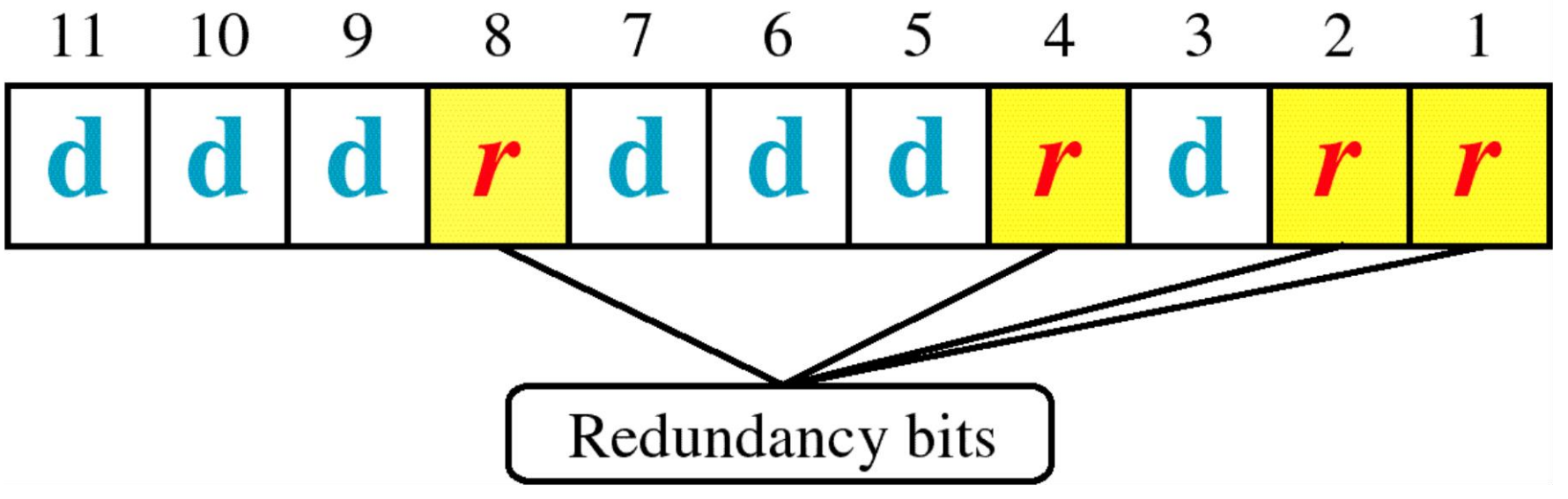
The value of r must satisfy the following relation:

