

数字逻辑设计

高翠芸

School of Computer Science

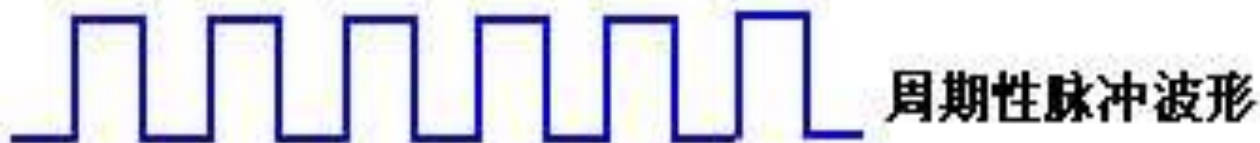
gaocuiyun@hit.edu.cn

基本概念和数制编码



- 基本概念
- 数制
- 编码
 - BCD码 (BCD code)
 - 余3码 (Excess-3 code)
 - 格雷码 (Gray code)

脉冲波形



数字电路中的脉冲波形

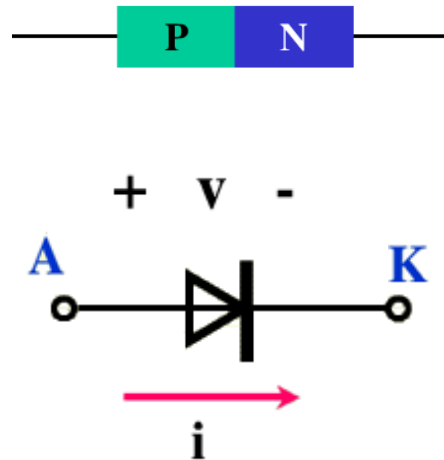
开关器件

数字系统使用的是具有两种状态的开关器件

- 如：二极管、三极管



二极管由PN结组成，具有单向导电性



$$V \geq V_{ON}$$

二极管导通

$$V < V_{ON}$$

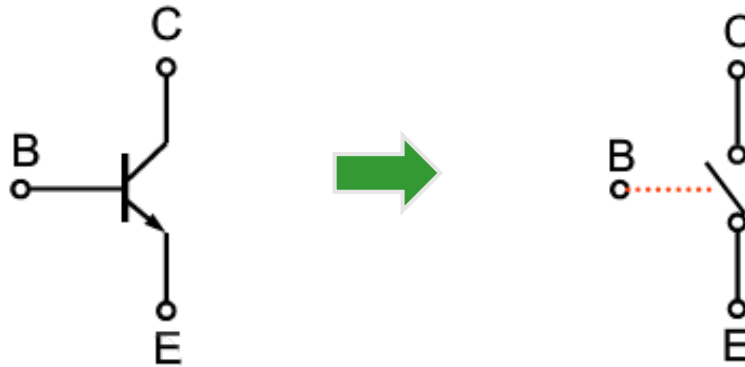
二极管截止



开关器件

三极管

- 利用三极管的**饱和、截止**状态作开关
- 三极管开关的**通、断**受基极b的电位高低控制



由于大多数开关器件只能取两个不同的值，
所以数字系统内部使用二进制也就很自然了。

问题：为何使用二进制？



- 电路简单
- 对电器元件要求不高
- 可靠稳定
- 精确
- 易于存储
- 方便计算机处理

问题：为何使用二进制？

- 逻辑运算：+,-,×,÷
- 逻辑推理判断：
 - 举重比赛的评判电路
 - 自动售饮料机电路
 - 时序锁
 -



分析方法与模拟电路不同

模拟电路

微变等效电路

——电路分析

数字电路

逻辑分析方法

数学工具：

布尔代数

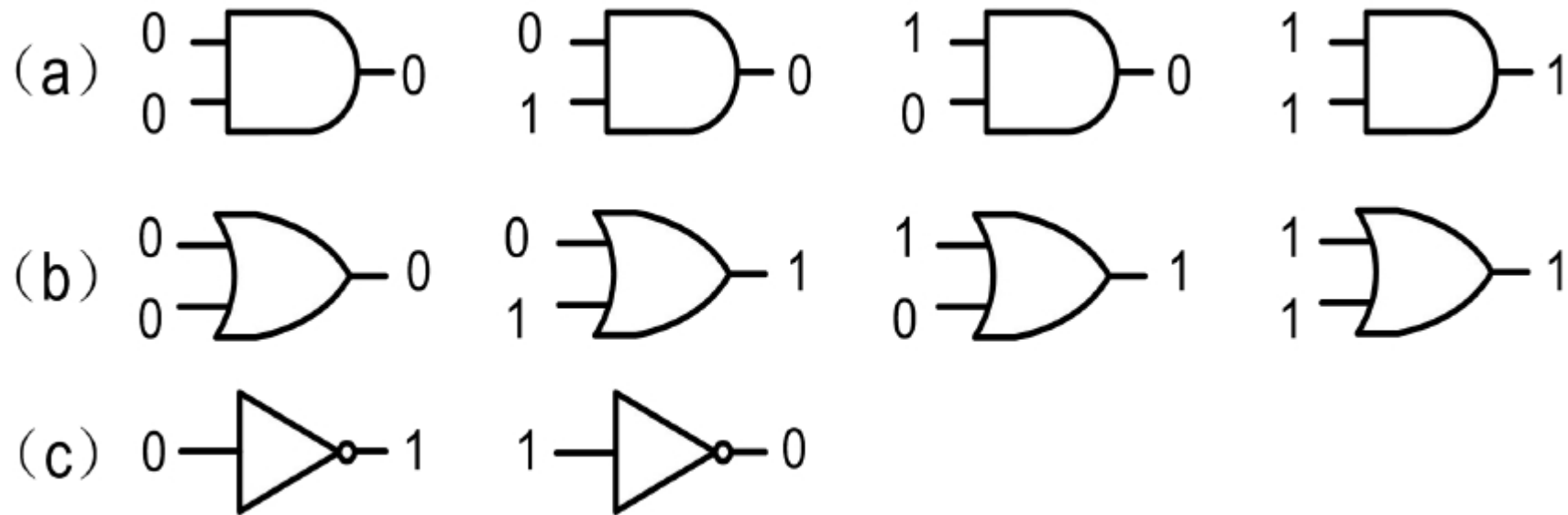
描述方法：

真值表

表达式

功能表等

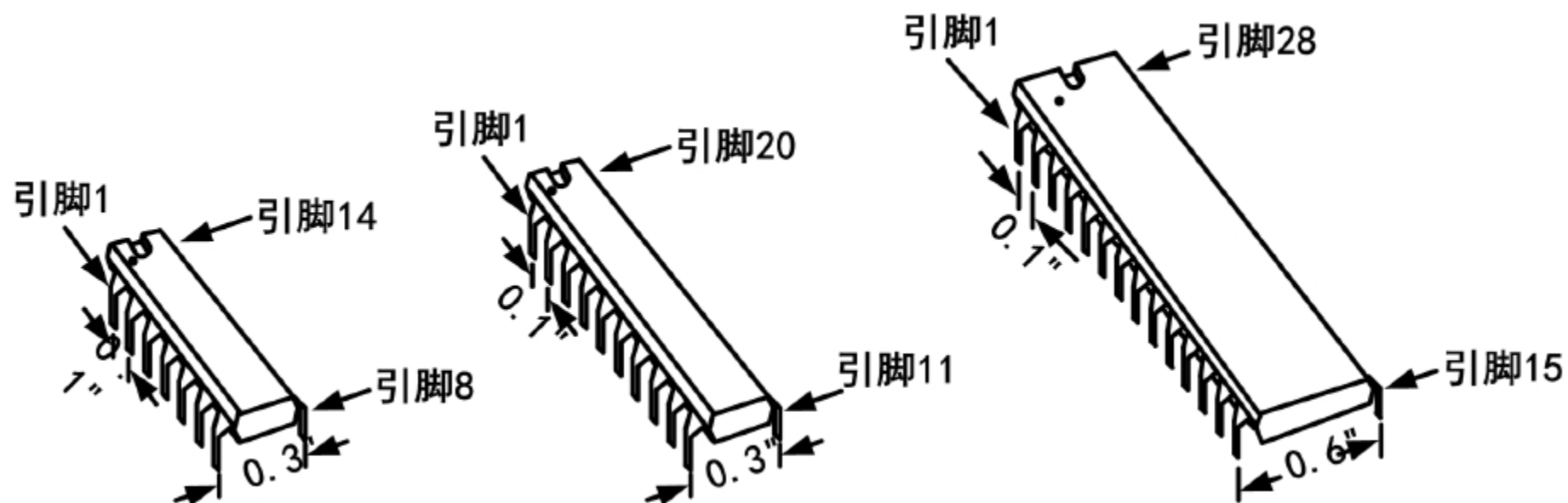
逻辑电路和门电路



(a) AND Gate (b) OR Gate (c) NOT Gate or Inverter

