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]补 = [X]补 + [Y]补
  - [X - Y]补 = [X]补 + [-Y]补
- 简述双符号法溢出判断方法?
  - 符号位 S 变成 S_2S_1
  - • 正数->00
  - • 负数->11
  - 溢出判断条件： $OF = S_1 \oplus S_2$
- 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_i，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—原码乘法

定点数运算—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

$$
\begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 1 \\
\times & & & & 1 & 0 & 1 & 1 \\
\hline
 & & & & 1 & 1 & 0 & 1 \\
 & & & 1 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
 & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

如何描述这一过程?

定点数运算—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

$$
\begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 1 \\
\times & & & & 1 & 0 & 1 & 1 \\
\hline
 & & & & 1 & 1 & 0 & 1 \\
 & & & 1 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
 & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

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

定点数运算—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

$$\begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 1 \\
\times & & & & 1 & 0 & 1 & 1 \\
\hline
 & & & & 1 & 1 & 0 & 1 \\
 & & & 1 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
 & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}$$

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

第一层加1101	0 0 0 0	<-临时变量D，初值为0
	+ 1 1 0 1	Y=1011;D = D + X
	1 1 0 1	
第二层加1101	0 1 1 0 1	D右移
	+ 1 1 0 1 0	Y=101;D = D>>1 + X
	1 0 0 1 1 1	
第三层加0000	1 0 0 1 1 1	D右移
	+ 0 0 0 0 0 0	Y=10;D = D>>1 + X
	1 0 0 1 1 1	
第四层加1101	0 1 0 0 1 1 1	D右移
	+ 1 1 0 1 0 0 0	Y=1;D = D>>1 + X
	1 0 0 0 1 1 1 1	

<<是左移操作， >>是右移操作

定点数运算—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

				1	1	0	1
×				1	0	1	1
				1	1	0	1
			1	1	0	1	
		0	0	0	0		
	1	1	0	1			
1	0	0	0	1	1	1	1

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			A_0	操作
0	0	0	0	0	1	0	1	1	A_0=1, +X
+0	1	1	0	1					
0	1	1	0	1					
0	0	1	1	0	1	1	0	1	→右移一次
+0	1	1	0	1					A_0=1, +X
1	0	0	1	1	1	1	0	1	
0	1	0	0	1	1	1	1	0	→右移一次
0	0	0	0	0					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					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

定点数运算—乘法

练习，[X]原=0.100111, [Y]原=1.100111，求积

结果为：1. 010111110001

	D						A					A_0	操作
00	0	0	0	0	0	0	1	0	0	1	1	1	A_0=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							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							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							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							A_0=0,+0
00	0	1	0	0	0	1	0	0	0	1	1	0	
00	0	0	1	0	0	0	1	0	0	0	1	1	右移一次
+00	1	0	0	1	1	1							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

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

定点数运算—乘法

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					
0 0	1	1	0	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_0=1，+X
0 0	1	1	0	1	1	1					
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

定点数运算—乘法

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...

$$
\begin{array}{rl}
1\ 1\ 0\ 1 & \\
\times \qquad 1\ 0\ 0\ 1 & \\
\hline
1\ 1\ 0\ 1 & +|X| \\
1\ 1\ 0\ 1\ 0 & +2|X| \\
\hline
0\ 1\ 1\ 1\ 0\ 1\ 0\ 1 &
\end{array}
\qquad
\begin{array}{rl}
1\ 1\ 0\ 1 & \\
\times \qquad 1\ 1\ 0\ 0 & \\
\hline
0\ 0\ 0\ 0 & +0 \\
1\ 0\ 0\ 1\ 1\ 1 & +3|X| \\
\hline
1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 &
\end{array}
$$

规则似乎很直观：

$$
\begin{cases}
Y_{i+1}Yi = 00, +0 \\
Y_{i+1}Yi = 01, +|X| \\
Y_{i+1}Yi = 10, +2|X| \\
Y_{i+1}Yi = 11, +3|X|
\end{cases}
$$

定点数运算—乘法

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…

$$
\begin{array}{r l}
 & 1\ 1\ 0\ 1 \\
\times & 1\ 0\ 0\ 1 \\
\hline
 & 1\ 1\ 0\ 1 \quad +|X| \\
 1\ 1\ 0\ 1\ 0 & \quad +2|X| \\
\hline
0\ 1\ 1\ 1\ 0\ 1\ 0\ 1 &
\end{array}
\qquad
\begin{array}{r l}
 & 1\ 1\ 0\ 1 \\
\times & 1\ 1\ 0\ 0 \\
\hline
 & 0\ 0\ 0\ 0 \quad +0 \\
 1\ 0\ 0\ 1\ 1\ 1 & \quad +3|X| \\
\hline
1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 &
\end{array}
$$

