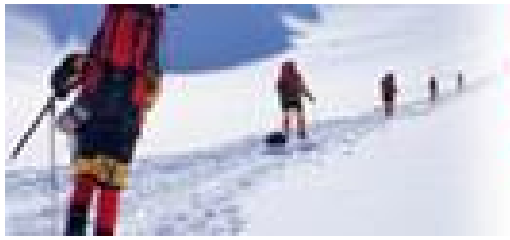


Data Communications and Networking Fourth Edition

Forouzan

第10章

检错与纠错

第三部分 数据链路层

- 数据链路层把物理层的原始传输设施转换成一条负责点到点（逐跳）通信的链路；
- 数据链路层具体任务：
  - 成帧
  - 寻址
  - 流量控制
  - 差错控制
  - 介质访问控制
- 局域网和广域网实现（注意：局域网/广域网是数据链路层的概念）

本章主要内容

- 引言
- 块编码
- 线性块编码
- 循环编码
- 校验和

数据在传输过程中可能遭到破坏，一些应用需要进行检错和纠错。

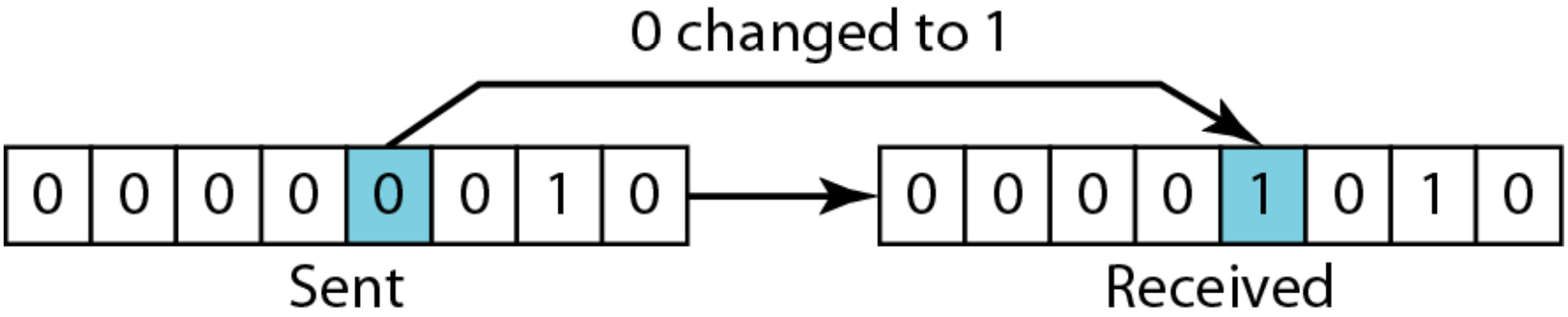
10-1 引言

- 由于干扰，位流从一点流动到另一点时，可能受到不可预测的变化，这些干扰可能会改变信号的波形；
- 差错类型：
  - 单个位（单比特）差错（single-bit error）
  - 突发性差错（burst error）

在单比特差错中，数据单元（如一个字节、字符或分组）中仅有一比特发生变化。

图10.1 单个位差错

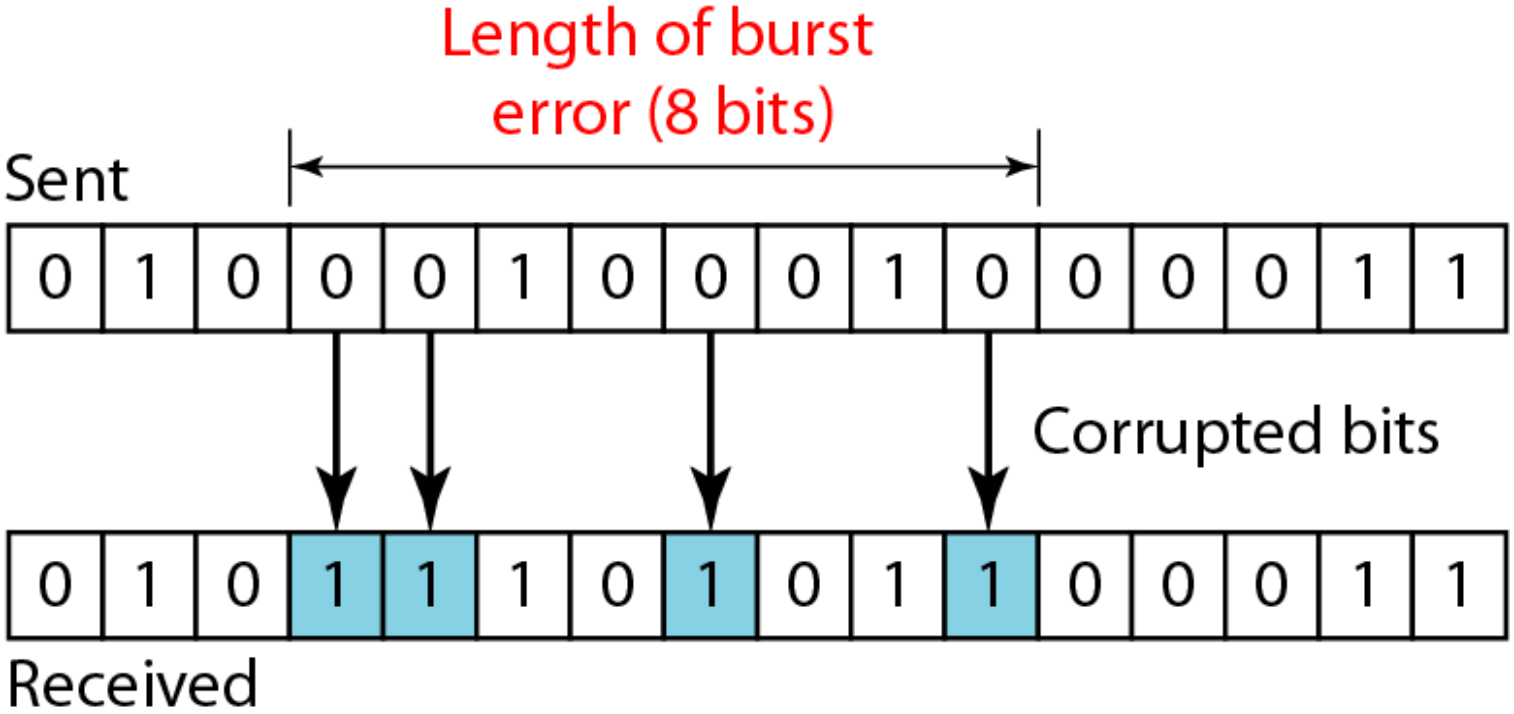
p 串行传输中，单个位差错很少出现，因为单个比特持续的时间很短（100Mbps速率1比特持续时间？），通常远小于噪声信号的时长



一个突发性差错意味着数据单元中两位或多位发生变化。

图10.2 长度为8位的突发性差错

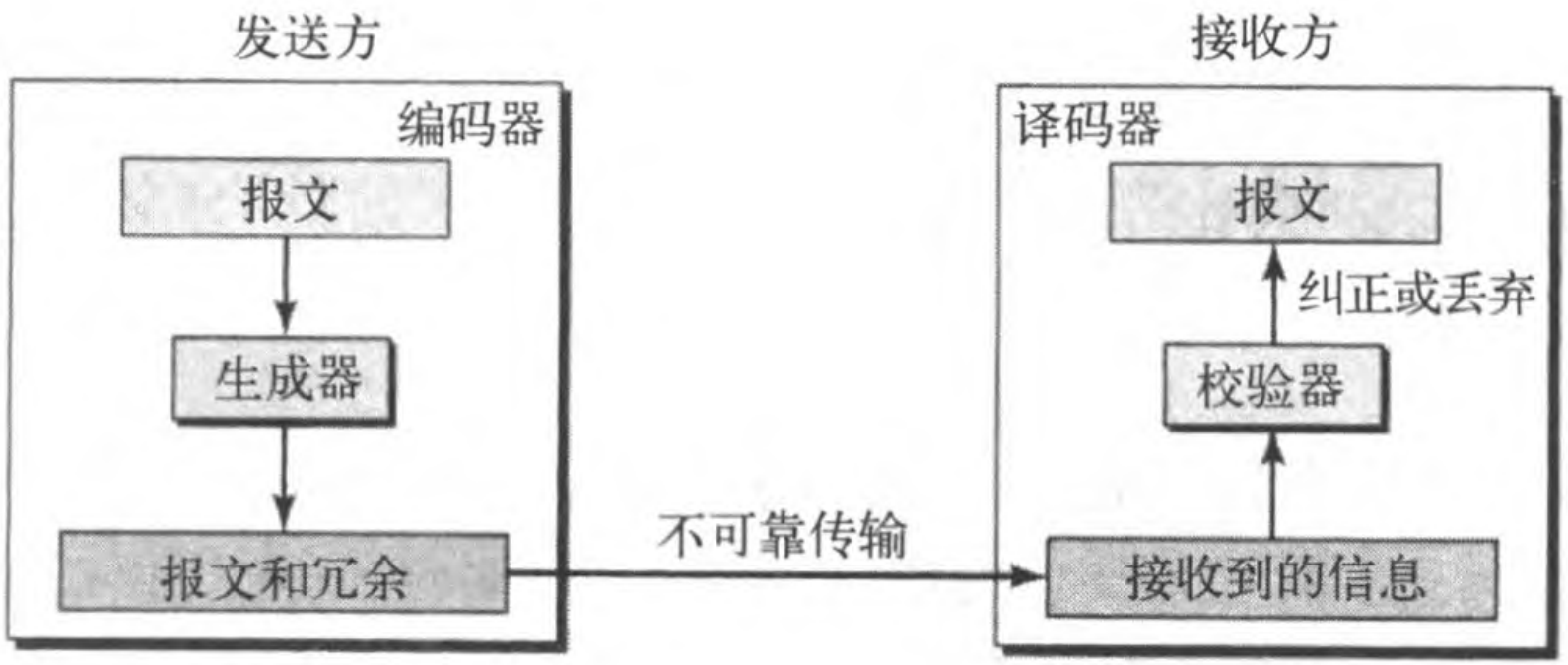
p 突发性差错更容易发生，因为噪声信号的持续时间通常大于一个比特的持续时间，意味着会影响多个位（并不是所有这些位都发生改变）



检错或纠错的核心概念是冗余。
为了检测或纠正错误，我们需要发送除了数据外的额外（冗余）位。

- 纠错比检错更难；
- 检错只看是否发生错误，而纠错需要知道被破坏的比特的个数，还要准确定位它们的位置（例如：8位数据单元发生了单比特或两个比特的错误，情形不一样）；
- 两种主要的纠错方法：
  - 前向纠错：接收方通过使用冗余位尝试推测报文的方法（实现起来较复杂）。如果差错个数少，这是可能的；
  - 重传纠错：接收方检测出有差错则要求发送方重新发送报文的技术

