



第三部分

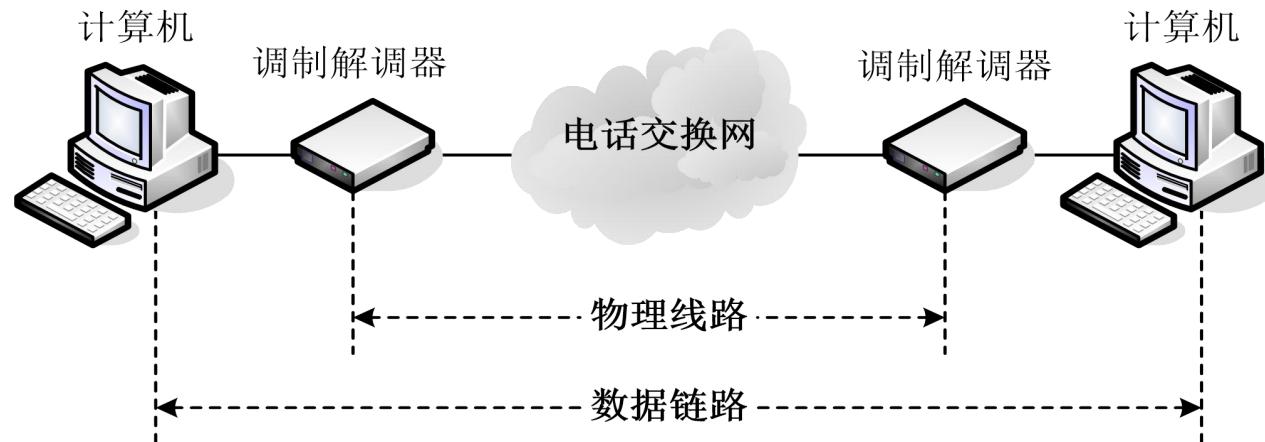
Chapter 10

Error Detection and Correction

检错与纠错

物理线路和数据链路

- ◆ 物理线路通常是指实际的一段物理电路，用于连接两个传输设备或一台计算机与一个传输设备。
- ◆ 在两台计算机之间传输数据，除了需要一条物理线路及相应的传输设备之外，还需要协议来控制数据在线路上的传输，以保证数据传输过程的正确性。实现这些协议的硬件、软件与物理线路共同构成了数据链路。



数据链路层的必要性

- ◆ 通信线路由传输介质与设备组成。物理线路是指没有采用差错控制的传输介质与设备，物理线路传输数据信号是存在差错的，普通电话线路不采取差错控制措施将无法达到计算机网络的要求。
- ◆ 设计数据链路层的目的是在原始的、有差错的物理线路的基础上，采取差错检测、差错控制与流量控制等方法，将有差错的物理线路改进成无差错的数据链路，以便向网络层提供高质量的服务。
- ◆ 从网络参考模型的角度来看，物理层以上各层都有改善数据传输质量的责任，数据链路层是其中最重要的一层。

数据链路层的主要功能

- ◆ **链路管理**

数据链路的建立、维持和释放称为链路管理。

- ◆ **帧同步**

物理层的比特序列按数据链路层协议的规定被封装成帧来传输，帧同步是指接收方能从接收的比特序列中正确判断出一帧的开始与结束位。

- ◆ **流量控制**

通信双方需要协调数据发送与接收速率是否匹配。如果数据链路拥塞或接收方不能及时接收，则发送方需要控制自己的发送速率。

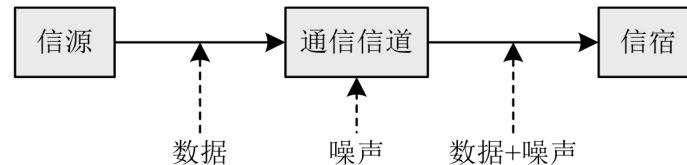
- ◆ **差错控制**

数据链路层的主要功能是将有差错的物理线路变成无差错的数据链路。

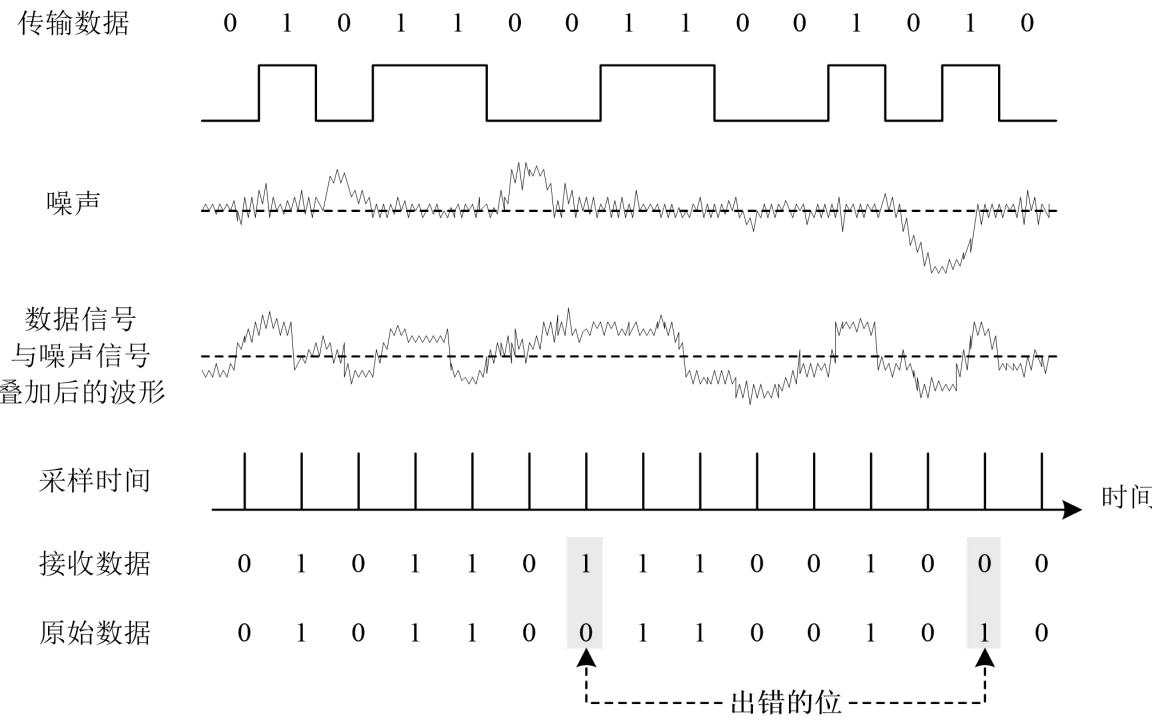
- ◆ **透明传输**

当传输数据中出现控制字符时，就必须采取适当的处理措施，防止接收方将数据内容误认为是控制信息。任何一帧封装的数据无法保证不出现该比特序列，这就是帧传输“透明性”问题。

传输差错的产生过程



(a)



(b)

误码率

- ◆ 误码率是指二进制比特在数据传输中出错的概率，是衡量数据传输系统在正常工作状态下传输可靠性的参数。误码率在数值上近似等于 $P_e = N_e/N$ ，其中 N 为传输的二进制比特总数， N_e 为被传错的比特数。
- ◆ 对于一个实际的数据传输系统，不能笼统地说误码率越低就越好，要根据实际传输要求提出误码率要求。传输速率一定时，要求的误码率越低，设备的复杂性越高，造价也越高。
- ◆ 差错出现具有随机性，在测量一个数据传输系统时，只有被测的二进制位数越多，才越接近真正的误码率值。

10-1 INTRODUCTION

差错检测和差错纠正

Topics discussed in this section:

Types of Errors

差错类型

Redundancy

冗余

Detection Versus Correction 检错和纠错

Forward Error Correction Versus Retransmission 前向纠错和重传

Coding 编码

Modular Arithmetic 模运算

差错类型 —— 单比特差错

数据单元中仅有一比特发生变化。

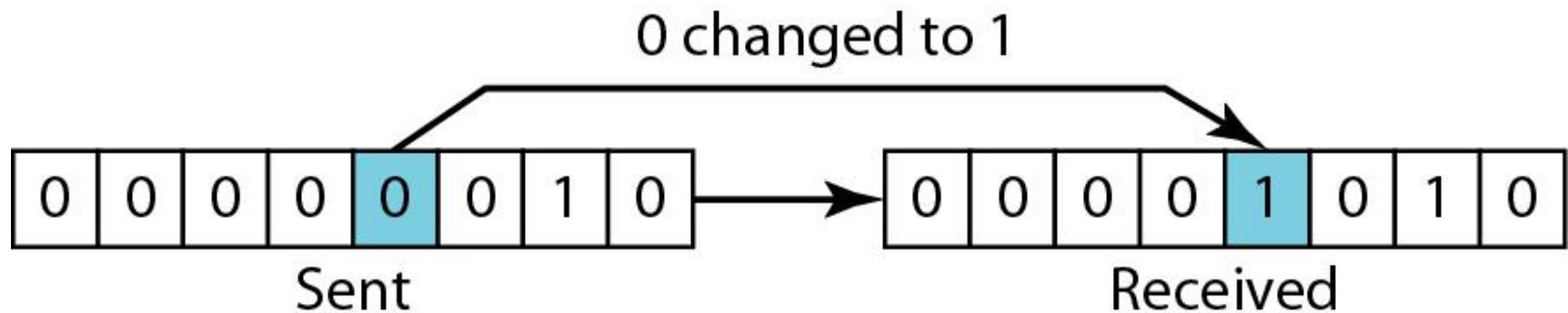


Figure 10.1 单个位差错 (*Single-bit error*)

差错类型 —— 突发性差错

数据单元中有两位或多位发生变化。

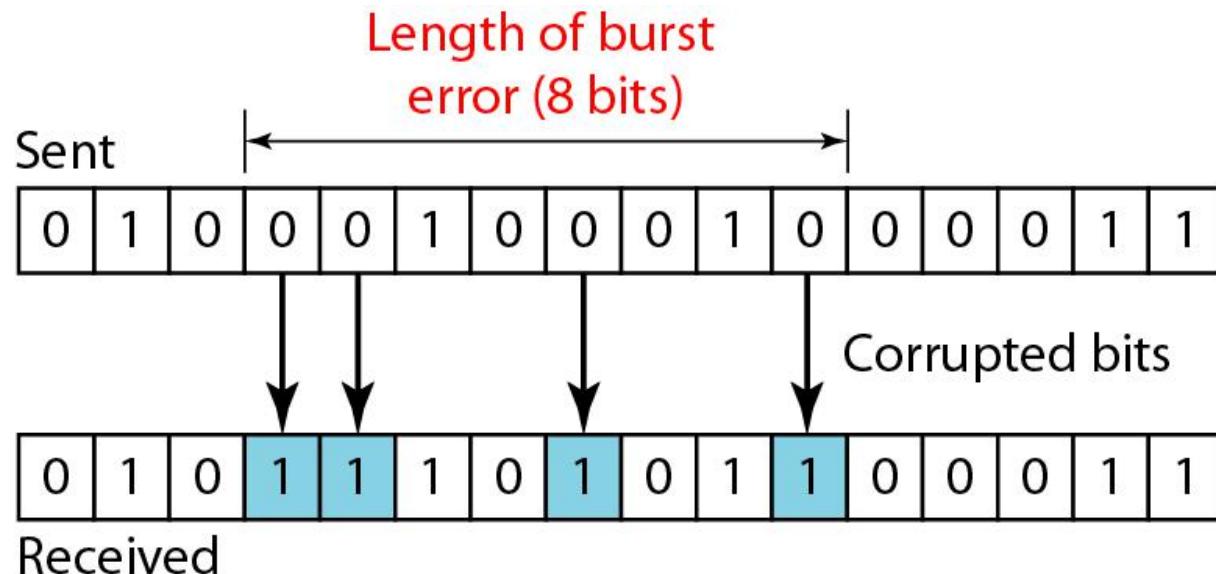
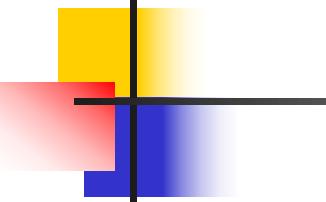


Figure 10.2 8位突发性差错 (*Burst error of length 8*)



