

华中科技大学

2021

计算机组成原理

· 实验报告 ·

专 业： 计算机科学与技术

班 级： CS1906

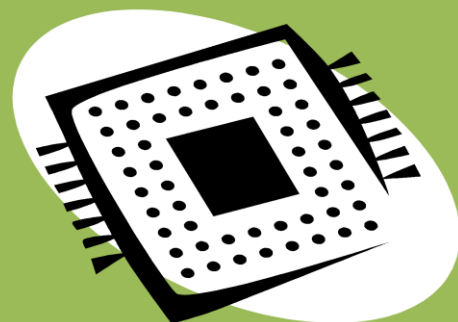
学 号： U201915115

姓 名： 郑舟

电 话： 13260547315

邮 件： 954988021@qq.com

完成日期： 2021. 12. 15



计算机科学与技术学院

华中科技大学课程实验报告

目 录

| | | |
|----------|-----------------|-----------|
| 1 | CPU 设计实验 | 1 |
| 1.1 | 设计要求 | 1 |
| 1.2 | 方案设计 | 2 |
| 1.3 | 实验步骤 | 10 |
| 1.4 | 故障与调试 | 10 |
| 1.5 | 测试与分析 | 12 |
| 2 | 总结与心得 | 14 |
| 2.1 | 实验总结 | 14 |
| 2.2 | 实验心得 | 14 |
| | 参考文献 | 16 |

1 CPU 设计实验

1.1 设计要求

要求设计三级时序变长指令周期的 RISC-V 单总线 CPU，以及单总线支持中断机制的现代时序 RISC-V CPU。要求支持的指令如下。

表 1.1 要求 CPU 支持的指令及描述

| 指令 | 汇编代码 | 指令功能 | 备注 |
|------|-----------------|---|--------------|
| lw | lw rd,imm(rs1) | $R[rd] \leftarrow M[R[rs1] + \text{SignExt}(imm)]$ | |
| sw | sw rs2,imm(rs1) | $M[R[rs1] + \text{SignExt}(imm)] \leftarrow R[rs2]$ | |
| beq | beq rs1,rs2,imm | if($R[rs1] == R[rs2]$) $PC \leftarrow PC + \text{SignExt}(imm) \ll 1$ | |
| slt | slt rd,rs1,rs2 | If ($rs1 < rs2$) $R[rd] \leftarrow 1$ else $R[rd] \leftarrow 0$ | |
| addi | addi rd,rs1,imm | $R[rd] \leftarrow R[rs1] + \text{SignExt}(imm)$ | |
| eret | eret | $PC \leftarrow EPC, IE \leftarrow 1$ | 只在中断机制中实现该指令 |

1.1.1 单总线三级时序变长指令周期 RISC-V CPU 设计

CPU 的总体架构已经给出，需要完成的任务是三级时序变长指令周期的硬布线控制器，使 CPU 能够运行表 1.1 中给出的前 5 条指令。具体任务包括设计指令译码器，设计，变长指令周期时序发生器状态机的设计，变长指令周期时序发生器输出函数的设计，硬布线控制器组合逻辑单元的设计，最后连接各单元设计硬布线控制器，使用排序程序检验 CPU 能否正确运行。

1.1.2 支持中断机制的现代时序 RISC-V 单总线 CPU

CPU 的总体架构同样已经给出，需要分别完成现代时序 RISC-V 单总线 CPU 的微程序控制器和硬布线控制器。

对微程序控制器，具体任务包括设计指令译码器，设计支持中断的微程序入口查找逻辑，设计支持中断的微程序条件判别测试逻辑，最后连接各单元设计，然后运行带中断排序程序检验 CPU 能否正确运行。

华中科技大学课程实验报告

2. 时序发生器状态机的设计

根据有限状态机的状态转换条件，填写 Excel 的状态转换表，如图 1.3 所示，主要注意 slt 指令和 addi 指令没有计算周期，与 lw, sw, beq 在取指周期的最后一个周期的状态转换情况不同。输入完成后即可得到次态与现态之间的逻辑表达式，然后将得到的逻辑表达式输入 logsim 的分析组合逻辑电路功能中即可自动生成逻辑电路。

