第 1 章　数字电子技术概论

本章介绍数字电子技术的基本概念与数学工具，内容包括：数字信号的基本概念及主要参数、数字集成电路的基本常识、计算机等数字设备中常用的数制与编码、逻辑代数基础、逻辑函数的描述方法、逻辑函数的化简等。这些内容是分析和设计数字电路的基础。

1.1　数字电路的基本概念

随着电子计算机的普及以及通信技术和现代电子技术的快速发展，人类已进入信息时代，在信息社会，数字电子技术得到了广泛的应用和发展，它不仅广泛应用于现代数字通信、雷达、自动控制、测量仪表、医疗设备等各个科技领域，而且进入了人们的日常生活，如智能手机、数码相机、数字电视、影碟机等。可以预料，数字电子技术在人类迈向信息社会的进程中，将起到越来越重要的作用。可以毫不夸张地说，数字电路是计算机和数字通信的重要基石，它们构成了计算机和数字通信设备的硬件基础。

本节将简要介绍数字电路的一些基本概念，以及数字集成电路的发展趋势，让读者在学习数字电子技术之前，首先建立起数字电路的整体概念。

1.1.1　模拟信号与数字信号

1．模拟量与数字量

自然界中存在着各种物理量，其形式千差万别，但就其变化规律而言，可以分为模拟量和数字量两大类。模拟量是指不管在时间上还是在数值上均连续变化的物理量，如温度、压力、速度等；数字量是指不管在时间上还是数值上均不连续的（离散的）物理量，如人口的统计数、产品的个数等。

在实际应用中，许多物理量的测量值既可以用模拟形式表示，也可以用数字形式表示。例如：测量某个电压值，用指针式电压表测量时，结果是模拟量的形式；而用数字式电压表测量时，结果是数字量的形式。

利用现代电子技术，可以很方便地实现模拟量与数字量之间的相互转换。

2．模拟信号与数字信号

在电子设备中，表示模拟量的电信号称为模拟信号（Analog Signal）。例如，正弦波信号就是典型的模拟信号。

表示数字量的电信号称为数字信号（Digital Signal）。例如，矩形波信号就是典型的数字信号。数字信号有时又称为脉冲信号。

数字信号的波形是逻辑电平对时间的图形表示。数字信号有两种传输波形，一种称为电平型，另一种称为脉冲型。电平型数字信号以一个时间节拍内信号是高电平还是低电平来表示“1”或“0”，且每个“1”或“0”信号所占的时间间隔都相等，1 个“1”或 1 个“0”称为 1 位（bit），几个连续的高（低）电平，就是几位“1”（“0”），图 1.1(a)所示为 9bit 数字信号。而脉冲型数字

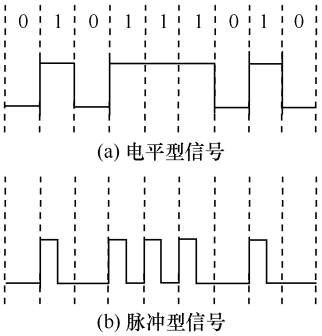


图 1.1　数字信号的传输波形

信号以一个时间节拍内有无脉冲来表示“1”或“0”。如图 1.1(b)所示也为 9bit 数字信号。由图可见，电平型和脉冲型数字信号在波形上有显著差别，即电平型数字信号波形在一个节拍内不会归零，而脉冲型数字信号波形在一个节拍内会归零。

与模拟信号相比，数字信号具有抗干扰能力强、存储处理方便等优点。

3．模拟电路与数字电路

与电路所处理的信号形式相对应，传送、变换、处理、产生模拟信号的电子电路称为模拟电路（Analog Circuit），而传送、变换、处理、产生数字信号的电子电路称为数字电路（Digital Circuit）。

1.1.2　数字信号的主要参数

由图 1.1 可知，数字信号只有两个取值，故称为二值信号，两个取值分别用符号“1”和符号“0”表示，一般用符号“1”表示电路的高电平，而用符号“0”表示电路的低电平。在实际的数字系统中，数字信号波形并没有图 1.1 那么理想。当波形从低电平跳变到高电平，或从高电平跳变到低电平时，边沿没有那么陡峭，而要经历一个过渡过程，分别用上升时间 t_r 和下降时间 t_f 描述。

数字信号的种类很多，在数字系统中，主要应用的是矩形脉冲。下面以电压矩形脉冲为例，来说明数字信号的主要参数，实际的电压矩形脉冲波形如图 1.2 所示。

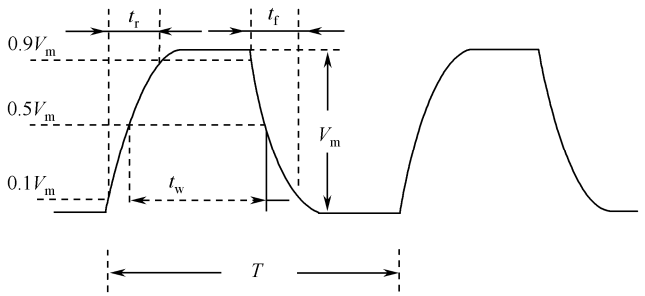


图 1.2　电压矩形脉冲波的主要参数

矩形波数字信号的主要参数有：

⑴ 脉冲幅值 V_m　矩形波电压信号变化的最大值。

⑵ 脉冲上升时间 t_r　脉冲上升沿从 $0.1V_m$ 上升到 $0.9V_m$ 所需要的时间。

⑶ 脉冲下降时间 t_f　脉冲下降沿从 $0.9V_m$ 下降到 $0.1V_m$ 所需要的时间。

⑷ 脉冲宽度 t_w　在脉冲上升沿与脉冲下降沿 $0.5V_m$ 两点间的时间间隔。

⑸ 脉冲周期 T　在周期性脉冲序列中，两个相邻脉冲间的时间间隔为脉冲周期。有时也使用频率 $f=1/T$ 表示单位时间内脉冲重复的次数。

⑹ 占空比 $q=t_w/T$　脉冲宽度与脉冲周期的比值称为占空比。占空比一般用百分数表示，并且将占空比 $q=50\%$ 的矩形波称为方波。

1.1.3　数字技术的发展及其应用

电子技术是 20 世纪发展最迅速、应用最广泛的技术，使得工业、农业、科研、教育、医疗、文化娱乐以及人们的日常生活发生了根本性的变革。特别是数字电子技术，在近 40 多年来，取得了令人瞩目的进步。

电子技术的发展是以电子器件的发展为基础的。20 世纪初直至中叶，主要使用的电子器件是真空管，也称电子管。随着固体微电子学的进步，第一只晶体三极管于 1947 年问世，开创了电子技术的新领域。20 世纪 60 年代初，模拟集成电路和数字集成电路相继问世。到 20 世纪 70 年代末，随着微处理器的问世，电子器件及其应用出现了崭新的局面。1988 年，集成工艺可在 1cm^2 的硅片上集成 3500 万个元件，说明集成电路制造技术已经进入甚大规模阶段。从 20 世纪 80 年代中期开始，专用集成电路（Application Specific Integrated Circuit，ASIC）制作技术日趋成熟，标志着数字集成电路发展到了新的阶段。ASIC 是将一个复杂的数字系统制作在一块半导体芯片上，构成体积小、质量轻、功耗低、速度高、成本低且具有保密性的系统级芯片。ASIC 芯片的实现，可以由用户通过软件编程，将自己设计的数字系统制作在厂家生产的可编程逻辑器件（Programmable Logic Device，PLD）半成品芯片上，从而得到所需的系统级芯片。ASIC 芯片的实现，也可以由芯片生产厂家以全定制的方法批量生产。

