面向21世纪高等学校电子技术基础教材

数字电子技术基础

主　编　付　华　高迎慧
副主编　韩伟韬　顾德英　朱凤龙

东北大学出版社

内 容 简 介

本书共分 9 章，系统地介绍了数字电子技术的基本理论、数字电路的分析与设计方法、集成数字电路的功能与应用。还介绍了半导体存储器及 PAL, GAL, FPGA, CPLD 等各类可编程器件的结构特点、工作原理和使用方法。

除第 9 章外，各章均有练习题及本章小结，便于学生课后复习之用。本书与《模拟电子技术基础》配套使用，教学时数 80 学时左右（不含实验）。

本书可作为理工科高等学校的教学用书。

图书在版编目(CIP)数据

数字电子技术基础/付华等编 .—沈阳：东北大学出版社，2002.4
ISBN 7-81054-752-6

Ⅰ.数… Ⅱ.付… Ⅲ.数字电路-电子技术 Ⅳ.TN79

中国版本图书馆 CIP 数据核字(2002)第 097129 号

出 版 者：东北大学出版社
（邮编：110004 地址：沈阳市和平区文化路 3 号巷 11 号）
出 版 人：李鑫兴
印 刷 者：东北大学印刷厂
发 行 者：新华书店总店北京发行所发行
开 本：787mm×1092mm 1/16
字 数：464 千字
印 张：18.125
出版时间：2002 年 4 月第 1 版
印刷时间：2002 年 4 月第 1 次印刷
责任编辑：孟 颖
责任出版：杨华宁
封面设计：唐敏智

定 价：25.00 元

垂询电话：024—83680267（社务办） 024—83680265（传 真）
83687331（市场部） 83687332（出版部）
E-mail：neuph@neupress.com
http://www.neupress.com

前　言

电子技术分为模拟电子技术和数字电子技术，数字电子技术是当前发展最快的学科之一。就逻辑器件而言，已从20世纪40年代的电子管、50年代的晶体管、60年代的小规模集成电路，发展到中规模集成电路、大规模集成电路、超大规模集成电路及可编程逻辑器件，这为数字电路设计提供了更加完善、方便的器件。相应地，数字电路的设计过程和设计方法也在不断地改进和发展。由于半导体技术的迅速发展和微型计算机的广泛应用，数字电子技术在现代科学技术领域占有很重要的地位，在各个领域中得到了广泛的应用。

本教材是根据国家教育部主持制定的电子技术基础课程的教学基本要求编写的。在内容选取和材料安排上，突出基本概念、基本理论和基本方法。在明确阐述基本概念、基本方法的同时，列举典型例题，以便读者加深对各知识要点的理解，并能熟练应用。在阐述集成电路时，重点介绍其外部特性和使用方法；适当削减小规模集成电路的内容，扩充中、大规模集成电路的内容，使本书的内容适应当前脉冲数字技术的发展现状。本书还在每一章的小结中进一步强调了本章的重点内容和各个知识点之间的联系，各章后面还附有较多的习题。

本书由付华、高迎慧任主编。其中第1章、第2章、第3章由付华编写，第8章由付华、高迎慧合作编写，第4章、第5章由韩伟韬编写，第6章由高迎慧编写，第7章由顾德英、周立合作编写，第9章由朱凤龙编写。主编提出全书编写大纲并对全书做了统稿工作。徐耀松为本书做了许多具体工作。

由于编者水平所限，书中难免存在缺点和不足，敬请广大读者批评指正。

编　者

2001年10月

目　录

第 1 章　逻辑代数基础

1.1　概　　述

1.1.1　数字信号与数字电路

1.1.1.1　数字信号与模拟信号

在自然界诸多的物理量中，按其变化规律与时间的相应联系，可分为模拟量和数字量两大类。

所谓数字量，是指在时间上和数量上都是离散的。也就是说，它们的变化在时间上是不连续的，只发生在一系列离散的时间上。同时，它们的数值大小和每次的增减变化都是某一个最小数量单位的整数倍，小于这个最小数量单位的数值没有任何物理意义。数字量是离散的，只能按有限个数或可数的量化单位(又称为增量、量化层)取值。量化单位的选择取决于所要求的精度。与数字量相对应的电信号称为数字信号。用来传递、加工和处理数字信号的电路称为数字电路(Digital Circuit)，它可用来实现各种逻辑功能。例如，教室里学生的人数为数字量，其最小的数量单位就是 1 个；用电子电路记录从自动生产线上输出的零件数目(数字量)时，每送出一个零件便给电子电路一个信号，使之记 1，而平时没有零件送出时加给电子电路的信号 0，则不记数。可见，零件数目这个信号是一个数字信号，因为它无论在时间上还是在数量上都是不连续的。

