第1章 基本知识

欢迎来到数字系统设计领域！众所周知，21世纪是信息的时代，如何对各种各样的信息进行描述、传递、处理和存储呢？应该说，人类迄今为止找到的最佳信息表达形式是“数字”！数字系统已经成为各个领域，乃至人们日常生活的重要组成部分。在我们研究计算机以及其他数字系统中那些神奇的硬件是如何工作的之前，必须首先了解有关数字系统设计的基本知识。本章在对数字系统基本概念作简单介绍的基础上，重点讨论数字系统中数据的表示形式。

1.1 概 述

1.1.1 数字系统

什么是数字系统？简单地说，数字系统是一个能够对数字信号进行加工、传递和存储的实体，它由实现各种功能的数字逻辑电路相互连接而成。例如，广泛应用于科学计算、数据处理和过程控制等领域的数字计算机就是一种典型的数字系统。

1. 数字信号

在客观世界中，存在各种不同的物理量。按其变化规律可以分为两种类型：一类是连续量，另一类是数字量。所谓连续量是指在时间上和数值上均作连续变化的物理量，例如，温度、压力等。在工程应用中，为了方便处理和传送这种物理量，通常用某一种连续量去模拟另一种连续量，例如，用电压的变化模拟温度的变化等。因此，人们习惯将连续量称为**模拟量**。表示模拟量的信号称为**模拟信号**。对模拟信号进行处理的电路称为模拟电路。反之，另一类物理量的变化在时间上和数值上都是离散的，或者说断续的。例如，学生成绩记录，工厂产品统计，电路开关的状态等。这类物理量的变化可以用不同的数字反映，所以称为**数字量**。表示数字量的信号称为**数字信号**。

数字系统中处理的是数字信号，当数字系统要与模拟信号发生联系时，必须经模/数(A/D)转换电路和数/模(D/A)转换电路对信号类型进行变换。例如，某控制系统框图如图1.1所示。

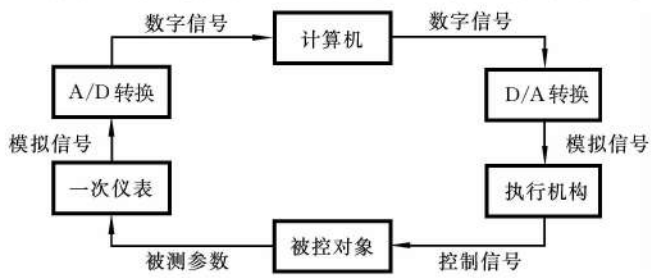


图1.1 某控制系统框图

2. 数字逻辑电路

用来处理数字信号的电子线路称为**数字电路**。由于数字电路的各种功能是通过逻辑运算和逻辑判断来实现的，所以，又将数字电路称为**数字逻辑电路**或者**逻辑电路**。数字逻辑电路与模拟电路相比，具有如下特点：

① 电路的基本工作信号是二值信号。它表现为电路中电压的“高”或“低”、开关的“接通”或“断开”、晶体管的“导通”或“截止”等两种稳定的物理状态。

② 电路中的半导体器件一般都工作在开、关状态，对电路进行研究时，主要关心输出和输入之间的逻辑关系。

③ 电路结构简单、功耗低、便于集成制造和系列化生产。产品价格低廉、使用方便、通用性好。

④ 由数字逻辑电路构成的数字系统工作速度快、精度高、功能强、可靠性好。

由于具有上述特点，所以数字逻辑电路的应用十分广泛。随着半导体技术和工艺的发展，出现了数字集成电路，人们已不再用分立元件去构造实现各种逻辑功能的部件，而是采用标准集成电路进行逻辑设计。因此，数字集成电路是数字系统实现各种功能的物质基础。

数字集成电路的基本逻辑单元是逻辑门，任何一个复杂的数字部件均可由逻辑门构成。一块集成电路芯片所容纳的逻辑门数量反映了芯片的集成度，集成度越高，单个芯片所实现的逻辑功能就越强。通常，按照单个芯片所集成的逻辑门数量将数字集成电路分为小规模(SSI)、中规模(MSI)、大规模(LSI)和超大规模(VLSI)等几种类型。

3. 数字系统的层次结构

任何复杂的数字系统都是由最底层的基本电路开始逐步向上构建起来的。从底向上，复杂度逐层增加，功能不断增强。图 1.2 所示为数字系统的层次结构。

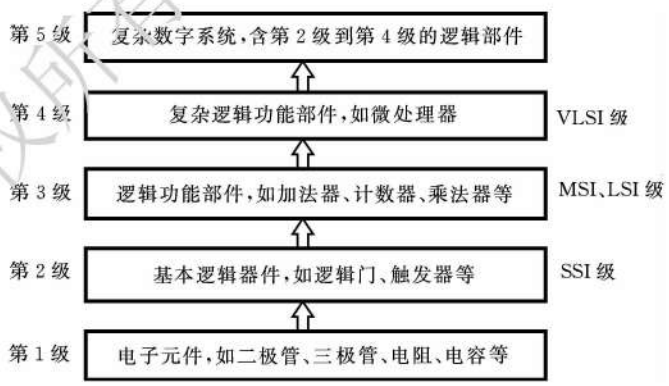


图 1.2　数字系统的层次结构

如上所述，集成电路是构成数字系统的物质基础，数字系统设计时考虑的基本逻辑单元为逻辑门，一旦理解了基本逻辑门的工作原理，便不必过于关心门电路内部电子线路的细节，而是更多地关注它们的外部特性及用途，以便实现更高一级的逻辑功能。

