实验 2_2 报告

2016K8009909006
刘杰

一、实验任务（10%）

设计静态 5 级流水简单 MIPS CPU，第二阶段任务如下：

1. 新增如下 19 条机器指令：ADD、ADDI、SUB、SLTI、SLTIU、ANDI、ORI、XORI、SLLV、 SRAV、SRLV、DIV、DIVU、MULT、MULTU、MFHI、MFLO、MTHI、MTLO。

2. 考虑指令间的数据相关，采用前递处理。

3. 用 booth 两位乘和华莱士树实现乘法，用迭代算法实现除法。

4. 通过仿真和上板运行 lab2_func_2 来验证设计，提交 lab2_2 作品和报告。

二、实验设计（30%）

1. 指令前递相关数据通路设计。

为解决数据相关，采用前递技术解决，前递技术的具体实现是在流水线的运算器前通过多路选择直接把前面指令的运算输出作为后面指令的输入。下面具体介绍前递通路的选择：

运算器（包括乘除法）前的通路应该为 6 选 1，分别为正常的 ALU 通路，LO 寄存器值，HI 寄存器值，EX 级 ALU 运算结果，MEM 级 ALU 运算结果，WB 级写回值。大致设计如下图：

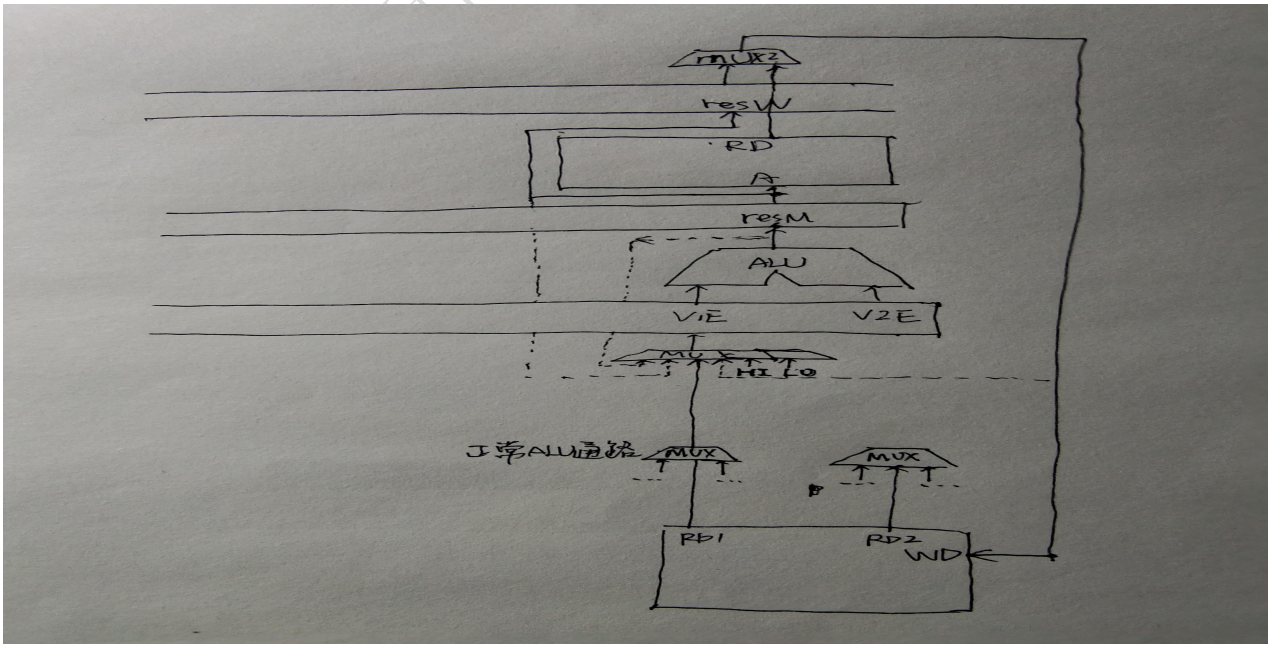

图 1

选择通路的控制需要根据 ID 级源寄存器号，将其与 EX 级写回寄存器号 destE，MEM 级写回寄存器号 destM，WB 级写回寄存器号 destW 比较。以运算器左端为例，如果 ID 级源寄存器号 RD1 为 0 或者 ID_RS 与 destW,destM,destE 都不相等，选正常 ALU 通路；当 RD1 非零且 RD1=destE，如果 EX 级不是 LW 指令，选择 EX 级 ALU 运算结果，如果 EX 级是 LW 指令，阻塞；当 RD1 非零且 RD1=destM，如果 MEM 级不是 LW 指令，选择 MEM 级 ALU 运算结果，如果 MEM 级是 LW 指令，阻塞;当 RD1 非零且 RD1=destW，选择 WB 级写回值。选择 HI 和 LO 的情况是 RD1 非零且等于 EX,MEM,WB 级的写回寄存器号，且该级的指令为从 HI/LO 寄存器搬移数据的指令。

