计算机组成与系统结构

第二章 运算方法和运算器（3）

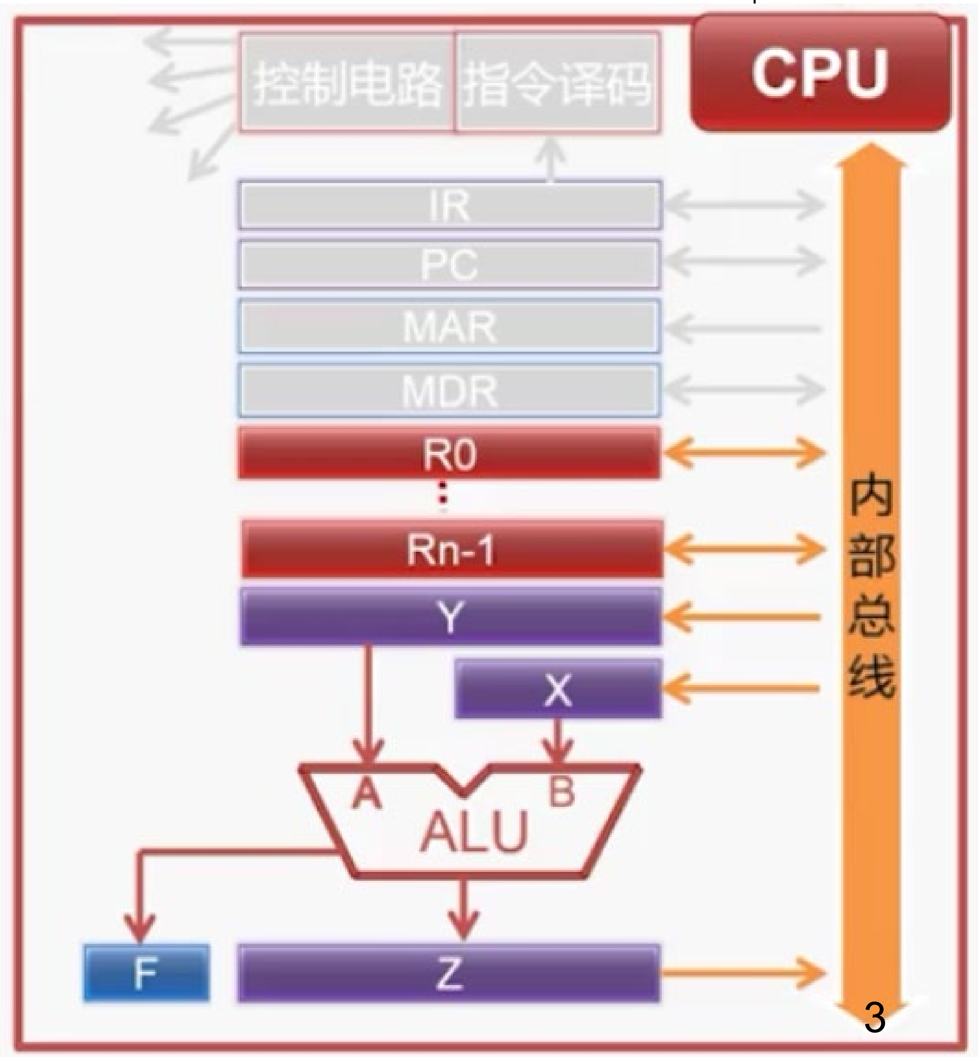
吕昕晨
lvxinchen@bupt.edu.cn

网络空间安全学院

运算方法与运算器

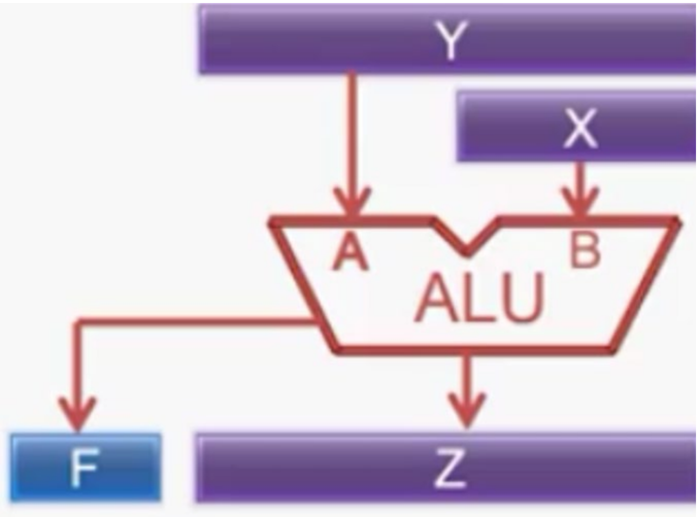
模型机—CPU运算器

- 运算器用于进行算数运算和逻辑运算
- 算数运算
  - 加、减、乘、除
- 逻辑运算
  - 非、与、或
- 数表示方式
  - 定点数
  - 浮点数



第二章剩余教学安排

- 本节课
  - 定点数（理论+硬件）
    - 乘法（重点）
    - 除法（重难点）
- 下节课
  - 浮点数
    - 加、减、乘、除法
  - 运算器总线结构



第二章 运算方法和运算器

- 串行移位乘法器（扩展）
- 并行阵列乘法器
- 带符号数乘法
- 定点除法运算

硬件设计

问题引入：十进制与二进制乘法

- 例，x=13，y=14，求x × y=?
- 十进制方法

$$\begin{array}{lccccl}
 & & 1 & 3 & & (x) \\
\times & & 1 & 4 & & (y) \\
\hline
 & & 5 & 2 & & \\
+ & 1 & 3 & & & \\
\hline
 & 1 & 8 & 2 & & (z)
\end{array}$$

- 二进制方法

$$\begin{array}{lccccccccl}
 & & & & & 1 & 1 & 0 & 1 & (x) \\
\times & & & & & 1 & 1 & 1 & 0 & (y) \\
\hline
 & & & & & 0 & 0 & 0 & 0 & \\
 & & & & 1 & 1 & 0 & 1 & & \\
 & & & 1 & 1 & 0 & 1 & & & \\
+ & & 1 & 1 & 0 & 1 & & & & \\
\hline
 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 &
\end{array}$$

- 10110110=0+2+4+0+16+32+0+128=182
- 思考：有何异同？有何规律？

单选题 1分

已知不带符号的二进制整数A=11011，B=10101，求A × B?

A 1 0 0 1 1 1 0 1 1 1

B 1 0 0 0 1 1 0 1 1 1

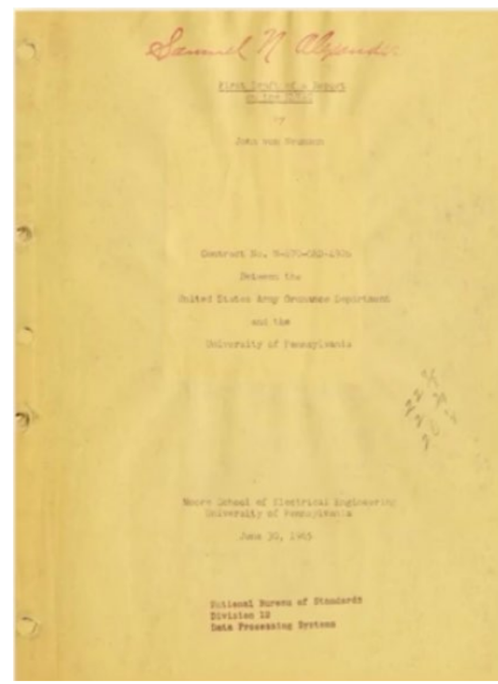
C 1 0 0 0 1 0 0 1 1 1

D 1 1 0 0 1 1 0 1 1 1

回顾：EDVAC报告草案

- 电子管是一种“全或无”设备（all-or-none）
  - 适合表示只有两个数值的系统，即二进制
- 二进制可以大幅度地简化乘法和除法的运算过程
  - 尤其是对于乘法，不再需要十进制乘法表
- 十进制才是适合人使用的
  - 输入输出设备需要承担二进制与十进制转

约翰·冯·诺依曼
John Von Neumann
1903~1957



Between the
United States Army Ordnance Department
and the
University of Pennsylvania

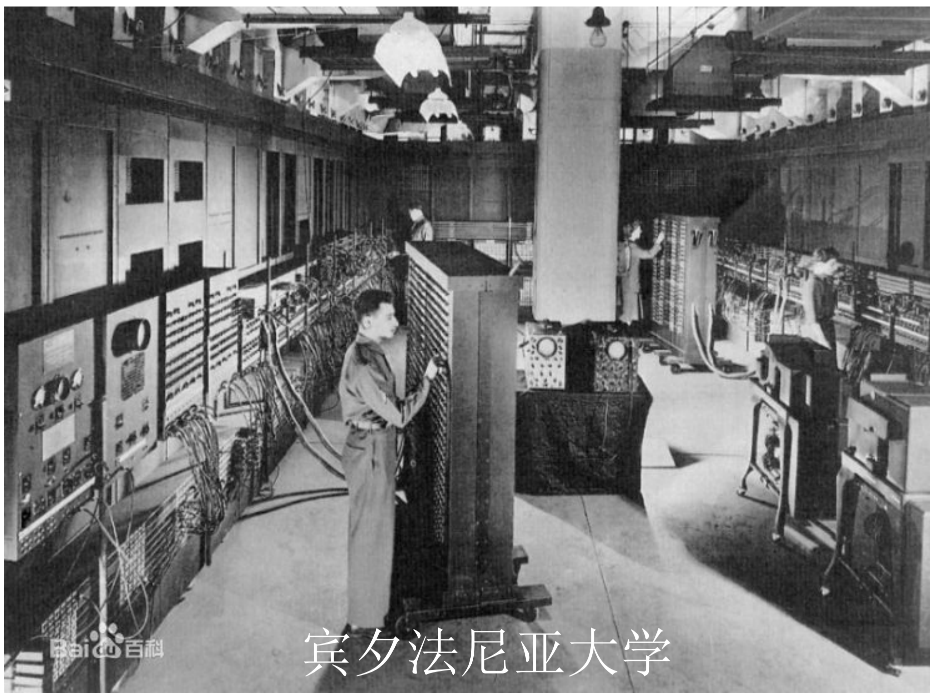
Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

