个人赛设计报告

学校：安徽大学
姓名：郑忠强

目录

表目录

图目录

一、设计简介

提交设计： 50MB 频率的单发射顺序 5 段流水的 MIPS cpu。它能实现比赛所有的指令功能并通过全部测试。

特色：本次设计极限提高 cpu 的 ipc,在整个流水中出现及其少的气泡插入：产生气泡的地方只有：

1、 load 指令是访存 baseram,

2、 store 指令后一条是 load 指令，

3、 lw 指令后一条运算指令(store 指令写入存储器的数据发生 raw 相关也不暂停，通过在 mem 阶段获取 store 指令的写入数据不在是在 id 阶段获取 store 指令存储到数据区的数据)与其发生 raw 数据相关

二、设计方案

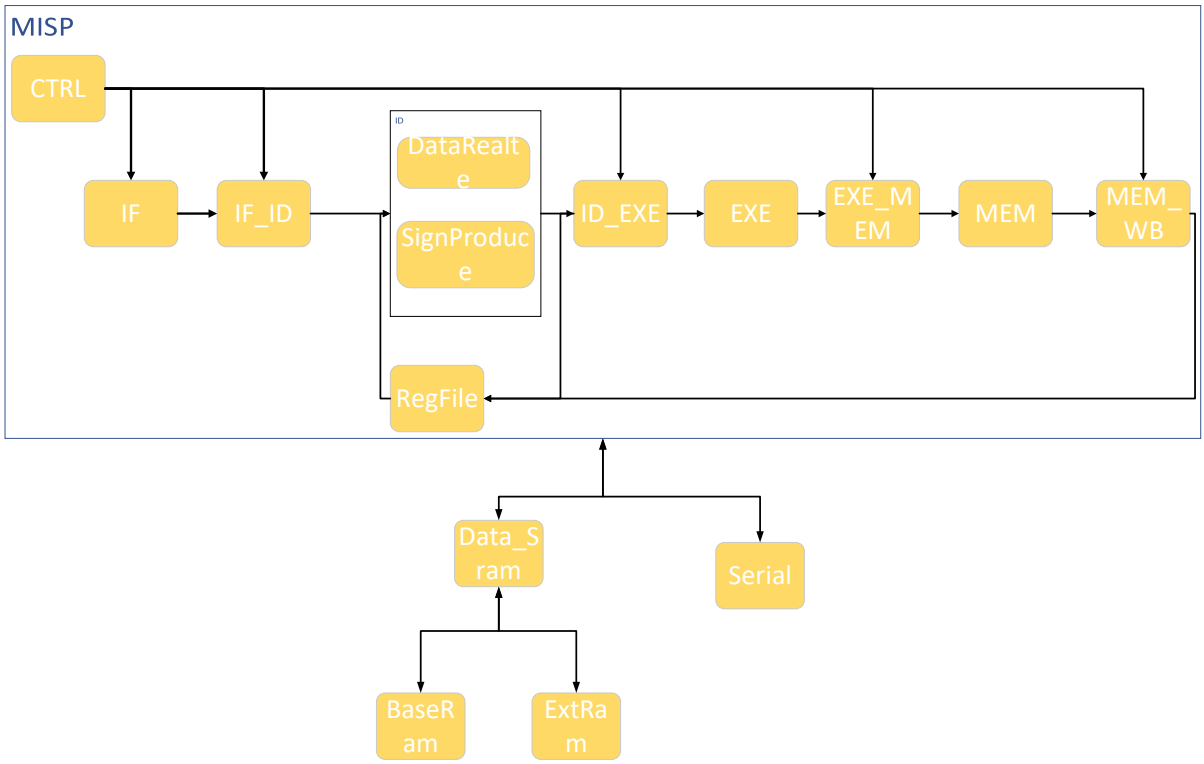
（一）总体设计思路

总体设计：使用标准的 5 段流水的设计架构，有三个模块构成：1、MIPS 模块实现 cpu,2、Serial 实现串口访问，3、Data_Sram 实现将 cpu 访问转为 SRam 类型接口。

