

个人赛设计报告

学校：安徽新华学院

姓名：范紫骝

目 录

一、设计简介	2
(一) 提交设计	2
(二) 特色	2
二、设计方案	2
(一) 总体设计思路	2
(二) myCPU 模块设计	4
(三) pc_reg 模块设计	5
(四) if_id 模块设计	6
(五) id 模块设计	7
(七) ex 模块设计	12
(八) ex_mem 模块设计	14
(九) mem 模块设计	17
(十) mem_wb 模块设计	19
(十一) regfile 模块设计	20
(十二) stall_ctrl 模块设计	21
(十三) extram_cache 模块设计	22
(十四) cache 模块设计	24
(十五) external_device 模块设计	26
(十六) baseram 模块设计	28
(十七) extram 模块设计	29
(十八) serial 模块设计	31
(十九) external_device_ctrl 模块设计	32
三、设计结果	33
(一) 设计交付物说明	33
(二) 设计演示结果	35
四、参考设计说明	36
五、参考文献	36

一、设计简介

（一）提交设计

顺序单发射 5 级流水线 MIPS CPU。在确保测试通过的情况下，主频最高可支持 61.79MHz。可执行大赛要求的基本 22 条指令和 3 条个人随机指令（SLT, SRA, JALR），已通过一级、二级、三级和性能测试（均 100 分）。

（二）特色

CPU 设计实现延迟槽技术、尽量减少了流水线气泡的产生、伪 Dcache（历史数据记录堆）。

1. 【说明】

设计的伪 Dcache 采用全相联的地址映射策略 和 LRU（最近最少使用）替换策略，在测试通过的最高主频下，伪 Dcache 可存储数据的容量为 16B。

2. 【解释】

之所以说是“伪 Dcache”，是因为该 cache 不具备写回功能，且该 cache 的功能是记录历史中写入 ExtRAM 中的数据，尽可能减少因为 load 数据相关现象（id 模块需要的数据存在上一条指令，且上一条指令是 load 指令，需要的数据还未从 sram 中取出）而暂停流水线的问题。

二、设计方案

（一）总体设计思路

总体设计：经典的顺序单发射 5 级流水线 CPU，大致分为 2 个部分：myCPU 模块和 external_device 模块。 myCPU 模块实现 cpu 核功能，external_device 模块为外部存储器数据访问处理模块（包含对 SRAM、串口的处理）。

myCPU 模块包含 5 个阶段：IF（取值）阶段、ID（译码）阶段、EX（执行）阶段、MEM（访存）阶段、WB（写回）阶段。除此之外，还包含每个阶段之间相应的暂存器：IF_ID、ID_EX、EX_MEM、MEM_WB，以及流水线暂停控制模块 stall_ctrl 和历史数据记录堆 extram_cache（伪 Dcache）模块。

external_device 模块包含 4 个子模块：baseram 模块、extram 模块、serial 模块以及 external_device_ctrl 模块。前三个子模块分别处理与 BaseRAM、ExtRAM、串口（UART）

的数据交互，最后一个子模块是将前三个模块中在外部储存器（或串口）中读取的数据统一处理（选择其一）送往 `cpu` 核中。

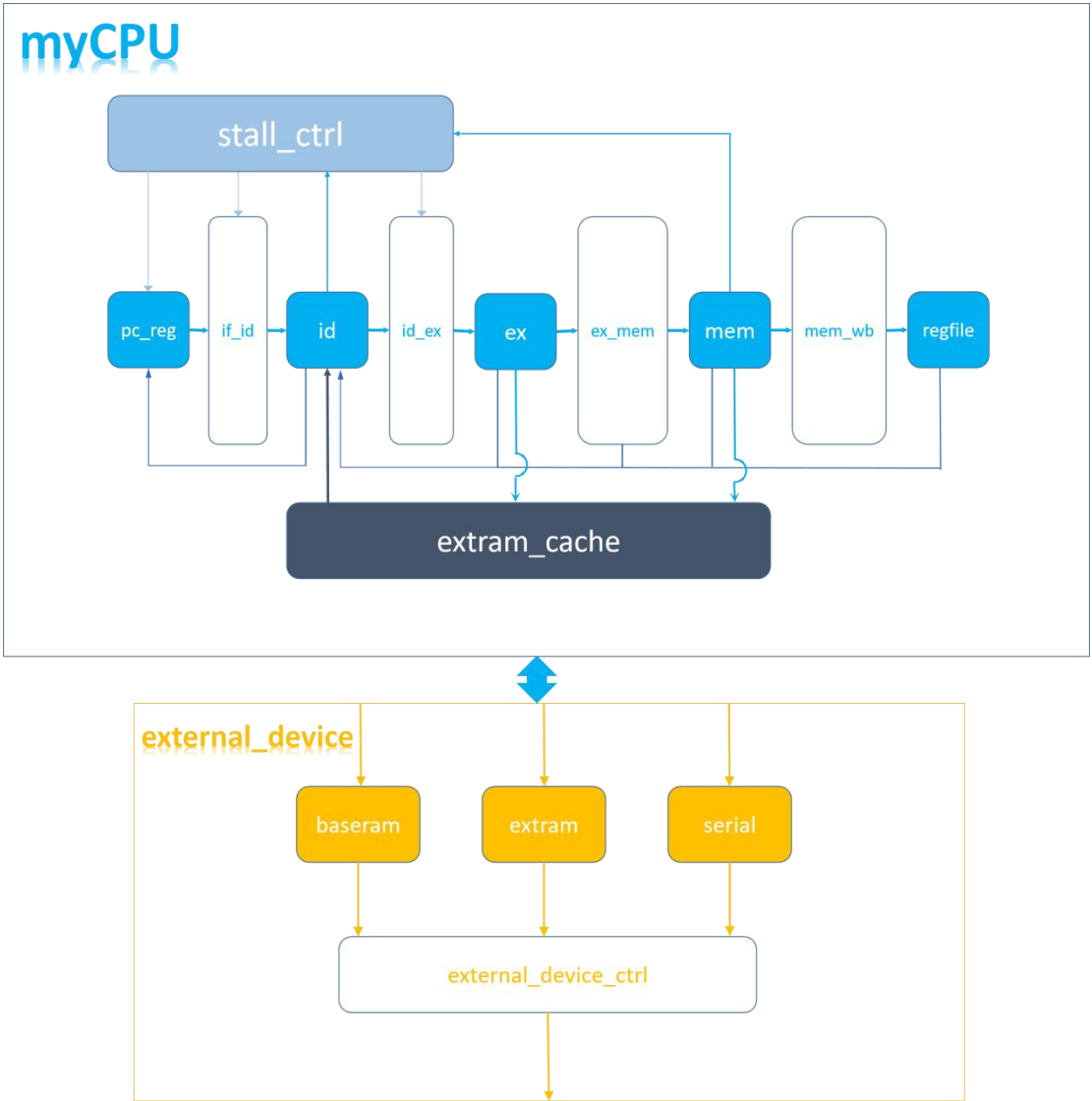


图 1 CPU 总体设计图

(二) myCPU 模块设计

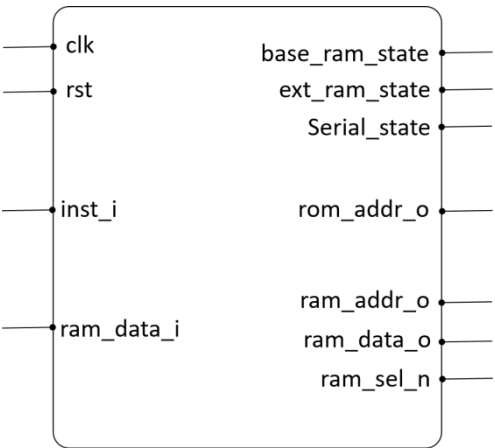


图 2 myCPU 模块端口图

模块功能:

