第 5 章　同步时序逻辑电路

同步时序逻辑电路是时序逻辑电路的一个主要分支，它与其他逻辑电路最大的区别在于电路状态改变方式是同步进行的。

本章首先介绍同步时序逻辑电路的一些基本概念，然后详细讨论同步时序逻辑电路的设计和分析方法，并对一些在数字系统尤其是计算机电路中常用的同步时序逻辑电路模块的基本原理进行讨论，最后给出同步时序逻辑电路的 VHDL 描述方法。

5.1　同步时序逻辑电路概述

5.1.1　同步时序逻辑电路模型及特点

1. 同步时序逻辑电路的模型

图 5-1 给出了同步时序逻辑电路的结构模型，它的存储元件使用边沿触发的触发器元件，所有触发器受统一时钟信号 CP 的控制，可保证电路所有的触发器能够在同一个时钟边沿同步地发生状态变化。

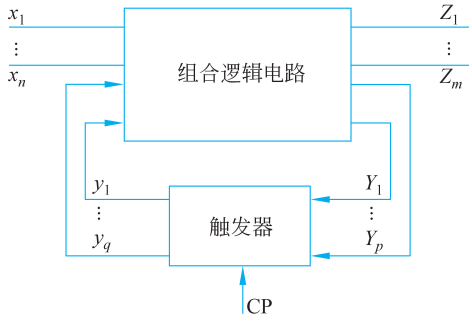

图 5-1　同步时序逻辑电路的模型

按照时序机的定义，同步时序逻辑电路由以下几部分组成。

（1）输入变量有限集合 $I=\{x_1,x_2,\cdots,x_n\}$，其中 n 个输入变量，有 2^n 种输入组合。

（2）输出变量有限集合 $O=\{Z_1,Z_2,\cdots,Z_m\}$。

（3）内部状态变量有限非空集合 $Q=\{y_1,y_2,\cdots,y_q\}$，其中 q 为状态变量的个数，共有 2^q 种状态组合。

（4）输出函数集合 $Z=\{Z_i=f_i(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $i=1,2,\cdots,m\}$。

当 Z 是输入和状态的函数（即 $I\times Q\rightarrow O$）时，该同步时序逻辑电路为 Mealy 型电路。

当 Z 仅是状态的函数（即 $Q\rightarrow O$）时，该同步时序逻辑电路为 Moore 型电路。

（5）次态函数集合 $N=\{y_j^{n+1}=g_j(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $j=1,2,\cdots,q\}$。

同步时序逻辑电路选用触发器作为存储元件，次态函数集合 N 分成了两部分。

① 触发器状态方程：$y_j^{n+1}=g_j(Y,y_j^n)$，$j=1,2,\cdots q$，其中 Y 是触发器的激励函数集合，也是所有激励变量$\{Y_1,Y_2,\cdots,Y_p\}$的子集。

② 触发器激励函数集合：$\{Y_k^{n+1}=h_k(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $k=1,2,\cdots,p\}$。

2. 同步时序逻辑电路的特点

同步时序逻辑电路的模型决定了同步时序逻辑电路具有如下特点。

（1）电路状态的变化受统一的时钟脉冲控制。

（2）电路状态的改变，只发生在时钟的上升沿或下降沿时刻。

（3）每个时钟脉冲的上升沿或下降沿只能使电路状态发生一次变化。

3. 同步时序逻辑电路的统一时钟信号

统一的时钟脉冲是同步时序逻辑电路工作的基本步调，由于存在统一的时钟脉冲，同步时序逻辑电路才能按预定的状态序列有条不紊地工作。时钟信号是周期固定、连续不停的 0、1 交替序列。图 5-2 给出了时钟信号示例。上升沿是指信号 0→1 变化的时刻，也可以表示为“↑”；下降沿是指信号 1→0 变化的时刻，也可以表示为“↓”。

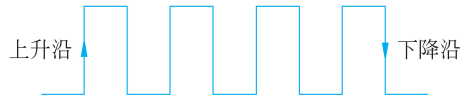

图 5-2　时钟信号

在统一时钟脉冲控制下，同步时序逻辑电路所有的触发器在同一个时钟边沿（上升沿或下降沿，与使用的触发器类型有关）发生状态的改变，并且每个时钟边沿所有的触发器状态只发生一次变化。这种工作方式使同步时序逻辑电路的状态变化容易控制，不会受到其他激励信号上的各种毛刺的干扰，提高了电路的可靠性。

