处理器体系结构

第三章 处理器中的数值运算A

--整数运算及其电路实现

(Micro-processor Architecture)

处理器中的数值运算

图灵.论可计算数及其在判定问题中的应用,1936.

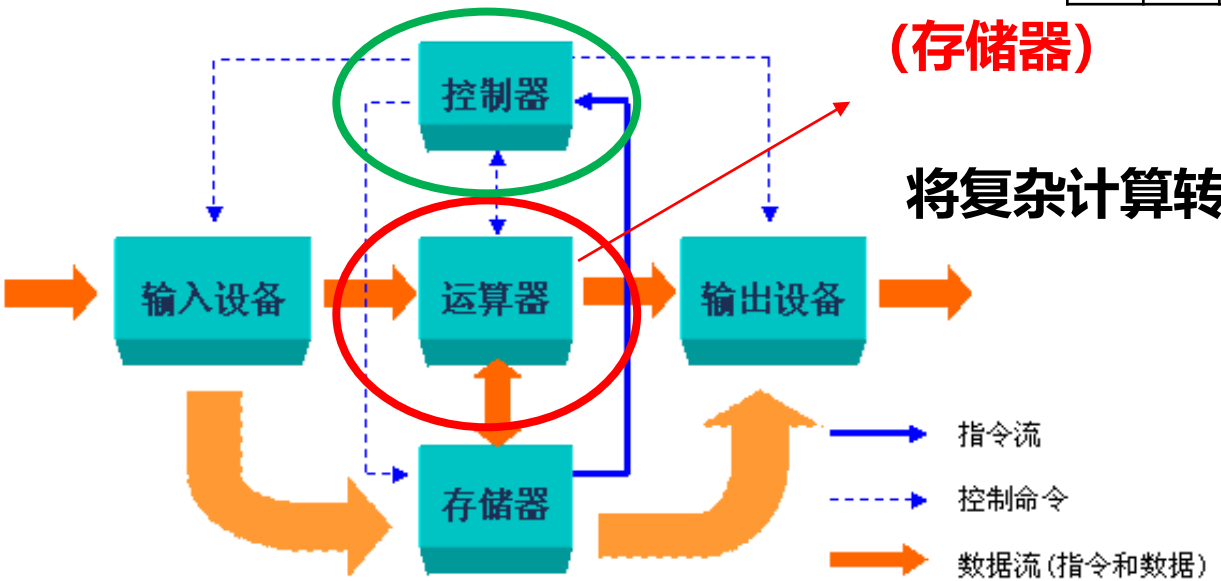
纸带:

(存储器)

读写头指示器 (指针)

将复杂计算转换成非常小的机械操作过程

->加、减、乘、除...



“关于EDVAC的报告草案” –冯.诺依曼

目录

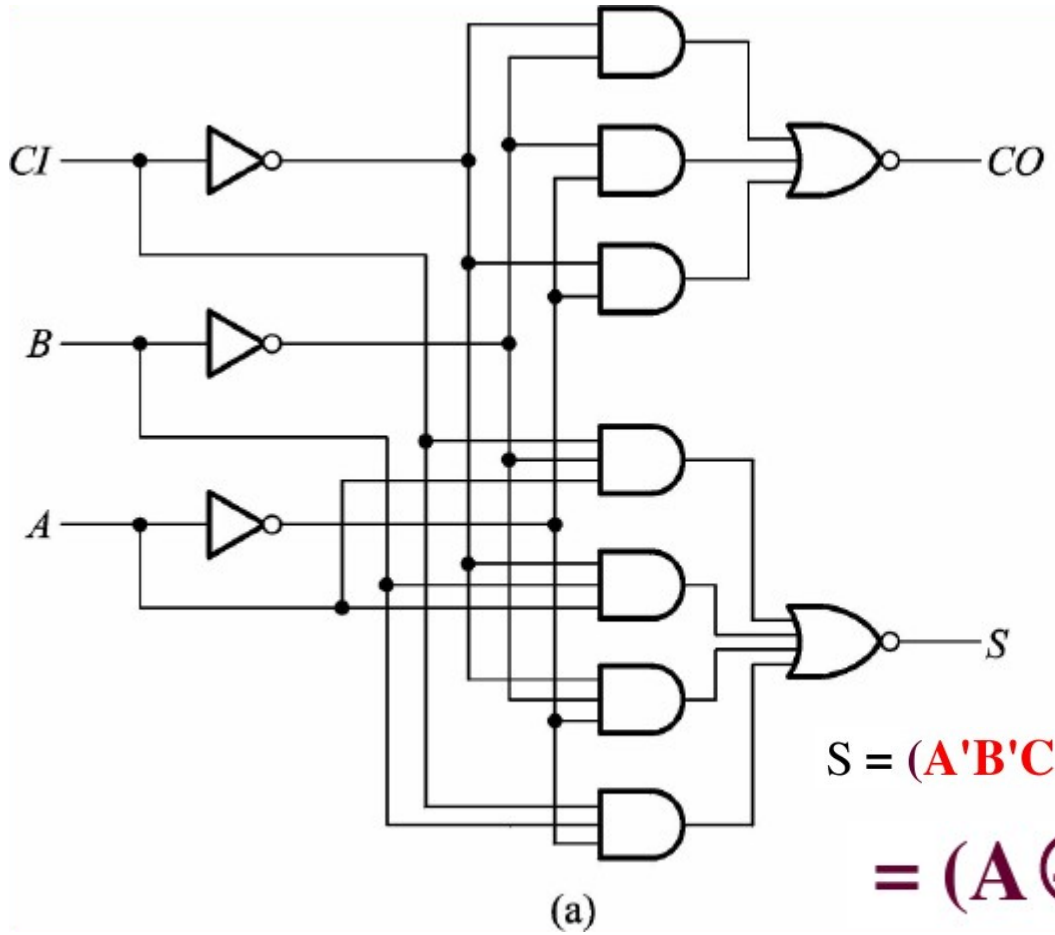
整数运算及其电路实现

1. 整数加减法运算
2. 乘法运算及其电路实现
3. 除法运算及其电路实现
4. MIPS对乘除法运算的支持

目录

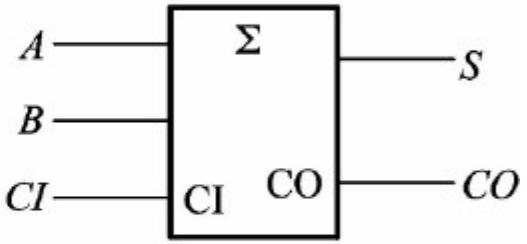
整数运算及其电路实现

1. 整数加减法运算--全加器单元



(a)

