



计算机组成原理

第 5 讲

左德承

哈尔滨工业大学计算学部
容错与移动计算研究中心

2.3 定点运算

一、移位运算

1. 移位的意义

$$15.\text{ m} = 1500.\text{ cm}$$

小数点右移 2 位

机器用语 15 相对于小数点 左移 2 位

（ 小数点不动 ）

左移 绝对值扩大

右移 绝对值缩小

在计算机中，移位与加减配合，能够实现乘除运算

2. 算术移位规则

符号位不变

	码 制	添补代码
正数	原码、补码、反码	0
负数	原 码	0
	补 码	左移 添 0
		右移 添 1
	反 码	1

例6.16

2.3

设机器数字长为 8 位（含 1 位符号位），写出 $A = +26$ 时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = +26 = +11010$

则 $[A]_{\text{原}} = [A]_{\text{补}} = [A]_{\text{反}} = 0,0011010$

移位操作	机 器 数	对应的真值
	$[A]_{\text{原}} = [A]_{\text{补}} = [A]_{\text{反}}$	
移位前	0,0011010	+26
左移一位	0,0110100	+52
左移两位	0,1101000	+104
右移一位	0,0001101	+13
右移两位	0,0000110	+6

例6.17

2.3

设机器数字长为 8 位（含 1 位符号位），写出 $A = -26$ 时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = -26 = -11010$

原码	移位操作	机 器 数	对应的真值
	移位前	1,0011010	- 26
	左移一位	1,0110100	- 52
	左移两位	1,1101000	- 104
	右移一位	1,0001101	- 13
	右移两位	1,0000110	- 6

补码

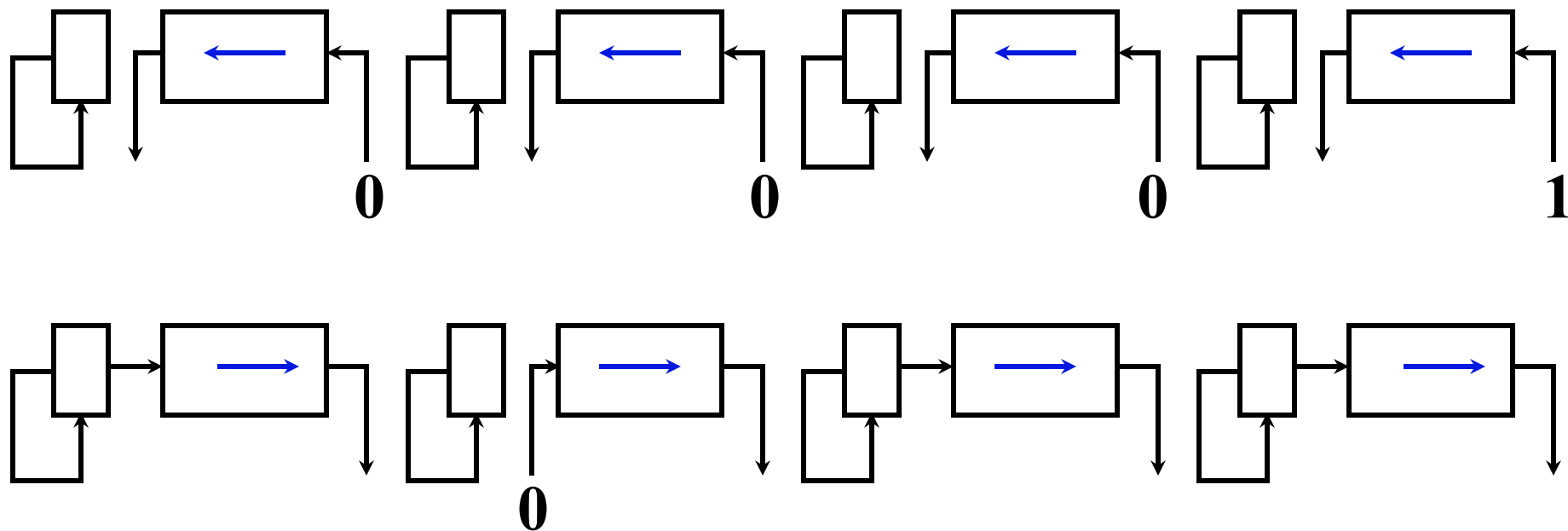
移位操作	机 器 数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

