第 1 章　逻辑代数基础

逻辑代数是分析数字逻辑电路的数学工具，也是进行逻辑设计的理论基础。本章从逻辑变量和逻辑运算的概念出发，主要介绍逻辑代数的公式、规则和定理，逻辑函数及其表示方法，逻辑函数的化简方法和变换方法。

1849 年，英国数学家乔治·布尔（George Boole）首先提出了描述客观事物关系的数学方法——布尔代数。后来，由于布尔代数被广泛地应用于开关电路和数字逻辑电路的分析和设计上，所以也把布尔代数叫做开关代数或逻辑代数。逻辑代数是分析和设计数字逻辑电路的数学工具。

1.1　逻辑变量和逻辑运算

1.1.1　逻辑变量

逻辑变量用于描述客观事物对立统一的两个方面。逻辑代数中通常用单个大写字母或单个大写字母加下标表示逻辑变量。在二值逻辑中，每个逻辑变量的取值只有 0 和 1 两种可能，这里 0 和 1 已不再表示数量的大小，只代表两种不同的逻辑状态，如电平的高和低、电流的有和无、灯的亮和灭、开关的闭合和断开等。

1.1.2　基本逻辑运算

逻辑代数中有三种最基本的逻辑运算：与、或、非，也称为逻辑与、逻辑或和逻辑非。

1．与运算

逻辑与（逻辑乘）表示这样一种逻辑关系：只有决定事物结果的全部条件同时具备时，结果才发生。例如图 1-1 所示的电路是一个简单的与逻辑关系，灯 Y 受两个串联开关 A、B 的控制，仅当开关 A 与 B 同时闭合时，灯 Y 才亮，否则灯灭。

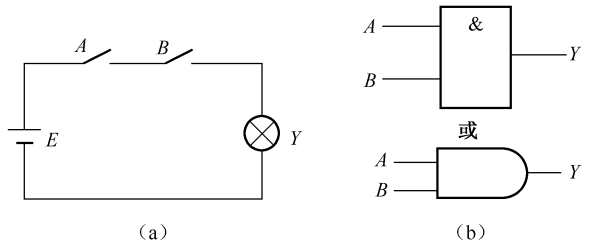


图 1-1　与逻辑电路示例及其符号

现用“1”来表示开关“闭合”及灯“亮”；用“0”表示开关“断开”及灯“灭”。那么可以列出表 1-1 所示的逻辑真值表。所谓逻辑真值表是指把逻辑变量所有可能的取值组合及其对应结果列成一种表格，可简称为真值表。

表 1-1　与逻辑运算真值表

A	*B*	*Y*
0	0	0
0	1	0
1	0	0
1	1	1

在逻辑代数中，上述与逻辑关系还可以表示成如下的逻辑函数式：$Y = A \cdot B$，式中“·”为“与”的逻辑运算符号，也称为逻辑乘，使用时也可将其省略，写成 $Y = AB$。

在逻辑电路中，能实现与运算逻辑功能的电路称为与门，图 1-1（b）所示为与门的国标符号和国际常用符号。

2．或运算

逻辑或（逻辑加）表示这样一种逻辑关系：在决定事物结果的诸多条件中只要任何一个满足，结果就会发生。例如图 1-2 所示的电路是一个简单的或逻辑关系，灯 *Y* 受两个并联开关 *A*、*B* 的控制，只要 *A*、*B* 中任何一个开关闭合时，灯 *Y* 便亮。

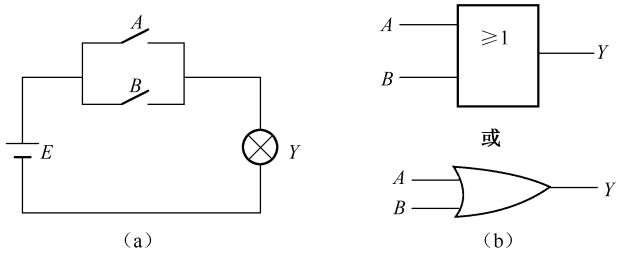


图 1-2　或逻辑电路示例及其符号

或逻辑运算真值表如表 1-2 所示，也可以写出如下的或逻辑函数式：$Y = A + B$，式中“+”为“或”的逻辑运算符号。

表 1-2　或逻辑运算真值表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

在逻辑电路中，能实现或运算逻辑功能的电路称为或门，图 1-2（b）为或门的国标符号和国际常用符号。

3．非运算

逻辑非（逻辑求反）表示这样一种逻辑关系：只要条件具备了，结果便不会发生；而条件不具备时，结果一定发生。例如，图 1-3（a）所示的电路是一个简单的非逻辑关系，灯 Y 受开关 A 的控制，当开关 A 接通时，灯 Y 不亮；当开关 A 断开时，灯 Y 反而亮。

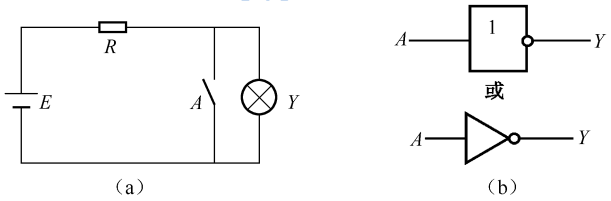


图 1-3　非逻辑电路示例及其符号

非逻辑运算的真值表如表 1-3 所示，也可以写出如下的非逻辑函数式：$Y = \overline{A}$。通常 A 称为原变量，$\overline{A}$ 称为反变量。

表 1-3　非逻辑运算真值表

A	Y
0	1
1	0

在逻辑电路中，能实现非运算逻辑功能的电路称为非门（也叫反相器），图 1-3（b）为非门的国标符号和国际常用符号。

1.1.3 复合逻辑运算

实际的逻辑问题往往比与、或、非基本逻辑复杂，不过它们都可以用与、或、非组合成的复合逻辑来实现。最常见的复合逻辑运算有与非、或非、与或非、异或、同或逻辑等。图 1-4 给出了这些复合逻辑运算的图形符号及运算符号，表 1-4 至表 1-8 是它们的真值表。

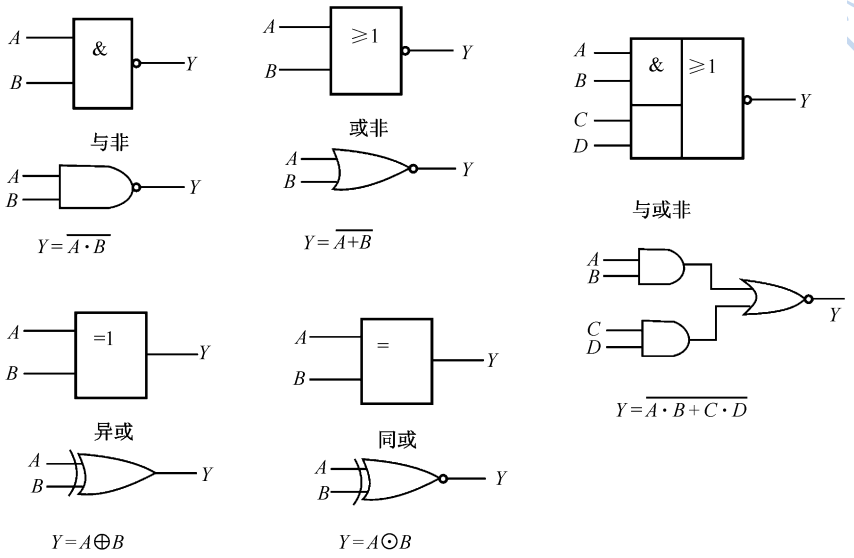


图 1-4 复合逻辑的图形符号和运算符号

表 1-4 与非逻辑的真值表

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

表 1-5 或非逻辑的真值表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

表 1-6 与或非逻辑的真值表

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1

续表

A	B	C	D	Y
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

表 1-7　异或逻辑的真值表

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

表 1-8　同或逻辑的真值表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

在与非逻辑中，将 A、B 先进行与运算，然后将结果求反，最后得到的即 A、B 的与非运算结果，因此与非运算看做是与运算和非运算的组合。图 1-4 中图形符号上的小圆圈表示非运算。同样，或非逻辑是或运算和非运算的组合。

在与或非逻辑中，A、B 之间以及 C、D 之间都首先是与的关系，然后把它们与运算的结果进行或运算，最后再进行非运算。因此，只要 A、B 或 C、D 任何一组同时为 1，Y 就是 0；只有当每一组输入都不全是 1 时，输出 Y 才是 1。

异或是这样一种逻辑关系：当 A、B 不同时，输出 Y 为 1；当 A、B 相同时，输出 Y 为 0。异或也可以用与、或、非的组合表示：

$$A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$$

同或与异或相反，当 A、B 相同时，输出 Y 等于 1；当 A、B 不同时，输出 Y 等于 0。同或也可以写成与、或、非的组合形式：

$$A \odot B = A \cdot B + \overline{A} \cdot \overline{B}$$

而且，由表 1-7 和表 1-8 可知，异或和同或互为反运算，即：

$$A \oplus B = \overline{A \odot B}; \quad \overline{A \oplus B} = A \odot B$$

1.2 逻辑代数的公式和定理

1.2.1 逻辑代数的基本公式和常用公式

1. 基本公式

逻辑代数的基本公式也叫布尔恒等式，它们反映了逻辑代数运算的基本规律，其正确性都可以用列逻辑真值表的方法加以验证。

（1）常量与变量关系公式

$$A \cdot 1 = A,\ A \cdot 0 = 0,\ A + 1 = 1,\ A + 0 = A,\ A \cdot \overline{A} = 0$$

（2）变量之间关系公式

交换律：$A \cdot B = B \cdot A$，$A + B = B + A$，$A + \overline{A} = 1$

结合律：$(A \cdot B) \cdot C = A \cdot (B \cdot C)$，$(A + B) + C = A + (B + C)$

分配律：$A \cdot (B + C) = A \cdot B + A \cdot C$，$A + B \cdot C = (A + B) \cdot (A + C)$

互补律：$A + \overline{A} = 1$，$A \cdot \overline{A} = 0$

重叠律：$A + A = A$，$A \cdot A = A$

还原律：$\overline{\overline{A}} = A$

反演律（德·摩根定律）：$\overline{AB} = \overline{A} + \overline{B}$，$\overline{A + B} = \overline{A} \cdot \overline{B}$

2. 常用公式

逻辑代数的常用公式是利用基本公式导出的，在逻辑函数的化简中可直接运用。

（1）$A + A \cdot B = A$，$A \cdot (A + B) = A$

证：$A + A \cdot B = A(1 + B) = A$

可见，在两个乘积项相加时，若其中一项以另一项为因子，则该项是多余的，可以删去。

（2）$A + \overline{A} \cdot B = A + B$

证：$A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B) = A + B$

可见，在两个乘积项相加时，若一项取反后是另一项的因子，则此因子是多余的，可以消去。

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

证：$A \cdot B + A \cdot \overline{B} = A(B + \overline{B}) = A \cdot 1 = A$

可见，当两个乘积项相加时，若它们分别含有 B 和 $\overline{B}$ 两个因子而其他因子相同，则两项定能合并，且可将 B 和 $\overline{B}$ 两个因子消去。

（4）$A\cdot(A+B)=A$

证： $A\cdot(A+B)=A\cdot A+A\cdot B=A+A\cdot B=A\cdot(1+B)=A\cdot 1=A$

可见，变量 A 与包含变量 A 的或式相乘时，其结果等于 A，即可以将或式消去。

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

证：
$$\begin{aligned}A\cdot B+\overline{A}\cdot C+B\cdot C&=A\cdot B+\overline{A}\cdot C+(A+\overline{A})B\cdot C\\&=A\cdot B+\overline{A}\cdot C+A\cdot B\cdot C+\overline{A}\cdot B\cdot C\\&=A\cdot B(1+C)+\overline{A}\cdot C(1+B)\\&=A\cdot B+\overline{A}\cdot C\end{aligned}$$

可见，若两个乘积项中分别包含有因子 A 和 $\overline{A}$，而这两个乘积项的其他因子都是第三个乘积项（可含其他因子）的因子时，则第三个乘积项是多余的，可以消去。

（6）$A\cdot\overline{A\cdot B}=A\cdot\overline{B}$；　$\overline{A}\cdot\overline{A\cdot B}=\overline{A}$

证：① $A\cdot\overline{A\cdot B}=A\cdot(\overline{A}+\overline{B})=A\cdot\overline{A}+A\cdot\overline{B}=A\cdot\overline{B}$

