

# 计算机组成与结构 —运算方法2

计算机科学与技术学院



# 温故 —关于数制表示

- 补码加减法的过程?
  - $[X + Y] \stackrel{?}{\Rightarrow} = [X] \stackrel{?}{\Rightarrow} + [Y] \stackrel{?}{\Rightarrow}$
  - $[X Y] \stackrel{?}{=} [X] \stackrel{?}{=} + [-Y] \stackrel{?}{=}$
- 简述双符号法溢出判断方法?
  - 符号位 S 变成 S<sub>2</sub>S<sub>1</sub>
  - 正数->00
  - 负数->11
  - 溢出判断条件: OF = S<sub>1</sub> ⊕ S<sub>2</sub>
- 简述半加器和全加器?
- 没有进位输入(Carry In)的加法器
- 包含进位输入(Carry In)的加法器



### 温故 —关于数制表示

- 简述行波进位加法器?
  - 自低位向高位,依次计算、进位,计算、进位
- 简述先行进位加法器?
- 各级的进位彼此是独立产生,通过输入数据A,B和C\_in可先行 获取进位C<sub>i</sub>,减小进位产生的延时



# 本节学习要点

- 原码乘法 (一位/两位)
- 补码乘法—booth
- 除法 (原码/补码)



# 定点数运算一原码乘法



#### 原码一位乘法就是模拟手工乘法:

• 符号 $S_m = S_A \oplus S_b$ ,数值 $|m| = |A| \times |B|$ 

例,[X]原=0.1101,[Y]原=1.1011,求积

解: 符号,  $S=0 \oplus 1=1$ ; 数值部分如下:

#### 模拟手工乘法





#### 原码一位乘法就是模拟手工乘法:

• 符号 $S_m = S_A \oplus S_b$ ,数值 $|m| = |A| \times |B|$ 

例,[X]原=0.1101,[Y]原=1.1011,求积

解: 符号,  $S=0 \oplus 1=1$ ; 数值部分如下:

#### 模拟手工乘法



·如何描述这一过程?

- 1. 设一临时变量D = 0
- **2**. 若Y末位为1, D = D + X
- 3. D右移一位; Y右移一位
- 4. 若Y有剩余位数,回到第2步
- 5. 结束



#### 原码一位乘法就是模拟手工乘法:

• 符号 $S_m = S_A \oplus S_b$ ,数值 $|m| = |A| \times |B|$ 

#### 例,[X]原=0.1101,[Y]原=1.1011,求积

解: 符号,  $S=0 \oplus 1=1$ ; 数值部分如下:

#### 模拟手工乘法

- 1. 设一临时变量D=0
- 2. 若Y末位为1, D = D + X
- 3. D右移一位; Y右移一位
- 4. 若Y有剩余位数,回到第2步
- 5. 结束

#### 通过临时变量D累加逐层结果

_			~=	_					2370	
			0	0	0	0	<-	临	时	变量D,初值为0
	第一层 加1101	+	1	1	0	1			V-	=101 <mark>1</mark> ;D = D + X
	<b>,,,</b>		1	1	0	1			T -	-1011,D = D + X
			0	1	1	0	1	b		D右移
	第二层 加1101	+	1	1	0	1	0		<b>V</b> -	=10 <mark>1</mark> ;D = D>>1 + X
	<b>,,,</b>	1	0	0	1	1	1		I -	-101,D = D>>1 + X
			1	0	0	1	1	1		D右移
4	第三层 加0000	+	0	0	0	0	0	0	,	Y=1 <mark>0</mark> ;D = D>>1 + X
	<b>,</b>		1	0	0	1	1	1		1-10,D-D>1+X
			0	1	0	0	1	1	1	D右移
	第四层 加1101	+	1	1	0	1	0	0	0	V-1·D - D>>1 ± V
		1	0	0	0	1	1	1	1	Y=1;D = D>>1 + X



#### 原码一位乘法就是模拟手工乘法:

• 符号 $S_m = S_A \oplus S_b$ , 数值 $|m| = |A| \times |B|$ 

例,[X]原=0.1101,[Y]原=1.1011,求积

解: 符号,  $S=0 \oplus 1=1$ ; 数值部分如下:

#### 模拟手工乘法

				1	1	0	1	
X				1	0	1	1	
				1	1	0	1	
			1	1	0	1		
		0	0	0	0			
100	1	1	0	1		4		
4	^	^	_	7			7	

- 1. 设一临时变量D=0
- 2. 若Y末位为1, D = D + X
- 3. D右移一位; Y右移一位
- 4. 若Y有剩余位数,回到第2步
- 5. 结束

		]	D				A	$ \mathbf{A}_0 $	操作
0	0	0	0	0	1	0	1	1	$A_0=1, +X$
+0	1	1	0	1	i/				
0	1	1	0	1	-				
0	0	1	1	0	1	1	0	1	→右移一次
+0	1	1	0	1		Y			$A_0=1, +X$
1	0	0	1	1	1	1	0	1	K
0	1	0	0	1	1	1	1	0	→右移一次
0	0	0	0	0		- 7			$A_0=0, +0$
0	1	0	0	1	1	1	1	0	
0	0	1	0	0	1	1	1	1	→右移一次
+0	1	1	0	1		J			$A_0=1, +X$
1	0	0	0	1	1	1	1	_1_	
0	1	0	0	0	1	1	1	1	→右移一次

#### 原码一位乘法运算过程

最终结果, [X]原x[Y]原=1.10001111



练习,[X]原=0.100111,[Y]原=1.100111,求积

结果为: 1.010111110001

1.	设一	-临时变量 $D$	= 0
----	----	-----------	-----

- 2. 若Y末位为1, D = D + X
- 3. D右移一位; Y右移一位
- 4. 若Y有剩余位数,回到第2步
- 5. 结束

				-			•		
				)			Α	A <sub>0</sub>	操作
00	0	0	0	0	0	0	l1 0 0 1 1	1	A <sub>0</sub> =1,+X
+00	1	0	0	1	1	1			
00	1	0	0	1	1	1			
00	0	1	0	0	1	1	1 1 0 0 1	1 /	右移一次
+00	1	0	0	1	1	1		1	$A_0 = 1, +X$
00	1	1	1	0	1	0	1 1 0 0 1	1	
00	0	1	1	1	0	1	0 1 1 0 0	1	右移一次
+00	1	0	0	1	1	1	ļ ļ		$A_0 = 1, +X$
01	0	0	0	1	0	0	0 1 1 0 0	1	/
00	1	0	0	0	1	0	0 0 1 1 0	0	右移一次
+00	0	0	0	0	0	0	I		$A_0 = 0, +0$
00	1	0	0	0	1	0	0 0 1 1 0	0	
00	0	1	0	0	0	1	0 0 0 1 1	0	右移一次
+00	0	0	0	0	0	0	I		A <sub>0</sub> =0,+0
00	0	1	0	0	0	1	0 0 0 1 1	0	
00	0	0	1	0	0	0	10001	1	右移一次
+00	1	0	0	1	1	1		77.7	$A_0 = 1, +X$
00	1	0	1	1	1	1	1 0 0 0 1	1	
00	0	1	0	1	1	1	1 1 0 0 0	1	右移一次



### 练习, [X]原=0.110111, [Y]原=1.1001, 求积 1.01 1110 1111

- 1. 设一临时变量D = 0
- **2**. **若**Y**末位为1**, D = D + X
- 3. D右移一位; Y右移一位
- 4. 若Y有剩余位数,回到第2步
- 5. 结束

			]	D					Y		操作
0 0	0	0	0	0	0	0	1	0	0	1	$A_0=1, +X$
0 0	1	1	0	1	1	1			Carrie	7	
0 0	1	1	0	1	1	1				1	
0 0	0	1	1	0	1	1	1	1	0	0	右移一次
0 0	0	0	1	1	0	1	1	1	1	0	右移一次
0 0	0	0	0	1	1	0	1	1	1	1	右移一次 A <sub>0</sub> =1, +X
0 0	1	1	0	1	1	1	ţi.c.	223		a	
0 0	1	1	1	1	0	1	1	1	1	1	
0 0	0	1	1	1	1	0	1	1	1	1	右移一次



#### 原码二位乘法,

一次乘两位数字? 这个在十进制下很反人类啊..., 但在二进制下, 似乎还好...