Note

为了检测或纠正差错，需要发送除了数据外的额外（冗余）位。

检错和纠错

- 检错：是否发生错误
- 纠错：哪里发生错误
- 纠错方法：前向纠错和重传

编码方案

块编码 & 卷积编码

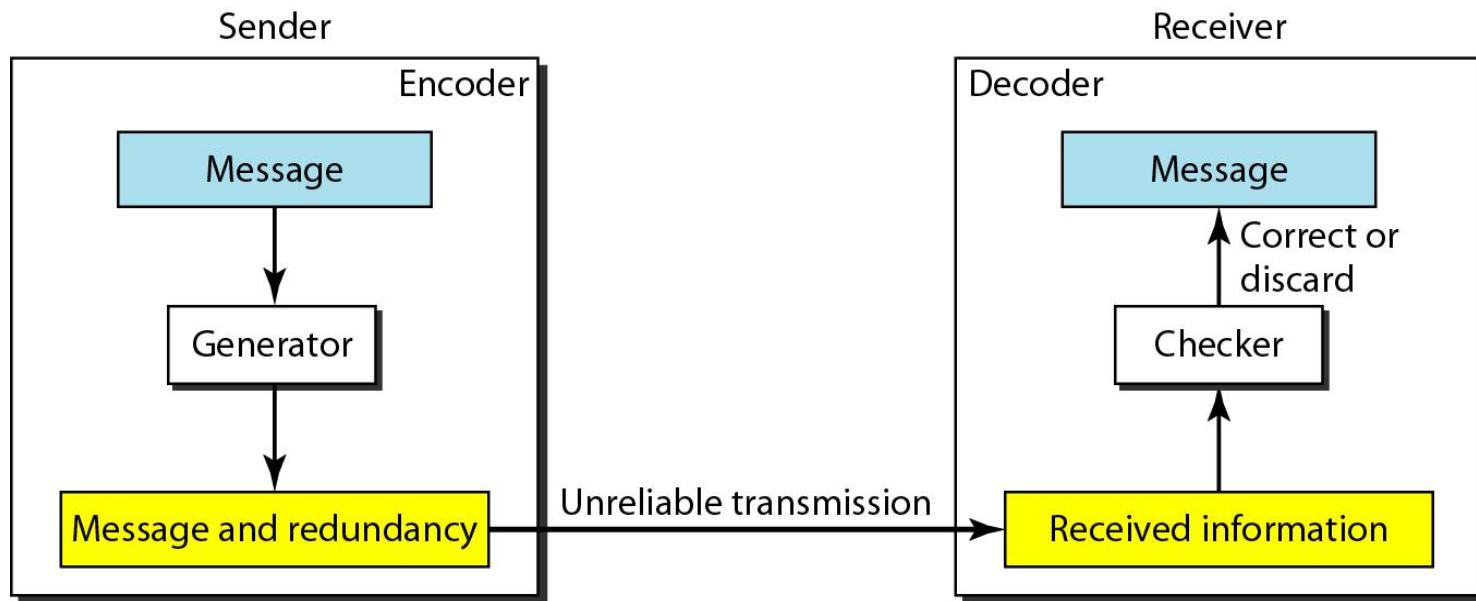


Figure 10.3 编码器和译码器的结构

模运算

使用有限范围的正整数，其上限称为模数N。在模N运算中，只使用0到N-1的整数。

$$0 \oplus 0 = 0$$

$$1 \oplus 1 = 0$$

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

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

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

$$\begin{array}{r} 1 & 0 & 1 & 1 & 0 \\ \oplus & 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 \end{array}$$

c. Result of XORing two patterns

Figure 10.4 模 2运算和异或运算关系

10-2 BLOCK CODING 块编码

在块编码中，把报文划分成块，每块有 k 位，称为**数据字**，并增加 r 个冗余位使其长度变为 $n = k + r$ ，形成 n 位的块称为**码字**。

Topics discussed in this section:

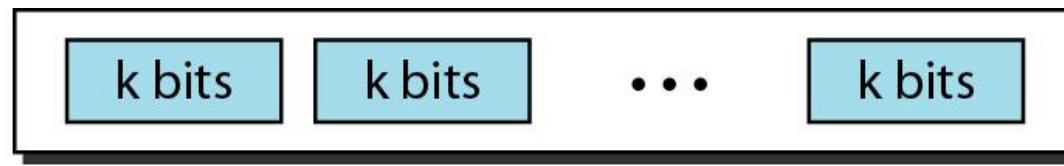
Error Detection 检错

Error Correction 纠错

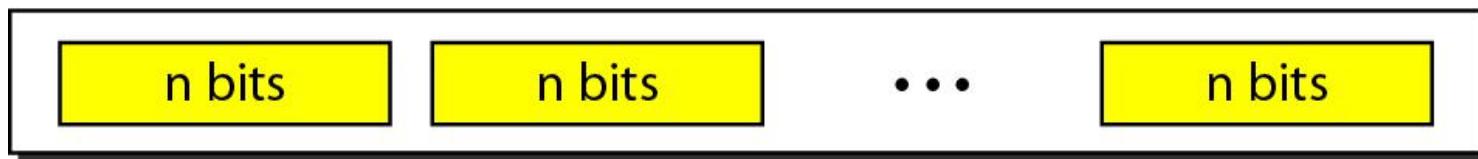
Hamming Distance 汉明距离

Minimum Hamming Distance 最小汉明距离

Figure 10.5 数据字和码字



2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

Table 4.2 4B/5B 编码

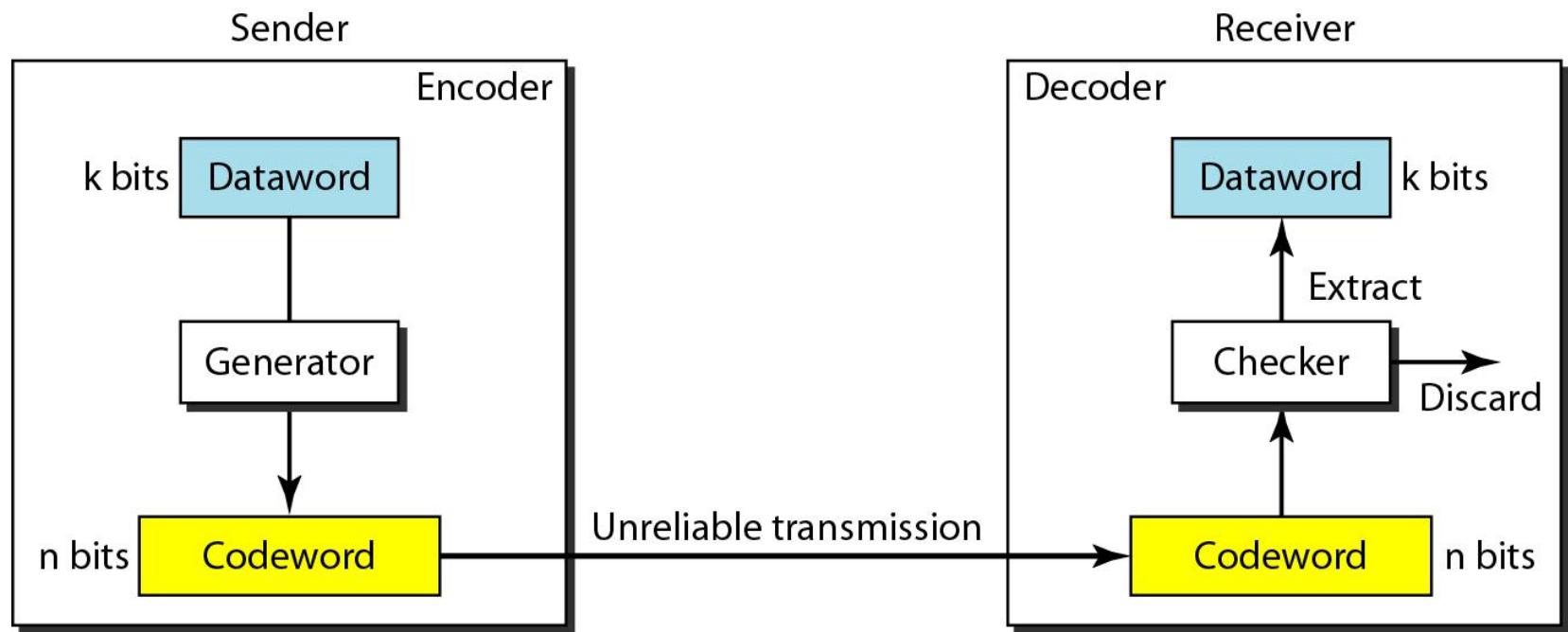
<i>Data Sequence</i>	<i>Encoded Sequence</i>	<i>Control Sequence</i>	<i>Encoded Sequence</i>
0000	11110	Q (Quiet)	00000
0001	01001	I (Idle)	11111
0010	10100	H (Halt)	00100
0011	10101	J (Start delimiter)	11000
0100	01010	K (Start delimiter)	10001
0101	01011	T (End delimiter)	01101
0110	01110	S (Set)	11001
0111	01111	R (Reset)	00111
1000	10010		
1001	10011		
1010	10110		
1011	10111		
1100	11010		
1101	11011		
1110	11100		
1111	11101		

差错检测

- 接收方有（或能找到）有效码字的列表
- 原来码字已改变成无效的码字

如果码字在传输中被破坏，但接收到的码字仍然是一个有效的码字，差错则无法被检测到。

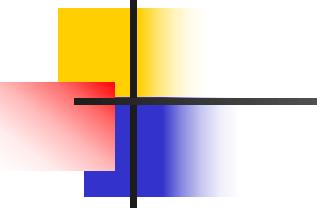
Figure 10.6 块编码的差错检测过程



例 10.2 只能检测 1 比特检错码的例子。假设 $k = 2$, $n = 3$, 发送方把 01 编码成 011 发送给接收方。

1. 接收方收到 011。这是有效码字，接收方提取出数据字 01。
2. 码字在传输中被破坏，接收方收到 111，是无效码字，丢弃。
3. 码字在传输中被破坏，接收方收到 000，是有效码字。接收方提取出数据字 00，这两个破坏位形成的差错无法被检测到。

<i>Datawords</i>	<i>Codewords</i>
00	000
01	011
10	101
11	110



Note

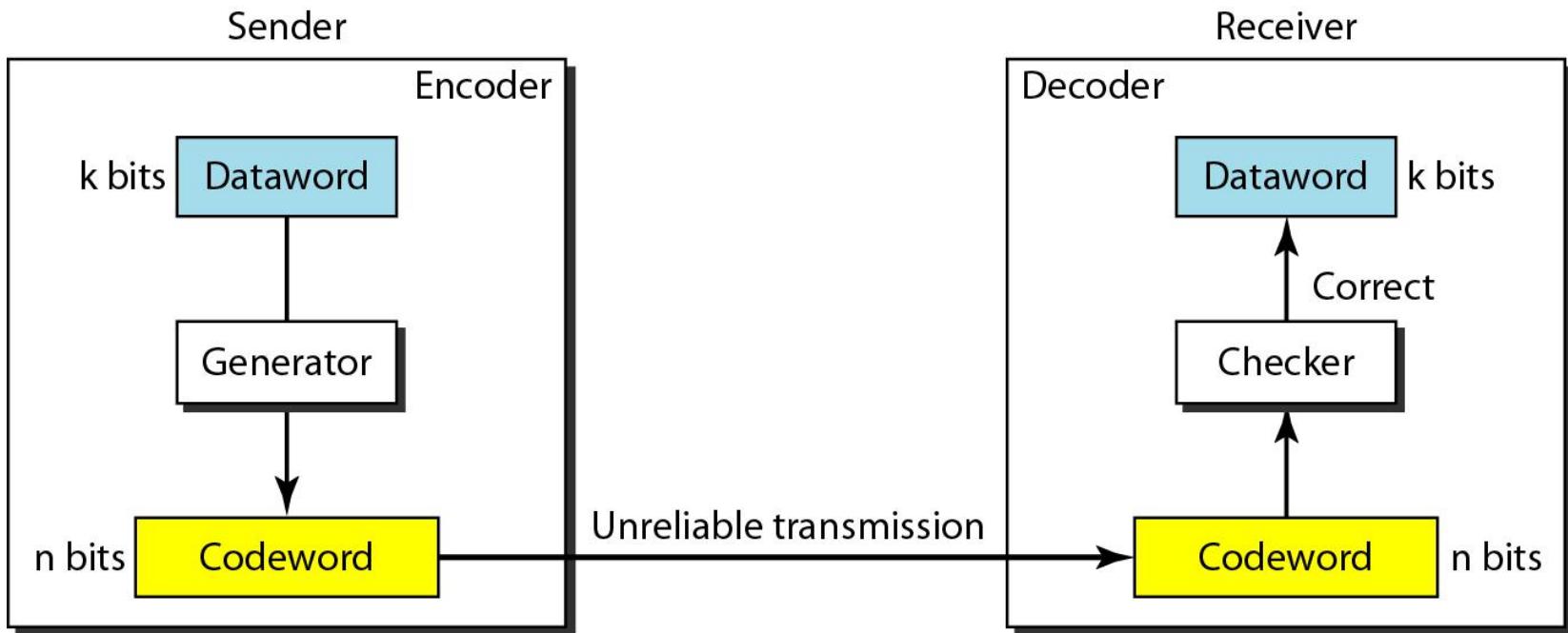
检错码是为某些类型的差错而设计的，
因此只能检测这些类型的差错，
其它类型的差错则无法检测到。

