***2024***



**逻辑与计算机系统设计 ·实验报告·**

j0242087[1]

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： |  |
| 学 号： |  |
| 姓 名： | [作者] |
| 电 话： |  |
| 邮 件： |  |
| 完成日期： |  |



[1 运动码表系统设计 2](#_Toc202003983)

[1.1 设计要求 2](#_Toc202003984)

[1.2 方案设计 2](#_Toc202003985)

[1.3 实验步骤 10](#_Toc202003986)

[1.4 故障与调试 10](#_Toc202003987)

[1.5 测试与分析 11](#_Toc202003988)

[1.6 实验总结 11](#_Toc202003989)

[1.7 实验心得 11](#_Toc202003990)

[2 CPU设计实验 13](#_Toc202003991)

[2.1 设计要求 13](#_Toc202003992)

[2.2 方案设计 13](#_Toc202003993)

[2.3 实验步骤 21](#_Toc202003994)

[2.4 故障与调试 21](#_Toc202003995)

[2.5 测试与分析 22](#_Toc202003996)

[2.6 实验总结 22](#_Toc202003997)

[2.7 实验心得 22](#_Toc202003998)

# 运动码表系统设计

## 设计要求

构建一个小型数字系统：运动码表。包含了组合逻辑电路设计，同步时序电路设计，寄存器数据传输等内。运动码表功能如下表 1‑1 运动码表功能

表 ‑ 运动码表功能

|  |  |
| --- | --- |
| 按键 | 功能 |
| Start | 计时器归零，重新开始计时 |
| Stop | 停止计时，显示计时数据 |
| Store | 尝试更新系统记录，并显示系统记录 |
| Reset | 复位，计时=00.00，系统记录=99.99 |

## 方案设计

### 2路选择器设计

2路选择器（1位）的原理较为简单，可以得知输出Out=X1 Sel + X0 ~Sel，利用Logisim的电路分析功能即可自动实现电路，如图 1‑1 2路选择器所示

16位的2路选择器只需将1位的简单并联即可。

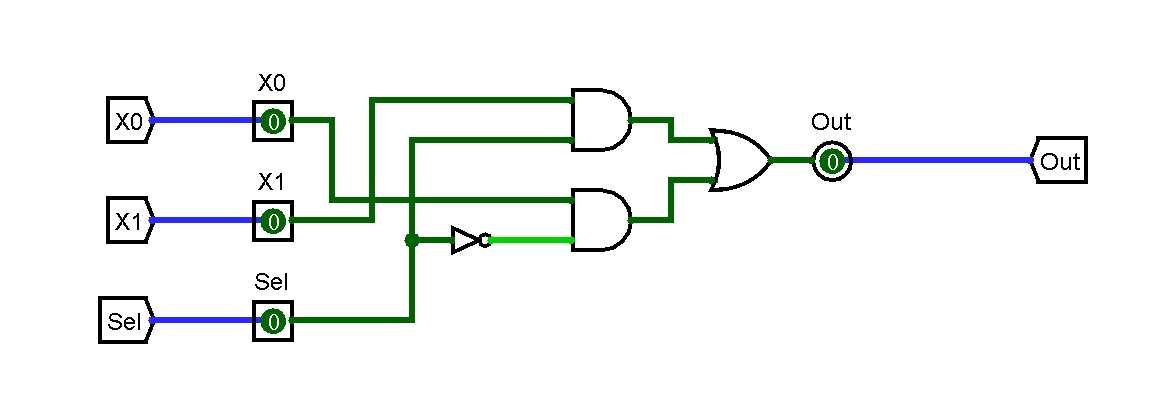


图 1‑1 2路选择器

### 无符号比较器设计

4位无符号比较器的逻辑函数表达式可以直接写出，利用电路分析功能生成电路。

Great= X3 ~Y3 + ~(X3 ^ Y3) X2 ~Y2 + ~(X3 ^ Y3) ~(X2 ^ Y2) X1 ~Y1 + ~(X3 ^ Y3) ~(X2 ^ Y2) ~(X1 ^ Y1) X0 ~Y0

Equal= ~(X3 ^ Y3) ~(X2 ^ Y2) ~(X1 ^ Y1) ~(X0 ^ Y0)

Less= ~X3 Y3 + ~(X3 ^ Y3) ~X2 Y2 + ~(X3 ^ Y3) ~(X2 ^ Y2) ~X1 Y1 + ~(X3 ^ Y3) ~(X2 ^ Y2) ~(X1 ^ Y1) ~X0 Y0

16位无符号比较器需要使用4个4位无符号比较器，将4个4位无符号比较器共12个输出视为输入，写出16位无符号比较器的输出与它们的关系，自动生成电路，然后将12个输入替换为4个4位无符号比较器的输出即可，如图 1‑2 16位无符号比较器。

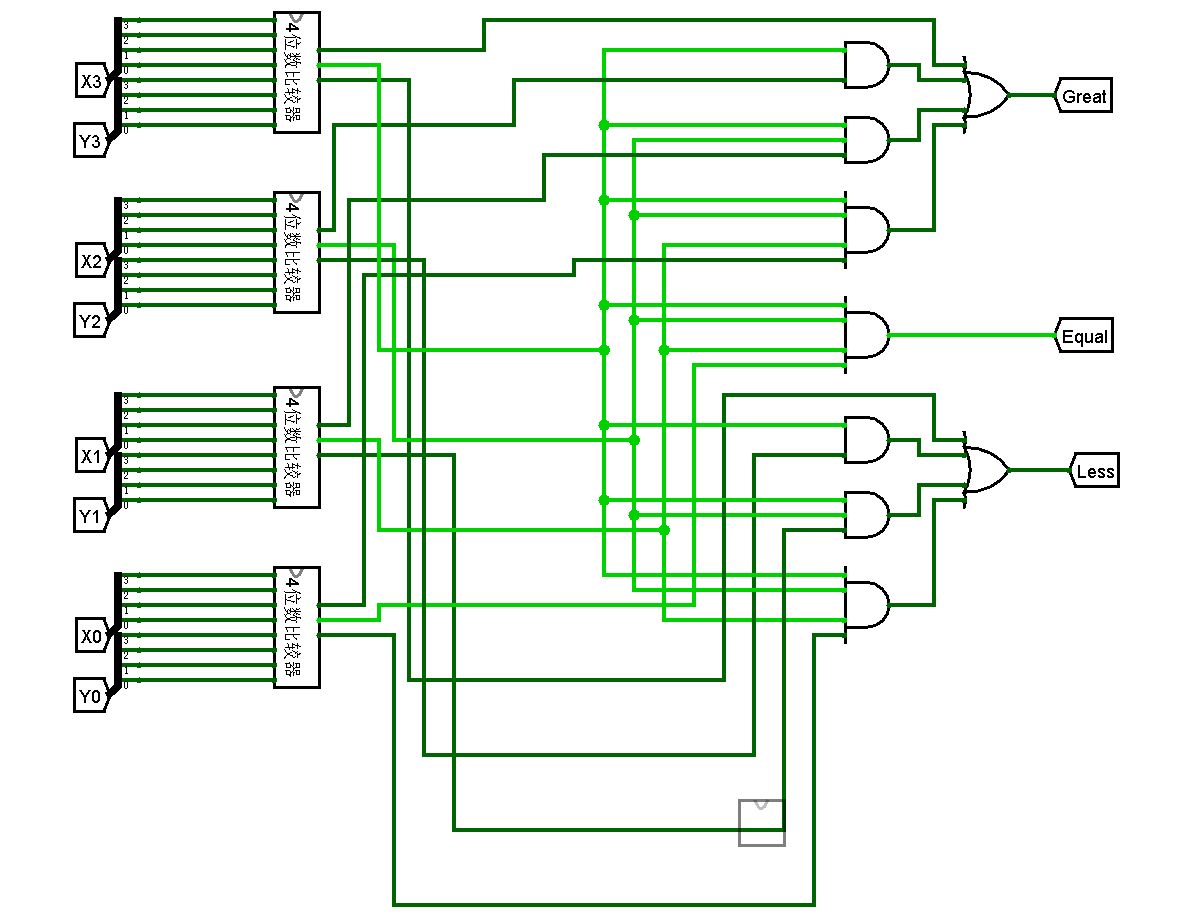


图 1‑2 16位无符号比较器

### 码表数码管驱动

利用真值表构建电路即可，如图 1‑3 码表数码管驱动。

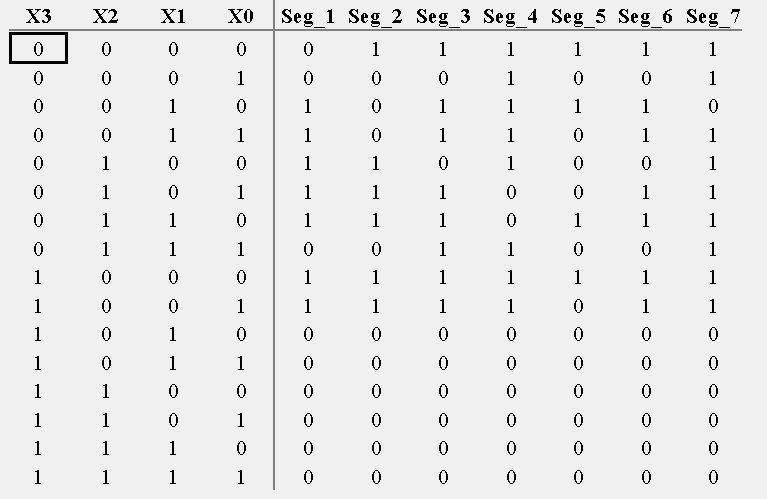


图 1‑3 码表数码管驱动

### 并行加载寄存器

4位并行加载寄存器只需要将Din和Q的4个bit分别与4个D寄存器相连，CLK和En与所有寄存器相连即可。16位并行加载寄存器相当于将4位并行加载寄存器中的4个寄存器的位置替换为4位并行加载寄存器。

### BCD计数器

BCD计数器是一个Moore型的同步时序逻辑电路，功能是完成0到9的循环计数。输出非常简单，Q输出即为当前的状态，Cout当计数为9时输出1。状态转换也很简单，0-转换到1，1转换到2……9转换到0，而由于D寄存器的特性，下一状态的BCD码值就是状态转换部分的输出，状态转换真值表如图 1‑4 BCD计数器状态转换。