在研究同步时序逻辑电路时，通常不把统一时钟信号作为输入信号处理，而是将它当成一种默认的时间基准，电路状态的改变只能发生在统一时钟信号的边沿时刻。若把某个时钟脉冲边沿到来前电路所处的状态称为现态，则该时钟脉冲边沿作用后电路的状态称为次态，即前一个脉冲的次态是后一个脉冲的现态。

5.1.2　同步时序逻辑电路的描述方法

以下几种同步时序逻辑电路的描述方法虽然都可用于描述同步时序逻辑电路的工作原理，但各自的用途不同。

1. 用状态图描述

这种方法使用直观的图形方法描述电路的功能，可以用于电路设计初期的电路建模阶段。

2. 用状态表描述

这种方法使用真值表方法描述电路的功能，可以用于电路的简化设计阶段。

3. 用时序图描述

这种方法使用波形描述电路的功能，也可以用于电路设计初期的电路建模阶段。

4. 用函数表达式描述

这种方法使用数学的方法描述电路的功能，可以用于电路的实现阶段。同步时序逻辑电路的函数表达式有两种描述方法。

1）输出函数与次态函数

输出函数：$Z_i=f_i(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $i=1,2,\cdots,m$。

次态函数：$y_j^{n+1}=g_j(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $j=1,2,\cdots,q$。

这种描述方法只说明了输出和次态与输入和现态的关系，并未表明采用什么电路实现。

2）输出函数、激励函数和状态方程

输出函数：$Z_i=f_i(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $i=1,2,\cdots,m$。

状态方程：$y_j^{n+1}=g_j(Y,y_j^n)$，其中 $j=1,2,\cdots,q$。

激励函数：$Y_k^{n+1}=h_k(x_1,x_2,\cdots,x_n,y_1^n,y_2^n,\cdots,y_q^n)$，其中 $k=1,2,\cdots,p$。

这种描述方法不仅说明了输出和次态与输入和现态的关系，还表明了采用什么样的触发器作为存储元件。

5. 用原理图描述

这种方法使用原理图符号描述电路的原理性结构，再利用门电路实现组合逻辑函数，最后利用触发器实现状态存储电路。

6. 用 VHDL 描述

这种方法使用 VHDL 描述同步时序逻辑电路，与用 VHDL 描述触发器的程序结构基本相同，不同之处在于，同步时序逻辑电路需要在一个进程的同一个时钟边沿产生所有状态的次态，而触发器只需要处理一个状态。

5.2 同步时序逻辑电路的设计

电路设计的基本任务是按照要实现的功能描述给出电路实现。

同步时序逻辑电路的设计过程与组合逻辑电路的设计过程基本一致，主要包括两个环节：一是建立同步时序逻辑电路模型，二是用合适的集成逻辑器件实现电路。

1. 建立同步时序逻辑电路模型

在实际的电路设计中，功能描述是多种多样的，可以是直接的状态图、状态表、函数表达式，也可以是波形图、表格、文字等描述方法。无论什么样的描述方法，关键是要理解功能描述表达的含义，建立正确的电路模型，这样才能够设计出符合功能描述的电路。

同步时序逻辑电路模型的建立过程如下。

(1) 根据功能描述，确定输入信号、输出信号，建立原始状态图和原始状态表。这个环节实际上是对次态函数和输出函数的建模，其关键是正确理解功能描述表达的含义，用状态图和状态表全面、正确地描述电路的状态、输出及状态迁移过程。

(2) 将原始状态表化简为最小化状态表。原始状态图、原始状态表的状态个数是在分析阶段根据功能描述人为设定的，所以原始状态的个数可能存在冗余。由于冗余的状态会增加触发器元件个数，因此需要通过原始状态表的化简消除冗余状态，使状态表中的状态个数最少。

(3) 状态分配，将符号化状态表转换为二进制状态表。原始状态表可能使用符号表示，由于最终需要用触发器实现，因此要把符号化状态表转换为二进制状态表。这就需要根据状态个数确定使用触发器的数目，进而给每个状态指定二进制编码。

