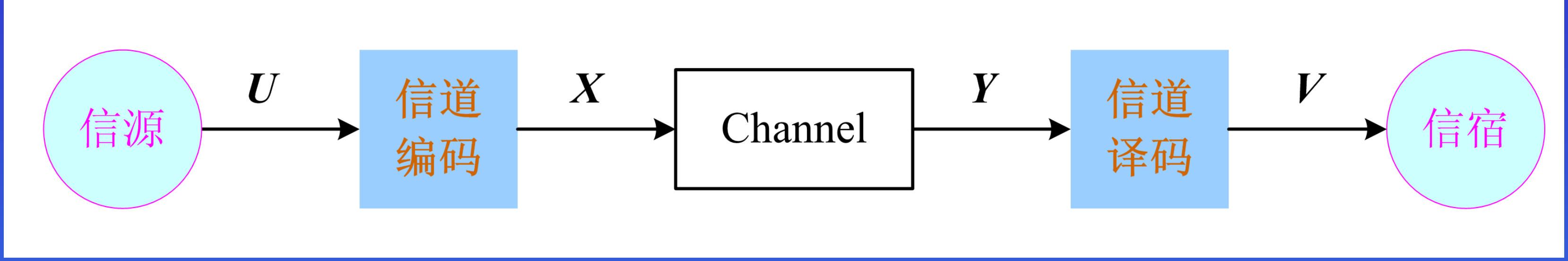
《通信原理》

第9章

杨鸿文

yanghong@bupt.edu.cn

信道编码是干什么的？



- 数据通过信道传输时，差错在所难免。信道编码旨在提供一种对差错的保护技术
  - 信道编码的概念相当广泛，除另有说明之外，本章主要考虑二进制编码及BSC信道：U、X、Y、V都是二进制序列

本章内容

- 相关基本概念
- 线性分组码
- 循环码
- 卷积码
- 交织
- Turbo码、LDPC码
- 编码调制

信道差错问题

- 一组*k*个比特通过信道裸传，难免会出现差错。
  - 根据第5章或第6章的误码率公式，除非信噪比无限大，否则误码的概率总是>0的
- 接收端不可能知道该组中哪个比特错了，甚至也不可能知道该组中是否存在错误。
  - 例如：接收到1110，那么在接收机看来，可能是
    - 发送的本来就是1110
    - 发送的本来是0000，但因为前3个比特出错，所以我看到了1110
    - 一共有16种可能性，接收机自己不可能排除任何一种可能性

信道编码

- 信道编码将*k*个信息比特映射为*n*个编码比特后通过信道传输，指望改传这*n*个比特后
  - 对于出现机会高的错误，收端可以借助编码的内部结构来自行纠正
    - 借助编码的结构，定位出具体哪个比特是错的，然后反转该比特
  - 或者至少能让收端知道是分组中否有错
    - 收端通过编码的结构获知接收分组中一定有错，但不清楚错在哪里

FEC and ARQ

- FEC （Forward Error Correction）：
  - 系统设计的思路是依靠纠错编码，让接收端单独搞定差错
- ARQ (Automatic Repeat reQuest)
  - 系统设计的思路是依靠检错编码，接收端判断收到的分组是有错。如果有，则要求发端重传
- HARQ (Hybrid ARQ)
  - HARQ是FEC和ARQ的结合：收端搞定能搞定的差错，如果错误太多，它没有把握搞定，再要求发端重传

信道编码何以能做到这些？

- 奥妙在于冗余
  - k个信息比特有$M=2^k$种不同。裸传时，对收端来说，任何一种k比特组合都是不能排除的
  - 每一组k比特信息经编码后成为一个n长的码字，全体码字的集合C中装有$M=2^k$个不同的码字。
  - n个编码比特自身有2^n种组合，但编码器发送的码字只能来自C，即存在冗余
  - 相当于n维M进制调制
    - 根据MFSK的经验，扩大维数有望获得好处
  - 每个发送符号携带k/n比特信息，称k/n为编码率

收端

- 收端收到n个可能包含错误的比特
- 收端知道发端编码所用的码字集合C，但不知道发的是C中的哪一个
- 如何检错：
  - 如果收到的n个比特不在C中，收端可以确信：传输中出了错误
- 如何纠错：
  - 按就近原则把不正确的接收码组判决为C中最近的那个码字
    - 完全类似第6章对星座图的判决

数学基础

- 全体实数连同其加减乘除四则算术规则构成了一个代数系统，称为实数域。
- 最简单的代数系统是二元域，记为GF(2)。它只有0、1这两个数，在算术规则方面把1+1的结果改成了0，此外不变。
  - 乘法：乘0得0，乘1等于没乘
  - 加法：加0等于没加，新规定1+1=0
- 本章以下除不言自明的场合外，默认为GF(2)

GF(2^n)上的向量空间

- 类似于R^n代表元素为实数的n维向量的集合，GF(2^n)代表元素为GF(2)的n维向量的集合
  - 共有2^n个向量
  - 对加法封闭
  - 任何向量都可以用一组n个线性无关的基向量组合而成
- 若C是GF(2^n)的一个k维子向量空间，则
  - C有2^k个元素
  - C的元素可以用一组k个线性无关的基向量组合而成
  - 存在一个$n-k$维的子空间$C^{\perp}$，其元素与C的元素正交

伽罗华

Evariste Galois

1811-1832

码重及码距

- 令c是$GF(2^n)$中的一个向量，c中1的个数称为c的码重或汉明重量，记为$w(c)$
- 令c_1、c_2是$GF(2^n)$中的两个向量，c_1和c_2的元素不相同的位置的个数称为它们的码距或汉明距，记为$d(c_1,c_2)$
- 显然：$w(c_1+c_2)=d(c_1,c_2)$
  - 加法规则是：相同得0，相异得1。因此和向量中的1表示两个向量在此位置上不相同，0表示该位置上相同。

汉明

Richard Hamming

1915 - 1998

最小码距

- 一种(n,k)分组码定义为$GF(2^n)$的一个子集C，它有2^k个不同的元素，称这些元素为码字。
  - 也就是：送给编码器k个比特，编码器将输出n个比特
- C中任意两个不同码字之间的最小汉明距称为该码的最小码距。
- 发送某个码字$\mathbf{c}$，如果错了x个，则接收向量$\mathbf{y}$与$\mathbf{c}$的距离将是$w(\mathbf{c}+\mathbf{y})=x$
- 设C中码字之间的最小距离是奇数$d=2t+1$，发送某个码字$\mathbf{c}$，错了x个后成为$\mathbf{y}$，那么当$x\le t$时，C中离$\mathbf{y}$最近的一定还是$\mathbf{c}$。

纠错能力

- 推论：
  - 如果一种编码的最小码距是$2t+1$，则该码一定可以纠正t位错
    - 可以纠正t位错的意思是：发送任意一个码字c，如果传输中出现的错误数在$1\sim t$的范围内，则译码器一定可以译为c
    - “可以纠正t位错”不表示超过t位错时，译码器不可能正确译码，而是说：t内可以保证，t外不能保证所有错误类型都可译对
      - 注意没有说：超过t个错时，一定译错

检错能力

- 推论2：
  - 如果一种编码的最小码距是$e+1$，则该码一定可以发现e位错
    - “发现e位错”的意思是：如果接收向量中存在$1\sim e$个错，接收端可以知道“存在错误”这个事实，但不知道是几个错，也不知道具体那个比特错

线性分组码

- 如果C构成GF(2^n)一个线性向量子空间，则称其为线性分组码
  - 换言之就是对加法封闭：即若c_1、c_2属于C，则c_1+c_2也一定属于C
- 若干性质
  - 全零码字属于C
  - 最小码距等于非全0码字之外的最小码重
  - C是一个k维子空间
  - GF(2^n)存在一个$r=n-k$维的子空间，其元素与C中的任意元素正交

线性分组码例：(5,1)重复码

- 将k=1个比特重复n=5遍。
- C中有两个码字：11111和00000
- 码距是5，可纠正2位错
- 编码率是1/5

线性分组码例：(4,3)偶校验码

- 给k=3个信息比特后缀$n-k$=1个校验比特，使码字中有偶数个1
- 全部码字有8个：
  - 0000 0011 0101 0110
  - 1001 1010 1100 1111
- 编码率是3/4
- 最小码距是2，可以检出所有奇数位错

线性分组码例：(7,4)汉明码

- (7,4)汉明码给4个信息比特$u_3u_2u_1u_0$后缀3个校验比特$p_2p_1p_0$

$$(c_6,c_5,\cdots,c_0)=(u_3,u_2,u_1,u_0,p_2,p_1,p_0)$$

- 校验规则：

$$\begin{cases} p_2=c_2=u_2+u_1+u_0 \\ p_1=c_1=u_3+u_1+u_0 \\ p_0=c_0=u_3+u_2+u_1 \end{cases}$$

- 全部$2^k=16$个码字：
  - 0000000 0001110 0010111 0011001
  - 0100101 0101011 0110010 0111100
  - 1000011 1001101 1010100 1011010
  - 1100110 1101000 1110001 1111111

- 码率是4/7
- 最小码距是3，可以纠正1位错

基

- C是一个线性向量子空间，故存在一组线性无关的基向量，使得C中所有向量均可表达为这组基的线性组合

$$\boldsymbol{c} = u_{k-1}\boldsymbol{g}_1 + u_{k-2}\boldsymbol{g}_2 + \cdots + u_0\boldsymbol{g}_k$$

生成矩阵

- 写成矩阵形式

$$(c_{n-1}, c_{n-2}, \cdots, c_0) = (u_{k-1}, u_{k-2}, \cdots, u_0)\begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{pmatrix}$$

$$= (u_{k-1}, u_{k-2}, \cdots, u_0)\begin{pmatrix} g_{1,n-1} & g_{1,n-2} & \cdots & g_{1,0} \\ g_{2,n-1} & g_{2,n-2} & \cdots & g_{2,0} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,n-1} & g_{k,n-2} & \cdots & g_{k,0} \end{pmatrix}$$

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}$$

生成矩阵

- 线性分组码是把k维的信息向量$\boldsymbol{u}$通过线性变换G扩张成n维的$\boldsymbol{c}$。通过这种扩张使码距扩大

- G的一些性质
  - G有k行n列
  - 每个码字是G的各行的线性组合
  - G的每一行是一个码字
  - G的各行线性无关
- 注意：给定C时，G不唯一

系统码

- 如果信息比特$\boldsymbol{u}$原样出现在$\boldsymbol{c}$中，这样的编码称为系统码。

$$\left(c_{n-1}, c_{n-2}, \cdots, c_0\right) = \left(u_{k-1}, u_{k-2}, \cdots, u_0, p_{r-1}, p_{r-2}, \cdots, p_0\right)$$