$$CO = AB + BC_I + AC_I$$
$$= AB + CI \cdot (A + B)$$

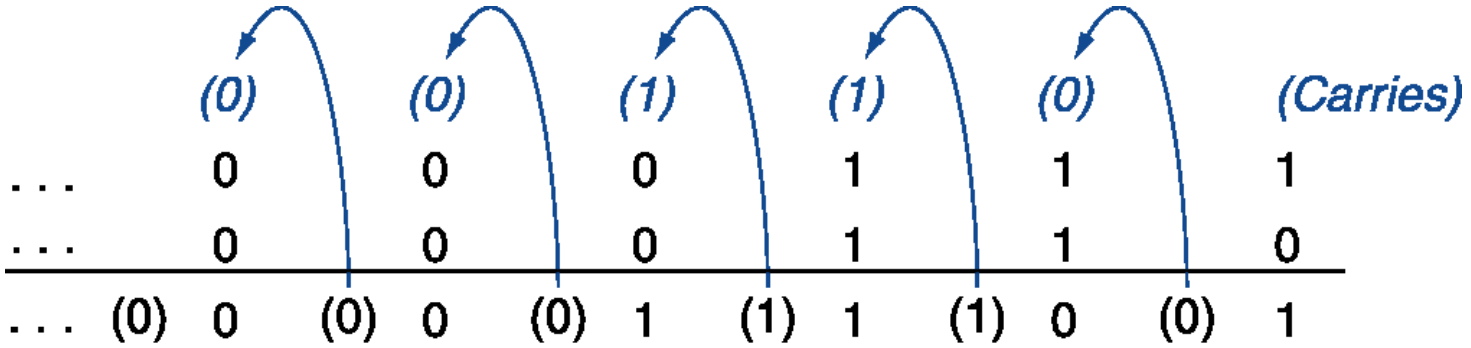


(b)

$$S = (A'B'C'_I + A'BC_I + AB'C_I + ABC'_I)'$$
$$= (A \oplus B) \oplus C_I$$

1.整数加减法运算--整数加法

例：7+6



溢出：数值计算结果超出数的表示范围

什么时候发生溢出?

1、 +ve与-ve操作数相加 ->不产生溢出

2、两个+ve操作数相加，结果符号位是1 ->产生溢出

3、两个-ve操作数相加，结果符号位是0 ->产生溢出

1.整数加减法运算--整数减法

减法可转换为加法：加上操作数的反数

例: 7 – 6 = 7 + (–6)

+7: 0000 0000 … 0000 0111

–6: 1111 1111 … 1111 1010

+1: 0000 0000 … 0000 0001

什么时候发生溢出?

1、两个+ve或两个-ve相减 ->不产生溢出

2、-ve减去+ve，结果符号位为0 ->产生溢出

3、+ve减去-ve，结果符号位为1 ->产生溢出

1.整数加减法运算--溢出的处理

一些语言会忽略溢出，比如C语言

编译成MIPS汇编语言时会采用addu、addui、subu指令

另一些语言会唤醒异常，处理溢出，比如Ada、Fortran语言

编译成MIPS汇编语言时会采用add、addi、sub指令

溢出时，唤醒中断：

1.将PC保存于EPC中

2.跳转至预先定义好的中断处理程序地址

3.Mfc0指令可以取回EPC值，返回原来的地址

饱和操作

当发生溢出时，取最大值

音频、视频

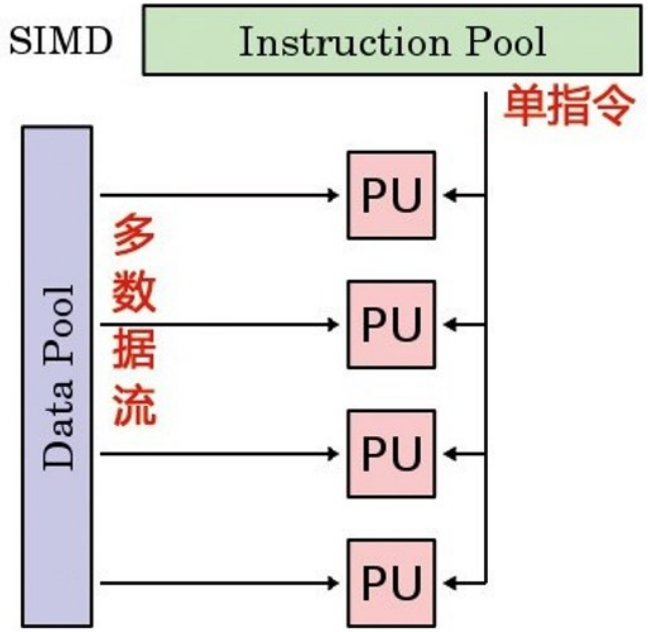
1.整数加减法运算--子字并行

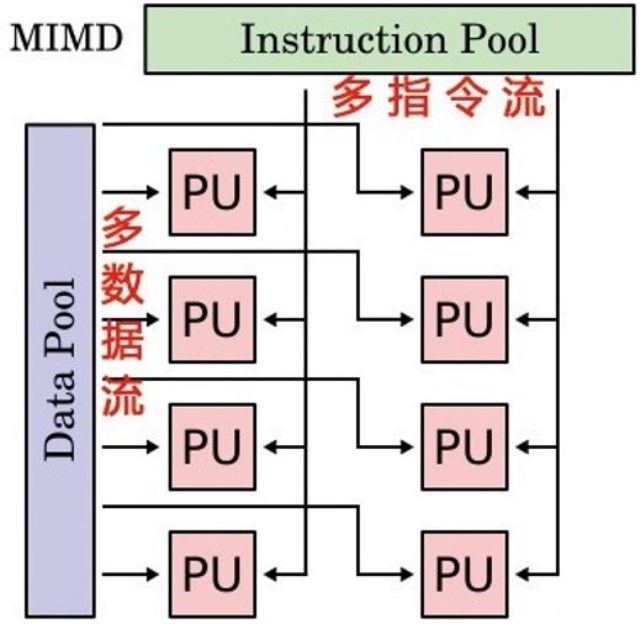
多媒体中的运算一般以8bits或16bits的向量为单位

->直接使用32或64位加法器是一种资源浪费

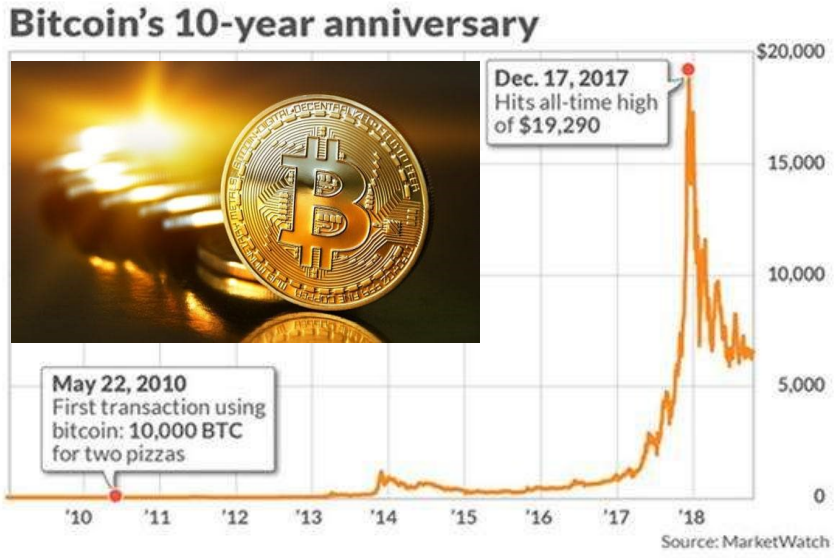
子字并行技术：以64bit加法器为例，可以对8×8bits、4×16bits或2×32bits的向量进行运算

