浙江大学

本科实验报告

课程名称：计算机体系结构

姓　　名：刘思锐

学　　院：计算机科学与技术学院

系：

专　　业：计算机科学与技术

学　　号：3200102708

指导教师：陈文智

2022 年　10 月　25 日

浙江大学实验报告

课程名称：计算机体系结构 实验类型：综合

实验项目名称：Pipelined CPU supporting exception & interrupt

学生姓名：刘思锐 专业：计算机科学与技术 学号：3200102708

同组学生姓名：陈镛屹 指导老师：陈文智

实验地点：曹西 301 实验日期：2022 年 10 月 25 日

一、 实验目的和要求

(1) Understand the principle of CPU exception & interrupt and its processing procedure.
(2) Master the design methods of pipelined CPU supporting exception & interrupt.
(3) Master methods of program verification of Pipelined CPU supporting exception & interrupt.

二、 实验内容和原理

(1) 实验内容

i. Design of Pipelined CPU supporting exception & interrupt.
   1. Design datapath
   2. Design Co-processor & Controller

ii. Verify the Pipelined CPU with program and observe the execution of program

(2) 实验原理

i. RISC-V 机器发生中断、异常时，硬件应该自动完成如下内容：
   1. 异常指令的 PC 被保存在 mepc 中，PC 被设置为 mtvec。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。
   2. 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。
   3. 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 中。
   4. 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。

ii. 除对中断和异常的处理之外，中断命令还需要与正常操作 CSR 的指令相兼容。

三、 实验过程和数据记录

(1) 因为发生中断、异常时需要向多个 CSR 地址中写入，而 CSR 一个时钟周期只能写入一个数据，因此该实验一定需要状态机进行多时钟处理。

i. 生成异常、中断信号与其原因。

StartState 置 1 表示当前出现了异常、异常需要处理。而因为 Interrupt 外部中断是异步的，这里 StartState 也是由组合逻辑生成的异步信号。

```
parameter badInstructionType = 2'd0;
parameter ecallType = 2'd1;
parameter interruptType = 2'd2;
parameter mretType = 2'd3;
assign exceptionType =  (badInstructionType & {2{illegal_inst | l_access_fault | s_access_fault}}) |
                        (ecallType & {2{ecall_m}}) |
                        (interruptType & {2{interrupt & mstatus[3]}}) |
                        (mretType & {2{mret}});

wire startState;
assign startState = illegal_inst | l_access_fault | s_access_fault |
                    ecall_m | interrupt & mstatus[3] | mret | (state != 3'd0);
```

ii. 检测到异常和中断时的处理

1. 清空流水线

```
// 检测到错误，flush所有寄存器，下一次IF的地址发生变化
assign reg_FD_flush = (state == 3'd1) || (state == 3'd7);
assign reg_DE_flush = (state == 3'd1) || (state == 3'd7);
assign reg_EM_flush = (state == 3'd1) || (state == 3'd7);
assign reg_MW_flush = (state == 3'd1) || (state == 3'd7);
assign RegWrite_cancel = (state == 3'd1) || (state == 3'd7);
```

2. 生成 PC redirect mux 信号控制跳转

```
    assign PC_redirect = csr_r_data_out;
//    assign PC_redirect = exceptionType == mretType? csr_r_data_out:mepc;
    assign redirect_mux = (state == 3'd1) || (state == 3'd7);
```

iii. 设计状态机控制部分

1. State 0 表示当前没有发生中断、异常。
2. State 1-4 表示当前正在进行异常处理。
3. 因为发生外部中断需要跳转到的 MTVEC 由 CSR 读出，是同步信号；而外部中断是异步信号，因此需要对 exceptionType 做出

特判，当外部中断时需要一个额外的缓冲周期 state 7，以保证可以正常读取 MTVEC 并跳转。

```
// 控制state自增
always @(posedge startState or posedge clk or posedge rst) begin
//    always @(posedge clk or posedge rst) begin
        if (rst) begin
            state <= 3'd0;
        end
        else if (state == 3'd0 & startState) begin
            if(exceptionType != interruptType) begin
                state <= 3'd1;
            end
            else begin
                state <= 3'd7;
                prev_epc_cur <= epc_cur;
                prev_epc_next <= epc_next;
            end
            thisType <= exceptionType;
            EPC <= epc_cur;
        end
        else if(state == 3'd7) begin
            state <= 3'd1;
        end
        else if (state == 3'd1) begin
            state <= 3'd2;
        end
        else if (state == 3'd2) begin
            state <= 3'd3;
        end
        else if (state == 3'd3) begin
            state <= 3'd4;
        end
```

iv. 设计状态机功能部分

1. State 0，没有发生中断或异常

```
always @(negedge clk) begin
    if (rst) begin
        csr_w <= 0;
    end
    else if (state == 3'd0) begin
        csr_w     <= csr_rw_in;
        csr_raddr <= csr_rw_addr_in;
        csr_waddr <= csr_rw_addr_in;
        csr_wsc   <= csr_wsc_mode_in;
        csr_wdata <= csr_w_imm_mux ? {{27{0}},csr_w_data_imm} : csr_w_data_reg;
    end
```

