普通高等教育“十二五”规划教材

数字电子技术基础

SHUZI DIANZI JISHU JICHU

汤秀芬　李　虹　主编

北京邮电大学出版社
www.buptpress.com

数字电子技术基础

汤秀芬　李　虹　主编

北京邮电大学出版社
www.buptpress.com

内容简介

本书是依据教育部教学指导委员会电子信息科学与电气信息类平台课程教学基本要求编写。全书共9章，主要内容包括数制和码制，逻辑代数，逻辑门电路，组合逻辑电路，锁存器和触发器，时序逻辑电路，半导体存储器可编程逻辑器件，脉冲波形的产生和整形，数/模和模/数转换器等，各章末都附有小结和习题，以利于学生理论联系实际，巩固所学知识。

本书可作为高等院校电子、电气、计算机等信息类本科专业“数字电子技术基础”课程的教材，也可供其他各相关专业的学生和从事电子技术工作的工程技术人员参考。

图书在版编目（CIP）数据

数字电子技术基础 / 汤秀芬，李虹主编. --北京：北京邮电大学出版社，2014.12
ISBN 978-7-5635-4211-6

Ⅰ. ①数… Ⅱ. ①汤… ②李… Ⅲ. ①数字电路－电子技术 Ⅳ. ①TN79

中国版本图书馆 CIP 数据核字（2014）第 277485 号

书　　名：数字电子技术基础
著作责任者：汤秀芬　李　虹　主编
责 任 编 辑：付兆华
出 版 发 行：北京邮电大学出版社
社　　址：北京市海淀区西土城路 10 号（邮编：100876）
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：北京源海印刷有限责任公司
开　　本：787 mm×1 092 mm　1/16
印　　张：17.75
字　　数：460 千字
版　　次：2014 年 12 月第 1 版　2014 年 12 月第 1 次印刷

ISBN 978-7-5635-4211-6　　定　价：38.00 元

前　言

本书是依据教育部教学指导委员会电子信息科学与电气信息类平台课程教学基本要求编写，作者根据多年的教学经验积累和课程建设成果，参考国内外诸多经典教材，编写了本书。该教材具体特点如下：

① 注重基础性。全书对基本概念、基本原理和基本分析方法的阐述清晰透彻、深入浅出。内容编排遵循由简单到复杂、循序渐进的原则，遵循"先器件后电路、先基础后应用"的规律。教材层次分明、重点突出，使学生能够对基础知识牢固掌握并能灵活运用。

② 注重实用技能的培养，引入简单易学的虚拟仿真软件 Multisim。Multisim 是 EWB 升级版，以界面形象直观、操作方便、易学易用、仿真分析功能强大等优点突出。Multisim 与 EWB 相比，器件库大大扩充，包含有上万种器件，器件模型设计得更加精确，虚拟仪器品种增多，使用数量也不受限制，仿真分析项目增多，分析结果更加精确、可靠。

③ 加强理论联系实际。本书的主要目的之一就是使读者能设计和构建真正的数字电路，因而我们更注重分析在硬件设计中有些实际意义的问题，如数字集成电路的速度、功耗、驱动能力、噪声容限等对数字电路性能的影响、不同系列门电路之间的连接等。常用集成数字器件采用当前主流芯片，力求体现最新的数字电子技术。

本书共 9 章，各章主要内容及特点如下：

第 1 章介绍数字电路的基本概念、数制、二进制算术运算及码制。

第 2 章介绍逻辑代数、逻辑函数的表示方法及其化简、用 Multisim 11.0 进行逻辑函数的化简与变换。目前一些比较流行的 EDA 软件都具有自动化简和变换逻辑函数式的功能。本章提到的 Multisim 11.0 就是其中的一种。利用这些软件，可以很容易地在计算机上完成逻辑函数的化简或变换。

第 3 章介绍 CMOS、TTL 等几种常见集成门电路的结构、逻辑功能、电气特性和主要参数。为了适应数字集成电路的发展现状，本章在内容选取上采用了以 CMOS 电路为主，TTL 为辅的方案，强调 CMOS 基本单元电路的研究。增加双极型 CMOS 门电路的有关内容。面向工程应用，介绍了门电路的电气特性、门电路 3 种不同输出结构（推拉、OD、三态）使用方法、门电路的接口。本章内容对读者建立数字硬件电路的基础非常重要。

