第5章 时序逻辑的分析与设计

逻辑电路有两大类：一类是组合逻辑电路，另一类是时序逻辑电路。

组合逻辑电路：电路的输出只与当时的输入有关，而与电路以前的状态无关。

时序逻辑电路：电路的输出不仅与当前输入有关，还与以前的输入有关。

时序逻辑电路按其工作方式不同，又分为同步时序逻辑电路和异步时序逻辑电路。本章从时序逻辑电路的基本概念入手，重点讨论同步时序逻辑电路的分析和设计方法，同时对异步时序的概念及分析和设计方法也作简单的介绍。


视频讲解

5.1 时序逻辑电路的结构与类型

组合逻辑电路是由门电路构成的，其结构如图5-1所示。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。组合逻辑电路的输出可用下列输出函数集来描述：

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

输出 Z_i 仅是输入 x_i 的函数，即只与当前的输入有关。

时序逻辑电路的结构如图5-2所示，它由组合逻辑和存储器件两部分构成。图中 x_1，$x_2,\cdots,x_n$ 为时序逻辑电路的外部输入；$Z_1,Z_2,\cdots,Z_m$ 为时序逻辑电路的外部输出；y_1，$y_2,\cdots,y_r$ 为时序逻辑电路的内部输入(或称状态)；$Y_1,Y_2,\cdots,Y_p$ 为时序逻辑电路的内部输出(或称激励)。

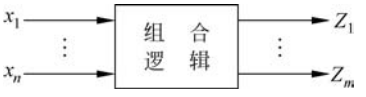


图5-1 组合逻辑电路的结构

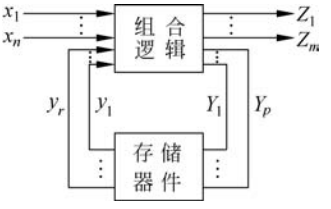


图5-2 时序逻辑电路的结构

时序逻辑电路的组合逻辑部分用来产生电路的输出和激励，存储器件部分是用其不同的状态($y_1,y_2,\cdots,y_r$)来“记忆”电路过去的输入情况的。

如图5-2所示的时序逻辑电路逻辑功能函数的一般表达式为

$$Z_i=g_i(x_1,x_2,\cdots,x_n;\ y_1,y_2,\cdots,y_r),\quad i=1,2,\cdots,m \tag{5-1}$$

$$Y_j=f_j(x_1,x_2,\cdots,x_n;\ y_1,y_2,\cdots,y_r),\quad j=1,2,\cdots,p \tag{5-2}$$

式(5-1)称为输出函数，式(5-2)称为激励函数。这两个函数都与变量 x，y 有关，也即电路的

输出不仅与电路的输入有关,而且与电路的状态有关。

时序逻辑电路按其工作方式可分为同步时序逻辑电路和异步时序逻辑电路。同步时序逻辑电路的存储器件由时钟控制触发器组成,并且有统一的时钟信号,只有当时钟信号到来时,电路状态($y_1,y_2,\cdots,y_r$)才发生变化。其余时间,即使输入发生变化,电路的状态也不会改变。时钟信号来之前的状态称为现态,记为 y_i^n(右上标也可省略);时钟信号到来之后的电路状态称为次态,记为 y_i^{n+1}。异步时序逻辑电路的存储器件可为触发器或延迟元件,电路中没有统一的时钟信号。

由于时序逻辑电路与组合逻辑电路在结构和性能上不同,因此在研究方法上两者也有所不同。组合逻辑电路的分析和设计所用到的主要工具是真值表,而时序逻辑电路的分析和设计所用到的工具主要是状态表和状态图。

同步时序逻辑电路在形式上又分成 Mealy 型和 Moore 型,它们在用状态表、状态图描述时其格式略有不同。

5.1.1 Mealy 型电路

如果同步时序逻辑电路的输出是输入和现态的函数,即 $Z_i=f_i(x_1,x_2,\cdots,x_n;\ y_1,y_2,\cdots,y_p)$,$i=1,2,\cdots,m$,则称该电路为 Mealy 型电路。也就是说输出与输入有直接的关系,输入的变化会影响输出的变化。

Mealy 型同步时序逻辑电路状态表的格式如表 5-1 所示。表格的上方从左到右列出输入 $x_1,x_2,\cdots,x_n$ 的全部组合,表格左边从上到下列出电路的全部状态 y,表格的中间列出对应不同输入组合的现态下的次态 y^{n+1} 和输出 Z。这个表的读法是,处于状态 y 的时序电路,当输入 x 时,输出为 Z,在时钟脉冲的作用下,电路进入次态 y^{n+1}。

例如,某同步时序逻辑电路有一个输入 x,一个输出 Z,4 个状态 A、B、C、D,该时序逻辑电路的状态表如表 5-2 所示。

表 5-1 Mealy 型电路状态表格式

现　态	输　入		
	…	x	…
⋮			
y		y^{n+1}/Z	
⋮			

表 5-2 某 Mealy 型电路状态表

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

从该状态表可看出,若电路的初态为 A,当输入 $x=1$ 时,输出 $Z=1$,在时钟脉冲的作用下,电路进入次态 C。假定电路的输入序列为

$$x:\ 10100110$$

那么,与每个输入信号对应的输出响应和状态转移情况为

时钟:	1	2	3	4	5	6	7	8
x:	1	0	1	0	0	1	1	0
y:	A	C	B	A	D	A	C	D
y^{n+1}:	C	B	A	D	A	C	D	A
Z:	1	1	0	0	0	1	0	0

需要指出的是：

(1) 如果同步时序逻辑电路的初始状态不同，那么尽管输入序列相同，但输出响应序列和状态转移序列也将不同。

(2) 电路的现态和次态是相对某一时刻而言的，该时刻的次态就是下一个时刻的现态。

Mealy 型电路的状态图格式如图 5-3 所示，在状态图中，每个状态用一个圆圈表示，圈内用字母或数字表示状态的名称，用带箭头的直线或弧线表示状态的转移关系，并把引起这一转移的输入条件和相应的输出标注在有向线段旁边。例如上面某电路的状态表可描述为如图 5-4 所示的状态图。

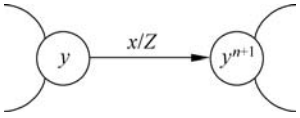


图 5-3　Mealy 型电路状态图

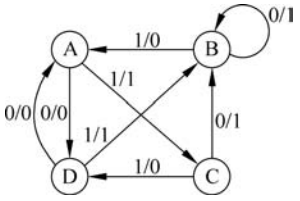


图 5-4　某电路的状态图

5.1.2　Moore 型电路

如果同步时序逻辑电路的输出仅是现态的函数，即 $Z=f_i(y_1,y_2,\cdots,y_p)$，$i=1,2,\cdots,m$，则称该电路为 Moore 型电路。也就是说该时序逻辑电路可能没有输入，或输入与输出没有直接关系。Moore 型电路的状态表格式如表 5-3 所示。因为 Moore 型电路的输出 Z 仅与电路的状态 y 有关，所以将输出单独作为一列，其值完全由现态确定。次态与 Mealy 型一样，由现态和输入共同确定。该表的读法是，当电路处于状态 y 时，输出为 Z。若输入 x，在时钟脉冲的作用下，电路进入次态 y^{n+1}。例如某 Moore 型时序逻辑电路的状态表如表 5-4 所示，当电路处于 A 状态时，其输出为 0。若 $x=1$，在时钟脉冲作用下，电路进入状态 B，新的输出为 1。假定电路的初始状态为 B，输入序列为

x：1　1　0　0　1　0　0　1

那么电路的状态转换序列和输出响应序列为

时钟：	1	2	3	4	5	6	7	8
x：	1	1	0	0	1	0	0	1
y：	B	C	A	C	B	C	B	B
y^{n+1}：	C	A	C	B	C	B	B	C
Z：	1	0	0	0	1	0	1	1

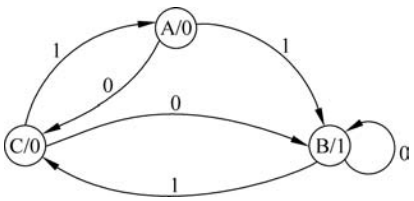


图 5-5　某 Moore 型电路的状态图

Moore 型电路的状态图与 Mealy 型电路状态图的区别仅在于 Moore 型电路的输出与状态一起标注在圆圈内，如上例的 Moore 型电路的状态图如图 5-5 所示。

表 5-3　Moore 型电路状态表格式

现　态	输　入			输出
	…	x	…	
⋮				
y		y^{n+1}		Z
⋮				

表 5-4　某 Moore 型电路状态表

y	x		Z
	0	1	
A	C	B	0
B	B	C	1
C	B	A	0

5.2　同步时序逻辑电路的分析

时序逻辑电路的分析就是对一个给定的时序逻辑电路，研究在一系列输入信号作用下，电路将会产生怎样的输出，进而说明该电路的逻辑功能。

在输入序列的作用下，时序电路的状态和输出变化规律通常表现在状态表、状态图或时间图中。因此，分析一个给定的同步时序电路，实际上是要求出该电路的状态表、状态图或时间图，以此确定该电路的逻辑功能。

本节将介绍分析同步时序电路的两种方法，并通过示例分析，了解和熟悉几种常用的数字逻辑电路。

视频讲解

5.2.1　同步时序逻辑电路的分析方法

同步时序电路的分析方法有两种：表格法和代数法。两种方法的分析过程示意图如图 5-6 所示。

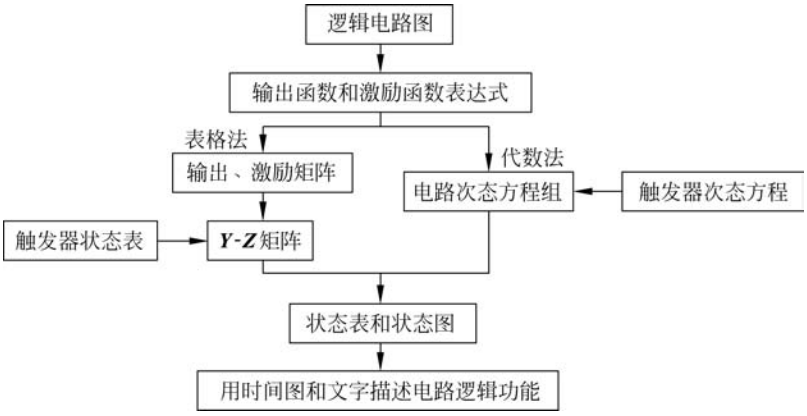


图 5-6　同步时序逻辑电路的分析过程示意图

下面介绍两种分析方法的一般步骤。

1. 表格法的一般步骤

(1) 根据给定的同步时序逻辑电路，写出输出函数表达式和激励函数表达式。

(2) 列出激励矩阵，即将激励函数以卡诺图的形式表示出来，若干激励合成激励矩阵。

(3) 根据所用触发器的状态表及激励矩阵、输出矩阵(输出函数的卡诺图形式)形成 ***Y-Z*** 矩阵。***Y-Z*** 矩阵实际就是二进制形式的状态表。

(4) 由 ***Y*-*Z*** 矩阵可得时序电路的状态表、状态图。

(5) 假定某一输入序列画出时间图，并用文字描述电路的逻辑功能。

2. 代数法的一般步骤

(1) 同表格法的(1)。

(2) 把激励函数表达式代入该电路触发器的次态方程，导出电路的次态方程组。

(3) 根据电路的次态方程组和输出函数表达式做出同步时序电路的状态表，画出状态图。

(4) 同表格法(5)。

两种方法的本质是相同的，视具体情况灵活选用。下面举例说明。


视频讲解

例 5-1 分析如图 5-7 所示的同步时序电路的逻辑功能。假定在初态为 00 时，输入 x 的序列为 0000011111，画出时间图。

解：由电路图可写出激励函数、输出函数。

$$K_0 = J_0 = 1$$

$$K_1 = J_1 = x \oplus y_0 = x\bar{y}_0 + \bar{x}y_0$$

$$Z = \overline{x \cdot \bar{y}_1} = \bar{x} + y_1$$

方法一：表格法。

将激励函数、输出函数表示在卡诺图上，如图 5-8 所示。

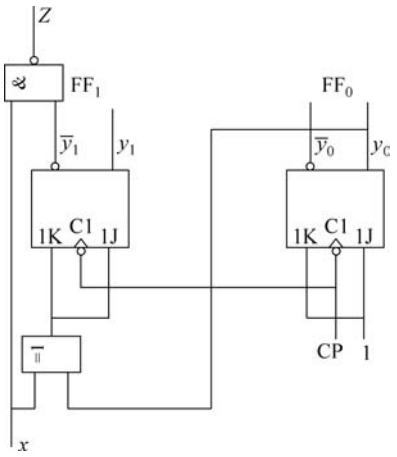


图 5-7 例 5-1 图

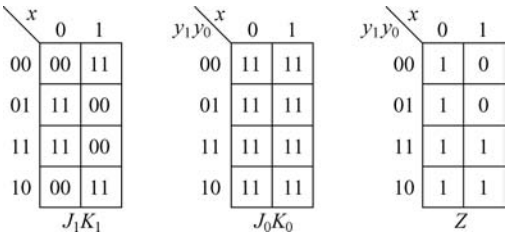


图 5-8 J、K 和 Z 的卡诺图

将 J、K 的卡诺图合并画到一个卡诺图上便可得电路的激励矩阵，如表 5-5 所示。再根据 JK 触发器的状态表和输出矩阵，可将激励矩阵转换成 ***Y*-*Z*** 矩阵，如表 5-6 所示。

表 5-5 激励矩阵 J_1K_1 和 J_0K_0

y_1y_0	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

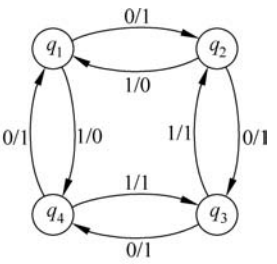
表 5-6 *Y*-*Z* 矩阵

y_1y_0	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

Y*-*Z 矩阵实际上就是二进制状态表,将编码 00、01、10、11 分别用状态 q_1、q_2、q_3、q_4 表示,代入 ***Y*-*Z*** 矩阵可得状态表,由状态表可画出状态图,如图 5-9 所示。

Q	x	
	0	1
q_1	$q_2/1$	$q_4/0$
q_2	$q_3/1$	$q_1/0$
q_3	$q_4/1$	$q_2/1$
q_4	$q_1/1$	$q_3/1$

(a) 状态表



(b) 状态图

图 5-9　状态表和状态图

该电路是一个 Mealy 型时序电路。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

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

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,且 4 个状态为一个循环,是一个模 4 的二进制可逆计数器。

假定计数器的初态 y_1y_0 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。下面先利用状态图做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 q_2(CP_1 到来后的状态)。在 CP_2 到来前,电路处于现态 q_2,当 $x=0$ 时,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做出时间图。由于状态 y 由 y_1y_0 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表示的 Y_1、Y_0 时间图。例如,q_2 的代码为 01,则在 Y_1、Y_0 的时间图中,Y_1 为低电平,Y_0 为高电平。图 5-10 表示该电路的时间图。

方法二:代数法。

以上过程用代数法也能很简单地求出结果。因为 JK 触发器的次态方程为

$$Q^{n+1}=J\overline{Q}+\overline{K}Q$$

对于本例的逻辑图,两个触发器的次态方程为

$$y_1^{n+1}=J_1\overline{y}_1+\overline{K}_1y_1$$

$$y_0^{n+1}=J_0\overline{y}_0+\overline{K}_0y_0$$

将已求得的电路的激励函数代入该次态方程组就可得该电路的次态方程组。

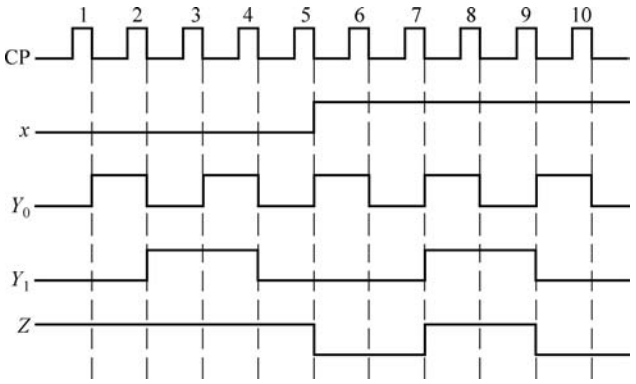


图 5-10　可逆计数器的时间图

$$y_1^{n+1}=J_1\bar{y}_1+\overline{K}_1y_1=(x\oplus y_0)\bar{y}_1+\overline{(x\oplus y_0)}y_1$$
$$=\bar{x}\bar{y}_1y_0+\bar{x}y_1\bar{y}_0+x\bar{y}_1\bar{y}_0+xy_1y_0$$
$$y_0^{n+1}=J_0\bar{y}_0+\overline{K}_0y_0=\bar{y}_0$$

将电路的次态方程组和输出函数表示到卡诺图上，如图 5-11 所示；将两个次态卡诺图与输出函数的卡诺图合并就形成了二进制形式的状态表，如表 5-7 所示。可以看出该二进制形式的状态表与上面表格法求得的 ***Y*-*Z*** 矩阵是一样的。

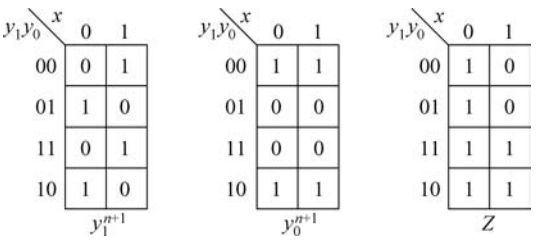


图 5-11　y_1^{n+1}、y_0^{n+1} 和 Z 的卡诺图

表 5-7　二进制形式的状态表

y_1y_0	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

例 5-2　分析如图 5-12 所示的同步时序电路。

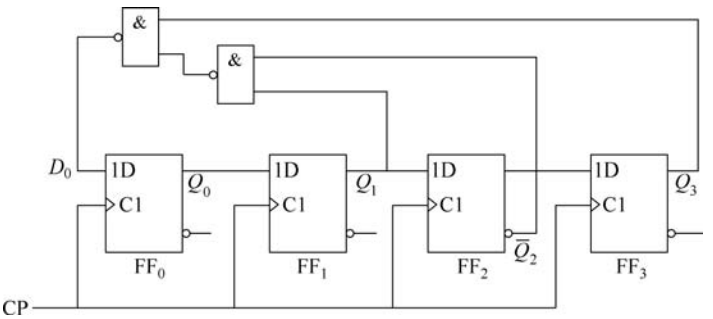


图 5-12　例 5-2 图

解：注意，本例比较特殊，没有外部输入，也没有外部输出。

首先写出它的激励函数。

$$D_0=\overline{Q_3\overline{Q}_2Q_1}=\overline{Q}_3+\overline{Q}_2Q_1$$

$$D_1=Q_0$$

$$D_2=Q_1$$

$$D_3=Q_2$$

将以上4个激励函数一起画到一幅卡诺图上就得到激励矩阵，如表5-8所示。因为D触发器的次态方程为$Q^{n+1}=D$，即次态与激励相等，所以求出的激励矩阵也就是$\boldsymbol{Y}$矩阵或二进制形式的状态表。很容易可得该电路的状态图如图5-13所示。

表5-8　激励矩阵

Q_1Q_0	Q_3Q_2			
	00	01	11	10
0　0	0 0 0 1	1 0 0 1	1 0 0 0	0 0 0 0
0　1	0 0 1 1	1 0 1 1	1 0 1 0	0 0 1 0
1　1	0 1 1 1	1 1 1 1	1 1 1 0	0 1 1 1
1　0	0 1 0 1	1 1 0 1	1 1 0 0	0 1 0 1

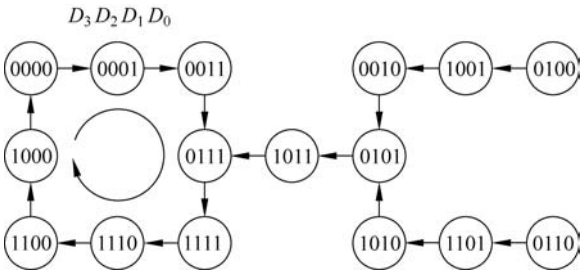


图5-13　状态图

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：

$$Q_0\rightarrow Q_1\quad Q_1\rightarrow Q_2\quad Q_2\rightarrow Q_3\quad \overline{Q}_3\rightarrow Q_0$$

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，图左半部8个状态形成闭环，称为“有效序列”，右半部8个状态称为“无效序列”。如果该时序电路在某种偶然因素的作用下，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，经过若干节拍后，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。

该电路的时间图如图5-14所示。根据$Q_0\sim Q_3$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。在计算机中，常常用它作为节拍信号发生器。

视频讲解

例5-3　分析图5-15的串行加法器电路，该电路有两个输入端x_1和x_2，用来输入加数和被加数。有一个输出端Z，用来输出相加的“和”。JK触发器用来存储“进位”，其状态y^n为低位向本位的进位，y^{n+1}为本位向高位的进位。