当前的集成电路制造技术已使集成电路芯片内部的布线，细微到亚微米和深亚微米（0.13～0.09μm）量级。随着芯片上元件和布线的缩小，芯片的功耗降低而速度提高。最新生产的微处理器的时钟频率高达 3GHz。

数字技术应用的典型代表是电子计算机，电子计算机是随着电子技术的发展而发展的。数字电子技术的发展衍生出计算机的不断发展和完善，计算机技术的影响已遍及人类生产与生活的各个领域，掀起了一场“数字革命”。

数字技术被广泛地应用于广播、电视、通信、医学诊断、测量、控制、文化娱乐以及家庭生活等方面。由于数字信号具有便于存储、处理和传输的特点，使得许多传统使用模拟技术的领域，转而运用数字技术。

1．数字集成电路的发展趋势

当前，数字集成电路正朝着大规模、低功耗、高速度、可编程、可测试和 CMOS 化方向发展。

⑴ 大规模

随着数字电子技术的发展，一块半导体硅片上已经可以集成上百万个数字逻辑门，即使是一个相当复杂的数字系统，也有可能用单片数字集成电路予以实现。可以预见，将来的数字集成电路的集成规模会越来越大，集成规模的提高将极大地提高数字系统的可靠性，减小系统的体积，降低系统的功耗和成本。

⑵ 低功耗

数字系统的功耗很大程度上，取决于所使用的集成电路芯片或模块，人们通常总是希望功耗越低越好。因此，低功耗是数字集成电路的当然选择。现在，即使是包含上百万个逻辑门的超大规模数字集成电路芯片，其功耗也可低达 mW 级。

⑶ 高速度

随着社会的进步，需要处理的信息量越来越大，这就要求所使用的集成电路工作速度越来越高。正是在这样的需求背景下，个人计算机才从当初的 PC 快速发展到今天的奔腾计算机。处理速度为 1.5GHz 的奔腾 4 处理器，于 2000 年由计算机 CPU 芯片巨头美国 Intel 公司向全世界推出。用于核武器模拟试验的运算速度为 12.3 万亿次每秒的超级计算机“白色 ASCI”已由 IBM 公司研制成功，并安装在美国能源部设在加利福尼亚的劳伦斯·利弗莫尔国家实验室。一种旨在探明人体蛋白质结构，运算速度高达 1000 万亿次每秒的名为“蓝色基因”的超级计算机，也早已列入了 IBM 公司的研究计划。虽然计算机的这种高速度在很大程度上依赖于并行处理技术，但集成

电路芯片本身的高速度之作用不容置疑。

⑷ 可编程

早期出现的 MSI/LSI（中规模/大规模）数字集成电路芯片，其功能是由生产厂家根据用户的一般需求而在生产时决定的。大多数情况下，用户使用这种通用型集成电路芯片来实现各种逻辑功能还是非常方便的。但是，当数字系统比较复杂时，所需要的逻辑模块数量往往比较多，这不仅增大了系统的体积和功耗，也降低了系统的可靠性。此外，使用常规模块设计数字系统，也无法防止别人的分析和仿制，设计者的知识产权及合法权益无法得到有效保护。

为了解决上述问题，现在的许多 LSI/VLSI（大规模/超大规模）数字集成电路芯片具有“可编程”的特性。即厂家在生产这些模块时，只生产“半定制”的产品，模块的具体功能由用户根据实际需要进行现场“编程”来决定。这种可编程逻辑器件（PLD），一般具有多次“可编程”甚至“在系统可编程”（In-System Programmable，ISP）的能力，以及“硬件保密”的能力，这不仅为用户研制开发产品带来了极大的方便和灵活性，而且也大大提高了产品的可靠性和保密性。

⑸ 可测试

数字集成电路的规模越来越大，功能也越来越复杂。为了便于数字系统的使用与维护，要求所使用的逻辑模块具有“可测试性”，用户可方便地对其进行“故障诊断”。“可测试”已成为未来数字集成电路的一个重要的发展趋势。

⑹ CMOS 化

数字集成电路芯片所用的器件材料以硅材料为主，在高速电路中，也使用化合物半导体材料，如砷化镓等。晶体管−晶体管逻辑门电路（Transistor-Transistor Logic，TTL）问世较早，其生产工艺经过不断改进，是至今仍在使用的基本逻辑器件之一。但是，随着金属−氧化物−半导体（Metal-Oxide-Semiconductor，MOS）工艺，特别是互补金属−氧化物−半导体（Complementary-Metal-Oxide-Semiconductor，CMOS）工艺的发展，CMOS 集成电路器件具有很高的电路集成度和工作速度，并且功耗很低，因此，TTL 集成电路器件的主导地位已被 CMOS 集成电路器件所取代。

2．数字电路中的操作

在数字电路中，主要有两种操作，即对数字量的算术操作和对逻辑量的逻辑操作。

⑴ 算术操作

数字电路可以对各种数字量进行算术操作，完成加、减、乘、除等基本算术运算。电子计算机之所以称为计算机，就是因为其 CPU 中的运算器，由于采用数字电路而可以对各种数据进行快速的算术运算，使得“计算”成为电子计算机的一个重要特色。

⑵ 逻辑操作

数字电路不仅可以对各种数字量进行算术运算，而且可以对各种逻辑量进行逻辑运算。数字电路具有根据逻辑变量取值进行逻辑推理和逻辑判断的能力。为了突出这一特点，有时也将数字电路称为数字逻辑电路，甚至叫逻辑电路。电子计算机就因为这种逻辑思维能力而被称为“电脑”。

1.1.4 数字集成电路的分类及特点

前面给出了模拟电路和数字电路的概念，实际上，电子电路按功能分为模拟电路和数字电路。根据数字电路的结构特点及其对输入信号的响应规则的不同，数字电路可分为组合逻辑电路和时序逻辑电路。数字电路中的电子器件，如二极管、三极管，工作于开关状态，时而饱和

导通，时而截止，构成电子开关。这些电子开关是组成逻辑门电路的基本器件。逻辑门电路又是数字电路的基本单元，如果将这些门电路及其他元器件集成在一片半导体芯片上，就构成了数字集成电路。

很多情况下，我们将数字集成电路称为芯片、模块、器件。若干数字集成电路芯片按照一定的方案连接在一起，可以构成功能强大的数字电路系统，我们称之为数字系统。

1．数字集成电路的分类

从集成度来看，数字集成电路可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）、超大规模集成电路（VLSI）和甚大规模集成电路（ULSI）五类。所谓集成度，是指每一片芯片所包含的逻辑门电路的个数。表 1.1 所示为数字集成电路的分类情况。