$r=n-k$是校验位的个数

$$\boldsymbol{c} = \left(\boldsymbol{u}, \boldsymbol{p}\right)$$

$$\boldsymbol{G} = \left(\boldsymbol{I}, \boldsymbol{Q}\right)$$

系统码的生成矩阵示例

- (5,1)重复码 $\boldsymbol{G}=\begin{pmatrix}1 & 1 & 1 & 1 & 1\end{pmatrix}$
- (4,3)偶校验码

$$\boldsymbol{G}=\begin{pmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\end{pmatrix}$$

注意：

(1) 矩阵G的各行是从C中任意选出的k个线性无关码字

(2) 系统码的G的各行是C中的这样一些码字，其前k位是10…0、010…0、…、0…01

系统码的生成矩阵示例

- (7,4)汉明码

$$\boldsymbol{G}=\begin{pmatrix}\boldsymbol{g}_1\\\boldsymbol{g}_2\\\boldsymbol{g}_3\\\boldsymbol{g}_4\end{pmatrix}=\begin{pmatrix}1&0&0&0&0&1&1\\0&1&0&0&1&0&1\\0&0&1&0&1&1&1\\0&0&0&1&1&1&0\end{pmatrix}$$

注意：

(1) 若不规定为系统码，G不唯一

(2) 对于正常的编码，任何G都可以化为系统码形式，方法是对G的各行进行线性组合（初等行变换），以获得形如10…0xxx、010…0xxx、…、0…01xxx这样的码字

化任意生成矩阵为系统码的生成矩阵示例

$$\boldsymbol{G}=\begin{pmatrix}\boldsymbol{g}_1\\ \boldsymbol{g}_2\\ \boldsymbol{g}_3\\ \boldsymbol{g}_4\end{pmatrix}=\begin{pmatrix}0&1&0&1&0&1&1\\0&1&1&1&1&0&0\\1&1&1&0&0&0&1\\1&0&1&0&1&0&1\end{pmatrix}$$

$$\boldsymbol{g}_1 + \boldsymbol{g}_2 = (0010111) = \tilde{\boldsymbol{g}}_3$$

$$\boldsymbol{g}_3 + \boldsymbol{g}_4 = (0100101) = \tilde{\boldsymbol{g}}_2$$

$$\boldsymbol{g}_1 + \tilde{\boldsymbol{g}}_2 = (0001110) = \tilde{\boldsymbol{g}}_4$$

$$\tilde{\boldsymbol{g}}_3 + \boldsymbol{g}_4 = (1000011) = \tilde{\boldsymbol{g}}_1$$

$$\tilde{\boldsymbol{G}} = \begin{pmatrix} \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \\ \tilde{\boldsymbol{g}}_3 \\ \tilde{\boldsymbol{g}}_4 \end{pmatrix}$$

生成矩阵的作用：编码

$$\boldsymbol{c}=\boldsymbol{uG}=u_{k-1}\boldsymbol{g}_1+u_{k-2}\boldsymbol{g}_2+\cdots+u_0\boldsymbol{g}_k$$

- 根据$\boldsymbol{u}$的内容选取G的行向量，其和就是编码结果
- 系统码的好处：
  - 便于输出信息：
    - 系统码$\boldsymbol{c}$的前半截已经是$\boldsymbol{u}$，非系统码还需要将$\boldsymbol{c}$映射为$\boldsymbol{u}$
  - 编码中的计算也能少一些：
    - 非系统码需要计算$\boldsymbol{uG}$，系统码只需计算$\boldsymbol{uQ}$，矩阵$\boldsymbol{Q}$比矩阵$\boldsymbol{G}$小

监督矩阵

- 因为C是GF(2^n)的k维子空间，故存在$r=n-k$维的零空间$C^{\perp}$，其任意元素与C的任意元素正交
- 任意取$C^{\perp}$的r个线性无关基向量$\boldsymbol{h}_1$、$\boldsymbol{h}_2$、…、$\boldsymbol{h}_r$，构成矩阵

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \vdots \\ \boldsymbol{h}_{n-k} \end{pmatrix}$$

称其为C的监督矩阵

监督矩阵的性质

- $\boldsymbol{H}$有n–k行，n列
- 对于C中的任意码字$\boldsymbol{c}$，恒有 $\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}}=\boldsymbol{0}$
- $\boldsymbol{H}$的各行线性无关
- 若C的最小码距是d，则H的任意d–1列线性无关
  - 若d=3，则$\boldsymbol{H}$的任意两列不同
  - 若d=4，则$\boldsymbol{H}$的任意两列不同，且任一列不是其他两列之和
  - …
  - $\boldsymbol{H}$的任意一列不等于其他某1列、2列、…、d–1列之和

- 令列向量$\boldsymbol{h}_i$表示$\boldsymbol{H}$的第i列，则

$$\begin{aligned}\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}} &= \begin{pmatrix}\boldsymbol{h}_1 & \boldsymbol{h}_2 & \cdots & \boldsymbol{h}_n\end{pmatrix}\begin{pmatrix}c_{n-1}\\ c_{n-2}\\ \vdots\\ c_0\end{pmatrix}\\ &= c_{n-1}\boldsymbol{h}_1 + c_{n-2}\boldsymbol{h}_2 + \cdots + c_0\boldsymbol{h}_n\\ &= 0\end{aligned}$$

- 即$\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}}$是$\boldsymbol{H}$的某些列之和（$\boldsymbol{c}$的元素是0或1），其和必为0

- 已知最小码距是d，因此
  - 存在c，它有d个1，即$\boldsymbol{H}$有d列之和为0，即$\boldsymbol{H}$有d列线性相关
  - 不存在非全零的c，它有$d-1$个或更少的1，即$\boldsymbol{H}$的任意 $d-1$列不可能被一组不全为之数（GF(2)中数只能是0、1）加权和为0，即$\boldsymbol{H}$的任意$d-1$列线性不相关

监督矩阵与生成矩阵的关系

- $\boldsymbol{G}$的每一行都属于C，故有$\boldsymbol{HG}^{\mathrm{T}}=\mathbf{0}$
- $\boldsymbol{H}$的秩是$r=n-k$，故可化为$\boldsymbol{H}=(\boldsymbol{P},\boldsymbol{I}_r)$的形式
  - 即$C^{\perp}$中一定存在这样的向量，其后r位是10…0，010…0，…，0…01
- 若$\boldsymbol{G}$是系统码的生成矩阵，则$\boldsymbol{G}=(\boldsymbol{I}_k,\boldsymbol{Q})$

$$(\boldsymbol{P},\boldsymbol{I}_r)\begin{pmatrix}\boldsymbol{I}_k\\ \boldsymbol{Q}^{\mathrm{T}}\end{pmatrix}=0$$

$$\boldsymbol{P}=\boldsymbol{Q}^{\mathrm{T}}$$

监督矩阵示例

- (4,3)偶校验码

$$\boldsymbol{H}=\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}$$

- (5,1)重复码

$$\boldsymbol{H}=\begin{pmatrix}1 & 1 & 0 & 0 & 0\\1 & 0 & 1 & 0 & 0\\1 & 0 & 0 & 1 & 0\\1 & 0 & 0 & 0 & 1\end{pmatrix}$$

- (7,4)汉明码

$$\boldsymbol{H}=\begin{pmatrix}0 & 1 & 1 & 1 & 1 & 0 & 0\\1 & 0 & 1 & 1 & 0 & 1 & 0\\1 & 1 & 1 & 0 & 0 & 0 & 1\end{pmatrix}$$

错误图样

- 发送C中的码字$\boldsymbol{c}$，收到的向量$\boldsymbol{y}$可以表示为

$$\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$$

$$(y_{n-1}, y_{n-2}, \cdots, y_0) = (c_{n-1}, c_{n-2}, \cdots, c_0) + (e_{n-1}, e_{n-2}, \cdots, e_0)$$

- 称$\boldsymbol{e}=(e_{n-1},e_{n-2},\ldots,e_0)$为错误图样，其元素为1表示该位置出错，为0表示该位置发送的比特正确

译码

- 译码器收到的是*y*，需要根据*y*来判断
  - *y*是否有错，即：*e*是否是全0向量？
  - 如果有错，*e*=？
    - 若译码器知道*e*，就可以知道$c=y+e$
- 困难：
  - *c*是*C*中的任意元素，*e*是GF(2^n)中的任意元素，有大量的(c,e)组合都能给出相同的*y*
- 我们并不求解*c*必然是什么，只求解*c*最有可能是什么？
  - 寻求最可能的错误图样，即寻求可纠正错误图样

伴随式

- 译码器根据收到y算出这样一个向量

$$\boldsymbol{s}=\boldsymbol{H}\boldsymbol{y}^{\mathrm{T}}$$

  - **称为**y所对应的伴随式

- 由于

$$\boldsymbol{s}=\boldsymbol{H}\boldsymbol{y}^{\mathrm{T}}=\boldsymbol{H}\left(\boldsymbol{c}+\boldsymbol{e}\right)^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{e}^{\mathrm{T}}$$

  - 所以，若$s\neq 0$，说明$e\neq 0$，即发现y中有错
  - 给定s时，满足$He^{\mathrm{T}}=s$的所有e只是$GF(2^n)$中的一个子集，即：我们现在可以排除很多的错误图样，只需在这个子集中找最可能的错误图样

可能的错误图样

- 如下线性方程组的解有2^k个

$$\boldsymbol{He}^{\mathrm{T}} = \boldsymbol{s}$$

- 我们有$n-k$个方程，n个变量。因此不存在唯一解
- 任意固定k个变量，可得到其余$n-k$个变量的唯一解。“固定k个变量”有2^k种选择，因此该线性方程组的解有2^k个

最可能错误图样是误码最少的错误图样

- 设BSC信道的错误率是p，错误图样$\boldsymbol{e}$中有x个错误的概率是

$$P(x)=C_n^x p^x(1-p)^{n-x}=\frac{C_n^x}{(1-p)^n}\left(\frac{p}{1-p}\right)^x$$

- 当$p<<0.5$时，$P(x)$随x增加而迅速减小

编码为什么能降低错误率？

- 无编码时，错误的可能图样有2^k种。编码后，通过伴随式可排除一些错误图样，但剩余的可能错误图样还是2^k种
- 然而：无编码时，有一个比特错，整个信息分组就是错的。有编码时，只要错误个数不超过t，译码后就没有错
- 对于正常设计的系统，错1个本身是小概率事件，同时错t个概率将是更加的小

