



## 本科实验报告

课程名称: 数字逻辑电路设计

姓 名: NoughtQ

学 院: 计算机科学与技术学院

专 业: 计算机科学与技术

邮 箱:

QQ 号:

电 话:

指导教师: 洪奇军

报告日期: 2024 年 4 月 18 日

# 浙江大学实验报告

课程名称: 数字逻辑设计 实验类型: 综合

实验项目名称: 变量译码器的设计和应用

学生姓名: 钱梓洋 学号: 3230103502 同组学生姓名: 官欣

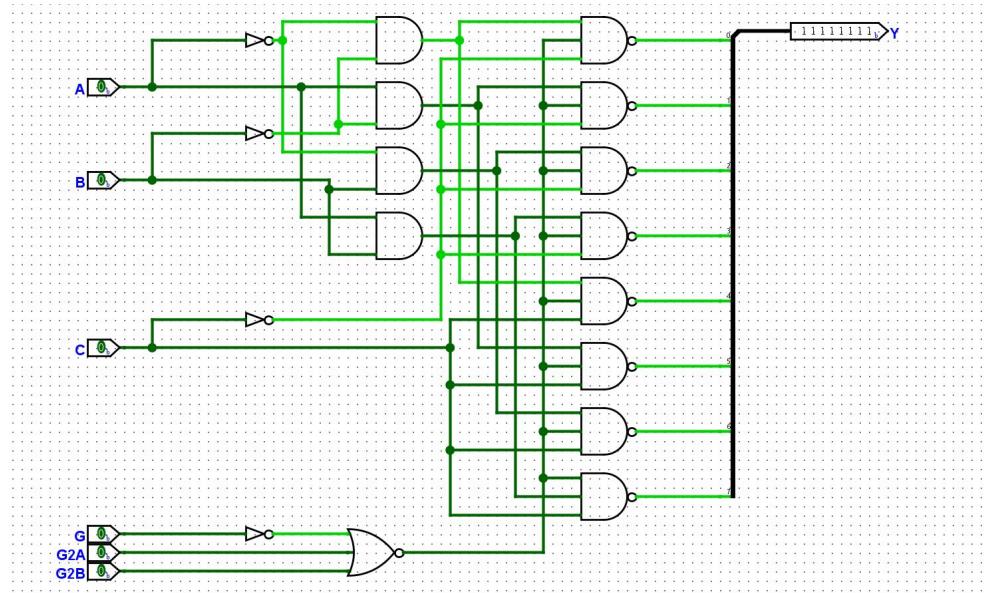
实验地点: 紫金港东四 511 室 实验日期: 2024 年 3 月 28 日

## 一、操作方法与实验步骤

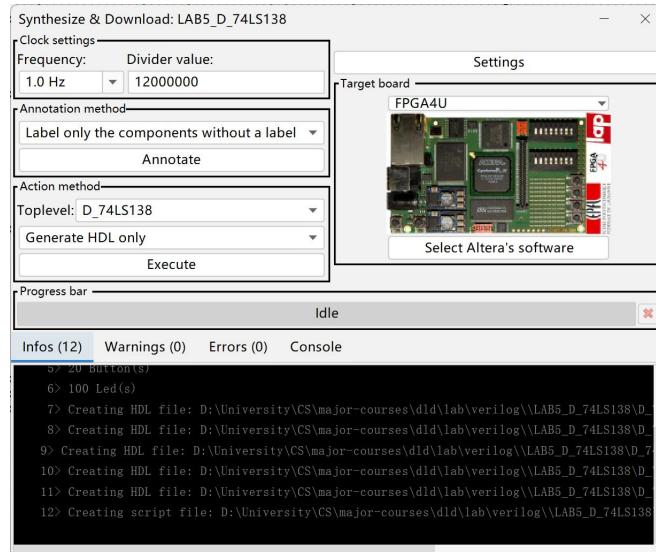
### 1. 原理图设计实现 74LS138 译码器模块

(1) 在 Logisim 中新建项目 LAB5\_D\_74LS138, 然后将其导出为 Verilog。  
注意点:

- ①将 Circuit Name 修改为 D\_74LS138
- ②本例中, 存在有多条线的信号, 需要用分流器汇集多条线的信号
- ③输出数据位数需改成 8 位

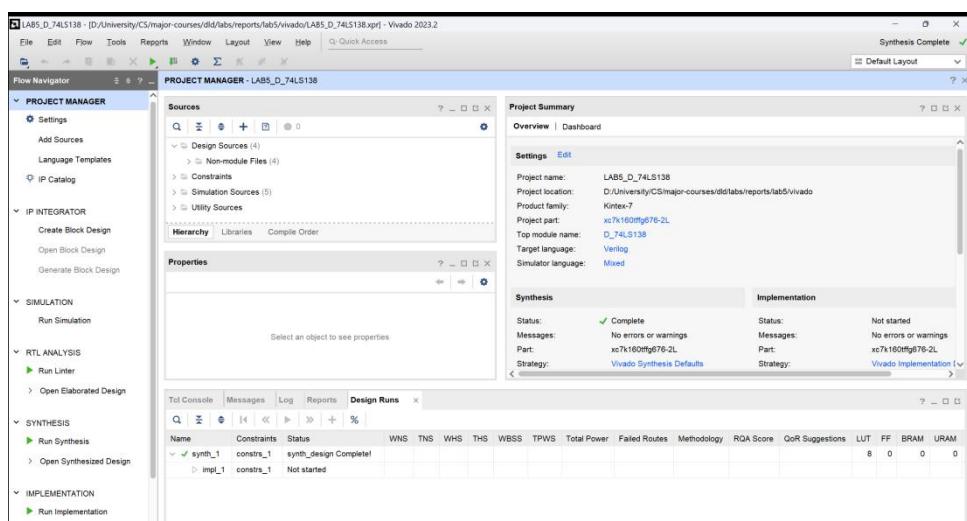
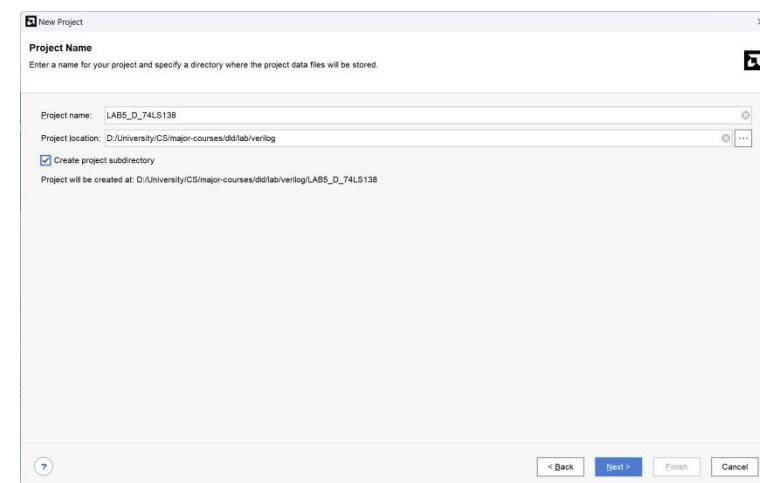


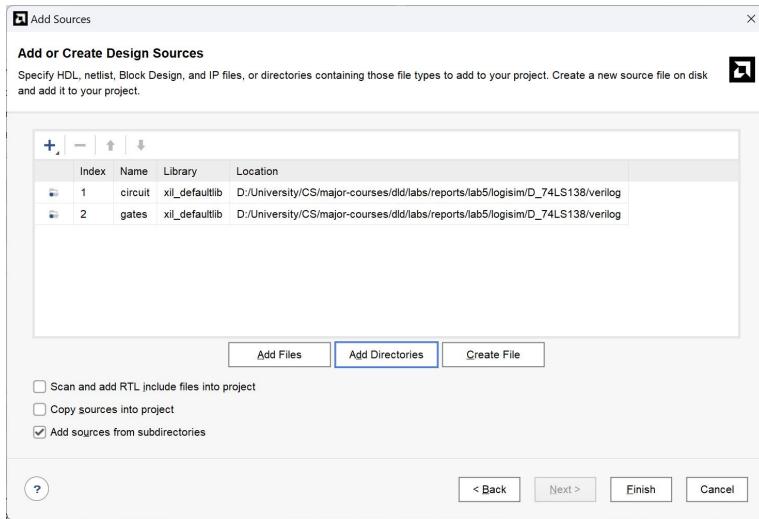
电路图 1



导出为 Verilog

(2) 在 Vivado 中新建第一个工程 LAB5\_D\_74LS138，并在工程里添加转化为 Verilog 的电路图的目录内的 verilog/circuit 和 verilog/gates 这两个目录，然后综合运行（Run Synthesis）。





(3) 新建仿真文件，输入仿真代码，得到波形图。

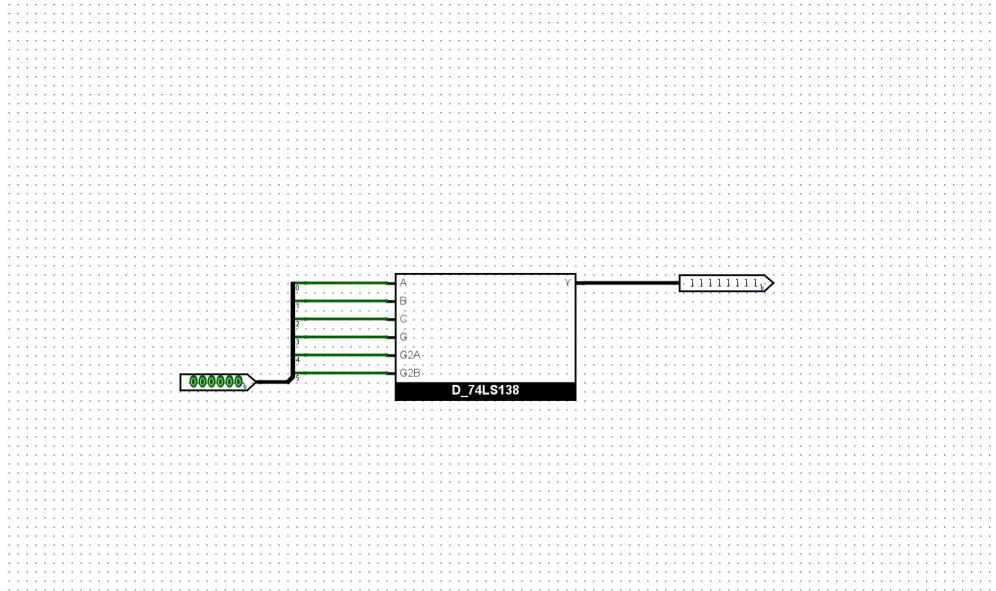
```
1. module D_74LS138_TB();
2.     reg A;
3.     reg B;
4.     reg C;
5.     reg G;
6.     reg G2A;
7.     reg G2B;
8.     wire [7:0] Y;
9.     D_74LS138 u1(A,B,C,G,G2A,G2B,Y);
10.    integer i;
11.    initial begin
12.        A=0;B=0;C=0;G=1;G2A=0;G2B=0;#100;
13.        for (i=0;i<=7;i=i+1) begin {C,B,A}=i; #100; end
14.        G=0;G2A=0;G2B=0;#100;
15.        G=1;G2A=1;G2B=0;#100;
16.        G=1;G2A=0;G2B=1;#100;
17.    end
18.    endmodule
```