2. State 1-4 按照异常、中断处理的要求分别写入 MEPC、MCAUSE、TVAL、MSTATUS

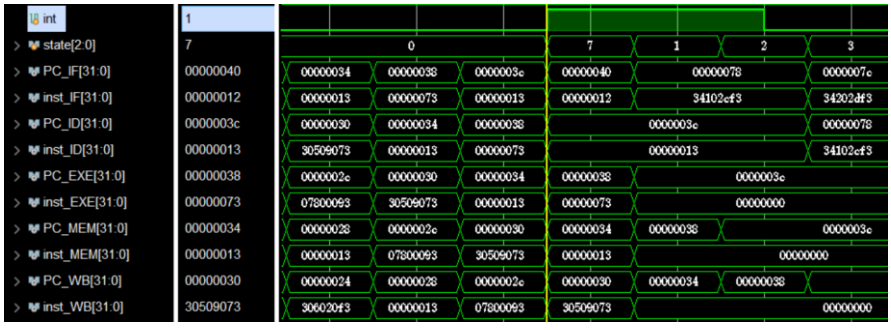
```
        else if (state == 3'd1) begin
            csr_w <= thisType != 2'd3;
            csr_raddr <= jumpDest;
            csr_waddr <= MEPC;
            csr_wdata <= thisType == interruptType? prev_epc_next:epc_cur;
            csr_wsc   <= 2'b01;

            cause <=  ecall_m ? 32'hb : (l_access_fault ? 32'h5 :
                        (s_access_fault ? 32'h7 : (illegal_inst ? 32'h2 : 32'hb)));
        end
        else if (state == 3'd2) begin
            csr_w <= thisType != 2'd3;
            csr_waddr <= MCAUSE;
            csr_wdata <= cause;
            csr_wsc   <= 2'b01;
        end
        else if (state == 3'd3) begin
            csr_w     <= 1;
            csr_waddr <= MTVAL;
            csr_wdata <= EPC;
            csr_wsc <= 2'b01;
        end
        else if (state == 3'd4) begin
            csr_w     <= 1;
            csr_waddr <= MSTATUS;
            csr_wdata <= thisType == mretType ?
                        {mstatus[31:8], 1'b0, mstatus[6:4], mstatus[7], mstatus[2:0]}:
                        {mstatus[31:8], mstatus[3], mstatus[6:4], 1'b0, mstatus[2:0]};
        end
    end
```

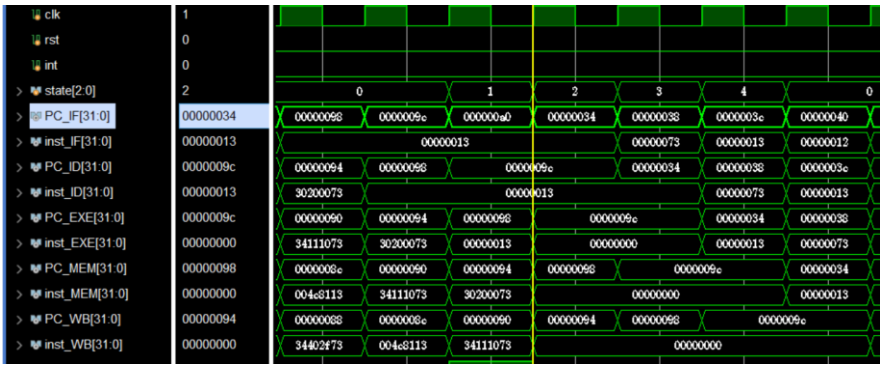
v. 连接 CSR 寄存器组件

```
CSRRegs csr(.clk(clk),.rst(rst),.csr_w(csr_w),.raddr(csr_raddr),.waddr(csr_waddr),
    .wdata(csr_wdata),.rdata(csr_r_data_out),.mstatus(mstatus),.csr_wsc_mode(csr_wsc), .mepc(mepc));
```

