计算机组成与系统结构

第二章 运算方法和运算器

吕昕晨

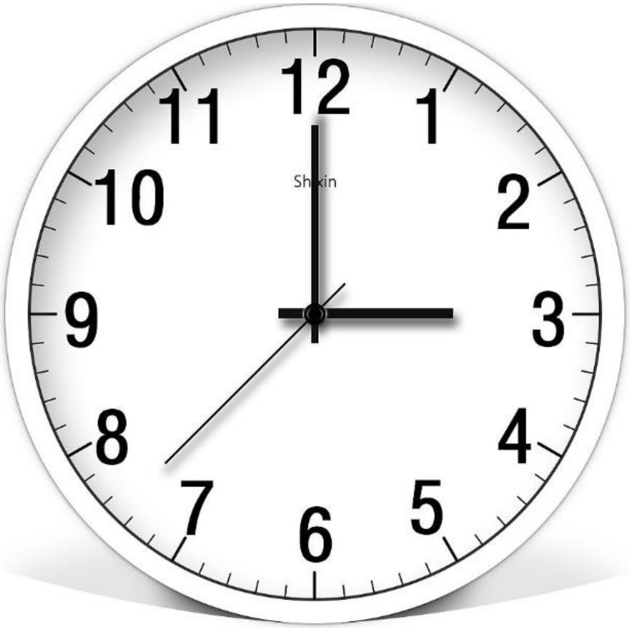
lvxinchen@bupt.edu.cn

网络空间安全学院

原码、反码、补码、移码

- 原码
  - 符号位加上真值的绝对值
- 反码
  - 正数的反码是其本身
  - 负数的反码是在其原码的基础上，符号位不变，其余各位取反
- 补码
  - 正数的补码就是其本身
  - 负数的补码是在反码的基础上+1
- 移码
  - 补码的符号位取反（无论正负）

补码定义与运算



- 时钟：现在3点钟
  - 前拨4小时→11点
  - 后拨8小时→11点
  - - 4 = + 8 (mod 12)
  - 如果a = b (mod m)，c = d (mod m)
    (1)a ± c = b ± d (mod m)
    (2)a * c = b * d (mod m)
  - n位定点数？数范围→模数
- 定点小数$x_0.x_1x_2\ldots x_n$，以2为模 [-1,1]
- 定点整数$x_0x_1x_2\ldots x_n$，以2^{n+1}为模 $[-2^n,2^n-1]$
  - 定点小数$x_0.x_1x_2\ldots x_n$

$$[x]_{补}=\begin{cases} x & 1>x\geq 0 \\ 2+x & 0\geq x>-1 \end{cases}$$

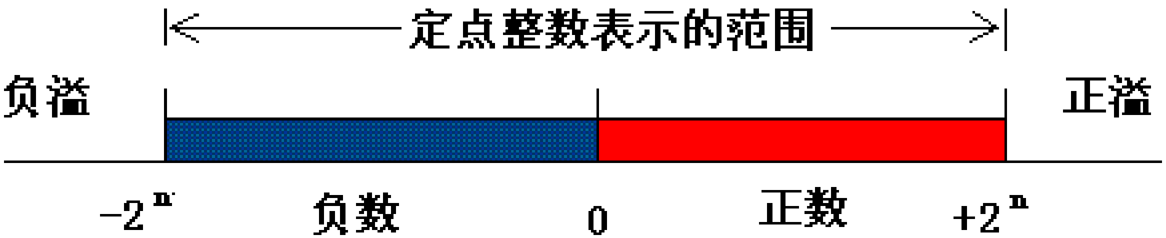
符号 $\begin{cases} 0，正小数 \\ 1，负小数 \end{cases}$

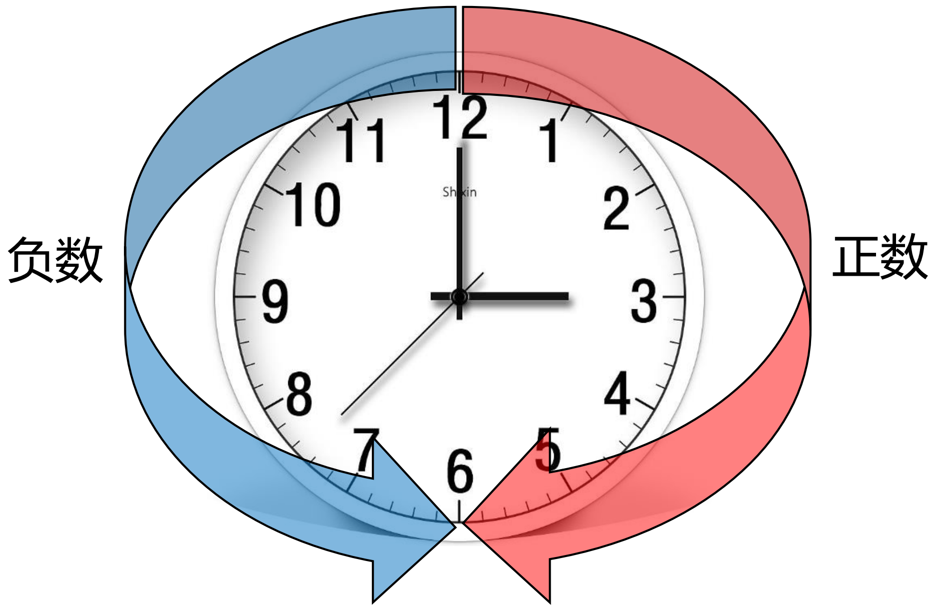
[例]将十进制真值(- 127, - 1,0, + 1, + 127)列表表示成二进制数及原码、反码、补码、移码值。

真值x（十进制）	真值x（二进制）	[x]原	[x]反	[x]补	[x]移
-127	-01111111	11111111	10000000	10000001	00000001
-1	-00000001	10000001	11111110	11111111	01111111
		00000000	00000000		
0	00000000			00000000	10000000
		10000000	11111111		
+1	+00000001	00000001	00000001	00000001	10000001
+127	+01111111	01111111	01111111	01111111	11111111

溢出概念与检测方法

溢出的概念



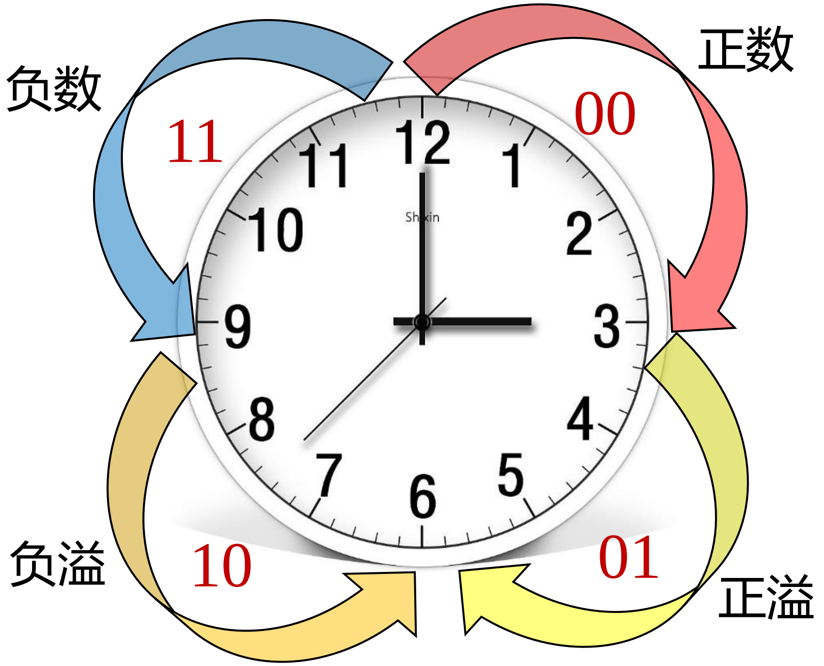


双符号位溢出检测法

1、双符号位法

（变形补码—扩大一倍范围）

$[x]_{补} = 2^{n+2} + x \quad (\bmod 2^{n+2})$



S_{f1}	S_{f2}	
0	0	正确（正数）
0	1	正溢
1	0	负溢
1	1	正确（负数）

S_{f1} 表示正确的符号，逻辑表达式为$V=S_{f1} \oplus S_{f2}$，可以用异或门来实现

溢出概念与检测方法

[例17] x=+01100 , y=+01000 , 求 x+y 。

解： $[x]_{补}$ = 001100 , $[y]_{补}$ = 001000

	$[x]_{补}$	0 0 1 1 0 0
+	$[y]_{补}$	0 0 1 0 0 0
	$[x+y]_{补}$	0 1 0 1 0 0 (表示正溢)

