计算机组成原理

Principle of Computer Organization

✓第二章 运算方法与运算器

第二部分

北京邮电大学

计算机学院

戴志涛

定点乘除法运算

乘除法运算的机器实现方法

1.完全软件实现

移位加算法

不设乘除法运算的硬件电路，而是由软件利用运算器中的加法和移位操作编程进行乘除法运算

2.加法器增加硬件辅助电路实现

移位加算法

利用运算器中的加法器硬件电路和移位电路，再设计必要的扩展电路，用硬件通过加法和移位操作实现乘除法运算

3.专用乘除法器实现

在运算器中除了设置加法器之外，再增加硬件电路设置高速乘除法部件，直接完成乘除法运算

定点乘法运算算法

- 原码一位乘法运算
- 补码一位乘法运算
- 原码两位乘法运算
- 原码并行乘法运算
- 补码并行乘法运算

原码一位乘法运算

令$[X]_{原} = X_f.X_1\ X_2\ X_3\ldots\ldots X_n$

$[Y]_{原} = Y_f.Y_1\ Y_2\ Y_3\ldots\ldots Y_n$

则

$$[X\times Y]_{原} = (X_f \oplus Y_f) + (|X|\times|Y|)$$

$$= (X_f \oplus Y_f) + (0.X_1X_2\cdots X_n \times 0.Y_1Y_2\cdots Y_n)$$

积的符号：被乘数与乘数二符号的异或值

积的数值：被乘数与乘数二数的绝对值之积

手工乘法运算

例. 0.1101×（-0.1011）

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

上符号： 1.10001111

部分积

令$y=0.y_1y_2\ldots\ldots y_n$

则$xy=x\times(y_1\times2^{-1}+y_2\times2^{-2}+\ldots\ldots+y_n\times2^{-n})$

$=2^{-1}\times(xy_1+xy_2\times2^{-1}+\ldots\ldots+xy_n\times2^{-n+1})$

$=2^{-1}\times(xy_1+2^{-1}(xy_2+2^{-1}(xy_3+\ldots\ldots$

$+2^{-1}(xy_n+0)\ldots\ldots))=\ldots\ldots$

令$z_0=0$

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

$z_2=2^{-1}(xy_{n-1}+z_1)$

$\ldots\ldots$ $z_i=2^{-1}(xy_{n-i+1}+z_{i-1})$ $\ldots\ldots$ **部分积**

$z_n=2^{-1}(xy_1+z_{n-1})=xy$

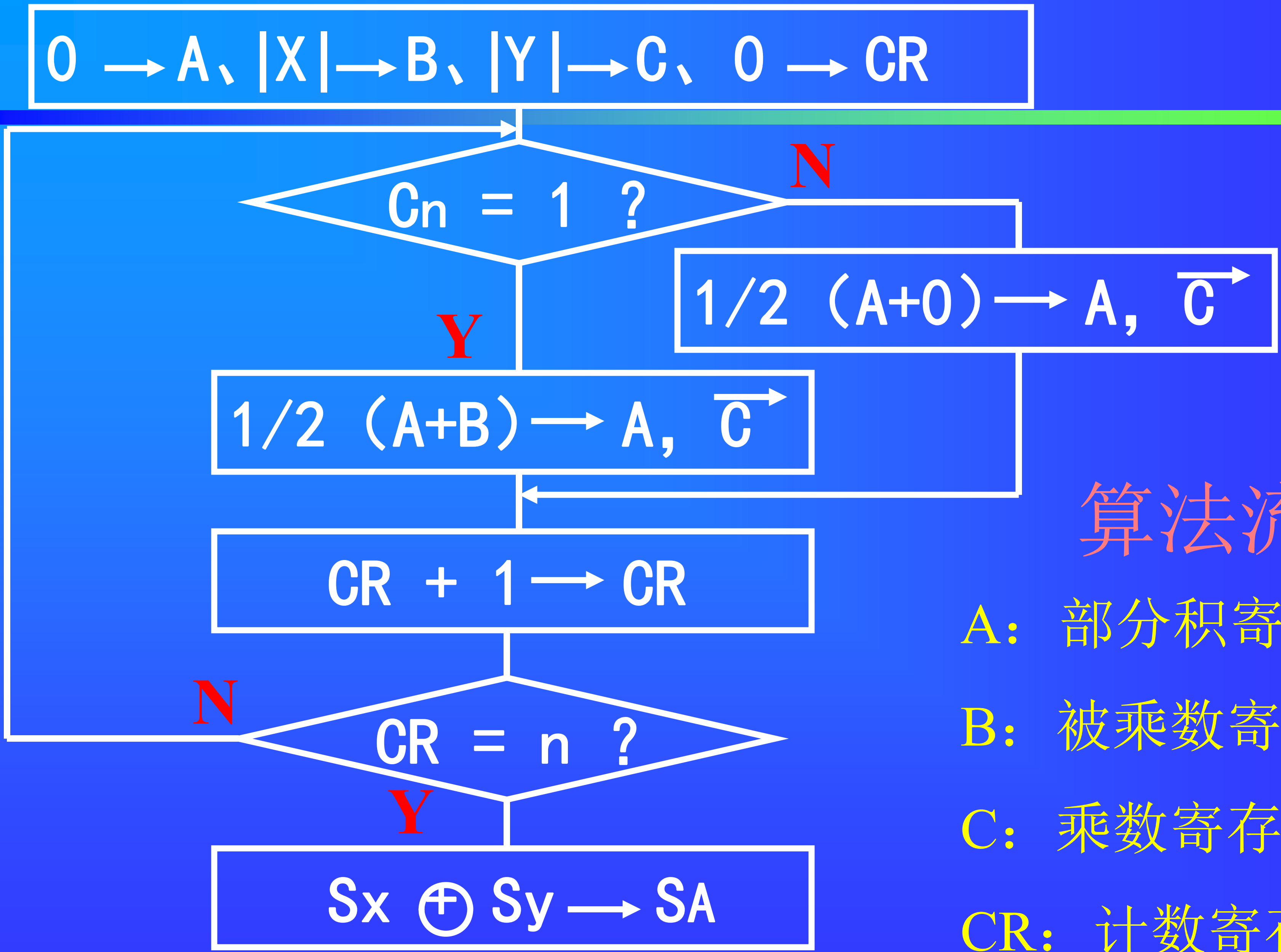
A 部分积 Z 乘数 Y C B 00.1101 被乘数

	部分积 Z	乘数 Y	C	
	00 0000	1 0 1 **1**		$z_0=0, y_4=1$
+x	00 1101			xy_4+z_0
	00 1101			
右移一位→	00 0110	1 1 0 **1**	1(丢弃)	$z_1=2^{-1}(xy_4+z_0)$
+x	00 1101			
	01 0011			
右移一位→	00 1001	1 1 1 **0**	1(丢弃)	
+0	00 0000			
	00 1001			
右移一位→	00 0100	1 1 1 **1**	0(丢弃)	
+x	00 1101			
	01 0001			
右移一位→	00 1000	1 1 1 1	1(丢弃)	
	乘积高位	乘积低位		

