Contents

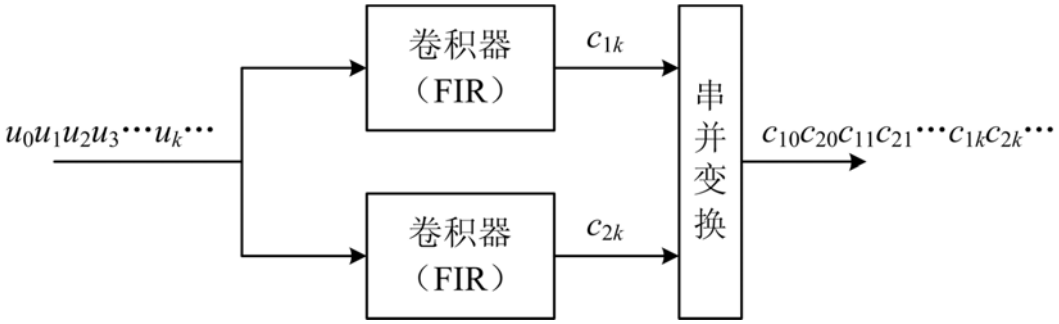
卷积码及维特比译码

前言

- 信息论告诉我们：当且仅当码长无穷时，速率趋进于信道容量，误码率趋近于0
- 分组码的缺陷
  - 译码依赖于码字内所有码位
  - 码长趋近于无穷时，译码时间也趋近于无穷，并且复杂度趋向于无穷
- 卷积码
  - 码长趋近于无穷，译码时间和复杂度有限

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。

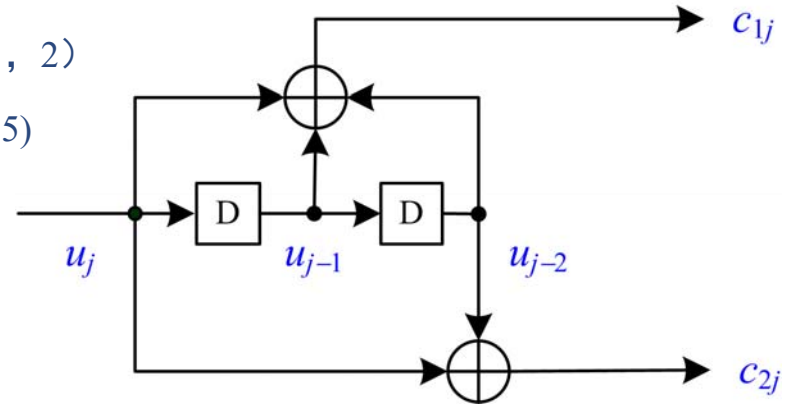


- 描述
  - (n,k,m)
    - k为每次输入到卷积编码器的bit数，
    - n为每个k元组码字对应的卷积码输出n元组码字，
    - m为编码存储度，m+1= K为编码约束度m称为约束长度
  - (a,b) ：是指生成多项式

卷积码编码器示例：

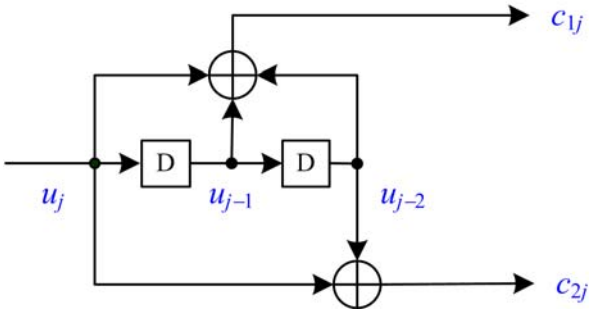
结构描述(2，1，2)

多项式描述(7,5)



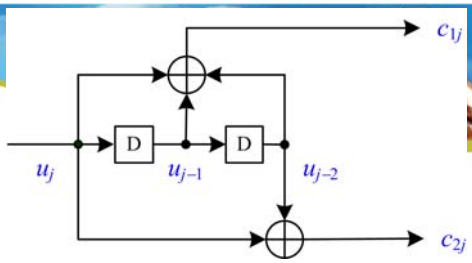
- 第1路的冲激响应是111，八进制表示是7；第2路的冲激响应是101，八进制表示是5。记此卷积码为(7,5)卷积码
- 两路的生成多项式为$1+x+x^2$，$1+x^2$

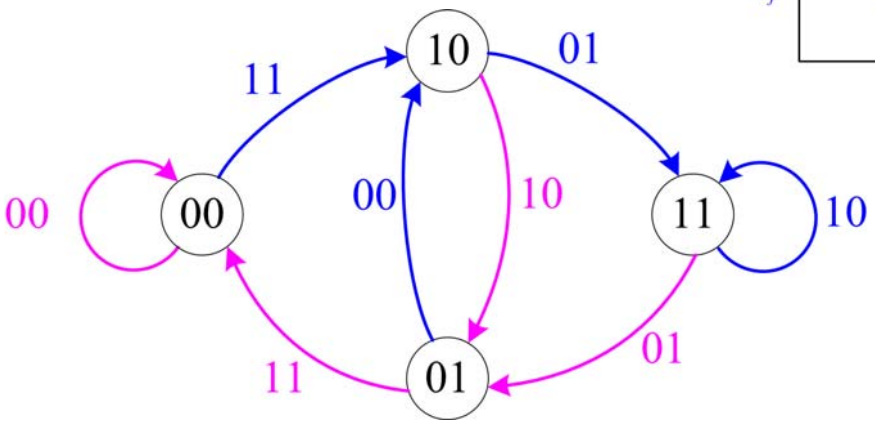
卷积码的状态



- 编码器中，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。未说明出发或到达时，“编码器状态”一词默认为到达状态
- (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。