差错纠正

- 检错：接收方只需知道接收到的码字是无效的。
 -
- 纠错：接收方需要知道发送的原来码字。

纠错比检错更复杂，需要更多冗余位。

Figure 10.7 纠错码的编码器和译码器的结构



Example 10.3

只能纠正 1 比特错误的纠错码例子。假设数据字是 01，生成码字 01011。这个码字在传输中被破坏，接收方收到 01001。

检错：码字不在表中，说明发生了差错。

纠错：假定只有 1 位被破坏。与表中码字比较，只有第二个码字与接收到的码字有一位不同，接收方用 01011 代替 01001，根据表格找到原来数据。

<i>Dataword</i>	<i>Codeword</i>
00	00000
01	01011
10	10101
11	11110

汉明距离

- ◆ 定义：两个相同长度字的汉明距离是对应位不同的数量。它是差错编码的核心概念。
- ◆ 表示： $d(x, y)$ （总是大于0）
- ◆ 计算：对两个字异或后计算 1 的个数

Example 10.4

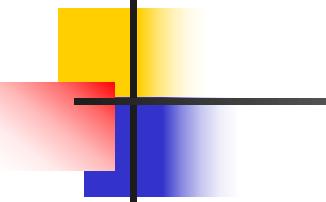
用异或计算汉明距离

1. 汉明距离 $d(000, 011)$ 是 2, 因为

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

2. 汉明距离 $d(10101, 11110)$ 是 3, 因为

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



Note

最小汉明距离是一组字中所有可能对的
最小汉明距离。

Example 10.5

求表 10.1 编码方案的最小汉明距离。

Solution

计算所有的汉明距离：

$$\begin{array}{llll} d(000, 011) = 2 & d(000, 101) = 2 & d(000, 110) = 2 & d(011, 101) = 2 \\ d(011, 110) = 2 & d(101, 110) = 2 & & \end{array}$$

得到 d_{\min} 是 2。

<i>Datawords</i>	<i>Codewords</i>
00	000
01	011
10	101
11	110

Example 10.6

求表 10.2 编码方案的最小汉明距离。

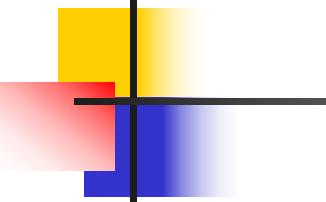
Solution

计算所有的汉明距离：

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

得到 d_{\min} 为 3。

<i>Dataword</i>	<i>Codeword</i>
00	00000
01	01011
10	10101
11	11110



Note

- ◆ 编码方案写成 $C(n, k)$ 和一个单独的 d_{\min} 表达式， n 是码字的长度， k 是数据的位数， d_{\min} 是最小汉明距离。
- ◆ 接收到的码字和发送的码字之间的汉明距离是传输中被破坏的位数。
- ◆ 为了保证检测出最多 s 个错误，块编码中最小汉明距离一定是 $d_{\min} = s + 1$ 。

Example 10.7

表10.1第一个编码方案的最小汉明距离为 2，所以只能检测单比特错误。

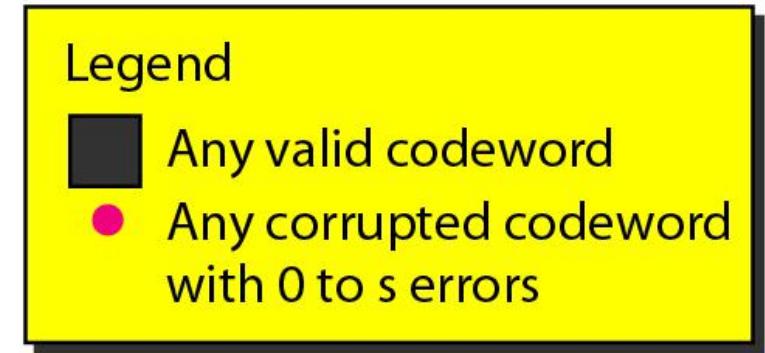
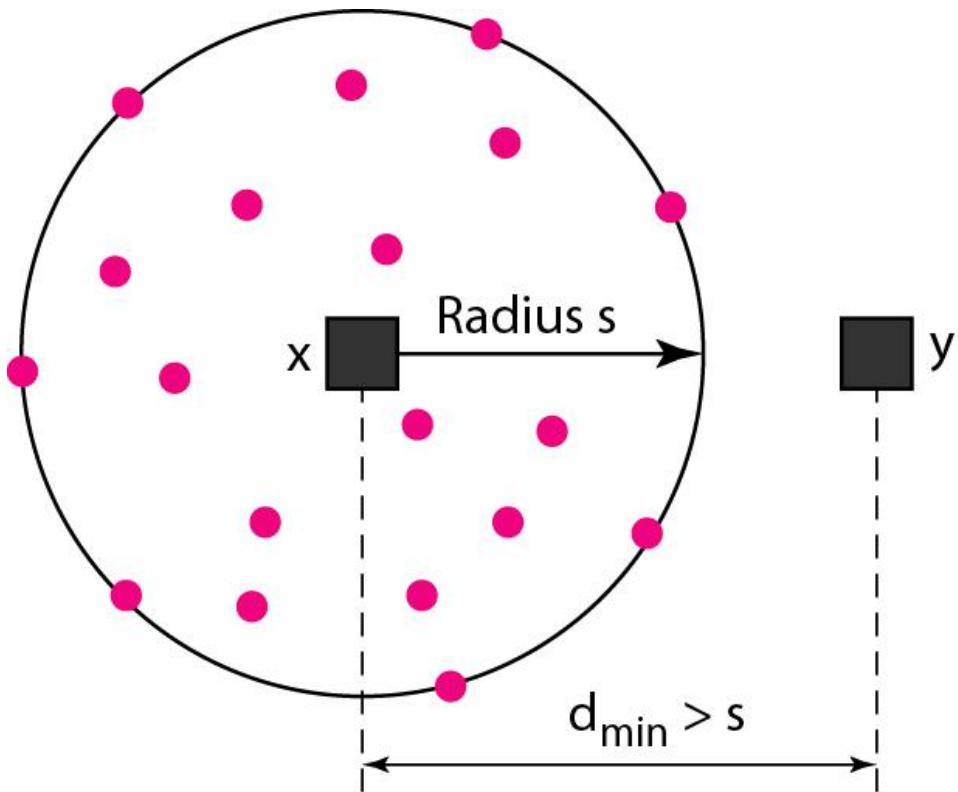
<i>Datawords</i>	<i>Codewords</i>
00	000
01	011
10	101
11	110

Example 10.8

表10.2第二个编码方案的最小汉明距离为3，所以最多能检测 2 比特差错。

<i>Dataword</i>	<i>Codeword</i>
00	00000
01	01011
10	10101
11	11110

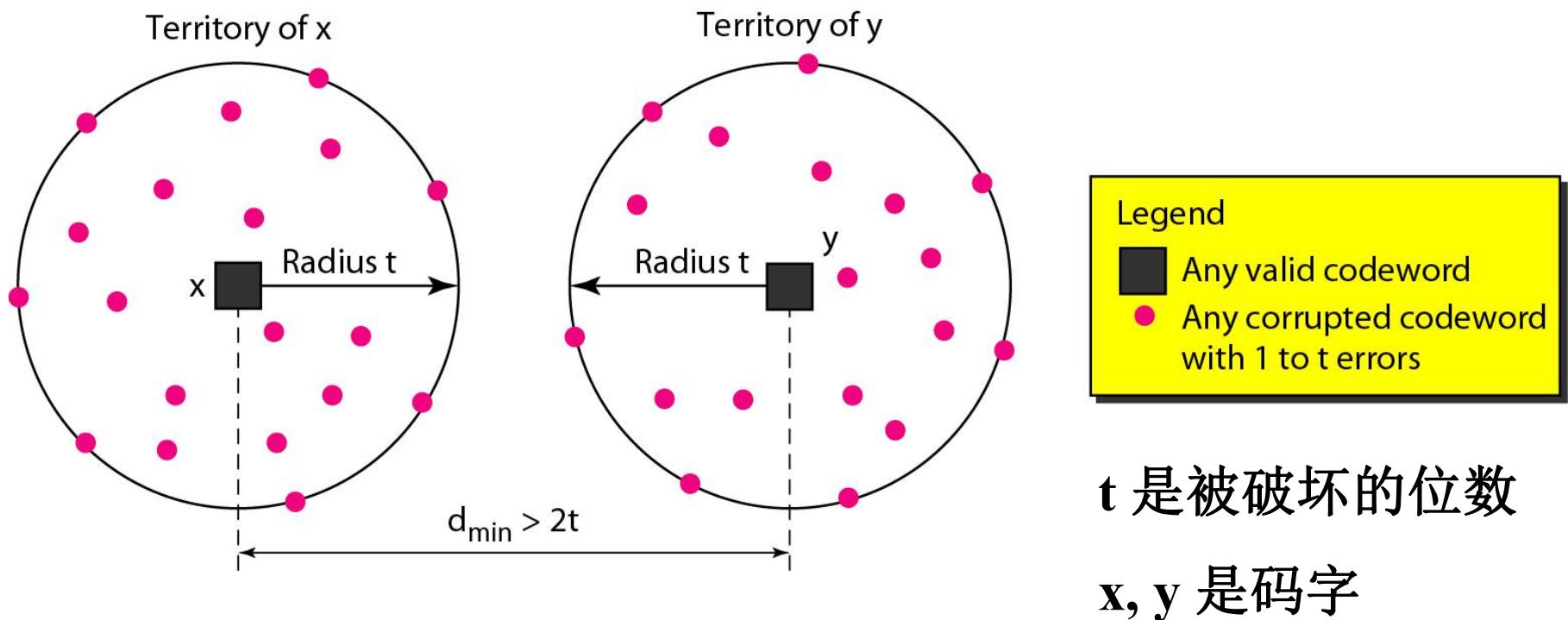
Figure 10.8 差错检测中 d_{\min} 的几何意义

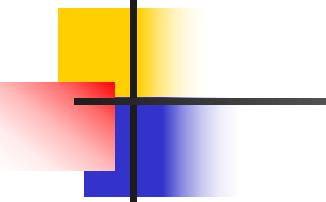


s 是差错的位数

x是码字

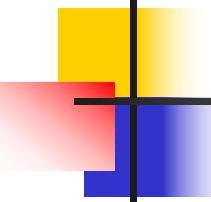
Figure 10.9 差错纠正中 d_{\min} 的几何意义





Note

为了保证最多能纠正 t 个差错，块编码中的
最小汉明距离是 $d_{\min} = 2t + 1$ 。



Example 10.9

$d_{min} = 4$ 的编码方案，检错和纠错能力分别是多少？

Solution

这个方案最多检测到 3个差错 ($s = 3$)，但是它只能纠正 1个差错。即如果这个编码用于纠错，它的部分能力被浪费了。

纠错码需要的最小距离是奇数 (3, 5, 7, ...)。

10-3 LINEAR BLOCK CODES

目前，几乎所有使用的块编码都属于一个称为**线性块编码**的子集。在线性块编码中，任两个有效码字的异或（即模二加）生成另一个有效码字。（需要抽象代数里有限域的概念）

Topics discussed in this section:

Minimum Distance for Linear Block Codes 线性块码的最小距离
Some Linear Block Codes 一些线性块编码

Example 10.10

表10.1和10.2都属于线性块编码。任何两个码字异或的结果是有效码字。

<i>Datawords</i>	<i>Codewords</i>
00	000
01	011
10	101
11	110

<i>Dataword</i>	<i>Codeword</i>
00	00000
01	01011
10	10101
11	11110

Example 10.11

线性块编码的最小汉明距离：具有最小1的个数的非0有效码字中 1的个数。

表 10.1 中，非零码字 1 的个数分别是 2，2 和 2，所以最小汉明距离是 $d_{min} = 2$ 。在表 10.2 中，非零码字的 1 的个数分别是 3，3 和 4，所以得到 $d_{min} = 3$ 。