上式说明，当 A 和一个乘积项的非相乘，且 A 为该乘积项的因子时，则 A 这个因子可以消去。

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

上式说明，当 $\overline{A}$ 和一个乘积项的非相乘，且 A 为该乘积项的因子时，其结果就等于 $\overline{A}$。

1.2.2　逻辑代数的基本定理

1．代入定理

在任何一个含有某个变量的等式中，若用另外一个逻辑式代入式中所有这个变量的位置，等式仍然成立，这就是所谓的代入定理。

因为任何一个逻辑式和逻辑变量一样，只有 0 和 1 两种可能取值，所以用一个函数式取代某个变量，等式自然成立。

代入定理可将逻辑代数中的基本公式和常用公式推广为多变量的形式。

例 1.1　用代入定理证明德 • 摩根定理的多变量情况。

解　已知二变量的德 • 摩根定理为

$$\overline{A\cdot B}=\overline{A}+\overline{B}，\quad \overline{A+B}=\overline{A}\cdot\overline{B}$$

现以$(B\cdot C)$代入左边等式中B的位置，同时以$(B+C)$代入右边等式中B的位置，可得到

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

2．反演定理

对于任意一个逻辑式Y，若将其中所有的“·”换成“+”，“+”换成“·”，0换成1，1换成0，原变量换成反变量，反变量换成原变量，则得到的结果就是$\overline{Y}$，这就是所谓的反演定理。

反演定理为求取已知逻辑式的反逻辑式提供了方便。在使用反演定理时还需注意遵循以下两个规则：

（1）仍需遵守原逻辑式“先括号、然后乘、最后加”的运算优先次序；

（2）不属于单个变量上的非号应保留不变。

例 1.2 已知$Y_1=A\cdot\overline{B}+\overline{C}\cdot D$，$Y_2=AB+\overline{C+\overline{DB}+\overline{BC}}$，求$\overline{Y_1}$和$\overline{Y_2}$。

解 根据反演定理可写出：

$$\overline{Y_1}=(\overline{A}+B)\cdot(C+\overline{D})$$
$$\overline{Y_2}=(\overline{A}+\overline{B})\cdot\overline{\overline{C}\cdot(D+\overline{B})}\cdot\overline{\overline{B}+\overline{C}}$$

3．对偶定理

若两逻辑式相等，则它们的对偶式也相等，这就是对偶定理。

所谓对偶式是这样定义的：对于任何一个逻辑式Y，若将其中的“·”换成“+”，“+”换成“·”，0换成1，1换成0，则得到一个新的逻辑式Y'，这个Y'就叫做Y的对偶式，或者说Y和Y'互为对偶式。

利用对偶定理，有时可简化证明：为了证明两个逻辑式相等，可以通过证明它们的对偶式相等来完成。

例 1.3 试证明$A+BC=(A+B)(A+C)$。

解 首先写出等式两边的对偶式，得到$A(B+C)$和$AB+AC$。根据基本公式中的分配律可知，这两个对偶式是相等的，亦即$A(B+C)=\ AB+AC$。由对偶定理即可以确定原来的等式也成立。

1.3 逻辑函数及其表示方式

在实际问题中，往往是用“与”、“或”、“非”这三种逻辑运算符号把有关的逻辑变量连接起来，以构成一定的逻辑关系。如果以代表原因和条件的逻辑变量作为输入，以结果变量作为输出，那么，当输入逻辑变量如A、B、C、…的值确

定后，其输出变量的值也就被唯一地确定了，即 Y 与 A、B、C、…之间构成了函数关系，则称 Y 为 A、B、C、…的逻辑函数，记做

$$Y = F(A,B,C,\cdots)$$

即用一个逻辑函数表达式来表示。由于变量和输出的取值只有 0、1 两种状态，所以我们所讨论的都是二值逻辑函数。

任何一个具体的因果关系都可以用一个逻辑函数来描述。

1.3.1　逻辑函数的表示方法

常用的逻辑函数表示方法有逻辑真值表（简称真值表）、逻辑函数式、逻辑图和卡诺图等。本节只介绍前面三种方法。

1．逻辑真值表

将输入变量所有的取值组合所对应的输出值找出来，列成表格，即可得到真值表。

例如图 1-5 是楼上楼下都可控制的楼梯照明灯电路。单刀双掷开关 A 装在楼上，B 装在楼下。设开关向上合为 1，向下合为 0；灯 Y 亮为 1，灯灭为 0。根据电路的工作原理不难知道，只有 A、B 同时为 1 或同时为 0 时，Y 才等于 1；否则 Y 等于 0。于是可以列出电路的真值表 1-9。

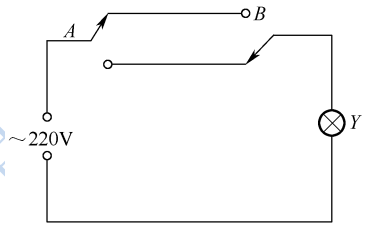


图 1-5　楼梯照明电路

表 1-9　图 1-5 电路的真值表

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

2．逻辑函数式

把输出与输入之间的逻辑关系写成与、或、非等运算的组合式，即得到了该逻辑关系的逻辑函数式。在图 1-5 电路中，灯 Y 的状态（亮与灭）是开关 A、B

状态（向上合与向下合）的函数，根据对电路功能的要求和与、或、非的逻辑定义，“A 和 B 同时向上合，或 A 和 B 同时向下合时，灯 Y 亮；否则灯 Y 灭”，因此得到的输出逻辑函数式为

$$Y = AB + \overline{A}\,\overline{B}$$

3．逻辑图

将逻辑函数中各变量之间的与、或、非关系用图形符号表示出来，就可以画出表示函数关系的逻辑图。

为了画出图 1-5 所示电路的逻辑图，只要用逻辑运算的图形符号代替逻辑函数式中的代数运算符号便可得到图 1-6 所示的逻辑图。

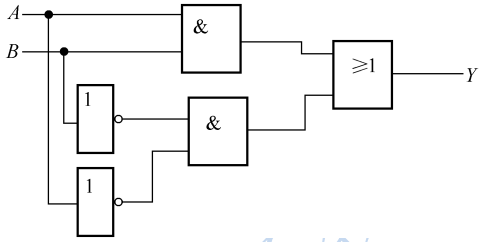


图 1-6　图 1-5 电路的逻辑图

4．各种表示方法间的相互转换

（1）从逻辑函数式列出真值表

将输入变量取值的所有组合状态逐一代入逻辑式求出函数值，列成表，即可得到真值表。

例 1.4　已知逻辑函数式 $Y = \overline{A}BC + AC + \overline{B}C$，求出它对应的真值表。

解　将 A、B、C 的各种取值逐一代入函数式中计算 Y 的值，将计算的结果列表，即得表 1-10 的真值表。

表 1-10　例 1.4 的真值表

A	B	C	$\overline{A}BC$	AC	$\overline{B}C$	Y
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	0	0	0
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	1	0	1	1	1
1	1	0	0	0	0	0
1	1	1	0	1	0	1

（2）从逻辑函数式画出逻辑电路图

将逻辑函数式中所有的与、或、非运算符号用图形符号代替，并依据“先括号，然后乘，最后加”的运算优先顺序把这些图形符号连接起来，就可以画出逻辑图了。

例 1.5　已知逻辑函数 $Y=\overline{\overline{A}BC+A\overline{B}C+AB\overline{C}}$，画出对应的逻辑电路图。

解　将式中所有的与、或、非运算符号用图形符号代替，并依据运算的优先顺序把这些图形符号连接起来，就得到了图 1-7 的逻辑图。

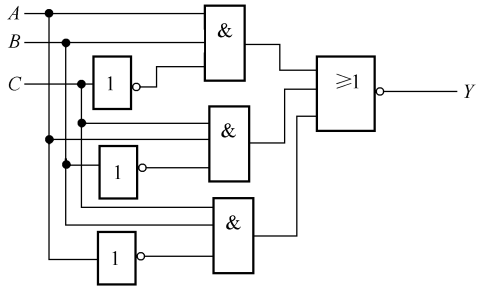


图 1-7　例 1.5 的逻辑图

（3）从逻辑图写出逻辑函数式

从输入端到输出端逐级写出每个图形符号对应的逻辑式，就可以得到最后的逻辑函数式了。

例 1.6　已知函数的逻辑图如图 1-8 所示。试写出它的逻辑函数式。

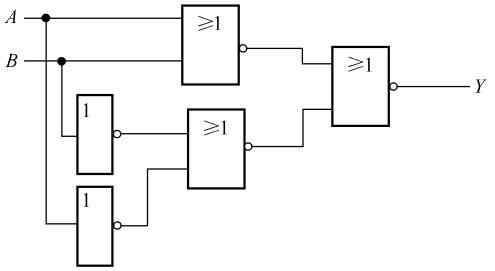


图 1-8　例 1.6 的逻辑图

解　从输入端 A、B 开始逐个写出每个图形符号输出端的逻辑式，即得到

$$Y=\overline{\overline{A+B}+\overline{\overline{A}+\overline{B}}}$$

将该式变换后可得

$$Y=\overline{\overline{A+B}+\overline{\overline{A}+\overline{B}}}=(A+B)(\overline{A}+\overline{B})=\overline{A}B+A\overline{B}=A\oplus B$$

可见，输出 Y 和输入 A、B 之间是异或逻辑关系。

（4）从真值表写出逻辑函数式

例 1.7 已知一个奇偶判别函数的真值表如表 1-11 所示，试写出它的逻辑函数式。

表 1-11 例 1.7 的函数真值表

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1…→ $\bar{A}BC$
1	0	0	0
1	0	1	1…→ $A\bar{B}C$
1	1	0	1…→ $AB\bar{C}$
1	1	1	0

解 由真值表可见，只有当 A、B、C 三个输入变量中两个同时为 1 时，Y 才为 1。因此，在输入变量取值为以下 3 种情况时，Y 将等于 1：

$$A=0,\ B=1,\ C=1$$

$$A=1,\ B=0,\ C=1$$

$$A=1,\ B=1,\ C=0$$

而当 $A=0$、$B=1$、$C=1$ 时，必然使乘积项 $\bar{A}BC=1$；当 $A=1$、$B=0$、$C=1$ 时，必然使乘积项 $A\bar{B}C=1$；而当 $A=1$、$B=1$、$C=0$ 时，必然使乘积项 $AB\bar{C}=1$，因此 Y 的逻辑函数式应当等于这 3 个乘积项之和，即：

$$Y=\bar{A}BC+A\bar{B}C+AB\bar{C}$$

通过例 1.7 可以总结出从真值表写出逻辑函数式的一般方法如下：

①找出真值表中使逻辑函数 $Y=1$ 的那些输入变量取值的组合；

②每组输入变量取值的组合对应一个乘积项，其中取值为 1 的写成原变量，取值为 0 的写成反变量；

③将这些乘积项相加，即得 Y 的逻辑函数式。

1.3.2 逻辑函数的标准形式

逻辑函数的标准形式包括“最小项之和”和“最大项之积”两种形式。它们分别由最小项和最大项构成。

1．最小项的概念及其性质

（1）最小项

在 n 变量的逻辑函数中，若 m 是包含 n 个因子的乘积项，这 n 个变量均以原变量或反变量的形式在 m 中出现一次，且仅出现一次，则称 m 为这组变量的最小项。

在最小项中，变量可以是原变量的形式，也可以是反变量的形式，因此 n 个变量就有 2^n 个最小项。例如 A、B、C 三个变量的最小项有 $\overline{A}\overline{B}\overline{C}$ 、$\overline{A}\overline{B}C$ 、$\overline{A}B\overline{C}$ 、$\overline{A}BC$ 、$A\overline{B}\overline{C}$ 、$A\overline{B}C$ 、$AB\overline{C}$ 、ABC 共 8（2^3）个最小项。

输入变量的每一组取值都使对应的一个最小项逻辑值等于 1。例如在 3 变量 A、B、C 的最小项中，当 $A=1$，$B=1$，$C=0$ 时，使 $AB\overline{C}=1$。如果把 $AB\overline{C}$ 的取值 110 看做一个二进制数，那么它所对应的十进制数就是 6。为了今后使用的方便，将 $AB\overline{C}$ 这个最小项记做 m_6。按照这一约定，依次类推，可列出三变量最小项编号表，如表 1-12 所示。

