第 5 章　时序逻辑电路

在组合电路中加入用来存储电路历史信息(称为状态)的存储元件即可构成时序电路。时序电路与组合电路的主要区别是：组合电路的输出只与当前的输入有关；而时序电路的输出不仅与当前的输入有关,而且还与当时的状态有关。由于时序电路的特点与组合电路不同,所以对时序电路的研究方法也与组合电路有所不同。在组合电路设计中,使用函数的真值表来描述一个组合逻辑问题；而在时序电路设计中,则使用时序机的状态表来描述一个时序逻辑问题。

时序电路分为同步时序电路和异步时序电路。同步时序电路的特点是电路有统一的时钟脉冲,在次态函数(又称为激励函数)的控制下,只有当时钟脉冲到达时电路的状态才发生改变；新的状态一旦建立,又形成新的激励,但直到下一个时钟脉冲到达之前电路的状态不会发生新的变化。同步时序电路工作稳定、可靠,而且分析和设计较为简单,这是同步时序电路被广泛采用的原因所在。但是,在有些场合,如当电路的工作速度较高或者输入是随机变化的场合,或是电路中不同部件的工作速度相差较为悬殊的场合,往往采用异步时序电路。

本章只介绍同步时序电路。

5.1　时序电路与时序机

5.1.1　时序电路的结构和特点

为了充分理解时序电路的结构和特点,不妨先回顾总结一下组合电路的结构和特点。图 5.1 展示了组合电路的结构模型,图中,$x_1,x_2,\cdots,x_n$ 为某一时刻的输入；$Z_1,Z_2,\cdots,Z_m$ 为该时刻的输出。显然,组合电路的逻辑功能可用下列输出函数集来描述：

$$Z_i=f_i(x_1,x_2,\cdots,x_n),\quad i=1,2,\cdots,m$$

由此可见,组合电路的特点是电路在任何时刻的输出 Z_i 仅与该时刻的输入 $x_1,x_2,\cdots,x_n$ 有关,而与该时刻以前的输入无关。

时序电路是由组合电路和存储元件两部分构成的,它可以用图 5.2 所示的结构模型来表示。对图中组合电路而言,它有两组输入和两组输出。其中,输入 $x_1,x_2,\cdots,x_n$ 称为时序电路的外部输入；输入 $y_1,y_2,\cdots,y_k$ 称为时序电路的内部输入,它是存储元件的输出反馈到组合电路的输入；输出 $Z_1,Z_2,\cdots,Z_m$ 称为时序电路的外部输出；而输出 $Y_1,Y_2,\cdots,Y_p$ 称为时序电路的内部输出。

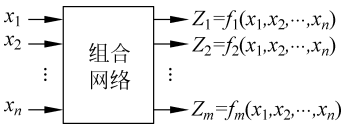


图 5.1　组合电路的结构模型

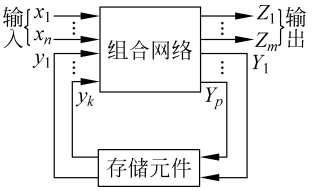


图 5.2　时序电路的结构模型

时序电路通过存储元件的不同状态来“记忆”以前时刻的输入。状态是时序电路的一个重要概念，它用来表示时序电路的过去属性。通常称 $y_1,y_2,\cdots,y_k$ 为时序电路的现态，而时序电路的外部输出 Z_i 和内部输出 Y_j 是当前输入 x 和现态 y 的函数：

$$Z_i=f_i(x_1,x_2,\cdots,x_n;\ y_1,y_2,\cdots,y_k),\quad i=1,2,\cdots,m$$

$$Y_j=g_j(x_1,x_2,\cdots,x_n;\ y_1,y_2,\cdots,y_k),\quad j=1,2,\cdots,p$$

一般称 Z_i 为输出函数，Y_j 为激励函数或次态函数。

因此，时序电路的输出不仅与该时刻的输入有关，还与当时的状态有关。这就是时序电路的主要特点。表 5.1 概括了组合电路和时序电路的区别。

表 5.1　组合电路和时序电路的区别

类　别	组 合 电 路	时 序 电 路
电路特性	输出只与当前输入有关	输出与当前输入和状态有关
电路结构	不含存储元件	含存储元件
函数描述	用输出函数描述	用输出函数和次态函数描述

5.1.2　时序机的定义

时序机在有的文献中亦称为有限状态机或有限自动机，它是一个从实际中抽象出来的数学模型。任何一个时间离散系统，只要满足下述定义，不管是具体的物理机器（如一个时序电路），还是抽象的虚拟“机器”（如一个算法），都可以称为时序机。

时序机是这样一个系统，它可以用 5 个参量来表征：

$$M=(I,O,Q,N,Z)$$

其中，I 为时序机的输入字母有限非空集合；O 为时序机的输出字母有限非空集合；Q 为时序机的内部状态有限非空集合；N 为时序机的次态函数，它表示输入及状态到次态的映射，即 $I\times Q\to Q$；Z 为时序机的输出函数，它有以下两种情况。

（1）若输出函数 Z 是输入和状态的函数，即 $Z:I\times Q\to O$，那么该时序机称米利（Mealy）型时序机。

（2）若输出函数 Z 仅是其状态的函数，即 $Z:Q\to O$，那么该时序机称穆尔（Moore）型时序机。

能够用时序机来描述的例子是广泛存在的。如日常生活中的电话系统、自动售票机、号码锁、数字系统中的触发器、时序逻辑电路，甚至数字计算机等都可用时序机来描述。

5.1.3 时序机的状态表和状态图

在工程应用中,时序机通常用更直观的形式——状态表和状态图来表示。所谓状态表,就是用表格方式来描述时序机的输入、状态和输出之间的关系,而状态图是用图解方式来描述上述关系,两种方法相辅相成,经常配合使用。由于两种类型时序机的状态表和状态图在表示方法上有所不同,因此下面分别进行讨论。

1. Mealy 机的状态表和状态图

Mealy 机的状态表和状态图反映了时序机的输出与它的输入以及现态之间的关系。设时序机的输入 I 为 $I_1,I_2,\cdots,I_n$,时序机的内部状态 Q 为 $q_1,q_2,\cdots,q_k$,则状态表的一般形式如表 5.2 所示。

表 5.2 Mealy 机状态表的一般形式

Q	I		
	I_1	$\cdots$	I_n
q_1	$N(q_1,I_1),Z(q_1,I_1)$	$\cdots$	$N(q_1,I_n),Z(q_1,I_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
q_k	$N(q_k,I_1),Z(q_k,I_1)$	$\cdots$	$N(q_k,I_n),Z(q_k,I_n)$

表中,q_i 行和 I_j 列($i=1,\cdots,k$; $j=1,\cdots,n$)相交处的项表示当时序机处于状态 q_i 并在输入 I_j 时的下一状态和输出,这个项表示成 $N(q_i,I_j)$、$Z(q_i,I_j)$,这里 N 和 Z 分别是时序机的次态函数和输出函数。

状态图是这样建立的:用小圆圈表示状态 q_i,用有向箭头表示状态转换的方向,并在箭头上标上 I_j/Z_k,它表示在输入值 I_j 的情况下,状态由 q_i 转换到下一状态 q_{i+1} 时,其输出为 Z_k,如图 5.3 所示。

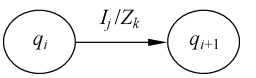


图 5.3 Mealy 机状态转换的表示

应该注意,在状态表和状态图中没有标出时钟脉冲。而实际上,对同步时序电路而言,只有在时钟脉冲作用下才发生状态的转换(这是由同步触发器的特点决定的)。这里规定:在某时钟脉冲到达以前电路所处的状态称为电路对该时钟脉冲的现态;而把该时钟脉冲到达之后电路的状态称为电路对该时钟脉冲的次态。还有一点要指出的是,对 Mealy 型时序电路来说,当输入发生变化时,输出立即跟着变化,而电路状态要等到下一时钟脉冲到达后才发生变化,状态变化后,输出再次随之发生变化。

2. Moore 机的状态表和状态图

由于 Moore 机的输出仅与现态有关,因此在 Moore 机的状态表中,每一现态 q_i 的行都应有相同的输出,而与输入无关,因此可以把它们提出来新开一列。Moore 机状态表的一般形式如表 5.3 所示。

表 5.3　Moore 机状态表一般形式

Q	I			
	I_1	…	I_n	Z
q_1	$N(q_1, I_1)$	…	$N(q_1, I_n)$	$Z(q_1)$
⋮	⋮	⋮	⋮	⋮
q_k	$N(q_k, I_1)$	…	$N(q_k, I_n)$	$Z(q_k)$

在 Moore 机的状态图中，输出 Z_k 应与状态 q_i 写在一起，表示 Z_k 只与状态有关，即在圆圈内标以 q_i/Z_k；输入仍标在箭头上，如图 5.4 所示。

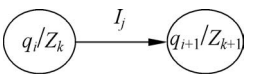


图 5.4　Moore 机状态转换的表示

由于 Moore 机的输出仅与现态有关，因此新的输出是由转换后的状态决定的，也就是说，先有状态的变化，再形成新的输出。这一点与 Mealy 型电路有所不同，而这一不同，归根到底是由于两种类型时序电路的输出不同引起的。

5.1.4　完全定义机和不完全定义机

在组合电路中，如果一个布尔函数的真值表中所有输出值都是确定的，则此函数称为完全定义函数；否则，称为不完全定义函数。

在时序电路中，如果一个时序机的状态表中所有的次态/输出都是确定的，则此时序机称为完全定义机；否则，称为不完全定义机。

不完全定义机在实际中是常常遇到的。例如，当时序机处于状态 q_i 时，由于不可能（或不允许）出现输入 I_k，所以当从状态 q_i 向下一状态转换时，次态/输出是随意的，或没有意义的。如下面将要介绍的基本 RS 触发器不允许同时输入置 1 和置 0 信号就是其中一例。有时，即使是完全定义机，往往在给其状态表的符号状态进行二进制编码时，会使完全定义机变成不完全定义机。例如，一个模 6 二进制加 1、减 1 计数器，其状态表如表 5.4 所示。表中，当 $x=0$ 时，进行加 1 计数；当 $x=1$ 时，进行减 1 计数。为了表示 6 个状态 $q_0 \sim q_5$，至少需要 3 个存储元件（触发器），设用 y_1、y_2 和 y_3 来表示。经过二进制编码后的状态表如表 5.5 所示。从表中可以看出，由于二进制编码，产生了两行无定义的次态和输出。这说明即使给定一个完全定义机，经过二进制赋值后，也会变成不完全定义机。

表 5.4　计数器的状态表

Q	x		
	0	1	Z
q_0	q_1	q_5	0
q_1	q_2	q_0	0
q_2	q_3	q_1	0
q_3	q_4	q_2	0
q_4	q_5	q_3	0
q_5	q_0	q_4	1

表 5.5 经过二进制编码后的状态表

$y_1y_2y_3$	x		
	0	1	Z
0 0 0	001	101	0
0 0 1	010	000	0
0 1 0	011	001	0
0 1 1	100	010	0
1 0 0	101	011	0
1 0 1	000	100	1
1 1 0	d	d	d
1 1 1	d	d	d

5.2 触发器

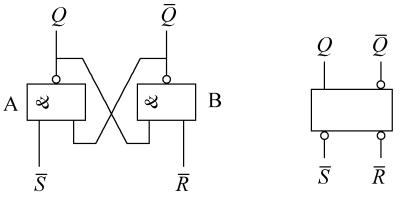
触发器是一种具有记忆功能、能存储二进制信息的逻辑电路,是构成时序逻辑电路的基本单元。

触发器具有两个基本特征:第一,具有两个稳定状态,分别称为“0”状态和“1”状态,触发器的这两个稳定状态可以分别表示一位二进制代码 0 和 1。在没有外界信号作用时,触发器维持原来的稳定状态不变,即触发器具有记忆功能。触发器又称为双稳态触发器;第二,在一定的外界信号作用下,触发器可以从一个稳定状态转变到另一个稳定状态。这表明触发器可以接收信号,并保存下来。外界信号的作用称为触发,这也是触发器名称的由来。触发器从一个稳态转变到另一个稳态的过程,称为翻转。

触发器按照逻辑功能的不同可分为 RS 触发器、JK 触发器、T 触发器和 D 触发器等几种类型。

5.2.1 基本 RS 触发器

基本的 RS 触发器是电路结构最简单的一种触发器,也是构成其他触发器的基础。由与非门构成的基本 RS 触发器的逻辑图和逻辑符号如图 5.5 所示。



(a) 逻辑图 (b) 逻辑符号

图 5.5 基本 RS 触发器

该触发器由两个两输入与非门 A 和 B 的输入/输出端交叉连接而成，它有两个输入端和两个输出端。Q 为触发器的“1”输出端，简称“1”端，$\overline{Q}$ 为“0”端。通常将 Q 端的状态定义为触发器的状态，即当 $Q=1$ 时称触发器为“1”态；当 $Q=0$ 时称触发器为“0”态。在正常情况下，Q 与 $\overline{Q}$ 端的输出状态总是彼此互补的。当 $Q=1$ 时，$\overline{Q}=0$；反之，当 $Q=0$ 时，$\overline{Q}=1$。

