

数字逻辑实验二 实验报告

**学 院： 计 算 机 工 程 学 院**

**班 级： 计算2114**

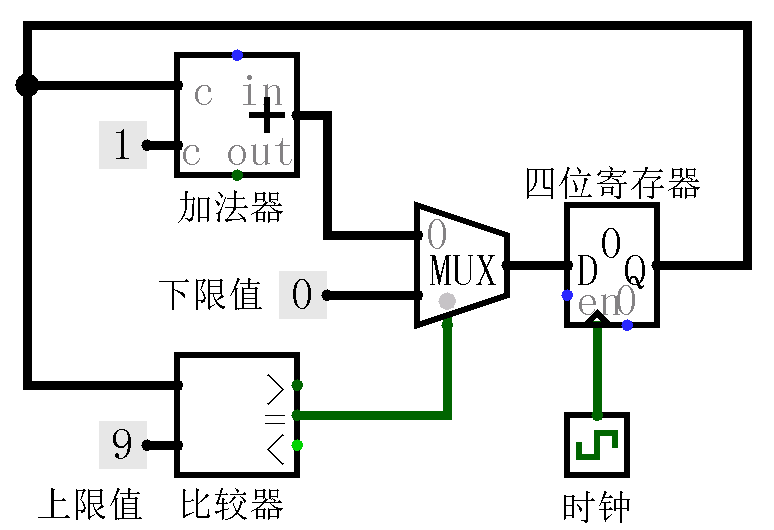
**姓 名: 庄佳强**

**学 号： 202121331104**

二〇二一 年 4 月 12日

# 实验一.D

# 一 题目要求

**设计一个二进制正计时器** 为了能够计时，可以利用时钟、触发器和运算器搭建出一个能够同时进行运算和存储的电路。比如，下面这个电路在每一个时钟的上升沿，计数器的值会增加1，但达到上限值后又会清零：  
  
  
上图中的设计采用的是Logisim封装好的寄存器。你需要根据这个结构，用你自己设计的4位寄存器替换掉Logisim提供的4位寄存器，实现出一个能够进行0~9秒正向计时的数字秒表。这个设计的具体要求如下：

* + 1. 存储单元（比如D触发器）必须使用基本的逻辑门进行搭建；
    2. 使用一个7段数码管展示计数器的值（你可以使用Logisim提供的自带译码功能的“16进制数字显示器”）；
    3. 设计一个重置的功能，用户可以利用重置按钮，将计时器的值恢复为0；按钮是一个可以自动恢复原值的输入，可以扮演类似家用电器中的“重置按钮”的作用，你可以在Logisim当中找到这个器件；

正常工作时，七段数码管上将循环显示0—1—2—3—4—5—6—7—8—9—0...（不能出现0~9之外的数字）；  
   
注意，Logisim封装好的寄存器具有确定的初始值，而我们自己搭建的寄存器的初始值是不确定的。一种实现重置功能的方法是利用数字电路中的复用器，以达到如下伪代码的功能：  
 if (重置按钮被按下) 寄存器的输入端口 = 0；  
 else 寄存器输入端口 = 循环变化的计数值；

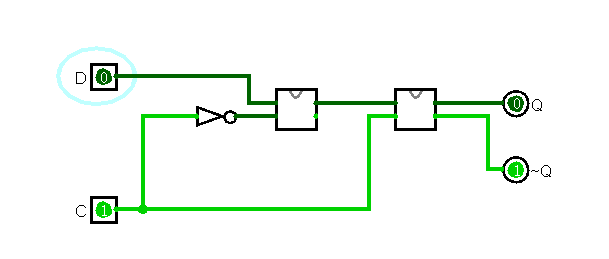
# 二 设计思路

1. 利用自制的D触发器，来输出数字
2. 输出的数字返回到加法器中加一，在通过数据选择器回到D触发器。
3. 当D触发器返回的数据为9时，数据到比较器，比较器输出1到数据选择器，让他返回0重置数据。