表 1-12　三变量最小项的编号表

最小项	使最小项为 1 的变量取值			对应的十进制数	编号
	A	B	C		
$\overline{A}\overline{B}\overline{C}$	0	0	0	0	m_0
$\overline{A}\overline{B}C$	0	0	1	1	m_1
$\overline{A}B\overline{C}$	0	1	0	2	m_2
$\overline{A}BC$	0	1	1	3	m_3
$A\overline{B}\overline{C}$	1	0	0	4	m_4
$A\overline{B}C$	1	0	1	5	m_5
$AB\overline{C}$	1	1	0	6	m_6
ABC	1	1	1	7	m_7

（2）最小项的性质

从最小项的定义出发，可以证明最小项具有如下重要性质：

①在输入变量的任何取值下必有一个最小项，而且仅有一个最小项的值为 1。

②全体最小项之和为 1。

③任意两个最小项的乘积为 0。

④具有逻辑相邻性的两个最小项之和可以合并成一项并消去一对因子。

若两个最小项只有一个因子不同，则称这两个最小项具有逻辑相邻性。例如 $AB\overline{C}$ 和 $\overline{A}B\overline{C}$ 两个最小项仅第一个因子不同，所以它们具有逻辑相邻性。将这两

个最小项相加时，定能合并成一项并将那一对不同的因子消去。

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

2．逻辑函数的最小项之和形式

利用 $A+\overline{A}=1$ 可以把任何一个逻辑函数化为最小项之和的标准形式。这种标准形式在计算机辅助分析和设计中得到了广泛的应用。

例 1.8 将下列逻辑函数展开为最小项之和的形式。

① $Y_1=AB+B\overline{C}$

② $Y_2=\overline{A}BC+AB\overline{C}\,\overline{D}+CD$

解 $Y_1=AB+B\overline{C}=AB(C+\overline{C})+(A+\overline{A})B\overline{C}$

$$=ABC+AB\overline{C}+ABC+\overline{A}B\overline{C}=\sum_i m_i \quad (i=2,6,7)$$

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

$$=\overline{A}BCD+\overline{A}BC\overline{D}+AB\overline{C}\,\overline{D}+ABCD+\overline{A}BCD+A\overline{B}CD+\overline{A}\,\overline{B}CD$$

$$=m_7+m_6+m_{12}+m_{15}+m_{11}+m_3=\sum_i m_i \quad (i=3,6,7,11,12,15)$$

3．最大项的概念及其性质

（1）最大项

在 n 变量的逻辑函数中，若 M 为 n 个变量之和，而且这 n 个变量均以原变量或反变量的形式在 M 中出现一次，且仅出现一次，则称 M 为这组变量的最大项。

例如 A、B、C 三个变量的最大项有 $\overline{A}+\overline{B}+\overline{C}$、$\overline{A}+\overline{B}+C$、$\overline{A}+B+\overline{C}$、$\overline{A}+B+C$、$A+\overline{B}+\overline{C}$、$A+\overline{B}+C$、$A+B+\overline{C}$、$A+B+C$ 共 8（2^3）个最大项。对于 n 个变量则有 2^n 个最大项。可见，n 变量的最大项数目和最小项数目是相等的。

输入变量的每一组取值都使对应的一个最大项的逻辑值等于 0。例如在 3 变量 A、B、C 的最大项中，当 $A=1$，$B=0$，$C=1$ 时，使 $\overline{A}+B+\overline{C}=0$。若将使最大项为 0 的 ABC 取值看做一个二进制数，并以其对应的十进制数给最大项编号，则（$\overline{A}+B+\overline{C}$）可记做 M_5。由此得到三变量最大项编号表，如表 1-13 所示。

（2）最大项的性质

根据最大项的定义同样也可以得到它的主要性质，这就是：

①在输入变量的任一取值下必有一个最大项，而且仅有一个最大项的值为 0。

②全体最大项之积为 0。

③任意两个最大项之和为 1。

④只有一个变量不同的两个最大项的乘积等于各相同变量之和。

表 1-13　3 变量最大项的编号表

最大项	使最大项为 0 的变量取值			对应的十进制数	编号
	A	B	C		
$A+B+C$	0	0	0	0	M_0
$A+B+\overline{C}$	0	0	1	1	M_1
$A+\overline{B}+C$	0	1	0	2	M_2
$A+\overline{B}+\overline{C}$	0	1	1	3	M_3
$\overline{A}+B+C$	1	0	0	4	M_4
$\overline{A}+B+\overline{C}$	1	0	1	5	M_5
$\overline{A}+\overline{B}+C$	1	1	0	6	M_6
$\overline{A}+\overline{B}+\overline{C}$	1	1	1	7	M_7

如果将表 1-12 和表 1-13 加以对比则可发现，最大项和最小项之间存在如下关系

$$M_i=\overline{m_i} \tag{1.1}$$

4．逻辑函数的最大项之积形式

可以证明，任何一个逻辑函数都可以化为最大项之积的标准形式。

上面已经证明，任何一个逻辑函数皆可化为最小项之和的形式。同时，从最小项的性质又知道全部最小项之和为 1。由此可见，若给定逻辑函数为$Y=\sum m_i$，则$\sum m_i$以外的那些最小项之和必为$\overline{Y}$，即

$$\overline{Y}=\sum_{k\neq i} m_k \tag{1.2}$$

故得到

$$Y=\overline{\sum_{k\neq i} m_k} \tag{1.3}$$

利用反演定理可将上式变换为最大项乘积的形式

$$Y=\prod_{k\neq i}\overline{m_k}=\prod_{k\neq i} M_k \tag{1.4}$$

这就是说，如果已知逻辑函数为$Y=\sum m_i$时，定能将 Y 化成编号为 i 以外的那些最大项的乘积。

例 1.9　试将逻辑函数 $Y=AB+B\overline{C}$ 化为最大项之积的标准形式。

解　前面已经得到了它的最小项之和形式为

$$Y=\sum_i m_i \quad (i=2,6,7)$$

根据式（1.4）可得

$$Y=\prod_{k\neq i} M_k = M_0\cdot M_1\cdot M_3\cdot M_4\cdot M_5$$

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

1.4 逻辑函数的公式化简法

1.4.1 最简逻辑式的概念

一个具体的问题经过逻辑抽象得到的逻辑函数表达式，不一定是最简单的逻辑表达式。在进行逻辑运算时往往会看到，同一个逻辑函数可以写成不同的逻辑表达式，而这些逻辑表达式的简繁程度往往相差甚远。逻辑表达式越是简单，它所表示的逻辑关系越明显，同时也有利于用最少的电子器件实现这个逻辑函数。因此，通常需要通过化简的手段找出逻辑函数的最简形式。

例如有两个逻辑函数：

$$Y=\overline{A}\overline{B}\overline{C}+\overline{A}BC+ABC \qquad (1.5)$$

$$Y=\overline{A}B+BC \qquad (1.6)$$

将它们的真值表列出后可知，它们是同一个逻辑函数。显然下式比上式简单得多。

式（1.5）和式（1.6）都是由几个乘积项相加组成的，我们把这种形式的逻辑式称为与－或逻辑式，或叫做逻辑函数的“积之和”形式。

在与－或逻辑式中，若其中包含的乘积项已经最少，而且每个乘积项里的因子也不能再减少时，则称此逻辑函数式为最简形式。

化简逻辑函数的目的就是要消去多余的乘积项和每个乘积项中多余的因子，以得到逻辑函数的最简形式。

一个逻辑函数的乘积项少，表明电路所需元器件少；而每个乘积项中的因子少，表明电路的连线少。这样不但降低了电路的成本，又提高了设备的可靠性。所以，简化逻辑函数是逻辑设计中的重要步骤。

1.4.2 逻辑函数的公式化简法

逻辑函数的公式化简法，就是反复利用逻辑代数的基本公式和常用公式消去函数式中多余的乘积项和多余的因子，以求得函数式的最简形式。

公式化简法没有固定的步骤，现将经常使用的方法归纳如下。

1．并项法

利用公式 $AB+A\overline{B}=A$，可以将两项合并为一项，并消去 B 和 $\overline{B}$ 这一对因子。而且，根据代入定理可知，A 和 B 都可以是任何复杂的逻辑式。

例 1.10　试用并项法化简下列逻辑函数：

$$Y_1=\overline{A}BC+AC+\overline{B}C$$

$$Y_2=\overline{A}BCD+\overline{\overline{A}BCD}$$

解　$Y_1=\overline{A}BC+AC+\overline{B}C=\overline{A}BC+(A+\overline{B})C=\overline{A}BC+\overline{\overline{A}BC}=C$

$Y_2=\overline{A}BCD+\overline{\overline{A}BCD}=(\overline{A}BC+\overline{\overline{A}BC})D=D$

2．吸收法

利用公式 $A+AB=A$ 可以将 AB 消去，A 和 B 同样也可以是任何一个复杂的逻辑式。

例 1.11　试用吸收法化简下列逻辑函数：

$$Y_1=A\overline{B}+A\overline{B}D+A\overline{B}\overline{C}D$$

$$Y_2=\overline{A\overline{B}}+\overline{A}CD+\overline{B}CD+\overline{B}CDEF$$

解　$Y_1=\overline{A}B+\overline{A}B\overline{D}\quad\overline{A}B\overline{C}D(1A\overline{B}\quad+D)\quad\overline{C}1$

$Y_2=\overline{A}B+\overline{A}C\overline{B}\quad B\overline{C}\overline{D}\quad B\overline{C}D\overline{E}\overline{F}+\quad\overline{A}+\quad\overline{B}\quad+AC\overline{D}\quad=\overline{B}$

3．消项法

利用 $AB+\overline{A}C+BC=AB+\overline{A}C$ 及 $AB+\overline{A}C+BCD=AB+\overline{A}C$ 将项 BC 及 BCD 消去。其中 A、B、C、D 都可以是任何复杂的逻辑式。

例 1.12　用消项法化简下列逻辑函数：

$$Y_1=\overline{A}BC+\overline{\overline{A}BD}+CDEF$$

$$Y_2=ABC+\overline{\overline{A}BC}+A\overline{B}D+\overline{A}BD+\overline{A}BCD+BCD\overline{E}$$

解　$Y_1=\overline{A}BC+\overline{\overline{A}BD}\quad CD\overline{E}F\quad\overline{A\overline{B}}C$

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

$Y_2=\quad ABC+\overline{\overline{A}BC}\quad\overline{A}B\overline{C}\quad A\overline{B}D\quad\overline{A\overline{B}}D\quad AB\overline{C}D$

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

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

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

4．配项法

利用公式 $A+A=A$ 可以在逻辑函数式中重复写入某一项，有时能获得更加简

单的结果；利用公式 $A+\overline{A}=1$ 可以在函数式中的某一项乘以 $(A+\overline{A})$，然后拆成两项分别与其他项合并，有时能得到更加简单的化简结果。

例 1.13 试化简下列逻辑函数：

$$Y_1 = AB\overline{C} + \overline{A}BC + ABC$$

$$Y_2 = \overline{A}\overline{B} + \overline{B}\overline{C} + BC + AB$$

解 $Y_1 = AB\overline{C} + \overline{A}BC + AB(C+\overline{C}) + (A+\overline{A})BC = AB\overline{C} + AB + BC$

$$\begin{aligned} Y_2 &= \overline{A}\overline{B} + \overline{B}\overline{C} + BC + AB \\ &= \overline{A}\overline{B}(C+\overline{C}) + \overline{B}\overline{C} + (A+\overline{A})BC + AB \\ &= \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} + \overline{B}\overline{C} + ABC + \overline{A}BC + AB \\ &= AB + \overline{B}\overline{C} + \overline{A}C(B+\overline{B}) \\ &= AB + \overline{B}\overline{C} + \overline{A}C \end{aligned}$$

5．消去因子法

利用公式 $A+\overline{A}B = A+B$ 可将 $\overline{A}B$ 中的 $\overline{A}$ 消去。A、B 均可以是任何复杂的逻辑式。

例 1.14 试用消因子法化简下列逻辑函数：

$$Y_1 = \overline{A} + ABC$$

$$Y_2 = AB + \overline{A}C + \overline{B}C$$

解 $Y_1 = \overline{A} + ABC = \overline{A} + BC$

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

在化简复杂的逻辑函数时，往往需要灵活、交替地综合运用上述方法，才能得到最后的化简结果。

例 1.15 化简逻辑函数：

$$Y = B\overline{C} + AB\overline{C}E + \overline{B}(\overline{\overline{A}\overline{D}} + AD) + B(\overline{A}D + A\overline{D})$$

