数字逻辑设计

秦阳
School of Computer Science

csyqin@hit.edu.cn

任意十进制数(Decimal Number) D 可表示如下
:第i位的权(Weight)； r 是计数制的基数（Base or Radix）

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

$$= \sum_{i=-n}^{p-1} d_i \times r^i$$

推广：

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

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

- 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

转换方法：

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前面和LSB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：$\mathbf{10111000.1101_2=270.64_8=B8.D_{16}}$

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Cin		和S	进位输出Cout
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：
  $-(2^{n-1}-1) \sim +(2^{n-1}-1)$

补码数制

- 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，等于 **2^n-1 -A**，即
- 符号位不变，其余各位在原码基础上按位取反

二进制补码表示法

- 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＝0 0 0 0 0 0 0 0**
  - 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**

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

二进制补码的加减法

- $[X]_{补}+[Y]_{补}=[X+Y]_{补}$ （mod M）

  两个数的补码之和等于两数之和的补码。

- $[X]_{补}-[Y]_{补}=[X]_{补}+[-Y]_{补}=[X-Y]_{补}$ （mod 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

➢ 求 15 – 13 = ？ (用补码)

∵ $(15-13)_{补}=(15)_{补}-(13)_{补}$

$=(15)_{补}+(-13)_{补}$

直接做减法运算

	0	0	0	0	1	1	1	1	(15)
−	0	0	0	0	1	1	0	1	(13)
	0	0	0	0	0	0	1	0	(2)

转换为补码做加法运算

进位	1	1	1	1	1	1	1		
	0	0	0	0	1	1	1	1	$(15)_{补}$
+	1	1	1	1	0	0	1	1	$(-13)_{补}$
1	0	0	0	0	0	0	1	0	$(2)_{补}$

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

➢ 求 13 – 15 = ？ (用补码)

∵ （13-15)补 = (13)补+ (–15)补

进位 0 0 0 0 0 0 1

	0	0	0	0	1	1	0	1	(13)补
+	1	1	1	1	0	0	0	1	(-15)补
	1	1	1	1	1	1	1	0	(-2)补

➢ 求 –13 – 15 = ？ (用补码)

∵ （-13-15)补 = (-13)补+ (–15)补

进位 1 1 1 0 0 1 1

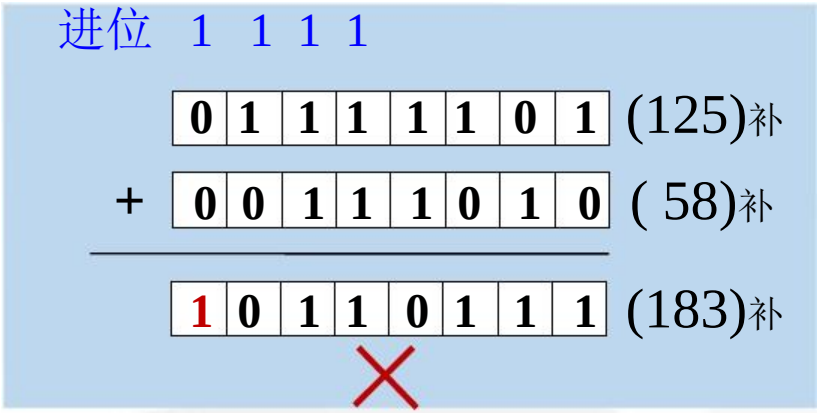
	1	1	1	1	0	0	1	1	(-13)补
+	1	1	1	1	0	0	0	1	(-15)补
1	1	1	1	0	0	1	0	0	(-28)补