图10.3 编码器和译码器的结构（通过编码实现冗余）



本课程介绍块编码，卷积编码超出了课程范围。

模运算：在模N运算中，只使用0到N-1的整数。

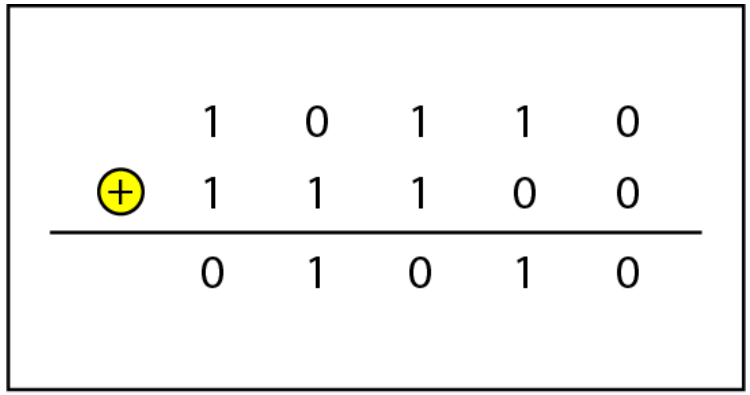
如果数字大于N，则除以N并取余数；如果是负数，需要加上N的倍数使其变成正整数

图10.4 模2运算（没有进位与借位）和异或运算关系

- 加：0+0=0 0+1=1 1+0=1 1+1=0
- 减：0-0=0 0-1=1 1-0=1 1-1=0

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

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

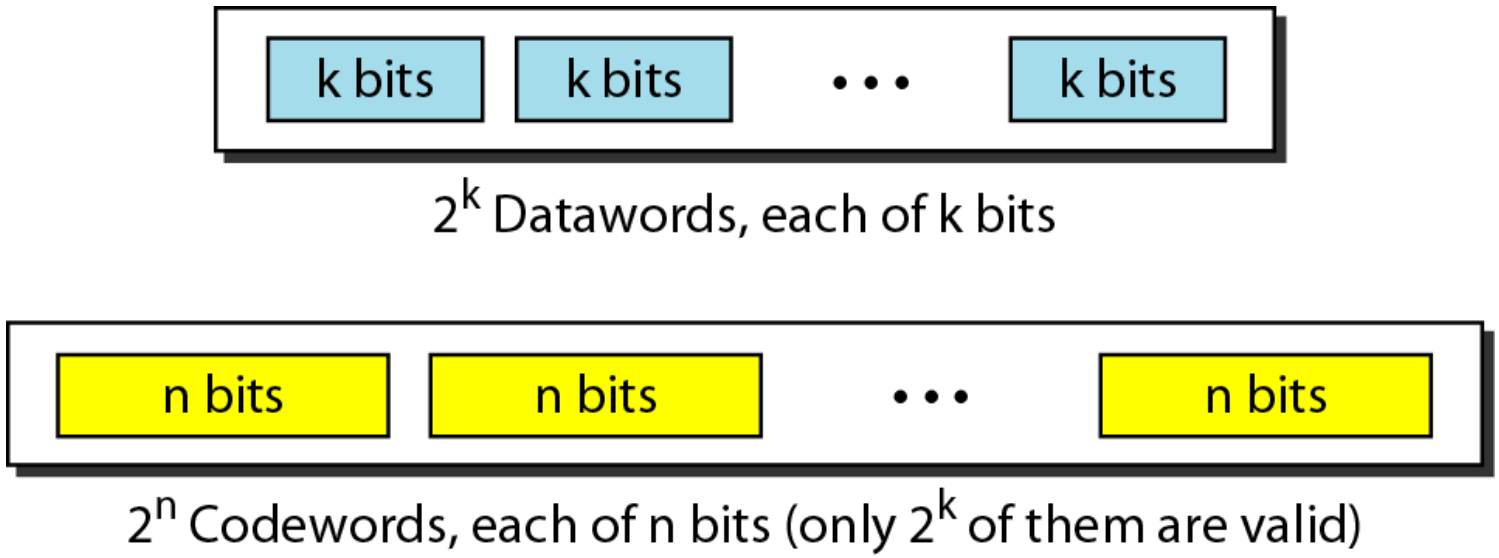


c. Result of XORing two patterns

10-2 块编码

- 在块编码中，把报文划分成块，每块k位，称为数据字（dataword），并增加r个冗余位使其长度变为n=k+r，形成n位块，称为码字（codeword）；
- 后续介绍如何选择或计算r个冗余位；
- 块编码是一对一，相同的数据字总是编码成相同的码字

图10.5 数据字和码字（2^n-2^k个无效或其他目的）

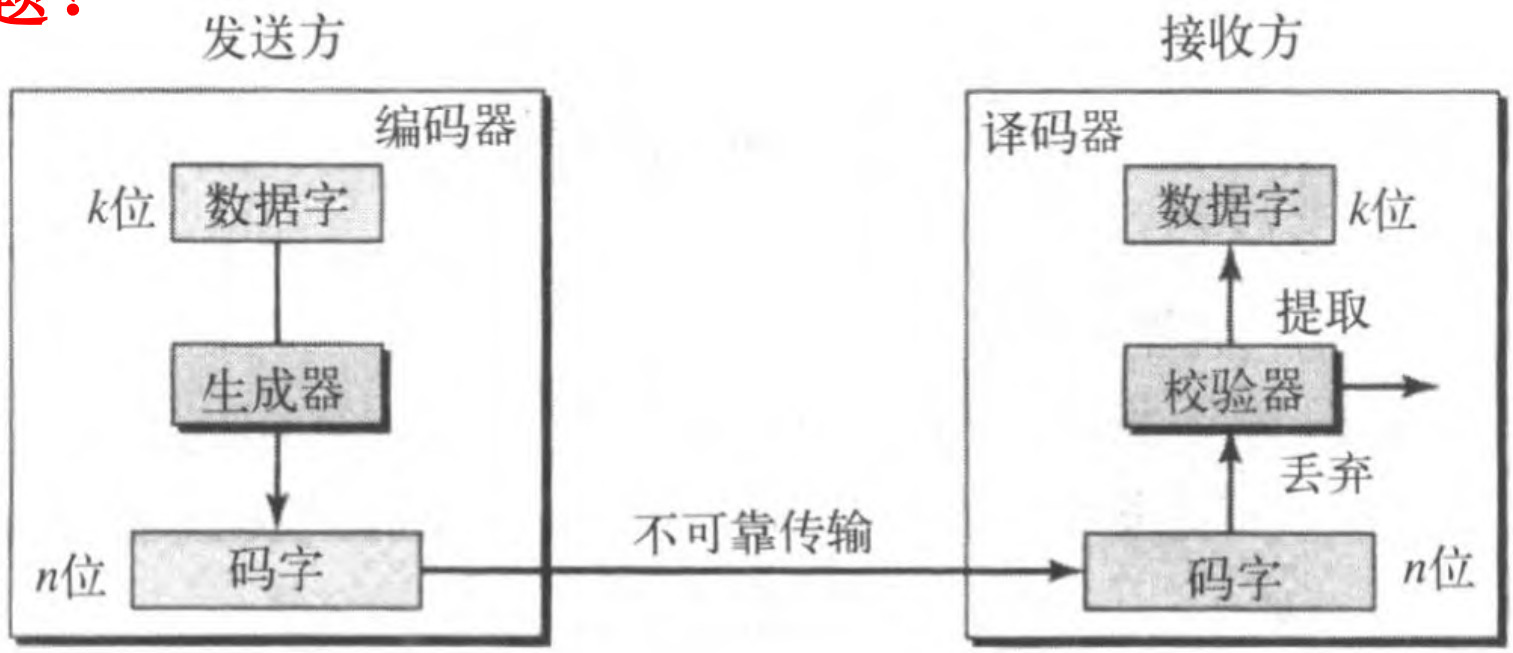


例10.1

第四章讨论的4B/5B码就是这种编码的一个好例子。我们可以看到，由于k=4，n=5，则数据字有2^k=16个，码字有2^n=32个。我们从32个码字里精心挑选16个码字用于报文传输，其余16个用于其它目的或不用。

图10.6 块编码的差错检测过程

- 满足两个条件可以检测出一个差错：
  - ø接收方有有效码字的列表；
  - ø原来码字已改变成无效的码字
- 问题？



例10.2

假设k=2，n=3。表10.1给出了数据字和码字列表。
假设发送方把数据字01编码成011并发送给接收方，考虑以下的情况：
1. 接收方接收到011，它是有效的码字，接收方从它提取出数据字01。
2. 码字在传输中被破坏，接收方接收到111（最左边的位被破坏），这是无效的码字，被丢弃。
3. 码字在传输中被破坏，接收方接收到000（右边两位被破坏），这是个有效码字，接收方错误地提取出数据字00。这两个破坏位形成的差错无法被检测到。

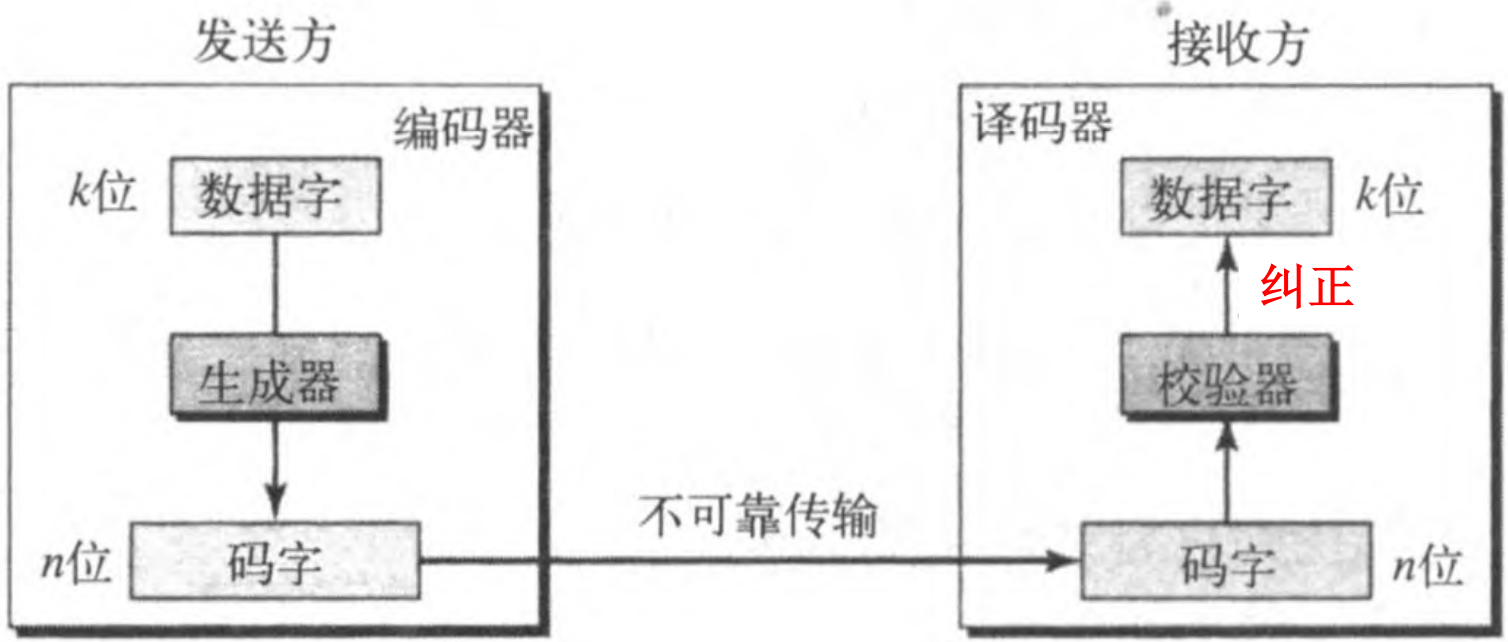
表10.1 差错检测码（例10.2）

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

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（或情形）的差错，其它类型（或情形）的差错无法被检测到。

图10.7 纠错码的编码器和译码器（P178）的结构

p纠错更复杂，接收方需要推测出发送的原码字，通常比检错需要更多的冗余位



例10.3

给例10.2加入更多的冗余位，看接收方在不知道实际发送的码字的情况下是否能纠正差错。我们给2位数据字加入3个冗余位，形成5位码字，表10.2给出了数据字和码字。

假设数据字是01，发送方参照该表（或使用算法）生成码字01011。这个码字在传输中被破坏，接收方接收到01001（右边第2位发生差错）。首先，接收方发现接收到的码字不在表中，这意味着发生了差错（必须在纠错前进行检错）。假定只有1位被破坏，接收方使用以下策略来推测正确的码字。

例10.3（cont）

1. 用表中第一个码字与接收到的码字进行比较（01001和00000），因为有两位不同所以接收方确定第一个码字不是发送的那个码字。
2. 相同的原因可知表中的第三个和第四个码字不可能是原来的码字。
3. 原来码字一定是表中的第二个码字，因为与接收到的码字只有一位不同。接收方用01011代替01001，并参照表找到原来数据字01 。

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

差错控制编码的一个核心概念是汉明距离，两个字的汉明距离是对应位不同的数量。对两个字进行异或操作并计算1的个数，就可以很容易得出汉明距离。

例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：

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

用于设计编码的度量是最小汉明距离，最小汉明距离是一组字中所有可能对的汉明距离的最小值。

例10.5

求表10.1编码方案的最小汉明距离。

解：

我们先求出所有的汉明距离。

$d(000, 011) = 2$	$d(000, 101) = 2$	$d(000, 110) = 2$	$d(011, 101) = 2$
$d(011, 110) = 2$	$d(101, 110) = 2$		

得到d_{min}为2。

例10.6

求表10.2编码方案的最小汉明距离。

解：

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

得到d_{min}是3。

p假设s个差错发生在传输中，发送的码字和接收到的码字间的汉明距离是d：

- Ø如果我们的编码能检测出最多s个差错，那么两个有效编码间的最小汉明距离必须是s+1，这样接收到的码字才不会与有效码字匹配；换言之，如果所有有效码字间的最小距离是s+1，那么接收到的码字不会被错误地认为是另一个正确的码字；
- Ø距离小于s+1的码字接收方不会认为是有效码字，差错可以被检测到。

