14. 卷积码

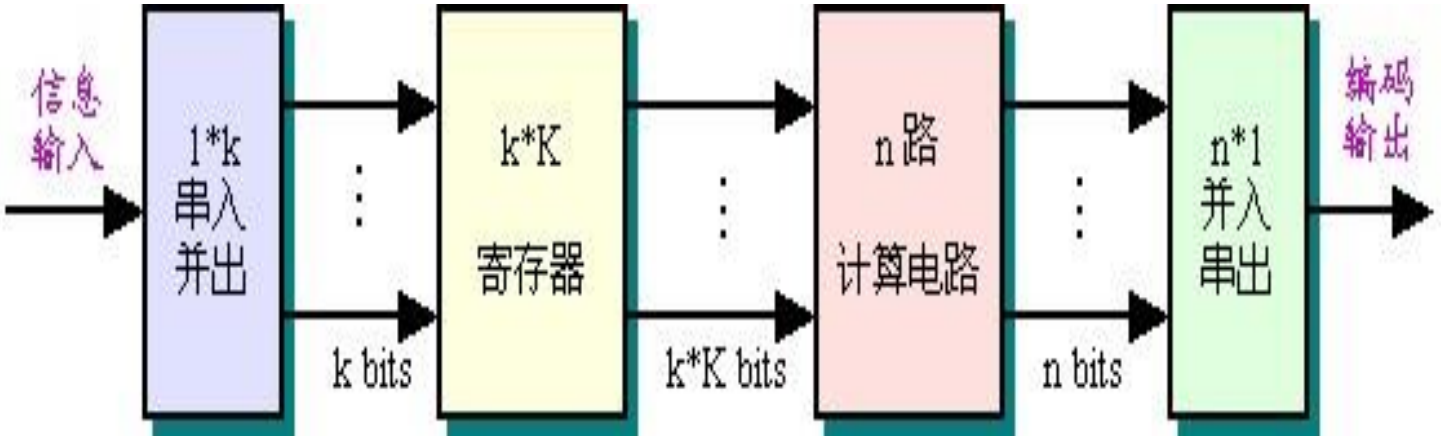
- 1955年，Elias首先提出了卷积码的概念
- 稍后，Wozencraft提出了**序列译码算法**，且很快就开始了实验研究
- 1963年，梅西提出了**门限译码算法**，在有线和无线通信的数据传输中得到应用
- 1967年，Viterbi提出了一种最佳最大似然译码算法—**Viterbi译码**
- Viterbi译码算法+改进的序列译码算法使卷积码在七十年代初期就用于深空和卫星通信中
- 卷积码在无线通信系统中的应用：LTE标准中规定控制信道采用咬尾卷积码

目录

- 卷积码的编码
- 卷积码的图描述
- 距离分布多项式
- 性能分析及构造准则
- 卷积码的译码
- 打孔卷积码和咬尾卷积码

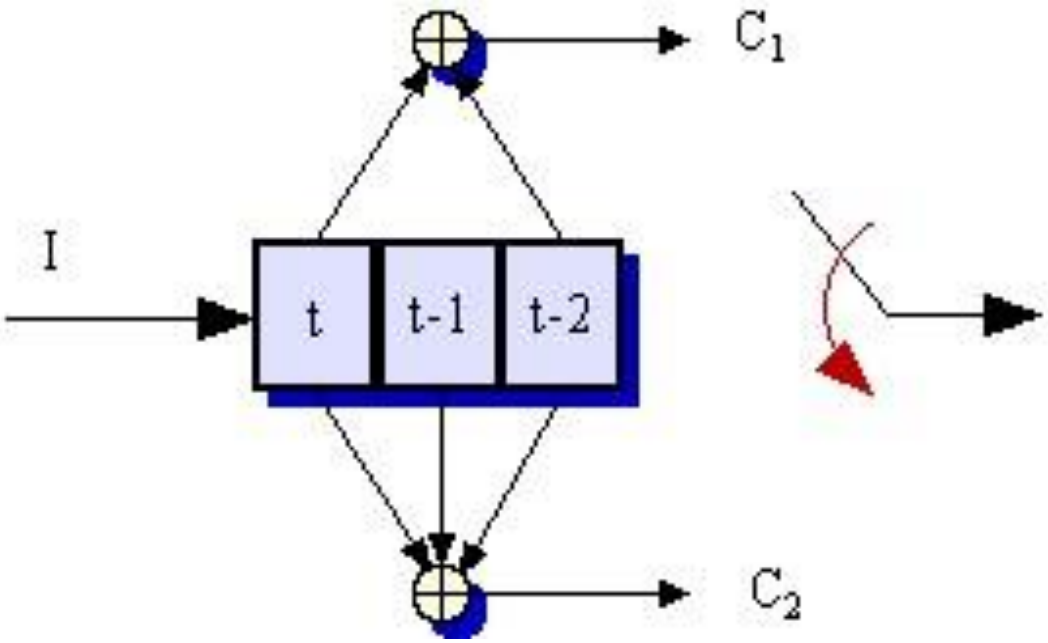
卷积码的编码

- 卷积码的编码器
  - 一般用 (n, k, K) 或 (K, R=k/n) 来描述一种卷积码，其中： K为约束长度， R = k/n为编码速率.
  - 卷积码编码器原理图如下：



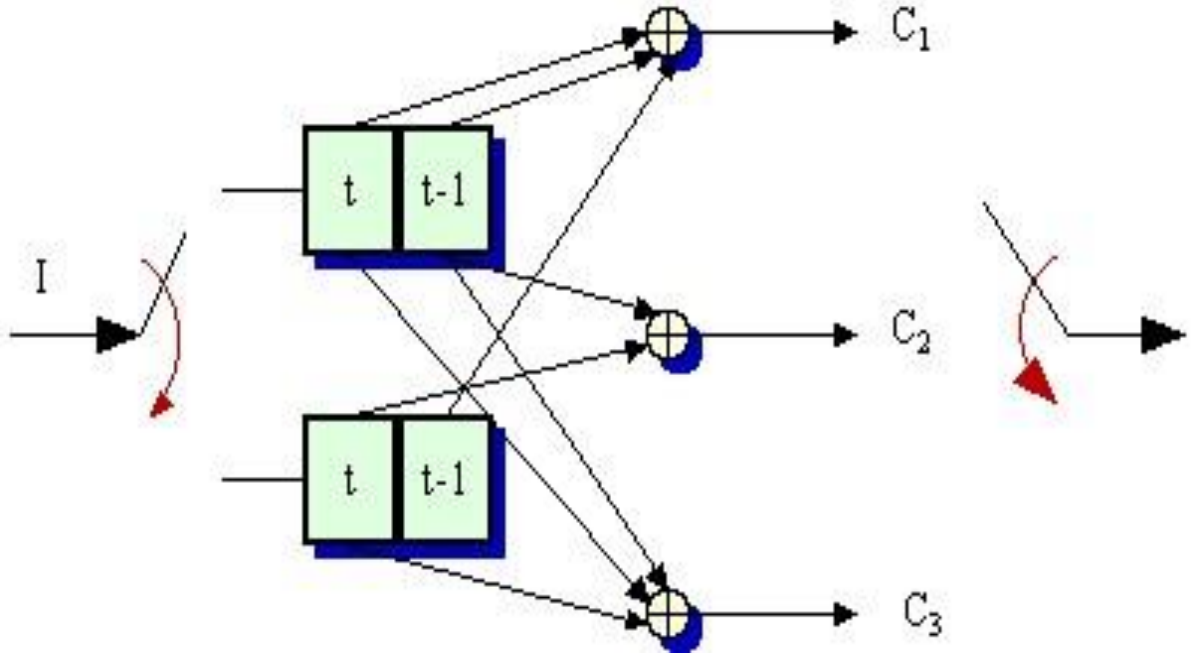
卷积码的编码

- 卷积码的编码器
  - 例16.1:(3，1/2)卷积码的编码器



卷积码的编码

- 卷积码的编码器
  - 例16.2:(2, 2/3)卷积码的编码器



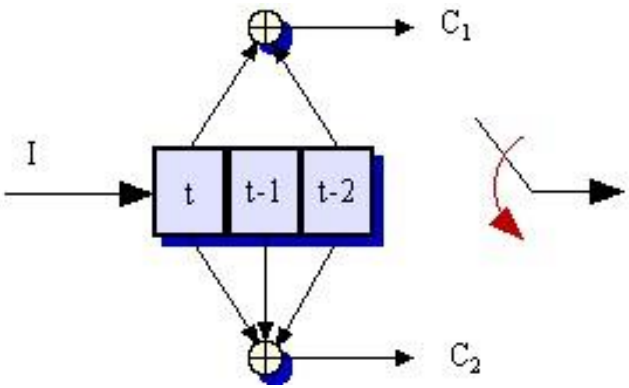
- 标量生成矩阵
  - 例16.1: $\begin{cases}\mathbf{g}^{(1)}=\begin{pmatrix}g_0^{(1)} & g_1^{(1)} & g_2^{(1)}\end{pmatrix}=\begin{pmatrix}1 & 0 & 1\end{pmatrix}\\ \mathbf{g}^{(2)}=\begin{pmatrix}g_0^{(2)} & g_1^{(2)} & g_2^{(2)}\end{pmatrix}=\begin{pmatrix}1 & 1 & 1\end{pmatrix}\end{cases}$

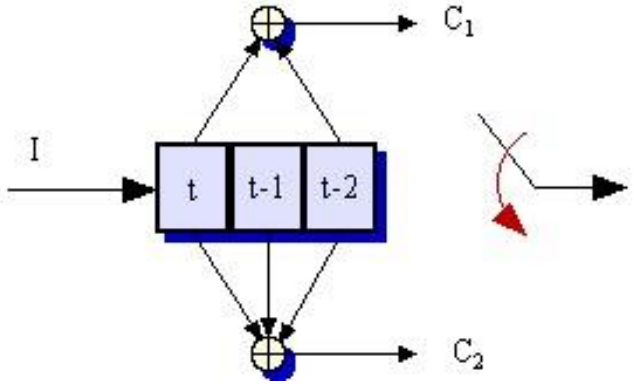
    (续)

$$\begin{cases}\mathbf{v}^{(1)}=\mathbf{u}*\mathbf{g}^{(1)}\\ \mathbf{v}^{(2)}=\mathbf{u}*\mathbf{g}^{(2)}\end{cases}\qquad\begin{cases}v_t^{(1)}=u_t+u_{t-2}\\ v_t^{(2)}=u_t+u_{t-l}+u_{t-2}\end{cases}$$

若$\mathbf{u}=\begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix}$

则$\begin{cases}\mathbf{v}^{(1)}=\begin{pmatrix}1 & 1 & 1 & 0 & 0 & 1\end{pmatrix}\\ \mathbf{v}^{(2)}=\begin{pmatrix}1 & 0 & 0 & 0 & 1 & 1\end{pmatrix}\end{cases}$





- 标量生成矩阵
  - 例16.1:设$\mathbf{u}=\begin{pmatrix}u_0 & u_1 & u_2 & \cdots\end{pmatrix}$

    (续) $\mathbf{v}=\begin{pmatrix}v_0^{(1)} & v_0^{(2)} & v_1^{(1)} & v_1^{(2)} & v_2^{(1)} & v_2^{(2)} & \cdots\end{pmatrix}$

    $\mathbf{v}=\mathbf{u}\cdot\mathbf{G}$

则$\mathbf{G}=\begin{pmatrix} g_0^{(1)} & g_0^{(2)} & g_1^{(1)} & g_1^{(2)} & g_2^{(1)} & g_2^{(2)} & & \\ & & g_0^{(1)} & g_0^{(2)} & g_1^{(1)} & g_1^{(2)} & g_2^{(1)} & g_2^{(2)} \\ & & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$

$=\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & & \\ & & 1 & 1 & 0 & 1 & 1 & 1 \\ & & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$

卷积码的编码

- 标量生成矩阵
  - 例16.2:(续)

$$\mathbf{g}_1^{(1)} = \begin{pmatrix} g_{1,0}^{(1)} & g_{1,1}^{(1)} \end{pmatrix} = \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad \mathbf{g}_2^{(1)} = \begin{pmatrix} g_{2,0}^{(1)} & g_{2,1}^{(1)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

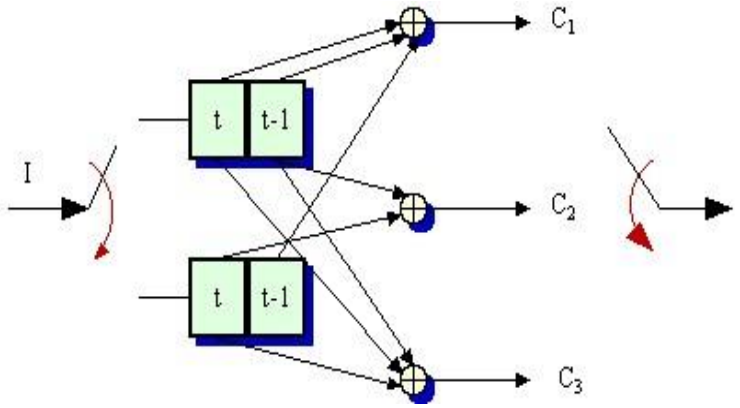
$$\mathbf{g}_1^{(2)} = \begin{pmatrix} g_{1,0}^{(2)} & g_{1,1}^{(2)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix} \qquad \mathbf{g}_2^{(2)} = \begin{pmatrix} g_{2,0}^{(2)} & g_{2,1}^{(2)} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

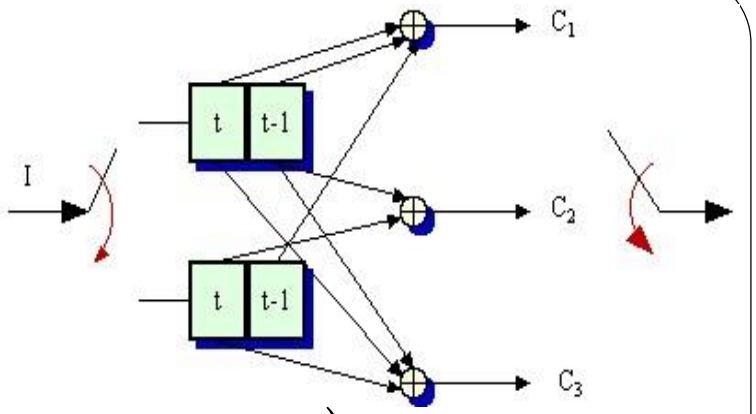
$$\mathbf{g}_1^{(3)} = \begin{pmatrix} g_{1,0}^{(3)} & g_{1,1}^{(3)} \end{pmatrix} = \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad \mathbf{g}_2^{(3)} = \begin{pmatrix} g_{2,0}^{(3)} & g_{2,1}^{(3)} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

$$\mathbf{v}^{(1)} = \mathbf{u}^{(1)} * \mathbf{g}_1^{(1)} + \mathbf{u}^{(2)} * \mathbf{g}_2^{(1)}$$

$$\mathbf{v}^{(2)} = \mathbf{u}^{(1)} * \mathbf{g}_1^{(2)} + \mathbf{u}^{(2)} * \mathbf{g}_2^{(2)}$$

$$\mathbf{v}^{(3)} = \mathbf{u}^{(1)} * \mathbf{g}_1^{(3)} + \mathbf{u}^{(2)} * \mathbf{g}_2^{(3)}$$





- 标量生成矩阵
  - 例16.2:(续）

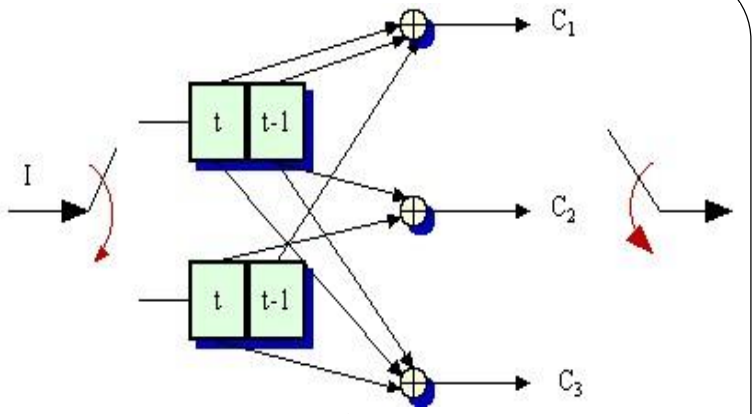
设$\mathbf{u}=\left(u_0^{(1)} \quad u_0^{(2)} \quad u_1^{(1)} \quad u_1^{(2)} \quad u_2^{(1)} \quad u_2^{(2)} \quad \cdots\right)$

