实验七预习材料

一、信道编译码

1.1 基本原理

信道编码，就是在待发送的信息序列中人为地按一定规则加入保护成分（监督码元），然后将构成的新序列作为发送序列；接收端的信道译码器按相应逆规则进行译码，从中发现错误或纠正错误。从信息传输的角度来看，监督码元是冗余的，这种冗余度降低了信息传输的效率；但从系统可靠性角度来看，冗余度的增加带来了检（纠）错能力的提升，增强了数字信号的抗干扰能力。因此，信道编码又称差错控制编码。

设编码后的发送序列共包含 n 个码元，其中信息码元数为 k，监督码元数为 $(n-k)$。信息码元数与总码元数的比值 k/n 称为编码效率（码率），监督码元数与信息码元数的比值 $(n-k)/k$ 称为冗余度。一般来说，冗余度越大，信息抗干扰能力越强，系统可靠性越高。信道编码的本质即通过降低信息传输速率（有效性）来提高信息的抗干扰能力（可靠性）。

1.2 分类

信道编码一般分为两类：分组编码和卷积编码。

在分组编码中，信息序列被划分为若干个长度为 k 的码组，每个信息码组按一定的编码规则映射成长度为 n 的发送码组。各码组的映射关系是独立的，编码器的输出仅与当前输入的 k 个信息码元有关，而与先前的序列无关。对于$(n，k)$分组码，码率定义为 k/n。常见分组码包括汉明码、循环码、BCH 码、LDPC 码等。

在卷积编码中，虽然也将长度为 k 的信息码组编成长度为 n 的发送码组，但其监督位不仅和当前输入的 k 个信息码元有关，同时也与前面$(N-1)$个信息码组有关，故一个码组中的监督码元监督着 N 个信息码组。对于$(n，k，N)$卷积码，称 N 为编码约束度，码率定义为 k/n。Turbo 码也属于卷积码。

二、卷积编码

2.1 基本原理

在分组码中，编码器产生的 n 位码元的一个码组，完全取决于这段时间中的 k 比特输入信息，该码组中的监督位仅监督本码组中的 k 比特信息，而与其他码组无关；而卷积码在编码时虽然也将 k 比特的信息段编成 n 比特的码组，但其监督位不仅和当前的 k 比特有关，同时也与前面 $m=(N-1)$个信息段有关，故一个码组中的监督码元监督着 N 个信息段。对于$(n，k，N)$卷积码，称 N 为编码约束

度，码率定义为 k/n。

卷积编码的一般原理框图如图 1 所示。编码器由三种主要元件构成，包括(N-1) k 级移存器、n 个模二加法器和一个旋转开关。每个模二加法器的输入端连接到若干移存器的输出端，输出端连接到旋转开关上。将时间分成等间隔的时隙，在每个时隙中有 k 比特待编码数据从左端进入移存器，并且各级移存器暂存的信息向右移 k 位。旋转开关每时隙旋转一周，输出 n 比特编码数据($n>k$) 。