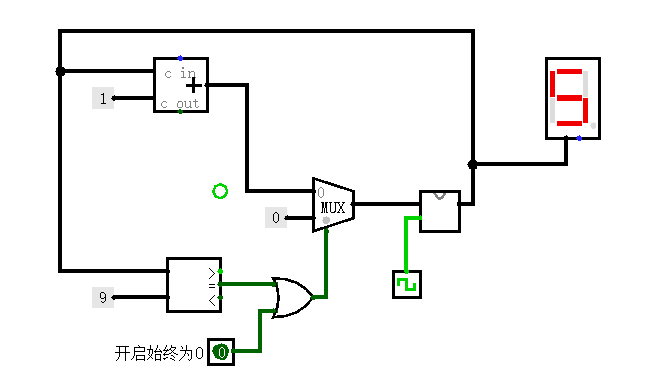
# 三 成果展现

1. **电路设计图**

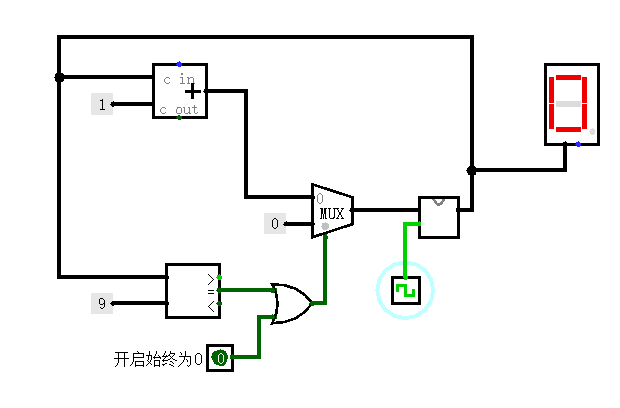
D触发器

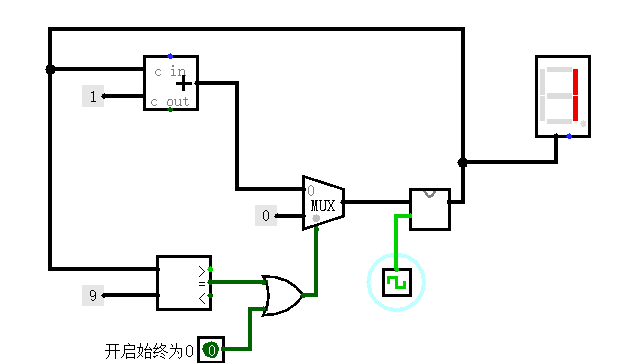


二进制正计时器

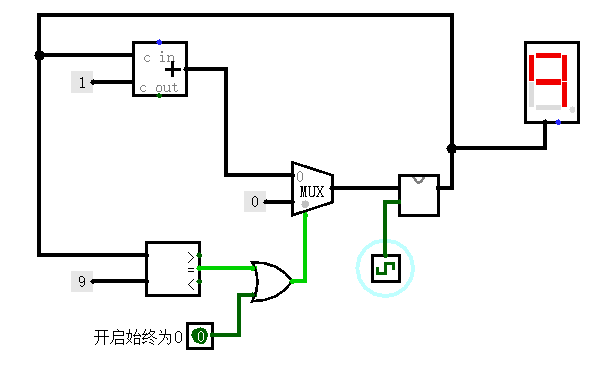


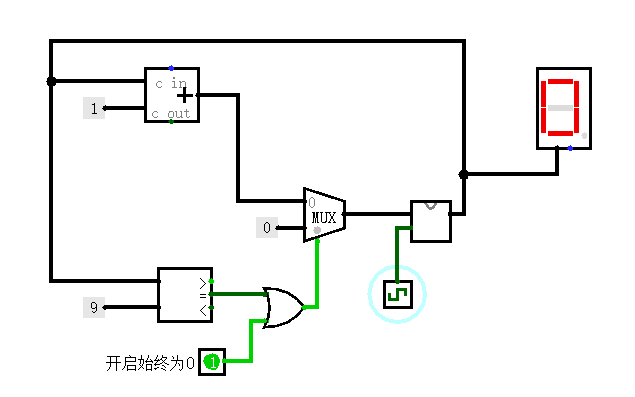
1. **实验数据**





......





# 四 实验细节

由于自制的，没有初始值，需要使用开关给他附上一个初始值。

# 五 实验总结

通过这个实验，我学会了循环的用法和数据复位的方式。

还有加法器累加的方式。

# 实验一.E

# 一 题目要求

**设计一个走马灯**  
 走马灯是一种利用LED灯动态闪烁实现的应用电路，被广泛应用于装饰领域（应用场景如下图所示）。  
   
 采用走马灯原理的夜景灯效果  
你需要结合带并行数据加载的移位寄存器和LED，结合Logisim提供的LED设计出一个走马灯。具体设计要求如下：

LED灯的数量不少于8个；

为走马灯提供一个数据预加载的模式，允许用户初始化走马灯内的数据（你依然可以利用复用器来实现如下伪代码：  
 if （初始化模式） 移位寄存器 = 初始化数据；  
 else 移位寄存器 = 内部数据；  
）；