$\mathbf{v}=\left(v_0^{(1)} \quad v_0^{(2)} \quad v_0^{(3)} \quad v_1^{(1)} \quad v_1^{(2)} \quad v_1^{(3)} \quad \cdots\right)$

$\mathbf{v}=\mathbf{u}\cdot\mathbf{G}$

则$\mathbf{G}=$

$$\begin{pmatrix}
g_{1,0}^{(1)} & g_{1,0}^{(2)} & g_{1,0}^{(3)} & g_{1,1}^{(1)} & g_{1,1}^{(2)} & g_{1,1}^{(3)} & & & \\
g_{2,0}^{(1)} & g_{2,0}^{(2)} & g_{2,0}^{(3)} & g_{2,1}^{(1)} & g_{2,1}^{(2)} & g_{2,1}^{(3)} & & & \\
 & & & g_{1,0}^{(1)} & g_{1,0}^{(2)} & g_{1,0}^{(3)} & g_{1,1}^{(1)} & g_{1,1}^{(2)} & g_{1,1}^{(3)} \\
 & & & g_{2,0}^{(1)} & g_{2,0}^{(2)} & g_{2,0}^{(3)} & g_{2,1}^{(1)} & g_{2,1}^{(2)} & g_{2,1}^{(3)} \\
 & & & & & & \ddots & \ddots & \ddots
\end{pmatrix}$$



- 标量生成矩阵
  - 例16.2:(续）

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

若$\mathbf{u}=(1\quad 1\quad 0\quad 1\quad 1\quad 0)$

则$\mathbf{v}=(1\quad 1\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0\quad 1\quad 1\quad 1\quad 1)$

- 标量生成矩阵
  - 一般地，(K, k/n)卷积码的标量生成矩阵为

$$\mathbf{G}=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{K-1} & & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} & & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_{K-3} & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} & \\ & & & \ddots & & & & \ddots\end{bmatrix}$$

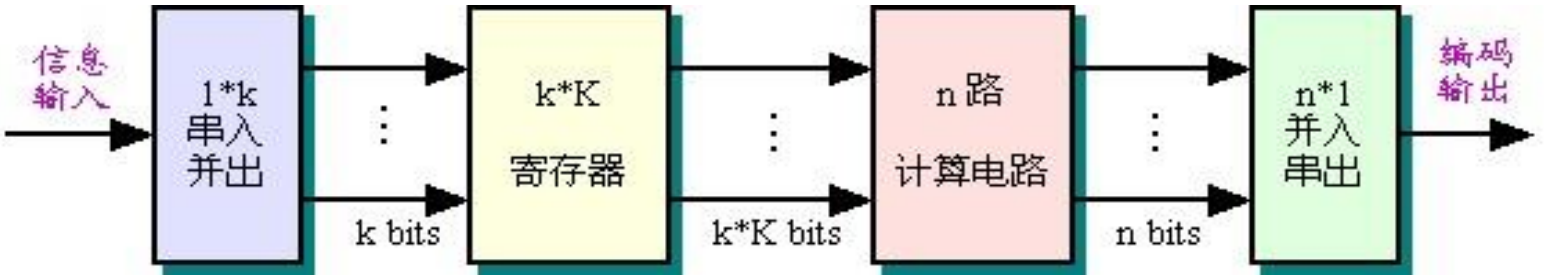
其中G_γ ($\gamma = 0, 1, \ldots, K-1$)是k×n阶矩阵，

性质：(1) **G**中每一k行组恒等于前面的行组，但右移了n位

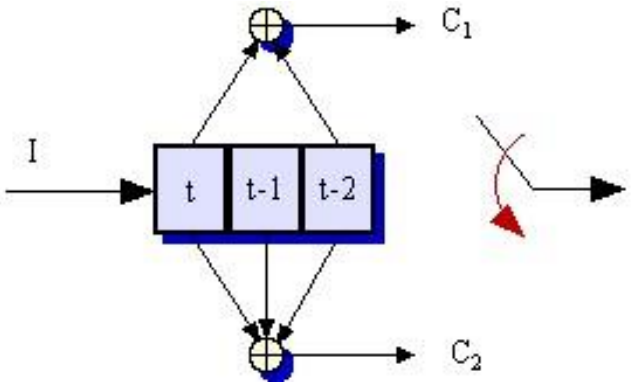
(2) 与信息序列为任意长相应，**G**是半无限矩阵

卷积码的编码

- 编码效率
  - 设信息序列$\mathbf{u}^{(i)}$的长度为L，　$i = 1, 2, \ldots, k$
  - 编码序列$\mathbf{v}^{(j)}$的长度为$L+K-1$，　$j = 1, 2, \ldots, n$
  - 信息比特数为 kL
  - 全部码包含 $n(K+L-1)$ 比特.
  - 若$\mathbf{V} = \mathbf{U} \cdot \mathbf{G}_L$, 则矩阵$\mathbf{G}_L$是矩阵$\mathbf{G}$的截短
  - 此时相当于 $(n(K+L-1), kL)$ 线性分组码
  - 效率：$R_L = R \cdot \left(1 - \frac{K-1}{K+L-1}\right)$



串行序列 长度kL　　并行序列 长度L　　并行序列 长度L+K-1　　串行序列 长度n(L+K-1)



- 多项式生成矩阵
  - 例16.1:(续)(3, 1/2)卷积码

$$\mathbf{v}^{(1)} = \mathbf{u} * \mathbf{g}^{(1)} \Rightarrow v^{(1)}(x) = u(x) \cdot g^{(1)}(x)$$

$$\mathbf{v}^{(2)} = \mathbf{u} * \mathbf{g}^{(2)} \Rightarrow v^{(2)}(x) = u(x) \cdot g^{(2)}(x)$$

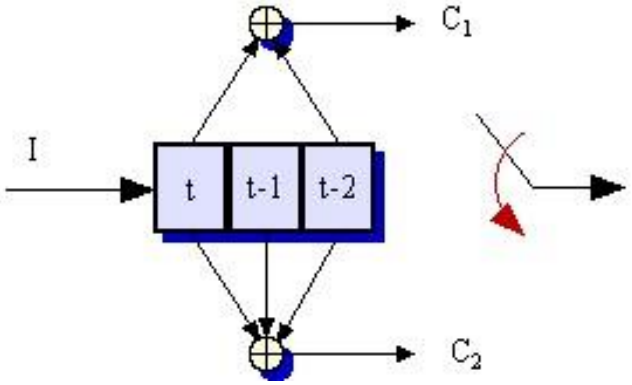
其中$u(x) = u_0 + u_1 x + u_2 x^2 + \cdots$

$$v^{(1)}(x) = v_0^{(1)} + v_1^{(1)} x + v_2^{(1)} x^2 + \cdots$$

$$v^{(2)}(x) = v_0^{(2)} + v_1^{(2)} x + v_2^{(2)} x^2 + \cdots$$

而$g^{(1)}(x) = g_0^{(1)} + g_1^{(1)} x + g_2^{(1)} x^2 = 1 + x^2$

$$g^{(2)}(x) = g_0^{(2)} + g_1^{(2)} x + g_2^{(2)} x^2 = 1 + x + x^2$$



- 多项式生成矩阵
  - 例16.1:(续)

有$\left(v^{(1)}(x) \quad v^{(2)}(x)\right)=u(x)\cdot\left(g^{(1)}(x) \quad g^{(2)}(x)\right)$

或$\mathbf{v}(x)=\mathbf{u}(x)\cdot\mathbf{G}(x)$

其中$\mathbf{G}(x)=\left(1+x^2 \quad 1+x+x^2\right)$

若$\mathbf{u}=\left(1 \quad 1 \quad 0 \quad 1\right)$

则$u(x)=1+x+x^3$

$$v^{(1)}(x)=1+x+x^2+x^5 \Rightarrow \mathbf{v}^{(1)}=\left(1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1\right)$$

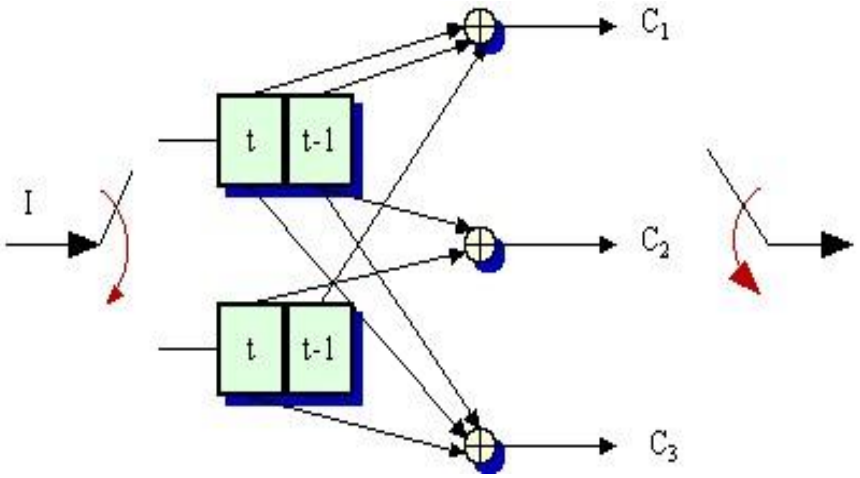
$$v^{(2)}(x)=1+x^4+x^5 \Rightarrow \mathbf{v}^{(2)}=\left(1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1\right)$$

与前面结论一致!

卷积码的编码

- 多项式生成矩阵
  - 例16.2:(续)

$$\mathbf{G}(x)=\begin{bmatrix}1+x & x & 1+x\\ x & 1 & 1\end{bmatrix}$$



卷积码的编码

- 多项式生成矩阵
  - 一般地，设$\mathbf{u}(x)=\mathbf{u}_0+\mathbf{u}_1x+\mathbf{u}_2x^2+\cdots$

$$=\left[u^{(1)}(x)\quad u^{(2)}(x)\quad\cdots\quad u^{(k)}(x)\right]^T$$

$$\mathbf{v}(x)=\mathbf{v}_0+\mathbf{v}_1x+\mathbf{v}_2x^2+\cdots$$

$$=\left[v^{(1)}(x)\quad v^{(2)}(x)\quad\cdots\quad v^{(n)}(x)\right]^T$$

则有$\mathbf{v}(x)=\mathbf{u}(x)\cdot\mathbf{G}(x)$

$$\mathbf{G}(x)=\left[g_i^{(j)}(x)\right]_{k\times n}\quad i=1,2,\cdots,k\quad j=1,2,\cdots,n$$

其中$g_i^{(j)}(x)$描述n个输出中第j个符号与k个输入中第i个信息的关系.

- 多项式生成矩阵

定义16.1: 在卷积码中, 若前k个输出序列就是重复k个输入序列, 则称为**系统卷积码**.

系统卷积码的多项式生成矩阵为:

$$\mathbf{G}(x)=\begin{bmatrix} 1 & 0 & \cdots & 0 & g_1^{(k+1)}(x) & g_1^{(k+2)}(x) & \cdots & g_1^{(n)}(x) \\ 0 & 1 & \cdots & 0 & g_2^{(k+1)}(x) & g_2^{(k+2)}(x) & \cdots & g_2^{(n)}(x) \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & g_k^{(k+1)}(x) & g_k^{(k+2)}(x) & \cdots & g_k^{(n)}(x) \end{bmatrix}$$

卷积码的编码

- 多项式生成矩阵
  - 多项式生成矩阵的八进制码表示法

例如：IMT-2000建议中，关于卷积码的建议为:

R	K	$g^{(1)}$	$g^{(2)}$	$g^{(3)}$	$g^{(4)}$
1/2	9	753	561	N/A	N/A
1/3	9	557	663	711	N/A
1/4	9	765	671	513	473

- 多项式生成矩阵
  - 多项式生成矩阵的八进制码表示法（续）

其中，（9, 1/2）卷积码的多项式为：

$(753)_8 \rightarrow (111101011)_2$

$\rightarrow g^{(1)}(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8$

$\rightarrow v^{(1)}(t) = u(t)+u(t-1)+u(t-2)+u(t-3)+u(t-5)$
$+u(t-7)+u(t-8)$，

$(561)_8 \rightarrow (101110001)_2$

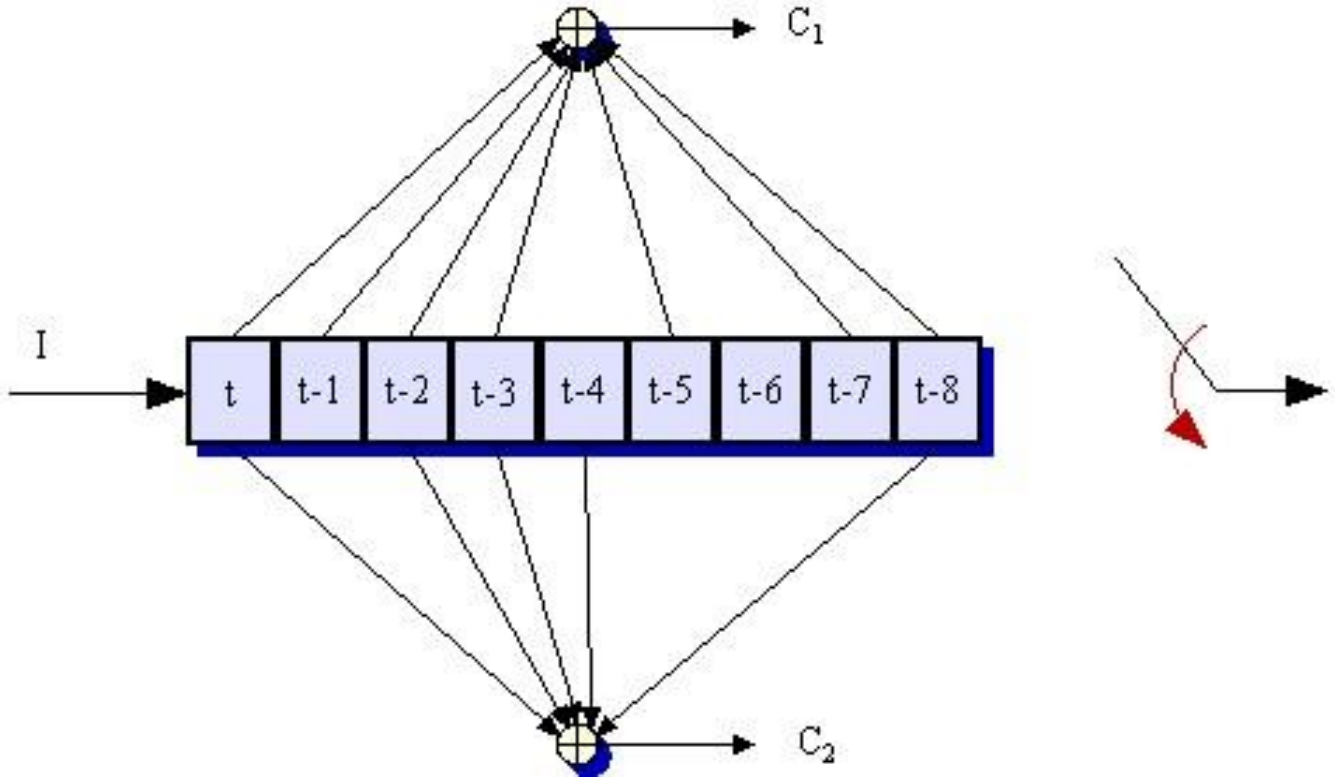
$\rightarrow g^{(2)}(x) = 1 + x^2 + x^3 + x^4 + x^8$

$\rightarrow v^{(2)}(t) = u(t)+u(t-2)+u(t-3)+u(t-4)+u(t-8)$.