BCD计数器实现如图 1‑5 BCD计数器。

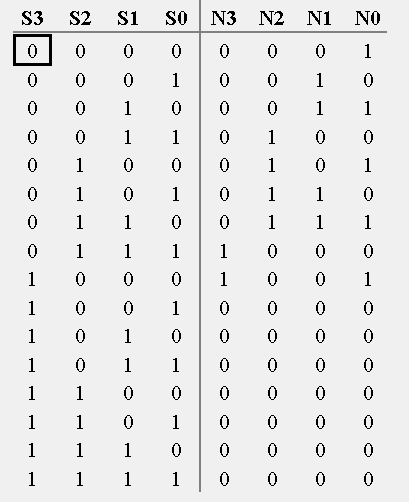


图 1‑4 BCD计数器状态转换

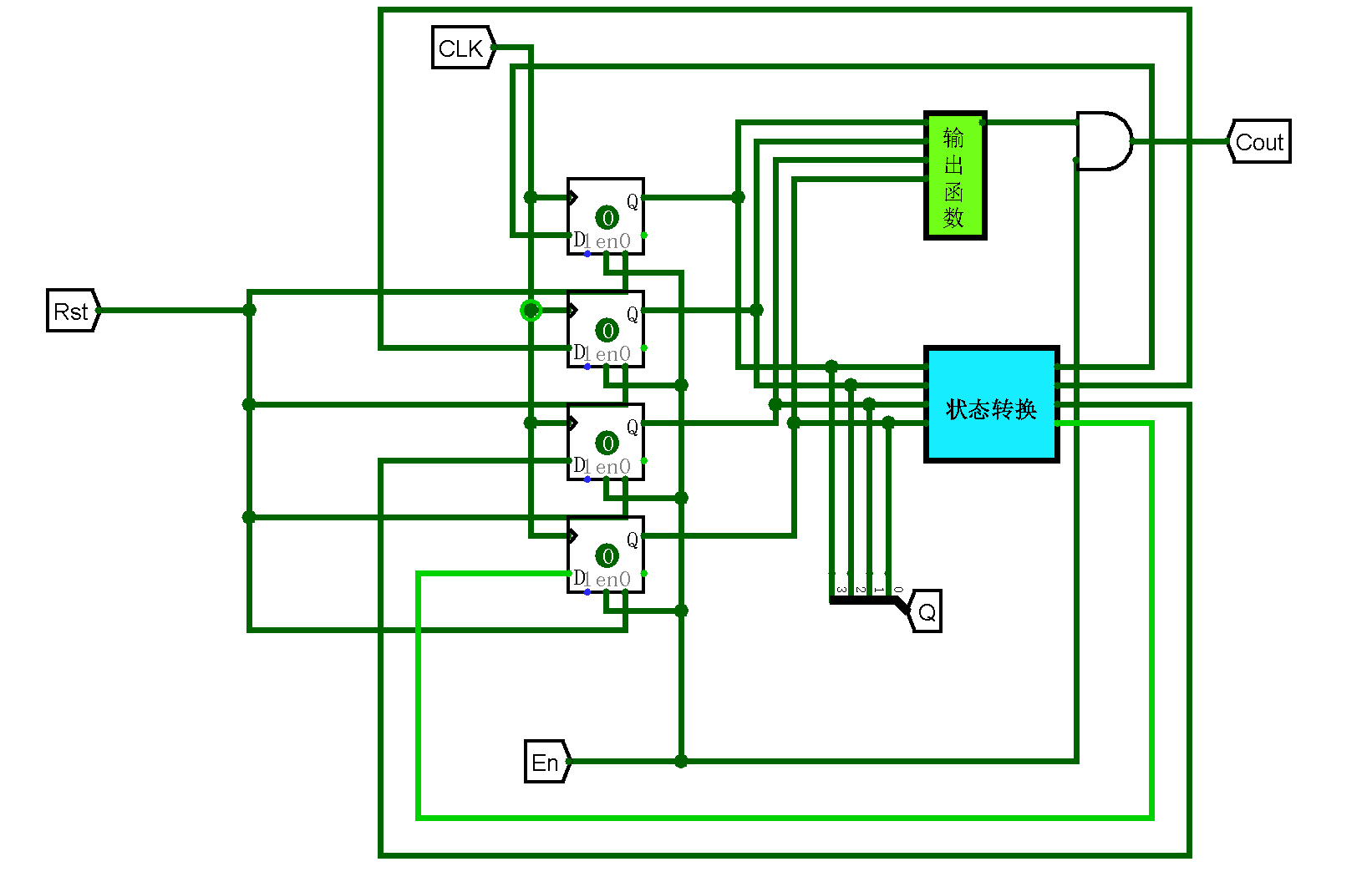


图 ‑ BCD计数器

### 码表计数器

使用4个BCD计数器构建码表计数器，分别代表10s，1s，1/10s，1/100s，16位输出。只需要将低位的计数器的Cout信号连接到高位的使能端即可完成低位计数到9时相邻高位在时钟到来时加1的进位功能。实现如图 1‑6 码表计数器。

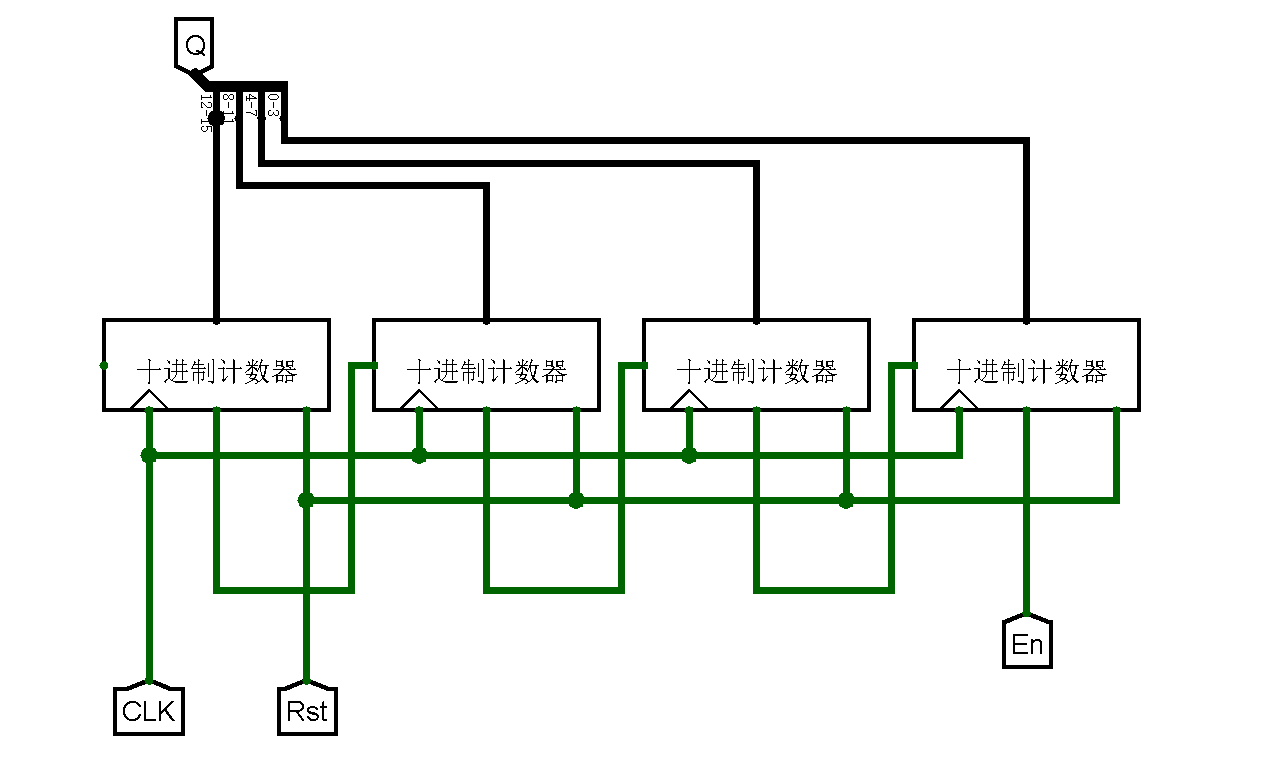


图 1‑6 码表计数器·

### 码表控制器

至此码表的大部分元件都已经设计出来了，下面需要将各个部件连接起来，这就需要构建码表控制器。码表控制器是一个同步时序逻辑电路，输入是时钟信号和外部的控制信号（Start，Stop，Store，Reset），输出是码表内部所需的各种控制信号。内部控制信号可以考察各个元件的控制信号和输入来源，如表 1‑2 元件控制信号和输入输出所示。借助图表可以分析码表控制器需要输出TM-En,TM-Rst,SD-En,SD-Sel,DP-Sel这几个控制信号。此外考虑到比较器的输出应当也会使状态机的状态改变，所以NewRecord也一并加入控制器的输入中。

表 1‑2 元件控制信号和输入输出

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能部件 | 控制信号 | 输入 | 输出 | 输入来源 | 输入控制 |
| 时间计数器TM | TM-En,TM-Rst | CLK | TM.Q |  |  |
| 16位寄存器SD | SD-En | CLK,Din | SD.Q | 99.99或当前记录 | SD-Sel |
| 数码管显示DP |  | Din | DisplayInfo | TM.Q或SD.Q | DP-Sel |
| 比较器 |  | 当前记录和SD.Q | NewRecord |  |  |

明确了控制器的输入输出之后，就需要根据码表的功能设计状态转移图（图 1‑7 状态转换图），之后借助同步时序电路状态转换表生成状态转换逻辑表达式（图 1‑8 状态转换逻辑）和输出函数逻辑表达式（图 1‑9 输出函数逻辑）。最后将6种状态所需的3个寄存器、输入输出、状态转换逻辑元件和输出函数逻辑元件连接即可（图 1‑10 码表控制器）。