表 1.1　数字集成电路的分类

分　类	门 的 个 数	典型数字集成电路
小规模集成电路	最多 12	逻辑门、触发器
中规模集成电路	12～99	计数器、加法器
大规模集成电路	100～9999	小型存储器、门阵列
超大规模集成电路	10000～99999	大型存储器、微处理器
甚大规模集成电路	10^6 以上	可编程逻辑器件（PLD）、多功能专用集成电路（ASIC）

2．数字集成电路的特点

与模拟电路相比，数字电路具有抗干扰能力强、可靠性高、精确性和稳定性好、功耗低、速度高、通用性广、便于集成、便于故障诊断、便于系统维护等突出优点。以抗干扰能力和可靠性为例，数字电路不仅可以通过整形去除叠加于传输信号上的噪声与干扰，而且可以进一步利用差错控制技术对传输信号进行检错和纠错。

1.2　数制

本节首先介绍常用计数体制，包括十进制、二进制和十六进制，然后介绍数制之间相互转换的方法。在日常生活中，人们习惯于使用十进制，而在数字系统中常采用二进制、八进制和十六进制等。

1.2.1　十进制

数制（Number System）是人类表示数值大小的各种方法的统称。迄今为止，人类都是按照进位方式来实现计数的，这种计数制度称为进位计数制，简称进位制。大家熟悉的十进制，就是一种典型的计数体制。

一种数制中，允许使用的数符的个数，称为这种数制的**基数**（Radix）。例如，十进制（Decimal）中允许使用 0、1、2、3、4、5、6、7、8、9 共 10 个数符，其中最大数符是 9，因此，十进制的基数为 10。一般而论，r 进制的基数就是 r，允许使用的最大数符为 $r-1$。

一种数制中，表示数中不同位置上数字的单位数值，称为**权**（Power）。例如，十进制数 635.78，左边第一位是百位（数字 6 代表 600），权为 10^2；左边第二位是十位（数字 3 代表 30），权为 10^1；左边第三位是个位（数字 5 代表 5），权为 10^0；小数点右边第一位是十分位（数字 7 代表 7/10），

权为10^{-1}；小数点右边第二位是百分位（数字 8 代表 8/100），权为10^{-2}。

十进制是以 10 为基数的计数体制，在日常生活和工作中是最常用的。它有 0、1、2、3、4、5、6、7、8、9 共十个数符，计数规律是“逢十进一”，即在计数的过程中，一旦计数满十，就向高位进一，故称为十进制。

任何一个十进制数，按位置计数法都可表示为

$$(D)_{10}=(a_{n-1}a_{n-2}\cdots a_1a_0a_{-1}\cdots a_{-m})_{10}$$

位置计数法实际上是如下多项式计数法（也称按位权展开式）省略各位权值和运算符号并增加小数点（小数点也称为基点）后的简记形式，即

$$\begin{aligned}(D)_{10}&=a_{n-1}10^{n-1}+a_{n-2}10^{n-2}+\cdots+a_1 10^1+a_0 10^0+a_{-1}10^{-1}+\cdots+a_{-m}10^{-m}\\&=\sum_{i=-m}^{n-1}a_i\cdot 10^i\end{aligned}$$

式中，i 表示数中的第 i 位；a_i 为第 i 位的数符，它可以是 0～9 这 10 个数符中的任何一个；n、m 为正整数，n 表示整数部分的位数；m 表示小数部分的位数；10 表示计数制的基数，D 的下标为 10，表示 D 是一个十进制数；10^i 为第 i 位的权。可见，任何一个十进制数都可以按位权展开，即把每一位的位权值与各自的数符相乘，然后对每一项求和。例如，

$$(2561.347)_{10}=2\times10^3+5\times10^2+6\times10^1+1\times10^0+3\times10^{-1}+4\times10^{-2}+7\times10^{-3}$$

生活中除了十进制外，人们根据计数的不同要求，也采用十二进制、六十进制等。按照以上方法，可以写出任意进制数的按位权展开式：

$$\begin{aligned}(D)_N&=a_{n-1}N^{n-1}+a_{n-2}N^{n-2}+\cdots+a_1N^1+a_0N^0+a_{-1}N^{-1}+\cdots+a_{-m}N^{-m}\\&=\sum_{i=-m}^{n-1}a_i\cdot N^i\end{aligned}$$

式中，N 称为计数的基数，a_i 为第 i 位的数符，N^i 称为第 i 位的权。

1.2.2 二进制

在数字电路和计算机中，机器码（计算机能执行的程序代码）是用二进制（Binary）表示的。二进制是以 2 为基数的计数体制，它只有 0、1 两个数符，计数规律是“逢二进一”，故称为二进制。在二进制数中，每个数位的位权值为 2 的幂。因此，二进制数也可以按位权展开：

$$\begin{aligned}(D)_2&=(a_{n-1}a_{n-2}\cdots a_1a_0a_{-1}\cdots a_{-m})_2\\&=a_{n-1}2^{n-1}+a_{n-2}2^{n-2}+\cdots+a_1 2^1+a_0 2^0+a_{-1}2^{-1}+\cdots+a_{-m}2^{-m}\\&=\sum_{i=-m}^{n-1}a_i\cdot 2^i\end{aligned}$$

式中，a_i 是第 i 位的数符，只能是 0 或 1，n、m 为正整数，2 是二进制的基数，2^i 表示第 i 位的权。

例如，可将二进制数 11010.101 表示为

$$(11010.101)_2=1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$

在数字系统中，采用二进制是比较方便的，因为二进制只有两个数符 0 和 1。因此，二进制数的每一位数字都可以用某些元器件所具有的两个不同的稳定状态来表示，例如三极管的饱和工作状态与截止工作状态，某些电子器件输出端的高电平工作状态和低电平工作状态。只要用其中一种状态表示 1，而用另一种状态表示 0，就可以表示二进制数。但是，用二进制表示一个数时通常位数会很多，书写和阅读很不方便，而且与人们习惯的计数方法不尽相同，因而需要把二进

制数与其他进制数进行相互转换，以达到不同的应用目的。

1.2.3 十六进制

十六进制（Hexadecimal）是以 16 为基数的计数体制。它有 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F 共十六个数符，计数规律是“逢十六进一”，故称为十六进制，各位数的位权值是 16 的幂。十六进制数可以按位权展开为

$$(D)_{16} = a_{n-1}16^{n-1} + a_{n-2}16^{n-2} + \cdots + a_1 16^1 + a_0 16^0 + a_{-1}16^{-1} + \cdots + a_{-m}16^{-m}$$

$$= \sum_{i=-m}^{n-1} a_i \cdot 16^i$$

式中，a_i 为第 i 位的数符，取值范围是 0～9 及 A～F。例如，

$$(D)_{16} = (\text{E5D7.A3})_{16} = 14\times16^3 + 5\times16^2 + 13\times16^1 + 7\times16^0 + 10\times16^{-1} + 3\times16^{-2}$$

