COD23-GRP31 设计报告

清华大学计算机系计 15 班
朱煜章 郭高旭 沈政诺

2023.12.24

文章导航

1 设计简介

1.1 项目背景

本项目是 2023 年计算机组成原理课程的课程大实验作品。项目实现了基于 RiscV32 指令集的 7 级流水线 CPU，并以课程提供的 FPGA 实验平台为基础，实现了完整的 CPU 微架构。

1.2 项目概述

本项目使用 SystemVerilog 编写。CPU 的各个流水段直接实现在内部，组成部件设计为模块，降低了各个结构的耦合度。

我们采用 7 级流水的基本架构，流水线结构大致分为指令地址映射、取指和缓存、指令译码、执行、数据地址映射、读写内存和异常处理、写回。

我们实现了要求实现的所有 RV32I 指令、CSR 指令、页表指令以及要求添加的 4 条指令，通过了 ucore 系统测试。

2 设计方案

2.1 总体设计思路

本 CPU 采用 7 级流水线的基本架构，整个流水线数据通路大致结构如下：

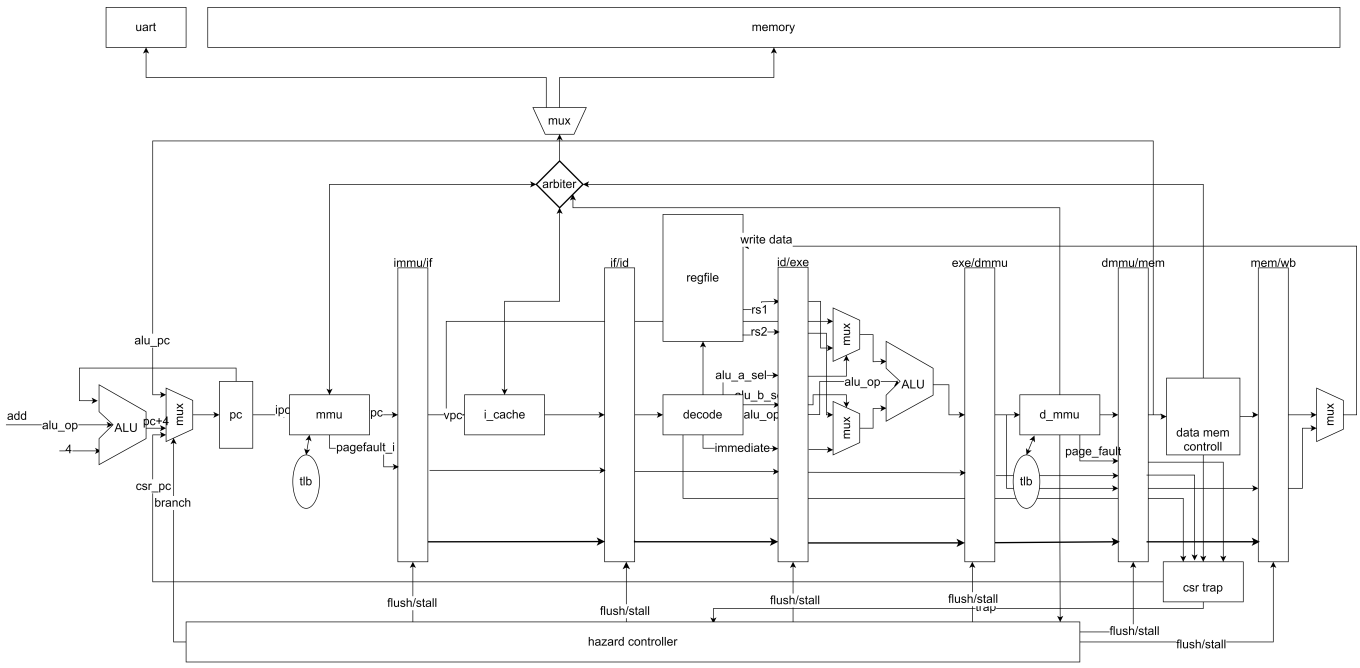

图 1: 数据通路

涉及的模块有：地址翻译、取指和指令缓存、指令解码、寄存器堆、计算和分支、读写内存、异常处理等。下面对本组所有扩展功能的实现做一说明讲解。

2.2 地址翻译模块设计

我们实现了 Riscv32 架构中规定的 TLB-Based MMU。

MMU 包含 TLB 和两级页表（LV1 和 LV2）。TLB 用于高效缓存最近的虚拟地址到物理地址的映射。为简化实现，TLB 仅有一项，由于程序的连续性，在实际测试中仍有较高命中率。

输入的虚拟地址 VA_I 包括 32 位，分为偏移 (VA_OFFSET)、LV2 VPN(VA_VPN_2)、LV1 VPN(VA_VPN_1)。输出的物理地址 PA_O 包括 32 位，由 LV2 页表的 PTE 中的 PPN 和虚拟地址的偏移组成。