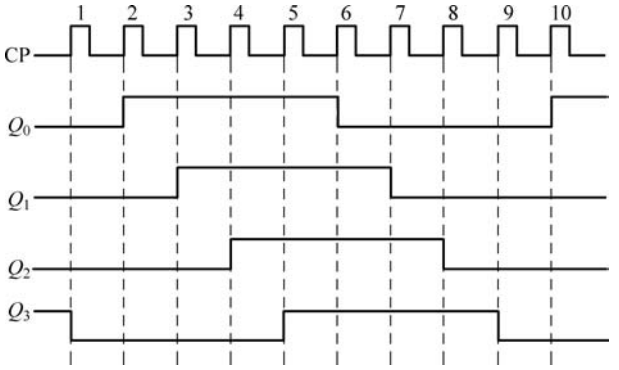


图 5-14　时间图

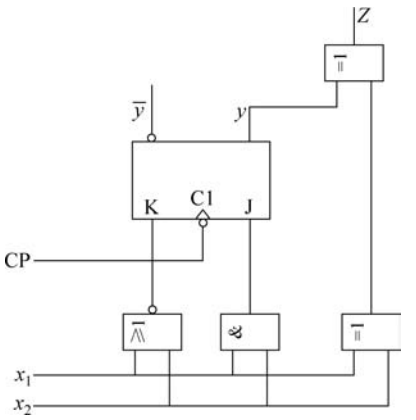


图 5-15　例 5-3 的电路图

解：首先写出电路的激励函数和输出函数表达式。

$$Z = x_1 \oplus x_2 \oplus y$$

$$J = x_1 x_2$$

$$K = \overline{x_1 + x_2}$$

JK 触发器的次态方程为

$$y^{n+1} = J\overline{y} + \overline{K}y$$

将激励函数表达式代入，得电路的次态方程为

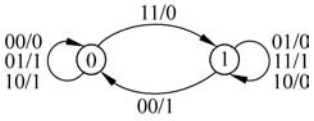
$$y^{n+1} = x_1 x_2 \overline{y} + \overline{\overline{(x_1 + x_2)}}y = x_1 x_2 \overline{y} + x_1 y + x_2 y$$

$$= x_1 x_2 + x_1 y + x_2 y$$

根据电路的次态方程可做出它的状态表和状态图，如图 5-16 所示。

y	x_1x_2			
	00	01	11	10
0	0/0	0/1	1/0	0/1
1	0/1	1/0	1/1	1/0

(a) 状态表



(b) 状态图

图 5-16　例 5-3 的状态表和状态图

设电路初始状态为 0。加数 x_1＝1 0 1 1，被加数 x_2＝0 0 1 1，加数、被加数均按照先低位后高位的顺序串行地加到相应的输入端。输出 Z 也是从低位到高位串行地输出的。

根据状态图做出的响应序列为

CP：　1　2　3　4

x_1x_2：11　11　00　10

y^n：　0　1　1　0

y^{n+1}：1　1　0　0

Z：　0　1　1　1

从以上状态响应序列可以看出,每位相加产生的进位由触发器保存了下来,以便参加下一位的相加。从输出响应序列可以看出,x_1 和 x_2 相加的"和"由 Z 端输出。

可以看出,两数相加的和为 $Z=1\ 1\ 1\ 0$。

由于该电路的输入和输出均是在时钟脉冲作用下,按位串行输入加数和被加数、串行输出"和"数的,故称此加法器为串行加法器。

如果需要保存相加的"和"数,可在输出端连接一个"串行输入/并行输出"的移位寄存器。加数和被加数也可事先放入"并行输入/串行输出"的移位寄存器中。

从这个例子可以看到,用组合逻辑电路实现的功能有的也可用时序电路来实现,不同的是,组合电路采用的是并行工作方式,而时序电路采用的是串行工作方式。因此,在完成同样的逻辑功能的情况下,组合电路比时序电路工作速度快,但时序电路的结构较组合电路简单。

视频讲解

例 5-4 分析如图 5-17 所示的节拍信号发生器电路。

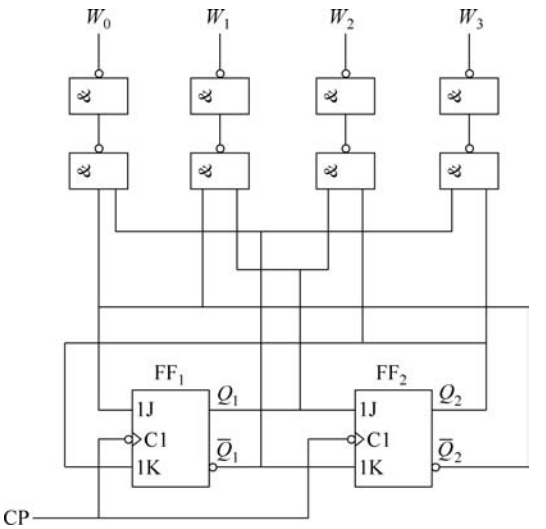


图 5-17 例 5-4 的图

解:首先写出激励函数和输出函数。

$$J_1=\overline{Q}_2,\quad K_1=Q_2,\quad J_2=Q_1,\quad K_2=\overline{Q}_1$$

$$W_0=\overline{Q}_1\overline{Q}_2,\quad W_1=Q_1\overline{Q}_2,\quad W_2=Q_1Q_2,\quad W_3=\overline{Q}_1Q_2$$

JK 触发器的次态方程为

$$Q_1^{n+1}=J_1\overline{Q}_1+\overline{K}_1Q_1$$

$$Q_2^{n+1}=J_2\overline{Q}_2+\overline{K}_2Q_2$$

将激励函数表达式代入,得电路的次态方程组为

$$Q_1^{n+1}=\overline{Q}_1\overline{Q}_2+Q_1\overline{Q}_2$$

$$Q_2^{n+1}=Q_1\overline{Q}_2+Q_1Q_2$$

根据电路的次态方程组就可得电路的状态表如表 5-9 所示。

表 5-9　例 5-4 的状态表

现　态		次　态		输　出			
Q_2	Q_1	Q_2^{n+1}	Q_1^{n+1}	W_0	W_1	W_2	W_3
0	0	0	1	1	0	0	0
0	1	1	1	0	1	0	0
1	1	1	0	0	0	1	0
1	0	0	0	0	0	0	1

这是一个 Moore 型电路，输出仅与现态有关。根据状态表可做出时间图如图 5-18 所示。由时间图可以看出，触发器 FF_2，FF_1 构成模 4 计数器，8 个与非门用来组合产生 4 个节拍电平信号，电路在时钟脉冲的作用下，按一定的顺序轮流地输出节拍信号。

节拍信号发生器通常用在计算机的控制器中。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平和节拍脉冲信号，以控制计算机完成一条指令的执行。

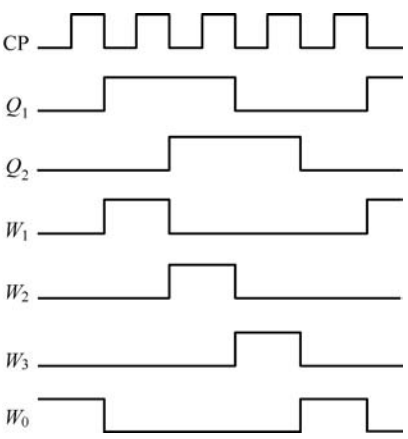


图 5-18　时间图


视频讲解

5.2.2　常用同步时序逻辑电路

1. 寄存器

寄存器用于寄存一组二值代码，它被广泛地用于各类数字系统和数字计算机中。

因为 1 个触发器能存储 1 位二进制代码，所以用 N 个触发器组成的寄存器能存储 N 位二进制代码。

对寄存器中的触发器只要求它们具有置 1、置 0 的功能即可，因而无论是用同步 RS 结构触发器，还是用主从结构或边沿触发结构的触发器，都可以组成寄存器。

图 5-19 是一个用同步 RS 触发器组成的 4 位寄存器的实例——74LS75 的逻辑图。由同步 RS 触发器的动作特点可知，在 CP 的高电平期间，Q 端的状态跟随 D 端的状态而变，在 CP 变成低电平以后，Q 端将保持 CP 变为低电平时 D 端的状态。

74LS175 则是用维持阻塞触发器组成的 4 位寄存器，它的逻辑图如图 5-20 所示。根据维持阻塞结构触发器的动作特点可知，触发器输出端的状态仅仅取决于 CP 上升沿到达时刻 D 端的状态。可见，虽然 74LS75 和 74LS175 都是 4 位寄存器，但由于采用了不同结构类型的触发器，其动作特点是不同的。为了增加使用的灵活性，有些寄存器电路还加了一些附加控制电路，如异步置 0(将寄存器的数据直接清除，而不受时钟信号的控制)、输出三态控制和“保持”等功能。所谓“保持”就是将触发器的输出反馈到输入，当 CP 信号到达时下一个状态仍保持原来的状态。例如 CMOS 电路 CC4076 就属于这样一种寄存器，它的逻辑图如图 5-21 所示。

这是一个具有三态输出的 4 位寄存器。

当 $LD_A+LD_B=1$ 时，电路处于装入数据的工作状态，输入数据 D_0、D_1、D_2、D_3 经与或门 G_5、G_6、G_7、G_8 分别加到 4 个触发器的输入端。在 CP 信号的下降沿到达后，将输入数据存入对应的触发器中。

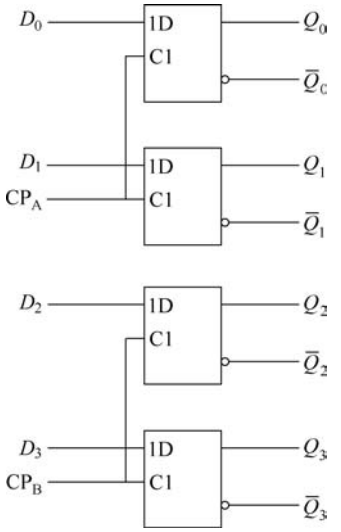


图 5-19　74LS75 的逻辑图

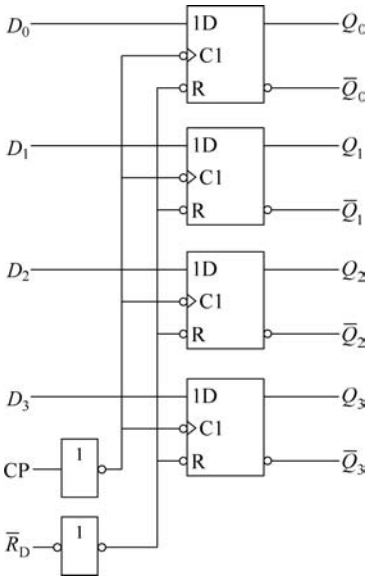


图 5-20　74LS175 的逻辑图

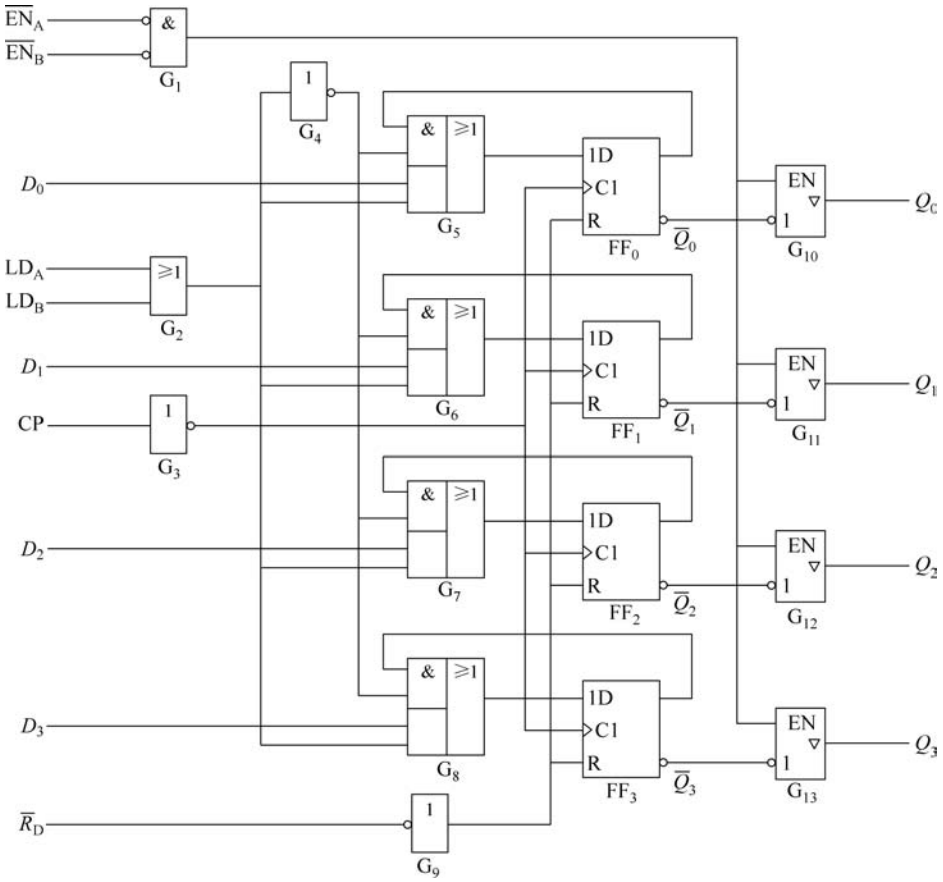


图 5-21　CC4076 逻辑图

当 $LD_A+LD_B=0$ 时，电路处于保持状态。触发器的 Q 端经与或门接回自己的输入端，故 CP 到达时还是原来的状态值。

当 $\overline{EN}_A=\overline{EN}_B=0$ 时，门 G_1 输出高电平，使三态门 $G_{10}\sim G_{13}$ 处于工作状态，电路正常输出。当 $\overline{EN}_A$、$\overline{EN}_B$ 中任一个为高电平时，则 G_1 输出为低电平，使 $G_{10}\sim G_{13}$ 处于高阻态，将触发器与输出端的联系切断。

当 $\overline{R}_D=0$ 时，将寄存器中的数据清除。

上面介绍的三个寄存器电路中，接收数据时所有位的代码是同时输入的，而且触发器中的数据是并行地出现在输出端的，因此将这种输入、输出方式叫并行输入、并行输出方式。

2. 移位寄存器

移位寄存器除了具有存储代码的功能以外，还具有移位功能。所谓移位功能，是指寄存器里存储的代码能在移位脉冲的作用下依次左移或右移。因此，移位寄存器不但可以用来寄存代码，还可以用来实现数据的串行-并行转换、数值的运算以及数据处理等。

例如，由边沿触发结构的 D 触发器组成的 4 位移位寄存器（见图 5-22），其中第一个触发器（左边）的输入端接收输入信号，其余的每个触发器输入端均与前面一个触发器的 Q 端相连。当 CP 的上升沿同时作用于所有触发器时，加到寄存器输入端 D_i 的代码存入 FF_0，其余触发器的状态为原左边一位触发器的状态，即总的效果是将寄存器里原有的代码右移了一位。

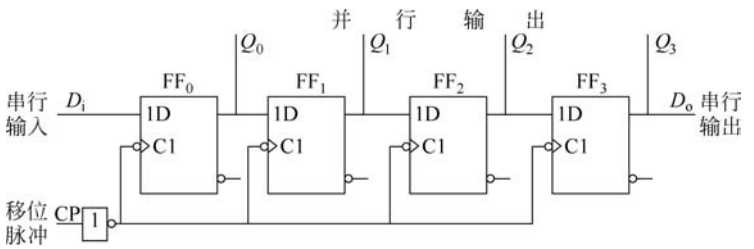


图 5-22　用 D 触发器构成的移位寄存器

例如，在 4 个时钟周期内输入代码依次为 $Q_0Q_1Q_2Q_3=0000$，那么在移位脉冲（也就是触发器的时钟脉冲）的作用下，移位寄存器里代码的移动情况将如表 5-10 所示。可以看到，经过 4 个 CP 信号以后，串行输入的 4 位代码全部移入了移位寄存器中，同时在 4 个触发器的输出端得到了并行输出的代码。因此，利用移位寄存器可以实现代码的串行-并行转换。为便于扩展逻辑功能和增加使用的灵活性，在定型生产的移位寄存器集成电路上有的又附加了左、右移控制，数据并行输入、保持、异步置零等功能。如 74LS194A 就是一个 4 位双向移位寄存器，它的逻辑图如图 5-23 所示。

表 5-10　移位寄存器中代码的移动情况

CP 的顺序	输入 D_i	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	1	1	1	0	1

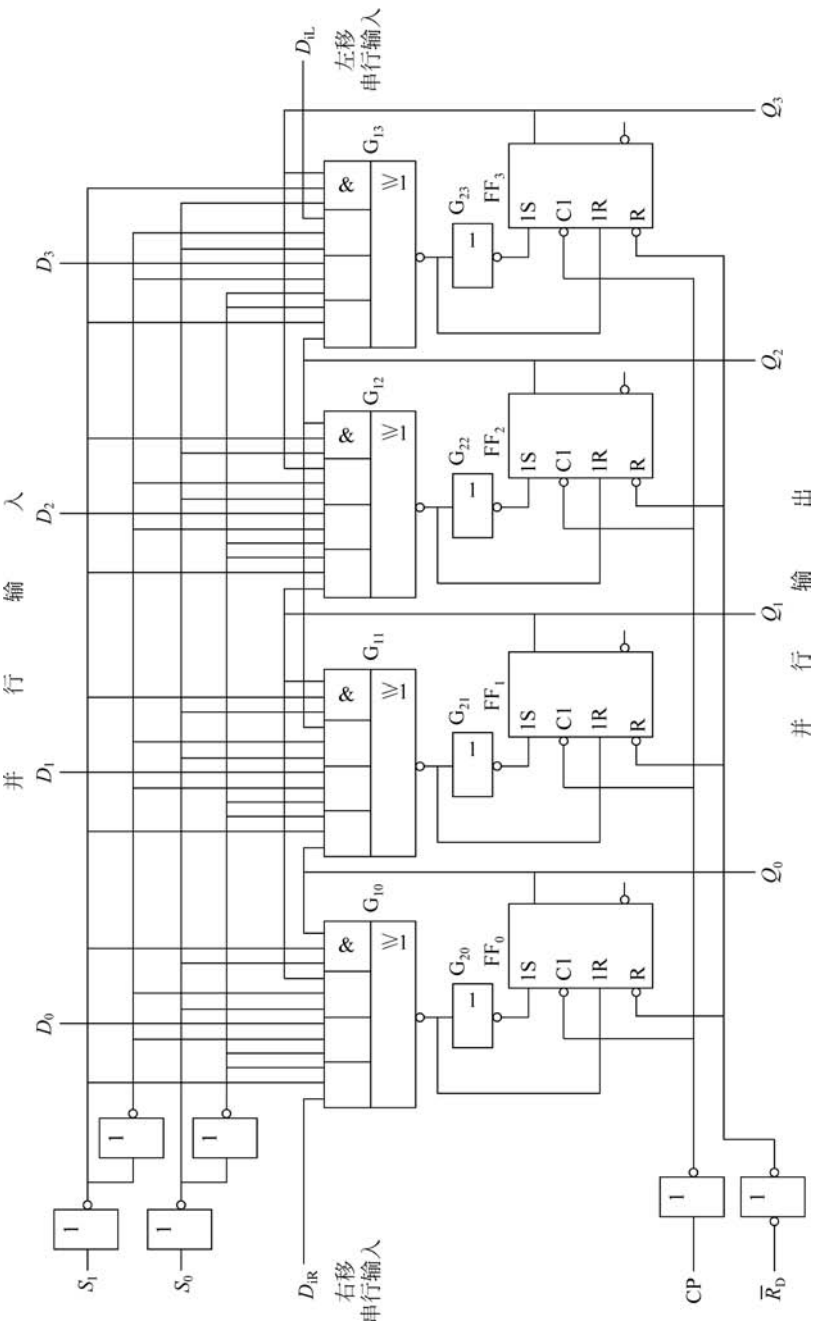


图 5-23　4 位双向移位寄存器 74S194A 的逻辑图

图 5-23 中的 D_{iR} 为数据右移串行输入端；D_{iL} 为数据左移串行输入端；$D_0 \sim D_3$ 为数据并行输入端；$Q_0 \sim Q_3$ 为数据并行输出端。移位寄存器的工作状态由控制端 S_1 和 S_0 的状态指定，其功能如表 5-11 所示。

表 5-11　74LS194A 的功能表

$\overline{R}_D$	S_1	S_0	工 作 状 态
0	d	d	置零
1	0	0	保持
1	0	1	右移
1	1	0	左移
1	1	1	并行输入

现以第二位触发器 FF_1 为例，分析一下 S_1、S_0 为不同取值时移位寄存器的工作状态。由图 5-23 可见，FF_1 的输入控制电路是由门 G_{11} 和门 G_{21} 组成的一个具有互补输出的 4 选 1 数据器的互补输出作为 FF_1 的输入信号。

当 $S_1=S_0=0$ 时，G_{11} 最右边的输入信号 Q_1^n 被选中，使触发器 FF_1 的输入为 $S=Q_1^n$，$R=\overline{Q}_1^n$，故 CP 上升沿到达时 FF_1 被置成 $Q_1^{n+1}=Q_1^n$。因此，移位寄存器工作在保持状态。

当 $S_1=S_0=1$ 时，G_{11} 左边第二个输入信号 D_1 被选中，使触发器 FF_1 的输入为 $S=D_1$、$R=\overline{D}_1$，故 CP 上升沿到达时 FF_1 被置成 $Q_1^{n+1}=D_1$，移位寄存器处于数据并行输入状态。

当 $S_1=0$，$S_0=1$ 时，G_{11} 最左边的输入信号 Q_0^n 被选中，使触发器 FF_1 的输入为 $S=Q_0^n$，$R=\overline{Q_0^n}$，故 CP 上升沿到达时 FF_1 被置成 $Q_1^{n+1}=Q_0^n$，移位寄存器工作在右移状态。

当 $S_1=1$，$S_0=0$ 时，G_{11} 右边第二个输入信号 Q_2^n 被选中，使触发器 FF_1 的输入为 $S=Q_2^n$，$R=\overline{Q}_2^n$，故 CP 上升沿到达时触发器被置成 $Q_1^{n+1}=Q_2^n$，这时移位寄存器工作在左移状态。

此外，$\overline{R}_D=0$ 时 $FF_0 \sim FF_3$ 将同时被置成 $Q=0$，所以正常工作时应使 $\overline{R}_D$ 处于高电平。

