NSCSCC2024 龙芯杯个人赛 MIPS 赛道设计报告

学校：北京科技大学

姓名：刁明轩

一、设计简介

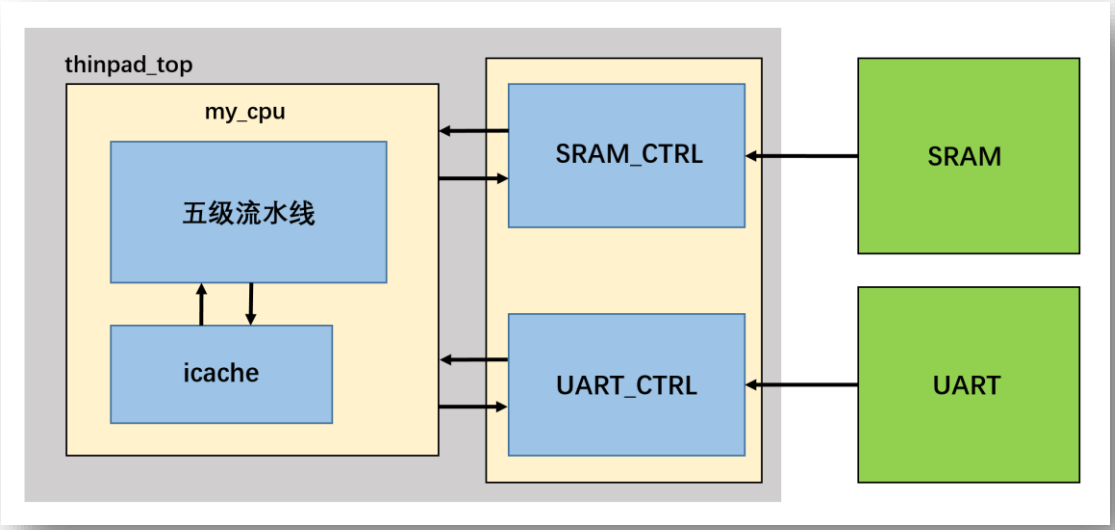
该作品实现了基于 MIPS 指令集的 32 位单发射五级流水线 CPU（取指-译码-执行-访存-回写），支持 MIPS-C1、MIPS-C2、MIPS-C3 指令集中的全部指令以及三条随机指令 SLT, SRAV, BLEZ。

该 CPU 能实现从 SRAM 中读写数据、读指令，并且在向 BaseRAM 读取数据时（而非指令）通过输出信号使 CPU 暂停。除此之外，CPU 通过数据前递解决了数据冲突问题、通过流水线暂停解决访存冲突问题、通过在译码阶段判断分支指令以及延迟槽技术解决分支指令后存在气泡指令的情况。

该作品可以通过三级功能测试和性能测试，时钟频率可以稳定运行在 55MHz，性能测试的时间为 0.114s、0.163s，0.419s。

二、设计思路和实现

该作品在五级流水的前提下添加了 SRAM 和 UART 控制模块，保证 CPU 和外设能够正确交互，同时添加 icache 进一步提高取指速度。下面是作品的各个模块框架图：



MIPS（Microprocessor without Interlocked Pipeline Stages）**是一种采取精**简指令集（RISC）**的**处理器架构，**其五**级流水线结构是 MIPS 处理器**的核心**设计。**五**级流水线**可以将指令的**执行分为五个阶段，**从而**实现指令**的并行**执行，**提高**电子元件的利用率以及处理器的性能。该项目就是对 MIPS 经典五级流水线的复现和优化。

（一）五级流水线的设计

在五级流水线中，**一条指令的**执行被分为 **取指、译码、执行、访存、回写 五个**阶段。

（1）**取指**阶段（Instruction Fetch）

该阶段可以根据 pc **地址从指令存**储器中取出，**并将其送入指令寄存器** IR **中**，**同**时，pc **地址需要根据情况自**动更新成下一条指令的地址或者跳转**指令的目的地址**。

该模块的输入输出信号如下：

类型	名称	位 宽	作用

		(bit)	
input	clk	1	时钟信号
input	rst	1	**复位信号**
input	stall	6	**流水**线暂停信号
input	branch_flag	1	**是否**发生转移
input	branch_target_address	32	转移目标地址
output	pc	32	pc **地址**
output	ce	1	**指令存**储器使能信号

