数字电路课程设计指导

北京理工大学
大连理工大学
沈阳工业大学
大连海运学院 等编

数字系统设计方法

大连理工大学出版社

数字系统设计方法

数字电路课程设计指导

大连理工大学　北京理工大学
大连海运学院　沈阳工业大学　等编

大连理工大学出版社

（辽）新登字16号

主　　编　韩振振　李亚伯
主　　审　张　著

参编人员　唐志宏　恽雪如　程震先
李福连　谢公福　孙广贵
张捷贤　王汉卿　陈顺义
王杏林

数字系统设计方法

Shuzi Xitong Sheji Fangfa

大连理工大学　北京理工大学
大连海运学院　沈阳工业大学　等编

大连理工大学出版社出版发行　　邮政编码：116024
大连理工大学印刷厂印刷
开本：787×1092　1/16　　印张：10 1/8　　字数：244千字
1992年7月第1版　　1992年7月第1次印刷
印数：0001—5000册
责任编辑：郭学满　　封面设计：姜严军
责任校对：刘淑凤
ISBN 7-5611-0650-5/TN·8　　定价：3.00元

前　言

随着电子科学技术的飞速发展，近年来数字电路课程的教学内容正从小规模集成电路过渡到中大规模集成电路，从单元电路过渡到数字系统，其理论基础、分析方法和设计方法都需要有较大的更新、充实和完善。目前“数字系统设计方法”已成为同行学者最关注的课题，而数字电路课程设计又是当前该课程教学改革的重点。为此，我们几所院校的教师，把近年来出现的数字系统的几种描述方法，设计方法介绍给大家，并将几年来所积累的课程设计题目、毕业设计题目、科研题目经过加工整理，一并编入本教材中。

数字系统一般可以划分为控制器和处理器。控制器发出命令信号，处理器执行命令信号所规定的操作，通过操作序列和检验序列完成一个计算任务。本书用硬件程序来描述数字系统的工作、用明细表定义处理器、用状态表定义控制器。为完成控制器的设计，本书介绍了算法状态机（ASM）和备有记忆文件的状态图（MDS）以及寄存器传输逻辑（RTL）。在数字系统的实现中，还介绍了现在仍然广泛采用的试凑法，并以较大篇幅介绍了可编程逻辑器件PAL、GAL和它们在数字系统设计中的应用。

本书以不同的设计方法例举了5个设计课题，从设计任务到系统实现都做了全面、系统的阐述。此外，还选编了13个课题，有的完成了设计全过程，有的只完成到系统级，以便读者发挥个人才智，完成整个设计任务。本书可作为工科电子类大学生数字电路课程设计指导书，也可作为从事电子技术工作的工程技术人员的参考书。

本书由大连理工大学韩振振、李亚伯主编，由北京理工大学张著主审。参加本书编写工作的有大连理工大学唐志宏、北京理工大学恽雪如、程震先，大连海运学院李福连、谢公福、沈阳工业大学孙广贵、张捷贤，锦州工学院王汉卿，抚顺石油学院陈顺义和哈尔滨船舶工程学院王杏林。在编写过程中，谢劼、杨齐祥和郭争光做了有益的工作。

由于编者水平有限，书中会有缺点、错误，恳请读者批评指正。

编　者

1992.5.22 于大连

目　录

第一章　数字系统设计

§1.1　数字系统划分

人们常常把数字系统划分成两个部分——数据处理器和控制器(见图1.1)。

数据处理器由寄存器和组合电路组成。寄存器用于暂存信息。组合电路实现对数据的加工和处理。在一个计算步骤，控制器发出命令信号给数据处理器，数据处理器完成命令信号所规定的操作。在下一个计算步骤，控制器发出另外一组命令信号，命令数据处理器完成相应的操作。通过多步操作(操作序列)，数字系统完成一个计算任务，控制器接收数据处理器的状态信息及外输入选择下一个计算步骤。

