实验报告成绩：	成绩评定日期：

2021～2022 学年秋季学期

A3705060050《计算机系统》必修课

课程实验报告

班级：人工智能 1901

组长：程思睿 20195206

组员：马韬洵 20195253

组员：孙辉 20195192

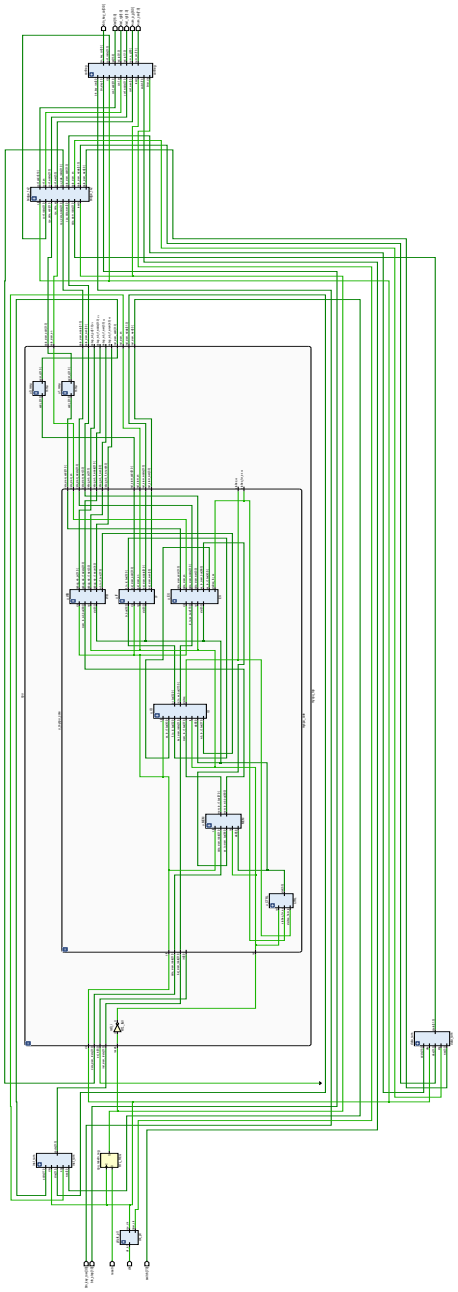
报告日期：2021.12.18

一．总体设计

1.工作量

第一贡献：程思睿；　第二贡献：马韬洵；　第三贡献：孙辉

2.连线图



3.实验总览

3.1 总共完成了 52 条指令

3.1.1 算数运算指令 14 条

（ADD，ADDI,ADDU,ADDIU,SUB,SUBU,SLT,
SLTI,SLTU,SLTIU,DIV,DIVU,MULT,MULU）

3.1.2 逻辑运算指令 8 条

(AND,ANDI,LUI,NOR,OR,ORI,XOR,XORI)

3.1.3 位移指令 6 条

(SLLV,SLL,SRAV,SRA,SRLV,SRL)

3.1.4 分支跳转指令 12 条

(BEQ,BNE,BGEZ,BGTZ,BLEZ,BLTZ,
BGEZAL,BLTZAL,J,JAL,JR,JALR)

3.1.5 数据移动指令 4 条

(MFHI,MFLO,MTHI,MTLO)

3.1.6 访存指令 8 条

(LB,LBU,LH,LHU,LW,SB,SH,SW)