National Bureau of Standards
Division 12
Data Processing Systems

埃尼阿克与EDVAC

- 十进制与二进制计算机系统
  - 埃尼阿克 → 十进制
  - EDVAC → 二进制



宾夕法尼亚大学

美国弹道研究院

二进制乘法规律总结

- 二进制乘法规律
  - 如果当前乘数位为“1”
    - 将被乘数抄写对应位置
    - 被乘数左移、乘数右移
  - 如果当前乘数位为“0”
    - 将全“0”放置于对应位
  - 对应位求和
- 区别：
  - 十进制复杂 （九九乘法表、加减法进位）
  - 二进制便捷 （判断是否为0、抄写）
  - 冯诺依曼体系采用二进制重要原因

	1 1 0 1	(被乘数)
×	1 1 1 0	(乘数)
	0 0 0 0	
	1 1 0 1	
	1 1 0 1	
+	1 1 0 1	
	1 0 1 1 0 1 1 0	(乘积)

请复述二进制乘法手算规则：

- 二进制乘法规律——**移位求和**
  - 如果当前乘数位为“1”
    - 将被乘数抄写对应位置
    - 被乘数左移、乘数右移
  - 如果当前乘数位为“0”
    - 将全“0”放置于对应位
  - 对应位求和

```
          1 1 0 1   (被乘数)
×         1 1 1 0   (乘数)
-------------------
          0 0 0 0
        1 1 0 1
      1 1 0 1
+   1 1 0 1
-------------------
  1 0 1 1 0 1 1 0   (乘积)
```

二进制乘法—串行移位

- 实现目标
  - 节约硬件资源
  - 复用加法器
- 移位运算
  - 被乘数：左移
  - 乘数：右移
- 操作规则
  - 若乘数最低位为“1”
    - 乘积+=被乘数
  - 否则，空操作

	1101	(被乘数)
×	1110	(乘数)
移位过程	0000	
	11010	
	110100	
+	1101000	
求和结果	0000	
	11010	
	1001110	
	10110110	(乘积)

串行乘法器程序—位运算

```
int multiply(int a, int b) {
    //将乘数和被乘数都取绝对值
    int multiplicand = a < 0 ? add(~a, 1) : a;
    int multiplier = b < 0 ? add(~b , 1) : b;

    //计算绝对值的乘积
    int product = 0;
    while(multiplier > 0) {
        if((multiplier & 0x1) > 0) {//每次考察乘数的最后一位
            product = add(product, multiplicand);
        }
        multiplicand = multiplicand << 1;//每运算一次，被乘数要左移一位
        multiplier = multiplier >> 1;//每运算一次，乘数要右移一位（可对照上图理解）
    }
    //计算乘积的符号
    if((a ^ b) < 0) {
        product = add(~product, 1);
    }
    return product;
}
```

结果寄存器

第1步：product = add(product, multiplicand);

第2步：multiplicand = multiplicand << 1;

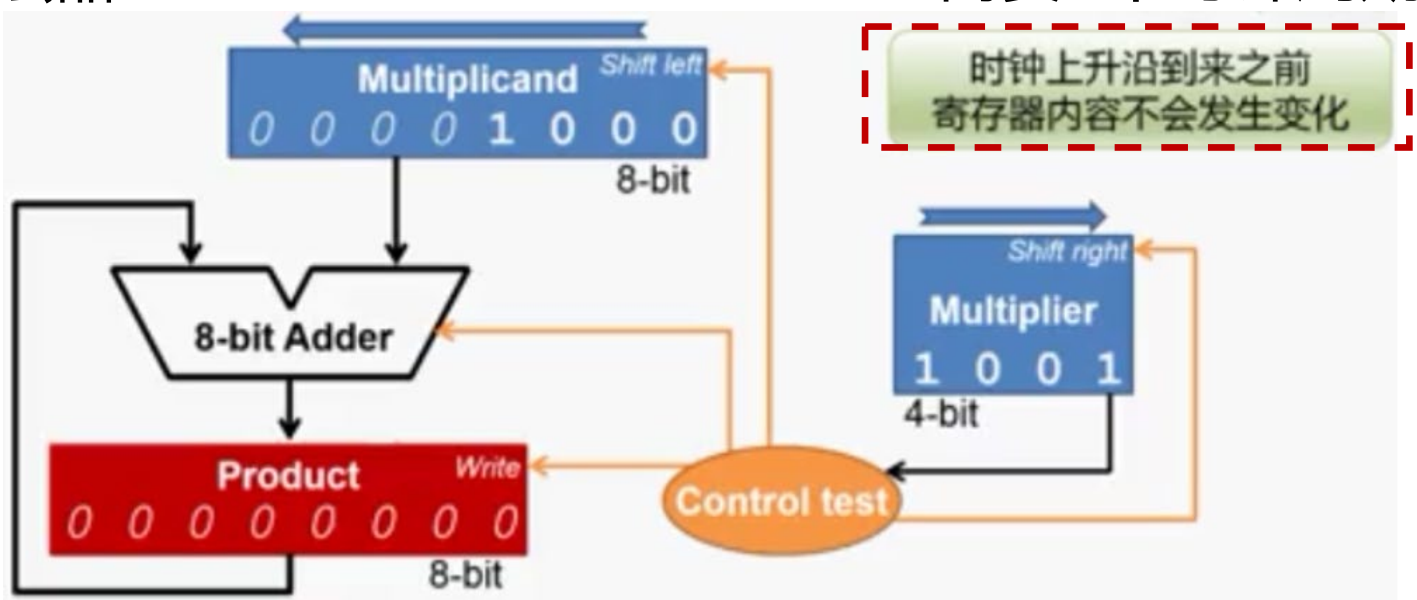
第3步：multiplier = multiplier >> 1;

```
              1 1 0 1   (被乘数)
×             1 1 1 0   (乘数)
--------------------------
              0 0 0 0
            1 1 0 1
          1 1 0 1
+       1 1 0 1
--------------------------
      1 0 1 1 0 1 1 0   (乘积)
```

- 对于n位乘法器（乘数）
  - 需要3n个时钟周期

扩展：串行移位乘法器—效率

- 4位乘法器组成
  - 4位乘数寄存器（右移）
  - 8位被乘数寄存器（左移）
  - 8位乘积寄存器
  - 加法器
- 优化执行流程
  - 单一时钟周期
    - 移位、相加操作（同时）
  - 对于n位乘法器（乘数）
    - 需要n个时钟周期



