通信原理

第11章 差错控制编码

第11章差错控制编码

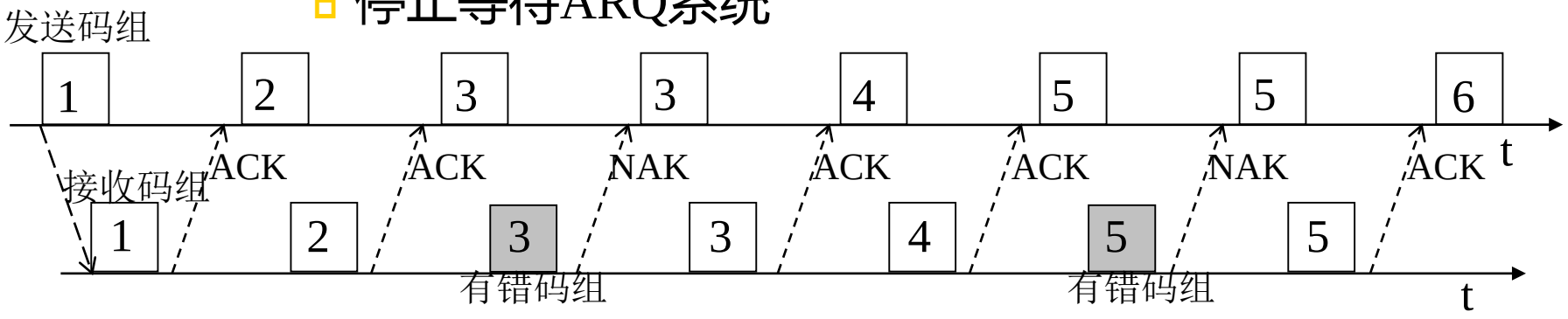
- **11.1 概述**
  - 信道分类：从差错控制角度看
    - 随机信道：错码的出现是随机的
    - 突发信道：错码是成串集中出现的
    - 混合信道：既存在随机错码又存在突发错码
  - 差错控制技术的种类
    - 检错重发
    - 前向纠错
    - 反馈校验
    - 检错删除

第11章差错控制编码

- 差错控制编码：常称为**纠错编码**
  - **监督码元：**上述4种技术中除第3种外，都是在接收端识别有无错码。所以在发送端需要在信息码元序列中增加一些差错控制码元，它们称为监督码元。
  - 不同的编码方法，有不同的**检错**或**纠错**能力。
  - **多余度**：就是指增加的监督码元多少。例如，若编码序列中平均每两个信息码元就添加一个监督码元，则这种编码的多余度为1/3。
  - **编码效率**(简称**码率**)：设编码序列中信息码元数量为k，总码元数量为n，则比值k/n就是码率。
  - **冗余度：**监督码元数$(n-k)$和信息码元数k之比。
  - 理论上，差错控制以降低信息传输速率为代价换取提高传输可靠性。

第11章差错控制编码

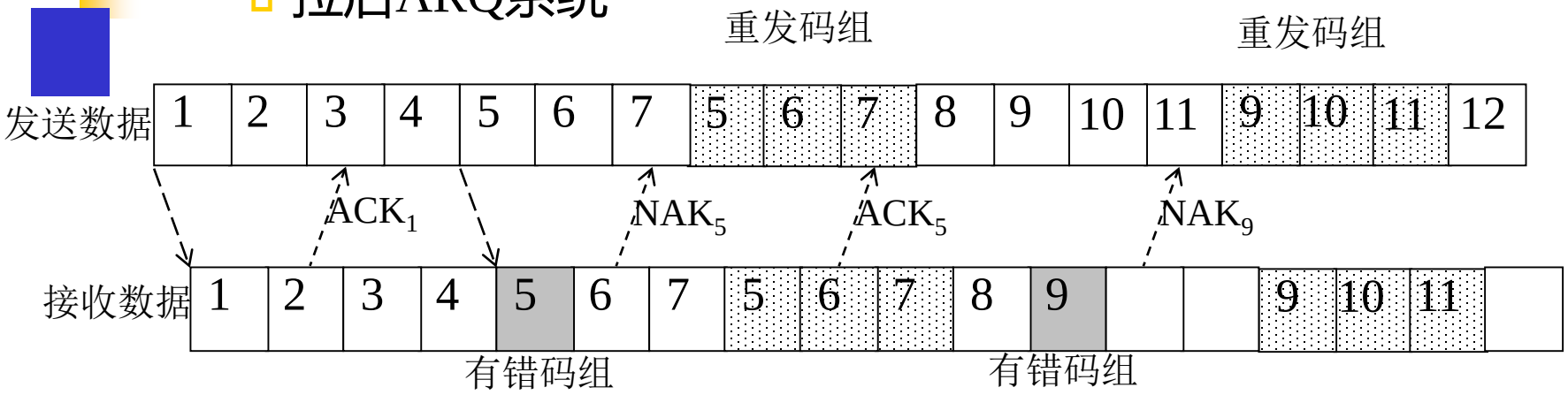
- 自动要求重发(ARQ)系统
  - 3种ARQ系统
    - 停止等待ARQ系统



- 数据按分组发送。每发送一组数据后发送端等待接收端的确认(ACK)答复，然后再发送下一组数据。
- 图中的第3组接收数据有误，接收端发回一个否认(NAK)答复。这时，发送端将重发第3组数据。
- 系统是工作在半双工状态，时间没有得到充分利用，传输效率较低。

第11章差错控制编码

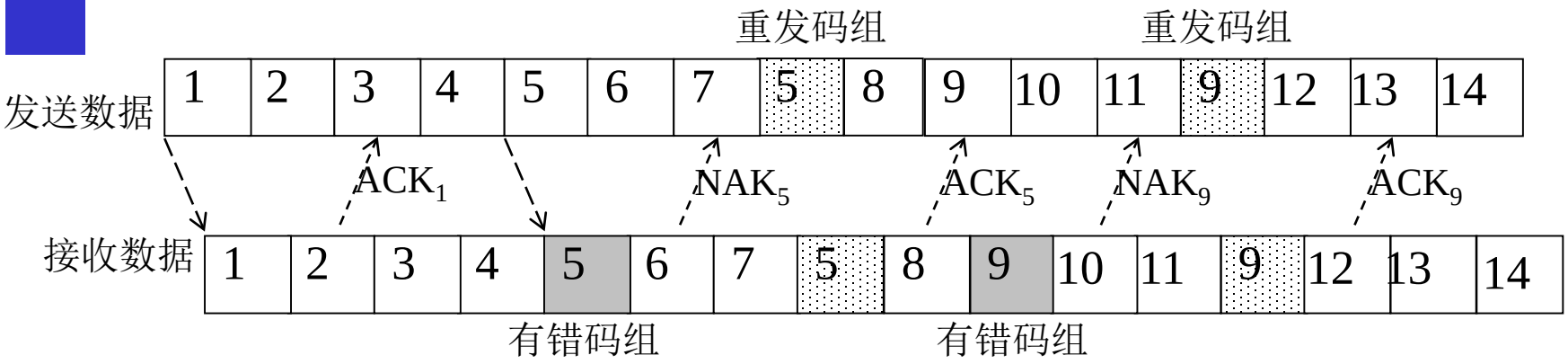
- 拉后ARQ系统



- 发送端连续发送数据组，接收端对于每个接收到的数据组都发回**确认(ACK)**或**否认(NAK)**答复。
- 例如，图中第5组接收数据有误，则在发送端收到第5组接收的否认答复后，从第5组开始重发数据组。
- 在这种系统中需要对发送的数据组和答复进行编号，以便识别。显然，这种系统需要双工信道

第11章差错控制编码

- 选择重发ARQ系统



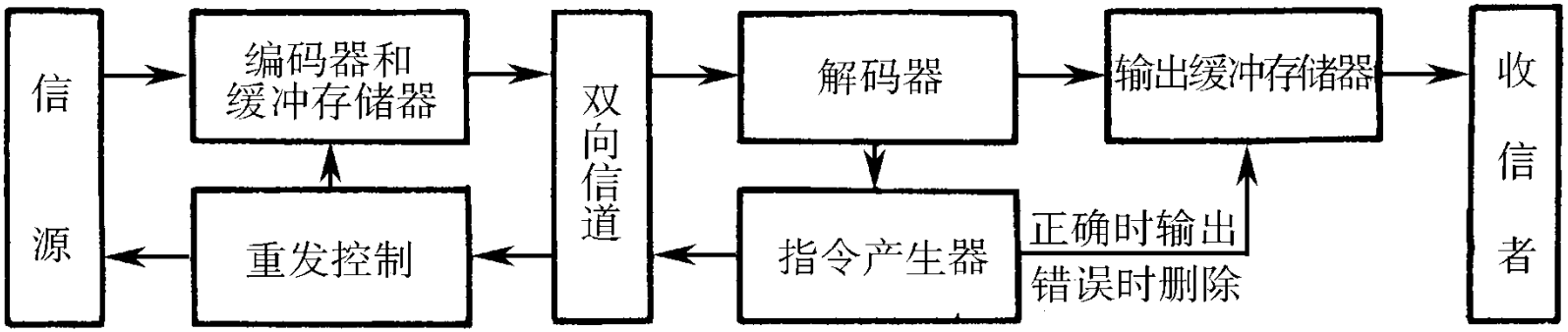
- 它只重发出错的数据组，因此进一步提高了传输效率。

第11章差错控制编码

- ARQ的主要优点：和前向纠错方法相比
  - 监督码元较少即能使误码率降到很低，即码率较高；
  - 检错的计算复杂度较低；
  - 检错用的编码方法和加性干扰的统计特性基本无关，能适应不同特性的信道。
- ARQ的主要缺点：
  - 需要双向信道来重发，不能用于单向信道，也不能用于一点到多点的通信系统。
  - 因为重发而使ARQ系统的传输效率降低。
  - 在信道干扰严重时，可能发生因不断反复重发而造成事实上的通信中断。
  - 在要求实时通信的场合，例如电话通信，往往不允许使用ARQ法。

第11章差错控制编码

- ARQ系统的原理方框图



- 在发送端，输入的信息码元在编码器中被分组编码（加入监督码元）后，除了立即发送外，还暂存于缓冲存储器中。若接收端解码器检出错码，则由解码器控制产生一个重发指令。此指令经过反向信道送到发送端。由发送端重发控制器控制缓冲存储器重发一次。
- 接收端仅当解码器认为接收信息码元正确时，才将信息码元送给收信者，否则在输出缓冲存储器中删除接收码元。
- 当解码器未发现错码时，经过反向信道发出不需重发指令。发送端收到此指令后，即继续发送后一码组，发送端的缓冲存储器中的内容也随之更新。

第11章差错控制编码

11.2 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。

第11章差错控制编码

- 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**。

第11章差错控制编码

- 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。

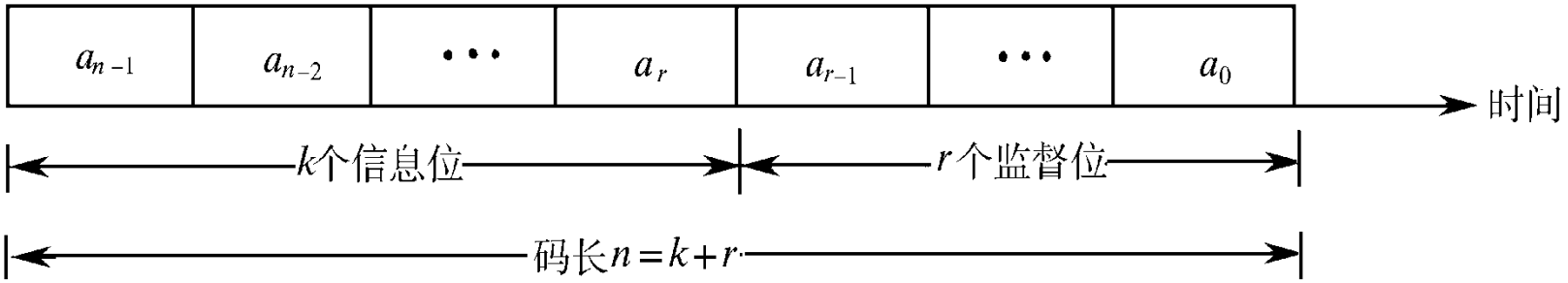
第11章差错控制编码

- 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：举例如下

	信息位	监督位
晴	00	0
云	01	1
阴	10	1
雨	11	0

第11章差错控制编码

- 分组码的一般结构



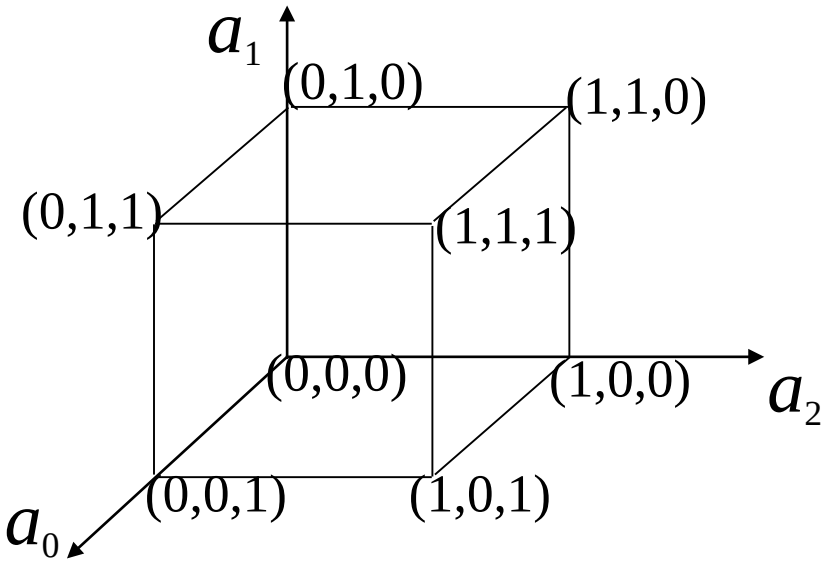
- 分组码的符号：(n, k)
  - N - 码组的总位数，又称为码组的长度（码长），
  - k - 码组中信息码元的数目，
  - $n-k=r$ - 码组中的监督码元数目，或称监督位数目。

第11章差错控制编码

- 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$d_0 = 2$。

第11章差错控制编码

- 码距的几何意义



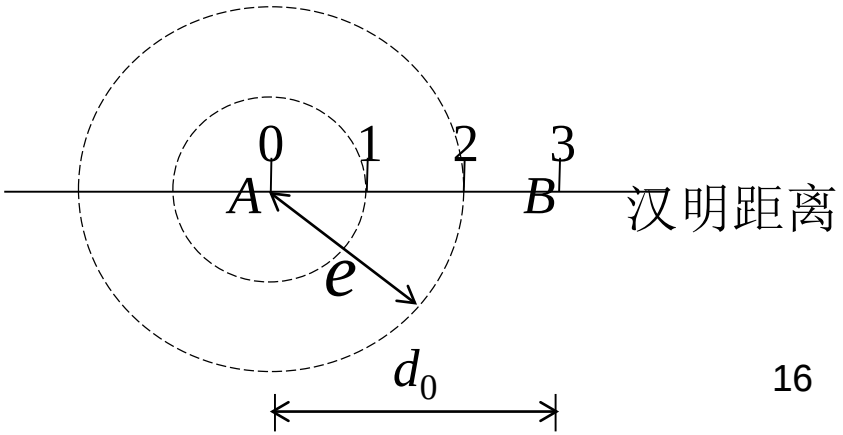
- 对于3位的编码组，可以在3维空间中说明码距的几何意义。
- 每个码组的3个码元的值(a_1, a_2, a_3)就是此立方体各顶点的坐标。而上述码距概念在此图中就对应于各顶点之间沿立方体各边行走的几何距离。
- 由此图可以直观看出，上例中4个准用码组之间的距离均为2。

第11章差错控制编码

- 码距和检纠错能力的关系
  - 一种编码的最小码距d_0的大小直接关系着这种编码的检错和纠错能力
  - 为检测e个错码，要求最小码距 $d_0 \geq e+1$

    【证】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。



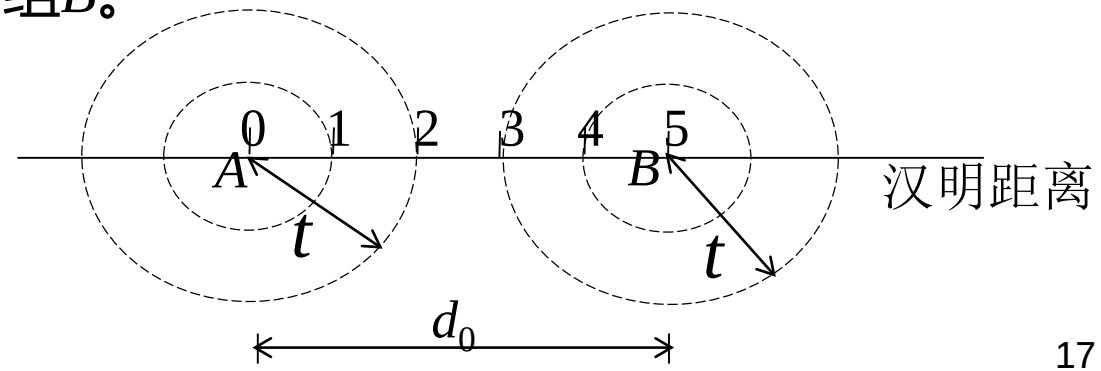
第11章差错控制编码

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$(e + 1)$。

- 为了纠正t个错码，要求最小码距$d_0 \geq 2t + 1$

【证】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。



第11章差错控制编码

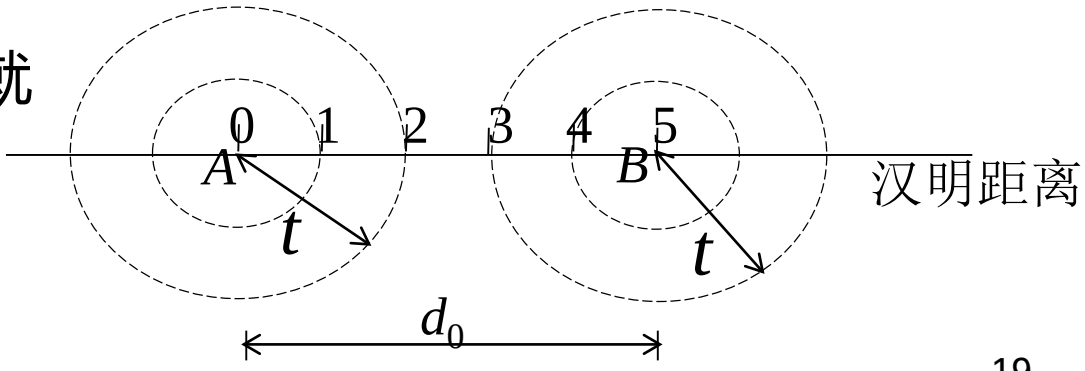
若这种编码中除码组A和B外，还有许多种不同码组，但任两码组之间的码距均不小于5，则以各码组的位置为中心以2为半径画出之圆都不会互相重叠。这样，每种码组如果发生不超过两位错码都将能被纠正。因此，当最小码距$d_0 = 5$时，能够纠正2个错码，且最多能纠正2个。若错码达到3个，就将落入另一圆上，从而发生错判。故一般说来，为纠正t个错码，最小码距应不小于$(2t + 1)$。

第11章差错控制编码

- 为纠正t个错码，同时检测e个错码，要求最小码距

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

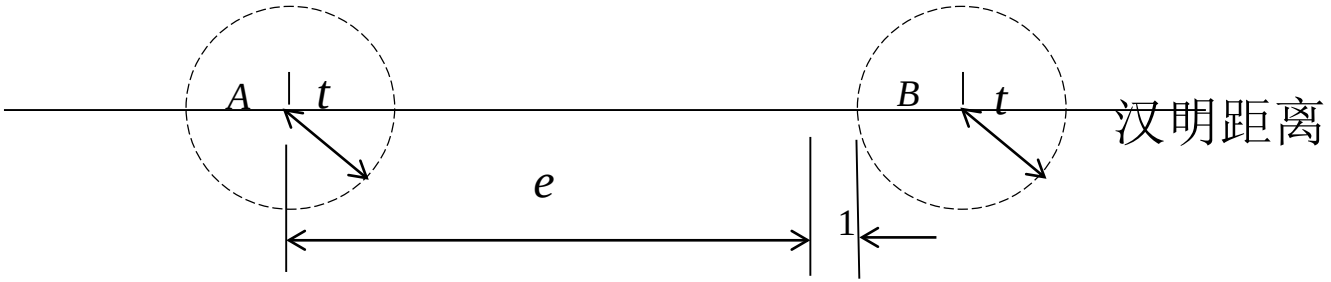
在解释此式之前，先来分析下图所示的例子。图中码组A和B之间距离为5。按照检错能力公式，最多能检测4个错码，即$e = d_0 - 1 = 5 - 1 = 4$，按照纠错能力公式纠错时，能纠正2个错码。但是，不能同时作到两者，因为当错码位数超过纠错能力时，该码组立即进入另一码组的圆内而被错误地“纠正”了。例如，码组A若错了3位，就会被误认为码组B错了2位造成的结果，从而被错“纠”为B。这就是说，检错和纠错公式不能同时成立或同时运用。



