实验五——变量译码器设计与使用

姓名：张琦	学号：3180103162	专业：软件工程
课程名称：数字逻辑设计	同组学生：无	实验日期：2019-10-16
实验地点：紫金港东 4-509	指导老师：洪奇军	

一、实验目的和要求

1. 掌握变量译码器的逻辑构成和逻辑功能
2. 用变量译码器实现组合函数
3. 掌握变量译码器的典型应用（地址译码的具体方法）
4. 了解存储器编址的概念
5. 采用原理图设计电路模块
6. 原理图设计实现74LS138译码器模块
7. 用74LS138译码器实现楼道灯控制
8. 进一步熟悉ISE平台及下载实验平台物理验证

二、实验内容和原理

2.1 实验内容

2.1.1 原理图设计实现 74LS138 译码器模块

2.1.2 用74LS138译码器实现楼道灯控制

2.2 实验原理

2.2.1 译码器原理

译码器是将一种输入编码转换成另一种编码的电路，即将给定的代码进行“翻译”并转换成指定的状态或输出信号（脉冲或电

平）。

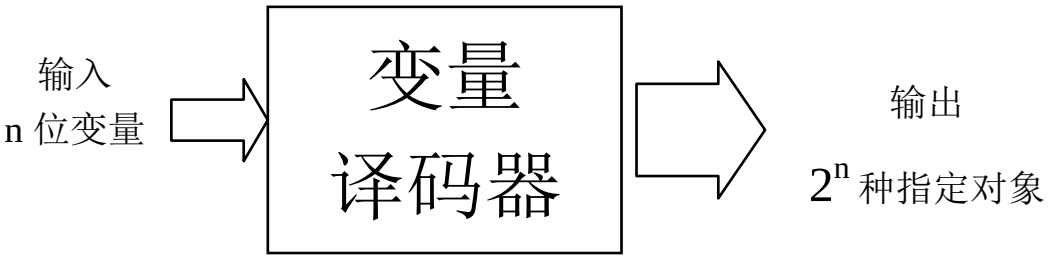
译码可分为：变量译码、显示译码。

变量译码一般是将一种较少位输入变为较多位输出的器件，如2ⁿ译码和8421BCD码译码。

显示译码主要进行2进制数显示成10进制或16进制数的转换，可分为驱动LED和LCD两类。

2.2.2 变量译码器

变量译码器是一个将n个输入变为2^n个最小项输出的多输出端的组合逻辑电路。n通常在2~64之间。



2.2.2.1 74LS138变量译码器

输入		译码器输出（低电平有效）							
使能	变量								
$GG_{2A}G_{2B}$	CBA	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×11	×××	1	1	1	1	1	1	1	1
0××	×××	1	1	1	1	1	1	1	1
100	000	**0**	1	1	1	1	1	1	1
100	001	1	**0**	1	1	1	1	1	1
100	010	1	1	**0**	1	1	1	1	1
100	011	1	1	1	**0**	1	1	1	1
100	100	1	1	1	1	**0**	1	1	1
100	101	1	1	1	1	1	**0**	1	1
100	110	1	1	1	1	1	1	**0**	1
100	111	1	1	1	1	1	1	1	**0**

表1 74LS138变量译码器功能表

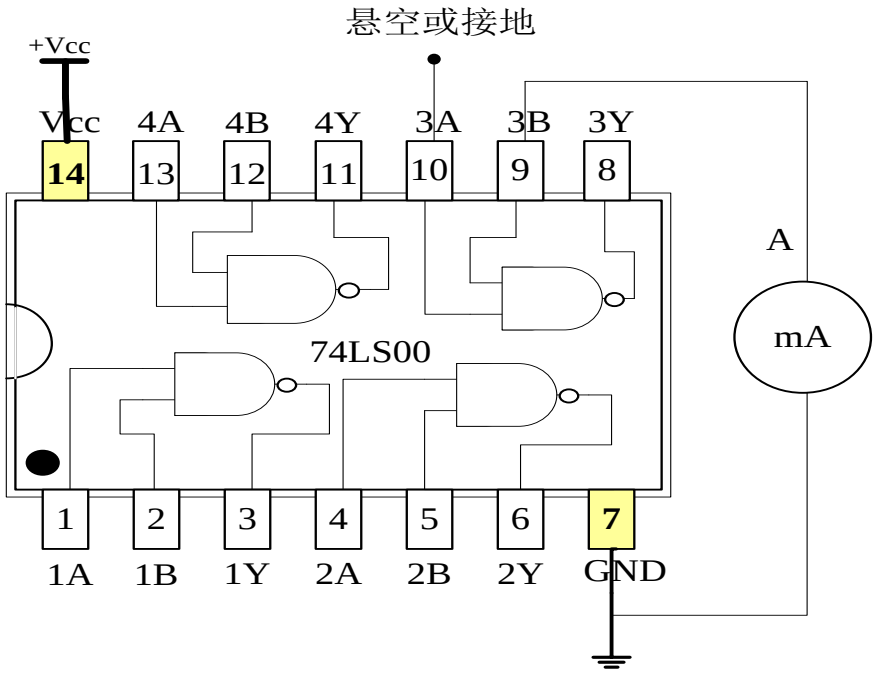
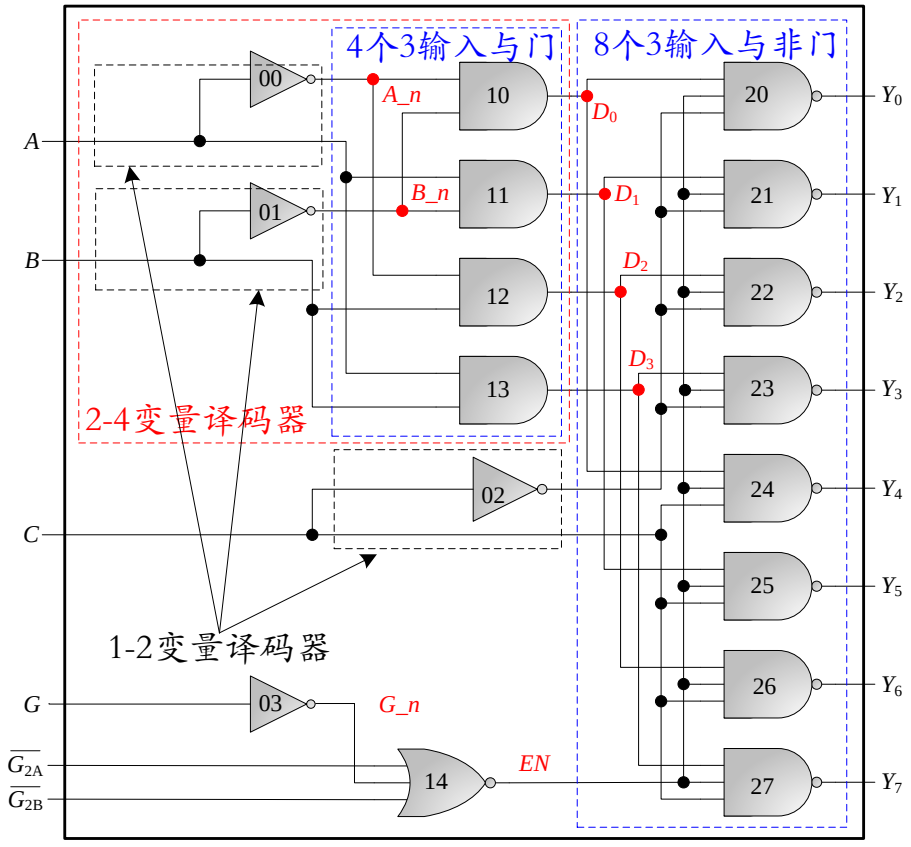


图1 74LS138变量译码器引脚



带3个使能端的3-8译码器的逻辑结构由三级门电路构成，输出低电平有效。

图2 74LS138变量译码器原理图

2.2.2.2 74LS139变量译码器

输入		译码器输出（低电平有效）			
使能	变量				
G	BA	Y_0	Y_1	Y_2	Y_3
1	××	1	1	1	1
0	00	0	1	1	1
0	01	1	0	1	1
0	10	1	1	0	1
0	11	1	1	1	0

表2 74LS139变量译码器功能表

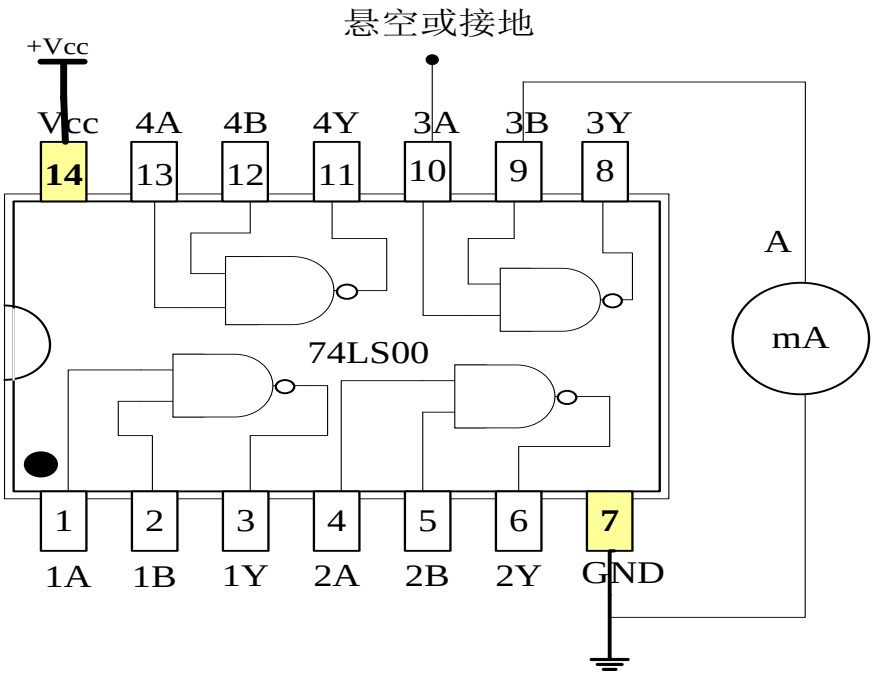


图3 74LS139变量译码器引脚

2.2.3 用变量译码器实现组合函数

变量译码器的输出对应所有输入变量的最小项组合，如果将函数转换成最小项和的形式，则可以用变量译码器实现函数的组合电路：

$$F = \bar{S}_3\bar{S}_2S_1 + \bar{S}_3S_2\bar{S}_1 + S_3\bar{S}_2\bar{S}_1 + S_3S_2S_1$$

