数 字 电 路

Digital Circuits

02_数制与代码

张俊霞
zjx@ustc.edu.cn

内容提纲

- **数制及其转换**
- **二进制数算术运算**
- **二进制代码**

回顾：数据类型

- **数值型数据**
  - 整数
    - 有符号整数: **short, int, long**
    - 无符号整数: **unsigned short, ……**
  - 浮点数
    - 单精度：**float**
    - 双精度：**double**
- **字符型数据** **char**
- **逻辑型数据** **bool**

数制

- **数制：计数的体制，也称进位计数制**
  - 多位数码构成方式　– 低位到高位进位规则
- **十进制（Decimal）**
  - 十个数码：0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - 进位规则：逢十进一

  $151.75 = 1\times10^{2}+5\times10^{1}+1\times10^{0}+7\times10^{-1}+5\times10^{-2}$
- **任意进制**

$$(k_{n-1}\cdots k_1k_0 \bullet k_{-1}k_{-2}\cdots k_{-m})_R = \sum_{i=-m}^{n-1} k_i * R^i$$

基　权　MSD　LSD

二、八、十六进制

- **二进制 (Binary)**
  - R=2; k_i = 0, 1
- **十六进制 (Hexadecimal)**
  - R=16; k_i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- **八进制 (Octal)**
  - R=8; k_i = 0, 1, 2, 3, 4, 5, 6, 7

优点：电路实现简单、可靠
缺点：使用不方便

为克服二进制的缺点而引入

数制转换

- **任意进制 → 十进制**
  - 按位加权求和　$(N)_R = \sum_{i=-m}^{n-1} k_i * R^i$
- **十进制 → 任意进制**
  - 整数部分：除基取余，先得较低位
  - 小数部分：乘基取整，先得较高位
- **二、八、十六进制之间**
  - 二进制 → 八/十六进制：每3/4位分组转换
  - 八/十六进制 → 二进制：每位转换成3/4位二进制数码

示例—不同进制转换

- $(100101.101)_2$ =(37.625)$_{10}$ = (25.A)$_{16}$
- $(F15.6)_{16}$ =(3861.375)$_{10}$ = (7425.3)$_8$
- $(205)_8$ =(133)$_{10}$ = (10000101)$_2$
- $(45.6)_{10}$ =(101101.1001…)$_2$ = (2D.9…)$_{16}$

二进制的位权表

2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{20}	2^{30}
2	4	8	16	32	64	128	256	512	1K	1M	1G

回顾：运算类型

- 算术运算　+，−，*，/，%
- 关系运算　>，⩾，<，⩽，= =，！=
- 逻辑运算　&&，||，！
- 位运算　&，|，^，>>，<<

二进制数算术运算

- 加、减运算与十进制数运算规则类似，不同在于进位或借位规则
- 乘、除运算可以用加、减和移位运算实现
- 数据用补码表示，减法可转化为加法运算

加、减、乘、除算术运算
全部可以用加法和移位运算来实现

示例—无符号数算术运算

```
  1 0 1 0
+ 0 1 0 1
---------
  1 1 1 1
```

```
  1 0 1 0
- 0 1 0 1
---------
  0 1 0 1
```

```
    1 0 1 0
  × 0 1 0 1
  ---------
    1 0 1 0
  0 0 0 0
  1 0 1 0
0 0 0 0
---------
1 1 0 0 1 0
```

```
            1. 0 1 1
        ---------
111 ) 1 0 1 0
        1 1 1
      -------
        1 1 0 0
          1 1 1
        -------
          1 0 1 0
            1 1 1
          -------
            1 1 … … 余数
```


有符号数的表示

- 原码、反码和补码
  - 最高位表示符号：0-正数，1-负数
  - 余下位表示数值
- 对于正数，三种码相同，余下位=数值位
- 对于负数，三种码不同
  - 原码：余下位 = 数值位
  - 反码：余下位 = 取反(数值位)
  - 补码：余下位 = 取反(数值位) + 1

示例—原码、反码、补码

- 用4位二进制数码表示有符号数N

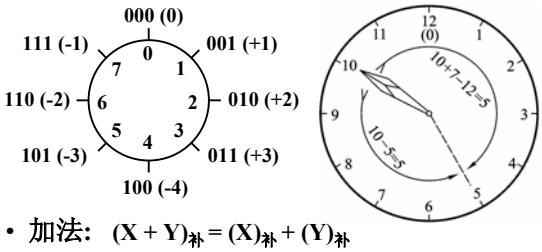
N	$(N)_{原}$	$(N)_{反}$	$(N)_{补}$
+ 10	0010	0010	0010
+ 0	0000	0000	0000
- 0	1000	1111	0000
-101	1101	1010	1011
-1000	X	X	1000