第11章差错控制编码

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**。

第11章差错控制编码

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。

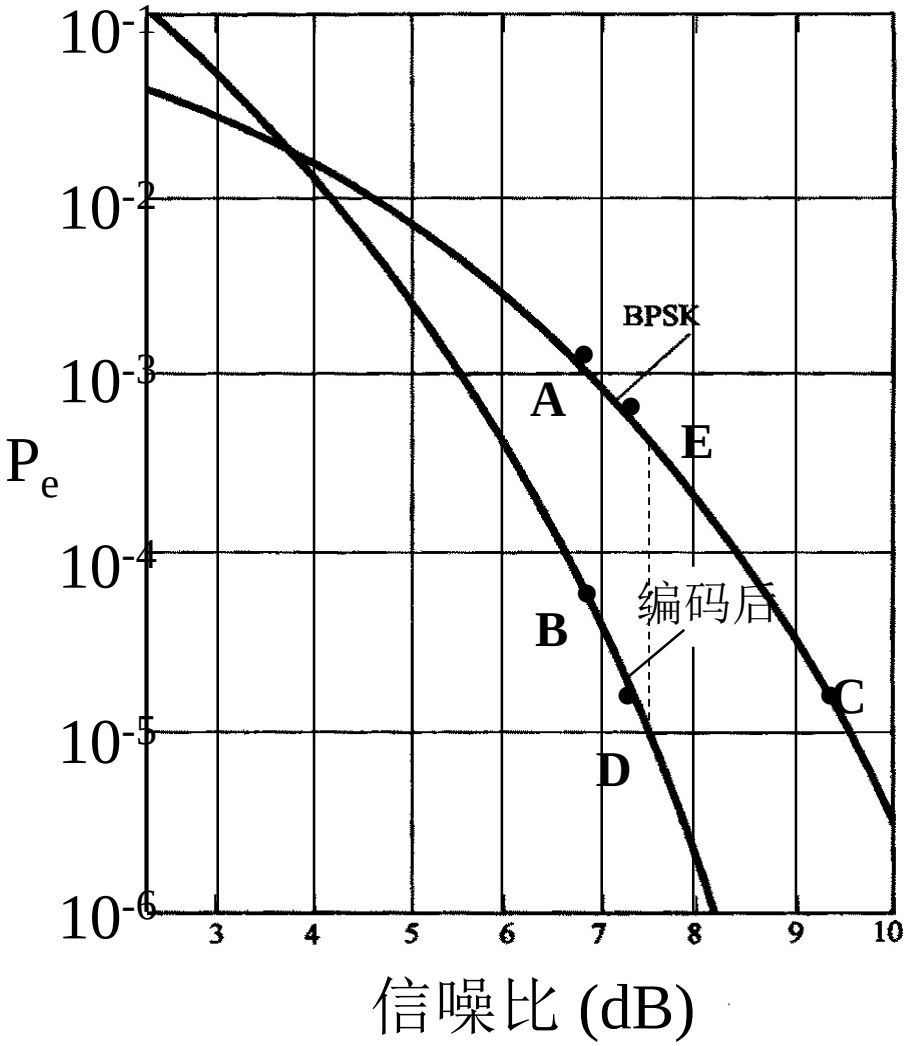
第11章差错控制编码

11.3 纠错编码的性能

- 系统带宽和信噪比的矛盾：
  - 由上节所述的纠错编码原理可知，为了减少接收错误码元数量，需要在发送信息码元序列中加入监督码元。这样作的结果使发送序列增长，冗余度增大。若仍须保持发送信息码元速率不变，则传输速率必须增大，因而增大了系统带宽。系统带宽的增大将引起系统中噪声功率增大，使信噪比下降。信噪比的下降反而又使系统接收码元序列中的错码增多。一般说来，采用纠错编码后，误码率总是能够得到很大改善的。改善的程度和所用的编码有关。

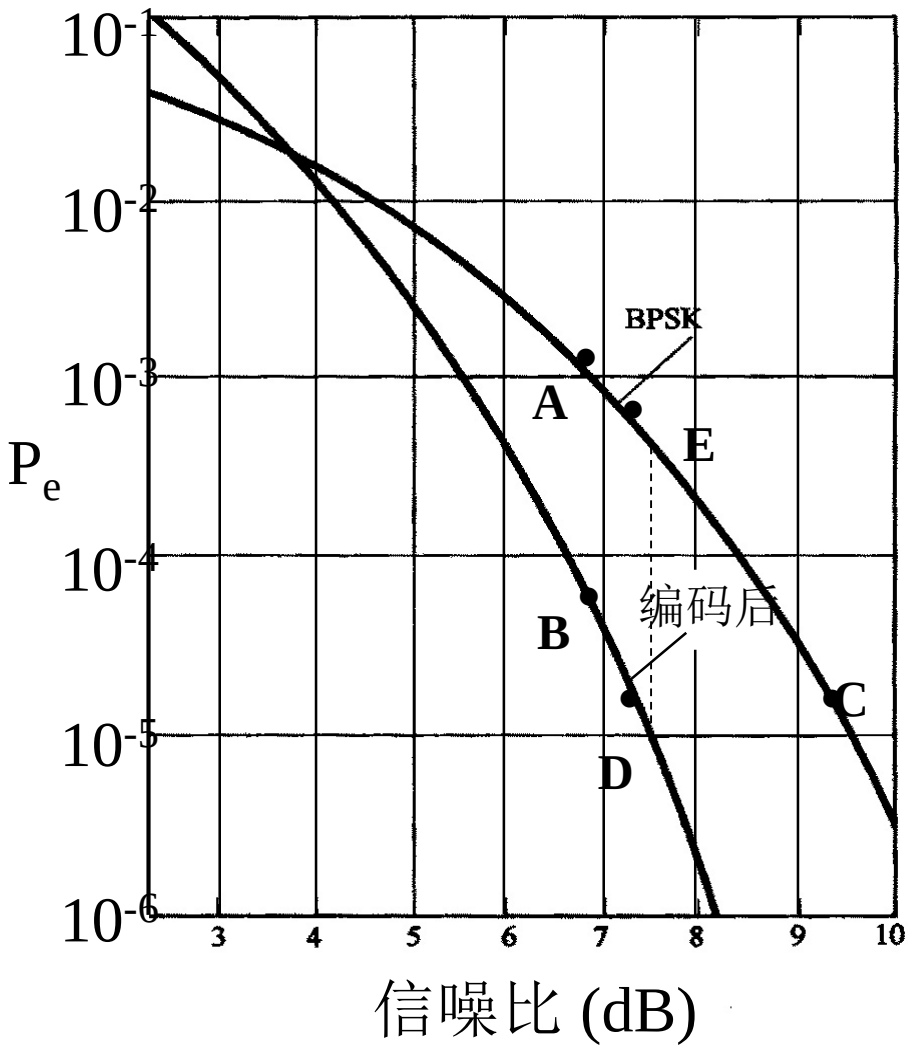
第11章差错控制编码

- 编码性能举例
  - 未采用纠错编码时，若接收信噪比等于7dB，编码前误码率约为8×10^{-4}，图中A点，在采用纠错编码后，误码率降至约4×10^{-5}，图中B点。这样，不增大发送功率 就能降低误码率约一个半数量级。



第11章差错控制编码

- 由图还可以看出，若保持误码率在10^{-5}，图中C点，未采用编码时，约需要信噪比$E_b / n_0 = 10.5$ dB。在采用这种编码时，约需要信噪比7.5 dB，图中D点。可以节省功率2 dB。通常称这2 dB为编码增益。
- 上面两种情况付出的代价是带宽增大。



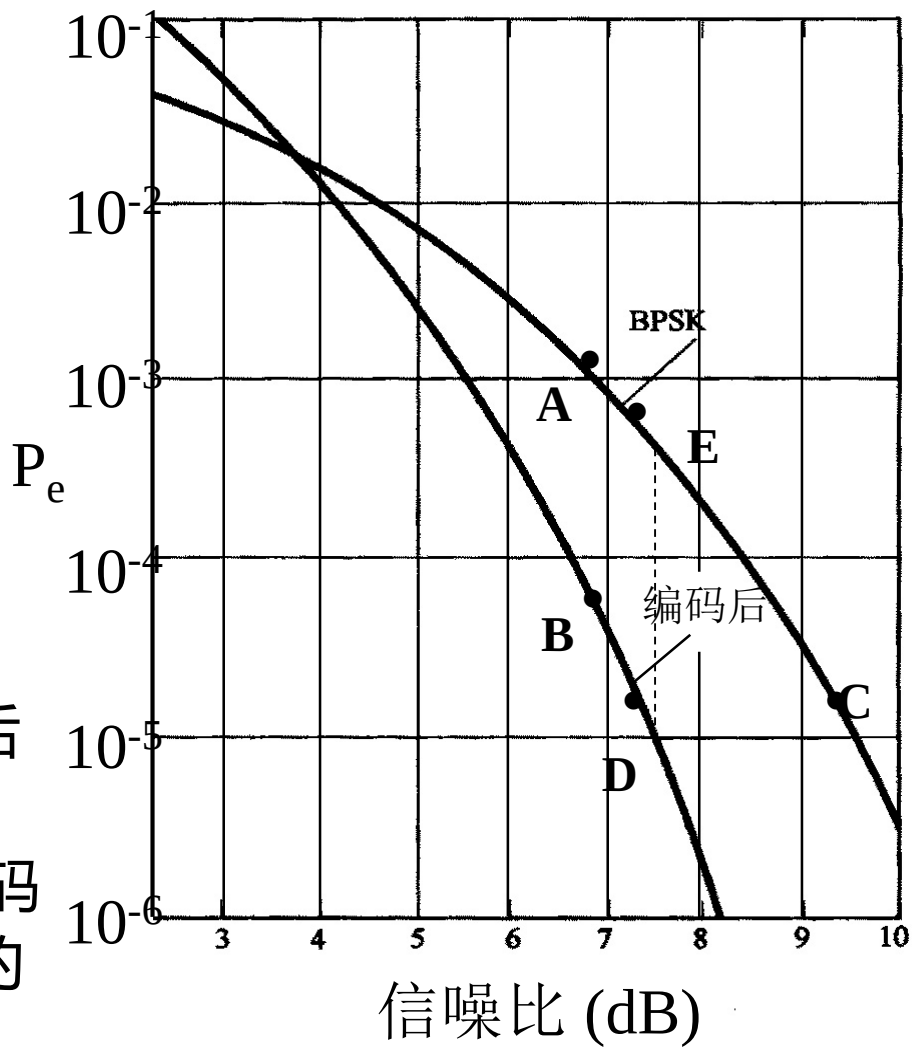
第11章差错控制编码

- 传输速率和E_b/n_0的关系

对于给定的传输系统

$$\frac{E_b}{n_0}=\frac{P_sT}{n_0}=\frac{P_s}{n_0(1/T)}=\frac{P_s}{n_0R_B}$$

式中，R_B为码元速率。若希望提高传输速率，由上式看出势必使信噪比下降，误码率增大。假设系统原来工作在图中C点，提高速率后由C点升到E点。但加用纠错编码后，仍可将误码率降到D点。这时付出的代价仍是带宽增大。



第11章差错控制编码

11.4简单的实用编码

11.4.1 奇偶监督码

- 奇偶监督码分为奇数监督码和偶数监督码两种，两者的原理相同。在偶数监督码中，无论信息位多少，监督位只有1位，它使码组中“1”的数目为偶数，即满足下式条件：

$$a_{n-1} \oplus a_{n-2} \oplus \cdots \oplus a_0 = 0$$

式中a_0为监督位，其他位为信息位。

这种编码能够检测奇数个错码。在接收端，按照上式求“模2和”，若计算结果为“1”就说明存在错码，结果为“0”就认为无错码。

奇数监督码与偶数监督码相似，只不过其码组中“1”的数目为奇数：

$$a_{n-1} \oplus a_{n-2} \oplus \cdots \oplus a_0 = 1$$

第11章差错控制编码

11.4.2 二维奇偶监督码（方阵码）

二维奇偶监督码的构成

它是先把上述奇偶监督码的若干码组排成矩阵，每一码组写成一行，然后再按列的方向增加第二维监督位，如下图所示

$$
\begin{array}{cccccc}
a_{n-1}^{1} & a_{n-2}^{1} & \cdots & a_{1}^{1} & a_{0}^{1} \\
a_{n-1}^{2} & a_{n-2}^{2} & \cdots & a_{1}^{2} & a_{0}^{2} \\
\cdots & \cdots & & \cdots & \\
a_{n-1}^{m} & a_{n-2}^{m} & \cdots & a_{1}^{m} & a_{0}^{m} \\
c_{n-1} & c_{n-2} & \cdots & c_{1} & c_{0}
\end{array}
$$

图中$a_0^1\ a_0^2\ \ldots\ a_0^m$为$m$行奇偶监督码中的$m$个监督位。

$c_{n-1}\ c_{n-2}\ \ldots\ c_1\ c_0$为按列进行第二次编码所增加的监督位，它们构成了一监督位行。

第11章差错控制编码

- 二维奇偶监督码的性能
  - 这种编码有可能检测偶数个错码。因为每行的监督位虽然不能用于检测本行中的偶数个错码，但按列的方向有可能由$c_{n-1}\ c_{n-2}\ \cdots\ c_1\ c_0$等监督位检测出来。有一些偶数错码不可能检测出来。例如，构成矩形的4个错码，譬如图中

    $$a_{n-2}^{2} \quad a_{1}^{2} \quad a_{n-2}^{m} \quad a_{1}^{m}$$

    错了，就检测不出。
  - 这种二维奇偶监督码适于检测突发错码。因为突发错码常常成串出现，随后有较长一段无错区间。
  - 由于方阵码只对构成矩形四角的错码无法检测，故其检错能力较强。
  - 二维奇偶监督码不仅可用来检错，还可以用来纠正一些错码。 例如，仅在一行中有奇数个错码时。

第11章差错控制编码

11.4.3 恒比码

- 在恒比码中，每个码组均含有相同数目的“1”（和“0”）。由于“1”的数目与“0”的数目之比保持恒定，故得此名。
- 这种码在检测时，只要计算接收码组中“1”的数目是否对，就知道有无错码。
- 恒比码的主要优点是简单和适于用来传输电传机或其他键盘设备产生的字母和符号。对于信源来的二进制随机数字序列，这种码就不适合使用了。

第11章差错控制编码

11.4.4 正反码

- 正反码的编码：
  - 它是一种简单的能够纠正错码的编码。其中的监督位数目与信息位数目相同，监督码元与信息码元相同或者相反则由信息码中“1”的个数而定。
  - 例如，若码长$n = 10$，其中信息位 $k = 5$，监督位 $r = 5$。其编码规则为：
    - 当信息位中有奇数个“1”时，监督位是信息位的简单重复；
    - 当信息位有偶数个“1”时，监督位是信息位的反码。
    - 例如，若信息位为11001，则码组为1100111001；若信息位为10001，则码组为1000101110。

第11章差错控制编码

- 正反码的解码
  - 在上例中，先将接收码组中信息位和监督位按模 2 相加，得到一个5位的合成码组。然后，由此合成码组产生一个校验码组。
  - 若接收码组的信息位中有奇数个“1”，则合成码组就是校验码组；若接收码组的信息位中有偶数个“1”，则取合成码组的反码作为校验码组。
  - 最后，观察校验码组中“1”的个数，按下表进行判决及纠正可能发现的错码。

第11章差错控制编码

- 校验码组和错码的关系

	校验码组的组成	错码情况
1	全为“0”	无错码
2	有4个“1”和1个“0”	信息码中有1位错码，其位置对应校验码组中“0”的位置
3	有4个“0”和1个“1”	监督码中有1位错码，其位置对应校验码组中“1”的位置
4	其他组成	错码多于1个

例如，若发送码组为1100111001，接收码组中无错码，则合成码组应为11001⊕11001=00000。由于接收码组信息位中有奇数个“1”，所以校验码组就是00000。按上表判决，结论是无错码。

第11章差错控制编码

若传输中产生了差错，使接收码组变成1000111001，则合成码组为10001⊕11001 = 01000。由于接收码组中信息位有偶数个“1”，所以校验码组应取合成码组的反码，即10111。由于其中有4个“1”和1个“0”，按上表判断信息位中左边第2位为错码。

若接收码组错成1100101001，则合成码组变成11001⊕01001 = 10000。由于接收码组中信息位有奇数个“1”，故校验码组就是10000，按上表判断，监督位中第1位为错码。

最后，若接收码组为1001111001，则合成码组为10011⊕11001 = 01010，校验码组与其相同，按上表判断，这时错码多于1个。

- 上述长度为10的正反码具有纠正1位错码的能力，并能检测全部2位以下的错码和大部分2位以上的错码。

第11章差错控制编码

11.5 线性分组码

- 基本概念
  - **代数码**：建立在代数学基础上的编码。
  - **线性码**：按照一组线性方程构成的代数码。在线性码中信息位和监督位是由一些线性代数方程联系着的。
  - **线性分组码**：按照一组线性方程构成的分组码 。

本节将以汉明码为例引入线性分组码的一般原理。

第11章差错控制编码

- **汉明码**
  - ~能够纠正1位错码且编码效率较高的一种线性分组码
  - 汉明码的构造原理。
    - 在偶数监督码中，由于使用了一位监督位a_0，它和信息位$a_{n-1} \cdots a_1$一起构成一个代数式：

$$a_{n-1} \oplus a_{n-2} \oplus \cdots \oplus a_0 = 0$$

在接收端解码时，实际上就是在计算

$$S = a_{n-1} \oplus a_{n-2} \oplus \cdots \oplus a_0$$

若$S = 0$，就认为无错码；若$S = 1$，就认为有错码。现将上式称为**监督关系式**，S称为**校正子**。由于校正子S只有两种取值，故它只能代表有错和无错这两种信息，而不能指出错码的位置。

第11章差错控制编码

- 若监督位增加一位，即变成两位，则能增加一个类似的监督关系式。由于两个校正子的可能值有4中组合： 00，01，10，11，故能表示4种不同的信息。若用其中1种组合表示无错，则其余3种组合就有可能用来指示一个错码的3种不同位置。同理，r个监督关系式能指示1位错码的$(2r-1)$个可能位置。
- 一般来说，若码长为n，信息位数为k，则监督位数$r=n-k$。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

$$2^r-1\geq n \quad \text{或} \quad 2^r\geq k+r+1$$

下面通过一个例子来说明如何具体构造这些监督关系式。

第11章差错控制编码

- 例：设**分组码(*n*, *k*)**中$k = 4$，为了纠正1位错码，由上式可知，要求监督位数 $r \geq 3$。若取 $r = 3$，则$n = k + r = 7$。我们用a_6 $a_5 \ldots a_0$表示这7个码元，用S_1、S_2和S_3表示3个监督关系式中的校正子，则S_1、S_2和S_3的值与错码位置的对应关系可以规定如下表所列：

$S_1 S_2 S_3$	错码位置	$S_1 S_2 S_3$	错码位置
001	a_0	101	a_4
010	a_1	110	a_5
100	a_2	111	a_6
011	a_3	000	无错码

第11章差错控制编码

由表中规定可见，仅当一位错码的位置在a_2、a_4、a_5或a_6时，校正子S_1为1；否则S_1为零。这就意味着a_2、a_4、a_5和a_6四个码元构成偶数监督关系：

$$S_1 = a_6 \oplus a_5 \oplus a_4 \oplus a_2$$

同理，a_1、a_3、a_5和a_6构成偶数监督关系：

$$S_2 = a_6 \oplus a_5 \oplus a_3 \oplus a_1$$

以及a_0、a_3、a_4和a_6构成偶数监督关系

$$S_3 = a_6 \oplus a_4 \oplus a_3 \oplus a_0$$

第11章差错控制编码

- 在发送端编码时，信息位a_6、a_5、a_4和a_3的值决定于输入信号，因此它们是随机的。监督位a_2、a_1和a_0应根据信息位的取值按监督关系来确定，即监督位应使上3式中S_1、S_2和S_3的值为0（表示编成的码组中应无错码）：

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

上式经过移项运算，解出监督位

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

给定信息位后，可以直接按上式算出监督位， 结果见下表：

第11章差错控制编码

信息位 $a_6a_5a_4a_3$	监督位 $a_2a_1a_0$	信息位 $a_6a_5a_4a_3$	监督位 $a_2a_1a_0$
0000	000	1000	111
0001	011	1001	100
0010	101	1010	010
0011	110	1011	001
0100	110	1100	001
0101	101	1101	010
0110	011	1110	100
0111	000	1111	111

第11章差错控制编码

- 接收端收到每个码组后，先计算出S_1、S_2和S_3，再查表判断错码情况。例如，若接收码组为0000011，按上述公式计算可得：$S_1 = 0$，$S_2 = 1$，$S_3 = 1$。由于$S_1\ S_2\ S_3$等于011，故查表可知在a_3位有1错码。

- 按照上述方法构造的码称为汉明码。表中所列的(7, 4)汉明码的最小码距$d_0 = 3$。因此，这种码能够纠正1个错码或检测2个错码。由于码率$k/n = (n - r)/n = 1 - r/n$，故当$n$很大和$r$很小时，码率接近1。可见，汉明码是一种高效码。

第11章差错控制编码

- 线性分组码的一般原理
  - 线性分组码的构造
    - ***H***矩阵

上面(7, 4)汉明码的例子有

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

现在将上面它改写为

$$\left.\begin{array}{l} 1\cdot a_6 + 1\cdot a_5 + 1\cdot a_4 + 0\cdot a_3 + 1\cdot a_2 + 0\cdot a_1 + 0\cdot a_0 = 0 \\ 1\cdot a_6 + 1\cdot a_5 + 0\cdot a_4 + 1\cdot a_3 + 0\cdot a_2 + 1\cdot a_1 + 0\cdot a_0 = 0 \\ 1\cdot a_6 + 0\cdot a_5 + 1\cdot a_4 + 1\cdot a_3 + 0\cdot a_2 + 0\cdot a_1 + 1\cdot a_0 = 0 \end{array}\right\}$$

上式中已经将“⊕”简写成“+”。

第11章差错控制编码

$$\left.\begin{aligned}
&1\cdot a_6+1\cdot a_5+1\cdot a_4+0\cdot a_3+1\cdot a_2+0\cdot a_1+0\cdot a_0=0\\
&1\cdot a_6+1\cdot a_5+0\cdot a_4+1\cdot a_3+0\cdot a_2+1\cdot a_1+0\cdot a_0=0\\
&1\cdot a_6+0\cdot a_5+1\cdot a_4+1\cdot a_3+0\cdot a_2+0\cdot a_1+1\cdot a_0=0
\end{aligned}\right\}$$

上式可以表示成如下矩阵形式：

$$\begin{bmatrix}1110100\\1101010\\1011001\end{bmatrix}\begin{bmatrix}a_6\\a_5\\a_4\\a_3\\a_2\\a_1\\a_0\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}\qquad(\text{模}2)$$

上式还可以简记为

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

第11章差错控制编码

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

式中

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

$$\boldsymbol{A}=[a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0]$$

$$\boldsymbol{0}=[000]$$

右上标“T”表示将矩阵转置。例如，$\boldsymbol{H}^{\mathrm{T}}$是$\boldsymbol{H}$的转置，即$\boldsymbol{H}^{\mathrm{T}}$的第一行为$\boldsymbol{H}$的第一列，$\boldsymbol{H}^{\mathrm{T}}$的第二行为$\boldsymbol{H}$的第二列等等。

将$\boldsymbol{H}$称为**监督矩阵**。

只要监督矩阵$\boldsymbol{H}$给定，编码时监督位和信息位的关系就完全确定了。

第11章差错控制编码

- ***H***矩阵的性质：

  1) ***H***的行数就是监督关系式的数目，它等于监督位的数目r。***H***的每行中“1”的位置表示相应码元之间存在的监督关系。例如，***H***的第一行1110100表示监督位a_2是由a_6 a_5 a_4之和决定的。***H***矩阵可以分成两部分，例如

$$\boldsymbol{H}=\left[\begin{array}{c:c}1110 & 100\\ 1101 & 010\\ 1011 & 001\end{array}\right]=\left[\boldsymbol{P}\boldsymbol{I}_r\right]$$

  式中，***P***为$r\times k$阶矩阵，***I***r为$r\times r$阶单位方阵。我们将具有[***P*** ***I***r]形式的***H***矩阵称为**典型阵**。

第11章差错控制编码

2) 由代数理论可知，***H***矩阵的各行应该是线性无关的，否则将得不到 *r*个线性无关的监督关系式，从而也得不到*r*个独立的监督位。若一矩阵能写成典型阵形式[***P*** ***I***r]，则其各行一定是线性无关的。因为容易验证[***I***r]的各行是线性无关的，故[***P*** ***I***r]的各行也是线性无关的。

- ***G***矩阵： 上面汉明码例子中的监督位公式为

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

也可以改写成矩阵形式：

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

第11章差错控制编码

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

或者写成

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

式中，$\boldsymbol{Q}$为一个$k \times r$阶矩阵，它为$\boldsymbol{P}$的转置，即 $\boldsymbol{Q} = \boldsymbol{P}^{\mathrm{T}}$