卷积码的编码

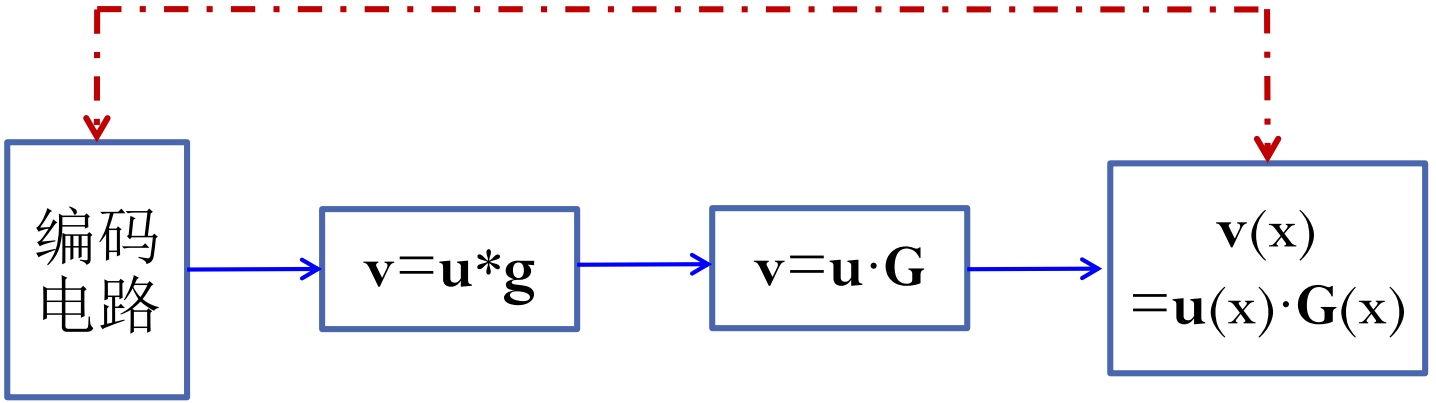
- 多项式生成矩阵
  - 多项式生成矩阵的八进制码表示法（续）

（9，1/2）卷积码的编码器为：



卷积码的编码

- 小结



g--编码电路
冲击响应

G----半无限
标量生成矩阵

G(x)----k×n维
多项式生成矩阵

目录

- 卷积码的编码
- 卷积码的图描述
  - ✓状态转换图
  - ✓网格图
  - ✓树图
- 距离分布多项式
- 性能分析及构造准则
- 卷积码的译码
- 打孔卷积码和咬尾卷积码

卷积码的图描述

- 状态转换图

基本要素

状态: 由卷积码编码器的移位寄存器内容确定.

设 (K, k/n)卷积码第i个移位寄存器含有B_i个以前的信息位.

定义 $B=\sum_{i=1}^{k} B_i$ 为编码器总存储量

则编码器共有2^B个不相同的可能状态.

激励: 由最新输入的k个比特表示.

响应: 相应输出.

更新状态: 由新输入引起寄存器状态的更新.

卷积码的图描述

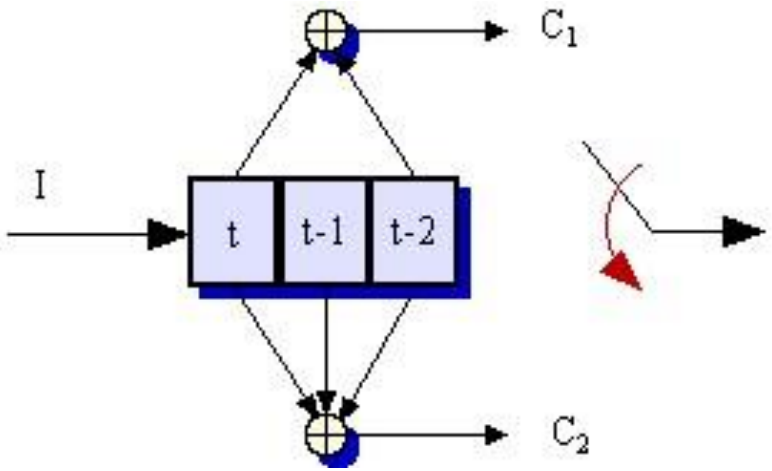
- 状态转换图

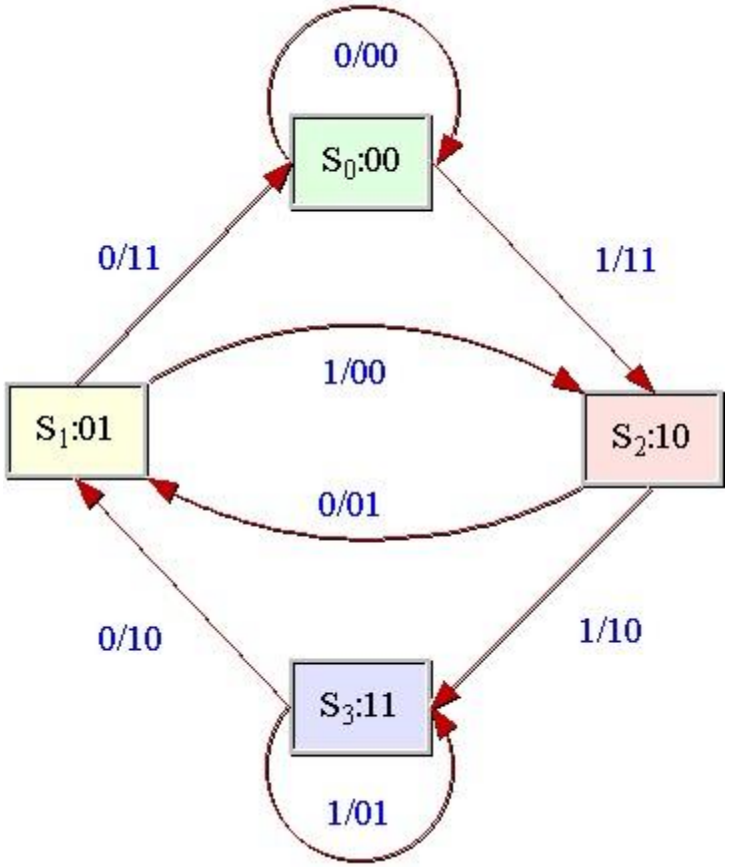
表示方法

- 图中以方块表示可能的状态，并加以注明
- 如果两个状态对应一对可能的状态变化，则把它们用箭头连起来
- 箭头的方向表示状态的变化方向
- 并在线旁注明编码的激励和相应

卷积码的图描述

- 状态转换图
  - 例16.1中(3，1/2)卷积码, 其状态转换图为：





卷积码的图描述

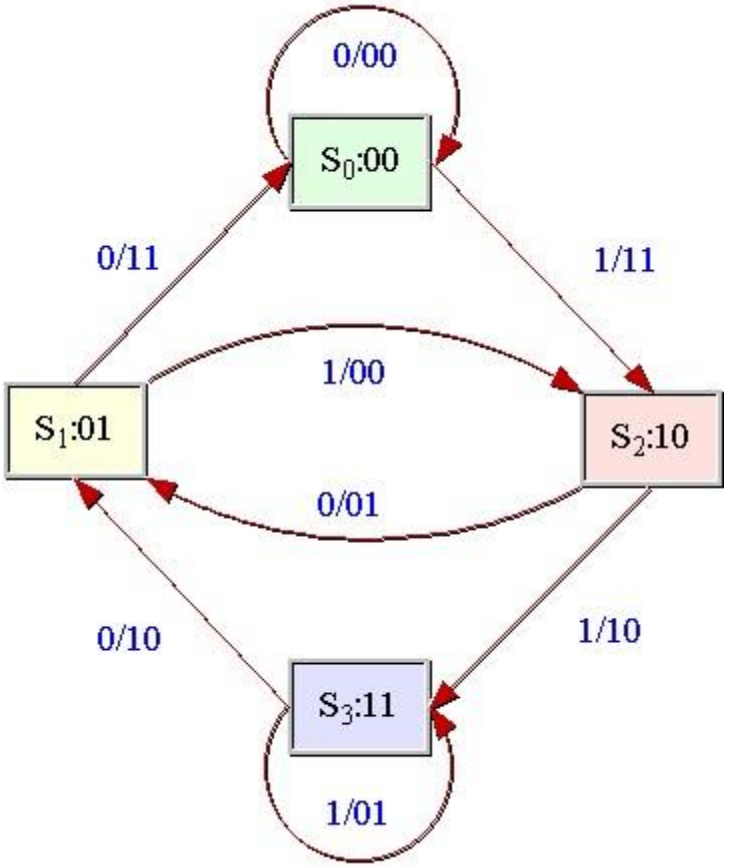
- 状态转换图

利用状态转换图编码：

- 假定编码器始于状态s_0(全零状态)
- 对应于任一给定信息序列的编码过程如下：
  - 沿着由信息序列所决定的路径，并记下每一分支上标示的相应输出
  - 到了最后给信息序列附加K-1组全零序列，使编码器转换到状态s_0

卷积码的图描述

- 状态转换图
  - 例16.1中(3，1/2)卷积码，利用状态图编码：



信息序列：1101

编码输出序列：

111010000111

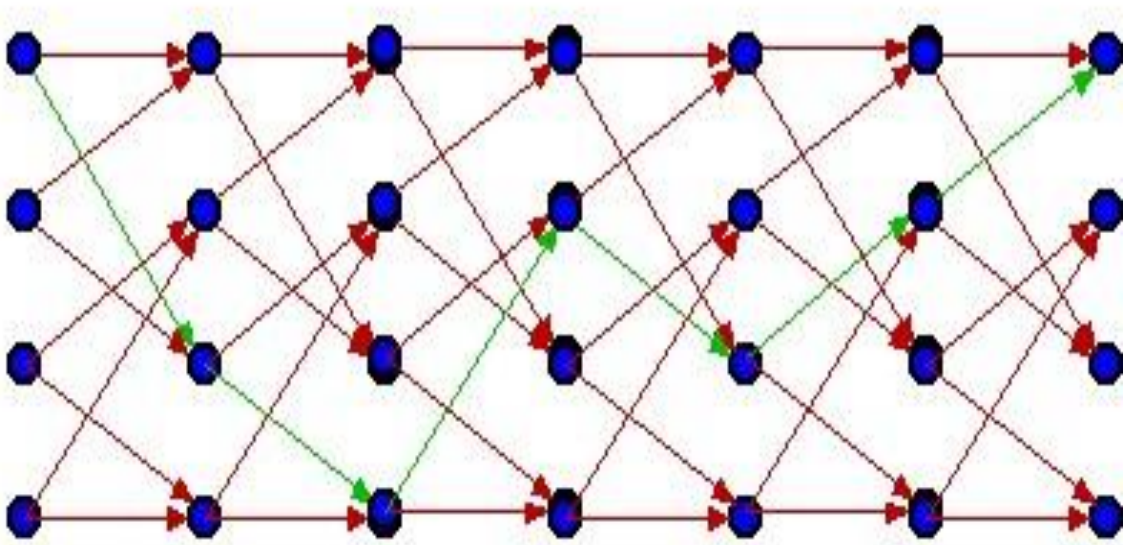
卷积码的图描述

- 网格图
  - 为了改善状态图的描述，记录时间的影响，可以把每一时刻的状态排成一列
  - 状态的变化在相邻两个时刻的状态间发生
  - 仍用连线、箭头和线旁的标注表示状态的变化、激励和输出

卷积码的图描述

- 网格图
  - 例16.1中的(3，1/2)卷积码, 其网格图如下:

激励	0	1	状态
响应	00	11	00
	11	00	01
	01	10	10
	10	01	11

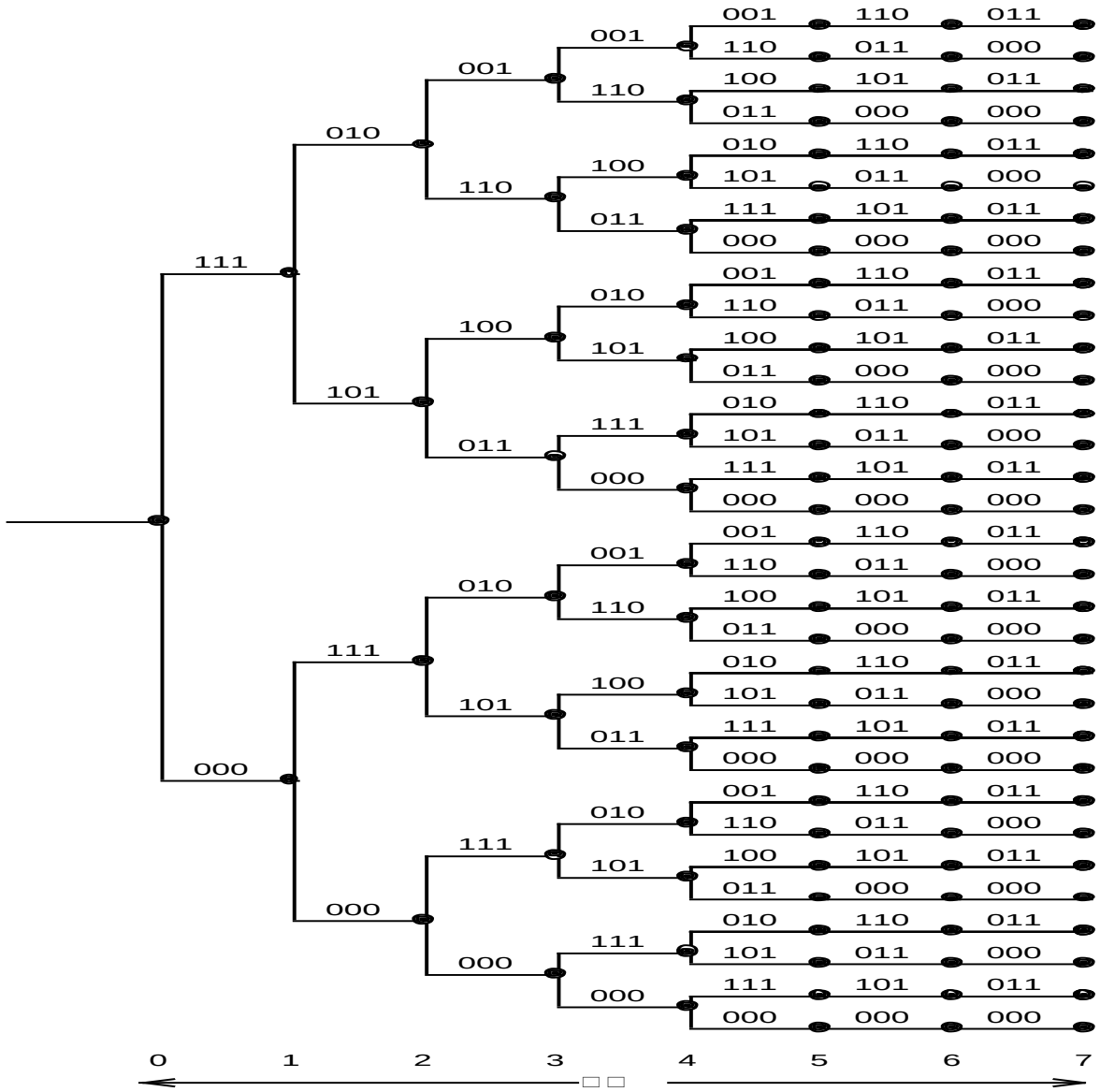


卷积码的图描述

- 树图
  - 例:设(3, 1/3)卷积码的多项式生成矩阵为

    $\mathbf{G}(x) = (1+x, 1+x^2, 1+x+x^2)$

    当信息序列长度$L = 5$时，其码树图如下：


111
010
001
001
110
110
011
100
101
011
101
100
010
110
111
000
000
111
010
001
001
110
110
011
100
101
011
101
100
010
110
111
000
000
111
010
001
001
110
110
011
100
101
011
101
100
010
110
111
000
000
111
010
001
001
110
110
011
100
101
011
101
100
010
110
111
000
000
0
1
2
3
4
5
6
7

