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 2002

组长：梁吉涛　　20206373

组员：李梓豪　　20206391

组员：高　杨　　20206376

报告日期：2022.12.21

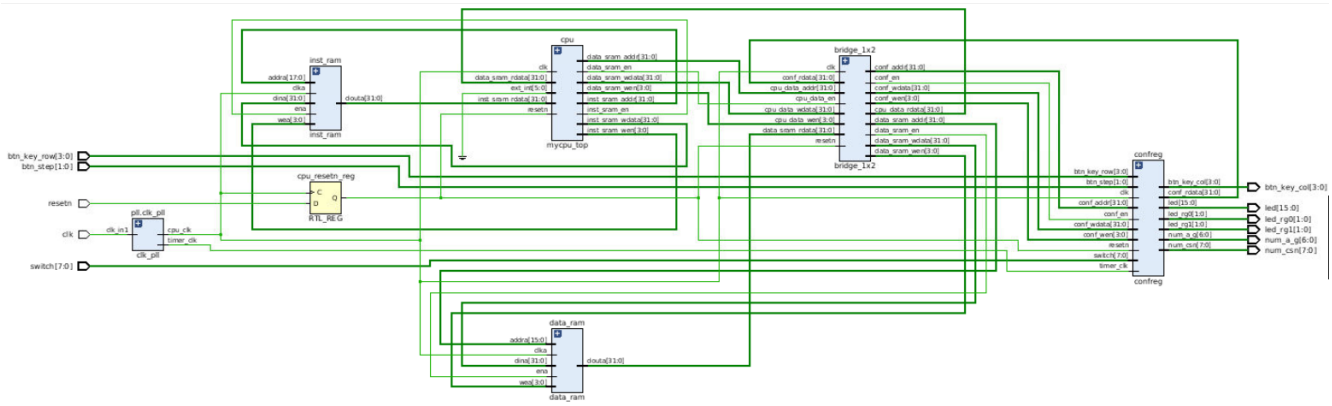
目录

一. 分工

梁吉涛：50%　　高杨：　25%　　李梓豪：25%

二. 总体设计

1. 连线图



2. 完成的指令数

算数运算指令：ADD， ADDI, ADDU, ADDIU, SUB, SUBU, SLT, SLTI, SLTU, SLTIU, DIV, DIVU, MULT, MULU 共 14 条

逻辑运算指令： AND, ANDI, LUI, NOR, OR, ORI, XOR, XORI 共八条

位移指令：SLLV, SLL, SRAV, SRA, SRLV, SRL 共六条

分支跳转指令：BEQ, BNE, BGEZ, BGTZ, BLEZ, BLTZ， BGEZAL， BLTZAL， J, JAL, JR, JALR 共 12 条

数据移动指令：MFHI, MFLO, MTHI, MTLO 共四条

访存指令：LB, LBU, LH, LHU, LW, SB, SH, SW 共八条

总计 52 条指令

3. 程序运行环境：

linux 操作系统

4. 使用工具：

vivado，visual code

三.单个流水段说明

1.IF 段

1.1 整体功能

IF 段负责以程序计数器 PC 中的内容作为地址，从存储器中取出指令并放入指令寄存器，即取指，并将取到的指令传到 ID 段。

1.2 输入输出端口与信号

传入：

clk：传入时钟周期信号

rst：复位信号，负责初始化各种数据

stall：暂停信号，用于暂停流水线

br_bus：来自 ID 段的输入，跳转指令的相关信号

传出：

if_id_to_bus：打包 IF 段信息传到 ID 段的总线

inst_sram_wen：存储器写入的使能信号

