Data Communications and Networking Fourth Edition

Forouzan

Chapter 10

Error Detection and Correction

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Note

Data can be corrupted during transmission.

Some applications require that errors be detected and corrected.

10-1 INTRODUCTION

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

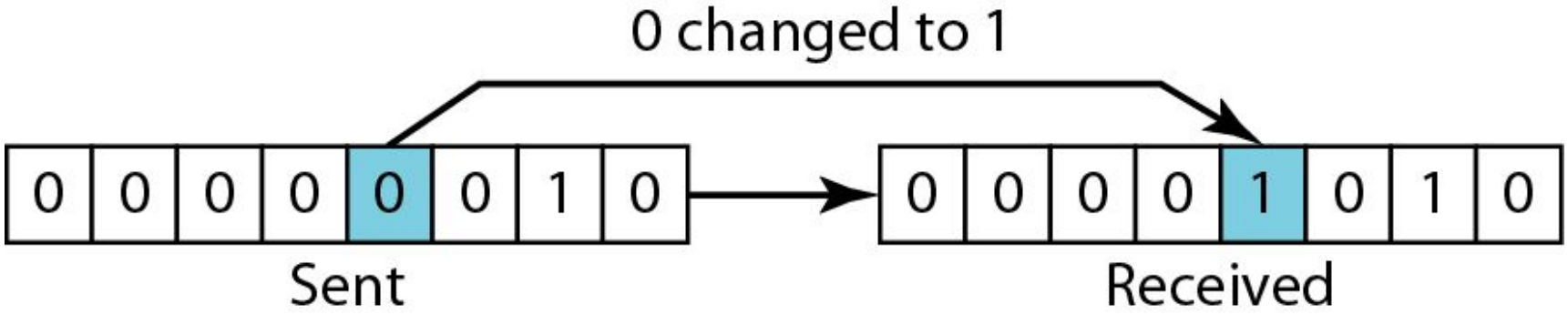
Topics discussed in this section:

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

Note

In a single-bit error, only 1 bit in the data unit has changed.

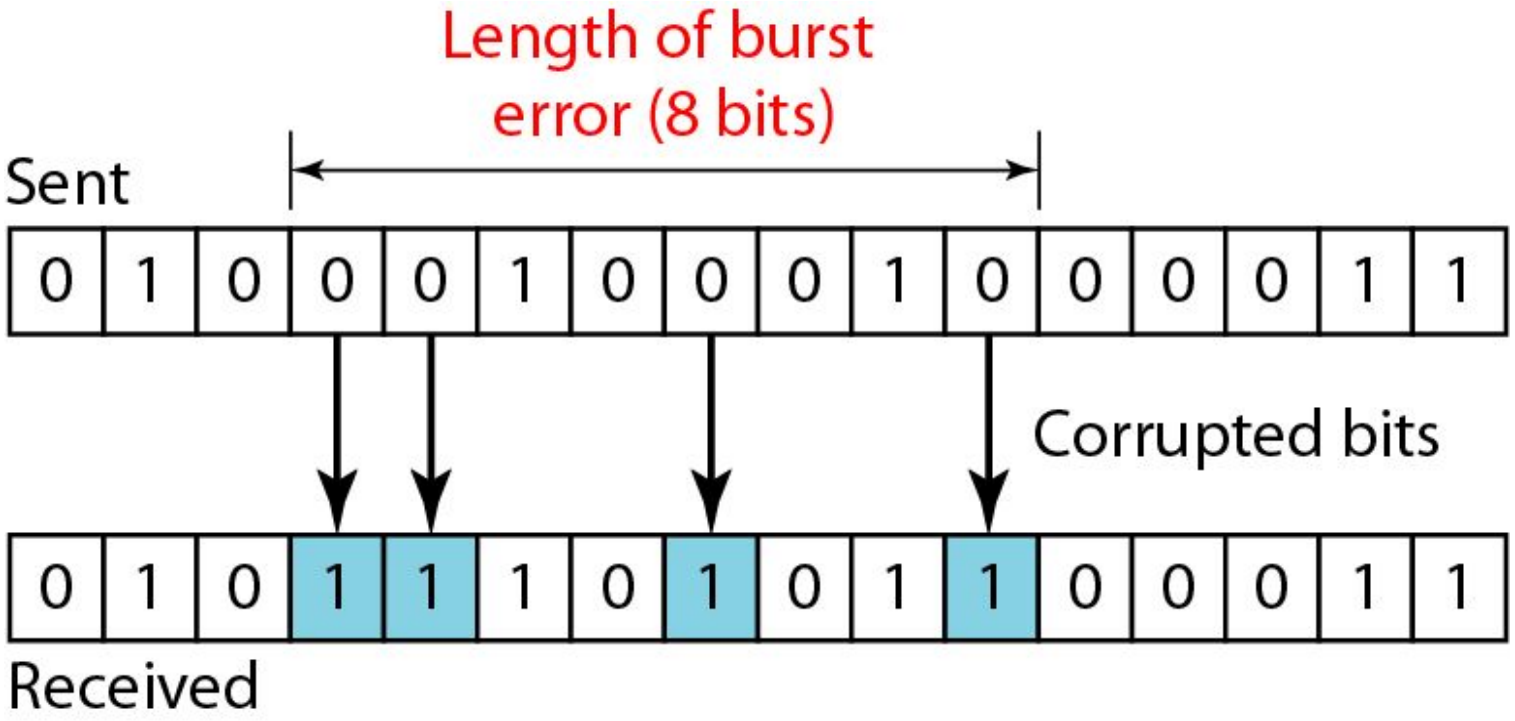
Figure 10.1 *Single-bit error*



Note

A burst error means that 2 or more bits in the data unit have changed.

