高等学校
试用教材

数字逻辑

王玉龙 编

高等教育出版社

高等学校试用教材

数字逻辑

王玉龙编

高等教育出版社

内 容 提 要

本书系统地介绍了数字电路逻辑设计的基本理论和方法。全书共十章。第一二两章介绍了数字设计的基本理论，第三至第七章系统地介绍了小规模集成电路的数字设计，这一部分既是本书的重点，也是学习其它数字设计方法的基础。最后三章则分别介绍了采用中、大规模集成电路和微处理器进行的数字设计，以及用计算机辅助设计逻辑线路的基本概念和方法。

本书是在作者使用多年的"数字逻辑"讲义的基础上，增加了某些有关数字设计的新技术编写而成的。取材较新，逻辑性较强。在讲述方法上，注意了知识的循序渐进，叙述清楚易懂。为便于读者领会所讲内容，书中给出了大量例题，每章后还附有一定数量的练习题。

本书可作为高等学校有关专业的教材或参考书，也可供科技人员参考。

高等学校试用教材

数 字 逻 辑

王 玉 龙 编

高等教育出版社出版

新华书店北京发行所发行

北 京 印 刷 三 厂 印 装

开本 787×1092 1/16 印张 28 字数 642000

19 87 年 5 月第 1 版 19 37 年 5 月第 1 次印刷

印数 00 001—10 130

书号 13010·01380 定价 4.25 元

前　言

“数字逻辑”是“数字电路逻辑设计”的简称，其内容是讲述应用数字电路进行数字系统逻辑设计(简称数字设计)的方法。随着微电子技术的迅速发展和微处理器的广泛应用，数字设计出现了两种截然不同的方式：硬接线逻辑和程序逻辑，并可用下列五种方法实现：

1. 应用小规模集成电路(SSI)的门和触发器。
2. 应用中、大规模集成电路(MSI 和 LSI)的功能块。
3. 应用中、大规模集成电路的只读存储器(ROM)。
4. 应用大规模集成电路的可编程序逻辑阵列(PLA)。
5. 应用微处理器的程序。

本书是以原教育部制订的计算机专业的“数字逻辑”教学大纲为基础，并考虑到上述数字设计的发展方向编写而成的。全书包括下列六部分：数字设计的基础理论(第一、二章)，组合线路的分析与设计(第三、四章)，时序线路的分析与设计(第五～七章)，采用中大规模集成电路进行的数字设计(第八章)，用微处理器的程序实现的数字设计(第九章)及计算机辅助逻辑设计(第十章)。全书用了五章(第三～七章)的篇幅讲述小规模集成电路的数字设计，其目的在于为学习其它几种数字设计方法打下坚实的基础。第八章集中介绍了用中大规模集成电路实现数字设计的三种方法。第九章则以 8080 微处理器的指令系统为背景，介绍了用程序逻辑实现数字设计的方法。为便于读者理解，也为了作为数字系统的一个实例，本章的开始部分还简要介绍了微计算机的基本组成及其工作原理。第十章介绍了用计算机辅助设计逻辑线路的基本概念和方法，以便使读者对数字设计的这一新手段有一个初步了解。

作为一本教材或教学参考书，除尽可能地反映这一领域里的最新技术外，还要着眼于培养学生分析问题和解决问题的能力。为此，本书作了如下考虑：在内容组织上，以讲述“方法”为重点，力求为学生提供独立分析和设计逻辑线路的“工具”，而不是向学生灌输各种各样的逻辑线路；在讲述方法上，力求作到一个问题的提出，着重于思路的分析，尽量避免就事论事，以使学生了解其来龙去脉，从而启发学生独立解决问题的能力；在文字叙述上，力求作到叙述流畅，说理清楚，以便于学生自学。当然，上述这些考虑是否能够真正实现，还有待于教学实践的检验。

在选用本书作为“数字逻辑”课的教材或教学参考书时，或许会遇到内容众多与学时甚少的矛盾。我们建议可根据教学计划的要求，删去某些章节。如目录中带 * 号的部分。当然，也可以把这些部分作为选学的内容。我们编写本书的一个指导思想是尽可能保持“数字逻辑”在内容上的完整性，并为不同程度的学生提供因材施教的条件。

