**Lab 5报告**

学号：2020K8009929034、2020K8009929019、2020K8009929037

姓名：胡康、李子恒、吕星宇

箱子号：05

一、实验任务（10%）

在lab4的基础上，添加乘除法指令，并完成可以配套操作的访存指令（ld和st）。此外，完成更多的转移指令，使CPU的指令增加到46条指令。

二、实验设计（40%）

（一）总体设计思路

1. 在译码阶段，复用decoder完成指令的相应译码，并复用部分指令的数据通路和控制通路。

2. 对于乘除法，调用IP核来完成相应的操作。

3. 新增的四条跳转指令功能与已有的beq、bne非常相似，只是跳转的判断条件不同，因此设计上整体复用原有的数据通路，在其基础上增加对应的判断条件。

4. 新增的六条访存指令整体数据通路亦可复用原有的ld.w和st.w指令，并在此基础上修改数据的选择和加载信号，从而产生正确结果。

（二）重要模块1设计：除法器（以无符号为例）

定制IP核完成无符号除法，我们的工作主要是完成握手信号的连接。

1. 工作原理

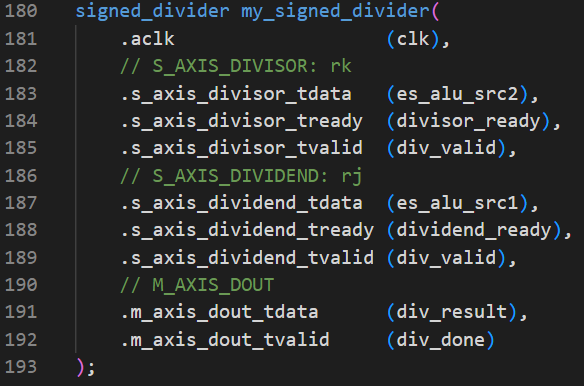


图1：除法器实例化

执行s\_axis\_dividend\_tdata/s\_axis\_divisor\_tdata，将商放在m\_axis\_dout\_tdata的高位，余数放在低位。

1. 接口定义

这一部分的核心是ready，valid和done三种信号的握手，其余信号省略。

当时钟上升沿来临时，将valid置为1，发起握手请求；随后ready信号也会拉高（周期性拉高），在下一个时钟上升沿时，两者均为高则完成握手，data传入IP核，进行运算。

最后，当div\_done拉高时，表明计算结束，阻塞放开，流水线启动。

1. 功能描述

调用IP核完成除法。

1. 重要模块2设计：跳转指令控制信号
2. 工作原理

跳转指令的控制信号如图2所示：

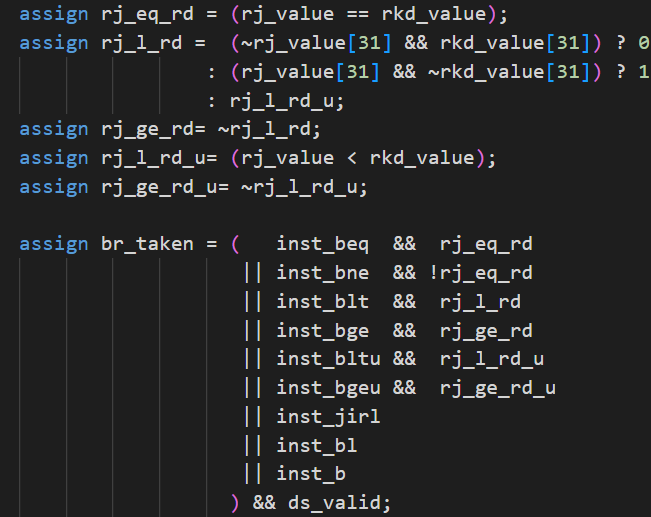


图2：跳转指令控制信号

本实验新增了四条跳转指令，跳转逻辑分别是无符号小于、无符号大于等于、有符号小于、有符号大于等于。显然只需实现两个小于的判断，再对结果取反即可获得两个大于等于的取值。而有符号小于与无符号小于的比较方式也有所不同；小于号<可以直接对两个数进行无符号比较，但对于有符号数则无法这样直接比较。一种方法是采用signed标识符，另一种是考虑两个数字的正负分情况讨论。稳妥起见，这里采用后者实现，得到了四个新增信号的赋值逻辑，进一步更改br\_taken和br\_target的逻辑即可实现相应跳转。