实现 cpu 核功能，与 sram 进行信息交互，接收 sram 传来的指令和数据，传送 sram 指令地址以及 sram 的数据交互参数（如 ram 控制状态）。

表 2.1 myCPU 模块输入端口表

输入 input	端口名称	位宽	来自	作用
	clk	1bit	thinpad_top 模块	时钟端
	rst	1bit	thinpad_top 模块	复位端
	inst_i	32bit	external_device 模块	从 basesram 中读取的指令
	ram_data_i	32bit	external_device 模块	从外部存储器中读取的数据

表 2.2 myCPU 模块输出端口表

输出 output	端口名称	位宽	送往	作用
	base_ram_state	4bit	external_device 模块	访存 baseram 的状态
	ext_ram_state	3bit	external_device 模块	访存 extram 的状态
	Serial_state	4bit	external_device 模块	访存串口的状态
	rom_addr_o	32bit	external_device 模块	从 pc_reg 模块取出的指令地址
	ram_addr_o	32bit	external_device 模块	从 mem 模块取出的访存数据地址

	ram_data_o	32bit	external_device 模块	从 mem 模块取出的写入 sram 的数据
	ram_sel_n	4bit	external_device 模块	从 mem 模块取出的写入 sram 的字节选取

（三）pc_reg 模块设计

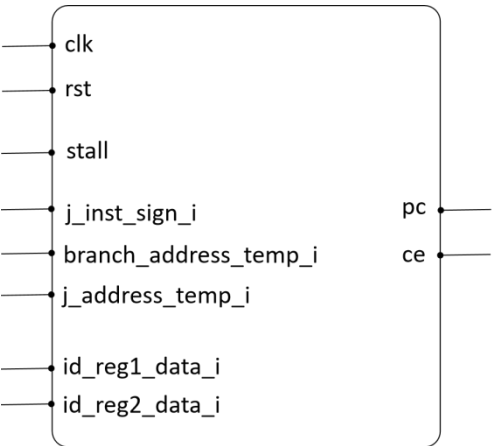


图 3 pc_reg 模块端口图

模块功能：

指令地址寄存器（取指模块），可存储指令地址。这里在译码模块中取出部分信号和数据作为转移指令是否跳转的依据，减少译码模块对于确定操作数的相关路径长度。

表 3.1 pc_reg 模块输入端口表

输入	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端
	rst	1bit	myCPU 模块	复位端
	stall	1bit	stall_ctrl 模块	暂停 if 阶段
	j_inst_sign_i	5bit	id 模块	每一位包含 1 或 2 种转移指令确认产生信号，从高位至低位依次代表 bgtz、bne、beq、jr/jalr、jal、j 指令信号
	branch_address_temp_i	32bit	id 模块	beq、bne、bgtz 分支指令跳转成功

				的转移地址
	j_address_temp_i	32bit	id 模块	j、jal、jr、jalr 跳转指令的转移地址
	id_reg1_data_i	32bit	id 模块	id（译码）模块的操作数 1，用于分支指令是否跳转的判断条件
	id_reg2_data_i	32bit	id 模块	id（译码）模块的操作数 2，用于分支指令是否跳转的判断条件

表 3.2 pc_reg 模块输出端口表

输出	端口名称	位宽	送往	作用
out	pc	32bit	if_id 模块 /myCPU 模块	指令地址，用于取指
- put	ce	1bit	mem 模块	指令使能，用于取指

（四）if_id 模块设计

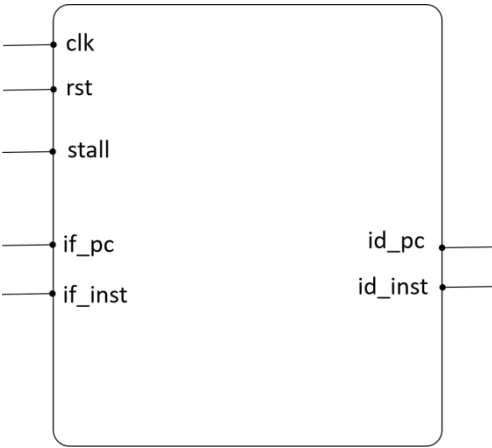


图 4 if_id 模块端口图

模块功能：

取指阶段和译码阶段的信号暂存器。

表 4.1 if_id 模块输入端口表

输入	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端

in- put	rst	1bit	tmyCPU 模块	复位端
	stall	1bit	stall_ctrl 模块	暂停 if_id 模块
	if_pc	32bit	pc_reg 模块	-
	if_inst	32bit	pc_reg 模块	-

表 4.2 if_id 模块输出端口表

输出 - put	端口名称	位宽	送往	作用
	id_pc	32bit	id 模块	-
	id_inst	32bit	id 模块	-

（五）id 模块设计

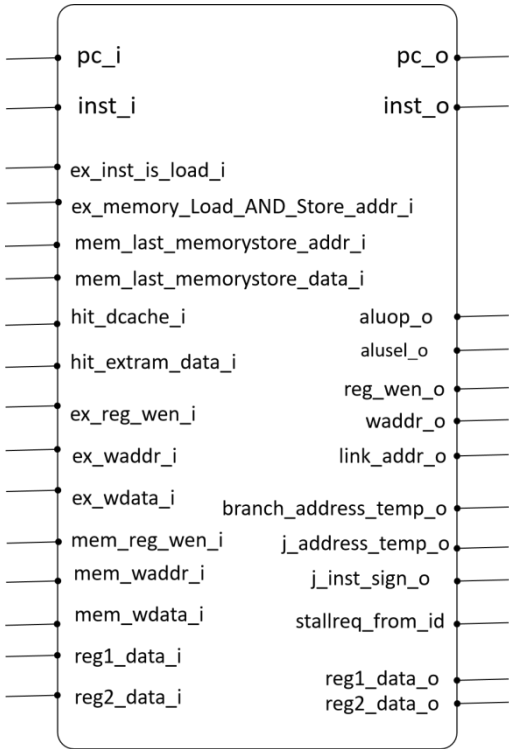


图 5 id 模块端口图

模块功能:

译码模块, 对指令进行译码操作, 模块代码里使用了分类识别的方法, 即先确认指令的类型, 对传入执行模块的信号参数进行修改, 再进一步确定具体是哪条指令, 最后对一些专用指令的信号进行修改, 这样有效减少了代码复杂度。