所谓模拟量，是指在时间上或数值上都是连续的。表示模拟量的信号叫做模拟信号，把工作在模拟信号下的电子电路称为模拟电路。例如，气温这种物理量是模拟量，因为气温的变化总是连续的。热电偶在工作时输出的电压信号是模拟信号，因为在任何情况下被测温度都不可能发生突跳，所以测得的电压信号无论在时间上还是在数量上都是连续的。而且，这个电压信号在连续变化过程中的任何一个取值都有具体的物理意义，即表示一个相应的温度。

1.1.1.2　数字电路的特点

在进行信息传递和处理时，数字方式与模拟方式相比，具有精度和可靠性高、使用灵活、易于器件标准化等优点。与模拟电路相比，数字电路具有以下一些特点。

① 数字电路中一般都采用二进制。二进制只有两个数码 0 和 1，因此它的每一位数都可以用任何具有两个不同稳定状态的元件来表示，也就是说，凡是具有两个稳定状态的元件，其两种不同的状态都可以用来表示二进制数的两个数码 0 和 1。如三极管的饱和与截止，继电器接点的闭合和断开，灯泡的亮和不亮等。只要规定其中一种状态表示 1，另一种状态表示 0，就可以表示二进制数。二进制的数字装置所用元件少，所以数字电路基本单元电路简单，对电路中各元件参数的精度要求不高，只要能正确区分两种截然不同的状态即可。因此数字电路集成化较容易，且可提高集成度。

② 数字电路抗干扰能力强。因为数字电路传递、加工和处理的是只有 0 和 1 的二值信号，不易受外界的干扰。数字电路可以用增加二进制数的位数的方法来提高电路的精度，因此，信号在传递、加工和处理过程中可以较容易地提高精度。

③ 数字信号便于长期存储。目前广泛使用的各种硬磁盘、软磁盘及光盘等都是存储二值信号的部件，这样，既可以使大量宝贵的信息资源得以妥善保存，又可以大大压缩存储空间，使用起来十分方便。这个优点在当今信息时代尤其重要。

④ 数字信号保密性好。在数字电路中可以进行加密处理，使可贵的信息资源不被窃取。

⑤ 在数字电路中，可以采用标准的逻辑部件和编程逻辑器件来构成各种数字系统，设计方便，使用灵活。

由于数字电路具有上述特点，因此其发展十分迅速，在电子计算机、数控技术、网络技术、通信技术、数字仪表等各方面都得到了越来越广泛的应用。

1.1.2 数制和码制

1.1.2.1 常用数制

用数字量表示物理量的大小时，仅有一位数码往往不够用，因此经常需要用进位计数的方法组成多位数码使用。通常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计数制，简称数制。下面介绍几种常用的数制。

(1) 十进制

十进制是人们最熟悉的一种数制，也是数字电路中经常使用的进位计数制。在十进制数中，每一位有 0～9 十个数码，所以计数的基数是 10。超过 9 的数必须用多位数表示，其中低位和相邻高位之间的关系是“逢十进一”，故称为十进制。例如

$$345.67 = 3\times10^2 + 4\times10^1 + 5\times10^0 + 6\times10^{-1} + 7\times10^{-2}$$

所以任意一个十进制数 D 均可展开为

$$D = \sum k_i \times 10^i \tag{1-1}$$

其中 k_i 是第 i 位的系数，它可以是 0～9 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(1-1)中的 10，即可得到任意进制(N 进制)数展开式的一般形式

$$D = \sum k_i N^i \tag{1-2}$$

式中 i 的取值与式(1-1)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。

从计数电路的角度看来，采用十进制是不方便的。因为构成计数电路的基本想法是把电路的状态跟数码对应起来，而十进制的十个数码，必须由十个不同的而且能严格区分的电路状态与之对应，这样将在技术上带来许多困难，而且也不经济。因此在计数电路中一般不直接采用十进制。