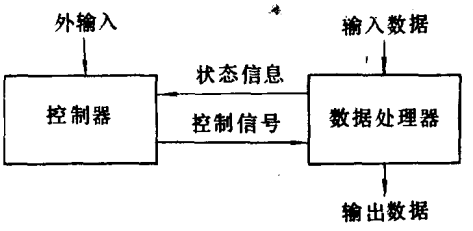


图1.1　数字系统结构

1.1.1　数据处理器表示法

数据处理器的结构示于图1.2. 它由两个部分组成：

1．实现算术、逻辑操作的组合电路；

2．暂存信息的寄存器。

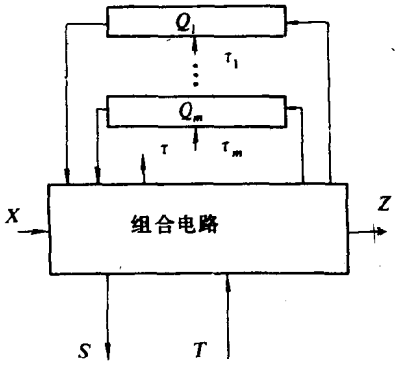


图1.2　数据处理器结构

输入信号 $X(X_1, X_2, \cdots, X_k)$和输出信号 $Z(Z_1, Z_2, \cdots, Z_n)$ 表示通过数据处理器的数据。控制信号 $T(T_1, T_2, \cdots, T_r)$ 是控制器发出的命令信号，决定在时钟脉冲出现时，数据处理器完成什么操作。输出信号 $\tau(\tau_1, \tau_2, \cdots, \tau_m)$ 是由控制信号 T 形成的，加在寄存器的功能控制端，实现寄存器操作的功能选择。状态信号 $S(S_1, S_2, \cdots, S_u)$ 是数据处理器产生的信号，它反馈给控制器，决定下一个步骤。它是输入信号及寄存器内容的函数。

1.1.2　数据处理器的信号关系

一、寄存器次态

设所有寄存器都是由边沿触发器构成的。时钟脉冲有效边沿作用之前的寄存器状态称为寄存器的现状态。信号 τ 作用到寄存器的功能控制端，选择寄存器及指定的寄存器操作。在

时钟有效边沿到来时，寄存器接收信息改变状态。这时寄存器的状态叫寄存器次态。其表达式为：

$$Q \leftarrow F(X, Q, T) \tag{1-1}$$

该式表明，寄存器的次态同以下三个因素有关：

1．输入信号X；

2．寄存器现态Q；

3．控制信号T．

二、数据处理器输出

数据处理器输出信号Z的表达式为

$$Z = G(X, Q, T) \tag{1-2}$$

它是输入信号X，寄存器的现态Q和控制信号T的函数，同状态信号S无关。在有些情况下，输出信号是寄存器的现态值。这时，输出表达式为

$$Z = G(Q) \tag{1-3}$$

三、状态变量

除了最简单的计算之外，操作序列都不是固定的，控制器要根据数据处理器发出的状态变量S及外输入X，选择下一个计算步骤。状态变量S是通过对被处理信息的检验产生的，反应被处理信息的状态。其关系式为

$$S = R(X, Q) \tag{1-4}$$

由此可见：

1．状态变量S与控制信号T无关；

2．状态变量S与寄存器状态和外输入X有关；

3．控制器根据其现态值与状态变量S的现行值确定控制信号的现行值。

四、控制信号T

控制器的输出信号T决定：

1．数据处理器实现的信息传输，即决定哪些寄存器实现什么操作；

2．数据处理器的输出信号。

为实现一个计算任务，必须经过多步操作（操作序列），所以控制信号应是一组信号。例如，$T_1, T_2, \cdots, T_r$，其中每一个元素表示不同的信息处理任务。常常用助记符号表示控制信号，例如：CLR、ADD、INC、DEC等等。其中

CLR —— 寄存器清“0”；

ADD —— 两个寄存器内容相加；

INC —— 寄存器内容加1；

DEC —— 寄存器内容减1．

采用助记符号既便于了解数据处理器应实现的操作，又便于建立控制器的硬件程序。

1.1.3 数据处理器明细表

数据处理器的硬件组成由数据处理任务决定。规定数据处理任务的表格叫明细表。明细表包含两个子表，一个是操作表，另一个是状态变量表。操作表列出在控制信号作用下，数

据处理器应实现的操作和产生的输出；状态变量表定义数据处理器输出的状态变量。

设一个简单的数据处理器如图 1.3 所示。它有单一输入信号 X 及控制信号 T_0、T_1、T_2 和 T_3（分别记之为 NOP、ADDA、ADDB和CLAB），含有两个寄存器 A 和 B，输出状态信号 S_1 和 S_2 及信号 Z，其明细表示于表 1.1.