(4) 根据二进制状态表，写出电路的次态函数和输出函数。二进制状态表事实上就是次态函数和输出函数的真值表，利用真值表可以很容易写出电路次态函数和输出函数的表达式。

2. 用合适的集成逻辑器件实现电路

根据不同的电路规模、电路实现条件，可以选择相应的集成逻辑器件来实现电路。

(1) 用触发器和门电路实现同步时序逻辑电路。用触发器作为存储元件时，首先要确定使用触发器的类型，选定触发器后，触发器的状态方程就确定了，同时触发器的激励端也确定了。这时就需要利用触发器的状态表和激励表，确定激励函数真值表，将次态函数分解为触发器的状态方程和激励函数表达式，再结合输出函数表达式画出电路。

(2) 用触发器和 SPLD 器件实现同步时序逻辑电路。用 SPLD 器件的与或门阵列替代门电路实现时序电路的组合逻辑电路部分，存储元件仍然需要使用触发器，其电路实现过程与用触发器和门电路实现时序逻辑电路的过程完全相同。在有些 SPLD 器件中也集成了一部分触发器电路，这样可以减少使用分立的触发器元件，其电路实现过程与使用专门的触发器元件实现电路的方法基本相同。这部分内

容本章不再专门讨论。

(3) 用 VHDL 描述同步时序逻辑电路。根据电路的需要，可以采用行为描述法、数据流描述法或结构描述法对电路功能进行描述。

5.2.1　建立原始状态图和原始状态表

状态图和状态表能够直观、清晰地反映同步时序电路的逻辑功能。所以，同步时序电路设计的第一步是建立能够描述功能要求的原始状态图和原始状态表。

原始状态图和原始状态表通常是根据问题的描述直接建立起来的，它们是对设计要求的最原始的抽象，是设计电路的初始依据。如果原始状态图不能正确地反映设计要求，那么依此设计出来的电路将会是错误的。因此，建立正确的原始状态图和原始状态表是同步时序电路设计中最关键的环节。

同步时序电路有米利(Mealy)型和摩尔(Moore)型两种模型，具体将电路设计成哪种模型，有的由设计要求规定，有的可由设计者选择。不同模型对应的电路结构不同，设计时，应根据问题中的信号形式、电路所需器件的多少等进行综合考虑。

状态图和状态表是可以相互转换的，由状态图转换为状态表时，只需将所有状态在状态表的最左列列出作为现态，将所有输入组合在状态表的第二行列出作为输入，根据状态图的迁移方向(次态)和迁移条件(输入输出)，在状态表的现态和输入交叉位置填上对应的次态和输出即可，无关的次态和输出填入任意项 X。

建立原始状态图没有统一的方法。一般而言，对于不同的电路功能描述，其状态个数有确定和不确定两种情况，其状态图的建立方法是不一样的。

1. 状态个数确定的情况

通过对功能描述的分析，能够明确确定最少状态个数时，首先分别给每个状态分配一个不同的符号表示，在状态图上画出所有的状态。然后分析功能描述的含义，在状态图上标出每个状态在所有输入情况下的状态迁移方向和对应的输出取值。

需要强调的是，对于有 n 个外部输入信号的状态图，从每个状态出发，必须要考虑所有 2^n 种输入组合情况下对应的次态和输出，否则状态图将是不完整的。

例 5-1　某模 5 加 1 和加 2 计数器，有一个输入 x 和一个输出 Z。输入 x 为加 1、加 2 控制信号，当 $x=0$ 时，计数器在时钟脉冲作用下进行加 1 计数；当 $x=1$ 时，计数器在时钟脉冲作用下进行加 2 计数；当电路计满 5 个状态后，输出 Z 产生一个 1 信号作为进位输出；平时 Z 输出为 0。试建立该计数器的 Mealy 型原始状态图和状态表。