Figure 10.2 *Burst error of length 8*



Redundancy

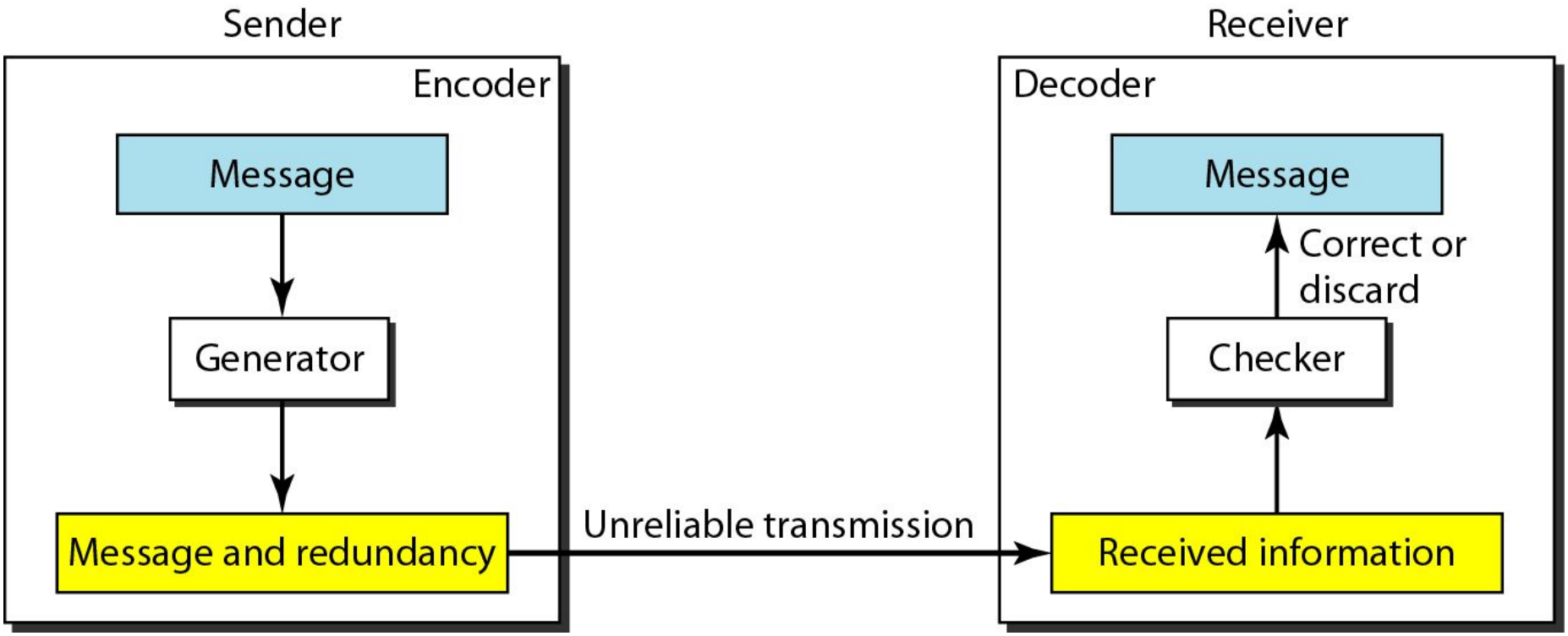
- The central concept in error detection and correction
- The redundant bits are added by the sender and removed by the receiver
- Their presence helps the receiver to detect and correct corrupted bits

Redundancy

Note

To detect or correct errors, we need to send extra (redundant) bits with data.

Figure 10.3 *The structure of encoder and decoder*



Coding

- Redundancy is achieved through various coding schemes
- The coding schemes are divided into two broad categories
- Block coding
- Convolution coding

Note

In this book, we concentrate on block codes; we leave convolution codes to advanced texts.

Note

In modulo-N arithmetic, we use only the integers in the range 0 to N −1, inclusive.