状态转移图

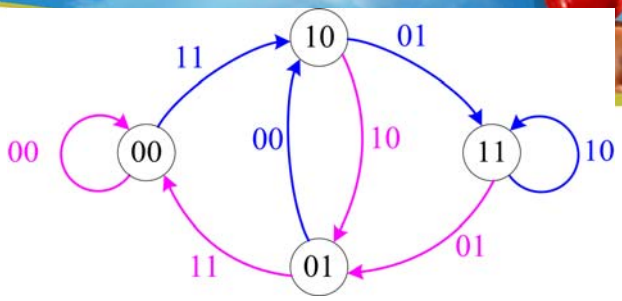


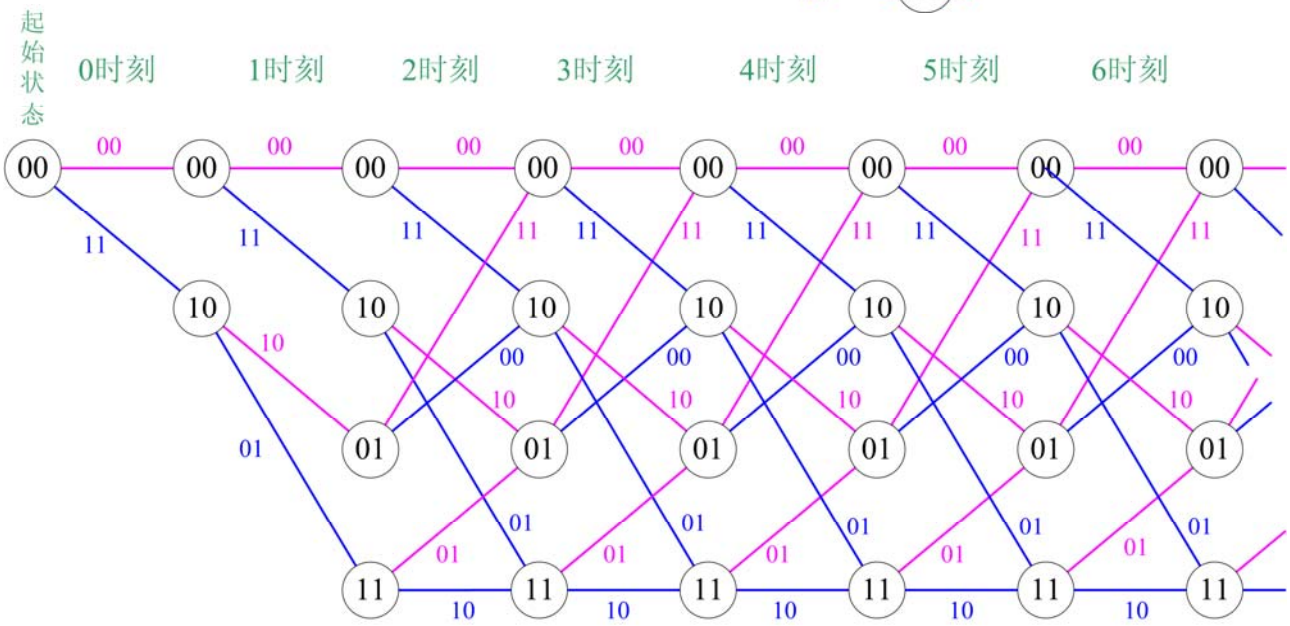


- 红线表示输入信息为0，蓝线表示输入信息为1。线旁的数字表示对应的编码器输出
- 从每个状态出发，可到达两个不同状态
- 每个到达状态都是来自两个不同的出发状态
- 输入的信息比特一定等于到达状态的第1位

格图

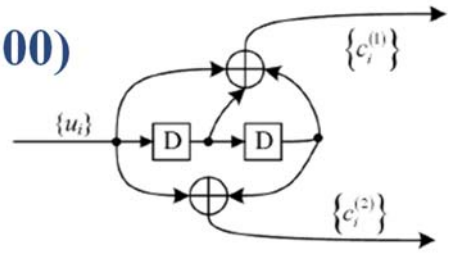
- 格图
  - 保持了时序的清晰性，表达简洁
  - 研究维特比译码算法的重要工具

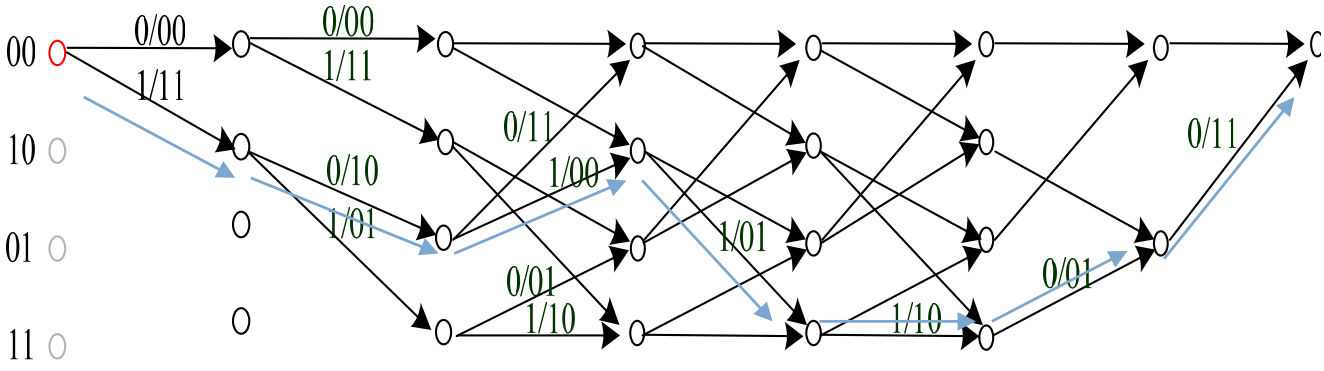




例：格图方法编码

(7, 5) 码输入的信息序列*u*=(1011100)



