第3章 逻辑电路基础

逻辑电路是设计计算机硬件的基础。逻辑代数是分析和设计逻辑电路的基本工具。构成逻辑电路的基本单元是逻辑门电路和触发器，逻辑电路包括组合逻辑电路和时序逻辑电路。计算机硬件中常用的组合逻辑电路有三态门、加法器、译码器、数据选择器等，常用的时序逻辑电路有计数器、锁存器、数据寄存器和移位寄存器等。

3.1 逻辑代数基础

逻辑代数是研究逻辑变量及其相互关系的一门科学，是英国数学家乔治·布尔(Geoge Boole)于1849年提出来的，因此也称为**布尔代数**。逻辑代数是分析和设计数字电路的基础。

3.1.1 逻辑常量、逻辑变量

逻辑代数是二值代数，只有两个常量，分别用0和1表示，这里的0和1不表示数值的大小，而是表示对立的两种状态，如开关的断开和闭合、信号的有和无、电平的高和低。在数字系统中，用逻辑电平来表示0和1。逻辑电平指的是一定的电压范围，有高电平(H)和低电平(L)之分，高电平对应一种状态，而低电平则对应另一种不同的状态。正逻辑和负逻辑是对逻辑电平表示逻辑常量的一种约定，如果用高电平表示逻辑1，用低电平表示逻辑0，称为正逻辑表示；反之，如果用高电平表示逻辑0，用低电平表示逻辑1，称为负逻辑表示。数字系统中一般采用正逻辑表示法。

逻辑代数中的变量称为**逻辑变量**，如逻辑变量 F、A、B 等，与普通代数的变量不同的是，任何逻辑变量的取值只有两种可能性：0或1。

3.1.2 逻辑代数的基本运算

1. 3种基本运算

逻辑代数包含3种基本的逻辑运算：逻辑与运算、逻辑或运算以及逻辑非运算。

对于逻辑变量 a，b，逻辑与运算可表示为 $a \cdot b$ 或 $a \wedge b$ 或 ab，只有当所有的逻辑变量取值为1时，结果才为1，否则为0。

逻辑或运算可表示为 $a+b$ 或 $a \vee b$，只要有一个逻辑变量取值为1时，结果就为1，只有当所有的逻辑变量取值为0时，结果才为0。

逻辑非运算是一元运算，也称取反运算。对逻辑变量 a，非运算可表示为 $\bar{a}$，其结果是逻辑变量 a 取反，即0取反为1，1取反为0。对于一个逻辑变量，也称之为**原变量**，该变量

取反后称为**反变量**，如 a 为原变量，$\overline{a}$ 为反变量。

逻辑代数3种基本逻辑运算的功能如表3.1所示。

表3.1　逻辑代数的3种基本逻辑运算的功能

与运算			或运算			非运算	
a	b	ab	a	b	$a+b$	a	$\overline{a}$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

2. 复合运算

把与、或、非3种基本运算组合起来使用称为复合运算。常见的复合运算有与非、或非、与或非、同或、异或等。

与非运算是先将逻辑变量进行与运算，再将结果进行非运算。如逻辑变量 A、B、C 的与非表达式为 $\overline{ABC}$。

或非运算是先将逻辑变量进行或运算，再将结果进行非运算。如逻辑变量 A、B、C 的或非表达式为 $\overline{A+B+C}$。

与或非运算是将逻辑变量进行与运算，再将每个与项进行或非运算。如逻辑变量 A、B、C、D 的一种与或非表达式为 $\overline{AB+CD}$。

异或和同或运算是两个变量与运算和或运算组成的复合运算，分别用⊕和⊙表示。如逻辑变量 A、B。

异或：$A \oplus B = A\overline{B} + \overline{A}B$

同或：$A \odot B = AB + \overline{A}\overline{B}$

3.1.3　基本逻辑公式、定理和规则

1. 逻辑公式

逻辑代数有一些常用的基本公式，这些公式是逻辑定理、逻辑规则的推导基础，也是逻辑化简的依据，公式本身可以通过真值表证明。逻辑代数的基本公式如表3.2所示。

表3.2　逻辑代数的基本公式

公式名称	公式	
交换律	$A+B=B+A$	$A \cdot B = B \cdot A$
结合律	$(A+B)+C=A+(B+C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
分配律	$A \cdot (B+C)=A \cdot B+A \cdot C$	$A+(B \cdot C)=(A+B) \cdot (A+C)$
吸收律	$A+AB=A \quad A+\overline{A}B=A+B$	$A(A+B)=A \quad A(\overline{A}+B)=AB$
包含律	$AB+\overline{A}C+BC=AB+\overline{A}C$	$(A+B)(\overline{A}+C)(B+C)=(A+B)(\overline{A}+C)$
互不律	$A+\overline{A}=1$	$A \cdot \overline{A}=0$
0—1律	$A+0=A \quad A+1=1$	$A \cdot 1=A \qquad A \cdot 0=0$
对合律	$\overline{\overline{A}}=A$	
重叠律	$A+A=A$	$A \cdot A=A$
反演律	$\overline{A+B}=\overline{A} \cdot \overline{B}$	$\overline{AB}=\overline{A}+\overline{B}$

2. 逻辑定理

定理 1 德·摩根(De Morgan)定理

(a) $\overline{x_1+x_2+\cdots+x_n}=\overline{x_1}\cdot\overline{x_2}\cdot\cdots\cdot\overline{x_n}$

(b) $\overline{x_1\cdot x_2\cdot\cdots\cdot x_n}=\overline{x_1}+\overline{x_2}+\cdots+\overline{x_n}$

即 n 个变量的“或”的“非”等于各变量的“非”的“与”；n 个变量的“与”的“非”等于各变量的“非”的“或”。

定理 2 香农(Shannon)定理

$$\overline{f(x_1,x_2,\cdots,x_n,0,1,+,\cdot)}=f(\overline{x_1},\overline{x_2},\cdots,\overline{x_n},1,0,\cdot,+)$$

即任何函数的反函数(或称补函数)，可以通过对该函数的所有变量取反，并将常量 1 换为 0，0 换为 1，运算符“+”换为“·”，“·”换为“+”而得到。

例 3.1 已知函数 $F=\overline{A}B+A\overline{B}(C+\overline{D})$，求其反函数 $\overline{F}$。

解：$\overline{F}=\overline{\overline{A}B+A\overline{B}(C+\overline{D})}=\overline{\overline{A}B}\cdot\overline{A\overline{B}(C+\overline{D})}$

$=(A+\overline{B})\cdot(\overline{A\overline{B}}+\overline{C+\overline{D}})=(A+\overline{B})((\overline{A}+B)+\overline{C}D)$

$=(A+\overline{B})\cdot(\overline{A}+B+\overline{C}D)$

利用香农定理，可以直接写出

$$\overline{F}=(A+\overline{B})\cdot(\overline{A}+B+\overline{C}D)$$

定理 3 展开定理

(a) $f(x_1,x_2,\cdots,x_i,\cdots,x_n)=x_if(x_1,x_2,\cdots,1,\cdots,x_n)+\overline{x_i}f(x_1,x_2,\cdots,0,\cdots,x_n)$

(b) $f(x_1,x_2,\cdots,x_i,\cdots,x_n)=[x_i+f(x_1,x_2,\cdots,0,\cdots,x_n)]\cdot[\overline{x_i}+f(x_1,x_2,\cdots,1,\cdots,x_n)]$

即任何布尔函数都可以对它的某一变量 x_i 展开，可以按(a)式展开成“与-或”形式，也可以按(b)式展开成“或-与”形式。

3. 逻辑规则

规则 1 代入规则

对于任何一个逻辑等式，将其中任意一个变量所有出现的位置都以某个逻辑变量或逻辑函数同时取代后，等式依然成立。

规则 2 反演规则

在逻辑代数中，常将 $\overline{F}$ 叫作逻辑函数 F 的**反函数**或**补函数**。反演规则是将一个逻辑函数表达式 F 中所有“与”运算符换为“或”运算符；所有“或”运算符换为“与”运算符；所有原变量换为反变量；所有反变量换为原变量；“0”换为“1”；“1”换为“0”，所得新的逻辑表达式为原函数的反函数。获得反函数的规则就是反演规则。

规则 3 对偶规则

设 F 为一个逻辑表达式，若将 F 中的“与”运算符换为“或”运算符；将“或”运算符换为“与”运算符；将“1”换为“0”，将“0”换为“1”；所得新的逻辑函数表达式称为 F 的**对偶式**，记作 F'，获得对偶式规则称为对偶规则。

如果两个逻辑函数表达式相等，那么它们的对偶式也一定相等。

3.1.4 逻辑函数

逻辑函数反映了逻辑变量之间的关系,在数字系统中,将逻辑变量作为输入,将运算结果作为输出,当输入变量的取值确定以后,输出的值便被唯一地确定下来,这种输出变量和输入变量间的逻辑关系称为**逻辑函数**。记为

$$Y=F(A_1,A_2,\cdots,A_n)$$

其中 $A_1,A_2,\cdots,A_n$ 为输入逻辑变量,取值是 0 或 1;Y 为输出逻辑变量,取值是 0 或 1,F 为某种逻辑关系,这种逻辑关系由与、或、非 3 种基本运算决定。

逻辑函数常用逻辑表达式表示,其功能也可以通过真值表或卡诺图描述。

1. 真值表

真值表是一种由逻辑变量的所有可能取值组合及其对应的逻辑函数值所构成的表格。由于一个逻辑变量只有 0 和 1 两种可能的取值,故 n 个逻辑变量一共有 2^n 种可能的取值组合,将这 n 种取值可能和每一种取值下的函数值用表格的形式表示出来就是真值表。

例 3.2 某奇偶性判断电路,输入变量为 A、B、C,输出为 F,该电路的功能是,当 A、B、C 中 1 的个数为奇数时,F 为 1,否则 F 为 0。该电路实现的逻辑关系可用如表 3.3 所示的真值表表示。

表 3.3 三变量奇偶判断电路真值表

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

在上面的例子中,对输入变量所有输入的组合都有确定的输出结果,但在实际问题中,往往存在这样两种情况:

(1) 由于某种特殊限制,使得输入变量的某些取值组合根本不会出现。

(2) 虽然每种输入组合都可能出现,但其中的某些输入取值组合究竟使函数值为 1 还是为 0,对输出没有影响。

这部分输入组合称为**无关最小项**,简称无关项,也称为**约束项**。在真值表中无关项对应的函数取值用 d 表示。包含无关项的逻辑函数称不完全确定的逻辑函数。

2. 逻辑表达式

逻辑表达式是由逻辑变量和与、或、非 3 种基本的逻辑运算组成的表达式。如

$$Y_1=F(A,B,C)=\overline{A}B+A\overline{C}+ABC$$

$$Y_2=F(A,B,C)=(A+B+C)(B+\overline{C})$$

上述逻辑函数 Y_1 中,逻辑表达式由 3 个与项进行或运算组成,函数的这种形式称为

与或表达式，其中，$\overline{A}B$、$A\overline{C}$ 和 ABC 由与运算组成，称为**与项**（也称为**乘积项**）。如果一个与项包含了该函数的所有变量，每个变量均以原变量或反变量的形式出现，且仅出现一次，这样的与项称为**最小项**（也称为**标准积**）。逻辑函数 Y_1 中，与项 ABC 是最小项。

n 个变量最多可以组成 2^n 个最小项，为了方便使用，将最小项进行标号，用 m_i 表示，下标 i 的编号规则：把乘积项的原变量记作“1”，反变量计作“0”，把每个与项表示为一个二进制数，这个二进制数所对应的十进制数就是 i 的值。如 3 个变量 A、B、C 最多可以组成 8 个最小项：$\overline{A}\overline{B}\overline{C}$，$\overline{A}\overline{B}C$，$\overline{A}B\overline{C}$，$\overline{A}BC$，$A\overline{B}\overline{C}$，$A\overline{B}C$，$AB\overline{C}$，$ABC$。

这 8 个最小项相对应的简写形式分别是 m_0，m_1，m_2，…，m_7。三变量全部最小项及编号如表 3.4 所示。

上述逻辑函数 Y_2 中，逻辑表达式由两个或项进行与运算组成，函数的这种形式称为**或与表达式**，其中，$(A+B+C)$和$(B+\overline{C})$由或运算组成，称为**或项**（也称为**相加项**），如果一个或项包含了该函数的所有变量，每个变量均以原变量或反变量的形式出现，且仅出现一次，这样的或项称为**最大项**（也称为**标准和**）。逻辑函数 Y_2 中，或项$(A+B+C)$是最大项。

同样，n 个变量最多可以组成 2^n 个最大项，为了方便使用，将最大项进行标号，用 M_i 表示，下标 i 的编号规则：把乘积项的原变量记作“0”，反变量记作“1”，把每个或项表示为一个二进制数，这个二进制数所对应的十进制数就是 i 的值。如 3 个变量 A、B、C 最多可以组成 8 个最小项：$\overline{A}+\overline{B}+\overline{C}$，$\overline{A}+\overline{B}+C$，$\overline{A}+B+\overline{C}$，$\overline{A}+B+C$，$A+\overline{B}+\overline{C}$，$A+\overline{B}+C$，$A+B+\overline{C}$，$A+B+C$。

这 8 个最大项相对应的简写形式分别是 M_7，M_6，…，M_0。三变量全部最大项及编号如表 3.4 所示。

表 3.4　三变量全部最小项和最大项及编号

变量	最小项及编号		最大项及编号	
ABC	编号	最小项	编号	最大项
000	m_0	$\overline{A}\overline{B}\overline{C}$	M_0	$A+B+C$
001	m_1	$\overline{A}\overline{B}C$	M_1	$A+B+\overline{C}$
010	m_2	$\overline{A}B\overline{C}$	M_2	$A+\overline{B}+C$
011	m_3	$\overline{A}BC$	M_3	$A+\overline{B}+\overline{C}$
100	m_4	$A\overline{B}\overline{C}$	M_4	$\overline{A}+B+C$
101	m_5	$A\overline{B}C$	M_5	$\overline{A}+B+\overline{C}$
110	m_6	$AB\overline{C}$	M_6	$\overline{A}+\overline{B}+C$
111	m_7	ABC	M_7	$\overline{A}+\overline{B}+\overline{C}$

同一个逻辑函数可以有不同的表达式，但其标准形式是唯一的。逻辑函数的标准形式有两种，即标准与或式和标准或与式。

全部由最小项相或组成的逻辑表达式称为标准与或式，也称为标准积之和。如逻辑函数

$$F(A,B,C)=\overline{A}\overline{B}C+\overline{A}B\overline{C}+A\overline{B}C+ABC$$

该函数是标准与或式，包含 4 个最小项，可以简记为

$$F(A,B,C)=m_1+m_2+m_5+m_7=\sum m(1,2,5,7)$$

全部由最大项相与组成的逻辑表达式称为标准或与式,也称为标准和之积。如逻辑函数

$$F(A,B,C)=(\overline{A}+\overline{B}+C)(\overline{A}+B+C)(A+B+\overline{C})$$

该函数是标准或与式,包含3个最大项,可以简记为

$$F(A,B,C)=M_6M_4M_1=\prod M(1,4,6)$$

逻辑函数的标准形式可以在一般形式的基础上利用逻辑公式进行逻辑变换得到,也可以在真值表的基础上直接得出。在真值表中,使函数取值为1的那些最小项相或,就构成了函数的"标准积之和"形式;使函数取值为0的最大项相与,就构成了函数的"标准和之积"形式。如例3.2,根据表3.3可直接写出该函数的"标准积之和"形式和"标准和之积"形式,分别是

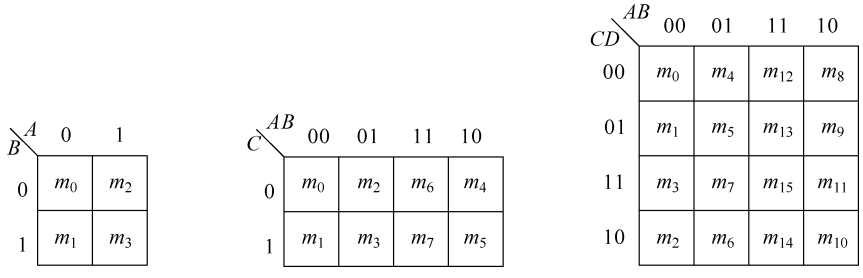
$$F(A,B,C)=\overline{A}\overline{B}C+\overline{A}B\overline{C}+A\overline{B}\overline{C}+ABC=\sum m(1,2,4,7)$$

$$F(A,B,C)=(A+B+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)=\prod M(0,3,5,6)$$

同一函数的最大项与最小项是互斥的,即如果真值表中的某一行作为函数的最小项,那么它就不可能是同一函数的最大项;反之亦然。换句话说,一个布尔函数的最小项的集合与其最大项的集合互为补集。因此,若已知一布尔函数的"标准积之和"形式,根据互补的原则就可以很容易写出该函数的"标准和之积"形式。

3. 卡诺图

卡诺图是真值表的变形,与真值表一样,也是采用穷举的方法把输入变量所有输入组合及其函数值列举出来,所不同的是卡诺图是一种用二维小方格来构成图形,一个小方格对应一个最小项,n个变量的卡诺图包含2^n个小方格。其中,2^n个小方格构成卡诺图的规则是利用最小项的相邻性,所谓相邻最小项是指两个最小项只有一位不一样。具体的做法:将全部变量按顺序分成两组,每一组变量按格雷码取值排列。图3.1(a)、图3.1(b)、图3.1(c)分别是二变量、三变量和四变量卡诺图。对于五变量及以上的卡诺图方格比较多,逻辑函数中用得较少。



(a) 二变量卡诺图　(b) 三变量卡诺图　(c) 四变量卡诺图

图3.1　卡诺图

用卡诺图表示逻辑函数一般分为两步,首先根据逻辑变量的个数画出卡诺图框架,然后将逻辑函数的最小项填到对应的小方格中,用1表示,其他方格用0表示或者不表示。

3.1.5　逻辑函数化简

同一个逻辑函数往往有多种不同的逻辑表达式,有的简单,有的复杂,差别很大。在数

字系统中，不同的表达式对应的硬件电路是不一样的，简单的表达式对应的硬件电路简单，实现起来不仅可以节省硬件成本，还可以提高电路的可靠性。因此，在进行逻辑电路设计时，对逻辑函数化简就显得十分重要。

逻辑代数中常用与或式衡量一个逻辑函数是不是最简，此时要求函数表达式中与项的个数要最少，每个与项中的变量数要最少。

1. 公式化简法

公式化简法是利用逻辑公式、定理和规则对逻辑函数进行等价变换，最终转化成最简的形式。

例 3.3 化简 $F=A\overline{C}+ABC+AC\overline{D}+CD$

解：$F=A\overline{C}+ABC+AC\overline{D}+CD$

$=A(\overline{C}+BC)+C(A\overline{D}+D)$

$=A(\overline{C}+B)+C(A+D)$

$=A\overline{C}+AB+AC+CD$

$=A(\overline{C}+C)+AB+CD$

$=A+AB+CD$

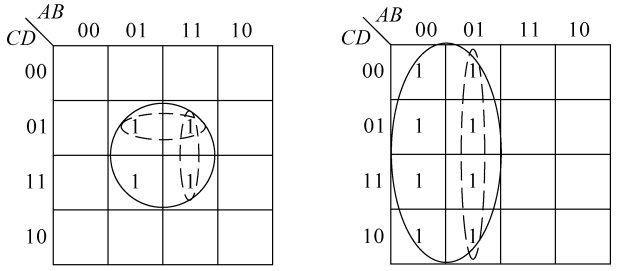
$=A+CD$

理论上，不管一个逻辑函数包含多少个逻辑变量，利用公式化简法最终都可以将其化成最简，但是公式化简法技巧性太强，而且结果也不好判断是不是最简。

2. 卡诺图化简法

在卡诺图上，两个相邻的最小项有一个变量相异，进行或运算可以消去这个变量，4 个相邻的最小项有两个变量相异，进行或运算可以消去这两个变量，以此类推，2^n 个最小项相邻有 n 个最小项相异，进行或运算可以消去这 n 个变量。

对四变量及以下卡诺图而言，相邻最小项在卡诺图上的位置有两种情况，即物理位置相邻和端点相邻。物理位置相邻是指同一行（或列）上的 2^n 个靠在一起的最小项是相邻的，如图 3.2 所示的虚线圈内的最小项；靠在一起的两两互为相邻的 2^n 最小项是相邻的，如图 3.2 所示的实线圈内的最小项。



(a) 情况1　　(b) 情况2

图 3.2　物理位置相邻最小项

端点相邻是指同一行（或列）上最左边和最右边（或最上边和最下边）的两个最小项是相邻的，如图 3.3 所示的虚线圈内的最小项；端点上两两互为相邻的 2^n 最小项是相邻的，如图 3.3 所示的实线圈内的最小项。

