计算机组成原理与系统结构

第四章

运算方法与运算器

http://jpkc.hdu.edu.cn/computer/zcyl/dzkjdx/

杭州電子科技大學
HANGZHOU DIANZI UNIVERSITY

第 4 章 运算方法与运算器

BACK

4.3 定点数除法运算及实现

一 原码除法及实现

二 补码除法及实现

三 阵列除法器

串行乘法算法

一、原码除法及实现

1 手工除法算法

2 原码恢复余数算法

3 原码不恢复余数算法

4 原码除法的硬件实现

1、手工除法算法

X=+0.1011， Y=-0.1101

X÷Y

❖ 手工除法：

- 判断被除数是否大于除数，决定商 0/1 。
- 若商 1，则减除数，再添加一位数或者 0
- 若商 0，则直接添加一位数或者 0

```
           0.1101
        ┌──────────
0.1101  │ 0.10110
             1101
          ─────────
            10010
             1101
          ─────────
            10100
             1101
          ─────────
             0111
```

改进手工算法→适合机器运算：

- 计算机通过做**减法测试**来实现判断：结果大于等于0，表明够减，商1；结果小于0，表明不够减，商0。
- 计算机**将余数左移一位，再直接与不右移的除数相减。**
- 即：减（+除数相反数的补码）

  左移，构成循环

2、原码恢复余数算法

- 假设 $[X]_{原}=X_S.X_1X_2\cdots\cdots X_n$，$[Y]_{原}=Y_S.Y_1Y_2\cdots\cdots Y_n$，Q 是 X÷Y 的商，$Q_S$ 是商的符号，R 是 X÷Y 的余数，R_S 是余数的符号

- 原码除法运算的规则是：

（1）$Q_S = X_S \oplus Y_S$，$R_S = X_S$，$|Q| = |X| \div |Y| - |R| \div |Y|$

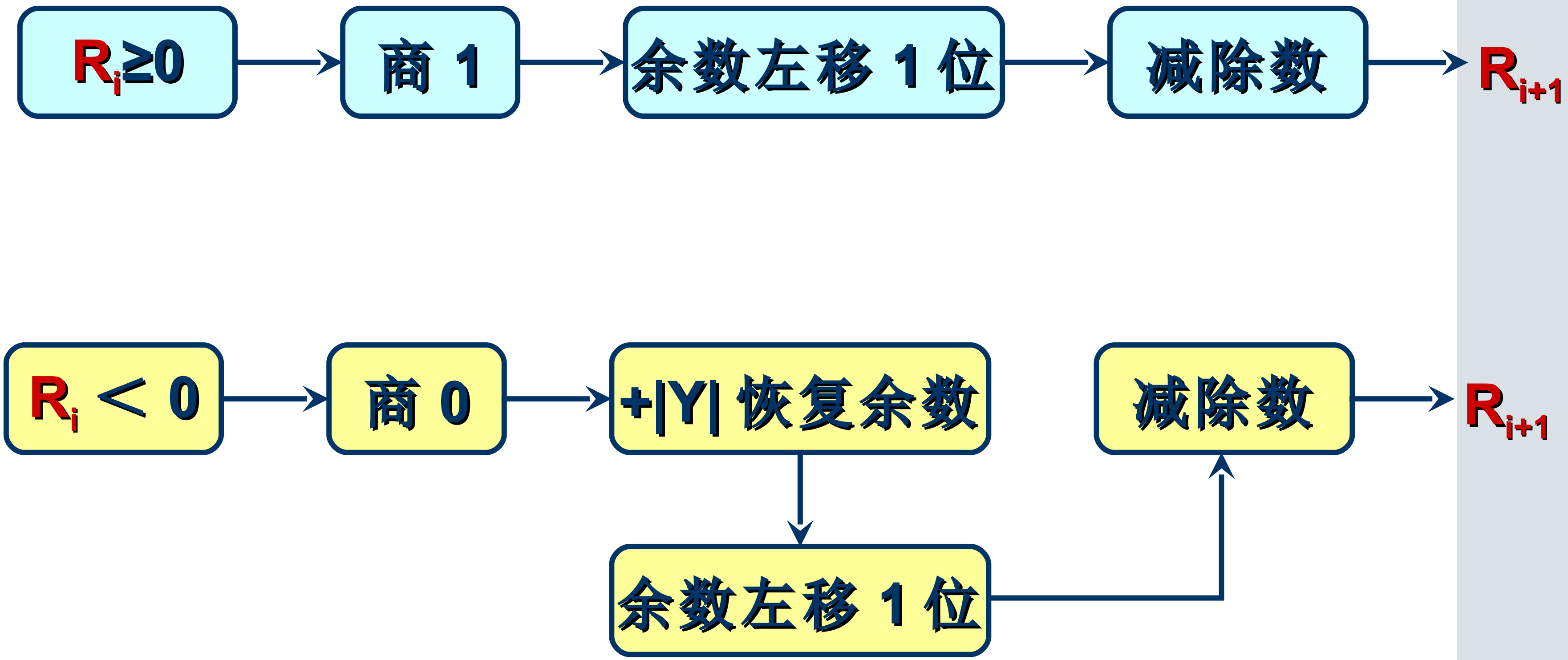
（2）余数和被除数、除数均采用**双附加符号位**；**初始余数为 $|X|$**。

2、原码恢复余数算法

（3）每次用余数减去 |Y| （通过加上 $[-|Y|]_{补}$ 来实现），若结果的符号位为 0，则够减，上商 1，余数左移一位；若结果的符号位为 1，则不够减，上商 0，先加 |Y| 恢复余数，然后余数左移一位。

（4）循环操作步骤 3，共做 n+1 次，最后一次不左移，但若最后一次上商 0，则必须 +|Y| 恢复余数；若为定点小数除法，余数则为最后计算得到的余数右移 n 位的值。