例：设 X=0.1101, Y=0.1011, 求X•Y

X•Y=0.10001111

移位4次，运算完成



算法流程

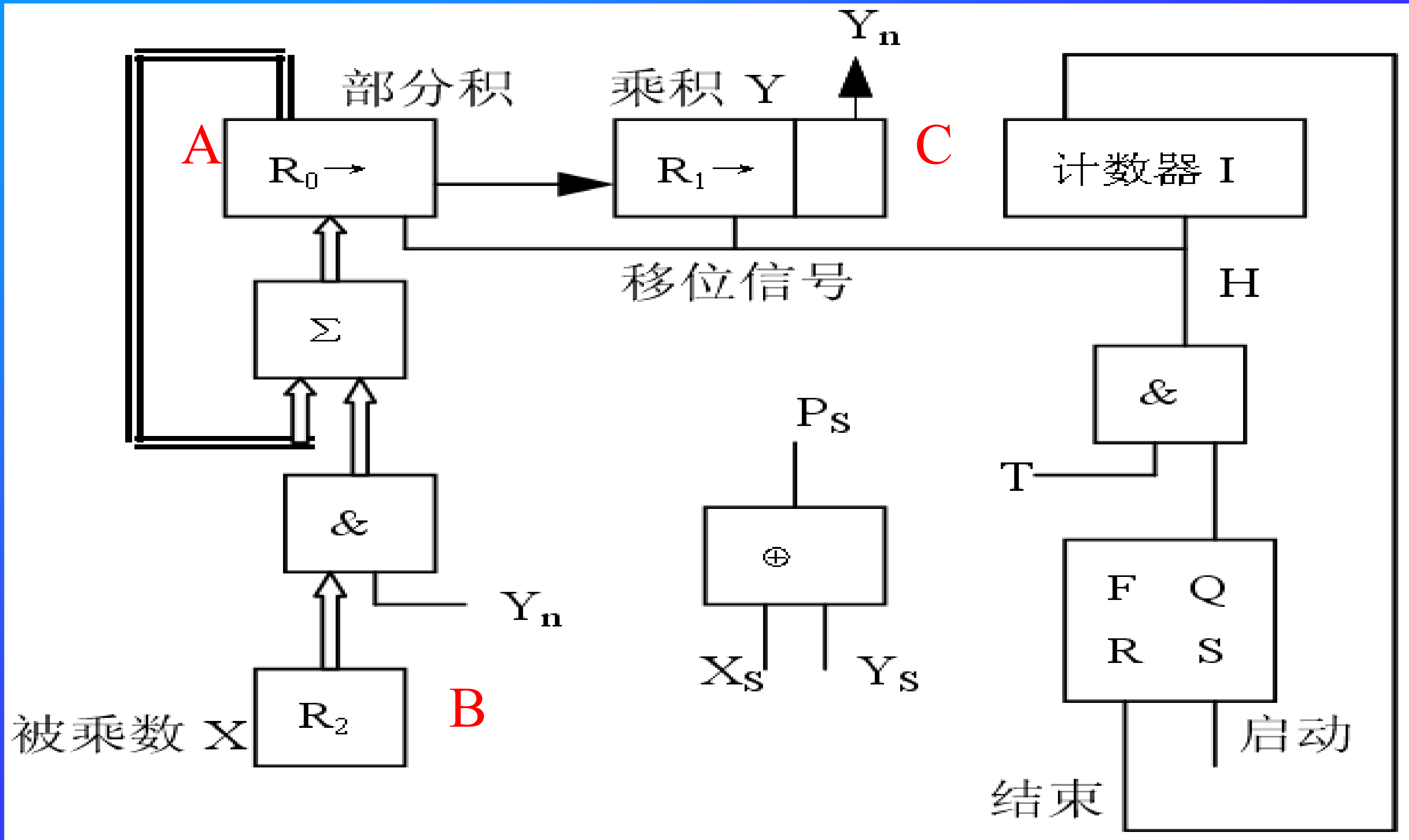
A： 部分积寄存器

B： 被乘数寄存器

C： 乘数寄存器

CR： 计数寄存器

原码一位乘法的硬件电路



无符号数阵列乘法

$A = a_{m-1} \ldots a_1 a_0$
$B = b_{n-1} \ldots b_1 b_0$

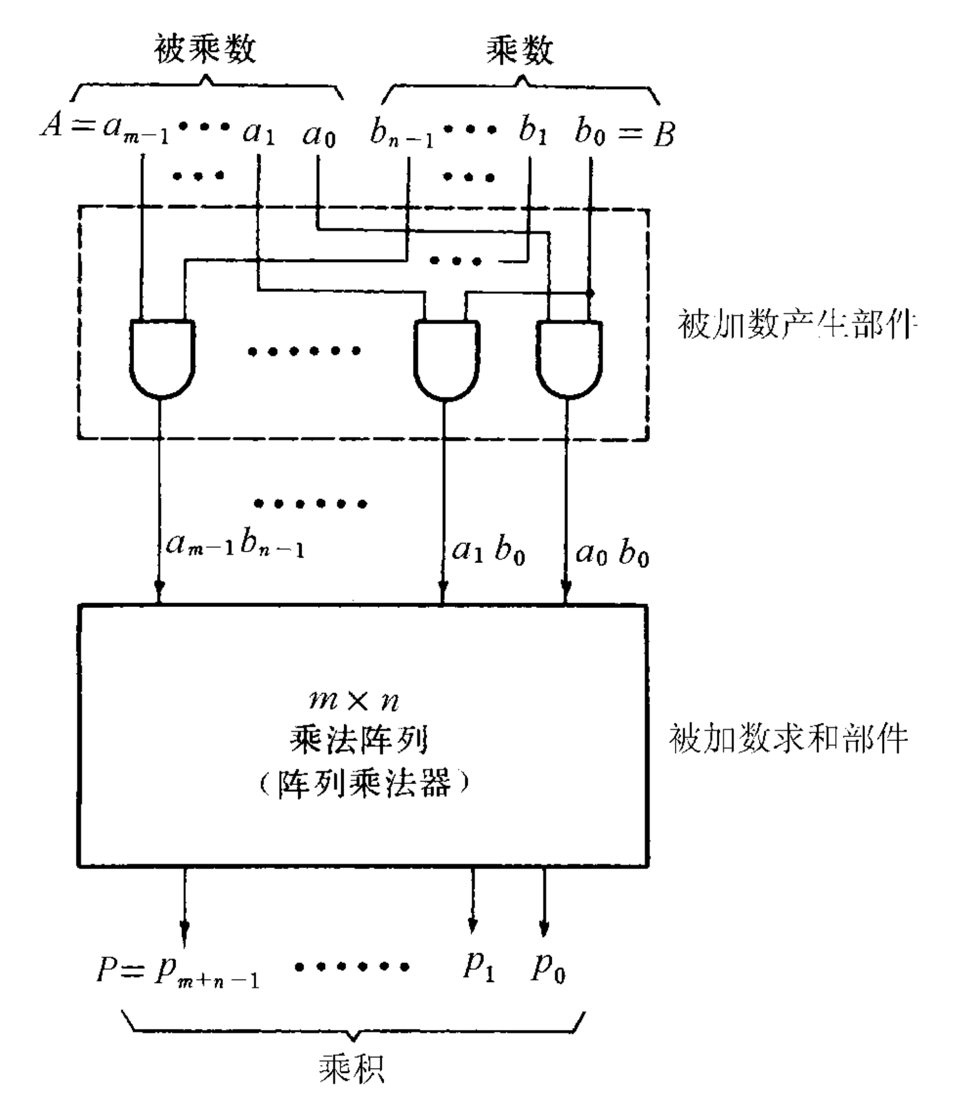
$$a = \sum_{i=0}^{m-1} a_i 2^i \qquad b = \sum_{j=0}^{n-1} b_j 2^j$$

$P = A*B = p_{m+n-1} \ldots p_1 p_0$

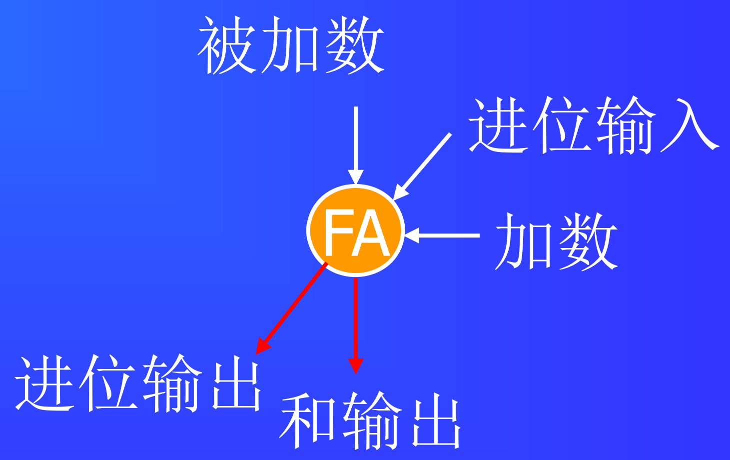
$$p = ab = \sum_{i=0}^{m-1} a_i 2^i \bullet \sum_{j=0}^{n-1} b_j 2^j = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} (a_i b_j) 2^{i+j} = \sum_{k=0}^{m+n-1} p_k 2^k$$

$$\begin{array}{rcccccccc}
 & & & & a_{m-1} & a_{m-2} & \cdots & a_1 & a_0 = A \\
\times) & & & & & b_{n-1} & \cdots & b_1 & b_0 = B \\
\hline
 & & & & a_{m-1}b_0 & a_{m-2}b_0 & \cdots & a_1b_0 & a_0b_0 \\
 & & & a_{m-1}b_1 & a_{m-2}b_1 & \cdots & a_1b_1 & a_0b_1 & \\
 & & \cdot & & & & \cdot & & \\
 & \cdot & & & & \cdot & & & \\
+) & a_{m-1}b_{n-1} & a_{m-2}b_{n-1} & \cdots & a_1b_{n-1} & a_0b_{n-1} & & & \\
\hline
p_{m+n-1} & p_{m+n-2} & p_{m+n-3} & \cdots & p_{n-1} & \cdots & & p_1 & p_0 = P
\end{array}$$