$$F = 001 + 010 + 100 + 111$$

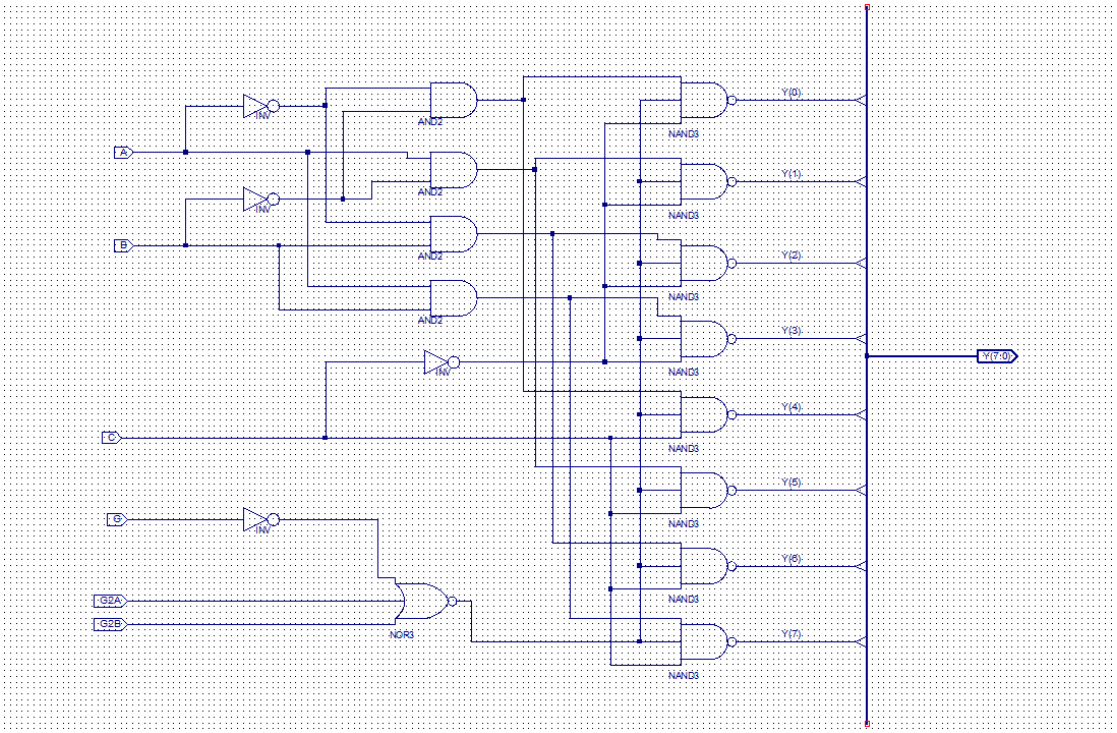


图4 74LS138译码器原理图

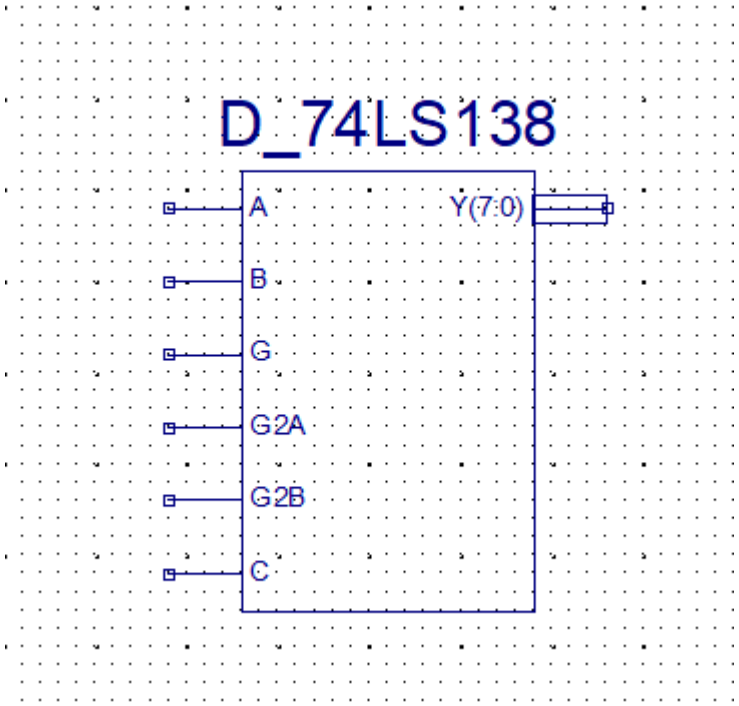


图5 D_74LS138元件输入输出原理图

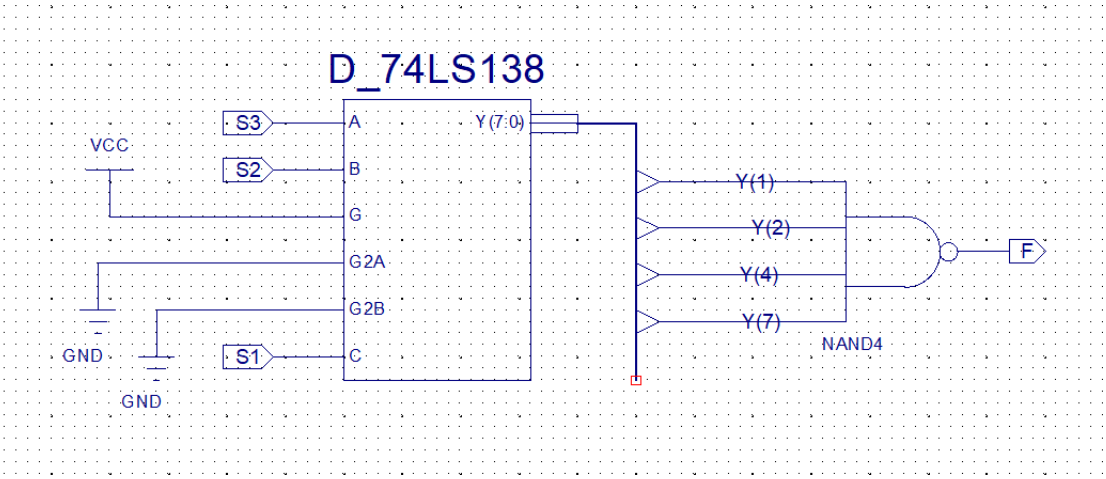


图6 D_74LS138实现楼道灯控制原理图

三、主要仪器设备

1.装有Xilinx ISE 14.7的计算机1台

2.SWORD开发板

四、操作方法与实验步骤

4.1 原理图设计实现74LS138译码器模块

4.1.1 新建工程，工程名称用D_74LS138_SCH。（第一个工程）

4.1.2 新建 Schematic 源文件，文件名称用D_74LS138。

4.1.3 原理图方式进行设计。

4.1.4 Check Design Rules，检查错误。

4.1.5 View HDL Functional Model，查看并学习 Verilog HDL 代码。

Verilog HDL代码：

```
1.  `timescale 1ns / 1ps
2.
3.  module D_74LS138(A,
4.              B,
5.              C,
6.              G,
```

```
7.              G2A,
8.              G2B,
9.              Y);
10.
11.   input A;
12.   input B;
13.   input C;
14.   input G;
15.   input G2A;
16.   input G2B;
17.  output [7:0] Y;
18.
19.  wire XLXN_2;
20.  wire XLXN_5;
21.  wire XLXN_13;
22.  wire XLXN_19;
23.  wire XLXN_28;
24.  wire XLXN_33;
25.  wire XLXN_36;
26.  wire XLXN_38;
27.  wire XLXN_43;
28.
29.  INV  XLXI_1 (.I(A),
30.           .O(XLXN_2));
31.  INV  XLXI_2 (.I(B),
32.           .O(XLXN_5));
33.  INV  XLXI_3 (.I(C),
34.           .O(XLXN_19));
35.  INV  XLXI_4 (.I(G),
36.           .O(XLXN_13));
37.  AND2  XLXI_5 (.I0(XLXN_5),
38.           .I1(XLXN_2),
39.           .O(XLXN_28));
40.  AND2  XLXI_6 (.I0(XLXN_5),
41.           .I1(A),
42.           .O(XLXN_33));
43.  AND2  XLXI_7 (.I0(B),
44.           .I1(XLXN_2),
45.           .O(XLXN_36));
46.  AND2  XLXI_8 (.I0(B),
47.           .I1(A),
48.           .O(XLXN_38));
49.  NAND3  XLXI_9 (.I0(XLXN_19),
50.           .I1(XLXN_43),
```

```
        .I2(XLXN_28),
        .O(Y[0]));
NAND3  XLXI_10 (.I0(XLXN_19),
        .I1(XLXN_43),
        .I2(XLXN_33),
        .O(Y[1]));
NAND3  XLXI_11 (.I0(XLXN_19),
        .I1(XLXN_43),
        .I2(XLXN_36),
        .O(Y[2]));
NAND3  XLXI_12 (.I0(XLXN_19),
        .I1(XLXN_43),
        .I2(XLXN_38),
        .O(Y[3]));
NAND3  XLXI_13 (.I0(C),
        .I1(XLXN_43),
        .I2(XLXN_28),
        .O(Y[4]));
NAND3  XLXI_14 (.I0(C),
        .I1(XLXN_43),
        .I2(XLXN_33),
        .O(Y[5]));
NAND3  XLXI_15 (.I0(C),
        .I1(XLXN_43),
        .I2(XLXN_36),
        .O(Y[6]));
NAND3  XLXI_16 (.I0(C),
        .I1(XLXN_43),
        .I2(XLXN_38),
        .O(Y[7]));
NOR3  XLXI_18 (.I0(G2B),
        .I1(G2A),
        .I2(XLXN_13),
        .O(XLXN_43));
endmodule
```

4.1.6 仿真，加入激励代码。

激励代码：

```
`timescale 1ns / 1ps

module D_74LS138_D_74LS138_sch_tb();

// Inputs
```

```
6.    reg A;
7.    reg B;
8.    reg G;
9.    reg G2A;
10.   reg G2B;
11.   reg C;
12.
13. // Output
14.   wire [7:0] Y;
15.
16. // Bidirs
17.
18. // Instantiate the UUT
19.   D_74LS138 UUT (
20.       .A(A),
21.       .B(B),
22.       .G(G),
23.       .G2A(G2A),
24.       .G2B(G2B),
25.       .C(C),
26.       .Y(Y)
27.   );
28. // Initialize Inputs
29.   //initial
30.    integer i;
31.    initial begin
32.              C = 0;
33.              B = 0;
34.              A = 0;
35.
36.              G = 1;
37.              G2A = 0;
38.              G2B = 0;
39.              #50;
40.
41.              for(i=0;i<=7;i=i+1) begin
42.                       {C,B,A} = i;
43.              #50;
44.              end
45.
46.              assign G = 0;
47.              assign G2A = 0;
48.              assign G2B = 0;
49.              #50;
```

```
50.
51.             assign G = 1;
52.             assign G2A = 1;
53.             assign G2B = 0;
54.             #50;
55.
56.             assign G = 1;
57.             assign G2A = 0;
58.             assign G2B = 1;
59.             #50;
60.         end
61. endmodule
```