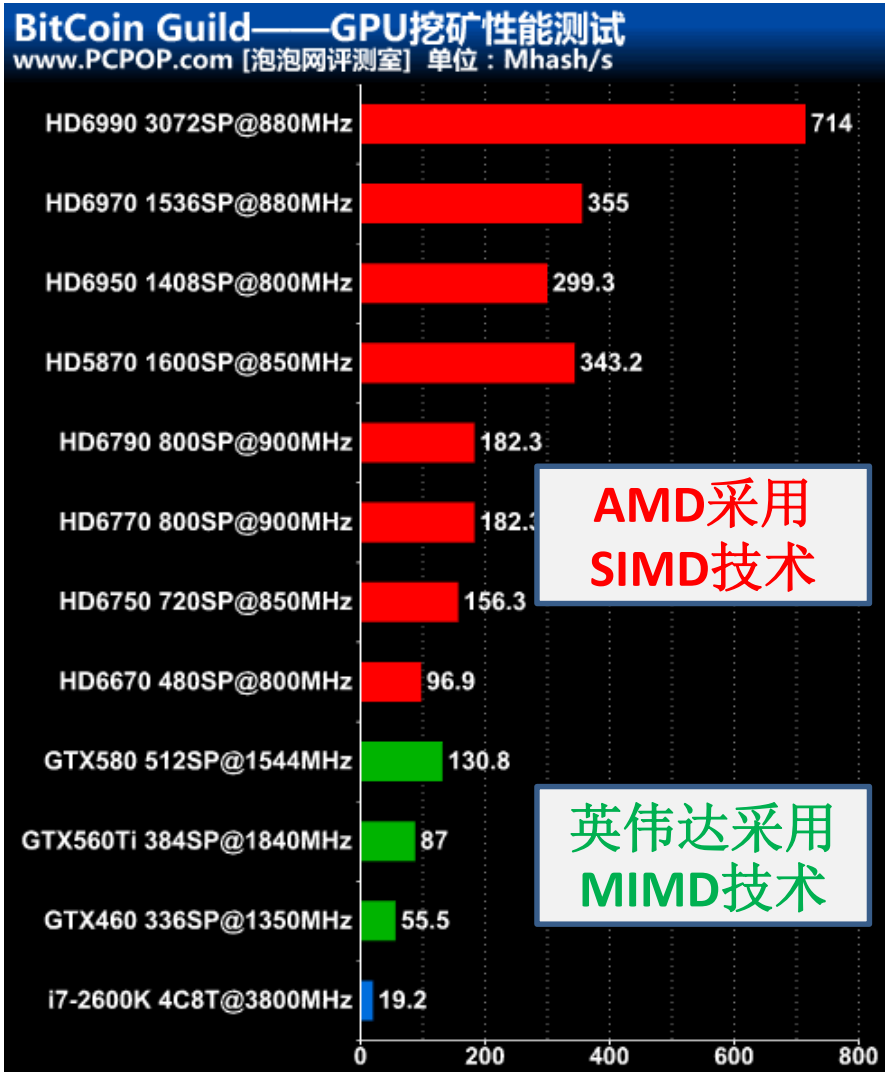
->也叫SIMD：Single instruction，multiple data





1.整数加减法运算--子字并行





SIMD：为达目的，使出洪荒之力；不考虑效率的暴力推进，理论最大性能高，适合单一操作的专用计算（如：挖矿）

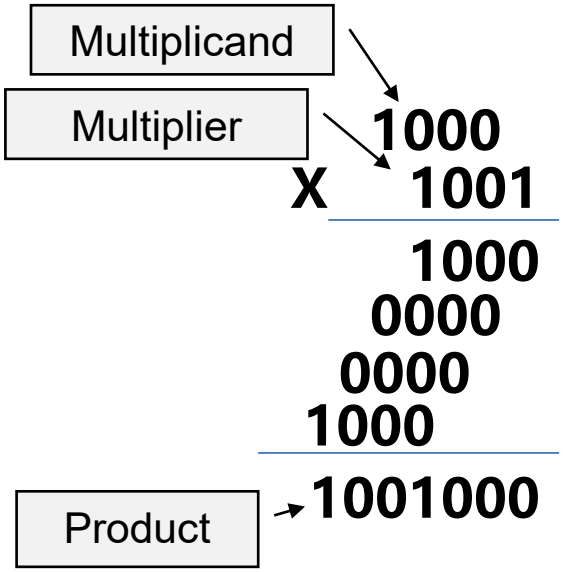
MIMD：弹无虚发，招招致命；同等算力使用更多的晶体管，提高执行效率，适合通用计算