上式表示，在信息位给定后，用信息位的行矩阵乘矩阵$\boldsymbol{Q}$就产生出监督位。

第11章差错控制编码

我们将Q的左边加上1个$k \times k$阶单位方阵，就构成1个矩阵G

$$G = [I_k Q] = \begin{bmatrix} 1000 \vdots 111 \\ 0100 \vdots 110 \\ 0010 \vdots 101 \\ 0001 \vdots 011 \end{bmatrix}$$

G称为**生成矩阵**，因为由它可以产生整个码组，即有

$$[a_6 a_5 a_4 a_3 a_2 a_1 a_0] = [a_6 a_5 a_4 a_3] \cdot G$$

或者

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

因此，如果找到了码的生成矩阵G，则编码的方法就完全确定了。具有$[I_k Q]$形式的生成矩阵称为**典型生成矩阵**。由典型生成矩阵得出的码组A中，信息位的位置不变，监督位附加于其后。这种形式的码称为**系统码**。

第11章差错控制编码

- ***G***矩阵的性质：

  1) ***G***矩阵的各行是线性无关的。因为由上式可以看出，任一码组***A***都是***G***的各行的线性组合。***G***共有k行，若它们线性无关，则可以组合出2^k种不同的码组***A***，它恰是有k位信息位的全部码组。若***G***的各行有线性相关的，则不可能由***G***生成2^k种不同的码组了。

  2) 实际上，***G***的各行本身就是一个码组。因此，如果已有k个线性无关的码组，则可以用其作为生成矩阵***G***，并由它生成其余码组。

第11章差错控制编码

- 错码矩阵和错误图样
  - 一般说来，$\boldsymbol{A}$为一个n列的行矩阵。此矩阵的n个元素就是码组中的n个码元，所以发送的码组就是$\boldsymbol{A}$。此码组在传输中可能由于干扰引入差错，故接收码组一般说来与$\boldsymbol{A}$不一定相同。
  - 若设接收码组为一n列的行矩阵$\boldsymbol{B}$，即

$$\boldsymbol{B} = [b_{n-1} b_{n-2} \cdots b_1 b_0]$$

则发送码组和接收码组之差为

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

它就是传输中产生的**错码**行**矩阵**

$$\boldsymbol{E} = [e_{n-1} e_{n-2} \cdots e_1 e_0]$$

式中

$$e_i = \begin{cases} 0, & \text{当} b_i = a_i \\ 1, & \text{当} b_i \neq a_i \end{cases}$$

第11章差错控制编码

因此，若$e_i = 0$，表示该接收码元无错；若$e_i = 1$，则表示该接收码元有错。

$$\boldsymbol{B} - \boldsymbol{A} = \boldsymbol{E} \qquad \text{可以改写成} \qquad \boldsymbol{B} = \boldsymbol{A} + \boldsymbol{E}$$

例如，若发送码组$\boldsymbol{A} = [1000111]$，错码矩阵$\boldsymbol{E} = [0000100]$，则接收码组$\boldsymbol{B} = [1000011]$。

错码矩阵有时也称为**错误图样**。

第11章差错控制编码

- 校正子S

当接收码组有错时，$E \neq 0$，将B当作A代入公式$(A \cdot H^{T} = 0)$后，该式不一定成立。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。在未超过检错能力时，上式不成立，即其右端不等于0。假设这时该式的右端为S，即

$$B \cdot H^{T} = S$$

将$B = A + E$代入上式，可得

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

由于$A \cdot H^{T} = 0$，所以

$$S = E \cdot H^{T}$$

式中S称为校正子。它能用来指示错码的位置。

S和错码E之间有确定的线性变换关系。若S和E之间一一对应，则S将能代表错码的位置。

第11章差错控制编码

- 线性分组码的性质
  - **封闭性：**是指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。

    这就是说，若$\boldsymbol{A}_1$和$\boldsymbol{A}_2$是一种线性码中的两个许用码组，则$(\boldsymbol{A}_1+\boldsymbol{A}_2)$仍为其中的一个码组。这一性质的证明很简单。若$\boldsymbol{A}_1$和$\boldsymbol{A}_2$是两个码组，则有

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

    将上两式相加，得出

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

    所以$(\boldsymbol{A}_1 + \boldsymbol{A}_2)$也是一个码组。

    由于线性码具有封闭性，所以两个码组($\boldsymbol{A}_1$和$\boldsymbol{A}_2$)之间的距离（即对应位不同的数目）必定是另一个码组$(\boldsymbol{A}_1 + \boldsymbol{A}_2)$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）。

第11章差错控制编码

11.6 循环码

11.6.1 循环码原理

- **循环性**：循环性是指任一码组循环一位（即将最右端的一个码元移至左端，或反之）以后，仍为该码中的一个码组。在下表中给出一种(7, 3)循环码的全部码组。

码组编号	信息位	监督位	码组编号	信息位	监督位
	$a_6a_5a_4$	$a_3a_2a_1a_0$		$a_6a_5a_4$	$a_3a_2a_1a_0$
1	000	0000	5	100	1011
2	001	0111	6	101	1100
3	010	1110	7	110	0101
4	011	1001	8	111	0010

例如，表中的第2码组向右移一位即得到第5码组；第6码组向右移一位即得到第7码组。

第11章差错控制编码

一般说来，若$(a_{n-1}\ a_{n-2}\ \ldots a_0)$是循环码的一个码组，则循环移位后的码组

$$(a_{n-2}\ a_{n-3}\ \ldots\ a_0\ a_{n-1})$$

$$(a_{n-3}\ a_{n-4}\ \ldots\ \ a_{n-1}\ a_{n-2})$$

$$\ldots\ \ldots\ \ldots$$

$$(a_0\ a_{n-1}\ \ldots a_2\ a_1)$$

也是该编码中的码组。

第11章差错控制编码

- 码多项式
  - 码组的多项式表示法

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

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

例如，上表中的任意一个码组可以表示为

$$T(x)=a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0$$

其中第7个码组可以表示为

$$T(x)=1\cdot x^6+1\cdot x^5+0\cdot x^4+0\cdot x^3+1\cdot x^2+0\cdot x+1$$

$$=x^6+x^5+x^2+1$$

这种多项式中，x仅是码元位置的标记，例如上式表示第7码组中a_6、a_5、a_2和a_0为“1”，其他均为0。因此我们并不关心x的取值。

第11章差错控制编码

- 码多项式的按模运算
  - 在整数运算中，有模n运算。例如，在模2运算中，有

$$1+1=2\equiv 0 \quad (\text{模}2),$$

$$1+2=3\equiv 1 \quad (\text{模}2),$$

$$2\times 3=6\equiv 0 \quad (\text{模}2)$$

等等。一般说来，若一个整数m可以表示为

$$\frac{m}{n}=Q+\frac{p}{n}, \qquad p<n$$

式中，Q - 整数，

则在模 n 运算下，有

$$m\equiv p \quad (\text{模}n)$$

即，在模 n 运算下，一个整数m等于它被 n 除得的余数。

第11章差错控制编码

- 在码多项式运算中也有类似的按模运算。

若一任意多项式$F(x)$被一 n 次多项式$N(x)$除，得到商式$Q(x)$和一个次数小于n的余式$R(x)$，即

$$F(x)=N(x)Q(x)+R(x)$$

则写为 $F(x)\equiv R(x)$ （模$N(x)$）

这时，码多项式系数仍按模2 运算，即系数只取 0 和1。

例如，x^3被(x^3+1)除，得到余项1。所以有

$$x^3\equiv 1 \quad （模(x^3+1)）$$

同理 $x^4+x^2+1\equiv x^2+x+1$ （模(x^3+1)）

因为

$$\begin{array}{r} x \\ x^3+1 \overline{)\, x^4+x^2+1} \\ \underline{x^4+x} \\ x^2+x+1 \end{array}$$

应当注意，由于在模2运算中，用加法代替了减法，故余项不是x^2-x+1，而是x^2+x+1。

第11章差错控制编码

- 循环码的码多项式
  - 在循环码中，若T(x)是一个长为n的许用码组，则$x^i \cdot T(x)$在按模x^n+1运算下，也是该编码中的一个许用码组，即若

$$x^i \cdot T(x) \equiv T'(x) \qquad (\text{模}(x^n+1))$$

则$T'(x)$也是该编码中的一个许用码组。

【证】因为若 $T(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0$

则

$$x^i \cdot T(x)=a_{n-1}x^{n-1+i}+a_{n-2}x^{n-2+i}+\cdots+a_{n-1-i}x^{n-1}+\cdots+a_1x^{1+i}+a_0x^i$$

$$\equiv a_{n-1-i}x^{n-1}+a_{n-2-i}x^{n-2}+\cdots+a_0x^i+a_{n-1}x^{i-1}+\cdots+a_{n-i}$$

$$(\text{模}(x^n+1))$$

所以，这时有

$$T'(x)=a_{n-1-i}x^{n-1}+a_{n-2-i}x^{n-2}+\cdots+a_0x^i+a_{n-1}x^{i-1}+\cdots+a_{n-i}$$

第11章差错控制编码

$$T'(x) = a_{n-1-i}x^{n-1} + a_{n-2-i}x^{n-2} + \cdots + a_0 x^i + a_{n-1}x^{i-1} + \cdots + a_{n-i}$$

上式中$T'(x)$正是$T(x)$代表的码组向左循环移位i次的结果。因为原已假定$T(x)$是循环码的一个码组，所以$T'(x)$也必为该码中一个码组。例如，循环码组

$$T(x) = x^6 + x^5 + x^2 + 1$$

其码长$n = 7$。现给定$i = 3$，则

$$\begin{aligned} x^3 \cdot T(x) &= x^3(x^6 + x^5 + x^2 + 1) \\ &= x^9 + x^8 + x^5 + x^3 \\ &= x^5 + x^3 + x^2 + x \qquad (\text{模}(x^7 + 1)) \end{aligned}$$

其对应的码组为0101110，它正是表中第3码组。

由上述分析可见，一个长为n的循环码必定为按模$(x^n + 1)$运算的一个余式。

第11章差错控制编码

- 循环码的生成矩阵$\boldsymbol{G}$
  - 由上节中公式

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

可知，有了生成矩阵$\boldsymbol{G}$，就可以由k个信息位得出整个码组，而且生成矩阵$\boldsymbol{G}$的每一行都是一个码组。例如，在此式中，若$a_6 a_5 a_4 a_3 = 1000$，则码组$\boldsymbol{A}$就等于$\boldsymbol{G}$的第一行；若$a_6 a_5 a_4 a_3 = 0100$，则码组$\boldsymbol{A}$就等于$\boldsymbol{G}$的第二行；等等。由于$\boldsymbol{G}$是k行n列的矩阵，因此若能找到k个已知码组，就能构成矩阵$\boldsymbol{G}$。如前所述，这k个已知码组必须是线性不相关的，否则给定的信息位与编出的码组就不是一一对应的。

  - 在循环码中，一个(n, k)码有2^k个不同的码组。若用$g(x)$表示其中前$(k-1)$位皆为“0”的码组，则$g(x)$，$x\,g(x)$，$x^2\,g(x)$，…，$x^{k-1}\,g(x)$都是码组，而且这k个码组是线性无关的。因此它们可以用来构成此循环码的生成矩阵$\boldsymbol{G}$。

第11章差错控制编码

- 在循环码中除全“0”码组外，再没有连续k位均为“0”的码组，即连“0”的长度最多只能有$(k-1)$位。否则，在经过若干次循环移位后将得到一个k位信息位全为“0”，但监督位不全为“0”的一个码组。这在线性码中显然是不可能的。因此，$g(x)$必须是一个常数项不为“0”的$(n-k)$次多项式，而且这个$g(x)$还是这种(n, k)码中次数为$(n-k)$的唯一多项式。因为如果有两个，则由码的封闭性，把这两个相加也应该是一个码组，且此码组多项式的次数将小于$(n-k)$，即连续“0”的个数多于$(k-1)$。显然，这是与前面的结论矛盾的，故是不可能的。我们称这唯一的$(n-k)$次多项式$g(x)$为码的生成多项式。一旦确定了$g(x)$，则整个(n, k)循环码就被确定了。

第11章差错控制编码

- 因此，循环码的生成矩阵$\boldsymbol{G}$可以写成

$$\boldsymbol{G}(x)=\begin{bmatrix} x^{k-1}g(x) \\ x^{k-2}g(x) \\ \vdots \\ xg(x) \\ g(x) \end{bmatrix}$$

- 例：在上表所给出的$(7, 3)$循环码中，$n = 7, k = 3, n - k = 4$。由此表可见，唯一的一个$(n - k) = 4$次码多项式代表的码组是第二码组0010111，与它相对应的码多项式（即生成多项式）$g(x) = x^4 + x^2 + x + 1$。将此$g(x)$代入上式，得到

$$\boldsymbol{G}(x)=\begin{bmatrix} x^{2}g(x) \\ xg(x) \\ g(x) \end{bmatrix} \quad \text{或} \quad \boldsymbol{G}(x)=\begin{bmatrix} 1011100 \\ 0101110 \\ 0010111 \end{bmatrix}$$

由于上式不符合$\boldsymbol{G}=[\boldsymbol{I}_k\boldsymbol{Q}]$的形式，所以它不是典型阵。不过，将它作线性变换，不难化成典型阵。

我们可以写出此循环码组，即

$$T(x)=[a_6a_5a_4]\boldsymbol{G}(x)=[a_6a_5a_4]\begin{bmatrix}x^2g(x)\\ xg(x)\\ g(x)\end{bmatrix}$$

$$=a_6x^2g(x)+a_5xg(x)+a_4g(x)$$

$$=(a_6x^2+a_5x+a_4)g(x)$$

上式表明，所有码多项式$T(x)$都可被$g(x)$整除，而且任意一个次数不大于$(k-1)$的多项式乘$g(x)$都是码多项式。需要说明一点，两个矩阵相乘的结果应该仍是一个矩阵。上式中两个矩阵相乘的乘积是只有一个元素的一阶矩阵，这个元素就是$T(x)$。为了简洁，式中直接将乘积写为此元素。

第11章差错控制编码

- 如何寻找任一(n, k)循环码的生成多项式

  由上式可知，任一循环码多项式$T(x)$都是$g(x)$的倍式，故它可以写成

  $$T(x) = h(x)\cdot g(x)$$

  而生成多项式g(x)本身也是一个码组，即有

  $$T'(x) = g(x)$$

  由于码组$T'(x)$是一个$(n-k)$次多项式，故$x^k\,T'(x)$是一个n次多项式。由下式

  $$x^i \cdot T(x) \equiv T'(x) \qquad (\text{模}(x^n+1))$$

  可知，$x^k\,T'(x)$在模(x^n+1)运算下也是一个码组，故可以写成

  $$\frac{x^k T'(x)}{x^n+1} = Q(x) + \frac{T(x)}{x^n+1}$$

$$\frac{x^k T'(x)}{x^n+1} = Q(x) + \frac{T(x)}{x^n+1}$$

上式左端分子和分母都是n次多项式，故商式$Q(x) = 1$。因此，上式可以化成

$$x^k T'(x) = (x^n+1) + T(x)$$

将$T(x)$和$T'(x)$表示式代入上式，经过化简后得到

$$x^n+1 = g(x)[x^k + h(x)]$$

上式表明，生成多项式$g(x)$应该是$(x^n + 1)$的一个因子。这一结论为我们寻找循环码的生成多项式指出了一条道路，即循环码的生成多项式应该是(x^n+1)的一个$(n-k)$次因式。例如，$(x^7 + 1)$可以分解为

$$x^7+1 = (x+1)(x^3+x^2+1)(x^3+x+1)$$

为了求$(7, 3)$循环码的生成多项式$g(x)$，需要从上式中找到一个$(n-k) = 4$次的因子。不难看出，这样的因子有两个，即

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

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

以上两式都可作为生成多项式。不过，选用的生成多项式不同，产生出的循环码码组也不同。

第11章差错控制编码

11.6.2 循环码的编解码方法

循环码的编码方法

编码原则

在编码时，首先要根据给定的(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$g(x)$。

由于所有码多项式$T(x)$都可以被$g(x)$整除。根据这条原则，就可以对给定的信息位进行编码：

设$m(x)$为信息码多项式，其次数小于k。用x^{n-k}乘$m(x)$，得到的$x^{n-k}\,m(x)$的次数必定小于n。用$g(x)$除$x^{n-k}\,m(x)$，得到余式$r(x)$，$r(x)$的次数必定小于$g(x)$的次数，即小于$(n - k)$。将此余式$r(x)$加于信息位之后作为监督位，即将$r(x)$和$x^{n-k}\,m(x)$相加，得到的多项式必定是一个码多项式。因为它必须能被$g(x)$整除，且商的次数不大于$(k - 1)$。

第11章差错控制编码

- 编码步骤：
  - 用x^{n-k}乘$m(x)$。这一运算实际上是在信息码后附加上$(n-k)$个“0”。例如，信息码为110，它相当于$m(x)=x^2+x$。当$n-k=7-3=4$时，$x^{n-k}m(x)=x^4(x^2+x)=x^6+x^5$，它相当于1100000。
  - 用$g(x)$除$x^{n-k}m(x)$，得到商$Q(x)$和余式$r(x)$，即

$$\frac{x^{n-k}m(x)}{g(x)}=Q(x)+\frac{r(x)}{g(x)}$$

例如，若选定$g(x)=x^4+x^2+x+1$，则

$$\frac{x^{n-k}m(x)}{g(x)}=\frac{x^6+x^5}{x^4+x^2+x+1}=(x^2+x+1)+\frac{x^2+1}{x^4+x^2+x+1}$$

上式相当于

$$\frac{1100000}{10111}=111+\frac{101}{10111}$$

第11章差错控制编码

- 编出的码组$T(x)$为

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

在上例中，$T(x) = 1100000 + 101 = 1100101$，它就是上表中的第7码组。

第11章差错控制编码

- 循环码的解码方法
  - 解码要求：检错和纠错。
  - 检错解码原理：由于任意一个码组多项式$T(x)$都应该能被生成多项式$g(x)$整除，所以在接收端可以将接收码组$R(x)$用原生成多项式$g(x)$去除。当传输中未发生错误时，接收码组与发送码组相同，即$R(x) = T(x)$，故接收码组$R(x)$必定能被$g(x)$整除；若码组在传输中发生错误，则$R(x) \neq T(x)$，$R(x)$被$g(x)$除时可能除不尽而有余项，即有

$$R(x)/g(x) = Q(x) + r(x)/g(x)$$

因此，就以余项是否为零来判别接收码组中有无错码。

需要指出，有错码的接收码组也有可能被$g(x)$整除。这时的错码就不能检出了。这种错误称为不可检错误。不可检错误中的误码数必定超过了这种编码的检错能力。

第11章差错控制编码

- 纠错解码原理：为了能够纠错，要求每个可纠正的错误图样必须与一个特定余式有一一对应关系。因为只有存在上述一一对应的关系时，才可能从上述余式唯一地决定错误图样，从而纠正错码。因此，原则上纠错可按下述步骤进行：
  - 用生成多项式$g(x)$除接收码组$R(x)$，得出余式$r(x)$。
  - 按余式$r(x)$，用查表的方法或通过某种计算得到错误图样$E(x)$；例如，通过计算校正子S和查表，就可以确定错码的位置。
  - 从R(x)中减去$E(x)$，便得到已经纠正错码的原发送码组$T(x)$。
- 通常，一种编码可以有几种纠错解码方法，上述解码方法称为捕错解码法。

- 目前多采用软件运算实现上述编解码运算。

第11章差错控制编码

11.6.3 截短循环码

- 截短目的：在设计纠错编码方案时，常常信息位数k、码长n和纠错能力都是预先给定的。但是，并不一定有恰好满足这些条件的循环码存在。这时，可以采用将码长截短的方法，得出满足要求的编码。
- 截短方法：设给定一个(n, k)循环码，它共有2^k种码组，现使其前i $(0 < i < k)$个信息位全为“0”，于是它变成仅有2^{k-i}种码组。然后从中删去这i位全“0”的信息位，最终得到一个$(n-i, k-i)$的线性码。将这种码称为截短循环码。
- 截短循环码性能：循环码截短前后至少具有相同的纠错能力，并且编解码方法仍和截短前的方法一样。
- 例：要求构造一个能够纠正1位错码的$(13, 9)$码。这时可以由$(15, 11)$循环码的11种码组中选出前两信息位均为“0”的码组，构成一个新的码组集合。然后在发送时不发送这两位“0”。于是发送码组成为$(13, 9)$截短循环码。

第11章差错控制编码

11.6.4 BCH码

- 什么是BCH码？它是一种获得广泛应用的能够纠正多个错码的循环码，是以3位发明这种码的人名(Bose - Chaudhuri - Hocguenghem)命名的。BCH码的重要性在于它解决了生成多项式与纠错能力的关系问题，可以在给定纠错能力要求的条件下寻找到码的生成多项式。有了生成多项式，编码的基本问题就随之解决了。
- BCH码分类：
  - 本原BCH码：其生成多项式$g(x)$中含有最高次数为m的本原多项式，且码长为$n = 2^m - 1$，($m \geq 3$，为正整数)。
  - 非本原BCH码：其生成多项式中不含这种本原多项式，且码长n是$(2^m - 1)$的一个因子，即码长n一定除得尽$2^m - 1$。
  - 本原多项式的概念将在下一章介绍。

第11章差错控制编码

- BCH码的性能：
  - 码长n与监督位、纠错个数 t 之间的关系：

    对于正整数m ($m \geq 3$)和正整数$t < m / 2$，必定存在一个码长为$n = 2^m - 1$，监督位为$n - k \leq mt$，能纠正所有不多于t个随机错误的BCH码。若码长$n = (2^m - 1) / i$（$i > 1$，且除得尽$(2^m - 1)$），则为非本原BCH码。
  - 汉明码是能够纠正单个随机错误的码。可以证明，具有循环性质的汉明码就是能纠正单个随机错误的本原BCH码。
  - 例如，(7, 4)汉明码就是以$g_1(x) = x^3 + x + 1$或$g_2(x) = x^3 + x^2 + 1$生成的BCH码，而用$g_3(x) = x^4 + x + 1$或$g_4(x) = x^4 + x^3 + 1$都能生成(15, 11)汉明码。

第11章差错控制编码

- BCH码的设计
  - 在工程设计中，一般不需要用计算方法去寻找生成多项式$g(x)$。因为前人早已将寻找到的$g(x)$列成表，故可以用查表法找到所需的生成多项式。
  - 下表给出了码长$n \leq 127$的二进制本原BCH码生成多项式。

第11章差错控制编码

$n = 3$

k	t	$g(x)$
1	1	7

$n = 7$

k	t	$g(x)$
4	1	13
1	3	77

$n = 15$

k	t	$g(x)$
11	1	23
7	2	721
5	3	2467
1	7	77777

$n = 63$

k	t	$g(x)$
57	1	103
51	2	12471
45	3	1701317
39	4	166623567
36	5	1033500423
30	6	157464165347
24	7	17323260404441
18	10	1363026512351725
16	11	6331141367235453
10	13	47262230552725015 5
7	15	5231045543503271737
1	31	全部为1

