2.4 定点除法运算

提纲

2.4.1 原码除法运算原理

- 两个原码表示的数相除时，商的符号由两数的符号按位异或求得，商的数值部分由两数的数值部分相除求得。

- 设有n位定点小数(定点整数也同样适用)：
  - 被除数 x, 其原码为 $[x]_{原} = x_f . x_{n-1} \cdots x_1 x_0$
  - 除数 y，　其原码为 $[y]_{原} = y_f . y_{n-1} \cdots y_1 y_0$
- 则有商$q = x/y$，其原码为
  - $[q]_{原} = (x_f \oplus y_f)(0.x_{n-1} \cdots x_1 x_0 / 0.y_{n-1} \cdots y_1 y_0)$

2.4.1 原码除法运算原理

- 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。商的数值部分的运算，实质上是两个正数求商的运算。根据我们所熟知的十进制除法运算方法，很容易得到二进制数的除法运算方法，所不同的只是在二进制中，商的每一位不是“1”就是“0”，其运算法则更简单一些。

2.4.1 原码除法运算原理

```
          0.2 1 4
      ------------
9 8 5 / 2 1 1
        0 0 0
      ------------
        2 1 1 0
        1 9 7 0
      ------------
          1 4 0 0
            9 8 5
      ------------
            4 1 5 0
            3 9 4 0
      ------------
                2 1 0
```

```
                0.2 1 4
          ------------
0.9 8 5 / 0.2 1 1
          0.0 0 0
        ------------
          0.2 1 1 0
          0.1 9 7 0
        ------------
            0.0 1 4 0 0
              0.0 0 9 8 5
        ------------
              0.0 0 4 1 5 0
              0.0 0 3 9 4 0
        ------------
                  0.0 0 0 2 1 0
```

2.4.1 原码除法运算原理

					0.	1	1	0	1
0 1 1 0 1 /	0	1	0	1	1				
	0	0	0	0	0				
		1	0	1	1	0			
		0	1	1	0	1			
			1	0	0	1	0		
			0	1	1	0	1		
				0	1	0	1	0	
				0	0	0	0	0	
					1	0	1	0	0
					0	1	1	0	1
						0	1	1	1

2.4.1 原码除法运算原理

					0.	1	1	0	1
0.	1	1	0	1	0.	1	0	1	1
					0.	0	0	0	0
					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	0	0	
					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

2.4.1 原码除法运算原理

- 设被除数 x = 0.1001，除数 y = 0.1011，模仿十进制除法运算，以手算方法求 x ÷ y 的过程如下:

2.4.1 原码除法运算原理

	0.1101	商 q
0.1011	0.10010	$x\ (r_0)$ 被除数小于除数，商 0
	−0.01011	$2^{-1}y$ 除数右移 1 位，减除数，商 1
	0.001110	r_1 得余数 r1
	−0.00101 1	$2^{-2}y$ 除数右移 1 位，减除数，商 1
	0.000011 0	r_2 得余数 r2
	−0.000101 1	$2^{-3}y$ 除数右移 1 位，不减除数，商 0
	0.0001100	r_3 得余数 r3
	−0.0001011	$2^{-4}y$ 除数右移 1 位，减除数，商 1
	−0.0000001	r_4 得余数 r4

- 得 x ÷ y 的商q = 0.1101，余数为r = 0.00000001。

2.4.1 原码除法运算原理

- 上面的笔算过程可叙述如下:
  - 1. 判断 x 是否小于 y? 现在 $x < y$, 故商的整数位商“0”, x 的低位补0, 得余数 r_0。
  - 2.比较 r_0 和 $2^{-1}y$, 因 $r_0 > 2^{-1}y$, 表示够减, 小数点后第一位商“1”, 作 $r_0 - 2^{-1}y$, 得余数 r_1。
  - 3. 比较 r_1 和 $2^{-2}y$, 因 $r_1 > 2^{-2}y$, 表示够减, 小数点后第二位商“1”, 作 $r_1 - 2^{-2}y$, 得余数 r_2。
  - 4. 比较 r_2 和 $2^{-3}y$, 因 $r_2 < 2^{-3}y$, 不够减, 小数点后第三位商“0”, 不作减法, 得余数 $r_3 (= r_2)$。
  - 5. 比较 r_3 和 $2^{-4}y$, 因 $r_3 > 2^{-4}y$, 表示够减, 小数点后第四2位商“1”, 作 $r_3 - 2^{-4}y$, 得余数 r_4,
  - 共求四位商, 至此除法完毕。