在本书编写过程中，曾得到北京航空学院计算机科学和工程系的陈望梅和杨文龙两位教授的大力支持和帮助。在本书的最后整理过程中又得到邵洪余等同志的帮助，在此，向他们表示衷心的感谢。

本书虽是在多次使用的“数字逻辑”讲义的基础上改编而成，但由于笔者水平所限，书中仍难免有错误和不妥之处，敬请读者批评指正。

1986 年 1 月于北方工业大学计算机系

目　录

第一章　数制及编码

通常，一个数值可用两种不同的方法表示：即按“值”表示和按“形”表示。按“值”表示要求在选定的进位制中表示出正确的数值，如对于数值“正十五点三”和“负四点八”，在十进制中可分别表示为＋15.3 和 －4.8。按“形”表示则是按照一定的编码方法形式地表示出数值，如用ASCII码表示数值“正十五点三”，其形式为

2B　31　35　2E　33

由上例不难看出，采用按“值”表示的方法时，需要解决三个问题：一是恰当地选用“数字符号”（或称数码）及其组合规则；二是正确地给出小数点的位置；三是正确地表示出数的正、负符号。本章的前几节将分别讨论这三个问题在计算机中是怎样解决的。采用按“形”表示时，先要确定编码的规则，然后按此规则编出所需的代码。本章的最后一节将介绍几种计算机中常用的编码。

1.1　进位计数制

要表示一个数，首先要选择适当的数字符号并规定其组合规律，也就是确定所选用的进位计数制。所谓进位计数制，顾名思义，就是一种按进位方式实现计数的制度，简称进位制。常用的进位制有十进制、二进制或八进制等。在十进制中，任何数都是用十个数字符号(0,1,2,3,4,5,6,7,8,9)按逢十进一的规则组成的；在二进制中，任何数都是用两个数字符号(0, 1)按逢二进一的规则组成的；在八进制中，任何数都是用八个数字符号(0,1,2,3,4,5,6,7)按逢八进一的规则组成的。尽管这些进位制所采用的数字符号及其进位规则不同，但有一个共同的特点，即数是按进位的方式计量的。表 1.1 列出了十进制、二进制及 R 进制的特点，及根据这些特点组成的表示式。

不难看出，每种进位制都有一个基本特征数，被称为进位制的“基数”，基数表示了进位制所具有的数字符号的个数及进位的规则。显然，十进制的基数为“十”，二进制的基数为“二”，R 进制的基数为“R”，它们在各自的进位制中都表示为“10”（见表 1.1），读作“么零”。

在同一个进位制中，不同位置上的同一个数字符号所代表的值是不同的。例如：

	1	1	1	1	·	1	1
	↓	↓	↓	↓		↓	↓
十进制中	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}
二进制中	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}

为了描述进位制数的这一性质，定义某一进位制数中各位“1”所表示的值为该位的“权”，或称“位权”。

基数和权是进位制的两个要素，正确地理解了它们的含义，便可掌握进位制的全部内容。例如，根据基数和权的概念，可列出不同进位制数的相互关系，如表 1.2 所示；还可列出二进制数各位的权，如表 1.3 所示。