为走马灯设计一种显示模式（显示模式可以由你设计，比如移位、闪烁等视觉效果）；

使用你自己设计的D触发器完成设计；

# 二 设计思路

1. 利用D触发器制作走马灯的样式。
2. 利用二进制数来模拟走马灯的显示方式。

利用前面的二进制正计时器中的循环思路设计一个累加器，让二进制数可以一个个进位显示。

1. 利用开关重置数据。

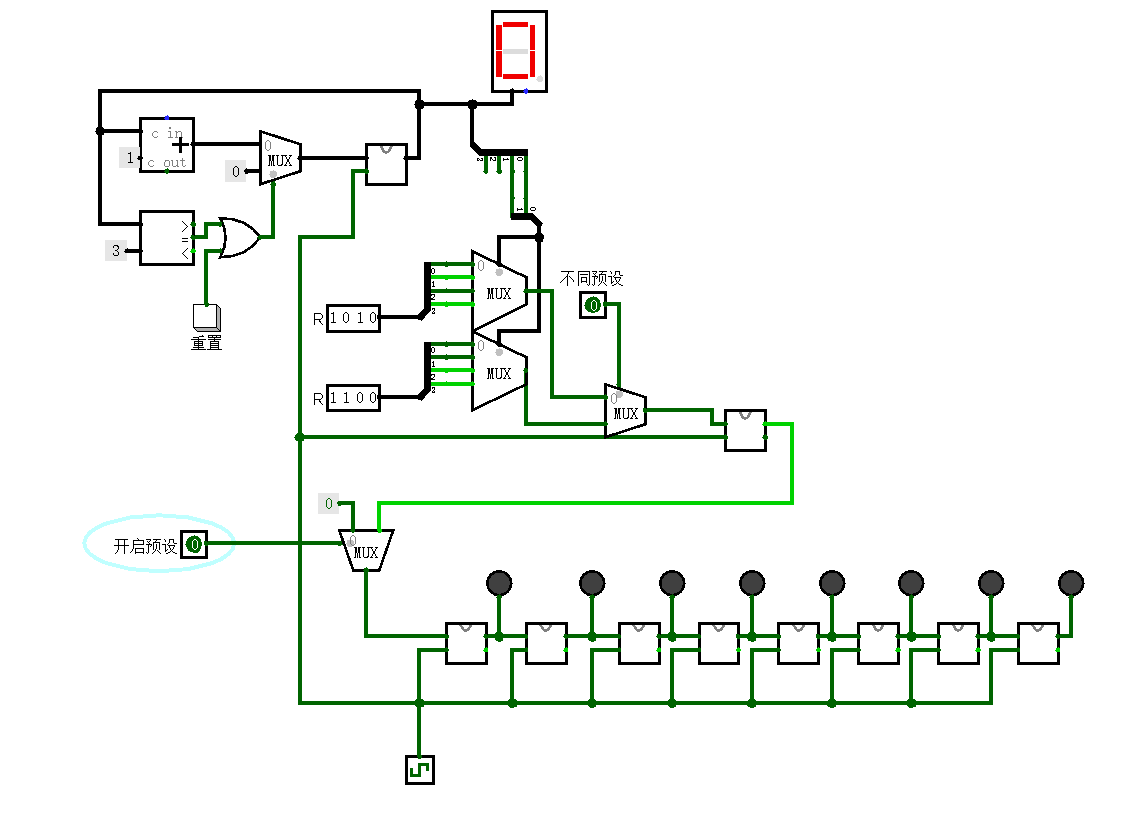
# 三 成果展现

**1.电路设计图**

1.走马灯

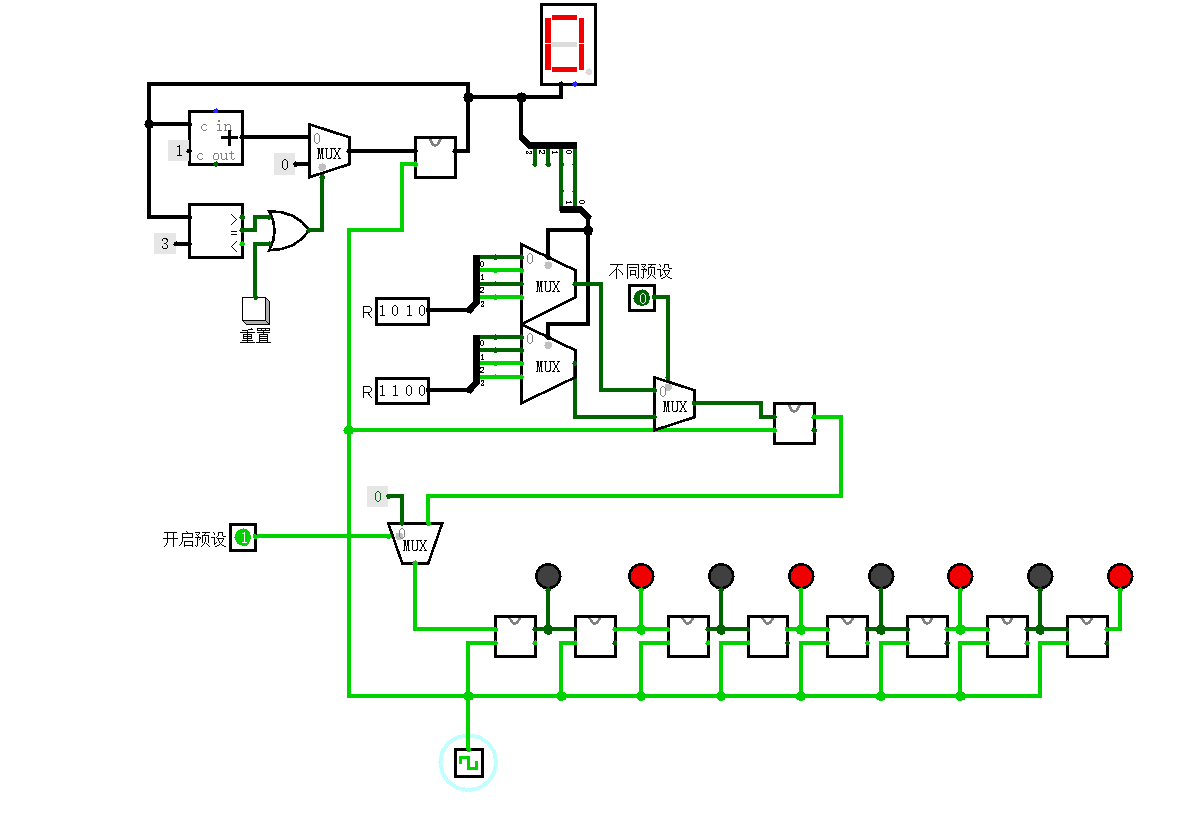
# 

当开启预设为0时

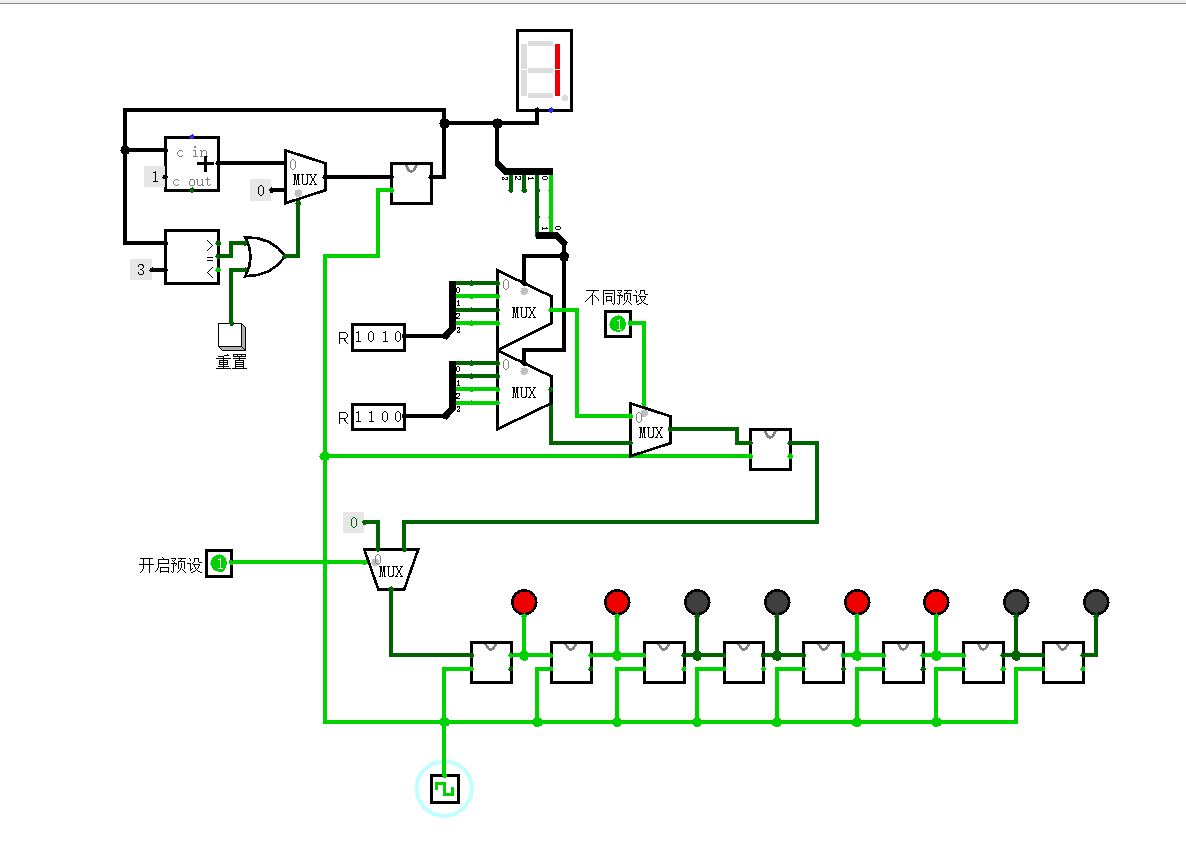


为重置状态

当开启预设时，且预设为0时



当开始预设时，且预设为1时



# 四 实验细节

1.需要使用一个开关来重置循环器，不然，初开无法使用。

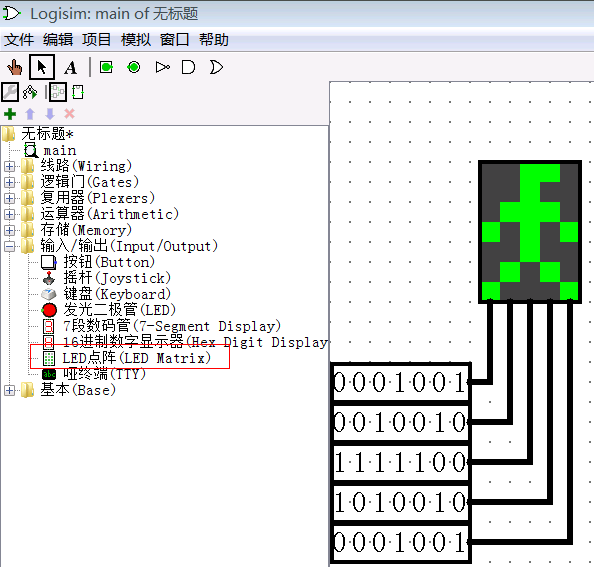
