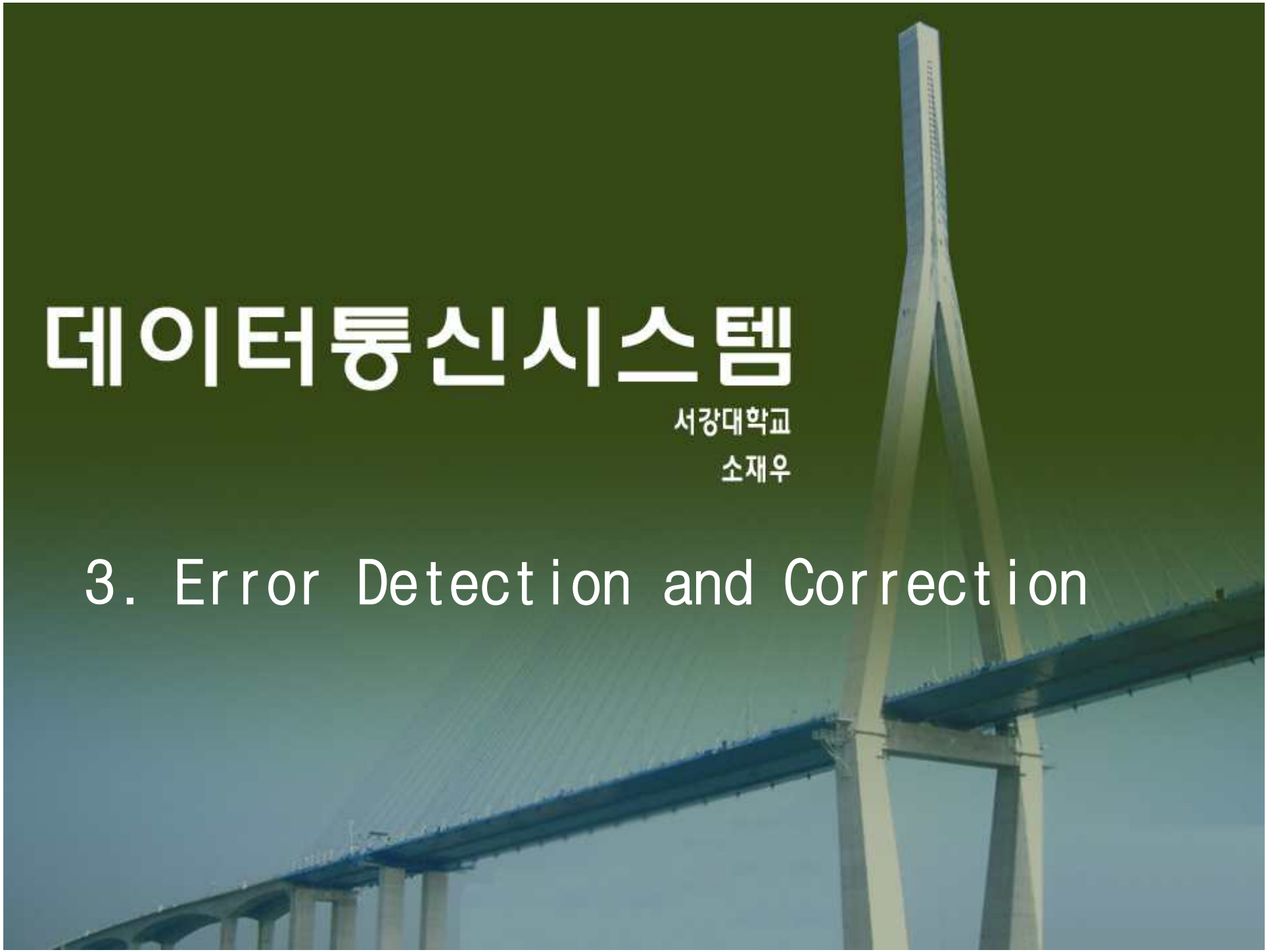


데이터통신시스템

서강대학교
소재우

3. Error Detection and Correction

학습에 앞서

▪학습개요

- 데이터통신에서 오류 검출 방식 및 정정 방식을 학습한다.

▪학습목표

- 오류 검출 및 정정 방식을 설명하고, 동작을 설명할 수 있어야 한다.
- Hamming Code, CRC, Checksum 방식과 동작을 설명할 수 있어야 한다.

1. Introduction

Let us first discuss some issues related, directly or indirectly, to error detection and correction.

Topics discussed in this section:

Types of Errors
Redundancy
Detection Versus Correction
Forward Error Correction Versus Retransmission
Coding
Modular Arithmetic

1. Introduction

- Type of Errors
  - Single-bit Error: only 1 bit of a given data unit is changed from 1 to 0 or from 0 to 1
  - Burst Error: 2 or more bits in the data unit have changed.

Figure 10.1 *Single-bit error*

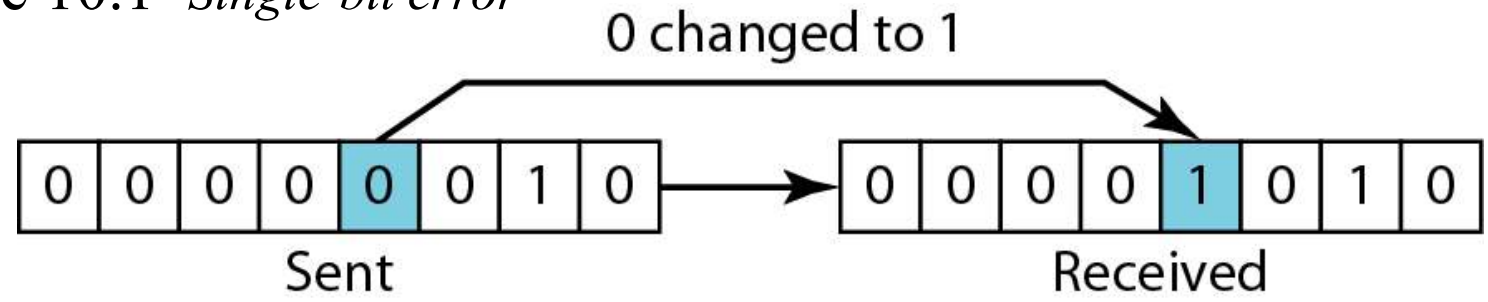
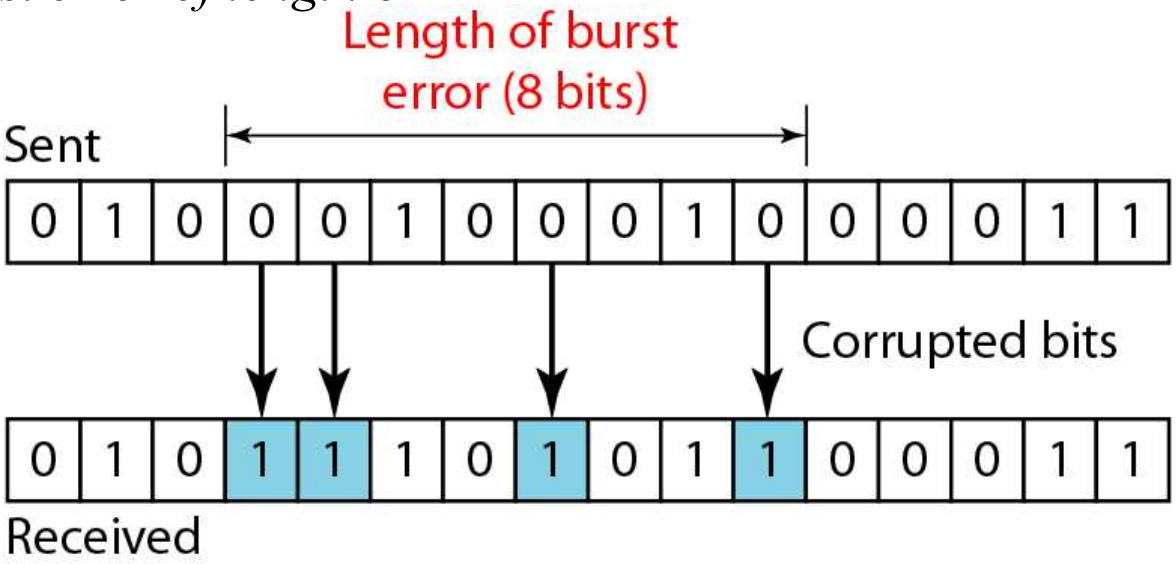


Figure 10.2 *Burst error of length 8*



1. Introduction

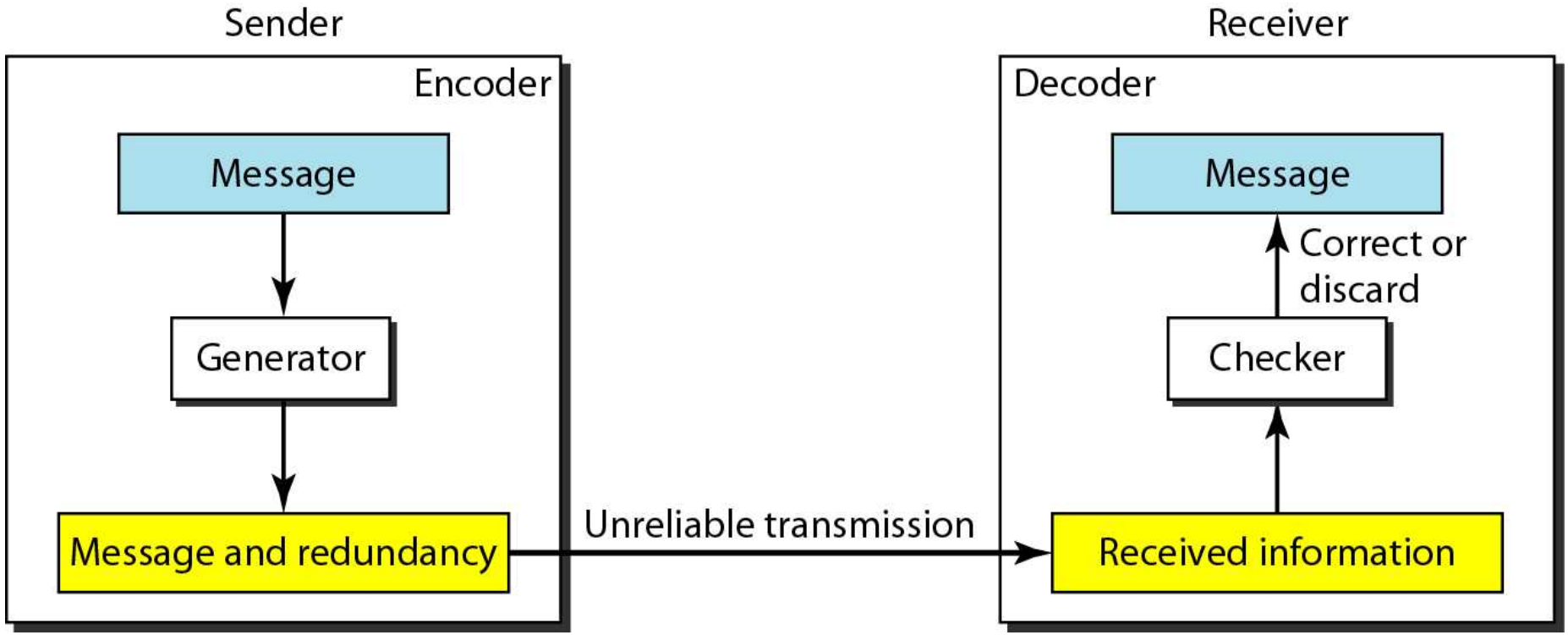
- Redundancy
  - To detect or correct errors, we need to send extra (Redundancy) bit with data.