| 当前状态(现态) | | | | | 输入信号 | | | | | | | 下一状态 (次态) | | | | |
|----------|----|----|----|------------|------|----|-----|-----|------|------|------|------------|----|----|----|----|
| S3 | S2 | S1 | S0 | 现态 10进制 | LW | SW | BEQ | SLT | ADDI | ERET | IntR | 次态 10进制 | N3 | N2 | N1 | N0 |
| 0 | 0 | 0 | 0 | 0 | | | | | | | | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | | | | | | | | 2 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 2 | | | | | | | | 3 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 3 | 1 | | | | | | | 4 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 3 | | 1 | | | | | | 4 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 3 | | | 1 | | | | | 4 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 3 | | | | 1 | | | | 6 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 3 | | | | | 1 | | | 6 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 4 | | | | | | | | 5 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 5 | | | | | | | | 6 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 6 | | | | | | | | 7 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 7 | | | | | | | | 8 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 8 | | | | | | | | 0 | 0 | 0 | 0 | 0 |

图 1.3 三级时序硬布线控制器时序发生器状态机状态转换表

3. 时序发生器输出函数设计

时序发生器输出的是机器周期信号和节拍电位信号。一条指令有三个机器周期，其中取值周期有四个节拍电位，计算周期有两个节拍电位，执行周期有三个节拍电位，一个状态对应一个节拍电位，将这些信息填入 Excel 表中，如图 1.4，通过 Excel 得到输出与现态之间的逻辑表达式，然后利用 logsim 的分析组合逻辑电路功能即可。

| 当前状态(现态) | | | | | 输出 | | | | | | | | | | | |
|----------|----|----|----|------------|-----|------|-----|------|----|----|----|----|------|-------|-------|-------|
| S3 | S2 | S1 | S0 | 现态 10进制 | Mif | Mcal | Mex | Mint | T1 | T2 | T3 | T4 | Out9 | Out10 | Out11 | Out12 |
| 0 | 0 | 0 | 0 | 0 | 1 | | | | 1 | | | | | | | |
| 0 | 0 | 0 | 1 | 1 | 1 | | | | | 1 | | | | | | |
| 0 | 0 | 1 | 0 | 2 | 1 | | | | | | 1 | | | | | |
| 0 | 0 | 1 | 1 | 3 | 1 | | | | | | | 1 | | | | |
| 0 | 1 | 0 | 0 | 4 | | 1 | | | 1 | | | | | | | |
| 0 | 1 | 0 | 1 | 5 | | 1 | | | | 1 | | | | | | |
| 0 | 1 | 1 | 0 | 6 | | | 1 | | 1 | | | | | | | |
| 0 | 1 | 1 | 1 | 7 | | | 1 | | | 1 | | | | | | |
| 1 | 0 | 0 | 0 | 8 | | | 1 | | | | 1 | | | | | |

图 1.4 三级时序硬布线控制器时序发生器输出函数表

4. 硬布线控制器组合逻辑单元

组合逻辑单元的主要任务是根据机器周期电位和节拍电位给出不同指令每个节拍的的控制信号。首先要分析每个节拍不同指令需要的不同信号，然后将其填写到 Excel 表格中，如图 1.5 所示。通过 Excel 自动生成控制信号与输入的机器周期电位、节拍电位

华中科技大学课程实验报告

现态连接到输出函数既可输出状态节拍信号。电路图如图 1.6 所示。

1.2.2 支持中断机制的现代时序 RISC-V 单总线 CPU

1. RISC-V 指令译码器设计

设计方法与 1.2.1 中指令译码器的设计一样，电路图见图 1.2。

2. 支持中断的微程序入口查找逻辑

任务是通过输入的信号，判断在取指的最后一个微程序完成后应跳转到的微程序的地址。每个程序对应的微程序入口地址都不同，所以每一个信号对一个入口地址即可，按图 1.7 填写 Excel 表格自动生成逻辑表达式，使用 logisim 自动生成组合逻辑电路即可。

| LW | SW | BEQ | SLT | ADDI | ERET | 入口地址 10进制 | S4 | S3 | S2 | S1 | S0 |
|----|----|-----|-----|------|------|--------------|----|----|----|----|----|
| 1 | | | | | | 4 | 0 | 0 | 1 | 0 | 0 |
| | 1 | | | | | 9 | 0 | 1 | 0 | 0 | 1 |
| | | 1 | | | | 14 | 0 | 1 | 1 | 1 | 0 |
| | | | 1 | | | 19 | 1 | 0 | 0 | 1 | 1 |
| | | | | 1 | | 22 | 1 | 0 | 1 | 1 | 0 |
| | | | | | 1 | 25 | 1 | 1 | 0 | 0 | 1 |