4. 典型的数字系统——数字计算机

数字计算机是一种能够自动、高速、精确地完成数值计算、数据加工和控制、管理等功能的数字系统。

(1) 数字计算机的组成

数字计算机由存储器、运算器、控制器、输入设备、输出设备以及适配器等主要部分组成，各部分通过总线连成一个整体。图 1.3 所示为数字计算机的一般结构。

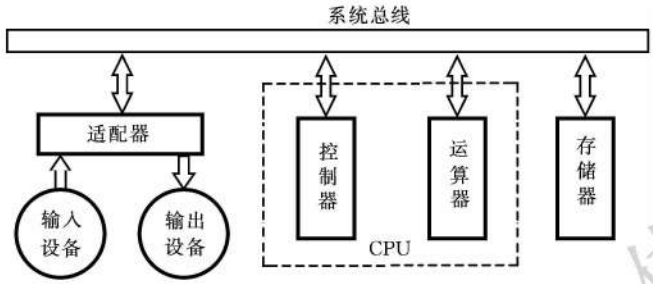


图 1.3 数字计算机的一般结构

(2) 计算机的发展历程

1946 年，在美国宾夕法尼亚大学诞生了世界上第一台数字计算机——ENIAC，人们将其誉为数字计算机的始祖。该台机器共用了 18 000 多个电子管，占用空间的长度超过 30 m，重量达 30 t，而运算速度仅 5 000 次/s。显然，用现代的眼光来看这是一台体积庞大、耗费多而又十分低效的计算机，但正是这台机器的研制成功奠定了数字计算机的基础，成为科学史上一次划时代的创新。

根据组成计算机的主要元器件的不同，计算机的发展历程如表 1.1 所示。

表 1.1 数字计算机的发展历程

主要器件	开始时间(年)	运算速度
电子管	1946	几千次/秒～几万次/秒
晶体管	1958	几万次/秒～几十万次/秒
小、中规模集成电路	1965	几十万次/秒～几百万次/秒
大规模、超大规模集成电路	1971	几百万次/秒～几千万次/秒
巨大规模集成电路	1986	几千万次/秒～几百亿次/秒

计算机发展的历程表明，从 1946 年以来，总的发展规律是大约每隔 5 年运算速度提高 10 倍，可靠性提高 10 倍，成本降低 10 倍，体积缩小 10 倍。自 1970 年以来，计算机的生产数量每年递增 25%。可见，计算机的发展速度是十分惊人的。据专家预测，这种发展趋势还将继续维持至少 10 年以上。

1.1.2 数字逻辑电路的类型和研究方法

1. 数字逻辑电路的类型

根据一个电路有无记忆功能，可将数字逻辑电路分为组合逻辑电路和时序逻辑电路两种类型。

如果一个逻辑电路在任何时刻的稳定输出仅取决于该时刻的输入，而与电路过去的输入无关，则称之为**组合逻辑电路**(Combinational Logic Circuit)。由于这类电路的输出与过去的输入信号无关，所以不需要有记忆功能。例如，一个“多数表决器”，由于表决的结果仅取决于各个参与表决成员当时的态度是“赞成”还是“反对”，因此，它属于组合电路。

如果一个逻辑电路在任何时刻的稳定输出不仅取决于该时刻的输入，而且与过去的输入相关，则称之为**时序逻辑电路**(Sequential Logic Circuit)。由于这类电路的输出与过去的输入相关，所以需要有记忆功能，通常采用电路中记忆元件的状态来反映过去的输入信号。例如，一个统计输入脉冲信号个数的计数器，它的输出结果不仅与当时的输入脉冲相关，还与前面收到的脉冲个数相关，因此，计数器是一个时序逻辑电路。时序逻辑电路按照是否有统一的时钟信号进行同步，又可进一步分为**同步时序逻辑电路**和**异步时序逻辑电路**。

2. 数字逻辑电路的研究方法

对数字系统中逻辑电路的研究有两个主要任务：一是分析，二是设计。对一个给定的数字逻辑电路，研究它所实现的逻辑功能和电路的工作性能称为**逻辑电路分析**；根据客观提出的功能要求，在给定条件下构造出实现预定功能的逻辑电路称为**逻辑电路设计**，简称**逻辑设计**或者**逻辑综合**。

随着集成电路技术的飞跃发展，数字逻辑电路的分析和设计方法在不断发生变化。但不管怎样变化，用逻辑代数作为基本理论的传统方法仍不失为逻辑电路分析和设计的基本方法。传统方法详细讨论了从问题的逻辑抽象到功能实现的全过程，可以说是至今为止最成熟、最基本的方法。该方法是建立在小规模集成电路基础之上的，它以技术经济指标作为评价一个设计方案优劣的主要性能指标，设计时追求的是如何使一个电路达到最简。因此，在组合逻辑电路设计时，通过逻辑函数化简，尽可能使电路中的逻辑门和连线数目达到最少。而在时序逻辑电路设计时，则通过状态化简和逻辑函数化简，尽可能使电路中的记忆元件、逻辑门和连线数目达到最少。但值得指出的是，一个**最简**的方案并不等于一个**最佳**的方案，最佳方案应满足全面的性能指标和实际应用要求。所以，在用传统方法求出一个实现预定功能的最简方案之后，往往要根据实际情况进行相应调整。

随着中、大规模集成电路的出现和集成电路规模的迅速发展，单个芯片内部容纳的逻辑功能越来越强，因而实现某种逻辑功能所需要的门和触发器数量不再成为影响经济指标的突出问题。如何用各种廉价的中、大规模集成电路组件构造满足各种功能的、经济合理的电路，这无疑给设计人员提出了更高的要求。要适应这种要求就必须充分了解各种器件的逻辑结构和外部特性，做到合理选择器件，充分利用每一个已选器件的功能，用灵活多变的方法完成各类逻辑电路或功能模块的设计。此外，**可编程逻辑器件**(Programmable Logic Devices，简称 PLD)的出现，给逻辑设计带来了一种全新的方法。人们不再用常规硬线连接的方法去构造电路，而是借助丰富的计算机软件对器件编程来实现各种逻辑功能，这无疑给逻辑设计者带来了极大的方便。

