《数字逻辑与数字系统》实验报告

天津大学本科生实验报告

学院 计算机科学与技术 年级 2019 级 班级 一班 姓名 李润泽 学号 3019244266 课程名称 数字逻辑与数字系统

实验日期 2021/7/2 成绩

实验项目名称 单周期 MIPS 处理器的设计与实现

一. 实验目的

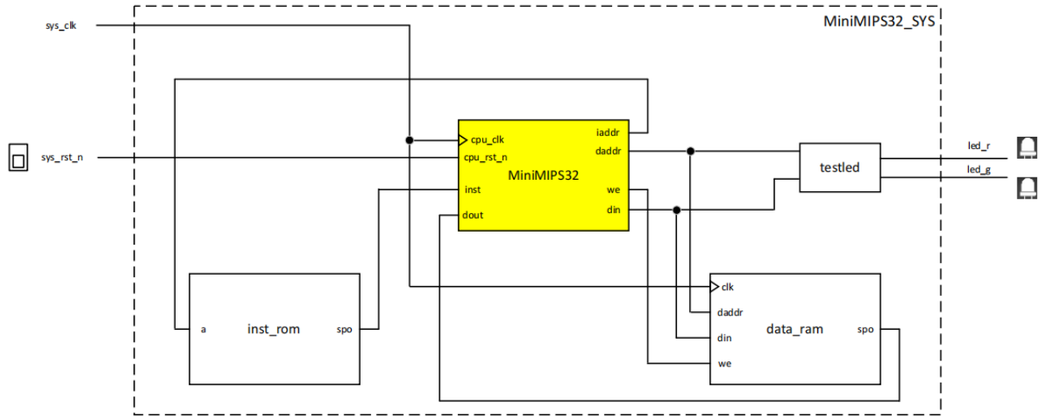
1. 熟悉 MIPS 处理器的常用指令集（10 条）
2. 掌握单周期处理器数据通路和控制单元的设计方法
3. 基于增量方式，实现单周期 MIPS 处理器；
4. 基于测试用例对所设计的单周期 MIPS 处理器进行功能验证。

二. 实验内容

基于 SystemVerilog HDL 设计并实现单周期 MIPS 处理器——MiniMIPS32。

该处理器具有如下特点：

- 32 位数据通路
- 小端模式
- 支持 10 条指令：lw、sw、lui、ori、addiu、addu、slt、beq、bne 和 j
- 寄存器文件由 32 个 32 位寄存器组成，采用异步读/同步写工作模式
- 采用哈佛结构（即分离的指令存储器和数据存储器），指令存储器由 ROM 构成，采用异步读工作模式；数据存储器由 RAM 构成，采用异步读/同步写工作模式。



本实验的顶层模块 MiniMIPS32_SYS 如图所示。

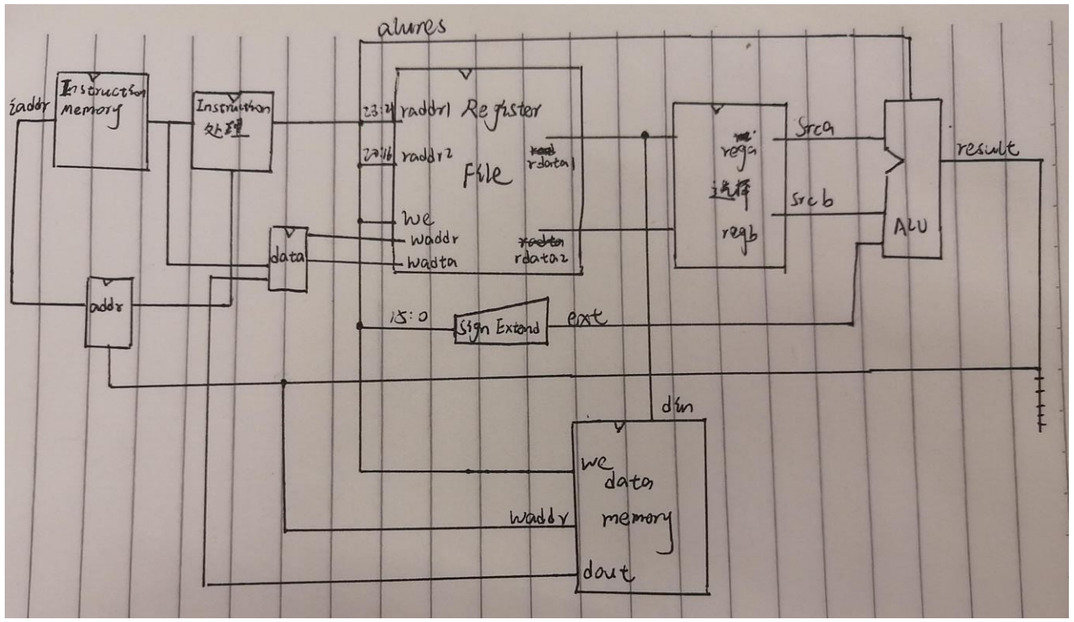
端口名	方向	宽度（位）	作用
sys_clk	输入	1	系统输入时钟，主频为 25MHz（40ns）
sys_rst_n	输入	1	系统复位，低电平有效，连接拨动开关 S
led_r	输出	1	用于显示程序是否正确，红色 LED 灯。如果程序正确，不点亮，否则显示红色。
led_g	输出	1	用于显示程序是否正确，绿色 LED 灯。如果程序正确，显示绿色，否则不点亮。