#### 原码二位乘法,

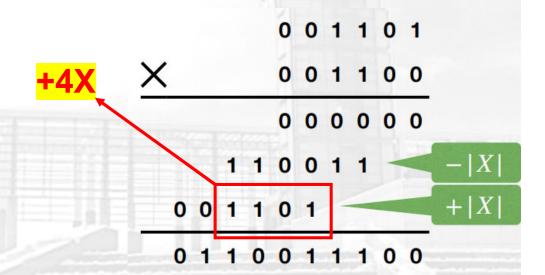
一次乘两位数字?这个在十进制下很反人类啊...,但在二进制下,似乎还好...

### 规则还可以更进一步优化:

$$3|X| = +4|X| - |X|$$

⇒先减|X|, 平移2位后加|X|

怎么做减法? 把X和D用补码表示





#### 规则还可以更进一步优化:

3|X| = +4|X| - |X|

⇒先减|X|,平移2位后加|X|

怎么做减法?把X和D用补码表示

### 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 0

#### 通过临时变量D累加逐层结果

		^			щн	در	Ca		73.	:/JF		一位二个
								<.	៕-	闹	变	量D,初值为0
	第一层 加0000	+ 0	0	0	0	0	0		•	/_	4 4	00.D = D + Y
	WHOOO			0					)	ſ <b>-</b>	1 1	00;D=D+X
		_	_	0	_	_	_		_			末两位为11
>	第二层 减1101	+ 1	1	0	0	1	1	0	0			Y=11;
	-			0								D = D >> 2 - X
												末两位为00,欠1
-	第三层 加1101	+ 0	0	1	1	0	1	0	0	0	0	D = D>>2 + X
		0	0	1	0	0	1	1	1	0	0	D - D//2 + X
												COLUMN TO SERVICE AND ADDRESS OF THE PARTY O

<<是左移操作, >>是右移操作



### 原码二位乘法

#### 规则还可以更进一步优化:

|3|X| = +4|X| - |X|

⇒先减|X|,平移2位后加|X|

怎么做减法?把X和D用补码表示

#### 补充的规则:

无论是否"欠"|X|, 一次最多+2|X|;

如果"当轮"+"欠的">2,继续"欠"下去~



### 标记是否要+4|X|

Y <sub>i+1</sub>	Y <sub>i</sub>	C	/	操	作
0	0	0	+0,	右移2次,	C=0
0	0	1	+ X ,	右移2次,	C=0
0	1	0	+ X ,	右移2次,	C=0
0	1	1	+2 X ,	右移2次,	C=0
1	0	0	+2 X ,	右移2次,	C=0
1	0	1	$- \mathbf{X} $ ,	右移2次,	C=1
1	1	0	$- \mathbf{X} $	右移2次,	C=1
1	1	1	+0,	右移2次,	C=1



### 原码二位乘法, 完整规则表

#### 标记是否进位

$Y_{i+1}$	Y <sub>i</sub>	C		操	作
0	0	0	+0,	右移2次,	C=0
0	0	1	+ X ,	右移2次,	C=0
0	1	0	+ X ,	右移2次,	C=0
0	1	1	+2 X ,	右移2次,	C=0
1	0	0	+2 X ,	右移2次,	C=0
1	0	1	$- \mathbf{X} $ ,	右移2次,	C=1
1	1	0	$- \mathbf{X} $ ,	右移2次,	C=1
× 1	1	1	+0,	右移2次,	C=1

例,[X]原=0.100111,[Y]原=1.100111,求积

解: [2X]补=01.001110; [-X]补=1.011001

ぞ	子号	立			L	)	-				4	A		操作
<b>0</b> 1	0 1	0 1	0	0 1	0 1	0	0 0	0	1	0	0	1	1 1	C=0 -X
1 1 0	1 1 0	1 1 1	0 1 0	1 1 0	1 0 1	0 1 I	0 1 1	1 0 0	0	1	1	0	0 1	C=1 →右移二次 C=1,+2X
0 0 0	0 0 0	1 0 1	0 0 0	0 1 0	0 0 1	1 0 1	0 0 1	0 1 0	0	0	0	1	1_0	C=0 →右移二次 C=0,+2X
0	0	1	0	1 1	1	1	1 1	1 1	l	1	0	0	0 1	C=0 →右移二次