4.1.6 生成逻辑符号图和VF文件。

4.2 验证D_74LS138，实现楼道灯控制。

4.2.1 原理图

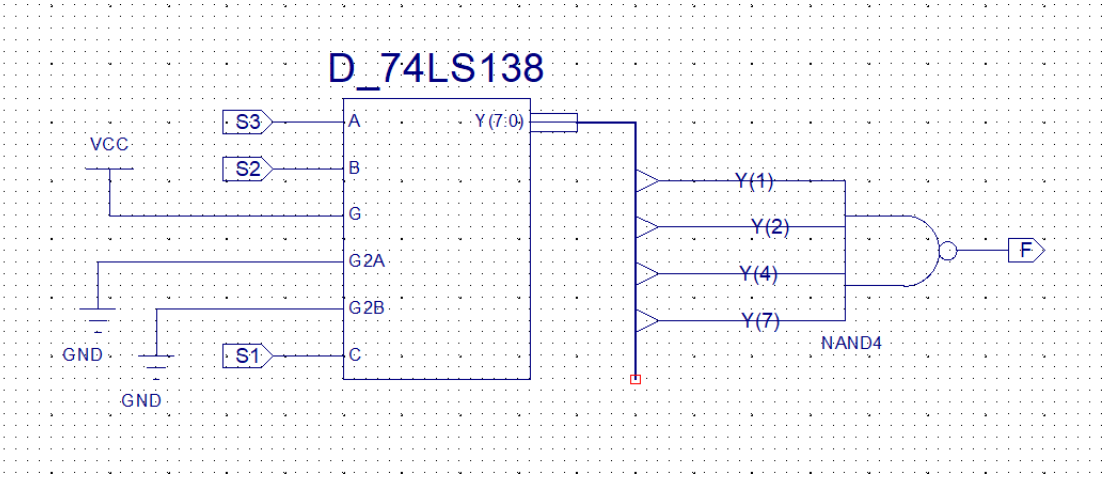


图7 D_74LS138实现楼道灯控制原理图

4.2.2 引脚约束

UCF引脚约束：

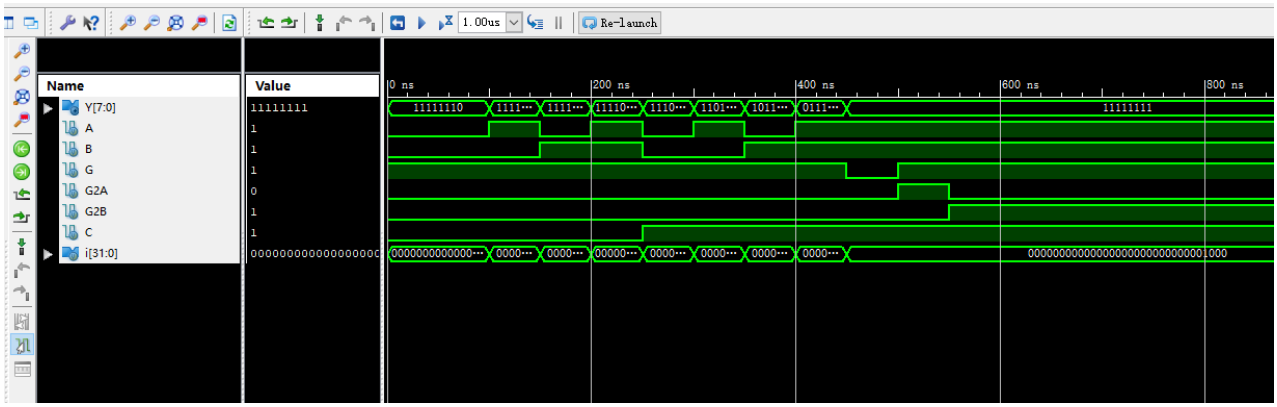
```
1.  NET "F" LOC = U21 | IOSTANDARD = LVCMOS33 ;
2.  NET "S1" LOC = AE13 | IOSTANDARD = LVCMOS15 ;
3.  NET "S2" LOC = AF13 | IOSTANDARD = LVCMOS15 ;
4.  NET "S3" LOC = AF10 | IOSTANDARD = LVCMOS15 ;
5.  NET "G" LOC = AF8 | IOSTANDARD = LVCMOS15 ;
6.  NET "G2A" LOC = AE8 | IOSTANDARD = LVCMOS15 ;
7.  NET "G2B" LOC = AF12 | IOSTANDARD = LVCMOS15 ;
```

4.2.3 下载验证。

五、实验结果与分析

5.1 变量译码器-74LS138的实现

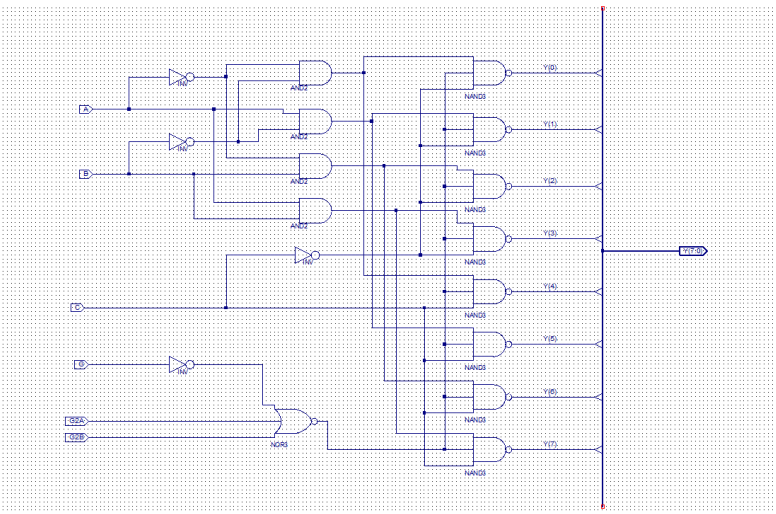
5.1.1 74LS138仿真图



5.1.2 仿真图分析

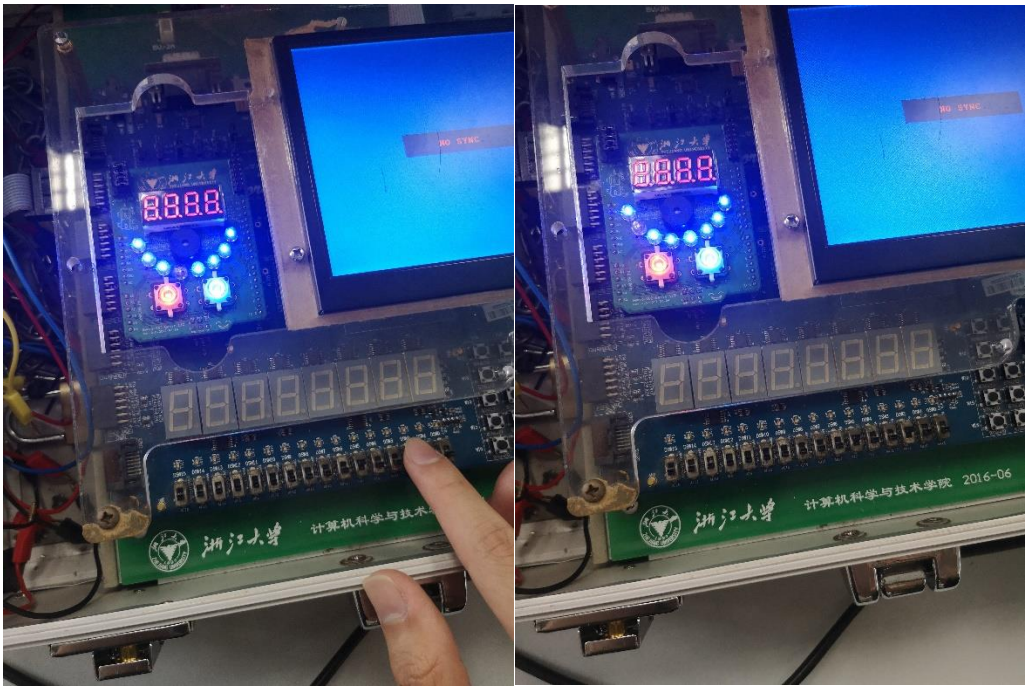
该译码器是将 CBA 三个输入的二进制数翻译为对应的八进制数，输出低电平有效。在仿真结果中，CBA 从000 开始变化，每次加一，依次为000,001,010,011,100,101,110,111，对应到 Y[7:0]被对应到的数电平变为 0，所以Y[0]，Y[1]，Y[2]，Y[3]，Y[4]，Y[5]，Y[6]，Y[7]依次变为 0，而其余的Y[n]在其他时刻总为一。当使能G为0或者G2A、G2B同时为1时Y[0]到Y[7]全部为1。

5.1.3 74LS138结构分析



74LS138 中有 3 个使能端，其逻辑结构由三级门电路组成，输出低电平有效。A先与 B组成 2-4 变量译码器，此结果再与 C 组成 1-2 变量译码器，总体成为3-8译码器。G、G2A、G2B三个使能端共同工作，当G2A、G2B为0且G为1同时满足时，74LS138 译码器才会正常工作。

5.1.4 74LS138下载结果



G2A、G2B为0且G为1同时满足时，74LS138 译码器才会正常工作。灯，且控制开关为低、低、低表示000时，8盏LED灯中第一盏未亮，符合电路逻辑。

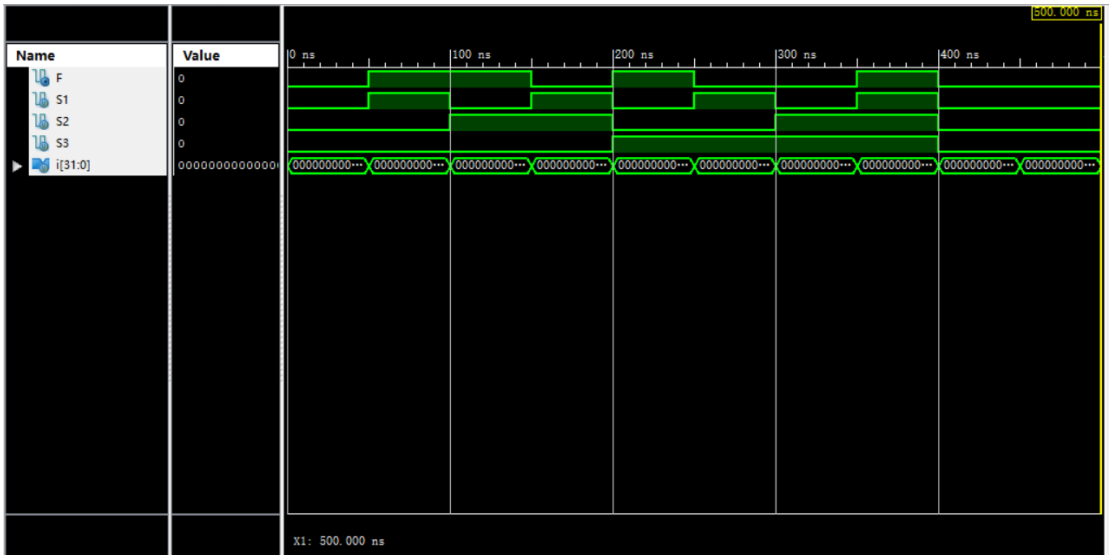
5.1.5 验证实验的真值表

经SWORD实验板验证，所得真值表与预期一致。

输入		译码器输出							
使能	变量	(低电平有效)							
GG2AG2B	CBA	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
×11	×××	1	1	1	1	1	1	1	1
0××	×××	1	1	1	1	1	1	1	1
100	0	0	1	1	1	1	1	1	1
100	1	1	0	1	1	1	1	1	1
100	10	1	1	0	1	1	1	1	1
100	11	1	1	1	0	1	1	1	1
100	100	1	1	1	1	0	1	1	1
100	101	1	1	1	1	1	0	1	1
100	110	1	1	1	1	1	1	0	1
100	111	1	1	1	1	1	1	1	0