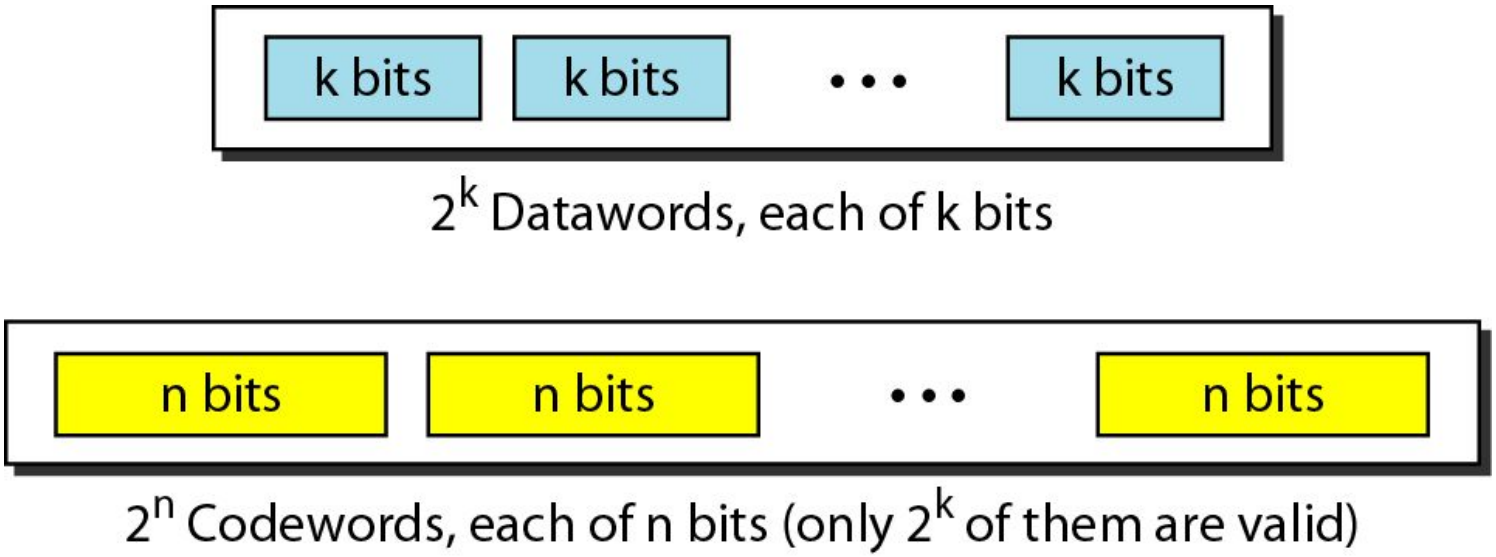
10-2 BLOCK CODING

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

Figure 10.5 *Datawords and codewords in block coding*



Example 10.1

The 4B/5B block coding discussed in Chapter 4 is a good example of this type of coding. In this coding scheme, $k = 4$ and $n = 5$. As we saw, we have $2^k = 16$ datawords and $2^n = 32$ codewords. We saw that 16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