MIPS 由：取指 IF，译码 ID，运算 EXE，访存 MEM，写回 WB，5 个阶段构成，同时包括他们的暂存区：IF_ID，ID_EXE,EXE_MEM,MEM_WB,还有一个全局控制流水暂停的 CTRL,

信号命名规则：zzz(那个阶段产生的（这个部分有可能不含有）)_xxx（那个阶段使用的）_yyy(信号含义)_i/oL（i 表示 input，o 表示 output,含义 L 表示是低电平有效否则默认高电平有效)。Eg：ex_regs_we_i,表示 exe 阶段的寄存器组（regs）写使能（we）输入（I）高电平有效

图表 1 MIPS CPU 结构图



模块信号表 1　MIPS 模块信号

信号	位宽	含义
rom_inst_i	32	输入：pc 在指令区读取到的指令
ram_rdata_i	32	输入：数据存储区读取的数据
serial_rdata_i	32	输入：串口读出数据
rom_raddr_o	32	输入：指令区访问地址
ram_rwaddr_o	32	输出：数据存储区读写地址
ram_we_o	1	输出：数据存储区写使能
ram_rwsel_o	4	输出：数据存储区写字节
ram_wdata_o	32	输出；数据存储器写数据
ram_req_o	1	输出：数据存储区有效
write_wb_regs_data_o	1	输出：修正 store 指令写数据过时的信号
wb_regs_data_o	32	输出：修正 store 指令写数据

（二）IF 模块设计

1.1、 **IF 阶段实现功能：**IF 阶段完成:对运行指令的 pc 指令计算

模块信号表 2　IF 模块信号

信号	位宽	含义
stop_i	6	输入：Ctrl 发出暂停流水的信号，6 位宽，stop_i[0]=1 表示暂停 if 阶段
banch_flag_i	1	输入：Id 阶段发出的标志发生跳转的信号
banch_address_i	32	输入：Id 阶段发出的跳转指令
pc_o	32	输出：IF 阶段输出的 pc 值

（三）IF_ID 模块设计

功能：存储 if 阶段的 pc 指令和 sram 读出的指令，

模块信号表 3　IF_ID 模块信号

信号	位宽	含义
RstL	1	复位信号低电平有效
clk	1	时钟信号
Pc_i	32	输入：来自 if 阶段输入的 pc
Inst_i	32	输入：Sram 读出的指令
Stop_i	6	输入：来自 Ctrl 发出的暂停信号，stop[1]=1 表示暂停该阶段
Pc_o	32	输出：pc 值
Inst_o	32	输出：指令值

（四）ID 模块设计

本阶段有两个子模块构成 1、Data_Relevant,2、Signal_Produce

设计想法：来自于：华中科技大学慕课（地址见参考设计说明），

功能：对指令进行解码：

1、在本阶段会产生每条指令的控制信号，

2、exe 阶段运算数和运算类型，

2、mem 阶段的读写使能信号，sw/sb 的写入数据，

3、wb 阶段的写使能，写地址。

访问寄存器组

模块信号表 4　ID 模块信号

信号	位宽	含义
pc_i	32	输入：Id 阶段执行的指令 pc 值
inst_i		输入：Id 阶段执行的指令
regs_rdata1_i,	32	输入：Id 阶段指令读出寄存器组的数据 1
regs_rdata2_i,	32	输入：Id 阶段读出寄存器组的数据 2
ex_regs_we_i,	1	输入：Exe 阶段指令寄存器组写使能信号
ex_regs_waddr_i,	5	输入：Exe 阶段写地址