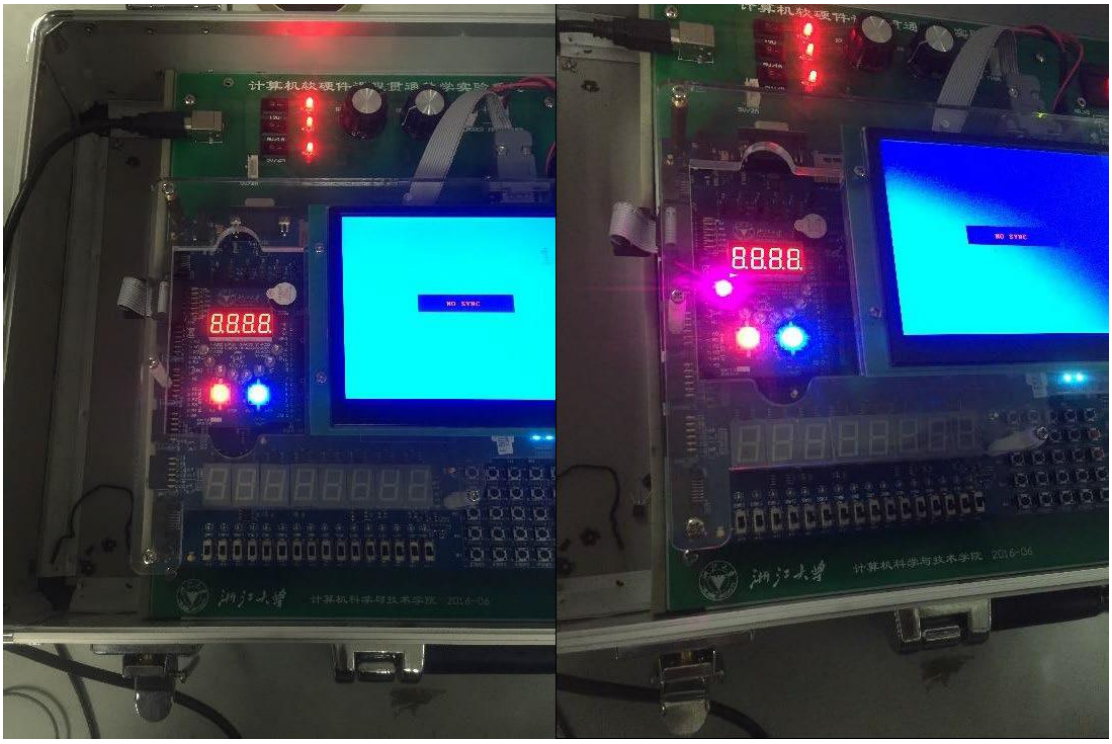
5.2 用74LS138译码器实现楼道灯控制器

5.2.1 lampctrl仿真实验



由仿真结果可知，当S1、S2、S3中有奇数个为1时输出为1，否则为0。

5.2.2 实验结果



只有当使能G开启，G2A、G2B都关闭的时候楼道灯才能正常工作，开启一个开关，灯变亮，此时开启另一个开关的时候灯熄灭，符合实验目的中的楼道灯控制部分。灯的开关控制过程存在少许的延时，猜测是各个门电路产生的延迟。

5.2.1 S1 S2 S3控制楼道灯的真值表

S_3	S_2	S_1	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

六、讨论与心得

与前四次不太一样，本次实验主要是在ISE平台通过绘制原理图实现的，因此更注重理论与实践的结合。本次完整的ISE平台操作让我熟悉了ISE平台的各项操作流程，同时也收获了一些经验教训：在绘制连线过程中，我们应该反复检查，单击需要检查的某一条连线，观察它连接的各个组件是否是我们需要的状态。

通过实验课的操作，我也对理论课中学习到的译码器有了更深入的了解，这也加深了我的理论知识。

实验六——七段数码管显示译码器设计与应用

姓名：张琦	学号：3180103162	专业：软件工程
课程名称：数字逻辑设计	同组学生：无	实验日期：2019-10-23
实验地点：紫金港东 4-509	指导老师：洪奇军	

一、实验目的和要求

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Xilin ISE环境及SWORD实验平台

二、实验内容和原理

2.1 实验内容

2.1.1 原理图设计实现显示译码MyMC14495模块

2.1.2 用MyMC14495模块实现数码管显示

2.2 实验原理

2.2.1 数字显示器件原理

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

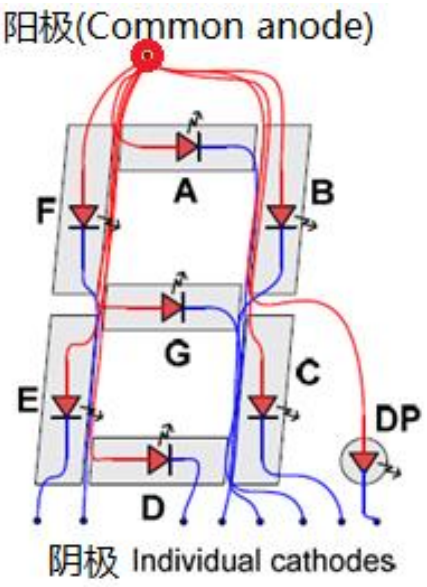


图1 数码管示意图

2.2.2 数码管控制原理

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 2.2.3 用变量译码器实现组合函数。

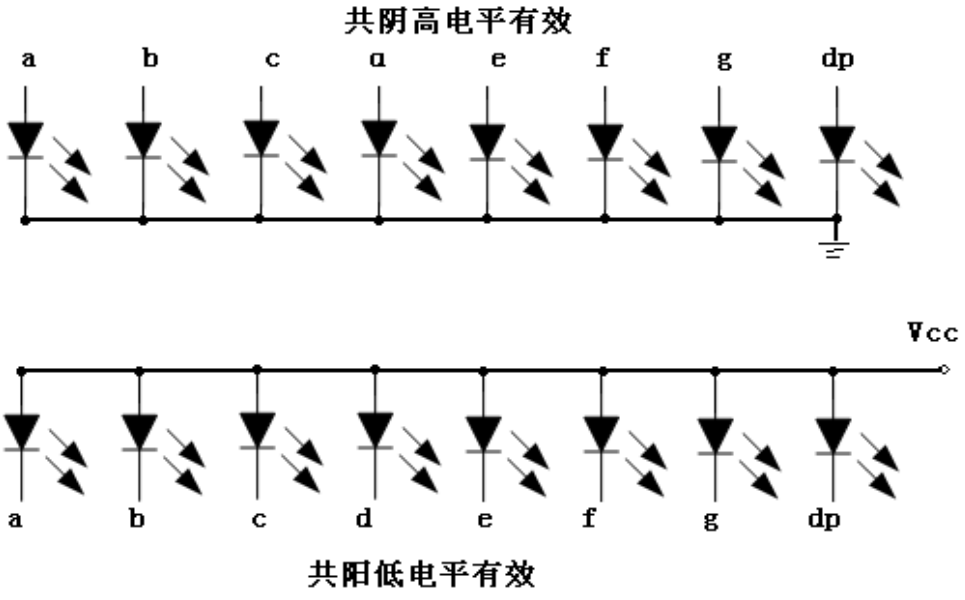


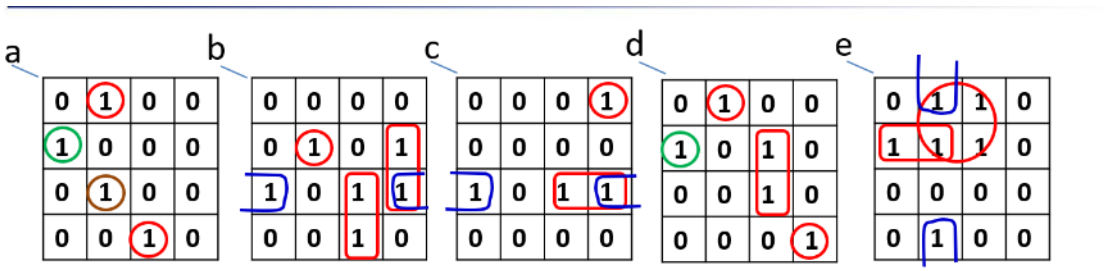
图2 共阴（阳）控制（7段数码管）示意图

2.2.3 显示原理

通过 4-16 译码器与七段数码管的一一对应来实现数字的显示。其对应关系如下：

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

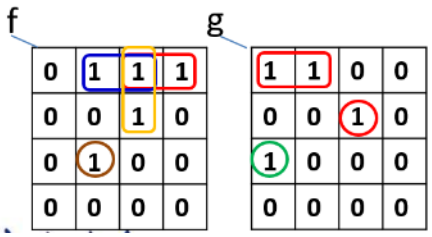
表1 译码器与七段数码管对应关系



$a = \bar{D}_3\bar{D}_2\bar{D}_1D_0+\bar{D}_3D_2\bar{D}_1\bar{D}_0+D_3\bar{D}_2D_1D_0+D_3D_2\bar{D}_1D_0$

$b = \bar{D}_3D_2\bar{D}_1D_0+D_2D_1\bar{D}_0+D_3D_2\bar{D}_0+D_3D_1D_0$

$c = \bar{D}_3\bar{D}_2D_1\bar{D}_0+D_3D_2\bar{D}_0+D_3D_2D_1$



$d = \bar{D}_3\bar{D}_2\bar{D}_1D_0+\bar{D}_3D_2\bar{D}_1\bar{D}_0+D_2D_1D_0+D_3\bar{D}_2D_1\bar{D}_0$

$e = \bar{D}_3D_0+\bar{D}_3D_2\bar{D}_1+\bar{D}_2\bar{D}_1D_0$

$f = \bar{D}_3\bar{D}_2D_0+\bar{D}_3\bar{D}_2D_1+\bar{D}_3D_1D_0+D_3D_2\bar{D}_1D_0$