inst_sram_wen：存储器读取的使能信号

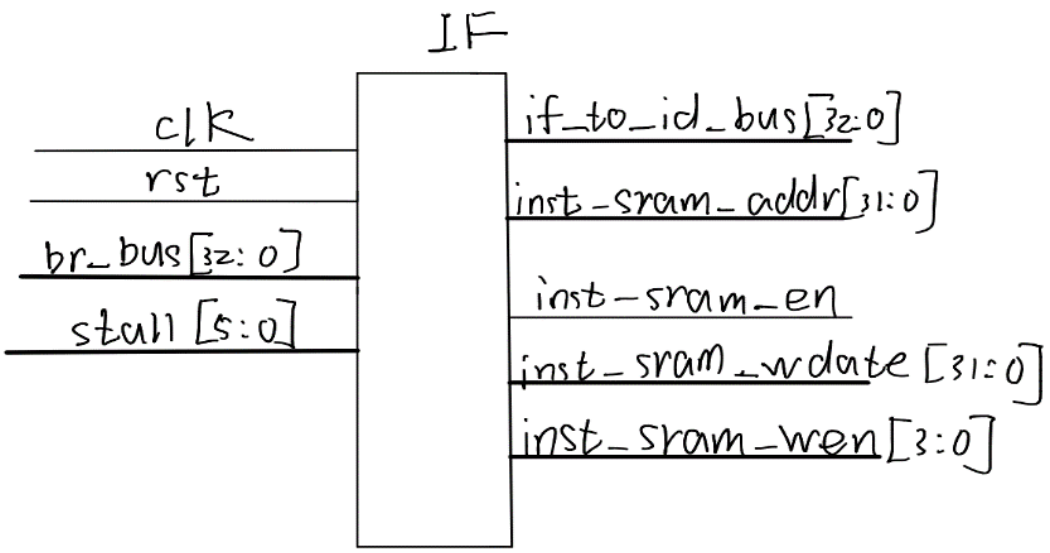
inst_sram_addr：存储器地址

inst_sram_wdata：存储器写入内存数据

1.3 功能模块介绍

br_bus：从 ID 段传来的跳转指令，判断是否需要跳转。如果需要，就使用里面的跳转地址，否则就 PC+4；获得 PC 指令；获得使能信号 ce。

1.4 结构示意图



2. ID 段

2.1 整体功能

ID 段的主要功能是译码，并用 IR 中的寄存器地址去访问通用寄存器组，读出所需的操作数。

2.2 输入输出端口与信号

传入：

clk：传入时钟周期信号

rst：复位信号，负责初始化各种数据

stall：暂停信号，用于暂停流水线

if_to_id_bus：IF 段传来的总线

inst_sram_rdata：要写入内存的数据

Stallreq：向 CTRL 模块传递 ID 是否要暂停的信号

ex_to_rf_bus：从 EX 段传回 ID 段的总线，用于解决数据相关

mem_to_rf_bus：从 MEM 段传回 ID 段的总线，用于解决数据相关

wb_to_rf_bus：从 WB 段传回 ID 段的总线，用于解决数据相关

传出：

id_to_ex_bus：打包 ID 段的信息传给 EX 段的总线

br_bus：传给 IF 段来进行跳转指令

2.3 功能模块介绍

ex_to_rf_bus、mem_to_rf_bus、wb_to_rf_bus：分别来自 EX 段、MEM 段、WB 段传回给 ID 段的总线，用于解决数据相关。

此处指的数据相关是 RAW 相关，前一条指令还未写入，下一条指令需要使用的情况。将上条指令的 alu 运算结果，传给下一条指令的 alu 运算前，即利用定向技术，将数据提前传给下一条指令的 alu 运算前。此外影注意的是，存在不能利用此技术暂停的 RAW 相关。若上一条指令是访存指令，则只能在 MEM 段得到结果，传给下一条指令，此时不能用定向技术，而是要插入暂停。