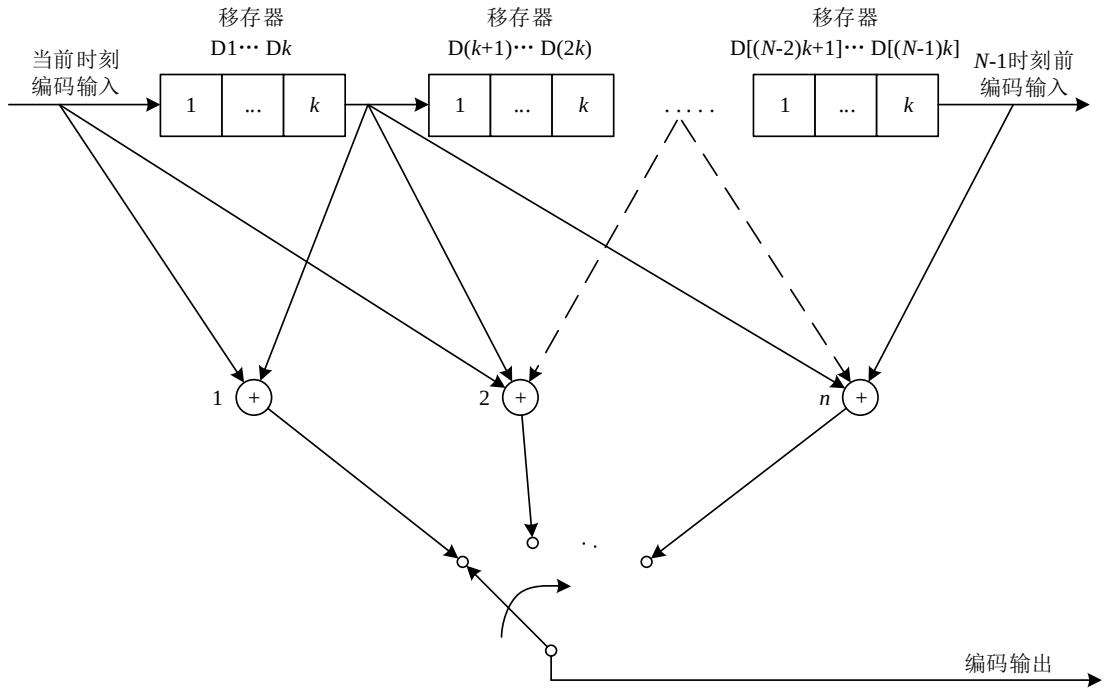

图 1 卷积编码一般原理框图

（2，1，3）卷积码编码器如图 2 所示。编码器包括 2 级移存器和 2 个模二加法器。编码前将各级移存器清零，待编码数据按照 $m_0^{(0)}m_1^{(0)}\ldots m_{i-1}^{(0)}m_i^{(0)}\ldots$的顺序依次输入编码器（下标表示码元在不同时隙输入/输出编码器，上标表示每个时隙输入/输出编码器的不同码元），输出码元 $c_i^{(0)}$与 $c_i^{(1)}$的计算公式见式（1）,式中⊕符号表示模二加法。每个时隙有一位待编码数据输入编码器，移存器中的数据依次右移，按式（1）计算两位编码数据输出。

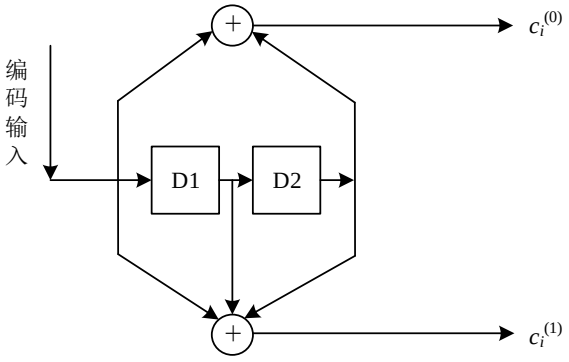

图 2 （2,1,3）卷积编码器原理框图

$$c_i^{(0)}=m_i^{(0)}\oplus m_{i-2}^{(0)}$$
$$c_i^{(1)}=m_i^{(0)}\oplus m_{i-1}^{(0)}\oplus m_{i-2}^{(0)} \quad (1)$$

2.2 描述方式

2.2.1 代数描述方式

卷积码的代数描述方式包括生成矩阵法、生成矢量法和生成多项式法。

生成矩阵法：

卷积码的一般原理框图如图 1 所示。在某个特定的时刻 i，由 k 比特码元组成的码组 $\mathbf{m}_i$ 输入到编码器中，相应的输出结果是由 n 比特码元组成的码组 $\mathbf{c}_i$。若输入序列用 $\mathbf{M}=(\mathbf{m}_0,\mathbf{m}_1,\mathbf{m}_2,\ldots\ldots)$表示，输出序列用 $\mathbf{C}=(\mathbf{c}_0,\mathbf{c}_1,\mathbf{c}_2,\ldots\ldots)$表示，则卷积码的编码过程可用如式（2）所示的矩阵形式表示，称 **G** 为该卷积码的生成矩阵。卷积码的生成矩阵与线性分组码的生成矩阵的定义相同，区别在于由于卷积编码的输入序列的长度是半无限的，生成矩阵也是一个半无限矩阵。

$$\mathbf{C}=\mathbf{MG} \quad (2)$$

生成矢量法：

另一种描述生成矩阵的方式是以 n 个矢量组合来表示，每个矢量包含了移位寄存器输出与某一模二加法器之间连接关系的信息。某矢量第 i 个元素为“1”表示相应第 i 级移存器输出与该模二加法器相连，反之为“0”表示该级移存器输出与该模二加法器不相连。

以（2，1，3）卷积编码器为例。由图 2 可知，第一个模二加法器与当前输入比特、第 2 级移存器输出相连，第二个模二加法器与当前输入比特、第 1、2 级移存器输出相连，由此可以得到生成矢量：

$$\mathbf{g}_0=[101]$$
$$\mathbf{g}_1=[111] \quad (3)$$

该生成矢量通常采用八进制方式表示为 $\mathbf{g}=(\mathbf{g}_0,\mathbf{g}_1)=(5,7)$。$\mathbf{g}_0$ 和 $\mathbf{g}_1$ 作为编码器的脉冲响应，当某一时刻的信息序列 **M** 输入编码器时，相应两个编码输出比特分别为：

$$c^{(0)}=\mathbf{M}*\mathbf{g}_0$$
$$c^{(1)}=\mathbf{M}*\mathbf{g}_1 \quad (4)$$