1. 功能描述

通过跳转控制信号判断当前指令是否跳转，完成四条跳转指令的新增。

1. 重要模块3设计：访存指令相关计算

1、工作原理

load类型的指令在从数据RAM得到结果后，需要根据指令的类型和访存地址的低二位确定要存回寄存器堆的数据。为此，新增ld\_vaddr信号指示访存地址的后二位，同时生成ld.b，ld.bu，ld.h，ld.hu，ld.w的结果，再通过指令控制信号进行多路选择，将相应的结果存入mem\_result中。该过程如下图所示：

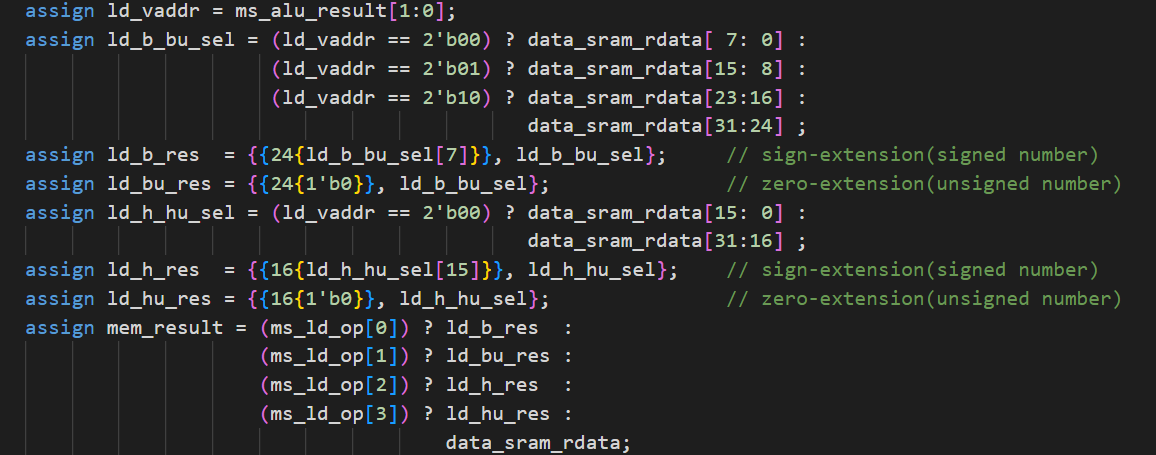


图3：ld指令相关计算过程

store类型的指令在从寄存器堆得到结果后，需要根据指令的类型和访存地址的低二位确定存入数据RAM的内容。为此，新增st\_vaddr信号指示访存地址的后二位，同时生成mem\_write\_strb信号，再根据mem\_write\_strb信号截取相应的字节，将结果存入mem\_write\_data中。该过程如下图所示：

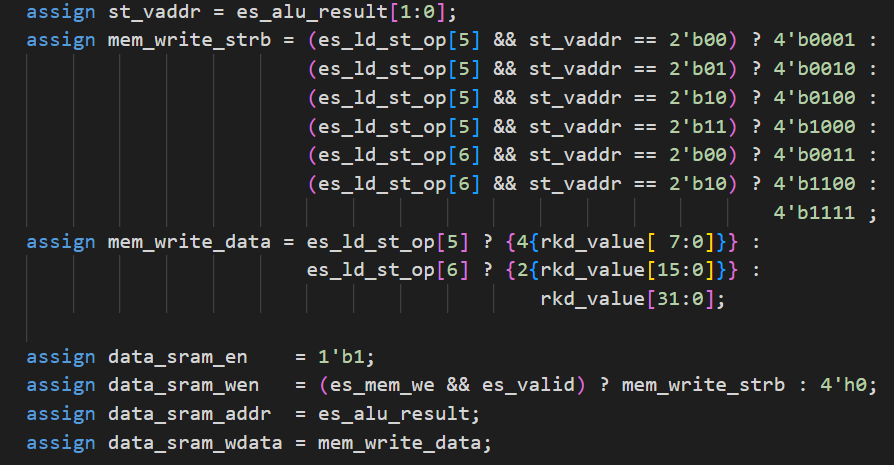


图4：st指令相关计算过程

2、功能描述

通过访存指令具体类型和访存地址低二位计算要存回的数据，然后整体复用之前的数据通路，完成数据写回。

3、注意事项

load类型的指令在访存阶段计算相应的结果，store类型的指令则要在执行阶段计算出结果，因为store类型的指令需要在访存阶段向RAM中写入数据，这意味着计算结果需要提前一个周期拿到。

