数据通信

差错检测与纠正

殷亚凤
yafeng@nju.edu.cn
http://cs.nju.edu.cn/yafeng/
Room 301, Building of CS

内容解答

- 直接调频，间接调频：
  - 直接调频在实现线性调频的要求下，可以获得相对较大的频偏，但会导致FM波的中心频率偏移，频率稳定度差；间接调频波突出的优点是**载波中心频率的稳定性**可以做得较高，但可能得到的最大频偏较小。
- 直接调相，间接调相：

差错检测与纠正

1. 差错类型 (Types of Error)
2. 差错检测 (Error Detection)
3. 差错纠正 (Error Correction)

差错类型 Types of Error

错误原因

- 不可靠通信信道 （Unreliable communication channels）
- 传输损伤，噪声

单比特错误 single bit errors

- 只改变**一个比特**，并不影响临近的其他比特
- 白噪声，信噪比恶化

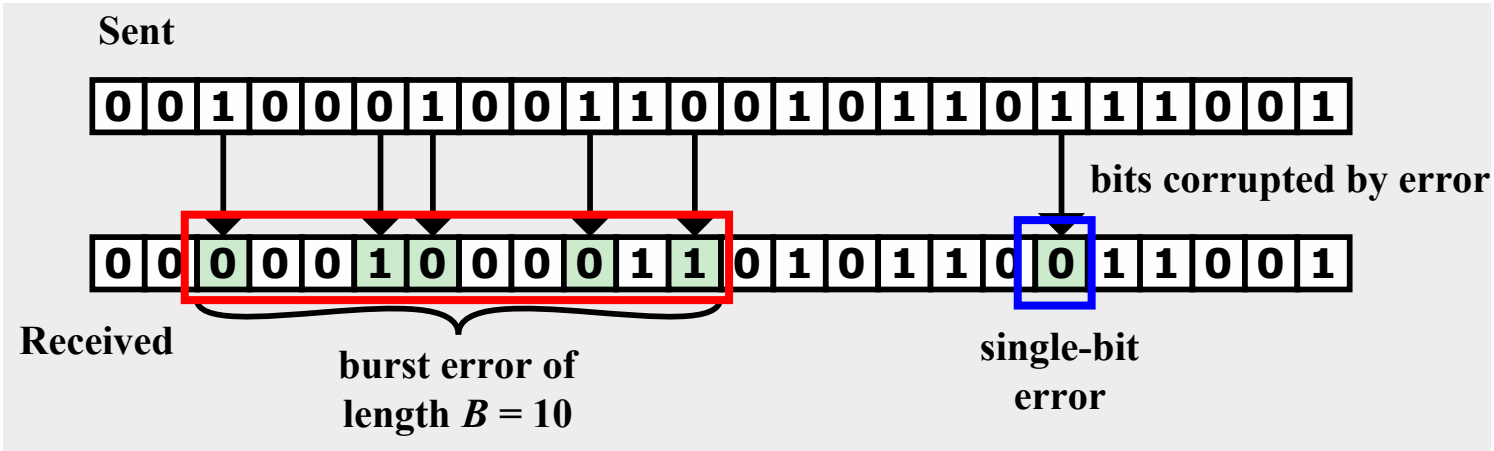
突发性错误 burst errors

- **长度为B**的突发性错误
- 冲击噪声，无线信道信号衰落

突发性差错

- IEEE Std 100， ITU-T 建议书 Q.9所做定义：

差错突发：在一组比特中，任意两个连续的差错比特之间间隔的正确比特数总是小于x，x为一个给定值。因此在差错突发时的最后一个差错比特与下一次突发的第一个差多比特之间会有x个以上的正确比特间隔。



- 数据率越高，突发性差错影响越大

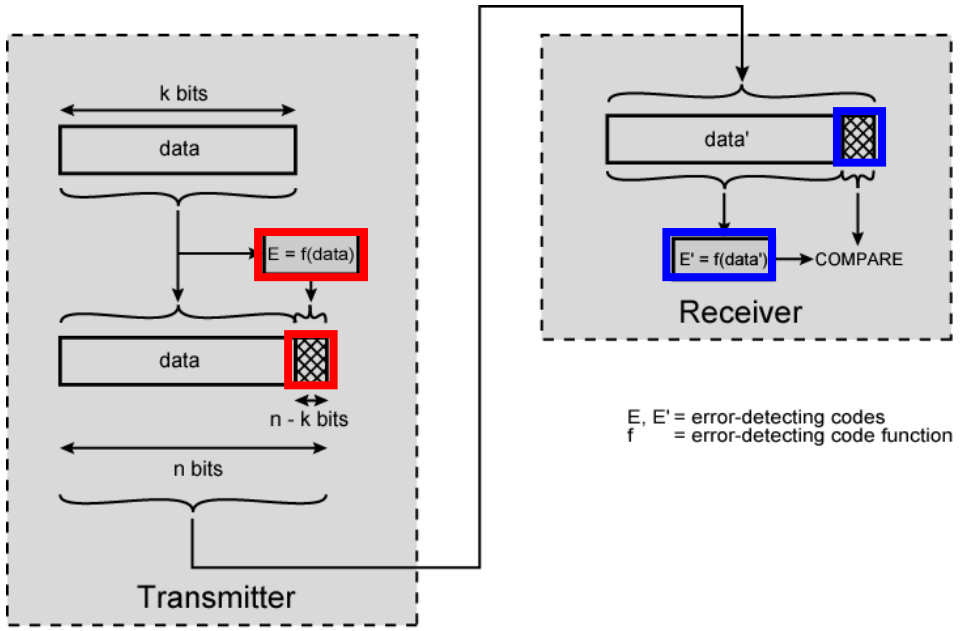
假设发生了**1μs**的冲激噪声事件或衰落时间，当数据率为10Mbps时，导致的差错突发为**10比特**，当数据率为100Mbps时，就会出现**100比特**的突发性差错。

差错检测与纠正

1. 差错类型 (Types of Error)
2. **差错检测 (Error Detection)**
3. 差错纠正 (Error Correction)

差错检测

- 通过使用**差错检测码**（error detecting code）来检测错误
  - 由发送端添加到数据块后面一些额外的比特
    - k比特数据块，增加(n-k)比特检测码，(n-k)<k
  - 接收端重新执行差错检测计算，并**与接收到的差错检测码比较**
  - 不同：检测到差错



奇偶校验 Parity Check

- 奇偶校验：使整个字符中1的个数为偶数（**偶校验**）或奇数（**奇校验**）

偶校验 – 偶数个1

- 一般用于同步传输

奇校验 – 奇数个1

- 一般用于异步传输

- 在数据块的末尾附加奇偶校验比特
- e.g. 字符传输，7比特字符1110001 → 附加1比特奇偶校验比特**1**1110001
- 如果有两个（或任意偶数个）比特因错误而翻转，就会出现检测不到的差错

奇偶校验 Parity Check

- **二维奇偶校验**

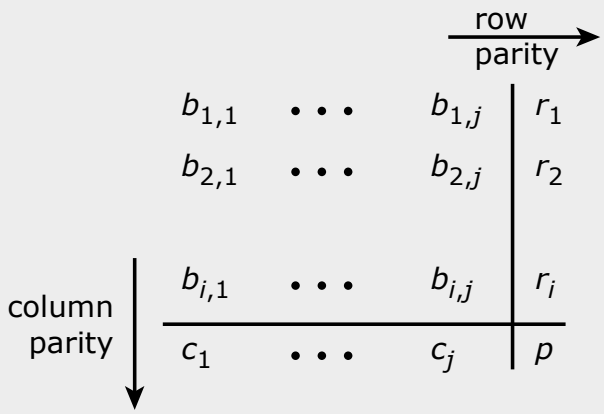
1. **每一行**i上添加改行的偶校验比特ri；
2. **每一列**j上添加该列的偶校验比特cj；
3. **整体**添加一个奇偶校验比特p

→**i+j+1个**奇偶校验比特

✓更强的错误检测

✓具有一定的纠错能力

✓矩形的四个错误无法检测



(a) Parity calculation

0	1	1	1	0	1
0	1	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	0	0	1	1	0

(b) No errors

0	1	1	1	0	1
0	⓪	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	0	0	1	1	0

row parity error

column parity error

(c) Correctable single-bit error

0	1	1	1	1	1	0	1
0	⓪	1	1	0	①	1	0
0	0	1	1	0	0	1	1
0	⓪	0	0	0	⓪	0	0
1	0	1	1	1	1	1	0
1	1	0	0	0	1	1	0

(d) Uncorrectable error pattern

因特网检验和

- 因特网检验和是许多因特网协议所采用的差错检测码，包括IP，TCP，UDP
- 利用**二进制反码运算**以及**反码求和**
  - 将两个数字视为无符号二进制整数，然后相加
  - 如果最左边有进位比特，则和再加1（循环进位）

$$\begin{array}{r} 0011 \\ +\ 1100 \\ \hline 1111 \end{array}$$

$$\begin{array}{r} 1101 \\ +\ 1011 \\ \hline 11000 \\ +\ \quad 1 \\ \hline 1001 \end{array}$$

- 相比于奇偶校验，因特网校验和差错检测能力更强
- 涉及简单的加法和比较运算，开销小
- 链路层使用循环冗余校验（CRC）情况下，因特网校验和作为补充到端到端的差错检测

因特网检验和

- 将检验和字段全部置为0;
- 将手部中所有的八位组执行反码求和;
- 将计算结果进行反码运算

00 01 F2 03 F4 F5 F6 F7 00 00

Partial sum	0001 F203 F204
Partial sum	F204 F4F5 1E6F9
Carry	E6F9 1 E6FA
Partial sum	E6FA F6F7 1DDF1
Carry	DDF1 1 DDF2
Ones complement of the result	220D

(a) Checksum calculation by sender

Partial sum	0001 F203 F204
Partial sum	F204 F4F5 1E6F9
Carry	E6F9 1 E6FA
Partial sum	E6FA F6F7 1DDF1
Carry	DDF1 1 DDF2
Partial sum	DDF2 220D FFFF

(b) Checksum verification by receiver

循环冗余校验

循环冗余校验Cyclic Redundancy Check (CRC)

- 一种最常用也最有效的差错检验码
- 给定**k位的比特块**，发送器生成一个**n-k位**的比特序列，即**帧检验序列** *frame check sequence* (FCS)
- 含有**n比特的帧能被**一些预先设定好的数值**整除**
- 接收器用同样的数值对接收到的帧进行除法运算，若没有余数，则认为没有差错

模2运算

模2运算是实现循环冗余检验的一种处理方法：

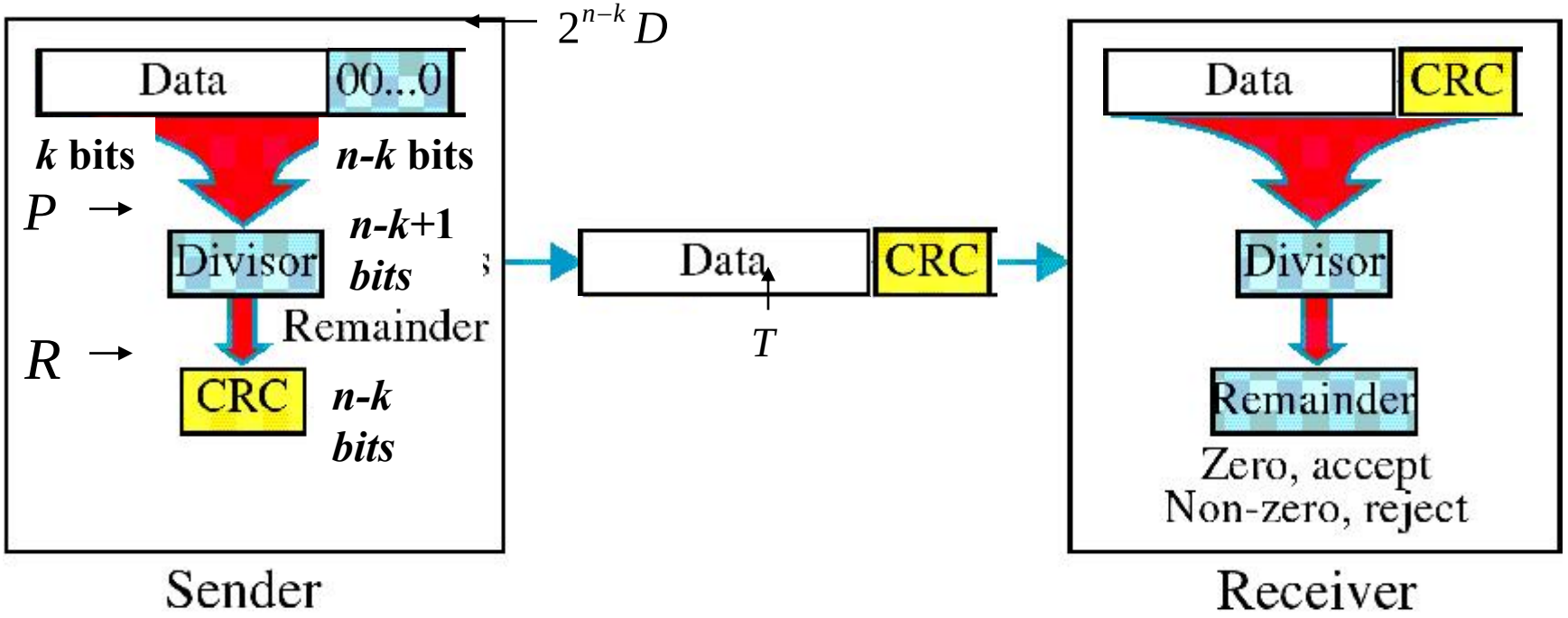
- 模2运算使用无进位的二进制加法，恰好是异或操作

$$\begin{array}{r} 1111 \\ +1010 \\ \hline 0101 \end{array} \qquad \begin{array}{r} 1111 \\ -0101 \\ \hline 1010 \end{array} \qquad \begin{array}{r} 11001 \\ \times\ 11 \\ \hline 11001 \\ 11001 \\ \hline 101011 \end{array}$$