图 3-9 例 3-12 的二位乘法的运算过程

原码两位乘法运算过程

最终结果,[X]原x[Y]原 =1.010111110001



练习, [X]原=0.110110, [Y]原=1.101101,

求积

解: [-X]补=1.001010;

[2X]补=01.101100;

结果为: 1.1001 0111 1110

V	V	C		柘	<i>U</i> ⊢
Y i+1	Y <sub>i</sub>	C		操 ————	作 —————
0	0	0	+0,	右移2次,	C=0
0	0	1	+ X ,	右移2次,	C=0
0	1	0	+  <b>X</b>  ,	右移2次,	C=0
0	1	1	+2 X ,	右移2次,	C=0
1	0	0	+2 X ,	右移2次,	C=0
1	0	1	$- \mathbf{X} $ ,	右移2次,	C=1
1	1	0	$- \mathbf{X} $ ,	右移2次,	C=1
1	1	1	+0,	右移2次,	C=1

	D	A	操作
0 0 0 0 0 0	0 0 0 0 0 0 1 1 0 1 1 0	1 0 1 1 0 1	C=0,+X
0 0 0 0 0 0 1 1 1	1 1 0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 0	1 0 1 1 <u>0 1</u> 1 0 1 0 <u>1 1</u>	右移两次 C=0,-X
1 1 1 1 1 1 1 1 1	0 1 0 1 1 1 1 1 0 1 0 1 0 0 1 0 1	1 0 1 0 <u>1 1</u> 1 1 1 0 <u>1 0</u>	右移两次 C=1,-X
1 1 0 1 1 1 0 0 0	1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0	1 1 1 0 <u>1 0</u> 1 1 1 1 1 0	右移两次 C=1,+X
000	1 0 0 1 0 1	1 1 1 1 1 0	



练习, [X]原=0.110110, [Y]原=1.101101,

求积

解: [-X]补=1.001010;

[2X]补=01.101100;

结果为: 1.1001 0111 1110

Y <sub>i</sub>	C		操	作
0	0	+0,	右移2次,	C=0
0	1	+  <b>X</b>  ,	右移2次,	C=0
1	0	+ X ,	右移2次,	C=0
1	1	+2 X ,	右移2次,	C=0
0	0	+2 X ,	右移2次,	C=0
0	1	$- \mathbf{X} $ ,	右移2次,	C=1
1	0	-  <b>X</b>  ,	右移2次,	C=1
1	1	+0,	右移2次,	C=1
	0 0 1 1 0 0	0 1 1 0	0 0 +0, 0 1 + X , 1 0 + X , 1 1 1 +2 X , 0 0 +2 X , 0 1 - X , 1 0 - X ,	0       0       +0, 右移2次,         0       1       + X , 右移2次,         1       0       + X , 右移2次,         1       1       +2 X , 右移2次,         0       0       +2 X , 右移2次,         0       1       - X , 右移2次,         1       0       - X , 右移2次,

				)					F	١,			操作
0 0 0	0	0	0	0	0	0	1	0	1	1	0	1	C=0,+X
000	1	1	0	1	1	0			1			999	
0 0 0	1	1	0	1	1	0	1	0	1	1	0	1	
0 0 0	0	0	1	1	0	1	1	0	1	0	1	1	右移两次
1 1 1	0	0	1	0	1	0							C=0,-X
111	0	1	0	1	1	1	1	0	1	0	1	1	
111	1	1	0	1	0	1	1	1	1	0	1	0	右移两次
111	0	0	1	0	1	0							C=1,-X
1 <mark>1 0</mark>	1	1	1	1	1	1	1	1	1	0	1	0	
1 1 1	1	0	1	1	1	1	1	1	1	1	1	0	右移两次
0 0 0	1	1	0	1	1	0					4		C=1,+X
0 0 0	1	0	0	1	0	1	1	1	1	1	1	0	





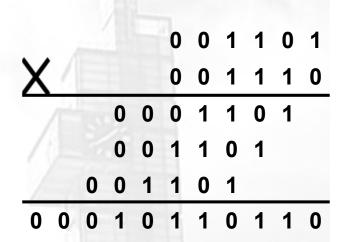
### 补码乘法

- 校正法
- Booth法
  - 两位Booth法



					0	0	1	1	0	1	
X					0	0	1	1	1	0	
					0	0	0	0	0	0	
				0	0	1	1	0	1		
			0	0	1	1	0	1			
		0	0	1	1	0	1				
	0	0	0	0	0	0					
0	0	0	0	0	0						
0	0	0	1	0	1	1	0	1	1	0	

