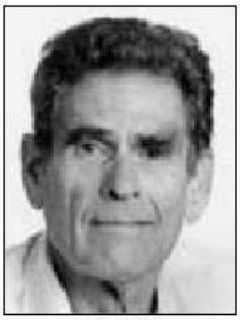
第四章 卷积码

卷积码与分组码不同，其编码器具有记忆性，即编码器的当前输出不仅与当前输入有关，还跟以前时刻的输入有关。速率 R=k/n、存储器阶数为 m 的卷积编码器可用 k 个输入、n 个输出、输入存储器阶数为 m 的线性序贯电路实现，即输入在进入编码器后仍会多呆 m 个时间单元。通常，n 和 k 都是比较小的整数，k<n，信息序列被分成长度为 k 的分组，码字(codeword)被分成长度为 n 的分组。当 k＝1 时，信息序列无需分组，处理连续进行。值得注意的是，卷积码不象分组码，较大的最小距离和低错误概率不是通过增加 k 和 n 实现的，而是通过增加存储器阶数 m 实现的。

我们这一章关注卷积编码过程、卷积编码器如何用状态图表示、如何推导卷积码的重量枚举（weight-enumerating）函数、卷积码的几种距离测量方法。

卷积码是 1955 年由 Elias（1923～2001）首次提出的，随后 Wozencraft 和 Reiffen 提出了序贯译码方法（对具有较大约束长度的卷积码非常有效）。1963 年，Massey 提出了一个效率不高、但易于实现的译码方法－－门限译码，这使得卷积码大量应用于卫星和无线信道的数字传输中。



Peter Elias

On December 7, 2001, the field of information theory lost another of its true giants, Peter Elias, who passed away at his home in Cambridge, Massachusetts, a victim of the mysterious and dreadful ailment, Creutzfeld-Jakob Disease.

A.J.Viterbi 毕业于麻省理工学院（MIT），分别获博士/硕士/学士学位,是无线通信领域国际权威专家，被称为 CDMA 之父。现为美国 Qualcomm 公司首席科学家，加州大学 Los Angeles 分校（UCLA）教授，IEEE Fellow（国际电子电气工程师协会会士）,美国总统国家科学和工程委员会成员。

1967 年，Viterbi 提出了最大似然（ML, Maximum Likelihood）译码算法，易于实现具有较小约束长度的卷积码的软判决译码。Viterbi 算法配合序贯译码的软判决，使卷积码在 20 世纪 70 年代广泛应用在深空和卫星通信系统。1974 年，Bahl, Cocke, Jelinek 以及 Raviv（BCJR）针对 具有不等先验概率信息比特的卷积码，提出了最大后验概率（MAP，maximum a posteriori probability）译码算法。BCJR 算法近年来广泛应用到软判决迭代译码方案中，其中信息比特的先验概率在每次迭代时都发生变化。对卷积码描述最详尽的书籍是：

R. Johannesson and K.S. Zigangirov, Fundamentals of Convolutional Coding. IEEE Press, Piscataway , N. J., 1999.

4.1 卷积码的编码

卷积码的编码分为两类：前馈和反馈，在每类中又可分为系统和非系统形式。首先考虑非系统形式的前馈编码器。

==========

例 4.1 速率 R＝1/2 的非系统前馈卷积编码器

图 4.1 为速率 R＝1/2、存储器阶数 m＝3 的非系统前馈卷积编码器框图，该编码器中输入信息比特 k=1、n＝2 个模 2 加法器、m＝3 个延时单元。由于模 2 加法器是一个线性运算，因此编码器是一个线性系统，所有卷积码都可用这类线性前馈移位寄存器编码器实现。

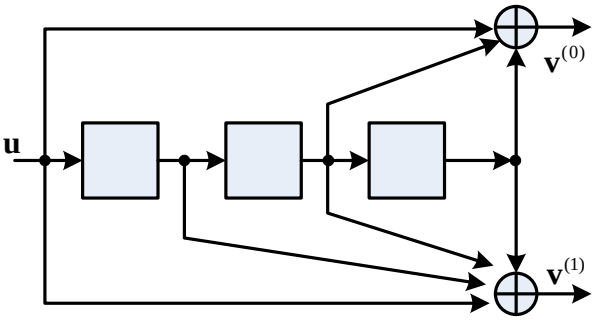


图 4.1 速率 R＝1/2 的非系统前馈卷积编码器

信息序列 $\mathbf{u}=(u_0,u_1,u_2,...)$ 进入编码器，每次 1 比特，由于编码器是一个线性系统，两个编码器输出序列 $\mathbf{v}^{(0)}=(v_0^{(0)},v_1^{(0)},v_2^{(0)},...)$ 和 $\mathbf{v}^{(1)}=(v_0^{(1)},v_1^{(1)},v_2^{(1)},...)$ 可通过输入序列 **u** 和两个编码器脉冲响应的卷积得到。计算脉冲响应时，可设 **u**=(1 0 0 …)，然后观测两个输出序列。对一个具有 m 阶存储器的编码器，脉冲响应能够持续最多 m+1 个时间单元，可写为 $\mathbf{g}^{(0)}=(g_0^{(0)},g_1^{(0)},g_2^{(0)},...,g_m^{(0)})$ 和 $\mathbf{g}^{(1)}=(g_0^{(1)},g_1^{(1)},g_2^{(1)},...,g_m^{(1)})$。对上图的编码器，有

$$\begin{aligned}\mathbf{g}^{(0)}&=(1\ 0\ 1\ 1)\\ \mathbf{g}^{(1)}&=(1\ 1\ 1\ 1)\end{aligned}\qquad(4.1)$$

脉冲响应 $\mathbf{g}^{(0)}$ 和 $\mathbf{g}^{(1)}$ 称为编码器的生成器序列。这样，编码方程为：

$$\begin{aligned}\mathbf{v}^{(0)}&=\mathbf{u}\otimes\mathbf{g}^{(0)}\\ \mathbf{v}^{(1)}&=\mathbf{u}\otimes\mathbf{g}^{(1)}\end{aligned}\qquad(4.2)$$

其中 $\otimes$ 表示离散卷积，且所有运算都是模 2 加运算，对于所有 $l\geq 0$，有：

$$\begin{aligned}v_l^{(j)}&=\sum_{i=0}^{m}u_{l-i}g_i^{(j)}\\ &=u_lg_0^{(j)}+u_{l-1}g_1^{(j)}+\cdots+u_{l-m}g_m^{(j)}\quad,j=0,1\end{aligned}\qquad(4.3)$$

其中对于所有 l<i，$u_{l-i}\triangleq 0$，这样对图 4.1 所示的编码器，有：

$$\begin{aligned}v_l^{(0)}&=u_l\qquad+u_{l-2}+u_{l-3}\\ v_l^{(1)}&=u_l+u_{l-1}+u_{l-2}+u_{l-3}\end{aligned}\qquad(4.4)$$

编码后，两个输出序列复用成一个序列，称为码字（codeword），表示为

$$\mathbf{v}=(v_0^{(0)},v_0^{(1)},v_1^{(0)},v_1^{(1)},v_2^{(0)},v_2^{(1)},\cdots) \tag{4.5}$$

例如信息序列 **u**=(1 0 1 1 1)，则输出序列为：

$$\begin{aligned}\mathbf{v}^{(0)}&=(1\quad 0\quad 1\quad 1\quad 1)\otimes(1\quad 0\quad 1\quad 1)=(1\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0\quad 1)\\ \mathbf{v}^{(1)}&=(1\quad 0\quad 1\quad 1\quad 1)\otimes(1\quad 1\quad 1\quad 1)=(1\quad 1\quad 0\quad 1\quad 1\quad 1\quad 0\quad 1)\end{aligned} \tag{4.6}$$

复用成一个序列为：$\mathbf{v}=(11,01,00,01,01,01,00,11)$ （4.7）

如果将生成器序列 $\mathbf{g}^{(0)}$ 和 $\mathbf{g}^{(1)}$ 写出矩阵形式，为：

$$\mathbf{G}=\begin{bmatrix} g_0^{(0)}g_0^{(1)} & g_1^{(0)}g_1^{(1)} & g_2^{(0)}g_2^{(1)} & \cdots & g_m^{(0)}g_m^{(1)} & & \\ & g_0^{(0)}g_0^{(1)} & g_1^{(0)}g_1^{(1)} & \cdots & g_{m-1}^{(0)}g_{m-1}^{(1)} & g_m^{(0)}g_m^{(1)} & \\ & & g_0^{(0)}g_0^{(1)} & \cdots & g_{m-2}^{(0)}g_{m-2}^{(1)} & g_{m-1}^{(0)}g_{m-1}^{(1)} & g_m^{(0)}g_m^{(1)} \\ & & & \ddots & & & \ddots \end{bmatrix} \tag{4.8}$$

其中空白区域为全 0，这样编码方程可写为矩阵形式：

$$\mathbf{v}=\mathbf{uG} \tag{4.9}$$

G 称为该编码器的生成矩阵。我们注意到 **G** 中的每一行都与前一行相同，只是向右移位了 n=2 位，它是一个半无限矩阵，对应于信息序列 **u** 是一个任意长度的序列。如果 **u** 只有有限长 h，则 **G** 具有 h 行、2(m+h)列，**v** 的长度为 2(m+h)。例如上例中 **u**=(1 0 1 1 1)，则

$$\begin{aligned}\mathbf{v}&=\mathbf{uG}\\ &=(1\quad 0\quad 1\quad 1\quad 1)\begin{bmatrix} 11 & 01 & 11 & 11 & & & & \\ & 11 & 01 & 11 & 11 & & & \\ & & 11 & 01 & 11 & 11 & & \\ & & & 11 & 01 & 11 & 11 & \\ & & & & 11 & 01 & 11 & 11 \end{bmatrix}\\ &=(11\quad 01\quad 00\quad 01\quad 01\quad 01\quad 00\quad 11)\end{aligned} \tag{4.10}$$

与我们前面的计算一致!!

==========

==========

例 4.2 速率 R=2/3 的非系统前馈卷积编码器

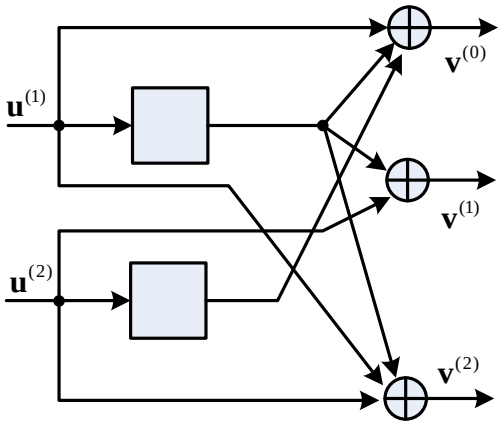


图 4.2

编码速率 R=2/3、存储器阶数 m=1 的编码器结构如图 4.2 所示，该编码器由 k=2 个移位寄存器、m=1 个时延单元、n=3 个模 2 加法器组成。信息序列进入编码器时每次进入 k=2 个比特，可写为 $\mathbf{u}=(\mathbf{u}_0,\mathbf{u}_1,\cdots)=(u_0^{(1)}u_0^{(2)},u_1^{(1)}u_1^{(2)},u_2^{(1)}u_2^{(2)},\cdots)$，或作为两个输入序列 $\mathbf{u}^{(1)}=(u_0^{(1)},u_1^{(1)},u_2^{(1)},\cdots)$ 和 $\mathbf{u}^{(2)}=(u_0^{(2)},u_1^{(2)},u_2^{(2)},\cdots)$。对应于每个输入序列，都有三个生成器序列。设 $\mathbf{g}_i^{(j)}=(g_{i,0}^{(j)},g_{i,1}^{(j)},\cdots,g_{i,m}^{(j)})$ 表示对应于输入 i 和输出 j 的生成器序列，这样我们可得到图 4.2 所示编码器的生成器序列为：

$$\begin{aligned}&\mathbf{g}_1^{(0)}=(1\ 1)\quad \mathbf{g}_1^{(1)}=(0\ 1)\quad \mathbf{g}_1^{(2)}=(1\ 1)\\&\mathbf{g}_2^{(0)}=(0\ 1)\quad \mathbf{g}_2^{(1)}=(1\ 0)\quad \mathbf{g}_2^{(2)}=(1\ 0)\end{aligned}\tag{4.11}$$

这样我们可写出编码方程如下：

$$\begin{aligned}\mathbf{v}^{(0)}&=\mathbf{u}^{(1)}\otimes\mathbf{g}_1^{(0)}+\mathbf{u}^{(2)}\otimes\mathbf{g}_2^{(0)}\\\mathbf{v}^{(1)}&=\mathbf{u}^{(1)}\otimes\mathbf{g}_1^{(1)}+\mathbf{u}^{(2)}\otimes\mathbf{g}_2^{(1)}\\\mathbf{v}^{(2)}&=\mathbf{u}^{(1)}\otimes\mathbf{g}_1^{(2)}+\mathbf{u}^{(2)}\otimes\mathbf{g}_2^{(2)}\end{aligned}\tag{4.12}$$

卷积运算意味着

$$\begin{aligned}v_l^{(0)}&=u_l^{(1)}+\quad\ +u_{l-1}^{(1)}+u_{l-1}^{(2)}\\v_l^{(1)}&=\quad\ \ u_l^{(2)}+u_{l-1}^{(1)}\\v_l^{(2)}&=u_l^{(1)}+u_l^{(2)}+u_{l-1}^{(1)}\end{aligned}\tag{4.13}$$

