计算机组成原理

第 6 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

5. 补码乘法

(1) 补码一位乘运算规则

以小数为例　设 被乘数　$[x]_{补} = x_0.x_1x_2 \cdots x_n$

乘数　$[y]_{补} = y_0.y_1y_2 \cdots y_n$

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$[y]_{补}$，去掉符号位，操作同 ①

最后 加$[-x]_{补}$，校正

③ Booth 算法（被乘数、乘数符号任意）2.3

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

$[x \cdot y]_{补}$

$-[x]_{补} = +[-x]_{补}$

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

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

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

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

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

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

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

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

附加位 y_{n+1}

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

④ Booth 算法递推公式

$[z_0]_{补} = 0$

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

$\vdots$

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

$[x \cdot y]_{补} = [z_n]_{补}+(y_1-y_0)[x]_{补}$ 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	→1
0 1	1	$+[x]_{补}$ →1
1 0	-1	$+[-x]_{补}$→1
1 1	0	→1

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

解：

00.0000	1.0101 0	
+ 11.1101		$+[-x]_{补}$
11.1101		
11.1110	1 1010 1	→1
+ 00.0011		$+[x]_{补}$
00.0001	1	
00.0000	11 1010	→1
+ 11.1101		$+[-x]_{补}$
11.1101	11	
11.1110	111 10 1	→1
+ 00.0011		$+[x]_{补}$
00.0001	111	
00.0000	1111 1 0	→1
+ 11.1101		$+[-x]_{补}$
11.1101	1111	最后一步不移位