根据伴随式确定可纠正错误图样

- 设$s \neq 0$
  - 先看$\boldsymbol{H}$是否有某一列等于$\boldsymbol{s}$
    - 若有则可纠正错误图样就是单比特错，错误位置对应该列位置
  - 若无，则排除单比特错，继续看是否有两列的和等于$\boldsymbol{s}$
    - 若有，确定为双比特错。错误位置就是这两列的位置
  - …

例

- (7,4)汉明码的译码器收到1110000，求译码结果。

$$
\boldsymbol{H}=\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}
$$

例

- (7,4)汉明码的译码器收到1110000
- 用***H***算出$\boldsymbol{s}=(001)^T$
  - 即***H***的前3列之和
- ***s***是***H***的最后一列，故可纠正错误图样是0000001
  - 译码结果是11110001

汉明码

- 汉明码是这样一种线性分组码，其***H***的列包含全零向量之外的所有m长二进制向量
  - 码长$n=2^m-1$
  - 信息位个数：$k=n-m$
  - 校验位个数：m
  - 最小码距等于3
    - 可纠1位错

循环码

- 循环码是线性分组码的一种
- 称C为循环码，若其满足
  - 加法封闭性（线性分组码）
  - 循环移位封闭性（循环性）
    - 若$(c_{n-1},c_{n-2},\ldots,c_0)$属于$C$
    - 则$(c_{n-2},c_{n-3},\ldots,c_0, c_{n-1})$也属于$C$

循环码示例

- 下面这个*C*是循环码：
  - 0000000 0001011 0010110 0011101
  - 0100111 0101100 0110001 0111010
  - 1000101 1001110 1010011 1011000
  - 1100010 1101001 1110100 1111111
- 这个码同时也是(7,4)汉明码
- 所有重复码都是循环码
- 所有偶校验码都是循环码

多项式

- 一个码组除了可以表述为GF(2^n)上的向量外，还可以表述为一个次数不超过n–1的，系数在GF(2)上的多项式

$$\left(b_{n-1}, b_{n-2}, \cdots, b_0\right) \quad \Rightarrow \quad \boldsymbol{b} \in \mathrm{GF}\left(2^n\right)$$

$$\Rightarrow b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1 x + b_0$$

循环性的多项式表述

- 用多项式的语言来说，循环封闭性就是

$$\text{if} \quad c(x) \in C(x)$$

$$\text{then} \quad \left[xc(x) \right]_{\bmod x^n+1} \in C(x)$$

$$
\begin{aligned}
x\,c(x) &= x\left(c_{n-1}x^{n-1}+c_{n-2}x^{n-2}+\cdots+c_1x+c_0\right)\\
&= c_{n-1}x^{n}+c_{n-2}x^{n-1}+\cdots+c_1x^{2}+c_0x\\
&= c_{n-1}x^{n}+c_{n-2}x^{n-1}+\cdots+c_1x^{2}+c_0x+c_{n-1}+c_{n-1}\\
&= c_{n-1}\left(x^{n}+1\right)+\left(c_{n-2}x^{n-1}+c_{n-3}x^{n-2}+\cdots+c_0x+c_{n-1}\right)
\end{aligned}
$$

- 除以x^n+1后得到

$$c_{n-2}x^{n-1}+c_{n-3}x^{n-2}+\cdots+c_0x+c_{n-1}$$

  - 对应的码字是

$$\left(c_{n-2},c_{n-3},\cdots,c_0,c_{n-1}\right)$$

生成多项式

- 集合$C(x)$中除0多项式外，次数最低的那个多项式$g(x)$ 称为该循环码的生成多项式
  - 唯一
  - 零次项是1
  - $C(x)$中的多项式都是$g(x)$的倍式
  - 任何$g(x)$的倍式，若次数不超过$n-1$，一定在$C(x)$中
  - $g(x)$的次数等于校验位的个数
  - $g(x)$是x^n+1的一个因式

生成多项式示例

- 所有偶校验码 $g(x)=x+1$

- 所有重复码

$$g(x)=x^{n-1}+x^{n-2}+\cdots+x+1=\frac{x^n+1}{x+1}$$

- 前述的(7,4)循环码

$$g(x)=x^3+x+1$$

例：码长为7时可能的循环码

$$x^7+1=(x+1)\left(x^3+x+1\right)\left(x^3+x^2+1\right)$$

- 可以有
  - 一种(7,1)循环码—重复码
  - 一种(7,6)循环码—偶校验码
  - 两种(7,4)循环码—都是汉明码
  - 两种(7,3)循环码—m序列的循环+全0

系统循环码的编码

- 将$u(x)x^{n-k}$（即左移$n-k$位）后补一个$n-k-1$次多项式$r(x)$（即校验位），使$u(x)x^{n-k}+r(x)$能被$g(x)$整除
- 用$g(x)$及其移位可组合出所求的码字
  - 例：生成多项式为1011，对1110编码
    - 1011000和0101100的和是1110100

用生成多项式得到G和H

- 用$g(x)$将单1信息向量编为系统码
- 由此可得到系统码的***G***
- 然后可得到对应的***H***

循环码的伴随式

- 接收向量y可以表述为多项式$y(x)$
- 称$y(x)$除以$g(x)$ 的余式$s(x)$为伴随式
- 若余式不为0，则确定$y(x)$中有错
- 进一步可寻找错误数最少的错误图样

CRC

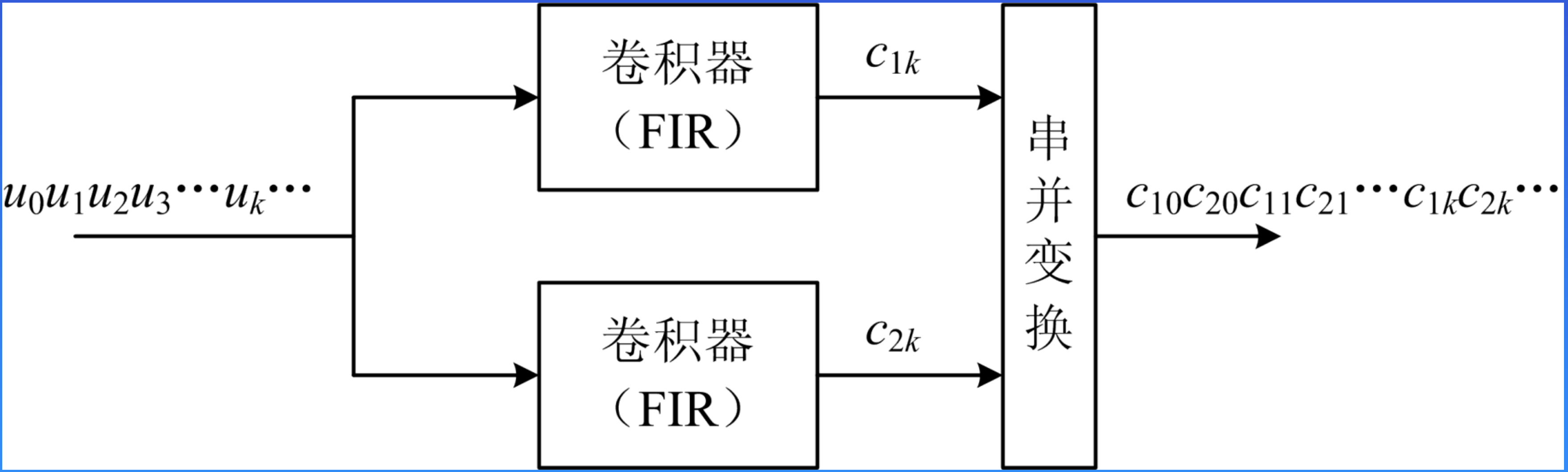
- CRC不是循环码，是线性分组码
- 与循环码的类似之处是：
  - CRC的所有码字也都是生成多项式的因式
- 与循环码的差别是：
  - 其生成多项式不一定是x^n+1的因子
- CRC是最流行的检错编码
  - 所有不能被$g(x)$整除的错误图样都可以检出

BCH码、RS码

- BCH码和RS码都属于循环码
  - 对于较短的码长，BCH是目前已知的非常好的码
  - RS码是多进制BCH码的一个特例
    - 所谓多进制的意思是：编码中的符号不是0、1，而是例如0、1、2、3这样的四进制符号。同时遵循GF(4)的算术规则
    - 对于高码率（k/n大）及擦除信道，RS经常是首选
      - 擦除信道：数据或者正确到达，或者丢失。译码器知道那个数据丢失了

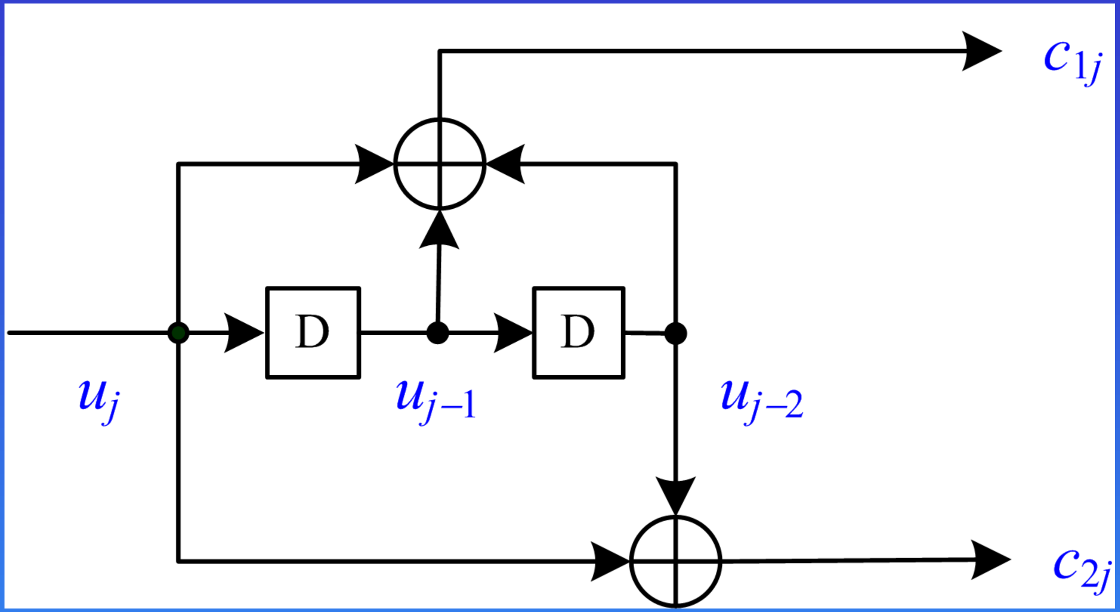
卷积码

- 将1路信息序列通过*n*路卷积器，将*n*路输出串并变换为1路后送往信道，此即卷积码。



- 在更一般的形式中，输入是*k*路，经过一个多入多出的线性系统后输出*n*路
- 卷积器也可以是IIR

卷积码编码器示例：(7,5)卷积码



- 第1路的冲激响应是111，八进制表示是7；第2路的冲激响应是101，八进制表示是5。记此卷积码为(7,5)卷积码
- 两路的生成多项式为$1+x+x^2$，$1+x^2$

