

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 2202

组长：兰天

组员：徐浩淞

报告日期：2024.1.4

目录

1. 实验设计

1.1 人员分工

1.2 总体设计

1.3 运行环境及工具

2. 各模块详细设计

2.1 IF 模块

2.2 ID 模块

2.3 EX 模块

2.4 MEM 模块

2.5 WB 模块

2.6 CTRL 模块

2.7 HILO 寄存器模块

3. 实验感受及建议

3.1 兰天部分

3.2 徐浩淞部分

4. 参考资料

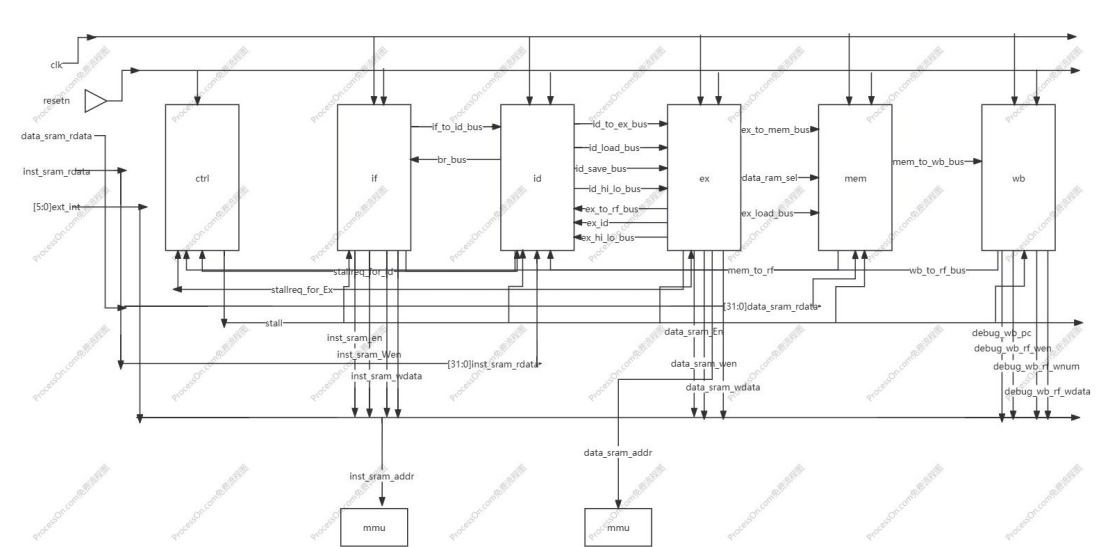
1. 实验设计

1.1 人员分工

兰天：

徐浩淞：负责实现了取指、译码、执行、访存和写回阶段的功能，并设计了支持乘法和除法操作的 hi 和 lo 寄存器。控制单元的设计确保了流水线的动态暂停与恢复，特别处理了数据冒险和分支指令的情况。通过设计调试信号，提供了程序计数器和寄存器写入状态的追踪功能。参与编写实验报告。（任务占比 50%）

1.2 总体设计



1.3 运行环境及工具

软件工具：

Vivado（用于硬件描述语言的编译和仿真），

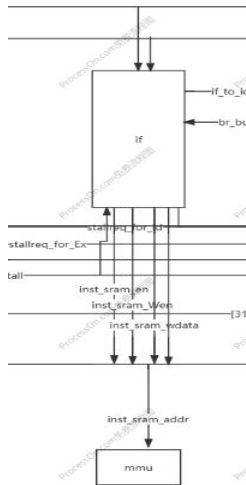
VSCode（用于代码编辑），

GitHub（版本管理）。

2. 各模块详细设计

2.1 IF 模块

结构：



输入输出：

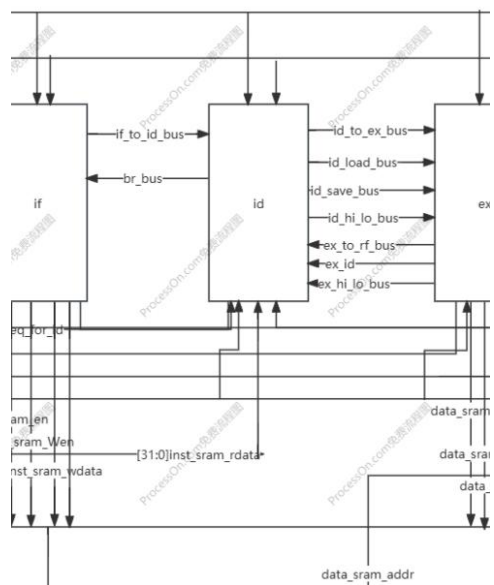
序号	接口	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	暂停信号
4	br_bus	33	输入	分支跳转信号（延迟槽）
5	if_to_id_bus	33	输入	if 段到 id 段的数据传输
6	inst_sram_en	1	输出	指令寄存器读写使能信号
7	inst_sram_wen	4	输出	指令寄存器写使能信号
8	inst_sram_addr	32	输出	指令寄存器地址
9	inst_sram_wdata	32	输出	指令寄存器数据