3. 计数器

在数字系统中计数器是使用最多的一种电路。它不仅能用于对时钟脉冲计数，还可以用于分频、定时、产生节拍脉冲和脉冲序列以及进行数字运算等。

计数器的种类繁多，本节主要讨论同步计数器。目前生产的同步计数器芯片基本上分为二进制和十进制两种，下面分别举例说明。

1）同步二进制计数器

图 5-24 是由 T 触发器构成的同步二进制加法计数器。由图可得到它的激励函数和输出函数的表达式为

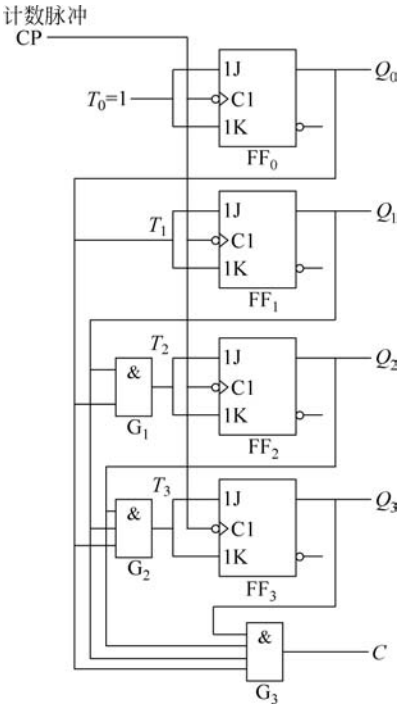


图 5-24　用 T 触发器构成的同步二进制加法计数器

视频讲解

$$T_0=1,\quad T_1=Q_0,\quad T_2=Q_0Q_1,\quad T_3=Q_0Q_1Q_2$$

$$C=Q_0Q_1Q_2Q_3$$

T 触发器的次态方程为

$$Q^{n+1}=T\overline{Q}+\overline{T}Q$$

将激励函数代入,得电路的次态方程组为

$$\begin{cases}Q_0^{n+1}=\overline{Q}_0\\ Q_1^{n+1}=Q_0\overline{Q}_1+\overline{Q}_0Q_1\\ Q_2^{n+1}=Q_0Q_1\overline{Q}_2+\overline{Q_0Q_1}Q_2\\ Q_3^{n+1}=Q_0Q_1Q_2\overline{Q}_3+\overline{Q_0Q_1Q_2}Q_3\end{cases}$$

整理得

$$Q_0^{n+1}=\overline{Q}_0$$

$$Q_1^{n+1}=Q_0\overline{Q}_1+\overline{Q}_0Q_1$$

$$Q_2^{n+1}=Q_0Q_1\overline{Q}_2+\overline{Q}_0Q_2+\overline{Q}_1Q_2$$

$$Q_3^{n+1}=Q_0Q_1Q_2\overline{Q}_3+\overline{Q}_0Q_3+\overline{Q}_1Q_3+\overline{Q}_2Q_3$$

将该方程组反映到卡诺图上得 ***Y*** 矩阵,如表 5-12 所示。

表 5-12　*Y* 矩阵

Q_1Q_0	Q_3Q_2			
	00	01	11	10
0　0	0 0 0 1	0 1 0 1	1 1 0 1	1 0 0 1
0　1	0 0 1 0	0 1 1 0	1 1 1 0	1 0 1 0
1　1	0 1 0 0	1 0 0 0	0 0 0 0	1 1 0 0
1　0	0 0 1 1	0 1 1 1	1 1 1 1	1 0 1 1

根据 ***Y*** 矩阵和输出函数,很容易得到该电路的状态图(见图 5-25)和时间图(见图 5-26)。

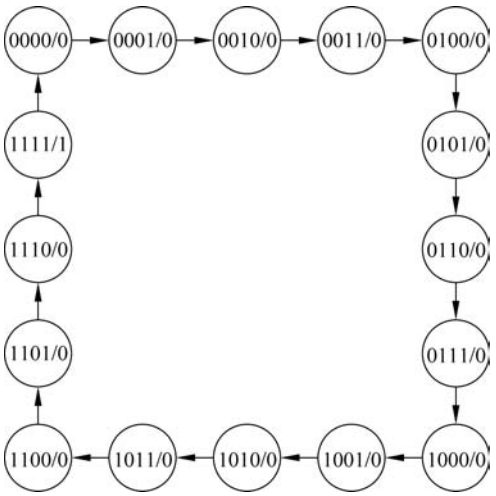


图 5-25　状态图

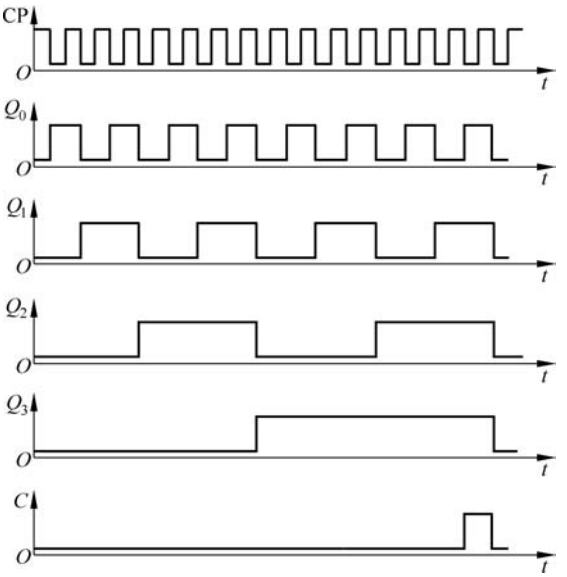


图 5-26　时间图

从时间图上可以看出，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 $f_0/2$、$f_0/4$、$f_0/8$、$f_0/16$。针对计数器的这种分频功能，也把它叫作分频器。此外，每输入 16 个计数脉冲，计数器工作一个循环，并在输出端 C 产生一个进位输出信号，所以又把这个电路叫十六进制计数器。n 位二进制计数器也称为 2^n 进制计数器，它所能计到的最大数为 2^n-1。

在实际生产的计数器芯片中，往往还附加一些控制电路，以增加电路的功能和使用的灵活性。如中规模集成芯片 74161，逻辑图如图 5-27 所示。这个电路除了二进制加法计数功能外，还具有预置数、保持和异步置零等功能。图 5-27 中 $\overline{\text{LD}}$ 为预置数控制端，$D_0 \sim D_3$ 为数据输入端，C 为进位输出端，$\overline{R}_D$ 为异步置零（复位）端，EP 和 ET 为工作状态控制端。74161 的功能表如表 5-13 所示。

表 5-13　4 位同步二进制计数器 74161 的功能表

CP	$\overline{R}_D$	$\overline{\text{LD}}$	EP	ET	工 作 状 态
d	0	*d*	*d*	*d*	置零
↑	1	0	*d*	*d*	预置数
d	1	1	0	1	保持
d	1	1	*d*	0	保持（但 $C=0$）
↑	1	1	1	1	计数

由图 5-27 可见，当 $\overline{R}_D=0$ 时所有触发器将同时被置零，而且置零操作不受其他输入端状态的影响。

当 $\overline{R}_D=1$，$\overline{\text{LD}}=0$ 时，电路工作在预置数状态，这时门 $G_{16} \sim G_{19}$ 的输出始终是 1，所以

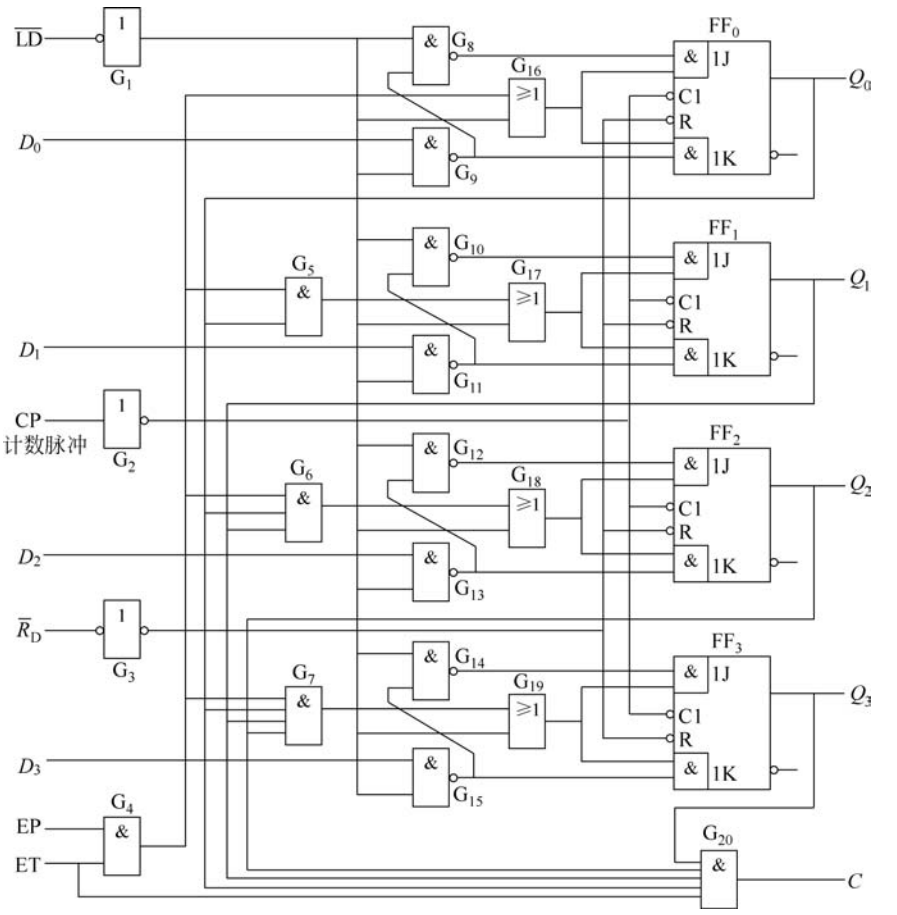


图 5-27　4 位同步二进制计数器 74161 的逻辑图

FF_0～FF_3 输入端 J、K 的状态由 D_0～D_3 的状态决定。例如，若 $D_0=1$，则 $J_0=1$，$K_0=0$，CP 上升沿到达后 FF_0 被置 1。

当 $\overline{R}_D=\overline{LD}=1$ 而 EP=0，ET=1 时，由于这时门 G_{16}～G_{19} 的输出均为 0，亦即 FF_0～FF_3 均处在 $J=K=0$ 的状态，所以 CP 信号到达时它们保持原来的状态不变，同时 C 的状态也得到保持。如果 ET=0，则 EP 不论为何状态，计数器的状态也将保持不变，但这时进位输出 C 等于 0。

当 $\overline{R}_D=\overline{LD}=EP=ET=1$ 时，电路工作在计数状态，与图 5-24 电路的工作状态相同。从电路的 0000 状态开始连续输入 16 个计数脉冲时，电路将从 1111 状态返回 0000 状态，C 端从高电平跳变至低电平，可以利用 C 端输出的高电平或下降沿作为进位输出信号。

74LS161 在内部电路结构形式上与 74161 有些区别，但外部引线的配置、引脚排列以及功能表都和 74161 相同。


视频讲解

2）同步十进制计数器

图 5-28 是用 T 触发器构成的同步十进制加法计数器电路。由图 5-28 可写出电路的激励函数、输出函数的表达式为

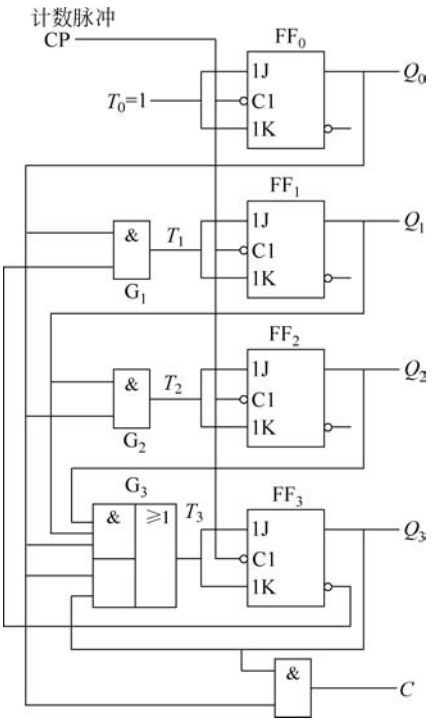


图 5-28　同步十进制加法计数器电路

$$\begin{cases} T_0 = 1 \\ T_1 = Q_0\overline{Q}_3 \\ T_2 = Q_0Q_1 \\ T_3 = Q_0Q_1Q_2 + Q_0Q_3 \\ C = Q_0Q_3 \end{cases}$$

T 触发器的次态方程为

$$Q^{n+1} = T\overline{Q} + \overline{T}Q$$

将激励函数代入，得电路的次态方程组为

$$\begin{cases} Q_0^{n+1} = \overline{Q}_0 \\ Q_1^{n+1} = Q_0\overline{Q}_3\overline{Q}_1 + \overline{Q_0\overline{Q}_3}Q_1 \\ Q_2^{n+1} = Q_0Q_1\overline{Q}_2 + \overline{Q_0Q_1}Q_2 \\ Q_3^{n+1} = (Q_0Q_1Q_2 + Q_0Q_3)\overline{Q}_3 + \overline{(Q_0Q_1Q_2 + Q_0Q_3)}Q_3 \end{cases}$$

整理得

$$\begin{aligned} Q_0^{n+1} &= \overline{Q}_0 \\ Q_1^{n+1} &= Q_0\overline{Q}_1\overline{Q}_3 + \overline{Q}_0Q_1 + Q_1Q_3 \\ Q_2^{n+1} &= Q_0Q_1\overline{Q}_2 + \overline{Q}_0Q_2 + \overline{Q}_1Q_2 \\ Q_3^{n+1} &= Q_0Q_1Q_2\overline{Q}_3 + \overline{Q}_0Q_3 \end{aligned}$$

将电路的次态方程组反映到卡诺图上，得 $\boldsymbol{Y}$ 矩阵如表 5-14 所示。

表 5-14　Y 矩阵

Q_1Q_0	Q_3Q_2			
	00	01	11	10
0　0	0 0 0 1	0 1 0 1	1 1 0 1	1 0 0 1
0　1	0 0 1 0	0 1 1 0	0 1 0 0	0 0 0 0
1　1	0 1 0 0	1 0 0 0	0 0 1 0	0 1 1 0
1　0	0 0 1 1	0 1 1 1	1 1 1 1	1 0 1 1

由 $\boldsymbol{Y}$ 矩阵很容易可得状态图如图 5-29 所示。从图上可看出有效序列有 10 个状态,进行十进制的加法计数,从 0000～1001 重复计数。另外 6 个状态为无效序列,但能自动进入有效序列,该电路具有自恢复功能。

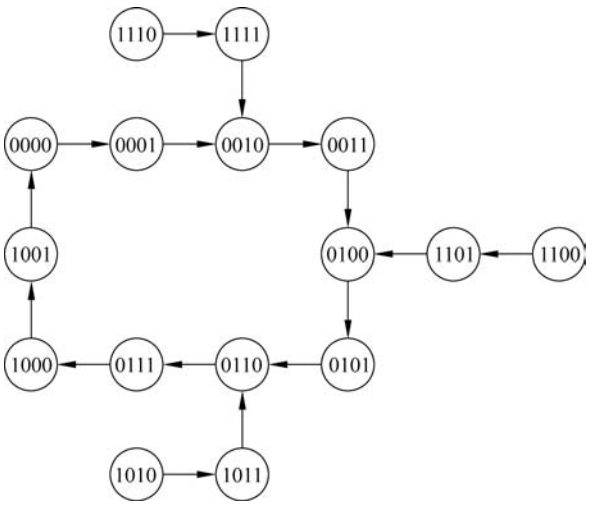


图 5-29　同步十进制加法计数器状态图

与二进制计数器类似,中规模集成芯片 74160 为同步十进制加法计数器,逻辑图如图 5-30 所示。它除了计数功能外,还有预置数、保持、异步置零等功能。图中的控制信号及功能表与上面讨论的 74161 完全一样,只是 74160 是十进制而 74161 是十六进制。

3) 任意进制计数器

从降低成本考虑,集成电路的定型产品必须有足够大的批量,因此目前常见的计数器芯片在记数进制上只做成应用较广的几种类型,如十进制、十六进制、7 位二进制、12 位二进制、14 位二进制等。如需要其他任意进制时,只能用现有产品的进制计数器加一些辅助电路来实现。

假定已有 N 进制计数器,而需要得到 M 进制计数器。下面分两种情况来讨论。

① $M<N$ 的情况。

在 N 进制计数器的顺序计数过程中,设法使之越过 $N-M$ 个状态,就可以得到 M 进制计数器了。实现跳跃的方法有置零法(或称复位法)和置数法(或称置位法)两种。

视频讲解

下面通过实例来说明这两种方法。

例 5-5　试利用同步十进制计数器 74160 接成同步六进制计数器。

解: 74160 的逻辑图及功能在前面已讨论过了,它兼有异步置零和同步置数功能,所以

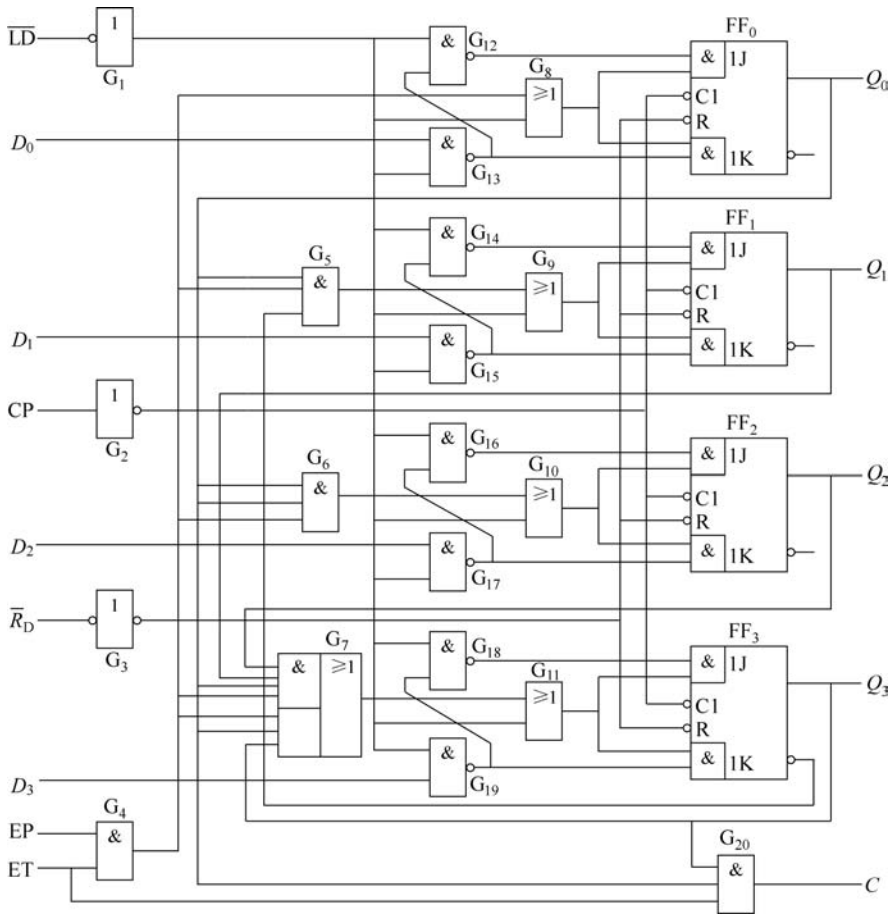


图 5-30　同步十进制加法计数器 74160 逻辑图

置零法和置数法均可采用。

如图 5-31 所示的电路是采用异步置零法接成的六进制计数器。当计数器从 0000(S_0)计成 $Q_3Q_2Q_1Q_0=0110$(即 S_M)状态时,担任译码器的门 G 输出低电平信号给 $\overline{R}_D$ 端,将计数器置零,回到 0000 状态。电路的状态图如图 5-32 所示。

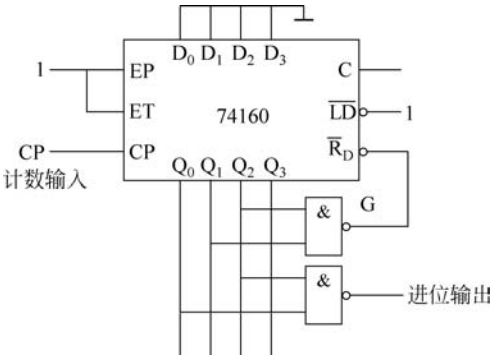


图 5-31　置零法将 74160 接成六进制计数器

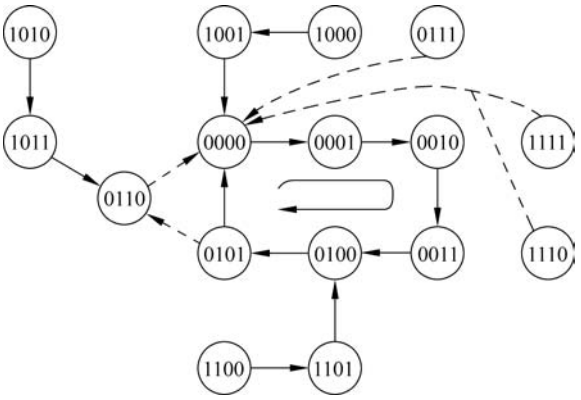


图 5-32　状态图

由于置零信号随着计数器被置零而立即消失,所以置零信号持续时间极短,如果触发器的复位速度有快有慢,则可能动作慢的触发器还未来得及复位,置零信号就已经消失,导致电路误动作。因此,这种接法的电路可靠性不高。