图 1.7 微程序入口查找逻辑设计表

3. 支持中断的微程序条件判别测试逻辑

任务是根据微指令字中的判别测试字段和条件反馈信息生成后续地址，输出是地址在多路选择器对应的序号。在没有判别测试字段信号时直接到下址字段，在有 P0 判别测试信号时跳转到微程序入口地址；在有 P1 判别测试信号时，若 equal 信号为 1 跳转到 beq 微指令地址，equal 为 0 时跳转到取指微程序入口；在有 P2 判别测试信号时，若有中断信号则跳转到中断入口，否则跳转到取值微指令入口。将对应值填入 Excel 表格，如图 1.8，生成逻辑表达式并利用 logisim 自动生成组合逻辑电路。

| P0 | P1 | P2 | equal | IntR | S2 | S1 | S0 |
|----|----|----|-------|------|----|----|----|
| 0 | 0 | 0 | | | 0 | 0 | 0 |
| 1 | | | | | 0 | 0 | 1 |
| | 1 | | 1 | | 0 | 1 | 0 |
| | 1 | | 0 | | 1 | 0 | 0 |
| | | 1 | | 0 | 1 | 0 | 0 |
| | | 1 | | 1 | 0 | 1 | 1 |

图 1.8 条件判别逻辑组合逻辑设计表

4. 支持中断的微程序控制器设计

本次实验使用直接表示法和计数器法对微指令进行编码。分析每个微程序需要给

下址。

同时由于 beq 指令判断 equal 为 0 后相当于到了最后一条微程序，需要进行中断判断，而这在判别测试逻辑中没有体现出来，所以需要对判别测试逻辑输入的最后一条微程序标志做一些处理，即当 P2 为 1，或者 P1 为 1 且 equal 为 0 时都算作到了最后一条微程序。电路图如图 1.10。

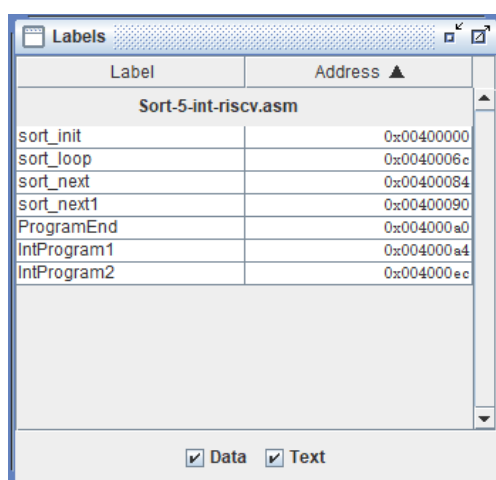
5. 支持中断的微程序单总线 CPU 设计

主要任务是完成与中断相关的硬件模块，主要包括异常程序地址计数器 EPC，中断使能寄存器 IE，中断控制器等模块的连接，题目中已经给出这几个模块以及中断控制信号分解出的几个控制具体元器件的信号。

异常程序地址计数器 EPC 是用来保存断点地址的，故其输入输出都应该连接到内总线上，与程序计数器 PC 交换数据，输入端使用 mEPCin 控制断点地址的写入，输出端加一个三态门使用信号 mEPCout 控制完成中断时恢复断点地址。

中断使能寄存器 IE 在可以接收中断时值为 0，否则为 1，所以开中断信号接异步置 0 端，关中断信号接异步置 1 端。IE 的值取反和中断控制期间的中断信号进行与操作生成输入 CPU 的中断请求信号。

中断控制器的中断信号处理方法上面已经说过了，其输出的中断号是用来查找中断服务程序地址的。使用 RARS 汇编器汇编带中断的排序程序，可以查找到中断程序入口地址，如图 1.11 所示，使用多路选择器选择应该输出的中断服务程序地址，地址应送入 PC，所以通过一个由 Addrout 控制的三态门输入内总线。电路图如图 1.12 所示。



| Label | Address ▲ |
|----------------------|------------|
| Sort-5-int-riscv.asm | |
| sort_init | 0x00400000 |
| sort_loop | 0x0040006c |
| sort_next | 0x00400084 |
| sort_next1 | 0x00400090 |
| ProgramEnd | 0x004000a0 |
| IntProgram1 | 0x004000a4 |
| IntProgram2 | 0x004000ec |