解 $Y = B\overline{C} + AB\overline{C}E + \overline{B}(\overline{\overline{A}\overline{D}} + AD) + B(\overline{A}D + A\overline{D})$

$$\begin{aligned} &= B\overline{C}(1+AE) + \overline{B}(\overline{A \oplus D}) + B(A \oplus D) \\ &= B\overline{C} + A \oplus D \end{aligned}$$

1.5 逻辑函数的卡诺图化简

用公式法化简逻辑函数时，往往需要一定的经验技巧，而且对结果是否为最简需具备一定的判断力，规律性不强。当变量不多时，采用卡诺图法化简逻辑函

数则比较直观。

1.5.1　逻辑函数的卡诺图表示法

1．卡诺图的构成

将 n 变量逻辑函数的全部最小项各用一个小方块表示，并使具有逻辑相邻性的最小项在几何位置上也相邻地排列起来，所得到的图形称为 n 变量的卡诺图。它的得名来自于它的提出者——美国工程师卡诺（Karnaugh）。卡诺图也是逻辑函数的一种表示方法。

图 1-9（a）、（b）、（c）、（d）分别为二到五变量的卡诺图。相应的最小项可用变量的标准积来标出，也可以用最小项 m_i 来标出。

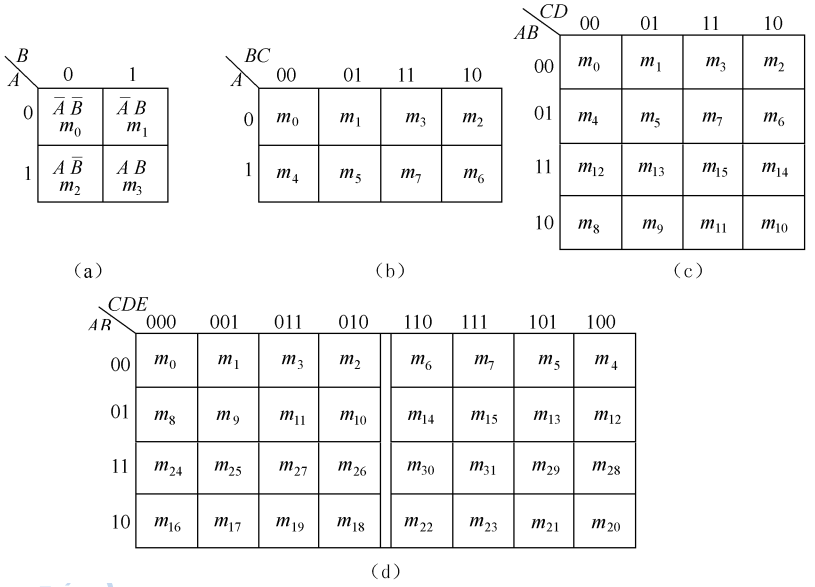


图 1-9　二到五变量最小项的卡诺图

图形两侧标注的 0 和 1 表示使对应小方格内的最小项为 1 的变量取值。同时，这些 0 和 1 组成的二进制所对应的十进制数大小也就是对应最小项的编号。

为了保证图中几何位置相邻的最小项在逻辑上也具有相邻性，在制作卡诺图时要特别注意变量组合值的排列规则。其原则是，每行（列）与相邻行（列）之间的变量组合值中，仅有一个变量发生变化（0→1 或 1→0）。相邻行（列）是指上下及左右相邻，也包括紧靠上下两边及紧靠左右两边的行、列相邻。因此，从几何位置上应当把卡诺图看成是上下、左右闭合的图形。

综上所述，卡诺图的特点是：n 个变量的卡诺图具有 2^n 个小方块，它们分别

与 2^n 个最小项相对应。相邻两个小方块中变量仅有一个发生变化，其他的都相同；反过来，仅有一个变量发生变化的小方块是相邻的小方块。

2．逻辑函数的卡诺图表示

既然任何一个逻辑函数都能表示为若干最小项之和的形式，那么自然也就可以设法用卡诺图来表示任意一个逻辑函数。具体方法是首先把逻辑函数化为最小项之和的形式，然后在卡诺图上与这些最小项对应的位置填入 1，其余位置上填入 0，就得到了表示该逻辑函数的卡诺图。也就是说，任何一个逻辑函数都等于它的卡诺图中填入 1 的那些最小项之和。

例 1.16 用卡诺图表示下列逻辑函数：

$$Y = ABC\overline{D} + B\overline{C} + AD$$

解 首先将 Y 化为最小项之和的形式：

$$\begin{aligned} Y &= ABC\overline{D} + (A+\overline{A})B\overline{C}(D+\overline{D}) + A(B+\overline{B})(C+\overline{C})D \\ &= ABC\overline{D} + AB\overline{C}D + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + ABCD + A\overline{B}CD \\ &\quad + AB\overline{C}D + A\overline{B}\,\overline{C}D \\ &= m_{14} + m_{13} + m_5 + m_4 + m_{12} + m_{15} + m_{11} + m_9 \\ &= \sum_i m_i (i = 4,5,9,11,12,13,14,15) \end{aligned}$$

然后画出四变量的卡诺图，在对应于函数式中各最小项的位置填入 1，其余位置上填入 0，即得到该逻辑函数的卡诺图，如图 1-10 所示。

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	0

图 1-10 例 1.16 的卡诺图

1.5.2 用卡诺图化简逻辑函数

利用卡诺图化简逻辑函数的方法称为卡诺图化简法或图形化简法。化简的基本方法是合并相邻最小项，并消去不同的因子。从卡诺图的结构可知，由于在卡诺图中几何位置的相邻性与逻辑上的相邻性是一致的，因而相邻小方块所

对应的最小项只有一个变量发生变化，其余取值相同。因此，利用公式 $AB+A\overline{B}=A$ 可把卡诺图上相邻小方块所对应的最小项合并为一个乘积项，并消去互补的变量因子。

1．合并最小项的规则

①若两个最小项相邻，则可合并为一项并消去一对因子。合并后的结果中只剩下公共因子。

②若四个最小项相邻并排列成一个矩形（或正方形）组，则可合并为一项并消去两对因子。合并后的结果中只包含公共因子。

③若八个最小项相邻并排列成一个矩形（或正方形）组，则可合并为一项并消去三对因子。合并后的结果中只包含公共因子。

例如，已知某四变量的卡诺图如图 1-11 所示。由图可见，m_2 和 m_{10} 是两个逻辑值为 1 的上下相邻最小项，这两个最小项 $\overline{A}B\overline{C}\overline{D}$ 及 $A\overline{B}\overline{C}\overline{D}$ 之间只有 A 变量发生变化（互补），可将其圈起来，作为一方格圈，合并后将 A 和 $\overline{A}$ 这一对因子消去，只剩下公共因子 $\overline{B}\overline{C}\overline{D}$。

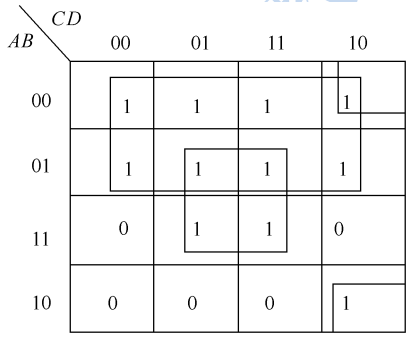


图 1-11　相邻最小项合并举例

m_5、m_7、m_{13} 和 m_{15} 是 4 个逻辑值为 1 的相邻最小项，合并后得到

$$\overline{A}\overline{B}\overline{C}D+\overline{A}BCD+AB\overline{C}D+ABCD=\overline{A}BD(\overline{C}+C)+ABD(C+\overline{C})$$

$$=(A+\overline{A})BD=BD$$

可见，合并后消去了 A 和 $\overline{A}$、C 和 $\overline{C}$ 两对因子，只剩下公共因子 B 和 D。

同理，m_0~m_7 是八个逻辑值为 1 的相邻最小项，合并后得到结果为 $\overline{A}$。

至此，可以归纳出合并最小项的一般规则是：如果有 2^n 个最小项相邻（$n=1,2,\cdots$）并排列成一个矩形组，则它们可以合并为一项，并消去 n 对因子，合并后的结果中仅包含这些最小项的公共因子。

2．卡诺图化简法的步骤

用卡诺图化简逻辑函数可以按照如下步骤进行：

①将函数化为最小项之和的形式；

②画出表示该逻辑函数的卡诺图；

③找出可以合并的最小项（画方格圈）；

④得到化简后的乘积项及其逻辑或的结果。

方格圈的选取原则是：

（1）用卡诺图化简逻辑函数时，每一个最小项（也就是填有 1 的小方块）必须被圈，不能遗漏。

（2）某一个最小项可以多次被圈，但每次被圈时，圈内至少包含一个新的最小项。

（3）圈越大，则消去的变量越多，合并项越简单。圈内小方块的个数应是 $N=2^i(i=0,1,2,\cdots)$。

（4）合并时应检查是否最简。即在保证乘积项最少的前提下，各乘积项变量的因子应最少。在卡诺图上乘积项最少也就是可合并的最小项组成的方格圈数目最少，而各乘积项的因子最少也就是每个可合并的最小项方格圈中应包含尽可能多的最小项。

（5）有时用圈 0 的方法更简便，但得到的化简结果是原函数的反函数。

例 1.17 用卡诺图化简法将下式化简为最简与－或函数式：

$$Y=\overline{A}B+A\overline{B}+B\overline{C}+\overline{B}C$$

解 首先画出表示函数 Y 的卡诺图，如图 1-12 所示。

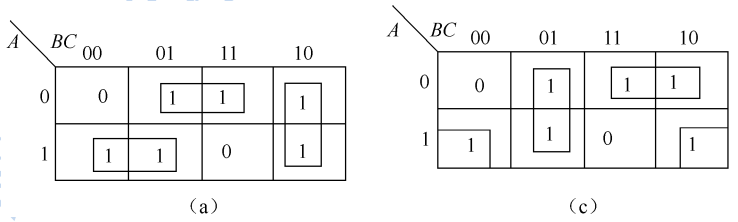


图 1-12 例 1.17 的卡诺图

其次，需要找出可以合并的最小项。将可以合并的最小项圈出，由图 1-12（a）和（b）可见，有两种合并最小项的方案。如果按 1-12（a）图合并最小项得到：

$$Y=A\overline{B}+\overline{A}C+B\overline{C}$$

而按图 1-12（b）合并最小项则得到：

$$Y=A\overline{C}+\overline{A}B+\overline{B}C$$

两个化简结果都符合最简与－或式的标准。因此有时一个逻辑函数化简结果不是唯一的。

例 1.18　用卡诺图化简法将下式化简为最简与－或函数式：

$$Y = A\overline{B}\overline{C}D + BCD + \overline{A}D + A\overline{B}\overline{C}\overline{D}$$

解　首先画出表示函数 Y 的卡诺图，如图 1-13 所示。事实上，在填写 Y 的卡诺图时，并不一定要将 Y 化为最小项之和的形式。例如式中 $\overline{A}D$ 项，在填写 Y 的卡诺图时可以直接在卡诺图上对应 $A = 0$，$D = 1$ 的空格里填入 1。按照这种方法，就可以省去将 Y 化为最小项之和这一步骤了。

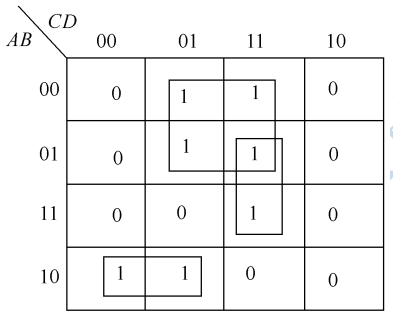


图 1-13　例 1.18 的卡诺图

然后，把可以合并的最小项圈出，并得到最简与－或函数式如下：

$$Y = \overline{A}D + BCD + A\overline{B}\overline{C}$$

1.6　具有无关项的逻辑函数及其化简

1.6.1　逻辑函数的无关项

前面讨论的逻辑函数，对应任意一组输入变量值，函数都有确定的输出：或为 1，或为 0。若有 n 个输入变量，则其共有 2^n 个输入变量的组合值，然而，在实际情况中会遇到这样的逻辑函数：它有 n 个输入变量，但函数值仅取决于其中的 K 个组合值，而与（$2^n - K$）个组合值无关。有两种情况可使这（$2^n - K$）个组合值（最小项）不能给函数的输出以确定值。其一是输入变量的这（$2^n - K$）个组合值（最小项）在该逻辑函数中不会出现或不允许出现；其二是这（$2^n - K$）个组合值（最小项）出现时，对函数的输出值没有影响。

