SHANGHAI UNIVERSITY

2021-2022 学年

夏季学期课程报告

计算机硬件综合大型作业

广告灯控制器

小组序号: 2-6

项目名称: 广告灯控制器

指导老师: 张云华

组员学号姓名:

21121310 李欣益

21121319 刘彦辰

21121321黄培远

21121326梁峻玄

21121331王家旭

计算机工程与科学学院

报告日期 2023 年 7 月 4 日

**目录**

[1 项目内容 1](#_Toc139391647)

[2 项目意义 1](#_Toc139391648)

[3 项目设计 1](#_Toc139391649)

[3.1 抽象设计和问题规划 1](#_Toc139391650)

[3.2 芯片原理 2](#_Toc139391651)

[3.2.1 74LS138 2](#_Toc139391652)

[3.2.2 74LS161 3](#_Toc139391653)

[3.2.3 D 触发器 3](#_Toc139391654)

[3.2.4 74LS373 4](#_Toc139391655)

[3.3 自制元件 4](#_Toc139391656)

[3.3.1 4\_16\_Decoder 4](#_Toc139391657)

[3.3.2 UPDATE\_CHECK 5](#_Toc139391658)

[3.3.3 FRAME\_LATCH 6](#_Toc139391659)

[3.3.4 Pattern\_[7..0] 7](#_Toc139391660)

[3.3.5 Pattern\_[7..0]\_Frame\_[15..0] 7](#_Toc139391661)

[3.4 电路设计 7](#_Toc139391662)

[4 波形模拟 9](#_Toc139391663)

[4.1 16 帧为周期循环播放 9](#_Toc139391664)

[4.2 切换显示模式 10](#_Toc139391665)

[4.3 暂停显示 10](#_Toc139391666)

[4.4 重置 10](#_Toc139391667)

[4.5 实机演示 11](#_Toc139391668)

[5 优化方案 11](#_Toc139391669)

[5.1 利用三态门代替 LATCH (74LS373) 11](#_Toc139391670)

[5.2 利用存储芯片替代 Frame 元件 11](#_Toc139391671)

[6 项目小结 12](#_Toc139391672)

# 项目内容

设计一个广告灯控制器, 要求如下:

1. 利用 6 个数码管 + 16 个灯 (必须有组合).
2. 至少 8 中以上模式的显示方式.

# 项目意义

本次实践项目通过设计广告灯控制器, 增加灯闪烁模式的可变形与多样性, 带来更加丰富、吸引人的效果. 不同的动态效果和切换模式可以吸引观众的注意力, 增加观赏的乐趣和参与感, 具有一定的实际意义.

而作为计算机系的学生, 本次硬件大作业提供了一个实践的机会. 通过参与计算机硬件大型项目这一实践环节, 我们对之前的课程进行了全面回顾和审查, 例如数字逻辑与计算机组成原理课程的基本原理、基本概念, 以提升我们对所学理论知识的系统性、联结性和综合性. 这培养了我们独立思考、自主分析和解决问题的能力, 同时提升了我们的硬件设计能力, 并深化了对计算机系统整体的理解, 使我们具备设计完整电路的综合能力. 通过小组成员的协作合作, 我们成功地设计了一个具有实际意义的电路成品, 这将有助于我们更好地胜任结合计算机系统软硬件的应用开发工作.

# 项目设计

## 抽象设计和问题规划

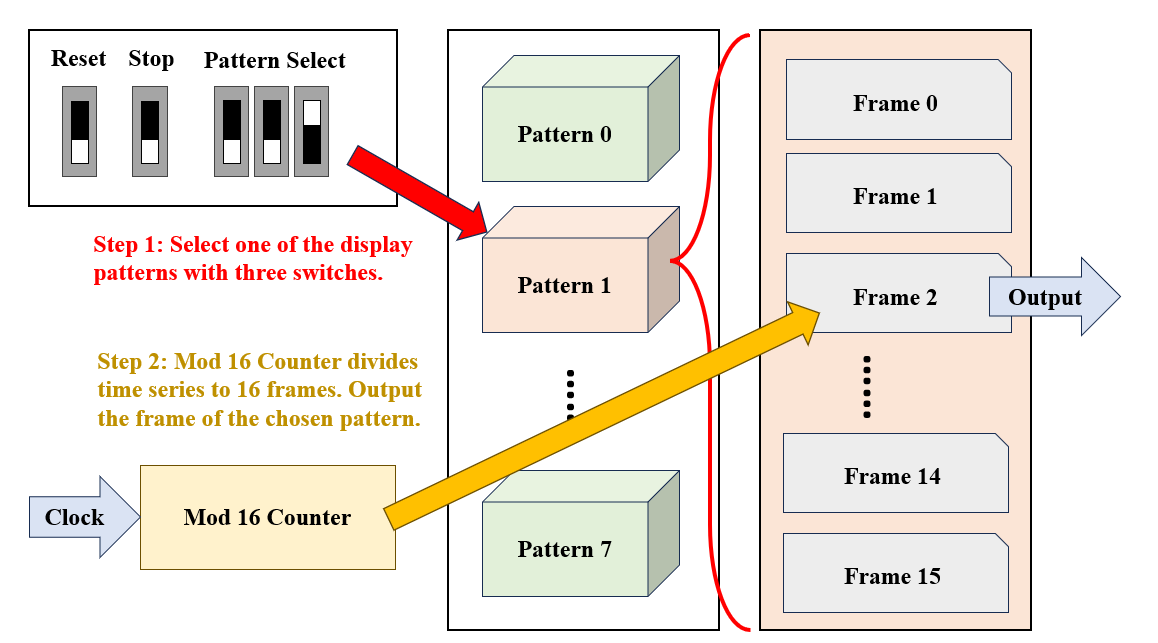


图 3.1.1. 抽象设计图

图 3.1.1 展示了我们项目的抽象设计图. 总体来说, 三个开关用于选择想要的显示模式 (一共有 8 种模式可以选择, 分别为 Pattern 0 .. 7). 同时, 一个模 16 计数器接受时钟信号, 在每个上升沿加一; 因此连续的时间序列被分为了 16 帧 (Frame). 对于唯一被选中的显示模式, 我们按照时间序列的顺序依次输出每一帧数据; 而所有没有被选中的帧应该通过某种方式被排除在输出总线以外, 避免对输出总线的写冲突.

另外, Stop 开关被用于暂停当前输出, 恢复低电平后能够从暂停点继续输出; 而 Reset 开关被用于将某个显示模式的输出帧重置为第一帧 (Frame 0), 恢复低电平后从第一帧开始输出. 当输出模式被切换, 应该执行 Reset 操作; 也就是新的模式应该总是从第一帧开始输出.

为满足上述设计要求, 我们使用了多组 74 系列芯片以及门电路. 同时, 为了模块化设计需要, 我们定义了若干自己的元件. 因此 3.2 节将介绍我们使用的内置芯片原理, 而 3.3 节将介绍我们的自制元件.

## 芯片原理

### 74LS138

74LS138 为3 线-8线[译码器](https://baike.so.com/doc/5931545-6144473.html" \t "_blank), 共有 54LS138和 74LS138 两种线路结构型式. 54LS138为军用, 74LS138为民用.