控制信号 T 产生寄存器功能选择信号 τ．形成 τ 的逻辑网络由寄存器的功能表和应实现的操作决定。

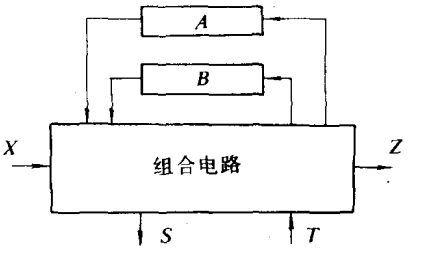


图 1.3　简单的数据处理器

表 1.1　明　细　表

操　作　表		状 态 变 量 表	
控制信号	操　作	状态变量	定　义
NOP　(T_0)	不操作	S_1	$X>0$
ADDA　(T_1)	$A\leftarrow A+X$	S_2	$X<0$
ADDB　(T_2)	$B\leftarrow B+X$	输出	
CLAB　(T_3)	$A\leftarrow 0$	$Z=A$	
	$B\leftarrow 0$		

1.1.4　控制器模型

实现一个计算任务，必存在一个算法，即通过一个有序的操作序列和检验序列完成计算任务。数据处理器负责对数据的操作和检验。控制器规定算法的步骤。控制器在每一个计算步骤给数据处理器发出命令信号，同时接收来自数据处理器的状态变量，确定下一个计算步骤，以确保算法按正确的次序实现，所以控制器决定数据处理器的操作及操作序列。控制器决定算法步骤，必须有记忆能力，所以它是一个时序电路，应包含存贮器。存贮器记忆控制器处在哪一个计算步骤，该发出什么样的命令信号，其框图示于图1.4.

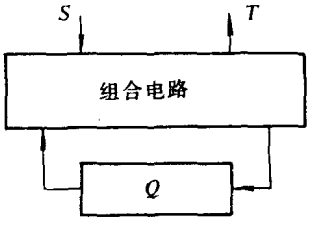


图 1.4　控制器模型

存贮器的状态（即控制器的状态）同计算步骤相对应。在一个状态下，控制器根据收到的状态变量和外输入产生控制信号 T．在时钟到来时，控制器转换到下一个状态（为了简便，今后把外输入看做是状态变量的元素，不再单独考虑）。

根据图1.4，控制器的特性由下面两个方程式决定：

控制信号方程　　$T=F(S,Q)$　　(1-5)

次态方程　　$Q\leftarrow G(S,Q)$　　(1-6)

描述时序电路工作的是状态转换表。控制器的状态转换表示于表 1.2，表格的行表示现

状态（Q_J），表格的列表示状态变量的取值（S_I）。第 J 行，第 I 列的内容为

$$G(S_I, Q_J)/F(S_I, Q_J) \qquad (1-7)$$

它表示控制器的次态是 $G(S_I, Q_J)$，控制信号 T 之值是 $F(S_I, Q_J)$。

表 1.2 控制器状态转换表

Q \ S	S_A	…	S_I	…	S_V
Q_A					
⋮					
Q_J			$G(S_I, Q_J)$ / $F(S_I, Q_J)$		
⋮					
Q_W		…		…	

已知数据处理器的明细表和控制器的状态转换表，就可以根据输入信号确定一个计算任务的完成过程，现举例说明。

已知一个数字网络，其数据处理器包含两个寄存器 A 和 N，输入信号为 X．数据处理器明细表示于表1.3，控制器的状态转换表示于表 1.4．该系统通过将 X 加到 A 来累计 X 之值。当 $A+X \geq 10$ 时，X 与 A 最后一次求和，试求 X、A 相加的次数。

表 1.3 明 细 表

操作表			状态变量表	
控制信号	操 作	输 出	状态变量	定 义
CLR	$N \leftarrow 0$	$Z = N$	S	$A+X \geq 10$
	$A \leftarrow 0$			
ADD	$A \leftarrow A+X$			
	$N \leftarrow N+1$			
OUT	不操作	$Z = N$		

表 1.4 状态转换表

Q \ S	0	1
Q_A	Q_A / ADD	Q_B / ADD
Q_B	ϕ	Q_C / OUT
Q_C	ϕ	Q_A / CLR

设输入数据序列为 5、4 和 3．控制器初始状态为 Q_A．在初始状态，寄存器 A 和 N 均被清“0”．根据系统明细表和状态转换表，便可导出整个计算过程（见表 1.5）。输入 X 之值决定经过几个步骤完成计算任务。