解：根据题意，电路有一个输入 x，一个输出 Z，在计数脉冲 CP 作用下进行同步计数。

由于是模 5 计数器，所以电路应有 5 个状态，分别对应计数值 0～4。首先画出 5 个状态如图 5-3(a)所

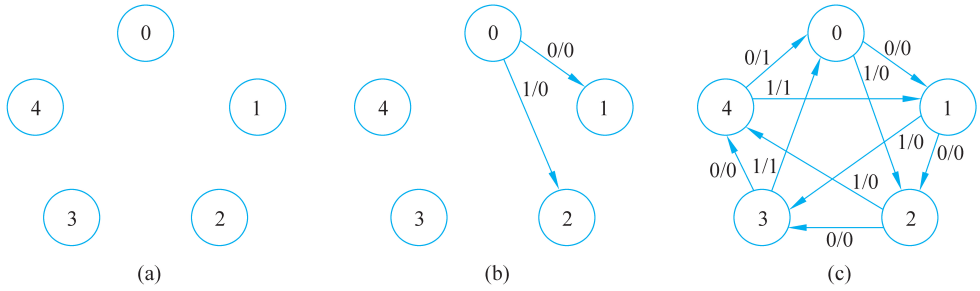

图 5-3　例 5-1 的状态图

(a)5 个状态；(b)部分状态图；(c)完整状态图

示。从状态 0 出发，分别考虑 $x=0$ 和 $x=1$ 两种情况，可得其状态的次态和输出，如图 5-3(b)所示。同样，从每个状态出发，分别考虑 $x=0$ 和 $x=1$ 两种情况，可得完整的状态，如图 5-3(c)所示。

将状态图中每个状态在 $x=0$ 和 $x=1$ 时的迁移次态和输出情况填入表 5-1 中，即构成模 5 加 1 和加 2 计数器的状态表。

表 5-1　例 5-1 的状态表

现　态	次态,输出	
	$x=0$	$x=1$
0	1,0	2,0
1	2,0	3,0
2	3,0	4,0
3	4,0	0,1
4	0,1	1,1

由状态图构造状态表时，首先将状态 0～4 在状态表的最左列列出，将输入 x 的两种取值情况在状态表的第二行列出；然后在状态表中每一个现态和输入交叉位置填上状态图中对应的次态和输出，即可构成完整的状态表。

2. 状态个数不确定的情况

通过对功能描述的分析，在不能明确知道最少状态个数时，可以采用以下过程建立原始状态图。由于状态个数不确定，在原始状态图和状态表中一般用字母或数字表示状态。

(1) 设定一个初始状态 A，表示电路没有任何输入时的初始状态。

(2) 从每一个已有的现态出发，对于所有可能的输入组合，分别考虑每一种输入组合情况下应该产生的次态和输出，建立现态与次态之间的迁移关系。

如果要迁移的下一个状态在状态图中不存在，则增加一个新状态，同时在这两个状态之间画一个迁移箭头并标注迁移条件(输入输出)；如果要迁移的下一个状态在状态图中已经存在，则直接在这两个状态之间画一个迁移箭头并标注迁移条件(输入输出)。

(3) 对每个新产生的状态，重复步骤(2)，直到没有新状态产生，这时就构成了一个完整的状态图。

需要注意的是，建立完整、正确的原始状态图是同步时序电路设计中最关键的一步，当问题描述比较复杂时，不能为了减少状态个数而建立不完整或错误的状态图，冗余的状态通过后期状态表的化简可以消除。

例 5-2　某序列检测器有一个输入端和一个输出端。在输入端输入一串随机的二进制代码 x，当输入序列中出现“011”序列时，输出 Z 产生一个 1 输出，平时 Z 输出 0。典型输入、输出序列如下。

输入 x：1　0　1　0　1　1　1　0　0　1　1　0

输出 Z：0　0　0　0　0　1　0　0　0　0　1　0

试得到该序列检测器的原始状态图和原始状态表。

解：假定用 Mealy 型同步时序逻辑电路设计该序列检测器，原始状态图的建立过程如下。

设电路的初始状态为 A。在 A 状态下，如果电路输入为 0，由于未得到“011”序列，所以输出 Z 为 0。由于输入的 0 是序列“011”中的第一个信号，所以应该用一个新状态将它记住，在状态图中增加一个

新状态 B 记住收到了第一个 0，然后在状态 A 和 B 之间建立一个迁移关系，迁移条件为 $0/0(x/Z)$。当处在初始状态 A 时电路输入为 1，输出 Z 为 0，由于 1 不是序列“011”的第一个信号，故不需要记忆，可令其停留在状态 A，继续等待第一个 0 的出现。该转换关系如图 5-4(a)所示。

