Computer system III Lab 1 Report

3200105787 张云策

一.实验目的

- 回顾 OS 在硬件层面的抽象
- 回顾硬件设计，仿真，调试方法
- 在硬件上运行简易 ELF 程序: Naive Kernel

二.实验模块

CSR设计模块：

```
module CSR(
    input clk,
    input rst,
    input [31:0] inst,
    input [6:0] op_code,
    input [2:0] funct3,
    input [11:0] csr_addr,
    input [4:0] zimm,
    input [31:0] csr_din,
    input [31:0] pc_out,
    output reg [31:0] csr_dout,
    output reg [31:0] csr_pc_next,
    output reg csr_pc_write,
    output reg csr_write
    );
        reg [31:0] mtvec;
        reg [31:0] mepc;
        reg [31:0] mstatus;

        initial begin
            mtvec<=0;
            mepc<=0;
            mstatus<=0;
            csr_pc_next<=0;
            csr_write<=0;
        end
        always @(*) begin
        csr_write=0;
            if (rst == 1) begin
                csr_write=0;
            end
            else if (inst==32'h00000073 || inst==32'h00100073)begin//ECALL
                   csr_write = 1;
               end else if (inst==32'h30200073)begin
                csr_write = 1;
               end
               else begin
                 if (inst[6:0]==7'b1110011) begin
                csr_write =1;
                end
```

```
        end
        end



        always @(posedge clk or posedge rst)begin
            if(rst==1)begin
            mtvec<=0;
            mepc<=0;
            mstatus<=0;
            end

            else begin
            if (op_code==7'b1110011) begin
              if (csr_addr == 12'h300) begin
                  case(funct3)
                     3'b011:begin
                        mstatus <= mstatus & csr_din;
                     end
                     3'b111:begin
                         mstatus <= mstatus & {27'b0,zimm};
                     end
                     3'b010:begin
                         mstatus <= mstatus | csr_din;
                     end
                     3'b110:begin
                         mstatus <= mstatus | {27'b0,zimm};
                     end
                     3'b001:begin
                         mstatus <= csr_din;
                     end
                     3'b101:begin
                         mstatus <= {27'b0,zimm};
                     end
                  endcase
               end
               else if(csr_addr == 12'h305)begin
                   case(funct3)
                   3'b011:begin
                   mtvec <= mtvec & csr_din;
                   end
                   3'b111:begin
                       mtvec <= mtvec & {27'b0,zimm};
                    end
                   3'b010:begin
                       mtvec <= mtvec | csr_din;
                   end
                   3'b110:begin
                       mtvec <= mtvec | {27'b0,zimm};
                    end
                    3'b001:begin
                      mtvec <= csr_din;
                    end
                    3'b101:begin
                      mtvec <= {27'b0,zimm};
                    end
                   endcase
                 end
```