循环冗余校验

T-要发送的n比特； D-k比特数据块； F-(n-k)比特检验序列； P-(n-k+1)比特除数

问题：如何选择F, 使得 $T\%P=0,\ \text{i.e.,}\ (2^{n-k}D+F)\%P=0$ $T=2^{n-k}D+F$



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

循环冗余校验

1. 给定

报文 $D = 1010001101$ (10 bits)

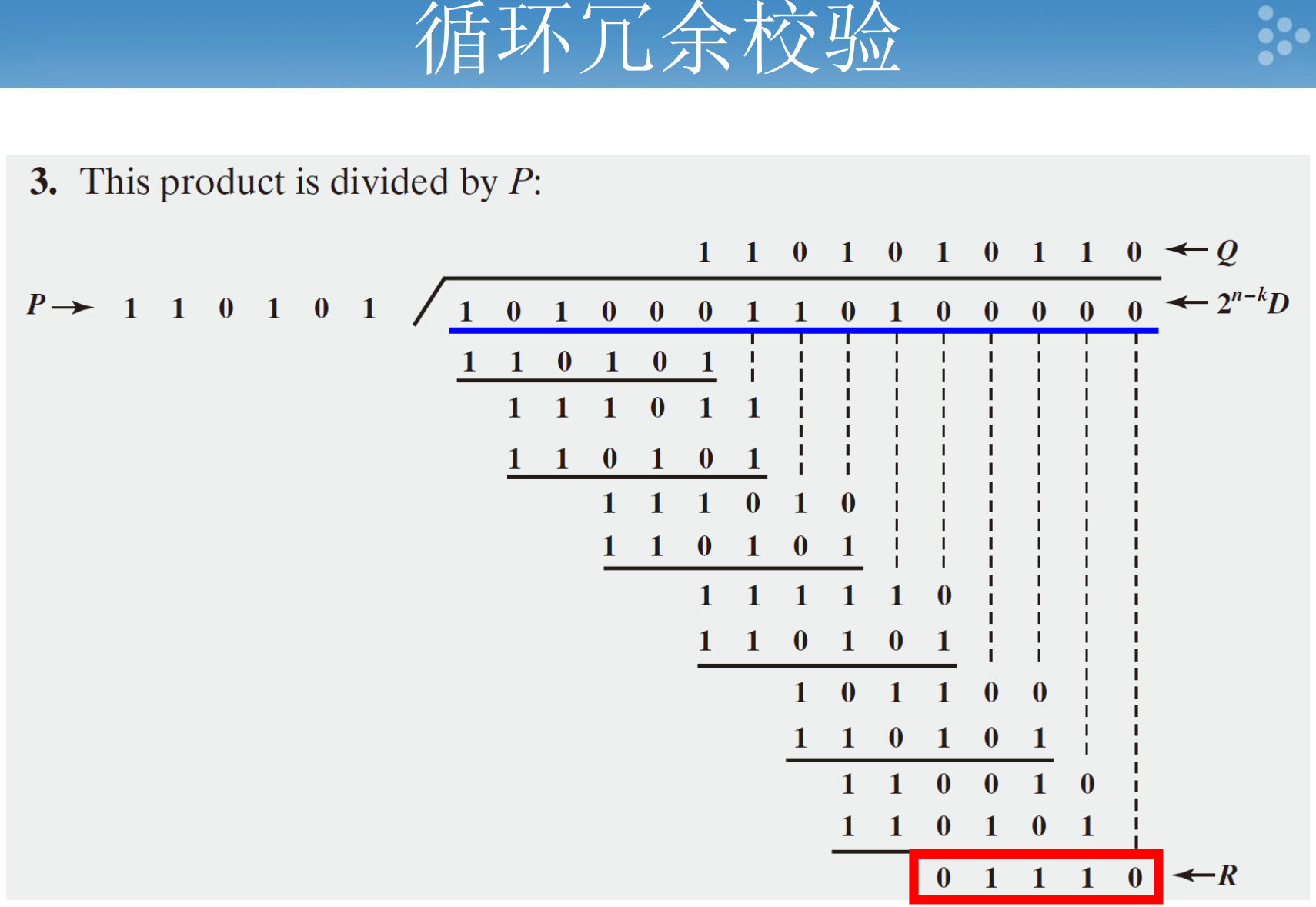
预定比特序列 $P = 110101$ (6 bits)

帧检验序列 $R =$ to be calculated (5 bits)

Thus, $n = 15$, $k = 10$, and $(n - k) = 5$.

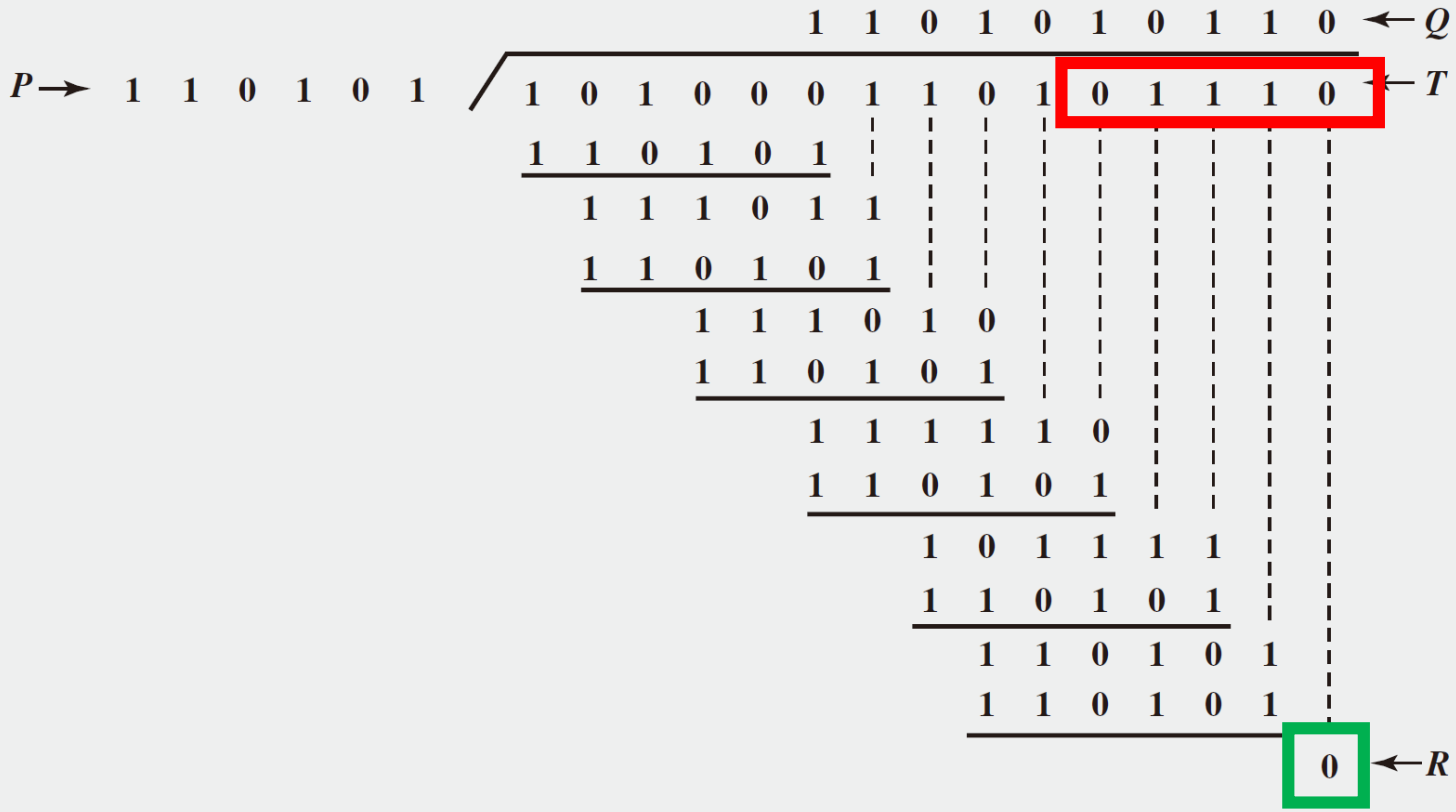
2. The message is multiplied by 2^5, yielding 101000110100000.

循环冗余校验


3. This product is divided by P:
1 1 0 1 0 1 0 1 1 0 ← Q
P → 1 1 0 1 0 1
1 0 1 0 0 0 1 1 0 1 0 0 0 0 0 ← 2^{n−k}D
1 1 0 1 0 1
1 1 1 0 1 1
1 1 0 1 0 1
1 1 1 0 1 0
1 1 0 1 0 1
1 1 1 1 1 0
1 1 0 1 0 1
1 0 1 1 0 0
1 1 0 1 0 1
1 1 0 0 1 0
1 1 0 1 0 1
0 1 1 1 0 ← R

循环冗余校验

4. The remainder is added to 2^5D to give $T = 101000110101110$, which is transmitted.
5. If there are no errors, the receiver receives T intact. The received frame is divided by P:



Because there is no remainder, it is assumed that there have been no errors.

多项式

- 多项式表示
  - D=110011　D(X)=X^5+X^4+X+1
  - P=11001　　P(X)=X^4+X^3+1

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

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

D = 1010001101 → $D(X) = X^9+X^7+X^3+X^2+1$,

P = 110101, → $P(X) = X^5+X^4+X^2+1$.

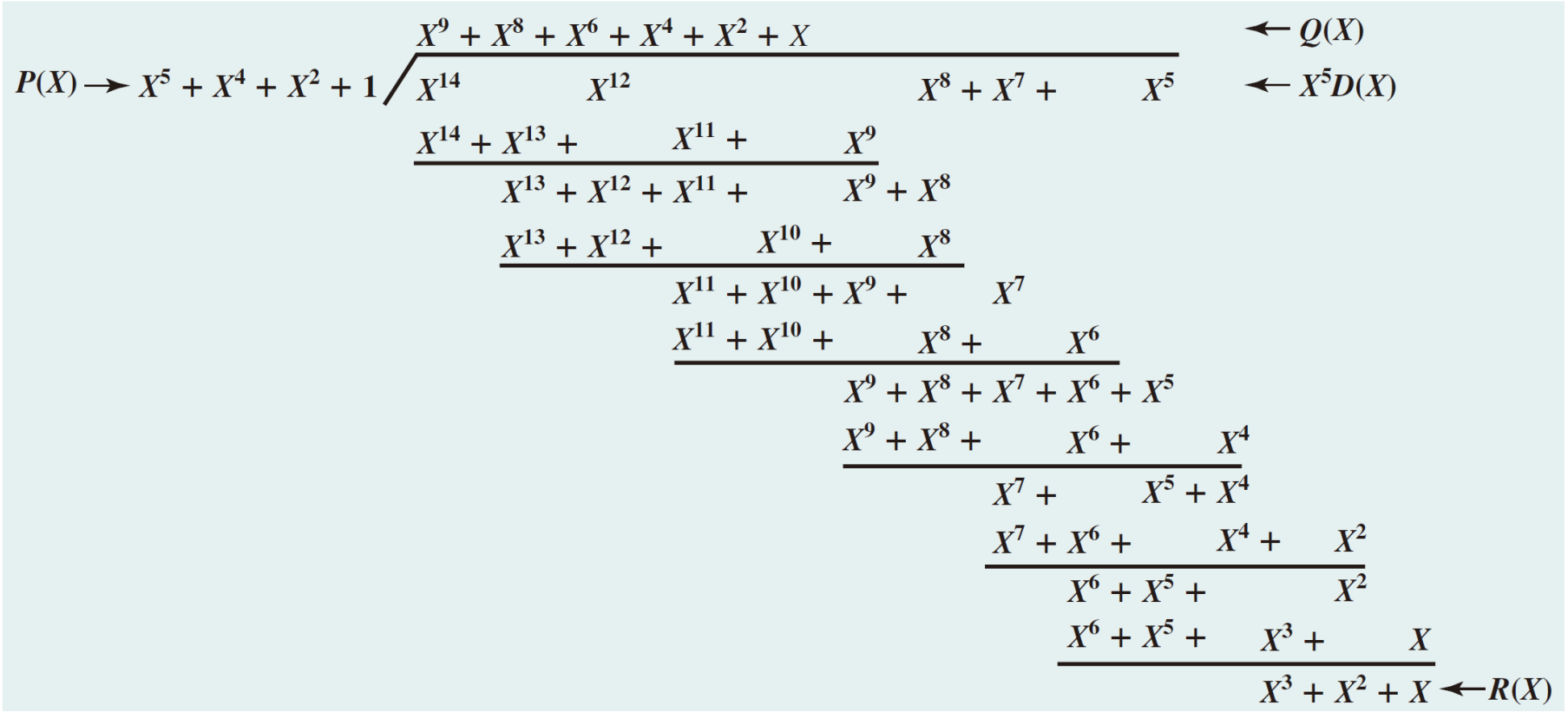
R = 01110, → $R(X) = X^3+X^2+X$.