补码右移

补码右移

补码右移

补码右移

$[x]_{补} = 0.0011$

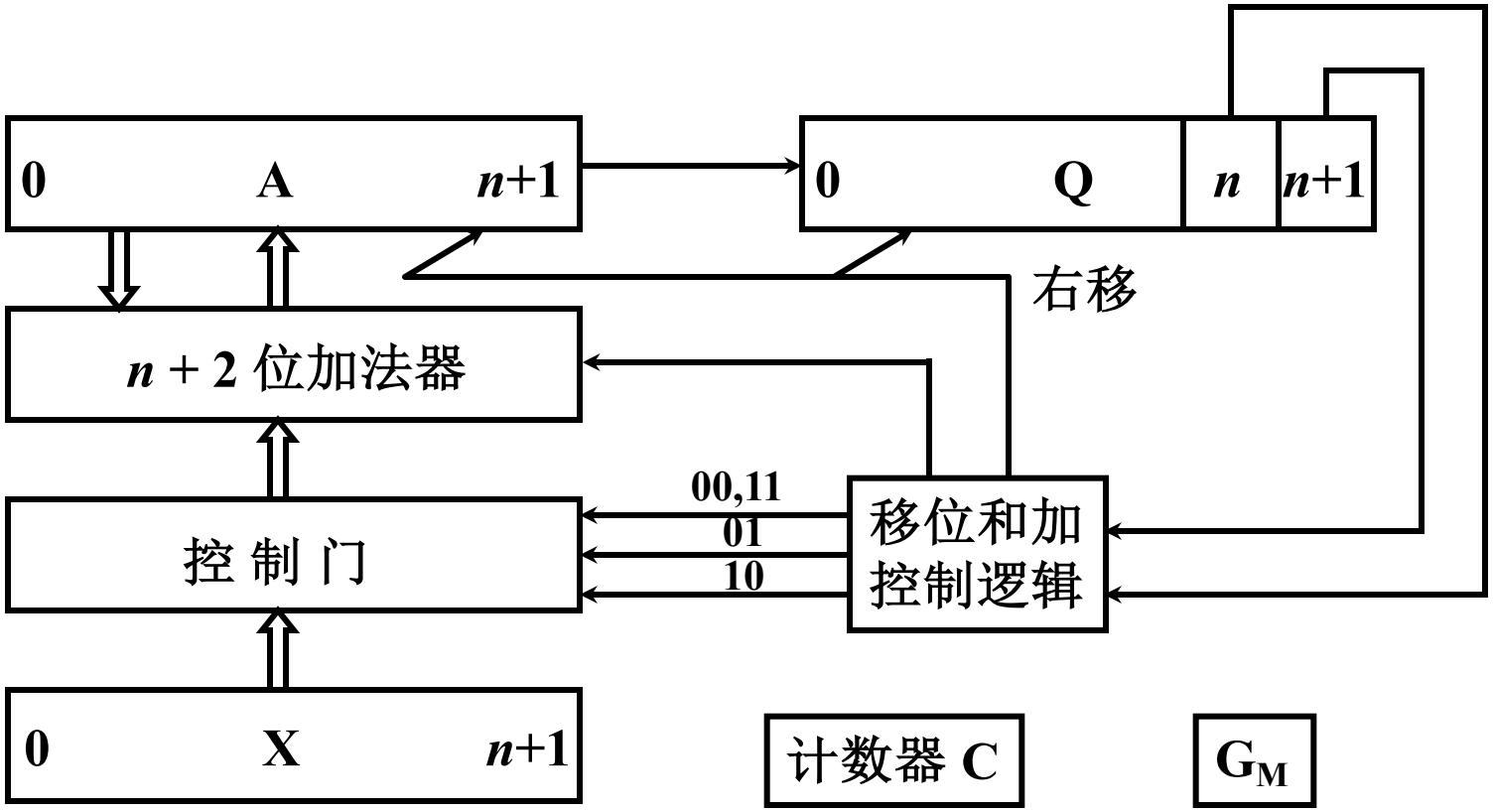
$[y]_{补} = 1.0101$

$[-x]_{补} = 1.1101$

$\therefore [x \cdot y]_{补}$

$= 1.11011111$

(2) Booth 算法的硬件配置



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

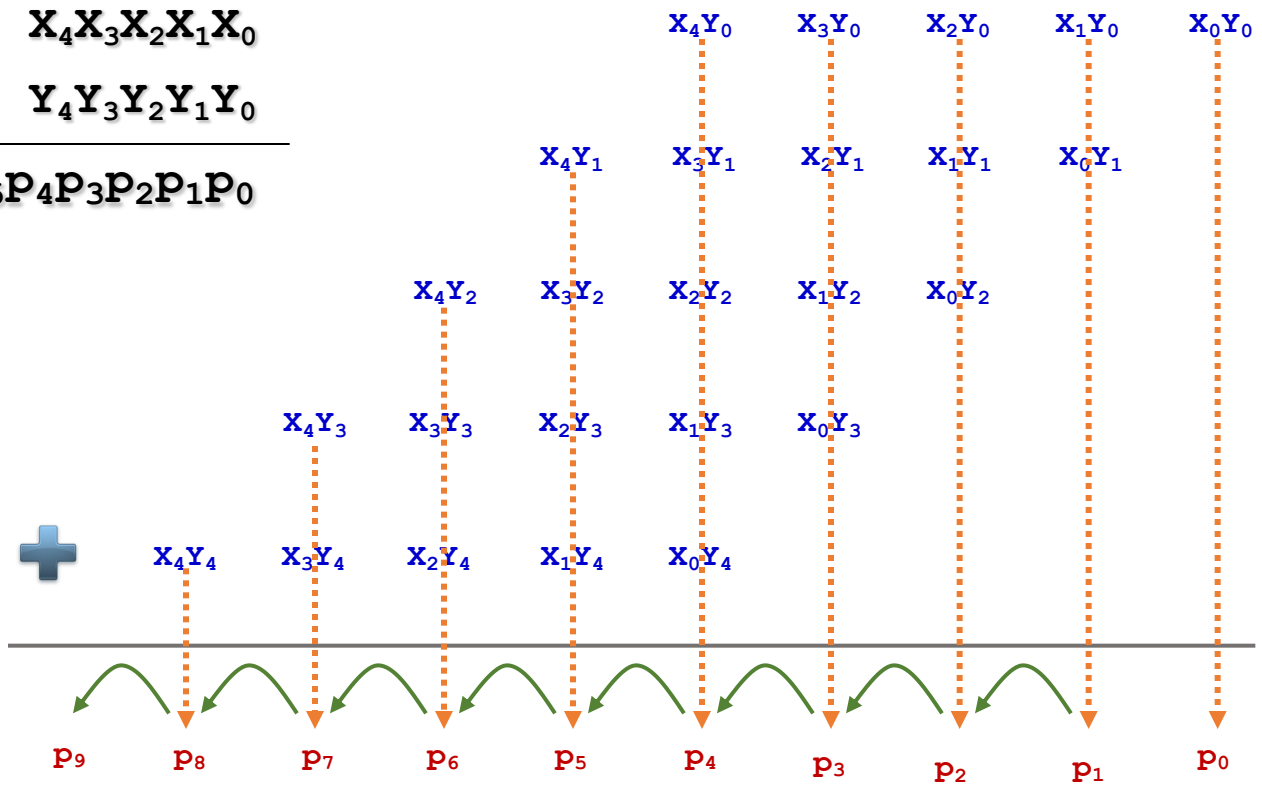
- 不同的乘法运算需有不同的硬件支持

乘法运算实现方法

- 执行乘法运算子程序实现乘法运算
  - 零成本、Intel 8008/8080、RISCV32-I指令集
- 利用加法器多次累加实现乘法运算
  - 原码一位乘法的运算方法与逻辑实现
  - 补码一位乘法的运算方法与逻辑实现
  - 成本低
- 设置专用乘法器实现乘法运算
  - 成本高 （原码、补码乘法器）