无符号数阵列乘法

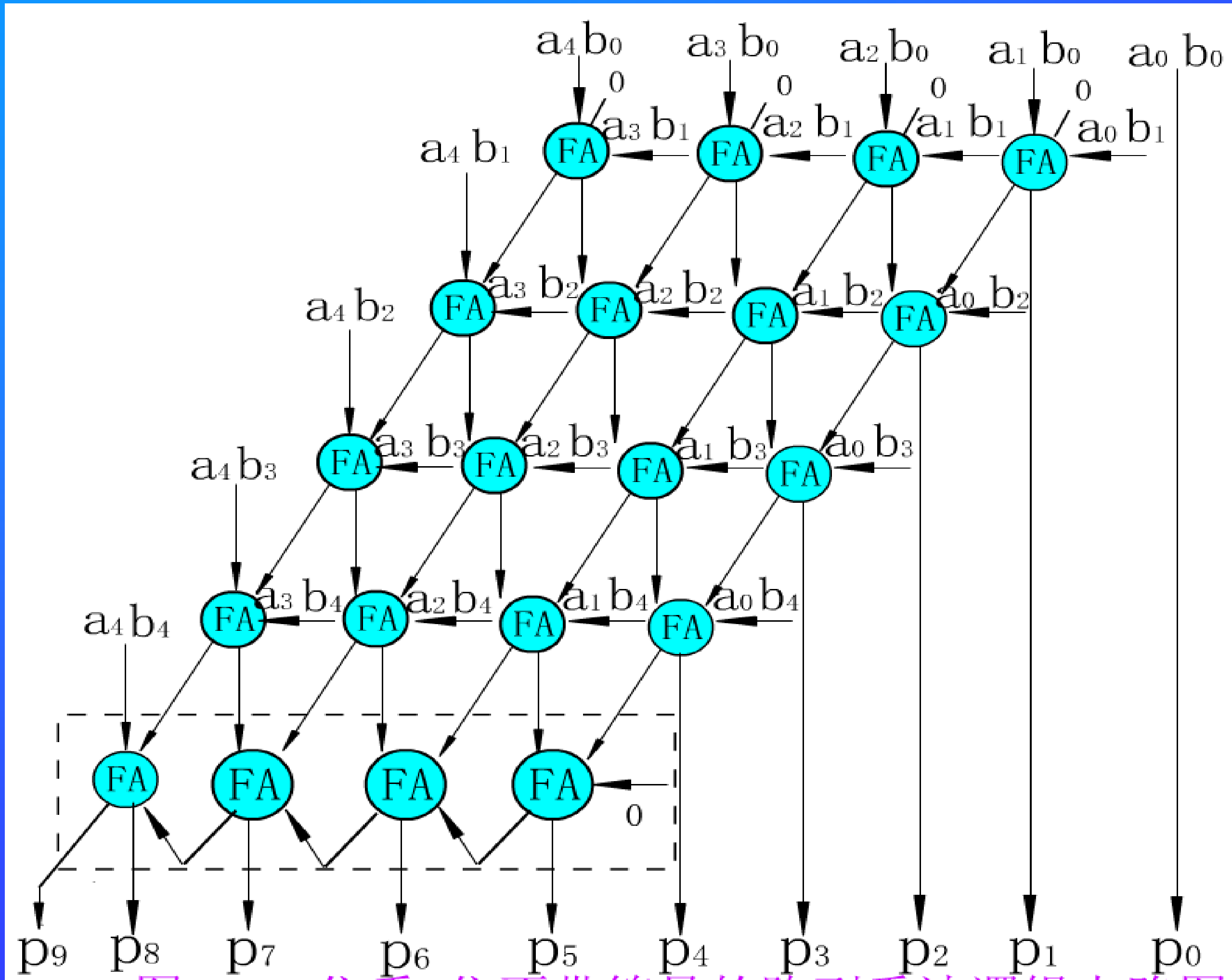


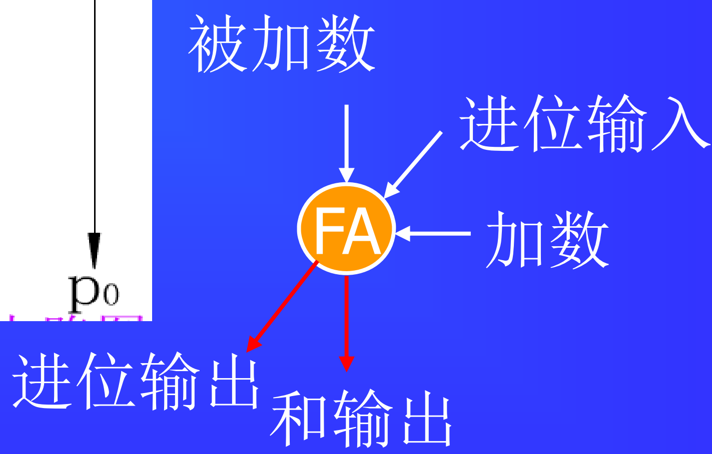
被乘数
乘数
$A=a_{m-1}\cdots a_1\ a_0$
$b_{n-1}\cdots b_1\ b_0=B$
被加数产生部件
$a_{m-1}b_{n-1}$
$a_1 b_0$
$a_0 b_0$
$m\times n$
乘法阵列
（阵列乘法器）
被加数求和部件
$P=p_{m+n-1}\cdots p_1\ p_0$
乘积