$g = \bar{D}_3\bar{D}_2\bar{D}_1+\bar{D}_3D_2D_1D_0+D_3D_2\bar{D}_1\bar{D}_0$

图3 a-g的卡诺图化简结果

2.2.4 MyMC14495模块原理

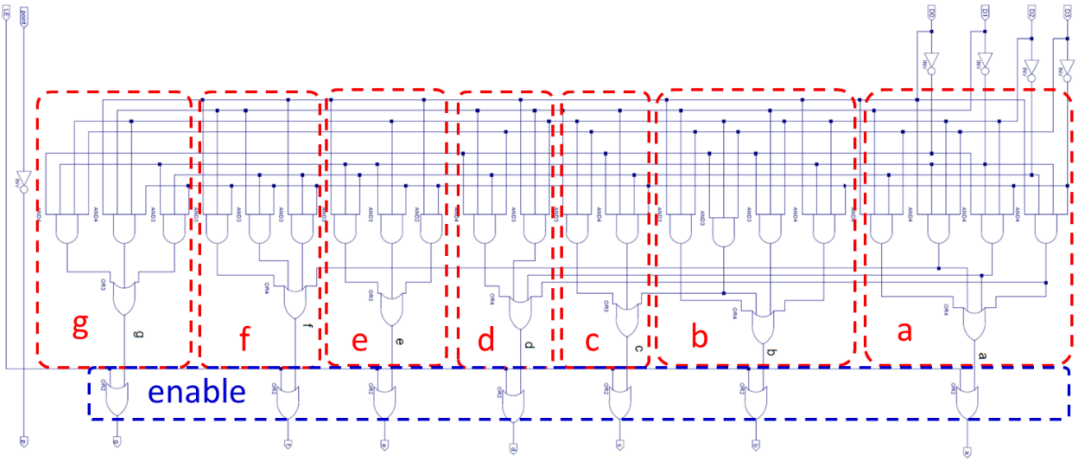


图3 MyMC14495模块原理图

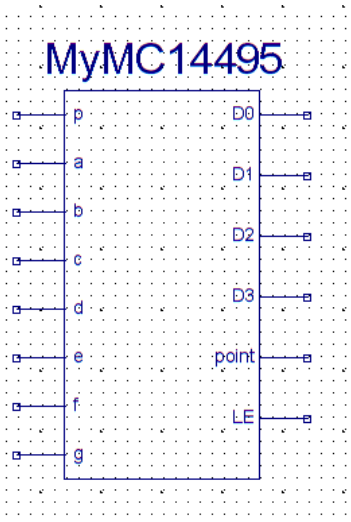


图4 MyMC14495模块图

2.2.5 多位七段数码管显示原理

2.2.5.1 静态显示

每个7段码对应一个显示译码电。

2.2.5.2 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。

2.2.5.3 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。

2.2.5.4 4位七段码结构

正极：公共端

七段信号并联

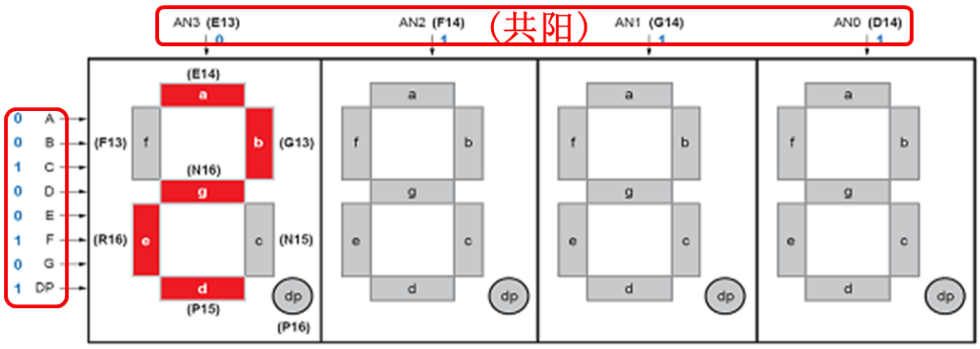


图5 多位七段数码管显示原理图

三、主要仪器设备

1.装有Xilinx ISE 14.7的计算机1台

2.SWORD开发板

四、操作方法与实验步骤

4.1 设计实现MY_MC14495

4.1.1 新建工程，工程名称用MyMC14495。

4.1.2 新建源文件，文件名称用MyMC14495。

4.1.3 原理图方式进行设计。

4.1.4 Check Design Rules，检查错误。

4.1.5 View HDL Functional Model，查看并学习 Verilog HDL代码。

Verilog HDL代码：

```
1. `timescale 1ns / 1ps
2. 
3. module MyMC14495(D0,
4.                   D1,
5.                   D2,
6.                   D3,
7.                   LE,
8.                   point,
9.                   a,
10.                  b,
11.                  c,
12.                  d,
13.                  e,
14.                  f,
15.                  g,
16.                  p);
17. 
18.    input D0;
19.    input D1;
20.    input D2;
21.    input D3;
22.    input LE;
23.    input point;
24.   output a;
25.   output b;
26.   output c;
27.   output d;
28.   output e;
29.   output f;
30.   output g;
31.   output p;
32. 
33.   wire XLXN_22;
34.   wire XLXN_24;
35.   wire XLXN_25;
36.   wire XLXN_80;
37.   wire XLXN_81;
38.   wire XLXN_82;
39.   wire XLXN_85;
40.   wire XLXN_86;
41.   wire XLXN_87;
42.   wire XLXN_88;
43.   wire XLXN_91;
44.   wire XLXN_92;
```

```
  wire XLXN_93;
  wire XLXN_94;
  wire XLXN_96;
  wire XLXN_97;
  wire XLXN_99;
  wire XLXN_101;
  wire XLXN_102;
  wire XLXN_104;
  wire XLXN_105;
  wire XLXN_110;
  wire XLXN_112;
  wire XLXN_116;
  wire XLXN_117;
  wire XLXN_120;
  wire XLXN_121;
  wire XLXN_123;
  wire XLXN_124;
  wire XLXN_125;
  wire XLXN_126;
  wire XLXN_127;

  OR2  XLXI_1 (.I0(LE),
               .I1(XLXN_120),
               .O(g));
  OR2  XLXI_2 (.I0(LE),
               .I1(XLXN_126),
               .O(f));
  OR2  XLXI_3 (.I0(LE),
               .I1(XLXN_127),
               .O(e));
  OR2  XLXI_4 (.I0(LE),
               .I1(XLXN_125),
               .O(d));
  OR2  XLXI_5 (.I0(LE),
               .I1(XLXN_124),
               .O(c));
  OR2  XLXI_6 (.I0(LE),
               .I1(XLXN_123),
               .O(b));
  OR2  XLXI_7 (.I0(LE),
               .I1(XLXN_121),
               .O(a));
  INV  XLXI_8 (.I(point),
               .O(p));
```

```
89.   AND4  XLXI_10 (.I0(XLXN_110),
90.                  .I1(XLXN_112),
91.                  .I2(D2),
92.                  .I3(D3),
93.                  .O(XLXN_22));
94.   AND4  XLXI_11 (.I0(D0),
95.                  .I1(D1),
96.                  .I2(D2),
97.                  .I3(XLXN_117),
98.                  .O(XLXN_24));
99.   AND3  XLXI_12 (.I0(XLXN_112),
100.                 .I1(XLXN_116),
101.                 .I2(XLXN_117),
102.                 .O(XLXN_25));
103.      OR3  XLXI_19 (.I0(XLXN_22),
104.                    .I1(XLXN_24),
105.                    .I2(XLXN_25),
106.                    .O(XLXN_120));
107.      OR3  XLXI_20 (.I0(XLXN_86),
108.                    .I1(XLXN_87),
109.                    .I2(XLXN_88),
110.                    .O(XLXN_127));
111.      OR4  XLXI_21 (.I0(XLXN_80),
112.                    .I1(XLXN_81),
113.                    .I2(XLXN_82),
114.                    .I3(XLXN_85),
115.                    .O(XLXN_126));
116.      OR4  XLXI_23 (.I0(XLXN_91),
117.                    .I1(XLXN_92),
118.                    .I2(XLXN_93),
119.                    .I3(XLXN_94),
120.                    .O(XLXN_125));
121.      OR4  XLXI_25 (.I0(XLXN_101),
122.                    .I1(XLXN_102),
123.                    .I2(XLXN_104),
124.                    .I3(XLXN_105),
125.                    .O(XLXN_123));
126.      OR4  XLXI_26 (.I0(XLXN_96),
127.                    .I1(XLXN_85),
128.                    .I2(XLXN_93),
129.                    .I3(XLXN_94),
130.                    .O(XLXN_121));
131.      AND3  XLXI_32 (.I0(D0),
132.                     .I1(D1),
```

```
                    .I2(XLXN_117),
                    .O(XLXN_80));
    AND3  XLXI_33 (.I0(D1),
                    .I1(XLXN_116),
                    .I2(XLXN_117),
                    .O(XLXN_81));
    AND3  XLXI_34 (.I0(D0),
                    .I1(XLXN_116),
                    .I2(XLXN_117),
                    .O(XLXN_82));
    AND3  XLXI_35 (.I0(D0),
                    .I1(XLXN_112),
                    .I2(XLXN_116),
                    .O(XLXN_86));
    AND3  XLXI_36 (.I0(XLXN_112),
                    .I1(D2),
                    .I2(XLXN_117),
                    .O(XLXN_87));
    AND2  XLXI_38 (.I0(D0),
                    .I1(XLXN_117),
                    .O(XLXN_88));
    AND4  XLXI_39 (.I0(XLXN_110),
                    .I1(D1),
                    .I2(XLXN_116),
                    .I3(D3),
                    .O(XLXN_91));
    AND4  XLXI_40 (.I0(XLXN_110),
                    .I1(D1),
                    .I2(XLXN_116),
                    .I3(XLXN_117),
                    .O(XLXN_99));
    AND3  XLXI_41 (.I0(D0),
                    .I1(D1),
                    .I2(D2),
                    .O(XLXN_92));
    AND3  XLXI_42 (.I0(D1),
                    .I1(D2),
                    .I2(D3),
                    .O(XLXN_97));
    AND3  XLXI_43 (.I0(D0),
                    .I1(D1),
                    .I2(D3),
                    .O(XLXN_101));
    AND3  XLXI_44 (.I0(XLXN_110),
```

```
                    .I1(D2),
                    .I2(D3),
                    .O(XLXN_102));
    AND3  XLXI_45 (.I0(XLXN_110),
                    .I1(D1),
                    .I2(D2),
                    .O(XLXN_104));
    AND4  XLXI_46 (.I0(D0),
                    .I1(XLXN_112),
                    .I2(D2),
                    .I3(XLXN_117),
                    .O(XLXN_105));
    AND4  XLXI_79 (.I0(D0),
                    .I1(D1),
                    .I2(XLXN_116),
                    .I3(D3),
                    .O(XLXN_96));
    AND4  XLXI_80 (.I0(D0),
                    .I1(XLXN_112),
                    .I2(D2),
                    .I3(D3),
                    .O(XLXN_85));
    AND4  XLXI_115 (.I0(XLXN_110),
                     .I1(XLXN_112),
                     .I2(D2),
                     .I3(XLXN_117),
                     .O(XLXN_93));
    AND4  XLXI_116 (.I0(D0),
                     .I1(XLXN_116),
                     .I2(XLXN_112),
                     .I3(XLXN_117),
                     .O(XLXN_94));
    OR3  XLXI_137 (.I0(XLXN_97),
                    .I1(XLXN_99),
                    .I2(XLXN_102),
                    .O(XLXN_124));
    INV  XLXI_138 (.I(D0),
                    .O(XLXN_110));
    INV  XLXI_139 (.I(D1),
                    .O(XLXN_112));
    INV  XLXI_140 (.I(D2),
                    .O(XLXN_116));
    INV  XLXI_141 (.I(D3),
                    .O(XLXN_117));
```

```
221.    endmodule
```