<i>Datawords</i>	<i>Codewords</i>
00	000
01	011
10	101
11	110

<i>Dataword</i>	<i>Codeword</i>
00	00000
01	01011
10	10101
11	11110

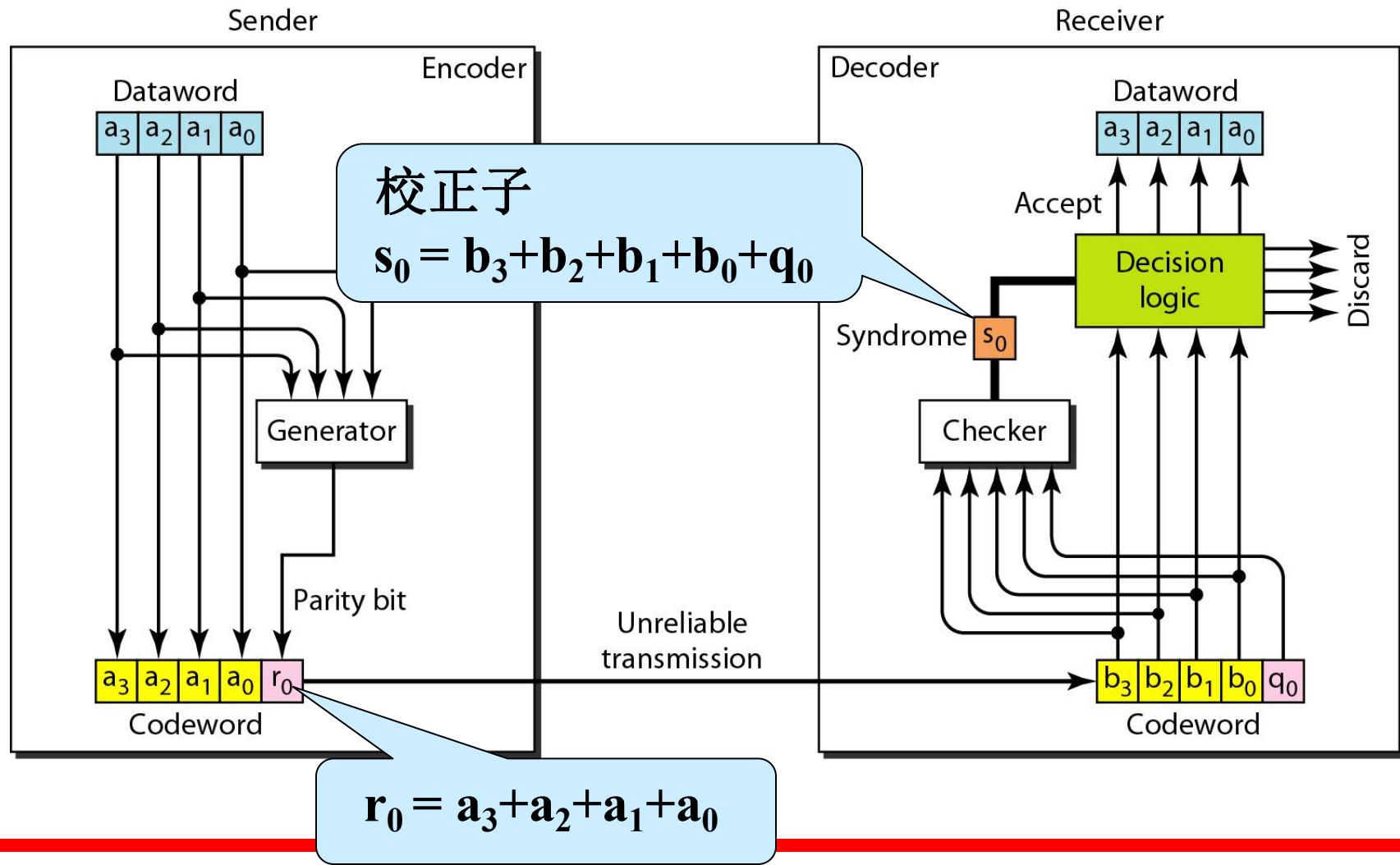
简单奇偶校验编码

简单奇偶校验码是 $n = k + 1$, 且 $d_{\min} = 2$ 的单比特检错编码。

<i>Datawords</i>	<i>Codewords</i>	<i>Datawords</i>	<i>Codewords</i>
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

Table 10.3 简单的奇偶校验码 C(5, 4)

Figure 10.10 奇偶校验码的编码器和译码器（模二加运算）



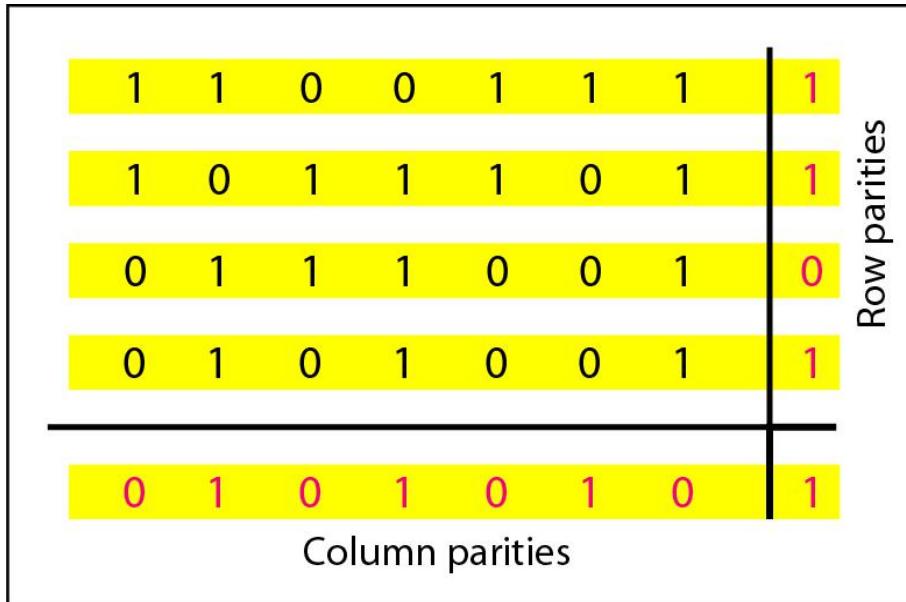
Example 10.12

假设发送方发送数据字 1011，生成码字 10111。

1. 没有差错 —— 正确生成数据字
2. 单个位差错 —— 不生成数据字
3. 两个位差错 —— 生成错误数据字，未发现差错
4. 三个位差错 —— 不生成数据字，发现差错

简单奇偶校验码能检出奇数个差错

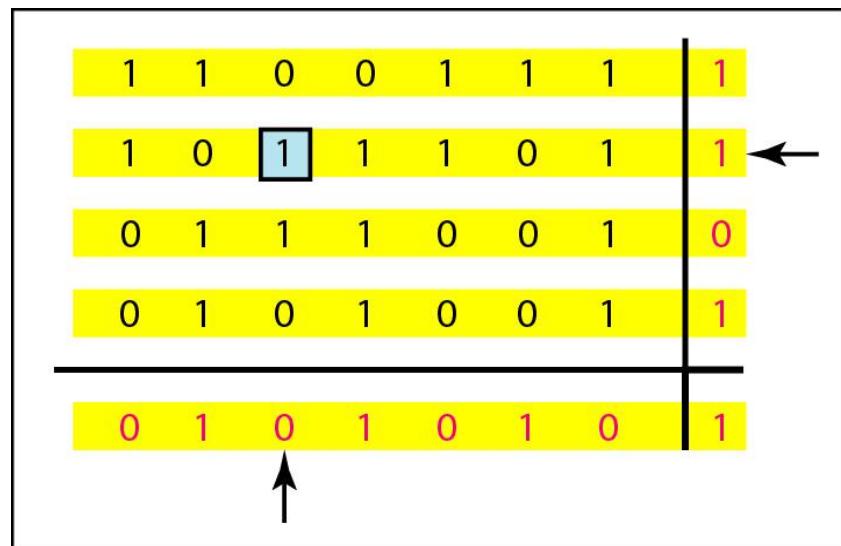
两维奇偶校验编码



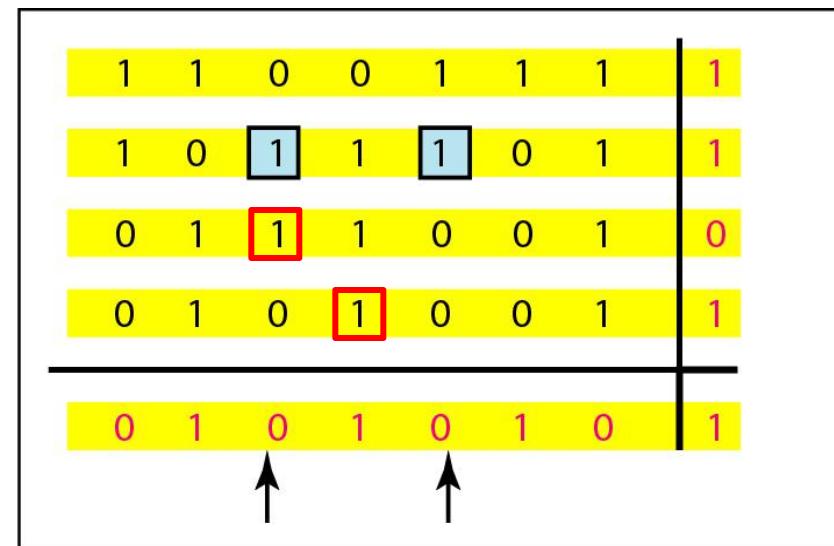
a. Design of row and column parities

Figure 10.11 两维奇偶校验编码设计

Figure 10.11 两维奇偶校验



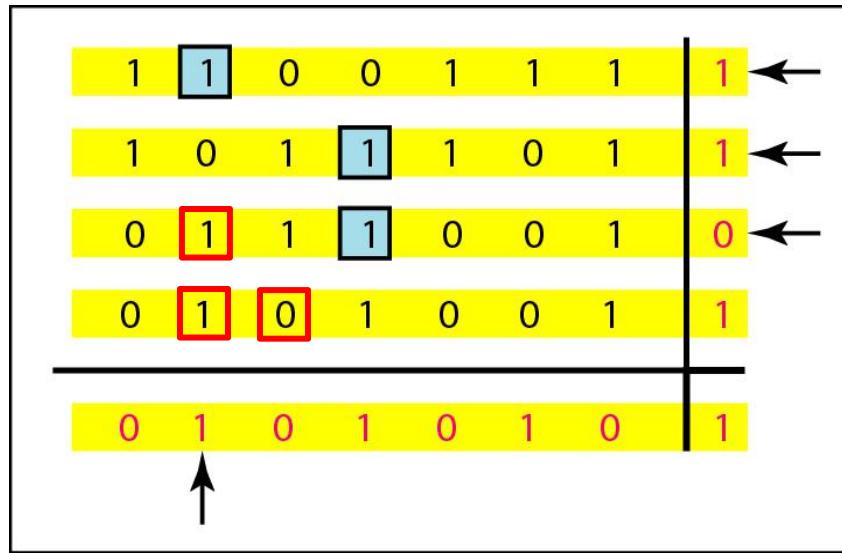
b. One error affects two parities



c. Two errors affect two parities

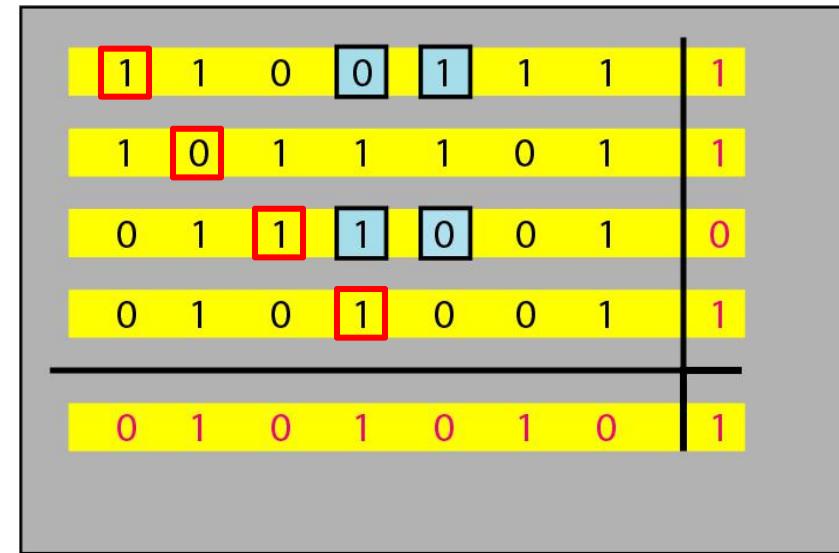
at least

Figure 10.11 两维奇偶校验



d. Three errors affect four parities

at least affect two parities



e. Four errors cannot be detected

sometimes may be detected

两维奇偶校验能检测出所有3位或3位以下的错误（因为此时至少在某一行或某一列上有一位错）、奇数位错以及很大一部分偶数位错。

汉明编码

本书只讨论最小汉明距离为 $d_{\min} = 3$ 的汉明码，它最多能检测出 2 位差错和最多纠正 1 位差错。

选择 $m \geq 3$ 的整数，则码字长 n 、数据字长 k 和校验位 r 的关系为

$$n = 2^m - 1, \quad k = n - m, \quad r = m$$

Table 10.4 汉明码 C(7, 4)