为了克服这个缺点,经常采用如图 5-33 所示的改进电路。图中的与非门 G_1 起译码器的作用,当电路进入 0110 状态时,它输出低电平信号。与非门 G_2 和 G_3 组成了基本 RS 触发器,以它 $\overline{Q}$ 端输出的低电平作为计数器的置零信号。

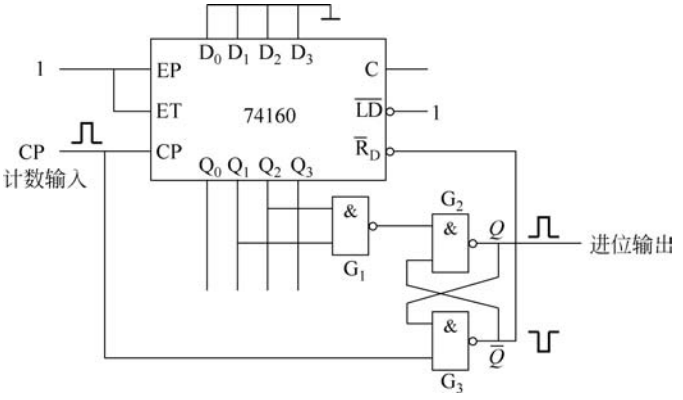


图 5-33　图 5-31 电路的改进

若计数器从 0000 状态开始计数,则第 6 个计数输入脉冲上升沿到达时计数器进入 0110 状态,G_1 输出低电平,将基本 RS 触发器置 1,$\overline{Q}$ 端的低电平立刻将计数器置零。这时虽然 G_1 输出的低电平信号随之消失了,但基本 RS 触发器的状态仍保持不变,因而计数器的置零信号得以维持。直到计数脉冲回到低电平以后,基本 RS 触发器被置零,$\overline{Q}$ 端的低电平信号才消失。可见,加到计数器 $\overline{R}_D$ 端的置零信号宽度与输入计数脉冲高电平持续时间相等。

同时,进位输出脉冲也可以从基本 RS 触发器的 Q 端引出。该脉冲的宽度与计数脉冲高电平宽度相等。有些计数器产品中,将 G_1,G_2,G_3 组成的附加电路直接制作在计数器芯片上,这样在使用时就不用外接附加电路了。

74160 是异步置零，一旦置零信号出现，立即把计数器清零，而不必等脉冲的到来。所以上面的计数器电路一进入 0110(S_M)状态后，立即又被置成 0000(S_0)状态，所以 S_M 状态仅在极短的瞬间出现，在稳定的状态循环中不包括 S_M 状态。而采用置数法就不一样了，因为 74160 是同步置数，产生了置数信号后再等下一个脉冲来到才完成置数，故产生置数的状态包含在稳定的状态循环中。

采用置数法时可以从计数循环中的任何一个状态置入适当的数值而跳越 $N-M$ 个状态，得到 M 进制计数器。图 5-34 中给出了两个不同的方案，其中图 5-34(a)的接法是用 $Q_3Q_2Q_1Q_0=0101$ 状态译码产生$\overline{LD}$信号，下一个 CP 信号到达时置入 0000 状态，从而跳过 0110～1001 这 4 个状态，得到六进制计数器，如图 5-35 中的实线所表示的那样。

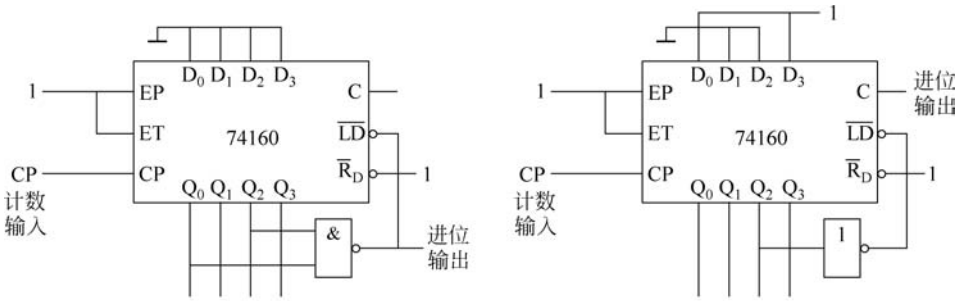


图 5-34　用置数法将 74160 接成六进制计数器

若采用图 5-34(b)电路的方案，则可以从 C 端得到进位输出信号。在这种接法下，用 0100 状态译码产生$\overline{LD}=0$ 信号，下个 CP 信号到来时置入 1001(如图 5-35 中的虚线所示)，因而循环状态中包含了 1001 这个状态，每个计数循环都会在 C 端给出一个进位脉冲。

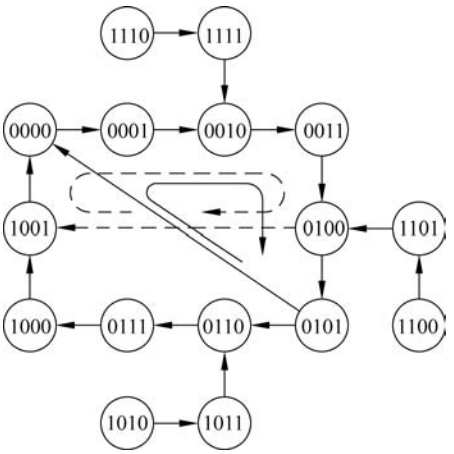


图 5-35　状态图

由于 74160 的预置数是同步式的，即$\overline{LD}=0$ 以后，还要等下一个 CP 信号到来时才置入数据，而这时$\overline{LD}=0$ 的信号已稳定地建立了，所以不存在异步置零法中因置零信号持续时间过短而可靠性不高的问题。

② $M>N$ 的情况。

这时必须用多片 N 进制计数器组合起来,才能构成 M 进制计数器。各片之间(或称为各级之间)的连接方式可分为串行进位方式、并行进位方式、整体置零方式和整体置数方式几种。下面仅以两级之间的连接为例说明这 4 种连接方式的原理。

若 M 可以分解为两个小于 N 的因数相乘,即 $M=N_1\times N_2$,则可采用串行进位方式或并行进位方式将一个 N_1 进制计数器和一个 N_2 进制计数器连接起来,构成 M 进制计数器。

在串行进位方式中,以低位片的进位输出信号作为高位片的时钟输入号;在并行进位方式中,以低位片的进位输出信号作为高位片的工作状态控制信号(计数的使能信号)。两片的 CP 输入端同时接计数输入信号。

视频讲解

例 5-6 试用两片同步十进制计数器接成百进制计数器。

解:本例中 $M=100$,$N_1=N_2=10$,将两片 74160 直接按并行进位方式或串行进位方式连接即得百进制计数器。

如图 5-36 所示的电路是并行进位方式的接法。以第(1)片的进位输出 C 作为第(2)片的 EP 和 ET 输入,每当第(1)片计成 9(1001)时 C 变为 1,下个 CP 信号到达时第(2)片为计数工作状态,计入 1,而第(1)片计成 0(0000),它的 C 端回到低电平。第(1)片的 EP 和 ET 恒为 1,始终处于计数工作状态。

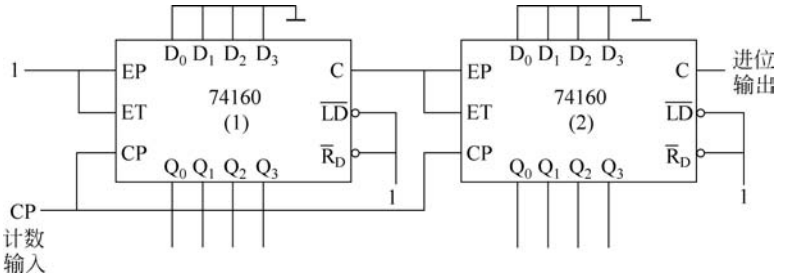


图 5-36 例 5-6 电路的并行进位方式

如图 5-37 所示的电路是串行进位方式的连接方法。两片 74160 的 EP 和 ET 恒为 1,都工作在计数状态。第(1)片每计到 9(1001)时 C 端输出变为高电平,经反相器后使第(2)片的 CP 端为低电平。下个计数输入脉冲到达后,第(1)片计成 0(0000)状态,C 端跳回低电平,经反相后使第(2)片的输入端产生一个正跳变,于是第(2)片计入 1。可见,在这种接法下两片 74160 不是同步工作的。

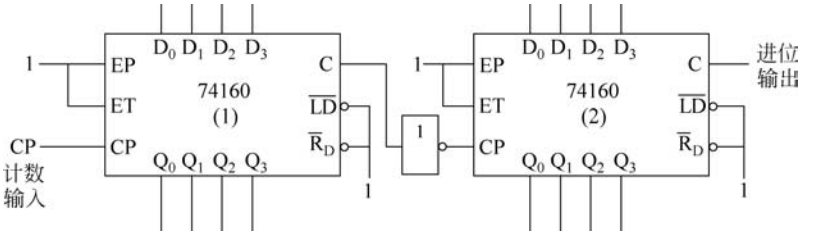


图 5-37 例 5-6 电路的串行进位方式

在 N_1、N_2 不等于 N 时，可以先将两个 N 进制计数器分别接成 N_1 进制计数器和 N_2 进制计数器，然后再以并行进位方式或串行进位方式将它们连接起来。

当 M 为大于 N 的素数时，不能分解成 N_1 和 N_2，上面讲的并行进位方式和串行进位方式就行不通了，这时必须采取整体置零方式或整体置数方式构成 M 进制计数器。

所谓整体置零方式，是首先将两片 N 进制计数器按最简单的方式接成一个大于 M 进制的计数器（例如 $N\times N$ 进制），然后在计数器计为 M 状态时译出异步置零信号 $\overline{R}_D=0$，将两片 N 进制计数器同时置零。这种方式的基本原理和 $M<N$ 时的置零法是一样的。

整体置数方式也一样，在 $N\times N$ 进制的基础上进行，基本原理和 $M<N$ 时的置数法类似，但要求已有的 N 进制计数器本身必须具备预置数功能。当 M 不是素数时整体置零法和置数法也可以使用。

例 5-7　试用两片同步十进制计数器 74160 接成二十九进制计数器。

解：因为 $M=29$ 是一个素数，所以必须用整体置零法或整体置数法构成二十九进制计数器。图 5-38 是整体置零方式的接法。首先将两片 74160 以并行进位方式连成一个百进制计数器。当计数器从全 0 状态开始计数，计入 29 个脉冲时，经门 G_1 译码产生的低电平信号立刻将两 74160 同时置零，于是便得到了二十九进制计数器。需要注意的是计数过程中第(2)片 74160 不出现 1001 状态，因而它的 C 端不能给出进位信号。而且，门 G_1 输出的脉冲持续时间极短，也不宜作进位输出信号。如果要求输出进位信号的持续时间为一个时钟信号周期，则应从电路的 28 状态译出当电路计入 28 个脉冲后门 G_2 输出变为低电平，第 29 个计数脉冲到达后门 G_2 的输出跳变为高电平。

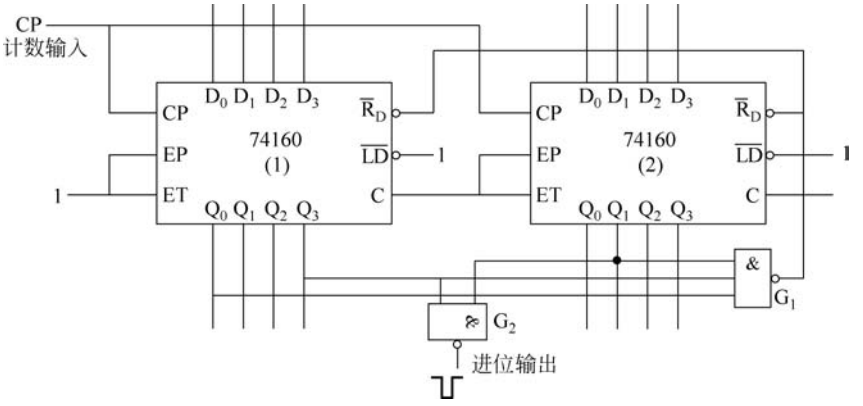


图 5-38　例 5-7 电路的整体置零方式

通过这个例子可以看到，整体置零法不仅可靠性较差，而且往往还要另加译码电路才能得到需要的进位输出信号。

采用整体置数方式可以避免置零法的缺点。如图 5-39 所示的电路是采用整体置数法接成的二十九进制计数器。首先仍需将两片 74160 接成百进制计数器，然后将电路的 28 状态译码产生 $\overline{LD}=0$ 信号，同时加到两片 74160 上，在下个计数脉冲（第 29 个输入脉冲）到达时，将 0000 同时置入两片 74160 中，从而得到二十九进制计数器。进位信号可以直接由门 G 的输出端引出。

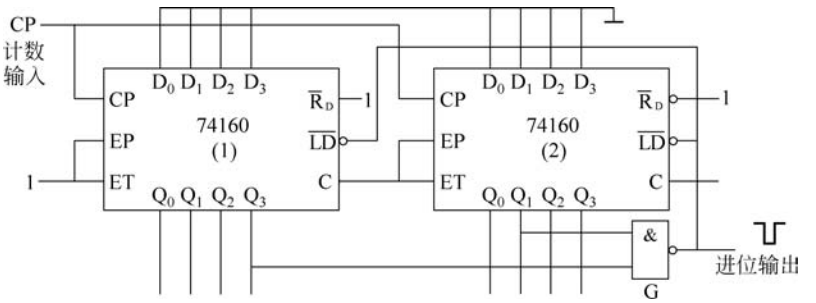


图 5-39　例 5-7 电路的整体置数方式

5.3　同步时序逻辑电路的设计

同步时序逻辑电路的设计也称同步时序逻辑电路的综合。实际上设计是分析的逆过程,就是根据给定的逻辑功能要求,设计出能实现其逻辑功能的时序电路。设计的流程如图 5-40 所示,一般步骤如下:

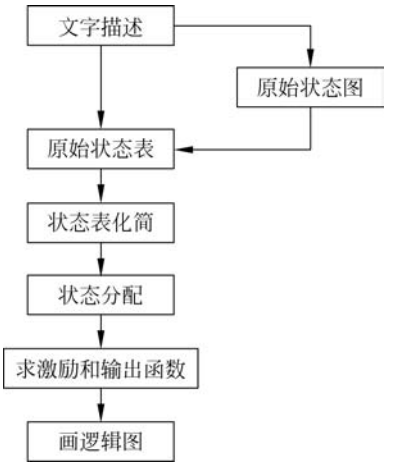


图 5-40　同步时序电路设计流程图

(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) 采用状态化简方法将原始状态表化为最简状态表。

(3) 进行状态分配(或状态赋值)。即将状态符号用代码表示,得到二进制形式的状态表。

(4) 根据二进制状态表和选用的触发器特性求电路的激励函数和输出函数。求激励函数可用表格法或代数法,具体方法在举例中讨论。

(5) 根据激励函数和输出函数的表达式,画出所要求的逻辑图。

一般说来,同步时序电路设计按上面 5 个步骤进行。但是,对于某些特殊的同步时序电路,由于状态数量和状态编码方案都已给定,上述设计步骤中的状态化简和状态编码便可以省略,从第(1)步直接跳到第(4)步。

视频讲解

5.3.1　建立原始状态表

建立原始状态表的方法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。目前还没有一个建立原始状态图的系统的算法,主要是采用直观的经验方法。设计一个时序电路首先应该考虑其包括几个状态、状态间如何进行转换、怎样产生输出。

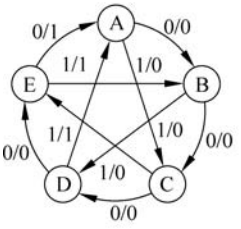
一般的过程是这样:首先假定一个初始状态 A,从这个初始状态 A 开始,每加入一个输入,就可确定其次态和输出。该次态可能是现态本身,也可能是已有的另一个状态,或是新

增加的一个状态。继续这个过程，直到每种输入的可能性、每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为止。

例 5-8 建立一个模 5 的加 1/加 2 计数器的状态图和状态表。

解：对于模 5 计数器，显然应有 5 个状态，设为 A～E，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，故需设一个控制输入信号 x，当 $x=0$ 时加 1，$x=1$ 时加 2，Z 为输出，表示有进位。

经以上分析后，可画出该计数器的原始状态图和状态表，如图 5-41 所示。



(a) 状态图

Q	x	
	0	1
A	B/0	C/0
B	C/0	D/0
C	D/0	E/0
D	E/0	A/1
E	A/1	B/1

x/Z

(b) 状态表

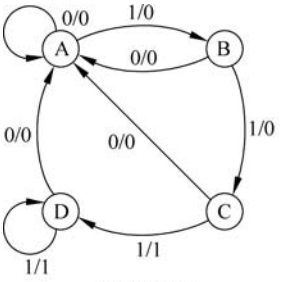
图 5-41 例 5-8 的状态图和状态表

例 5-9 有一个串行数据检测器。对它的要求是：连续输入 3 个或 3 个以上的 1 时输出为 1，其他输入情况输出为 0。例如：

输入序列 x：1 0 1 1 0 0 1 1 1 0 1 1 1 1 0

输出序列 Z：0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0

解：设电路在没有输入 1 以前的状态(初态)为 A，输入一个 1 以后的状态为 B，连续输入两个 1 以后的状态为 C，连续输入 3 个或 3 个以上 1 以后的状态为 D，此时输出 Z 为 1。当输入一个 0 时，不管当时电路处于何种状态，电路都将回到初始状态 A，表示检测器需要重新记录连续输入 1 的个数。根据以上分析可得该检测器的原始状态图和状态表如图 5-42 所示。



(a) 状态图

Q	x	
	0	1
A	A/0	B/0
B	A/0	C/0
C	A/0	D/1
D	A/0	D/1

x/Z

(b) 状态表

图 5-42 例 5-9 的状态图和状态表

5.3.2 状态表的化简

在建立原始状态表的过程中，为了满足给定的功能要求，可能引入了多余的状态。电路

中状态的数目越多,所需的存储元件就越多。因此,在得到原始状态表后,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,使实现它的电路最简单。

完全确定状态表和不完全确定状态表的化简方法有所不同。下面分别讨论。

1. 完全确定状态表的化简


视频讲解

完全确定状态表的化简是建立在状态等价这个概念的基础上的。为此先讨论等价的几个概念。

1) 等价的概念

① 等价状态:设A和B是时序电路状态表的两个状态,如果从A和B开始,任何加到时序电路上的输入序列均产生相同的输出序列,则称状态A和B为等价状态或等价状态对,并记为(A,B)或{A B}。等价状态可以合并。

② 等价状态的传递性:若状态A和B等价,状态B和C等价,则状态A和C也等价,记为(A,B),(B,C)→(A,C)。

③ 等价类:彼此等价的状态集合,称为等价类。若(A,B)和(B,C),则有等价类(A,B,C)。

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所有的最大等价类。下面介绍具体的化简方法。

2) 化简方法

这里介绍一种叫隐含表的方法。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。

根据等价状态的定义,两个状态是否等价的条件可归纳为如下两点:

第一,在各种输入取值下,它们的输出完全相同。

第二,在第一个条件满足的前提下,它们的次态满足下列条件之一,即

① 次态相同;

② 次态交错;

③ 次态循环;

④ 次态对等价。

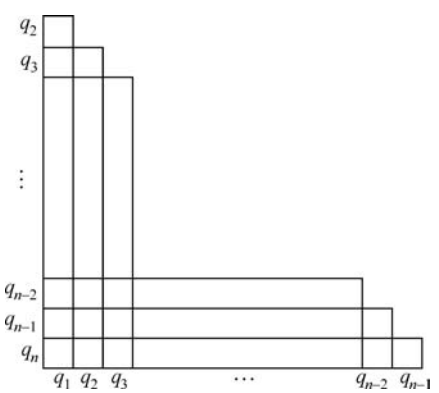


图 5-43 隐含表格式

这里的次态交错是指在某种输入取值下,如S_i的次态为S_j,S_j的次态为S_i。次态循环是次态之间的关系构成闭环,如S_i、S_j的次态是S_l、S_k,而S_l、S_k的次态又是S_i、S_j。

化简的具体步骤如下:

(1) 画隐含表。隐含表是一个三角形矩阵。设原始状态表有n个状态q_1～q_n,在隐含表的水平方向标以状态q_1,q_2,…,q_{n-1},垂直方向标以q_2,q_3,…,q_n,即垂直方向"去头",水平方向"少尾"。隐含表中每个小方格表示一个状态对(q_i,q_j)。隐含表的格式如图5-43所示。

(2) 顺序比较。顺序比较隐含表中各状态之间的关系，比较结果有如下三种情况：

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，q_i 和 q_j 等价，在隐含表相应的方格内标以√。

② q_i 和 q_j 输出不同，表示 q_i 和 q_j 不等价，在对应的方格内标以×。

③ q_i 和 q_j 输出相同，但次态既不相同，也不交错。q_i 和 q_j 是否等价有待进一步考察，在对应的方格内标以 q_i 和 q_j 的次态对。

(3) 关联比较。顺序比较中不能确定的关系的状态对标在方格中，由关联比较进一步考察。这一步在隐含表上直接进行，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。

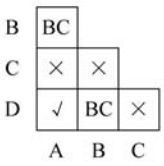
(4) 找最大等价类，作最简状态表。关联比较后，由等价的传递性可确定最大等价类。

注意：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 5-10 化简如图 5-44(a)所示的原始状态表。

Q	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

Q	x	
	0	1
a	a/0	b/0
b	a/0	c/0
c	a/0	c/1

(c) 最简状态表

图 5-44 例 5-10 图

解：化简步骤如下。

① 画隐含表，如图 5-44(b)所示。

② 顺序比较，结果如图 5-44(b)所示。

③ 关联比较。AB→BC→×

BD→BC→×

说明 BC 不等价，那么 AB，BD 也不等价。

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)

