浙江大學

计算机逻辑设计基础

Lab 6

Author: 苏煜程

Student ID: 3220105481

Date: 2023 年 11 月 8 日

浙江大学实验报告

课程名称：计算机逻辑设计基础

实验名称：7 段数码管显示译码器设计与应用

学生姓名：苏煜程　　专业：人工智能（图灵班）　　学号：3220105481

同组学生姓名：张延泽　　指导老师：董亚波

实验地点：东 4-509　　实验日期：2023 年 10 月 26 日

1　实验目的

1. 掌握七数码管显示原理
2. 掌握七段码显示译码设计
3. 进一步熟悉 Xilinx ISE 环境及 SWORD 实验平台

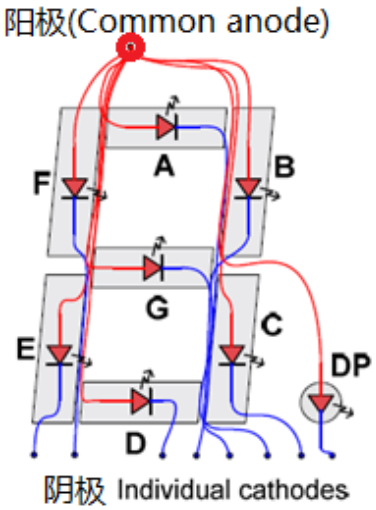
2　实验任务

1. 任务 1：原理图设计实现显示译码 MyMC14495 模块
2. 任务 2：用 MyMC14495 模块实现数码管显示

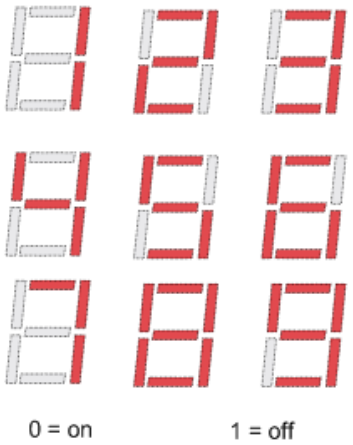
3　实验原理

3.1　七段数码管显示原理

如图，由 7+1 个 LED 构成数字显示器件，每个 LED 显示数字的一段，另一个为小数点。



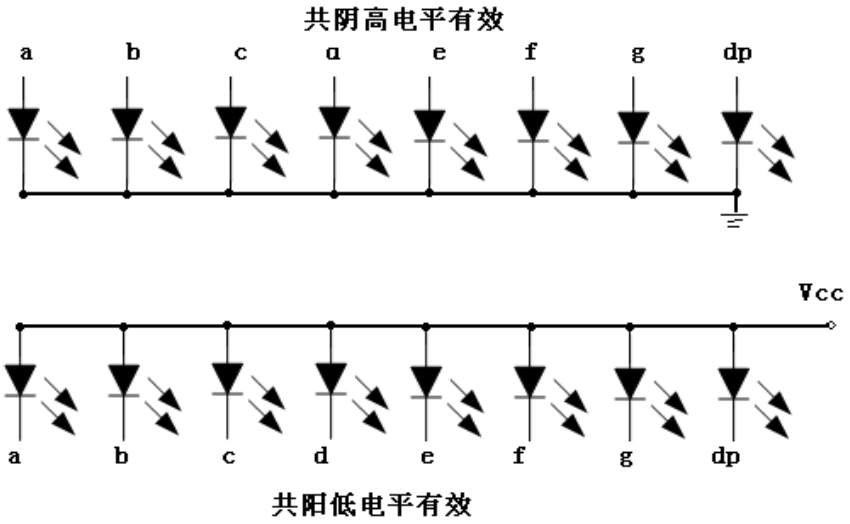
X	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
B	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
D	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0



共阴（阳）控制

LED 的正极 (负极) 连在一起，另一端作为点亮的控制：

- 共阳：正极连在一起，负极 = 0，点亮
- 共阴：负极连在一起，正极 = 1，点亮



Hex 7-segment decoder

接受一个二进制数输入，输出控制七段数码管显示对应十六进制数字。其真值表如下：

Hex	$D_3D_2D_1D_0$	BI/LE	a	b	c	d	e	f	g	p
0	0 0 0 0	0	0	0	0	0	0	0	1	p
1	0 0 0 1	0	1	0	0	1	1	1	1	p
2	0 0 1 0	0	0	0	1	0	0	1	0	p
3	0 0 1 1	0	0	0	0	0	1	1	0	p
4	0 1 0 0	0	1	0	0	1	1	0	0	p
5	0 1 0 1	0	0	1	0	0	1	0	0	p
6	0 1 1 0	0	0	1	0	0	0	0	0	p
7	0 1 1 1	0	0	0	0	1	1	1	1	p
8	1 0 0 0	0	0	0	0	0	0	0	0	P
9	1 0 0 1	0	0	0	0	0	1	0	0	P
A	1 0 1 0	0	0	0	0	1	0	0	0	P
B	1 0 1 1	0	1	1	0	0	0	0	0	P
C	1 1 0 0	0	0	1	1	0	0	0	1	P
D	1 1 0 1	0	1	0	0	0	0	1	0	P
E	1 1 1 0	0	0	1	1	0	0	0	0	P
F	1 1 1 1	0	0	1	1	1	0	0	0	P
X	x x x x	1	1	1	1	1	1	1	1	1