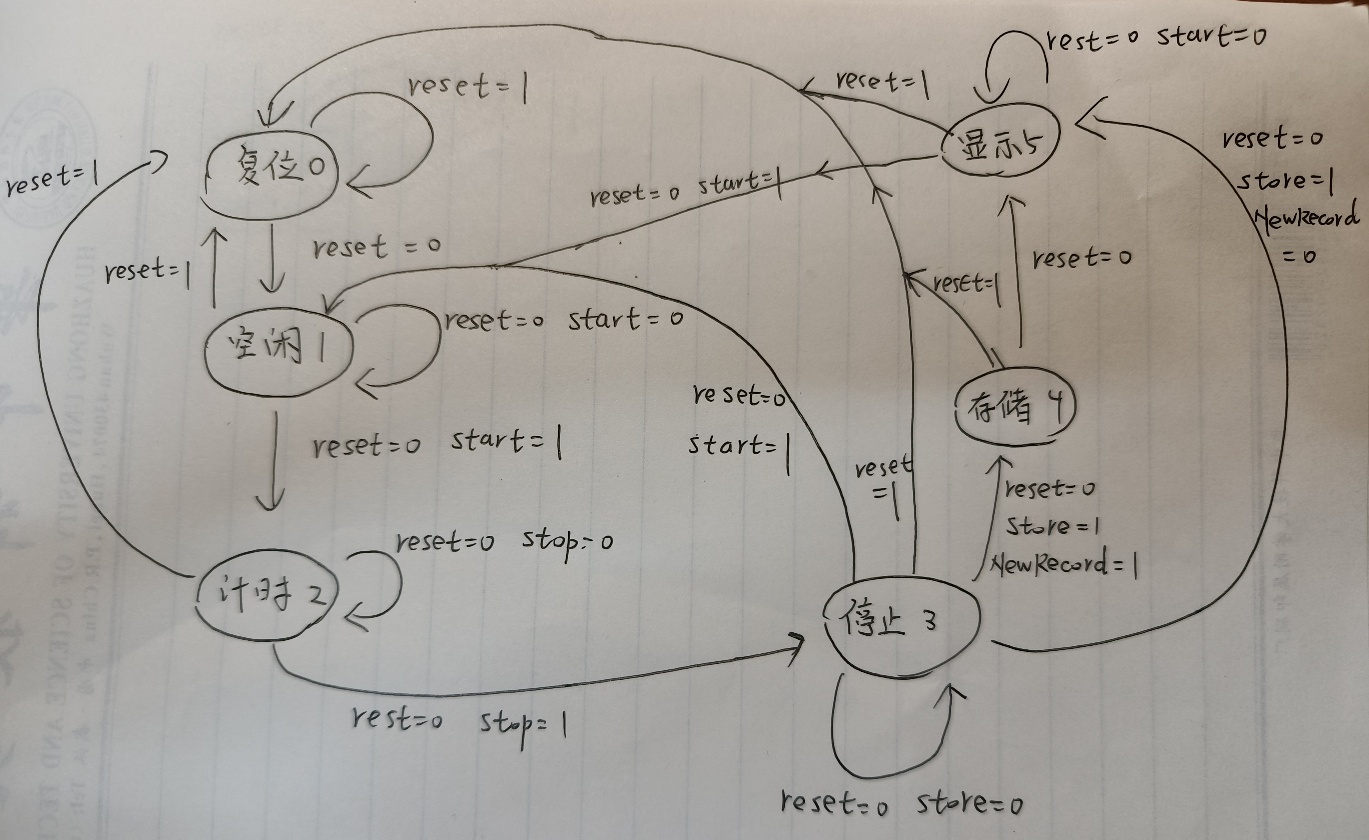


图 1‑7 状态转换图

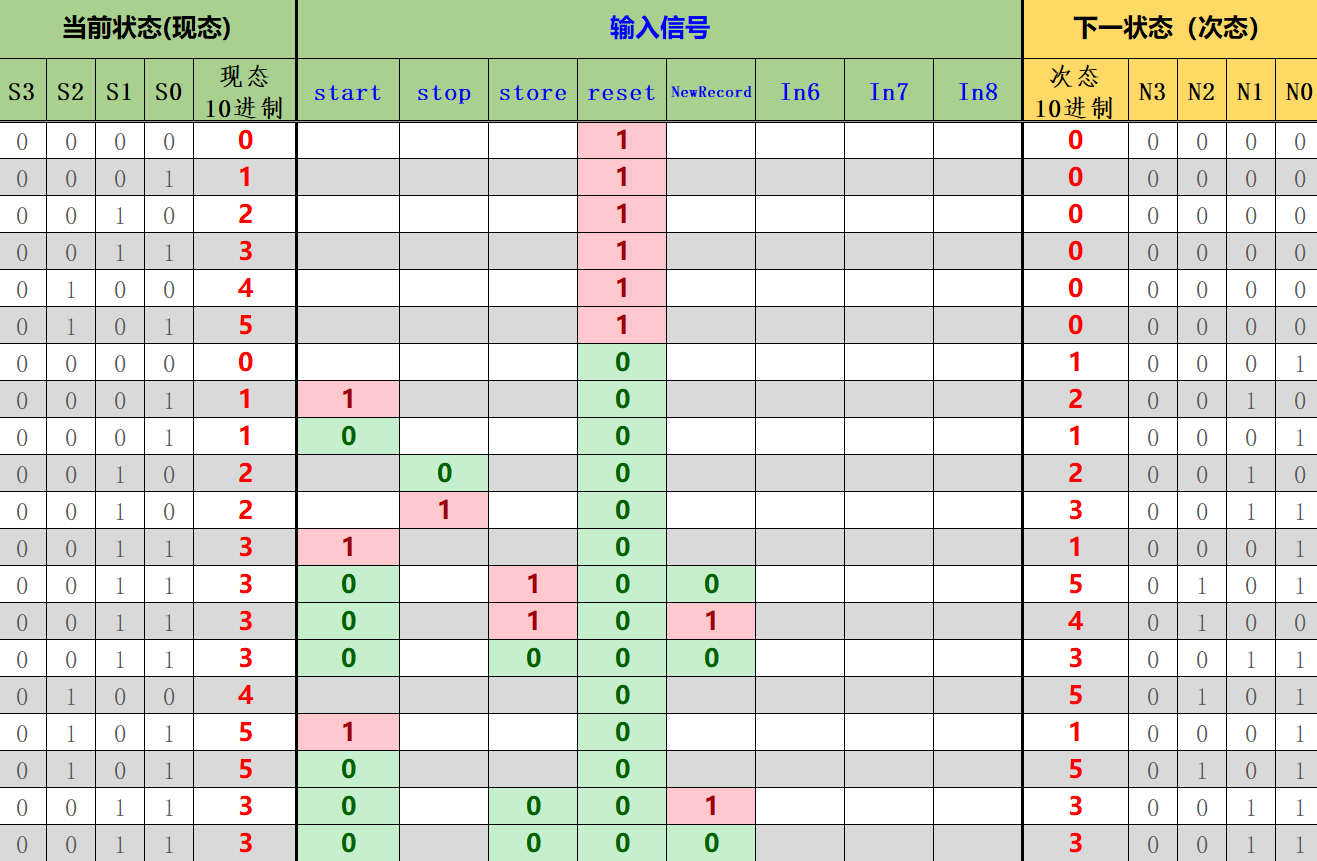


图 1‑8 状态转换逻辑

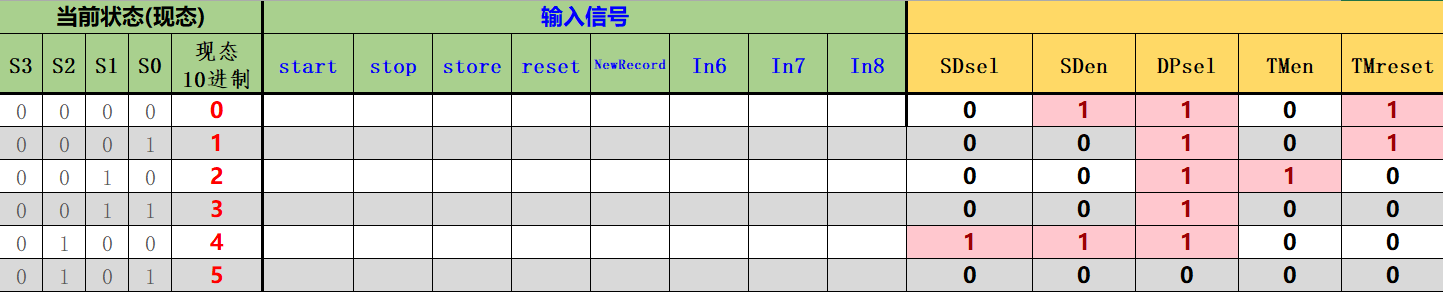


图 ‑ 输出函数逻辑

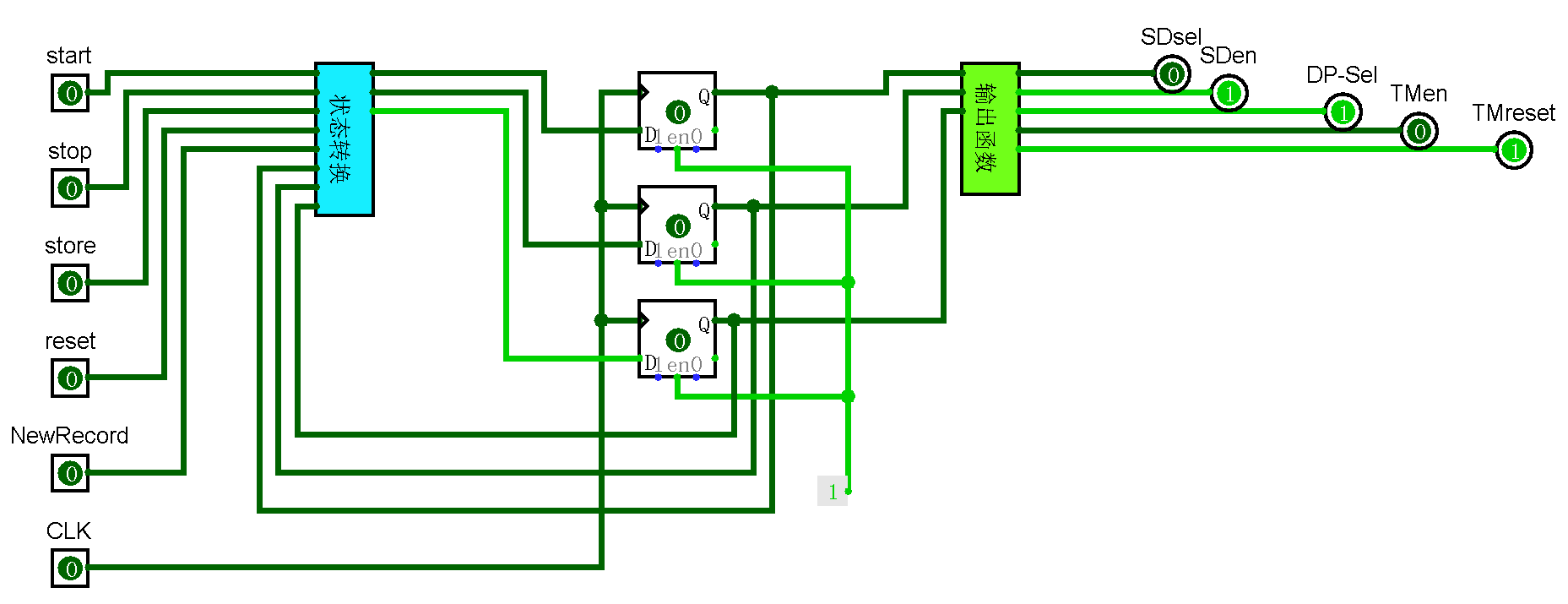


图 1‑10 码表控制器

### 系统集成

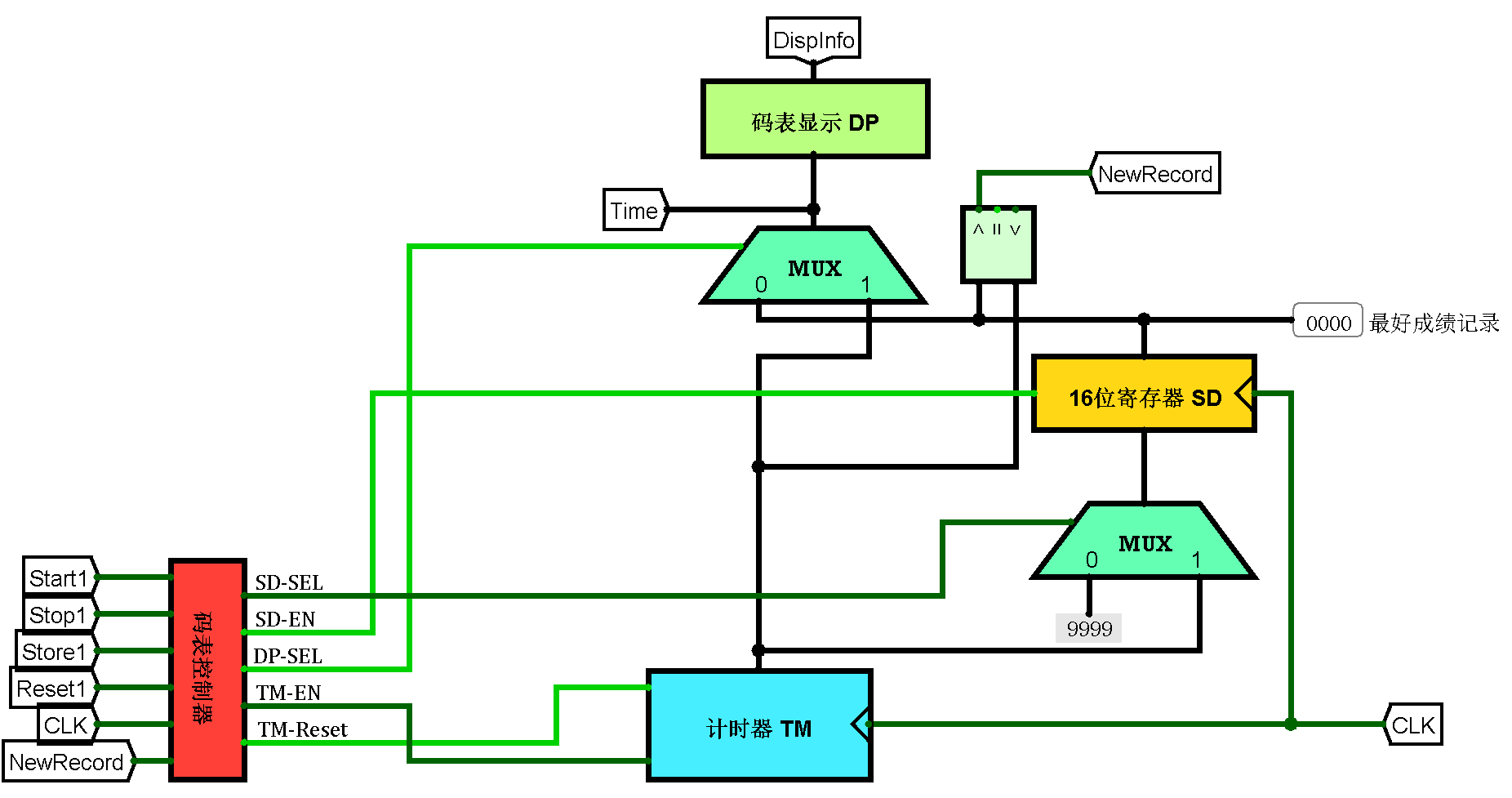
根据各个元件的输入输出把它们按照逻辑连接在一起即可，如图 1‑11 系统集成所示。

图 1‑11 系统集成

## 实验步骤

1. 直接在Logisim中完成简单部件的设计
2. 借助相关Excel表格或者Logisim电路分析功能生成逻辑表达式
3. 利用逻辑表达式完成复杂部件的设计
4. 设计数据通路，完成系统集成
5. 提交平台测试

## 故障与调试

### 码表计数器计数故障

**故障现象：** 码表计数器在复位信号为1时仍然不会复位，而是继续计数。

**原因分析：** 仔细检查电路发现复位信号输入被修改后导致其在码表计数器封装上的位置改变，测试程序无法找到复位信号输入。