ex_regs_wdata_i,	32	输入：Exe 阶段寄存器组写数据
ex_memtoreg_i,	1	输入：Exe 阶段写入数据是否来自 mem 阶段读出数据
mem_regs_we_i,	1	输入：Mem 阶段寄存器组写使能
mem_regs_waddr_i,	5	输入：Mem 阶段寄存器组写地址
mem_regs_wdata_i,	32	输入：Mem 阶段寄存器组写数据
regs_raddr1_o	5	输出：Id 阶段寄存器组读地址 1
regs_raddr2_o	5	输出：Id 阶段寄存器组读地址 1
alu_op_o,	4	输出：exe 阶段执行的操作类型
alu_oper1_o,	32	输出：exe 阶段执行的操作数 1
alu_oper2_o,	32	输出：exe 阶段执行的操作数 2
mem_req_o,	1	输出:mem 阶段的芯片使能
mem_we_o	1	输出:mem 阶段的写使能
mem_rwbyte_o	1	输出:mem 阶段访问的是双字还是字节
mem_wdata_o,	32	输出:mem 阶段的写入数据
regs_we_o,	1	输出:WB 阶段写使能
regs_waddr_o	5	输出:WB 阶段写地址
memtoreg_o	1	输出：WB 阶段写数据来自 mem 阶段的读出数据（lw,lb 指令）
banch_flag_o,	1	输出：id 阶段发出跳转的信号
banch_address_o,	32	输出：id 阶段发出的跳转地址
return_address_we_o	1	输出：id 阶段跳转指令的返回地址写入寄存器组的写使能
return_address_o,//	32	输出：id 阶段 跳转指令的放回地址
id_stop_o,	1	输出：id 阶段 请求暂停的信号
id_exe_relate_o	1	输出:id 阶段指令和 exe 阶段指令发生 raw 相关，r 如

		果 exe 阶段指令是 load 指令，exe 阶段将发出暂停流水的信号
id_sw_relate_o	1	输出：id 阶段是 store 指令和 exe 阶段发生 raw 相关

4.1Data_Relevant：

由于 id 阶段回访寄存器组，导致存在和 exe，mem 阶段的 raw 数据相关，这个模块就是用于判别本阶段访问的寄存器组地址是否是 exe 阶段和 mem 阶段执行的指令的写入地址相同。

模块信号表 5　ID 的 Data_Relevant 模块信号

信号	位宽	含义
ex_regs_we_i	1	输入：Exe 阶段指令寄存器写使能
ex_memtoreg_i,	1	输入：Exe 阶段指令是 load 指令
ex_regs_waddr_i	5	输入：Exe 阶段指令寄存器组写地址
mem_regs_we_i	1	输入：Mem 阶段指令寄存器写使能
mem_regs_waddr_i,	5	输入：Mem 阶段寄存器组写地址
regs_rwaddr1_i,	5	输入：Id 阶段指令寄存器读地址 1
regs_rwaddr2_i	5	输入：Id 阶段指令寄存器读地址 2
exe_relate1_o,	1	输出：读取地址 1 和 exe 阶段指令发送 raw 相关
mem_relate1_o,	1	输出：读取地址 1 和 mem 阶段指令发送 raw 相关
exe_relate2_o,	1	输出：读取地址 1 和 exe 阶段指令发送 raw 相关
mem_relate2_o	1	输出：读取地址 1 和 mem 阶段指令发送 raw 相关

4.2 控制信号产生器

模块信号表 6　ID 的 signal_Produce 模块信号

信号	位宽	含义
Op_i	6	输入：Id 阶段指令的[31：26]
Func_i	6	输入：Id 阶段指令的[5:0
Aluop_o	4	输出：输出 id 阶段指令对应的运算类型