三、实验过程（50%）

（一）实验流水账

2022.9.30 19：00——23：00 完成exp10；

2022.10.6 20：30——21：30 实现新增的访存指令；

2022.10.7 21：00——23：00 实现新增的跳转指令；

2022.10.8 14：00——16：00 完成exp10部分实验报告的书写；

2022.10.9 20：30——22：00 完成exp11跳转指令部分实验报告的书写；

2022.10.11 8：30——9：30 完成exp11访存指令部分实验报告的书写。

（二）错误记录

1、错误1：除法器ip核的时序错误

（1）错误现象

div\_valid未能正确拉高，产生错误。波形会一直前进，无法停下。

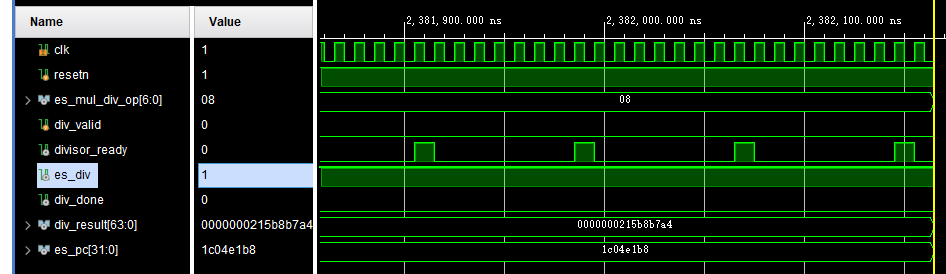


图5：错误1波形截图

（2）分析定位过程

分析div\_valid的来源，是时序的问题，于是开始用纸笔摆弄时序，发现错误原因。

（3）错误原因

分析div\_valid的来源，发现是来源于bus\_r（es\_mul\_div\_op），这需要一次clk才能将bus中的值传入bus\_r，这本是正常的一步。但是由于div\_valid是由时序控制的，其进入条件为(ds\_to\_es\_valid && es\_allowin)，于是会发生下述问题：

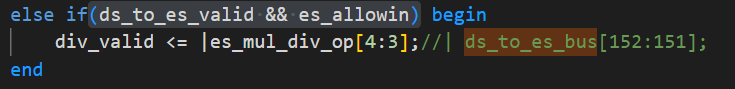


图6：错误1出错代码

在时钟上升沿来临时，ID阶段的div指令将进入EXE阶段，此时，es\_ready\_go的逻辑是组合的，于是在clk后，es\_mul\_div\_op的值成功从bus\_r中读了出来，但是此时es\_allowin也同时被拉低，于是div\_valid的值无法发生变化，不能拉高，最终导致错误。

（4）修正效果

将其直接连上ds\_to\_es\_bus即可，这样便可以解决时序的问题。

（5）归纳总结

对于axis总线标准以及时序跳变不清晰。

2、错误2：访存指令的控制信号没有更新

（1）错误现象

新增load类型指令写回阶段写回的数据不正确。

（2）分析定位过程

检查load类型指令计算结果，发现mem\_result计算正确，说明数据计算模块设计未出现问题。但mem\_final\_result信号却传回错误。检查波形，发现mem\_final\_result选取了ms\_alu\_result而不是mem\_result作为要写回的数据。这说明错误发生在控制信号ms\_res\_from\_mem中。

沿流水线追溯，找到ID阶段相关控制信号的定义上，发现定义中只考虑了inst\_ld\_w信号，而没有考虑新增的四个load类型信号。

（3）错误原因

只增加了访存信号结果计算模块，而没有相应地修改ID阶段的各个控制信号。

（4）修正效果

增加其它四个信号，并相应地修改其它与load和store信号有关的控制信号后，结果正确，并通过所有检查点。

四、实验总结

小组合作要搞好分工，另外还要理解好小组使用的代码。