

3.1.6 原码一位乘法如何进行?

1. 原码一位乘法的运算规则

设 $x = x_1, x_1 x_2 \dots x_n, y = y_1, y_1 y_2 \dots y_n$, 乘积为 P , 乘积的符号位为 P_f , 则

$$P_f = x_f \oplus y_f \quad |P| = |x| \cdot |y|$$

求 $|P|$ 的运算规则如下:

(1) 被乘数和乘数均取绝对值参加运算, 符号位单独考虑。

(2) 被乘数取双符号, 部分积的长度同被乘数, 初值为 0。

(3) 从乘数的最低位 y_n 开始判断, 若 $y_n = 1$, 则部分积加上被乘数 $|x|$, 然后右移一位; 若 $y_n = 0$, 则部分积加上 0, 然后右移一位。

(4) 重复(3), 判断 n 次。

2. 原码一位乘法的实现

定点运算部件的框图如图 3.1.2 所示。

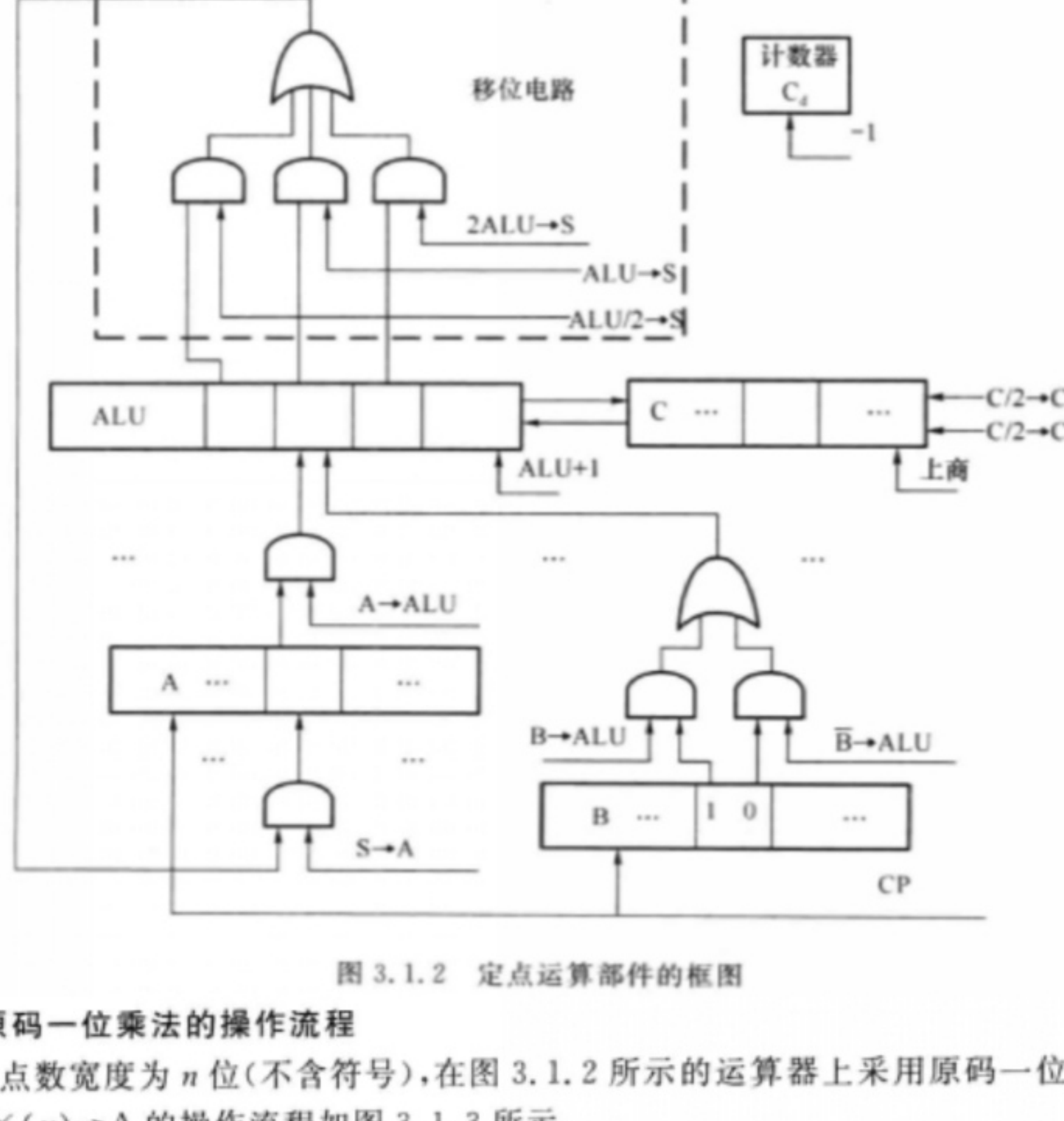


图 3.1.2 定点运算部件的框图

3. 原码一位乘法的操作流程

设定点数宽度为 n 位(不含符号), 在图 3.1.2 所示的运算器上采用原码一位乘法完成运算 $(x) \times (y) \rightarrow A$ 的操作流程如图 3.1.3 所示。