3.2 程序运行环境

Windows，intel i5

3.3 实用工具

Vivado，visual code

二．单个流水段说明

1. IF

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

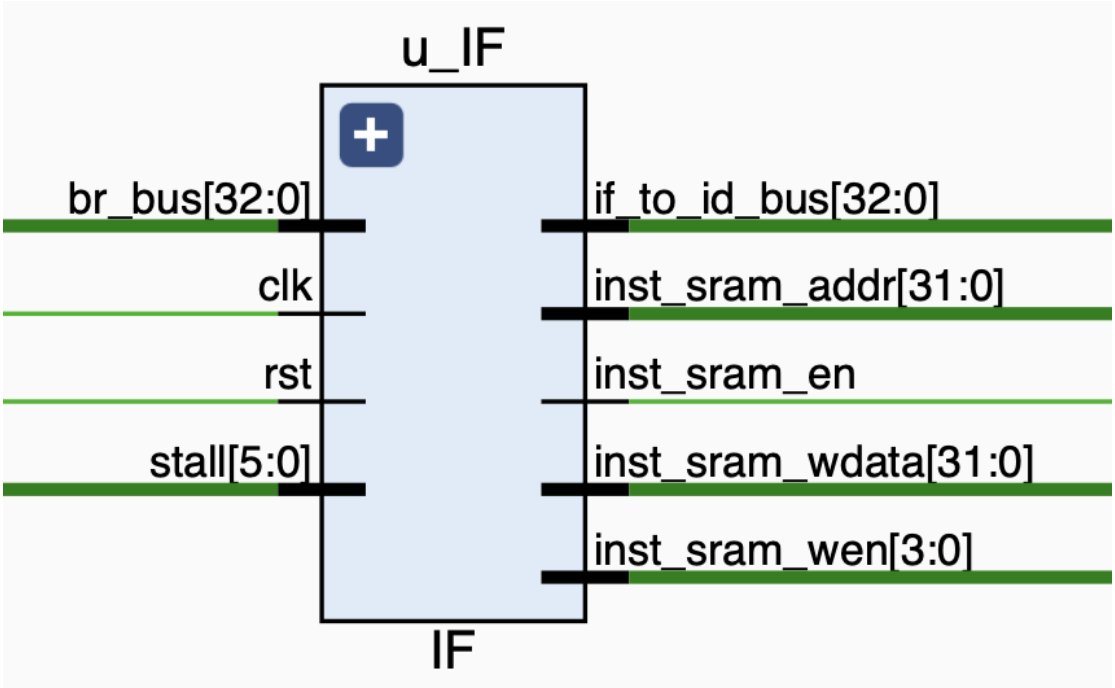
传入时钟周期 clk;复位信号，负责初始化各项数据 rst;停止信号，负责暂停流水线 stall;从 ID 段传入，存放指令跳转的相关信息 br_bus;IF 段到 ID 段的总线，

将 IF 段的数据进行打包传到 ID 段 if_to_id_bus;能不能从内存读取指令的使能 inst_sram_en;能不能向内存中写入数据的使能 inst_sram_wen;写入内存时的写入地址 inst_sram_addr;写入内存的数据 inst_sram_wdata.

1.3 功能模块说明

从 ID 段传来的跳转指令 br_bus.判断是否需要跳转，若跳转使用跳转地址，否则选择 pc+4next_pc = br_e ? br_addr : pc_reg + 32'h4;获得 pc 指令 pc_reg，获得指令执行使能 ce

1.4 结构示意图



2.ID

2.1 整体功能说明

ID 段负责译码，将 IF 段传来的指令进行拆分解析。

2.2 端口，信号介绍

传入时钟周期 clk；复位信号，负责初始化各项数据 rst；停止信号，负责暂停流水线 stall；向 CTRL 模块传递 ID 是否要暂停的信号 Stallreq；　从 IF 段获得的总线 if_to_id_bus；要写入内存的数据 inst_sram_rdata；　从 WB 段写回 ID 段的总线，用来解决数据相关 wb_to_rf_bus；　从 EX 段写回 ID 段的总线，用来解决数据相关

ex_to_rf_bus；从 MEM 段写回 ID 段的总线，用来解决数据相关

mem_to_rf_bus；ID 段到 EX 段的总线，将 IF 段的数据进行打包

id_to_ex_bus；传给 IF 段用来进行跳转指令 br_bus。

2.3 功能模块说明

从 mem 段传到 ID 段用来解决数据相关的总线 mem_to_rf_bus;

从 EX 段传到 ID 段用来解决数据相关的总线 ex_to_rf_bus;

用来解决数据相关，确保从寄存器中读出的数据是上一步写入的，防止读后写的现象发生（如果从 IF 段传入的指令不生效则将从 rs，rt 寄存器读出的数据置为 0，如果 ex 段需要写入且 ex 段写入的数据的地址是 rs 寄存器的地址，那么让 rs 寄存器中的数据直接等于 ex 段写入的数据，如果 mem 段需要写入且 mem 段要写入的地址与 rs 寄存器的地址相同，那么让 rs 寄存器中的数据直接等于 mem 段要写入的数据，不用再等一段时间才能写入，用来解决读后写等数据相关的情况，同理，rt 寄存器处理从 EX 段和 MEM 段传来的数据时也用类似处理 rs 寄存器的方法来处理数据相关问题）always 中用来赋值，当复位信号为 1 时（即需要复位）则给总线和停止指令赋初值 0，当复位信号为 0 时（即指令开始运行）如果需要停止则将总线用从 IF 段传来的 if_to_id_bus 赋值，并且将停止使能 id_stop 置为 0.如果不停止则将总线用从 IF 段传来的 if_to_id_bus 赋值，并且将停止使能 id_stop 置为 1.从 WB 段传到 ID 段用来写入寄存器的写入使能，要写入的寄存器的地址以及要写入的数据的总线 wb_to_rf_bus;寄存器，用来读取数据和写入并保存数据，其中本次所设计的寄存器包括两个可以用来读取的寄存器 rs 和 rt 以及一个用来写入数据的接口（即该寄存器可以同时支持两个读和一个写）。其中寄存器的接口的作用分别是传入时钟周期 clk；rs 寄存器的地址 raddr1；从 rs 寄存器读出的数据 rdata1；rt 寄存器的地址 raddr2；从 rt 寄存器读出的数据 rdata2；是否写入寄存器的使能 we；将要写入的寄存器的地址 waddr；要写入寄存器的数据 wdata。用来定义一系列操作，