表中给出了顶层模块 MiniMIPS32_SYS 的输入/输出端口。

最终设计实现的单周期 MIPS 处理器能够运行所提供的 6 个测试用例 mem.S，i-type.S，r-type.S，branch.S，sort_sim.S 和 sort_board.S。其中，前 5 个只能用于功能仿真；最后一个可以上传到远程 FPGA 硬件云平台完成功能验证，如果测试通过则 LED 灯 led_g 被点亮为绿色，否则 LED 灯 led_r 被点亮为红色。

三．实验原理与步骤（注：步骤不用写工具的操作步骤，而是设计步骤）

1. **画出所实现的单周期 MIPS 处理器原理图。**



2. **写出单周期 MIPS 处理器所需的所有控制信号，并通过表格列出每条指令所对应的控制信号的取值。**

operate	op	Se	alures	regWe	we	imm0_5	imm6_10
lw	100011	1	10	1	0	-	-
sw	101011	1	10	0	1	-	-
lui	1111	0	0	1	0	-	-
ori	1101	0	1	1	0	-	-
addiu	1001	1	10	1	0	-	-
addu	0	0	10	1	0	100001	0
slt	0	0	11	1	0	101010	0
beq	100	1	100	0	0	-	-
bne	101	1	100	0	0	-	-
j	10	0	-	0	0	-	-

其中

- op：当前指令操作码
- Se：符号扩展识别码
- alures：alu 操作码
- regWe：是否需要写入寄存器
- we：是否需要写入数据存车
- imm_{0_5}：取当前指令 0-5 位
- imm_{6_10}：取当前指令 6-10 位

3. 叙述每条指令在单周期 MIPS 处理器中的执行过程。

首先要在 Instrction memory 中取指令，并做预处理，即改成符合小端的格式。

- **lw：**指令的高 6 位为 100011，则可以判断此指令为 lw。
  首先，将 alures 置为 010，regWe 置为 1，将 Se 置为 1。
  其次，需要对操作数进行符号扩展。将指令的低 16 位扩展为 32 位存入 ext。
  其次，从 data_ram 中取值并写入寄存器。将指令 21-25 位存入 base，并找到对应该地址的寄存器，将其中的值取出，与 ext 输入 ALU 中进行加法计算，结果存入 result。从 data_ram 中找到地址为 result 的数据，存入 dout，取指令 16-20 位存入 rt，将 dout 写入地址为 rt 的寄存器。

- **sw：**指令的高 6 位为 101011，则可以判断此指令为 sw。
  首先，将 alures 置为 010，regWe 置为 1，we 置为 1，将 Se 置为 1。
  其次，需要对操作数进行符号扩展。将低 16 位存入 offset，并进行符号扩展，存入 ext。其次，从寄存器中取值写入 data_ram。将指令 21-25 位存入 base，并找到对应该地址的寄存器，将其中的值取出，与 ext 输入 ALU 中进行加法计算，结果存入 result。取指令 16-20 位存入 rt，将地址为 rt 的寄存器中的值取出，写入地址为 result 的存储器中。

- **lui：**指令的高 6 位为 001111，则可以判断此指令为 lui。
  首先，将 alures 置为 000，regWe 置为 1。
  其次，将 16 位立即数 imm 写入寄存器 rt 的高 16 位，寄存器 rt 的低 16 位置 0。即将指令的低 16 位存入 imm16。imm16 输入 ALU 进行左移操作，结果存入 result。取指令 16-20 位存入 rt，将 result 写入地址为 rt 的寄存器。