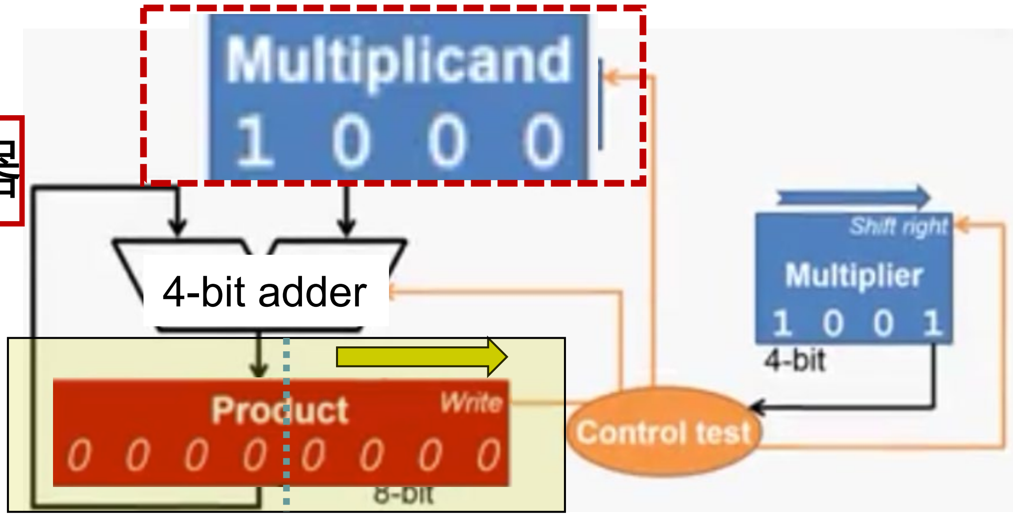
串行移位乘法器及优化—面积1

- 组成
  - 4位乘数寄存器（右移）
  - 8位被乘数寄存器（左移）
  - 8位乘积寄存器
  - 加法器

面积优化1

4位被乘数寄存器
8位乘积寄存器（右移）

4位加法器

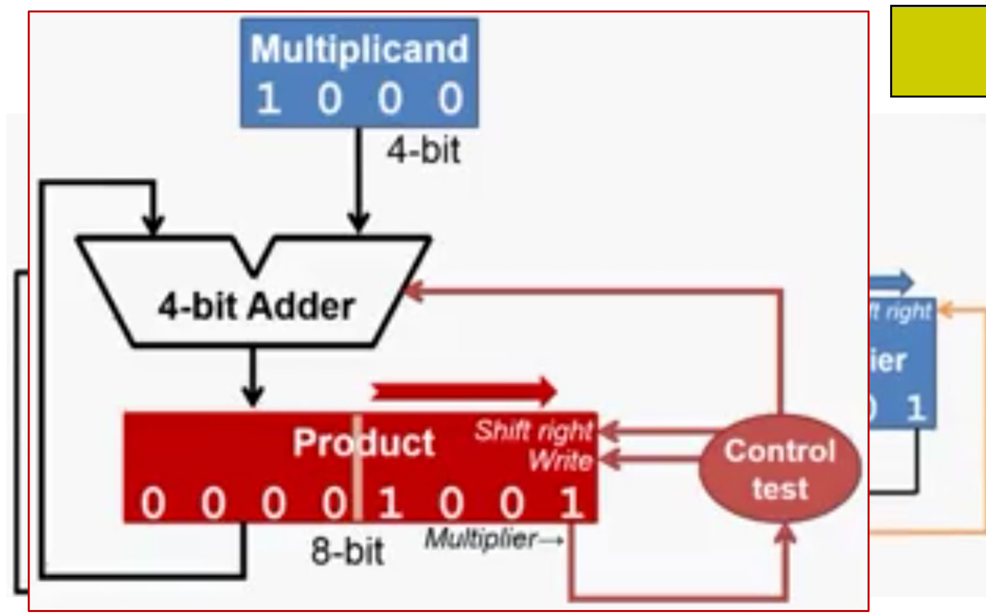


串行移位乘法器及优化—面积2

- 组成
  - 4位乘数寄存器（右移）
  - 4位被乘数寄存器
  - 8位乘积寄存器（右移）
  - 加法器

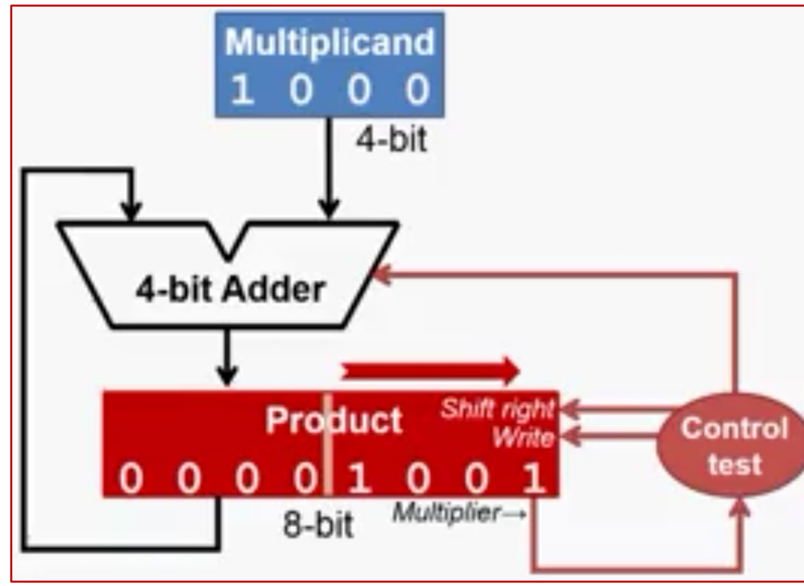
面积优化2

8位乘积寄存器（右移）



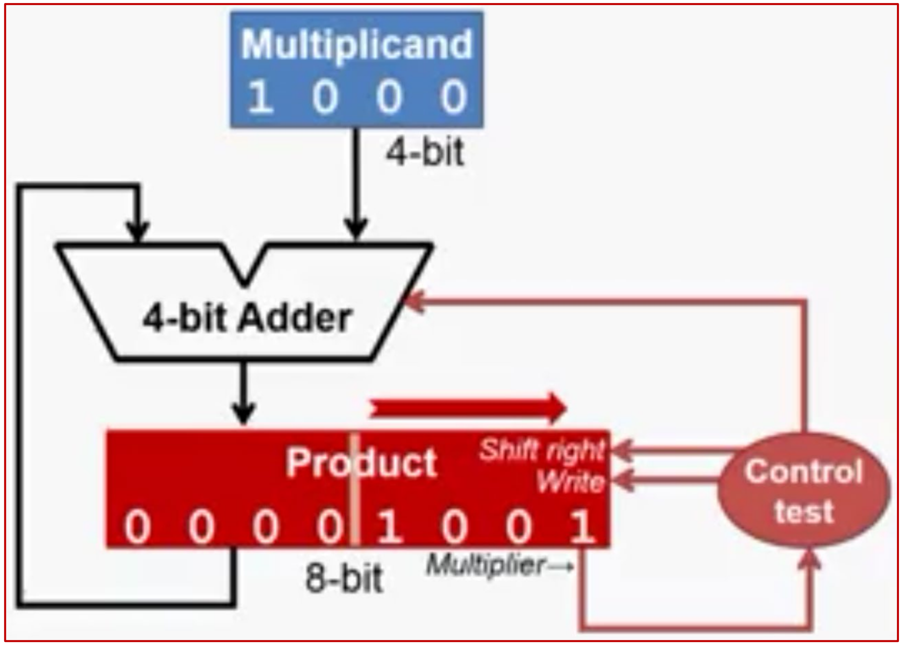
串行移位乘法器优化总结

- 效率优化（减少延迟）
  - 相加、左移、右移可同时进行（寄存器特点）
- 面积优化（减少不必要硬件）
  - 乘积寄存器增加右移功能，乘积初始值置于其中高4位，随着运算过程不断右移
  - 取消乘数寄存器，乘数初始置于“乘积寄存器”低4位
  - 加法器缩减为4位宽，乘积寄存器只有高4位参与运算



串行移位乘法器分析

- N位乘法器结构特点
  - N位被乘数寄存器
  - 2N位乘积寄存器（右移）
    - 高N位→加法器输出
    - 低N位→乘数
  - N位加法器
- 优点
  - 结构简单、易于实现
  - 复用加法器功能
- 思考：串行乘法器问题?
  - 效率较低，N个时钟周期



第二章 运算方法和运算器

- 串行移位乘法器
- 并行阵列乘法器

硬件设计

- 带符号数乘法
- 定点除法运算

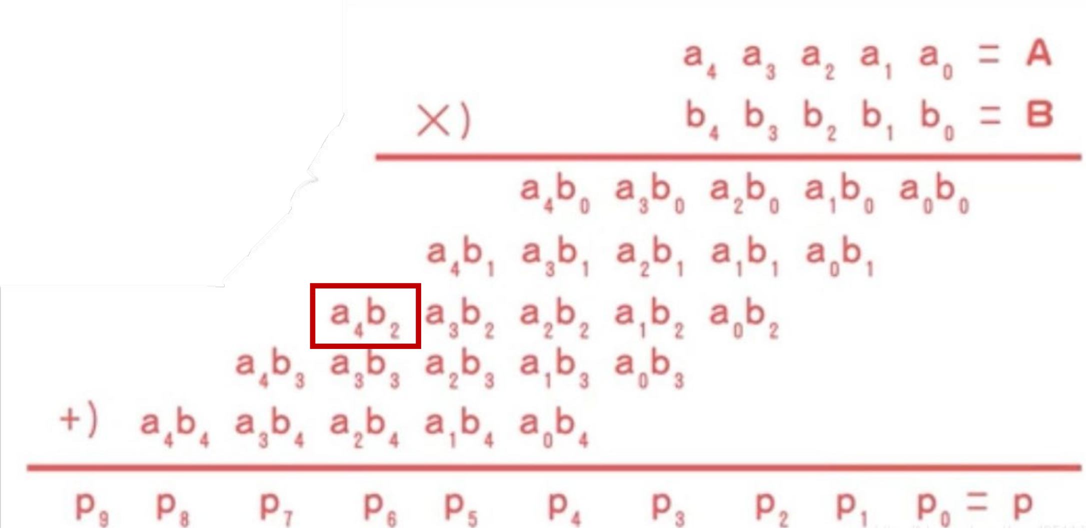
并行阵列乘法器

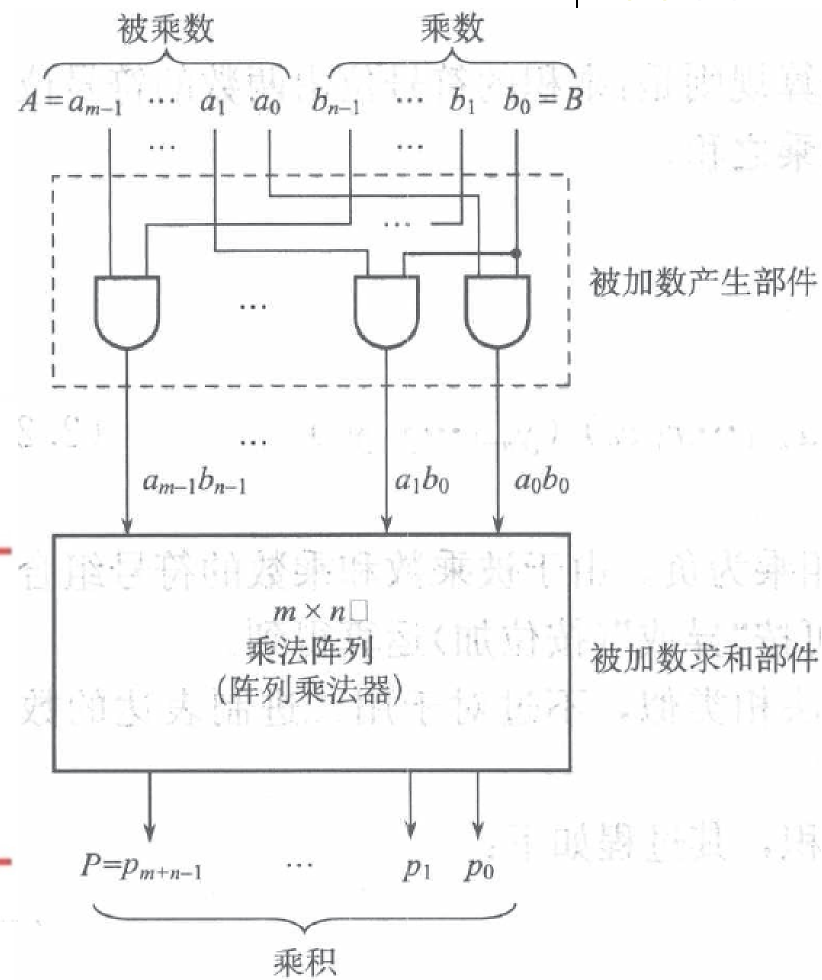
- 优化思路
  - 去掉移位过程（N个时钟周期）
  - 通过乘数与被乘数直接产生所有中间数据
  - 重新组织全加器，实现乘积求和

```
          1 1 0 1   (被乘数)
 ×        1 1 1 0   (乘数)
 ------------------
          0 0 0 0
        1 1 0 1
      1 1 0 1
 +  1 1 0 1
 ------------------
  1 0 1 1 0 1 1 0   (乘积)
```