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

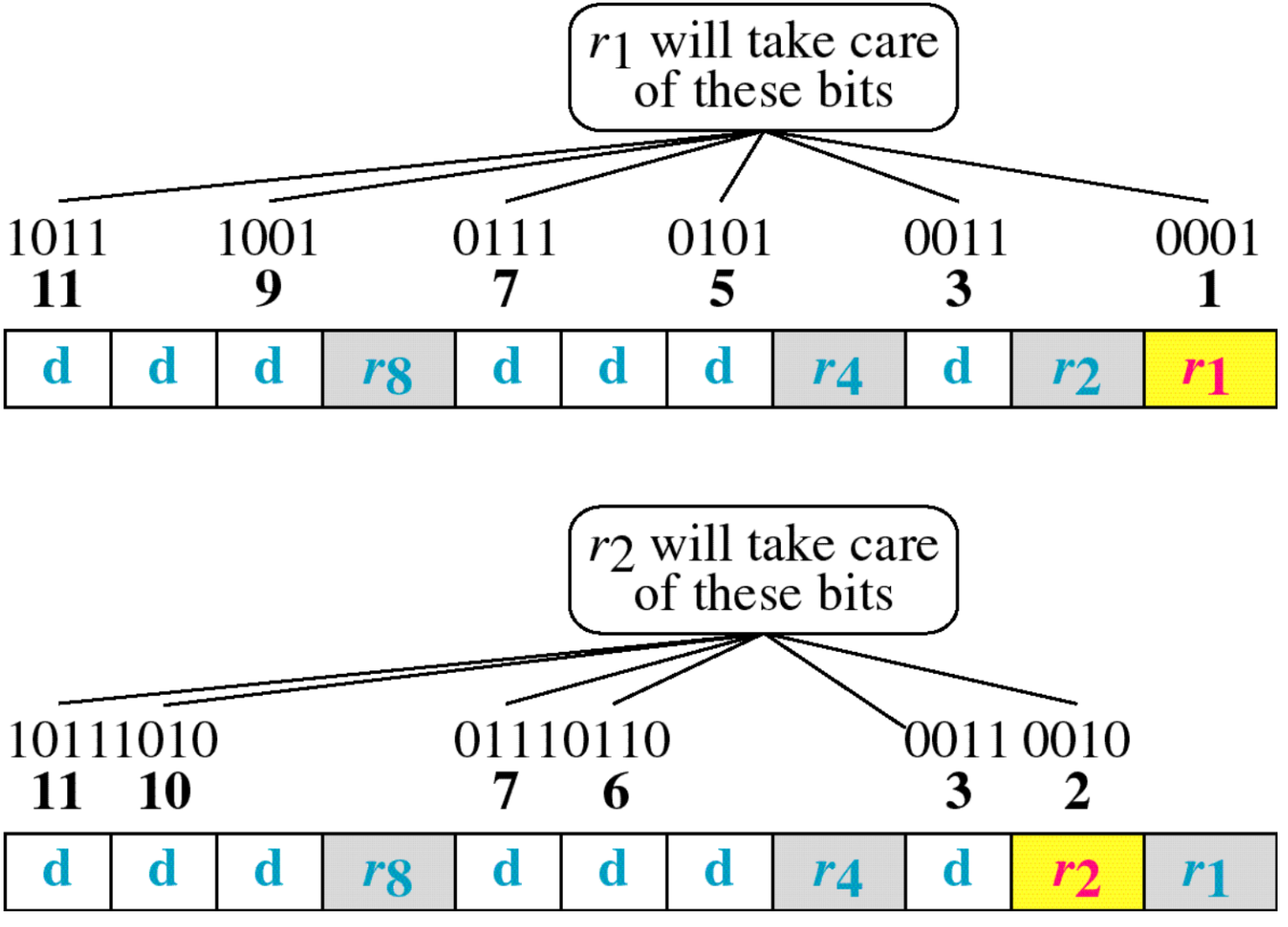
Error Correction



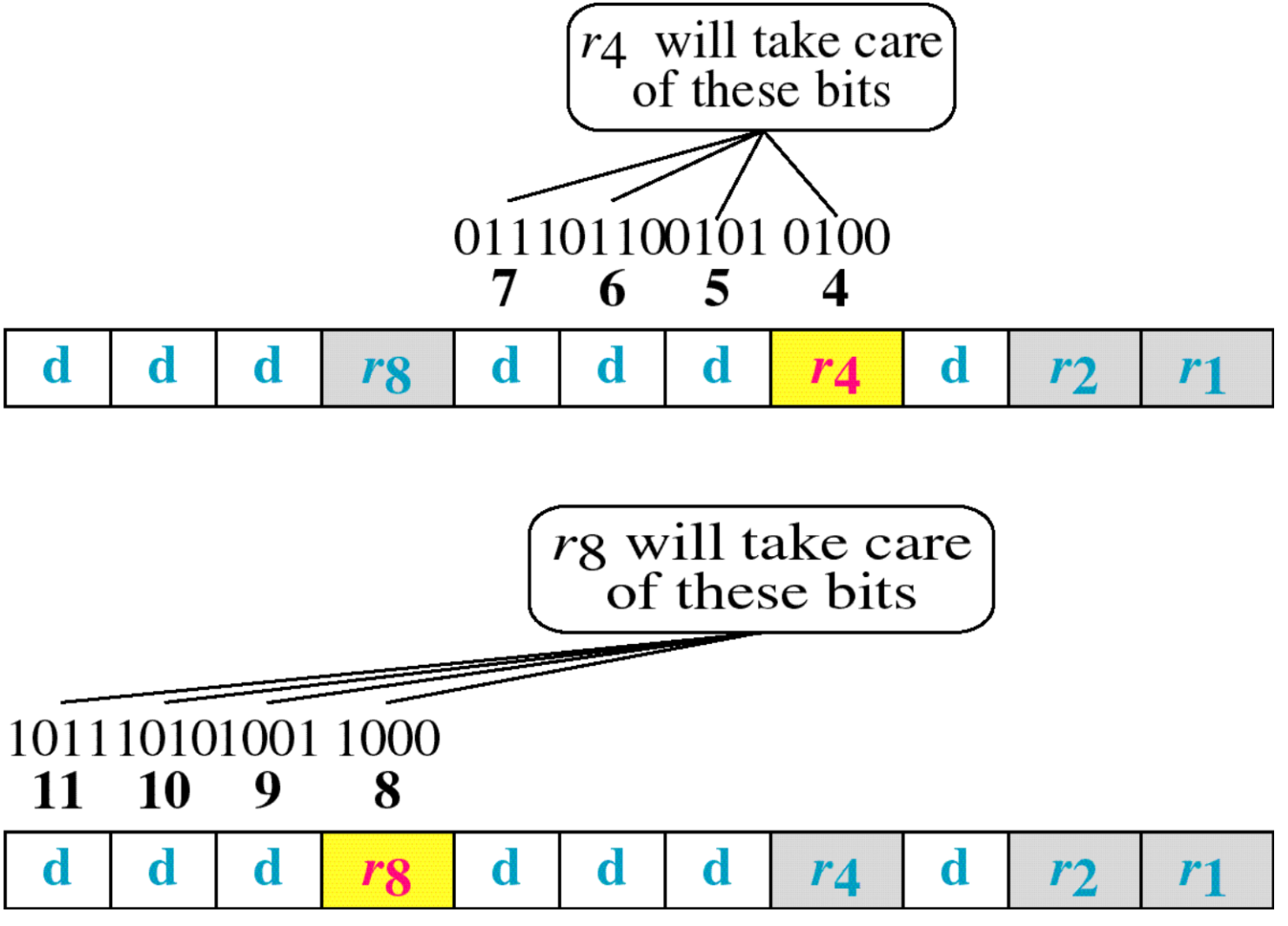
Hamming Code