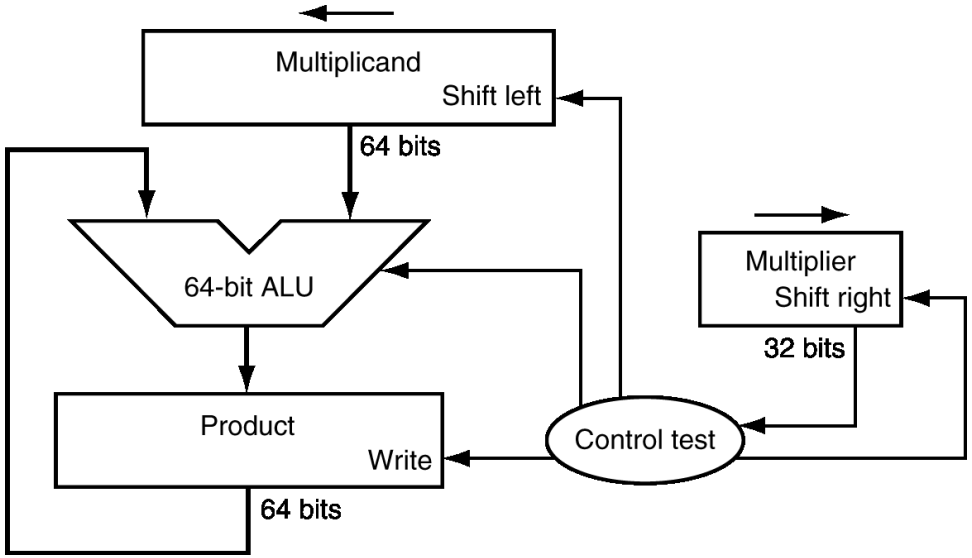
目录

整数运算及其电路实现

2. 乘法运算及其电路实现

基本长乘运算及其电路实现：



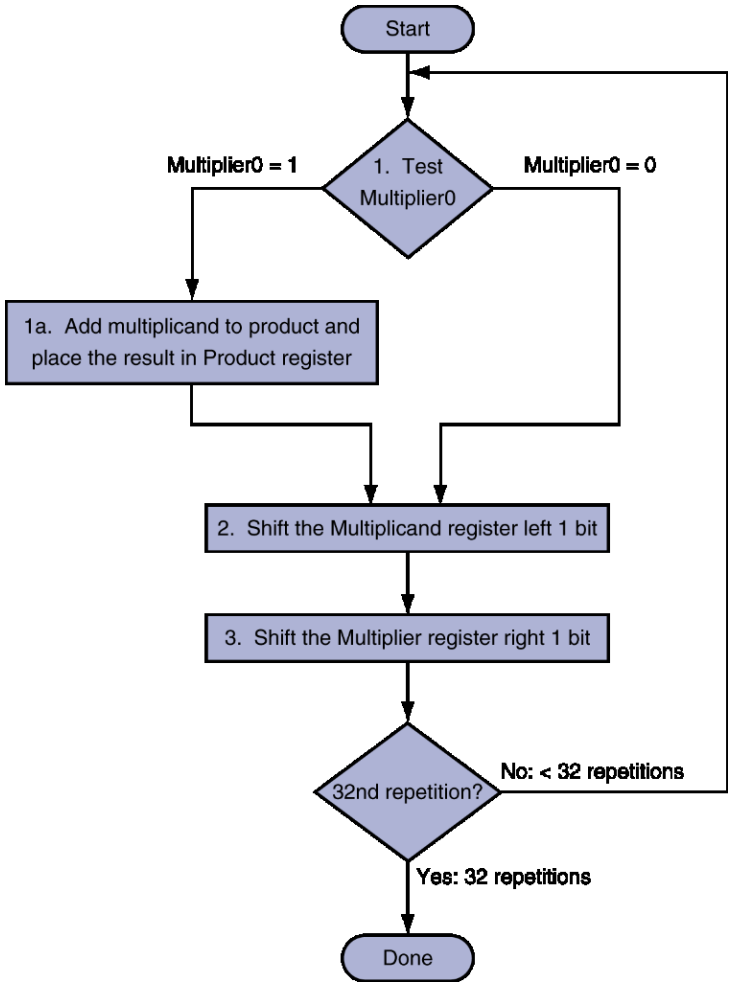


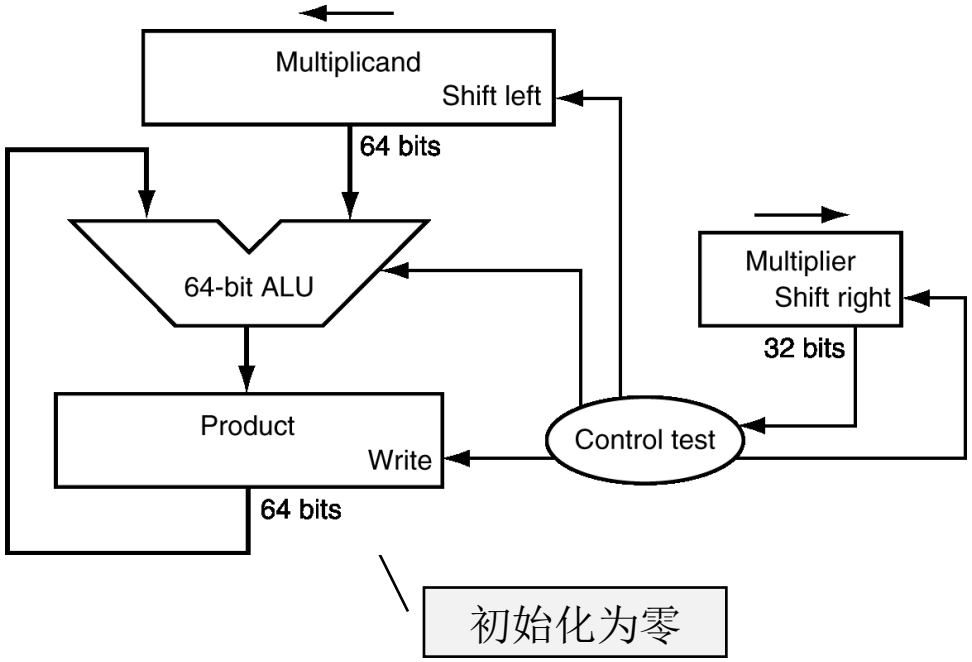
积的最大长度?