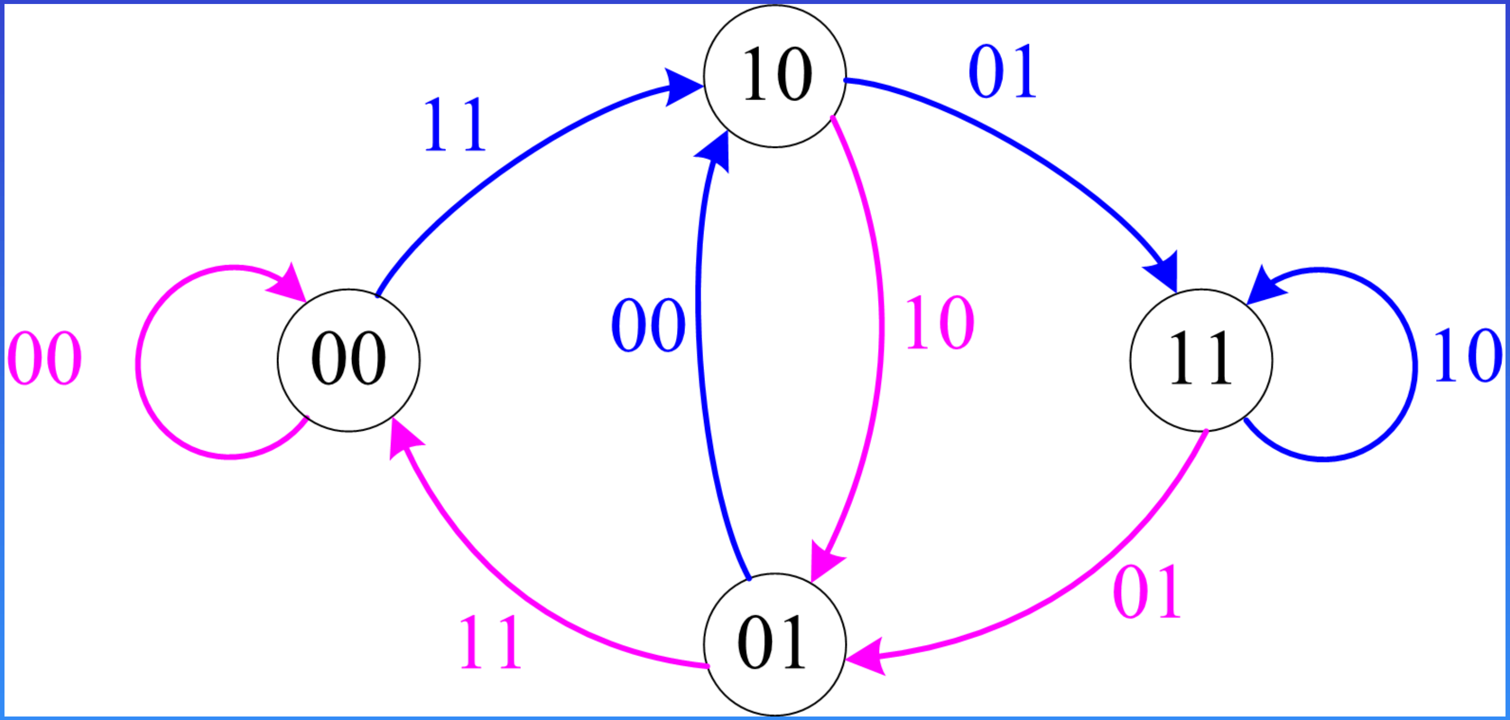
卷积码的状态

- 在上图中，定义$s_j=(u_j,u_{j-1})$为卷积码在j时刻的到达状态。相应地称$s_{j-1}=(u_{j-1},u_{j-2})$为$j$时刻的出发状态。以下未说明出发或到达时，“状态”一词默认指到达状态
- (7,5)卷积码的状态有4种可能取值
  - 一般而言，状态数=2^m，其中m是状态向量的长度，也即存储器的个数。

状态的重要性

- 给定出发状态s_{j-1}和当前的输入u_j，可以确定出到达状态s_j及当前的输出$c_{1j}\ c_{2j}$
- 给定状态的变化序列$s_0s_1s_2\ldots$，将能确定出输入序列$u_0u_1u_2\ldots$以及输出序列$c_{10}c_{20}c_{11}c_{21}\ldots$
  - 默认初始状态s_{-1}为0
- 也就是说：卷积码的全部信息都包含在状态的变化序列中