2.4.1 原码除法运算原理

- 机器与人运算过程不同，人会心算一看就知道够不够减。但机器却必须先作减法，若余数为正才知道够减；若余数为负才知道不够减

- 不够减时必须恢复原来的余数以便再继续往下运算，称为恢复余数法。要恢复原来的余数，只要当前的余数加上除数即可。但由于要恢复余数，使除法进行过程的步数不固定，因此控制比较复杂

- 实际中常用不恢复余数法又称加减交替法。其特点是运算过程中如出现不够减则不必恢复余数，根据余数符号，可以继续往下运算，因此步数固定，控制简单

2.4.1 原码除法运算原理

- 不恢复余数的除法原理
- 设第i次求商的余数为$R_i=R_{i-1}-Y_i$，$[R_i]_{补}=[R_{i-1}]_{补}$（双符号位00）$+[-Y_i]_{补}$（双符号位11），其中，R_{i-1}是上次求商的余数（被除数），Y_i是上次求商的除数（根据运算到第几次，是原除数做相应移位后的数据）
- 若低符号位运算向前产生的进位为1，则最高符号位为0，最高符号位代表运算后的数的真正符号，说明$R_{i-1}>Y_i$，也即减法够减，商1，此时不需要恢复余数，根据除法运算规则，下一次除数的余数为：$R_{i+1}=R_i-2^{-1}Y_i$
- 若低符号位运算向前产生的进位为0，则最高符号位为1，最高符号位代表运算后的数的真正符号，说明$R_{i-1}<Y_i$，也即减法不够减，商0，此时需要恢复余数，即需要恢复R_{i-1}，计算方法为：R_{i-1}（本次的余数应该为的数）$=R_i$（本次计算的结果）$+Y_i$，计算根据除法运算规则，下一次除数的余数为：$R_{i+1}-2^{-1}Y=R_i+Y_i-2^{-1}Y=R_i+2^{-1}Y$

2.4.1 原码除法运算原理

- 不恢复余数的除法

- 运算规则如下：
  - ①首先用被除数减去除数，得到的结果称为余数；
  - ②若符号位运算向前产生的进位为“0”，则商“0”，将除数向右错开1位，再用余数加上除数；若符号位运算向前产生的进位为“1”，则商“1”，将除数向右错开1位，再用余数减去除数；
  - ③重复②，若商的符号位为1位，数值位为n位，则重复②的操作共n+1次，最后一次上商后余数不再运算。