- Detection Versus Correction
  - The correction of errors is more difficult than the detection. In error detection, we are looking only to see if any error has occurred.

- Forward Error Correction Versus Retransmission
  - Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. If the number of errors is small, this is possible.
  - Retransmission: The receiver detects the occurrence of an error and asks the sender to resend the message.

- Coding
  - Redundancy is achieved through various coding schemes.
  - We can divide coding schemes into two broad categories : block coding and convolution coding.

Figure 10.3 *The structure of encoder and decoder*



1. Introduction

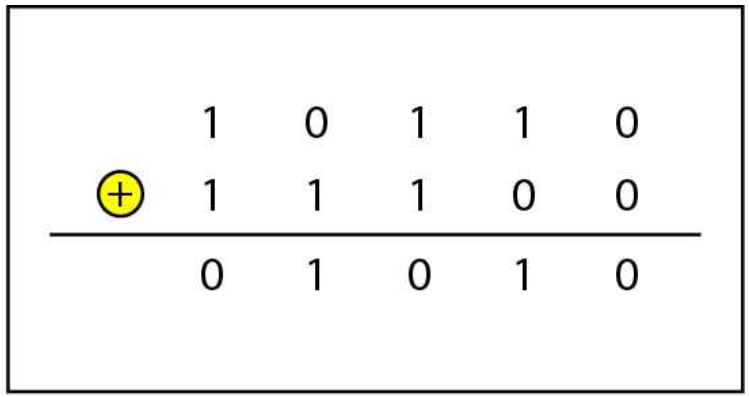
- Modular Arithmetic
  - Discuss a concept basic to computer science in general and to error detection and correction in particular: modular arithmetic
  - In modulo-N arithmetic, we use only the integers in the range 0 to N-1, inclusive.

  - Modulo-2 Arithmetic
    - Adding: 0+0=0 0+1=1 1+0=1 1+1=0
    - Subtracting: 0-0=0 0-1=1 1-0=1 1-1=0

Figure 10.4 *XORing of two single bits or two words*

a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.



c. Result of XORing two patterns

2. Block Coding

In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords.

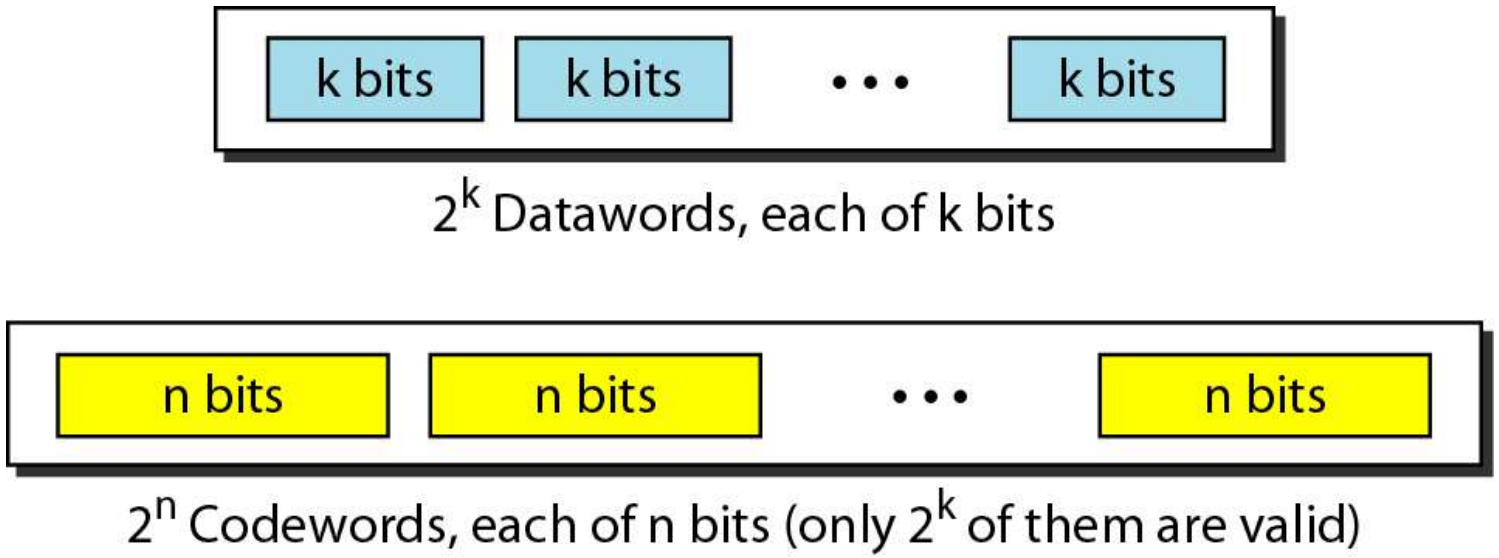
Topics discussed in this section:

Error Detection
Error Correction
Hamming Distance
Minimum Hamming Distance

2. Block Coding

- We divide our message into blocks, each of k bits, called datawords.
- We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords.
- We have 2^n-2^k codewords that are not used. We call these codewords invalid or illegal.

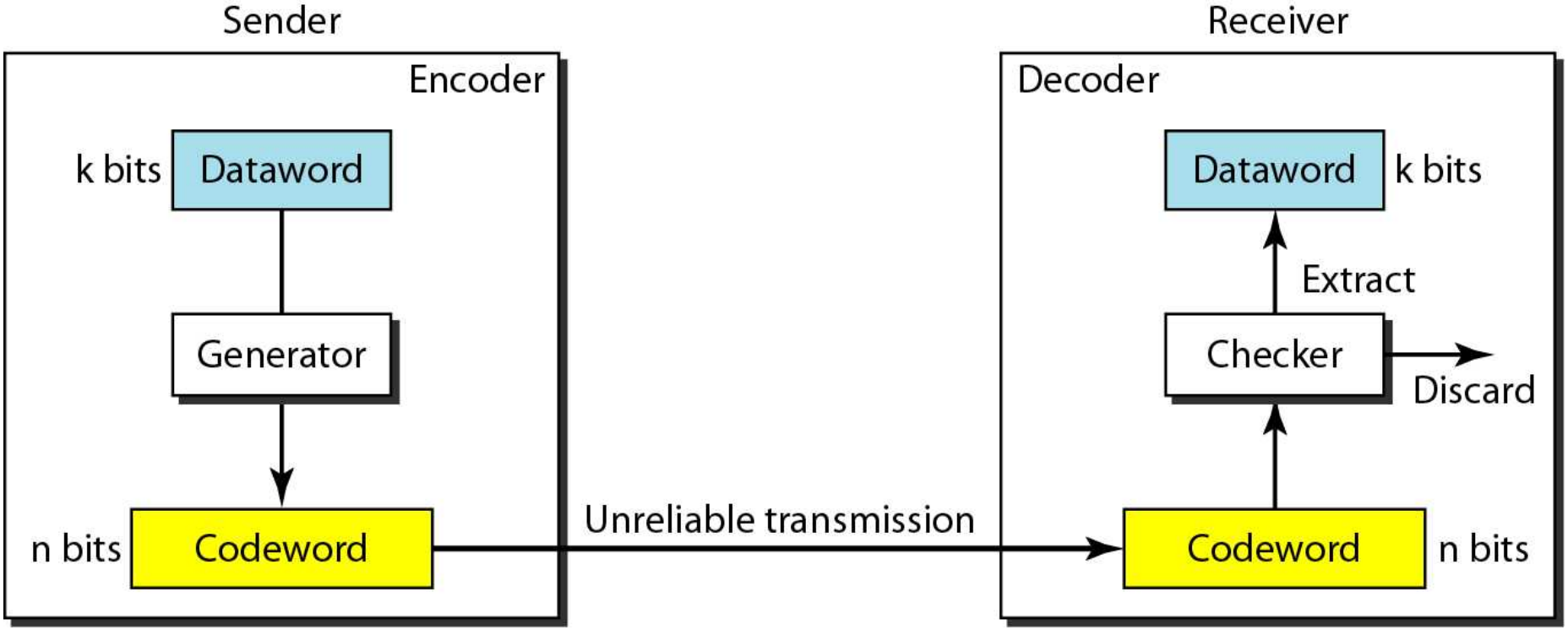
Figure 10.5 *Datawords and codewords in block coding*



2. Block Coding

- Error Detection
  - 1. The receiver has a list of valid codewords.
  - 2. The original codeword has changed to an invalid one.

