浙江大学

体系结构实验报告

课程名称:	计算机体系结构
实验项目:	Dynamically Scheduled Pipelines
专　　业:	计算机科学技术
学生姓名:	李浩浩
学　　号:	3220105930
指导老师:	何水兵
实验日期:	2024年11月25日

一、实验目的、要求及任务

（一）实验目的

- Understand the principle of piplines that support multicycle operations.
- Master the design methods of piplines that support multicycle operations.
- Master verification methods of pipelined CPU supporting multicycle operations.

（二）实验任务

- Redesign the pipelines with IF/ID/FU/WB stages and FU stage supporting multicycle operations.
- Redesign of CPU Controller.
- Verify the Pipelined CPU with program and observe the execution of program.

二、实验原理

在设计 CPU 功能扩展和模块化的过程中，功能单元 (FU) 阶段被细分为五种功能模块，包括算术逻辑单元、访存单元、乘法单元、除法单元和跳转单元。每种单元完成指令所需的时钟周期不同，例如，算术逻辑单元（ALU）执行一条指令仅需一个时钟周期，而访存单元则需要两个周期，其他单元也有各自的延迟。

为解决数据相关问题，如果一条指令需要依赖之前指令的计算结果，那么必须等到前一指令的结果写回（即 WB 阶段结束）后，后续指令才能进入功能单元 (FU) 阶段。

在处理控制相关问题时，CPU 默认假设分支或跳转指令不发生跳转。若在功能单元阶段发现分支预测错误，则所有在该指令之后捕获的指令（共计两条）将被撤销，并从正确的目标地址开始重新取指。

三、实验过程及数据记录

FU_ALU.v 补全部分

```
    reg state;
    assign finish = state == 1'b1;
    initial begin
        state = 0;
    end

    reg[3:0] Control;
    reg[31:0] A, B;

    always@(posedge clk) begin
        if(EN & ~state) begin // state == 0
            Control <= ALUControl;            //to fill sth.in
            A <= ALUA;
            B <= ALUB;
            state <= 1;
        end
        else state <= 0;
    end
```

FU_div

```
`timescale 1ns / 1ps

module FU_div(
    input clk, EN,
    input[31:0] A, B,
    output[31:0] res,
    output finish
);

    wire res_valid;
    wire[63:0] divres;

    reg state;
    assign finish = res_valid & state;
    initial begin
        state = 0;
    end

    reg A_valid, B_valid;
    reg[31:0] A_reg, B_reg;

    always@(posedge clk) begin
        if(EN & ~state) begin // state == 0
            A_reg <= A;
            B_reg <= B;
            A_valid <= 1;
            B_valid <= B ? 1 : 0;
            state <= 1;
        end
        else if(res_valid) begin
            A_valid <= 0;
            B_valid <= 0;
            state <= 0;
        end
    end


    divider div(.aclk(clk),
        .s_axis_dividend_tvalid(A_valid),
        .s_axis_dividend_tdata(A_reg),
        .s_axis_divisor_tvalid(B_valid),
```

```
        .s_axis_divisor_tdata(B_reg),
        .m_axis_dout_tvalid(res_valid),
        .m_axis_dout_tdata(divres)
    );

    assign res = divres[63:32];

endmodule
```

FU_jump.v

```
`timescale 1ns / 1ps

module FU_jump(
        input clk, EN, JALR,
        input[2:0] cmp_ctrl,
        input[31:0] rs1_data, rs2_data, imm, PC,
        output[31:0] PC_jump, PC_wb,
        output cmp_res, finish
);

    reg state;
    assign finish = state == 1'b1;
        initial begin
        state = 0;
    end

        reg JALR_reg;
        reg[2:0] cmp_ctrl_reg;
        reg[31:0] rs1_data_reg, rs2_data_reg, imm_reg, PC_reg;

        always@(posedge clk) begin
        if(EN & ~state) begin // state == 0
            JALR_reg <= JALR;
                        cmp_ctrl_reg <= cmp_ctrl;
                        rs1_data_reg <= rs1_data;
                        rs2_data_reg <= rs2_data;
                        imm_reg <= imm;
                        PC_reg <= PC;
            state <= 1;
        end
        else state <= 0;
    end

        cmp_32 cmp(.a(rs1_data_reg), .b(rs2_data_reg), .ctrl(cmp_ctrl_reg), .c(cmp_res));
        add_32 a(.a(PC_reg), .b(32'd4), .c(PC_wb));
        add_32 b(.a(JALR_reg ? rs1_data_reg : PC_reg), .b(imm_reg), .c(PC_jump));

endmodule
```

