实验报告成绩:	成绩评定日期:

2022 ~ 2023 学年秋季学期

《计算机系统》必修课

课程实验报告

班级：人工智能（一）2001

组长：李易阳

组员：钟青霖 初美汐

报告日期：2022.12.21

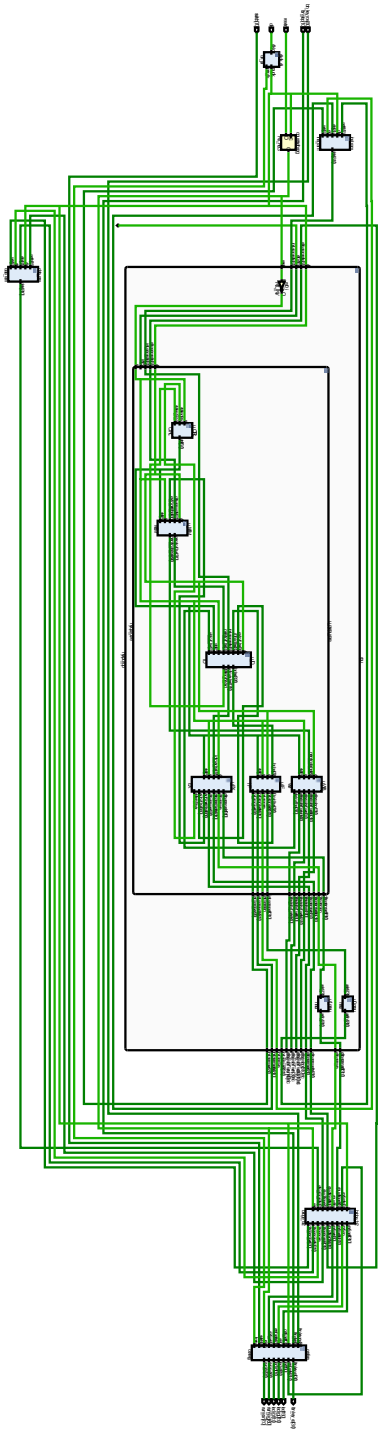
目录

一、 总体设计

1. 工作量

李易阳 80%　　初美汐 10%　　钟青霖 10%

2. 连线图



3. 完成的指令

共完成 52 条指令

运算指令：

```
wire inst_add,   inst_addi,    inst_addu,   inst_addiu,
     inst_lui,
     inst_sub,   inst_subu,
     inst_and,   inst_andi,
     inst_ori,   inst_or,       inst_xor,    inst_nor,    inst_xori,
     inst_sltu,  inst_slt,      inst_slti,   inst_sltiu,
     inst_mult, inst_multu,   inst_div,    inst_divu;
```

跳转指令：

```
wire inst_bne,   inst_beq,
     inst_bgez, inst_bgtz,   inst_bgezal,
     inst_blez, inst_bltz,   inst_bltzal,
     inst_jal,   inst_jr,       inst_j,        inst_jalr;
```

内存指令：

```
wire inst_sb,    inst_sh,      inst_sw,
     inst_lb,    inst_lbu,     inst_lh,      inst_lhu,    inst_lw;
```

移位指令:

```
wire inst_sll,   inst_sllv,   inst_sra,    inst_srav,   inst_srl,    inst_srlv;
```

