数字逻辑电路

主编　刘常澍
主审　赵雅兴
编写　刘常澍　尤晓明
　　王　炜　郑　桐

·北京·

图书在版编目(CIP)数据

ISBN 7-118

中国版本图书馆 CIP 数据核字(9　)第　　号

(北京市海淀区紫竹院南路 23 号)
(邮政编码　100044)
北京奥隆印刷厂印刷
新华书店经售
*
开本 787×1092　1/ 16　印张 21　483 千字
2002 年 2 月第 1 版　　2002 年 2 月北京第 1 次印刷
印数:1—4000 册　　定价:31.00 元

内 容 简 介

本书系按照大学本科数字电路课程教学基本要求而编写。

全书共 7 章:数字逻辑基础,晶体管开关与逻辑门电路,触发器与波形变换、产生电路,组合逻辑电路,时序逻辑电路,存储器与可编程逻辑器件,可测性设计及边界扫描技术。本书特点:基础理论深入浅出,注重实践,附有大量例题和习题,最后两章介绍了新的数字技术和器件。重视采用国家标准图形符号,并对其所表示的意义进行简要解释,使读者在学习本书过程中逐渐学会认读常用的逻辑符号。

本书适合高等工科院校电子信息、通信、自动化等专业学生用作技术基础课教材及考研参考书,也可供其他相关专业师生或社会读者阅读参考。

前　言

人类社会已进入信息时代。信息的处理、存储、传输要靠电子信息技术、计算机技术、网络通信技术等来完成,而这些技术都是以数字技术为基础的。数字系统和数字设备已广泛应用于各个领域,并且是必不可少的和关键的部分。因而电子信息技术、计算机技术以及相关技术领域的工程师和技术人员必须掌握数字系统的基础知识。

目前我国经济建设正以前所未有和世界少有的速度健康发展,需要大量各个层次的科技人才,因而教育的发展非常迅猛。近年来高等院校扩大招生规模,各类教育也蓬勃发展,对于各种教材需求迫切,为此我们在多年教学实践的基础上,按照本科生数字电路课程教学基本要求编写了本书,可供电子信息、通信、自动化等专业及相关专业教学使用。

全书共有 7 章内容:数字逻辑基础,晶体管开关与逻辑门电路,触发器与波形变换、产生电路,组合逻辑电路,时序逻辑电路,存储器与可编程逻辑器件,可测性设计及边界扫描技术。前 3 章对于基础内容做了必要的讲述,以够用为度。组合逻辑电路与时序逻辑电路两(5、6)章内容较多,可在课内讲授基本概念和原理部分,一些基本内容的扩展和例题可以让学生自学。存储器与可编程逻辑器件近年来发展迅猛,已经成为微电子技术进步的重要标志。第 6 章对一些新发展进行了简要介绍。第 7 章介绍近些年发展起来的数字系统的新技术,即可测性设计及边界扫描技术,涉及到电路设计与测试融为一体的设计概念,其电路形式与传统的电路形式有很大区别,在讲授时可视具体情况掌握取舍,也可在另设的相关课程中安排。许多院校将 A/ D、D/ A 的内容安排在电子测量或其他课程中,故本书未将其编入。

书中采用国家标准图形符号,在出现符号的地方对其所表示的意义进行简要解释,使读者在学习本书的过程中逐渐学会认读常用的逻辑符号。书中备有大量例题和习题,对于自学者提供更多的参考和练习。

本书由赵雅兴教授和刘常澍组织选定内容,列出目录,由尤晓明编写第 1 章和第 4 章,王炜编写第 2 章和第 3 章,郑桐编写第 6 章,刘常澍编写第 5 章和第 7 章内容。赵雅兴教授审阅了全部书稿,并提出了许多建议和修改意见。由刘常澍对全书校订统编。

本书编写过程中杨淑琴、张耀娟、刘军参加了绘图、编程等许多必要的工作。由于编者的经验不丰、水平有限,书中难免存在不妥乃至错误之处,敬请广大读者不吝赐教。

编　者

2001 年 9 月于天津

目　　录

第 1 章　数字逻辑基础

引　　言

现代电子工程中,数字电路的应用越来越多。因为数字电路中电压和电流只有两个状态,如电压的高或低、电流的通或断,在逻辑电路中可以很方便地用 1 和 0 两个数字表示。这种电路表现出许多优点。