例：wire inst_ori, inst_lui, inst_addiu;用来判断传进来的指令属于那一种操作，

例：assign inst_addi = op_d[6'b00_1000];sel_alu_src1[0]用来判断该指令是否需要从 rs 读取数据，如果需要则将指令以或的形式添加再后面。

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]是否要对立即数进行有符号扩展

sel_alu_src2[2]是否要对该指令的 PC 进行+8 处理。

sel_alu_src2[3]是否要对立即数进行无符号扩展

alu_op 用来判断传入加法器的数据要进行哪种类型的操作，如果要进行某一操作，则将该操作所对应的为置为 1，将其它置为 0.

data_ram_en 用来判断该指令是否需要对内存进行处理，如果需要置为 1

lb_lh_lw 表示是否要将从虚地址读出的数据写入 rt 寄存器，如果是 sw 指令那么置为 1111，代表取四个字节的数据存入到 rt 寄存器中；若是 lh 指令置为 0011，代表取连续的两个字节并进行无符号扩展并存入 rt 寄存器中；若是 lhu 指令置为 0111，代表取连续的两个字节并进行有符号扩展并存入 rt 寄存器中；若是 lb 指令置为 0001，代表取一个字节并进行无符号扩展并存入 rt 寄存器中；若是 lbu 指令置为 0101，代表取一个字节并进行有符号扩展并存入 rt 寄存器中。

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。

sel_rf_res 判断写入寄存器的结果是在 EX 段还是 MEM 段得出的。

hilo_read 判断该指令是否需要从 hilo 寄存器中读取数据（即从 hilo 寄存器中读取 hi 寄存器和 lo 寄存器的数据并将数据存到 rd 寄存器中）