规则似乎很直观：

$$
\begin{cases}
Y_{i+1}Yi = 00, +0 \\
Y_{i+1}Yi = 01, +|X| \\
Y_{i+1}Yi = 10, +2|X| \\
Y_{i+1}Yi = 11, +3|X|
\end{cases}
$$

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

$$
\begin{array}{r l}
 & 0\ 0\ 1\ 1\ 0\ 1 \\
\times & 0\ 0\ 1\ 1\ 0\ 0 \\
\hline
 & 0\ 0\ 0\ 0\ 0\ 0 \\
 1\ 1\ 0\ 0\ 1\ 1 & \quad -|X| \\
 0\ 0\ \boxed{1\ 1\ 0\ 1} & \quad +|X| \\
\hline
0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 &
\end{array}
$$

+4X

定点数运算—乘法

规则还可以更进一步优化：

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

⇒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
X         0 0 1 1 0 0
-----------------------
          0 0 0 0 0 0
      1 1 0 0 1 1
  0 0 1 1 0 1
-----------------------
  0 0 1 0 0 1 1 1 0 0
```

通过临时变量D累加逐层结果

第一层 加0000	0 0 0 0 0 0	<-临时变量D，初值为0
	+0 0 0 0 0 0	Y=1100;D = D + X
	0 0 0 0 0 0	
第二层 减1101	0 0 0 0 0 0 0 0	末两位为11
	+1 1 0 0 1 1 0 0	Y=11; D = D>>2 – X
	1 1 0 0 1 1 0 0	
第三层 加1101	1 1 1 1 0 0 1 1 0 0	末两位为00，欠1
	+0 0 1 1 0 1 0 0 0 0	D = D>>2 + X
	0 0 1 0 0 1 1 1 0 0	

<<是左移操作， >>是右移操作

定点数运算—乘法

原码二位乘法

规则还可以更进一步优化：

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

⇒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~

$\Rightarrow$

标记是否要$+4|X|$

Y_{i+1}	Y_i	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	−|X|，　右移2次，C=1
1	1	0	−|X|，　右移2次，C=1
1	1	1	+0，　右移2次，C=1

定点数运算—乘法

原码二位乘法，完整规则表

标记是否进位

Y_{i+1}	Y_i	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	－|X|，　右移2次，C＝1
1	1	0	－|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

符号位	D	A	操作
0 0 0	0 0 0 0 0 0	1 0 0 1 1 1	C=0
1 1 1	0 1 1 0 0 1		−X
1 1 1	0 1 1 0 0 1		C=1
1 1 1	1 1 0 1 1 0	0 1 1 0 0 1	→右移二次
0 0 1	0 0 1 1 1 0		C=1,+2X
0 0 1	0 0 0 1 0 0		C=0
0 0 0	0 1 0 0 0 1	0 0 0 1 1 0	→右移二次
0 0 1	0 0 1 1 1 0		C=0,+2X
0 0 1	0 1 1 1 1 1		C=0
0 0 0	0 1 0 1 1 1	1 1 0 0 0 1	→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

定点数运算—乘法

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_{i+1}	Y_i	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	－|X|，右移2次，C=1
1	1	0	－|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 0 1 1 0 1 0 1 1	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 0	1 0 1 0 1 1 1 1 1 0 1 0	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 1 0 1 1 1 1 1 0	右移两次 C=1,+X
0 0 0	1 0 0 1 0 1	1 1 1 1 1 0	

定点数运算—乘法

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_{i+1}	Y_i	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	−|X|，　右移2次，C=1
1	1	0	−|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 0 1 1 0 1 0 1 1	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 0	1 0 1 0 1 1 1 1 1 0 1 0	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 1 0 1 1 1 1 1 0	右移两次 C=1,+X
0 0 0	1 0 0 1 0 1	1 1 1 1 1 0	

定点数运算—补码乘法

定点数运算—补码乘法

补码乘法

- ~~校正法~~
- Booth法
  - ~~两位Booth法~~

定点数运算—补码乘法

					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	1	1	0	1	
			0	0	1	1	0	1		
		0	0	1	1	0	1			
0	0	0	1	0	1	1	0	1	1	0

定点数运算—补码乘法

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

只关心Y为1的位

定点数运算—补码乘法