复用后，码字为：

$$\mathbf{v}=(v_0^{(0)}v_0^{(1)}v_0^{(2)},v_1^{(0)}v_1^{(1)}v_1^{(2)},v_2^{(0)}v_2^{(1)}v_2^{(2)},\cdots)\tag{4.14}$$

例如：如果 $\mathbf{u}^{(1)}=(1\ 0\ 1)$ 和 $\mathbf{u}^{(2)}=(1\ 1\ 0)$，则

$$\begin{aligned}\mathbf{v}^{(0)}&=(101)\otimes(11)+(110)\otimes(01)=(1001)\\\mathbf{v}^{(1)}&=(101)\otimes(01)+(110)\otimes(10)=(1001)\\\mathbf{v}^{(2)}&=(101)\otimes(11)+(110)\otimes(10)=(0011)\end{aligned}\tag{4.15}$$

所以输出序列为：

$$\mathbf{v}=(110,000,001,111)\tag{4.16}$$

生成矩阵为：

$$\mathbf{G}=\begin{bmatrix}g_{1,0}^{(0)}g_{1,0}^{(1)}g_{1,0}^{(2)} & g_{1,1}^{(0)}g_{1,1}^{(1)}g_{1,1}^{(2)} & \cdots & g_{1,m}^{(0)}g_{1,m}^{(1)}g_{1,m}^{(2)} & & \\ g_{2,0}^{(0)}g_{2,0}^{(1)}g_{2,0}^{(2)} & g_{2,1}^{(0)}g_{2,1}^{(1)}g_{2,1}^{(2)} & \cdots & g_{2,m}^{(0)}g_{2,m}^{(1)}g_{2,m}^{(2)} & & \\ & g_{1,0}^{(0)}g_{1,0}^{(1)}g_{1,0}^{(2)} & & g_{1,m-1}^{(0)}g_{1,m-1}^{(1)}g_{1,m-1}^{(2)} & g_{1,m}^{(0)}g_{1,m}^{(1)}g_{1,m}^{(2)} & \\ & g_{2,0}^{(0)}g_{2,0}^{(1)}g_{2,0}^{(2)} & & g_{2,m-1}^{(0)}g_{2,m-1}^{(1)}g_{2,m-1}^{(2)} & g_{2,m}^{(0)}g_{2,m}^{(1)}g_{2,m}^{(2)} & \\ & \ddots & & & \ddots & \end{bmatrix}\tag{4.17}$$

编码方程仍写为 $\mathbf{v}=\mathbf{uG}$，注意：$\mathbf{G}$ 中每 k＝2 行都与前 2 行相同，只是向右移 n=3 位。如上例中，$\mathbf{u}^{(1)}=(101)$ 和 $\mathbf{u}^{(2)}=(110)$，则 $\mathbf{u}=(\mathbf{u}_0,\mathbf{u}_1,\mathbf{u}_2)=(11,01,10)$，有：

$$\mathbf{v}=\mathbf{uG}=(11,01,10)\begin{bmatrix}101 & 111 & & \\ 011 & 100 & & \\ & 101 & 111 & \\ & 011 & 100 & \\ & & 101 & 111 \\ & & 011 & 100\end{bmatrix} \tag{4.18}$$

$$=(110,000,001,111)$$

与我们前面的计算一致!!

═══════════════════════════════

在上两例中，从存储输入序列的 k 个移位寄存器到 n 个模 2 加法器的连接，直接对应于在 kn 个生成器序列中的非零项，如式（4.1）和（4.11）所示。从上述的例子中可以看出，当输入序列数 k>1 时，复杂度明显增加。

下面给出与移位寄存器长度有关的四个定义：

定义 4.1：设 v_i 是卷积编码器（有 k 个输入序列）第 i 个移位寄存器的长度，i＝1, 2,…,k。

例如图 4.1 中速率 R＝1/2 的编码器，v＝3；图 4.11 中速率 R＝2/3 的编码器，$v_1 = v_2 =1$。

定义 4.2：编码器的存储器阶数 m 定义为：

$$m=\max_{1\le i\le k} v_i \tag{4.19}$$

即 m 是所有 k 个移位寄存器的最大值。

例如图 4.1 中速率 R＝1/2 的编码器，$m=v_1=3$；图 4.11 中速率 R＝2/3 的编码器，$m=\max(v_1,v_2)=\max(1,1)=1$。

定义 4.3：编码器的全局约束长度 v 定义为：

$$v=\sum_{1\le i\le k} v_i \tag{4.20}$$

即，v 是所有 k 个移位寄存器的总和。

例如图 4.1 中速率 R＝1/2 的编码器，$v=v_1=3$；图 4.11 中速率 R＝2/3 的编码器，$v=v_1+v_2=2$。

定义 4.4：全局约束长度为 v、编码速率 R＝k/n 的卷积编码器常表示为一个（n,k,v）编码器。

例如图 4.1 中速率 R＝1/2 的编码器是（2，1，3）编码器，图 4.11 中速率 R＝2/3 的编码器是（3，2，2）编码器。

因此，对于（n，1，v）编码器，全局约束长度 v 就等于存储器阶数 m。

═══════════════════════════════

例 4.3：（4，3，3）非系统前馈卷积编码器

（4，3，3）非系统前馈卷积编码器如图 4.3 所示，寄存器长度 v_1=0, v_2=1, v_3=2。存储器阶数 m＝2，全局约束长度 v＝3，生成器序列为：

$$\begin{array}{llll} g_1^{(0)}=(100) & g_1^{(1)}=(100) & g_1^{(2)}=(100) & g_1^{(3)}=(100) \\ g_2^{(0)}=(000) & g_2^{(1)}=(110) & g_2^{(2)}=(010) & g_2^{(3)}=(100) \\ g_3^{(0)}=(000) & g_3^{(1)}=(010) & g_3^{(2)}=(101) & g_3^{(3)}=(101) \end{array} \tag{4.21}$$

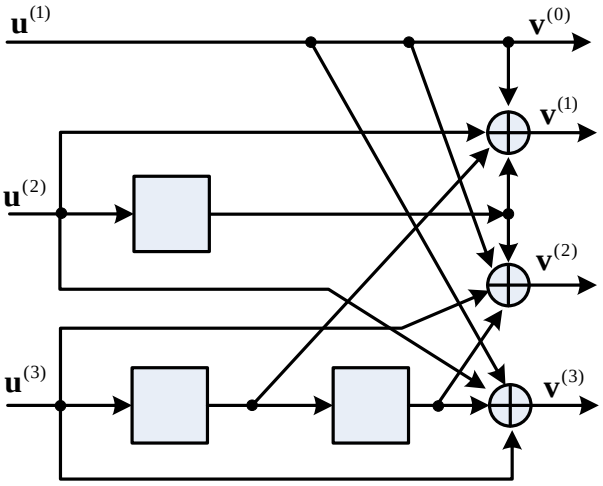


图 4.3 （4，3，3）非系统前馈卷积编码器

通常，一个存储器阶数为 m 的（n，k，v）前馈编码器，其生成矩阵可写为：

$$\mathbf{G}=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_m & & \\ & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m \\ & & \ddots & & & & \end{bmatrix} \tag{4.22}$$

其中 $\mathbf{G}_l$ 是一个 k×n 的子矩阵，表示为：

$$\mathbf{G}_l=\begin{bmatrix} g_{1,l}^{(0)} & g_{1,l}^{(1)} & \cdots & g_{1,l}^{(n-1)} \\ g_{2,l}^{(0)} & g_{2,l}^{(1)} & \cdots & g_{2,l}^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,l}^{(0)} & g_{k,l}^{(1)} & \cdots & g_{k,l}^{(n-1)} \end{bmatrix} \tag{4.23}$$

仍需要注意的是：生成矩阵 **G** 中的 k 行（一组）都与前 k 行（一组）相同，只是向右移 n 位。对于一个信息序列 $\mathbf{u}=(\mathbf{u}_0,\mathbf{u}_1,\cdots)=(u_0^{(1)}u_0^{(2)}\cdots u_0^{(k)},u_1^{(1)}u_1^{(2)}\cdots u_1^{(k)},\cdots)$，码字 $\mathbf{v}=\mathbf{u}\mathbf{G}=(v_0^{(0)}v_0^{(1)}\cdots v_0^{(n-1)},v_1^{(0)}v_1^{(1)}\cdots v_1^{(n-1)},\cdots)$。

码字 **v** 是生成矩阵 **G** 的线性组合，因此（n，k，v）是一个线性码。

在线性系统中，时域中的卷积可用更方便的多项式乘法来代替。这样编码方程中每个序列都可以用一个对应的多项式来代替，例如对一个（2，1，v）编码器，编码方程变为：

$$\begin{aligned}\mathbf{v}^{(0)}(D)&=\mathbf{u}(D)\mathbf{g}^{(0)}(D)\\ \mathbf{v}^{(1)}(D)&=\mathbf{u}(D)\mathbf{g}^{(1)}(D)\end{aligned} \tag{4.24}$$

其中

$$\mathbf{u}(D)=u_0+u_1D+u_2D^2+\cdots \tag{4.25a}$$

是信息序列。

$$\mathbf{v}^{(0)}(D)=v_0^{(0)}+v_1^{(0)}D+v_2^{(0)}D^2+\cdots$$
$$\mathbf{v}^{(1)}(D)=v_0^{(1)}+v_1^{(1)}D+v_2^{(1)}D^2+\cdots \quad (4.25b)$$

是编码后的序列。

$$\mathbf{g}^{(0)}(D)=g_0^{(0)}+g_1^{(0)}D+\cdots+g_m^{(0)}D^m$$
$$\mathbf{g}^{(1)}(D)=g_0^{(1)}+g_1^{(1)}D+\cdots+g_m^{(1)}D^m \quad (4.25c)$$

是生成多项式。

这样码字可写为：

$$\mathbf{V}(D)=\left[\mathbf{v}^{(0)}(D),\mathbf{v}^{(1)}(D)\right] \quad (4.26a)$$

或经过复用后，写为：

$$\mathbf{v}(D)=\mathbf{v}^{(0)}(D^2)+D\mathbf{v}^{(1)}(D^2) \quad (4.26b)$$

D 可看作为一个时延算子，D 的指数表示相对于序列中的初始 bit 延时了多少时间单元。

==========

例如图 4.1 所示的（2，1，3）编码器，生成多项式 $\mathbf{g}^{(0)}(D)=1+D^2+D^3$ 及 $\mathbf{g}^{(1)}(D)=1+D+D^2+D^3$，对信息序列 $\mathbf{u}(D)=1+D^2+D^3+D^4$，编码方程为：

$$\mathbf{v}^{(0)}(D)=(1+D^2+D^3+D^4)(1+D^2+D^3)=1+D^7$$
$$\mathbf{v}^{(1)}(D)=(1+D^2+D^3+D^4)(1+D+D^2+D^3)=1+D+D^3+D^4+D^5+D^7 \quad (4.27)$$

码字可写为：

$$\mathbf{V}(D)=\left[1+D^7,1+D+D^3+D^4+D^5+D^7\right] \quad (4.28a)$$

或

$$\mathbf{v}(D)=1+D+D^3+D^7+D^9+D^{11}+D^{14}+D^{15} \quad (4.28b)$$

==========

注意：生成多项式的最低阶项（常数项）对应于移位寄存器连接的最左端，最高阶项对应于移位寄存器连接的最右端。例如图 4.1 中，从移位寄存器到输出 $\mathbf{v}^{(0)}$ 的连接是 $\mathbf{g}^{(0)}=(1011)$，对应的生成多项式为 $\mathbf{g}^{(0)}(D)=1+D^2+D^3$。

在（n，1，v）编码器中，由于移位寄存器的最右端必须连接到至少一个输出上，因此至少有一个生成多项式的阶数必然等于移位寄存器的长度 m，即

$$m=\max_{0\le j\le n-1}\left[\deg \mathbf{g}^{(j)}(D)\right] \quad (4.29)$$

在 k>1 的（n，k，v）编码器中，对于每个输入（共有 k 个输入），都有 n 个生成多项式。每一组 n 个生成多项式表示着从移位寄存器到 n 个输出的连接序列，因此，有

$$v_i=\max_{0\le j\le n-1}\left[\deg\ \mathbf{g}_i^{(j)}(D)\right],\quad 1\le i\le k \quad (4.30)$$

其中 $\mathbf{g}_i^{(j)}(D)$ 是第 i 个输入到第 j 个输出的生成多项式。

由于编码器是一个线性系统，$\mathbf{u}^{(i)}(D)$表示第 i 个输入序列，$\mathbf{v}^{(j)}(D)$表示第 j 个输出序列，生成多项式$\mathbf{g}_i^{(j)}(D)$可解释为输入 i 到输出 j 的转移函数。对于有 k 个输入、n 个输出的线性系统，共有 kn 个转移函数，可用 k×n 的生成矩阵表示：

$$\mathbf{G}(D)=\begin{bmatrix}\mathbf{g}_1^{(0)}(D) & \mathbf{g}_1^{(1)}(D) & \cdots & \mathbf{g}_1^{(n-1)}(D)\\ \mathbf{g}_2^{(0)}(D) & \mathbf{g}_2^{(1)}(D) & \cdots & \mathbf{g}_2^{(n-1)}(D)\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{g}_k^{(0)}(D) & \mathbf{g}_k^{(1)}(D) & \cdots & \mathbf{g}_k^{(n-1)}(D)\end{bmatrix} \qquad (4.31)$$