溢出概念与检测方法

[例18] x=-1100 , y=-1000 , 求 x+y 。

解： $[x]_{补} = 110100$, $[y]_{补} = 111000$

$$
\begin{array}{lll}
 & [x]_{补} & 1\ 1\ 0\ 1\ 0\ 0 \\
+ & [y]_{补} & 1\ 1\ 1\ 0\ 0\ 0 \\
\hline
 & [x+y]_{补} & 1\ 0\ 1\ 1\ 0\ 0
\end{array}
$$

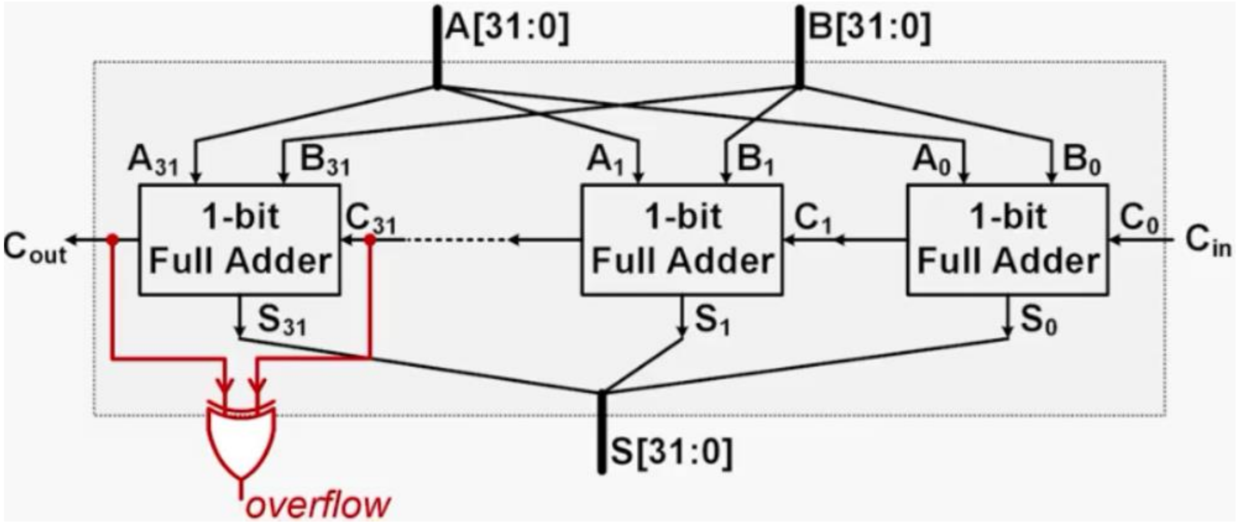
(表示负溢)

单符号位溢出检测

- Cf为符号位产生的进位，C0为最高有效位产生
- C_f　C_0

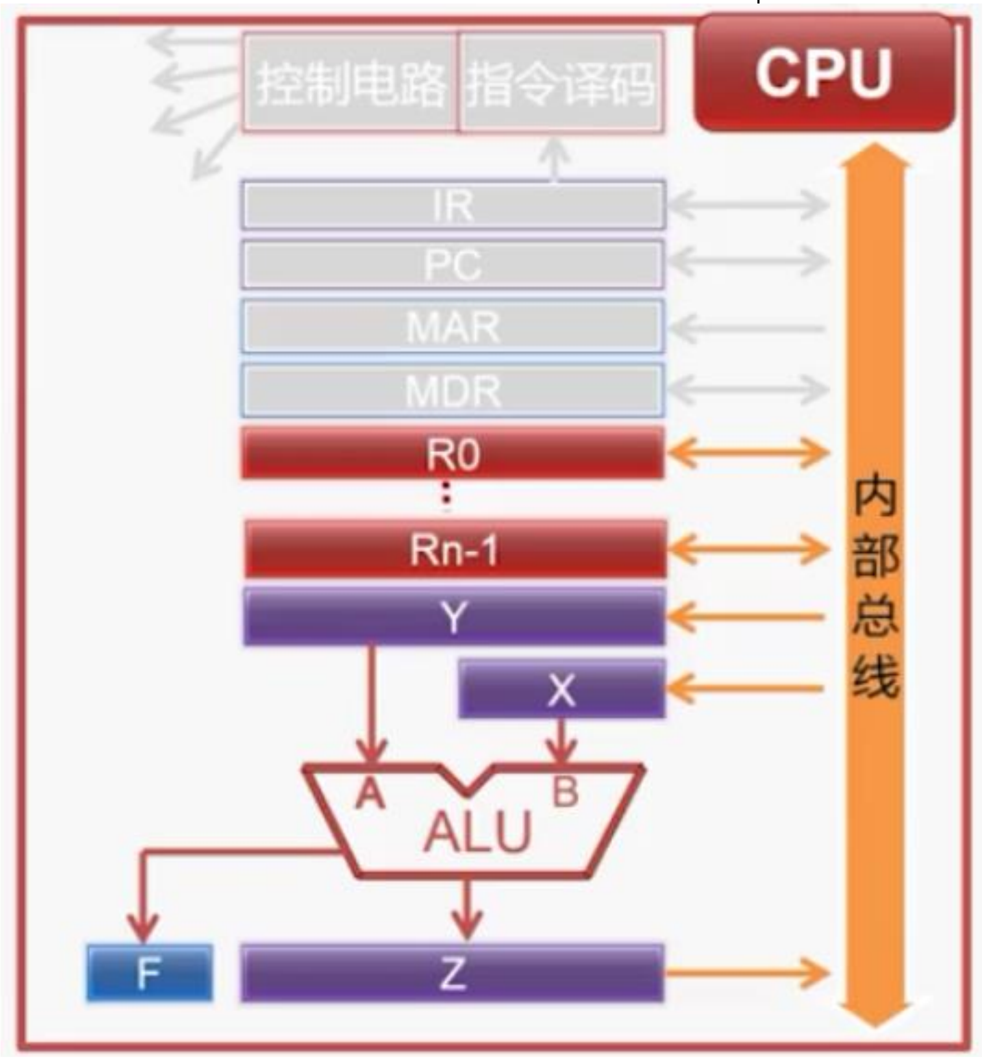
C_f	C_0	
0	0	正确（正数）
0	1	正溢
1	0	负溢
1	1	正确（负数）

$$\boxed{V = Cf \oplus C0}$$



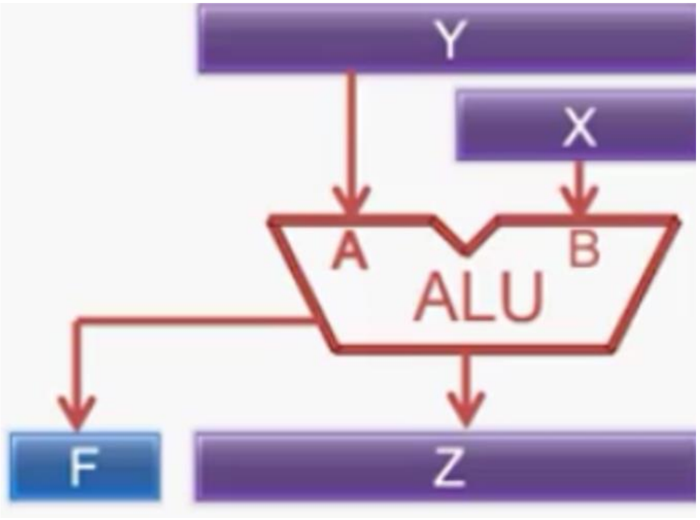
模型机——CPU运算器

- 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



本周教学安排

- 直播内容
  - 定点数
    - 乘法
    - 除法
- 录播内容
  - 运算器总线结构
  - 浮点数
    - 加、减、乘、除法



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

十进制与二进制乘法

- 例，x=13，y=14，求x × y=?
- 十进制方法

```
       1 3  (x)
×      1 4  (y)
-------------
       5 2
+    1 3
---------
     1 8 2  (z)
```