此例说明，一旦知道控制器状态转换表和数据处理器明细表，就可以很容易地描述数字系统的工作过程。以后我们将研究怎样根据算法列出数据处理器的明细表和控制器的状态转换表。

表 1.5 计 算 步 骤

步　　骤	1	2	3	4	5
输 入 X	5	4	3	ϕ	ϕ
状态变量 S	0	0	1	1	1
状　　态	Q_A	Q_A	Q_A	Q_B	Q_C
控制信号	ADD	ADD	ADD	OUT	CLR
输 出 Z	ϕ	ϕ	ϕ	3	3

§1.2 系 统 定 时

为了简便起见，这里假设：

1．数字系统是同步系统，控制器的状态变化以及数据处理器中的寄存器传输操作，都由系统时钟控制。

2．所有输入信号都与时钟脉冲同步。如存在异步信号，应将它转换成同步信号。

3．时钟脉冲同时到达所有存贮元件的时钟输入端。

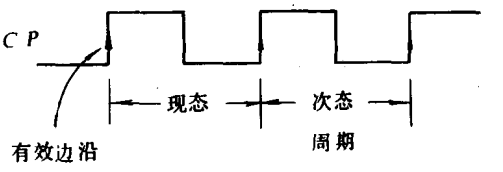


图 1.5 系统时钟波形

时钟脉冲多为周期性电压方波（见图 1.5）。由于中规模集成电路的寄存器和计数器多半是上升边触发，所以，数字系统中，时钟脉冲的有效边沿多为上升边。通常，时钟周期从 20 纳秒到几十微秒。在系统调试阶段，可进行单步调试。

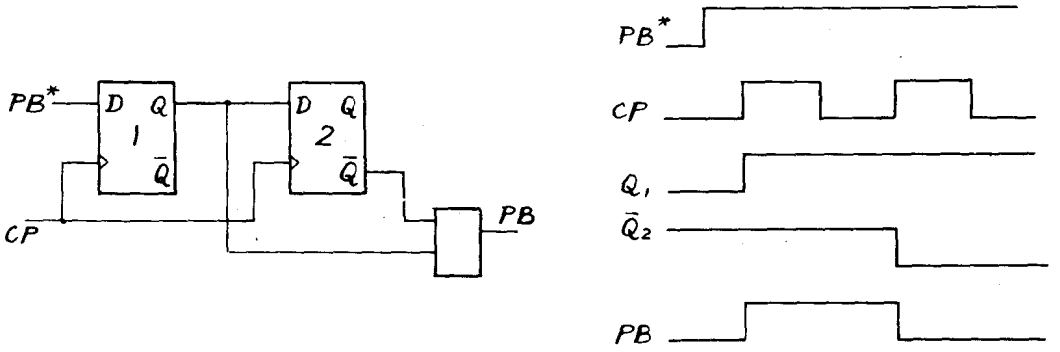
1.2.1 最小时钟周期

时钟脉冲有效边沿到达之前，与寄存器传输操作有关的信号均应达到稳定值，这样才能正确实现寄存器传输操作。

在同步系统中，时钟脉冲有效边沿出现之后，寄存器内容更新，同时输入信号变化（同步输入）。这两组信号用来形成状态变量 S（式 1-4）。S 达到稳定后，控制器根据状态变量和控制器状态形成控制信号 T（式 1-5）。T 稳定之后，才能建立稳定的寄存器功能选择信号 τ 和电路的输出信号 Z（式 1-2）。这时，下一个时钟脉冲有效边沿才允许出现。时钟的最小周期由这段时间间隔决定。

1.2.2 异步信号转换成同步信号的电路

系统之外的信号是随机的，与系统时钟无关。为保证系统正常工作，应将异步信号转换成同步信号。设开关或按钮信号（已消振）PB^* 是异步信号，其中用 * 表示异步信号。利用图 1.6 所示电路，可将异步信号 PB^* 转换成同步信号 PB（单脉冲）。