编码方案三个参数：码字长度n，数据字长度k，以及最小汉明距离d_{min}。

为了保证检测出所有情况下最多s个错误，块编码中最小汉明距离一定是$d_{min}=s+1$。

例10.7

我们第一个编码方案（表10.1）的最小汉明距离是2，这个编码方案保证检测到单个差错。
例如，如果发送第三个码字101，发生了一个差错，那么接收到的码字就不能与任何一个有效码字匹配。但是如果发生了两个差错，接收到的码字可能与某一个有效码字匹配，因此差错无法被检测到。

例10.8

第二个编码方案（表10.2）有d_{min}=3，这个编码保证能检测到最多2个差错。可以看到，当发送任一个有效码字时，发生两个差错得到的码字不在有效码字表中，接收方不会被欺骗。但是，三个差错的一些组合会把一个有效码字改变成另一个有效码字，接收方会接受接收到的码字，而无法检测到差错。

图10.8 得出差错检测中d_{min}的几何意义

p 只有$d_{min}>s$，才能使得其他有效码字一定位于圆外，从而推导出落在圆内的都是无效的码字

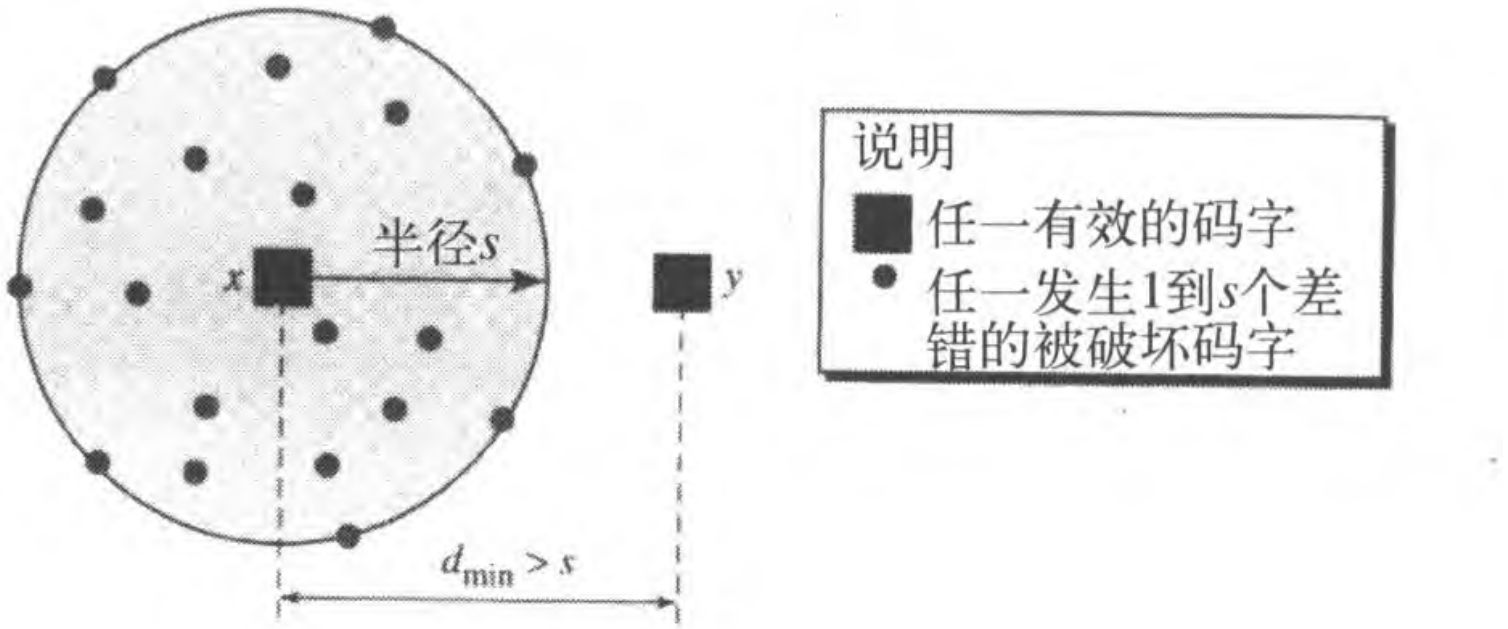
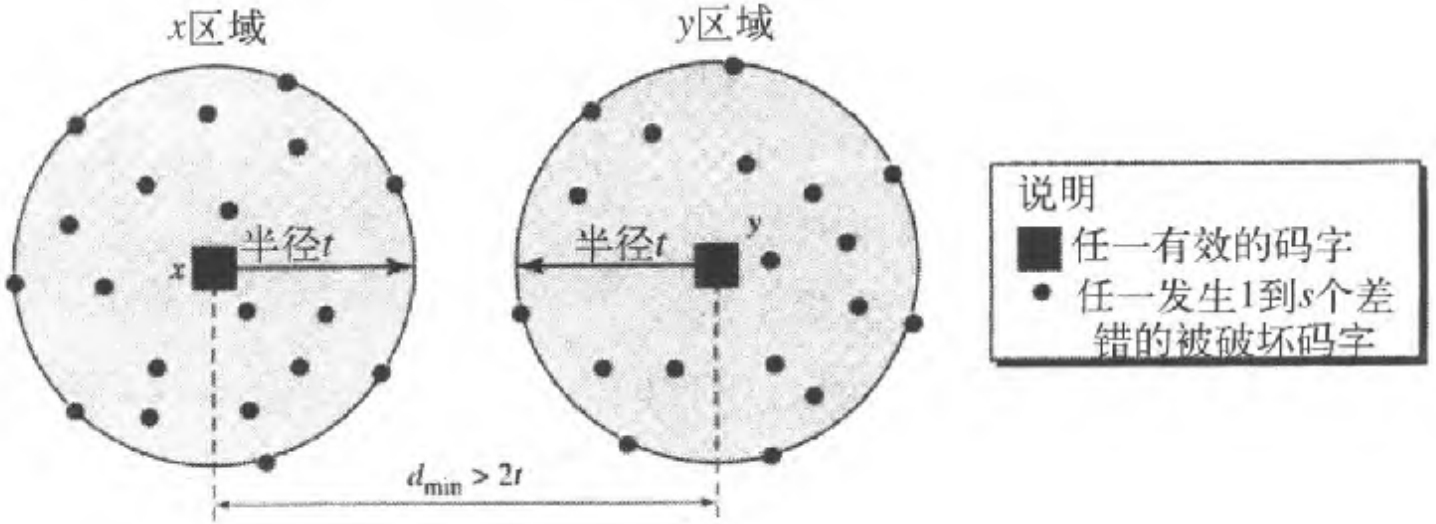


图10.9 差错纠正中d_{min}的几何意义

- 假设每个有效码字都有以t为半径的圆形区域并且该有效码字在圆心。例如，假设码字x被破坏的位小于或等于t位，那么这个被破坏码字位于圆内或圆周上；
- 如果接收方接收到属于该区域的一个码字，就确定原来的码字就是位于圆心的码字；
- 注意：假设最多只能发生t个差错，否则决策是错误的



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

例10.9

汉明距离为4的编码方案，检错和纠错能力分别是多少？

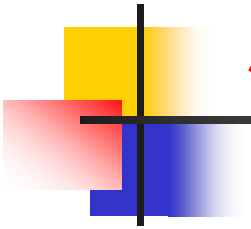
解：

这个方案保证检测到最多3个差错（s=3），但是它能纠正1个差错。换言之，如果这个编码用于纠错，它的部分能力被浪费了。纠错编码需要的最小距离是奇数（3，5，7，...）。

10-3 线性块编码

- 当前，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（即模二加法）生成另一个有效码字（线性块编码的正式定义需要抽象代数理论，超出了本书范围-数学是基础，很重要）



例10.10

表10.1和10.2都属于线性块编码。

例10.11

线性块编码最小汉明距离求法：是具有最少1的个数的非0有效码字中1的个数。

在第一个编码方案（表10.1）中，非零码字的1的个数是2、2和2，因此，最小汉明距离是$d_{min}=2$。
在第二个编码方案（表10.2）中，非零码字的1的个数是3、3和4，因此$d_{min}= 3$。

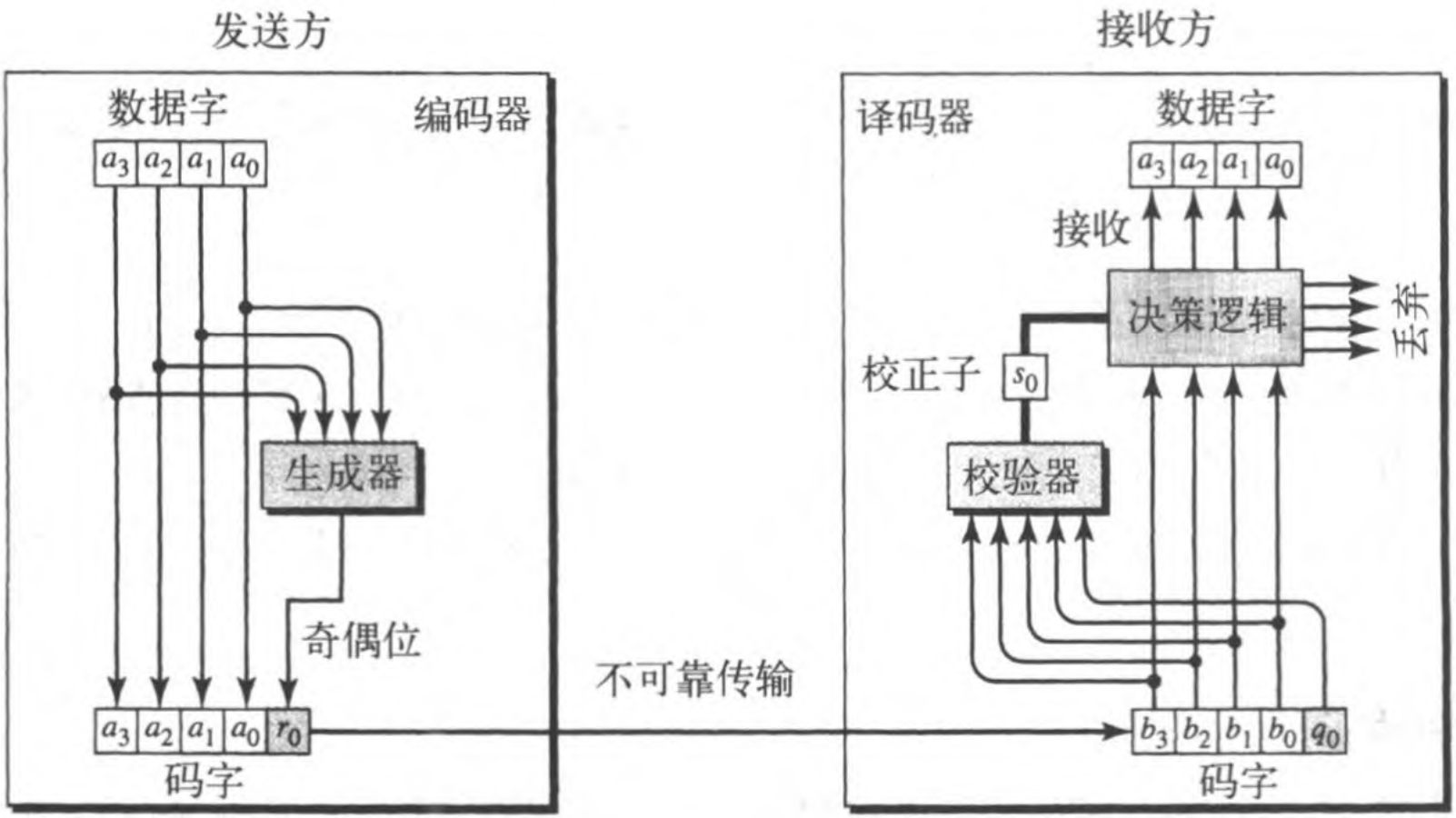
常见线性块编码之简单奇偶校验编码

讨论1的个数为偶数的情形：简单的奇偶校验码是$n=k+1$，且$d_{min}=2$（为什么？）的单比特检错码。

表10.3 简单的奇偶校验码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

图10.10 简单奇偶校验编码的编码器和译码器 （模二加运算，$S_0=0$）



例10.12

假设发送方发送数据字1011，从这个数据字生成的码字是10111，它发送给接收方。我们检查五种情况：

1. 没有差错发生，接收到的码字是10111。校正子是0，生成数据字1011。

2. 一个单个位差错改变了a_1，接收到的码字是10011。校正子是1，没有数据字生成。

3. 一个单个位差错改变了r_0，接收到的码字是10110。校正子是1，没有数据字生成。注意：虽然没有数据字位被破坏，但是因为编码不够复杂不能说明破坏位的位置，因此仍然没有数据字生成。