二进制手工乘法运算

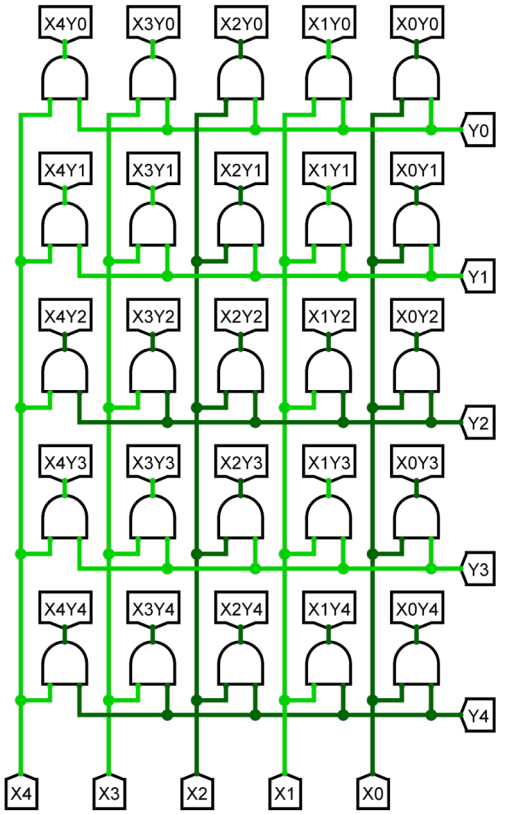
$$
\begin{array}{rr}
 & X_4X_3X_2X_1X_0 \\
\times & Y_4Y_3Y_2Y_1Y_0 \\
\hline
 & P_9P_8P_7P_6P_5P_4P_3P_2P_1P_0
\end{array}
$$