Sign_o	28	输出：Id 阶段指令的控制信号

Sign_o 信号解码可以参考附件中的 excel 文件的介绍

（五）ID_EXE 阶段：保存 id 阶段的输出，就不再介绍了

模块信号表 7　ID_EXE 模块信号

信号	位宽	含义
Alu_op_i	4	输入：exe 阶段执行的运算类型
alu_oper1_i	32	输入：exe 阶段执行的操作数 1
alu_oper2_i	32	输入：exe 阶段执行的操作数 2
mem_req_i	1	输入:mem 阶段的芯片使能
mem_we_i	1	输入:mem 阶段的写使能
mem_rwbyte_i	1	输入:mem 阶段访问的是双字还是字节
mem_wdata_i	32	输入:mem 阶段的写入数据
regs_we_i	1	输入:WB 阶段写使能
regs_waddr_i	5	输入:WB 阶段写地址
memtoreg_i	1	输入：WB 阶段写数据来自 mem 阶段的读出数据（lw,lb 指令）
return_address_we_i	1	输入：id 阶段跳转指令的返回地址写入寄存器组的写使能
return_address_i	32	输入：id 阶段 跳转指令的放回地址
id_sw_relate_i	1	输入：id 阶段是 store 指令和 exe 阶段发生 raw 相关
alu_oper1_o	32	输出：exe 阶段执行的操作数 1
alu_oper2_o	32	输出：exe 阶段执行的操作数 2
mem_req_o	1	输出:mem 阶段的芯片使能
mem_we_o	1	输出:mem 阶段的写使能
mem_rwbyte_o	1	输出:mem 阶段访问的是双字还是字节
mem_wdata_o	32	输出:mem 阶段的写入数据
regs_we_o	1	输出:WB 阶段写使能
regs_waddr_o	5	输出:WB 阶段写地址
alu_op_o	4	输出：exe 阶段执行的操作类型
alu_oper1_o	32	输出：exe 阶段执行的操作数 1

alu_oper2_o	32	输出：exe 阶段执行的操作数 2
mem_req_o	1	输出:mem 阶段的芯片使能
mem_we_o	1	输出:mem 阶段的写使能
mem_rwbyte_o	1	输出:mem 阶段访问的是双字还是字节
mem_wdata_o	32	输出:mem 阶段的写入数据
regs_we_o	1	输出:WB 阶段写使能
regs_waddr_o	5	输出:WB 阶段写地址
memtoreg_o	1	输出：WB 阶段写数据来自 mem 阶段的读出数据（lw,lb 指令）
return_address_we_o	1	输出：id 阶段跳转指令的返回地址写入寄存器组的写使能
return_address_o,	32	输出：id 阶段 跳转指令的放回地址
id_sw_relate_o	1	输出：id 阶段是 store 指令和 exe 阶段发生 raw 相关

（六）EXE 模块设计

完成对运算的计算，并传输 mem 阶段的信号和 wb 阶段信号
包含一个 alu 运算模块实现对输入的操作数据根据运算类型进行运算
模块信号表 8　exe 模块信号

信号	位宽	含义
alu_op_i	**1**	**输入：alu 运算类型**
alu_oper1_i	32	输入：alu 运算操作数 1
alu_oper2_i	**32**	**输入：alu 运算操作数 1**
mem_req_i	1	输入:mem 阶段的芯片使能
mem_we_i	**1**	**输入:mem 阶段的写使能**
mem_rwbyte_i	1	输入:mem 阶段访问的是双字还是字节
mem_wdata_i	**32**	**输入:mem 阶段的写入数据**
regs_we_i	1	输入:WB 阶段写使能
regs_waddr_i	**5**	**输入:WB 阶段写地址**
memtoreg_i	1	输入：WB 阶段写数据来自 mem 阶段的读出数据（lw,lb 指令）
return_address_we_i	1	输入：id 阶段跳转指令的返回地址写入寄存器组的写使能
return_address_i　,	32	输入：id 阶段