第 4 章介绍组合逻辑电路的分析与设计的方法、常用的组合逻辑电路。通过一个例子简单介绍如何使用 Multisim 11.0 分析组合逻辑电路。其目的是使读者更好地理解电子设计自动化。由于 Multisim 11.0 的操作主要涉及原理图的输入和仿真，参照本章提供操作步骤，读者可以在很短的时间内掌握该软件的基本操作。

第 5 章介绍锁存器和触发器。锁存器和触发器虽然同为记忆单元，但具有不同的动作特点，本章对二者进行了严格区分。

第 6 章介绍时序逻辑电路的分析与设计的方法、常用的时序逻辑电路。通过一个例子简单介绍如何使用 Multisim 11.0 分析时序逻辑电路。

第 7 章介绍半导体存储器和可编程逻辑器件。针对各种半导体存储器具有的相似的内部结构和工作原理的特点，重点对各种存储单元的差异和特点进行分析，使读者能较快地领会各种半导体存储器的特点和应用场合。可编程逻辑器件发展很快，但基本结构和工作原理并没有太大变化，因此，本章重点分析简单可编程逻辑器件的结构和工作原理，在简单可编程逻辑器件的基础上引入目前的主流器件 CPLD/FPGA。

第 8 章介绍施密特触发器、单稳态触发器、多谐振荡器 3 种电路的工作原理、分析方法、主要参数计算，并给出了这 3 种电路的许多应用举例。通过一个例子介绍如何使用 Multisim 11.0 分析 555 定时器。

第 9 章介绍数/模和模/数转换器的基本原理和主要技术指标。数/模转换器和模/数转换器种类较多，本章选取在集成芯片中常用的电路类型。

参加本书编写工作的有汤秀芬(第 1～6 章)、李虹(第 7～9 章)。汤秀芬任本书主编，负责全书的整体规划和统稿工作。

本书编写力求条理清晰，语言准确，文字简洁，图表规范。由于我们的能力和水平有限，书中不妥之处，恳请读者给予批评指正。

作者

目　　录

第1章 数制和码制

1.1 概　　述

人们在自然界感知的形形色色物理量中，就其变化规律的特点而言，可以分为两大类。一类物理量是一系列离散的时刻取值，数值的大小和每次的增减都是最小数量单位的整数倍，而小于这个最小数量单位的数值没有任何物理意义，即它们是一系列时间离散、数值也离散的信号。我们把这一类物理量称为数字量，表示数字量的信号称为数字信号。将工作在数字信号下的电子电路称为数字电路。例如，我们统计通过某一条生产流水线的零件数量，得到的就是一个数字量，最小数量单位的“1”代表一个零件，小于1的数值已经没有任何物理意义。

另外一类物理量的变化在时间上连续，数值上也是连续的。我们把这一类物理量称为模拟量，表示模拟量的信号称为模拟信号，将工作在模拟信号下的电子电路称为模拟电路。在工程技术上，为了便于处理和分析，通常用传感器将模拟量转换为与之成比例的电压或电流信号，然后再送到电子系统中进一步处理。例如，热电偶工作时，输出的电压或电流信号就是一种模拟信号，因为被测的温度不可能发生突跳，所以测出的电压或电流，无论在时间上还是在数值上都是连续的。

随着计算机科学与技术突飞猛进的发展，绝大多数电子系统都采用计算机来对信号进行处理。由于计算机无法直接处理模拟信号，所以需要将模拟信号转换为数字信号，然后送到数字电路(可以是专用的数字信号处理电路，也可以是通用的计算机)进行处理，最后再将处理结果根据需要转换为相应的模拟信号输出。

数字信号通常都是用数码形式给出的。不同的数码可以用来表示数量的大小。用数码表示数量大小时，仅用一位数码往往不够用，因此，经常需要用进位计数制的方法组成多位数码使用。这种多位数码的构成方式以及从低位到高位的进位规则称为数制。人们在日常生活中经常遇到计数问题，并且习惯于用十进制数。在数字电路中，经常使用的计数进制除了我们最熟悉的十进制以外，更多的是使用二进制、八进制和十六进制。