反码

移位操作	机 器 数	对应的真值
移位前	1,1100101	– 26
左移一位	1,1001011	– 52
左移两位	1,0010111	– 104
右移一位	1,1110010	– 13
右移两位	1,1111001	– 6

3. 算术移位的硬件实现

2.3



(a) 真值为正

(b) 负数的原码

(c) 负数的补码

(d) 负数的反码

← 丢 1 出错

出错

正确

正确

→ 丢 1 影响精度

影响精度

影响精度

正确

4. 算术移位和逻辑移位的区别

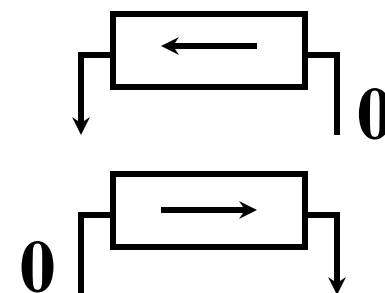
2.3

算术移位 有符号数的移位

逻辑移位 无符号数的移位

逻辑左移 低位添 0，高位移丢

逻辑右移 高位添 0，低位移丢



例如 **01010011**

10110010

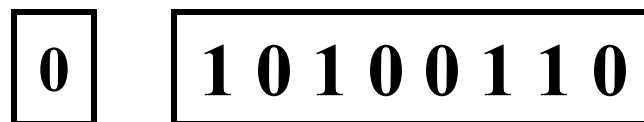
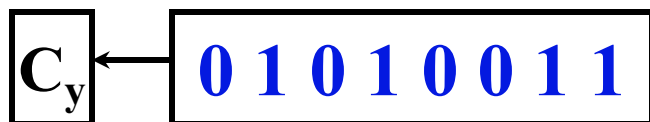
逻辑左移 **10100110**

逻辑右移 **01011001**

算术左移 **00100110**

算术右移 **11011001** (补码)

高位 1 移丢



二、加减法运算

2.3

1. 补码加减运算公式

(1) 加法

整数 $[A]_{\text{补}} + [B]_{\text{补}} = [A+B]_{\text{补}} \pmod{2^{n+1}}$

小数 $[A]_{\text{补}} + [B]_{\text{补}} = [A+B]_{\text{补}} \pmod{2}$

(2) 减法

$$A-B = A+(-B)$$

整数 $[A-B]_{\text{补}} = [A+(-B)]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2^{n+1}}$

小数 $[A-B]_{\text{补}} = [A+(-B)]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2}$

连同符号位一起相加，符号位产生的进位自然丢掉

2. 举例

2.3

例 6.18 设 $A = 0.1011$, $B = -0.0101$

求 $[A + B]_{\text{补}}$

验证

$$\begin{array}{rcl} \text{解: } [A]_{\text{补}} & = & 0.1011 \\ + [B]_{\text{补}} & = & 1.1011 \\ \hline [A]_{\text{补}} + [B]_{\text{补}} & = & 10.0110 = [A + B]_{\text{补}} \\ \therefore A + B & = & 0.0110 \end{array}$$

$$\begin{array}{r} 0.1011 \\ - 0.0101 \\ \hline 0.0110 \end{array}$$

例 6.19 设 $A = -9$, $B = -5$

求 $[A + B]_{\text{补}}$

验证

$$\begin{array}{rcl} \text{解: } [A]_{\text{补}} & = & 1, 0111 \\ + [B]_{\text{补}} & = & 1, 1011 \\ \hline [A]_{\text{补}} + [B]_{\text{补}} & = & 11, 0010 = [A + B]_{\text{补}} \\ \therefore A + B & = & -1110 \end{array}$$