第11章差错控制编码

$n = 31$			$n = 127$		
k	t	$g(x)$	k	t	$g(x)$
26	1	45	120	1	211
21	2	3551	113	2	41567
16	3	107657	106	3	11554743
11	5	5423325	99	4	3447023271
6	7	313365047	92	5	624730022327
1	15	17777777777	85	6	130704476322273
			78	7	26230002166130115
			71	9	6255010713253127753
			64	10	1206534025570773100045
			57	11	235265252505705053517721
			50	13	544465125233140124215014 21
			43	15	177217722136512275212205743 43
			36	≥15	3146074666522075044764574721735
			29	≥22	403114461367670603667530141176155
			22	≥23	123376070404722522435445626637647043
			15	≥27	220570424456045547705230137622176043 53
			8	≥31	70472640527510306514762242715677331302 17
			1	63	全部为1

第11章差错控制编码

- 表中给出的生成多项式系数是用8进制数字列出的。例如，$g(x) = (13)_8$是指$g(x) = x^3 + x + 1$，因为$(13)_8 = (1011)_2$，后者就是此3次方程$g(x)$的各项系数。
- 下表列出了部分非本原BCH码的生成多项式参数。

n	k	t	$g(x)$	n	k	t	$g(x)$
17	9	2	727	47	24	5	43073357
21	12	2	1663	65	53	2	10761
23	12	3	5343	65	40	4	354300067
33	22	2	5145	73	46	4	1717773537
41	21	4	6647133				

第11章差错控制编码

- 戈莱码：

在上表中的(23, 12)码称为戈莱(Golay)码。它能纠正3个随机错码，并且容易解码，实际应用较多。

- 扩展BCH码：

BCH码的长度都为奇数。在应用中，为了得到偶数长度的码，并增大检错能力，可以在BCH码生成多项式中乘上一个因式$(x+1)$，从而得到扩展BCH码$(n+1, k)$。扩展BCH码相当于在原BCH码上增加了一个校验位，因此码距比原BCH码增加1。扩展BCH码已经不再具有循环性。例如，广泛实用的扩展戈莱码(24, 12)，其最小码距为8，码率为1/2，能够纠正3个错码和检测4个错码。它比汉明码的纠错能力强很多，付出的代价是解码更复杂，码率也比汉明码低。此外，它不再是循环码了。

第11章差错控制编码

11.6.5 RS码： 它是一类具有很强纠错能力的多进制BCH码。

- 若仍用n表示RS码的码长，则对于m进制的RS码，其码长需要满足下式：　　$n = m - 1 = 2^q - 1$

  式中 $q \geq 2$，为整数。
- 对于能够纠正t个错误的RS码，其监督码元数目为$r = 2t$，这时的最小码距$d_0 = 2t + 1$。
- RS码的生成多项式为

  $$g(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^{2t})$$

  式中，α - 伽罗华域GF(2q)中的本原元。
- 若将每个m进制码元表示成相应的q位二进制码元，则得到的二进制码的参数为：

  码长：$n = q(2^q - 1)$（二进制码元）

  监督码：$r = 2qt$　　（二进制码元）

第11章差错控制编码

- 由于RS码能够纠正t个m进制错码，或者说，能够纠正码组中t个不超过q位连续的二进制错码，所以RS码特别适用于存在突发错误的信道，例如移动通信网等衰落信道中。此外，因为它是多进制纠错编码，所以特别适合用于多进制调制的场合。

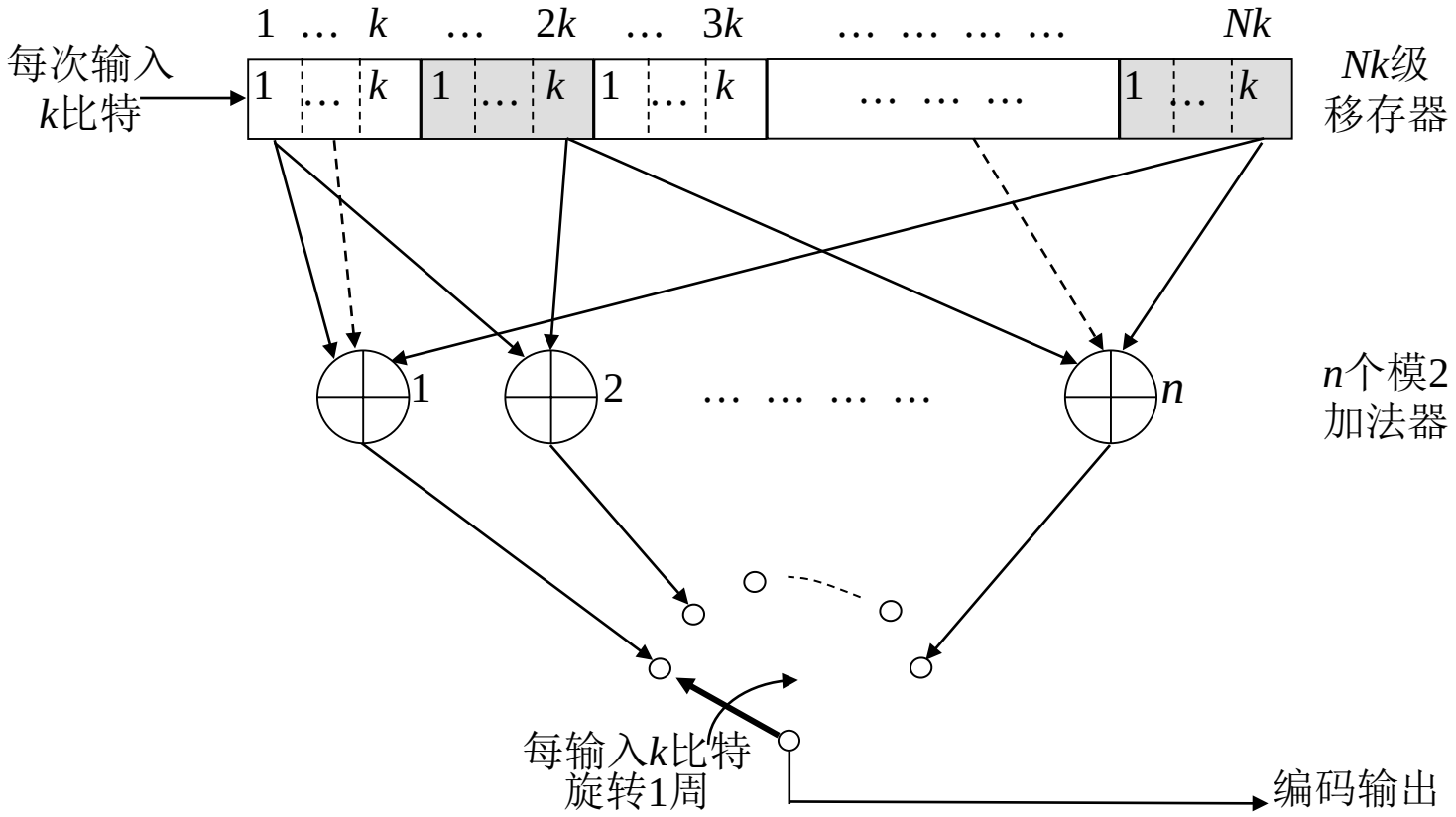
第11章差错控制编码

11.7 卷积码

- 非分组码概念：
  - 卷积码是一种非分组码。通常它更适用于前向纠错，因为对于许多实际情况它的性能优于分组码，而且运算较简单。
  - 卷积码在编码时虽然也是把k个比特的信息段编成n个比特的码组，但是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称为编码约束度，并将nN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。

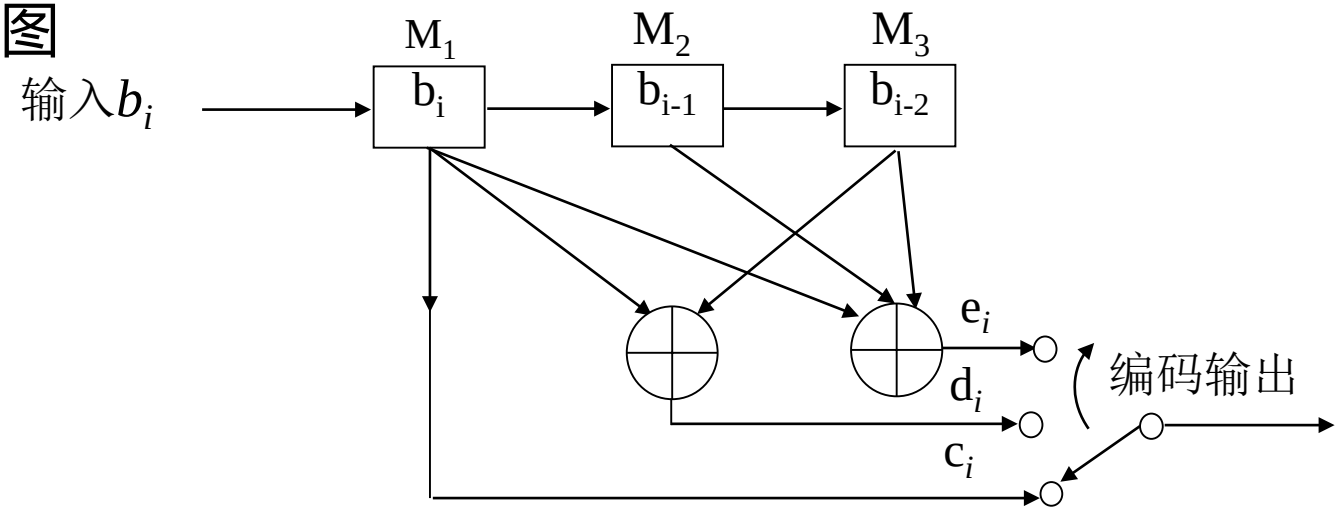
第11章差错控制编码

- 11.7.1 卷积码的基本原理
  - 编码器原理方框图



第11章差错控制编码

- 例： $(n, k, N) = (3, 1, 3)$卷积码编码器
  - 方框图



  - 设输入信息比特序列是$\ldots b_{i-2}\ b_{i-1}\ b_i\ b_{i+1}\ldots$，则当输入$b_i$时，此编码器输出3比特$c_i\ d_i\ e_i$，输入和输出的关系如下：

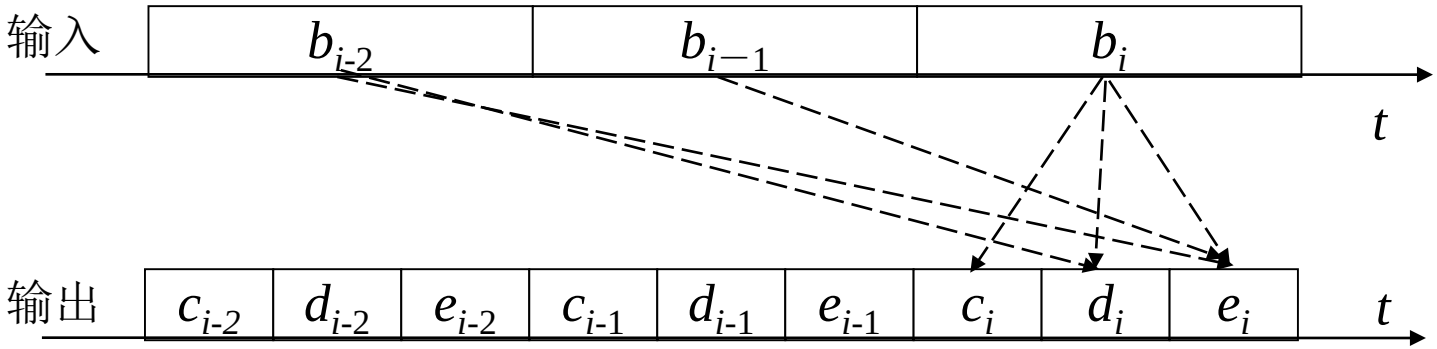
$$c_i = b_i$$

$$d_i = b_i \oplus b_{i-2}$$

$$e_i = b_i \oplus b_{i-1} \oplus b_{i-2}$$

第11章差错控制编码

在下图中用虚线示出了信息位b_i的监督位和各信息位之间的约束关系。这里的编码约束长度nN等于9。



11.7.2 卷积码的代数表述

上式表示卷积码也是一种线性码。一个线性码完全由一个监督矩阵***H***或生成矩阵***G***所确定。下面就来寻找这两个矩阵。

- 监督矩阵H

  现在仍从上面的实例开始分析。假设上图中在第1个信息位b_1进入编码器之前，各级移存器都处于“0”状态，则监督位d_i、e_i和信息位b_i之间的关系可以写为

第11章差错控制编码

$$\begin{cases} d_1 = b_1 \\ e_1 = b_1 \\ d_2 = b_2 \\ e_2 = b_2 + b_1 \\ d_3 = b_3 + b_1 \\ e_3 = b_3 + b_2 + b_1 \\ d_4 = b_4 + b_2 \\ e_4 = b_4 + b_3 + b_2 \\ \cdots \quad \cdots \quad \cdots \end{cases}$$

左式可以改写为

$$\begin{cases} b_1 + d_1 = 0 \\ b_1 + e_1 = 0 \\ b_2 + d_2 = 0 \\ b_1 + b_2 + e_2 = 0 \\ b_1 + b_3 + d_3 = 0 \\ b_1 + b_2 + b_3 + e_3 = 0 \\ b_2 + b_4 + d_4 = 0 \\ b_2 + b_3 + b_4 + e_4 = 0 \\ \cdots \quad \cdots \quad \cdots \end{cases}$$

在上面两个式子和后面的式子中，为简便计，用“+”代替“⊕”。

将上式用矩阵表示时，可以写成

第11章差错控制编码

$$\begin{bmatrix} 11 \\ 101 \\ 00011 \\ 100101 \\ 10000011 \\ 100100101 \\ 00010000011 \\ 000100100101 \\ \cdots\cdots\cdots\cdots \end{bmatrix} \begin{bmatrix} b_1 \\ d_1 \\ e_1 \\ b_2 \\ d_2 \\ e_2 \\ b_3 \\ d_3 \\ e_3 \\ b_4 \\ d_4 \\ e_4 \end{bmatrix} = [O]$$

与11.5节公式$\boldsymbol{H} \cdot \boldsymbol{A}^{\mathrm{T}} = \boldsymbol{0}^{\mathrm{T}}$对比，可以看出监督矩阵为

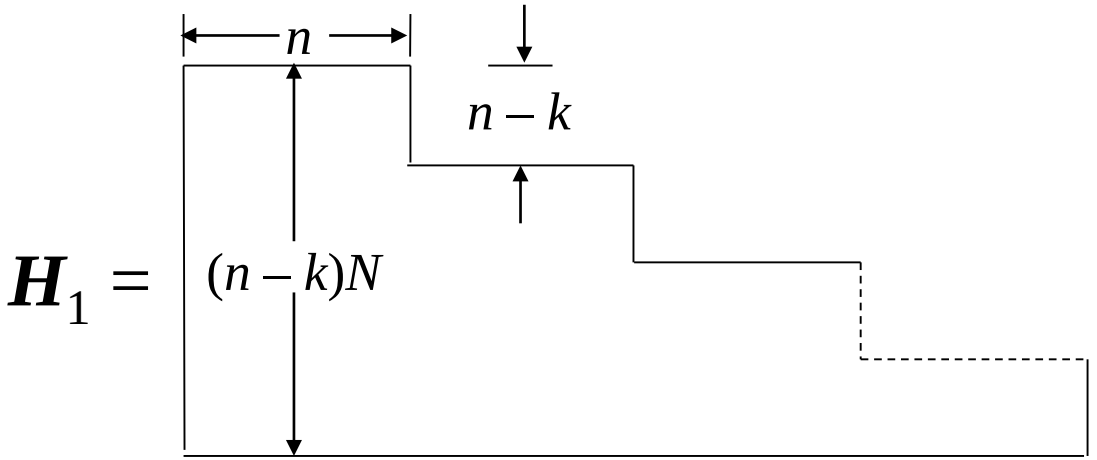
第11章差错控制编码

$$\boldsymbol{H}=\begin{bmatrix}11\\101\\00011\\100101\\10000011\\100100101\\00010000011\\000100100101\\\cdots\cdots\cdots\end{bmatrix}$$

由此例可见，卷积码的监督矩阵$\boldsymbol{H}$是一个有头无尾的半无穷矩阵。此外，这个矩阵的每3列的结构是相同的，只是后3列比前3列向下移了两行。例如，第4 ~ 6列比第1 ~ 3列低2行。自第7行起，每两行的左端比上两行多了3个“0”。

第11章差错控制编码

虽然这样的半无穷矩阵不便于研究，但是只要研究产生前9个码元（9为约束长度）的监督矩阵就足够了。不难看出，这种截短监督矩阵的结构形式如下图所示：



由此图可见，$\boldsymbol{H}_1$的最左边是n列$(n\text{-}k)N$行的一个子矩阵，且向右的每n列均相对于前n列降低$(n-k)$行。

第11章差错控制编码

此例中码的截短监督矩阵可以写成如下形式：

$$\boldsymbol{H}_1 = \begin{bmatrix} 11 \\ 101 \\ 00011 \\ 100101 \\ 10000011 \\ 100100101 \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{I}_2 & & & & \\ \boldsymbol{P}_2 & \boldsymbol{O}_2 & \boldsymbol{P}_1 & \boldsymbol{I}_2 & & \\ \boldsymbol{P}_3 & \boldsymbol{O}_2 & \boldsymbol{P}_2 & \boldsymbol{O}_2 & \boldsymbol{P}_1 & \boldsymbol{I}_2 \end{bmatrix}$$

式中

$\boldsymbol{I}_2 = \begin{bmatrix} 10 \\ 01 \end{bmatrix}$ —— 2阶单位方阵；

$\boldsymbol{P}_i$ —— 1×2阶矩阵，$i = 1, 2, 3$；

$\boldsymbol{O}_2$ —— 2阶全零方阵。

第11章差错控制编码

一般说来，卷积码的截短监督矩阵具有如下形式：

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

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。

有时还将$\boldsymbol{H}_1$的末行称为基本监督矩阵$\boldsymbol{h}$

$$\boldsymbol{h}=[\boldsymbol{P}_N\ \boldsymbol{O}_{n-k}\ \boldsymbol{P}_{N-1}\ \boldsymbol{O}_{n-k}\ \boldsymbol{P}_{N-2}\ \boldsymbol{O}_{n-k}\cdots\boldsymbol{P}_1\ \boldsymbol{I}_{n-k}]$$

它是卷积码的一个最重要的矩阵，因为只要给定了$\boldsymbol{h}$，则$\boldsymbol{H}_1$也就随之决定了。或者说，我们从给定的$\boldsymbol{h}$不难构造出$\boldsymbol{H}_1$。

第11章差错控制编码

- **生成矩阵G**

上例中的输出码元序列可以写成

$$[\ b_1\ d_1\ e_1\ b_2\ d_2\ e_2\ b_3\ d_3\ e_3\ b_4\ d_4\ e_4 \cdots]$$

$$= [\ b_1\ b_1\ b_1\ b_2\ b_2\ (b_2 + b_1)\ b_3\ (b_3 + b_1)\ (b_3 + b_2 + b_1)\ b_4\ (b_4 + b_2)\ (b_4 + b_3 + b_2) \cdots]$$

$$= [b_1 b_2 b_3 b_4 \cdots] \begin{bmatrix} 111 & 001 & 011 & 000 & 0\cdots \\ 000 & 111 & 001 & 011 & 0\cdots \\ 000 & 000 & 111 & 001 & 0\cdots \\ 000 & 000 & 000 & 111 & 0\cdots \\ 000 & 000 & 000 & 000 & 1\cdots \\ 000 & 000 & 000 & 000 & 0\cdots \\ 000 & 000 & 000 & 000 & 0\cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

第11章差错控制编码

此码的生成矩阵G即为上式最右矩阵：

$$G=\begin{bmatrix} 111 & 001 & 011 & 000 & 0\cdots \\ 000 & 111 & 001 & 011 & 0\cdots \\ 000 & 000 & 111 & 001 & 0\cdots \\ 000 & 000 & 000 & 111 & 0\cdots \\ 000 & 000 & 000 & 000 & 1\cdots \\ 000 & 000 & 000 & 000 & 0\cdots \\ 000 & 000 & 000 & 000 & 0\cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

它也是一个半无穷矩阵，其特点是每一行的结构相同，只是比上一行向右退后3列（因现在n = 3）。

第11章差错控制编码

- 截短生成矩阵：类似地，也有截短生成矩阵

$$\boldsymbol{G}_1 = \begin{bmatrix} 111 & 001 & 011 \\ 000 & 111 & 001 \\ 000 & 000 & 111 \end{bmatrix} = \begin{bmatrix} I_1 & Q_1 & O & Q_2 & O & Q_3 \\ & & I_1 & Q_1 & O & Q_2 \\ & & & & I_1 & Q_1 \end{bmatrix}$$

式中 I_1 - 一阶单位方阵；

Q_i - 2×1阶矩阵。

与$\boldsymbol{H}_1$矩阵比较可见，$\boldsymbol{Q}_i$是矩阵$\boldsymbol{P}_i^{\mathrm{T}}$的转置：

$$\boldsymbol{Q}_i = \boldsymbol{P}_i^{\mathrm{T}} \qquad (i = 1, 2, \ldots)$$

一般说来，截短生成矩阵具有如下形式：

第11章差错控制编码

$$G_1 = \begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{Q}_1 & \boldsymbol{O}_k & \boldsymbol{Q}_2 & \boldsymbol{O}_k & \boldsymbol{Q}_3 & \cdots & \boldsymbol{O}_k & \boldsymbol{Q}_N \\ & & \boldsymbol{I}_k & \boldsymbol{Q}_1 & \boldsymbol{O}_k & \boldsymbol{Q}_2 & \cdots & \boldsymbol{O}_k & \boldsymbol{Q}_{N-1} \\ & & & & \boldsymbol{I}_k & \boldsymbol{Q}_1 & \cdots & \boldsymbol{O}_k & \boldsymbol{Q}_{N-2} \\ & & & & & & \cdots & \vdots & \\ & & & & & & & \boldsymbol{I}_k & \boldsymbol{Q}_1 \end{bmatrix}$$

式中

$\boldsymbol{I}_k$ - k阶单位方阵;

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

$\boldsymbol{O}_k$ - k阶全零方阵。

并将上式中矩阵第一行称为基本生成矩阵

$$\boldsymbol{g} = [\boldsymbol{I}_k\ \boldsymbol{Q}_1\ \boldsymbol{O}_k\ \boldsymbol{Q}_2\ \boldsymbol{O}_k\ \boldsymbol{Q}_3 \ldots \boldsymbol{O}_k\ \boldsymbol{Q}_N]$$