①工作状态的不连续性,使电路可靠性和稳定性大为提高。

②从理论上讲,信号处理的过程中不会产生失真。

③信息的传输、运算、处理和保存变得更为方便。

④对电路的实时控制准确有效。

⑤与计算机及外围电路兼容,便于利用计算机进行运算和控制处理。

⑥数字电路可以很方便地级联和扩展,中、大规模的数字集成电路的生产和应用都呈现出广阔的空间。

正是基于上述原因,数字电路在近年来得到长足的进步,使得电子设备的可靠性和准确性得到充分实现。

1.1　数字信号及数字电路

1.1.1　模拟量与数字量

在我们周围的自然界中存在的物理量千变万化,但就其变化规律而言,可以分成模拟量和数字量两大类。模拟量是在时间上和数值上连续变化的物理量,如温度、海拔和气压等等。数字量是时间上和数值上都是不连续的、或者说是离散的物理量,例如对生产线上生产产品的计件、人口统计等。

模拟量的数字化是对模拟量分离取值的过程。比如气温,一般只按整度数记录,最小的统计单位是“度”,而实际气温变化是连续的。所以,记录气温的过程实际是对模拟量数字化的结果。

数字量有一个最小数量单位,每次数值大小变化都是它的整数倍,而小于这个最小数量单位的数值没有任何意义。比如人口统计中的最小数量单位是一人、千人或万人。

1.1.2　数字信号和数字电路

表示数字量的信号叫做数字信号,工作在数字信号下的电子电路叫做数字电路。数字信号只有 0 和 1 两个量,用电路状态表示这两个量非常方便,如电压的高低、晶体管的

饱和与截止、开关的通断等等。由于数字电路中只有 0 和 1 两个数字,在数字运算时采用二进制数制,与我们习惯的十进制有所不同。相对模拟电路而言,数字电路具有误差小、抗干扰性强、精度高、数据容易保存等优点。

1.2 二进制数

1.2.1 二进制数表示法

通常我们在计数的过程中,会采取一定的计数规则,这就是数制。例如,日常生活中通常采用的数制是十进制,它有 0,1,…,9 等 10 个数字,计数的规则就是“逢十进一”。即在计数过程中,一旦计数满十,就向高位进一。可以用这样的表示式来表示一个 n 位整数、m 位小数的十进制数

$$(D)_{10} = k_{n-1}\times 10^{n-1} + k_{n-2}\times 10^{n-2} + \cdots + k_1\times 10^1 + k_0\times 10^0 + k_{-1}\times 10^{-1} + k_{-2}\times 10^{-2} + \cdots + k_{-m}\times 10^{-m} = \sum_{i=-m}^{n-1} k_i\times 10^i \tag{1-1}$$

式中,k_i 为第 i 位的系数;10 为基数;10^i 为第 i 位的权。我们所说的不同的数,主要体现出权的大小,如十位、百位、千位等等;其次是系数不同。任意一个十进制数可按位展开:即把每一位的位权值与各自的系数相乘,然后对每一项求和。如

$$(1234.056)_{10} = 1\times 10^3 + 2\times 10^2 + 3\times 10^1 + 4\times 10^0 + 0\times 10^{-1} + 5\times 10^{-2} + 6\times 10^{-3}$$

除了十进制外,人们根据计数的不同要求,也采用十二进制、六十进制等等。按照上述方法,我们可以写出任意进制数的表示式

$$(D)_N = k_{n-1}\times N^{n-1} + k_{n-2}\times N^{n-2} + \cdots + k_1\times N^1 + k_0\times N^0 + k_{-1}\times N^{-1} + k_{-2}\times N^{-2} + \cdots + k_{-m}\times N^{-m} = \sum_{i=-m}^{n-1} k_i\times N^i \tag{1-2}$$

上式可表示为一个 N 进制的数,其系数为 0,1,…, N - 1。

在数字电路和计算机中,都采用二进制计数,即采用“逢二进一”的计数方法。同样,我们可以用下式表示一个二进制数

$$(D)_2 = \sum_{i=-m}^{n-1} k_i\times 2^i \tag{1-3}$$

二进制数的基数是 2,系数只有 0 和 1 两个数字,在数字电路和计算机中可以很方便地进行处理运算。但二进制数通常位数很多,而且与人们习惯的计数方法不尽相同,因而有时需要二进制数与其他进制数进行相互转换,以达到不同的应用目的。