- 二进制方法

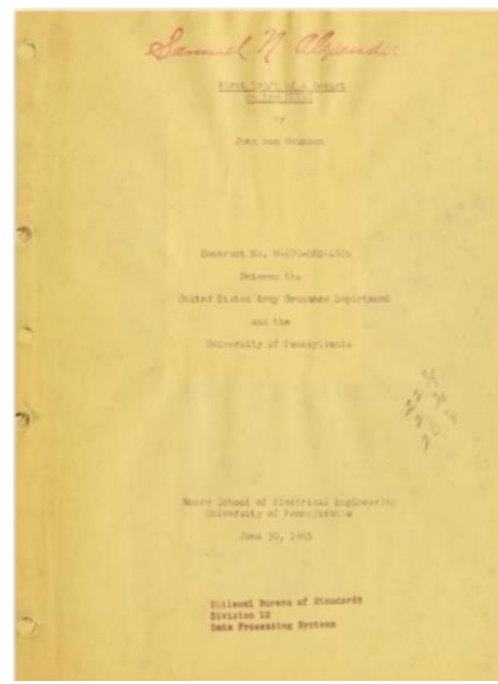
```
            1 1 0 1   (x)
×           1 1 1 0   (y)
--------------------------
            0 0 0 0
          1 1 0 1
        1 1 0 1
+     1 1 0 1
--------------
    1 0 1 1 0 1 1 0
```

- 10110110=21+22+24+25+27=182
- 思考：有何异同？有何规律？

EDVAC报告草案

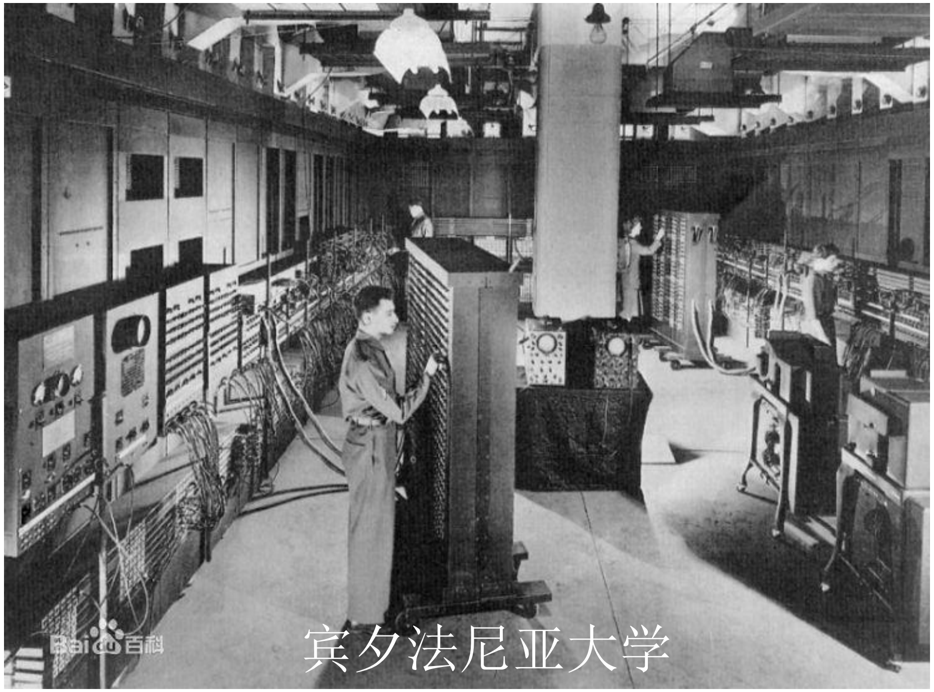
- 电子管是一种“全或无”设备（all-or-none）
  - 适合表示只有两个数值的系统，即二进制
- 二进制可以大幅度地简化乘法和除法的运算过程
  - 尤其是对于乘法，不再需要十进制乘法表
- 十进制才是适合人使用的
  - 输入输出设备需要承担二进制与十进制转

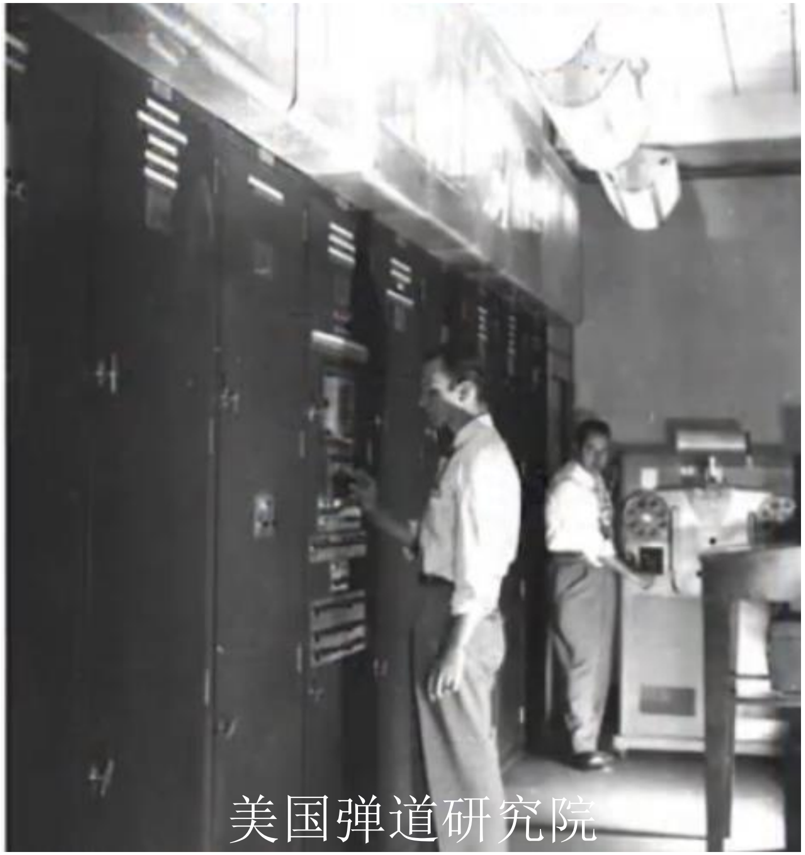
约翰·冯·诺依曼
John Von Neumann
1903~1957



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





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)
```

设置

请复述二进制乘法手算规则：

移位相加

$$
\begin{array}{cccccccccl}
 & & & & & 1 & 1 & 0 & 1 & \text{(被乘数)} \\
\times & & & & & 1 & 1 & 1 & 0 & \text{(乘数)} \\
\hline
 & & & & & 0 & 0 & 0 & 0 & \\
 & & & & 1 & 1 & 0 & 1 & & \\
 & & & 1 & 1 & 0 & 1 & & & \\
+ & & 1 & 1 & 0 & 1 & & & & \\
\hline
 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \text{(乘积)}
\end{array}
$$

正常使用主观题需2.0以上版本雨课堂

作答

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

					1	1	0	1	(被乘数)
×					1	1	1	0	(乘数)
					0	0	0	0	
移位过程				1	1	0	1	0	
			1	1	0	1	0	0	
+		1	1	0	1	0	0	0	
					0	0	0	0	
				1	1	0	1	0	
		1	0	0	1	1	1	0	
	1	0	1	1	0	1	1	0	(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;
}
```