（a）　　（b）

图 1.6　单脉冲电路

（a）电路；（b）波形图

§1.3　硬件编程语言

控制器对数据处理器的控制过程可以写成硬件程序，即硬件描述语言。我们把硬件程序作为从算法转换到状态转换表的一个中间步骤。通过对硬件编程语言的学习，也可加深对计算、检验步骤的理解。控制器处在一个状态，即一个计算步骤，向数据处理器发出命令信号，命令数据处理器完成相应的操作，并根据数据处理器返回的状态变量，决定控制器的下一个状态，即下一个计算步骤。既然控制器的工作可用硬件描述语言表示，那么控制器的状态转换表的内容应同语言的内容相一致。程序中的语句对应控制器的状态。语句既包含控制器发出的命令信号，还包含根据状态变量决定的次态信息。

1.3.1　简单语句

硬件程序像任何程序一样，是由语句序列组成的。为了编写这些语句，必须建立正式的词法规律和语句习惯。这些规律和习惯应尽可能地简单易懂，并同状态转换表密切相关。

程序中可能有许多语句，常常必须访问其中的某个语句，为此应对语句做标记。标记是语句的一个成分。它由一个到七个字母和数字组成，第一个符号是字母。例如：A1、START、FINISH 等等。泛指一个标记时，用角形括号将标记的元素括上，即〈标记〉。

控制器的输入是外输入和状态变量。为了简便起见，今后把外输入看做是状态变量的元素，不再单独考虑。状态变量可能是多位变量 $S_1, S_2, \cdots, S_n$，它的取值为 $S_A, S_B, \cdots, S_L$，也可用助记符号表示它们。例如，状态变量 S 包含三个变量 S_1、S_2 和 S_3，其中

$$S_1 - X>0, \quad S_2 - X=0, \quad S_3 - X<0$$

即 $X>0$ 时，$S_1=1$；$X=0$ 时，$S_2=1$；$X<0$ 时，$S_3=1$．按照这个规定，助记符号所表示的状态变量的组合（$S_1\ S_2\ S_3$）为

XGT=(100)— $X>0$，　　XGEQ=(110)— $X\geq 0$，

XLT=(001)— $X<0$，　　XNEQ=(101)— $X\neq 0$，

XEQ=(010)— $X=0$

在编程中，是否使用助记符号由编程者自己决定。

控制器的输出信号T命令数据处理器执行特定的操作。常常用助记符号表示T，例如：NOP、CLR、ADD和SUB等等，其目的是便于记忆或联想数据处理器执行的操作。

任何编程语言都在程序的适当点加入注释，对语句做附加说明。〈注释〉为文字说明，不起程序作用。它紧跟在符号/*之后，例如：

```
/* THIS A COMMENT
/* COMPUTE THE MAGNITUDE OF X
```

〈注释〉也可用中文书写，它只起解释和说明的作用，便于编程者记忆程序和语句的作用。

状态转换表规定控制器的次态和输出，所以一个语句亦应提供下列信息：

1．实现这个语句应完成的操作；

2．确定被实现的下一个语句。

一个简单语句由下列成分构成：

〈标记〉　〈操作〉　〈下一个语句〉　〈注释〉

语句的第一个部分是语句的符号名称。标记部分的定义是

〈标记〉::=〈标记〉|空项

"::="指明左侧的量由右侧的量定义。上式意味着，〈标记〉不是〈标记〉就是空项（无标记）。符号|指明它两侧的量中的任何一个都可表示〈标记〉。

语句的第二部分是操作部分，其定义为

〈操作〉::=〈控制信号〉

这表明，操作部分是用助记符号表示的控制信号。

语句的第三部分是下一个语句。它提供的是控制器的次态信息，其定义为

〈下一个语句〉::=〈标记〉|空项

这是语句的地址部分。如果〈下一个语句〉用〈标记〉表示，那么下一个要执行的是由〈标记〉命名的语句；如果是空项，那么下一个要执行的是语句序列中的下一个语句。

简单语句的最后一部分是注释部分，其定义为

〈注释〉::=〈注释〉|空项

任何一个语句都可能有一个附加的注释，用来表明程序中语句的目的或者设计者附加的说明。现举例说明怎样利用简单语句描述一个完整的计算任务。

```
/*这是重复加三组输入数据并显示结果的程序
SIGAD.   CLR        /*清"0"累加寄存器
         ADD        /* 加第一组数据
         ADD        /* 加第二组数据
         ADD        /* 加第三组数据
         DISP SIGAD /*显示结果，重复计算
```