**解决方案：** 改正封装的引脚位置。

### 码表控制器状态机故障

**故障现象：** 如图 1‑12 1.4.2 码表控制器状态机故障所示。

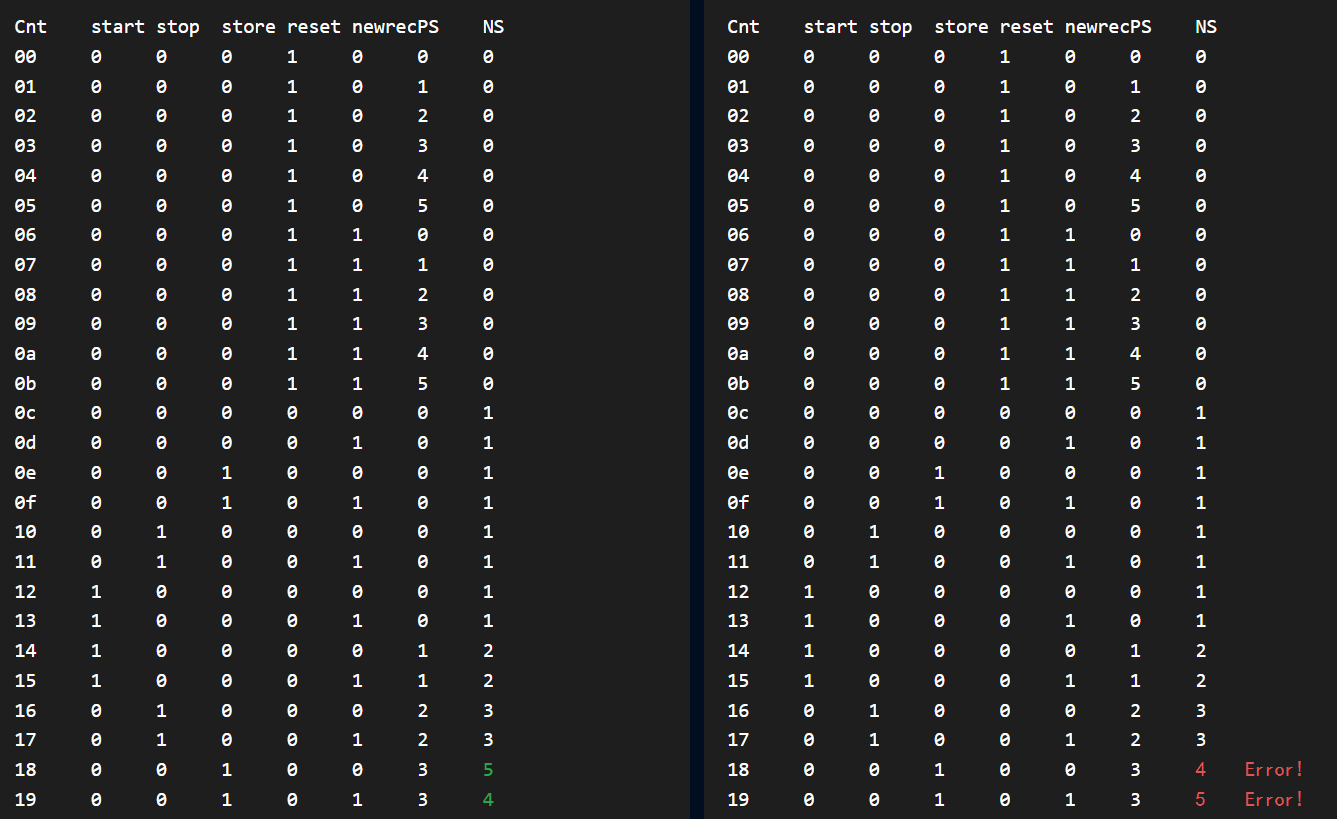


图 1‑12 1.4.2 码表控制器状态机故障

**原因分析：** 状态3下，store=1，newrecord=1时应该转到状态4，store=1，newrecord=0时应该转到状态5。

**解决方案：** 修改状态转移逻辑。

### 系统集成故障

**故障现象：** 秒表的计数错误。

**原因分析：** 检查电路，码表显示输入的2路选择器0端连接了计时器，1端连接了数据寄存器，而DP-Sel设计的本意应该是0端连接数据寄存器，1端连接计时器。

**解决方案：** 改正码表的数据通路连接。

## 测试与分析

本地测试未发现异常，实验平台也测试通过，综合来看码表实验成功。

## 实验总结

本次实验主要完成了如下几点工作：

1. 完成了2路选择器，无符号比较器，码表数码管驱动，16位并行加载寄存器，BCD计数器，码表计数器，码表控制器等部件和整个系统的集成。
2. 实现了一个具有计时，暂停，存储和显示最优结果的数字码表。