2. 乘法器设计

乘法器设计主要分为 Booth 部分积生成逻辑，华莱士树模块设计两大块。Booth 部分积生成逻辑采用课本提供的逻辑图,这里只需将图 2 的 16×16 修改为 33×33:

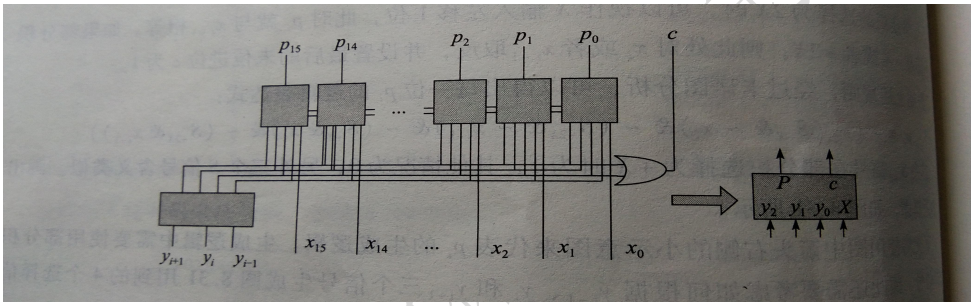

图 2

为实现有无符号乘法，须将乘数和被乘数扩展为 33 位，而 33 位乘法会产生 17 个个部分积，所以图 3 需添加一个 Booth 部分积生成模块，共计 17 个。另外被乘数 X 依旧只需要扩展到 64 位，因为最后乘积只有 64 位。把被乘数送入 Booth 部分积生成模块前，须将其左移。按下图从右至左分别左移 0，2，…，32 位。

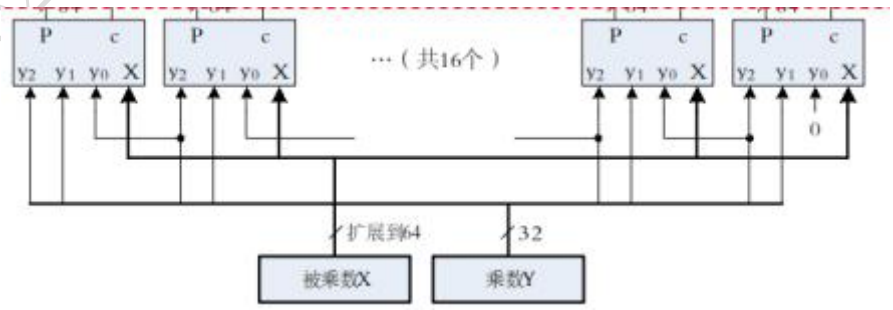

图 3

在图 4 这一步添加时钟驱动，把生成的 17 个部分积的输出用寄存器存储，来划分流水级。此外 switch 每个输出，即华莱士树模块的输入为 17 位。每个 17 位输出是 17 个部分积的相同位拼接而成。

图 4

华莱士树模块设计参考讲义，按讲义上的图连线即可，在此不做赘述。值得注意的是第一个华莱士树模块，即图 5 右起第一个模块的 14bit 的进位 Cin 应为前 14 个 booth 部分积的输出 c 拼接而成。

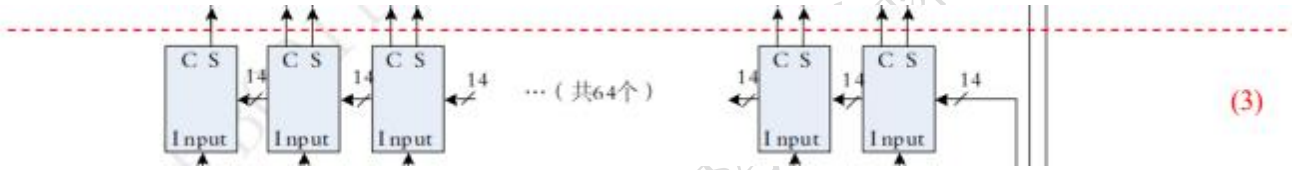

图 5

最后的乘法结果用 64 位加法实现，被加数和加数有 64 个华莱士树的输出 S 和 C 拼接而成，而 C 只有 63 个，所以在最低位拼上第 15 个 booth 生成模块的输出 c。另外 C，S 加法最终结果还需要加上第 16 个 booth 生成模块的输出 c。至此完成乘法器全部设计。

3. 除法器设计

除法器采用迭代算法，有无符号除法区别只是在迭代前后是否计算和调整商和余数的符号位。迭代前将被除数高 32 位补零，记作 div_temp，进行迭代时，除法器内部 count 累加 32 次后算出除法结果。每次迭代须将当前步骤下的 div_temp 和除数相减，结果为负则相应位商为 0；非负数则相应位商为 1，并 div_temp 的相应位更新为减法结果。当 count 最后一次累加时写入商和余数的寄存器，complete 拉高除法完成。