集成电路

- 单晶硅片(Wafer) – 》模片 (Die)
- 双列直插式封装(Dual Inline-pin Package)



集成电路

- ✱ 小规模集成 (SSI, Small-Scale Integration):
1-20 Gates
- ✱ 中规模集成 (MSI, Medium-Scale Integration):
20-200 Gates
- ✱ 大规模集成 (LSI, Large-Scale Integration):
200-1,000,000 Gates
- ✱ 超大规模集成 (VLSI, Very Large-Scale Integration):
Over 1,000,000 Transistors

可编程逻辑器件

- ✱ **可编程阵列逻辑 (PAL, Programmable Array Logic)**
- ✱ **可编程逻辑器件 (PLD, Programmable Logic Device)**
- ✱ **复杂可编程逻辑器件 (CPLD, Complex PLD)**
- ✱ **现场可编程门阵列 (FPGA, Field-Programmable Gate Array)**

复杂集成电路

- ✱ 半定制IC(Semi-Custom IC)

非再现工程成本 (NRE, Non-Recurring Engineering Cost): \$10,000 - \$500,000

- ✱ 全定制IC(Custom IC) NRE Cost: Over \$500,000

印制电路板

- ✿ 印制线路板
(PWB, Printed-Wiring Boards)
- ✿ 表面安装技术
(SMT, Surface-Mount Technology)
- ✿ 多芯片模块
(MCM, Multi-Chip Module)
- ✿ 片上系统
(SoC, System on Chip)
- ✿ 片上网络
(NoC, Network on Chip)



