中国科学院大学计算机组成原理（研讨课）

实 验 报 告

学号：2021K8009925006　姓名：冯浩瀚　专业：计算机科学与技术

实验序号：04　实验名称：定制 RISC-V 功能型处理器设计

注 1：撰写此 Word 格式实验报告后以 PDF 格式保存 SERVE CloudIDE 的 /home/serve-ide/cod-lab/reports 目录下（注意：reports 全部小写）。文件命名规则：prjN.pdf，其中“prj”和后缀名“pdf”为小写，“N”为 1 至 4 的阿拉伯数字。例如：prj1.pdf。PDF 文件大小应控制在 5MB 以内。此外，实验项目 5 包含多个选做内容，每个选做实验应提交各自的实验报告文件，文件命名规则：prj5-projectname.pdf，其中“-”为英文标点符号的短横线。文件命名举例：prj5-dma.pdf。具体要求详见实验项目 5 讲义。 注 2：使用 git add 及 git commit 命令将实验报告 PDF 文件添加到本地仓库 master 分支，并通过 git push 推送到 SERVE GitLab 远程仓库 master 分支（具体命令详见实验报告）。 注 3：实验报告模板下列条目仅供参考，可包含但不限定如下内容。实验报告中无需重复描述讲义中的实验流程。

一、 逻辑电路结构与仿真波形的截图及说明（比如关键 RTL 代码段{包含注释}及其对应的逻辑电路结构图{自行画图，推荐用 PPT 画逻辑结构框图，复制到 word 中}、相应信号的仿真波形和信号变化的说明等）

本次实验的任务基于prj3，将代码从MIPS指令集迁移到RISC-V指令集中，代码的结构几乎完全一致，因此综合得到的逻辑电路结构以及仿真波形也几乎完全一致，不再赘述。

二、 实验过程中遇到的问题、对问题的思考过程及解决方法（比如 RTL 代码中出现的逻辑 bug，逻辑仿真和 FPGA 调试过程中的难点等）

1. 代码中出现语法错误：位扩展语法要注意中括号的使用，如图所示。

```
wire [31:0] J_imm = {{12{valid_Instruction[31]}}, valid_Instruction[19:12], valid_Instruction[20], valid_Instruction[30:21], 1'b0};
```

2. 在 Alu_control 中漏考虑了 I-type 指令（即 jalr 指令）也需要进行 ADD 运算：以后要仔细检查核对自己列的指令表中内容是否正确

```
wire [2:0] ALU_control = {{3{type == `RAlu && funct3 == 3'b000}} & {funct7[5], 2'b10}} | // add & sub
                         {{3{type == `IAlu && funct3 == 3'b000}} & `ALUOP_ADD} | // addi
                         {{3{type == `RAlu && funct3[2:1] == 2'b11 }} & ~funct3} | // and & or
                         {{3{type == `RAlu && funct3 == 3'b100}} & funct3} | // xor
                         {{3{type == `IAlu && funct3[2:1] == 2'b11 }} & ~funct3} | // andi & ori
                         {{3{type == `IAlu && funct3 == 3'b100}} & funct3} | // xori
                         {{3{type == `RAlu && funct3[2:1] == 2'b01}} & {~funct3[0], 2'b11}} | // slt & sltu
                         {{3{type == `IAlu && funct3[2:1] == 2'b01}} & {~funct3[0], 2'b11}} | // slti & sltiu
                         {{3{type == `BType && funct3[2:1] == 2'b00}} & `ALUOP_SUB} | // beq & bne
                         {{3{type == `BType && funct3[2:1] == 2'b10}} & `ALUOP_SLT} | // blt & bge
                         {{3{type == `BType && funct3[2:1] == 2'b11}} & `ALUOP_SLTU} | // bltu & bgeu
                         {{3{type == `ILoad || type == `SType || type == `IType}} & `ALUOP_ADD}; // lw & sw & jalr
```

3. 本次实验在 microbench 阶段未发生 bug，因此未使用 fpga_emu 进行调试。

三、 对讲义中思考题（如有）的理解和回答

1. 通过功能和性能评估，对比 RISC-V/MIPS 指令集译码器的实现开销，理解 RISC-V 指令格式的设计思想

答：下表列出了两个指令集分别在 fpga_run 阶段 9 个 microbench 的周期数

	MIPS	RISC-V	相对减少的周期数
15pz	529665412	525402450	0.80%
bf	46342578	38812381	16.25%
dinic	1702924	1480479	13.06%
fib	179414184	181091746	-0.94%
md5	403514	384697	4.66%
qsort	703126	784194	-11.53%
queen	6843655	6905774	-0.91%
sieve	1191547	744278	37.54%
ssort	52487070	44712232	14.81%
总计	818754010	800318231	2.25%

由上表，虽然同属于精简指令集，RISC-V 指令集在总体上比 MIPS 指令集周期

数少,在相同的机器上运行时所需时间更少。特别是 sieve(质数筛选)中,RISC-V 实现了接近 40%的周期缩减。

四、 在课后，你花费了大约 5 小时完成此次实验。

五、 对于此次实验的心得、感受和建议（比如实验是否过于简单或复杂，是否缺少了某些你认为重要的信息或参考资料，对实验项目的建议，对提供帮助的同学的感谢，以及其他想与任课老师交流的内容等）

本次实验基于上个实验的代码，总体较为容易，因此出现的 bug 也较少，调试过程也较为轻松。最终，感谢老师以及助教团队的辛勤付出。