利用卡诺图最小项相邻的特点，可以对逻辑函数进行化简。一般步骤是先将逻辑函数

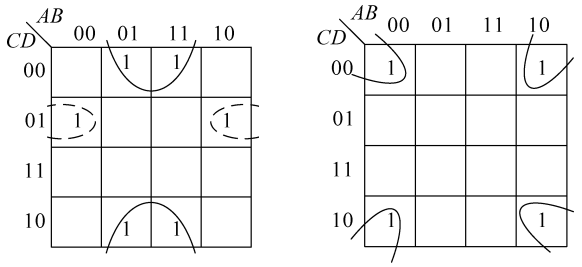


图 3.3　端点相邻最小项

用卡诺图表示,然后在卡诺图上将相邻最小项圈起来,最后将每一个圈对应的最小项相或即可得到逻辑函数的最简与或式。其中,画圈原则:圈包含的1的个数是2^n,圈尽可能的大(消去的变量多,对应的与项包含的变量个数少),圈尽可能的少(对应的与项个数少),所有的1都要被圈进来。

另外,如果卡诺图中包含无关项d,由于无关项对应的输入变量取值组合根本不会出现,或者尽管可能出现,但相应的函数值是什么无关紧要,因此,在化简画圈时,无关项d可以根据需要取值0或者取值1。

例 3.4　用卡诺图化简逻辑函数$F(A,B,C,D)=\sum m(0,2,5,7,8,10)$。

解:

① 逻辑函数F用卡诺图表示,如图3.4(a)所示;

② 找相邻最小项,画圈,如图3.4(b)所示,虚线圈是2个最小项相邻,实线圈是4个最小项相邻;

③ 写出每个圈所对应的与项,如图3.4(b)所示。

因此,逻辑函数化简后的结果为$F=\overline{A}BD+\overline{B}\overline{D}$。

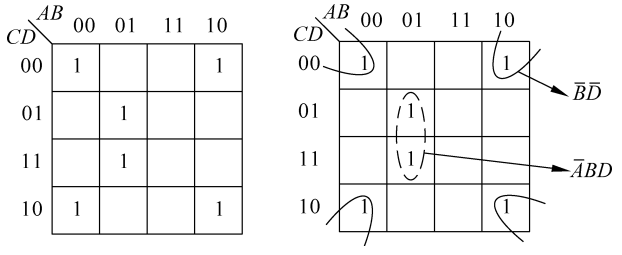
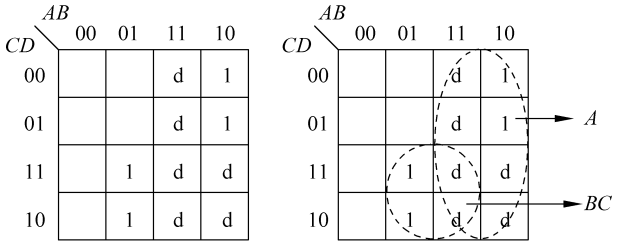


图 3.4　卡诺图及化简

例 3.5　逻辑变量A、B、C、D输入8421码,如果8421码的值大于5则输出变量F为1,否则为0。画出逻辑函数$F(A,B,C,D)$的卡诺图并化简。

解:8421码的编码范围为0000～1001,编码1010～1111为非法编码,即无关项。逻辑函数$F(A,B,C,D)$的卡诺图如图3.5(a)所示,化简方法如图3.5(b)所示。化简结果:$F=A+BC$。

卡诺图化简法虽然简单,但只适用于逻辑变量不多的逻辑函数。



(a) 卡诺图　　(b) 化简方法

图 3.5　卡诺图及化简

3.2　逻 辑 电 路

逻辑函数描述的逻辑功能，在数字系统中最终可以通过逻辑门电路、集成电路或可编程逻辑电路实现。

3.2.1　逻辑门电路

数字系统中，常用的门电路有与门、或门、非门、与非门和或非门等，分别实现逻辑函数中的与运算、或运算、非运算、与非运算和或非运算，图 3.6 是这些常用门电路的国家标准和国际电工委员会标准符号、美国传统符号和中国传统符号。

3.2.2　门电路的实现

基本的逻辑门电路可以由二极管门电路、晶体管门电路、TTL 门电路和 CMOS 门电路等构造。

1. 二极管门电路

二极管具有单向导电性，可用作开关电路，通过电压控制开关状态，当在两端加正向电压时导通，此时二极管可视为开关闭合状态，如图 3.7(a)所示；当加反向电压时截止，二极管可视为开关断开状态，如图 3.7(b)所示。

利用二极管的单向导电性可以构成基本门电路，图 3.8 所示为两个二极管和一个电阻构成的最简单的与门电路，其中，A、B 为两输入变量，Y 为输出变量。表 3.5 为该与门电路真值表。

表 3.5　与门电路真值表和逻辑电平

A		B		Y	
电平	逻辑值	电平	逻辑值	电平	逻辑值
L	0	L	0	L	0
L	0	H	1	L	0
H	1	L	0	L	0
H	1	H	1	H	1

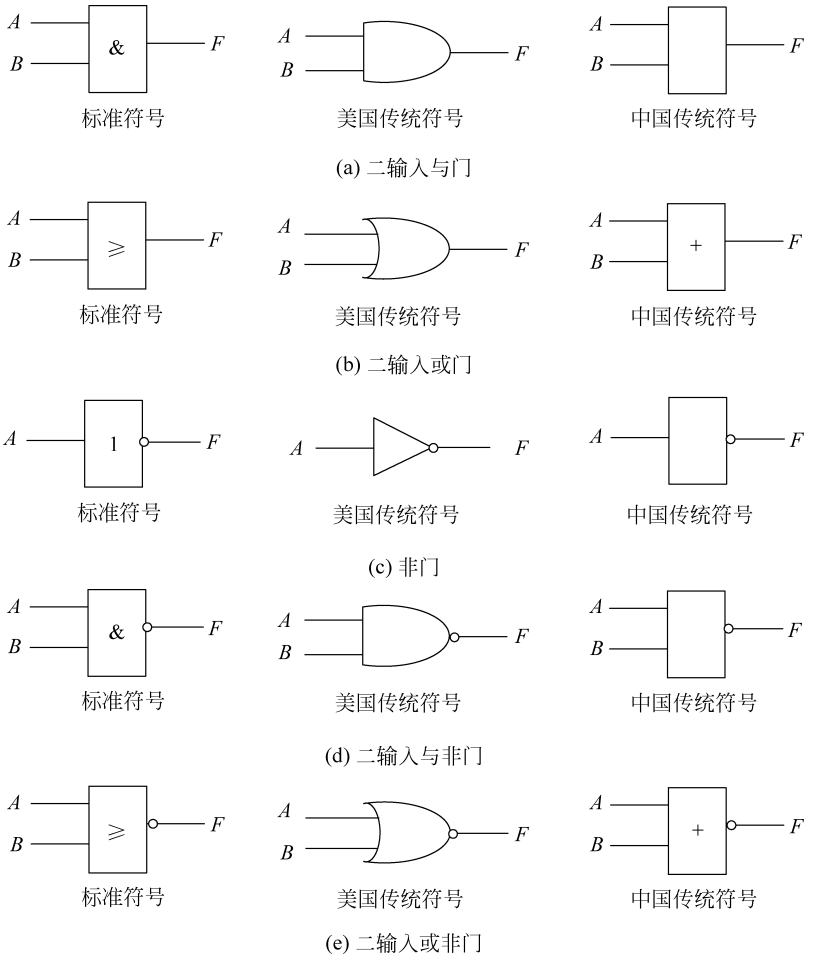


图 3.6　基本逻辑门的图形符号

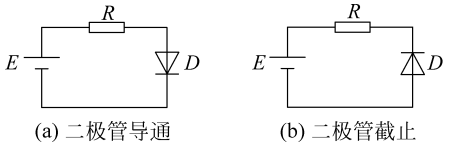


图 3.7　二极管的开关特性

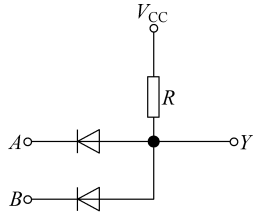
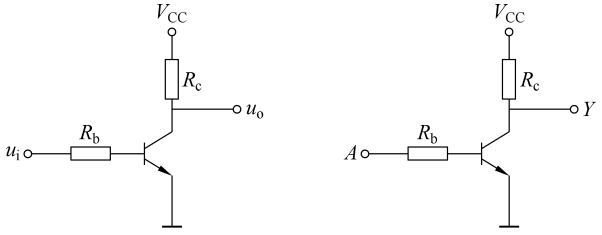


图 3.8　二极管与门电路

2. 晶体管门电路

晶体管也可用作开关电路，通过基极电压 u_i 控制开关状态，如图 3.9(a)所示，当 u_i 为高电平时，三极管饱和导通，u_o 为低电平；当 u_i 为低电平时，三极管截止，u_o 为高电平。图 3.9(b)所示为晶体管构造的非门电路，A 为输入变量，Y 为输出变量。表 3.6 为该非门电路真值表。



(a) 晶体管开关电路　　(b) 晶体管非门电路

图 3.9　晶体管开关电路和非门电路

表 3.6　非门电路真值表和逻辑电平

A		Y	
电平	逻辑值	电平	逻辑值
L	0	H	1
H	1	L	0

由二极管、晶体管等组成的逻辑门电路，结构简单、成本低，但存在严重的缺点，如输出高低电平的数值与输入高低电平的数值不相等，如果将这些门的输出作为下一级门的输入，将产生高低电平的偏移，因此，在实际应用中一般都采用集成门电路，常用的集成门电路分为 TTL 和 CMOS 两大类。

3. TTL 门电路

晶体管-晶体管逻辑(Transister-Transister-Logic，TTL)电路，是数字集成电路的一大门类，它采用双极型工艺制造，具有高速度和品种多等特点。

图 3.10(a)所示为 TTL 与非门电路，电路由输入级、倒相级和输出级三个部分组成。其中 D_1、D_2 是钳位二极管，用于限制输入端出现的负极性干扰脉冲，分析时可暂不考虑。多发射极三极管可以等效为一个二极管组成的与门电路。设输入信号的高、低电平分别为 3.6V 和 0.3V。

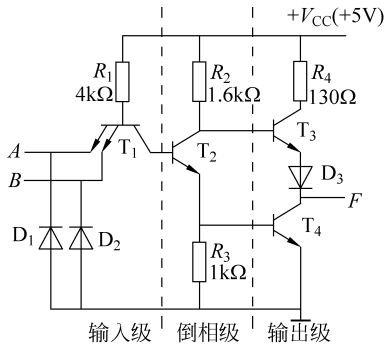
当输入 A、B 至少有一个低电平，即输入端 A、B 中至少有一个为 0.3V 时，电路的工作情况如图 3.10(b)所示，二极管 D_4、D_5 中至少有一个导通，使 P 点电位为 1V，这时二极管 D_6 和三极管 T_2、T_4 截止，其集电极电流都近似为 0，二极管 D_3、三极管 T_3 导通，假设 T_3 导通时基极—发射极之间的压降 u_{BE_3} 为 0.7V，则输出电压为 $u_F = V_{CC} - u_{R_2} - u_{BE_3} \approx u_{D_3} \approx 5V - 0V - 0.7V - 0.7V = 3.6V$，即输出为高电平。