使用生成矩阵，（n，k，v）前馈编码器的编码方程可写为：

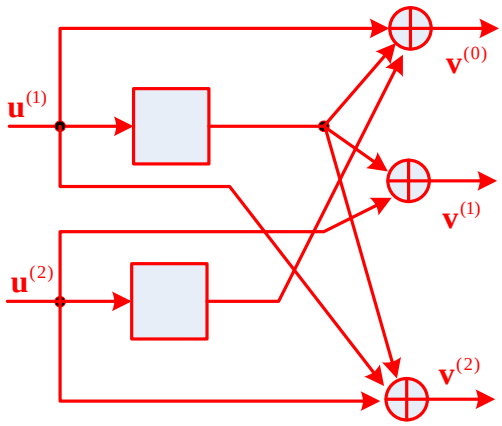
$$\mathbf{V}(D)=\mathbf{U}(D)\mathbf{G}(D) \qquad (4.32)$$

其中 $\mathbf{U}(D)\triangleq\left[\mathbf{u}^{(1)}(D),\mathbf{u}^{(2)}(D),\cdots\mathbf{u}^{(k)}(D)\right]$ 是 k-tuple 输入序列，$\mathbf{V}(D)\triangleq\left[\mathbf{v}^{(0)}(D),\mathbf{v}^{(1)}(D),\cdots\mathbf{v}^{(n-1)}(D)\right]$是 n-tuple 输出序列（码字），复用后，码字可表示为：

$$\mathbf{v}(D)=\mathbf{v}^{(0)}(D^n)+D\mathbf{v}^{(1)}(D^n)+\cdots+D^{n-1}\mathbf{v}^{(n-1)}(D^n) \qquad (4.33)$$

==========

对于图 4.2 的（3，2，2）编码器



$\mathbf{g}_1^{(0)}=(1\ 1)$ $\mathbf{g}_1^{(1)}=(0\ 1)$ $\mathbf{g}_1^{(2)}=(1\ 1)$

$\mathbf{g}_2^{(0)}=(0\ 1)$ $\mathbf{g}_2^{(1)}=(1\ 0)$ $\mathbf{g}_2^{(2)}=(1\ 0)$

$$\mathbf{G}(D)=\begin{bmatrix}1+D & D & 1+D\\ D & 1 & 1\end{bmatrix} \qquad (4.34)$$

对于输入序列$\mathbf{u}^{(1)}(D)=1+D^2$和$\mathbf{u}^{(2)}(D)=1+D$，码字为：

$$\begin{aligned}\mathbf{V}(D)&=\left[\mathbf{v}^{(0)}(D),\mathbf{v}^{(1)}(D),\mathbf{v}^{(2)}(D)\right]\\&=\left[1+D^2,1+D\right]\begin{bmatrix}1+D & D & 1+D\\ D & 1 & 1\end{bmatrix}\\&=\left[1+D^3,\ \ 1+D^3,\ \ D^2+D^3\right]\end{aligned} \qquad (4.35a)$$

也可写为：

$$\mathbf{v}(D)=1+D+D^8+D^9+D^{10}+D^{11} \qquad (4.35b)$$

==========

我们也可以将式（4.31）、（4.32）、（4.33）的复用码字 **v**(D)写成：

$$\mathbf{v}(D)=\sum_{i=1}^{k}\mathbf{u}^{(i)}(D^n)\mathbf{g}_i(D) \qquad (4.36)$$

其中

$$\mathbf{g}_i(D)=\mathbf{g}_i^{(0)}(D^n)+D\mathbf{g}_i^{(1)}(D^n)+\cdots+D^{n-1}\mathbf{g}_i^{(n-1)}(D^n),\ 1\le i\le k \qquad (4.37)$$

是第 i 个输入到输出的复合生成多项式。

==========

例如图 4.1 所示的（2，1，3）编码器，复合生成多项式为：

$$\begin{aligned}\mathbf{g}(D)&=\mathbf{g}^{(0)}(D^2)+D\mathbf{g}^{(1)}(D^2)\\&=1+D+D^3+D^4+D^5+D^6+D^7\end{aligned} \qquad (4.38)$$

当输入信息多项式$\mathbf{u}(D)=1+D^2+D^3+D^4$时，码字为：

$$\begin{aligned}\mathbf{v}(D)&=\mathbf{u}(D^2)\mathbf{g}(D)\\&=(1+D^4+D^6+D^8)(1+D+D^3+D^4+D^5+D^6+D^7)\\&=1+D+D^3+D^7+D^9+D^{11}+D^{14}+D^{15}\end{aligned} \qquad (4.39)$$

==========

卷积编码器的一个重要子类是系统编码器，在一个系统编码器中，前 k 个输出序列（称为系统输出序列）正好是 k 个输入序列的拷贝，即：

$$\mathbf{v}^{(i-1)}=\mathbf{u}^{(i)},\quad i=1,2,\cdots,k \qquad (4.40)$$

生成器序列满足：

$$\mathbf{g}_i^{(j)}=\begin{cases}1, & if\quad j=i-1\\0, & if\quad j\ne i-1\end{cases}\quad i=1,2,\cdots k \qquad (4.41)$$

在系统前馈编码器中，生成矩阵可表示为：

$$\mathbf{G}=\begin{bmatrix}\mathbf{I}\ \mathbf{P}_0 & \mathbf{0}\ \mathbf{P}_1 & \mathbf{0}\ \mathbf{P}_2 & \cdots & \mathbf{0}\ \mathbf{P}_m & & \\ & \mathbf{I}\ \mathbf{P}_0 & \mathbf{0}\ \mathbf{P}_1 & \cdots & \mathbf{0}\ \mathbf{P}_{m-1} & \mathbf{0}\ \mathbf{P}_m & \\ & & \mathbf{I}\ \mathbf{P}_0 & \cdots & \mathbf{0}\ \mathbf{P}_{m-2} & \mathbf{0}\ \mathbf{P}_{m-1} & \mathbf{0}\ \mathbf{P}_m \\ & & & \ddots & & & \end{bmatrix} \qquad (4.42)$$

其中 **I** 是 k×k 的单位阵，**0** 是 k×k 的全 0 阵，$\mathbf{P}_l$是$k\times(n-k)$的矩阵，为：

$$\mathbf{P}_l=\begin{bmatrix} g_{1,l}^{(k)} & g_{1,l}^{(k+1)} & \cdots & g_{1,l}^{(n-1)} \\ g_{2,l}^{(k)} & g_{2,l}^{(k+1)} & \cdots & g_{2,l}^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,l}^{(k)} & g_{k,l}^{(k+1)} & \cdots & g_{k,l}^{(n-1)} \end{bmatrix} \tag{4.43}$$

类似地，k×n 的生成矩阵变为：

$$\mathbf{G}(D)=\begin{bmatrix} 1 & 0 & \cdots & 0 & \mathbf{g}_1^{(k)}(D) & \cdots & \mathbf{g}_1^{(n-1)}(D) \\ 0 & 1 & \cdots & 0 & \mathbf{g}_2^{(k)}(D) & \cdots & \mathbf{g}_2^{(n-1)}(D) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \mathbf{g}_k^{(k)}(D) & \cdots & \mathbf{g}_k^{(n-1)}(D) \end{bmatrix} \tag{4.44}$$

由于前 k 个输出序列是系统的，即为 k 个输入序列，它们也被称为***输出信息序列***，后面的 n－k 个输出序列被称为***输出校验序列***。通常，需要 kn 个生成多项式来定义（n，k，*v*）非系统前馈编码器，需要 k×(n－k)个生成多项式来定义系统前馈编码器。

对于式（4.42）或（4.44）生成矩阵所示的系统前馈编码器，其对应的系统校验矩阵可直接表示为：

$$\mathbf{H}=\begin{bmatrix} \mathbf{P}_0^T & \mathbf{I} & & & & & & & & & & \\ \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I} & & & & & & & & \\ \mathbf{P}_2^T & \mathbf{0} & \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & & & & & \\ \mathbf{P}_m^T & \mathbf{0} & \mathbf{P}_{m-1}^T & \mathbf{0} & \mathbf{P}_{m-2}^T & \mathbf{0} & \cdots & \mathbf{P}_0^T & \mathbf{I} & & & \\ & & \mathbf{P}_m^T & \mathbf{0} & \mathbf{P}_{m-1}^T & \mathbf{0} & \cdots & \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I} & \\ & & & & \mathbf{P}_m^T & \mathbf{0} & \cdots & \mathbf{P}_2^T & \mathbf{0} & \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & \mathbf{I} & \ddots \end{bmatrix} \tag{4.45}$$

其中 **I** 是（n－k）×（n－k）的单位阵，**0** 是（n－k）×（n－k）的全零矩阵，类似地，（n－k）×n 的校验矩阵可表示为：

$$\mathbf{H}(D)=\begin{bmatrix} \mathbf{g}_1^{(k)}(D) & \mathbf{g}_2^{(k)}(D) & \cdots & \mathbf{g}_k^{(k)}(D) & 1 & 0 & \cdots & 0 \\ \mathbf{g}_1^{(k+1)}(D) & \mathbf{g}_2^{(k+1)}(D) & \cdots & \mathbf{g}_k^{(k+1)}(D) & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}_1^{(n-1)}(D) & \mathbf{g}_2^{(n-1)}(D) & \cdots & \mathbf{g}_k^{(n-1)}(D) & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.46a}$$

其中 **H**(D)的后（n－k）列是（n－k）×（n－k）的单位阵，上式也可写为：

$$\mathbf{H}(D)=\begin{bmatrix} \mathbf{h}_1^{(0)}(D) & \mathbf{h}_1^{(1)}(D) & \cdots & \mathbf{h}_1^{(k-1)}(D) & 1 & 0 & \cdots & 0 \\ \mathbf{h}_2^{(0)}(D) & \mathbf{h}_2^{(1)}(D) & \cdots & \mathbf{h}_2^{(k-1)}(D) & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_{n-k}^{(0)}(D) & \mathbf{h}_{n-k}^{(1)}(D) & \cdots & \mathbf{h}_{n-k}^{(k-1)}(D) & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.46b}$$

其中与第（i－1）个输出信息序列相联系的第（j－k＋1）个校验多项式定义为 $\mathbf{h}_{j-k+1}^{(i-1)}(D)=\mathbf{g}_i^{(j)}(D)$，j＝k, k+1, …, n-1，i＝1, 2, …, k。为了简化表达，可用 $\mathbf{h}_i^{(j)}(D)$，i＝1，2，…,n－k，j＝0，1，…,k－1 来表达。

任何一个有效的码字 **v**（或 **V**(D)）必须满足校验方程：

$$\mathbf{v}\mathbf{H}^{\mathrm{T}}=\mathbf{0} \qquad (4.47)$$

或

$$\mathbf{v}(D)\mathbf{H}^{\mathrm{T}}(D)=\mathbf{0}(D) \qquad (4.48)$$

==========

例 4.4 速率 R＝1/2 的系统前馈卷积编码器

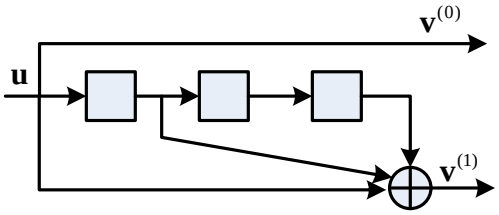


图 4.4 速率 R＝1/2 的系统前馈卷积编码器

（2，1，3）系统前馈编码器如图 4.4 所示，生成器序列为 g(0)=(1000)和 g(1)=(1101)，生成矩阵为

$$\mathbf{G}=\begin{bmatrix} 11 & 01 & 00 & 01 & & \\ & 11 & 01 & 00 & 01 & \\ & & 11 & 01 & 00 & 01 \\ & & & \ddots & & \ddots \end{bmatrix} \qquad (4.49)$$

及

$$\mathbf{G}(D)=\begin{bmatrix} 1 & 1+D+D^3 \end{bmatrix} \qquad (4.50)$$

当输入序列$\mathbf{u}(D)=1+D^2+D^3$时，输出信息序列为：

$$\mathbf{v}^{(0)}(D)=\mathbf{u}(D)\mathbf{g}^{(0)}(D)=(1+D^2+D^3)(1)=1+D^2+D^3 \qquad (4.51)$$

输出校验序列为：

$$\begin{aligned}\mathbf{v}^{(1)}(D)&=\mathbf{u}(D)\mathbf{g}^{(1)}(D)=(1+D^2+D^3)(1+D+D^3)\\&=1+D+D^2+D^3+D^4+D^5+D^6\end{aligned} \qquad (4.52)$$

这样，码字就为：

$$\begin{aligned}\mathbf{V}(D)&=\begin{bmatrix}\mathbf{v}^{(0)}(D) & \mathbf{v}^{(1)}(D)\end{bmatrix}\\&=\begin{bmatrix}1+D^2+D^3 & 1+D+D^2+D^3+D^4+D^5+D^6\end{bmatrix}\end{aligned} \qquad (4.53a)$$

或

$$\begin{aligned}\mathbf{v}(D)&=\mathbf{v}^{(0)}(D^2)+D\mathbf{v}^{(1)}(D^2)\\&=1+D+D^3+D^4+D^5+D^6+D^7+D^9+D^{11}+D^{13}\end{aligned} \qquad (4.53b)$$

也可写为：

$$\mathbf{v}=(11,01,11,11,01,01,01) \qquad (4.54)$$

利用式（4.45）和式（4.46b），我们可得到校验矩阵：

$$\mathbf{H}=\begin{bmatrix} 11 & & & & & \\ 10 & 11 & & & & \\ 00 & 10 & 11 & & & \\ 10 & 00 & 10 & 11 & & \\ & 10 & 00 & 10 & 11 & \\ & & 10 & 00 & 10 & 11 \\ & & & \ddots & & \ddots \end{bmatrix} \qquad (4.55)$$

及

$$\mathbf{H}(D)=\left[\mathbf{h}^{(0)}(D) \quad 1\right]=\left[1+D+D^3 \quad 1\right] \qquad (4.56)$$