图 1.11 排序程序汇编后各标签地址

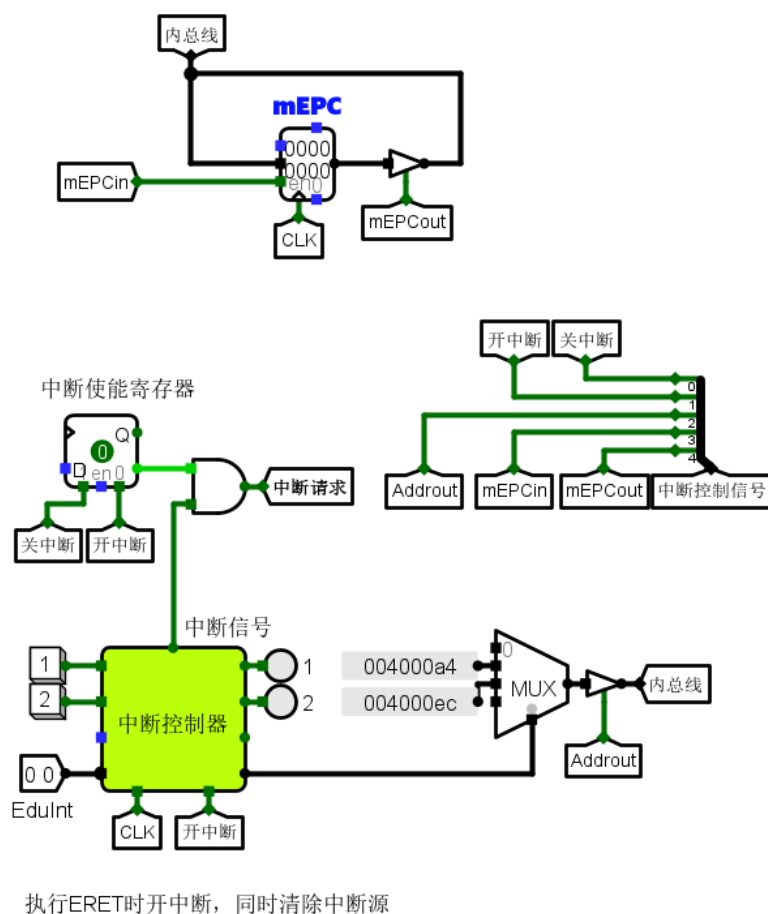


图 1.12 中断相关硬件模块电路图

6. 支持中断的现代时序硬布线控制器状态机设计

根据有限状态机的状态转换条件，填写 Excel 的状态转换表，如图 1.13 所示。表的 ERET 为 1 那一行之前的部分，不看 IR 那一列，就是不带中断的现代时序状态转换表。为了支持中断，需要加上几个状态，其实就对应微程序中的第 25，26，27 号微程序，25 号状态就是 ERET 需要给出信号的状态，26 号状态就中断响应的第一个状态，所有指令执行完后如果没有 IR 信号则直接回取指周期第一个状态，否则转到 26 号状态。需要注意的是这里将 beq 指令判断 equal 信号为 0 时同样算作了最后一条指令，故 15 号状态在 equal 为 0 时同样要进行有无中断信号的判断。

7. 支持中断的硬布线控制器

寄存器的左端为次态，右端为现态，现态和指令信号与判断信号输入状态机生成次态。因为之前设计的控制寄存器中的微程序和硬布线状态刚好一一对应，所以为了方便起见就用控制寄存器代替硬布线控制器中的组合逻辑。电路图如图 1.14。

| 当前状态(现态) | | | | | | 输入信号 | | | | | | | | 下一状态(次态) | | | | | |
|----------|----|----|----|----|------------|------|----|-----|-----|------|------|----|-------|------------|----|----|----|----|----|
| S4 | S3 | S2 | S1 | S0 | 现态 10进制 | LW | SW | BEQ | SLT | ADDI | ERET | IR | EQUAL | 次态 10进制 | N4 | N3 | N2 | N1 | N0 |
| 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | 2 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 2 | | | | | | | | | 3 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 3 | 1 | | | | | | | | 4 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 3 | | 1 | | | | | | | 9 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 1 | 3 | | | 1 | | | | | | 14 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 3 | | | | 1 | | | | | 19 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 3 | | | | | 1 | | | | 22 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 4 | | | | | | | | | 5 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 5 | | | | | | | | | 6 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 6 | | | | | | | | | 7 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 7 | | | | | | | | | 8 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 8 | | | | | | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 9 | | | | | | | | | 10 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 10 | | | | | | | | | 11 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 11 | | | | | | | | | 12 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 12 | | | | | | | | | 13 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 13 | | | | | | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 14 | | | | | | | | | 15 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 15 | | | | | | | | 1 | 16 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 15 | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 16 | | | | | | | | | 17 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 17 | | | | | | | | | 18 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 18 | | | | | | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 19 | | | | | | | | | 20 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 20 | | | | | | | | | 21 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 21 | | | | | | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 22 | | | | | | | | | 23 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 23 | | | | | | | | | 24 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 24 | | | | | | | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 3 | | | | | | 1 | | | 25 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 8 | | | | | | | 1 | | 26 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 13 | | | | | | | 1 | | 26 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 15 | | | | | | | 1 | 0 | 26 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 18 | | | | | | | 1 | | 26 | 1 | 1 | 0 | 1 | 0 |

