**实验3报告4**

第32小组

施璠、袁峥

一、实验任务（10%）

1. 设计一款静态 5级流水简单 MIPS CPU。

2. 本次实验要求延续 lab2实验中的以下要求：

(1) CPU复位从虚拟地址 0xbfc00000处取指。

(2) CPU虚实地址转换采用：虚即是实。

(3) CPU对外访存接口为取指、数据访问分开的同步 SRAM接口。

(4) CPU只实现一个操作模式：核心模式，不要求实现其他操作模式。

(5) 不要求支持例外和中断。

(6) CPU顶层连出写回级的 debug信号，以供验证平台使用。

3. 整个实验中，最后要求实现 MIPS I指令集，除 了 ERET（非 MIPS I）、MTC0、MFC0、BREAK、SYSCALL指令，其余指令均要求实现，共 56条指令。

(1) 要求实现 MIPS架构的延迟槽技术，延迟槽不再设定为 NOP 指令，可能是任意指令。

(2) 控制相关由分支指令造成，通过延迟槽技术可以完美解决。

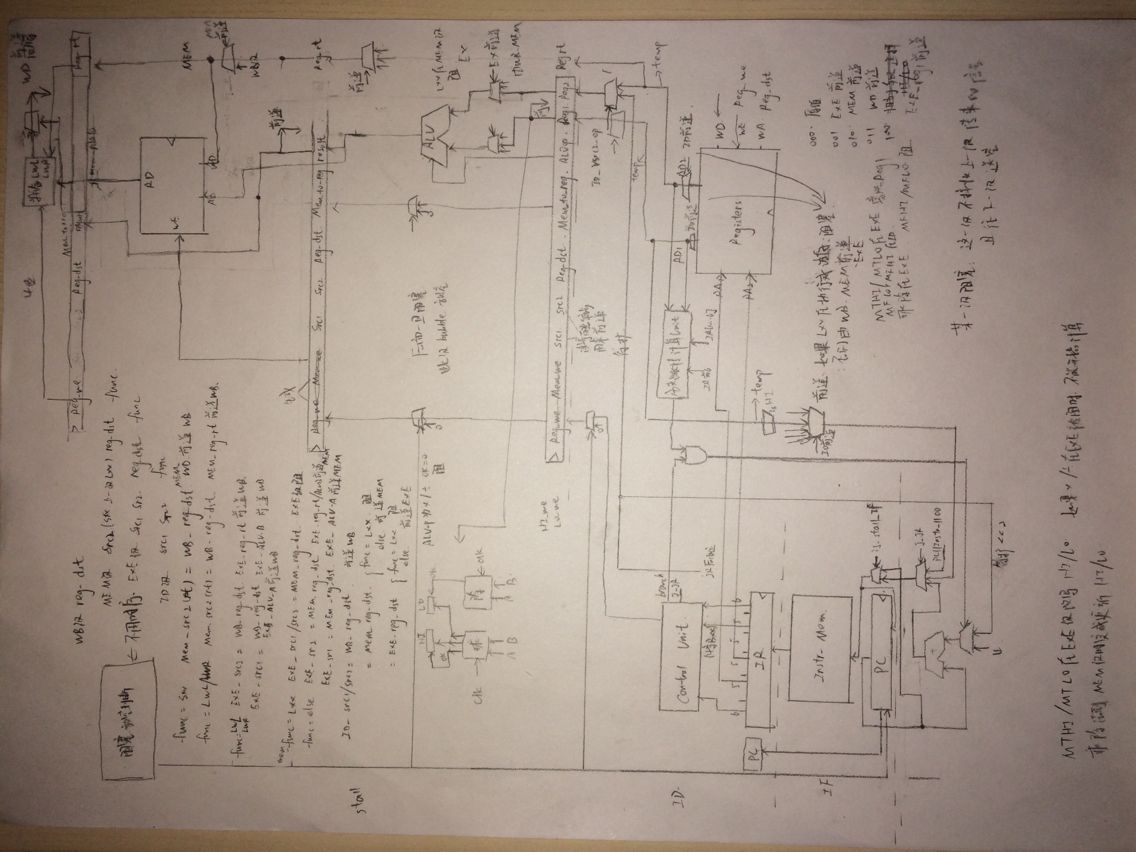
(3) 结构相关即某一级流水停顿了，会阻塞上游的流水级。

(4) 要求数据相关采用前递处理。

(5) 乘除法指令实现可以调用 Xilinx的乘除法 IP，推荐能力有余的同学自行编写乘除法器，乘法采用booth算法+华莱士、除法采用迭代算法。

二、实验设计（30%）

整个CPU设计共分为12个模块，分别为ALU、IF\_stage、ID\_stage、EXE\_stage、MEM\_stage、WB\_stage、next\_pc、regfile、stall、mycpu、divider和multiplier。其中IF\_stage、ID\_stage、EXE\_stage、MEM\_stage和WB\_stage分别为CPU五级流水的阶段，ALU模块在EXE\_stage级进行算逻运算，next\_pc产生下条指令地址，regfile为寄存器堆，stall为流水线的阻塞总调度，mycpu是整个设计的顶层，负责其他各个模块的整体调度，divider和multiplier为自行设计的乘除法器。整体设计图如下：