通过比较 TLB 的有效位、标签和 VPN，设置 TLB_HIT 信号，判断是否发生 TLB 命中。若不命中，则根据状态机状态，在 MMU_LV1 和 MMU_LV2 状态下，向存储器发出读请求，获取对应的 PTE。在 MMU_LV2 状态下，如果访问成功，更新 TLB 的标签和 PTE。如果发生页错误，则拉高信号 PAGE_FAULT_O，逐级传递给异常处理模块。

2.3 取指和指令缓存模块设计

这里主要介绍我们指令缓存的实现。指令缓存模块属于取指流水段，其中包含指令缓存块和指令重取状态机。

Cache 使用 64 个缓存条目（cache entry），每个缓存条目有 57 位，其中包括一个有效位、24 位标签、和一个 32 位数据。24 位标签是由于我们只实现了 icache，总是 4 地址对齐，因此可以忽略低二位。CACHE_REGS 是一个包含 64 个缓存条目的寄存器数组。Cache 采用直接映射方式，使用 6 位索引（cache index）和 24 位标签（cache tag）。

在 CACHE_IDLE 状态，如果发生缓存命中，则立即提供数据，否则向存储器发出读请求。

在 CACHE_MEM_READ 状态，等待存储器的响应，将数据写入缓存。

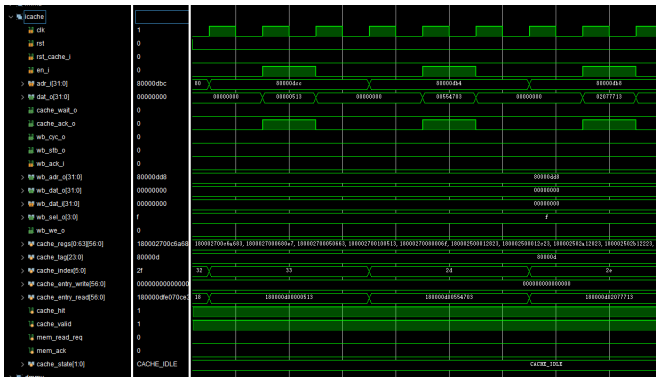

图 2: 缓存波形

2.4 CSR 和中断异常模块设计

我们将中断异常处理和 CSR 寄存器在同一个模块中实现。CSR 寄存器本身的结构相当简单，只需要注意不同 CSR 寄存器位的对应关系和权限问题即可。

中断异常的处理逻辑如下：当检测到缺页/地址不对齐/非法指令/ecall 等系统调用指令的时候，将对应的信号传递到读写内存对应的 MEM 流水段，由该模块统一处理：由 MEDELEG 指示异常对应的特权态，然后

将对应的 pc 写入 MEPC 或 SEPC、错误原因写入 MCAUSE 或 SCAUSE、错误信息写入 MTVAL 或 STVAL、更改 MSTATUS 等寄存器的标志位。若为中断，则还要关闭对应权限的中断使能位。然后将 pc 跳转到对应的处理程序位置。从高等级权限态返回时，也需要更改 MSTATUS 的对应位。