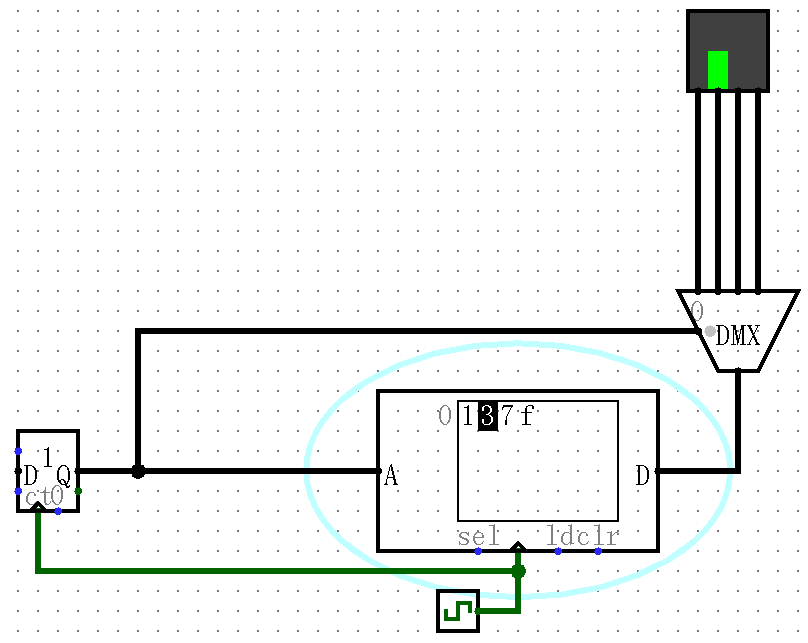
2.需要一个开关开重置走马灯，不然也无法使用。

# 五 实验总结

这是循环和数之间的结合导出，为后面提供了思路。

# 实验一.F

# 一 题目要求

使用存储器阵列模型存储显示数据  
 在我们的理论课上，我们已经设计出一个4字×2位的存储器阵列，并且能够对每个字进行独立的读或写操作。存储器阵列的主要价值在于提供更大规模的数据存储。在本部分实验当中，我们将使用一个存储器阵列保存点阵屏所需要的显示数据。这个工作原理如下图所示：  
  
你可以从Logisim的菜单中找到点阵屏模块，其原理是点阵屏的每一列使用一个专门的二进制数控制亮灭：  
  
我们可以将希望显示的信息预先存储在存储器阵列当中，然后通过改变地址间接地改变存储器阵列的输出，最终达到期望的视觉效果。一个简单的电路结构如下如所示：  
  
图中的存储器阵列存储的”137f”是一组预设的数据，在计数器、时钟和解复用器的帮助下，存储器阵列输出的显示数据将产生类似于电池充电的动画效果。本部分实验具体设计要求如下：