当电路处于状态 B 时，若输入 x 为 0，它不是“011”序列的第二个信号，但仍可作为序列中的第一个信号，故可令电路停留在状态 B，输出为 0；若输入 x 为 1，则意味着收到了“011”序列的前两位 01，需用一个新状态 C 将它记住，故此时电路由状态 B 迁移到状态 C，输出为 0。状态图如图 5-4(b)所示。

当电路处于状态 C 时，若输入 x 为 0，则收到的连续 3 位代码为“010”，不是关心的序列“011”，但此时最后输入的 0 依然可以作为序列的第一个信号，故此时应迁移到状态 B，输出 0；若输入 x 为 1，则表示收到了序列“011”，需用一个新状态 D 记住，此时迁移到状态 D，输出 1。状态图如图 5-4(c)所示。

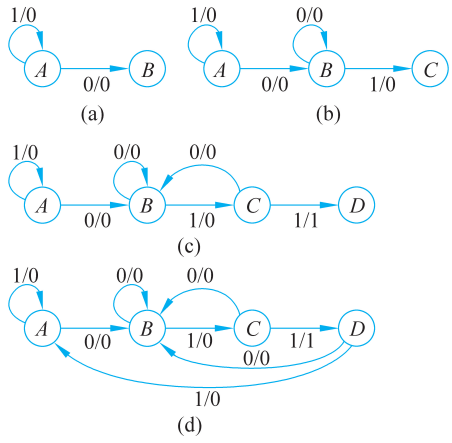

图 5-4　例 5-2 的 Mealy 型状态图

当电路处于状态 D 时，若输入 x 为 0，则应迁移到状态 B，由于最后输入的三位已经不是“011”，输出应为 0；若输入 x 为 1，此时最后输入的三位为“111”，则应输出 0，迁移到状态 A，等待 0 的到来。至此，状态图中没有未考虑的新状态，得到的状态图就是该序列检测器完整的 Mealy 型状态图，如图 5-4(d)所示。

相应的原始状态表如表 5-2 所示。

表 5-2　例 5-2 的 Mealy 型状态表

现　　态	次态，输出	
	$x=0$	$x=1$
A	B，0	A，0
B	B，0	C，0
C	B，0	D，1
D	B，0	A，0

若用 Moore 型同步时序电路实现“011”序列检测器的逻辑功能，则电路输出完全取决于状态，而与输入无直接联系。在画状态图时，应将输出标记在代表各状态的圆圈内。

假定电路初始状态为 A，并用状态 B、C、D 分别表示收到了输入 x 送来的 0、01、011。显然，根据题意，仅当处于状态 D 时电路输出为 1，其他状态下输出均为 0。

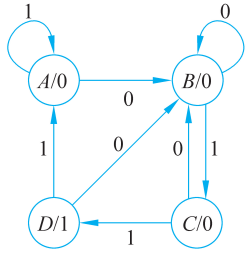

图 5-5　例 5-2 的 Moore 型状态图

当从初始状态 A 开始，输入端 x 正好依次输入 0、1、1 时，则状态从 A 转至 B、B 转至 C、C 转至 D。考虑到 A 状态下输入为 1 时，它不是指定序列中的第一位信号，不必记忆，可令状态停留在 A；B 状态下输入为 0 时，它不是指定序列的第二位，但可作为指定序列的第一位，故可使状态停留在 B；C 状态下输入 0 时，它不是指定序列的第三位，但同样可作为第一位，故使状态转向 B；D 状态下输入 0 时，同样应转向 B，而输入为 1 时，则应使状态进入 A。完整的 Moore 型原始状态图如图 5-5 所示。

相应的原始状态表如表 5-3 所示。

表 5-3　例 5-2 的 Moore 型状态表

现　态	次　态		输　出
	$x=0$	$x=1$	Z
A	B	A	0
B	B	C	0
C	B	D	0
D	B	A	1

例 5-3　设计一个用于引爆控制的同步时序电路，该电路有一个输入端 x 和一个输出端 Z。平时输入 x 始终为 0，一旦需要引爆，则从 x 输入连续的 4 个 1 信号(不能被 0 间断)，电路收到第 4 个 1 后在输出端 Z 产生一个 1 信号点火引爆，该电路连同引爆装置一起被炸毁。试建立该电路的 Mealy 型状态图和状态表。