第1步

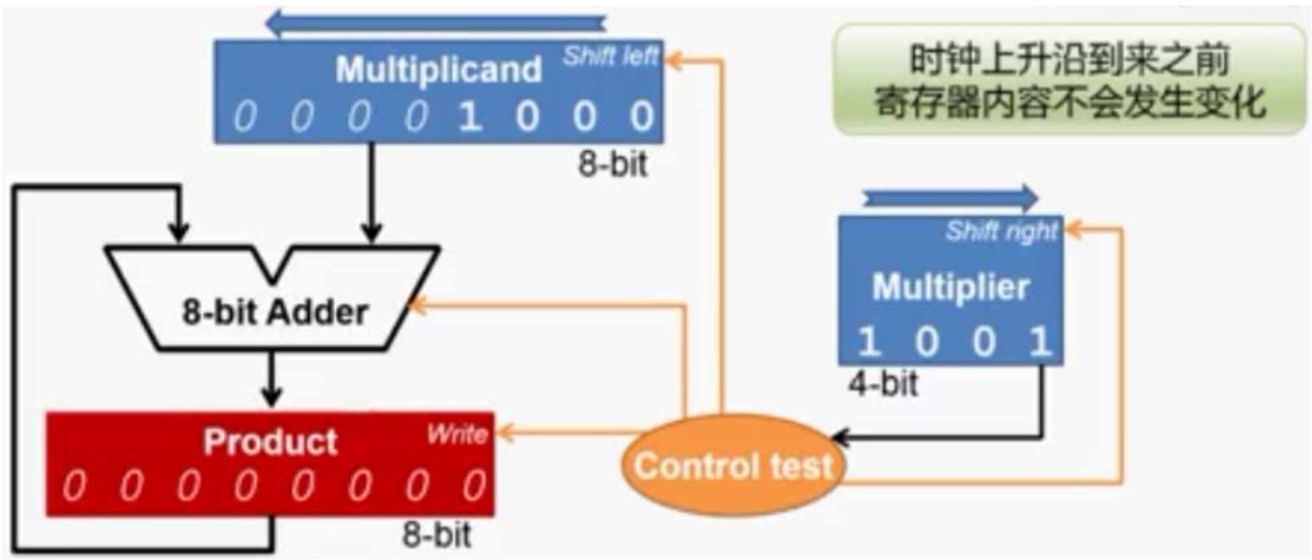
第2步

第3步

- 对于n位乘法器（乘数）
  - 需要3n个时钟周期

串行移位乘法器及优化（1）

- 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

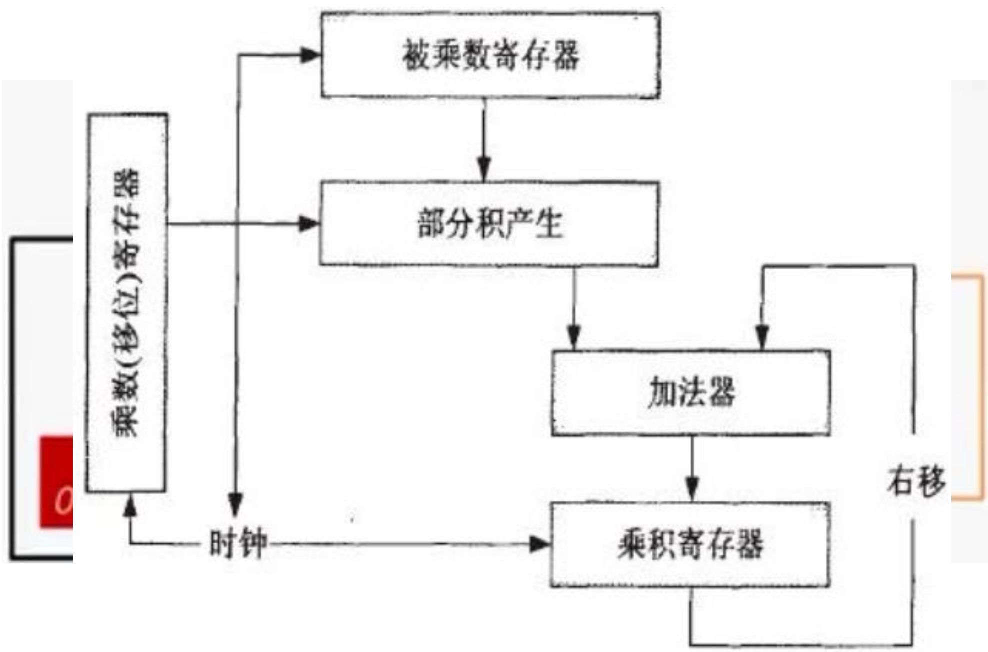


串行移位乘法器及优化（2）

- 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）

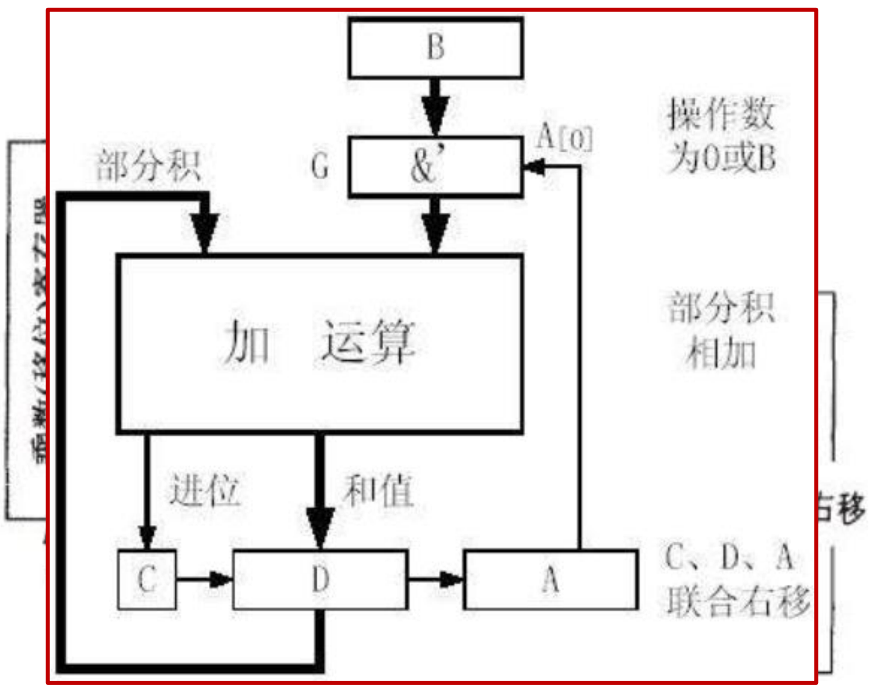


串行移位乘法器及优化（3）

- 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

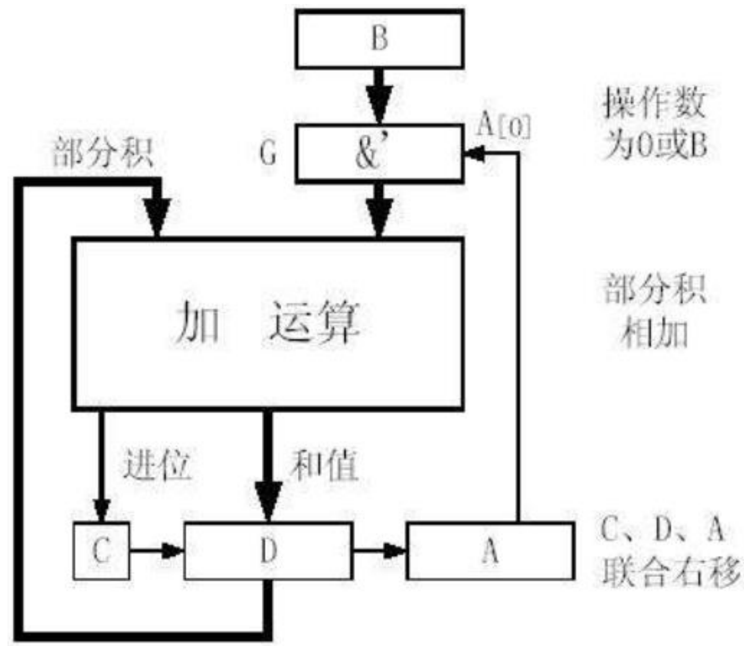
面积优化2

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
- 缺点
  - 效率较低，N个时钟周期



单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

经过优化后的N位乘法器中寄存器包括

A N位被乘数寄存器、2N位乘积寄存器

B 2N位被乘数寄存器、N位乘数寄存器、2N位乘积寄存器