hilo 指令:

```
wire inst_mfhi, inst_mflo,   inst_mthi,   inst_mtlo;
```

4. 程序运行环境及使用工具

环境：cg 平台服务器

工具：Vivado，VScode

二、单个流水段说明

1. IF 段

1.1 整体功能说明

IF 段负责取指，将得到的指令传递给 ID 段进行后续操作。

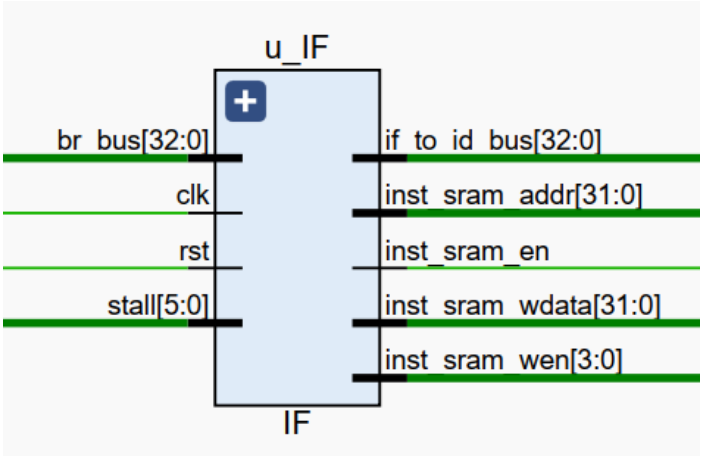
1.2 端口、信号介绍

input	clk	时钟周期信号
input	rst	复位信号
input	stall	停止信号
input	br_bus	从 ID 段传入，存放指令跳转信息
output	if_to_id_bus	从 if 到 id 的总线
output	inst_sram_en	
output	inst_sram_wen	
output	inst_sram_addr	
output	inst_sram_wdata	

1.3 功能模块说明

从 ID 段传来的 br_bus 包含两个信息，跳转使能信号 br_e 和跳转目的地址 br_addr。若 br_e 为真，则将 next_pc 赋为 br_addr，否则赋为 pc_reg + 32'h4。

1.4 结构示意图



2. ID 段

2.1 整体功能说明

ID 段负责译码，将从 IF 段传来的指令进行解析。

2.2 端口、信号介绍

input	clk	时钟周期信号
input	rst	复位信号
input	stall	停止信号
input	inst_sram_rdata	从 ID 段传入，存放指令跳转信息
intput	if_to_id_bus	从 if 到 id 的总线
intput	ex_to_rf_bus	从 ex 写回 id 的总线，用来解决数据相关
intput	wb_to_rf_bus	从 wb 写回 id 的总线，用来解决数据相关
intput	mem_to_rf_bus	从 mem 写回 id 的总线，用来解决数据相关
output	stallreq_for_load	向 CTRL 模块告知 id 是否需要暂停
output	id_to_ex_bus	从 id 到 ex 的总线
output	br_bus	向 if 段传递是否需要跳转的信息

2.3 功能模块说明

数据相关：

如果后续流水段传来的数据相关总线中写使能为真且目的地址和 id 段的 rs 或 rt 相同，则将 id 段对应的操作数改为数据相关总线中的写数据。

```
    // if previous rf_we and it's waddr == current, do forwarding
    assign rdata1 = (ex_rf_we && ex_rf_waddr == rs) ? ex_rf_wdata :
                    (mem_rf_we && mem_rf_waddr == rs) ? mem_rf_wdata :
                    (wb_rf_we && wb_rf_waddr == rs) ? wb_rf_wdata :
rf_rdata1;
    assign rdata2 = (ex_rf_we && ex_rf_waddr == rt) ? ex_rf_wdata :
                    (mem_rf_we && mem_rf_waddr == rt) ? mem_rf_wdata :
                    (wb_rf_we && wb_rf_waddr == rt) ? wb_rf_wdata :
rf_rdata2;
```

Stallreq_for_load 实现：

如果前一条是 load 指令，load 的结果最早也要在 mem 段才能拿到，那么检查 rs 或 rt 是否是 load 的目的地址，如果是则 stallreq_for_load 为真。

