高等学校简明通用系列规划教材

DIGITAL LOGIC
A SIMPLIFIED
APPROACH

数字逻辑简明教程

主　编　江小安　朱贵宪
副主编　黄　同　侯维刚

西安电子科技大学出版社
http://www.xduph.com

高等学校简明通用系列规划教材

数字逻辑简明教程

主　编　江小安　朱贵宪
副主编　黄　同　侯维刚

西安电子科技大学出版社

内 容 简 介

本书共八章，包括数制与编码、逻辑代数与逻辑函数化简、集成逻辑门电路、组合逻辑电路、时序逻辑电路、脉冲波形的产生与变换、数/模与模/数转换、半导体存储器和可编程逻辑器件等。每章均有一定量的例题和练习题。

本书有较宽的适用面，既适用于高等学校工科计算机各专业本科生，也适用于其他相关专业的本科生、高等职业院校专科生，也可作为计算机专业和其他电子领域的工程技术人员的学习参考用书。

图书在版编目(CIP)数据

数字逻辑简明教程/江小安，朱贵宪主编. —西安：西安电子科技大学出版社，2015.1
高等学校简明通用系列规划教材
ISBN 978-7-5606-3545-3

Ⅰ. ① 数… Ⅱ. ① 江… ② 朱… Ⅲ. ① 数字逻辑—高等学校—教材 Ⅳ. ① TP302.2

中国版本图书馆 CIP 数据核字(2014)第 292989 号

策　　划　云立实
责任编辑　阎　彬　董柏娴
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西华沐印刷科技有限责任公司
版　　次　2015 年 1 月第 1 版　2015 年 1 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 14.5
字　　数　341 千字
印　　数　1～3000 册
定　　价　26.00 元
ISBN 978-7-5606-3545-3/TP
XDUP 3837001-1

前　言

“数字逻辑”课程是计算机类各专业和电子信息类专业的重要专业基础课，也是学习计算机相关课程的硬件基础课，同时还是今后工作中十分重要的技能基础。因此学好“数字逻辑”，对计算机各专业和电子信息类各专业的学生而言是十分重要的。

本书共八章。第一章数制与编码，讲述数字逻辑系统中常用的数制、二进制的算术运算及常用的编码。第二章逻辑代数及逻辑函数化简，讲述基本逻辑运算、逻辑运算的基本公式和法则及逻辑函数的化简。第三章集成逻辑门电路，由于我们主要考虑的是门电路的应用，故只讲述 TTL 门电路和 CMOS 门电路的外特性及使用门电路的注意事项和门电路之间的接口电路，对门电路的内部电路并不涉及。第四章组合逻辑电路，含组合逻辑电路的分析与设计，重点讲述集成中规模组合逻辑电路的原理及应用。第五章时序逻辑电路，介绍触发器、时序逻辑电路的分析与设计，重点讲述时序逻辑部件(集成计数器和集成移位寄存器)的原理和应用。第六章脉冲波形的产生与变换，着重讲述 555 定时器的工作原理和应用。第七章数/模与模/数转换，主要讲述数/模转换与模/数转换器的分类及电路的转换原理。第八章半导体存储器和可编程逻辑器件，考虑到很多学校已单独开设与可编程逻辑器件有关的课程(“可编程逻辑器件”或“FPGA”或“电子线路的自动设计”等)，因此重点讲述只读存储器(ROM)在数字电路中的应用，可编程逻辑器件只简要介绍其发展趋势。

本书突出基本概念、基本理论、基本方法，文字上力求叙述流畅、说理清楚，并为读者提供独立分析和设计逻辑电路的思路和方法，内容上不追求系统性和完整性而重在实用性。如逻辑函数的标准式，只讲述常用的最小项标准式，而对不常用的最大项标准式不加以引入。

本书由西安电子科技大学江小安教授、安阳工学院朱贵宪副教授任主编，延安大学黄同老师、西安航空学院侯维刚老师任副主编，全书由江小安教授统稿。

由于编者水平有限，书中难免有不足之处，敬请读者批评指正。

编　者

2014 年 4 月

目　录

第一章　数制与编码 …… 1
1.1　数制 …… 1
1.1.1　十进制 …… 1
1.1.2　二进制 …… 2
1.1.3　八进制与十六进制 …… 2
1.1.4　各种数制之间的转换 …… 4
1.2　二进制数的算术运算 …… 5
1.2.1　二进制数的四则运算 …… 5
1.2.2　原码、反码及补码 …… 6
1.2.3　溢出及补码运算中溢出的判断 …… 8
1.3　常用的编码 …… 9
1.3.1　二一十进制码(BCD码) …… 9
1.3.2　可靠性编码 …… 11
1.3.3　字符代码 …… 12
练习题 …… 13