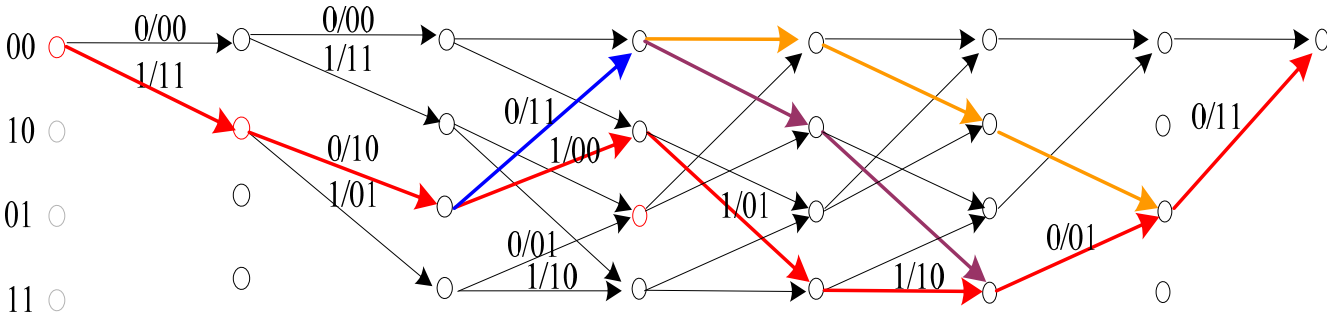


输出的编码序列*c*=(11 10 00 01 10 01 11)

最佳的译码算法 — 穷举法

- M个信息比特的所有可能的 2^M 种组合进行编码，把编码结果与接收序列比较，计算汉明距离
- 汉明距离最小的序列就是最有可能的发送序列
- 复杂度与存储量是信息比特数M的指数函数，实用性太差，只有理论意义
- 需要利用格图中的特殊结构来缩减运算量

递推到各状态的最短路径

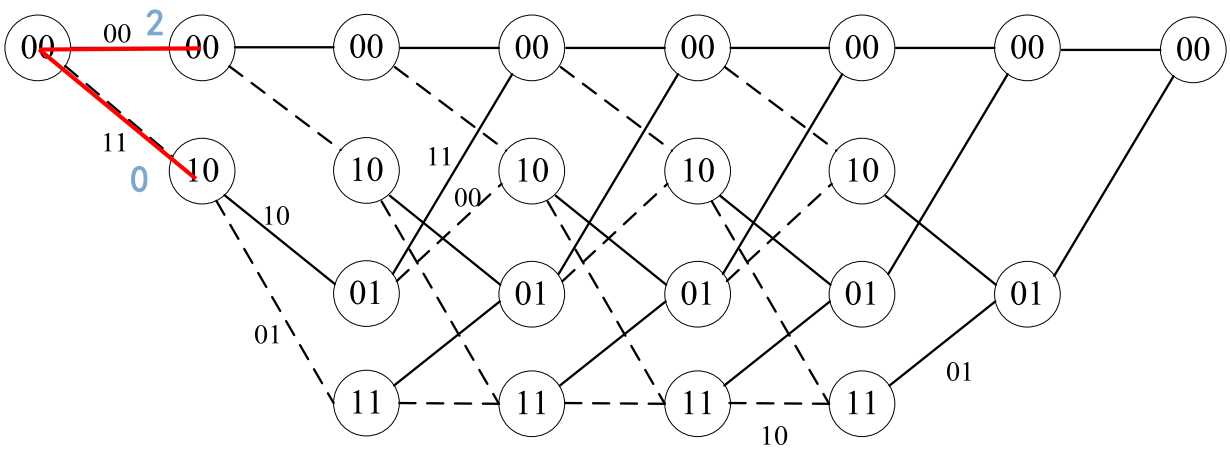


- 0状态起始→0状态结束
- 第n步时到达所有状态的最短路径分别已知，第n+1步到达各状态的最短路径易知
  - 路径减少1，复杂度降低1
  - 每个状态只保留最短距径：幸存路径
  - 局部非最短路径尽早删除，节省计算量

VA:第一步

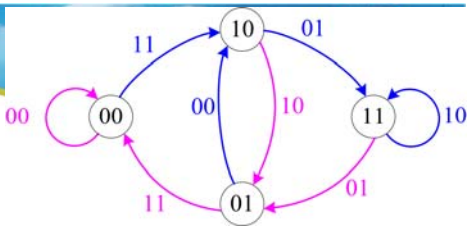
接收的有错误的序列

11 10 11 00 11 00 11

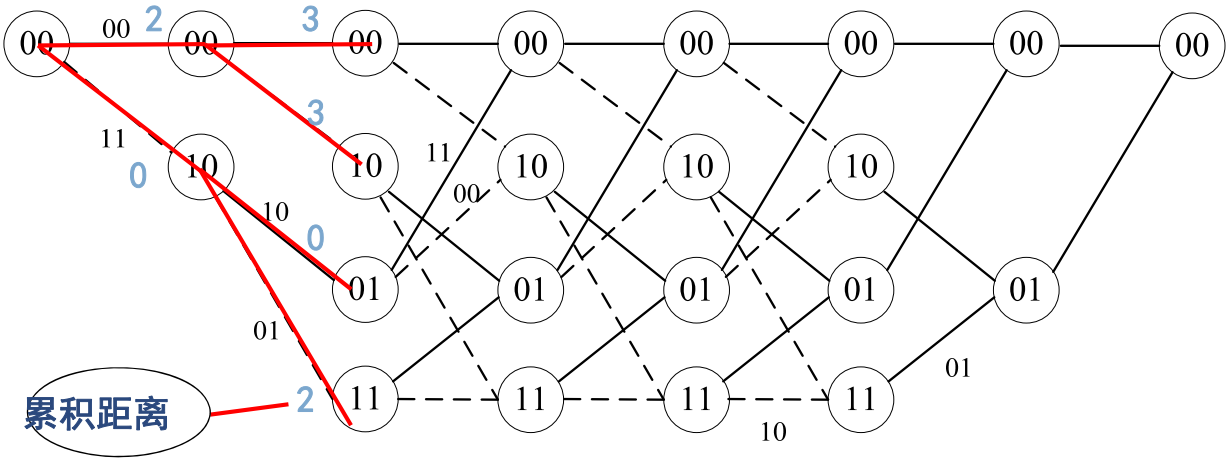


输出的编码序列c=(11 10 00 01 10 01 11)