对于式（4.53）表示的码字，我们可验证：

$$\mathbf{vH}^T=[11,01,11,11,01,01,01]\begin{bmatrix} 1 & 1 & 0 & 1 & & & \\ 1 & 0 & 0 & 0 & & & \\ & 1 & 1 & 0 & 1 & & \\ & 1 & 0 & 0 & 0 & & \\ & & 1 & 1 & 0 & 1 & \\ & & 1 & 0 & 0 & 0 & \\ & & & 1 & 1 & 0 & 1 \\ & & & 1 & 0 & 0 & 0 \\ & & & & 1 & 1 & 0 \\ & & & & 1 & 0 & 0 \\ & & & & & 1 & 1 \\ & & & & & 1 & 0 \\ & & & & & & 1 \\ & & & & & & 1 \end{bmatrix}=\mathbf{0} \qquad (4.56)$$

或

$$\mathbf{V}(D)\mathbf{H}^T(D)=\left[1+D^2+D^3 \quad 1+D+D^2+D^3+D^4+D^5+D^6\right]\begin{bmatrix} 1+D^2+D^3 \\ 1 \end{bmatrix} \qquad (4.57)$$

$$=\mathbf{0}(D)$$

==========

==========

例 4.5 速率 R＝2/3 的系统前馈卷积编码器

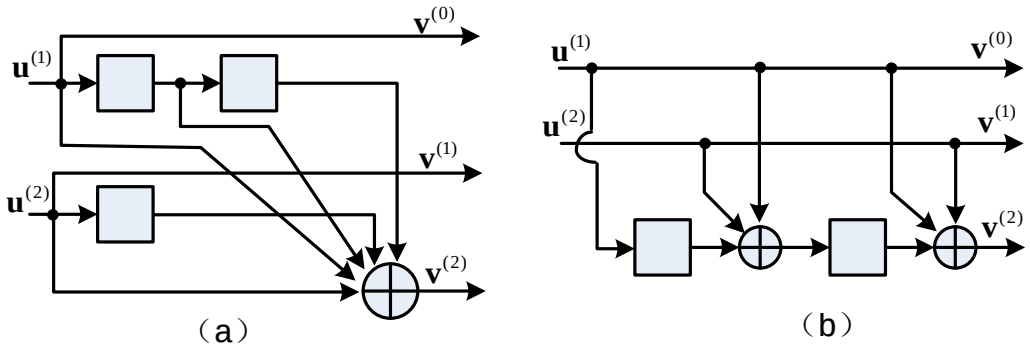


图 4.5 速率 R＝2/3 的系统前馈卷积编码器的两种表示方式

（a）controller canonical form（b）observer canonical form

生成矩阵：

$$\mathbf{G}(D)=\begin{bmatrix}1 & 0 & 1+D+D^2\\ 0 & 1 & 1+D\end{bmatrix} \qquad (4.58)$$

根据式（4.46b），校验矩阵可写为：

$$\mathbf{H}(D)=\begin{bmatrix}\mathbf{h}^{(0)}(D) & \mathbf{h}^{(1)}(D) & 1\end{bmatrix}=\begin{bmatrix}1+D+D^2 & 1+D & 1\end{bmatrix} \qquad (4.59)$$

在图 4.5（a）所示的方式中，编码器需要共 $\nu=\nu_1+\nu_2=3$ 个时延单元，因此它是一个（3，2，3）编码器。再者，由于输出信息序列为 $\mathbf{v}^{(0)}(D)=\mathbf{u}^{(1)}(D)$ 及 $\mathbf{v}^{(1)}(D)=\mathbf{u}^{(2)}(D)$，输出校验序列就为：

$$\begin{aligned}\mathbf{v}^{(2)}(D)&=\mathbf{u}^{(1)}(D)\mathbf{g}_1^{(2)}(D)+\mathbf{u}^{(2)}(D)\mathbf{g}_2^{(2)}(D)\\&=(1+D+D^2)\mathbf{u}^{(1)}(D)+(1+D)\mathbf{u}^{(2)}(D)\end{aligned} \qquad (4.60)$$

也可以用图 4.5(b)来表示，注意这种方式只需要 ν＝2 个时延单元，因此是一个（3，2，2）编码器。

因此，对于同样的卷积码，可以用不同的方式表示。我们知道，卷积码可以用状态图表示（共有 2^{ν} 个状态），在进行最大似然译码和最大后验概率译码时，其复杂度正比于编码器状态图中的状态数，因此我们希望寻找具有最小状态数（即最小全局约束长度 ν）的编码器实现。

[在介绍过系统反馈编码器后我们仍将继续讨论最小编码器实现的问题。]

＝＝

例 4.6 速率 R＝1/3 的系统反馈卷积编码器

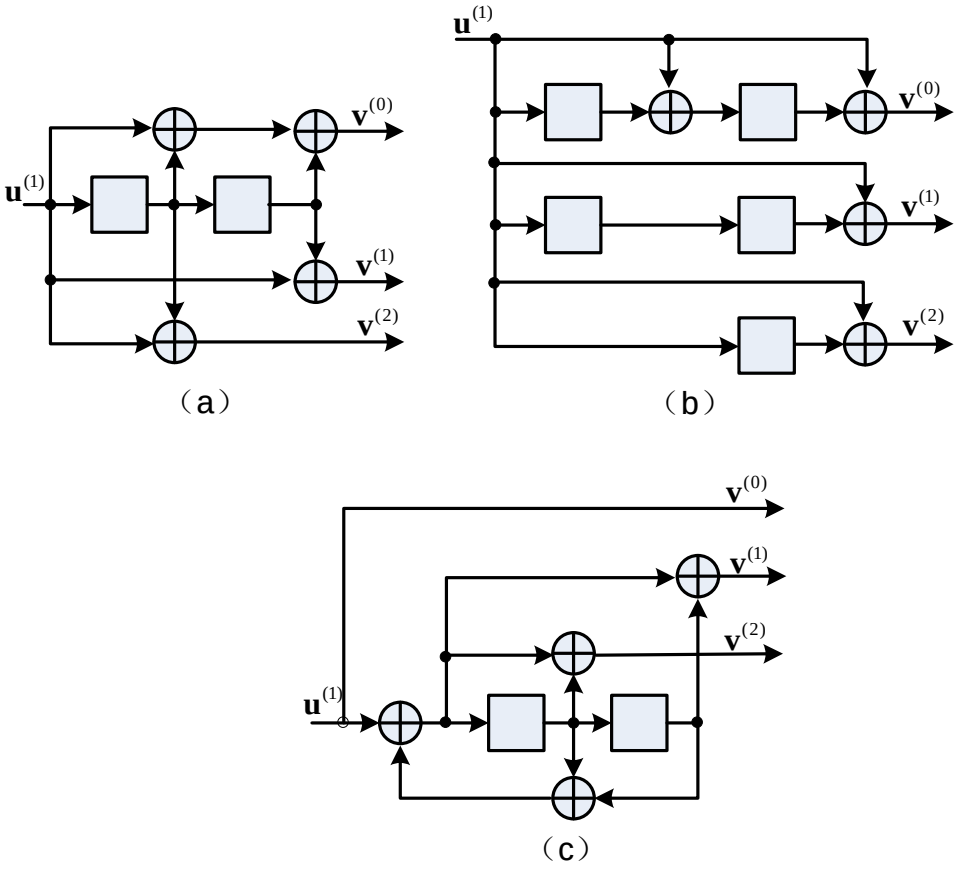


图 4.6 （a）非系统前馈编码器 controller canonical form
（b）非系统前馈编码器 observer canonical form（c）系统反馈编码器 controller canonical form

考虑一个编码速率 R＝1/3 的非系统前馈卷积编码器的 CCF 方式，如图 4.6（a）所示，其生成矩阵为：

$$\mathbf{G}(D)=\left[\mathbf{g}^{(0)}(D)\quad \mathbf{g}^{(1)}(D)\quad \mathbf{g}^{(2)}(D)\right]=\left[1+D+D^2\quad 1+D^2\quad 1+D\right] \tag{4.61}$$

它是一个（3，1，2）的编码器，有 4 种状态；图 4.6（b）所示的 OCF 方式具有相同的生成矩阵，但它是一个（3，1，5）编码器，有 32 种状态。如果将 G(D)除以多项式 $\mathbf{g}^{(0)}(D)=1+D+D^2$，如图 4.6（c）所示，会得到一个 CCF 方式系统反馈编码器的生成矩阵：

$$\begin{aligned}\mathbf{G}'(D)&=\left[1\quad \mathbf{g}^{(1)}(D)/\mathbf{g}^{(0)}(D)\quad \mathbf{g}^{(2)}(D)/\mathbf{g}^{(0)}(D)\right]\\&=\left[1\quad (1+D^2)/(1+D+D^2)\quad (1+D)/(1+D+D^2)\right]\end{aligned} \tag{4.62}$$

此时 $\mathbf{G}'(D)$ 所表示的编码器的脉冲响应具有无限长周期，即 $\mathbf{G}'(D)$ 的反馈移位寄存器实现是一个无限脉冲响应（IIR）线性系统，因此，对应于 $\mathbf{G}'(D)$ 的生成矩阵 $\mathbf{G}'$ 就包含有无限长的序列。例如，生成多项式 $\mathbf{g}^{(1)}(D)/\mathbf{g}^{(0)}(D)$ 的比值为 $(1+D^2)/(1+D+D^2)$ = 1+D+D^2+D^4+D^5+D^7+D^8+D^{10}+...，对应的无限长序列为（11101101101...）。

注意：由$\mathbf{G}'(D)$产生的编码器输出序列与由$\mathbf{G}(D)$产生的编码器输出序列完全相同，因为：在由 **G**(D)产生的码中，如果由信息序列 **u**(D)产生码字 **V**(D)，即 **V**(D)=**u**(D)**G**(D)；在由$\mathbf{G}'(D)$产生的码中，$\mathbf{u}'(D)=(1+D+D^2)\mathbf{u}(D)$，因此产生相同的码字 **V**(D)，即$\mathbf{V}(D)=\mathbf{u}'(D)\mathbf{G}'(D)$。

由于$\mathbf{G}'(D)$是系统的，我们可用式（4.46a）得到校验矩阵：

$$\mathbf{H}'(D)=\begin{bmatrix}(1+D^2)/(1+D+D^2) & 1 & 0\\ (1+D)/(1+D+D^2) & 0 & 1\end{bmatrix} \qquad (4.63)$$

其中$\mathbf{H}'(D)$对$\mathbf{G}(D)$和$\mathbf{G}'(D)$都是有效的校验矩阵。

==========

通常，如果 k×n 的多项式矩阵 **G**(D)是一个编码速率 R＝k/n 的非系统卷积编码器的生成矩阵，则可通过初等行变换把它变为系统生成矩阵$\mathbf{G}'(D)$，形成系统反馈编码器。在反馈编码器中，由于单个非零输入的响应具有无限长周期，由反馈编码器产生的码称为递归卷积码(RCC，Recursive Convolutional Code)。递归码的这种特性是并行级连卷积码（Turbo 码）具有优异性能的关键因素。

==========

例 4.7：在例 4.2 中所示的非系统前馈卷积码中，其生成矩阵 **G**(D)如式（4.34）所示，现重写如下：

$$\mathbf{G}(D)=\begin{bmatrix}1+D & D & 1+D\\ D & 1 & 1\end{bmatrix} \qquad (4.34)$$

它是一个（3，2，2）卷积码，状态数为 4。为了将它转化为一个等效的系统反馈编码器，我们可进行如下的初等行变换：

（1）第一行→→[1/(1+D)][第一行]；

（2）第二行→→第二行＋[D][第一行]；

（3）第二行→→[(1+D)/(1+D+D^2)][第二行]；

（4）第一行→→第一行＋[D/(1+D)][第二行]；

这样就可得到系统反馈形式的生成矩阵：

$$\mathbf{G}'(D)=\begin{bmatrix}1 & 0 & 1/(1+D+D^2)\\ 0 & 1 & (1+D^2)/(1+D+D^2)\end{bmatrix} \qquad (4.64)$$

根据式（4.46b）可得到系统校验矩阵：

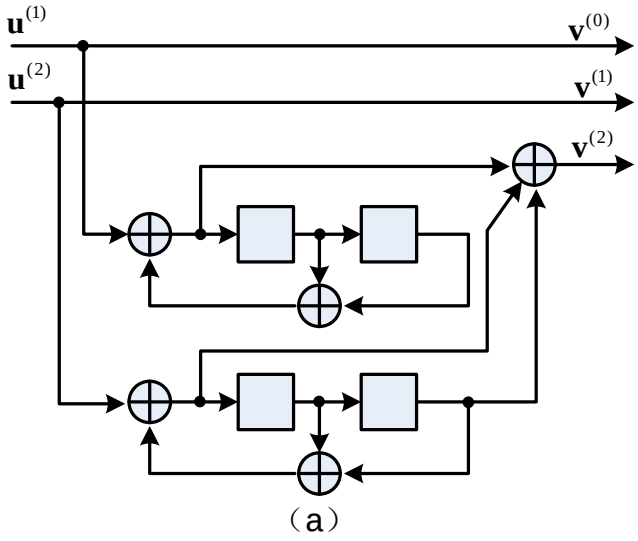
$$\begin{aligned}\mathbf{H}'(D)&=\left[\mathbf{h}^{(0)}(D)/\mathbf{h}^{(2)}(D) \quad \mathbf{h}^{(1)}(D)/\mathbf{h}^{(2)}(D) \quad 1\right]\\ &=\left[1/(1+D+D^2) \quad (1+D^2)/(1+D+D^2) \quad 1\right]\end{aligned} \qquad (4.65a)$$