表 5.1 id 模块输入端口表

输入 input	端口名称	位宽	来自	作用
	pc_i	32bit	if_id 模块	-
	inst_i	32bit	if_id 模块	-
	ex_inst_is_load_i	1bit	ex 模块	ex 模块执行 load 指令的标志
	ex_memory_Load_AND_Store_addr_i	32bit	ex 模块	ex 模块执行 load 或 store 指令时的访存地址
	mem_last_memorystore_addr_i	32bit	ex_mem 模块	ex_mem 暂存器记录的最近一次存储的地址, 用于解决数据相关问题
	mem_last_memorystore_data_i	32bit	ex_mem 模块	ex_mem 暂存器记录的最近一次存储的数据, 用于解决数据相关问题
	hit_dcache_i	1bit	extram_cache 模块	命中伪 dcache 的标志, 用于解决数据相关问题
	hit_extram_data_i	32bit	extram_cache 模块	命中伪 dcache 的数据, 用于解决数据相关问题
	ex_reg_wen_i	1bit	ex 模块	ex 模块执行指令写寄存器标志, 用于解决数据相关问题
	ex_waddr_i	5bit	ex 模块	ex 模块执行指令写寄存器的地址, 用于解决数据相关问题

	ex_wdata_i	32bit	ex 模块	ex 模块执行指令写寄存器的数据，用于解决数据相关问题
	mem_reg_wen_i	1bit	mem 模块	mem 模块执行指令写寄存器标志，用于解决数据相关问题
	mem_waddr_i	5bit	mem 模块	mem 模块执行指令写寄存器的地址，用于解决数据相关问题
	mem_wdata_i	32bit	mem 模块	mem 模块执行指令写寄存器的数据，用于解决数据相关问题
	reg1_data_i	32bit	regfile 模块	不发生数据相关的寄存器操作数 1
	reg2_data_i	32bit	regfile 模块	不发生数据相关的寄存器操作数 2

表 5.2 id 模块输出端口表

输出 out -	端口名称	位宽	送往	作用
	pc_o	32bit	id_ex 模块	-
	inst_o	32bit	id_ex 模块	-
	aluop_o	3bit	id_ex 模块	运算的子类型（准确类型）
	alusel_o	3bit	id_ex 模块	运算的类型，例如逻辑、算数、移位、访存
	reg1_data_o	32bit	id_ex 模块/ pc_reg 模块	源操作数 1
	reg2_data_o	32bit	id_ex 模块/ pc_reg 模块	源操作数 2
	reg_wen_o	1bit	id_ex 模块	写寄存器使能

	waddr_o	5bit	id_ex 模块	写寄存器地址
	reg1_ren_o	1bit	id_ex 模块	读寄存器数据 1 使能
	reg2_ren_o	1bit	id_ex 模块	读寄存器数据 2 使能
	reg1_addr_o	5bit	id_ex 模块	读寄存器数据 1 的地址
	reg2_addr_o	5bit	id_ex 模块	读寄存器数据 2 的地址
	j_inst_sign_o	5bit	pc_reg 模块	每一位包含 1 或 2 种转移指令确认产生信号，从高位至低位依次代表 bgtz、bne、beq、jr/jalr、jal、j 指令信号
	j_address_temp_o	32bit	pc_reg 模块	beq、bne、bgtz 分支指令跳转成功的转移地址
	branch_address_temp_o	32bit	pc_reg 模块	j、jal、jr、jalr 跳转指令的转移地址
	link_addr_o	32bit	id_ex 模块	寄存 jal, jalr 指令专用写寄存器的值
	stallreq_from_id	1bit	stall_ctrl 模块	id（译码）模块存在 load 数据相关，需要发射流水线暂停信号

（六）id_ex 模块设计

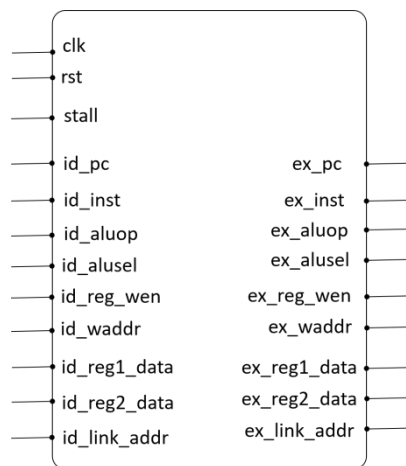


图 6 id_ex 模块端口图

模块功能：

译码阶段和执行阶段的信号暂存器。

表 6.1 id_ex 模块输入端口表

输入 input	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端
	rst	1bit	myCPU 模块	复位端
	stall	1bit	stall_ctrl 模块	暂停 id_ex 模块
	id_pc	32bit	id 模块	-
	id_inst	32bit	id 模块	-
	id_aluop	3bit	id 模块	-
	id_alusel	3bit	id 模块	-
	id_reg_wen	1bit	id 模块	-
	id_waddr	5bit	id 模块	-
	id_reg1_data	32bit	id 模块	-
	id_reg2_data	32bit	id 模块	-
	id_link_addr	32bit	id 模块	-

表 6.2 id_ex 模块输出端口表

输出 output	端口名称	位宽	送往	作用
	ex_pc	32bit	ex 模块	-
	ex_inst	32bit	ex 模块	-
	ex_aluop	3bit	ex 模块	-
	ex_alusel	3bit	ex 模块	-
	ex_reg_wen	1bit	ex 模块	-

	ex_waddr	5bit	ex 模块	-
	ex_reg1_data	32bit	ex 模块	-
	ex_reg2_data	32bit	ex 模块	-
	ex_link_addr	32bit	ex 模块	-

（七）ex 模块设计

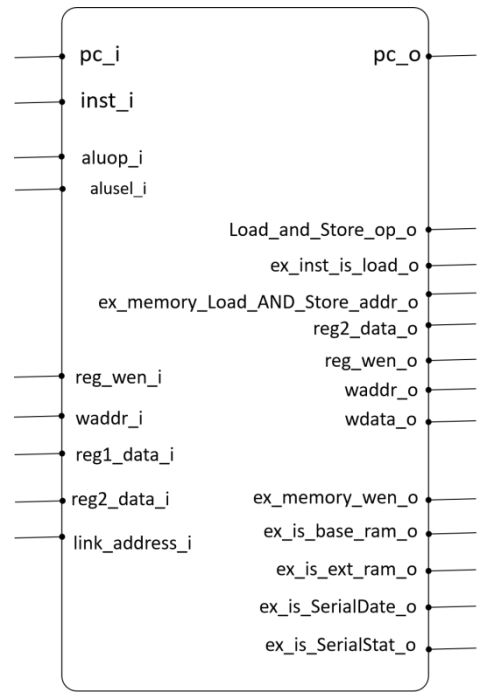


图 7 ex 模块端口图

模块功能：

执行模块，对译码阶段的操作数进行运算，这里特别将访存阶段的访存 sram 数据在这里提前计算，这是为了避免在访存阶段计算时会延迟得到从 sram 传来的数据，间接需要降频才能够使得有足够的时间得到正确的 sram 数据。

表 7.1 ex 模块输入端口表

输	端口名称	位宽	来自	作用
入	pc_i	32bit	id_ex 模块	-

input	inst_i	32bit	id_ex 模块	-
	alusel_i	3bit	id_ex 模块	确定执行运算的类型
	aluop_i	3bit	id_ex 模块	确定执行运算的具体类型
	reg_wen_i	1bit	id_ex 模块	写寄存器使能
	waddr_i	5bit	id_ex 模块	写寄存器地址
	reg1_data_i	32bit	id_ex 模块	操作数 1
	reg2_data_i	32bit	id_ex 模块	操作数 2
	link_address_i	32bit	id_ex 模块	寄存 jal, jalr 指令专用 写寄存器的值

表 7.2 ex 模块输出端口表

输出 output	端口名称	位宽	送往	作用
	pc_o	32bit	ex_mem 模块	-
	Load_and_Store_op_o	4bit	ex_mem 模块	准确识别为加载指令和存储指令的类型
	ex_inst_is_load_o	3bit	ex_mem 模块 /id 模块 /extram_cache 模块	ex 阶段执行 load 类指令标志, 供 id (译码) 模块 load 竞争冒险 (数据相关) 条件判断
	ex_memory_Load_AND_Store_addr_o	32bit	ex_mem 模块 /id 模块 /extram_cache 模块	加载指令或存储指令访存地址
	reg2_data_o	32bit	ex_mem 模块	源操作数 2, 用