C 2N位被乘数寄存器、2N位乘积寄存器

D N位被乘数寄存器、N位乘数寄存器、2N位乘积寄存器

提交

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

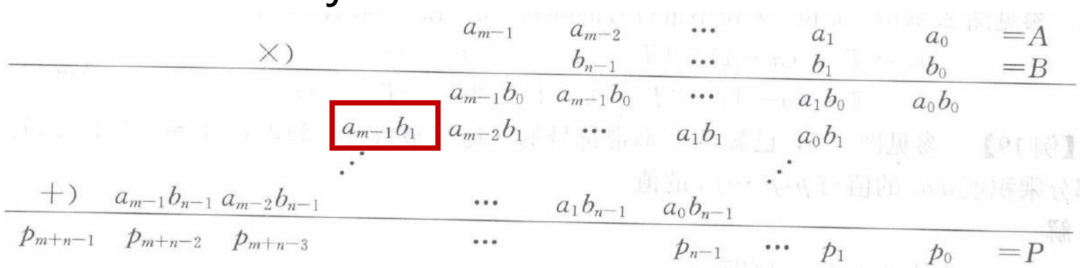
并行阵列乘法器

- 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

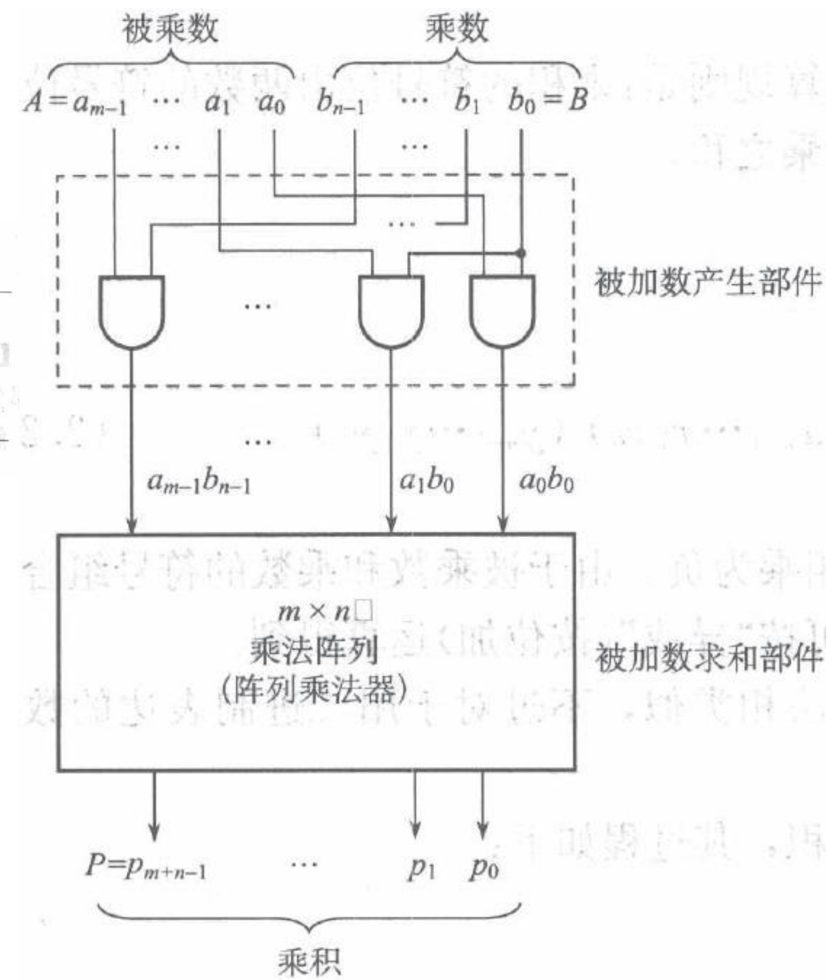
```
              1 1 0 1     (被乘数)
×             1 1 1 0     (乘数)
--------------------------
              0 0 0 0
            1 1 0 1
          1 1 0 1
+       1 1 0 1
--------------------------
      1 0 1 1 0 1 1 0     (乘积)
```

并行阵列乘法器结构

- 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



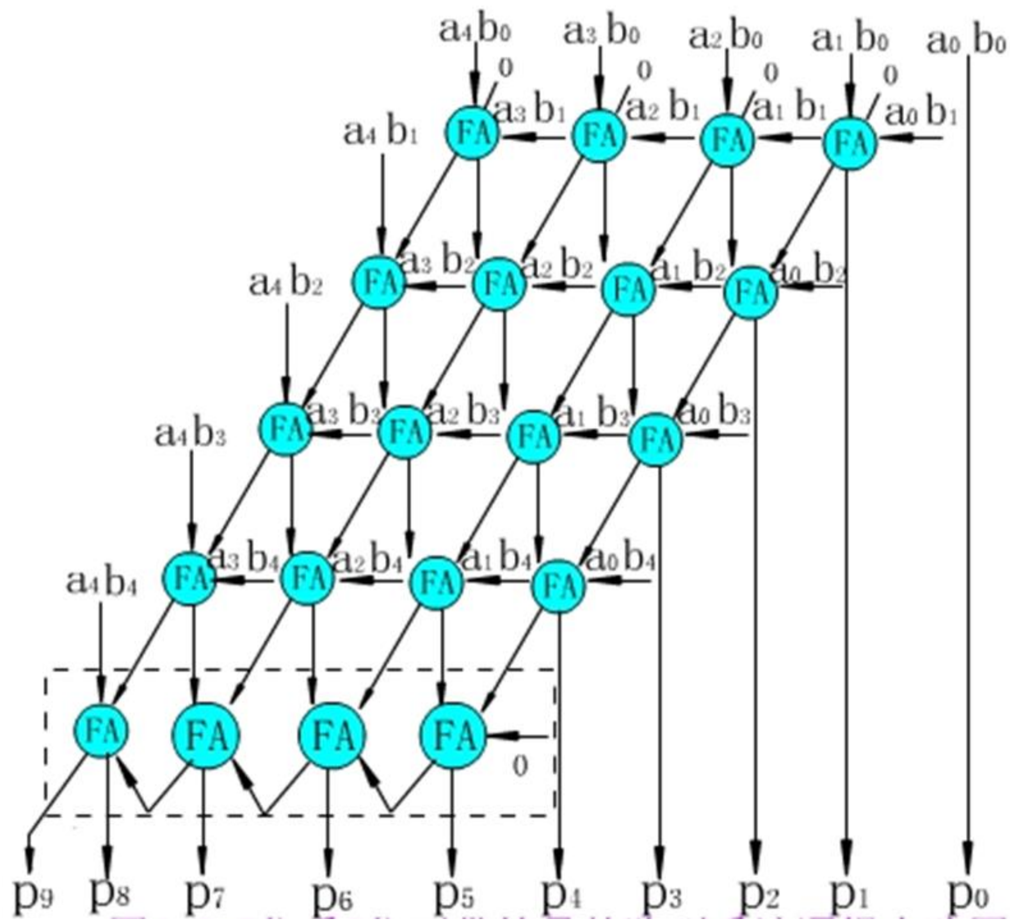
  - 电路组成：m*n个与门
- 乘法阵列
  - 阵列全加器组合，实现乘积求和功能