$$\begin{array}{r} -1001 \\ + -0101 \\ \hline -1110 \end{array}$$

例 6.20 设机器数字长为 8 位（含 1 位符号位） **2.3**

且 $A = 15$, $B = 24$, 用补码求 $A - B$

解: $A = 15 = 0001111$

$B = 24 = 0011000$

$[A]_{\text{补}} = 0, 0001111$ $[B]_{\text{补}} = 0, 0011000$

$+ [-B]_{\text{补}} = 1, 1101000$

$[A]_{\text{补}} + [-B]_{\text{补}} = 1, 1110111 = [A - B]_{\text{补}}$

$\therefore A - B = -1001 = -9$

练习 1 设 $x = \frac{9}{16}$ $y = \frac{11}{16}$, 用补码求 $x+y$

$x + y = -0.1100 = -\frac{12}{16}$ 错

练习 2 设机器数字长为 8 位（含 1 位符号位）

且 $A = -97$, $B = +41$, 用补码求 $A - B$

$A - B = +1110110 = +118$ 错

3. 溢出判断

2.3

(1) 一位符号位判溢出

参加操作的 **两个数**（减法时即为被减数和“求补”以后的减数）**符号相同，其结果的符号与原操作数的符号不同，即为溢出**

硬件实现

最高有效位的进位 \oplus 符号位的进位 = 1 溢出

如

$1 \oplus 0 = 1$	} 有 溢出
$0 \oplus 1 = 1$	
$0 \oplus 0 = 0$	} 无 溢出
$1 \oplus 1 = 0$	

(2) 两位符号位判溢出

2.3

$$[x]_{\text{补}} = \begin{cases} x & 1 > x \geq 0 \\ 4 + x & 0 > x \geq -1 \pmod{4} \end{cases}$$

$$[x]_{\text{补}} + [y]_{\text{补}} = [x + y]_{\text{补}} \pmod{4}$$

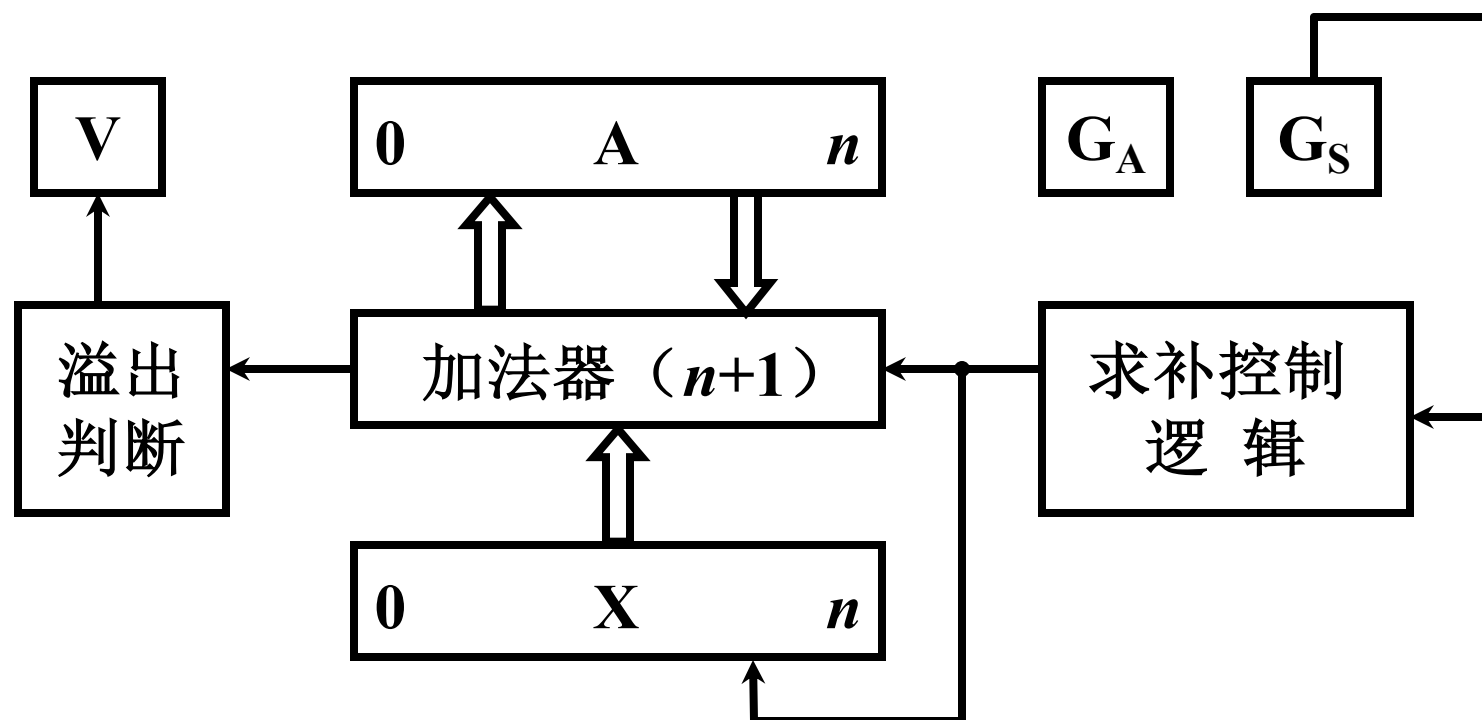
$$[x - y]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}} \pmod{4}$$