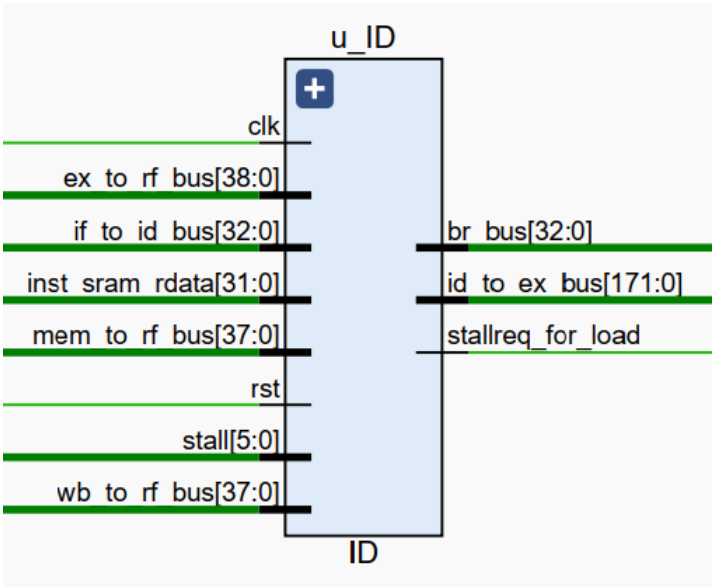
```
// if previous inst is load, check if stall is needed
wire rs_in_ex, rt_in_ex;
assign rs_in_ex = (ex_rf_we && ex_rf_waddr == rs) ? 1'b1 : 1'b0;
assign rt_in_ex = (ex_rf_we && ex_rf_waddr == rt) ? 1'b1 : 1'b0;
assign stallreq_for_load = ex_ram_read && (rs_in_ex | rt_in_ex);
```

hilo_op:

如果指令为乘除法或 hilo 相关的指令，则将它们编码送入 id 到 ex 的总线，方便 ex 段根据指令对 hilo 寄存器进行操作。

```
// hilo_op
wire [7:0] hilo_op;
assign hilo_op = {
    inst_mfhi,  inst_mflo,  inst_mthi,  inst_mtlo,
    inst_mult,  inst_multu, inst_div,   inst_divu
};
```

2.4 结构示意图



3. EX 段

3.1 整体功能说明

EX 段负责执行，借助 alu 运算单元得到指令相关的结果。

3.2 端口、信号介绍

input	clk	时钟周期信号
input	rst	复位信号
input	stall	停止信号
input	id_to_ex_bus	从 id 到 ex 的总线
output	ex_to_mem_bus	从 ex 到 mem 的总线
output	ex_to_rf_bus	从 ex 写回 id 的总线，用来解决数据相关
output	stallreq_for_ex	向 CTRL 模块告知 ex 是否需要暂停
output	data_sram_en	是否需要操作内存
output	data_sram_wen	内存的写使能
output	data_sram_addr	需要写入内存的地址
output	data_sram_wdata	需要写入内存的数据

3.3 功能模块说明

数据存储:

根据 store 指令的类别和 alu 计算结果获得 data_sram_wen、处理 data_sram_wdata

```
assign data_sram_en    = data_ram_en;
assign data_sram_wen   = inst_sw ? 4'b1111 :
                         inst_sb & alu_result[1:0]==2'b00 ? 4'b0001 :
                         inst_sb & alu_result[1:0]==2'b01 ? 4'b0010 :
                         inst_sb & alu_result[1:0]==2'b10 ? 4'b0100 :
                         inst_sb & alu_result[1:0]==2'b11 ? 4'b1000 :
                         inst_sh & alu_result[1:0]==2'b00 ? 4'b0011 :
                         inst_sh & alu_result[1:0]==2'b10 ? 4'b1100 :
                         4'b0;

assign data_sram_addr  = alu_result;

assign data_sram_wdata = inst_sw ? rf_rdata2 :
                         inst_sb ? {4{rf_rdata2[ 7: 0]}} :
                         inst_sh ? {2{rf_rdata2[15: 0]}} :
                         32'b0;
```