乘法阵列实现

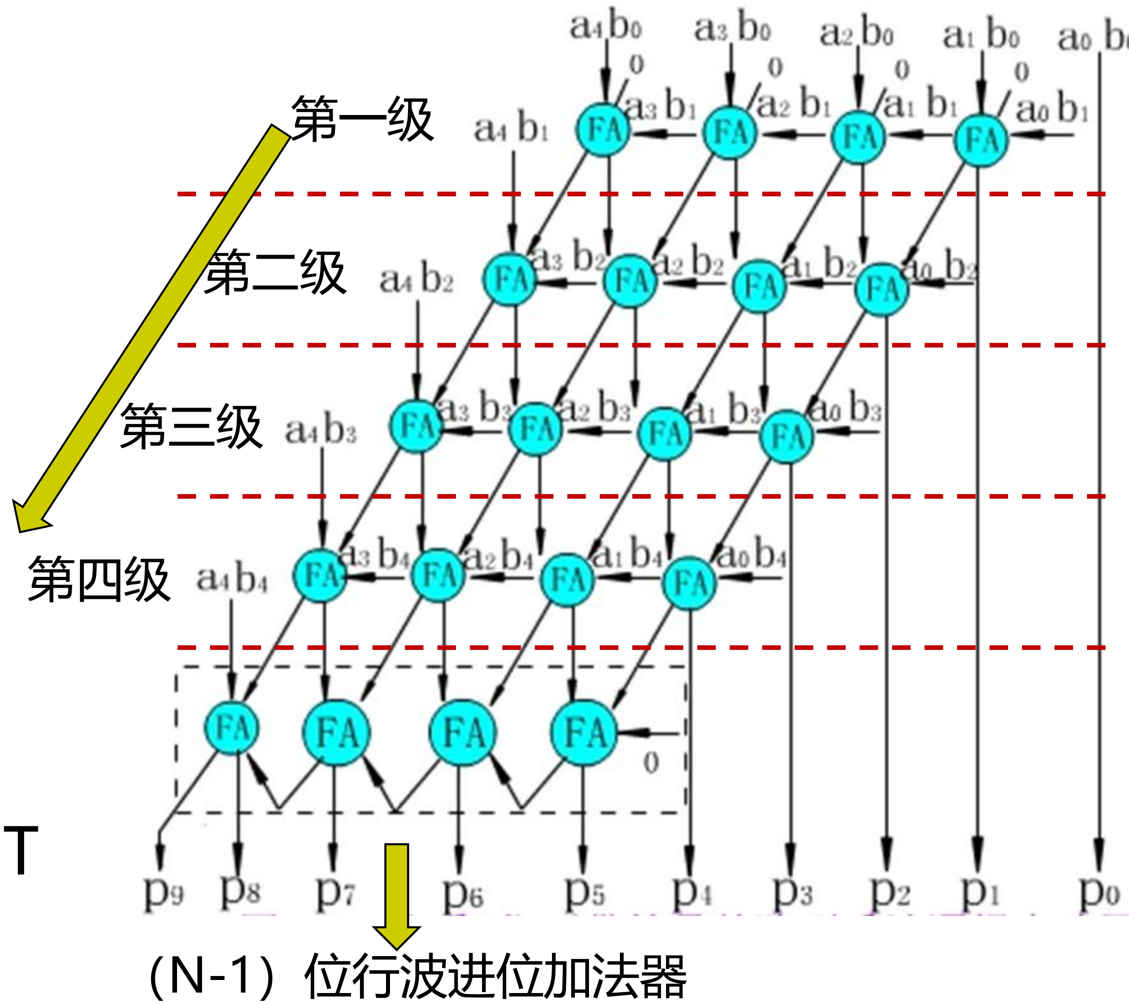
- 结构特点
  - 排列方式与手写相同
- 全加器输出
  - 斜线：进位输出
  - 竖线：和输出
  - N(N-1)个全加器

```
            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

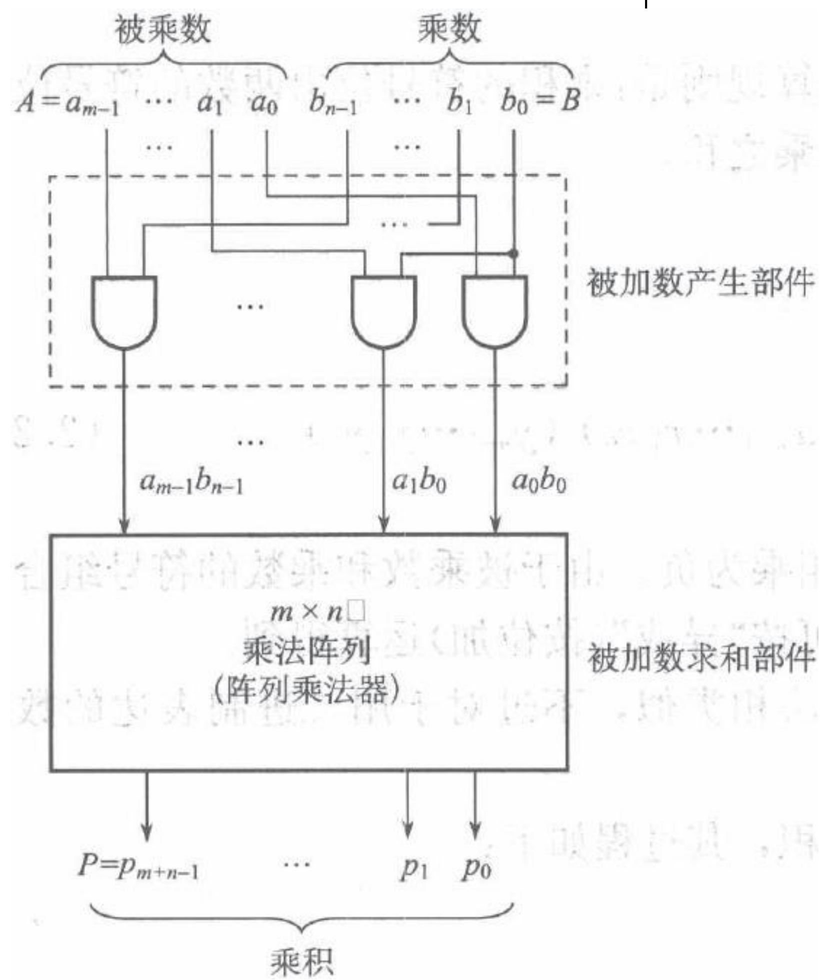


并行阵列乘法器延迟分析（2）

- 倍加数生成
  - 延迟：T（与门)

$$T + (n - 1)6T + (N - 1)2T + 3T = (8n - 4)\mathrm{T}$$

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



单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

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

提交

手写乘法过程

[例19] 已知不带符号的二进制整数A=11011，B=10101，求每一部分乘积项a_ib_j的值与$p_9p_8\ldots p_0$的值。

解：

	1 1 0 1 1	＝ A (27_{10})
×	1 0 1 0 1	＝ B (21_{10})
	1 1 0 1 1	$a_4b_0=1, a_3b_0=1, a_2b_0=0, a_1b_0=1, a_0b_0=1$
	0 0 0 0 0	$a_4b_1=0, a_3b_1=0, a_2b_1=0, a_1b_1=0, a_0b_1=0$
	1 1 0 1 1	$a_4b_2=1, a_3b_2=1, a_2b_2=0, a_1b_2=1, a_0b_2=1$
	0 0 0 0 0	$a_4b_3=0, a_3b_3=0, a_2b_3=0, a_1b_3=0, a_0b_3=0$
＋	1 1 0 1 1	$a_4b_4=1, a_3b_4=1, a_2b_4=0, a_1b_4=1, a_0b_4=1$
	1 0 0 0 1 1 0 1 1 1	＝ P

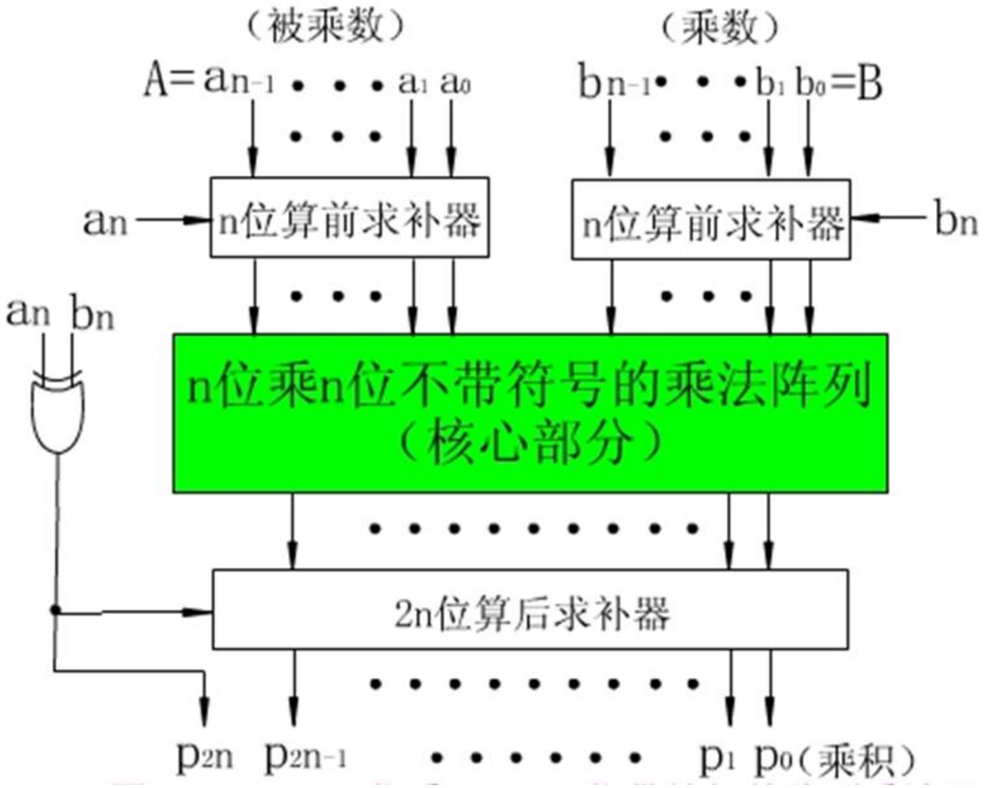
$P = p_9p_8p_7p_6p_5p_4p_3p_2p_1p_0 = 1000110111\ (567_{10})$

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

有符号数存储方式——补码

- 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



求补电路（对2求补）

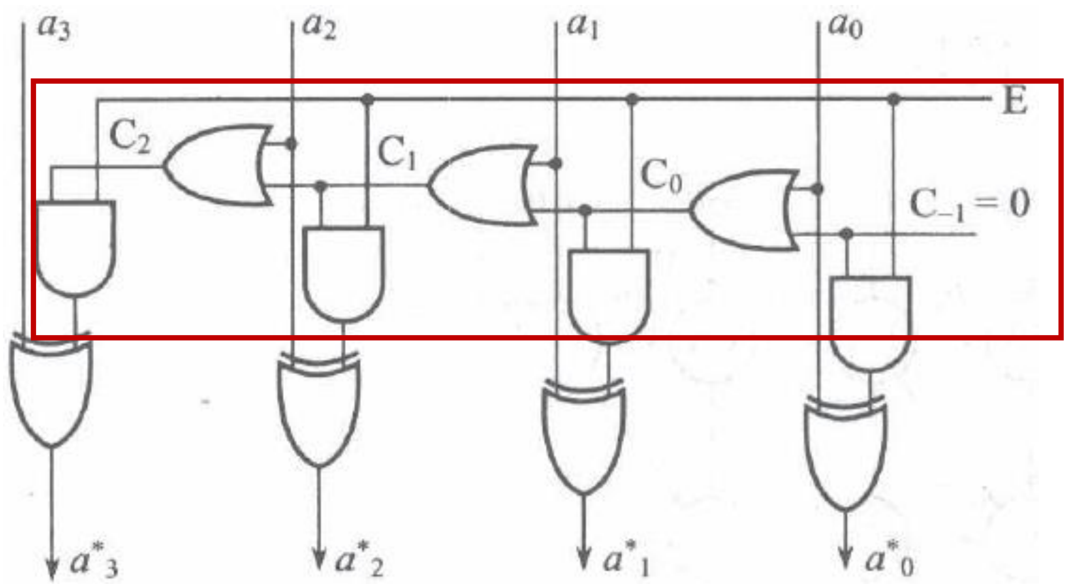
- 例 x=-1011110， 写出其原码与补码
  - 原码为1 10111 10
  - 补码为1 01000 10
- 补码转换性质
  - 按位取反，末位加一（加法器）
  - 最右端往左边扫描，直到第一个1的时候，该位和右边各位保持不变，左边各数值位按位取反（扫描）

- $[x]_{原}$= 1 11110　　　　　补：1 00010

不变，左边数值位取反

求补电路