需要特别提醒注意的是，在十六进制中，用英文字母 A、B、C、D、E、F 分别表示十进制数的 10、11、12、13、14、15。

十进制、二进制和十六进制的数符、权、运算规则、对应关系，如表 1.2 所示。

表 1.2 常用数制及其对应关系

名　称	十 进 制	二 进 制	十 六 进 制
数符	0、1、2、3、4、5、6、7、8、9	0、1	0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F
第 i 位的权	10^i	2^i	16^i
运算规则	逢十进一，借一为十	逢二进一，借一为二	逢十六进一，借一为十六
对应关系	0	0	0
	1	1	1
	2	10	2
	3	11	3
	4	100	4
	5	101	5
	6	110	6
	7	111	7
	8	1000	8
	9	1001	9
	10	1010	A
	11	1011	B
	12	1100	C
	13	1101	D
	14	1110	E
	15	1111	F
	16	10000	10

1.2.4 数制之间的相互转换

一般来说，人们比较熟悉的是十进制，而电子计算机等数字设备中常使用二进制或十六进制。为了便于人机对话，因而有必要进行各种数制间的转换。

1．各种进制转换为十进制

将二进制、八进制、十六进制及其他进制，转换为十进制的方法是相同的：首先写出待转换的其他进制数的按权展开式，然后求出数符与位权之积，并把各项乘积求和，即可得到转换后的十进制数。例如，

$$(1101.101)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3}$$
$$= 8 + 4 + 1 + 0.5 + 0.125 = (13.625)_{10}$$
$$(172.46)_8 = 1\times8^2 + 7\times8^1 + 2\times8^0 + 4\times 8^{-1} + 6\times 8^{-2}$$
$$= 64 + 56 + 2 + 0.5 + 0.09375 = (122.59375)_{10}$$
$$(4E6.8)_{16} = 4\times16^2 + 14\times16^1 + 6\times16^0 + 8\times16^{-1}$$
$$= 1024 + 224 + 6 + 0.5 = (1254.5)_{10}$$

2．二进制与十六进制的相互转换

从表 1.2 可见，二进制数与十六进制数之间的对应关系非常简单，由于十六进制的基数是 $16 = 2^4$，所以 1 位十六进制数对应于 4 位二进制数。

⑴ 二进制数转换成十六进制数

把二进制数从小数点开始分别向右和向左划分成 4 位一组，每组便是 1 位十六进制数。如果不足 4 位时，则在二进制数整数部分高位添 0，或在小数部分低位添 0 来补足 4 位一组，然后把 4 位二进制数用相应的十六进制数来代替，即可将二进制数转换为十六进制数。例如，

$$(1101111100011.100101)_2 = (\underline{0001}\ \underline{1011}\ \underline{1110}\ \underline{0011}.\underline{1001}\ \underline{0100})_2 = (1BE3.94)_{16}$$

⑵ 十六进制数转换成二进制数

十六进制数转换成二进制数的方法与上述过程相反：将十六进制数的每一个数符用相应的 4 位二进制数替代，并且除去整数部分高位无效的 0 和小数部分末尾无效的 0，即可将十六进制数转换为二进制数。例如，

$$(2BA.5C)_{16} = (\underline{0010}\ \underline{1011}\ \underline{1010}.\underline{0101}\ \underline{1100})_2 = (1010111010.010111)_2$$

3．十进制数转换为二进制数

十进制数转换为二进制数时，十进制数的整数部分和小数部分的转换方法是不同的，需要分别进行。

⑴ 整数部分的转换

十进制整数转换为二进制数，其结果必然也是整数。

将十进制整数转换为二进制数，采用**除 2 取余法**，即十进制整数部分连续除以 2，直至商为 0，所得到的余数就是转换的结果，即所需要的二进制数。需要特别提醒注意的是，最先得到的余数是相应二进制数的最低位（最低位常用符号 LSB 表示），最后得到的余数是相应二进制数的最高位（最高位常用符号 MSB 表示）。例如，十进制整数 54 转换为二进制数的过程如下：

```
        0 ← 1 ← 3 ← 6 ← 13 ← 27 ← 54 | ÷ 2
        ↓   ↓   ↓   ↓   ↓    ↓
（MSB） 1   1   0   1   1    0   （LSB）
```

所以，$(54)_{10} = (110110)_2$。

类似地，十进制整数 153 转换为八进制数的过程如下：

```
        0 ← 2 ← 19 ← 153 | ÷ 8
        ↓   ↓    ↓
（MSB） 2   3    1   （LSB）
```

所以，$(153)_{10} = (231)_8$。

⑵ 小数部分的转换

十进制小数转换为二进制数，其结果必然也是小数。

将十进制小数转换为二进制数，采用**乘 2 取整法**，即十进制小数部分乘以 2，所得乘积的整数部分就是等值二进制小数的最高位（MSB），所得乘积的小数部分再乘以 2，所得乘积的整数部分就是等值二进制小数的次高位，所得乘积的小数部分再乘以 2……如此继续，直到所得乘积的小数部分为 0 或满足精度要求时为止。

需要特别提醒注意的是，最先得到的整数是相应二进制小数的最高位（MSB），最后得到的整数是相应二进制小数的最低位（LSB）。另外，这种转换有时是有误差的。

例如，十进制小数 0.40625 转换为二进制数的过程如下：

$$\begin{array}{l} \times 2 \mid 0.40625 \rightarrow 0.8125 \rightarrow 0.625 \rightarrow 0.25 \rightarrow 0.5 \rightarrow 0 \\ \qquad\qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \\ \text{（MSB）}0 \qquad\quad 1 \qquad\quad 1 \qquad\quad 0 \qquad\quad 1\text{（LSB）} \end{array}$$

所以，$(0.40625)_{10} = (0.01101)_2$。

类似地，十进制小数 0.513 转换为八进制数的过程如下：

$$\begin{array}{l} \times 8 \mid 0.513 \rightarrow 0.104 \rightarrow 0.832 \rightarrow 0.656 \rightarrow 0.248 \rightarrow 0.984 \rightarrow 0.872 \\ \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow \\ \text{（MSB）}4 \qquad\quad 0 \qquad\quad 6 \qquad\quad 5 \qquad\quad 1 \qquad\quad 7\text{（LSB）} \end{array}$$

转换到此已满足精度要求，得$(0.513)_{10} = (0.406517)_8 + e$，剩余误差 $e < 8^{-6}$。

如果待转换的十进制数既有整数部分又有小数部分，则要先将两部分分别转换，再把转换结果并列在一起。例如，

$$(54.40625)_{10} = (110110.01101)_2$$

$$(153.513)_{10} = (231.406517)_8$$

1.2.5 带符号数的表示方法

前面讨论各种进制的数时，没有考虑数的正负问题，下面介绍带符号数的常用表示方法。

1．原码与补码表示法

⑴ 原码表示法

带符号数的原码（Sign Magnitude）表示法是，数值部分用二进制数表示，符号部分用 0 表示“+”，用 1 表示“−”，即采用符号位加绝对值的表示方法，这样形成的一组二进制数称为该带符号数的原码，有时也称为真值。n 位二进制原码所能表示的十进制数的范围为 $-(2^{n-1}-1) \sim +(2^{n-1}-1)$。例如，十进制正数 71 的 8bit 二进制原码为 01000111，十进制负数 −71 的 8bit 二进制原码为 11000111。

⑵ 补码表示法

原码表示法虽然直观，数值的大小与符号可以一目了然，但由于原码的计算规则比较复杂，其电路实现时不太方便。因此，在电子计算机和数字系统中很少采用原码表示法来表示数值。