FU_mem.v

```
`timescale 1ns / 1ps

module FU_mem(
    input clk, EN, mem_w,
    input[2:0] bhw,
    input[31:0] rs1_data, rs2_data, imm,
    output[31:0] mem_data,
    output finish
);

    reg[1:0] state;
    assign finish = state[0] == 1'b1;
    initial begin
        state = 0;
    end

    reg mem_w_reg;
    reg[2:0] bhw_reg;
    reg[31:0] rs1_data_reg, rs2_data_reg, imm_reg;

    always@(posedge clk) begin
        if(EN & ~|state) begin // state == 0
            mem_w_reg <= mem_w;
            bhw_reg <= bhw;
            rs1_data_reg <= rs1_data;
            rs2_data_reg <= rs2_data;
            imm_reg <= imm;
            state <= 2;
        end
        else state <= state >> 1;
    end

    wire[31:0] addr;
    add_32 a(.a(rs1_data_reg), .b(imm_reg), .c(addr));

    RAM_B ram(.clka(clk),.addra(addr),.dina(rs2_data_reg),.wea(mem_w_reg),
        .douta(mem_data),.mem_u_b_h_w(bhw_reg));

endmodule
```

FU_mul.v

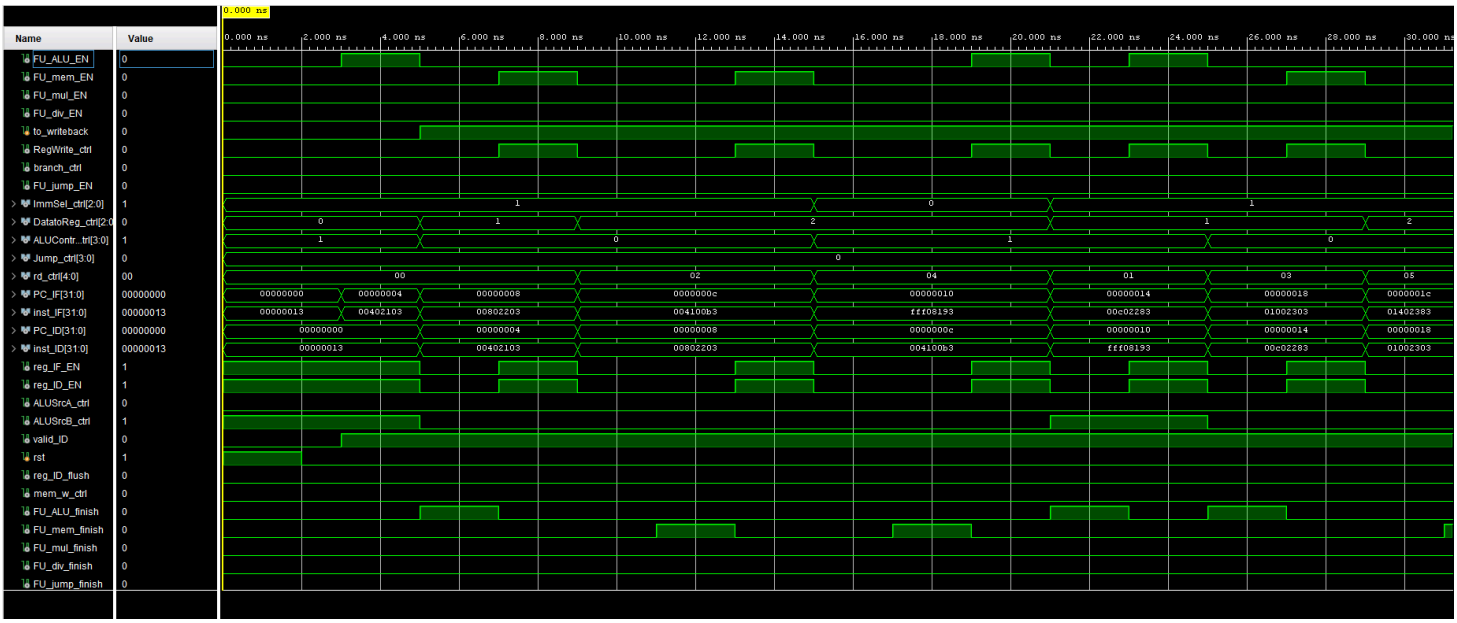
```
`timescale 1ns / 1ps

module FU_mul(
    input clk, EN,
    input[31:0] A, B,
    output[31:0] res,
    output finish
);
    wire [63:0] mulres;
    reg[6:0] state;
    assign finish = state[0] == 1'b1;
    initial begin
        state = 0;
    end

    reg[31:0] A_reg, B_reg;

    always@(posedge clk) begin
        if(EN & ~|state) begin // state == 0
            A_reg <= A;
            B_reg <= B;
            state[6] <= 1'b1;
        end
        else state <= state >> 1;
    end

    multiplier mul(.CLK(clk),.A(A_reg),.B(B_reg),.P(mulres));

    assign res = mulres[31:0];

endmodule
```