图 1.13 现代时序硬布线时序产生器状态转换表

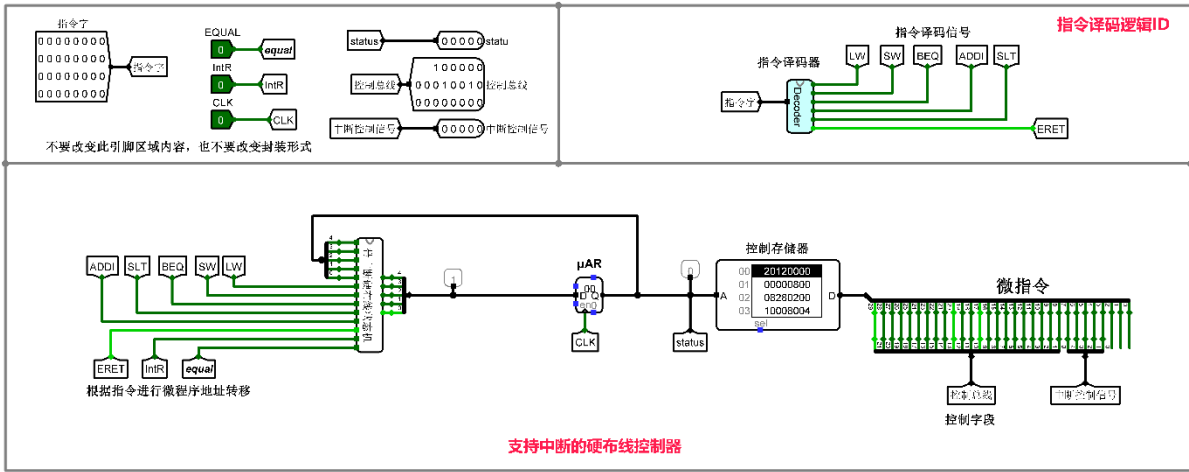


图 1.14 现代时序硬布线控制器电路图

1.3 实验步骤

- (1) 根据相关逻辑填写相关 Excel 表格，生成逻辑表达式。
- (2) 利用生成的逻辑表达式，使用 logisim 的分析组合逻辑电路功能生成那些较为复杂的逻辑电路，然后正确连接各部件，完成实验指定功能。
- (3) 在 educoder 平台上提交测试。
- (4) 运行冒泡排序算法进行联调测试，看 CPU 能否正确工作。

1.4 故障与调试

上面方案设计中给出的所有思路都是排除故障后的最终方案，本节给出的是实验过程中遇到的问题，由于撰写报告时实验已经完成并通过测试，所以没有复现这些错误，仅仅是对原因和解决方案做一些记录。

1.4.1 指令问题

故障现象：三级时序中 beq 指令计算周期第二个节拍给出的信号错误。

原因分析：一开始多给了一个 slt 信号，我以为要减一下再判断 equal，但是查看 ALU 电路发现 equal 不需要给 ALU 操作信号。

解决方案：去掉 beq 指令计算周期第二个节拍给出的 slt 信号。

1.4.2 时序问题

故障现象：在测试三级时序硬布线控制器时输出信号的值都正确但是顺序混乱。

原因分析：一开始没有处理好寄存器触发方式，在 CPU 的总体架构中可以看到寄存器都是在上跳沿锁存新值的，而发生错误时状态机中的状态寄存器也是上跳沿改变状态，所以造成了时序问题。