其中“*”表示卷积运算，对应编码输出序列 **C** 为 $c^{(0)}$与 $c^{(1)}$交织的结果：

$$\mathbf{C}=(\mathbf{c}_0,\mathbf{c}_1,\cdots)$$
$$=\left(c_0^{(0)},c_0^{(1)},c_1^{(0)},c_1^{(1)},\cdots\right) \quad (5)$$

根据式（4）可得输入码元、移存器输出与编码输出码元的关系如表 1 所示：

表 1 输入码元、移存器输出与编码输出码元的关系

当前移存器输出（D2D1）	输入码元 $m_i^{(0)}$	下一时刻移存器输出（D2D1）	编码输出（$c_i^{(0)}c_i^{(1)}$）
00	0	00	00
	1	01	11
01	0	10	01
	1	11	10
10	0	00	11
	1	01	00
11	0	10	10
	1	11	01

生成多项式法：

最后一类用于描述卷积码的代数方法称为生成多项式法，即将编码器中移位寄存器输出与模二加法器的连接关系以及输入、输出序列表示为延时算子 D 的多项式。输入序列 **M** 关于 D 的多项式可表示为：

$$M(D)=\mathbf{m}_0+\mathbf{m}_1D+\mathbf{m}_2D^2+\mathbf{m}_3D^3... \quad (6)$$

式中 D 表示移存器中一个存储单元的单位时延，D 的幂的次数表示相对于时间起点的单位时延数目，时间起点通常选在第一个比特进入的时刻。延时算子 D 的引入使时域中的输入序列 **M** 变为变换域中的多项式 M（D）。

用 D 表示移位寄存器输出与模二加法器之间的连接关系时，多项式第 i 项系数为“1”表示相应第 i 级移存器输出与模二加法器相连，反之为“0”表示该级移存器输出与模二加法器不相连。图 2 描述的（2,1,3）卷积编码器的生成多项式如下：

$$\begin{aligned} g_0(D)&=1+D^2 \\ g_1(D)&=1+D+D^2 \end{aligned} \quad (7)$$

卷积运算相当于在变换域中作乘法，因此式（4）可写成如下的输出变换：

$$\begin{aligned} c^{(0)}(D)&=M(D)g_0(D) \\ c^{(1)}(D)&=M(D)g_1(D) \end{aligned} \quad (8)$$

2.2.2 几何描述方式

卷积码的几何描述方式就是基于编码器状态转移的图形表示方法，包括树图法、状态图法和网格图法。

按照各级移存器的内容来表示某一时刻编码器的状态。图 2 显示，（2,1,3）卷积编码器共占用 2 级移存器，即有 2^2=4 个状态（移存器 D2 的内容表示最高位，D1 的内容表示最低位；S_0 表示 00 状态，S_1 表示 01 状态，S_2 表示 10 状态，S_3 表示 11 状态）。若编码器的当前状态为 S_m（各级移存器的内容为 $m_{i-2}^{(0)}\,m_{i-1}^{(0)}$），编码输入为 $m_i^{(0)}$，则下一时刻状态跳转至 S_n（各级移存器的内容为 $m_{i-1}^{(0)}\ m_i^{(0)}$），

编码输出 $c_i^{(0)}$与 $c_i^{(1)}$按照式（1）计算得到。

对于（2,1,3）卷积编码器，其部分树图如图 3 所示。码树的节点表示编码器的状态：当输入信息比特为 0 时，码树向上分支移动；当输入信息比特为 1 时，码树向下分支移动；各分支上的数字表示编码输出。随着信息序列不断地输入到编码器中，码树由上一级节点进入下一级节点，最终形成一条路径，组成该路径的各分支上所标记的两位编码输出组成输出序列。每一条路径表示一组确定的编码输入及其对应的编码输出：从根节点 S_0 出发，若第一个输入信息比特为 0，码树向上分支移动，编码输出 00 并转移至节点 S_0；若第一个输入信息比特为 1，码树向下分支移动，编码输出 11 并转移至节点 S_1。