<i>Datawords</i>	<i>Codewords</i>	<i>Datawords</i>	<i>Codewords</i>
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

$$m = 3, \ n = 7, \ k = 4 \rightarrow C(7, 4)$$

Figure 10.12 汉明码的编码器和解码器的结构（冗余位也是模2运算）

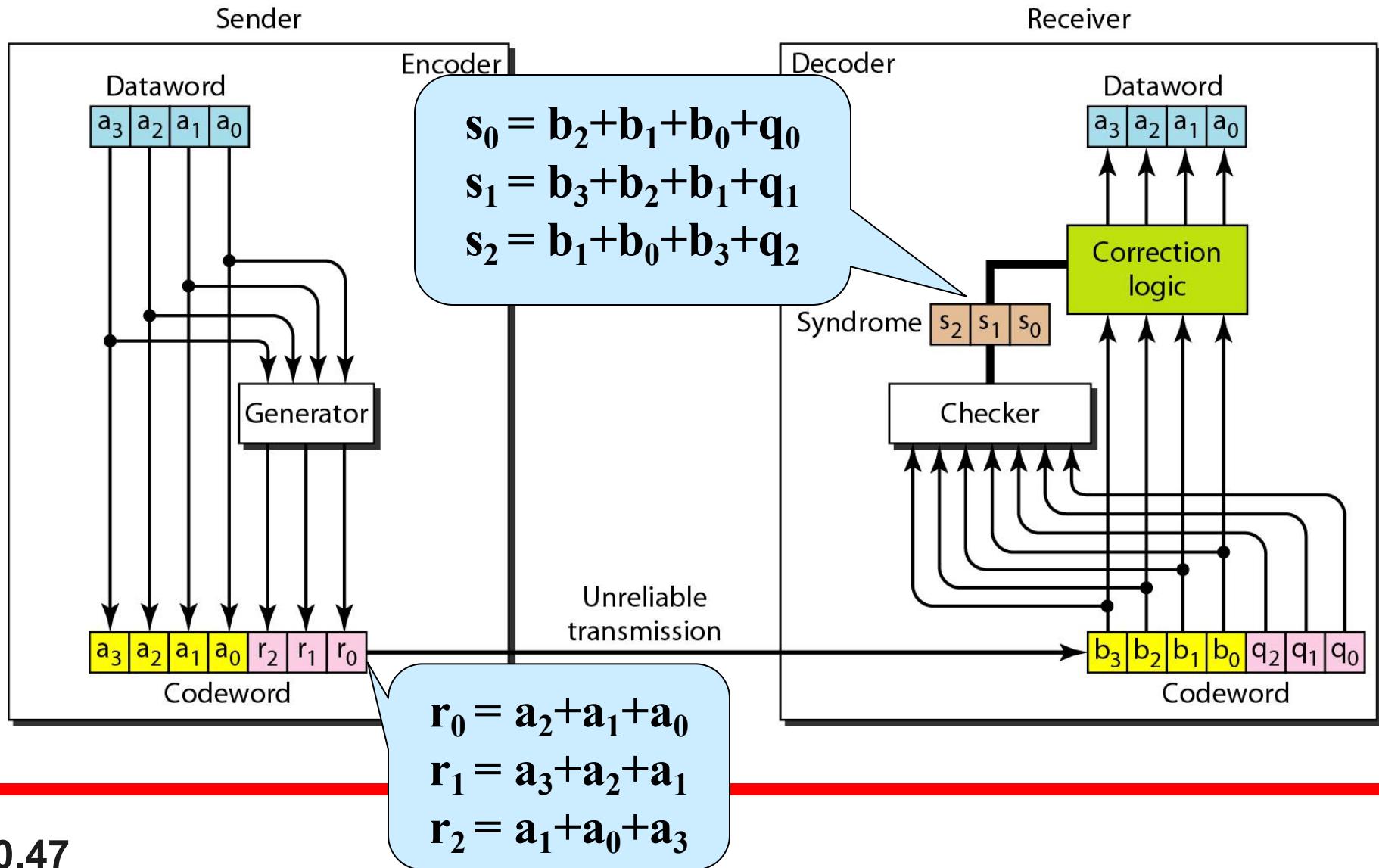


Table 10.5 译码器纠错逻辑分析器的逻辑判定

<i>Syndrome</i>	000	001	010	011	100	101	110	111
<i>Error</i>	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

- 生成器不关心无差错或奇偶位的差错
- 校正子的值是每个比特值的组合

$$\begin{aligned}s_0 &= b_2 + b_1 + b_0 + q_0 \\s_1 &= b_3 + b_2 + b_1 + q_1 \\s_2 &= b_1 + b_0 + b_3 + q_2\end{aligned}$$

Example 10.13

跟踪 3个数据字从发送端到目的端的路径。

1. 数据字 0100 变成码字 0100011， 接收到码字0100011。
校正子是 000（无差错）， 最后数据字是 0100。
2. 数据字 0111 变成码字 0111001， 接收到码字 0011001。
校正子是 011， b_2 发生差错， 反转后得到数据字 0111。
3. 数据字 1101 变成码字 1101000， 接收到码字0001000。
校正子是101， b_0 发生差错， 反转后得到数据字 0000，
是错误的数据字。说明此编码检测不到两个差错。

Example 10.14

若数据字至少为 7 比特，计算满足这个条件的 n 和 k 。

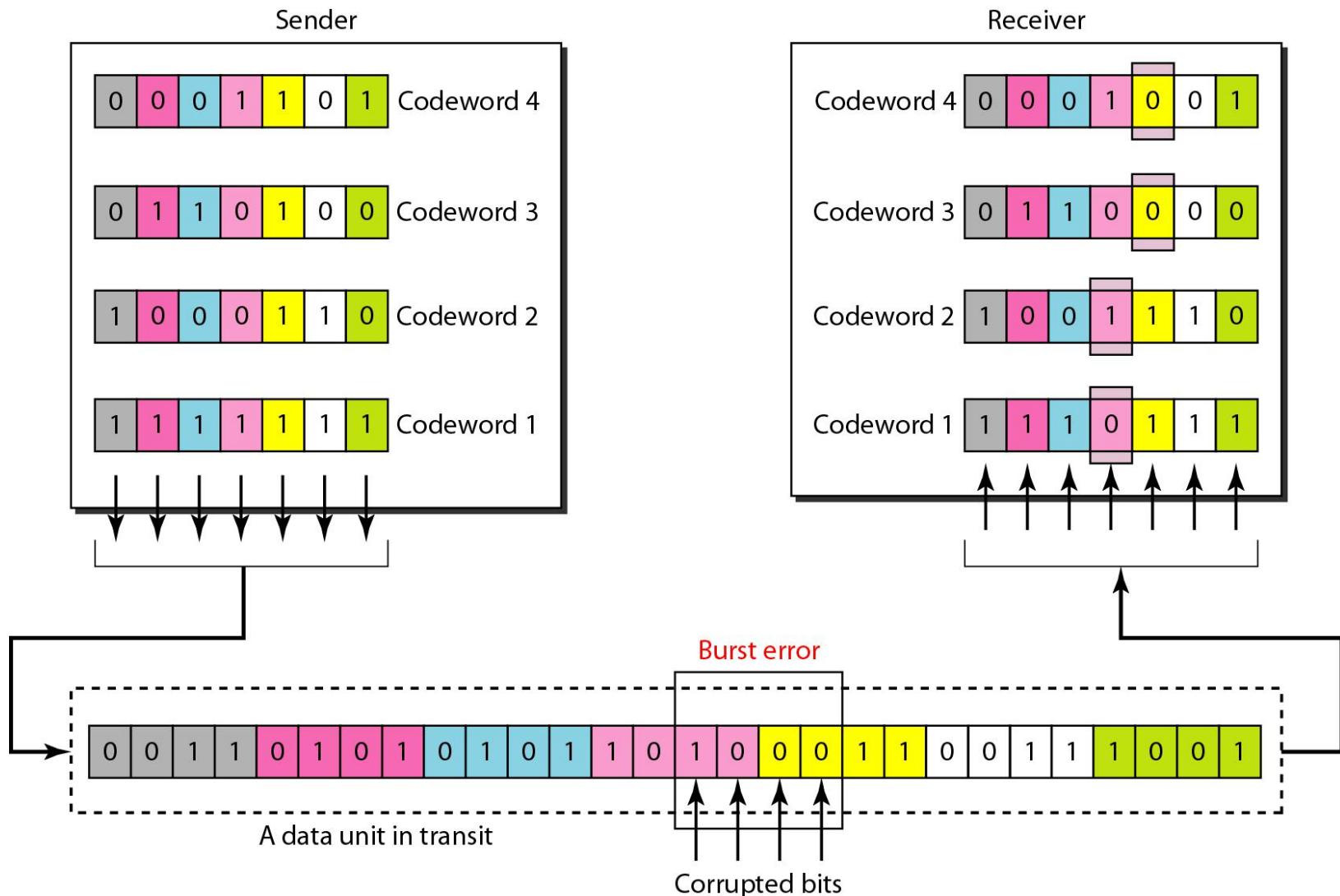
Solution

要求 $k = n - m \geq 7$, 或者 $2^m - 1 - m \geq 7$ 。

1. 如果 $m = 3$, 则 $n = 2^3 - 1 = 7$, $k = 7 - 3 = 4$, 不符合。
2. 如果 $m = 4$, 则 $n = 2^4 - 1 = 15$, $k = 15 - 4 = 11$, 满足条件。因此编码是:

C(15, 11)

Figure 10.13 汉明码提高突发错误能力



10-4 CYCLIC CODES 循环码

循环码是有一个附加性质的特殊线性块编码。这个性质是如果码字循环移位（旋转），其结果是另一个循环码字。

Topics discussed in this section:

- | | |
|----------------------------|---------|
| Cyclic Redundancy Check | 循环冗余校验码 |
| Hardware Implementation | 硬件实现 |
| Polynomials | 多项式 |
| Cyclic Code Analysis | 循环码性能分析 |
| Advantages of Cyclic Codes | 循环码的优点 |
| Other Cyclic Codes | 其它循环码 |

Table 10.6 $C(7, 4)$ 的循环冗余校验码 CRC

<i>Dataword</i>	<i>Codeword</i>	<i>Dataword</i>	<i>Codeword</i>
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

循环冗余校验 (Cyclic Redundancy Check, CRC)

- k 位数据字, n 位码字, 数据字右边加上 $n - k$ 位 0, 变成 n 位后传给生成器;
- 生成器用长度为 $n - k + 1$ 的除数去除增加长度后的数据字 (模 2 除法) ;
- 除法的商被丢弃, $n - k$ 位余数加到数据字上生成码字。
- 校验器用相同的除数去除码字, 得到的余数是 $n - k$ 位校正子;
- 如果校正子全 0, 码字最左边 k 位被接收为数据字, 否则丢弃。