显示数据

显示模块

存储器阵列

使用你自己搭建的4×4存储阵列替换上图中封装好的存储器阵列；

你可以使用Logisim已封装好的计数器来产生变化的地址；

使用点阵屏展示如上图所示的动画效果；

存储器能够允许用户初始化存储器阵列中的显示数据；

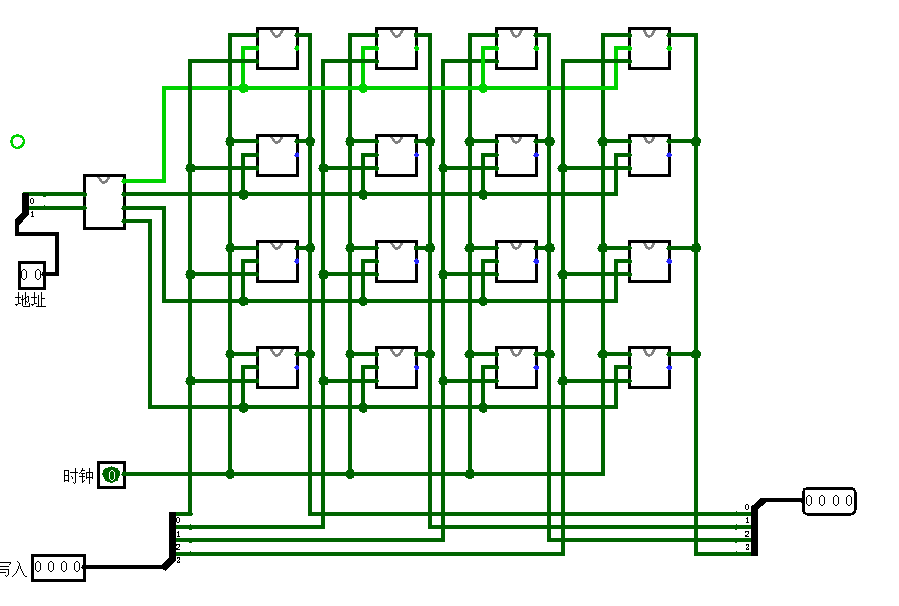
# 二 设计思路

1.先自制一个4×4矩阵更换题目中封装好的矩阵。

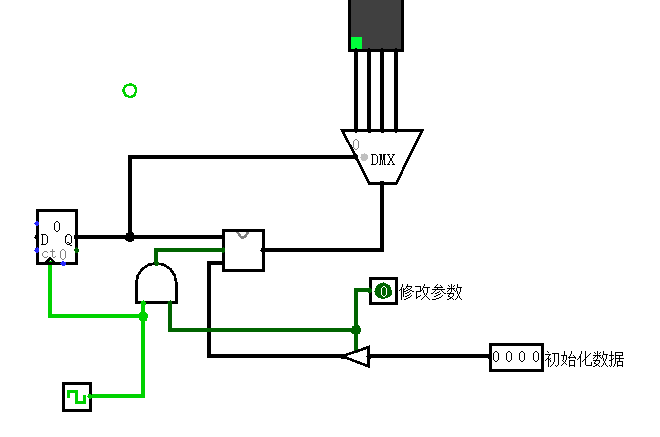
2.在用三态缓冲器设计初始化区域和写入。

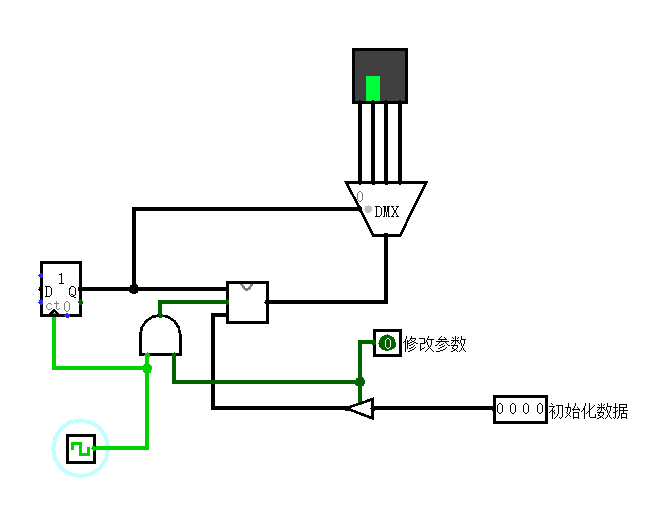
# 三 成果展现

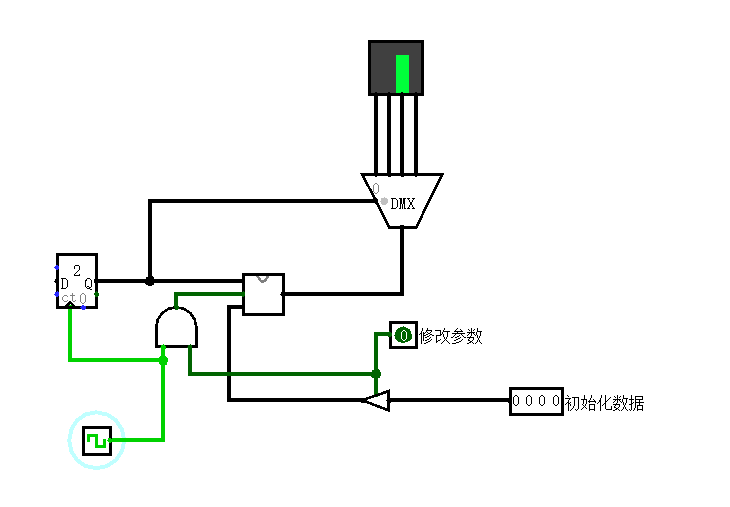
矩阵部分

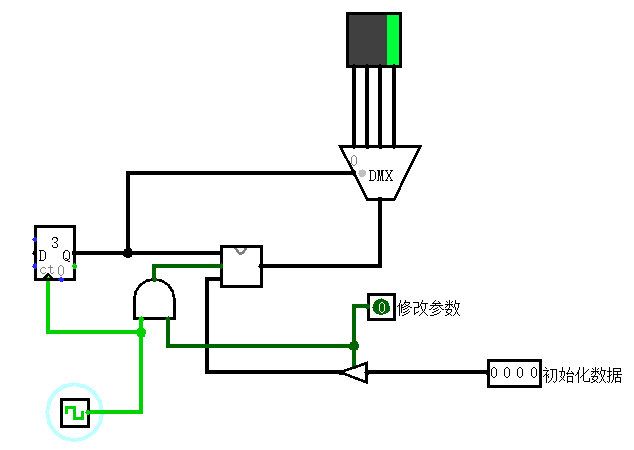


