《数字逻辑与数字系统》实验报告

学院 智能与计算学院 年级 2022 级 班级 4 班 姓名 陆子毅学号 3022206045

课程名称 数字逻辑与数字系统 实验日期 2024.6.24 成绩_____

同组实验者_____

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 SystemVerilogHDL 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 4-3 所示。

该顶层模块由 4 部分构成，每部分的功能如下：

MiniMIPS32 是本实验所需要设计完成的单周期 MIPS 处理器。输入端口 inst 和 dout 分别用于从指令存储器和数据存储器中接收指令和数据。输出端口 iaddr 和 daddr 用于传输访问指令存储器和数据存储器的地址。输出端口 we 用于给出数据存储器的写使入数据存储器的数据。

inst_rom 是指令存储器，用于存放所需要执行的指令。inst_rom 指令存储器的深度为

天津大学本科生实验报告

256，宽度为 32 位，故其容量为 1KB。inst_rom 具有一个输入端口 a 和输出端口 spo，其中输入端口 a 用于接收待访问指令的地址，其宽度为 8 位，输出端口 spo 用于输出指令，其宽度为 32 位。

data_ram 是数据存储器，用于存放需要访问的数据。data_ram 数据存储器的深度为 256，宽度为 32 位，故其容量为 1KB。输入端口 daddr 为 8 位，用于给出访存地址。输入端口 din 为 32 位，用于给出待写入数据存储器中的数据。输入端口 we 为写使能信号，当其值“1”时，表示进行写操作，否则进行读操作。输出端口 spo 用于从数据存储器中读出数据。

testled 是测试模块，用于判断当前执行的程序是否正确。如果正确，则两个 LED 灯（led_r 和 led_g）分别为红色和不点亮状态，否则两个 LED 灯（led_r 和 led_g）分别为不点亮状态和绿色。