多项式

D = 1010001101 → $D(X) = X^9 + X^7 + X^3 + X^2 + 1$,

P = 110101, → $P(X) = X^5 + X^4 + X^2 + 1$.

R = 01110, → $R(X) = X^3 + X^2 + X$.



多项式

- **预定比特序列P要比要求的帧检验序列FCS多一个比特**
- 检测差错等价于选择P使得**差错不能被P整除**
- 广泛使用的P(X)序列

CRC-12

$$x^{12}+x^{11}+x^{3}+x+1$$

CRC-16

$$x^{16}+x^{15}+x^{2}+1$$

CRC-CCITT

$$x^{16}+x^{12}+x^{5}+1$$

CRC-32

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

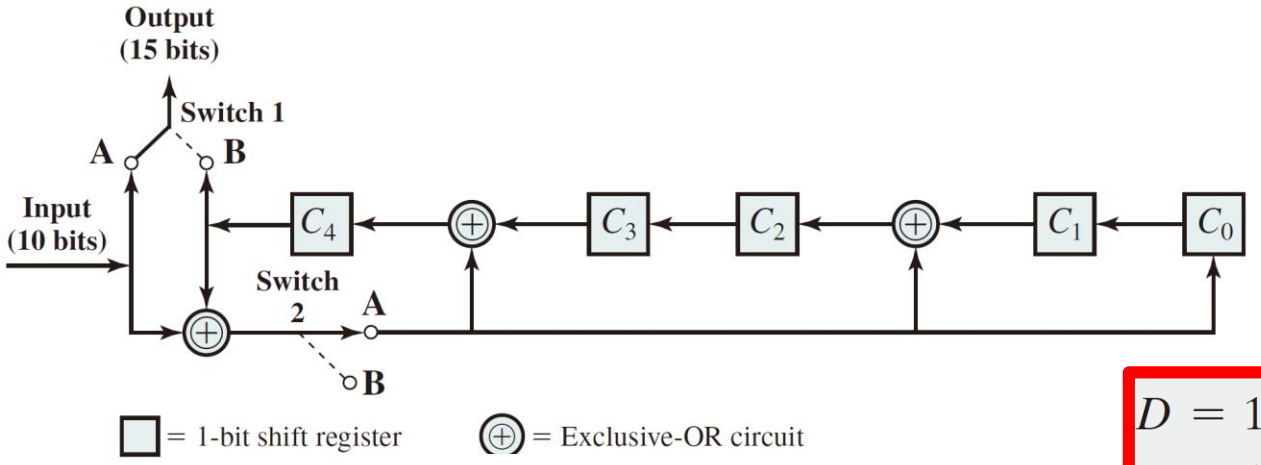
数字逻辑

- CRC过程可以由一些异或门和移位寄存器组成的除法电路实现：
  - 寄存器含有n-k比特，等于FCS长度；
  - 共有n-k个异或门；
  - 异或门是否存在，对应于多项式除数P(X)中的某一项是否存在，除了项1和X^{n-k}

数据 $D = 1010001101; \quad D(X) = X^9 + X^7 + X^3 + X^2 + 1$

除数 $P = 110101; \quad P(X) = X^5 + X^4 + X^2 + 1$

数字逻辑



(a) 移位寄存器的实现

$D = 1010001101;$
$P = 110101;$

	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
Step 5	1	0	0	1	0	1	0	1	0
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				

Message to be sent

(b) Example with input of 1010001101

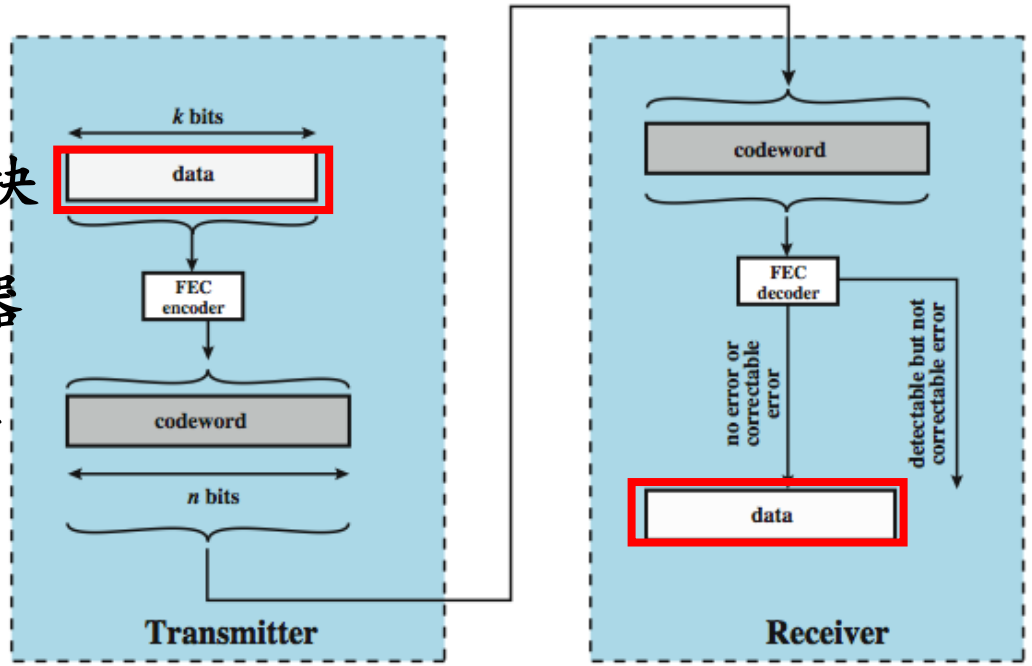
除以多项式$X^5+X^4+X^2+1$的移位寄存器电路

差错检测与纠正

1. 差错类型 (Types of Error)
2. 差错检测 (Error Detection)
3. **差错纠正 (Error Correction)**

差错纠正

- **差错检测**需要数据块重传机制（ARQ chapter 7）
- **差错重传**不适用与信道状态较差的链路
  - 无线链路：高比特差错率导致大量重传
  - 卫星链路：长传播时延导致效率低下
- **前向纠错（FEC）**：接收器能够在接收过程中根据传输的比特来纠



k比特数据块

编码器

n比特码字

码字

解码器

数据块

前向纠错

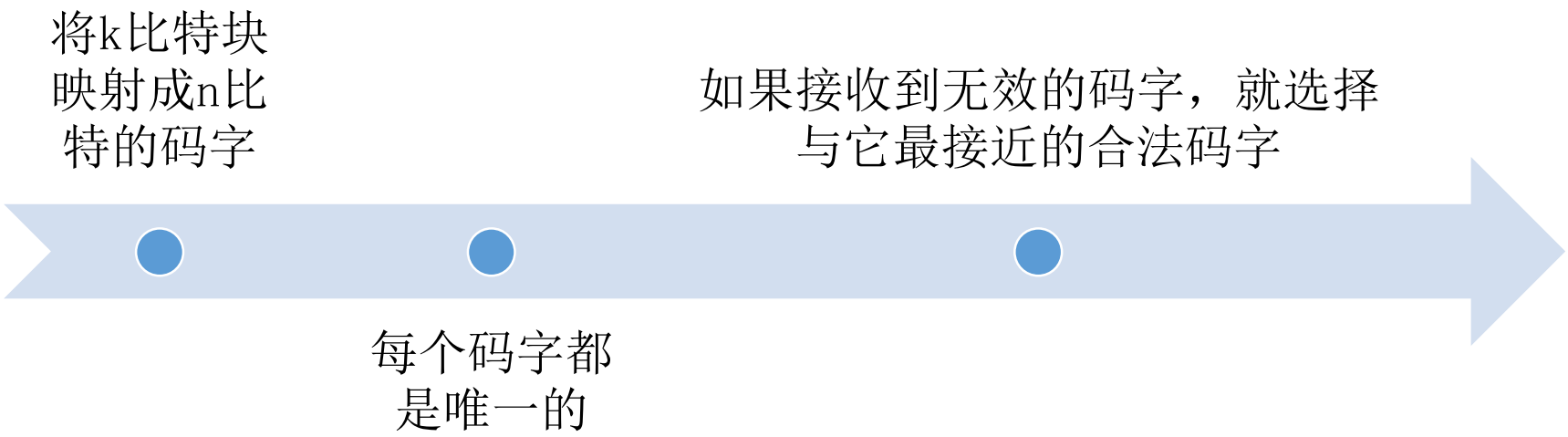
FEC得到的四种可能输出：

- **无差错**：没有比特差错，FEC解码输入与原码字一致，解码器生成原数据块
- **可检测，可纠正差错**：即使收到的数据块与被传输的码字不同，FEC解码器也能通过映射产生原数据块
- **可检测，不可纠正之差错**：解码器能够检测到差错，但无法纠正
- **不可检测之差错**：解码器未能检测到已经出现的差错，产生与原数据块不一致的k比特数据块

差错纠正流程

- 差错纠正通过在传输报文上附加冗余信息完成：冗余信息使接收器能够推算出原报文

- 块纠错码：一种广泛使用的纠错码



块码原理

- **汉明距离**：d(v1，v2)是指v1和v2之间不同比特的个数

$$v_1 = 011011, \quad v_2 = 110001$$

$$d(v_1, v_2) = 3$$

Data Block	Codeword
00	00000
01	00111
10	11001
11	11110

00100

$$d(00000, 00100) = 1;$$
$$d(11001, 00100) = 4;$$
$$d(00111, 00100) = 2;$$
$$d(11110, 00100) = 3$$

- 如果接收到一个非法码字，那么选择与它最近（最短距离）的合法码字。

块码原理

- (n，k)块码：共有2^n个码字，其中合法码字为2^k个
- (n-k)/k：编码的**冗余度**
- k/n：**编码率**
- 假设一个编码由码字$w_1, w_2, \ldots w_s$组成，其中$s=2^n$，则编码的**最短距离**为为：

$$d_{\min} = \min_{i \neq j} [d(\mathbf{w}_i, \mathbf{w}_j)]$$

- 如果 $d_{\min} \geq (2t+1)$ 正t个比特差错
- 如果 $d_{\min} \geq 2t$ 能检测t比特差错，纠正小于等于t-1个比特差错

纠正： $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$　　**检测：** $t = d_{\min} - 1$

纠错编码的基本原理

- 块码的设计要点
  - 对于给定的k和n，希望d_{min}尽可能大
  - 编码解码过程相对简单，内存开销和处理时间尽量小
  - 希望附加比特数$(n-k)$较少，减少带宽
  - 希望附加比特数$(n-k)$较多，减少差错率
    - 后两个目标相悖，需折衷考虑
- 编码增益
  - 编码后误码率降低

纠错编码的基本原理

分组码基本原理：举例说明如下。

- 设有一种由3位二进制数字构成的码组，它共有8种不同的可能组合。若将其全部用来表示天气，则可以表示8种不同天气，

  例如：“000”（晴），“001”（云），
  “010”（阴），“011”（雨），
  “100”（雪），“101”（霜），
  “110”（雾），“111”（雹）。

- 其中任一码组在传输中若发生一个或多个错码，则将变成另一个信息码组。这时，接收端将无法发现错误。

纠错编码的基本原理

若在上述8种码组中只准许使用4种来传送天气，例如：

“000”＝晴　“011”＝云　“101”＝阴　“110”＝雨

- 这时，虽然只能传送4种不同的天气，但是接收端却有可能发现码组中的一个错码。
- 例如，若“000”（晴）中错了一位，则接收码组将变成“100”或“010”或“001”。这3种码组都是不准使用的，称为**禁用码组**。
- **接收端在收到禁用码组时，就认为发现了错码**。当发生3个错码时，“000”变成了“111”，它也是禁用码组，故这种编码也能检测3个错码。
- 但是这种码不能发现一个码组中的两个错码，因为发生两个错码后产生的是**许用码组**。

纠错编码的基本原理

检错和纠错

- 上面这种编码**只能检测**错码，**不能纠正**错码。例如，当接收码组为禁用码组“100”时，接收端将无法判断是哪一位码发生了错误，因为晴、阴、雨三者错了一位都可以变成“100”。

- 要能够**纠正**错误，还要**增加多余度**。例如，若规定许用码组只有两个：“000”（晴），“111”（雨），其他都是禁用码组，则能够检测两个以下错码，或能够纠正一个错码。