此例中，第一个语句的标记提供了该程序段的名称和地址。接下去的三个语句按次序实现数

表 1.6 状态转换表

状　态	次态／输出
Q_A	Q_B／CLR
Q_B	Q_C／ADD
Q_C	Q_D／ADD
Q_D	Q_E／ADD
Q_E	Q_A／DISP

据求和。最后一个语句显示计算结果，然后返回到第一个语句，所以在该语句的下一个语句位置上写有第一个语句的语句标号，即SIGAD是要执行的下一个语句。

为将这个程序转换成控制器的状态转换表，应找出语句同状态的对应关系。这里，每一个语句对应一个状态。SIGAD语句对应状态Q_A，其它语句分别对应状态Q_B、Q_C、Q_D和Q_E．与这个程序对应的状态转换表示于表1.6．由此表可看出，状态变化的次序是Q_A，Q_B直到Q_E．Q_E相当于语句

DISP　　SIGAD

由此语句返回到指令序列的起点，并重复计算。

1.3.2 复合语句

写硬件程序时，常把一个语句序列看做是一个单个语句，这样的语句序列称为复合语句，其定义为

〈复合语句〉::={〈语句〉
⋮
〈语句〉}

定义指明，〈复合语句〉是由许多语句构成的，“{”表示复合语句的起点，“}”表示语句的终点。复合语句中的语句也可能是条件语句或是复合语句。它们是按次序执行的，只是最后一个语句，在下一个语句位置写有语句标记。例如：

```
OUT:{DISP              /* 显示结果
    {INC A             /* 复合语句中的复合语句
     INC B
     INC C}
     DEC
     CLR NEW}          /* 下一个语句是NEW
```

1.3.3 条件语句

条件语句的定义为

〈条件语句〉::=〈标记〉　IF　〈状态变量
{
〔S_A〕〈语句〉
⋮　　⋮
〔S_K〕〈语句〉
DEFAULT〈语句〉
}

大括号内的所有语句都是条件语句主体内的语句。如状态变量取值S_A，那么执行〔S_A〕语句；

如果状态变量的取值不在 $S_A \cdots S_K$ 之内，那么执行DEFAULT语句。如语句中无DEFAULT值，那么条件语句是不操作语句，下一个要执行的语句是跟随条件语句之后的语句。

例1

```
CHECK: IF  S
       {
       〔XGT〕   ADD  B4
       〔XLT〕   {CMA
                 INC  B2}
       〔XEQ〕   NOP
       }
```

此例中，如 $X>0$，下一个语句是 B_4；$X<0$，执行复合语句，离开复合语句主体时，下一个语句是 B_2；$X=0$ 时，下一个语句是条件语句主体之后的语句。

例2

```
TESTA: IF  S1
       {
       〔0〕  INC A
       〔1〕 {ADD A
             ADD B
             IF  S2
             {
             〔0〕  NOP  OUT 1
             〔1〕  INC B  TEST A
             }
             }
       }
```

此例中，如 $S_1=0$，执行操作INCA，下一个语句是主体语句之后的语句；如 $S_1=1$，执行复合语句。复合语句中包含条件语句。在复合语句的条件语句中，若 $S_2=0$，不操作，下一个语句是OUT1；若 $S_2=1$，执行操作INC B，下一个语句是TEST A.

下面举例说明，怎样将包含条件语句的硬件程序写成状态转换表。已知硬件程序如下：

```
START : CLR
        ADD
    A1: IF  S
        {
        〔XLT〕 SUB OUT
        〔XEQ〕 INC  A1
        〔XGT〕 ADD
        }
        ADD
  OUT : DEC  A1
```

其状态转换表示于表1.7. 表中列出 $X>0$、$X<0$ 和 $X=0$ 三种情况。

表 1.7 状 态 转 换 表

语句标记	S / 状态	XLT	XEQ	XGT
START	Q_A	Q_B/CLR	Q_B/CLR	Q_B/CLR
	Q_B	Q_C/ADD	Q_C/ADD	Q_C/ADD
A1	Q_C	Q_E/SUB	Q_C/INC	Q_D/ADD
	Q_D	Q_E/ADD	Q_E/ADD	Q_E/ADD
OUT	Q_E	Q_C/DEC	Q_C/DEC	Q_C/DEC

当程序中的条件语句包含复合语句时，由程序导出状态转换表的工作就复杂一些了。例如：

```
START : CLR
  NEW : IF S                          /* 第一个条件语句
        {
        〔XLT〕 {ADD                   /* 第一个复合语句
                INC START}
        DEFAULT{SUB                   /* 第二个复合语句
                INC
                SUB}
        }
        IF S                          /* 第二个条件语句
        {
        〔XGT〕  ADD NEW
        }
        ADD START                     /* 重复计算
```

程序中有两个条件语句。在第一个条件语句中，$X<0$ 时，执行第一个复合语句；否则执行另一个复合语句。在第二个条件语句中，$X>0$ 时，执行一个语句，否则不操作。与此程序对应的状态转换表示于表 1.8.