数字设计层次

- ✱ 器件物理层(Device Physics Level)
- ✱ IC 制造过程级
(IC Manufacturing Process Level)
- ✱ 晶体管级 (Transistor Level)
- ✱ 门电路结构级(Gates Structure Level)
- ✱ 逻辑设计级 (Logic Design Level)
- ✱ 整体系统设计 (Overall System Design)

基本概念和数制编码

- 基本概念
- 数制
- 编码
 - BCD码 (BCD code)
 - 余3码 (Excess-3 code)
 - 格雷码 (Gray code)



数制和编码

- 数制
- 数字的表示

$$D = d_{p-1} d_{p-2} \dots d_1 d_0 . d_{-1} d_{-2} \dots d_{-n}$$

- LSB (least significant bit)
- MSB (most significant bit)

按位计数制

任意十进制数D 可表示如下:

$$\begin{aligned} D &= d_{p-1} d_{p-2} \dots d_1 d_0 . d_{-1} d_{-2} \dots d_{-n} \\ &= \sum_{i=-n}^{p-1} d_i \times r^i \end{aligned}$$

推广:

$$B = \sum b_i \times 2^i$$

$$H = \sum h_i \times 16^i$$

第*i*位的权(Weight); *r* 是计数制的基数 (Base or Radix)

✳ 按位计数制的特点

- 1) 采用**基数** (Base or Radix), R进制的基数是R
- 2) **基数**确定数符的**个数**。如十进制的数符为: 0、1、2、3、4、5、6、7、8、9, 个数为10; 二进制的数符为: 0、1, 个数为2
- 3) 逢**基数**进一

