NSCSCC2023 龙芯杯个人赛 MIPS 赛道设计报告

学校：哈尔滨工业大学（深圳）
姓名：张露晗

一、设计简介

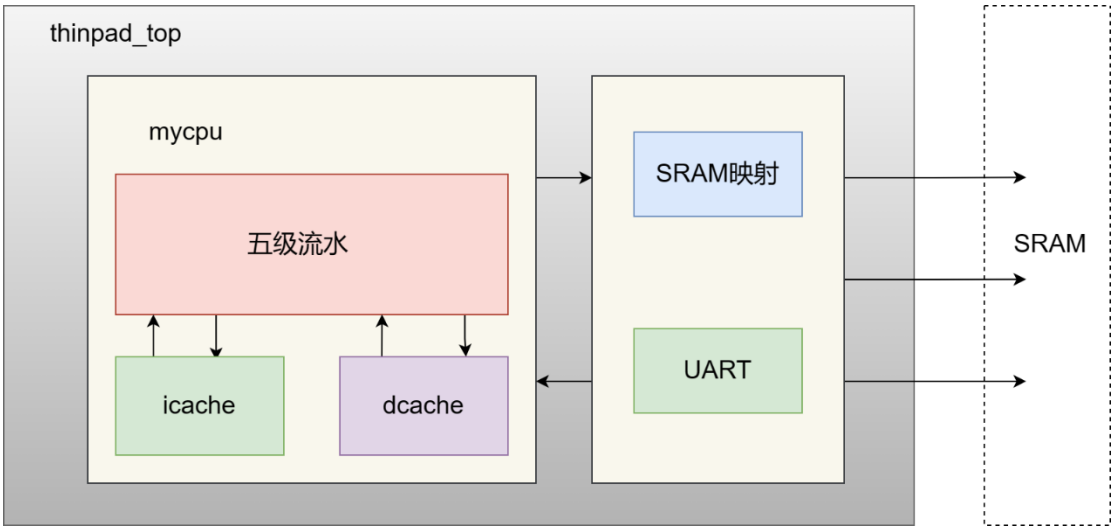
该作品是基于 MIPS32-R2 的一款添加了 cache 的单发射五级流水线 cpu，实现了比赛要求的 MIPS-C1、MIPS-C2、MIPS-C3 指令集中的所有指令，完成了要求的 SUB, SRAV, BGEZ 三条随机指令测试。数据相关问题采用了数据前递的方法解决，Load-Use 冒险采用流水线暂停的方式解决，BaseRAM 结构冒险采用 SRAM 控制器仲裁解决，该 cpu 支持延迟槽，解决了分支跳转的问题。

该作品目前可以通过三级功能测试和性能测试，时钟频率最高可达到 107MHz，性能测试的时间为 0.074s、0.109s、0.260s。目前的 dcache 功能还需要继续完善，性能有望再提升。

二、设计方案

（一）总体设计思路

该作品在经典五级流水线的基础上增加了 icache 和 dcache 模块，使得该处理器能够突破 SRAM 的性能瓶颈。另外采用了 RAM_Serial_ctrl 模块控制串口和 SRAM 的映射。以下主要针对处理器、cache、RAM_Serial_ctrl 控制器 3 个模块进行介绍。



（二）处理器模块设计

MIPS（Microprocessor without Interlocked Pipeline Stages）是一种经典的 RISC（Reduced Instruction Set Computer）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。

1. 取指阶段（Instruction Fetch）：

取指阶段负责从指令存储器中读取指令，并将其送入指令寄存器。取指阶段的主要任务是根据程序计数器（Program Counter，PC）中存储的下一条指令的地址，从指令存储器中读取指令，并将其存储在 IR 中。同时，PC 也需要更新为下一条指令的地址。

2. 译码阶段（Instruction Decode）：

译码阶段负责解析指令，并将其转化为相应的操作。在译码阶段，指令的操作码（Opcode）被提取出来，并根据操作码进行相应的操作，如读取寄存器中的数据、计算跳转地址等。此外，译码阶段还负责将指令的源操作数和目的操作数从寄存器文件中读取出来，并将其传递给执行阶段。

3. 执行阶段（Execution）：

执行阶段是五级流水线中最复杂的阶段，它负责执行指令的实际操作。执行阶段根据指令的操作码和源操作数，进行相应的运算或逻辑操作。例如，对于算术指令，执行阶段负责进行加法、减法、乘法等操作；对于逻辑指令，执行阶段负责进行与、或、非等操作。执行阶段还负责计算内存地址、进行操作数的读取和写入等操作。

4. 访存阶段（Memory Access）：

访存阶段负责处理与内存的交互。在该阶段，执行阶段计算得到的内存地址将被用于读取或写入内存中的数据。对于读数据的指令，访存阶段将从内存中读取数据，并将其传递给下一个阶段；对于写数据的指令，访存阶段将把数据写入内存。