表 1.1 进位制数

	十 进 制	二 进 制	R 进 制
特点	1. 具有十个数字符号0，1，2，…，9。 2. 由低位向高位是逢十进一。	1. 具有两个数字符号0，1。 2. 由低位向高位是逢二进一。	1. 具有R个数字符号0，1，…，$(R-1)$。 2. 由低位向高位是逢R进一。
举例	$1982.31=1\times10^3+9\times10^2+8\times10^1+2\times10^0+3\times10^{-1}+1\times10^{-2}$	$1011.01=1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$	设$R=8$(八进制) $1075.42=1\times8^3+0\times8^2+7\times8^1+5\times8^0+4\times8^{-1}+2\times8^{-2}$
一般式	$S=(K_{n-1}K_{n-2}\cdots K_0K_{-1}K_{-2}\cdots K_{-m})_{10}=\sum_{i=n-1}^{-m}K_i(10)_{10}^i$ 式中，m,n为正整数；$K_i=0,1,2,\cdots9$。	$S=(K_{n-1}K_{n-2}\cdots K_0K_{-1}K_{-2}\cdots K_{-m})_2=\sum_{i=n-1}^{-m}K_i(2)_{10}^i$ 式中，m,n为正整数，$K_i=0,1$。	$S=(K_{n-1}K_{n-2}\cdots K_0K_{-1}K_{-2}\cdots K_{-m})_R=\sum_{i=n-1}^{-m}K_i(R)_{10}^i$ 式中，m,n为正整数，$K_i=0,1,2,\cdots,(R-1)$。
基数	$(10)_{10}$	$(10)_2=(2)_{10}$	$(10)_R=(R)_{10}$
第 i 位的权	$(10)_{10}^i$	$(10)_2^i=(2)_{10}^i$	$(10)_R^i=(R)_{10}^i$

表 1.2 不同基数的进位制数

数值	$R=10$	$R=2$	$R=3$	$R=4$	$R=8$	$R=16$
零	0	0	0	0	0	0
一	1	1	1	1	1	1
二	2	10	2	2	2	2
三	3	11	10	3	3	3
四	4	100	11	10	4	4
五	5	101	12	11	5	5
六	6	110	20	12	6	6
七	7	111	21	13	7	7
八	8	1000	22	20	10	8
九	9	1001	100	21	11	9
十	10	1010	101	22	12	A
十一	11	1011	102	23	13	B
十二	12	1100	110	30	14	C
十三	13	1101	111	31	15	D
十四	14	1110	112	32	16	E
十五	15	1111	120	33	17	F
十六	16	10000	121	100	20	10
十七	17	10001	122	101	21	11
十八	18	10010	200	102	22	12
十九	19	10011	201	103	23	13
二十	20	10100	202	110	24	14

表 1.3 二进制数各位的权

第 i 位	权 2^i	对应二进制数	第 i 位	权 2^i	对应二进制数
0	$2^0=1$	1			
1	$2^1=2$	10	-1	$2^{-1}=0.5$	0.1
2	$2^2=4$	100	-2	$2^{-2}=0.25$	0.01
3	$2^3=8$	1000	-3	$2^{-3}=0.125$	0.001
4	$2^4=16$	10000	-4	$2^{-4}=0.0625$	0.0001
5	$2^5=32$	100000	-5	$2^{-5}=0.03125$	0.00001
6	$2^6=64$	1000000	-6	$2^{-6}=0.015625$	0.000001
7	$2^7=128$	10000000	-7	$2^{-7}=0.0078125$	0.0000001
⋮	⋮	⋮	⋮	⋮	⋮
$n-1$	2^{n-1}	$\underbrace{10\cdots\cdots0}_{n-1}$	$-m$	2^{-m}	$\underbrace{0.0\cdots\cdots01}_{m-1}$

由上可知，任何进位制数都可表示为两种形式。例如，在十进制中，数值“一千九百八十二点三二”可表示为

$$1982.32$$

或

$$1\times10^3+9\times10^2+8\times10^1+2\times10^0+3\times10^{-1}+2\times10^{-2}$$

一般地，在R进制中可有

$$\begin{aligned}(S)_R&=(K_{n-1}K_{n-2}\cdots K_0K_{-1}K_{-2}\cdots K_{-m})_R \\ &=K_{n-1}R^{n-1}+K_{n-2}R^{n-2}+\cdots+K_0R^0+K_{-1}R^{-1}+K_{-2}R^{-2} \\ &\quad+\cdots+K_{-m}R^{-m}\end{aligned} \tag{1.1}$$

$$=\sum_{i=n-1}^{-m}K_iR^i \tag{1.2}$$

式中，n表示整数的位数，m表示小数的位数，$K_i=0,1,\cdots,(R-1)$中的任何一个。

通常，我们把式(1.1)称为并列表示法或位权记数法，把式(1.2)称为多项式表示法或按权展开式。

1.2 进位制数的相互转换

由前一节可知，同一个数值可以用不同进位制数表示，其形式是不同的。例如，数值“十一”，由表 1.2 可知，它可表示为

$$\begin{aligned}(11)_{10}&=(1011)_2=(102)_3 \\ &=(23)_4=(13)_8 \\ &=(B)_{16}\end{aligned}$$