通过卡诺图我们可以简化原式从而得到如下函数：

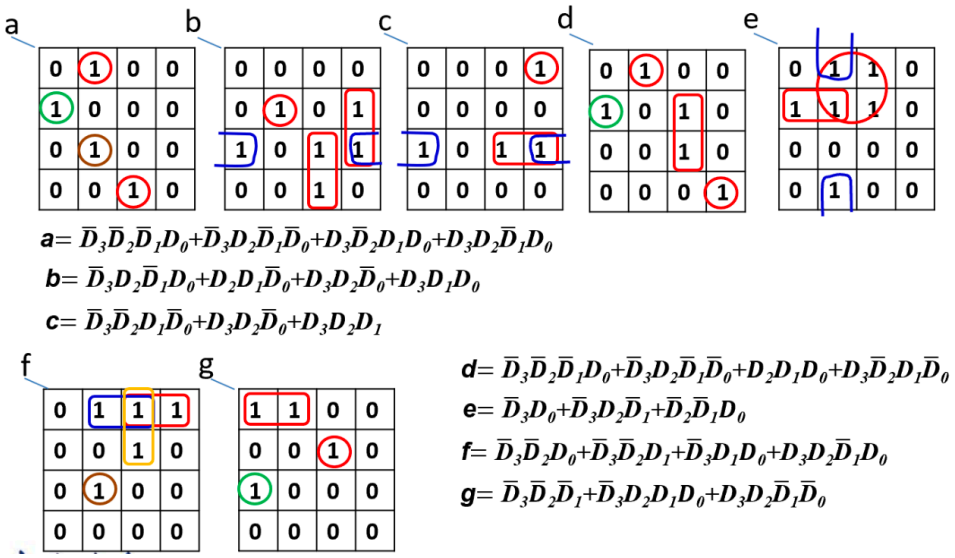


图 2: 通过卡诺图化简

由此得到电路图:

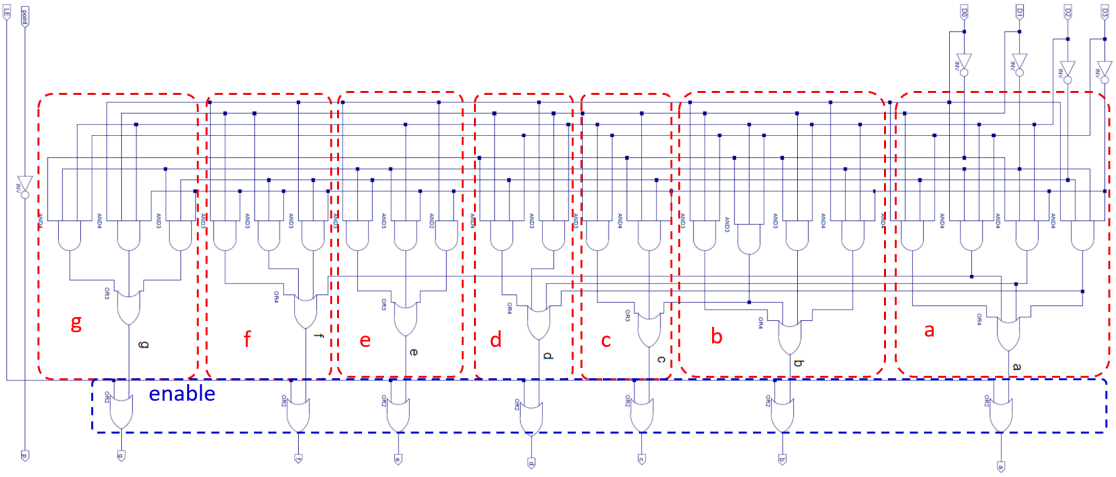


图 3: 电路图

3.2 多位七段数码管显示原理

静态显示 每个 7 段码对应一个显示译码电路

动态扫描显示：时分复用显示 利用人眼视觉残留，一个 7 段码译码电路分时为每个 7 段码提供译码

控制时序 用定时计数信号控制公共极，分时输出对应七段码的显示信号 (动态扫描)