例如，用 8421BCD 码表示十进制的 10 个数字符号时，只有 0000，0001，…，1001 等 10 种组合有效，而 1010~1111 这 6 种组合是不会出现的。如用 $ABCD$ 表

示此 8421BCD 码，则 $A\bar{B}C\bar{D}$ 、 $A\bar{B}CD$ 、 $AB\bar{C}\bar{D}$ 、 $AB\bar{C}D$ 、 $ABC\bar{D}$ 和 $ABCD$ 是与这种编码无关的组合。再如计算器的加法、减法、乘法三种运算（分别用 A、B、C 表示），任何时候只允许进行一种操作，不允许两种或三种操作同时进行，即只能是 000，001，010，100 四种情况之一，而 $\bar{A}BC$ 、 $A\bar{B}C$ 、 $AB\bar{C}$ 和 ABC 是被禁止的，这就是说 A、B、C 是一组具有约束的变量。

一般把逻辑函数的输出值中不会出现或不允许出现的最小项称为约束项。

有时还会遇到在输入变量的某些取值下逻辑函数的输出值是 0 还是 1 皆可，并不影响电路的功能。在这些变量取值下，其值等于 1 的那些最小项称为任意项。

在存在约束项的情况下，由于约束项的值始终等于 0，所以既可以把约束项写进逻辑函数式中，也可以把约束项从逻辑函数中删除而不影响函数值。同样，既可以把任意项写入函数式，也可以不写进去。因为输入变量的取值使这些任意项为 1 时，函数值是 1 还是 0 无所谓。

因此，又把约束项和任意项统称为无关项。这里所说的无关是指是否把这些最小项写入逻辑函数式无关紧要，可以写入也可以删除。

在填卡诺图时，无关项的小方块用×表示。在化简逻辑函数时既可以认为它是 1，也可以认为它是 0。

1.6.2 具有无关项的逻辑函数的化简

在利用卡诺图化简具有无关项的逻辑函数时，如果能合理利用这些无关项，一般都可以得到更加简单的结果。合并最小项时，究竟是把卡诺图上的×作为 1（即认为函数式中包含了这个最小项）还是作为 0（即认为函数式中不包含这个最小项）对待，应以得到的相邻最小项方格圈最大，而且以方格圈的数目最少为原则。

例 1.19 化简具有约束项的逻辑函数：

$$Y=\bar{A}\bar{B}CD+\bar{A}BC\bar{D}+\bar{A}B\bar{C}D+\bar{A}BCD+A\bar{B}C\bar{D}+A\bar{B}CD+ABC\bar{D}+ABCD$$

约束条件为：

$$\bar{A}\bar{B}CD+\bar{A}BCD+A\bar{B}\bar{C}\bar{D}=0$$

解 画出函数 Y 的卡诺图，如图 1-14 所示。

由图可见，若将其中的约束项 m_3、m_7 看成 1，而 m_8 看成 0，则可将 m_2、m_3、m_6、m_7、m_{10}、m_{11}、m_{14} 和 m_{15} 合并为 C，将 m_1、m_3、m_5 和 m_7 合并为 $\bar{A}D$，于是得到

$$Y=C+\bar{A}D$$

例 1.20 试化简逻辑函数：

$$Y=\sum_i m_i(i=0,2,4,6,8,9,10,11,12,13,14,15)$$

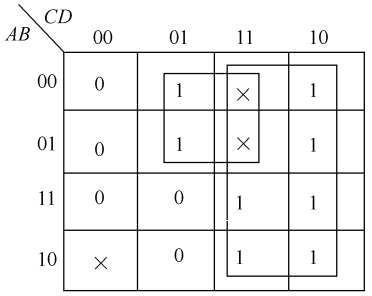


图 1-14　例 1.19 的卡诺图

已知约束条件为：

$$m_1 + m_5 = 0$$

解　首先画出函数 Y 的卡诺图，如图 1-15 所示。

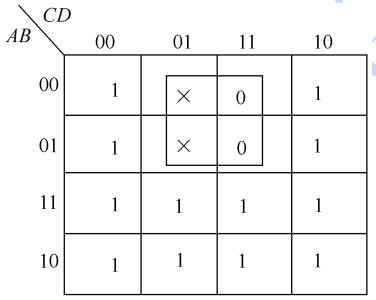


图 1-15　例 1.20 的卡诺图

由图可见，将×作为 0 处理，用圈 0 较方便，但化简结果得到的是 $\overline{Y}$，化简结果如下：

$$\overline{Y} = \overline{A}D$$

则

$$Y = A + \overline{D}$$

1.7　逻辑函数的变换与实现

1.7.1　逻辑函数的表达形式

前面我们所学习的逻辑函数的一般形式或最简形式通常都是以与－或的形式表达的，它需要使用与门和或门两种类型的器件。但是在实际应用的时候，有时只有一种类型的器件，或者是希望电路只用一种类型的器件来实现，以保证系统的经济性和可靠性。那么就需要对现有的与－或形式的逻辑函数进行变换，变换

成所需要的其他形式的逻辑函数。

这些其他形式包括与非－与非式、或－与式、或非－或非式、与或非式等。它们对最简形式的定义也与对最简与－或式的定义一样，即函数式中相加的乘积项不能再减少，而且每项中相乘的因子不能再减少时，则函数式为最简形式。

1．与－或式

与－或式的特点是先与运算后或运算，例如：

$$Y = AB + CD \tag{1.7}$$

与－或式用与逻辑和或逻辑实现，其逻辑图见图 1-16（a）所示。

2．与非－与非式

与非－与非式是逻辑电路设计中最常用的一种表达形式。例如：

$$Y = \overline{\overline{AB} \cdot \overline{CD}} \tag{1.8}$$

与非－与非式仅用与非逻辑实现，其逻辑图见图 1-16（b）所示。

3．或－与式

或－与式的特点是先或运算后与运算，例如：

$$Y = (A + B)(C + D) \tag{1.9}$$

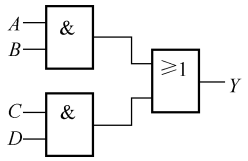
或－与式用或逻辑和与逻辑实现，其逻辑图见图 1-16（c）所示。

4．或非－或非式

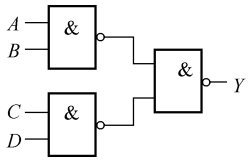
或非－或非式也是逻辑电路设计中常用的一种表达形式。例如：

$$Y = \overline{\overline{A + B} + \overline{C + D}} \tag{1.10}$$

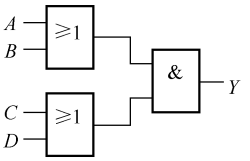
或非－或非式仅用或非逻辑实现，其逻辑图见图 1-16（d）所示。



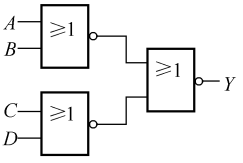
（a）与–或式逻辑图



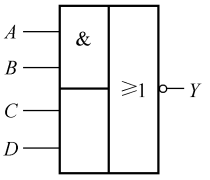
（b）与非–与非式逻辑图



（c）或–与式逻辑图



（d）或非–或非式逻辑图



（e）与或非式逻辑图

图 1-16　各种逻辑表达形式对应的逻辑图

5．与或非式

与或非式也是逻辑电路设计中常用的一种表达形式，其格式如下：

$$Y=\overline{AB+CD} \tag{1.11}$$

与或非式直接用与或非逻辑实现，其逻辑图见图 1-16（e）所示。

1.7.2　逻辑函数的变换

通过逻辑函数的化简得到最简的与－或形式以后，可以通过公式变换得到其他类型的函数形式。

假设一个逻辑函数的最简与－或式是：

$$Y=AB+\overline{B}C$$

现在我们将它变换成其他形式的逻辑函数。

1．与－或式变换为与非－与非式

与非－与非式可由与－或式按还原律两次取反后，再用德·摩根定律展开得到。

因此　　$$Y=AB+\overline{B}C=\overline{\overline{AB+\overline{B}C}}=\overline{\overline{AB}\cdot\overline{\overline{B}C}}$$

其中，$\overline{B}$ 也可由与非门实现。

2．与－或式变换为或－与式

方法 1　根据逻辑函数的两种标准形式“最小项之和”和“最大项之积”之间的相互关系，可将与－或式变换为或－与式。

由 $Y=AB+\overline{B}C=AB(C+\overline{C})+(A+\overline{A})\overline{B}C$

$$=\sum_i m_i(i=1,5,6,7)=\prod_{k\neq i}M_k=M_0\cdot M_2\cdot M_3\cdot M_4$$

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

即可得或－与式。但这种方法得到的或－与式不一定为最简形式。

方法 2　根据对偶定理两次取对偶式来实现。

由 $Y=AB+\overline{B}C$ 得对偶式并展开：

$$Y'=(A+B)\cdot(\overline{B}+C)=A\overline{B}+AC+BC$$

再取对偶，$Y=(Y')'=(A+\overline{B})\cdot(A+C)\cdot(B+C)$

得到更简单的结果。

3．与－或式变换为或非－或非式

或非－或非式可由或－与式按还原律两次取反后，再用德·摩根定理展开得到。因此，利用与－或式变换为或－与式后的结果可进一步得到或非－或非式。

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

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

4．与一或式变换为与或非式

先求逻辑函数“最小项之和”的形式，再求出反函数的与一或形式，最后取反，可得到原函数的与或非式。

因此

$$Y=AB+\overline{B}C=\sum_i m_i(i=1,5,6,7)$$

$$\overline{Y}=\sum_{k\neq i} m_k(k=0,2,3,4)$$

$$=\overline{A}\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}\overline{C}=\overline{A}\overline{B}+\overline{B}\overline{C}$$

$$Y=\overline{\overline{A}\overline{B}+\overline{B}\overline{C}}$$

是最简与或非式。

1.8 辅修内容

1.8.1 数制和码制

1．数制

用数字量表示物理量的大小，仅用一位数码往往不够，因此经常需要用进位计数的方法组成多位数码使用。我们把多位数码中每一位的构成方法以及从低位到高位的进位规则称为数制。

在数字电路中经常使用的计数进制除了十进制以外，还经常使用二进制和十六进制。

（1）十进制

十进制是日常生活和工作中最常使用的进位计数制，在十进制中，每一位有0～9十个数码，所以计数的基数是10。超过9的数必须用多位表示，其中低位和高位之间的关系是“逢十进一”，故称为十进制。例如：

$$143.75 = 1\times10^{2}+4\times10^{1}+3\times10^{0}+7\times10^{-1}+5\times10^{-2}$$

所以任意一个十进制数 D 均可展开为

$$D=\sum k_i\times10^{i} \qquad (1.12)$$

其中 k_i 是第 i 位的系数，它可以是0～9这十个数码中的任何一个。若整数部分的位数是 n，小数部分的位数为 m，则 i 包含从 $n-1$ 到0的所有正整数和从–1到 $-m$ 的所有负整数。

若以 N 取代式（1.12）中的10，即可得到任意进制（N 进制）数展开式的普

遍形式

$$D=\sum k_i \times N^i \tag{1.13}$$

式中 i 的取值与式（1.12）中的规定相同。N 称为计数的基数，k_i 为第 i 位的系数，N_i 称为第 i 位的权。

（2）二进制

目前在数字电路中应用最广的是二进制。在二进制数中，每一位仅有 0 和 1 两个可能的数码，所以计数基数为 2。低位和相邻高位间的进位关系是“逢二进一”，故称为二进制。

根据式（1.13），任何一个二进制数均可展开为

$$D=\sum k_i \times 2^i \tag{1.14}$$

$$(101.11)_2=1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}=(5.75)_{10}$$

上式中分别使用下脚注的 2 和 10 表示括号中的数是二进制数和十进制数。有时也用 B（Binary）和 D（Decimal）代替 2 和 10 这两个脚注。

（3）十六进制

十六进制数的每一位有 16 个不同的数码，分别用 0～9、A（10）、B（11）、C（12）、D（13）、E（14）、F（15）表示。因此，任意一个十六进制数均可展开为

$$D=\sum k_i \times 16^i \tag{1.15}$$

$$(2A.7F)_{16}=2\times16^1+10\times16^0+7\times16^{-1}+15\times16^{-2}=(42.4960937)_{10}$$

