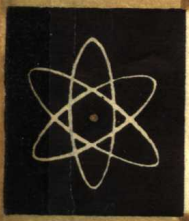
电子技术丛书



逻辑设计基础

逻辑设计基础

马绍汉 编著

山东科学技术出版社
一九八〇年·济南

内 容 简 介

本书系统地介绍了数字系统的逻辑设计。对数字系统数的表示、布尔代数、组合逻辑设计、时序逻辑设计、错误检测逻辑以及数字电子计算机算术运算的算法等内容，作了较详细地论述。可供从事数字系统设计、生产、维修的技术人员及高等院校、中专有关数字电子计算机专业师生参考。也可作为中学生课外读物。

电子技术丛书

逻辑设计基础

马绍汉 编著

*

山东科学技术出版社出版

山东省新华书店发行

山东新华印刷厂潍坊厂印刷

*

787×1092毫米 32开本　6.875印张　124千字

1980年3月第1版　1980年3月第1次印刷

印数：1—9,500

书号 15195·50　定价 0.57 元

出 版 说 明

电子技术是目前应用最广泛的先进技术之一，是衡量一个国家现代化水平的标志。为了普及电子技术知识，提高电子技术水平，早日实现四个现代化，我社决定编辑出版这套《电子技术丛书》。

这套丛书由山东大学《电子技术丛书》编委会主编。本丛书包括通讯、电视、微波技术、信息处理、电子器件及计算机等方面的内容，将分专题陆续编辑出版。本丛书力求做到理论联系实际、文字通俗易懂、内容简明扼要、切合实用，尽量方便广大读者阅读。

本丛书可供从事电子工业生产、科研的有关人员，高等院校、中专电子学专业师生及业余爱好者参考。

山东科学技术出版社

一九七九年六月

前　言

数字电子计算机及其它数字电子控制装置，统称为数字系统，在国民经济各部门中有着广泛的应用。数字系统的设计，包括总体设计、逻辑设计和电气设计三部分。逻辑设计包括开关理论、逻辑线路设计、集成电路系列、逻辑线路故障诊断和计算机辅助逻辑设计等内容。后两部分内容，是近年来逻辑设计的新发展，将逐步形成一个独立的学科。

本书从讨论数字电子计算机逻辑设计的实际问题入手，系统地介绍了数字系统的逻辑设计。为了适应逻辑设计的发展，书中对布尔差分及其在错误检测逻辑中的应用，也作了深入地讨论。可供从事数字系统设计、生产、维修的技术人员及高等院校、中专有关数字电子计算机专业师生参考。也可作为中学生课外读物。

本书在编写过程中，承蒙汪嘉业等同志热情帮助，在此表示感谢。

编　者

一九七九年五月

目　录

第一章　数字系统中数的表示

在日常生活中，人们习惯使用十进制计数方式，如10尺等于1丈，10角等于1元等。但在某些场合下，也使用非十进制计数方式，如计算时间用六十进制计数方式，60秒等于1分，60分等于1小时。

在数字电子计算机等数字系统中，要用物理元件的状态表示数。由于两种状态的物理元件容易实现，都可用0和1表示，所以用物理元件最便于表示二进制数。二进制计数系统本身具有运算简单、节省器材等优点，已广泛应用于数字系统中。有些数据处理计算机，有大量的十进制数输入、输出和进行运算。为了减少十进制数和二进制数的转换时间和转换中的误差，这种计算机常常直接对十进制数作各种运算，但十进制数字仍用二进制代码表示。

第一节　二进制数和八进制数

十进制数的每一位可以是0，1，2，…，9十个数字中的任一个。运算规则是"逢十进一，借一当十"。任何一个可用有限小数表示的十进制数 N，均可表示为：

$$N = q_n 10^n + q_{n-1} 10^{n-1} + \cdots + q_0 10^0 + q_{-1} 10^{-1} + \cdots + q_{-k} 10^{-k}$$

$$= \sum_{j=-k}^{n} q_j 10^j \qquad (1—1)$$

式中：$q_j = 0, 1, 2, \cdots, 9$。例如：

$$934.51 = 9\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1} + 1\times10^{-2}$$