1. 当一个选通端 (E1) 为高电平, 另两个选通端 ((/E2))和(/E3)) 为低电平时, 可将地址端 (A0、A1、A2) 的[二进制编码](https://baike.so.com/doc/6337160-6550773.html" \t "_blank)在Y0至Y7对应的输出端以低电平译出. (即输出为Y0至Y7的非) 比如: A2A1A0=110时, 则Y6输出端输出低[电平信号](https://baike.so.com/doc/6807070-7024021.html" \t "_blank).
2. 利用 E1、E2和E3可级联扩展成 24 线[译码器](https://baike.so.com/doc/5931545-6144473.html" \t "_blank);若外接一个[反相器](https://baike.so.com/doc/6499299-6713014.html" \t "_blank)还可级联扩展成 32 线译码器.
3. 若将选通端中的一个作为数据输入端时, 74LS138还可作[数据分配器](https://baike.so.com/doc/7894385-8168480.html" \t "_blank).
4. 可用在8086的[译码](https://baike.so.com/doc/5384212-5620621.html" \t "_blank)电路中, [扩展内存](https://baike.so.com/doc/6282676-6496135.html).

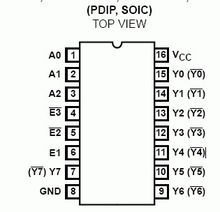


图 3.2.1. 74LS138 引脚图

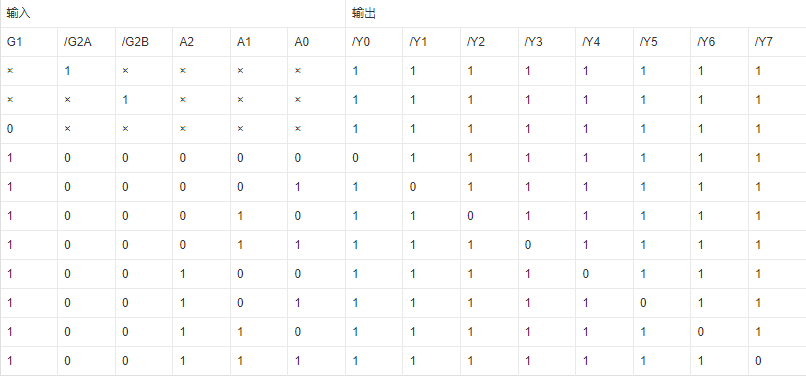


图 3.2.2. 74LS138 真值表

### 74LS161

74LS161为二进制同步计数器, 具有同步预置数、异步清零以及保持等功能.

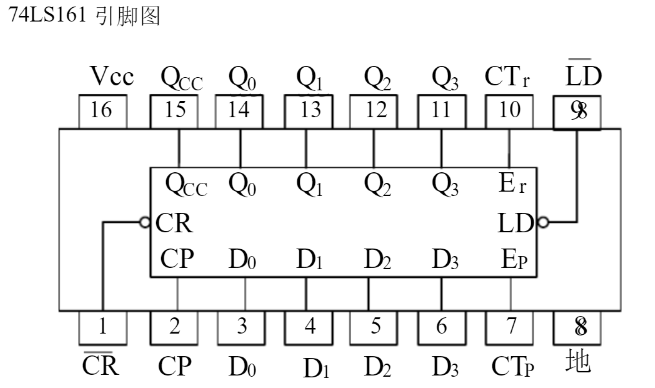


图 3.2.3. 74LS161 引脚图

如果清零端CR=“0”时, 计数器输出Q3、Q2、Q1、Q0都会马上为全“0”, 这个时候是异步复位功能. 当CR=“1”并且LD=“0”时, CP[信号](https://m.hqchip.com/app/1072)上升沿作用之后, 74LS161输出端Q3、Q2、Q1、Q0的状态分别与并行数据输入端D3, D2, D1, D0的状态一样, 这个时候是同步置数功能. 而只有当CR=LD=EP=ET=“1”、CP脉冲上升沿作用后, 计数器加1.

74LS161还有一个进位输出端CO, 其逻辑关系是CO=Q0·Q1·Q2·Q3·CET. 合理应用计数器的清零功能和置数功能, 一片74LS161可以组成16进制以下的任意进制分频器.

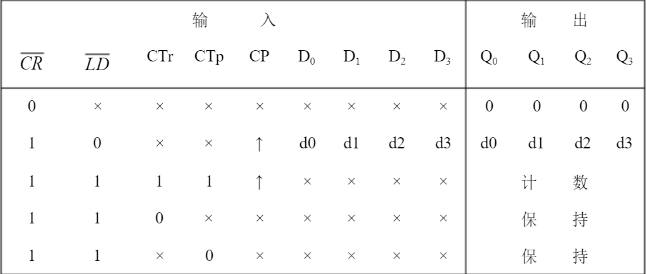


图 3.2.4. 74LS161 真值表

### D 触发器

D触发器是一个具有记忆功能的, 具有两个稳定状态的信息存储器件, 是构成多种[时序电路](https://baike.baidu.com/item/%E6%97%B6%E5%BA%8F%E7%94%B5%E8%B7%AF/4836638?fromModule=lemma_inlink" \t "_blank)的最基本逻辑单元, 也是数字逻辑电路中一种重要的单元电路.

因此, D触发器在数字系统和计算机中有着广泛的应用. 触发器具有两个稳定状态, 即0和1, 在一定的外界信号作用下, 可以从一个稳定状态翻转到另一个稳定状态.

D触发器有集成[触发器](https://baike.baidu.com/item/%E8%A7%A6%E5%8F%91%E5%99%A8/193146?fromModule=lemma_inlink" \t "_blank)和门电路组成的触发器. 触发方式有电平触发和边沿触发两种, 前者在CP(时钟脉冲)=1时即可触发, 后者多在CP的前沿（正跳变0→1）触发.

D触发器的次态取决于触发前D端的状态, 即次态=D. 因此, 它具有置0、置1两种功能.

对于边沿D触发器, 由于在CP=1期间电路具有维持阻塞作用, 所以在CP=1期间, D端的数据状态变化, 不会影响触发器的输出状态.

D触发器应用很广, 可用做[数字信号](https://baike.baidu.com/item/%E6%95%B0%E5%AD%97%E4%BF%A1%E5%8F%B7/915663?fromModule=lemma_inlink" \t "_blank)的寄存, 移位[寄存](https://baike.baidu.com/item/%E5%AF%84%E5%AD%98/11013202?fromModule=lemma_inlink" \t "_blank), 分频和波形发生器等等.

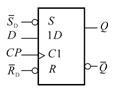


图 3.2.5. D 触发器引脚图



图 3.2.6. D 触发器真值表

### 74LS373

74LS373是[三态](https://baike.baidu.com/item/%E4%B8%89%E6%80%81/4409964?fromModule=lemma_inlink" \t "_blank)输出的八D[锁存器](https://baike.baidu.com/item/%E9%94%81%E5%AD%98%E5%99%A8/10801965?fromModule=lemma_inlink), 共有54S373和74LS373两种线路. 373 的输出端 Q0～[Q7](https://baike.baidu.com/item/Q7/10580180?fromModule=lemma_inlink" \t "_blank) 可直接与总线相连.