Figure 10.14 CRC 编码器和译码器

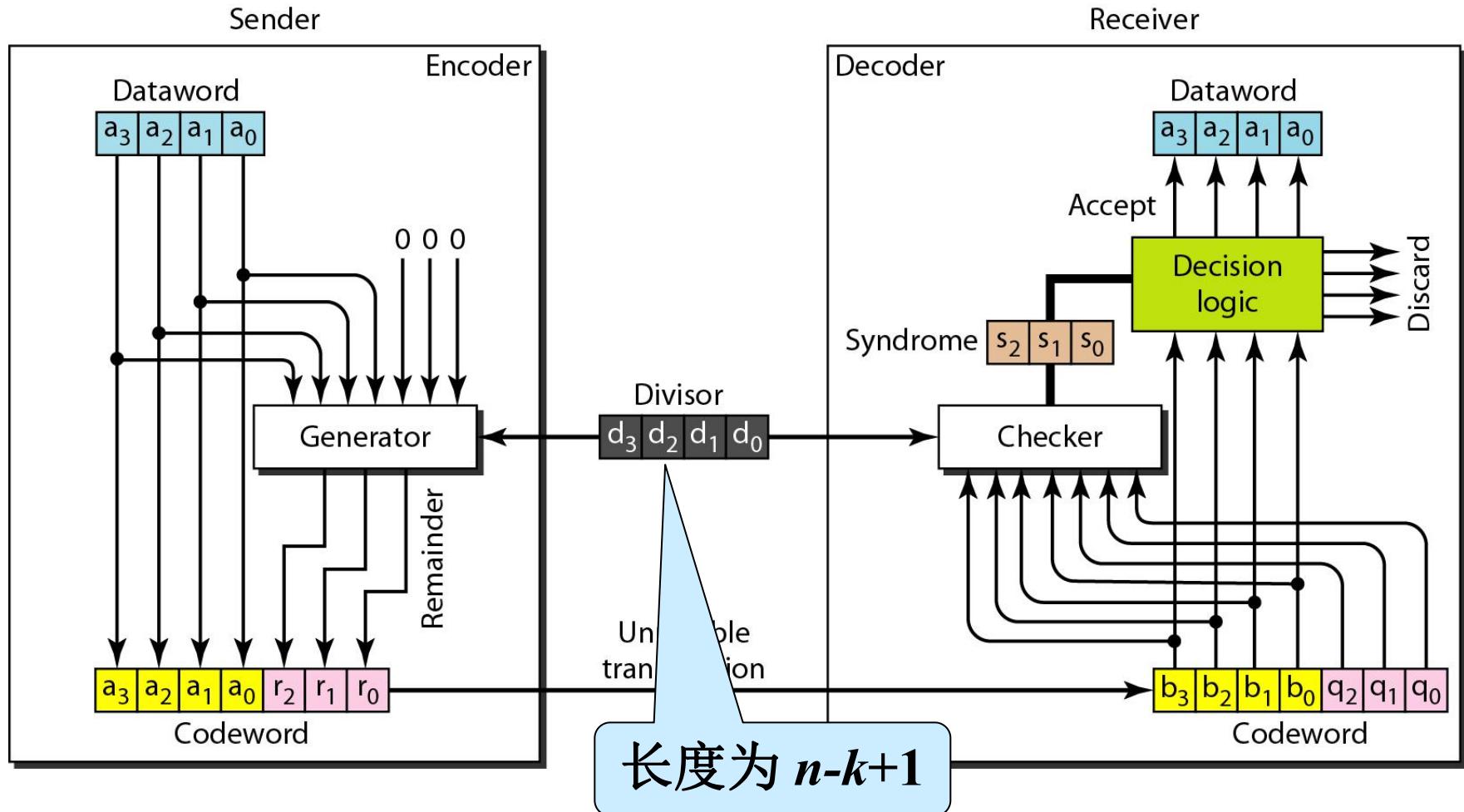


Figure 10.15 CRC编码器中的除法（模2运算）

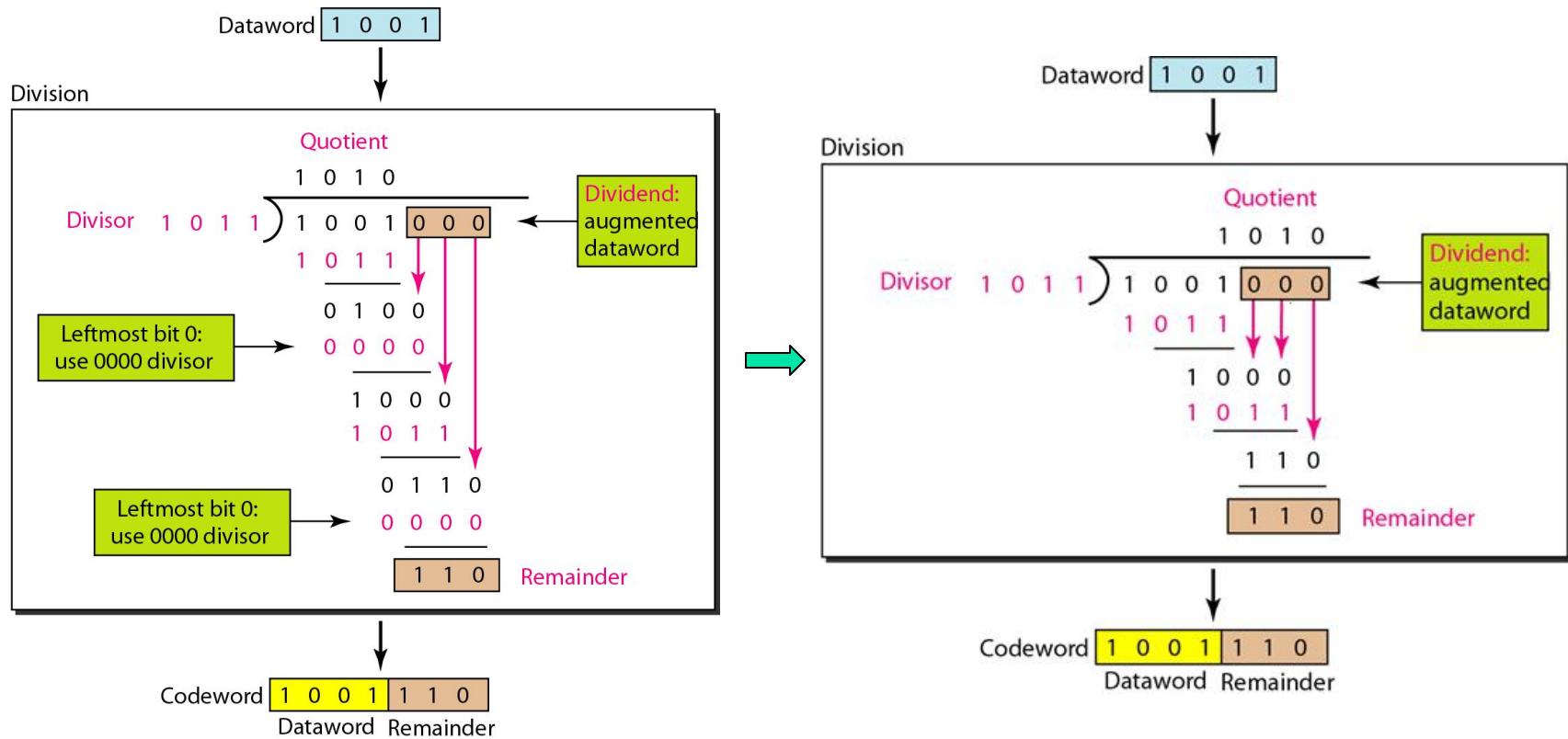


Figure 10.16 CRC的译码器中的除法（除尽和除不尽）

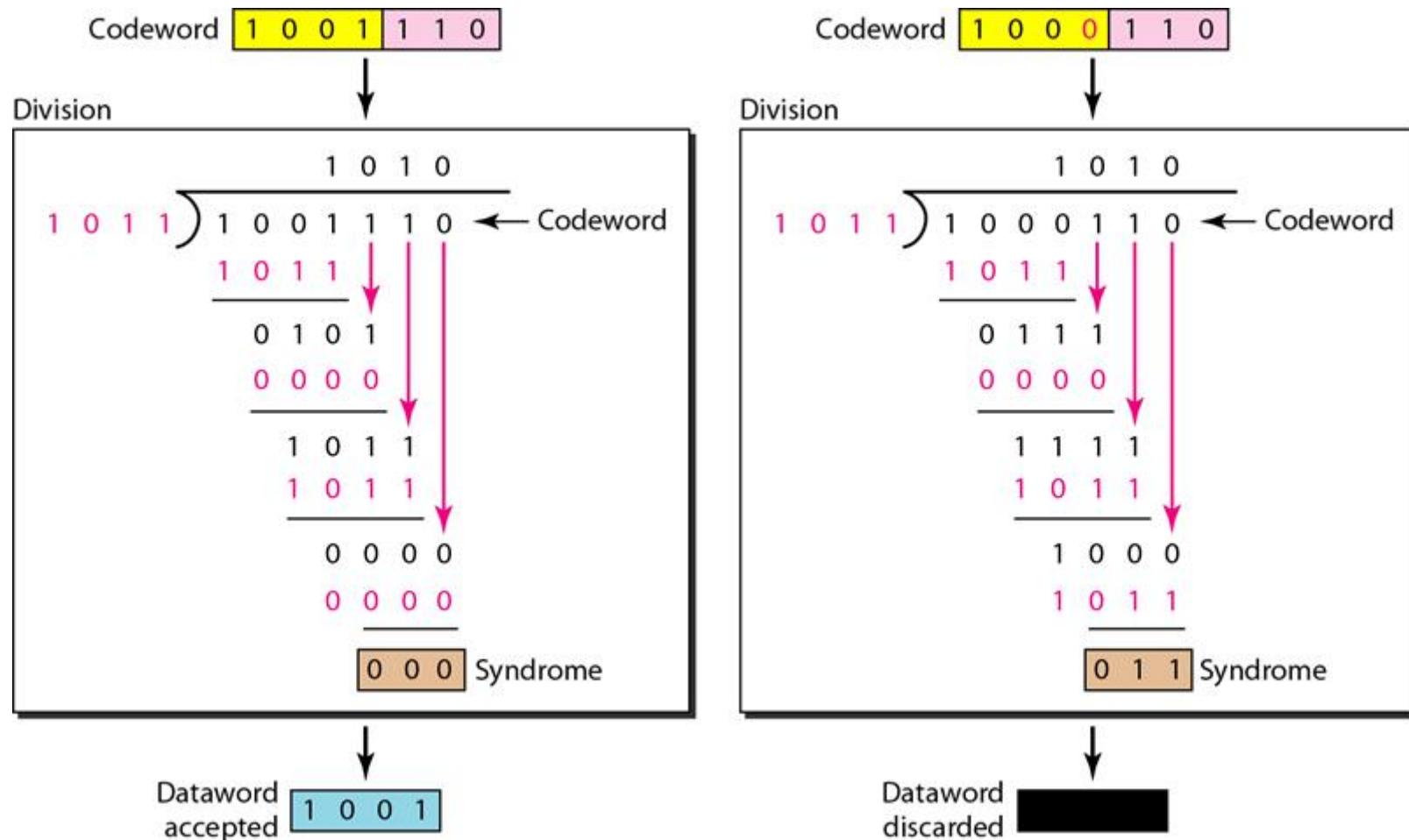


Figure 10.17 CRC的硬件实现

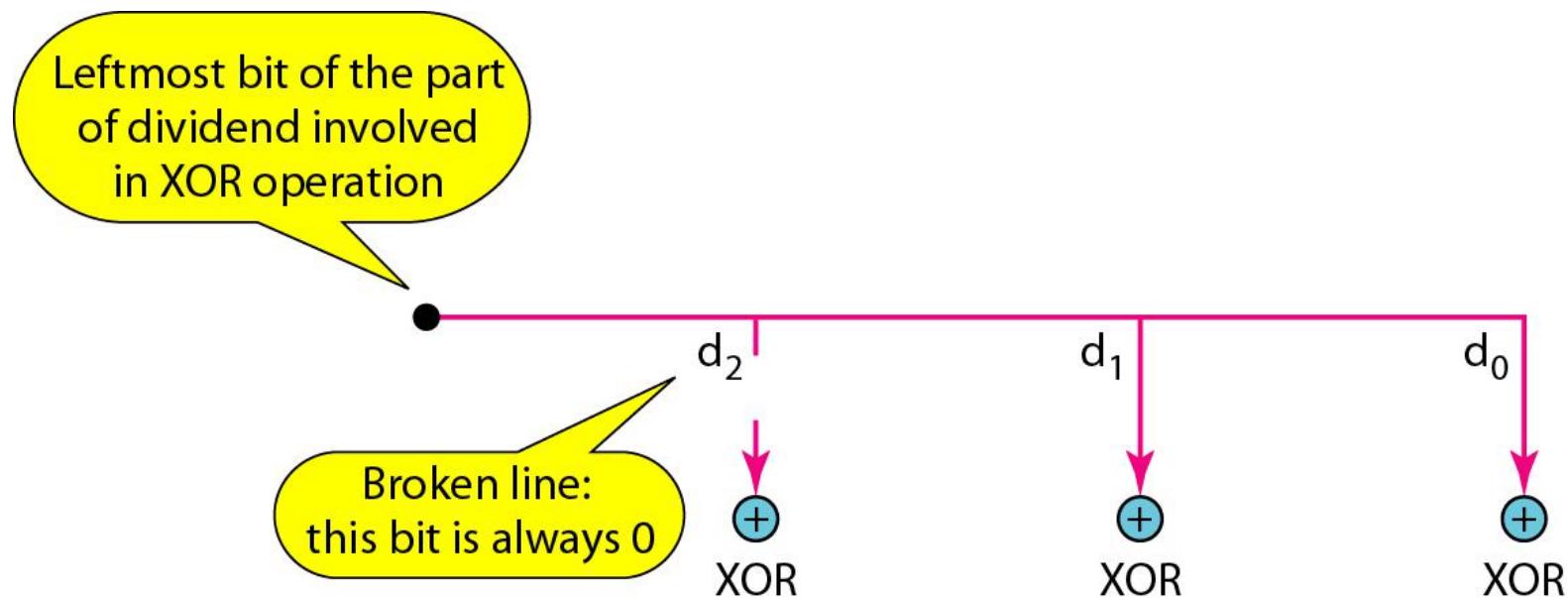


Figure 10.18 CRC编码器中除法模拟

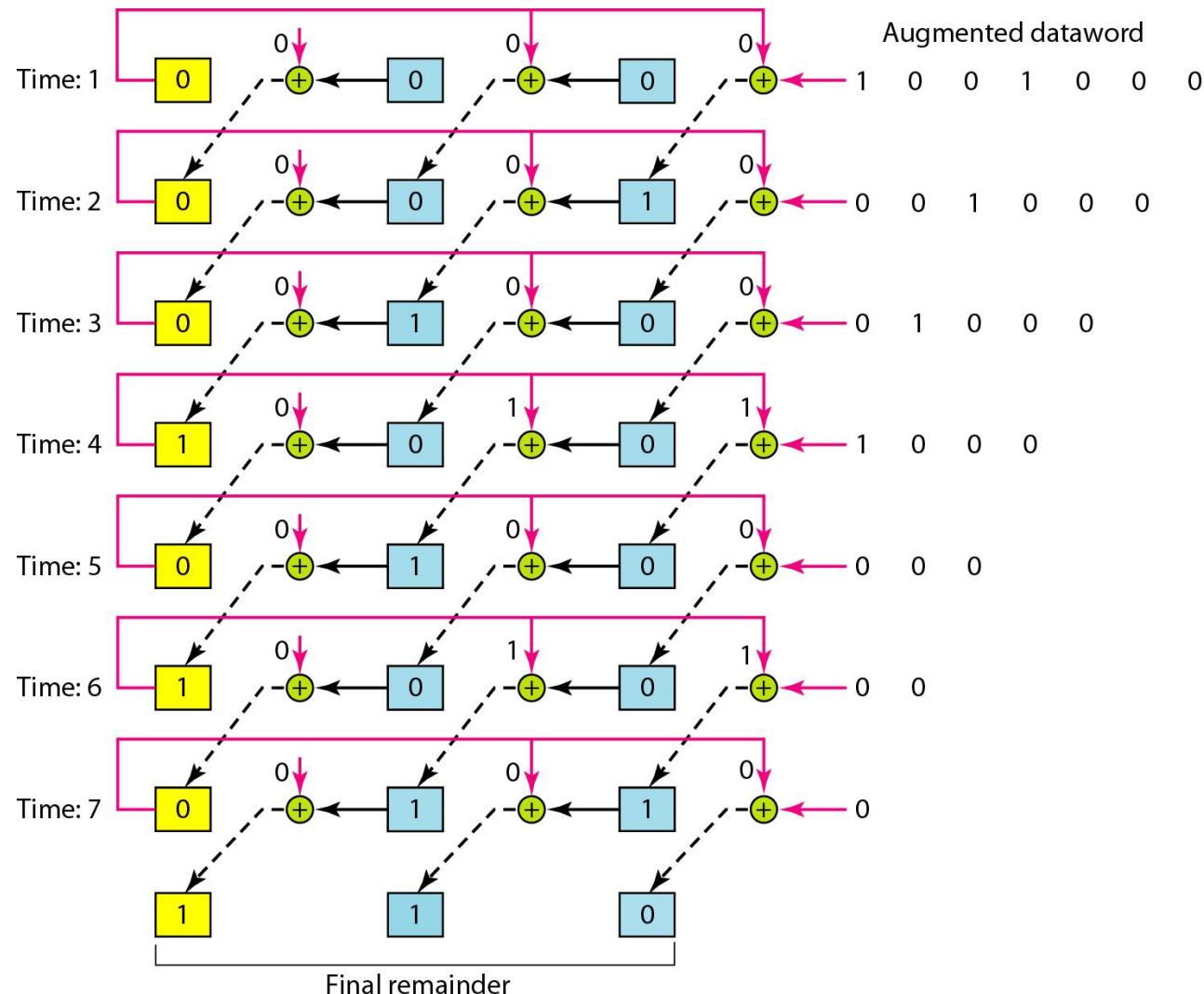


Figure 10.19 使用移位寄存器的CRC编码器

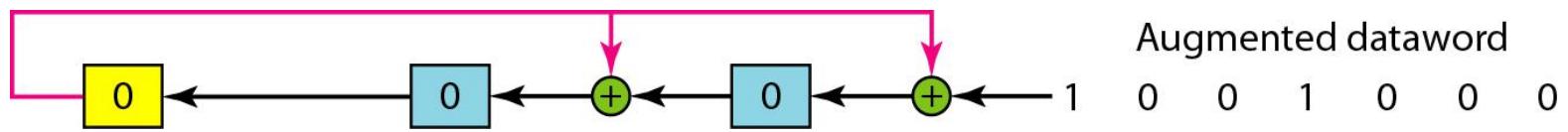
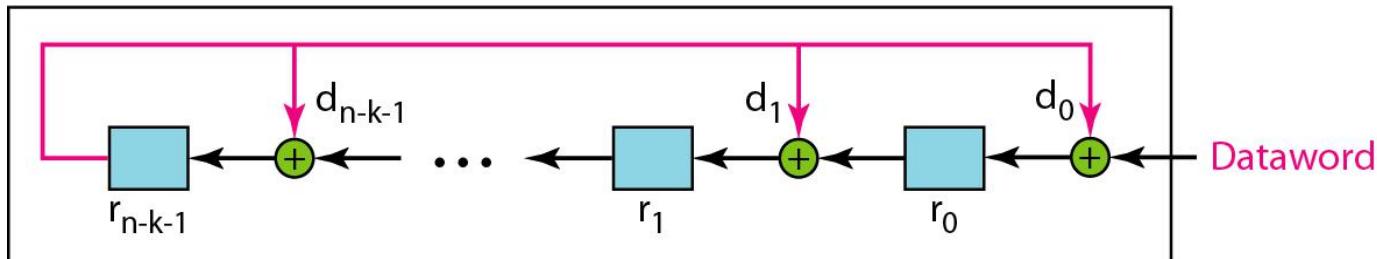


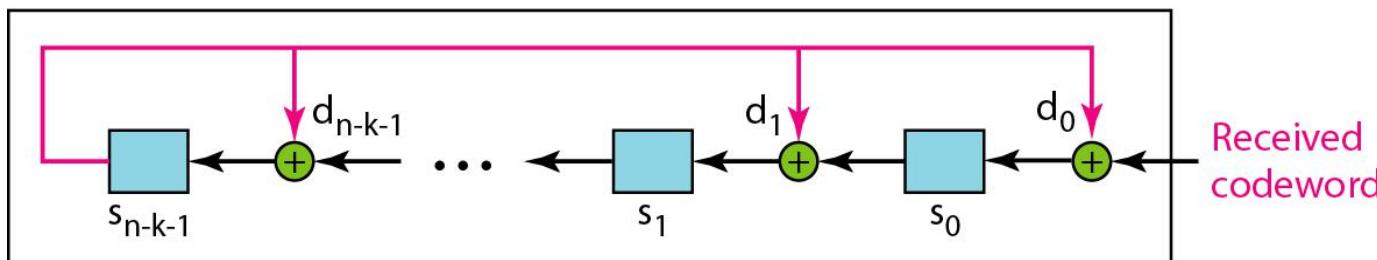
Figure 10.20 CRC编码器和译码器的一般设计

Note:

The divisor line and XOR are missing if the corresponding bit in the divisor is 0.



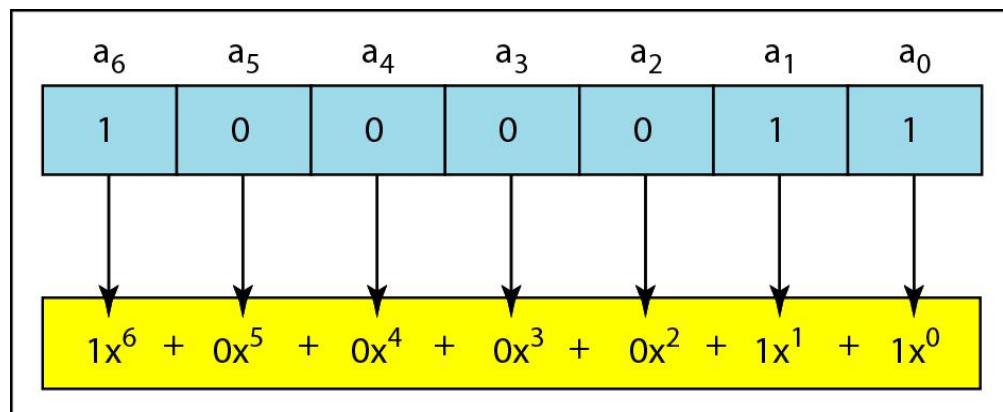
a. Encoder



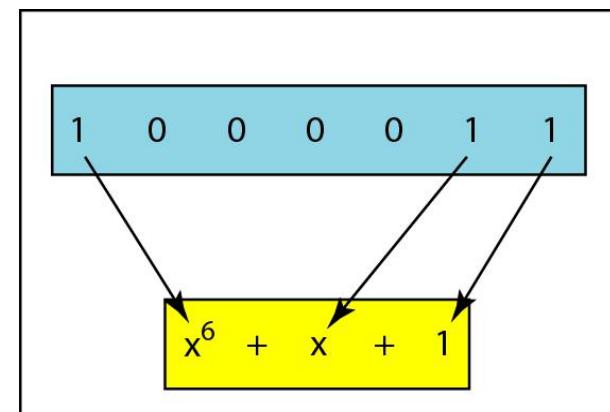
b. Decoder

多项式

若干个0和1组成的模式可以表示为以0和1为系数的多项式。