当两个数码分别表示两个数量大小时，两个二进制数可以进行算术运算。由于目前数字电路中的算术运算最终都是以二进制运算进行的，所以在这一章里，我们还将比较详细地讨论在数字电路中是采取什么方式完成二进制算术运算的。

不同的数码不仅可以用来表示数量的大小，而且可以用来表示不同的事物或事物的不同状态。这些特定的二进制数码称为代码。为了便于记忆和查找，以一定的规则编制代码，用以表示十进制数值、字母、符号等的过程称为编码；若需编码的信息有 N 项，则需用的二进制数

码的位数 n 应满足如下关系

$$2^n \geqslant N \tag{1.1.1}$$

将代码还原成所表示的十进制数、字母、符号等的过程称为解码或译码。

1.2 数 制

1.2.1 十进制

十进制(Decimal Number)是日常生活和工作中最常使用的进位计数制。十进制数采用0、1、2、3、4、5、6、7、8、9 十个不同的数码。在计数时,采用"逢十进一",例如 9+1=10,19+1=20。所谓十进制就是以 10 为基数的计数体制。每一个数码处于十进制数的不同数位时,它代表的数值是不同的,这些数值称为位权(Weight)。对于任意一个十进制数都可以按位权展开。例如,把十进制数"356.52"按权展开为

$$(356.52)_D = 3\times10^2 + 5\times10^1 + 6\times10^0 + 5\times10^{-1} + 2\times10^{-2}$$

一般的,任意十进制数可表示为

$$(N)_D = \sum_{i=-\infty}^{\infty} K_i \times 10^i \tag{1.2.1}$$

式(1.2.1)中 K_i 为基数"10"的第 i 次幂的系数,它可以是 0~9 中任何一个数字。

如果将式(1.2.1)中的基数"10"用字母 R 来代替,就可以得到任意进制数的表达式。

$$(N)_R = \sum_{i=-\infty}^{\infty} K_i \times R^i \tag{1.2.2}$$

式(1.2.2)中的取值与式(1.2.1)规定相同。R 称为计数的基数,K_i 是第 i 次幂的系数,R^i 称为第 i 位的权。

十进制数人们最熟悉,但数字电路实现起来困难。因为构成数字电路的基本思路是把电路的状态与数码对应起来。而十进制的 10 个数码要求电路有 10 个完全不同的状态,这样使得电路很复杂,因此,在数字电路中不直接处理十进制数。

1.2.2 二进制

目前,在数字电路中应用最广泛的是二进制(Binary Number)。二进制数只有"0"和"1"两个数码,在计数时"逢二进一",例如 1+1=10(读为"壹零")。必须注意,这里的"10"与十进制数的"10"是完全不同的,它并不代表数"拾"。左边的"1"表示 2^1 位数,右边的"0"表示 2^0 位数,也就是 $10=1\times2^1+0\times2^0$。所谓二进制就是以"2"为基数的计数体制。每个数位的权值为 2 的幂。

一般地说,二进制数可表示为

$$(N)_B = \sum_{i=-\infty}^{\infty} K_i \times 2^i \tag{1.2.3}$$

根据式(1.2.3),任何一个二进制数均可展开,并计算出它所表示的十进制数的大小。例如,把二进制数"1111.001"按权展开为

$$(1111.001)_B = 1\times2^3+1\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}=(15.125)_D$$

二进制数由于只需"0"和"1"两个状态，数字装置实现简单可靠，所用元件少。因而二进制是数字系统唯一认识的代码。但二进制书写太长。

公式中分别使用下脚注 B(Binary)和 D(Decimal)表示括号里的数是二进制数和十进制数。

1.2.3　十六进制和八进制

对于同一个数，用二进制数表示比用十进制数表示需要的位数多，书写和阅读都不方便，容易出错。为此，人们常采用二进制数的缩写形式：十六进制数和八进制数。采用十六进制数和八进制数比二进制数简短，易读易记，且与二进制之间的转换方便。因此，数字系统中普遍采用十六进制数和八进制数表示。