触发器的两个输入端 $\overline{S}$ 和 $\overline{R}$ 分别称为置 1 端和置 0 端。字母 S(Set)和 R(Reset)上的横杠及图 5.5 逻辑符号中的小圆圈均表示置 0、置 1 信号是低电平有效，即平时这两个端应接高电平，只有在输入端 $\overline{S}$ 或 $\overline{R}$ 接低电平(或负脉冲)信号时，触发器的状态才可能改变。

当 $\overline{S}=0$、$\overline{R}=1$ 时，不论 $\overline{Q}$ 为何种状态，都有 $Q=1$、$\overline{Q}=0$。可见，不论触发器原来的状态如何，当在 $\overline{S}$ 端加低电平信号时，触发器都将被置为 1 状态。而且在 $\overline{S}$ 端低电平信号消失后，即 $\overline{S}=1$、$\overline{R}=1$ 时，触发器的状态保持不变。

当 $\overline{S}=1$、$\overline{R}=0$ 时，不论 Q 状态如何，都有 $\overline{Q}=1$、$Q=0$。可见，不论触发器原来的状态如何，当在 $\overline{R}$ 端加低电平信号时，触发器都将被置为 0 状态。同理，在 $\overline{R}$ 端低电平信号消失后，触发器的状态保持不变。

当 $\overline{S}=1$、$\overline{R}=1$ 时，触发器状态保持不变。

当 $\overline{S}=0$、$\overline{R}=0$ 时，Q 与 $\overline{Q}$ 全为 1，使 Q 与 $\overline{Q}$ 彼此互补的逻辑关系遭到破坏。这是不正常情况，是不允许的。而且，当 $\overline{S}$ 和 $\overline{R}$ 低电平信号同时消失后，触发器的状态将无法确定，可能是 0，也可能是 1。

表 5.6 为基本 RS 触发器的真值表。其中，$\overline{S}$ 和 $\overline{R}$ 为触发器的输入信号；Q^n 为触发器接收信号之前的状态，称为现态；Q^{n+1} 为触发器接收信号之后的状态，称为次态。

表 5.6　基本 RS 触发器真值表

Q^n	$\overline{S}$	$\overline{R}$	Q^{n+1}
0	0	0	×
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	×
1	0	1	1
1	1	0	0
1	1	1	1

从使用触发器的角度出发，通常用简化的真值表，即功能表，表示一个触发器所能完成的功能。基本 RS 触发器的功能表如表 5.7 所示。

表 5.7　基本 RS 触发器功能表

$\overline{S}$	$\overline{R}$	Q^{n+1}	功　能
0	0	不正常	不允许
0	1	1	置 1
1	0	0	置 0
1	1	Q^n	保持

根据基本 RS 触发器的真值表,可以得到次态 Q^{n+1} 与现态 Q^n、输入 $\overline{S}$ 及 $\overline{R}$ 的逻辑关系表达式为:

$$\begin{cases} Q^{n+1}=S+\overline{R}Q^n \\ RS=0 \quad (约束条件) \end{cases} \tag{5.1}$$

式中,$RS=0$(或 $\overline{S}+\overline{R}=1$)表示不允许同时出现 $\overline{S}=0$、$\overline{R}=0$ 的情况,称为约束条件。式(5.1)常称为触发器的特征方程,也称为状态方程或次态方程。

在时序电路设计中,往往已知触发器的现态 Q^n 与次态 Q^{n+1},要求所需的输入信号值,称为输入激励。将不同现态和次态时的输入激励列成表格,称为触发器的激励表。表 5.8 为基本 RS 触发器的激励表。

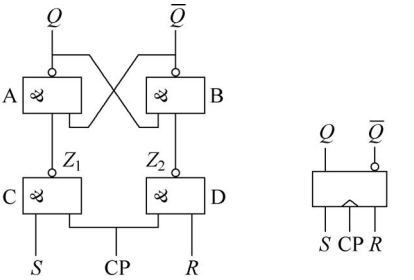
表 5.8 基本 RS 触发器激励表

Q^n	Q^{n+1}	$\overline{R}$	$\overline{S}$
0	0	d	1
0	1	1	0
1	0	0	1
1	1	1	d

5.2.2 同步 RS 触发器

实际的数字系统工作时,要求各部分电路要协调动作,因此系统中用一个同步信号来指挥电路各部分协调动作,该信号又称为时钟脉冲信号,简称时钟,用 CP(Clock Pulse)表示。电路中的触发器只有在时钟脉冲到来时,才按照输入信号改变状态。这种受时钟控制的触发器统称为同步触发器或钟控触发器。

同步 RS 触发器的逻辑图和逻辑符号如图 5.6 所示。该触发器由 4 个与非门构成,门 A 和门 B 构成基本 RS 触发器,门 C 和门 D 为控制门,CP 为控制信号。S 和 R 两个输入端分别是置 1 端和置 0 端,高电平有效。在逻辑符号图中,CP 输入端有标记“∧”,表示该输入端为脉冲信号。



(a) 逻辑图 (b) 逻辑符号

图 5.6 同步 RS 触发器

当 CP=0 时,门 C 和门 D 均输出高电平,门 A、B 构成的基本 RS 触发器处于保持状态。

当 CP=1 时,门 C、门 D 的输出 Z_1、Z_2 为:

$$Z_1 = \overline{S \cdot CP} = \overline{S} \tag{5.2}$$

$$Z_2 = \overline{R \cdot CP} = \overline{R} \tag{5.3}$$

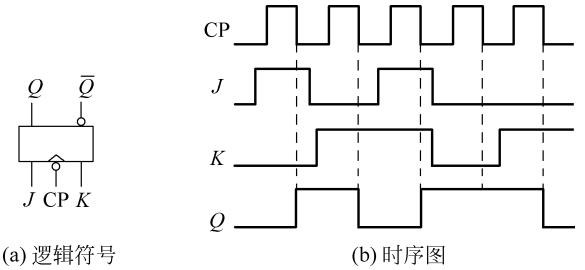
同步 RS 触发器的功能表如表 5.9 所示，和表 5.7 完全相同，只是增加了“CP＝1 时有效”这个条件。因此，同步 RS 触发器的特征方程与基本 RS 触发器的完全相同。

表 5.9　同步 RS 触发器功能表（CP＝1 时有效）

S	R	Q^{n+1}	功　能
0	0	Q^n	保持
0	1	0	置 0
1	0	1	置 1
1	1	不正常	不允许

5.2.3　JK 触发器

在 CP 操作下，根据输入信号 J、K，具有置 0、置 1、翻转和保持功能的电路，称为 JK 触发器，其逻辑符号和时序图如图 5.7 所示。JK 触发器的逻辑功能如表 5.10 所示。



(a) 逻辑符号　　(b) 时序图

图 5.7　JK 触发器

表 5.10　JK 触发器的次态真值表

J	K	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

其特性方程为

$$Q^{n+1} = J\overline{Q}^n + \overline{K}Q^n \text{（CP 下降沿到来后有效）} \tag{5.4}$$

JK 触发器的激励表如表 5.11 所示。

表 5.11 JK 触发器的激励表

Q^n	Q^{n+1}	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

5.2.4 D 触发器

在 CP 操作下,根据输入信号 D,具有置 0、置 1 功能的电路,称为 D 型触发器。根据对维持阻塞 D 触发器电路分析,D 触发器的逻辑功能如表 5.12 所示。

表 5.12 D 触发器的次态真值表

D	Q^n	Q^{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

其特性方程为

$$Q^{n+1}=D(\text{CP 上升沿到来后有效}) \tag{5.5}$$

表 5.13 为 D 触发器的激励表。

表 5.13 D 触发器的激励表

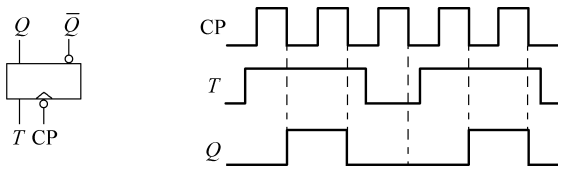
Q^n	Q^{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

5.2.5 T 触发器

在 CP 操作下,根据输入信号 T 的不同,具有保持和翻转功能的电路,称为 T 型触发器,其逻辑符号和时序图如图 5.8 所示。T 触发器的逻辑功能如表 5.14 所示。

表 5.14 T 触发器的次态真值表

T	Q^n	Q^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0



(a) 逻辑符号　　(b) 时序图

图 5.8　T 触发器

T 触发器是由 JK 触发器演变而来，即 J、K 相连作为信号输入端 T。因此其特性方程也可以由 JK 触发器的特性方程得出

$$\begin{aligned} Q^{n+1} &= J\overline{Q^n} + \overline{K}Q^n \\ &= T\overline{Q^n} + \overline{T}Q^n \\ &= T \oplus Q^n \text{（CP 下降沿到来后有效）} \end{aligned} \tag{5.6}$$

T 触发器的激励表如表 5.15 所示。

表 5.15　T 触发器的激励表

Q^n	Q^{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

5.3　同步时序电路的分析与设计

任何一个数字逻辑电路，总是从一组给定的输入，得到一组确定的输出。在组合电路中，多次重复出现的输入，可得到完全相同的输出。但是在时序电路中，如果一组输入多次重复出现，电路的输出却不尽相同，这是因为在同一输入的情况下，可能有不同的现态。因此，时序电路的分析与设计要比组合电路的分析与设计复杂得多。

设计一个同步时序电路，一般可按如下步骤进行。

(1) 根据逻辑问题的文字描述，建立原始状态表。进行这一步时，可先借助于原始状态图，再构成原始状态表。这一步得到的状态图和状态表是原始的，其中可能包含多余的状态。

(2) 采用状态化简方法，化简原始状态表。这一步得到一个用字符表示状态的简化状态表。

(3) 进行状态分配(或状态赋值)，即给予简化状态表中每个符号状态以二进制代码表示，这一步得到一个二进制状态表。

(4) 根据二进制状态表和选用的触发器的激励表，求电路的激励函数和输出函数。

(5) 根据激励函数和输出函数表达式，画出所要求的逻辑图。

同步时序电路的分析过程与设计过程正好相反，其主要步骤如下。

(1) 根据给定的时序电路,写出触发器的输入激励函数表达式以及电路的输出函数表达式,并由此画出激励矩阵和输出矩阵。

(2) 利用触发器的激励表(或状态表),将激励矩阵转换成 $\boldsymbol{Y}$ 矩阵。并与输出 $\boldsymbol{Z}$ 矩阵合并,得到 $\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵。

(3) 由 $\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵列出状态表,并画出状态图。

(4) 根据状态表或状态图,可作出网络的时间图或文字描述。

5.3.1 建立原始状态表

状态表是用来描述时序机的输入、状态和输出之间关系的表格。因而,建立状态表需要确定 3 个问题:一是电路应该包括几个状态;二是状态之间如何进行转换;三是怎样产生输出。解决这 3 个问题,至今尚没有一个系统的算法,目前所采用的方法仍然是直观的经验方法。

建立原始状态表可以先借助于原始状态图,画出原始状态图以后再列出原始状态表。

画原始状态图的过程是:首先假定一个初始状态 q_1;从这个初始状态 q_1 开始,每加入一个输入,就可确定其次态和输出;该次态可能是现态本身,也可能是已有的另一个状态,或是新增加的一个状态。继续这个过程,直到每一个现态向其次态的转换都已被考虑到,并且不再构成新的状态。输入也要考虑到各种可能取值。下面通过几个例子来说明上述方法。

【例 5.1】 列出一个模 5 加 1 和加 2 计数器的状态表。

显然这个计数器应有 5 个状态,设为 $q_0 \sim q_4$,以分别记住所输入的脉冲个数。由于这个计数器既可累加 1,又可累加 2,故需设定一个控制输入 x,并假定 $x=0$ 为加 1,$x=1$ 为加 2。输出 Z 为计满 5 时的进位(即溢出)信号。

经以上分析,可画出该计数器的状态图,如图 5.9 所示,并由此可列出状态表如表 5.16 所示。

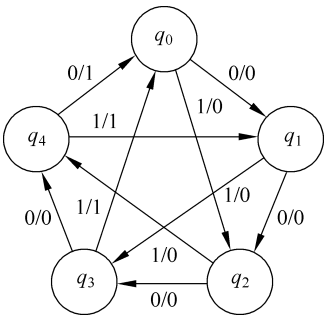


图 5.9 模 5 计数器状态图

表 5.16 模 5 计数器状态表

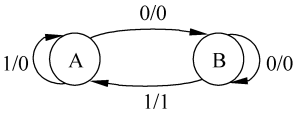
Q	x	
	0	1
q_0	q_1,0	q_2,0
q_1	q_2,0	q_3,0
q_2	q_3,0	q_4,0
q_3	q_4,0	q_0,1
q_4	q_0,1	q_1,1

【例 5.2】 设计一个“01”序列检测器。该电路有一个输入 x 和一个输出 Z。输入 x 为一串随机信号,当其中出现“01”序列时,检测器能识别出来,并产生输出信号 $Z=1$;对于其他输入情况,输出均为 0。例如