式中，圆括号外的下标表示进位制，这就是说，不同进位制只是描述数值的不同“手段”，因而它们可以相互转换，转换的前提是保证转换前后所表示的数值相等。

那么，怎样实现进位制数的相互转换呢？下面介绍四种常用算法，并说明怎样保证转换精度。

1.2.1 算法 1：多项式替代法

在介绍这种算法的规则之前，我们先来看一个简单例子。

例 将二进制数 1011.101 转换为十进制数。

即 $$(1011.101)_2=(?)_{10}$$

将二进制数的并列表示法转换为多项式表示法，则得

$$(1011.101)_2=[1\times(10)^{11}+0\times(10)^{10}+1\times(10)^{1}+1\times(10)^{0}+1\times(10)^{-1}+0\times(10)^{-10}+1\times(10)^{-11}]_2$$

将等式右边的所有二进制数转换为等值的十进制数，则得

$$(1011.101)_2=[1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}]_{10}$$

在十进制中计算等式右边之值，则得

$$(1011.101)_2=[8+2+1+0.5+0.125]_{10}$$
$$=(11.625)_{10}$$

可见，将二进制数转换为十进制数的方法是，将二进制数的各位在十进制中按权展开相加。这一方法可推广到任何两个 α,β 进制数之间的转换，其算法如图 1.1 所示。

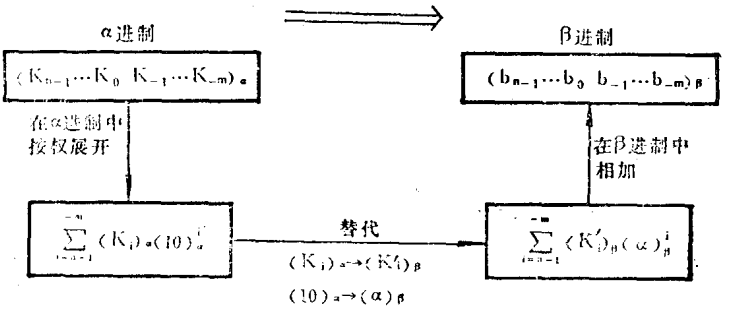


图 1.1 多项式替代法的算法示意图

1.2.2 算法 2：基数除/乘法

基数除/乘法实际包括基数除法及基数乘法两种，前者适用于整数转换，后者适用于小数转换。下面分别介绍这两种方法，我们都从简单的例子讲起，再引入算法的一般规则。

一、基数除法

引例 将十进制整数 92 转换为二进制数，即 $(92)_{10}=(?)_2$

设转换结果为

$$(92)_{10}=(b_{n-1}b_{n-2}\cdots b_0)_2$$
$$=[b_{n-1}2^{n-1}+b_{n-2}2^{n-2}+\cdots+b_02^0]_{10}$$

在十进制中计算该式，两边除以 2，则得

$$\frac{92}{2}=\frac{1}{2}(b_{n-1}2^{n-1}+b_{n-2}2^{n-2}+\cdots+b_0 2^0)$$

$$46=\underbrace{(b_{n-1}2^{n-2}+b_{n-2}2^{n-3}+\cdots+b_1 2^0)}_{\text{整数}}+\underbrace{\frac{b_0}{2}}_{\text{分数}}$$

两数相等，则其整数与分数部分必分别相等，故有

$$46=b_{n-1}2^{n-2}+b_{n-2}2^{n-3}+\cdots+b_1 2^0 \tag{1.3}$$

$$0=\frac{b_0}{2} \qquad b_0=0$$

将式(1.3)两边分别除以2，可得

$$\frac{46}{2}=\frac{1}{2}(b_{n-1}2^{n-2}+b_{n-2}2^{n-3}+\cdots+b_1 2^0)$$

$$23=\underbrace{(b_{n-1}2^{n-3}+b_{n-2}2^{n-4}+\cdots+b_2 2^0)}_{\text{整数}}+\underbrace{\frac{b_1}{2}}_{\text{分数}}$$