上式中的下脚注 16 表示括号里的数是十六进制，有时也有 H（Hexadecimal）代替这个脚注。

由于目前在微型计算机中普遍采用 8 位、16 位和 32 位二进制并行计算，而 8 位、16 位和 32 位的二进制数可以用 2 位、4 位和 8 位的十六进制数表示，因而用十六进制符号书写程序十分方便。

2．数制转换

（1）二－十转换

把二进制数转换为等值的十进制数称为二－十转换。转换时只要将二进制数按式（1.14）展开，然后把所有各项的数值按十进制数相加，就可以得到等值的十进制数了。例如：

$$(1011.01)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}=(11.25)_{10}$$

（2）十－二转换

所谓十－二转换，就是把十进制数转换成等值的二进制数。

首先讨论整数的转换。

假定十进制整数为$(S)_{10}$，等值的二进制数为$(k_nk_{n-1}\cdots k_0)_2$，则依式（1.14）可知

$$\begin{aligned}(S)_{10} &= k_n2^n + k_{n-1}2^{n-1} + \cdots + k_12^1 + k_02^0 \\ &= 2(k_n2^{n-1} + k_{n-1}2^{n-2} + \cdots + k_1) + k_0\end{aligned} \tag{1.16}$$

上式表明，若将$(S)_{10}$除以 2，则得到的商为 $k_n2^{n-1}+k_{n-1}2^{n-2}+\cdots+k_1$，而余数即 k_0。

同理，将式（1.16）中的商除以 2 得到的新的商可写成：

$$k_n2^{n-1} + k_{n-1}2^{n-2} + \cdots + k_1 = 2(k_n2^{n-2} + k_{n-1}2^{n-3} + \cdots + k_2) + k_1 \tag{1.17}$$

由式（1.17）不难看出，若将$(S)_{10}$除以 2 所得的商再次除以 2，则所得余数即为 k_1。

依此类推，反复将每次得到的商再除以 2，就可求得二进制数的每一位了。

例如，将$(173)_{10}$化为二进制数可如下进行：

$$\begin{array}{r|l} 2 & 173 \quad \cdots\cdots\text{余数} = 1 = k_0 \\ 2 & 86 \quad \cdots\cdots\text{余数} = 0 = k_1 \\ 2 & 43 \quad \cdots\cdots\text{余数} = 1 = k_2 \\ 2 & 21 \quad \cdots\cdots\text{余数} = 1 = k_3 \\ 2 & 10 \quad \cdots\cdots\text{余数} = 0 = k_4 \\ 2 & 5 \quad \cdots\cdots\text{余数} = 1 = k_5 \\ 2 & 2 \quad \cdots\cdots\text{余数} = 0 = k_6 \\ 2 & 1 \quad \cdots\cdots\text{余数} = 1 = k_7 \\ & 0 \end{array}$$

故$(173)_{10} = (10101101)_2$。

其次讨论小数的转换。

若$(S)_{10}$是一个十进制的小数，对应的二进制小数为$(0.k_{-1}k_{-2}\cdots k_{-m})_2$，则根据式（1.14）可知

$$(S)_{10} = k_{-1}2^{-1} + k_{-2}2^{-2} + \cdots + k_{-m}2^{-m}$$

将上式两边同乘以 2 得到

$$2(S)_{10} = k_{-1} + (k_{-2}2^{-1} + k_{-3}2^{-2}\cdots + k_{-m}2^{-m+1}) \tag{1.18}$$

式（1.18）说明，将小数$(S)_{10}$乘以 2 所的乘积的整数部分即为k_{-1}。

$$2(k_{-2}2^{-1} + k_{-3}2^{-2} + \cdots + k_{-m}2^{-m+1}) = k_{-2} + (k_{-3}2^{-1} + \cdots + k_{-m}2^{-m+2}) \tag{1.19}$$

亦即乘积的整数部分就是k_{-2}。

依此类推，将每次乘 2 后所得乘积的小数部分再乘以 2，便可求出二进制小数的每一位。

例如，将$(0.8125)_{10}$化为二进制小数时可如下进行：

0.8125

× 2 ……整数部分=1=k_{-1}

0.6250

× 2 ……整数部分=1=k_{-2}

0.2500

× 2 ……整数部分=0=k_{-3}

0.5000

× 2 ……整数部分=1=k_{-4}

故$(0.8125)_{10}=(0.1101)_2$。

（3）二－十六转换

把二进制数转换成等值的十六进制数称为二－十六转换。

由于 4 位二进制数恰好有 16 个状态，而把这 4 位二进制数看作一个整体时，它的进位输出又正好是逢十六进一，所以只要从低位到高位将每 4 位二进制数分为一组并代之以等值的十六进制数，即可得到对应的十六进制数。

例如，将$(01011110.10110010)_2$化为十六进制数时可得：

$(0101\quad 1110.\ 1011\quad 0010)_2$

↓　↓　↓　↓

$=(5\quad E.\quad B\quad 2)_{16}$

（4）十六－二转换

十六－二转换是指把十六进制数转换成等值的二进制数。转换时只需将十六进制数的每一位用等值的 4 位二进制数代替就行了。

例如，将$(8FA.C6)_{16}$化为二进制数时得到：

$(8\quad F\quad A.\quad C\quad 6)_{16}$

↓　↓　↓　↓　↓

$(1000\quad 1111\quad 1010.\quad 1100\quad 0110)_2$

（5）十六－十的转换

在将十六进制数转换成十进制数时，可根据式（1.15）将各位按权展开后相加得到。在将十进制数转换为十六进制数时，可以先转换成二进制数，然后再将得到的二进制数转换为等值的十六进制数。这两种转换方法上面已经讲过了。

3．码制

不同的数码不仅可以表示数量的大小，而且还能用来表示不同的事物。在后一种情况下，这些数码已没有表示数量大小的含义，只是表示不同事物的代号而已。这些数码称为代码。

为便于记忆和处理，在编制代码是总要遵循一定的规则，这些规则就叫做码制。

例如在用 4 位二进制数码表示 1 位十进制数的 0～9 这十个状态时，就有多种不同的码制。通常将这些代码称为二－十进制代码，简称 BCD（Binary Coded Decimal）代码。

表 1-14 中列出了几种常见的 BCD 代码，它们的编码规则各不相同。

表 1-14　几种常见的 BCD 码

编码种类 十进制数	8421 码	余 3 码	2421 码	5211 码	余 3 循环码
0	0000	0011	0000	0000	0010
1	0001	0100	0001	0001	0110
2	0010	0101	0010	0100	0111
3	0011	0110	0011	0101	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1010	1101	1100	1111
8	1000	1011	1110	1101	1110
9	1001	1100	1111	1111	1010
权	8421		2421	5211	

8421 码是 BCD 代码中最常用的一种。在这种编码方式中每一位二值代码的 1 都代表一个固定数值，把每一位的 1 代表的十进制数加起来，得到的结果就是它所代表的十进制数码。由于代码中从左到右每一位的 1 分别表示 8、4、2、1，所以把这种代码叫做 8421 码。每一位的 1 代表的十进制数称为这一位的权。

8421 码中每一位的权是固定不变的，它属于恒权代码。

余 3 码的编码规则与 8421 码不同，如果把每一个余 3 码看作 4 位二进制数，则它的数值要比它所表示的十进制数码多 3，故而将这种代码叫做余 3 码。

如果将两个余 3 码相加，所得的和将比十进制数和所对应的二进制数多 6。因此，在用余 3 码做十进制加法运算时，若两数之和为 10，正好等于二进制数的 16，于是便从高位自动产生进位信号。

此外，从表 1-14 中还可以看出，0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 的余 3 码互为反码，这对于求取对 10 的补码是很方便的。

余 3 码不是恒权代码。如果试图把每个代码视为二进制数，并使它等效的十进制数与所表示的代码相等，那么代码中每一位的 1 所代表的十进制数在各个代码中不能是固定的。

2421 码是一种恒权代码，它的 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 也互为反码，这个特点与余 3 码相仿。

5211 码是另一种恒权代码。待学了第 5 章中计数器的分频作用后可以发现，如果按 8421 码接成十进制计数器，则连续输入技术脉冲时 4 个触发器输出脉冲对于计数脉冲的分频比从低位到高位依次为 5:2:1:1。可见，5211 码每一位的权正好与 8421 码十进制计数器 4 个触发器输出脉冲的分频比相对应。这种对应关系在构成某些数字系统时很有用。

余 3 循环码是一种变权码，每一位的 1 在不同代码中并不代表固定的数值。它的主要特点是相邻的两个代码之间仅有一位的状态不同。因此，按余 3 循环码接成计数器时，每次状态转换过程中只有一个触发器翻转，译码时不会发生竞争一冒险现象（详见第 3.5 节）。

4．算术运算

当两个二进制数码表示两个数量大小时，它们之间可以进行数值运算。这种运算称为算术运算。二进制算术运算和十进制算术运算的规则基本相同，唯一的区别在于二进制数是逢二进一，而不是十进制数的逢十进一。

例如，两个二进制数 1001 和 0101 的算术运算有：

①加法运算

```
  1001
+ 0101
------
  1110
```

②减法运算

```
  1001
- 0101
------
  0100
```

③乘法运算

```
   1001
×  0101
-------
   1001
  0000
 1001
0000
-------
0101101
```

④除法运算

```
      1.11…
0101 ) 1001
       0101
       ----
       1000
       0101
       ----
        0111
        0101
        ----
        0010
```

在数字电路和数字电子计算机中，二进制数的正、负号也是用 0 和 1 表示的。在定点运算的情况下，以最高位作为符号位，正数为 0，负数为 1，以下各位的 0 和 1 表示数值。用这种方法表示的数码称为原码。例如：

$(0\ 1011001)_2=(+89)_{10}$

↑

符号位

↓

$(1\ 1011001)_2=(-89)_{10}$

为了简化运算电路，在数字电路中两数相减的运算是用它们的补码相加来完成的。二进制数的补码是这样定义的：

最高位为符号位，正数为 0，负数为 1；

正数的补码和它的原码相同；

负数的补码可通过将原码的数值位逐位求反，然后在最低位加 1 得到。

例 1.21 计算$(1001)_2-(0101)_2$。

$$\begin{array}{r} 1001 \\ -\ 0101 \\ \hline 0110 \end{array}$$

在采用补码运算时，首先求出$(+1001)_2$和$(-0101)_2$的补码，它们是：

$$[+1001]_{补} = \boxed{0}\ 1001$$

符号位

$$[-0101]_{补} = \boxed{1}\ 1011$$

符号位

然后将两个补码相加并舍去进位：

$$\begin{array}{r} 01001 \\ +\ 11011 \\ \hline 舍去\leftarrow 100100 \end{array}$$

则得到与前面同样的结果。这样就把减法运算转换成了加法运算。

此外，我们也不难发现，乘法运算可以用加法和移位两种操作实现，而除法运算可以用减法和移位操作实现。因此，二进制数的加、减、乘、除运算都可以用加法运算电路完成，这就大大简化了运算电路的结构。

1.8.2 用 Q-M 法化简逻辑函数

前已述及用卡诺图化简逻辑函数有一定的局限性。首先，当变量数目较多时（例如大于 5 以后），便失掉了简单、直观的优点。其次，在许多情况下要凭设计者的经验确定如何合并最小项，因而不利于用计算机去完成。

由奎恩（Quine）、麦克拉斯基（McLuskey）提出的 Q-M 法（亦称列表法）较好地克服了上述局限性。这种方法不仅适用于多变量逻辑函数的化简，而且由于有一定的规则和步骤可循，可以借助于计算机进行化简。

Q-M法的基本原理仍然是通过合并相邻最小项并消去多余因子而求得逻辑函数最简与－或表达式的。下面结合一个具体的例子介绍一下 Q-M 法的基本原理和化简步骤。假定需要化简的五变量逻辑函数为：

$$Y=\sum(0,2,3,8,10,14,15,22,24,27,31) \qquad (1.20)$$

则使用 Q-M 法化简的步骤如下：

①将函数化成最小项之和的标准形式，并按编号把这些最小项列出。最小项中的原变量用 1 表示，反变量用 0 表示，于是得到：

编　号	0	2	3	8	10	14
最小项	00000	00010	00011	01000	01010	01110
编　号	15	22	24	27	31	
最小项	01111	10110	11000	11011	11111	