(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(1-2)，任何一个二进制数均可展开为

$$D = \sum k_i \times 2^i \tag{1-3}$$

并可计算出它所表示的十进制数的大小。例如

$$(1101.11)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} = (13.75)_{10}$$

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

可见，二进制的基本运算规则简单，运算操作简便。因为二进制具有一定的优点，所以它在计算机技术中被广泛采用。但是，采用二进制也有一些缺点。用二进制表示一个数时，位数多，例如，十进制数 49 表示为二进制数时，即为 110001，使用起来不方便也不习惯。因此，在运算时原始数据多用人们习惯的十进制数，在送入机器时，就必须将十进制的原始数据转换成数字系统能接受的二进制数。而在运算结束后，再将二进制转换为十进制数，表示最终结果。

(3) 十六进制

十六进制数的每一位有十六个不同的数码，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

$$D = \sum k_i \times 16^i \tag{1-4}$$

并由此式计算了它所表示的十进制数值。例如

$$(3FA.5)_{16} = 3\times16^2 + 15\times16^1 + 10\times16^0 + 5\times16^{-1}$$

$$= 768 + 240 + 10 + 0.3125 = (1018.3125)_{10}$$

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

式中的下脚注 16 表示括号里的数是十六进制，有时也用 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 位十进制进行运算，而 8 位、16 位和 32 位的十进制数可以用 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简便。

1.1.2.2 数制转换

(1) 十进制数与非十进制数的转换

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(1-3)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

$$(11010.01)_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2} = (26.25)_{10}$$

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(1-3)可知

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

$$= 2(k_n2^{n-1} + k_{n-1}2^{n-2} + \cdots + k_1) + k_0 \tag{1-5}$$

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

同理，将式(1-5)中的商除以 2 得到的新的商为

$$k_n2^{n-1} + k_{n-1}2^{n-2} + \cdots + k_1 = 2(k_n2^{n-2} + k_{n-1}2^{n-3} + \cdots + k_2) + k_1 \tag{1-6}$$

由式(1-6)不难看出，若将$(S)_{10}$除以 2 所得的商再除以 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$(173)_{10}$化为二进制数可如下进行

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

故 $$(173)_{10}=(10101101)_2。$$

下面介绍小数的转换方法。

若$(S)_{10}$中一个十进制的小数，对应的二进制小数为$(0.k_{-1}k_{-2}\cdots k_{-m})_2$，则根据式(1-3)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

$$2(S)_{10}=k_{-1}+(k_{-2}2^{-1}+k_{-3}2^{-2}+\cdots+k_{-m}2^{-m+1}) \tag{1-7}$$

式(1-7)说明，将小数$(S)_{10}$乘以2所得乘积的整数部分即k_{-1}。

同理将乘积的小数部分再乘以2又可得到

$$2(k_{-2}2^{-1}+k_{-3}2^{-2}+\cdots+k_{-m}2^{-m+1})=k_{-2}+k_{-3}2^{-1}+\cdots+k_{-m}2^{-m+2} \tag{1-8}$$

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

$$
\begin{array}{r}
0.8125 \\
\times \quad 2 \\
\hline
1.6250 \quad \cdots\cdots \text{整数部分}=1=k_{-1}
\end{array}
$$

$$
\begin{array}{r}
0.6250 \\
\times \quad 2 \\
\hline
1.2500 \quad \cdots\cdots \text{小数部分}=1=k_{-2}
\end{array}
$$

$$
\begin{array}{r}
0.2500 \\
\times \quad 2 \\
\hline
0.5000 \quad \cdots\cdots \text{整数部分}=0=k_{-3}
\end{array}
$$

$$
\begin{array}{r}
0.5000 \\
\times \quad 2 \\
\hline
1.0000 \quad \cdots\cdots \text{小数部分}=1=k_{-4}
\end{array}
$$

故 $$(0.8125)_{10}=(0.1101)_2。$$

③ 十六进制数与十进制数的转换。在将十六进制数转换为十进制数时，可根据式(1-4)

将各位按权展开后相加求得。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。

④ 八进制数与十进制数的转换。八进制数转换为十进制数时，按式(1-2)计算可得。十进制数转换为八进制数时，可先转换成二进制数，然后再将得到的二进制数转换为等值的八进制数。