在十六进制(Hexadecimal Number)中，每个数位上使用的数码符号为 0、1、2、3、4、5、6、7、8、9、A(10)、B(11)、C(12)、D(13)、E(14)、F(15)共 16 个，其计数规则是"逢十六进一"，所谓十六进制就是以"16"为基数的计数体制。每个位的权值为 16 的幂。其表达式如下

$$(N)_H = \sum_{i=-\infty}^{\infty} K_i \times 16^i \tag{1.2.4}$$

在八进制(Octal Number)中，有 0、1、2、3、4、5、6、7 八个数码，其计数规则是"逢八进一"，所谓八进制就是以"8"为基数的计数体制。每位的权值为以 8 为底的幂。其表达式如下

$$(N)_O = \sum_{i=-\infty}^{\infty} K_i \times 8^i \tag{1.2.5}$$

根据式(1.2.4)和(1.2.5)，任何一个十六进制数和八进制数均可展开，并计算出它们所表示的十进制数的大小。例如，把八进制数"13.2"和十六进制数"F9"分别按权展开为

$$(13.2)_O = 1\times 8^1 + 3\times 8^0 + 2\times 8^{-1} = (11.25)_D$$

$$(F9)_H = 15\times 16^1 + 9\times 16^0 = (249)_D$$

公式中分别使用下脚注 H(Hexadecimal)和 O(Octal)表示括号里的数是十六进制和八进制。

1.2.4　不同进制数之间的转换

1. 其他进制转换成十进制

不同数制之间的转换方法有若干种。把非十进制数转换成十进制数，采用按权展开相加法。具体步骤是，首先把非十进制数写成按权展开的多项式，然后按十进制数的计数规则求其和。

例 1.2.1　试将二进制数$(1101.11)_B$转换为十进制数。

解：$(1101.11)_B = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2} = (13.75)_D$

例 1.2.2　将八进制数$(167.42)_O$转换为十进制数。

解：$(167.42)_O = 1\times 8^2 + 6\times 8^1 + 7\times 8^0 + 4\times 8^{-1} + 2\times 8^{-2} = (119.53125)_D$

例 1.2.3　将十六进制数$(4E6)_H$转换为十进制数。

解：$(4E6)_H = 4\times 16^2 + 14\times 16^1 + 6\times 16^0 = (1254)_D$

2. 十-二进制之间的转换

十进制数转换为二进制数时，整数部分和小数部分的方法不同，下面分别介绍。

对于整数部分，十进制整数$(N)_D$ 可写成

$$(N)_D = b_n \times 2^n + b_{n-1} \times 2^{n-1} + \cdots + b_1 \times 2^1 + b_0 \times 2^0 \tag{1.2.6}$$

$$\frac{1}{2}(N)_D = b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_1 \times 2^0 + \frac{b_0}{2} \tag{1.2.7}$$

由此可知，将十进制数除以 2，其余数为 b_0，得到的商为

$$b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_1 \tag{1.2.8}$$

同理，可将式(1.2.8)除以 2 得到的商写成

$$b_n \times 2^{n-1} + b_{n-1} \times 2^{n-2} + \cdots + b_1 = 2\left(b_n \times 2^{n-2} + b_{n-1} \times 2^{n-3} + \cdots + b_2 + \frac{b_1}{2}\right) \tag{1.2.9}$$

由式(1.2.9)不难看出，若将$(N)_D$除以 2 所得的商再除以 2，则所得余数即 b_1。依此类推，反复将每次得到的商再除以 2，直到商为 0 为止，就可求得二进制数的每一位了。

例 1.2.4 将十进制数$(25)_D$转换为二进制数。

解：根据上述原理，可按下列步骤将其转换为二进制数。

$$\begin{array}{r|l l l}
2 & 25 & \cdots\cdots 余1 \cdots\cdots K_0 & 低 \\
2 & 12 & \cdots\cdots 余0 \cdots\cdots K_1 & \downarrow \\
2 & 6 & \cdots\cdots 余0 \cdots\cdots K_2 & \downarrow \\
2 & 3 & \cdots\cdots 余1 \cdots\cdots K_3 & \downarrow \\
2 & 1 & \cdots\cdots 余1 \cdots\cdots K_4 & 高 \\
 & 0 & &
\end{array}$$

上述转换步骤可归纳为：十进制数 25 除 2 取余，直至商为 0，并将余数自上而下倒级联，即得到相应的二进制数。因此，$(25)_D = (11001)_B$。

对于小数部分，十进制整数$(N)_D$ 可写成

$$(N)_D = b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \cdots + b_{-(n-1)} \times 2^{-(n-1)} + b_{-n} \times 2^{-n} \tag{1.2.10}$$