				于存储指令写入 ram 的数据
	reg_wen_o	1bit	ex_mem 模块 /id 模块	写寄存器使能
	waddr_o	5bit	ex_mem 模块 /id 模块	写寄存器地址
	wdata_o	32bit	ex_mem 模块 /id 模块	写寄存器数据
	ex_memory_wen_o	1bit	ex_mem 模块	提前在执行阶段 判断访存写状 态，这里低电平 有效
	ex_is_base_ram_o	1bit	ex_mem 模块	提前判断访存地 址范围是否在 baseram
	ex_is_ext_ram_o	1bit	ex_mem 模块	提前判断访存地 址范围是否在 extram
	ex_is_SerialDate_o	1bit	ex_mem 模块	提前判断访存地 址范围是否在串 口（接受数据）
	ex_is_SerialStat_o	1bit	ex_mem 模块	提前判断访存地 址范围是否在串 口（状态）

（八）ex_mem 模块设计

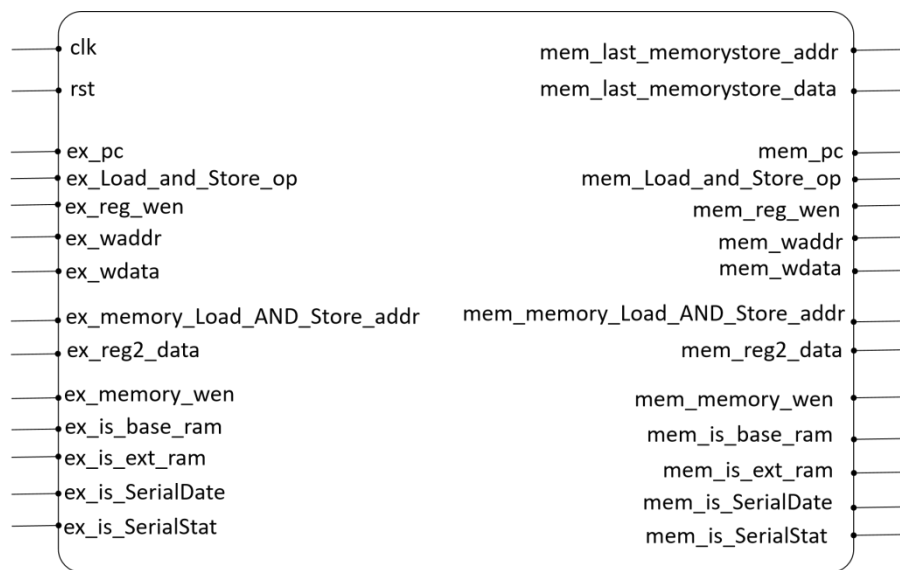


图 8 ex_mem 模块端口图

模块功能:

执行阶段和访存阶段的信号暂存器，这里为了修改因伪 dcache 延时 1 周期写入数据，而读取刚写入数据时却无法命中的问题，在这里特别做了“最近一次写入 ram”的记录变量。

表 8.1 ex_mem 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端
	rst	1bit	myCPU 模块	复位端
	ex_pc	32bit	ex 模块	-
	ex_Load_and_Store_op	4bit	ex 模块	-
	ex_reg_wen	1bit	ex 模块	-
	ex_waddr	5bit	ex 模块	-
	ex_wdata	32bit	ex 模块	-
	ex_memory_Load_AND_Store_addr	32bit	ex 模块	-
	ex_reg2_data	32bit	ex 模块	-
	ex_memory_wen	1bit	ex 模块	-

	ex_is_base_ram	1bit	ex 模块	-
	ex_is_ext_ram	1bit	ex 模块	-
	ex_is_SerialDate	1bit	ex 模块	-
	ex_is_SerialStat	1bit	ex 模块	-

表 8.2 ex_mem 模块输出端口表

输出 - put	端口名称	位宽	送往	作用
	mem_pc	32bit	mem 模块	-
	mem_Load_and_Store_op	4bit	mem 模块	-
	mem_reg_wen	1bit	mem 模块	-
	mem_waddr	5bit	mem 模块	-
	mem_wdata	32bit	mem 模块	-
	mem_memory_Load_AND_Store_addr	32bit	mem 模块	-
	mem_reg2_data	32bit	mem 模块	-
	mem_memory_wen	1bit	mem 模块	-
	mem_is_base_ram	1bit	mem 模块	-
	mem_is_ext_ram	1bit	mem 模块	-
	mem_is_SerialDate	1bit	mem 模块	-
	mem_is_SerialStat	1bit	mem 模块	-
	mem_last_memorystore_addr	32bit	id 模块	-
	mem_last_memorystore_data	32bit	id 模块	-

（九）mem 模块设计

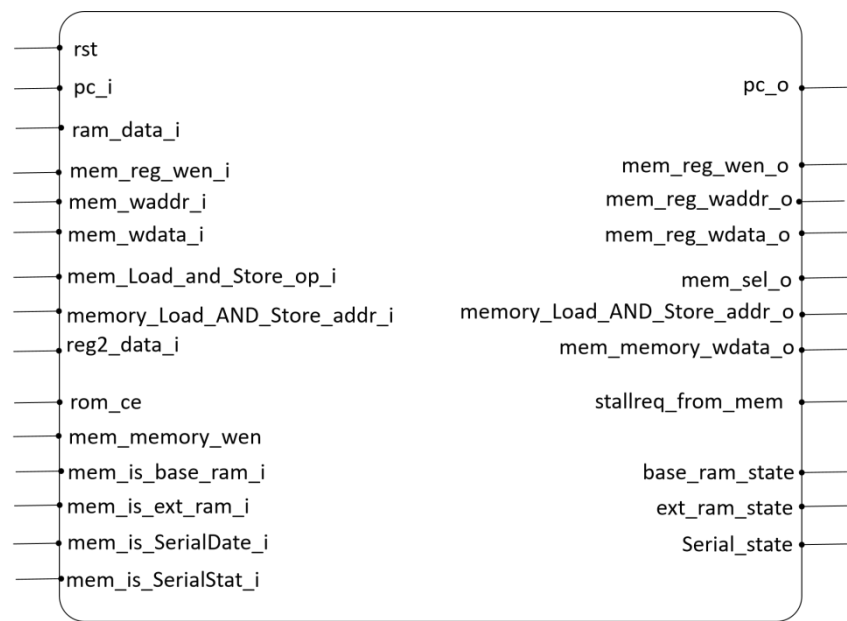


图 9 mem 模块端口图

模块功能：

访存模块，实现了对 baseram、extram、串口的访存状态机，可从外部存储器做数据交互。

表 9.1 mem 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	rst	1bit	myCPU 模块	复位端
	pc_i	32bit	ex_mem 模块	-
	ram_data_i	32bit	myCPU 模块	load 指令读取的数据
	mem_reg_wen_i	1bit	ex_mem 模块	写寄存器使能
	mem_waddr_i	5bit	ex_mem 模块	写寄存器地址
	mem_wdata_i	32bit	ex_mem 模块	写寄存器地址
	mem_Load_and_Store_op_i	4bit	ex_mem 模块	访存指令码

	memory_Load_AND_Store_addr_i	32bit	ex_mem 模块	访存地址
	reg2_data_i	32bit	ex_mem 模块	store 指令写入的数据
	rom_ce	1bit	pc_reg 模块	取指使能
	mem_memory_wen	1bit	ex_mem 模块	写存储器使能, 这里低电平有效
	mem_is_base_ram_i	1bit	ex_mem 模块	访存地址范围是否在 baseram
	mem_is_ext_ram_i	1bit	ex_mem 模块	访存地址范围是否在 extram
	mem_is_SerialDate_i	1bit	ex_mem 模块	访存地址范围是否在串口 (接收数据)
	mem_is_SerialStat_i	1bit	ex_mem 模块	访存地址范围是否在串口 (状态)