4 位七段码结构　正极：公共端，七段信号并联。结构如下图

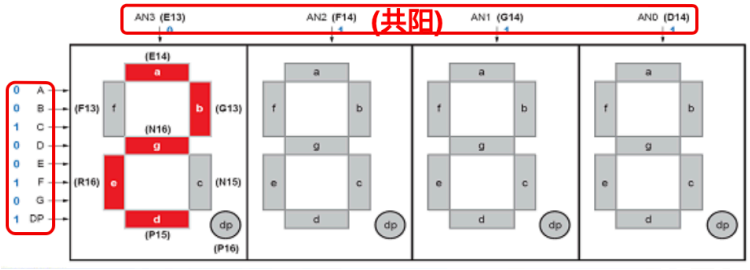


图 4: 4 位七段码结构

3.3　分时控制示意

动态扫描

1. 低电平与输入显示对应
2. 分时送 $a \sim g, p$
3. 可用序列信号控制

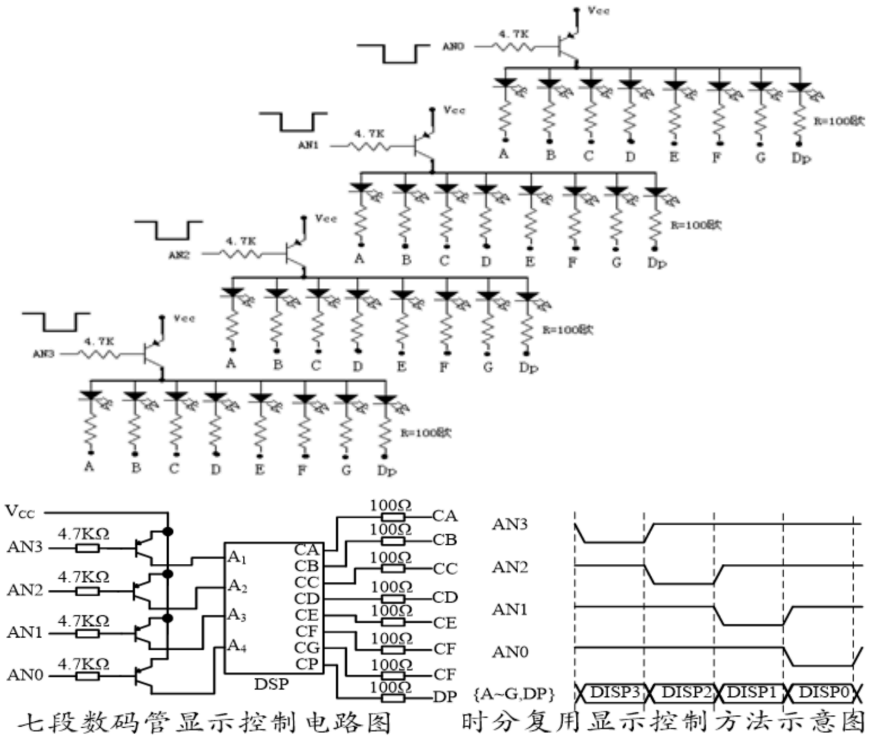


图 5: 分段控制示意

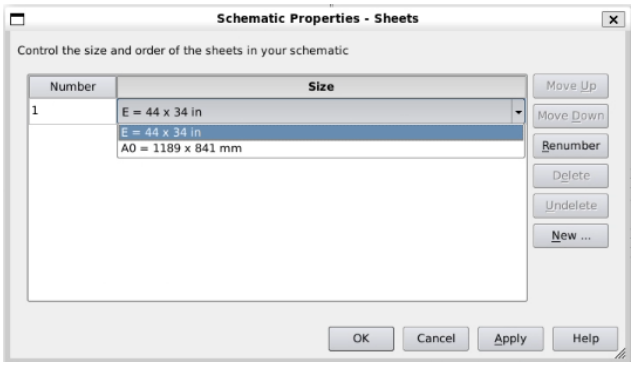
4 实验内容与步骤

4.1 原理图设计实现显示译码 MyMC14495 模块

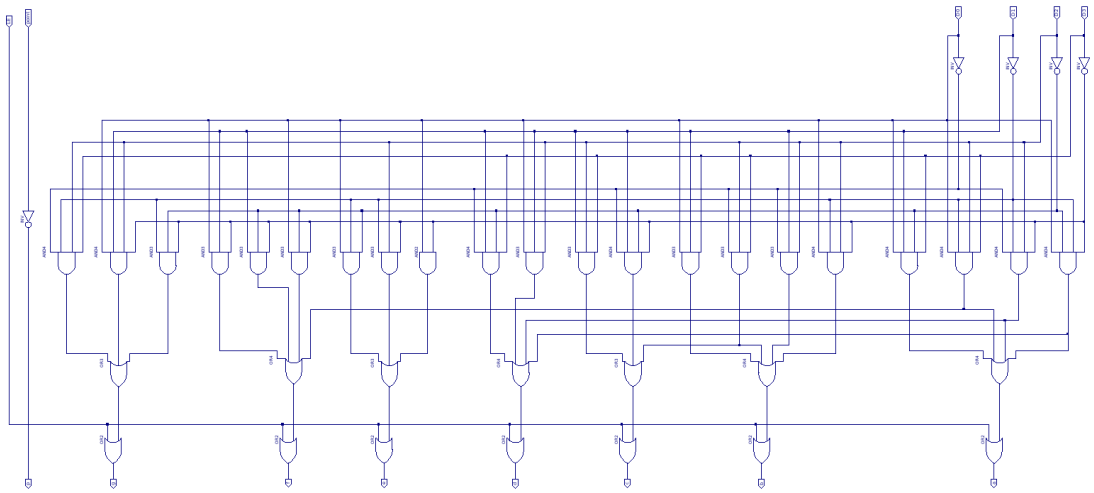
4.1.1 设计实现 MY_MC14495

1. 新建工程，工程名称用 MyMC14495。
2. 新建源文件，文件名称用 MyMC14495。
3. 原理图方式进行设计。

   用 ISE 的 Edit->Change Sheet Size 菜单项来改变原理图绘图区的尺寸



这里选择的是 44 × 34in 尺寸。
绘制出如下的原理图：



1. Check Design Rules，检查错误
2. View HDL Functional Model，查看并学习 Verilog HDL 代码，代码如下：

```
`timescale 1ns / 1ps

module MyMC14495(D0,
                 D1,
                 D2,
                 D3,
                 LE,
                 point,
                 a,
                 b,
                 c,
                 d,
                 e,
                 f,
                 g,
                 p);

    input D0;
    input D1;
    input D2;
    input D3;
    input LE;
    input point;
   output a;
   output b;
   output c;
   output d;
   output e;
   output f;
   output g;
   output p;

   wire XLXN_5;
   wire XLXN_15;
   wire XLXN_19;
   wire XLXN_20;
   wire XLXN_27;
   wire XLXN_28;
   wire XLXN_29;
   wire XLXN_30;
   wire XLXN_31;
   wire XLXN_32;
   wire XLXN_37;
   wire XLXN_38;
   wire XLXN_39;
   wire XLXN_40;
```

```
47    wire XLXN_42;
48    wire XLXN_43;
49    wire XLXN_44;
50    wire XLXN_45;
51    wire XLXN_46;
52    wire XLXN_47;
53    wire XLXN_48;
54    wire XLXN_49;
55    wire XLXN_50;
56    wire XLXN_55;
57    wire XLXN_57;
58    wire XLXN_58;
59    wire XLXN_59;
60    wire XLXN_60;
61    wire XLXN_61;
62    wire XLXN_62;
63    wire XLXN_63;
64    wire XLXN_64;
65
66    AND4  XLXI_1 (.I0(D0),
67                  .I1(XLXN_15),
68                  .I2(XLXN_19),
69                  .I3(XLXN_5),
70                  .O(XLXN_57));
71    AND4  XLXI_2 (.I0(XLXN_20),
72                  .I1(XLXN_19),
73                  .I2(D2),
74                  .I3(XLXN_5),
75                  .O(XLXN_44));
76    AND4  XLXI_3 (.I0(D0),
77                  .I1(XLXN_19),
78                  .I2(D2),
79                  .I3(D3),
80                  .O(XLXN_37));
81    AND4  XLXI_4 (.I0(D0),
82                  .I1(D1),
83                  .I2(XLXN_15),
84                  .I3(D3),
85                  .O(XLXN_55));
86    OR4  XLXI_17 (.I0(XLXN_55),
87                  .I1(XLXN_37),
88                  .I2(XLXN_44),
89                  .I3(XLXN_57),
90                  .O(XLXN_64));
91    OR2  XLXI_18 (.I0(LE),
92                  .I1(XLXN_64),
93                  .O(a));
```

```
94    AND4  XLXI_19 (.I0(D0),
95                   .I1(XLXN_19),
96                   .I2(D2),
97                   .I3(XLXN_5),
98                   .O(XLXN_50));
99    AND3  XLXI_20 (.I0(XLXN_20),
100                  .I1(D1),
101                  .I2(D2),
102                  .O(XLXN_49));
103   AND3  XLXI_21 (.I0(XLXN_20),
104                  .I1(D2),
105                  .I2(D3),
106                  .O(XLXN_47));
107   AND3  XLXI_22 (.I0(D0),
108                  .I1(D1),
109                  .I2(D3),
110                  .O(XLXN_48));
111   AND4  XLXI_43 (.I0(XLXN_20),
112                  .I1(D1),