（2）译码阶段（Instruction Decode）

该阶段可以分析取出的指令，**将其**转换成对应的控制信号。**在**译码阶段**中**，**通**过解析指令的操作码的组合推出指令进行的操作，**例如**读写寄存器**数据**、计算跳转地址等。译码阶段还需要准备好执行阶段需要的计算类型**和**计算数（**寄存器数据**、**立即数等**）。

该模块的输入输出信号如下：

类型	**名称**	**位** 宽 (bit)	**作用**
input	rst	1	**复位信号**
input	pc	32	pc **地址**
input	inst_data	32	**指令**
input	ex_alu_op	8	**上条指令的**计算小类
input	reg1_data	32	**寄存器** 1 **的数据**

input	reg2_data	32	寄存器 2 的数据
input	pre_ex_wreg	1	数据前推 上条指令的写使能
input	pre_ex_wdata	32	数据前推——上条指令的写数据
input	pre_ex_wd	5	数据前推——上条指令的目的寄存器号
input	pre_mem_wreg	1	数据前推——上上条指令的写使能
input	pre_mem_wdata	32	数据前推——上上条指令的写数据
input	pre_mem_wd	5	数据前推——上上条指令的目的寄存器号
input	is_in_delayslot_i	1	当前指令是否为延迟槽指令
output	reg1_oe	1	寄存器 1 读使能
output	reg2_oe	1	寄存器 2 读使能
output	reg1_addr	5	读寄存器 1 地址
output	reg2_addr	5	读寄存器 2 地址
output	alu_op	8	计算小类
output	alu_sel	3	计算大类
output	reg1_data_o	32	寄存器 1 真正的数据

			（数据前推问题）
output	reg2_data_o	32	寄存器 2 真正的数据（数据前推问题）
output	reg_waddr	5	目的寄存器地址
output	reg_we	1	寄存器写使能
output	next_inst_in_delayslot_o	1	下一条指令是否为延迟槽指令
output	branch_flag	1	是否发生跳转
output	branch_target_address	32	跳转地址
output	link_addr	32	需要保存的地址
output	is_in_delayslot_o	1	是否为延迟槽指令
output	inst_data_o	32	当前指令
output	stall_from_id	1	译码阶段是否需要暂停流水线

（3）执行阶段（Execution)

该阶段根据译码阶段的信息执行大部分指令的实际操作。执行阶段读入译码阶段计算好的源操作数和计算类型，进行相应的操作，例如逻辑运算、算术运算、地址运算等。同时执行阶段还将操作内存地址、寄存器读写数据和地址等继续送往下一级流水。

该模块的输入输出信号如下：

类型	名称	位 宽 (bit)	作用
input	rst	1	复位信号
input	alu_op_i	8	计算小类
input	alu_sel_i	3	计算大类
input	reg1_data	32	寄存器 1 的数据
input	reg2_data	32	寄存器 2 的数据
input	reg_waddr	5	目标寄存器地址
input	reg_we	1	寄存器写使能
input	link_address_i	32	需要写寄存器的地址
input	is_in_delayslot_i	1	指令是否在延迟槽
input	inst_data	32	当前指令
output	reg_waddr_o	5	目标寄存器地址
output	reg_we_o	1	寄存器写使能
output	reg_wdata_o	32	寄存器写数据
output	alu_op_o	8	计算小类
output	mem_addr_o	32	访存地址
output	reg2_data_o	32	寄存器 2 数据
output	stall_from_ex	1	执行阶段是否需要暂停流水线

（4）访存阶段（Memory Access）

该阶段负责处理指令对于内存的操作。**根据**执行阶段计算出的地址和访**存操作**（读写以及读写大小）。对于读内存指令，访存阶段从内存中读出**数据并将其送出**，对于写内存指令，访存阶段将数据写入内存。**同**时，访**存**阶段还会跟据指令操作内存的大小（读写字或字节等），输出对应的四**位字**节使能信号。