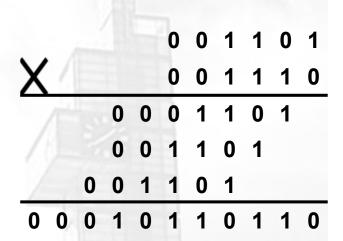
<—无脑计算需要6步 v.s 略用心机只需3步—>





					0	0	1	1	0	1	
X					0	0	1	1	1	0	
					0	0	0	0	0	0	'
				0	0	1	1	0	1		
			0	0	1	1	0	1			
		0	0	1	1	0	1				
	0	0	0	0	0	0					
0	0	0	0	0	0						
0	0	0	1	0	1	1	0	1	1	0	

<—无脑计算需要6步 v.s 略用心机只需3步—>



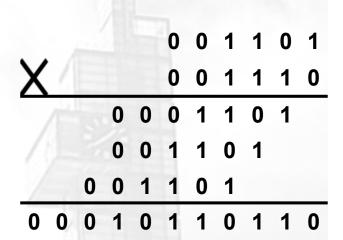
只关心Y为1的位



					0	0	1	1	0	1
X					0	0	1	1	1	0
					0	0	0	0	0	0
				0	0	1	1	0	1	
			0	0	1	1	0	1		
		0	0	1	1	0	1			
	0	0	0	0	0	0				
0	0	0	0	0	0					
0	0	0	1	0	1	1	0	1	1	0

<—无脑计算需要6步 v.s 略用心机只需3步—>

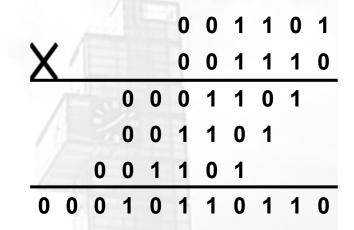
能不能更简化!?





						0	0	1	1	0	1	
X						0	0	1	1	1	0	
						0	0	0	0	0	0	•
					0	0	1	1	0	1		
				0	0	1	1	0	1			
			0	0	1	1	0	1				
	0	)	0	0	0	0	0					
0	0	)	0	0	0	0						
0	0	)	0	1	0	1	1	0	1	1	0	

### <—无脑计算需要6步 v.s 略用心机只需3步—>



### 能不能更简化!?

### 先看一个日常使用的技巧:

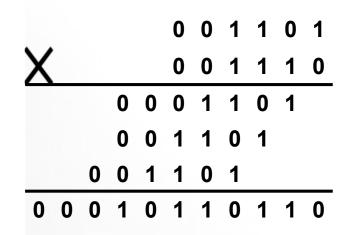
$$12345 \times 1001 = 12345 \times (1000 + 1)$$

$$2342 \times 999 = 2342 \times (1000 - 1)$$

#### 对二进制,也一样!

简化条件: 
$$1010 \times 0111 = 1010 \times (1000 - 1)$$
 存在"一串1"  $1010 \times 1110 = 1010 \times (10000 - 10)$ 



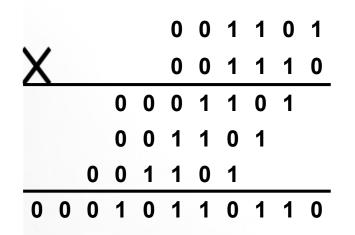


<一略用心机需要3步 v.s 运用技巧只需2步—>



问题来了:如何高效地识别"一串1"的开始和结束?





<---略用心机需要3步 v.s 运用技巧只需2步--->



问题来了:如何高效地识别"一串1"的开始和结束?