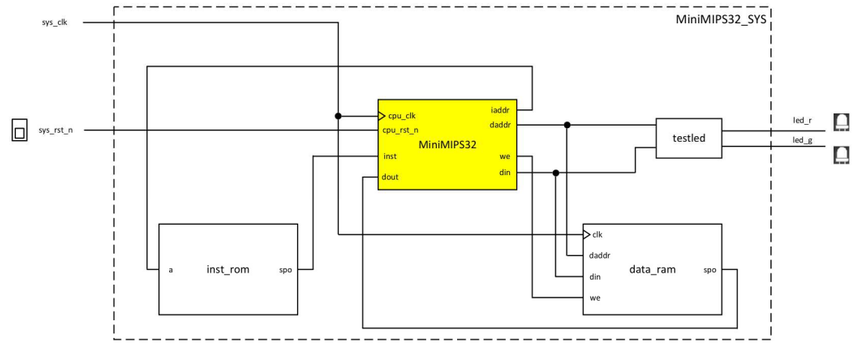


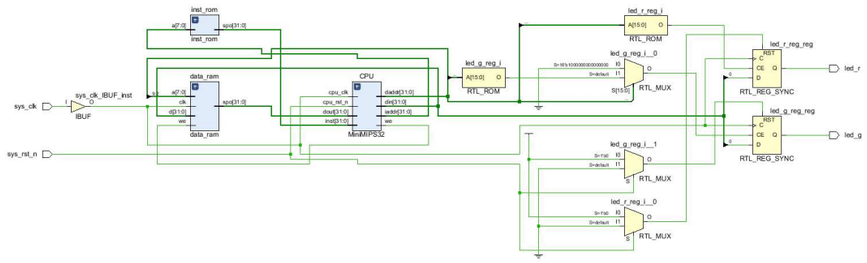
图 4-3 顶层模块 MiniMIPS32_SYS

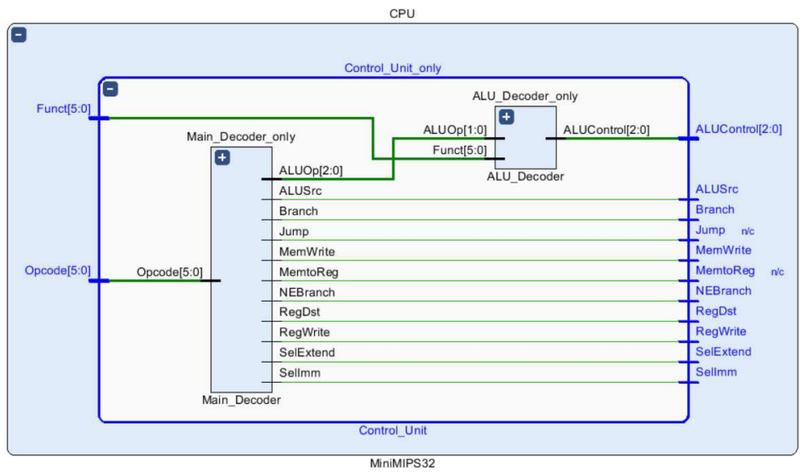
表 4-1 输入/输出端口

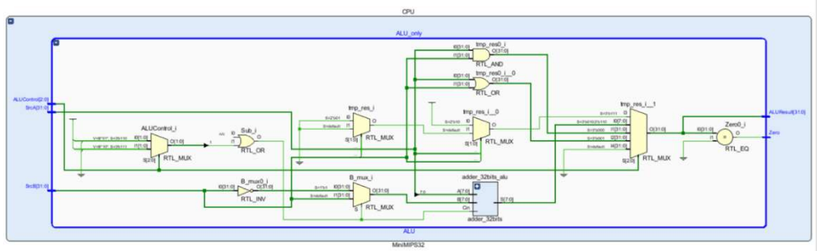
端口名	方向	宽度（位）	作用
sys_clk	输入	1	系统输入时钟，主频为 25MHz（40ns）
sys_rst_n	输入	1	系统复位，低电平有效，连接拨动开关 S
led_r	输出	1	用于显示程序是否正确，红色 LED 灯。如果程序正确，不点亮，否则显示红色。
led_g	输出	1	用于显示程序是否正确，绿色 LED 灯。如果程序正确，显示绿色，否则不点亮。