2、原码恢复余数算法



举例 1

- X=+0. 1011， Y=−0. 1101
- 用原码恢复余数算法计算 X÷Y。
- $[X]_{原}$ =0. 1011
- $[Y]_{原}$ =1. 1101
- |X|=0. 1011 |Y| =0. 1101
- $[-|Y|]_{补}$ =11. 0011
- $Q_S = X_S \oplus Y_S = 1$ $R_S = 0$

得 $[Q]_{原}$ =1. 1101

$[R]_{原}$ =0. 0000011

被除数/余数	商Q	操作说明
00.1011	0 0 0 0 0	
+ 11.0011		$+[-\|Y\|]_{补}$
11.1110	0 0 0 0 0	$R_0<0$，上商0
+ 00.1101		+|Y|恢复余数
00.1011		
01.0110	0 0 0 0 0	左移一位
+ 11.0011		$+[-\|Y\|]_{补}$
00.1001	0 0 0 0 1	$R_1>0$，上商1
01.0010	0 0 0 1 0	左移一位
+ 11.0011		$+[-\|Y\|]_{补}$
00.0101	0 0 0 1 1	$R_2>0$，上商1
00.1010	0 0 1 1 0	左移一位
+ 11.0011		$+[-\|Y\|]_{补}$
11.1101	0 0 1 1 0	$R_3<0$，上商0
+ 00.1101		+|Y|恢复余数
00.1010		
01.0100	0 1 1 0 0	左移一位
+ 11.0011		$+[-\|Y\|]_{补}$
00.0111	0 1 1 0 1	$R_4>0$，上商1

3、原码不恢复余数算法

- 又称为加减交替法：当不够减时，不恢复余数，用加上除数（+|Y|）的办法来继续求下一位商，其他操作不变。
- 加减交替法的规则如下：
  - 余数为正时，商上1，求下一位商的办法，是余数左移一位，再减去除数；
  - 当余数为负时，商上0，求下一位商的办法，是余数左移一位，再加上除数。
  - 若最后一次上商为0，而又需得到正确余数，则在这最后一次仍需恢复余数。

3 、原码不恢复余数算法

$R_i \geq 0$ → 商 1 → 余数左移 1 位 → 减除数 → R_{i+1}

$R_{i+1}=2R_i - |Y|$

$R_i < 0$ → 商 0 → +|Y| 恢复余数 → 余数左移 1 位 → 减除数 → R_{i+1}

$R_{i+1}=2(R_i+|Y|) - |Y|=2R_i+2|Y| - |Y|=2R_i+|Y|$

$R_i < 0$ → 商 0 → 余数左移 1 位 → 加除数 → R_{i+1}

举例 2

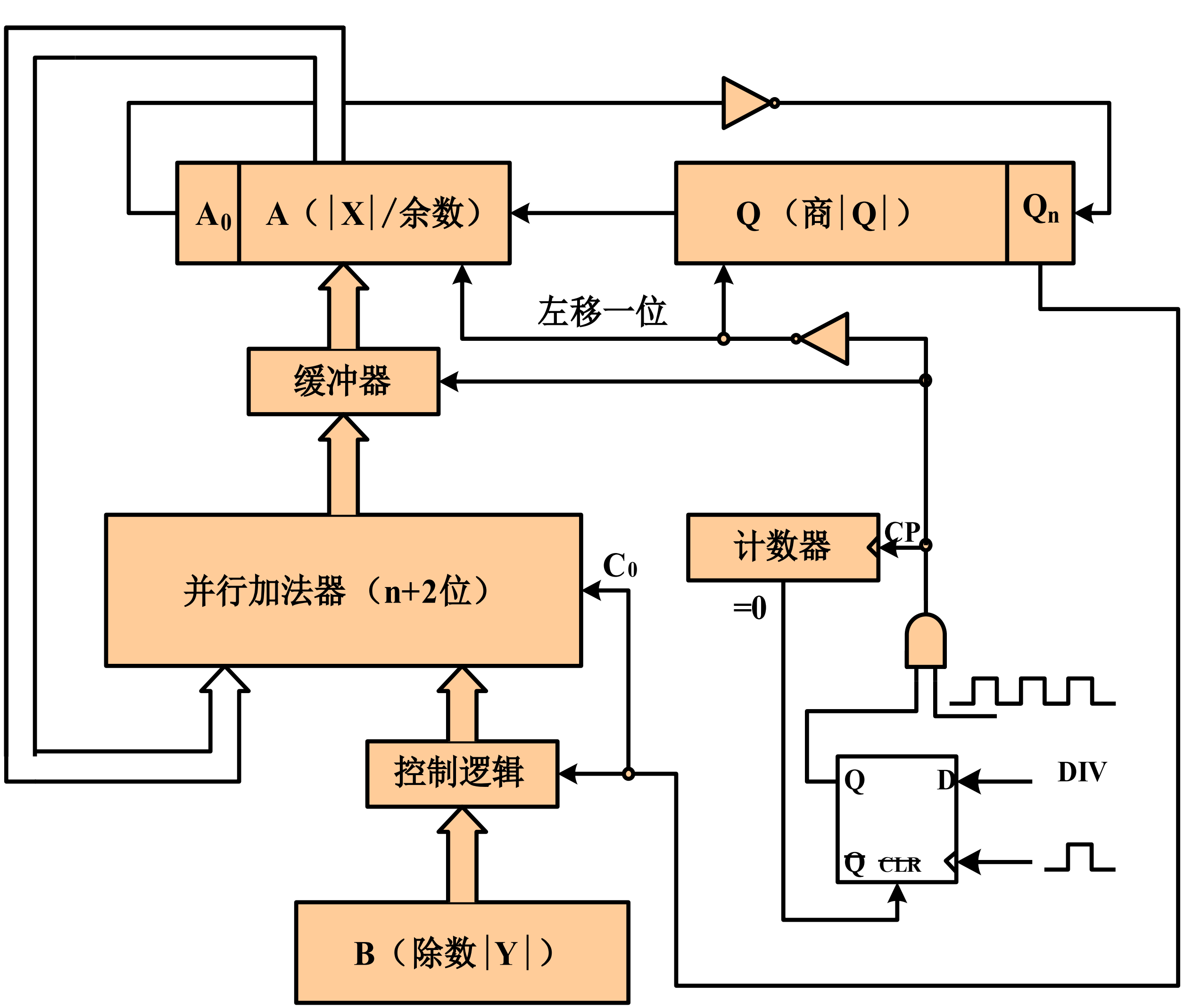
- X=+0.1011， Y=-0.1101，用原码不恢复余数算法计算 X÷Y。
- $[X]_{原}$=0.1011
- $[Y]_{原}$=1.1101
- |X|=0.1011 |Y|=0.1101
- $[-|Y|]_{补}$=11.0011
- $Q_S = X_S \oplus Y_S = 1$ $R_S = 0$