先计算相加数，然后逐列相加

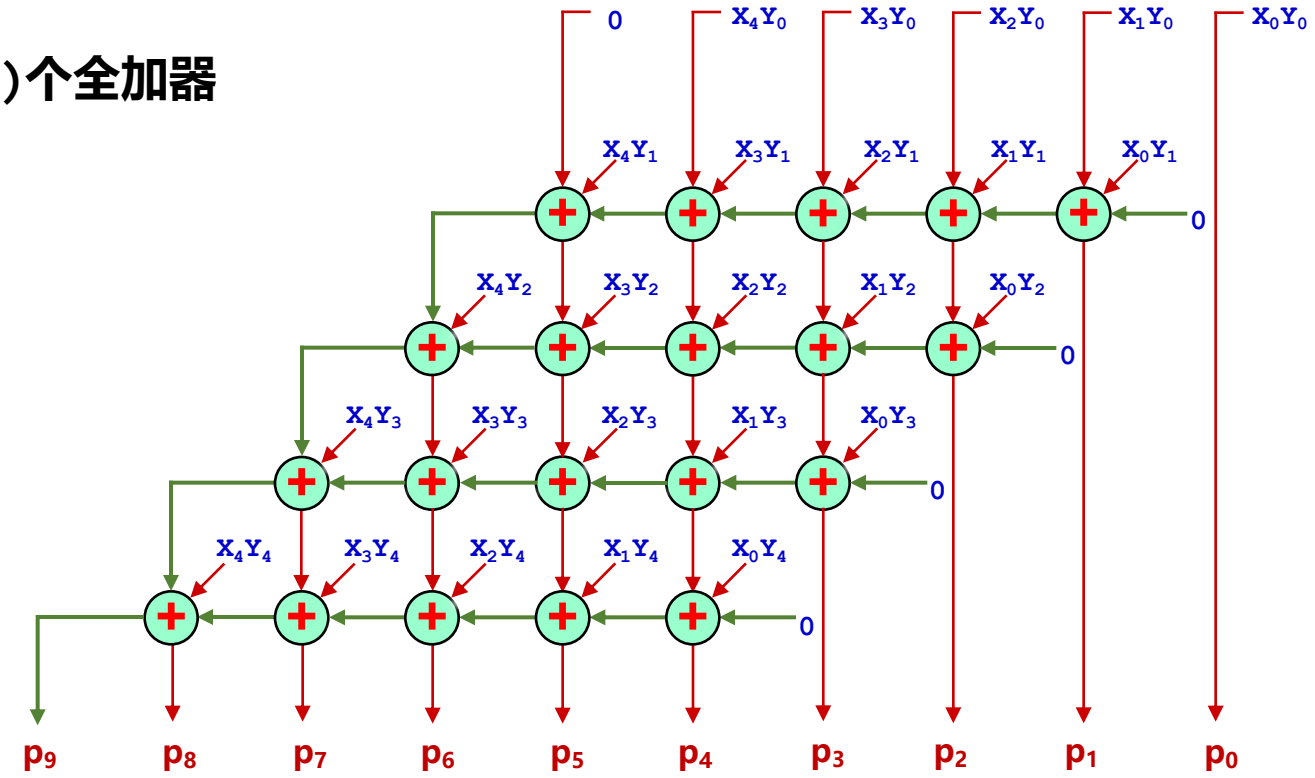
一位乘法逻辑实现

- R=X*Y
  - 1 × 1 = 1
  - 1 × 0 = 0
  - 0 × 1 = 0
  - 0 × 0 = 0
- 与门实现一位乘法
- 25个与门并发
- 一级门延迟，生成所有相加数