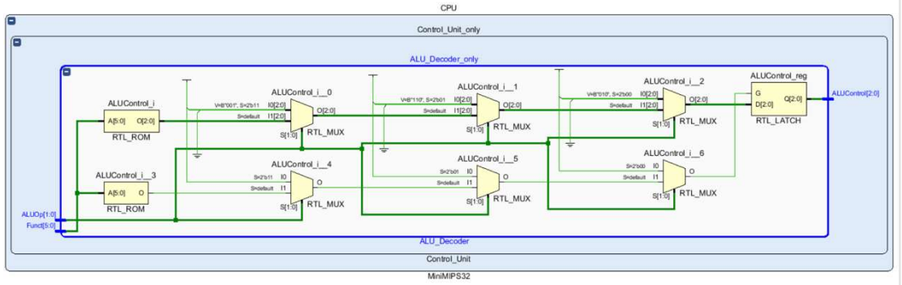
三．实验原理与步骤（注：步骤不用写工具的操作步骤，而是设计步骤）

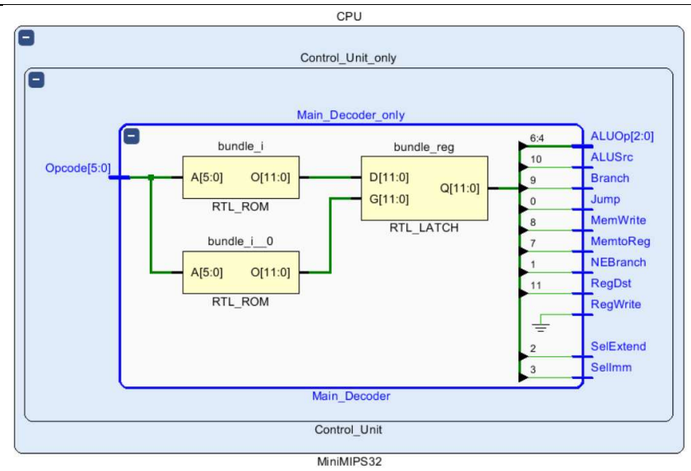
1. 画出所实现的单周期 MIPS 处理器原理图。











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。

Main_Decoder	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemtoReg	ALUOp	SelImm	SelExtend	NEBranch	Jump
6'b000000	1	1	0	0	0	0	10	X	X	0	0
6'b000010	0	X	X	X	0	X	X	X	X	X	1
6'b100011	1	0	1	0	0	0	00	0	1	0	0
6'b101011	0	X	1	0	1	0	00	0	1	0	0
6'b000100	0	X	1	0	0	1	01	X	X	0	0
6'b000101	0	X	1	0	0	1	01	X	X	1	0
6'b001111	1	0	1	0	0	0	00	1	X	0	0
6'b001101	1	0	1	0	0	0	11	0	0	0	0
6'b001001	1	0	1	0	0	0	00	0	1	0	0

ALU_Decoder	ALUControl
2'b00	3'b010
2'b01	3'b110
2'b11	3'b001
2'b10	(Funct)
6'b100000	3'b010
6'b100001	3'b010
6'b100010	3'b110
6'b100100	3'b000

3. 叙述每条指令在单周期 MIPS 处理器中的执行过程。

取址：通过 PC 寄存器的值从指令内存取出指令

pc <- pc+4

译码：经过 Control_Unit 产生控制信号

从寄存器文件取出相应寄存器值

立即数做各种扩展

两个数传入 Alu

计算：ALU 计算，得到结果和 Zero

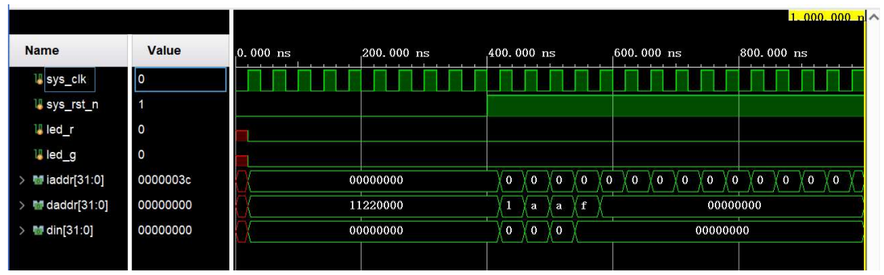
访存：根据 ALU 计算结果访问数据内存，写入或读出值

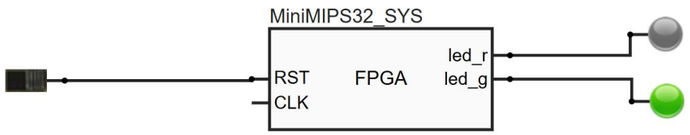
更新寄存器：把值写入寄存器文件

更新 PC：根据控制信号选择 PC 值来更新 PC

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。





五．实验中遇到的问题和解决办法

1.指令取不出来，没考虑小端问题，把数据和指令取出来之后每 8 位一组，按照小段重排即可，输出再次重排。

assign Instruction = {inst[7:0],inst[15:8],inst[23:16],inst[31:24]};

WD3 = {dout[7:0],dout[15:8],dout[23:16],dout[31:24]};

assign {din[7:0],din[15:8],din[23:16],din[31:24]} = RD2;

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？

把为了改成小端而添加的数据和指令重排模块去掉，调整为直接赋值

Instruction = {inst[7:0],inst[15:8],inst[23:16],inst[31:24]};

WD3 = {dout[7:0],dout[15:8],dout[23:16],dout[31:24]};

{din[7:0],din[15:8],din[23:16],din[31:24]} = RD2;

最简单的方式，直接改为：

Instruction = inst;

WD3 = dout;

din = RD2;

教师签字：

年　月　日