表 9.2 mem 模块输出端口表

输出 out - put	端口名称	位宽	送往	作用
	pc_o	32bit	mem_wb 模块	-
	mem_reg_wen_o	1bit	mem_wb 模块 /id 模块	写寄存器使能
	mem_reg_waddr_o	5bit	mem_wb 模块 /id 模块	写寄存器地址
	mem_reg_wdata_o	32bit	mem_wb 模块 /id 模块	写寄存器数据
	mem_sel_o	4bit	mem_wb 模块	访存外部存储器的字节选取
	memory_Load_AND_Store_addr_o	32bit	myCPU 模块	访存地址

	mem_memory_wdata_o	32bit	myCPU 模块	写存储器的数据
	stallreq_from_mem	1bit	stall_ctrl 模块	mem（访存）模块存在对 baseram 访存的结构冲突，需要发射流水线暂停信号
	base_ram_state	1bit	myCPU 模块 /extram_cache 模块	访存 baseram 的状态
	ext_ram_state	1bit	myCPU 模块 /extram_cache 模块	访存 extram 的状态
	Serial_state	1bit	myCPU 模块 /extram_cache 模块	访存串口的状态

（十）mem_wb 模块设计

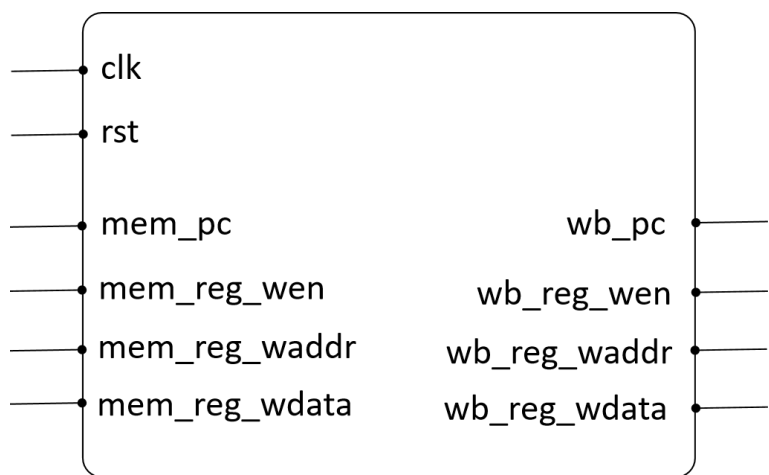


图 10 mem_wb 模块端口图

模块功能：

访存阶段和写回阶段的信号暂存器。

表 10.1 mem_wb 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端
	rst	1bit	myCPU 模块	复位端
	mem_pc	32bit	mem 模块	-
	mem_reg_wen	1bit	mem 模块	-
	mem_reg_waddr	5bit	mem 模块	-
	mem_reg_wdata	32bit	mem 模块	-

表 10.2 mem_wb 模块输出端口表

输出 out - put	端口名称	位宽	送往	作用
	wb_pc	32bit	regfile 模块	-
	wb_reg_wen	1bit	regfile 模块	-
	wb_reg_waddr	5bit	regfile 模块	-
	wb_reg_wdata	32bit	regfile 模块	-

（十一）regfile 模块设计

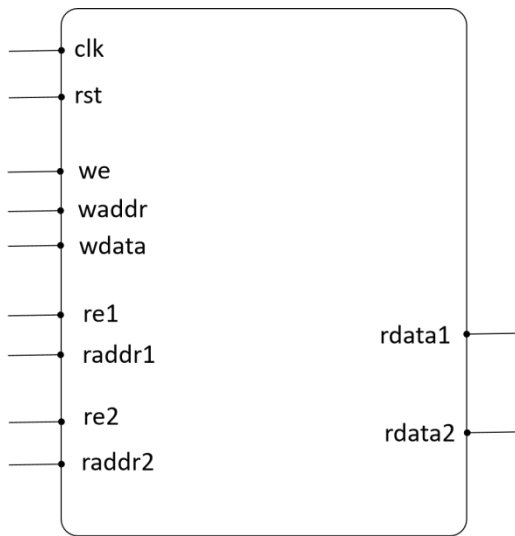


图 11 regfile 模块端口图

模块功能:

寄存器堆（写回模块），双端口读数据寄存器文件，可同时进行读出 2 个操作数。

表 11.1 regfile 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端
	rst	1bit	myCPU 模块	复位端
	we	1bit	mem_wb 模块	写寄存器堆使能
	waddr	5bit	mem_wb 模块	写寄存器堆地址
	wdata	32bit	mem_wb 模块	写寄存器堆数据
	re1	1bit	id 模块	读数据 1 使能
	raddr1	5bit	id 模块	读数据 1 的地址
	re2	1bit	id 模块	读数据 2 使能
	raddr2	5bit	id 模块	读数据 2 的地址

表 11.2 regfile 模块输出端口表

输出 out - put	端口名称	位宽	送往	作用
	rdata1	32bit	id 模块	读数据 1
	rdata2	32bit	id 模块	读数据 2

（十二）stall_ctrl 模块设计

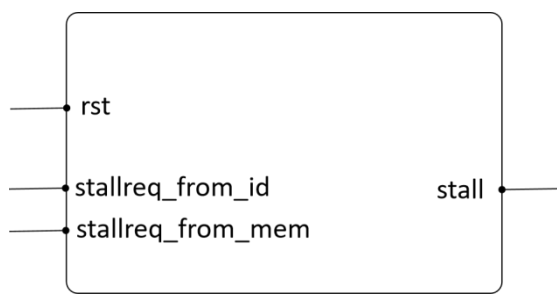


图 12 stall_ctrl 模块端口图

模块功能：

流水线暂停控制模块，设计减少了气泡的产生，保留了必要的暂停流水线带来的气泡，极限提高了 ipc。只有当译码模块产生“load 数据相关”或访存模块产生“baseram 结构访问冲突”才会发送暂停信号。

表 12.1 stall_ctrl 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	rst	1bit	myCPU 模块	复位端
	stallreq_from_id	1bit	id 模块	id 模块暂停流水线信号
	stallreq_from_mem	1bit	mem 模块	mem 模块暂停流水线信号

表 12.2 stall_ctrl 模块输出端口表

输出 out - put	端口名称	位宽	送往	作用
	stall	1bit	pc_reg 模块 /if_id 模块 /id_ex 模块	暂停流水线统一信号

（十三）extram_cache 模块设计

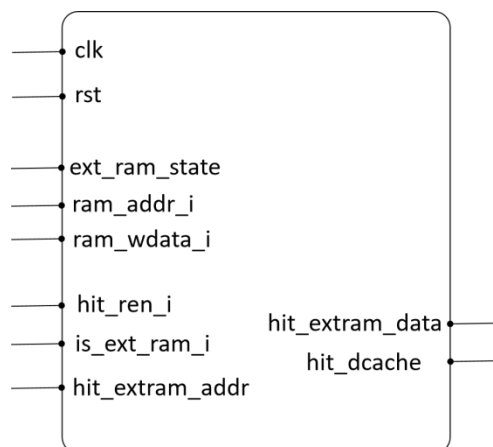


图 13 extram_cache 模块端口图

模块功能:

尽可能减少 id 模块产生数据相关现象且 ex 模块执行 load 指令而造成暂停流水线产生气泡问题。

表 13.1 extram_cache 模块输入端口表

输入 input	端口名称	位宽	来自	作用
	clk	1bit	myCPU 模块	时钟端
	rst	1bit	myCPU 模块	复位端
	ext_ram_state	3bit	mem 模块	用作传输至 extram 模块判断 extram 访问状态，ext_ram_state[2]代表对 extram 进行写数据操作，这里主要是借用这一位，在对 extram 写操作的同时，对 extram_cache 也进行写数据操作
	ram_addr_i	32bit	mem 模块	用作对 extram 写操作时的地址，同时也是对 extram_cache 写操作时保存记录数据在 ram 的地址，并非在 cache 中的写操作的地址
	ram_wdata_i	32bit	mem 模块	用作对 extram 写操作时的写入数据，同时也是对 extram_cache 写操作时保存记录的数据
	hit_ren_i	1bit	ex 模块	对应 ex_inst_is_load 信号，即在 ex 模块执行 load 指令时就对 cache 进行读取数据的操作

				尽可能减少 id 模块产生数据相关现象且 ex 模块执行 load 指令而造成暂停流水线问题
	is_ext_ram	1bit	ex 模块	判断 ex 模块执行 load 指令时访问的地址是否在 ExtRAM 上,这里是辅助上个端口 hit_ren_i 进行读操作,因为设计时避免相关路径过长, hit_extram_addr 是直接来自 ex 阶段计算出的访存地址,而并非确指访存 ExtRAM 的地址,为确保正确性,故加此端口
	hit_extram_addr	32bit	ex 模块	ex 阶段计算出的访存地址,而并非确指访存 ExtRAM 的地址

表 13.2 extram_cache 模块输出端口表

输出	端口名称	位宽	送往	作用
out	hit_extram_data	32bit	id 模块	命中伪 dcache 的数据
put	hit_dcache	1bit	id 模块	命中伪 dcache 的信号

（十四）cache 模块设计

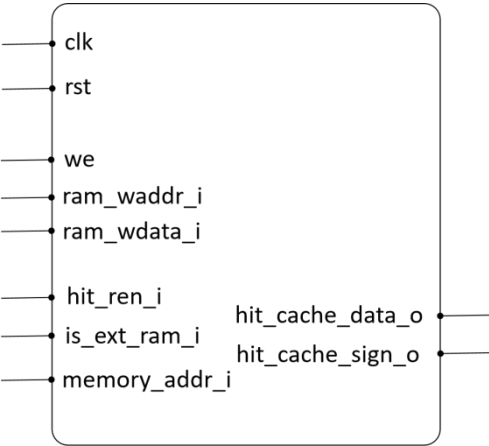


图 14 cache 模块端口图

模块功能:

作为 extram_cache 模块的存储数据模块,也可以说是历史数据记录堆,本质还是寄存器堆。

该伪 cache 具有延时 1 周期写入数据特点(在前面的 ex_mem 模块中提到过,并给出了优化办法)。该特点是因为,每个数据块(即寄存器),配有一个计数器,当写入信号为 1 时,需要得到写入寄存器地址(即寄存器号),由于设计没办法立即判断,只好在该周期时钟下降沿修改需要写入的寄存器号,在下一个时钟上升沿才会将数据写入寄存器里。

表 14.1 cache 模块输入端口表

输入 input	端口名称	位宽	来自	作用
	clk	1bit	extram_cache 模块	时钟端
	rst	1bit	extram_cache 模块	复位端
	we	1bit	extram_cache 模块	对 extram_cache 进行写数据操作
	ram_waddr_i	32bit	extram_cache 模块	对 extram_cache 写操作时保存记录数据在 ram 的地址,并非在 cache 中的写操作的地址
	ram_wdata_i	32bit	extram_cache 模块	对 extram_cache 写操作时保存记录的数据
	hit_ren_i	1bit	extram_cache 模块	ex 模块执行 load 指令时就对 cache 进行读取数据的操作信号,并非单纯的读信号
	is_ext_ram_i	1bit	extram_cache 模块	判断 ex 模块执行 load 指令时访问的地址是否在 ExtRAM 上
	memory_addr_i	32bit	extram_cache 模块	ex 阶段计算出的访存地址,而并非确指访存 ExtRAM 的地址

表 14.2 cache 模块输出端口表

输出 output	端口名称	位宽	送往	作用
	hit_cache_data_o	32bit	extram_cache 模块	命中伪 cache 的数据
	hit_cache_sign_o	1bit	extram_cache 模块	命中伪 cache 的信号

(十五) external_device 模块设计

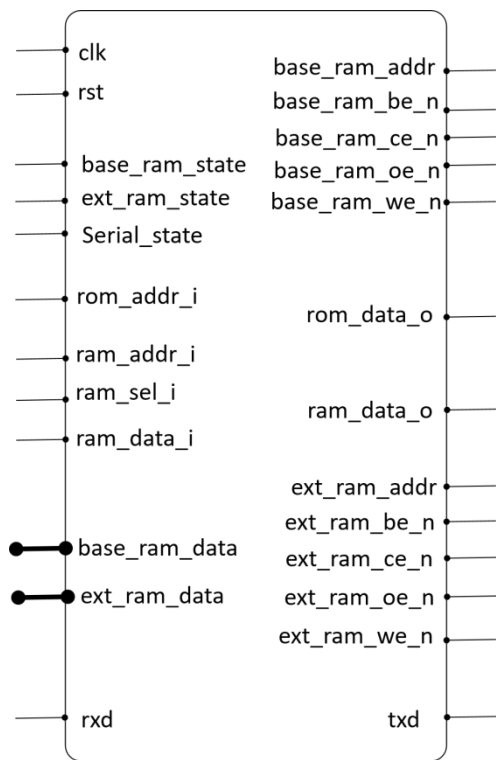


图 15 external_device

模块功能:

外部存储器模块，可与 sram、串口的数据交互。

表 15.1 external_device 模块输入端口表

输入	端口名称	位宽	来自	作用
	clk	1bit	thinpad_top 模块	时钟端
	rst	1bit	thinpad_top 模块	复位端
	base_ram_state	4bit	myCPU 模块	访存 baseram 的状态
	ext_ram_state	3bit	myCPU 模块	访存 extram 的状态
	Serial_state	4bit	myCPU 模块	访存串口的状态
	rom_addr_i	32bit	myCPU 模块	pc 指令地址
	ram_addr_i	32bit	myCPU 模块	访存 sram 地址
	ram_sel_i	4bit	myCPU 模块	访存地址字节选取，这里低电平有效
	ram_data_i	32bit	myCPU 模块	访存数据
	rxd	1bit	thinpad_top 模块	串口接收端