解决方案：将硬布线控制器中的状态寄存器触发方式改为下跳沿。

1.4.3 条件判别测试问题

故障现象：在现代时序微程序控制器的条件判别测试逻辑中，P1 为 1 时输出的值有误。

原因分析：P1 为 1 时出现的错误说明 beq 指令判断 equal 是否为 1 时出错。仔细分析 educoder 平台上的测试用例，发现其实 educoder 上考虑的情况很简单，在 15 号微指令仅根据 equal 信号决定时跳转到 beq 分支还是取指的第一条微程序，而我最初的想法是在 equal 为 0 时还要考虑中断，当有中断信号来到时进入中断处理程序。

解决方案：去掉 P1 为 1，equal 为 0 时对中断的判断。

1.4.4 微程序执行顺序问题

故障现象：执行 beq 指令时程序跳转错误。

原因分析：一开始我直接将下址地址连到加法器输出，但是由于在中断中使用计数器法计算下址字段，而实验中 beq 指令的第二个微程序下址字段默认为 0，所以在这样一个微指令中下址字段并不是直觉上的加 1，因此要对下址字段进行处理。

解决方案：我的解决方案是加一个多路选择器处理下址字段，当 P1 信号为 0 时正常输出加 1 的地址作为下址字段，P1 信号为 1 时输出 0 作为下址字段。还有一种方法是修改条件判别测试逻辑让 P1 为 1 且 equal 为 0 时跳转到 beq 分支，这里的 beq 分支是 0，P1 为 1 且 equal 为 1 时跳转到下址字段，这时下址字段是正常的加 1，相当于改变了第 15 条微程序下址字段的定义。我没有选择第二种方法的原因是无法通过条件判别逻辑的 educoder 测试，显然老师是不想让我们这么干的。

1.4.5 微程序的中断问题

故障现象：执行 beq 指令时若 equal 不为 0，当有中断信号时不会去处理中断。

原因分析：这其实是 1.4.3 问题带来的连锁问题，因为 equal 不为 0 且有中断时按道理就是要去处理中断的，然而按照我们的条件判别测试逻辑，只有 P2 为 1 时才会响应中断。

解决方案：我给出的解决方案是在条件判别逻辑模块，在原本应该输入 P2 的引脚改为输入一个 end 信号，end 信号由 P2 信号以及 P1 和 equal 取非逻辑与得到的信号进行逻辑或得到。还有一个解决方案是修改 15 号微程序，让它给出 P1 信号的同时给出一个 P2 信号，然后修改条件判别测试逻辑。我没选这个解决方案是因为之前的条件判别测试逻辑已经通过了，而且比较简单，按照这个思路的话条件判别测试逻辑会复杂一些。

华中科技大学课程实验报告

1.4.6 中断硬件问题

故障现象：无法触发中断

原因分析：IE 连接错误，没有理解为什么 IE 异步置 0 端是开中断，因为寄存器初始值是 0，所以寄存器值为 0 时才是开中断，而我却将 IE 的值去和中断信号逻辑与生成中断请求。

解决方案：将 IE 的值取非后再和中断信号逻辑与生成中断请求。

1.5 测试与分析

Educoder 在线评测全部通过。变长指令周期三级时序 CPU 本地联调结果如图 1.15，可以看到 0x80 开始的六个数字降序排序。支持中断的现代时序 CPU 在没有中断的情况下联调结果如图 1.16 所示，同样的从 0x80 开始的数据降序排序，可见两个 CPU 都可以正确运行排序程序。

```
000 fff00413 000004932084a02300140413 004484932084a0230014041300448493
006 2084a02300140413004484932084a023 00140413004484932084a02300140413
010 004484932084a0230014041300448493 2084a02300140413004484932084a023
016 0000041301c00493200429832004aa03 0149a2b3000286632134a02321442023
020 ffc48493 00940463fe0000e3 00440413 01c0049300940463fc0008e3 00000063
028 000000000000000000000000000000 00000000000000000000000000000000
030 000000000000000000000000000000 00000000000000000000000000000000
038 000000000000000000000000000000 00000000000000000000000000000000
040 000000000000000000000000000000 00000000000000000000000000000000
048 000000000000000000000000000000 00000000000000000000000000000000
050 000000000000000000000000000000 00000000000000000000000000000000
058 000000000000000000000000000000 00000000000000000000000000000000
060 000000000000000000000000000000 00000000000000000000000000000000
068 000000000000000000000000000000 00000000000000000000000000000000
070 000000000000000000000000000000 00000000000000000000000000000000
078 000000000000000000000000000000 00000000000000000000000000000000
080 000000600000005000000040000003 00000020000000100000000 fffffff
```