Figure 10.6 *Process of error detection in block coding*



  - An error-detection code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Example 10.2

- Let us assume that k = 2 and n = 3. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.
- Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:
- The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.
- The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

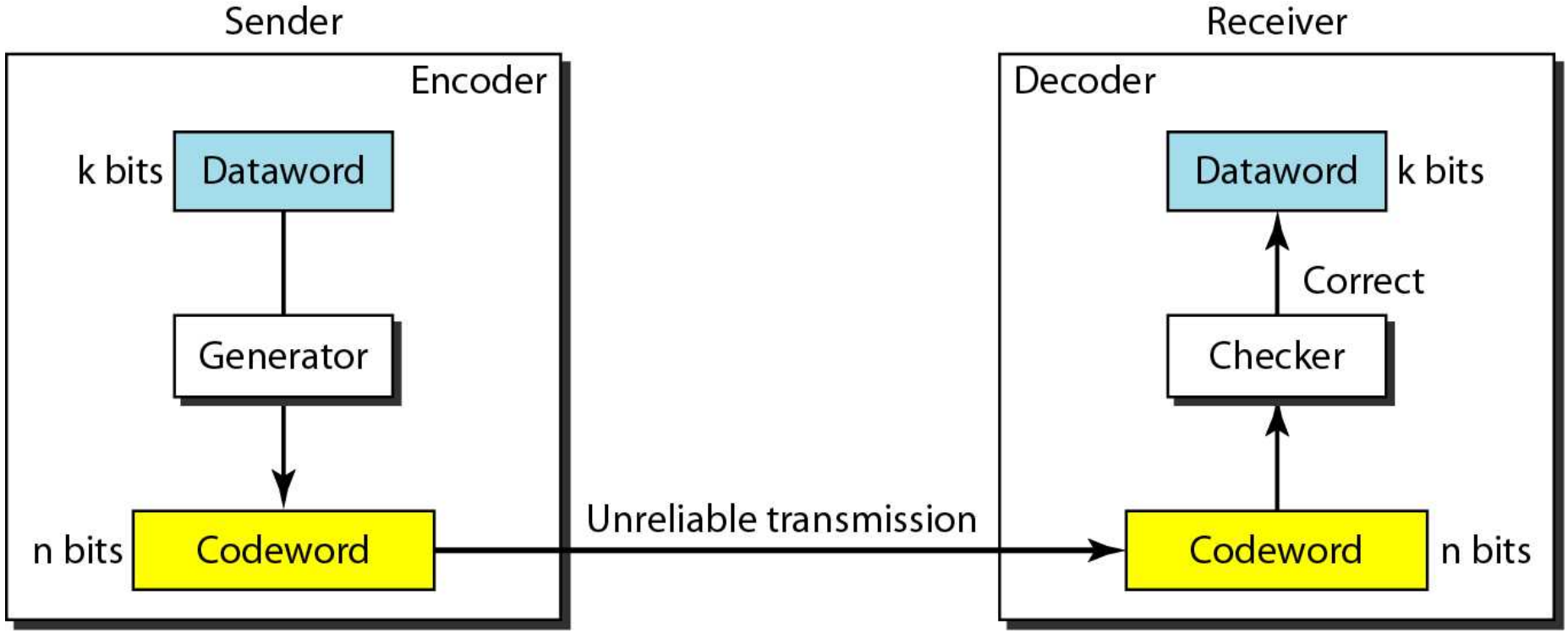
Table 10.1 *A code for error detection (Example 10.2)*

Datawords	*Codewords*
00	000
01	011
10	101
11	110

2. Block Coding

- Error Correction
  - Error correction is much more difficult than error detection.
  - In error correction the receiver needs to find the original codeword sent.
  - We can see that the idea is the same as error detection but the checker functions are much complex.

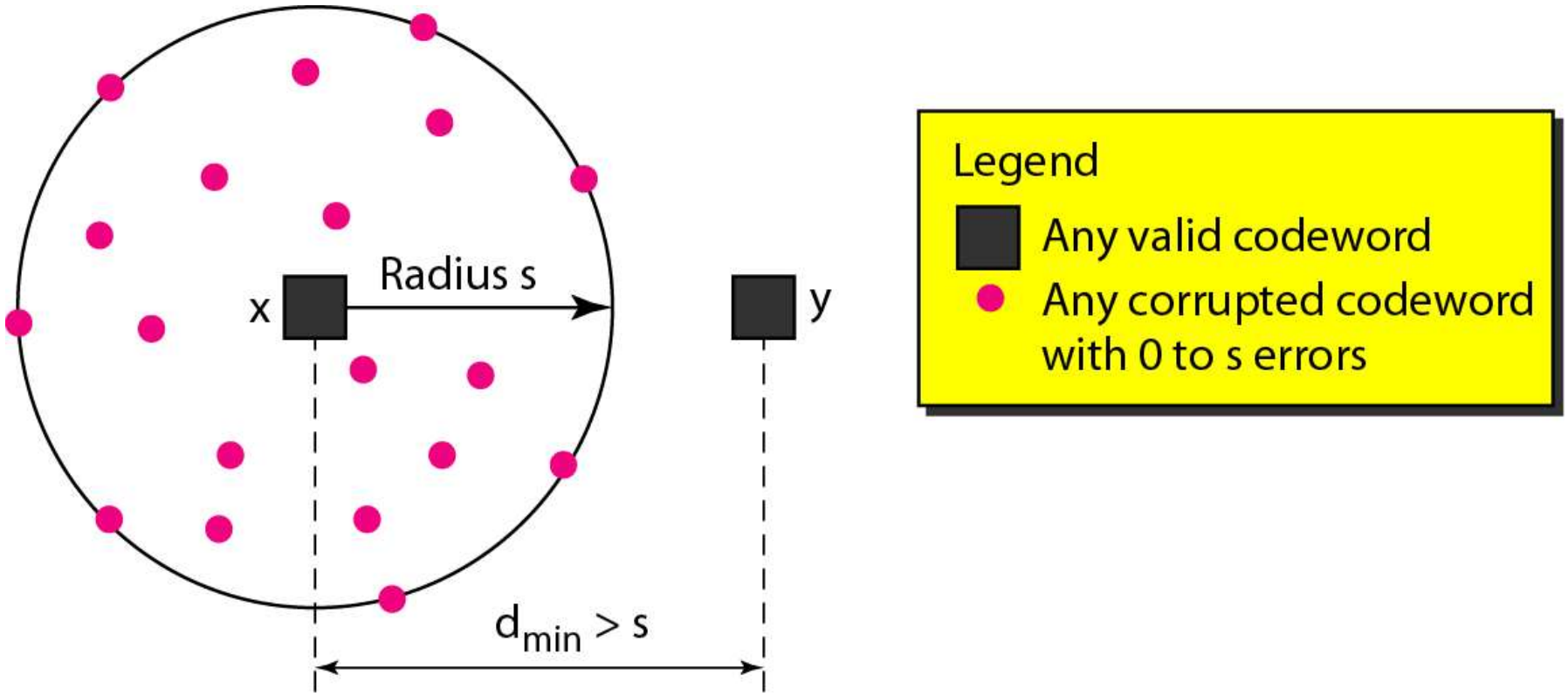
Figure 10.7 *Structure of encoder and decoder in error correction*



2. Block Coding

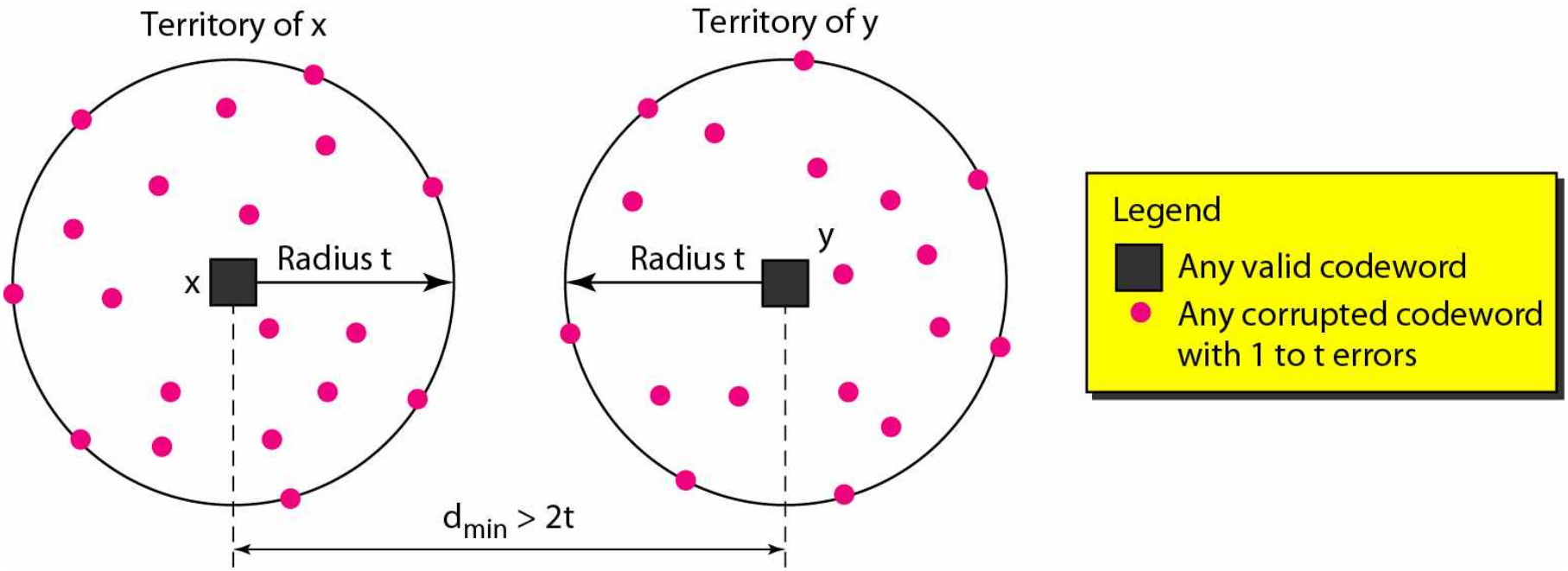
- Hamming Distance
  - The Hamming distance between two words is the number of differences between corresponding bits.
  - Example.
    - The Hamming distance d(000, 011) is 2 because 000 $\oplus$ 011 is 011 (two 1's)
- Minimum Hamming Distance
  - The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.
  - Three parameters : any coding scheme needs to have at least three parameters:
    - the codewords size n,
    - the dataword size k,
    - and the minimum Hamming distance d_{min}
  - Minimum Distance for Error Detection
    - To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s+1$
  - Minimum Distance for Error Correction
    - To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$