		输入：跳转指令的放回地址
pre_inst_exe_realte_i	1	输入：id 阶段是 store 指令和 exe 阶段发生 raw 相关
Sw_relate_i	1	当前 exe 阶段执行指令是 store 且上一条指令发生数据相关
alu_result_o	32	输出：Alu 运算结果
mem_req_o	1	输出：Mem 阶段有效
mem_we_o	1	输出：Mem 阶段写使能
mem_rwbyte_o	1	输出：Mem 阶段写字节
mem_rwaddr_o	32	输出：Mem 阶段读写地址
mem_wdata_o	32	输出：Mem 阶段写数据
regs_we_o	1	输出：WB 阶段写使能
regs_waddr_o	5	输出：WB 阶段写地址
memtoreg_o	1	输出：WB 阶段写入数据来自 MEME 阶段读出数据
stop_o	1	输出：申请暂停流水信号

（七）MEM 模块设计

模块信号表 9　Mem 模块信号

信号	位宽	含义
alu_result_i,	32	输入：指令运算结果
serial_rdata_i	32	输入；串口读出数据
mem_req_i	1	输入：访问存储器
mem_we_i	1	输入；存储器写使能
mem_rwbyte_i	1	输入：访存指令的访问字节
mem_rwaddr_i	32	输入：访存指令的访存地址
mem_wdata_i	32	输入：store 指令写数据
mem_rdata_i	32	输入：存储器读出数据
regs_waddr_i	5	输入:寄存器组写地址
regs_we_i	1	输入：寄存器组写使能
memtoreg_i,	1	输入：寄存器组写入数据来自 mem 阶段的读出数据
mem_req_o	1	数据存器有效

mem_we_o	1	输出：数据存储器写使能
mem_rwaddr_o	32	输出：数据存储器读写地址
mem_rwsel_o	4	输出：数据存储器写入字节
mem_wdata_o	32	输出：数据存储器写入数据
regs_we_o	1	输出：寄存器组写使能
regs_waddr_o	32	输出：寄存器组写入地址
regs_wdata_o	32	输出：寄存器组写入数据
stop_o	1	输出：暂停请求

（八）RegFile 模块设计

实现的是 2 个读端口，1 个写端口的 32 个寄存器的寄存器组，对 0 地址始终输出 32‘h0, 同时在此次会解决 raw 相关：即写入地址和读地址相同时，读出数据将会是此时的写入数据

模块信号表 10　regfile 模块信号

信号	位宽	含义

rf_in_re1	1	输入:1 号口寄存器组读使能
rf_in_raddr1	5	输入:1 号口寄存器组读地址
rf_in_re2	1	输入:2 号口寄存器组读使能
rf_in_raddr2	5	输入:2 号口寄存器组读地址
rf_in_we1	1	输入：寄存器组写使能
rf_in_waddr1	5	输入：寄存器组写地址
rf_in_wdata1	32	输入：寄存器组写数据
rf_out_rdata1	32	输出:1 号口寄存器组读出数据
rf_out_rdata2	32	输出:2 号口寄存器组读出数据

（九）CTRL 模块设计

功能：介绍 if，id,exe，mem，wb 阶段发出的请求暂停信号，输出暂停控制信号，此次参考【1】

模块信号表 11　ctrl 模块信号

信号	位宽	含义
stop_from_if_i	1	输入：来自 if 阶段发出的暂停请求信号
stop_from_id_i	1	输入：来自 id 阶段发出的暂停请求信号
stop_from_exe_i	1	输入：来自 exe 阶段发出的暂停请求信号
stop_from_mem_i	1	输入：来自 mem 阶段发出的暂停请求信号
stop_from_wb_i	1	输入：来自 wb 阶段发出的

		暂停请求信号
stop_o	6	输出：发出暂停

（十）Serial 模块设计

实现串口的读写和串口发送和接收功能，

模块信号表 12　serial 模块信号

信号	位宽	含义
serial_ce_iL	1	输入：串口使能（L 表示低电平有效，）
serial_we_iL	1	输入：串口写使能
serial_rwaddr_i	1	输入：串口读写地址
serial_wdata_i	32	输入：串口写入数据
write_wb_regs_data_i ,	1	输入：Store 指令更新使能
wb_regs_data_i	32	输入：Store 指令更新数据
rxd_i	1	输入：串口接收数据
txd_o	1	输出：串口发送数据
serial_rdata_o	32	输出：串口读出数据