解：该电路实际上是一个用于特殊场所的“1111”序列检测器。它与一般序列检测器不同的是，收到 4 个 1 后产生引爆信号，同时使电路自毁，故此时不再存在次态问题。

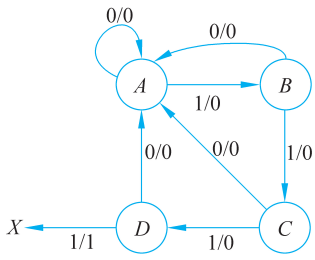

图 5-6　例 5-3 的 Mealy 型状态图

设状态 A 表示电路初始状态，状态 B 表示收到了第一个 1 输入，状态 C 表示收到了连续的 2 个 1 输入，状态 D 表示收到了连续的 3 个 1 输入。根据题意，A 状态下 x 为 1 时，输出为 0，转向状态 B；B 状态下 x 为 1 时，输出为 0，转向状态 C；C 状态下 x 为 1 时，输出为 0，转向状态 D；而 D 状态下 x 为 1 时，输出为 1，次态随意(实际上已不存在次态)。

A 状态下 x 为 0 时，可令输出为 0，停留在状态 A，而 B、C、D 这 3 个状态下 x 为 0 时，前面的 1 可能是误触发或干扰信号，故它们的输出 $Z=0$，且均应回到 A 等待可靠的引爆信号。据此，可得到该电路的 Mealy 型原始状态图如图 5-6 所示。

原始状态表如表 5-4 所示，表中用 X 表示不确定次态或不确定输出。

表 5-4　例 5-3 的 Mealy 型状态表

现　态	次态，输出	
	$x=0$	$x=1$
A	A，0	B，0
B	A，0	C，0
C	A，0	D，0
D	A，0	×，1

5.2.2　状态表化简

从原始状态图的建立过程可以看出，在明确知道状态个数的情况下，每个原始状态都是必不可少的，这样的状态表不需要化简。在状态个数不明确的情况下，原始状态可能有冗余。在设计具体电路时，状态数目的多少将直接决定电路中所需触发器数目的多少。为了降低电路的复杂性和电路成本，应尽可能地使状态表中包含的状态数达到最少，这时就需要进行状态表化简。

状态表化简就是采用某种化简技术从原始状态表中消去多余状态，得到一个既能正确地描述给定的逻辑功能，又能使状态数目达到最少的状态表，通常称这种状态表为最小化状态表。状态表化简的基本思想就是把那些具有相同含义的状态合并为一个状态，使化简后的状态在状态表中都是必不可少的。

原始状态表可能存在两种情况，一种是状态表中所有的次态和输出都是确定的完全定义状态表，另一种是不完全定义状态表。这两种状态表的化简方法有所不同，下面分别讨论。

1. 完全定义状态表的化简

完全定义状态表的化简建立在状态等效概念基础之上。完全定义状态表中，如果两个状态是等效的，则称这两个状态为等效状态对，可以合并为一个状态。如果 n 个状态相互之间都是等效的，则称这 n 个状态为状态等效类，可以合并为一个状态。完全定义状态表的化简过程就是找出能够覆盖原始状态表所有状态的最大等效类，合并等效类后得到最小化状态表的过程。

1）等效状态对(S_i,S_j)

假设状态 S_i 和 S_j 是完全定义状态表中的两个状态，如果对于所有可能的输入序列，分别从 S_i 和 S_j 出发，得到的次态和输出响应序列完全相同，则状态 S_i 和 S_j 是等效状态对，记作(S_i,S_j)。