输入序列 10011010001

输出序列 00010010001

该时序电路应能记住“01”序列中先出现的第一个元素“0”，又能记住后出现的第二个元素“1”，这时产生输出。因此，原始状态图可这样建立，如图 5.10(a)所示。假定电路处于初态 A，若输入为 1，因为“1”不是被识别的输入序列“01”的第一个元素，所以电路停留在状态 A 并输出 0；若输入为 0，这是被识别的输入序列“01”的第一个元素，电路转至状态 B 并输出 0。当电路处于状态 B 时，若输入为 0，这不是被识别的“01”序列的第二个元素，而仍是第一个元素，所以电路仍停留在状态 B 并输出 0；若输入为 1，这是被识别的“01”序列的第二个元素，这时已检出所要求的序列，电路输出为 1，且由于输入序列“01”已检测完毕，因此电路回到初始状态 A。由状态图可列出状态表，如图 5.10(b)所示。



y	x	
	0	1
A	B,0	A,0
B	B,0	A,1

(a) 状态图　　(b) 状态表

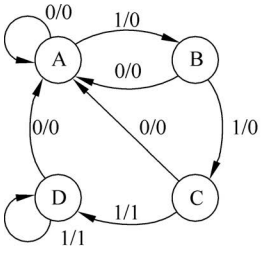
图 5.10　“01”序列检测器

【例 5.3】 设计一个“111”序列检测器。该电路有一个输入 x 和一个输出 Z。输入 x 为一连串随机信号，每当其中出现有 3 个或 3 个以上连续脉冲时，检测器输出为 1；其他情况，输出均为 0。例如

输入序列 101100111011110

输出序列 000000001000110

可以想象，电路有一个计数器，能记住连续输入 1 的个数。假定电路处于初态 A，当第一次输入 1 时，电路由状态 A 转入状态 B，并输出 0；第二个信号继续输入 1 时，电路由状态 B 转入状态 C，并输出 0；第三个信号继续输入 1 时，电路由状态 C 转至状态 D，并输出 1；此后若电路继续输入 1 时，电路仍停留在状态 D，并输出 1。这里，状态 B、C、D 为记录连续输入 1 的个数的状态，当接收到一个 0 之后，都表示序列检测器需要重新记录连续输入 1 的个数，故电路将回到初始状态 A。图 5.11 为“111”序列检测器的状态图和状态表。



y	x	
	0	1
A	A,0	B,0
B	A,0	C,0
C	A,0	D,1
D	A,0	D,1

(a) 状态图　　(b) 状态表

图 5.11　“111”序列检测器

最后，必须强调指出，建立原始状态表或状态图应着眼于正确性，要尽可能不遗漏一个状态和输入的可能取值。至于所设定的状态是否多余，不必过多注意，多余状态可以通过下

面要讲的状态化简来去掉。

5.3.2 状态表的化简

在建立原始状态表的过程中,可能引入多余的状态。显然,网络的状态越多,所需要的存储元件就越多。因此,在得到原始状态表后,设计的下一步工作就是进行状态表的化简,以尽量减少所需状态的数目。本小节将介绍状态表的化简方法。先介绍状态表化简的基本原理,然后分别介绍完全定义机和不完全定义机两类状态表化简的具体步骤。

1. 状态表化简的基本原理

在前面建立原始状态表的过程中可看出,设置电路状态的目的在于利用这些状态记住输入的历史情况,以对其后的输入产生不同的输出。如果所设置的两个状态,对任一输入序列产生的输出序列完全相同,则这两个状态可以合并为一个状态。状态表的化简就是根据这一原理进行的。例如,上一小节的"111"序列检测器的状态表,如图5.11(b)所示。表中,状态 C 和 D 在现输入 x 为0或1的情况下,所产生的输出分别相同,即

$$Z(C,0)=Z(D,0)=0,Z(C,1)=Z(D,1)=1$$

且所建立的次态也分别相同,即

$$N(C,0)=N(D,0)=A,N(C,1)=N(D,1)=D$$

在现输入下所建立的次态相同,这意味着从现态 C 和 D 开始,对于其后的所有输入序列所产生的输出序列一定都相同,故表中的状态 C 和 D 可以合并为一个状态。这样,原始状态表可化简为3个状态的状态表,如表5.17所示。

表5.17 化简后的状态表

y	x	
	0	1
A	A,0	B,0
B	A,0	C,0
C	A,0	C,1

上面是一个比较简单的例子,用以说明两个状态进行合并的基本原理。但是,两个状态可以合并不限于这一简单情况,还有更复杂的情况。下面通过两个例子进一步说明两个状态进行合并的条件。

【例5.4】 化简表5.18所示的原始状态表。

表5.18 原始状态表

y	x	
	0	1
A	C,1	B,0
B	C,1	E,0
C	B,1	E,0
D	D,1	B,1
E	E,1	B,1

考察表中的状态 B 和 C，在现输入 x 为 0 或 1 下，它们所产生的输出分别相同，即

$$Z(B,0)=Z(C,0)=1,Z(B,1)=Z(C,1)=0$$

而所建立的次态在 $x=1$ 时是相同的，即都为 E；在 $x=0$ 时分别等于对方的现态，即次态为现态的交错，可表示为

$$N(B,0)=C,N(C,0)=B$$

在这种情况下，尽管 $x=0$ 时现态 B 和 C 所建立的次态相应为 C 和 B，但由于这两个状态在不同输入下所产生的输出分别相同，故同样满足上述状态合并的条件。

同理，状态表中的状态 D 和 E 在现输入 x 为 0 或 1 下，它们所产生的输出分别相同。而所建立的次态在 $x=1$ 时是相同的，在 $x=0$ 时分别等于现态本身，即

$$N(D,0)=D,N(E,0)=E$$

在这种情况下，同样满足状态合并的条件。

设状态 B 和 C 合并为状态 q_1，D 和 E 合并为状态 q_2，且令状态 A 为状态 q_0，记为

$$q_0=\{A\},q_1=\{B,C\},q_2=\{D,E\}$$

代入表 5.18，则可以得到简化后的状态表如表 5.19 所示。

表 5.19　简化后的状态表

y	x	
	0	1
q_0	q_1,1	q_1,0
q_1	q_1,1	q_2,0
q_2	q_2,1	q_1,1

【例 5.5】 化简表 5.20 所示的原始状态表。

表 5.20　原始状态表

y	x	
	0	1
A	E,0	D,0
B	A,1	F,0
C	C,0	A,1
D	B,0	A,0
E	D,1	C,0
F	C,0	D,1

先考察状态 C 和 F。由表可知，不论输入 x 为 0 或 1，它们所产生的输出分别相同。当 $x=0$ 时，它们所建立的次态也相同；但当 $x=1$ 时，它们所建立的次态却不同。

$$N(C,1)=A,N(F,1)=D$$

因此，状态 C 和 F 能否合并取决于状态 A 和 D 能否合并。为此，需要进一步追踪 A 和 D 是否满足合并条件。

由表 5.20 可知，不论输入 x 为 0 或 1，由现态 A 和 D 所产生的输出分别相同。当 $x=1$ 时，它们所建立的次态为现态的交错；但当 $x=0$ 时，它们所建立的次态却不同：

$$N(A,0)=E,N(D,0)=B$$

因此,状态 A 和 D 能否合并取决于状态 E 和 B 能否合并。为此,需继续追踪 B 和 E 是否满足合并条件。

由表可知,不论输入 x 为0或1,由现态 B 和 E 产生的输出分别相同。当 $x=0$ 时,它们所建立的次态不同:

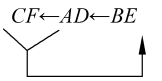
$$N(B,0)=A,N(E,0)=D$$

当 $x=1$ 时,它们所建立的次态也不同:

$$N(B,1)=F,N(E,1)=C$$

因此,状态 B 和 E 能否合并取决于状态 A 和 D 及状态 C 和 F 能否合并。

至此,发现状态 CF、AD 及 BE 能否各自合并,出现如下循环关系:



显然,由于这个循环中的各对状态,在不同的现输入下所产生的输出是分别相同的,因而从循环中的某一状态对出发,都能保证在所有的输入序列下所产生的输出序列均相同。因此,循环中的各对状态是可以合并的。令

$$q_1=\{A,D\},q_2=\{B,E\},q_3=\{C,F\}$$

代入表5.20,则得简化后的状态表如表5.21所示。

表5.21 简化的状态表

y	x	
	0	1
q_1	q_2,0	q_1,0
q_2	q_1,1	q_3,0
q_3	q_3,0	q_1,1

综上所述,状态表中两个状态可以合并为一个状态的条件,归纳如下。

(1) 在任一现输入下,现输出分别相同。

(2) 在所有不同的现输入下,次态分别为下列情况之一。

① 两个次态完全相同。

② 两个次态为其现态本身或交错。

③ 两个次态的某一后继状态可以合并。

④ 两个次态为状态对循环中的一个状态对。

上述两个条件必须同时满足,而第一个条件是状态合并的必要条件。

显然,从原始状态表可以很容易判断任何两个状态是否满足第一条件,但不太容易判别是否满足第二条件。下面介绍的两种状态表的化简方法,就是为解决这一问题而提出来的。首先介绍完全定义机状态表的化简方法,然后再介绍不完全定义机状态表的化简方法。

2. 完全定义机状态表的化简方法

1) 等价的概念

在介绍具体方法之前,先引入等价的概念。

(1) 等价状态。设 q_a 和 q_b 是时序机状态表的两个状态,如果从 q_a 和 q_b 开始,任何加

到时序机上的输入序列均产生相同的输出序列，则称状态 q_a 和 q_b 为等价状态或等价状态对，并记为(q_a,q_b)或$\{q_b,q_a\}$。显然，满足前面所述合并条件的两个状态是等价状态。

(2) 等价状态的传递性。若状态 q_1 和 q_2 等价，状态 q_2 和 q_3 等价，则状态 q_1 和 q_3 也等价，记为

$$(q_1,q_2),(q_2,q_3)\rightarrow(q_1,q_3)$$

(3) 等价类。彼此等价的状态集合，称为等价类。例如，若有(q_1,q_2)和(q_2,q_3)，根据等价状态的传递性，则有等价类(q_1,q_2,q_3)。

(4) 最大等价类。若一个等价类不是任何别的等价类的子集，则此等价类称为最大等价类。

显然，状态表化简的根本任务在于从原始状态表中找出最大等价类。

2) 化简方法——隐含表法

它的基本思想是：先对原始状态表中的各状态进行两两比较，找出等价状态对；然后利用等价的传递性，得到等价类；最后确定一组等价类，以建立最简状态表。

隐含表法的具体步骤如下。

(1) 画隐含表。隐含表的格式如图 5.12 所示。设原始状态表有 n 个状态$q_1\sim q_n$，在隐含表的水平方向标以状态 $q_1,q_2,\cdots,q_{n-1}$，垂直方向标以 $q_2,q_3,\cdots,q_n$。也就是隐含表垂直方向“缺头”，水平方向“少尾”。隐含表中的每一个小方格表示一个状态对(q_i,q_j)。

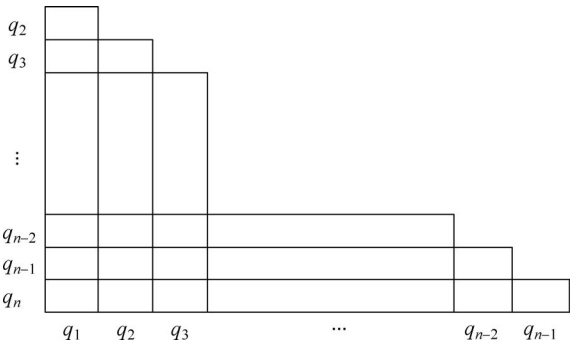


图 5.12　隐含表格式

(2) 顺序比较。顺序比较隐含表中各状态之间的关系，比较结果有以下 3 种情况。

① q_i 和 q_j 输出完全相同，次态也相同，或者为现态本身或者交错，表示 q_i 和 q_j 等价，在隐含表对应方格内标以“√”。

② q_i 和 q_j 输出不相同，表示 q_i 和 q_j 不等价，在对应方格内标以“×”。

③ q_i 和 q_j 输出完全相同，但其次态既不相同，又不交错，表示 q_i 和 q_j 是否等价，还待进一步考察，在对应方格内标以 q_i 和 q_j 的次态对。

(3) 关联比较。关联比较是要确定上一步待考察的次态对是否等价，并由此来确定原状态对是否等价。这一步应在隐含表上直接进行，以追踪后续状态对的情况。若后续状态对等价或出现循环，则这些状态对都是等价的；若后续状态对中出现不等价，则在它以前的状态对都是不等价的。

(4) 列最大等价类，作最简状态表。关联比较后，可以确定哪些状态是“等价对”，再由等价对构成“等价类”和“最大等价类”。不与其他任何状态等价的单个状态也是一个最大等价类。每个最大等价类可以合并为一个状态，并以一个新符号表示。这样，由一组新符号构成的状态表，便是所求的最简状态表。

【例 5.6】 化简图 5.13(a)所示的原始状态表。

化简步骤如下。

① 画隐含表,如图 5.13(b)所示。

② 顺序比较,结果如图 5.13(b)所示。

③ 关联比较,考察状态对 AB,若要 AB 等价,就需要 BC 等价。从隐含表上可直接看出 BC 不等价,因此 AB 也不等价,在相应方格上打“×”号。同理,BD 也不等价,如图 5.13(c)所示。

