

中国科学院大学

《计算机体系结构基础(研讨课)》实验报告

姓名 艾华春,李霄宇,王敬华

学号 2022K8009916011,2022K8009929029,2022K8009925009

实验项目编号 4 实验名称 异常和中断的支持

一、 逻辑电路结构与仿真波形的截图及说明

• 添加系统调用异常支持。

1. 添加控制寄存器 CRMD, PRMD, ESTAT, ERA, EENTRY, SAVE0-3

新建一个 csrfile 的模块, 里面包括所有控制寄存器的实现, 并在mycpu_top中进行例化, 与各个流水级处于并列地位。

模块接口包括两类, 用于指令访存的接口和与处理器硬件电路逻辑直接交互的接口。

接口名称	接口类型	含义	位宽	输入或输出
csr_re	指令访问	读使能	1	input
csr_num	指令访问	寄存器号	14	input
csr_rvalue	指令访问	寄存器返回值	32	output
csr_we	指令访问	写使能	1	input
csr_wmask	指令访问	写掩码	32	input
csr_wvalue	指令访问	写数据	32	input
wb_ex	硬件电路交互	异常处理触发信号	1	input
wb_ecode	硬件电路交互	异常类型	6	input
wb_esubcode	硬件电路交互	异常类型	9	input
wb_pc	硬件电路交互	触发异常的 pc 值	32	input
wb_vaddr	硬件电路交互	触发异常的虚地址	32	input
ertn_flush	硬件电路交互	ertn 指令	1	input
....				

在写回级流水级模块 wb_reg 中, 创建上述所有的接口。在 cpu_top 模块中, 将两个模块 wb_reg 和 csrfile 中对应的接口相连接。

最后, 在 csrfile 模块中创建了所有输入输出接口后, 按照讲义上的内容, 以每个 CSR 的各个域作为基本单位, 依次实现各个控制寄存器初始化, 被指令访问和修改, 被硬件电路逻辑访问和修改。

2. 增加 csrrd, csrwr, csrxchg 指令

在 ID 流水级模块中, 通过译码操作, 解析出 csr_we, csr_re, csr_num, csr_wmask 信息, 并创建从 ID 流水级到 WB 流水级的逐级传递的数据通路, 用来传递这些信号。

```
assign id_csr_re = inst_csrrd || inst_csrwr || inst_csxchg || inst_ertn; // 控制寄存器读使能
assign id_csr_num = csr_num; // 控制寄存器访问编号
assign id_csr_we = inst_csrwr || inst_csxchg; // 控制寄存器写使能
assign id_csr_wmask = inst_csxchg ? rj_value: ~32'b0; // 控制寄存器写掩码
```

在原来的通用寄存器读出数据 `rj_valuw`, `rkd_value` 的逐流水级传递数据通路的基础上, 延长其至写回级, 作为控制寄存器访问的数据和掩码。

例如, 原来 `rkd_value` 的逐级传递到 `wb` 流水级为止, 作为写回 `mem` 的数据。将其继续逐级传递到 `wb` 流水级, 并且作为 `csr` 控制寄存器的写数据。

```
assign csr_wvalue = rkd_value; // rd 寄存器的旧值作为控制寄存器的写数据。
```

3. 添加 `ertn` 和 `syscall` 指令

在 `id` 级添加相应的译码逻辑, 生成将当前指令是 `ertn` 指令的信号, 以及异常使能信号, 异常类型相关信号。并依附于流水级逐级传递,

至写回级模块通过第一步定义的接口, 与 `csrfile` 模块中的对应接口相连接。

```
assign id_ertn_flush = inst_ertn; // 当前指令是ertn

assign id_excep_INT = has_int; // 记录中断信号
assign id_excep_SYSCALL = inst_syscall; // 记录该条指令是否存在SYSCALL异常
assign id_excep_BRK = inst_break; // 记录该条指令是否存在BRK异常
assign id_excep_INE = no_inst; // 记录该条指令是否存在INE异常
assign id_excep_en = id_excep_SYSCALL | id_excep_BRK | id_excep_INE | if_excep_en;
//只要有一个异常就置1
```