十进制记数系统中的 10，称为这个记数系统的基数。

二进制数的每一位只能是 0，1 两个数字中的一个。运算规则是“逢二进一，借一当二”。其算术四则运算非常简单。例如：

加法：

$$\begin{array}{r} 10 \\ +\ 10 \\ \hline 100 \end{array} \qquad \begin{array}{r} 11 \\ +\ 11 \\ \hline 110 \end{array}$$

减法：

$$\begin{array}{r} 10 \\ -\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 100 \\ -\ 1 \\ \hline 11 \end{array}$$

乘法：

$$\begin{array}{r} 1011 \\ \times\ 1101 \\ \hline 1011 \\ 0000 \\ 1011 \\ 1011 \\ \hline 10001111 \end{array}$$

除法：

$$\begin{array}{r|l} & 1101 \\ \hline 1011 & 10001111 \\ & 1011 \\ \hline & 1101 \\ & 1011 \\ \hline & 1011 \\ & 1011 \\ \hline & 0000 \end{array}$$

根据二进制加法规则可知：

$$1 + 1 = (10)_2 = (2)_{10}$$

$$(10)_2 + (10)_2 = (100)_2 = (4)_{10} = (2^2)_{10}$$

$$(100)_2 + (100)_2 = (1000)_2 = (8)_{10} = (2^3)_{10}$$

式中：$(\ \ )_2$ 表示二进制数；$(\ \ )_{10}$ 表示十进制数。由此可知：

$(1)_2 = (2^0)_{10}$，$(10)_2 = (2^1)_{10}$，$(100)_2 = (2^2)_{10}$，$(1000)_2 = (2^3)_{10}$，…，$(0.1)_2 = (2^{-1})_{10}$，$(0.01)_2 = (2^{-2})_{10}$，…。任何

一个二进制数，都可以写成类似于式（1—1）的形式。例如：

$$10111 = 10000 + 100 + 10 + 1$$

$$= 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$$

$$10111.011 = 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1$$

$$+ 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3}$$

一般地说，任一具有有限位小数的二进制数 N，可表示为：

$$N = q_n 2^n + q_{n-1}2^{n-1} + \cdots + q_1 2^1 + q_0 2^0 + q_{-1}2^{-1} + q_{-2}2^{-2}$$

$$+ \cdots + q_{-k}2^{-k} = \sum_{j=-k}^{n} q_j 2^j \qquad (1—2)$$

式中：$q_j = 0$，1。

因此，在二进制记数系统中，其基数为2。式（1—1）和（1—2）是很类似的，如果在式（1—2）中用10来代替2，则两式就完全一样了。二进制小数每乘以一个2，相当于小数点向右移了一位；每除以一个2，相当于小数点向左移了一位。从这种意义上看，二进制记数系统中的基数2和十进制记数系统中的基数10作用是一样的。

一般来说，正整数 J 表示进位系统的基数，任一个 J 进制数 N，都可表示为：

$$N = \sum_{i=-k}^{n} q_i J^i \qquad (1—3)$$

式中：$q_i = 0,1,2,\cdots,(J-1)$；n 和 k 为正整数。每一个 q_i 都对应于同一个固定的值 J^i，J^i 称为 q_i 的权，相应的式（1--3）

称为 J 进制数按权展开式。

二进制数也有它的缺点，同一个数，如十进制数 23，在十进制记数系统中，只需两位数字表示，而在二进制记数系统中却要用五位数字（记为 10111）表示。也就是说，二进制数书写很不方便。为克服这个缺点，计算机上都采用八进制数或十六进制数来表示二进制数。由式（1—3）知，八进制数可表示为：

$$N = q_n 8^n + q_{n-1} 8^{n-1} + \cdots + q_1 8^1 + q_0 8^0 + q_{-1} 8^{-1} + \cdots + q_{-k} 8^{-k} = \sum_{i=-k}^{n} q_i 8^i \qquad (1—4)$$

式中：$q_i = 0,1,2,\cdots,7$。

二进制数可很方便地转换成八进制数，例如：

$$\begin{aligned}(1101101)_2 &= 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 \\ &\quad + 0\times 2^1 + 1\times 2^0 \\ &= 1\times 8^2 + (1\times 2^2 + 0\times 2^1 + 1\times 2^0)\times 8^1 \\ &\quad + (1\times 2^2 + 0\times 2^1 + 1\times 2^0)\times 8^0 \\ &= 1\times 8^2 + 5\times 8^1 + 5\times 8^0 = (155)_8\end{aligned}$$