例10.12（cont）

4. 一个差错改变了r_0，第二个差错改变了a_3，接收到的码字是00110。校正子是0，在接收方生成数据字00110。注意：这里因为校正子生成错误的数据字，简单奇偶校验译码器不能检测出偶数个差错，差错互相抵消使得校正子为0。

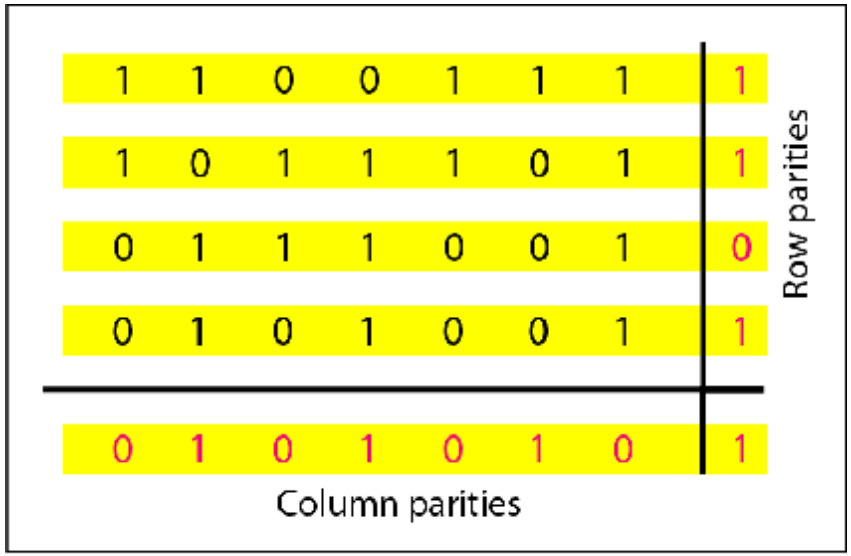
5. 三个位a_3、a_2和a_1被差错改变，接收到的码字是01011。校正子是1，没有数据字生成。这说明简单奇偶校验除了保证检测出一个单个位差错外，还能发现任何奇数个差错。

简单奇偶校验码能检测出奇数个差错。

Q：这与教材P180第四行的结论矛盾吗？

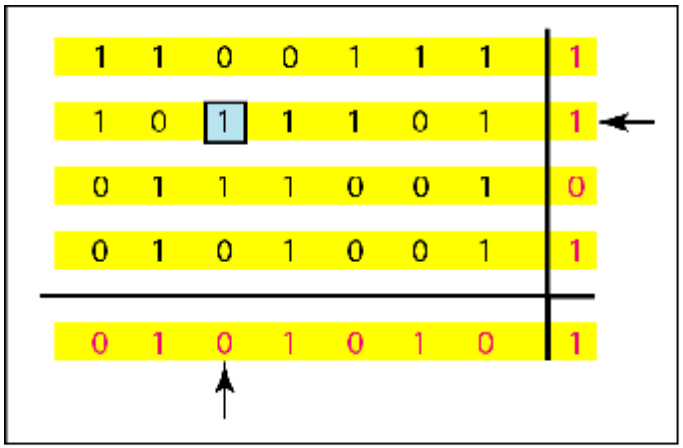
图10.11 两维奇偶校验编码

- 数据字以表格形式组织；
- 对于每一行和每一列，计算出一个奇偶校验位，然后将整个表格发送给接收方，接收方将分别得到每一行和每一列的校正子

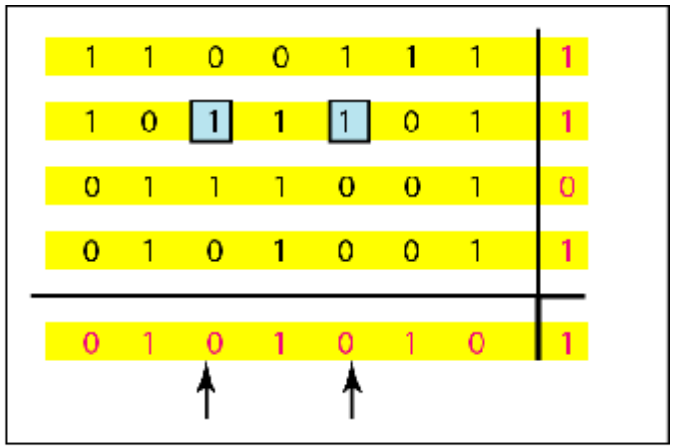


a. Design of row and column parities

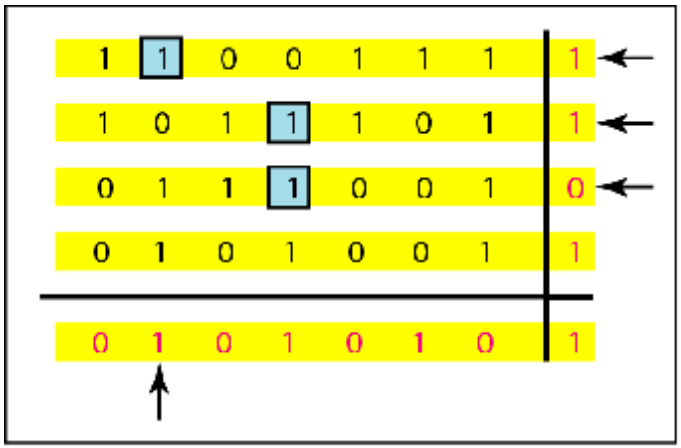
图10.11 两维奇偶校验编码（cont）-P183书上论述欠准确



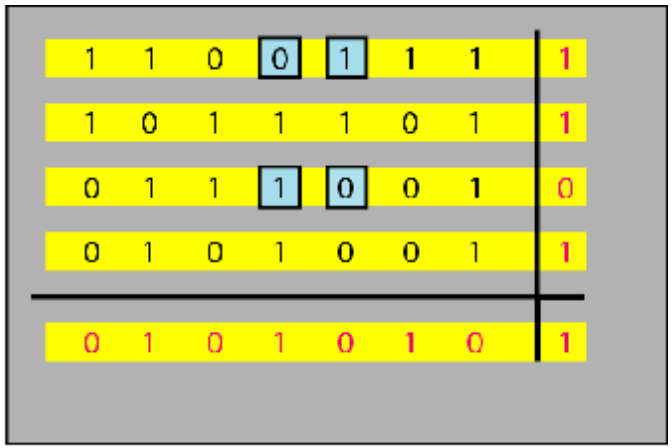
b. One error affects two parities



c. Two errors affect two parities



d. Three errors affect four parities



e. Four errors cannot be detected

汉明编码纠错码

汉明编码最初设计是$d_{min}=3$，它保证检测出最多2位差错和纠正最多1位差错。

推广之，最小汉明距离m（m>=3）、码字长n、数据字长k、校验位个数r的关系为$n=2^m-1$，k=n-m且r=n-k=m。

本书仅讨论$d_{min}=3$的汉明编码，$n=2^m-1$。

表10.4 汉明编码C(7, 4) - m=3，n=7，k=4

Datawords	*Codewords*	*Datawords*	*Codewords*
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

图10.12 汉明码的编码器和解码器结构（冗余位也是模2运算，P184）

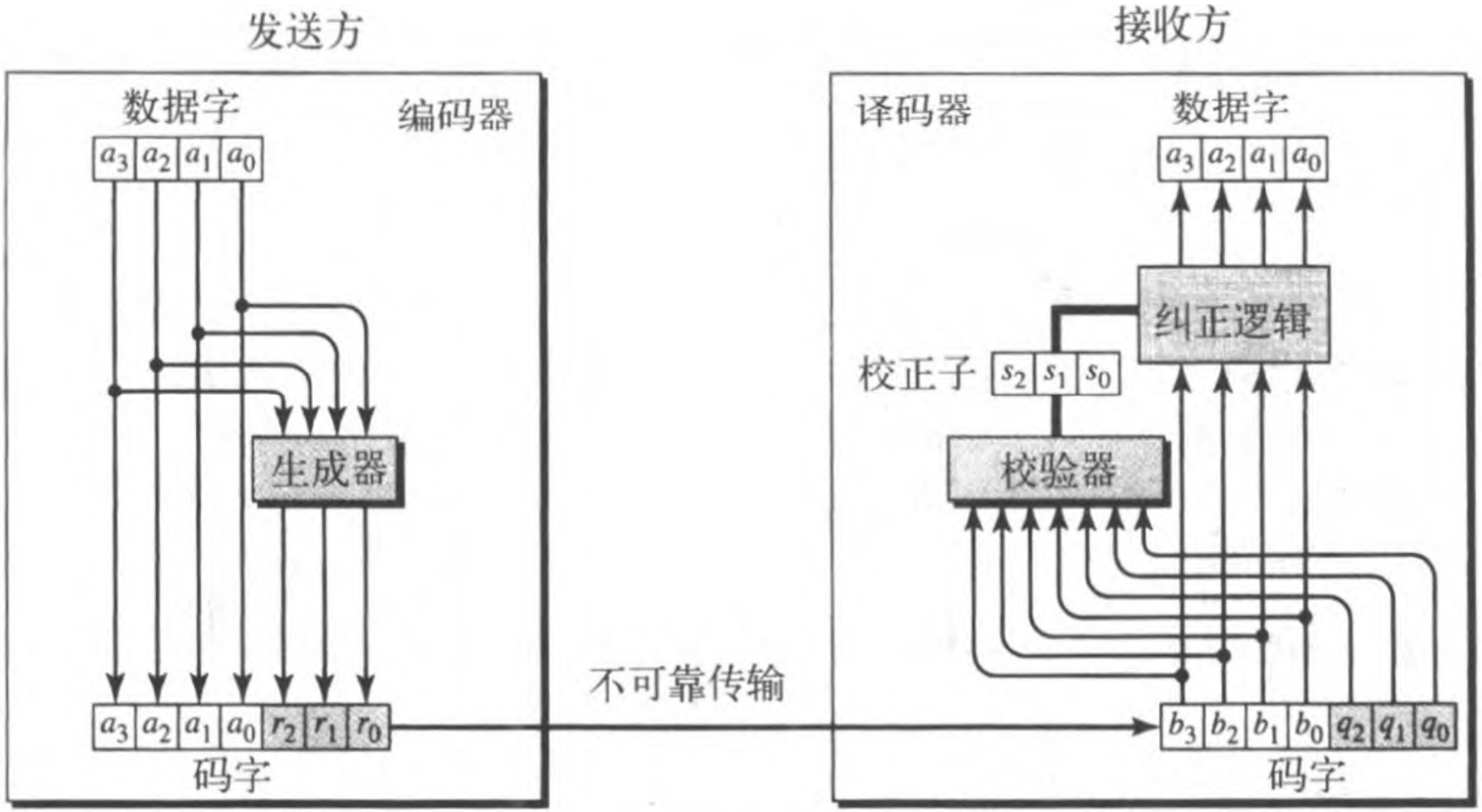


表10.5 由译码器的纠错逻辑分析器做出的逻辑判定

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

p表10.5情况定义了接收到的码字中无差错或者7位中有一位有差错；

p注意：1. 如果在传输中发生两个差错，生成的数据字不是正确的数据字（可检错）；2. 如果想用上面的编码用于纠错，还需要独特的设计。

例10.13

让我们跟踪3个数据字从发送方到目的地的路径。

1. 数据字0100变成码字0100011，接收到码字0100011，校正子是000，无差错，最后的数据字是0100。
2. 数据字0111变成码字0111001，接收到码字0011001，校正子是011，根据表10.5，b_2发生差错，反转b_2后（0变成1），最后的数据字是0111。
3. 数据字1101变成码字1101000，接收到码字0001000（两个差错），校正子是101，这意味着b_0发生差错，反转b_0后，我们得到0000，这是错误的数据字。这说明我们的编码不能纠正（P185，但能检测到）两个差错。

