中国科学院大学网络空间安全学院

计算机组成与结构研讨课

实验报告

实验序号：5　　实验名称：单总线 CPU 实验

1　实验电路图

1.1　MIPS 指令译码器设计

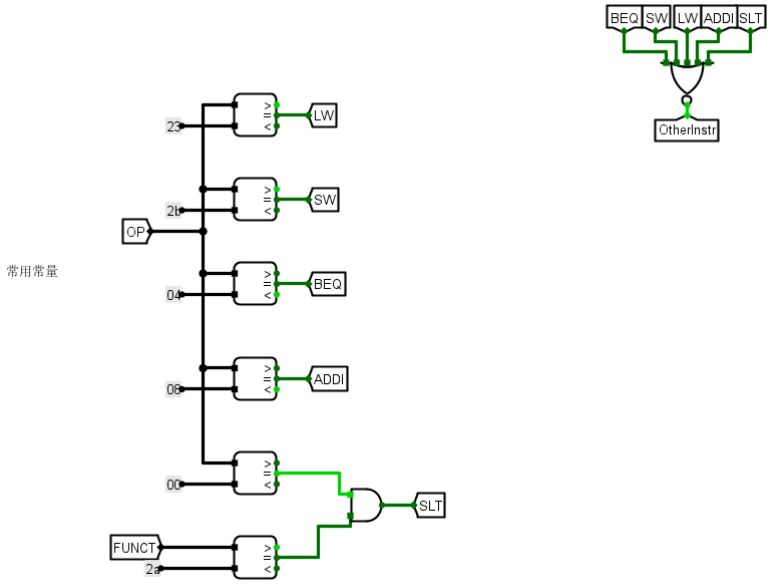

图 1: MIPS 指令译码器设计

设计思路：

输入：

操作码（Opcode），这是指令中的一部分，通常为 6 位，用来指定执行的操作类型（例如：加载、存储、算术操作等）。

功能码（Funct），可能用于 R 型指令，指示具体的操作，例如加法、减法、逻辑运算等。

译码过程：

译码器通过解码操作码（例如：BEQ, SW, LW, ADDI, SLT 等）来判断当前指令类型。

根据操作码，译码器决定下一步应该执行的操作（如数据传输、算术运算、跳转等），并生成相应的控制信号，供后续电路使用。

控制信号输出：

各个控制信号例如 ALU 操作控制信号、内存控制信号、寄存器选择信号等都由译码器产生，这些信号会被送往 ALU、寄存器堆、内存模块等。

具体逻辑：

根据操作码的不同组合，译码器可以通过与门、或门等逻辑电路处理出每种操作所需的控制信号。

1.2 变长指令周期—硬布线控制器设计

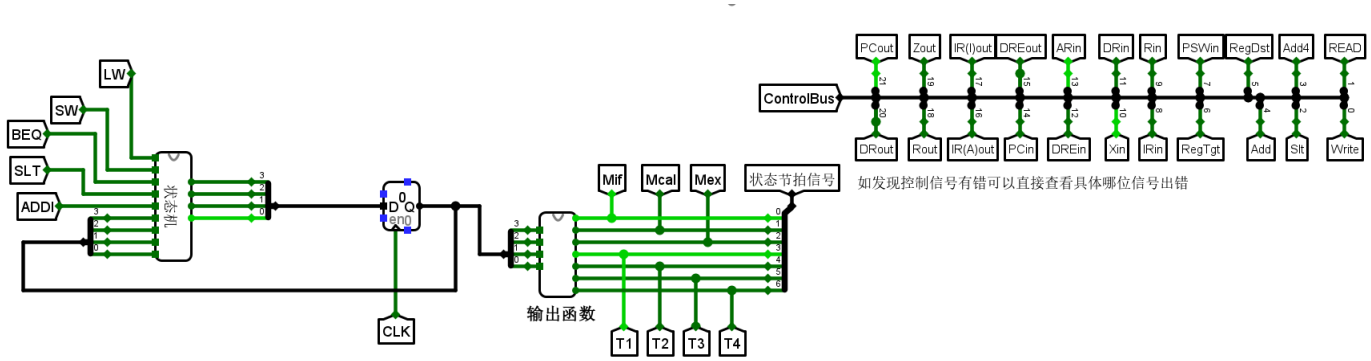

图 2: 变长指令周期—硬布线控制器设计

设计思路： 输入：

该设计中的输入包括来自指令译码器的控制信号以及时钟信号（CLK）。

控制器接收到译码器输出的各类信号（如 BEQ, LW, SW 等）后，确定指令的类型并开始控制执行过程。

信号生成：

控制器生成一系列信号（如写使能信号、读使能信号、ALU 控制信号等）来控制 CPU 的各个部件，例如：

PC 控制信号：控制程序计数器是否更新（例如用于跳转指令）。

寄存器操作信号：控制寄存器堆的读写。

内存操作信号：控制内存的读写操作。

ALU 操作信号：决定 ALU 执行的具体运算（加法、减法等）。

变长周期控制：

在变长指令周期中，硬布线控制器需要根据不同的指令类型生成对应的信号，这些信号会影响指令执行的不同阶段（例如取指、解码、执行、访存等）。

控制信号会根据指令的不同长度或复杂度进行调整，某些指令可能需要多个时钟周期来完成（例如 LOAD 或 STORE 操作），因此控制信号会在这些时钟周期内不断更新。

具体逻辑：

通过使用 D 触发器来存储状态，每个状态对应一个执行阶段。每个时钟周期后，硬布线控制器根据当前状态和输入信号生成新的控制信号，指引 CPU 执行下一步操作。

控制器会根据状态机的设计来调节各个模块的信号（例如 ALU 输入、寄存器堆读写、内存访问等）。

2 实验中遇到的问题

1. **状态寄存器触发方式设置错误**

   在变长指令周期的硬布线控制器设计中，出现了电路图连接方式无误但仍无法通过测试的情况。经检查后发现，状态寄存机默认的触发方式为上升沿触发，而在改电路设计中需要将触发方式改为下降沿触发，修改后成功通过测试。

2. **硬布线控制器组合逻辑单元设计繁琐易出错**

   硬布线控制器组合逻辑单元由 excel 填表生成逻辑表达式，但电路输出多达二十余个，填表并得出逻辑表达式的过程异常繁琐且极容易出错，需要仔细核对，最终花费了大量时间才完成实验。

3 实验心得

在设计变长指令周期的 MIPS 单总线 CPU 的过程中，核心挑战在于硬布线控制器的实现。

硬布线控制器的组合逻辑设计过程极其繁琐。由于控制信号众多，采用 Excel 表格生成逻辑表达式时，稍有不慎就会引入错误。尽管该方法能系统化地整理真值表，但面对二十多个输出信号时，人工核对的工作量巨大，且容易遗漏细节。最终，通过逐步验证和分段调试，才确保所有控制信号正确生成。这一过程让我认识到，logisim 在电路设计上仍存在极大的弊端。

总的来说，本次实验让我对 CPU 控制器的设计有了更深入的理解，尤其是在时序约束和组合逻辑优化方面积累了实践经验。未来若涉及类似设计，我会优先考虑更高效的自动化方法，同时更加注重时序分析的严谨性，以减少调试时间并提高电路可靠性。