当输入端 A、B 全为高电平时，电路的工作情况如图 3.10(c)所示，D_4、D_5 均不能导通，P 点电位被钳制在 2.1V，T_2、T_4 饱和导通，T_2 的集电极电位近似为 1V，T_3、D_5 截止，此时，电路输出电压为 0.3V，即输出为低电平。

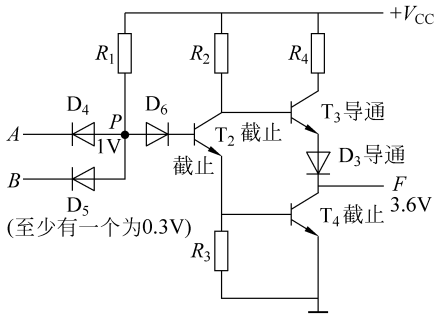
因此，该电路的逻辑功能如表 3.7 所示，输出与输入之间的逻辑关系为 $F = \overline{AB}$。

4. CMOS 门电路

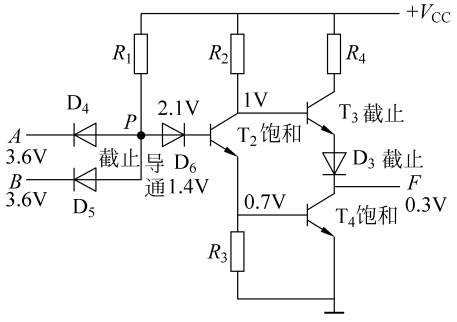
CMOS 门电路是在 TTL 门电路问世之后又一种广泛应用的数字集成器件。由于制造工艺的改进，CMOS 门电路的性能超越 TTL 而成为占主导地位的逻辑器件。CMOS 门电路的工作速度与 TTL 相当，而它的功耗和抗干扰能力则远优于 TTL。此外，几乎所有的超大规模存储器件以及 PLD 器件都采用 CMOS 工艺制造，且费用较低。



(a) 与非门电路



(b) 输入端至少有一个低电平



(c) 输入端全为高电平

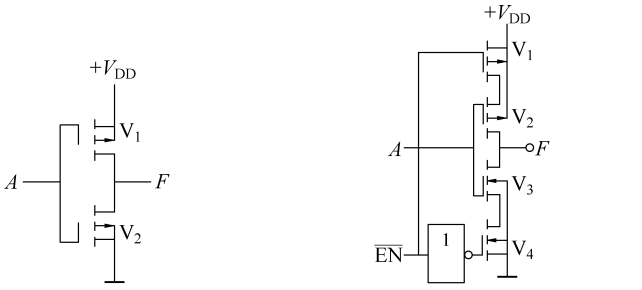
图 3.10　TTL 与非门电路

表 3.7　与非门电路真值表和逻辑电平

A		B		Y	
电平	逻辑值	电平	逻辑值	电平	逻辑值
L	0	L	0	L	0
L	0	H	1	L	0
H	1	L	0	L	0
H	1	H	1	H	1

图 3.11(a)为 CMOS 非门电路，其中 V_1 为 P 沟道增强型 MOS 管，衬底与源极相连接 V_{DD}。当栅源电压小于它的开启电压 UGS1(TH)(UGS1(TH)＜0V)时，管子导通。而 V_2 为 N 沟道增强型 MOS 管，衬底也是与源极相连接地。当栅源电压大于它的开启电压 UGS2(TH)(UGS2(TH)＞0V)时，管子导通。当输入端 A 为高电平时($V_A \approx V_{DD}$)，V_1 截止、V_2 导通，输出端 F 为低电平；当输入端 A 为低电平时($V_A \approx 0$V)，V_1 导通、V_2 截止，输出端 F 为高电平。所以电路输出与输入之间的逻辑关系为：$F=\overline{A}$。

图 3.11(b)是低电平有效的 CMOS 三态非门电路。当 $\overline{EN}=1$ 时，V_1、V_4 截止，这时不论 A 为何种状态，输出端 F 为高阻状态。当 $\overline{EN}=0$ 时，V_1、V_4 导通，这时电路为非门电路，$F=\overline{A}$。



(a) CMOS非门电路　　(b) 低电平有效的CMOS三态非门电路

图 3.11　CMOS 非门和三态门

3.2.3　集成电路

集成电路(Integrated Circuit,IC)就是将半导体器件、电阻、电容及导线等都制造在一个半导体基片(通常是硅片)上,构成一个功能完整的电路,然后封装起来。与分立元件电路相比,集成电路具有体积小、重量轻、可靠性高、功耗低和工作速度快等优点,自 20 世纪 50 年代末以来,得到了广泛的应用。随着半导体技术和制造工艺的迅速发展,在单块硅片上集成器件的规模越来越大,集成电路的功能也日趋复杂和完善。通常把单块硅片上集成的三极管的个数称为集成度。

根据集成度的大小,集成电路分为小规模集成电路(SSI)、中规模集成电路(MSI)、大规模集成电路(LSI)和超大规模集成电路(VLSI)。随着集成度的提高,单个芯片的功能也越来越强。一般来说,SSI 仅仅是器件(如门电路、触发器等)的集成,MSI 已是逻辑部件(如译码器、寄存器等)的集成,而 LSI 和 VLSI 则是整个数字系统或其子系统的集成。

根据功能是否确定,集成电路可分为标准逻辑器件和专用集成电路(Application Specific Integrated Circuit,ASIC)。标准逻辑器件的功能是完全确定的,用户只能使用其固有的功能,而不能对其改变。标准逻辑器件通常又称为通用集成电路,它具有生产量大、使用广泛、价格低廉等优点,如门电路、触发器和 MSI 电路等。随着数字系统规模的不断扩大,用标准逻辑器件实现,需要用很多芯片,且芯片之间、芯片与印制板之间的连线也相应增多,导致系统可靠性下降、成本提高、功耗增加和占用空间扩大等问题。因此,随着集成电路技术和计算机技术的发展,出现了把能完成特定功能的电路或系统集成到一个芯片内的专用集成电路 ASIC 上,使用 ASIC 不仅可以减少系统的占用空间,而且可以降低功耗,提高电路的可靠性和工作速度。ASIC 按制造过程的不同又分为全定制和半定制两大类。全定制集成电路是由制造厂家按用户提出的逻辑要求,针对某种应用而专门设计和制造的芯片,这类芯片专业性很强,适合在大批量生产的产品中使用,常见的存储器、中央处理器(CPU)等芯片,都属于全定制集成电路。半定制集成电路是制造厂家生产出的半成品,用户根据自己的要求,用编程的方法对半成品进行再加工,制成具有特定功能的专用集成电路,可编程逻辑器件(Programmable Logic Devices,PLD)就是这类半定制集成电路。

根据所用半导体器件的不同,集成电路可分双极型数字集成电路和单极型数字集成电路。前者以双极型晶体管为基本元件,后者以 MOS 管为基本元件。实际应用中,TTL 和 CMOS 电路最为普遍。

3.2.4 可编程逻辑电路

可编程逻辑器件按集成度划分,可分成低密度可编程逻辑器件和高密度可编程器件。常见的高密度可编程逻辑器件有CPLD(Complex PLD)以及FPGA等。下面介绍PLD和FPGA。

1. PLD

PLD属于半定制集成电路。由于任何组合逻辑电路对应的逻辑函数均可以转换为"与-或"表达式,任何时序逻辑电路都是由组合电路加上存储元件(如触发器)构成的,因此,任何一个逻辑电路最终都可以由与门阵列、或门阵列和触发器阵列实现。图3.12是PLD的基本结构示意图,主体电路是由与阵列后跟或阵列组成的,这两个阵列或者其中的一个阵列是可以编程的。输入变量通过与阵列,完成与运算,产生相应的与项;再将这些与项通过或阵列,完成或运算,得到所需"与-或"输出表达式。

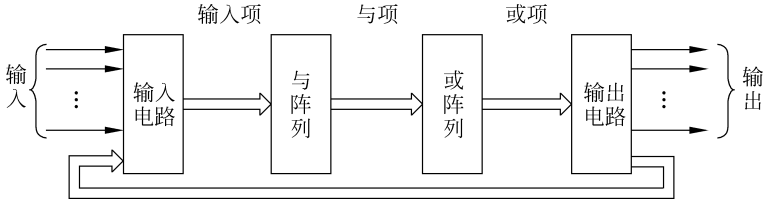


图3.12 PLD基本结构示意图

为了使输入信号具有足够的驱动能力,并产生原变量和反变量两个互补的输入信号,与阵列的每个输入端都与输入缓冲器相连,图3.13所示为PLD的典型输入缓冲器,它的两个输出B,C是其输入A的原和反,即$B=A$,$C=\overline{A}$。为使PLD具有多种输出方式——组合输出、时序输出、低电平输出、高电平输出等,往往在或阵列的后面还有各种输出电路,各种方式的输出都经三态电路而输出,输出信号还可以反馈到与阵列的输入端。

由于PLD的阵列规模大,用传统的逻辑表示方法很难描述PLD的内部电路,目前广泛采用PLD表示法。图3.14所示为三输入"与"门的两种表示法。传统表示法中与门的3个输入A,B,C在PLD表示法中称为3个输入项,而输出D称为与项。同样或门也采用类似的方法表示。

PLD阵列交叉点处的连接方式有3种,如图3.15所示。实点"·"表示硬线连接,即固定连接;"×"表示可编程连接;既没有"·"也没有"×"表示两线不连接。

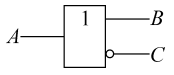


图3.13 输入缓冲器

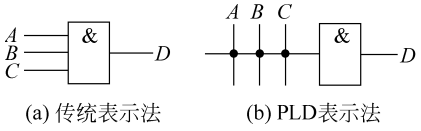


图3.14 两种与门表示法

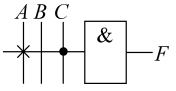
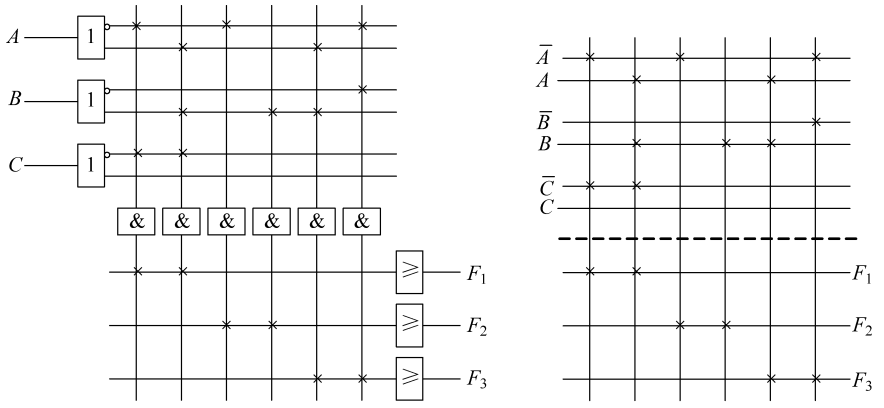