（十一）Data_Sram 模块设计

实现将 MIPS 的出的指令访存信息和数据访存信息转为 BaseRam 和 ExtRam 信息

模块信号表 13　Data_Sram 模块信号

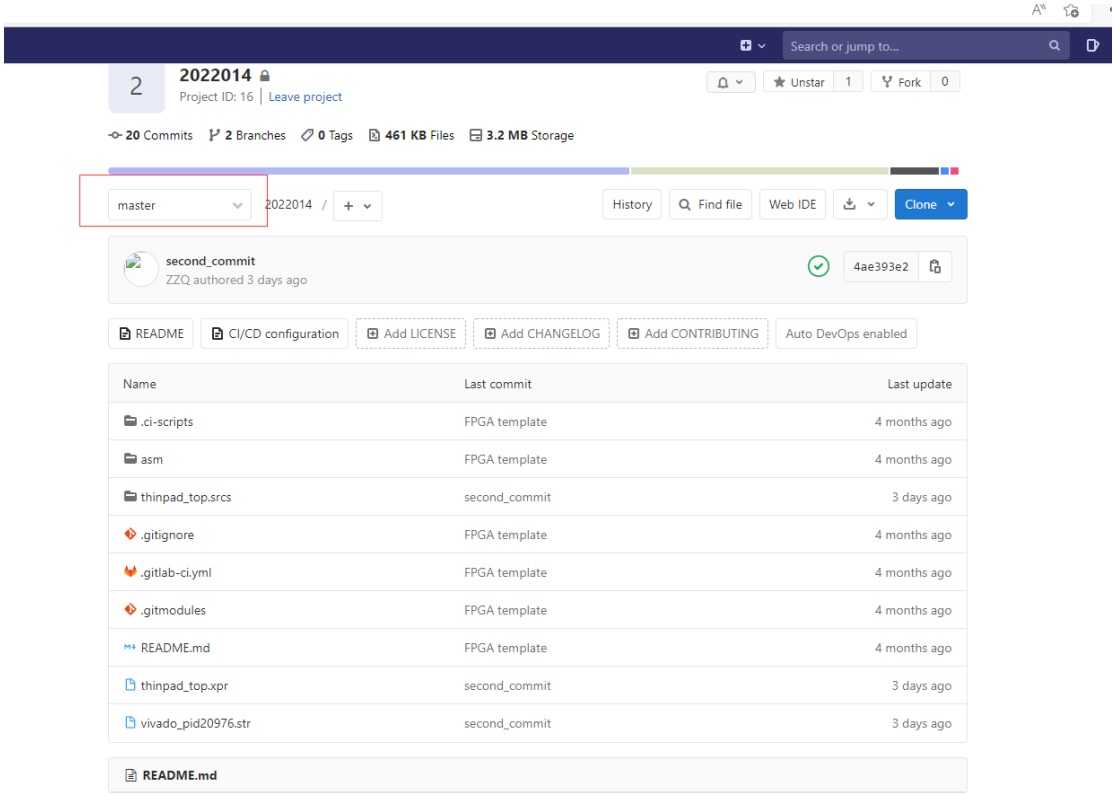
信号	位宽	含义
cpu_rom_raddr_i ,	1	输入：Cpu 发出的指令区读取地址即 pc
cpu_rom_rdata_o ,	32	输出：指令区读出指令
cpu_ram_ce_iL ,	1	输入：Cpu 发出的数据区使能
cpu_ram_we_iL ,	1	输入：Cpu 发出的数据区