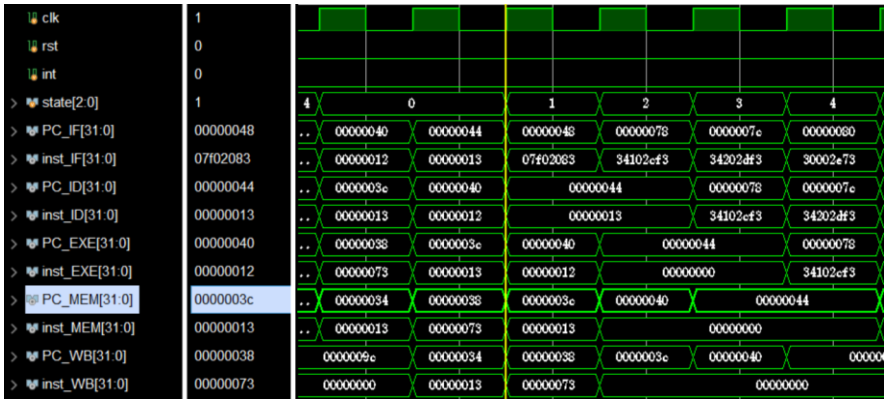
vi. 进行仿真观察结果

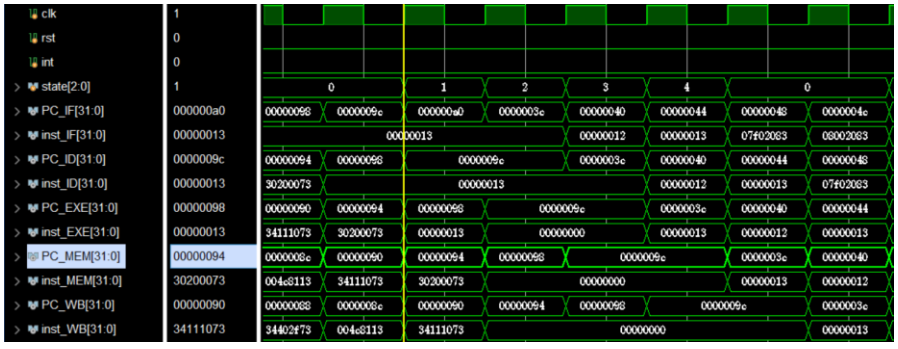
1. 发生中断时，经过一个缓冲周期，跳转到 0x78 地址；处理结束后 mret 回到中断处继续执行



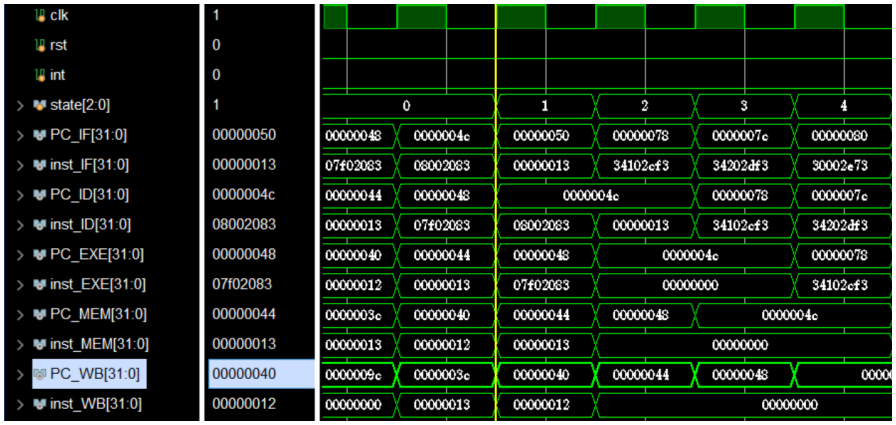


2. 在 WB 级响应 ECALL 指令；处理结束后 mret 回到异常的下一指令处继续执行





3. 在 WB 级发现非法指令；mret 时同 ECALL



四、　讨论与心得

最初为了避免时序上的冲突，我状态机的控制部分使用的并不是常见的组合逻辑而是 negedge clk，这样可以为处理异常、生成地址和信号供 CSR 读写提供便利。但是因为外部中断是一个异步的信号，时钟的 posedge、negedge 又都已经使用了，在接入外部中断时我遇到了无数时序上的麻烦。最终通过各种尝试终于在验收当天调出了用一个异步缓冲状态 state 7 将中断信号同步的方法。**上板忘记拍照了可以放过我吗 orz。**