或等效的非系统校验矩阵多项式：

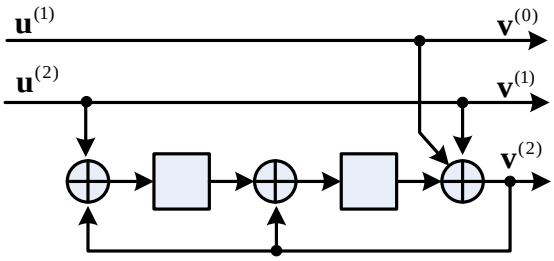
$$\begin{aligned}\mathbf{H}(D) &= \left[\mathbf{h}^{(0)}(D) \quad \mathbf{h}^{(1)}(D) \quad \mathbf{h}^{(2)}(D)\right] \\ &= \left[1 \quad 1+D^2 \quad 1+D+D^2\right]\end{aligned} \tag{4.65b}$$

其中$\mathbf{h}^{(j)}(D)$表示与第 j 个输出相联系的校验多项式。

同样，由$\mathbf{G}'(D)$产生的递归器与由$\mathbf{G}(D)$产生的非递归码完全相同，对应于式（4.64）的 CCF 方式系统反馈编码器如图 4.7（a）所示，显然，它是一个（3，2，4）编码器，状态数为 16。



（a）



（b）

图 4.7 (a)（3，2，4）CCF 方式系统反馈编码器（b）（3，2，2）OCF 系统反馈编码器

如果我们将编码器改成如图 4.7（b）所示的 OCF 方式，则它就是（3，2，2）编码器，状态数只有 4 个，这样译码时的复杂度就会大大降低。在前面例 4.2 中，非系统前馈编码器实现也只有 4 个状态数，因此，这两种方式都是最小编码器实现。

═══════════════════════════════════

4.2 卷积码的结构特性

由于卷积编码器是一个线性序贯电路，因此其运行状况可以用状态图来描述。**一个编码器的状态定义为其移位寄存器的内容。**对一个 CCF 方式的（n，k，v）编码器来说，在第 l 时刻、第 i 个支路移位寄存器（当$u_l^{(1)}, u_l^{(2)}, \cdots u_l^{(k)}$是编码器的输入）包含有 v_i 个比特，表示

为 $s_{l-1}^{(i)}, s_{l-2}^{(i)}, \cdots s_{l-v_i}^{(i)}$，其中 $s_{l-1}^{(i)}$ 表示最左边时延单元的内容，$s_{l-v_i}^{(i)}$ 表示最右边时延单元的内容，$1 \le i \le k$。

定义 4.5： 在 l 时刻的编码器状态 $\boldsymbol{\sigma}_l$ 是一个二进制的 v-tuple

$$\boldsymbol{\sigma}_l = \left(s_{l-1}^{(1)} s_{l-2}^{(1)} \cdots s_{l-v_1}^{(1)} s_{l-1}^{(2)} s_{l-2}^{(2)} \cdots s_{l-v_2}^{(2)} \cdots s_{l-1}^{(k)} s_{l-2}^{(k)} \cdots s_{l-v_k}^{(k)} \right) \qquad (4.66)$$

因此，我们从式（4.20）可知，共有 2^v 个不同状态。对一个（n，1，v）编码器，在 l 时刻编码器的状态简化为：

$$\boldsymbol{\sigma}_l = \left(s_{l-1} s_{l-2} \cdots s_{l-v} \right) \qquad (4.67)$$

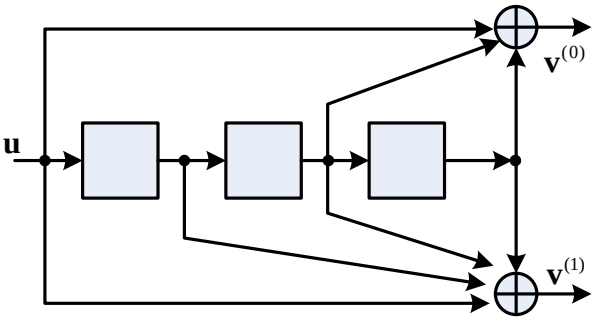


图 4.1 速率 R＝1/2 的非系统前馈卷积编码器

在前馈编码器的情况下，我们可从编码器框图（例如图 4.1）中知道：在第 l 时刻、第 i 个支路移位寄存器的内容是前 v_i 个输入，即 $u_{l-1}^{(i)}, u_{l-2}^{(i)}, \cdots u_{l-v_i}^{(i)}$，因此，编码器状态为：

$$\boldsymbol{\sigma}_l = \left(u_{l-1}^{(1)} u_{l-2}^{(1)} \cdots u_{l-v_1}^{(1)} u_{l-1}^{(2)} u_{l-2}^{(2)} \cdots u_{l-v_2}^{(2)} \cdots u_{l-1}^{(k)} u_{l-2}^{(k)} \cdots u_{l-v_k}^{(k)} \right) \qquad (4.68)$$

在（n，1，v）编码器中，上式简化为：

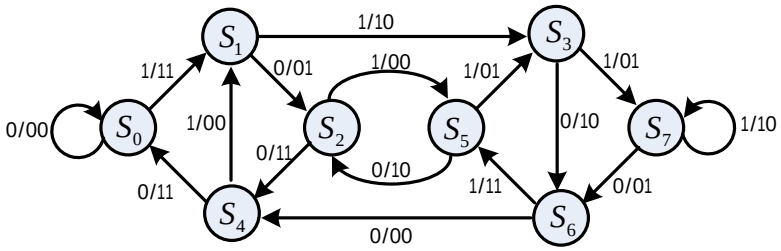
$$\boldsymbol{\sigma}_l = \left(u_{l-1} u_{l-2} \cdots u_{l-v} \right) \qquad (4.69)$$

对于 OCF 方式的编码器实现，编码器状态 $\boldsymbol{\sigma}_l$ 仍以式（4.66）定义，但它共有 n 个支路的移位寄存器，而不是 k 个，而且，OCF 方式的前馈编码器也不具备寄存器内容是其前 v_i 个输入这个特性，即式（4.69）只适用于 CCF 方式的前馈编码器实现。

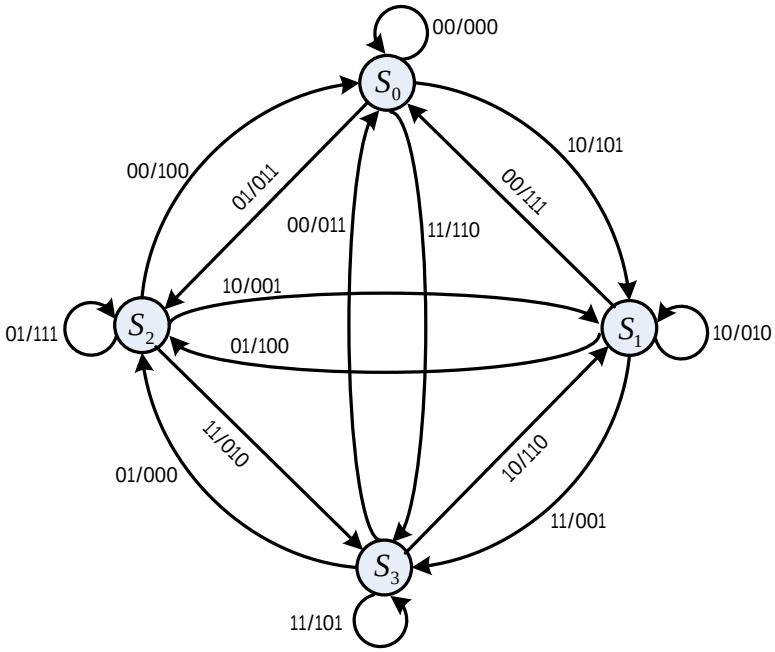
每当输入 k 比特组都会引起编码器的移位，即转移到一个新的状态，因此，在状态图中离开每个状态都有 2^k 个分支，每个分支对应于一个不同的输入比特组。对（n，1，v）编码器，离开每个状态只有两个分支。状态图中的每个分支标以 k 个输入（$u_l^{(1)}, u_l^{(2)}, \cdots u_l^{(k)}$）引起转移，产生 n 个对应的输出（$v_l^{(0)}, v_l^{(1)}, \cdots v_l^{(n-1)}$）。

＝＝

例 4.8：图 4.1 所示的 CCF 方式的（2,1,3）前馈编码器的状态图如图 4.8(a)所示，图 4.2 所示的 CCF 方式的（3,2,2）前馈编码器的状态图如图 4.8(b)所示。



（a）



（b）

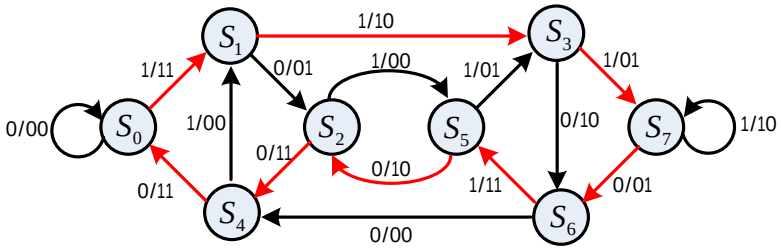
图 4.8 （a）(2,1,3)编码器的状态图（b）(3,2,2)编码器的状态图

==========

状态标识为$S_0, S_1, \cdots S_{2^v-1}$，即在$S_i$中，表示状态的二进制 v-tuple 表达$b_0, b_1, \cdots b_{v-1}$等效为用整数$i = b_0 2^{(v-1)} + b_1 2^{(v-2)} + \cdots + b_{v-1} 2^0$来表达。

注意：对于时不变卷积编码器（即编码器的生成矩阵不随时间改变），其状态图也是时不变的，因此相同的状态图在所有时间单元内都有效；但对于时变卷积编码器（例如 Turbo 码），其状态图是随时间变换的。

假设编码器初始状态在S_0（全 0 态），对于给定的信息序列根据状态图就可得到码字，最后还要补 mk 个 0 使状态返回到S_0。例如在图 4.8（a）所示的（2，1，3）编码器状态图中，如果输入序列为 **u**=(11101)，编码到最后时还要补 mk＝3 个 0 以清空寄存器，最后的输出序列为 v＝（11, 10, 01, 01, 11, 10, 11, 11）。



注意：灾难性的编码器会造成灾难性错误传播！

所谓灾难性错误传播，指有限数量的码元差错引起的无限数量的已译码数据比特差错。对于编码效率为 1/n 的编码方式，发生灾难性错误传播的条件是这些生成多项式有共同的多项式因子(阶数不低于 1)。

例如（2，1，2）编码器，其生成矩阵多项式为：

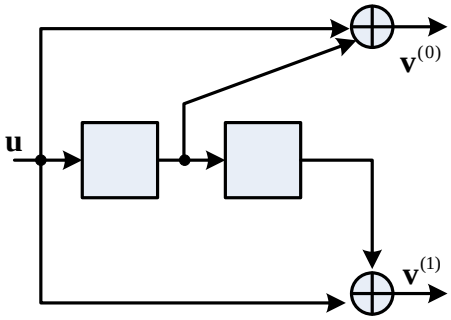


图 4.10 （2，1，2）灾难性编码器

$$\mathbf{G}(D)=\left[\mathbf{g}^{(0)}(D)\quad \mathbf{g}^{(1)}(D)\right]=\left[1+D\quad 1+D^2\right] \tag{4.70}$$

我们很容易看出：

$$GCD\left[\mathbf{g}^{(0)}(D)\quad \mathbf{g}^{(1)}(D)\right]=GCD\left[1+D\quad 1+D^2\right]=1+D \tag{4.71}$$

所以该编码器会引起灾难性错误传播。

例如，当信息序列为$\mathbf{u}(D)=\dfrac{1}{1+D}=1+D+D^2+\cdots$时（无限长），输出序列为$\mathbf{v}^{(0)}(D)=1$和$\mathbf{v}^{(1)}(D)=1+D$，也就是说，虽然信息序列重量为无限，输出码字的重量只有 3!!如 果它在 BSC 信道上传输，由于信道噪声的影响，这三个非 0 比特变成了 0，接收序列就变为了全 0 序列，经最大似然（Maximum Likelihood）译码器后判决信息序列$\hat{\mathbf{u}}(D)=\mathbf{0}(D)$，这也就意味着有限数量的信道错误会引起无限数量的译码错误。

对于任意码率的编码器，当且仅当状态图包含有一个全 0 输出的闭环路径（初始状态S_0的全 0 输出闭环路径除外）时，编码器是灾难性的。

例如上例中的编码器状态图为：

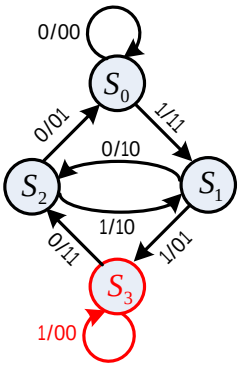


图 4.11 灾难性编码器（2，1，2）的状态图

围绕状态 S_3 有一个闭环路径，即 $S_3 \to S_3$ 的输出重量为 0。

对状态图加以修改可得到所有非零码字汉明重量（Hamming weights）的详细描述，即该码的*码字重量枚举函数*（WEF，Weight Enumerating Function）。状态 S_0 分为初始态和终止态（围绕 S_0 的全 0 输出闭环路径就去掉了），每个分支用分支增益 X^d 进行标识，其中 d 是该分支上 n 个编码后的输出比特重量。每条连接着初始态和终止态的路径都表示一个从状态 S_0 分开又在状态 S_0 汇聚的非零码字。路径增益是一条路径上所有分支增益的乘积。例如图 4.1 和图 4.2 所示编码器的修正状态图如图 4.12(a)和(b)所示。