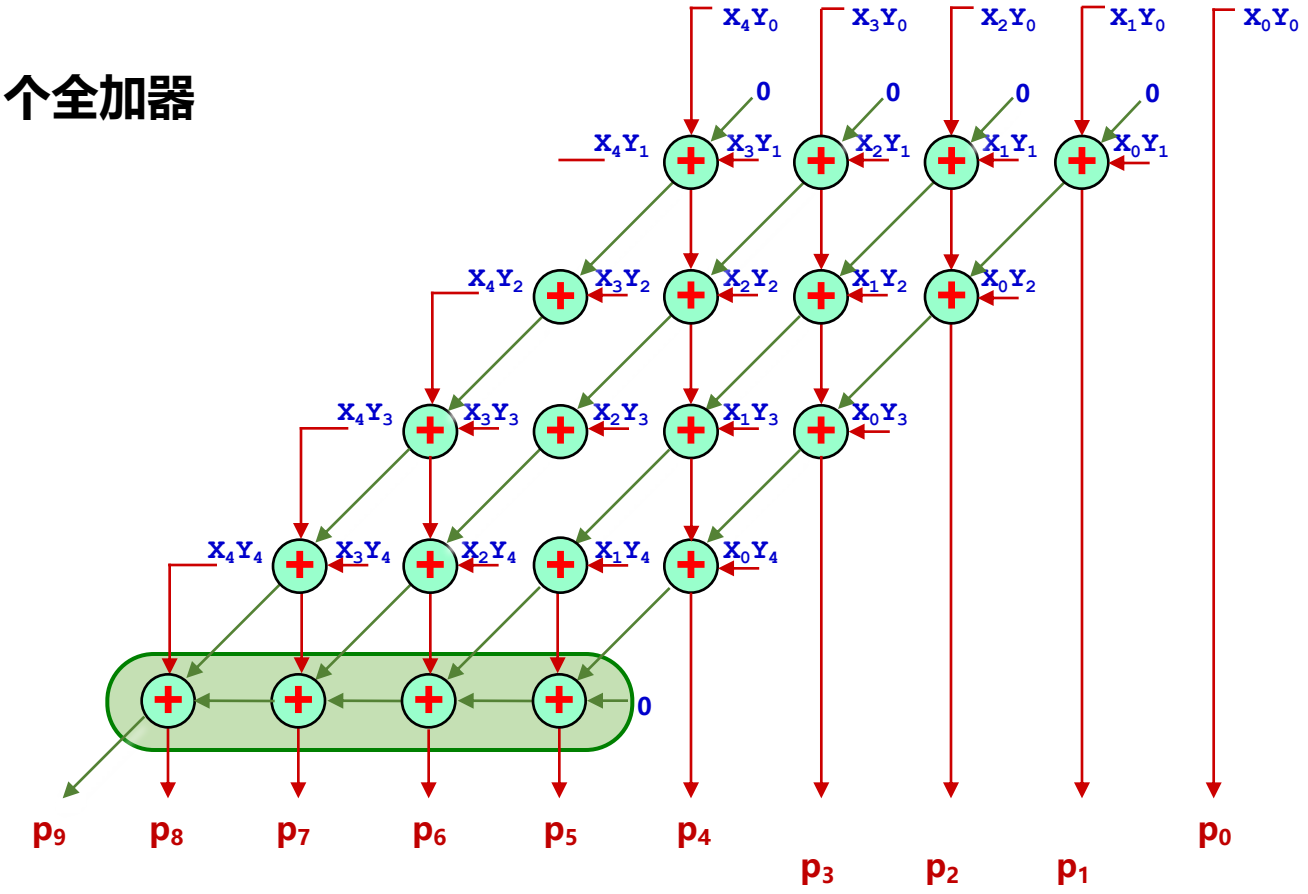
横向进位无符号阵列乘法器

- n*(n-1)个全加器



斜向进位无符号阵列乘法器

- n*(n-1)个全加器



乘法器性能提升

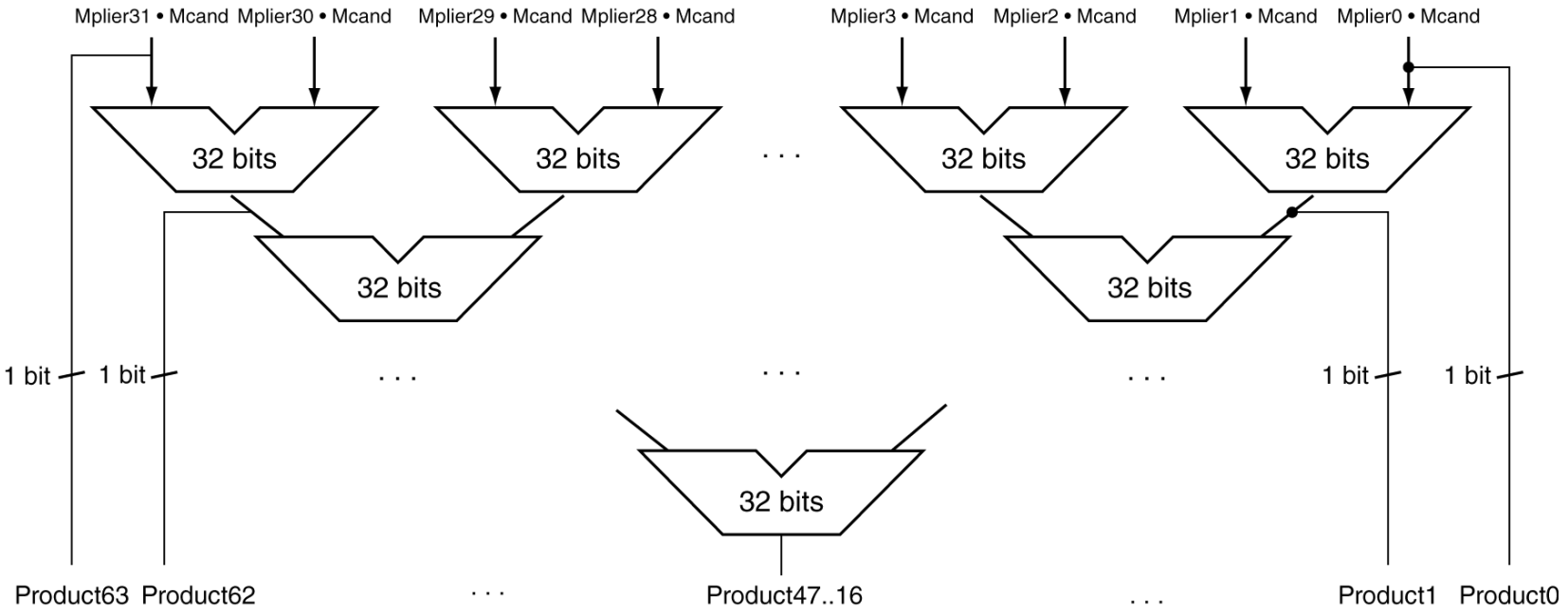
2.3

- 核心算法：n个部分积累加
- Booth一位乘 → Booth两位乘
  - 一位乘法：n个全加器，n^2个全加器时延，面积小 (Intel 8086)
  - 两位乘法： 减少相加数，速度更快，增加额外电路
- 斜向进位阵列乘法器 → 华莱士树
  - 斜向进位：（n^2-n）个全加器，n级全加器时延，面积大
  - 华莱士树： 更多全加器，$\log_2 n$ 级全加器时延，面积更大
- 主流乘法器
  - 二位booth算法 + 华莱士树 + 流水

快速乘法器（并行树）

2.3

- 快速的乘法运算主要思想：为乘数的每一位提供一个32位的加法器；一个用来输入被乘数和一乘数位相与的结果；一个是上一个加法器的输出。



- 方法：将31个加法器组织成一个并行树
- 优点：易于应用流水线设计执行，可以同步支持多个乘法。

四、除法运算

2.3

1. 分析笔算除法

$x = -0.1011 \quad y = 0.1101 \quad$ 求 $x \div y$

```
                    0 . 1 1 0 1
            ┌──────────────────
0 . 1 1 0 1 ) 0 . 1 0 1 1 0
              0 . 0 1 1 0 1
              ─────────────
              0 . 0 1 0 0 1 0
              0 . 0 0 1 1 0 1
              ─────────────
              0 . 0 0 0 1 0 1 0 0
              0 . 0 0 0 0 1 1 0 1
              ─────────────
              0 . 0 0 0 0 0 1 1 1
```