顶层文件补全部分

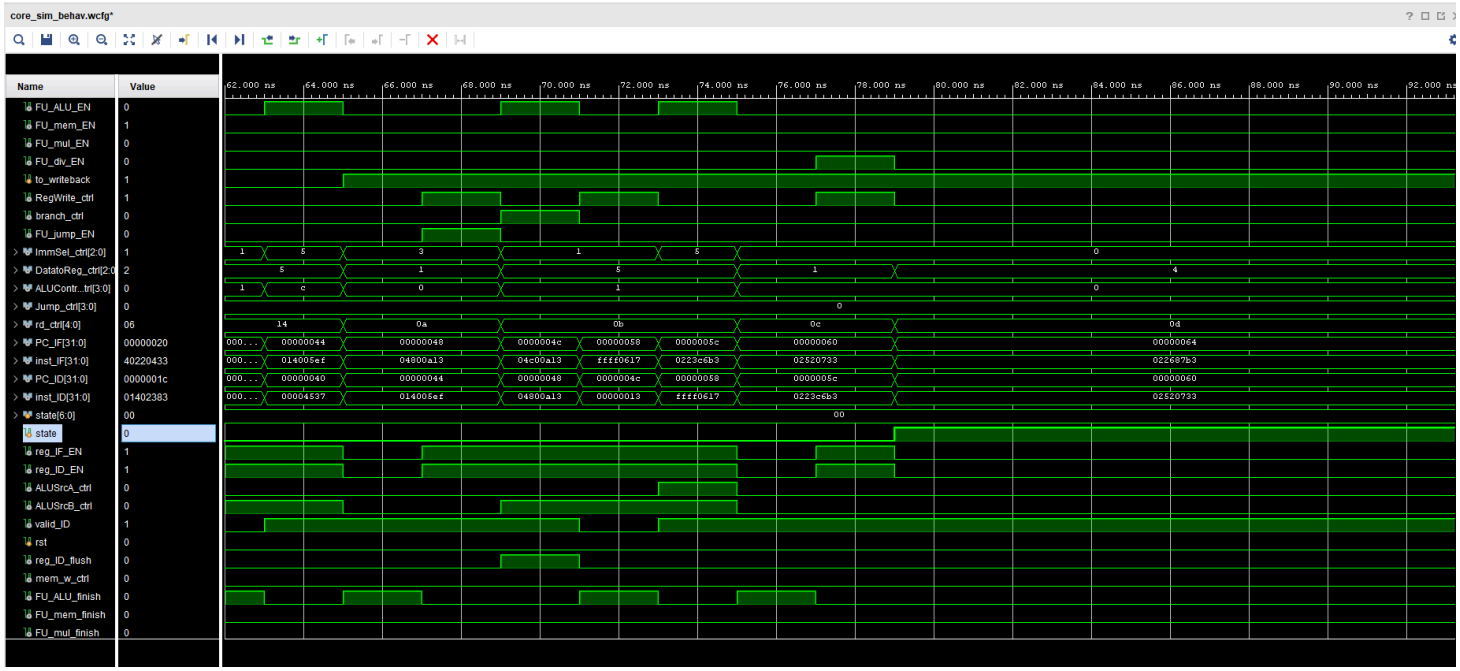
```
ImmGen imm_gen(.ImmSel(ImmSel_ctrl), .inst_field(inst_ID), .Imm_out(Imm_out_ID));

MUX2T1_32 mux_imm_ALU_ID_A(.I0(rs1_data_ID),.I1(PC_ID),.s(ALUSrcA_ctrl),.o(ALUA_ID));
MUX2T1_32 mux_imm_ALU_ID_B(.I0(rs2_data_ID),.I1(Imm_out_ID),.s(ALUSrcB_ctrl),.o(ALUB_ID));

MUX8T1_32 mux_DtR(.I0(),.I1(ALUout_WB),.I2(mem_data_WB),.I3(mulres_WB),
.I4(divres_WB),.I5(PC_wb_WB),.I6(),.I7(),.s(DatatoReg_ctrl),.o(wt_data_WB));
```