第二章　逻辑代数与逻辑函数化简 …… 15
2.1　基本概念 …… 15
2.1.1　逻辑变量与逻辑函数 …… 15
2.1.2　真值表 …… 16
2.2　三种基本逻辑运算 …… 17
2.2.1　逻辑乘("与"运算)——AND …… 17
2.2.2　逻辑加("或"运算)——OR …… 18
2.2.3　逻辑非(NOT) …… 19
2.3　常用的复合逻辑 …… 20
2.3.1　"与非"逻辑 …… 20
2.3.2　"或非"逻辑 …… 20
2.3.3　"与或非"逻辑 …… 21
2.3.4　"异或"逻辑及"同或"逻辑 …… 21
2.4　逻辑代数中常用的基本公式和法则 …… 23
2.4.1　基本公式 …… 23
2.4.2　基本法则 …… 25
2.4.3　基本公式的应用 …… 26
2.5　逻辑函数的代数法化简 …… 28
2.5.1　逻辑函数与逻辑图 …… 28
2.5.2　逻辑函数的化简原则 …… 29

2.5.3 与或逻辑函数的化简 …… 29
2.6 卡诺图化简 …… 31
2.6.1 卡诺图化简的基本原理 …… 31
2.6.2 逻辑函数的标准式——最小项 …… 32
2.6.3 卡诺图的结构 …… 34
2.6.4 逻辑函数的卡诺图表示法 …… 35
2.6.5 相邻最小项合并规律 …… 36
2.6.6 与或逻辑的化简 …… 36
2.6.7 其他逻辑形式的化简 …… 39
2.6.8 无关项及其应用 …… 42
练习题 …… 44

第三章 集成逻辑门电路 …… 47
3.1 TTL 集成逻辑门电路 …… 47
3.2 CMOS 集成逻辑门电路 …… 48
3.3 逻辑门电路的特性与参数 …… 49
3.3.1 传输特性 …… 49
3.3.2 输出高电平 U_{OH}、输出低电平 U_{OL} …… 50
3.3.3 噪声容限 …… 50
3.3.4 传输延迟时间 …… 50
3.3.5 功耗 …… 51
3.3.6 延时—功耗积 …… 52
3.3.7 扇入系数与扇出系数 …… 52
3.4 开路门与三态门 …… 53
3.4.1 开路门 …… 53
3.4.2 三态门 …… 54
3.5 集成逻辑门电路使用中的实际问题 …… 55
3.5.1 接口电路 …… 55
3.5.2 抗干扰措施 …… 59
练习题 …… 60

第四章 组合逻辑电路 …… 62
4.1 组合逻辑电路的分析 …… 62
4.2 组合逻辑电路的设计 …… 64
4.3 常用中规模组合逻辑部件的原理和应用 …… 66
4.3.1 运算电路 …… 67
4.3.2 编码器与译码器 …… 75
4.3.3 数据选择器与多路分配器 …… 88
4.3.4 数字比较器 …… 96
4.4 组合逻辑电路中的竞争与冒险 …… 99
4.4.1 竞争现象 …… 99
4.4.2 冒险现象 …… 99
4.4.3 冒险现象的判别 …… 100

4.4.4 冒险现象的消除 …… 101
练习题 …… 103

第五章 时序逻辑电路 …… 108
5.1 时序逻辑电路概述 …… 108
5.1.1 时序逻辑电路的特点 …… 108
5.1.2 时序逻辑电路的分类 …… 109
5.1.3 状态表和状态图 …… 109
5.2 触发器 …… 112
5.2.1 基本触发器 …… 112
5.2.2 集成触发器 …… 120
5.3 时序逻辑电路的分析 …… 125
5.3.1 同步时序逻辑电路分析举例 …… 125
5.3.2 异步时序逻辑电路分析举例 …… 129
5.4 同步时序逻辑电路的设计 …… 130
5.5 计数器 …… 135
5.5.1 计数器的分类 …… 135
5.5.2 集成计数器功能分析及其应用 …… 136
5.6 寄存器与移位寄存器 …… 146
5.6.1 寄存器 …… 146
5.6.2 移位寄存器 …… 148
5.6.3 集成移位寄存器功能分析及其应用 …… 149
5.7 序列信号产生电路 …… 157
5.7.1 序列信号产生电路的设计 …… 158
5.7.2 序列信号产生电路的分析 …… 161
练习题 …… 163