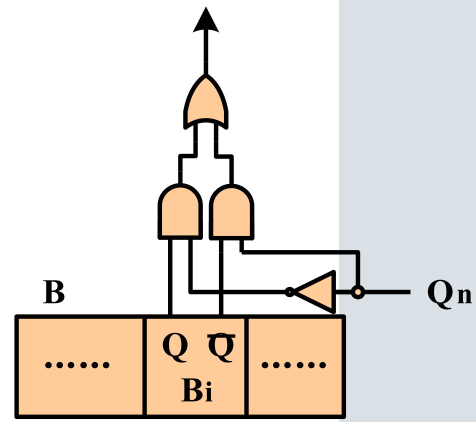
$[Q]_{原}$ =1.1101

$[R]_{原}$ =0.00000111

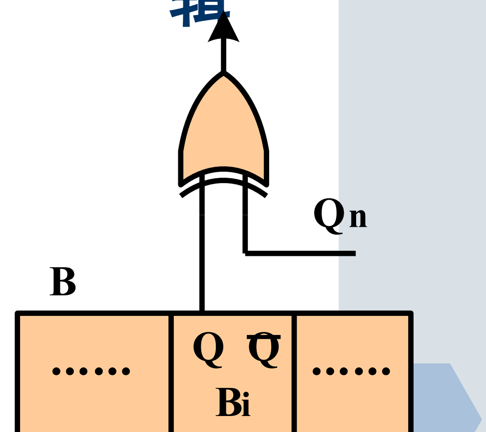
被除数/余数	商Q	操作说明
00.1011	0 0 0 0 0	
+ 11.0011		$+[-\|Y\|]_{补}$
11.1110	0 0 0 0 0	$R_0<0$，上商0
11.1100	0 0 0 0 0	左移一位
+ 00.1101		$+\|Y\|$
00.1001	0 0 0 0 1	$R_1>0$，上商1
01.0010	0 0 0 1 0	左移一位
+ 11.0011		$+[-\|Y\|]_{补}$
00.0101	0 0 0 1 1	$R_2>0$，上商1
00.1010	0 0 1 1 0	左移一位
+ 11.0011		$+[-\|Y\|]_{补}$
11.1101	0 0 1 1 0	$R_3<0$，上商0
11.1010	0 1 1 0 0	左移一位
+ 00.1101		$+\|Y\|$
00.0111	0 1 1 0 1	$R_4>0$，上商1