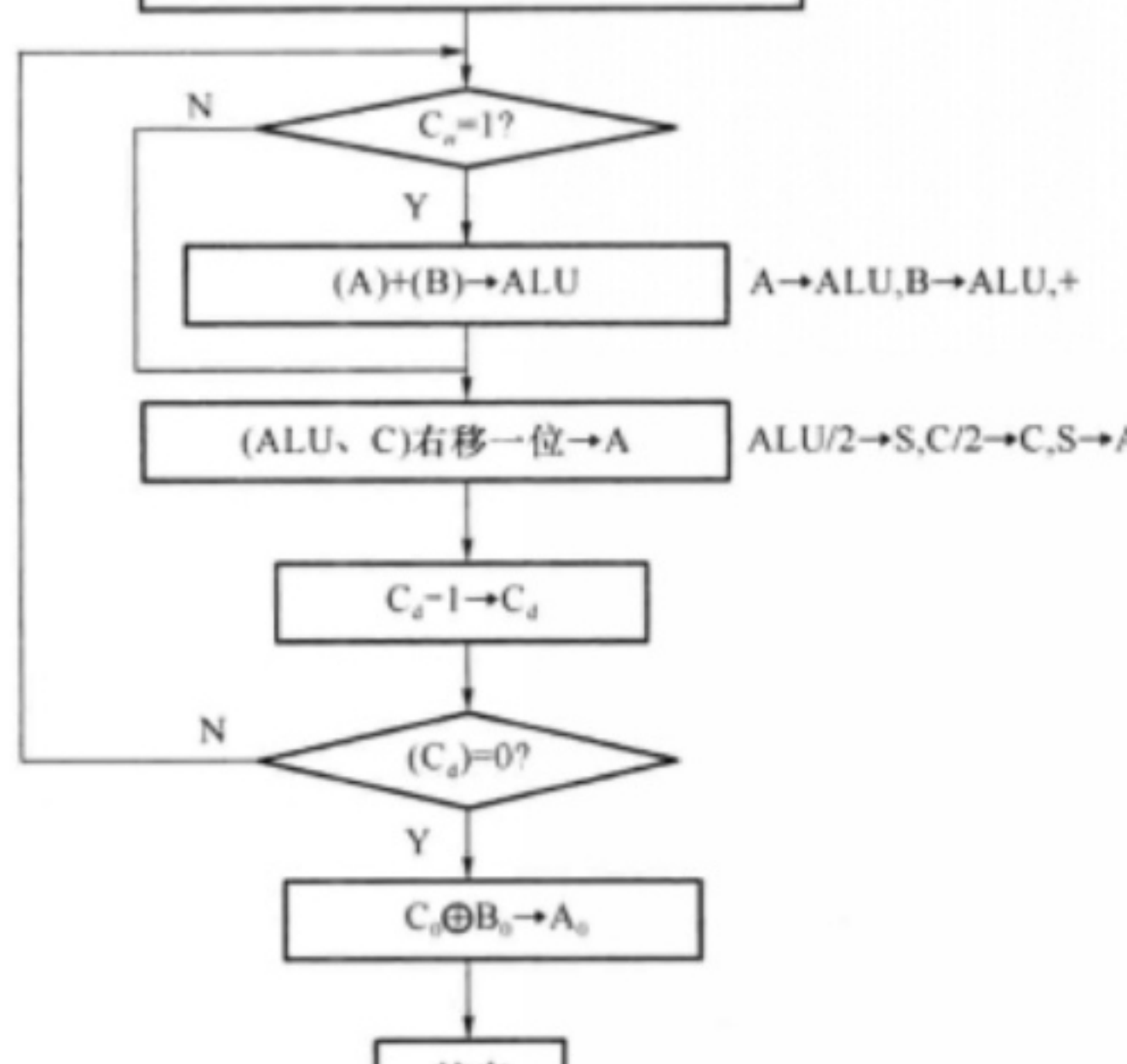


图 3.1.3 原码一位乘法的操作流程

3.1.7 原码两位乘法如何进行?