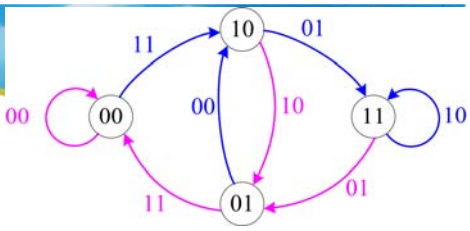
VA:第二步



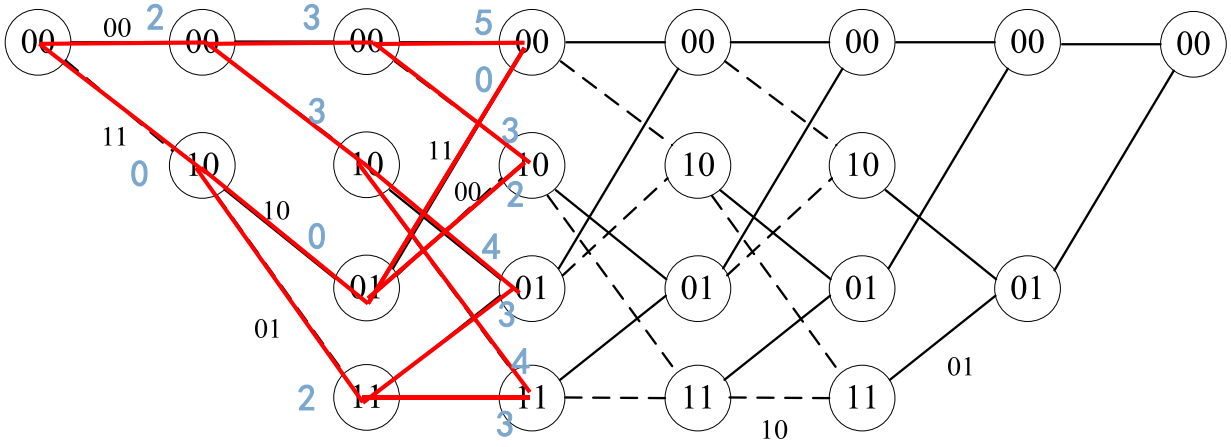
11 10 11 00 11 00 11



VA:第三步

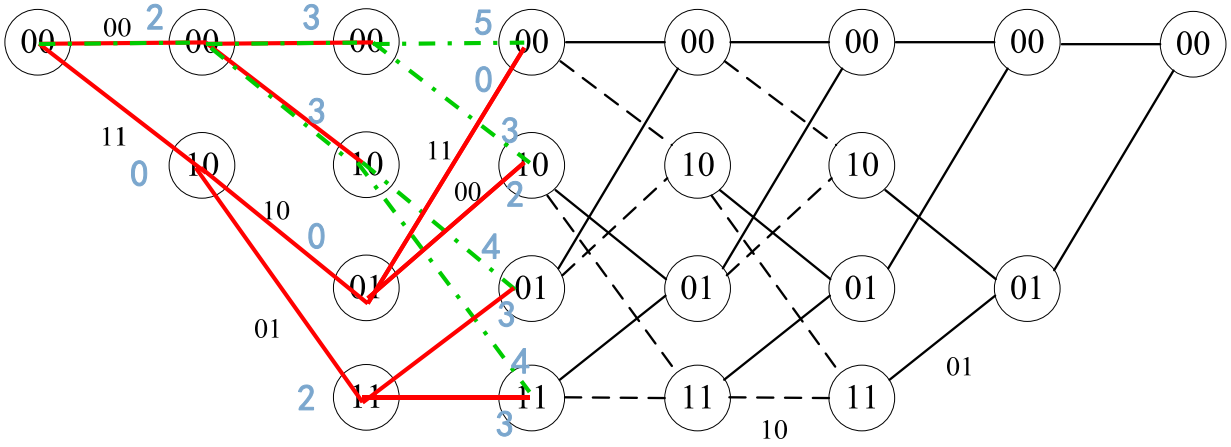


11 10 11 00 11 00 11



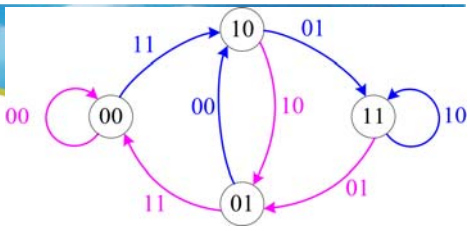
VA:第三步后的幸存路径

11 10 11 00 11 00 11

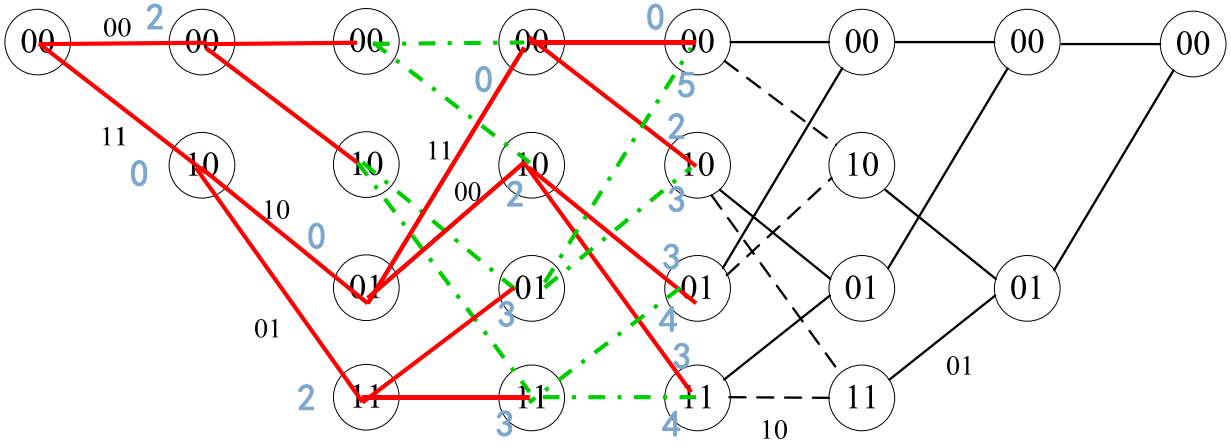


每个状态只保留最短距径

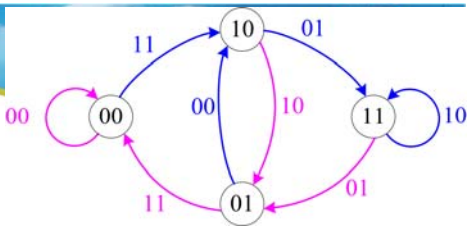
VA:第四步



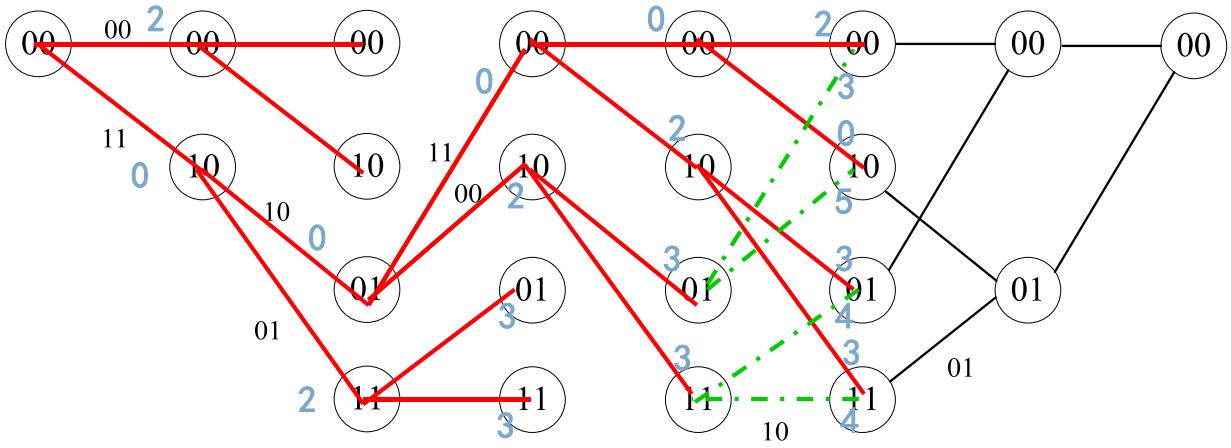