4、原码除法的硬件实现





控制电路逻辑



4、原码除法的硬件实现

- A：累加寄存器
- B：除数寄存器
- Q：商寄存器

- A、Q：左移寄存器

初始：
- A=|X|
- B=|Y|
- Q=0
- 计数器 =n+1

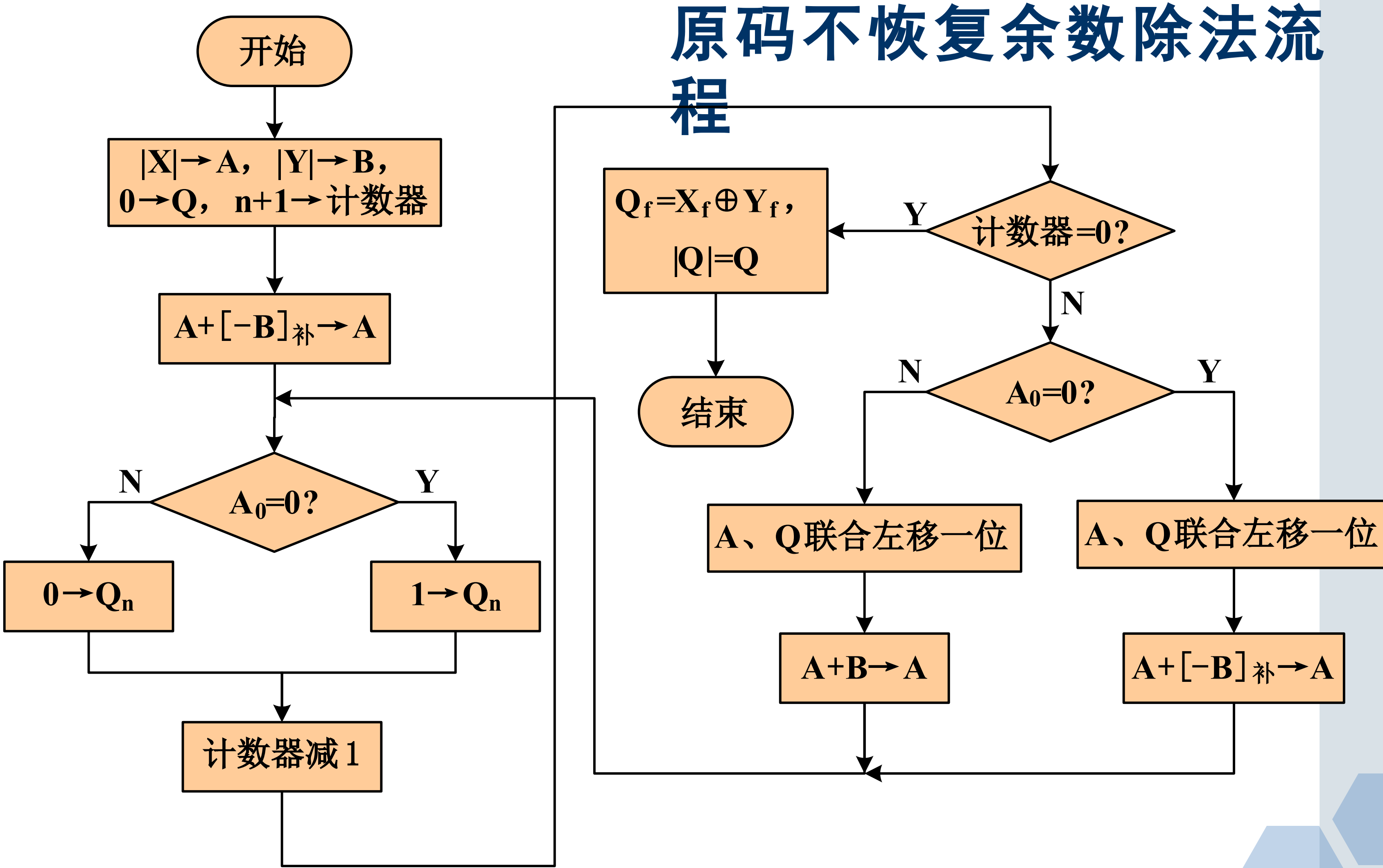
加减、商 n+1 次

左移 n 次

结果：
- A= 余数低位
- B =|Y|
- Q= 商
- 计数器 =0

4、原码除法的硬件实现

原码不恢复余数除法流程



二、补码除法及实现

❖ 1、补码除法算法

补码不恢复余数除法的规则。假设 $[X]_{补}=X_S.X_1X_2\cdots\cdots X_n$，$[Y]_{补}=Y_S.Y_1Y_2\cdots\cdots Y_n$，Q 是 X÷Y 的商，R 是余数，则补码除法运算的规则是：

① X 和 Y 以补码形式参加除法运算，商也以补码的形式产生。余数和被除数、除数均采用双符号位。

② 当 $[X]_{补}$与 $[Y]_{补}$同号时，第一次做 $[X]_{补}+[-Y]_{补}$操作，当异号时，第一次做 $[X]_{补}+[Y]_{补}$操作，得到第一次的部分余数 $[R0]_{补}$。

二、补码除法及实现

补码除法运算的规则是：

③ 当 $[Ri]_{补}$ 与 $[Y]_{补}$ 同号时，上商 1，然后余数先左移一位，加 $[-Y]_{补}$ 得到新余数 $[Ri+1]_{补}$；当 $[Ri]_{补}$ 与 [Y] 补异号时，上商 0，余数左移一位，加 $[Y]_{补}$ 得到新余数 $[Ri+1]_{补}$。

④ 循环操作步骤③，共做 n 次，得到 1 位商符和（n-1）位商的补码数值位，最末位采用

$[X]_{补}$ 与 $[Y]_{补}$	商符	第一次操作	$[R_i]_{补}$ 与 $[Y]_{补}$	上商		下一步操作
				真值	补码	
同号	0	减法 $[X]_{补}+[-Y]_{补}$	同号（够减）	1	1	余数左移一位，$+[-Y]_{补}$
			异号（不够减）	0	0	余数左移一位，$+[Y]_{补}$
异号	1	加法	同号（不够减）	0	1	余数左移一位，$+[-Y]_{补}$

二、补码除法及实现

❖ 第二种方法的运算规则为：

① X 和 Y 以补码形式参加除法运算，商也以补码的形式产生。余数和被除数、除数均采用双符号位。部分余数初始为 $[X]_补$，即 $[R0]_补=[X]_补$。

② 当 $[Ri]_补$与 $[Y]_补$同号时，上商 1，然后余数先左移一位，加 $[-Y]_补$得到新余数 $[Ri+1]_补$；当 $[Ri]_补$与 $[Y]_补$异号时，上商 0，余数左移一位，加 $[-Y]_补$得到新余数 $[Ri+1]_补$。

③ 循环操作步骤②，共做 n 次，得到 1 位商符和（n-1）位商的补码数值位，最末位采用恒置“1”法。

二、补码除法及实现

X=+0.1011，Y=−0.1101，用补码不恢复余数算法计算X÷Y。

[X]补=00.1011

[Y]补=11.0011

[−Y]补=00.1101

第一种方法：

被除数/余数	商Q	操作说明
00.1011	0 0 0 0 0	[X]补与[Y]补异号
+ 11.0011		+[Y]补
11.1110	0 0 0 0 1	[R₀]补与[Y]补同号，上商1
11.1100	0 0 0 1 0	左移一位
+ 00.1101		+[−Y]补
00.1001	0 0 0 1 0	[R₁]补与[Y]补异号，上商0
01.0010	0 0 1 0 0	左移一位
+ 11.0011		+[Y]补
00.0101	0 0 1 0 0	[R₂]补与[Y]补异号，上商0
00.1010	0 1 0 0 0	左移一位
+ 11.0011		+[Y]补
11.1101	0 1 0 0 1	[R₃]补与[Y]补同号，上商1
11.1010	1 0 0 1 1	左移一位，末位置1