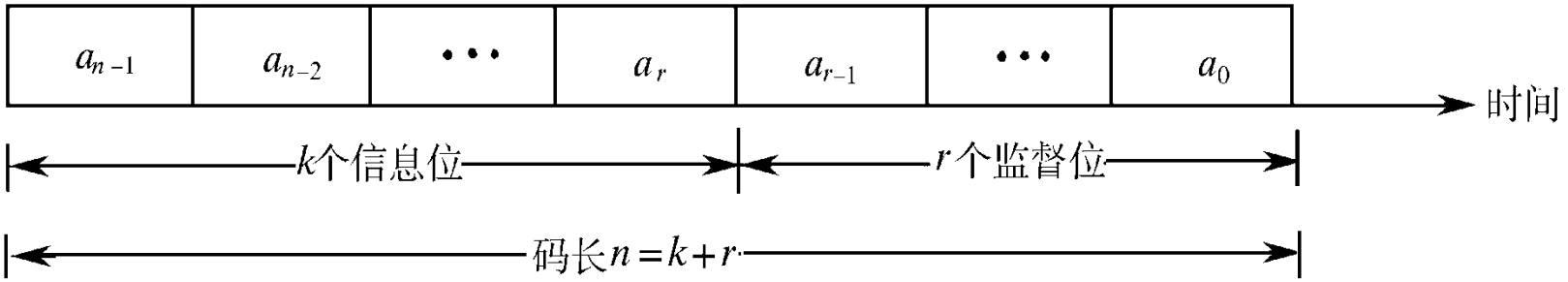
- 例如，当收到禁用码组“100”时，若当作**仅有一个错码**，则可以判断此错码发生在“1”位，从而**纠正**为“000”（晴）。因为“111”（雨）发生任何一位错码时都不会变成“100”这种形式。

- 但是，这时若假定错码数不超过**两个**，则存在两种可能性：“000”错一位和“111”错两位都可能变成“100”，因而只能检测出存在错码而**无法纠正**错码。

纠错编码的基本原理

分组码的结构

- 将信息码分组，为每组信息码**附加若干监督**码的编码称为**分组码**。
- 在分组码中，监督码元仅监督本码组中的信息码元。
- **信息位**和**监督位**的关系



- 例子：

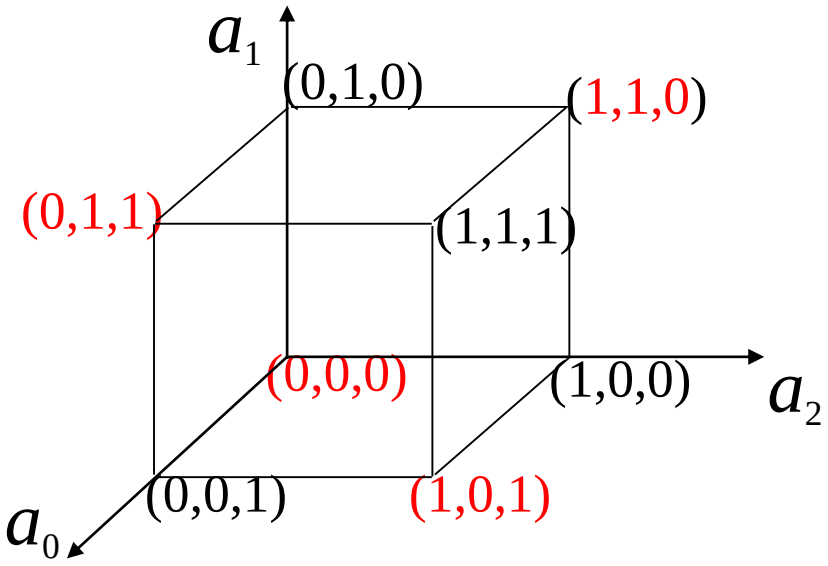
	信息位	监督位
晴	00	0
云	01	1
阴	10	1
雨	11	0

纠错编码的基本原理

- 分组码的码重和码距
  - **码重**：把码组中“1”的个数目称为码组的重量，简称**码重**。
  - **码距**：把两个码组中对应位上数字不同的位数称为码组的距离，简称**码距**。码距又称**汉明距离**。
  - 例如，“000”＝晴，“011”＝云，“101”＝阴，“110”＝雨，4个码组之间，任意两个的距离均为2。
  - **最小码距**：把某种编码中各个码组之间距离的最小值称为**最小码距**(d_0)。例如，上面的编码的最小码距$d_0 = 2$。

纠错编码的基本原理

- **码距**的几何意义

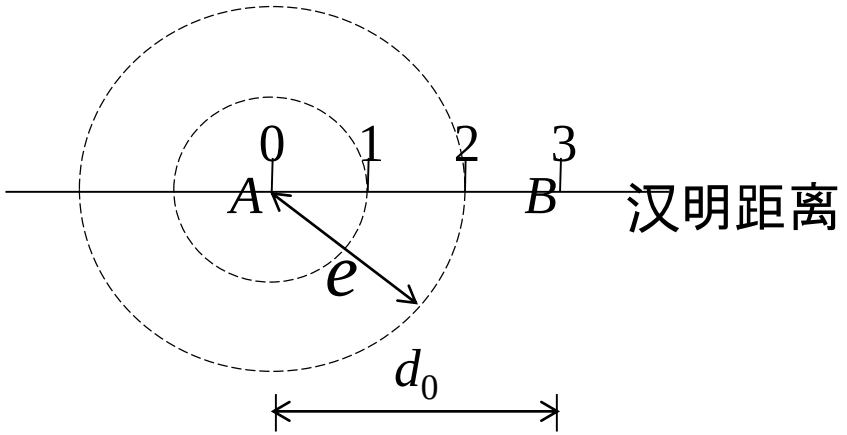


  - 对于3位的编码组，可以在3维空间中说明码距的几何意义。
  - 每个码组的3个码元的值(a_1, a_2, a_3)就是此立方体各顶点的坐标。而上述码距概念在此图中就对应于各顶点之间沿立方体各边行走的几何距离。
  - 由此图可以直观看出，上例中4个**准用码组之间的距离均为2**。

纠错编码的基本原理

- 码距和检纠错能力的关系
  - **为检测e个错码，要求最小码距 $d_0 \geq e+1$**
  - 同理，若一种编码的最小码距为d_0，则将能检测$(d_0 - 1)$个错码。反之，若要求检测e个错码，则最小码距d_0至少应不小于$(e+1)$。

[例] 设一个码组A位于O点。若码组A中发生一个错码，则我们可以认为A的位置将移动至以O点为圆心，以1为半径的圆上某点，但其位置不会超出此圆。 若码组A中发生两位错码，则其位置不会超出以O点为圆心，以2为半径的圆。因此，只要最小码距不小于3，码组A发生两位以下错码时，不可能变成另一个准用码组，因而能检测错码的位数等于2。



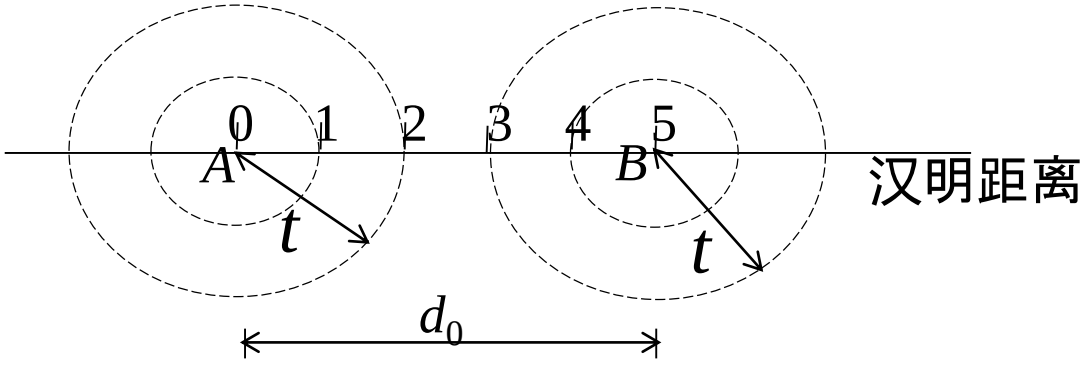
纠错编码的基本原理

- 码距和检纠错能力的关系

  -- **为了纠正t个错码，要求最小码距$d_0 \geq 2t+1$**

[例] 图中画出码组A和B的距离为5。码组A或B若发生不多于两位错码，则其位置均不会超出半径为2以原位置为圆心的圆。这两个圆是不重叠的。判决规则为：若接收码组落于以A为圆心的圆上就判决收到的是码组A，若落于以B为圆心的圆上就判决为码组B。

这样，就能够纠正两位错码。

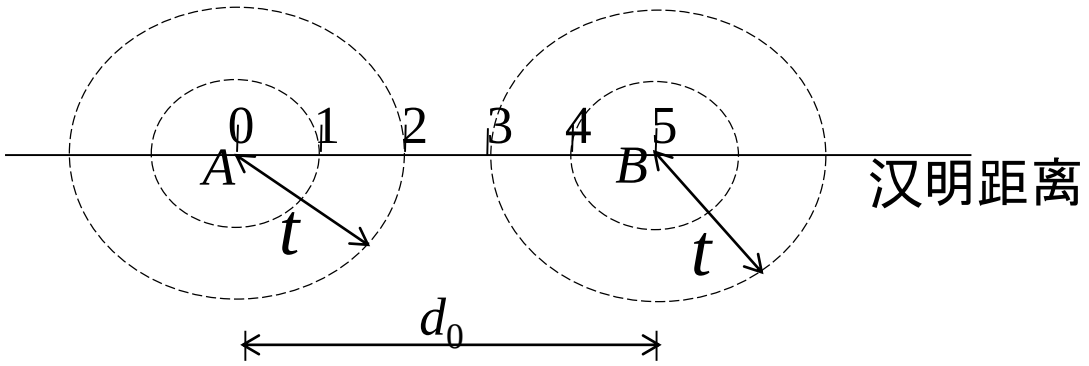


纠错编码的基本原理

- **为纠正t个错码，同时检测e个错码，要求最小码距**

$$d_0 \geq e+t+1 \qquad (e>t)$$

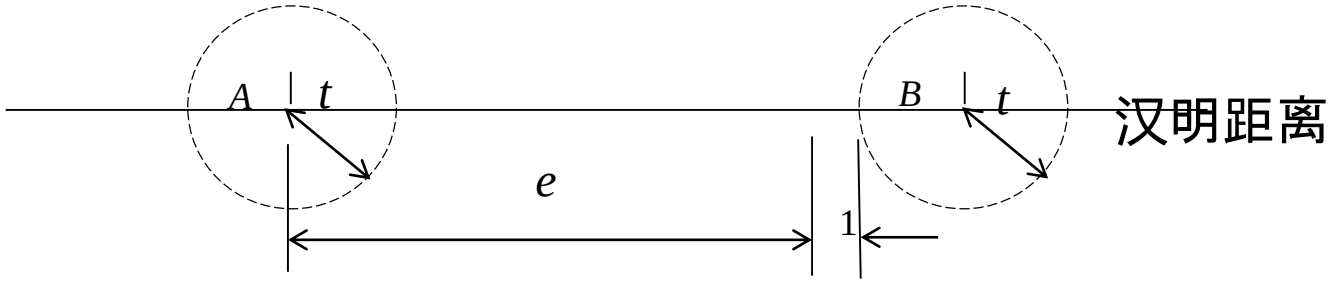
[例] 图中码组A和B之间距离为5。按照检错能力公式，最多能检测4个错码，即$e = d_0 - 1 = 5 - 1 = 4$，按照纠错能力公式纠错时，能纠正2个错码。但是，不能同时作到两者，因为当错码位数超过纠错能力时，该码组立即进入另一码组的圆内而被错误地“纠正”了。例如，码组A若错了3位，就会被误认为码组B错了2位造成的结果，从而被错“纠”为B。这就是说，**检错和纠错公式不能同时成立**或同时运用。



纠错编码的基本原理

所以，为了在可以纠正t个错码的同时，能够检测e个错码，就需要像下图所示那样，使某一码组（譬如码组A）发生e个错误之后所处的位置，与其他码组（譬如码组B）的纠错圆圈至少距离等于1，不然将落在该纠错圆上从而发生错误地“纠正”。因此，由此图可以直观看出，要求最小码距

$$d_0 \geq e+t+1 \qquad (e>t)$$



这种纠错和检错结合的工作方式简称**纠检结合**。

纠错编码的基本原理

- 这种工作方式是**自动在纠错和检错之间转换**的
  - 当错码数量少时，系统按前向纠错方式工作，以节省重发时间，提高传输效率；
  - 当错码数量多时，系统按反馈重发方式纠错，以降低系统的总误码率。
  - 所以，它适用于大多数时间中错码数量很少，少数时间中错码数量多的情况。

汉明码

- 编码原理
  - 码长为n，信息位数为k，监督位数$r=n-k$。如果希望用r个监督位构造出r个监督关系式来指示1位错码的n种可能位置，则要求 $2^r-1\geq n$ 或 $2^r\geq k+r+1$

…	12	11	10	9	8	7	6	5	4	3	2	1	位数
…	I8	I7	I6	I5		I4	I3	I2		I1			信息位
…					P_4				P_3		P_2	P_1	校验位