舍弃进位

二进制补码的加减法

➢ 求 **125+58 = ？ (用补码)**

因为（**125+58)**补 **= (125)**补**+ (58)**补



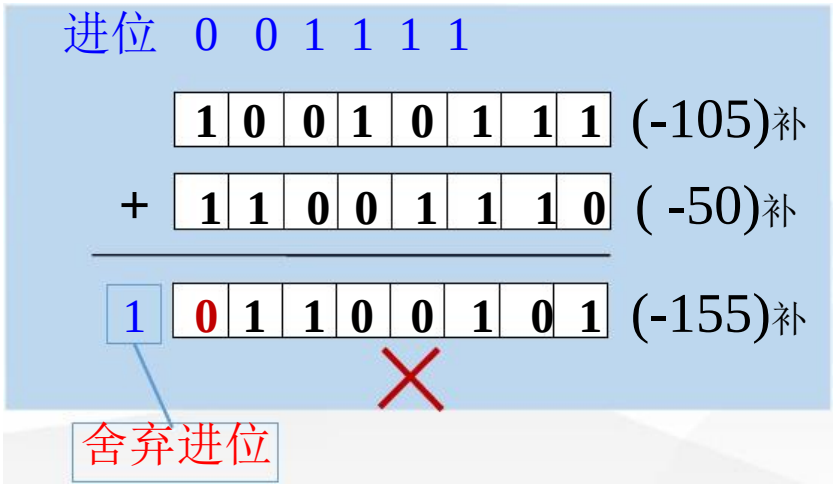
错误原因是:

- **8**位有符号数所能表示的补码数的最大值为**127.**
- 这里，**183 > 127**，导致结果错误。
- 我们把超出表示范围的这种情况称为溢出（**Overflow**）。

二进制补码的加减法

➢ 求 **-105-50 = ？ (用补码)**

因为（**-105-58)**补 **= (-105)**补**+ (-50)**补



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

- 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。

编码

变色龙，拱猪，接龙 .....

玩法很多，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

编法很多，本质上，就是0和1在不同编码规则下的组合而已。

BCD码

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

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 …

BCD码

Decimal	8421code	2421code	4221code	5421code
0	0000	0000	0000	0000
1	0001	0001	0001	0001
2	0010	0010（1000）	0010（0100）	0010
3	0011	0011（1001）	0011（0101）	0011
4	0100	0100（1010）	0110（1000）	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	1011
9	1001	1111	1111	1100

余3码

Decimal	8421 BCD	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

余3码（Excess-3 code）

- 无权码
- digit is not weighted
- Self-complementing
- 8421code + “0011”