Figure 10.8 *Geometric concept for finding d_{min} in error detection*



To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Figure 10.9 Geometric concept for finding dmin in error correction



To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

3. Line Block Codes

Almost all block codes used today belong to a subset called linear block codes. A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

Topics discussed in this section:

Minimum Distance for Linear Block Codes
Some Linear Block Codes

3. Line Block Codes

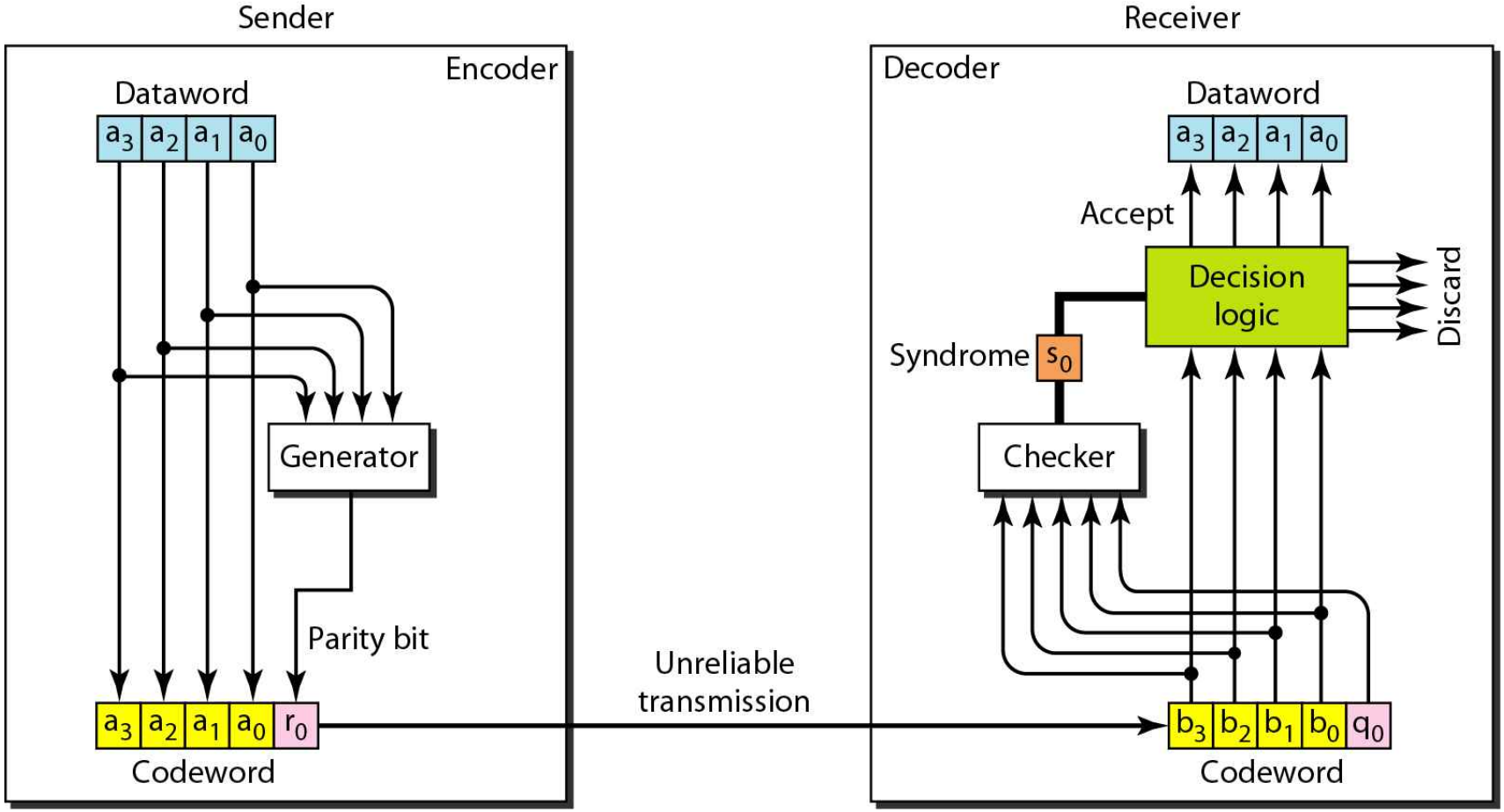
Simple parity-check code (n, k) where n = k+1

- A Simple parity-check code is a single-bit error-detecting code in which n = k+1 with d_{min} =2.
- A simple parity-check code can detect an odd number of erros.

Table 10.3 *Simple parity-check code C(5, 4)*

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure 10.10 *Encoder and decoder for simple parity-check code*



- $r_0 = a_3 + a_2 + a_1 + a_0$ (modulo−2)
- $s_0 = b_3 + b_2 + b_1 + q_0$ (modulo-2)

3. Line Block Codes

- Hamming Codes (n, k) where $n = 2^m - 1$, $k = n-m$
  - All hamming codes discussed in this book have $d_{min} = 3$.
  - The relationship between *m* and *n* in these codes is $n = 2^m - 1$, k=n-m
  - 1 bit error correction and several bits error detection
    - n: the codewords size
    - k: the datawords size
    - m: n-k, redundant size
  - We needs $\log_2(n+1)$ bits to indicate the value of (n+1). Hence,

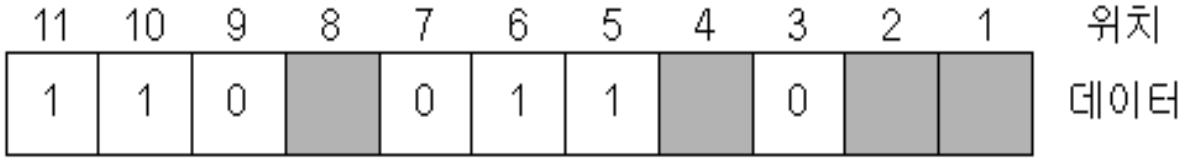
$$m \geq \log_2(n+1)$$

$$2^m \geq n+1 = m+k+1.$$

  - We achieves $n = 2^m - 1$ and $k = n-m$.

3. Line Block Codes

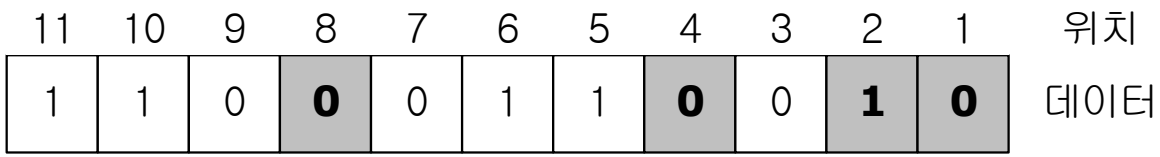
- Consider (11, 7) Hamming code.
  - The data to be sent is '1100110'.
  - STEP 1: the hamming bits are inserted at 2^3, 2^2, 2^1, and 2^0 respectively.



  - STEP 2: calculate the hamming bits

Decimal	Binary
11	1011
10	1010
6	0110
5	⊕ 0101
Exclusive-OR	0010

  - STEP 3: complete the codewords



3. Line Block Codes

– At the receiver site,
Case 1) if the following data is received,

11	10	9	8	7	6	5	4	3	2	1	위치
1	1	0	**0**	0	1	1	**0**	0	**1**	**0**	데이터

Do modulo-2 calculation as follows:

Decimal	Binary
11	1011
10	1010
6	0110
5	0101
2	⊕ 0010
Exclusive-OR	0000

Because the result is 0, there is no error.

3. Line Block Codes

– At the receiver site,
Case 1) if the following data is received,

11	10	9	8	7	6	5	4	3	2	1	위치
0	1	0	0	0	1	1	0	0	1	0	데이터

Do modulo-2 calculation as follows:

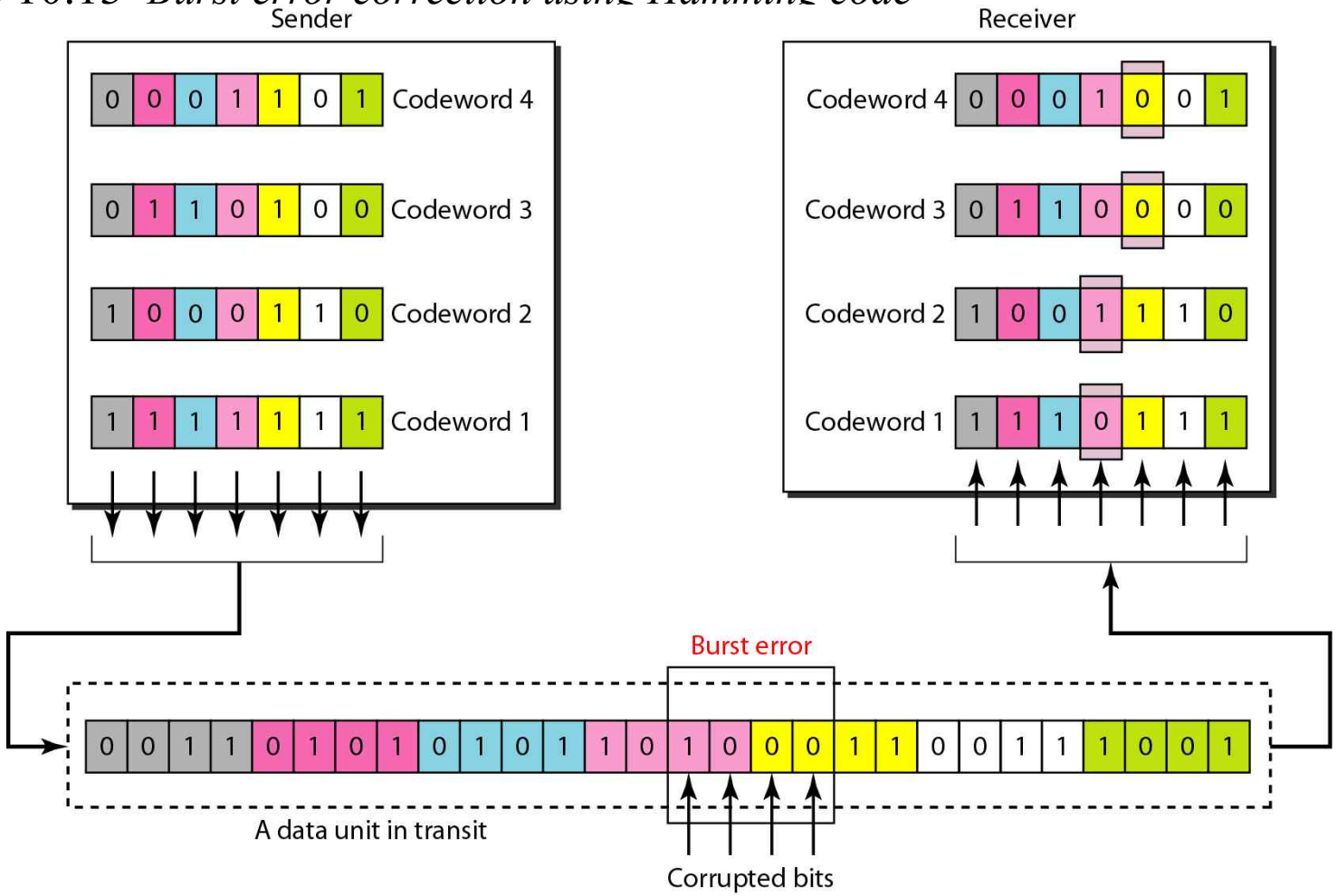
Decimal	Binary
10	1010
6	0110
5	0101
2	⊕ 0010
Exclusive-OR	1011

Because the result is 11, there is an error at the 11th bit.