故

$$23=b_{n-1}2^{n-3}+b_{n-2}2^{n-4}+\cdots+b_2 2^0$$

$$0=\frac{b_1}{2} \qquad b_1=0$$

可见，所要求的二进制数$(b_{n-1}b_{n-2}\cdots b_1 b_0)_2$的最低位$b_0$是十进制数92除2所得的余数。次低位$b_1$是所得的商46再除以2得到的余数，依此类推，继续用2除，直除到商为0时止，各次所得的余数即为要求的二进制数的b_2到b_{n-1}之值。整个计算过程如下：

```
2 | 9 2                          余数
  2 | 4 6 ·····················0   ↑ b0(低位)
    2 | 2 3 ···················0   |
      2 | 1 1 ·····  ····1       |
        2 | 5 ····· · ·····1     |
          2 | 2 ·· ·· ·····1     |
            2 | 1 ····· ··0      |
商为0止→      0 ·····  ·····1    ↓ b6(高位)
```

转换结果为$(92)_{10}=(1011100)_2$

上述将十进制整数转换为二进制整数的方法可推广到任何两个α,β进制数之间的转换，其算法如图1.2所示。

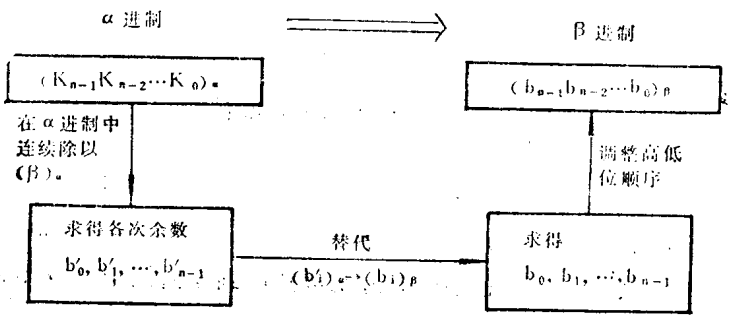


图1.2 基数除法的算法示意图

二、基数乘法

引例 将十进制小数 0.6875 转换为二进制数，即 $(0.6875)_{10}=(?)_2$

设转换结果为

$$(0.6875)_{10}=(0.b_{-1}b_{-2}\cdots b_{-m})_2$$
$$=(b_{-1}2^{-1}+b_{-2}2^{-2}+\cdots+b_{-m}2^{-m})_{10}$$

在十进制中计算该式，两边乘以 2，则得

$$0.6875\times 2=b_{-1}+(b_{-2}2^{-1}+b_{-3}2^{-2}+\cdots+b_{-m}2^{-m+1})$$
$$1.3750=\underbrace{b_{-1}}_{\text{整数}}+\underbrace{(b_{-2}2^{-1}+b_{-3}2^{-2}+\cdots+b_{-m}2^{-m+1})}_{\text{小数}}$$

两数相等，则其整数部分与小数部分必分别相等，故有

$$b_{-1}=1$$
$$0.3750=b_{-2}2^{-1}+b_{-3}2^{-2}\cdots+b_{-m}2^{-m+1} \tag{1.4}$$

将式(1.4)两边再分别乘以 2，可得

$$0.3750\times 2=b_{-2}+(b_{-3}2^{-1}+b_{-4}2^{-2}+\cdots+b_{-m}2^{-m+2})$$
$$0.75=b_{-2}+(b_{-3}2^{-1}+b_{-4}2^{-2}+\cdots+b_{-m}2^{-m+2})$$