hilo_we 是否要将乘法或除法得到的数据存入 hilo 寄存器中。

id_to_ex_bus 将从 id 段得到的需要在后面几段用到的中间结果进行打包，并将打包后的总线传给 EX 段。

br_e 判断是要跳转指令还是顺序执行指令。

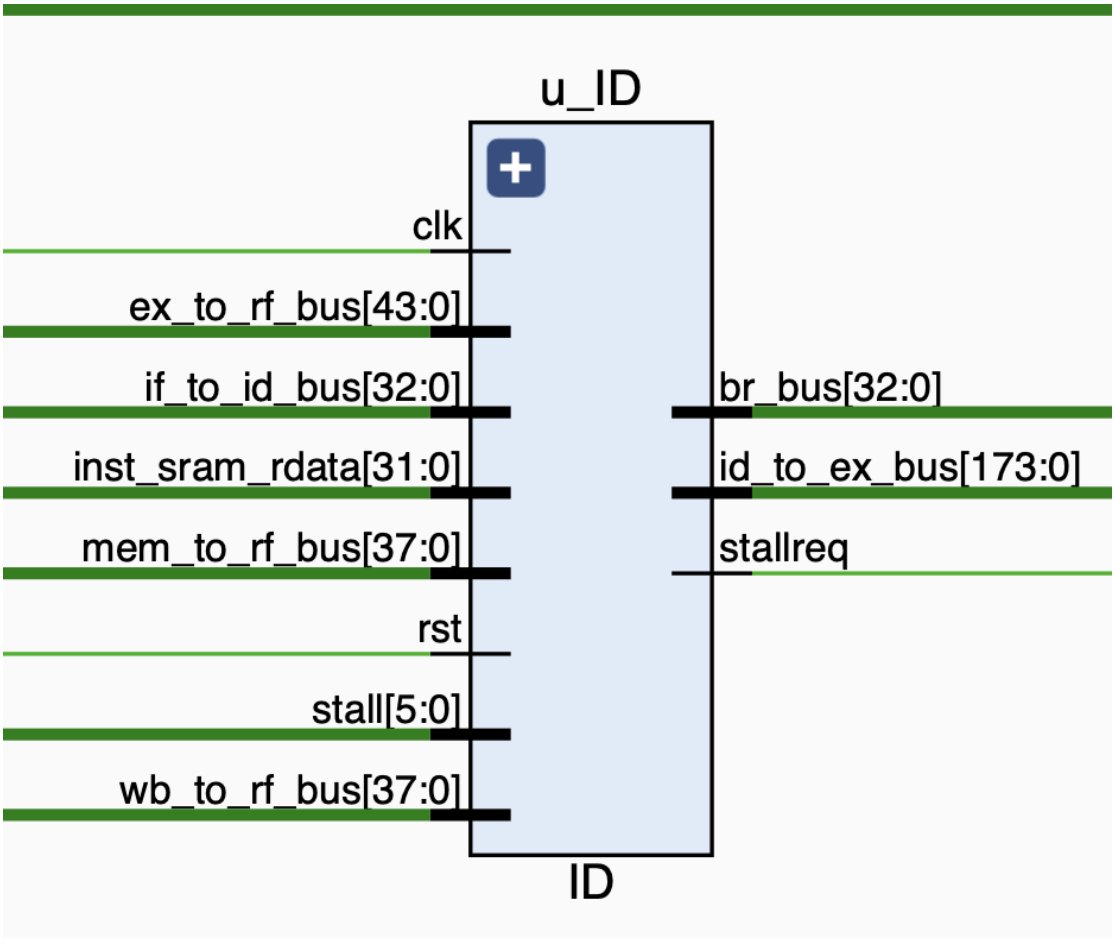
br_addr 下一个要执行的指令（是跳转之后的，不是顺序执行的）

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。

load_stop 用来判断该指令是否需要停止的因素，若为 1 则停止。

stallreq 用来判断该指令是否需要停止，如果 load_stop 为 1 且存在 rs 寄存器或 rt 寄存器的读后写等数据相关，则其赋值为 1，表示需要停止一个周期。

2.4 结构示意图



3.EX

3.1 整体功能说明

EX 段负责执行，EX 段对经 ID 段解析的指令进行运算，得到相关的结果

3.2 端口，信号介绍

传入时钟周期 clk；复位信号，负责初始化各项数据 rst；停止信号，负责暂停流水线 stall；从 ID 段传到 EX 段的总线 id_to_ex_bus；EX 段到 MEM 段的总线 ex_to_mem_bus；是否对内存有操作 data_sram_en；写入内存的使能 data_sram_wen；要写入的内存的地址 data_sram_addr；写入内存的数据 data_sram_wdata；EX 段到 ID 段的总线 ex_to_rf_bus；判断是否需要暂停（乘法，除法）stallreq_for_ex。

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，其中加法器的接口的作

用分别是判断是哪种运算的 alu_op，第一个操作数 alu_src1，第二个操作数 alu_src2，计算结果 alu_result。

乘法器 mul 负责乘法计算，基本原理是通过传入乘法的两个源操作数进行乘法运算，我们选用的乘法器具有多个周期，通过乘法的结束信号来判断乘法是否已经运算完成，如果没有完成则要对整个流水线进行暂停，等运算结束后再进行其他指令的执行，其中乘法器的接口的作用分别是：传入时钟周期 clk；复位信号，负责初始化各项数据 rst；是否是有符号的乘法 mul_signed；乘法源操作数 1 ina；乘法源操作数 2 inb；乘法开始信号 start_i；乘法结束信号 ready_o；乘法结果（64bit）result。

除法器 div 负责除法计算，基本原理是通过传入的被除数和除数进行除法运算，我们选用的除法器具有多个周期，通过除法的结束信号来判断除法是否已经运算完成，如果没有完成则要对整个流水线进行暂停，等运算结束后再进行其他指令的执行，其中除法器的接口的作用分别是：传入时钟周期 clk；复位信号，负责初始化各项数据 rst；判断是否为有符号除法运算 signed_div_i；被除数 opdata1_i；除数 opdata2_i；判断是否开始除法运算 start_i；判断是否要取消除法运算 annul_i；除法结果（64bit）result_o；除法运算是否结束 ready_o。

HI-LO 寄存器 hilo，该寄存器与 regfile 寄存不同，hilo 寄存器用来存放乘法和除法的数据，乘法的乘积的低半部分和高半部分分别写入 LO 寄存器和 HI 寄存器。除法的商写入 LO 寄存器中，余数写入 HI 寄存器中。其中 HI-LO 寄存器的接口的作用分别是：传入时钟周期 clk；复位信号，负责初始化各项数据 rst；是否写入 hilo 寄存器 we；写入 hilo 寄存器的地址 hilo；从 hi 寄存器读出的数据 hi；从 lo 寄存器读出的数据 lo。

always 中用来赋值，当复位信号为 1 时（即需要复位）则给 ID 段到 EX 段的总线和赋初值 0，当复位信号为 0 时（即指令开始运行）将总线用从 ID 段传来的 id_to_ex_bus 赋值.

data_sram_addr 用来判断是否用到了虚拟内存，如果用到了，则将取数据的地址赋值为将 base 寄存器的值加上符号扩展后的立即数 offset，如果没有用到则将该地址赋值为 0.

new_data_sram_wen 即从 ID 段传来的 data_ram_wen 用来判断传进来的指令是 sw，sb 还是 sw。

data_sram_wen 用来判断写入内存的存放位置，如果为 sw 那么代表从 rt 寄存器中处取到的全部四个字节的数据存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 00，那么代表将 rt 寄存器的低半字存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为