- **ori：**指令的高 6 位为 001101，则可以判断此指令为 ori。
  首先，将 alures 置为 001，regWe 置为 1。
  其次，寄存器 rs 中的值与 0 扩展至 32 位的立即数 imm 按位逻辑或，结果写入寄存器 rt 中。即将低 16 位存入 imm16，将指令 21-25 位存入 rs，取出地址为 rs 的寄存器的值，与 imm16 输入 ALU 进行逻辑或运算，结果存入 result。将指令 16-20 位存入 rt，将 result 写入地址为 rt 的寄存器。

- **addiu：**指令的高 6 位为 001001 ，则可以判断此指令为 addiu。
  首先，将 alures 置为 010，regWe 置为 1，将 Se 置为 1。
  其次，需要对操作数进行符号扩展。将低 16 位扩展为 32 位存入 ext。
  其次，进行加法操作，并将结果写入寄存器。将地址位于 rs 的寄存器的值取出，与 ext

输入 ALU 进行加法运算，结果存入 result，将 result 写入地址为 rt 的寄存器。

- addu：指令的高 6 位为 000000，则为 R 型指令，其 0-5 位为 100001，判断其指令类型为 addu。 首先，将 alures 置为 010，regWe 置为 1。
  其次，进行加法操作，并将结果写入寄存器。将地址为 rs 和 rt 的寄存器的值取出，输入 ALU 进行加法运算，结果存入 result。将 result 存入地址为 rd 的寄存器。

- slt：指令的高 6 位为 000000，则为 R 型指令，其 0-5 位为 101010。判断其指令类型为 slt。
  首先，将 alures 置为 011，regWe 置为 1。
  其次，将地址为 rs 和 rt 的寄存器的值取出，输入 ALU 进行比大小运算，结果存入 result。将 result 存入地址为 rd 的寄存器。

- beq：指令的高 6 位为 000100 ，则可以判断此指令为 beq。
  首先，将 alures 置为 100，regWe 置为 0。
  其次，将地址为 rs 和 rt 的寄存器的值取出，输入 ALU 进行比较，查看 ZF。若结果为真，说明 rs 与 rt 值相等，实现地址跳转，iaddr 变为 iaddr + offer + 4。

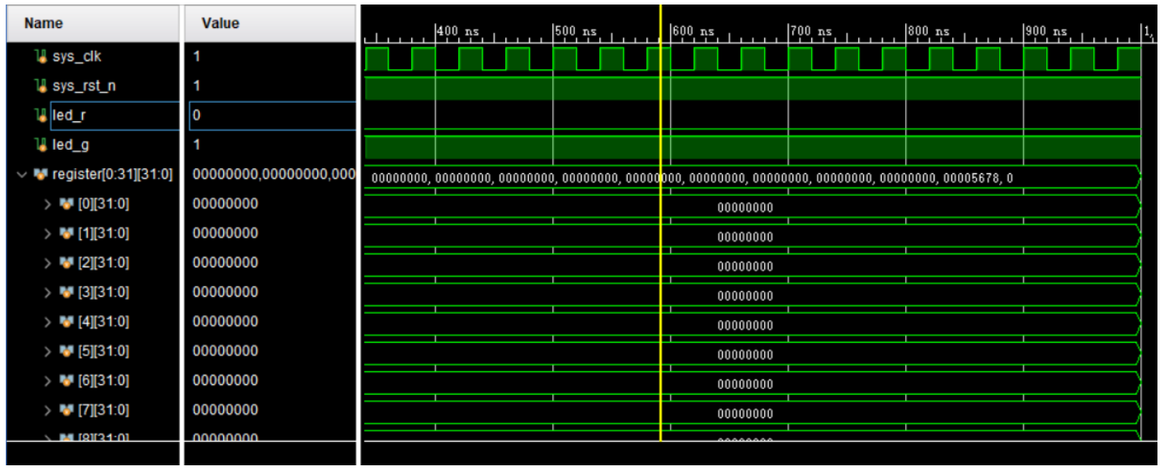
- bne：指令的高 6 位为 000101 ，则可以判断此指令为 bne。
  首先，将 alures 置为 100，regWe 置为 0。
  其次，将地址为 rs 和 rt 的寄存器的值取出，输入 ALU 进行比较，查看 ZF。若结果为假，说明 rs 与 rt 值不等，实现地址跳转，iaddr 变为 iaddr + offer + 4。