## 实验心得

1. 在设计过程中，我深刻体会到数字系统的模块化设计思想的重要性。从基本门电路到2路选择器、无符号比较器等基础组件，再到复杂的BCD计数器和状态控制器，每个模块都像精密的齿轮，必须确保单独功能完善才能实现整体协同运作。
2. 本次实验推动我深度掌握了Logisim的一些强大功能，包括电路分析，模拟，子电路封装等。
3. 通过状态机设计（图1-7），我领悟到控制流与数据流的辩证关系：控制器如同系统大脑，其输出的TM-En、SD-Sel等控制信号实质上是数据通路的"决策指令"。当计时器数值与寄存器记录在比较器中输出NewRecord信号时，这种数据驱动状态转移的机制生动展现了冯·诺依曼架构的精髓。
4. 对课设的建议：码表控制器状态机设计的测试基本固定了状态机的设计，但实际上状态机可能有不同的设计方法，课设或许可以增加这方面的可能性。

# CPU设计实验

## 设计要求

1. 绘制MIPS CPU数据通路，实现单周期硬布线控制器，设计实现单周期MIPS CPU。
2. 绘制多周期MIPS CPU数据通路，实现微程序控制器，利用微程序控制器设计实现多周期MIPS CPU。
3. 绘制多周期MIPS CPU数据通路，实现多周期硬布线控制器，利用硬布线控制器设计实现多周期MIPS CPU。

## 方案设计

### 单周期MIPS CPU数据通路设计

根据需要实现的指令（图 2‑1 单周期CPU指令），绘制表 2‑1 单周期CPU指令/部件输入（部分明显只有一种输入的部件输入已省略）。观察得知有4个部件输入有多种输入来源，为此我们需要添加相应的多路选择器和控制信号（PCSrc，RegDst，ALUSrc，MemtoReg）。



图 2‑1 单周期CPU指令

表 2‑1 单周期CPU指令/部件输入

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 指令/部件输入 | PC | R1# | R2# | RW# | SrcA | SrcB | WD |
| add | PC\_Add | rs | rt | rd | R1 | R2 | ALU\_Re |
| slt | PC\_Add | rs | rt | rd | R1 | R2 | ALU\_Re |
| addi | PC\_Add | rs |  | rt | R1 | SignImm | ALU\_Re |
| lw | PC\_Add | rs |  | rt | R1 | SignImm | ReadData |
| sw | PC\_Add | rs | rt |  | R1 | SignImm |  |
| beq | PC\_Branch | rs | rt |  | R1 | R2 |  |
| bne | PC\_Branch | rs | rt |  | R1 | R2 |  |
| 控制信号 | PCSrc |  |  | RegDst |  | ALUSrc | MemtoReg |

### 单周期MIPS CPU控制器设计

控制器的输入是指令，输出是各种控制信号。首先是上文中的RegDst，ALUSrc，MemtoReg，此外还有寄存器堆写使能RegWrite，运算器操作控制符AluOP，数据存储器写使能MemWrite，syscall指令的Halt停机信号。由于PCSrc的值还需要比较运算的结果才能确定，所以我们让控制器输出Beq，Bnq两个控制信号，在控制器之外构建PCSrc信号。