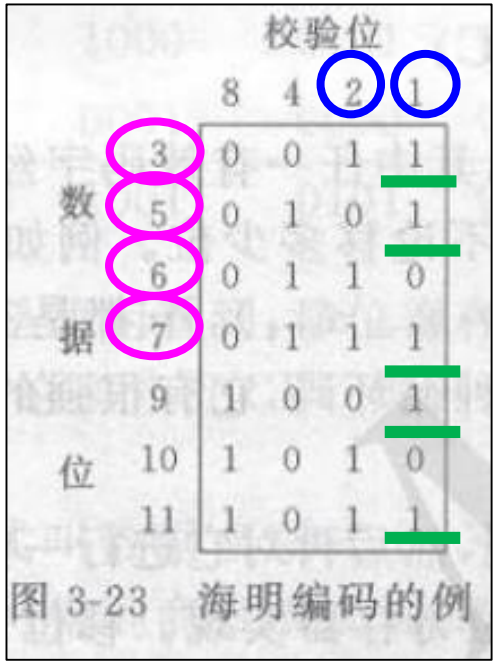


图 3-23 海明编码的例

- P1位负责校验海明码的第1、3、5、7、…（P1、D1、D2、D4、…）位，（包括P1自己）
- P2负责校验海明码的第2、3、6、7、…（P2、D1、D3、D4、…）位，（包括P2自己）
- P3负责校验海明码的第4、5、6、7、…（P3、D2、D3、D4、…）位，（包括P3自己）

汉明码

- 假设二进制代码为0101（位数n=4），根据$2^k \geq n+k+1$可以得出k的值是3，所以汉明码应为n+k=7位。

序号	7	6	5	4	3	2	1
名称	C4	C3	C2	P3	C1	P2	P1
传送数	0	1	0	1	1	0	1

如果按照配偶原则来配置汉明码，则P1应使1 3 5 7位中“1”的个数为偶数；P2应使2 3 6 7位中“1”的个数为偶数；P3应使4 5 6 7位中“1”的个数为偶数。

P1=③位+⑤位+⑦位，即P1=C1+C2+C4=1+0+0=1
P2=③位+⑥位+⑦位，即P2=C1+C3+C4=1+1+0=0
P3=⑤位+⑥位+⑦位，即P3=C2+C3+C4=0+1+0=1

汉明码

- 检错纠错，配置偶校验

序号	7	6	5	4	3	2	1
名称	C4	C3	C2	P3	C1	P2	P1
传送数	0	1	1	1	1	0	1

P1=①位+③位+⑤位+⑦位，即P1=1+1+1+0=1
P2=②位+③位+⑥位+⑦位，即P2=0+1+1+0=0
P3=④位+⑤位+⑥位+⑦位，即P3=1+1+1+0=1

P3P3P1=101=5

可以纠正1个比特错误

线性分组码一般原理

- 监督矩阵

$a_6 a_5 a_4 a_3 \; a_2 a_1 a_0$

$$\Rightarrow \begin{cases} a_2 = a_6 \oplus a_5 \oplus a_4 \\ a_1 = a_6 \oplus a_5 \oplus a_3 \\ a_0 = a_6 \oplus a_4 \oplus a_3 \end{cases} \Rightarrow$$

$$\begin{bmatrix} 1110100 \\ 1101010 \\ 1011001 \end{bmatrix} \begin{bmatrix} a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

（模2）$\Leftrightarrow$

$$\begin{cases} a_6 \oplus a_5 \oplus a_4 \oplus a_2 = 0 \\ a_6 \oplus a_5 \oplus a_3 \oplus a_1 = 0 \\ a_6 \oplus a_4 \oplus a_3 \oplus a_0 = 0 \end{cases}$$

简记为 $\boldsymbol{H} \cdot \boldsymbol{A}^{\mathrm{T}} = \boldsymbol{0}^{\mathrm{T}}$，

$$\boldsymbol{H} = \begin{bmatrix} 1110100 \\ 1101010 \\ 1011001 \end{bmatrix}$$

将H称为监督矩阵

Parity Check matrix

线性分组码一般原理

- 生成矩阵

$$\begin{cases} a_2 = a_6 \oplus a_5 \oplus a_4 \\ a_1 = a_6 \oplus a_5 \oplus a_3 \\ a_0 = a_6 \oplus a_4 \oplus a_3 \end{cases} \Rightarrow \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 1110 \\ 1101 \\ 1011 \end{bmatrix} \begin{bmatrix} a_6 \\ a_5 \\ a_4 \\ a_3 \end{bmatrix} = P \begin{bmatrix} a_6 \\ a_5 \\ a_4 \\ a_3 \end{bmatrix}$$

$$\Leftrightarrow \quad [a_2 a_1 a_0] = [a_6 a_5 a_4 a_3] \cdot \begin{bmatrix} 111 \\ 110 \\ 101 \\ 011 \end{bmatrix} = [a_6 a_5 a_4 a_3] \boldsymbol{Q}$$

$$\Leftrightarrow \quad [a_6 a_5 a_4 a_3 a_2 a_1 a_0] = [a_6 a_5 a_4 a_3] \cdot \boldsymbol{G} \qquad \boldsymbol{G} = [I_k \boldsymbol{Q}] = \left[\begin{array}{c:c} 1000 & 111 \\ 0100 & 110 \\ 0010 & 101 \\ 0001 & 011 \end{array}\right]$$

G 称为生成矩阵

线性分组码一般原理

- 译码
  - 若设接收码组为一n列的行矩阵$\boldsymbol{B}$，即 $\boldsymbol{B}=[b_{n-1}b_{n-2}\cdots b_1b_0]$
  - 则接收码组B和发送码组A之差为**错码**行**矩阵**

$$\boldsymbol{B}-\boldsymbol{A}=\boldsymbol{E}\ (\text{模}2)$$

  - 解码时，将$\boldsymbol{B}$当作$\boldsymbol{A}$代入公式$(\boldsymbol{A}\cdot\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{0})$后：

1）在未超过检错能力时，

$$\boldsymbol{S}=\boldsymbol{B}\cdot\boldsymbol{H}^{\mathrm{T}}=(\boldsymbol{A}+\boldsymbol{E})\,\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{A}\cdot\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{E}\cdot\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{E}\cdot\boldsymbol{H}^{\mathrm{T}}$$

***S*和错码*E*之间有确定的线性变换关系**。

若$\boldsymbol{S}$和$\boldsymbol{E}$之间一一对应，则**$\boldsymbol{S}$将能代表错码的位置**。

2）在错码较多，已超过这种编码的检错能力时，B变为另一许用码组，则该式仍能成立。这样的**错码是不可检测的**。

线性分组码的性质

- 一种线性码中的任意两个码组之和仍为这种码中的一个码组。

$$\boldsymbol{A}_1 \cdot \boldsymbol{H}^{\mathrm{T}} = 0, \quad \boldsymbol{A}_2 \cdot \boldsymbol{H}^{\mathrm{T}} = 0$$

$$\boldsymbol{A}_1 \cdot \boldsymbol{H}^{\mathrm{T}} + \boldsymbol{A}_2 \cdot \boldsymbol{H}^{\mathrm{T}} = (\boldsymbol{A}_1 + \boldsymbol{A}_2)\,\boldsymbol{H}^{\mathrm{T}} = 0$$

A_1，A_2是码组，则$(A_1 + A_2)$也是一个码组。

两个码组(A_1和A_2)之间的距离，必定是另一个码组$(A_1 + A_2)$的重量（即“1”的数目）。**因此，码的最小距离就是码的最小重量（除全“0”码组外）**

循环码

- 一个线性分组码，如果它的任一码字左移或右移一位后，得到的仍然是该码的一个码字，这种码称为循环码

一种（7，3）循环码的全部码字

序号	左移次数	信息码元	监督码元		序号	左移次数	信息码元	监督码元
		$a_6\ a_5\ a_4$	$a_3\ a_2\ a_1\ a_0$				$a_6\ a_5\ a_4$	$a_3\ a_2\ a_1 a_0$
0		0 0 0	0 0 0 0		4	6	1 0 0	1 1 1 0
1	0	**0 0 1**	**1 1 0 1**		5	4	1 0 1	0 0 1 1
2	5	0 1 0	0 1 1 1		6	3	1 1 0	1 0 0 1
3	1	0 1 1	1 0 1 0		7	2	1 1 1	0 1 0 0

循环码

码多项式：把码组中各码元当作是一个多项式的系数，即把一个长度为n的码组表示成

$$T(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0$$

x仅是码元位置的标记，我们并不关心x的取值。

(n，k)循环码的每个码字必处在以x^n+1为模运算的剩余类的某一类中。

循环码

- **循环码的生成矩阵**

$$\boldsymbol{A} = [a_6 a_5 a_4 a_3] \cdot \boldsymbol{G}$$

- 如果某一循环码的所有码多项式都是多项式$g(x)$的倍式，则称**$g(x)$为该码的生成多项式**（最低次的码多项式就是生成多项式），$g(x)$具有以下特性：

（1）$g(x)$是一个常数项为1的$r=n-k$次多项式；

（2）$g(x)$是x^n+1的一个因式；

（3）该循环码中其它码多项式都是$g(x)$的倍式。

循环码

- **循环码的生成矩阵**

例：前述(7,3)循环码，最低次码多项式为$x^4+x^3+x^2+1=g(x)$，则其它码多项式和对应码字为：

码多项式	对应码组：
$g(x)=x^4+x^3+x^2+1$	0011101
$xg(x)=x^5+x^4+x^3+x$	0111010
$x^2\cdot g(x)=x^6+x^5+x^4+x^2$	1110100
$x^3\cdot g(x)=x^7+x^6+x^5+x^3=x^6+x^5+x^3+1$	1101001
$x^4\cdot g(x)=x^8+x^7+x^6+x^4=x^6+x^4+x+1$	1010011
$x^5\cdot g(x)=x^9+x^8+x^7+x^5=x^5+x^2+x+1$	0100111
$x^6\cdot g(x)=x^{10}+x^9+x^8+x^6=x^6+x^3+x^2+x$	1001110

由此例也可以看出，能被$g(x)$除尽的次数不大于n-1的多项式，必为码多项式

循环码

- **循环码的生成矩阵**
  - g(x)是x^n+1的一个r次因子
  - 以(7，3)循环码为例（$n=7$，$r=4$），将x^7+1分解得：$x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$
  - $g(x)$可以有二种取法：$g(x)=(x+1)(x^3+x^2+1)=x^4+x^2+x+1$，或$g(x)=(x+1)(x^3+x+1)=x^4+x^3+x^2+1$ **（生成多项式并不是惟一的）**
  - 一旦g(x)确定，则(n，k)循环码的所有码字就确定了。由g(x)左移(乘x^i，i=1,2,…,n-1)就可以产生其它码字的码多项式。

循环码

- **循环码的生成矩阵**
  - g(x)是x^n+1的一个r次因子
  - 以(7，3)循环码为例（$n=7$，$r=4$），将x^7+1分解得：$x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$
  - $g(x)$可以有二种取法：$g(x)=(x+1)(x^3+x^2+1)=x^4+x^2+x+1$，或$g(x)=(x+1)(x^3+x+1)=x^4+x^3+x^2+1$ **（生成多项式并不是惟一的）**
  - 一旦g(x)确定，则(n，k)循环码的所有码字就确定了；用**k个互相独立的码多项式**$g(x)$、$xg(x)$、…、$x^{k-1}g(x)$可以构造出生成矩阵

$$G(x)=\begin{bmatrix} x^{k-1}\cdot g(x) \\ x^{k-2}\cdot g(x) \\ \vdots \\ x\cdot g(x) \\ g(x) \end{bmatrix} \qquad G(x)=\begin{bmatrix} x^2 g(x) \\ x\ g(x) \\ g(x) \end{bmatrix}=\begin{bmatrix} x^6+x^5+x^4+x^2 \\ x^5+x^4+x^3+x \\ x^4+x^3+x^2+1 \end{bmatrix} \qquad G=\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