功能说明：

指令获取阶段，实现了程序计数器（PC）的更新，分支跳转，指令的读取等功能。IF 段会读取输入的 clk、rst 等信号如上图。在这个模块中会先解包一些数据，如 br_bus 来控制分支。时钟信号来触发过程语句，根据 rst 和 stall[0]来处理 pc 值和 ce_reg(控制指令存储器的使能信号)。可以根据 pc 计算计算 next_pc 值。最后将利用持续赋值语句将变量与输出数据绑定。

2.2 ID 模块

结构:



输入输出:

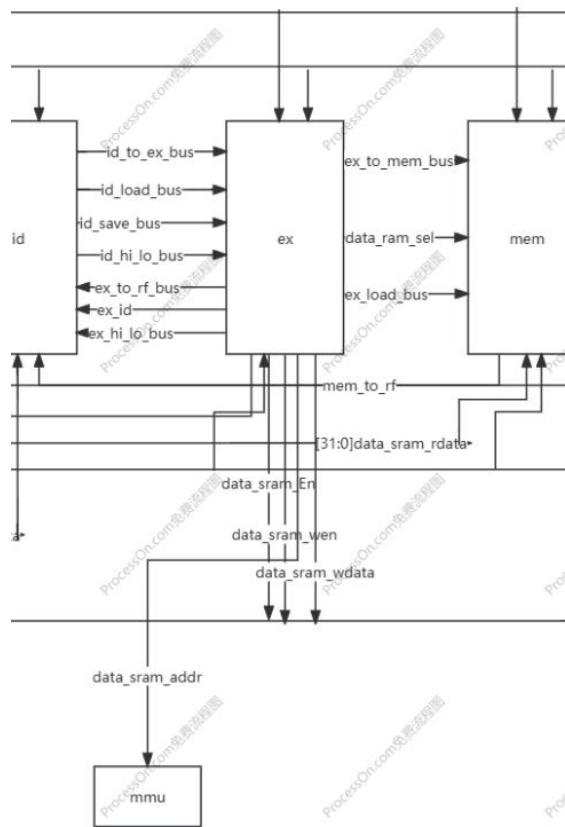
序号	接口	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	暂停信号
4	if_to_id_bus	33	输入	if 到 id 段的总线
5	inst_sram_rdata	32	输入	从指令存储器读取的指令
6	ex_id	1	输入	是否处于 Ex
7	wb_to_rf_bus	38	输入	从 WB 阶段到 RF 的数据传递总线
8	ex_to_rf_bus	38	输入	从 EX 阶段到 RF 的数据传递总线
9	mem_to_rf_bus	38	输入	从 MEM 阶段到 RF 的数据传递总线
10	ex_hi_lo_bus	66	输入	EX 阶段放入 hilo 的数据总线
11	stallreq	1	输出	暂停请求需求
12	id_hi_lo_bus	72	输出	ID 阶段放入 hilo 的数据总

				线
13	id_load_bus	5	输出	加载数据相关信号
14	id_save_bus	3	输出	存储数据相关信号
15	stallreq_for_bru	1	输出	分支相关的暂停
16	id_to_ex_bus	159	输出	ID 段到 EX 段的数据总线
17	br_bus	33	输出	分支跳转信号

功能说明：该模块是一个 指令解码模块，从取指阶段获取数据，并进行解码，并将结果传入 EX（执行阶段）。从输入总线 if_to_id_bus 获取指令地址和指令内容。 将指令内容 (inst) 拆分为操作码、操作数寄存器 (rs 和 rt)，以及立即数等部分。根据指令类型生成控制信号（例如 ALU 操作类型、数据存储使能信号等）。