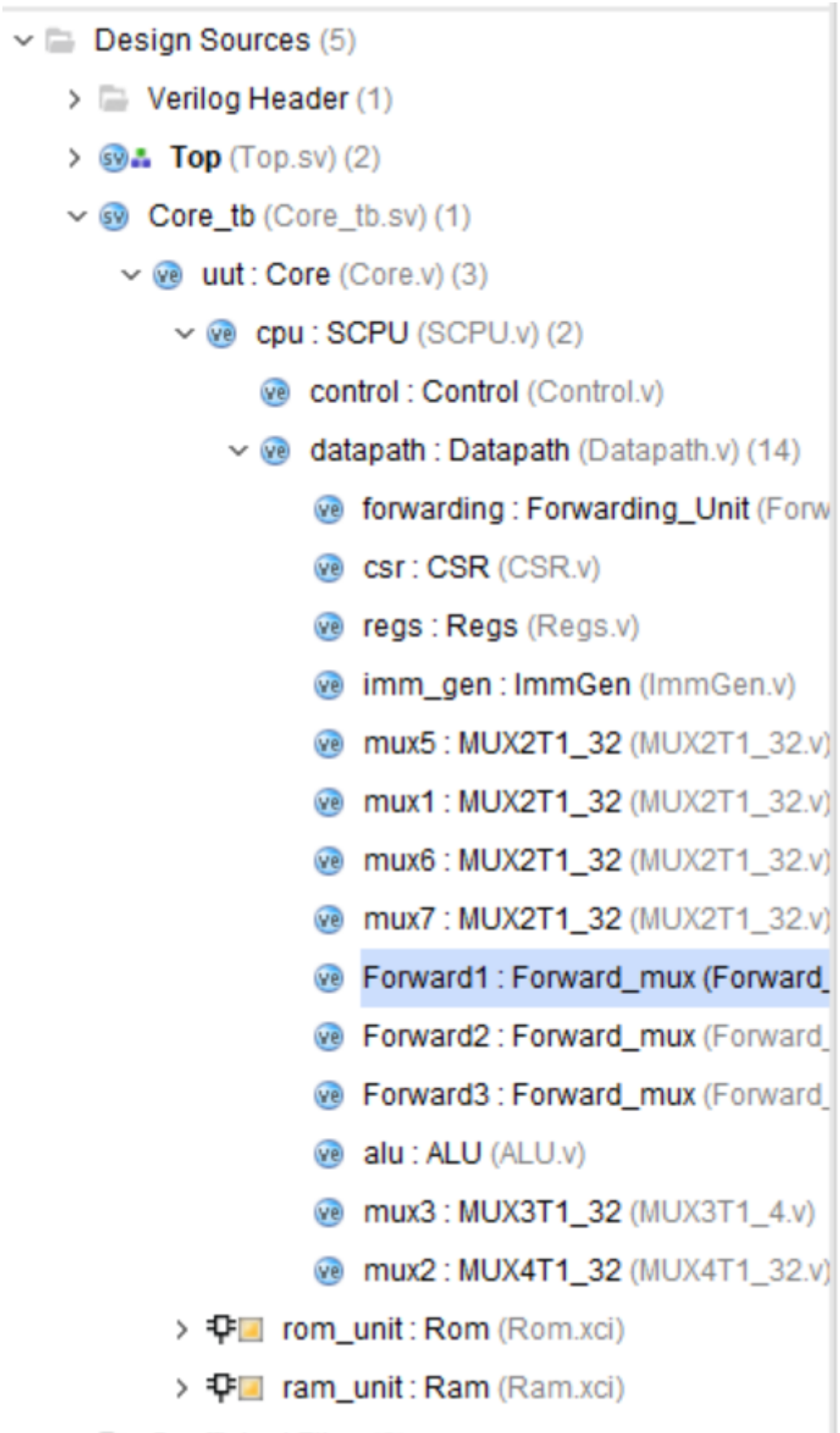
```
                else if(csr_addr == 12'h341)begin
                    case(funct3)
                        3'b011:begin
                            mepc <= mepc & csr_din;
                        end
                        3'b111:begin
                            mepc <= mepc & {27'b0,zimm};
                        end
                        3'b010:begin
                            mepc <= mepc | csr_din;
                        end
                        3'b110:begin
                            mepc <= mepc | {27'b0,zimm};
                        end
                        3'b001:begin
                          mepc <= csr_din;
                        end
                        3'b101:begin
                          mepc <= {27'b0,zimm};
                        end
                     endcase
                   end
                else if(csr_addr == 12'h000 || csr_addr == 12'h001)begin
mepc<=pc_out;end
                end
            end
        end

always @(*)begin
csr_pc_write=0;
    if(op_code==7'b1110011)begin
        if(csr_addr==12'h302)begin
            csr_pc_next=mepc;
            csr_pc_write=1;
        end
        else if(csr_addr==12'h000 || csr_addr==12'h001)begin
            csr_pc_next=mtvec;
            csr_pc_write=1;
        end
    end
end
        always @(*) begin

            if (rst == 1) begin
                csr_dout<=0;
            end

           else begin
             if (inst[6:0]==7'b1110011) begin
                if (inst[31:20] == 12'h300) begin
                    csr_dout <= mstatus;
                end
                else if(inst[31:20] == 12'h305)begin
                    csr_dout <= mtvec;
                end
                else if(inst[31:20] == 12'h341)begin
                    csr_dout <= mepc;
                end
```

```
            end
        end
        end
endmodule
```