(2) 二、八、十六进制间的相互转换

① 二-十六转换。把二进制数转换成等值的十六进制数，称为二-十六进制转换。由于4位二进制数恰好有16个状态，而把这4位二进制数看做一个整体时，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$(11011010.10110110)_2$化为十六进制数时可得

$$(1101,\ 1010.\ 1011,\ 0110)_2$$

$$(D\quad A.\quad B\quad 6)_{16}$$

所以 $$(11011010.10110110)_2=(DA.B6)_{16}。$$

② 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。转换时将十六进制数的每一位用等值的二进制数代替便可得到所求的二进制数。

例如，将$(AC9.F7)_{16}$化为二进制数时得到

$$(A\quad C\quad 9.\quad F\quad 7)_{16}$$

$$(1010\quad 1100\quad 1001.\quad 1111\quad 0111)_2$$

所以 $$(AC9.F7)_{16}=(101011001001.11110111)_2。$$

③ 二-八转换。把二进制数转换成等值的八进制数称为二-八进制转换。转换的方法是，将二进制数从低位到高位每三位分为一组，并以等值的八进制数代之，即得到八进制数。

④ 八-二转换。八-二转换是指把八进制数转换成等值的二进制数。转换时只需将八进制数的每一位用等值的二进制数代替就可以了。

1.1.2.3 码制

数字系统处理的信息中，一类是数值，一类是文字和符号，它们都可以用多位二进制数码来表示。可见，不同的数码不仅可以表示数量的不同大小，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。

例如在举重比赛时，为便于识别运动员，通常给每个运动员编一个号码。显然，这些号码仅仅表示不同的运动员，已失去了数量大小的含义。

为便于记忆和处理，在编制代码时总要遵循一定的规则，这些规则就叫做码制。

在数字电路中常使用二-十进制代码，简称BCD(Binary Coded Decimal)代码。所谓二-十进制码，就是用4位二进制数代码表示1位十进制数。4位二进制数有16种组合，表示十进制数只需要十种组合，因此用4位二进制码来表示十进制数可以有多种不同的选取方式。表1-1中列出了几种常见的BCD码，它们的编码规则各不相同。

8421码是BCD代码中最基本、最常用的一种。它是选用四位二进制码中前十个码字0000～1001分别代表它所对应的十进制数码，余下的6个码字不同。这种BCD码和四位二进制码一样，每位都有固定的权，是一种有权码。各位的权值为8、4、2、1，故称为8421BCD码。

表 1-1　　几种常见的 BCD 代码

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	

若用 8421BCD 码表示多位十进制数时，8421 码与十进制数之间的转换可按表 1-1 中给出的关系直接按位进行代换。如将$(459)_{10}$转换为 8421BCD 码，则只要写出每位十进制数的 8421BCD 码即可。即

$$(4 \quad 5 \quad 9)_{10}$$

$$(0100 \quad 0101 \quad 1001)_{8421BCD}$$

因此　　$(459)_{10}=(010001011001)_{8421BCD}$。

如果要将$(0001011101010000)_{8421BCD}$转换成为十进制数，则只要从最低位开始，将 BCD 码按每四位分组，然后写出每四位码对应的十进制数即可，即

$$(0001 \quad 0111 \quad 0101 \quad 0000)_{8421BCD}$$

$$(1 \quad 7 \quad 5 \quad 0)_{10}$$

所以　　$(0001011101010000)_{8421BCD}=(1750)_{10}$。

再如，将$(100110000111)_{8421BCD}$转换为十进制数。根据 8421BCD 码以 4 位二进制作为十进制数的 1 位，所以$(100110000111)_{8421BCD}=(987)_{10}$。

余 3 码的编码规则与 8421 码不同，如果把每一个余 3 码看做 4 位十进制数，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。实际上它是将 8421BCD 码的每个码组加 0011(即十进制数 3)的一种编码方法。余 3 码中各位二进制数并无固定的权值，是一种变权码。它的编码特点是相邻两个代码之间仅有一位的状态不同，所以，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-冒险现象。

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 码作十进制数加法运算时，若两数之和为 10，正好等于二进制数的 16，当采用四位二进制加法器时，便从高位自动产生进位信号。这是采用四位二进制加法器进行二-十进制代码运算时用余 3 码的优点所在。