仿真代码 1



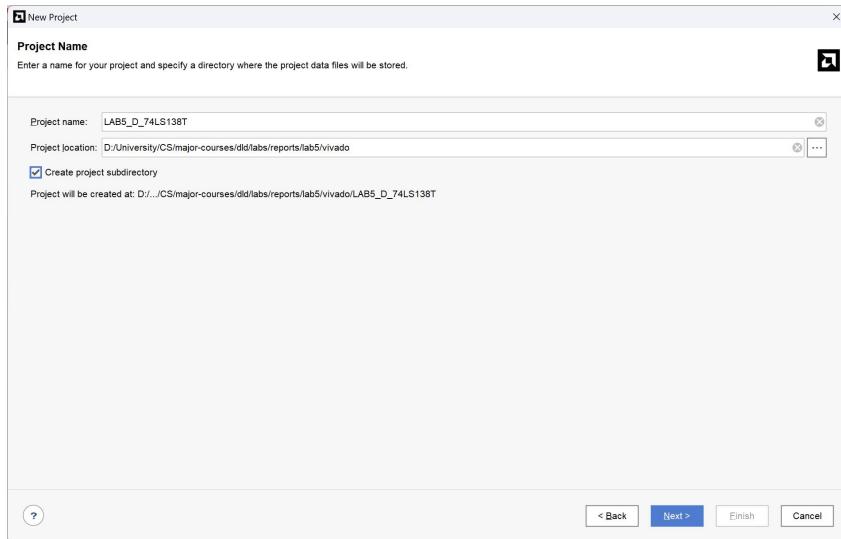
波形图 1

(4) 回到 Logisim，在原来的项目上新建电路图 D\_74LS138\_TEST，进入该电路图后点击 D\_74LS138，获得封装好的元件，然后搭建如图所示的电路，将其导出为 Verilog。



电路图 2

(5) 在 Vivado 上新建第二个工程 LAB5\_D\_74LS138T，并在工程里添加转化为 Verilog 的 D\_74LS138\_TEST 电路图的目录内的 verilog/circuit 和 verilog/gates 这两个目录，然后综合运行（Run Synthesis）。



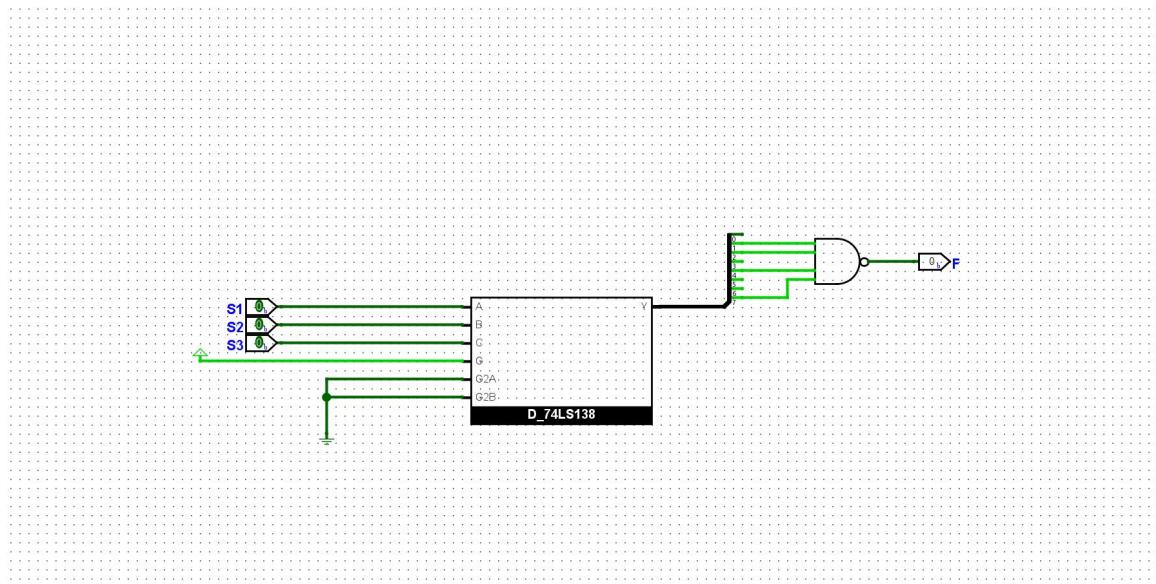
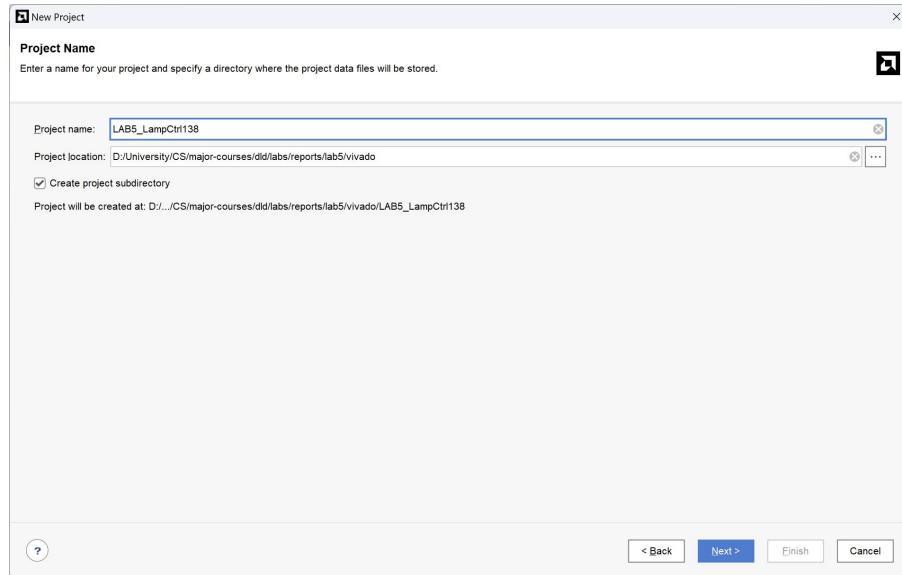
(6) 新建引脚约束文件，输入代码，然后烧录成 bit 文件，导入到 sword 板中，对照真值表，验证 D\_74LS138 译码器的逻辑功能。

1. set\_property IOSTANDARD LVCMOS15 [get\_ports SW[5]]
2. set\_property PACKAGE\_PIN Y12 [get\_ports SW[5]]
3. set\_property IOSTANDARD LVCMOS15 [get\_ports SW[4]]
4. set\_property PACKAGE\_PIN Y13 [get\_ports SW[4]]
5. set\_property IOSTANDARD LVCMOS15 [get\_ports SW[3]]
6. set\_property PACKAGE\_PIN AA12 [get\_ports SW[3]]
7. set\_property IOSTANDARD LVCMOS15 [get\_ports SW[2]]
8. set\_property PACKAGE\_PIN AA13 [get\_ports SW[2]]
9. set\_property IOSTANDARD LVCMOS15 [get\_ports SW[1]]
10. set\_property PACKAGE\_PIN AB10 [get\_ports SW[1]]
11. set\_property IOSTANDARD LVCMOS15 [get\_ports SW[0]]
12. set\_property PACKAGE\_PIN AA10 [get\_ports SW[0]]
13. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[7]]
14. set\_property PACKAGE\_PIN AF24 [get\_ports LED[7]]
15. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[6]]
16. set\_property PACKAGE\_PIN AE21 [get\_ports LED[6]]
17. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[5]]
18. set\_property PACKAGE\_PIN Y22 [get\_ports LED[5]]
19. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[4]]
20. set\_property PACKAGE\_PIN Y23 [get\_ports LED[4]]
21. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[3]]
22. set\_property PACKAGE\_PIN AA23 [get\_ports LED[3]]
23. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[2]]
24. set\_property PACKAGE\_PIN Y25 [get\_ports LED[2]]
25. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[1]]
26. set\_property PACKAGE\_PIN AB26 [get\_ports LED[1]]
27. set\_property IOSTANDARD LVCMOS33 [get\_ports LED[0]]
28. set\_property PACKAGE\_PIN W23 [get\_ports LED[0]]

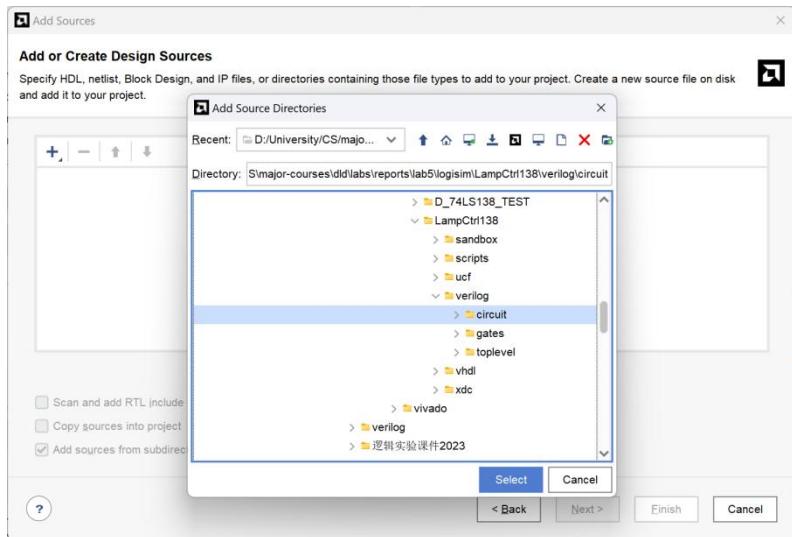
## 引脚约束代码 1

## 2. 用 74LS138 译码器实现楼道灯控制器

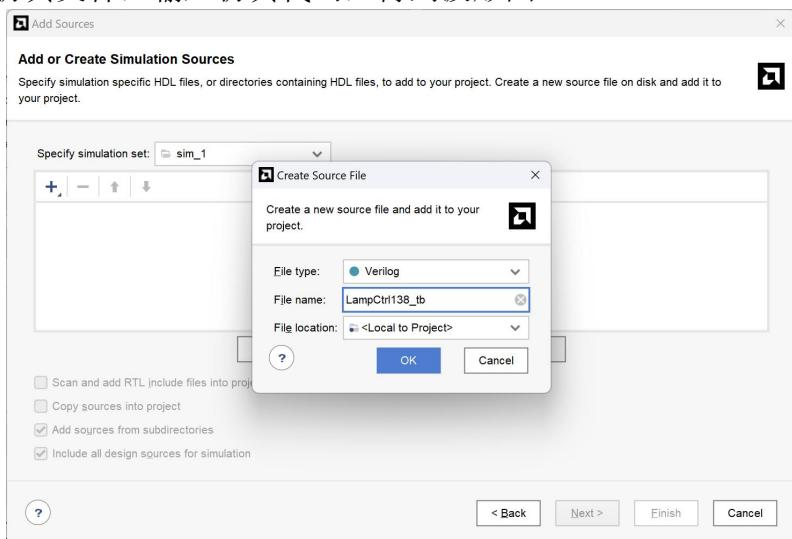
(1) 先在 Vivado 新建第三个工程 LAB5\_LampCtrl138。与实验 1 类似，在 Logisim 的项目中绘制如图所示的电路图，导出为 Verilog，然后导入到工程中。