第六章 脉冲波形的产生与变换 …… 169
6.1 概述 …… 169
6.2 555 定时电路 …… 170
6.2.1 基本组成 …… 170
6.2.2 工作原理及功能表 …… 171
6.3 单稳态电路 …… 172
6.3.1 电路组成 …… 172
6.3.2 工作原理及波形计算 …… 172
6.4 多谐振荡器 …… 174
6.4.1 电路组成 …… 174
6.4.2 工作原理及波形计算 …… 175
6.5 施密特电路 …… 177
6.5.1 电路组成 …… 177
6.5.2 工作原理 …… 178
6.5.3 主要应用 …… 178
练习题 …… 179

第七章　数/模与模/数转换 …… 181
7.1　DAC …… 181
7.1.1　DAC 的基本概念 …… 181
7.1.2　DAC 的电路形式及工作原理 …… 183
7.1.3　集成 DAC …… 186
7.2　ADC …… 186
7.2.1　ADC 的组成 …… 187
7.2.2　ADC 电路 …… 188
7.2.3　ADC 的主要技术指标 …… 195
7.2.4　集成 ADC …… 195
练习题 …… 197

第八章　半导体存储器和可编程逻辑器件 …… 198
8.1　半导体存储器 …… 198
8.1.1　只读存储器(ROM) …… 199
8.1.2　ROM 在组合逻辑设计中的应用 …… 200
8.1.3　ROM 的分类 …… 201
8.1.4　随机存取存储器(RAM) …… 204
8.1.5　存储器容量的扩展 …… 207
8.2　可编程逻辑器件 PLD …… 209
8.2.1　PLD 的电路简介 …… 210
8.2.2　PLD 的开发 …… 215
练习题 …… 218

附录一　常用逻辑符号对照表 …… 221
附录二　数字集成电路的型号命名法 …… 223

参考文献 …… 224

第一章　数制与编码

数字系统存在两种不同类型的基本运算，逻辑运算与算术运算。逻辑运算讨论的是输出信号与输入信号的逻辑关系，实现某种控制功能。而算术运算是对数据进行加工，实现数学计算。算术运算的对象是数，因此本章将介绍数字系统中常用的数。数字系统要用到很多符号、数字、字母等，在数字系统中如何表示它们就是本章将要介绍的另一个内容——编码。

1.1　数　　制

目前数的表示通常是采用进位制计数法，即将数划分为不同的数位，按位进行累计，累计到一定数量后，本位归零，同时向高位进位。由于数码的位置不同，同样的数码在不同的数位中表示的数值是不同的。低位数值小，高位数值大。进位制计数法使用较少的数码就能表示较大的数。

1.1.1　十进制

十进制是人们最熟悉的一种数制，它的进位规则是“逢十进一”。十进制中共有 0、1、2、3、4、5、6、7、8、9 这十个代码。如要表示大于 9 的数，则采用进位制计数法表示。如一个多位十进制数为

$$N=(1982.57)_{\mathrm{D}}$$

下标 D 表示是十进制数。将其展开表示为

$$N=1\times10^{3}+9\times10^{2}+8\times10^{1}+2\times10^{0}+5\times10^{-1}+7\times10^{-2}$$