余 3 码的特点也是 0 和 9、1 和 8、2 和 7、3 等所对应的码互为反码。因为余 3 码对 9 取补很容易，所以也常用于 BCD 码的运算电路中。

2421 码是一种有权代码，不过它的四个代码从左到右的权是 2、4、2、1，所以称为 2421 码。因为 2421 码的 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 也互为反码，便于对 9 取补，它是一种对 9 的自补代码，所以在一些运算电路中有时采用它比较方便。

5211 码是另一种恒权代码，待学会了第 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，则连续输入计数脉冲的 4 个触发器输出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。

1.1.3 算术运算和逻辑运算

在数字电路中，1 位十进制数码的 0 和 1 不仅可以表示数量的大小，而且可以表示两种不同的逻辑状态。例如，可以用 1 和 0 分别表示一件事情的是和非、真和伪、有和无、好和坏，或者表示电路的通和断、电灯的亮和灭，等等。这种只有两种对立逻辑状态的关系称为二值逻辑。

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，惟一的区别在于，二进制数是逢二进一，而十进制数是逢十进一。

例如，两个二进制数 1001 和 0101 的算术运算有

加法运算	减法运算	乘法运算	除法运算
1001	1001	1001	1.111…
+ 0101	− 0101	× 0101	0101 ⟌ 1001
1110	0100	1001	0101
		0000	1000
		1001	0101
		0000	0111
		0101101	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。以下各位为 0 和 1 表示数值。用这种方式表示的数码称为原码。例如

$$(\overset{\text{符号位}}{0}1011001)_2=(+89)_{10}$$

$$(\overset{\text{符号位}}{1}1011001)_2=(-89)_{10}$$

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。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；

正数的补码与其原码相同；

负数的补码可通过将原码的数值位逐位求反，然后在最低位上加 1 得到。

【例 1-1】 计算$(1001)_2-(0101)_2$。

【解】 根据二进制数的运算规则可知

$$\begin{array}{r} 1001 \\ -\ 0101 \\ \hline 0100 \end{array}$$

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

$$[+1001]_{补}=\underset{符号位}{0}1001$$

$$[-0101]_{补}=\underset{符号位}{1}1011$$

然后将两个补码相加，并舍去进位

$$\begin{array}{r} 0\ 1001 \\ +\quad 1\ 1011 \\ \hline 舍去\leftarrow 1\ 0\ 0100 \end{array}$$

则得到与前面同样的结果。这样，就把减法运算转化为加法运算。

采用补码做减法运算时，当被减数小于减数时，两个补码相加后所得的结果，是以差的补码形式出现的(有符号数)。

此外，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。

当两个二进制数码表示不同的逻辑状态时，它们之间可以按照指定的某种因果关系进行逻辑运算。这种逻辑运算和算术运算有本质上的不同。下面将重点介绍逻辑运算的各种规律。

1.2 逻辑代数中的三种基本运算

逻辑代数是讨论逻辑关系的一门科学，1849 年由英国数学家乔治·布尔(George Boole)创立，通常称为布尔代数。布尔代数早期用于分析开关网络，所以又称为开关代数。随着数字技术的发展，布尔代数成为逻辑设计的数学基础，在数字电路的分析和设计中得到广泛的应用。

逻辑代数和普通代数一样，也是用字母表示变量，但变量的含义和取值完全不同。逻辑变量所表示的是事件发生的前提(条件)存在不存在，逻辑判断的结果是真还是假，在具体线路中则表示电平是高还是低、脉冲的有或无等两种可能的状态。因此，逻辑变量只允许取两个不同的值。为了应用数学方法来表示逻辑关系，则用数字 0 和 1 作为变量的两个值。所以，逻辑变量中的 0 和 1 不能看做数值，它们之间不存在大小关系，只代表研究对象所具有的两种不同的逻辑状态。

逻辑代数有三种最基本的运算，这就是与逻辑运算、或逻辑运算及非逻辑运算。为便于理解它们的含义，先来看一个简单的例子。

图 1-1 中给出了三个指示灯的控制电路。在图 1-1(a)电路中，只有当两个开关同时闭合时，指示灯才会亮；在图 1-1(b)电路中，只要有任何一个开关闭合，指示灯就亮；而在图 1-1(c)电路中，开关断开时灯亮，开关闭合时灯反而不亮。