二进制、八进制与十六进制数

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	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

二进制与八进制和十六进制之间的转换

位数替换法：保持小数点不变，每位**八**进制数对应**3**位二进制数；每位**十六**进制数对应**4**位二进制数；

二进制转换为**八**进制或**十六**进制数时，从小数点开始向左右分组，在MSB(Most Significant Bit)前面和LSB(Least Significant Bit)后面可以加0；

八进制或**十六**进制转换为二进制数时，MSB前面和LSB后面的0不写；

例： $10111000.1101_2 = 270.64_8 = B8.D_{16}$

二进制加法运算(Binary Addition)

二进制加法真值表

输 入				输 出	
被加数X	加数Y	输入进位C _{in}		和S	进位输出C _{out}
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

二进制减法运算(Binary Subtraction)

二进制减法真值表

输 入				输 出	
被减数X	减数Y	输入借位B _{in}		差D	输出借位B _{out}
0	0	0		0	0
0	0	1		1	1
0	1	0		1	1
0	1	1		0	1
1	0	0		1	0
1	0	1		0	0
1	1	0		0	0
1	1	1		1	1

原码表示法

- ★最高有效位表示符号位 (Sign bit)

- ★**0 = 正, 1 = 负** (0 = plus, 1 = minus)

- ★其余各位是该数的绝对值

- ★ $01111111 = +127$

 $11111111 = -127$ $00101110 = +46$ $10101110 = -46$

- ★零有两种表示 (**+ 0、- 0**)

 $00000000 = +0$ $10000000 = -0$

- ★8位二进制码能够表示的带符号十进制数中,

最大的数是 +127, 而最小的数是 -127。

- ★ n位二进制整数表示的范围:

$$-(2^{n-1}-1) \sim +(2^{n-1}-1)$$

反码表示法

※ **正数**的二进制反码表示与原码相同

※ **负数**的二进制反码表示：

在 n 位系统中, 符号位不变, 其余各位在原码基础上按位取反

补码表示法

※ **正数**的二进制补码表示与原码相同

※ **负数**的二进制补码如何求取？

反码(Ones' - Complement) + 1

(零只有一种表示) $0 = 00000000$

※ 逐位取反


1 1 1 1 1 1 1 1

+ 1

※ 约定8位

0 0 0 0 0 0 0 0 = 0

基本概念和数制编码

- 基本概念
 - 数制
 - 编码
- 
- BCD码 (BCD code)
 - 余3码 (Excess-3 code)
 - 格雷码 (Gray code)

编码



变色龙，拱猪，接龙

玩法N多，本质上，就是54张牌在不同游戏规则下的组合而已

■ 编码

- BCD码
- 余3码
- 格雷码

编法N多，本质上，就是0和1在不同**编码规则**下的组合而已。

BCD码

BCD码 (Binary-Coded Decimal) 也叫二-十进制编码，用4位二进制数表示1位十进制数

4位二进制码共有 $2^4=16$ 种码组，在这16种代码中，可以任选10种来表示10个十进制数码

每位二进制数都带有权值

- 根据权值不同，称其为：

8421BCD

2421BCD

4221BCD ...

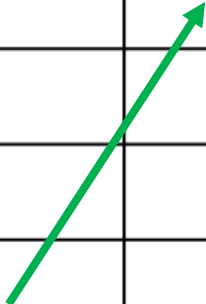
Decimal	8421BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD码

Decimal	8421BCD	2421BCD	4221BCD	5421BCD
0	0000	0000 (0000)	0000 (0000)	0000 (0000)
1	0001	0001 (0001)	0001 (0001)	0001 (0001)
2	0010	0010 (1000)	0010 (0100)	0010 (0010)
3	0011	0011 (1001)	0011 (0101)	0011 (0011)
4	0100	0100 (1010)	0110 (1000)	0100 (0100)
5	0101	1011 (0101)	1001 (0111)	1000 (0101)
6	0110	1100 (0110)	1100 (1010)	1001 (0110)
7	0111	1101 (0111)	1101 (1011)	1010 (0111)
8	1000	1110 (1110)	1110 (1110)	1011 (1011)
9	1001	1111 (1111)	1111 (1111)	1100 (1100)