循环码

- **编码步骤**：

  令$m(x)$为信息码多项式

  1）根据给定的(n, k)值选定生成多项式$g(x)$，
  即从$(x^n + 1)$的因子中选一个$(n - k)$次多项式作为$g(x)$。

  2）用x^{n-k}乘$m(x)$。即在信息码后附加上$(n - k)$个“0”。

  3）$x^{n-k}\, m(x) / g(x)$，得到商 $Q(x)$ 和 余式 $r(x)$

  4）编出的码组$T(x)$为

$$T(x) = x^{n-k}\, m(x) + r(x)$$

- **检错解码原理**：任意一个码组多项式$T(x)$都能被生成多项式$g(x)$**整除**

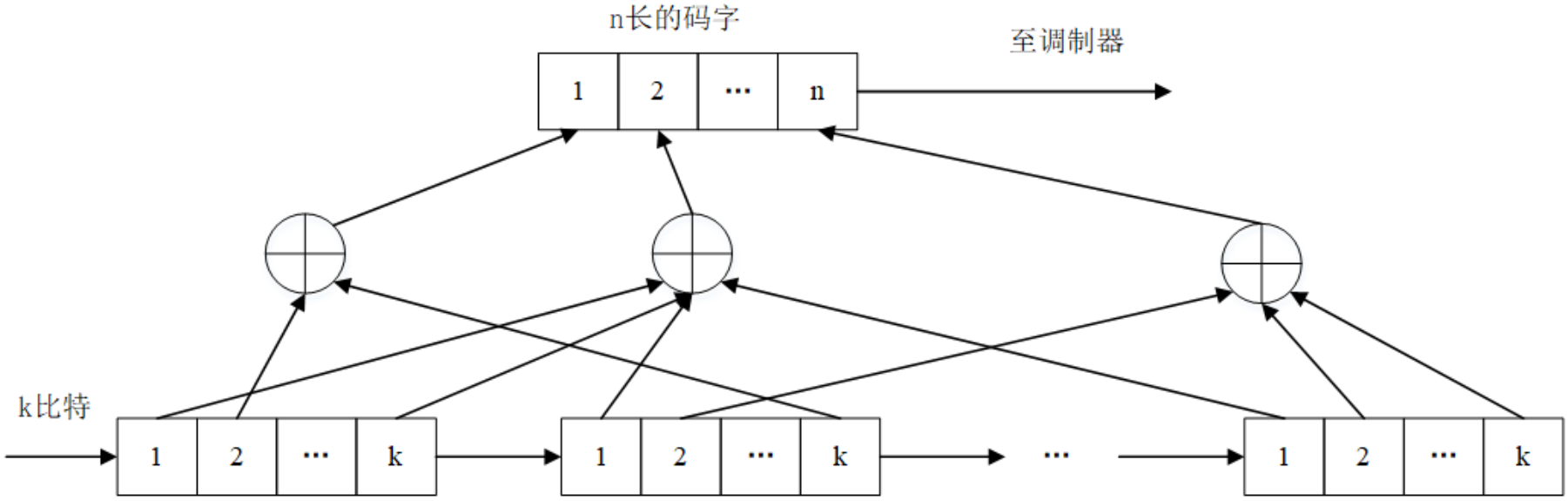
- **纠错解码原理**：按余式$r(x)$，用查表的方法或通过某种计算得到错误图样$E(x)$

卷积码

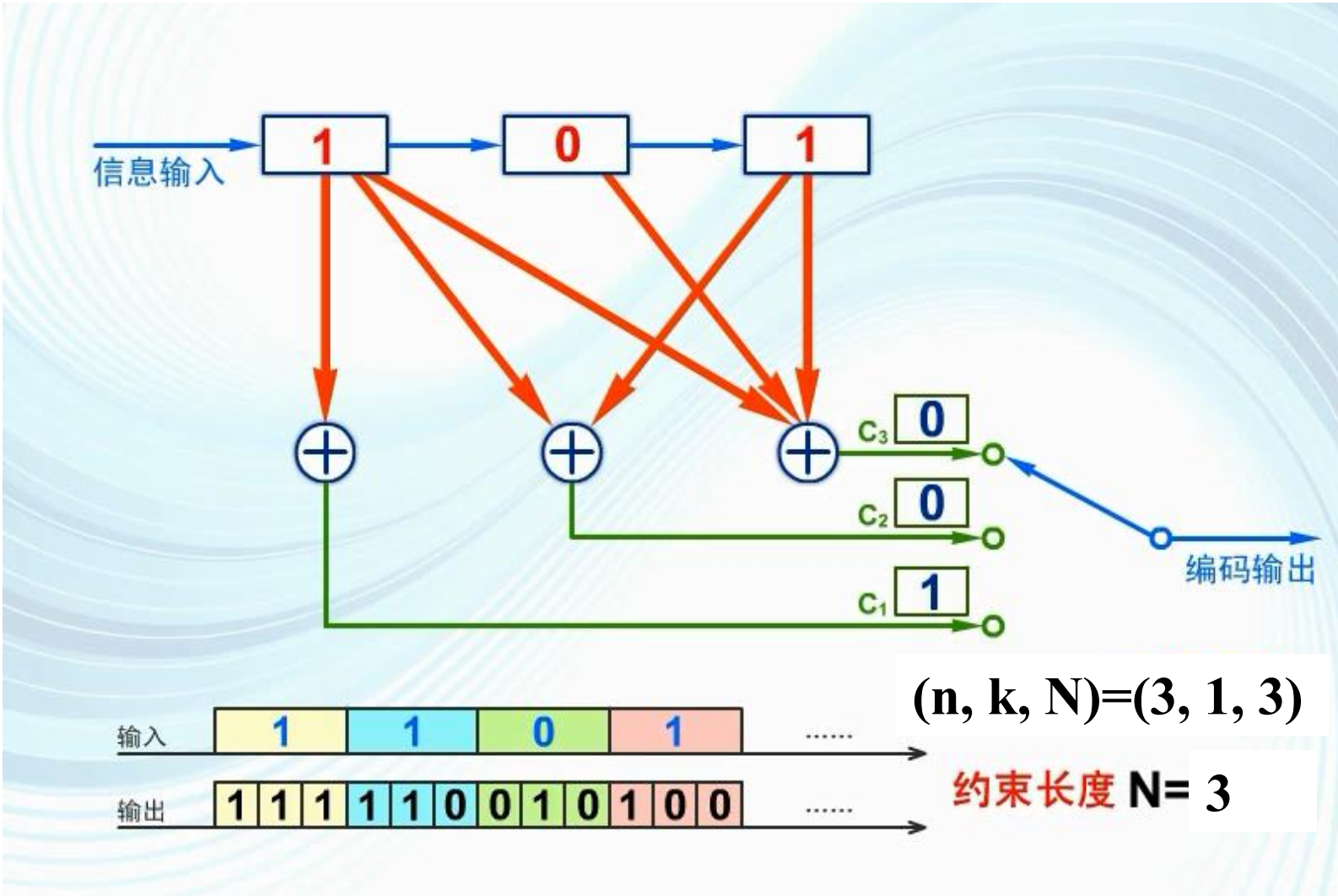
- Peter Elias, 1955; Andrew Viterbi 1967 maximum likelihood decode
- **卷积码是一种非分组码**。
- **卷积码中监督码元不仅和当前的*k*比特信息段有关，而且还同前面*m* = (*N* – 1)个信息段有关**。所以一个码组中的监督码元监督着*N*个信息段。通常将***N*称为编码约束长度**。
- 一般说来，对于卷积码，*k* 和 *n* 的值是比较小的整数。我们将卷积码记作**(*n*, *k*, *N*)**。码率则仍定义为*k* / *n*。

卷积码

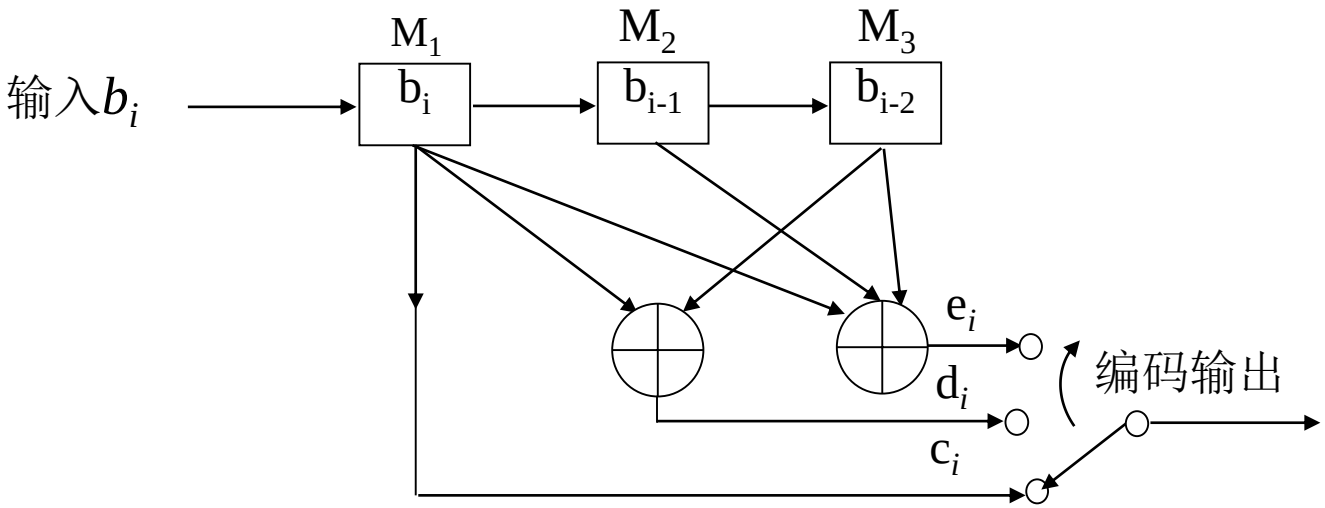
卷积码是一种有记忆的纠错码，编码规则是将k个信息比特编码形成n个比特，编码后的**n个码元**不但与当前输入的**k个信息**有关，仍与**之前的N-1组**的信息有关。



卷积码


信息输入
1
0
1
C3 0
C2 0
C1 1
编码输出
(n, k, N)=(3, 1, 3)
输入
1 1 0 1 .....
输出
1 1 1 1 1 0 0 1 0 1 0 0 .....
约束长度 N= 3

卷积码



$$c_i = b_i$$

$$d_1 = b_1$$

$$e_1 = b_1$$

$$d_2 = b_2$$

$$e_2 = b_2 + b_1$$

$$d_3 = b_3 + b_1$$

$$e_3 = b_3 + b_2 + b_1$$

$$d_4 = b_4 + b_2$$

$$e_4 = b_4 + b_3 + b_2$$

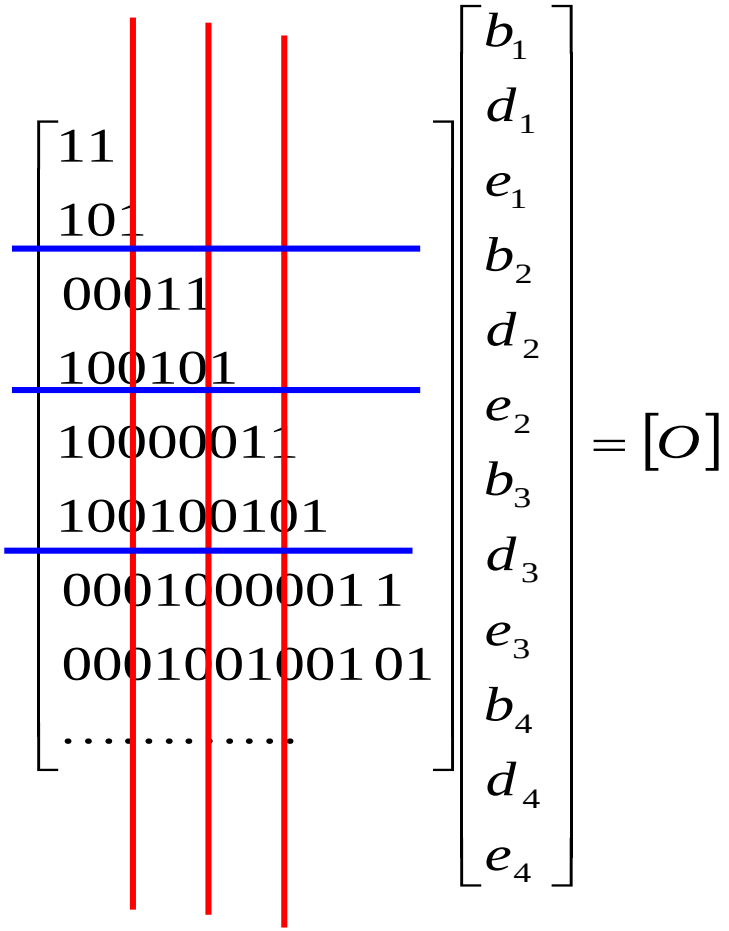
$$\cdots \quad \cdots \quad \cdots$$

监督位d_i、e_i和信息位b_i之间的关系

(n, k, N)=(3, 1, 3)

卷积码

$\boldsymbol{H}\cdot\boldsymbol{A}^{\mathrm{T}}=\boldsymbol{0}^{\mathrm{T}}$



(n, k, N)=(3, 1, 3)

截短**监督矩阵**的结构形式

$$\boldsymbol{H}=\begin{bmatrix} & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \end{bmatrix}$$

$$\boldsymbol{H}_1=\begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{I}_{n-k} & & & & & & & & \\ \boldsymbol{P}_2 & \boldsymbol{O}_{n-k} & \boldsymbol{P}_1 & \boldsymbol{I}_{n-k} & & & & & & \\ \boldsymbol{P}_3 & \boldsymbol{O}_{n-k} & \boldsymbol{P}_2 & \boldsymbol{O}_{n-k} & \boldsymbol{P}_1 & \boldsymbol{I}_{n-k} & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & & \\ \boldsymbol{P}_N & \boldsymbol{O}_{n-k} & \boldsymbol{P}_{N-1} & \boldsymbol{O}_{n-k} & \boldsymbol{P}_{N-2} & \boldsymbol{O}_{n-k} & \cdots & \boldsymbol{P}_1 & \boldsymbol{I}_{n-k} \end{bmatrix}$$