结果的双符号位 相同	未溢出	00, ×××××
		11, ×××××

结果的双符号位 不同	溢出	10, ×××××
		01, ×××××

最高符号位 代表其 真正的符号

4. 补码加减法的硬件配置



A、X 均 $n+1$ 位

用减法标记 G_S 控制求补逻辑

三、乘法运算

1. 分析笔算乘法

$$A = -0.1101 \quad B = 0.1011$$

$$A \times B = -0.10001111 \quad \text{乘积的符号心算求得}$$

$$\begin{array}{r}
 0.1101 \\
 \times 0.1011 \\
 \hline
 1101 \\
 1101 \\
 0000 \\
 1101 \\
 \hline
 0.10001111
 \end{array}$$

✓ 符号位单独处理

✓ 乘数的某一位决定是否加被乘数

? 4个位积一起相加

✓ 乘积的位数扩大一倍

2. 笔算乘法改进

$$A \cdot B = A \cdot 0.1011$$

$$= 0.1A + 0.00A + 0.001A + 0.0001A$$

$$= 0.1A + 0.00A + 0.001(A + 0.1A)$$

$$= 0.1A + 0.01[0 \cdot A + 0.1(A + 0.1A)]$$

$$= 0.1\{A + 0.1[0 \cdot A + 0.1(A + 0.1A)]\}$$

右移一位

$$= 2^{-1}\{1 \cdot A + 2^{-1}[0 \cdot A + 2^{-1}(1 \cdot A + 2^{-1}(1 \cdot A + 0))]\}$$

第一步 被乘数 $A + 0$

第二步 右移一位，得新的部分积

第三步 部分积 + 被乘数

⋮

第八步 右移一位，得结果

①

②

③

⑧

3. 改进后的笔算乘法过程（竖式） 2.3

部分积	乘数	说明
$\begin{array}{r} 0.0000 \\ + 0.1101 \\ \hline \end{array}$	$101\underline{1}$	初态，部分积 = 0 乘数为 1，加被乘数
$\begin{array}{r} 0.1101 \\ 0.0110 \\ + 0.1101 \\ \hline \end{array}$	$110\underline{1}$	→ 1，形成新的部分积 乘数为 1，加被乘数
$\begin{array}{r} 1.0011 \\ 0.1001 \\ + 0.0000 \\ \hline \end{array}$	$11\underline{10}$	→ 1，形成新的部分积 乘数为 0，加 0
$\begin{array}{r} 0.1001 \\ 0.0100 \\ + 0.1101 \\ \hline \end{array}$	$111\underline{1}$	→ 1，形成新的部分积 乘数为 1，加被乘数
$\begin{array}{r} 1.0001 \\ 0.1000 \\ \hline \end{array}$	1111	→ 1，得结果

- 乘法 运算可用 加和移位实现
 $n = 4$ ，加 4 次，移 4 次
 - 由乘数的末位决定被乘数是否与原部分积相加，
然后 \rightarrow 1 位形成新的部分积，同时 乘数 \rightarrow 1 位
(末位移丢)，空出高位存放部分积的低位。
 - 被乘数只与部分积的高位相加
- 硬件 3 个寄存器，具有移位功能
 1 个全加器