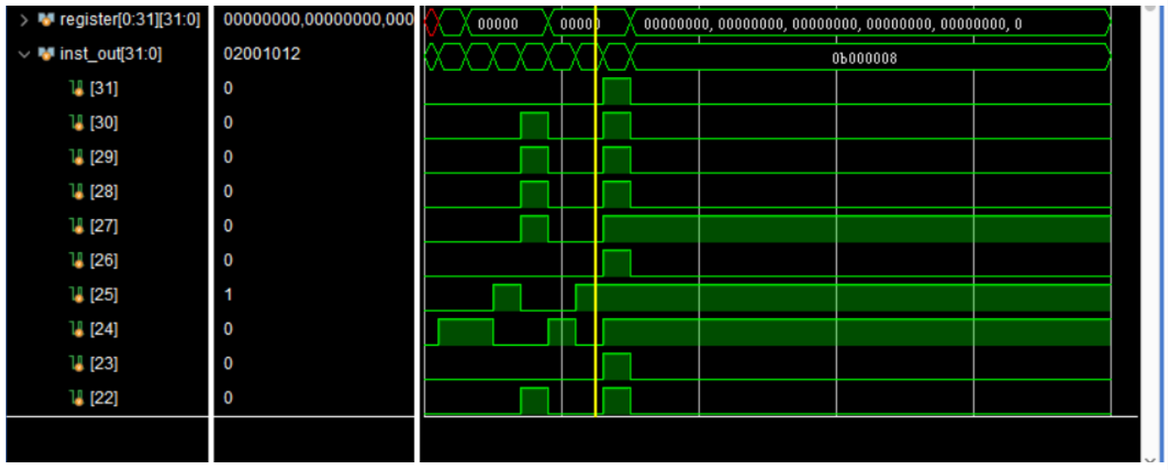
- j：指令的高 6 位为 000010，则可以判断此指令为 j。
  首先，将 regWe 置为 0。
  其次，实现地址跳转。将 instr_index 左移两位，iaddr 变为 iaddr + instr_index + 4。

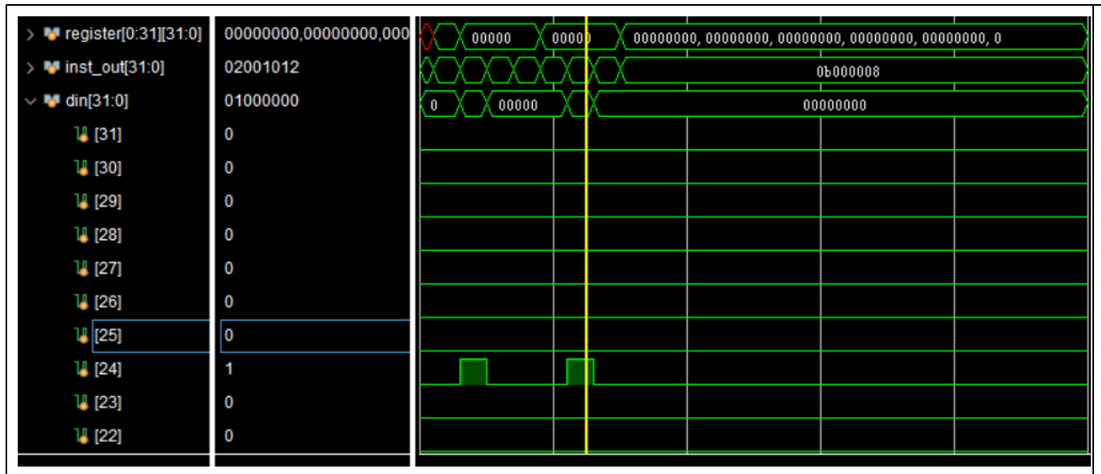
四．仿真与实验结果（注：仿真需要给出波形图截图，截图要清晰，如果波形过长，可以分段截取；实验结果为开发板验证的截图）

注：给出仿真波形图（仅需要提供一条指令的波形图），不需要给出板级截图。

以第四个测试样本，branch 为例。







五．实验中遇到的问题和解决办法

实验前要先了解各个指令的功能与执行逻辑，不能落下每一个步骤。比如，对于立即数指令，就要记得将立即数的位数进行扩展。

MiniMIPS32 实验使用的是小端模式，所以在读入数据（或指令）、写入数据时，都需要对数据（或指令）进行一些处理，即将数据改为小端格式。

六．附加题（若实验指导书无要求，则无需回答）

如果按照大端方式存储程序和数据，为了使处理器能够正常运行，需要对其进行哪些修改？

MiniMIPS32 实验使用的是小端模式，但在识别指令或者读写数据时，我们按照的是大端的格式。所以本实验在读入数据（或指令）、写入数据时，我们都需要对数据（或指令）进行一些处理，即将数据改为小端格式。

若实验按照大端方式存储程序和数据，我们就不需要对数据进行预处理，可以直接按照大端的格式对数据（指令）进行读写。

教师签字：

年　月　日