数据相关解决方法：分别将来自 EX 段、MEM 段、WB 段的存储器地址、存储器数据（EX 段是 alu 运算结果）、使能信号、hilo 寄存器的使能信号和数据传给 ID 段。对于非 Hilo 寄存器的数据相关，按照 EX 段、MEM 段、WB 段的传回顺序，依次判断传回来的使能信号，若是 1，则判断写回的地址是否和 rs 和 rt 地址相同，若相同，则直使用写回的数据，否则依旧用当前的数据。对于 hilo 寄存器，判断 hi 和 lo 的使能信号，若为 1，则直接使用写回的 hi 寄存器数据或者 lo 寄存器数据，否侧依旧使用当前 hilo 寄存器的数据。

sel_alu_src1[1]：用来判断是否需要从寄存器中读取指令。

sel_alu_src1[2]：用来判断是否需要进行 sa 的无符号扩展。

sel_alu_src2[0]：用来判断该指令是否需要从 rt 读取数据。

sel_alu_src2[1] ：用来判断是否要对立即数进行有符号扩展 。

sel_alu_src2[2] ：用来判断是否要 PC +8。

sel_alu_src2[3]： 用来判断是否要对立即数进行无符号扩展。

alu_op：判断传入 alu 模块的各项操作，若存在某项操作，将这项操作对应置为 1。

data_ram_en：判断是否是访存指令。

mem_op 里面包含八条访存指令。当存在相应的指令时，把对应的线路置为 1。

data_ram_wen：判断是否是要写入内存的指令。

sel_rf_res：判断是否是需要从内存中取数据的指令。

stallreq_for_load：判断是否需要插入暂停，需要则置为 1。

rf_we：写入使能，判断指令是否有写入寄存器的需要，若有置为 1。

sel_rf_dst[0]：判断指令是否有写入 rd 寄存器的需要，若有置为 1。

sel_rf_dst[1]：判断指令是否有写入 rt 寄存器的需要，若有置为 1。

sel_rf_dst[2]：判断指令是否有写入 31 号寄存器的需要，若有置为 1。

rf_waddr：计算要写入的寄存器的地址。

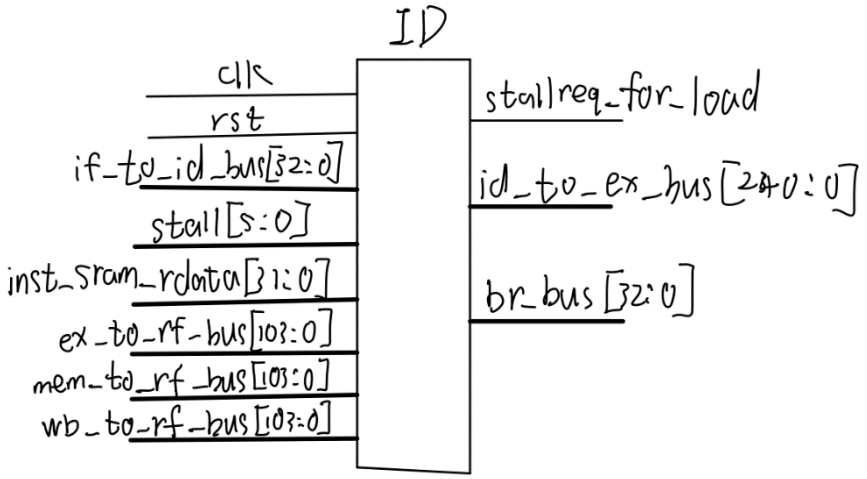
br_e：判断是要跳转指令还是顺序执行指令。

br_addr：跳转之后下一个要执行的指令。

br_bus：将上面得到的 br_e 和 br_addr 打包传给 IF 段，用来决定 IF 段下一个指令是按顺序取还是取跳转的指令。

id_to_ex_bus 将 ID 段数据打包传给 EX 段的总线。

2.4 结构示意图



3. EX 段

3.1 整体功能

EX 负责执行，对 ID 解析的指令进行运算

3.3 功能模块介绍

clk 时钟周期 ；rst 复位信号，负责初始化各项数据 ；停止信号，stall 负责暂停流水线 ;id_to_ex_bus 从 ID 段传到 EX 段的总线 ；

ex_to_mem_buse 从 EX 到 MEM 段的总线 ；data_sram_en 是否对内存有操作；data_sram_wen 写入内存的使 能；data_sram_addr 要写入的内存的地址；data_sram_wdata 写入内存的数据；ex_to_rf_bus EX 段到 ID 段的总线；stallreq_for_ex 判断是否需要暂停。