余3码

Decimal	8421BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100



- 无权码
- 自补性: 对9的自补码
- 8421BCD码+ “0011”

3. 典型格雷码 (Gray code)

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

Decimal	Binary	Gray code
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

任何两位相邻编码
只有1位码元不同

怎样计算任意给定的二进制数对应的典型格雷码？

1) 算法

- 复制最高位
- 从最高位开始，俩俩比较相邻位：
 - 二者相同取 0
 - 二者不同取 1
- 转换前后数据的位宽不变

Binary:

1 0 1 1 0 1 1 0 1

Gray Code:

1 1 1 0 1 1 0 1 1

如何由n位典型格雷码写n+1位典型格雷码

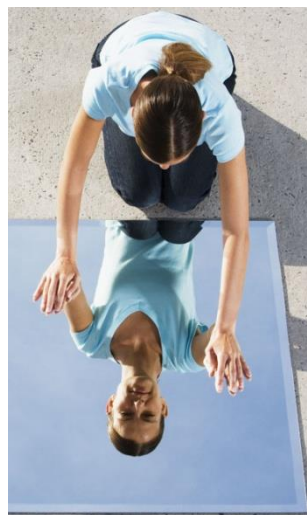
2) 反射法

1位

0
1

2位

0	0
0	1
1	1
1	0



3位

0 0 0

0 0 1

0 1 1

0 1 0

1 1 0

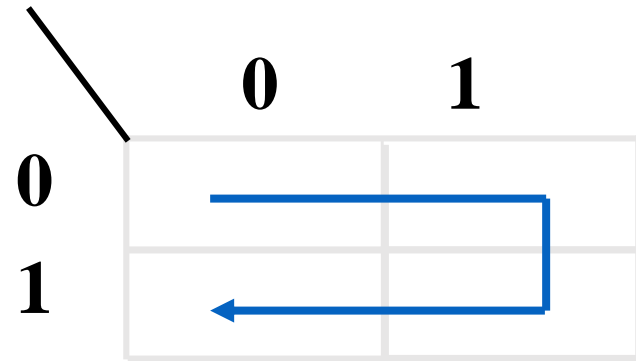
1 1 1

1 0 1

1 0 0

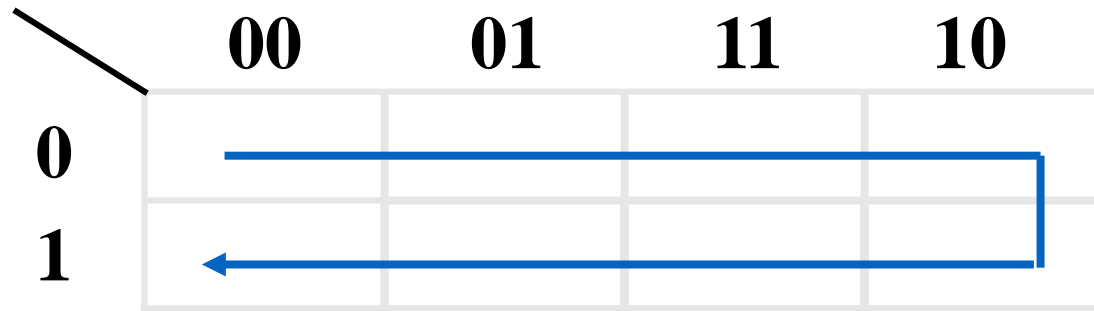
如何写n位典型格雷码

3) 图形法



2位格雷码

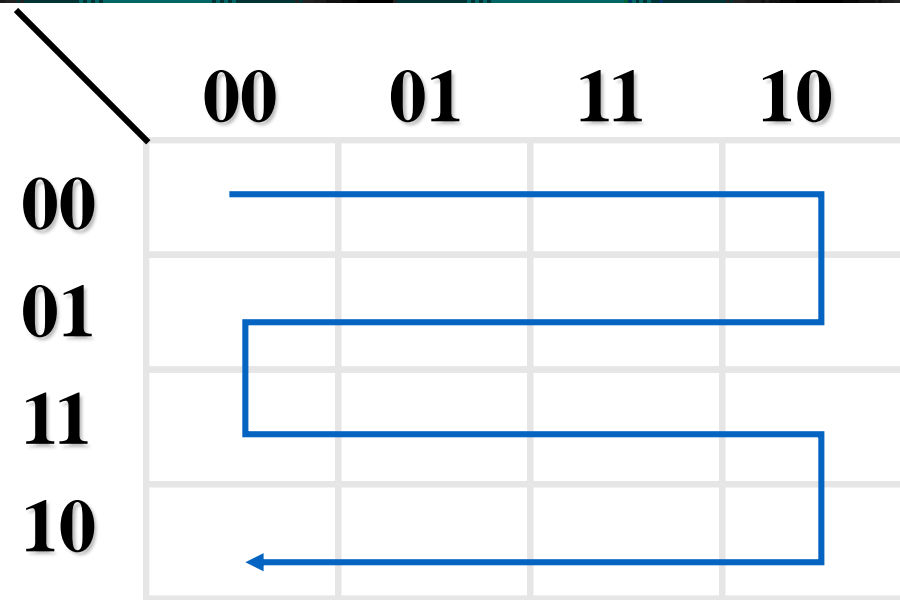
00、01、11、10



3位格雷码

000、001、011、
010、110、111、
101、100

Gray Code


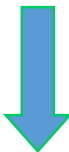


4位格雷码

0000、0001、0011、0010、0110、0111、0101、
0100、1100、1101、1111、1110、1010、1011、
1001、1000

Gray Code

Example 十进制: 3→4

	8421BCD	Gray Code