例10.14

我们需要至少7位的数据字，计算满足这个条件的k值和n值。

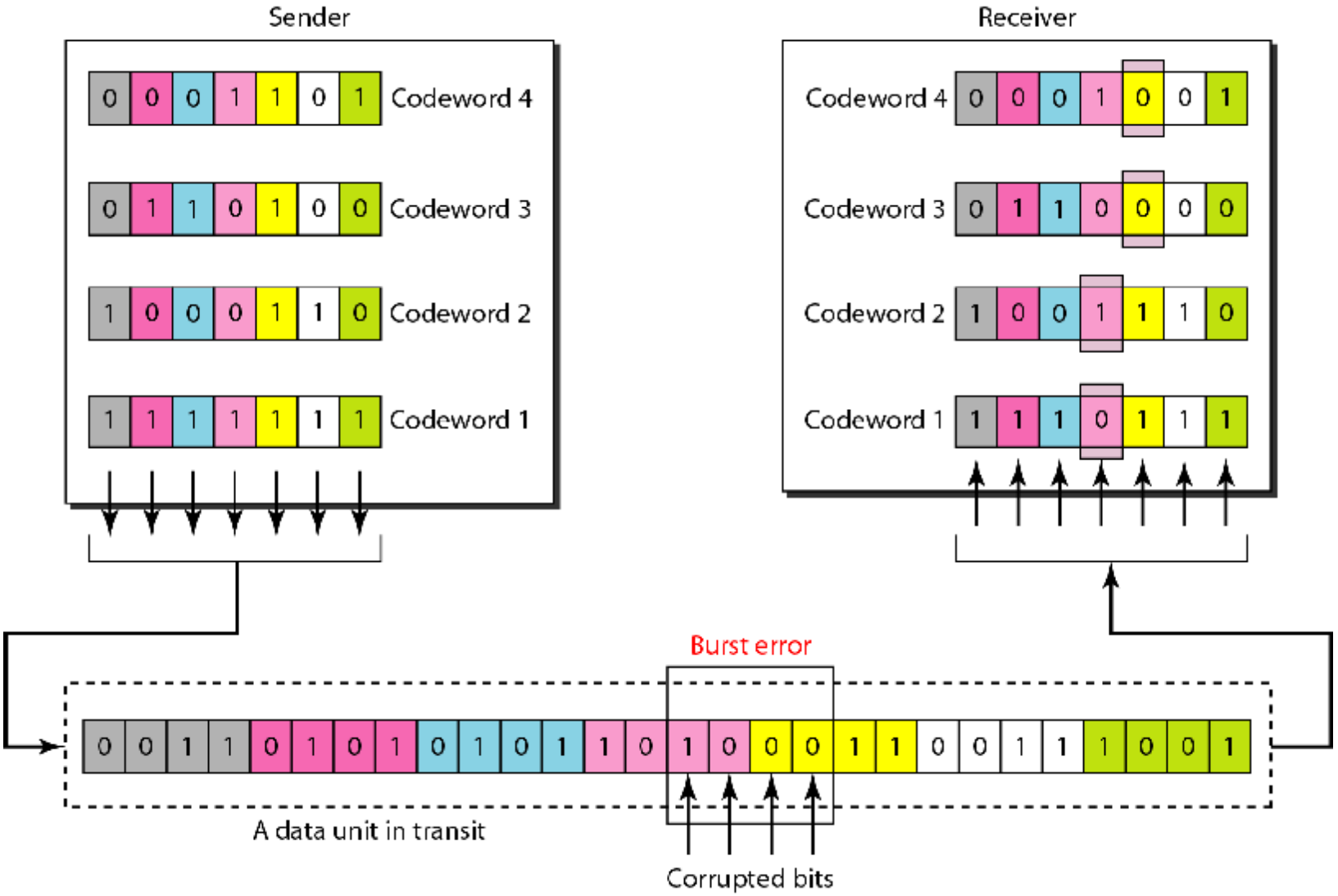
解：

我们要求k=n-m大于或等于7，或者2^m-1-m>=7。

1. 如果我们设m=3，结果是n=2^3-1=7，k=7-3=4，这是不能接受的。

2. 如果我们设m=4，那么n=2^4-1=15，k=15-4=11，这满足了条件，因此可以选择编码C(l5, 11)。

图10.13 使用交织编码技术的汉明码，提高突发错误纠正能力



10-4 循环编码

- 循环编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，结果还是另一个码字；
- 循环冗余码CRC：cyclic redundancy check

表10.6 C(7, 4)循环冗余校验码简称CRC编码

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

图10.14 CRC编码器和译码器的一种可能设计

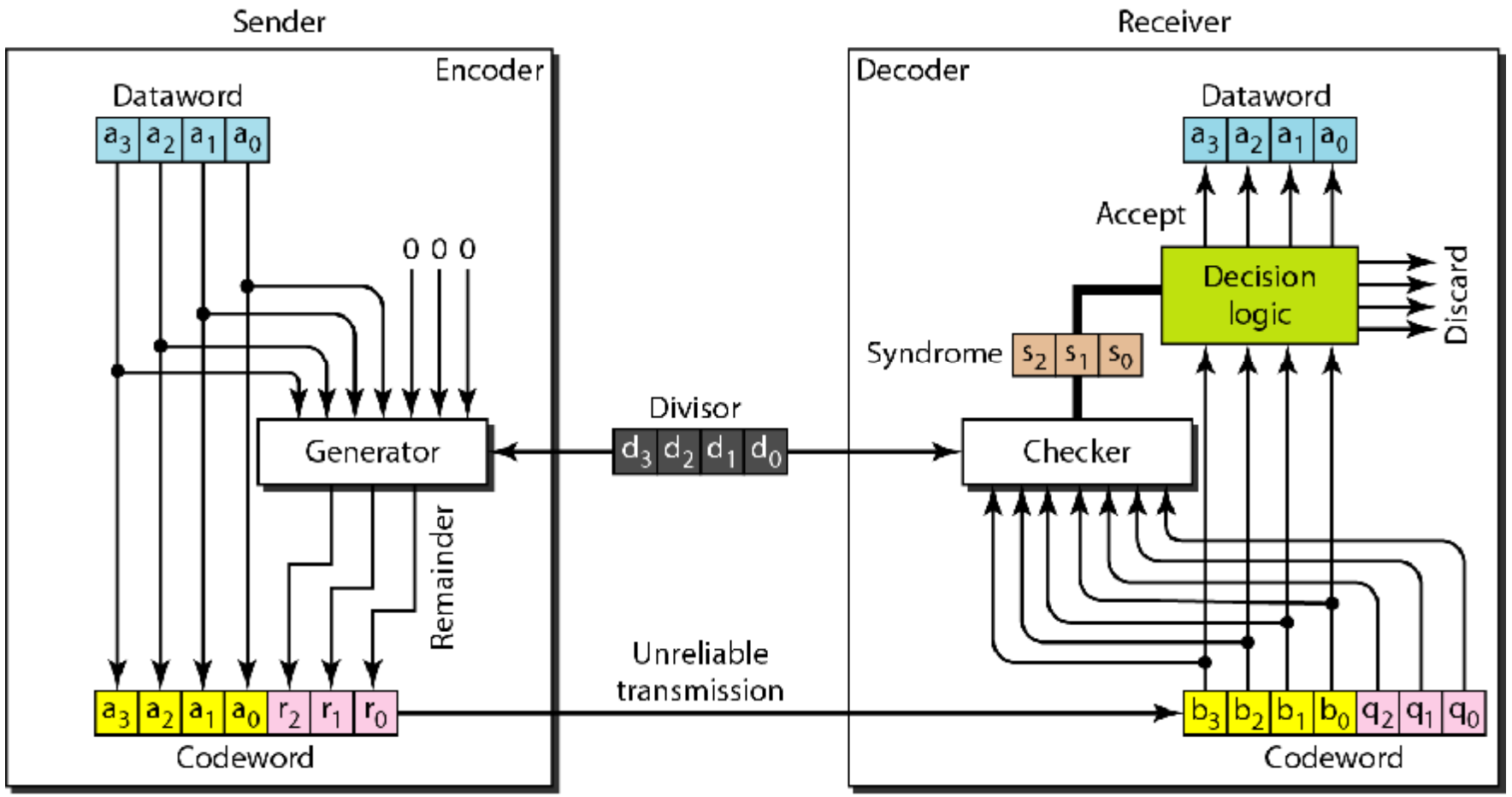


图10.14 CRC编码器和译码器的一种可能设计（cont.）

- 在编码器中，数据字有k位，码字有n位；在数据字的右边加上n-k个位0 ；n位结果传给生成器，生成器使用长度为n-k+1的除数（这个除数是预定义并经双方同意的）；生成器用除数除增加后的数据字（模2除法），除法的商被丢弃，余数（如$r_2r_1r_0$）加到数据字上生成码字；
- 译码器接收到可能被破坏的码字；全n位的副本传递给校验器，校验器产生的余数是n-k位的校正子，它被传递给决策逻辑分析器；如果校正子全0，码字最左边的k位被接收为数据字（被认为无差错）；否则，这k位被丢弃（有差错）

图10.15 CRC编码器中的除法（模2运算）

- 编码器将得到的数据字加上n-k个0，然后它被除数去除；
- 在每一步，除数的副本与被除数的4位进行异或操作，异或运算的结果（余数）是3位，1个额外位移下来加上去组成4位进行下一步；
- 注意：如果被除数最左边的位是0，这一步就不能使用正常的除数，需要使用全0的除数；
- 当没有可移下来的位时，就得到了结果，3位余数形成了校验位（$r_2r_1r_0$），它们附加在数据字上生成码字

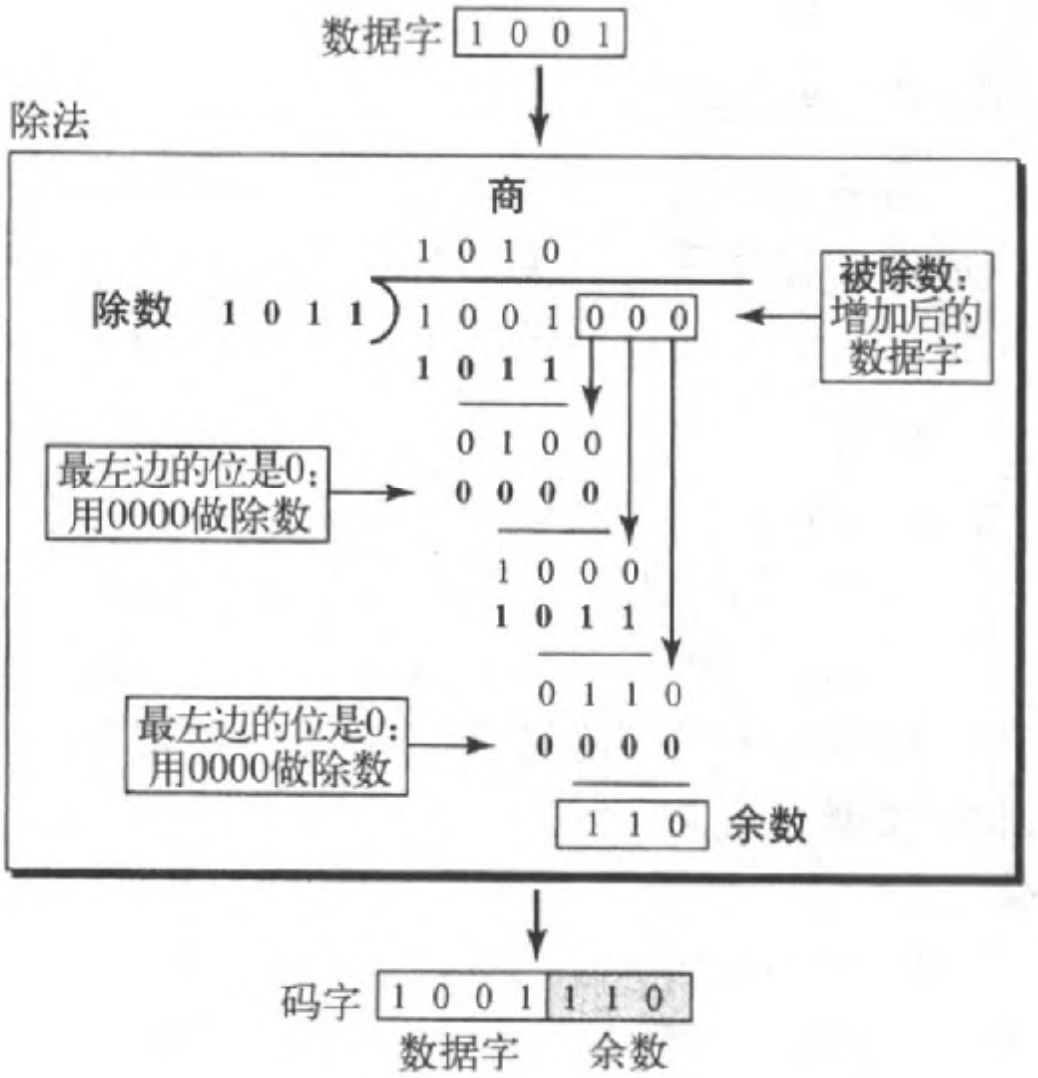
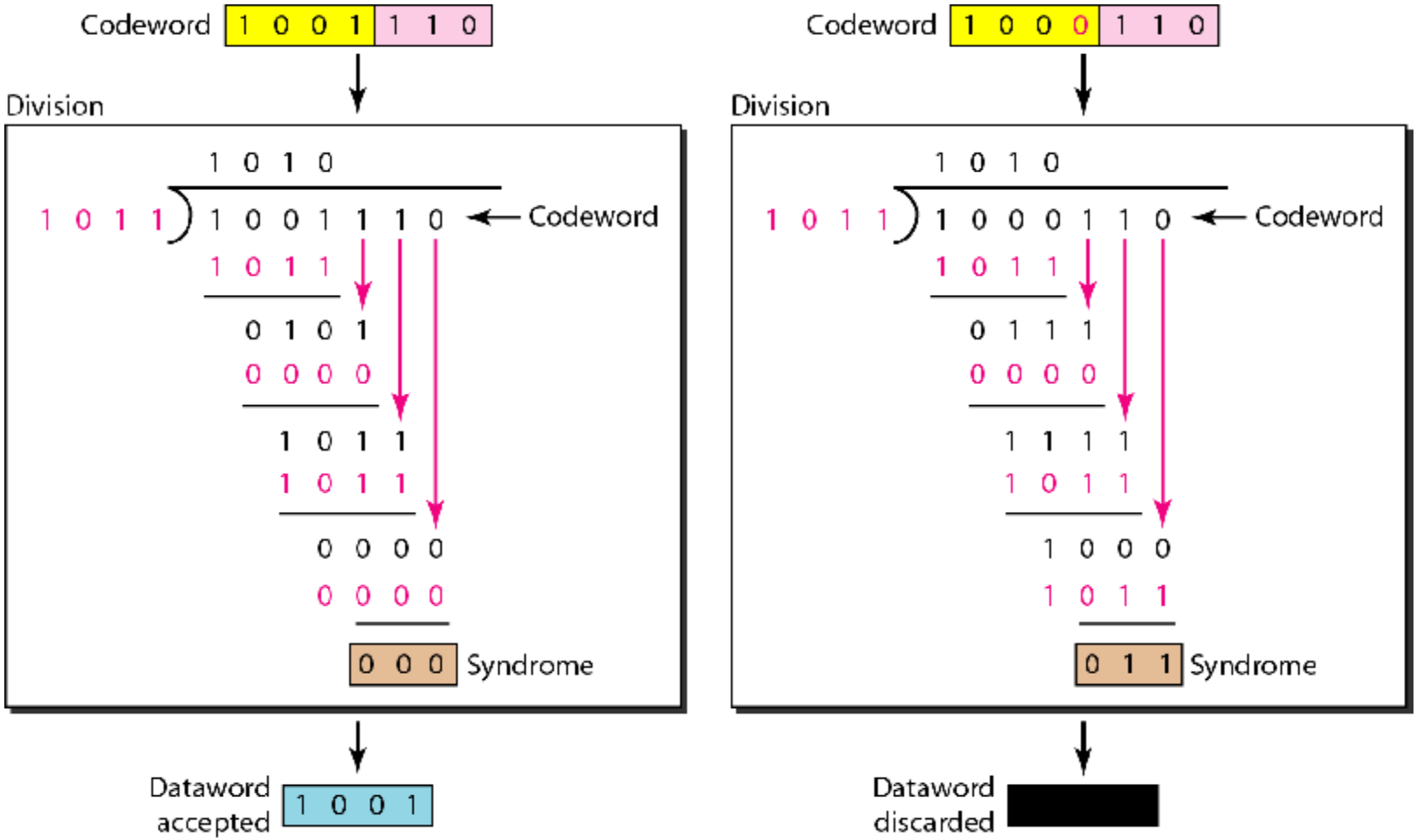