本组只实现了时钟中断，没有实现其他中断。

异常和中断的时候均不能读写内存或者写寄存器，因此要直接将指令替换为 nop 以防止不当的操作。中断相比异常，这点尤为重要：异常的指令一般都是无副作用的；而中断的指令原先是有意义的。所以在中断的时候要特别将这条指令的所有作用都取消掉（读写内存、写寄存器）。

2.5 性能对比

我们小组没有太关注性能优化，最终只通过了 CRYPTONIGHT

测试	初始版本（15Mhz）	icache（15Mhz）	icache（50Mhz）
CRYPTONIGHT	40.029s	15.065s	4.519s

表 1: 性能对比表

3 思考题

1. 流水线 CPU 设计与多周期 CPU 设计的异同

流水线 CPU 将指令执行过程划分为多个阶段，并允许同时执行多个指令的不同阶段，以提高整体执行效率。多周期 CPU 则将指令执行划分为多个较大的时钟周期，每个周期内完成特定阶段的操作。总体而言，流水线 CPU 具有更高的吞吐量和效率，但可能会面临数据冲突和流水线停顿等问题。而多周期 CPU 相对简单，更容易实现，但在同一时刻只能执行一条指令，效率相对较低。

插入等待周期（气泡）和数据旁路在处理数据冲突的性能上有什么差异。

插入等待周期（气泡）是为了解决数据冲突而在流水线中插入空操作周期，以防止数据错误。这种方法会导致性能下降，因为处理器在等待周期内无法执行实际操作。相比之下，数据旁路允许处理器从其他可用的数据来源获取数据，而不是等待数据准备就绪。这样可以避免插入等待周期，提高性能。因此，数据旁路在处理数据冲突时通常比插入等待周期更有效，因为它允许处理器更灵活地继续执行指令，而不是被迫等待。

2. 如何使用 Flash 作为外存，如果要求 CPU 在启动时，能够将存放在 Flash 上固定位置的监控程序读入内存，CPU 应当做什么样的改动？

1. **引导程序修改：**需要修改系统引导程序，以支持从 Flash 中加载监控程序。引导程序负责在系统启动时初始化硬件，并将监控程序加载到内存中的指定位置。

2. **Flash 接口初始化：** CPU 需要初始化 Flash 存储器的接口，确保能够正确地读取数据。这可能涉及到配置 Flash 控制器、设置时序参数等操作，以确保稳定和可靠的数据传输。

3. **地址映射：**确保监控程序的固定位置在内存地址空间中有正确的映射。这意味着需要在内存地址映射表或者其他相应的机制中指定 Flash 存储器的地址范围，以便 CPU 能够正确访问其中的数据。

3. 如何将 DVI 作为系统的输出设备，从而在屏幕上显示文字？

将部分内存地址映射给 DVI，通过总线访问。

4. 考虑支持虚拟内存的监控程序。如果要初始化完成后用 G 命令运行起始物理地址 0x80100000 处的用户程序，可以输入哪些地址？分别描述一下输入这些地址时的地址翻译流程。

可以输入 0x0 和 0x80100000。关键在于多个一级页表项可以映射到同一个二级页表项。

页表工作流程如下：

首先分别对两个虚拟地址进行解析，解析为偏移（`va_offset`）、LV2 VPN（`va_vpn_2`）、LV1 VPN（`va_vpn_1`）。输出的 32 位物理地址 `pa_o`，由 LV2 页表的 PTE 中的 PPN 和虚拟地址的偏移组成。通过 `tlb_hit` 信号判断是否发生 TLB 命中，比较 TLB 的有效位、标签和 VPN。根据状态机状态，在 `mmu_lv1` 和 `mmu_lv2` 状态下，向存储器发出读请求，获取对应的 PTE。在状态机中，如果发生页错误，设置 `page_fault_o` 为 1，通知上层。