1.2.2 二进制数和十进制数的互相转换

1. 二进制数 — 十进制数转换

把一个二进制数转换成十进制数的过程很简单,只需将二进制数按位(权)展开相加

即可。

例 1-1　将二进制数$(11010.011)_2$转换成十进制数。

解:$(11010.011)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} = (26.375)_{10}$

2　十进制数 — 二进制数转换

十进制数转换到二进制数的过程必须分两步走,要对整数部分和小数部分分别进行转换,然后再相加得到结果。

整数部分的转换过程如下:一个十进制的整数$(D)_{10}$总可以用二进制数展开,即

$$(D)_{10} = k_n2^n + k_{n-1}2^{n-1} + \cdots + k_12^1 + k_02^0$$

其二进制数写为$(k_n\ k_{n-1}\cdots k_0)_2$。若把$(D)_{10}$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;然后,再把商数除以 2,则所得余数即 k_1;…。依次将每次得到的商反复除以 2,取它们的余数 $k_0\cdots k_n$,就可求出二进制数的每一个数位了。所以,十进制数(整数部分)转换为二进制数的过程是采用逐次除以基数 2,再取余数的方法。

例 1-2　将十进制数$(27)_{10}$转换成二进制数。

解:按上述步骤进行

	商		余数	
2	27			
2	13	←	……1 = k_0	最低位
2	6		……1 = k_1	
2	3		……0 = k_2	
2	1		……1 = k_3	
	0		……1 = k_4	最高位

故$(27)_{10} = (11011)_2$。

小数部分的转换与整数部分的转换不同,是通过基数乘法实现的。

若$(D)_{10}$是一个十进制数的小数,对应的二进制小数为$(0.k_{-1}\ k_{-2}\cdots k_{-m})_2$,应有

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

如果将两边同乘以 2 得到

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

说明将小数$(D)_{10}$乘以 2 后,等式右边所得乘积的整数部分即 k_{-1},其余为小数部分。

我们取出 k_{-1},将小数部分再乘以 2 又可以得到

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

同样取出 k_{-2},将小数部分再乘以 2,…,当乘积为 0 后,就依次得到 $k_{-1}\ k_{-2}\cdots k_{-m}$。所以,十进制数(小数部分)转换为二进制数的过程是采用逐次乘以基数 2,再取整数的方法。

例 1-3　将十进制小数$(0.625)_{10}$转换成二进制小数。

解:按上述步骤进行

$$
\begin{array}{r}
0.625 \\
\times \quad 2 \\
\hline
1.250\cdots\cdots \text{取 } 1 \\
\times \quad 2 \\
\hline
0.500\cdots\cdots \text{取 } 0 \\
\times \quad 2 \\
\hline
1.000\cdots\cdots \text{取 } 1
\end{array}
$$

故$(0.625)_{10} = (0.101)_2$

有时,二进制数的位数很多,计算达到一定的精度即可。

例 1-4　将$(0.4)_{10}$转换成二进制小数。

解:按上述步骤进行

$$
\begin{array}{r}
0.4 \\
\times \quad 2 \\
\hline
0.8\cdots\cdots \text{取 } 0 \\
\times \quad 2 \\
\hline
1.6\cdots\cdots \text{取 } 1 \\
\times \quad 2 \\
\hline
1.2\cdots\cdots \text{取 } 1 \\
\times \quad 2 \\
\hline
0.4\cdots\cdots \text{取 } 0 \\
\cdots \qquad \cdots
\end{array}
$$

以下循环。故$(0.4)_{10} = 0.01100110\cdots$　　取需要的数位即可。

1.2.3　十六进制数

用二进制数表示一个数字位数会很长,书写和记忆都很不方便,因而可以把二进制数化成八进制或十六进制数。

八进制数由 0,1,…,7 八个数字组成,根据数制规律可知,1 位八进制数可表示成 3 位二进制数,因而转换时只需将二进制数的每 3 位对应转换成 1 位八进制数。当然带小数的数转换时,整数与小数应分别进行转换。即以小数点为基准,向左、向右每 3 位为一组(首尾不足 3 位的补足 3 位),每组对应转换成 1 位八进制数。