3. Line Block Codes

– Burst error correction using Hamming code
  - A Hamming code can only correct a single error or detect a double error.
  - However, there is a way to make it detect a burst error as follows:

Figure 10.13 *Burst error correction using Hamming code*



4. Cyclic Codes

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

- *For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.*

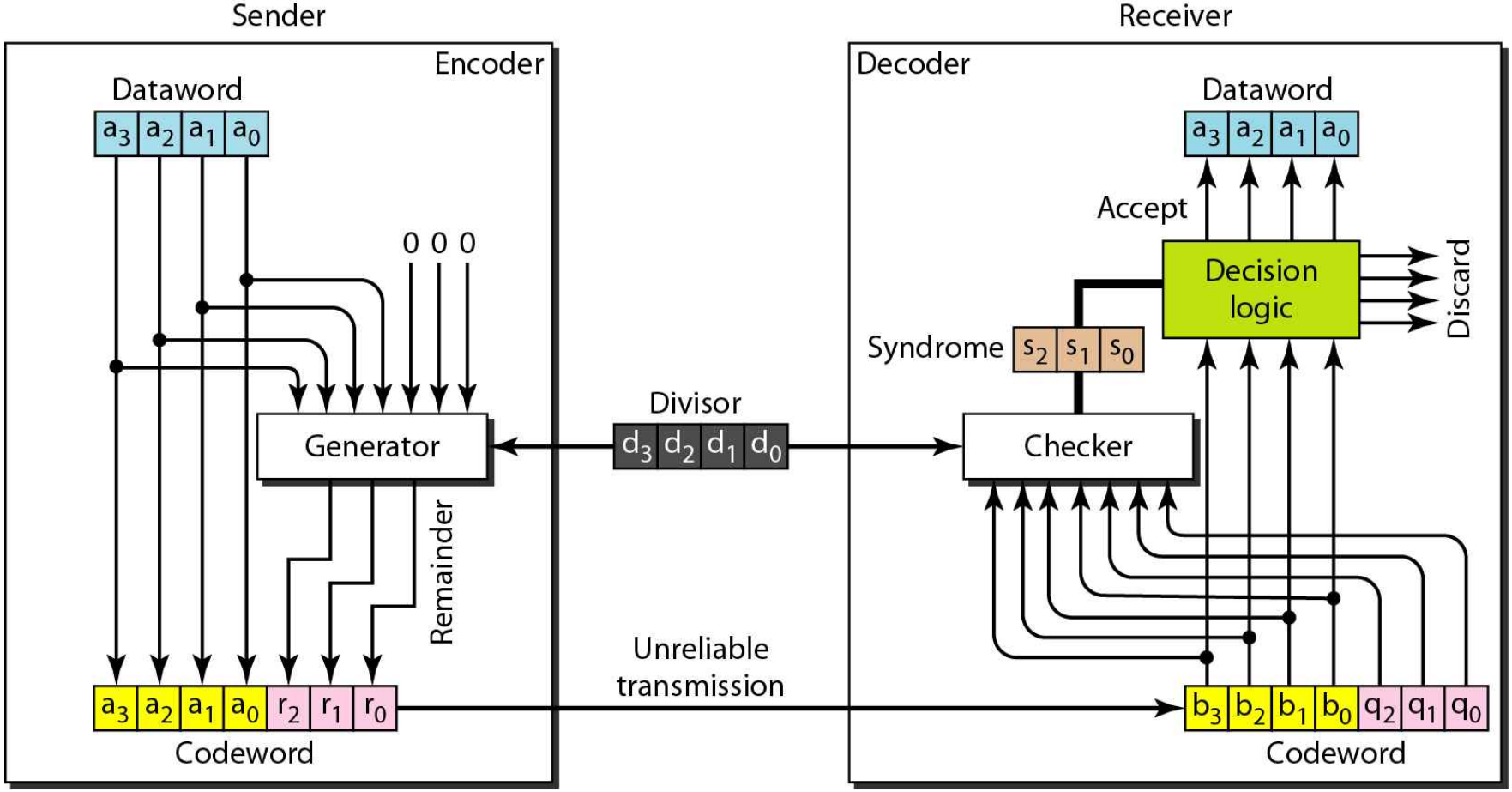
Topics discussed in this section:

Cyclic Redundancy Check
Hardware Implementation
Polynomials
Cyclic Code Analysis
Advantages of Cyclic Codes
Other Cyclic Codes

Cyclic Redundancy Check

- We simply discuss a category of cyclic codes called the cyclic redundancy (CRC) that is used in networks such as LANs and WANs.

Figure 10.14 *CRC encoder and decoder*



Cyclic Redundancy Check

Figure 10.15 *Division in CRC encoder*

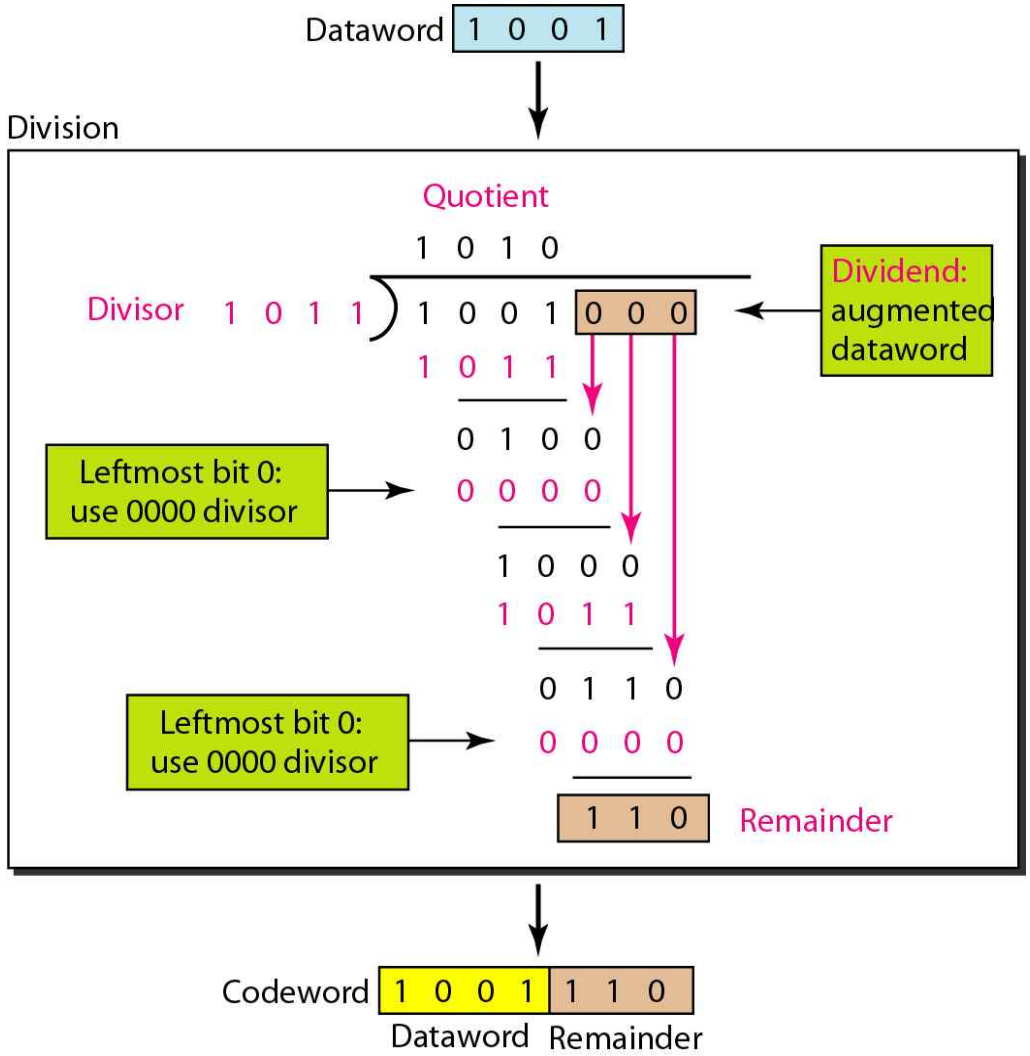
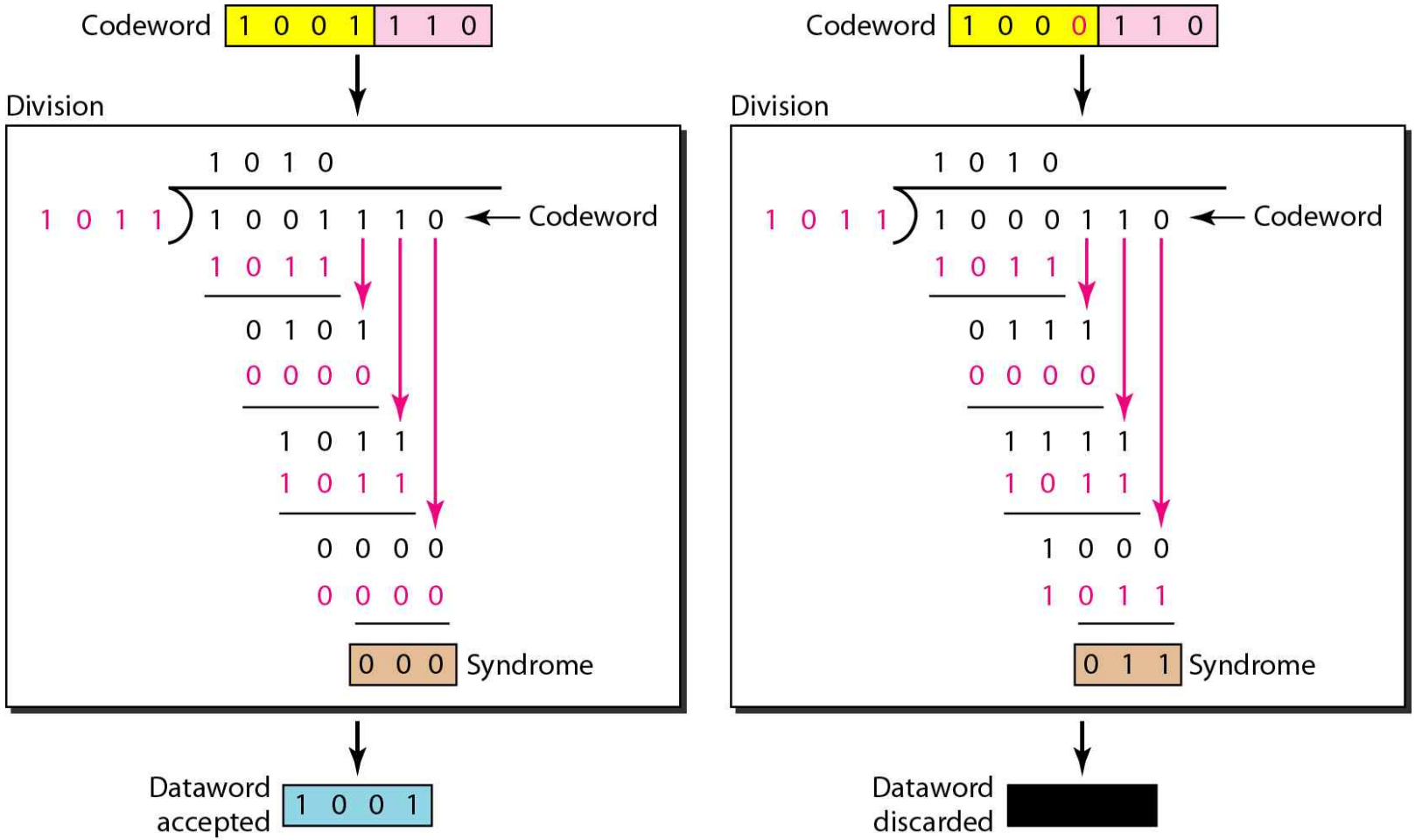