5. 假设第 a 个周期在 ID 阶段发生了 Illegal Instruction 异常，你的 CPU 会在周期 b 从中断处理函数的入口开始取指令执行，在你的设计中，b - a 的值为？

本组设计中对 Illegal Instruction 的处理在 MEM 段，ID 段与 MEM 段之间有 EXE、DMMU 两个流水段，因此 b-a 的值为 3。

4 心得体会

4.1 关于中断异常的经验与教训

时钟中断是需要特别留意的。我后期出现的问题基本都在于时钟中断。由于时钟中断实现有误，导致我在对队友的 cache 和 TLB 进行移除之后，现象不一样。按道理说，缓存和 TLB 之类部件的移除，不会导致除了速度以外的差别，于是我觉得他们这里实现的是错的~~于是就开始骂人子~~。但是速度的改变就会导致中断位置的改变，从而导致（由于时钟中断的实现有误）时钟中断的触发位置改变，从而导致了不一样的现象。这个问题我花了很长时间才发现，也是我造机过程中最大的失误。

——朱煜章

4.2 造机笑话

造机不如买机，买机不如租机。

以下是本组发生的一些造机笑话。全部保真，谨供大家取乐。

1. 调 ucore 时，某天晚上使用 vivado 进行仿真。仿真两个多小时，没能仿出问题来，C 盘跑满了。
2. 气不过，清理了一下 C 盘后，决定再试一次。第二次仿真三个小时，C 盘没跑满，机器蓝屏了，丢失了另一门课没保存的作业。

3. 然后决定使用 verilator 进行仿真。在本地 wsl 配它的环境。然后一顿误操作，将 wsl 内的内容全部删除了，释放了 40 G 容量，解放了跑满的 C 盘。
4. 炸掉 wsl 之后重新装，倒是配好环境了，但是发现 verilator 对内存的需求很大，我们的电脑跑不出来什么东西，内存就满了。
5. 最后使用钞能力，租了服务器跑 verilator 才跑出来，氪金 400 元。遂立下志向，等 35 岁被大厂开了之后，就去当网吧老板给大家租机器。

——朱煜章

4.3 和你合作真的太愉快了

就像蝴蝶飞不过沧海，又有谁忍心责怪？

在我们整整四天卡在同一个问题毫无进展的时候，我真的想过如果失败会怎样？但我还是选择 100% 相信我的队友。

感恩我的两个队友，我造机的大部分时间用在翘着二郎腿陪他们干活。我们每次的组队都这样 drama，整个造机过程中我那汹涌的情绪现在嘴边倒不出来。以后的课程大概没有组队机会了，这太不幸了。

——郭高旭

4.4 精彩瞬间

It's impossible, so more worthy to be believed.

——*Faust 6420*

在“奋战三星期，造台计算机”的过程中收获了难能可贵的造机体验，记忆最深刻的是宿舍中厅的下午，檐上的鸟雀落在窗台，以及枯坐在同一个屏幕前 debug 时灵感迸发的那些瞬间。非常感谢我的两个队友和计算机组成原理课程组的老师和助教，记忆不会被磨灭。