我们读作一千九百八十二点五七。其中 10^{3}、10^{2}、10^{1} 和 10^{0} 分别为千位、百位、十位和个位的权值。上述展开式又称为按权展开式。当某位的数码为 1 时，该位数的值就是权值。一个任意的十进制数可表示为

$$N=\sum_{i=-m}^{n-1}K_i\times10^{i}$$

其中，K_i 是 i 次幂的系数，它可以是 0～9 这十个数码中的任何一个数；n 为整数的位数；m 为小数位数。

如将式中的 10 用字母 R 代替，就可表示任意进制的数：

$$(N)_R=\sum_{i=-m}^{n-1}K_iR_i$$

其中，K_i 根据基数的不同，其取值是 0～$(R-1)$的代码中的任何一个数；R 称为进位基数。

十进制虽然是人们最熟悉的数制，但要用电路实现十分困难，它需要十个状态表示这十个代码。

1.1.2 二进制

二进制的进位规则是“逢二进一”，它只需要 0、1 这两个代码。如要表示大于 1 的数，则采用进位制计数法，如(下标 B 表示是二进制数)：

$$(N)_B=(1011.11)_B=1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$$
$$=8+2+1+0.5+0.25=(11.75)_D$$

其通式为

$$N_B=\sum_{i=-m}^{n-1}K_i2^i$$

由于只有 0、1 两个数码，只需两个状态表达即可，如电位的高和低、脉冲的有与无、开关的合与开等，电路实现十分容易。故数字系统采用二进制。但二进制书写起来太长，且不便于记忆，因此出现了八进制和十六进制，它们常用于编程和书写资料。

1.1.3 八进制与十六进制

1. 八进制

八进制的进位规则是“逢八进一”，它有 0、1、2、3、4、5、6、7 这八个数码。如要表示大于 7 的数，则采用进位制计数法，如(下标 O 表示为八进制数)：

$$N_O=(37.6)_O=3\times8^1+7\times8^0+6\times8^{-1}$$
$$=24+7+0.75=(31.75)_O$$

其通式为

$$N_O=\sum_{i=-m}^{n-1}K_i8^i$$

由于 $2^3=8$，故三位二进制数可以用一位八进制数表示，其对应关系如表 1-1 所示。

表 1-1 三位二进制数与一位八进制数的对应关系

三位二进制数	八进制数
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

这样可以压缩二进制数的长度，如二进制数为$(10110110.1010)_B$，用八进制数表示：整数从低位起每三位一组，最高位不足三位，前面加0；小数部分从高位起每三位一组，最低位不足三位，后面加0，然后用对应的八进制数表示即可。故

$$(10110110.1010)_B = (266.5)_O$$

显然书写长度压缩到原二进制长度的约1/3。

2. 十六进制

十六进制的进位规则是“逢十六进一”，它除了有0～9这十个数码外，再加上A～F这六个英文字母。如要表示大于15的数(即F)，则采用进位制计数法。如(下标H表示是十六进制数)：

$$\begin{aligned} N_H &= (5A \cdot B)_H \\ &= 5 \times 16^1 + 10 \times 16^0 + 11 \times 16^{-1} \\ &= 80 + 10 + 0.6875 \\ &= (90.6875)_D \end{aligned}$$

其通式为

$$N_H = \sum_{i=-m}^{n-1} K_i 16^i$$

由于$2^4=16$，故四位二进制数可用一位十六进制数表示。其对应关系如表1-2所示。

表1-2　四位二进制数与一位十六进制数的对应关系

四位二进制数	十六进制数
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	A(10)
1 0 1 1	B(11)
1 1 0 0	C(12)
1 1 0 1	D(13)
1 1 1 0	E(14)
1 1 1 1	F(15)

它同样可以压缩二进制数的长度。如二进制数为$(10110110.1010)_B$，用十六进制表示：整数部分低位起每四位一组，最高位不足四位，前面补0；小数部分，从高位起每四位一组，最低位不足四位后面加0。故