以输入信息序列 **M**=(0 1 0 0 ……)为例，其编码路径如图 3 中粗实线所示，相应的编码输出序列为 **C**=(00 11 01 11 ……)。

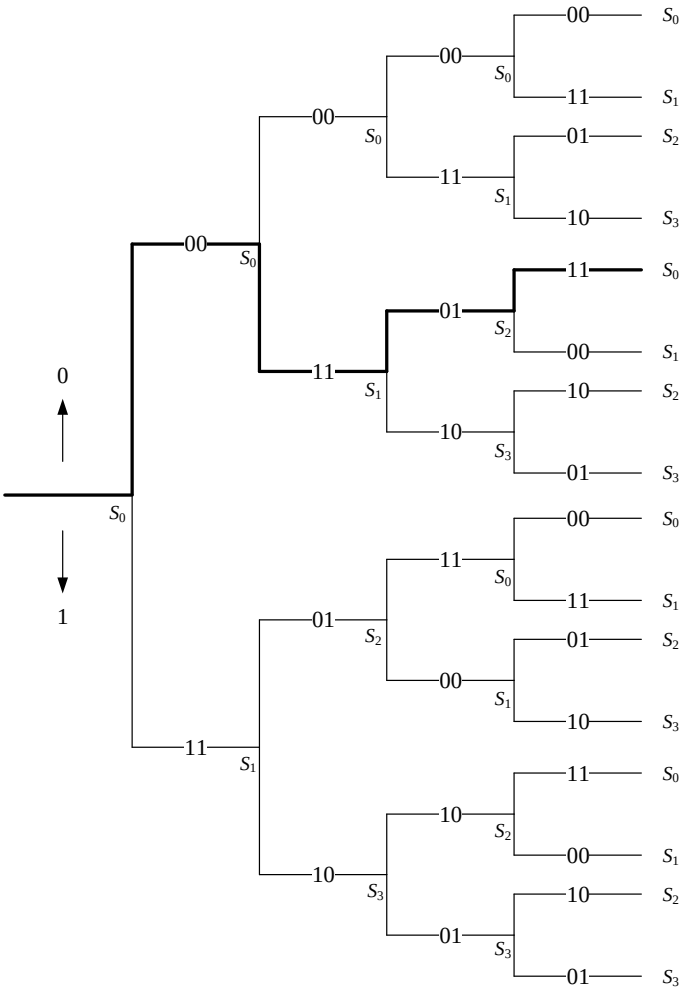

图 3 （2,1,3）卷积码部分树图

由上述树图可以看出，编码器当前时刻的状态与编码输入决定了编码器下一时刻的状态与编码输出，该编码器的状态转移关系如表 2 所示。

表 2 （2,1,3）卷积码状态转移关系

当前状态	输入 0		输入 1	
	下一状态	编码输出	下一状态	编码输出
S_0(00)	S_0	00	S_1	11
S_1(01)	S_2	01	S_3	10
S_2(10)	S_0	11	S_1	00
S_3(11)	S_2	10	S_3	01

按照表 2 的规律，可以画出状态转移图如图 4 所示。状态间的连线表示它们的转移情况，线上的数字表示编码输出，线型表示编码输入：实线表示输入数据为 0 时状态转移的路线，虚线表示输入数据为 1 时状态转移的路线。