卷积码的图描述

- 小结：

名称	基本要素	应用
状态转换图	状态 激励 响应 状态更新	描述编码路径 求距离分布多项式
网格图	状态 激励 响应 状态更新 时间	描述编码路径 Viterbi译码
树图	激励 响应 时间	描述编码路径 序列译码

目录

- 卷积码的编码
- 卷积码的图描述
- 距离分布多项式
- 性能分析及构造准则
- 卷积码的译码
- 打孔卷积码和咬尾卷积码

距离分布多项式

- 线性卷积码的性能主要取决于其输出序列在网格图上所对应路径之间的距离
- 对于线性码只要考虑全零路径与其它路径的距离
- 全零路径和其他路径的距离分布可以用**距离分布多项式**来表示
- **最小自由距离**:

$$d_{free} \equiv \min_{\mathbf{u}',\mathbf{u}''} \{ d(\mathbf{x}', \mathbf{x}'') : \mathbf{u}' \neq \mathbf{u}'' \}$$

其中**x**’和**x**”分别是信息序列**u**’和**u**”所对应的路径

距离分布多项式

$$A(D)=\sum_{d=0}^{n}A_d D^d$$

- 距离分布多项式：

$$T(D,L,I)=\sum_{d,l,i}A(d,l,i)D^d L^l I^i$$

其中A(d, l, i)表示

–重量(与全零路径距离)为d

–编码步数为l

–相应信息序列的重量为i

的路径个数

距离分布多项式

- 求距离分布多项式的方法

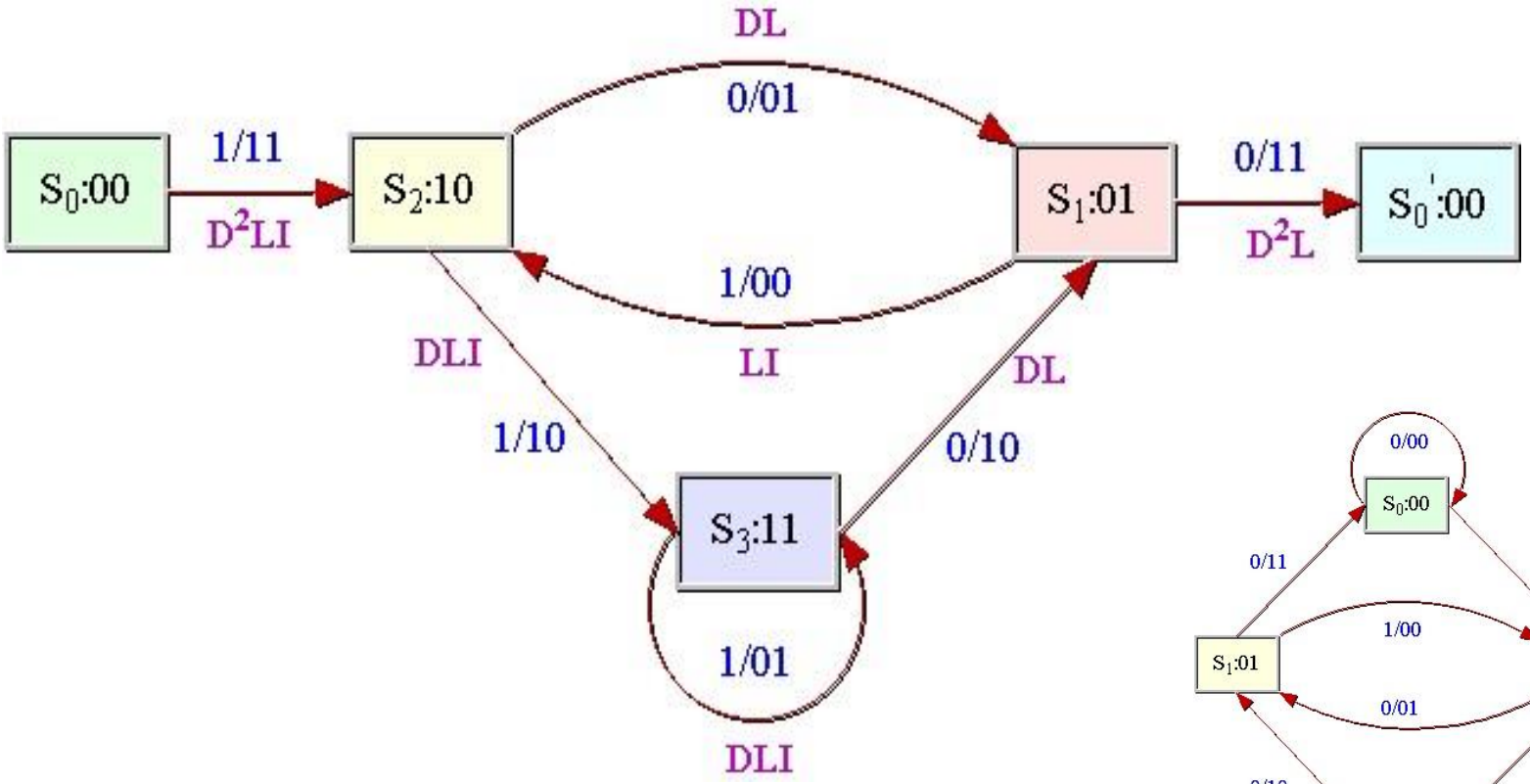
  将状态转换图修正为信号流图，就可以对所有非零码字的距离特性给出一个完整的描述

信号流图：

- 将全零状态s_0拆成起始状态和终止状态，删去绕状态s_0的自环
- 每一分支都用分支增益$D^d L I^i$ 标示，其中
  - ✓ d是此分支上n个编码比特的重量
  - ✓ l 表示进行一步编码
  - ✓ i 是此分支上k个信息比特的重量

距离分布多项式

- 求距离分布多项式的方法
  - 例16.1中的(3，1/2)卷积码, 其信号流图如下:



距离分布多项式

- 求距离分布多项式的方法
  - 信号流图中连接起始状态和终止状态的每一条路径都代表了一个非零码字
  - 路径增益是沿此路径的分支增益之积
    - 路径增益中D的幂次是相应码字的重量
    - 路径增益中L的幂次是编码步数
    - 路径增益中I的幂次是相应信息序列的重量
  - T(D, L, I)是从输入至输出的总增益

距离分布多项式

- 求距离分布多项式的方法

  求解输入到输出的增益主要有以下两种方法:

  (1) 梅西公式

  (2) 解方程组

距离分布多项式

- 求距离分布多项式的方法

(1) 梅西公式

前向路径：在信号流图中，一个连结初始状态和终止状态，且任何状态仅通过一次的路径称作前向路径.

令F_i是第i条前向路径的增益.

环路：从任一状态出发并回到该状态，而其它状态仅通过一次的闭合路径称作环路.

令C_i是第i个环路的增益.

不相连的环路：在一组环路中，若没有一个状态是属于一个以上的环路时，就称此组环路不相连.

距离分布多项式

- 求距离分布多项式的方法

(1) 梅西公式（续）

令 {i}是所有环路的集合,

{i', j'}是所有不相连的环路对集合,

{i", j", l"}是所有三重不相连的环路集合,

....

$$\text{定义}\Delta = 1 - \sum_{i} C_i + \sum_{i',j'} C_{i'} C_{j'} - \sum_{i'',j'',l''} C_{i''} C_{j''} C_{l''} + \cdots$$

Δ_i的定义与Δ相同, 但只是针对与第i个前向路径不相连的部分来定义.

距离分布多项式

- 求距离分布多项式的方法

(1) 梅西公式（续）

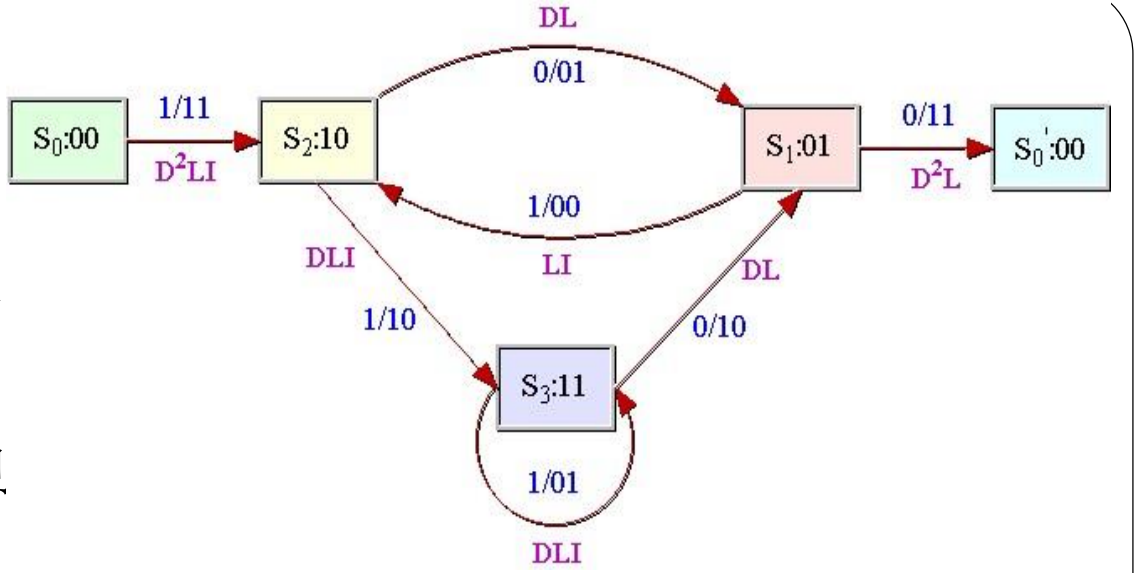
输入到输出增益可以用梅西公式表示:

$$T(D,L,I)=\frac{\sum f_i\Delta_i}{\Delta}$$

- 求距离分布多项式

  (1) 梅西公式（续）

  利用梅西公式求例



前向路径1: $s_0 \rightarrow s_2 \rightarrow s_1 \rightarrow s_0'$ $f_1=D^5L^3I$

前向路径2: $s_0 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1 \rightarrow s_0'$ $f_2= D^6L^4I^2$

环路1: $s_2 \rightarrow s_1 \rightarrow s_2$ $C_1=DL^2I$

环路2: $s_2 \rightarrow s_3 \rightarrow s_1 \rightarrow s_2$ $C_2=D^2L^3I^2$

环路3: $s_3 \rightarrow s_3$ $C_3=DLI$

距离分布多项式

- 求距离分布多项式的方法

(1) 梅西公式（续）

利用梅西公式求例16.1中（3，1/2）卷积码的

$$T(D,L,I)=\frac{\sum f_i\Delta_i}{\Delta}$$

$$=\frac{D^5L^3I(1-DLI)+D^6L^4I^2}{1-(DLI+DL^2I+D^2L^3I^3)+DLI\cdot DL^2I}$$

$$=\frac{D^5L^3I}{1-(DLI+DL^2I)}$$

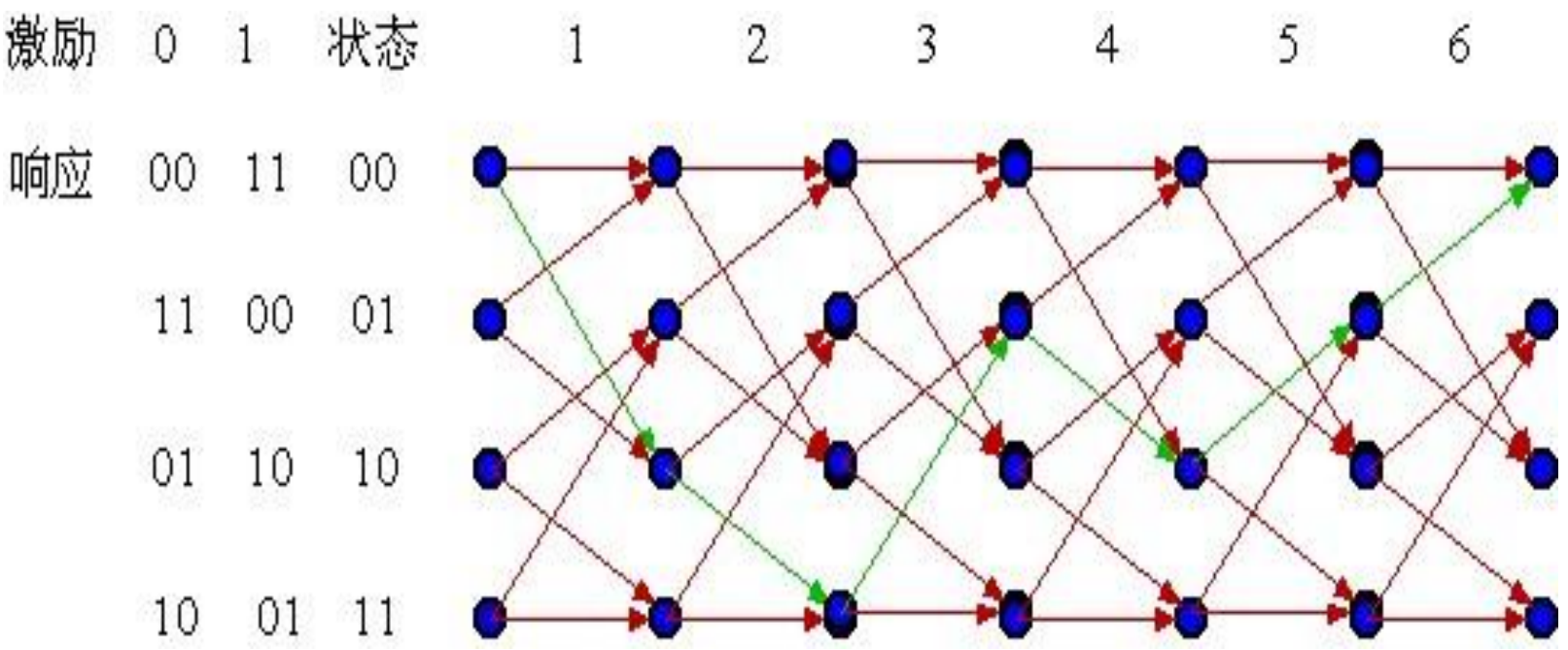
$$=D^5L^3I+D^6L^4I^2+D^6L^5I^2+\cdots$$

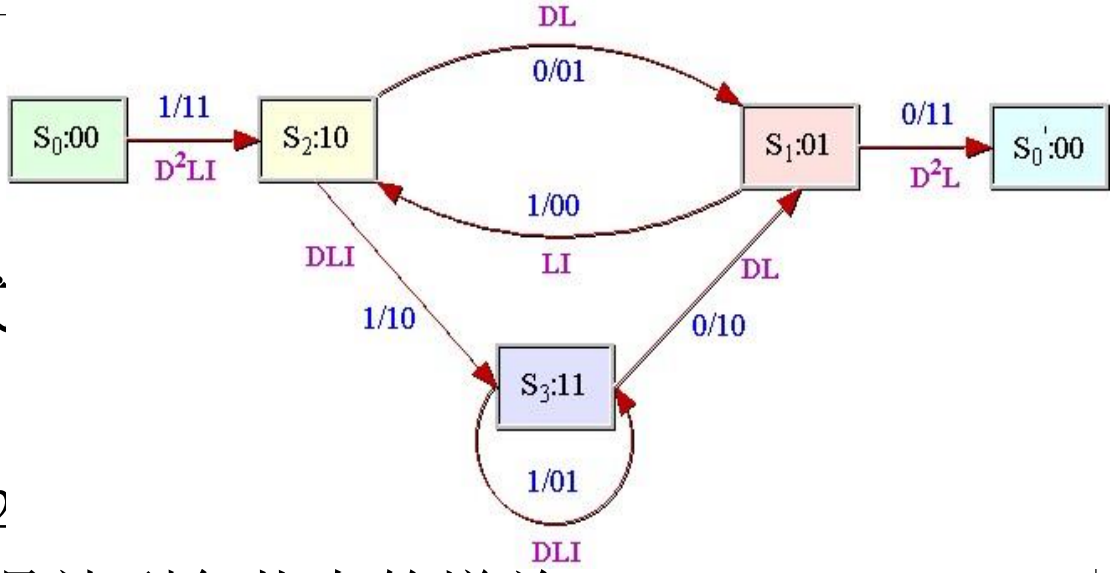
距离分布多项式

- 求距离分布多项式的方法

  (1) 梅西公式（续）

$$T(D,L,I)= D^5L^3I + D^6L^4I^2 + D^6L^5I^2 + \cdots$$





- 求距离分布多项式

(2) 解方程组

求例16.1中（3，1/2

设ξ_0，ξ_1，ξ_2，ξ_3，ξ_0'是达到各状态的增益.