得最简状态表如图 5-44(c)所示。

例 5-11 化简如图 5-45(a)所示的原始状态表。

解：化简步骤如下。

① 画隐含表，如图 5-45(b)所示。

② 顺序比较，结果如图 5-45(b)所示。

③ 关联比较。

Q	x	
	0	1
A	C/0	B/1
B	F/0	A/1
C	D/0	G/0
D	D/1	E/0
E	C/0	E/1
F	D/0	G/0
G	C/1	D/0

(a) 原始状态表

	A	B	C	D	E	F
B	CF					
C	×	×				
D	×	×	×			
E	BE	AE CF	×	×		
F	×	×	√	×	×	
G	×	×	×	CD DE	×	×

(b) 隐含表

Q	x	
	0	1
a	b/0	a/1
b	c/0	d/0
c	c/1	a/0
d	b/1	c/0

(c) 例5-11的最简状态表

图 5-45　例 5-11 的图

AB→CF→√,所以 AB 等价;

AE→BE→CF→√,AE,BE 构成循环。

所以 AE,BE 都等价。

DG →CD→×,则 DG 不等价。

↘ DE→×

④ 列出最大等价类。

本例中得最大等价类为

(A,B,E),(C,F),(D),(G)

将最大等价类(A,B,E),(C,F),(D),(G)分别用新符号 a,b,c,d 表示,得最简状态表如图 5-45(c)所示。

视频讲解

2. 不完全确定的状态表的化简

对于不完全确定的状态表的化简是建立在状态相容概念的基础上的。为此先讨论相容的几个概念。

1) 相容概念

① 相容状态。设 A 和 B 是时序电路状态表的两个状态,如果从 A 和 B 开始,任何加到时序电路的有效输入序列均产生相同的输出序列(除不确定的那些位之外),那么 A 和 B 是相容的,记作(A,B)。相容状态可合并。

② 相容状态无传递性。(A,B),(B,C),但不一定有(A,C)。

③ 相容类。所有状态之间都两两相容的状态集合。

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。

2）化简方法

与完全确定的状态表的化简过程大致相同，主要有以下几步：

(1) 作隐含表，找相容状态对。

(2) 画合并图，找最大相容类。合并图就是在圆周上标上代表状态的点，点与点之间的连线表示两状态之间的相容关系，而所有点之间都有连线的多边形就构成一个最大相容类。

(3) 做出最简状态表。这一步与完全给定的状态表化简不一样。首先需要从最大相容类(或相容类)中选出一组能覆盖原始状态表全部状态的相容类，这一组相容类必须满足以下三个条件。

① 覆盖性。所选相容类集合应包含原始状态表的全部状态。

② 最小性。所选相容类个数应最小。

③ 闭合性。所选相容类集合中的任一相容类，在原始状态表中任一输入条件下产生的次态应该属于该集合中的某一个相容类。

同时具有覆盖、最小、闭合三个条件的相容类集合，称为最小闭覆盖，这就组成了最简状态表。

例 5-12 简化如图 5-46(a)所示的状态表。

Q	x		
	0	1	Z
A	B	D	0
B	B	D	*d*
C	A	E	1
D	*d*	E	1
E	F	*d*	1
F	*d*	C	*d*

(a) 原始状态表

	A	B	C	D	E
B	√				
C	×	AB DE			
D	×	DE	√		
E	×	BF	AF	√	
F	CD	CD	CE	CE	√

(b) 隐含表

图 5-46 例 5-12 的图

解：化简步骤如下。

① 作隐含表，找相容状态对。隐含表如图 5-46(b)所示。顺序比较后进行关联比较。

② AF→CD→√

BC→ AB→√

└→DE→√

BD→DE→√

BE→BF→CD→√

CE→AF→√

CF→CE→√

DF→CE→√

则得到全部相容状态对。

(A,B)，(A,F)，(B,C)，(B,D)，(B,E)，(B,F)，

(C,D)，(C,E)，(C,F)，(D,F)，(D,E)，(E,F)

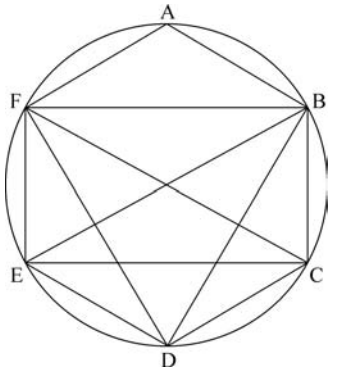


图 5-47　合并图

③ 作合并图,求最大相容类。

状态合并图如图 5-47 所示。图中 B,C,D,E,F 各点都有连线,构成一个全互连多边形,A,B,F 构成一个三角形。于是,找到两个最大相容类。

(A,B,F),(B,C,D,E,F)

④ 作最简状态表。

从最大相容类和相容类中选择一组能覆盖原始状态表中全部状态的相容类,假设就选两个最大相容类(A,B,F),(B,C,D,E,F),作闭合覆盖表,检查其是否满足覆盖性和闭合性。闭合覆盖表如表 5-15 所示。

表 5-15　例 5-12 的闭合覆盖表

相容类	覆　盖						闭　合	
	A	B	C	D	E	F	$X=0$	$X=1$
ABF	A	B				F	B	CD
BCDEF		B	C	D	E	F	ABF	CDE

从表 5-15 可看出:相容类集合(A,B,F),(B,C,D,E,F)覆盖了状态表图 5-46(a)的全部状态,而且每个相容类在任何一种输入情况下的次态集合都完全落在相容类集合中的某个相容类中,因此它们满足了闭合和覆盖这两个条件。此外,相容类的数目已不能再少,满足了最小条件。如果令 A′=(A,B,F),B′=(B,C,D,E,F),则可得最简状态表如表 5-16 所示。从表中可看出,当 $x=0$ 时,A′的次态为 A′或 B′,因为(A,B,F)在 $x=0$ 时的次态为 B,而 B 既属于相容类(A,B,F),又属于相容类(B,C,D,E,F),因此在表中相应位置填入 A′,B′,表明状态 A′在 $x=0$ 时的次态既可以是 A′也可以是 B′。进一步考虑,该最简状态表中只有 A′和 B′两个状态,所以可用无关项 d 表示,如表 5-17 所示。这样处理有利于相应电路的简化,这在后面进行设计电路时可得到进一步理解。有时选取的最大相容类作为相容类集合在闭合检查并不能使次态都属于某个相容类,即会出现不闭合的情况,这时可调整相容类集合的选取,有时不选用最大的相容类而选其中的一个子集反而能满足闭合、覆盖、最小的条件,得到最简状态表。这里不再举例说明,读者可参阅有关参考文献。

表 5-16　最简状态表

Q	x		Z
	0	1	
A′	A′B′	B′	0
B′	A′	B′	1

表 5-17　处理后的最简状态表

Q	x		Z
	0	1	
A′	d	B′	0
B′	A′	B′	1

视频讲解

5.3.3　状态分配

所谓状态分配,是指给最小化状态表中的每个字母或数字表示的状态,指定一个二进制代码,形成二进制状态表。

一般情况下,采用的状态编码方案不同,所得到的输出函数和激励函数的表达式也不同,从而设计出来的电路的复杂程度也不同。因此,状态编码的任务是:

(1) 确定状态编码的长度(即触发器的位数)。

(2) 寻找一种最佳的或接近最佳的状态分配方案,以便使所设计的时序电路最简要。

第一个任务较简单,设最简状态表的状态数为 N,状态编码的长度为 n,状态数 N 与状态编码长度 n 的关系为

$$2^{n-1} < N \leqslant 2^n$$

例如,某状态表的状态数 $N=4$,则状态分配时,二进制代码的位数应为 $n=2$,即需用两位触发器。

第二个任务就没有这么简单。因为状态编码长度确定后,究竟用哪种二进制代码代替哪个状态,这可以有许多种状态分配方案。

一般地,如状态数为 N,状态编码长度为 n,则可能的分配方案数 K_S 为

$$K_S = \frac{2^n!}{(2^n - N)!}$$

例如 $N=4$,$n=2$ 时,有 24 种状态分配方案,如表 5-18 所示。这 24 种分配方案中,实际上只有 3 种独立的分配方案,其他的方案实质上是等效的(对于电路的难易程度),彼此独立的分配方案 K_U 为

$$K_U = \frac{(2^n - 1)!}{(2^n - N)!\ n!}$$

当变量数目增加时,其分配方案的数就会急剧增大。表 5-19 表明了状态数与状态分配方案的关系。

表 5-18　$N=4$,$n=2$ 时的全部分配方案

状　态	方　案											
	1	2	3	4	5	6	7	8	9	10	11	12
A	00	10	01	11	00	01	10	11	00	10	01	11
B	01	11	00	10	10	11	00	01	11	01	10	00
C	11	01	10	00	11	10	01	00	01	11	00	10
D	10	00	11	01	01	00	11	10	10	00	11	01
状　态	方　案											
	13	14	15	16	17	18	19	20	21	22	23	24
A	00	01	10	11	00	10	01	11	00	01	10	11
B	11	10	01	00	10	00	11	01	01	00	11	10
C	10	11	00	01	01	11	00	10	10	11	00	01
D	01	00	11	10	11	01	10	00	11	10	01	00

表 5-19　状态数与状态分配方案总数的关系

状态数 N	二进制代码位数 n	独立分配方案数 K_U	状态数 N	二进制代码位数 n	独立分配方案数 K_U
1	0	—	6	3	420
2	1	1	7	3	840
3	2	3	8	3	840
4	2	3	9	4	10 810 800
5	3	140	10	4	75 675 600

在如此众多的状态分配方案中找出一种最佳的分配方案十分困难,且它还与采用什么类型的触发器有关系,因此没有必要将所有分配方案研究一遍。在实际工作中,常采用经验的方法,通过按一定的原则进行分配来获得接近最佳的分配方案。

状态分配的原则为:

(1) 在相同输入条件下,次态相同,现态应相邻编码。

(2) 在不同输入条件下,同一现态的次态应相邻编码。

(3) 输出完全相同,两个现态应相邻编码。

以上三个原则中,第一条最重要,应优先考虑。下面举例说明。

例 5-13 对表 5-20 的最简状态表进行状态分配。

表 5-20 例 5-13 状态表

Q	X	
	0	1
A	C/0	D/0
B	C/0	A/0
C	B/0	D/0
D	A/1	B/1

解:有 4 个状态,选用两位触发器 y_1y_0。

根据原则(1),AB、AC 应相邻编码;

根据原则(2),CD、AC、BD、AB 应相邻编码;

根据原则(3),AB、AC、BC 应相邻编码。

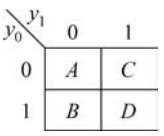


图 5-48 状态分配

综合上述要求,AB、AC 应给予相邻编码,这是三个原则都要求的。借用卡诺图,很容易得到满足上述相邻要求的状态分配方案,如图 5-48 所示。根据该图可得状态编码为

$$A=00,\quad B=01,\quad C=10,\quad D=11$$

将上述编码代入状态表得二进制状态表如表 5-21 所示。当然,上述分配方案不是唯一的。大多数情况下,根据以上三个原则进行状态分配是有效的。不同的状态分配方案并不影响同步时序电路的逻辑功能及稳定性,仅影响电路的复杂程度。

表 5-21 二进制状态表

y_1y_0	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

5.3.4 求激励函数和输出函数

在求出了二进制状态表后,可用表格法或代数法求激励函数、输出函数,进而可画出逻辑图。

1. 表格法


视频讲解

(1) 将二进制状态表变换成 ***Y*-*Z*** 矩阵。

将二进制状态表排成卡诺图的形式，即得 ***Y*-*Z*** 矩阵。例如，将表 5-21 的二进制状态表变换成 ***Y*-*Z*** 矩阵，如表 5-22 所示。

(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^{n+1}，而由现态 y_i 向次态 y_i^{n+1} 的转换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^{n+1} 代之以相应的触发器的激励值，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 JK 触发器来实现表 5-22 的 ***Y*-*Z*** 矩阵，则它的激励矩阵和输出矩阵如表 5-23 和表 5-24 所示。

表 5-22　*Y*-*Z* 矩阵

y_1y_0	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

表 5-23　激励矩阵 J_1K_1 和 J_0K_0

y_1y_0	x	
	0	1
0　0	$1d$，$0d$	$1d$，$1d$
0　1	$1d$，$d1$	$0d$，$d1$
1　1	$d1$，$d1$	$d1$，$d0$
1　0	$d1$，$1d$	$d0$，$1d$

表 5-24　输出矩阵 *Z*

y_1y_0	x	
	0	1
0　0	0	0
0　1	0	0
1　1	1	1
1　0	0	0

关于触发器的激励表在第 4 章已讨论过，为了设计电路方便查阅，将 RS、JK、D、T 触发器的激励表列于表 5-25～表 5-28，表中 Q 为现态，Q^{n+1} 为次态。

表 5-25　RS 触发器的激励表

Q	Q^{n+1}	R	S
0	0	d	1
0	1	1	0
1	0	0	1
1	1	1	d

表 5-26　JK 触发器的激励表

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

表 5-27　D 触发器的激励表

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

表 5-28　T 触发器的激励表

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

(3) 由激励和输出矩阵，求激励函数和输出函数。

激励矩阵可以看成各个输入激励填在同一个卡诺图上。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即可。同理，由输出矩阵可写出输出函数的最简表达式。

例如，根据表 5-23 和表 5-24 可得 J_1，K_1，J_0，K_0，Z 这 5 个卡诺图，如图 5-49 所示，并

由此可写出激励函数、输出函数表达式。

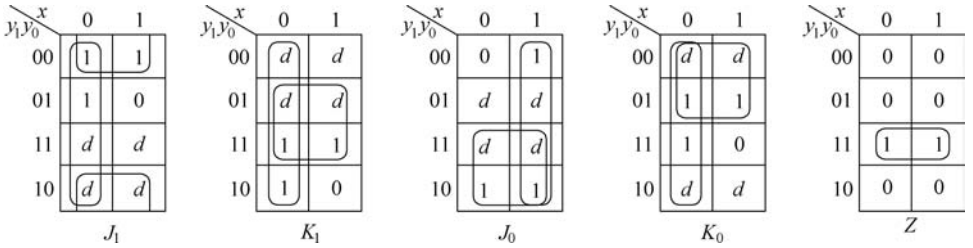


图 5-49　卡诺图

$$J_1=\bar{x}+\bar{y}_0 \quad K_1=y_0+\bar{x} \quad J_0=x+y_1 \quad K_0=\bar{x}+\bar{y}_1 \quad Z=y_1y_0$$

视频讲解

2. 代数法

根据 ***Y*-*Z*** 矩阵(二进制状态表)写出电路的次态方程如下：

$$y_1^{n+1}=\bar{x}\bar{y}_1+x\bar{y}_0$$

$$y_0^{n+1}=x\bar{y}_0+xy_1+y_1\bar{y}_0$$

因为 JK 触发器的次态方程为

$$Q^{n+1}=J\bar{Q}+\bar{K}Q$$

将电路的次态方程转换成以下形式：

$$\begin{aligned}y_1^{n+1}&=\bar{x}\bar{y}_1+x\bar{y}_0=\bar{x}\bar{y}_1+x\bar{y}_0(\bar{y}_1+y_1)\\&=\bar{x}\bar{y}_1+x\bar{y}_0\bar{y}_1+x\bar{y}_0y_1=(\bar{x}+x\bar{y}_0)\bar{y}_1+x\bar{y}_0y_1\\&=(\bar{x}+\bar{y}_0)\bar{y}_1+x\bar{y}_0y_1\end{aligned}$$

$$\begin{aligned}y_0^{n+1}&=x\bar{y}_0+xy_1+y_1\bar{y}_0=(x+y_1)\bar{y}_0+xy_1(y_0+\bar{y}_0)\\&=(x+y_1)\bar{y}_0+xy_1y_0\end{aligned}$$

与 JK 触发器次态方程比较得激励函数为

$$J_1=\bar{x}+\bar{y}_0 \quad K_1=\overline{x\bar{y}_0}=\bar{x}+y_0$$

$$J_0=x+y_1 \quad K_0=\overline{xy_1}=\bar{x}+\bar{y}_1$$

5.4　VHDL 时序电路的设计特点

视频讲解

5.4.1　电路的时钟控制

时序电路的输出和当前的输入以及历史状态有关，它具有“记忆”功能。常用的时序单元电路主要有寄存器、计数器等。构成这些单元电路的基础是触发器、时钟、复位/置位等信号。

时钟信号通常描述时序电路程序的执行条件。时钟边沿分上升沿和下降沿。一般时序电路的同步点在上升沿。为了描述时钟的属性，可以使用时钟信号的属性描述。时钟信号上升沿的属性描述表达式可写为

```
clk'event  AND  clk = '1';
```

同理，下降沿的属性描述只需将表达式中的 clk＝'1'改为 clk＝'0'。

在 VHDL 中，时序电路总是以时钟进程的形式来描述的，其描述方法有两种。

(1) 在时序电路描述中，时钟信号作为敏感信号显式地出现在 PROCESS 语句后的括号里。一般描述格式为

```
PROCESS(时钟信号名[,其他敏感信号])
BEGIN
 IF 时钟边沿表达式 THEN
   {语句; }
 END IF;
END PROCESS;
```

(2) 在时序电路描述中，时钟不列入进程的敏感信号，而用 WAIT ON 语句来控制程序的执行。在这种方式中，进程通常停留在 WAIT ON 语句上，这个点也称为进程的同步点，只有在时钟信号到来且满足边沿条件时，其余的语句才能执行。一般描述格式为

```
PROCESS
  BEGIN
    WAIT ON 时钟信号名 UNIT 时钟边沿表达式
    {语句; }
END PROCESS;
```

注意：对时钟边沿说明时，一定要说明是上升沿还是下降沿，WAIT ON 语句只能放在进程的最前面或最后面。

时序电路的初始状态一般由复位/置位信号来设置，设置方式有两种。

(1) 同步复位/置位方式：所谓同步复位/置位方式就是在复位/置位信号有效且给定的时钟边沿到来时，时序电路才被复位/置位。一般格式为

```
PROCESS(时钟信号名)
  BEGIN
    IF 时钟边沿表达式 AND 复位置位条件表达式  THEN
      [复位/置位语句; ]
    ELSE
      [其他执行语句; ]
    END IF;
  END PROCESS;
```

或

```
PROCESS
  BEGIN
    WAIT ON 时钟信号名 UNTIL 时钟边沿表达式
    IF 复位/置位条件表达式 THEN
        [复位/置位语句; ]
    ELSE
        [其他执行语句; ]
    END IF;
  END  PROCESS;
```

(2) 异步复位/置位方式：所谓异步复位/置位，就是复位/置位信号有效时，电路立即复位/置位，与时钟信号无关。在描述异步复位/置位电路时，在进程的敏感表中应同时加入时钟信号和复位/置位信号。一般格式为

```
PROCESS(时钟信号,复位/置位信号)
BEGIN
    IF 复位/置位条件表达式 THEN
        [复位/置位语句; ]
    ELSIF 时钟边沿表达式 THEN
        [其他执行语句; ]
    END IF;
END PROCESS;
```

视频讲解

5.4.2 状态图的 VHDL 描述

利用 VHDL 设计时序电路，不需要按照传统的设计方法进行烦琐的状态简化、状态分配、求解激励函数和输出函数等就可以简便地根据状态转移图直接进行描述。所有的状态均可表示为 CASE_WHEN 结构中的一条 CASE 语句，而状态的转移则通过 IF_THEN_ELSE 语句实现。时序电路分为 Moore 型和 Mealy 型电路，其 VHDL 描述略有差别。

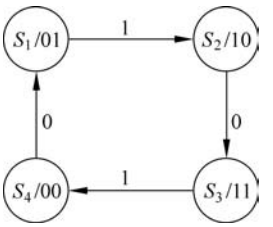


图 5-50 Moore 型电路的状态转移图

例 5-14 用 VHDL 描述 Moore 型电路的状态转移，状态转移图如图 5-50 所示。

解：VHDL 描述如下。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity moore is
  port(clk,datain,reset:in std_logic;
       dataout:out std_logic_vector(1 downto 0));
end moore;
architecture a of moore is
type state_type is (s1,s2,s3,s4);                    --用户自己定义的枚举类型
signal state:state_type;                             --信号声明
begin
  demo_process:process(clk,reset)                    --状态转移进程,clk,reset 为敏感信号
       begin
         if reset = '1' then state <= s1;            --初始状态为 s1,异步设置
         elsif clk'event and clk = '1' then          --当 clk 上升沿到来时执行下面的语句
               case state is
                  when s1 => if datain = '1' then
                                       state <= s2;
                                       end if;
                  when s2 => if datain = '0' then
                                       state <= s3;
                                       end if;
                  when s3 => if datain = '1' then
```