④ 列最大等价类。由关联比较结果,可得最大等价类为

$$(A,D),(B),(C)$$

令

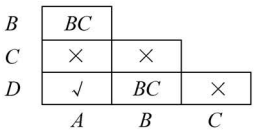
$$a=\{A,D\},b=\{B\},c=\{C\}$$

由此可做出简化状态表,如图 5.13(d)所示。

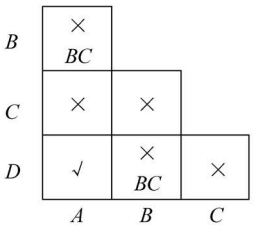
【例 5.7】 化简图 5.14(a)的原始状态表。

y	x	
	0	1
A	D,0	B,0
B	D,0	C,0
C	D,0	C,1
D	D,0	B,0

(a) 原始状态表



(b) 隐含表



(c) 关联比较

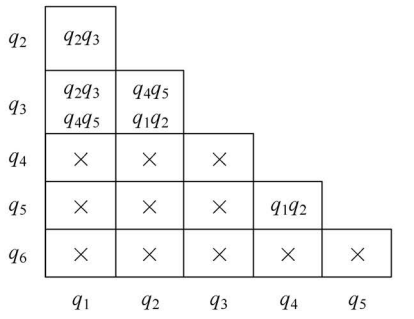
y	x	
	0	1
a	a,0	b,0
b	a,0	c,0
c	a,0	c,1

(d) 简化状态表

图 5.13　完全定义机状态表的化简方法

y	x		
	I_1	I_2	I_3
q_1	q_3,0	q_4,0	q_2,0
q_2	q_2,0	q_4,0	q_3,0
q_3	q_2,0	q_5,0	q_1,0
q_4	q_1,1	q_6,1	q_6,0
q_5	q_2,1	q_6,2	q_6,0
q_6	q_1,1	q_5,1	q_4,1

(a) 原始状态表



(b) 隐含表

y	x		
	I_1	I_2	I_3
A	A,0	B,0	A,0
B	A,1	C,1	C,0
C	A,1	B,1	B,1

(c) 简化状态表

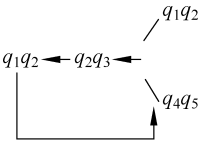
图 5.14　完全定义机状态表的化简方法

化简步骤如下。

① 画隐含表，如图 5.14(b)所示。

② 顺序比较。原始状态表中，所有输入（I_1、I_2 和 I_3）下的输出和次态都要比较，比较结果列于图 5.14(b)中。

③ 关联比较。考察状态 q_1 和 q_2 的后继状态，出现如下循环关系：



由于在循环链中各状态对的输出都是相同的，所以得到下列等价状态对：

$$(q_1,q_2),(q_2,q_3),(q_4,q_5)$$

由隐含表可看出，由于 q_2、q_3 和 q_4、q_5 等价，因而 q_1、q_3 也是等价的。

④ 列最大等价类。由关联比较结果可得到最大等价类为：

$$(q_1,q_2,q_3),\quad (q_4,q_5),(q_6)$$

令　　$A=\{q_1,q_2,q_3\},\quad B=\{q_4,q_5\},C=\{q_6\}$

可以作出简化状态表如图 5.14(c)所示。

3. 不完全定义机状态表的化简方法

1）相容的概念

在讨论不完全定义机状态表的化简方法之前，先引入相容的概念。

（1）相容状态。设 q_i 和 q_j 是不完全定义机状态表中的两个状态，如果它们的输出和次态在两者有定义时满足前面叙述的两个合并条件，则称 q_i 和 q_j 是相容状态，或称相容状态对。

例如，表 5.22 为一个不完全定义机状态表。表中，状态 A 和 B 为相容状态，B 和 C 也为相容状态，A 和 C 就不是相容状态。

表 5.22　不完全定义机状态表

y	x	
	0	1
A	B,0	d,0
B	A,d	B,0
C	d,1	B,0
D	C,1	D,d

（2）相容状态无传递性。若状态 q_i 和 q_j 相容，状态 q_j 和 q_k 相容，则状态 q_i 和 q_k 不一定相容，如表 5.22 中，A 和 B 相容，且 B 和 C 相容，但 A 和 C 却不相容。

（3）相容类。所有状态之间都是两两相容的状态集合，称为相容类。

（4）最大相容类。若一个相容类不是任何其他相容类的子集时，则称此相容类为最大

相容类。

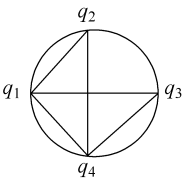
为了从相容状态方便地找到最大相容类,这里介绍一种状态合并图法。合并图是这样构成的:先将原始状态表中的每个状态以“点”的形式分布在一个圆周上,然后把各个相容状态的两个“点”用直线连起来,那么所得到的各“点”间都有连线的“多边形”就是一个相容类。如果这个相容类不包含在任何其他相容类之中,它就是一个最大相容类。

例如,设有一个不完全定义机的状态表如图 5.15(a)所示。由表可找到相容状态对为

$$(q_1,q_2),(q_1,q_3),(q_3,q_4),(q_1,q_4),(q_2,q_4)$$

y	x	
	0	1
q_1	q_1,0	q_4,d
q_2	q_1,0	q_4,0
q_3	q_1,0	q_4,1
q_4	q_1,0	q_3,d

(a) 状态表



(b) 合并图

图 5.15 不完全定义机的状态表及合并图

它的状态合并图可以这样构成:先把状态表的 4 个状态 q_1、q_2、q_3 和 q_4 以“点”的形式标在圆周上;然后,由各相容状态对用直线两两连起来,这样就得到状态合并图如图 5.15(b)所示。从图上可以看出,有两个各点间都有连线的“多边形”。因此,求得下列两个相容类:

$$(q_1,q_2,q_4),\quad (q_1,q_3,q_4)$$

由图还可看出,上述两个相容类不包含在其他任何相容类之中。因此,它们是最大相容类。

2) 化简方法——隐含表法

用隐含表法(也称相容法)化简不完全定义机状态表的过程与化简完全定义机状态表的过程大致相同,只是在最后构成最简状态表时有所不同。这点应予以注意。化简的具体步骤如下。

(1) 画隐含表,寻找相容状态对。隐含表的画法与完全定义机相同。画好隐含表后,逐一判别状态表中每对状态 q_i 和 q_j 的相容关系,判别结果有以下 3 种情况。

① 若 q_i 和 q_j 两个状态对应的输出(除随意项外)不相同,则表示这两个状态不相容,在隐含表的相应方格中标以“×”号。

② 若 q_i 和 q_j 的输出(除随意项外)相同,且次态相同、交错,或者包含随意项,则表示这两个状态相容,在相应的方格内标以“√”号。

③ 若 q_i 和 q_j 的输出(除随意项外)相同,但次态尚不能直接确定是否相容,则表示这两

个状态是否相容还待进一步考察，在对应的方格内填入其对应的不同次态对，这是其相容的条件。此时，利用隐含表继续追踪待定次态对。如果后续状态对相容或出现循环，则这些状态对都是相容的；如果后续状态对出现不相容，则这些状态对都是不相容的。

(2) 画状态合并图，找最大相容类。

(3) 作最小化状态表。这一步的任务是要从上面求得的最大相容类(或相容类)中选出一组能覆盖原始状态表全部状态且个数最少的相容类，这一组相容类必须满足如下 3 个条件：

① 覆盖性，即该组相容类应能覆盖原始状态表的全部状态。

② 最小性，即该组相容类的数目应为最小。

③ 闭合性，即该组相容类中的任一个相容类，它在原始状态表中任一输入下产生的次态应该属于该组内的某一个相容类。

选出这组满足上述三条件的相容类后，每个相容类用一个状态符号表示。这样，由这组状态就可以构成最小化状态表。

下面举例说明不完全定义机状态表化简的方法。

【例 5.8】 化简图 5.16(a)所示的原始状态表。

化简步骤如下。

① 画隐含表，找相容状态对。隐含表如图 5.16(b)所示。由隐含表可得到相容状态对如下：

$$(q_1,q_3),(q_2,q_6),(q_3,q_5),(q_1,q_5),(q_2,q_4),(q_4,q_6)$$

② 画合并图，找最大相容类。状态合并图如图 5.16(c)所示。由状态合并图可得到相容类如下：

$$(q_1,q_3,q_5),(q_2,q_4,q_6)$$

③ 作最小化状态表。先作覆盖闭合表，如图 5.16(d)所示。选择(q_1,q_3,q_5)和(q_2,q_4,q_6)这两个相容类，它们覆盖了原始状态表的全部状态；而且每个相容类在任一输入下次态属于这两个相容类中某一个；此外，这两个相容类不能再少了。因此，它们满足覆盖、闭合和最小 3 个条件。令 $q_1'=\{q_1,q_3,q_5\}$，$q_2'=\{q_2,q_4,q_6\}$，作出最小优化状态表如图 5.16(e)所示。

【例 5.9】 化简图 5.17(a)所示的原始状态表。

化简步骤如下。

① 画隐含表，找相容状态对。隐含表如图 5.17(b)所示。由隐含表可得相容状态对如下：

$$(q_1,q_2),(q_1,q_3),(q_1,q_4),(q_1,q_5),(q_2,q_3),(q_3,q_4),(q_4,q_5)$$

② 画状态合并图，找最大相容类。状态合并图如图 5.17(c)所示。由状态合并图可得到最大相容类如下：

$$(q_1,q_2,q_3),(q_1,q_3,q_4),(q_1,q_4,q_5)$$

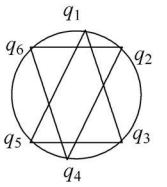
③ 作最小状态表。先作覆盖闭合表，如图 5.17(d)所示。这 3 个最大相容类满足覆盖

y	x	
	0	1
q_1	q_2,0	q_5,0
q_2	q_2,1	q_5,d
q_3	q_6,0	q_3,0
q_4	q_2,1	q_1,1
q_5	q_4,0	q_3,d
q_6	q_4,1	q_3,1

(a) 原始状态表

	q_1	q_2	q_3	q_4	q_5
q_2	×				
q_3	q_2q_6 q_3q_5	×			
q_4	×	q_1q_5	×		
q_5	q_2q_4 q_3q_5	×	q_4q_6	×	
q_6	×	q_2q_4 q_3q_5	×	q_2q_4 q_1q_3	×

(b) 隐含表



(c) 状态合并图

相容状态集	覆盖						闭合	
	q_1	q_2	q_3	q_4	q_5	q_6	x=0	x=1
$q_1q_3q_5$	q_1		q_3		q_5		$q_2q_6q_4$	q_3 q_5
$q_2q_4q_6$		q_2		q_4		q_6	q_2 q_4	$q_5q_1q_3$

(d) 覆盖闭合表

y	x	
	0	1
q_1'	q_2',0	q_1',0
q_2'	q_2',1	q_1',1

(e) 最小化状态表

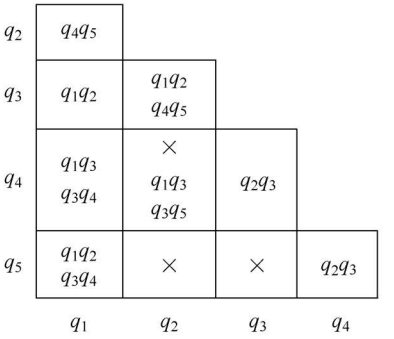
图 5.16 不完全定义机状态表的化简方法示例 1

闭合条件,但是否最小呢? 由于状态 q_2 仅属于(q_1,q_2,q_3),状态 q_5 仅属于(q_1,q_4,q_5),而相容类(q_1,q_2,q_3)和(q_1,q_4,q_5)覆盖了原始状态表的全部状态。因此,可以考虑选择这两个相容类。但在进一步检查闭合性时,发现(q_1,q_4,q_5)在输入 $x=0$ 时次态为 q_3q_4,它不属于所选的两个相容类中的任何一个。这说明选择相容类(q_1,q_2,q_3)和(q_1,q_4,q_5)不满足闭合性条件。如果选择相容类(q_1,q_2,q_3)和(q_4,q_5),可以发现它是满足覆盖、闭合和最小这 3 个条件的,如图 5.17(e)所示。这是唯一的一组解。令 $q_1'=\{q_1,q_2,q_3\}$,$q_2'=\{q_4,q_5\}$,作出最小化状态表如图 5.17(f)所示。

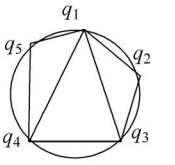
这个例子说明,有时选用最大相容类不一定能取得最小化。究竟是选用相容类还是最大相容类,应视具体情况而定。

y	x	
	0	1
q_1	q_4,d	q_1,d
q_2	$q_5,0$	q_1,d
q_3	$q_4,0$	q_2,d
q_4	q_3,d	q_3,d
q_5	$q_3,1$	q_2,d

(a) 原始状态表



(b) 隐含表



(c) 状态合并图

相容状态集	覆　盖					闭　合	
	q_1	q_2	q_3	q_4	q_5	$x=0$	$x=1$
$q_1q_2q_3$	q_1	q_2	q_3			$q_4\ q_5$	$q_1\ q_2$
$q_1q_3q_4$	q_1		q_3	q_4		$q_3\ q_4$	$q_1q_2q_3$
$q_1q_4q_5$	q_1			q_4	q_5	$q_3\ q_4$	$q_1q_2q_3$