综上，控制信号如图 2‑2 单周期MIPS CPU控制信号所示。



图 2‑2 单周期MIPS CPU控制信号

接下来通过比较器实现指令译码，参考图 2‑2 单周期MIPS CPU控制信号，使用简单的组合逻辑即可完成控制器（图 2‑3 单周期硬布线控制器）。

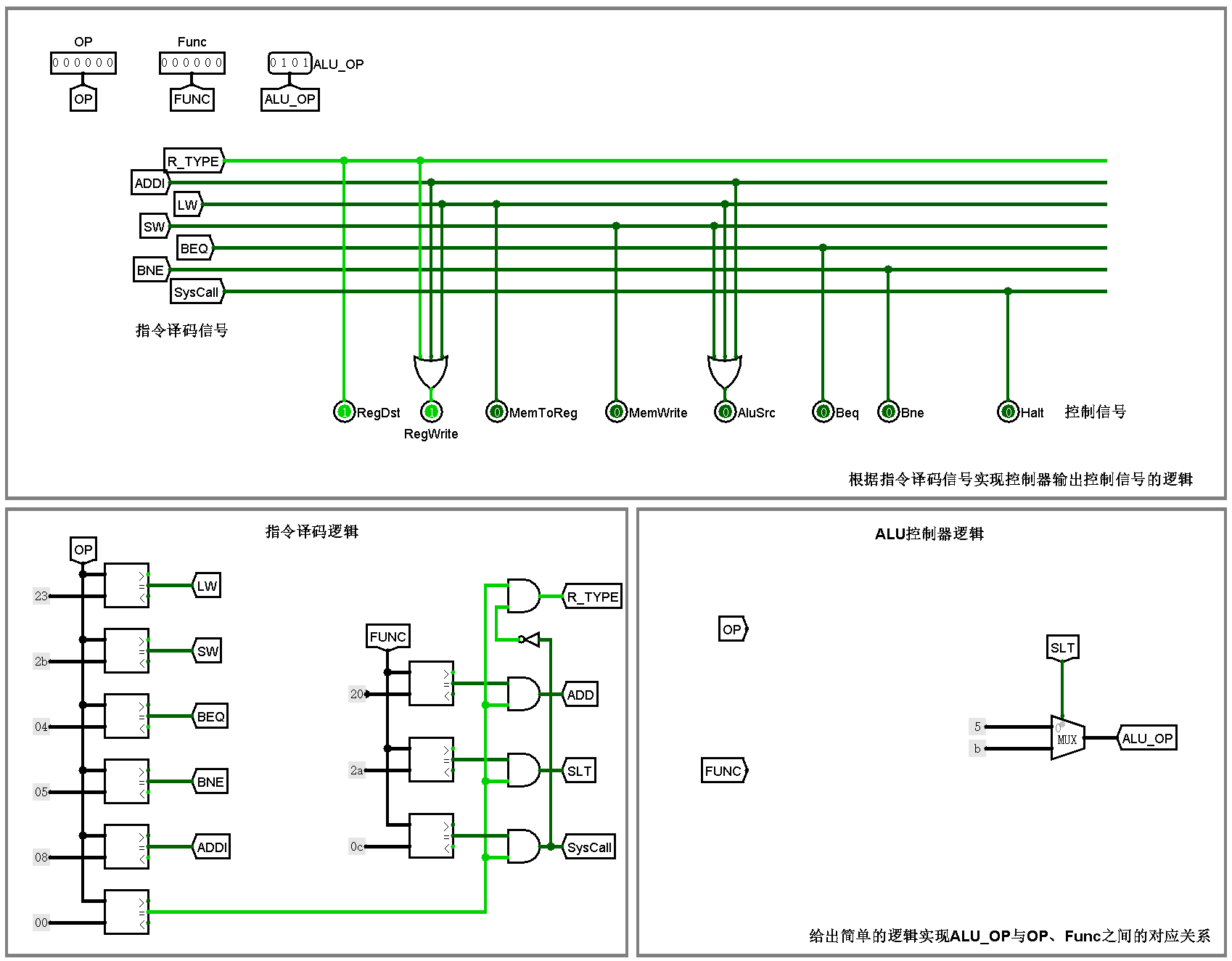


图 2‑3 单周期硬布线控制器

### 单周期MIPS CPU数据通路构建

设计好PCSrc的逻辑，利用控制器输出的控制信号控制其他部件，各部件简单相连即可完成单周期MIPS CPU的设计，如图 2‑4 单周期MIPS CPU。

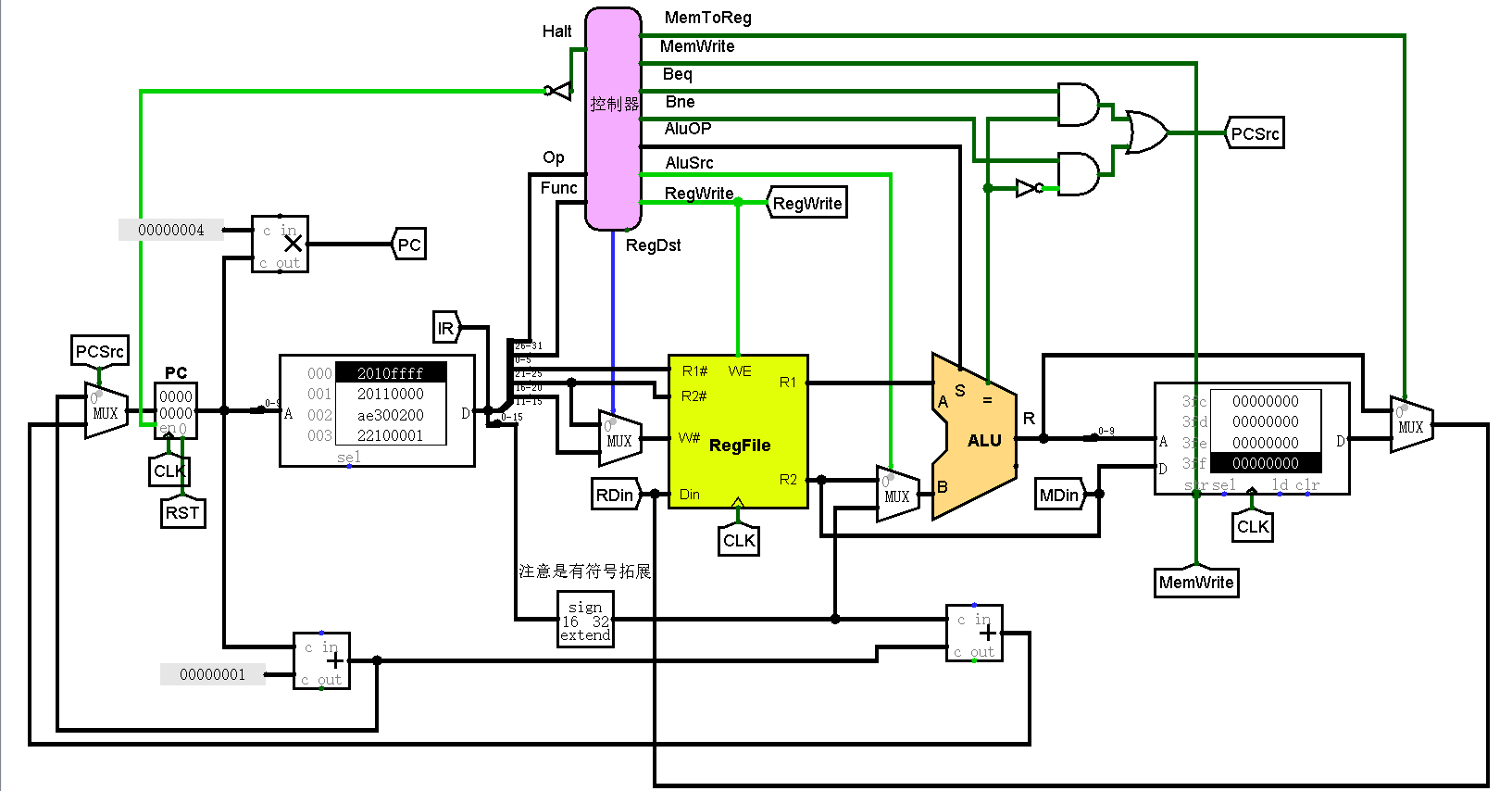


图 2‑4 单周期MIPS CPU

### 微程序多周期MIPS CPU控制信号

数据通路中隐含的多路选择器的控制信号通过指令/部件输入表的方法找出来，结合各个部件的控制信号，得到以下控制信号的集合：图 2‑5 微程序多周期MIPS CPU控制信号。



图 2‑5 微程序多周期MIPS CPU控制信号

### 微程序多周期MIPS CPU控制器设计

不同于单周期CPU，多周期CPU的一条指令在多个时钟周期内完成，我们将一个状态对应一个时钟周期，每个状态的值都作为地址从ROM中读取一条微指令，微指令中包含全部的控制信号，以及下一个状态的值（亦是下一条微指令的地址）。这样我们便设计了一个可以输出控制信号的控制器。状态流程图如图 2‑6 微程序多周期MIPS CPU状态流程图所示。但是我们发现S1状态的下一个状态有多种可能，所以我们需要设计一个微程序地址转移逻辑（图 2‑7 微程序地址转移逻辑），根据指令来判断下一个状态，并在微指令中加入一个P字段用来指示下一个状态的根据是地址转移逻辑还是下址字段。

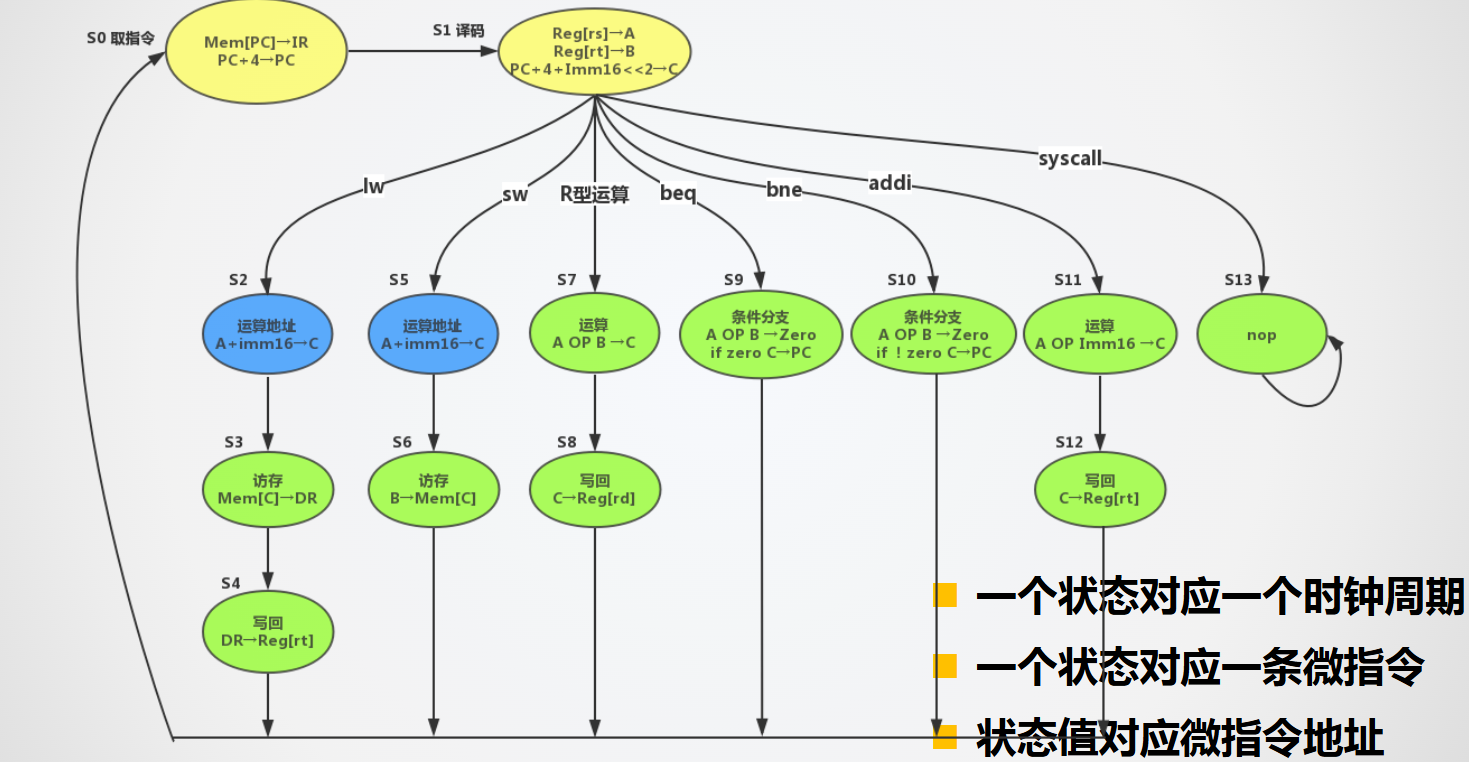


图 2‑6 微程序多周期MIPS CPU状态流程图

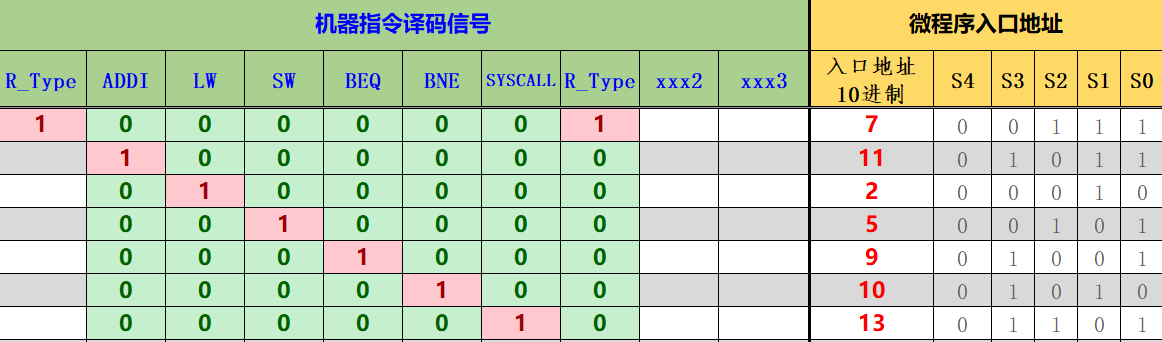


图 ‑ 微程序地址转移逻辑

之后根据控制信号的说明（图 2‑5 微程序多周期MIPS CPU控制信号）和状态流程图（图 2‑6 微程序多周期MIPS CPU状态流程图）来构建微指令并存入ROM中（图 2‑8 微指令）。之后简单连接各部件完成控制器（指令译码和ALU控制与上一关实验类似），如图 2‑9 微程序控制器所示。