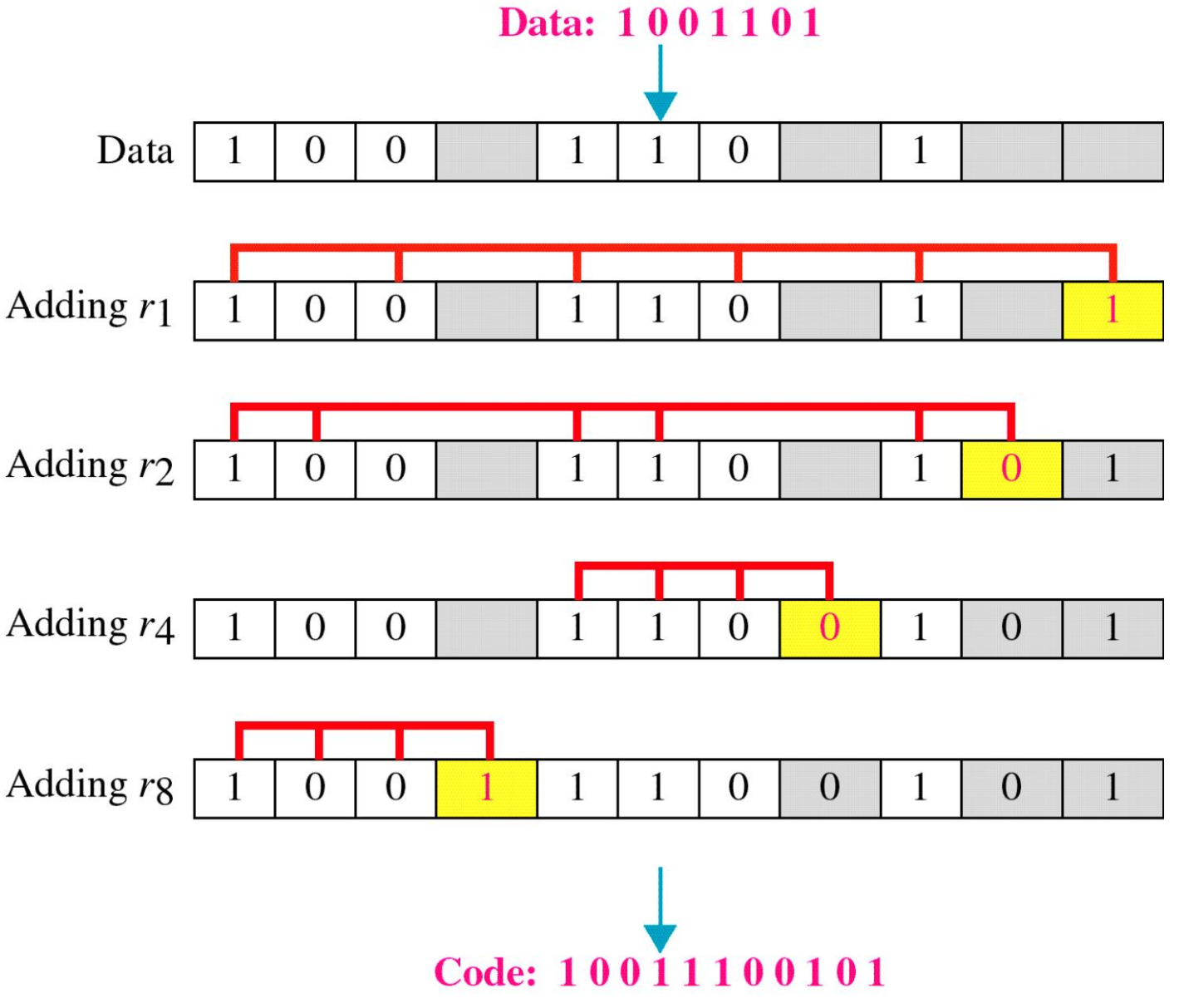
Hamming Code



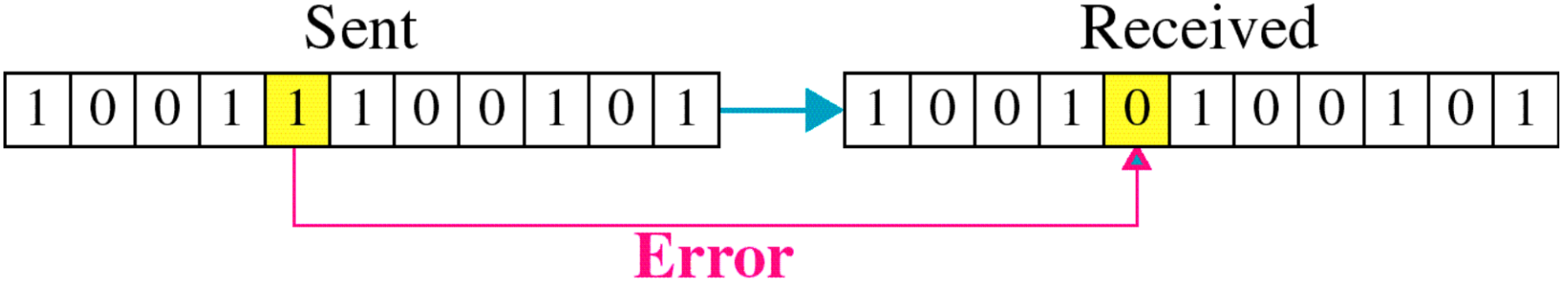
Hamming Code



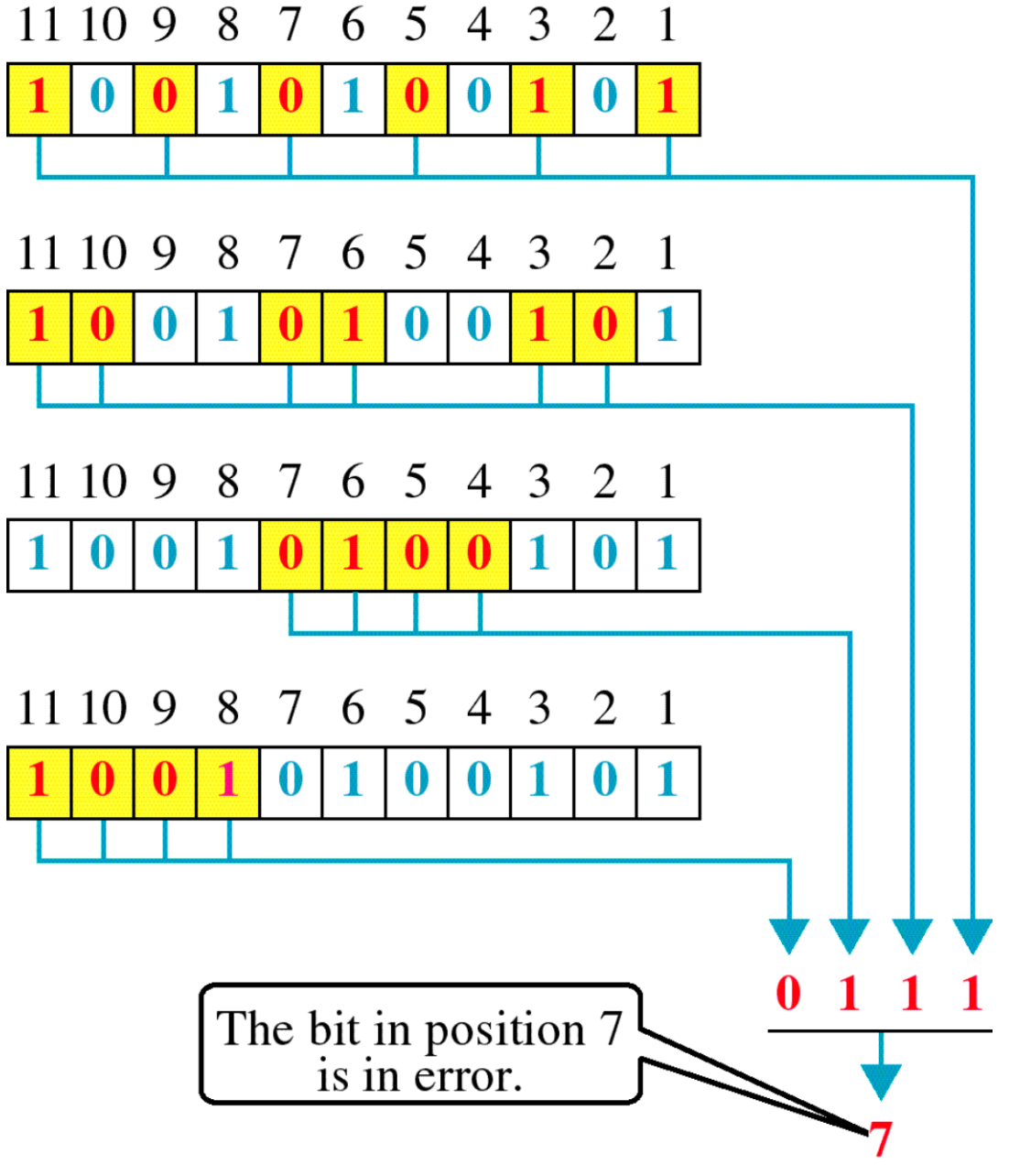
Example of Hamming Code



Single-bit error



Error Detection



Tutorial 7

- Calculate the number of parity(Redundancy bits).
  - If m=4
  - m=7

Tutorial 7

- A bit word 1011 is to be transmitted ..
  - Find the number of redundancy bits.
  - Construct the even parity seven bit hamming code for this data.

- If 7 bit hamming code is received as 1011011 .Assuming even parity state whether it is correct or not ?If not, locate the bit error.