第五届“龙芯杯”全国大学生计算机系统能力培养大赛

GenshinCPU
初赛设计报告

西北工业大学 2 队

江嘉熙
jshmjjx@mail.nwpu.edu.cn

杨益滔
nwpuyyt@mail.nwpu.edu.cn

魏天昊
weitianhao@mail.nwpu.edu.cn

申世东
seddon@mail.nwpu.edu.cn

2021 年 8 月

目录

第一部分　概述

1.1　项目简介

GenshinCPU 是一个实现在龙芯教学实验平台（Artix-7 XC7A200T）实现的基于 MIPS 32 Rev 1 指令集架构的处理器。其包含指令和数据缓存，能够正常运行功能测试、性能测试、记忆游戏和系统测试。其频率达到 155MHz，初赛性能分约 79 分。

1.1.1　设计参数

GenshinCPU 采用单发射七级流水线架构，CPU 对外通过 4 个接口进行通信，分别是 Uncache 的指令和数据接口、Cache 的指令和数据接口。接口采用 AXI3 接口，CPU 使用 2 路组相联 8KB ICache 和 2 路组相联 8KB DCache。CPU 采用使用 16 项全相联的 TLB 进行地址转换，并使用历史分支查找表（BHT）和返回地址栈 (RAS) 进行分支预测，分支预测整体准确率为 89%。

第二部分　CPU

2.1　流水线结构

2.1.1　总体架构

GenshinCPU 采用七级流水顺序单发射结构，分别为预取指（PREIF）、取指（IF）、译码 (ID)、执行 (EXE)、访存一 (MEM)、访存二 (MEM2)、写回 (WB) 七级。

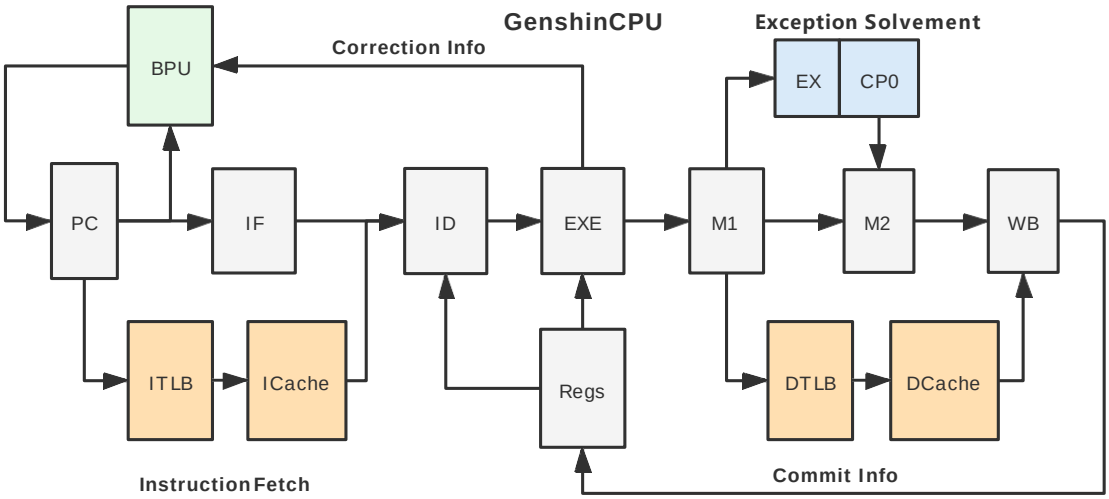

图 2.1: 流水线结构示意图

2.1.2　预取指（PREIF）级

预取指阶段将 PC 中存储的指令地址，根据 ICache 的需求拆分为 Tag,Index,Offset 段，由于采用 VIPT 的访问方式，直接将 Index 信号连入 Data Ram 和使用 Lutram 存储的 Tag Ram ，Tag Ram 当拍读出数据，Data Ram 下一拍读出数据。同时将 Tag 信号传入 MMU 进行虚实地址转换。将 MMU 输出的实地址和从 Tag Ram 读出的实地址进行比较，计算命中。同时锁存所有的访存信息。

2.1.3　取指级（IF）级

从 Data Ram 中读出相应的数据，计算所需数据。同时根据计算命中的结果，给出控制信号。如果缺失，则向 AXI 总线发起访存请求，重填之后提供相应的数据。如果锁存下来的访存信息表示该次访存是 Uncache 的，则启动 Uncache 状态机对外发起请求。

2.1.4 译码（ID）级

ID 级主要对指令进行译码，以及读取寄存器堆。在这一级会对数据冒险进行检测，采用前递和阻塞来解决。这样保证了送往执行级的数据都是正确的数据。与此同时，我们复用了 19 年清华大学参赛队伍定义的指令枚举变量，使得代码更具可读性，降低了代码的复杂度，借鉴了译码部分，减少了工作量。