在电子计算机和数字系统中，通常采用的带符号数表示法是补码（Complement）表示法。补码表示法的规则是，对于正数，补码与原码相同；对于负数，符号位仍为 1，但是二进制数值部分要按位取反，然后加 1。这样得到的一组二进制数称为该带符号数的补码。之所以称其为补

码，是因为该负数的补码，与该负数所对应原码的数值部分，满足互补关系，即二者的和为 2^n，此处 n 为二进制补码的位数。利用这一特点，可以快速计算一个带符号二进制数或十六进制数的补码。n 位二进制补码所能表示的十进制数的范围为 $-(2^{n-1}) \sim +(2^{n-1}-1)$。

例如，十进制正数 71 的 8bit 二进制原码为 01000111，十进制正数 71 的 8bit 二进制补码也是 01000111。

再如，十进制负数 –71 的 8bit 二进制原码为 11000111，十进制负数 –71 的 8bit 二进制补码为 10111001。另外，利用互补特性，求十进制负数 –71 的 8bit 二进制补码过程如下：

$$2^8-71=256-71=185=(10111001)_2$$

顺便指出，由补码求原码的方法，与由原码求补码的方法相同。也就是说，对于正数，原码与补码相同；对于负数，原码的符号位仍为 1，但数值部分要将补码的数值部分按位取反，然后加 1。

此外，当带符号数为纯小数时，其原码或补码的符号位位于小数点的前面，0 表示正数，1 表示负数，并且原来小数点前面的整数 0 不再表示出来。

例如，$(-0.101101)_2$ 的 8bit 二进制原码为$(1.1011010)_2$；$(-0.101101)_2$ 的 8bit 二进制补码为$(1.0100110)_2$。

⑶ 补码的运算

利用补码，可以方便地进行带符号数的加、减运算（减法运算要变换为加法运算）。但要注意的是，同号相加或异号相减时，有可能发生**溢出**。所谓溢出，就是指运算结果超出了原指定二进制数的位数所能表示的带符号数的范围。因此，当发生溢出时，需要增加二进制补码的位数，否则，运算结果将出错。是否溢出，可通过结果的符号位直观地做出判断：正数加正数，或正数减负数，结果均应为正数，否则有溢出；负数加负数，或负数减正数，结果均应为负数，否则有溢出。

例 1.1 试用 4bit 二进制补码计算 5 + 7。

解：因为$(5+7)_{补}=(5)_{补}+(7)_{补}$

$=0101+0111$

$=1100$

计算结果 1100 表示 −4 的补码，而实际正确结果应该为 12。错误产生的原因在于 4bit 二进制补码中，只有 3bit 是表示数值的，能表示的数之范围是 −8 ～+7，而本题的结果 12 需要用 4bit 数值位来表示，因而产生溢出。解决溢出的办法是进行位扩展，即用 5bit 二进制补码表示被加数 5、加数 7、结果 12，如下所示：

$$\begin{array}{rcccc} & 0 & 1 & 0 & 1 \\ + & 0 & 1 & 1 & 1 \\ \hline & [1] & 1 & 0 & 0 \end{array}$$

⑷ 反码表示法

这里，顺便给出反码的概念以及反码表示法。**反码**的符号部分与原码相同，即数的最高位也是符号位，而且用 0 表示正数，用 1 表示负数。反码的数值部分与数的符号有关：对于正数，反码的数值部分与其原码相同；对于负数，反码的数值部分是将原码的数值部分按位取反后所得。

2．定点数与浮点数表示法

如果考虑小数点的位置，那么带符号数还可以采用定点数和浮点数表示法。所谓**定点数**（Fixed-point number），是指在所采用的数据描述格式中，小数点的位置是固定不变的数据。所谓**浮点数**（Floating-point number），是指在所采用的数据描述格式中，小数点的位置是浮动的数据。

⑴ 定点数表示法

定点数表示法非常类似于原码表示法和补码表示法，不同之处仅在于定点数表示法中，小数点隐含于约定规则中，而不再出现在最后得到的定点数中。一般而论，当定点数用于带符号整数时，小数点约定在最低位（LSB）的后面；当定点数用于带符号小数时，小数点约定在最高位（MSB）的后面，也就是在符号位的后面。至于是使用原码还是使用补码，则由使用者自己决定。

⑵ 浮点数表示法

浮点数表示法类似于科学记数法。一般来说，任何一个二进制数 N 总可以表示成如下的浮点数形式：

$$N = 2^E \times M$$

式中，E 表示数 N 的阶码，M 表示数的尾数。尾数 M 一般用小数，它表示数 N 的有效数字；阶码 E 为整数，它指出小数点的实际位置；基数 2 是预先约定的，实际中不表示出来。因此，一个浮点数可以用一组二进制定点数来表示，在电子计算机中表示形式如下：

S_E	E	S_M	M

其中，S_E 为阶码符号，E 为阶码；S_M 为尾码符号，M 为尾码。阶码和尾码都可以用原码、反码、补码表示。

浮点数的运算有专门的算法，一般在计算机课程中会进行介绍。数据在计算机中是以二进制形式表示的，正数用原码表示，而负数用补码表示，更详细的相关内容请参见计算机方面的书籍。

1.3 编码

虽然电子计算机等数字设备采用二进制数据进行处理，但人们输入给它处理的却不仅仅是二进制数据，而是还包括字母、数字甚至控制符号。例如，像在带符号数表示法中，用二进制的 0 表示符号“+”，用二进制的 1 表示符号“−”，以便计算机进行处理一样，这些字母、数字、符号也必须用二进制数来表示。这种用若干位二进制数按一定规则表示给定字母、数字、符号或其他信息的过程称为编码（Encode），而编码的结果称为代码（Code）。反过来，将二进制代码还原成字母、数字、符号等的过程称为解码或译码（Decode）。

若需要编码的信息有 N 项，则需要的二进制数码的位数 n 应满足如下关系：

$$2^n \geqslant N$$

下面介绍几种常用的二进制编码。

1.3.1 二−十进制编码

二−十进制编码就是用 4bit 二进制数码表示 1 位十进制数中 0～9 这十个数符，简称 BCD 码（Binary-Coded-Decimal，二进制编码的十进制数）。

当采用 4bit 二进制数进行编码时，共有 16 种代码，理论上可以从 16 种代码中任选 10 种代码来表示十进制数中的十个数符，多余的 6 种代码称为禁用码，平时不允许使用。表 1.3 所示为几种常用的 BCD 码。

8421 BCD 码是一种最常用的 BCD 码，它由 4bit 自然二进制数 0000～1111 共 16 种代码中的前 10 种组成，即 0000～1001，其余 6 种代码是禁用码。其编码中每一位的权从左到右分别为 8、4、2、1，因此称为 8421 BCD 码，它属于有权码，有时也称为自然 BCD 码。

表 1.3　常用的 BCD 码

十进制数符	有　权　码			无　权　码	
	8421 BCD 码	2421 BCD 码	5421 BCD 码	余 3 码	余 3 循环码
0	0000	0000	0000	0011	0010
1	0001	0001	0001	0100	0110
2	0010	0010	0010	0101	0111
3	0011	0011	0011	0110	0101
4	0100	0100	0100	0111	0100
5	0101	1011	1000	1000	1100
6	0110	1100	1001	1001	1101
7	0111	1101	1010	1010	1111
8	1000	1110	1011	1011	1110
9	1001	1111	1100	1100	1010