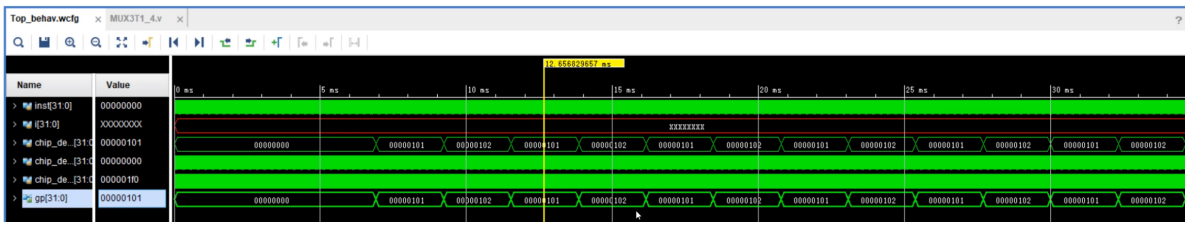
Forward选择器模块：

```
module Forward_mux(
    input [31:0] I0,
    input [31:0] I1,
    input [31:0] I2,
    input [31:0] I3,
    input e,
    input [1:0] s,
    output reg [31:0] o
    );
    always @(*) begin
    if(e!=1)
        case(s)
            2'b00: o = I0;
            2'b01: o = I1;
            2'b10: o = I2;
            2'b11: o = I3;
            default: ;
        endcase
        else o=I0;
    end
endmodule
```