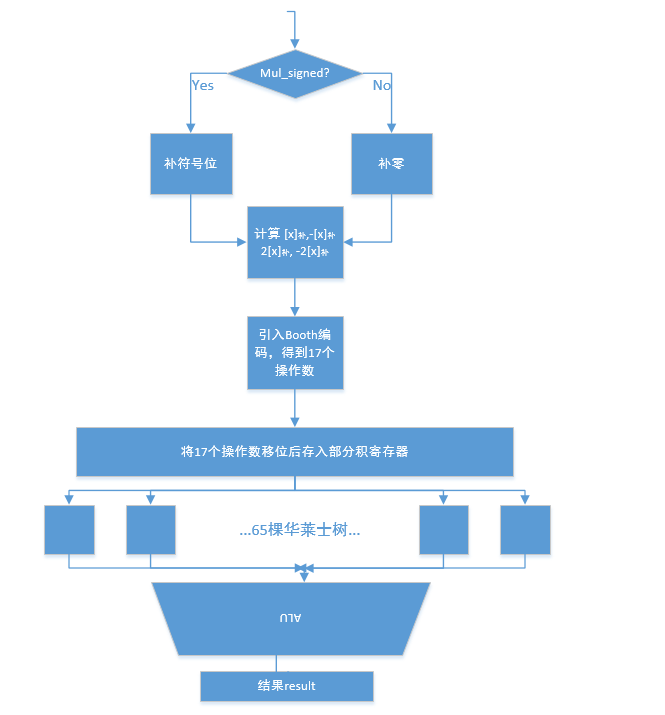
（一）乘法器模块

（1）基本概述

本周用booth编码和华莱士树实现了乘法器具体步骤如下：

1. 为了统一有符号乘法和无符号乘法，根据是否是有符号乘法，把乘数扩展为65位，有符号就补符号位，无符号就补0；
2. 计算[x]补, -[x]补, 2[x]补, -2[x]补 ；
3. 引入Booth编码，得到17个操作数；
4. 将17个操作数移位后存入部分积寄存器；
5. 得到65棵输入为17位的华莱士树；
6. 通过ALU计算得到结果。

（2）流程图



（3）接口定义

| **名称** | **方向** | **位宽** | **功能描述** |
| --- | --- | --- | --- |
| mul\_clk | IN | 1 | 时钟信号 |
| resetn | IN | 1 | 复位信号，低电平有效 |
| mul\_sign | IN | 1 | 有无符号运算信号，0为无符号运算，1为有符号运算 |
| x | IN | 32 | 乘数x |
| y | IN | 32 | 乘数y |
| valid\_in\_x | IN | 1 | 乘数x输入有效信号 |
| valid\_in\_y | IN | 1 | 乘数y输入有效信号 |
| result | OUT | 64 | 乘法运算结果输出 |
| valid\_out | OUT | 1 | 乘法运算结果输出有效信号 |

三、实验过程（60%）

（一）实验流水账

本周主要进行了乘法器的设计和性能的优化（两人大约各自花费10小时）

（二）错误记录

1、错误1

（1）错误现象

在初次尝试仿真中计算得到的结果与应该出现的结果不一样

（2）分析定位过程

考虑到testbench给出的是两组随机数的乘法，得到的数和结果都很复杂，不利于发现错误，就从2\*3出发，观察每一步的波形。

（3）错误原因

对[x]补, -[x]补, 2[x]补, -2[x]补 理解错误，输入到乘法器的接口时，x已经转换为补码形式，而在计算-[x]补时需要再计算一次补码。

（4）修正效果

在计算补码中得到了正确的加数，但还未出现正确结果，还存在别的问题。

（5）归纳总结

计算机组成原理是基础，对于不太清楚的要多查。

2、错误2

（1）错误现象

计算得到的结果和实际的结果一致，但是与testbench提供的结果不一致

（2）分析定位过程

考虑testbench的结果应该不会出错，所以应该是得到结果的时间的问题，发现应该是在第二拍出结果

（3）错误原因

没有用两拍实现乘法器

（4）修正效果

加入寄存器，使得乘法结果在第二拍给出，从而与testbench一致，在初始设定的2\*3计算中得到了正确的结果。

（5）归纳总结

要仔细阅读任务书中。

3、错误3

（1）错误现象

有符号乘法结果出错。

（2）分析定位过程

根据波形，把计算结果和实际结果相比较，发现高位丢失。

（3）错误原因

在符号位扩展的时候，只扩展了一位，应该扩展到65位

（4）修正效果

有符号乘法结果正确

（5）归纳总结

要正确理解负数在计算机中的存储方式，细致谨慎。

四、实验总结

（一）组员：袁峥

本周完成的主要工作是性能的提升和频率的测试。性能的提升方面，主要是更换了新的乘法器，之后进行测试后发现时序果然比以前短了一些。还有就是删除了一些没有用的信号，这主要是之前在对代码进行修改的时候没有修改干净，留下了一些残余，这样修改之后可能会对布局布线进行一些提升。在进行第四阶段的指令测试时比较顺利，一遍就完成了测试。接着运行性能测试程序，由于需要尽可能的提高频率，因此此处采用二分法，先测得一个跑不了的频率，然后用二分法不断逼近，测得极限频率，最后发现大约在130MHz左右。

（二）组员：施璠

在本阶段中的实验中，我总算改完了上周没改好的乘法器，通过这部分的实验，对补码的定义，booth编码，以及华莱士树有了更深刻的理解。上周没把乘法器写完的原因之一大概是第一遍写的有点草率，然后bug就很多，看着波形就头大，不知从何改起。后来选择从自定义数据开始测起，先测小数据的无符号乘法，再测随机数的无符号乘法，然后测小数据的有符号乘法，最后测所有随机数的乘法。这样分功能调试更有利于逐步跟踪，发现错误。反思bug多的原因还是概念不清又不够仔细。

五、实验结果

|  |  |
| --- | --- |
| Dhrystone运行实际total\_ns | 42300 |
| Coremark运行实际total\_ns | 3435750 |
| myCPU上板最高可运行频率 | 132MHz |
| Coremark/MHz | 2240/1000 |
| DMIPS/MHz | 1043/1000 |