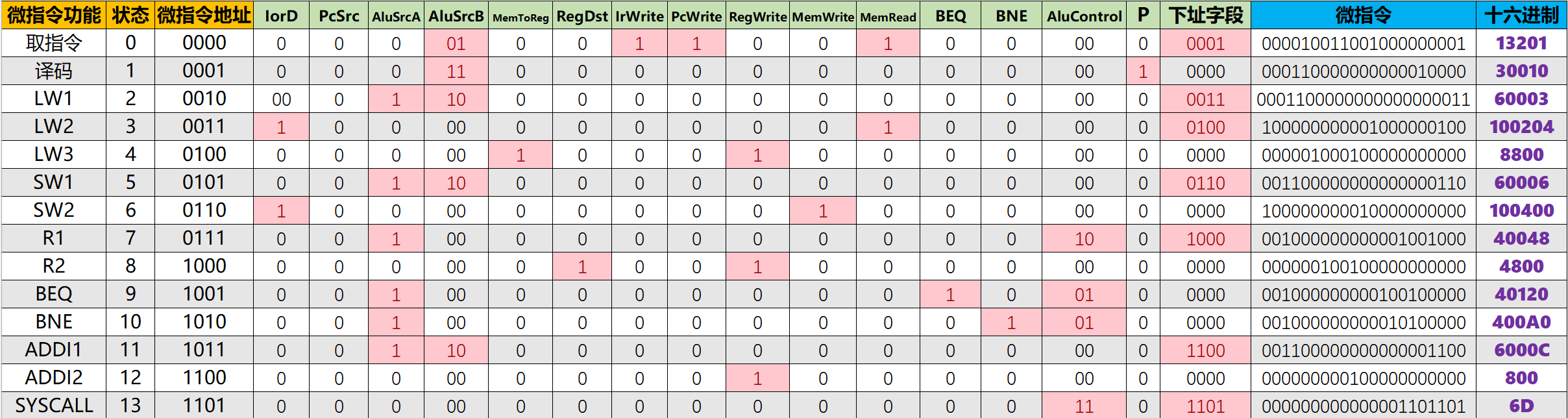


图 2‑8 微指令

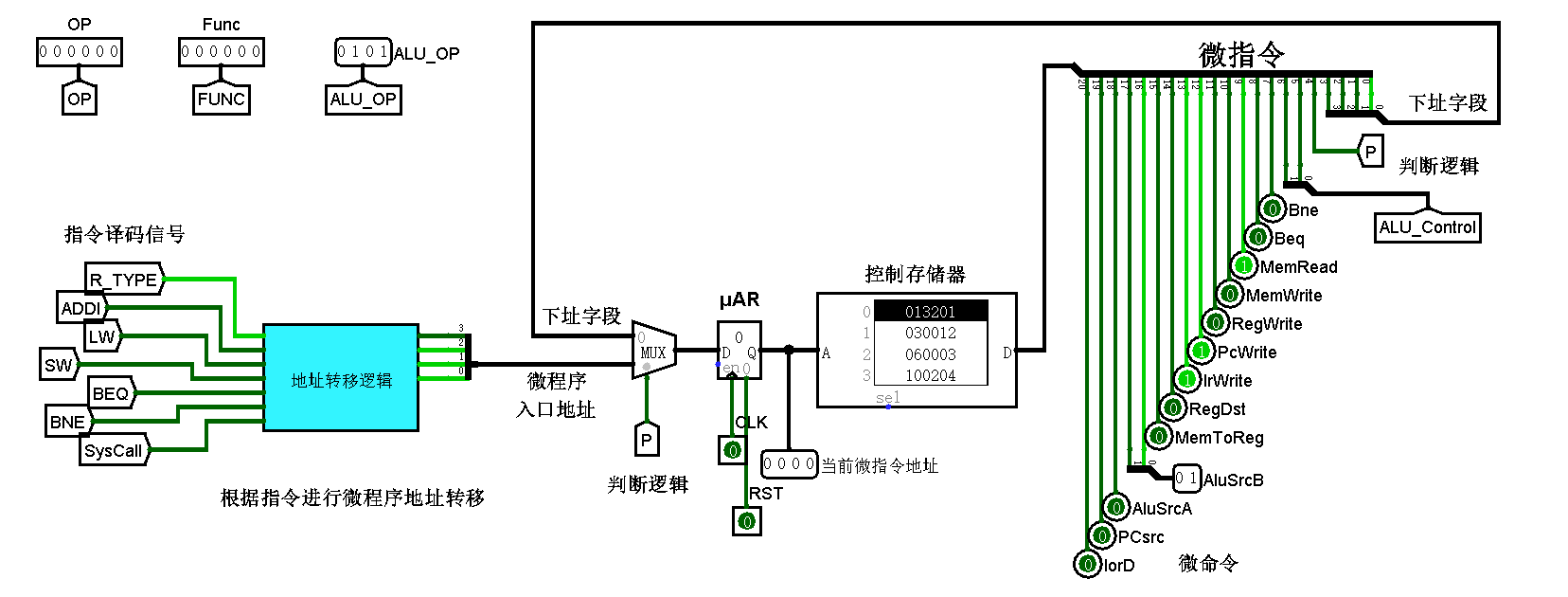


图 ‑ 微程序控制器

### 微程序多周期MIPS CPU数据通路构建

利用控制器输出的控制信号控制其他部件，各部件简单相连即可完成单周期MIPS CPU的设计，如图 2‑10 微程序多周期MIPS CPU数据通路构建所示。

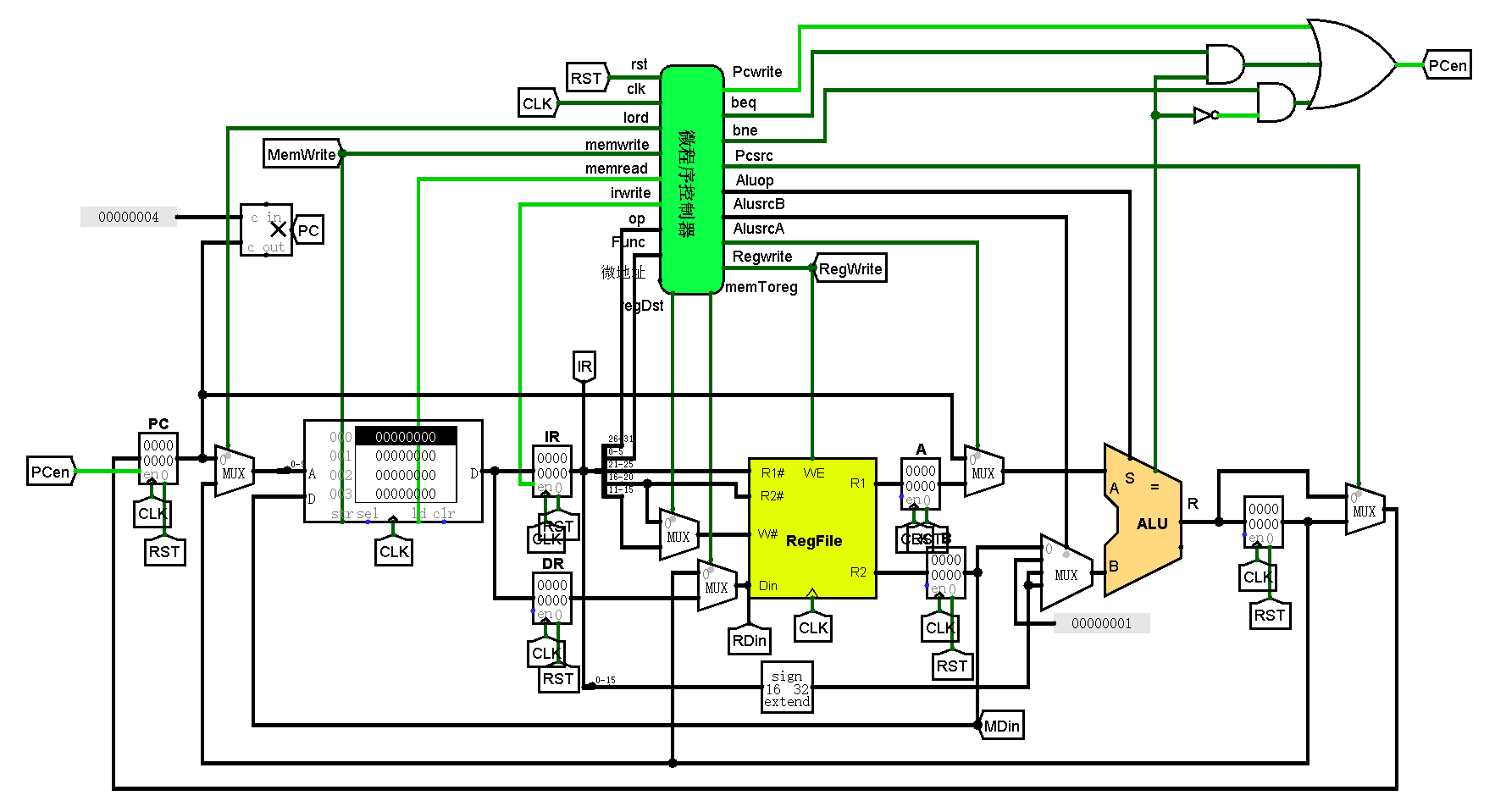


图 2‑10 微程序多周期MIPS CPU数据通路构建

### 硬布线控制器多周期MIPS CPU控制器设计

控制信号（图 2‑5 微程序多周期MIPS CPU控制信号）和状态流程（图 2‑6 微程序多周期MIPS CPU状态流程图）与上一个实验一样，不同之处在于微程序控制器使用微指令中的下址字段和地址转移逻辑实现状态的迁移，而这里使用状态机来实现。