故

$$b_{-2}=0$$
$$0.75=b_{-3}2^{-1}+b_{-4}2^{-2}+\cdots+b_{-m}2^{-m+2}$$

可见，要求的二进制数 $(0.b_{-1}b_{-2}\cdots b_{-m})_2$ 的最高位 b_{-1} 是十进制数 0.6875 乘 2 所得的整数部分，其小数部分再乘以 2 所得的整数部分即为 b_{-2} 之值。依此类推，继续用 2 乘，每次所得之乘积的整数部分就是要求的二进制数的 b_{-3} 至 b_{-m} 之值。整个计算过程如下：

0.6875		
× 2	整数	
1.3750……	1	b_{-1}(高位)
0.3750		
× 2		
0.7500……	0	
0.7500		
× 2		
1.5000……	1	
0.5000		
× 2		
1.0000……	1	↓b_{-4}(低位)

乘积为 0 结束→ 0.0000

故转换结果为 $(0.6875)_{10}=(0.1011)_2$

上述将十进制小数转换为二进制小数的方法可推广到任何两个 α,β 进制数之间的转换，其算法如图 1.3 所示。

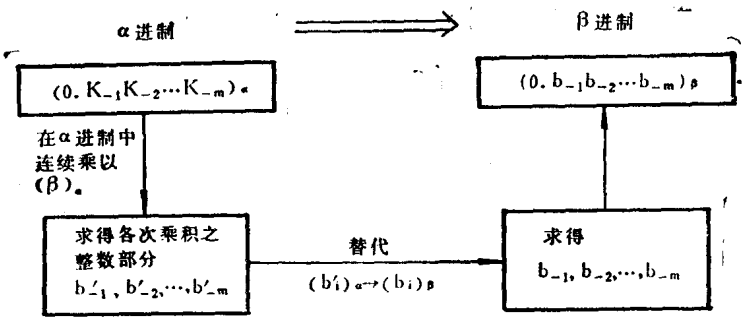


图 1.3　基数乘法的算法示意图

三、小结

由前讨论可知，基数除/乘法适用于任何 α,β进制数之间的转换，而转换过程的主要计算在 α 进制中进行。采用基数除/乘法将 α 进制数转换为 β 进制数时，整数部分用“除基取余”，即把要转换的 α 进制整数连续用 β 进制的基数(β_α)除，各次所得的余数即为所求的 β 进制数由低位到高位之值；小数部分则用“乘基取整”，即把要转换的 α 进制小数连续用 β 进制的基数(β_α)乘，各次所得的乘积的整数即为所求的 β 进制数由高位到低位之值。转换所得的 β 进制数的位数是这样确定的：整数的位数由商等于 0 确定，小数的位数或由乘积等于 0 或由所需精度确定。怎样根据精度要求确定小数的位数，我们将在本节的最后介绍。

下面，再举两个例子说明基数除/乘法的应用。

例 1　将十进制数 1025.25 转换为二进制数。

计算过程如下：

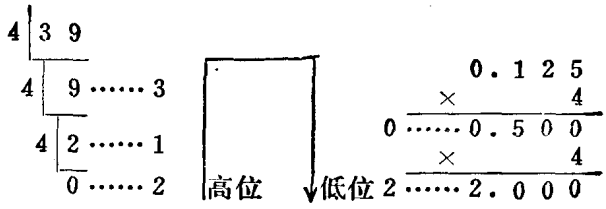
$$
\begin{array}{r|l}
2 & 1025 \\
2 & 512 \cdots\cdots 1 \\
2 & 256 \cdots\cdots 0 \\
2 & 128 \cdots\cdots 0 \\
2 & 64 \cdots\cdots 0 \\
2 & 32 \cdots\cdots 0 \\
2 & 16 \cdots\cdots 0 \\
2 & 8 \cdots\cdots 0 \\
2 & 4 \cdots\cdots 0 \\
2 & 2 \cdots\cdots 0 \\
2 & 1 \cdots\cdots 0 \\
 & 0 \cdots\cdots 1
\end{array}
\quad \text{低位} \quad \text{高位}
$$

$$
\begin{array}{r}
0.25 \\
\times \quad 2 \\
\hline
0 \cdots\cdots 0.50 \\
\times \quad 2 \\
\hline
1 \cdots\cdots 1.00
\end{array}
$$

