实验 4_1 报告

2016K8009909006
刘杰

一、实验任务（10%）

为 myCPU 增加例外与中断支持，完成功能测试，并支持运行一定的应用程序。 需要完成如下任务：

(1) 增加 MTC0、MFC0、ERET、SYSCALL 指令,实现系统调用例外支持。

(2) 增加 CP0 寄存器 STATUS、CAUSE、EPC。

(4) 运行 69 个功能点测试通过。

二、实验设计（30%）

（一）CP0 寄存器模块设计

类似通用寄存器堆的设计，增加支持例外和中断的 CP0 控制寄存器模块，模块主要信号如下表：

表 1：CP0 寄存器模块信号表

信号名	信号方向	信号位宽	信号描述
clk	input	1	时钟信号
resetn	input	1	复位信号，低电平复位
cp0_raddr	input	5	CP0 寄存器读地址
cp0_waddr	input	5	CP0 寄存器写地址
cp0_wdata	input	32	写入 CP0 寄存器数据
cp0_rdata	output	32	读出 CP0 寄存器数据
mtc0_wen	input	1	写使能信号，高电平才能写 CP0 寄存器
exception_cmt	input	1	例外提交信号，高电平有效
eret_cmt	input	1	例外返回信号，高电平有效
inst_in_ds	input	1	高电平代表例外指令在延迟槽
inst_pc	input	32	例外指令 PC
eret_pc	output	32	例外返回 PC

CP0 寄存器寄存器除了在执行 mtc0 和 mfc0 指令进行读写，在例外处理时也需要相应的读写操作。对于 STATUS 寄存器，在例外提交时，需要将 exl 位域置 1，在执行 eret 指令时将 exl 位域值 0 来标记例外处理过程。对于 CAUSE 寄存器，在例外提交时根据例外向量表改变 exccode 位域来表示例外类型，如 syscall 例外，exccode

位域置为 0x08，同时根据 inst_in_ds 信号判断例外指令是否在延迟槽改变 bd 位域。对于 EPC 寄存器，在例外提交且未被屏蔽时，将例外指令 PC 存入 EPC 寄存器，并在执行 eret 指令时，将 EPC 寄存器的值读出通过 eret_pc 端口输出。

为优化例外处理的电路，故所有例外在统一的流水级处理（暂定为写回级），即在例外发生后，先传递到写回级，在写回级提交例外，进行处理。

三、实验过程（60%）

（一）实验流水账

2018.11.17

9:00-11:00 阅读讲义和指令手册，设计电路并编写代码。

2018.11.17

11:20-12:00 仿真，在 SYSCALL 指令处发现 bug，修复 bug。

13:25-14:00 修复 bug 后仿真，在 SYSCALL 指令处再次发现 bug，修复 bug。

16:00-17:00 修复 bug 后仿真通过，上板通过.

2018.10.18

18:00-20:00 完成实验报告。

（二）错误记录

1、错误 1

（1）错误现象

syscall 例外发生之后，在处理该例外的第一条指令 PC 值发生错误

（2）分析定位过程

发现 PC 值与 trace 不匹配，错误 PC 值属于前一条指令 PC+4，但是正确的 PC 值是 0xbfc00380，查阅手册后发现例外处理入口为 0xbfc00380。

（3）错误原因

没有增加一条例外提交时 PC 来源的选择通路，使得提交例外后 PC 没有自动跳转到例外处理入口 0xbfc00380，从而发生错误。

（4）修正效果

添加例外提交时 PC 来源是 0xbfc00380 的选择通路，再次仿真通过。

（5）归纳总结（可选）

例外发生时由硬件跳转到确定的例外处理入口，这一过程不是软件做的。

2、错误 2

（1）错误现象

控制台报错，mfhi 指令取出的值错误。

（2）分析定位过程

mfhi 指令执行过程没有问题，HI 寄存器的值有误，往回查找最近一条写 HI 寄存器的指令，发现是 divu 指令后的一条 mthi 指令，在 mthi 没有将除法结果写入 HI 寄存器时，HI 寄存器的结果符合 trace 的比对结果。

（3）错误原因

除法指令是紧跟着 syscall 指令，syscall 指令之后的指令都不应该执行。而我实现的时候忘记清空 syscall 指令之后的流水线。

（4）修正效果

把 syscall 指令后的指令相关控制信号都设置为无效，确保 syscall 指令后的指令都未执行。

（5）归纳总结（可选）

例外发生时要记得清空例外指令后流水线。

四、实验总结（可选）

例外处理是在做添砖加瓦的工作，改动幅度不大。