- 功能组成
  - 按位扫描
    - 或门级联
  - 逐位取反
    (异或门)



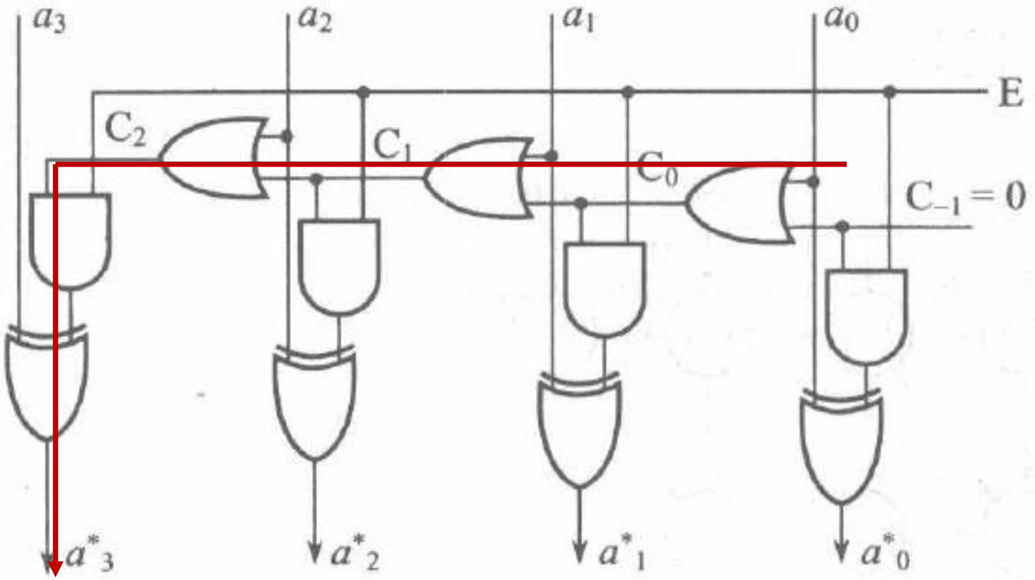
- 逻辑表达式

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

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

求补电路延迟

- 关键路径——（n+1）位求补电路
  - 按位扫描
    - nT
  - 使能求反
    - T (与)+3T (异或)
- 总延迟
  - nT+4T=(n+4)T



- 教材（2.24）式说明
  - 与/或门：2T
  - 异或门：3T

$$\boxed{t_{TC} = n \cdot 2T + 5T = (2n+5)T}$$

[例20] 设x=+15，y=-13，用带求补器的原码阵列乘法器求出乘积 x·y=?

解：$[x]_{原}=01111$, $[y]_{原}=11101$, $|x|=1111$, $|y|=1101$

符号位运算：$0 \oplus 1=1$

$$
\begin{array}{r}
1\,1\,1\,1 \\
\times \quad 1\,1\,0\,1 \\
\hline
1\,1\,1\,1 \\
0\,0\,0\,0 \\
1\,1\,1\,1 \\
+ \quad 1\,1\,1\,1 \\
\hline
1\,1\,0\,0\,0\,0\,1\,1
\end{array}
$$

乘积符号为1，算后求补器输出11000011，$[x \times y]_{原}=111000011$

换算成二进制数真值是 $x \cdot y = (-11000011)_2 = (-195)_{10}$

[例21] 设x=-15，y=-13，用带求补器的补码阵列乘法器求出乘积x·y=? 并用十进制数乘法进行验证。

解：$[x]_{补}=10001$, $[y]_{补}=10011$, 乘积符号位运算：$1\oplus1=0$

尾数部分算前求补器输出 |x|=1111 , |y|=1101

$$
\begin{array}{r}
1\,1\,1\,1 \\
\times \qquad 1\,1\,0\,1 \\
\hline
1\,1\,1\,1 \\
0\,0\,0\,0 \\
1\,1\,1\,1 \\
+ \quad 1\,1\,1\,1 \\
\hline
1\,1\;\;0\,0\;\;0\;\;0\,1\,1
\end{array}
$$

乘积符号为0，算后求补器输出11000011，$[x\times y]_{补}=011000011$

补码二进制数真值 $x\cdot y=0\times2^8+1\times2^7+1\times2^6+1\times2^1+1\times2^0$

$=(+195)_{10}$

十进制数乘法验证 $x\cdot y = (-15)\times(-13) = +195$

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

二进制除法

- 设有n位定点小数（定点整数也适用）
  - 被除数x，$[x]_{原}=x_f.x_{n-1}\ldots x_1x_0$