表 15.2 external_device 模块输出端口表

输出 - put	端口名称	位宽	送往	作用
	rom_data_o	32bit	myCPU 模块	读取的指令，送至 if_id 模块
	ram_data_o	1bit	myCPU 模块	读取的数据，送至 mem 模块
	base_ram_addr	20bit	thinpad_top 模块	访存 baseram 的地址
	base_ram_be_n	4bit	thinpad_top 模块	访存 baseram 的字节使能，低电平有效
	base_ram_ce_n	1bit	thinpad_top 模块	访存 baseram 的片选信号，低电平有效
	base_ram_oe_n	1bit	thinpad_top 模块	访存 baseram 的读使能，低电平有效
	base_ram_we_n	1bit	thinpad_top 模块	访存 baseram 的写使能，低电平有效
	ext_ram_addr	20bit	thinpad_top 模块	访存 extram 的地址
	ext_ram_be_n	4bit	thinpad_top 模块	访存 extram 的字节使能，低电平有效
	ext_ram_ce_n	1bit	thinpad_top 模块	访存 extram 的片选信号，低电平有效
	ext_ram_oe_n	1bit	thinpad_top 模块	访存 extram 的读使能，低电平有效
	ext_ram_we_n	1bit	thinpad_top 模块	访存 extram 的写使能，低电平有效
	txd	1bit	thinpad_top 模块	串口发送端

表 15.3 external_device 模块双向端口表

双向 输入 输出 in- out	端口名称	位宽	来自/送往	作用
	base_ram_data	32bit	thinpad_top 模块	访存 baseram 的数据
	ext_ram_data	32bit	thinpad_top 模块	访存 extram 的数据

(十六) baseram 模块设计

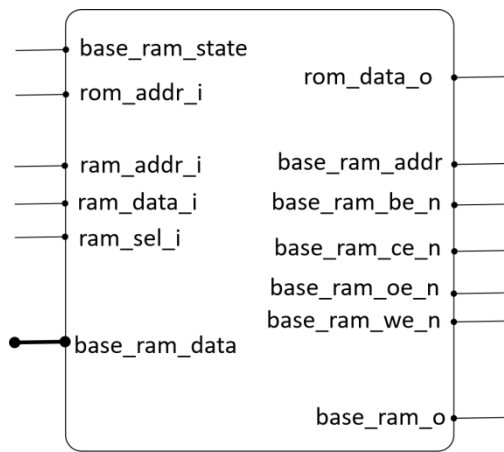


图 16 baseram 模块端口图

模块功能：

baseram 模块，可取指也可读写数据，对应 baseram 存储器的数据交互。

表 16.1 baseram 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	base_ram_state	4bit	external_device 模块	访存 baseram 的状态
	rom_addr_i	32bit	external_device 模块	读取 baseram 的指令地址
	ram_addr_i	1bit	external_device 模块	访存 baseram 的数据地址
	ram_data_i	32bit	external_device 模块	访存 baseram 的数据
	ram_sel_i	4bit	external_device 模块	访存 baseram 的字节选取

表 16.2 baseram 模块输出端口表

输出 out	端口名称	位宽	送往	作用
	rom_data_o	32bit	external_device 模块	读取的指令
	base_ram_addr	20bit	external_device 模块	访存 baseram 的地址

- put	base_ram_be_n	4bit	external_device 模块	访存 baseram 的字节使能，低电平有效
	base_ram_ce_n	1bit	external_device 模块	访存 baseram 的片选信号，低电平有效
	base_ram_oe_n	1bit	external_device 模块	访存 baseram 的读使能，低电平有效
	base_ram_we_n	1bit	external_device 模块	访存 baseram 的写使能，低电平有效
	base_ram_o	32bit	external_device_ctrl 模块	读取 baseram 的数据

表 16.3 baseram 模块双向端口表

双向 输入 输出 in- out	端口名称	位宽	来自/送往	作用
	base_ram_data	32bit	external_device 模块	访存 baseram 的数据

(十七) extram 模块设计

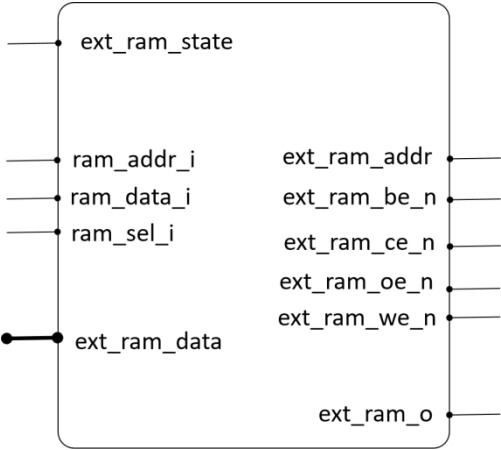


图 17 extram 模块端口图

模块功能：

extram 模块，可读写数据，对应 extram 存储器的数据交互。

表 17.1 extram 模块输入端口表

	端口名称	位宽	来自	作用
输入 in- put	ext_ram_state	4bit	external_device 模块	访存 extram 的状态
	ram_addr_i	1bit	external_device 模块	访存 extram 的数据地址
	ram_data_i	32bit	external_device 模块	访存 extram 的数据
	ram_sel_i	4bit	external_device 模块	访存 extram 的字节选取

表 17.2 extram 模块输出端口表

	端口名称	位宽	送往	作用
输出 out - put	ext_ram_addr	20bit	external_device 模块	访存 extram 的地址
	ext_ram_be_n	4bit	external_device 模块	访存 extram 的字节使能，低电平有效
	ext_ram_ce_n	1bit	external_device 模块	访存 extram 的片选信号，低电平有效
	ext_ram_oe_n	1bit	external_device 模块	访存 extram 的读使能，低电平有效
	ext_ram_we_n	1bit	external_device 模块	访存 extram 的写使能，低电平有效
	ext_ram_o	32bit	external_device_ctrl 模块	读取 extram 的数据

表 17.3 extram 模块双向端口表

双向 输入 输出	端口名称	位宽	来自/送往	作用
	ext_ram_data	32bit	external_device 模块	访存 extram 的数据

出				
in-				
out				

（十八）serial 模块设计

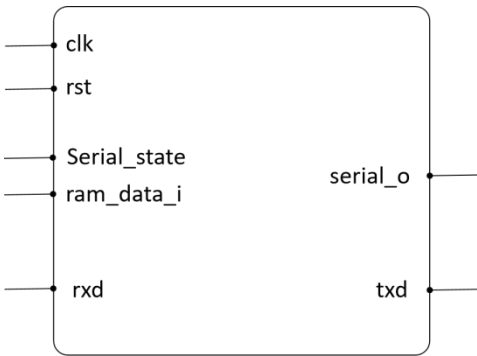


图 18 serial 模块端口图

模块功能：

serial 模块，可接收串口数据，也可读出串口工作状态。

表 18.1 serial 块输入端口表

输入	端口名称	位宽	来自	作用
	clk	1bit	external_device 模块	时钟端
	rst	1bit	external_device 模块	复位端
	Serial_state	4bit	external_device 模块	访存串口的状态
	ram_data_i	32bit	external_device 模块	访存数据
	rxd	1bit	external_device 模块	串口接收端