并行阵列乘法器—被加数生成

- 被加数产生部件
  - 与操作：交叉输入
  - a_i b_j 对应 第j行/第（i+j）列





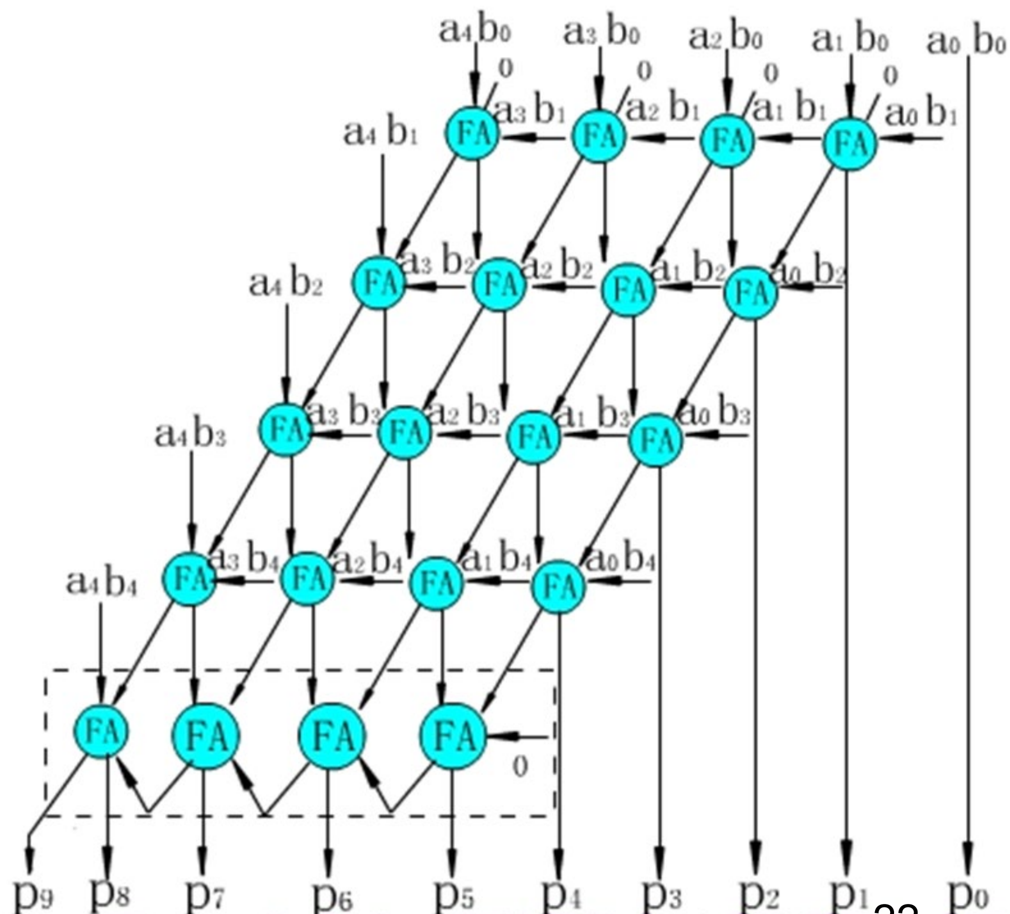
  - 电路组成：m*n个与门

乘法阵列：阵列全加器组合，实现乘积求和功能

并行阵列乘法器—乘法阵列

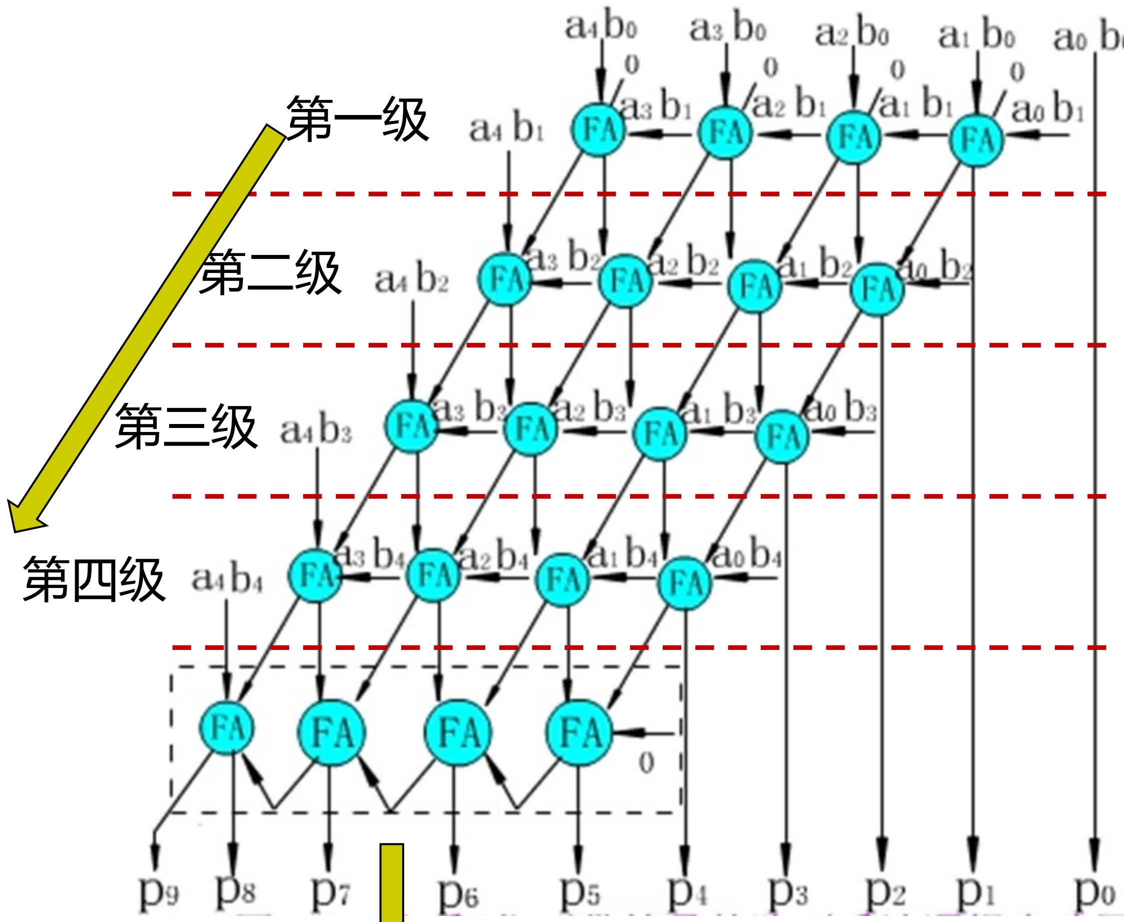
- 结构特点
  - 排列方式与手写相同
- 全加器输出
  - 斜线：进位输出
  - 竖线：和输出
  - N(N-1)个全加器

```
          1 1 0 1   (被乘数)
   ×      1 1 1 0   (乘数)
   ------------------
          0 0 0 0
        1 1 0 1
      1 1 0 1
 +  1 1 0 1
 ------------------
  1 0 1 1 0 1 1 0   (乘积)
```



并行阵列乘法器延迟分析（1）

- 全加器延迟
  - 和输出：6T
    (两级异或门)
  - 进位输出：2T
    (两级与/或门)
- 斜线阶段求和
  - 单级延迟：6T
  - 总延迟：(N-1)*6T
- 行波进位加法器
  - 延迟：(N-1)*2T+3T