3.3 功能模块说明

加法器 alu 负责通过 alu_op 判断要进行的是哪种运算，然后对 alu_src1 和 alu_src2 两个操作数进行运算得出结果 alu_result

乘法器 mul 有多个本周期 clk，先判断是否是有符号乘法 mul_signed，当收到乘法开始信号 start_1 开始运算，然后对 1_ina 和 2_ina 两个操作数进行运算，收到乘法结束信号 ready_to，根据结束信号判断运算运算是否完成，没有完成要对整个流水段暂停，等运算结束再进行其他指令，得到结果 result

alu_src1 用来判断加法器的第一个操作数是从 regfile 中读出的 rs 寄存器中的数，还是计算出的用于跳转指令的数以及立即数 sa 指定移位量对寄存器 rt 的值进行逻辑左移的数

alu_src2 用来判断加法器的第二个操作数是立即数有符号扩展之后的数，还是三十二位二进制的 8，还是立即数无符号扩展之后的数，还是从 regfile 中读出的 rt 寄存器中的数

除法器 div 有多个本周期 clk，先判断是有符号除法 signed_div_i，当收到乘法开始信号 start_i 开始运算，然后对被除数 opdatal_i 和除数 opdata2_i 两个操作数进行运算，收到乘法结束信号 annul_i，根据结束信号判断运算运算是否完成，没有完成要对整个流水段暂停，等运算结束再进行其他指令，得到结果 result_to

Hi-lo 寄存器用来存放城市出发得到的数据，乘法的乘积低半部分写入 lo 高半部分写入 hi，除法的商写入 lo，余数写入 hi，hi-lo 寄存器接口：clk 时钟周期，rst

复位信号，是否写入 hilo 寄存器 we，写入 hilo 寄存器地址 hilo，写入 hilo 寄存器数据 hi，lo

Always 用来赋值，当复位信号是 1，需要复位，将 id 到 ex 总线和赋初值 0，当复位信号 0，指令开始运行，将 id 到 ex 的总线用 id_to_ex_bus 赋值