113                  .I2(XLXN_15),
114                  .I3(XLXN_5),
115                  .O(XLXN_46));
116   AND3  XLXI_44 (.I0(D1),
117                  .I1(D2),
118                  .I2(D3),
119                  .O(XLXN_45));
120   AND3  XLXI_45 (.I0(D0),
121                  .I1(D1),
122                  .I2(D2),
123                  .O(XLXN_43));
124   AND4  XLXI_46 (.I0(XLXN_20),
125                  .I1(D1),
126                  .I2(XLXN_15),
127                  .I3(D3),
128                  .O(XLXN_42));
129   AND2  XLXI_47 (.I0(D0),
130                  .I1(XLXN_5),
131                  .O(XLXN_40));
132   AND3  XLXI_48 (.I0(XLXN_19),
133                  .I1(D2),
134                  .I2(XLXN_5),
135                  .O(XLXN_39));
136   AND3  XLXI_50 (.I0(D0),
137                  .I1(XLXN_19),
138                  .I2(XLXN_15),
139                  .O(XLXN_38));
140   AND3  XLXI_51 (.I0(D0),
```

```
                .I1(XLXN_15),
                .I2(XLXN_5),
                .O(XLXN_32));
    AND3  XLXI_52 (.I0(D1),
                .I1(XLXN_15),
                .I2(XLXN_5),
                .O(XLXN_31));
    AND3  XLXI_53 (.I0(D0),
                .I1(D1),
                .I2(XLXN_5),
                .O(XLXN_30));
    AND3  XLXI_54 (.I0(XLXN_19),
                .I1(XLXN_15),
                .I2(XLXN_5),
                .O(XLXN_29));
    AND4  XLXI_55 (.I0(D0),
                .I1(D1),
                .I2(D2),
                .I3(XLXN_5),
                .O(XLXN_28));
    AND4  XLXI_56 (.I0(XLXN_20),
                .I1(XLXN_19),
                .I2(D2),
                .I3(D3),
                .O(XLXN_27));
    OR4  XLXI_57 (.I0(XLXN_48),
                .I1(XLXN_47),
                .I2(XLXN_49),
                .I3(XLXN_50),
                .O(XLXN_63));
    OR3  XLXI_58 (.I0(XLXN_45),
                .I1(XLXN_46),
                .I2(XLXN_47),
                .O(XLXN_62));
    OR4  XLXI_59 (.I0(XLXN_42),
                .I1(XLXN_43),
                .I2(XLXN_44),
                .I3(XLXN_57),
                .O(XLXN_61));
    OR3  XLXI_60 (.I0(XLXN_38),
                .I1(XLXN_39),
                .I2(XLXN_40),
                .O(XLXN_60));
    OR4  XLXI_61 (.I0(XLXN_30),
                .I1(XLXN_31),
                .I2(XLXN_32),
                .I3(XLXN_37),
```

```
                .O(XLXN_59));
   OR3  XLXI_62 (.I0(XLXN_27),
                .I1(XLXN_28),
                .I2(XLXN_29),
                .O(XLXN_58));
   OR2  XLXI_63 (.I0(LE),
                .I1(XLXN_63),
                .O(b));
   OR2  XLXI_64 (.I0(LE),
                .I1(XLXN_62),
                .O(c));
   OR2  XLXI_65 (.I0(LE),
                .I1(XLXN_61),
                .O(d));
   OR2  XLXI_66 (.I0(LE),
                .I1(XLXN_60),
                .O(e));
   OR2  XLXI_67 (.I0(LE),
                .I1(XLXN_59),
                .O(f));
   OR2  XLXI_68 (.I0(LE),
                .I1(XLXN_58),
                .O(g));
   INV  XLXI_69 (.I(D3),
                .O(XLXN_5));
   INV  XLXI_70 (.I(D2),
                .O(XLXN_15));
   INV  XLXI_71 (.I(D1),
                .O(XLXN_19));
   INV  XLXI_72 (.I(D0),
                .O(XLXN_20));
   INV  XLXI_96 (.I(point),
                .O(p));
endmodule
```