Figure 10.6 *Process of error detection in block coding*

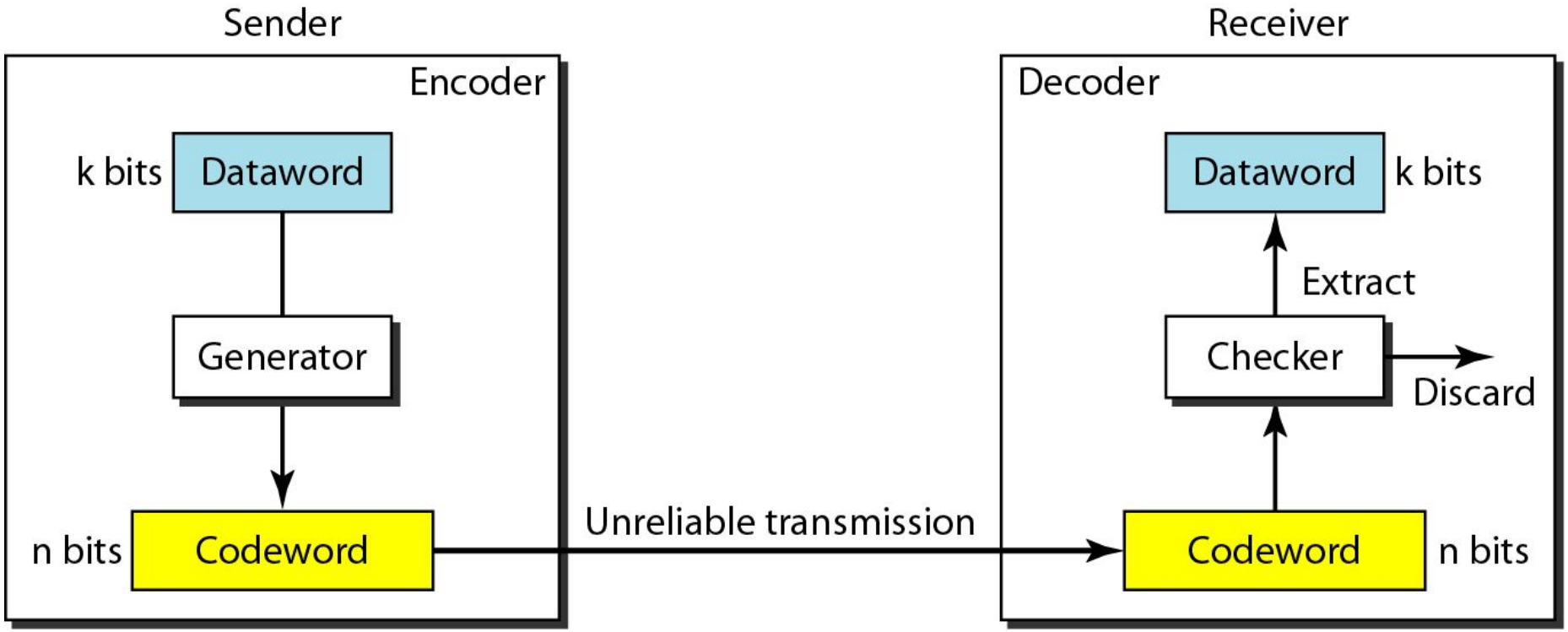


Table 10.1 *A code for error detection (Example 10.2)*

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

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:

1. *The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.*

Example 10.2 (continued)

2. *The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.*

3. *The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00.*

Thus two corrupted bits have made the error undetectable.

Note

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

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

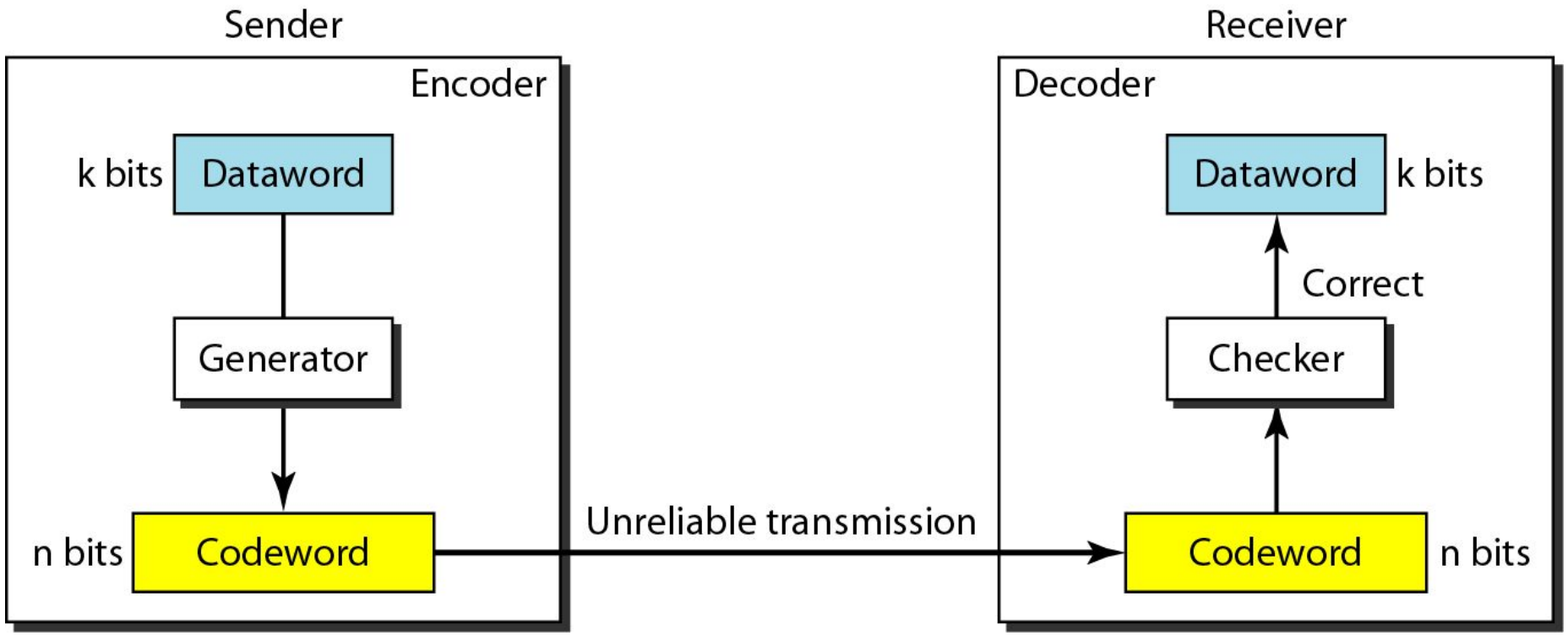
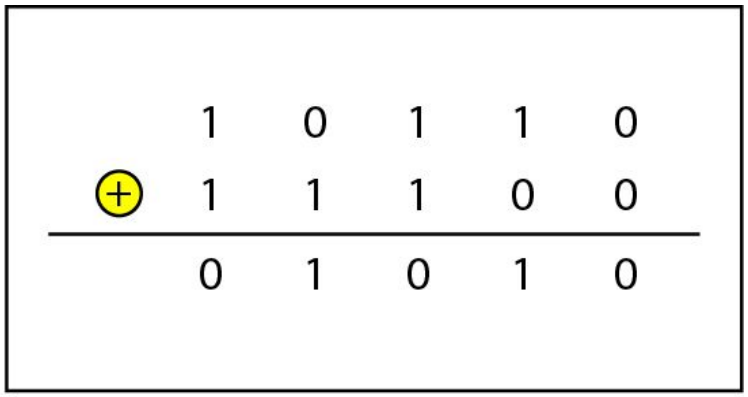


Figure 10.4 *XORing of two single bits or two words*

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

Note

The Hamming distance between two words is the number of differences between corresponding bits.