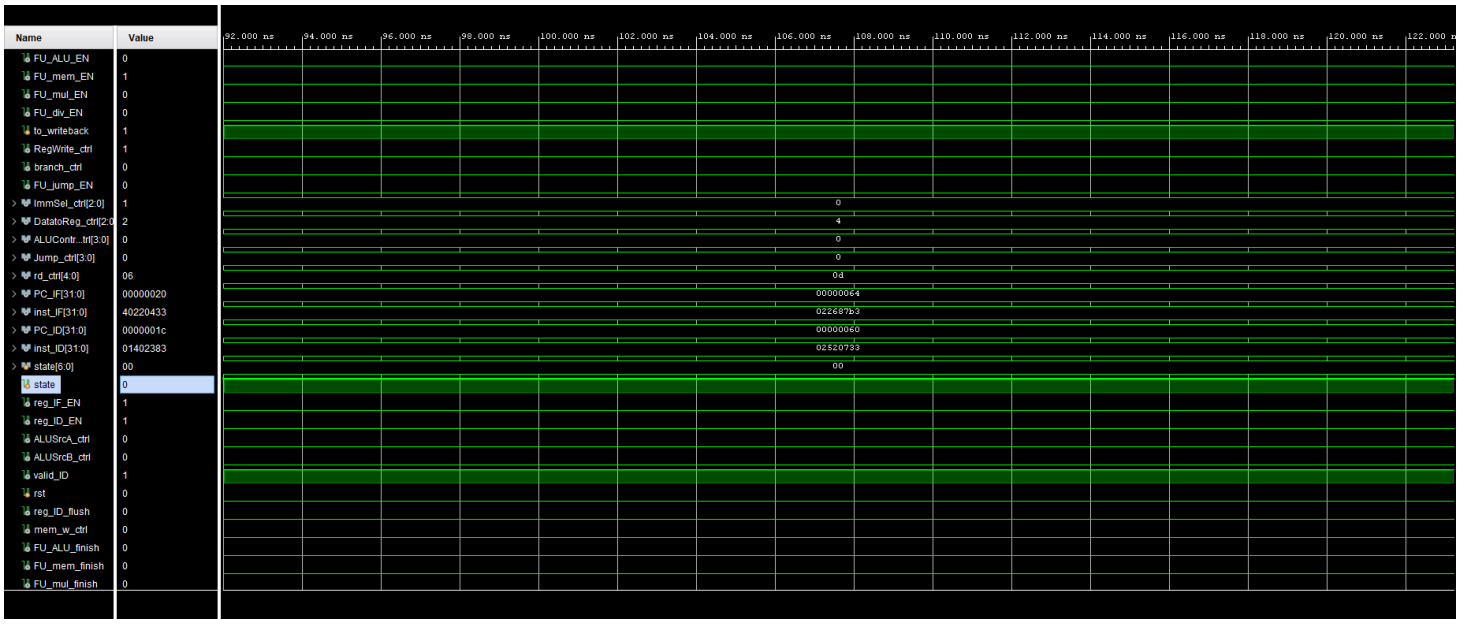
四、实验结果分析

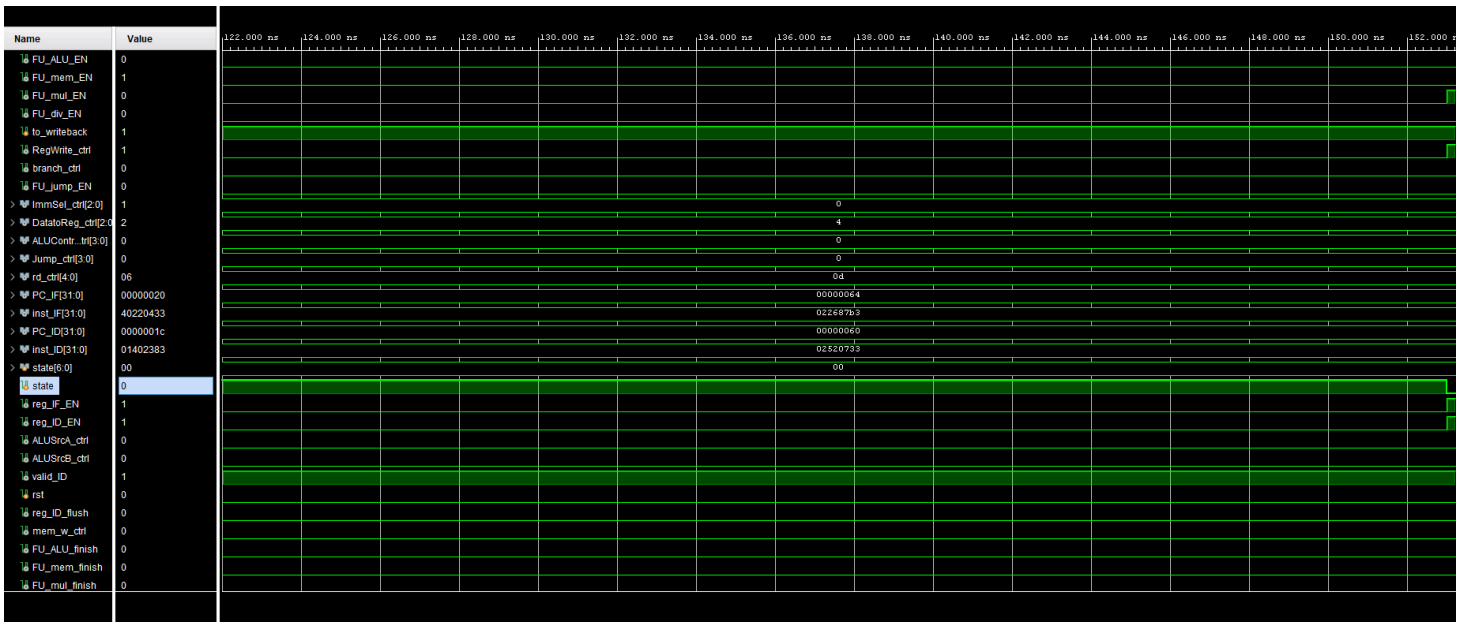
（一）仿真

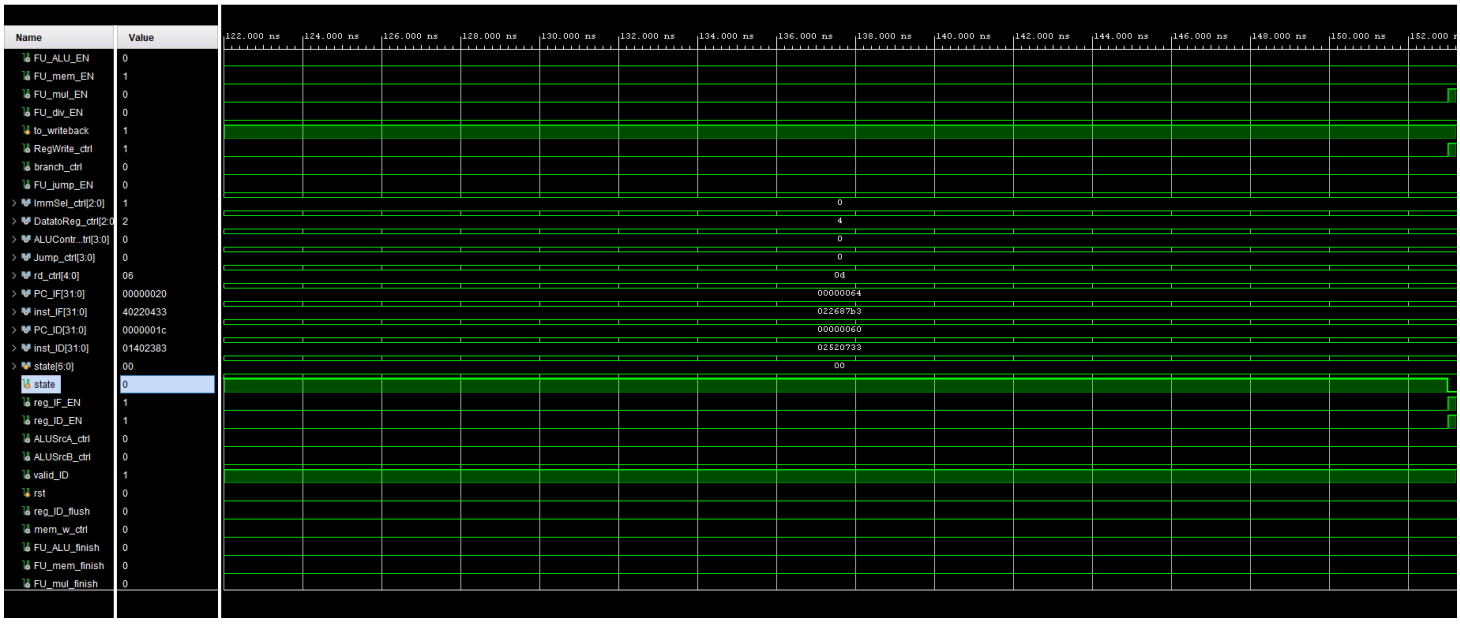


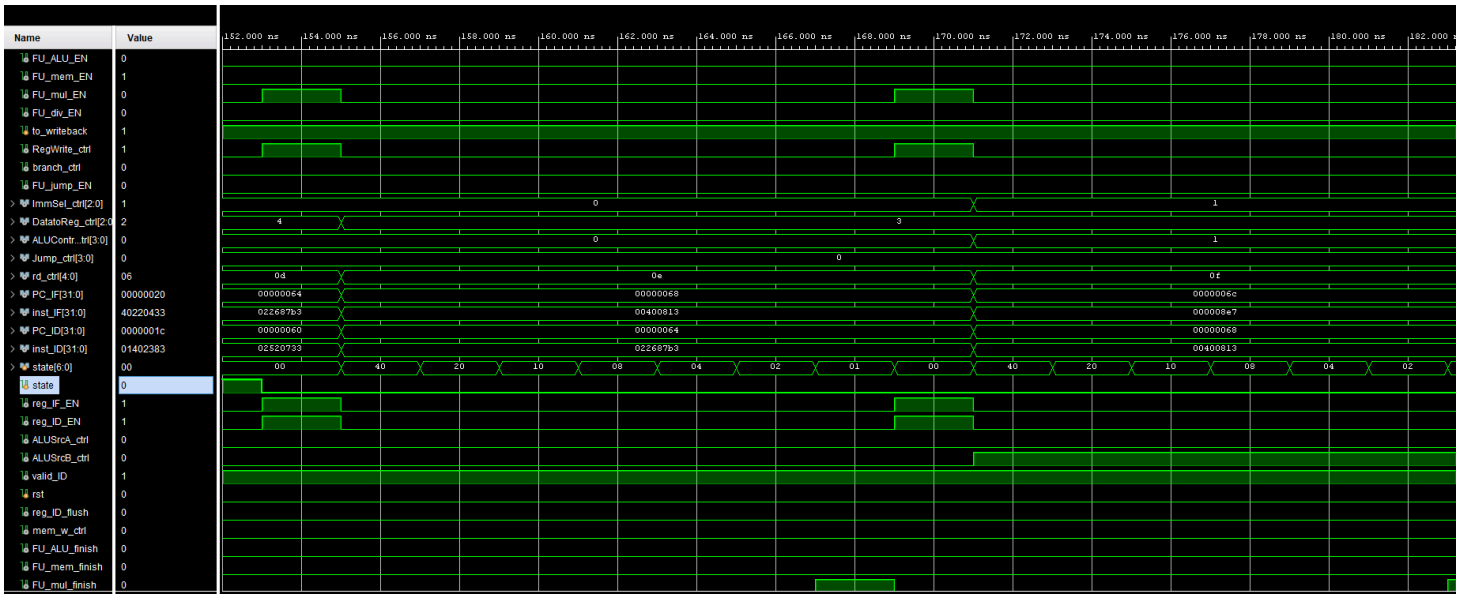

Name
Value
FU_ALU_EN 0
FU_mem_EN 0
FU_mul_EN 0
FU_div_EN 0
to_writeback 0
RegWrite_ctrl 0
branch_ctrl 0
FU_jump_EN 0
ImmSel_ctrl[2:0] 1
DatatoReg_ctrl[2:0] 0
ALUContr...trl[3:0] 1
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 00
PC_IF[31:0] 00000000
inst_IF[31:0] 00000013
PC_ID[31:0] 00000000
inst_ID[31:0] 00000013
reg_IF_EN 1
reg_ID_EN 1
ALUSrcA_ctrl 0
ALUSrcB_ctrl 1
valid_ID 0
rst 1
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
FU_div_finish 0
FU_jump_finish 0
32.000 ns
34.000 ns
36.000 ns
38.000 ns
40.000 ns
42.000 ns
44.000 ns
46.000 ns
48.000 ns
50.000 ns
52.000 ns
54.000 ns
56.000 ns
58.000 ns
60.000 ns
62.000 ns
05 06 07 08 09 10 14
0000001c 00000020 00000024 00000028 0000002c 00000030 00000034 00000040
01402383 40220433 ffd50493 00520c63 00420a63 03000a13 03400a13 00004537
00000018 0000001c 00000020 00000024 00000028 0000002c 00000030 00000034
01002303 01402383 40220433 ffd50493 00520c63 00420a63 03000a13 00000013