两个操作数的长度和

2.乘法运算及其电路实现

基本长乘运算及其电路实现：





2.乘法运算及其电路实现

基本长乘运算及其电路实现：

需要迭代32次！

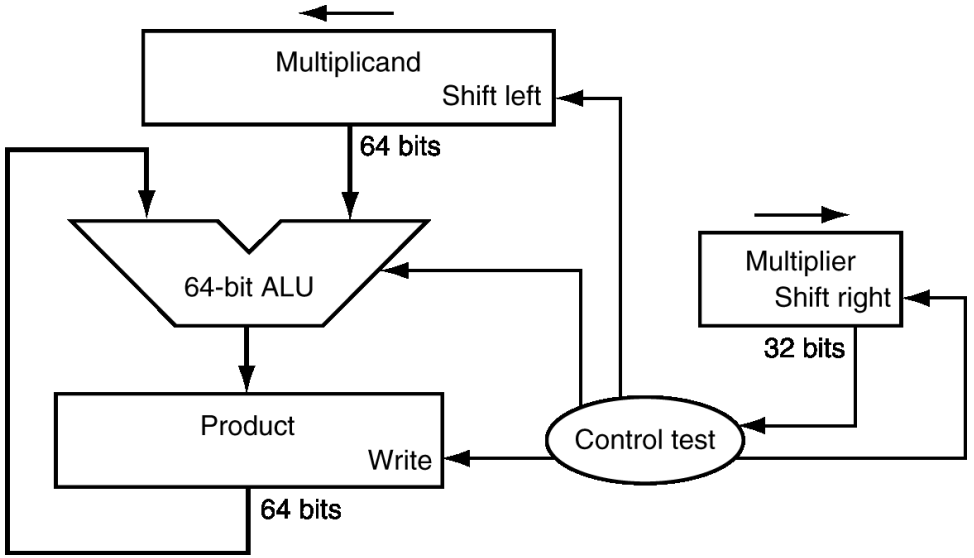
若每次运算需要3个时钟周期，则需要96个时钟 -> 比加法慢得多

缺点1：虽然在多数情况下，乘法出现的次数只有加法的1/100~1/20，但依然会对速度产生较大影响

缺点2：乘法器和寄存器的利用率较低

->需要优化！

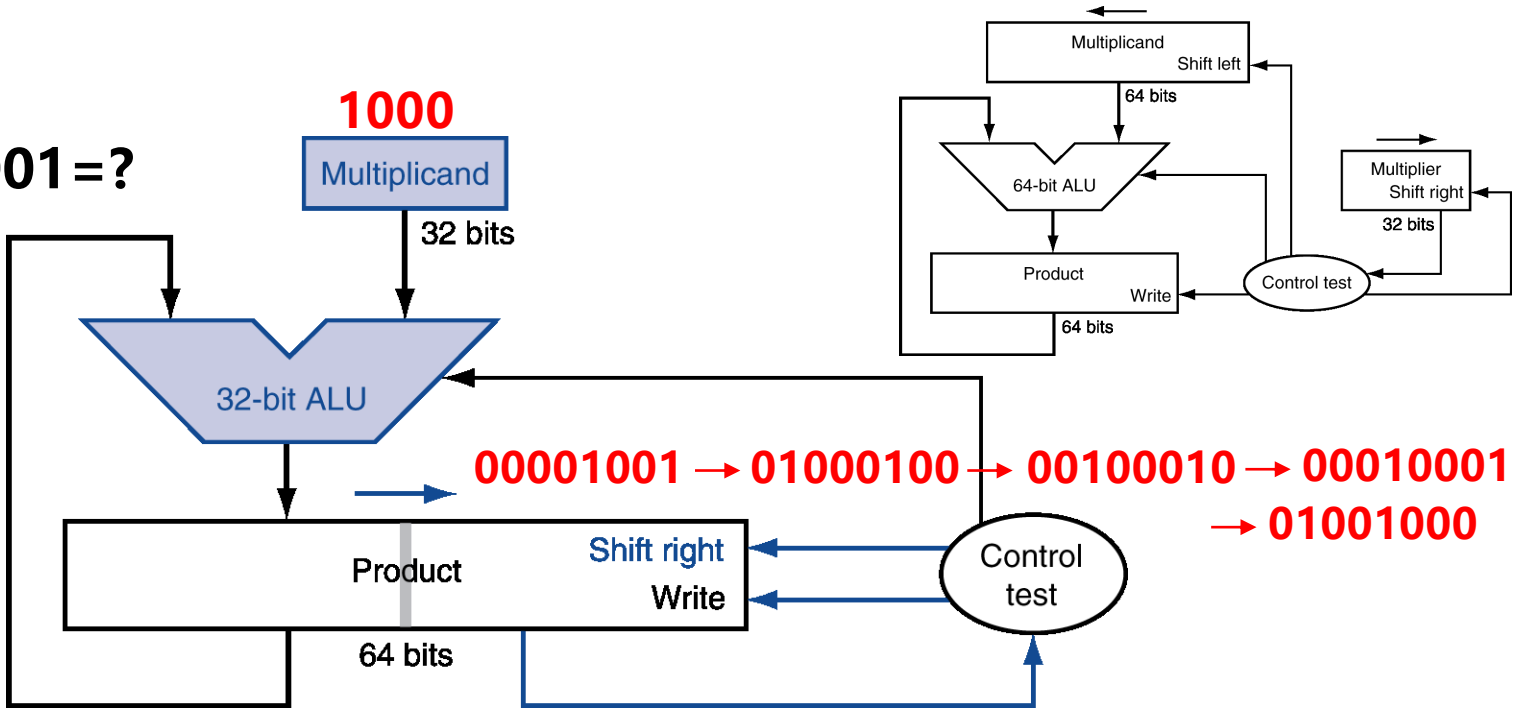
优化思路：并行，复用



2.乘法运算及其电路实现--改进的乘法器

优点：减少了32位被乘数寄存器、32位ALU、32位乘数寄存器

例如：1000*1001=?



每个周期进行一次部分乘法相加运算，同时进行位移

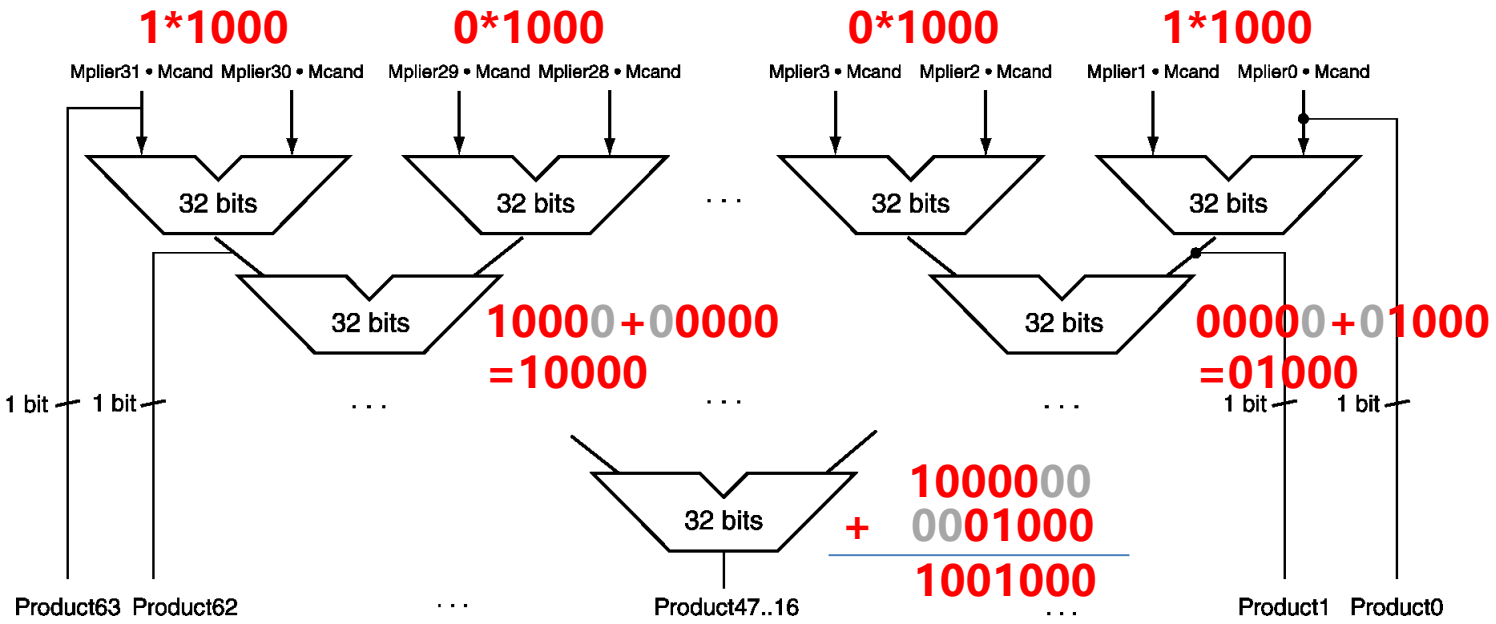
优点：每次迭代只需1个周期，因而只需32个周期即可完成运算

缺点：需要的时钟周期加长　　->当乘法的频率比较低的时候，是ok的

2.乘法运算及其电路实现--更快的乘法

并行加法树： 只需$\log_2(32)$个周期

例如：1000*1001=?