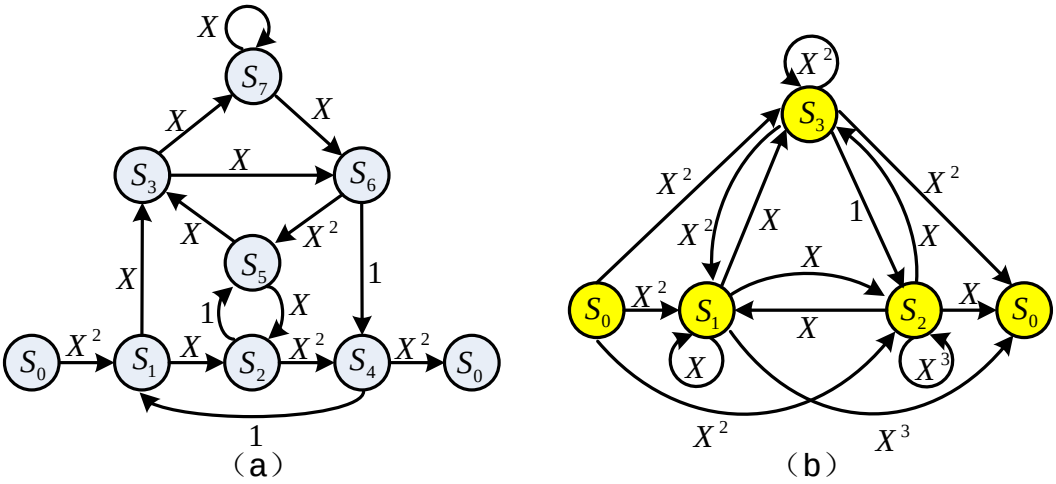
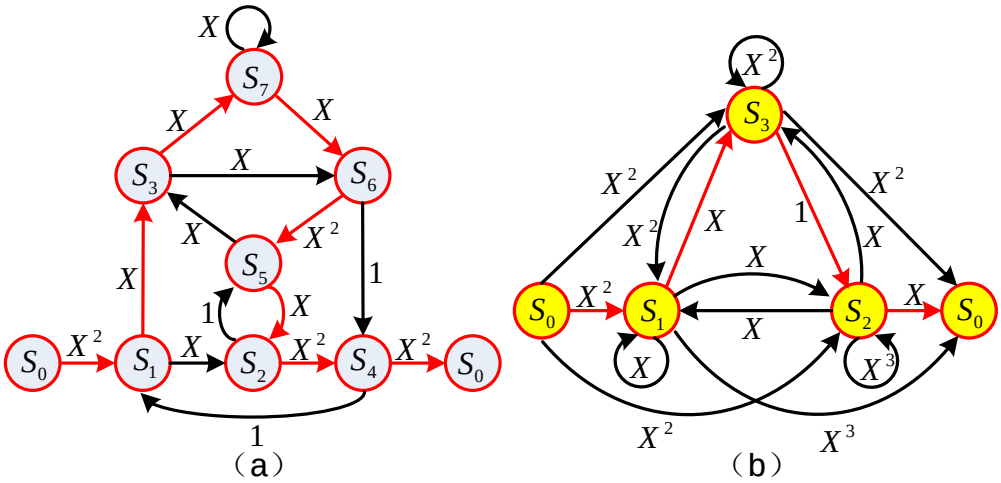


图 4.12 （a）(2,1,3)编码器的修正状态图（b）(3,2,2)编码器的修正状态图

图 4.12（a）中，表示状态序列 $S_0S_1S_3S_7S_6S_5S_2S_4S_0$ 的路径具有的路径增益为 $X^2 \cdot X^1 \cdot X^1 \cdot X^1 \cdot X^2 \cdot X^1 \cdot X^2 \cdot X^2 = X^1$ ，即对应的码字重量为 12；图 4.12（b）中，表示状态序列 $S_0S_1S_3S_2S_0$ 的路径具有的路径增益为 $X^2 \cdot X^1 \cdot X^0 \cdot X^1 = X^4$，即对应的码字重量为 4，如下图所示。



如果将修正状态图作为一个信号流图来考虑，就可得到码字的重量枚举函数（WEF），为：

$$A(X)=\sum_{d}A_d X^d \qquad (4.72)$$

其中 A_d 是重量为 d 的码字的数目。

在一个状态流图中，连接初始态和终止态的路径且经过任一状态不超过 2 次，则该路径称为前向路径（forward path），设 F_i 是第 i 个前向路径的增益。如果一个闭环路径，从任一状态开始，又返回到该状态，且经过任一状态不超过 2 次，则称为一个环（cycle），设 C_i 是第 i 个环的增益。在环集合中，如果没有一个状态属于该集合中的多个（≥2）环，则该环集合是不相交的（nontouching）。设 $\{i\}$是所有环集合，$\{i',j'\}$是所有不相交环对（pair）的集合，$\{i'',j'',l''\}$是所有不相交环三角的集合，依此类推。

我们定义

$$\Delta=1-\sum_{i}C_i+\sum_{i',j'}C_{i'}C_{j'}-\sum_{i'',j'',l''}C_{i''}C_{j''}C_{l''}+\cdots \qquad (4.73)$$

其中 $\sum_{i}C_i$ 是所有环增益的和，$\sum_{i',j'}C_{i'}C_{j'}$ 是所有不相交环对（pair）增益乘积之和，$\sum_{i'',j'',l''}C_{i''}C_{j''}C_{l''}$ 是所有不相交环三角增益乘积之和，依此类推。Δ_i 的定义与 Δ 类似，表示第 i 个前向路径上的所有状态，以及连接这些状态的分支，在计算 Δ_i 时都要去除掉。这样码字的重量枚举函数就为：

$$A(X)=\frac{\sum_i F_i\Delta_i}{\Delta} \qquad (4.74)$$

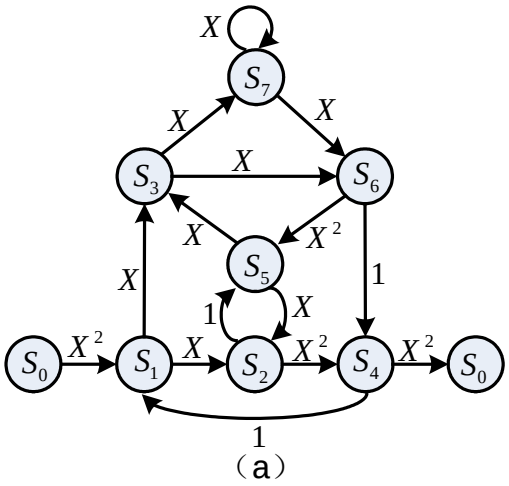
其中分子中的和是所有前向路径上的和。

＝＝

例 4.9：计算（2，1，3）卷积码的 WEF

考虑如图 4.1 所示的（2，1，3）非系统前馈编码器，其修正状态图如图 4.12(a)所示，在该图中共有 11 个环

Cycle 1: $S_1 S_3 S_7 S_6 S_5 S_2 S_4 S_1$ ($C_1=X^8$)
Cycle 2: $S_1 S_3 S_7 S_6 S_4 S_1$ ($C_2=X^3$)
Cycle 3: $S_1 S_3 S_6 S_5 S_2 S_4 S_1$ ($C_3=X^7$)
Cycle 4: $S_1 S_3 S_6 S_4 S_1$ ($C_4=X^2$)
Cycle 5: $S_1 S_2 S_5 S_3 S_7 S_6 S_4 S_1$ ($C_5=X^4$)
Cycle 6: $S_1 S_2 S_5 S_3 S_6 S_4 S_1$ ($C_6=X^3$)
Cycle 7: $S_1 S_2 S_4 S_1$ ($C_7=X^3$)
Cycle 8: $S_2 S_5 S_2$ ($C_8=X$)
Cycle 9: $S_3 S_7 S_6 S_5 S_3$ ($C_9=X^5$)
Cycle 10: $S_3 S_6 S_5 S_3$ ($C_{10}=X^4$)
Cycle 11: $S_7 S_7$ ($C_{11}=X$)



（a）

有 10 个不相交的环对：

Cycle Pair 1: (Cycle 2, Cycle 8) ($C_2C_8=X^4$)
Cycle Pair 2: (Cycle 3, Cycle 11) ($C_3C_{11}=X^8$)
Cycle Pair 3: (Cycle 4, Cycle 8) ($C_4C_8=X^3$)
Cycle Pair 4: (Cycle 4, Cycle 11) ($C_4C_{11}=X^3$)
Cycle Pair 5: (Cycle 6, Cycle 11) ($C_6C_{11}=X^4$)
Cycle Pair 6: (Cycle 7, Cycle 9) ($C_7C_9=X^8$)
Cycle Pair 7: (Cycle 7, Cycle 10) ($C_7C_{10}=X^7$)
Cycle Pair 8: (Cycle 7, Cycle 11) ($C_7C_{11}=X^4$)
Cycle Pair 9: (Cycle 8, Cycle 11) ($C_8C_{11}=X^2$)
Cycle Pair 10: (Cycle 10, Cycle 11) ($C_{10}C_{11}=X^5$)

有 2 个不相交的环三角：

Cycle Triple 1： (Cycle 4, Cycle 8, Cycle 11) ($C_4C_8C_{11}=X^4$)
Cycle Triple 2： (Cycle 7, Cycle 10, Cycle 11) ($C_7C_{10}C_{11}=X^8$)

除此之外，没有其他不相交环，因此

$$\begin{aligned}\triangle &= 1-(X^8+X^3+X^7+X^2+X^4+X^3+X^3+X+X^5+X^4+X) \\ &\quad +(X^4+X^8+X^3+X^3+X^4+X^8+X^7+X^4+X^2+X^5)-(X^4+X^8) \\ &= 1-2X-X^3 \end{aligned} \tag{4.75}$$

在图 4.12(a)中，有 7 个前向路径：

Foward Path 1：$S_0 S_1 S_3 S_7 S_6 S_5 S_2 S_4 S_0$ ($F_1=X^{12}$)
Foward Path 2：$S_0 S_1 S_3 S_7 S_6 S_4 S_0$ ($F_2=X^7$)
Foward Path 3：$S_0 S_1 S_3 S_6 S_5 S_2 S_4 S_0$ ($F_3=X^{11}$)
Foward Path 4：$S_0 S_1 S_3 S_6 S_4 S_0$ ($F_4=X^6$)
Foward Path 5：$S_0 S_1 S_2 S_5 S_3 S_7 S_6 S_4 S_0$ ($F_5=X^8$)
Foward Path 6：$S_0 S_1 S_2 S_5 S_3 S_6 S_4 S_0$ ($F_6=X^7$)
Foward Path 7：$S_0 S_1 S_2 S_4 S_0$ ($F_7=X^7$)

前向路径 1 和前向路径 5 涵盖了图中的所有状态，因此与这些路径不相交的子图不能包含任

何状态，即有：

$$\triangle_1=\triangle_5=1 \qquad (4.76)$$

与前向路径 3、前向路径 6 不相交的子图如图 4.13（a）所示，为：

$$\triangle_3=\triangle_6=1-X \qquad (4.77)$$

与前向路径 2 不相交的子图如图 4.13（b）所示，为：

$$\triangle_2=1-X \qquad (4.78)$$

与前向路径 4 不相交的子图如图 4.13（c）所示，为：

$$\triangle_4=1-(X+X)+(X^2)=1-2X+X^2 \qquad (4.79)$$

与前向路径 7 不相交的子图如图 4.13（d）所示，为：

$$\triangle_7=1-(X+X^4+X^5)+(X^5)=1-X-X^4 \qquad (4.80)$$

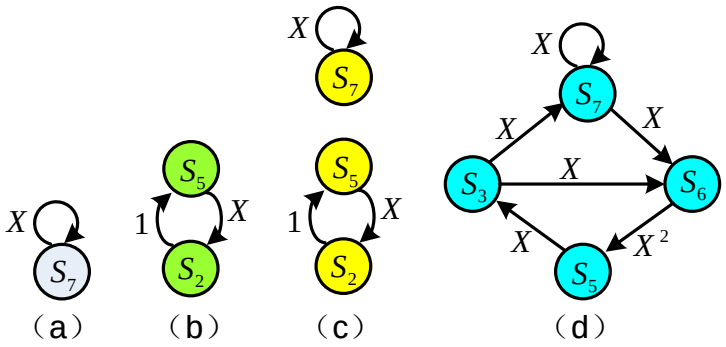


图 4.13 计算$\triangle_i$时的子图

则

$$A(X)=\frac{X^{12}\cdot 1+X^7(1-X)+X^{11}(1-X)+X^6(1-2X+X^2)}{+X^8\cdot 1+X^7(1-X)+X^7(1-X-X^4)}{1-2X-X^3}$$

$$=\frac{X^6+X^7-X^8}{1-2X-X^3} \qquad (4.81)$$

$$=X^6+3X^7+5X^8+11X^9+25X^{10}+\cdots$$

码字的 WEF 提供了所有从状态 S_0 分开、又回到状态 S_0 的所有非零码字的重量分布的详细描述，在这个式子中，我们知道，有 1 个码字重量为 6、3 个码字重量为 7、5 个码字重量为 8，依此类推。