如果把开关闭合作为条件(或导致事物结果的原因)，把灯亮作为结果，那么图 1-1 中的三个电路代表了三种不同的因果关系：

图 1-1(a)的例子表明，只有决定事物结果(灯亮)的全部条件(开关闭合)同时具备时，结果才发生。这种因果关系叫做逻辑与，或者叫做逻辑乘。

图 1-1(b)的例子表明，在决定事物结果的诸条件中只要有一个满足，结果就会发生。这种因果关系叫做逻辑或，也叫做逻辑加。

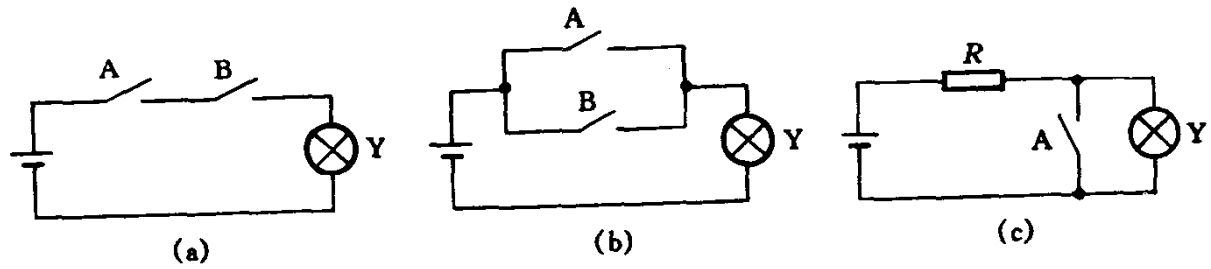

图 1-1　用于说明与、或、非定义的电路

(a) 串联开关电路　(b) 并联开关电路　(c) 单开关电路

图 1-1(c)的例子表明，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，这种因果关系叫做逻辑非，也叫做逻辑求反。也就是说，非逻辑就是逻辑的否定。

若以 A、B 表示开关的状态，并以 1 表示开关闭合，以 0 表示开关断开；以 Y 表示指示灯的状态，并以 1 表示灯亮，以 0 表示不亮，则可以列出以 0、1 表示的与、或、非逻辑关系的图表，见表 1-2，1-3，1-4。这种图表叫做逻辑真值表，或简称为真值表。

表 1-2　与逻辑运算的真值表

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

表 1-3　或逻辑运算的真值表

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

表 1-4　非逻辑运算的真值表

A	Y
0	1
1	0

在逻辑代数中，把与、或、非看做逻辑变量 A、B 间的三种最基本的逻辑运算，并以“·”表示与运算，以“+”表示或运算，以变量上边的“—”表示非运算。因此，对 A 和 B 进行与逻辑运算时可以写成

$$Y = A \cdot B \tag{1-9}$$

为简化书写，允许将 $A \cdot B$ 简写成 AB，略去逻辑相乘的运算符号“·”。A 和 B 的与逻辑关系可简写成 $Y = AB$，省略乘号“·”。

A 和 B 进行或逻辑运算时可写成

$$Y = A + B \tag{1-10}$$

对 A 进行非逻辑运算时可写成

$$Y = \bar{A} \tag{1-11}$$

$\bar{A}$ 读做“A 非”或“A 反”。把 A 叫做原变量，把 $\bar{A}$ 叫做反变量，A 和 $\bar{A}$ 是一个变量的两种形式。

同时，把实现与逻辑运算的单元电路叫做与门，把实现或逻辑运算的单元电路叫做或门，把实现非逻辑运算的单元电路叫做非门(也叫做反相器)。

与、或、非逻辑运算还可用图 1-2 所示的图形符号表示。这些图形符号也用于表示相应的门电路。图中上边一行是目前国家标准局规定的符号，下边一行是常见于国外一些书刊和资料上的符号。

实际的逻辑问题往往比与、或、非复杂得多，不过它们都可以用与、或、非的组合来实现。最常见的复合逻辑运算有与非、或非、与或非、异或、同或等。表 1-5 至表 1-9 给出了这些复合逻辑运算的真值表。图 1-3 是它们的图形逻辑符号和运算符号。