$$\begin{cases} \xi_2 = \xi_0 \cdot D^2LI + \xi_1 \cdot LI \\ \xi_1 = \xi_2 \cdot DL + \xi_3 \cdot DL \\ \xi_3 = \xi_2 \cdot DLI + \xi_3 \cdot DLI \\ \xi_0' = \xi_1 \cdot D^2L \end{cases}$$

解方程, 求得增益 $\dfrac{\xi_0'}{\xi_0} = T(D,L,I)$

目录

- 卷积码的编码
- 卷积码的图描述
- 距离分布多项式
- 性能分析及构造准则
- 卷积码的译码
- 打孔卷积码和咬尾卷积码

- 利用距离多项式计算错误概率

(1) 设某一路径和全零路径的距离为d, 从全零路径错到这条路径的概率为：

$$P_e(m \to m') \le z^d$$

其中 $z = \sum_y \sqrt{p(y/0)\,p(y/1)}$

对于二进制对称信道 $z = 2\sqrt{\varepsilon(1-\varepsilon)}$

(2) 假设这条路径有i个信息位为1，相应的误比特率为：

$$P_b(m \to m') \le \frac{iz^d}{k}$$

- 利用距离多项式计算错误概率（续）

(3) 所有可能的错误事件的并合界为：

$$P_b \le \sum A(d,l,i)(D^d L^l I^i)\frac{i}{k}\Bigg|_{\substack{D=z\\L=1\\I=1}}$$

$$= \frac{1}{k}\sum \frac{\partial}{\partial I} A(d,l,i)(D^d L^l I^i)\Bigg|_{\substack{D=z\\L=1\\I=1}}$$

$$= \frac{1}{k}\frac{\partial}{\partial I}[T(D,L,I)]\Bigg|_{\substack{D=z\\L=1\\I=1}}$$

- 渐进编码增益
  - **A**symptotic **C**oding **G**ain
  - 相同误比特率条件下,**编码**与**不编码**所需信噪比的差值.
  - 硬判决条件下:

$$\gamma \sim 10\log_{10}\left(\frac{Rd_{free}}{2}\right)dB$$

软判决条件下:

$$\gamma \sim 10\log_{10}\left(Rd_{free}\right)dB$$

构造准则

- 构造准则一: **最大化**最小自由距离d_{free}
  - 在决定高SNR时的性能起重要作用
- 构造准则二:**最小化**重量为d_{free}的路径数量
  - SNR较低时影响显著
- 构造方法:
  - 大部分卷积码的构造通过计算机搜索完成
  - 约束长度相对短(K<20)的情况下,存在基于Viterbi算法的有效搜索方法

构造准则

- R=1/2的最优卷积码

K	$g^{(0)}$	$g^{(1)}$	d_{free}	A_{dfree}	γ(dB)
2	3	1	3	1	1.76
3	5	7	5	1	3.98
4	13	17	6	1	4.77
5	27	31	7	2	5.44
6	53	75	8	1	6.02
7	117	155	10	11	6.99
8	247	371	10	1	6.99
9	561	753	12	11	7.78
10	1131	1537	12	1	7.78
11	2473	3217	14	14	8.45
12	4325	6747	15	14	8.75

目录

- 卷积码的编码
- 卷积码的图描述
- 距离分布多项式
- 性能分析及构造准则
- 卷积码的译码
- 打孔卷积码和咬尾卷积码

卷积码的译码

- Viterbi译码
  - Viterbi在1967年提出
  - 基于最大似然译码原理
  - 在误码率为10^{-6}—10^{-3}范围内，**软判决**Viterbi译码算法比**硬判决**Viterbi译码算法的编码增益高 2.0dB以上
  - 对于约束长度不超过15时，Viterbi译码算法易实现

例：选 (7, 3)极长码

$n = 2^3 - 1 = 7$

$k = m = 3$

$d_{min} = 2^{m-1} = 4$

$\mathbf{x}_1 = (0\ 0\ 0\ 0\ 0\ 0\ 0)$

$\mathbf{x}_2 = (1\ 1\ 1\ 0\ 1\ 0\ 0)$

$\mathbf{x}_3 = (0\ 1\ 1\ 1\ 0\ 1\ 0)$

$\mathbf{x}_4 = (0\ 0\ 1\ 1\ 1\ 0\ 1)$

$\mathbf{x}_5 = (1\ 0\ 0\ 1\ 1\ 1\ 0)$

$\mathbf{x}_6 = (0\ 1\ 0\ 0\ 1\ 1\ 1)$

$\mathbf{x}_7 = (1\ 0\ 1\ 0\ 0\ 1\ 1)$

$\mathbf{x}_8 = (1\ 1\ 0\ 1\ 0\ 0\ 1)$

例：

发码：$\mathbf{x}_m = (x_{m0}, x_{m1}, \cdots, x_{m6})$　　$x_{mi} \in \{0, 1\}$

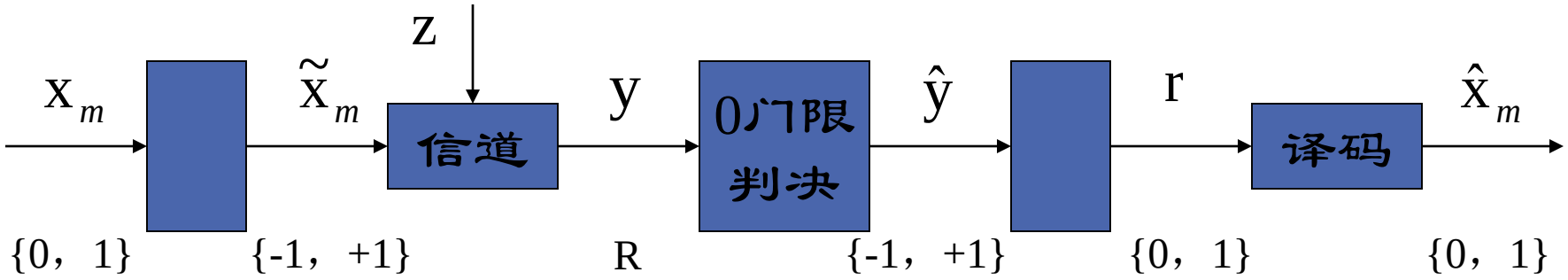
实际发送：$\tilde{\mathbf{x}}_m = (\tilde{x}_{m0}, \tilde{x}_{m1}, \cdots, \tilde{x}_{m6})$　　$\tilde{x}_{mi} \in \{-1, +1\}$

干扰：$\mathbf{z} = (z_0, z_1, \cdots, z_6)$　　$z_i \in R$

接收：$\mathbf{y} = \tilde{\mathbf{x}}_m + \mathbf{z}$　　$y_i \in R$

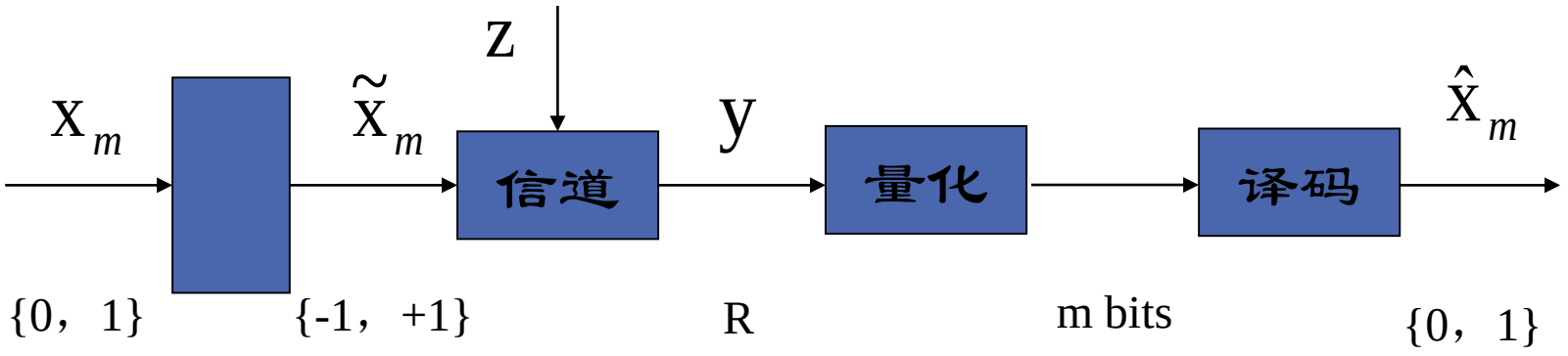
例：

硬判决： 先对 **y**进行门限判决，
得到 $\mathbf{r} = \{r_0, r_1, \ldots, r_6\}$ $r_i \in \{0, 1\}$，
再对 **r**进行译码.



例：

软判决： 直接对 $\mathbf{y} = \{y_0, y_1, \cdots, y_6\}, y_i \in R$ 进行译码.



例：

假如发送　$\mathbf{x}_4 = (0\ 0\ 1\ 1\ 1\ 0\ 1)$

实际发送　$\tilde{\mathbf{x}}_4 = (+1, +1, -1, -1, -1, +1, -1)$

接收　$\mathbf{y} = (+1.1, -0.1, +0.1, -0.9, -1.1, +0.9, -1.1)$

例：

硬判决译码：

门限判决得 $\hat{\mathbf{y}} = (+1, -1, +1, -1, -1, +1, -1)$

得到 $\mathbf{r} = (0\ 1\ 0\ 1\ 1\ 0\ 1)$

比较$\mathbf{r}$与 $\mathbf{x}_1 \sim \mathbf{x}_8$的距离，

认为距离最小的$\mathbf{x}_i$ (i = 1~8)为发码.

但 $d(\mathbf{x}_8) = d(\mathbf{x}_4) = d(\mathbf{x}_6) = 2$, 不易判决.

例：

软判决译码： 计算 $\langle \mathbf{y} \cdot \tilde{\mathbf{x}}_m \rangle$

$$\tilde{\mathbf{x}}_8 = (-1,-1,+1,-1,+1,+1,-1) \qquad \langle \mathbf{y} \cdot \tilde{\mathbf{x}}_8 \rangle = 0.9$$

$$\tilde{\mathbf{x}}_4 = (+1,+1,-1,-1,-1,+1,-1) \qquad \langle \mathbf{y} \cdot \tilde{\mathbf{x}}_4 \rangle = 4.9$$

$$\tilde{\mathbf{x}}_6 = (+1,-1,+1,+1,-1,-1,-1) \qquad \langle \mathbf{y} \cdot \tilde{\mathbf{x}}_6 \rangle = 1.7$$

判为$\mathbf{x}_4$.

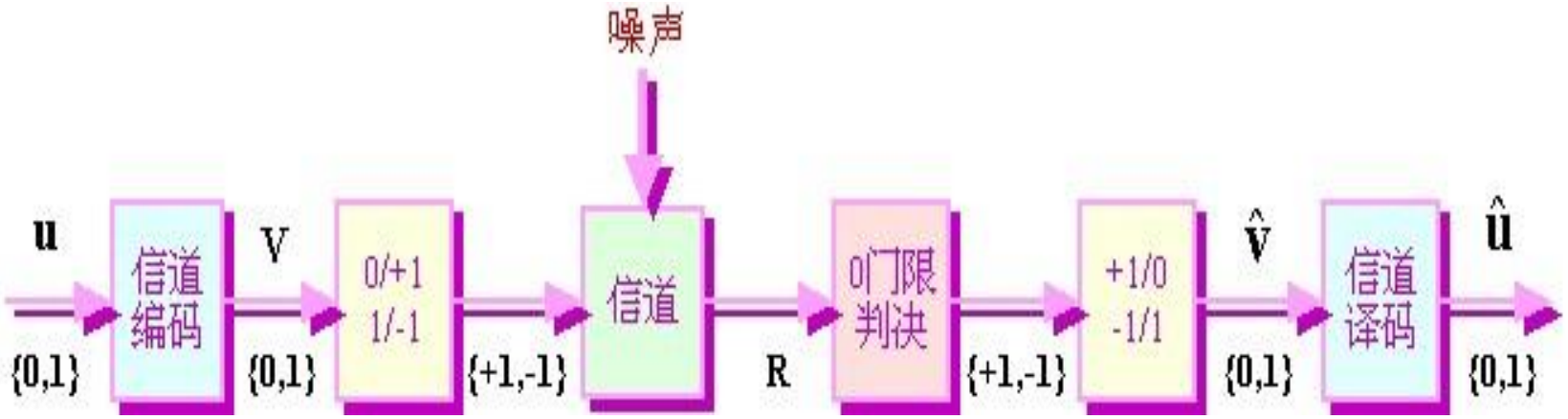
- 在相同误码率条件下，高斯信道中软判决译码比硬判决译码的信噪比低2~3dB
- 实际采用软判决译码时，还需进行量化，一般取Q=8或Q=16
- 量化级数越高，性能越好，但同时计算复杂度也越高