将状态图在时间上展开，可以得到部分网格图如图 5 所示。

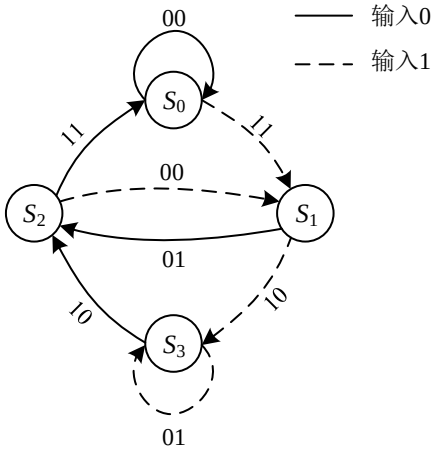

图 4 （2,1,3）卷积码状态转移图

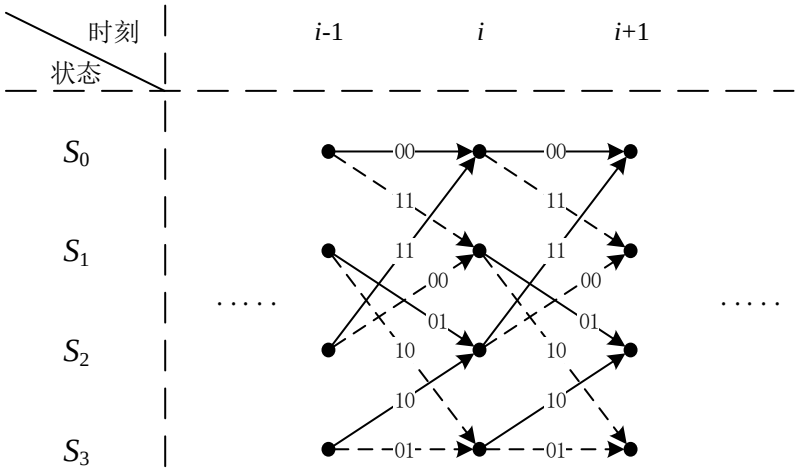

图 5 （2,1,3）卷积码部分网格图

三、Viterbi 译码

3.1 基本原理

Viterbi 译码是基于卷积码网格图的最大似然译码算法，其基本原理是将接收

到的信号序列与所有可能的发送信号序列比较，选择其中距离度量最小的序列作为发送信号序列。当接收序列长度为 L 时，有 2^L 种可能的路径；随着序列长度的增加，可能的路径数目以指数型增加。当发送序列较长时，译码过程需要消耗大量的时间；Viterbi 译码对此作了简化，即在接收一定长度的序列后，比较该段序列所对应路径的度量，保留其中累积度量最小的一条，淘汰其他路径，这大大减轻了译码计算量，提高了译码速率。

3.2 影响 Viterbi 译码性能的因素

3.2.1 判决方式

Viterbi 译码器的分支度量计算包括软判决与硬判决两种方式。

在硬判决方式中，当输入译码器的信号高于门限值时判为 1，低于门限值时判为 0，即以序列间的汉明距离作为度量值。该判决方式较简单，但无法保证门限值附近的电平判决的准确性；且仅有“0”和“1”两种汉明距离，不足以体现“正确”和“错误”的区别，通常只适合于二进制对称信道。

在软判决方式中，首先将信号经过信道的输出进行多电平量化（量化级＞2），再输入到 Viterbi 译码器中进行译码，即以欧氏距离作为度量值。量化级数越大，译码性能越好，该判决方式适合于离散无记忆信道。研究表明，采用软判决译码方式能使译码器增加近 3dB 增益。

3.2.2 译码深度

当接收序列长度为 L 时，有 2^L 种可能的路径，若 L 很大，存储所有路径信息是不现实的。Viterbi 译码采用截短译码的方式，即在接收一定长度的序列后，输出该段译码结果再进行下一阶段译码，该序列长度称为译码深度，记为 τ 。译码深度越大，译码器性能越好，但时延增加。对于(n，k，N)卷积码，译码深度通常取 τ=5N ~10N 。

3.3 译码过程

以（2,1,3）卷积码为例说明 Viterbi 译码算法的原理。

由图 5 可知，（2,1,3）卷积码的网格图共有 4 个状态，对于每个时隙的当前状态，均有来自前一时刻不同状态的 2 条分支到达该当前状态。以 S_1 状态为例：在 i 时刻到达 S_1 状态的 2 条分支分别来自 i-1 时刻的 S_0 状态和 S_2 状态，编码输入均为 1，编码输出分别为 11、00。

Viterbi 译码示意图见图 6。在某一时刻，到达 S_p 状态的 2 条分支分别来自前一时刻的 S_i 状态和 S_j 状态，分别计算每条分支的编码输出与当前译码输入的度量值，对各分支度量与前一时刻的对应路径度量值求和，在到达同一状态的 2 条分支中保留累计度量最小的分支，称之为幸存路径并保存该路径信息，更新各路径的度量值。不断重复以上“相加、比较、选择”过程直至达到译码深度，输出译