hilo 寄存器：

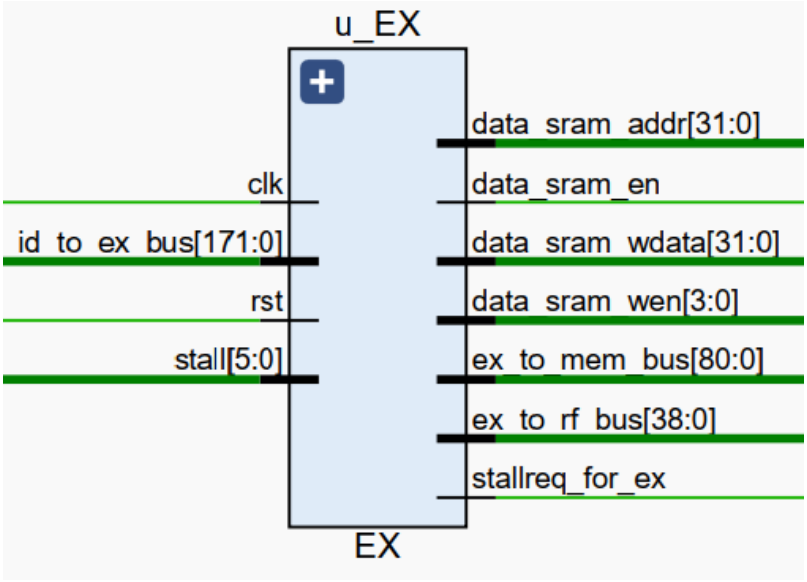
通过 id 段传来的 hilo_op 确定是否要向 hilo 寄存器写数据或读数据，图为 hilo 寄存器的内部实现。

```
reg [31:0] hi_rdata_r;
reg [31:0] lo_rdata_r;

always @(posedge clk) begin
    if (rst == 1'b1) begin
        {hi_rdata_r, lo_rdata_r} = 64'b0;
    end
    else if(we == 1'b1) begin
        hi_rdata_r <= hi_wdata;
        lo_rdata_r <= lo_wdata;
    end
end

assign hi_rdata = hi_rdata_r;
assign lo_rdata = lo_rdata_r;
```

Stallreq_for_ex 实现：

除法导致的延迟已由模板代码实现，只需要把 stall 信号传给 CTRL 模块进行下一步处理即可。

3.4 结构示意图



4. MEM 段

4.1 整体功能说明

Mem 段从内存接收数据，判断是使用从内存接收的数据还是从 ex 段传来的数据。

4.2 端口、信号介绍

input	clk	时钟周期信号
input	rst	复位信号
input	stall	停止信号
input	ex_to_mem_bus	从 ex 到 mem 的总线
input	data_sram_rdata	从内存读入的数据
output	mem_to_wb_bus	从 mem 到 wb 的总线
output	mem_to_rf_bus	从 mem 写回 id 的总线，用来解决数据相关