例 1-5　将$(10110.01011)_2$转换成八进制数。

解: $(010110.010\ 110)_2$

$(2\quad 6.\quad 2\quad 6)_8$

所以$(10110.01011)_2 = (26.26)_8$

十六进制数由 0,1,…,9 和 A,B,C,D,E,F 共十六个数字组成,其中 A,…,F 分别等值于十进制数中的 10,…,15 等数字。4 位二进制数可转换为 1 位十六进制数,转换方法可参照八进制数的转换方法进行,但须将二进制数的每 4 位对应转换成 1 位十六进制数。例如将二进制数 1011010.01111 转换成十六进制数是$(5A.78)_{16}$

欲将八进制数和十六进制数转换成十进制数,只需像二进制数转换时那样,按位权展

开求和即可。例如:将十六进制数$(5A.78)_{16}$转换成十进制数

$$(5A.78)_{16} = 5\times16^{1} + 10\times16^{0} + 7\times16^{-1} + 8\times16^{-2} = (90.46875)_{10}$$

十进制数直接转换成八进制和十六进制数比较繁琐,有些计算容易出错。一般是先化成二进制数,然后再变成八进制和十六进制数。例如:将$(43.3125)_{10}$转换成八进制和十六进制数

$$(43.3125)_{10} = (101011.0101)_{2} = (53.24)_{8} = (2B.5)_{16}$$

1.3 码制与编码

我们知道计算机可以存储和处理很多不同的信息,而实际上机器只能识别二进制数,因而我们要用数来表示各种各样的信息。用数来表示信息的方法很多,建立这种对应关系的过程就叫做编码,编码所依据的不同的编码规则就称为码制。所以,数码不仅可以表示数量的不同大小,而且还能表示不同的事物。表示事物的这些数码只是事物的代号,而没有数量含意。例如,不同公交车的行车编号,只是区别不同的行车路线,并不表示数量大小。代替不同事物的数码就叫代码。当然,数量也可以用数码来表示,但可以有不同的计数规则,甚至同一个数可以有不同的代码,因为依据的码制不同。

1.3.1 原码、反码和补码

我们书写一个带符号数时,可以在数的前面加上一个符号,如 + 3、- 0.5 等等。而计算机当中的正负号是用数码来表示的。通常带符号数的最高位为符号位:是 0 表示为正数;是 1 则为负数。符号位后面表示的是数值。在数字电路中,带符号数通常有原码、反码和补码三种表示方法。

1. 原码

用原码表示带符号数时,只需将符号位用 0 或 1 表示即可,后面的数字不变。如有两个带符号数 $N_1 = +1100101$ 和 $N_2 = -1100101$,用原码表示为

$$N_1 = (01100101)_{原} \qquad N_2 = (11100101)_{原}$$

原码表示方法简单,但在做运算时比较麻烦(尤其是减法运算),运算过程加长,造成电路成本增加和运算速度降低。为此,可采用补码表示方法来加以改善。

2. 反码

用反码表示一个数时,正数的表示方法与原码表示方法相同;如果是负数,最高位仍为符号位,记为 1,其余各位把原数值按位取反即可。如

$$N_1 = +1100101 = (01100101)_{反}$$

$$N_2 = -1100101 = (10011010)_{反}$$

3. 补码

在补码表示中,正数的表示方法也与原码表示方法相同;如果是负数,符号位仍为 1,其余各位不用数值本身,而取对 2^N 的补数。由于是二进制数,求负数的补码可先将其变成

反码,然后在最低位加 1 即可。例如

$$N_1 = +1100101 = (01100101)_{补}$$

$$N_2 = -1100101 = (10011011)_{补}$$

三种表示方法如表 1-1 所示。

表 1-1　原码、反码和补码的对应关系

十进制	二　进　制			十进制	二　进　制		
	原码	反码	补码		原码	反码	补码
-7	1111	1000	1001	0	0000	0000	0000
-6	1110	1001	1010	1	0001	0001	0001
-5	1101	1010	1011	2	0010	0010	0010
-4	1100	1011	1100	3	0011	0011	0011
-3	1011	1100	1101	4	0100	0100	0100
-2	1010	1101	1110	5	0101	0101	0101
-1	1001	1110	1111	6	0110	0110	0110
-0	1000	1111	0000	7	0111	0111	0111