幂次表示位所在的位置，系数表示位的值。



a. Binary pattern and polynomial



b. Short form

Figure 10.21 用多项式表示二进制字

多项式的运算

■ 加减法

- 加减相同幂次项的系数
- 加和减相同
- 加减是删除相同项，保留不同项

■ 乘除法：幂次相加减

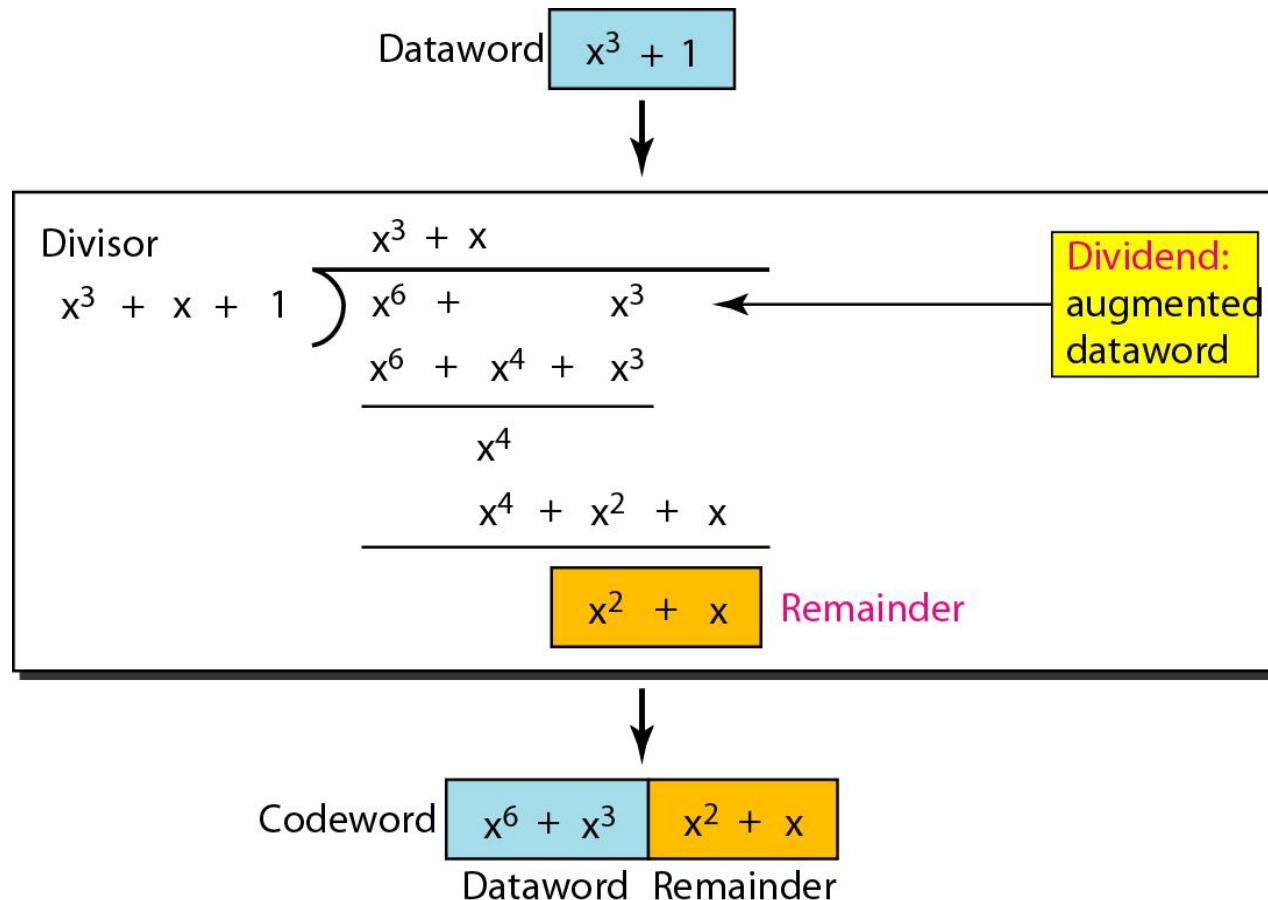
■ 多项式相乘：每项分别相乘，最后删除相同项

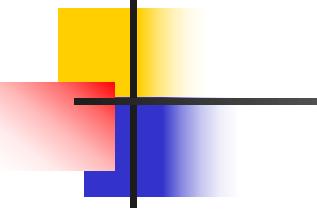
■ 多项式相除：同二进制除法

■ 移位

- 左移 m 位表示在右边加上 m 个0，每项乘以 x^m
- 右移 m 位表示删除右边 m 位，每项除以 x^m
- 没有负幂次

Figure 10.22 使用二项式的CRC除法





Note

循环码的除数通常称为生成多项式，
简称生成子（生成器）。

CRC性能分析

校正子： $s(x)$

如果 $s(x) \neq 0$, 一位或多位被破坏

如果 $s(x) = 0$,

a. 没有位被破坏，或者

b. 有一些位被破坏，但是译码器无法检测到（超出检错能力）。

循环编码分析

数据字: $d(x)$ 码字: $c(x)$ 生成多项式: $g(x)$
校正子: $s(x)$ 差错: $e(x)$

接收到的码字 = $c(x) + e(x)$

$$\frac{\text{接收到的码字}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

循环码中，有些可以被生成多项式 $g(x)$ 整除的差错无法被捕捉到。(发生的概率极小)