图 1.15：三级时序 CPU 联调结果

```
000 fff00113 fff00413 000004932084a023 00140413004484932084a02300140413
006 004484932084a0230014041300448493 2084a02300140413004484932084a023
010 00140413004484932084a02300140413 004484932084a0230014041300448493
016 2084a0230000041301c0049320042983 2004aa030149a2b3000286632134a023
020 21442023 ffc48493 00940463fe0000e3 0044041301c0049300940463fc0008e3
028 00000063008101130081202300912223 240004930004a403001404130084a023
030 0084a2230084a4230084a6230084a823 0084aa230084ac230084ae2300412483
038 00012403 ff810113 0020007300810113 0081202300912223280004930004a403
040 fff40413 0084a0230084a2230084a423 0084a6230084a8230084aa230084ac23
048 0084ae230041248300012403 ff810113 00200073000000000000000000000000
050 000000000000000000000000000000 00000000000000000000000000000000
058 000000000000000000000000000000 00000000000000000000000000000000
060 000000000000000000000000000000 00000000000000000000000000000000
068 000000000000000000000000000000 00000000000000000000000000000000
070 000000000000000000000000000000 00000000000000000000000000000000
078 000000000000000000000000000000 00000000000000000000000000000000
080 000000600000005000000040000003 00000020000000100000000 fffffff
```

图 1.16：现代时序 CPU 无中断联调结果

华中科技大学课程实验报告

支持中断的现代时序 CPU 在有一次按键 1 中断的情况下的联调结果如图 1.17，可以看到 0x90 开始的八个字单元全部加 1。有一次按键 2 中断的情况下联调结果如图 1.18 所示，可以看到 0xa0 开始的八个单元字全部减 1。再次按下 3 次 1 号中断和 5 次二号中断，由于中断执行过程中关中断，所以连续快速按按键是没有用的，要等一次中断处理完后再按下按键，运行结果如图 1.19 所示，1 号中断加了 3，二号中断减了 5，可见中断可以正常运行。

```
080 00000006000000050000000400000003 000000020000000100000000 ffffffff
088 00000000000000000000000000000000 00000000000000000000000000000000
090 00000001000000010000000100000001 00000001000000010000000100000001
098 00000000000000000000000000000000 00000000000000000000000000000000
0a0 00000000000000000000000000000000 00000000000000000000000000000000
```

图 1.17 现代时序 CPU 运行一次 1 号中断结果

```
080 00000006000000050000000400000003 000000020000000100000000 ffffffff
088 00000000000000000000000000000000 00000000000000000000000000000000
090 00000001000000010000000100000001 00000001000000010000000100000001
098 00000000000000000000000000000000 00000000000000000000000000000000
0a0 ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
```

图 1.18 现代时序 CPU 运行一次 2 号中断结果

```
080 00000006000000050000000400000003 000000020000000100000000 ffffffff
088 00000000000000000000000000000000 00000000000000000000000000000000
090 00000004000000040000000400000004 00000004000000040000000400000004
098 00000000000000000000000000000000 00000000000000000000000000000000
0a0 fffffffa fffffffa fffffffa fffffffa fffffffa fffffffa fffffffa fffffffa
```

图 1.19 现代时序 CPU 运行四次 1 号中断和六次二号中断结果

2 总结与心得

2.1 实验总结

本次实验主要完成了如下几点工作：

- 1) 实现了支持五条 RISC-V 指令的三级时序变长指令周期硬布线控制器。
- 2) 实现了一个可以进行冒泡排序的三级时序变长指令周期 CPU
- 3) 实现了支持五条 RISC-V 指令以及中断返回指令的现代时序微程序控制器。
- 4) 实现了支持五条 RISC-V 指令以及中断返回指令的现代时序硬布线控制器。
- 5) 实现了中断相关硬件模块的设计。
- 6) 实现了一个可以进行冒泡排序以及中断响应的现代时序 CPU