同样，如果基本生成矩阵$\boldsymbol{g}$已经给定，则可以从已知的信息位得到整个编码序列。

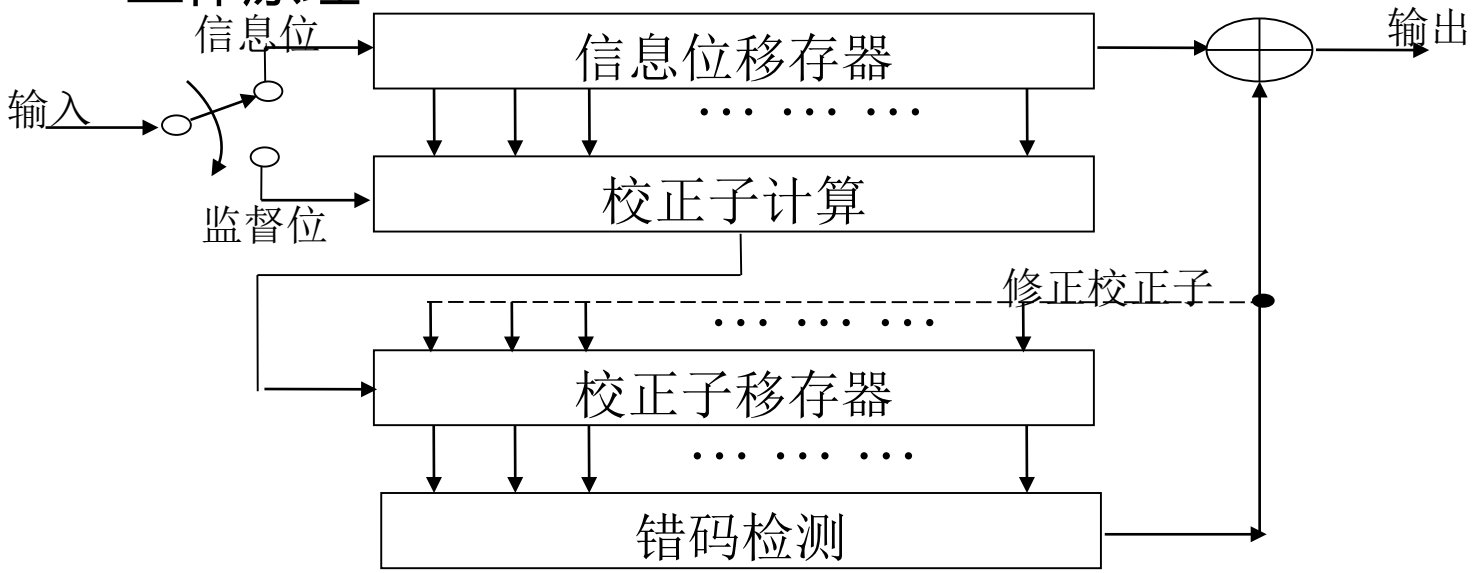
第11章差错控制编码

11.7.3 卷积码的解码

- 分类：
  - 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。大数逻辑解码对于约束长度较短的卷积码最为有效，而且设备较简单。
  - 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。针对无记忆信道提出的序贯解码就是概率解码方法之一。另一种概率解码方法是维特比算法。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。

第11章差错控制编码

- 大数逻辑解码
  - 工作原理



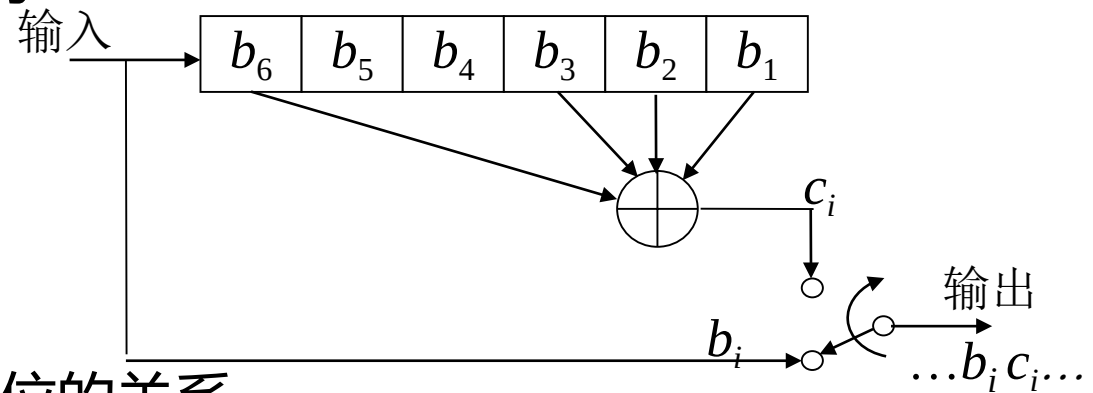
- 图中首先将接收信息位暂存于移存器中，并从接收码元的信息位和监督位计算校正子。然后，将计算得出的校正子暂存，并用它来检测错码的位置。在信息位移存器输出端，接有一个模2加电路；当检测到输出的信息位有错时，在输出的信息位上加“1”，从而纠正之。

第11章差错控制编码

- 这里的错码检测是采用二进制码的大数逻辑解码算法。它利用一组“正交”校验方程进行计算。
- 这里的“正交” 定义：若被校验的那个信息位出现在校验方程组的每一个方程中，而其他的信息位至多在一个方程中出现，则称这组方程为正交校验方程。这样就可以根据被错码影响了的方程数目在方程组中是否占多数来判断该信息位是否错了。下面将用一个实例来具体讲述这一过程。

第11章差错控制编码

- 例：(2, 1, 6)卷积码
  - 编码器方框图



  - 监督位和信息位的关系

    当输入序列为$b_1\ b_2\ b_3\ b_4\ \ldots$时，监督位为

$$c_1 = b_1$$
$$c_2 = b_2$$
$$c_3 = b_3$$
$$c_4 = b_1 + b_4$$
$$c_5 = b_1 + b_2 + b_5$$
$$c_6 = b_1 + b_2 + b_3 + b_6$$
$$\ldots\ \ldots\ \ldots\ \ldots$$

第11章差错控制编码

- 监督关系式

  参照11.5节中监督关系的定义式，容易写出

$$
\begin{aligned}
S_1 &= c_1 + b_1 \\
S_2 &= c_2 + b_2 \\
S_3 &= c_3 + b_3 \\
S_4 &= c_4 + b_1 + b_4 \\
S_5 &= c_5 + b_1 + b_2 + b_5 \\
S_6 &= c_6 + b_1 + b_2 + b_3 + b_6
\end{aligned}
$$

  上式中的S_i $(i = 1 \sim 6)$称为校正子。

- 正交校验方程组

  上式经过简单线性变换后，得出如下正交校验方程组：

第11章差错控制编码

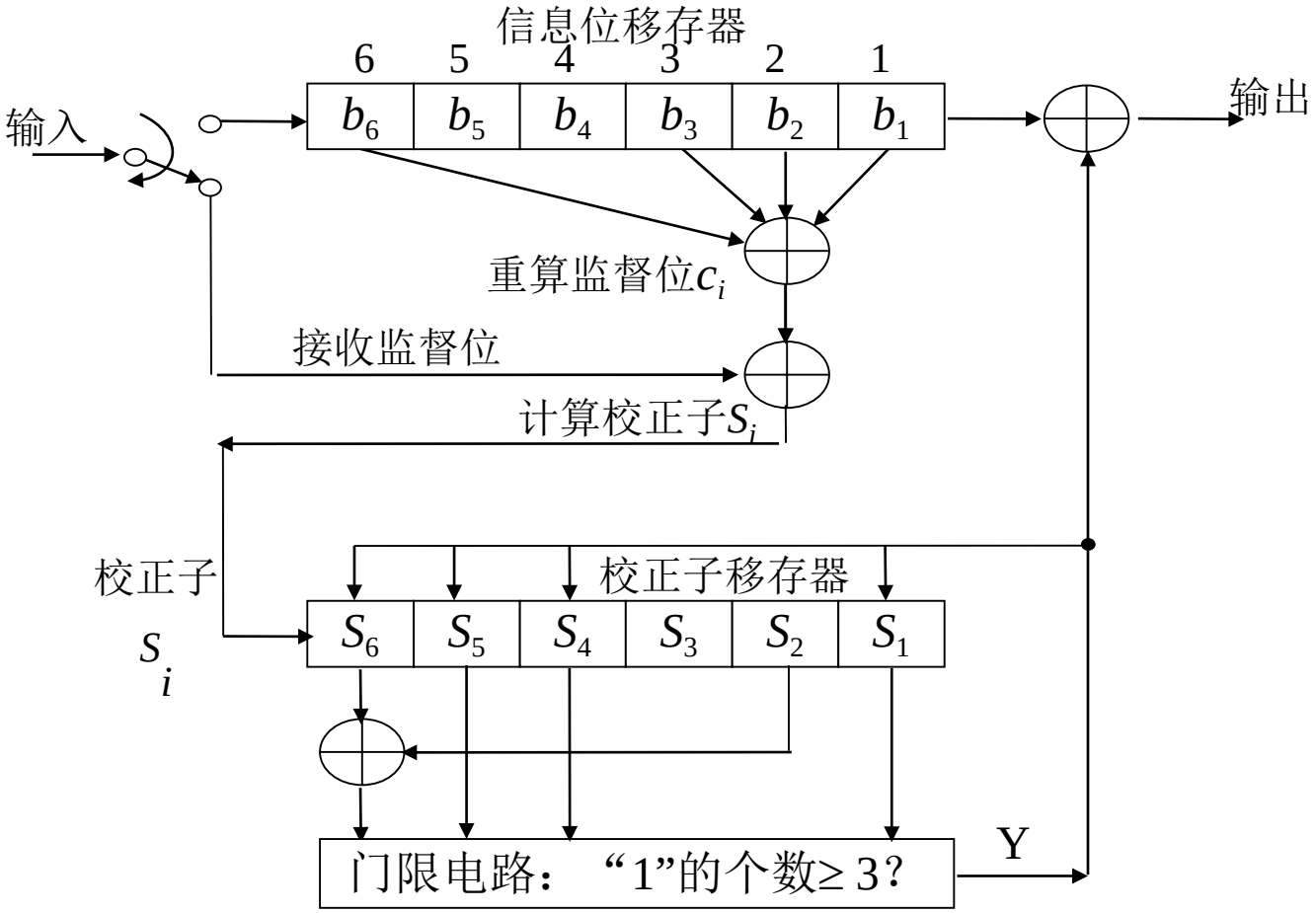
$$
\begin{cases}
S_1 = c_1 + b_1 \\
S_4 = c_4 + b_1 + b_4 \\
S_5 = c_5 + b_1 + b_2 + b_5 \\
S_2 + S_6 = c_2 + c_6 + b_1 + b_3 + b_6
\end{cases}
$$

在上式中，只有信息位b_1出现在每个方程中，监督位和其他信息位均最多只出现一次。因此，在接收端解码时，考察b_1、c_1至b_6、c_6等12个码元，仅当b_1出错时，式中才可能有3个或3个以上方程等于“1”。从而能够纠正b_1的错误。

第11章差错控制编码

- 解码器方框图

  按照这一原理画出的此(2, 1, 6)卷积码解码器方框图如下



第11章差错控制编码

由此图可见，当信息位出现一个错码时，仅当它位于信息位移存器的第6、3、2和1级时，才使校正子等于“1”。因此，这时的校正子序列为100111；

反之，当监督位出现一个错码时，校正子序列将为100000。

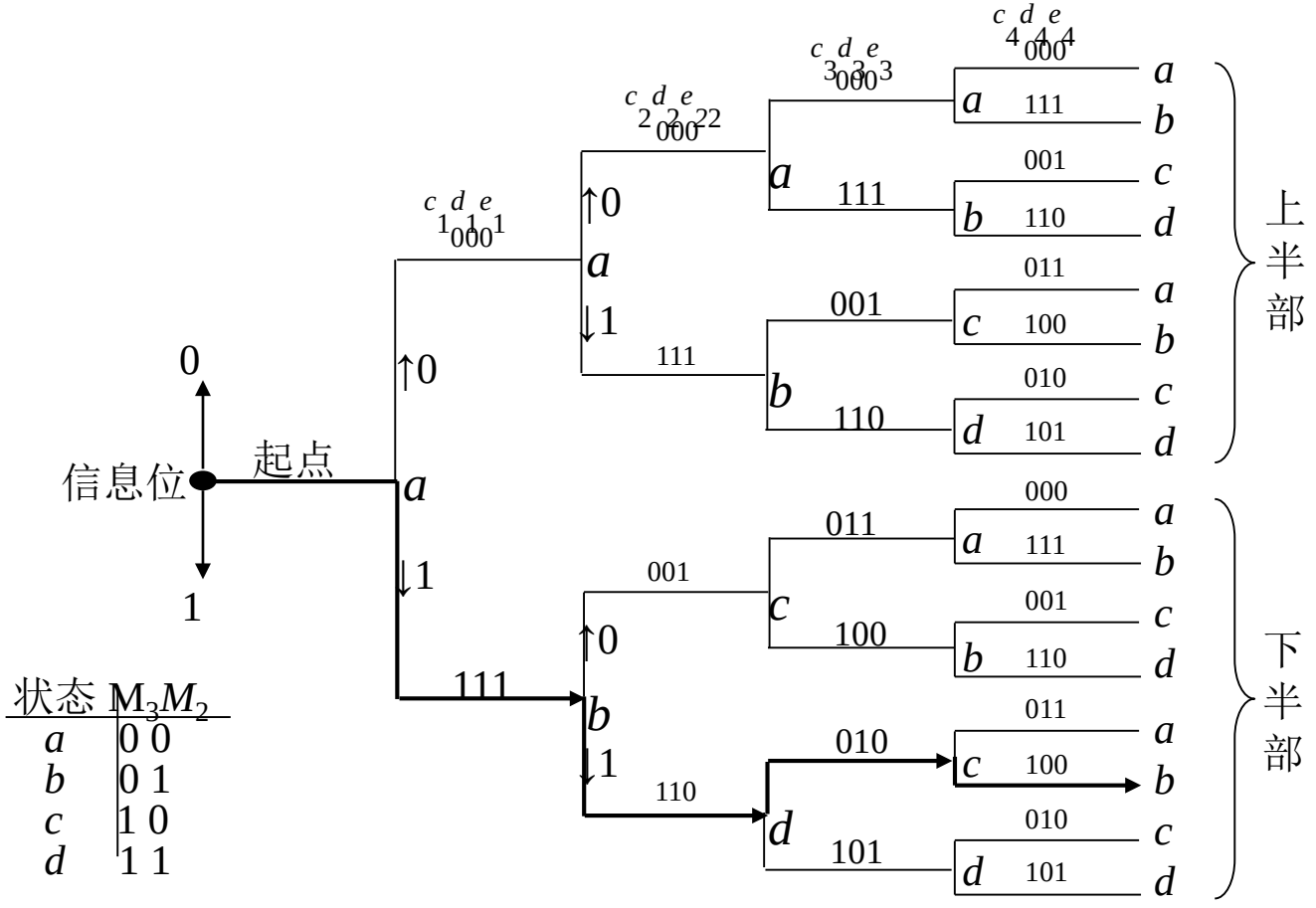
由此可见，当校正子序列中出现第一个“1”时，表示已经检出一个错码。后面的几位校正子则指出是信息位错了，还是监督位错了。

图中门限电路的输入代表式中4个方程的4个电压。门限电路将这4个电压（非模2）相加。当相加结果大于或等于3时，门限电路输出“1”，它除了送到输出端的模2加法器上纠正输出码元b_1的错码外，还送到校正子移存器纠正其中错误。

- 此卷积码除了能够纠正两位在约束长度中的随机错误外，还能纠正部分多于两位的错误。

第11章差错控制编码

- 卷积码的几何表述
  - 码树图：现仍以上面(3, 1, 3)码为例，介绍卷积码的码树



第11章差错控制编码

将图中移存器M_1，M_2和M_3的初始状态000作为码树的起点。现在规定：输入信息位为“0”，则状态向上支路移动；输入信息位为“1”，则状态向下支路移动。于是，就可以得出图中所示的码树。

设现在的输入码元序列为1101，则当第1个信息位$b_1 = 1$输入后，各移存器存储的信息分别为$M_1 = 1$，$M_2 = M_3 = 0$。此时的输出为$c_1\ d_1\ e_1 = 111$，码树的状态将从起点a向下到达状态b；此后，第2个输入信息位$b_2 = 1$，故码树状态将从状态b向下到达状态d。这时$M_2 = 1$，$M_3 = 0$，此时，$c_2 d_2 e_2 = 110$。第3位和后继各位输入时，编码器将按照图中粗线所示的路径前进，得到输出序列：111 110 010 100 …。

由此码树图还可以看到，从第4级支路开始，码树的上半部和下半部相同。这意味着，从第4个输入信息位开始，输出码元已经与第1位输入信息位无关，即此编码器的约束度$N = 3$。

第11章差错控制编码

若观察在新码元输入时编码器的过去状态，即观察M_2 M_3的状态和输入信息位的关系，则可以得出图中的*a* *b* *c*和*d*四种状态。这些状态和M_2 M_3的关系也在图中给出了。

- 码树图原则上还可以用于解码。在解码时，按照汉明距离最小的准则沿上面的码树进行搜索。例如，若接收码元序列为111 010 010 110 …，和发送序列相比可知第4和第11码元为错码。当接收到第4～6个码元“010”时，将这3个码元和对应的第2级的上下两个支路比较，它和上支路“001”的汉明距离等于2，和下支路“110”的汉明距离等于1，所以选择走下支路。

  类似地，当接收到第10～12个码元“110”时，和第4级的上下支路比较，它和上支路的“011”的汉明距离等于2，和下支路“100”的汉明距离等于1，所以走下支路。这样，就能够纠正这两个错码。

第11章差错控制编码

一般说来，码树搜索解码法并不实用，因为随着信息序列的增长，码树分支数目按指数规律增长；在上面的码树图中，只有4个信息位，分支已有$2^4 = 16$个。但是它为以后实用解码算法建立了初步基础。

第11章差错控制编码

- 状态图

上面的码树可以改进为下述的状态图。

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。

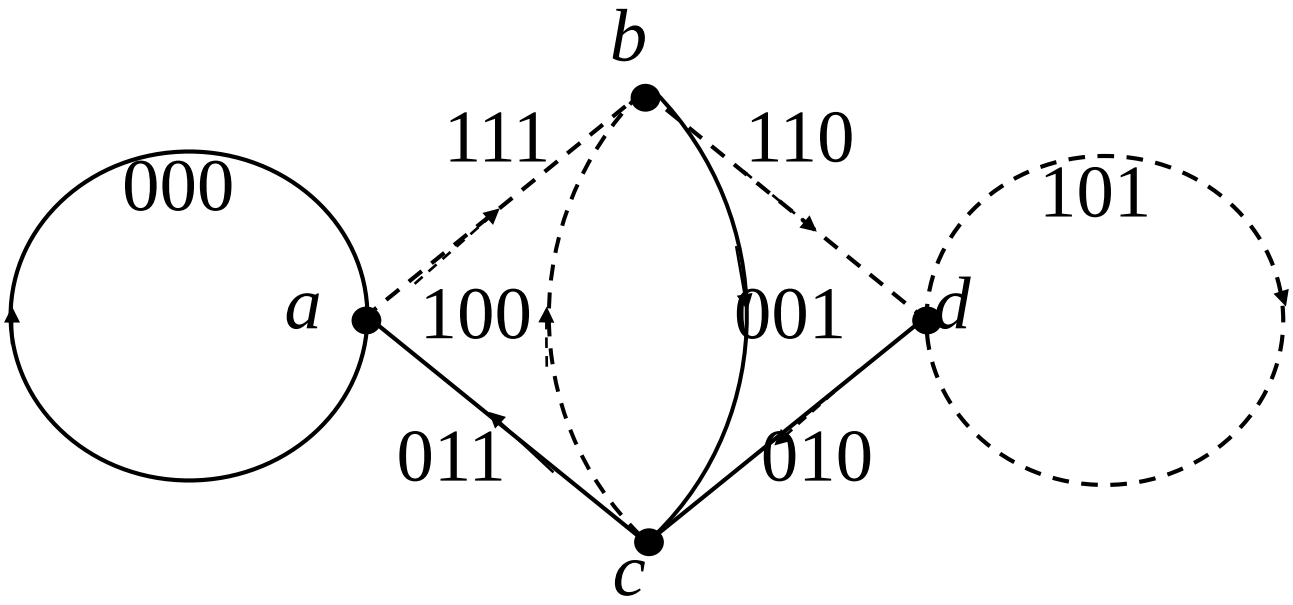
第11章差错控制编码

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	000	a (00)
	1	111	b (01)
b (01)	0	001	c (10)
	1	110	d (11)
c (10)	0	011	a (00)
	1	100	b (01)
d (11)	0	010	c (10)
	1	101	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。

第11章差错控制编码

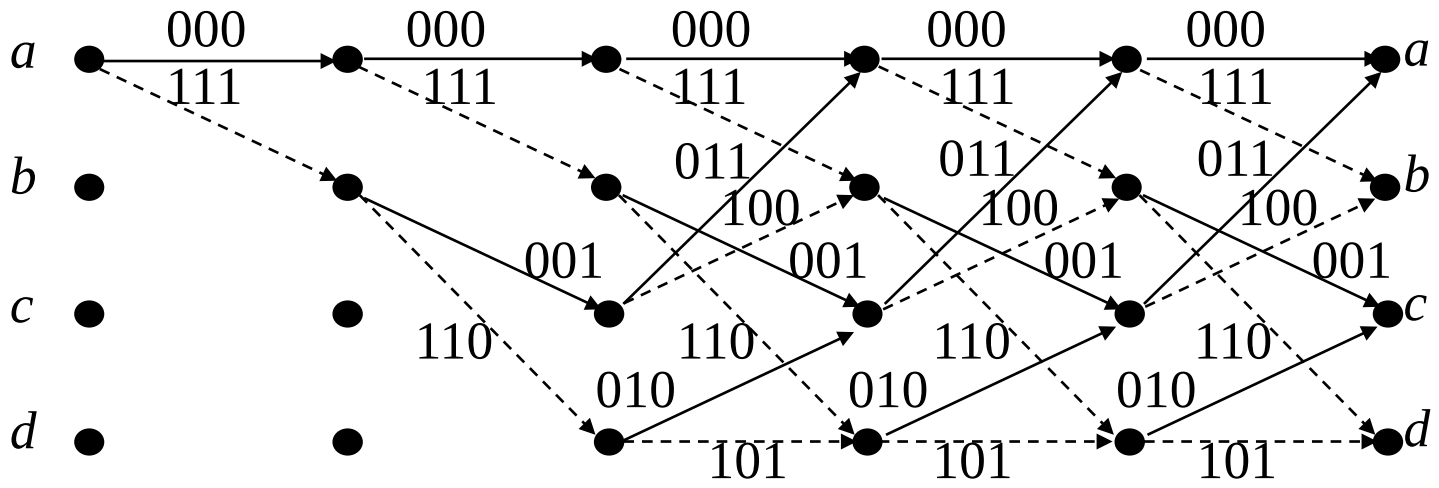


在此图中，虚线表示输入信息位为“0”时状态转变的路线；实线表示输入信息位为“1”时状态转变的路线。线条旁的3位数字是编码输出比特。利用这种状态图可以方便地从输入序列得到输出序列。