$$(10110110.1010)_B = (B6.A)_H$$

显然书写长度压缩到原二进制长度的约1/4。

1.1.4 各种数制之间的转换

由于十进制人们最熟悉；二进制是数字系统中采用的数制；八进制和十六进制是编程中书写资料时采用的数制。因此就涉及各种数制间如何转换的问题。

1. 其他进制转换为十进制

将其他进制转换为十进制采用的是按权展开相加的方法。其例子在前面讲述各种数制时已举例，此处不再重复。

2. 十进制转换为其他进制

一个数有整数和小数两部分，它们转换的方式是不同的。整数部分采用连除进位基数R取余法；小数部分采用连乘进位基数R取整法。

【例1-1】 $(92.6875)_D=(?)_B=(?)_O=(?)_H$

解 整数部分：

二进制除2

2	92	…	0 B_0
2	46	…	0 B_1
2	23	…	1 B_2
2	11	…	1 B_3
2	5	…	1 B_4
2	2	…	0 B_5
	1	…	1 B_6

八进制除8

8	92	…	4 O_0
8	11	…	3 O_1
	1	…	1 O_2

十六进制除16

16	92	…	12 (C) H_0
	5	…	5 H_1

$$(92)_D=(1011100)_B=(134)_O=(5C)_H$$

读者不难发现，其八进制和十六进制可直接由二进制转换得到。

小数部分：

二进制乘2

0.6875	
×	2
1 …	1.3750
×	2
0 …	0.7500
×	2
1 …	1.5000
×	2
1 …	1.0000

八进制乘8

0.6875	
×	8
5 …	5.5
×	8
4 …	4.0

十六进制乘16

0.6875	
×	16
B…	11.0

$$(0.6875)_D=(0.1011)_B=(0.54)_O=(0.B)_H$$

同理，其八进制和十六进制也可直接由二进制转换得到。

最后将两部分合起来即得结果，即

$$(92.6875)_D=(1011100.1011)_B=(134.54)_O=(5C.B)_H$$

如果小数部分取整以后得不到0，则根据误差要求而定。

3. 八进制与十六进制之间的转换

八进制转换为十六进制或十六进制转换为八进制，均通过二进制来实现。

【例 1-2】 八进制$(376.35)_O=(?)_H$

解 首先将八进制数写成二进制形式，每一位八进制数写出相应的二进制数，再对二进制数按前述方法，四位一组，写出十六进制数。

$$(376.35)_O=(11111110.011101)_B=(FE.74)_H$$

【例 1-3】 十六进制$(3A.F)_H=(?)_O$

解 首先将十六进制数写成二进制形式，每一位十六进制数写出对应的四位二进制数，再对二进制数按前述方法，三位一组，写出八进制数。

$$(3A.F)_H=(111010.1111)_B=(72.74)_O$$

1.2 二进制数的算术运算

1.2.1 二进制数的四则运算

同十进制数一样，二进制数也可以进行加、减、乘、除四则运算，且运算规则也相同，所不同的是进位基数不相同。十进制是“逢十进一”，二进制是“逢二进一”。由于二进制只有“0”和“1”两个数码，所以二进制的运算比十进制的运算简单，且易于用数字电路来实现。

例如，两个二进制数1111和0101的算术运算如下：

加法运算

```
  1111  (15)
+ 0101  ( 5)
-------------
 10100  (20)
```

减法运算

```
  1111  (15)
- 0101  ( 5)
-------------
  1010  (10)
```

加法运算的规则是：

$$0+0=0,\ 0+1=1,\ 1+0=1,\ 1+1=10$$

最低位相加时只是被加数与加数相加，没有低位的进位，这种不考虑低位进位的加法称为“半加”；而其他各位，除了被加数和加数相加之外，还必须考虑低位向本位的进位，即“被加数＋加数＋低位向本位的进位”，产生和数和向高位的进位，这种加法称为“全加”。

减法运算的规则是：

1－1＝0，1－0＝1，0－0＝0，0－1＝1　（向高位借1当2）

最低位相减时只是被减数与减数相减，没有低位的借位，这种不考虑低位向本位借位的减法称为“半减”；而其他各位，除了被减数和减数相减之外，还必须考虑低位向本位的借位，产生差和本位向高位的借位，这种减法称为“全减”。

需要指出的是，在数字系统中，为了减少设备量，用加法代替减法，即用加补码进行运算。关于补码和补码运算将在后面进行介绍。

乘法运算

```
   1111  (15)
 × 0101  ( 5)
 ------
   1111
  0000
 1111
0000
--------
1001011  (75)
```

除法运算

```
    11
101 1111
   101
   ---
    0101
     101
     ---
     000
```

由上述运算可看出，二进制乘法运算是由左移被乘数与加法运算组成的；除法运算是由右移除数与减法运算组成的。

如除法运算，若在规定位数除不尽，则商的位数由给定的精度来确定。

1.2.2　原码、反码及补码

前面所讲的二进制数，没有提到符号问题，故是一种无符号数。但在实际中，数有正、负之分，用“＋”和“－”表示。那么，在数字设备中“＋”、“－”是如何表示的呢？将二进制数前增加一个符号位：正数用“0”表示；负数用“1”表示。如绝对值为9的数，表示如下(用八位二进制码表示，最高位为符号位)：