11 10 11 00 11 00 11



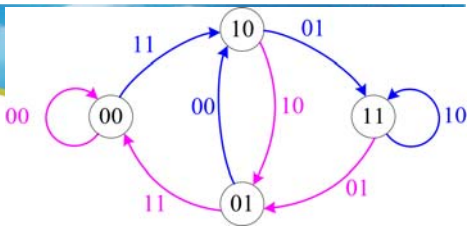
VA:第五步

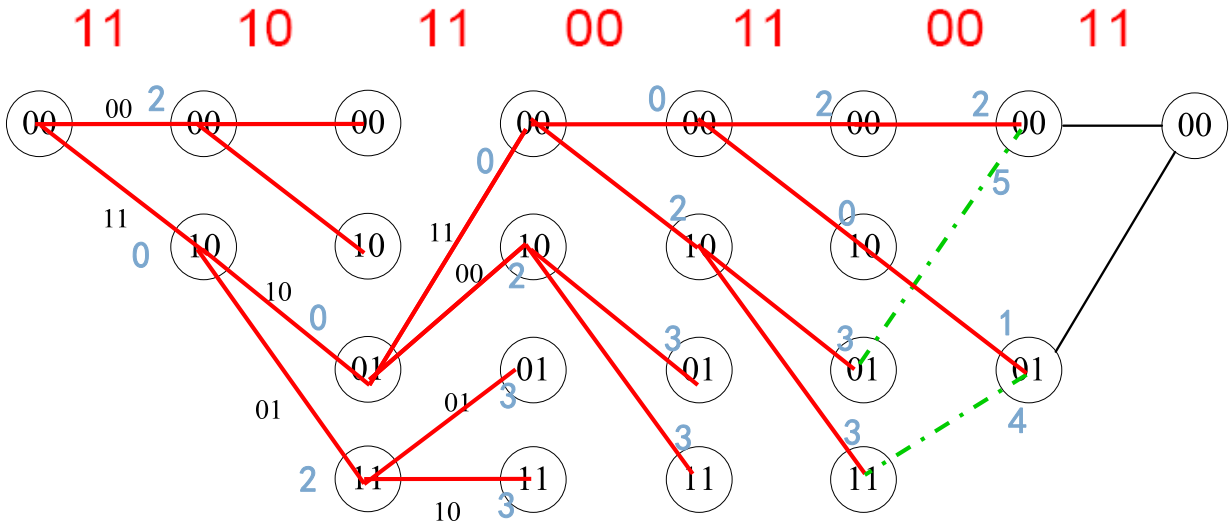


11 10 11 00 11 00 11

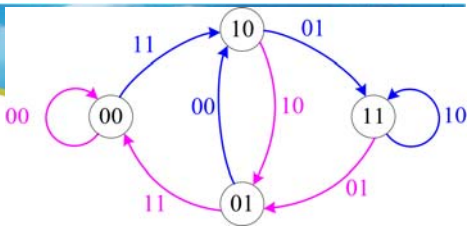


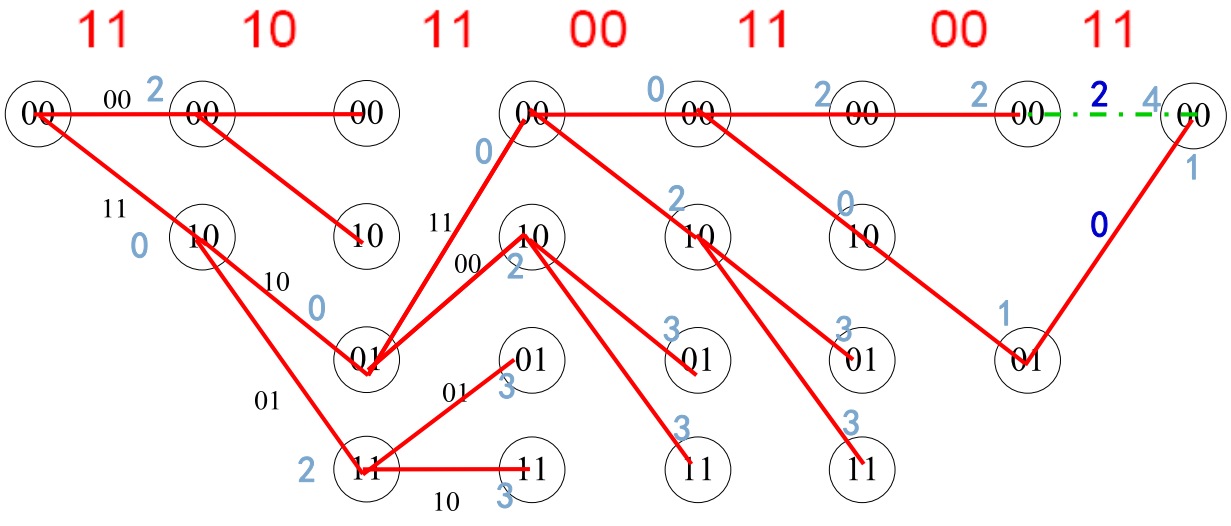
VA:第六步


10
01
11
00
00
10
11
10
00
01
11
01

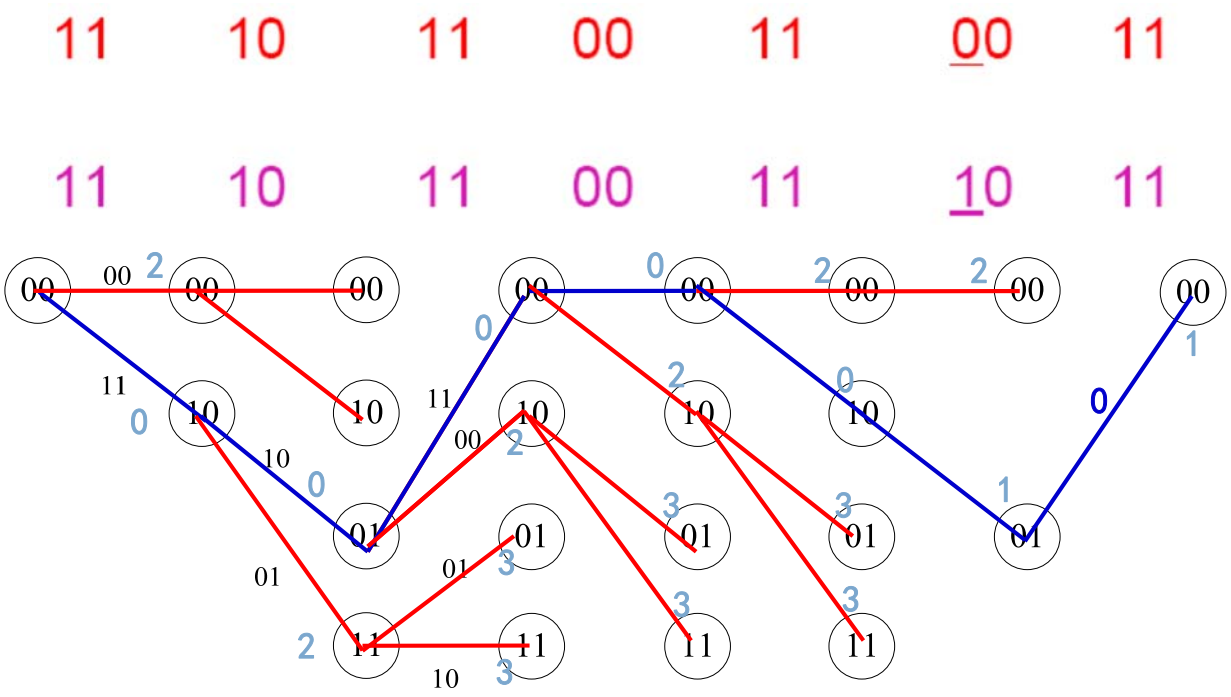

11 10 11 00 11 00 11
00 10 01 11

VA:第七步

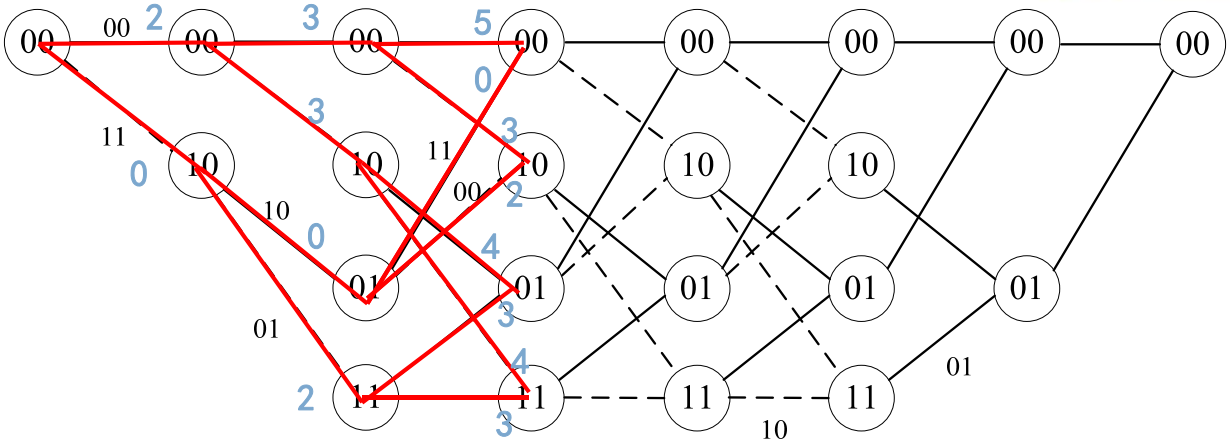