2.1.5 执行（EXE）级

EXE 级包含的模块主要有 ALU，以及乘除法器。乘除法均使用 Vivado IP 核，除法约 37 周期完成，乘法 3 周期完成。乘除法计算完毕后立即写回 HILO 寄存器。在这一级还会对分支预测进行校准，根据指令的实际跳转结果来更新 PC。

2.1.6 访存（MEM、MEM2）级

在我们的设计中访存单元总共分为两个阶段, 在 MEM1 级发起对 DTLB 的访问请求, 以及对 Dcache 的 Tag Ram(LutRam),DataRam(xpm) 发起请求. 在 MEM1 级的末尾判断是否命中, 在 MEM2 级获取 Dcache 的数据。对于 TLB 模块，如果 DTLB 未命中, 则阻塞流水线, 下一周期访问 TLB 模块。如果命中则更新 DTLB, 如果仍然未命中, 则报出异常。对于 DCache 模块, 发出访问请求之后, 在 MEM1 级的末尾阶段会判断该次访问是否命中, 如果命中则在 MEM2 级可以获得访问的数据。如果未命中, 则阻塞流水线, 向 AXI 总线发起访存请求。

2.1.7 写回阶段

写回阶段将得到写回结果，并将数据发送给寄存器。

2.2 指令集

CPU 在大赛要求的 57 条指令基础之上，增加了部分指令以启动操作系统。目前共实现了如下 89 条指令：

- **分支指令** BLTZ, BGEZ, BLTZAL, BGEZAL, BEQ, BNE, BLEZ, BGTZ, JR, JALR,J, JAL
- **逻辑指令** AND, OR, XOR, ANDI, ORI, XORI, NOR, SLL, SRL, SRA, SLLV, SRLV,SRAV
- **算术指令** ADD, ADDU, SUB, SUBU, ADDI, ADDIU, MUL, MULT, MULTU,DIV,DIVU, MADD, MADDU, MSUB, MSUBU, CLO, CLZ
- **访存指令** SB, SH, SW, LB, LH, LW, LBU, LHU, LWL, LWR, SWL, SWR
- **特权指令** SYSCALL, BREAK, TLBP, ERET, MTC0, MFC0, TLBWI, TLBWR, TLBR
- **条件移动指令** SLT, SLTU, SLTI, SLTIU
- **无条件移动指令** LUI, MFHI, MFLO, MTHI, MTLO

- **自陷指令** TEQ, TGE, TGEU, TLT, TLTU, TNTEQ, TGE, TGEU, TLT, TLTU, TNE, TEQI, TGEI, TGEIU, TLTI, TLTIU, TNEI

2.3 协处理器 CP0

CPU 实现了 MIPS 32 Rev 1 规范中协处理器 0 中的大部分寄存器，同时为了启动操作系统，实现了 MIPS 32 Rev 2 规范中的 EBase 寄存器。所有寄存器如下，名称摘录自参考资料 3:

- BadVAddr Register (CP0 Register 8, Select 0)
- Count Register (CP0 Register 9, Select 0)
- Status Register (CP Register 12, Select 0)
- Cause Register (CP0 Register 13, Select 0)
- Exception Program Counter (CP0 Register 14, Select 0)
- Index Register (CP0 Register 0, Select 0)
- Random Register (CP0 Register 1, Select 0)
- EntryHi Register (CP0 Register 10, Select 0)
- EntryLo0, EntryLo1 (CP0 Registers 2 and 3, Select 0)
- Context Register (CP0 Register 4, Select 0)
- PageMask Register (CP0 Register 5, Select 0)
- Wired Register (CP0 Register 6, Select 0)
- Compare Register (CP0 Register 11, Select 0)
- Processor Identication (CP0 Register 15, Select 0)
- EBase Register (CP0 Register 15, Select 1)
- Conguration Register (CP0 Register 16, Select 0)
- Conguration Register 1 (CP0 Register 16, Select 1)

2.4 内存管理

CPU 使用内存管理单元（MMU）以及相应的地址映射关系来进行虚地址到实地址的转换。针对 TLB 的设计，我们参考了 19 年中国科学院大学（UCAS）参赛队伍的 TLB Buffer 的设计。我们设计了独立的 ITLB Buffer 和 DTLB Buffer，它们与 TLB 进行交互。若 Buffer 发生缺失，将阻塞流水线一拍，并对 TLB 进行查找，将查找结果填回 Buffer。这使得我们的 CPU 在使用 16 项全相联的 TLB 的同时，仍然能够具有较高的频率。