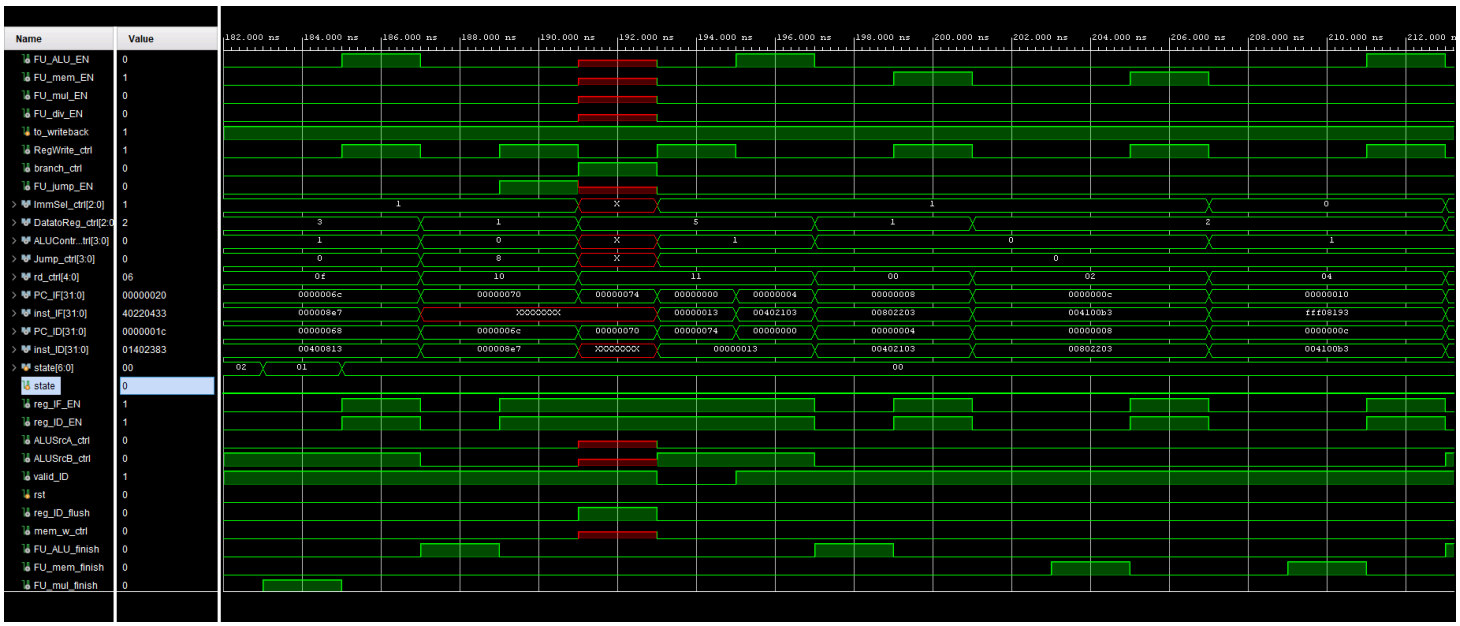

core_sim_behav.wcfg*
Name
Value
FU_ALU_EN 0
FU_mem_EN 1
FU_mul_EN 0
FU_div_EN 0
to_writeback 1
RegWrite_ctrl 1
branch_ctrl 0
FU_jump_EN 0
ImmSel_ctrl[2:0] 1
DatatoReg_ctrl[2:0] 2
ALUContr...trl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 06
PC_IF[31:0] 00000020
inst_IF[31:0] 40220433
PC_ID[31:0] 0000001c
inst_ID[31:0] 01402383
state[6:0] 00
state 0
reg_IF_EN 1
reg_ID_EN 1
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
62.000 ns
64.000 ns
66.000 ns
68.000 ns
70.000 ns
72.000 ns
74.000 ns
76.000 ns
78.000 ns
80.000 ns
82.000 ns
84.000 ns
86.000 ns
88.000 ns
90.000 ns
92.000 ns
14 0a 0b 0c 0d
00000044 00000048 0000004c 00000058 0000005c 00000060 00000064
014005ef 04800a13 04c00a13 ffff0617 0223c6b3 02520733 022687b3
00000040 00000044 00000048 0000004c 00000058 0000005c 00000060
00004537 014005ef 04800a13 00000013 ffff0617 0223c6b3 02520733