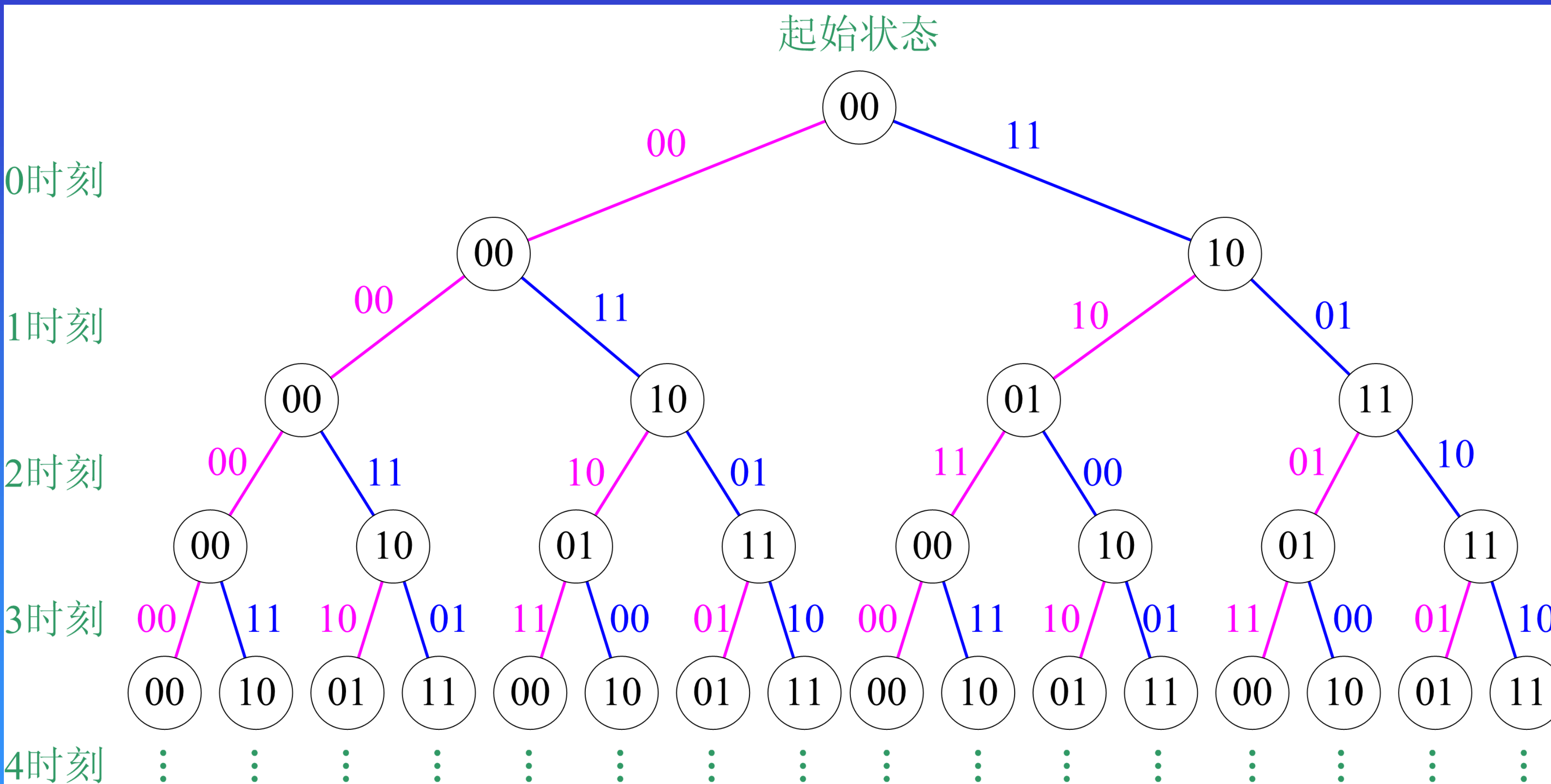
状态转移图



状态转移图说明

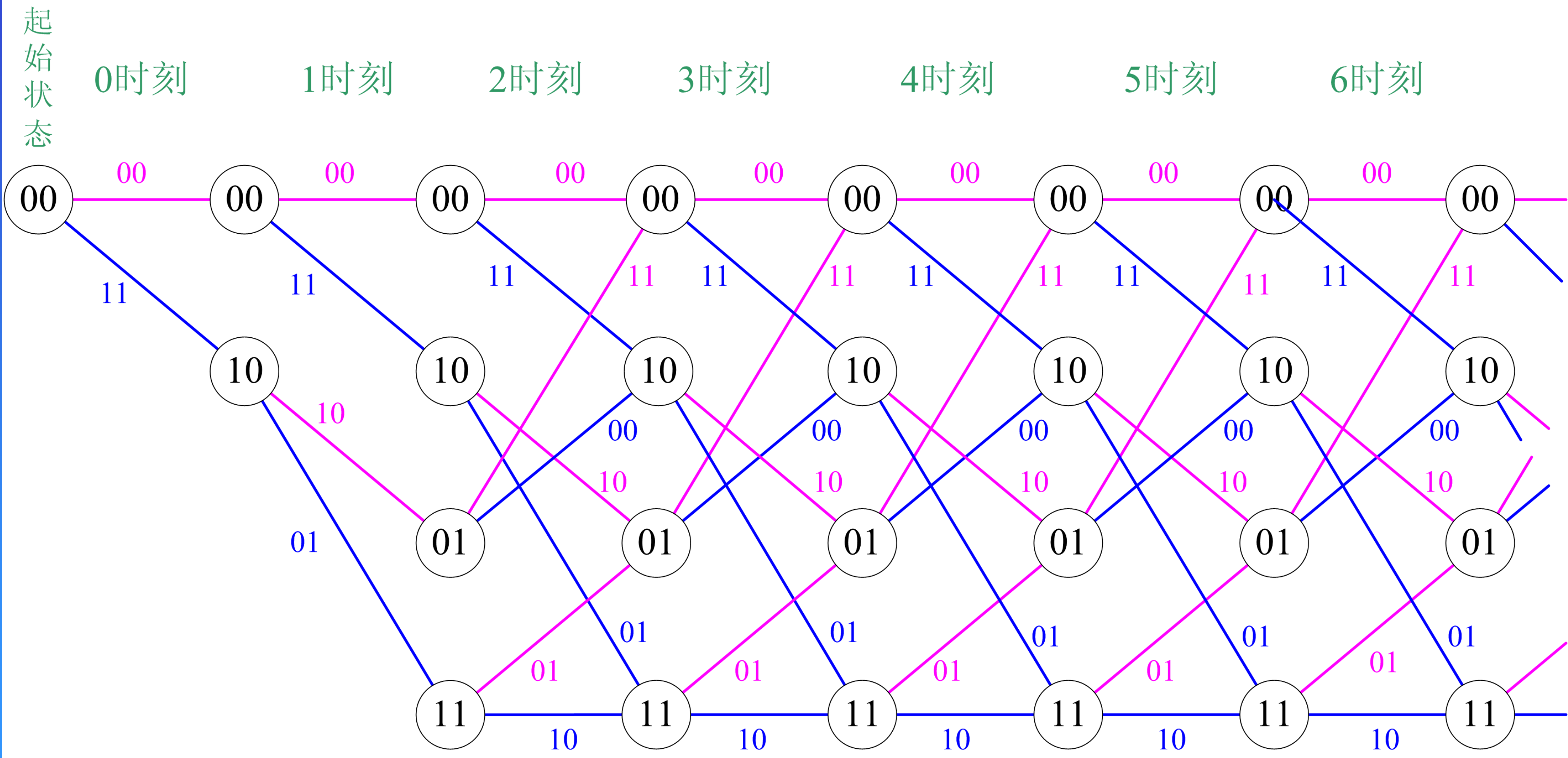
- 红线表示输入信息为0，蓝线表示输入信息为1。线旁的数字表示对应的编码器输出
- 从每个状态出发，可到达两个不同状态
- 每个到达状态都是来自两个不同的出发状态
- 输入的信息比特一定等于到达状态的第1位
- 就本例而言，每个状态出发的两条路上的编码结果正好反相
  - 需要的条件是：生成多项式的0次项都是1

卷积码的树图



起始状态
0时刻
1时刻
2时刻
3时刻
4时刻
00
00
11
00
10
00
11
10
01
00
10
01
11
00
11
10
01
11
00
01
10
00
10
01
11
00
10
01
11
00 11 10 01 11 00 01 10 00 11 10 01 11 00 01 10
00 10 01 11 00 10 01 11 00 10 01 11 00 10 01 11

相同状态往后看完全是一样的，故可合并同一时间相同的状态，画成更为紧凑的格图



卷积码的译码

- 任何一个编码输出序列，都对应着树图（格图）上的唯一一条路径
- 译码器要根据接收序列，找出这条路径
- 按照ML译码原则，译码器应该在树图的所有路径中，找出这样一条，其编码输出序列与译码器接收的序列之间的码距最小

问题

- 树图的第L时刻（从0数起）有2^{L+1}片树叶，意味着到L时刻时，可能的路径总数是2^{L+1}
- 对于较大的L，穷举所有路径，逐一同接收序列进行比对，这种方法不具有可操作性
  - 如同象棋放米粒的问题：第1格放2个，第2格放4个，第3格放8个、…。这就叫指数增长
- 需要利用树图中的特殊结构来缩减运算量

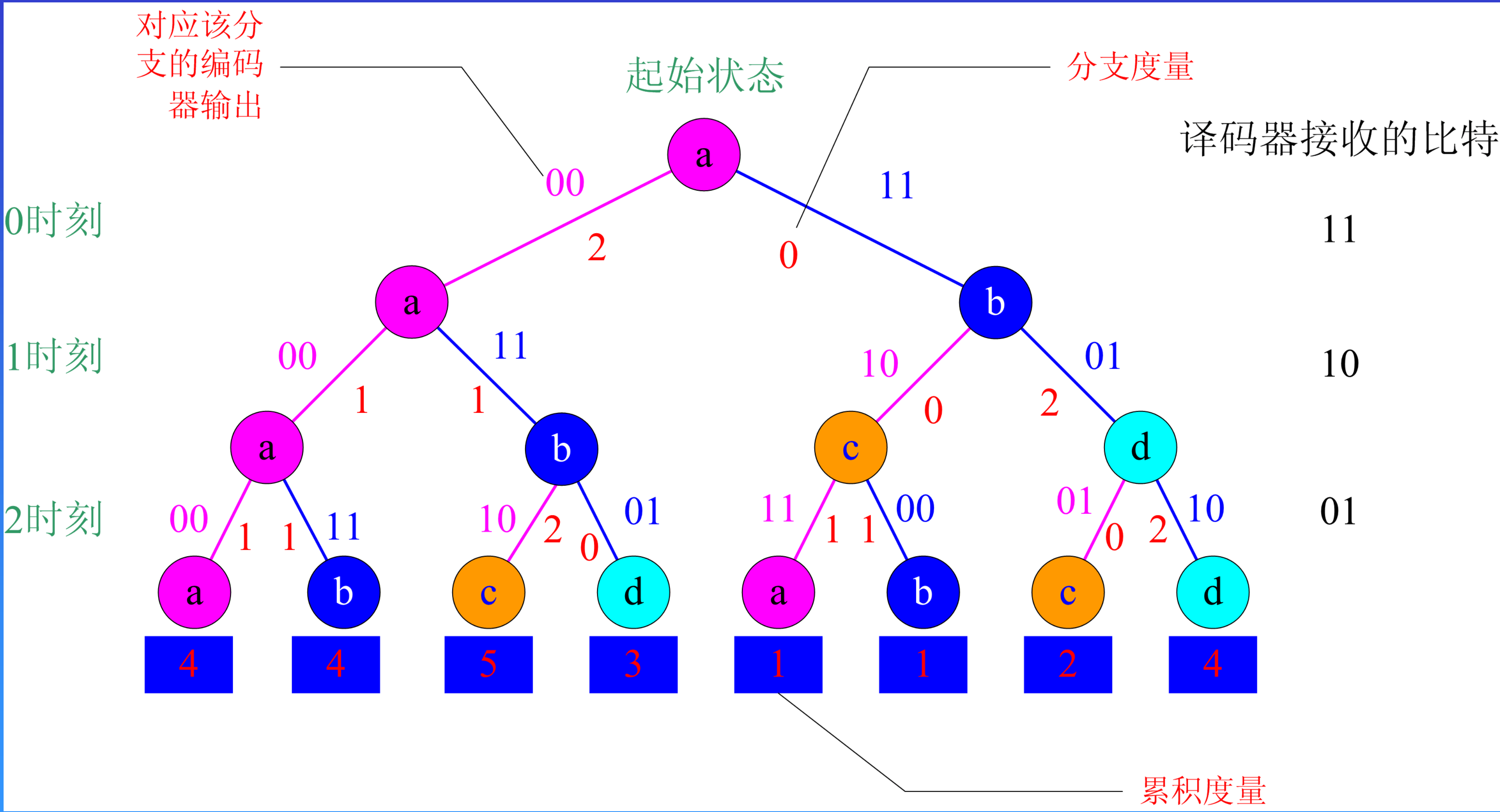
分支度量

- 以(7,5)卷积码为例，设j时刻接收的比特是$y_{1j}y_{2j}$
  - 树图在$j \geq 2$时刻有8种不同的分支（相同分支指：出发状态和到达状态相同），每个分支对应两比特编码输出$c_{1j}c_{2j}$
  - 这两比特编码输出与接收比特之间的汉明距称为该分支的分支度量

累积度量

- 从起始状态到j时刻的某个状态的路径是由各个树枝连成的，这些树枝的分支度量之和称为该路径的累积度量
- 在上述定义下，某个路径的累积度量实际是该路径与接收序列的汉明距
- ML译码就是要寻找到j时刻累积度量最小的路径

例



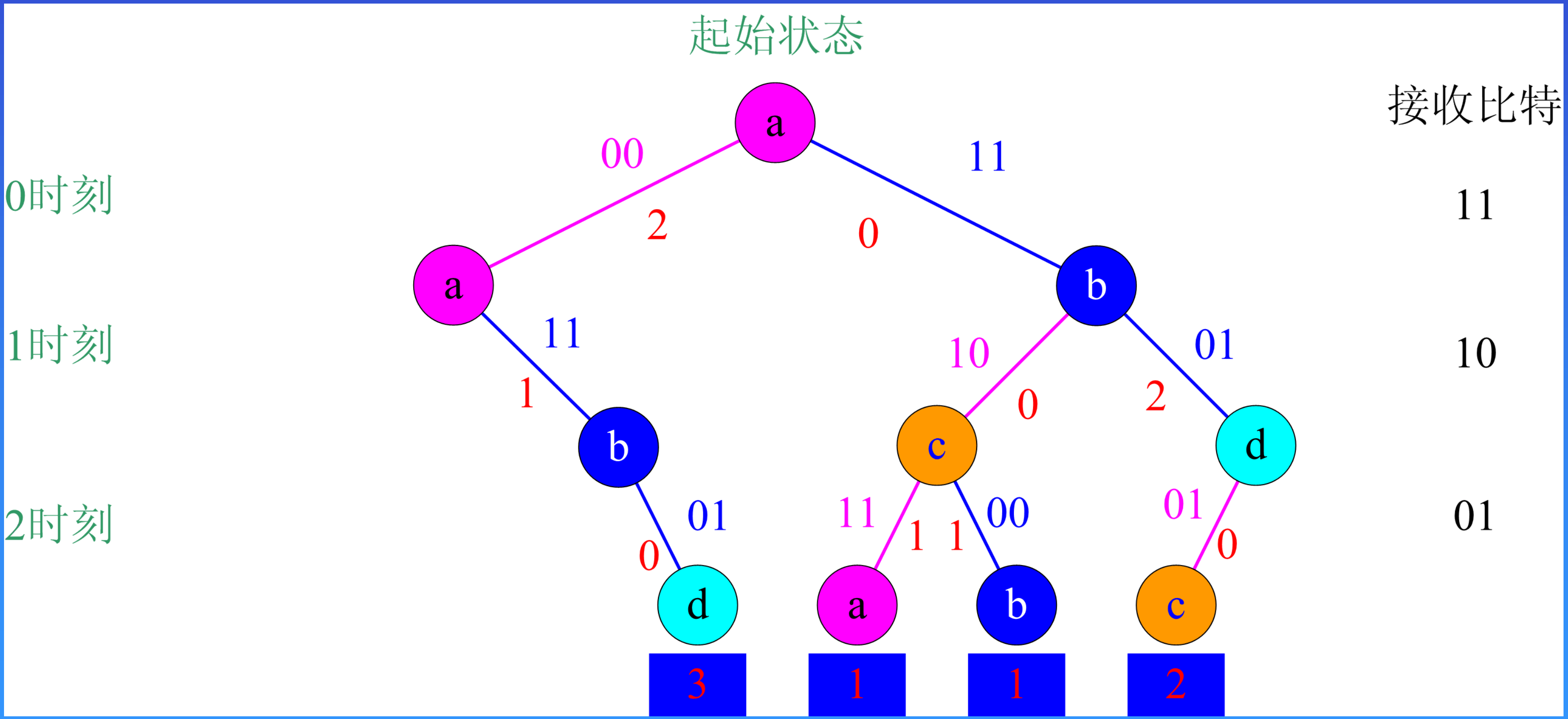
对应该分支的编码器输出
起始状态
分支度量
译码器接收的比特
0时刻
1时刻
2时刻
a
b
c
d
00
11
2
0
11
10
01
1
4
4
5
3
1
1
2
4
累积度量