原码表示方法简单,符合人们通常的记数习惯,但 0 的表示不是唯一的。补码的转换虽然稍微麻烦一些,但是 0 的表示方法是唯一的,而且补码减法运算可以用加法来实现。

4. 小数的表示与字长

不论原码、反码或补码,都是按进位关系和整数连续排列起来,但是当两个数进行运算时,必须整数和小数字长分别相等,即两个数的整数位数应相同,小数的位数也应相同。如果两个数的位数不同时或在运算时产生了进位超出了原来的位数(称为溢出),就应增加位数少的数的字长(扩展位数)。

扩展位数应整数、小数分别扩展,扩展的方法如下。

① 正数:无论带符号和不带符号的原码、反码和补码表示的正整数,一律在高位填 0 补足所少的位数;正小数,一律在低位填 0 补足所少的位数;

② 原码带符号的负数:整数在符号位后的高位填 0 补足所少的位数,小数在低位填 0 补足所少的位数;

③ 反码带符号的负数:整数在符号位后的高位填 1 补足所少的位数,小数在低位填 1 补足所少的位数;

④ 补码带符号的负数:整数在符号位后的高位填 1 补足所少的位数,小数在低位填 0 补足所少的位数。

例如,7.5 和 4.6875 的正、负数各种表示由原来的 4 位字长扩展为 8 位字长。

正数:7.5—0111.1000,4.6875—0100.1011 扩展后为

7.5—00000111.10000000,4.6875—00000100.10110000

负数原码:-7.5—1111.1000,-4.6875—1100.1011 扩展后为

-7.5—10000111.10000000,-4.6875—10000100.10110000

负数反码：-7.5—1000.0111，-4.6875—1011.0100 扩展后为

-7.5—11111000.01111111，-4.6875—11111011.01001111

负数补码：-7.5—1000.1000，-4.6875—1011.0101 扩展后为

-7.5—11111000.10000000，-4.6875—11111011.01010000

例 1-6　有两个数

$N_1 = +1011000$，$N_2 = +0100110$

计算 $N_1 - N_2$ 的值。

解：$N_1 - N_2$

$= N_1 + (-N_2)$

$= N_{1补} + (-N_2)_{补}$

$= 01011000 + 11011010$

用竖式表示

$$\begin{array}{r} 01011000 \\ +\ 11011010 \\ \hline \boxed{1}\ 00110010 \end{array}$$

$N_1 - N_2 = +0110010$

例 1-7　有两个数：

$N_1 = +0100110$，$N_2 = +1010010$

计算 $N_1 - N_2$ 的值。

解：$N_1 - N_2$

$= N_1 + (-N_2)$

$= N_{1补} + (-N_2)_{补}$

$= 00100110 + 10101110$

用竖式表示

$$\begin{array}{r} 00100110 \\ +\ 10101110 \\ \hline 11010100 \end{array}$$

$N_1 - N_2 = -0101100$

1.3.2　二—十进制(BCD)码

我们通常熟悉的是十进制数，而数字电路中采用的是二进制的计数方法，为了用二进制数表示十进制数，常采用 BCD(Binary Coded Decimal)码的表示方法。1 位十进制数最少要 4 位二进制数来表示，即可以从 0000、…、1111 中任取十个数来表示 0、1、…、9。通常的一种编码方法是取前10个数，称为8421BCD码，其对应关系见表1-2。它是一种含权码，即将二进制数按位相加就可得到相应的十进制数。这种码表示意义明显，转换也很方便，因而应用最多。除了 8421 码外，还有几种 BCD 码，分别介绍一下。

含权码还有 2421 码、5211 码等，但与 8421 码不同，它们的编码方式不是唯一的。如 2421 码中的 7 既可以表示成 1101，也可以表示成 0111。5211 码中 7 可以表示成 1100 或 1011。

余 3 码是一种无权码，将 8421 码表示的数码加 3 即得到相应的余 3 码。用余 3 码作加法时，若两个十进制数之和为 10 应产生进位，而两个对应余 3 码之和恰好等于二进制数的 16，于是高位进位而不再进行修正。另外，0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 互为反码(2421 码也有相似的特性)，这对于求取对 10 的补码运算非常方便。