imm_sign_extend 对立即数进行有符号扩展

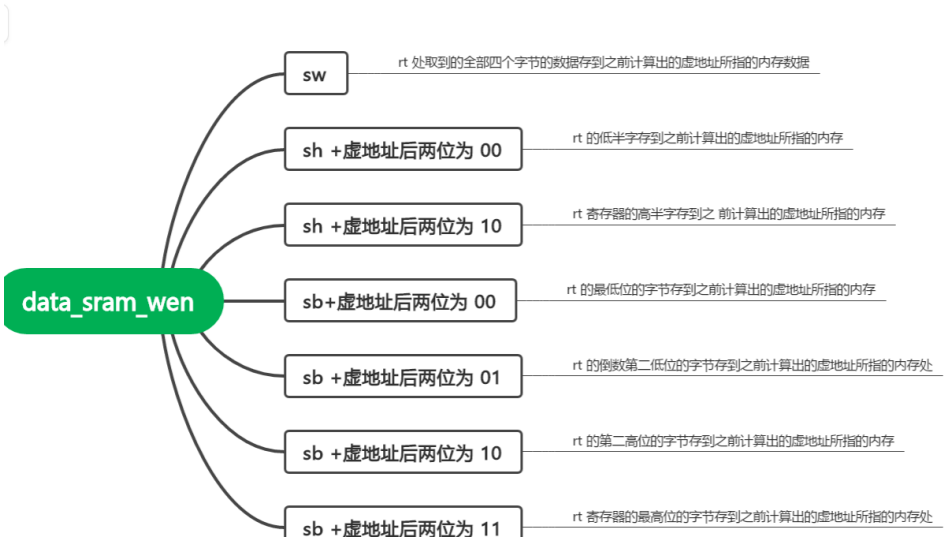
imm_zero_extend 对立即数进行无符号扩展

sa_zero_extend 对 sa 进行无符号扩展

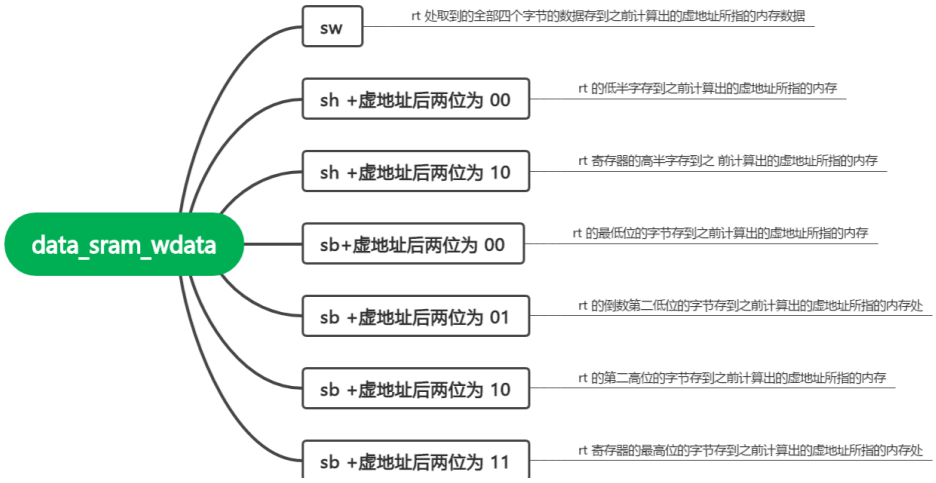
data_sram_addr 判断是否用到了虚拟内存，用到了写入 base 寄存器的值加上符号扩展后的立即数 offset，否则写入 0

new_data_sram_wen 从 ID 段传到 ex 的 data_ram_wen 用来判断传进来的指令是 sb 还是 sw

data_sram_wen 根据指令判断写入内存的位置



data_sram_wdata 根据指令写入内存中的数据



new_lb_lw_lh 将从内存中取出来的数据存入到 rt 寄存器中

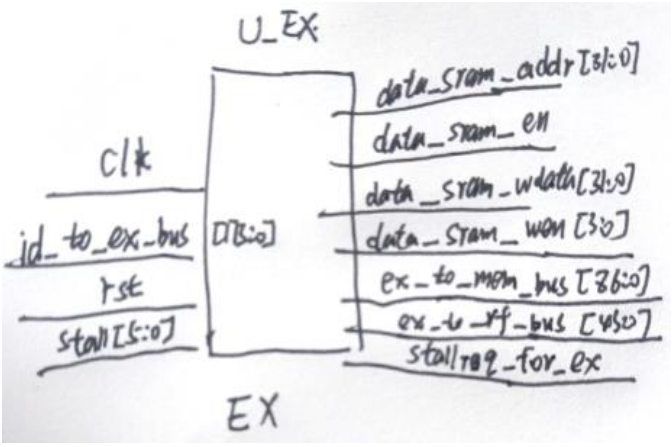
stallreq_for_ex 用于判断当前阶段是否需要暂停（根据乘法器和除法器模块中的结束信号判断，即当结束信号为 1 时表示乘法和除法已经执行完成了，可以不用再暂停了；当结束为 0 时表示乘法和除法还没有完成，还需要继续暂停）

hilo_data 写入 hilo 寄存器的数据，除法+允许写入就将除法结果 div_result 写入 hilo_data ；乘法+允许写入将乘法结果 mul_result 给 hilo_data 赋值；mthi 将 rs 寄存器中的值经过无符号扩展后赋值给给 hilo_data；mtlo 将 rs 寄存器中的值放在高位，低位赋 0 后，赋值给 hilo_data