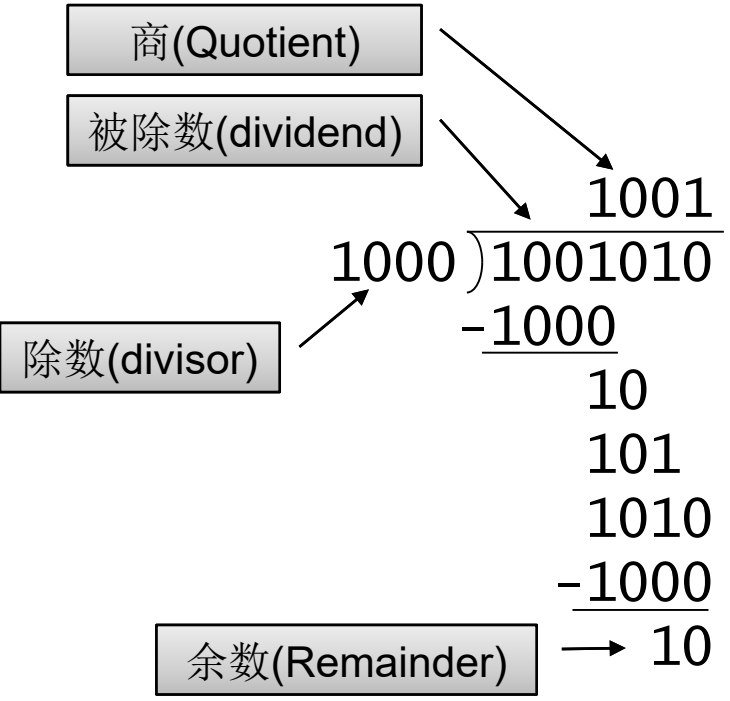
可以用流水线的方式并行执行多个乘法

缺点：硬件开销大 **->需考虑成本/性能的折中**

目录

整数运算及其电路实现

3. 除法运算及其电路实现



1. 检查除数是否为0

2. 长除法

如果除数小于等于被除数，则商中填1，相减；

否则商中填0，将被除数带下去

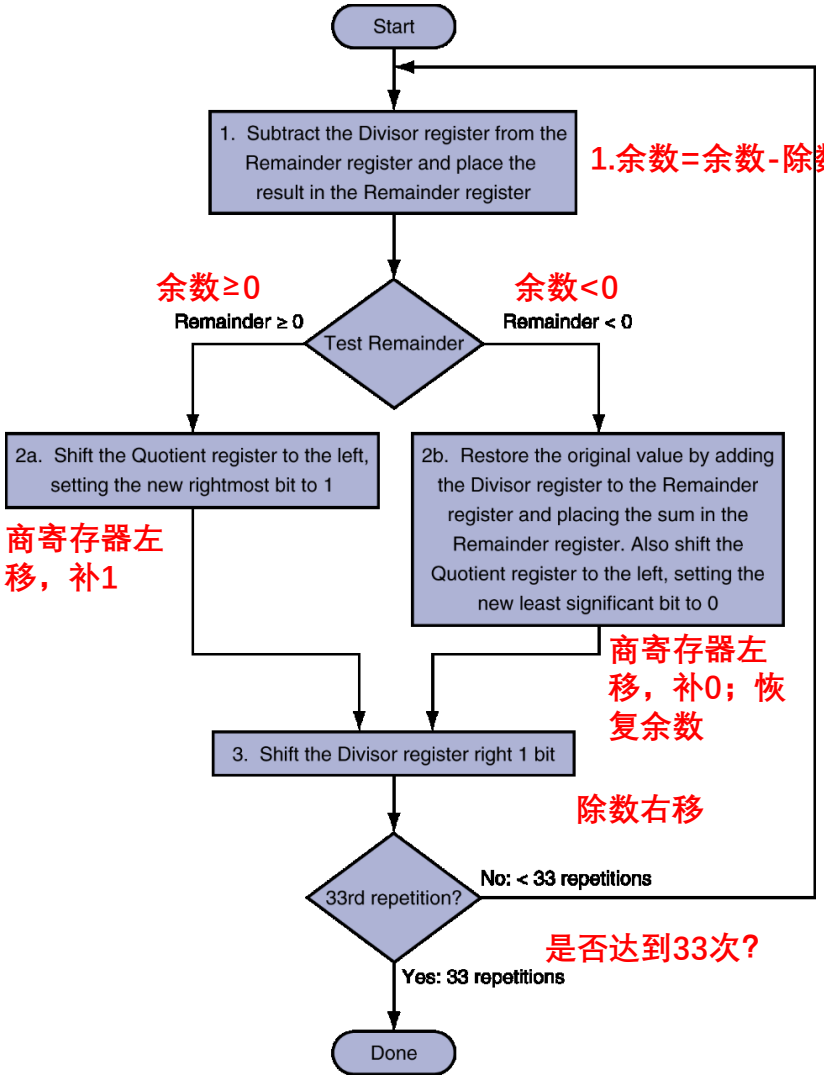
3. 撤销

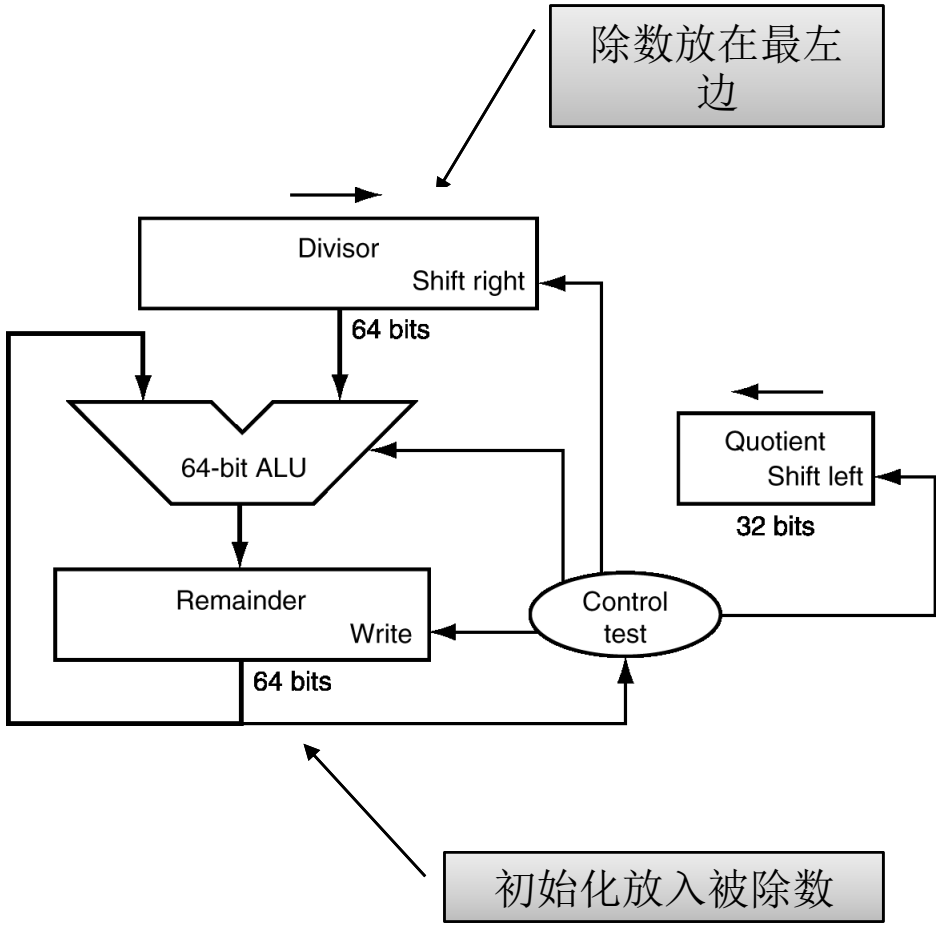
如果减法的余数小于零，则撤销

4. 有符号除法

先做绝对值除法，再调整商和余数的符号

3.除法运算及其电路实现



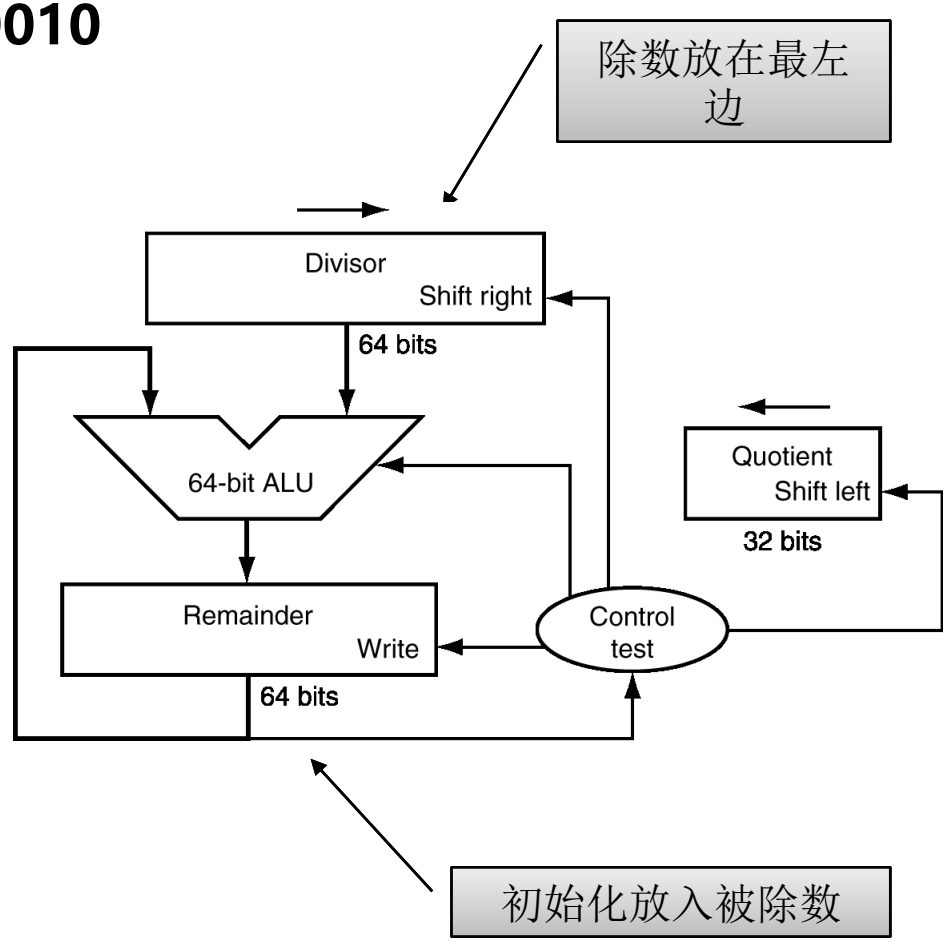


->需要33*3=99个周期完成运算

3.除法运算及其电路实现

例：01001010/1000=?　　1001+0010

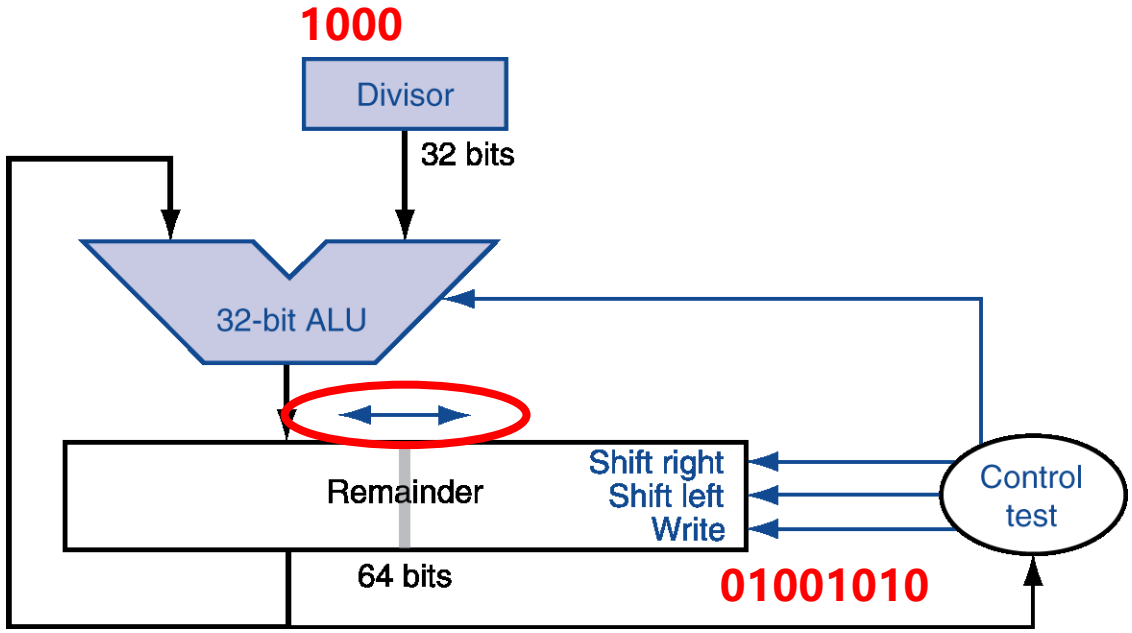
被除数：0100 1010		
除数	余数	商
1000 0000	01001010	0000
	-10000000	
	11001010	
	+10000000	
0100 0000	01001010	0000
	-01000000	
0010 0000	00001010	0001
	-00100000	
	11101010	
	+00100000	
0001 0000	00001010	0010
	-00010000	
	11111010	
	+00010000	
0000 1000	00001010	0100
	-00001000	
0000 0100	00000010	1001



缺点：硬件资源浪费、速度较慢(99个周期)

3.除法运算及其电路实现--改善的除法器

例：01001010/1000=?　　1001+0010



减少了32位除数寄存器、32位ALU、32位独立的商寄存器

每次需要1~2个周期完成运算，需要33~66个周期完成除法运算

每个周期进行一次部分余数减法

NOTE：除法器与乘法器非常类似

可以与乘法器采用同样的硬件

3.除法运算及其电路实现--不恢复除法

例：01001010/1000=?