卷积码的译码

- Viterbi译码

(1) 硬判决Viterbi译码

译码准则: 最小汉明距离译码准则

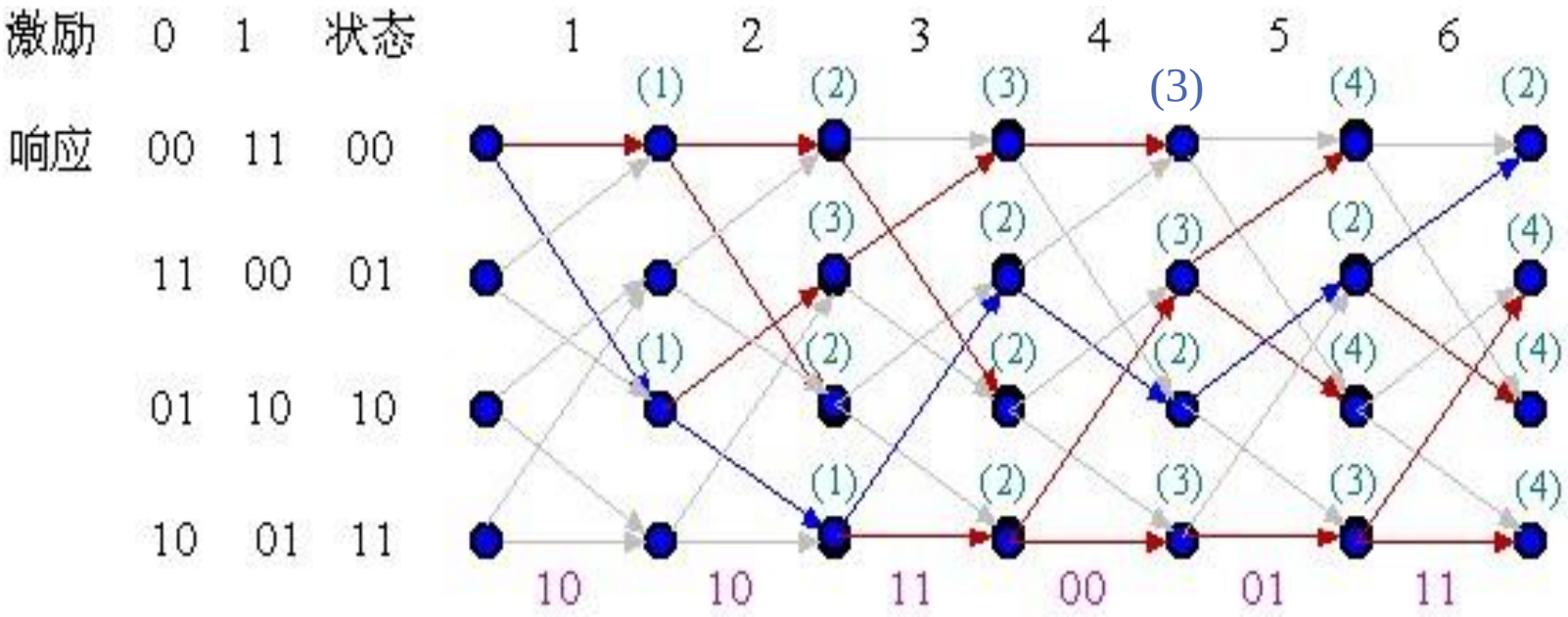


卷积码的译码

- Viterbi译码

(1) 硬判决Viterbi译码（续）

例16.1中的(3, 1/2)卷积码, 设收码**R** = (10 10 11 00 01 11)，其硬判决Viterbi译码过程如下：

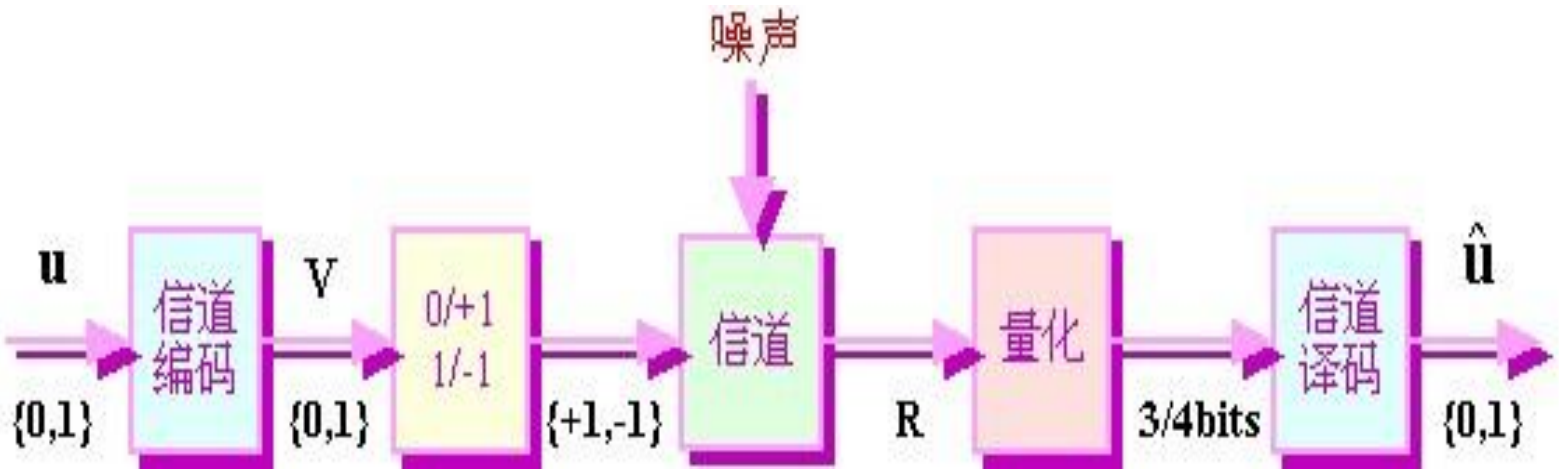


卷积码的译码

- Viterbi译码

(2) 软判决的Viterbi译码

译码准则: 最大相关译码准则

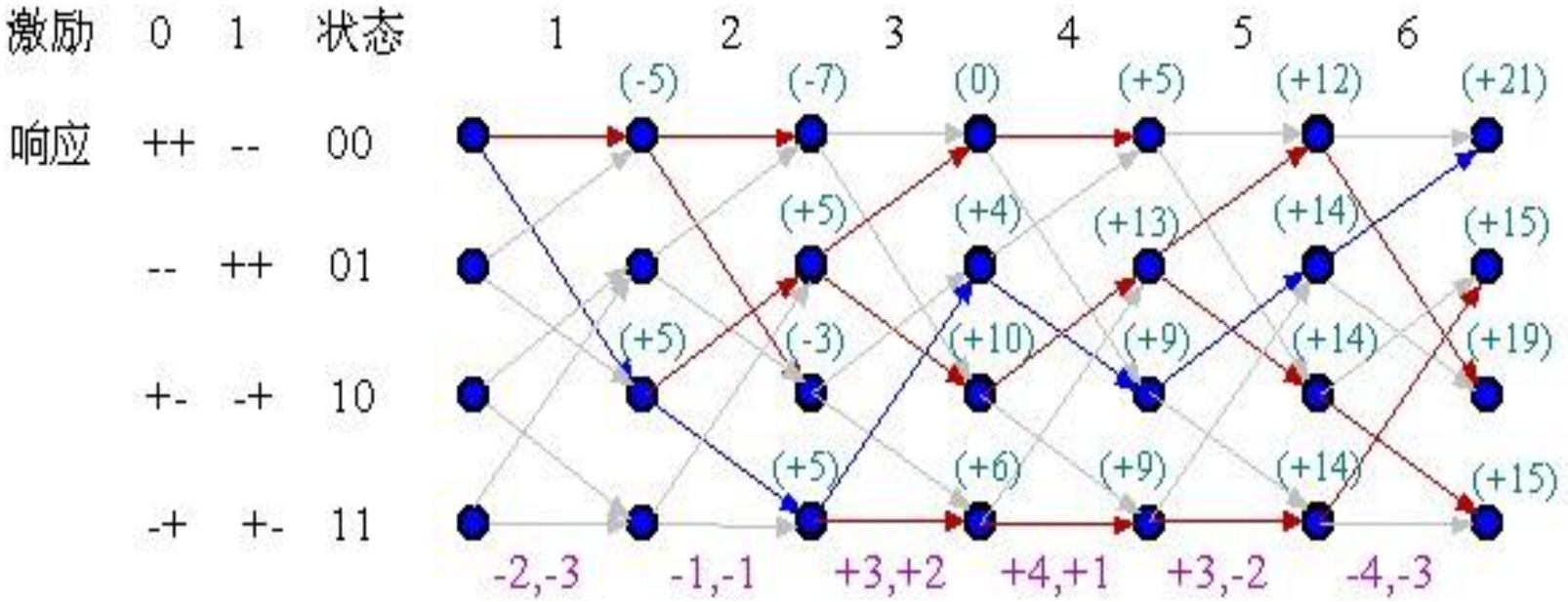


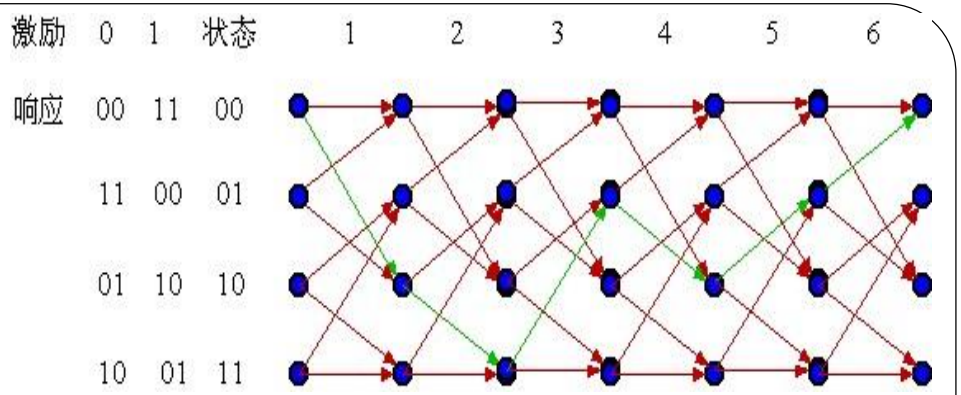
卷积码的译码

- Viterbi译码

(2) 软判决的Viterbi译码（续）

例16.1中的(3, 1/2)卷积码, 设收码**R** = (-2, -3; -1, -1; +3, +2; +4, +1; +3, -2; -4, -3)，其软判决Viterbi译码过程如下：





- Viterbi译码

(3)计算量分析

- (K, k/n)卷积码编码器共有$2^{k(K-1)}$个状态，
  即网格图上每个时刻对应$2^{k(K-1)}$个节点，
  而每个节点共有2^{k}个分支汇合
- 每一步译码需计算$2^{k(K-1)}\ 2^{k}=2^{kK}$个测度，
  存储$2^{k(K-1)}$个测度和相应的路径，
  其中每条路径的长度为 5kKbit
  （设译码存储路径长度τ=5K）
- Viterbi译码适用于小约束长度卷积码(K≤15)

卷积码的译码

- 序列译码
  - Wozencraft在1957年提出
  - 基于最大似然译码原理
  - 主要算法有Fano算法和堆栈算法
  - 对于约束长度较长（K>15）的卷积码，一般采用此算法

卷积码的译码

- 序列译码

–网格图和树图上第i条**路径**从第一分支到第B**分支**的量度为:

$$CM^{(i)} = \sum_{j=1}^{B}\sum_{m=1}^{n}\mu_{jm}^{(i)}$$

路径测度

分支测度

比特测度

–对于序列译码，定义

$$\mu_{jm}^{(i)} = \log_2 \frac{p(r_{jm}/c_{jm}^{(i)})}{p(r_{jm})} - \Re$$

其中$\Re$是一个正的常数，为路径选择指示

确保：若路径选择正确，路径量度将不断增加

若路径选择错误，路径量度将不断减少

卷积码的译码

- 序列译码
  - –对于二进制对称信道(BSC), 可选择

$$\mu_{jm}^{(i)} = \begin{cases} \log_2[2(1-p)] - R_c & if \quad r_{jm} = c_{jm}^{(i)} \\ \log_2 2p - R_c & if \quad r_{jm} \neq c_{jm}^{(i)} \end{cases}$$

比特测度

其中R_c = k/n为卷积码编码速率

卷积码的译码

- 序列译码
  - 例如在传输误码率为p = 0.1的二进制对称信道中传输R=1/3的卷积码，则

$$\mu_{jm}^{(i)} = \begin{cases} 0.52 & if \quad r_{jm} = c_{jm}^{(i)} \\ -2.65 & if \quad r_{jm} \neq c_{jm}^{(i)} \end{cases} \qquad d = 0$$

比特测度

  - 为简化计算，可设

$$\mu_{jm}^{(i)} = \begin{cases} 1 & if \quad r_{jm} = c_{jm}^{(i)} \\ -5 & if \quad r_{jm} \neq c_{jm}^{(i)} \end{cases}$$

比特测度

  - 或者表示为

$$\mu_j^{(i)} = 3 - 6d$$

$$\mu_j^{(i)} = \begin{cases} +3 & \text{if } d = 0 \\ -3 & \text{if } d = 1 \\ -9 & \text{if } d = 2 \\ -15 & \text{if } d = 3 \end{cases}$$