码序列。

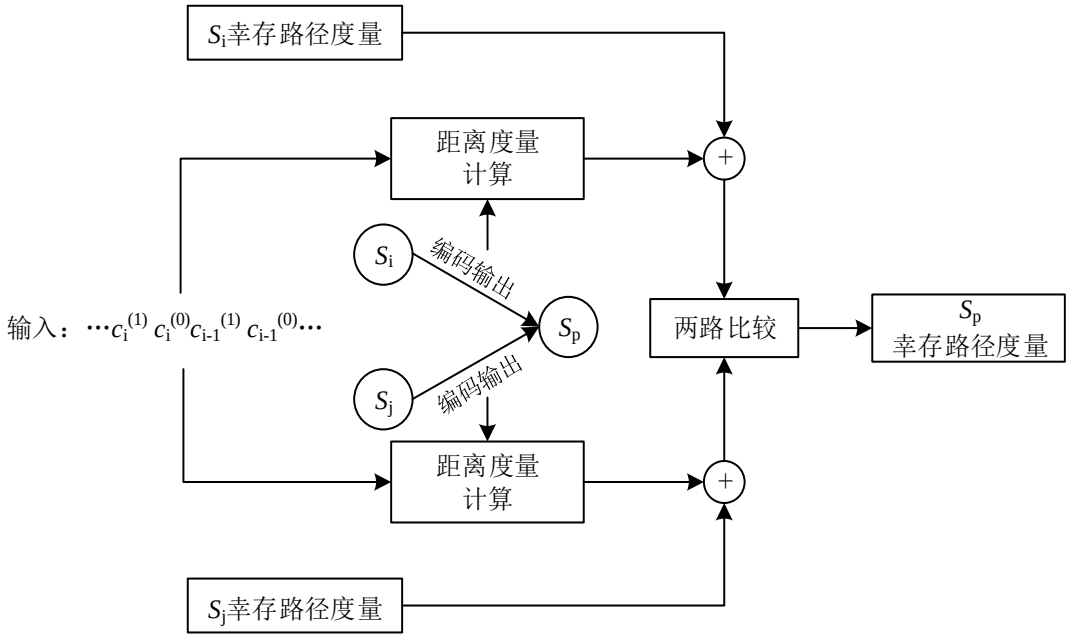

图 6 Viterbi 译码器基本结构框图

译码器的基本结构见图 7。分支度量计算单元用于计算前一时刻某一状态向该时刻某一新状态转移的分支度量，加比选单元对分支度量和此新状态先前的幸存路径度量求和，比较并保留累积度量最小的分支，更新幸存路径存储与路径度量存储，不断迭代至达到译码深度后回溯幸存路径，输出译码序列。

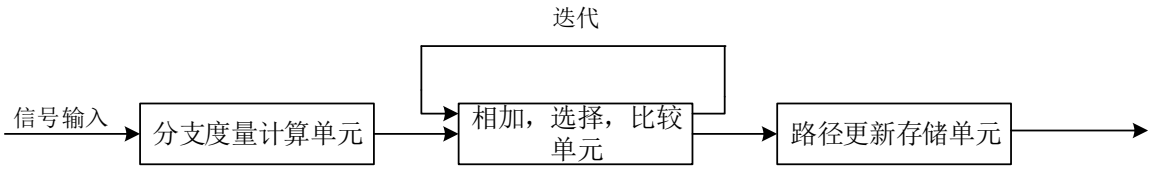

图 7 Viterbi 译码器的基本结构

3.4 （2，1，3）卷积码的 Viterbi 硬判决译码示例

假设输入编码器的序列为 1 1 0 1，为了使编码器在编码结束后回归 S_0 状态（即移存器中的信息位全部移出），在输入信息序列后添加 0 0 0；故编码后的发送序列为 11 10 10 00 01 11 00，假设接收到的序列为 11 11 10 00 01 11 10，其中第 4 个码元与第 13 个码元为错码。

从 S_0 状态出发，t=0 至 t=2 时刻共生成 4 条互不相交的路径分别到达 4 个状态，如图 8 所示；对各状态分别计算到达该状态的路径与译码输入的汉明距离度量值，计算结果如表 4 所示。

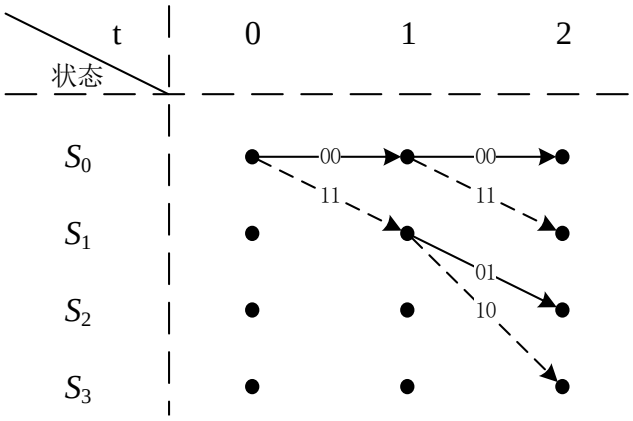

图 8 t=0 至 t=2 时刻的网格图

表 4 各状态初始度量值计算

到达状态	路径	对应路径编码输出	译码输入	路径度量
S_0	$S_0S_0S_0$	00 00	11 11	4
S_1	$S_0S_0S_1$	00 11		2
S_2	$S_0S_1S_2$	11 01		1
S_3	$S_0S_1S_3$	11 10		1

接下来以 t=2 时刻为起点，t=2 至 t=3 时隙中从 4 个状态出发的 8 条分支如图 9 所示：分别计算每条分支与译码输入的度量值，对各分支度量与前一时刻的对应路径度量值求和，在到达同一状态的 2 条分支中保留累计度量最小的分支（此处假设当 2 条分支累计度量值相同时保留上分支），称之为幸存路径并保存该路径信息，更新各路径的度量值。t=2 时刻至 t=3 时刻计算结果如表 5 所示，幸存路径网格图如图 10 所示。