将式(1.2.10)两边分别乘以 2，得

$$2 \times (N)_D = b_{-1} \times 2^0 + b_{-2} \times 2^{-1} + \cdots + b_{-(n-1)} \times 2^{-(n-2)} + b_{-n} \times 2^{-(n-1)} \tag{1.2.11}$$

式(1.2.11)说明，将十进制小数乘以 2，所得乘积的整数即为 b_{-1}。因此，将小数用基数 2 去乘，保留积的整数，再用积的小数继续乘 2，依次下去，直到乘积是 0 位或达到要求的精度进行“四舍五入”为止，其积的整数部分即为对应的二进制数的小数部分。

例 1.2.5 将$(0.706)_D$转换为二进制数，要求其误差不大于 2^{-10}。

解：

$0.706 \times 2 = 1.412 \ldots\ldots\ldots\ldots\ldots\ldots 1 \ldots\ldots\ldots b_{-1}$

$0.412 \times 2 = 0.824 \ldots\ldots\ldots\ldots\ldots\ldots 0 \ldots\ldots\ldots b_{-2}$

$0.824 \times 2 = 1.648 \ldots\ldots\ldots\ldots\ldots\ldots 1 \ldots\ldots\ldots b_{-3}$

$0.648 \times 2 = 1.296 \ldots\ldots\ldots\ldots\ldots\ldots 1 \ldots\ldots\ldots b_{-4}$

$0.296 \times 2 = 0.592 \ldots\ldots\ldots\ldots\ldots\ldots 0 \ldots\ldots\ldots b_{-5}$

$0.592 \times 2 = 1.184 \ldots\ldots\ldots\ldots\ldots\ldots 1 \ldots\ldots\ldots b_{-6}$

$0.184 \times 2 = 0.368 \ldots\ldots\ldots\ldots\ldots\ldots 0 \ldots\ldots\ldots b_{-7}$

$0.386 \times 2 = 0.735 \ldots\ldots\ldots\ldots\ldots\ldots 0 \ldots\ldots\ldots b_{-8}$

$0.736 \times 2 = 1.472 \ldots\ldots\ldots\ldots\ldots\ldots 1 \ldots\ldots\ldots b_{-9}$

通过乘 2 取整，直到乘积是 0 位或达到要求的精度，并将保留的整数自上而下级联。因要求其误差不大于 2^{-10}，所以乘到第 9 次 2 时，1.472 的小数小于 0.5，b_{-10}应为 0，得到二进制数为$(0.706)_D = (0.101101001)_B$，其误差不大于 2^{-10}。

依此类推，对于十进制数转换成其他进制数，只要把基数 2 换成其他进制的基数即可。

3. 其他进制间的转换

由于 3 位二进制数可以有 8 个状态，即 000～111，正好是八进制；而 4 位二进制数可以有 16 个状态，即 0000～1111，正好是十六进制，故可以把二进制数进行分组，其整数部分和小数部分可以同时进行转换。其方法是：以二进制数的小数点为基准，分别向左、向右，每 3 位（或 4 位）分一组。对于小数部分，最低位一组不足 3 位（或 4 位）时，必须在有效位右边补零，使其足位。然后，把每一组二进制数转换成八进制（或十六进制）数，并保持原排序。对于整数部分，最高位一组不足位时，可在有效位的左边补零，也可不补。

例 1.2.6　将$(1011110.1011001)_B$转换成八进制数和十六进制数。

解：$(1011110.1011001)_B=(001\ 011\ 110.101\ 100\ 100)_B$

$= (136.544)_O$

$(1011110.1011001)_B=(0101\ 1110.1011\ 0010)_B$

$= (5E.B2)_H$

例 1.2.7　将$(703.65)_O$和$(9F12.04A)_H$转换成二进制数。

解：$(703.65)_O=(111\ 000\ 011.110\ 101)_B$

$(9F12.04A)_H=(1001\ 1111\ 0001\ 0010.0000\ 0100\ 1010)_B$

若要将十进制转换成八进制或十六进制，可先转换成二进制，再分组后转换成八进制或十六进制。十进制、二进制、八进制和十六进制之间的关系如表 1.2.1 所示。

表 1.2.1　0～15 数码不同数制之间的关系对照

十进制数	二进制数	八进制数	十六进制数	十进制数	二进制数	八进制数	十六进制数
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