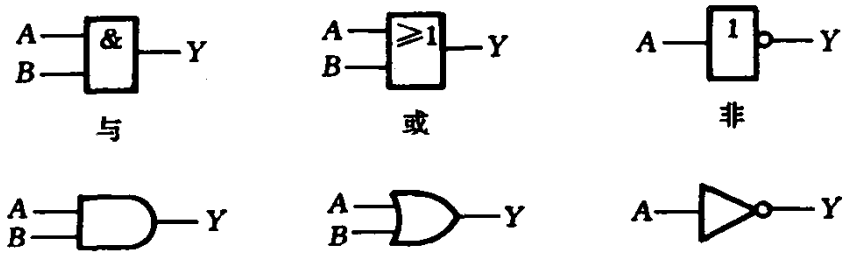

图 1-2　与、或、非的图形符号

表 1-5　　与非逻辑的真值表

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

表 1-6　　或非逻辑的真值表

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

表 1-7　　与或非逻辑的真值表

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

表 1-8　　异或逻辑的真值表

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

表 1-9　　同或逻辑的真值表

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

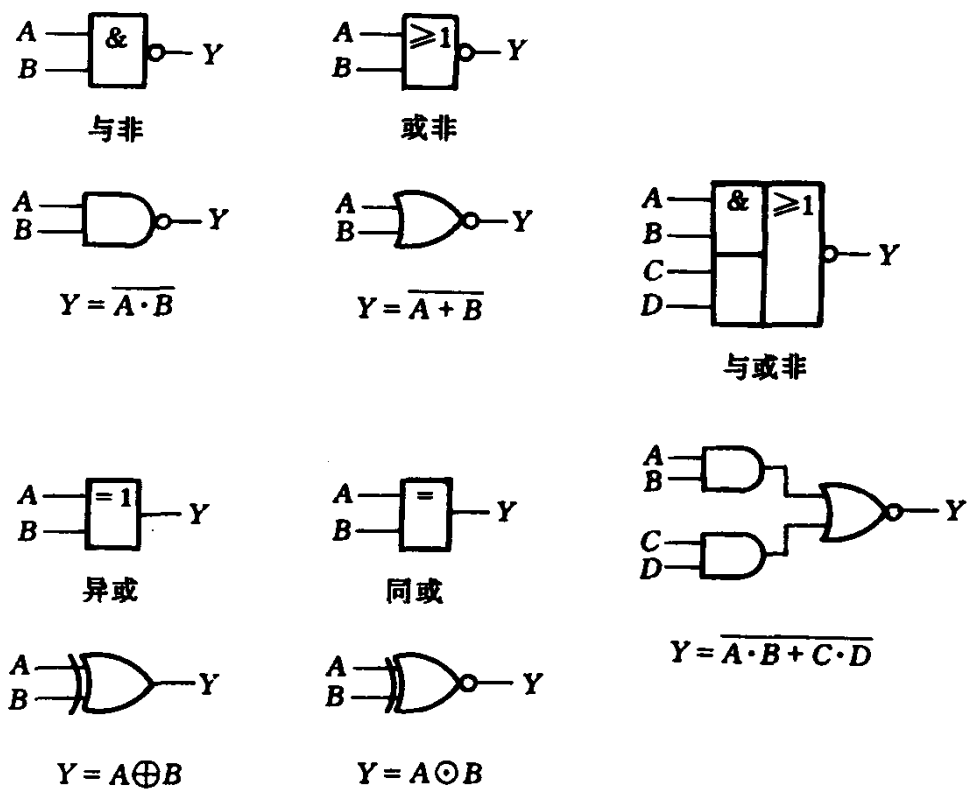

图 1-3　复合逻辑的图形符号和运算符号

由表 1-5 可见，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。与非逻辑关系可表示为

$$Y=\overline{A\cdot B} \tag{1-12}$$

因此，可以把与非运算看做与运算和非运算的组合。图 1-3 中图形符号上的小圆圈表示非运算。

在与或非逻辑中，A、B 之间以及 C、D 之间都是与的关系，只要 A、B 或 C、D 任何一组同时为 1，输出 Y 就是 0。只有当每一组输入都不会是 1 时，输出 Y 才是 1。与或非逻辑关系可表示为