4.1.2 仿真

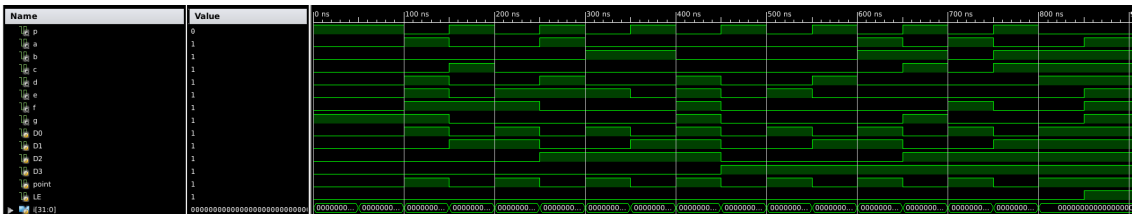
对 MyMC14495 模块进行仿真，激励代码如下

```
`timescale 1ns / 1ps

module MyMC14495_MyMC14495_sch_tb();

// Inputs
   reg D2;
   reg D3;
   reg D1;
```

```
9     reg D0;
10    reg point;
11    reg LE;
12
13  // Output
14    wire p;
15    wire g;
16    wire f;
17    wire e;
18    wire d;
19    wire c;
20    wire b;
21    wire a;
22
23  // Bidirs
24
25  // Instantiate the UUT
26    MyMC14495 UUT (
27                .D0(D0),
28                .D1(D1),
29                .D2(D2),
30                .D3(D3),
31                .point(point),
32                .LE(LE),
33                .p(p),
34                .g(g),
35                .f(f),
36                .e(e),
37                .d(d),
38                .c(c),
39                .b(b),
40                .a(a)
41    );
42  // Initialize Inputs
43  //   `ifdef auto_init
44                integer i;
45      initial begin
46                        D3 = 0;
47                        D2 = 0;
48                        D1 = 0;
49                        D0 = 0;
50                        LE = 0;
51                        point = 0;
52
53                        for (i = 0; i < 16; i = i + 1) begin
54                                #50
55                                {D3,D2,D1,D0} = i;
```

```
56                                  point = i;
57                          end
58
59                          #50
60                          LE = 1;
61                  end
62  //    `endif
63  endmodule
```

点击 Behavioral Check Synax，进行激励代码的查验。通过后，Process 窗口中选择 Simulate Behavioral Model，查看仿真激励波形。结果如下图：