当锁存允许端LE为高[电平](https://baike.baidu.com/item/%E7%94%B5%E5%B9%B3/5695184?fromModule=lemma_inlink" \t "_blank)时, Q 随数据D而变. 当LE为[低电平](https://baike.baidu.com/item/%E4%BD%8E%E7%94%B5%E5%B9%B3/6946314?fromModule=lemma_inlink" \t "_blank)时, D 被锁存在已建立的数据电平. 当LE端[施密特触发器](https://baike.baidu.com/item/%E6%96%BD%E5%AF%86%E7%89%B9%E8%A7%A6%E5%8F%91%E5%99%A8/3111144?fromModule=lemma_inlink" \t "_blank)的输入滞后作用, 使交流和直流噪声[抗扰度](https://baike.baidu.com/item/%E6%8A%97%E6%89%B0%E5%BA%A6/4988266?fromModule=lemma_inlink" \t "_blank)被改 400mV.



图 3.2.7. 74LS3743 真值表

## 自制元件

### 4\_16\_Decoder

由于我们想要将显示画面分割为 16 帧, 因此需要将 74LS161 输出的 4 位 16 进制数据进行译码. 因此利用了两片 74LS138 芯片组合为了一片 4: 16 译码器.

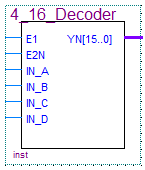
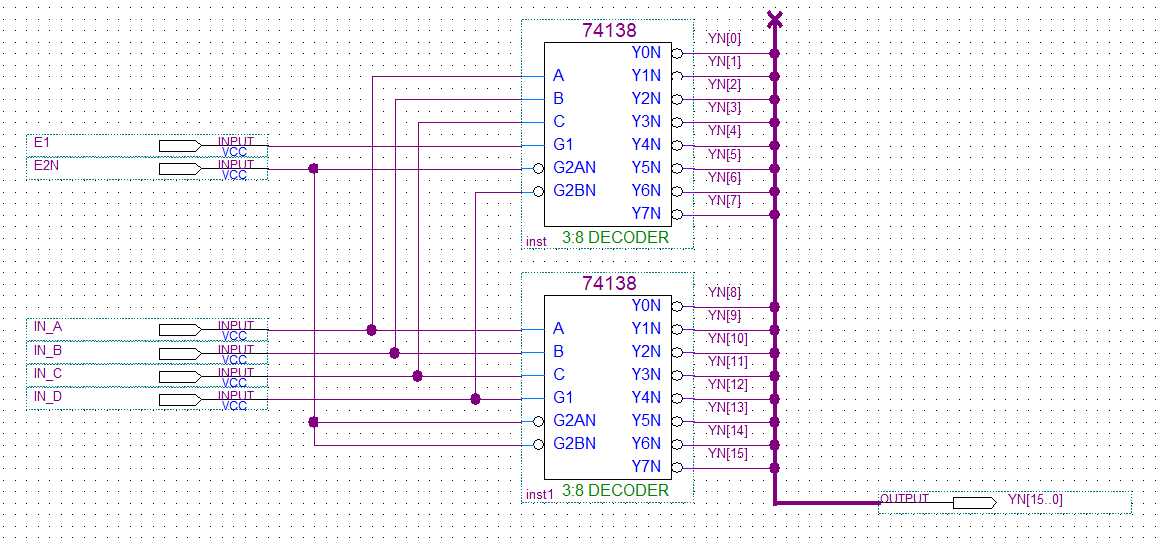
 

图3.3.1 4\_16\_Decoder(左)及原理图(右)

图2.2.1 展示了 4\_16\_Decoder 和其原理图. E1 和 E2N 作为使能端, 分别输入高电平和低电平才能使其有效. 输出为 YN[15..0] 总线, 表示解码后的 16 种情况.

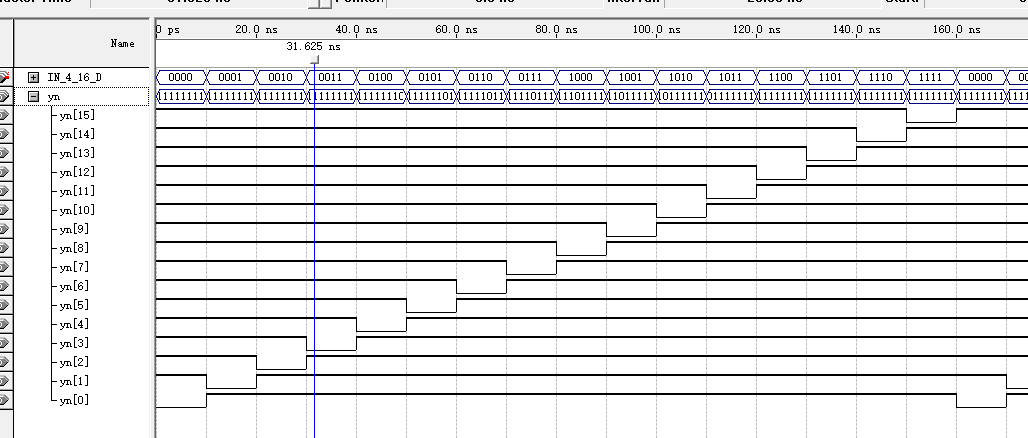


图3.3.2 4\_16\_Decoder 的波形图

图 3.3.2 在展示了 4\_16\_Decoder的测试波形图, 可以观察到测试结果符合我们的预期.

### UPDATE\_CHECK

当进行模式切换后, 我们希望立即重置全局帧数为第 0 帧, 因此需要设计一个 UPDATE\_CHECK 元件检查模式是否被切换.

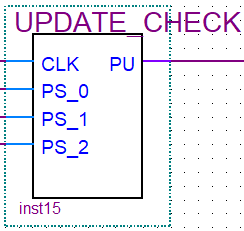
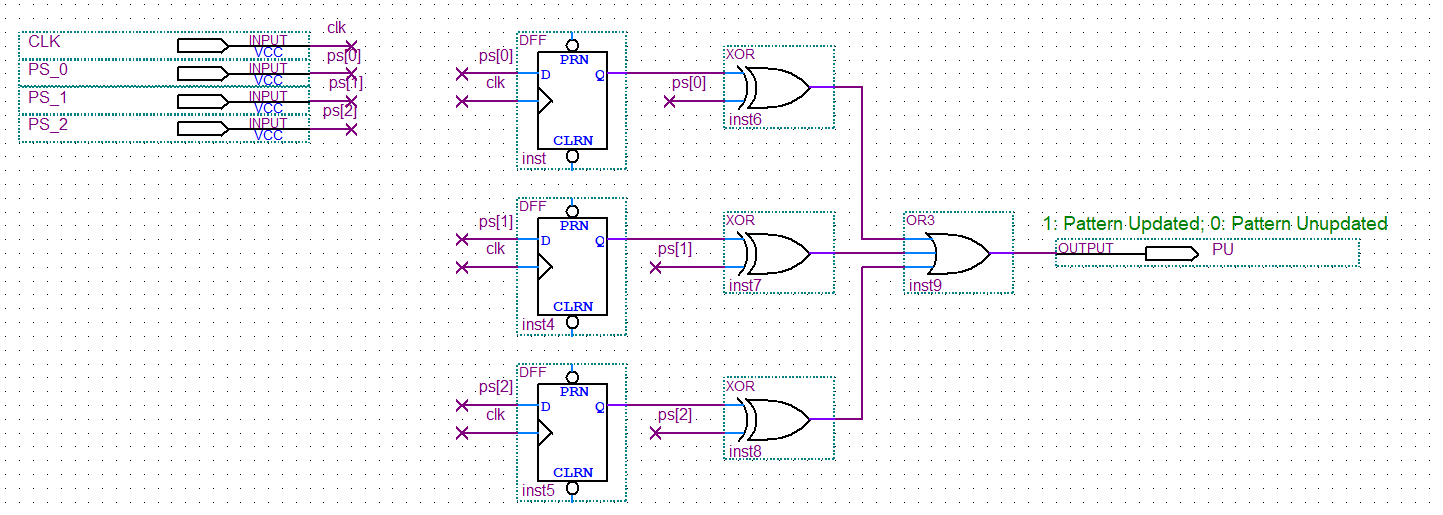
 