(d) 覆盖闭合表一

相容状态集	覆　盖					闭　合	
	q_1	q_2	q_3	q_4	q_5	$x=0$	$x=1$
$q_1q_2q_3$	q_1	q_2	q_3			q_4q_5	q_1q_2
q_4q_5				q_4	q_5	q_3	q_2q_3

(e) 覆盖闭合表二

y	x	
	0	1
q_1' q_2'	$q_2',0$ $q_1',1$	q_1',d q_1',d

(f) 最小化状态表

图 5.17　不完全定义机状态表的化简方法示例 2

最后,在结束本小节讨论之前,还需指出两点:

① 不完全定义机的状态表中,两状态的相容只对可应用输入序列有效,而不是对所有的输入序列都有效。这是由于不完全定义机状态表中状态的随意项引起的。所谓可应用输入序列是指:以 q_i 为初态,若某一输入序列中的每一个输入所建立的状态都是确定的,则该输入序列为状态 q_i 的可应用输入序列。例如,在表 5.22 中,对于状态 C,输入序列 1000 是可应用输入序列;而对于输入序列 1010 是不可应用输入序列。这是因为

输入序列	1 0 0 0	1 0 1 0
次态 C	$\underbrace{B\ A\ B\ A}$	$\underbrace{C\ B\ A\ d}$
	确定	不确定

② 完全定义机可看作是不完全定义机的特例。换句话说,不完全定义机更具有一般性。因而,不完全定义机状态表的化简方法也适用于完全定义机,两者可统一成一种方法。

5.3.3 状态分配

在求得时序电路的最简状态表后,下一个设计步骤就是进行状态分配。所谓状态分配,或称状态编码、状态赋值,就是给最简状态表中的每个符号状态指定一个二进制代码,形成二进制状态表。

状态分配的任务是要解决两个问题:一是根据简化状态表给定的状态数,确定所需触发器的数目;二是给每个状态指定二进制代码,以使所设计的电路最简单。

在实际工作中,设计人员主要还是凭经验,依据一定的原则,寻求接近最佳的状态分配方案。下面介绍一种状态分配的经验方法。

这里介绍的经验方法是基于如下思想:在选择状态编码时,尽可能地使次态和输出函数在卡诺图上“1”的分布为相邻,以便形成较大的圈。这种方法主要根据以下三条相邻原则。

(1) 在相同输入条件下,次态相同,现态应相邻编码。所谓相邻编码,是指两个状态的二进制代码仅有一位不同。

(2) 在不同输入条件下,对于同一现态,次态应相邻编码。

(3) 输出完全相同,两个现态应相邻编码。

在以上三条原则中,第一条最重要,应优先考虑。下面举例说明上述原则的用法。

【例 5.10】 对表 5.23 所示的简化状态表进行状态分配。

表 5.23 简化状态表

y	x	
	0	1
q_1	q_3,0	q_4,0
q_2	q_3,0	q_1,0
q_3	q_2,0	q_4,0
q_4	q_1,1	q_2,1

状态表中共有 4 个状态 q_1、q_2、q_3 和 q_4,其状态编码确定过程如下:

根据原则(1),q_1q_2、q_1q_3 应相邻编码;

根据原则(2)，q_3q_4、q_1q_3、q_2q_4、q_1q_2 应相邻编码；

根据原则(3)，q_1q_2、q_1q_3、q_2q_3 应相邻编码。

综合上述要求，q_1q_2、q_1q_3 应给予相邻编码，因为这是三条原则都要求的。可以借用卡诺图，很易得到满足上述相邻要求的状态分配方案，如图 5.18 所示。其中，y_1 和 y_2 表示触发器。因此，由图 5.18 可得状态编码为

$$q_1=00, q_2=01, q_3=10, q_4=11$$

将上述编码代入表 5.23 的简化状态表，就得到表 5.24 所示的二进制状态表，这就完成了状态分配。当然，上述分配方案不是唯一的。

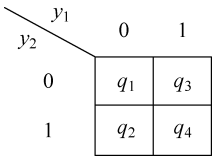


图 5.18　状态分配方案

表 5.24　二进制状态表

$y_1\ y_2$	x	
	0	1
0　0	10,0	11,0
0　1	10,0	00,0
1　0	01,0	11,0
1　1	00,1	01,1

必须指出，状态分配三条原则在大多数情况下是有效的，由它所得到的电路是比较简单的。但是，由于问题的复杂性，有时得到的结果并不令人满意，这是在实际使用中要注意的。还有一点要指出，对于同步时序电路来说，不同的状态分配方案并不影响网络工作的稳定性，仅影响网络的复杂程度。

5.3.4　确定激励函数和输出函数

在完成状态分配以后，时序电路设计的下一步工作就是确定激励函数和输出函数，并据此可画出逻辑图。

由状态分配所得到的二进制状态表，反映了次态 Y 与 x、y 的关系，也反映了输出 Z 与 x、y 的关系。当触发器选定后，由于 y 和 Y 在二进制状态表中是已知的，根据触发器的激励表，就可以求出激励函数表达式。输出函数的表达式可直接从二进制状态表求得。

确定激励函数和输出函数的具体步骤如下。

(1) 将二进制状态表变换成 ***Y*-*Z*** 矩阵。为了便于从二进制状态表求得函数的表达式，将这个表的变量取值按 Gray 码的顺序排列，这样就变换成了卡诺图的形式。把这种能反映函数 $Y_i=f_i(x,y)$ 和 $Z_i=g_i(x,y)$ 的卡诺图称为 ***Y*-*Z*** 矩阵。

例如，表 5.24 的二进制状态表变换成 ***Y*-*Z*** 矩阵如表 5.25 所示。

表 5.25　*Y*-*Z* 矩阵

$y_1\ y_2$	x	
	0	1
0　0	10,0	11,0
0　1	10,0	00,0
1　1	00,1	01,1
1　0	01,0	11,0

(2) 由 ***Y*-*Z*** 矩阵变换成激励矩阵和输出矩阵。

Y*-*Z 矩阵可看成由 ***Y*** 矩阵和 ***Z*** 矩阵两部分构成。***Y*** 矩阵给出每一现态 y_i 的次态值 Y_i,而由现态 y_i 向次态 Y_i 的转换是依靠触发器的输入激励,这个激励可根据所选触发器的激励表来确定。把 ***Y*** 矩阵中的次态值 Y_i 代之以相应触发器的激励值,就得到激励函数的卡诺图形式,这个卡诺图称为激励矩阵。由 ***Y*-*Z*** 矩阵的另一部分 ***Z*** 矩阵,直接可得输出矩阵。

例如,假定选 RS 触发器来实现表 5.25 的 ***Y*-*Z*** 矩阵,则它的激励矩阵和输出矩阵如表 5.26 和表 5.27 所示。

表 5.26 激励矩阵 R_1S_1 和 R_2S_2

y_1 y_2	x	
	0	1
0 0	01,d0	01,01
0 1	01,10	d0,10
1 1	10,10	10,0d
1 0	10,01	0d,01

表 5.27 输出矩阵 *Z*

y_1 y_2	x	
	0	1
0 0	0	0
0 1	0	0
1 1	1	1
1 0	0	0

(3) 由激励和输出矩阵,求激励函数和输出函数。

激励矩阵可以看成是各个输入激励填在同一个卡诺图上构成的。因此,在求各个激励函数时,只要分别画出各个输入激励的卡诺图,并由此写出各个激励函数的最简表达式。同理,由输出矩阵,可写出输出函数的最简表达式。

例如,由表 5.26 的激励矩阵 R_1S_1 和 R_2S_2,可以分别画出 R_1、S_1、R_2、S_2 4 个卡诺图,并由此写出这 4 个激励函数的表达式。表中 R_1 和 S_1 是触发器 y_1 的输入,R_2 和 S_2 是触发器 y_2 的输入。同理,由表 5.27 的输出矩阵,可写出输出函数的表达式。

将上述求激励函数和输出函数的过程简单表示如下:

Y*-*Z 矩阵 { ***Y*** 矩阵→激励矩阵→激励函数 ↑ 触发器激励表 ***Z*** 矩阵 → 输出函数

下面举例说明确定激励函数和输出函数的方法。

【例 5.11】 完成 5.3 节例 5.3“111”序列检测器的设计。

设计过程如下。

① 建立原始状态表。该序列检测器的原始状态表如图 5.11(b)所示。

② 状态化简。该序列检测器的简化状态表如表 5.17 所示。

③ 状态分配。简化状态表共有 3 个状态,所以需要用两位触发器 y_1 和 y_2,根据状态分

配的原则，一种较好的分配方案如图 5.19 所示。根据这种状态分配方案，A 为 00，B 为 01，C 为 10。于是，可得二进制状态表(即 $\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵)，如表 5.28 所示。

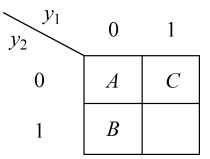


图 5.19　状态分配方案

表 5.28　Y-Z 矩阵 Y_1Y_2-Z

y_1y_2	x	
	0	1
0　0	00,0	01,0
0　1	00,0	10,0
1　1	dd,d	dd,d
1　0	00,0	10,1

④ 求激励函数和输出函数。若选用 JK 触发器作为存储元件，则根据 JK 触发器的激励表，可得到网络的激励矩阵如表 5.29 所示。

表 5.29　激励矩阵 J_1K_1，J_2K_2

y_1y_2	x	
	0	1
0　0	0d,0d	0d,1d
0　1	0d,d1	1d,d1
1　1	dd,dd	dd,dd
1　0	d1,0d	d0,0d

分别画出各激励函数 J_1、K_1、J_2、K_2 和输出函数 Z 的卡诺图，如图 5.20 所示。由此可得激励函数和输出函数为

$$J_1=xy_2,K_1=\bar{x},\quad J_2=x\bar{y}_1,\quad K_2=1,Z=xy_1$$

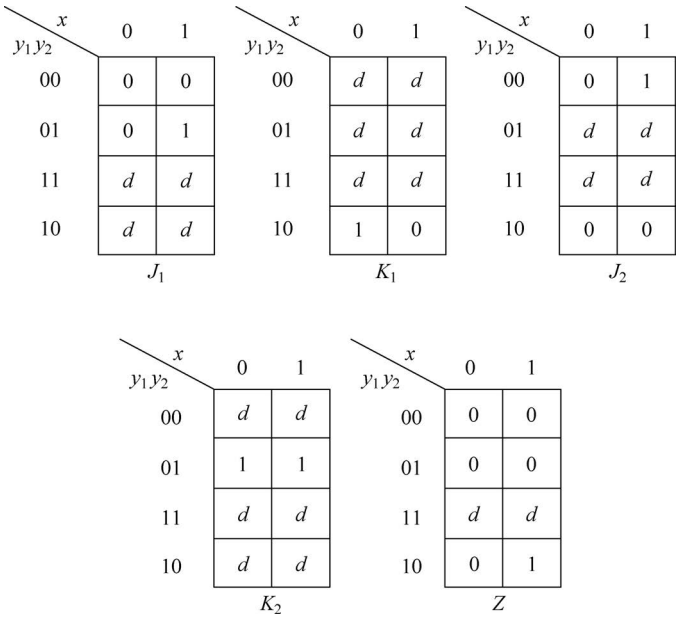


图 5.20　卡诺图

⑤ 画逻辑图。根据所求得的激励函数和输出函数,可画出“111”序列检测器的逻辑电路图,如图 5.21 所示。

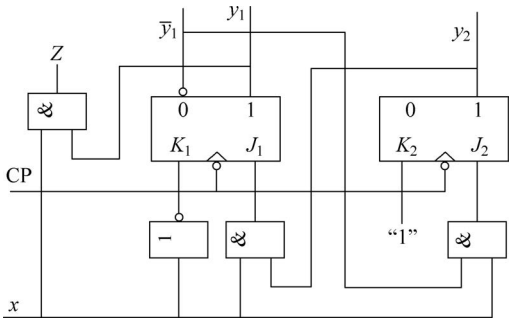


图 5.21 “111”序列检测器逻辑图

5.3.5 分析与设计举例

【例 5.12】 分析图 5.22 所示同步时序电路的逻辑功能。

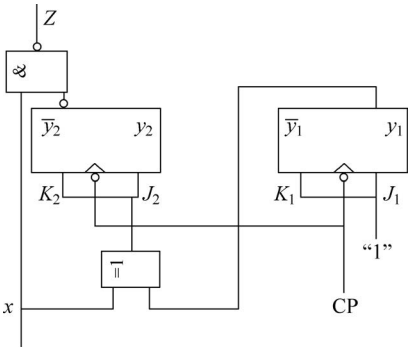


图 5.22 同步时序电路

根据电路逻辑图,可写出激励函数和输出函数表达式为

$$J_2=K_2=x\oplus y_1=x\bar{y}_1+\bar{x}y_1,J_1=K_1=1,\quad Z=\overline{x\bar{y}_2}=\bar{x}+y_2$$

根据这些表达式,分别画出它们的卡诺图如图 5.23 所示。将 J_2、K_2、J_1、K_1 的卡诺图合并画在一个卡诺图上,便得到网络的激励矩阵,如表 5.30 所示。