图10.16 两种情形下CRC译码器中的除法（除尽-没有差错、除不尽-被丢弃）



试证明CRC码字能够被预定的除数整除？

- 定义：
  - T=要发送的n比特帧
  - D=k比特数据块，或者报文，就是T中的前k个比特
  - F=(n-k)比特帧校验序列（FCS，即冗余位），就是T中的后(n-k)个比特
  - P=(n-k+1)比特的特定比特序列，即预定的除数
- 我们希望T/P没有余数。

证明：

由于$T = 2^{n-k}D+F$，假设$2^{n-k}D/P = Q+R/P$（Q为商而R为余数），故：

$$T = 2^{n-k}D+R$$（实际上F就是R）

于是：

$$T/P = (2^{n-k}D+R)/ P= 2^{n-k}D/P+R/P = Q+R/P+R/P$$

因为是模2运算，任何数与自己相加都是0，所以：

$$T/P = Q+(R+R)/P = Q$$

从而得知T能被P整除。

练习题

p给定报文D=1010001101，预定比特序列（即除数，或生成码）P=110101，请利用CRC计算帧校验序列R（CRC位，即余数），并给出真正的发送序列T。（10分钟时间）

练习题答案

p由于P是6比特，故余数部分为5比特，将 D*2^5 得到101000110100000，再除以P，得到余数R=01110，发送序列T=101000110101110

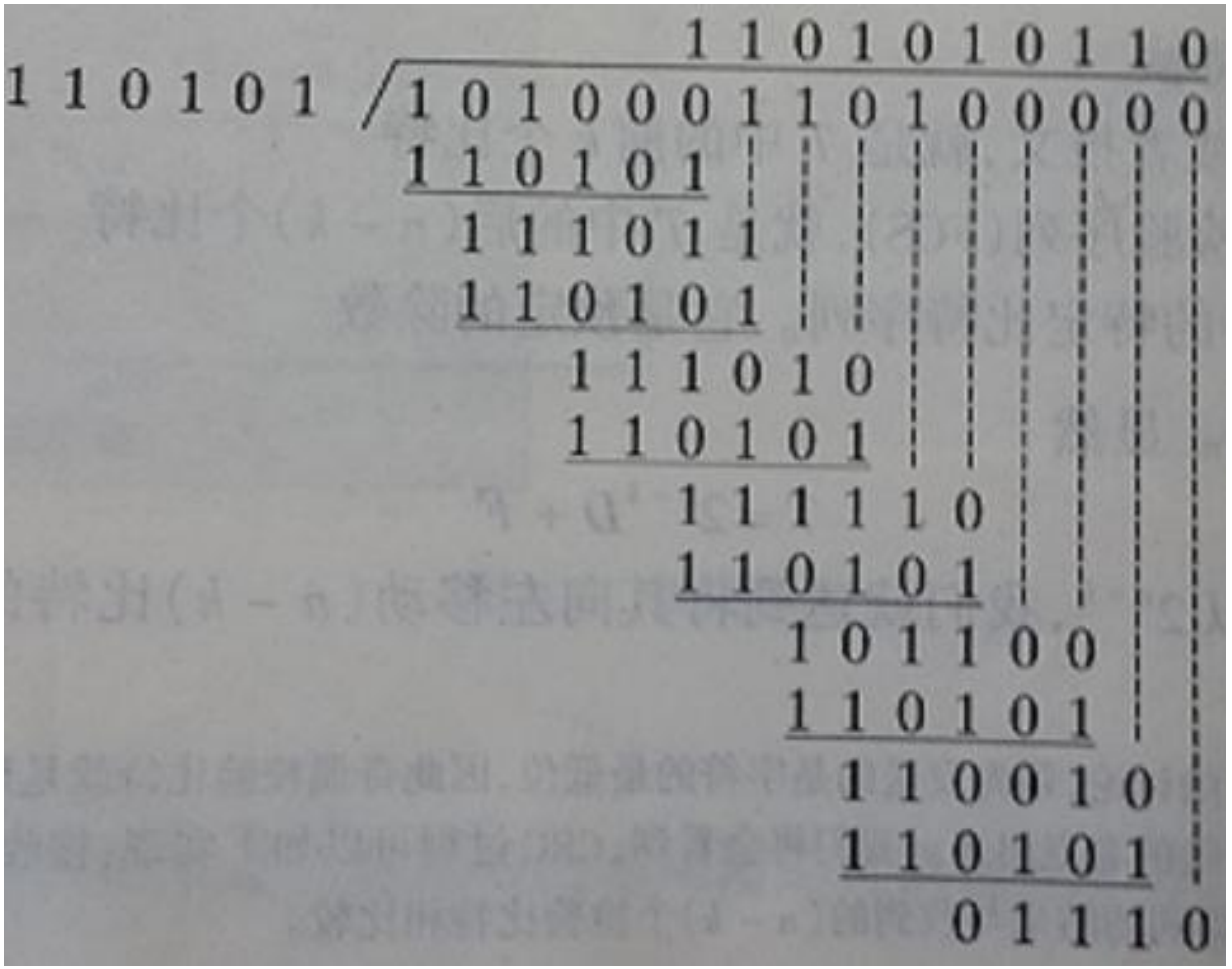
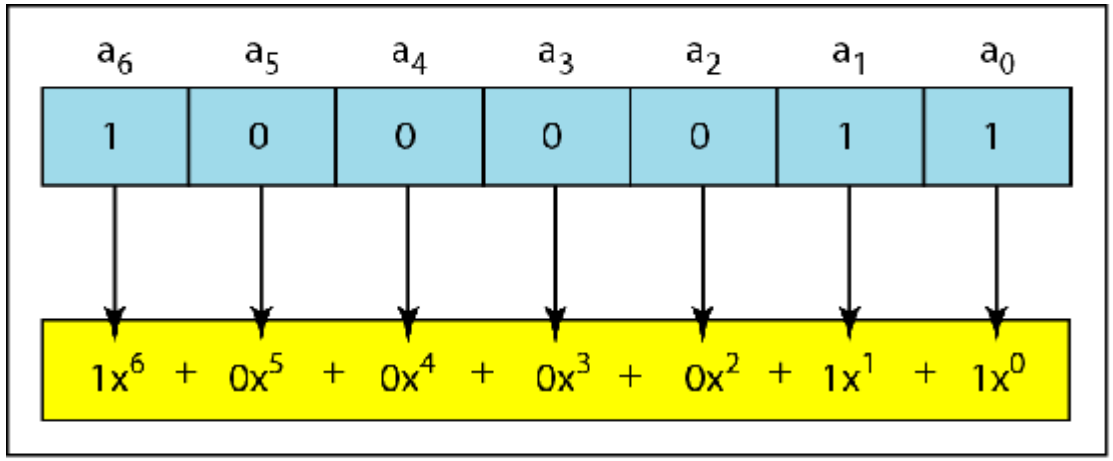
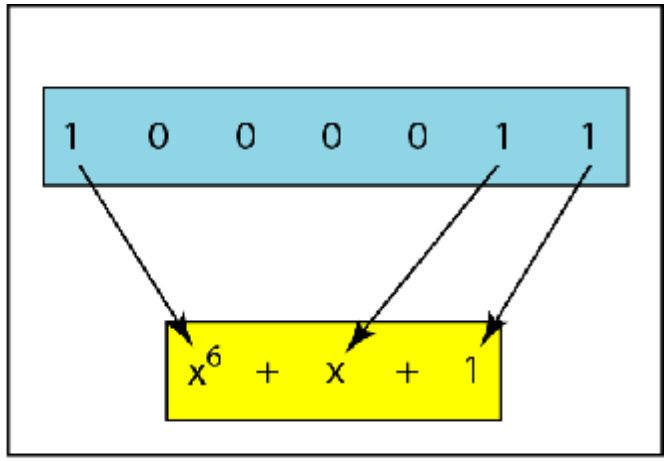