2421 BCD 码也是有权码，其编码中每一位的权从左到右分别为 2、4、2、1。它的特点是，将任意一个十进制数 *D* 代码各位取反，所得代码正好是 *D* 对 9 的补码。例如，十进制数 2 的 2421 BCD 代码为 0010，各位取反为 1101，由表 1.3 可知，1101 是十进制数 7 的 2421 BCD 代码，而 2 对 9 的补码是 7。这种特性称为自补性，具有自补性的代码称为自补码。

5421 BCD 码也是有权码，其编码中每一位的权从左到右分别为 5、4、2、1。

余 3 码是自补码，与 2421 BCD 码有类似的自补性。余 3 码是无权码，它的每一位没有一定的权值，但余 3 码可以由 8421 BCD 码加 3（0011）得出。

余 3 循环码也是一种无权码，它的特点是，任意两个相邻代码之间仅有 1 位的取值不同。例如，十进制数符 4、5 的两个代码 0100、1100 仅最高位不同。余 3 循环码也称为修改格雷码，下面将加以详细介绍。

1.3.2　格雷码

多位二进制代码在形成和传输的过程中，由于各位的变化速度不同而可能产生错误。为了减少这种错误，人们采用了可靠性编码的方法。这种方法使代码本身具有一种特征或能力，使得代码在形成中不易出错，或者这种代码出错时容易被发现，甚至能查出出错的位置并予以纠正。

可靠性编码有很多种，常用的可靠性编码有格雷（Gray）码、奇偶校验码等。下面介绍格雷码。

格雷码有多种形式，但它们都有一个共同特点，即从一个代码变为相邻的另一个代码时，只有 1 位发生变化。表 1.4 给出了一种典型的 4bit 格雷码。

表 1.4　典型的 4bit 格雷码

十进制码	二进制码	典型格雷码	十进制码	二进制码	典型格雷码
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

由表 1.4 可知，任意两个相邻的十进制数，它们的格雷码都仅有 1 位不同。例如，从 7→8，二进制码是 0111→1000，4 位均发生变化，而格雷码是 0100→1100，只有 1 位发生变化。这一特点有什么意义呢？在采用普通二进制码做加 1 计数时，例如从 7→8， 4 位均发生变化。如果 4 位变化不是同时进行的（实际上，是不会同时进行的），那么在计数过程中就可能出现短暂的粗大误差。例如，第 1 位先被置为 1，然后其他的位被置为 0，则会出现 0111→1111 的粗大误差。然而，格雷码从编码的形式上杜绝了出现这种错误的可能。

格雷码可以被用做二–十进制编码。表 1.5 给出了十进制数符 0～9 的两种格雷码。其中，修改格雷码又称为余 3 循环码，它具有循环性：十进制数的头尾两个数（0 和 9）的格雷码，也只有 1 位不同；十进制数的 1 和 8 的格雷码也只有 1 位不同；十进制数的 2 和 7 的格雷码也只有 1 位不同；十进制数的 3 和 6 的格雷码也只有 1 位不同；十进制数的 4 和 5 的格雷码也只有 1 位不同，构成循环。因此，格雷码有时也称为循环码。

表 1.5　十进制数的两种格雷码

十进制码	典型格雷码	修改格雷码	十进制码	典型格雷码	修改格雷码
0	0000	0010	5	0111	1100
1	0001	0110	6	0101	1101
2	0011	0111	7	0100	1111
3	0010	0101	8	1100	1110
4	0110	0100	9	1101	1010

1.3.3　ASCII 码

ASCII（American Standard Code for Information Interchange）码是一种字符编码，是美国信息交换标准代码的简称，如表 1.6 所示。它由 7bit 二进制数码构成，共表示 128 个字符，包括英文字母、数字、标点符号、控制字符和一些其他字符。ASCII 码用于计算机和计算机之间、计算机和外围设备之间的文字交互。

表 1.6　ASCII 码字符表

低位 / 高位		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	010	SP	!	“	#	$	%	&	‘	(	)	*	+	,	−	.	/
3	011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

例如，字母“A”的编码是 65，字母“a”的编码是 97，PC 键盘上的空格键的编码是 32，等等。当然，仅仅用 ASCII 码是不能完全表示所有字符的，如汉字、韩文、日文等都无法用 ASCII 码直接表示。ASCII 码字符表中，一些控制字符的含义如下：

NUL　Null 空白　　DC1　Device Control 1 设备控制 1

SOH　Start of Heading 标题开始　　DC2　Device Control 2 设备控制 2

STX	Start of Text 正文开始	DC3	Device Control 3 设备控制 3
ETX	End of Text 正文结束	DC4	Device Control 4 设备控制 4
EOT	End of Transmission 传输结束	NAK	Negative Acknowledge 否认
ENQ	Enquiry 询问	SYN	Synchronous Idle 同步空传
ACK	Acknowledge 确认	ETB	End of Transmission Block 块结束
BEL	Bell 响铃	CAN	Cancel 取消
BS	Backspace 退一格	EM	End of Medium 纸尽
HT	Horizontal Tabulation 水平列表	SUB	Substitute 替换
LF	Line Feed 换行	ESC	Escape 脱离
VT	Vertical Tabulation 垂直列表	FS	File Separator 文件分离符
FF	Form Feed 走纸	GS	Group Separator 字组分离符
CR	Carriage Return 回车	RS	Record Separator 记录分离符
SO	Shift Out 移出	US	Unit Separator 单元分离符
SI	Shift In 移入	SP	Space 空格
DLE	Data Link Escape 数据链路换码	DEL	Delete 删除

1.4 逻辑代数基础

逻辑代数（Logic Algebra）是研究逻辑变量及其相互关系的一门科学。由于它是英国数学家乔治·布尔于 1849 年首先提出来的，所以也称为布尔代数（Boolean Algebra）。后来，美国数学家、信息论的创始人香农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与设计中，成为数字逻辑电路的理论基础。

1.4.1 逻辑变量和逻辑函数

所谓逻辑，是指事物的因果关系所遵循的规律。数字电路也是研究逻辑的，即研究数字电路的输入、输出的因果关系，也就是研究输入和输出间的逻辑关系。为了对输入和输出间的逻辑关系进行数学表达和演算，所以提出了逻辑变量和逻辑函数两个术语。

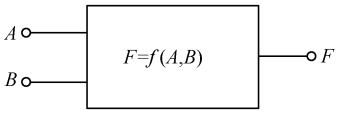


图 1.3 逻辑电路框图

一个逻辑电路的框图如图 1.3 所示，A、B 为输入变量，F 为输出变量，输出和输入之间的逻辑关系可表示为 $F = f(A,B)$。具有逻辑属性的变量称为逻辑变量，其中，A、B 为逻辑自变量，简称**逻辑变量**；F 为逻辑因变量，简称**逻辑函数**。当 A、B 的逻辑取值确定后，F 的逻辑值也随之唯一地被确定下来，表达式 $F = f(A,B)$ 反映了输出变量与输入变量之间的逻辑关系，称为逻辑表达式。