图 3.3.3 UPDATE\_CHECK(左)及原理图(右)

图 3.2.3 展示了 UPDATE\_CHECK 元件和原理图. 三个输入 PS\_[2..0] 表示 Pattern Select (模式选择), 即实验箱上的 3 个开关. 三个开关都与 D 触发器相连, 而且 Q 端会延迟 D 端的输入; 因此将三个 Q 端与原本的 PS 输入 (D 端) 异或后再相或, 得到输出 PU. 如果模式切换了 (即延时输出的 Q 端与当前 D 端不一致), PU 将输出高电平; 否则保持低电平.

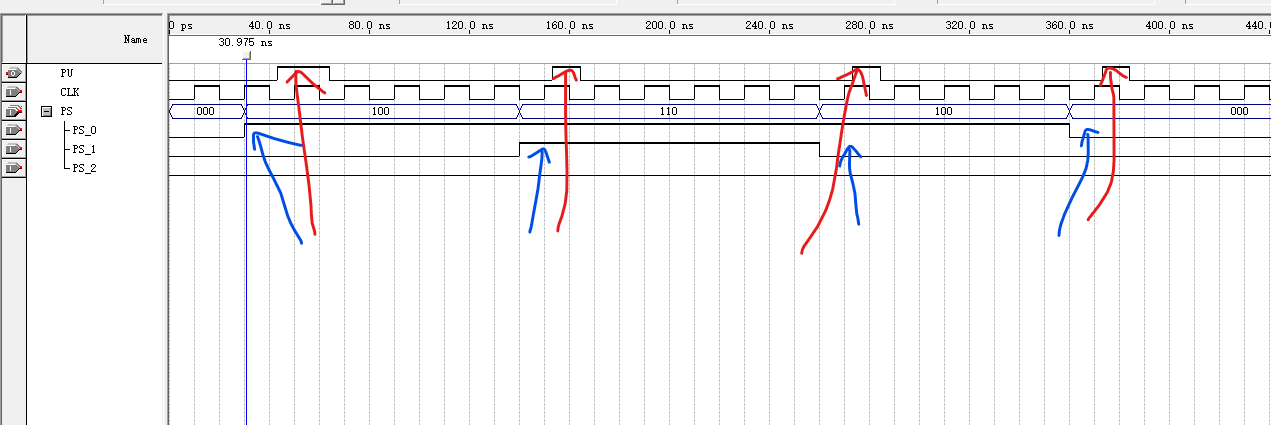


图3.3.4 UPDATE\_CHECK 波形仿真

图 3.3.4 展示了我们对 UPDATE\_CHECK 元件进行的波形仿真. PS (选择的模式) 一共切换了 4 次 (蓝色箭头), 我们也观察到切换后的一小段时间, PU 给出了高电平 (红色箭头), 之后恢复到低电平. 这与我们的预期一致.

### FRAME\_LATCH

当没有片选到某个模式的某帧时, 该帧仍然接在数据总线中. 为了避免数据总线的写入冲突, 我们设计了一个 FRAME\_LATCH 模块. 当某帧没有被选中, FRAME\_LATCH 的输出端应该为高阻态, 此时不会与总线中的数据产生冲突.

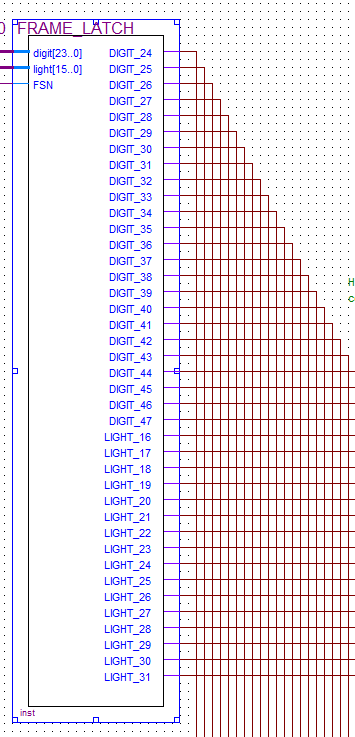
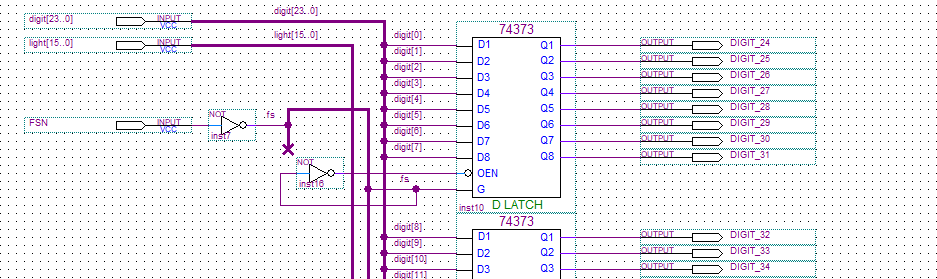
 

图 3.3.5. FRAME\_LATCH(左)及原理图(右)

图 3.3.5 展示了 FRAME\_LATCH 的实现细节. 由于要控制 6 个数码管和 16 个灯, 我们利用了 5 组 74LS373 芯片对 40 个输出信号进行锁存; 同时也有 40 个输入信号 (每一帧预设好的信号). FSN 是帧选择信号, 低电平时锁存器开启, 数据写入总线; 高电平时输出端为高阻态.