- 从上图可以看出的是：
  - 截止到时刻2，暂时并列最优的路径是*abca*或*abcb*
  - 不能确定的是：
    - 若树图继续延伸，到j时刻时，最优的路前3步是否一定会走*abca*或*abcb*
  - 可以确定是：
    - 到j时刻时最优的路前3步一定不会走*aaaa*
    - 这是因为：*aaaa*向后延伸出的所有可能路径和*abca*可延伸出的路径完全相同，而已知前3步*aaaa*比*abca*差

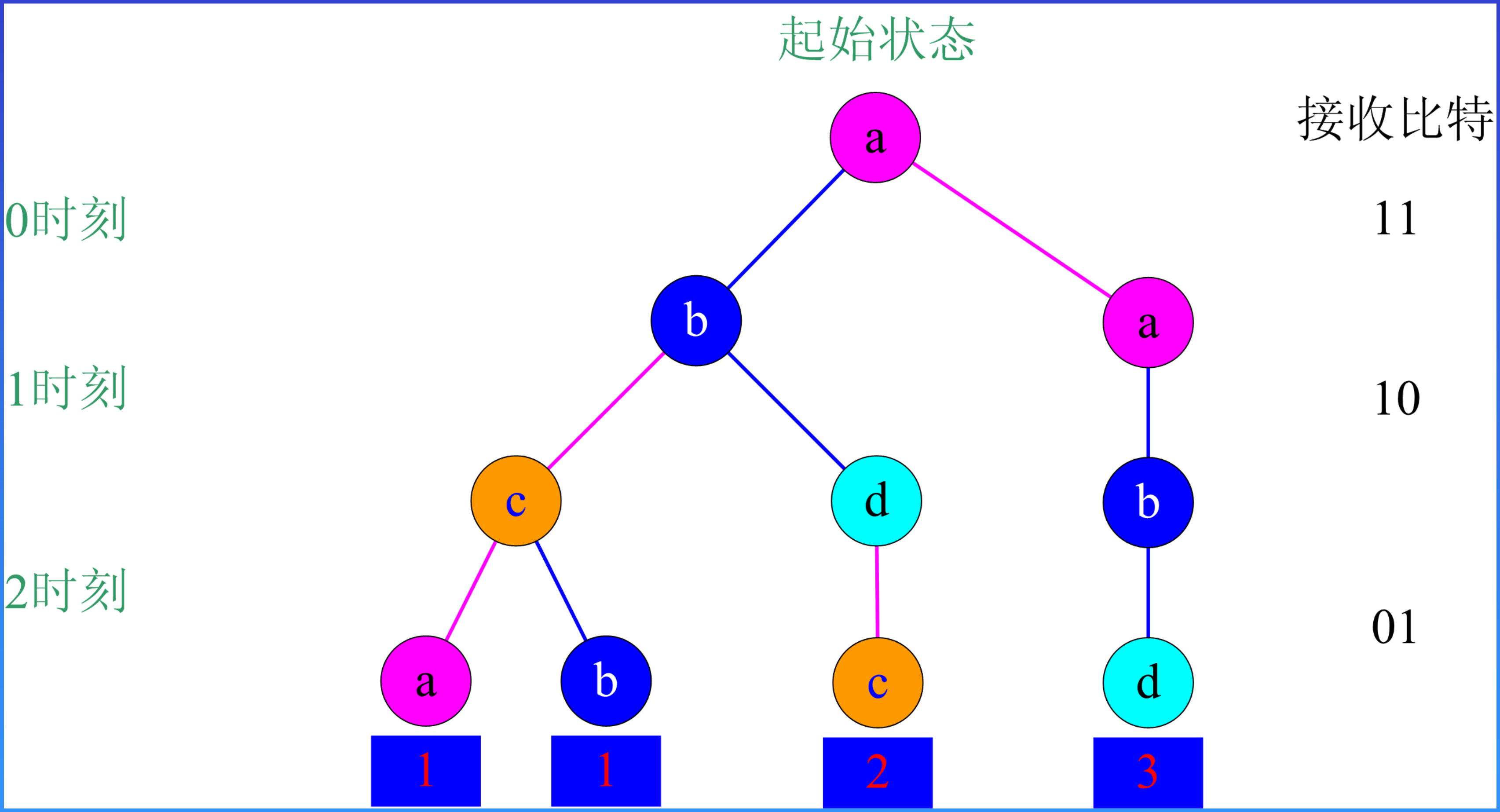
Viterbi译码

- 基于此，再继续向下探寻最优路径时，可以排除路径*aaaa*
- 基于同样理由，还可以
  - 排除*aaab*（输给*abcb*）
  - 排除*aabc*（输给*aadc*）
  - 排除*abdd*（输给*aabd*）
- 称保留的胜出者为幸存路径