ex_result EX 段计算出的结果，如果为 lw 类的指令，那么将此时刻地址下的内存数据赋给 ex_result；如果 mfhi 指令，那么将从 hilo 寄存器模块 中读出的 hi 寄存器中的值赋给 ex_result；如果 mflo 指令，那么将从 hilo 寄 存器模块中读出的 lo 寄存器中的值赋给 ex_result；如果都不是，就将从加法 器 alu 模块中计算出的结果赋值给 ex_result

ex_to_rf_bus 负责将 ex 段的指令码，写入使能，写入地址和写入数据返 回给 ID 段的数据通路，用来判断是否存在读后写等数据相关

3.4 结构示意图



4.MEM 段

4.1 整体功能

读入内存数据， 判断从内存传入还是从 ex 传入，并传给 wb

4.2 端口，信号介绍

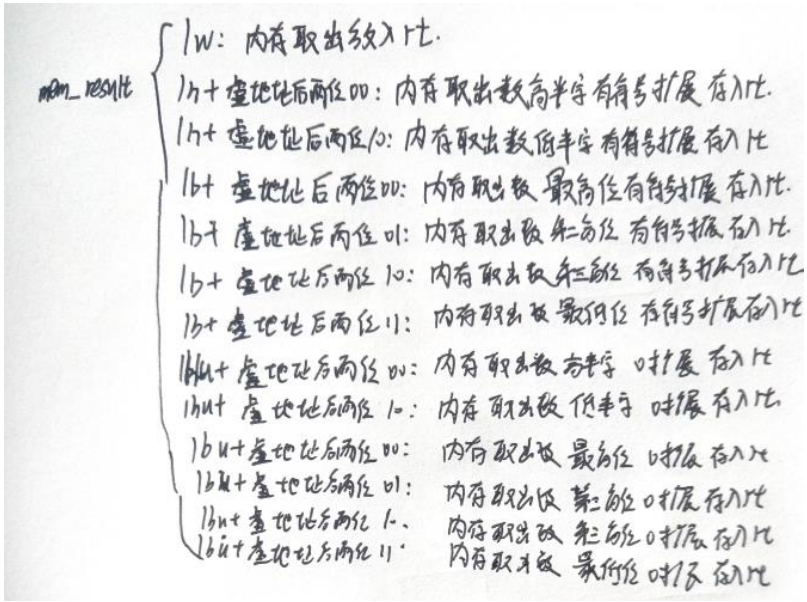
Clk 传入始终周期；rst 复位信号；stall 暂停流水线；ex_to_mem_bus 从 ex 到 mem 总线；data_sram_rdata 从内存读入数据，mem_to_wb_bus 从 mem 到 wb 的总线；mem_to_rf_bus 从 mem 到 id 的总线

4.3 功能模块介绍

ex_to_mem_bus_r 获取从 EX 段传来的总线

always 如果复位信号为 1 或着暂停信号为 1 ，将 ex_to_mem_bus_r 赋值 为 0，否则将 ex_to_mem_bus_r 用 EX 段传来的总线赋值