被加数
进位输入
FA
加数
进位输出
和输出

5位 ×5位阵列乘法器逻辑图



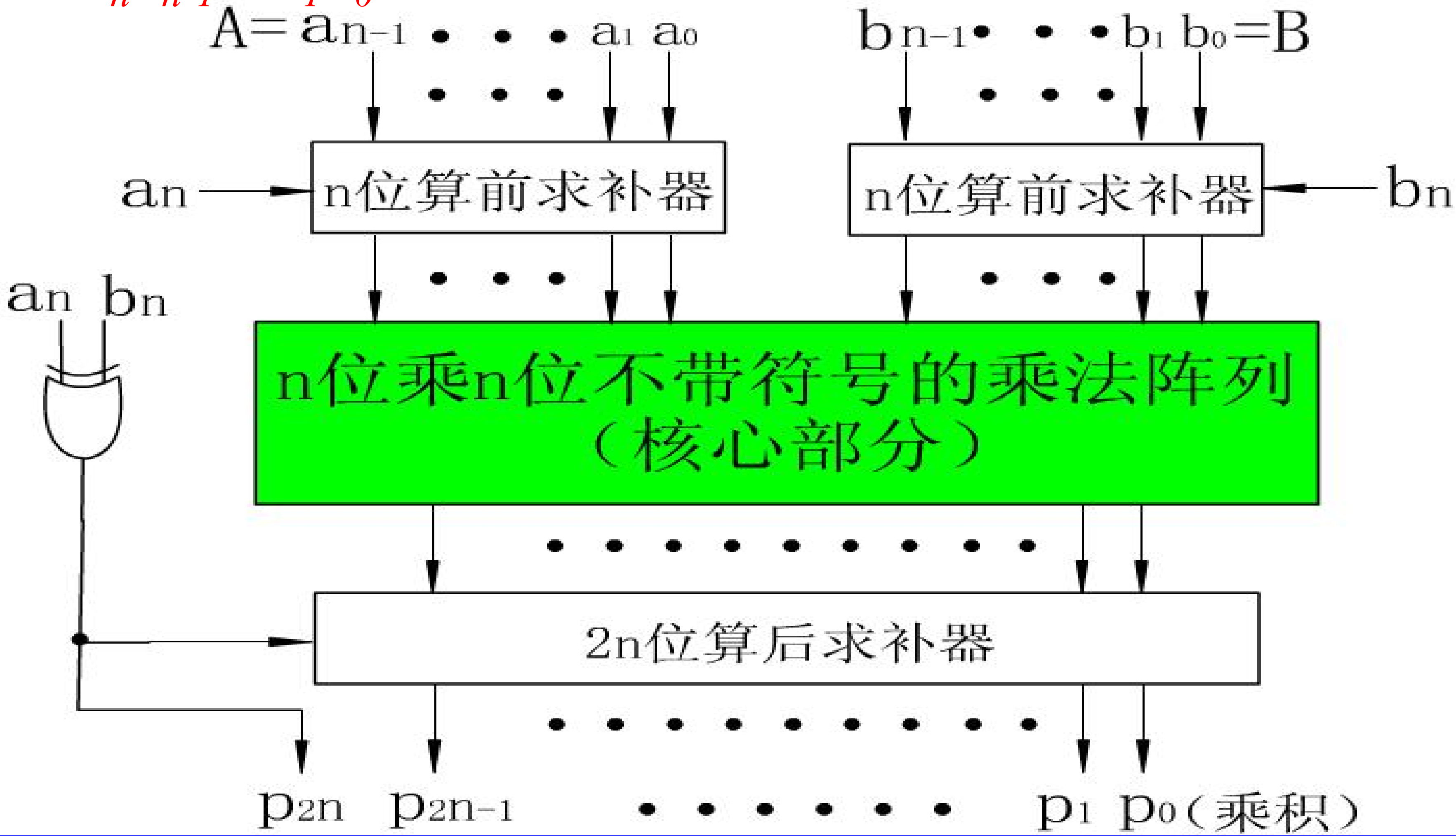


带符号数的并行乘法运算

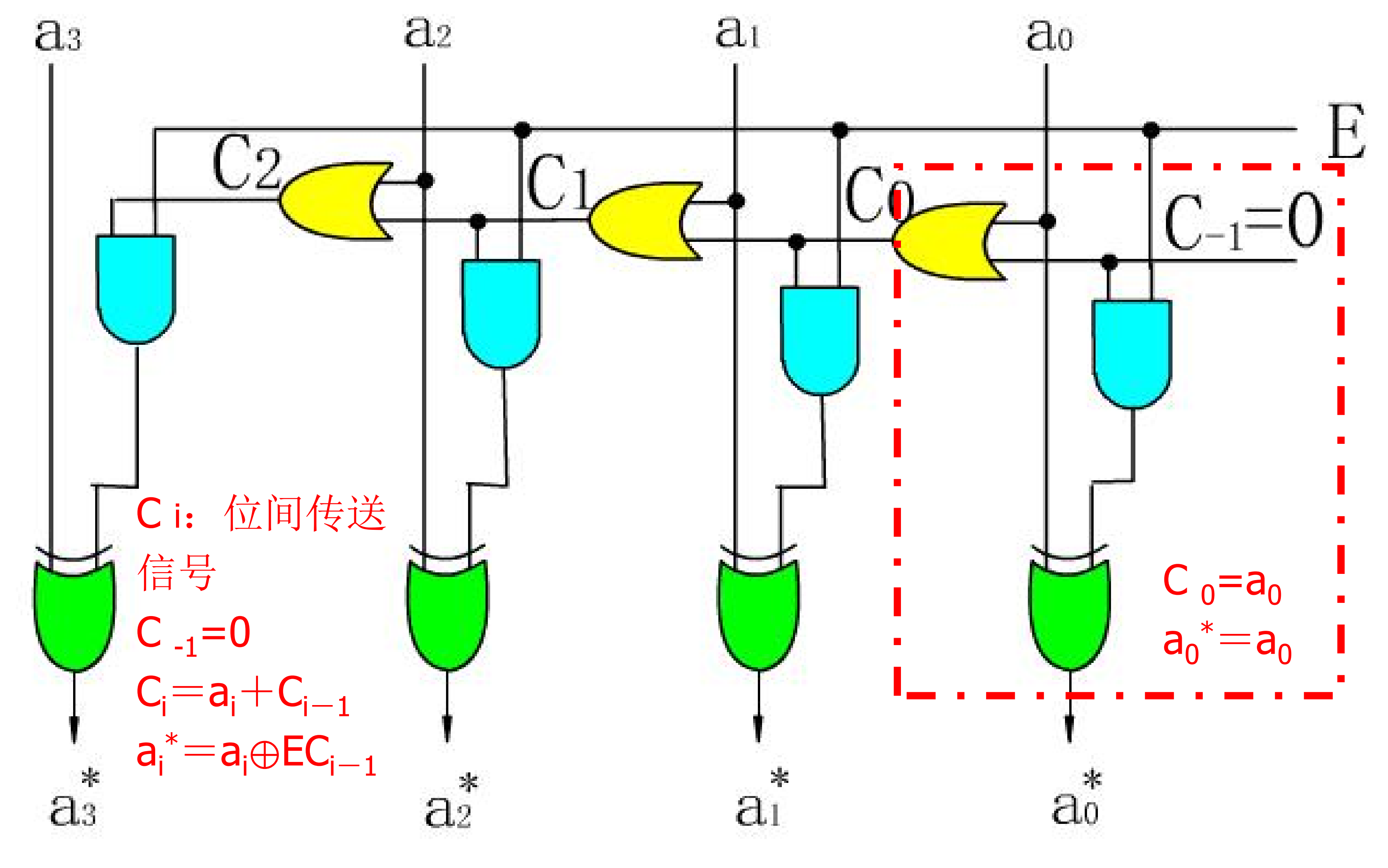
- 原码：
  - 绝对值参加无符号数乘法器运算
  - 符号位通过异或门得到乘积的符号
- 补码：
  - 直接补码乘法
  - 间接补码乘法
    - 乘数为正数：绝对值参加无符号数乘法器运算
    - 乘数为负数：由补码求得其绝对值后参加无符号数乘法器运算
    - 乘数的符号位异或得到乘积的符号
    - 若乘积为负数，需将乘积的绝对值经过求补电路得到积的补码

间接补码阵列乘法器

$A=a_n a_{n-1}\ldots a_1 a_0$（被乘数） $B=b_n b_{n-1}\ldots b_1 b_0$（乘数）



对2求补电路



C i：位间传送信号

$C_{-1}=0$

$C_i = a_i + C_{i-1}$

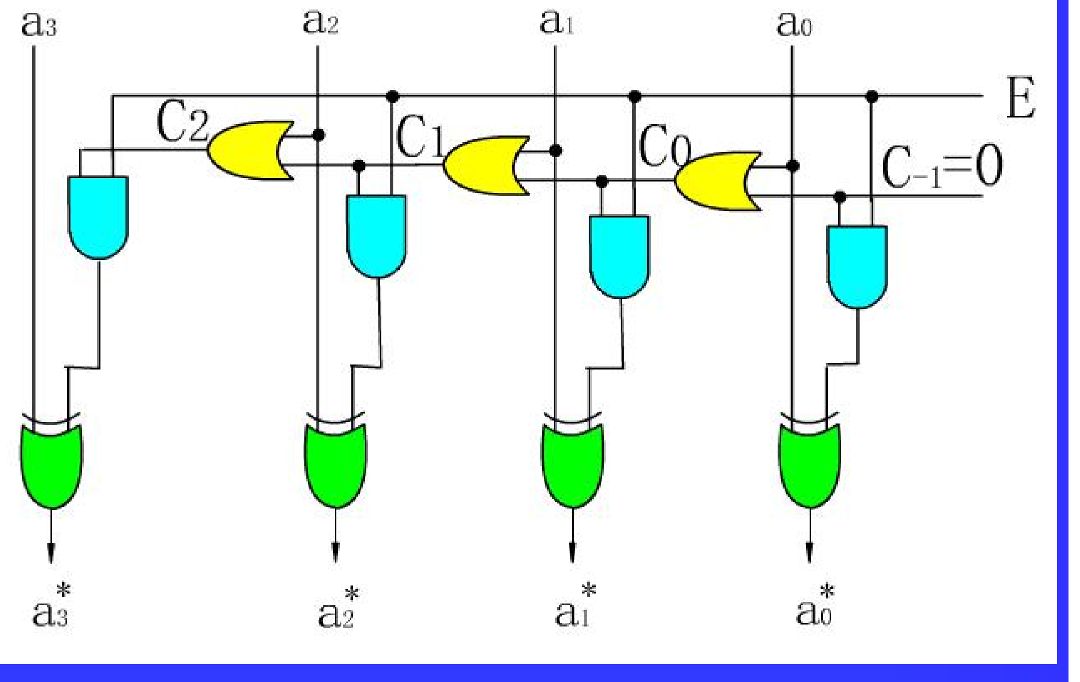
$a_i^* = a_i \oplus EC_{i-1}$

$C_0 = a_0$

$a_0^* = a_0$

对2求补器的时间延迟

- 对2求补器转换 n+1位绝对值所需的总时间延迟：
  - $t_{TC}=n\cdot 2T+5T=(2n+5)T$
  - 或门：2T
  - 与门：2T
  - 异或门：3T



原码除法运算原理

- 原码除法：
  - 商的符号：两数的符号异或
  - 商的绝对值：两数的绝对值相除
- 设有n位定点小数
  - 被除数x，$[x]_{原}=x_f.x_{n-1}\ldots x_1x_0$
  - 除数y，$[y]_{原}=y_f.y_{n-1}\ldots y_1y_0$
  - 则商q＝x/y，

    $[q]_{原}=(x_f\oplus y_f)+(0.x_{n-1}\ldots x_1x_0/0.y_{n-1}\ldots y_1y_0)$
- 除法：（2n位被除数）/（n位除数）＝（n位商）
  - 纯整数：被除数的高n位<除数，否则商至少有n+1位
  - 纯小数：被除数<除数，否则商溢出

手工除法运算

- 设被除数 $x=0.1001$，除数 $y=0.1011$
- 求 $x \div y$

0.1 1 0 1	商q	
0.1 0 1 1) 0.1 0 0 1 0	$x(r_0)$	被除数小于除数，商0
−0.0 1 0 1 1	$2^{-1}y$	除数右移1位，减除数，商1
0.0 0 1 1 1 0	r_1	得余数r1
−0.0 0 1 0 1 1	$2^{-2}y$	除数右移1位，减除数，商1
0.0 0 0 0 1 1 0	r_2	得余数r2
−0.0 0 0 1 0 1 1	$2^{-3}y$	除数右移1位，不减除数，商0
0.0 0 0 0 1 1 0 0	r_3	得余数r3
−0.0 0 0 0 1 0 1 1	$2^{-4}y$	除数右移1位，减除数，商1
−0.0 0 0 0 0 0 0 1	r_4	得余数r4

商q＝0.1101，余数＝0.00000001

原码除法运算原理

- 判断|x|-|y|是否够减
  - 令r=|x|-|y|
  - 若r>=0则够减
  - 若r<0则不够减
- 原码减法运算：
  - 减|y|用+[-|y|]$_{补}$实现

求两个正数原码的差

已知$[x]_{原}$和$[y]_{原}$，且$x>=0$，$y>=0$，

求$[x]_{原}-[y]_{原}$：

因$x>=0$，$y>=0$，故

$$[x]_{原}-[y]_{原}=[x]_{补}-[y]_{补}=[x]_{补}+[-y]_{补}$$
$$=[x]_{原}+[-y]_{补}$$

- 由$[y]_{原}=[y]_{补}$求$[-y]_{补}$：
  $[y]_{原}$连同符号位在内，各位取反，末位加1
- 当$[x]_{原}+[-y]_{补}$的求和结果：
  - 最高位无进位时，结果为负
  - 最高位有进位时，结果为正