由此可知，只需把二进制数从小数点向前每三位一分，便可把 1101101 分成“1”“101”“101”，然后把每三位二进制数化成八进制数。

对有小数的二进制数，小数点后面部分可从小数点开始向后每三位一分，然后每三位二进制数再化成八进制数。例如：

$$(11.10111)_2 = (3.56)_8$$

$$(10.1111011)_2=(2.754)_8$$

八进制数也很容易化成二进制数，只需把一个八进制数的每一位化成三位二进制数。例如：

$$(76.41)_8=(111110.100001)_2$$

$$(53.4)_8=(101011.1)_2$$

数字电子计算机工作者通常用八进制数来表示计算机中的二进制数或指令。

第二节　八进制数和十进制数之间的转换

一、八进制整数转换成十进制数

由式（1—4）可知，若N为八进制整数，则有：

$$N=\{\cdots\{[(q_n\cdot 8+q_{n-1})\cdot 8+q_{n-2}]\cdot 8+q_{n-3}\}\cdot 3+\cdots+q_1\}\cdot 8+q_0$$

式中：$q_i=0,1,2,\cdots,7\quad(i=0,1,2,\cdots,n)$。

只需依次算出：

$$\begin{aligned}
&q_n\times 8+q_{n-1}=M_{n-1}\\
&M_{n-1}\times 8+q_{n-2}=M_{n-2}\\
&M_{n-2}\times 8+q_{n-3}=M_{n-3}\\
&\cdots\cdots\\
&M_2\times 8+q_1=M_1\\
&M_1\times 8+q_0=(N)_{10}
\end{aligned}\qquad(1—5)$$

便可求出$(N)_{10}$。

【例 1】已知八进制数$(237)_8$，求其十进制数。

因为　$2\times 8+3=19$，　$19\times 8+7=159$

所以　$(237)_8=(159)_{10}$

二、十进制整数转换成八进制数

实际上式（1—5）已给出十进制整数转换成八进制数的算法。由式（1—5）最后一式可知，用 $(N)_{10}\div 8$ 得商 M_1，余数为 q_0；再用 $M_1\div 8$ 得商 M_2，余数为 q_1；…依次进行下去，最后可求得 q_0，q_1，q_2，…，q_{n-1}，q_n。于是便将十进制整数$(N)_{10}$转换成八进制数 $q_nq_{n-1}\cdots q_2q_1q_0$。

【例 2】求$(1895)_{10}$的八进制数。

$1895\div 8=236$……余数为 7，$M_1=236$，$q_0=7$

$236\div 8=29$……余数为 4，$M_2=29$，$q_1=4$

$29\div 8=3$……余数为 5，$M_3=3$，$q_2=5$

$3\div 8=0$ ……余数为 3，$M_4=0$，$q_3=3$

得　$(1895)_{10}=(3547)_8$。

三、八进制小数转换成十进制小数

由式（1—4）可知，若N为八进制小数，则有：

$$\begin{aligned}N&=q_{-1}\cdot 8^{-1}+q_{-2}\cdot 8^{-2}+\cdots+q_{-k}\cdot 8^{-k}\\&=\{\{\cdots\{[(q_{-k}\cdot 8^{-1}+q_{-k+1})\cdot 8^{-1}+q_{-k+2}]\cdot 8^{-1}\\&\quad+q_{-k+3}\}\cdot 8^{-1}+\cdots+q_{-2}\}\cdot 8^{-1}+q_{-1}\}\cdot 8^{-1}\end{aligned}$$

只需依次算出：

$$\begin{aligned}&q_{-k}\times 8^{-1}+q_{-k+1}=S_{-k+1}\\&S_{-k+1}\times 8^{-1}+q_{-k+2}=S_{-k+2}\\&S_{-k+2}\times 8^{-1}+q_{-k+3}=S_{-k+3}\qquad(1—6)\\&\cdots\cdots\end{aligned}$$