4.1.3 生成逻辑符号图

Create Schematic Symbol，系统生成 MyMC14495 模块的逻辑符号图文件，文件后缀 `.sym`。

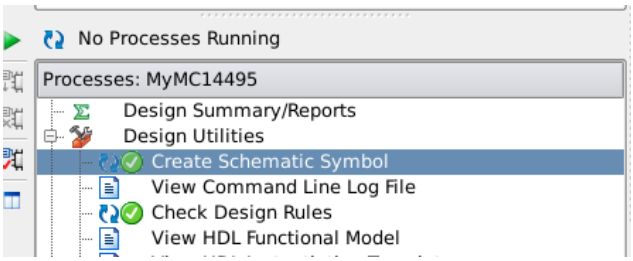


图 6: 生成逻辑符号

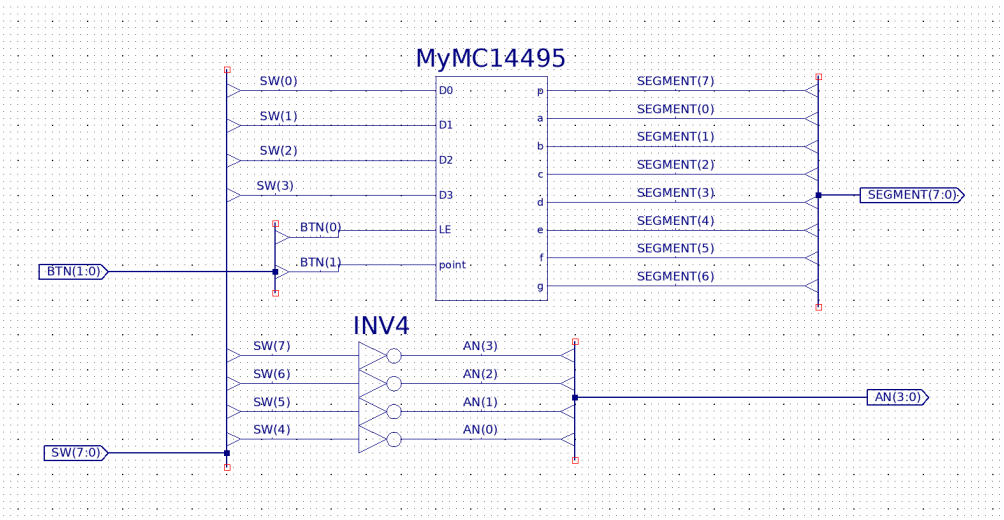
此外，自动生成的符号可以打开 `.sym` 文件修改。使用时必须复制 `.sym` 和 `.sch` 到对应工程目录。

4.2 实现数码管显示

4.2.1 电路实现

1. 新建工程 DispNumber_sch
2. 新建 schematic 文件 DispNumber_sch
3. 复制 `MyMC14495.sym` 和 `MyMC14495.sch` 到工程根目录
4. 在 symbols 框里的第一个元件，就是 MyMC14495

5. 绘制电路图。结果如下所示



4.2.2 引脚定义

新建引脚约束文件 K7.ucf，输入以下内容：

```
NET "SW[0]" LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "SW[1]" LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "SW[2]" LOC = AA13 | IOSTANDARD = LVCMOS15;
NET "SW[3]" LOC = AA12 | IOSTANDARD = LVCMOS15;
NET "SW[4]" LOC = Y13 | IOSTANDARD = LVCMOS15;
NET "SW[5]" LOC = Y12 | IOSTANDARD = LVCMOS15;
NET "SW[6]" LOC = AD11 | IOSTANDARD = LVCMOS15;
NET "SW[7]" LOC = AD10 | IOSTANDARD = LVCMOS15;

NET "BTN[0]"LOC = AF13 | IOSTANDARD = LVCMOS15;#SW[14]
NET "BTN[1]"LOC = AF10 | IOSTANDARD = LVCMOS15;#SW[15]

NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;#a
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;#c
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;#d
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;#e
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;#f
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;#point

NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;
```

4.2.3 下载验证

点击 Generate Programming Files, 通过后点击 Configure Target Device -> Manage Configuration Project，将文件导出到 SWORD 实验板上进行验证。

根据电路原理图和引脚约束文件，最左边的开关对应 “point”，即是否显示小数点；左边第二个开关对应 “LE”，即使能；从右往左四个开关依次对应 $D_0, D_1, D_2, D_3, SW_0, SW_1, SW_2, SW_3$。

LE = 1 当 LE = 1 时，相当于电路被 “关闭”(注意到电路中，将 LE 的值置为 1 时，abcdef 的输出均为 1，又因为我们本次采取的是共阳结构，故灯的各个灯管都不会亮起)

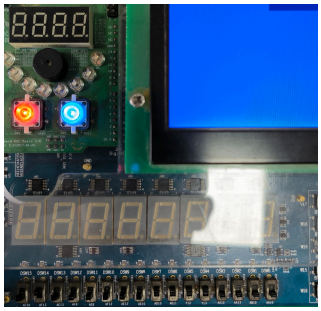
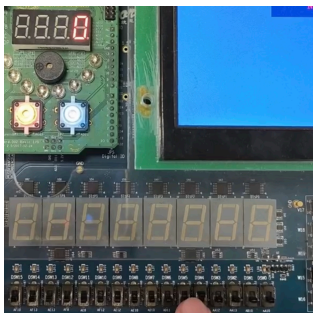
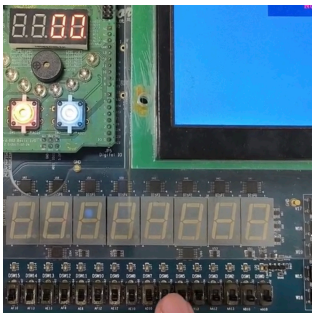