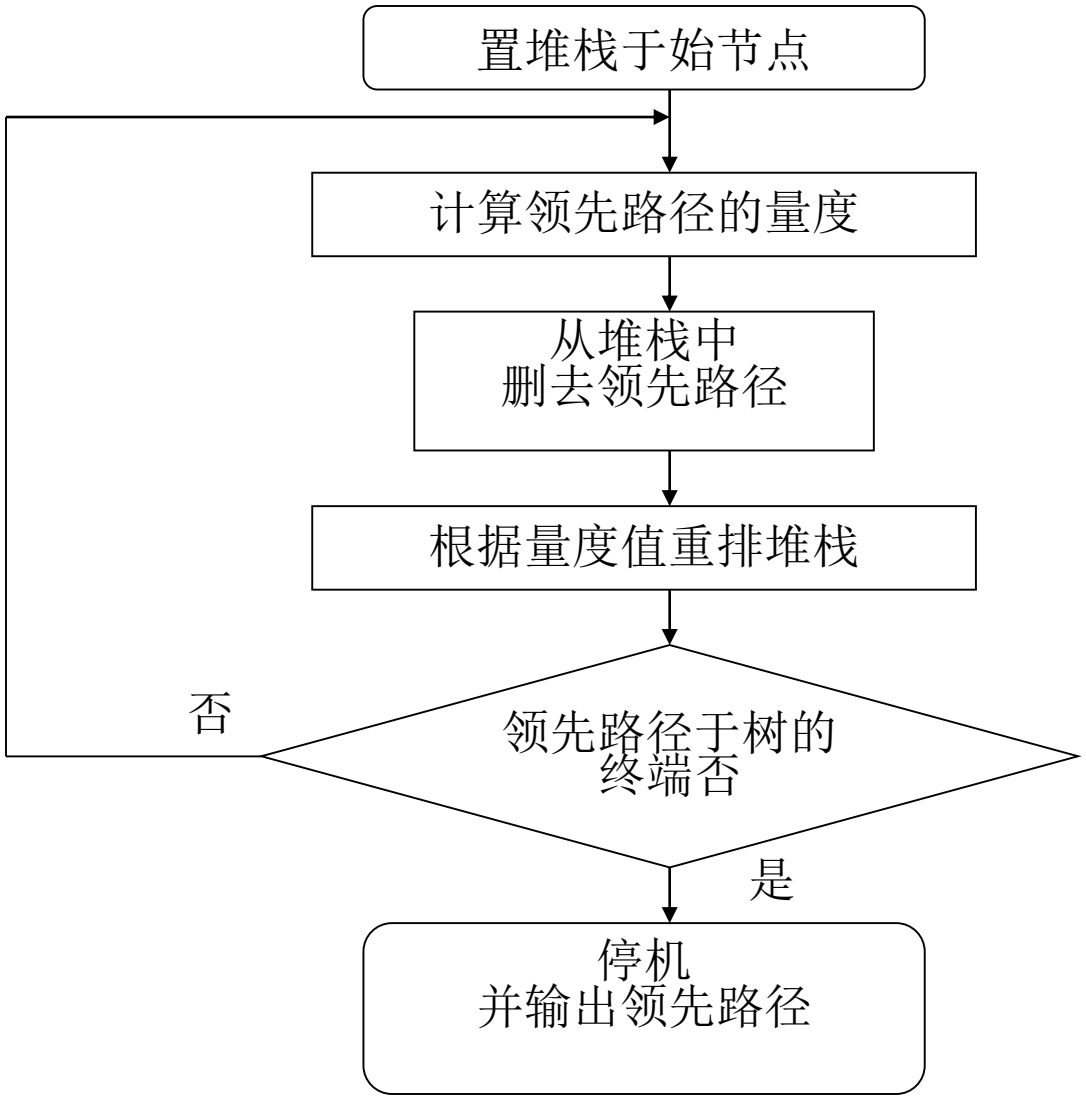
分支测度

卷积码的译码

- 序列译码

(1)堆栈译码算法

堆栈算法流程图



卷积码的译码

- 序列译码

  (1)堆栈译码算法

  例：设(3, 1/3)卷积码的多项式生成矩阵为

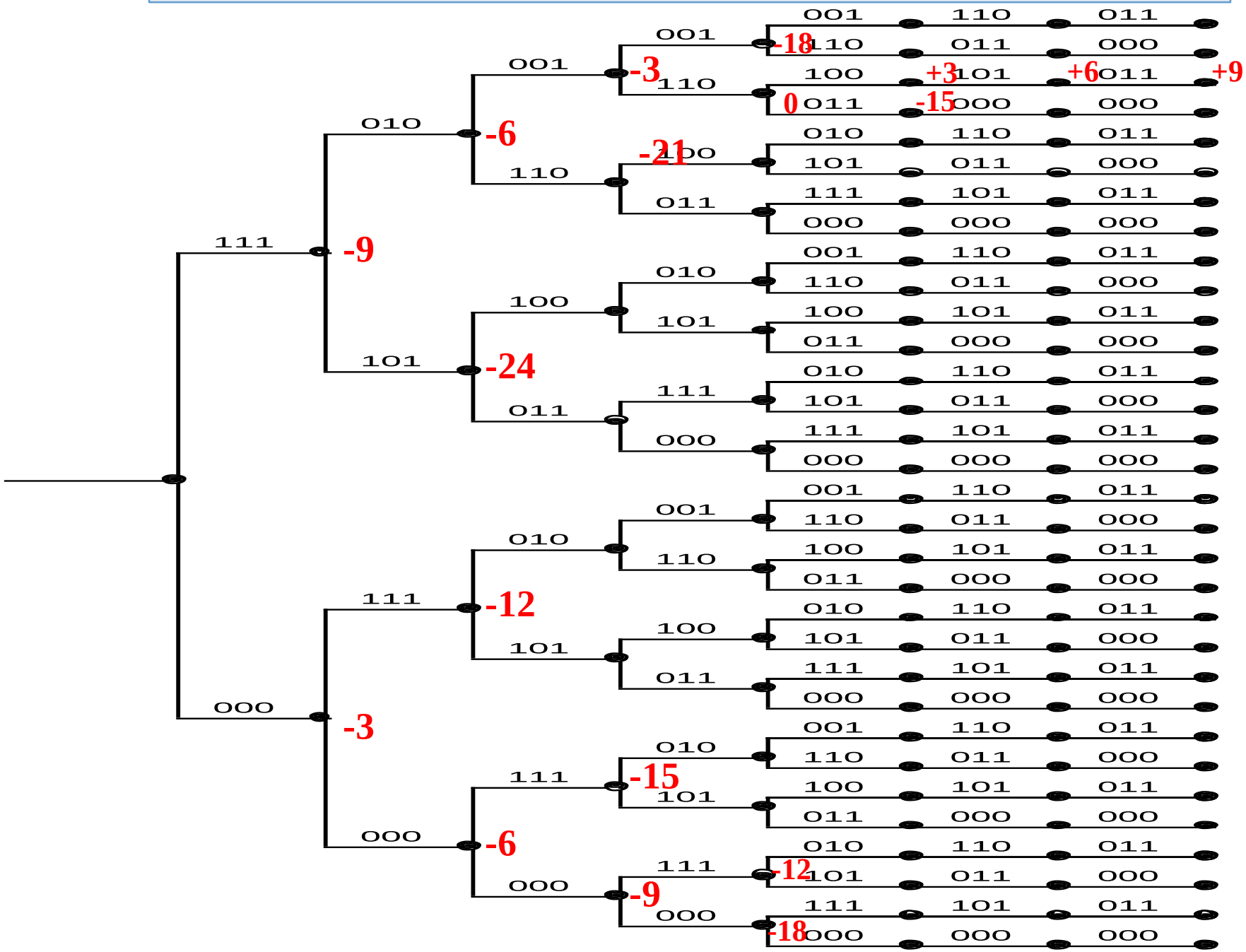
  $$\mathbf{G}(x) = (1+x, 1+x^2, 1+x+x^2),$$

  如果在传输误码率为p=0.1的BSC中传输此卷积码的一个码字，

  设收码为$\mathbf{r}$ = (010 010 001 110 100 101 011)

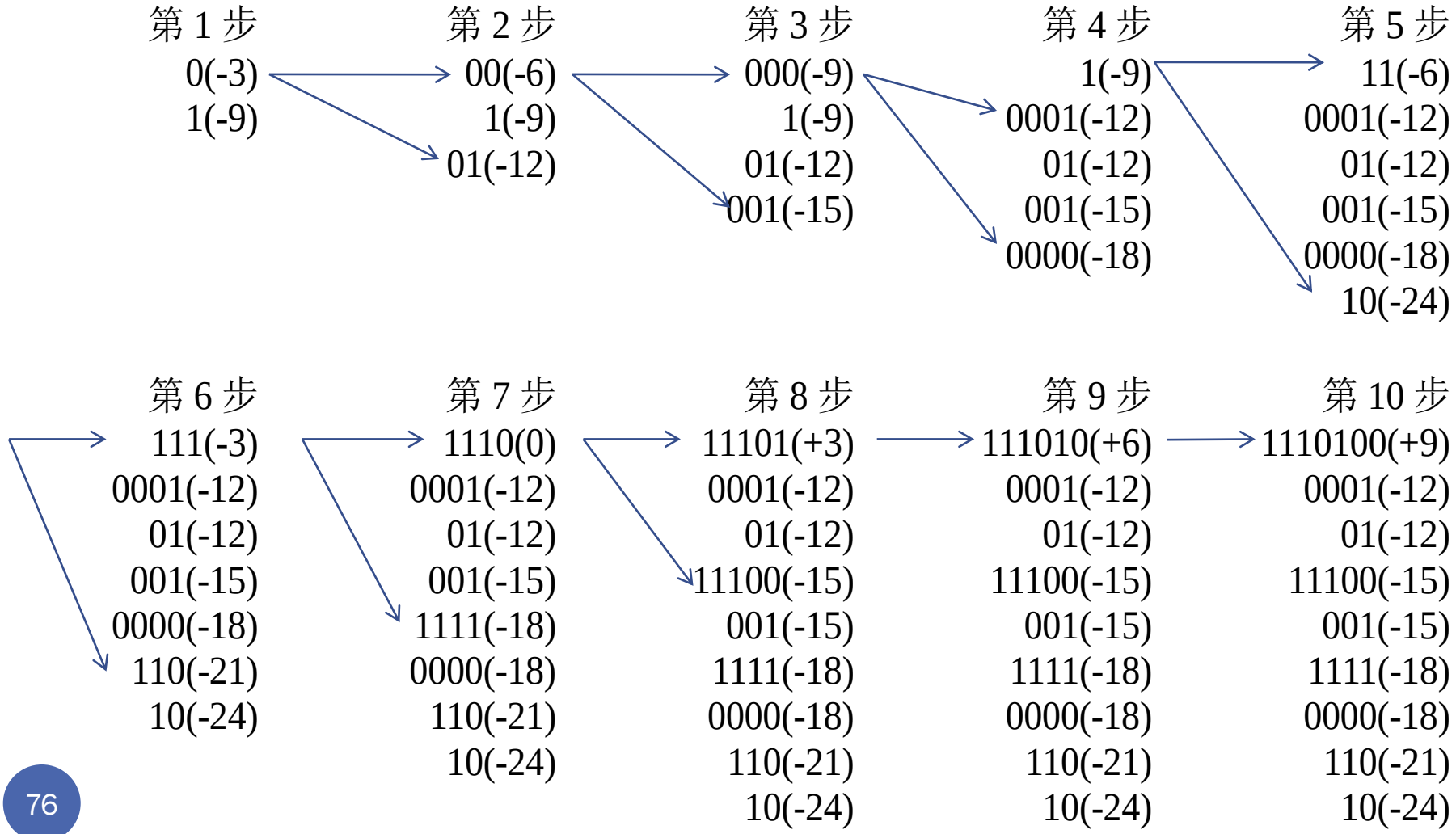
  则需经过10步译码，译码结果是将接收序列中的划线部分纠正

r = (010 010 001 110 100 101 011)



卷积码的译码

表 1：采用堆栈算法时存储器的存数



卷积码的译码

- 序列译码

  (1)堆栈译码算法

  例:(续)如果收码为**r**‘= (110 110 110 111 010 101 101)，

  则需经过20步译码，译码结果是将收码**r**‘中的划线部分纠正

  - 可见，堆栈算法的计算量与信道的质量有关，信道的传输误码率越低，则计算量越少

卷积码的译码

- 序列译码

  (1)堆栈译码算法

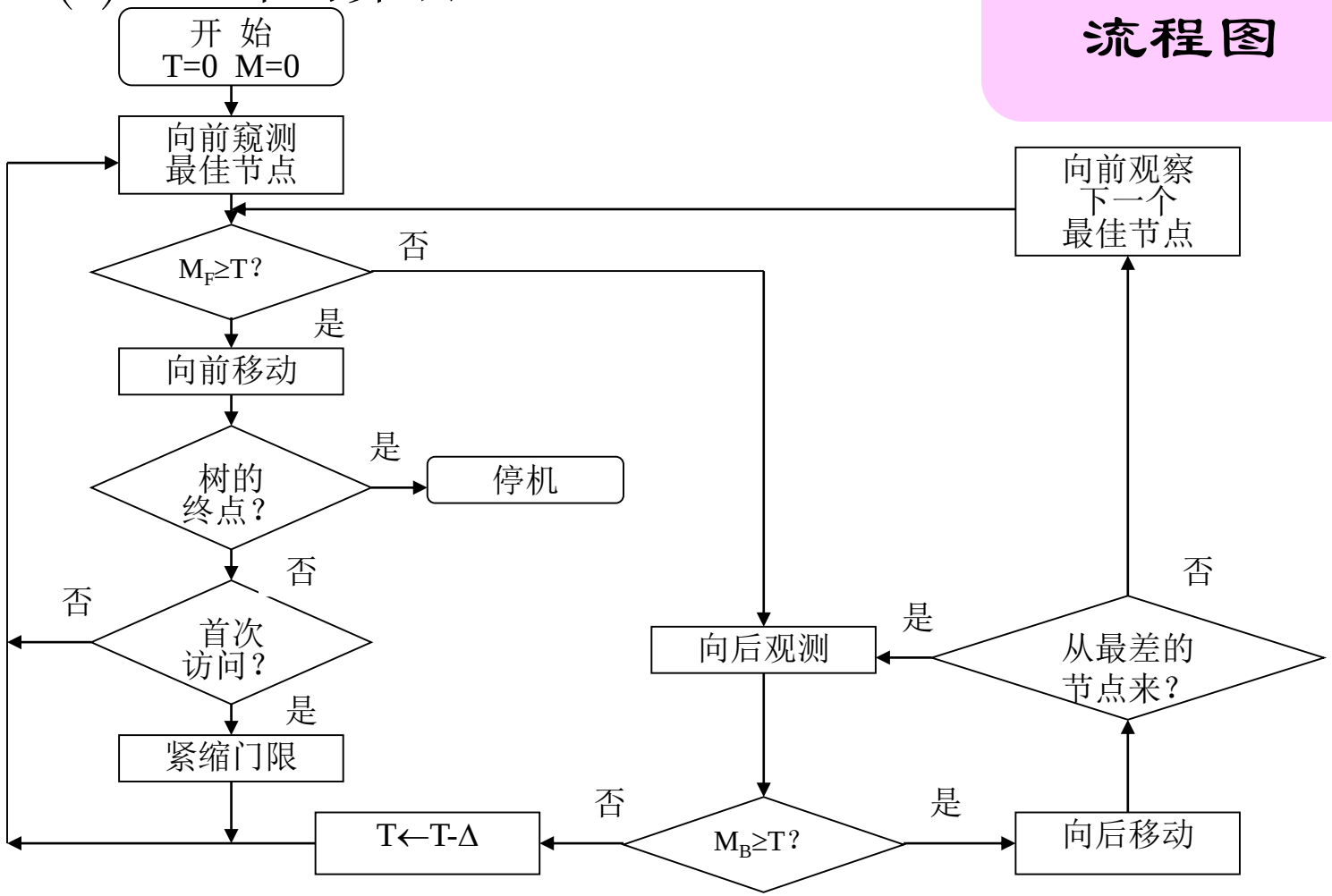
  比较：

  - 与Viterbi译码算法相比，其量度计算量少，但每步译码的重新排序使时延加大
  - 而与Fano算法相比，堆栈算法的算法简单，但存储量大

- 序列译码

  (2) Fano译码算法

Fano算法流程图



卷积码的译码

- 序列译码

  (2) Fano译码算法

  前面例题采用Fano算法的译码过程如表2

  此时门限增量Δ=3，需经过22步译码

  如果门限增量Δ=1，则需经过40步译码

LFB 表示“向前窥视最佳节点”，
LFNB 表示“向前窥视下一个最佳节点”
X　表示原点
原点的M_B为-∞

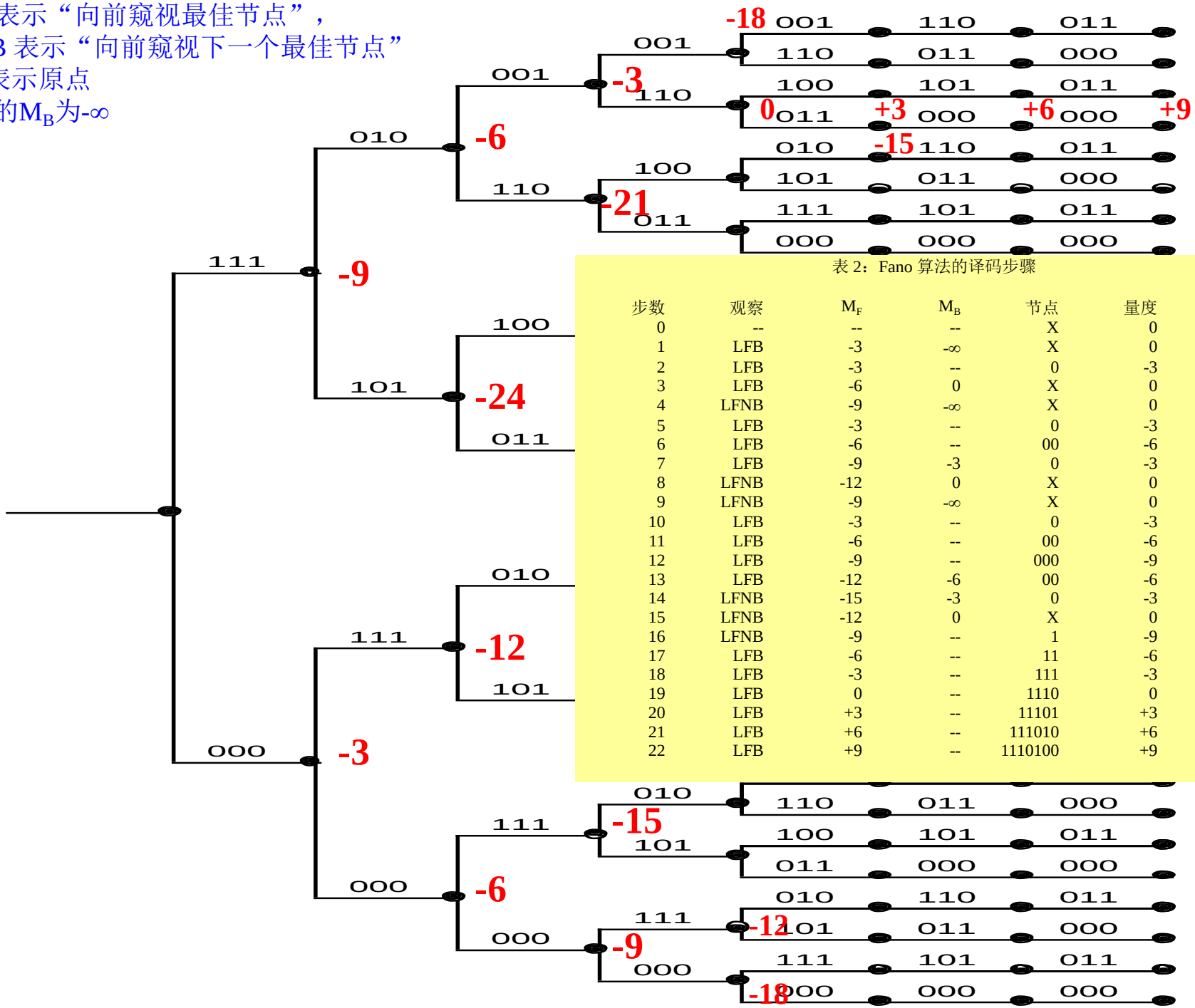


表 2：Fano 算法的译码步骤

步数	观察	M_F	M_B	节点	量度	T
0	--	--	--	X	0	0
1	LFB	-3	-∞	X	0	-3
2	LFB	-3	--	0	-3	-3
3	LFB	-6	0	X	0	-3
4	LFNB	-9	-∞	X	0	-6
5	LFB	-3	--	0	-3	-6
6	LFB	-6	--	00	-6	-6
7	LFB	-9	-3	0	-3	-6
8	LFNB	-12	0	X	0	-6
9	LFNB	-9	-∞	X	0	-9
10	LFB	-3	--	0	-3	-9
11	LFB	-6	--	00	-6	-9
12	LFB	-9	--	000	-9	-9
13	LFB	-12	-6	00	-6	-9
14	LFNB	-15	-3	0	-3	-9
15	LFNB	-12	0	X	0	-9
16	LFNB	-9	--	1	-9	-9
17	LFB	-6	--	11	-6	-6
18	LFB	-3	--	111	-3	-3
19	LFB	0	--	1110	0	0
20	LFB	+3	--	11101	+3	+3
21	LFB	+6	--	111010	+6	+6
22	LFB	+9	--	1110100	+9	停止