2.4.1 原码除法运算原理

- 例：设x=101001，y=-111，用原码阵列除法器计算x÷y。
- 解：$[x]_{原}$=0101001 $[y]_{原}$=1111

  商的符号位为：$x_f \oplus y_f = 0 \oplus 1 = 1$

  令x′ =101001，y′ =111，其中x′ 和y′ 分别为$[x]_{原}$和$[y]_{原}$的数值部分

  $[x']_{补}$=0101001，$[y']_{补}$=0111，$[-y']_{补}$=1001

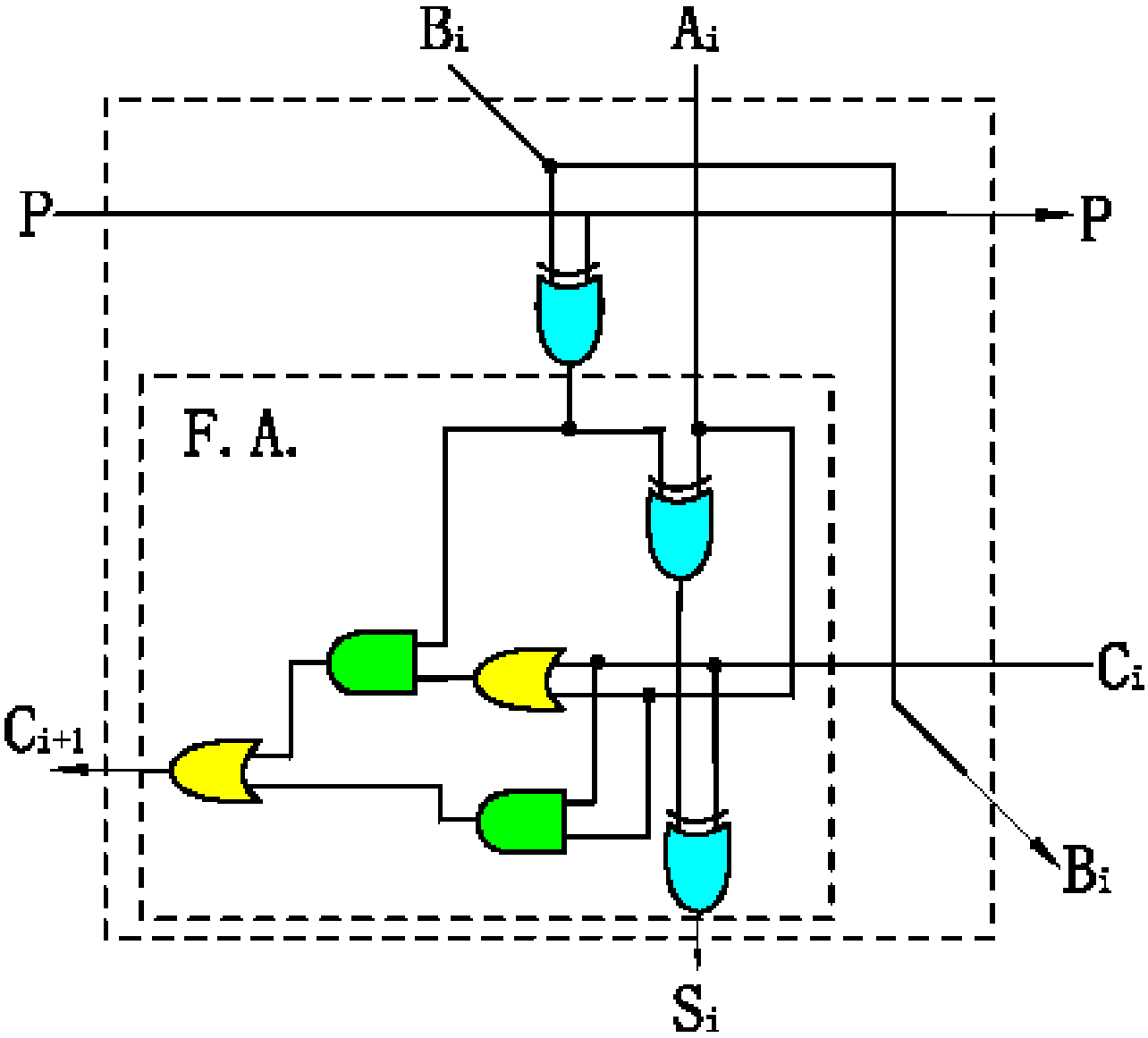
	被除数/余数	商	说明
	0101001		被除数$[x']_{补}$
$+[-y']_{补}$	1001		第一步减去除数，即$+[-y']_{补}$
	1110001	$q_0=0$	最高位向前产生的进位为 0，即商 0
$+[y']_{补}\longrightarrow$	0111		向右错开 1 位，加上除数，即$+[y']_{补}$
	001101	$q_1=1$	最高位向前产生的进位为 1，即商 1
$+[-y']_{补}\longrightarrow$	1001		向右错开 1 位，减去除数，即$+[-y']_{补}$
	11111	$q_2=0$	最高位向前产生的进位为 0，即商 0
$+[y']_{补}\longrightarrow$	0111		向右错开 1 位，加上除数，即$+[y']_{补}$
	0110	$q_3=1$	最高位向前产生的进位为 1，即商 1