运算规则:

(1) 符号位不参加运算, 最后的符号 $P_f = x_f \oplus y_f$ 。

(2) 部分积与被乘数均采用 3 位符号, 乘数末位增加 1 位 C , 其初值为 0。

(3) 按表 3.1.1 操作。

表 3.1.1 原码两位乘法算法

$y_{n-1}y_nC$	操作
0 0 0	加 0, 右移两位, $0 \rightarrow C$
0 0 1	加 $ x $, 右移两位, $0 \rightarrow C$
0 1 0	加 $ x $, 右移两位, $0 \rightarrow C$
0 1 1	加 $2 x $, 右移两位, $0 \rightarrow C$
1 0 0	加 $2 x $, 右移两位, $0 \rightarrow C$
1 0 1	减 $ x $, 右移两位, $1 \rightarrow C$
1 1 0	减 $ x $, 右移两位, $1 \rightarrow C$
1 1 1	加 0, 右移两位, $1 \rightarrow C$

(4) 若尾数字长 n 为偶数(不含符号), 乘数用双符号, 最后一步不移位; 若尾数字长 n 为奇数(不含符号), 乘数用单符号, 最后一步移一位。

3.1.8 补码一位乘法如何进行?

1. 补码一位乘法的运算规则

补码一位乘法的运算算法是 Booth 夫妇首先提出来的, 所以也称 Booth 算法, 其运算规则如下:

(1) 符号位参与运算, 运算的数均以补码表示。

(2) 被乘数一般取双符号位, 参加运算, 部分积初值为 0。

(3) 乘数可取单符号位, 以决定最后一步是否需要校正, 即是否加 $[-x]_H$ 。

(4) 乘数末位增设附加位 y_{n+1} , 且初值为 0。

(5) 按表 3.1.2 进行操作。

表 3.1.2 补码一位乘法算法

y_n (高位)	y_{n+1} (低位)	操作
0	0	部分积右移一位
0	1	部分积加 $[x]_H$, 右移一位
1	0	部分积加 $[-x]_H$, 右移一位
1	1	部分积右移一位

(6) 按照上述算法进行 $n+1$ 步操作, 但第 $n+1$ 步不再移位, 仅根据 y_0 与 y_1 的比较结果作相应的运算即可。

2. 补码一位乘法的操作流程

实现补码一位乘法的框图如图 3.1.2 所示, 设定点数宽度为 n 位(不含符号), 在该图所示的运算器上采用补码一位乘法完成运算 $(x) \times (y) \rightarrow A$ 的操作流程如图 3.1.4 所示。