式中 $\boldsymbol{I}_{n-k}$ — $(n-k)$阶单位方阵；

$\boldsymbol{P}_i$ — $k\times(n-k)$阶矩阵；

$\boldsymbol{O}_{n-k}$ — $(n-k)$阶全零方阵。

卷积码

一般说来，**生成矩阵**具有如下形式：

$$G_1 = \begin{bmatrix} I_k & Q_1 & O_k & Q_2 & O_k & Q_3 & \cdots & O_k & Q_N \\ & & I_k & Q_1 & O_k & Q_2 & \cdots & O_k & Q_{N-1} \\ & & & & I_k & Q_1 & \cdots & O_k & Q_{N-2} \\ & & & & & & \cdots & \vdots & \\ & & & & & & & I_k & Q_1 \end{bmatrix}$$

I_k - k阶单位方阵；

Q_i - $(n-k)\times k$阶矩阵；

O_k - k阶全零方阵。

- 将上式中矩阵第一行称为基本生成矩阵 $g = [I_k\ Q_1\ O_k\ Q_2\ O_k\ Q_3 \ldots O_k\ Q_N]$

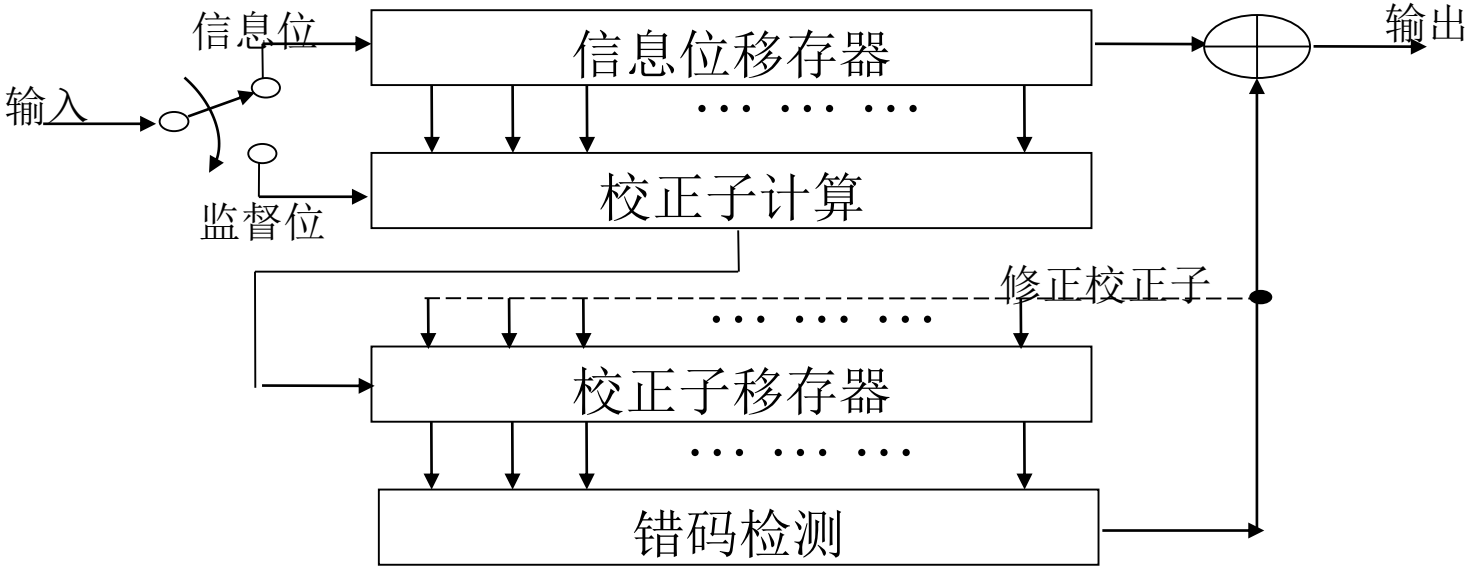
- 如果基本生成矩阵g已经给定，则可以从已知的信息位得到整个编码序列。

卷积码

- 卷积码的解码
  - 代数解码：利用编码本身的代数结构进行解码，不考虑信道的统计特性。
  - 概率解码：又称最大似然解码。它基于信道的统计特性和卷积码的特点进行计算。**维特比算法**。当码的约束长度较短时，它比序贯解码算法的效率更高、速度更快，目前得到广泛的应用。

卷积码

- 卷积码的解码
  - 代数解码：利用编码本身的代数结构进行解码，不考虑信道的统计特性。



大数逻辑解码，又称门限解码，是卷积码代数解码的最主要一种方法，它也可以应用于循环码的解码。

差错控制编码

- **状态图**

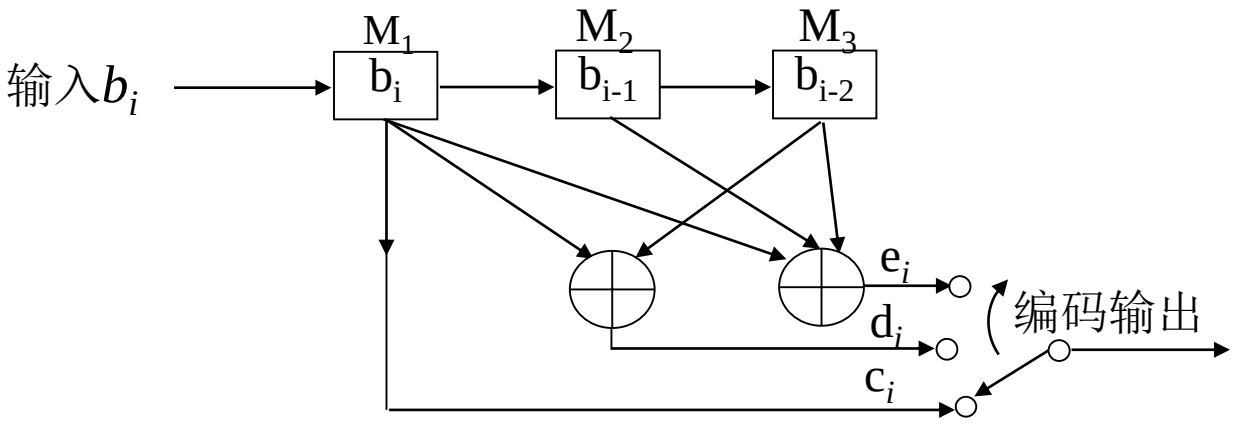
由上例的编码器结构可知，输出码元c_i d_i e_i决定于当前输入信息位b_i和前两位信息位b_{i-1}和b_{i-2}（即移存器M_2和M_3的状态）。在上图中已经为M_2和M_3的4种状态规定了代表符号a，b，c 和d。所以，可以将当前输入信息位、移存器前一状态、移存器下一状态和输出码元之间的关系归纳于下表中。

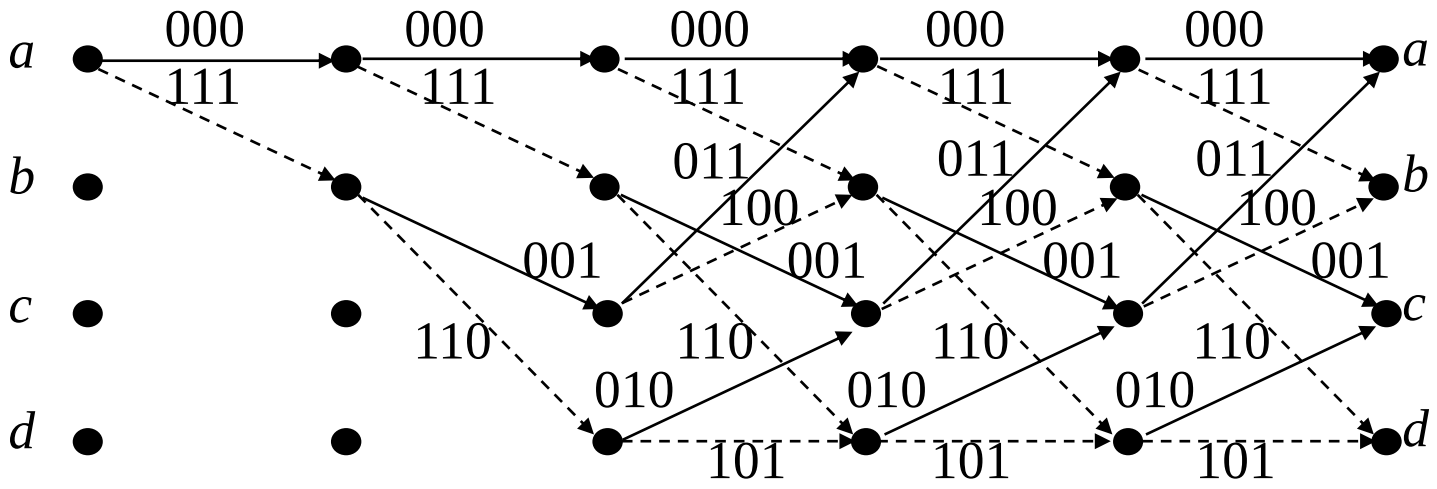
差错控制编码

移存器前一状态 $M_3\ M_2$	当前输入信息位 b_i	输出码元 $c_i d_i e_i$	移存器下一状态 $M_3\ M_2$
a (00)	0 1	000 111	*a* (00) *b* (01)
b (01)	0 1	001 110	*c* (10) *d* (11)
c (10)	0 1	011 100	*a* (00) *b* (01)
d (11)	0 1	010 101	*c* (10) *d* (11)

- 由上表看出，前一状态*a*只能转到下一状态*a*或*b*，前一状态*b*只能转到下一状态*c*或*d*，等等。
- 按照此表中的规律，可以画出状态图如下图所示。

卷积码网格图


输入b_i
M_1
b_i
M_2
b_{i-1}
M_3
b_{i-2}
e_i
d_i
c_i
编码输出


a
b
c
d
000
111
011
100
001
110
010
101
a
b
c
d

维特比解码

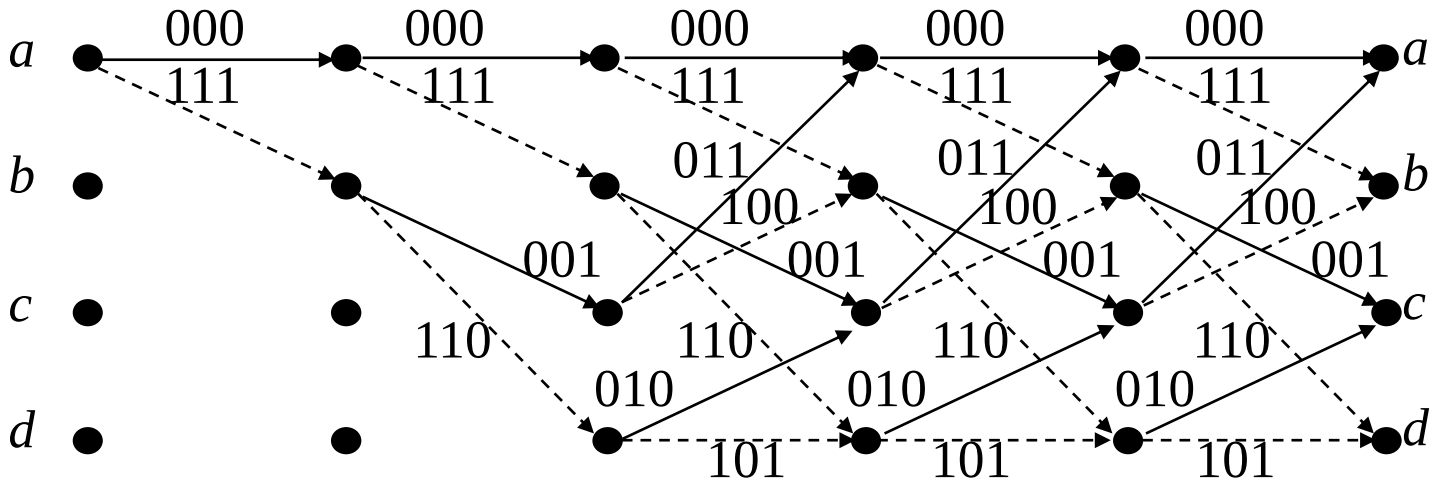
- 基本原理
  - **将接收到的信号序列和所有可能的发送信号序列比较**，选择其中**汉明距离最小的序列**认为是当前发送信号序列。
  - 若发送一个k位序列，则有2^k种可能的发送序列。计算机应存储这些序列，以便用作比较。
  - 当k较大时，存储量太大，使实用受到限制。维特比算法对此作了简化，使之能够实用。