2.4.2 并行除法器

- 一、可控加法/减法(CAS)单元

- 和阵列乘法器非常相似，阵列式除法器也是一种并行运算部件，采用大规模集成电路制造。与早期的串行除法器相比，阵列除法器不仅所需的控制线路少，而且能提供令人满意的高速运算速度。
- 阵列除法器有多种多样形式，如不恢复余数阵列除法器，补码阵列除法器等等。
- 先介绍可控加法/减法(CAS)单元，它将用于并行除法流水逻辑阵列中，它有四个输出端和四个输入端。当输入线P = 0时，CAS作加法运算；当P = 1时，CAS作减法运算。逻辑结构图：

2.4.2 并行除法器



可控加法/减法（CAS）单元有四个输入端和四个输出端。当输入线P=0时，CAS做加法运算；当P=1时，CAS做减法运算。

2.4.2 并行除法器

- CAS单元的输入与输出的关系可用如下一组逻辑方程来表示：

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

$$C_{i+1} = (A_i + C_i) \cdot (B_i \oplus P) + A_i C_i \qquad (2.32)$$

- 当P = 0时，方程式(2.32)就等于我们前面学习的一位全加器(FA)的公式：

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i B_i + B_i C_i + A_i C_i$$

- 当P = 1时，则得求差公式：

$$S_i = A_i \oplus \overline{B}_i \oplus C_i$$

$$C_{i+1} = A_i \overline{B}_i + \overline{B}_i C_i + A_i C_i \qquad (2.33)$$

其中$\overline{B}_i = B_i \oplus 1$。

2.4.2 并行除法器

- 在减法情况下，输入C_i称为借位输入，而C_{i+1}称为借位输出。
- $X \oplus Y = X\bar{Y} + \bar{X}Y$
- 为说明CAS单元的实际内部电路实现，将方程式(2.32)加以变换，可得如下形式：

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

$$S_i = A_i \oplus (B_i \oplus P) \oplus C_i$$
$$= A_i B_i \bar{C_i} P + A_i \bar{B_i} \bar{C_i} P + \bar{A_i} B_i C_i P + A_i B_i C_i \bar{P} +$$
$$A_i \bar{B_i} C_i P + \bar{A_i} \bar{B_i} \bar{C_i} P + \bar{A_i} \bar{B_i} C_i \bar{P} + \bar{A_i} \bar{B_i} C_i P$$