10
01
11
00
00
10
11
10
00
01
11
01


11 10 11 00 11 00 11
00 10 01 11

VA:最终结果



VA的一些定义



- 称汉明距离为度量,累积距离为累积度量,分支的距离为分支度量
- 到达每步的累积度量最小的那条路径为幸存路径
- 全局最优路径为最大似然路径

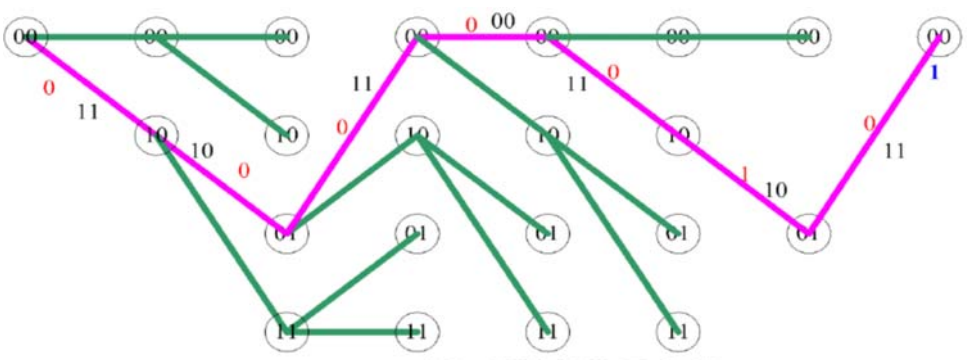
VA算法总结

- VA算法
  - 用前一状态的幸存路径与本步的分支度量计算出到达每个状态的累积度量，保留幸存路径
  - 从第一步开始迭代计算到最后一步得到全局最优路径
- 复杂度
  - 每步需要比较多路径数2^{K+m}，共L步, 复杂度为$O(L2^{K+m})$
  - 穷举法的复杂度$O(2^{L})$

K是约束长度，m是寄存器个数

截短VA算法