②按包含 1 的个数将最小项分组，如表 1-15 中最左边一列所示。

表 1-15　列表合并最小项

合并前的最小项（$\sum m_i$）							第一次合并结果（含 $n-1$ 个变量的乘积项）							第二次合并结果（含 $n-2$ 个变量的乘积项）						
编号	A	B	C	D	E		编号	A	B	C	D	E		编号	A	B	C	D	E	
0	0	0	0	0	0	√	0,2	0	0	0	-	0	√	0,2,8,10	0	-	0	-	0	P_8
2	0	0	0	1	0	√	0,8	0	-	0	0	0	√							
8	0	1	0	0	0	√	2,3	0	0	0	1	-	P_2							
3	0	0	0	1	1	√	2,10	0	-	0	1	0	√							
10	0	1	0	1	0	√	8,10	0	1	0	-	0	√							
24	1	1	0	0	0	√	8,24	-	1	0	0	0	P_3							
14	0	1	1	1	0	√	10,14	0	1	-	1	0	P_4							
22	1	0	1	1	0	P_1	14,15	0	1	1	1	-	P_5							
15	0	1	1	1	1	√	15,31	-	1	1	1	1	P_6							
27	1	1	0	1	1	√	27,31	1	1	-	1	1	P_7							
31	1	1	1	1	1	√														

③合并相邻的最小项。将表 1-15 中最左边一列里每一组的每一个最小项与相邻组里所有最小项逐一比较，若仅有一个因子不同，可定可合并，并消去不同的因子。消去的因子用“-”表示，将合并后的结果列于表 1-15 的第二列中。同时，在第一列中可以合并的最小项右边标以“√”号。

按照同样的方法再将第二列中的乘积项合并，合并后的结果写在第三列中。

如此进行下去，直到不能再合并为止。

④选择最少的乘积项。只要将表 1-15 中合并过程中没有用过的那些乘积项相加，自然就包含了函数 Y 的全部最小项，故得：

$$Y = P_1+P_2+P_3+P_4+P_5+P_6+P_7+P_8 \qquad (1.21)$$

然而上式并不一定是最简的与－或表达式。为了进一步将式（1.21）化简，将 P_1～P_8 各包含的最小项列成表 1-16。因为表中带圆圈的最小项仅包含在一个乘积项中，所以化简结果中一定包含它们所在的这些乘积项，即 P_1、P_2、P_3、P_7 和 P_8。而且，选取了这五项之和以后，已包含了除 m_{14} 和 m_{15} 以外所有 Y 的最小项。

这样，剩下的问题就是要确定化简结果中应否包括 P_4、P_5 和 P_6。为此，可以将表 1-16 简化为表 1-17 的形式。由于表中 P_4 行所有的 1 和 P_6 行所有的 1 皆包含在 P_5 行的 1 之中，亦即 P_5 行的最小项包含了 P_4 和 P_6 的所有最小项，故可将 P_4 和 P_6 两行删除（即从 Y 的表达式中将 P_4 和 P_6 去掉），从而得到最后的化简结果。

表 1-16　用列表法选择最少的乘积项

P_j \ m_i	0	2	3	8	10	14	15	22	24	27	31
P_1								①			
P_2		1	①								
P_3				1					①		
P_4					1	1					
P_5						1	1				
P_6							1				1
P_7										①	1
P_8	①	1		1	1						

表 1-17　表 1-16 的简化

P_j \ m_i	14	15
P_4	1	
P_5	1	1
P_6		1

$$Y = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8$$

$$= A\bar{B}CD\bar{E} + \bar{A}\bar{B}\bar{C}D + B\bar{C}\bar{D}\bar{E} + \bar{A}BCD + ABDE + \bar{A}\bar{C}\bar{E} \qquad (1.22)$$

1.8.3　VHDL 语言基础

数字电路的逻辑功能可用逻辑函数表达式、真值表、卡诺图、逻辑图及波形图来描述，这是传统意义上的描述方法。因此传统的数字系统设计是以中小规模的集成电路作为基本元件，随着电子技术的发展，集成电路制作工艺的进步，大规模集成电路特别是可编程逻辑器件 PLD 的迅速发展，数字系统设计的概念发生了质的变化。为了适应数字系统设计的这一变化，数字系统领域最新的趋势是数字电路给予文本的语言表述。

VHDL（Very High Speed Integrated Circuit Hardware Description Language）即超高速集成电路硬件描述语言。20 世纪 80 年代初美国国防部为使得承包电子系统项目的各公司之间的设计能被重复利用，制定了 VHDL，1987 年 12 月，VHDL 被正式接受为国际标准，编号为 IEEE Std 1076-1987，即 VHDL'87。1993 年被更新为 IEEE Std 1164-1993，即 VHDL'93。VHDL 语言成为 IEEE 的标准后，很快得到广泛应用，已经成为数字系统/ASIC 设计中的主要硬件描述语言。1995 年中国国家技术监督局组织编撰并出版了《CAD 通用技术规范》，推荐 VHDL 语言作为我国电子设计自动化硬件描述语言的国家标准。

1．VHDL 语言的标识符、常量及信号

（1）标识符

VHDL 语言中的标识符用来表示常量、变量、信号、端口、子程序或参数等的名称。使用标识符应遵守如下规则：

- 标识符由英文字母（a～z，A～Z）、数字（0～9）和下划线组成；
- 任何标识符必须以英文字母开头；
- 不允许出现两个以上连续的下划线，末字符不能为下划线；
- 标识符中不区分大小写字母；
- VHDL 定义的关键字不能用作标识符。

例 1.22　标识符举例：

encoder_1　　Decoder_2　　count　　mux

（2）常数

常数的定义和设置主要是为了使设计单元中的常数更容易阅读和修改。常数是一个固定的值，一旦被赋值，在程序中就不能再改变。常数说明语句的一般格式为：

CONSTANT 常数名:数据类型:=表达式;

例 1.23 常数定义举例：

```
CONSTANT VCC:REAL:=5.0;
CONSTANT count_mod:INTEGER:=10;
CONSTANT delay:TIME:=15ns;
```

常数所赋的值必须与定义的数据类型一致，否则出错。常数的适用范围取决于它被定义的位置。

（3）信号

信号是描述硬件系统的基本数据对象，它类似于电路中的连接线。信号说明语句的一般格式为：

```
SIGNAL 信号名:数据类型[约束条件:=初始值];
```

注意在上述格式中，方括号内的内容为可选项，即约束条件和初始值的设置不是必需的。

例 1.24 信号定义举例：

```
SIGNAL a,b:BIT;
SIGNAL bdate:BIT_VECTOR(15 DOWNTO 0);
```

（4）变量

VHDL 中的变量在电路中没有对应的硬件结构，它用于暂存数据，相当于一个暂存器。变量说明语句的一般格式为：

```
VARIABLE x:INTEGER;
VARIABLE count:INTEGER RANGE 0 TO 255;
```

一种语言的许多规定是一个有机的整体，相互之间存在着一定的联系。如在上述常数、信号、变量的介绍中，涉及数据类型等，它们的应用还涉及 VHDL 语言的其他内容。基于循序渐进的学习思路，此处仅对信号、变量做一般了解，当对 VHDL 语言进一步了解之后，再回头来讨论信号与变量的主要区别、使用场合等较深入的问题。

2．VHDL 的数据类型

如前所述，在 VHDL 语言中，常数、信号、变量都需要指定数据类型。因此，VHDL 语言提供了多种标准的数据类型，用户也可以自定义数据类型。这样，使 VHDL 语言的描述能力和灵活性进一步提高。但必须注意，VHDL 语言的数据类型的定义相当严格，不同类型之间的数据不能直接代入，即使数据类型相同，但位长不同时也不能直接代入。因此，在阅读 VHDL 程序时，要注意各种数据类型的定义和应用场合，以便自己能较熟练地使用 VHDL 语言编写程序。

VHDL 语言的数据类型可分为 VHDL 标准的数据类型、IEEE 标准的数据类型、用户自定义的数据类型等。

（1）VHDL 标准的数据类型

VHDL 标准的数据类型共有 10 种，如表 1-18 所示。

表 1-18　VHDL 标准的数据类型

数据类型		含义
整数	Integer	整数-(2^{31}-1)~(2^{31}-1)
实数	Real	浮点数-1.0E38~1.0E38
位	Bit	逻辑 0 或 1
位矢量	Bit-Vector	用双引号括起来的一组位数据
布尔量	Boolean	逻辑真或逻辑假，只能通过关系运算获得
字符	Character	ASCII 字符，所定义的字符量通常用单引号括起来
字符串	String	由双引号括起来的一个字符序列
正整数	Natural	整数的子集（大于 0 的整数）
错误等级	Severity Level	用于指示设计系统的工作状态

整数与数学中整数的定义相同。在 VHDL 中，整数的表示范围为-2147483647～2147483647，可进行加、减、乘、除等算术运算，不能用于逻辑运算。

布尔量没有数值的含义，不能用于算术运算，只能进行逻辑运算。布尔量数据的初始值一般总是假（FALSE）。

时间是一个物理数据，完整的时间数据包含整数和单位两部分，而且整数和单位之间至少应留一个空格的位置。例如：25 ns，10 ms。时间数据在系统仿真时，用于表示信号的延时，从而使模型系统能更逼近实际系统的运行环境。

（2）IEEE 预定义标准逻辑位与逻辑位矢量

上面介绍的 VHDL 标准数据类型 Bit 是一个逻辑型的位数据类型，这类数据取值只能是 0 和 1。而实际数字系统中存在不定状态和高阻态，为了便于仿真和描述具有三态的数字器件，IEEE 在 1993 年制定了新的标准 IEEE STD_1164，其中定义了两个重要的数据类型，即标准逻辑位 STD_LOGIC 和标准逻辑矢量 STD_LOGIC_VECTOR，规定 STD_LOGIC 型数据可以具有如表 1-19 中的 9 种不同值。

STD_LOGIC 和 STD_LOGIC_VECTOR 是在原 VHDL 语言以外添加的数据类型，因此在使用该类型数据时，在程序中必须写出库说明语句和使用包集合的说明语句。

表 1-19 STD_LOGIC 的取值及含义

STD_LOGIC 的值	说明	STD_LOGIC 的值	说明
U	初始值	W	弱信号不定
X	不定	L	弱信号 0
0	0	H	弱信号 1
1	1	-	不可能情况
Z	高阻		

（3）用户自定义的数据类型

在 VHDL 语言中，也可以由用户自己定义数据类型。用户定义数据类型的一般格式为：

TYPE 数据类型名 {,数据类型名}数据类型定义;

由用户定义的数据类型如数组类型、文件类型、时间类型等。

3．VHDL 语言的运算操作符

在 VHDL 语言中，共有 4 类操作符，可分别进行逻辑运算、关系运算、算术运算和其他运算。被操作符所操作的对象是操作数，操作数的类型应该和操作符所要求的类型一致。

（1）逻辑运算操作符

在 VHDL 语言中，逻辑运算操作符共有 6 种，其操作符和功能见表 1-20。

表 1-20 逻辑运算操作符

操作符	功能	操作符	功能
NOT	取反	NAND	逻辑与非
AND	逻辑与	NOR	逻辑或非
OR	逻辑或	XOR	逻辑异或

逻辑运算操作符的操作对象是逻辑型数据、逻辑型数组及布尔数据。在所有逻辑运算符中，NOT 的优先级别最高。

（2）关系运算操作符

在 VHDL 语言中有 6 种关系运算操作符，其操作符和功能见表 1-21。

（3）算术运算操作符

在 VHDL 语言中有 10 种算术运算操作符，其操作符和功能见表 1-22。

（4）其他运算操作符

表 1-23 中列出了 VHDL 语言中经常用到的几种其他运算操作符，事实上 VHDL 语言也规定了移位操作，有关此类运算符在后续内容中结合应用介绍。

表 1-21　关系运算操作符

操作符	功能	操作符	功能
=	等号	< =	小于等于
/=	不等号	>	大于
<	小于	> =	大于等于

表 1-22　算术运算操作符

操作符	功能	操作符	功能
+	加	MOD	求模
-	减	REM	取余
*	乘	**	乘方
/	除	ABS	取绝对值

表 1-23　其他几种运算操作符

操作符	功能	操作符	功能
< =	信号赋值	&	并置运算
:=	变量赋值	= >	关系运算符

4．VHDL 语言的基本设计单元

前面曾经指出，引入 VHDL 语言，增加了一种数字电路或系统的描述方法。比如已认识的 3 种基本逻辑运算，它们除可用真值表、逻辑表达式、电路符号等描述外，也可以用 VHDL 语言来描述。比如 $Y = ab$，若用 VHDL 语言描述则有

```
ENTITY and2 IS
    PORT(a,b:IN BIT;
         y:OUT BIT);
END and2
ARCHITECTURE example_1 OF and2 IS
BEGIN
    y<=a AND b;
END example_1;
```