在状态表中的每一种输入组合情况下，S_i 和 S_j 等效，应同时满足如下输出和次态条件。

（1）输出相同。

（2）次态属于下列情况之一。

① 次态相同。

次态相同是指在某一种输入情况下，现态 S_i 和 S_j 的次态均为 S_k。

② 次态交错或为各自的现态。

次态交错是指在某一种输入情况下，现态 S_i 的次态是 S_j，现态 S_j 的次态是 S_i。

次态是各自的现态是指在某一种输入情况下，现态 S_i 的次态是 S_i，现态 S_j 的次态是 S_j。

③ 次态循环或为等效对。

次态循环是指在某一种输入情况下，现态 S_i 的次态是 S_j，S_j 的次态是 S_k，S_k 的次态是 S_i，S_i、S_j 和 S_k 形成了次态循环。

次态为等效对是指在某一种输入情况下，S_i 的次态是 S_k，S_j 的次态是 S_l，而 S_k 与 S_l 是等效状态对。

以上情况在图 5-7 中给予了图示说明。

从状态表中找出等效状态对是状态表化简的第一步，当状态表中状态个数比较多时找出所有等效状态对的工作是比较烦琐的，为了进行有序比较，下面给出了寻找等效状态对的有效工具——隐含表。

2）利用隐含表，找到等效状态对

隐含表的基本结构如图 5-8 所示。隐含表是一个直角三角形阶梯网格，用以实现状态表中状态对之间的两两比较，从而决定任意两状态之间的等效关系。隐含表的横向和纵向格数相同，因为每个状态不用和自己比较，所以格数等于原始状态表中的状态数减 1。隐含表横向从第一个状态开始标注，纵向从第二个状态开始标注。

寻找等效状态对时，首先进行两两比较，在隐含表中每个交叉点标注两个状态是否等效，用“√”表示等效，用“×”表示不等效，不确定是否等效则标注其依赖的状态对。

两两比较完成后进行关联比较，对于那些不确定是否等效的状态对，要检查其依赖的状态对是否等效，如果依赖的状态对也不确定是否等效，则继续检查新的依赖状态对，直到有明确的等效或不等效状态对出现，或者形成循环。一旦出现一个不等效状态对，则整个状态对序列上的所有状态对都不等效；如果最后确定的状态对都等效，则整个状态对序列上的所有状态对都等效；如果形成循环，则整个状态

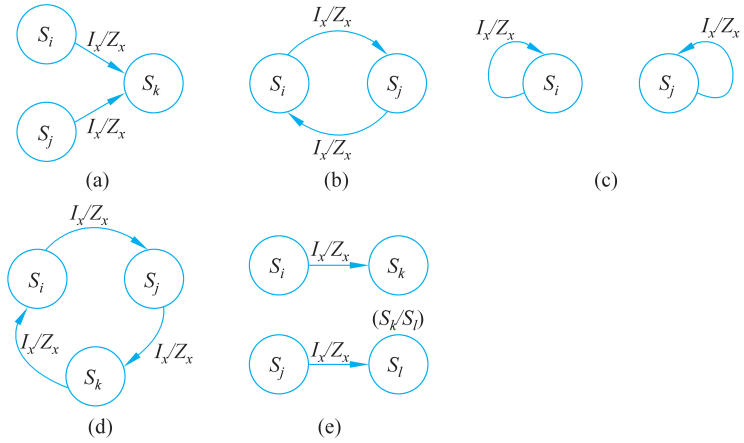

图 5-7　次态等效条件

(a) 次态相同；(b) 次态交错；(c) 各自的现态；(d) 次态循环；(e) 次态为等效对

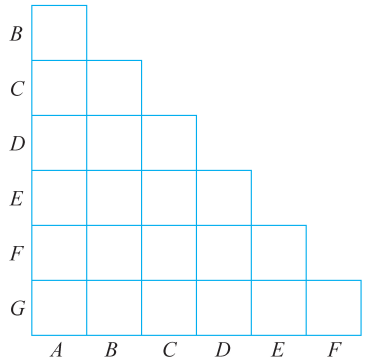

图 5-8　隐含表的基本结构

对循环上的所有状态对都等效。

3）利用等效状态的传递性，寻找最大等效类

等效状态具有传递性。假设 S_i 和 S_j 等效，S_j 和 S_k 等效，那么 S_i 和 S_k 也等效。记作(S_i,S_j)，$(S_j,S_k)\rightarrow(S_i,S_k)$。

等效类是彼此等效的状态构成的等效状态集合。根据等效状态的传递性，可以从等效对中寻找出由彼此等效的状态构成的等效类。例如，由(S_i,S_j)和(S_j,S_k)可以推出(S_i,S_k)，因此 S_i、S_j、S_k 属于同一等效类，记作$\{S_i,S_j,S_k\}$。