其次，面对日益复杂的集成电路芯片设计和数字系统设计，人们不得不越来越多地借助计算机辅助设计(Computer Aided Design，简称 CAD)。目前，已有各种电子设计自动化(Electronic Design Automatic，简称 EDA)软件在市场上出售。计算机辅助逻辑设计方法正在不断推广和应用。不少人认为计算机设计自动化已形成计算机科学中的一个独立的学科。

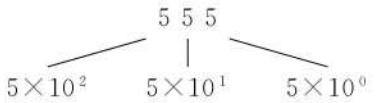
1.2 数制及其转换

1.2.1 进位计数制

数制是人们对数量计数的一种统计规律。日常生活中广泛使用的是十进制，而数字系统

中使用的是二进制。

十进制中采用了 0,1,…,9 共 10 个基本数字符号,进位规律是“逢十进一”。当用若干个数字符号并在一起表示一个数时,处在不同位置的数字符号,其值的含意不同。如



同一个字符 5 从左到右所代表的值依次为 500、50、5。该数又可表示成

$$5\times10^{2}+5\times10^{1}+5\times10^{0}$$

广义地说,一种进位计数制包含着基数和位权两个基本要素:

● **基数**是指计数制中所用到的数字符号的个数。在基数为 R 的计数制中,包含 0,1,…,$R-1$ 共 R 个数字符号,进位规律是“逢 R 进一”,称为 R 进位计数制,简称 R 进制。

● **位权**是指在一种进位计数制表示的数中,用来表明不同数位上数值大小的一个固定常数。不同数位有不同的位权,某一个数位的数值等于这一位的数字符号乘上与该位对应的位权。R 进制数的位权是 R 的整数次幂。例如,十进制数的位权是 10 的整数次幂,其个位的位权是 10^{0},十位的位权是 10^{1}……

一般来说,一个 R 进制数 N 可以有以下两种表示方法:

● **并列表示法**,又称位置计数法,其表达式为

$$(N)_R=(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_R$$

● **多项式表示法**,又称按权展开法,其表达式为

$$\begin{aligned}(N)_R&=K_{n-1}\times R^{n-1}+K_{n-2}\times R^{n-2}+\cdots+K_1\times R^{1}+K_0\times R^{0}\\&\quad+K_{-1}\times R^{-1}+K_{-2}\times R^{-2}+\cdots+K_{-m}\times R^{-m}\\&=\sum_{i=-m}^{n-1}K_iR^{i}\end{aligned}$$

其中,R 表示基数;n 为整数部分的位数;m 为小数部分的位数;K_i 为 R 进制中的一个数字符号,其取值范围为

$$0\leqslant K_i\leqslant R-1\qquad -m\leqslant i\leqslant n-1$$

例如,十进制数 2005.18 可以表示成

$$(2005.18)_{10}=2\times10^{3}+0\times10^{2}+0\times10^{1}+5\times10^{0}+1\times10^{-1}+8\times10^{-2}$$

1. 二进制

基数 $R=2$ 的进位计数制称为二进制。二进制数中只有 0 和 1 两个基本数字符号,进位规律是“逢二进一”。二进制数的位权是 2 的整数次幂。

任意一个二进制数 N 可以表示成

$$\begin{aligned}(N)_2&=(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_2\\&=K_{n-1}\times2^{n-1}+K_{n-2}\times2^{n-2}+\cdots+K_1\times2^{1}+K_0\times2^{0}\\&\quad+K_{-1}\times2^{-1}+K_{-2}\times2^{-2}+\cdots+K_{-m}\times2^{-m}\\&=\sum_{i=-m}^{n-1}K_i\times2^{i}\end{aligned}$$

其中,n 为整数位数;m 为小数位数;K_i 为 0 或者 1,$-m\leqslant i\leqslant n-1$。

例如，一个二进制数 1011.01 可以表示成

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

二进制数的运算十分简便，其运算规则如下：

加法规则	0+0=0	0+1=1	
	1+0=1	1+1=0	（进位为 1）
减法规则	0−0=0	0−1=1	（借位为 1）
	1−0=1	1−1=0	
乘法规则	0×0=0	0×1=0	
	1×0=0	1×1=1	
除法规则	0÷1=0	1÷1=1	

例如，二进制数 $A=11001$，$B=101$，则 $A+B$，$A-B$，$A\times B$，$A\div B$ 的运算为

```
    1 1 0 0 1          1 1 0 0 1
  +       1 0 1      -       1 0 1
  ---------------    ---------------
    1 1 1 1 0          1 0 1 0 0

    1 1 0 0 1                1 0 1
  × 1 0 1          101 ) 1 1 0 0 1
  ---------------          - 1 0 1
    1 1 0 0 1          ---------------
  0 0 0 0 0                  1 0 1
+ 1 1 0 0 1                - 1 0 1
---------------          ---------------
1 1 1 1 1 0 1                    0
```

二进制除了运算简单之外，还具有物理实现容易，存储和传送方便、可靠等优点。因为二进制中只有 0 和 1 两个数字符号，可以用电子器件的两种不同状态来表示一位二进制数，例如，可以用晶体管的截止和导通表示 1 和 0，也可以用电平的高和低表示 1 和 0 等，所以在数字系统中普遍采用二进制。

二进制的缺点是数的位数太长且字符单调，使得书写、记忆和阅读不方便。因此，人们在进行指令书写、程序输入和输出等工作时，通常采用八进制数或十六进制数作为二进制数的缩写。

2. 八进制

基数 $R=8$ 的进位计数制称为八进制。八进制数中有 0，1，…，7 共 8 个基本数字符号，进位规律是“逢八进一”。八进制数的位权是 8 的整数次幂。

任意一个八进制数 N 可以表示成

$$\begin{aligned}(N)_8&=(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_8\\&=K_{n-1}\times8^{n-1}+K_{n-2}\times8^{n-2}+\cdots+K_1\times8^1+K_0\times8^0\\&\quad+K_{-1}\times8^{-1}+K_{-2}\times8^{-2}+\cdots+K_{-m}\times8^{-m}\\&=\sum_{i=-m}^{n-1}K_i\times8^i\end{aligned}$$