2.5 中断和异常

我们的设计中, 在 MEM1 级, 我们对 CP0 完成读写操作, 并实现了对于精确异常的处理. 具体如下:

2.5.1 中断

CPU 支持两个软件中断 (SW0 SW1) ,6 个硬件中断 (HW0 HW5), 一个计时器中断, 计时器中断复用 HW5 硬件中断。

2.5.2 异常

CPU 实现了 MIPS 规范的精确异常，在访存级统一提交异常。CPU 中实现的异常如下:

- TLB Refill Exception
- TLB Invalid Exception
- TLB Modified Exception
- Overflow Exception
- System Call Exception
- Breakpoint Exception
- Reserved Instruction Exception
- Coprocessor Ununable Exception
- Trap
- Address Error Exception

2.6 缓存设计

CPU 实现了 ICache 和 DCache，分别响应 CPU 的取指和访存请求。在 Cache 设计上，我们实现了 2 路组相联 8KB ICache 和 2 路组相联 8KB DCache。存储的 RAM 使用 Xilinx XPM 简化了 ip 核的部署过程。使用 LutRam 能够当拍读出 Tag 计算命中。与此同时，我们借鉴了 19 年清华大学参赛队伍（编程是一件很危险的事情队）参赛作品中的"mux_byteenable" 函数，减少了 sb,sh 指令的压力。在替换策略的实现上，我们也参考了该队伍参赛作品中的可配置 PLRU 功能部件。

2.7 分支预测

在分支预测的设计上，我们参考了 20 年哈尔滨工业大学（深圳）的分支预测方式。采用 BHT(Branch History Table)+RAS(Return Address Stack) 的结构, 其中 BHT 为 256 项的历史分支查找表，RAS 为返回地址堆栈。我们将分支预测拆成查找和判断两级，使 CPU 能够保持较高的频率。分支预测的整体准确率为 89%。

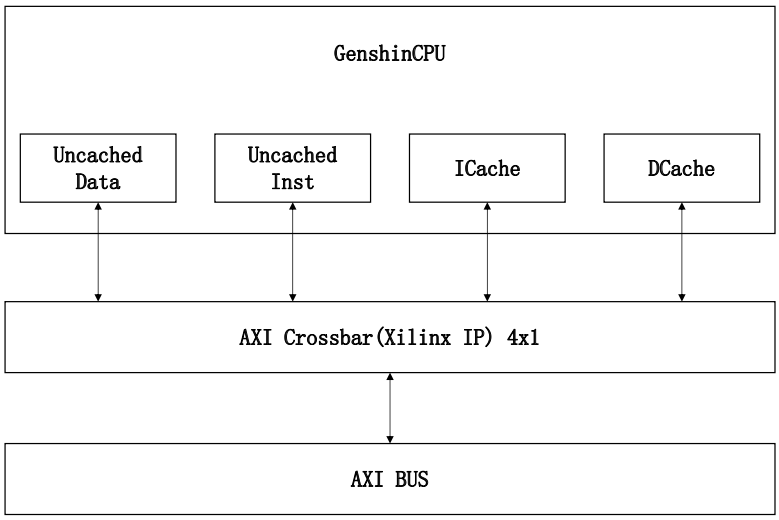

图 2.2: CPU 与外部通信

2.8 外部接口

CPU 和外部交互如图2.2 所示，为了整齐的管理 Cache 以及 Cpu 对外接口的访存，我们在 AXI 模块中，通过一组接口与 Cache 模块进行交互。这实际上就是将 Cache 的部分功能分离出来，缩小每一模块的职责，使得调试与编写时更为容易。CPU 对外有四个 AXI 接口，分别为经过缓存的 ICache、DCache 与不经过缓存的 ICache、DCache。为了满足大赛要求，我们使用了 Xilinx 提供的 AXI Crossbar IP，其包含 4 个 Slave 接口和一个 Master 接口，并将四个 AXI 接口整合为一个引出到外部 SOC 上。四个接口的优先级为 Uncached-Data > Uncached-Inst > DCache > ICache。

第三部分　附录

3.1　参考资料

MIPS® Architecture For Programmers Volume I-A: Introduction to the MIPS32 Architecture.rev3.02

MIPS® Architecture For Programmers Volume II-A: The MIPS32® Instruction Set

MIPS® Architecture For Programmers Volume III: The MIPS32® and microMIPS32™ Privileged Resource Architecture

Xilinx IP:: AXI Crossbar (2.1) LogiCORE IP Product Guide

3.2　参考仓库

3.2.1　19 年参赛作品

清华大学编程是一件很危险的事情队 https://github.com/trivialmips/nontrivial-mips

中国科学院大学 2 队 https://github.com/nscscc2019ucas/nscscc2019ucas