Example 10.4

Let us find the Hamming distance between two pairs of words.

1. *The Hamming distance d(000, 011) is 2 because*

$$000 \oplus 011 \text{ is } 011 \text{ (two 1s)}$$

2. *The Hamming distance d(10101, 11110) is 3 because*

$$10101 \oplus 11110 \text{ is } 01011 \text{ (three 1s)}$$

Note

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

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

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

- There are four different codewords which can have 6 possible pairs
- (000, 011)
- (000, 101)
- (000, 110)
- (011, 101)
- (011, 110)
- (101.110)

Table 10.2 *A code for error correction (Example 10.3)*

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Example 10.6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

Solution

We first find all the Hamming distances.

$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

The d_{min} in this case is 3.

Note

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

Note

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

Example 10.9

A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?

Solution

This code guarantees the detection of up to three errors ($s = 3$), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. Error correction codes need to have an odd minimum distance (3, 5, 7, . . .).

10-4 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.*

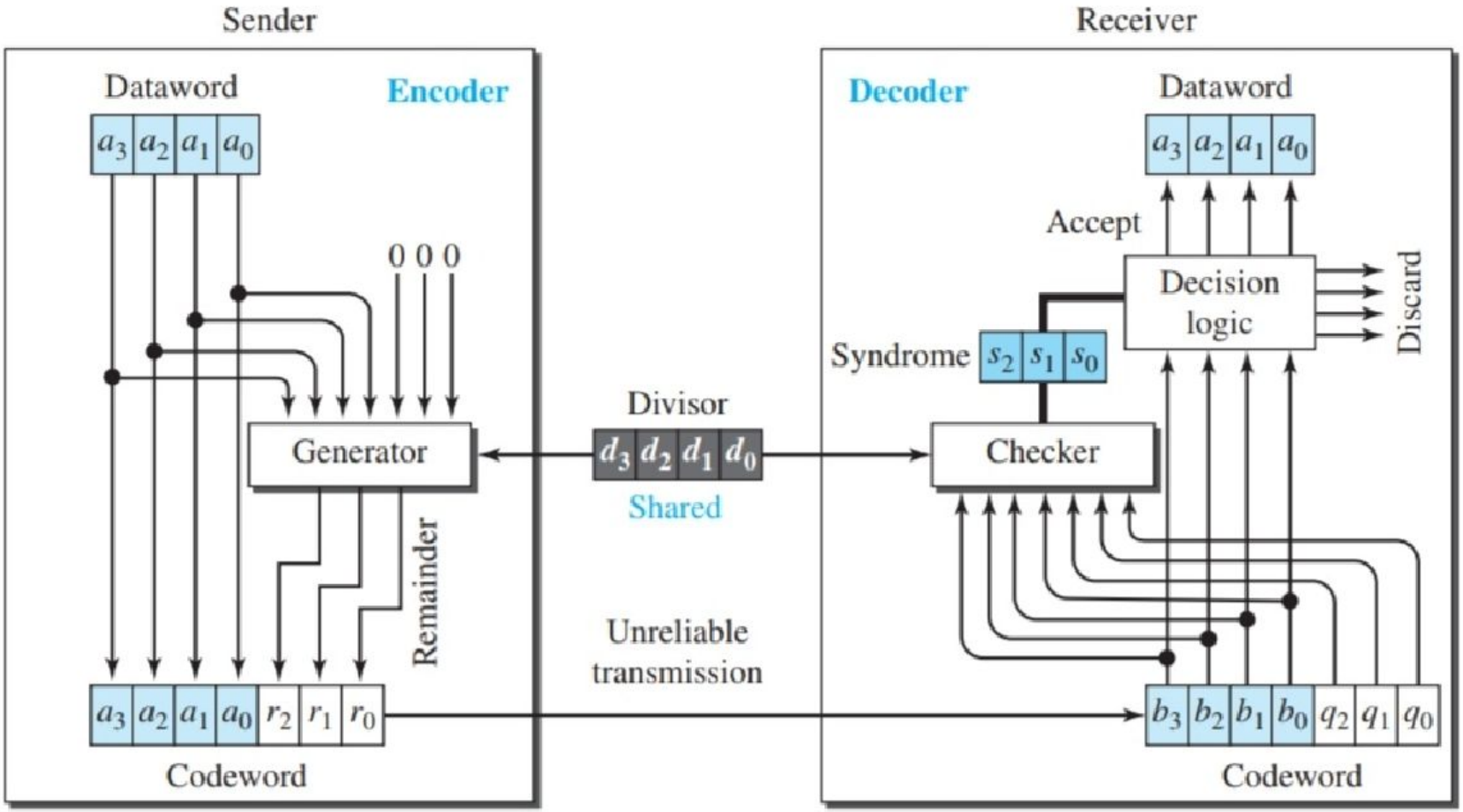
Topics discussed in this section:

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

Table 10.6 *A CRC code with C(7, 4)*

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

Figure 10.5 *CRC encoder and decoder*



In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding $n - k$ (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size $n - k + 1$ (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder ($r_2 r_1 r_0$) is appended to the dataword to create the codeword.

