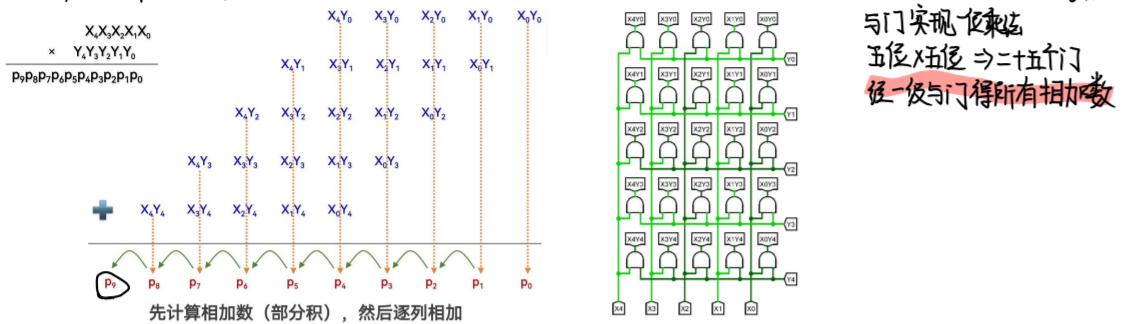


\*关键路径=进位链

CPU中乘法实现方法  
①执行乘法运算子程序实现乘法运算 无硬件成本 Intel 8088/8080、RISC32-I指令集  
②设置专用乘法器实现乘法运算 高硬件成本 [原码、补码乘法器] 大部分现代处理器