三、实验过程（60%）

（一）实验流水账

2018.10.4

20:00-23:00 阅读数据相关设计 pdf，安排实验步骤为先实现数据相关后自实现乘除法。

2018.10.5

13:00-14:00 在译码模块添加除了乘除法及 LOHI 寄存器相关指令外的新指令，添加数据相关控制逻辑和数据通路。

19:00-24:00 仿真，未到第一个测试点控制台报错，debug 发现第一个 bug。

2018.10.6

9:00-12:00 修改第一个 bug 后仿真，未到第一个测试点控制台报错，debug 发现第二个 bug。

14:00-15:00 修改第二个 bug 后仿真，通过除法前所有测试点。

15:00-17:00 添加乘除法指令和 HILO 寄存器相关指令，乘除法调用 IP。

19:00-23:00 仿真，在除法指令测试点报错，debug 发现第三个 bug。

2018.10.7

10:00-12:00 修改第三个 bug 后仿真，仿真通过，上板运行通过。

13:00-18:00 阅读讲义，编写除法器代码，模块验证通过后加入 cpu，仿真通过。

20:00-23:00 编写乘法器代码，模块验证未通过，发现第四个 bug。

2018.10.8

21:00-24:00 修改第四个 bug 后乘法器模块验证通过，加入 cpu 仿真通过，上板不通过，发现第五个 bug。

00:00-1:30 修改第五个 bug 后仿真上板都通过。

2018.10.9

9:00-13:00 写实验报告。

15:00-17:00 完成实验报告。

（二）错误记录

1、错误 1

（1）错误现象

控制台报错，无数据相关的指令写回寄存器的值不正确，进行不该发生数据前递的。

（2）分析定位过程

找到错误指令，其与上一条指令数据没有相关，查看前递，却发现上一条指令数据前递给了该指令，查看前递控制逻辑，发现没有错误。放大波形，发现指令寄存器返回指令不是严格的在时钟上升沿，而是有延迟。用于译码的指令是直接连线 inst_sram_rdata，所以在比较 ID 级源寄存器号和其他各级写回寄存器号时，前递正确的控制逻辑会出错。

（3）错误原因

时钟上升沿来临时，由于 inst_sram_rdata 不会立即更新，而 cpu 是直接取用 inst_sram_rdata 做译码，没有用寄存器存，所以前递的控制逻辑会出错。

（4）修正效果

把指令读地址提前一拍送进指令 sram，即 IF 级之前送进读地址，然后用 IR 寄存器存储 inst_sram_rdata。这样用于译码的指令在时钟上升沿来临就会立即更新。

（5）归纳总结（可选）

如果不放大波形，真的难发现这个 inst_sram_rdata 更新和时钟不完全同步。

2、错误 2

（1）错误现象

控制台报错，有数据相关的指令写回寄存器的值不正确，数据前递没有给对数。

（2）分析定位过程

找到错误指令，是 SW 指令，这条 SW 指令与之前的指令数据相关，但是没有把之前指令回寄存器的数据前递给 SW 指令作为 SW 指令写入 sram 的数据，查看代码，发现缺少这一部分的数据通路。

（3）错误原因

SW 指令写入 sram 的数据即使不参与运算，因为其来自寄存器堆，也会用到之前指令写回寄存器的值，所以也要考虑数据相关引发前递。

（4）修正效果

添加写入 sram 数据的前递选择通路，解决数据相关。

（5）归纳总结（可选）

考虑问题不全面，可能是做 CPU 的时候没有一次性添加好所有的数据通路，而是加一部分验证一部分的原因。

3、错误 3

（1）错误现象

控制台报错，写回寄存器号以及写回值都不对，发现取出的指令出错。

（2）分析定位过程

根据 PC 值找到指令，但发现 IR 寄存器的值不正确，查看 inst_sram_rdata 和使能信号，进一步查看发现运行除法时没有将取指使能信号拉低，多取出了一条错误的指令。

（3）错误原因

在阻塞流水线时没有拉低取指使能信号，导致取出错误指令。

（4）修正效果

在阻塞流水线的同时，将指令使能信号拉低，确保取指正确。

（5）归纳总结（可选）

要严格的控制使能信号。

4、错误 4

（1）错误现象

乘法器结果错误。

（2）分析定位过程

看教材发现被乘数要左移后才送到 booth 部分积生成模块。

（3）错误原因

讲义上没有标注，刚开始没仔细看书，所以设计时就直接把被乘数送到 booth 部分积生成模块。

（4）修正效果

被乘数送入 Booth 部分积生成模块前，将其左移，从右至左分别左移 0，2，…，32 位，修改后模块验证通过。

（5）归纳总结（可选）

看书要仔细。

4、错误 4

（1）错误现象

上板报错，数码管高低位从 25 开始不同。

（2）分析定位过程

Piazza 上有人出现相同问题，根据可能的错误原因找到华莱士树模块的端口的 input 和 output 写反。

（3）错误原因

华莱士树模块的端口的 input 和 output 写反。

（4）修正效果

更正端口，重新综合上板通过。

四、实验总结（可选）

先进行简单的数据通路设计，在自实现乘除法，循序渐进感觉很好。