原码除法运算原理

- 部分余数右边补0：部分余数左移
- 不够减的处理：
  - 恢复余数法
    - 当前的余数加上除数
  - 加减交替法（不恢复余数法）
    - 将恢复余数与下一步部分余数左移减去除数合并为加余数

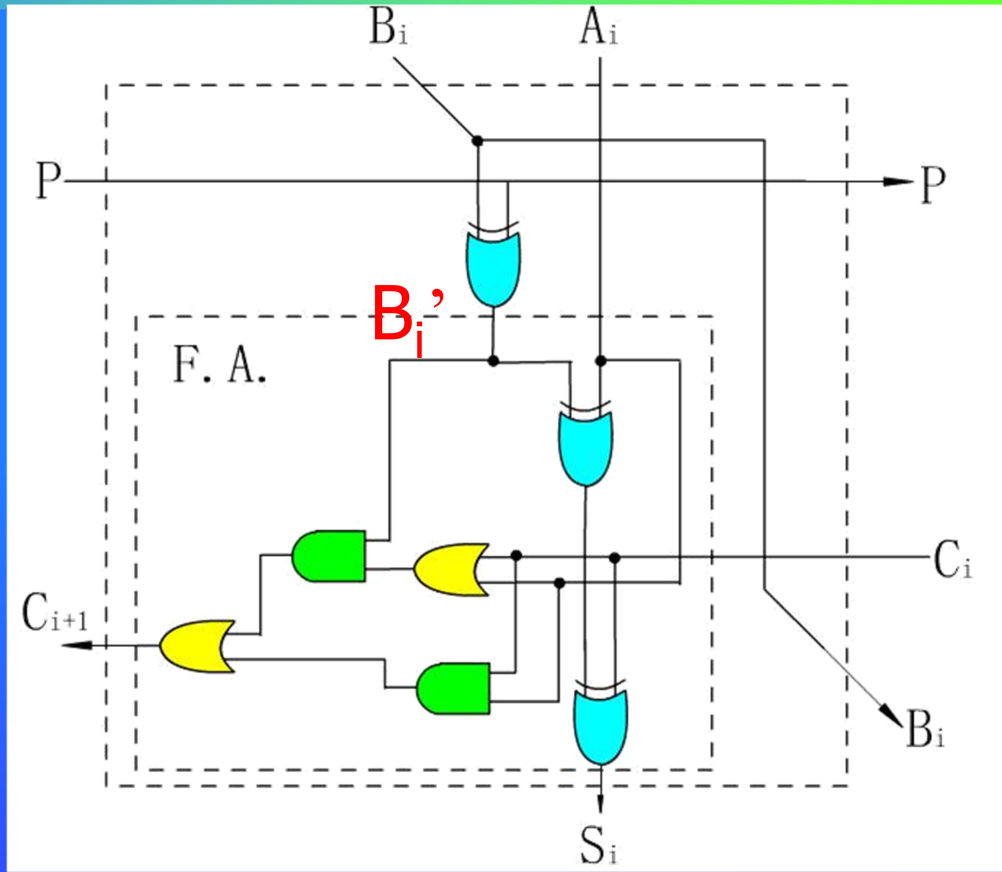
加减交替法

- 判断是否够减：被除数或部分余数左移减去除数得新余数
  $r'_i = 2r_{i-1} - y$
  - $r'_i >= 0$，r'_i左移减除数：
    - $r_i = r'_i$
    - $r'_{i+1} = 2r_i - y = 2r'_i - y$
  - $r'_i < 0$，r'_i左移加除数：
    - $r_i = r'_i + y$
    - $r'_{i+1} = 2r_i - y = 2(r'_i + y) - y = 2r'_i + y$

- 如果最终的余数是负数，需“纠余”得到除法运算的余数：
  $r_n = r'_n + y$

可控加法/减法单元（CAS）

- P=0:加法
- P=1:减法



- $S_i = A_i \oplus (B_i \oplus P) \oplus C_i$
- $C_{i+1} = (A_i + C_i) \cdot (B_i \oplus P) + A_i C_i$

可控加法/减法单元（CAS）

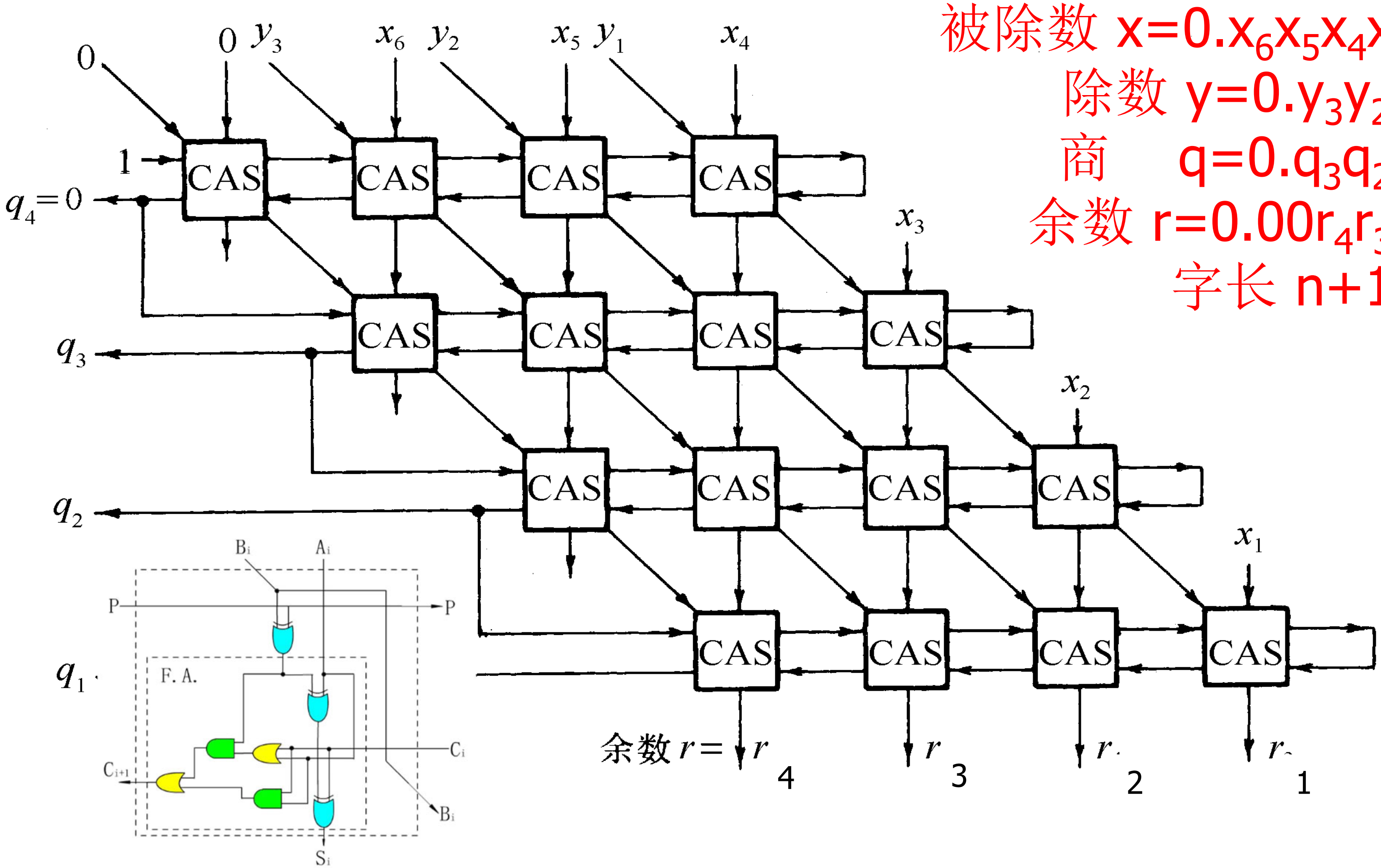
- CAS的输入与输出关系
  - $S_i=A_i\oplus(B_i\oplus P)\oplus C_i$
  - $C_{i+1}=(A_i+C_i)\cdot(B_i\oplus P)+A_iC_i$
- CAS延迟时间：3T

当P=0时	当P=1时
$S_i=A_i\oplus B_i\oplus C_i$	$S_i=A_i\oplus \overline{B}_i\oplus C_i$
$C_{i+1}=A_iB_i+B_iC_i+A_iC_i$	$C_{i+1}=A_i\overline{B}_i+\overline{B}_iC_i+A_iC_i$

$$S_i=A_i\oplus(B_i\oplus P)\oplus C_i= A_iB_iC_i\overline{P}+ A_iB_i\overline{C}_i\overline{P}+ A_i\overline{B}_iC_iP+ A_iB_iC_iP\overline{+}$$
$$A_i\overline{B}_iC_iP+ \overline{A}_i\overline{B}_i\overline{C}_iP+ \overline{A}_iB_i\overline{C}_iP+ \overline{A}_i\overline{B}_iC_i\overline{P}$$

$$C_{i+1}=(A_i+C_i)\cdot(B_i\oplus P)+A_iC_i= A_iB_i\overline{P}+ A_i\overline{B}_iP+ B_iC_i\overline{P}+\overline{B}_iC_iP+A_iC_i$$