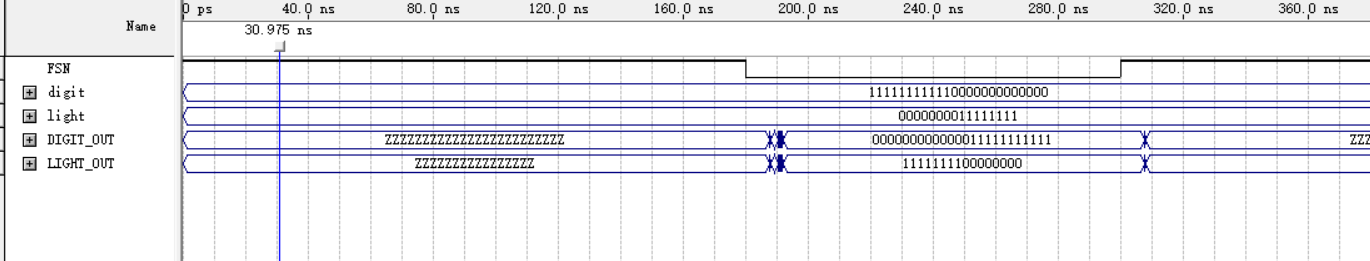


图 3.3.6. FRAME\_LATCH 的时序仿真

图 3.3.6 展示了 FRAME\_LATCH 的时序仿真. 可以看见, 当 FSN 为高电平, 输出端保持高阻态; 而当 FSN 为低电平, 输出端与预设信号一致. 需要注意的是, LATCH 在时序仿真时会出现毛刺现象.

### Pattern\_[7..0]

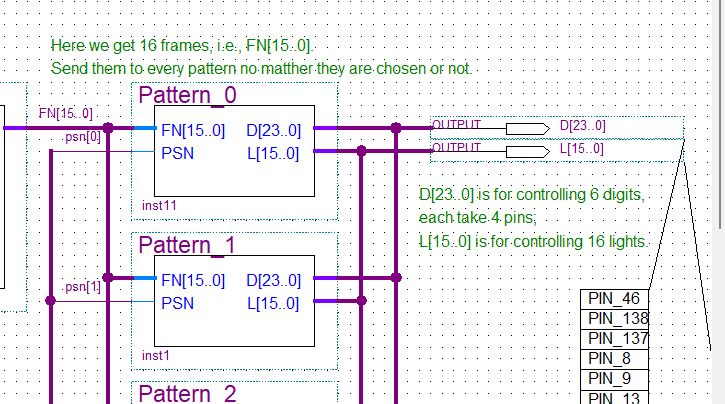


图 3.3.7. Pattern\_0 和 Pattern\_1

图 3.3.7 展示了 Pattern 元件, 具体设计将在 3.3.2 节展开. 总体来说, Pattern\_[7..0] 为 8 种封装好的模式元件, 同时接受 FN[15..0] 的信号进行帧选择; 而 Pattern 元件本身还要经过更上层的 PSN 信号进行片选. 当 PSN 为低电平, Pattern 元件内部会根据帧选择信号 (FN[15..0]) 输出对应帧的信息, 而没被选中的帧处于高阻态; 当 PSN 为高电平, 整个 Pattern 元件的输出会直接处于高阻态.

### Pattern\_[7..0]\_Frame\_[15..0]

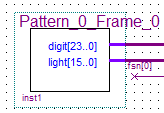
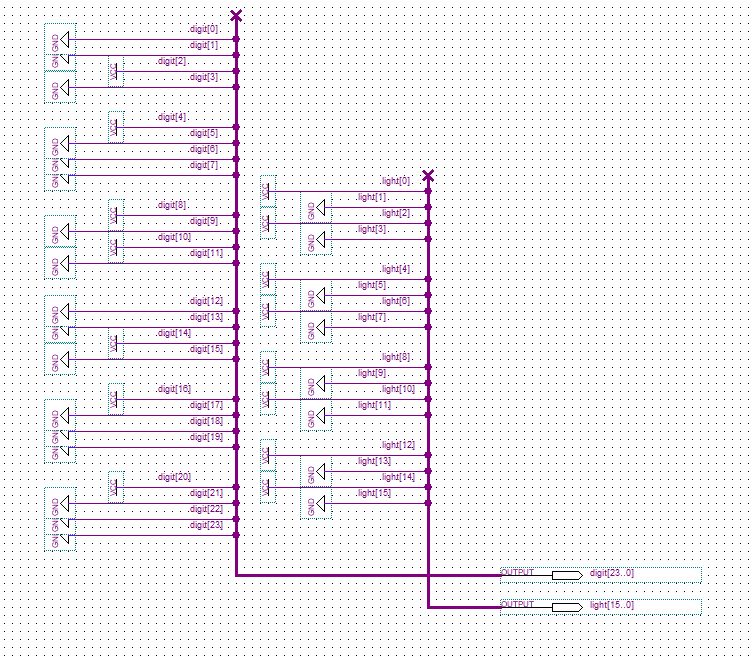
 

图 3.3.8. Pattern\_0\_Frame\_0(左)及原理图(右)

在我们设计的帧元件中, 每帧统一使用一样的输出模式, 只需要更改引脚的高低电平. 这样开发的好处在于, 每位组员可以方便地实现自己的创意.

为了避免写冲突, 每帧后面需要接上 3.3.3 节设计的 FRAME\_LATCH; 这样当一帧没有被选中, 在总线中的表现出高阻态, 不会影响其他帧的输出.

## 电路设计

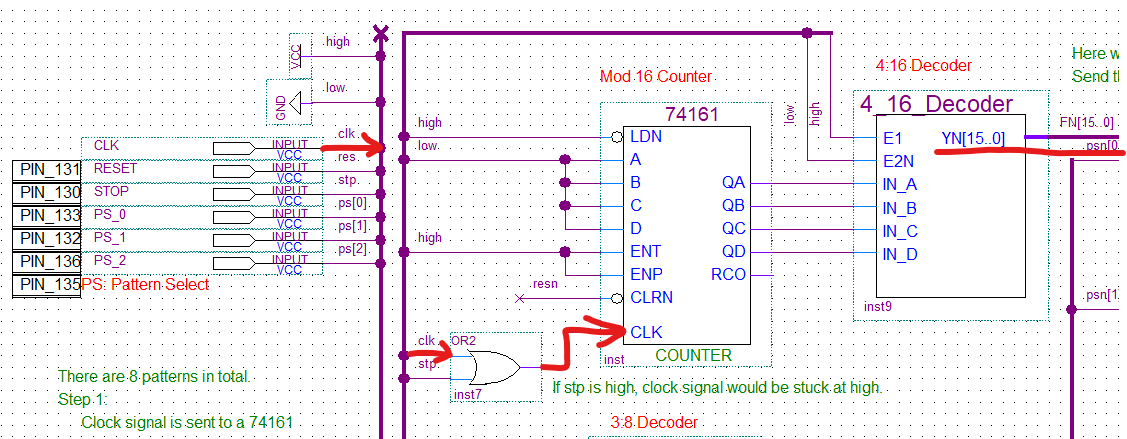


图 3.4.1. 时序信号译码为帧选信号

图 3.4.1 展示了如何对时序信号进行译码, 最终得到帧选信号. 时序信号 clk 传入总线中, 与 stp 相或后传入 74LS161 的 CLK 端. 根据 STOP 开关的高低电平, 分为以下两种情况:

1. STOP 开, 即 stp 为高电平, 则 CLK 输入始终为高电平, 74LS161 处于暂停状态, 输出端数据不会变化.
2. STOP 关, 即 stp 为低电平, 则 CLK 输入为正常的 CLK 时序信号. 此时 clk 的每个上升沿, 74LS161 的输出会加 1; 直到结果为16, 此时输出端归零 (即模 16).