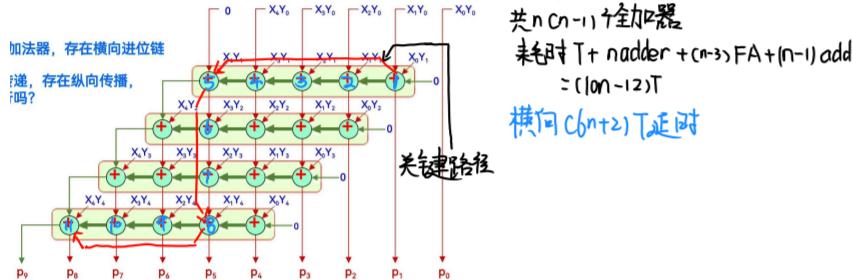
原码乘法 将符号单独运算 直接完成  
绝对值相乘 如何实现定点数绝对值相乘

二进制手工乘法运算

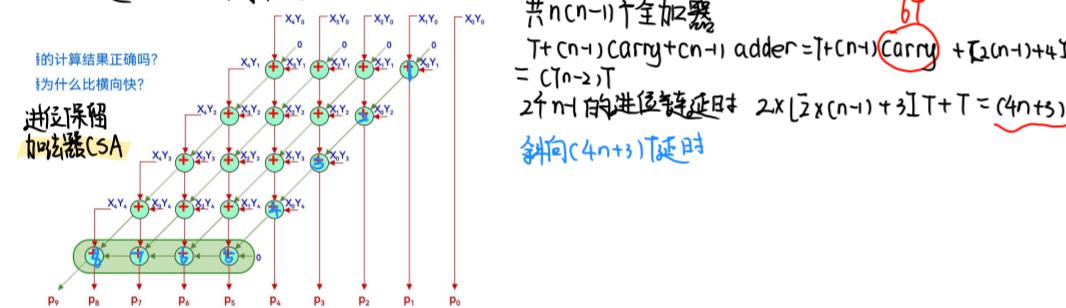


设置专用乘法器实现乘法运算

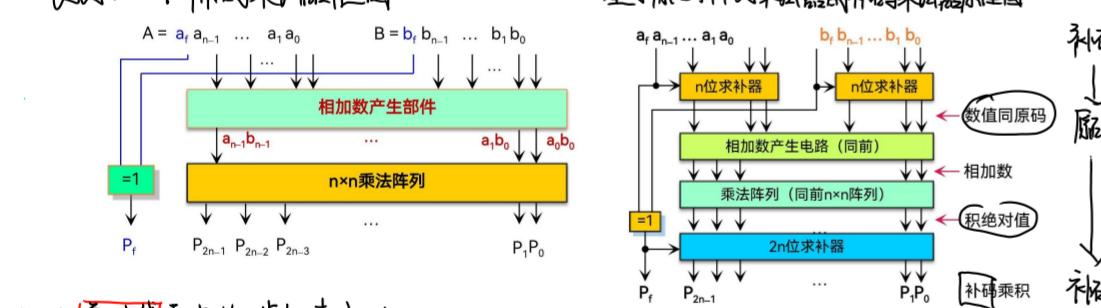
①横向进位列乘法器 ( $n(n-1)$ 个全加器,  $2n$ 级全加器延时)



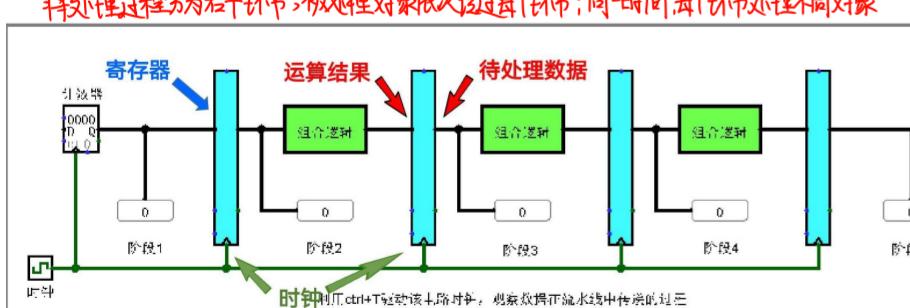
②斜向进位无符号阵列乘法器



$n \times n$ 位原码乘法器框图

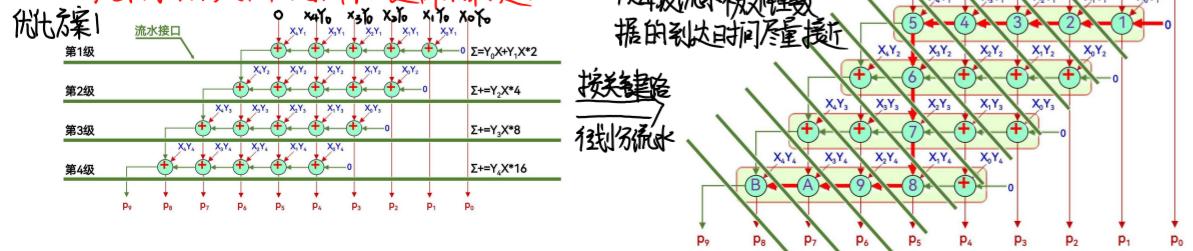


将处理过程分为若干环节；被处理对象依次经过每个环节；同一时间，每个环节处理不同对象



流水线不能减少数据输出延迟，可提高总的数据通路速率

流水线时钟频率取决于各个组合逻辑的最大延时



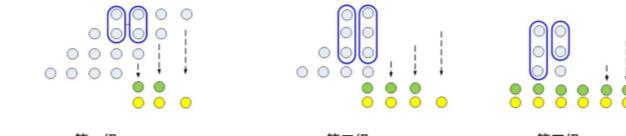
其他优化技术

① Booth两位乘法 (Intel 8086)

乘数从MSB开始，3位一组产生一个部分积，相邻组重叠一位，到LSB不足3位补零，部分积个数减少一半  
能直接实现补码运算

乘数位	编码位
000	0
001	+ 被乘数
010	+ 被乘数
011	+ 2×被乘数
100	- 2×被乘数
101	- 被乘数
110	- 被乘数
111	0

② Wallace树：CSA实现4×4计部分积和



电路  $\Rightarrow 3\text{HA} + 3\text{FA}$ ; 延时:  $HAT2FA$

Wallace树实现多数据并行求和，产生两行和 (较少加器有 $log n$ 级全加器延时  
连线约束)