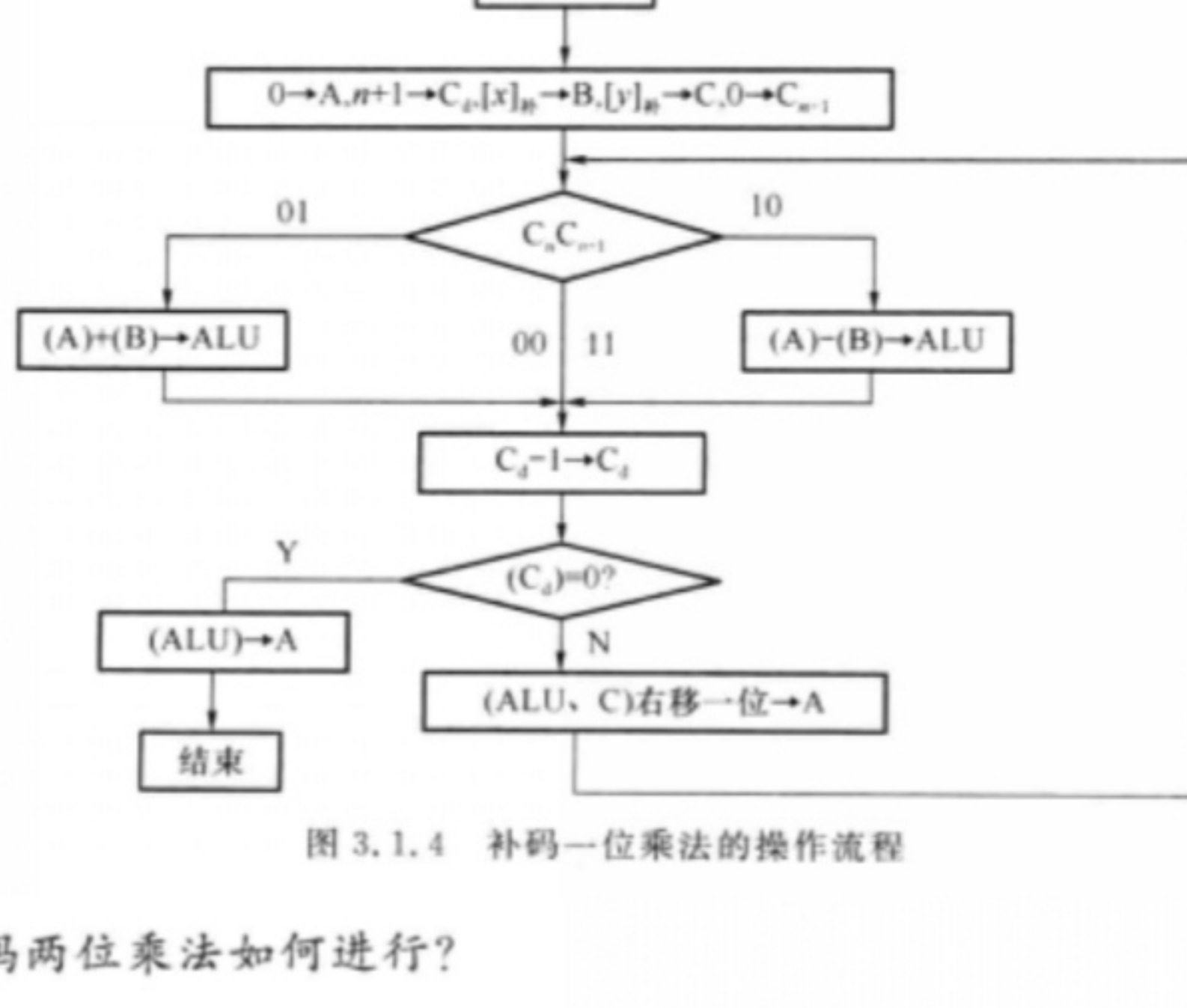


图 3.1.4 补码一位乘法的操作流程

3.1.9 补码两位乘法如何进行?

补码两位乘法的运算规则:

(1) 符号位参加运算, 两数均用补码表示。

(2) 部分积与被乘数均采用 3 位符号, 乘数末位增加一位 y_{n+1} , 其初值为 0。

(3) 按表 3.1.3 操作。

表 3.1.3 补码两位乘法算法

$y_{n-1}y_ny_{n+1}$	操作
0 0 0	加 0, 右移两位
0 0 1	加 $[x]_H$, 右移两位
0 1 0	加 $[x]_H$, 右移两位
0 1 1	加 $2[x]_H$, 右移两位
1 0 0	加 $2[-x]_H$, 右移两位
1 0 1	加 $[-x]_H$, 右移两位
1 1 0	加 $[-x]_H$, 右移两位
1 1 1	加 0, 右移两位

(4) 若尾数字长 n 为偶数(不含符号), 乘数用双符号, 最后一步不移位;

若尾数字长 n 为奇数(不含符号), 乘数用单符号, 最后一步移一位。

3.1.10 原码一位除法如何进行?

1. 原码一位除法的运算规则

设被除数 $[x]_H = x_1, x_1 x_2 \dots x_n$, 除数 $[y]_H = y_1, y_1 y_2 \dots y_n$, 则

商的符号为: $Q_f = x_f \oplus y_f$;

商的数值为: $|Q| = |x| / |y|$ 。

求 $|Q|$ 的加减交替法(不恢复余数法)运算规则:

(1) 符号位不参加运算, 并要求 $|x| < |y|$ 。

(2) 先用被除数减去除数, 当余数为正时, 商上 1, 余数左移一位, 再减去除数; 当余数为负时, 商上 0, 余数左移一位, 再加上除数。

(3) 当第 $n+1$ 步余数为负时, 需加上 $|y|$ 得到第 $n+1$ 步正确的余数, 最后余数为 $r^n \times 2^{-n}$ (余数与被除数同号)。

2. 原码一位除法的操作流程

实现原码一位除法的框图如图 3.1.5 所示。

设定点数宽度为 n 位(不含符号), 在图 3.1.5 所示的运算器上采用原码一位不恢复余数法完成 $(x) \div (y) \rightarrow C$, 余数 $\rightarrow A$, 其操作流程如图 3.1.5 所示。