并向每个流水级增加从 `wb` 发出的清空流水级的信号 `flush`。

从 `wb` 流水级向 `id` 流水级传递异常跳转地址 `era`, 如图所示 每个流水级, 接受到清空流水线的信号时, 将

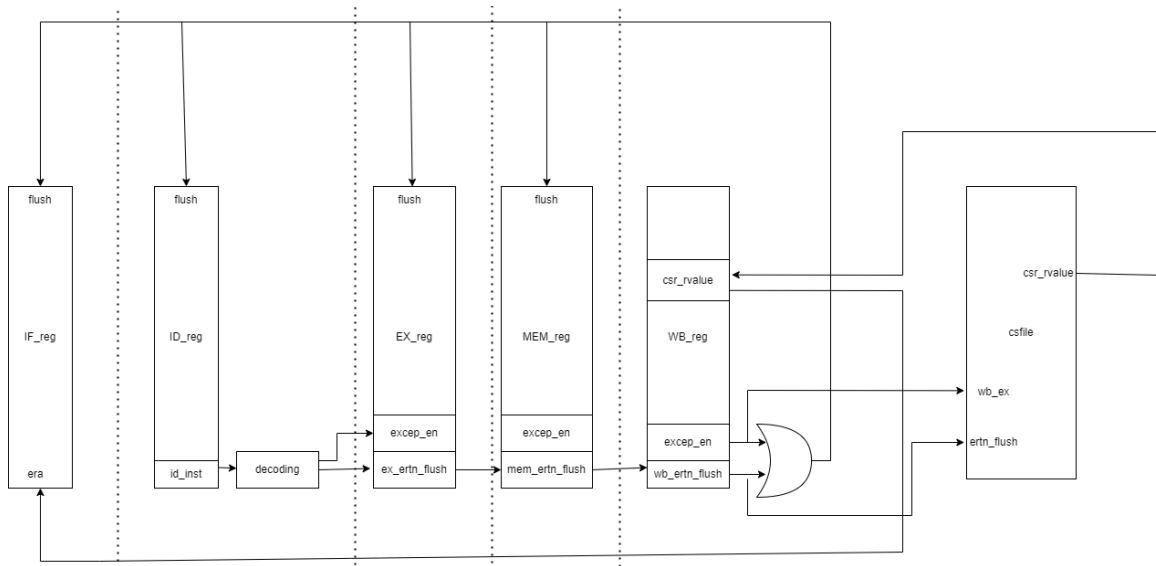


图 1: `syscall` 和 `ertn` 增加的部分数据通路

当前流水级的 `valid` 置 0。

在 `IF` 流水级, 如果接受到清空流水线的信号, 将下一个 `pc` 值设置为从 `wb` 传来的返回的异常处理地址。

```
assign pre_pc = flush ? wb_csr_rvalue // 将下一个pc值设置为从wb传来的返回的异常处理地址
              : br_taken ? br_target
              : seq_pc;
```

二、 实验过程中遇到的问题、对问题的思考过程及解决方法(比如 RTL 代码中出现的逻辑 bug, 逻辑仿真和 FPGA 调试过程中的难点等)

- wb 流水级的 flush 有效信号只持续一个 clk。

在清空流水线的设计时, 将 wb 流水级在发出 flush 信号时的下一个 clk 时, 也要将 wb_valid 置为 0, 避免重复清空流水线。

使得如果 IF 流水级的 allowin 由于读后写数据相关等原因因为 0 时, 不能及时把 pre_pc = wb_csr_rvalue 发送给 inst_sram, 而随着下一个 clk 的 wb_valid 拉低, 使从 wb 传到 if 的 wb_csr_rvalue 信号也失效。

在这种情况下, 不能正确地进行跳转到异常处理地址。

在分析波形图后, 确定上述 bug 后, 为 if_allowin 添加上规则, 使得接收到清空流水线时, 立马将 allowin 拉高, 在当前 clk 发送出 pc

```
assign if_allowin    =  ~if_valid           // valid是reg类型, 接受flush后最快下一个clk才能拉低
                    |  if_ready_go & id_allowin // id_allowin可能由于读后写阻塞, 拉低
                    |  flush; // 添加上规则, 立马将allowin拉高, 在当前clk发送出pc
```

三、 小组成员分工合作情况

王敬华负责

李霄宇负责

艾华春负责 exp12: 添加系统调用异常支持

实验报告为根据每人负责代码的部分, 写相应部分的报告。