```
        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—>

能不能更简化！？

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

定点数运算—补码乘法

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

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！

$$1010 \times 0111 = 1010 \times (1000 - 1)$$

$$1010 \times 1110 = 1010 \times (10000 - 10)$$

简化条件：

存在“一串1”

定点数运算—补码乘法

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

<—略用心机需要3步

v.s

运用技巧只需2步—>

在此处加$|X|$

在此处减$|X|$

```
                0 0 1 1 0 1
X               0 0 1 1 1 0
----------------------------
[-X]补  1 1 1 1 1 0 0 1 1
        0 0 1 1 0 1
----------------------------
        0 0 1 0 1 1 0 1 1 0
```

问题来了：如何高效地识别“一串1”的开始和结束？

定点数运算—补码乘法

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

<—略用心机需要3步

v.s

运用技巧只需2步—>

在此处加$|X|$　　在此处减$|X|$

```
                0 0 1 1 0 1
X               0 0 1 1 1 0
-------------------------------
[-X]补  1 1 1 1 1 0 0 1 1
        0 0 1 1 0 1
-------------------------------
        0 0 1 0 1 1 0 1 1 0
```

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的位置加$|X|$;
- 中间的“00”和“11”都不管!

y_i	y_{i-1}	操　作
0	0	部分积＋0，右移1位
0	1	部分积＋[X]补，右移1位
1	0	部分积＋[-X]补，右移1位
1	1	部分积＋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的位置加$|X|$；
- 中间的“00”和“11”都不管！

y_i	y_{i-1}	操　作
0	0	部分积＋0，右移1位
0	1	部分积＋$[X]_{补}$，右移1位
1	0	部分积＋$[-X]_{补}$，右移1位
1	1	部分积＋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

y_i	y_{i-1}	操　作
0	0	部分积＋0，右移1位
0	1	部分积＋$[X]_{补}$，右移1位
1	0	部分积＋$[-X]_{补}$，右移1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】：

$[X]_{补}$ ＝00.1010

$[-X]_{补}$ ＝11.0110

$[Y]_{补}$ ＝11.0011

∴ $[X\cdot Y]_{补}$ ＝1.01111110

在此处减|X|

符号	D	A	A_{-1}	操作说明
0 0	0 0 0 0	1 0 0 1 1	0	
1 1	0 1 1 0			＋$[-X]_{补}$
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			＋0
1 1	1 0 1 1			
1 1	1 1 0 1	1 0 1 0 0	1	右移1位
0 0	1 0 1 0			＋$[X]_{补}$
0 0	0 1 1 1			
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			＋$[-X]_{补}$
1 1	0 1 1 1	1 1 1 0 1	0	不移位

在此处加|X|

定点数运算—乘法

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]补=00.101101

[X]补=11.010011

结果为：1.0110 1000 0010

$[XY]_{补}$= 1.0110 1000 0010

$[XY]_{反}$= 1.0110 1000 0001

$[XY]_{原}$= 1.1001 0111 1110

符号	D	A	A_{-1}	操作说明
0 0	0 0 0 0 0 0	0 1 1 0 1 1 0	0	
0 0	0 0 0 0 0 0	0 0 1 1 0 1 1	0	右移1位
0 0	1 0 1 1 0 1			$+[-X]_{补}$
0 0	1 0 1 1 0 1	0 0 1 1 0 1 1	0	
0 0	0 1 0 1 1 0	1 0 0 1 1 0 1	1	右移1位
0 0	0 0 1 0 1 1	0 1 0 0 1 1 0	1	右移1位
1 1	0 1 0 0 1 1			$+[X]_{补}$
1 1	0 1 1 1 1 0	0 1 0 0 1 1 0	1	
1 1	1 0 1 1 1 1	0 0 1 0 0 1 1	0	右移1位
0 0	1 0 1 1 0 1			$+[-X]_{补}$
0 0	0 1 1 1 0 0	0 0 1 0 0 1 1	0	
0 0	0 0 0 1 1 1	0 0 0 0 1 0 0	1	右移2位
1 1	0 1 0 0 1 1			$+[X]_{补}$
1 1	0 1 1 0 1 0	0 0 0 0 1 0 0	1	
1 1	0 1 1 0 1 0	0 0 0 0 1 0 0	1	不移位