其中，n 为整数位数，m 为小数位数，K_i 表示 0～7 中的任何一个字符，$-m \leqslant i \leqslant n-1$。

3. 十六进制

基数 $R=16$ 的进位计数制称为十六进制。十六进制数中有 0，1，…，9，A，B，C，D，E，F 共 16 个数字符号，其中，A～F 分别表示十进制数的 10～15。进位规律为“逢十六进一”，十六进制数的位权是 16 的整数次幂。

任意一个十六进制数 N 可以表示成

$$\begin{aligned}(N)_{16} &= (K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_{16}\\ &= K_{n-1}\times 16^{n-1}+K_{n-2}\times 16^{n-2}+\cdots+K_1\times 16^1+K_0\times 16^0\\ &\quad +K_{-1}\times 16^{-1}+K_{-2}\times 16^{-2}+\cdots+K_{-m}\times 16^{-m}\\ &= \sum_{i=-m}^{n-1} K_i\times 16^i\end{aligned}$$

其中，n 为整数位数，m 为小数位数，K_i 表示 0～9 及 A～F 中的任何一个字符，$-m \leqslant i \leqslant n-1$。

表 1.2 列出了与十进制数 0～16 对应的二进制数、八进制数、十六进制数。

表 1.2　十进制数与二、八、十六进制数对照表

十进制数	二进制数	八进制数	十六进制数
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

1.2.2　数制转换

人们习惯使用的是十进制数，数字系统中普遍采用的是二进制数。人们在与二进制数打交道时，为了书写和阅读的方便，又通常使用八进制数或十六进制数，因此，产生了不同进位计数制之间的转换问题。下面讨论二进制数与十进制数、八进制数和十六进制数之间的相互转换。

1. 二进制数与十进制数之间的转换

(1) 二进制数转换为十进制数

二进制数转换成十进制数非常简单，只需将二进制数表示成按权展开式，并按十进制数的

运算法则进行计算，所得结果即为与该数对应的十进制数。例如，

$$\begin{aligned}(10110.101)_2 &= 1\times2^4+1\times2^2+1\times2^1+1\times2^{-1}+1\times2^{-3}\\ &= 16+4+2+0.5+0.125\\ &= (22.625)_{10}\end{aligned}$$

(2) 十进制数转换为二进制数

十进制数转换成二进制数时，应对整数和小数分别进行处理。整数转换采用**除 2 取余**的方法，小数转换采用**乘 2 取整**的方法。

1) 整数转换

整数转换采用**除 2 取余**法。将十进制整数 N 除以 2，取余数记为 K_0；再将所得商除以 2，取余数记为 K_1……以此类推，直至商为 0，取余数记作 K_{n-1} 为止，即可得到与 N 对应的 n 位二进制整数 $K_{n-1}\cdots K_1K_0$。

例如，将十进制整数 45 转换成二进制整数：

2	45	余数	
2	22	……1(K_0)	低位 ↑
2	11	……0(K_1)	
2	5	……1(K_2)	
2	2	……1(K_3)	
2	1	……0(K_4)	
	0	……1(K_5)	高位

即 $(45)_{10}=(101101)_2$

2) 小数转换

小数转换采用**乘 2 取整**法。将十进制小数 N 乘以 2，取整数部分记为 K_{-1}；再将其小数部分乘以 2，取整数部分记为 K_{-2}……以此类推，直至其小数部分为 0 或达到规定精度要求，取整数部分记作 K_{-m} 为止，即可得到与 N 对应的 m 位二进制小数 $0.K_{-1}K_{-2}\cdots K_{-m}$。

例如，将十进制小数 0.6875 转换成二进制小数：

			0.6875
	整数部分	×	2
高位	1(K_{-1})……		1.3750
		×	2
	0(K_{-2})……		0.7500
		×	2
	1(K_{-3})……		1.5000
		×	2
低位 ↓	1(K_{-4})……		1.0000

即 $(0.6875)_{10}=(0.1011)_2$

值得注意的是，有的十进制小数不能用有限位二进制小数精确表示。这时只能根据精度要求，求出相应的二进制位数近似地表示。一般当要求二进制数取 m 位小数时，可求出 $m+1$ 位，然后对最低位作 **0 舍 1 入**处理。

例如，将十进制小数 0.323 转换成二进制小数（保留 4 位小数）：

$$
\begin{array}{r}
0.323 \\
\times\quad 2 \\
\hline
\boxed{0}.646 \\
\times\quad 2 \\
\hline
\boxed{1}.292 \\
\times\quad 2 \\
\hline
\boxed{0}.584 \\
\times\quad 2 \\
\hline
\boxed{1}.168 \\
\times\quad 2 \\
\hline
\boxed{0}.336
\end{array}
$$

即　$(0.323)_{10} \approx (0.0101)_2$

此外，当一个十进制数既包含整数部分，又包含小数部分时，只需将整数部分和小数部分分别转换为二进制数，然后用小数点将两部分结果连起来即可。

例如，将十进制数 35.625 转换成二进制数：

$$
\begin{array}{r|l}
2 & 35 \\
2 & 17 \quad \cdots\cdots 1 \\
2 & 8 \quad \cdots\cdots 1 \\
2 & 4 \quad \cdots\cdots 0 \\
2 & 2 \quad \cdots\cdots 0 \\
2 & 1 \quad \cdots\cdots 0 \\
 & 0 \quad \cdots\cdots 1
\end{array}
$$

$$
\begin{array}{r}
0.625 \\
\times\quad 2 \\
\hline
\boxed{1}.250 \\
\times\quad 2 \\
\hline
\boxed{0}.500 \\
\times\quad 2 \\
\hline
\boxed{1}.000
\end{array}
$$