得[Q]补=1.0011 Q=−0.1101

二、补码除法及实现

第二种方法：

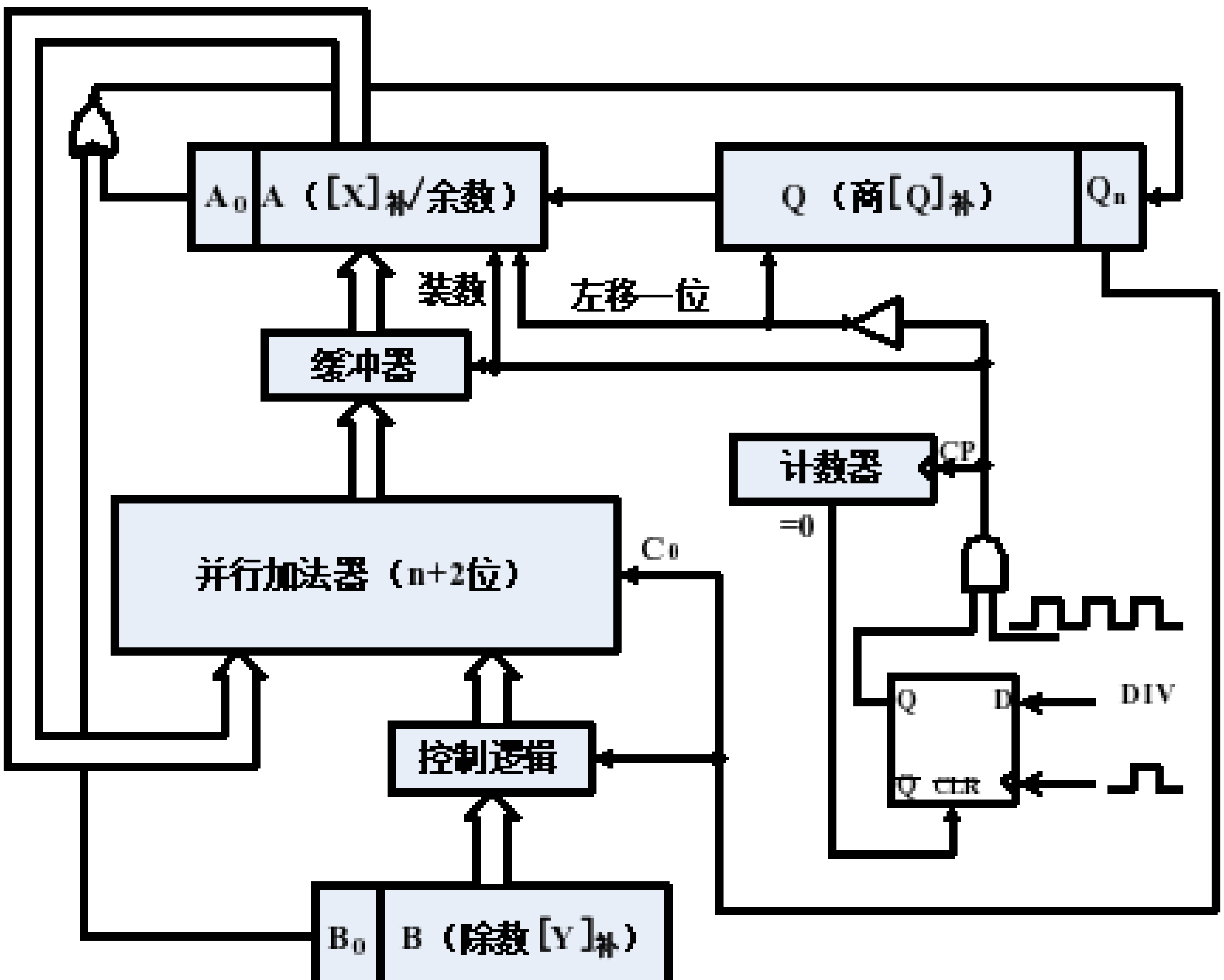
被除数/余数	商Q	操作说明
00.1011	0 0 0 0 0	$[R_0]_补$与$[Y]_补$异号，上商0
01.0110	0 0 0 0 0	左移一位
+ 11.0011		$+[Y]_补$
00.1001	0 0 0 0 0	$[R_1]_补$与$[Y]_补$异号，上商0
01.0010	0 0 0 0 0	左移一位
+ 11.0011		$+[Y]_补$
00.0101	0 0 0 0 0	$[R_2]_补$与$[Y]_补$异号，上商0
00.1010	0 0 0 0 0	左移一位
+ 11.0011		$+[Y]_补$
11.1101	0 0 0 0 1	$[R_3]_补$与$[Y]_补$同号，上商1
11.1010	0 0 0 1 1	左移一位，末位置1