4.1.6 仿真

```
1.  `timescale 1ns / 1ps
2.
3.  module MyMC14495_MyMC14495_sch_tb();
4.
5.  // Inputs
6.    reg LE;
7.    reg point;
8.    reg D0;
9.    reg D1;
10.   reg D2;
11.   reg D3;
12.
13. // Output
14.   wire g;
15.   wire p;
16.   wire a;
17.   wire b;
18.   wire c;
19.   wire d;
20.   wire e;
21.   wire f;
22.
23. // Bidirs
24.
25. // Instantiate the UUT
26.   MyMC14495 UUT (
27.       .g(g),
28.       .LE(LE),
29.       .point(point),
30.       .p(p),
31.       .a(a),
32.       .b(b),
33.       .c(c),
34.       .d(d),
35.       .e(e),
36.       .f(f),
37.       .D0(D0),
38.       .D1(D1),
39.       .D2(D2),
40.       .D3(D3)
41.   );
```

```
// Initialize Inputs
  integer i;
   initial begin
      D3 = 0;
      D2 = 0;
      D1 = 0;
      D0 = 0;
      LE = 0;
      point = 0;
      for (i=0; i<=15;i=i+1) begin
         #50;
         {D3,D2,D1,D0}=i;
         point = i;
      end
      #50;
      LE = 1;
   end
endmodule
```