（N-1）位行波进位加法器

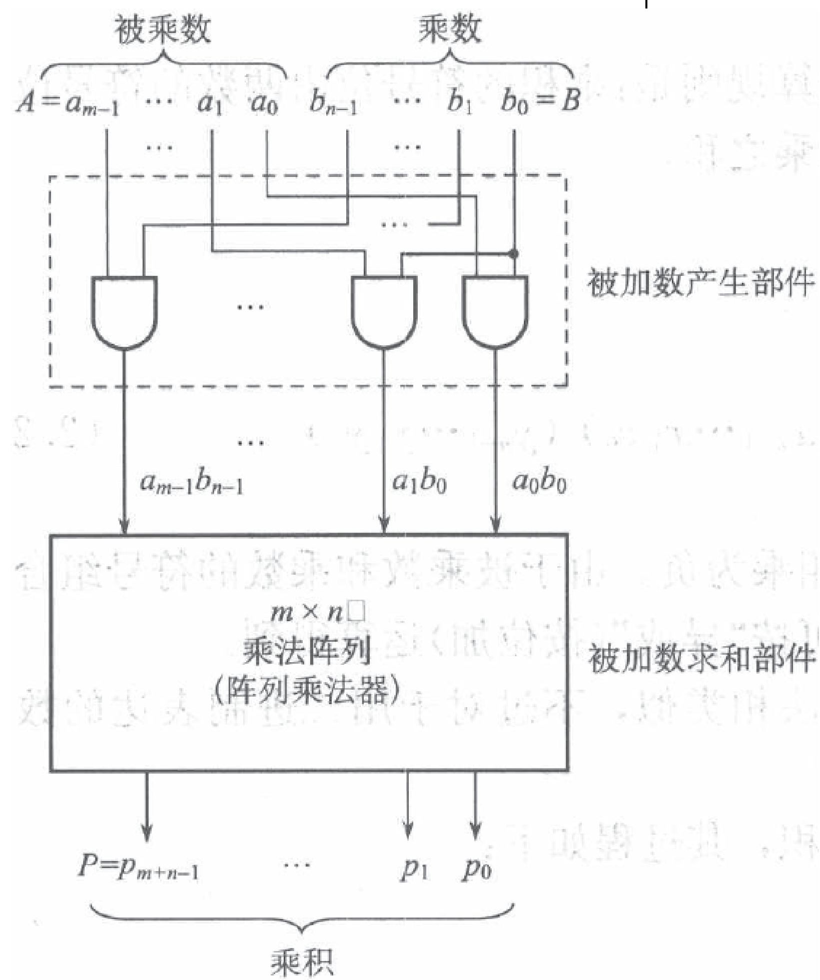
并行阵列乘法器延迟分析（2）

- 被加数生成
  - 延迟：T（与门）

$$T + (n-1)6T + (N-1)2T + 3T$$

总延迟 $= (8n-4)\mathrm{T}$

- 乘法阵列
  - 阶段求和：(N-1)*6T
  - 行波进位：(N-1)*2T+3T

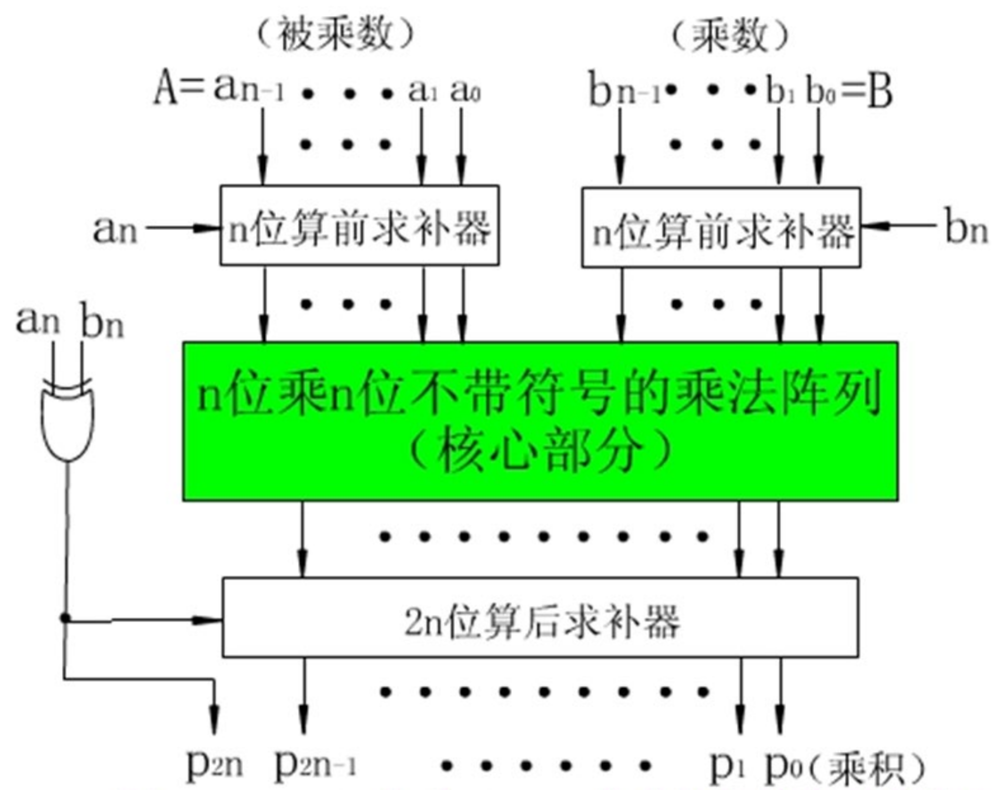


第二章 运算方法和运算器

- 串行移位乘法器
- 并行阵列乘法器
- 带符号数乘法（重点）
- 定点除法运算

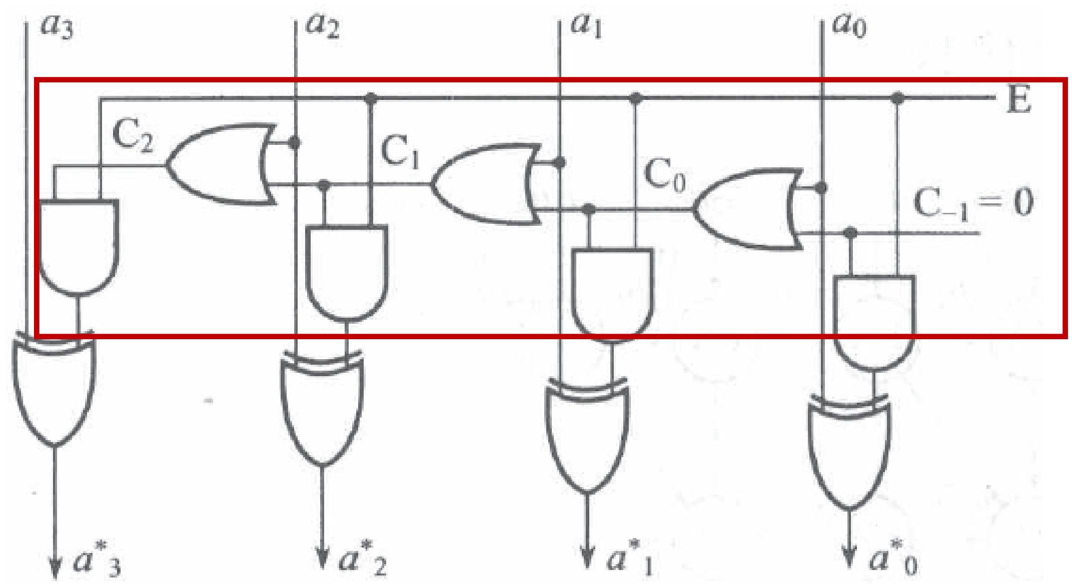
有符号数存储方式——补码

- 思路——转换原码
- 补码性质
  - $[\ [A]_{补}\ ]_{补} = [\ A\ ]_{原}$
- 带符号乘法器构思路
  - 算前求补
  - 乘法器
  - 算后求补
  - 注意：符号位处理



对2求补电路

- 功能：最右端往左边扫描，直到第一个1的时候，该位和右边各位保持不变，左边各数值位按位取反
- 功能组成
  - 按位扫描
    - 或门级联
  - 逐位取反(异或门)



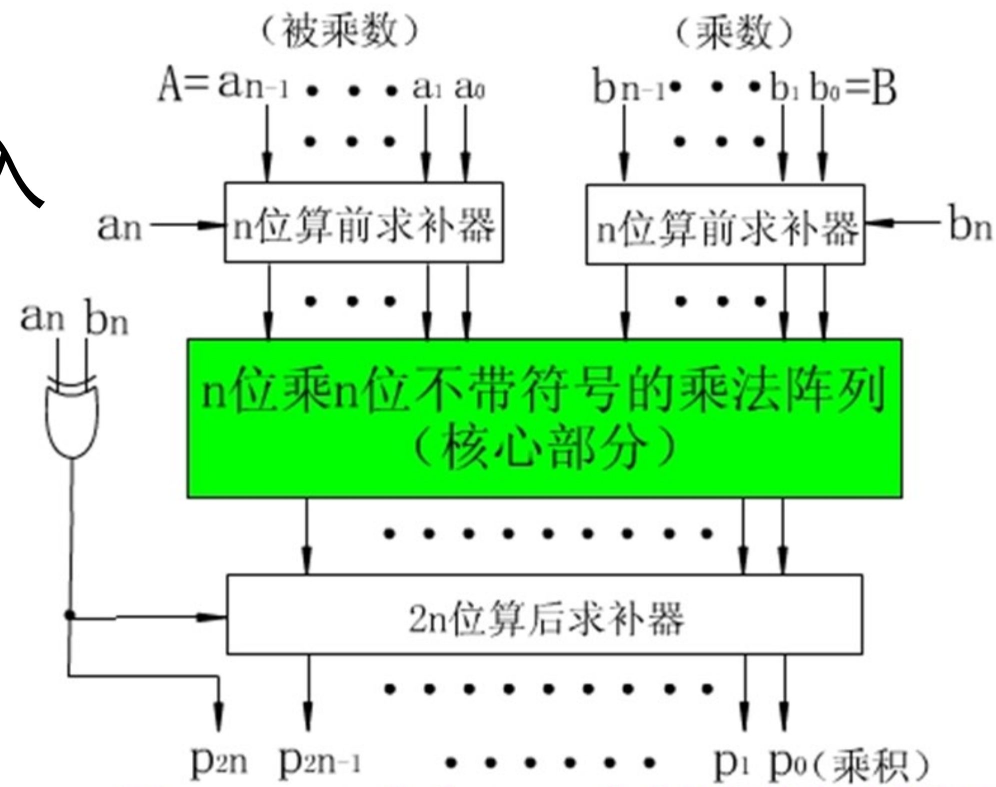
- 逻辑表达式

$$C_{-1}=0,\quad C_i=a_i+C_{i-1}$$

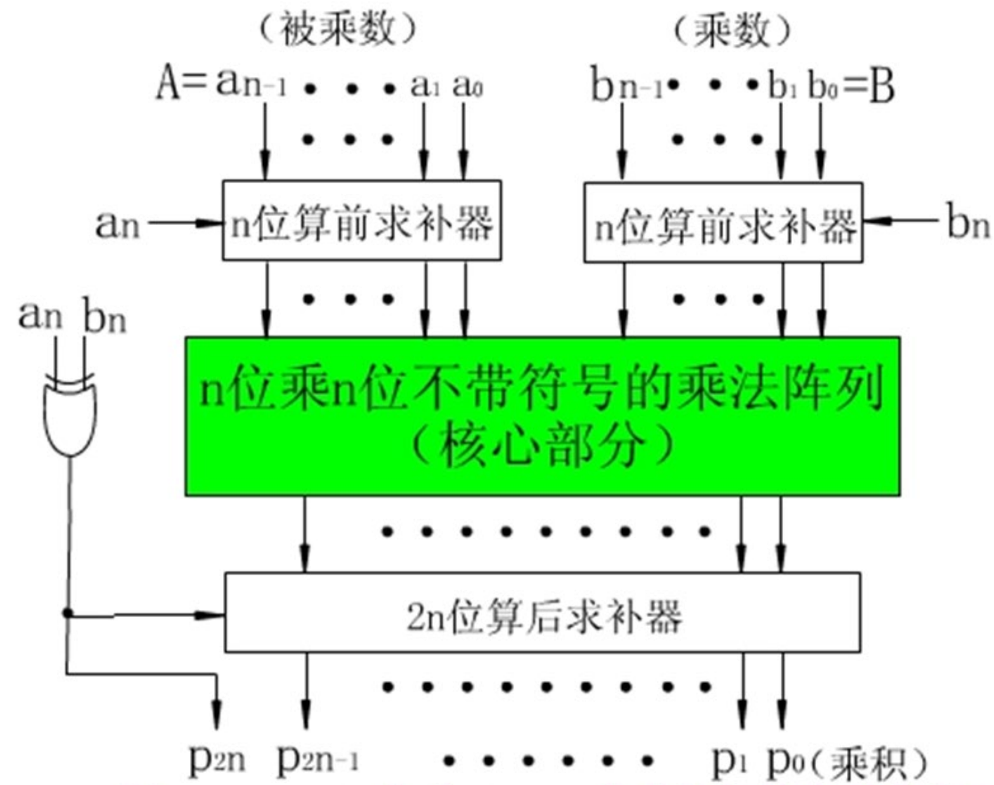
$$a_i^*=a_i\oplus EC_{i-1},\qquad 0\leqslant i\leqslant n$$

带符号乘法器——求解总结

- 求解步骤
  - 判断原码/补码输入
  - 符号位计算
  - （算前求补）
  - 乘法（不带符号）
  - （算后求补）



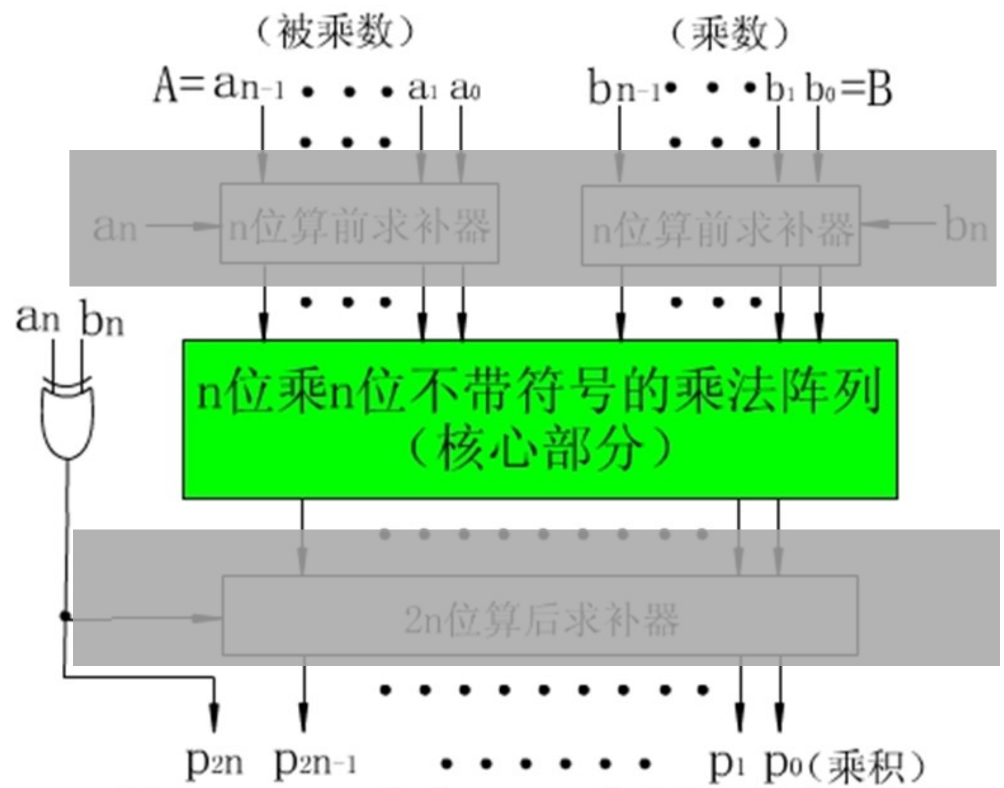
习题[2-7-1)] 已知x=11011，y=-11111
用原码阵列乘法器、补码阵列乘法器，计算x×y?