如果 stp 为低电平, 输入的时序信号由 74LS161 累加, 再经过 4\_16\_Decoder 解码, 最终 YN[15..0] 端得到 16 位译码后的输出.

74LS161 的 CLRN 端还接入了 resn 信号; 当 resn 为低电平, 74LS161 的数据会清零 (从零开始计数). resn 的逻辑由图 3.4.2 展示.

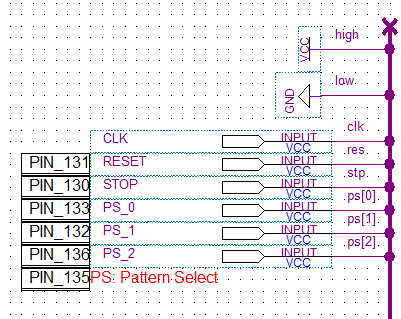
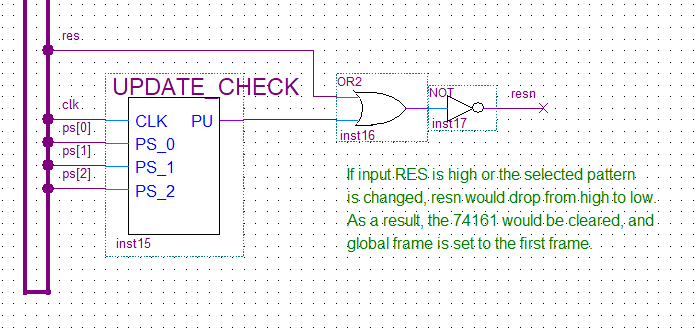
 

图 3.4.2. 输入(左), resn的逻辑(右)

当 RESET 输入为高电平, resn 直接置为低电平. 此时根据图 3.4.1, 74LS161 的计数清零. 也就是某种模式的帧输出会置为第一帧 (Frame 0); 直到 RESET 输入低电平, 开始从第一帧按照时序信号一帧帧向后输出.

此外, 模式选择的三个开关 PS\_[2..0] 也被接入 UPDATE\_CHECK 元件. 根据 3.3.2 节, 当开关信号发生变化 (也就是显示模式改变), resn 也会被置为低电平; 此时新的模式将从第一帧开始输出.

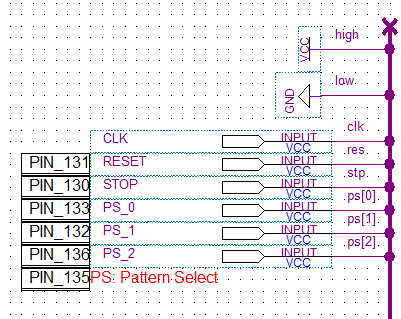
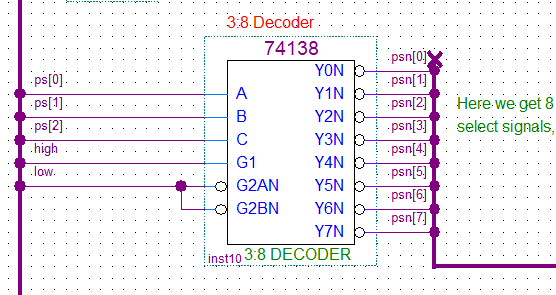
 

图 3.4.3. 输入(左), 对 ps[2..0] 译码得到 psn[7..0] (右)

从图 3.4.3 中可以看到, 我们对 PS\_[2..0] 的输入信号输入 74LS138 进行译码, 最终得到八位模式选择信号 psn[7..0].

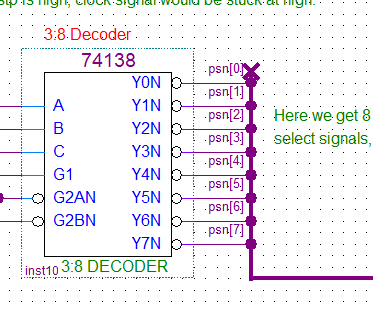
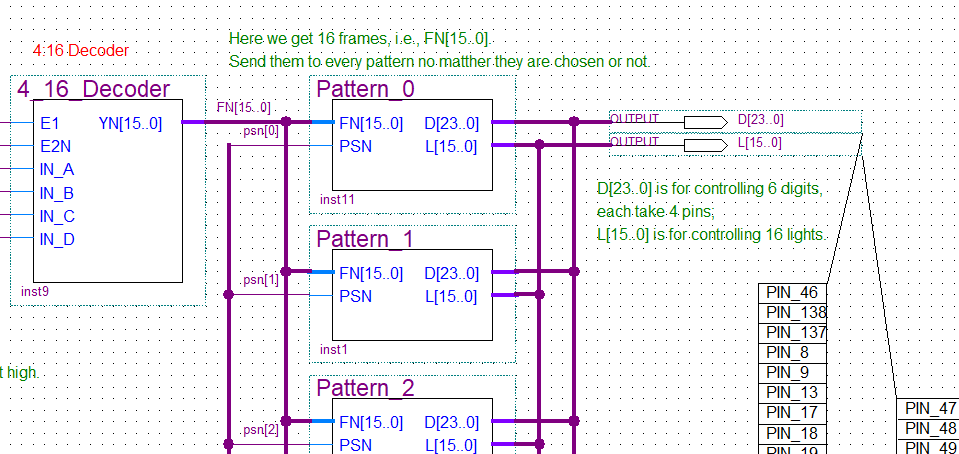
 

图 3.4.4. 译码后的 8 位模式选择信号 psn[7..0] (左) 和 16 位帧选择信号 FN[15..0] 分别传入各个 Pattern

图 3.4.4 展示了模式选择信号 (psn) 和帧选择信号 (FN) 如何传入各个 Pattern. 帧选择信号会统一传入所有 Pattern, 而 psn 的每位会传入对应的 Pattern. 当某位 psn 为低电平, 其对应的 Pattern 就会被选中; 此时该 Pattern 会根据 FN 来输出对应的帧.

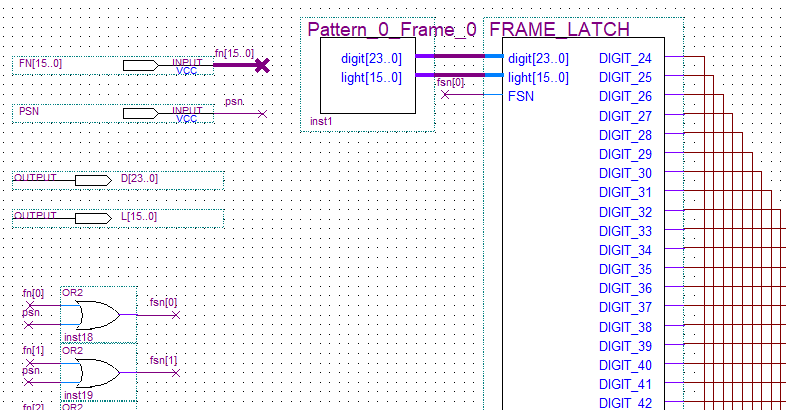


图 3.4.5. Pattern\_0 内部的部分展示

图 3.4.5 展示了 Pattern\_0 内部的部分设计. 每个Pattern 内部设有 16 个可更改的 Frame 元件, 并且每个 Frame 元件后接有一个 FRAME\_LATCH 作为输出锁存, 具体原理可以参考 3.3.3 节和 3.3.5 节.