加减交替法阵列除法器



被除数 $x=0.x_6x_5x_4x_3x_2x_1$

除数 $y=0.y_3y_2y_1$

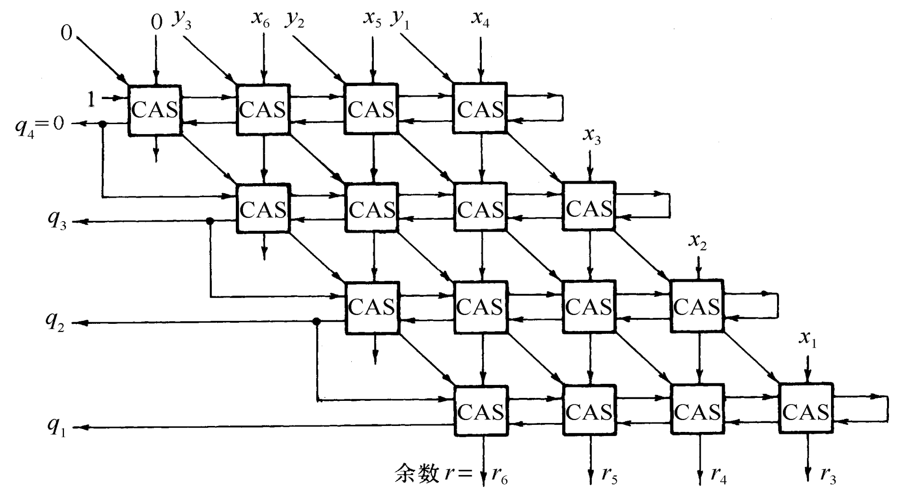
商 $q=0.q_3q_2q_1$

余数 $r=0.00r_4r_3r_2r_1$

字长 $n+1=4$

加减交替法阵列除法器

- 2n位除以n位
  - 硬件复杂度：
    - $(n+1)^2$个CAS单元
  - 除法执行时间：
    - $t_d = 3(n+1)^2T$



(b) 4 位除 4 位阵列除法器

浮点数的机器表示与运算

机器零

- **机器零**：计算机看成零值的浮点数
  - 当浮点数的尾数为0，且其阶码为0时，浮点数被当作0
  - 当浮点数的绝对值过小，以致阶码过小时，不管其尾数为何值，均用0表示该数

浮点数的规格化

- **有效位数**
  - 0.00123——有效位数3位
  - 0.0012300——有效位数5位
- 浮点数的各种表示方法中，精度最高的表示法？
  - 能够表示最多的有效位数
  - 最高有效位就是有效数字

浮点数的规格化

- ➢规格化浮点数（normalized number）
  - ❑ 对浮点数0：阶码（或阶）和尾数均为0
    - ⌧x=0：m=0，e=0
  - ❑ 对非0浮点数：尾数的绝对值大于（或大于等于）1/2
  - ❑ **规格化的浮点数有唯一的形式**
- ➢当用定点小数的原码表示浮点数的尾数时：
  - ❑满足1/2≤|m|<1的x为规格化的浮点数
    - ⌧$[m]_{补}$形如x.1xxxxxxx）
  - ❑反之则为非规格化浮点数

浮点数的规格化

- 当用定点小数的补码表示浮点数的尾数时：
  - 正数，满足$1/2 \leqslant m<1$的x为规格化的浮点数
    - $[m]_补$形如0.1xxxxxxx
  - 负数，满足$-1 \leqslant m<-1/2$的x为规格化的浮点数
    - $[m]_补$形如1.0xxxxxxx
- 对于用补码表示尾数的规格化数，必有：
  - 尾数的符号位不等于最高有效位

负数：满足m=-1/2？

$[-1/2]_补 = [-0.100000]_补 = 1.100000$ 非规格化

$[-1]_补 = 2+(-1)$ （定义） $= 1.000000$ 规格化

浮点数的机器表示

➤ IEEE 754标准浮点数的格式　　$N = m \times R^e = m \times 2^e$

32位短浮点数

31	30 … 23	22 … 0
S	E	M

64位长浮点数

63	62 … 52	51 … 0
S	E	M

- 数符S：浮点数的符号位，0表示正数，1表示负数
- 尾码M：定点小数，表示尾数的绝对值，小数点约定在尾数字段的最前面
- 阶E：移码表示

移码表示法便于比较两个指数的大小和对阶

浮点数的机器码与真值的关系

32位
短浮点数

31	30　　　23	22　　　0
S	E	M

- IEEE 754标准**规格化**的32位浮点数
  - E=e+127
  - 尾数：1.M　　　[绝对值0.1xxxx->1.xxx]
  - 真值：$x = (-1)^{S} \times (1.M) \times 2^{E-127}$
- IEEE 754标准**规格化**的64位浮点数的真值

$$x = (-1)^{S} \times (1.M) \times 2^{E-1023}$$

浮点数的机器表示：例2

将十进制数20.59375转换成IEEE754 32位浮点数的二进制格式

解：1）分别将整数和小数部分转换成二进制数

$$(20.59375)_{10}=10100.10011$$

2）移动小数点到第1、2位之间（规格化）

$$10100.10011=1.010010011\times 2^4$$

3）可以得到：

S=0，E=e+127=131，M=010010011

4）32位浮点数的二进制存储格式：

$0100\ 0001\ 1010\ 0100\ 1100\ 0000\ 0000\ 0000=(41A4C000)_{16}$

浮点数的机器表示：例1

若IEEE754 32位浮点数 x 的二进制存储格式为 $(41360000)_{16}$，求浮点数的十进制值

解：将十六进制数展开，得到二进制格式

十六进制	4	1	3	6	0	0	0	0
二进制	0100	0001	0011	0110	0000	0000	0000	0000

$$\underset{S}{0}\underbrace{100\ 0001\ 0}_{\text{阶码(E)}}\underbrace{011\ 0110\ 0000\ 0000\ 0000\ 0000}_{\text{尾码(M)}}$$

$$x=(-1)^{S}\times 1.M\times 2^{E-127}$$
$$=(-1)^{0}\times(1.011011)\times 2^{10000010-01111111}$$
$$=+(1.011011)\times 2^{3}=+1011.011=(11.375)_{10}$$

IEEE P754标准32位浮点数定义

对32位浮点数N，**定义**：

- 若E=255且$M<>0$，则N＝NaN（无定义数据）
- 若E=255且M＝0，则N＝$(-1)^S\infty$（正无穷大，负无穷大）
- 若E=0且M＝0，则N＝ $(-1)^S0$（机器0）
- 若$0<E<255$，则N＝$(-1)^S(2)^{E-127}(1.M)$（规格化数）
- 若E=0且$M<>0$，则N＝ $(-1)^S(2)^{-126}(0.M)$（非规格化数）

【例9】浮点数的最值

假设由S、E、M三个字段组成的一个32位二进制字所表示的非零规格化浮点数 x，真值为：

$$x = (-1)^S \times (1.M) \times 2^{E-128}$$

问：它所表示的规格化的最大正数、最小正数、最大负数、最小负数是多少？

[解:]

(1)最大正数

0 11 111 111 111 111 111 111 111 111 111 11

$$x = [1 + (1 - 2^{-23})] \times 2^{127}$$

(2)最小正数

0 00 000 000 000 000 000 000 000 000 000 00

$$x = 1.0 \times 2^{-128}$$

【例6】浮点数的最值

假设由*S*、*E*、*M*三个字段组成的一个32位二进制字所表示的非零规格化浮点数 x，真值为：

$$x = (-1)^S \times (1.M) \times 2^{E-128}$$

问：它所表示的规格化的最大正数、最小正数、最大负数、最小负数是多少？

[解:]