第11章差错控制编码

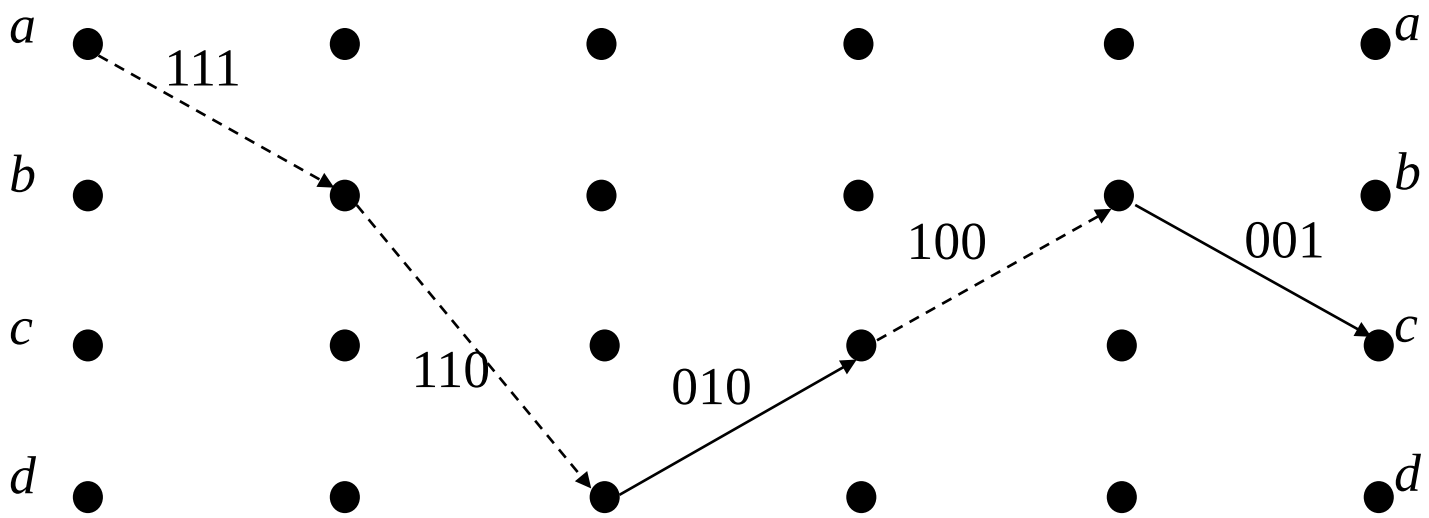
- 网格图

  将状态图在时间上展开，可以得到网格图如下：



图中画出了5个时隙。在此图中，仍用虚线表示输入信息位为“0”时状态转变的路线；实线表示输入信息位为“1”时状态转变的路线。可以看出，在第4时隙以后的网格图形完全是重复第3时隙的图形。这也反映了此(3, 1, 3)卷积码的约束长度为3。

第11章差错控制编码



在上图中给出了输入信息位为11010时，在网格图中的编码路径。图中示出这时的输出编码序列是：111 110 010 100 011…。由上述可见，用网格图表示编码过程和输入输出关系比码树图更为简练。

有了上面的状态图和网格图，下面就可以讨论维特比解码算法了。

第11章差错控制编码

- 维特比解码算法
  - 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。

    现在仍用上面(3, 1, 3)卷积码的例子来说明维特比算法的原理。

第11章差错控制编码

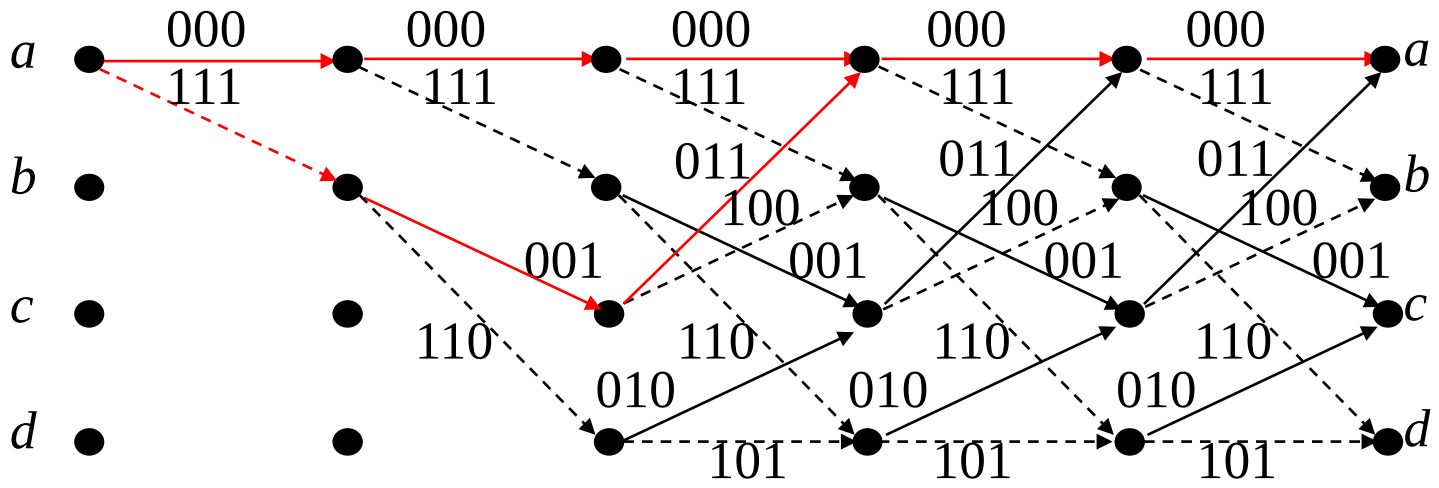
- 例： (3, 1, 3)卷积码

  设现在的发送信息位为1101，为了使图中移存器的信息位全部移出，在信息位后面加入3个“0”，故编码后的发送序列为111 110 010 100 001 011 000。并且假设接收序列为111 010 010 110 001 011 000，其中第4和第11个码元为错码。

  由于这是一个$(n, k, N) = (3, 1, 3)$卷积码，发送序列的约束度$N = 3$，所以首先需考察$nN = 9$比特。第1步考察接收序列前9位“111 010 010”。由此码的网格图可见，沿路径每一级有4种状态a, b, c和d。每种状态只有两条路径可以到达。故4种状态共有8条到达路径。

  现在比较网格图中的这8条路径和接收序列之间的汉明距离。

第11章差错控制编码



例如，由出发点状态*a*经过3级路径后到达状态*a*的两条路径中上面一条为“000 000 000”。它和接收序列“111 010 010”的汉明距离等于5；下面一条为“111 001 011”,它和接收序列的汉明距离等于3。同样，由出发点状态*a*经过3级路径后到达状态*b*、*c*和*d*的路径分别都有两条，故总共有8条路径。在下表中列出了这8条路径和其汉明距离。

第11章差错控制编码

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

现在将到达每个状态的两条路径的汉明距离作比较，将距离小的一条路径保留，称为幸存路径。若两条路径的汉明距离相同，则可以任意保存一条。这样就剩下4条路径了，即表中第2, 4, 6和8条路径。

第11章差错控制编码

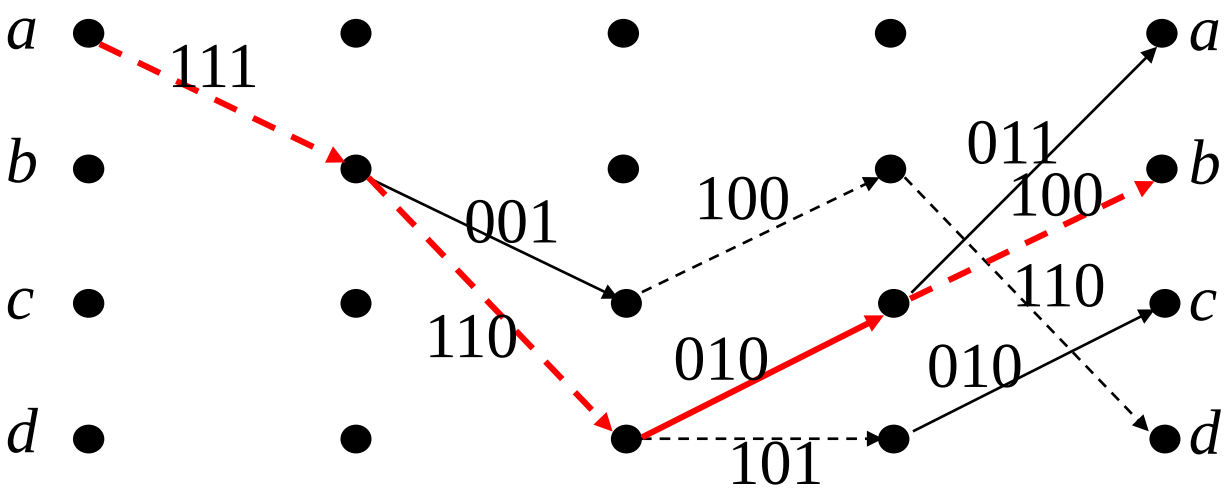
第2步继续考察接收序列的后继3个比特“110”。计算4条幸存路径上增加1级后的8条可能路径的汉明距离。结果如下表。

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

表中最小的总距离等于2，其路径是abdc+b，相应序列为111 110 010 100。它和发送序列相同，故对应发送信息位1101。

按照表中的幸存路径画出的网格图示于下图中。

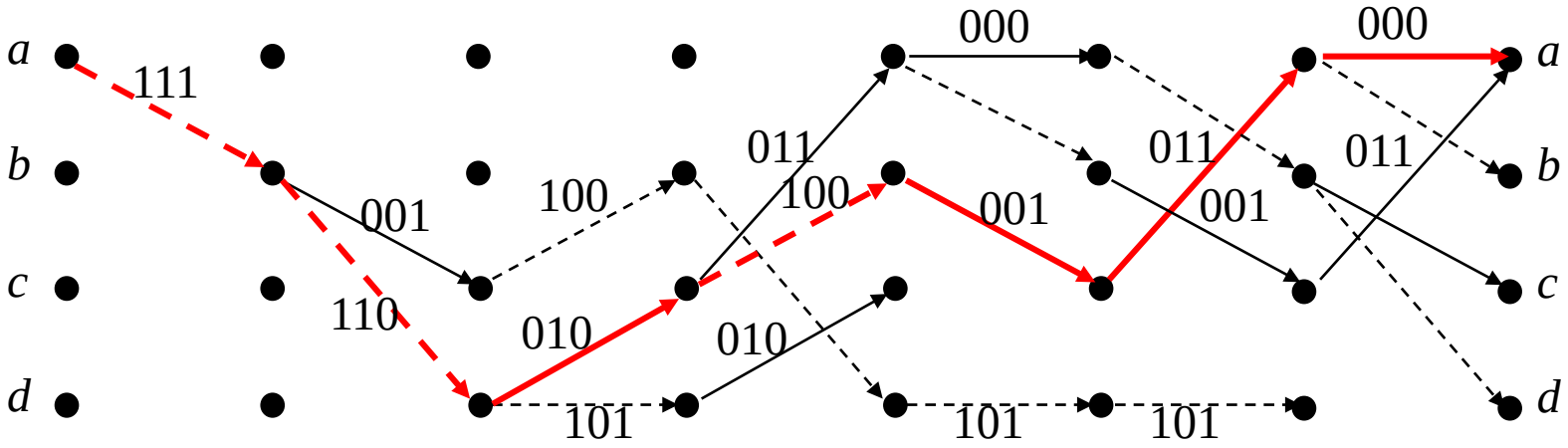
第11章差错控制编码



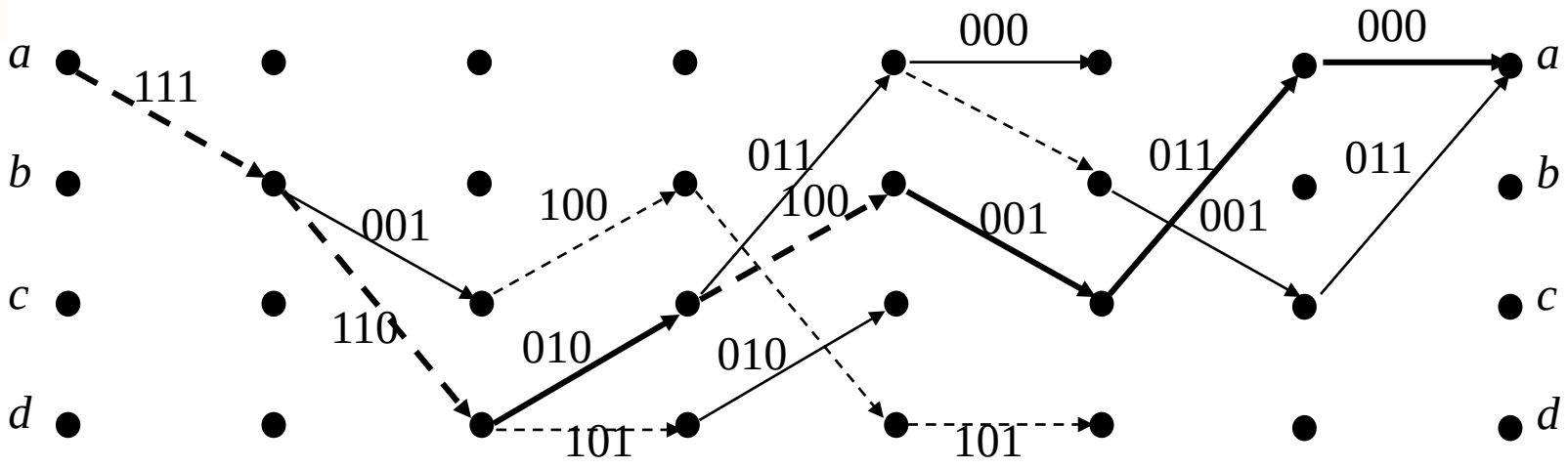
图中粗红线路径是汉明距离最小（等于2）的路径。

第11章差错控制编码

上面提到过，为了使输入的信息位全部通过编码器的移存器，使移存器回到初始状态，在信息位1101后面加了3个“0”。若把这3个“0”仍然看作是信息位，则可以按照上述算法继续解码。这样得到的幸存路径网格图示于下图中。



图中的粗红线仍然是汉明距离最小的路径。但是，若已知这3个码元是（为结尾而补充的）“0”，则在解码计算时就预先知道在接收这3个“0”码元后，路径必然应该回到状态*a*。而由图可见，只有两条路径可以回到a状态。所以，这时上图可以简化成下图。



在上例中卷积码的约束度$N=3$，需要存储和计算8条路径的参量。由此可见，维特比解码算法的复杂度随约束长度N按指数形式2^N增长。故维特比解码算法适合约束度较小（$N \leq 10$）的编码。对于约束度大的卷积码，可以采用其他解码算法。

第11章差错控制编码

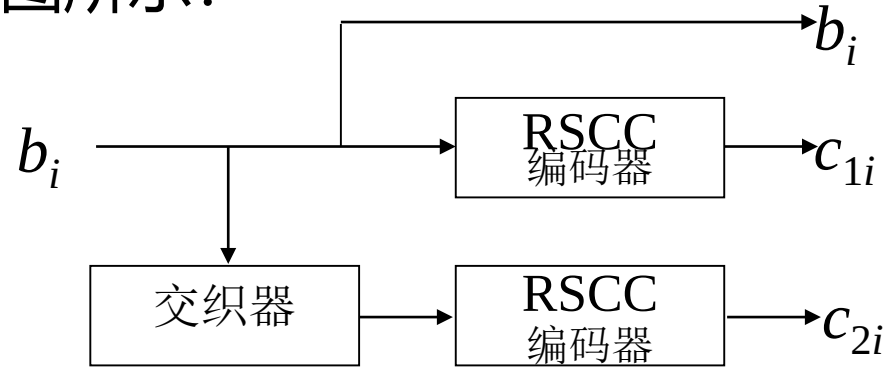
11.8 Turbo码

- 什么是Turbo码？它是一种特殊的链接码。由于其性能接近信息理论上能够达到的最好性能，所以在编码理论上是带有革命性的进步。这种码，特别是解码运算，非常复杂，这里只对其基本概念作一简明介绍。
- 基本原理
  - 由于分组码和卷积码的复杂度随码组长度或约束度的增大按指数规律增长，所以为了提高纠错能力，人们大多不是单纯增大一种码的长度，而是将两种或多种简单的编码组合成复合编码。
  - Turbo码的编码器在两个并联或串联的分量码编码器之间增加一个交织器，使之具有很大的码组长度，能在低信噪比条件下得到接近理想的性能。

第11章差错控制编码

- Turbo码的译码器有两个分量码译码器，译码在两个分量译码器之间进行迭代译码，故整个译码过程类似涡轮（turbo）工作，所以又形象地称为Turbo码。

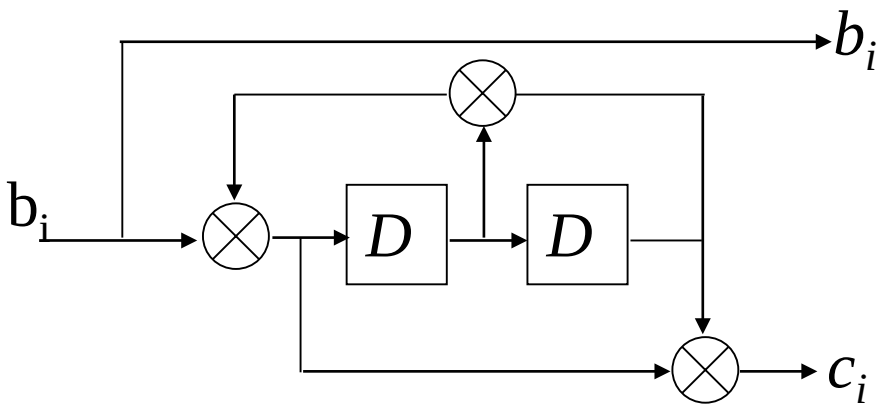
- 编码器的基本结构：它由一对递归系统卷积码(RSCC)编码器和一个交织器组成，如下图所示：



  - RSCC编码器和卷积码编码器之间的主要区别是从移存器输出端到信息位输入端之间有反馈路径。原来的卷积码编码器像是一个FIR数字滤波器。增加了反馈路径后，它就变成了一个IIR滤波器，或称递归滤波器。
  - 两个RSCC编码器是相同的。它们的输入经过一个交织器并联。此Turbo码的输入信息位是b_i，输出是$b_i c_{1i} c_{2i}$，故码率等于1/3。

第11章差错控制编码

- RSCC编码器举例：
  - 方框图：如下图所示



  - 它是一个码率等于1/2的卷积码编码器，输入为b_i，输出为$b_i c_i$。
  - 因为输出中第1位是信息位，所以它是系统码。

第11章差错控制编码

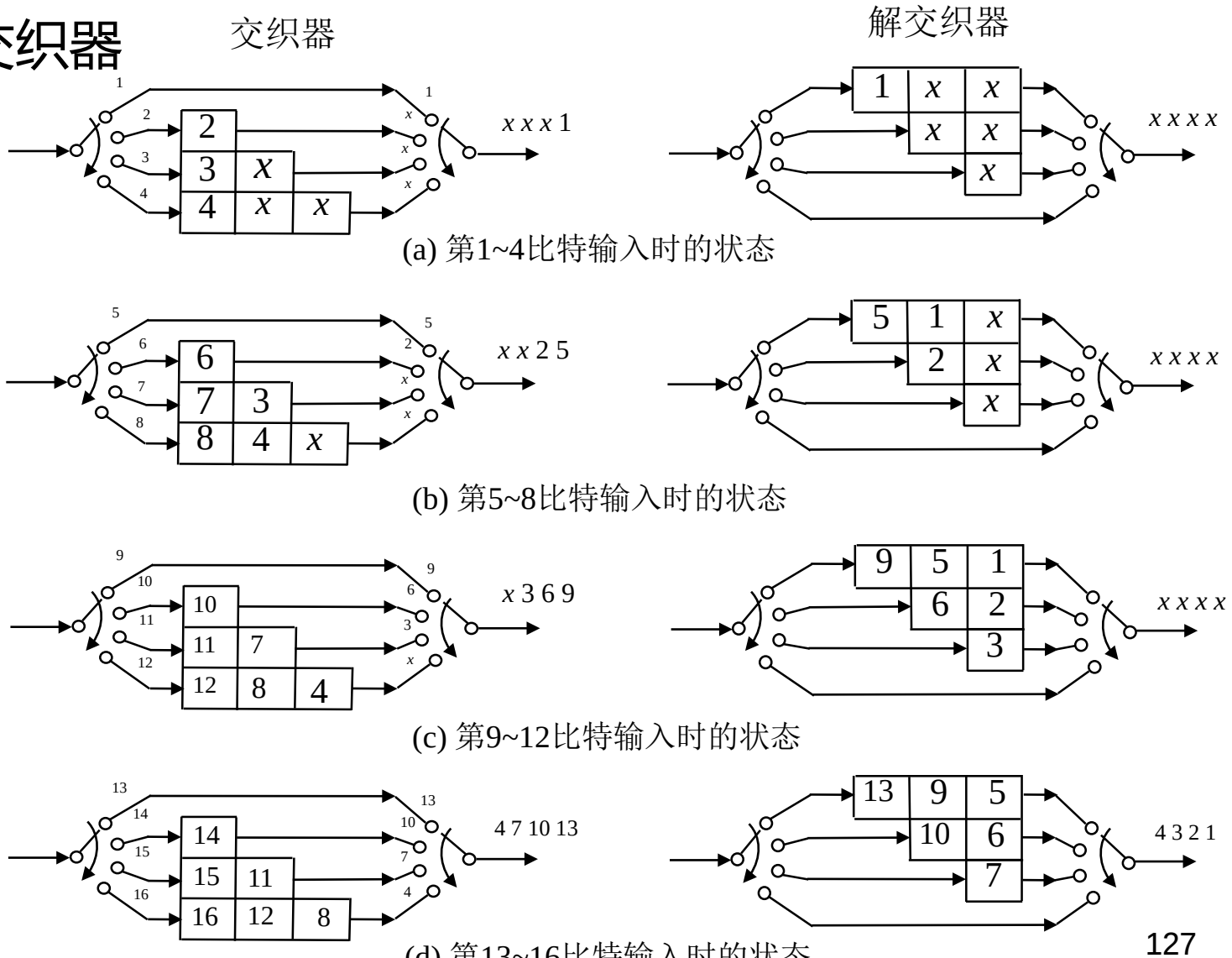
- 矩阵交织器：
  - 原理方框图：见右图
  - 其基本形式是矩阵交织器，它由容量为$(n\text{-}1)m$比特的存储器构成。将信号码元按行的方向输入存储器，再按列的方向输出。

a_{11}	a_{12}	…	…	…	a_{1m}
a_{21}	a_{22}	…	…	…	a_{2m}
…	…	…	…	…	…
a_{n1}	a_{n2}	…	…	…	a_{nm}

