实验 3 报告

学号：2017K8009922027
姓名：张磊
箱子号：39

一、实验任务（10%）

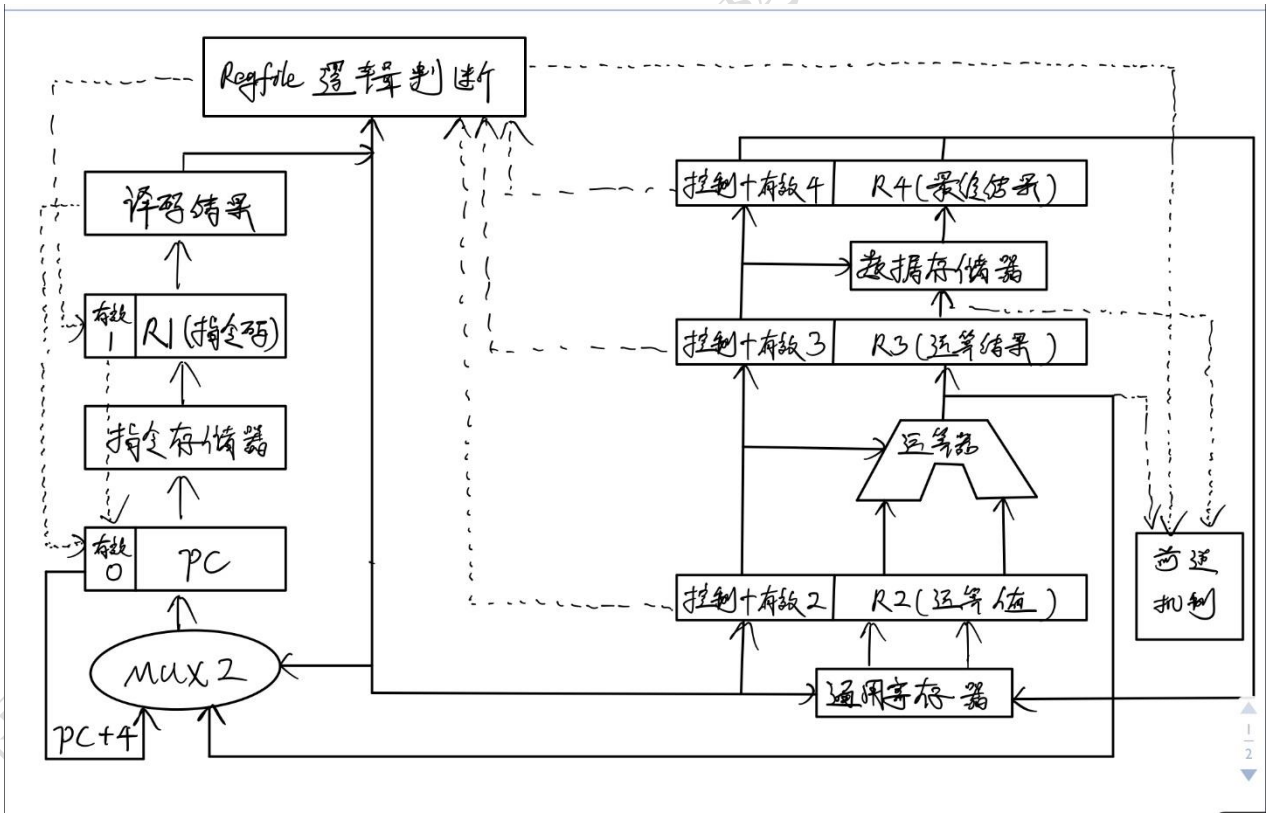
任务一：通过阅读代码和作图，明确各个模块的功能和各个模块间的关系，理解流水线的工作原理；

