重庆大学课程设计报告

课程设计题目:	MIPS SOC 设计与性能优化
学　　院:	计算机学院
专 业 班 级:	计算机科学与技术(卓越)02/01 班
年　　级:	2019
学　　生:	李燕琴　　杨思怡
学　　号:	20195633　　20195217
完 成 时 间:	2022 年　01 月　07 日
成　　绩:	90
指 导 教 师:	钟将

重庆大学教务处制

项目	分值	优秀 $100 > x \geq 90$	良好 $90 > x \geq 70$	中等 $80 > x \geq 70$	及格 $70 > x \geq 60$	不及格 $x < 60$	评分
		参考标准					
学习态度	15	学习态度认真，科学作风严谨，严格保证设计时间并按任务书中规定的进度开展各项工作	学习态度比较认真，科学作风良好，能按期圆满完成任务书规定的任务	学习态度尚好，遵守组织纪律，基本保证设计时间，按期完成各项工作	学习态度尚可，能遵守组织纪律，能按期完成任务	学习马虎，纪律涣散，工作作风不严谨，不能保证设计时间和进度	
技术水平与实际能力	25	设计合理、理论分析与计算正确，实验数据准确，有很强的实际动手能力、经济分析能力和计算机应用能力，文献查阅能力强、引用合理、调查调研非常合理、可信	设计合理、理论分析与计算正确，实验数据比较准确，有较强的实际动手能力、经济分析能力和计算机应用能力，文献引用、调查调研比较合理、可信	设计合理，理论分析与计算基本正确，实验数据比较准确，有一定的实际动手能力，主要文献引用、调查调研比较可信	设计基本合理，理论分析与计算无大错，实验数据无大错	设计不合理，理论分析与计算有原则错误，实验数据不可靠，实际动手能力差，文献引用、调查调研有较大的问题	
创新	10	有重大改进或独特见解，有一定实用价值	有较大改进或新颖的见解，实用性尚可	有一定改进或新的见解	有一定见解	观念陈旧	
论文(计算书、图纸)撰写质量	50	结构严谨，逻辑性强，层次清晰，语言准确，文字流畅，完全符合规范化要求，书写工整或用计算机打印成文；图纸非常工整、清晰	结构合理，符合逻辑，文章层次分明，语言准确，文字流畅，符合规范化要求，书写工整或用计算机打印成文；图纸工整、清晰	结构合理，层次较为分明，文理通顺，基本达到规范化要求，书写比较工整；图纸比较工整、清晰	结构基本合理，逻辑基本清楚，文字尚通顺，勉强达到规范化要求；图纸比较工整	内容空泛，结构混乱，文字表达不清，错别字较多，达不到规范化要求；图纸不工整或不清晰	

指导教师评定成绩：

指导教师签名：

MIPS SOC 设计报告

李燕琴、杨思怡

1 设计简介

本次设计为双发射五级流水线处理器,支持 MIPS32 体系结构。处理器成功通过六类指令的单独测试及功能测试,并成功上板。在一定程度上说明了设计的有效性。

1.1 小组分工说明

- 李燕琴:负责取指模块,前推逻辑,冒险模块,译码模块重构,regfile 多端口模块,分支模块,ALU 及 HILO 模块,访存模块
- 杨思怡:负责指令拆分及译码模块,发射控制逻辑模块,异常基本逻辑

2 设计方案

2.1 总体设计思路

LaunchMIPS 设计为双发射五级流水线,整体分为取指,译码,执行,访存,写回五个阶段。取指通过 64bit 的 inst_ram64 实现同时对两条指令的取值。当 pc[2:0] 为 000 时,正常返回两条指令;当 pc[2:0] 为 100 时,只返回一条指令;其他情况则导致 pc 异常。

取指阶段设计的核心为指令 FIFO,其内部存储指令及其对应 PC 地址。

在译码阶段分别为主流水线和辅流水线设置单独译码器,以对从指令 FIFO 取出的两条指令分别译码生成对应信号。同时该阶段由发射控制器根据双发射处理逻辑决定两条指令的发射情况。考虑到数据冲突问题,LaunchMIPS 在该阶段进行数据前推处理。

在执行阶段,主流水线和辅流水线可能并存,故同样设置了两个 ALU。该阶段还包括对分支跳转指令的处理,得到是否跳转及下条 PC 地址等信息。下条 PC 地址将被送入 PC 寄存器中用作后续更新处理。