The decoder receives the codeword (possibly corrupted in transition). A copy of all n bits is fed to the checker, which is a replica of the generator. The remainder produced by the checker is a syndrome of $n - k$ (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 left-most bits of the codeword are accepted as the dataword (interpreted as no error); otherwise. the 4 bits are discarded (error).

Figure 10.15 *Division in CRC encoder*

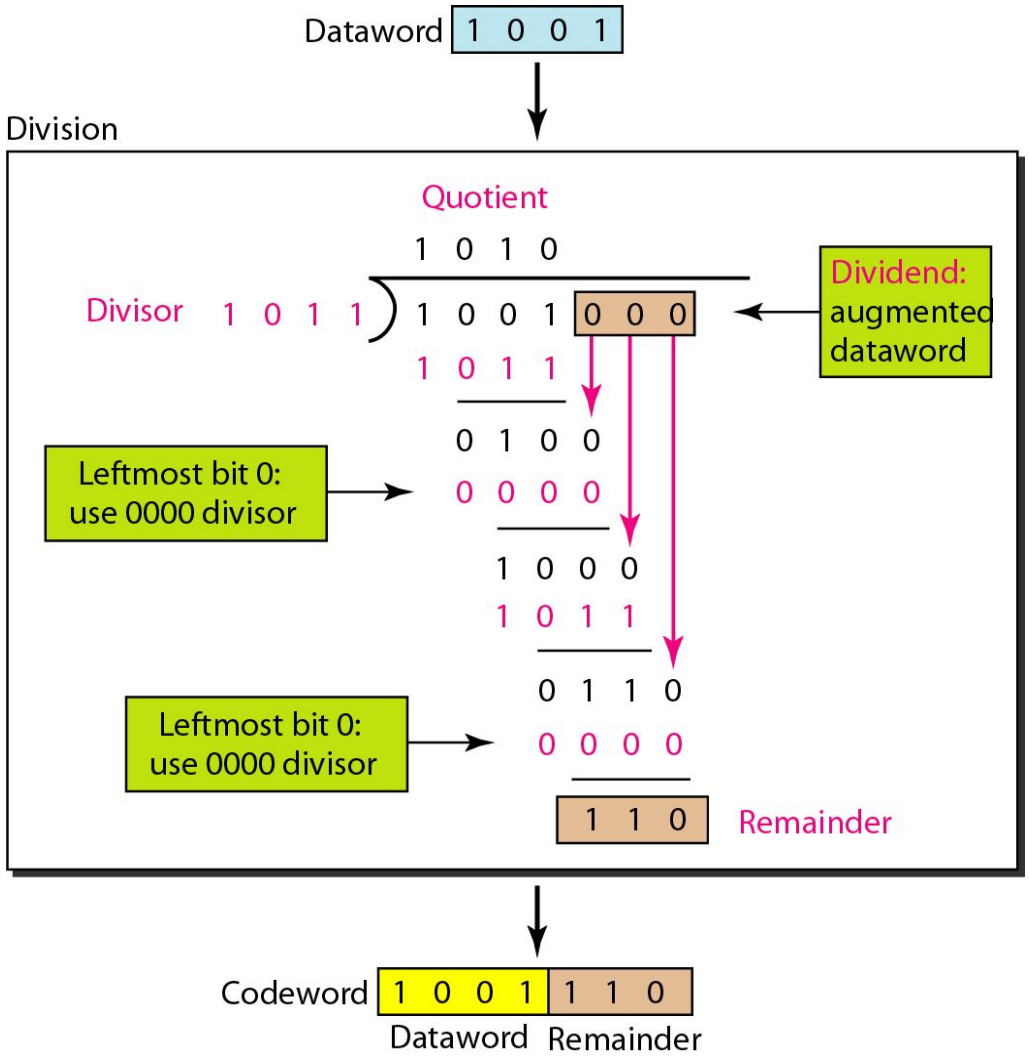


Figure 10.16 *Division in the CRC decoder for two cases*

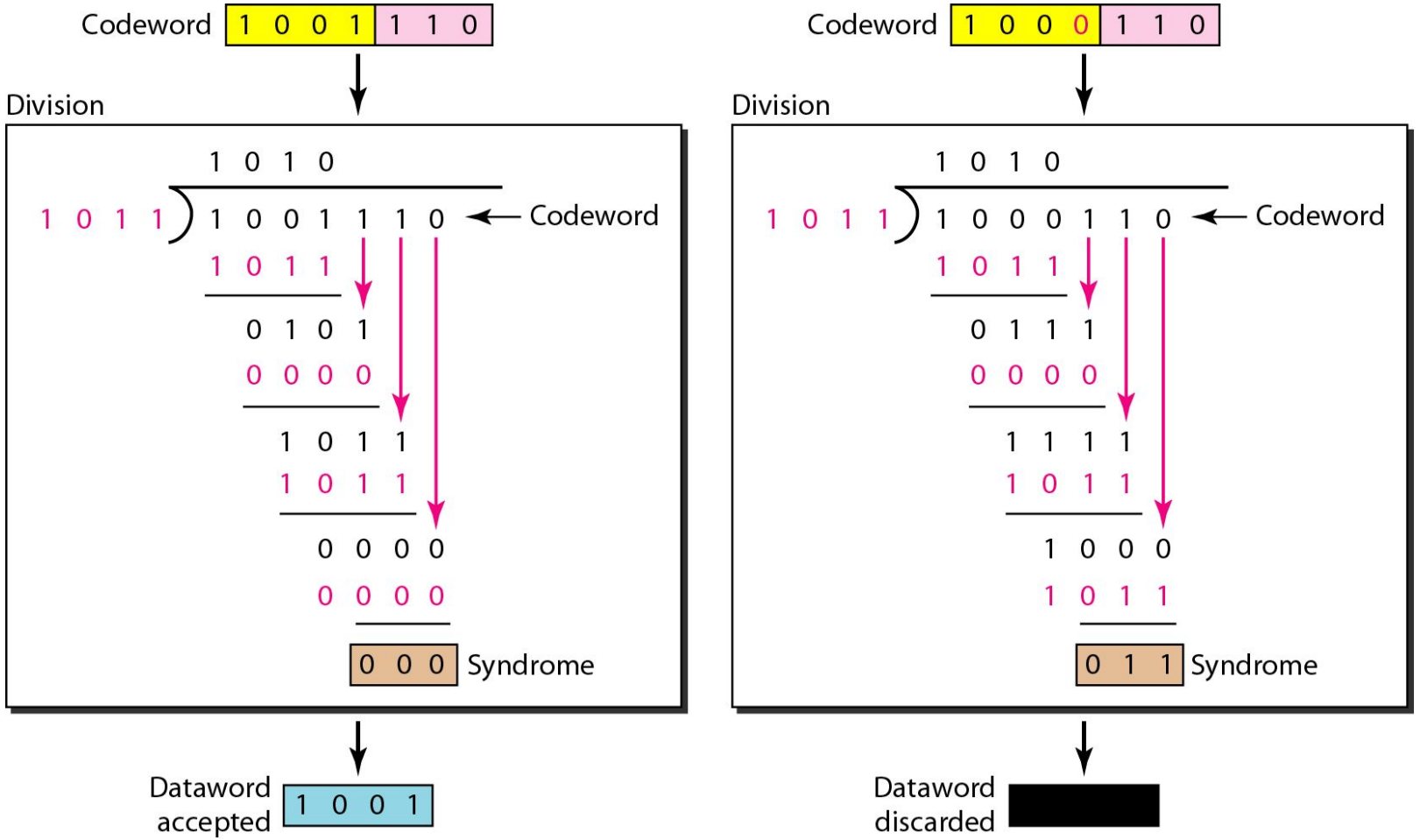
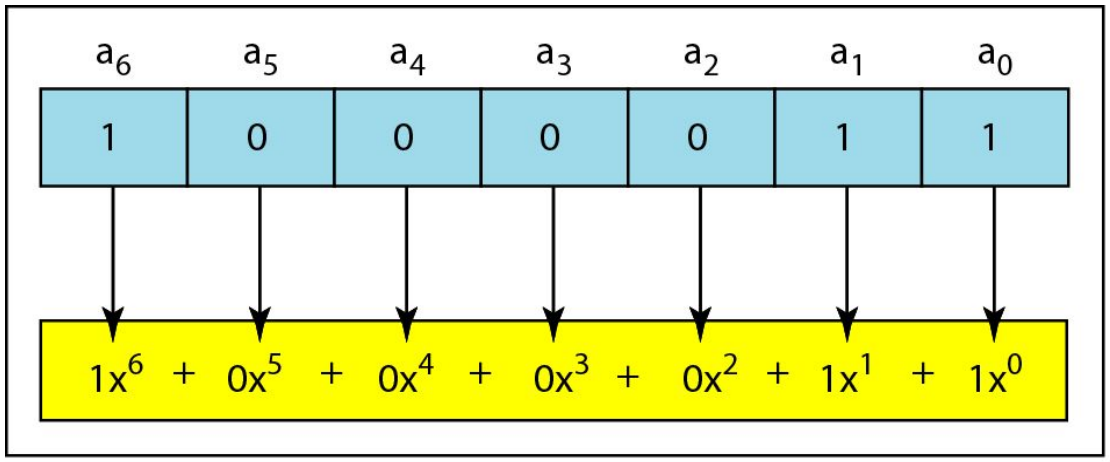
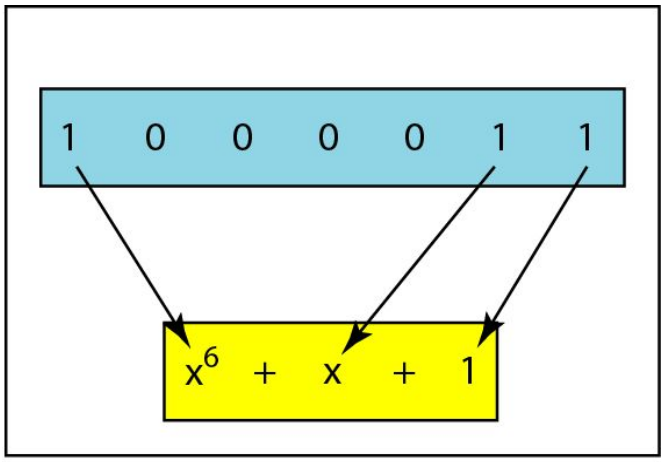