10，那么代表将 rt 寄存器的高半字存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 00，么代表将 rt 寄存器的最低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 01，么代表将 rt 寄存器的倒数第二低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 10，么代表将 rt 寄存器的第二高位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 11，么代表将 rt 寄存器的最高位的字节存到之前计算出的虚地址所指的内存处；

data_sram_wdata 表示 sw，sb，sh 三条指令要根据虚拟地址写入内存中的数据。如果为 sw 那么代表从 rt 寄存器中处取到的全部四个字节的数据存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 00，那么代表将 rt 寄存器的低半字存到之前计算出的虚地址所指的内存处；如果为 sh 指令且虚地址后两位为 10，那么代表将 rt 寄存器的高半字存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 00，么代表将 rt 寄存器的最低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 01，么代表将 rt 寄存器的倒数第二低位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 10，么代表将 rt 寄存器的第二高位的字节存到之前计算出的虚地址所指的内存处；如果为 sb 指令且虚地址后两位为 11，么代表将 rt 寄存器的最高位的字节存到之前计算出的虚地址所指的内存处；

new_lb_lw_lh 表示将从内存中取出来的数据存入到 rt 寄存器中，如果为 lw 那么将从内存中取出来的数据存入到 rt 寄存器中；若为 lh 且虚拟地址后两位为 00，那么将从内存中取出的数的高半字进行有符号扩展后存入 rt 寄存器中；若为 lh 且虚拟地址后两位为 10，那么将从内存中取出的数的低半字进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 00，那么将从内存中取出的数的最高位字节进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 01，那么将从内存中取出的数的第二高位字节进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 10，那么将从内存中取出的数的第三高位字节进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 11，那么将从内存中取出的数的最低位字节进行有符号扩展后存入 rt 寄存器中；若为 lhu 且虚拟地址后两位为 00，那么将从内存中取出的数的高半字进行 0 扩展后存入 rt 寄存器中；若为 lhu 且虚拟地址后两位为 10，那么将从内存中取出的数的低半字进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 00，那么将从内存中取出的数的最高位字节进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 01，那么将从内存中

取出的数的第二高位字节进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 10，那么将从内存中取出的数的第三高位字节进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 11，那么将从内存中取出的数的最低位字节进行 0 扩展后存入 rt 寄存器中。

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

alu_src1 用来判断加法器的第一个操作数选择哪个（从 regfile 中读出的 rs 寄存器中的数，之前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一个操作数选择哪个（立即数有符号扩展之后的数，三十二位二进制的 8，立即数无符号扩展之后的数，以及从 regfile 中读出的 rt 寄存器中的数）

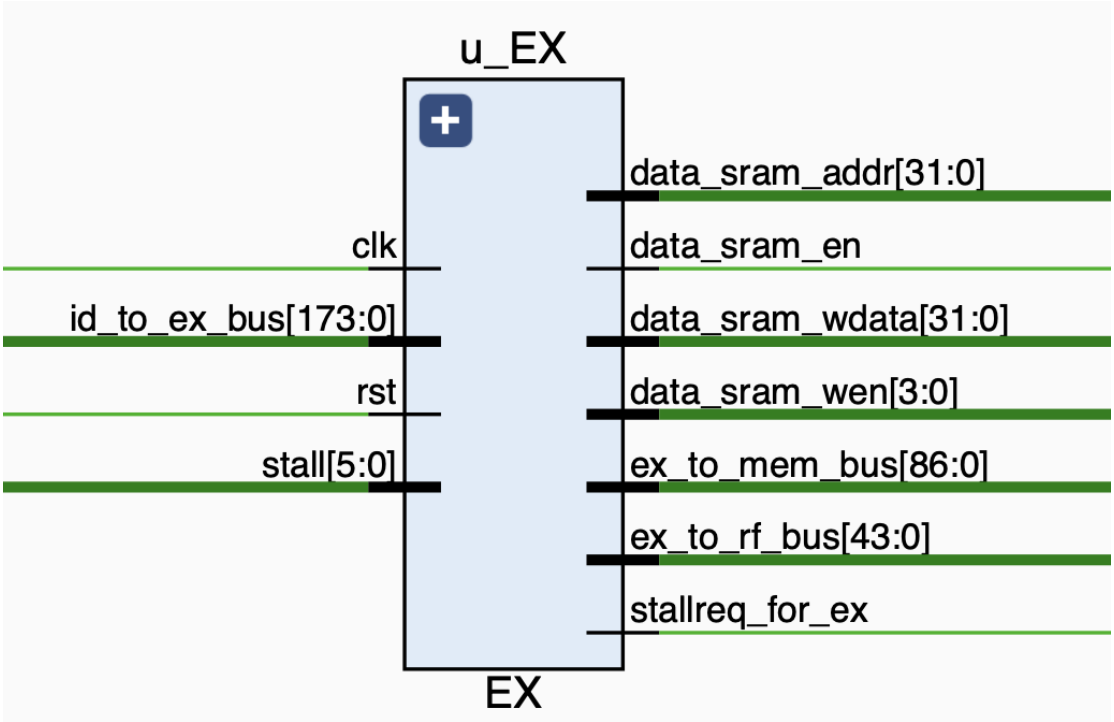
stallreq_for_ex 用于判断当前阶段是否需要暂停（根据乘法器和除法器模块中的结束信号判断，即当结束信号为 1 时表示乘法和除法已经执行完成了，可以不用再暂停了；当结束为 0 时表示乘法和除法还没有完成，还需要继续暂停）