- *VA*译码时延为$L+K$个时钟周期
- 译码进行到一定步数后前面路径可能已聚合
  - 如果已聚合，*ML*路径必然经过此聚合路径
- 如无聚合则以最有可能的路径(当前累积度量最小的幸存路径),输出前面的比特
  - 局部最佳不等于全局最优，性能有所损失
- 当$L>5K$时,性能非常接近最大似然译码



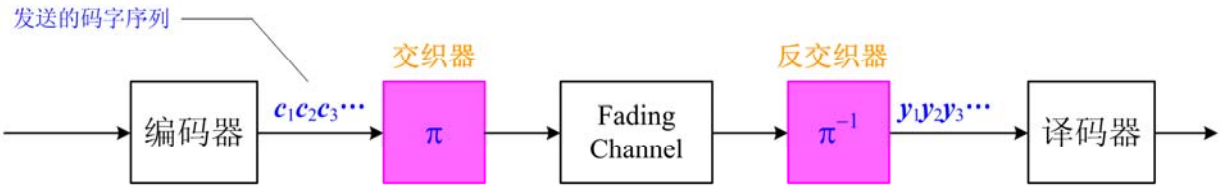
交织

- 交织就是洗牌，即打散次序，例如

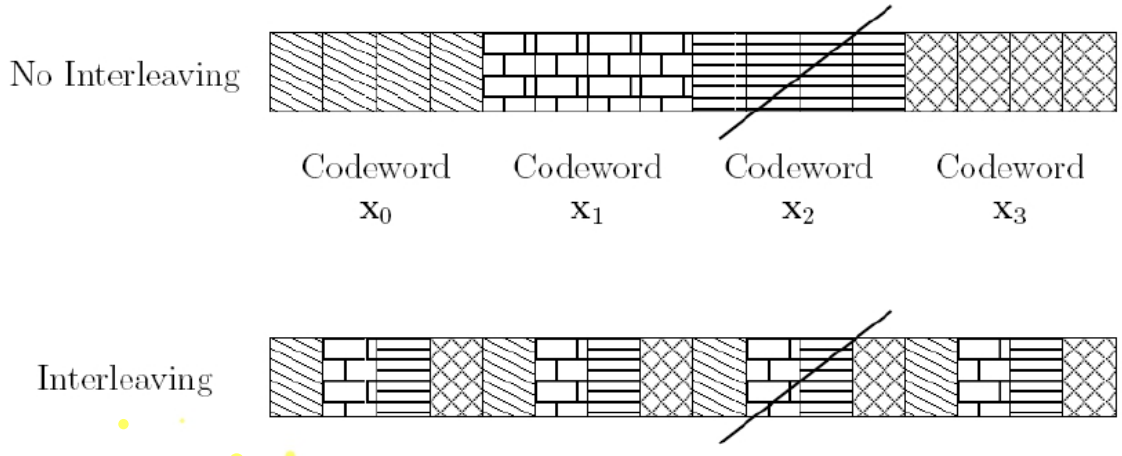
  原序列：$a_1a_2a_3a_4a_5a_6a_7a_8$

  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）
- 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