被除数：0100 1010		
除数	余数	商
1000 0000	01001010	0000
	-10000000	
	11001010	
	+10000000	
0100 0000	01001010	0000
	-01000000	
0010 0000	00001010	0001
	-00100000	
	11101010	
	+00100000	
0001 0000	00001010	0010
	-00010000	
	11111010	
	+00010000	
0000 1000	00001010	0100
	-00001000	
0000 0100	00000010	1001

要点：b-a+a-0.5a=b-a+0.5a

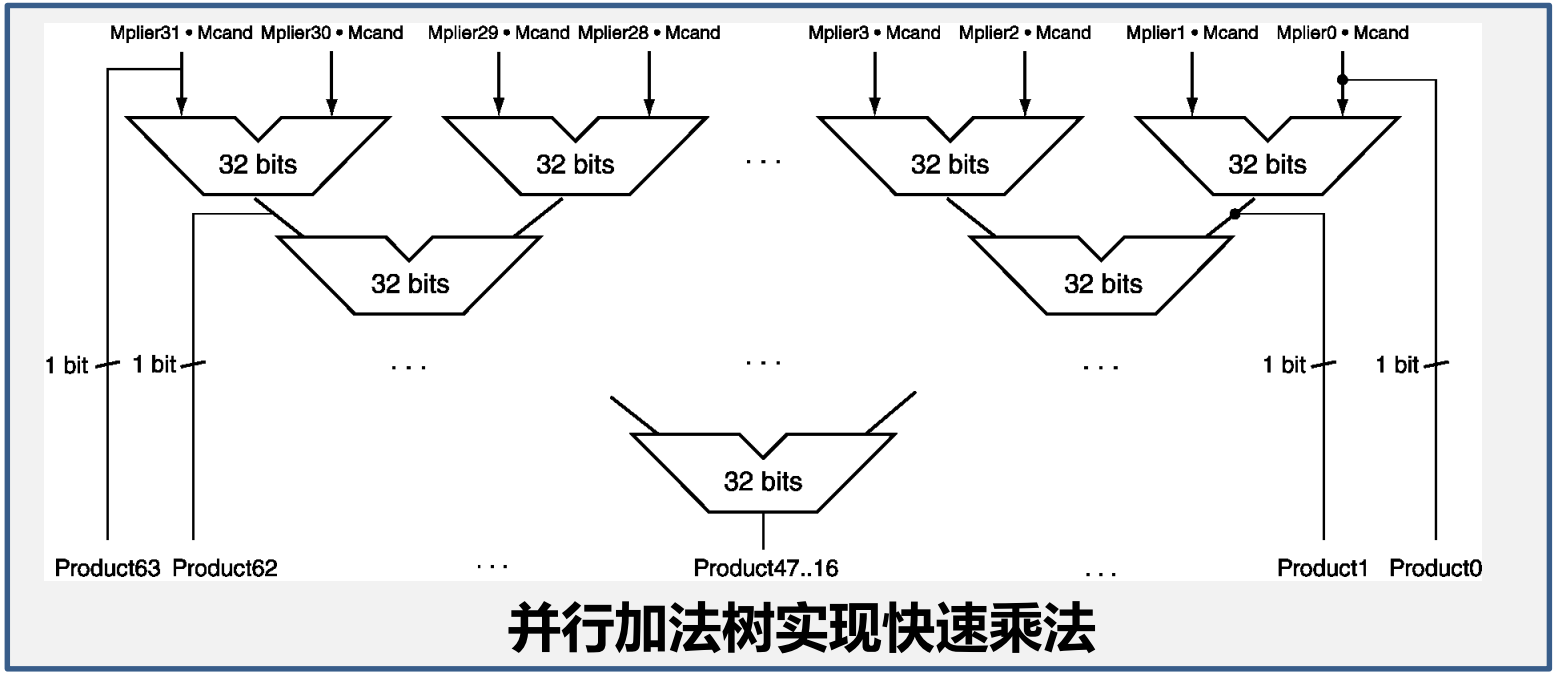
被除数：0100 1010		
除数	余数	商
1000 0000	01001010	0000
	-10000000	
0100 0000	11001010	0000
	+01000000	
0010 0000	00001010	0001
	-00100000	
0001 0000	11101010	0010
	+00010000	
0000 1000	11111010	0100
	+00001000	
0000 0100	00000010	1001

->只需要33个周期完成除法

3.除法运算及其电路实现--更快的除法

对于除法，不能采用类似乘法的并行硬件

原因：是否要进行减法取决于余数的符号



并行加法树实现快速乘法

快速除法(SRT除法)：每一步产生多个位的商（通过查找表进行猜测，后续矫正错误猜测）

->仍然需要很多步

目录

整数运算及其电路实现

1. 整数加减法运算
2. 乘法运算及其电路实现
3. 除法运算及其电路实现
4. MIPS对乘除法运算的支持

4. MIPS对乘除法运算的支持--乘法

乘积保存于两个32位的寄存器：

HI：高32位

LO：低32位

指令：

Mult rs, rt　/　multu rs, rt

->产生64位乘积

Mfhi rd　/　mflo rd

->取出HI/LO中的数据，保存至rd位置

->可以通过测试HI值来检测乘积是否超过32位

Mul rd, rs, rt

->将乘积的低32位存入rd

4. MIPS对乘除法运算的支持--除法

使用HI/LO寄存器保存结果

HI：保存32位余数

LO：保存32位商

相关指令

Div rs, rt　/　divu rs, rt

使用mfhi和mflo获取结果

没有溢出、除零检查：编程时要按需求进行检查

NOTE：MIPS中，乘法和除法共用同一组硬件

小结

整数运算及其电路实现

1. 整数加减法运算
2. 乘法运算及其电路实现
3. 除法运算及其电路实现
4. MIPS对乘除法运算的支持

思考：为什么不能用类似并行加法树的方式优化除法速度?