即　$(35.625)_{10} = (100011.101)_2$

2. 二进制数与八进制数、十六进制数之间的转换

(1) 二进制数与八进制数之间的转换

由于 $2^3 = 8$，所以 1 位八进制数所能表示的数值恰好等于 3 位二进制数所能表示的数值，即八进制中的基本数字符号 0～7 正好和 3 位二进制数的 8 种取值 000～111 对应。因此，二进制数与八进制数之间的转换可以按位进行。

二进制数转换成八进制数时，以小数点为界，分别往高、往低每 3 位为一组，最后不足 3 位时用 0 补充，然后写出每组对应的八进制字符，即为相应的八进制数。

例如，将二进制数 11100101.01 转换成八进制数：

$$
\underbrace{0\ 1\ 1}_{3}\ \underbrace{1\ 0\ 0}_{4}\ \underbrace{1\ 0\ 1}_{5}\ .\ \underbrace{0\ 1\ 0}_{2}
$$

即　$(11100101.01)_2 = (345.2)_8$

八进制数转换成二进制数时，只需将每位八进制数用 3 位二进制数表示。

例如，将八进制数 56.7 转换成二进制数：

$$\overbrace{1\ 0\ 1}^{5}\ \overbrace{1\ 1\ 0}^{6}\ .\ \overbrace{1\ 1\ 1}^{7}$$

即 $(56.7)_8=(101110.111)_2$

(2) 二进制数与十六进制数之间的转换

二进制数与十六进制数之间的转换类似于二进制数与八进制数之间的转换，只不过是4位二进制数对应1位十六进制数，即4位二进制数的取值0000～1111分别对应十六进制字符0～F。

二进制数转换成十六进制数时，以小数点为界，分别往高、往低每4位为一组，最后不足4位时用0补充，然后写出每组对应的十六进制字符即可。

例如，将二进制数101110.011转换成十六进制数：

$$\underbrace{0\ 0\ 1\ 0}_{2}\ \underbrace{1\ 1\ 1\ 0}_{E}\ .\ \underbrace{0\ 1\ 1\ 0}_{6}$$

即 $(101110.011)_2=(2E.6)_{16}$

十六进制数转换成二进制数时，只需将每位十六进制数用4位二进制数表示。

例如，将十六进制数5A.B转换成二进制数：

$$\overbrace{0\ 1\ 0\ 1}^{5}\ \overbrace{1\ 0\ 1\ 0}^{A}\ .\ \overbrace{1\ 0\ 1\ 1}^{B}$$

即 $(5A.B)_{16}=(1011010.1011)_2$

由于八进制数、十六进制数与二进制数的转换很方便，因此，常用作二进制数的缩写。

1.3 带符号二进制数的代码表示

在对带符号数进行算术运算时，必然涉及数的符号问题。人们通常在一个数的前面用“+”号表示正数，用“-”号表示负数。而在数字系统中，符号和数值一样是用0和1来表示的，一般将数的最高位作为符号位，用0表示正，用1表示负。其格式为

符号位	数值位

为了区分一般书写表示的带符号二进制数和数字系统中的带符号二进制数，通常将用“+”、“-”表示正、负的二进制数称为符号数的**真值**，而把将符号和数值一起编码表示的二进制数称为**机器数**或**机器码**。常用的机器码有原码、反码和补码三种。

1.3.1 原码

用原码表示带符号二进制数时，符号位用0表示正，1表示负；数值位保持不变。原码表示法又称为符号-数值表示法。

1. 小数原码的定义

设二进制小数 $X=\pm 0.x_{-1}x_{-2}\cdots x_{-m}$，则其原码定义为

$$[X]_{原}=\begin{cases}X & 0\leqslant X<1\\ 1-X & -1<X\leqslant 0\end{cases}$$

例如，若 $X_1=+0.1011$，$X_2=-0.1011$，则 X_1 和 X_2 的原码为

$$[X_1]_{原}=0.1011$$
$$[X_2]_{原}=1-(-0.1011)=1.1011$$

根据定义，小数“0”的原码可以表示成 0.0…0 或 1.0…0。

2. 整数原码的定义

设二进制整数 $X=\pm x_{n-1}x_{n-2}\cdots x_0$，则其原码定义为

$$[X]_{原}=\begin{cases}X & 0\leqslant X<2^n\\ 2^n-X & -2^n<X\leqslant 0\end{cases}$$

例如，若 $X_1=+1101$，$X_2=-1101$，则 X_1 和 X_2 的原码为

$$[X_1]_{原}=01101$$
$$\begin{aligned}[X_2]_{原}&=2^4-(-1101)\\&=10000+1101\\&=11101\end{aligned}$$

同样，整数“0”的原码也有两种形式，即 00…0 和 10…0。

采用原码表示带符号的二进制数简单易懂，但实现加、减运算不方便。当进行两数加、减运算时，要根据运算及参加运算的两个数的符号来确定是加还是减。如果是做减法，则还需根据两数的大小确定被减数和减数，以及运算结果的符号。显然，这将增加运算的复杂性。

1.3.2　反码

用反码表示带符号的二进制数时，符号位与原码相同，即用 0 表示正，用 1 表示负；数值位与符号位相关，正数反码的数值位和真值的数值位相同，而负数反码的数值位是真值的数值位按位变反。

1. 小数反码的定义

设二进制小数 $X=\pm 0.x_{-1}x_{-2}\cdots x_{-m}$，则其反码定义为

$$[X]_{反}=\begin{cases}X & 0\leqslant X<1\\ (2-2^{-m})+X & -1<X\leqslant 0\end{cases}$$

例如，若 $X_1=+0.1011$，$X_2=-0.1011$，则 X_1 和 X_2 的反码为

$$[X_1]_{反}=0.1011$$
$$\begin{aligned}[X_2]_{反}&=2-2^{-4}+X_2\\&=10.0000-0.0001-0.1011\\&=1.0100\end{aligned}$$