Name
Value
FU_ALU_EN 0
FU_mem_EN 1
FU_mul_EN 0
FU_div_EN 0
to_writeback 1
RegWrite_ctrl 1
branch_ctrl 0
FU_jump_EN 0
ImmSel_ctrl[2:0] 1
DatatoReg_ctrl[2:0] 2
ALUContr...trl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 06
PC_IF[31:0] 00000020
inst_IF[31:0] 40220433
PC_ID[31:0] 0000001c
inst_ID[31:0] 01402383
state[6:0] 00
state 0
reg_IF_EN 1
reg_ID_EN 1
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
92.000 ns
94.000 ns
96.000 ns
98.000 ns
100.000 ns
102.000 ns
104.000 ns
106.000 ns
108.000 ns
110.000 ns
112.000 ns
114.000 ns
116.000 ns
118.000 ns
120.000 ns
122.000 ns
0
4
0
0
0d
00000064
022687b3
00000060
02520733
00


Name
Value
FU_ALU_EN 0
FU_mem_EN 1
FU_mul_EN 0
FU_div_EN 0
to_writeback 1
RegWrite_ctrl 1
branch_ctrl 0
FU_jump_EN 0
ImmSel_ctrl[2:0] 1
DatatoReg_ctrl[2:0] 2
ALUContr...trl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 06
PC_IF[31:0] 00000020
inst_IF[31:0] 40220433
PC_ID[31:0] 0000001c
inst_ID[31:0] 01402383
state[6:0] 00
state 0
reg_IF_EN 1
reg_ID_EN 1
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
122.000 ns
124.000 ns
126.000 ns
128.000 ns
130.000 ns
132.000 ns
134.000 ns
136.000 ns
138.000 ns
140.000 ns
142.000 ns
144.000 ns
146.000 ns
148.000 ns
150.000 ns
152.000 ns
0
4
0
0
0d
00000064
022687b3
00000060
02520733
00


Name
Value
FU_ALU_EN 0
FU_mem_EN 1
FU_mul_EN 0
FU_div_EN 0
to_writeback 1
RegWrite_ctrl 1
branch_ctrl 0
FU_jump_EN 0
ImmSel_ctrl[2:0] 1
DatatoReg_ctrl[2:0] 2
ALUContr...trl[3:0] 0
Jump_ctrl[3:0] 0
rd_ctrl[4:0] 06
PC_IF[31:0] 00000020
inst_IF[31:0] 40220433
PC_ID[31:0] 0000001c
inst_ID[31:0] 01402383
state[6:0] 00
state 0
reg_IF_EN 1
reg_ID_EN 1
ALUSrcA_ctrl 0
ALUSrcB_ctrl 0
valid_ID 1
rst 0
reg_ID_flush 0
mem_w_ctrl 0
FU_ALU_finish 0
FU_mem_finish 0
FU_mul_finish 0
122.000 ns
124.000 ns
126.000 ns
128.000 ns
130.000 ns
132.000 ns
134.000 ns
136.000 ns
138.000 ns
140.000 ns
142.000 ns
144.000 ns
146.000 ns
148.000 ns
150.000 ns
152.000 ns
0
4
0
0
0d
00000064
022687b3
00000060
02520733
00





（二）上板

还未上板。

五、实验心得

这次实验一共debug三回，前两回是因为在不该出现红色的地方出现了红色，排查之后发现分别是rom和ram文件没有正确被项目识别到，将路径改为绝对路径之后遂解决。第三次是结尾没有出现红色，仔细核对指令，我发现程序卡在了`div x13, x7, x2`这个指令上。然后将`du`内的变量加入仿真视图之后发现其`state`一直处于`1`这个状态，原因是`B_valid`一直处于未知状态导致`res_valid`不为真，进而导致`state`无法归零，然后发现了一个不起眼的低级错误：

```
always@(posedge clk) begin
        if(EN & ~state) begin // state == 0
            A_reg <= A;
            B_reg <= B;
            A_valid <= 1;
            B_valid <= B_reg ? 1 : 0;<----这是错的，B_reg与B_valid将被同时赋值，不能利用B_reg给
B_valid赋值
            state <= 1;
        end
        else if(res_valid) begin
            A_valid <= 0;
            B_valid <= 0;
            state <= 0;
        end
    end
```

这次的实验debug比较得心应手，不像最开始只会瞪眼法，像无头苍蝇一样到处找问题。