实验 5-3 报告

学号 2016K8009915009 2016K8009937003
姓名　　钟　赟　　吴　双

一、实验任务（10%）

(1) CPU 顶层修改为AXI 接口。CPU 对外只有一个AXI 接口，内部完成取指和数据访问的仲裁。
(2) 将CPU集成到SoC_AXI_Lite 系统中。
(3) 完成固定延迟的功能测试。

二、实验设计（30%）

（一）SRAM-AXI 转换桥设计

转换桥中使用状态机，通过仲裁器从类 SRAM 接口接收信号和向其返回结果。类 sram 接口数据端的读与写操作按顺序执行, 在一个操作完成操作之前不会进行下一个；AXI 总线每次只执行一条读操作或写操作, 在一条读操作完成之前不进行其他的操作。修改了转换桥中的 data_size，实现非对齐访存。

状态机设计：状态机采用经典的三段式结构，为了减少代码量，在三个 always 模块中，每一段状态机模块中包含四个部分：类 SRAM 接口的指令通道、类 SRAM 接口的数据通道、AXI 接口的读通道、AXI 接口的写通道。

仲裁逻辑设计：仲裁逻辑负责处理多个读操作发生的情况。在指令和数据产生冲突时，如果先读指令，那么可能会因为等待数据而阻塞流水线，降低效率，因此我们采用读数据的优先级高于读指令。仲裁逻辑依靠一个记录当前请求类型的寄存器 req_n 来实现。

（二）修改 CPU 顶层接口

myCPU 顶层修改为对外 axi 总线接口和 debug 接口，包括有两个模块：axi_ifc 和 cpu_core。axi_ifc 为转换桥，cpu_core 来自之前的流水线 cpu, 修改其端口为类 sram 接口。

本次实验主要内容是如何设计各流水级之间的握手信号逻辑，以及对于两个类 sram 端口 会出现的各种情况的考虑。以 inst_sram_like 端口为例，对一个 pc 值对应的指令，其可能的状态有：

发送请求：inst_req && !inst_addr_ok

等待数据返回： !inst_data_ok （注意这个状态下指令可能处于 if 阶段也可能处于 de 阶段）

数据返回：inst_data_ok （此状态下指令可能出于 if 阶段也可能处于 de 阶段）

在发送请求时，inst_req 何时赋值为 1 需要仔细考虑。在复位时，inst_req 需要保持为 0；在报出例外的时刻 inst_req 也需要为 0，防止进入例外处理程序部分时还收到报出例外前的数据。另外如果因为阻塞导致 inst_data_ok 为 1 时，指令还未进入 de 级，之后 if 级就不能再发送 inst_req 直到指令进入 id 级，防止重复取同一指令。在这一情况下还需要一个特定的寄存器存储提前返回的数据。

三、实验过程（60%）

（一）实验流水账

2018-12-08　09:00-13:00　阅读讲义，编写代码。

2018-12-08　17:00-26:00　debug，仿真终于通过第一个测试点。

2018-12-10　18:00-25:00　debug 结束，FPGA 上板通过。

2018-12-11　14:00-17:00　修改代码风格，撰写实验报告。

（二）错误记录

1、错误 1

（1）错误现象

某信号修改几番后仍为不定态 X。

（2）分析定位过程

跟踪发现产生该信号的所有信号均为确定态，仔细排查，发现是多驱动的原因。

（3）错误原因

驱动该信号的语句有两句，产生多驱动。

（4）修正效果

删除该信号的其中一句驱动语句，修正后该信号变为确定态。

2、错误 2

（1）错误现象

跳转指令处未发生跳转。

（2）分析定位过程

查看 test.s 发现该指令为 b 指令，没有数据相关，进一步查看发现 next_pc 只保留了一拍。

（3）错误原因

Decode 阶段的指令直接使用 inst_rdata。

（4）修正效果

修改 instrutcion 的相关逻辑，增加寄存器存放返回指令。

3、错误 3

（1）错误现象

执行 lw 指令，数据返回后流水线无法向下一级流动。

（2）分析定位过程

查看波形发现这一时刻当 data_addr_ok 返回之后，mem 级的指令应该流向下一级，但是此时 mem_readygo 为 0。

（3）错误原因

Mem_readygo 原先的判断逻辑中，只考虑了最理想的情况：发出 data_req 并且收到 data_addr_ok。由于转

换桥 并不是同步发送请求的，可能在 wb 可以允许进入的时候，addr_ok 早已到达，因此导致错误的 mem_readygo 赋值。

（4）修正效果

增加状态寄存器 state_wait_data，表示当 mem 阻塞时 addr_ok 已经到达，mem_readygo 的判断增加选择语句 (state_wait_data) ? 1'd1 :……。

4、错误 4

（Pc 和地址相同向寄存器堆写入错误的值。

（2）分析定位过程

向前查看发现出现数据相关，在数据到达 wb 级之前，表示数据相关的信号已经变为 0，判断是数据相关的信号出错。

（3）错误原因

之前的数据相关逻辑默认的是只停止一拍，并且在 wb 级和其他级出现相关时全部采用前递，当数据在好几拍之后才返回时就会出错。

（4）修正效果

修改数据相关的控制逻辑，在 wb 出现需要使用 data_rdata 的情况时，应该阻塞到数据返回（或者数据已经返回过）；mem 阶段 addr_ok 拉高时，hazard_mem 才能变为 0。

5、错误 5

（1）错误现象

除法指令执行出错，没有完成除法的执行。

（2）分析定位过程

定位到除法指令开始执行的时刻，发现除法指令执行时 EX 级流水线没有发生阻塞。

（3）错误原因

流水线控制逻辑不完善，没有添加除法指令的阻塞信号。

（4）修正效果

完善 EX 级流水线控制信号，修正后，该处不再报错。

四、实验总结（可选）

本次实验的难度较大，尤其是对于 inst 类 sram 通道和 data 类 sram 通道的状态转换，需要考虑非常多的情况，和实验 2 一样需要突破自己之前的思考模式，debug 和理解的过程相当痛苦……