实验 2 - 流水线异常和中断设计

浙江大学计算机体系结构实验

DDL: 2022.10.26 23:59

1 实验目的

- 了解 RISC-V 简单的异常和中断
- 了解如何在流水线中添加异常和中断机制

2 实验环境

- **HDL**：Verilog、SystemVerilog、Chisel
- **IDE**：Vivado
- **开发板**：NEXYS A7 (XC7A100T-1CSG324C) 或 Sword 4.0 (XC7K325T-2FFG676)

3 实验原理

3.1 特权级

RISC-V 有三种特权级：M mode, S mode, U mode。设置不同的特权级可以用来管理系统资源的使用，提供软件的保护。其中，M mode 的级别最高，是 RISC-V 硬件必须实现的，在 M mode 下跑的代码都被认为是安全可信的。U mode 级别最低，在该模式下执行用户程序，对应于操作系统的用户态。S mode 介于 M 和 U 之间，该模式下执行操作系统内核程序，对应于操作系统的内核态。**特权级的编码如表1所示**，本次实验仅考虑其中的 M mode。

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	*Reserved*	
3	11	Machine	M

表 1: RISC-V privilege levels.

3.2 CSR

CSR(Control and Status Registers) 是用来记录 CPU 中一些重要信息的寄存器，可以把它简单地看作特权寄存器。我们要实现的 CSR 有 mstatus, mtvec, mepc, mcause, mtval。这些寄存器都属于前述的 M mode，所以它们以字母 m 开头，只有 M mode 下可被访问。各 CSR 的定义、编号、权限等具体信息请参考Privileged Spec。

3.3 异常和中断

RISC-V 手册中把异常 (exception) 定义为与某条指令关联的运行时发生的事件，把中断 (interrupt) 定义为会产生 unexpected 的控制流转移的外部异步事件。使用 trap 来指代 exception 或者 interrupt。

我们需要实现的三种 exception 如下：

- 访问错误异常：当物理内存的地址不支持访问类型时发生，（例如尝试写入 ROM）
- 环境调用异常：执行 ecall 指令时发生
- 非法指令异常：在译码阶段发现无效操作码时发生

发生异常/中断时，硬件自动经历如下的状态转换：

- 异常指令的 PC 被保存在 mepc 中，PC 被设置为 mtvec。对于异常，mepc 指向导致异常的指令；对于中断，它指向中断处理后应该**恢复执行的位置**。
- 根据异常来源设置 mcause，并将 mtval 设置为出错的地址或者其它适用于特定异常的信息字。
- 把控制状态寄存器 mstatus 中的 MIE 位置零以**禁用中断**，并把先前的 MIE 值保留到 MPIE 位中。
- 将发生异常之前的权限模式保留在 mstatus 的 MPP 域中，再把权限模式更改为 M。

完成 trap 的处理后，软件通常执行 mret 以恢复至正常执行流，mret 的主要操作有：

- 从 mepc 取值以恢复 PC
- 更新权限模式，修改 mstatus 的 MIE、MPIE、MPP 等域

3.4 实现指令

需要实现的 CSR 相关指令如表2所示。具体的指令含义请参考Unprivileged Spec和Privileged Spec。

csr		rs1	001	rd	1110011	CSRRW
csr		rs1	010	rd	1110011	CSRRS
csr		rs1	011	rd	1110011	CSRRC
csr		uimm	101	rd	1110011	CSRRWI
csr		uimm	110	rd	1110011	CSRRSI
csr		uimm	111	rd	1110011	CSRRCI
000000000000		00000	000	00000	1110011	ECALL
0011000	00010	00000	000	00000	1110011	MRET