2.3 EX 模块

结构：



输入输出：

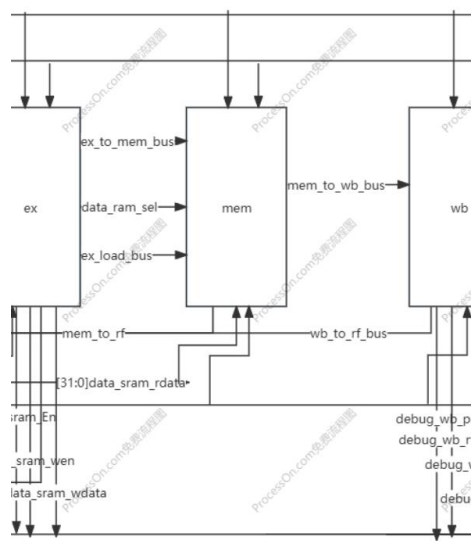
序号	接口	宽度	输入/输出	作用
----	----	----	-------	----

1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	1	输入	暂停信号
4	id_to_ex_bus	169	输入	ID 段到 EX 段数据总线
5	id_load_bus	5	输入	加载数据相关信号
6	load_save_bus	3	输入	存储数据相关信号
7	id_hi_lo_bus	72	输入	ID 阶段放入 hilo 的数据总线
8	ex_ro_mem_bus	80	输出	EX 段到 MEM 段的数据总线
9	ex_to_rf_bus	38	输出	EX 段到写回阶段的数据总线
10	ex_hi_lo_bus	66	输出	EX 段放入 hilo 的数据总线
11	stallreq_for_ex	1	输出	对 EX 段的暂停请求
12	data_sram_en	1	输出	内存数据的使能信号
13	data_sram_wen	4	输出	内存数据的读使能信号
14	data_sram_addr	32	输出	内存数据的地址
15	data_sram_wdata	32	输出	写入数据
16	data_ram_sel	4	输出	数据选择信号
17	ex_load_bus	5	输出	EX 段奥 MEM 的加载数据总线

功能说明：实现了一个 执行阶段（EX） 的模块，用于处理指令流水线中从解码阶段（ID）传递来的信号，并生成数据供后续流水线阶段使用。从 id_to_ex_bus 中提取 ALU 操作数和操作码，进行算术或逻辑运算。处理分支跳转、立即数操作等特殊指令类型。根据操作码和选择信号（如 alu_op）调用算术逻辑单元（ALU）进行计算。计算结果、控制信号等通过 ex_to_mem_bus 传递给内存阶段（MEM）。生成寄存器写回信号，通过 ex_to_rf_bus 与写回阶段交互。

2.4 MEM 模块

结构:



输入输出:

序号	接口	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	暂停信号
4	ex_to_mem_bus	80	输入	ex 传给 mem 的数据
5	data_sram_rdata	32	输入	内存中要写入寄存器的数据
6	data_ram_sel	4	输入	内存数据的选择信号
7	ex_load_bus	5	输入	ex 段读取的数据
8	stallreq_for_load	1	输出	ex 段的 stall 请求
9	mem_to_wb_bus	70	输出	mem 传给 wb 的数据
10	mem_to_rf_bus	38	输出	mem 传给寄存器的数据

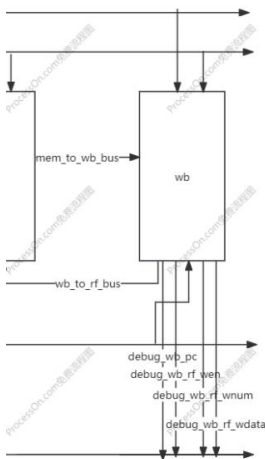
功能说明:

MEM 模块处理 CPU 中的存储操作，主要是 load 和 store 指令。根据指令类型 (lb、lbu、lh、lhu、lw、sb、sh) 访问内存，并选择不同的字节。对于 load 类指令，MEM 模块根据地址的最低两位选择字节并读取；对于 store 类指令，选择字节写使能并将数据写入存储器。模块处理 EX 阶段传来的结果，并将其传递到 WB 阶

段或寄存器堆。通过控制信号（如 stall 信号）管理流水线暂停。还提供访存结果并将其传递给后续模块（wb）。

2.5 WB 模块

结构：



输入输出：

序号	接口	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	rst	1	输入	复位信号
3	stall	6	输入	暂停信号
4	mem_to_wb_bus	70	输入	mem 传给 wb 的数据
5	wb_to_rf_bus	38	输出	wb 传给寄存器的数据
6	debug_wb_pc	32	输出	用来 debug 的 pc 值
7	debug_wb_rf_wen	4	输出	用来 debug 的写使能信号
8	debug_wb_rf_wnum	5	输出	用来 debug 的写寄存器地址
9	debug_wb_rf_wdata	32	输出	用来 debug 的写寄存器数据