该模块的输入输出信号如下：

类型	**名称**	**位** 宽 (bit)	**作用**
input	rst	1	**复位信号**
input	reg_waddr	5	**目**标寄存器号
input	reg_we	1	**寄存器写使能**
input	reg_wdata	32	**寄存器写数据**
input	alu_op	8	计算小类
input	mem_addr_i	32	访存地址
input	reg2_data	32	**寄存器** 2 **数据**
input	mem_data_i	32	**写内存数据**
output	reg_waddr_o	5	**目**标寄存器号
output	reg_we_o	1	**寄存器写使能**
output	reg_wdata_o	32	**寄存器写数据**
output	mem_addr_o	32	访存地址
output	mem_we_o	1	访存写使能
output	mem_be_o	4	访存字节使能

output	mem_data_o	32	读内存数据
output	mem_ce_o	1	**内存使能**

（5）**回写**阶段（Write Back)

该阶段负责将特定数据写入特定寄存器。**根据前几**级流水计算出的数据**和操作信号将**结果写入目标寄存器中。**保**证之后的指令可以使用这个结果进行后续的操作。

（二）icache 的设计

该项目中还添加了 icache 进一步**提高指令的**访问效率。**由于个人**赛的 SRAM **数据**线宽度只有 32 **位**，**并且没有** AXI 总线，**故每次**访存只能读取一**个字**。**也是由于**这个原因，**本**项目并没有添加 dcache（**同**时 dcache **必**须采**用** Write-through **策略**，实际测试并没有太大的性能提升）。

基于上述背景，icache 设计思路如下：

1 **每个** cache 块大小一个字，**一共** 32 块

2 **采用直接映射方式**

3 对于指令地址，**将**[21:7]**作**为 tag，[6:2]**作**为 index（5 **位**宽满足 32 **个** cache 块的需求），[1:0]**用于**块内寻址

4 **一位有效位**

（三）SRAM_CTRL 的设计

该模块用于管理 CPU **和** SRAM **的交互**，**根据** CPU **的**访存地址确定操作 RAM **是** BaseRAM 还是 ExtRAM。**并且根据** CPU **需求**赋值 SRAM **的使能信号**。

该控制模块的输入输出信号如下：

类型	**名称**	**位 宽** (bit)	**作用**
input	clk	1	时钟信号
input	rst	1	**复位信号**
input	inst_vaddr	32	**取**值虚拟内存地址
input	inst_ce	1	**指令存**储器使能
output	inst_data	32	**指令数据**
output	ram_rdata	32	读内存数据
input	ram_vaddr	32	访存虚拟地址
input	ram_wdata	32	**写内存数据**
input	ram_we	1	**内存写使能**
input	ram_be	4	**内存字**节使能
input	ram_ce	1	**内存使能**
inout	base_ram_data	32	base 读写数据
output	base_ram_addr	32	base **操作地址**
output	base_ram_be_n	4	base **字**节使能
output	base_ram_ce_n	1	base **存**储器使能
output	base_ram_oe_n	1	base 读使能
output	base_ram_we_n	1	base **写使能**
inout	ext_ram_data	32	ext 读写数据
output	ext_ram_addr	32	ext **操作地址**

output	ext_ram_be_n	4	ext 字节使能
output	ext_ram_ce_n	1	ext 存储器使能
output	ext_ram_oe_n	1	ext 读使能
output	ext_ram_we_n	1	ext 写使能
output	stall_or_Base_relate	1	由于读写 baseRAM 中的数据（而非指令）导致的流水线暂停

（四）UART_CTRL 的设计

该模块用于实现 CPU 和 UART 串口的数据交换，根据访存地址判断 CPU 是否需要和串口交互（获取串口状态 or 获取串口数据 or 向串口发送数据），根据 CPU 的需求对串口信号赋值。本项目采用模板工程中提供的两个串口模块实现。