$$S_{-3}\times 8^{-1}+q_{-2}=S_{-2}$$

$$(S_{-2}\times 8^{-1}+q_{-1})\times 8^{-1}=(N)_{10}$$

便可求出$(N)_{10}$。

【例 3】求$(0.645)_8$的十进制数。

因为　$5\times 0.125+4=4.625$,

$$(4.625\times 0.125+6)\times 0.125=0.822265625$$

所以　$(0.645)_8=(0.822265625)_{10}$

四、十进制小数转换成八进制小数

由式(1—6)可找出十进制小数转换成八进制小数的方法。这个方法即已知$(N)_{10}$，求q_{-1}，q_{-2}，…，q_{-k+1}，q_{-k}。

因为　$0\leqslant q_i\leqslant 7$　$(i=-1,-2,\cdots,-k)$，由式（1—6）不难证明：$0\leqslant S_i<8$　$(i=-k+1,-k+2,\cdots,-2)$。

由式（1—6）最后一式可知，$(N)_{10}\times 8=S_{-2}\times 8^{-1}+q_{-1}$。由于$0\leqslant S_{-2}\times 8^{-1}<1$，所以$q_{-1}$是$(N)_{10}\times 8$的整数部分；而$S_{-2}\times 8^{-1}$是$(N)_{10}\times 8$的小数部分。再把小数部分乘 8，得：

$$S_{-2}=S_{-3}\times 8^{-1}+q_{-2}$$

S_{-2}的整数部分为q_{-2}；小数部分为$S_{-3}\times 8^{-1}$。再把小数部分乘 8，得：

$$S_{-3}=S_{-4}\times 8^{-1}+q_{-3}$$

又可求出S_{-3}的整数部分为q_{-3}；小数部分为$S_{-4}\times 8^{-1}$。依次进行下去，可求得q_{-4}，q_{-5}，…。得　$(N)_8=0\cdot q_{-1}q_{-2}\cdots$。

【例 4】求$(0.95)_{10}$的八进制数。

因为　$0.95\times 8=7.6$，　$S_{-2}\times 8^{-1}=0.6$，　$q_{-1}=7$

$$0.6\times 8=4.8,\quad S_{-3}\times 8^{-1}=0.8,\quad q_{-2}=4$$

$$0.8\times 8=6.4,\quad S_{-4}\times 8^{-1}=0.4,\quad q_{-3}=6$$

$$0.4\times 8=3.2,\quad S_{-5}\times 8^{-1}=0.2,\quad q_{-4}=3$$

所以 $(0.95)_{10}=(0.7463\cdots)_8$

二进制数与十进制数之间的转换，既可用八进制数作为中间转换过渡，也可用类似于八进制数与十进制数的转换直接求出。国内外有的计算机也常用十六进制数表示二进制数，十六进制数每位数字可以是0，1，2，3，4，5，6，7，8，9，*A*，*B*，*C*，*D*，*E*，*F*十六个字中的一个。此处*A*，*B*，*C*，*D*，*E*，*F*分别表示10，11，12，13，14，15。十六进制数与十进制数之间的转换方法，读者可自行推出。

第三节　数的定点和浮点表示

我们知道，对十进制数，如67.819，可表示为：$67.819=10^2\times 0.67819$；数0.000653可表示为：$0.000653=10^{-3}\times 0.653$。对二进制数，如10011.101，也可表示为

$$10011.101=2^{101}\times 0.10011101$$

一般来说，对任何一个二进制数N可表示为：

$$N=2^j\times S \qquad (1—7)$$

式中：j为二进制整数；S为二进制小数。j称为数N的阶码，而S称为数N的尾数。当N为式（1—7）的形式，且$1>|S|\geqslant\frac{1}{2}$时，则$2^j\times S$称为规格化数*。对规格化数而言，

* 这个定义也随计算机不同而异，有的计算机要求把数表示成$N=4^j\times S$，此时规格化数要求$1>|S|\geqslant 1/4$。

阶码 j 实际上指定了小数点的位置。例如：

$$2^1\times 0.1011=1.011,\quad 2^{10}\times 0.1011=10.11$$

在计算机中，一般把数表示为式（1—7）的形式，如果 j 可以变化，则这样的数称为浮点数。能进行浮点数运算的计算机，称为浮点机。为了充分利用机器的有效位，提高运算精度，浮点机内部存放的浮点数要求是规格化数。如果在计算机中，j 只能取某一固定值，则这种数称为定点数。能进行定点数运算的计算机，称为定点机。定点机一般取 $j=0$，也就是定点机中的数只能表示 $|N|<1$ 的数。定点数不分规格化数和非规格化数。也有的计算机同时具有定点运算和浮点运算的功能。