4. 原码乘法

(1) 原码一位乘运算规则

以小数为例

$$\text{设 } [x]_{\text{原}} = x_0 \cdot x_1 x_2 \cdots x_n$$

$$[y]_{\text{原}} = y_0 \cdot y_1 y_2 \cdots y_n$$

$$\begin{aligned} [x \cdot y]_{\text{原}} &= (x_0 \oplus y_0) \cdot (0 \cdot x_1 x_2 \cdots x_n)(0 \cdot y_1 y_2 \cdots y_n) \\ &= (x_0 \oplus y_0) \cdot x^* y^* \end{aligned}$$

式中 $x^* = 0 \cdot x_1 x_2 \cdots x_n$ 为 x 的绝对值

$y^* = 0 \cdot y_1 y_2 \cdots y_n$ 为 y 的绝对值

乘积的符号位单独处理 $x_0 \oplus y_0$

数值部分为绝对值相乘 $x^* \cdot y^*$

(2) 原码一位乘递推公式

$$\begin{aligned}
 x^* \cdot y^* &= x^*(0.y_1y_2 \dots y_n) \\
 &= x^*(y_12^{-1} + y_22^{-2} + \dots + y_n2^{-n}) \\
 &= 2^{-1}(y_1x^* + 2^{-1}(y_2x^* + \dots 2^{-1}(y_nx^* + \underbrace{0}_{z_0}) \dots)) \\
 &\quad \underbrace{\hspace{10em} \dots \hspace{10em} z_1}_{z_n}
 \end{aligned}$$

$$z_0 = 0$$

$$z_1 = 2^{-1}(y_nx^* + z_0)$$

$$z_2 = 2^{-1}(y_{n-1}x^* + z_1)$$

$$\vdots$$

$$z_n = 2^{-1}(y_1x^* + z_{n-1})$$

例6.21 已知 $x = -0.1110$ $y = 0.1101$ 求 $[x \cdot y]_{\text{原}}$

2.3

解：

数值部分的运算

部分积	乘数	说明
0.0000 + 0.1110	110 <u>1</u>	部分积 初态 $z_0 = 0$ + x^*
逻辑右移 0.1110 0.0111 + 0.0000	011 <u>0</u>	$\rightarrow 1$, 得 z_1 + 0
逻辑右移 0.0111 0.0011 + 0.1110	0 101 <u>1</u>	$\rightarrow 1$, 得 z_2 + x^*
逻辑右移 1.0001 0.1000 + 0.1110	10 110 <u>1</u>	$\rightarrow 1$, 得 z_3 + x^*
逻辑右移 1.0110 0.1011	110 0110	$\rightarrow 1$, 得 z_4