该控制模块的输入输出信号如下：

类型	名称	位 宽 (bit)	作用
input	clk	1	时钟信号
input	CPU_send	32	CPU 发送串口的数据
output	CPU_receive	32	CPU 接收串口的数据
input	mem_addr	32	访存地址
input	mem_we	1	内存写使能
input	rxd	1	串口信号
output	txd	1	串口信号

output	stall_for_Uart_relate	1	和 Uart 进行交互导致的流水线暂停

三、设计结果

（一）设计运行结果

一级测评结果：

二级测评结果：

三级测评结果：

性能测试结果：

提交截止时间：2024-08-05 00:00:00

最终版本标记时间：2024-07-23 20:16:56

任务 ID	提交时间	版本名称	状态	得分
44119	2024-07-23 20:15:34	e479bd16	Finished	100

性能测试时间：

100

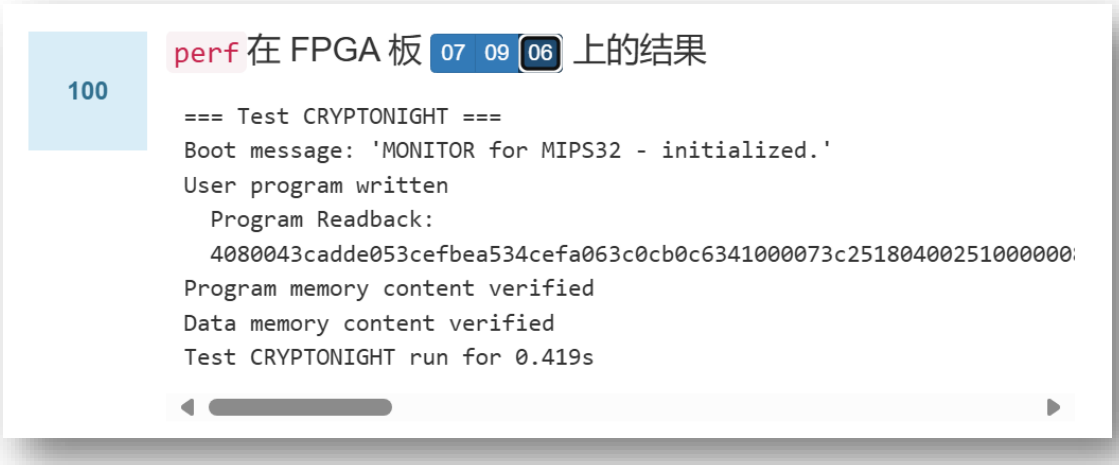
perf 在 FPGA 板 07 09 06 上的结果

```
=== 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 0.114s
```

100

perf 在 FPGA 板 07 09 06 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 0.163s
```



（二）设计交付说明

```
├──constrs_1
|    └──new
|            thinpad_top.xdc
|
├──sim_1
|    ├──imports
|    |
|    ├──new
|    |
|    └──tmp
|
└──sources_1
     ├──ip
     |    └──pll_example    // 时钟分频模块
     |
```

```
├──mycpu
|       ctrl.v      // 流水线暂停信号控制模块
|       defines.v   // 宏定义
|       ex.v        // 执行阶段
|       ex_mem.v    // 执行 -> 访存
|       id.v        // 译码阶段
|       id_ex.v     // 译码 -> 执行
|       if_id.v     // 取指 -> 译码
|       I_Cache.v  // icache
|       mem.v       // 访存阶段
|       mem_wb.v    // 访存 -> 回写
|       mycpu.v     // CPU 顶层模块
|       pc_reg.v    // 取指阶段
|       regfile.v   // 寄存器模块
|       SRAM_CTRL.v // SRAM 控制模块
|       UART_CTRL.v  // UART 控制模块
|
└──new
        async.v     // 串口模块
        SEG7_LUT.v
        thinpad_top.v// 顶层模块
        vga.v
```

四、设计参考说明

1、雷思磊.《自己动手写 CPU》中 MIPS 的流水线设计以及顶层接口定义

2、龙芯杯 NSCSCC2023 个人赛开源代码中有关 UART 串口地址、icache，SRAM 仲裁器的设计。

五、参考文献

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

[2] 汪文祥 刑金璋 . CPU 设计实战. 北京: 机械工业出版社,2021

六、致谢

感谢赖同学对于开发过程中遇到的串口交互问题的建议和解决方案的提出，以及王学长对于项目中 dcache（后续删除）和 icache 模块设计思路和开发的讲解，使我受益匪浅。

同时感谢张老师对于本人小学期免修申请过程中提供的帮助，使我有充足的时间来学习 CPU 和撰写代码。