==========

在修正状态图中，如果对每个分支用W^{ω}来标识对应的非零信息，其中ω表示该分支上 k 信息 bit 的重量；每经过一个分支标识一个 L 因子，则该码的码字输入－输出重量枚举函数（IOWEF, Input-Output Weight Enumerating Function）表示为：

$$A(W,X,L)=\sum_{\omega,d,l}A_{\omega,d,l}W^{\omega}X^{d}L^{l} \qquad (4.82)$$

其中因子$A_{\omega,d,l}$表示具有重量 d、信息重量为ω、的经过 l 个分支的码字数。这样图 4.1 所示编码器的增加修正状态图如图 4.14 所示。

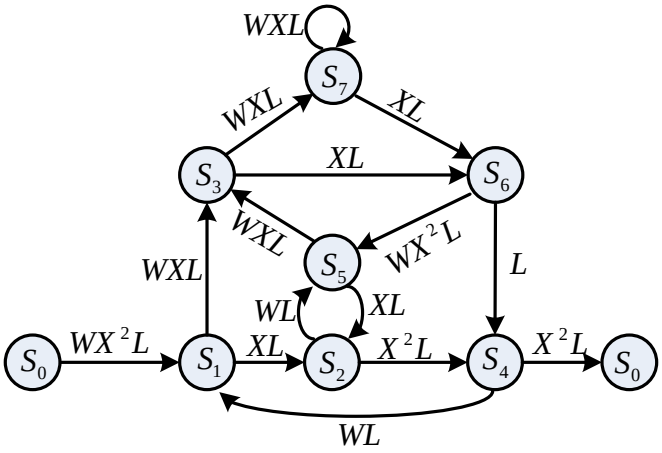


图 4.14 （2，1，3）编码器的增加修正状态图

＝＝

例 4.10：对于图 4.14 所示状态图，我们可得到：

$$\begin{aligned}\triangle=1-&(X^8W^4L^7+X^3W^3L^5+X^7W^3L^6+X^2W^2L^4+X^4W^4L^7+X^3W^3L^6\\&+X^3WL^3+XWL^2+X^5W^3L^4+X^4W^2L^3+XWL)+(X^4W^4L^7\\&+X^8W^4L^7+X^3W^3L^6+X^3W^3L^5+X^4W^4L^7+X^8W^4L^7+X^7W^3L^6\\&+X^4W^2L^4+X^2W^2L^3+X^5W^3L^4)-(X^4W^4L^7+X^8W^4L^7)\\=1-&XW(L+L^2)-X^2W^2(L^4-L^3)-X^3WL^3-X^4W^2(L^3-L^4)\end{aligned}\tag{4.83}$$

以及

$$\begin{aligned}\sum_i F_i\Delta_i &= X^1W^4L^8\cdot 1+X^7W^3L^6(1-XWL^2)+X^1W^3L^7(1-XWL)\\&+X^6W^2L^5\left[1-XW\left(L+L^2\right)+X^2W^2L^3\right]+X^8W^4L^8\cdot 1\\&+X^7W^3L^7(1-XWL)+X^7WL^4(1-XWL-X^4W^2L^3)\\&=X^6W^2L^5+X^7WL^4-X^8W^2L^5\end{aligned}\tag{4.84}$$

因此，码字 IOWEF 为：

$$\begin{aligned}A(W,X,L)&=\frac{X^6W^2L^5+X^7WL^4-X^8W^2L^5}{1-XW(L+L^2)-X^2W^2(L^4-L^3)-X^3WL^3-X^4W^2(L^3-L^4)}\\&=X^6W^2L^5+X^7(WL^4+W^3L^6+W^3L^7)\\&\quad+X^8(W^2L^6+W^4L^7+W^4L^8+2W^4L^9)+\cdots\end{aligned}\tag{4.85}$$

这意味着重量为 6 的码字有 1 个，它经过了 5 个分支，输入的信息重量为 2；重量为 7 的码字有 3 个，经过 4 个分支的输入信息重量为 1，经过 6 个分支的输入信息重量为 3，经过 7 个分支的输入信息重量为 3，依此类推。

＝＝

在 $A(W,X,L)$ 中的 WL^{v+1} 项对应于输入重量为 1，它表示这是通过增加修正状态图的最短路径，但应注意，它并不是输出码字的最小重量路径。如在例 4.10 中，项 X^7WL^4 表示最短路径，但项 $X^6W^2L^5$ 表示最小重量路径。

还应注意的是：WEF A(X)是码（code）的特性，对不同编码器表示它是不变的；而 IOWEF A(W,X,L)是编码器特性，它依赖于输入序列和码字（codeword）之间的映射关系。

IOWEF 的另一个表示方式是只包含输入和输出重量的信息，而不包括路径信息，即将 A(W,X,L)中的 L 变量设为 1：

$$A(W,X)=\sum_{\omega,d}A_{\omega,d}W^{\omega}X^{d}=A(W,X,L)\big|_{L=1} \qquad (4.86)$$

其中 $A_{\omega,d}=\sum_{l}A_{\omega,d,l}$ 表示输出序列重量为 d、输入信息重量为 ω 的码字数目。

类似地，从 IOWEF 也能够得到 WEF A(X)：

$$A(X)=\sum_{d}A_{d}X^{d}=A(W,X)\big|_{W=1}=A(W,X,L)\big|_{W=L=1} \qquad (4.87)$$

其中 $A_{d}=\sum_{\omega}A_{\omega,d}$ 。

在计算 Turbo 码的 WEF 时需要一个重要工具：码字***条件重量枚举函数***（CWEF，Conditional Weight Enumerating Function），它列举出与特定信息重量相关的所有码字重量。当信息重量为 ω 时，CWEF 为：

$$A_{\omega}(X)=\sum_{d}A_{\omega,d}X^{d} \qquad (4.88)$$

直接从式（4.86），IOWEF 可表示为：

$$A(W,X)=\sum_{\omega}W^{\omega}A_{\omega}(X) \qquad (4.89)$$

很显然，CWEF 也是编码器的特性。

＝＝

例 4.11：式（4.85）简化的 IOWEF 变为：

$$A(W,X)=A(W,X,L)\big|_{L=1}=W^{2}X^{6}+(W+2W^{3})X^{7}+(W^{2}+4W^{4})X^{8}+\cdots \qquad (4.90)$$

表明有一个码字输出序列重量为 6，此时的输入信息重量为 2；有 3 个码字输出序列重量为 7，其中 1 个输入信息重量为 1，2 个输入信息重量为 3，依此类推。这样可得到 WEF 为：

$$A(X)=A(W,X)\big|_{W=1}=X^{6}+3X^{7}+5X^{8}+\cdots \qquad (4.91)$$

这与前面式（4.81）计算的一致!!

为了计算这种编码器的 CWEF，在状态图中，我们必须找到从初始态到终止态与给定重量信息序列相关的输出序列重量。例如，在图 4.14 中，有 1 个重量为 1 的信息序列 **u**＝（1000）；有 3 个重量为 2 的信息序列 **u**＝（11000），（101000）以及（1001000）；有 9 个重量为 3 的信息序列 **u**＝（111000），（1101000），（1011000），（(10101000），（11001000），（10011000），（101001000），（100101000）以及（1001001000），等等（信息序列中最后 3 个 0 表示清空寄存器）。由此，我们可得到：

$A_1(X)=X^7$ （4.92a）

$A_2(X)=X^6+X^8+X^{10}$ （4.92b）

$A_3(X)=2X^7+3X^9+3X^{11}+X^{13}$ （4.92c）

利用式（4.89），我们可得到：

$$A(W,X)=\sum_{\omega}W^{\omega}A_{\omega}(X)$$
$$=WX^7+W^2(X^6+X^8+X^{10})+W^3(2X^7+3X^9+3X^{11}+X^{13})+\cdots \quad (4.93)$$

由于计算的方法不同，所以式（4.90）和式（4.93）不容易一眼看出是相同的，但如果将式（4.85）的分子和分母按照 W 的升序重新排列，并将 L 设为 1，我们可得到：

$$A(W,X)=\frac{WX^7+W^2(X^6-X^8)}{1-W(2X+X^3)}$$
$$=WX^7+W^2(X^6+X^8+X^{10})+W^3(2X^7+3X^9+3X^{11}+X^{13})+\cdots \quad (4.94)$$

这样结果就和式（4.93）一致了！

==========

在截断编码器中，输入序列长度限制为 $\lambda=h+m$ 个 block，即包含 h 个信息 block，m 个终止 block。为了适应这种变化，我们将 IOWEF A(W,X,L)的分子分母按照 L 的升序排列，并在最后的结果中将大于 L^{λ} 的项忽略掉。

==========

例 4.12：我们将式（4.85）重写为：

$$A(W,X,L)=\frac{L^4WX^7+L^5W^2(X^6-X^8)}{1-LWX-L^2WX-L^3\left[WX^3+W^2(X^4-X^2)\right]-L^4W^2(X^2-X^4)} \quad (4.95)$$

设编码器在输入 $\lambda=8$ 个 block 后就终止，即只考虑 h=5 个输入信息 block 和 m=3 个终止 block，并在最后的结果中去除掉阶数大于 L^8 的项，有：

$$A(W,X,L)=L^4WX^7+L^5W^2X^6+L^6(W^2X^8+W^2X^8)+L^7[W^2X^{10}+W^3(X^7+X^{11})$$
$$+W^4X^8]+L^8[3W^3X^9+W^4(X^8+X^{10}+X^{12})+W^5X^9] \quad (4.96)$$

在上式中，只有 15 项，这是因为我们只考虑第一个比特是 1 的信息序列，且不考虑信息序列如 **u**=（10001000），因为它离开和回到状态 S_0 两次。注意：W^5 是最大输入重量，因为 $\lambda-m=h=5$。

现在我们计算该截短编码器的简化 IOWEF、条件重量枚举函数（CWEF）以及重量枚举函数（WEF）：

$$A(W,X)=A(W,X,L)|_{L=1}=WX^7+W^2(X^6+X^8+X^{10})+W^3(2X^7+3X^9+X^{11})$$
$$+W^4(2X^8+X^{10}+X^{12})+W^5X^9 \quad (4.97a)$$

$$A_1(X)=X^7 \quad (4.97b)$$

$$A_2(X)=X^6+X^8+X^{10} \quad (4.97c)$$

$$A_3(X)=2X^7+3X^9+X^{11} \quad (4.97d)$$

$$A_4(X)=2X^8+X^{10}+X^{12} \quad (4.97e)$$

$$A_5(X)=X^9 \quad (4.97f)$$

$$A(X)=A(W,X)|_{W=1}=X^6+3X^7+3X^8+4X^9+2X^{10}+X^{11}+X^{12} \qquad (4.97g)$$

注意：在式（4.97d）中的 CWEF $A_3(X)$ 与式（4.92c）中的表达式不同，这是由于在未截短编码器中，重量 $\omega=3$ 的信息序列长度超过 h＝5 比特，因此在截短编码器中，它们就不是有效输入序列了。

＝＝

以上计算码字 WEFs 的方法中只考虑离开和回到全 0 态一次的那些非零码字，这是由于在用最大似然译码（MLD）计算 BER 时所需。

为了确定信息 BER（ML 译码），可将码字 WEFs 表达略加改变，形成比特 WEFs，用来表示与给定重量的码字相关的非零信息比特数。对于一个重量为 ω 的信息，比特 CWEF 定义为：

$$B_\omega(X)=\sum_d B_{\omega,d}X^d \qquad (4.98)$$

其中 $B_{\omega,d}=(\omega/k)A_{\omega,d}$，其含义表示输出序列非零比特总数除以每个时间单元输入的信息比特数 k（重量为 d 的码字序列是由重量为 ω 的信息序列产生的）。

比特 IOWEF 可表示为：

$$B(W,X)=\sum_{\omega,d}B_{\omega,d}W^\omega X^d=\sum_\omega W^\omega B_\omega(X) \qquad (4.99)$$

比特 WEF 为：

$$B(X)=\sum_d B_d X^d=B(W,X)|_{W=1} \qquad (4.100)$$

其中 $B_d=\sum_\omega B_{\omega,d}$。

这三个比特重量枚举函数 $B_\omega(X)$、$B(W,X)$ 以及 $B(X)$ 都是编码器的特性。

从 $A_{\omega,d}$ 和 $B_{\omega,d}$ 的定义可知，对任意 ω 和 d，有：

$$B_d=\sum_\omega B_{\omega,d}=\sum_\omega(\omega/k)A_{\omega,d} \qquad (4.101)$$

上式可用于建立码字 IOWEF A（W，X）和比特 WEF B（X）之间的关系：

$$\begin{aligned}B(X)&=\sum_d B_d X^d=\sum_{\omega,d}(\omega/k)A_{\omega,d}X^d\\&=\frac{1}{k}\cdot\left.\frac{\partial\left[\sum_{\omega,d}A_{\omega,d}W^\omega X^d\right]}{\partial W}\right|_{W=1}=\frac{1}{k}\cdot\left.\frac{\partial A(W,X)}{\partial W}\right|_{W=1}\end{aligned} \qquad (4.102)$$

也就是说，比特 WEF B（X）可以直接从码字 IOWEF A（W，X）计算得到。

在译码时可以看到，最大似然译码进行的判决在每个时间单元是 k 个信息比特的译码，因此，为了确定每个信息比特的译码错误概率，我们这里定义的比特 WEFs 中包含了一个 1/k 因子。

＝＝

例 4.13：

对于（2，1，3）编码器（我们多次举例用的编码器），将式（4.102）应用到式（4.90）所示的码字 IOWEF 中，得到比特 WEF：

$$
\begin{aligned}
B(X) &= \frac{1}{k}\cdot\frac{\partial A(W,X)}{\partial W}\bigg|_{W=1} \\
&= \frac{\partial\left[W^2X^6+(W+2W^3)X^7+(W^2+4W^4)X^8+\cdots\right]}{\partial W}\bigg|_{W=1} \\
&= 2X^6+7X^7+18X^8+\cdots
\end{aligned}
\qquad (4.103)
$$