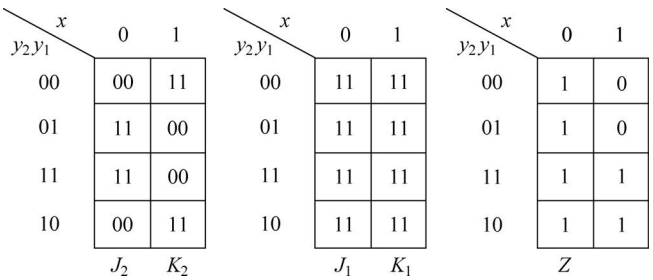


图 5.23 J、K、Z 的卡诺图

表 5.30　激励矩阵 J_2K_2、J_1K_1

y_2y_1	x	
	0	1
0　0	00,11	11,11
0　1	11,11	00,11
1　1	11,11	00,11
1　0	00,11	11,11

根据 JK 触发器的状态表将激励矩阵和输出 $\boldsymbol{Z}$ 矩阵变换成 $\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵，如表 5.31 所示。

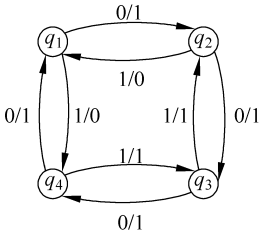
表 5.31　$\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵

y_2y_1	x	
	0	1
0　0	01,1	11,0
0　1	10,1	00,0
1　1	00,1	10,1
1　0	11,1	01,1

由 $\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵列状态表，画状态图。令编码 00、01、10、11 分别用状态 q_1、q_2、q_3、q_4 表示，代入 $\boldsymbol{Y}$-$\boldsymbol{Z}$ 矩阵可得状态表，由此可给出状态图，如图 5.24 所示。

	y_2y_1	x	
		0	1
00	q_1	q_2,1	q_4,0
01	q_2	q_3,1	q_1,0
10	q_3	q_4,1	q_2,1
11	q_4	q_1,1	q_3,1

(a) 状态表



(b) 状态图

图 5.24　状态表和状态图

该电路是一个 Mealy 型时序机。由状态表和状态图可以看出，当输入 $x=0$ 时，在时钟脉冲 CP 的作用下，电路的状态按加 1 顺序变化，即

$$00\rightarrow01\rightarrow10\rightarrow11\rightarrow00\rightarrow\cdots$$

当 $x=1$ 时，在时钟脉冲 CP 的作用下，电路的状态按减 1 顺序变化，即

$$11\rightarrow 10\rightarrow 01\rightarrow 00\rightarrow 11\rightarrow\cdots$$

因此,该电路既具有加 1 计数功能,又具有减 1 计数功能,是一个二进制可逆计数器。

有时还需要用时间图来形象地描述电路的逻辑功能。时间图反映了时序电路在某一给定初态下,对给定输入序列的响应。下面介绍由状态图作时间图的方法。

假定计数器的初态 y_2y_1 为 00(即 q_1),输入 x 的序列为 0000011111,计数器在时钟脉冲 CP 控制下工作。先利用状态图作出时序电路的状态响应序列,而后再作时间图。状态响应序列如下:

CP	1	2	3	4	5	6	7	8	9	10
x	0	0	0	0	0	1	1	1	1	1
$y(Y)$	q_1	q_2	q_3	q_4	q_1	q_2	q_1	q_4	q_3	q_2
Z	1	1	1	1	1	0	0	1	1	0

在 CP_1 到来前,时序电路处于现态 q_1,当 $x=0$ 时,由状态图可知,输出 $Z=1$,次态为 0(即 CP_1 到来后的状态)。在 CP_2 到来前,电路处于现态 q_2,当 $x=0$,产生输出 1,次态为 q_3。以此类推,可得到整个状态响应序列。然后,再根据状态响应序列作出时间图。由于状态 y 由 y_2y_1 来表示,所以只要将状态 q_i 按二进制代码表示后,就可画出按电平高低的 Y_2、Y_1 时间图。例如,q_2 的代码为 01,则在 Y_2、Y_1 的时间图中,Y_2 为低电平,Y_1 为高电平。图 5.25 表示该电路的时间图。

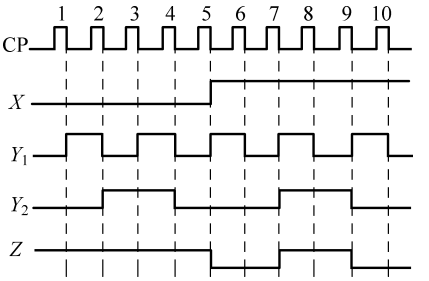


图 5.25　可逆计数器的时间图

【例 5.13】 分析图 5.26 所示同步时序电路。

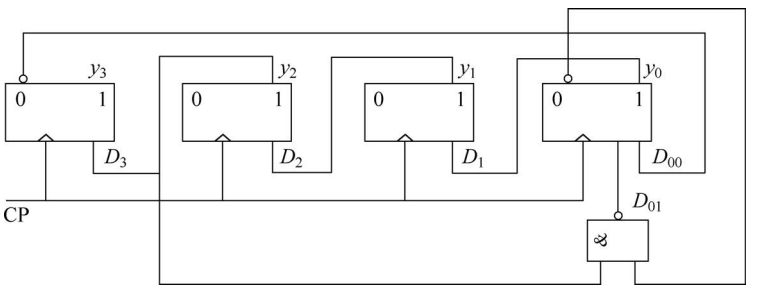


图 5.26　同步时序电路

列出激励函数,求激励矩阵。激励函数为

$$D_{00}=\bar{y}_3,D_{01}=\overline{y_2\cdot\bar{y}_0}=\bar{y}_2+y_0,D_0=D_{00}\cdot D_{00}$$

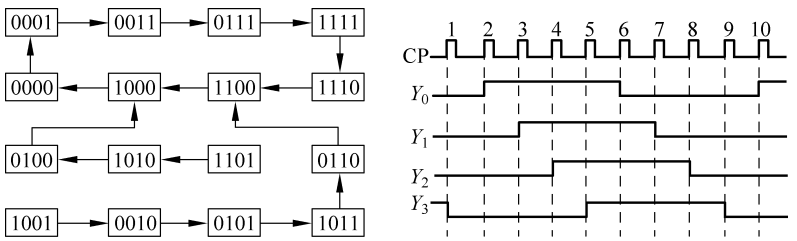
$$D_1=y_0,D_2=y_1,D_3=y_2$$

将 4 个状态变量 $y_0 \sim y_3$ 的 16 种组合代入上式，得激励矩阵如表 5.32 所示。注意，本例电路没有外部输入，也没有外部输出 Z。

表 5.32　激励矩阵和状态表

y_3	y_2	y_1	y_0	D_3	D_2	D_1	D_{01}	D_{00}	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	0	1	1	1	0	0	1	1
0	0	1	0	0	1	0	1	1	0	1	0	1
0	0	1	1	0	1	1	1	1	0	1	1	1
0	1	0	0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	0	1	1	1	1	0	1	1
0	1	1	0	1	1	0	0	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	1	0	0	1	1	0	0	0	1	0
1	0	1	0	0	1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	1	0	0	1	1	0
1	1	0	0	1	0	0	0	0	1	0	0	0
1	1	0	1	1	0	1	1	0	1	0	1	0
1	1	1	0	1	1	0	0	0	1	1	0	0
1	1	1	1	1	1	1	1	0	1	1	1	0

由激励矩阵和 D 触发器的激励表可得到 $\boldsymbol{y}$ 矩阵，即状态表，如表 5.32 右边一栏所示。根据二进制状态表可作出状态图，如图 5.27(a)所示。



(a) 状态图　　(b) 时间图

图 5.27　状态图和时间图

由状态图可以看出，这是一个循环移位计数器。在计数时循环移位规则如下：

$$y_0 \to y_1 \quad y_1 \to y_2 \quad y_2 \to y_3 \quad \overline{y}_3 \to y_0$$

这种计数器的循环长度 $l=2n$，其中 n 为位数，这里 $n=4$，$l=8$。

由状态图还可看出，图上半部 8 个状态形成闭环，称为“有效序列”；下半部 8 个状态称为“无效序列”。如果该时序电路在某种偶然因素作用下，使电路处于“无效序列”中的某一状态，则它可以在时钟脉冲 CP 的作用下，经过若干 CP 后，将会自动进入有效序列。因此，该计数器称为具有自恢复功能的扭环移位计数器。

这种计数器在数字系统的控制电路中用得较多。图 5.27(b)是它的时间图。根据 $Y_0 \sim Y_3$ 这 4 个基本波形，经过简单组合，可以形成各种不同的时序控制波形。

【例 5.14】 用 T 触发器设计一位数字的 8421BCD 码同步加 1 计数器。

设计步骤如下。

① 建立状态表。由于计数器的工作状态很有规律,所以可以直接建立二进制状态表。这里,由于计数状态 $n=10$,故需要 4 个触发器 $y_1 \sim y_4$,其状态表如表 5.33 所示。

表 5.33 激励矩阵和状态表

y_1	y_2	y_3	y_4	Y_1	Y_2	Y_3	Y_4	Z	T_1	T_2	T_3	T_4
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1
1	0	1	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
1	0	1	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
1	1	0	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
1	1	0	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
1	1	1	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*
1	1	1	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*	*d*

② 求激励函数和输出函数。根据状态表和 T 触发器的激励表,可得到网络的激励矩阵,如表 5.33 所示。分别画出激励函数 $T_1 \sim T_4$ 和输出函数 Z 的卡诺图如图 5.28 所示。由卡诺图化简,可得到网络的激励函数和输出函数为

$$T_1=y_1y_4+y_2y_3y_4,\ T_2=y_3y_4,\ T_3=\bar{y}_1y_4,\ T_4=1,\ Z=y_1y_4$$

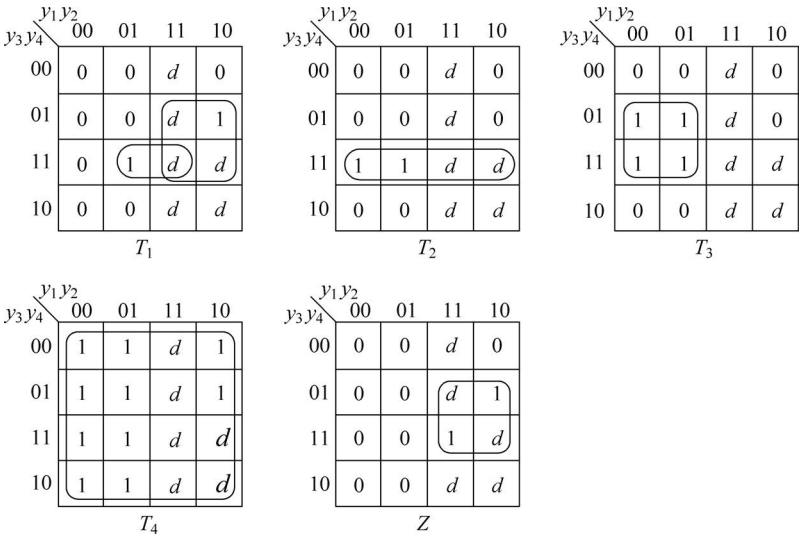


图 5.28 卡诺图

③ 画逻辑图。根据上述激励函数和输出函数，可画出所要求的 BCD 码十进制计数器的逻辑图如图 5.29 所示。

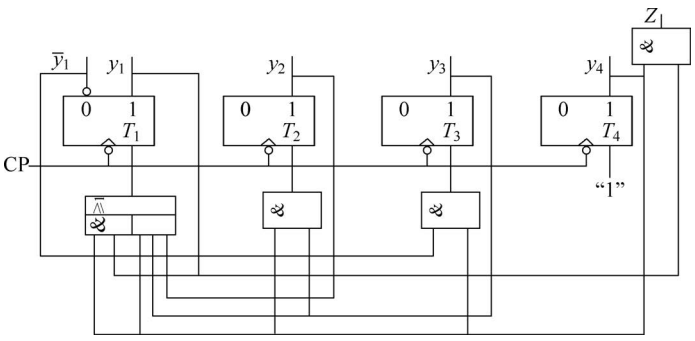


图 5.29　BCD 码十进制计数器逻辑图

对于未完全确定的 6 种状态(即 1010～1111)，可由上述激励函数表达式和 T 触发器的激励表，得到它们的状态转换图如图 5.30 所示。例如，状态“1010”为不完全确定状态，即 $y_1=1, y_2=0, y_3=1, y_4=0$，将其代入激励函数表达式得：$T_1=T_2=T_3=0, T_4=1$，则状态“1010”在该激励下将转换到状态“1011”；再由“1011”，求得 $T_1=T_2=T_4=1, T_3=0$，则状态“1011”将转换到状态“0110”，从而进入计数循环中。其余 4 个不确定状态可按同理分析进入计数循环。

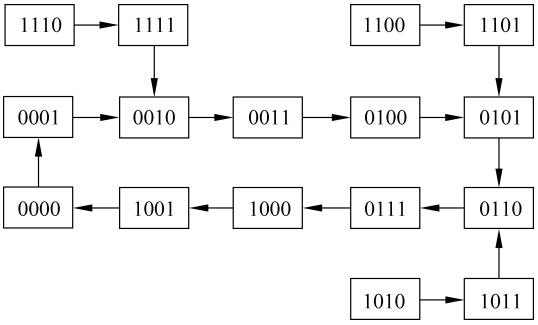


图 5.30　BCD 码十进制计数器的状态转换图

由图 5.30 所示状态转换图可见，图中没有孤立状态。因而，所设计的图 5.28 所示的电路是能够自行恢复的。也就是说，不论电路处于何种初始状态，它都可以自动地进入计数循环中，而不会发生“挂起”现象。