✓商符单独处理

? 心算上商

? 余数不动低位补“0”
减右移一位的除数

? 上商位置不固定

$x \div y = -0.1101$ 商符心算求得

余数 0.0000111

2. 笔算除法和机器除法的比较

2.3

笔算除法	机器除法
商符单独处理	符号位异或形成
心算上商	$\|x\|-\|y\|>0$ 上商 1 $\|x\|-\|y\|<0$ 上商 0
余数 不动 低位补“0” 减右移一位 的除数	余数 左移一位 低位补“0” 减 除数
2 倍字长加法器	1 倍字长加法器
上商位置 不固定	在寄存器 最末位上商

3. 原码除法

以小数为例

$$[x]_{原} = x_0.x_1x_2 \cdots x_n$$

$$[y]_{原} = y_0.y_1y_2 \cdots y_n$$

$$\left[\frac{x}{y}\right]_{原} = (x_0 \oplus y_0).\frac{x^*}{y^*}$$

式中 $x^* = 0.x_1x_2 \cdots x_n$ 为 x 的绝对值

$y^* = 0.y_1y_2 \cdots y_n$ 为 y 的绝对值

商的符号位单独处理 $x_0 \oplus y_0$

数值部分为绝对值相除 $\frac{x^*}{y^*}$

约定

小数定点除法 $x^* < y^*$　整数定点除法 $x^* > y^*$

被除数不等于 0

除数不能为 0

(1) 恢复余数法

2.3

例6.24 $x = -0.1011$ $y = -0.1101$ 求 $[\frac{x}{y}]_{原}$

解：$[x]_{原} = 1.1011$ $[y]_{原} = 1.1101$ $[y^*]_{补} = 0.1101$ $[-y^*]_{补} = 1.0011$

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

②

被除数（余数）	商	说　明
0.1011	0.0000	
+ 1.0011		$+[-y^*]_{补}$
1.1110	0	余数为负，上商 0
+ 0.1101		恢复余数　$+[y^*]_{补}$
0.1011	0	恢复后的余数
1.0110	0	← 1
+ 1.0011		$+[-y^*]_{补}$
0.1001	01	余数为正，上商 1
1.0010	01	← 1
+ 1.0011		$+[-y^*]_{补}$

逻辑左移

逻辑左移

2.3

被除数（余数）	商	说　明
0.0101	011	余数为正，上商 1
0.1010	011	← 1
+ 1.0011		$+[-y^*]_{补}$
1.1101	0110	余数为负，上商 0
+ 0.1101		恢复余数 $+[y^*]_{补}$
0.1010	0110	恢复后的余数
1.0100	0110	← 1
+ 1.0011		$+[-y^*]_{补}$
0.0111	01101	余数为正，上商 1

逻辑左移

逻辑左移

$$\frac{x^*}{y^*} = 0.1101$$

$$\therefore \quad \left[\frac{x}{y}\right]_{原} = 0.1101$$

上商 5 次

第一次上商判溢出

移 4 次

余数为正　上商 1

余数为负　上商 0，恢复余数

(2) 不恢复余数法（加减交替法）

- 恢复余数法运算规则

余数 $R_i>0$ 上商“1”，$2R_i-y^*$

余数 $R_i<0$ 上商“0”，R_i+y^* 恢复余数

$$2(R_i+y^*)-y^*=2R_i+y^*$$

- 不恢复余数法运算规则

上商“1” $2R_i-y^*$

上商“0” $2R_i+y^*$

加减交替

例6.25 $x=-0.1011$ $y=-0.1101$ 求 $[\frac{x}{y}]_{原}$ 2.3

解：

0.1011	0.0000	
+1.0011		$+[-y^*]_{补}$
1.1110	0	余数为负，上商 0
1.1100	0	← 1
+0.1101		$+[y^*]_{补}$
0.1001	01	余数为正，上商 1
1.0010	01	← 1
+1.0011		$+[-y^*]_{补}$
0.0101	011	余数为正，上商 1
0.1010	011	← 1
+1.0011		$+[-y^*]_{补}$
1.1101	0110	余数为负，上商 0
1.1010	0110	← 1
+0.1101		$+[y^*]_{补}$
0.0111	01101	余数为正，上商 1