观察 VHDL 对与门的描述，其程序组成可明显地分为两个部分，分别称为实体（entity）和结构体（architecture）。实体用于描述电路的输入输出端口，结构体用于描述电路的逻辑功能。当然对于复杂的数字系统，其程序组成将会稍微复杂一些，但其基本结构仍为实体和结构体。

（1）实体

实体在电路或系统中，主要是说明其输入输出端口，即使体说明部分规定了

设计单元的输入输出结构信号或引脚。实体的一般格式为：

```
ENTITY 实体名 IS
    [类属参数说明;]
    [端口说明;]
END 实体名;
```

实体描述从“ENTITY 实体名 IS”开始，至“END 实体名”结束。习惯上用大写字母表示实体的框架，此大写字母是 VHDL 语言的保留字，在程序中是不可缺省的。

在前述二输入与门的 VHDL 描述中，实体名称为 and2，端口说明部分为：

```
PORT (a,b : IN BIT;
      y : OUT BIT);
```

其中 PORT 是端口说明的关键字，a、b、y 是端口名；IN、OUT 说明端口方向，分别为输入和输出；BIT 说明数据类型是位逻辑数据类型。总之，二输入与门的实体表明，a、b 为输入端口，y 为输出端口，各端口的信号取值只能是逻辑 0 和 1。

（2）结构体

结构体具体地表明所对应实体的行为、器件及内部的连接关系，即它定义了具体的逻辑功能。结构体的一般格式为：

```
ARCHITECTURE 结构体名 OF 实体名 IS
    [定义语句 内部信号,常数,数据类型,函数等定义;]
BEGIN
    [并行处理语句];
END 结构体名;
```

一个结构体从“ARCHITECTURE 结构体名 OF 实体名 IS”开始，至“END 结构体名”结束。ARCHITECTURE 结构体名是结构体的关键字，结构体名给出了该结构体的名称，OF 后面的实体名表明了该结构体所对应的是哪个实体，用 IS 结束结构体的命名。

在前述二输入与门的 VHDL 描述中，结构体命名为 example_1，在 BEGIN 与 END 之间的并行处理语句为“y<=a AND b;”，它描述了结构体的行为及其连接关系，实际上是二输入与门的逻辑表达式的描述语句。输入信号 a、b 进行与运算，其运算结果赋值给输出信号 y。

VHDL 语言源程序最基本的设计单元仅由实体和结构体两部分组成，而这种组成形式在使用中具有一定的条件限制，即实体和结构体中所使用的数据类型必须为 STD 库中所定义的，如 BIT。而 STD 库已自动挂接在 VHDL 语言的编译器中，因而不需要在设计单元的描述中给予说明。设计单元的实体只与一个结构体

相对应。当不满足上述条件时，VHDL 语言的基本设计单元还应包括库说明、包集合说明和配置描述，即 VHDL 语言程序的完整设计单元包括五个组成部分：库说明、包集合说明、实体、结构体、配置。

库（library）是用来存放可编译的设计单元的地方，可以放置若干个程序包。库说明语句用于说明设计单元中所用到的资源库。包集合用于罗列用到的信号定义、常数定义、数据类型、器件语句等，它是一个可编译的设计单元，是库结构中的一个层次。配置语句用于描述层与层之间的连接关系及实体与结构体之间的连接关系。当一个实体中包括多个结构体时，通过配置语句来指定与相应实体对应的结构体。用 VHDL 语言进行设计的具体例子在后续内容中介绍。

本章主要讲述逻辑代数的基本运算、逻辑代数的公式和定理、逻辑函数的表示方法和逻辑函数的化简等内容。

逻辑代数的基本运算有与、或、非三种，必须掌握其逻辑功能和逻辑符号。实际逻辑问题往往比与、或、非运算复杂得多，不过它们都可以用与、或、非的组合来实现。常见的复合逻辑运算有与非、或非、与或非、异或、同或等。

逻辑函数的表示方法有逻辑函数式、真值表、逻辑图和卡诺图 4 种。这 4 种方法之间可以任意地互相转换。根据具体情况，可以选择最适当的一种方法表示所研究的逻辑函数。

逻辑函数的化简方法是本章的重点。本章先后介绍了逻辑代数的公式化简法和卡诺图化简法。公式化简法的优点是它的使用不受任何条件的限制。但是这种方法没有固定的步骤可循，规律性不强，需要一定的经验和技巧。卡诺图化简法的优点是简单、直观，而且有一定的化简步骤可循。初学者容易掌握，而且化简过程中也易于避免差错。然而，当逻辑变量超过 5 个以上时，将失去简单、直观的优点。

一般化简得到的最简形式为与－或式，除此之外，常用的逻辑函数形式还有与非－与非式、或非－或非式和与或非式等。各种形式逻辑函数之间可以相互转换。

Q-M 法的基本原理仍然是通过合并相邻最小项的方法化简逻辑函数，它有一定的化简步骤，特别适合于机器运算。

VHDL 语言是较为流行的硬件描述语言之一，本章最后介绍了 VHDL 语言基础。

习题一

1-1 逻辑代数中三种最基本的逻辑运算是什么？

1-2 什么叫真值表？它有什么用处？你能根据给定的逻辑问题列出真值表吗？

1-3 逻辑函数的表示方法共有几种？试分别说出它们之间相互转换的方法。

1-4 已知逻辑函数的真值表如表 1-24（a）、（b）所示，试写出对应的逻辑函数式。

表 1-24

（a）

A	B	C	Y
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	0

（b）

A	B	C	Y_1	Y_2
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

1-5 写出下列函数的对偶函数 Y' 及反函数 $\overline{Y}$ 。

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

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

（3）$Y=AB+\overline{A}\overline{B}$

（4）$Y=A[(B+C\overline{D})+\overline{E}]$

1-6 试用列真值表的方法证明下列异或运算公式。

（1）$A\oplus 0=A$

（2）$A\oplus 1=\overline{A}$

（3）$A\oplus A=0$

（4）$A\oplus\overline{A}=1$

（5）$(A\oplus B)\oplus C=A\oplus(B\oplus C)$

（6）$A(B\oplus C)=AB\oplus AC$

（7）$A\oplus\overline{B}=\overline{A\oplus B}=A\oplus B\oplus 1$

1-7 写出图 1-17 中各逻辑图的逻辑函数式，并化简为最简与或式。

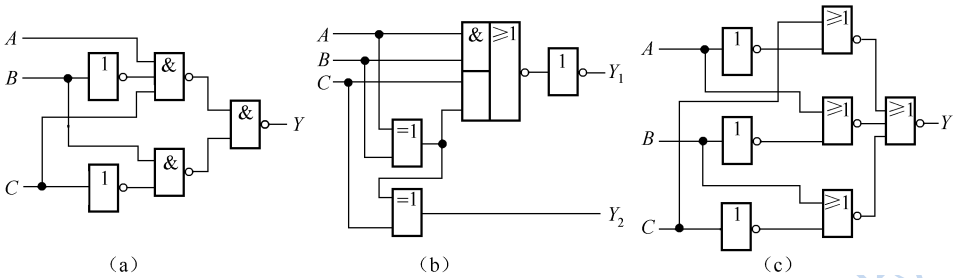


图 1-17

1-8　写出图 1-18 中各卡诺图所表示的逻辑函数式。

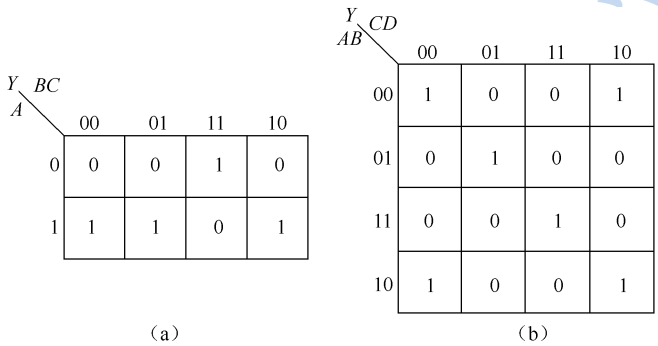


图 1-18

1-9　用逻辑代数的基本公式和常用公式将下列逻辑函数化为最简与或形式。

（1）$Y=A\overline{B}+B+\overline{A}B$

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

（3）$Y=\overline{\overline{A}BC}+\overline{A\overline{B}}$

（4）$Y=A\overline{B}CD+ABD+A\overline{C}D$

（5）$Y=A\overline{B}(\overline{A}CD+\overline{AD+\overline{B}\overline{C}})(\overline{A}+B)$

（6）$Y=AC(\overline{C}D+\overline{A}B)+BC(\overline{\overline{\overline{B}+AD}+CE})$

（7）$Y=A\overline{C}+ABD+AC\overline{D}+CD$

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

（9）$Y=B\overline{C}+AB\overline{C}E+\overline{B}(\overline{\overline{A}\overline{D}+AD})+B(A\overline{D}+\overline{A}D)$

1-10　将下列各函数式化为最小项之和的形式。

（1）$Y=\overline{A}BC+BC+A\overline{B}$

（2）$Y=A\overline{B}+C$

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

1-11 将下列各式化为最大项之积的形式。

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

（2）$Y=A\overline{B}+C$

（3）$Y=BC\overline{D}+C+\overline{A}D$

（4）$Y(A,B,C)=\sum(m_1,m_2,m_4,m_6,m_7)$

1-12 用卡诺图化简法将下列函数化为最简与或形式。

（1）$Y=ABC+AD+\overline{C}\overline{D}+A\overline{B}C+\overline{A}C\overline{D}+A\overline{C}D$

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

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

（4）$Y=\overline{A}\overline{B}+AC+\overline{B}C$

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

（6）$Y(A,B,C)=\sum(m_0,m_1,m_2,m_5,m_6,m_7)$

（7）$Y(A,B,C)=\sum(m_1,m_3,m_5,m_7)$

（8）$Y(A,B,C,D)=\sum(m_0,m_1,m_2,m_3,m_4,m_6,m_8,m_9,m_{10},m_{11},m_{14})$

（9）$Y(A,B,C,D)=\sum(m_0,m_1,m_2,m_5,m_8,m_9,m_{10},m_{12},m_{14})$

1-13 化简下列逻辑函数（方法不限）。

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

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

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

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

（5）$Y=\overline{A\overline{B}\overline{C}D+A\overline{C}DE+B\overline{D}\overline{E}+A\overline{C}\overline{D}E}$

1-14 证明下列逻辑恒等式（方法不限）。

（1）$A\overline{B}+B+\overline{A}B=A+B$

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

（3）$\overline{(A+B+\overline{C})\overline{C}D}+(B+\overline{C})(A\overline{B}D+\overline{B}\overline{C})=1$

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

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

1-15 什么叫约束项，什么叫任意项，什么叫逻辑函数式中的无关项？

1-16 将下列函数化为最简与或函数式。

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

给定约束条件为

$$A\overline{B}C\overline{D}+A\overline{B}CD+AB\overline{C}\overline{D}+AB\overline{C}D+ABC\overline{D}+ABCD=0$$

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

给定约束条件为

$$ABC+ABD+ACD+BCD=0$$

1-17　试画出用与非门和反相器实现下列函数的逻辑图。

（1）$Y=AB+BC+AC$

（2）$Y=(\overline{A}+B)(A+\overline{B})C+\overline{BC}$

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

（4）$Y=A\overline{BC}+\overline{(\overline{AB}+\overline{A}\overline{B}+BC)}$

1-18　试画出用或非门和反相器实现下列函数的逻辑图。

（1）$Y=A\overline{B}C+B\overline{C}$

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

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

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

1-19　试画出用或非门实现下列函数的逻辑图。

（1）$Y=\overline{A}\overline{B}+AC+\overline{B}C$

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

1-20　按下列要求，用门电路实现逻辑关系：

$$Y(A,B,C,D)=\sum m(1,3,4,7,13,14,15)$$

（1）与门－或门实现；

（2）与非－与非门实现；

（3）与或非门实现；

（4）或非－或非门实现。

1-21　什么是 VHDL 语言？一个完整的 VHDL 语言程序设计单元包括几个组成部分？