		写使能
cpu_ram_sel_iL ,	4	输入：Cpu 发出的访问字节
cpu_ram_rwaddr_i ,	32	输入：Cpu 发出的访问地址
cpu_ram_wdata_i ,	32	输入：Cpu 发出的数据区写入地址
cpu_ram_rdata_o ,	32	输出数据区读出数据
write_wb_regs_data_i ,	1	输入：Store 指令更新使能
wb_regs_data_i ,	32	输入：Store 指令更新数据
ext_ram_ce_oL ,	不介绍	不做介绍
ext_ram_oe_oL ,		
ext_ram_we_oL ,		
ext_ram_addr_o ,		
ext_ram_be_oL ,		
ext_ram_rwdata_io ,		
base_ram_ce_oL ,		
base_ram_oe_oL ,		
base_ram_we_oL ,		
base_ram_addr_o ,		
base_ram_be_oL ,		
base_ram_rwdata_io		
ext_ram_ce_oL ,		
ext_ram_oe_oL ,		
ext_ram_we_oL ,		

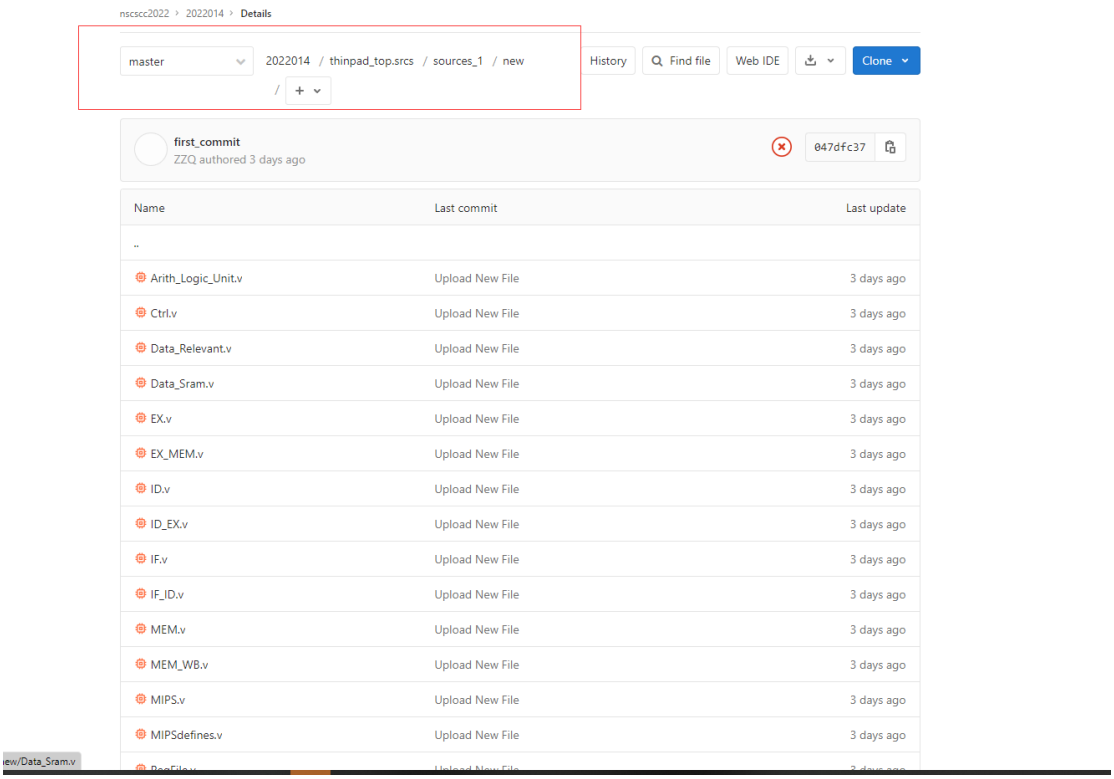
三、设计结果

（一）设计交付物说明

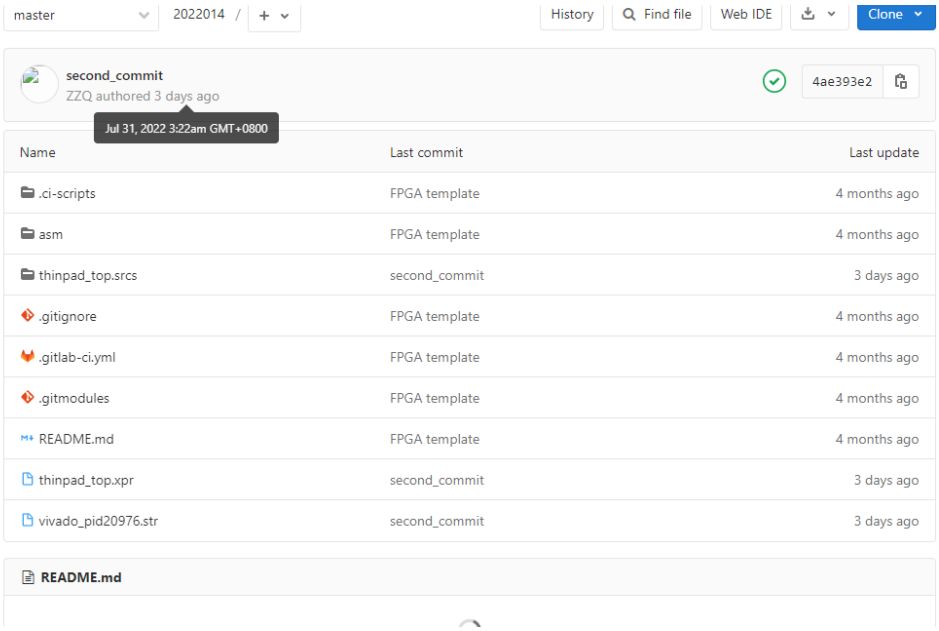
提交代码在 master 分支中



Cpu 所有模块都在此，



设计报告将放在此目录下



（二）设计演示结果


测试结果
100
perf 在 FPGA 板 01 02 03 上的结果
=== Test CRYPTONIGHT ===
Boot message: 'MONITOR for MIPS32 - initialized.'
User program written
Program Readback:
4080043cadde053cefbea534cefa063c0cb0c6341000073c25180400251000000
Program memory content verified
Data memory content verified