\*采用乘法器  $\Rightarrow$  被乘数乘以乘数的每一位得部分积，部分求和得乘积

CSA：进位保留加法器(斜向进位加法器)

Booth两位编码：将乘数两位产生一个部分积，减少部分积总数，直接进行补码运算

Wallace树加法器：多数据并行压缩

流水线 提高乘法器时钟频率

利用加法器与配置相应的配件完成乘法

原码部分积累加的数学表达

$$[X]_{\text{原}} = X_0 X_1 X_2 \dots X_{n-1} X_n; [Y]_{\text{原}} = Y_0 Y_1 Y_2 \dots Y_{n-1} Y_n$$

$$\Sigma_1 = X Y_n$$

$$\Sigma_2 = X Y_{n-1} + X Y_n 2^{-1}$$

$$\Sigma_3 = X Y_{n-2} + (X Y_{n-1} + X Y_n 2^{-1}) 2^{-1} = X Y_{n-2} + X X Y_{n-1} 2^{-1} + X X Y_n 2^{-2}$$

$$X \times Y = X [Y_1 2^{-1} + Y_2 2^{-2} + \dots + Y_n 2^{-n}]$$

只用到了与、移位和求和运算

作位加法，一定移位，~~操作n次加法和移位；跨号位单独计算~~

开始  $i=0, \Sigma=0$

$[X]_{\text{原}} \times [Y]_{\text{原}}$

$\Sigma = \Sigma + X$

$\Sigma = \Sigma + Y$

$i = i + 1$

$P_0 = X_i \oplus Y_0$

结束

乘数判断位  $Y_{n-1}$

■ 指令：分无符号数乘指令、带符号整数乘指令

■ 无符号：若  $P=0$ ，则不溢出

■ 带符号：若  $P$  每位都等于  $P$  的最高位（即均为符号位），则不溢出

■ 乘法指令可生成溢出标志，高级语言编译器使用标志位或乘积高位来判断是否溢出

■ 变量与常数间快速乘法（整数乘法比浮点乘法快很多）

编译器在处理变量与常数相乘时，用其它快速运算指令代替乘法

$X \times 2^0 \rightarrow x \ll 4 + x \ll 2 \Rightarrow$  和直接相乘结果相同

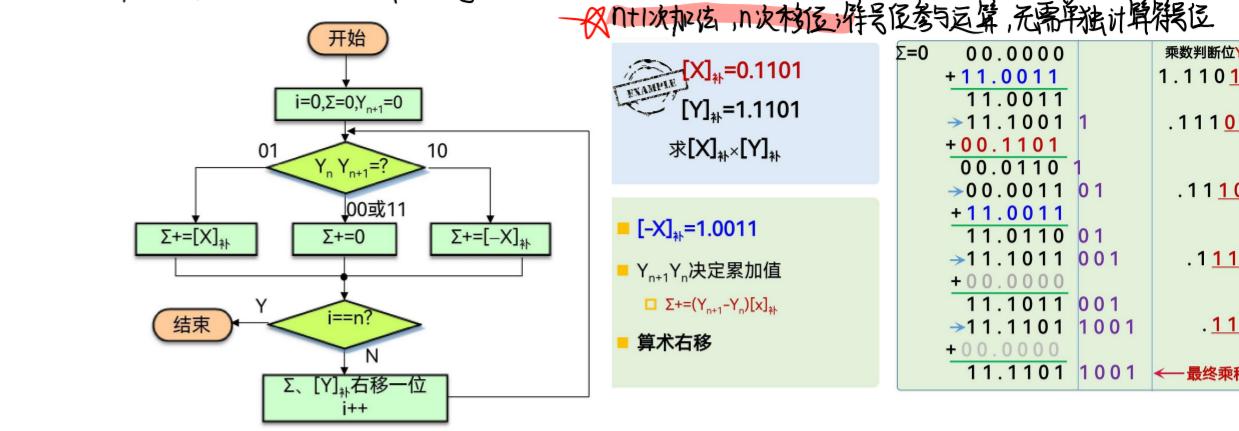
$X \times 15 \rightarrow x \ll 4 - x \Rightarrow$  和直接相乘结果相同