习题[2-7-1)] 已知x=11011，y=-11111
用原码阵列乘法器、补码阵列乘法器，计算x×y?

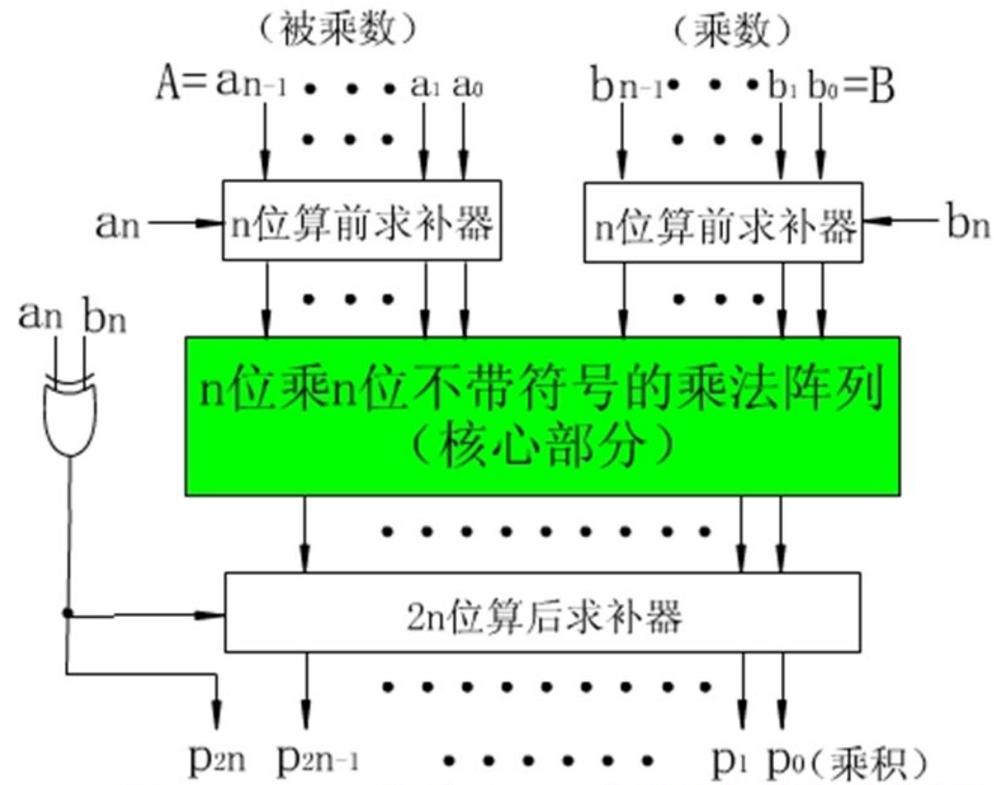
- 原码乘法（输入原码、输出原码）
- 符号位：0⊕1=1
- 无需算前算后求补
- |x|=11011，|y|=11111
- 乘法:

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



- $[x \times y]_{原}$ = 1 1101000101

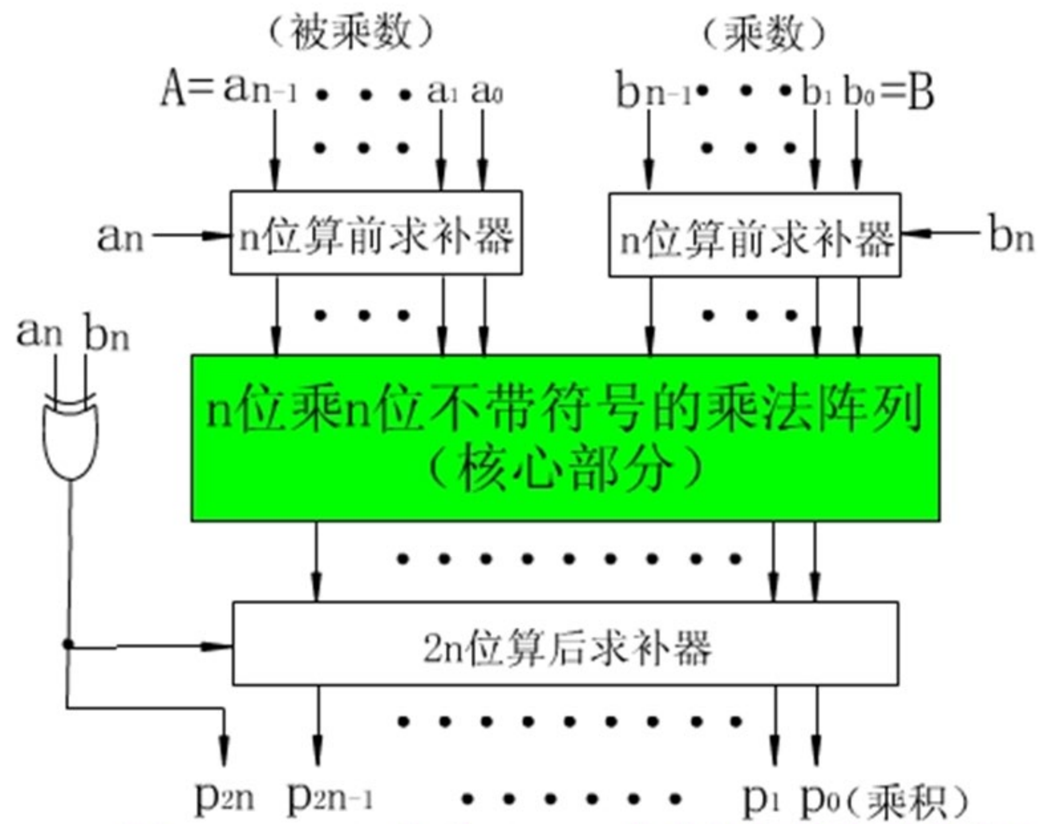
习题[2-7-1)] 已知x=11011，y=-11111
用原码阵列乘法器、补码阵列乘法器，计算x×y?



习题[2-7-1)] 已知x=11011，y=-11111
用原码阵列乘法器、补码阵列乘法器，计算x×y?

- 补码乘法（输入补码、输出补码）
- [x]补=0 11011，[y]补=100001
- 符号位：0⊕1=1
- 算前求补
- |x|=11011，|y|=11111
- 乘法:

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



- $[x \times y]_{原}$ = 1 1101000101
- 算后求补 $[x \times y]_{补}$ = 1 0010111011

第二章 运算方法和运算器

- 串行移位乘法器
- 并行阵列乘法器
- 带符号数乘法
- 定点除法运算（重难点）

二进制除法—手算过程

0.1 1 0 1
0.1 1 0
0.1 1
0.1

商q

0.1 0 1 1

除数

- 除法规则
  - 比较被除数与除数大小
    - 若够减，对应位商1
    - 若不够减，对应位商0
  - 除数右移
  - 终止条件：余数<除数?

\- 0.0 0 0 0 0 0 0 1 r_4

被除数

除数右移1位,减除数

得余数r1

除数右移1位,减除数

得余数r2

除数右移1位,不减除数

得余数r3

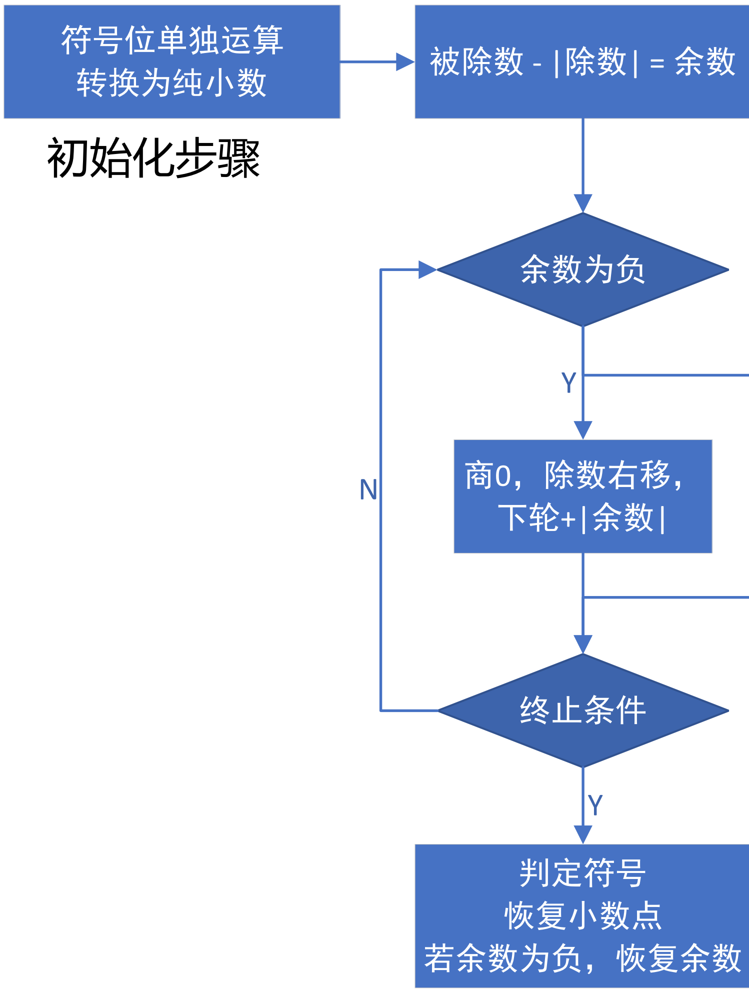
除数右移1位,减除数

得余数r4

计算机除法流程

- 人工除法时，人可以比较被除数（余数）和除数的大小来确定商1（够减）或商0（不够减）
- 机器除法时，余数为正表示够减，余数为负表示不够减。不够减时必须恢复原来余数，才能继续向下运算。这种方法叫恢复余数法，控制比较复杂。
- 不恢复余数法（加减交替法——重点）
  - 余数为正，商1，下次除数右移做减法
  - 余数为负，商0，下次除数右移做加法

