LA32R 乱序双发射处理器设计报告

中国科学技术大学 神威·巢湖之光队
杨家和、吕思源、刘然熙、程思翔

一、设计简介

本次大赛我们实现了一个基于龙芯架构 32 位精简版指令集的 CPU，使用 Chisel 开发，属于乱序双发射微架构，参考《超标量处理器设计》一书[1]，采用统一物理寄存器堆的架构，并实现了指令/数据 Cache、分支预测等模块。

本 CPU 可以正确运行功能测试的 58 条测试，支持 AXI 接口，且可以达到 77MHz 的主频，但目前只能正确运行一条性能测试，性能测试分数为 0.134。

二、设计方案

（一）总体设计思路

本 CPU 有 11 级流水线结构，可分为：

- 前端：预取指(PF)、取指(IF)、预译码(PD)、译码(ID)、重命名(RN)
- 后端：发射(IS)、读寄存器堆(RF)、执行(EX)、写回(WB)、提交(CMT)

每个流水级的功能如下：

- **预取指**：包含 PC 和分支预测器，提供两条指令的 PC，并将地址送入指令 Cache 中。
- **取指**：从指令 Cache 中读取两条指令。
- **预译码**：初步处理分支指令。对于 BL、B 和非跳转指令，可以直接知道跳转结果，若与分支预测结果不同，则冲刷前两个流水级并重新取指。对于分支预测器没有记录的条件跳转指令，根据跳转方向给出预测（向前则跳转）。然后，指令被送入 FIFO。
- **译码**：生成后端需要的所有控制信号。同时，访问“空闲寄存器列表”，为目的寄存器（rd）分配物理寄存器。
- **重命名**：访问 cRAT（内容寻址的重命名映射表），得出源寄存器对应的物理寄存器，写入新的映射关系，并处理 RAW 和 WAW 相关。然后，将指令存入重排序缓存（ROB），根据译码结果将指令送入对应的发射队列。
- **发射**：采用分布式、非数据捕捉、压缩式的发射队列，有 1 个特权与乘除法发射队列、2 个 ALU 发射队列（其中一个支持处理分支指令）和 1 个访存发射队列。

- **读寄存器堆**：包含有 8 个读端口与 4 个写端口的寄存器堆，支持写优先。同时，特权指令读取 CSR 寄存器。
- **执行**：ALU 与分支指令只需 1 个周期，乘除需 3 个周期，访存需 2 个周期。
- **写回**：将结果写回到寄存器堆与 ROB 中，并通过旁路网络前递到执行阶段。
- **提交**：ROB 将最旧的已完成的指令提交，更新 aRAT，用以在分支预测失败时恢复处理器状态。

（二）分支预测模块设计

分支预测模块主要采用了 GHR、PHT 预测跳转方向，BTB、RAS 预测跳转地址。PHT 大小参数化，索引方式采用了 G-Share 的变种，即索引地址高位为 GHR 与同等位宽的 PC 中间部分异或得到，索引地址高位等同于 PC 低位。具体的 GHR 位宽、索引位宽均在 Chisel 源码中实现了参数化，可调节。PHT 以及 BTB 均为 Block Memory 实现，因此预测需要一个周期完成。BTB 为直接映射形式，仅一路，大小参数化，根据指令的第 2 位分成两个存储器，以便同时进行两条指令的预测。RAS 存储返回跳转地址。

GHR 会推测更新，以便及时的利用全局历史。在提交模块中根据指令提交信息维护了绝对正确的 GHR 和 RAS，用于在预测错误时恢复 GHR 和 RAS。BTB、PHT 根据提交模块传递的正确跳转地址、方向进行更新。

（三）Cache 模块设计

Cache 模块主要分为 Icache 与 Dcache，采用一级 Cache 设计，便于维护存储一致性，并且对于 FPGA 设备一级 Cache 效果已经较好，时序也相对较好。

Icache 采用两路组相联结构，每个缓存行分为 valid, tag, data 三部分。采用 LRU 替换策略。一次读两条指令。状态机分为 idle, miss, refill, wait 四部分。当未 miss 时一直处于 idle 状态，顺序的取出相连的两条指令，miss 时通过总线读出指令并完成 refill。

Dcache 也采用两路组相联结构。相较 Icache 多出一个 dirty 位。采用写回设计，提高 cache 效率。除去与 Icache 相同的状态，Dcache 加入状态 wait，配合 wrt 状态机完成在写回时的状态。