根据定义，小数“0”的反码有两种表示形式，即 0.0…0 和 1.1…1。

2. 整数反码的定义

设二进制整数 $X=\pm x_{n-1}x_{n-2}\cdots x_0$，则其反码定义为

$$[X]_{反}=\begin{cases}X & 0\leqslant X<2^n\\ (2^{n+1}-1)+X & -2^n<X\leqslant 0\end{cases}$$

例如，若 $X_1=+1001$，$X_2=-1001$，则 X_1 和 X_2 的反码为

$$[X_1]_{反}=01001$$
$$[X_2]_{反}=(2^5-1)+X$$
$$=(100000-1)+(-1001)$$
$$=11111-1001$$
$$=10110$$

同样，整数“0”的反码也有两种形式，即 00…0 和 11…1。

采用反码进行加、减运算时，无论两数相加还是两数相减，均可通过加法实现。其加、减运算规则如下：

$$[X_1+X_2]_{反}=[X_1]_{反}+[X_2]_{反}$$
$$[X_1-X_2]_{反}=[X_1]_{反}+[-X_2]_{反}$$

运算时，符号位和数值位一样参加运算。当符号位有进位产生时，应将进位加到运算结果的最低位，才能得到最后结果。

例如，若 $X_1=+0.1110$，$X_2=+0.0101$，则求 X_1-X_2 可通过反码相加实现。运算如下：

$$[X_1-X_2]_{反}=[X_1]_{反}+[-X_2]_{反}=0.1110+1.1010$$

$$\begin{array}{rccccc} & 0. & 1 & 1 & 1 & 0 \\ + & 1. & 1 & 0 & 1 & 0 \\ \hline & \boxed{1} & 0. & 1 & 0 & 0 \\ + & & & & & \longrightarrow 1 \\ \hline & 0. & 1 & 0 & 0 & 1 \end{array}$$

即 $[X_1-X_2]_{反}=0.1001$

由于结果的符号位为 0，表示是正数，故

$$X_1-X_2=+0.1001$$

1.3.3 补码

用补码表示带符号的二进制数时，符号位与原码、反码相同，即用 0 表示正，用 1 表示负；数值位与符号位相关，正数补码的数值位与真值相同，而负数补码的数值位是真值的数值位按位变反，并在最低位加 1。

1. 小数补码的定义

设二进制小数 $X=\pm 0.x_{-1}x_{-2}\cdots x_{-m}$，则其补码定义为

$$[X]_{补}=\begin{cases} X & 0\leqslant X<1 \\ 2+X & -1\leqslant X<0 \end{cases}$$

例如，若 $X_1=+0.1011$，$X_2=-0.1011$，则 X_1 和 X_2 的补码为

$$[X_1]_{补}=0.1011$$
$$[X_2]_{补}=2+X$$
$$=10.0000-0.1011$$
$$=1.0101$$

小数“0”的补码只有一种表示形式，即 0.0…0。

2. 整数补码的定义

设二进制整数 $X=\pm x_{n-1}x_{n-2}\cdots x_0$，则其补码定义为

$$[X]_{补}=\begin{cases}X & 0\leqslant X<2^n\\ 2^{n+1}+X & -2^n\leqslant X<0\end{cases}$$

例如，若 $X_1=+1010$，$X_2=-1010$，则 X_1 和 X_2 的补码为

$$[X_1]_{补}=01010$$
$$[X_2]_{补}=2^5+X$$
$$=100000-1010$$
$$=10110$$

同样，整数“0”的补码也只有一种表示形式，即 00…0。

采用补码进行加、减运算时，其加、减运算均可以通过加法实现，运算规则如下：

$$[X_1+X_2]_{补}=[X_1]_{补}+[X_2]_{补}$$
$$[X_1-X_2]_{补}=[X_1]_{补}+[-X_2]_{补}$$

运算时，符号位和数值位一样参加运算，若符号位有进位产生，则应将进位丢掉才能得到正确结果。

例如，若 $X_1=-1001$，$X_2=+0011$，则采用补码求 X_1-X_2 的运算如下：

$$[X_1-X_2]_{补}=[X_1]_{补}+[-X_2]_{补}=10111+11101$$

$$\begin{array}{lccccc} & 1 & 0 & 1 & 1 & 1\\ +) & 1 & 1 & 1 & 0 & 1\\ \hline \text{丢掉}\ \boxed{1} & 1 & 0 & 1 & 0 & 0\end{array}$$

即 $[X_1-X_2]_{补}=10100$

由于结果的符号位为 1，表示是负数，故

$$X_1-X_2=-1100$$

显然，采用补码进行加、减运算最方便。

1.4　几种常用的编码

1.4.1　十进制数的二进制编码

在数字系统的输入/输出过程中，为了既满足系统中使用二进制数的要求，又适应人们使用十进制数的习惯，通常使用 4 位二进制代码对十进制数字符号进行编码，简称为二-十进制代码，或称 BCD(Binary Coded Decimal)码。它既有二进制数的形式，又有十进制数的特点，便于传递、处理。

十进制数中有 0～9 共 10 个数字符号，4 位二进制代码可组成 16 种状态，从 16 种状态中取出 10 种状态来表示 10 个数字符号的编码方案很多，但不管哪种编码方案都有 6 种状态不允许出现。根据代码中每一位是否有固定的权，将 BCD 码分为有权码和无权码两种。常用的 BCD 码有 8421 码、2421 码和余 3 码 3 种，它们与十进制数字符号对应的编码如表 1.3 所示。