灯部分









# 四 实验细节

一开始没有数据，需要先写入数据。

# 五 实验总结

通过这个实验我学会了矩阵的写入和读出，掌握了计数器的使用，和数据的开始写入和断开写入。

# 实验二.附加二

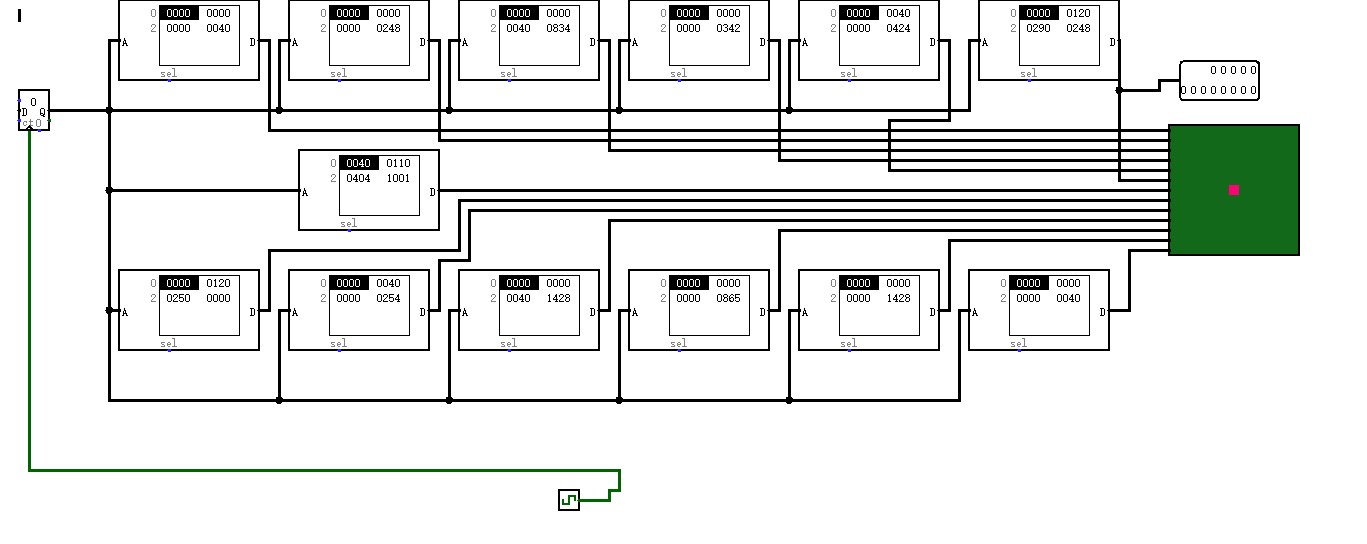
# 一 题目要求

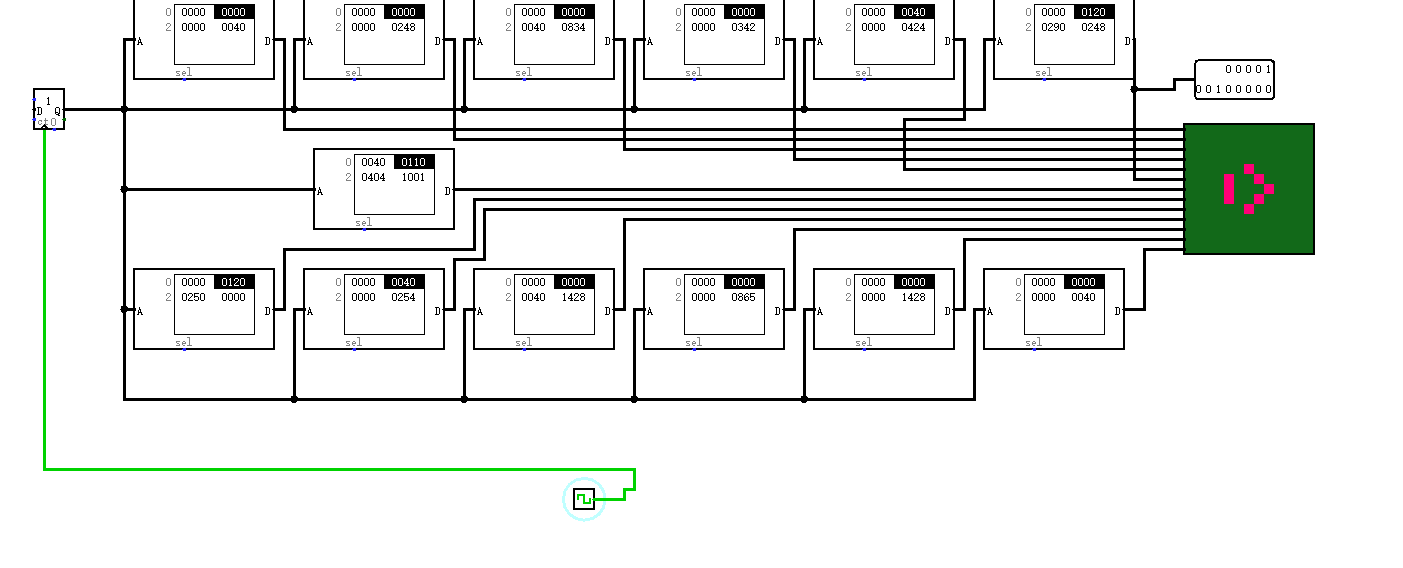
在基础题(f)的基础之上，实现你感兴趣的显示效果（比如旋转的风车、爆炸的烟花），你可以自由地更改电路的结构并使用Logisim封装好的时序器件。

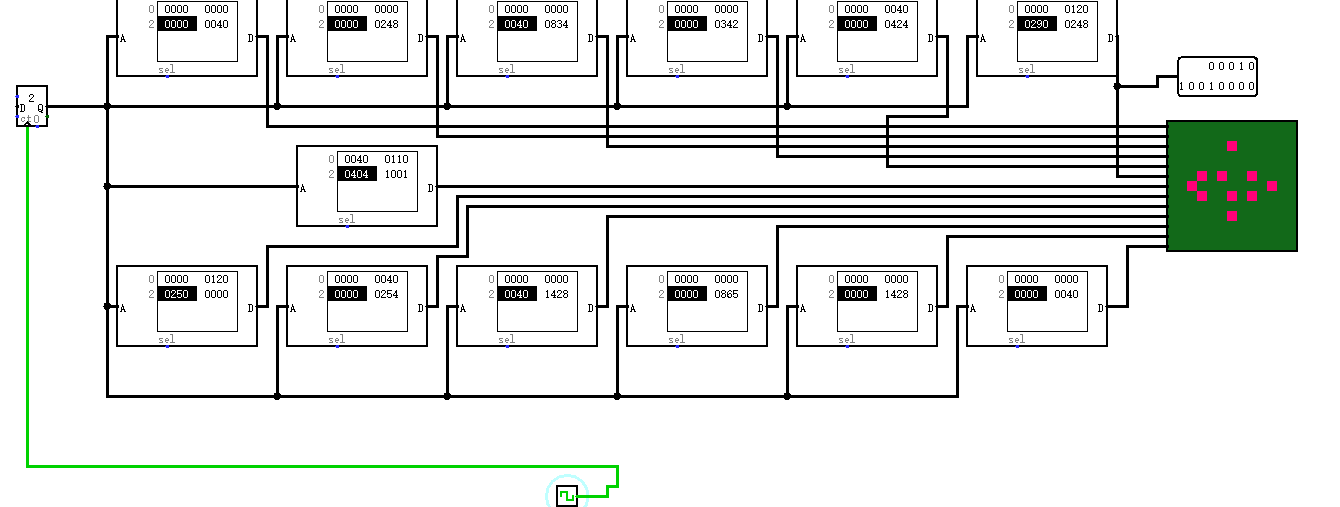
# 二 设计思路

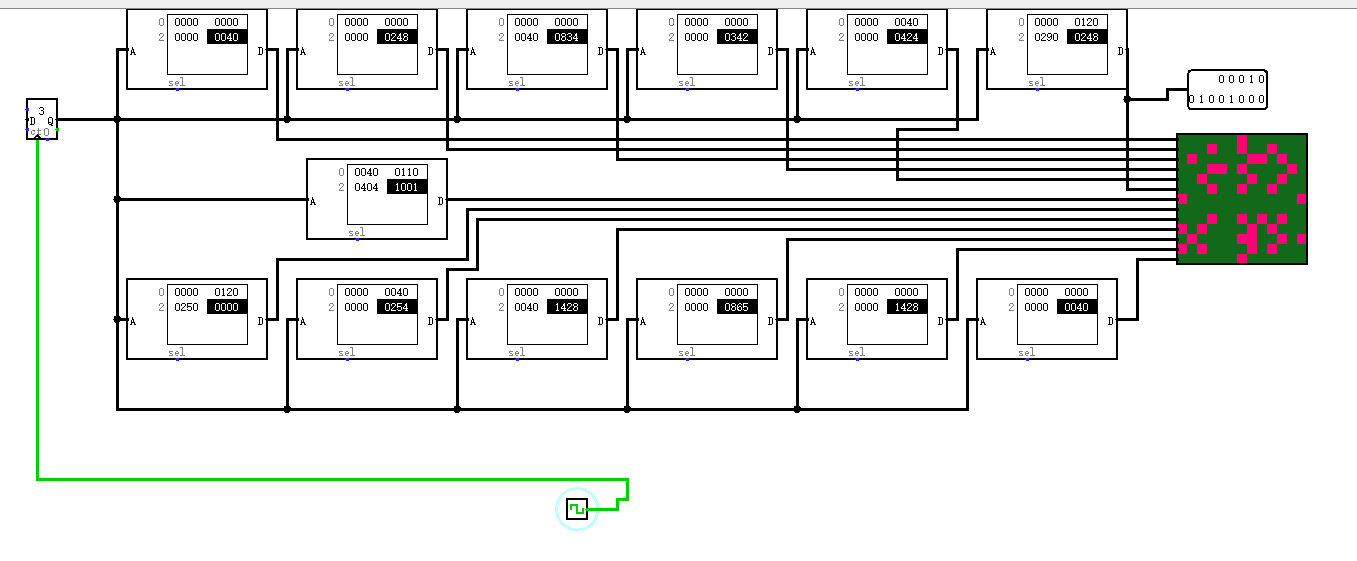
使用程序中自带的随机存储器实现时刻点的存储，在用时钟和LED点阵表现出来，实现类烟花的点阵效果。