3	0 011	0 010
		
4	0 100	0 110
	3 位码元改变	1 位码元改变



Gray Code ——连续变化时, 比较可靠

小 结

- 基本概念
- 数制
- 编码
 - BCD码 (BCD code)
 - 余3码 (Excess-3 code)
 - 格雷码 (Gray code)

小 结

- 概述
- 课程简介
- 基本概念
- 数制
- 编码
 - BCD码 (BCD code)
 - 余3码 (Excess-3 code)
 - 格雷码 (Gray code)
 -

对哪部分内容有疑问？

- ☐ A 无
- ☐ B 考核方式
- ☐ C 教材
- ☐ D 其他

提交

二进制表示法

4 位有符号二进制数的原码、反码、补码对照表

十进制数	二进制数		
	原码	反码	补码
-8	——	——	1000
-7	1111	1000	1001
-6	1110	1001	1010
-5	1101	1010	1011
-4	1100	1011	1100
-3	1011	1100	1101
-2	1010	1101	1110
-1	1001	1110	1111
-0	1000	1111	0000
+0	0000	0000	0000

十进制数	二进制数		
	原码	反码	补码
+1	0001	0001	0001
+2	0010	0010	0010
+3	0011	0011	0011
+4	0100	0100	0100
+5	0101	0101	0101
+6	0110	0110	0110
+7	0111	0111	0111

补码表示法

- ✱ 基数补码 (Radix – Complement)
- ✱ 从 r_n 中减去该数
- ✱ 基数减1补码 (反码) (Diminished Radix – Complement)
- ✱ 从 $r_n - 1$ 中减去该数

二进制补码的加减法

➤ $[X]_{\text{补}} + [Y]_{\text{补}} = [X+Y]_{\text{补}} \pmod{M}$

两个数的补码之和等于两数之和的补码。

➤ $[X]_{\text{补}} - [Y]_{\text{补}} = [X]_{\text{补}} + [-Y]_{\text{补}} = [X-Y]_{\text{补}} \pmod{M}$