$$Y=\overline{A\cdot B+C\cdot D} \tag{1-13}$$

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

$$A\oplus B=A\cdot\overline{B}+\overline{A}\cdot B \tag{1-14}$$

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

$$A\odot B=A\cdot B+\overline{A}\cdot\overline{B} \tag{1-15}$$

而且，由表 1-8 和 1-9 可见，异或和同或互为反运算，即

$$A\odot B=\overline{A\oplus B};\quad A\oplus B=\overline{A\odot B} \tag{1-16}$$

1.3 逻辑代数的基本公式和常用公式

1.3.1 基本公式

表 1-10 给出了逻辑代数的基本公式，这些公式也叫布尔恒等式。

表 1-10　　逻辑代数的基本公式

序　号	公　式	序　号	公　式
(1)	$0\cdot A=0$	(10)	$\overline{1}=0;\ \overline{0}=1$
(2)	$1\cdot A=A$	(11)	$1+A=1$
(3)	$A\cdot A=A$	(12)	$0+A=A$
(4)	$A\cdot\overline{A}=0$	(13)	$A+A=A$
(5)	$A\cdot B=B\cdot A$	(14)	$A+\overline{A}=1$
(6)	$A\cdot(B\cdot C)=(A\cdot B)\cdot C$	(15)	$A+B=B+A$
(7)	$A\cdot(B+C)=A\cdot B+A\cdot C$	(16)	$A+(B+C)=(A+B)+C$
(8)	$\overline{A\cdot B}=\overline{A}+\overline{B}$	(17)	$A+B\cdot C=(A+B)\cdot(A+C)$
(9)	$\overline{\overline{A}}=A$	(18)	$\overline{A+B}=\overline{A}\cdot\overline{B}$

式(1)、(2)、(11)和(12)给出了变量与常量间的运算规则。

式(3)和(13)是同一变量的运算规律，也叫重叠律。

式(4)和(14)表示变量与它的反变量之间的运算规律，也称为互补律。

式(5)和(15)为交换律，式(6)和(16)为结合律，式(7)和(17)为分配律。

式(8)和(18)是著名的德·摩根(De·Morgan)定理，亦称反演律。在逻辑函数的化简和变换中经常要用到这一对公式。

式(9)表明，一个变量经过两次求反运算之后还原为其本身，所以该式又称为还原律。

式(10)是对 0 和 1 求反运算的规则，它说明 0 和 1 互为求反的结果。

这些公式的正确性可以用列真值表的方式加以验证。如果等式成立，那么将任何一组公式代入两边所得到的结果应该相等。因此，等式两边所对应的真值表也必然相同。

【例 1-2】 用真值表证明表 1-10 中式(17)的正确性。

【解】 已知表 1-10 中的式(17)为

$$A + B \cdot C = (A + B) \cdot (A + C)$$

将 A，B，C 所有可能的取值组合逐一代入上式的两边，算出相应的结果，即得到表 1-11 的真值表。

表 1-11　　式 (17) 的 真 值 表

A	B	C	$B \cdot C$	$A + B \cdot C$	$A + B$	$A + C$	$(A + B) \cdot (A + C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

可见，等式两边对应的真值表相同，故等式成立。

1.3.2 若干常用公式

表 1-12 中列出了几个常用公式。这些公式是利用基本公式导出的。直接运用这些导出公式可以给化简逻辑函数的工作带来很大方便。

表 1-12　　若 干 常 用 公 式

序　　号	公　　式
(19)	$A + A \cdot B = A$
(20)	$A + \bar{A} \cdot B = A + B$
(21)	$A \cdot B + A \cdot \bar{B} = A$
(22)	$A \cdot (A + B) = A$
(23)	$A \cdot B + \bar{A} \cdot C + B \cdot C = A \cdot B + \bar{A} \cdot C$；$A \cdot B + \bar{A} \cdot C + B \cdot C \cdot D = A \cdot B + \bar{A} \cdot C$
(24)	$A \cdot \overline{A \cdot B} = A \cdot \bar{B}$；$\bar{A} \cdot \overline{A \cdot B} = \bar{A}$

现将表 1-12 中的各式证明如下。

式(19)　$A + A \cdot B = A$

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

上式说明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。

式(20)　$A + \bar{A} \cdot B = A + B$

证：根据公式(17)有

$$A + \bar{A} \cdot B = (A + \bar{A}) \cdot (A + B) = 1 \cdot (A + B) = A + B$$

或　　$A + \bar{A} \cdot B = A \cdot (1 + B) + \bar{A} \cdot B = A + A \cdot B + \bar{A} \cdot B = A + (A + \bar{A})B = A + B$

这一结果表明，两个乘积项相加时，如果一项取反后是另一项的因子，则此因子是多余的，可以消去。

式(21)　$A \cdot B + A \cdot \bar{B} = A$