浮点运算比定点运算复杂。设两浮点数：

$$N_1=2^{j_1}\times S_1,\quad N_2=2^{j_2}\times S_2$$

$$N_1+N_2=2^{j_1}\times S_1+2^{j_2}\times S_2$$

若 $j_1=j_2$，则：　$N_1+N_2=2^{j_1}(S_1+S_2)$

也就是阶码相等的两数相加，阶码保持不变，只要尾数相加就可以了。若两数的阶码不相等，它们相加时，则要把它们的阶码变成相等后才能相加。在计算机中，把它们的阶码变成相等的工作，称为对阶。对阶时，要求阶码小的向阶码大的看齐（详见第五章）。例如，$N_1=2^{11}\times 0.1001$，$N_2=2^{01}\times 0.1100$

$$N_2=2^{01}\times 2^{10}\times 2^{-10}\times 0.1100=2^{11}\times 0.0011$$

$$N_1+N_2=2^{11}\times(0.1001+0.0011)=2^{11}\times 0.1100$$

两数相减时，同样要进行对阶

$$N_1 - N_2 = 2^{11} \times 0.1001 - 2^{01} \times 0.1100$$
$$= 2^{11} \times 0.1001 - 2^{11} \times 0.0011$$
$$= 2^{11} \times (0.1001 - 0.0011) = 2^{11} \times 0.0110$$
$$= 2^{10} \times 0.1100$$

两浮点数相乘时，阶码相加，尾数相乘；两浮点数相除时，阶码相减，尾数相除。即：

$$N_1 \times N_2 = (2^{j_1} \times S_1) \times (2^{j_2} \times S_2) = 2^{j_1+j_2} \times (S_1 \times S_2)$$

$$N_1 \div N_2 = (2^{j_1} \times S_1) \div (2^{j_2} \times S_2) = 2^{j_1-j_2} \times (S_1 \div S_2)$$

定点数的运算不存在对阶问题，因为定点数的阶码都等于某一个固定的常数（一般为零）。因此，定点运算比浮点运算简单，相应的定点机的运算、控制部件的结构也简单些。但它表示数的范围小，如阶码都取零的定点数，仅能表示绝对值小于 1 的数。因此，定点机在算题时，要求一切数据和中间结果的绝对值均小于 1，这样给使用者带来不少麻烦，而浮点机却能避免这种麻烦。计算机究竟采用定点还是浮点，应根据服务对象而定。一般科技计算用的计算机，都采用浮点表示数；小型专用机或控制机，则采用定点表示数。

第四节　二——十进制数

数字电子计算机用二进制数进行运算，为了使用方便，输入输出的数据通常使用十进制形式。为使十进制数能存放在计算机中，通常采用二——十进制数表示十进制数。这种编码方式也称为 BCD 码，它是用四位二进制数表示一位十进制

数，其对应规律见表 1—1。

表 1—1　　二——十进制数(BCD 码)

十进制	二——十进制	十进制	二——十进制
0	0 0 0 0	5	0 1 0 1
1	0 0 0 1	6	0 1 1 0
2	0 0 1 0	7	0 1 1 1
3	0 0 1 1	8	1 0 0 0
4	0 1 0 0	9	1 0 0 1

这个四位二进制数的每一位，分别对应着十进制常数 8，4，2，1。这种编码方式，称为 8421 编码。例如，十进制数 427 的二——十进制数为 0100　0010　0111；反之，二——十进制数 0011　1001　0110 表示十进制数 396。

计算机在输入输出过程中用二——十进制数，其内部用二进制数运算。因此，有一个二——十进制数和二进制数之间的转换问题，这可由计算机自身来完成。下面具体讨论这种转换的算法。

一、二——十进制数转换成二进制数

设一个十进制数为 $0.a_1a_2a_3\cdots a_n$，当将它写成二——十进制数形式，输入到计算机内部存放时，对计算机而言，它是一个二进制数，其值为：

$$\frac{a_1}{16}+\frac{a_2}{16^2}+\frac{a_3}{16^3}+\cdots+\frac{a_n}{16^n} \qquad (1—8)$$

但这个数真实的值是：