维特比解码

- 例子
  - 发送序列 111 110 010 100 001 011 000
  - 接收序列 111 010 010 110 001 011 000

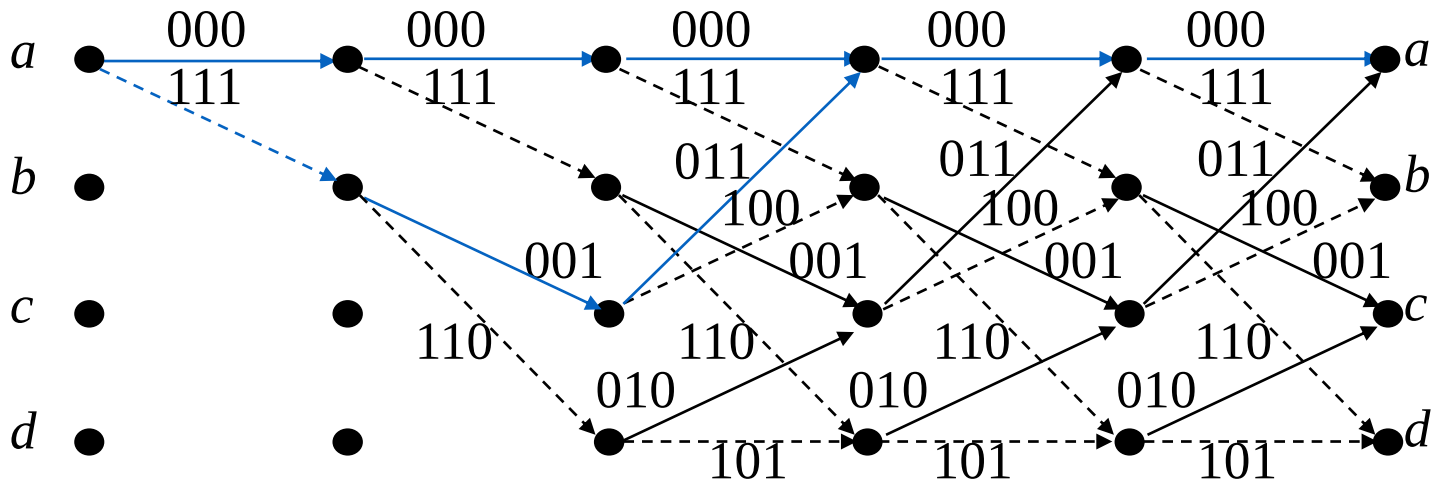
发送信息为1101000

(n, k, N)=(3, 1, 3)



维特比解码

- 例子
  - 发送序列 111 110 010 100 001 011 000
  - 接收序列 111 010 010 110 001 011 000



维特比解码

- 例子
  - 发送序列 111 110 010 100 001 011 000
  - 接收序列 111 010 010 110 001 011 000

序号	路径	对应序列	汉明距离	幸存否
1	*aaaa*	000 000 000	5	否
2	*abca*	111 001 011	3	是
3	*aaab*	000 000 111	6	否
4	*abcb*	111 001 100	4	是
5	*aabc*	000 111 001	7	否
6	*abdc*	111 110 010	1	是
7	*aabd*	000 111 110	6	否
8	*abdd*	111 110 101	4	是

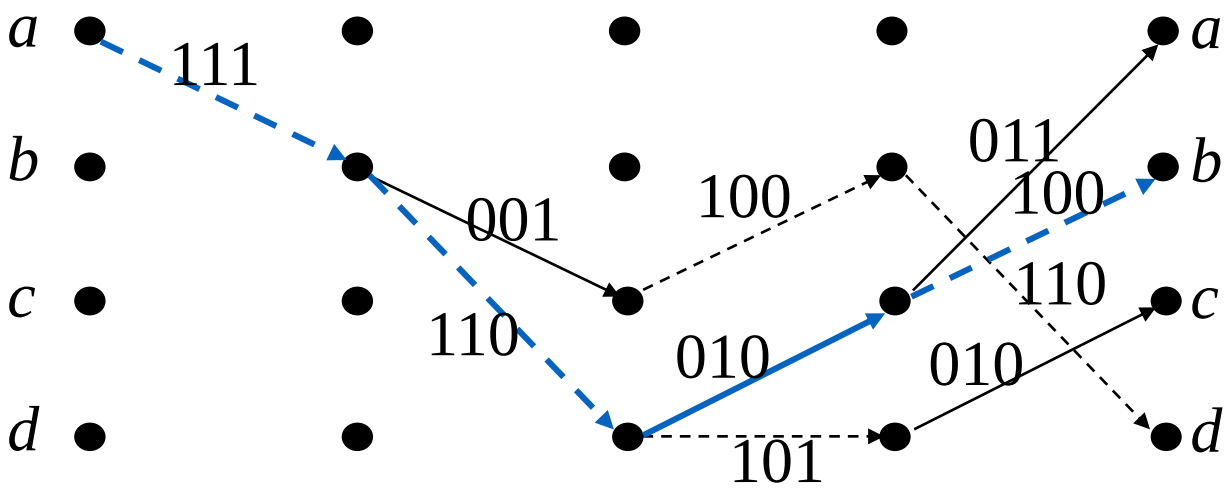
维特比解码

- 例子
  - 发送序列 111 110 010 100 001 011 000
  - 接收序列 111 010 010 110 001 011 000

序号	路径	原幸存路径的距离	新增路径段	新增距离	总距离	幸存否
1	*abca+a*	3	*aa*	2	5	否
2	*abdc+a*	1	*ca*	2	3	是
3	*abca+b*	3	*ab*	1	4	否
4	*abdc+b*	1	*cb*	1	2	是
5	*abcb+c*	4	*bc*	3	7	否
6	*abdd+c*	4	*dc*	1	5	是
7	*abcb+d*	4	*bd*	0	4	是
8	*abdd+d*	4	*dd*	2	6	否

维特比解码

- 例子
  - 发送序列 111 110 010 100 001 011 000
  - 接收序列 111 010 010 110 001 011 000



图中路径是汉明距离最小（等于2）的路径。

课程思考/测试题

课程相关问题（辅助理解，无需提交，不计成绩）：

2 简述奇偶校验方法。

3 检错码与纠错码有什么不同?各有什么优点?

4 CRC 计算可以用软件实现，但多数是使用硬件来实现的，为什么?

5. 在数据传输过程中，若接收方收到的二进制比特序列为 10110011010，接收双方采用的生成多项式为 $G(x)=x^4+x^3+1$，则该二进制比特序列在传输中是否出现了差错?如果没有出现差错，发送数据的比特序列和 CRC 校验码的比特序列分别是什么?

6. 要发送的数据比特序列为 1010001101，CRC 校验生成多项式为 $G(x)=x^5+x^4+x^2+1$，试计算 CRC 校验码。

7. 对于 10 比特要传输的数据，如果采用海明码(Hamming code)校验，需要增加的冗余信息为多少比特?

课程习题（作业）

课本（截止日期：4.21日晚23:55）：

6.2； 6.5； 6.6； 6.10； 6.13； 6.15； 6.17

提交方式：http://cslabcms.nju.edu.cn

- 请大家尽量提交word文件，方便系统查重。
  （命名：学号+姓名+第*章）
- 若提交遇到问题请及时发邮件或在下一次上课时反馈。

课程习题（作业）

6.2 你认为在每个字符中包含一个奇偶校验比特是否会改变接收到正确报文的概率？

6.3 两个正在通信的设备用1比特的偶校验进行差错检测。发送方发送了字节10101010，由于信道噪声，接收方收到的字节是10011010。接收方能否检测到这个差错？为什么？

6.4 设想一个帧由两个4比特长的字符组成。假设比特差错率为10^{-3}，且每个比特彼此无关。

a. 接收到的帧含有至少一个差错的概率为多少？

b. 再给每个字符增加一个奇偶校验比特。这时的概率又是多少？

6.5 请指出图6.3(a)中的p值不管是作为所有数据比特的奇偶校验比特，还是所有行奇偶校验比特的奇偶校验比特，或是所有列奇偶校验比特的奇偶校验比特，结果都是一样的。

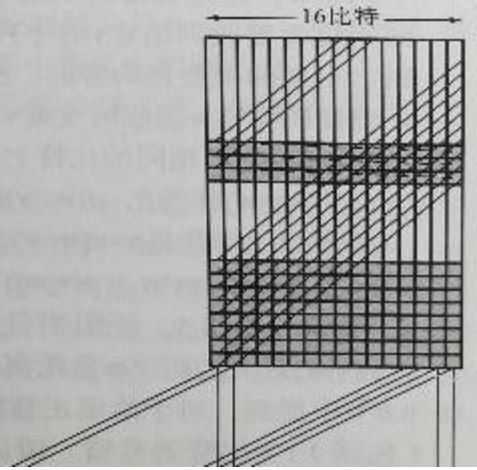
6.6 为数据块E3 4F 23 96 44 27 99 F3计算其因特网检验和。再进行验证计算。

6.7 因特网检验和的一个很好的特点是字节顺序无关性。小端字节序（little endian）的计算机在存储十六进制数时，最低有效字节在最后（如Intel处理器）。而大端字节序（big endian）的计算机将最低有效字节放在最前面（如IBM大型机）。请解释为什么检验和不需要考虑字节顺序？

6.8 高速运输协议（Xpress Transfer Protocol，XTP）使用了一个32比特的检验和函数，它被定义为两个16位函数的组合（concatenation）：XOR和RXOR，如图6.10所示。XOR函数计算列的奇偶校验。RXOR是一种对角奇偶校验，它先旋转数据块的每个连续16比特字中的1比特，然后再执行按位异或运算。

a. 这个检验和能检测出所有由奇数个差错比特引起的错误吗？请解释。

b. 这个检验和能检测出所有由偶数个差错比特引起的错误吗？如果不能，请给出将会导致检验失败的差错模式的特征。



6.9 在计算FCS时，使用模2算法而不是二进制算法的意义是什么？

6.10 使用CRC-CCITT多项式，为一个1后面跟有15个0的报文计算生成的16比特CRC码。

a. 使用长除法。

b. 使用图6.4所示的移位寄存器机制。

课程习题（作业）

6.13 某CRC的结构可用于为11比特报文生成4比特的FCS。生成多项式为X^4+X^3+1。

a. 画出能够实现这一任务的移位寄存器电路（见图6.4）。

b. 用这个生成多项式将比特序列10011011100（最左边的是最低位）编码，并写出码字。

c. 现在假设在这个码字中的第7个比特（从最低位数起）有差错，请指出差错检测算法是如何检测到这个差错的。

6.14 a. 在CRC差错检测机制中，选择$P(X)=X^4+X+1$。请为比特序列10010011011编码。

b. 假设因信道带来的差错模式为100010000000000（也就是分别在位置1和5从1跳变到0或从0跳变到1）。那么接收到的比特序列是什么？这个差错能被检测出来吗？

c. 如果差错模式为100110000000000，重复问题(b)。

6.15 在通信标准中经常使用的是一种经修改的CRC过程。它的定义如下：

$$\frac{X^{16}D(X)+X^kL(X)}{P(X)}=Q+\frac{R(X)}{P(X)}$$

$$\text{FCS}=L(X)+R(X)$$

其中，

$$L(X)=X^{15}+X^{14}+X^{13}+\cdots+X+1$$

并且k是被检验的比特数（地址字段、控制字段以及信息字段）。

a. 用语言描述这个过程的效果。

b. 解释其可能带来的好处。

c. 画出实现$P(X)=X^{16}+X^{12}+X^5+1$的移位寄存器电路。

课程习题（作业）

6.16 计算以下码字两两之间的汉明距离：

a. 00000, 10101, 01010

b. 000000, 010101, 101010, 110110

6.17 6.6节讨论了以最小距离作为选择依据的块纠错码。也就是说，假设某编码由s个等可能性的码字组成，每个码字的长度为n，那么对每个接收到的序列$\mathbf{v}$，接收器会为它选择一个码字$\mathbf{w}$，使距离$d(\mathbf{w}, \mathbf{v})$最小。我们希望证明这种机制从下述角度来看是“理想”的，即当接收器收到给定$\mathbf{v}$的序列时，该序列为码字$\mathbf{w}$的概率$p(\mathbf{w}|\mathbf{v})$最大。因为我们假设所有码字出现的机会是均等的，所以使$p(\mathbf{w}|\mathbf{v})$最大的码字与使$p(\mathbf{v}|\mathbf{w})$最大的码字是一致的。

a. 要使码字在$\mathbf{w}$接收时变成$\mathbf{v}$，必须在传输中产生恰好$d(\mathbf{w}, \mathbf{v})$个差错。这些差错必须发生在$\mathbf{w}$和$\mathbf{v}$中不相同的比特上。假设$\beta$是特定比特在传输中出错的概率，$n$是码字的长度，写出$p(\mathbf{v}|\mathbf{w})$作为$\beta$，$d(\mathbf{w}, \mathbf{v})$和$n$的函数表达式。提示：差错的比特数是$d(\mathbf{w}, \mathbf{v})$，而没有差错的比特数是$n-d(\mathbf{w}, \mathbf{v})$。

b. 再通过计算$p(\mathbf{v}|\mathbf{w}_1)/p(\mathbf{v}|\mathbf{w}_2)$比较两个不同的码字$\mathbf{w}_1$，$\mathbf{w}_2$的$p(\mathbf{v}|\mathbf{w}_1)$和$p(\mathbf{v}|\mathbf{w}_2)$。

c. 假设$0 < \beta < 0.5$，证明当且仅当$d(\mathbf{v}, \mathbf{w}_1) < d(\mathbf{v}, \mathbf{w}_2)$时有$p(\mathbf{v}|\mathbf{w}_1) > p(\mathbf{v}|\mathbf{w}_2)$。这就证明了使$p(\mathbf{v}|\mathbf{w})$最大的码字$\mathbf{w}$是距离$\mathbf{v}$最小的码字。

总结

问题？

殷亚凤
yafeng@nju.edu.cn
http://cs.nju.edu.cn/yafeng/
Room 301, Building of CS