图3.15 PLD阵列交叉点

通过PLD表示法将逻辑电路描述出来得到的图称为阵列图。如某电路对应的逻辑函数为$F_1=\overline{A}\overline{C}+AB\overline{C}$,$F_2=\overline{A}+B$,$F_3=AB+\overline{A}\overline{B}$,该电路的阵列图如图3.16(a)所示,图3.16(b)为该阵列图的简化形式。

用可编程逻辑器件设计数字系统,相对于传统的用标准逻辑器件(门、触发器和MSI电



(a) 电路的阵列图　　(b) 阵列图的简化形式

图 3.16　PLD 阵列图

路)设计数字系统有很多优点,主要表现在以下方面:

(1) 减小了系统体积。

(2) 增强了逻辑设计的灵活性

(3) 提高了系统的处理速度和可靠性。

(4) 缩短了设计周期,降低了系统成本。

(5) 具有加密功能。

2. FPGA

FPGA(Field Programmable Gate Array)是现场可编程门阵列,是专用集成电路领域中的一种半定制电路,既解决了定制电路的不足,又克服了原有可编程器件门电路数有限的缺点。

FPGA 采用了逻辑单元阵列(Logic Cell Array,LCA)这样一个概念,内部包括可配置逻辑模块(Configurable Logic Block,CLB)、输出输入模块(Input Output Block,IOB)和内部连线(Interconnect)三个部分。与 PLD 以与阵列、或阵列、触发器结构方式构成逻辑行为不同,FPGA 是以查找表(Look-Up-Table,LUT)来实现组合逻辑,每个查找表连接到一个 D 触发器的输入端,触发器再驱动其他逻辑电路或驱动 I/O,由此构成了既可实现组合逻辑功能又可实现时序逻辑功能的基本逻辑单元模块,这些模块间利用金属连线互相连接或连接到 I/O 模块。

LUT 是 FPGA 可编程的最小逻辑构成单元,LUT 本质上就是一个 RAM,一个 N 输入的查找表可以实现 N 输入变量的任何组合逻辑功能。目前 FPGA 中多使用四输入的 LUT,每一个 LUT 可以看成一个有 4 位地址线的 16×1 的 RAM。当用户通过原理图或 HDL 描述了一个逻辑电路以后,FPGA 开发软件会自动计算逻辑电路的所有可能的结果,并把结果事先写入 RAM,这样,每输入一个信号进行逻辑运算就等于输入一个地址进行查表,找出地址对应的内容并输出。图 3.17 所示为四输入与门的 LUT 实现方式。由此可见,FPGA 的逻辑是通过向内部静态存储单元加载编程数据实现的,存储在存储器单元中的值决定了逻辑单元的逻辑功能以及各模块间或模块与 I/O 间的连接方式,最终决定了 FPGA 所能实现的功能。

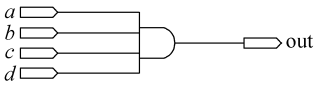
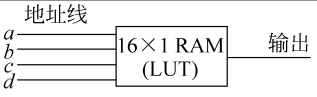
<table>
<tr><td colspan="2">实际逻辑电路</td><td colspan="2">LUT的实现方式</td></tr>
<tr><td colspan="2"> 
</td><td colspan="2"> 
</td></tr>
<tr><td>a,b,c,d 输入</td><td>逻辑输出</td><td>地址</td><td>RAM中存储的内容</td></tr>
<tr><td>0000</td><td>0</td><td>0000</td><td>0</td></tr>
<tr><td>0001</td><td>0</td><td>0001</td><td>0</td></tr>
<tr><td>⋮</td><td>0</td><td>⋮</td><td>0</td></tr>
<tr><td>1111</td><td>1</td><td>1111</td><td>1</td></tr>
</table>

图 3.17　四输入与门的 LUT 实现方式

3.2.5　逻辑电路的设计模式

在数字电子技术发展的过程中,逻辑电路设计出现了基于分立元件的设计、基于集成电路的设计和基于可编程逻辑器件的设计等几种设计模式。

基于分立元件的设计模式是传统的设计模式,它是数字逻辑设计的基础。该模式追求设计目标的最小化,即通过函数和状态表的化简,尽量减少门电路和触发器的数量,以得到一种实现给定功能的最经济的设计方案。

随着集成电路的发展,数字系统的逻辑设计方法发生了根本性的变化,出现了基于集成电路的设计模式。由于集成电路芯片内部的门电路或触发器的数量都是确定的,因此该模式的追求目标不再是最小化,而是以集成电路芯片的功能为基础,从要求的功能出发,合理地选用器件、充分利用器件的功能,尽量减少相互连线,并在必要时使用传统的设计方法设计辅助的接口电路。这种设计模式通常以使用的芯片数量和价格达到最低作为技术和经济的最佳指标,所设计的电路具有体积小、功耗低、可靠性高及易于设计、调试和维护等优点,因此发展异常迅猛。但是,要采用集成电路进行逻辑设计,首先必须熟悉集成电路芯片的功能和用法,了解其灵活性,才能有效地利用它们实现各种逻辑功能。而且,标准器件的逻辑功能固定,用户只能使用而不能修改器件逻辑,缺乏灵活性。

随着可编程逻辑器件的发展,出现了基于可编程逻辑器件的逻辑电路设计模式,即运用 EDA 技术完成逻辑电路的设计。EDA 即电子设计自动化(Electronic Design Automation),是指以大规模可编程逻辑器件为设计载体,以计算机为工具进行电子设计,利用 EDA 软件平台完成硬件系统的设计、仿真、测试与下载。FPGA/CPLD 设计流程如图 3.18 所示。

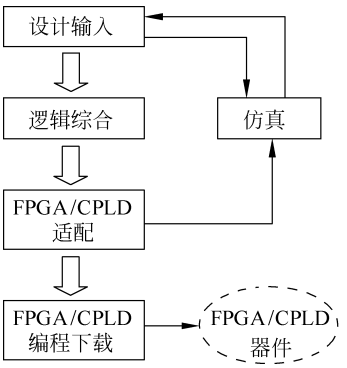


图 3.18　FPGA/CPLD 设计流程

3.3　计算机中常用的组合逻辑电路

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。组合逻辑电路由门电路组成,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。计算机中常见的

组合逻辑电路有加法器、译码器、多路选择器和三态门等。

3.3.1 加法器

1. 半加器

能完成两个一位二进制数的相加并求得“和”及“进位”的逻辑电路，称为**半加器**(Half Adder，HA)，逻辑符号如图 3.19 所示。其中 A 和 B 分别为两个一位二进制数的输入端，S 和 C 分别为相加后形成的“和”及“进位”输出端，半加器真值表如表 3.8 所示。

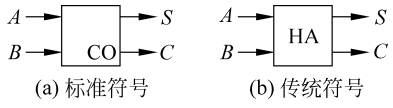
表 3.8 半加器真值表

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

根据真值表，可写出“和”及“进位”输出表达式分别是 $F=\overline{A}B+A\overline{B}$，$C=AB$。

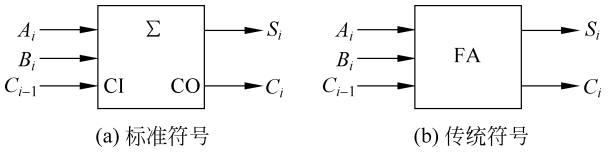
2. 全加器

完成两个一位二进制数相加，并考虑低位来的进位，求得“和”及“进位”的逻辑电路称为**全加器**(Full Adder，FA)，逻辑符号如图 3.20 所示。其中 A_i 和 B_i 分别为两个一位二进制数的输入端，C_{i-1} 为低位来的进位输入端，S_i 和 C_i 分别为相加后的“和”及向高位的“进位”输出端。全加器真值表如表 3.9 所示。



(a) 标准符号　(b) 传统符号

图 3.19 半加器的逻辑符号



(a) 标准符号　(b) 传统符号

图 3.20 全加器的逻辑符号

表 3.9 全加器真值表

A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

根据真值表，可写出“和”及向高位的“进位”输出函数的标准形式：

$$S_i=\sum m(1,2,4,7),\quad C_i=\sum m(3,5,6,7)$$

化简后的最简与或式为

$$S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

$$C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1}$$

也可以变换为

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1}$$

3. 二进制加法器

实现多位二进制数加法运算的电路称为二进制加法器。按各位数相加方式的不同,可将加法器分为串行加法器和并行加法器两种。**串行加法器**采用串行运算方式,数据串行输入,运算结果串行输出,参加运算的操作数从最低位开始逐位相加至最高位,最后得出和数。串行加法器电路简单、易于实现,但运算速度太慢。为了提高加法运算的速度,可采用按并行方式运算的**并行加法器**,即能并行提供二进制加数和被加数的每一位并将各位同时相加的加法器。图3.21所示为4位并行加法器逻辑符号,图中A_1,A_2,A_3,A_4为二进制被加数输入端;B_1,B_2,B_3,B_4为二进制加数输入端;C_0为低位来的进位输入端;C_4为向高位的进位输出端;S_1,S_2,S_3,S_4为和数输出端。按进位位传递方式的不同,并行加法器又可分为串行进位并行加法器和超前进位并行加法器两种。

1) 串行进位并行加法器

串行进位并行加法器是由一位全加器级联而成的,全加器的个数等于二进制数的位数,如图3.22所示。串行进位加法器的特点是:被加数和加数的各位能同时并行地到达各自的输入端,而各位全加器的进位输入仍是按照由低向高逐级串行传送的,形成一个进位链,这种进位连接方式称为**行波进位**(Carry-Ripple)或**串行进位**。由于每一位相加的和都与本位的进位输入有关,因此,最高位必须等到各低位全部相加完毕并送来进位信号之后才能产生运算结果。显然这种加法器的运算速度仍较慢,而且位数越多,速度就越慢。

