Haribote MIPS CPU 设计报告

武汉大学 1 队
郑晖、原昊博、范文骞、陈子轩

一、设计简介

本实验实现了一个 Haribote MIPS CPU。该 MIPS CPU 可以执行 MIPS 指令之中 89 条指令，包括{j, jal, beq, bne, blez, bgtz, addi, addiu, slti, sltiu, andi, ori, xori, lui, lb, lh, lwl, lw, lbu, lhu, lwr, sb, sh, swl, sw, swr, ll, pref, sc, mfc0, mtc0, bltz, bgez, tgei, tgeiu, tlti, tltiu, teqi, tnei, bltzal, bgezal, sll, srl, sra, sllv, srlv, srav, jr, jalr, movz, movn, syscall, break, sync, mfhi, mthi, mflo, mtlo, mult, multu, div, divu, add, addu, sub, subu, and, or, xor, nor, slt, sltu, tge, tgeu, tlt, tltu, teq, tne, tlbr, tlbwi, tlbp, eret, madd, maddu, mul, msub, msubu, clz, clo}。根据初赛要求，封装成了 AXI 接口，用 bram 实现了 2 路 LRU 共 16KB 的 ICache 和 2 路 LRU 共 8KB 的 DCache，并按照初赛要求写出了可容纳 32 个页表项的 TLB，同时根据 AXI3 协议，为 ICache、DCache 和 uncached 段 load/store 写出了状态机尽量利用 axi 的 burst 传输，提高访问 ram、外设的效率，并保证了准确性。

（一）设计变更说明

计划在决赛之前进一步增加 MIPS 指令，跑起来 Linux 内核，并向 SoC 中添加之前写的一些外设接口，构建一个比较完整的 SoC，然后利用实验板的 lcd 触摸屏做一个切水果的游戏。如果完成了这些，计划继续利用板上的自由引脚外接摄像头，用摄像头接收数据，用硬件实现一个数字识别的模块，将结果和摄像头接收数据显示在 lcd 屏幕上。

二、设计方案

（一）总体设计思路

整个系统分为三个文件夹和一个 mycpu_top.v 顶层文件。其中，三个文件夹分别是 Cache、Bus、CPU，而 mycpu_top.v 文件则是对 CPU 文件夹下 Mips.v 的例化。所以，整个 SoC 只包含了 CPU 核心，没有自行添加外设之类的，而 Bus 也只是 CPU 内部用于 AXI 请求分配的总线，Cache 也只有一级，使用 bram 做 data，属于片内 cache。

1、Cache

该文件夹下包含了两个重要的文件 ICache.v 和 DCache.v 文件，分别是 Instruction Cache

和Data Cache的顶层文件，其中例化了CachelineBlock，该模块使用reg记录着一个CacheLine的tag、valid、dirty属性，供ICache、DCache中的状态机使用。然后就是例化了block_cacheblock_data_8KB(或者block_cacheblock_data)用于存储整个cache的data段。

2、Bus

该文件夹下分为AXI_Load_Bus和AXI_Store_Bus，分布负责AXI read/write channel信号的分配。每个Bus包括三个文件，例如AXI_Load_Bus包含AXI_Cache_Load_Bus.v这是该部分的顶层文件，其内容是例化了bus_arbiter.v和bus_master_mux.v。bus_arbiter主要是通过不同master发来的req请求，进行优先级判断，然后向选中的req请求方送回grnt信号表明给予其bus使用权，同时产生busy信号通知其他master。在Mips中例化主要用于接管ICache、DCache和uncachedLoader的AXI请求管理。

3、CPU

该文件夹下分为3个文件夹和一个顶层Mips.v文件，三个文件夹分别是Datapath、Generatic、Define。其中，Define主要包含了各个模块的一些宏定义，Generatic包含了一些在Datapath使用的小组件，Datapath则是包含了IF、ID、EXE、MEM、WB五个文件夹分别代表了五级流水，具体说明如下：

（1）、IF

其下包含了两个文件夹PC和IF_ID_REG，PC主要是对PC_reg、PC_plus4和PC数值异常检测的封装，IF_ID_REG是流水线reg可以记录从ICache读来的数据、instruction的asid等信息。

（2）、ID

其下包含了ID_EXE_REG用于存储信号，Control_Unit用于译码和有关instruction的异常检测（如break、syscall、RI等），victimInstDector用于获得发生异常时受影响instruction的信息（如asid、is_delayslot等），Hazard_Detection_Unit用于产生lw、jmp的阻塞（这里之所以要阻塞jmp是因为jmp在ID级判断是否跳转，需要阻塞一周期适应旁路单元），剩下的就是Registers、hi_lo_reg、COP0，统一在WB阶段posedge clk写回，COP0主要用于处理异常。