加减交替法—流程图（重点）



第一次先做减法

加减交替流程

- 判定商与加减
- 全程补码运算
- 带符号右移

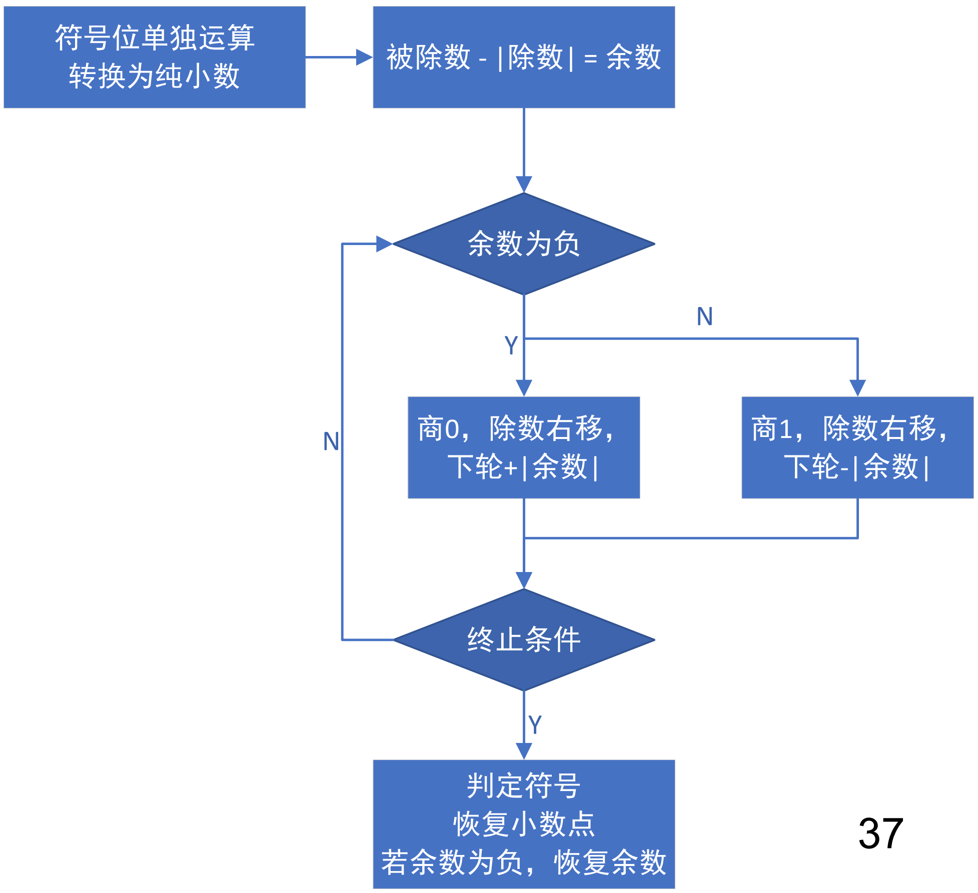
终止条件：商已达到机器字长

- 字长：一般除数位数

- 恢复余数：最后一次正余数

加减交替法例题1

[例23] $x=0.101001$, $y=0.111$, 求 $x \div y$。（采用加减交替法）



加减交替法例题1

[例23] x =0.101001, y =0.111, 求 $x \div y$。

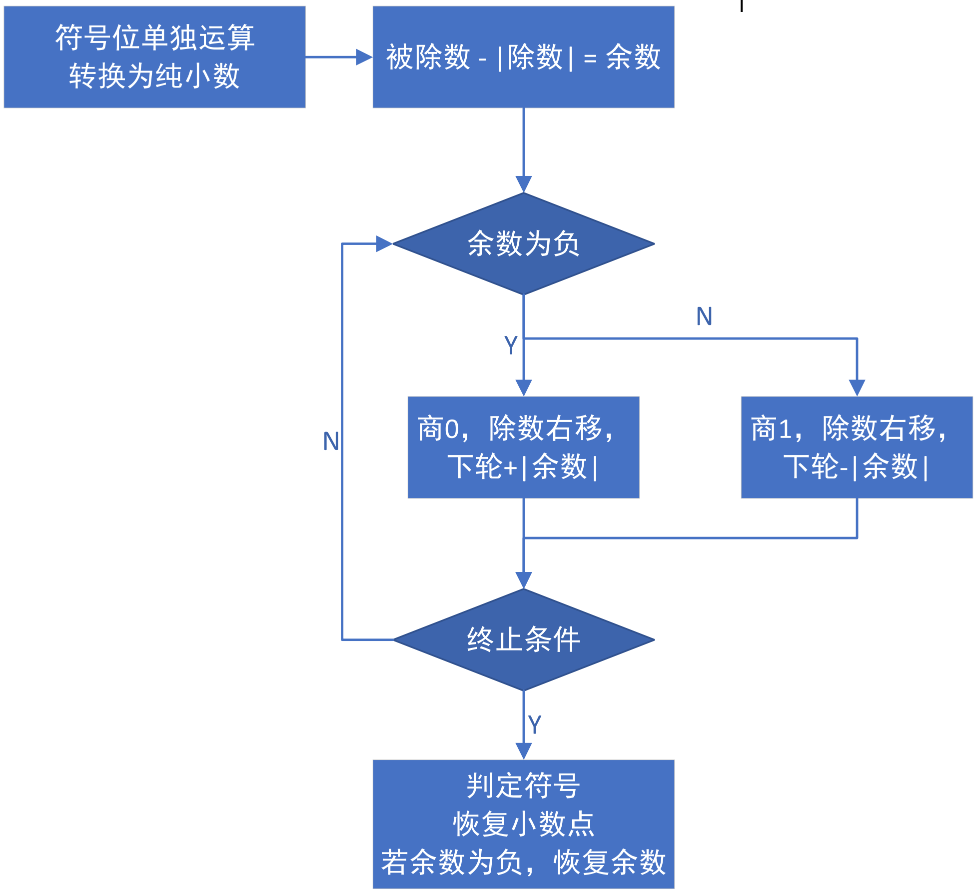
[解:] $[x]_{补}$=0.101001 , $[y]_{补}$=0.111 , $[-y]_{补}$=1.001

		0.1 0 1 0 0 1		；	被除数
起始位置	$+[-y]_{补}$	1.0 0 1	小数点后3位 y=0.111	；	第一步减除数y
		1.1 1 0 0 0 1	<0 q_0=0	；	余数为负,商0
	$+[y]_{补}\rightarrow$	0.0 1 1 1		；	除数右移1位加
		0.0 0 1 1 0 1	>0 q_1=1	；	余数为正,商1
	$+[-y]_{补}\rightarrow$	1.1 1 0 0 1		；	除数右移2位减
		1.1 1 1 1 1 1	<0 q_2=0	；	余数为负,商0
	$+[y]_{补}\rightarrow$	0.0 0 0 1 1 1		；	除数右移3位加
	补码	0.0 0 0 1 1 0	>0 q_3=1	；	余数为正,商1

真值

商 $q=q_0.q_1q_2q_3$=0.101 , 余数r=0.000110

习题[2-8-1)] $x = 11000$, $y = -11111$, 用原码除法，求 $x \div y$。 **(采用加减交替法)**



习题[2-8-1)] $x=11000$, $y=-11111$, 用原码除法，求 $x \div y$。

- 符号位： $0 \oplus 1=1$
- |x|=11000，|y|=11111
- 纯小数表示，小数点左移5位，|x|=0.11000，|y|=0.11111
- [| x |]=0.11000, [| y |]补=0.11111 , [- | y |]补=1.00001

小数点后5位

	0.1 1 0 0 0		； 被除数
$+[-y]_{补}$	1. 0 0 0 0 1		； 第一步减除数y
	1.1 1 0 0 1	<0 $q_0=0$	； 余数为负,商0
$+[y]_{补}\rightarrow$	0.0 1 1 1 1 1		； 除数右移1位加
	0.0 1 0 0 0 1	>0 $q_1=1$	； 余数为正,商1
$+[-y]_{补}\rightarrow$	1.1 1 0 0 0 0 1		； 除数右移2位减
	0.0 0 0 0 0 1 1	>0 $q_2=1$	； 余数为正, 商1
$+[-y]_{补}\rightarrow$	1.1 1 1 0 0 0 0 1		； 除数右移3位减
	1.1 1 1 0 0 1 1 1	<0 q3=0	； 余数为负, 商0

习题[2-8-1)] x＝11000, y＝-11111, 用原码除法，求 $x \div y$。

1.1 1 1 0 0 1 1 1 <0 q3=0 ； 余数为负, 商0

＋$[y]_补$→ 0.0 0 0 0 0 1 1 1 1 1 ； 除数右移4位加

1.1 1 1 1 0 1 1 0 1 <0 q4=0； 余数为负, 商0

＋$[y]_补$→ 0.0 0 0 0 0 0 1 1 1 1 1 ； 除数右移5位加

---

1. 1 1 1 1 1 1 1 0 0 1 <0 q5=0； 余数为负， 商0

- 商真值|$x \div y$|=0.11000，原码除法 $[x \div y]_原$=1.11000
- 余数：0.0000011
- 小数点右移5位（补偿）：0.11