hilo_data 表示写入 hilo 寄存器的数据，如果允许写入且当前指令为除法类型，那么用除法结果 div_result 给 hilo_data 赋值；如果允许写入且当前指令为乘法类型，那么用乘法结果 mul_result 给 hilo_data 赋值；如果为 mthi 指令，那么将 rs 寄存器中的值经过无符号扩展后赋值给给 hilo_data；如果为 mtlo 指令，那么将 rs 寄存器中的值放在高位，低位赋 0 后，赋值给 hilo_data。

ex_result 为再 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

4.1 整体功能说明

在 MEM 段进行对内存数据的读入，通过判断，决定使用的数据是从内存中传入的还是从 EX 段传入的，并将其传到 WB 段

4.2 端口，信号介绍

传入时钟周期 clk;复位信号，负责初始化各项数据 rst;停止信号，负责暂停流水线 stall;EX 段到 MEM 段总线 ex_to_mem_bus；从内存中读入的数据 data_sram_rdata；MEM 段到 WB 段的总线 mem_to_wb_bus；MEM 段到 ID 段的总线 mem_to_rf_bus

4.3 功能模块说明

ex_to_mem_bus_r 负责获得从 EX 段传来的总线。always 模块中如果复位信号为 1（即处于复位状态下）或着暂停信号为 1 时，将 ex_to_mem_bus_r 赋值为 0，否则将 ex_to_mem_bus_r 用从 EX 段传来的总线赋值。

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

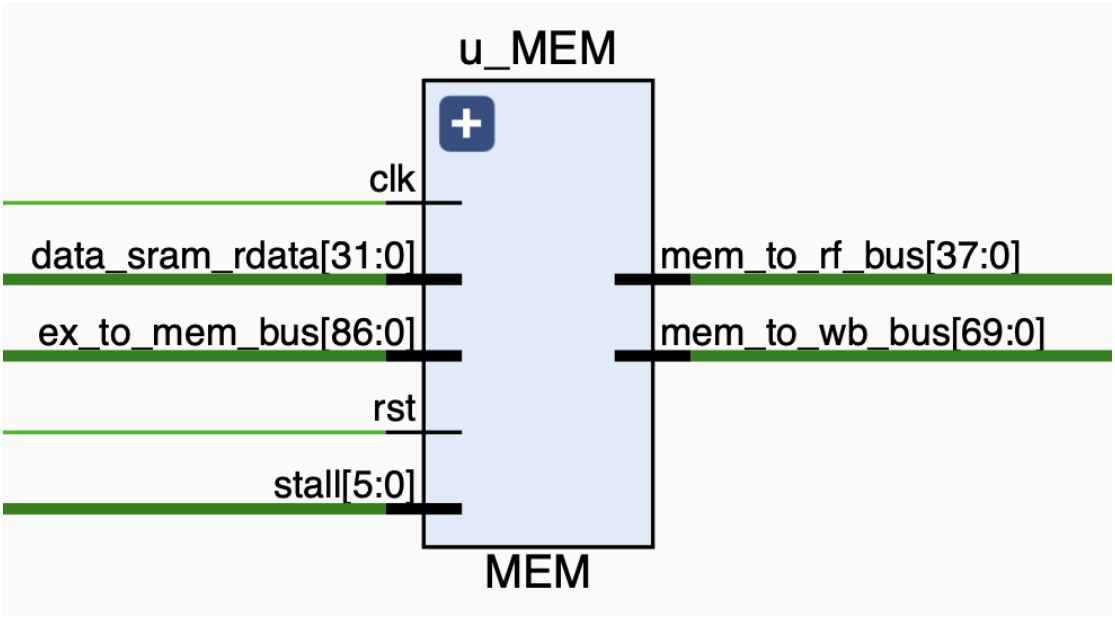
mem_result 表示将从内存中取出来的数据存入到 rt 寄存器中，如果为 lw 那么将从内存中取出来的数据存入到 rt 寄存器中；若为 lh 且虚拟地址后两位为 00，那么将从内存中取出的数的高半字进行有符号扩展后存入 rt 寄存器中；若为 lh 且虚拟地址后两位为 10，那么将从内存中取出的数的低半字进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 00，那么将从内存中取出的数的最高位字节进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 01，那么将从内存中取出的数的第二高位字节进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 10，那么将从内存中取出的数的第三高位字节进行有符号扩展后存入 rt 寄存器中；若为 lb 且虚拟地址后两位为 11，那么将从内存中取出的数的最低位字节进行有符号扩展后存入 rt 寄存器中；若为 lhu 且虚拟地址后两位为 00，那么将从内存中取出的数的高半字进行 0 扩展后存入 rt 寄存器中；若为 lhu 且虚拟地址后两位为 10，那么将从内存中取出的数的低半字进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 00，那么将从内存中取出的数的最高位字节进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 01，那么将从内存中取出的数的第二高位字节进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 10，那么将从内存中取出的数的第三高位字节进行 0 扩展后存入 rt 寄存器中；若为 lbu 且虚拟地址后两位为 11，那么将从内存中取出的数的最低位字节进行 0 扩展后存入 rt 寄存器中。