表 1.3 常用的 3 种 BCD 码

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

1. 8421 码

8421 码是最常用的一种有权码，其 4 位二进制码从高位至低位的权依次为2^3、2^2、2^1、2^0，即为 8、4、2、1。显然，这与普通的 4 位二进制数的权是一样的。因此，按 8421 码编码的 0～9 与用 4 位二进制数表示的 0～9 完全一样。值得注意的是，4 位二进制数中的 1010～1111 不允许在 8421 码中出现，因为没有十进制数字符号与其对应。此外，十进制数字符号的 8421 码与相应 ASCII 码的低 4 位相同，这一特点有利于简化输入/输出过程中 BCD 码与字符代码的转换。所以，8421 码是一种人机联系时广泛使用的中间形式。

8421 码与十进制数之间的转换是按位进行的，即十进制数的每一位与 4 位二进制编码对应。例如，

$$(258)_{10}=(0010\ 0101\ 1000)_{8421码}$$

$$(0001\ 0010\ 0000\ 1000)_{8421码}=(1208)_{10}$$

2. 2421 码

2421 码是另一种有权码，其 4 位二进制码从高位至低位的权依次为 2、4、2、1。若一个十进制字符 X 的 2421 码为 $a_3a_2a_1a_0$，则该字符的值为

$$X=2a_3+4a_2+2a_1+a_0$$

例如，

$$(1101)_{2421码}=(7)_{10}$$

2421 码不具备单值性，为了与十进制字符一一对应，2421 码不允许出现 0101～1010 的 6 种状态。

在 2421 码中，十进制字符 0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 的各码位互为相反。具有这种特性的代码称为**对 9 的自补代码**，即一个数的 2421 码只要自身按位变反，便可得到该数对 9 的补数的 2421 码。例如，4 对 9 的补数是 5，将 4 的 2421 码 0100 按位变反，便可得到 5 的 2421 码 1011。具有这一特征的 BCD 码可给运算带来方便，因为直接对 BCD 码进行运算时，可利用其对 9 的补数将减法运算转化为加法运算。

2421 码与十进制数之间的转换同样是按位进行的，例如，

$$(258)_{10}=(0010\ 1011\ 1110)_{2421码}$$

$$(0010\ 0001\ 1110\ 1011)_{2421码}=(2185)_{10}$$

3. 余 3 码

余 3 码是由 8421 码加上 0011 形成的一种无权码，由于它的每个字符编码比相应 8421

码多 3，故称为余 3 码。例如，十进制字符 5 的余 3 码等于 5 的 8421 码 0101 加上 0011，即为 1000。同样，余 3 码中也有 6 种状态 0000、0001、0010、1101、1110 和 1111 是不允许出现的。

余 3 码也是一种对 9 的自补代码，因而可给运算带来方便。其次，在将两个余 3 码表示的十进制数相加时，能正确产生进位信号，但对“和”必须修正。修正的方法是：如果有进位，则结果加 3；如果无进位，则结果减 3。

余 3 码与十进制数之间的转换也是按位进行的，值得注意的是每位十进制数的编码都应余 3。例如，

$$(256)_{10} = (0101\ 1000\ 1001)_{余3码}$$

$$(1000\ 1001\ 1001\ 1011)_{余3码} = (5668)_{10}$$

1.4.2　可靠性编码

顾名思义，可靠性编码的作用是为了提高系统的可靠性。代码在形成和传送过程中都可能发生错误。为了使代码本身具有某种特征或能力，尽可能减少错误的发生，或者出错后容易被发现，甚至检查出错误的码位后能予以纠正，因而形成了各种编码方法。下面介绍两种简单、常用的可靠性编码。

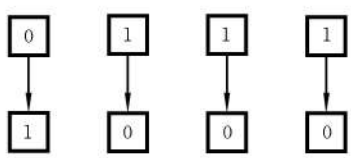
1. 格雷码

格雷码(Gray Code)的特点是：任意两个相邻的数，其格雷码仅有一位不同。表 1.4 给出了与 4 位二进制码对应的典型格雷码。

表 1.4　与 4 位二进制码对应的典型格雷码

十进制数	4 位二进制码	典型格雷码	十进制数	4 位二进制码	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

在数字系统中，数字 0 或 1 是用电子器件的不同状态来表示的。当数据按升序或降序变化时，若采用普通二进制数，则每次增 1 或者减 1 时，可能引起若干位发生变化。例如，用 4 位二进制数表示的十进制数由 7 变为 8 时，要求 4 位同时发生变化，即由 0111 变为 1000。与 4 位二进制数对应的 4 个电子器件的状态变化如下：



显然，当电子器件的变化速度不一致时，便会产生错误代码，如产生 1111(假定最高位变化比低 3 位快)、1001(假定最低位变化比高 3 位慢)等错误代码。尽管这种错误代码出现的时

间是短暂的，但有时是不允许的，因为它将形成干扰，影响数字系统的正常工作。格雷码从编码上杜绝了这类错误的发生。

格雷码可以从普通二进制数转换得到。设二进制数为 $B=B_{n-1}B_{n-2}\cdots B_{i+1}B_i\cdots B_1B_0$，对应格雷码为 $G=G_{n-1}G_{n-2}\cdots G_{i+1}G_i\cdots G_1G_0$，则有

$$G_{n-1}=B_{n-1}$$

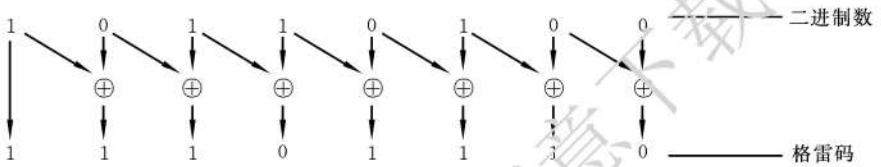
$$G_i=B_{i+1}\oplus B_i \qquad 0\leqslant i\leqslant n-2$$