5. 写回阶段（Write Back）：

写回阶段负责将执行阶段计算得到的结果写回到寄存器文件中。在该阶段，执行阶段计算得到的结果将被写入目的操作数所对应的寄存器中。这样，下一条指令就可以使用这个结果作为源操作数来执行相应的操作。

（三）Cache 模块设计

在本项目中，iCache（指令缓存）和 dCache（数据缓存）是两个重要的模块，用于提高指令和数据的访问效率。iCache 用于存储指令，而 dCache 用于存储数据。下面将对本项目

的 iCache 和 dCache 模块的设计进行详细的介绍。

- iCache（指令缓存）的设计：

iCache 是用于存储指令的缓存模块，其设计目标是提高指令的访问效率，减少指令的访问延迟。比赛提供的工程模板 SRAM 数据线宽度只有 32 位，因此访存一次最多也只能读取一个字，基于这个设定，本项目的 icache 设计如下：

a) Cache 块大小：一个字。

b) 缓存映射方式：直接映射，将指令地址直接映射到缓存行。

c) 缓存结构：根据观察，比赛性能测试需要跑的代码长度一般在 32 个字左右，因此设计 icache 大小 32x32=1kb。采用直接映射，每块只一个字，块内寻址 2 位,cache 地址 5 位，而 basesram 大小 4MB，地址 22 位，tag 22-5-2=15 位，另外有一位 valid 位。

- dCache（数据缓存）的设计：

dCache 是用于存储数据的缓存模块，其设计目标是提高数据的访问效率，减少数据的访问延迟。仿照 iCache 设计，dCache 设计如下：

a) Cache 块大小：一个字。

b) 缓存映射方式：直接映射，将数据地址直接映射到缓存行。

c) 写策略：写直达，当 cpu 要求写，则在 cache 和 sram 中同时写，以保持 cache 和 sram 的一致性。

d) 缓存结构：dCache 仿照 iCache 设计，大小 32x32=1kb。采用直接映射，每块只一个字，块内寻址 2 位,cache 地址 5 位，而 basesram 和 extsram 大小 8MB，地址 23 位，tag 23-5-2=16 位，另外有一位 valid 位。

- 后续优化：

为了进一步提高数据的访问效率，dCache 可以采取 writebuffer 设计，目前还未实现。

（四）RAM_Serial_ctrl 控制器模块设计

RAM_Serial_ctrl 模块：整个模块采用组合逻辑设计，主要功能为对来自 icache 和 dcache 的信息进行处理，包括内存地址映射、处理串口收发数据、以及连接 BaseRAM 和 ExtRAM 从而获取数据，对应 RAM_Serial_ctrl.v 文件

三、设计结果

（一）设计交付物说明

```
thinpad_top.srcs
├── constrs_1
│   └── new
│       └── thinpad_top.xdc
├── sim_1
│   ├── imports
│   ├── new
│   └── tmp                    //功能测试文件
│       ├── lab1
│       │   └── lab1.bin
│       └── lab2
│       |   └── lab2.bin
│       └── lab3
│       |   └── kernel.bin
└── sources_1
    ├── ip
    │   └── pll_example        //时钟分频模块
    ├── new
    │   ├── async.v            //串口实例化模块
    │   ├── SEG7_LUT.v
    │   ├── thinpad_top.v      //顶层文件
    │   └── vga.v
    └── mycpu
        ├── dcache.v           //dcache 文件
        ├── defines.v          //宏定义
        ├── ex.v               //ex 阶段
        ├── ex_mem.v           //ex 到 mem 的过渡寄存器
        ├── icache.v           //icache 文件
        ├── id.v               //id 阶段
        ├── id_ex.v            //id 到 ex 的过渡寄存器
        ├── if_id.v            //if 到 id 的过渡寄存器
        ├── mem.v                  //mem 阶段
```

```
├── mem_wb.v                //mem 到 wb 的过渡寄存器
├── pc_reg.v                //pc
├── SRAM_UART_ctrl.v        //内存映射及串口控制模块
├── regfile.v               //寄存器堆
├── ctrl.v          //流水线暂停控制
└── mycpu.v                 //mycpu 顶层模块
```

（二）设计演示结果

功能测试：

	得分
一级评测	100
二级测评	100
三级测评	100
性能测试	100

性能测试：

	运行时间
STREAM	0.074 s
MATRIX	0.109 s
CRYPTONIGHT	0.260 s

四、参考设计说明

1.雷思磊.《自己动手写 CPU》中 OpenMIPS 的接口定义及流水线框架的设计

2.龙芯杯 NSCSCC2022 个人赛开源代码 XZMIPS 中关于 sram 控制器的接口设计

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] 姚永斌 . 超标量处理器设计. 北京：清华大学出版社,2014.