转换结果为$(1025.25)_{10}=(10000000001.01)_2$。

例 2 将十进制数 39.125 转换为四进制数。

计算过程如下：



转换结果为$(39.125)_{10}=(213.02)_4$。

1.2.3 算法 3：混合法

前面介绍的两个算法虽然都适用于任何 α、β 进制数之间的转换，但转换过程的计算却是在不同进制中进行的。多项式替代法是在 β 进制中进行计算，基数除/乘法是在 α 进制中进行计算。由于人们非常熟悉十进制，因而都设法使进位制数之间的转换在十进制中进行计算，这就是二进制数到十进制数的转换都采用多项式替代法，而十进制数到二进制数的转换都采用基数除/乘法的原因。

如果 α、β 进制都不是人们所熟悉的进位制，则采用前述两种算法进行数制转换就很不方便。此时，我们可以采用混合法，即先把 α 进制数转换为十进制数；再把十进制数转换为 β 进制数，从而使 α 到 β 进制的转换过程的计算都在十进制中进行。所谓混合法，就是多项式替代法和基数除/乘法的综合，其规则是

1. 用多项式替代法将 α 进制数 $(S)_\alpha$ 转换为十进制数 $(S)_{10}$。
2. 用基数除/乘法将十进制数 $(S)_{10}$ 转换为 β 进制数 $(S)_\beta$。

该算法如图 1.4 所示。

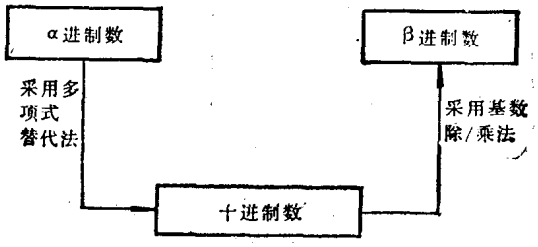


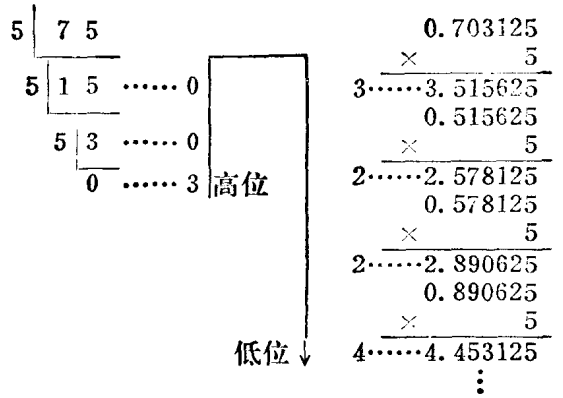
图 1.4 混合法的算法示意图

例 将四进制数 1023.231 转换为五进制数。

用多项式替代法，将四进制数转换为十进制数：

$$\begin{aligned}(1023.231)_4&=(1\times4^3+0\times4^2+2\times4^1+3\times4^0+2\times4^{-1}+3\times4^{-2}+1\times4^{-3})_{10}\\&=(64+0+8+3+0.5+0.1875+0.015625)_{10}\\&=(75.703125)_{10}\end{aligned}$$

用基数除/乘法，将所得的十进制转换为五进制数：



转换结果为

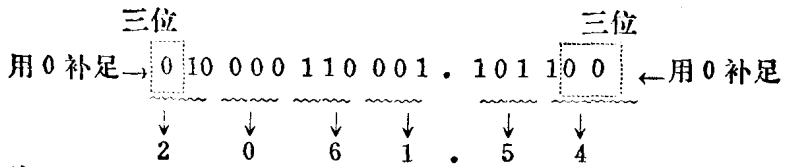
$$(1023.231)_4=(300.3224)_5\ (\text{取 4 位小数})$$

1.2.4 算法 4: 直接转换法

当数 S 由 α 进制转换为 β 进制时，如果 α 与 β 进制的基数满足 2^k（k 为整数）关系，那么采用直接转换法比采用上述任一算法要简单得多。下面，先举一个简单例子，说明直接转换法的基本规则。