任务二：通过 debugg 的过程，熟悉 CPU 实验的开发环境，增强对 CPU 设计出错类型的识别和调试能力；

二、实验设计（40%）

（一）总体设计思路

数据流从 PC 到 IF 阶段，再到 ID 阶段，再到 EX 阶段，再到 MEM 阶段，最后到 WB 阶段由总线进行传输，控制逻辑又译码结果及运算结果决定；



（二）重要模块 1 设计：ALU 模块

ALU 用于完成几乎所有的 CPU 内部的运算操作；

1、工作原理

通过输入两个操作数和操作类型，运算得出最终结果，再由外部进一步逻辑判断下一步的动作；

2、接口定义

每部分的接口是什么。如果写报告的时间充裕，可以以表格形式列出；如果时间仓促，该节可以一笔带过。

名称	方向	位宽	功能描述
Alu_op	IN	12	控制 alu 的操作类型
Alu_src1	IN	32	操作数 1
Alu_src2	IN	32	操作数 2
Alu_result	OUT	32	运算结果

3、功能描述

通过 ALU_OP 判断运算逻辑，由操作数作出对应的运算，最后输出运算结果；

（三）重要模块 2 设计：regfile 模块

Regfile 中的 32 个 32 位寄存器用于存放大部分 cpu 操作的数据；

1、工作原理

在 32 位机器上采用 32 位宽的寄存器，位宽合理，32 个寄存器即可满足需求，个数合理；

2、接口定义

每部分的接口是什么。如果写报告的时间充裕，可以以表格形式列出；如果时间仓促，该节可以一笔带过。

名称	方向	位宽	功能描述
Clk	IN	1	时钟信号
Raddr1	IN	5	读地址 1
Raddr2	IN	5	读地址 2
Rdata1	OUT	32	读数据 1
Rdata2	OUT	32	读数据 2
We	IN	1	写使能
Waddr	IN	5	写地址
Wdata	IN	32	写数据

3、功能描述

根据输入的读地址异步返回读数据，根据写使能，同步写入数据，0 号寄存器永远为 0。

（四）重要模块 3 设计：ID 模块

此模块完成对指令的译码操作，决定了下一步各块的控制逻辑；

1、工作原理

数据采用总线传输，效率较高，译码逻辑按位操作，效率较高；

2、接口定义

每部分的接口是什么。如果写报告的时间充裕，可以以表格形式列出；如果时间仓促，该节可以一笔带过。

名称	方向	位宽	功能描述
Clk	IN	1	时钟信号
Reset	IN	1	复位信号
Es_allowin	IN	1	握手信号
Ds_allowin	OUT	1	握手信号
Fs_to_ds_valid	IN	1	握手信号
fs_to_ds_bus	IN	64	Fs 传到 ds 的数据
ds_to_es_valid	OUT	1	握手信号
ds_to_es_bus	OUT	136	Ds 传到 es 的数据
br_bus	OUT	33	ds 传回 fs 的数据（跳转）
ws_to_rf_bus	IN	38	Wb 到寄存器堆的数据

3、功能描述

译码逻辑由一位逻辑运算完成；同时完成分支跳转的判断；并完成寄存器写回的任务；

三、实验过程（50%）

（一）实验流水账

记录哪一天，几点到几点，做了什么事，结果如何。事情不要展开来写。

1. 2019 年 9 月 15 日 13:30-14:20 完成了虚拟机环境配置和 obj 文件的编译；
2. 2019 年 9 月 15 日 14:30-15:00 完成了 cup132_gettrace 的仿真；
3. 2019 年 9 月 15 日 15:00-23:40 找到了前 4 个 bug；
4. 2019 年 9 月 16 日 9:00-11:50 找到了后 3 个 bug，上板成功；
5. 2019 年 9 月 16 日 19:30-23:30 完成任务一；

