**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。

1. 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

说明所提交设计的目录层次，各目录下对应的内容是什么。提供所提交设计进行仿真、综合、上板演示的必要操作提示步骤。

（二）设计演示结果

以文字、图、表等形式展示设计的演示结果。

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.