例6.21 结果

2.3

① 乘积的符号位 $x_0 \oplus y_0 = 1 \oplus 0 = 1$

② 数值部分按绝对值相乘

$$x^* \cdot y^* = 0.10110110$$

$$\text{则 } [x \cdot y]_{\text{原}} = 1.10110110$$

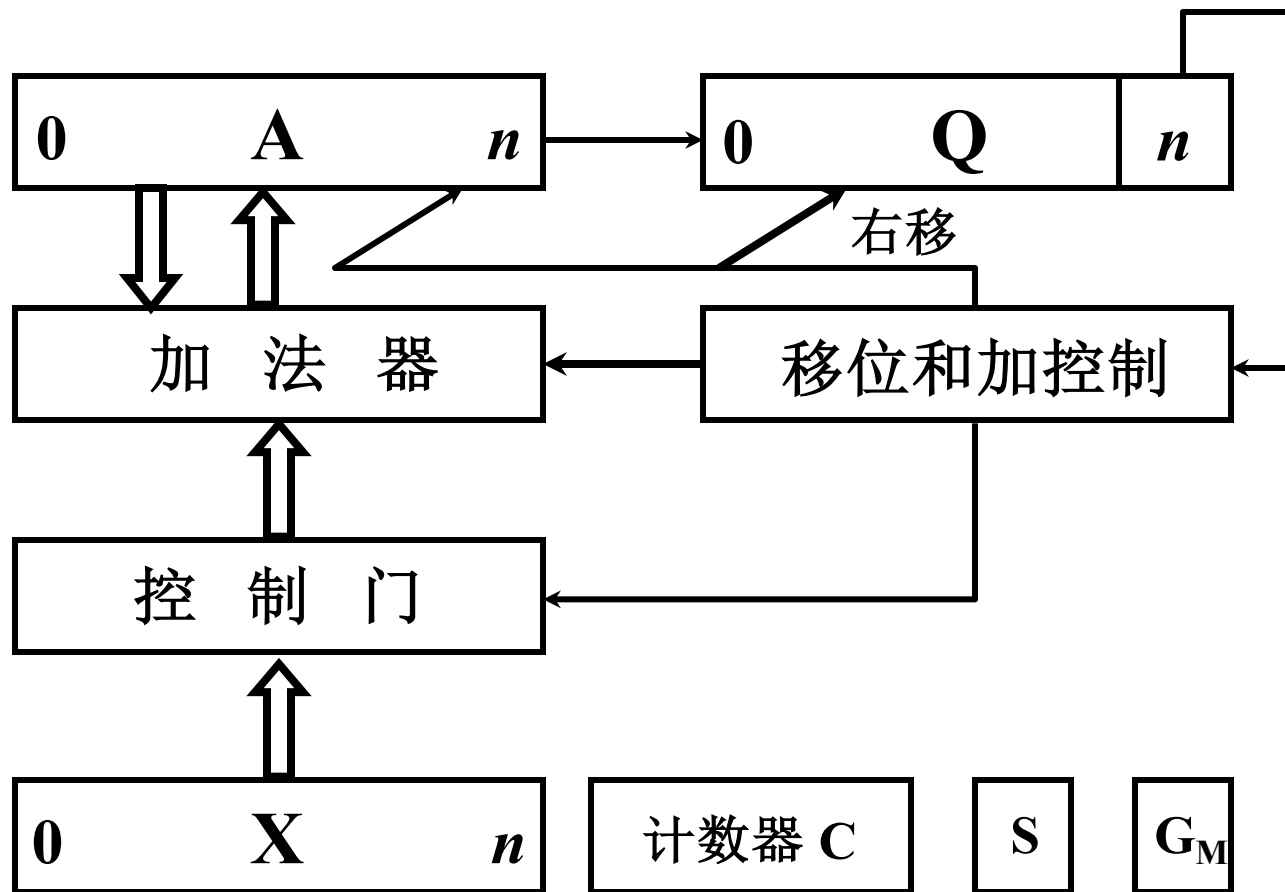
特点 绝对值运算

用移位的次数判断乘法是否结束

逻辑移位

(3) 原码一位乘的硬件配置

2.3



A、X、Q 均 $n+1$ 位

移位和加受末位乘数控制

5. 补码乘法

2.3

(1) 补码一位乘运算规则

以小数为例 设被乘数 $[x]_{\text{补}} = x_0.x_1x_2 \cdots x_n$
乘数 $[y]_{\text{补}} = y_0.y_1y_2 \cdots y_n$

① 被乘数任意，乘数为正

同原码乘 但加和移位按补码规则运算
乘积的符号自然形成

② 被乘数任意，乘数为负

乘数 $[y]_{\text{补}}$ ，去掉符号位，操作同 ①

最后加 $[-x]_{\text{补}}$ ，校正

③ Booth 算法（被乘数、乘数符号任意） 2.3

设 $[x]_{\text{补}} = x_0.x_1x_2 \cdots x_n$ $[y]_{\text{补}} = y_0.y_1y_2 \cdots y_n$