5.4　常用的同步时序电路

常用的同步时序电路有寄存器、计数器、节拍信号发生器等。其中，寄存器和计数器在计算机及其他数字系统中应用极为广泛，是数字系统的重要组成部分。

5.4.1 寄存器

寄存器是由触发器组成的用来寄存二进制数码的逻辑部件,它是计算机中最基本的逻辑部件。通常,寄存器应具有以下 4 种功能:

(1) 清除数码。将数码寄存器中所寄存的原始数据清除掉,在逻辑上只要将所有触发器的置"0"端连接在一起,作为置"0"信号的输入端。当需要清除数码时,可在置"0"输入端加一个置 0 脉冲,寄存器将全部处于"0"状态。

(2) 接收数码。在接收信号的作用下,将外部输入数据接收到寄存器中。

(3) 寄存数码。数码寄存器接收了数据代码后,只要不出现"清除""接收"等信号,寄存器应保留原寄存数据不变。

(4) 输出数码。在输出控制信号作用下,控制数码寄存器中的数据输出。

有些寄存器还具有移位逻辑功能,称为移位寄存器。实现移位功能,只要将寄存器的每一位触发器输出连到下一位触发器的数码输入端即可。在 CP 脉冲作用下,寄存器中的数码在移位控制信号控制下左移或右移。

在实际应用中,往往要求寄存器同时具有接收、右移、左移、保持等多种逻辑功能,这种寄存器的逻辑图如图 5.31 所示。该电路具有 4 个并行数据输入端允许并行载入外部数据;两个串行输入端,一个用于左移,一个用于右移;串行输出可利用 4 个并行输出中的一个实现。图中,每位触发器的输入激励函数 D_i 表达式为

$$D_i = K_{右} y_{i+1} + K_{左} y_{i-1} + K_{接} S_i + K_{保} y_i$$

式中,$K_{保} = \overline{K}_{右} \cdot \overline{K}_{左} \cdot \overline{K}_{接}$。保持的作用就是将各触发器的数码送到本触发器中,也就是构成自身反馈 $D_i = y_i$,达到保持原有数据不变的目的。

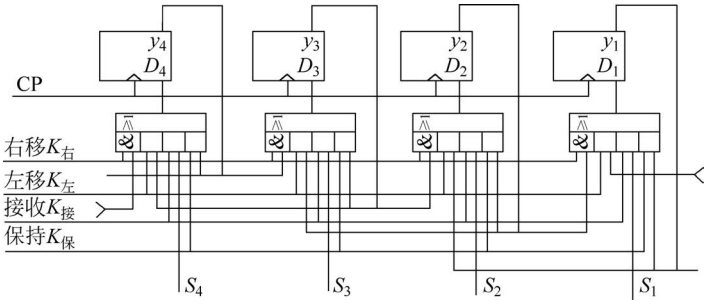


图 5.31 具有右移、左移、接收、保持功能的移位寄存器

利用移位寄存器不仅可以方便地实现串行数据到并行数据或并行数据到串行数据的转换,还可用作序列信号发生器。

5.4.2 计数器

计数器是用来记录脉冲数目的数字电路,它也是数字设备中的基本逻辑部件。如计算

机中用来记录指令执行顺序的指令计数器，用以记录乘除法步数的乘除计数器等。此外，计数器还常用来实现分频、定时等逻辑功能。

计数器的种类很多，按工作方式分，它可分为异步计数器和同步计数器；按进位制分，可分为二进制计数器和非二进制计数器；此外，按工作特点又可分为加 1 计数器、减 1 计数器、可逆计数器和环形移位计数器。以上各种计数器，有些在前面时序电路的分析和设计举例中已涉及。这里主要介绍一下同步计数器。

同步计数器的计数脉冲(即 CP)同时加到各触发器的 CP 端，当计数脉冲到来时，各触发器同时改变状态。因此，同步计数器又称为并行计数器。同步计数器的总的延迟时间与计数器的位数无关，因而速度可以大大提高。

同步计数器的设计方法，可以按一般同步时序网络的设计方法进行。但由于计数器的状态转换很有规律，故可以直接建立二进制状态表(不必进行状态化简和状态分配)。然后，根据所用触发器的激励表，把二进制状态表(或 ***Y-Z*** 矩阵)变换成激励矩阵。最后，求出激励函数和输出函数，画出逻辑图。这样的方法前面已做了较充分的论述，下面主要介绍另外的同步计数器设计方法。

【例 5.15】 用 T 触发器设计一个模 16 同步加 1 计数器。

在一个多位二进制数的末位上加 1 时，根据二进制加法运算规则可知，若从低位到高位数第 i 位是第一个“0”位，即以下各位皆为“1”时，则第 i 位及以下各位应改变状态(由 0 变成 1，由 1 变成 0)；第 i 位以上各位不变。而最低位的状态在每次加 1 时都要改变。例如

$$\begin{array}{r} 1\ 0\ 1\ 0\ 1\ 1\ 1 \\ +\qquad\qquad 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 0 \end{array}$$

用 T 触发器构成同步计数器，则每次 CP 信号(也就是计数脉冲)到达时应使该翻转的那些触发器输入控制端 $T=1$，不该翻转的 $T=0$。由此可知，当计数器用 T 触发器构成时，第 i 位触发器输入端的逻辑式应为

$$T_i = Q_{i-1} \cdot Q_{i-2} \cdot \cdots \cdot Q_1 \cdot Q_0 = \prod_{j=0}^{i-1} Q_j \quad (i=1,2,\cdots,n-1)$$

只有最低位例外，按照计数规则，每次输入计数脉冲时它都要翻转，故 $T_0=1$。因而有

$$\begin{cases} T_0 = 1 \\ T_1 = Q_0 \\ T_2 = Q_0 Q_1 \\ T_3 = Q_0 Q_1 Q_2 \end{cases}$$

图 5.32 所示电路就是按上式构成的 4 位二进制同步加法计数器。将上式代入 T 触发器的特性方程式得到电路的状态方程：

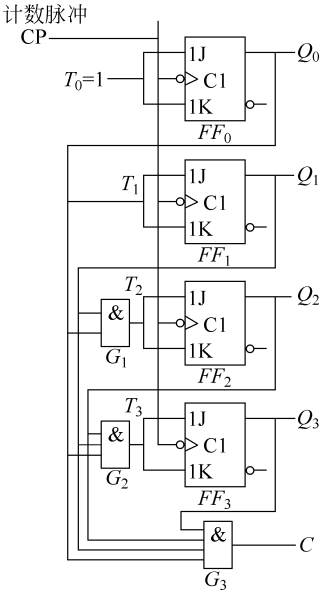


图 5.32　用 T 触发器构成的模 16 同步加 1 计数器

$$\begin{cases} Q_0^{n+1} = \overline{Q}_0 \\ Q_1^{n+1} = Q_0\overline{Q}_1 + \overline{Q}_0 Q_1 \\ Q_2^{n+1} = Q_0 Q_1 \overline{Q}_2 + \overline{Q_0 Q_1} Q_2 \\ Q_3^{n+1} = Q_0 Q_1 Q_2 \overline{Q}_3 + \overline{Q_0 Q_1 Q_2} Q_3 \end{cases}$$

电路的输出为:

$$C = Q_0 Q_1 Q_2 Q_3$$

根据上述状态方程和输出方程可求出电路的状态转换表如表 5.34 所示。利用第 16 个计数脉冲到达时 C 端电位的下降沿可作为向高位计数器电路进位的输出信号。

表 5.34 模 16 同步加 1 计数器的状态转换表

计数顺序	电路状态				等效十进制数	进位输出 C
	Q_3	Q_2	Q_1	Q_0		
0	0	0	0	0	0	0
1	0	0	0	1	1	0
2	0	0	1	0	2	0
3	0	0	1	1	3	0
4	0	1	0	0	4	0
5	0	1	0	1	5	0
6	0	1	1	0	6	0
7	0	1	1	1	7	0
8	1	0	0	0	8	0
9	1	0	0	1	9	0
10	1	0	1	0	10	0
11	1	0	1	1	11	0
12	1	1	0	0	12	0
13	1	1	0	1	13	0
14	1	1	1	0	14	0
15	1	1	1	1	15	1

图 5.33 给出了图 5.32 电路的时序图。由时序图上可以看出,若计数输入脉冲的频率为 f_0,则 Q_0、Q_1、Q_2 和 Q_3 端输出脉冲的频率将依次为 $\frac{1}{2}f_0$、$\frac{1}{4}f_0$、$\frac{1}{8}f_0$ 和 $\frac{1}{16}f_0$。针对计数器的这种分频功能,也把它称为分频器。

在同步计数器中,非模 2^i 计数器和模 2^i 计数器的设计方法是相同的,只是非模 2^i 计数器需要检查不确定状态的自恢复性能,参见 5.3 节例 5.14 的 BCD 码十进制计数器。此外,同步计数器也可以设计成不按二进制码顺序计数,而按 Gray 码顺序计数,即每次计数只有一位触发器改变状态。其设计方法与前面相同,这里不再多述了,读者根据前面方法不难进行设计。

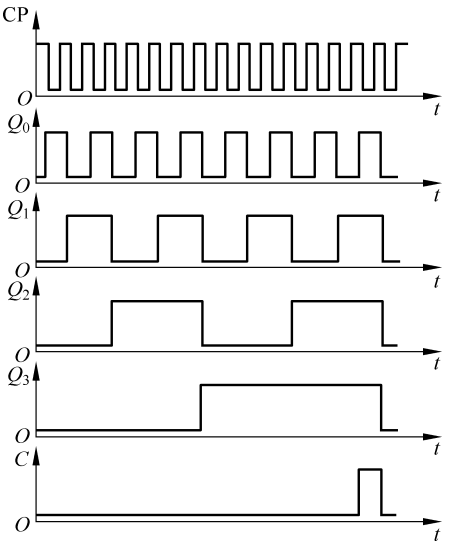


图 5.33　模 16 同步加 1 计数器的时序图

5.4.3　节拍信号发生器

计算机在执行一条指令时，总是把一条指令分成若干基本动作，由控制器发出一系列节拍电位，每个节拍电位控制计算机完成一个或几个基本动作。节拍信号发生器就是用来产生节拍电位的逻辑部件。按其结构来分，节拍信号发生器可分为计数型和移位型两种。

1. 计数型节拍信号发生器

计数型节拍信号发生器由计数器和译码器构成。图 5.34 是一个能产生 4 个节拍电位的节拍信号发生器。输出电位为 $W_0 \sim W_3$，输出脉冲是 $m_0 \sim m_3$，它的工作波形如图 5.35 所示。

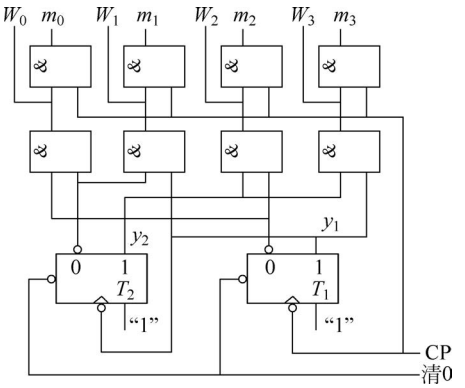


图 5.34　计数型节拍信号发生器

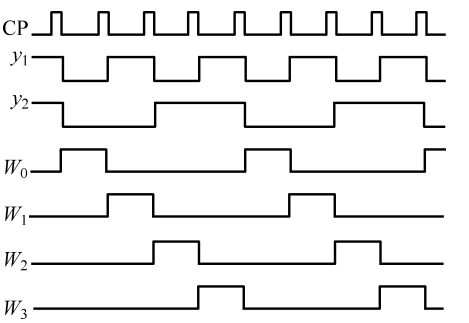


图 5.35　节拍信号发生器的波形图

2. 移位型节拍信号发生器

图 5.36 是一个能产生 4 个节拍电位的移位型节拍信号发生器。4 个 D 触发器接成环形移位寄存器,输出直接取自触发器的输出端。假设寄存器初态 $y_4y_3y_2y_1=1000$,在 CP 脉冲作用下,其状态转换图如图 5.37(a)所示。这种循环状态是我们所希望的,称为“有效时序”。在有效时序下的时间波形图如图 5.37(b)所示。

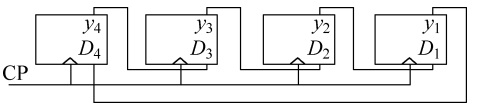


图 5.36 移位型节拍信号发生器

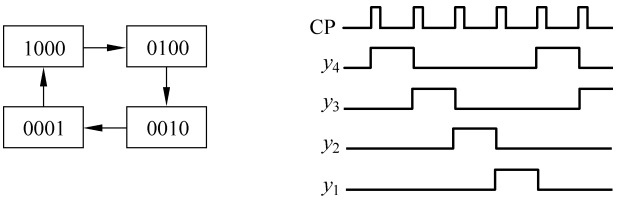


图 5.37 4 位环形移位寄存器的有效时序

但是,图 5.36 这种简单的环形移位寄存器并不能保证电路一定工作在有效序列。随着移位寄存器中的初始状态不同,将产生不同的状态转换。例如,初始状态为 1100、1110、1010 时,其状态图如图 5.38(a)~(c)所示,它们都可以构成循环。显然,这是我们所不希望的,一般称为“无效时序”。当寄存器的初始状态为 0000 或 1111 时,其状态图如图 5.38(d)所示,寄存器将始终在该状态“空转”,通常称为“空转时序”。一旦由于某种因素,电路进入无效时序或空转时序,则它就不能再回到有效序列了。