  - 若输入码元序列是：$a_{11}a_{12}\ldots a_{1m}\ a_{21}a_{22}\ldots a_{2m}\ldots a_{n1}\ldots a_{nm}$，则输出序列是：$a_{11}a_{21}\ldots a_{n1}a_{12}a_{22}\ldots a_{n2}\ldots a_{1m}\ldots a_{nm}$。
  - 交织的目的是将突发错码分散开，变成随机错码。例如，若图中第1行的m个码元构成一个码组，并且连续发送到信道上，则当遇到脉冲干扰，造成大量错码时，可能因超出纠错能力而无法纠正错误。但是，若在发送前进行了交织，按列发送，则能够将集中的错码分散到各个码组，从而有利于纠错。这种交织器常用于分组码。

第11章差错控制编码

- 卷积交织器
  - 方框图举例



第11章差错控制编码

- 它是由3个移存器构成。第1个移存器只有1比特容量；第2个移存器可以存2比特；第3个移存器可以存3比特。
- 交织器的输入码元依次进入各个移存器。
- 在图 (a)的交织器中示出，第1个输入码元没有经过存储而直接输出；第2个输入码元存入第1个移存器中；第3个输入码元存入第2个移存器中；第4个码元存入第3个移存器中。在这4个码元期间，交织器的输出为“1 *x x x*”。这里的“*x*”表示移存器初始的随机状态。
- 在图 (b)中的交织器则示出第5至8个码元输入时的工作状态。
- 在图 (c)和 (d)中示出的是第9至12个码元以及第13至16个码元输入时的工作状态。
- 这样，交织器输出码元的次序将是：1 *x x x* 5 2 *x x* 9 6 3 *x* 13 10 7 4。

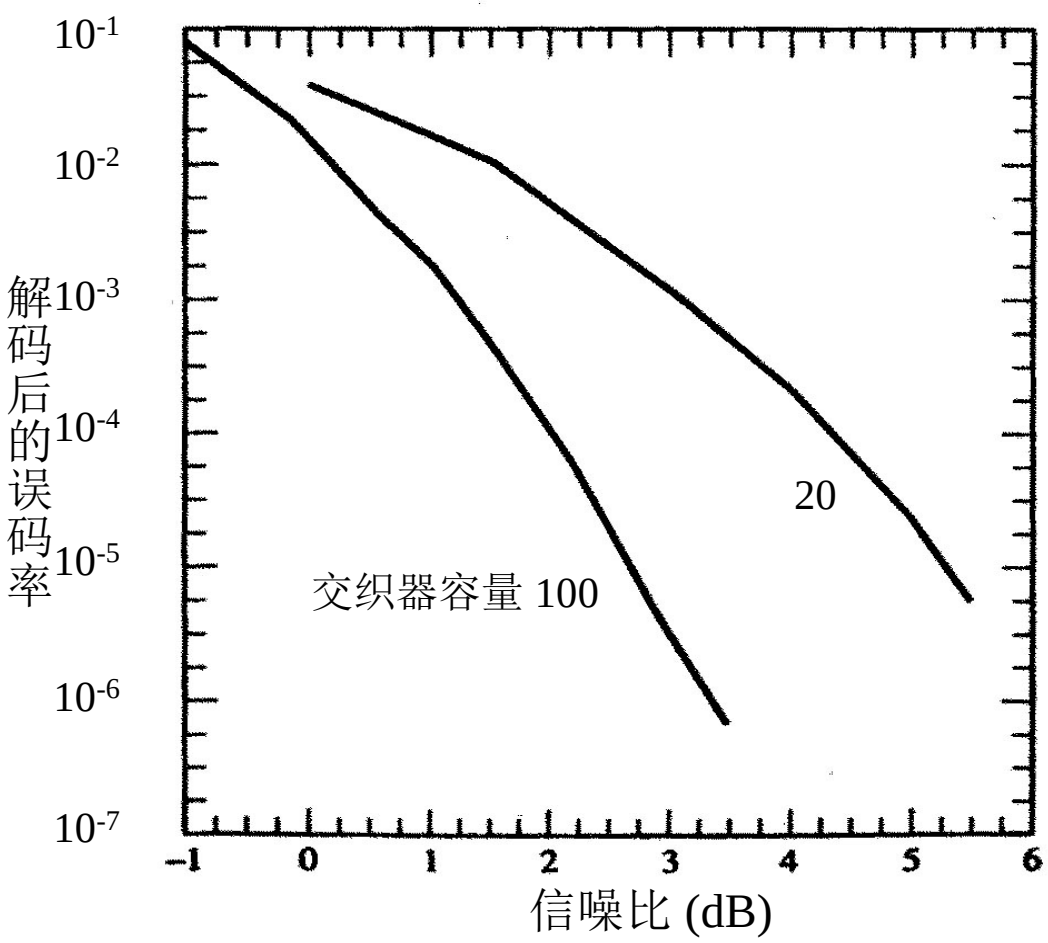
第11章差错控制编码

- 接收端解交织器的工作过程与此相反，如图所示，解交织器的输出码元的次序将是：$x\,x\,x\,x\,x\,x\,x\,x\,x\,x\,x\,x\,1\,2\,3\,4$，其中前面接收的12个码元无意义，从第13个码元开始才是有效码元。

- 一般说来，第1个移存器的容量可以是k比特，第2个移存器的容量是$2k$比特，第3个移存器的容量是$3k$比特，…，直至第N个移存器的容量是Nk比特。
- 卷积交织法和矩阵交织法相比，主要优点是延迟时间短和需要的存储容量小。卷积交织法端到端的总延迟时间和两端所需的总存储容量均为$k(N+1)N$个码元，是矩阵交织法的一半。

第11章差错控制编码

- 交织器容量和误码率关系
  - 由此曲线可以看到，交织器容量大时误码率低，这是因为交织范围大可以使交织器输入码元得到更好的随机化。



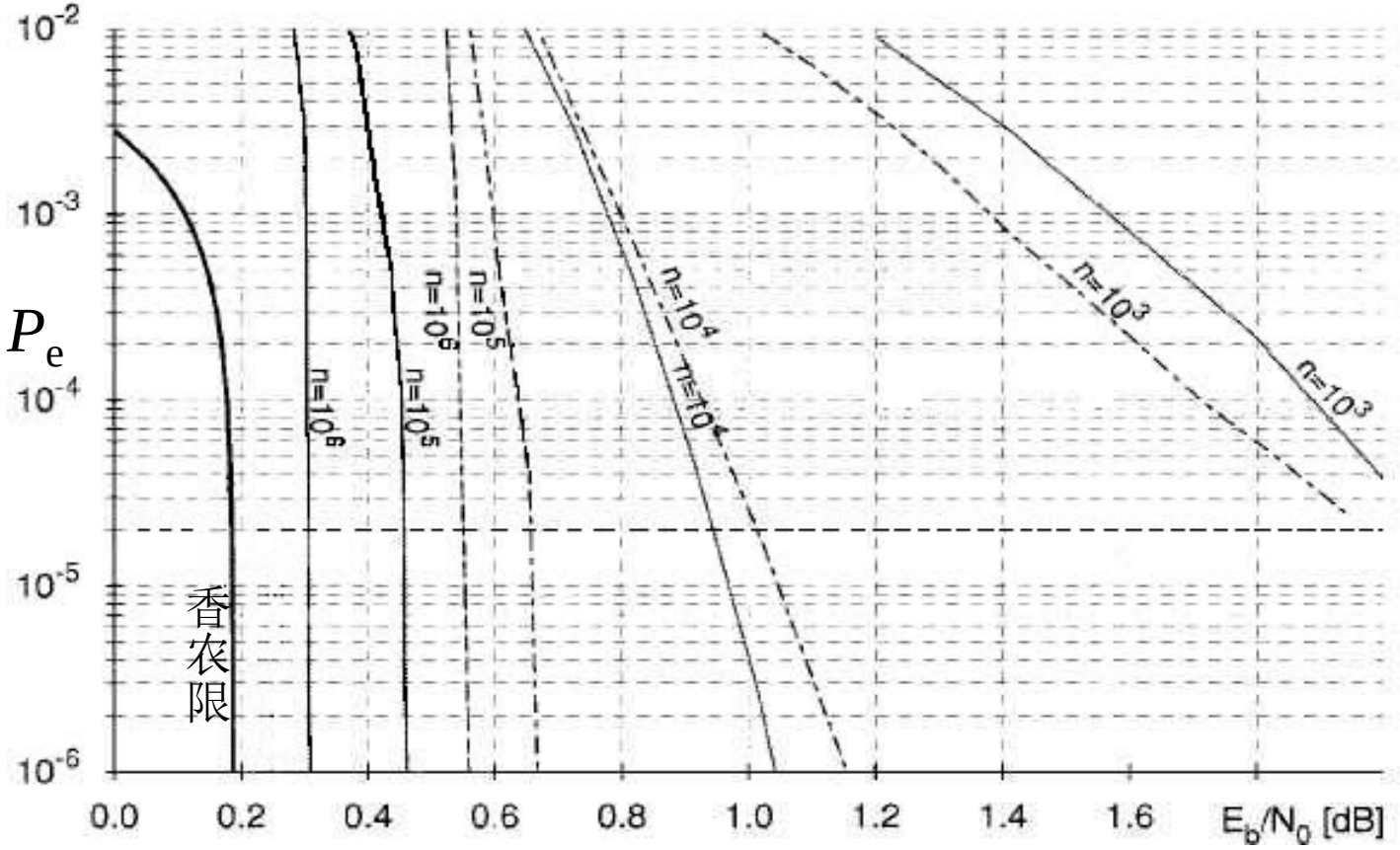
第11章差错控制编码

11.9 低密度奇偶校验码

- 低密度奇偶校验(LDPC)码是一种线性分组码，和Turbo码同属于复合码类。两者的性能相近，且两者的译码延迟都相当长，所以它们更适用于一些实时性要求不很高的通信。但是LDPC码比Turbo码的译码简单，更易实现。
- LDPC码的分类：
  - **规则LDPC码：** ***H***矩阵每列具有相同个数的“1”
  - **非规则LDPC码：** ***H***矩阵每列中 “1”的个数不一定相同
  - 非规则LDPC码是在规则LDPC码基础上发展出的，它使解码性能得到改善，使误码率性能比Turbo码还好。

第11章差错控制编码

- 非规则LDPC码和Turbo码的误比特率性能比较



图中的虚线是Turbo码的性能，实线是LDPC码的性能

当码长n大约在10^4以上时，LDPC码的性能才比Turbo码好。

第11章差错控制编码

- LDPC码的构造：
  - LDPC码和普通的奇偶监督码一样，可以由有n列、m行的奇偶监督矩阵$\boldsymbol{H}$确定；n是码长，m是校正子个数。但是其$\boldsymbol{H}$矩阵和普通奇偶监督码的有所不同：
    - 首先它是稀疏矩阵，即矩阵中“1”的个数很少，密度很低；设$\boldsymbol{H}$矩阵每列有j个“1”，每行有k个“1”，则应有$j << m$，$k << n$，且$j \geq 3$。
    - 其次其$\boldsymbol{H}$矩阵的任意两行的元素不能在相同位置上为“1”，即$\boldsymbol{H}$矩阵中没有四角由“1”构成的矩形。
  - LDPC码通常用上述3个参量(n, j, k)表示。在编码时，设计好H矩阵后，由H矩阵可以导出生成矩阵G。这样，对于给定的信息位，不难算出码组。

第11章差错控制编码

- LDPC码的解码方法

  LDPC码的解码方法也和一般的奇偶监督码的解码方法不同。基本的解码算法称为置信传播算法，通常简称BP算法。这种算法实质上是求最大后验概率，类似于一般的最大似然准则解码算法，但是它需要进行多次迭代运算，逐步逼近最优的解码值。

  LDPC码的具体编解码算法十分复杂，这里不再深入讨论。

第11章差错控制编码

11.10 网格编码调制

11.10.1网格编码调制(TCM)的基本概念

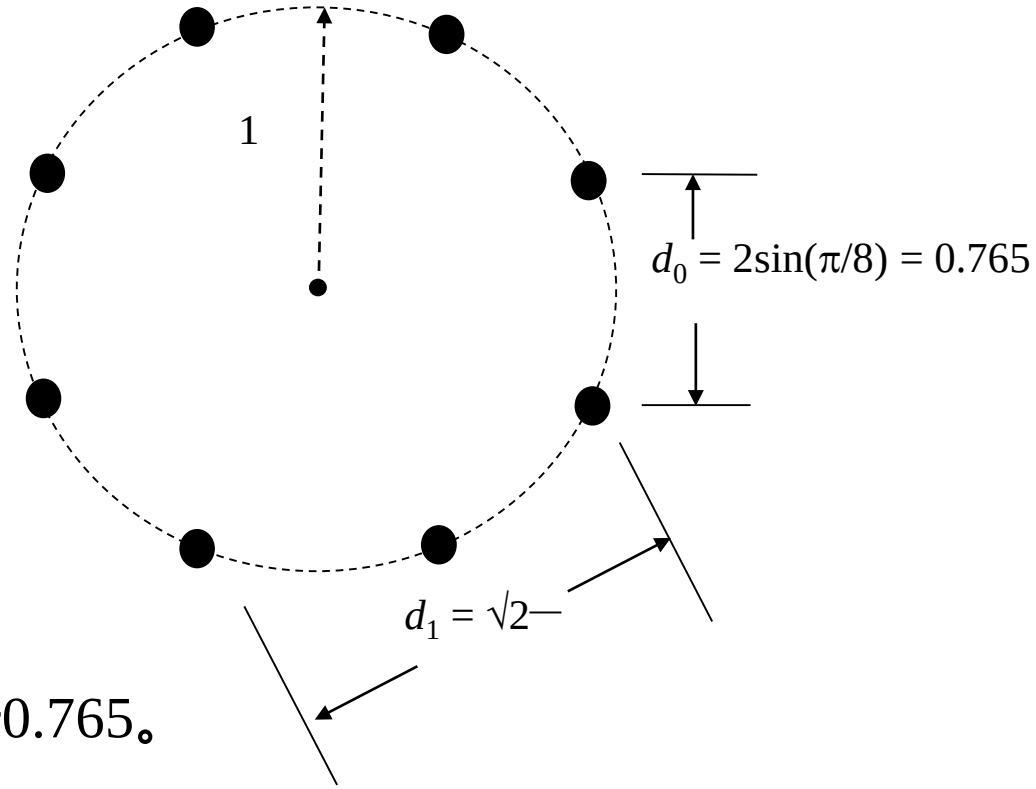
下面将利用一个实例给出TCM的基本概念

- 复习QPSK系统：

  QPSK是一个4相相移键控系统，它的每个码元传输2 比特信息。若在接收端判决时因干扰而将信号相位错判至相邻相位，则将出现错码。现在，将系统改成8PSK，它的每个码元可以传输3 比特信息。但是我们仍然令每个码元传输2 比特信息。第3 比特用于纠错码，例如，采用码率为2/3的卷积码。这时接收端的解调和解码是作为一个步骤完成的，不像传统作法，先解调得到基带信号后再为纠错去解码。

第11章差错控制编码

- 在纠错编码理论中，码组间的最小汉明距离决定着这种编码的纠错能力。在TCM中，由于是直接对于已调信号（现在是8PSK信号）解码，码元之间的差别是载波相位之差，这个差别是欧氏距离。
- 在右图中示出了8PSK信号星座图中的8个信号点。图中已假设信号振幅等于1，则相邻两信号点的欧氏距离d_0等于0.765。

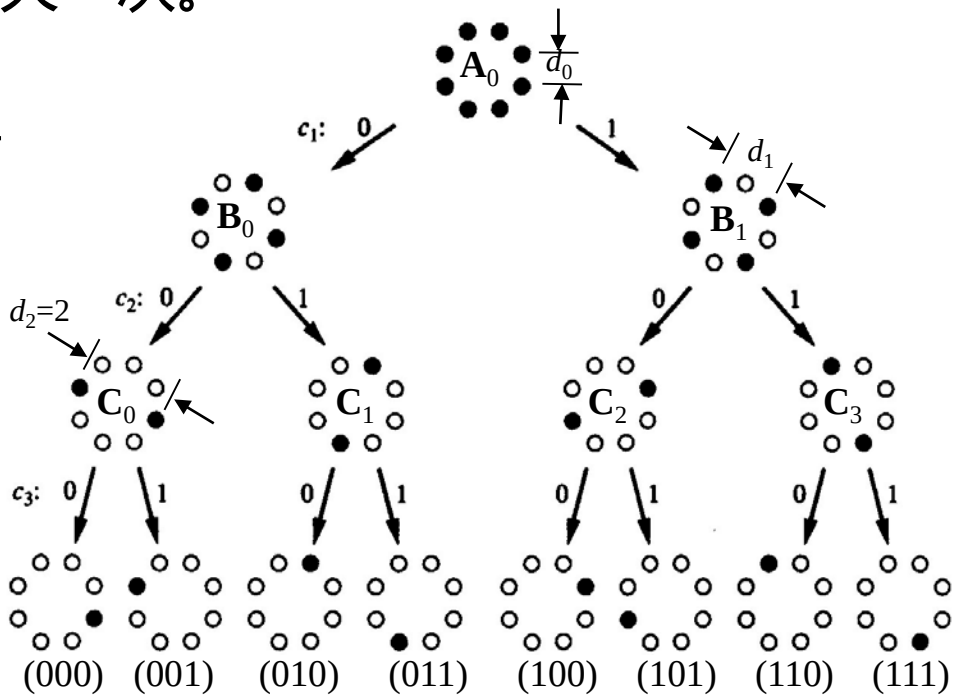


第11章差错控制编码

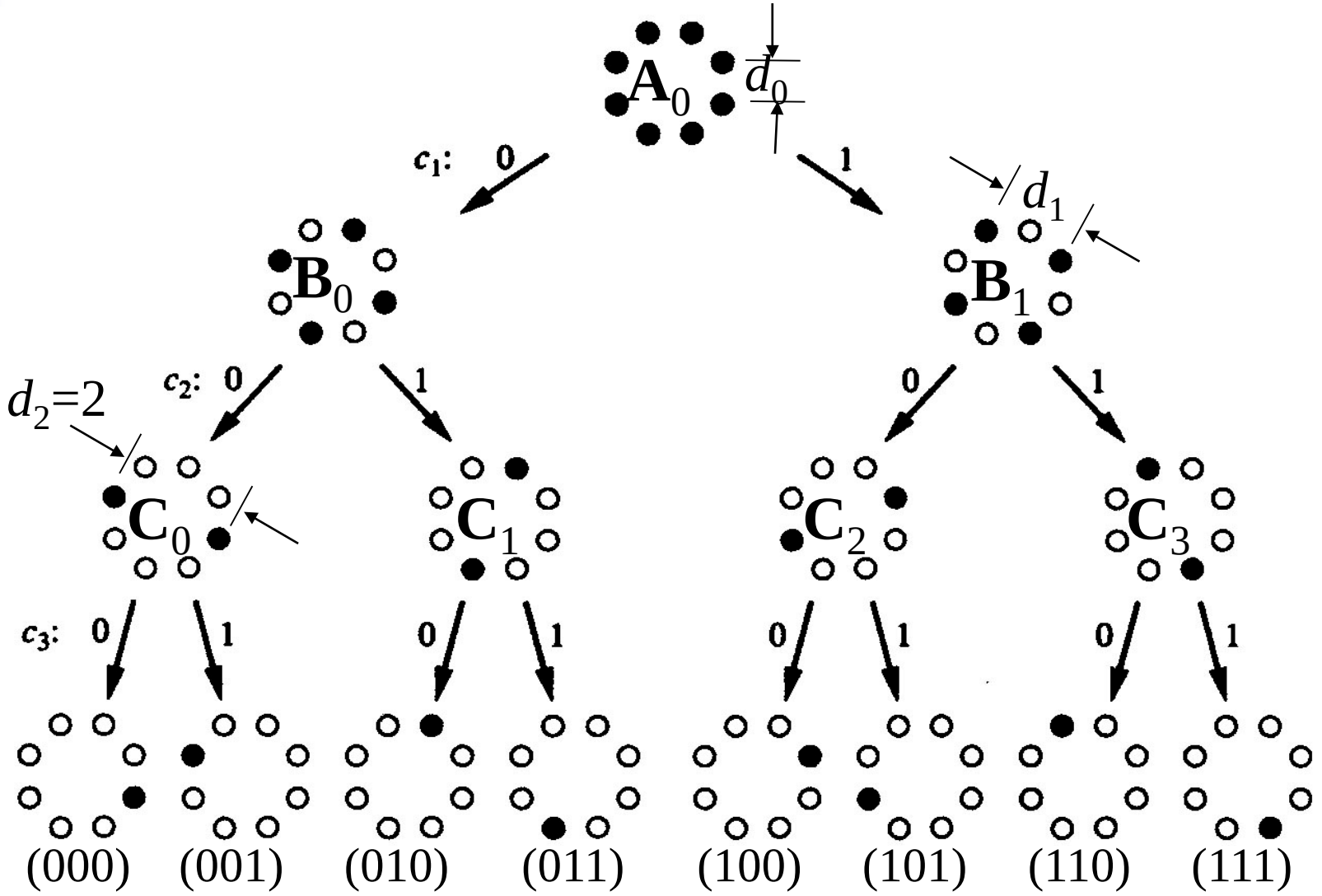
- 两个信号序列的欧氏距离越大，即它们的差别越大，则因干扰造成互相混淆的可能性越小。
- 图中的信号点代表某个确定相位的已调信号波形。
- 为了利用卷积码维特比解码的优点，这时仍然需要用到网格图。但是，和卷积码维特比解码时的网格图相比，在TCM中是将这些波形映射为网格图，故TCM网格图中的各状态是波形的状态。

第11章差错控制编码

- 11.10.2 TCM信号的产生
  - 集划分方法
    - 基本原则：将信号星座图划分成若干子集，使子集中的信号点间距离比原来的大。每划分一次，新的子集中信号点间的距离就增大一次。
    - 例：见右图
      - $\mathbf{A}_0$是8PSK信号的星座图，其中任意两个信号点间的距离为d_0。这个星座被划分为$\mathbf{B}_0$和$\mathbf{B}_1$两个子集，在子集中相邻信号点间的距离为d_1。