电路图 3



(2) 添加仿真文件，输入仿真代码，得到波形图。

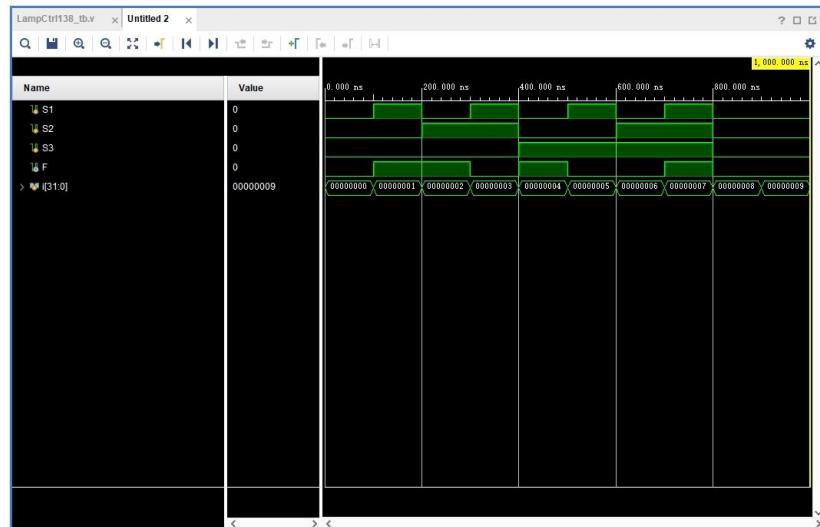


```

1. module LampCtrl138_tb();
2.   reg S1;
3.   reg S2;
4.   reg S3;
5.   wire F;
6.   integer i;
7.   LampCtrl138 lamp1(F,S1,S2,S3);
8.   initial begin
9.     for (i=0;i<=8;i=i+1)
10.      begin {S3,S2,S1}=i;#100; end
11.   end
12. endmodule

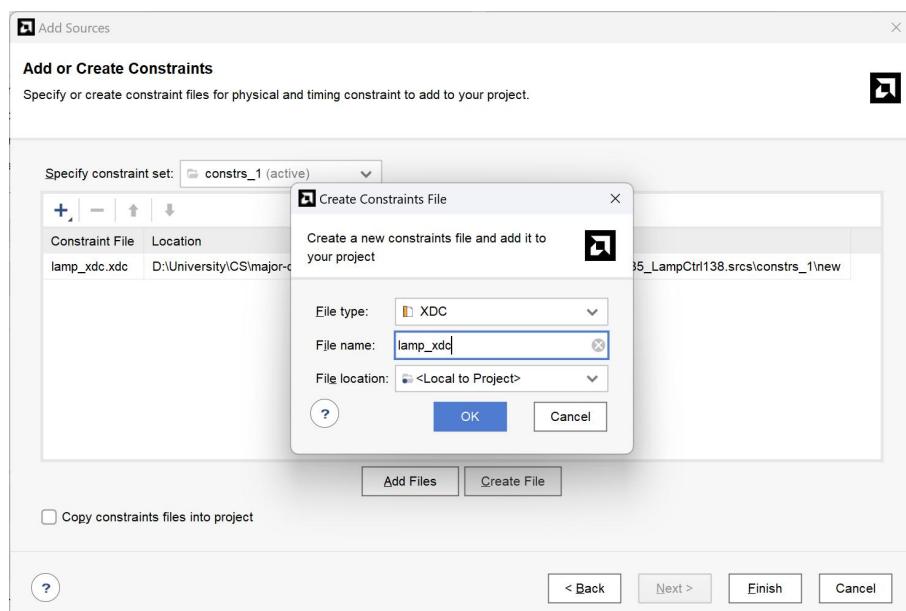
```

仿真代码 2



波形图 2

(3) 添加引脚约束文件，输入代码，然后烧录成 bit 文件，导入到 sword 板中，进行功能测试。



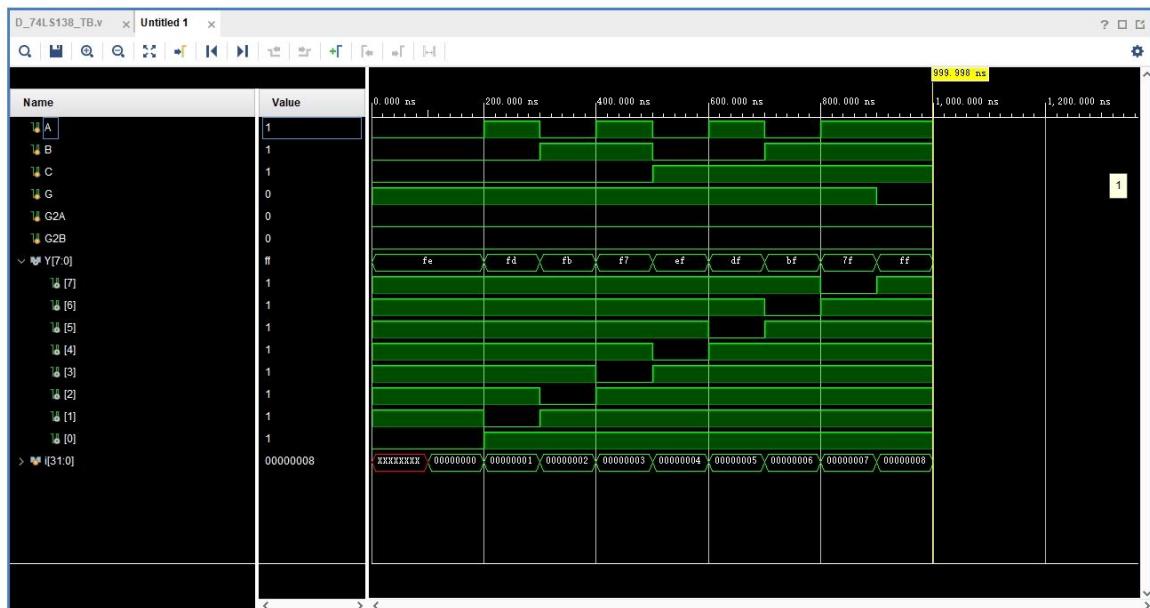
1. set\_property IOSTANDARD LVCMOS15 [get\_ports S1]
2. set\_property PACKAGE\_PIN AA10 [get\_ports S1]
- 3.
4. set\_property IOSTANDARD LVCMOS15 [get\_ports S2]
5. set\_property PACKAGE\_PIN AB10 [get\_ports S2]
- 6.
7. set\_property IOSTANDARD LVCMOS15 [get\_ports S3]
8. set\_property PACKAGE\_PIN AA13 [get\_ports S3]
- 9.
10. set\_property IOSTANDARD LVCMOS15 [get\_ports F]
11. set\_property PACKAGE\_PIN AF24 [get\_ports F]