输入的 FN 信号与模式选择信号 psn 或操作后传入对应帧的 FRAME\_LATCH; 只有模式被选中 (psn 为低电平) 且帧被选中 (fsn 的某位为低电平), FRAME\_LATCH 的右端才会允许输出, 将 FRAME 元件的预设值打入总线; 否则 FRAME\_LATCH 的输出端将为高阻态, 不影响总线上的已有数据.

# 波形模拟

## 16 帧为周期循环播放

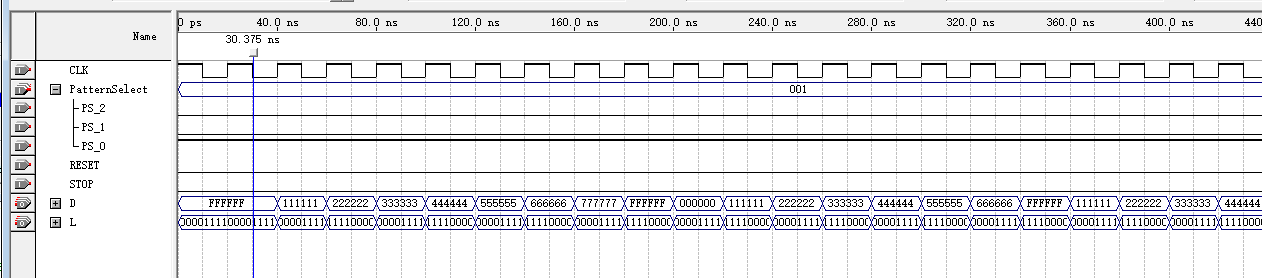


图 4.1.1. Pattern\_1 的波形模拟

从图 4.1.1 中可以看到, 在 RESET 和 STOP 都为低电平, 且没有切换显示模式的情况下, Pattern\_1 的输出以 16 帧为周期循环播放.

## 切换显示模式

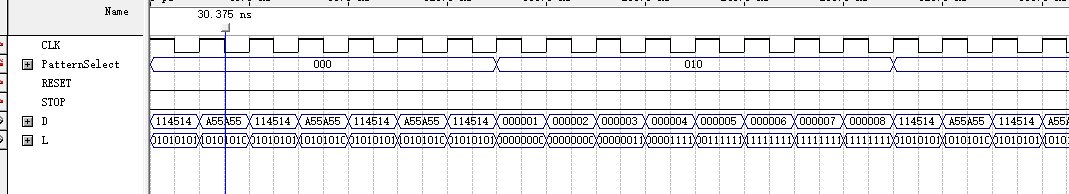


图 4.2.1. 初始模式 000, 切换到模式 010, 再切换回 000

图 4.2.1 展示了切换模式情况下的波形的模拟. 可以看到, 当模式从 Pattern\_0 切换为 Pattern\_2 后, 输出如期从 Pattern\_2 的第一帧 (Frame 0) 开始; 从 Pattern\_2 切换回 Pattern\_0 时也同样从 Pattern\_0 第一帧开始播放.

## 暂停显示

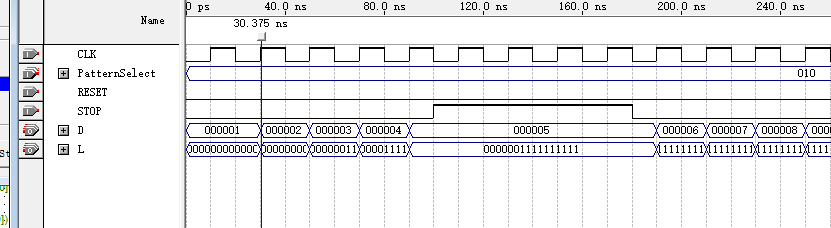


图 4.3.1. Pattern\_2 在第 5 帧 STOP 置高电平

图 4.3.1 展示了当 STOP 置为高电平出现的情况. 可以看见, 输出维持在第 5 帧, 当 STOP 拨回低电平后, 继续从第 6 帧开始输出.

## 重置

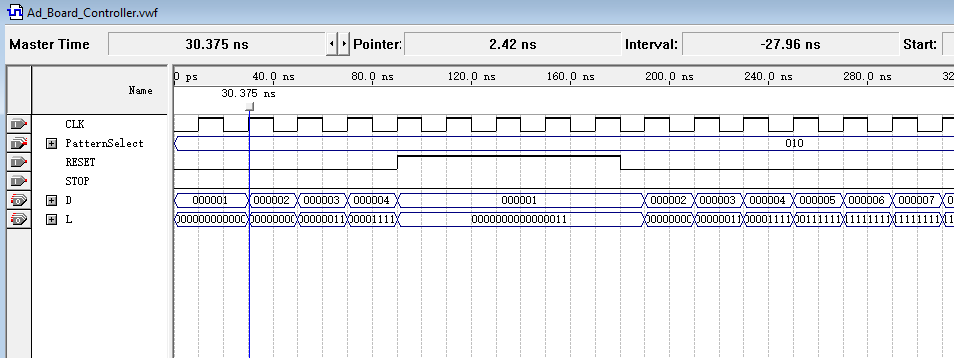


图 4.4.1. Pattern\_2 在第 5 帧 RESET 置高电平

从图 4.4.1 可以看出, 当 Pattern\_2 在第五帧 RESET 置高电平后, 输出维持在第一帧; 当 RESET 拨回低电平, 输出从第一帧开始.