mem_result 将内存中取出来的数据存入到 rt 寄存器



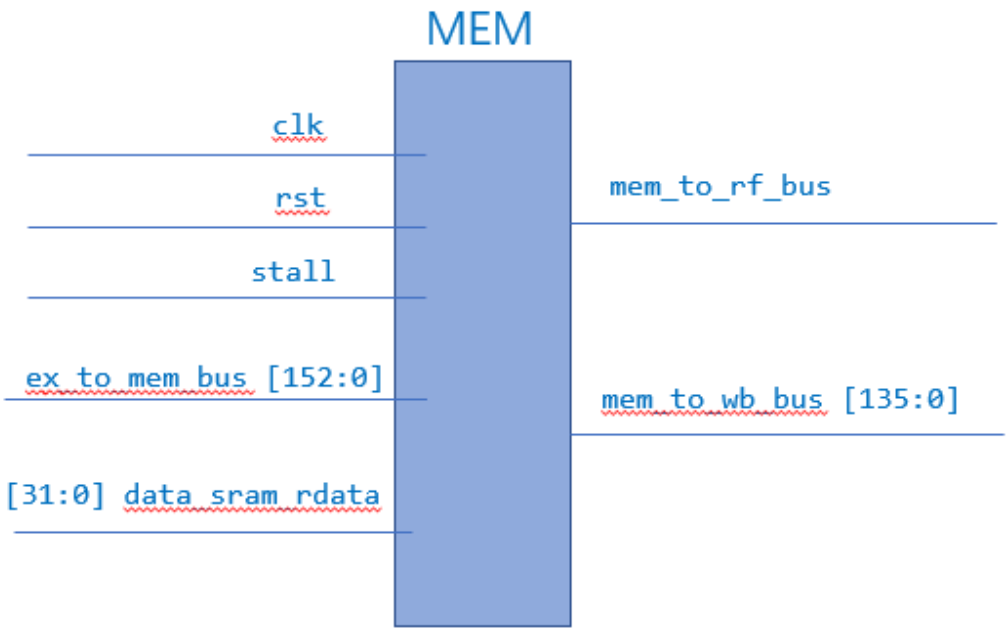
mem_inst_lw 用来判断当前指令是不是 lw 指令

mem_inst_lb 用来判断当前指令是不是 lb 或 lbu 指令

mem_inst_lh 用来判断当前指令是不是 lh 或 lhu 指令

rf_wdata 根据 sel_rf_res 进行判断是写入 MEM 段算出的数据还是从 EX 传来的数据。

4.4 结构示意图



5. WB 段

5.1 整体功能

WB 段负责将数据写回。

5.2 端口，信息介绍

Input wire:

①clk:传入时钟周期；

②rst：复位信号，负责初始化各项数据；

③stall: 停止信号，负责暂停流水线；

④mem_to_wb_bus：MEM 段到 WB 段的总线；

Ouput wire:

⑤wb_to_rf_bus：WB 段到 ID 段的总线；

⑥debug_wb_pc：当前阶段的指令；

⑦debug_wb_rf_wen：当前阶段的写入使能；

⑧debug_wb_rf_wnum：当前的写入地址

⑨debug_wb_rf_wdata：当前的写入数据。

5.3 功能模块介绍

①mem_to_wb_bus_r 负责获得从 MEM 段传来的总线。

②always 模块：

如果复位信号为 1（即处于复位状态下）或着暂停信号为 1 时，

将 mem_to_wb_bus_r 赋值为 0，否则将 mem_to_wb_bus_r 用

MEM 段传来的总线赋值。

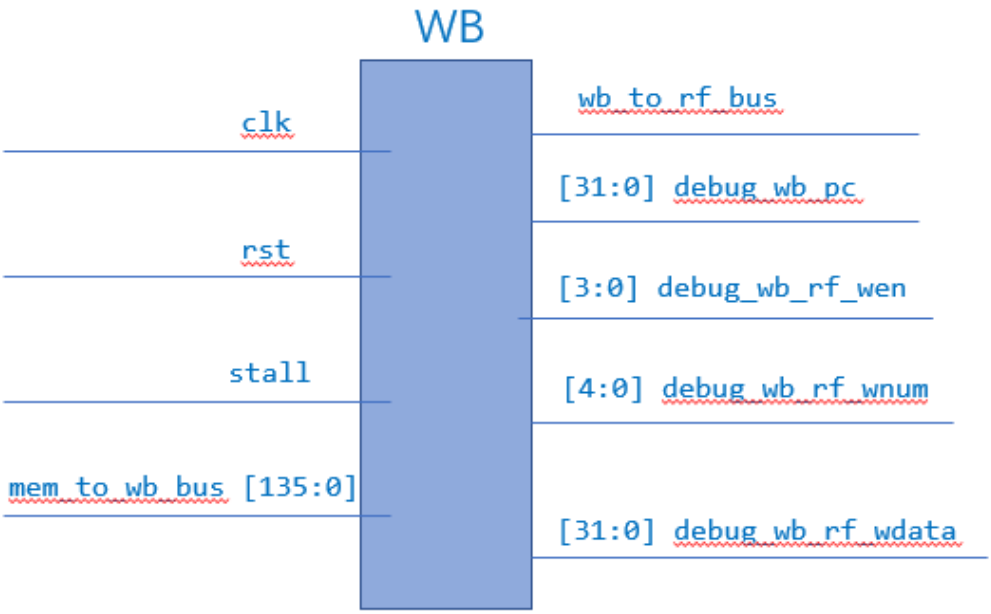
③wb_to_rf_bus ：从 WB 段传回 ID 段的总线。

④debug_wb_pc ：检测当前的 pc 值是否正确。

⑤debug_wb_rf_wnum ：检测当前的写入的地址值是否正确。

⑥debug_wb_rf_wdata ：检测当前的写入的数据是否正确。

5.4 结构示意图



四．组员实验感受，改进意见

梁吉涛实验感受：

通过这次实验，我加深了对计算机系统这门课的理解，并学习了 CPU 的一些基本原理和结构，学会如何构建线性流水的 CPU 结构。在此次实验中，我也遇到了许多问题：在做数据相关的时候，没有在顶层模块实例化传回的线路，导致出现信号为“Z”的情况；在输送线路的时候，由于线路不够宽，导致高位的“1”信号被补“0”，对比机制报错；写指令使能信号是四位，结果只置了一位，导致写进去的地址错误而报错等等一系列问题。但是通过这些问题，我

可以找到自己身上的不足之处，熟练运用并掌握了很多计算机系统这门课的知识，理论化为实践，这是这次实验课我最大的收获。

李梓豪实验感受：

① 通过对 git 使用的学习，熟练了小组协作时文件更新的高效有序处理方式；

② 通过最开始的小练习学习了 verilog 语言的一些简单语法与使用规则，虽然一开始进行实验设计时，还是有些手足无措，不知道如何通过 pc 值来寻找问题所在及指令的添加等方式，经过慢慢地尝试、一次次的实验课讲解与组员、助教的指导，开始有了上手修改的能力，也可以开始过点了。印象最深的还是在 halo 寄存器与乘除法添加后接手修改，通过对《自己动手做 cpu》一书的仔细阅读后，了解了几条数据移动指令的问题所在，解决了这里的数据相关。

③ 多多注重理论实践结合，才能掌握的更好。

高杨实验感受：

通过此次试验，自己对 cpu 的基本原理有了更加深入的理解，结合课本上 cpu 的整个流水段，学以致用，加深印象，实验过程中， 刚开始完全一无所知，从创建 git 仓库，到学习 verilog 语言，到开始用 vivado，一步步，在陌生中探索进步，所幸助教讲的详细，文件表述清楚，最终顺利完成实验，中间也有很多心酸经历尤其是 Vivado 运行出错时，根据地址找到指令的时候，要去分析那个地方出错 ，要是指令没添加还好，有些总线长度不够的就很难发现，在这个过程中，自己也是学会尝试用波形图去寻找错误，先从众多信号中找自己认为相关的信息，然后加入猜测可能出错的位置，接着具体分析，最终将问题解决，本次实验使得我们看到了自我的差距和经验的不足，以后需要勤奋的学习的同时更应多注重实际的运用。

五．参考资料

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. 《A06_vivado 安装说明_v1.00》

4. 《A07_vivado 使用说明_v1.00》

5. 《A09_CPU 仿真调试说明_v1.00》

6. 《A08_交叉编译工具链安装_v1.00