在相关Excel表格中填写状态迁移逻辑（图 2‑11 硬布线控制器状态机），用生成的次态逻辑表达式自动生成状态机FSM子电路。

控制存储器ROM中的控制信号可以照搬微程序控制器中的，将下址字段和P字段弃之不用。指令译码和ALU控制与上一关实验相同。最终实现如图 2‑12 硬布线控制器。

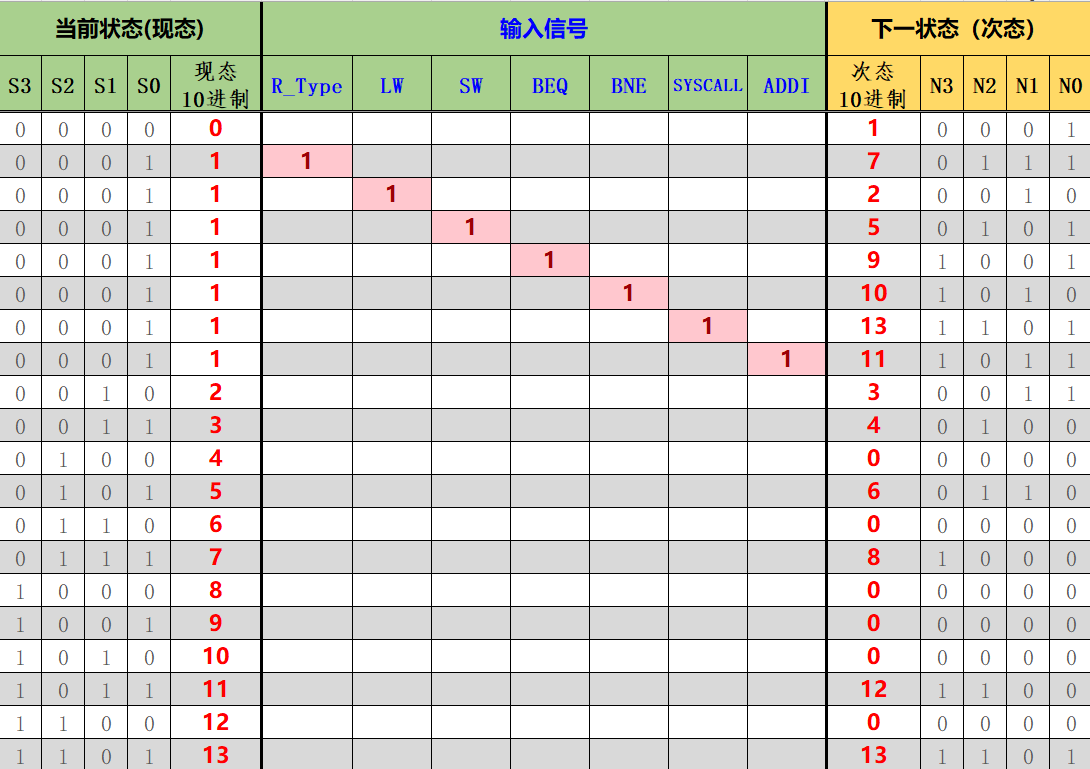


图 2‑11 硬布线控制器状态机

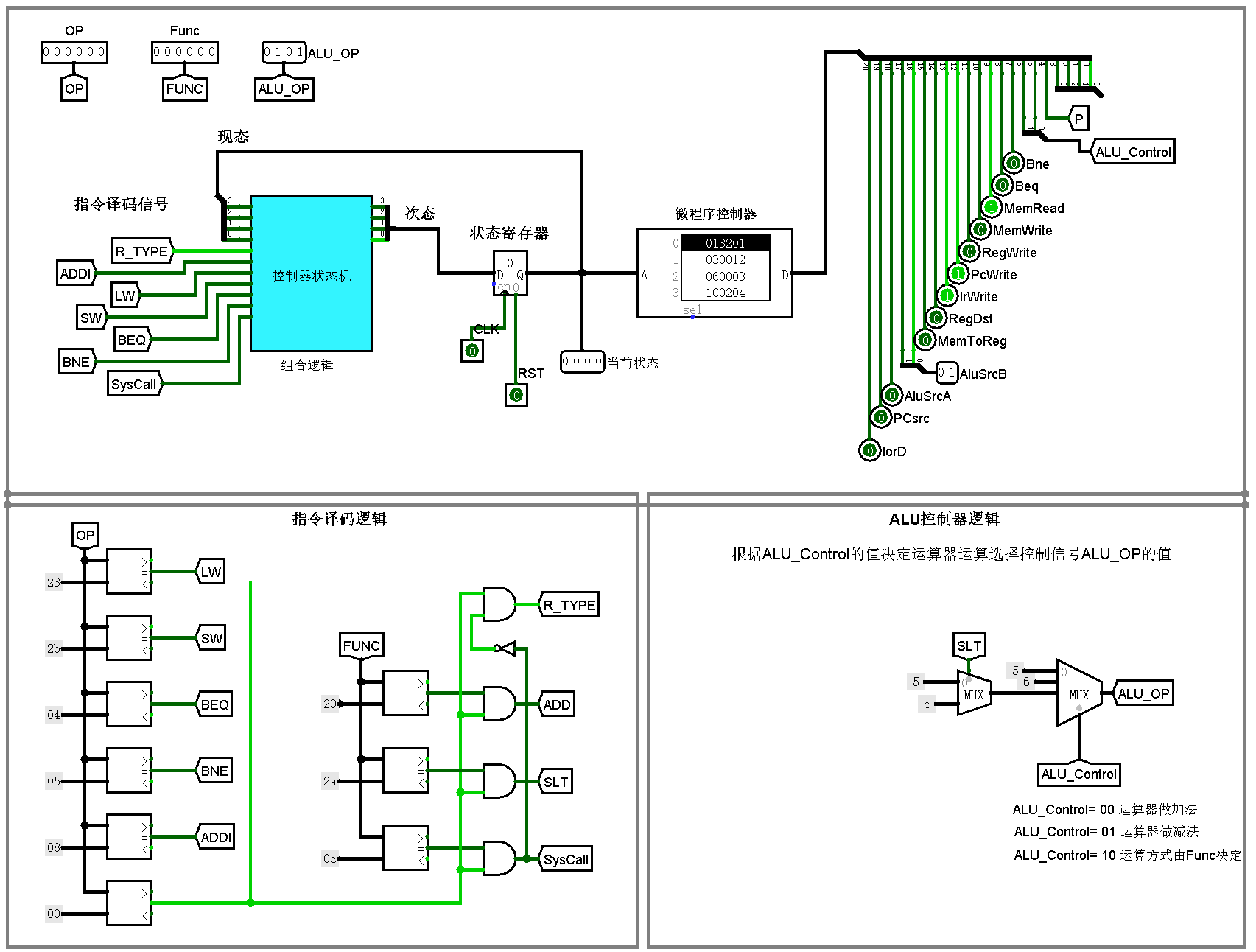


图 ‑ 硬布线控制器

### 硬布线控制器多周期MIPS CPU数据通路构建

利用控制器输出的控制信号控制其他部件，各部件简单相连即可完成单周期MIPS CPU的设计，如图 2‑13 硬布线控制器多周期MIPS CPU所示。

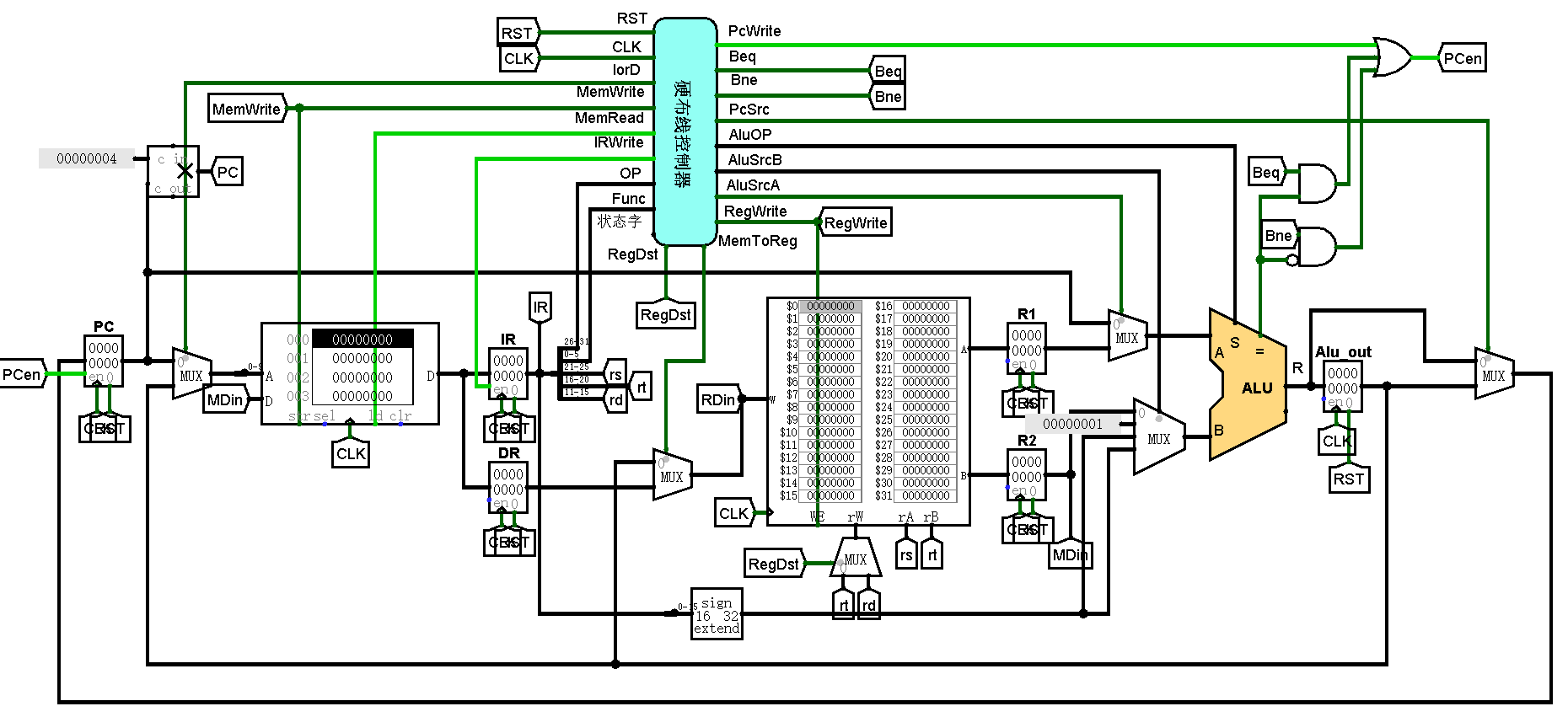


图 2‑13 硬布线控制器多周期MIPS CPU

## 实验步骤

1. 设计控制器
2. 构建数据通路
3. 测试联调

## 故障与调试

### 单周期MIPS CPU运行时PC值出错

故障现象：在涉及跳转指令时单周期MIPS CPU运行时PC值出错。

原因分析： 仔细检查跳转指令所走的数据通路，发现立即数的拓展器不是符号扩展。

解决方案： 修改为符号扩展。

### 微程序控制器多周期MIPS CPU运行出错

故障现象：取不出指令。

原因分析： 没有载入程序sort.hex。

解决方案： 载入程序。

### beq指令运行出错

故障现象：beq指令后无法继续取指令。

原因分析：微指令ROM的下址字段设置错误，导致地址转移逻辑失效。

解决方案：重新核对微指令编码，确保beq指令结尾下址字段回到状态S0。

## 测试与分析

本地测试3种CPU均能完成排序程序，实验平台也测试通过，综合来看本次实验成功。

## 实验总结

本次实验主要完成了如下几点工作：

1. 完成了单周期MPIS CPU，微程序控制器多周期MPIS CPU和硬布线控制器多周期MPIS CPU。
2. 实现了能完成运行8种MIPS指令的CPU。

## 实验心得

1. 通过构建单周期和多周期CPU，我深刻理解了计算机系统的层次化设计思想。从ALU、寄存器堆等基础部件到完整的CPU数据通路，每个模块都需要严格验证功能正确性才能保证系统级联后的稳定性。特别是在多周期CPU设计中，状态机的精确划分（如取指、译码、执行等阶段）让我体会到时序控制的重要性。
2. 单周期硬布线控制器：直接通过组合逻辑生成控制信号，设计简单但效率较低；微程序控制器：通过微指令ROM实现状态跳转，灵活性高但需要精心设计微程序地址转移逻辑；多周期硬布线控制器：结合状态机与组合逻辑，在性能和复杂度之间取得平衡。这三种实现方式让我对不同控制策略的优劣有了直观认识。
3. 调试能力的提升：实验中遇到的故障（如2.4.1节PC跳转错误）促使我深入分析数据通路信号流向。例如，发现PC跳转出错时，通过追踪相关数据通路过程，最终定位问题根源。
4. 对课程的建议：有时候平台测试出错，但评测的结果会因为太长而无法显示全部，对调试很不方便，希望可以改进。

|  |
| --- |
| 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:** |
| 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:** |