# 三 成果展现









# 四 实验细节

注意个个时间点的点的位置。

# 五 实验总结

总之制作出了类烟花效果，但因为点阵太少，无法做出很好的效果，仍需改进。

# 实验二.附加三

# 一 题目要求

利用PYNQ-Z2 FPGA开发板上的时钟，产生一个1秒钟周期性变化的时钟信号。用这个时钟信号去触发一个计数器的工作，再把计数器的值显示在七段数码管上。

# 二 设计思路

利用vivado设计，再焊入FPGA开发板中。

# 三 成果展现

源码：

module clock\_demo(

         input clk,

         input reset,

         output reg [6:0]segments

    );

    reg [31:0]counter;

    reg [3:0]datas;

   always @(posedge clk)

      if(reset)begin

         segments<=0;

         counter<=0;

        end

      else begin

          counter<=(counter==125000000)?0:counter+1;

          if(counter==125000000)begin

              if(datas==9)

             datas<=0;

         else

          datas<=datas+1;

          end

         case (datas)

               'h0:     segments = 7'b111\_1110;

               'h1:     segments = 7'b011\_0000;

               'h2:     segments = 7'b110\_1101;

               'h3:     segments = 7'b111\_1001;

               'h4:     segments = 7'b011\_0011;

               'h5:     segments = 7'b101\_1011;

               'h6:     segments = 7'b101\_1111;

               'h7:     segments = 7'b111\_0000;

               'h8:     segments = 7'b111\_1111;

               'h9:     segments = 7'b111\_0011;

               //'ha:     segments = 7'b111\_0111;

               //'hb:     segments = 7'b001\_1111;

               //'hc:     segments = 7'b000\_1101;

               //'hd:     segments = 7'b011\_1101;

               //'he:     segments = 7'b100\_1111;

               //'hf:     segments = 7'b100\_0111;

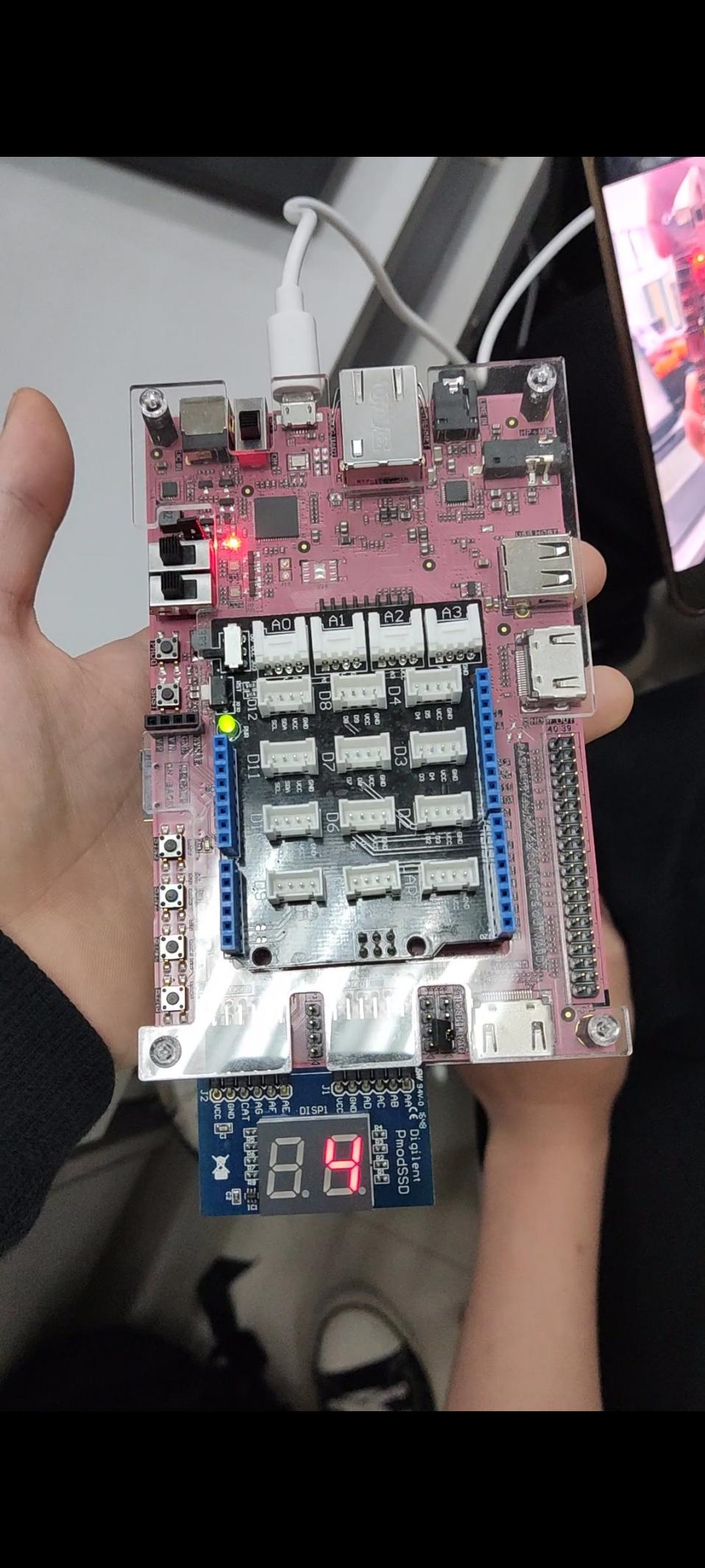
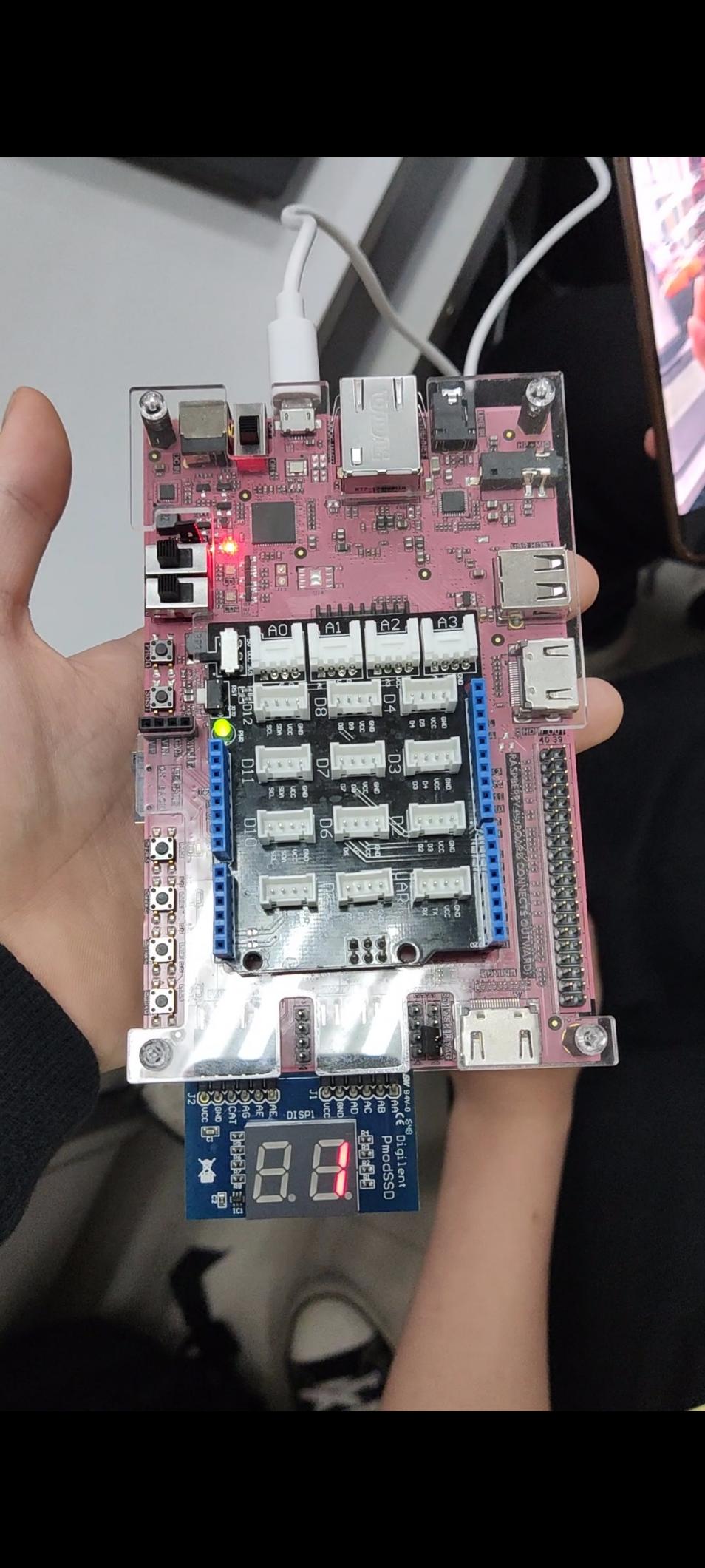
             default:   segments = 7'b000\_0000;