rf_wdata 用来存放数据，根据 sel_rf_res 进行判断是使用 MEM 段算出的数据还是从 EX 传来的数据。

mem_to_wb_bus 表示从 MEM 段传到 WB 段的总线。

mem_to_rf_bus 表示从 MEM 段传回 ID 段的总线。

4.4 结构示意图



5.WB

5.1 整体功能说明

WB 段负责将数据写回，

5.2 端口，信号介绍

传入时钟周期 clk；复位信号，负责初始化各项数据 rst；停止信号，负责暂停流水线 stall；MEM 段到 WB 段的总线 mem_to_wb_bus；WB 段到 ID 段的总线 wb_to_rf_bus；当前阶段的指令 debug_wb_pc；当前阶段的写入使能 debug_wb_rf_wen；当前的写入地址 debug_wb_rf_wnum；当前的写入数据 debug_wb_rf_wdata。

5.3 功能模块说明

mem_to_wb_bus_r 负责获得从 MEM 段传来的总线。

always 模块中如果复位信号为 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 用从 MEM 段传来的总线赋值。

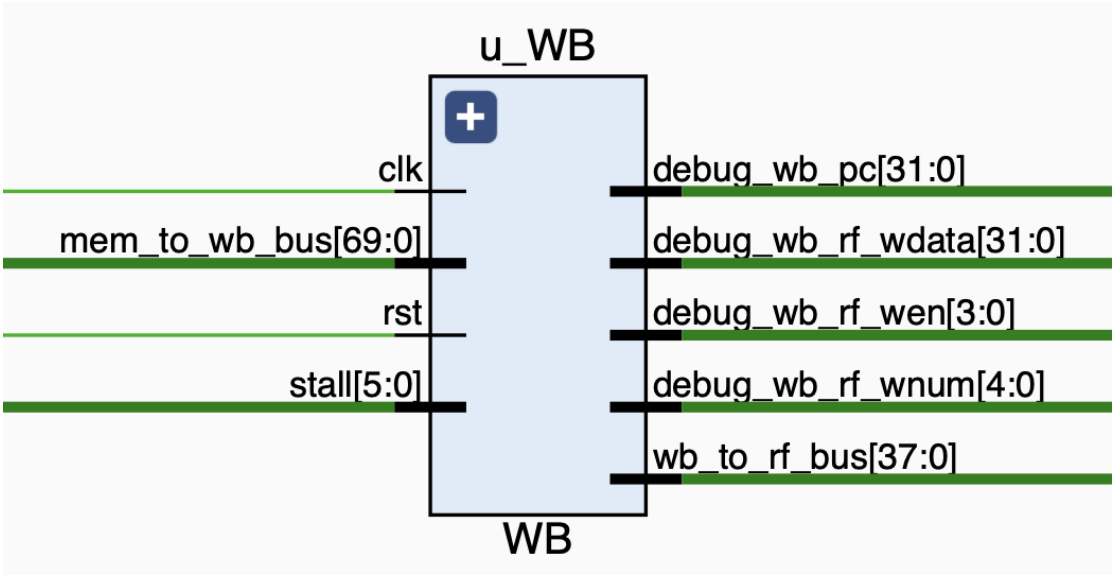
wb_to_rf_bus 表示从 WB 段传回 ID 段的总线。

debug_wb_pc 是用来检测当前的 pc 值是否正确。

debug_wb_rf_wnum 是用来检测当前的写入的地址值是否正确。

debug_wb_rf_wdata 是用来检测当前的写入的数据是否正确。

5.4 结构示意图



6.自己实现的 MUL 模块

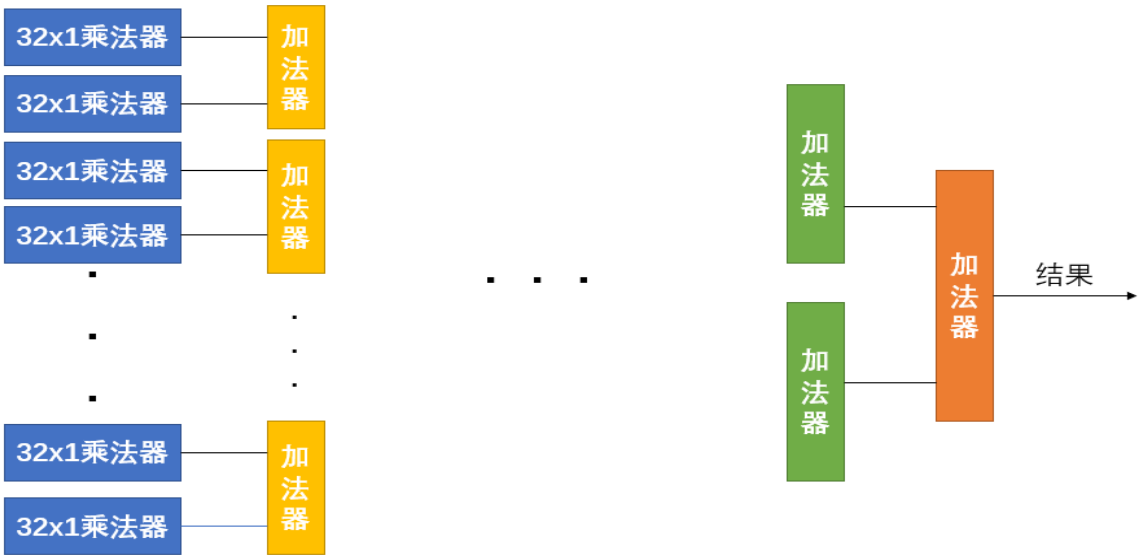
在实现 MUL 模块中，我们实现了多种 MUL 乘法，比如加法树和移位乘法（因为是高位宽数据所以没有采用查找表算法）。

6.1 加法树乘法器

1.1.1 基本思想：

加法树的思想在于用加法器每两个元素加在一起，其最大的特点在于用资源的增加而提高运行速度，逻辑关系简单明了。它结合了查找表和移位相加的有点，速度比较快。

1.1.2 图示：



6.2 移位相加乘法器

1.1.1 基本思想：

移位相加乘法器的计算流程为从被乘数的左边（最低位）开始，如果第 i 位为 1，则乘数左移 i （i = 0,1,2,……，size -1）位之后与之前的值相加，若最低位为 0 ，则保持不变，直至被乘数的最高位。耗用资源相对少，运算速度比较慢。

6.3 代码实现

最后经过选择，我们决定多个周期实现加法树乘法器的实现。

6.4 接口说明

```
module mul (
    input wire clk,
    input wire resetn,