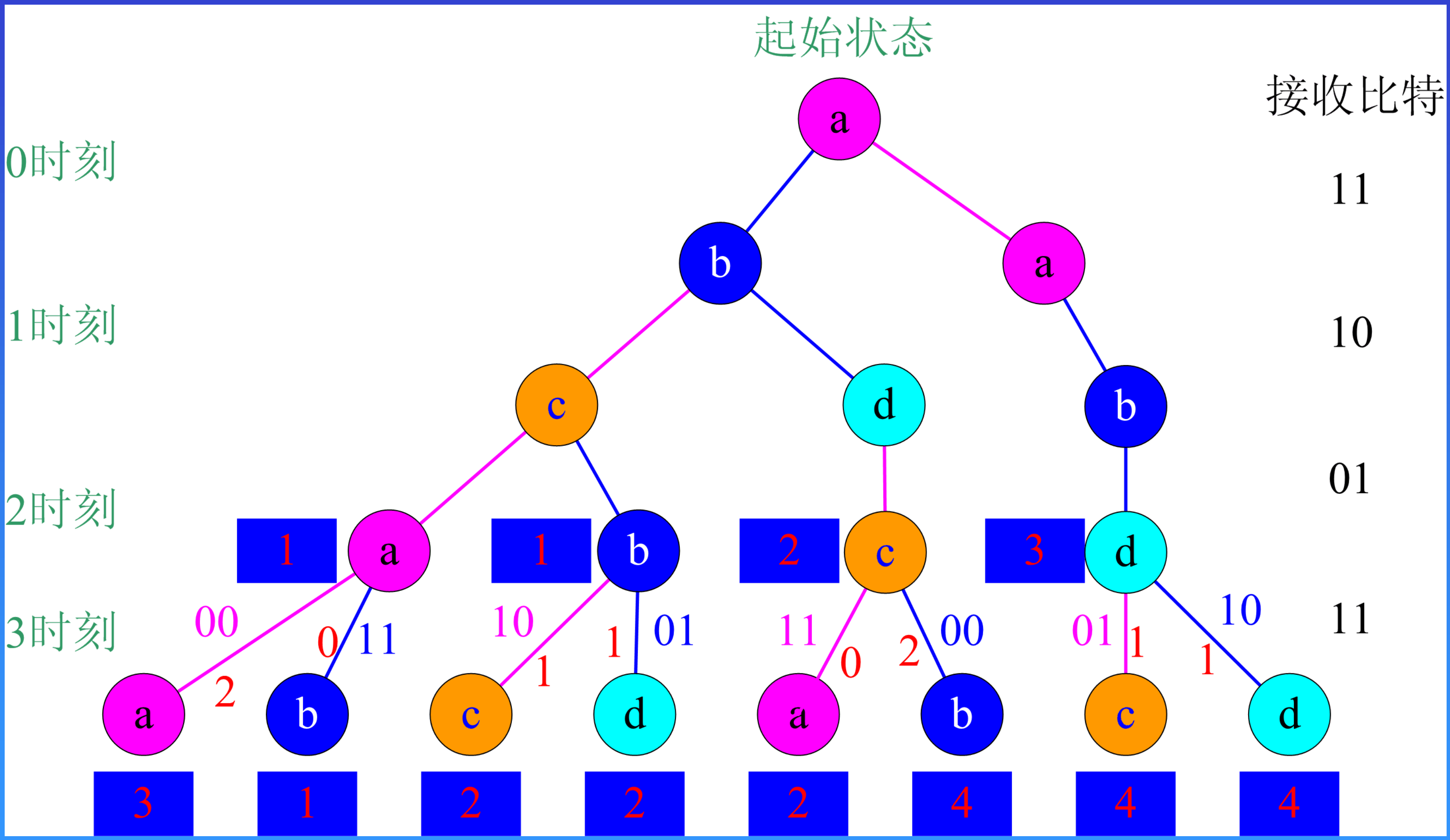
在树图上剪去可以排除的树枝后，成为



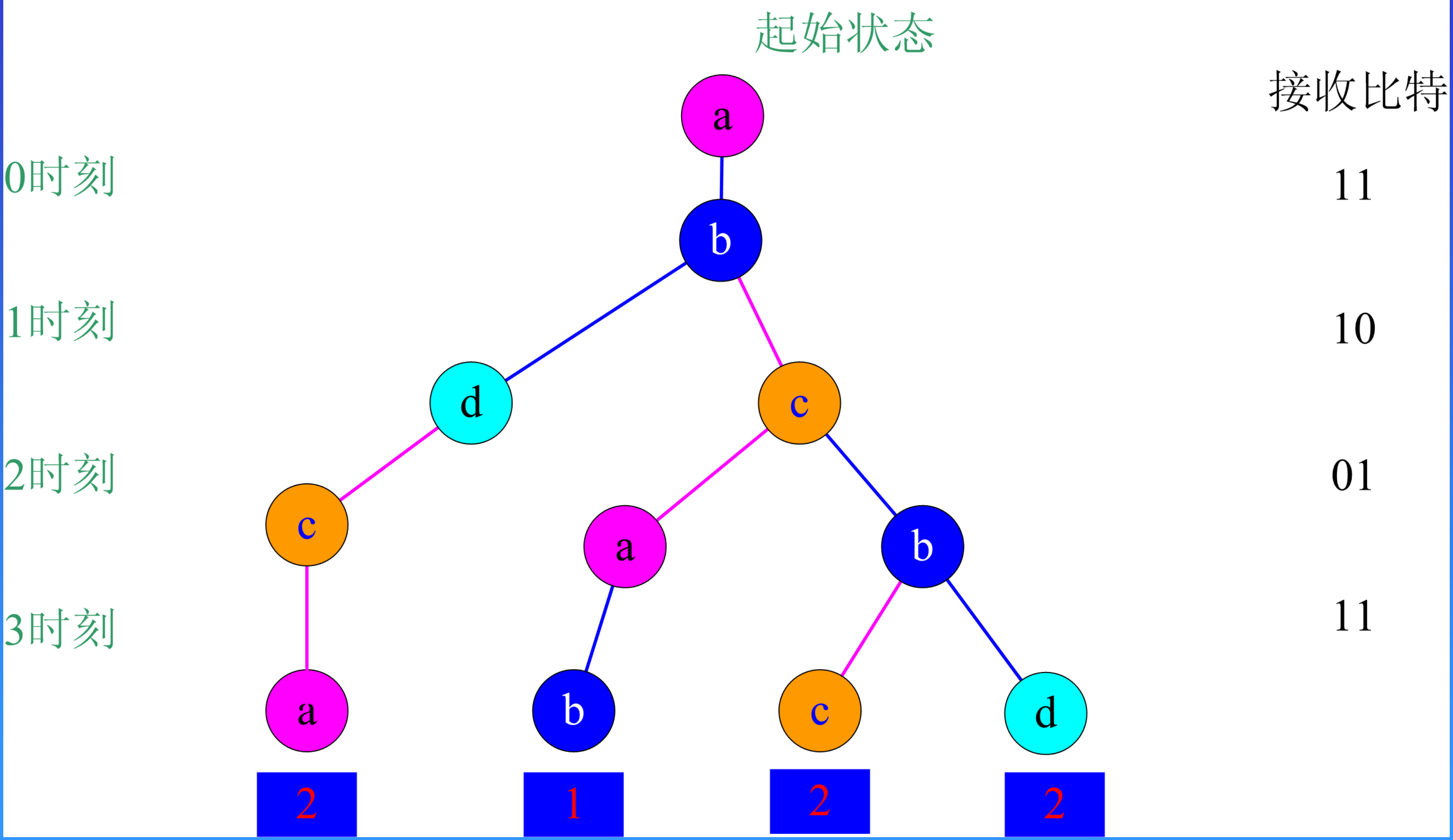
整理重画为



3时刻：假设输入为11


起始状态
接收比特
0时刻
1时刻
2时刻
3时刻
11
10
01
11
a
b
a
c
d
b
a
b
c
d
1
1
2
3
00
0
11
2
10
1
1
01
11
0
2
00
01
1
1
10
a
b
c
d
a
b
c
d
3
1
2
2
2
4
4
4

剪枝，重画：

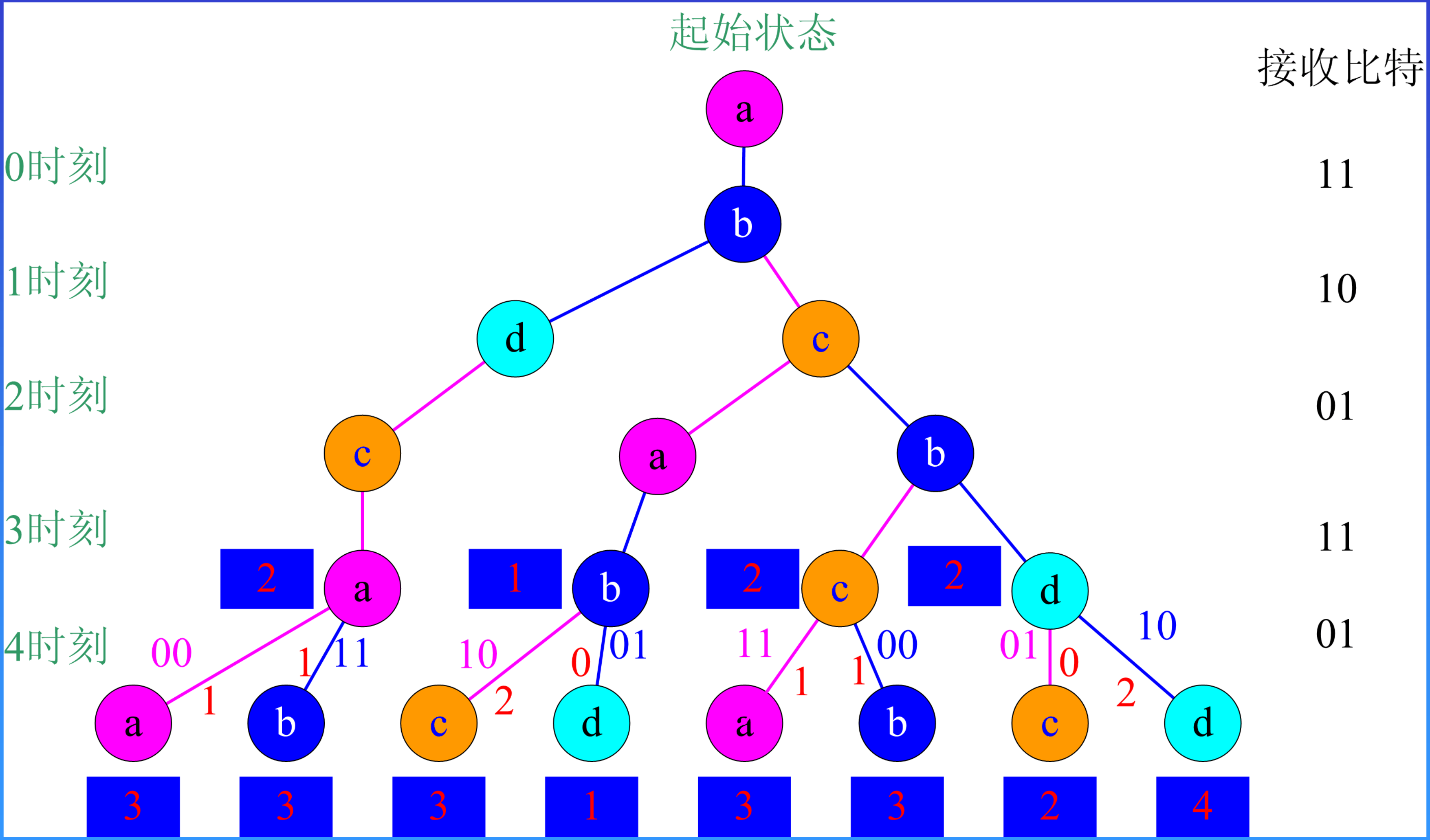


- 此时我们已经能确定：
  - 不管最终找出的最优路径是什么，它一定在第1步（0时刻）经过ab
  - 因此，如果必要的话，可以输出比特u_0的译码结果为

$$\hat{u}_0 = 1$$

- 注意：这个结果是ML译码对该比特的认为。
  - 不表明：发送的u_0肯定就是1
  - 但表明：不存在其他的判决方法，它的判决整体上能比我更可靠

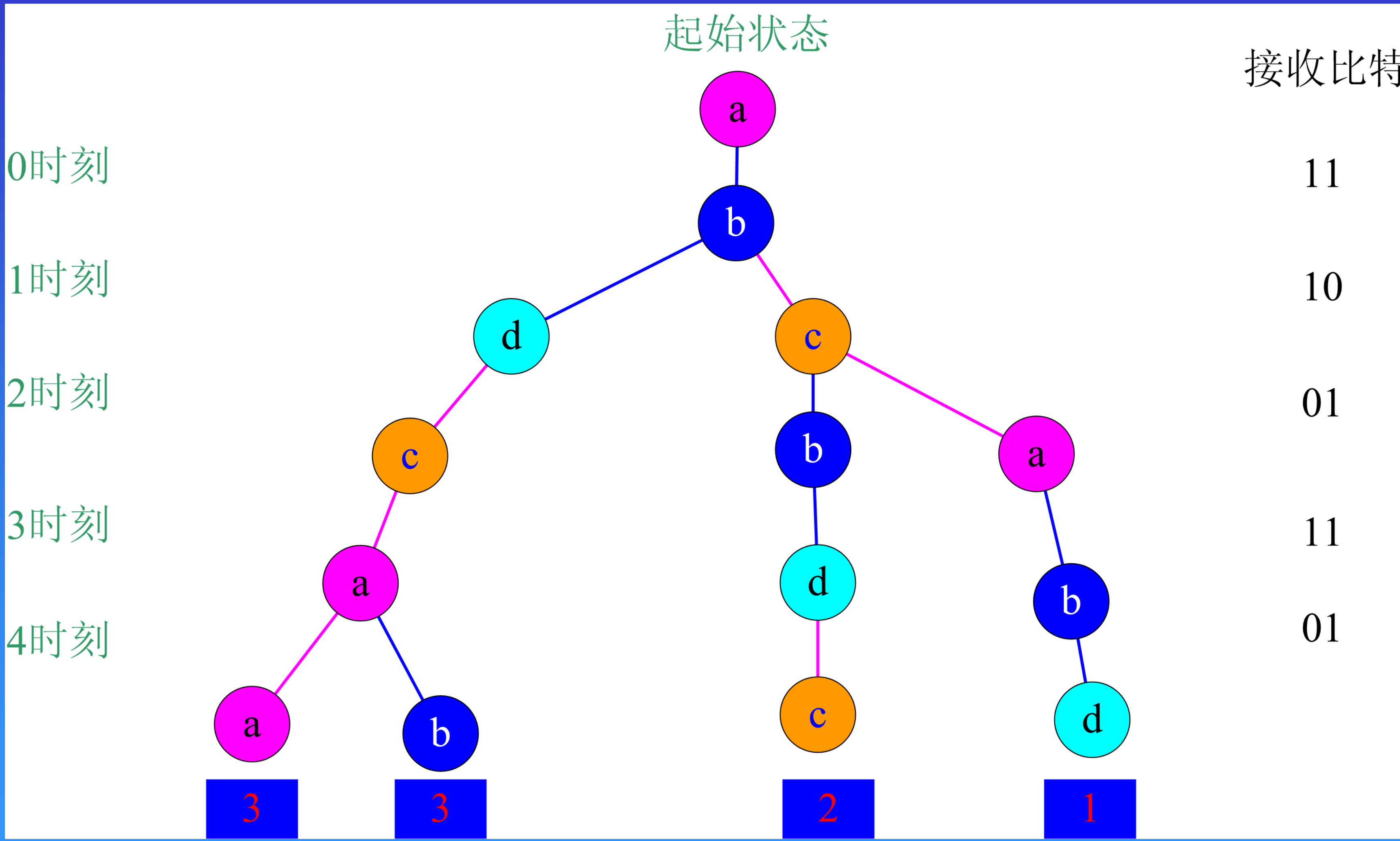
4时刻：假设输入为01


起始状态
接收比特
0时刻
1时刻
2时刻
3时刻
4时刻
11
10
01
11
01
a
b
d
c
c
a
b
a
b
c
d
2
1
2
2
00
1
1
11
10
2
0
01
11
1
1
00
01
0
2
10
a
b
c
d
a
b
c
d
3
3
3
1
3
3
2
4