例 1.2.8　将$(3DB)_H$转换成八进制。

解：$(3DB)_H=(0011\ 1101\ 1011)_B=(001\ 111\ 011\ 011)_B=(1733)_O$

复习思考题

1.2.1　写出 5 位二进制数、5 位八进制数和 5 位十六进制数的最大数。

1.2.2　在十-二进制数转换中，整数部分的转换方法和小数部分的转换方法有何不同？

1.2.3　为什么在计算机或数字系统中通常采用二进制数？

1.3　二进制算术运算

1.3.1　二进制算术运算的特点

当两个二进制数码表示两个数量大小时，它们之间可以进行算术运算。二进制数的加、减、乘、除 4 种运算的运算规则与十进制数基本相同，两者唯一的区别在于进位或借位规则不同。

例 1.3.1　计算两个二进制数 1011 和 0001 的和。

解：

$$\begin{array}{r}
1011\\
+\ 0001\\
\hline
1100
\end{array}$$

所以 $1011+0001=1100$。

二进制数的加法规则是“逢二进一”。

例 1.3.2　计算两个二进制数 1010 和 1001 的差。

解：

$$\begin{array}{r}
1010\\
-\ 1001\\
\hline
0001
\end{array}$$

所以 $1010-1001=0001$。

二进制数的减法规则是“借一当二”。

例 1.3.3　计算两个二进制数 1010 和 0101 的积。

解：

$$\begin{array}{r}
1010\\
\times\ 0101\\
\hline
1010\\
0000\\
1010\\
0000\\
\hline
110010
\end{array}$$

所以 $1010\times0101=110010$。

由上述运算过程可见，乘法运算是由左移被乘数与加法运算组成的。

例 1.3.4　计算两个二进制数 1001 和 0101 之商。

解：

$$
\begin{array}{r}
1.11\cdots \\
0101 \overline{)\,1001} \\
\underline{0101} \\
1000 \\
\underline{0101} \\
0110 \\
\underline{0101} \\
0010
\end{array}
$$

所以 $1001 \div 0101 = 1.11\cdots$。

由上述运算过程可见，除法运算是由右移被除数与减法运算组成的。

若能将减法运算转化为某种形式的加法运算，那么全部的算术运算只需要用“移位”和“相加”两种操作完成。数字电路中运算电路的结构就会大为简化。

1.3.2　反码、补码和补码运算

1. 原码

前面只考虑了二进制数的正数，当涉及负数时，就要用带符号的二进制数表示。那么，数的正、负又如何表示呢？通常采用的方法是在二进制数的前面增加一位符号位。正数的符号位用“0”表示，负数的符号位用“1”表示。这种形式的二进制数称为原码。例如

$$(+5)_D = (0101)_{原}$$
$$(-5)_D = (1101)_{原}$$

二进制 X 的原码的定义为

$$[X]_{原} = \begin{cases} 0X' \\ 1X' \end{cases} \tag{1.3.1}$$

式(1.3.1)中 X' 是数值部分。

为了简化电路，在数字电路中，将负数用补码表示，可将减法运算变为加法运算。下面，首先介绍补码的概念，然后举例说明负数的求补方法及减法运算，同时，为了便于得到补码，引入反码的概念。

2. 反码

正数的反码与原码相同，负数的原码除了符号位外的数值部分按位取反，即 1 改为 0，0 改为 1，在电路上是很容易实现的。例如

$$(+5)_D = (0101)_{原} = (0101)_{反}$$
$$(-5)_D = (1101)_{原} = (1010)_{反}$$

二进制 X 的反码的定义为

$$[X]_{反} = \begin{cases} 0X' \\ 1[(2^n - 1) - X'] \end{cases} \tag{1.3.2}$$

式(1.3.2)中 n 是码的位数(包括符号位)，X' 是数值部分。

3. 补码

正数的补码与原码相同，负数的补码除了符号位外的数值部分按位取反加 1。例如

$$(+5)_D = (0101)_{原} = (0101)_{补}$$
$$(-5)_D = (1101)_{原} = (1011)_{补}$$

二进制 X 的补码的定义为

$$[X]_{补} = \begin{cases} 0X' \\ 1[2^n - X'] \end{cases} \tag{1.3.3}$$