数　　真值　　原码

＋9＝＋0001001＝00001001

－9＝－0001001＝10001001

我们将加了符号位的二进制数称为原码。原码的优点是易于辨认，因为它的数值部分就是该数的绝对值，而且与真值和十进制数的转换十分方便。但在采用原码进行运算时，如两个异号数相减，则首先判定哪个数的绝对值大，将绝对值大的数作为被减数，绝对值小的数作为减数，所得差数的符号与绝对值大的数的符号保持一致。这样数字设备就要增加判定数大小的比较设备，且用减法电路完成减法运算，这样就增加了设备量。为了减少设备量，数字设备一般都采用补码进行运算，用加法代替减法。

采用补码进行运算，在日常生活中最常用的例子就是钟表。如现在是六点整，而钟表停在九点，要返回六点，可以倒拨三个小时返回到六点，这是减法，即9－3＝6。也可采用顺拨九个小时到六点，这是采用的加的方法，即9＋9＝18＝12＋6。由于钟表是十二进制，进位自动丢失，这样就用加法9＋9的运算代替了减法9－3的运算。由于9＋3＝12，正好

是其进位模数。9 和 3 互为补数，即 9 是－3 的补数，我们称为补码。上述调钟表的过程可用图 1－1 表示。

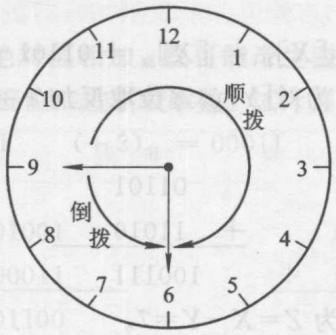

图 1－1 “钟表”说明补码的运算

上述例子说明减法可以用加补码的方法实现。下面再举十进制减法来说明补码运算。

$$8-4=8+(-4)^{*}=8+6=14=10+4=4$$

对于十进制而言，进位位自动丢失。其中，$(-4)^{*}$ 表示－4 的补码，它等于 $10-4=6$。

补码的定义为，当进位模为 R 时，

$$N^{*}\begin{cases}A & (A\text{ 为正数时})\\ R-|A| & (A\text{ 为负数时})\end{cases}$$

对二进制数而言（n 位二进制），

$$N^{*}\begin{cases}A & (A\text{ 为正数时})\\ 2^{n}-|A| & (A\text{ 为负数时})\end{cases}$$

从定义可看出，求补码时仍要用减法。对于二进制的补码，可以通过逻辑运算和加法运算求出，即将原码保持符号位不变，其余各位逐位取反（反码）再加 1。

【例 1－4】 求 ＋6 和－6 的四位二进制的原码、反码和补码。

解 按定义，正数三者均相同。

$$+6\rightarrow 00110(\text{原码})\rightarrow 00110(\text{反码})\rightarrow 00110(\text{补码})$$

$$-6\rightarrow 10110(\text{原码})\xrightarrow{\text{取反}}11001(\text{反码})\xrightarrow{+1}11010(\text{补码})$$

如按定义求补码，四位二进制数其进位制是 $2^{4}=10000$，$10000-0110=1010$，前再加符号位，得 11010。结果相同。

【例 1－5】 已知数 $X=+13$，$Y=+6$，请用原码、补码计算 $Z=X-Y$。

(1) 采用原码运算。

$$[X]_{\text{原}}=01101$$

$$[Y]_{\text{原}}=00110$$

首先判别相减两数的大小，本例是 $X>Y$，故 X 为被减数，Y 为减数，且符号位与 X 相同。

$$\begin{array}{rl} 01101 & (13)\\ -\ 00110 & (6)\\ \hline 00111 & (7)\end{array}$$