第11章差错控制编码

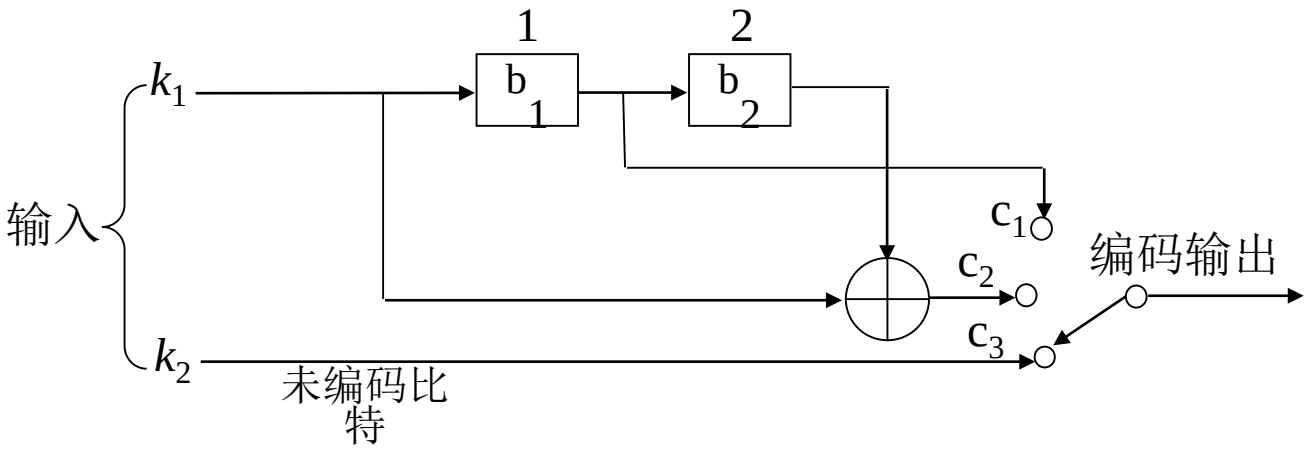

A0
d0
c1: 0
1
d1
B0
B1
d2=2
c2: 0
1
0
1
C0
C1
C2
C3
c3: 0
1
0
1
0
1
0
1
(000)
(001)
(010)
(011)
(100)
(101)
(110)
(111)

第11章差错控制编码

- 在上图中已经示出$d_1 > d_0$。将这两个子集再划分一次，得到4个子集：$\boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3$，它们中相邻信号点间的距离为$d_2 = 2$。显然，$d_2 > d_1 > d_0$。
- 在这个例子中，需要根据已编码的3个比特来选择信号点，即选择波形的相位。
- c_1, c_2, 和c_3表示已编码的3个码元，图中最下一行注明了$(c_1c_2c_3)$的值。若c_1等于“0”，则从A_0向左分支走向B_0；若c_1等于“1”，则从A_0向右分支走向B_1。第2和3个码元c_2和c_3也按照这一原则选择下一级的信号点。

第11章差错控制编码

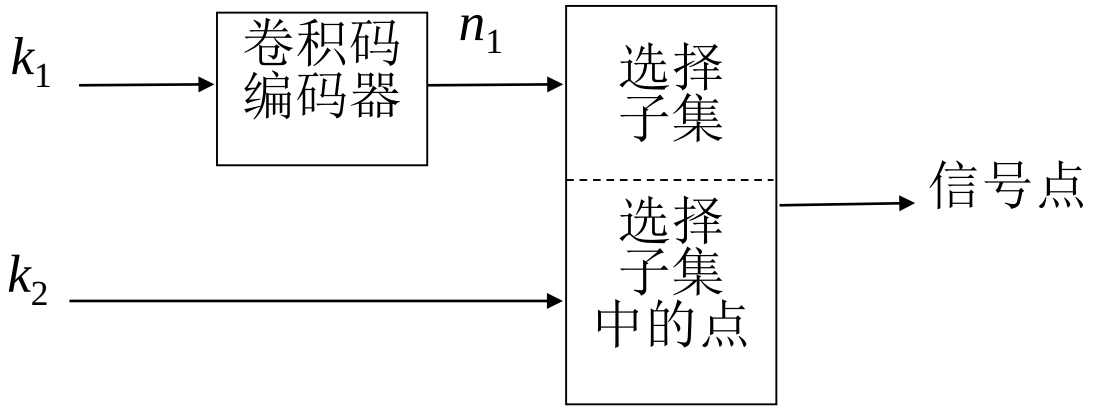
- 卷积码编码器的方框图：



由上图可见，这个卷积码的约束长度等于3。编码器输出的前两个比特c_1和c_2用来选择星座图划分的路径，最后1个比特c_3用于选定星座图第3级（最低级）中的信号点。

第11章差错控制编码

- TCM编码器结构
  - 方框图
  - 原理：



将k比特输入信息段分为k_1和k_2两段；前k_1比特通过一个(n_1, k_1, m)卷积码编码器，产生n_1比特输出，用于选择信号星座图中划分之一，后面的k_2比特用于选定星座图中的信号点。

这表明星座图被划分为2^{n_1}个子集，每个子集中含有个信号点。

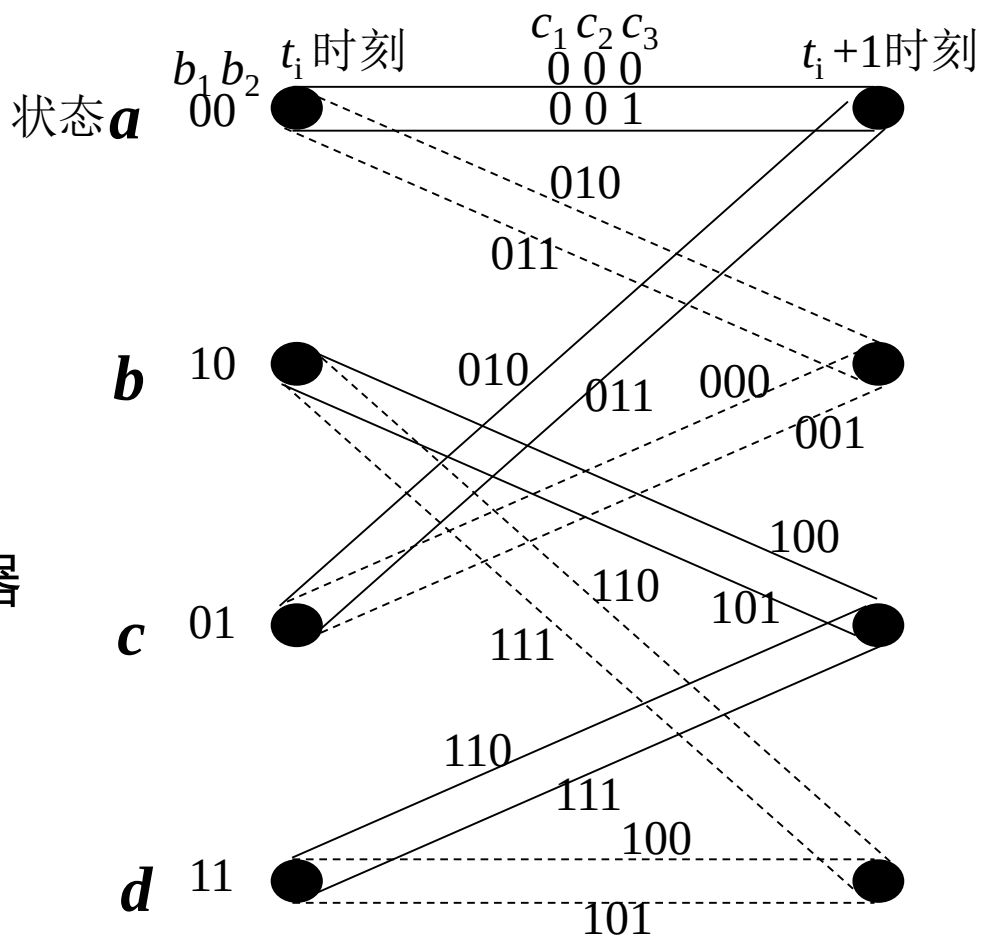
在上例编码器方框图中$k_1 = k_2 = 1$

第11章差错控制编码

- TCM系统8PSK的网格图

由于未编码比特有两种取值，所以每个状态下，有两根线。例如，设初始状态$b_1\ b_2 = 00$，$k_1 = k_2 = 0$。

当输入信号序列k_1为“0110100”时，移存器状态和输出c_1与c_2之间的关系示于下表中。



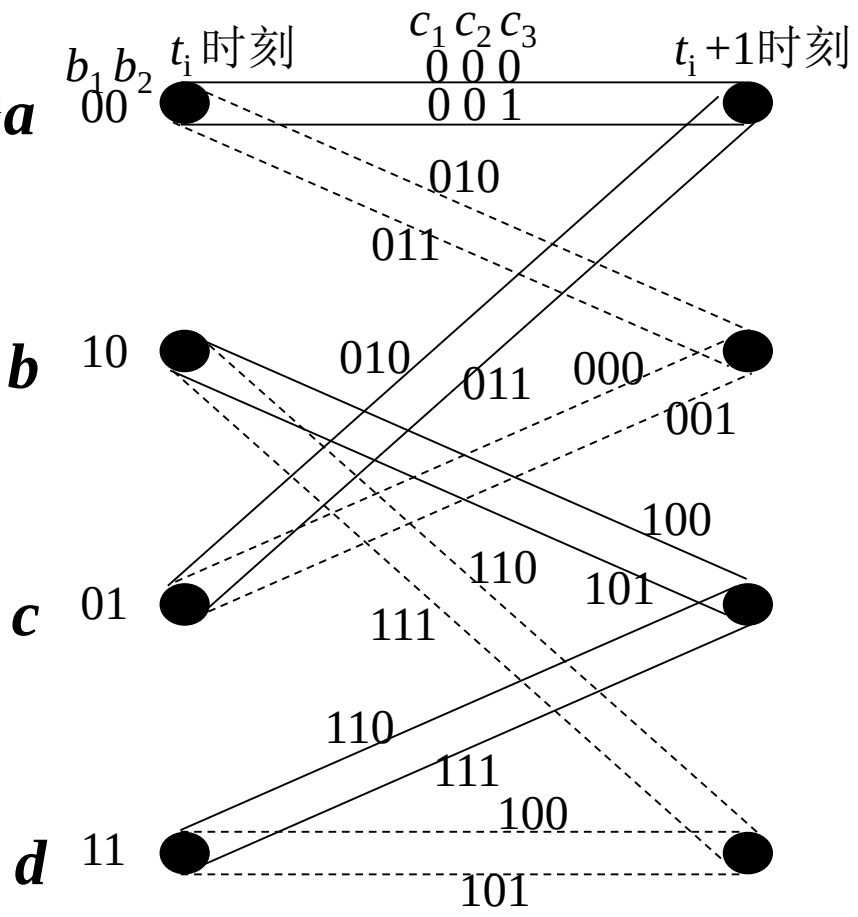
第11章差错控制编码

- 移存器状态和输出之间的关系

k_1	b_1	b_2	状 态	c_1	c_2
	0	0	***a***	0	0
0	0	0	***a***	0	0
1	0	0	***a***	0	1
1	1	0	***b***	1	1
0	1	1	***d***	1	1
1	0	1	***c***	0	0
0	1	0	***b***	1	0
0	0	1	***c***	0	1
0	0	0	***a***	0	0

第11章差错控制编码

- 在第1个输入码元“1”到达后，输出码元c_1和c_2由“00”变成“01”，但是这时的输入信息位k_2可能是“0”或“1”，所以输出$c_1\ c_2\ c_3$可能是“010”或“011”，这就是右图中最高的两条平行虚线。



- 在第1个输入码元“1”进入b_1后，$b_1\ b_2$的状态由“00”(***a***)变到“10”(***b***)，输出$c_1\ c_2\ c_3$可能是“110”或“111”，$b_1\ b_2$的状态由***b***变到***d***，如图中虚线所示。依此类推。

第11章差错控制编码

- 网格图和星座图之间的对应关系
  - 每对平行转移必须对应最下一级划分同一子集中的两个信号点。例如，图中的“000”和“001”同属于子集$\mathbf{C}_0$，“010”和“011”同属于子集$\mathbf{C}_1$，等等。这些对信号点具有最大的欧氏距离($d_2 = 2$)。
  - 从某一状态出发的所有转移，或到达某一状态的所有转移，必须属于同一上级子集。例如，图中从状态$\boldsymbol{a}$出发的转移“000”、“001”、“010”和“011”都属于子集$\mathbf{B}_0$。或者说，此两对平行转移应具有最大可能的欧氏距离。

第11章差错控制编码

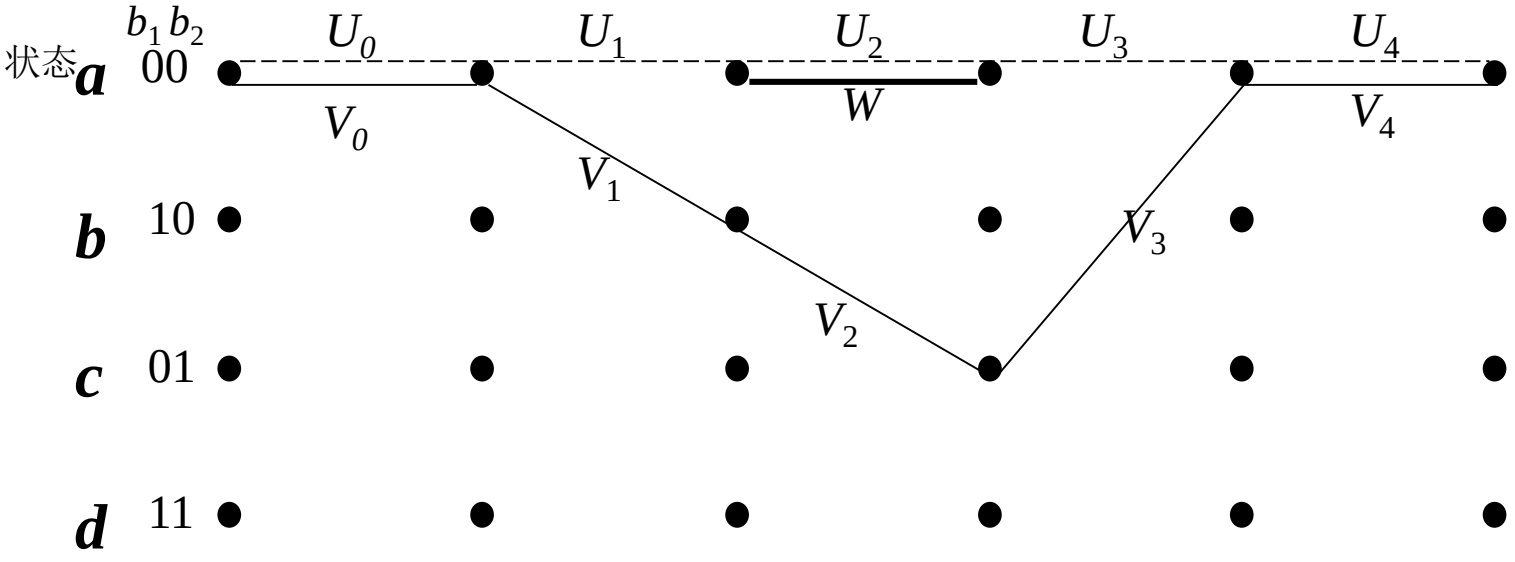
11.10.3 TCM信号的解调

TCM信号的解调算法

- 通常采用维特比算法，但是现在的网格图表示的状态是波形，而不是码组。解码器的任务是计算接收信号序列路径和各种可能的编码网格路径间的距离。若所有发送信号序列是等概率的，则判定与接收序列距离最小的可能路径（又称为最大似然路径）为发送序列。
- 因为卷积码是线性码，它具有封闭性，故要考察的路径距离与所用的测试序列无关。所以，不失一般性，可以选用全“0”序列作为测试序列。

第11章差错控制编码

- 例：8PSK信号解码路径
  - 用全“0”序列作为测试序列时，如下图中虚线路径U所示。图中还用实线示出另一许用波形序列路径V，它从全“0”序列路径分开又回到全“0”序列路径。
  - 若发送序列是全“0”序列，但是接收序列有错误，使接收序列路径离开全“0”路径然后又回到全“0”序列，且中间没有返回状态***a***，则解码器需要比较此接收序列路径和U的距离与接收序列路径和V的距离之大小。若后者小，则将发生一次错误判决。这里的距离是指欧氏距离。



第11章差错控制编码

- 自由欧氏距离Fed:
  - 自由欧氏距离是指许用波形序列集合中各元素之间的最小距离。它决定了产生错误判决的概率。自由欧氏距离越大，错误判决概率越小。
  - 在上例中，U和V两条路径间的欧氏距离d由下式决定:

$$\begin{aligned} d^2 &= d^2(U_1,V_1)+d^2(U_2,V_2)+d^2(U_3,V_3) \\ &= d^2(000,010)+d^2(000,100)+d^2(000,010) \\ &= \left(\sqrt{2}\right)^2+(0.765)^2+\left(\sqrt{2}\right)^2 = 2+0.585+2 = 4.585 \end{aligned}$$

  - 上式是按照在欧氏空间求矢量和的方法计算的。因此，

$$d = \sqrt{4.585} = 2.14$$

第11章差错控制编码

- 另外一种许用波形序列的路径是，U_1WU_3（见上图）。它和V序列相似，从状态***a***开始，离开U（虚线路径），再回到状态***a***。这个路径和U的距离等于

$$\begin{aligned} d^2 &= d^2(U_1,U_1)+d^2(U_2,W)+d^2(U_3,U_3) \\ &= d^2(000,000)+d^2(000,001)+d^2(000,010) \\ &= 0+(2)^2+0=4 \end{aligned}$$

即 $d = 2$

比较上面两条路径可见，路径U_1WU_3和路径V相比，前者和路径U的距离更小。并且，可以逐个验证，这是和路径U距离最小的许用序列的路径。因此，按照上述定义，上式中的距离就是这种编码的自由欧氏距离。故可以将其写为

$$d_{\text{Fed}} = 2$$

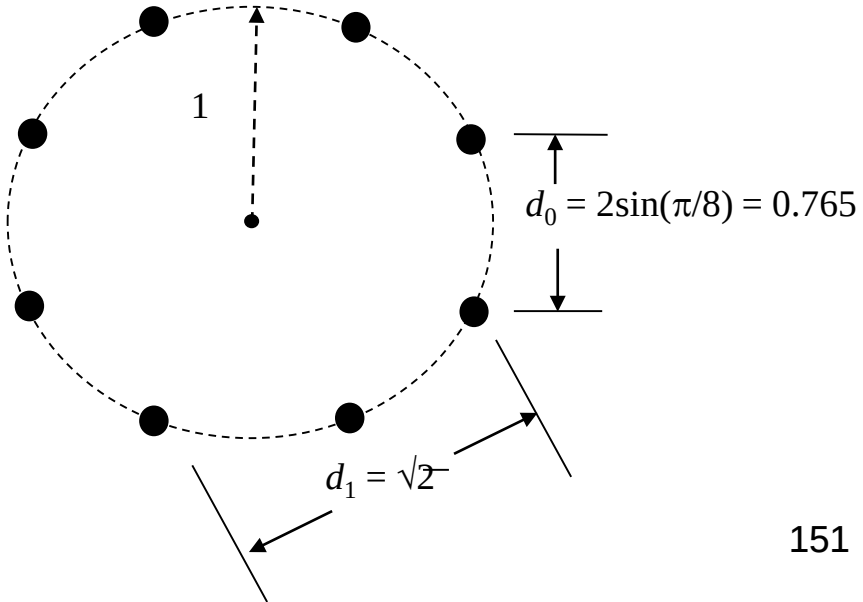
第11章差错控制编码

- 另一方面，未编码的QPSK信号的相继码元（波形）没有约束。若将其自由欧氏距离作为参考距离d_{ref}，则由下图可知，

$$d_{ref} = d_1 = \sqrt{2}$$

- 所以，可以证明，和未编码QPSK系统相比，8PSK的TCM系统可以获得的渐近编码增益等于

$$G_{8PSK/QPSK} = 20\lg(d_{Fed}/d_{ref}) = 3.01 \quad \text{(dB)}$$



第11章差错控制编码

- 在下表中列出了通过大量仿真计算得出的部分8PSK/TCM系统的(渐近)编码增益。

状态数目	k	$G_{8PSK/QPSK}$
4	1	3.01
8	2	3.60
16	2	4.13
32	2	4.59
64	2	5.01
128	2	5.17
256	2	5.75

第11章差错控制编码

11.11 小结

信道编码的目的是提高信号传输的可靠性。信道编码的基本原理是在信号码元序列中增加监督码元，并利用监督码元去发现或纠正传输中发生的错误。在信道编码只有发现错码的能力而无纠正错码的能力时，必须结合其他措施纠正错码，否则只能将发现为错码的码元删除。这些手段统称为差错控制。

按照加性干扰造成错码的统计特性不同，可以将信道分为三类：随机信道、突发信道和混合信道。每种信道的错码特性不同，所以需要采用不同的差错控制技术来减少或消除其中的错码。差错控制技术共有四种，即检错重发、前向纠错、检错删除和反馈校验，其中前3种都需要采用编码。

编码序列中信息码元数量k和总码元数量n之比 k/n称为码率。而监督码元数(n-k)和信息码元数k之比 (n-k)/k称为冗余度。

检错重发法通常称为ARQ。ARQ和前向纠错方法相比的主要优点是：监督码元较少，检错的复杂程度较低，能适应不同特性的信道。但是ＡＲＱ系统需要双向信道，并且传输效率较低，不适用于实时性要求高的场合，也不适用于一点到多点的通信系统。

第11章差错控制编码

一种编码的纠错和检错能力决定于最小码距。在保持误码率恒定条件下，采用纠错编码所节省的信噪比称为编码增益。

纠错编码分为分组码和纠错码两大类。由代数关系式确定监督位的分组码称为代数码。在代数码中，若监督位和信息位的关系是由线形代数方程式决定的，则称这种编码为线性分组码。奇偶监督码就是一种最常用的线性分组码。汉明码是一种能纠正１位错码的效率较高的线性分组码。具有循环性的线性分组码称为循环码。ＢＣＨ码是能够纠正多个随机错误的循环码。而ＲＳ码则是一种具有很强纠错能力的多进制ＢＣＨ码。

在线性分组码中，发现错码和纠正错码是利用监督关系式计算校正子来实现的。由监督关系式可以构成监督矩阵。右部形成一个单位矩阵的监督矩阵称为典型监督矩阵。由生成矩阵可以产生整个码组。左部形成单位矩阵的生成矩阵称为典型生成矩阵。由典型生成矩阵得出的码组称为系统码。在系统码中，监督位附加在信息位的后面。线性码具有封闭性。封闭性是指一种线性码中任意两个码组之和仍为这种编码中的一个码组。

循环码的生成多项式g(x)应该是 (x^n+1) 的一个(n-k)次因子。在设计循环码时可以采用将码长截短的方法，满足设计对码长的要求。

ＢＣＨ码分为两类：本原ＢＣＨ码和非本原ＢＣＨ码。在ＢＣＨ码中，(２３，１２)码称为戈莱码，它的纠错能力强并且容易解码，故应用较多。为了得到偶数长度ＢＣＨ码，可以将其扩展为(n+1，k)的扩展ＢＣＨ码。

第11章差错控制编码

ＲＳ码是多进制ＢＣＨ码的一个特殊子类。它的主要优点是：特别适合用于多进制调制的场合，和适合在衰落信道上纠正突发性错码。

卷积码是一类非分组码。卷积码的监督码元不仅和当前的ｋ个比特信息段有关，而且还同前面m＝（Ｎ－１）个信息段有关。所以它监督着Ｎ个信息段。通常将Ｎ称为卷积码的约束度。

卷积码有多种解码方法，以维特比解码算法应用最广泛。

Ｔｕｒｂｏ码是一种特殊的链接码。由于其性能近于理论上能够达到的最好性能，所以它的发明在编码理论上是带有革命性的进步。

ＴＣＭ是一种将调制和纠错编码结合在一起的体制。它能同时节省发送功率和带宽。