2.2 实验心得

- 1) 熟悉了 logisim 的各个部件，尤其是译码器和多路选择器的使用，学会了使用工程化方法自动生成组合逻辑电路，以及常用的 debug 方法。
- 2) 熟悉了 RISC-V 五条核心指令在 CPU 中的执行流程，每一个时钟节拍应该做什么，并通过这五条指令的实现了解了单总线 CPU 一般指令执行流程的分析方法，核心思想是内总线上始终只能有一个寄存器在输出数据。
- 3) 熟悉了三级时序硬布线控制器的设计方法。硬布线控制器设计分为时序发生器的设计和用于输出信号的组合逻辑单元的设计，组合逻辑单元根据机器周期、节拍电位和指令信号给出操作信号，而时序发生器负责产生机器周期电位和节拍电位。时序发生器可以分为用来改变状态的有限状态机和一个负责将状态信号处理为机器周期和节拍电位信号的输出函数。如果为了方便设计，有限状态机可以仅按照时钟周期改变状态，这就意味着所有指令的机器周期数和机器周期的节拍电位数一致，这样就是定长指令周期；为了提高效率，有限状态机还可以根据指令类型改变状态，这样的话不同指令的机器周期数和节拍数就可以不同，这样就是变长指令周期。
- 4) 熟悉了现代时序的控制器的设计方法，包括硬布线和微程序。现代时序设计主要就是设计一个有限状态机，该状态机的次态由输入信号和现态共同决定，

华中科技大学课程实验报告

每一个指令执行的不同节拍都有一个状态，根据该状态给出输出。按照这个思想设计状态机，然后设计一个组合电路根据状态信号输出控制信号，这样就是现代时序硬布线控制器；如果将状态和输出抽象为一条条微指令，状态的转移路径就是微指令的执行顺序，这样就是现代时序微程序控制器，设计微程序控制器的主要任务是设计微程序给出的信号以及地址转移逻辑。

- 5) 了解了 CPU 处理中断的方式以及单总线 CPU 上简单的支持单级中断的硬件如何设计。处理单级中断主要是几个步骤：关中断，保存断点地址，判断中断类型并送对应中断服务程序地址送 PC，中断处理完后断点地址送回 PC，开中断。开关中断对应的硬件是中断使能寄存器，存取断点地址对应的硬件是异常程序地址计数器，判断中断类型取中断服务程序地址对应的硬件是中断控制器，将功能与硬件相对照，连接上相应的控制信号并将输入输出连接至总线即可。
- 6) 在实验中对于课堂上讲授的 CPU 知识有了更深刻的感受，虽然在课堂和书本上学习了很多东西，但是实际动手做了之后还是很有不一样的感受，比如填写 Excel 表的时候，一个信号一个信号对照着填的感受和在课本上单纯的看是绝对不一样的。
- 7) 最后总结下来，整个计算机组成原理实验确实让我掌握了一些硬知识，首先是让我对计算机的硬件组成有了一个总体的认识，同时也和上个学期的数字逻辑的课程呼应，让我了解了如何使用数字逻辑的知识去设计冯诺依曼体系结构计算机中的运算器、存储器和控制器，尤其是本次 CPU 设计的实验，让我知道了同步时序逻辑电路的真正作用，而在数字逻辑课程中同步时序逻辑电路部分都只是在做题，实验涉及的基本是组合逻辑电路设计和各种计数器的设计，其难点都是在于异步进位的处理上，对同步时序逻辑电路缺乏真正实体上的认识。希望以后的数字逻辑课程的实验设计上可以和组成原理课程衔接更加紧密一些。最后在实验的过程中遇到了不少的问题，不过在组原群里面的大萝卜老师在我们的实验的过程中对于我们提出的问题都给出了很及时和详细的解答，同时群里面也有很多同学们在热心的回答我们的问题，很感谢大家的帮助。

参考文献

- [1] DAVID A. PATTERSON(美). 计算机组成与设计硬件/软件接口(原书第 5 版). 北京:机械工业出版社.
- [2] David Money Harris(美). 数字设计和计算机体系结构(第二版). 机械工业出版社
- [3] 谭志虎, 秦磊华, 吴非, 肖亮. 计算机组成原理. 北京:人民邮电出版社, 2021 年.
- [4] 谭志虎, 秦磊华, 胡迪青. 计算机组成原理实践教程. 北京:清华大学出版社, 2018 年.

• 指导教师评定意见 •

一、原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字：郑舟



二、对课程实验的学术评语（教师填写）

三、对课程实验的评分（教师填写）

| 评分项目 (分值) | 课程目标 1 工具应用 (10 分) | 课程目标 2 设计实现 (70 分) | 课程目标 3 验收与报告 (20 分) | 最终评定 (100 分) |
|--------------|--------------------------|--------------------------|---------------------------|-----------------|
| 得分 | | | | |

指导教师签字：_____