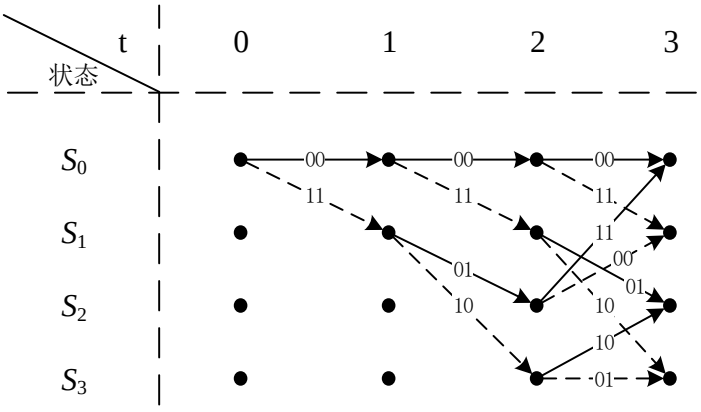

图 9 t=0 至 t=3 时刻的网格图

表 5 第一次“相加、比较、选择”过程计算结果

到达状态	路径	幸存路径度量	新增分支	对应分支编码输出	译码输入	分支度量	累计度量	是否幸存
S_0	$S_0S_0S_0+S_0$	4	S_0S_0	00	10	1	5	否
	$S_0S_1S_2+S_0$	1	S_2S_0	11		1	2	是
S_1	$S_0S_0S_0+S_1$	4	S_0S_1	11		1	5	否
	$S_0S_1S_2+S_1$	1	S_2S_1	00		1	2	是
S_2	$S_0S_0S_1+S_2$	2	S_1S_2	01		2	4	否
	$S_0S_1S_3+S_2$	1	S_3S_2	10		0	1	是
S_3	$S_0S_0S_1+S_3$	2	S_1S_3	10		0	2	是
	$S_0S_1S_3+S_3$	1	S_3S_3	01		2	3	否

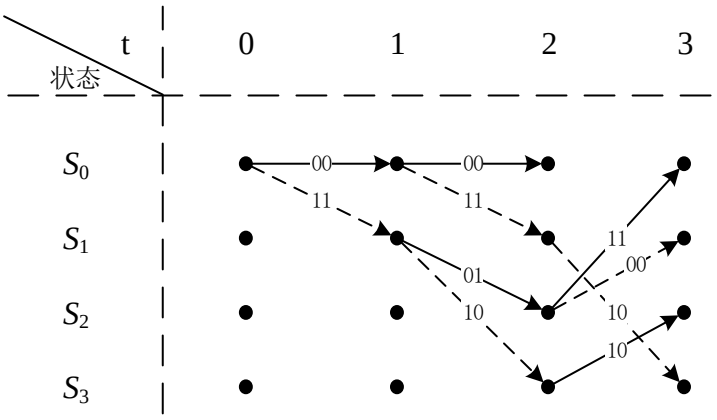

图 10 第一次“相加、比较、选择”后的幸存路径网格图

t=3 至 t=4 时隙中从 4 个状态出发的 8 条分支如图 11 所示；分别计算每条分支与译码输入的度量值，对各分支度量与前一时刻的对应路径度量值求和，在到达同一状态的 2 条分支中保留累计度量最小的分支（此处假设当 2 条分支累计度量值相同时保留上分支），称之为幸存路径并保存该路径信息，更新各路径的度量值。t=3 时刻至 t=4 时刻计算结果如表 6 所示，幸存路径网格图如图 12 所示。