```
                                    state <= s4;
                                    end if;
                when s4 => if datain = '0' then
                                    state <= s1;
                                    end if;
                    end case;
                end if;
  end process;
  output_p:process(state)                    -- 输出变化进程,状态为敏感信号
            begin
              case state is
                when s1 => dataout <= "01";
                when s2 => dataout <= "10";
                when s3 => dataout <= "11";
                when s4 => dataout <= "00";
              end case;
            end process;
end a;
```

Mealy 型电路的 VHDL 描述与上面的程序大体相同,差别就在于输出变化进程中的输出信号需要根据输入信号的变化来确定输出值,可以用 IF_THEN_ELSE 语句来实现。

例 5-15 用 VHDL 描述 Mealy 型电路的状态转移,状态转移图如图 5-51 所示。

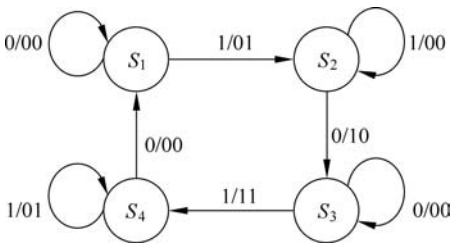


图 5-51 Mealy 型电路的状态转移图

解:VHDL 描述如下。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity mealy is
  port(clk,datain,reset:in std_logic;
       dataout:out std_logic_vector(1 downto 0));
end mealy;
architecture a of mealy is
type state_type is (s1,s2,s3,s4);          -- 用户自己定义的枚举类型
signal state:state_type;                   -- 信号声明
begin
  demo_process:process(clk,reset)          -- 状态转移进程,clk,reset 为敏感信号
       begin
         if reset = '1' then state <= s1;  -- 初始状态为 s1,异步设置
         elsif clk'event and clk = '1' then -- 当 clk 上升沿到来时执行下面的语句
```

```
                case state is
                   when s1 => if datain = '1' then
                                        state <= s2;
                                        end if;
                   when s2 => if datain = '0' then
                                        state <= s3;
                                        end if;
                   when s3 => if datain = '1' then
                                        state <= s4;
                                        end if;
                   when s4 => if datain = '0' then
                                        state <= s1;
                                        end if;
                   end case;
                end if;
  end process;
  output_p:process(state)                          --输出变化进程,状态为敏感信号
             begin
              case state is
                when s1 => if datain = '1' then dataout <= "01";
                                                  --输出值取决于输入值与现态
                                     else  dataout <= "00";
                                end if;
                when s2 => if datain = '0' then dataout <= "10";
                                     else  dataout <= "00";
                                end if;
                when s3 => if datain = '1' then dataout <= "11";
                                     else  dataout <= "00";
                                end if;
                when s4 => if datain = '0' then dataout <= "00";
                                     else  dataout <= "01";
                                end if;
                  end case;
             end process;
end a;
```

从上述两个例子可以看出,Mealy 型电路的输出是现态和现输入的函数,而 Moore 型电路的输出只与现态有关。

5.5 同步时序逻辑电路设计举例

以上花了较多的篇幅讨论了同步时序逻辑电路的设计方法。一般而言,设计方法要比分析方法复杂一些,对于前面讲的设计步骤,应根据实际情况灵活运用。下面举几个设计实例。

例 5-16 将例 5-9 的 111…序列检测器的问题进一步完成设计。

解:在例 5-9 中已经得到该检测器的原始状态表。现重列出如表 5-29 所示,经简化为表 5-30。

表 5-29　原始状态表

Q	x	
	0	1
A	A/0	B/0
B	A/0	C/0
C	A/0	D/1
D	A/0	D/1

表 5-30　最简状态表

Q	x	
	0	1
A	A/0	B/0
B	A/0	C/0
C	A/0	C/1

简化状态表共三个状态，所以需要两位触发器 Q_1 和 Q_0。根据状态分配的原则，一种较简单的分案如图 5-52 所示。根据这个状态分配方案可得二进制状态表(***Y*-*Z*** 矩阵)，如表 5-31 所示。

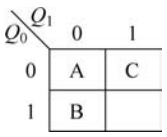


图 5-52　状态分配

若选用 JK 触发器作为存储元件。如用表格法，可根据 JK 触发器的激励表得到电路的激励矩阵，如表 5-32 所示。

表 5-31　***Y*-*Z*** 矩阵

Q_1Q_0	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

表 5-32　激励矩阵 J_1K_1 和 J_0K_0

Q_1Q_0	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_0,K_0 和输出函数 Z 的卡诺图，如图 5-53 所示。

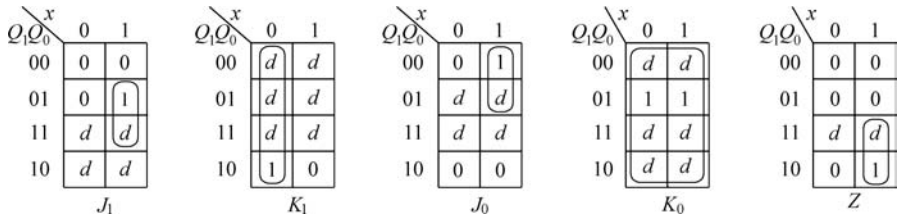


图 5-53　卡诺图

由卡诺图可得激励函数和输出函数为

$$J_1=xQ_0,\quad K_1=\bar{x},\quad J_0=x\bar{Q}_1,\quad K_0=1,\quad Z=xQ_1$$

如用代数法，则可根据 ***Y*-*Z*** 矩阵先写出电路的次态方程。

$$Q_1^{n+1}=xQ_1+xQ_0$$

$$Q_0^{n+1}=x\bar{Q}_1\bar{Q}_0$$

因 JK 触发器的次态方程为

$$Q^{n+1}=J\bar{Q}+\bar{K}Q$$

将电路次态方程转换成以下形式：

$$Q_1^{n+1}=xQ_1+xQ_0(Q_1+\bar{Q}_1)=xQ_0\bar{Q}_1+xQ_1$$

$$Q_0^{n+1}=x\bar{Q}_1\bar{Q}_0=x\bar{Q}_1\bar{Q}_0+0Q_0$$

与 JK 触发器次态方程相比较就得激励函数为

$$J_1 = xQ_0, \quad K_1 = \bar{x}, \quad J_0 = x\bar{Q}_1, \quad K_0 = 1$$

结果与表格法一致,根据激励函数和输出函数可得逻辑图如图 5-54 所示。从这个逻辑图可推出实际的电路状态图如图 5-55 所示。该状态图表明,当电路进入无效状态 11 后,若 $x=1$ 则次态转入 10;若 $x=0$ 则次态转入 00,因此这个电路是能够自启动的。

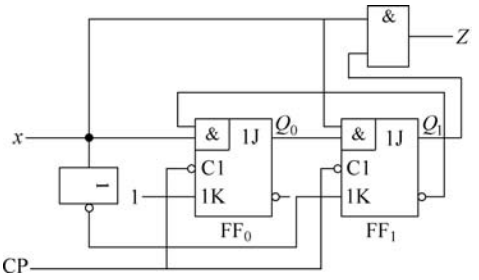


图 5-54　例 5-16 逻辑图

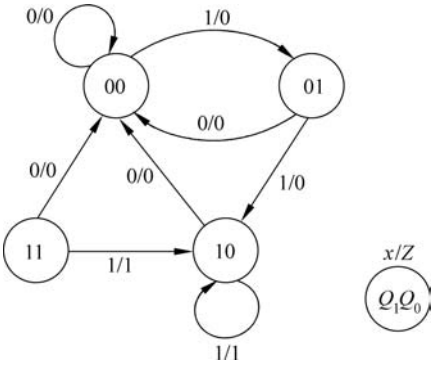


图 5-55　图 5-54 的状态图

本例中若改用 D 触发器,则由于 D 触发器的次态方程为 $Q^{n+1}=D$,即电路的次态方程就是 D 触发器的激励方程。

$$D_1 = xQ_1 + xQ_0 = x\overline{\bar{Q}_1\bar{Q}_0}$$

$$D_0 = x\bar{Q}_1\bar{Q}_0$$

逻辑图如图 5-56 所示。

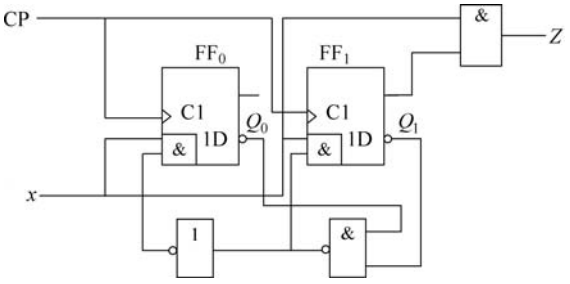


图 5-56　用 D 触发器组成的序列检测器

这个电路用 VHDL 的状态图描述如下:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity sequence_detector is
  port(clk,x,RD: in std_logic;
    Z: out std_logic);
end  sequence_detector;
architecture one of sequence_detector is
type state_type is (A,B,C);                    --用户自己定义的枚举类型
signal state:state_type;                       --信号声明
```

```
begin
  process(clk,RD)                                    -- 状态转移进程,clk,RD 为敏感信号
          begin
            if   RD = '0' then   state < = A;        -- 初始状态为 A
            elsif clk'event and clk = '0' then       -- 当 clk 下降沿到来时执行下面的语句
                  case state is
                       when A = > if   x = '1' then
                                       state < = B;
                                end if;
                       when B = > if   x  = '1' then
                                       state < = C;
                                  else  state < = A;
                                end if;
                       when C = > if   x  = '0' then
                                       state < = A;
                                end if;
                  end case;
            end if;
  end process;
  output_p:process(state)                            -- 输出变化进程,状态为敏感信号
           begin
            case state is
             when C = > if   x = '1' then
                        Z < = '1';                   -- 输出值取决于输入值与现态
                   else  Z < = '0';
                  end if;
             when   others = > Z < = '0';            -- 其余情况输出为零
            end case;
  end process;
end one;
```

该 VHDL 程序运行的仿真图如图 5-57 所示,这是 Mealy 型时序电路,输入变量 X 的变化直接影响到输出变量 Z,而状态的变化会等到时钟有效边沿到来时才发生。还可注意到,此电路是 111…序列检测器,即检测到 3 个及 3 个以上连续的 1 时,输出 $Z=1$,但从时间仿真图中看到第二个脉冲的有效边沿后就出现 $Z=1$,这就是 Mealy 型电路的特点,这时实际上已进入第三个节拍段,也就是第三个 1 已经出现,Z 的变化与输入 X 同步,所以有 $Z=1$。

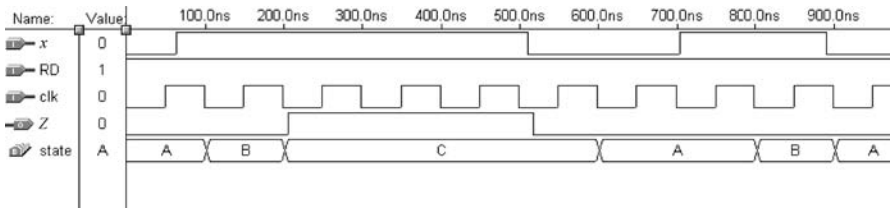


图 5-57　111…序列检测器的时间仿真图

例 5-17　设计一个自动售饮料机的逻辑电路,它的投币口每次只能投入一枚五角或一元的硬币。投入一元五角硬币后,机器会自动给出一杯饮料;投入两元(两枚一元)硬币后,在给出饮料的同时找回一枚五角的硬币。

解：取投币信号为输入逻辑变量，投入一枚一元硬币时用 $A=1$ 表示，未投入时 $A=0$；投入一枚五角硬币用 $B=1$ 表示，未投入时 $B=0$。给出饮料和找钱为两个输出变量，分别以 Y，Z 表示。给出饮料时 $Y=1$，不给时 $Y=0$；找回一枚五角硬币时 $Z=1$，不找时 $Z=0$。

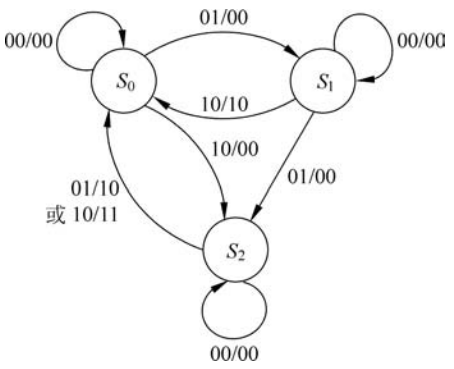


图 5-58　例 5-17 的状态图

假定通过传感器产生的投币信号($A=1$ 或 $B=1$)在电路转入新状态的同时也随之消失，否则将被误认作又一次投币信号。

设未投币前电路的初始状态为 S_0，投入五角硬币以后为 S_1，投入一元硬币(包括投入一枚一元硬币和投入两枚五角硬币的情况)以后为 S_2。再投入一枚五角硬币后电路返回 S_0，同时输出为 $Y=1$，$Z=0$；如果投入的是一枚一元硬币，则电路也应返回 S_0，同时输出为 $Y=1$，$Z=1$。因此，电路的状态数 $M=3$ 已足够。根据以上分析，可得自动售饮料机的逻辑电路的状态图如图 5-58 所示。

根据状态图可得状态表如表 5-33 所示。因为正常工作中不会出现 $AB=11$ 的情况，所以这时次态和输出均为无关项。又因该状态表已为最简形式，所以不必再进行化简过程。

表 5-33　例 5-17 的状态表

状　态	AB			
	00	01	11	10
S_0	$S_0/00$	$S_1/00$	d/dd	$S_2/00$
S_1	$S_1/00$	$S_2/00$	d/dd	$S_0/10$
S_2	$S_2/00$	$S_0/10$	d/dd	$S_0/11$

状态分配。由于状态表中有三个状态，取触发器的位数 $n=2$，即 Q_1Q_0 就满足要求，假如令 $S_0=00$，$S_1=01$，$S_2=10$，$Q_1Q_0=11$ 作无关状态，则得二进制状态表(***Y*-*Z*** 矩阵)如表 5-34 所示。

表 5-34　二进制状态表(*Y*-*Z* 矩阵)

Q_1Q_0	AB			
	00	01	11	10
0　0	00/00	01/00	dd/dd	10/00
0　1	01/00	10/00	dd/dd	00/10
1　1	dd/dd	dd/dd	dd/dd	dd/dd
1　0	10/00	00/10	dd/dd	00/11

若电路选用 D 触发器实现，则刚刚求出的 ***Y*-*Z*** 矩阵中的 ***Y*** 矩阵也就是激励矩阵。这是因为 D 触发器的次态方程为 $Q^{n+1}=D$。根据激励矩阵可得激励和输出的卡诺图如图 5-59 所示。

根据卡诺图可得激励函数和输出函数的表达式为

$$D_1=Q_1\overline{AB}+\overline{Q_1Q_0}A+Q_0B$$

$$D_0=\overline{Q_1Q_0}B+Q_0\overline{A}\,\overline{B}$$

$$Z=Q_1A$$

$$Y=Q_1B+Q_1A+Q_0A$$

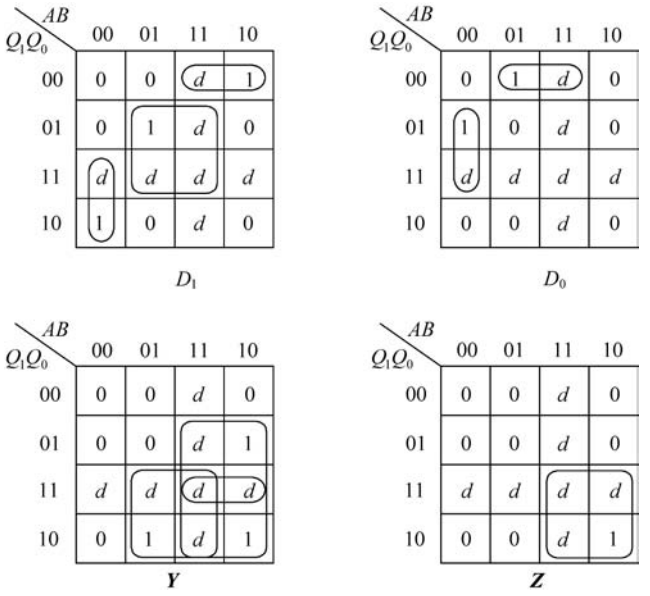


图 5-59　卡诺图

根据激励函数和输出函数可得逻辑图如图 5-60 所示，该逻辑图的实际状态图如图 5-61 所示。实际的状态图画法是将卡诺图化简过程中圈进去的无关项作为 1，没有圈进去的无关项作为 0，就可推出。

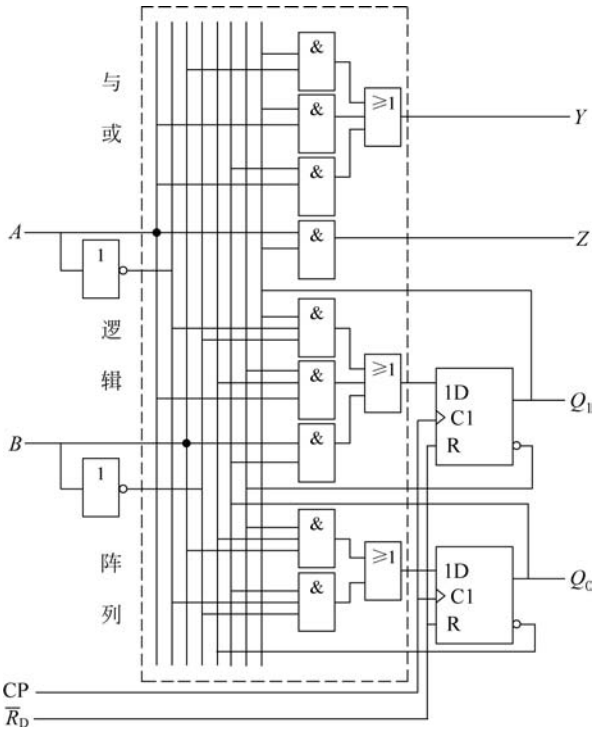


图 5-60　例 5-17 的逻辑图

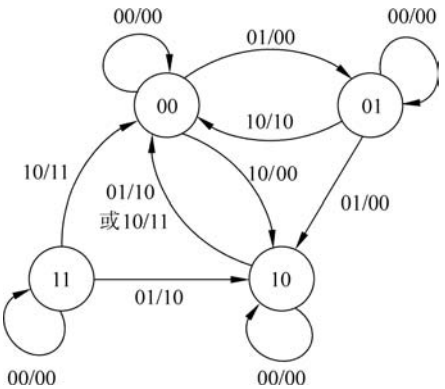


图 5-61　图 5-60 电路的状态图

从图5-61可看出,当电路进入无效状态11以后,在无输入信号的情况下(即$AB=00$)不能自行返回有效循环,所以不能自启动。当$AB=01$或$AB=10$时电路在时钟信号作用下虽然能返回有效循环中去,但收费结果是错误的。因此,在开始工作时应在异步置零端$\overline{R}_D$上加入低电平信号将电路置为00状态。

这个电路用VHDL的状态图描述如下:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity vendor is
  port(clk, A,B,RD: in std_logic;
       Y,Z: out std_logic);
end  vendor;
architecture one of vendor is
type state_type is (s0,s1,s2);                          -- 用户自己定义的枚举类型
signal state:state_type;                                -- 信号声明
begin
  process(clk,RD)                                       -- 状态转移进程,clk,RD为敏感信号
          begin
            if  RD = '0' then  state <= s0;             -- 初始状态为s0
            elsif clk'event and clk = '0' then          -- 当clk下降沿到来时执行下面的语句
                  case state is
                     when s0 => if  A = '1' then
                                    state <= s2;
                                elsif  B = '1'then
                                    state <= s1;
                                end if;
                     when s1 => if  A  = '1' then
                                    state <= s0;
                                elsif  B = '1'then
                                    state <= s2;
                                end if;
                     when s2 => if  A  = '1'or B = '1' then
                                    state <= s0;
                                end if;
                  end case;
            end if;
  end process;
  output_p:process(state)                               -- 输出变化进程,状态为敏感信号
           begin
            case state is
             when s1 => if  A = '1' then
                                Y <= '1';               -- 输出值取决于输入值与现态
                                Z <= '0';
                        end if;
             when s2 => if  A = '1' then
                                Y <= '1';               -- 输出值取决于输入值与现态
                                Z <= '1';
                          elsif  B = '1' then
                                Y <= '1';
```

```
                    Z <= '0';
                end if;
            when others => Y <= '0'; Z <= '0';          -- 其余情况输出为零
        end case;
    end process;
end one;
```

例 5-18　试设计一个带有进位输出端的十三进制计数器。

解：首先进行逻辑抽象。

因为计数器的工作特点是在时钟信号的操作下自动地依次从一个状态转为下一个状态，所以它没有输入逻辑变量，只有进位输出信号。因此，计数器是属于 Moore 型的一种简单时序电路。

取进位信号为输出逻辑变量 C，同时规定有进位输出时 $C=1$，无进位输出时 $C=0$。

十三进制计数器应该有 13 个有效状态，若分别用 S_0，S_1，…，S_{12} 表示，则按题意可以画出如图 5-62 所示的电路状态转换图。

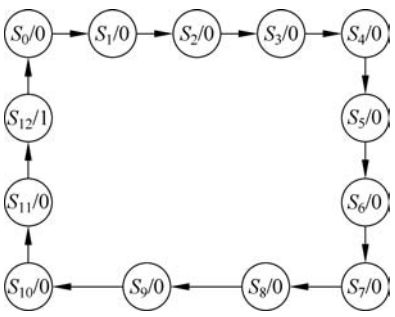


图 5-62　例 5-18 的状态图

因为十三进制计数器必须用 13 个不同的状态表示已经输入的脉冲数，所以状态转换图已不能再化简。根据状态数可知，现要求 $N=13$，故应取触发器位数 $n=4$，因为

$$2^3<13<2^4$$

假如对状态分配无特殊要求，可以取自然二进制数的 0000～1100 作为 S_0～S_{12} 的编码，于是得到了表 5-35 中的状态编码。

表 5-35　例 5-18 电路的状态表

状态变化顺序	状态编码				进位输出 C	等效十进制数
	Q_3	Q_2	Q_1	Q_0		
S_0	0	0	0	0	0	0
S_1	0	0	0	1	0	1
S_2	0	0	1	0	0	2
S_3	0	0	1	1	0	3
S_4	0	1	0	0	0	4
S_5	0	1	0	1	0	5
S_6	0	1	1	0	0	6
S_7	0	1	1	1	0	7
S_8	1	0	0	0	0	8
S_9	1	0	0	1	0	9
S_{10}	1	0	1	0	0	10
S_{11}	1	0	1	1	0	11
S_{12}	1	1	0	0	1	12
S_0	0	0	0	0	0	0