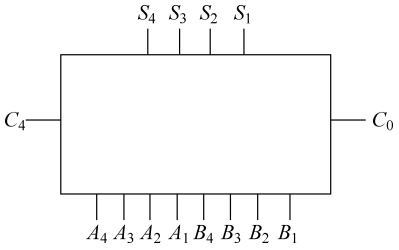


图3.21 4位并行加法器逻辑符号

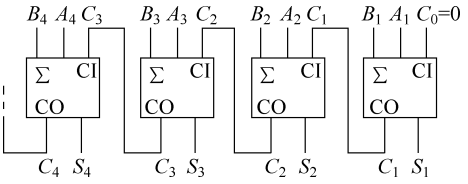


图3.22 串行进位并行加法器逻辑图

2) 超前进位并行加法器

由图3.22可以看出,进位信号C_1是A_1、B_1和C_0的函数,进位信号C_2是A_2、B_2和C_1的函数,也就是A_2、B_2、A_1、B_1和C_0的函数,以此类推,所有的进位信号都是输入端的函数,在并行加法器中,输入信号同时产生,因此进位信号也可以同时生成,这种同时形成各位进位的方法称为**超前进位**(Carry Look Ahead,CLA)或**先行进位**、**并行进位**。超前进位并行加法器使各位的进位信息同时形成,而不需要依赖低位来的进位,克服了由于行波进位造成的延时等待,提高了运算速度。可根据全加器的输出表达式推出超前进位的原理。

根据全加器进位表达式 $C_i=A_iB_i+(A_i\oplus B_i)C_{i-1}$ 可知，A_iB_i 取决于本位参加运算的两个数，而与低位进位无关，因此称 A_iB_i 为进位产生函数，可记作 G_i；$(A_i\oplus B_i)C_{i-1}$ 不仅与本位的两个数有关，还依赖于低位送来的进位，因此称 $A_i\oplus B_i$ 为进位传递函数，记作 P_i，则 $C_i=G_i+P_iC_{i-1}$。

以 4 位并行加法器为例，每一位的进位表达式可表示为

$$C_1=G_1+P_1C_0$$
$$C_2=G_2+P_2C_1$$
$$C_3=G_3+P_3C_2$$
$$C_4=G_4+P_4C_3$$

将以上各式逐项代入，都用最低进位 C_0 表示，可得

$$C_1=G_1+P_1C_0$$
$$C_2=G_2+P_2C_1=G_2+P_2G_1+P_2P_1C_0$$
$$C_3=G_3+P_3C_2=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1C_0$$
$$C_4=G_4+P_4C_3=G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1+P_4P_3P_2P_1C_0$$

从上式可得 4 位超前进位链的线路图如图 3.23 所示，4 位进位输出信号仅由 G_i、P_i 及最低位 C_0 决定，而 $G_i=A_iB_i$，$P_i=A_i\oplus B_i$，当 A、B 各位同时到来时，经过三级门的延时（第一级生产各 G_i 和 P_i，第二级为各 G_i 和 P_i 的与级，第三级为它们的或级），各位进位 C_i 同时产生。

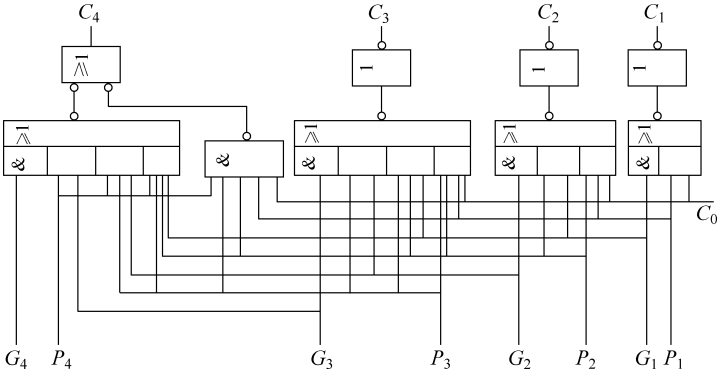


图 3.23　4 位超前进位链线路图

设与或非门的级延迟时间为 $1.5t$，与非门的级延迟时间为 $1t$，则当 G_i 和 P_i 形成后，只需 $2.5t$ 就可以产生全部进位。

图 3.24 是由超前进位链构成的 4 位并行加法器的逻辑电路图，参加运算的两个操作数分别为 $A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$，运算结果为 $S_3S_2S_1S_0$，C_0 为低位的进位输入，C_4 为向高位的进位输出。由图 3.24 可以看出，输入两个参加运算的操作数以后，运算结果可以同时产生，运算速度比行波进位加法器的速度快。但当位数增加时，进位形成逻辑的输入变量增多，进位信号 C_i 的逻辑表达式也会变得越来越复杂，以致超出器件规定的扇入系数。因此，当位数增多时，用这种方法是不现实的。一般的做法是将加法器分组，常用的分组进位结构有组内并行、组间串行的进位链和组内并行、组间并行的进位链两种。

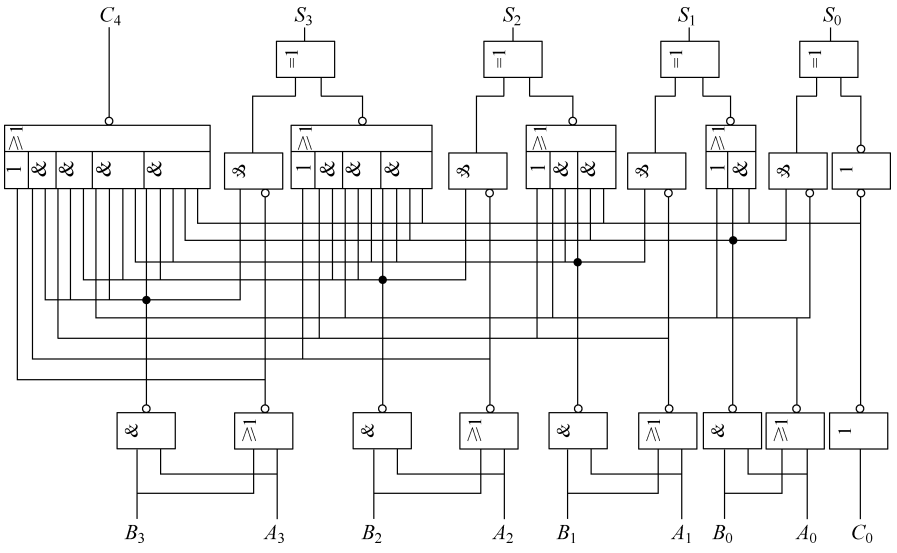


图 3.24　由超前进位链构成的 4 位并行加法器

3.3.2　译码器

译码器是计算机中最常用的逻辑部件之一，用来对输入信号进行“翻译”，识别出其含义并产生相应的输出信号。译码器的种类很多，常见的中规模译码器有二进制译码器、二-十进制译码器和七段显示译码器等几类。

1. 二进制译码器

二进制译码器的功能是将 n 个输入变量“翻译”成 2^n 个输出函数，且每个输出函数对应于输入变量的一个最小项。因此，二进制译码器都具有 n 个译码输入端和 2^n 个译码输出端，称 $n-2^n$ 线译码器。除此之外，一般还有用于控制该译码器是否能够正常工作的一个或多个控制输入端，通常称为使能输入端或使能端。当所有使能输入端都为有效电平时，对应每一组输入代码，输出端仅有一个为有效电平，其余均为无效电平。当使能端不全部有效时，输出端均为无效电平。有效电平既可以为高电平也可以为低电平，分别称为高电平译码器和低电平译码器。

表 3.10 为 3-8 线二进制译码器的真值表，A，B，C 为译码输入端；$\overline{Y}_0$，$\overline{Y}_1$，…，$\overline{Y}_7$ 为译码输出端；En 为使能输入端。由真值表可知，当使能信号为有效电平高电平时，对于 A，B，C 的一组输入值，输出 $\overline{Y}_0$，$\overline{Y}_1$，…，$\overline{Y}_7$ 中都有且仅有一个为 0(低电平有效)，其余均为 1，此即译码。其他情况译码器不工作，输出端全为 1(无效电平)。

表 3.10　3-8 线译码器的真值表

使能输入	译码输入			输　　出							
En	C	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1

续表

使能输入	译码输入			输　出							
En	C	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
1	0	1	0	1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	0	1	1	1	1
1	1	0	0	1	1	1	1	0	1	1	1
1	1	0	1	1	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	0

2. 二-十进制译码器

二-十进制译码器的功能是将 4 位 BCD 码(一般为 8421 BCD 码)的 10 组代码“翻译”成 10 个对应信号输出的逻辑电路。因为其输入和输出线的条数分别为 4 和 10,所以又称为 4-10 线译码器。

二-十进制译码器的真值表如表 3.11 所示,表中 A_3、A_2、A_1、A_0 为输入端,$\overline{Y}_0 \sim \overline{Y}_9$ 为输出端,输出端低电平有效,由表 3.11 可知,$A_3A_2A_1A_0$ 输入为 8421 BCD 码,只有一个输出端为有效电平,代表相应的十进制数码。

表 3.11　二-十进制译码器的真值表

序号	输入				输出									
	A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0

3. 七段显示译码器

七段显示译码器的作用是将 BCD 码表示的十进制数转换为七段 LED 显示器的 7 个驱动输入端,图 3.25 所示为七段显示译码器和七段 LED 显示器连接图,其中 B_8,B_4,B_2,B_1 为 BCD 码输入端,输出 $a \sim g$ 为七段 LED 显示器的 7 个驱动输入端。七段 LED 显示器由 7 个条形发光二极管(LED)组成,不同段 LED 的亮、灭组合,即可显示不同的数字。LED 显示器有共阳极和共阴极连接两种形式,共阳极形式是将 7 个 LED 的阳极连在一起并接高电平,当

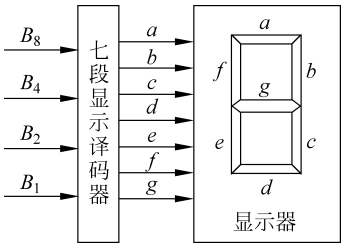


图 3.25　七段显示译码器和七段 LED 显示器连接图

需要某段 LED 点亮时,就让其阴极接低电平即可,否则接高电平。共阴极形式和共阳极形式相反,将 7 个 LED 的阴极连在一起并接低电平,当需要某段 LED 点亮时,就让其阳极接高电平即可,否则接低电平。表 3.12 为共阴极 LED 显示器的 BCD 码七段显示译码器真值表。

表 3.12　共阴极 LED 显示器的 BCD 码七段显示译码器真值表

	B_8	B_4	B_2	B_1	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*
11	1	0	1	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*
12	1	1	0	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*
13	1	1	0	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*
14	1	1	1	0	*d*	*d*	*d*	*d*	*d*	*d*	*d*
15	1	1	1	1	*d*	*d*	*d*	*d*	*d*	*d*	*d*