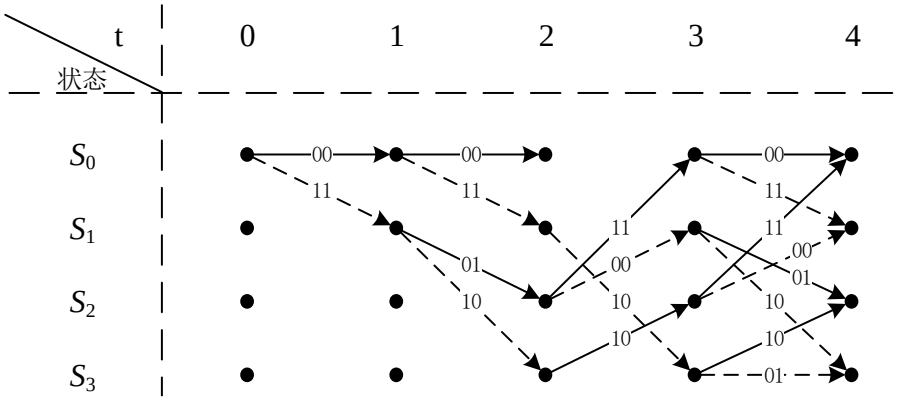

图 11 t=0 至 t=4 时刻的网格图

表 6 第二次“相加、比较、选择”过程计算结果

到达状态	路径	幸存路径度量	新增分支	对应分支编码输出	译码输入	分支度量	累计度量	是否幸存
S_0	$S_0S_1S_2S_0+S_0$	2	S_0S_0	00	00	0	2	是
	$S_0S_1S_3S_2+S_0$	1	S_2S_0	11		2	3	否
S_1	$S_0S_1S_2S_0+S_1$	2	S_0S_1	11		2	4	否
	$S_0S_1S_3S_2+S_1$	1	S_2S_1	00		0	1	是
S_2	$S_0S_1S_2S_1+S_2$	2	S_1S_2	01		1	3	是
	$S_0S_0S_1S_3+S_2$	2	S_3S_2	10		1	3	否
S_3	$S_0S_1S_2S_1+S_3$	2	S_1S_3	10		1	3	是
	$S_0S_0S_1S_3+S_3$	2	S_3S_3	01		1	3	否

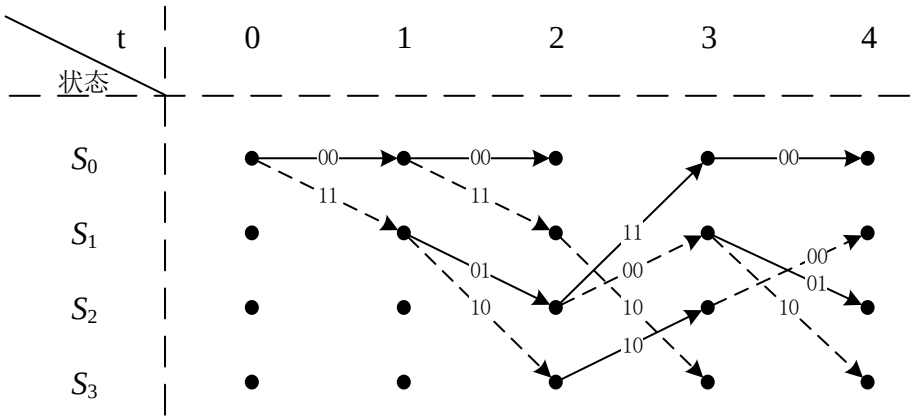

图 12 第二次“相加、比较、选择”后的幸存路径网格图

不断重复以上“相加、比较、选择”过程直至达到译码深度，输出译码序列。

将为了使编码器在编码结束后回归全零状态而在输入信息序列后添加的 0 0 0 可看成译码输入，按照上述步骤继续进行译码，得到的幸存路径网格图如图 13 所示。已知接收到额外添加的 0 0 0 码元后，路径回归全零状态 S_0，因此最终译码路径为图 13 中粗线所示，输出序列 1 1 0 1 0 0 0。

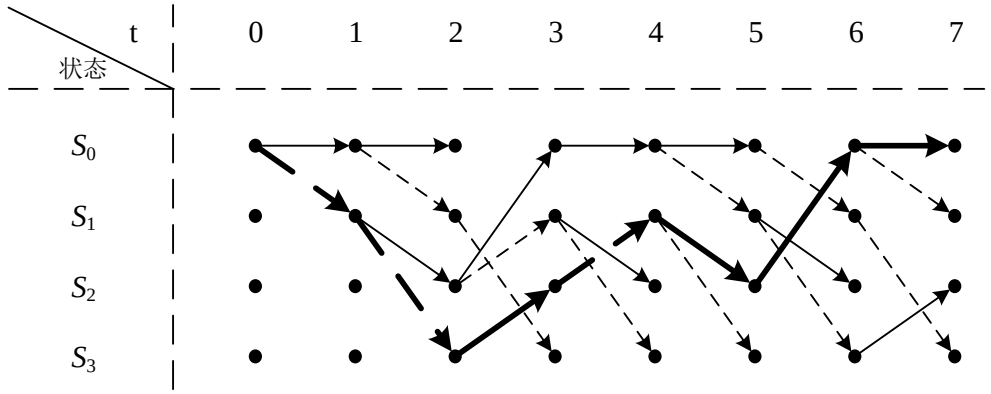

图 13 译码输入 1 1 0 1 0 0 0 时的幸存路径网格图