(3)最小负数

1 11 111 111 111 111 111 111 111 111 111 11

$$x = -[1 + (1 - 2^{-23})] \times 2^{127}$$

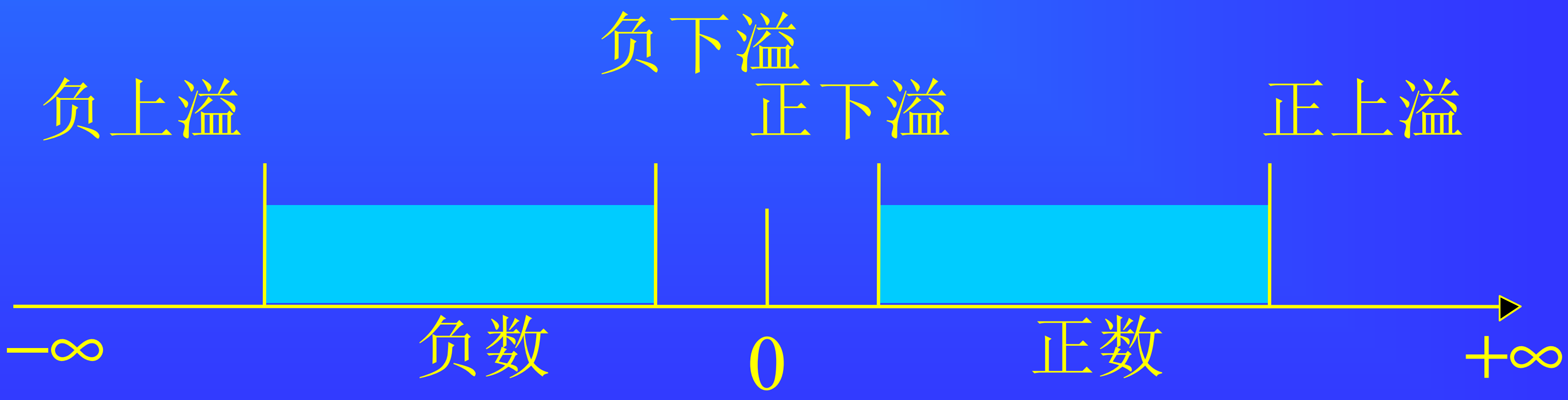
(4)最大负数

1 00 000 000 000 000 000 000 000 000 000 00

$$x = -1.0 \times 2^{-128}$$

溢出

- 定点数：正溢和负溢
  - 正溢：运算结果为正，且超出所能表示的范围
  - 负溢：运算结果为负，且超出所能表示的范围
- 浮点数：上溢和下溢
  - 上溢(overflow)：结果的绝对值大于所能表示的最大绝对值（+∞，－∞）
  - 下溢(underflow)：结果的绝对值小于所能表示的最小绝对值（机器零）



课堂练习

下列说法中正确的是（　　　　　　）。

A. 采用变形补码进行加减法运算可以避免溢出。

B. 只有定点数运算才有可能溢出，浮点数运算不会产生溢出。

C. -127的反码等于0的移码。

D. 只有将两个正数相加时才有可能产生溢出。

【解】

$(-127)_{反}=(0)_{移}=1\ 0000000$

C

浮点加减法运算

- 两个浮点数x和y:

$$x = 2^{E_x} \cdot M_x \quad y = 2^{E_y} \cdot M_y$$

- 浮点加减法的运算规则

$$x \pm y = (M_x 2^{E_x - E_y} \pm M_y) 2^{E_y} \qquad E_x \le E_y$$

- 浮点加减运算的操作过程
  - 第一步：0操作数检查
  - 第二步：比较阶码大小并完成对阶
  - 第三步：尾数进行加或减运算
  - 第四步：结果规格化、舍入和溢出处理

对阶

- **原则：**
  - 小阶向大阶看齐
- **方法：**
  - 逐位移位比较：小阶的尾数每次右移一位，其阶码加1，直到两数的阶码相等为止
  - 求阶差：
    - 右移的位数等于阶差 $\Delta E = E_x - E_y$
    - 哪个数移位由阶差的正负决定：
      - 若 $\Delta E = 0$：两数阶码相等
      - 若 $\Delta E > 0$：$E_x > E_y$；则y的尾数移位
      - 若 $\Delta E < 0$：$E_x < E_y$；则x的尾数移位

移码的加减法运算

- 移码的定义：
  - $[x]_{移}=2^n+x \qquad 2^n>x\geq-2^n$
- $[x]_{移}+[y]_{移}=2^n+x+2^n+y$

  $=2^n+(2^n+(x+y))=2^n+[x+y]_{移}$
- 要得到正确的移码形式结果：
  - $[x]_{移}+[y]_{移}$的符号位取反
- 混合使用移码和补码：$[x]_{移}+[y]_{补}$

移码的加减法运算法则

- 移码加法：
  - $[x]_{移}+[y]_{补}=2^n+x+2^{n+1}+y=2^{n+1}+(2^n+(x+y))$
    $=2^n+(x+y)=[x+y]_{移}$
  - 即 $[x+y]_{移}=[x]_{移}+[y]_{补}\ (\mathrm{mod}\ 2^{n+1})$
- 移码减法：
  - $[x-y]_{移}=[x]_{移}+[-y]_{补}$
- **双符号位阶码加法器**
  - **移码的高符号位恒为0**
  - **补码的两符号位相同**

双符号位	结果
00	负（无溢出）
01	正（无溢出）
10	正溢出
11	负溢出

结果规格化

- **左规：向左规格化**
  - 尾数为原码：应使结果为x.1xxxxx
  - 尾数为补码：应使符号位与最高有效位不相等
  - IEEE754浮点格式：应使尾数变为1.M形式
- **右规：向右规格化**
  - 若尾数求和的结果为01.x…x或10.x…x，应将运算结果右移以实现规格化表示
  - 规则：尾数右移1位，阶码加1

舍入处理

- 对阶或右规时，尾数右移使低位部分超出可表示位数
- 尾数多余位数的处理方法：
  - 截断处理
    - 无条件丢弃尾数保留的最低位之后的全部数值
    - 优点：处理简单
    - 缺点：影响结果的精度
  - 舍入处理
    - 目的：减小误差
    - 运算过程中保留右移移出的若干高位的值，最后再按某种规则用这些位的值修正尾数

IEEE 754标准中的舍入处理

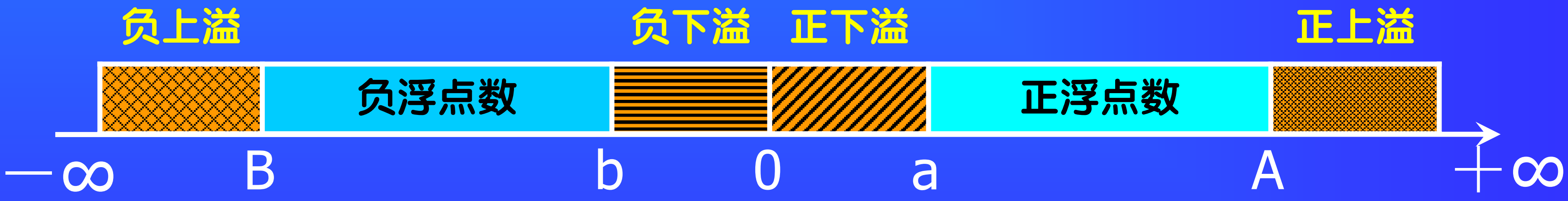
- **就近舍入："四舍五入"**
  - 例：若舍掉的数字是10010，则最低有效位加1
  - 例：若舍掉的数字是01111，则简单截尾
  - 例：若舍掉的数字是10000：
    - 若最低有效位现为0，则截尾
    - 若最低有效位现为1，则向上进一并使其变为0
- **朝0舍入：简单截尾**
  - 舍入后绝对值变小

IEEE 754标准中的舍入处理

- 朝+∞舍入：舍入后真值变大
  - 正数：只要舍弃位不全为0，则向最低有效位进1
  - 负数：简单截尾
- 朝-∞舍入：舍入后真值变小
  - 正数：简单截尾
  - 负数：只要舍弃位不全为0，则向最低有效位进1