3.3.3　多路选择器

多路选择器又叫数据选择器或多路开关,简称 MUX(Multiplexer)。它是一种多输入单输出的组合逻辑电路,完成从多路输入数据中选择一路送至输出端的功能。多路数据的选择是受控制信号控制的,一般称为地址选择信号。通常,对于一个具有 2^n 路输入和一路输出的 MUX 有 n 个地址选择端,它的每一种取值组合都对应选中一路输入送至输出端。

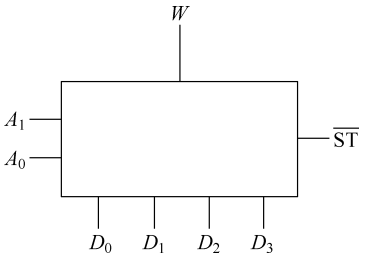


图 3.26　4 路选择器的逻辑符号

图 3.26 为 4 路选择器的逻辑符号,$D_0 \sim D_3$ 为数据输入端;A_1,A_0 为地址选择输入端;$\overline{\text{ST}}$ 为使能输入端(低电平有效);W 为数据输出端。表 3.13 为 4 路选择器的真值表,可以看出,在使能端有效的情况下,根据地址端的不同组合,分别从输入数据中选择一路输出。

表 3.13　4 路选择器的真值表

选择输入		数据输入				使能输入	输出
A_1	A_0	D_0	D_1	D_2	D_3	ST	W
×	×	×	×	×	×	1	0
0	0	D_0	×	×	×	0	D_0
0	1	×	D_1	×	×	0	D_1
1	0	×	×	D_2	×	0	D_2
1	1	×	×	×	D_3	0	D_3

3.3.4 三态门

三态门是具有三个输出状态的逻辑电路，三态门的输出端除了 0 和 1 两种状态外，还有第三种状态，称为高阻状态或禁止状态，在这种状态下，输出端相当于断开(虚断)。

图 3.27 为高电平有效的三态门逻辑符号，其输出 Y 受使能端 E 的控制，当 E 为有效电平时输出状态为正常的 0 或 1，当 E 为无效电平时输出为高阻态，功能表如表 3.14 所示，其中 Z 表示高阻态。

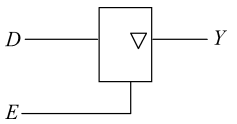


图 3.27 高电平有效的三态门逻辑符号

表 3.14 高电平有效的三态门功能表

E	D	Y
0	×	Z
1	0	0
1	1	1

三态门主要应用于总线传送，可以分为单向或双向数据传送。

图 3.28 所示是用三态门构成的单向数据总线。在任何时刻，n 个三态门中只允许其中一个使能端 E_i 有效，该门处于工作状态，相应的数据 D_i 就被送到总线上去，而其他门的使能端均无效，处于高阻状态。若某一时刻同时有两个以上的三态门使能端有效，就有两个以上的三态门工作，那么总线上传输的信息就会出错。

图 3.29 所示是具有双向三态缓冲的数据总线。当控制使能端 EN 为高电平时，三态门 G_1 处于工作状态，三态门 G_2 处于高阻状态，数据 D_1 经 G_1 传输到总线上；当控制使能端 EN 为低电平时，三态门 G_2 处于工作状态，三态门 G_1 处于高阻状态，总线上的数据 D_2 通过三态门 G_2 接收。这样，通过改变控制使能端 EN 的状态，实现数据的分时双向传送。

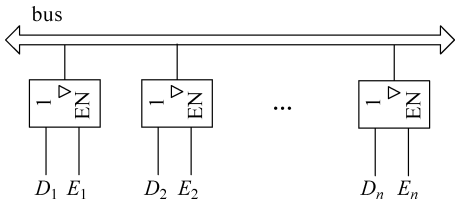


图 3.28 三态门构成的单向传输

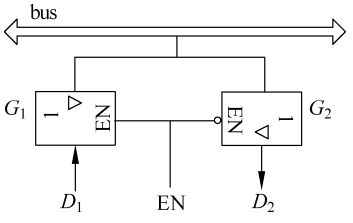


图 3.29 双向三态缓冲

3.4 计算机中常用的时序逻辑电路

时序逻辑电路由组合逻辑和存储器件两部分构成，电路的输出不仅与当前输入有关，还与以前输入有关。触发器、锁存器、寄存器都可以作为存储器件，用来存储二进制信息。触发器强调的是工作机制，多指边沿触发；锁存器强调的是它的功能特性，多指电平触发。触发器、锁存器是逻辑电路的术语，而寄存器通常是 n 位触发器的组合，是计算机的术语。

3.4.1 基本 R-S 触发器和 D 触发器

1. 基本 R-S 触发器

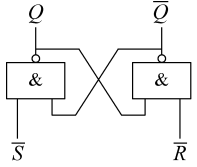
最基本的触发器是基本 R-S 触发器,它可以由两个与非门(或者两个或非门)交叉连接而成,有两个稳定的输出状态,可用来保存1位二进制信息。图3.30为基本 R-S 触发器的逻辑图、功能表和时序图,由图可以看出,基本 R-S 触发器的功能为:

(1) 当输入 $\overline{R}\overline{S}=11$ 时,触发器保持原来的稳定状态;

(2) 当输入 $\overline{R}\overline{S}=10$ 时,其 Q 端置1,但此期间输入不能发生变化;

(3) 当输入 $\overline{R}\overline{S}=01$ 时,其 Q 端清0,但此期间输入不能发生变化;

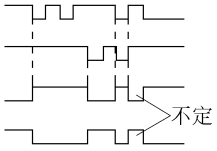
(4) 当输入 $\overline{R}\overline{S}=00$ 时,其 Q 端与 $\overline{Q}$ 端都为1,但此期间输入信号同时变为11时,如果左边的与非门先清0,Q 端就输出为0,反之输出为1,这取决于与非门的延迟时间。由于究竟哪个门快,使用者无法把握,因此称这样的输出为不确定,基本 R-S 触发器不允许出现这种状态。



(a) 逻辑图

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	功能描述
1	1	不变		保持
1	0	0	1	复位
0	1	1	0	置位
0	0	不定		失效

(b) 功能表



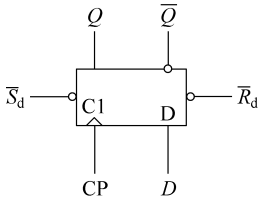
(c) 时序图

图3.30 基本 R-S 触发器

由此可以看出,基本 R-S 触发器不允许两个输入端同时为0,接收数据期间也不允许输入端发生变化,也没有时钟控制。

2. D 触发器

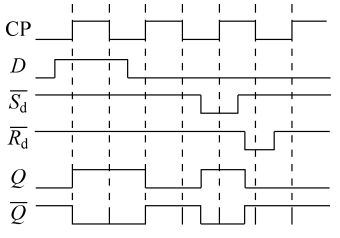
在基本 R-S 触发器的基础上再增加一定结构的与非门即可组成比较完善的 D 触发器。D 触发器常用来构成寄存器、数据缓冲器和计数器等。图3.31所示为上升沿 D 触发器的逻辑符号、功能表和波形图,其中,CP表示时钟脉冲输入信号,D 为触发器的数据输入端,Q 为触发器的状态输出端,Q^n 表示现态,Q^{n+1} 表示次态,$\overline{R}_d$ 和 $\overline{S}_d$ 分别是直接复位端和直接置位端。



(a) 逻辑图

$\overline{S}_d$	$\overline{R}_d$	CP	D	Q^n	Q^{n+1}
0	1	×	×	×	1
1	0	×	×	×	0
1	1	↑	1	×	1
1	1	↑	0	×	0

(b) 功能表



(c) 波形图

图3.31 D 触发器

由功能表可以看出:

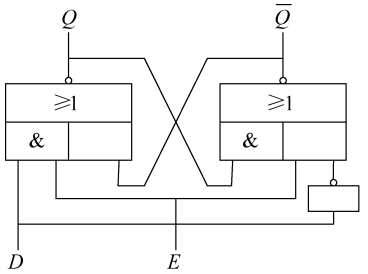
(1) 复位信号和置位信号不受时钟信号控制,不管时钟信号如何,只要复位信号或置位信号有效,立即完成触发器的复位或置位;

(2) 当复位信号和置位信号无效时，在时钟信号 CP 的上升沿，输入端 D 的数据传送到输出端 Q。

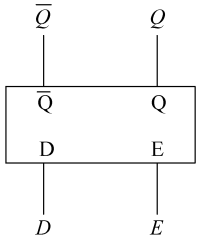
3.4.2 锁存器

锁存器是由若干个触发器构成的能存储多位二进制代码的时序逻辑电路。锁存器的输出受锁存信号(也称使能信号)的控制，当锁存信号有效时，输出随着输入变化；当锁存信号无效时，输出状态不变，此时输出对于输入是透明的，因此锁存器也称为透明锁存器。

图 3.32(a)和图 3.32(b)是在基本 R-S 触发器的基础上增加三个门电路构成的一位锁存器逻辑图和图形符号，D 为输入信号，E 为锁存信号，高电平有效，Q 为锁存器的输出状态，图 3.32(c)为该锁存器功能表。



(a) 逻辑图



(b) 图形符号

E	D	Q	$\overline{Q}$
1	0	0	1
1	1	1	0
0	×	不变	不变

(c) 功能表

图 3.32　一位锁存器

由功能表可以看出，当锁存信号 E 有效时，能锁存输入数据 D；当 E 无效时，锁存器输出状态不变，与输入无关。

3.4.3 数据寄存器

寄存器是一种常见的时序逻辑电路，常常用来存放数据、指令等。寄存器是由具有存储功能的触发器构成的，一个触发器存储一位二进制信息，用 n 个触发器组成的寄存器能存储 n 位二进制信息。

图 3.33 所示为 4 个边沿 D 触发器构成的寄存器的逻辑图。D_3 到 D_0 是并行数据输入端，$\overline{CR}$ 是异步清 0 端，CP 是时钟脉冲输入端，Q_3 到 Q_0 是并行数据输出端。功能表如表 3.15 所示。由表可知，当清 0 端 $\overline{CR}$ 有效时，4 个 D 触发器清 0，即 $Q_3Q_2Q_1Q_0=0000$；当清 0 端无效，CP 的上升沿时，触发器被置数，即 $Q_n=D_n$；当清 0 端无效，CP 无上升沿时，触发器状态保持不变。