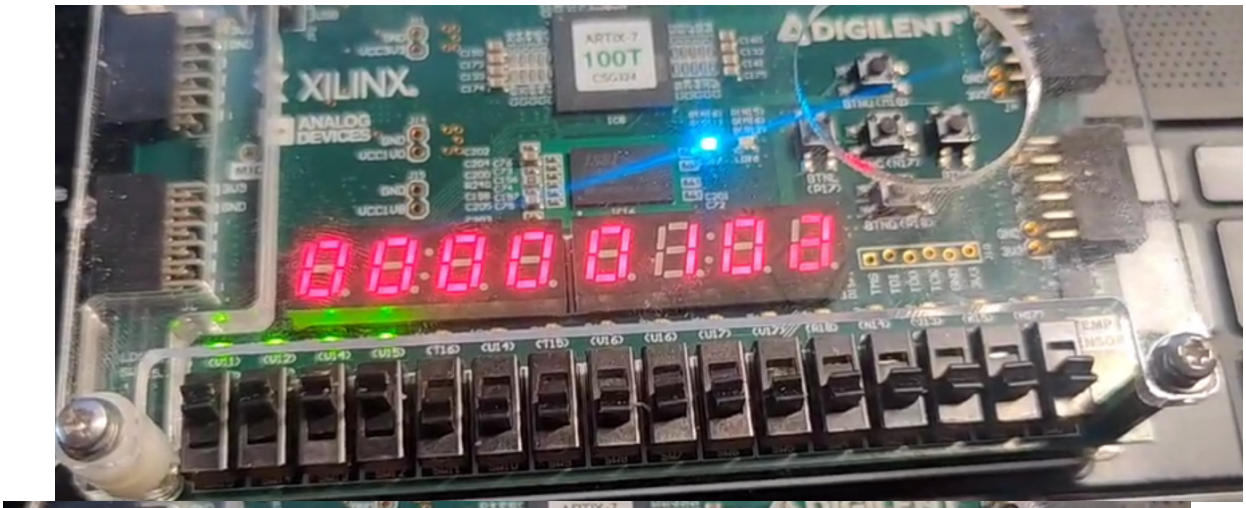
仿真：

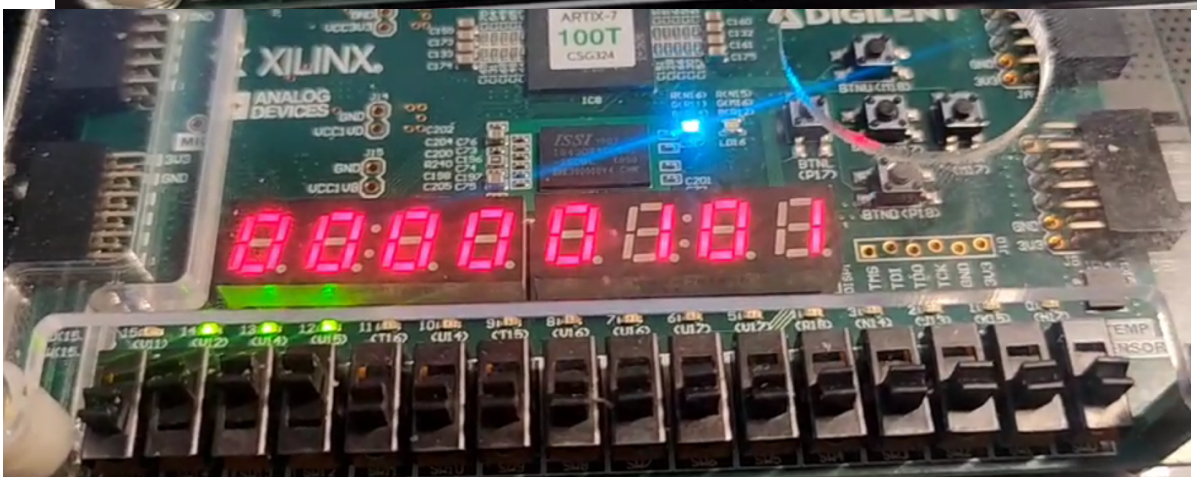

Design Sources (5)
Verilog Header (1)
Top (Top.sv) (2)
Core_tb (Core_tb.sv) (1)
uut : Core (Core.v) (3)
cpu : SCPU (SCPU.v) (2)
control : Control (Control.v)
datapath : Datapath (Datapath.v) (14)
forwarding : Forwarding_Unit (Forw
csr : CSR (CSR.v)
regs : Regs (Regs.v)
imm_gen : ImmGen (ImmGen.v)
mux5 : MUX2T1_32 (MUX2T1_32.v)
mux1 : MUX2T1_32 (MUX2T1_32.v)
mux6 : MUX2T1_32 (MUX2T1_32.v)
mux7 : MUX2T1_32 (MUX2T1_32.v)
Forward1 : Forward_mux (Forward_
Forward2 : Forward_mux (Forward_
Forward3 : Forward_mux (Forward_
alu : ALU (ALU.v)
mux3 : MUX3T1_32 (MUX3T1_4.v)
mux2 : MUX4T1_32 (MUX4T1_32.v)
rom_unit : Rom (Rom.xci)
ram_unit : Ram (Ram.xci)

仿真结果：


Top_behav.wcfg
MUX3T1_4.v
Name
Value
inst[31:0]
00000000
i[31:0]
XXXXXXXX
chip_de...[31:0]
00000101
chip_de...[31:0]
00000000
chip_de...[31:0]
000001f0
gp[31:0]
00000101

上板结果：





三.实验思考题

从硬件设计的角度看，发生异常和发生中断是否有区别?

- 中断由CPU外的中断信号触发启动，系统无法得知何时会出现中断情况；异常则是由CPU自己检测得出，由控制器产生。
- 中断发生后，需要CPU优先处理，并且需要处理后再继续进行下一任务；异常大都出现在用户态，由CPU进行同步处理。
- 中断通常不会导致异常的出现；但异常可能导致中断发生。