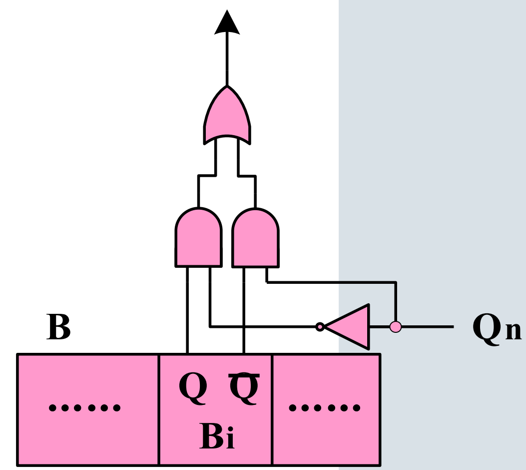
Q的符号取反，得$[Q]_补$=1.0011

。

二、补码除法及实现

补码不恢复余数除法的电路框图（第二种方法）

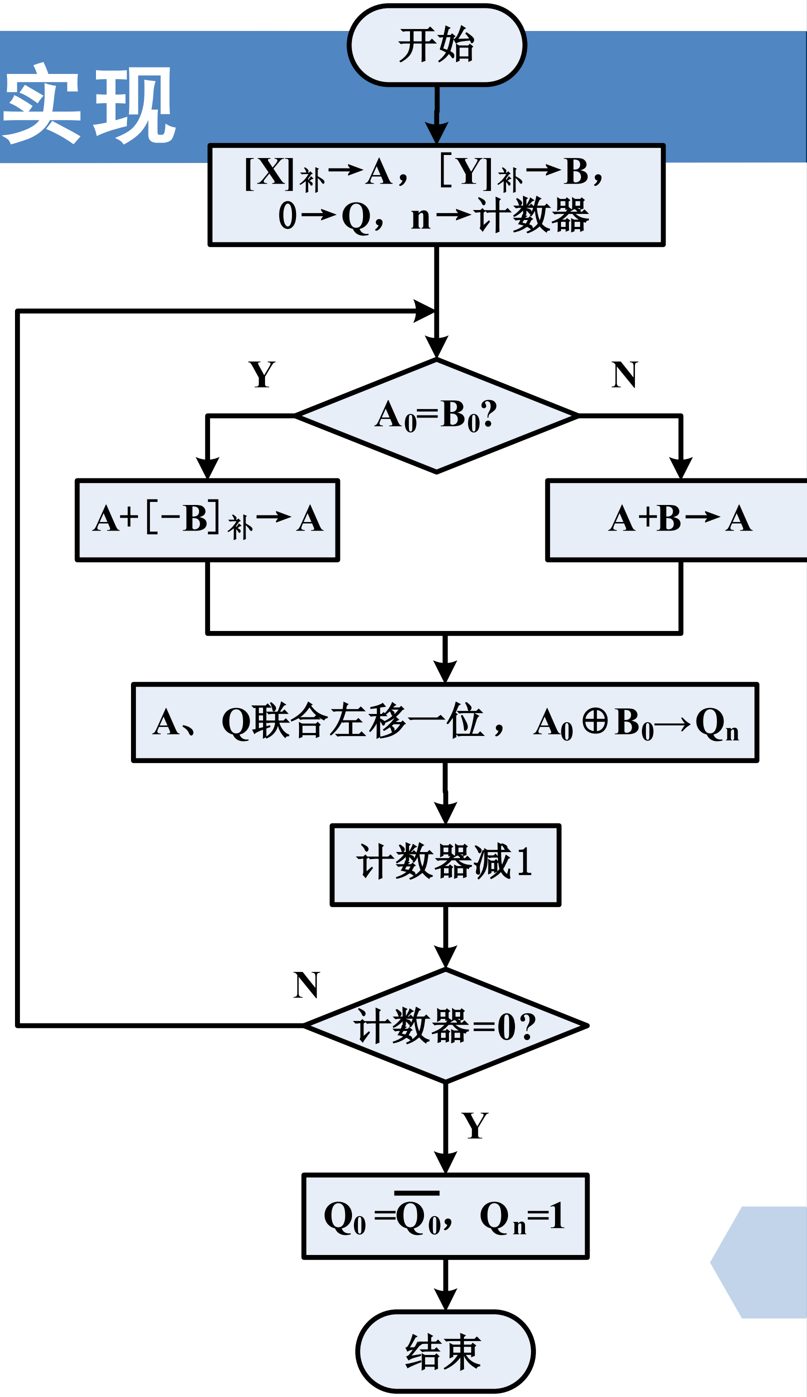




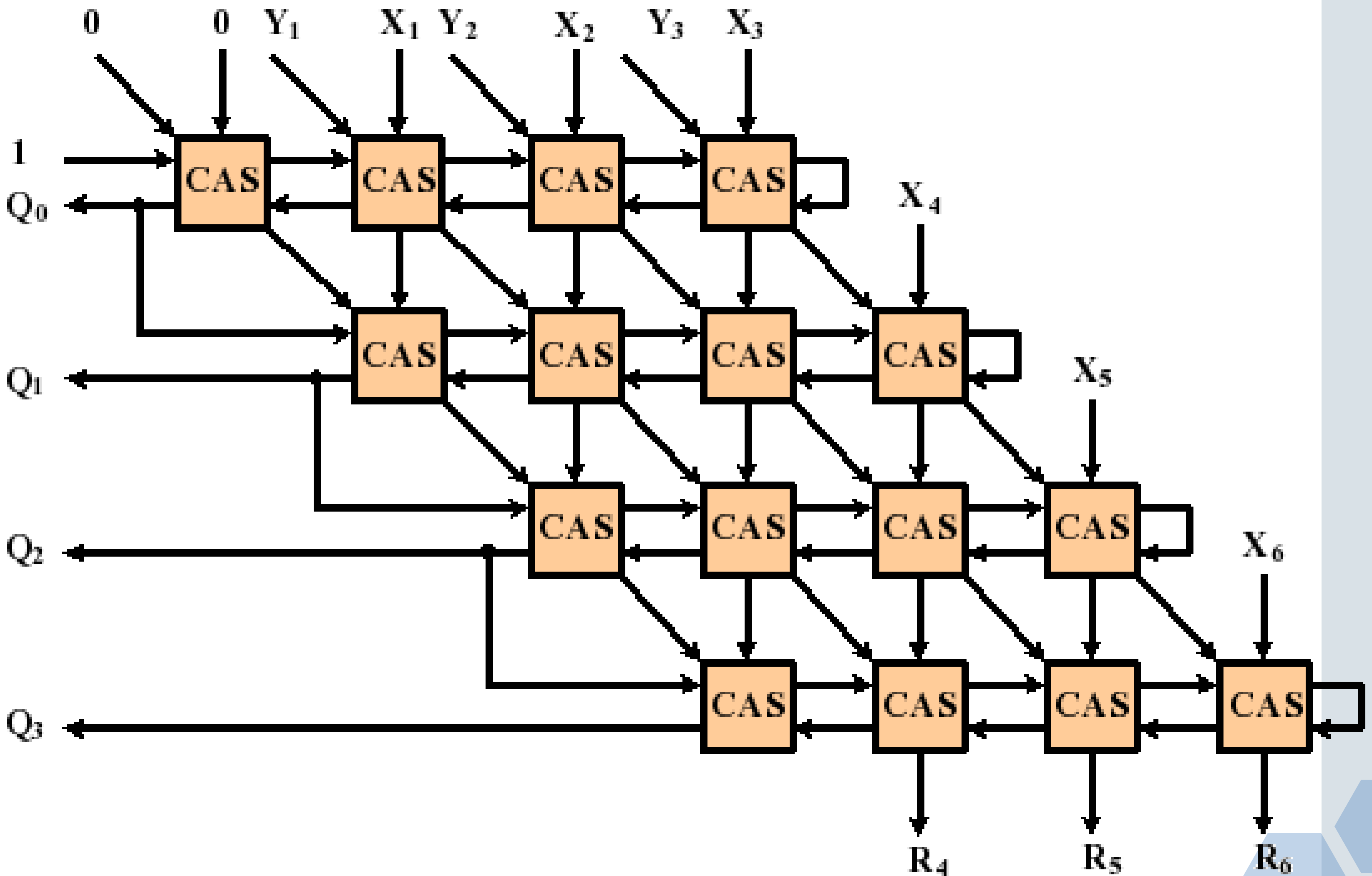
控制电路逻辑

二、补码除法及实现

补码不恢复余数除法流程



三、阵列除法器



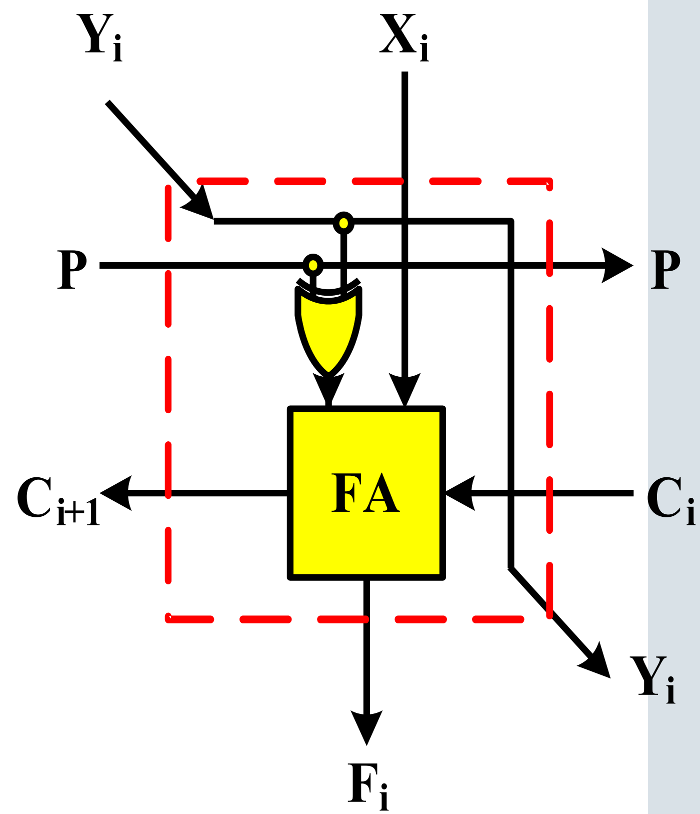
三、阵列除法器

- 定点整数：
  - 被除数 $X=X_1X_2X_3X_4X_5X_6$
  - 除数 $Y=Y_1Y_2Y_3$
  - 商 $Q=Q_1Q_2Q_3$ （ Q_0= 0 ）
  - $R= R_4\ R_5\ R_6$

- 定点小数
  - $X=0.X_1X_2X_3X_4X_5X_6$
  - 除数 $Y= 0.\ Y_1Y_2Y_3$
  - 商 $Q=0.Q_1Q_2Q_3$ （ Q_0= 0 ）
  - R=0.000 R R R