表 1.8 状 态 转 换 表

语句标记	状态变量 / 状态	XLT	XEQ	XGT
START	Q_A	Q_B /CLR	Q_B /CLR	Q_B /CLR
NEW	Q_B	Q_{B1}/ADD	Q_{B2}/SUB	Q_{B2}/SUB
	Q_{B1}	Q_A/INC	Q_A /INC	Q_A /INC
	Q_{B2}	Q_{B3}/INC	Q_{B3} /INC	Q_{B3}/INC
	Q_{B3}	Q_C/SUB	Q_C /SUB	Q_C /SUB
	Q_C	Q_D/NOP	Q_D /NOP	Q_B /ADD
	Q_D	Q_A/ADD	Q_A /ADD	Q_A /ADD

硬件程序描述控制器的工作是很方便的，但是为了用逻辑电路实现控制器应使用状态转换表。下面通过例题，熟悉建立硬件程序和状态转换表的方法。

1.3.4 流程图

常常用流程图表示算法，流程图使用四种符号。

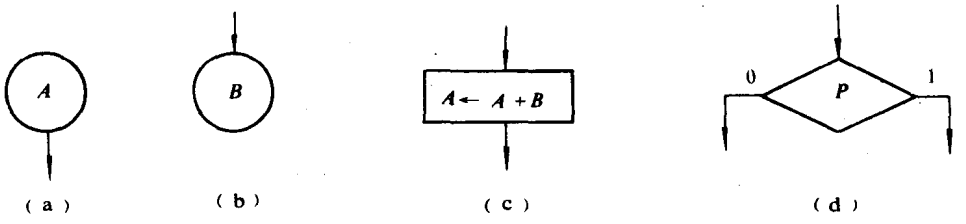


图 1.7 流程图符号

（a）入口点；（b）出口点；（c）传输框；（d）判断框

入口点示于图1.7（a），指明算法的起点或算法的继续点。当算法太长，一页写不完另起一页时，就需要一个继续点。有入口点，就应有出口点(见图1.7（b))。入口点和出口点的应用举例示于图1.8. 通过点 A_2，算法从一页转入另一页。通过点 A_1，算法从一页返回另一页。

传输框(图1.7（c)) 指明，在算法的特定点要完成的寄存器传输操作，其中用RTL符号表示法表示寄存器传输操作。

判断框（图1.7（d)) 中的变量 P 是被检验的状态变量。条件满足时，选择一个引出分支，否则选定另一个引出分支。

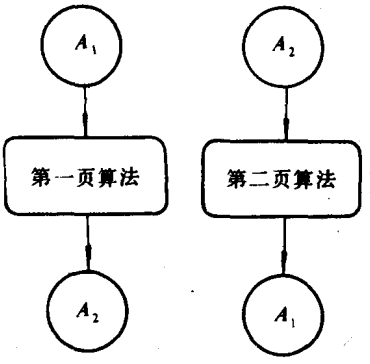


图 1.8 入口点和出口点应用举例

流程图的画法是按照事件发生的先后次序排列的，与电路状态无明显的对应关系。

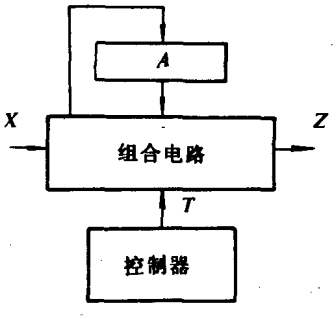


图 1.9 数字系统框图

1.3.5 例题

例 1：函数求值。

任务：求函数 $Z=4*X1+2*X2+X3$ 的值。

输出：计算完成时，提供输出 Z。

解：系统中应有一个寄存器存放计算结果。系统框图示于图1.9。函数求值使用两种运算，乘2和求和。左移