#### Booth算法:

- 从右向左,遇到"1 0"就是"一串1"的开始,在1的位置减|X|;
- 从右向左,遇到"0 1"就是"一串1"的结束,在0的位置m|X|;
- 中间的"00"和"11"都不管!

y <sub>i</sub>	<b>y</b> i-1	操作
0	0	部分积十0,右移1位
0	1	部分积+ <b>[X]<sub>补</sub>,右移1</b> 位
1	0	部分积+[-X] <sub>补</sub> ,右移 <b>1</b> 位
1	1	部分积十0,右移1位



### booth乘法

#### Booth算法:

- 从右向左,遇到"1 0"就是"一串1"的开始,在1的位置减|X|;
- 从右向左,遇到"0 1"就是"一串1"的结束,在0的位置m|X|;
- 中间的"00"和"11"都不管!

y <sub>i</sub>	<b>y</b> i-1	操作
0	0	部分积+0,右移1位
0	1	部分积+ <b>[X]<sub>补</sub>,右移1</b> 位
1	0	部分积+[-X] <sub>补</sub> ,右移 <b>1</b> 位
1	1	部分积十0,右移1位

①: booth乘法的乘数和被乘数还有结果都应由补码表示。

②: booth乘法计算前应在乘数末尾补零。 (防止末尾出现11的情况)

③: booth乘法的符号位参与计算。

④: booth乘法应以双符号位方式进行计算,防止结果溢出。



### 补码乘法

### **在此处减**|X|

y <sub>i</sub>	<b>y</b> <sub>i-1</sub>	操作
0	0	部分积+0,右移1位
0	1	部分积+ <b>[X]<sub>补</sub>,右移1</b> 位
1	0	部分积+[-X] <sub>补</sub> ,右移 <b>1</b> 位
1	1	部分积+0,右移1位

【例】X=0.1010, Y= -0.1101, 利用Booth法求积。

#### 【解】:

 $[X]_{ih} = 00.1010$ 

 $[-X]_{\stackrel{?}{k}} = 11.0110$ 

[Y]<sub>\*</sub>| = 11.0011

∴ [X·Y]ネト = 1.01111110

符	号			)				Α			A <sub>-1</sub>	操作说明
0	0	0	0	0	0	1	0	0	1	1	0	
_1	1	0	1	1	0			/	Ė			+[- <b>x</b> ] <sub>≱</sub>
1	1	0	1	1	0							
1	1	1	0	1	1	0	1	0	0	1	1	右移1位
0	0	0	0	0	0	1	2					+0
1	1	1	0	1	1			7				
1	1	1	1	0	1	1	0	1	0	0	1	右移1位
0	0	1	0	1	0			ľ				+[X] <sub>¾</sub> ,
0	0	0	1	1	1			1				在此处加 X
0	0	0	0	1	1	1	1	0	1	0	0	右移1位
0	0	0	0	0	0							+0
0	0	0	0	1	1							
0	0	0	0	0	1	1	1	1	0	1	0	右移1位
1	1	0	1	1	0							+[− <b>X</b> ] <sub>ネト</sub>
1	1	0	1	1	1	1	1	1	0	1	0	不移位



练习, [X]原=0.110110, [Y]原=1.101101

采用booth算法求积

等效: [X]原= 1.101101, [Y]原= 0.110110

解: [Y]补=00.110110;

[-X]\*\=00.101101

[X]补=11.010011

结果为: 1.0110 1000 0010

 $[XY]_{k} = 1.0110\ 1000\ 0010$ 

 $[XY]_{\overline{\bowtie}} = 1.0110\ 1000\ 0001$ 

[XY]<sub>原</sub>= 1.1001 0111 1110

符号	D	Α	<b>A</b> <sub>-1</sub>	操作说明
0 0	000000	0110110	0	
0 0	000000	0011011	0	右移1位
0 0	101101			+[- <b>x</b> ] <sub>≱⊦</sub>
0 0	101101	0011011	0	
0 0	0 1 0 1 1 0	1001101	1	右移1位
0 0	001011	0100110	1	右移1位
1 1	0 1 0 0 1 1			+ <b>[X]</b> <sub>补</sub>
1 1	0 1 1 1 1 0	0100110	1	
1 1	101111	0010011	0	右移1位
0 0	101101			+[- <b>x</b> ]≱⊦
0 0	0 1 1 1 0 0	0010011	0	
0 0	000111	0000100	1	右移2位
11	010011	1		+[X] <sub>≱⊦</sub>
1 1	0 1 1 0 1 0	0000100	1	MITTER
1 1	0 1 1 0 1 0	0000100	1	不移位