将状态表表示成卡诺图形式,也就是得到的 ***Y*-*Z*** 矩阵(二进制状态表)如表 5-36 所示。卡诺图中不会出现的三种状态 1101,1110 和 1111 作为无关项处理。

表 5-36　例 5-18 的 *Y*-*Z* 矩阵

Q_3Q_2	Q_1Q_0			
	00	01	11	10
0　0	0001/0	0010/0	0100/0	0011/0
0　1	0101/0	0110/0	1000/0	0111/0
1　1	0000/1	*dddd*/*d*	*dddd*/*d*	*dddd*/*d*
1　0	1001/0	1010/0	1100/0	1011/0

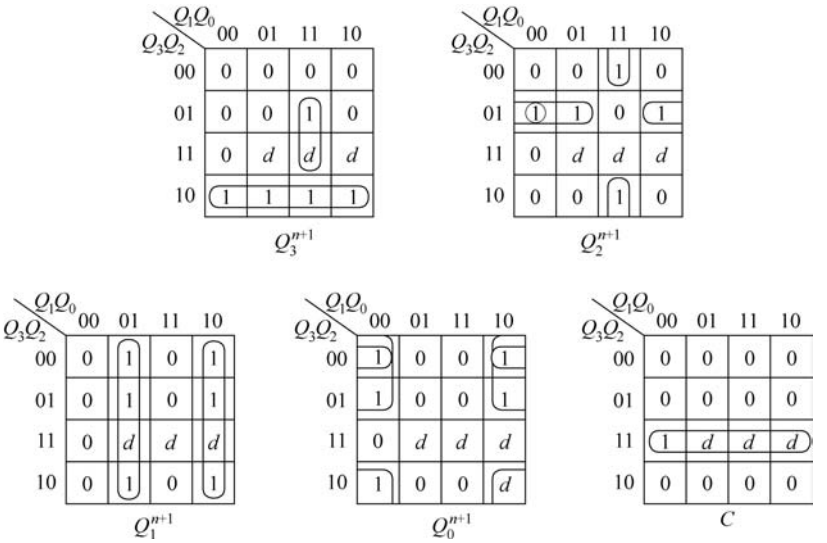


图 5-63　卡诺图

将表 5-36 的 ***Y*-*Z*** 矩阵分解为 5 张卡诺图,如图 5-63 所示,从卡诺图很容易求出电路的次态方程和输出方程为

$$Q_3^{n+1}=Q_3\overline{Q}_2+Q_2Q_1Q_0$$

$$Q_2^{n+1}=\overline{Q}_3Q_2\overline{Q}_1+\overline{Q}_3Q_2\overline{Q}_0+\overline{Q}_2Q_1Q_0$$

$$Q_1^{n+1}=\overline{Q}_1Q_0+Q_1\overline{Q}_0$$

$$Q_0^{n+1}=\overline{Q}_3\overline{Q}_0+\overline{Q}_2\overline{Q}_0$$

$$C=Q_3Q_2$$

如果选用 JK 触发器组成这个电路,则应将以上电路的次态方程变换成 JK 触发器次态方程的标准形式,即 $Q^{n+1}=J\overline{Q}^n+\overline{K}Q^n$,就可以找出激励函数了。为此,将电路次态方程改写为

$$Q_3^{n+1}=Q_3\overline{Q}_2+Q_2Q_1Q_0(Q_3+\overline{Q}_3)=(Q_2Q_1Q_0)\overline{Q}_3+\overline{Q}_2Q_3$$

$$Q_2^{n+1}=(Q_0Q_1)\overline{Q}_2+(\overline{Q}_3\overline{Q_1Q_0})Q_2$$

$$Q_1^{n+1}=Q_0\overline{Q}_1+\overline{Q}_0Q_1$$

$$Q_0^{n+1}=(\overline{Q}_3+\overline{Q}_2)\overline{Q}_0+\overline{1}Q_0=\overline{Q_3Q_2}\overline{Q}_0+\overline{1}Q_0$$

与 JK 触发器的次态方程进行比较得各个触发的激励函数为

$$\begin{cases} J_3 = Q_2Q_1Q_0, & K_3 = Q_2 \\ J_2 = Q_1Q_0, & K_2 = \overline{\overline{Q_3Q_1Q_0}} \\ J_1 = Q_0, & K_1 = Q_0 \\ J_0 = \overline{Q_3Q_2}, & K_0 = 1 \end{cases}$$

根据激励函数和输出函数得逻辑图，如图 5-64 所示。

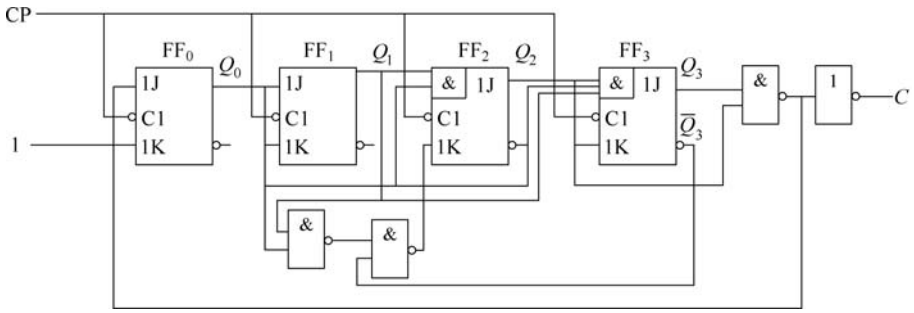


图 5-64　十三进制同步计数器电路

最后还应检查电路能否自启动。将三个无效状态 1101、1110 和 1111 分别代入最后改写过的电路次态方程中计算，所得次态分别为 0010、0010 和 0000，故电路能自启动。图 5-65 就是图 5-64 逻辑电路的实际状态图。

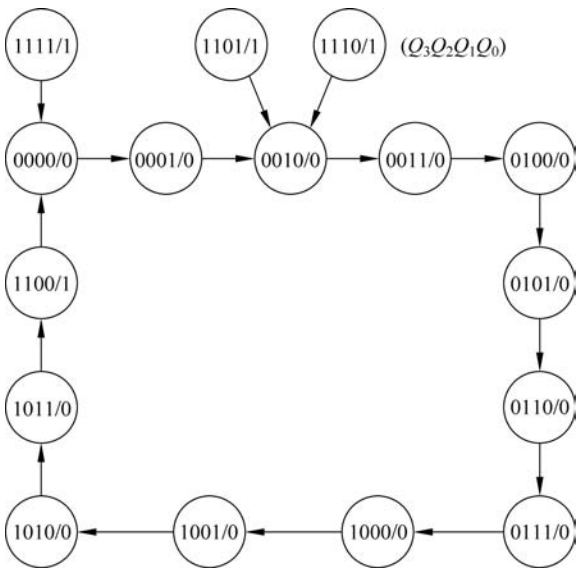


图 5-65　图 5-64 的状态图

该计数器用 VHDL 描述如下：

```
LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;
 USE IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
ENTITY CNT13 IS
  PORT (CLK:IN STD_LOGIC;
        Q:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
        C: OUT STD_LOGIC);
END cnt13;
ARCHITECTURE ONE OF CNT13 IS
BEGIN
  PROCESS(CLK)
    VARIABLE QI:STD_LOGIC_VECTOR(3 DOWNTO 0);
  BEGIN
    IF CLK'EVENT AND CLK = '0' THEN                    -- 检测时钟下降沿
      IF QI <"1100"THEN QI: = QI + 1;                  -- 计数
        ELSE QI: = (OTHERS = >'0');
      END IF;
VEND IF;
    IF QI = "1100" THEN  C < = '1';                    -- 计数等于12,输出进位信号
      ELSE  C < = '0';
    END IF;
      Q < = QI;                                        -- 将计数值向端口输出
  END PROCESS;
END ONE;
```

5.6 异步时序逻辑电路

前面讨论的同步时序电路的特点是电路由统一的时钟触发内部状态的变化。尽管逻辑门、触发器均有延时,但延时之和小于时钟周期,故在下一个时钟脉冲到来前,电路已处于稳定状态。

而异步时序电路没有统一的时钟脉冲,电路状态的改变完全由外部输入信号的变化引起。根据输入信号的不同,异步时序电路又分为脉冲型异步时序电路和电平型异步时序电路。顾名思义,脉冲型异步时序电路的输入包含脉冲信号,而电平异步时序电路的输入仅由电平信号构成。

由于异步时序电路中没有统一的时钟信号,所以分析、设计的方法也与同步时序电路不同。脉冲型、电平型的异步时序电路的分析与设计的方法也不尽相同,由于篇幅限制,下面主要就脉冲型异步时序电路的分析与设计进行讨论。

5.6.1 脉冲异步时序逻辑电路的分析

1. 脉冲异步时序逻辑电路的特点

(1) 与同步时序电路类似,在脉冲异步时序逻辑电路中,记忆部分也是由触发器组成的,但时钟脉冲并不一定送到每位触发器的时钟端。

(2) 输入都以脉冲的形式出现,以0表示没有输入脉冲、1表示有输入脉冲。

(3) 在同一个时刻,只允许一个输入。例如:设 x_1、x_2、x_3 为三个输入,则输入组合000,100,010,001是允许出现的,其他的组合形式不允许出现。其中000表示没有输入,其他依次表示输入 x_1、x_2、x_3。

(4) 在第一个输入脉冲引起的整个电路响应完全结束后，才允许第二个输入脉冲到来，否则电路会出现不可预测的状态。

2. 分析步骤

与同步时序电路分析类似，异步时序电路的分析也是利用状态表、状态图作为工具，分析步骤也类似：

(1) 写出电路的输出函数、激励函数表达式。

(2) 列出电路的次态方程组。

(3) 列出电路次态真值表。

(4) 做出状态表和状态图。

(5) 画出时间图并用文字描述电路的逻辑功能。

3. 分析举例

例 5-19 分析如图 5-66 所示的脉冲异步时序电路。

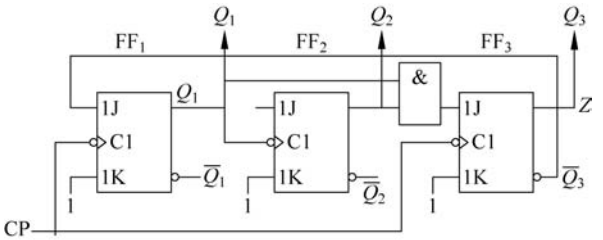


图 5-66 例 5-19 的脉冲异步时序电路

解：从电路图可以看出，这个电路的时钟 CP 仅送到 FF_1、FF_3 的时钟输入端，而没有送到 FF_2 的时钟输入端。FF_2 的时钟输入端接 FF_1 的输出 Q_1，所以该电路是异步脉冲时序电路。另外这个电路没有外部输入，仅仅是一个时钟 CP 的输入，JK 触发器的时钟脉冲为下降沿有效。

第一步，写出电路的激励函数、输出函数。

$$CP_1 = CP_3 = CP;\ CP_2 = Q_1 \quad \text{(异步时序电路的时钟端也是激励)}$$

$$J_1 = \overline{Q}_3;\ K_1 = 1$$

$$J_2 = K_2 = 1$$

$$J_3 = Q_2Q_1;\ K_3 = 1$$

$$Z = Q_3$$

第二步，将激励函数代入触发器的次态方程得到电路的次态方程组。

JK 触发器的次态方程为 $Q^{n+1} = (J\overline{Q} + \overline{K}Q)CP\downarrow$，这里加了 $CP\downarrow$，就是次态的变化发生在有效时钟的作用下，原来分析同步时序电路时，因为每个触发器的变化都是在统一时钟的作用下发生的，故时钟的作用是默认的。异步时序电路的触发器时钟端不是接统一时钟的，所以特地标上时钟有效时发生变化。

$$Q_1^{n+1} = (J_1\overline{Q}_1 + \overline{K}_1Q_1)CP_1\downarrow = (\overline{Q}_3\overline{Q}_1 + \overline{1}Q_1)CP\downarrow = \overline{Q}_3\overline{Q}_1CP\downarrow$$

$$Q_2^{n+1} = (J_2\overline{Q}_2 + \overline{K}_2Q_2)CP_2\downarrow = (1\overline{Q}_2 + \overline{1}Q_2)Q_1\downarrow = \overline{Q}_2Q_1\downarrow$$

$$Q_3^{n+1} = (J_3\overline{Q}_3 + \overline{K}_3Q_3)CP_3\downarrow = (Q_2Q_1\overline{Q}_3 + \overline{1}Q_3)CP\downarrow = Q_2Q_1\overline{Q}_3CP\downarrow$$

第三步,列出电路次态真值表。

列出电路的现态 $Q_3Q_2Q_1$ 的各种组合,并注意到 CP 下降沿的作用,代入上面求出的电路次态方程组中,得到相应时刻的次态和输出。

当初始状态 $Q_3Q_2Q_1=000$ 时。如果 CP=0(没有下降沿),则 $CP_3CP_2(Q_1)CP_1=000$,说明没有时钟输入,触发器不发生变化,所以次态 $Q_3^{n+1}Q_2^{n+1}Q_1^{n+1}=000$;如果 CP=1(出现下降沿),则 $CP_3CP_2(Q_1)CP_1=101$,$Q_3^{n+1}=Q_2Q_1\overline{Q}_3CP\downarrow=0$,$Q_2^{n+1}=\overline{Q}_2Q_1\downarrow=0$,$Q_1^{n+1}=\overline{Q}_3\overline{Q}_1\,CP\downarrow=1$,所以次态 $Q_3^{n+1}Q_2^{n+1}Q_1^{n+1}=001$。

当现态 $Q_3Q_2Q_1=001$ 时,如果 CP=1,这时 $Q_1^{n+1}=\overline{Q}_3\overline{Q}_1CP\downarrow=0$,$Q_1$ 从 1 变为 0,即出现了 $Q_1\downarrow$,那么 $CP_2=Q_1=1$,$Q_2^{n+1}=\overline{Q}_2Q_1\downarrow=1$,$Q_3^{n+1}=Q_2Q_1\overline{Q}_3CP\downarrow=0$,所以次态 $Q_3^{n+1}Q_2^{n+1}Q_1^{n+1}=010$。其余以此类推,就得出表 5-37 的电路次态真值表。

表 5-37　例 5-19 的电路次态真值表

输入脉冲数	Q_3	Q_2	Q_1	CP_3	$CP_2(Q_1)$	CP_1	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Z
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	1	0
2	0	0	1	1	1	1	0	1	0	0
3	0	1	0	1	0	1	0	1	1	0
4	0	1	1	1	1	1	1	1	0	0
5	1	0	0	1	0	1	0	0	0	1
6	1	0	1	1	1	1	0	1	0	1
7	1	1	0	1	0	1	0	1	0	1
8	1	1	1	1	1	1	0	0	0	1

要注意的是:表中 CP=1,表示时钟输入端有下降沿到达;CP=0,表示没有时钟信号到达,触发器保持原来的状态不变。各个触发器的 CP 端信号不是在同一时刻出现的,有先后差错,但在状态变化稳定后,下一轮时钟信号才会再出现。

第四步,做出状态表(表 5-38)和状态图(图 5-67)。

表 5-38　例 5-19 的状态表

Q_3	Q_2	Q_1	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Z
0	0	0	0	0	1	0
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	0	0	0	1

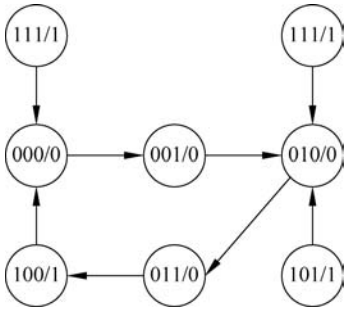


图 5-67　例 5-19 的状态图

第五步,画时间图,如图 5-68 所示。功能分析如下:

从上面画出的状态图、状态表及时间图可以看出,该电路在有效脉冲信号作用下,状态在 000 到 100 这 5 个状态之间进行循环,其他三个状态 101、110、111 在一个脉冲作用后会

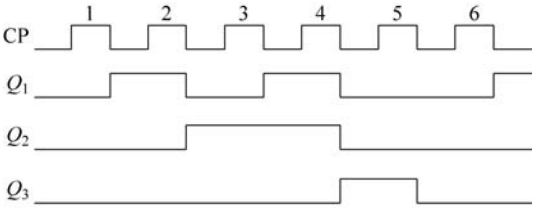


图 5-68　例 5-19 的时间图

自动进入有效循环中。所以该电路为异步五进制计数器。

5.6.2　脉冲异步时序逻辑电路的设计

脉冲异步时序逻辑电路的设计方法与同步时序逻辑电路的设计基本类似。不同的是设计脉冲异步时序逻辑电路时，每个触发器的 CP 端不再是同一个时钟脉冲，而是与其他输入端一样作为触发器的一个激励来考虑。另外，为了使电路工作可靠，输入信号必须是串行序列脉冲，在第二个脉冲达到时，第一个脉冲所引起的电路响应必须已经完成，电路处于稳定状态。也就是前面所说的“在同一个时刻，只允许一个输入”。

由于电路中没有统一时钟，电路中触发器的时钟作为激励来处理，这就意味可以通过控制时钟端的输入脉冲的有无来控制触发器翻转还是不翻转。基于这一思想，在设计脉冲异步时序逻辑电路时，可以使用表 5-39～表 5-42 所列的 4 种常用触发器(带 CP)的激励表。

表 5-39　RS 触发器的激励表(CP)

Q	Q^{n+1}	R	S	CP
0	0	d	0	d
0	0	d	d	0
0	1	0	1	1
1	0	1	0	1
1	1	0	d	d
1	1	d	d	0

表 5-40　JK 触发器的激励表(CP)

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

表 5-41　T 触发器的激励表(CP)

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

表 5-42　D 触发器的激励表(CP)

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

从表 5-39～表 5-42 可以看出，在要求触发器状态保持不变时，有两种不同的处理方式：一是令 CP 为 d，输入端取相应的值；二是令 CP 为 0，输入端取任意值。例如，当要使 D 触发器维持不变时，可令 CP 为 d，D 为 Q；也可令 CP 为 0，D 为 d。这使激励函数的确定更加灵活。一般选择 CP 为 0，输入为任意值。

例 5-20 设计一个脉冲异步时序逻辑检测器。该电路有三个输入 x_1、x_2、x_3,一个输出 Z,当检测到输入脉冲序列为 $x_1 \rightarrow x_2 \rightarrow x_3$ 时,输出 Z 为 1,其后当检测到输入脉冲出现 x_2 时,输出 Z 由 1 变为 0。

解:分析题意,可以得到输入、输出信号的波形关系如图 5-69 所示。

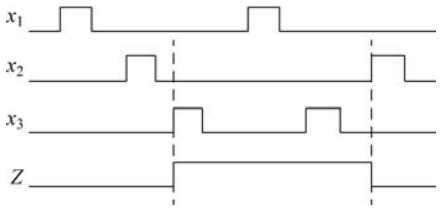


图 5-69 例 5-20 的波形图

首先可以参照同步时序电路设计那样建立电路的原始状态图和状态表。假设 A 为初始态,B 为接收到 x_1 的状态,C 为接收到脉冲序列 $x_1 \rightarrow x_2$ 的状态,D 为接收到脉冲序列 $x_1 \rightarrow x_2 \rightarrow x_3$ 的状态,这样可以得到部分原始状态图,如图 5-70(a)所示,然后再从每个状态出发,做出所有可能输入条件下的状态转换关系,从而建立完成的原始状态图如图 5-70(b)所示。

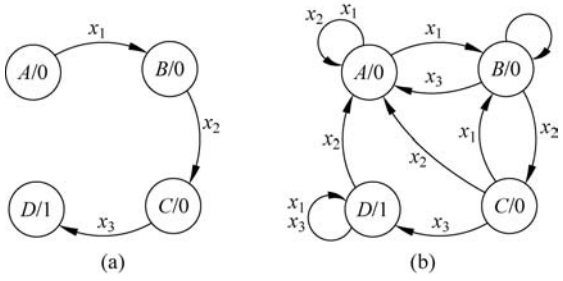


图 5-70 例 5-20 的原始状态图

由图 5-70(b)可以得到原始状态表如表 5-43 所示。

表 5-43 例 5-20 的原始状态表

Q	Q^{n+1}			Z
	x_1	x_2	x_3	
A	B	A	A	0
B	B	C	A	0
C	B	A	D	0
D	D	A	D	1

按照状态表化简规则,该原始状态表已经是最简形式。该电路为 Moore 型,共有 4 个状态,可以两个状态变量 Q_2Q_1 来表示,根据状态分配原则,将 A 分配 00、B 分配 01、C 分配 11、D 分配 10,得到二进制状态表如表 5-44 所示。

如采用 D 触发器来实现,将 CP 看作激励,D 触发器的次态方程可以写成

$$Q^{n+1}=D\mathrm{CP}+Q\overline{\mathrm{CP}}$$

表 5-44　例 5-20 的二进制状态表

Q_2Q_1	$Q_2^{n+1}Q_1^{n+1}$			Z
	x_1	x_2	x_3	
0　0	01	00	00	0
0　1	01	11	00	0
1　1	01	00	10	0
1　0	10	00	10	1

根据表 5-42，在 D 触发器的激励表中，如状态没有变化，可以使 CP＝0，D 为任意，也可以令 $D=Q$，CP 为任意。根据化简的需求，可以灵活地运用，使得电路有最简的结果。将表 5-44 与表 5-42 一起来对照，可得到简化了的激励矩阵，如图 5-71 所示。