引脚约束代码 2

## 二、实验结果与分析

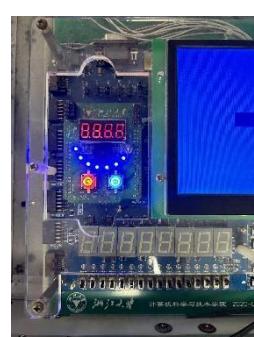
## 1. 原理图设计实现 74LS138 译码器模块

### (1) 译码器的仿真结果



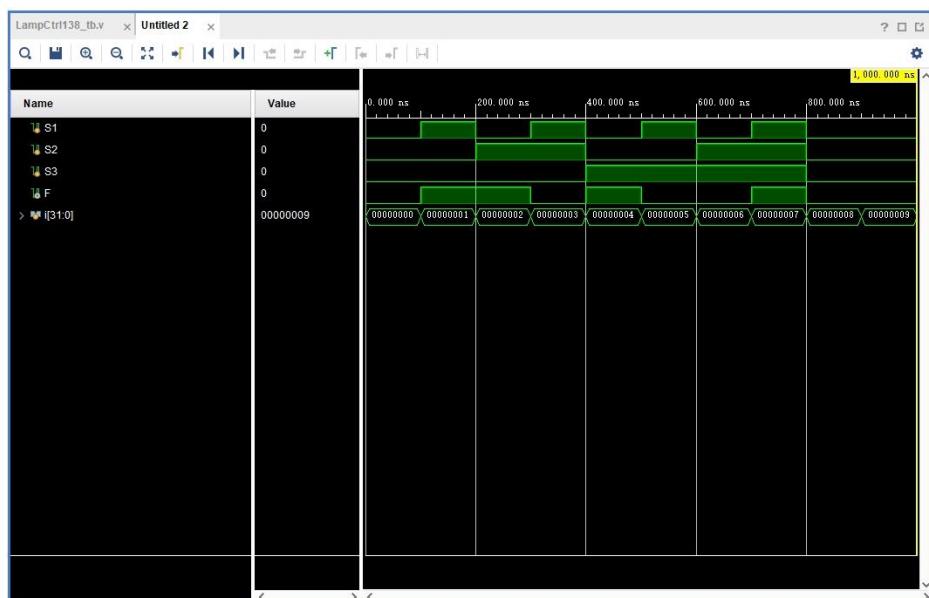
将波形图与下列真值表作比较，可以确定译码器功能成功实现。

## (2) bit 文件在 sword 板上运行的结果

			
CBA = 000 1 号灯熄灭	CBA = 001 2 号灯熄灭	CBA = 010 3 号灯熄灭	CBA = 011 4 号灯熄灭
			
CBA = 100 5 号灯熄灭	CBA = 101 6 号灯熄灭	CBA = 110 7 号灯熄灭	CBA = 111 8 号灯熄灭

## 2. 用 74LS138 译码器实现楼道灯控制器

### (1) 楼道灯的仿真结果

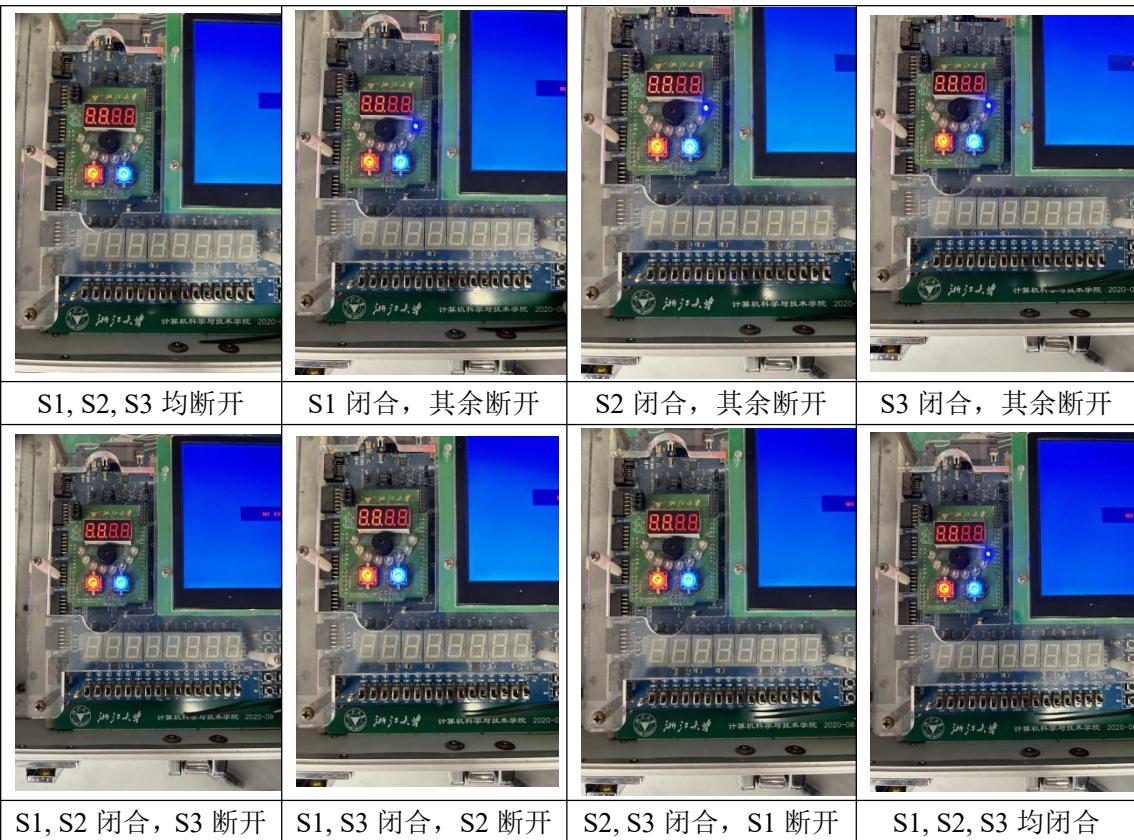


根据波形图，绘制以下真值表：

S1	S2	S3	F
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

不难发现，当一个开关闭合或三个开关同时闭合时，灯亮；两个开关闭合或开关都断开时，灯灭。这符合楼道灯的逻辑。

(2) bit 文件在 sword 板上运行的结果



注：楼道灯是八个灯中最右边那盏

### 三、讨论、心得

上次的实验只是带领我们熟悉 logisim 和 vivado 这两个工具，但这次实验我们就需要用这两个工具搭建电路了。在本次实验中，我认识了 logisim 中的分流

器，它可以将多个信号汇总到一起去；同时我对 verilog 语法有了一定粗浅的了解，虽然难以讲清细节，但我能大致判断这些语句能完成什么任务。

好在上课前稍微预习了一下，整个实验还算完成得比较顺利，有点超出我的预料。但在实验过程中，我还是遇到了一些困难，比如实验一在运行仿真的时候，刚开始它运行的不是我后来添加的仿真文件，导致出来的波形图是错的；但后来被我瞎弄一通后，竟然让它成功运行了正确的仿真文件，我不清楚中间发生了什么事；课后在自己电脑上运行实验室电脑里创建的 vivado 文件，发现由于版本不一样，我的电脑好像无法正常运行 vivado 文件，但我不清楚原因。

通过这次实验，我意识到我还尚未完全掌握好 vivado 工具和 verilog 语法，因此课后我得自学一番；我还得花时间研究一下如何在新版 vivado 上运行旧版 vivado 里创建的文件，这样方便我在自己电脑上先配置好。