## 实机演示

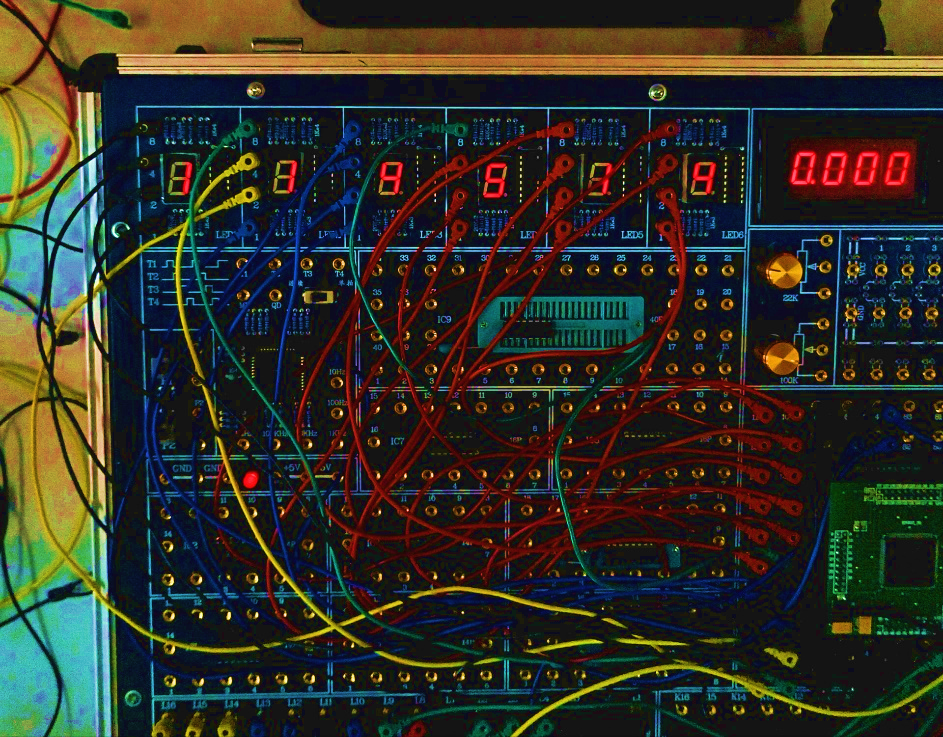


图 4.5.1. 实验箱上的对 Pattern\_0 的演示

图 4.5.1. 展示了试验箱上对 Pattern\_0 的演示, 可以发现当 STOP 和 RESET 都为低电平, 数码管如期交替显示 114514 和 A55. 但是我们没有预料到这个数码管是 8421 数码管, 无法显示 ABCEDF! 导致在试验箱上当显示内容为字母是, 数码管没有显示内容.

# 优化方案

## 利用三态门代替 LATCH (74LS373)

在设计电路时, 我们希望避免总线写入冲突, 因此使用了 LATCH. 但是在搜索资料时, 发现网上的很多帖子说应该尽量避免使用 LATCH, 因为它会使时序电路分析变得非常困难. 当我们在进行波形模拟时, 也发现 LATCH 会使输出信号产生很多 “毛刺”.

当项目验收完后, 我们继续研究了相关知识, 发现避免总线写入冲突不一定要使用 LATCH. 本质上说, 我们只要在每个输出引脚后添加一个三态门就能解决问题.

## 利用存储芯片替代 Frame 元件

拿到报告的题目时, 我们第一个想到的就是对时间帧和模式选择进行编址, 然后传入一个存储芯片; 根据地址输出. 我们要求的输出位数是 40 位; 因此如果存储芯片的位数不够, 可以进行位扩展.

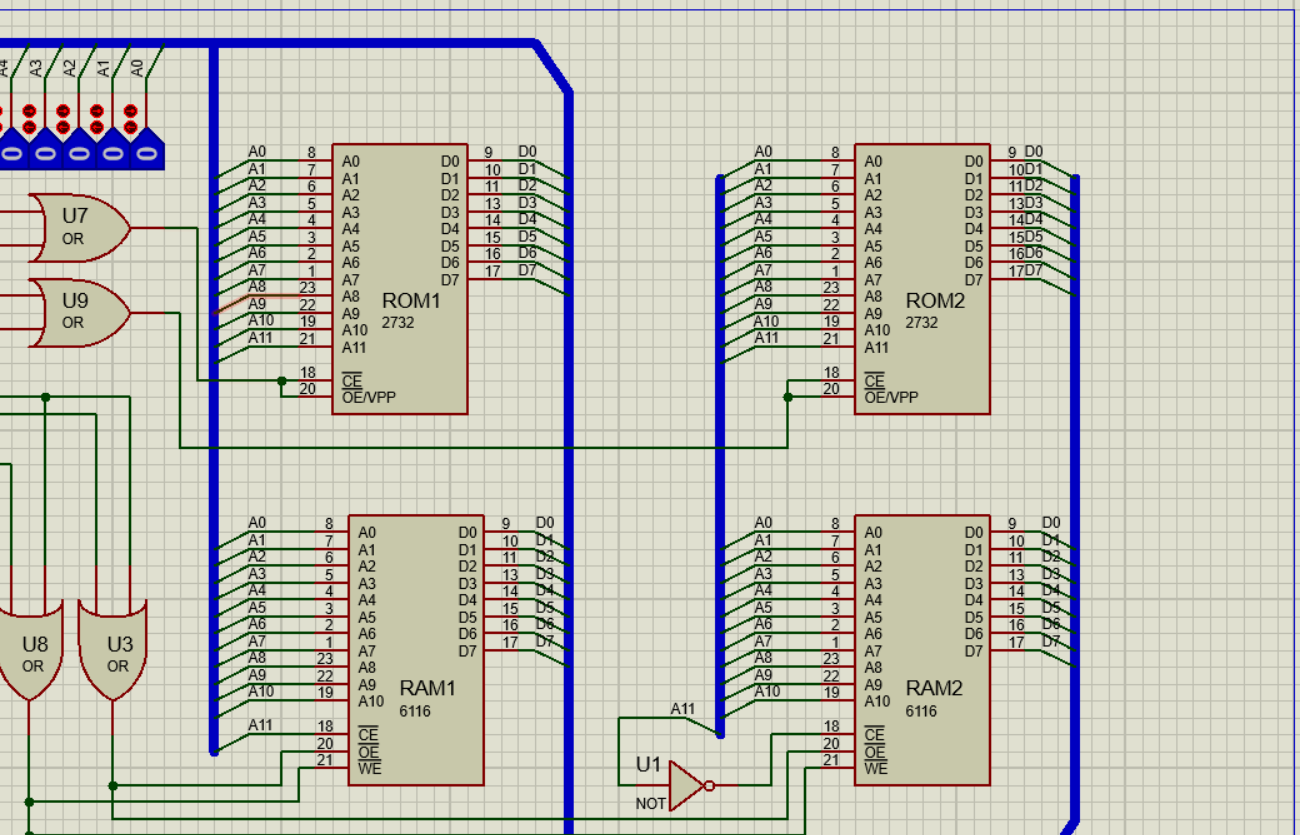


图 5.2.1. Proteus 软件中对存储器进行字位扩栈

图 5.2.1 展示了在 Proteus 中对存储器进行字位扩展. 这是我上学期计算机组成原理做的项目. 我们知道, 时间信号有 4 位, 模式选择信号有 3 位, 因此我们只要编一个 7 位的地址传入存储器即可.

但是, Quartus 软件中使用存储器实在太麻烦了, 我了解到通信的同学的硬件课都使用 Proteus. 实际使用经验告诉我 Proteus 比 Quartus 好操作太多了. 希望学校可以更新系统, 今后直接使用 Proteus 上硬件课.

# 项目小结

本次实践项目, 我们根据项目内容, 利用已有的芯片, 如: 74LS138、74LS161、74LS373、D触发器等等, 以及根据实现和分析电路而设计出的自制元件, 如上文介绍. 我们在软件QuartusII中利用上述芯片和元件, 合理分析与设计数字电路, 设计了八种不同方式的广告灯展示方式. 该项目包含八种广告灯的展示方式, 以及暂停模式、重置模式两种功能. 我们利用《数字逻辑》课程上所学的知识, 成功并完美的对理论知识进行了实践, 深刻感受到了学习硬件课程的意义, 为我们今后的计算机课程学习打下了夯实的基础.