图10.21 多项式：循环冗余校验码可以用多项式表示

- 理解循环编码以及如何分析它们的更好办法是将它们表示为多项式；
- 每一项的幂次表示位所在的位置，系数表示位的值；
- 多项式的次数是多项式中的最高幂次



a. Binary pattern and polynomial

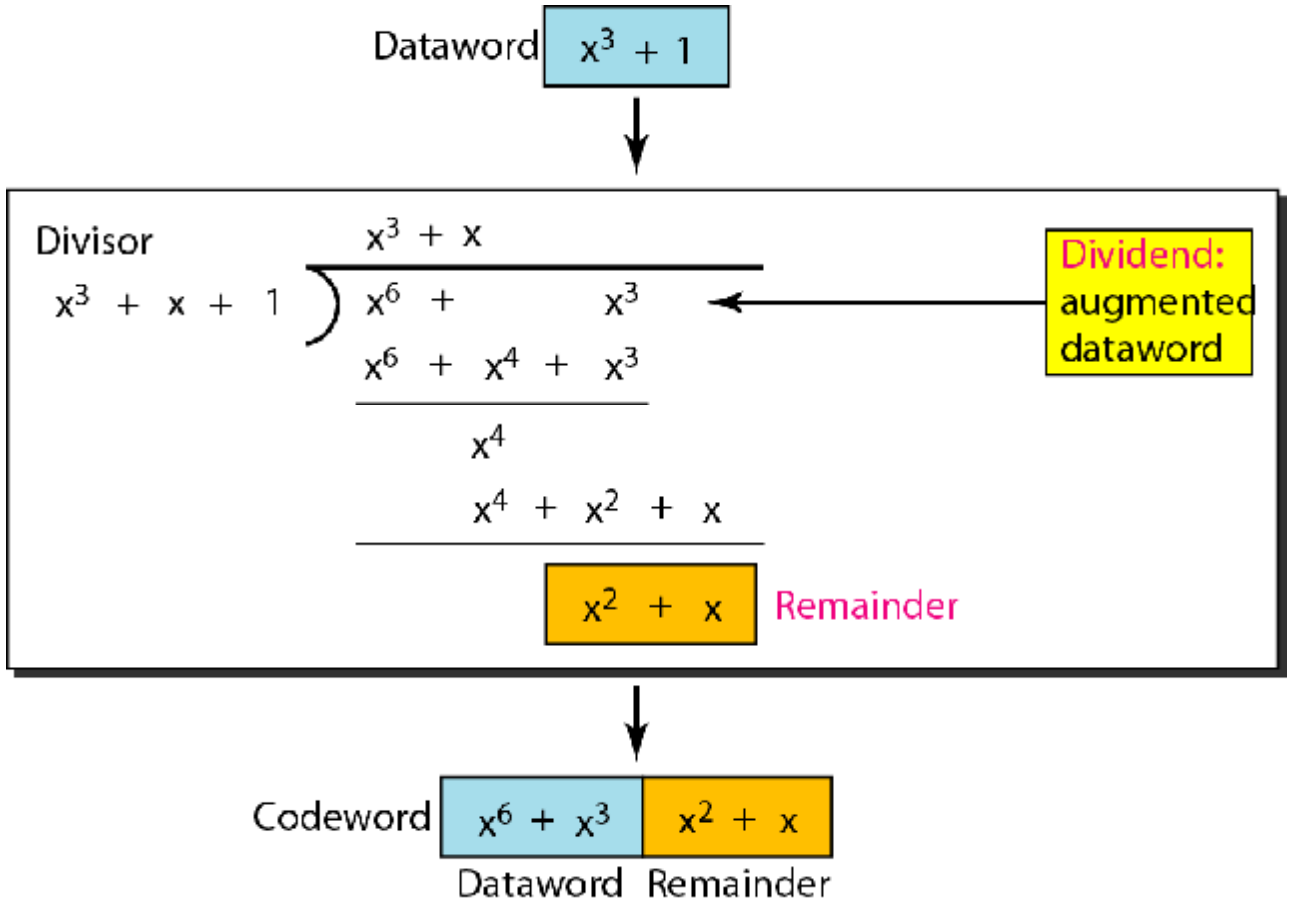


b. Short form

多项式运算（P191）

- 加减法：加和减相同；加减是删除相同项把不同项合并在一起（但如果三个多项式相加有3个相同项就要删除其中两个保留一个）；
- 项乘法和除法：幂次加减；
- 两个多项式相乘：第一个多项式的每一个项必须乘以第二个多项式的所有项，然后相同项对被删除简化；
- 多项式除法：与二进制除法相同；
- 移位：左移表示在右边加上若干个额外0，右移表示删除右边若干位

图10.22 使用多项式的CRC除法



循环编码中的除数通常称为生成多项式，简称为生成器

循环编码分析

p 假设f(x)为二进制系数的多项式，d(x)为数据字，c(x)为码字，g(x)为生成多项式，s(x)为校正子，e(x)为差错，则：

在循环编码中，

1. 如果s(x)!=0，则一位或多位被破坏。
2. 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。

p 哪些差错无法被译码器检测到呢？

接收到的码字=c(x)+e(x)　（表示发送的码字和差错之和）

则：

接收到的码字/g(x) = c(x)/g(x) + e(x)/g(x)

根据码字的定义，等式右边的第一项没有余数，校正子实际上是右边第二项的余数。所以，如果右边第二项没有余数，即e(x)为0或e(x)可以被g(x)整除，则校正子为0（代表可以被接收）

循环码中，那些可以被生成多项式**g(x)**整除的差错无法被捕捉到（发生的概率很小）。

若生成多项式至少有两项，且x^0（最右边的位）的系数是1，则所有单比特错误都可以被捕捉到（？）。

因为$x^i/(x^t+1)$始终有余数（t!=0）！

例10.15

如下哪个g(x)可以保证捕捉到单个位差错？对于每种情况，不能捕捉的单个差错是什么？

a. x+1

b. x^3

c. 1

例10.15

解：

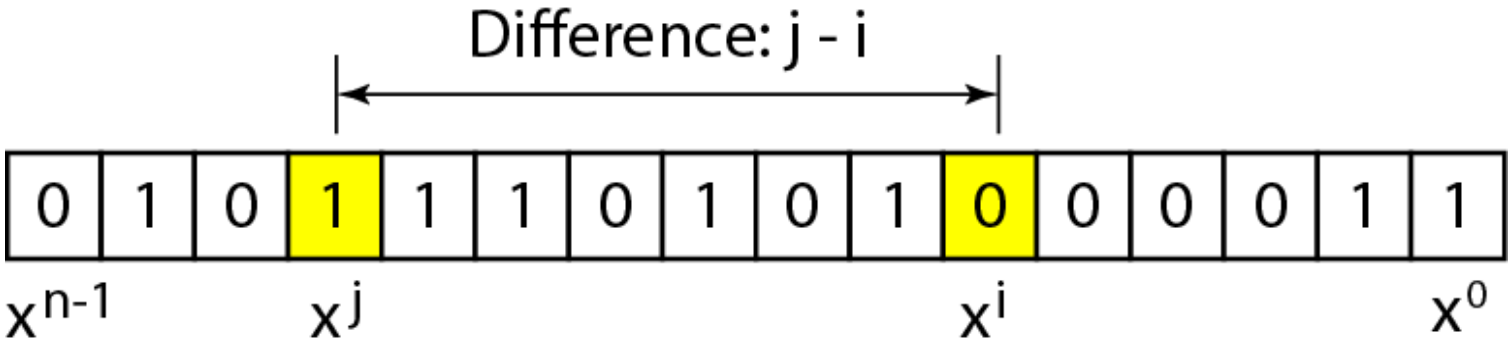
a. 没有一个x^i可以被x+1整除。换言之，$x^i/(x+1)$总有余数，因此，校正子不是零，任何单个位差错都可以被捕捉到。

b. 如果i>=3， x^i可以被g(x)整除。x^i/x^3的余数是0，接收方便会被蒙蔽而认为没有差错，虽然实际是一个差错。注意：这种情况下，被破坏的位必须位于第4位或更高位，它能捕捉到位1到位3的单个差错。

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。

图10.23 使用多项式表示的两个独立的单比特差错

- 差错$e(x) = x^j + x^i = x^i(x^{j-i} + 1)$
- 假设g(x)多于一项且其中一项是x^0，则要使得g(x)能整除e(x)，则g(x)必须能整除$x^{j-i}+1$，所以，要想检测出两个独立的单比特差错，g(x)一定不能整除x^t+1（$0<t<=n-1$）



若生成多项式不能整除$x^t + 1$（$0<t<=n-1$），那么所有独立的双差错都能被检测到。

例10.16

求出以下与两个独立的单个位差错相关的生成多项式的情况。

a. $x + 1$　　*b.* $x^4 + 1$　　*c.* $x^7 + x^6 + 1$　　*d.* $x^{15} + x^{14} + 1$

解：

a. 对生成多项式来说，这是一个很差的选择，任何两个相邻的差错都不能被检测到。

b. 不能检测到相隔4个位置的两个差错，这两个差错能位于任何位置，但如果它们的距离是4则无法检测到。

c. 这是个好的选择。

d. 如果t小于32768，这个多项式不能整除类型为x^t+1的任何差错。这表示一个码字中两个独立的差错相邻或者最多离开32768位都能被这个生成多项式检测到。

包含因子x+1的生成多项式能检测到所有奇数个差错。

P193： $(x+1)*(x^3+x^2+1) = x^4+x^2+x+1$

p针对突发性差错，假设L为差错的长度，r为冗余位个数，则：

Ø所有L<=r的突发性差错都会被检测到。
Ø所有L=r+1的突发性差错有$1-(1/2)^{r-1}$的概率会被检测到。
Ø所有L>r+1的突发性差错有$1-(1/2)^{r}$的概率会被检测到。

例10.17

下面的生成多项式检测突发错误的能力如何？

a. $x^6 + 1$　　*b.* $x^{18} + x^7 + x + 1$　　*c.* $x^{32} + x^{23} + x^7 + 1$

解：

a. 这个生成多项式可以检测出所有长度小于或等于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%，长度大于或等于8的突发性差错无法被检测到概率是千分之16。

b. 这个生成多项式可以检测出所有长度小于或等于18位的突发性差错；长度为19的突发性差错无法被检测到概率是百万分之8，长度大于或等于20的突发性差错无法被检测到概率是百万分之4。

C. 这个生成多项式可以检测出所有长度小于或等于32位的突发性差错，长度为33的突发性差错无法被检测到的概率是千万分之5，长度大于或等于34的突发性差错无法被检测到的概率是千万分之3。

生成多项式检错能力总结，高性能生成多项式需要有以下特性：

1. 它应该至少有两项。
2. x^0项的系数应该是1。
3. 它应该不能整除x^t+1 (t在2至n-1之间)。
4. 它应该有因子x+1。

表10.7 一些常用的标准生成多项式

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

循环编码的优点

- 循环编码在检测单个位差错、双差错、奇数个差错和突发性差错中有非常令人满意的性能；
- 可以很容易地以软硬件实现；
- 当以硬件实现时，速度尤其快，这使得循环编码成为许多网络的好选择。

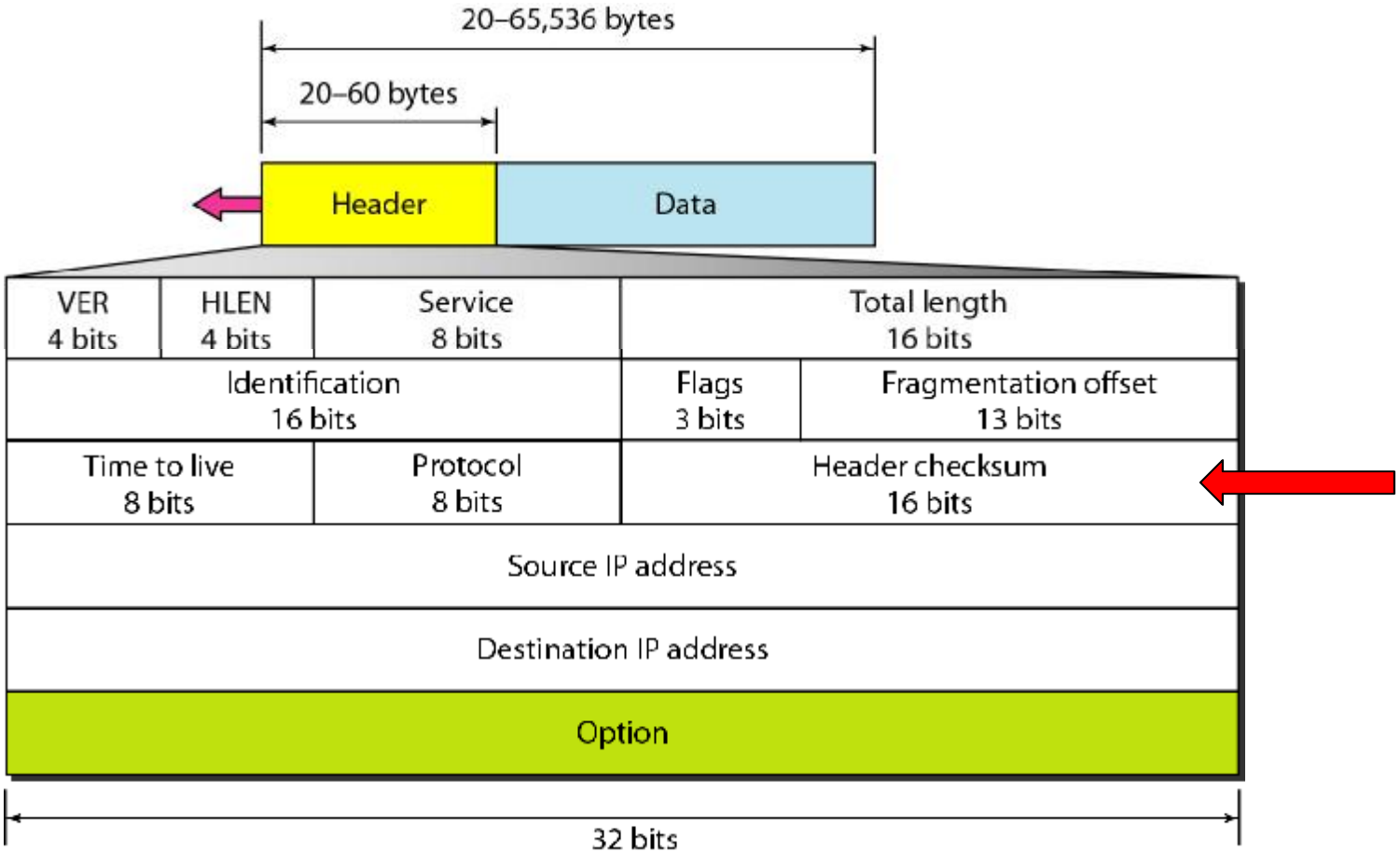
10-5 校验和（checksum）

p校验和在因特网中被许多不属于数据链路层的高层协议使用；

p像线性和循环编码一样，校验和也是基于冗余的概念，许多协议仍然使用校验和进行检错，虽然趋势是它将被CRC替代；

p这表示CRC还用于除了数据链路层之外的其他层。

IPv4数据报格式


20–65,536 bytes
20–60 bytes
Header
Data
VER
4 bits
HLEN
4 bits
Service
8 bits
Total length
16 bits
Identification
16 bits
Flags
3 bits
Fragmentation offset
13 bits
Time to live
8 bits
Protocol
8 bits
Header checksum
16 bits
Source IP address
Destination IP address
Option
32 bits