    input wire mul_signed, //signed is 1, unsigned is 0
    input wire [31:0] ina, //乘数
    input wire [31:0] inb, //被乘数
    input wire start_i, //判断是否开始做乘法运算
    output reg signed [63:0] result,//乘法结果
    output reg ready_o //在多周期中判断周期运算是否结束

);
```

在每一个周期内，我们完成对于一层的计算，然后将结果保存在寄存器中，直到下一周期再取出寄存器的值进行下一层的运算，直到算出最后结果。将 ready_o 置 1，结果输出。否则外界会一直请求暂停。

三．组员实验感受，改进意见

程思睿实验感受：

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。

马韬洵实验感受：

通过本次实验，我对 CPU 的基本原理与其相关的流水线有了进一步的了解与认识，摆脱了之前只在书本上了解到的片面认知。在进行实验的过程中，我更加深入的了解了 MIPS 寄存器文件基本概念及其部分指令集所代表的含义，进一步熟悉了多路选择器、译码器、解复用器等组件的使用，在培养了我对于汇编代码能够进行简单阅读与分析从而得出每步相应结果的能力，也养成了我在 Vivado 运行出错时，能够尝试去寻找波形图中的错误，从而使问题得到解决的能力。本次实验使我巩固了上课中所讲授的知识，让其不止停留于理论，而是下放到实际实践操作中去，使我们对其有了物理上的了解，也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。

孙辉实验感受：

通过这次制作 CPU 的实验，我对 CPU 的五个段，流水以及数据相关有了更加直观的了解，对 vivado 的使用和在程序出问题是的 debug 让我学会了 verilog 的部分语法和如何看波形图。在实验的最开始，面对 cpu 的代码时，从完全看不懂各个文件的用途，到能渐渐能明白各个阶段大体所要作的工作，再到最后能够编写相关功能的代码，此次实验不仅让我学习了 CPU 的相关知识，还让我学会了如何在面对一项全新的工作时慢慢将复杂的代码看懂编写，在进行实验时我还查阅了关于 CPU 的相关资料，例如《自己动手做 cpu_雷思磊》等，这不仅复习了我之前学过的计算机系统的知识还让我学到了一些课堂上没讲到的实践中的知识，进一步增加了我对 cpu 的了解。最后在此次实验中我不仅巩固了上课所学内容，还增加了实践经验，只要认真去做，每个阶段都会有不同的收获.

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. 《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》