Test CRYPTONIGHT run for 0.461s
下载 03 的追踪数据


显示编译结果列表
测试结果
100
perf 在 FPGA 板 01 02 03 上的结果
=== Test MATRIX ===
Boot message: 'MONITOR for MIPS32 - initialized.'
User program written
Program Readback:
4080043c4180053c4280063c60000724251800001a006710804003004052030021
Program memory content verified
Data memory content verified
Test MATRIX run for 0.179s
下载 02 的追踪数据
6655
2022-07-31 03:20:08
thinpad_top.bit


选择文件
未选择文件
显示编译结果列表
测试结果
100
perf 在 FPGA 板 01 02 03 上的结果
=== Test STREAM ===
Boot message: 'MONITOR for MIPS32 - initialized.'
User program written
Program Readback:
1080043c4080053c3000063c213086000500861004 00a5240000828cfcffa2acf
Program memory content verified
Data memory content verified
Test STREAM run for 0.110s
下载 01 的追踪数据

四、参考设计说明

参考设计：

1、在设计 Sram 访问接口的参考了龙芯杯发布的 2020 比赛作品中葛启丰的作品的 sram 设计方法。龙芯教育/nscscc-wiki - Gitee.com

2、总体结构设计借鉴了《自己动手写 CPU》.(雷思磊)，

3、控制器结合了华中科技大学的 mooc 的设计结构，计算机硬件系统设计_华中科技大学_中国大学 MOOC(慕课) (icourse163.org)

五、参考文献

【1】雷思磊.《自己动手写 CPU》.