逻辑左移

逻辑左移

逻辑左移

逻辑左移

$[x]_{原}=1.1011$

$[y]_{原}=1.1101$

$[x^*]_{补}=0.1011$

$[y^*]_{补}=0.1101$

$[-y^*]_{补}=1.0011$

例6.25 结果

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

② $\frac{x^*}{y^*} = 0.1101$

$\therefore \quad [\frac{x}{y}]_{原} = 0.1101$

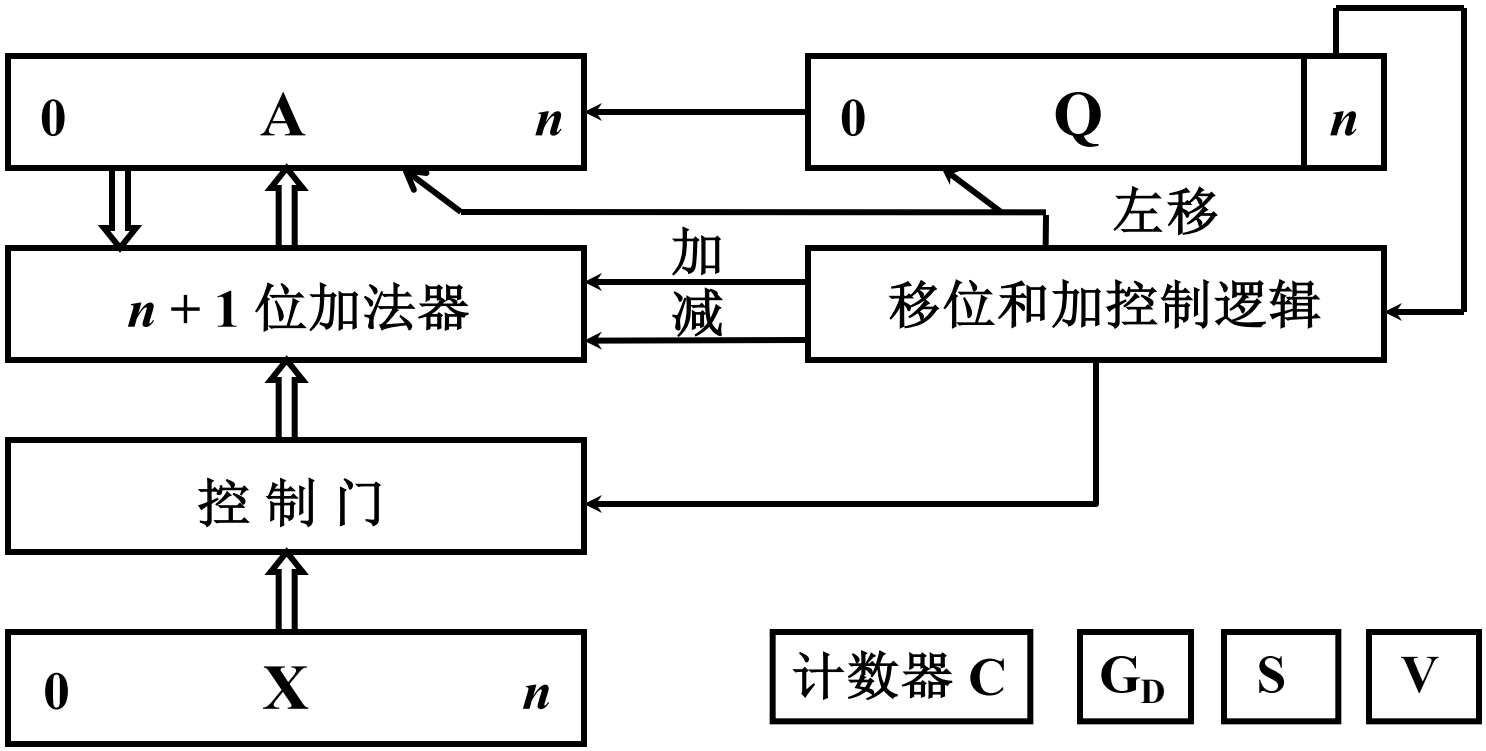
特点 上商 $n+1$ 次

第一次上商判溢出

移 n 次，加 $n+1$ 次

用移位的次数判断除法是否结束

(3) 原码加减交替除法硬件配置



A、X、Q 均 $n+1$ 位

用 Q_n 控制加减交替

6.4 浮点四则运算

一、浮点加减运算

$$x = S_x \cdot 2^{j_x} \qquad y = S_y \cdot 2^{j_y}$$

1. 对阶

(1) 求阶差

$$\Delta j = j_x - j_y = \begin{cases} = 0 & j_x = j_y \quad \text{已对齐} \\ > 0 & j_x > j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \leftarrow 1,\ j_x - 1 \\ y \text{ 向 } x \text{ 看齐} & \checkmark\ S_y \rightarrow 1,\ j_y + 1 \end{cases} \\ < 0 & j_x < j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & \checkmark\ S_x \rightarrow 1,\ j_x + 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \leftarrow 1,\ j_y - 1 \end{cases} \end{cases}$$