在访存阶段,首先进行异常的处理,没有异常的情况下再进行访存处理。

在写回阶段,将结果写入相应地址。master 和 slave 同时写入 regfile,且当主辅流水线写入地址相同时,只写入最近的结果,即 slave 的结果。

设计代码详见链接: LaunchMIPS

2.2 数据通路

整体数据通路图见1。

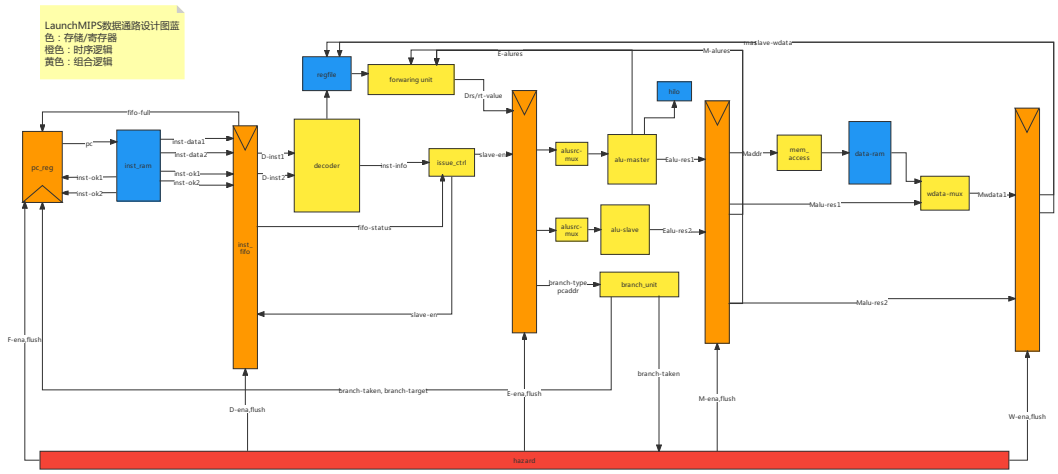

图 1: 数据通路图

3 设计过程

3.1 设计流水账

3.1.1 01.01

(1) 查找 Sirius 相关资料；
(2) 学习 Sirius 的实现框架和结构图

3.1.2 01.02

(1) 基于龙芯杯代码搭建大致 Sirius 框架；
(2) 取指阶段的理解
(3) FIFO 的理解

3.1.3 01.03

(1) 完成取指阶段
(2) 完成 inst_ram64 及相应 inst_diff 的搭建

3.1.4 01.04

(1) 指令的拆分

(2) 指令的译码
(3) dual_issue 判断模块
(4) 前述模块添加至 datapath
(5) 前推逻辑实现及添加至 datapath
(6) regfile 多端口模块实现及添加至 datapath
(7) branch unit 实现
(8) ALU 设计
(9) HILO 实现
(10) 访存前的解析
(11) 写回数据多选器实现

3.1.5 01.05

(1) 完成 Sirius 框架搭建
(2) 冒险模块处理
(3) ArithmeticTest
(4) DataMoveInstTest
(5) LogicInstTest
(6) ShiftInstTest

3.1.6 01.06

(1) j_BTest
(2) S_LInstTest
(3) obj1 测试
(4) obj2 测试
(5) 异常逻辑添加

3.1.7 01.07

(1) obj3 测试
(2) 完整 obj 测试及上板

3.2 错误记录

3.2.1 错误 1

(1) 错误现象:PC 错误,写寄存器数据错误,写寄存器号错误

(2) 分析定位过程: 写回时,发现除法执行阶段,reg_wen 一直使能,导致 trace 测试一直往后测试,导致错误产生。
(3) 错误原因: 除法未阻塞写回阶段
(4) 修正效果: 成功通过 obj1 测试

3.2.2 错误 2

(1) 错误现象:PC 错误,写寄存器数据正确,写寄存器号正确
(2) 分析定位过程: 在前几条指令内发生错误,且这些指令和 obj1 测试类似。
(3) 错误原因:测试 coe 和 trace 文件对应错误
(4) 修正效果: 成功通过 obj2 测试

4 设计结果

4.1 设计交付物说明

包括将 32bit 的 inst_ram.coe 转为 64bit 的 inst_ram.coe 文件的 python 程序源文件。以及 rtl 文件下核心代码文件。具体目录说明见图目录说明图2。

4.2 设计演示结果

具体结果见功能测试通过图3。

5 参考设计说明

代码基于 2021 年 NSCSCC 参赛队伍CPUchildren 项目;
总体设计主要参考 2019 年 NSCSCC 北京邮电大学Sirius 项目;
FIFO 延迟槽设计参考 2020 年 NSCSCCUltra_NSCSCC 项目.