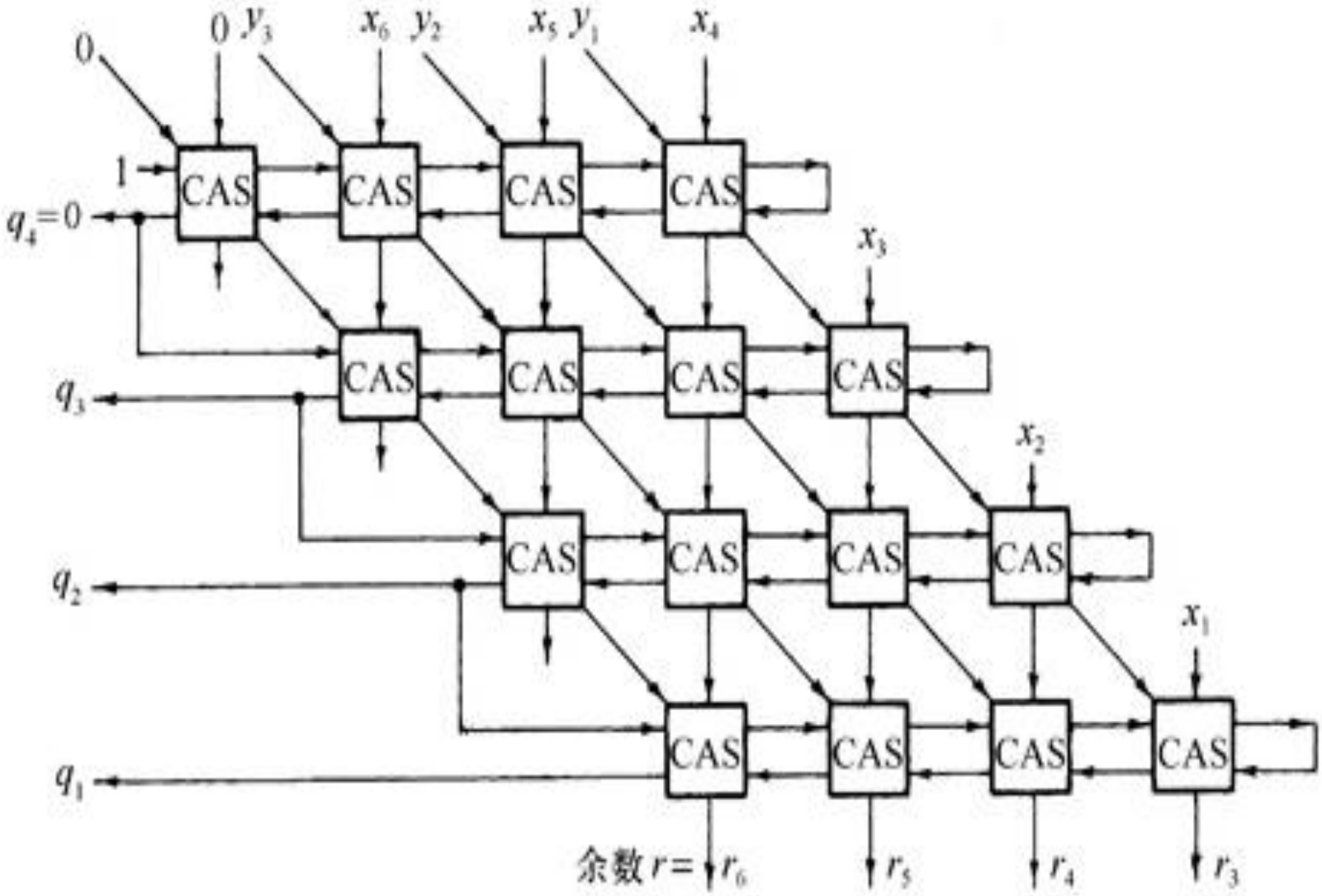
- 在这两个表达式中，每一个都能用一个三级组合逻辑电路（包括反向器）来实现
- 求所有的非1T，求所有的与1T，求所有的或1T，因此每一个基本的CAS单元的延迟时间为3T单元

2.4.2 并行除法器

- 二、不恢复余数的阵列除法器

- 先假定所有被处理的数都是正的小数。
- 不恢复余数的除法也就是加减交替法。在不恢复余数的除法阵列中，每一行所执行的操作究竟是加法还是减法，取决于前一行输出的符号与被除数的符号是否一致。当出现不够减时，部分余数相对于被除数来说要改变符号。这时应该产生一个商位“0”，除数首先沿对角线右移，然后加到下一行的部分余数上。当部分余数不改变它的符号时，即产生商位“1”，下一行的操作应该是减法。下图是4位除4位的不恢复余数阵列除法器的逻辑原理图。

2.4.2 并行除法器



2.4.2 并行除法器

- 其中

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 $r=0.00r_6r_5r_4r_3$

除数右移

- 由上图看出，该阵列除法器是用一个可控加法/减法(CAS)单元所组成的流水阵列来实现的。推广到一般情况，一个2n位除以n位的加减交替除法阵列由$(n+1)^2$个CAS单元组成，其中两个操作数(被除数与除数)都是正的。

2.4.2 并行除法器

- 最上面一行所执行的初始操作经常是减法。因此最上面一行的控制线P固定置成“1”。减法是用2的补码运算来实现的，这时右端各CAS单元上的反馈线用作初始的进位输入。每一行最左边的单元的进位输出决定着商的数值。将当前的商反馈到下一行，我们就能确定下一行的操作。由于进位输出信号指示出当前的部分余数的符号，因此，它将决定下一行的操作将进行加法还是减法。
- 对不恢复余数阵列除法器来说，在进行运算时，沿着每一行都有进位(或借位)传播，同时所有行在它们的进位链上都是串行连接。而每个CAS单元的延迟时间为3T单元，因此，对一个2n位除以n位的不恢复余数阵列除法器来说，单元的数量为$(n+1)^2$，考虑最大情况下的信号延迟，其除法执行时间为：

$$td = 3(n+1)^2T$$

其中n为尾数位数。