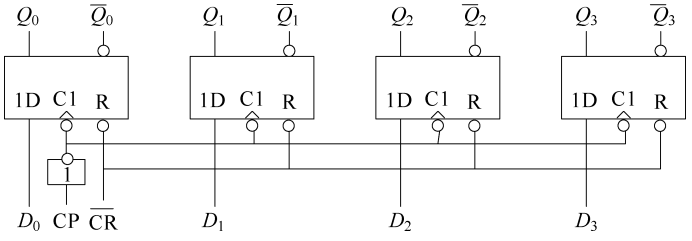


图 3.33　4 个边沿 D 触发器构成的寄存器逻辑图

表 3.15　4 个边沿 D 触发器构成的寄存器功能表

输　入						输　出			
$\overline{CR}$	CP	D_3	D_2	D_1	D_0	Q_3	Q_2	Q_1	Q_0
0	×	×	×	×	×	0	0	0	0
1	↑	D_3	D_2	D_1	D_0	D_3	D_2	D_1	D_0
1	0	×	×	×	×	不变			

3.4.4　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,即指寄存器中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。按照移位方式不同,分为单向移位寄存器和双向移位寄存器两类。

图 3.34 是由边沿 D 触发器组成的单向右移的 4 位移位寄存器,其中触发器 FF_3 的输入端接收串行输入信号,其余的每个触发器输入端均与前边一个触发器的输出端相连。当 CP 的上升沿同时作用所有触发器时,加到寄存器输入端 D_i 的代码存入 FF_3,其余触发器 FF_i 的状态为前一个触发器 FF_{i+1} 的状态,即总的效果是将寄存器里原有代码右移了一位。

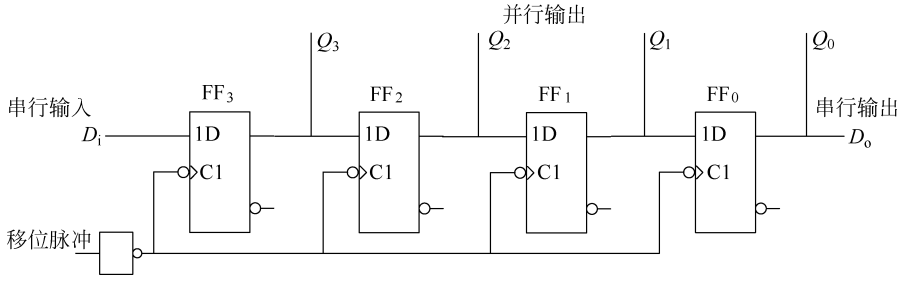


图 3.34　用 D 触发器构成的移位寄存器

3.4.5　计数器

计数器是计算机、数字仪表中一种常用的逻辑电路,它不仅能用来对时钟脉冲计数,还可以用于分频、定时、产生节拍脉冲、进行数字运算等。计数器种类繁多,根据计数器中数字的编码方式的不同,分为二进制计数器、二-十进制计数器和循环码计数器等;根据计数过程中数字的增减趋势的不同分为加法计数器、减法计数器和加/减计数器;根据时钟作用方式不同分为同步计数器和异步计数器等。

图 3.35 所示是 4 位二进制同步加法计数器,由 4 个 JK 触发器和门电路组成,CP 为计数脉冲,C_o 为进位输出。状态表如表 3.16 所示。由表可以看出,每来一个计数脉冲 CP,计数器的状态加 1,当计数到最大值时,送出进位信号 C_o,下一个计数脉冲 CP 到来时,计数器和 C_o 清 0,计数器又从 0 开始计数。

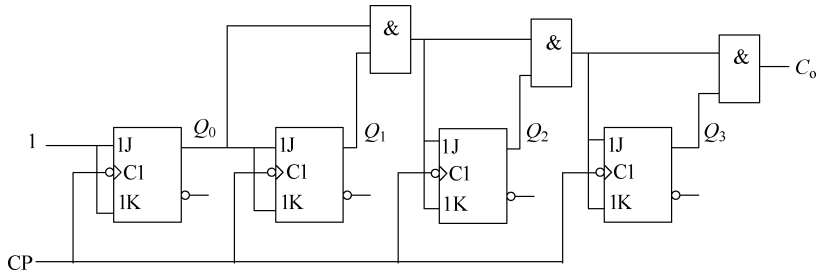


图 3.35　由 JK 触发器组成的 4 位二进制同步加法器

表 3.16　4 位二进制同步加法计数器状态表

CP	Q_3	Q_2	Q_1	Q_0	C_o
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1
16	0	0	0	0	0

习　　题

3.1　用逻辑代数的公式、定理和规则将下列逻辑函数化简为最简与或表达式。

(1) $F=AB+\overline{A}\ \overline{B}\ C+BC$

(2) $F=A\overline{B}+B+BCD$

(3) $F=(A+B+C)(\overline{A}+B)(A+B+\overline{C})$

(4) $F=BC+D+\overline{D}(\overline{B}+\overline{C})(AC+B)$

3.2　用卡诺图化简法求出下列逻辑函数的最简与或表达式。

(1) $F(A,B,C,D)=\overline{A}\overline{B}+\overline{A}\overline{C}D+AC+B\overline{C}$

(2) $F(A,B,C,D)=BC+D+\overline{D}(\overline{B}+\overline{C})(AD+B)$

(3) $F(A,B,C,D)=\prod M(2,4,6,10,11,12,13,14,15)$

(4) $F(A,B,C,D)=\sum m(0,2,7,13,15)+\sum d(1,3,4,5,6,8,10)$

3.3 比较 PLD 和 FPGA 的异同。

3.4 举例说明三态门主要有哪些应用。

3.5 分析超前进位并行加法器能提高运算速度的原因。

3.6 用一个四选一数据选择器设计一判定电路。该电路的输入为 8421 BCD 码,当输入的数字大于 1 且小于 6 时输出为 1,否则为 0。

3.7 用 4 位二进制并行加法器设计一个余三码表示的 1 位十进制加法器。

3.8 图 3.36 所示电路是由两个同步十进制计数器 74160 组成的计数器。试分析这是多少进制的计数器,两个之间是几进制。

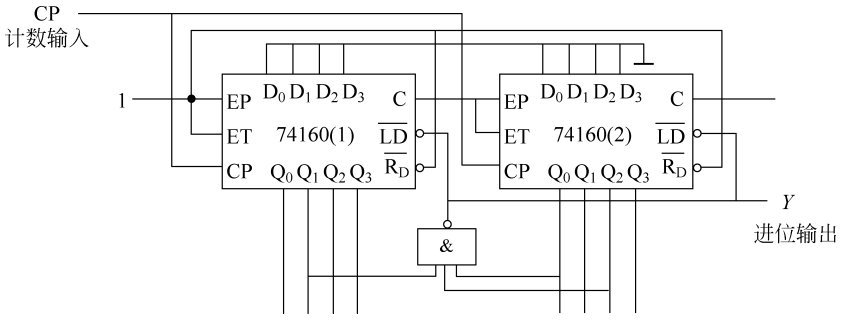


图 3.36 题 3.8 图

3.9 图 3.37 所示电路中的每一方框均为低电平有效的 2-4 线译码器。使能端为低电平有效。要求:

(1) 写出电路工作时 $\overline{F}_{10}$,$\overline{F}_{20}$,$\overline{F}_{30}$,$\overline{F}_{40}$的逻辑表达式。

(2) 说出电路的逻辑功能。

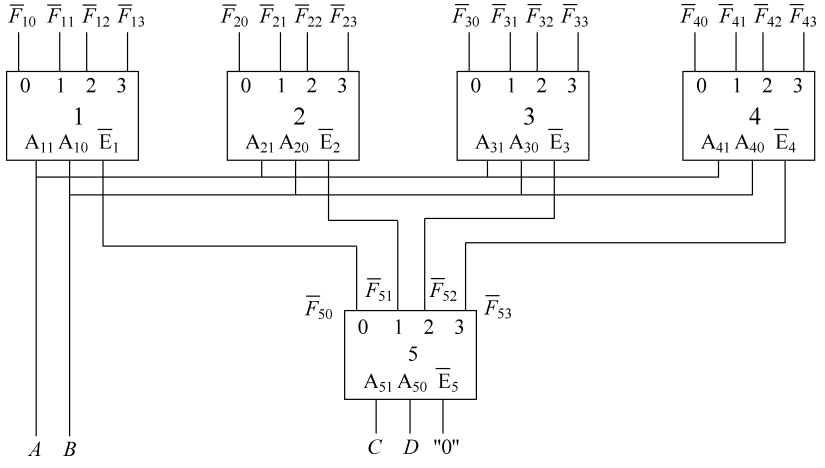


图 3.37 题 3.9 图

3.10 用四选一数据选择器和门电路设计下列十进制代码转换器。

(1) 8421 BCD 码转换为余三码。

(2) 余三码转换为 8421 BCD 码。

第4章 运算方法与运算器

运算器是计算机的信息加工处理部件，其基本功能是完成算术运算、逻辑运算和移位操作。本章介绍定点数和浮点数的加减乘除运算方法及其硬件实现电路，之后增加逻辑运算电路构成算术逻辑单元，最后介绍运算器数据通路的组织方式。

4.1 定点数加减法运算

4.1.1 原码加减运算方法

原码加减法运算的特点如下：

(1) 符号位不能和数据一起参与运算；

(2) 符号位和加减法指令共同作为运算的依据。

下面讨论原码加减法运算的规则：

(1) 加法指令需执行“同号求和，异号求差”；减法指令需执行“异号求和，同号求差”。

(2) 求和时，将两操作数的数值位相加得到和的数值位。若数值最高位产生进位，则结果溢出。和的符号位采用第一操作数(被加数/被减数)的符号。

(3) 求差时，第一操作数的数值位加上第二操作数(加数/减数)的数值位的补码。分两种情况讨论：

① 最高数值位有进位，表明加法结果为正，所得数值位正确，结果的符号位采用第一操作数的符号。

② 最高数值位无进位，表明加法结果为负(补码形式)，应对其求补，还原为绝对值形式的数值位，结果的符号位为第一操作数的符号变反。

例 4.1 已知$[X]_{原}=0.1101$，$[Y]_{原}=1.1001$，求$[X+Y]_{原}$和$[X-Y]_{原}$。

解：先求$[X+Y]_{原}$，依运算规则，对两数求差，即 .1101－.1001＝.1101＋.0111＝1.0100。

因为最高数值位产生进位，所得数值位.0100 正确，再加上第一操作数的符号位 0，则

$$[X+Y]_{原}=0.0100$$

再求$[X-Y]_{原}$，依运算规则，应对两数求和，即 .1101＋.1001＝1.0110。

因为数值最高位产生进位，即结果溢出。

可以看出，原码加减运算规则比较复杂，下面讨论计算机中常用的补码加减运算的方法。