6 总结

回望这几天的开发之路,确实百感交集。万事开头难这句话不是骗人的。最开始接触双发射时只能看优秀设计的 design report 去理解逻辑。一个指令 fifo 的实现我们讨论了一个多小时也没有清晰的结果。也在 Sirius 和自身之前的代码实现之间纠结过哪种实现更佳。奇奇怪怪的 bug 和难搞的 vivado,更是一个也不好惹。日常的熬夜自是不必说。

```
2021-CquCOlab-src
│  inst32_inst64.py            将32bit的inst_sram.coe转为64bit的inst_sram_64.coe
│  inst_sram_64.xci            inst_sram_64配置
│  readme.txt                  文件目录说明
│
├─rtl
│  │  soc_lite_top.v           顶层文件，修改cpu_inst_wen、cpu_inst_wdata、cpu_inst_rdata的位宽
│  │
│  └─myCPU                     myCPU源代码文件
│          alu_master.v        master分支的alu
│          alu_slave.sv        slave分支的alu
│          branch_judge.sv     分支判断逻辑
│          cp0_reg.sv          CP0寄存器
│          datapath.v          数据通路
│          decoder.sv          译码
│          defines.vh          宏定义
│          div.v               除法模块
│          exception.sv        异常因素解析
│          flopenr.v           触发器
│          forward_mux.sv      前推选择
│          forward_top.sv      前推顶层
│          hazard.v            控制冒险处理
│          hilo_reg.v          hilo寄存器
│          instdec.v           指令ascii编码
│          inst_diff.sv        2条指令读取控制
│          inst_fifo.sv        指令fifo控制数据通路F阶段
│          issue_ctrl.sv       2条指令发射控制
│          mem_access.sv       访存转换
│          mycpu_top.v         cpu顶层
│          pc_reg.sv           pc寄存器
│          regfile.sv          通用寄存器堆
│
└─testbench
        mycpu_tb.v             测试文件，修改测试的clk边沿出发
```

图 2: 文件目录说明图

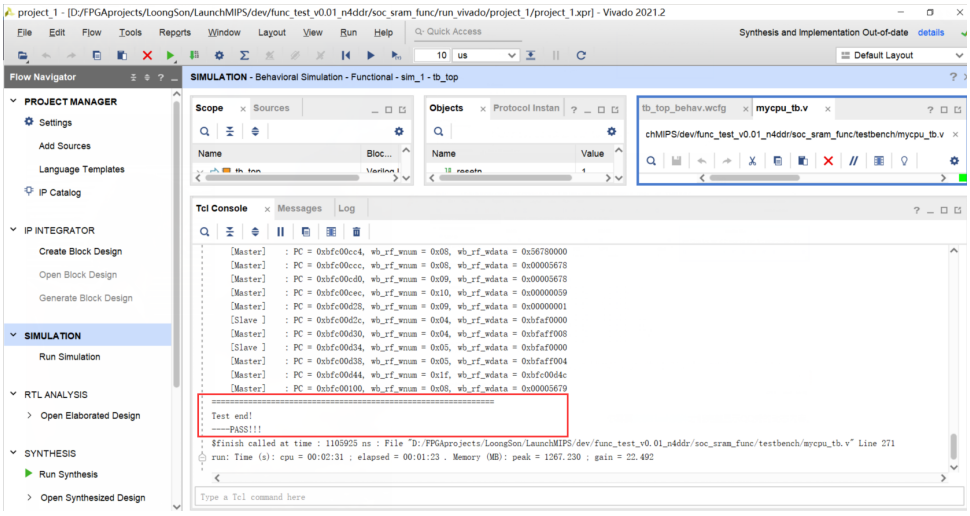

图 3: 功能测试通过图

在这个过程中,我们深切体会到开发的不易,也明白坚持目标的重要性,也对双发射有了更深入的了解。总而言之,这一路收获满满,算是满载而归。

7 供同学们吐槽之用。有什么问题都可以直接写在这。

7.1 李燕琴

1. vivado 跑得挺慢,崩得挺快
2. 写信号的时候用一个加一个,注意位宽,要不然奇怪的 bug 教你做人
3. 先画电路图! 先画电路图! 先画电路图! 重要的事情说三遍!

7.2 杨思怡

长夜漫漫。

参考文献

[1] 2019NSCSCC Sirius 项目
[2] 2020NSCSCC NonTrivialMIPS 项目
[3] 雷思磊. 自己动手写 CPU. 电子工业出版社, 2014.
[4] 2021NSCSCC 龙芯杯本队代码