典型格雷码

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

典型格雷码（Gray code）

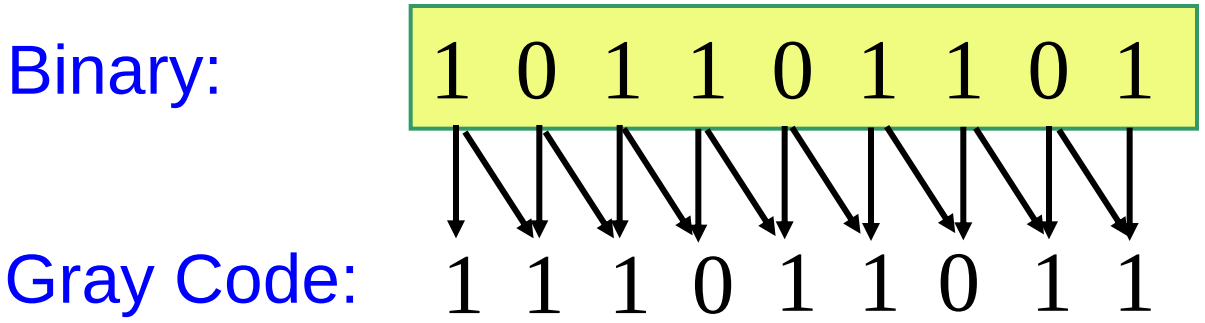
- 无权码

  任何两位相邻编码只有1位码元不同

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

怎样计算任意给定的二进制数对应的典型格雷码？

- 计算法
  - 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



如何由n位典型格雷码写n+1位典型格雷码

- 反射法

1位

0

1

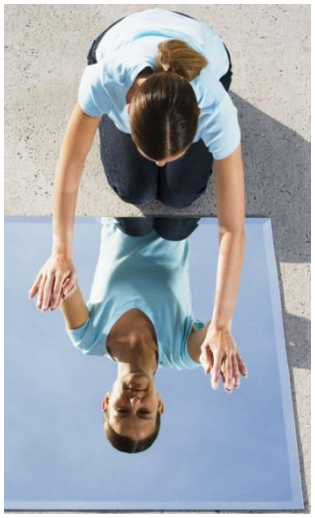
2位

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

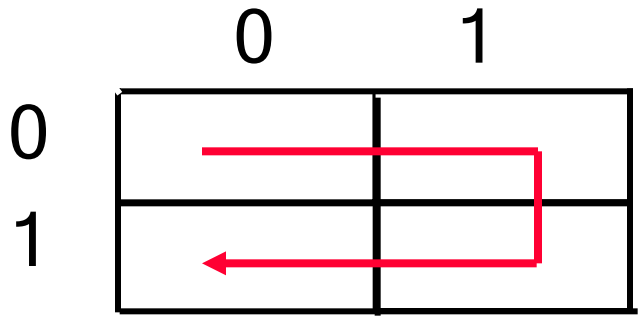
1 1 1

1 0 1

1 0 0

如何写n位典型格雷码

- 图形法



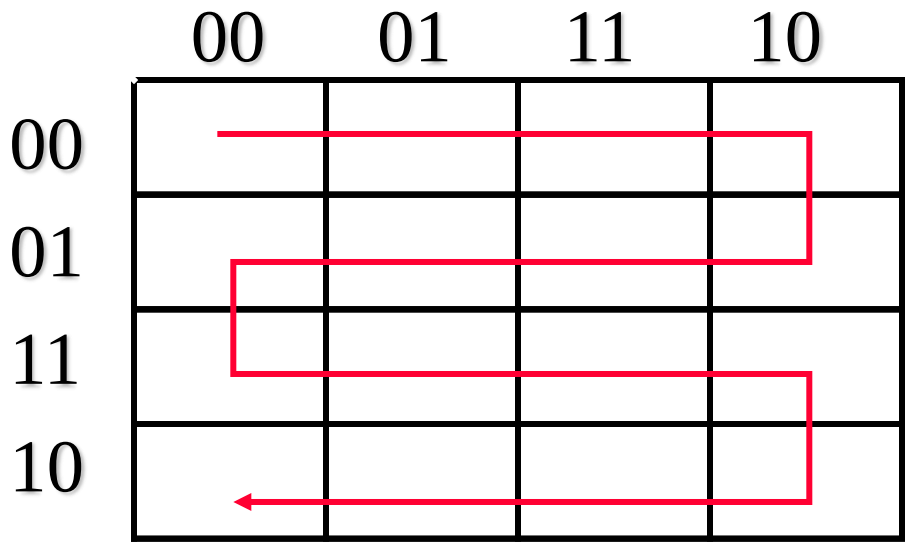
2位格雷码

00、01、11、10

3位格雷码

000、001、011、
010 、110、111、
101、100

Gray Code



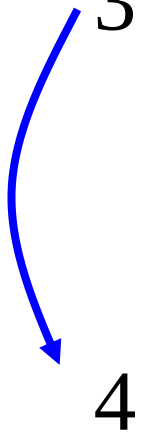
4位格雷码

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

Gray Code

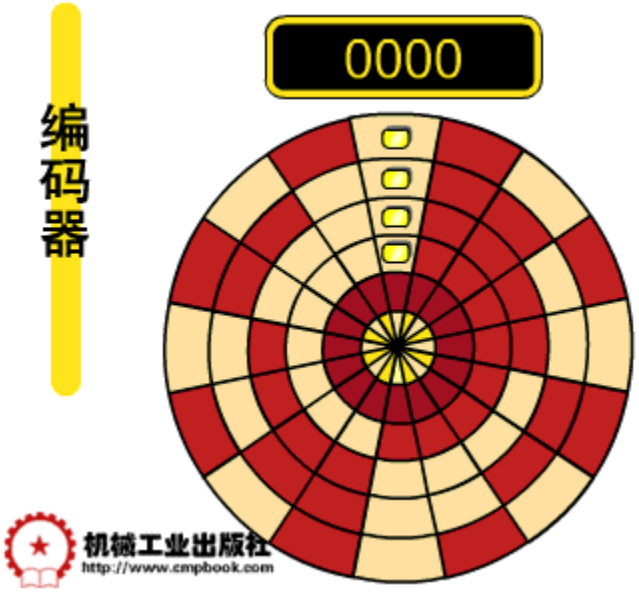
Example　十进制：3→4

	8421BCD	Gray Code
3	0011	0010
4	0100	0110
	3 位码元改变	1 位码元改变



- Gray Code ——连续变化时，比较可靠

Gray Code



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