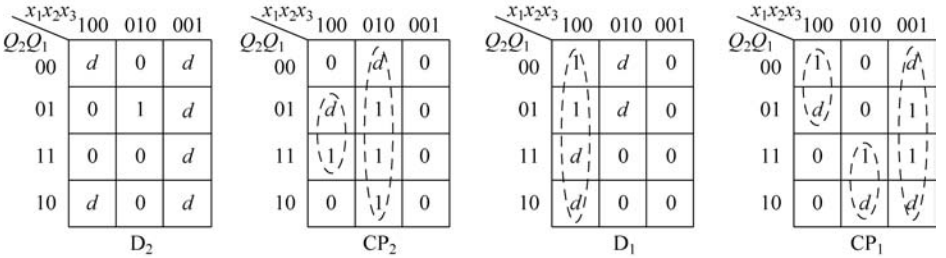


图 5-71　例 5-20 的简化卡诺图

图 5-71 的卡诺图实际上应该是 5 变量的卡诺图，但由于脉冲异步时序电路不允许两个或多个输入脉冲同时出现，即输入的变量组合不允许出现 011、101、110、111，而 000 时，电路保持不变，故可将 5 变量卡若图简化成图 5-71 的形式。但此时卡诺图的各列是不相邻的，化简仅仅是在给定的列中进行，每列只允许一个输入变量出项。经如图 5-71 所示的合并方案得激励函数为

$$D_2=x_2\overline{Q}_2Q_1 \quad \mathrm{CP}_2=x_1Q_1+x_2$$

$$D_1=x_1 \quad \mathrm{CP}_1=x_1\overline{Q}_2+x_2Q_2+x_3$$

由表 5-43 也可得输出函数为

$$Z=Q_2\overline{Q}_1$$

根据激励函数、输出函数表达式可以画出如图 5-72 的逻辑电路图。

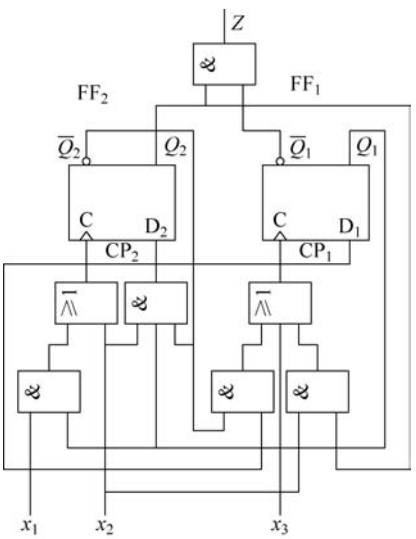


图 5-72　例 5-20 的逻辑电路图

5.7 小　　结

时序逻辑电路的特点是：电路的输出不仅与当前输入有关，还与以前的输入有关。

本章用大量的篇幅讨论了同步时序电路的分析与设计方法。同步时序电路工作稳定、可靠，设计简单，在数字系统中被广泛采用。与组合逻辑电路比较，主要多了存储元件部分。作为存储元件的器件主要有RS触发器、JK触发器、D触发器、T触发器，根据电路中所用触发器类型的特性，利用表格法或代数法可对同步时序电路进行分析和设计。

寄存器、计数器是数字系统中最常用的时序逻辑电路构件，其功能是：在某一时刻将数据并行打入其中进行保存，或通过移位寄存器的移位功能实现数据左移、右移、并入并出、串入并出、并入串出等逻辑功能。

同步时序电路分析的步骤为：①根据逻辑图写出输出函数和激励函数表达式。②根据所用触发器的特性用代数法或表格法求电路的$\boldsymbol{Y}$-$\boldsymbol{Z}$矩阵(二进制状态表)。③根据$\boldsymbol{Y}$-$\boldsymbol{Z}$矩阵得状态表和状态图。④根据状态表和状态图作时间图并用文字描述电路的逻辑功能。

同步时序电路的设计步骤为：①根据给定的逻辑要求作原始状态图和状态表。②状态表化简。③状态分配。④根据选用的触发器特性用表格法或代数法求激励函数和输出函数的表达式。⑤根据激励函数、输出函数表达式画逻辑图。

硬件描述语言VHDL对时序电路的描述与对组合电路的描述有所不同。

而异步时序电路没有统一的时钟脉冲，电路状态的改变完全由外部输入信号的变化引起。根据输入信号的不同，异步时序电路又分为脉冲型异步时序电路和电平型异步时序电路。顾名思义，脉冲型异步时序电路的输入包含脉冲信号，而电平异步时序电路的输入仅由电平信号构成。

异步时序电路的分析、设计的方法与同步时序电路不同。脉冲型、电平型的异步时序电路的分析与设计的方法也不尽相同，此章中简要介绍了脉冲型异步时序电路的分析与设计。

5.8 习题与思考题

1. 简化如表5-45和表5-46所示的状态表。

表5-45　题1表(a)

Q	X	
	0	1
A	E/0	D/1
B	A/1	F/0
C	C/0	A/1
D	B/0	A/1
E	D/1	C/0
F	C/0	D/1
G	H/1	G/1
H	C/1	B/1

表5-46　题1表(b)

Q	X	
	0	1
A	D/d	C/0
B	D/1	E/d
C	d/d	E/1
D	A/0	C/d
E	B/1	C/d

2. 根据状态分配方法，分别对状态表 5-47 和表 5-48 进行状态分配，列出二进制状态表。

表 5-47　题 2 表(a)

Q	X	
	0	1
A	A/0	B/0
B	C/0	B/0
C	D/0	B/0
D	B/1	A/0

表 5-48　题 2 表(b)

Q	X	
	0	1
A	B/0	E/0
B	D/0	A/1
C	D/1	A/0
D	B/1	C/1
E	A/0	A/0

3. 试分析如图 5-73 所示的时序电路的逻辑功能，画出状态表和状态图。

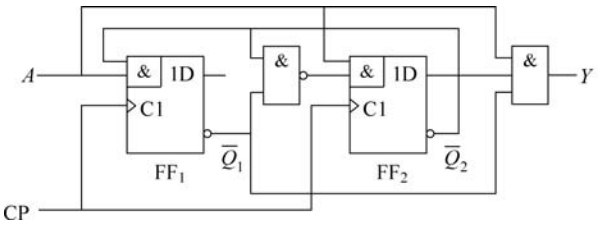


图 5-73　题 3 的图

4. 试分析如图 5-74 所示的时序电路的逻辑功能，画出状态表和状态图，检查电路能否自启动。

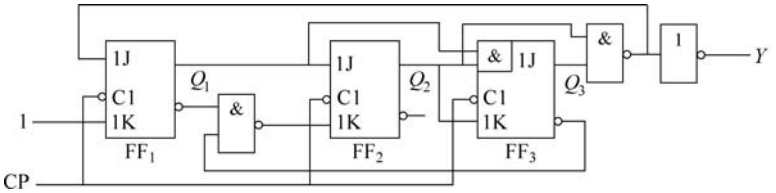


图 5-74　题 4 的图

5. 试分析如图 5-75 所示的时序电路，画出状态表和状态图，检查电路能否自启动，说明电路实现的功能。

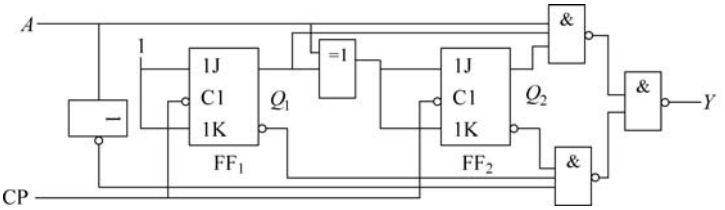


图 5-75　题 5 的图

6. 试分析如图 5-76 所示的时序电路，画出状态表和状态图，并做出当电平输入 x 为 0110101 序列时电路的时间图。

7. 试分析如图 5-77 所示的时序电路,画出状态表和状态图,并做出当电平输入 x 为 0110110 序列时电路的时间图。

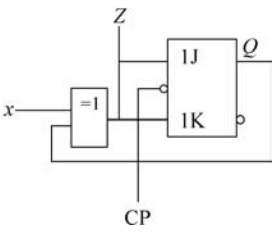


图 5-76　题 6 的图

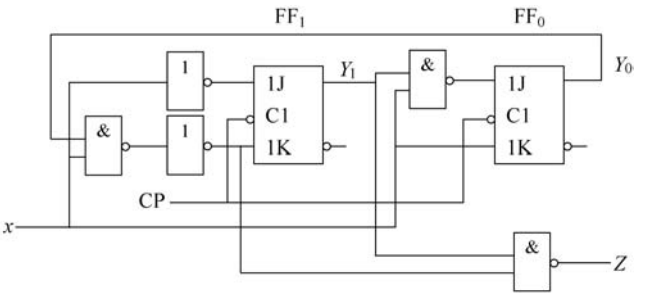


图 5-77　题 7 的图

8. 分析如图 5-78 所示的计数器电路,说明这是多少进制的计数器。十进制计数器 74160 的功能表(同 74LS161 的功能表)见表 5-13。

9. 分析如图 5-79 所示的计数器电路,画出电路的状态图,说明这是多少进制的计数器。十六进制计数器 74LS161 的功能表见表 5-13。

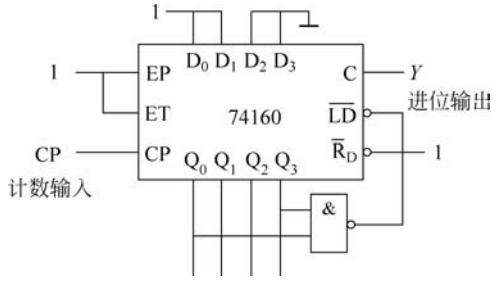


图 5-78　题 8 的图

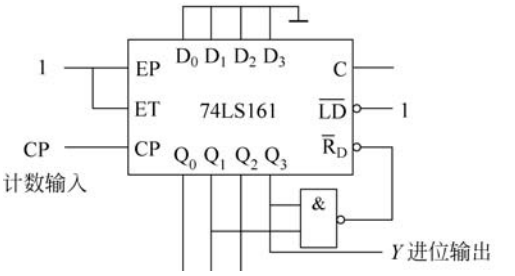


图 5-79　题 9 的图

10. 用 4 位同步二进制计数器 74LS161 接成十二进制计数器,标出输入、输出端。可以附加必要的门电路。74LS161 的功能见表 5-13。

11. 试分析图 5-80 的计数器在 $M=1$ 和 $M=0$ 时各为几进制。74160 的功能见表 5-13。

12. 图 5-81 的电路是可变进制计数器。试分析当控制变量 A 为 1 和 0 时电路各为几进制计数器。74LS161 的功能见表 5-13。

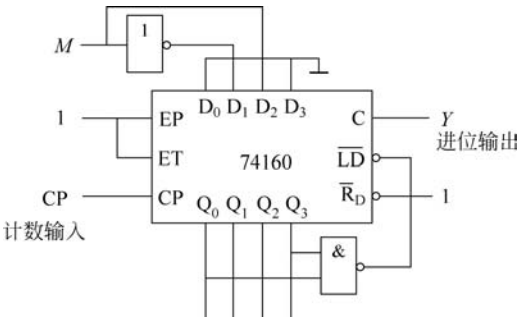


图 5-80　题 11 的图

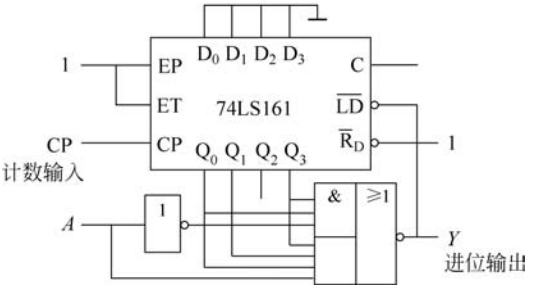


图 5-81　题 12 的图

13. 设计一个可控进制计数器，当输入控制变量 $M=0$ 时工作在五进制，$M=1$ 时工作在十五进制。请标出计数输入端和进位输出端。

14. 试分析图 5-82 计数器电路的分频比（即 Y 与 CP 的频率之比）。74LS161 的功能见表 5-13。

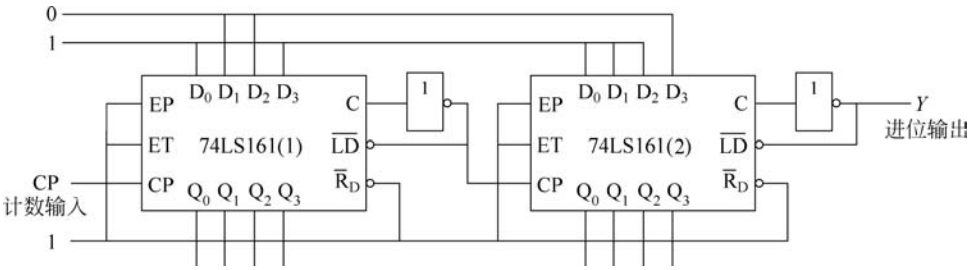


图 5-82　题 14 的图

15. 图 5-83 的电路是由两片同步十进制计数器 74160 组成的计数器。试分析这是多少进制的计数器，两片之间是几进制。74160 的功能见表 5-13。

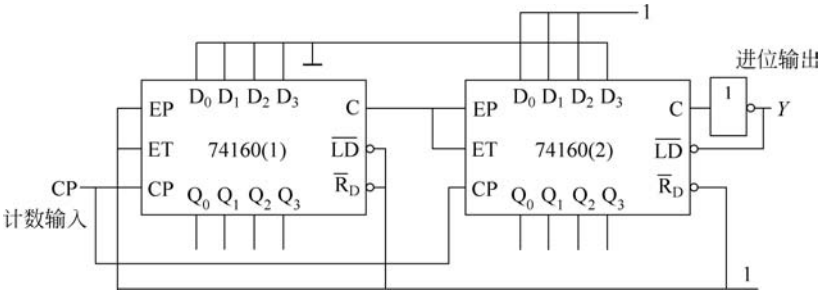


图 5-83　题 15 的图

16. 分析图 5-84 给出的电路，说明这是多少进制的计数器，两片之间是多少进制。74LS161 的功能见表 5-13。

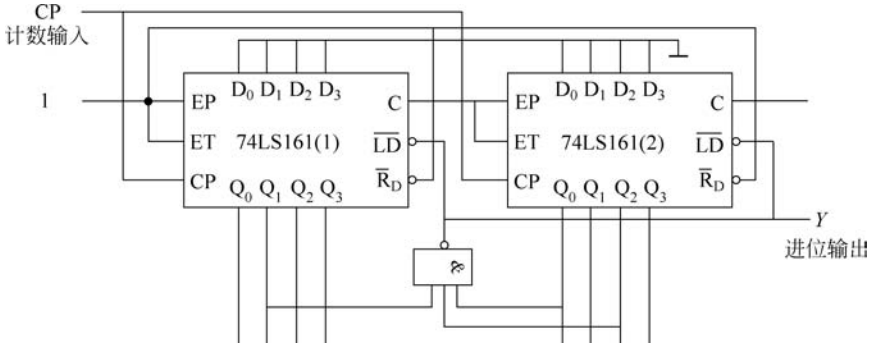


图 5-84　题 16 的图

17. 用同步十进制计数器芯片 74160 设计一个 365 进制的计数器。要求各位间为十进制关系。允许附加必要的门电路。74160 的功能见表 5-13。

18. 作1010序列检测器的状态图、状态表。已知检测器的输入输出序列如下(序列可以重叠)。

输入:0 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0

输出:0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0

19. 设计一个代码检测器,其电路串行输入余3码。当出现非法数字时,电路输出为0,否则输出为1。试做出状态图,并用VHDL描述。

20. 设计一个同步1011序列检测器,序列1011不可重叠,试用JK触发器和适当的门电路实现之,并用VHDL描述。

21. 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　0 1 0 1 0 1 1 0 1

Z　0 0 0 1 0 1 0 0 1

22. 用JK触发器和门电路设计一个4位循环码计数器,它的状态转换表应如表5-49所示。

表5-49　题22的表

计数顺序	电路状态				进位输出 C
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	1	0
3	0	0	1	0	0
4	0	1	1	0	0
5	0	1	1	1	0
6	0	1	0	1	0
7	0	1	0	0	0
8	1	1	0	0	0
9	1	1	0	1	0
10	1	1	1	1	0
11	1	1	1	0	0
12	1	0	1	0	0
13	1	0	1	1	0
14	1	0	0	1	0
15	1	0	0	0	1
16	0	0	0	0	0

23. 设计一个控制步进电动机三相六状态工作的逻辑电路。如果用1表示电机绕组导通,0表示电机绕组截止,则三个绕组 ABC 的状态转换图应如图5-85所示。M 为输入控制变量,当 $M=1$ 时为正转,$M=0$ 时为反转。

24. 设计一个自动售邮票机的逻辑电路,并用VHDL描述出来。每次只允许投入一枚五角或一元的硬币,累计投入两元硬币给出一张邮票。如果投入一元五角硬币以后再投入

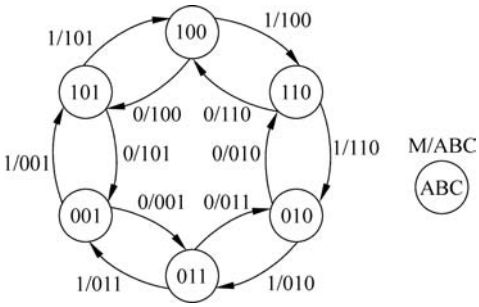


图 5-85　题 23 的图

一枚一元硬币，则给出邮票的同时还应找回五角钱。要求设计的电路能自启动。

25. 请分析以下的 VHDL 描述，说明所定义的各种信号有什么作用，再说明电路完成的是什么功能。

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
 ENTITY  counter  IS
 PORT (clock,clear,count:IN STD_LOGIC;
       q:OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
 END counter;
 ARCHITECTURE one OF counter IS
 SIGNAL pre_q: STD_LOGIC_VECTOR(3 DOWNTO 0);
   BEGIN
    PROCESS(clock,clear,count)
     BEGIN
      IF clear = '1'  THEN
        pre_q <= pre_q - pre_q;
     ELSIF (clock = '1'  AND  clock'EVENT) THEN
       IF count = '1'  THEN
         pre_q <= pre_q + 1;
       END IF;
     END IF;
    END PROCESS;
    q <= pre_q;
END ONE;
```

26. 请分析下面的 VHDL 描述，说明电路完成的是什么功能。

(1)
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
 ENTITY  counter  IS
 PORT (clk,clr_1,ld_1,enp,ent: IN STD_LOGIC;
       d:IN std_logic_vector(3 DOWNTO 0);
       q:OUT std_logic_vector(3 DOWNTO 0);
      rco:OUT STD_LOGIC);
 END counter
 ARCHITECTURE  one OF counter IS
```

```
    SIGNAL  iq: std_logic_vector(3 DOWNTO 0);
      BEGIN
       PROCESS(clk,ent_1,iq)
        BEGIN
         IF clk ' EVENT AND clk  = '1'   THEN
           IF clr_1  = '1'   THEN   iq < = (OTHERS = >'0');
           ELSIF ld_1 = '0'   THEN   iq < = d;
           ELSIF (ent AND enp) =  '1' AND (iq = 9) THEN iq < = ( '0', '0', '0', '0');
           ELSIF (ent AND enp) =  '1'   THEN iq < = iq + 1;
           END IF;
         END IF;
         IF (iq  = 9) AND (ent = '1') THEN rco < = '1';
           ELSE   rco < = '0';
         END IF;
        END PROCESS;
          q < = iq;
END ONE;
```

(2)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY ls160 IS PORT(
   data: in std_logic_vector(3 downto 0);
   clk,ld,p,t,clr:in std_logic;
   count: buffer std_logic_vector(3 downto 0);
   tc:out std_logic);
END ls160;

ARCHITECTURE behavior OF ls160 IS
BEGIN
tc < = '1' when (count = "1001" and p = '1' and t = '1' and ld = '1' and clr = '1') else '0';

cale:
     process(clk,clr,p,t,ld)
     begin
     if(rising_edge(clk)) then
      if(clr = '1')then
          if(ld = '1')then
                if(p = '1')then
                      if(t = '1')then
                            if(count = "1001")then
                                count < = "0000";
                            else
                                count < = count + 1;
                            end if;
                      else
                            count < = count;
                      end if;
                else
                      count < = count;
                end if;
```

```
            else
                  count <= data;
            end if;
      else
            count <= "0000";
      end if;
      end if;
end process cale;
END behavior;
```

(3)
```
library  ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity sequencdcheck is
port
 (
  clk:in std_logic;
  reset:in std_logic;
  din:in std_logic;
  true:out std_logic
 );
end sequencdcheck;

architecture arc of sequencdcheck is
 type state_type is(s1,s2,s3);
 signal state:state_type;
 signal din_d:std_logic;
  begin
 process(clk)
  begin
    if clk'event and clk = '1' then
           din_d <= din;
    end if;
end process;
   _____
process(clk,reset)
 begin
   if reset = '1'then
       true <= '0';
      state <= s1;
 elsif clk'event and clk = '1'  then
   case state is
        when s1 =>
        if din_d = '1' then
             state <= s2;
        else
             state <= s1;
        end if;
        true <= '0';
        when s2 =>
```

```
            if din_d = '0' then
                state < = s3;
            else
                state < = s2;
            end if;
            true < = '0';
            when s3 = >
            if din_d = '1' then
                state < = s1;
                true < = '1';
            else
                state < = s3;
                true < = '0';
            end if;
            when others = >
            state < = state;
        end case;
    end if;
end process;
  ------------
end arc;
```

27. 分析如图 5-86 所示的脉冲异步时序逻辑电路,指出该电路的功能。

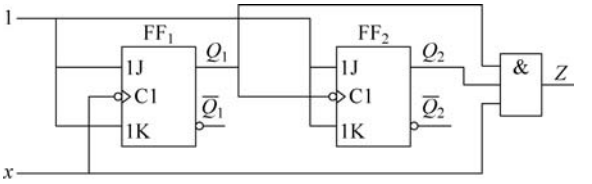


图 5-86　题 27 的图

28. 试用 D 触发器设计 $x_1 \rightarrow x_2 \rightarrow x_2 \rightarrow \cdots$序列检测器。