是否优化取决于组成运算周期数是否小于乘法开销

设计专用硬件电路时也可能采用此策略

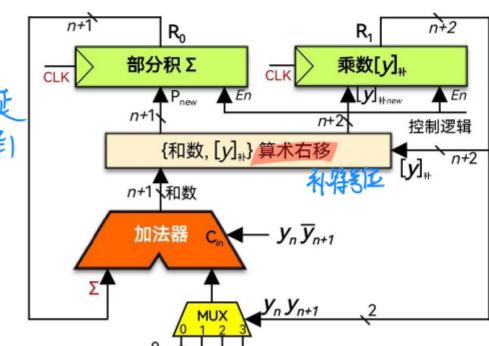
补码乘法实现  $\Rightarrow$  1位 booth算法 ( $n$ 个全加器,  $n$ 个时钟周期, 面积小)

与原码乘法的区别 (1) 相邻两位  $Y_i$  决定累加的值 (2) 有  $n+1$  次累加,  $n$  次移位



$[X]_{\text{#}} = 0.1101$	00.0000	1.11010
$\rightarrow 11.0011$	11.0011	.11101
$\rightarrow 11.1011$	00.0110	1.1110
$\rightarrow 00.0111$	01	.111
$\rightarrow 11.0111$	001	.111
$\rightarrow 11.1101$	1001	.111
$\rightarrow 11.1101$	1001	.111
$\rightarrow 00.0000$	1001	.111
$\rightarrow 11.1101$	1001	.111
$\rightarrow 11.1101$	1001	.111

硬件实现



主流乘法器  $\Rightarrow$  2位 Booth算法 + CSA + Wallace树 + 流水

② 整数乘运算的溢出问题

硬件保留  $2n$  位乘积，就结果而言是不会溢出的  
乘法指令的操作数长度为  $n$ ，而乘积长度为  $2n$

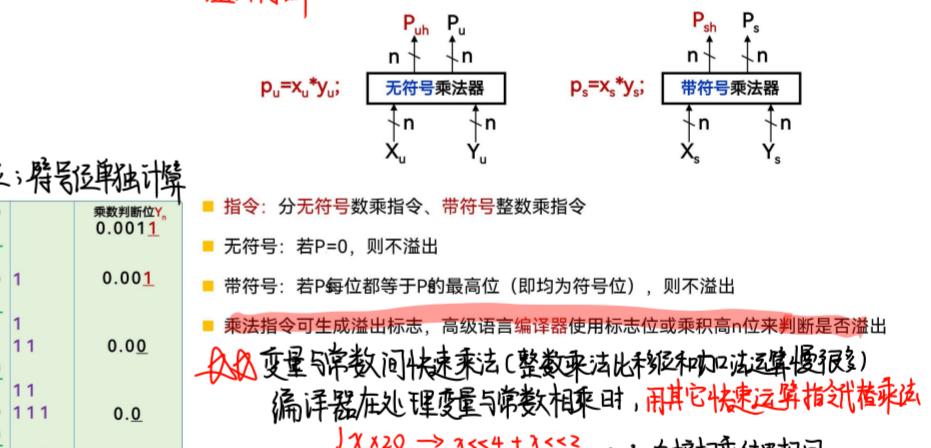
IA32中

8×8乘积存在 AX (16位)

16×16乘积存在 DX-AX (32位)

32×32乘积存在 EDX-EAX (64位)

MIPS中  
32位乘法运算的16位乘积有放在寄存器和Lo中  $\Rightarrow$  结果而不溢出  
溢出判断



硬件实现



$\{\Sigma, Y\} = \{\Sigma + Y_n | X, Y\} / 2$  连同进位右移