(2) 对阶原则

小阶向大阶看齐

例如 $x = 0.1101 \times 2^{01}$　　$y = (-0.1010) \times 2^{11}$

求 $x+y$

解：$[x]_{补} = 00, 01; 00.1101$　　$[y]_{补} = 00, 11; 11.0110$

1. 对阶

① 求阶差　$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$　　$= 00, 01$

$+ \quad 11, 01$

$11, 10$

阶差为负（-2）　　$\therefore S_x \longrightarrow 2 \quad j_x + 2$

② 对阶　　$[x]_{补'} = 00, 11; 00.0011$

2. 尾数求和

$[S_x]_{补'} = 00.0011$　　对阶后的$[S_x]_{补'}$

$+ \quad [S_y]_{补} = 11.0110$

11.1001

$\therefore [x+y]_{补} = 00, 11; 11.1001$

3. 规格化

2.4

(1) 规格化数的定义

$$r = 2 \qquad \frac{1}{2} \le |S| < 1$$

(2) 规格化数的判断

$S>0$	规格化形式	$S<0$	规格化形式
真值	0.1××…×	真值	–0.1××…×
原码	0.1××…×	原码	1.1××…×
补码	0.1××…×	补码	1.0××…×
反码	0.1××…×	反码	1.0××…×

原码　不论正数、负数，第一数位为1

补码　符号位和第 一数位不同

特例

$$S = -\frac{1}{2} = -0.100\cdots0$$

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = 1.100\cdots0$$

∴ $[-\frac{1}{2}]_{补}$ 不是规格化的数

$$S = -1$$

$$[S]_{补} = 1.000\cdots0$$

∴ $[-1]_{补}$ 是规格化的数

(3) 左规

尾数左移一位，阶码减 1，直到数符和第一数位不同为止

上例　$[x+y]_{补} = 00,\ 11;\ 11.\ 1001$

左规后　$[x+y]_{补} = 00,\ 10;\ 11.\ 0010$

$$\therefore x + y = (-0.1110) \times 2^{10}$$

(4) 右规

当 尾数溢出（ >1）时， 需 右规

即尾数出现 01. ×× …×或 10. ×× … ×时

尾数右移一位，阶码加 1

例6.27 $x = 0.1101 \times 2^{10}$ $y = 0.1011 \times 2^{01}$ 2.4

求 $x+y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：$[x]_{补} = 00, 010; 00.\ 110100$

$[y]_{补} = 00, 001; 00.\ 101100$

① 对阶

$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$ = 00, 010

+ 11, 111

100, 001

阶差为 +1 $\therefore S_y$ 1, j_y+1

$\therefore [y]_{补'} = 00, 010; 00.\ 010110$

② 尾数求和

$[S_x]_{补}$ = 00. 110100

+ $[S_y]_{补'}$ = 00. 010110 对阶后的$[S_y]_{补'}$

01. 001010 尾数溢出需右规

③ 右规

$[x+y]_{补} = 00,\ 010;\ 01.\ 001010$

右规后

$[x+y]_{补} = 00,\ 011;\ 00.\ 100101$

$\therefore x+y = 0.\ 100101 \times 2^{11}$

4. 舍入

在 对阶 和 右规 过程中，可能出现 尾数末位丢失 引起误差，需考虑舍入

(1) 0 舍 1 入法

(2) 恒置 “1” 法

例 6.28　$x=(-\frac{5}{8})\times 2^{-5}$　$y=(\frac{7}{8})\times 2^{-4}$

求 $x-y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：　$x=(-0.101000)\times 2^{-101}$　　$y=(0.111000)\times 2^{-100}$

$[x]_{补}=11,011;\ 11.011000$　　$[y]_{补}=11,100;\ 00.111000$

① 对阶

$$[\Delta j]_{补}=[j_x]_{补}-[j_y]_{补}=11,011+00,100=11,111$$

阶差为 –1　　$\therefore S_x \longrightarrow 1,\ j_x+1$

$\therefore [x]_{补'}=11,100;\ 11.101100$

② 尾数求和

$$[S_x]_{补'} = 11.101100$$
$$+\ [-S_y]_{补} = 11.001000$$
$$\overline{\qquad\qquad\qquad}$$
$$110.110100$$

③ 右规

$$[x-y]_{补} = 11,100;\ 10.110100$$

右规后

$$[x-y]_{补} = 11,101;\ 11.011010$$

$$\therefore x-y = (-0.100110)\times 2^{-11}$$

$$= (-\frac{19}{32})\times 2^{-3}$$

5. 溢出判断

设机器数为补码，尾数为 规格化形式，并假设阶符取 2 位，阶码的数值部分取 7 位，数符取 2 位，尾数取 n 位，则该 补码 在数轴上的表示为