例10.18

假设我们要发送到目的地的数据是5个4位数字。除了发送这些数字外，我们还发送这些数字的和用于检错校验。例如，如果这组数字是（7、11、12、0、6），那么我们发送（7、11、12、0、6、36），这里36是原来数字的和。接收方将这5个数字相加并将相加的结果与这和比较。如果两者相同，接收方就认为没差错，接收这5个数字并丢弃和；否则就认为有差错，不接收这个数据。

例10.19

如果我们发送和的负值（补数）—称为校验和，就会使接收方的工作更容易。在这个例子中，我们发送（7、11、12、0、6、-36）。接收方将所有接收到的数字（包括校验和）相加。如果结果是0，就认为没差错，否则就认为有差错。

反码（one’s complement）

- 只使用n位表示0到2^n-1的无符号数字；
- 如果这个数字多于n位，那么最左边的额外位要加到最右边的n位；
- 在反码算法中，一个数的负数可用该数所有位取反（0变成1，1变成0）表示，这与从2^n-1减去这个数字一样

例10.20

在反码运算中如何只用4位表示数字21?

解:

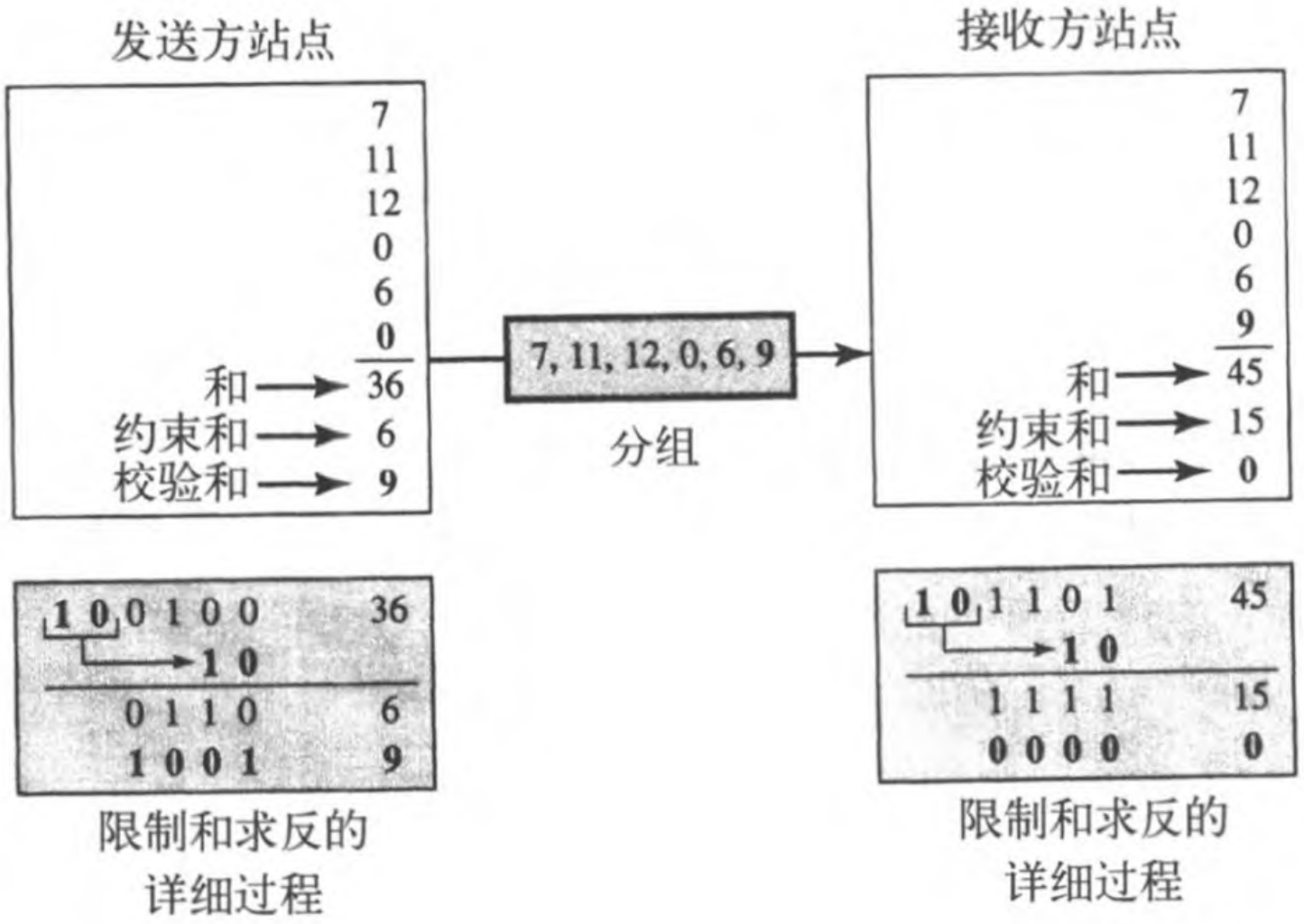
数字21的二进制表示是10101（需要5位），可以把最左边的位加到最右边的4位上，0101+1=0110或6。

例10.21

用反码解决负数问题：
在反码运算中如何只使用4位表示数字-6?

解：
在反码算法中，一个数的负数是将所有位取反。正6是0110，负6是1001，如果只考虑无符号数字，它是9。换言之，6的补数是9。在反码运算中，另一种求一个数的反码的方法是2^n-1减去这个数（即16-1-6）。

图10.24 例10.22-用反码重做例10.19



因特网16位校验和

发送方计算方法：

1. 报文被划分为16位字。
2. 校验和字的值设为0。
3. 所有字包括校验和使用反码运算相加。
4. 对这个和求反变成校验和。
5. 校验和随数据一起发送。

因特网16位校验和

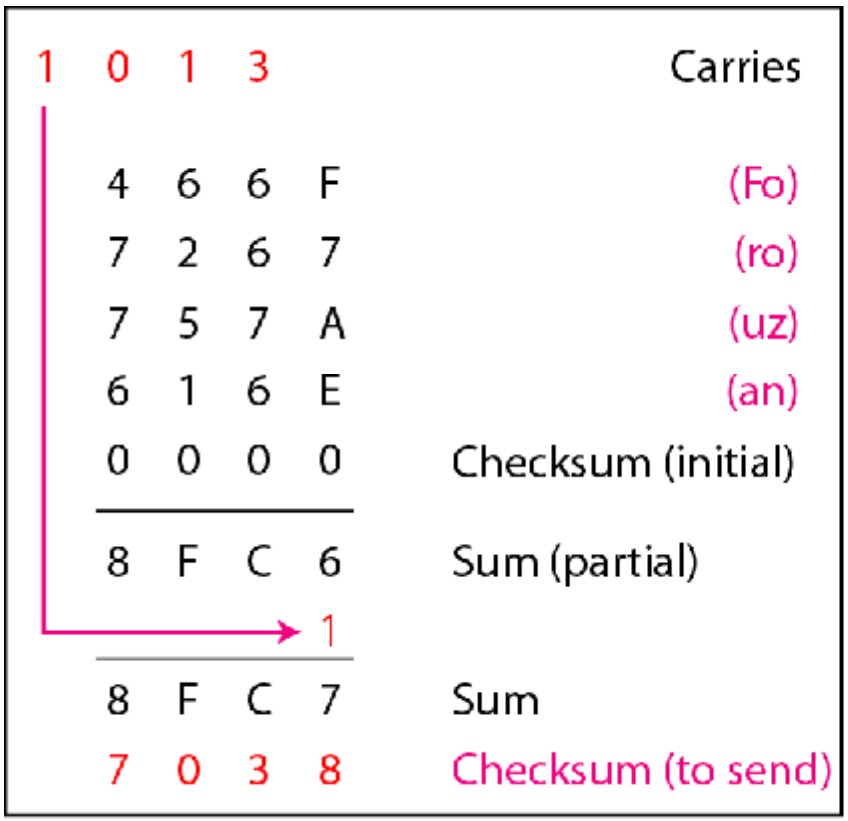
接收方差错检测过程：

1. 报文（包括校验和）被划分成16位字。
2. 用反码加法将所有字相加。
3. 对该和求反，生成新的校验和。
4. 如果校验和的值是0，接收报文，否则就丢弃报文。

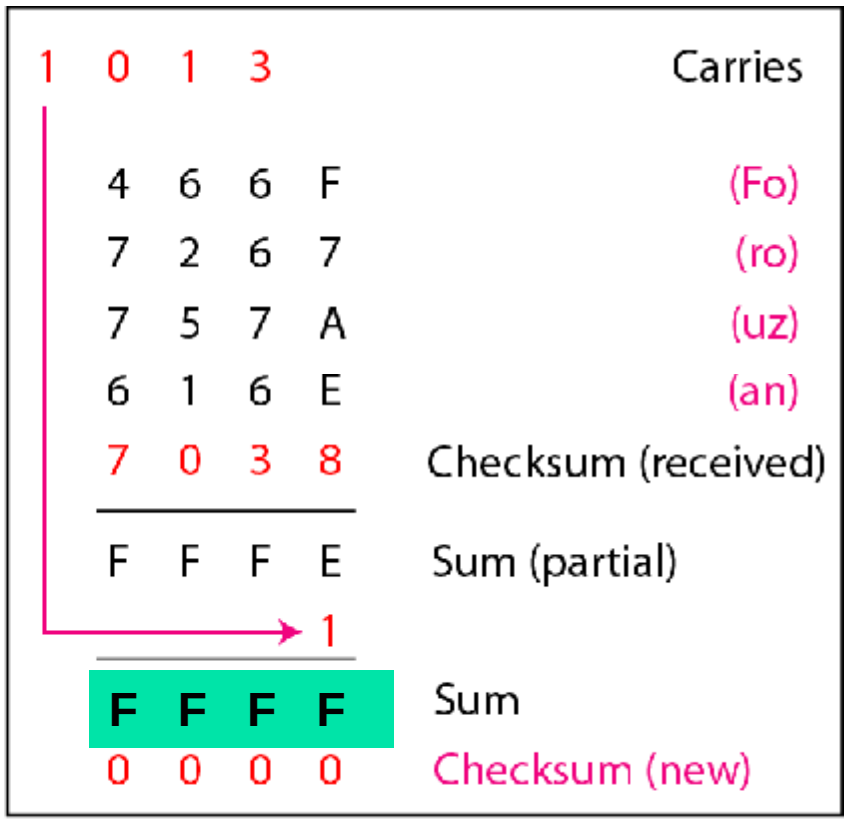
例10.23

让我们计算8个字符（Forouzan）文本的校验和。该文本需要划分成2个字节（16位）的字。使用ASCII表（见附录A）把每个字符变成2位十六进制数。例如，F表示成0x46，o表示成0x6F。图10.25说明了如何在发送方站点和接收方站点计算校验和。

图10.25 例10.23



a. Checksum at the sender site



a. Checksum at the receiver site

校验和性能

- 传统的校验和使用较少的位（16位）来检测任何长度（有时是数千位）报文中的差错，但是，在差错检测能力上它没有CRC强；
- 例如，增加一个字的值而另一个字以相同的值减小，因为和以及校验和不变，所以这两个差错无法被检测到；
- 还有，如果多个字的值增加，但是总的变化量是65535的倍数，则和以及校验和也不变，这表示差错也无法被检测到；
- 因特网的趋势是由CRC代替校验和。