三、设计结果

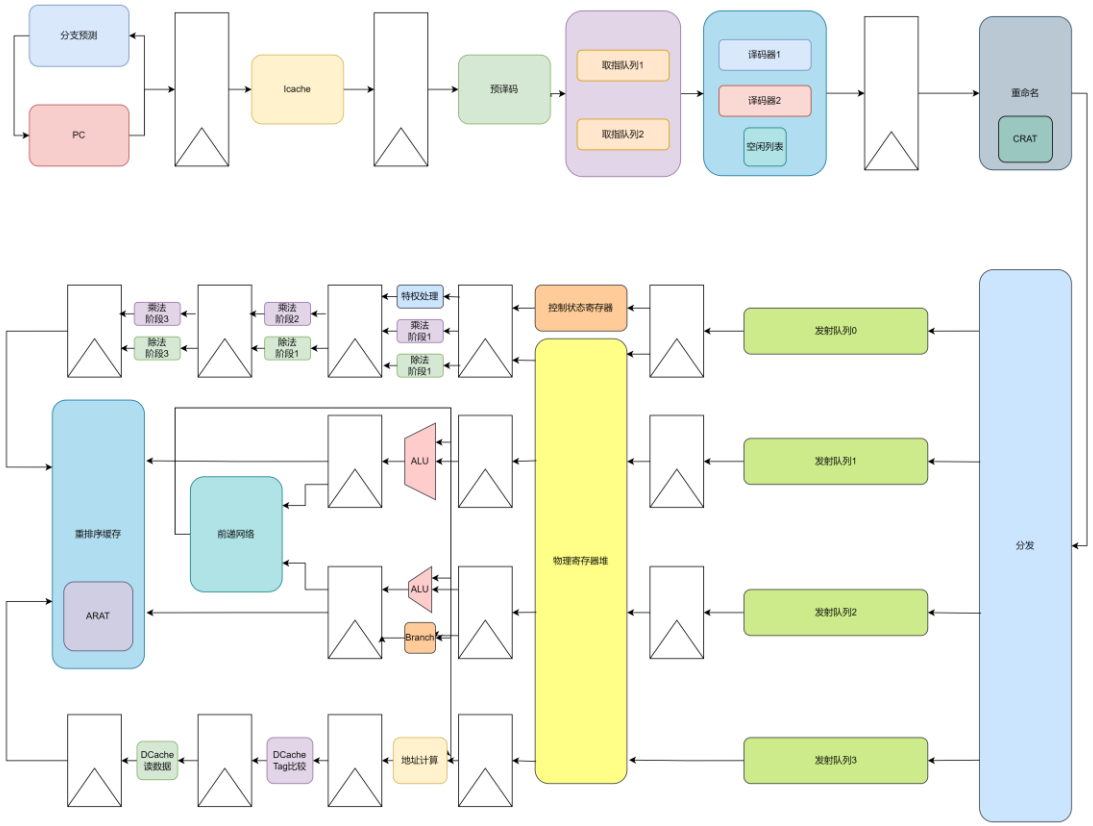
（一） 设计交付物说明

文件夹 submission/ustc_3_yangjiahe 的结构

```
|-score.xlsx
|-design.pdf
|--src/
|       |--mycpu/
|       |     |--xilinx_ip/
|       |--vivado_cannot/
|       |--perf_clk_pll.xci
|--chisel/                          -------项目设计的 chisel 源代码
```

（二）　设计演示结果

下图是本团队设计的 CPU 的数据通路



四、参考设计说明

本项目的架构和一写代码结构参考了一些成熟的设计。这些参考的代码所在仓库列在下

方。这里我们感谢这些仓库给本项目带来的帮助。

1. 第七届“龙芯杯”特等奖 NOP-Core

   https://github.com/NOP-Processor/NOP-Core.git

2. LA32R-pipeline-Scalahttps://github.com/MaZirui2001/LA32R-pipeline-scala.git
3. openla500 https://gitee.com/loongson-edu/nscscc-openla500
4. 卅 https://gitee.com/MJ_Wang/spinal-loong-arch-core/tree/master
5. ChiselMIPS https://github.com/NSCSCC-2022-TJU/ChiselMIPS.git
6. NENE Core https://github.com/AlwenXXD/nscscc2021
7. YuQuan Project https://github.com/Tang-Haojin/YuQuan.git
8. ucas-nscscc-2022 https://gitee.com/ucas-nscscc-2022/mycpu.git

五、参考文献

[1] 姚永斌. 超标量处理器设计[M]. 北京：清华大学出版社, 2014.
[2] 汪文祥 邢金璋.CPU 设计实战[M]. 北京：机械工程出版社，2021
[3] 李东声 任子木 孙小明 李鹏.高性能超标量 CPU[M]. 北京：机械工程出版社，2023