$[x \cdot y]_{\text{补}}$

$-[x]_{\text{补}} = +[-x]_{\text{补}}$

$$= [x]_{\text{补}} (0.y_1 \cdots y_n) - [x]_{\text{补}} \cdot y_0$$

$$= [x]_{\text{补}} (y_1 2^{-1} + y_2 2^{-2} + \cdots + y_n 2^{-n}) - [x]_{\text{补}} \cdot y_0$$

$$2^{-1} = 2^0 - 2^{-1}$$

$$= [x]_{\text{补}} (-y_0 + y_1 2^{-1} + y_2 2^{-2} + \cdots + y_n 2^{-n})$$

$$2^{-2} = 2^{-1} - 2^{-2}$$

$$= [x]_{\text{补}} [-y_0 + (y_1 - y_1 2^{-1}) + (y_2 2^{-1} - y_2 2^{-2}) + \cdots + (y_n 2^{-(n-1)} - y_n 2^{-n})]$$

$$= [x]_{\text{补}} [(y_1 - y_0) + (y_2 - y_1) 2^{-1} + \cdots + (y_n - y_{n-1}) 2^{-(n-1)} + (0 - y_n) 2^{-n}]$$

$$= [x]_{\text{补}} [(y_1 - y_0) + (y_2 - y_1) 2^{-1} + \cdots + (y_{n+1} - y_n) 2^{-n}]$$

附加位 y_{n+1}

$$y'_1 2^{-1} + \cdots + y'_n 2^{-n}$$

④ Booth 算法递推公式

2.3

$$[z_0]_{\text{补}} = 0$$

$$[z_1]_{\text{补}} = 2^{-1} \{ (y_{n+1} - y_n) [x]_{\text{补}} + [z_0]_{\text{补}} \} \quad y_{n+1} = 0$$

⋮

$$[z_n]_{\text{补}} = 2^{-1} \{ (y_2 - y_1) [x]_{\text{补}} + [z_{n-1}]_{\text{补}} \}$$

$$[x \cdot y]_{\text{补}} = [z_n]_{\text{补}} + (y_1 - y_0) [x]_{\text{补}}$$

最后一步不移位

如何实现
 $y_{i+1} - y_i$?

y_i	y_{i+1}	$y_{i+1} - y_i$	操作
0	0	0	$\rightarrow 1$
0	1	1	$+ [x]_{\text{补}} \rightarrow 1$
1	0	-1	$+ [-x]_{\text{补}} \rightarrow 1$
1	1	0	$\rightarrow 1$

例6.23 已知 $x = +0.0011$ $y = -0.1011$ 求 $[x \cdot y]_{\text{补}}$ **2.3**

解:
$$\begin{array}{r|l|l} 00.0000 & 1.010\underline{1} & 0 \\ + 11.1101 & & +[-x]_{\text{补}} \end{array}$$

补码
右移

$$\begin{array}{r|l|l} 11.1101 & & \\ 11.\underline{1}110 & 1 & 101\underline{0} & 1 & \rightarrow 1 \\ + 00.0011 & & & & +[x]_{\text{补}} \end{array}$$

补码
右移

$$\begin{array}{r|l|l} 00.0001 & 1 & \\ 00.\underline{0}000 & 11 & 10\underline{1} & 0 & \rightarrow 1 \\ + 11.1101 & & & & +[-x]_{\text{补}} \end{array}$$

补码
右移

$$\begin{array}{r|l|l} 11.1101 & 11 & \\ 11.\underline{1}110 & 111 & 1\underline{0} & 1 & \rightarrow 1 \\ + 00.0011 & & & & +[x]_{\text{补}} \end{array}$$

补码
右移

$$\begin{array}{r|l|l} 00.0001 & 111 & \\ 00.\underline{0}000 & 1111 & \underline{1} & 0 & \rightarrow 1 \\ + 11.1101 & & & & +[-x]_{\text{补}} \end{array}$$