逻辑变量与一般代数变量不同，逻辑变量的取值只有真和假两种取值，为了方便起见，在逻辑代数中分别用 1 和 0 表示逻辑变量的这两种不同取值，“1”表示逻辑真；“0”表示逻辑假。由于数字电路中的两种状态（高电平状态、低电平状态）可以与逻辑变量的这两种取值相对应，所以数字电路有时也称为数字逻辑电路，简称为逻辑电路。值得注意的是，逻辑变量的两种取值 1 和 0 仅代表逻辑变量的两种不同状态，本身既无数值含义，也无大小关系，无论是自变量还是因变量，都只能取 1 和 0 两种值。

例如，在图 1.4 所示电路中，指示灯 F 是否点亮取决于开关 A 是否接通，所以开关与灯之间的因果关系为逻辑关系。开关 A 为输入变量，不妨假设 $A=1$ 表示开关接通，$A=0$ 表示开关断开；灯 F 为逻辑函数，不妨假设 $F=1$ 表示灯亮，$F=0$ 表示灯灭。那么，F 关于 A 的逻辑表达式为 $F=A$。

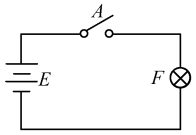


图 1.4　指示灯控制电路

如果逻辑函数是多变量函数，即 $F=f(A,B,C,\cdots)$，那么逻辑函数的表达式就比较复杂，它由逻辑变量 $A,B,C,\cdots$，以及算子“·”（与）、“+”（或）、“−”（非）、括号、等号等组成。例如，

$$F=A;\quad Z=A\cdot B;\quad G=\overline{A};\quad H=A\cdot(B+\overline{C})$$

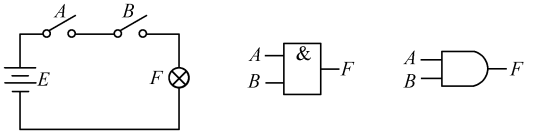
在上述逻辑表达式中，A、B、C 为逻辑变量，F、Z、G、H 为逻辑函数，逻辑变量上加一横杠的为逻辑反变量，不加横杠的为逻辑原变量。

1.4.2　三种基本逻辑运算及逻辑符号

逻辑与、逻辑或、逻辑非是逻辑代数中的三种**基本逻辑运算**。实现这三种逻辑运算的电路，则分别称为与门、或门、非门，它们是基本逻辑门。

1．与运算

与运算也称为逻辑乘、逻辑与。其含义为，只有当决定某一事件的所有条件全部具备时，该事件才会发生，否则该事件不会发生。



(a) 电路图　(b) 国标与门符号　(c) 欧美与门符号

图 1.5　与逻辑电路示意图及与门符号

⑴ 与门符号

与逻辑的概念可以用图 1.5(a)所示的指示灯控制电路来说明。灯 F 亮作为事件发生，开关 A、B 的闭合作为事件发生的条件。从图 1.5(a)可以看出，只有开关 A、B 同时闭合（$A=1$，$B=1$），灯 F 才会亮（$F=1$），满足与逻辑关系。在数字电路中，常把能够实现与运算功能的基本单元称为“与门”，其逻辑符号如图 1.5(b)和图 1.5(c)所示。图 1.5(b)是国标与门符号，方框中的“&”为与运算的定性符。图 1.5(c)是欧美国家使用的与门符号。

⑵ 与运算的逻辑表达式和真值表

与运算的运算符为小圆点“·”，为了简便，有时也将小圆点省略。上述两个逻辑变量 A、B 的与逻辑函数 F，其表达式为

$$F=A\cdot B=AB \qquad (1.1)$$

为了清楚地看出与运算的逻辑功能，常将逻辑自变量 A、B 的各种可能取值及其对应的逻辑函数 F 的值列在一张表上，这张表通常叫真值表。与运算的真值表如表 1.7 所示。由真值表可以得到与运算的运算规则为

$$0\cdot 0=0 \qquad 0\cdot 1=0 \qquad 1\cdot 0=0 \qquad 1\cdot 1=1$$

与运算可以推广到多变量的情形，即 $F=A\cdot B\cdot C\cdots$。

表 1.7　与逻辑真值表

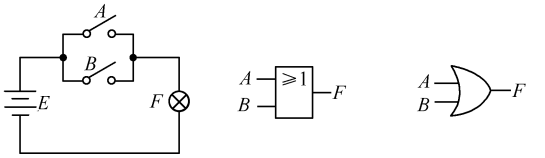
A	B	$F=AB$
0	0	0
0	1	0
1	0	0
1	1	1

2．或运算

或运算也称为逻辑加、逻辑或。其含义为，在决定某一事件的所有条件中，只要有一个条件或一个以上条件具备时，该事件就会发生。

⑴ 或门符号

或逻辑的概念可以用图 1.6(a)所示的指示灯控制电路来说明。灯 F 亮作为事件发生，开关 A、B 的闭合作为事件发生的条件。从图 1.6(a)可以看出，只要开关 A、B 中任意一个闭合，或者开关 A、B 同时闭合（$A=1$，$B=1$），灯 F 就会亮（$F=1$），满足或逻辑关系。在数字电路中，常把能够实现或运算功能的基本单元称为“或门”，其逻辑符号如图 1.6(b)和图 1.6(c)所示。图 1.6(b)是国标或门符号，方框中的“≥”为或运算的定性符。图 1.6(c)是欧美国家使用的或门符号。



(a) 电路图　(b) 国标或门符号　(c) 欧美或门符号

图 1.6　或逻辑电路示意图及或门符号

⑵ 或运算的逻辑表达式和真值表

或运算的运算符为“+”。上述两个逻辑变量 A、B 的或逻辑函数 F，其表达式为

$$F=A+B \tag{1.2}$$

或运算的真值表如表 1.8 所示。由真值表可以得到或运算的运算规则为

$$0+0=0 \qquad 0+1=1 \qquad 1+0=1 \qquad 1+1=1$$

或运算也可以推广到多变量：$F=A+B+C+\cdots$。

表 1.8　或逻辑真值表

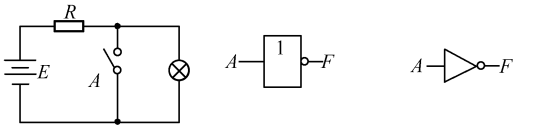
A	B	$F=A+B$
0	0	0
0	1	1
1	0	1
1	1	1

3．非运算

非运算也称为逻辑非。其含义为，条件具备时，该事件不会发生；条件不具备时，该事件会发生。

⑴ 非门符号

非逻辑的概念可以用图 1.7(a)所示的指示灯控制电路来说明。灯 F 亮作为事件发生，开关 A 的闭合作为事件发生的条件。从图 1.7(a)可以看出，只要开关 A 闭合（$A=1$），灯 F 就不会亮（$F=0$）；开关 A 断开（$A=0$）时，灯 F 会亮（$F=1$），满足非逻辑关系。在数字电路中，常把能够实现非运算功能的基本单元称为“非门”，其逻辑符号如图 1.7(b)和图 1.7(c)所示。图 1.7(b)是国标非门符号，方框中的“1”为“缓冲”定性符。图 1.7(c)是欧美国家使用的非门符号。非门有时被称为反相缓冲器。