结果为 $Z=00111$，其真值为＋7。

(2) 采用补码运算。

$$X-Y=X+[-Y]^{*}$$

写出 X 和 $-Y$ 的补码如下：

$$[X]_{补}=[X]_{原}=01101$$

$$[-Y]_{补}\text{ 为对 }[Y]_{原}\text{ 逐位取反加 }1\rightarrow 11010$$

故

$$\begin{array}{r} 01101 \\ +\ 11010 \\ \hline 100111 \end{array}$$

其符号进位自然丢失，故其结果为 $Z=X-Y=7$。

【例 1-6】 将例 1-5 的数进行计算：$Y-X$

(1) 采用原码运算。

显然 $X>Y$，故计算结果与 $[X]_{原}$ 的符号相反。

$$\begin{array}{r} 01101 \\ -\ 00110 \\ \hline 10111 \end{array}$$

(2) 采用补码运算。

$$Y-X=Y+[-X]$$

$$[Y]_{补}=00110$$

$$[X]_{补}=10011$$

$$\begin{array}{r} 00110 \\ +\ 10011 \\ \hline 11001 \end{array}$$

此处需注意，补码运算其结果也是补码。如结果是正数，可直接读出，如例 1-5；如结果是负数，不可直接读出，应将结果 11001 再求补一次才是正确结果，即 10111(负 7)。

从上述运算的例子可看出，补码运算有如下特点：

(1) 将所有参与运算的数均用补码表示。

(2) 用加法代替减法。

(3) 符号位也按二进制的规则参与运算，也产生和数和向高位的进位。所得和数结果的符号位是正确的。

(4) 补码运算的结果也是补码。当结果是正数时，可直接读出结果的值；当结果是负数时，一定要将其求补一次，才能得到正确的值。

1.2.3 溢出及补码运算中溢出的判断

如果运算的结果大于数字设备所能表示数的范围就产生溢出。溢出现象应当作一种故障来处理，因为它会使结果数发生错误。例如，某数字设备用八位二进制表示数，则它所能表示补码数的范围为 01111111～10000000，即＋127～－128。如运算结果大于＋127 或小于－128 均产生溢出，结果错误。

由于补码运算存在丢失进位现象，运算结果正确。因此应区分溢出与正常进位。

异号两数相加时，实际是两数的绝对值相减，不可能产生溢出，产生的进位是正确进位。

同号两数相加时，是两者绝对值相加，既可能产生溢出，也可能出现正确进位。

【例 1-7】 某数字设备用五位二进制表示数，试计算 9+3，9+12，−9−3 和−9−12。

$$(+9)_{补}=01001 \qquad (+3)_{补}=00011 \qquad (+12)_{补}=01100$$

即

$$\begin{array}{r} 01001 \\ +\ 00011 \\ \hline 01100 \end{array} \qquad \begin{array}{r} 01001 \\ +\ 01100 \\ \hline 10101 \end{array}$$

01100＝＋12 结果正确。10101＝－1011＝－11 结果显然是错误的，两个正数相加，其结果为负数。这就是产生了溢出。因为五位二进制数的补码最大只能表示 01111＝＋15，而 9＋12＝＋21 超过设备所能表示的最大数。

$$(-9)_{补}=10111$$
$$(-3)_{补}=11101$$
$$(-12)_{补}=10100$$

即

$$\begin{array}{r} 10111 \\ +\ 11101 \\ \hline (1)10100 \end{array} \qquad \begin{array}{r} 10111 \\ +\ 10100 \\ \hline (1)01011 \end{array}$$

10100＝－1100＝－12 结果正确。而 01011＝＋11，显然是错误的，两个负数相加其结果为正数。这也是产生了溢出。因为五位二进制数的补码所能表示的最小负数为－16，而－9－12＝－21 超过设备所能表示的最小数。

根据上述运算，我们可以找出判断溢出还是正常进位的规律。

凡是最高位和次高位均产生进位或均无进位，则运算结果正确；若最高位和次高位只有一位产生进位，另一个无进位，则运算结果错误，产生了溢出。在数字系统中，通常是利用最高两位的进位位来判断是否产生溢出。在第二章讲到基本逻辑运算时，将会讲到用异或逻辑即可。

1.3 常用的编码

对若干个不同的数据或信息，按一定的规律分别给其指定一个代表符号的过程叫编码。这些代表给定数据和信息的符号叫代码，简称码。在数字系统中，所有的代码都是用若干位二进制码元“0”和“1”的不同组合构成的。因此，这种代码习惯上称为二进制代码。这里“二进制”并无“进位”的含义，只是强调采用的是二进制数的数码符号而已。n 位的二进制码元，共有 2^n 种不同的组合，可以用其代表 2^n 种不同的信息。

1.3.1 二—十进制码(BCD 码)

二—十进制码是用二进制码元来表示十进制数符“0～9”的代码，简称 BCD 码(Binary