最大等效类是不被任何别的等效类包含的等效类。这里所指的最大，并不是指包含的状态最多，而是指它的独立性，即使只有一个状态，只要它不被包含在别的等效类中，也是最大等效类。

需要注意的是，由于等效状态具有传递性，因此任意一种状态最终只能出现在一个最大等效类中，任何两个最大等效类中不可能出现同一个状态。

4）合并最大等效类，作出最小化状态表

最大等效类是彼此等效的所有状态构成的集合，它们可以合并成一个状态。原始状态表的化简过程，就是寻找最大等效类，然后将每个最大等效类合并为一个新状态，从而得到最小化状态表的过程。

例 5-4　化简表 5-5 所示的原始状态表。

表 5-5　例 5-4 的状态表

现　态	次态,输出	
	$x=0$	$x=1$
A	C,0	B,1
B	F,0	A,1
C	F,0	G,0
D	D,0	E,0
E	C,0	E,1
F	C,0	G,0
G	C,1	D,0

解：

(1) 作隐含表。根据画隐含表的规则，可以得到与给定状态表对应的隐含表框架，如图 5-9 所示。由于原始状态表中有 $A\sim G$ 共 7 个状态，所以隐含表的横向和纵向各有 6 个方格。纵向从上到下依次为 $B\sim G$，横向从左到右依次为 $A\sim F$。表中每个方格代表一个状态对，如左上角的方格代表状态对 A 和 B，右下角的方格代表状态对 F 和 G。

(2) 顺序比较。顺序比较时，由于输出逻辑值是确定的 0 或者 1，其是否等效一目了然，因此可以根据输出条件快速确定哪些状态对不等效，剩余的少数不确定状态对再根据次态等效条件进行比较。这样一来，可以大大减少两两比较的工作量。例如，在状态表 5-5 中的 $x=0$ 列，由于状态 G 的输出与其他状态的输出均不相同，因此可以快速确定状态 G 和其他所有状态不等效，在隐含表中 G 所在的行全部标记“×”；同样，在 $x=1$ 列，根据输出是否相同很快可以确定状态 A、B、E 与其他状态也不等效，在隐含表中相应位置直接标记“×”。由输出条件确定的隐含表如图 5-9 所示。

剩余的少数状态对只需按照次态等效条件顺序进行两两比较。状态表 5-5 中，A、B 是否等效取决于 C、F，因此在 A、B 的交叉点填入状态对 CF；同样，A、E 是否等效取决于 B、E，因此在 A、E 的交叉点填入状态对 BE；B、E 是否等效取决于 A、E 和 C、F，因此在 B、E 的交叉点填入状态对 AE 和 CF。后面依次比较 C、D、C、F 和 D、F，得到的隐含表如图 5-10 所示。其中 C、F 的次态出现了状态交错，因此 C、F 是等效的，在隐含表的相应方格内填入“√”。

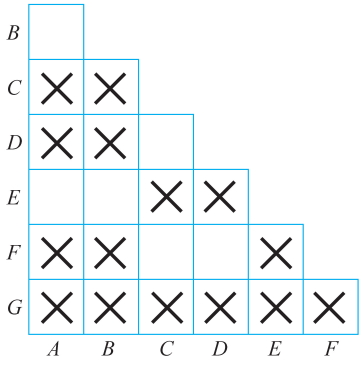

图 5-9　由例 5-4 输出条件得到的隐含表

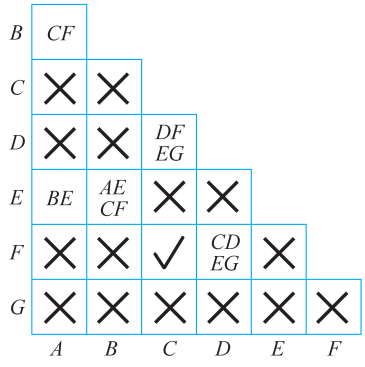

图 5-10　由例 5-4 次态条件得到的隐含表

经过顺序比较后，还有一些状态对尚未确定是否等效，故应接着进行关联比较。