Figure 10.21 *A polynomial to represent a binary word*

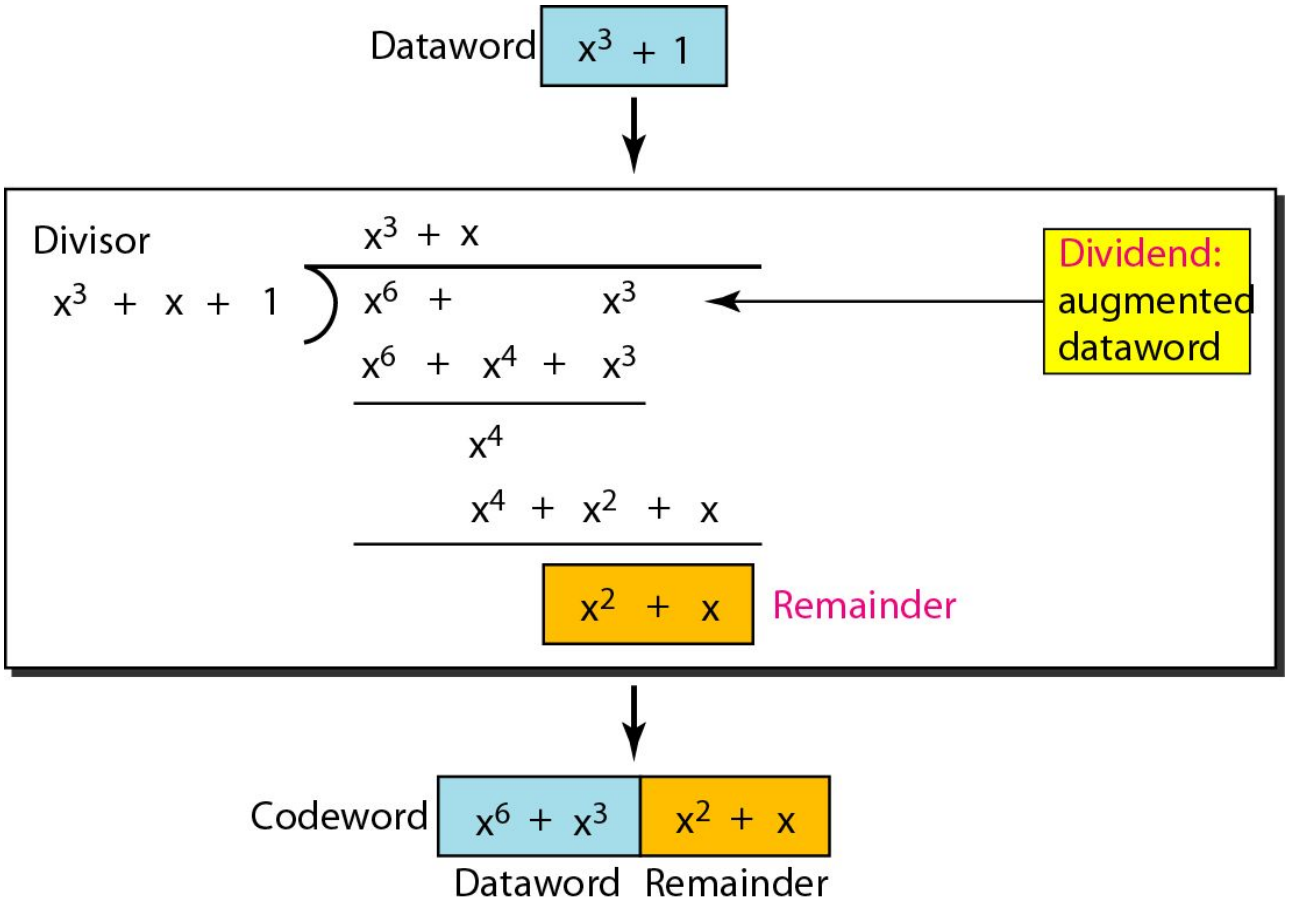


a. Binary pattern and polynomial



b. Short form

Figure 10.22 *CRC division using polynomials*



Note

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

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

Note

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.

Note

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

Note

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

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