Figure 10.16 *Division in the CRC decoder for two cases*



- Hardware Implementation
  - One of the advantages of a cyclic code is that the encoder and decoder can easily and cheaply be implemented in hardware by using a handful of electronic devices.

Figure 10.17 *Hardwired design of the divisor in CRC*

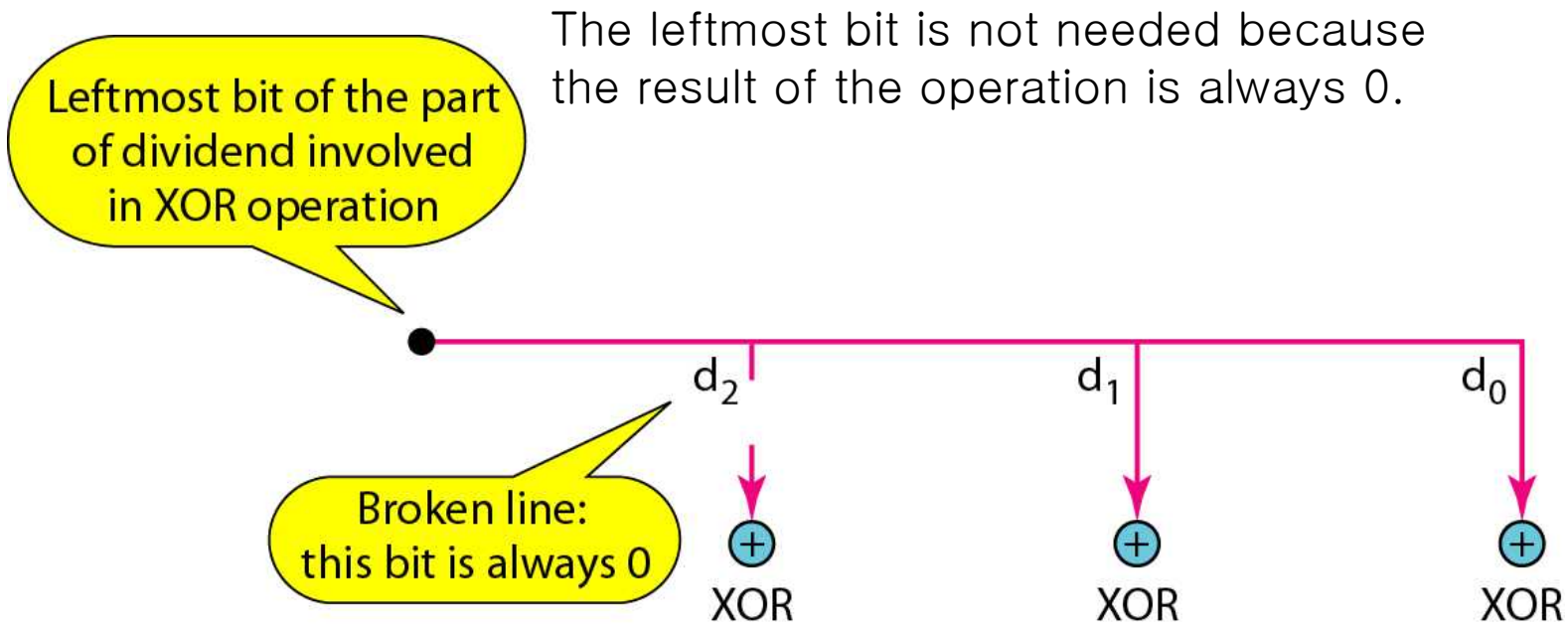
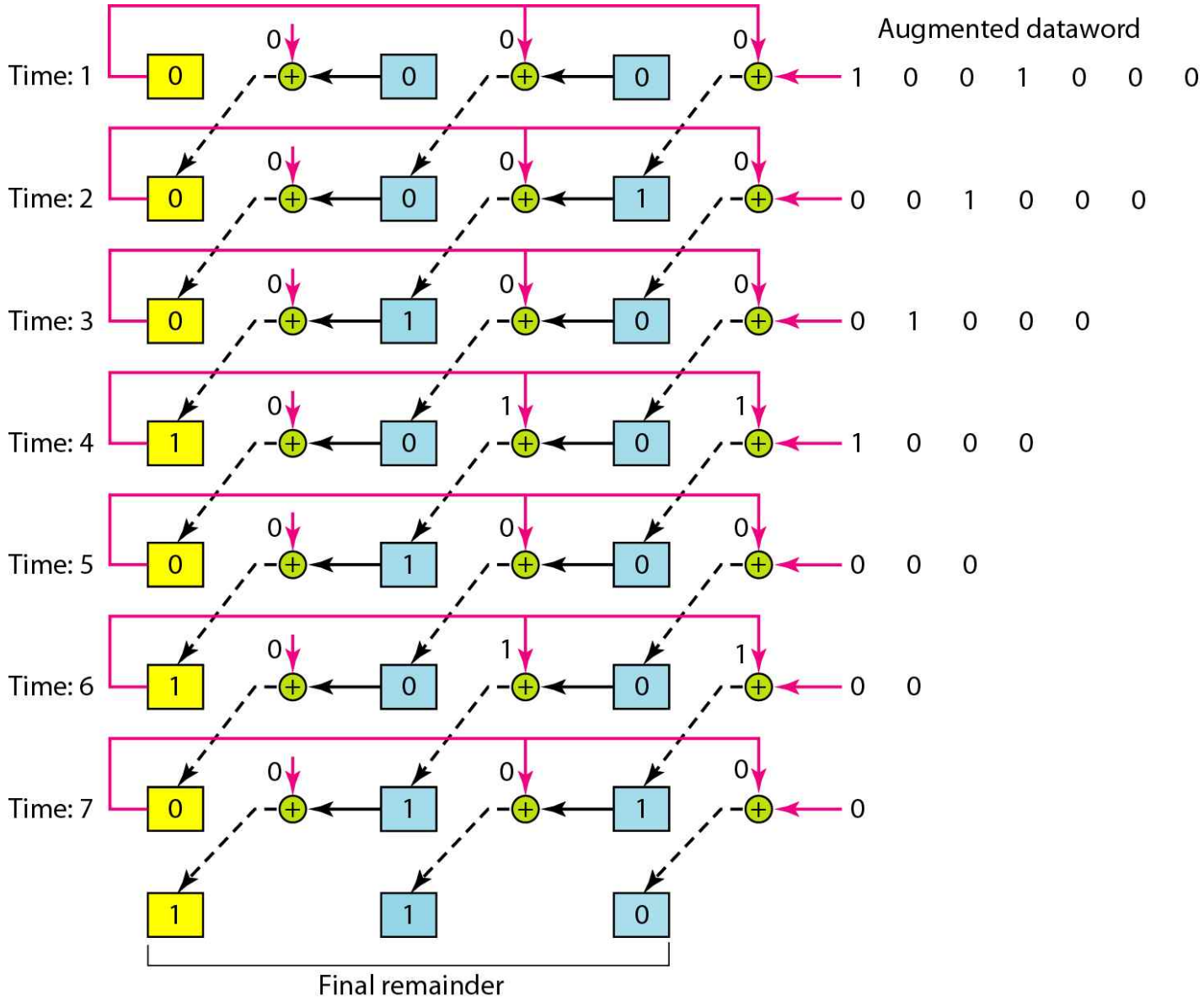
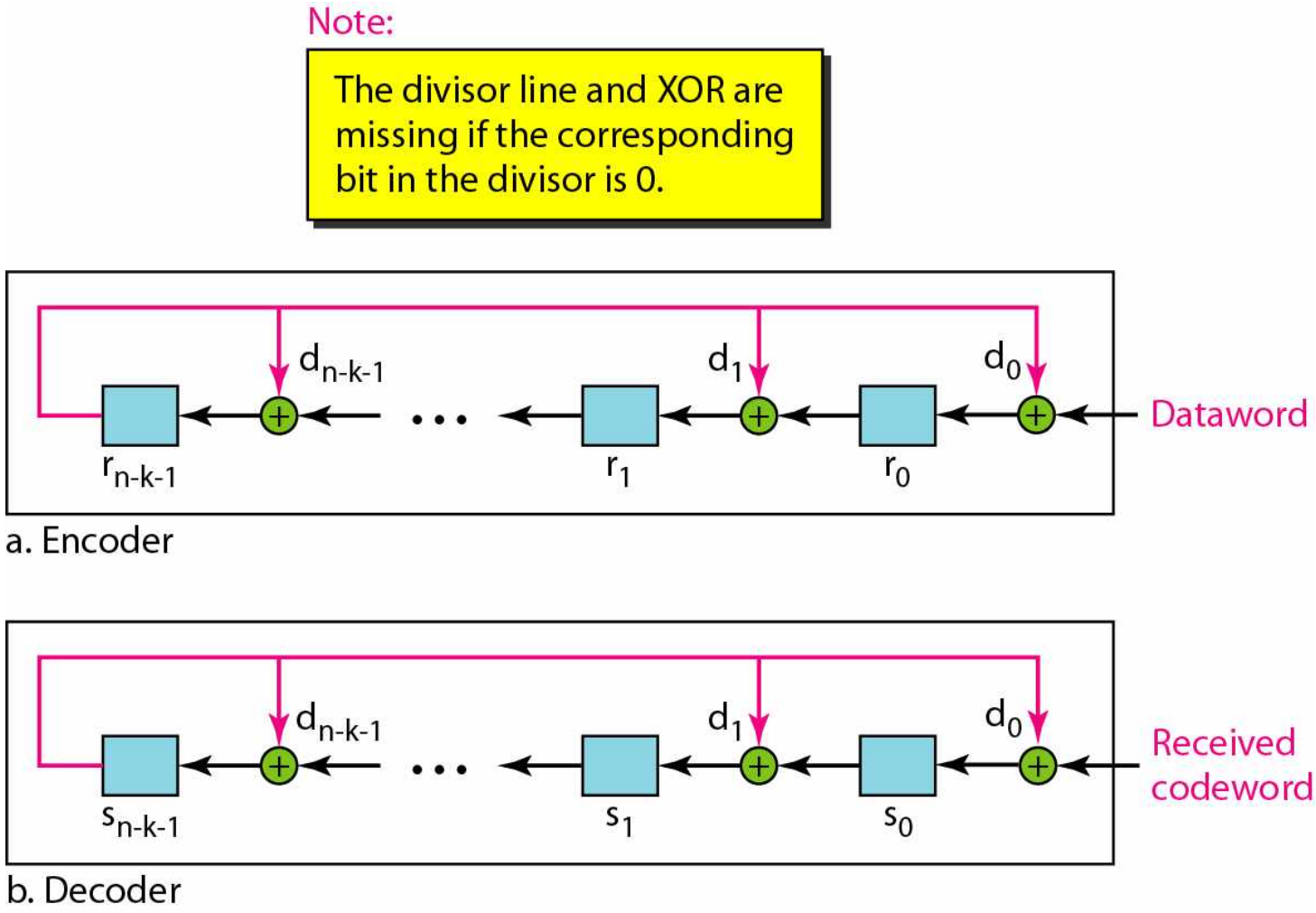