4位二进制数码的不同解释

二进制数码	无符号数	原码	补码	反码
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	-0	-8	-7
1001	9	-1	-7	-6
1010	10	-2	-6	-5
1011	11	-3	-5	-4
1100	12	-4	-4	-3
1101	13	-5	-3	-2
1110	14	-6	-2	-1
1111	15	-7	-1	-0

补码运算

- 采用补码，可以用加法来实现减法运算



- 加法：$(X+Y)_{补}=(X)_{补}+(Y)_{补}$
- 减法：$(X-Y)_{补}=(X)_{补}+(-Y)_{补}=(X)_{补}+\overline{(Y)_{补}}+1$

示例—补码运算

- $(X)_{补}$ = 1010 → $(-X)_{补}$ = ?　0110（+6）
- X = -5, Y = 3
  - → $(X+Y)_{补}$ = ?　1110（-2）
  - → $(X-Y)_{补}$ = ?　1000（-8）

溢出

- n位二进制补码表示范围：-2^{n-1} ～ $+2^{n-1}-1$
- 溢出：运算的结果超出了补码的表示范围
  - 可以通过位扩展，提高精度，解决该问题
- 出现场合
  - 同号相加，和的符号与被加数的相反
  - 异号相减，差的符号与被减数的相反
- 判别：最高数值位进位和符号位进位
  - 相同，则未溢出
  - 不相同，则溢出

示例—溢出判别

- 采用4位补码运算
- X = +5, Y = -7
  - → $(X+Y)_{补}$ = ?　1110（-2，正确）
  - → $(X-Y)_{补}$ = ?　1100（-4，溢出）
- X = -8, Y = -3
  - → $(X+Y)_{补}$ = ?　0101（+5，溢出）
  - → $(X-Y)_{补}$ = ?　1011（-5，正确）

二进制代码

- 用来表示不同事物的二进制数码
  - 编码：以一定的规则，编制代码的过程
  - 译码：将代码还原成所表示事物的过程
  - 码制：编制代码所要遵循的规则
- 二进制代码的位数n，与待编码事物的个数N之间应满足：$2^n \geqslant N$
- 常用代码：ASIIC码、BCD码、校验码、格雷码

ASCII码

- 美国信息交换标准代码
  - 7位二进制代码, 共可表示128个字符, 其中
  - 94个可显示和打印字符
  - 34个控制字符

低4位 \ 高3位	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000 0	NUL	DLE	SP	0	@	P	`	p
0001 1	SOH	DC1	!	1	A	Q	a	q
0010 2	STX	DC2	"	2	B	R	b	r
0011 3	ETX	DC3	#	3	C	S	c	s
0100 4	EOT	DC4	$	4	D	T	d	t
0101 5	ENQ	NAK	%	5	E	U	e	u
0110 6	ACK	SYN	&	6	F	V	f	v
0111 7	BEL	ETB	'	7	G	W	g	w
1000 8	BS	CAN	(	8	H	X	h	x
1001 9	HT	EM	)	9	I	Y	i	y
1010 A	LF	SUB	*	:	J	Z	j	z
1011 B	VT	ESC	+	;	K	[	k	{
1100 C	FF	FS	,	<	L	\	l	|
1101 D	CR	GS	-	=	M	]	m	}
1110 E	SO	RS	.	>	N	^	n	~
1111 F	SI	US	/	?	O	_	o	DEL

格雷码

- 编码顺序依次变化时，相邻代码仅有一位不同
- 最小和最大之间也只有一位不同，也称循环码

编码顺序	二进制码	格雷码	编码顺序	二进制码	格雷码
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

BCD码

- 二-十进制码(Binary-Coded Decimal)
  - 用4位二进制数码，来表示一位十进制数码
  - 有很多种方案，不同的方案得到不同的BCD码
- 常用BCD码
  - 有权码：8421码、5421码、2421码
  - 无权码：余3码、余3循环码

$(45.6)_{10}$ =(0100 0101.0110)$_{8421\ BCD}$

常用BCD码

十进制数	8421码	2421 码	5421 码	余3码	余3循环码
0	0000	0000	0000	0011	0010
1	0001	0001	0001	0100	0110
2	0010	0010	0010	0101	0111
3	0011	0011	0011	0110	0101
4	0100	0100	0100	0111	0100
5	0101	1011	1000	1000	1100
6	0110	1100	1001	1001	1101
7	0111	1101	1010	1010	1111
8	1000	1110	1011	1011	1110
9	1001	1111	1100	1100	1010

The End