新情况：

- 此时到达状态*a*的两条路径有相同的度量，如何确定谁是幸存路径？
  - 正确的做法：随便选一个
  - 也许最终的最优路径根本不在4时刻通过*a*
  - 但也可能：最终的最优路径的确在4时刻通过*a*
    - 此时：最终有两条并列最优的路径。若无其他信息，你没有依据去倾向其中的一条

于是得到：



起始状态
接收比特
0时刻
1时刻
2时刻
3时刻
4时刻
11
10
01
11
01
a
b
d
c
c
b
a
a
d
b
a
b
c
d
3
3
2
1

以后：

- 以后将一直持续这一过程
- 目前实践中卷积码一般是分组运用的：即每次发送L个信息比特
  - 编码器一般要在每L个信息比特之后缀上2个0（称为尾比特）以使编码器状态回零
  - 此时：译码器在进行$L+2$步后，输出到达a的幸存路径

Viterbi译码的复杂度随L线性增长：

- Viterbi译码中，时间每向前一走步，我们需要在8条候选路径中挑出4条幸存路径
- 运算量与信息比特数*L*成线性关系（信息比特数若加倍，需要的运算量也加倍）
- 而在穷举式ML搜索中，路径数随*L*指数增长
- 另外：本领域的专业人员习惯用格图，而不是树图，来表述Viterbi译码过程

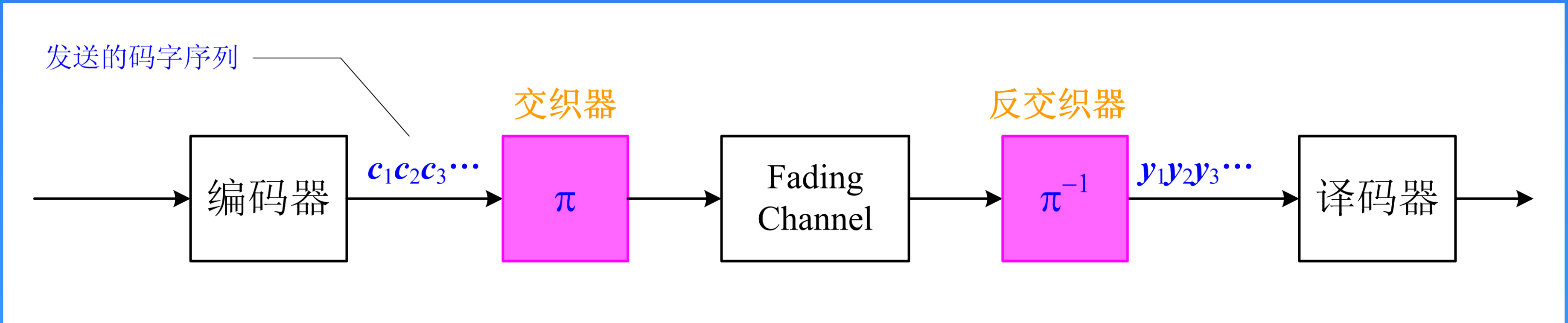
交织

- 交织就是洗牌，即打散次序，例如

  原序列：$a_1a_2a_3a_4a_5a_6a_7a_8$

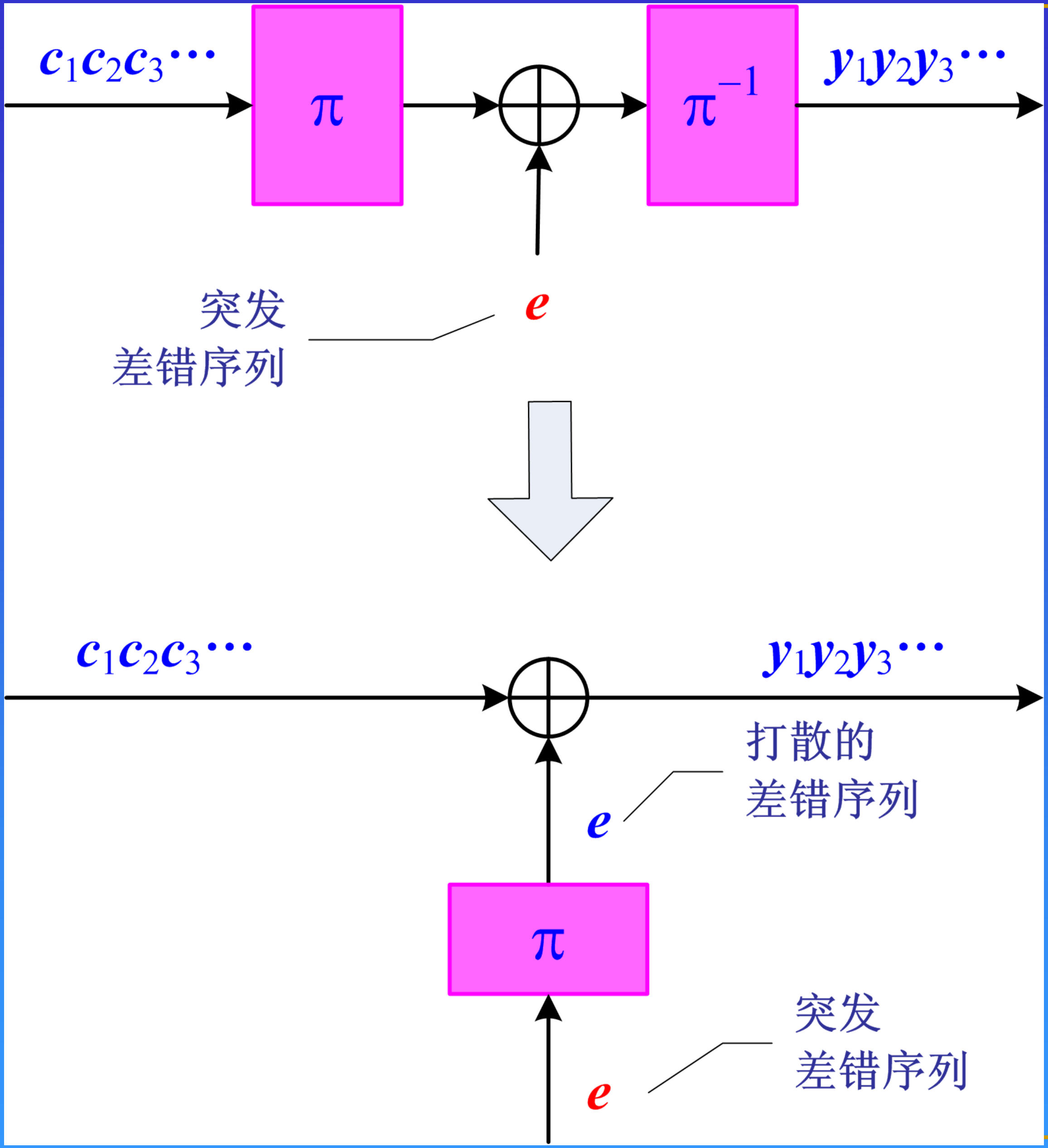
  交织后：$a_7a_1a_3a_6a_2a_5a_4a_8$

- 交织的用途之一是打散突发错（先前所学的编码大部分对突发错无能为力）



突发错

- 对于BSC信道，不同比特是否出错是独立的随机事件，这种类型的错误叫独立差错
- 无线衰落信道中的差错往往表现为突发差错，即：某一段时间差错率非常高，其他时间差错率非常低。
- 注：
  - 突发错≠连续出错。BSC信道也会出现连续的错误，只是出现概率要小得多（概率相乘的关系）。



发端把多个码字交织后发送，在信道中遇到突发错

收端反交织。

总效果是：信道错误的次序被打乱了

常用的交织器：分组交织（块交织）

按行写入，按列读出

a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}
a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}
a_{29}	a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}

假设是5个(7,4)汉明码码字

- 编码器输出的比特顺序

a_1	a_2	a_3	…	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	…	a_{34}	a_{35}

  - 若其中连续3个比特$a_{11}a_{12}a_{13}$出错，则第2个码字必然译错

- 信道发送的比特顺序为

a_1	a_8	a_{15}	…	a_{30}	a_3	a_{10}	a_{17}	a_{24}	…	a_{28}	a_{35}

  - 假设错误仍然发生在相同的位置上：$a_3a_{10}a_{17}$出错
  - 解交织后，这3个错分散在3个汉明码字上。每个码字错1个，都能译对。

编码最好能做到多好？

- 根据信道编码定理，如果码长n充分长，如果编码率R小于信道容量C，如果采用ML译码，那么一定存在一种编码，其错误率能随n的增加而趋向0。
- 前面讲的这些编码和Shannon的极限还有非常大的差距。
- 目前已存在能接近Shannon极限的编码
  - Turbo code
  - LDPC code

Coded Modulation

- 编码能降低误码率，但代价是频谱效率变低
- 高阶调制如8PSK、16QAM能提高频谱效率，但代价是误码率恶化
- 编码调制(CM)试图将二者结合起来，以获得二者各自的优点
  - 需要仔细设计，否则获得的可能是二者各自的缺点
  - 此领域的主要成就：TCM、BICM