(a) 电路图　(b) 国标非门符号　(c) 欧美非门符号

图 1.7　非逻辑电路示意图及非门符号

⑵ 非运算的逻辑表达式和真值表

逻辑变量 A 的非逻辑函数 F，其表达式为

$$F=\overline{A} \tag{1.3}$$

非运算的真值表如表 1.9 所示。由真值表可以得到非运算的运算规则为

$$\overline{0}=1 \qquad \overline{1}=0$$

表 1.9　非逻辑真值表

A	$F=\overline{A}$
0	1
1	0

4．复合逻辑运算

复合逻辑运算是由与、或、非三种基本逻辑运算组合而成的，经常用到的有与非、或非、与或非、异或、同或等复合逻辑运算，逻辑符号如图 1.8 所示，上面一行是国标符号，下面一行是欧美国家使用的符号。

在复合逻辑运算中，要特别注意运算的优先顺序，复合逻辑运算的优先顺序为：圆括号→非运算→与运算→或运算。

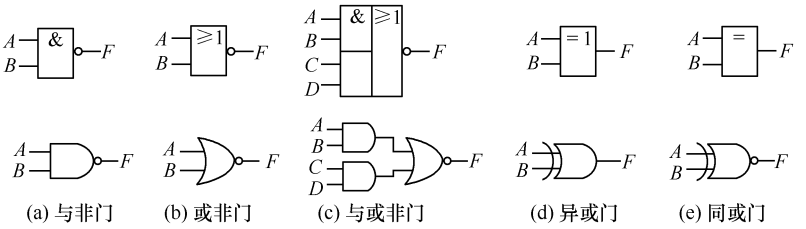


图 1.8　复合逻辑门符号

⑴ 与非逻辑运算

与非逻辑运算是与运算和非运算的组合，先将输入变量 A、B 进行与运算，然后将结果求反，最后得到 A、B 的与非运算结果。其逻辑表达式为

$$F = \overline{A \cdot B} \tag{1.4}$$

与非逻辑运算的真值表如表 1.10 所示。由真值表可见，对于与非运算，输入变量中只要有 0，输出就为 1。或者说，只有输入变量全部为 1 时，输出才为 0。其逻辑符号如图 1.8(a)所示，图上小圆圈表示非运算。

表 1.10　与非逻辑真值表

A	B	$F=\overline{A\cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

⑵ 或非逻辑运算

或非逻辑运算是或运算和非运算的组合，先将输入变量 A、B 进行或运算，然后将结果求反，最后得到 A、B 的或非运算结果。其逻辑表达式为

$$F = \overline{A + B} \tag{1.5}$$

或非逻辑运算的真值表如表 1.11 所示。由真值表可见，对于或非运算，输入变量中只要有 1，输出就为 0。或者说，只有输入变量全部为 0 时，输出才为 1。其逻辑符号如图 1.8(b)所示，图上小圆圈表示非运算。

表 1.11　或非逻辑真值表

A	B	$F=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

⑶ 与或非逻辑运算

与或非逻辑运算是与运算、或运算、非运算的组合，先将输入变量 A、B 和 C、D 分别进行与运算，然后将结果进行或运算，再将结果求反，得到与或非的运算结果，其逻辑表达式为

$$F = \overline{A \cdot B + C \cdot D} \tag{1.6}$$

由上式可见，只要输入变量 A、B 或 C、D 中任何一组的变量同时为 1，输出就为 0；只有当每一组输入变量不全是 1 时，输出才为 1。其逻辑符号如图 1.8(c)所示。

⑷ 异或逻辑运算

异或运算的逻辑关系为，当两个输入变量 A、B 值不相同时，输出为 1；而当两个输入变量相同时，输出为 0，即输入有异，输出为 1。异或也可以用与、或、非的组合表示，逻辑表达式为

$$F = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B} \tag{1.7}$$

式中，“⊕”为异或逻辑运算的运算符，异或运算的真值表如表 1.12 所示，其逻辑符号如图 1.8(d)所示。

表 1.12　异或逻辑真值表

A	B	$F=A\oplus B=\overline{A}\cdot B+A\cdot\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

⑸ 同或逻辑运算

同或运算的逻辑关系为，当两个输入变量 A、B 值不相同时，输出为 0；而当两个输入变量相同时，输出为 1，即输入相同，输出为 1。同或也可以用与、或、非的组合表示，逻辑表达式为

$$F = A \odot B = A \cdot B + \overline{A}\,\overline{B} \qquad (1.8)$$

式中，“⊙”为同或逻辑运算的运算符，同或运算的真值表如表 1.13 所示，其逻辑符号如图 1.8(e)所示。

表 1.13　同或逻辑真值表

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

由表 1.12 和表 1.13 可见，异或运算与同或运算互为反运算，即有

$$A \oplus B = \overline{A \odot B} \qquad A \odot B = \overline{A \oplus B}$$

1.4.3　逻辑函数的描述方法

一般来说，一个比较复杂的逻辑电路，往往是受多种因素控制的，就是说有多个逻辑变量。输出变量与输入变量之间逻辑函数的描述方法并不是唯一的，常用的逻辑函数描述方法有逻辑表达式、真值表、逻辑图、时序图和卡诺图等。这里先介绍前面 4 种方法，逻辑函数的卡诺图描述方法稍后再做介绍。

1．逻辑表达式描述法

用与、或、非三种逻辑运算符，以及括号构成的表示逻辑函数与逻辑变量之间关系的代数式，称为逻辑函数表达式。例如，异或函数的逻辑表达式 $F = A\overline{B} + \overline{A}B$，它描述函数 F 与变量 A、B 的关系是：当 A、B 变量取值相异时，函数值为“1”；否则，函数值为“0”。

2．真值表描述法

真值表是将输入逻辑变量的各种可能取值和相应的函数值排列在一起而组成的表格。在真值表左边一栏列出全部逻辑变量的可能取值组合，然后将每组变量取值的函数值对应地填入表格右边的一栏内，所得表格称为真值表。

由于一个逻辑变量只有 0 和 1 两种可能的取值，则 n 个逻辑变量一共就有 2^n 种可能的取值组合。例如，逻辑表达式 $F = AB + \overline{A}C$ 的真值表如表 1.14 所示。由于该函数有 3 个输入变量，所以共有 $2^3 = 8$ 种输入取值组合。在列真值表时，输入变量的取值组合一般按照二进制数递增的顺序排列，这样做既不易遗漏，也不会重复。

表 1.14　函数 $F = AB + \overline{A}C$ 的真值表

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

真值表的特点，一是直观明了，输入变量取值一旦确定，即可在真值表中查出相应的函数值；二是把一个实际的逻辑问题抽象成一个逻辑函数时，使用真值表是最方便的。所以，在设计逻辑电路时，总是先根据设计要求列出真值表。但当变量比较多时，真值表比较大，显得过于烦琐。

3．逻辑图描述法

将逻辑函数中各变量之间的与、或、非等逻辑关系用相应的逻辑门的电路符号表示出来的图