（二）错误记录

1、错误 1：握手信号错误

（1）错误现象

导致 PC 一直为 X，且仿真无法停止，也无法识别出错误；

（2）分析定位过程

从 debug_pc 开始追查导致其为 X 的变量，最后在 ID 阶段发现 id_valid 信号未被 reset，也没有被赋值；

（3）错误原因

握手信号握手失败；

（4）修正效果

说明你修正这个错误的方法，并说明它是否有效。

```
assign ds_ready_go    = 1'b1;
assign ds_allowin     = !ds_valid || ds_ready_go && es_allowin;
assign ds_to_es_valid = ds_valid && ds_ready_go;
always @(posedge clk) begin
    if (reset) begin
        ds_valid <= 1'b0;
    end
    else if (ds_allowin) begin
        ds_valid <= fs_to_ds_valid;
    end

    if (fs_to_ds_valid && ds_allowin) begin
        fs_to_ds_bus_r <= fs_to_ds_bus;
    end
end
```

添加了控制 ds_valid 信号的代码，有效：

```
[   2067 ns] Error!!!
    reference: PC = 0xbfc00000, wb_rf_wnum = 0x08, wb_rf_wdata = 0xffffffff
    mycpu    : PC = 0xbfc00000, wb_rf_wnum = 0x08, wb_rf_wdata = 0xxxxxxxX
```

开始报错；

2、错误 2：信号为 Z

（1）错误现象

导致 wb_rf_wdata 信号为 X；

```
[   2067 ns] Error!!!
    reference: PC = 0xbfc00000, wb_rf_wnum = 0x08, wb_rf_wdata = 0xffffffff
    mycpu    : PC = 0xbfc00000, wb_rf_wnum = 0x08, wb_rf_wdata = 0xxxxxxxX
------------------------------------------------------------
```

（2）分析定位过程

找到引起 wb_rf_wdata 变化的变量，一直找到 ID.v 代码中，发现 load_op 悬空，没有赋值；

（3）错误原因

Load_op 没有赋值；

（4）修正效果

给 load_op 赋值，有效（虽然还有问题）：

```
[   2067 ns] Error!!!
    reference: PC = 0xbfc00000, wb_rf_wnum = 0x08, wb_rf_wdata = 0xffffffff
    mycpu    : PC = 0xbfc00000, wb_rf_wnum = 0x08, wb_rf_wdata = 0xfffffffe
------------------------------------------------------------
```

3、错误 3：alu 调用变量名用错

（1）错误现象

```
alu u_alu(
    .alu_op     (es_alu_op    ),
    .alu_src1   (es_alu_src2  ),
    .alu_src2   (es_alu_src2  ),
    .alu_result (es_alu_result)
    );
```

导致两个操作数相加结果不对；

（2）分析定位过程

执行 ADD 操作，结果不对，于是找到操作数和 ALU，发现，ALU 调用出错；

（3）错误原因

ALU 调用出错

（4）修正效果

改正变量名；有效；

4、错误 4：位宽错误

（1）错误现象

位宽错误，导致 br_taken 一直为 0；

（2）分析定位过程

由于 pc 跨度较大，可判断为进行了跳转操作，但并没有跳转，因此，应该是跳转信号错误，最后在 ID 阶段的代码中找到了原因，是 br_bus 的位宽错误，导致 br_taken 一直为 0；

```
assign br_bus       = {br_taken,br_target};
```