Figure 10.18 *Simulation of division in CRC encoder*



- General Design
  - A general design for the encoder and decoder is shown in under Figure

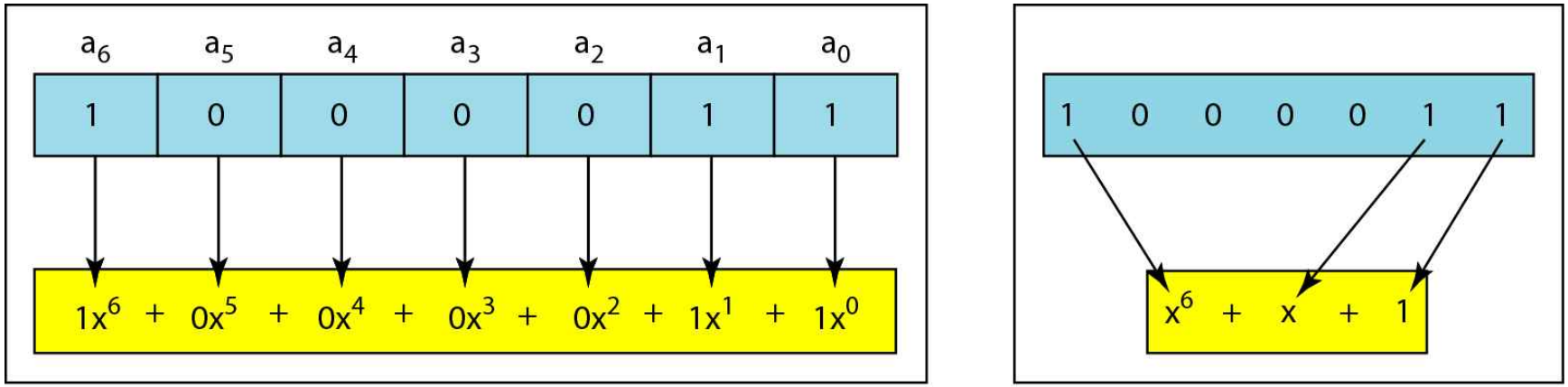
Figure 10.20 *General design of encoder and decoder of a CRC code*



Polynomials

- A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1. The position of the bit; the coefficient shows the value of the bit
- Degree of a Polynomial : the highest power in the polynomial
  - For example, the degree of the polynomial x^6+x+1 is 6.
- Adding, Multiplying, and Dividing
  - $(x^5+x^4+x^2) + (x^6+x^4+x^2) = x^6+x^5$
  - $x^3 \times x^4 = x^7$
  - $x^5 / x^2 = x^3$

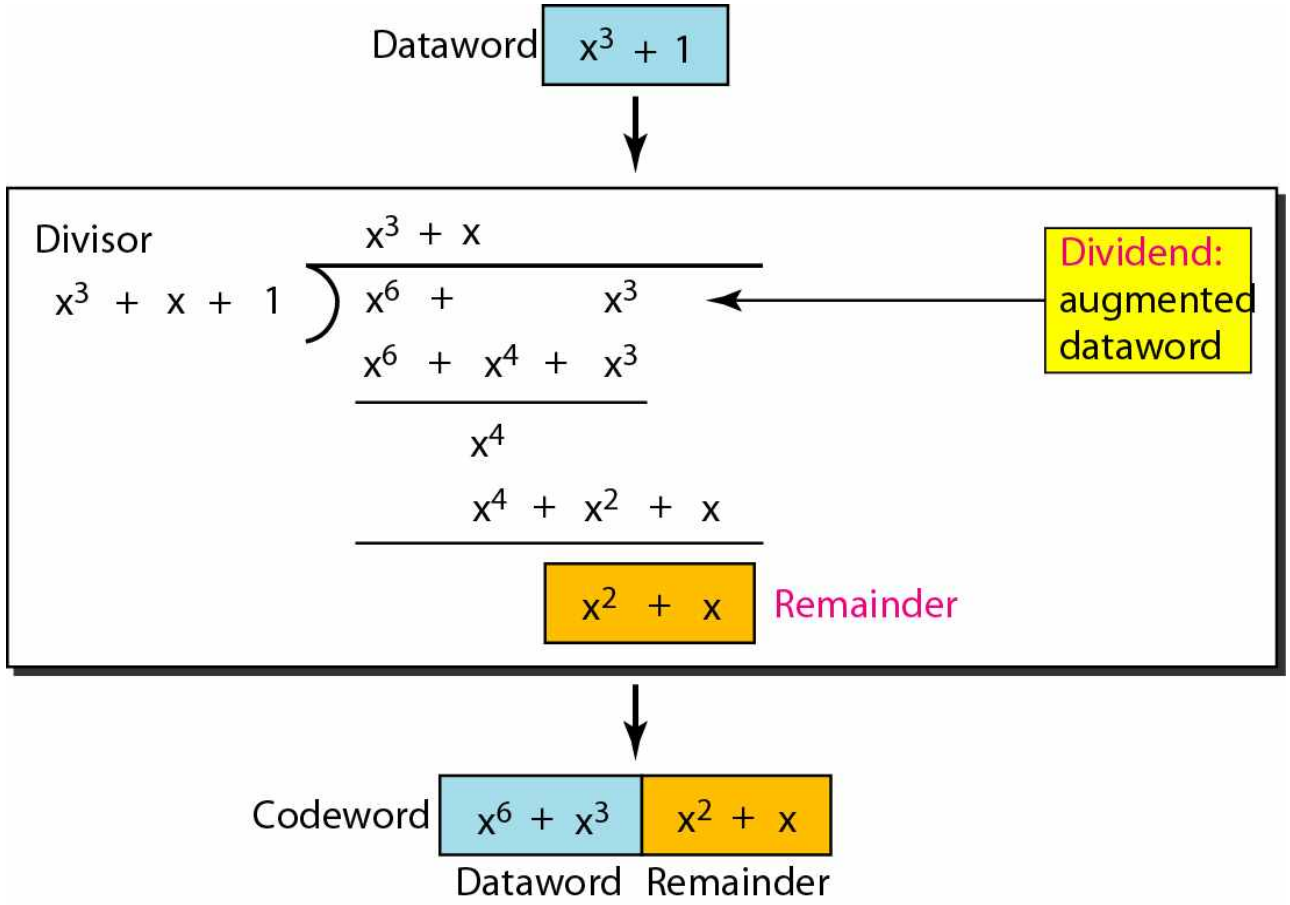
Figure 10.21 *A polynomial to represent a binary word*



a. Binary pattern and polynomial

b. Short form

Figure 10.22 *CRC division using polynomials*



The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

Cyclic Code Analysis

- We define the following, where f(x) is a polynomial with binary coefficients.