溢出处理

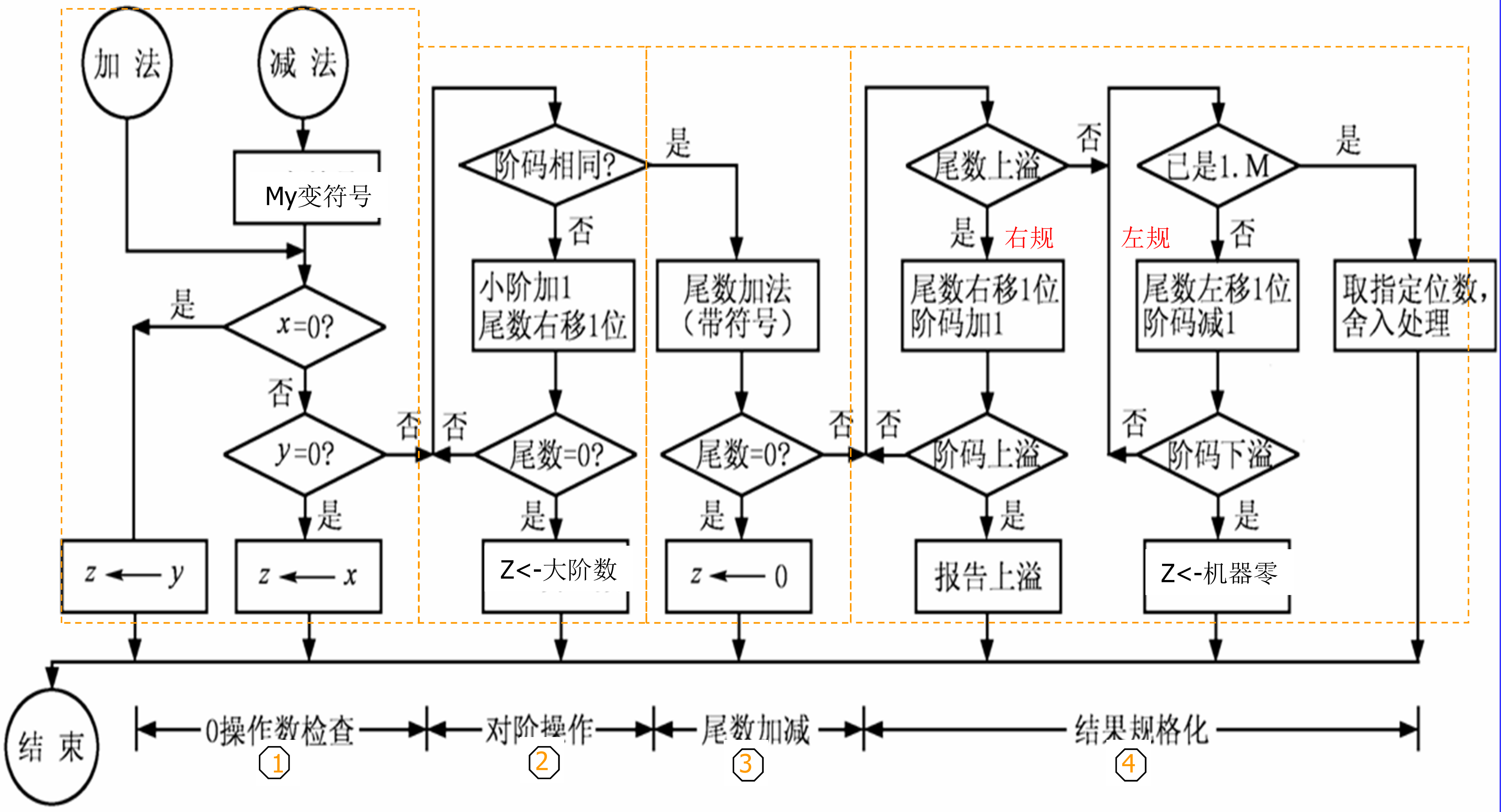
- 浮点数的溢出表现为其阶码的溢出
  - 若阶码正溢，则浮点数上溢
    - （绝对值太大：＋∞和－∞ ）
  - 若阶码负溢，则浮点数下溢
    - （过于靠近数轴原点：机器0）
- 【尾数上溢：右规 】



A—最大正数　　a—最小正数

B—最小负数　　b—最大负数

浮点加减运算的操作流程实例



浮点加减法运算

【例28】设 $x = 2^{010} \times 0.11011011$，

$y = 2^{100} \times (-0.10101100)$

求 x + y。设浮点数阶和尾数均以补码表示。

解：阶码采用双符号位，尾数采用单符号位

两浮点数可表示为

$[x]_{浮}$=00 010，0.11011011

$[y]_{浮}$=00 100，1.01010100

(1) 求阶差并对阶

$\Delta E = E_x - E_y = [E_x]_{补} + [-E_y]_{补}$

$= 00\ 010 + 11\ 100 = 11\ 110 = [-2]_{补}$

Mx右移2位，Ex加2：

$[x]_{浮}$=00 100，0.00110110(11)

浮点加减法运算

(2) 尾数求和

$$
\begin{array}{r}
0.00110110 \quad (11) \\
+\ 1.01010100 \\
\hline
1.10001010 \quad (11)
\end{array}
$$

(3) 规格化处理

左规：尾数为1.00010101(1)，阶码为00 011

(4) 舍入处理：　“0舍1入”

1.00010101＋0.00000001＝1.00010110

(5) 溢出判断

阶码符号位为00，无溢出

运算结果：$x+y = 2^{011} \times (-0.11101010)$

浮点乘法、除法运算

- 设两浮点数 x 和 y：
  - $x = 2^{Ex} \cdot Mx$， $y = 2^{Ey} \cdot M_y$
- 浮点乘法运算规则：
  - $x \times y = 2^{(Ex+Ey)} \cdot (M_x \times M_y)$
- 浮点除法运算规则
  - $x \div y = 2^{(Ex-Ey)} \cdot (M_x \div M_y)$
- 浮点数乘除运算步骤：
  - 0 操作数检查
  - 阶码加/减操作
  - 尾数乘/除操作
  - 结果规格化、舍入及溢出处理

浮点乘法、除法运算

[例30] 设 $x=2^{-5}\times 0.0110011$， $y=2^{3}\times(-0.1110010)$ 阶码用4位移码表示，尾数用8位补码表示。

求$[x\times y]_{浮}$。用间接补码运算器完成尾数乘法运算，运算结果尾数保留高8位，并对尾数低位值进行舍入处理。

[解:]

移码采用双符号位，尾数补码采用单符号位

$[M_x]_{补}=0.0110011$，$[M_y]_{补}=1.0001110$

$[E_x]_{移}=00\ 011$，$[E_y]_{移}=01\ 011$，$[E_y]_{补}=00\ 011$

$[x]_{浮}=00\ 011, 0.0110011$

$[y]_{浮}=01\ 011, 1.0001110$

浮点乘法、除法运算

[解:]

$[M_x]_补 = 0.0110011$，$[M_y]_补 = 1.0001110$

$[E_x]_移 = 00\ 011$，$[E_y]_移 = 01\ 011$，$[E_y]_补 = 00\ 011$

$[x]_浮 = 00\ 011, 0.0110011$；$[y]_浮 = 01\ 011, 1.0001110$

(1) 阶码求和

$$[E_x + E_y]_移 = [E_x]_移 + [E_y]_补 = 00\ 011 + 00\ 011 = 00\ 110$$

浮点乘法、除法运算

[解:] $[M_x]_{补}=0.0110011$，$[M_y]_{补}=1.0001110$

$[E_x]_{移}=00\ 011$，$[E_y]_{移}=01\ 011$ ，$[E_y]_{补}=00\ 011$

$[x]_{浮}=00\ 011, 0.0110011$； $[y]_{浮}=01\ 011, 1.0001110$

(2) 尾数乘法运算

$$[M_x]_{补}\times[M_y]_{补}=[0.0110011]_{补}\times[1.0001110]_{补}$$

$[M_y]$算前求补： $[M_y]_{原}=1.1110010$

尾数绝对值进行无符号数乘法：

$$0.0110011\times0.1110010=0.01\ 0110\ 1011\ 0110$$

算后求补：$[M_x\times M_y]_{原}= 1.01\ 0110\ 1011\ 0110$

$$[M_x\times M_y]_{补}= 1.10\ 1001\ 0100\ 1010$$

浮点乘法、除法运算

[解:]

$[M_x]_{补}=0.0110011$，$[M_y]_{补}=1.0001110$

$[E_x]_{移}=00\ 011$，$[E_y]_{移}=01\ 011$，$[E_y]_{补}=00\ 011$

$[x]_{浮}=00\ 011, 0.0110011$

$[y]_{浮}=01\ 011, 1.0001110$

(3) 规格化处理

$[M_x \times M_y]_{补}=1.10\ 1001\ 0100\ 1010$

左规，阶码：00 101；尾数：1.0100101 0010100

(4) 舍入处理：尾数＝1.0100101

$[x \times y]_{浮}=00\ 101,1.0100101$

$x \times y=2^{-3} \times (-0.1011011)$

第二章小结

- 数据的表示方法及其机器存储：
  - 便于存储、便于运算
- 算术运算和逻辑运算的运算方法
- 运算器的组成（实现）
  - 性能与成本的平衡

本章重点

- 数值数据的表示方法：
  - 机器数的表示范围、精度和特点
  - 定点数的机器码各种码制之间的转换
  - 浮点数IEEE754标准格式
- 定点算术运算的实现
  - 溢出的概念，溢出的检测方法
  - 加减法的统一
  - 先行进位
  - 运算器的硬件复杂度和性能
  - 定点运算器的组成和结构
- 浮点运算方法

课堂练习

IEEE 754标准规定的32位浮点数格式中，符号位为1位，阶码为8位，尾数为23位。则它所能表示的最大规格化正数为（　　　）。

A. $+(2-2^{-23})\times 2^{+127}$　　B. $+(1-2^{-23})\times 2^{+127}$

C. $+(2-2^{-23})\times 2^{+255}$　　D. $2^{+127}-2^{-23}$

【解析】 $x=(-1)^S \times (1.M) \times 2^{E-127}$

- 若E=255且M<>0，则N=NaN（无定义数据）
- 若E=255且M=0，则N=$(-1)^s\infty$

$1.111\ldots\ldots1 \times 2^{254-127}$

【解】 A

北京郵電大学

作业

1. 主教材p61：6
2. 已知x=15，y=－13，请用间接补码乘法计算xy。
3. 设有浮点数$x=2^5\times(+9/16)$，$y=2^3\times(-13/16)$，阶码用4位（含一位符号位）移码表示，尾数用5位（含一位符号位）原码表示，求真值x/y=？ 要求写出完整的浮点运算步骤，并用原码加减交替法完成尾数除法运算。
4. IEEE754标准规定的64位浮点数格式中，符号位为1位，阶码为11位，尾数为52位。则它所能表示的最小规格化负数为（　　　）。

计算机组成原理

Principle of Computer Organization

➤ 第二章 运算方法与运算器

本章结束