（3）错误原因

Br_bus 位宽为 32，而 br_taken 和 br_target 位宽合在一起是 33，所以赋值漏掉了 br_taken；

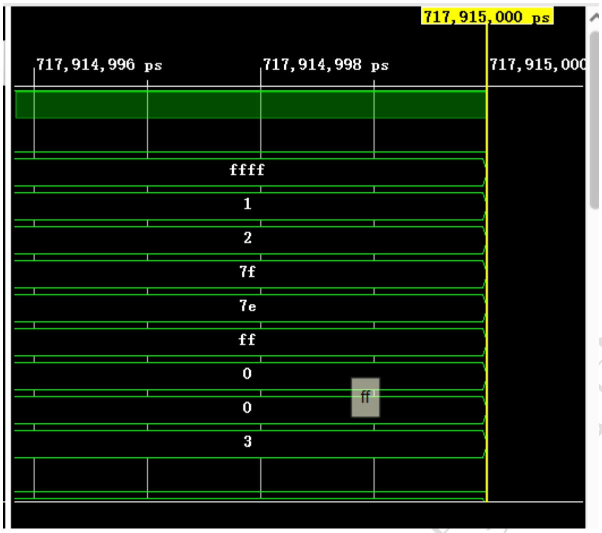
（4）修正效果

将 br_bus 位宽修改为 33 后，成功，向后运行了许多步；

5、错误 5：组合环

（1）错误现象

波形停止不前；



（2）分析定位过程

由于上学期组成原理时我在 ALU 部分写出了组合环，所以这次我直接来到 ALU 中查找，果然发现，alu_result 和 or_result 形成了组合环；

（3）错误原因

Alu 中 or_result 和 alu_result 形成组合环；

```
assign and_result = alu_src1 & alu_src2;
assign or_result  = alu_src1 | alu_src2 | alu_result;
assign nor_result = ~or_result;
```

```
assign alu_result = ({32{op_add|op_sub}} & add_sub_result)
                  | ({32{op_slt       }} & slt_result)
                  | ({32{op_sltu      }} & sltu_result)
                  | ({32{op_and       }} & and_result)
                  | ({32{op_nor       }} & nor_result)
                  | ({32{op_or        }} & or_result)
                  | ({32{op_xor       }} & xor_result)
                  | ({32{op_lui       }} & lui_result)
                  | ({32{op_sll       }} & sll_result)
                  | ({32{op_srl|op_sra}} & sr_result);
```

（4）修正效果

删去 or_result 赋值中的右边的 alu_result;有效，波形继续前进；

6、错误 6：位宽错误

（1）错误现象

导致 ALU 计算结果正确但输出结果错误；

（2）分析定位过程

由于是 wdata 报错，所以一路查找所有可能引起 wdata 变化的值，最终在 alu 里边发现，sr 结果赋值时位宽错误；

（3）错误原因

Alu 输出结果位宽错误；

```
assign sr_result   = sr64_result[30:0];
```

（4）修正效果

改正位宽；有效，仿真通过？？

```
==============================================================
Test end!
----PASS!!!
$finish called at time : 1728025 ns : File "D:/Vivado/UCAS_CD
```

7、错误 7：阻塞赋值用成非阻塞赋值

（1）错误现象

阅读代码时发现，因此并未观察到现象；

（2）分析定位过程

因为该部分赋值出现在时序逻辑中，所以只能使用阻塞赋值；

（3）错误原因

在时序逻辑中使用非阻塞赋值；

```
always @(posedge clk) begin
    if (reset) begin
        ms_valid <= 1'b0;
    end
    else if (ms_allowin) begin
        ms_valid <= es_to_ms_valid;
    end

    if (es_to_ms_valid && ms_allowin) begin
        es_to_ms_bus_r  = es_to_ms_bus;
    end
end
```

（4）修正效果

将非阻塞赋值改为阻塞赋值；

```
always @(posedge clk) begin
    if (reset) begin
        ms_valid <= 1'b0;
    end
    else if (ms_allowin) begin
        ms_valid <= es_to_ms_valid;
    end

    if (es_to_ms_valid && ms_allowin) begin
        es_to_ms_bus_r  <= es_to_ms_bus;
    end
end
```

四、实验总结（可选）

1. 本次实验，熟悉了 CPU 实验的开发环境，再次加深了判断 bug 类型和 bug 调试的能力；
2. 对经典的 MIPS CPU 5 级流水有了初步的了解，但还是有些模糊，课后会再自主查阅资料学习一下；
3. 本次实验由于一开始比较糊涂，所以调前几个 Bug 时花费了较长时间，之后几个 bug 就比较快；
4. 对于画设计图还是思绪比较混乱，会抓紧时间多学习锻炼画图能力；