表 1-2 列出了几种常用的 BCD 代码。

1.3.3　格雷(Gray)码

格雷码是一种无权码，也称为循环码。其特点是：每两个相邻代码中的数码仅有一位不同，其余各位均相同。而且首尾(0 和 15)两个代码也仅有一位不同，构成“循环”。显然，在数码变化时采用格雷码可大大减少错码的可能性。表 1-3 列出了 4 位格雷码与十进制数的对应关系。

表 1-2　几种常用的 BCD 代码

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

表 1-3　格雷码数码表

十进制数	格雷码	十进制数	格雷码
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

1.4　逻辑代数基本知识

逻辑代数最初是由英国数学家布尔(G. Boole)首先提出来的,被称为布尔代数。后来香农(Shannon)将布尔代数用到开关矩阵电路中,因而又称为开关代数。现在逻辑代数被广泛用于数字逻辑电路和计算机电路的分析和设计中,成为数字逻辑电路的理论基础。逻辑代数的变量称为逻辑变量。逻辑变量与一般代数变量不同,逻辑变量的取值只有 0 和 1,就是说逻辑电路中只有两种逻辑状态。这里的 1 和 0 可以由数字系统中的电平的高低、开关的断通和信号的有无来表示。因而,它们已没有数量大小的概念,只表示两种不同的逻辑状态。

1.4.1　基本逻辑运算

逻辑运算中,最基本的运算是与、或、非三种。

1. 与逻辑

我们举例说明与的逻辑关系。由图 1-1 中给出的控制电路可以看出:只有当两个开关同时闭合时,灯才会亮,否则,灯就不亮。从而可以得出这样的因果关系:只有当决定事物某一结果的全部条件同时具备时,这个结果才会发生,这种因果关系叫做逻辑与,或者叫逻辑乘。这种逻辑关系表达式为

$$Y = A \cdot B \tag{1-4}$$

式中,Y 是逻辑函数;A 和 B 是逻辑变量。运算符号“·”表示与逻辑,与逻辑运算要有两个以上的变量。我们可以把开关的接通和断开分别设为 1 和 0,这样 A 和 B 可以有 1 和 0 两个值;而灯的亮、灭也能表示为 1 和 0,用 Y 来表示。把所有逻辑变量和逻辑函数的状态值列出来做成表格,称为真值表。真值表对于分析逻辑关系、简化逻辑运算都是非常有用的。二变量的与逻辑真值表如表 1-4 所列,它清楚地表明了与逻辑关系。数字逻辑电路中的与逻辑符号如图 1-4(a)所示。

表 1-4　与逻辑真值表

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

图 1-1　具有与逻辑关系的电路

2　或逻辑

或逻辑关系也可以用电灯的控制电路反映出来。如图 1-2 所示,当有一个以上的开关接通后,灯就会亮。所以,当在决定事物某一结果的各个条件中,只要有一个或更多的条件满足,结果就会发生,这种因果关系叫做逻辑或,也叫做逻辑加。这种逻辑关系表达式为

$$Y = A + B \tag{1-5}$$

运算符号“ + ”表示或逻辑,或逻辑运算也要有两个以上的变量。表 1-5 给出了或逻辑的真值表,可以看出其逻辑关系。或逻辑符号如图 1-4(b)所示。

3　非逻辑

非逻辑关系表示的是一种变量和函数互为逆运算的过程,一如数码的取反运算。在图 1-3 电路中,开关断开时,指示灯亮,开关闭合时指示灯反而不亮。此例表明,只要条件具备了,结果便不发生;而条件不具备时,结果一定发生。这种因果关系叫做逻辑非,也叫做逻辑反、逻辑否。这种逻辑关系表达式为

$$Y = \overline{A} \tag{1-6}$$

图 1-2　具有或逻辑关系的电路

图 1-3　具有非逻辑关系的电路

运算符号“ - ”表示非逻辑,非逻辑运算可以是一个变量。从真值表可以看出非逻辑的逻辑关系。非逻辑符号如图 1-4(c)所示。

表 1-5　或逻辑真值表

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

表 1-6　非逻辑真值表

A	Y
0	1
1	0

数字逻辑电路中,与、或、非逻辑运算用图 1-4 所示的逻辑图形符号表示,图中(1)为国家标准规定的符号;(2)为过去沿用的图形符号;(3)为部分国外资料中常用的图形符