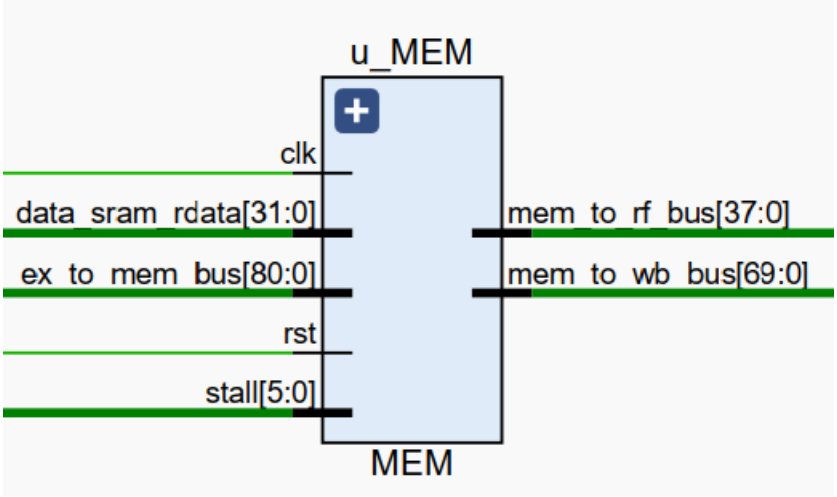
4.3 功能模块说明

Load 指令：

通过前段传来的 mem_op 解码出具体是哪种 load 指令，然后根据 ex_result 处理从内存读到的 data_sram_rdata。

```
//load data from memory
assign mem_result = inst_lw  ? data_sram_rdata :
                    inst_lb  & ex_result[1:0]==2'b00 ? {{24{data_sram_rdata[ 7]}}, data_sram_rdata[ 7: 0]} :
                    inst_lb  & ex_result[1:0]==2'b01 ? {{24{data_sram_rdata[15]}}, data_sram_rdata[15: 8]} :
                    inst_lb  & ex_result[1:0]==2'b10 ? {{24{data_sram_rdata[23]}}, data_sram_rdata[23:16]} :
                    inst_lb  & ex_result[1:0]==2'b11 ? {{24{data_sram_rdata[31]}}, data_sram_rdata[31:24]} :
                    inst_lbu & ex_result[1:0]==2'b00 ? {{24{1'b0}}, data_sram_rdata[ 7: 0]} :
                    inst_lbu & ex_result[1:0]==2'b01 ? {{24{1'b0}}, data_sram_rdata[15: 8]} :
                    inst_lbu & ex_result[1:0]==2'b10 ? {{24{1'b0}}, data_sram_rdata[23:16]} :
                    inst_lbu & ex_result[1:0]==2'b11 ? {{24{1'b0}}, data_sram_rdata[31:24]} :
                    inst_lh  & ex_result[1:0]==2'b00 ? {{16{data_sram_rdata[15]}}, data_sram_rdata[15: 0]} :
                    inst_lh  & ex_result[1:0]==2'b10 ? {{16{data_sram_rdata[31]}}, data_sram_rdata[31:16]} :
                    inst_lhu & ex_result[1:0]==2'b00 ? {{16{1'b0}}, data_sram_rdata[15: 0]} :
                    inst_lhu & ex_result[1:0]==2'b10 ? {{16{1'b0}}, data_sram_rdata[31:16]} :
                    32'b0;

assign rf_wdata = sel_rf_res ? mem_result : ex_result;
```

4.4 结构示意图



5. WB 段

5.1 整体功能说明

负责将数据写回

5.2 端口、信号介绍

input	clk	时钟周期信号
input	rst	复位信号
input	stall	停止信号
input	mem_to_wb_bus	从 mem 到 wb 的总线
output	mem_to_rf_bus	从 mem 写回 id 的总线，用来解决数据相关
output	debug_wb_pc	用来检测当前 pc 值是否正确
output	debug_wb_rf_wen	用来检测 wen 值是否正确

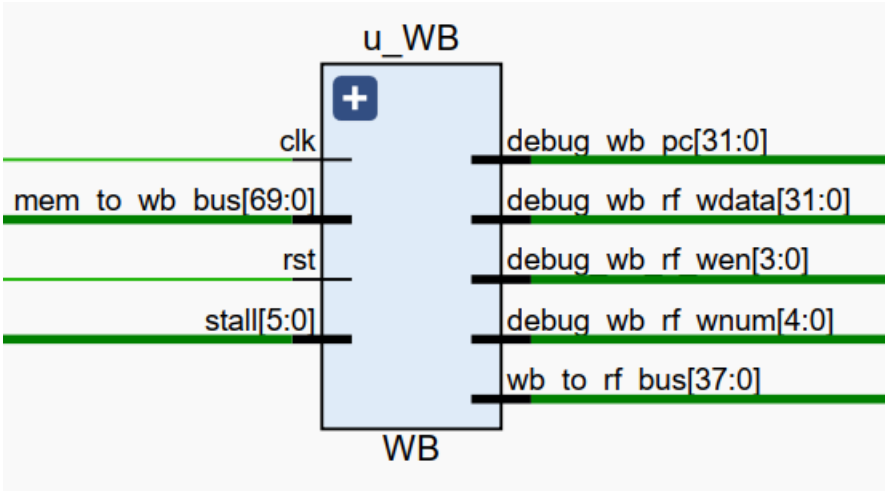
output	debug_wb_rf_wnum	用来检测地址值是否正确
output	debug_wb_rf_wdata	用来检测写入的数据是否正确