三、阵列除法器

- 构成的基本部件：可控加减单元 CAS
- 算法：加减交替算法
- P=1 做减法； P=0 做加法

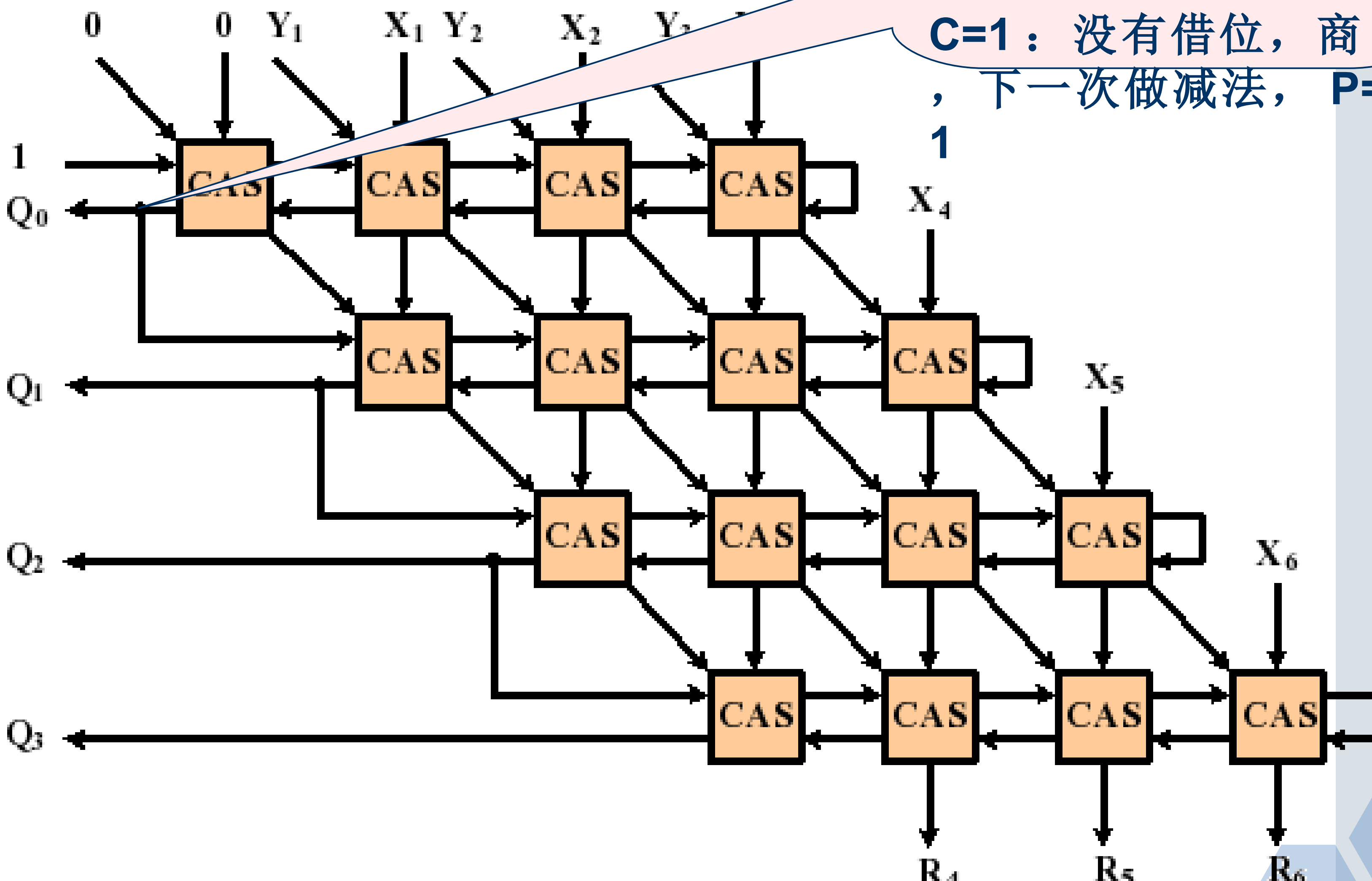


可控加减单元 CAS

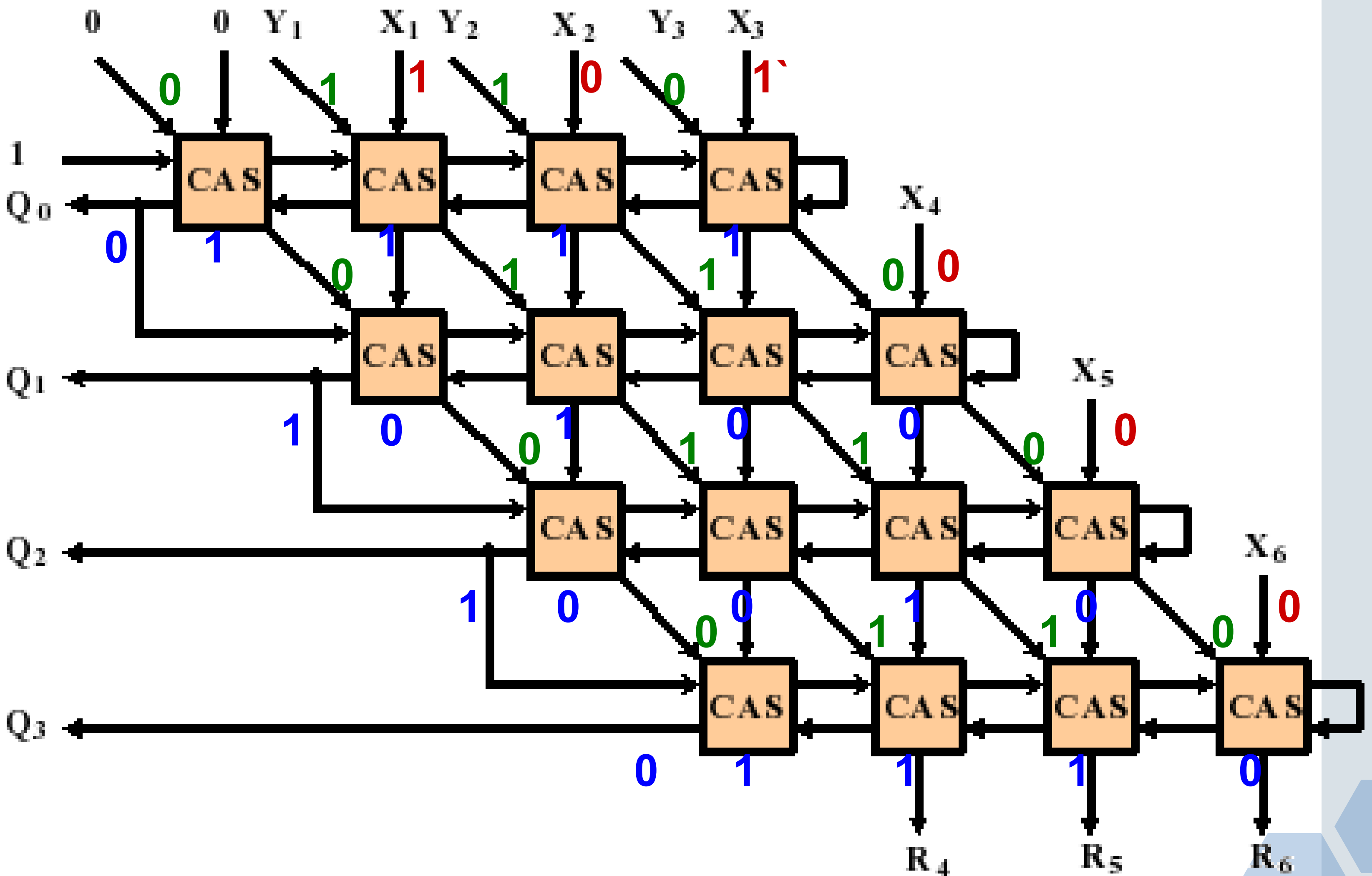
三、阵列除法器

C=0：有借位，商 0，下一次做加法，P=0

C=1：没有借位，商 1，下一次做减法，P=1



举例：0.101000÷0.110



The End !