表 18.2 serial 块输出端口表

输出	端口名称	位宽	送往	作用
	txd	1bit	external_device 模块	串口发送端
	serial_o	32bit	external_device 模块	读取串口的数据/工作状态

put				
-----	--	--	--	--

（十九）external_device_ctrl 模块设计

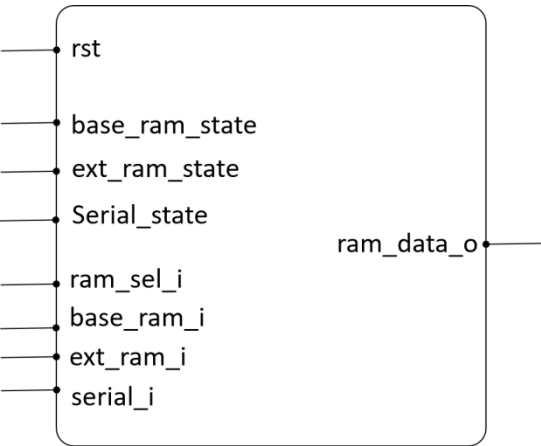


图 19 external_device_ctrl 模块端口图

模块功能：

外部存储器数据统一处理模块，将 baseram、extram、serial 三个模块读出的数据通过 sram 状态机控制选取哪个数据送回 cpu 核。

表 18.1 external_device_ctrl 模块输入端口表

输入 in- put	端口名称	位宽	来自	作用
	rst	1bit	external_device 模块	复位端
	base_ram_state	4bit	external_device 模块	访存 baseram 的状态
	ext_ram_state	3bit	external_device 模块	访存 extram 的状态
	Serial_state	4bit	external_device 模块	访存串口的状态
	ram_sel_i	4bit	external_device 模块	访存数据
	base_ram_i	32bit	baseram 模块	读取 baseram 的数据
	ext_ram_i	32bit	extram 模块	读取 extram 的数据
	serial_i	32bit	external_device 模块	读取 baseram 的数据/工作状态

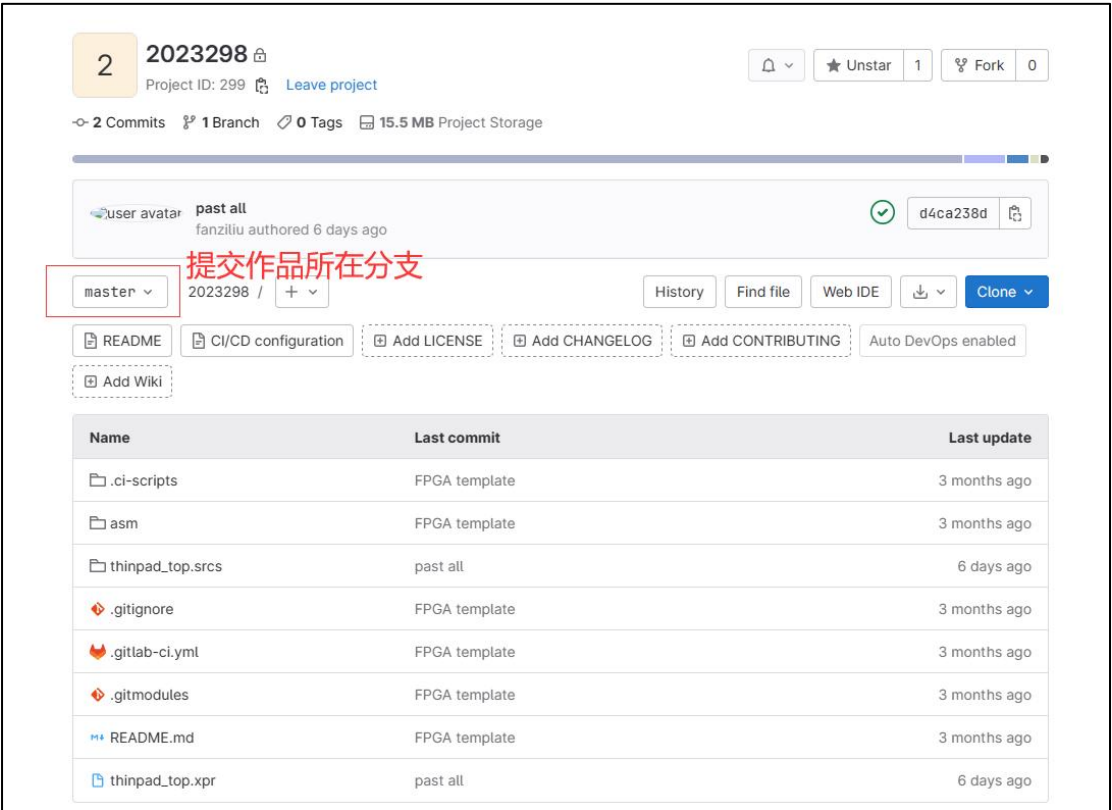
表 18.2 external_device_ctrl 模块输出端口表

输出	端口名称	位宽	送往	作用
	ram_data_o	32bit	external_device 模块	读取的数据选择其一送回 cpu 核内

out				的 mem 模块
-				
put				

三、设计结果

（一）设计交付物说明



提交代码在 master 分支中，源文件目录如下：

```
thinpad_top.srscs
├── constrs_1
│   └── new
│       └── thinpad_top.xdc //约束文件（可不关注），添加了一些延时，使得访存更稳定
├── sim_1
│   ├── imports
│   │   ├── CFImemory64Mb_bottom.mem
│   │   └── CFImemory64Mb_top.mem
```

```

|   └─ new
|   |   └─ include
|   |   └─ 28F640P30.v
|   |   └─ clock.v
|   |   └─ cpld_model.v
|   |   └─ flag_sync_cpld.v
|   |   └─ sram_model.v           //baseram 和 extram 实例化模块（可不关注）
|   |   └─ tb.sv                 //功能仿真
└─ sources_1
    └─ ip                        //所使用的 ip 核,在 vivado 的 IP Catalog 中添加修改
    |   └─ fifo                  //先进先出队列，用于优化串口输入输出
    |   └─ pll_example           //时钟分频模块
    └─ new
        └─ async.v              //串口实例化模块
        └─ baseram.v
        └─ cache.v
        └─ ex.v
        └─ ex_mem.v
        └─ external_device.v
        └─ external_device_ctrl.v
        └─ extram.v
        └─ extram_cache.v
        └─ id.v
        └─ id_ex.v
        └─ if_id.v
        └─ mem.v
        └─ mem_wb.v
        └─ myCPU.v
        └─ pc_reg.v
        └─ regfile.v

```

```

|   └─ SEG7_LUT.v
|   └─ serial.v
|   └─ stall_ctrl.v
|   └─ thinpad_top.v      //顶层文件
|   └─ vga.v
...

```

(二) 设计演示结果

性能测试总用时: 0.620s

测试结果

100

perf 在 FPGA 板 06 07 05 上的结果

=== Test STREAM ===

Boot message: 'MONITOR for MIPS32 - initialized.'

User program written

Program Readback:

1080043c4080053c3000063c21308600050086100400a5240000828cfcffa2acfi

Program memory content verified

Data memory content verified

Test STREAM run for 0.102s

下载 06 的追踪数据

测试结果

100

perf 在 FPGA 板 06 07 05 上的结果

=== Test MATRIX ===

Boot message: 'MONITOR for MIPS32 - initialized.'

User program written

Program Readback:

4080043c4180053c4280063c60000724251800001a00671080400300405203002

Program memory content verified

Data memory content verified

Test MATRIX run for 0.145s

下载 07 的追踪数据

测试结果

100

perf 在 FPGA 板 06 07 05 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 0.373s

下载 05 的追踪数据

四、参考设计说明

参考设计：

1. 在设计流水线暂停控制 `stall_ctrl` 模块、`serial`（串口）模块中访问串口代码参考了龙芯杯 NSCSCC2022 选手(西北工业大学)夏卓的作品中 `stall` 模块、`RAM_Serial_ctrl` 模块（https://github.com/xiazhuo/nsc2022_personal）
2. 总体设计借鉴了《自己动手写 CPU》（雷思磊）^[1]
3. `serial`（串口）模块中添加了 `vivado` 软件自带的 IP 核：FIFO Generator

五、参考文献

- [1]. 雷思磊. 自己动手写 CPU[M]. 电子工业出版社：雷思磊，2014.