表 2: 实现的指令

3.5 参考设计

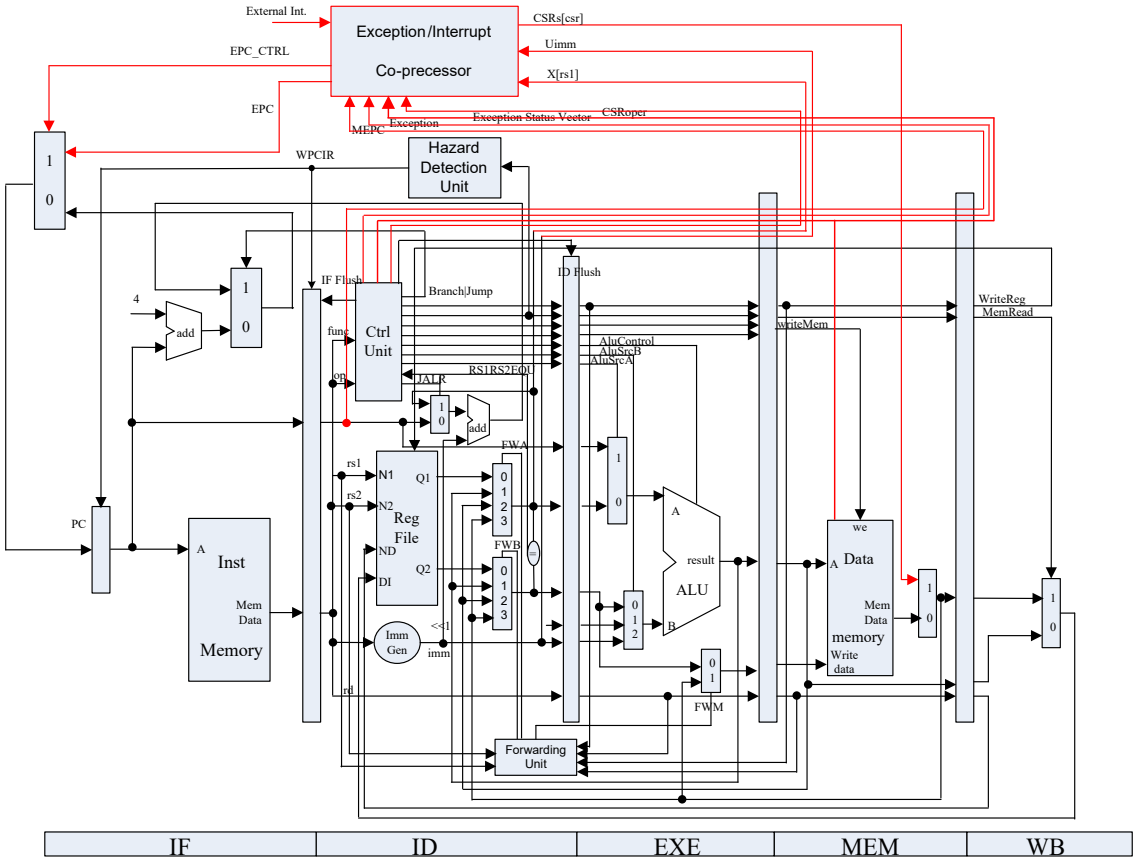

图 1: 异常实现参考图

4 实验要求

1. 实现 csrrw, csrrs, csrrc, csrrwi, csrrsi, csrrci, ecall, mret 指令
2. 实现 mstatus, mtvec, mepc, mcause, mtval 寄存器
3. 实现3.3节中列出的三种异常和**外部中断**，需要实现精确异常
4. 通过仿真测试和上板验证

5 实验步骤

1. 根据 RISC-V 非特权级手册和特权级手册在流水线内加入异常和中断机制
2. 在给定的 SoC 中，加入自己的 CPU，通过仿真测试和上板验证

6 注意事项

1. 本次实验仅需实现 RISC-V 的 **M mode**，因此不必考虑对 CSR 的错误操作，例如越级的读写
2. RISC-V 没有明确指定存放 CPU 当前特权级的寄存器，所以你可以自行定义（虽然目前只有一个特权级）
3. 简单起见，处理异常时 mtval 的写入值可以只考虑两种情况
   - 访问错误异常时，写入错误的地址
   - 非法指令异常时，写入错误的指令
4. 本次实验主要在 ExceptionUnit.v 实现，但也需要注意其他模块：
   - 确定不同异常的触发位置，适当添加新的信号，以得到 mtval 需要的信息
   - CSR 的基本读写操作已经在 CSRRegs.v 实现，你可以直接使用这个模块，也可以修改它以在处理异常时更高效地实现 CSR 读取/修改
5. 目录 lab2_ref 提供了测试指令和数据的相关信息以及仿真结果的参考
   - 其中如 csrr，csrw，csrs，csrc 这样的指令为伪指令，不需要在硬件实现，其定义见 RISC-V 非特权手册的 pseudoinstructions 表
6. 上板前记得检查 FPGA 型号是否正确，小板子只要是 xc7a100tcsg324 都行

7 思考题

1. 精确异常和非精确异常的区别是什么?

2. 阅读测试代码，第一次导致 trap 的指令是哪条？trap 之后的指令做了什么？如果实现了 U mode，并以 U mode 从头开始执行测试指令，会出现什么新的异常?

3. 为什么异常要传到最后一段即 WB 段后，才送入异常处理模块？可不可以一旦在某一段流水线发现了异常就送入异常处理模块，如果可以请说明异常处理模块应该如何处理异常；如果不可以，请说明理由。

注：思考题写入实验报告内