两个数的补码之差等于两数之差的补码。

◆ 注意：

➤ 参与运算的操作数均为补码，运算的结果仍然以补码表示。

➤ 运算时，符号位和数值位按同样的规则参加运算，结果的符号位由运算得出。

➤ 补码总是对确定的模而言，如果运算结果超过了模，则应将模（即进位）丢掉才能得到正确结果。

二进制补码的加减法

补码的加减运算

➤ 求 $15 - 13 = ?$ (用补码)

直接做减法运算

$$\begin{array}{r} 00001111 \quad (15) \\ - 00001101 \quad (13) \\ \hline 00000010 \quad (2) \end{array}$$

$$\begin{aligned} \because (15 - 13)_{\text{补}} &= (15)_{\text{补}} - (13)_{\text{补}} \\ &= (15)_{\text{补}} + (-13)_{\text{补}} \end{aligned}$$

转换为补码做加法运算

$$\begin{array}{r} \text{进位 } 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\ 00001111 \quad (15)_{\text{补}} \\ + 11110011 \quad (-13)_{\text{补}} \\ \hline 1 \quad 00000010 \quad (2)_{\text{补}} \end{array}$$

舍弃进位

◆ 注意：

➤ 在进行二进制补码的加法运算时，被加数与加数的位数要相等，即让两个二进制数补码的符号位对齐。

➤ 两个二进制数的补码要采用相同的位数表示。

二进制补码的加减法

补码的加减运算

➤ 求 $13 - 15 = ?$ (用补码)

∴ $(13 - 15)_{\text{补}} = (13)_{\text{补}} + (-15)_{\text{补}}$

进位 0 0 0 0 0 0 1

0	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---

 $(13)_{\text{补}}$
+

1	1	1	1	0	0	0	1
---	---	---	---	---	---	---	---

 $(-15)_{\text{补}}$

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

 $(-2)_{\text{补}}$

➤ 求 $-13 - 15 = ?$ (用补码)

∴ $(-13 - 15)_{\text{补}} = (-13)_{\text{补}} + (-15)_{\text{补}}$

进位 1 1 1 0 0 1 1

1	1	1	1	0	0	1	1
---	---	---	---	---	---	---	---

 $(-13)_{\text{补}}$
+

1	1	1	1	0	0	0	1
---	---	---	---	---	---	---	---

 $(-15)_{\text{补}}$

1	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---

 $(-28)_{\text{补}}$

舍弃进位

二进制补码的加减法

补码的加减运算

➤ 求 $125+58 = ?$ (用补码)

因为 $(125+58)_{\text{补}} = (125)_{\text{补}} + (58)_{\text{补}}$

进位

1 1 1 1

0	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---

 (125)_补

+

0	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---

 (58)_补

1	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

 (183)_补

✗

二进制补码的加减法

补码的加减运算

➤ 求 $-105-50 = ?$ (用补码)

因为 $(-105-50)_{\text{补}} = (-105)_{\text{补}} + (-50)_{\text{补}}$

进位 0 0 1 1 1 1

1	0	0	1	0	1	1	1	$(-105)_{\text{补}}$
+								
1	1	0	0	1	1	1	0	$(-50)_{\text{补}}$
<hr/>								
1	0	1	1	0	0	1	0	$(-155)_{\text{补}}$

舍弃进位

错误原因是：

➤ 8位有符号数所能表示的补码

数的最小值为-128.

➤ 这里， $-155 < -128$ ，也产生了溢出。

➤ 发生溢出的原因是因为和的位数是固定的。

二进制补码的加减法

补码的加减运算

- 溢出的判别对有符号数的运算是非常重要的，它表明结果是否超出范围。
- 溢出仅发生在两个同符号的数（两个正数或者两个负数）相加的情况下。
 - 如果两个正数相加的结果大于机器所能表示的最大正数，称为**正溢出**。
 - 如果两个负数相加的结果小于机器所能表示的最小负数，称为**负溢出**。
- 出现溢出后，机器将无法正确地表示运算结果，因此，在计算机中，有专门的电路用来检测两个数相加时产生的溢出。
- 这个检测单元将通知计算机的控制单元发生了溢出，运算结果是错误的。

Binary Code

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

一种可靠性编码

Gray Code

编码器

0000