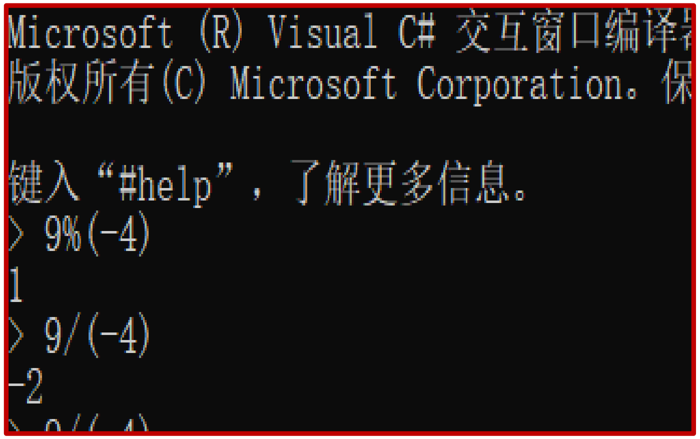
0.0 1 0 0 0 1 >0 q_1=1 ； 余数为正,商1

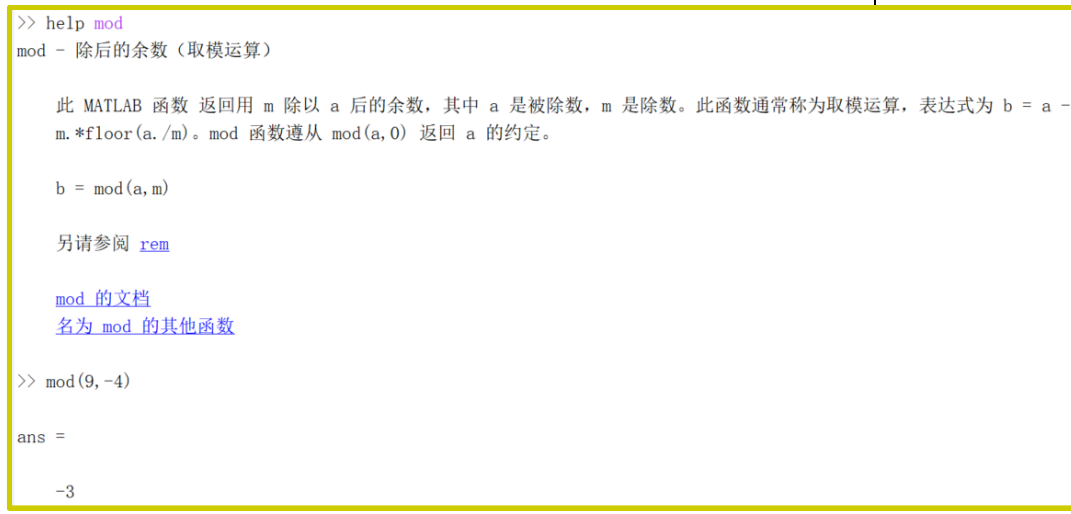
＋$[-y]_补$→ 1. 1 1 0 0 0 0 1 ； 除数右移2位减

0.0 0 0 0 0 1 1 >0 q_2=1 ； 余数为正, 商1

0.0 0 0 0 0 1 1 >0 q_2=1 ； 余数为正, 商1

习题[2-8-1)] x＝11000，y＝-11111，用原码除法，求 $x \div y$。





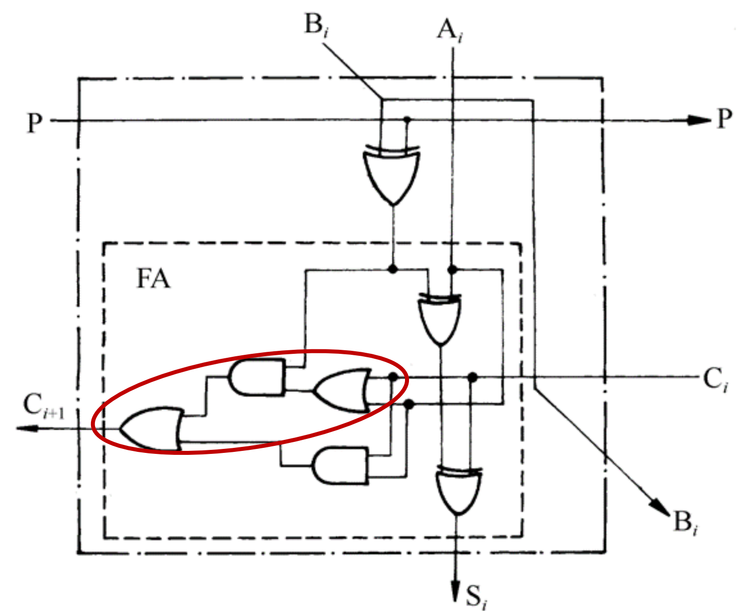
- **1）规定：余数为正数；2）余数表示方法：n=k*q+r**

- 商真值|$x \div y$|=0.11000，原码除法 $[x \div y]_{原}$=1.11000
- 余数：0.0000011
- 小数点右移5位（补偿）：0.11

	0.0 1 0 0 0 1	>0 q_1=1	；余数为正,商1
$+[-y]_{补} \rightarrow$	1. 1 1 0 0 0 0 1		；除数右移2位减
	0.0 0 0 0 0 1 1	>0 q_2=1	；余数为正，商1

扩展：可控加减法单元（CAS）

- 1位二进制可控加减法单元
  - 输入数据Ai、Bi，进位输入Ci+1
  - 控制端口：P （异或门）
    - P=0，作加法运算
    - P=1，作减法运算
  - 输出端口：和Si、进位输出Ci+1
  - 级联端口：Bi、P
  - 进位输出延迟：3T
- 逻辑函数

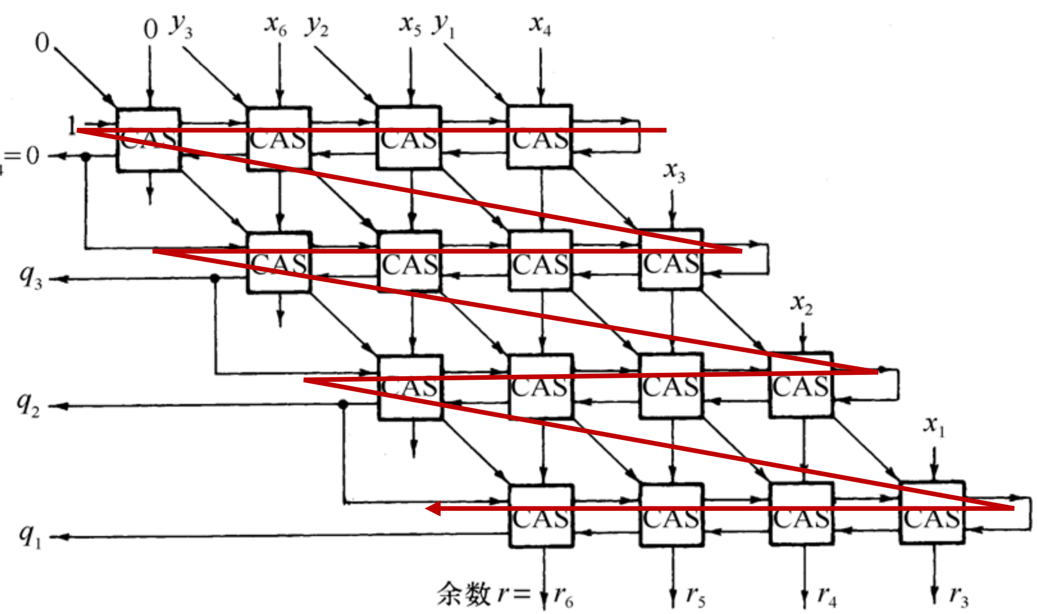


$$S_i = A_i \oplus (B_i \oplus P) \oplus C_i$$

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

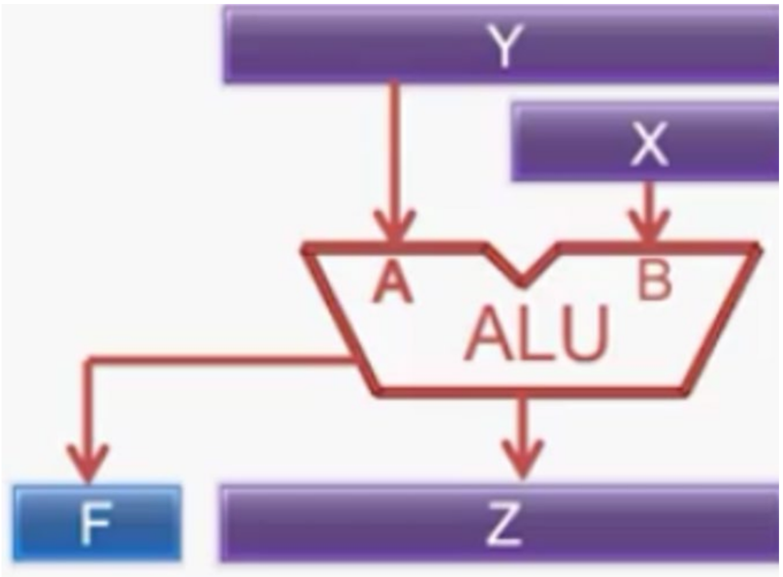
扩展：并行除法器—加减交替

- 4位除4位——不恢复余数阵列除法器
  - 被除数 $x=0.x_6x_5x_4x_3x_2x_1$ (双倍长)
    除数 $y=0.y_3y_2y_1$
    商数 $q=0.q_3q_2q_1$
    余数 $r=0.00r_6r_5r_4r_3$
    字长 $n+1=4$
- 组成
  - 对应于除法人工过程



总结

- 定点乘法器
  - 串行移位乘法器
  - 并行阵列乘法器
  - 带符号乘法（重点）
    - 原码乘法、补码乘法
- 定点除法器
  - 加减交替法（重点）
  - 可控加减法单元、并行除法器



总结

- 运算方法（定点）
  - 定点数方法
    - 纯整数、纯小数
    - 数值范围分析
  - 数的机器码表示
    - 原码、反码、补码、移码
    - 相互转换方法
      - 原反补码
        - 正数：相同
        - 负数：反码数值位相反、补码反码末位加1
      - 移码
        - 符号位与补码相反，数值位相同
    - 校验方法
      - 奇偶校验（结果中1的个数为奇数、偶数）
  - 定点加减法
    - 补码加减法正确性证明
    - 溢出概念与判断
      - 情况：仅针对有符号数（补码环域）
      - 判别方法
        - 单符号位：数值最高位进位与符号位进位、异或
        - 双符号位法：00、01（正溢）、10（负溢）、11
  - 定点乘法
    - 无符号：乘法手算
    - 有符号乘法
      - 算前求补、算后求补
      - 符号位单独计算、原码阵列乘法器
  - 定点除法
    - 加减交替法
      - 方法：余正减，余负加
      - 终止条件：除数有效位
      - 注意：纯小数、带符号循环移位
    - 恢复余数法

总结

- **运算器（定点）**
  - 逻辑运算
  - 算术运算（加减法器）
    - 半加器、全加器
    - 行波进位、超前进位（信号产生）
    - 时延分析：关键路径
    - 加减法统一：补码（异或）
    - 溢出判断：异或（单符号位）
  - ALU
    - 函数信号发生器、74181算数逻辑单元
    - 内部总线方式：单总线、双总线、三总线
  - 定点乘除法器
    - 串行乘法、阵列乘法电路
    - 对二求补电路：（负数）从右往左扫描，最低位1左侧（除符号位）全部取反
    - 加减可控单元（CAS）、并行除法器