图 7: EN=1, SW index = 1111, 灯均不亮

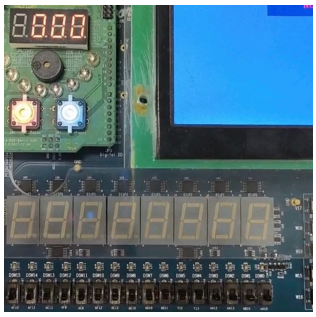
LE = 0 此时可以通过右边八个开关来控制哪些灯亮以及显示什么数字，效果如图



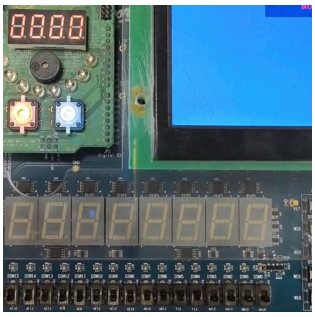
(a) SW index = 0001



(b) SW index = 0011



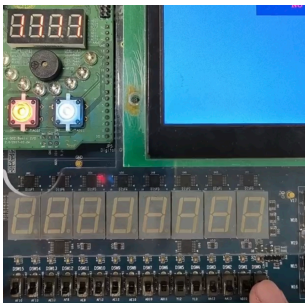
(c) SW index = 0111



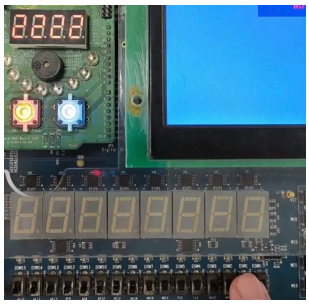
(d) SW index = 1111

图 8: point=LE=0, 显示数字为 0

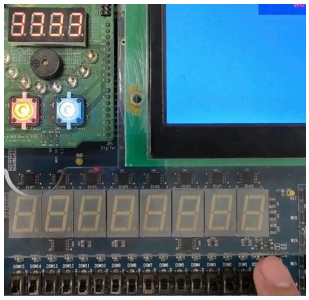
改变最右边四个开关的开闭，使其显示不同的数字：



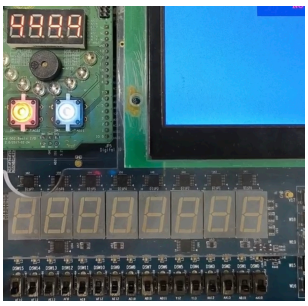
(a) D index = 0001



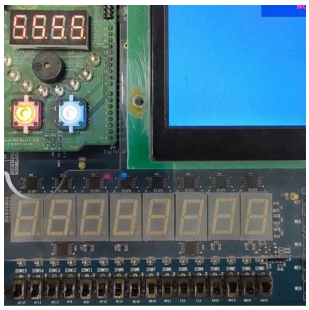
(b) D index = 0010



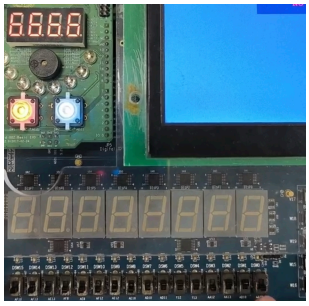
(c) D index = 0011



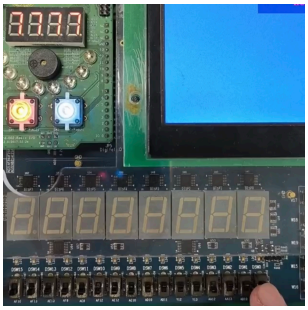
(d) D index = 0100



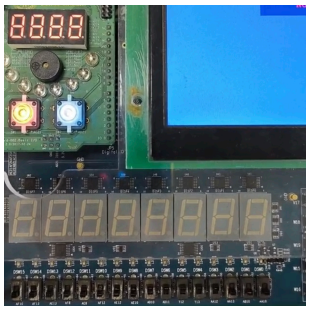
(e) D index = 0101



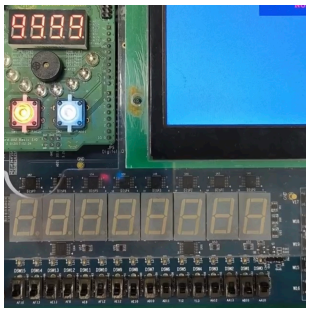
(f) D index = 0110



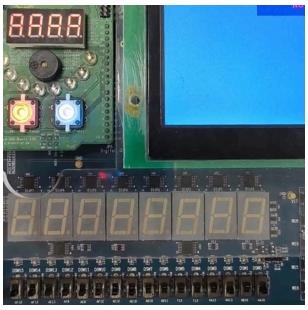
(g) D index = 0111



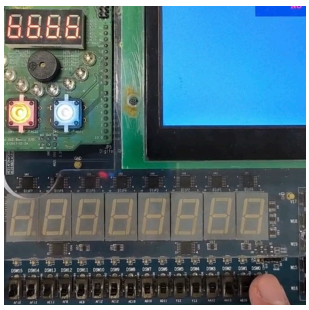