# 浙江大学实验报告

课程名称: 数字逻辑设计 实验类型: 综合

实验项目名称: 七段数码管显示译码器设计与应用

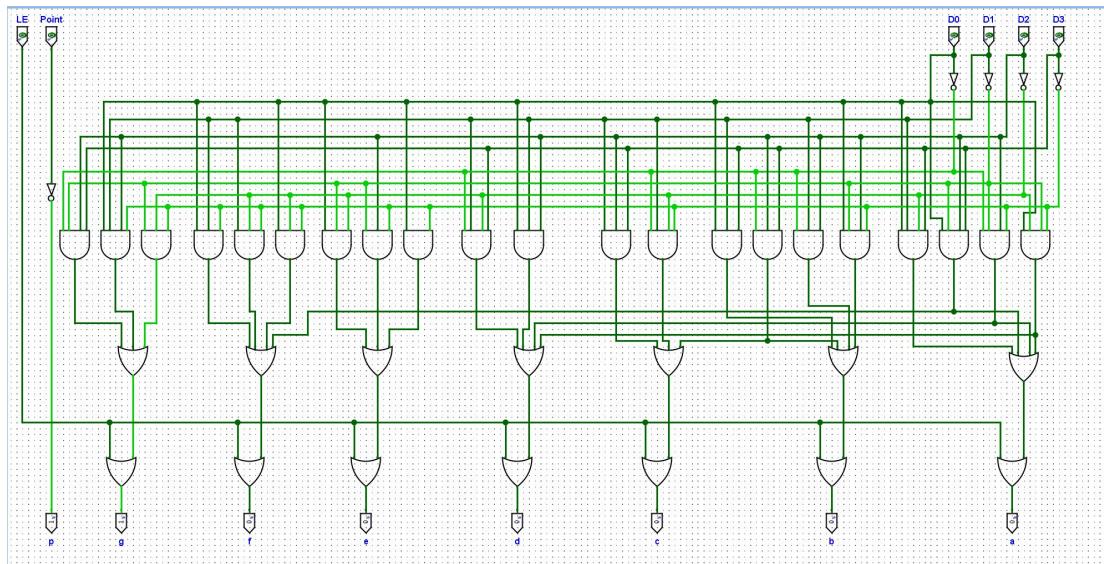
学生姓名: 钱梓洋 学号: 3230103502 同组学生姓名: 官欣

实验地点: 紫金港东四 511 室 实验日期: 2024 年 4 月 11 日

## 一、操作方法与实验步骤

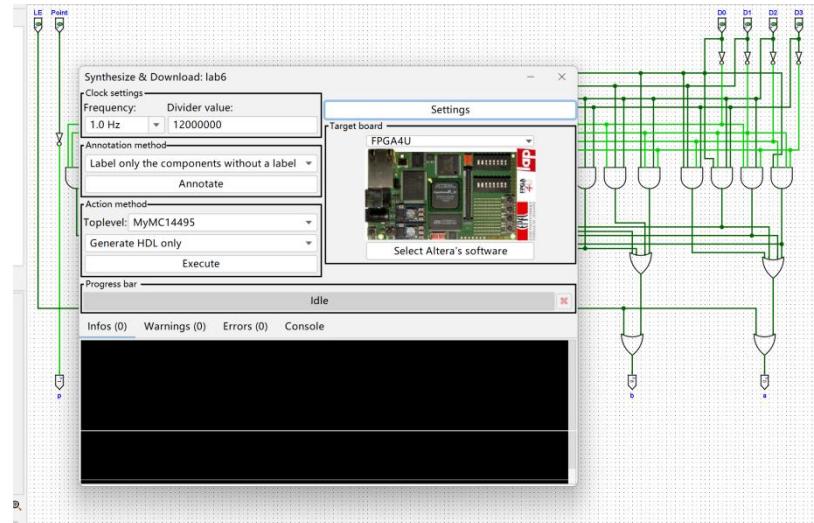
### 1. 原理图设计实现译码器 MyMC14495 模块

(1) 在 Logisim 软件中新建项目 LAB6\_MyMC14495，并新建电路图 MyMC14495，逻辑电路图如下所示：

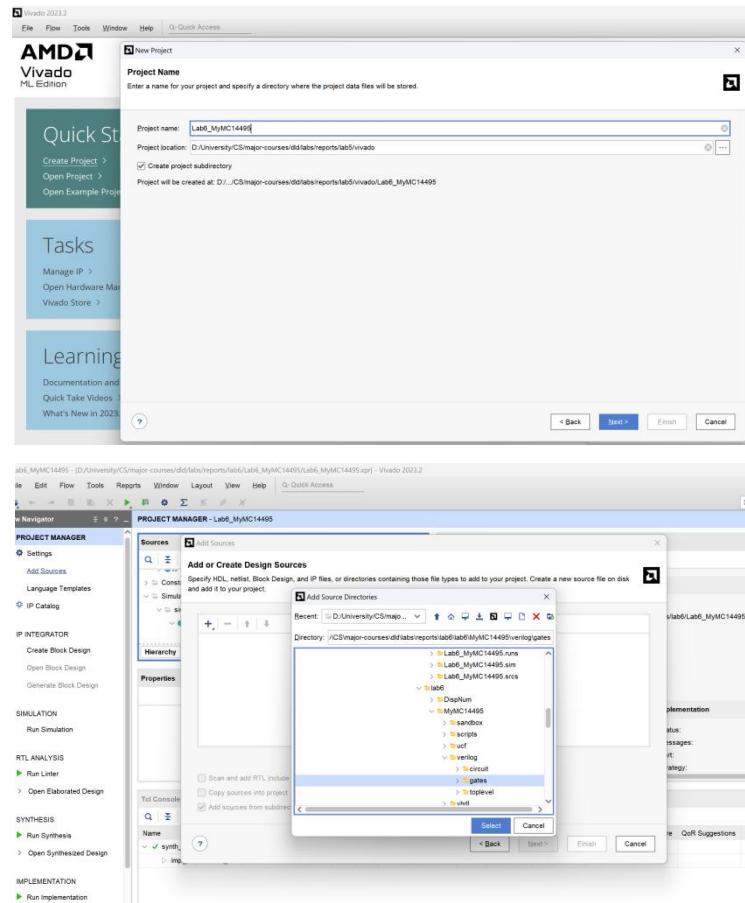


电路图 1

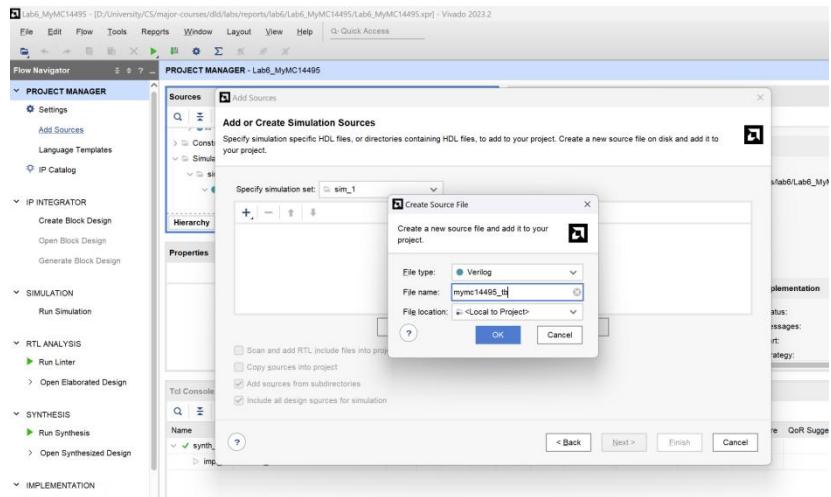
(2) 然后将该电路图转换成 Verilog 代码。



(3) 在 Vivado 上新建名为 Lab6\_MyMC14495 的工程，并将上一步得到的 Verilog 代码中的 circuit 和 gates 两个目录导入工程中，然后点击“Run Synthesis”进行综合。



(4) 新建仿真文件，并输入以下代码，然后点击“Run Simulation”，生成仿真波形，并检查波形图是否符合预期。



```

1. `timescale 1ns / 1ps
2. module MyMC14495_MyMC14495_sch_tb();
3. // Inputs
4. reg D3;
5. reg D2;
6. reg D0;
7. reg D1;
8. reg LE;
9. reg Point;
10. // Output
11. wire a;
12. wire b;
13. wire c;
14. wire d;
15. wire e;
16. wire f;
17. wire g;
18. wire p;
19. // Instantiate the UUT
20. MyMC14495 UUT (
21. .D3(D3),
22. .D2(D2),
23. .D0(D0),
24. .a(a),
25. .b(b),
26. .c(c),
27. .d(d),
28. .e(e),
29. .f(f),
30. .D1(D1),
31. .g(g),

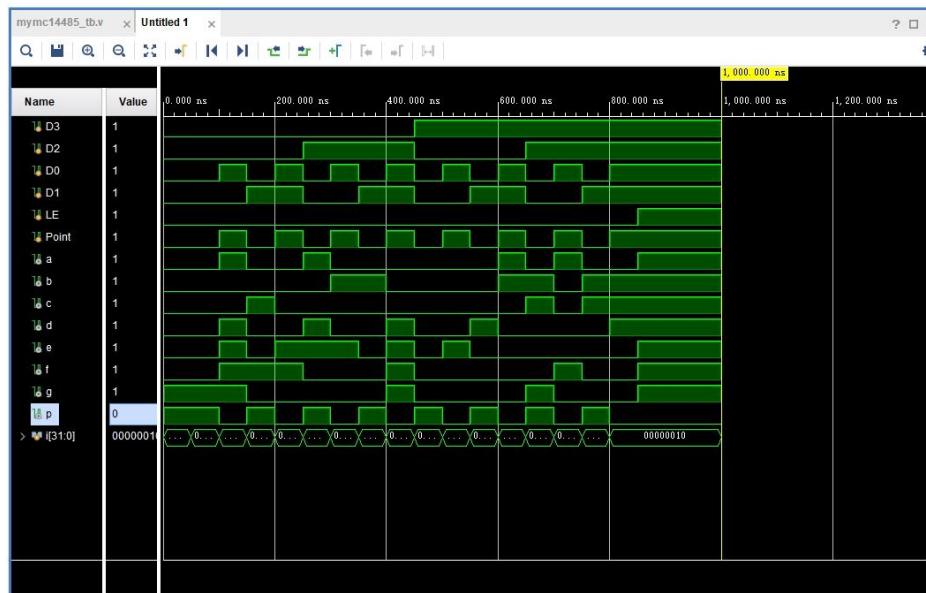
```

```

32.    .LE(LE),
33.    .Point(Point),
34.    .p(p)
35.    );
36.    // Initialize Inputs
37.    integer i;
38.    initial begin
39.        D3 = 0;
40.        D2 = 0;
41.        D1 = 0;
42.        D0 = 0;
43.        LE = 0;
44.        Point = 0;
45.        for(i = 0; i <= 15; i = i+1)begin
46.            #50;
47.            {D3,D2,D1,D0} = i;
48.            Point = i;
49.        end
50.        #50
51.        LE = 1;
52.    end
53.    endmodule

```

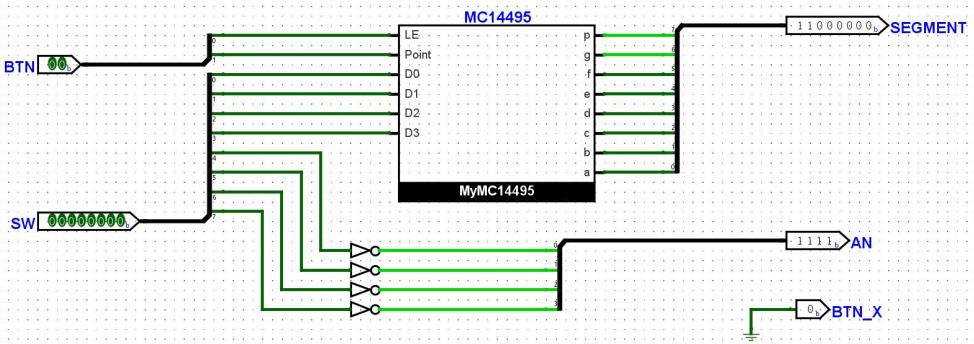
仿真代码



波形图

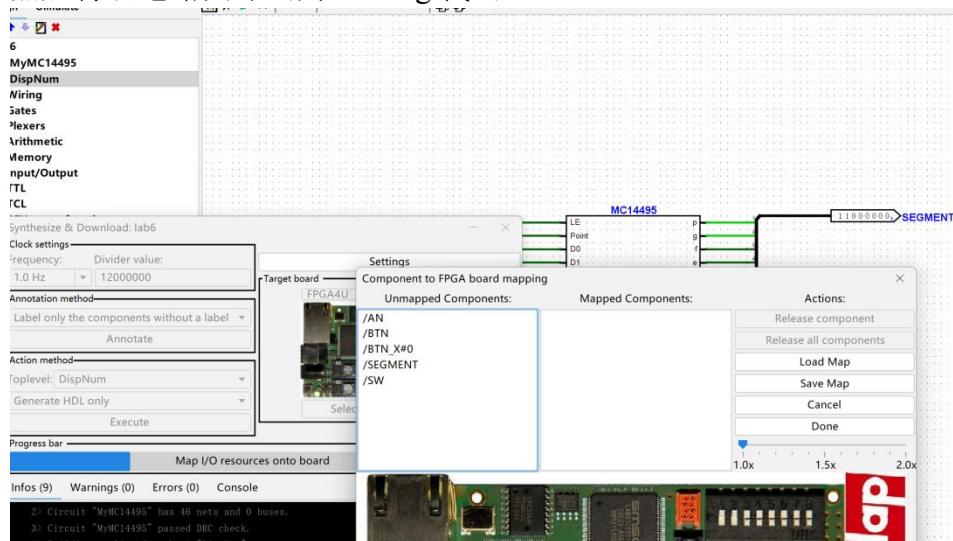
## 2. 用 MyMc14495 模块实现数码管显示

(1) 在第一部分创建的 Logisim 项目中新建电路图 DispNumber\_sch, 绘制如图所示的电路图:

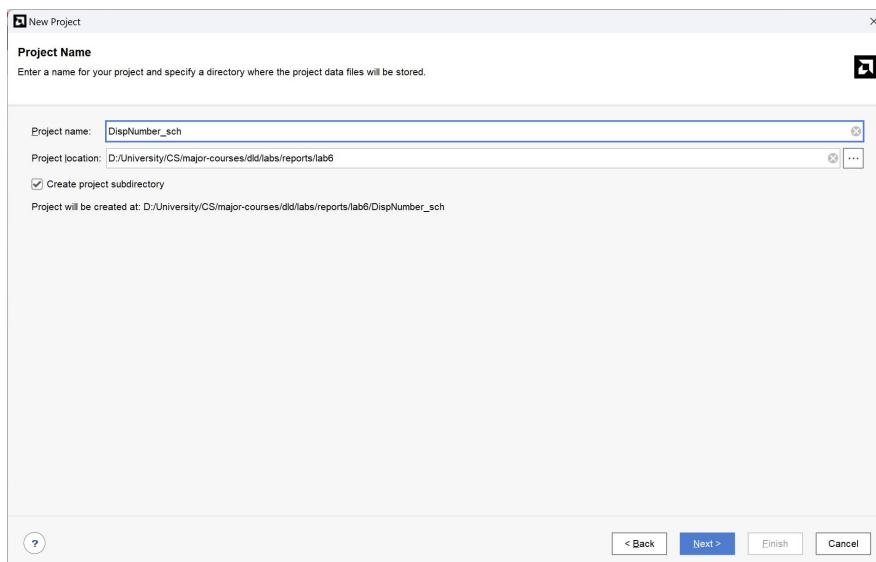


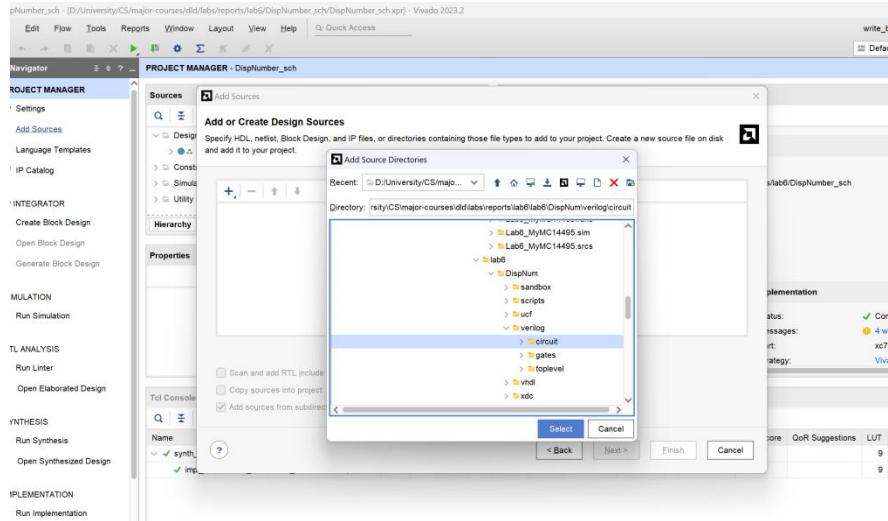
电路图 2

(2) 然后将该电路图导出为 Verilog 代码

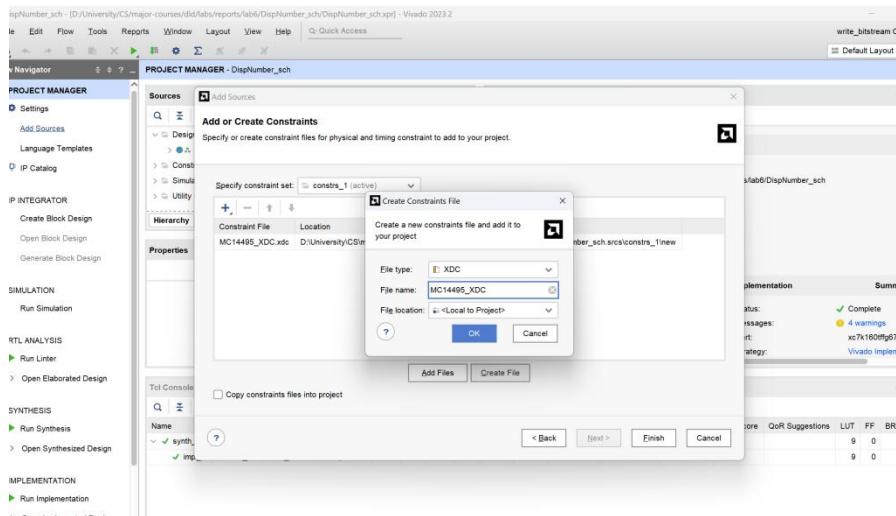


(3) 在 Vivado 上新建名为 DispNumber\_sch 的工程，并将上一步得到的 Verilog 代码中的 circuit 和 gates 两个目录导入工程中，然后点击“Run Synthesis”进行综合。





(4) 新建引脚约束文件，并输入以下代码。然后烧录成 bit 流文件，并下载至实验箱，观察是否按照正常逻辑运行。



1. *## Constraints file for Lab6*
- 2.
3. *# Switches as inputs*
4. *set\_property PACKAGE\_PIN AA10 [get\_ports {SW[0]}]*
5. *set\_property PACKAGE\_PIN AB10 [get\_ports {SW[1]}]*
6. *set\_property PACKAGE\_PIN AA13 [get\_ports {SW[2]}]*
7. *set\_property PACKAGE\_PIN AA12 [get\_ports {SW[3]}]*
8. *set\_property PACKAGE\_PIN Y13 [get\_ports {SW[4]}]*
9. *set\_property PACKAGE\_PIN Y12 [get\_ports {SW[5]}]*
10. *set\_property PACKAGE\_PIN AD11 [get\_ports {SW[6]}]*
11. *set\_property PACKAGE\_PIN AD10 [get\_ports {SW[7]}]*
12. *set\_property IOSTANDARD LVCMS15 [get\_ports {SW[0]}]*
13. *set\_property IOSTANDARD LVCMS15 [get\_ports {SW[1]}]*
14. *set\_property IOSTANDARD LVCMS15 [get\_ports {SW[2]}]*
15. *set\_property IOSTANDARD LVCMS15 [get\_ports {SW[3]}]*

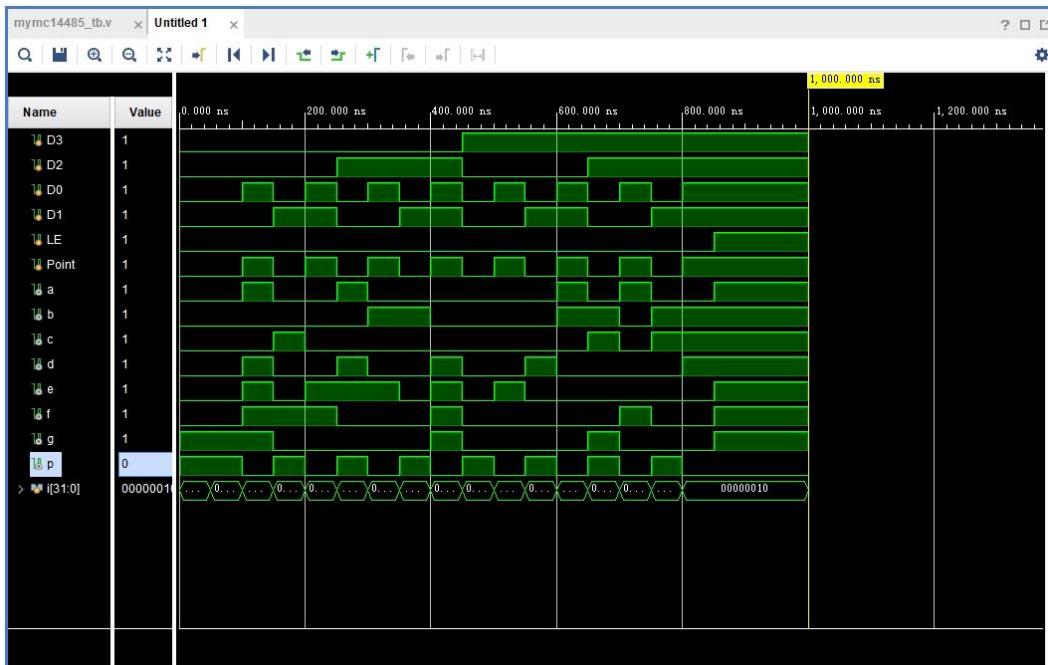
```
16. set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
17. set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
18. set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
19. set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
20.
21. # Key as inputs
22. set_property PACKAGE_PIN AE10 [get_ports {BTN[0]}]
23. set_property IOSTANDARD LVCMOS15 [get_ports {BTN[0]}]
24. set_property PACKAGE_PIN AE12 [get_ports {BTN[1]}]
25. set_property IOSTANDARD LVCMOS15 [get_ports {BTN[1]}]
26. set_property PACKAGE_PIN V17 [get_ports {BTN_X}]
27. set_property IOSTANDARD LVCMOS18 [get_ports {BTN_X}]
28.
29. # Arduino-Segment & AN
30. set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
31. set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
32. set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
33. set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
34. set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
35. set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
36. set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
37. set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
38. set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
39. set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
40. set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
41. set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
42. set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
43. set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
44. set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
45. set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
46. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
47. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
48. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
49. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
50. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
51. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
52. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
53. set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
```

引脚约束文件

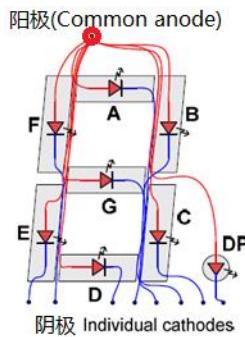
## 二、实验结果与分析

### 1. 原理图设计实现显示译码器 MyMC14495 模块

仿真波形分析：



波形图



七段数码管显示原理图

我们将仿真激励波形图与七段数码管显示原理图进行比对、分析后，不难发现仿真结果完全正确，说明 MC14495 原理图绘制正确。

## 2. 用 MyMC14495 模块实现数码管显示

BTN	SW[7:4]	SW[3:0]	效果图	说明
10	0001	0001		有小数点出现
00	0001	0001		没有小数点
01	0001	0001		使能信号高电平，输出无效
00	1111	0000		SW[7:4]能正确控制数码管的亮灭

00	0010	0010		
00	0100	0011		
00	1000	0100		
00	0011	0101		

00	0101	0110		
00	1001	0111		
00	0110	1000		
00	1010	1001		

00	1100	1010		
00	0111	1011		
00	1011	1100		
00	1101	1101		

00	1110	1110		
00	1111	1111		

### 三、讨论、心得

幸亏在上课前稍微预习了一下，我因此才发现这次的电路图绘制有些复杂，所以我提前绘制好了电路图。本以为接下来会完成地比较轻松，结果刚上来第一个实验的仿真代码有问题——后来发现是名称上的误差：我在 Logisim 上给小数点的输入为命名为 Point，而课件上命名为 point。然而尽管我发现了这一问题，但我没注意整段代码中出现了好几个 point，而我只发现了部分，幸好小组的同学帮我找出了所有未修改的 point。解决完这一问题后，波形图顺利绘制出来了。

第二个实验问题更多：因为我们忘记要想将 bit 流文件传输至实验箱，还需要点击“Program Device”，因此我们以为烧录好 bit 流文件，就相当于将代码传输至实验箱内。结果我们在实验箱上搞了半天，只能看到相同的奇怪的结果（可能是之前小组的实验吧）。刚开始我还以为是我原理图画错了，但仔细比对后发现并没有什么问题。最终是老师提醒我们要点击那个“Program Device”，我们才顿时茅塞顿开，发现问题所在。除此之外，我们按助教的文档编写引脚约束文件，没有注意到它的 BTN 对应的引脚是数码管右边的那堆按钮。还好我们在旁边听到老师的讲解，于是及时将 BTN 的引脚对应至数码管下面的开关，这才没有发生奇怪的错误。

这次实验反映出了很多问题，我从中得到了一些教训：注意自己的命名与课件或文档的命名是否一致，否则会发生错误；熟记 Vivado 软件使用流程，不可错过一个环节，这是基本功；要学会仔细阅读引脚约束文件，看看对应的引脚是