其中，运算“⊕”称为“异或”运算，运算规则是

$$0\oplus 0=0 \qquad 0\oplus 1=1 \qquad 1\oplus 0=1 \qquad 1\oplus 1=0$$

异或运算又称为“模 2 加”，即不计进位的二进制加法。

例如，二进制数 10110100 对应的格雷码为 11101110，转换过程如下：



2. 奇偶检验码

二进制信息在传送时，可能由于外界干扰或其他原因而发生错误，即可能有的 1 错为 0 或者有的 0 错为 1。奇偶检验码(Parity Check Code)是一种能够检验出这类错误的代码。这种代码由两部分组成：一是信息位，即需要传递的信息本身，可以是位数不限的一组二进制代码；二是奇偶检验位，它仅有一位。奇偶检验位的编码方式有两种：一种是使信息位和检验位中“1”的个数为奇数，称为奇检验；另一种是使信息位和检验位中“1”的个数为偶数，称为偶检验。例如，二进制代码 1001101 奇偶检验码表示时的两种编码方式为

信息位(7 位)	采用奇检验的检验位(1 位)	采用偶检验的检验位(1 位)
1 0 0 1 1 0 1	1	0

表 1.5 列出了与 8421 码对应的奇偶检验码。

表 1.5　8421 码的奇偶检验码

十进制数码	采用奇检验的 8421 码		采用偶检验的 8421 码	
	信息位	检验位	信息位	检验位
0	0000	1	0000	0
1	0001	0	0001	1
2	0010	0	0010	1
3	0011	1	0011	0
4	0100	0	0100	1
5	0101	1	0101	0
6	0110	1	0110	0
7	0111	0	0111	1
8	1000	0	1000	1
9	1001	1	1001	0

奇偶检验码的工作原理如图 1.4 所示。

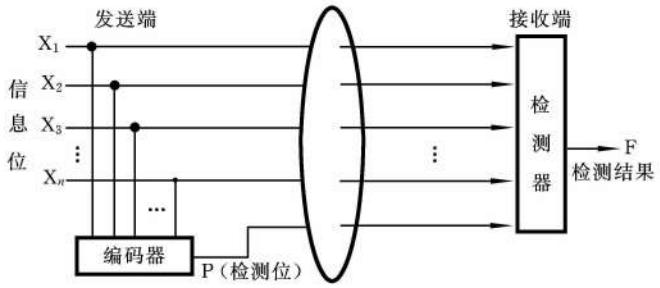


图 1.4　奇偶检验码的工作原理图

采用奇偶检验码进行错误检测时，在发送端由编码器根据信息位编码产生奇偶检验位，形成奇偶检验码发往接收端；接收端通过检测器检查代码中含“1”个数的奇偶，判断信息是否出错。例如，当采用偶检验时，若收到的代码中含奇数个“1”，则说明发生了错误。但判断出错后，并不能确定是哪一位出错，也就无法纠正。因此，奇偶检验码只有检错能力，没有纠错能力。其次，这种码只能发现单错，不能发现双错，但由于单错的概率远远大于双错，所以，这种编码还是很有实用价值的。加之它编码简单、容易实现，因而，在数字系统中被广泛采用。

*1.4.3　字符编码

数字系统中处理的数据除了数字之外，还有字母、运算符号、标点符号以及其他特殊符号，人们将这些符号统称为字符。所有字符在数字系统中必须用二进制编码表示，通常将其称为字符编码（Alphanumeric Code）。

最常用的字符编码是美国信息交换标准码，简称 ASCII 码（American Standard Code for Information Interchange）。ASCII 码用 7 位二进制码表示 128 种字符，编码规则如表 1.6 所示。由于数字系统中实际是用一个字节表示一个字符，所以使用 ASCII 码时，通常在最左边增加一位奇偶检验位。

表 1.6　7 位 ASCII 码编码表

低 4 位代码	高 3 位代码（$a_7a_6a_5$）							
（$a_4a_3a_2a_1$）	000	001	010	011	100	101	110	111
0000	NUL	DEL	SP	0	@	P	、	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{

续表

低4位代码 ($a_4a_3a_2a_1$)	高3位代码($a_7a_6a_5$)							
	000	001	010	011	100	101	110	111
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	−	o	DEL

注：NUL 空白 SOH 序始 STX 文始 ETX 文终 EOT 送毕 ENQ 询问
ACK 承认 BEL 告警 BS 退格 HT 横表 LF 换行 VT 纵表
FF 换页 CR 回车 SO 移出 SI 移入 DEL 转义 DC1 机控1
DC2 机控2 DC3 机控3 DC4 机控4 NAK 否认 SYN 同步 ETB 组终
CAN 作废 EM 载终 SUB 取代 ESC 扩展 FS 卷隙 GS 群隙
RS 录隙 US 元隙 SP 间隔 DEL 抹掉

习 题 一

1.1 什么是模拟信号？什么是数字信号？试各举一例。

1.2 数字逻辑电路具有哪些主要特点？

1.3 数字逻辑电路可分为哪两种类型？主要区别是什么？

1.4 最简电路是否一定最佳？为什么？

1.5 把下列不同进制数写成按权展开形式：

(1) $(4517.239)_{10}$ (2) $(10110.0101)_2$ (3) $(325.744)_8$ (4) $(785.4AF)_{16}$

1.6 将下列二进制数转换成十进制数、八进制数和十六进制数：

(1) 1110101 (2) 0.110101 (3) 10111.01

1.7 将下列十进制数转换成二进制数、八进制数和十六进制数(二进制数精确到小数点后4位)：

(1) 29 (2) 0.27 (3) 33.33

1.8 如何判断一个二进制正整数 $B=b_6b_5b_4b_3b_2b_1b_0$ 能否被$(4)_{10}$整除？

1.9 写出下列各数的原码、反码和补码：

(1) 0.1011 (2) −10110

1.10 已知$[N]_{补}=1.0110$，求$[N]_{原}$，$[N]_{反}$和N。

1.11 将下列余3码转换成十进制数和2421码：

(1) 011010000011 (2) 01000101.1001

1.12 试用8421码和格雷码分别表示下列各数：

(1) $(111110)_2$ (2) $(1100110)_2$