  - 除数y，　$[y]_{原}=y_f.y_{n-1}\ldots y_1y_0$

- 商 $q = x / y$
  - $[q]_{原}=(x_f\oplus y_f) + (0.x_{n-1}\ldots x_1x_0/0.y_{n-1}\ldots y_1y_0)$
  - 商的符号运算$q_f=x_f\oplus y_f$与原码乘法一样，用模2求和得到。

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
  - 比较被除数与余数大小
    - 若够减，对应位商1
    - 若不够减，对应位商0
  - 除数右移
  - 直到余数<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）
  - 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。
  - 控制简单，有规律。

补码除法流程——加减交替法

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

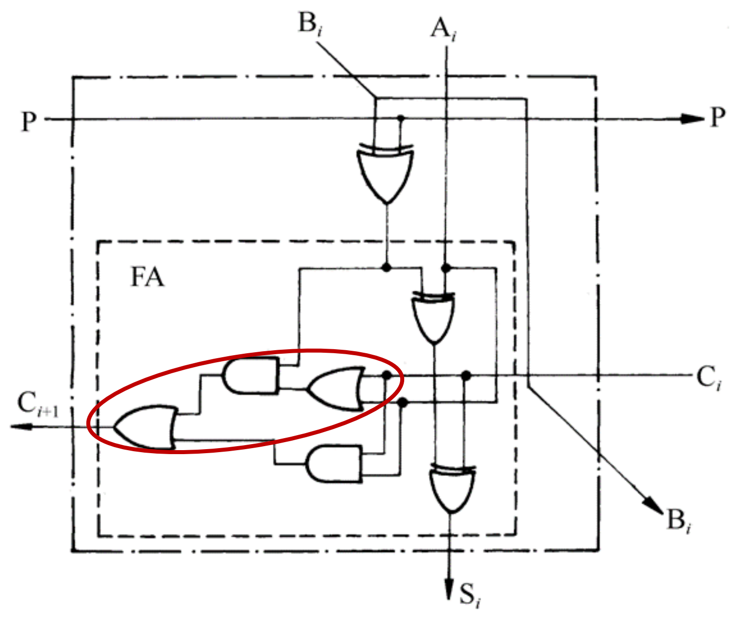
[解:] $[x]_{补}=0.101001$, $[y]_{补}=0.111$, $[-y]_{补}=1.001$

	0.1 0 1 0 0 1		；	被除数
$+[-y]_{补}$	1.0 0 1		；	第一步减除数y
	1.1 1 0 0 0 1	<0 $q_4=0$	；	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 $q_3=1$	；	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
	0.0 0 0 1 1 0	>0 $q_1=1$	；	余数为正,商1

商$q=q_4.q_3q_2q_1=0.101$, 余数$r=0.000110$

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
    - 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



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

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

并行除法器——加减交替

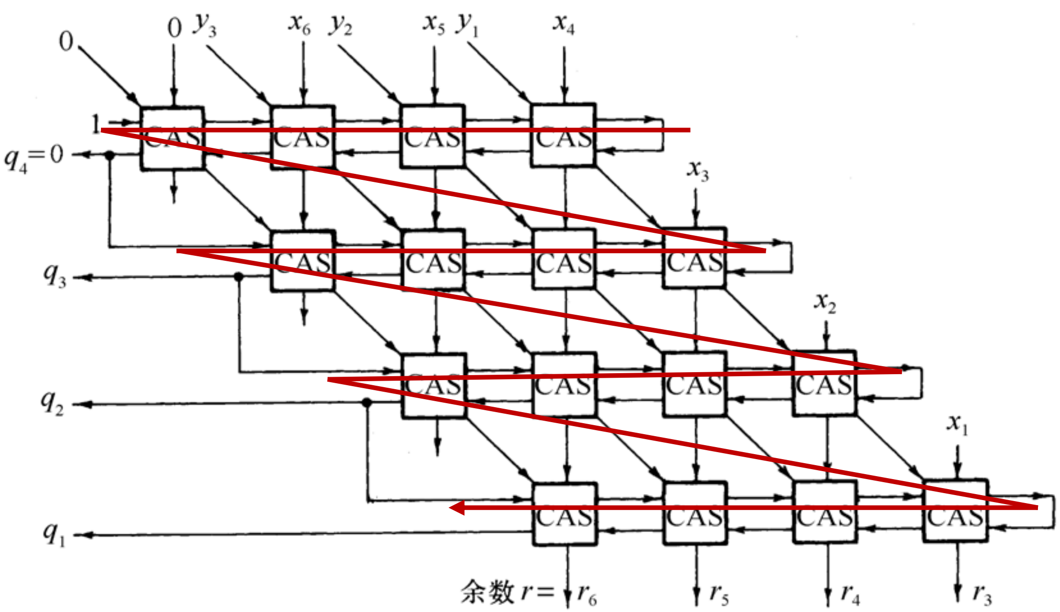
- 4位除4位——不恢复余数阵列除法器
  - 被除数 $x=0.x_6x_5x_4x_3x_2x_1$ (双倍长)

    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
  - CAS数目：$(n+1)^2$
  - 延迟：$3T\ (n+1)^2$



总结

- 定点乘法器
  - 串行移位乘法器
    - 实现与优化
  - 并行阵列乘法器
    - 延迟分析
  - 带符号乘法设计
    - 原码乘法、补码乘法
- 定点除法器
  - 加减交替法
  - 可控加减法单元