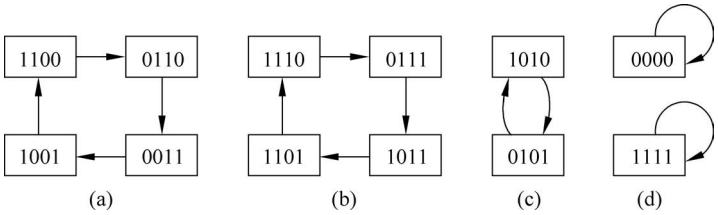


图 5.38 4 位环形移位寄存器的其他时序

怎样使环形移位寄存器只工作在一种有效时序下呢?一种办法是预置初值,即通过置 0 置 1 输入端,使寄存器预置成所需状态。但这种办法不能保证电路在受到某种干扰时,仍能工作在有效时序。另一种办法是加反馈逻辑,使电路具有自恢复能力,即当电路由于某种因素而进入无效序列时,能自动回到原有效序列。这是一种较好的办法。下面介绍采用反馈逻辑的设计方法。

反馈逻辑的加接方法有好多种,我们采用加在第一级激励输入端的方案,即断开图 5.36 中 y_1 与 D_4 的连接线,在 D_4 端加一个反馈逻辑电路,这个反馈逻辑电路能使移位寄存器从

无效时序或空转时序引导到有效时序中。其设计步骤如下。

(1) 列状态表。状态表中先列出有效时序,其余状态的转换只要使寄存器的 4 位状态进入有效时序即可。其状态表如表 5.35 左边部分所示。

表 5.35　状态表和激励矩阵

	y_4	y_3	y_2	y_1	Y_4	Y_3	Y_2	Y_1	D_4	D_3	D_2	D_1
有效时序	1	0	0	0	0	1	0	0	0	1	0	0
	0	1	0	0	0	0	1	0	0	0	1	0
	0	0	1	0	0	0	0	1	0	0	0	1
	0	0	0	1	1	0	0	0	1	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0
	0	0	1	1	0	0	0	1	0	0	0	1
	0	1	0	1	0	0	1	0	0	0	1	0
	0	1	1	0	0	0	1	1	0	0	1	1
	0	1	1	1	0	0	1	1	0	0	1	1
	1	0	0	1	0	1	0	0	0	1	0	0
	1	0	1	0	0	1	0	1	0	1	0	1
	1	0	1	1	0	1	0	1	0	1	0	1
	1	1	0	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	1	0	0	1	1	0
	1	1	1	0	0	1	1	1	0	1	1	1
	1	1	1	1	0	1	1	1	0	1	1	1

(2) 求激励函数。由状态表可得激励矩阵,如表 5.35 右边部分所示。由此可求得激励函数为

$$D_4=\bar{y}_4\bar{y}_3\bar{y}_1,D_3=y_4,D_2=y_3,D_1=y_2$$

(3) 画逻辑图。根据激励函数可画出具有“反馈逻辑”的 4 位移位寄存器逻辑图,如图 5.39 所示。

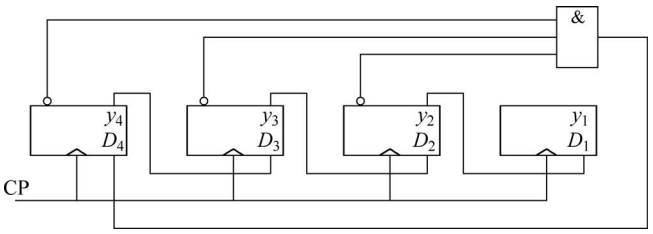


图 5.39　加“反馈逻辑”的 4 位环形移位寄存器

以上这种环形移位寄存器可以直接用作节拍电位信号发生器,只要从各触发器的输出端直接引出信号即可。但是,这种寄存器的状态利用率很低。一个 n 位环形移位寄存器可有 2^n 个状态,却只用了其中 n 个状态,用以产生 n 个节拍信号。如果采用扭环形移位寄存器,当位数仍为 n 时,则可产生 $2n$ 个节拍信号。

3. 序列信号发生器

在数字信号的传输和数字系统的测试中,有时需要用到一组特定的串行数字信号。通常把这种串行数字信号称为序列信号。产生序列信号的电路称为序列信号发生器。

序列信号发生器的构成方法有多种。一种比较简单、直观的方法是用计数器和数据选择器组成。例如,需要产生一个 8 位的序列信号 00010111(时间顺序为自左而右),则可用一个八进制计数器和一个 8 选 1 数据选择器组成,如图 5.40 所示。其中八进制计数器取自 74LS161(4 位二进制计数器)的低 3 位。74LS152 是 8 选 1 数据选择器。

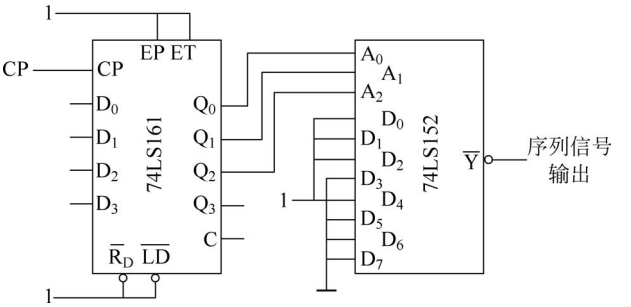


图 5.40 用计数器和数据选择器组成的序列信号发生器

在需要修改序列信号时,只要将该序列信号加到 $D_0 \sim D_7$ 即可实现,而不需对电路结构作任何更动。因此,使用这种电路既灵活又方便。

习题 5

5.1 已知时序网络的状态表如题表 5.1 所示,试作出它的状态图。

题表 5.1

Y	Y^{n+1}		Z
	$x=0$	$x=1$	
A	C	B	0
B	C	D	0
C	D	B	0
D	B	A	1

5.2 设有如题表 5.2 所示的 3 个完全定义机的状态表,试求每一状态表等价的最简状态表。

题表　5.2(a)

y	x	
	0	1
1	4,0	2,0
2	3,1	1,0
3	2,1	5,0
4	1,0	2,0
5	4,1	1,0

题表　5.2(b)

y	x	
	0	1
1	8,0	7,1
2	3,0	5,0
3	2,0	1,0
4	5,1	8,0
5	8,0	4,1
6	5,1	3,0
7	1,1	8,0
8	4,0	6,1

题表　5.2(c)

y	x_1x_2			
	00	01	11	10
1	2,0	3,0	2,1	1,0
2	5,0	3,0	2,1	4,1
3	1,0	2,0	3,1	4,1
4	3,0	4,0	1,1	2,0
5	3,0	3,0	3,1	5,0

5.3　试求题表 5.3 中不完全定义机的最大相容类。

题表　5.3

y	x_1x_2			
	00	01	11	10
1	d,d	5,1	1,d	3,1
2	d,d	2,d	5,1	6,1
3	6,0	d,d	d,d	2,1
4	d,1	d,d	3,0	d,d
5	1,d	5,0	6,d	d,d
6	4,0	6,d	d,d	d,d

5.4 试化简题表 5.4 所示的 3 个不完全定义机的状态表。

题表 5.4(a)

y	x_1x_2			
	00	01	11	10
1	3,0	3,d	4,d	3,d
2	4,1	3,0	d,d	1,d
3	1,d	1,1	d,d	d,d
4	2,d	d,d	3,d	5,d
5	2,d	5,d	3,d	4,d

题表 5.4(b)

y	x	
	0	1
1	d,0	4,d
2	1,1	5,1
3	d,0	4,d
4	3,1	2,1
5	5,d	1,d

题表 5.4(c)

y	x	
	0	1
1	3,d	d,d
2	d,d	6,0
3	4,1	5,d
4	6,1	d,d
5	5,d	1,d
6	4,1	7,1
7	2,0	3,0

5.5 试用 JK 触发器设计一个“101”序列检测器。该同步时序网络有一根输入线 x,一根输出线 Z。对应于每个连续输入序列“101”的最后一个 1,输出 $Z=1$,其他情况下 $Z=0$。例如:

$$x \quad 010101101$$
$$Z \quad 000101001$$

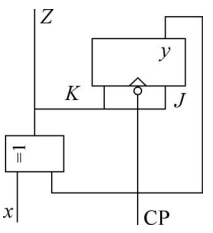
5.6 设有如题表 5.6 所示的 Moore 型时序机状态表,试用 T 触发器和与非门设计此时序电路。

题表　5.6

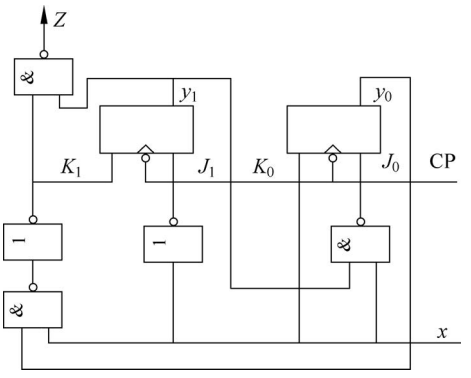
y	x		
	0	1	Z
q_1	q_2	q_3	0
q_2	q_1	q_4	1
q_3	q_2	q_2	0
q_4	q_1	q_4	1

5.7　试分析题图 5.7 所示同步时序网络，作出它的状态表和状态图，并画出当输入 x 为 0110101 序列时网络的时间图。

5.8　试分析题图 5.8 所示同步时序网络，作出它的状态表和状态图，画出当电平输入 x 为 0110110 序列时网络的时间图。



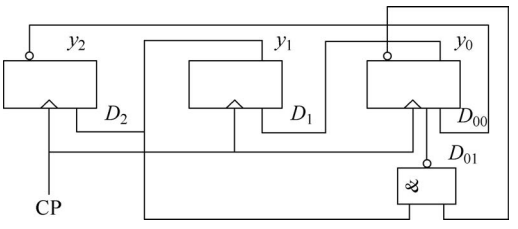
题图　5.7



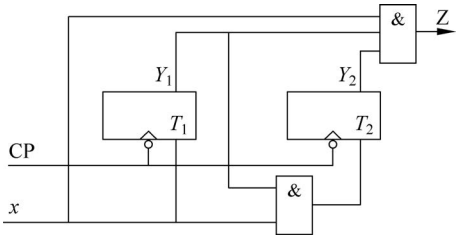
题图　5.8

5.9　试分析题图 5.9 所示同步时序网络，作出它的状态表和状态图。

5.10　试分析题图 5.10 所示同步时序网络的逻辑功能，作出它的状态表和状态图，并画出当输入 x 的序列为 1011101，初态 $y_2y_1=00$ 时网络的时间图。



题图　5.9



题图　5.10

5.11　试用 JK 触发器设计一个 Gray 码十进制计数器。

5.12　试用 T 触发器设计一个模 8 二进制可逆同步计数器。设用外部输入 x 控制加 1 或减 1，当 $x=0$ 时减 1；当 $x=1$ 时加 1。

5.13　设计一个具有下述特点的计数器。

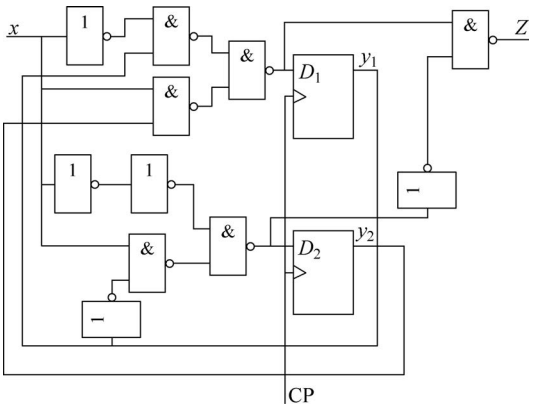
(1) 计数器有两个控制输入 C_1 和 C_2，C_1 用以控制计数器的模数，而 C_2 用以控制计数

器的加减。

(2)如 $C_1=0$,则计数器为模 3 计数;如 $C_1=1$,则计数器为模 4 计数。

(3)如 $C_2=0$,则计数器为加 1 计数;如 $C_2=1$,则计数器为减 1 计数。

5.14 设有如题图 5.14 所示时序网络,试作出其状态表。



题图 5.14

5.15 设某一时序网络有一个输入 x 和一个输出 Z。输入由 5 个符号组成,这些符号或为 0 或为 1。当 5 个符号中恰有 3 个符号为 1,两个符号为 0,且前两个符号为 11,则输出 $Z=1$。试建立此网络的状态图和状态表。

5.16 采用 D 触发器,实现题表 5.16 所示状态表的逻辑电路。

题表 5.16

y	x		
	x_1	x_2	x_3
A	A,0	B,0	C,1
B	B,0	C,0	D,0
C	C,0	D,0	A,1
D	D,0	A,0	B,1