- 将突发误码转换为零星误码



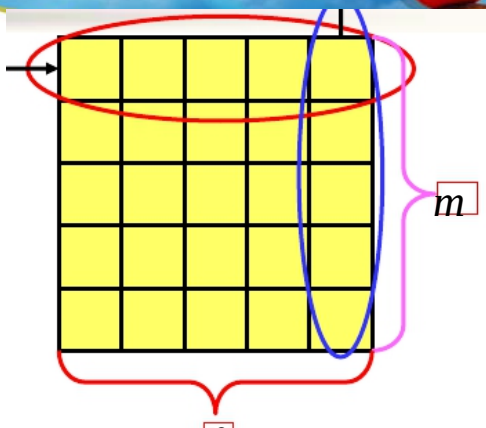
这个词的含义是“交织”！

常用交织器：行列交织器

❖基本原理

- 为了对付突发的信道差错，交织器改变发送码元的时间顺序
- 将原本相邻的码元在时间上的距离最大化
- 例子：考虑一个（n，　k）分组码，其交织后的输出为

交织器的性能



❖宽度

- 就是分组码的码长n
- 决定于所采用的分组码

❖深度

- 深度m决定了相邻码元交织后的间隔
- m又称交织深度
- 若分组码能纠b个突发错误，则交织后能纠mb个突发错误

交织深度越大，则纠突发误码能力越强

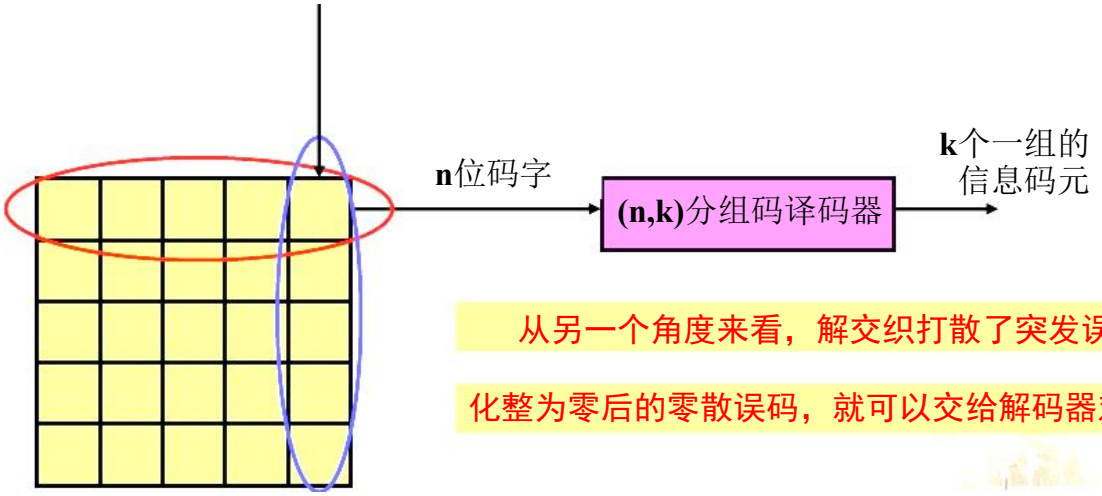
是不是交织深度越大越好？

不是，交织深度带来解码延时

解交织

❖在接收端必须解交织

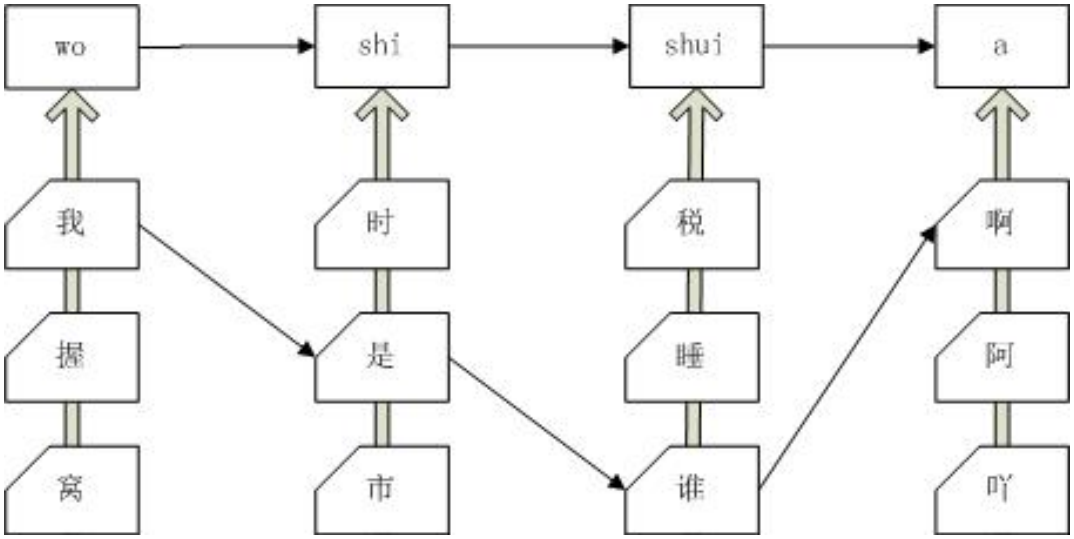
- 在通信接收机设计中中一定要学会“以其人之道，还治其人之身”的办法
- 解交织的方法就是“竖着写入，横着读出”



从另一个角度来看，解交织打散了突发误码

化整为零后的零散误码，就可以交给解码器对付了

Viterbi被广泛应用到分词、词性标注、输入法、语音识别等场景

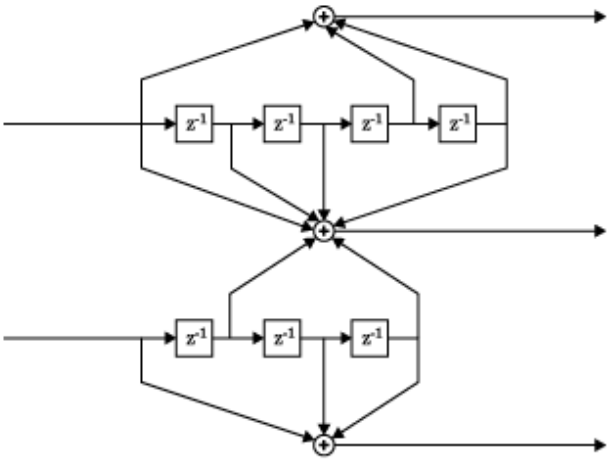


- 输入法：
  - 转移概率即是二元语言模型
  - 其输出概率即是多音字对应不同拼音的概率。
- 将上图中的拼音换成语音，就成了语音识别问题，转移概率仍然是二元语言模型，其输出概率则是语音模型，即语音和汉字的对应模型。

如何读matlab中的CC函数

trellis = poly2trellis([5 4],[23 35 0; 0 5 13])

- 码率2/3
- 状态个数：2^7=128
- constraint length of the upper path is 5
- and the constraint length of the lower path is 4



	二进制	八进制
	1 0 , 0 1 1	23
	1 1 , 1 0 1	35
	0 ,1 0 1	5
	1 ,0 1 1	13