上式中 B（X）的因子表示不同输出序列重量多对应的非零输入信息比特的总数。

如：$2X^6$，表示重量为 6 的输出序列对应着非零输入信息比特数为 2，从状态图中可看出这一点。重量为 6 的输出序列只有一个支路，为 S0→S1→S3→S6→S4→S0，如下图（a）所示，所以对应的非零输入信息比特数（W 的指数）为 2。

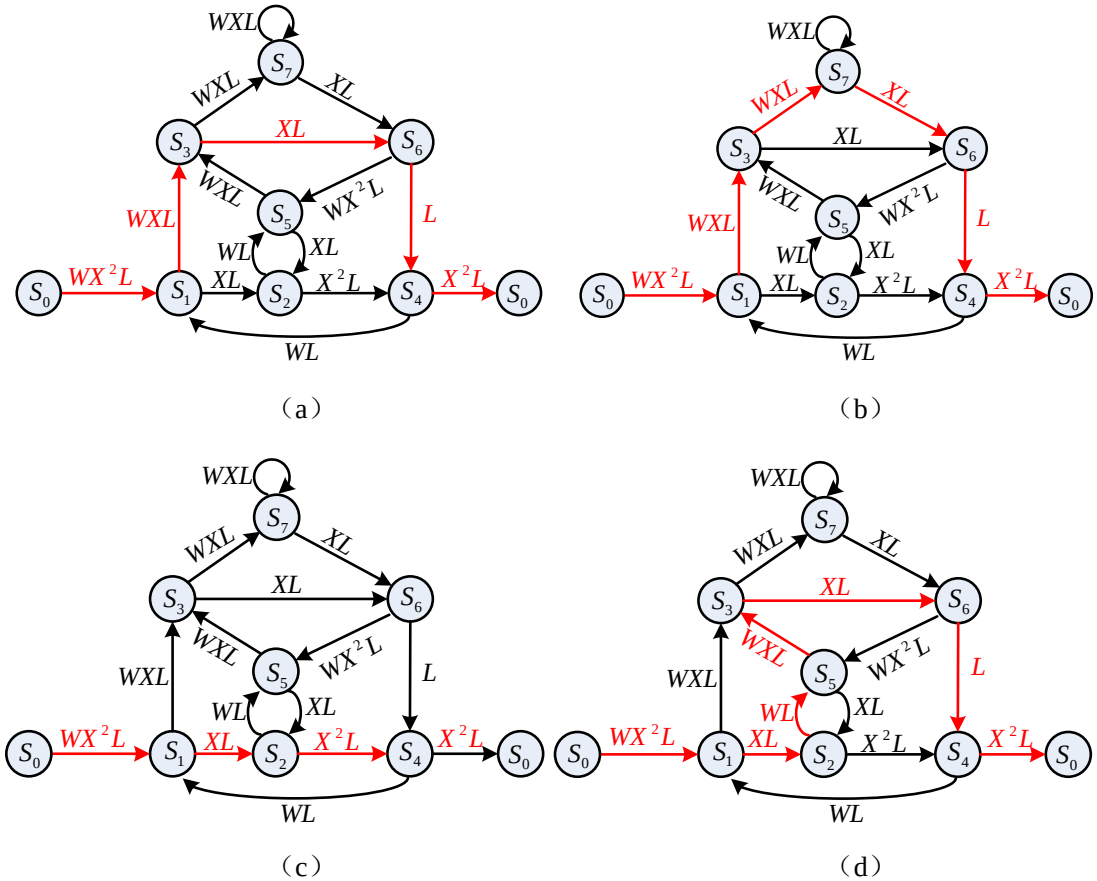


图 4.15 特定重量输出序列对应的支路

重量为 7 的输出序列只有 3 个支路，如上图（b）、（c）、（d）所示，所对应的非零输入信息比特数（W 的指数）为 3＋1＋3＝7。

4.3 卷积码的距离特性

卷积码的性能取决于所用的译码算法以及该码的距离特性，本节我们介绍几种距离度量方法，而它们与性能的关系在译码时再介绍。

对卷积码来说，最重要的距离度量是最小自由距离 d_{free}。

定义 4.6：卷积码的最小自由距离定义为：

$$d_{free} \triangleq \min_{\mathbf{u}',\mathbf{u}''}\left\{d\left(\mathbf{v}',\mathbf{v}''\right):\mathbf{u}'\neq\mathbf{u}''\right\} \tag{4.104}$$

其中 $\mathbf{v}'$ 和 $\mathbf{v}''$ 是码字，分别对应于输入序列 $\mathbf{u}'$ 和 $\mathbf{u}''$。

在上式中，假设了码字 $\mathbf{v}'$ 和 $\mathbf{v}''$ 具有有限长度，且是从全 0 态 S_0 开始、在全 0 态 S_0 结束，即会在信息序列 $\mathbf{u}'$ 和 $\mathbf{u}''$ 后附加 m 个终止 block。如果 $\mathbf{u}'$ 和 $\mathbf{u}''$ 长度不同，就在短序列后补 0，使得对应的码字序列有相同的长度。因此，d_{free} 是该码集合中任意两个有限长度码字的最小距离。

由于卷积码是线性码，有：

$$\begin{aligned} d_{free} &= \min_{\mathbf{u}',\mathbf{u}''}\left\{\omega\left(\mathbf{v}'+\mathbf{v}''\right):\mathbf{u}'\neq\mathbf{u}''\right\} \\ &= \min_{\mathbf{u}}\left\{\omega\left(\mathbf{v}\right):\mathbf{u}\neq\mathbf{0}\right\} \\ &= \min_{\mathbf{u}}\left\{\omega\left(\mathbf{uG}\right):\mathbf{u}\neq\mathbf{0}\right\} \end{aligned} \tag{4.105}$$

其中 $\mathbf{v}$ 是输出码字序列，对应于输入信息序列 $\mathbf{u}$。因此，d_{free} 是由有限长输入信息序列产生的输出序列中的最小重量。在状态图中，即为从全 0 态离开、又回到全 0 态的有限长路径的最小输出重量，因此是 WEF A（X）中的最低阶数。如对于例 4.1 中的（2，1，3）编码器，其 $d_{free}=6$，例 4.2 中的（3，2，2）编码器，其 $d_{free}=3$。

卷积码另一个重要的距离度量是列距离函数（CDF，Column Distance Function）：d_l。设

$$[\mathbf{v}]_l = \left(v_0^{(0)}, v_0^{(1)}\cdots v_0^{(n-1)}, v_1^{(0)}, v_1^{(1)}\cdots v_1^{(n-1)}\cdots v_l^{(0)}, v_l^{(1)}\cdots v_l^{(n-1)}\right) \tag{4.106}$$

表示码字 $\mathbf{v}$ 的 l 阶切断。设

$$[\mathbf{u}]_l = \left(u_0^{(1)}, u_0^{(2)}\cdots u_0^{(k)}, u_1^{(1)}, u_1^{(2)}\cdots u_1^{(k)}\cdots u_l^{(1)}, u_l^{(2)}\cdots u_l^{(k)}\right) \tag{4.107}$$

表示输入信息序列 $\mathbf{u}$ 的 l 阶切断。

定义 4.7：l 阶的列距离函数（d_l）定义为：

$$\begin{aligned} d_l &\triangleq \min_{[\mathbf{u}']_l,[\mathbf{u}'']_l}\left\{d\left([\mathbf{v}']_l,[\mathbf{v}'']_l\right):[\mathbf{u}']_0\neq[\mathbf{u}'']_0\right\} \\ &= \min_{[\mathbf{u}]_l}\left\{\omega\left([\mathbf{v}]_l\right):[\mathbf{u}]_0\neq\mathbf{0}\right\} \end{aligned} \tag{4.108}$$

因此，d_l 是前（l+1）个单位时间输入为非 0 信息序列时，输出码字的最小重量。

根据该码的生成矩阵，有：

$$[\mathbf{v}]_l = [\mathbf{u}]_l[\mathbf{G}]_l \tag{4.109}$$

其中$[\mathbf{G}]_l$ 是 k(l+1)×n(l+1)的矩阵，是生成矩阵 $\mathbf{G}$（式 4.22）的前 n(l+1)列，表示为：

$$[\mathbf{G}]_l=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_l\\ & \mathbf{G}_0 & \cdots & \mathbf{G}_{l-1}\\ & & \ddots & \vdots\\ & & & \mathbf{G}_0\end{bmatrix},\quad l\le m \qquad (4.110a)$$

$$[G]_l=\begin{bmatrix}\mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m & & & & \\ & \mathbf{G}_0 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m & & & \\ & & \ddots & \vdots & \vdots & \vdots & \ddots & & \\ & & & \mathbf{G}_1 & \mathbf{G}_2 & \mathbf{G}_3 & \cdots & \mathbf{G}_m & \\ & & & \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_{m-1} & \mathbf{G}_m \\ & & & & \mathbf{G}_0 & \mathbf{G}_1 & \cdots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} \\ & & & & & \mathbf{G}_0 & \vdots & \vdots & \vdots \\ & & & & & & \ddots & \mathbf{G}_1 & \mathbf{G}_2 \\ & & & & & & & \mathbf{G}_0 & \mathbf{G}_1 \\ & & & & & & & & \mathbf{G}_0 \end{bmatrix},\quad l>m \qquad (4.110b)$$

$$d_l=\min_{[\mathbf{u}]_l}\left\{\omega\left([\mathbf{u}]_l[\mathbf{G}]_l\right):\ [\mathbf{u}]_0\neq\mathbf{0}\right\} \qquad (4.111)$$

由此可看出，它只依赖于 G 的前 n(l+1)列，这也是“Column Distance Function”名字的由来。

这里应注意，式（4.110）[以及式（4.22）]所示的生成矩阵 $\mathbf{G}$ 的形式都假设是一个前馈编码器实现，即多项式生成矩阵有 m 阶的存储器。对于反馈编码器，其生成矩阵是一个有理函数，相关的生成序列具有无限长周期，k×n 的子矩阵 $\mathbf{G}_l$ 就会在式（4.110）中向右无限扩展。此时，CDF 仍然用式（4.111）表示。

定义 4.7 意味着 d_l 不随 l 的增加而下降，即它是 l 的单调非降函数。计算 d_l 一个有效方法是利用序贯译码算法进行计算。例如（2，1，16）非系统码，$\mathbf{G}(D)=[1+D+D^2+D^5+D^6+D^8+D^{13}+D^{16},\ 1+D^3+D^4+D^7+D^9+D^{10}+D^{11}+D^{12}+D^{14}+D^{15}+D^{16}]$，其 CDF 如图 4.16 所示。

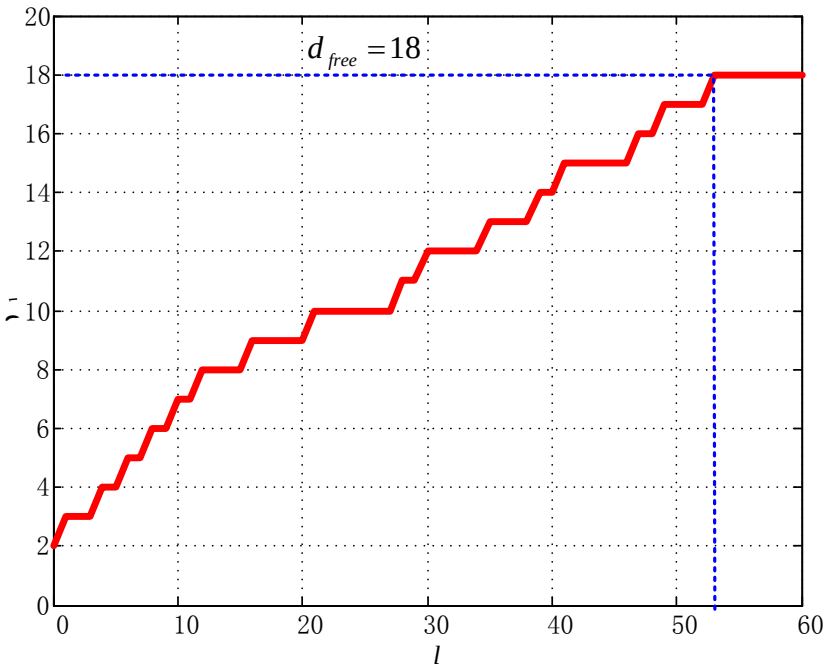


图 4.16 （2，1，16）码的列距离函数

考虑两种特定情况：$l=m$ 以及 $l\to\infty$。对于 $l=m$，d_m 称为最小距离，因此 d_m 也常表示为 d_{min}。从式（4.111）我们可看出，d_{min} 表示前（m+1）个单位时间输入为非 0 信息序列时，输出码字的最小重量。例如，在图 4.16 中，$d_{min}=d_{16}=9$。早期研究卷积码的时候，都把 d_{min} 作为最重要的一个参数，因为那时的译码技术（如多数逻辑译码，majority-logic decoding）需要（m+1）个单位时间的译码存储，而现在主流的译码技术主要是最大似然（ML）译码、最大后验概率（MAP）译码，因此 d_{free} 和 CDF 取代了 d_{min} 作为距离参数，因为这些译码算法对译码存储没有限制。

当 $l\to\infty$ 时，

$$\lim_{l\to\infty} d_l = d_{free} \qquad (4.112)$$

因此，d_l 最终会达到 d_{free}，并保持恒定。【一般而言，当 l 取值到 $3v$ 左右，就会达到 d_{free}。】