$$\begin{array}{r|l|l} 11.1101 & 1111 & & \text{最后一步不移位} \end{array}$$

$$[x]_{\text{补}} = 0.0011$$

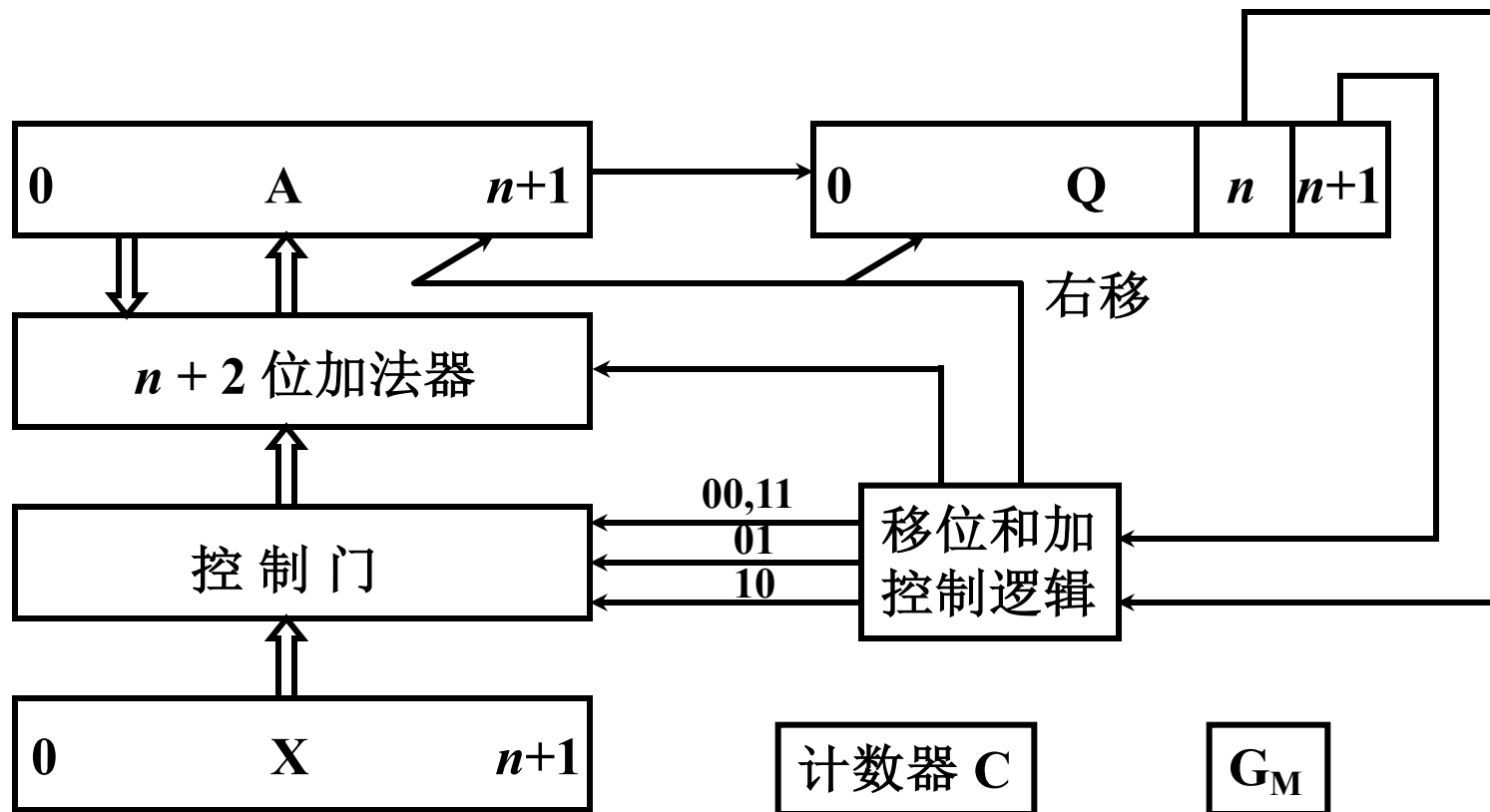
$$[y]_{\text{补}} = 1.0101$$

$$[-x]_{\text{补}} = 1.1101$$

$$\therefore [x \cdot y]_{\text{补}} = 1.11011111$$

(2) Booth 算法的硬件配置

2.3



A、X、Q 均 $n+2$ 位

移位和加法操作受乘数末两位控制

累加器乘法小结

- 整数乘法与小数乘法完全相同
可用 逗号 代替小数点
- 原码乘 符号位 单独处理
补码乘 符号位 自然形成
- 原码乘去掉符号位运算 即为无符号数乘法
- 不同的乘法运算需有不同的硬件支持