引例 将二进制数 10000110001.1011 转换为八进制数。

该例中，$\alpha=2,\beta=8$. 显然，$2^3=8$ 即 $\alpha^k=\beta,k=3$. 故可按下列方法直接把二进制数转换成八进制数：



转换结果为

$$(10000110001.1011)_2=(2061.54)_8$$

转换所用的规则是，以小数点为界，将二进制数的整数部分由低位到高位，小数部分由高位到低位，分成三位一组，头尾不足三位的补 0，然后将每组的三位二进制数转换为一位八进制数。

据此规则，我们很容易把八进制数转换为二进制数。例如，可用下列方法将八进制数 1037.26 直接转换为二进制数：

1	0	3	7	.	2	6	（八进制）
↓	↓	↓	↓		↓	↓	
001	000	011	111		010	110	（二进制）

即 $(1037.26)_8=(1000011111.01011)_2$

现在，以整数为例证明上述二、八进制数之间的直接转换关系。设整数 S 的二进制多项式表示为

$$(S)_2=a_{n-1}2^{n-1}+\cdots+a_3 2^3+a_2 2^2+a_1 2^1+a_0 2^0$$

整数 S 的八进制多项式表示为

$$(S)_8=b_{m-1}8^{m-1}+\cdots+b_3 8^3+b_2 8^2+b_1 8^1+b_0 8^0$$

令 $(S)_2=(S)_8$，则得

$$a_{n-1}2^{n-1}+\cdots+a_3 2^3+a_2 2^2+a_1 2^1+a_0 2^0$$
$$=b_{m-1}8^{m-1}+\cdots+b_3 8^3+b_2 8^2+b_1 8^1+b_0 8^0$$

两边除以 8(即 2^3)，得

$$(a_{n-1}2^{n-4}+\cdots+a_3 2^0)+\frac{1}{8}(a_2 2^2+a_1 2^1+a_0 2^0)$$
$$=(b_{m-1}8^{m-2}+\cdots+b_2 8^1+b_1 8^0)+\frac{1}{8}b_0$$

两数相等，则其整数部分与分数部分必分别相等，故得

$$\frac{1}{8}(a_2 2^2+a_1 2^1+a_0 2^0)=\frac{1}{8}b_0$$
$$a_2 2^2+a_1 2^1+a_0 2^0=b_0$$

即

$$(a_2\ a_1\ a_0)_2=(b_0)_8$$

同理，根据整数部分相等，两边再除以 8，便可求得

$$(a_5\ a_4\ a_3)_2=(b_1)_8$$

依此类推，便证明了二、八进制数之间的直接转换规则。这一规则可推广到任何基数满足 $\alpha^k=\beta$ (或 $\alpha=\beta^k$)的两个进位制之间的转换，如下所述：

1. 若 α 与 β 进制的基数满足 $\alpha^k=\beta$，则 k 位 α 进制数可直接转换为一位 β 进制数。
2. 若 α 与 β 进制的基数满足 $\alpha=\beta^k$，则一位 α 进制数可直接转换为 k 位 β 进制数。
3. k 位一组的分组规则是，整数从低位到高位，小数从高位到低位，且头尾不足 k 位时补 0。

1.2.5 转换位数的确定

在进行数制转换时，必须保证转换所得的数的精度。对于 α 进制中的整数，理论上总是可以准确地转换为有限位的 β 进制数，因而从原理上讲不存在转换精度问题。但对 α 进制中的小数而言，却不一定能转换为有限位的 β 进制数，会出现无限位(循环或不循环)小数情况。例如：

$$(0.2)_{10}=(0.00110011\cdots\cdots)_2$$

因而，在实现小数转换时，必须考虑转换精度问题，即需根据精度要求确定转换所得的小数的位数。

设 α 进制小数为 k 位，为保证转换精度，需取 j 位的 β 进制小数，则必有

$$(0.1)_\alpha^k=(0.1)_\beta^j$$

在十进制中应写成

$$\left(\frac{1}{\alpha}\right)_{10}^k=\left(\frac{1}{\beta}\right)_{10}^j$$