否是我要的。

# 浙江大学实验报告

课程名称: 数字逻辑设计 实验类型: 综合

实验项目名称: 多路选择器的设计和应用

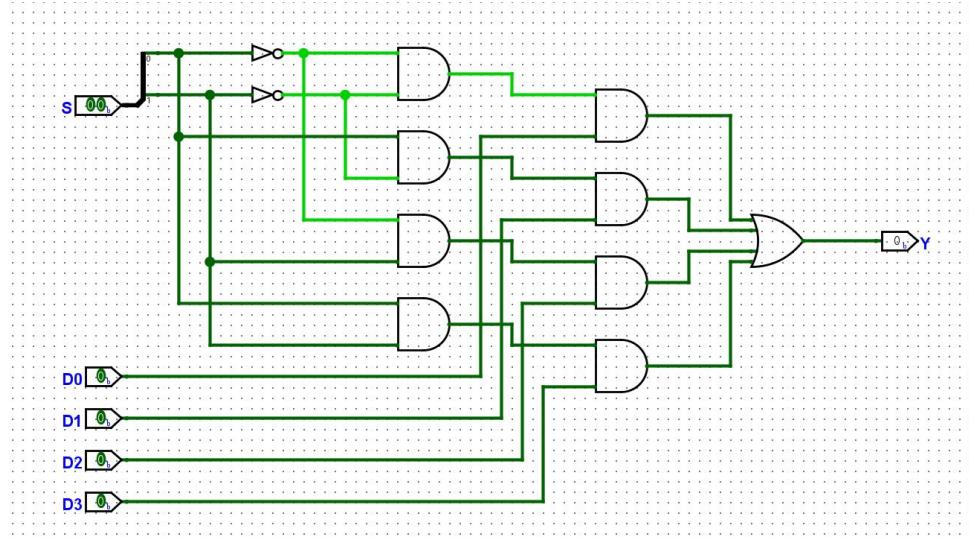
学生姓名: 钱梓洋 学号: 3230103502 同组学生姓名: 官欣

实验地点: 紫金港东四 511 室 实验日期: 2024 年 4 月 18 日

## 一、操作方法与实验步骤

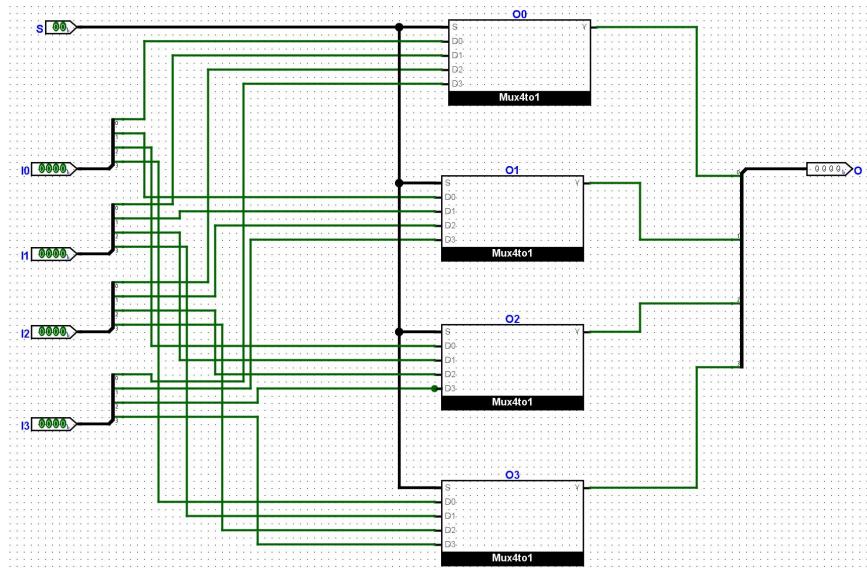
### 1. 数据选择器设计

(1) 绘制 1 位 4-1 多路选择器原理图: 在 Logisim 上新建项目 lab7, 然后在里面新建电路图 Mux4to1, 并绘制如下电路图。



电路图 1

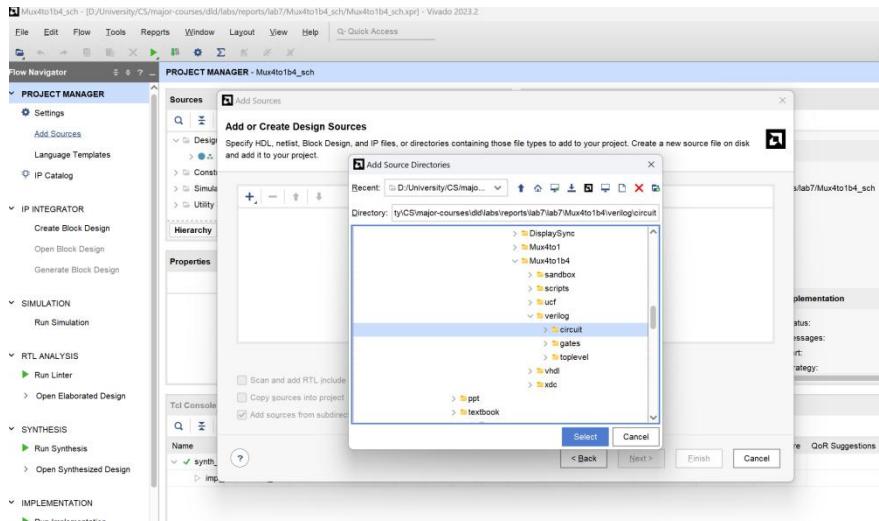
(2) 绘制 4 位 4-1 多路选择器原理图: 在刚刚创建的项目中再新建电路图 Mux4to1b4, 并绘制如下电路图。注意这里可以利用第一步创建的 Mux4to1, 以此简化电路图的绘制。



电路图 2

(3) 将 Mux4to1b4 电路图导出为 Verilog 代码。

(4) 在 Vivado 上新建项目 Mux4to1b4，将刚刚生成的 Verilog 代码导入到该项目中。



(5) 添加以下仿真代码，然后综合运行(Run Synthesis)，最后进行仿真运行(Run Simulation)，得到波形图。

```

1. `timescale 1ns / 1ps
2.
3. module mux4to14b_sim();
4. // Inputs
5. reg [1:0] S;
6. reg [3:0] I0;
7. reg [3:0] I1;
8. reg [3:0] I2;
9. reg [3:0] I3;
10. // Output
11. wire [3:0] O;

```

```

12.
13.     Mux4to1b4 test(
14.         .S(S),
15.         .I0(I0),
16.         .I1(I1),
17.         .I2(I2),
18.         .I3(I3),
19.         .O(O)
20.     );
21. // Initialize Inputs
22.     integer i, j;
23.     initial begin
24.         S=0;
25.         I0=0;
26.         I1=0;
27.         I2=0;
28.         I3=0;
29.         for(j=0;j<=3;j=j+1)begin
30.             S=j;
31.             I0=0;
32.             I1=0;
33.             I2=0;
34.             I3=0;
35.             for(i=0;i<=15;i=i+1)begin
36.                 #50;
37.                 if(j==0) I0 = i;
38.                 else if(j==1) I1 = i;
39.                 else if(j==2) I2 = i;
40.                 else if(j==3) I3 = i;
41.             end
42.         end
43.     end
44. endmodule

```

## 2.计分板设计

(1) 在 Vivado 上新建项目 ScoreBoard, 然后添加 CreateNumber 计分模块的 Verilog 代码 (如下所示)。

```

1. module CreateNumber(
2.     input [3:0]      btn,
3.     output reg [15:0] num
4. );
5.

```

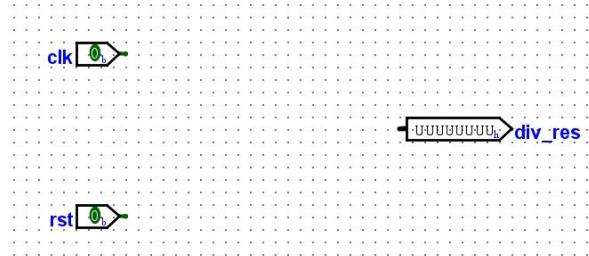
```

6.     wire [3:0] A, B, C, D;
7.
8.     initial num <= 16'b1010_1011_1100_1101;
9.
10.    assign A = num[15:12] + 4'b1;
11.    assign B = num[11: 8] + 4'b1;
12.    assign C = num[ 7: 4] + 4'b1;
13.    assign D = num[ 3: 0] + 4'b1;
14.
15.   always @(posedge btn[0]) num[15:12] <= A;
16.   always @(posedge btn[1]) num[11: 8] <= B;
17.   always @(posedge btn[2]) num[ 7: 4] <= C;
18.   always @(posedge btn[3]) num[ 3: 0] <= D;
19.
20. endmodule

```

(2) 绘制时钟分频器原理图：时钟分频器的输出在每个时钟信号上升沿自增。复位信号为同步复位，当时钟信号的正边沿到来且复位信号为有效时（本实验中复位信号为高电平有效）进行复位。

在 Logisim 新建电路图 clkdiv，并绘制如下电路图(注意只要给出端口 clk, rst, clkdiv[31:0]即可，不需要做其他连接)：

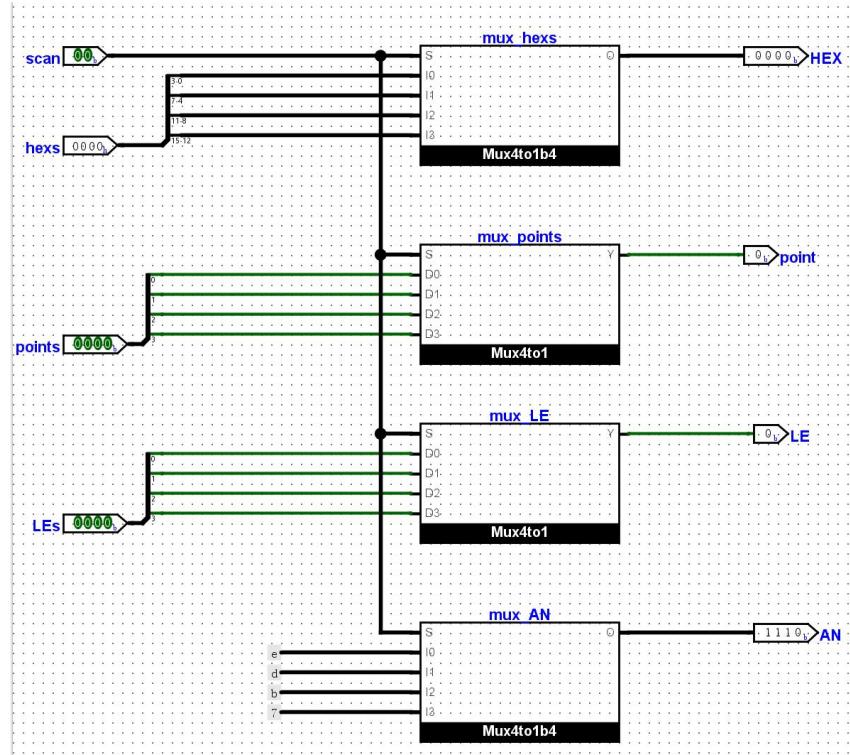


电路图 3

(3) 绘制 DisplaySync 模块原理图：本模块为动态扫描模块的子模块，起作用在于实现数字与使能信号的选择逻辑。

在 Logisim 新建电路图 DisplaySync，并绘制如下电路图：

(注意：这里需要利用之前创建的 Mux4to1 和 Mux4to1b4 电路图；这里的输入还用到了 Constant 模块)