(h) D index = 1000



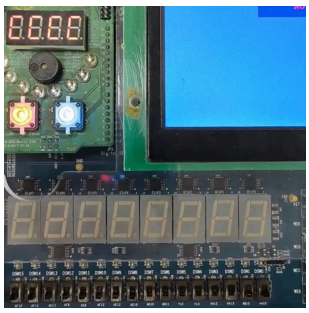
(i) D index = 1001



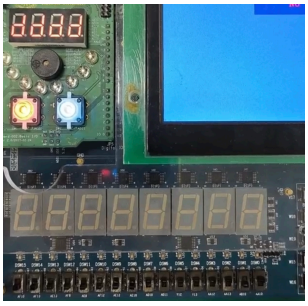
(j) D index = 1010



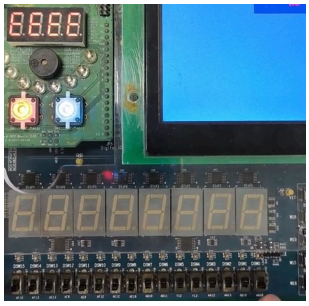
(k) D index = 1011



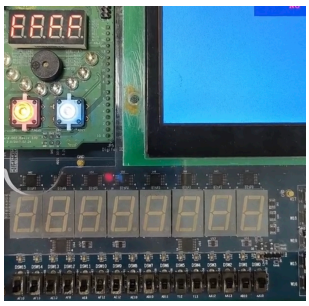
(l) D index = 1100



(a) D index = 1101

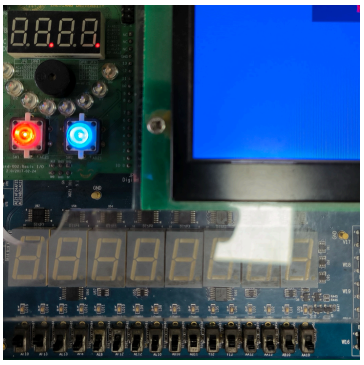


(b) D index = 1110



(c) D index = 1111

小数点功能：



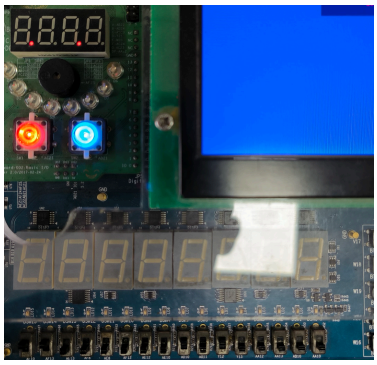
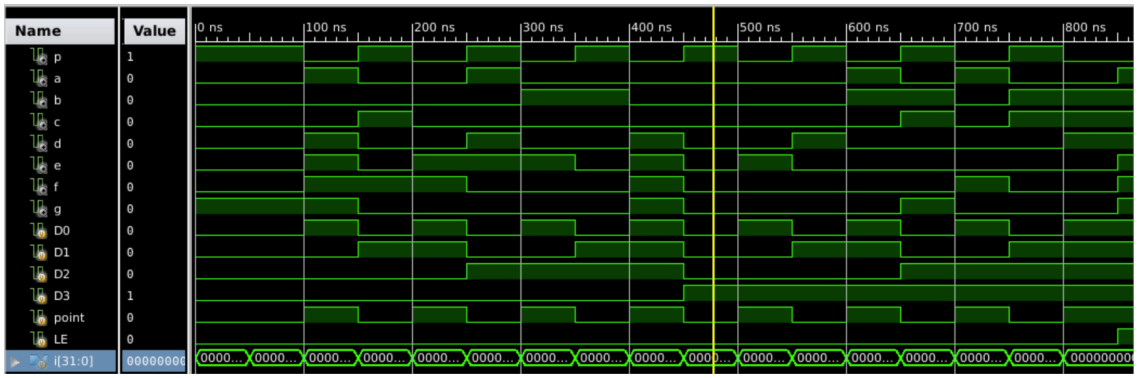


图 11: point=1,LE=0 改变中间开关，观察小数点数量

经过验证小数点的效果符合预期。

5 实验结果分析

分析仿真结果 仿真波形如图：



以黄线处为例，$D_3, D_2, D_1, D_0 = 1000$，即要显示的数字为 $1000 = 8$ 。此时观察输出全为 0，即所有灯都亮起，显示的数字为 8。

其他输入同样能够得到显示对应数字的输出，说明仿真结果正确。

6 讨论与心得

Lab6 的原理图相对较为复杂，起初使用标准的绘图板尺寸难以绘制出整洁的图示，因此需要对绘图板的尺寸进行适度的调整。在绘制过程中，如果出现错误，可以通过比对物理验证过程中出现错误的数值与相应的仿真图来核查相应的电路模块。此外，在进行修改时，我们可以在选项中将其切换为 “Select Segment”，方便删除或移动线段。

总的来说，本次实验并没有涉及特别复杂的理论内容，相关原理我们在之前已经学过，最大的挑战主要集中在电路图的绘制部分。