5.3 功能模块说明

Wb 段数据相关实现：

```
assign wb_to_rf_bus = {
    rf_we,
    rf_waddr,
    rf_wdata
};
```

5.4 结构示意图



三、 组员实验感受、改进意见

李易阳：

通过本次实验我充分理解了我们所学的流水线相关的理论知识，之前在课堂上学习时，更多是背下概念加上简单的理解，而我做完该实验之后对于很多知识已经有了形象地理解，完全了解了其出现的原因，也可以对该章的知识进行进一步的理解和消化。在做实验的过程中，我们遍历了每一个指令，对于这些指令在流水线中每一阶段所进行的操作都有了十分详细的跟踪和了解。这在我们日常书本学习的基础上进一步扩展了我们的知识面，在课堂上，受限于课程时长，我们无法对于具体的指令进行跟踪，但是在实验时，我们对每一条指令都做了详细的分析，并且亲手搭建了它们的通路，同时通过分析波形图解决相关问题，让我们更透彻地了解了数据通路的作用，对于我们的知识理解有很大的帮助。

初美汐：

1）已经能够基本熟练掌握 verilog 硬件设计及仿真操作流程，更加了解了对 verilog 的使用，学会了 verilog 的一些高级用法，学会了使用一些快捷键，大大提高了实验效率。

2） 对利用 FPGA 实验箱进行硬件验证的步骤也比较清晰了。这几次实验让我深化了对存储器、运算器、PC 及 AR 寄存器等的理解，而不是仅仅将知识禁锢在书本的抽象化描述中。通过设计电路，波形仿真，进行硬件测试，我们能够站在机器的角度去一步步地思考指令、微命令、数据接受计算机处理的工作方式。更重要的是，本次实验锻炼了我的实际动手能力，也培养了我认真负责、耐心细致的科学严谨态度。更加熟悉了核心指令的功能，初步掌握对不同指令的一些实现方式。

3） 加深了对课程所学内容的理解，对 CPU 设计的整个流程更加熟悉，对设计的一些细节也理解得更加透彻。

钟青霖：

计算机系统这次实验部分对我有很大的帮助，实验内容上强化了对于计算机系统课程部分内容的理解。在实验要求的推动下还学习了 Verilog 的相关知识以及基础的 linux。同时这次的小组作业与我往次的小组分工也有所不同，组长主要承担起了项目进度的担子，而我们则专注于各自跟着实验要求来开展各自地实验来学习。组长会及时为我们讲解他所做的新的进展，并为我们解答我们当前所面对的问题。整个实验流程下来算是一个相对比较耗时耗力的工作，但对于知识储备以及个人能力没有太多额外的要求，无论从实验内容的设定上，以及老师与学长的认真负责都足以可见实验部分是十分照顾我们的，也十分感谢老师为了我们所额外作出的辛苦付出，感谢老师和助教学长对我们每一组每一位的关照。

四、 参考资料

1. 《自己动手做 cpu》– 雷思磊
2. 《A03_“系统能力培养大赛”MIPS 指令系统规范_v1.01》
3. 《A07_vivado 使用说明_v1.00》
4. 《A09_CPU 仿真调试说明_v1.00》
5. 《A11_Trace 比对机制使用说明_v1.00》