Dataword : d(x) Codeword : c(x) Generator : g(x)
Syndrome : s(x) Error : e(x)

In a cyclic code,
If $s(x) \neq 0$, one or more bits is corrupted.
If $s(x) = 0$, either
  a. No bit is corrupted. or
  b. Some bits are corrupted, but the decoder failed to detect them.

Digital Signal Service

- Received codeword = c(x) + e(x)
- The receiver divides the received codeword by g(x) to get the syndrome.

$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

  - The first term does not have a remainder.
  - If the second term does not have a remainder,
    - e(x) = 0 or
    - e(x) is divisible by g(x).

In a cyclic code, those *e*(x) errors that are divisible by *g*(x) are not caught.

4. Cyclic Codes

- Standard Polynomials

Table 10.7 *Standard polynomials*

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

- Other Cyclic Codes
  - One of the most interesting codes is the Reed-solomon code used today for both detection and correction.

5. Checksum

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking

Topics discussed in this section:

Idea
One's Complement
Internet Checksum

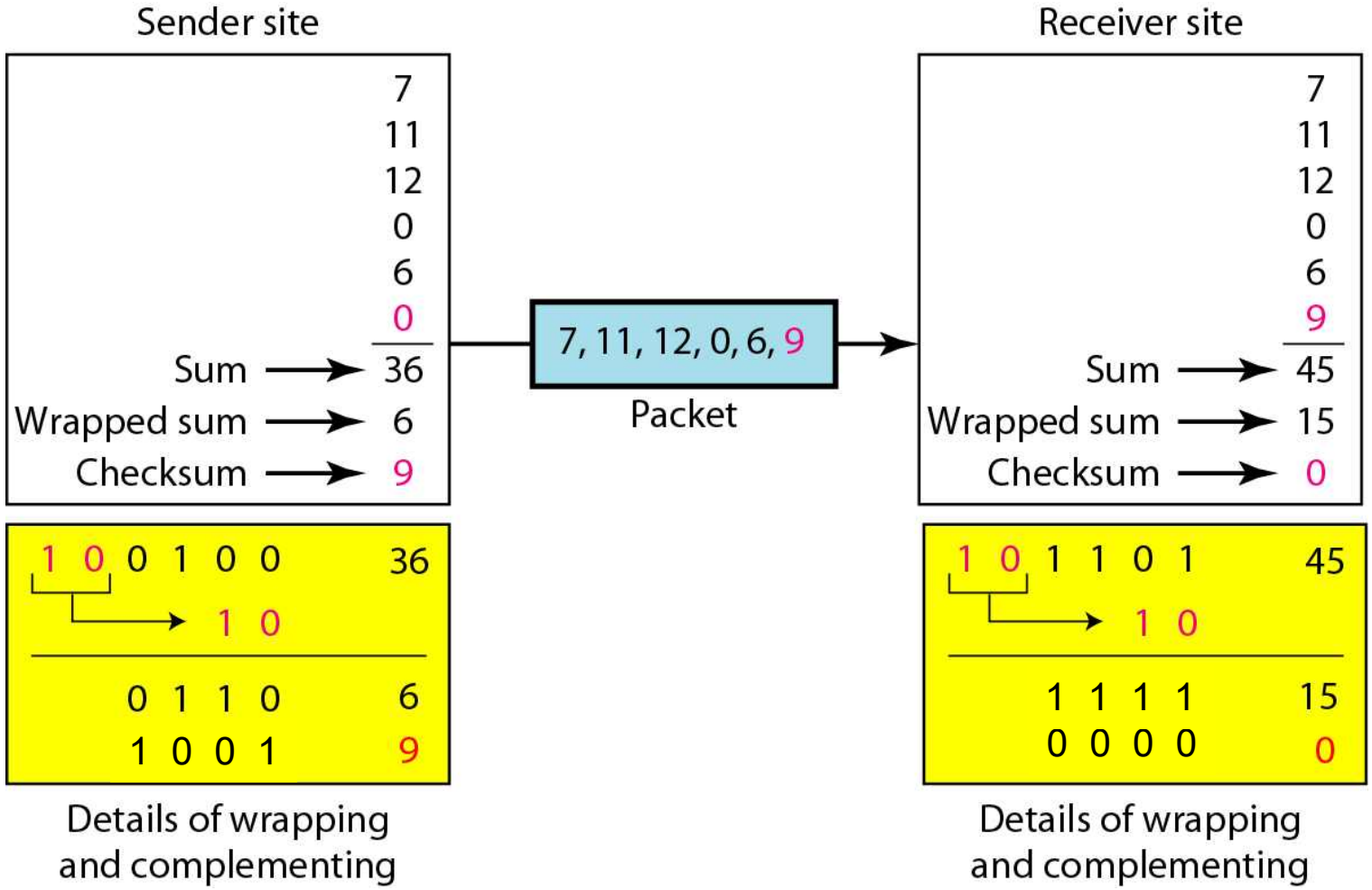
5. Checksum

- Idea
  - For example,
  - If the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers.
  - To make the job of the receiver easier, we send (7, 11, 12, 0, 6, -36), where -36 is the negative (complement) of the sum, called the checksum.

- One's Complement
  - One solution is to use one`s complement arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and 2^n -1 using only n bits.
  - If the number has more than n bits,
    - STEP 1: the extra leftmost bits is added to the n rightmost bits (wrapping).
    - STEP 2: one's complement
      - A negative number can be represented by inverting all bits.
      - This is the same as subtracting the number from 2^n – 1.

5. Checksum

- Example 10.22
  - The data information is given by (7, 11, 12, 0, 6).



5. Checksum

- Internet Checksum
  - The sender calculates the checking by following these steps.

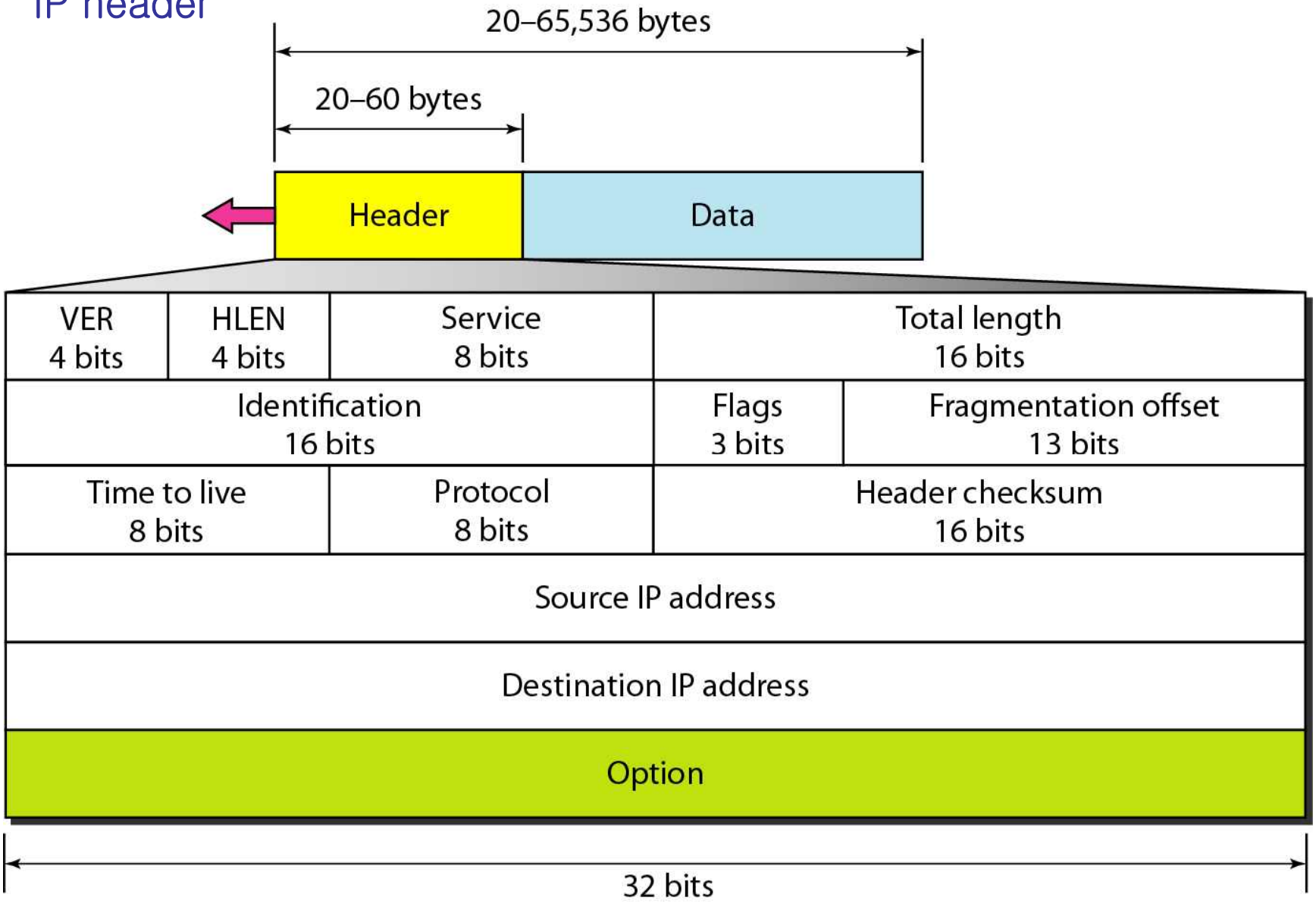
Sender site:
1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

  - The receiver uses the following steps for error detection.

Receiver site:
1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3. The sum is complemented and becomes the new checksum.
4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

5. Checksum

IP header



Summary

- Errors can be categorized as a single-bit error or a burst error. A single-bit error has one bit error per data unit. A burst error has two or more bit errors per data unit.
- Redundancy is the concept of sending extra bits for use in error detection.
- Three common redundancy methods are parity check, cyclic redundancy check (CRC), and checksum.
- Errors are corrected through retransmission and by forward error correction.
- The Hamming code is an error correction method using redundant bits. The number of bits is a function of the length of the data bits.