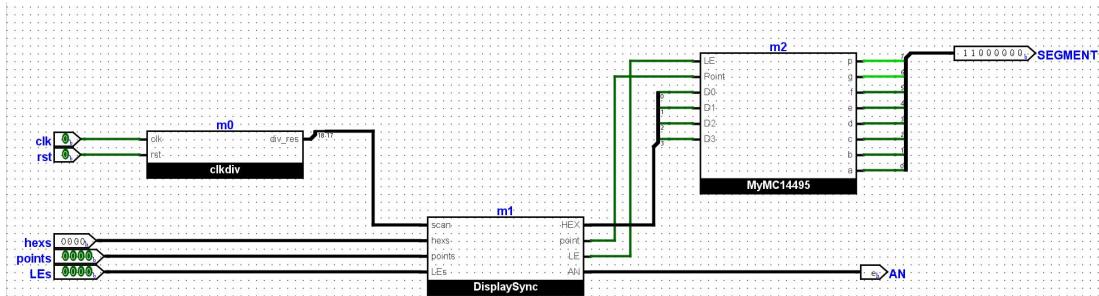


电路图 4

(4) 绘制 DisplayNumber 模块原理图：动态扫描模块，需要使用之前完成的 clkdiv, DisplaySync, MyMC14495(lab6 中设计的) 模块。

使用时钟分频器获得合适的扫描信号，这里我们选用 clkdiv[18:17] 两位信号。使用刚刚编写完成的 DisplaySync 模块选择合适的数字和使能信号，并将四位数字和小数点控制信号输出到 MyMC14495 模块实例中得到七段信息。

在 Logisim 上新建电路图 DisplayNumber，并绘制如下电路图：



电路图 5

(5) 将 DisplayNumber 电路图导出为 Verilog 代码，并导入到 Vivado 的 ScoreBoard 项目中。

(6) 顶层模块的设计：作为顶层模块，将使用之前完成的所有模块作为子模块。实现一个“计分板”应用，在 Arduino 上七段数码管查看数字结果，使用开关控制亮灭与小数点，使用四个按钮为数字实现自增。

在 Vivado 中创建 Verilog 代码 top.v，输入如下代码，然后将 top.v 作为顶层模块。

```
1. module top(
2.     input clk,
3.     input [7:0] SW,
4.     input [3:0] btn,
5.     output[3:0] AN,
6.     output[7:0] SEGMENT,
7.     output    BTN_X
8. );
9.
10.    assign BTN_X = 1'b0;
11.
12.    wire [15:0] num;
13.
14.    CreateNumber create_inst(
15.        .btn(btn),
16.        .num(num)
17.    );
18.
19.    DisplayNumber disp_inst(
20.        .clk(clk),
21.        .rst(1'b0),
22.        .hexs(num),
23.        .points(SW[7:4]),
24.        .LEs(SW[3:0]),
25.        .AN(AN),
26.        .SEGMENT(SEGMENT)
27.    );
28.
29. endmodule
```



top.v 左边的三点表示该文件已设为顶层文件

(7) 添加引脚约束文件，代码如下。然后进行烧录，生成 bit 流文件，将其

导入到 sword 板上进行实验。

```
1.      # Main clock
2.      set_property PACKAGE_PIN AC18 [get_ports clk]
3.      set_property IOSTANDARD LVCMS18 [get_ports clk]
4.
5.      create_clock -period 10.000 -name clk [get_ports "clk"]
6.
7.      # Switches as inputs
8.      set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
9.      set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
10.     set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
11.     set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
12.     set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
13.     set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
14.     set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
15.     set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
16.     set_property IOSTANDARD LVCMS15 [get_ports {SW[0]}]
17.     set_property IOSTANDARD LVCMS15 [get_ports {SW[1]}]
18.     set_property IOSTANDARD LVCMS15 [get_ports {SW[2]}]
19.     set_property IOSTANDARD LVCMS15 [get_ports {SW[3]}]
20.     set_property IOSTANDARD LVCMS15 [get_ports {SW[4]}]
21.     set_property IOSTANDARD LVCMS15 [get_ports {SW[5]}]
22.     set_property IOSTANDARD LVCMS15 [get_ports {SW[6]}]
23.     set_property IOSTANDARD LVCMS15 [get_ports {SW[7]}]
24.
25.      # Key as inputs
26.      set_property PACKAGE_PIN W16 [get_ports BTN_X]
27.      set_property IOSTANDARD LVCMS18 [get_ports BTN_X]
28.      set_property PACKAGE_PIN V18 [get_ports {btn[3]}]
29.      set_property IOSTANDARD LVCMS18 [get_ports {btn[3]}]
30.      set_property PACKAGE_PIN V19 [get_ports {btn[2]}]
31.      set_property IOSTANDARD LVCMS18 [get_ports {btn[2]}]
32.      set_property PACKAGE_PIN V14 [get_ports {btn[1]}]
33.      set_property IOSTANDARD LVCMS18 [get_ports {btn[1]}]
34.      set_property PACKAGE_PIN W14 [get_ports {btn[0]}]
35.      set_property IOSTANDARD LVCMS18 [get_ports {btn[0]}]
36.
37.      set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets btn*]
38.
39.      # Arduino-Segment & AN
40.      set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
41.      set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
42.      set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
43.      set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
```

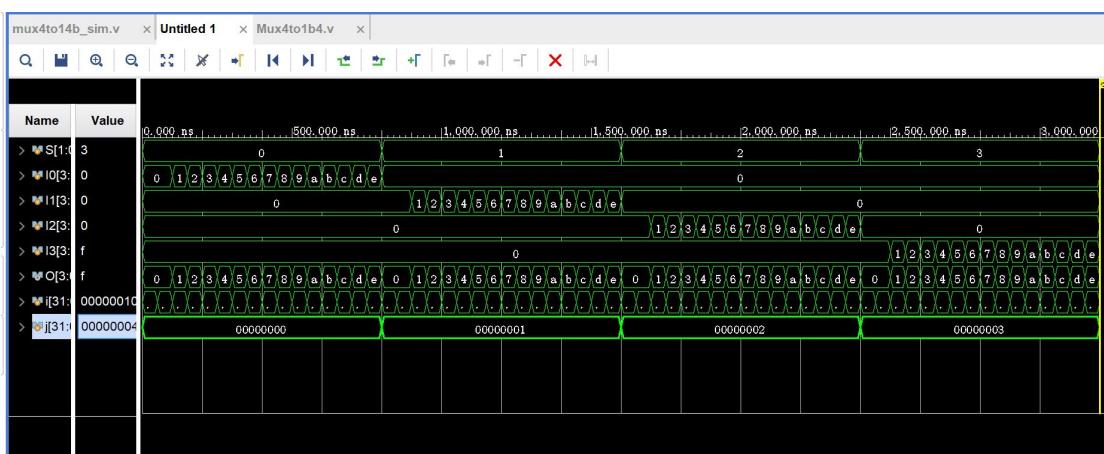
```

44. set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
45. set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
46. set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
47. set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
48. set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
49. set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
50. set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
51. set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
52. set_property IOSTANDARD LVCMS33 [get_ports {AN[0]}]
53. set_property IOSTANDARD LVCMS33 [get_ports {AN[1]}]
54. set_property IOSTANDARD LVCMS33 [get_ports {AN[2]}]
55. set_property IOSTANDARD LVCMS33 [get_ports {AN[3]}]
56. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[0]}]
57. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[1]}]
58. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[2]}]
59. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[3]}]
60. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[4]}]
61. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[5]}]
62. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[6]}]
63. set_property IOSTANDARD LVCMS33 [get_ports {SEGMENT[7]}]

```

## 二、实验结果与分析

### 1. 数据选择器设计

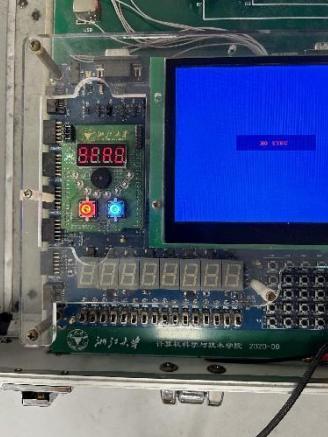


波形图

由于仿真代码里通过 for 循环遍历了所有可能的输入情况，因此该波形图可以直观反映多路选择器的设计是否正确。从图中可以看出，我们成功实现了 4 位 4-1 多路选择器的功能。

## 2. 计分板设计

实验图片	说明
	只显示 2,3 两位
	只显示 1,3 两位
	4 位均显示, 且开启 1,3,4 位上的小数点

	初始状态
	改变第 1 位
	改变第 2 位
	改变第 3 位



改变第 4 位

注：由于按键时手会抖动，故每次按按钮数字可能会增加好几位，这是正常现象。如果想要按 1 次按钮只增加 1 位，需要设计加入防抖动模块。

### 三、讨论、心得

我感觉这次实验相比之前而言，难度上来了不小。首先是多模块的设计：要想实现整个计分板的功能，需要设计好几个模块——Mux4to1b4, DisplaySync, CreateNumber 等等，而且还用到 lab6 设计的 MyMC14495 模块；最后还要使用顶层模块实现整体功能。这让我充分体验到分层设计的思想。

其次，通过这次实验，我对 Vivado 的理解又深了一层。之前我对 Vivado 中三种文件类型，尤其是 design source 和 simulation source 还傻傻分不清。因此，我在第二个小实验中误把 top.v 放入仿真文件部分中，导致烧录 bit 流文件时老是报错；再加上报错信息显示的是引脚约束代码有误，因此我始终没有弄清问题出在哪里。直到老师过来点拨，我才发现自己又犯了愚蠢的错误。经过修正，我们成功实现了计分板的功能。现在，我认识到：仿真文件是用来生成波形图的，与现实的实验板没有任何关系，所以要进行实际的实验的话，要将代码放入 Design Sources 里。

虽然我在课前已经完成了大半部分，就差没有烧录 bit 流文件了，但我们还是花了两节课的时间进行实验，而且主要卡在上面提到的那个问题。由此可见，我对 Vivado 的功能和 Verilog 语法还是不太熟悉。我相信在今后的实验中，我会将这些绊脚石变成我前进道路上的垫脚石，使我更加娴熟地完成实验。