（3）、EXE

其下主要是ALU和Divider，用于进行计算。Forwarding_Unit用于产生rf_rdata0、rf_rdata1、hi_rdata等数据的旁路信号，EXE_MEM_REG系流水线寄存器。

（4）、MEM

其下有 MEM_WB_REG 为封装的流水线寄存器，modify_tlbr_result 是将读出的 tlbr 读出的数据根据 EXE 级的 rd 进行处理，获得相应的数据用于旁路，TLBExcDetector 主要是用于检测 TLB 的产生的异常 modified_store_data 本是用于根据 store_type 修改即将写入 mem 或者外设的数据，但是在 DCache 自带使能端没有什么意义，可以忽略，uncachedLoader 和 uncachedStorer 主要是对 uncached 段的访存，MMU 则包括了 TLB 和 memmap。

（5）、WB

这一级包含 modified_load_data，是用于根据 load_type 修改写入 rf 的数据。

三、设计结果

（一）设计交付物说明

—cpu132_gettrace	gs132 工程，测试用文件夹
—soc_axi_func	axi func test 工程
—soc_axi_pref	axi pref test 工程
—soc_axi_pref_demo	axi pref test demo 工程
—soft	测试用源代码，含 coe、mif 文件
—design.pdf	设计文档
—score.xls	分数

（二）设计演示结果

本实验实现了一个 Haribote MIPS CPU，是一个基本的五级流水线，直接用 soc_axi_func、soc_axi_pref 中的 bit 文件下板，即可得到我们设计 MIPS CPU 的运行效果。

四、参考设计说明

本实验是我在自己大二下学期 SoC 实验课上写的 CPU 架构下（https://github.com/Fassial/SoC-Lab），参考上一届武汉大学学长的代码（git@bitbucket.org:IzzyTang/yxmips.git，他的仓库是 privated 的），写出了 sram 接口的 v1.0，跑通了 sram_func_test。然后参考清华大学第一届的参赛代码（https://github.com/z4yx/NaiveMIPS-HDL）和比赛提供的 A13TLB 文档写出了带有 TLB 的 MMU，我为了适应自己架构，又加入了一些旁路信号，并为了加入 TLB 的异常机制，大改了之前的 COP0，新支

持了所有有关 TLB 的异常，并通过了比赛所发布的 TLB 检测。之后便是写改成 axi 接口和 Cache，Cache 的第一版写法是参考了清华大学第一年的参赛作品，但是关于 Cache Miss 时候的请求部分我是自己阅读 AXI3 协议，debug 两天写出来的，后来 sim 测试出了问题发现有些访问内存是 uncached，我就又花了一天时间加入了 uncachedLoader 和 uncachedStorer，并加入了内部 AXI 接口的 Bus，并修改了一些控制信号，使状态机运行稳定。在通过了 axi_func_test 之后，便开始 pref_test，这一部分我也是自己将 cacheline 中的 reg 改成用 distributed ram 实现，然后发现效果不好，检查了一下板上资源，又改成 block ram 实现，往上拉了一下频率，但由于内部流水线设计的问题，比如写回阶段布线跨越三个流水级使得布线 delay 很高，以及 divider 的 logic delay，拉到 38M 左右就拉不动了。

五、参考文献

[1] David A. Patterson/John L. Hennessy. 《计算机组成与设计 硬件/软件接口(MIPS 版)》[M]. 中国: 机械工业出版社, 2013.

[2] 水头一寿/米泽辽. 《CPU 自制入门》[M]. 中国: 人民邮电出版社, 2014.

[3] Dominic Sweetman. 《See MIPS Run Linux》[M]. 中国: 机械工业出版社, 2008.

[4] 胡振波. 《手把手教你设计 CPU——RISCV 处理器》[M]. 中国: 人民邮电出版社, 2018.

[5] 吴厚航. 《勇敢的芯伴你玩转 Altera FPGA》[M]. 中国: 清华大学出版社, 2017.

[6] 李亚民. 《计算机原理与设计》[M]. 中国: 清华大学出版社, 2011.

[7] MD00086, MIPS Architecture for Programmers Volume 2-A: The MIPS32 Instruction Set Manual[S]. 美国: MIPS, 2016.

[8] ARM IHI 002B, AMBA AXI Protocol v1.0[S]. 美国: ARM, 2003.

[9] MD00090, MIPS32 Architecture For Programmers Volume 3: The MIPS32 Privileged Resource Architecture[S]. 美国: MIPS, 2001.