单个位差错

- 单个位差错 $e(x) = x^i$
- $e(x)$ 不能被 $g(x)$ 整除

若生成多项式至少有两项，且 x^0 的系数是1，则所有单比特错误都可以检出。

Example 10.15

下面三个生成多项式的纠错能力如何？

- a. $x + 1$
- b. x^3
- c. 1

Solution

- a. 没有一个 x^i 可以被 $x + 1$ 整除，任何单个位差错都可以被捕捉到。
- b. 如果 $i \geq 3$ ，即被破坏的位位于第 4 位或更高位， x^i 可以被 $g(x)$ 整除，余数为 0，接收方认为没有差错。只能捕捉到位 1 到位 3 的单个位差错。
- c. i 的所有值使得 x^i 可以被 $g(x)$ 整除，没有单个位差错可以被捕捉到。 $g(x)$ 是无用的。

两个独立的单个位差错

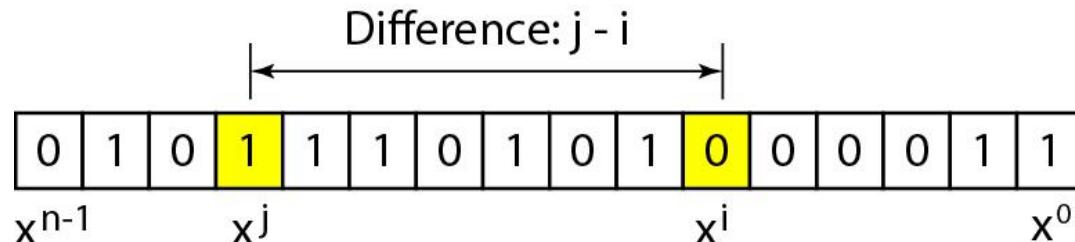


Figure 10.23 使用多项式表示的两个独立的单个位差错

- $e(x) = x^i + x^j = x^i (x^{j-i} + 1) = x^i (x^t + 1)$
- $x^t + 1$ 不能被 $g(x)$ 整除

若生成多项式不能整除 $x^t + 1$ (t 在 1 和 $n-1$ 之间)，那么所有独立的双比特错误都能被检测到。

Example 10.16

下面的生成多项式对两个独立的单比特差错检测能力如何？

- a. $x + 1$
- b. $x^4 + 1$
- c. $x^7 + x^6 + 1$

问题：还能
检测出其它
错误吗？

Solution

- a. 不能检测出任何两个相邻差错能被 $x + 1$ 整除。
- b. 不能检测相隔 4 个位置的两个差错。这两个差错可以位于任何位置，但是如果距离是 4 则无法被检测到。
- c. 可以被检测到。

附：证明

由等比数列求和公式得

$$1 - x + x^2 - x^3 + \cdots - x^{t-2} + x^{t-1}$$

$$= \frac{1 - (-x)^t}{1 - (-x)}$$

t 为奇数时，根据上式得到

$$x^t + 1 = (x + 1)(x^{t-1} - x^{t-2} + \cdots - x + 1)$$

奇数个差错

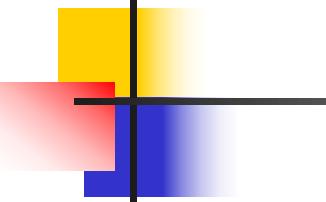
包含 $x+1$ 因子的生成多项式能检测到所有奇数个差错。

例如 $g(x) = x^4 + x^2 + x + 1 = (x + 1)(x^3 + x^2 + 1)$

突发性差错

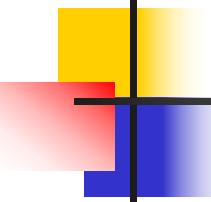
$$e(x) = x^j + \cdots + x^i = x^i(x^{j-i} + \cdots + 1)$$

- 1、如果生成多项式能检测到单个差错，则不能整除 x^i 。
- 2、讨论 $(x^{j-i} + \cdots + 1) / (x^r + \cdots + 1)$ 的余数
 - 如果 $j - i < r$ ，余数永远不会为0。记作 $j - i = L - 1$ ，即 $L < r + 1$ 或 $L \leq r$ 。表示所有长度小于或等于 r 的突发性差错都能被检测到。
 - 某些情况中（少见），如果 $j - i = r$ 或 $L = r + 1$ ，校正子为0，差错无法被检测到。
 - 某些情况中（少见），如果 $j - i > r$ 或 $L > r + 1$ ，校正子为0，差错无法被检测到。



Note

- 所有 $L \leq r$ 的突发性差错均可被检测到。
- 所有 $L = r + 1$ 的突发性差错有 $1 - (1/2)^{r-1}$ 的概率被检测到。
- 所有 $L > r + 1$ 的突发性差错有 $1 - (1/2)^r$ 的概率被检测到。



Example 10.17

生成多项式 $x^6 + 1$ 纠正突发性差错的能力如何？

Solution

可以检测出所有长度小于等于 6 位的突发性差错，
长度为 7 的突发性差错无法被检测到的概率是
3%，长度大于等于 8 的突发性差错无法被检测到
的概率是 0.8%。

高性能生成多项式的特性

1. 至少有两项
2. x^0 的系数一定不为 0
3. 不能整除 $x^t + 1$ ($2 \leq t \leq n - 1$)
4. 应当有因子 $x + 1$

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

Table 10.7 一些常用的标准生成多项式

10-5 CHECKSUM 校验和与反码

用校验和进行简单检错的方法常用于Internet的其它高层协议中。

Topics discussed in this section:

Idea 概念

One's Complement 反码

Internet Checksum 因特网中的校验和

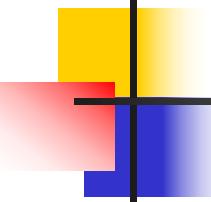
Example 10.18

发送数据时，同时还发送它们的和（用于检错校验，因此称为校验和）。

假设发送 5个 4位数字为 $(7, 11, 12, 0, 6)$ ，发送时发送 $(7, 11, 12, 0, 6, 36)$ ， 36 是 5个数字的和。

也可以发送和的负值（补数） $(7, 11, 12, 0, 6, -36)$ ，这样接收器处理更简单些。

校验和



Example 10.20

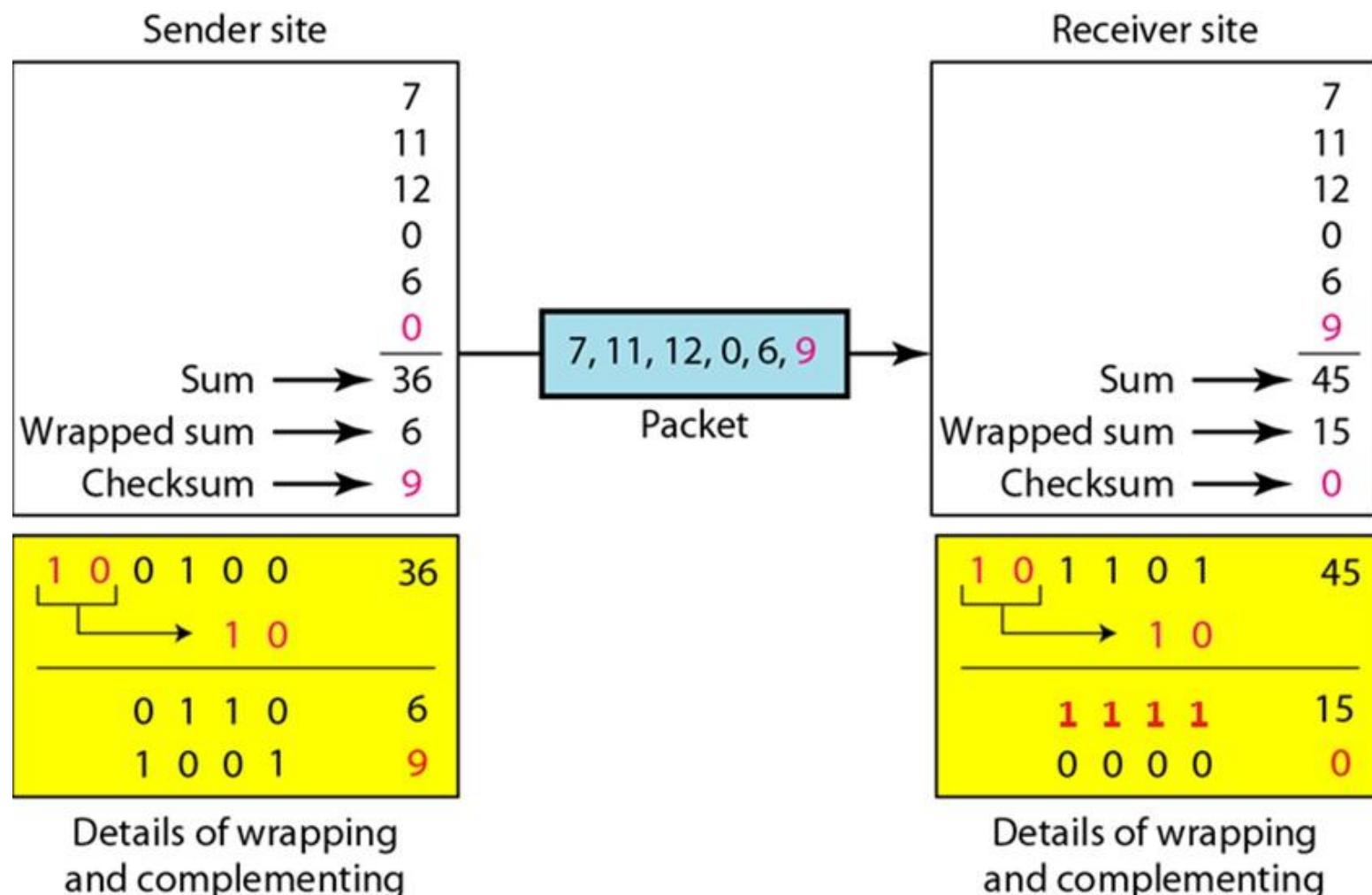
用反码来解决数据段的进位和借位问题。
如何用 4位表示数字 21和 -6?

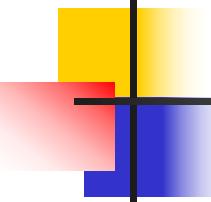
Solution

21的二进制数是 10101， 将最高位 1加到最低位，
得到 $(0101 + 1) = 0110$ 或 6。

6的二进制数是 0110， -6对其取反， 得到 1001。也
可以用 6的补数 $(2^4 - 1) - 6 = 9$ ， 9的二进制数是
1001。

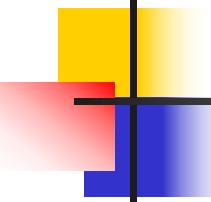
Figure 10.24 Example 10.22 分段、进位和反码等相关的操作





因特网校验和的步骤：(发送方)

1. 报文被划分为16-bit 字。
2. 校验和字的初始值设为 0。
3. 所有字包括校验和使用反码运算相加。
4. 对累加和求反码变成校验和。
5. 校验和随数据一起发送。



因特网校验和的步骤：(接收方)

1. 报文（包括校验和）被划分成16位字。
 2. 使用反码运算将所有字相加。
 3. 对该和求反码生成新校验和。
 4. 如果校验和是0，接收报文，否则丢弃。
-

Figure 10.25 Example 10.23 分段、进位和反码等相关的操作

1 0 1 3	Carries
4 6 6 F	(Fo)
7 2 6 F	(ro)
7 5 7 A	(uz)
6 1 6 E	(an)
0 0 0 0	Checksum (initial)
8 F C 6	Sum (partial)
8 F C 7	Sum
7 0 3 8	Checksum (to send)

a. Checksum at the sender site

1 0 1 3	Carries
4 6 6 F	(Fo)
7 2 6 F	(ro)
7 5 7 A	(uz)
6 1 6 E	(an)
7 0 3 8	Checksum (received)
F F F E	Sum (partial)
F F F F	Sum
0 0 0 0	Checksum (new)

a. Checksum at the receiver site

作业

P199

15, 16, 23, 26, 30, 32(a)(b)