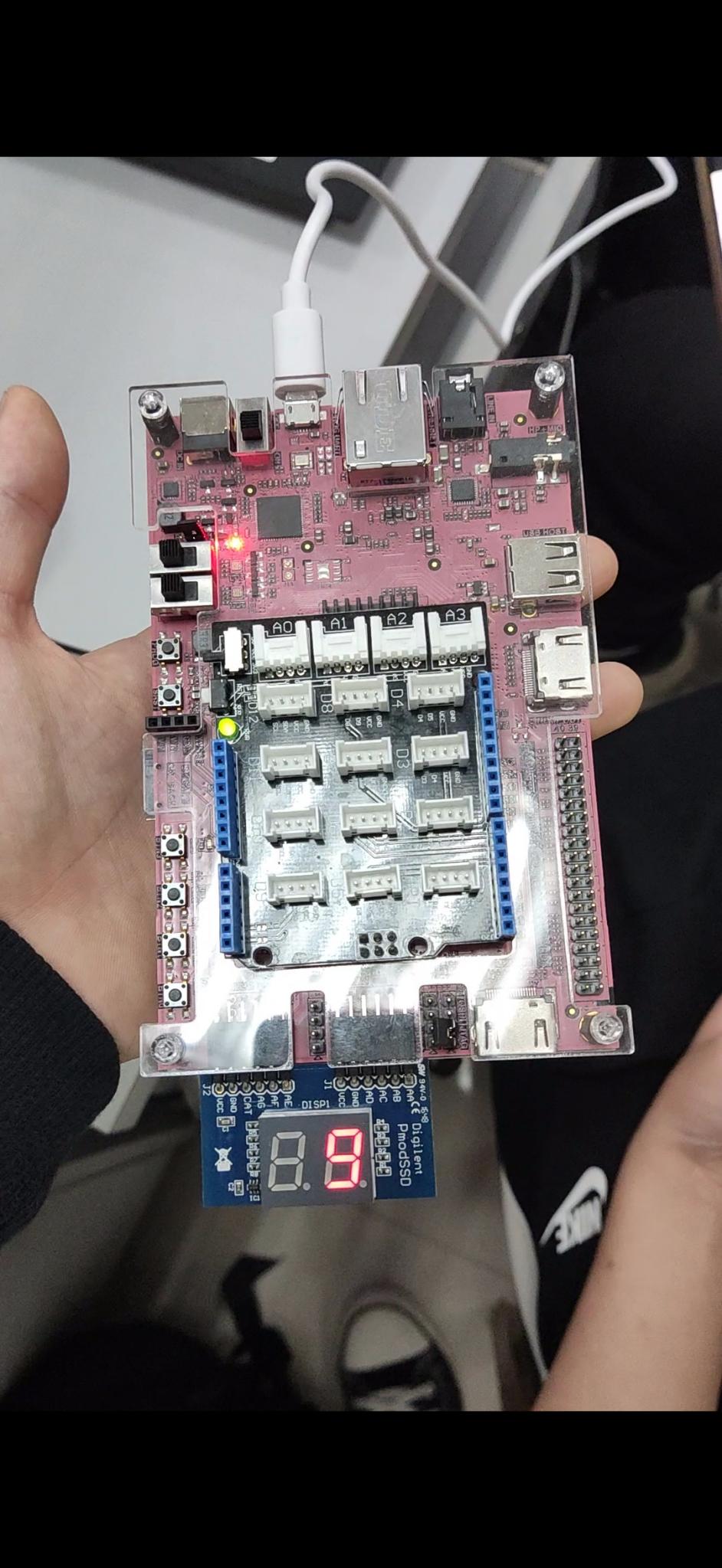
         endcase

         end

endmodule

运行结果：





# 四 实验细节

当数字到9后应该重新之为零。

# 五 实验总结

学习到了vivado和FPGA开发板的进阶玩法，学到了用时钟控制元素变化。