功能说明：

WB 模块负责从 MEM/WB 流水线寄存器读取数据，并将其写回寄存器堆。它使用一个寄存器 `mem_to_wb_bus_r` 存储从 MEM/WB 寄存器读取的数据。通过 `wb_to_rf_bus` 输出写回寄存器堆的数据。该模块根据时钟和复位信号控制数据的

传递，并通过 stall 信号管理流水线的暂停。在调试模式下，WB 模块还会提供调试信号，包括 PC 值、寄存器写使能信号、写入寄存器的地址和数据。

2.6 CTRL 模块

输入输出：

序号	接口	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	stallreq_for_ex	1	输入	执行阶段是否请求暂停
3	stallreq_for_bru	5	输入	load 命令是否请求暂停
4	stall	6	输出	暂停信号

功能说明：

CTRL 模块根据不同的请求信号控制流水线暂停。有三个输入信号：stallreq_for_ex、stallreq_for_bru 和 stallreq_for_load，用于请求在不同阶段暂停。根据这些信号的状态，stall 信号会设置不同的值。stall[0]表示 PC 是否保持不变，其他位表示各个流水线阶段是否暂停。模块根据复位信号初始化 stall 为 0。当接收到暂停请求时，stall 信号会根据请求的阶段设置相应的值。

2.7 HILO 寄存器模块

输入输出：

序号	接口	宽度	输入/输出	作用
1	clk	1	输入	时钟信号
2	stall	6	输入	暂停信号
3	hi_we	1	输入	hi 写使能信号
4	lo_we	1	输入	lo 写使能信号
5	hi_wdata	32	输出	hi 寄存器写的的数据
6	lo_wdata	32	输出	lo 寄存器写的的数据
7	hi_rdata	32	输出	hi 寄存器读的数据
8	lo_rdata	32	输出	lo 寄存器读的数据

功能说明：

当 hi_we 和 lo_we 均为 1 时，reg_hi 和 reg_lo 会同时接收 hi_wdata 和 lo_wdata 的数据。

当 hi_we 为 0, lo_we 为 1 时, reg_lo 会接收 lo_wdata 数据。

当 hi_we 为 1, lo_we 为 0 时, reg_hi 会接收 hi_wdata 数据。

hi_rdata 和 lo_rdata 分别输出 reg_hi 和 reg_lo 中的数据。

该模块用于处理协处理器中的 hi 和 lo 寄存器。hi 和 lo 寄存器用于存储乘法和除法的结果, 其中 hi 存储高位, lo 存储低位。hi 和 lo 寄存器写入操作通过 hi_we 和 lo_we 信号控制, 读取操作通过 hi_rdata 和 lo_rdata 输出。

3. 实验感受及建议

3.1 兰天部分

经过实验学会了 verilog 的部分语法, 能补全本次实验的代码。当然也第一次使用 github 进行协作。在分支合并中导致文件错误, 导致后面的调试又花了部分时间。而在代码编写过程中, 我阅读《自己动手写 CPU》等资料来进行代码编写, 其中对于变量的数据连接弄的比较混乱, 后来在建立图像后得以解决这个问题。这次实验较难, 需要很强的团队合作能力, 我和同学的协作能力也在这次过程中得以提高。

3.2 徐浩淞部分

通过本次实验, 我学习了如何使用 github 与同伴协同进行一个项目的编写和管理同步, 虽然在合并分支的过程会导致某些文件的丢失, 但是总体还是挺好用的。同时对五级流水 cpu 结构有了更清楚的认知, 对各个模块的功能更加熟悉, 学习了新的编程语言的使用, 虽然都没有非常精通, 但是还是比较有收获。主要建议就是题目对于我们来说还是感觉过于棘手, ppt 给出的流程不够明确和详细, 对于很多内容无从下手, 或许可以从具体的实现路径方向来优化课程安排。

4. 参考资料

- 1、雷思磊 著《自己动手写 CPU》 电子工业出版社
- 2、龙芯杯官方的参考文档
- 3、潘文明 著 《手把手教你学 FPGA 的设计》
- 4、助教给出的 verilog 基础文档
- 5、龙芯杯官方资料包