——沈政诺

4.5 debug

少写少错，不写不错

debug 的时候窗台上的珠颈斑鸠、喜鹊、乌鸦、麻雀都挨了我们不少骂。对它们表示诚恳的歉意。

还是觉着我们有点被造机 pua 了，只是个简单的占比也不算高的作业而已，为什么激发了我们这么多情绪？善哉，善哉。

我负责的 tlb，cache 部分没出什么大错，这教育我：如果一段代码正确运行了一百万次，在出现问题时不应该首先怀疑它；挑点简单的写，比较不容易出 bug。

——郭高旭

效果展示

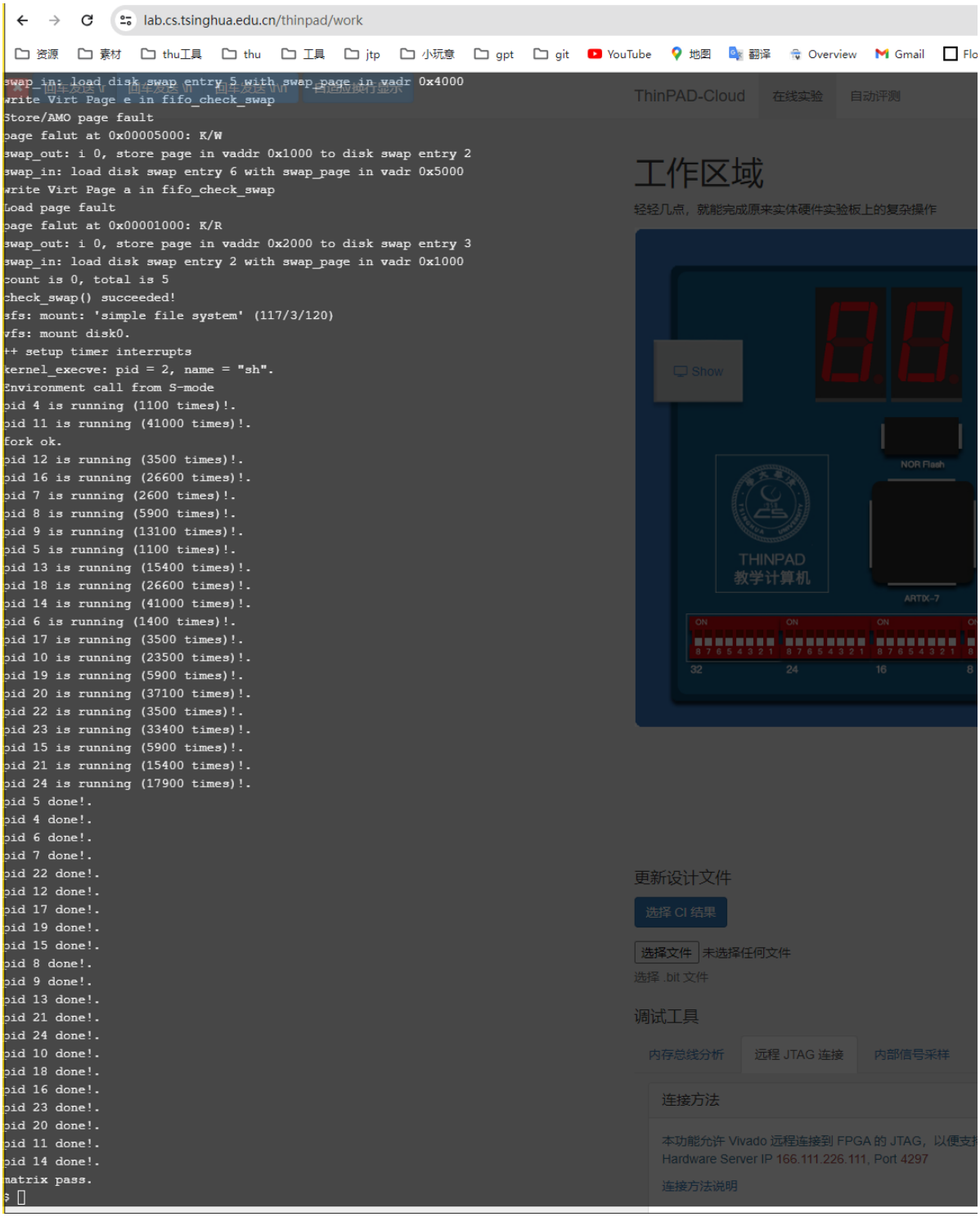

图 3: uCore matrix 指令运行结果

设计交付物说明

这里是本组的代码仓库。其中还包括了我们的 verilator 验证脚本。仓库地址

小组分工

- 朱煜章：流水线基本结构、中断异常机制以及 M、S、U 态切换，主要的调试工作
- 郭高旭：页表及虚拟地址转换、TLB、指令缓存、verilator 配置
- 沈政诺：RV32I 基础指令集和额外指令添加、流水段拆分