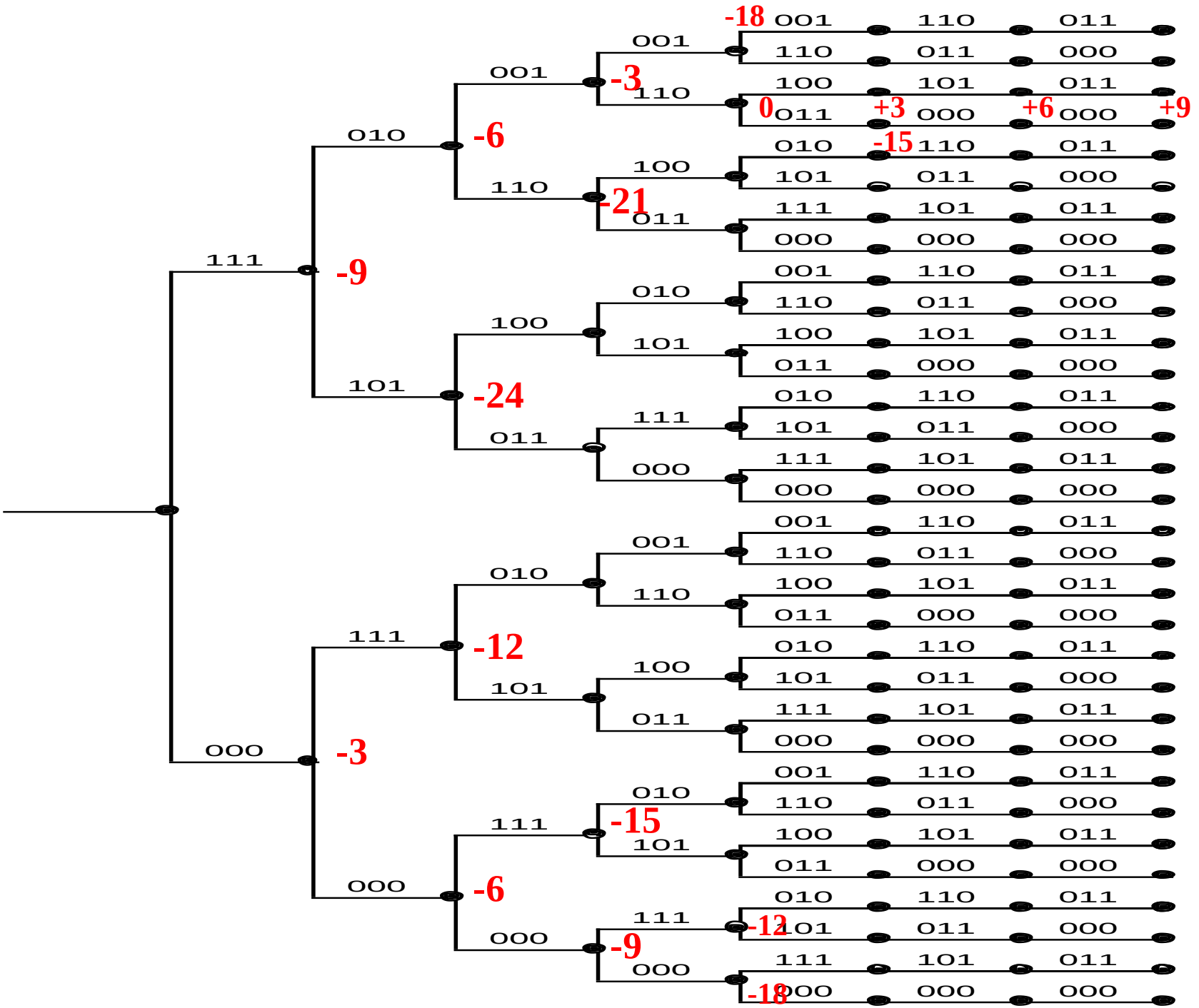

111
-9
000
-3
010
-6
101
-24
111
-12
000
-6
001
-3
100
-21
010
-15
101
-18
001
110
011
000
100
101
011
0
+3
+6
+9
010
-15
110
011
101
011
000
111
101
011
000
000
000
001
110
011
010
110
011
100
101
011
101
011
000
011
000
000
010
110
011
111
101
011
000
101
011
000
111
101
011
000
000
000
001
110
011
001
110
011
010
110
011
100
101
011
101
011
000
011
000
000
010
110
011
100
101
011
101
011
000
111
101
011
000
000
000
001
110
011
010
110
011
111
100
101
011
101
011
000
011
000
000
010
110
011
000
101
011
000
111
-12
101
011
000
111
-9
000
101
011
000
-18
000
000
000

表 2：Fano 算法的译码步骤

步数	观察	M_F	M_B	节点	量度	T
0	--	--	--	X	0	0
1	LFB	-3	-∞	X	0	-3
2	LFB	-3	--	0	-3	-3
3	LFB	-6	0	X	0	-3
4	LFNB	-9	-∞	X	0	-6
5	LFB	-3	--	0	-3	-6
6	LFB	-6	--	00	-6	-6
7	LFB	-9	-3	0	-3	-6
8	LFNB	-12	0	X	0	-6
9	LFNB	-9	-∞	X	0	-9
10	LFB	-3	--	0	-3	-9
11	LFB	-6	--	00	-6	-9
12	LFB	-9	--	000	-9	-9
13	LFB	-12	-6	00	-6	-9
14	LFNB	-15	-3	0	-3	-9
15	LFNB	-12	0	X	0	-9
16	LFNB	-9	--	1	-9	-9
17	LFB	-6	--	11	-6	-6
18	LFB	-3	--	111	-3	-3
19	LFB	0	--	1110	0	0
20	LFB	+3	--	11101	+3	+3
21	LFB	+6	--	111010	+6	+6
22	LFB	+9	--	1110100	+9	停止

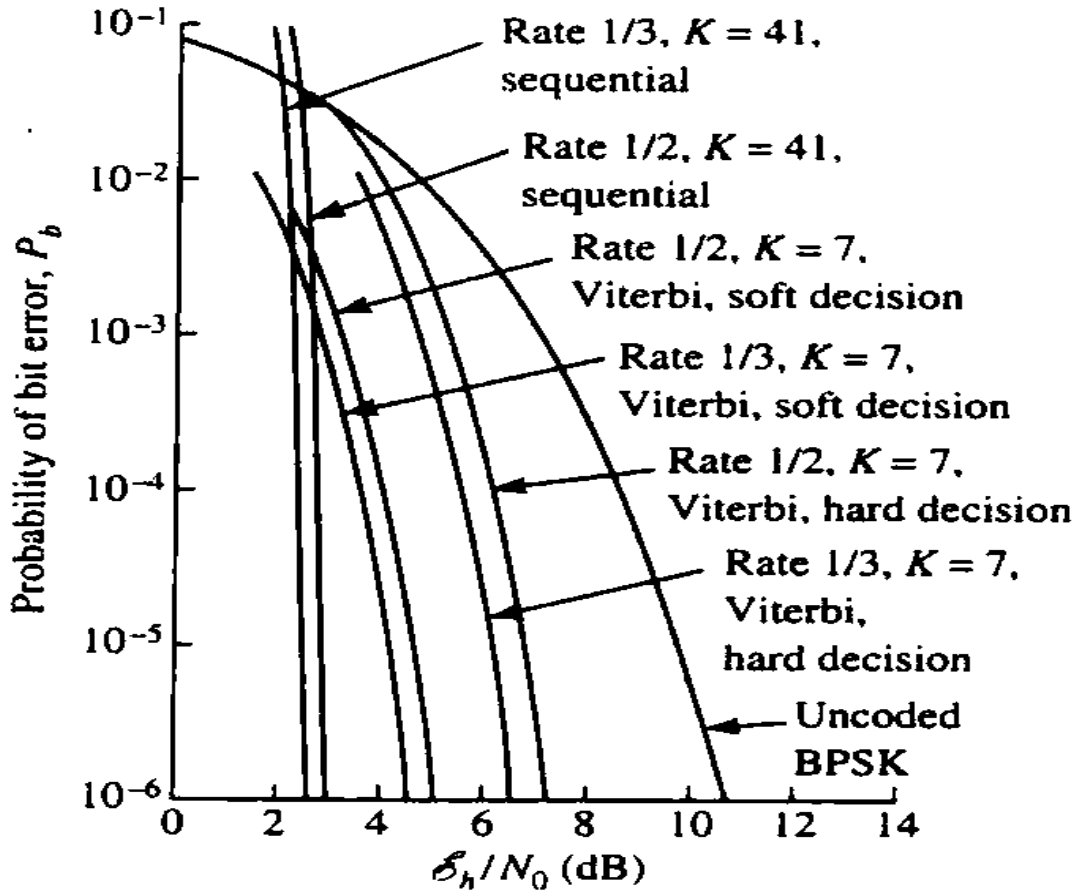
- 序列译码

  (2) Fano译码算法

  - Fano 算法所完成的计算次数与门限增量的选择有关：
    若Δ太小，其计算次数就很大
    Δ选的较大将会减少计算次数
    但若Δ太大，译出的可能不是最大似然路径

  - 比较：与Viterbi译码算法相比，其译码时延大，但存储量少

卷积码的译码


Rate 1/3, K = 41, sequential
Rate 1/2, K = 41, sequential
Rate 1/2, K = 7, Viterbi, soft decision
Rate 1/3, K = 7, Viterbi, soft decision
Rate 1/2, K = 7, Viterbi, hard decision
Rate 1/3, K = 7, Viterbi, hard decision
Uncoded BPSK
Probability of bit error, P_b
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
2
4
6
8
10
12
14
$\mathcal{E}_b/N_0$ (dB)

卷积码的译码

- 门限译码
  - 梅西在1963年提出
  - 大数逻辑译码算法
  - 译码设备简单，速度快，并适合于有突发错误的信道

卷积码的译码

- 门限译码

特点:

(1) 将卷积码化为分组码来译

(2) 每译一组k个信息比特后，要重新构成新的分组码来译

(3) 每次译码后，将译出的信息反馈输入到新的分组中

卷积码的译码

- 三种译码算法的比较
  - Viterbi译码算法具有最佳性能，但硬件实现复杂
  - 序列译码算法在性能和硬件实现方面介于两者之间
  - 门限译码算法性能相对最差，但硬件容易实现

目录

- 卷积码的编码
- 卷积码的图描述
- 距离分布多项式
- 性能分析及构造准则
- 卷积码的译码
- 打孔卷积码和咬尾卷积码

打孔卷积码

- 当信道传输质量较好时，不需要纠错能力很强的纠错码，而希望提高编码速率
- 通过删除码中某些码位而提高编码速率，称此方法为 Punctured convolutional code

- 例:(K= 7， R=1/2)卷积码

G_1: $(1\ 7\ 1)_8 \Rightarrow (0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1)_2 \Rightarrow$

$g_1(x) = 1 + x + x^2 + x^3 + x^6$,

G_2: $(1\ 3\ 3)_8 \Rightarrow (0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1)_2 \Rightarrow$

$g_2(x) = 1 + x^2 + x^3 + x^5 + x^6$,

$d_{free} = 10$

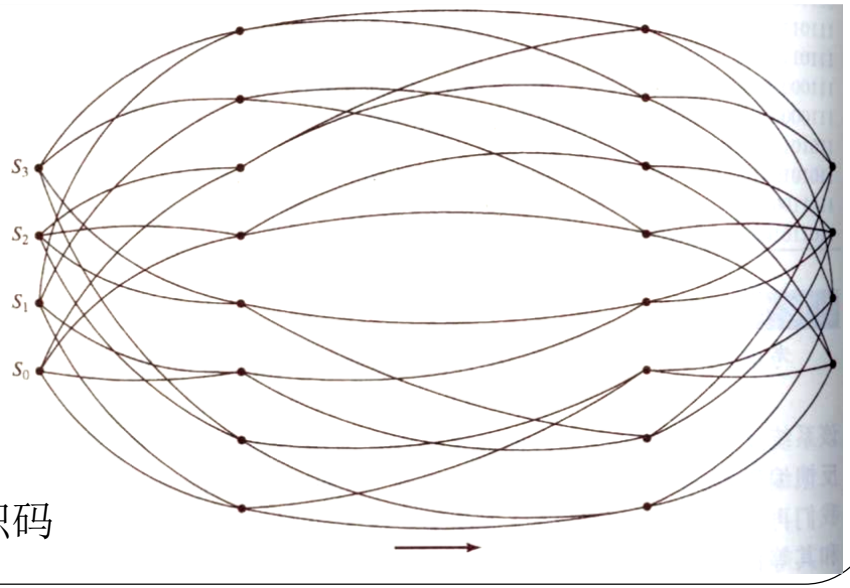
打孔卷积码

- 编码时按下表 打孔删除编码的输出
- 译码时采用Viterbi算法，但在计算分支测度时不考虑删除位，译码器结构不变

编码速率 1/2	X 1 Y 1		QPSK	I: X_1 Q: Y_1	$d_{free}=10$
编码速率 2/3	X 1 0 Y 1 1	2→4→3	QPSK	I: $X_1 Y_2 Y_3$ Q: $Y_1 X_3 Y_4$	$d_{free}=6$
编码速率 3/4	X 1 0 1 Y 1 1 0	3→6→4	QPSK	I: $X_1 Y_2$ Q: $Y_1 X_3$	$d_{free}=5$
编码速率 5/6	X 10101 Y 11010	5→10→6	QPSK	I: $X_1 Y_2 Y_4$ Q: $Y_1 X_3 X_5$	$d_{free}=4$
编码速率 7/8	X 1000101 Y 1111010	7→14→8	QPSK	I: $X_1 Y_2 Y_4 Y_6$ Q: $Y_1 Y_3 X_5 X_7$	$d_{free}=3$

咬尾卷积码

- 咬尾卷积码(Tail-biting convolutional code) 不要求编码器从s_0状态开始, 最后终止于s_0状态, 只要求起始状态和终止状态相同
- 将2^{kL}个码字分为$2^{k(K-1)}$组(编码器共有$2^{k(K-1)}$个状态), 要求每组码字起始状态和终止状态相同
- 咬尾卷积码编码器输入全部信息序列后,即停止输出, 因而$R_{bt}=k/n$



L=6的四状态咬尾剪卷积码

咬尾卷积码

$$\mathbf{G}=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{K-1} & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_{K-3} & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} \\ & & & \ddots & & & \ddots \end{bmatrix}$$

$$\mathbf{G}_L=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{K-1} & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} & \\ & & \ddots & & & \ddots & \\ & & & \mathbf{G}_0 & \cdots & \mathbf{G}_{K-2} & \mathbf{G}_{K-1}\end{bmatrix}$$

保留L行组
即kL行

$$\mathbf{G}_L^b=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{K-1} & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} \\ & & \ddots & & & \vdots \\ & & & \mathbf{G}_0 & \cdots & \mathbf{G}_{K-3}\end{bmatrix}$$

$$\mathbf{G}_L^{tb}=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{K-1} & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{K-2} & \mathbf{G}_{K-1} \\ \mathbf{G}_{K-1} & & \ddots & & & \vdots \\ \mathbf{G}_{K-2} & \mathbf{G}_{K-1} & & \mathbf{G}_0 & \cdots & \mathbf{G}_{K-3}\end{bmatrix}$$

保留L列组
即nL列

咬尾卷积码

- (3,1/2)咬尾卷积码的构造

L	$g^{(0)}$	$g^{(1)}$	d_{min}	A_{dmin}
2	3	7	2	1
3	1	5	3	4
4	1	7	4	14
5	3	7	4	10
6	3	7	4	9
7	5	7	4	7
8	5	7	4	2
9	5	7	5	18
10	5	7	5	12
11	5	7	5	11

作业

- 习题16.1，16.3，16.5，16.6