4.1.7 生成逻辑符号图和VF文件。

4.2 实现数码管显示

4.2.1 新建工程DispNumber_sch。

4.2.2 新建schematic文件DispNumber_sch。

4.2.3 复制MyMC14495.sym和.vf到工程根目录。

4.2.4 调用MyMC14495

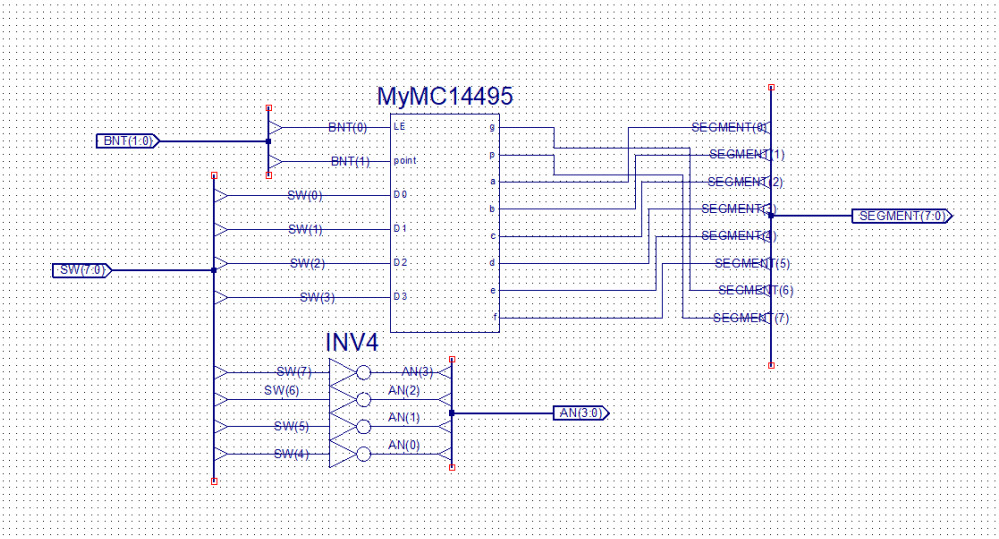


图6 数码管显示示意图

4.2.5 UCF引脚约束

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

NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;#a_out
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b_out
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;#c_out
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;#d_out
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;#e_out
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;#f_out
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g_out
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;
```

4.2.6 下载验证。

五、实验结果与分析

5.1 MyMC14495仿真波形激励结果

5.1.1 MyMC14495仿真图

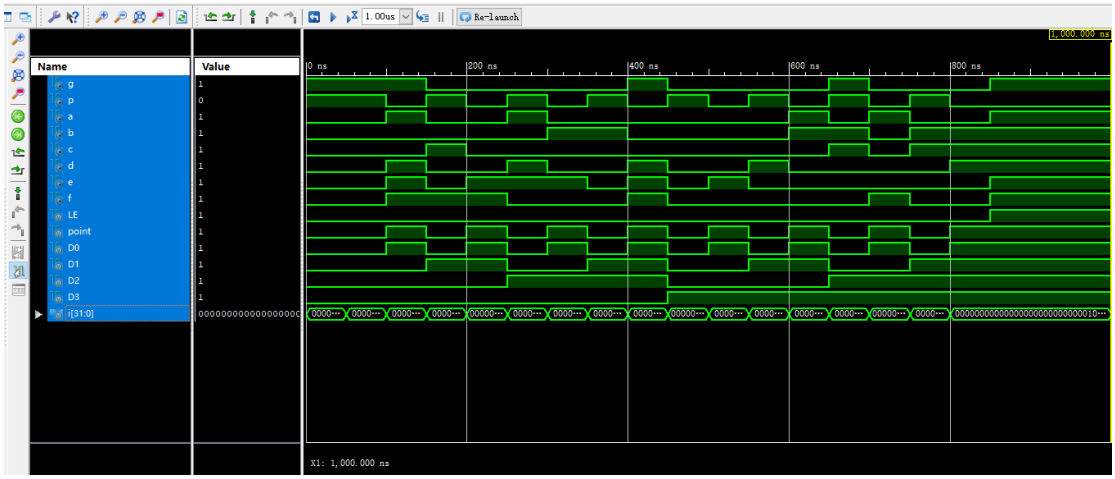


图7 MyMC14495仿真波形示意图

5.1.2 仿真图分析

通过输入和输出可以看出，当D0、D1、D2、D3 为 0000 时，abcdef为0且g为1，由于数码管设计为共阳低电平有效，对应数码管即除中间不亮外其余亮起表示数字 0；当D0、D1、D2、D3为1000时，只有bc为0其余为1，对应数码管即只有右侧两个数码管亮起表示数字1；对剩余情况一一对应可以看出数码管表示与{D3 D2 D1 D0}的二进制表示数字一致，符合电路设计逻辑。

5.2 SWORD开发板下载验证结果

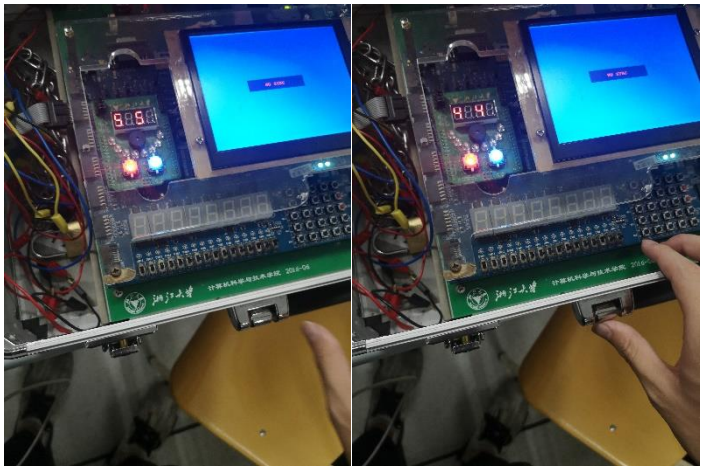


图8 实验结果照片节选

实验结果分析：在用SWORD开发板验证MyMC14495过程中，左边第一个开关负责控制小数点，第二个开关是使能开关。右边靠左侧的四个开关负责控制数码管是否亮起，最右侧四个开关负责控制显示数字大小，通过二进制代码转换显示对应的数字。与预期实验结果一致。

六、讨论与心得

本次实验在画器件内部原理图的部分消耗了非常多时间，在画好之后因为一些细节问题前后进行了几次的改动。因此在以后的实验中还需要更加注意画图的细节，避免出现一些低级错误。在连接和调试时因为对引脚约束的不熟悉导致对实现的功能使用时还不能熟练使用。通过这次实验我较好地完成了实验目的，也对ISE平台有了更深入的理解。

实验七——多路选择器设计及应用

姓名：张琦	学号：3180103162	专业：软件工程
课程名称：数字逻辑设计	同组学生：无	实验日期：2019-10-30
实验地点：紫金港东 4-509	指导老师：洪奇军	

一、实验目的和要求

1. 掌握数据选择器的工作原理和逻辑功能
2. 掌握数据选择器的使用方法
3. 掌握4位数码管扫描显示方法
4. 4位数码管显示应用—记分板设计

二、实验内容和原理

2.1 实验内容

2.1.1 数据选择器设计

2.1.2 计分板设计

2.2 实验原理

2.2.1 四选一多路选择器：MUX4to1

根据事件简化真值表

输出控制信号全部最小项与或结构

信息输入	控制端	选择输出	
I0 I1 I2 I3	S1 S0	o	输出项
I0 I1 I2 I3	0 0	I0	S1S0 I0
I0 I1 I2 I3	0 1	I1	S1S0 I1
I0 I1 I2 I3	1 0	I2	S1S0 I2
I0 I1 I2 I3	1 1	I3	S1S0 I3

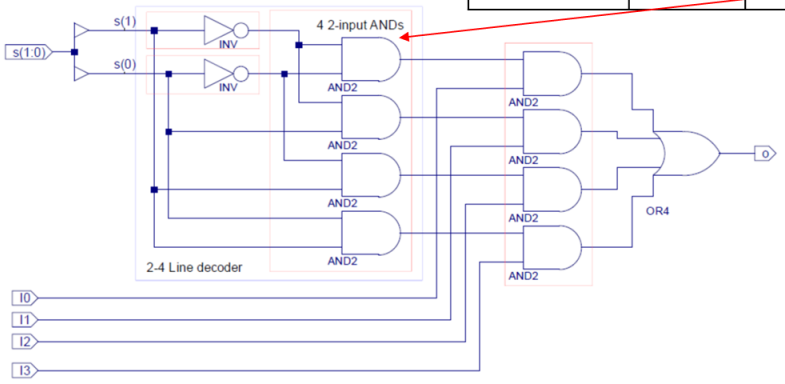


图1 MUX4to1示意图与真值表

2.2.2 多路选择器位扩展

控制结构不变，每路输入向量化

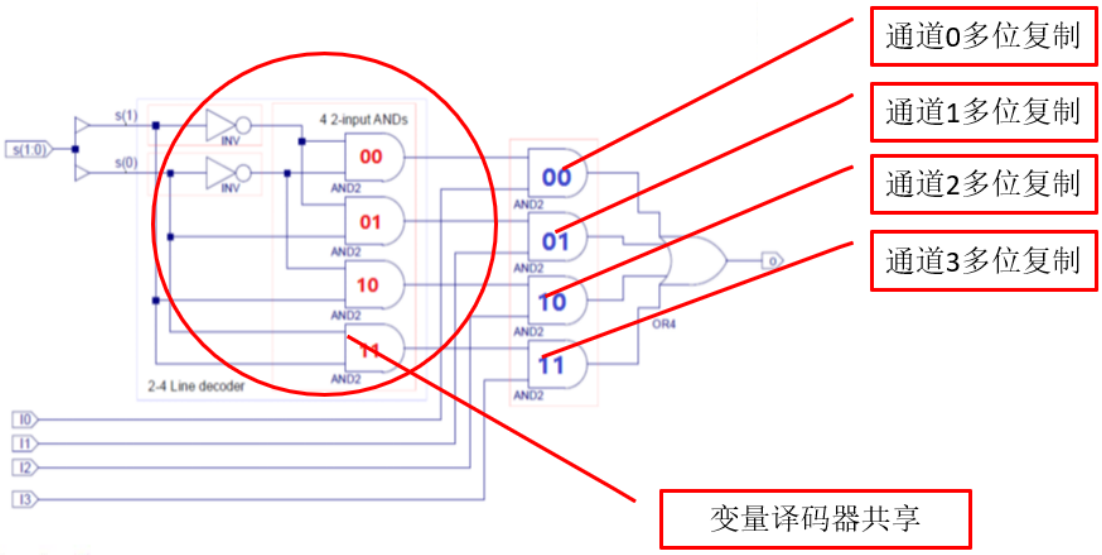


图2 多路选择器位扩展示意图

2.2.3 4位四选一扩展：MUX4to1b4

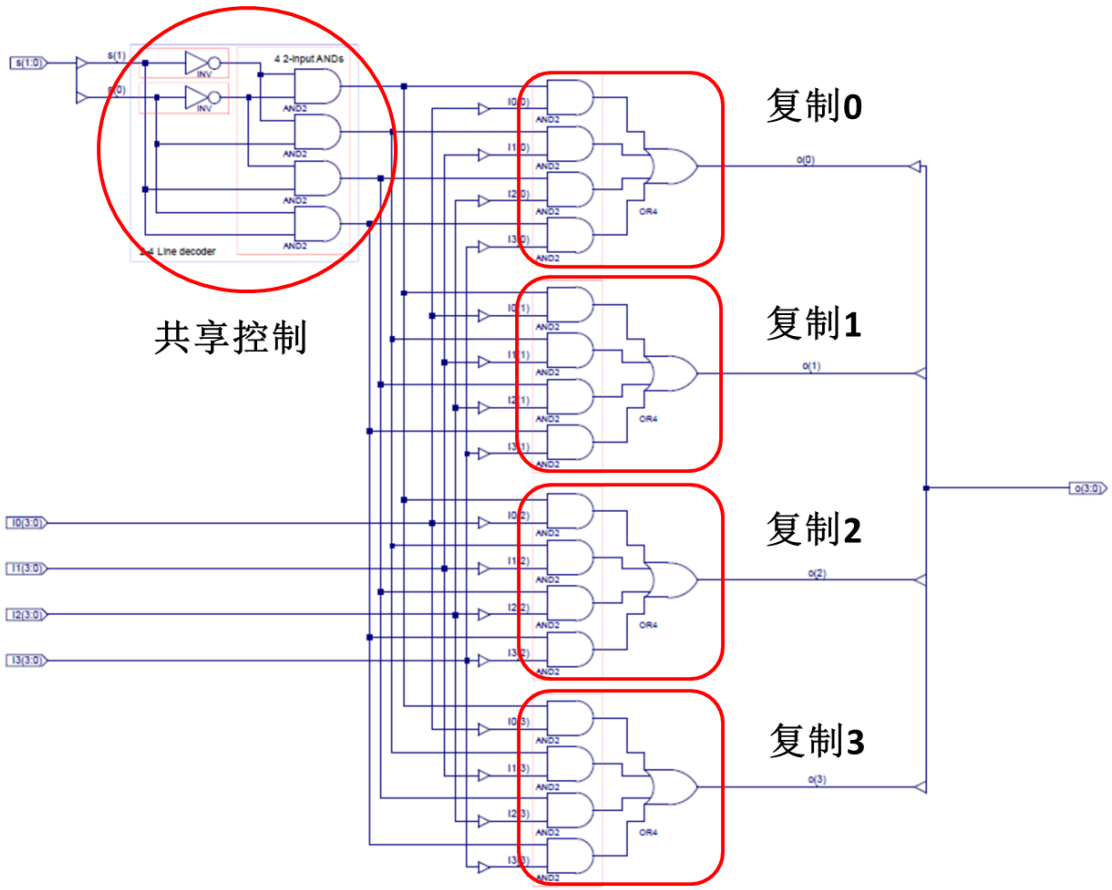


图3 4位四选一扩展：MUX4to1b4示意图

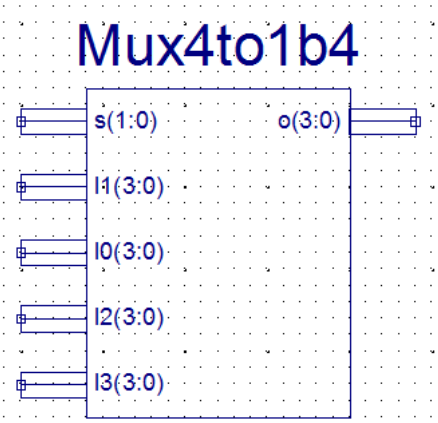


图4 MUX4to1b4模块图

2.2.4 动态扫描显示方案

扫描信号来自计数器：时序转化为组合电路。由板载时钟clk(100MHz)作为计数器时钟，分频后输入到数据选择器的控制端，作为数码管扫描信号。计数器的分频系数要适当，眼睛舒适即可。

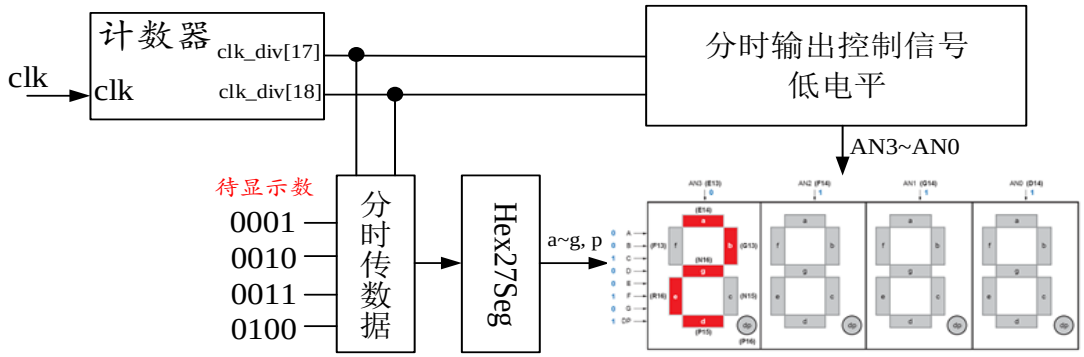


图5 动态扫描显示原理图

2.2.5 多路选择器应用

通过运用4位4选1多路选择器和动态扫描显示，我们可以使 4个七段数码管显示四个不一样的数值。

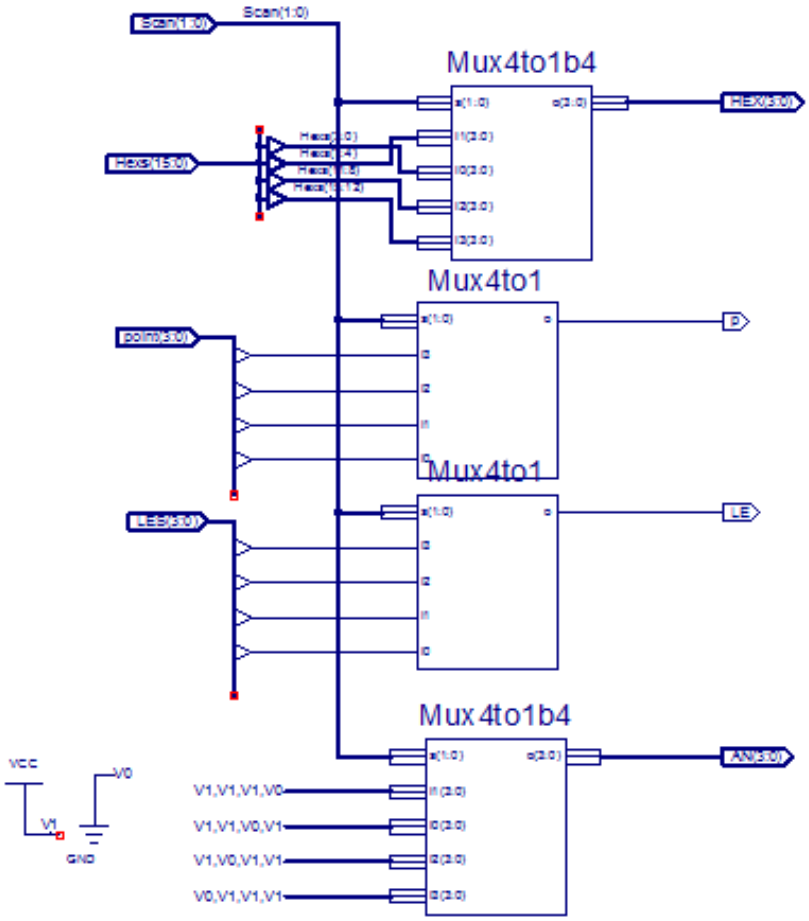


图6 DisplaySync示意图

2.2.6 时钟计数分频器

32位时钟计数分频器：可输出2-2^{32}分频信号，可用于一般非同步类时钟信号延时较高，要求不高的时钟也可以用。本实验多位七段显示器动态扫描可用。

2.2.7 设计disp_num显示模块

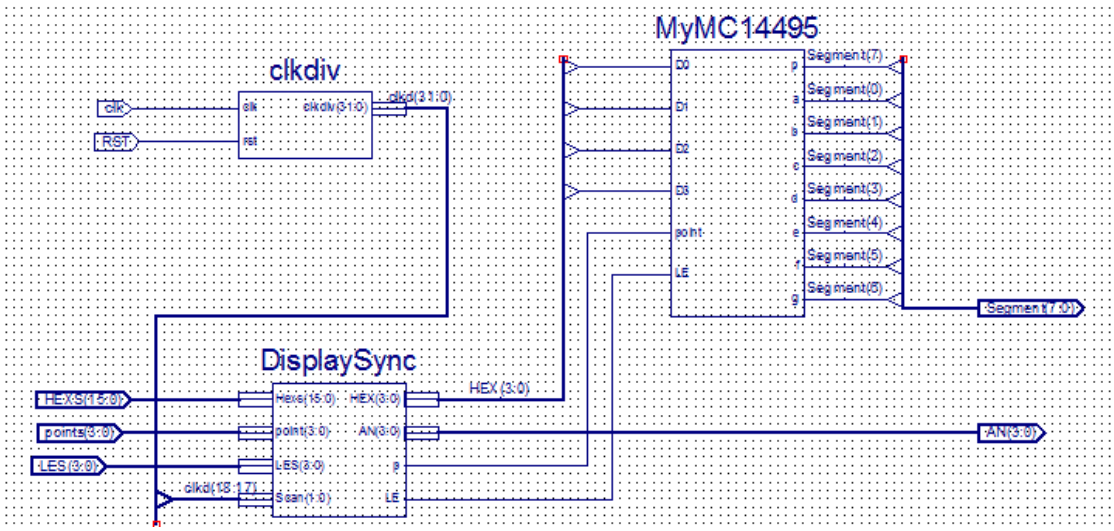


图7 disp_num示意图

2.2.8 设计按键输入模块

三、主要仪器设备

1.装有Xilinx ISE 14.7的计算机1台

2.SWORD开发板

四、操作方法与实验步骤

4.1 数据选择器设计

4.1.1 新建工程，工程名称用Mux4to1b4_sch。

4.1.2 新建源文件，类型是Schematic，文件名称用Mux4to1b4。

4.1.3 原理图方式进行设计。

4.1.4 Check Design Rules，检查错误。

4.1.5 View HDL Functional Model，查看并学习 Verilog HDL 代码。

4.1.6 仿真

4.2 记分板应用设计

4.2.1 新建工程

工程名称用ScoreBoard。

Top Level Source Type用HDL。

new source文件名称：DisplaySync.sch。

绘制原理图实现DisplaySync：

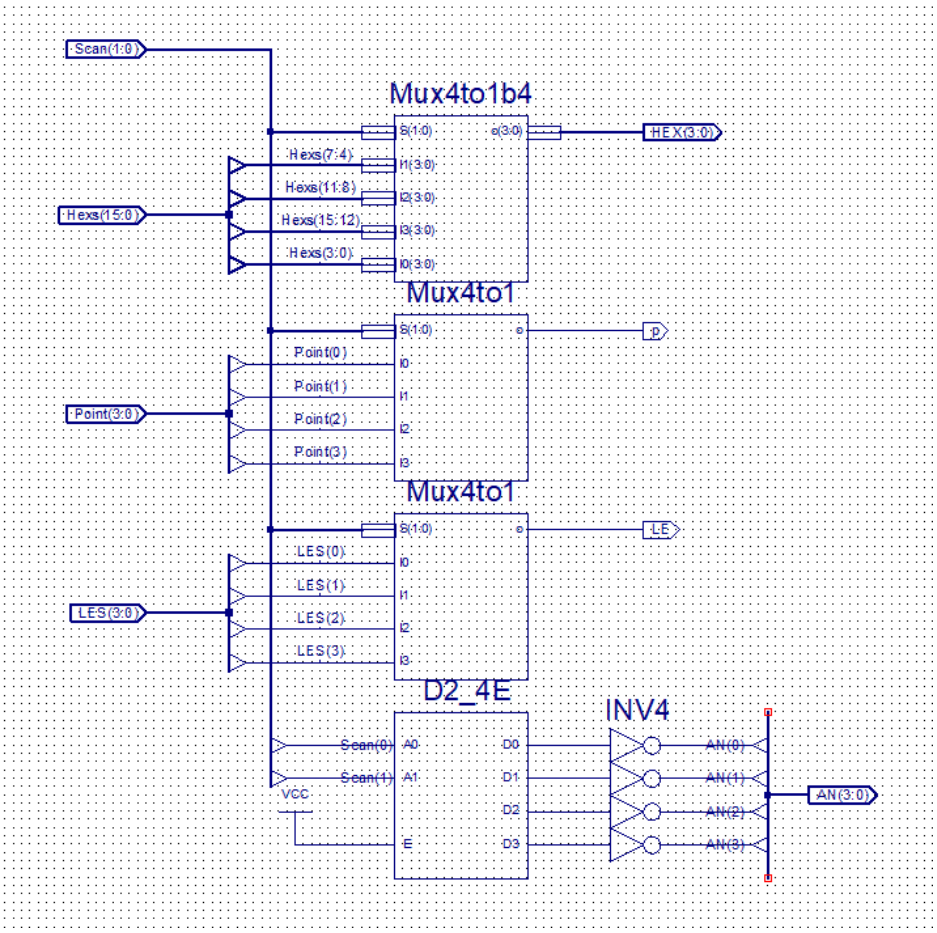


图8 原理图实现DisplaySync

Verilog代码（case语句）实现DisplaySync:

```
`timescale 1ns / 1ps
module dispsync(input [15:0]Hexs,
                input [1:0]Scan,
                input [3:0]point,
                input [3:0]LES,
                output reg[3:0]Hex,
                output reg p,LE,
                output reg[3:0]AN
    );
     always @*begin
          case(Scan)
            2'b00: begin Hex<=Hexs[3:0]; AN<=4'b 1110; p<=point[0];LE<=LES[0];end
            2'b01: begin Hex<=Hexs[7:4]; AN<=4'b 1101; p<=point[1];LE<=LES[1];end
            2'b10: begin Hex<=Hexs[11:8];   AN<=4'b 1011; p<=point[2];LE<=LES[2];end
            2'b11: begin Hex<=Hexs[15:12];   AN<=4'b 0111; p<=point[3];LE<=LES[3];end
          endcase

    end

endmodule
```

4.2.2 根据原理设计动态扫描同步输出模块，文件名称clkdiv.v。

Verilog代码（clkdiv.v）：

```
`timescale 1ns / 1ps
module clkdiv(input clk,
              input rst,
              output reg[31:0]clkdiv
             );
// Clock divider-时钟分频器

    always @ (posedge clk or posedge rst) begin
      if (rst) clkdiv <= 0;
      else clkdiv <= clkdiv + 1'b1;
    end

endmodule
```

4.2.3 根据原理设计通用计数分频模块

新建源文件，类型是Schematic，文件名称用disp_num。

原理图方式进行设计显示模块。

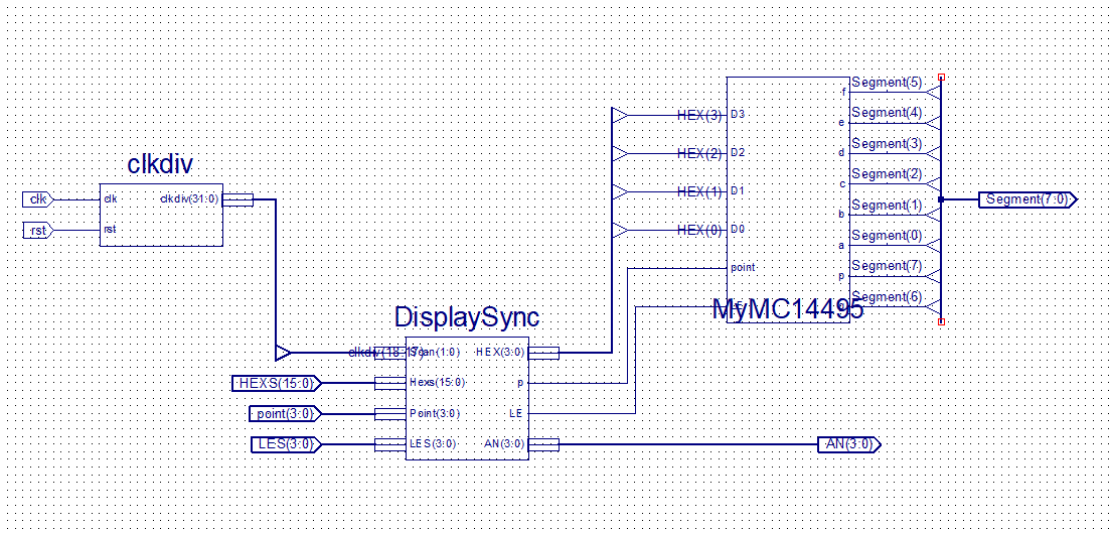


图9 disp_num原理图设计显示模块

4.2.4 新建源文件top,并右键设为“Top Module”。

Verilog代码（top.v）：

```
module top(
    input wire clk,
    input wire [7:0] SW,
    input wire [3:0] btn,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
    );
    wire [15:0] num;

    CreateNumber c0(btn,num);

    disp_num d0(clk, num, SW[7:4], SW[3:0], 1'b0, AN, SEGMENT);

endmodule
```

4.2.5 CreateNumber模块设计

四个按键，各按一下，4个4位2进制数分别加1。

Verilog代码（CreateNumber.v）：

```
module CreateNumber(
    input wire [3:0] btn,
    output reg [15:0] num
```

```
    );
    wire [3:0] A,B,C,D;

    initial num <= 16'b1010_1011_1100_1101;

    assign A = num[ 3: 0] + 4'd1;
    assign B = num[ 7: 4] + 4'd1;
    assign C = num[11: 8] + 4'd1;
    assign D = num[15:12] + 4'd1;

    always@(posedge btn[0]) num[ 3: 0]<= A;
    always@(posedge btn[1]) num[ 7: 4]<= B;
    always@(posedge btn[2]) num[11: 8]<= C;
    always@(posedge btn[3]) num[15:12]<= D;

endmodule
```

4.2.6 UCF引脚约束

```
NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18 ;

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 = AF8 | IOSTANDARD = LVCMOS15;#SW[12]
NET "btn[1]" LOC = AE13 | IOSTANDARD = LVCMOS15;#SW[13]
NET "btn[2]" LOC = AF13 | IOSTANDARD = LVCMOS15;#SW[14]
NET "btn[3]" LOC = AF10 | IOSTANDARD = LVCMOS15;#SW[15]
NET "btn[0]" CLOCK_DEDICATED_ROUTE = FALSE;
NET "btn[1]" CLOCK_DEDICATED_ROUTE = FALSE;
NET "btn[2]" CLOCK_DEDICATED_ROUTE = FALSE;
NET "btn[3]" CLOCK_DEDICATED_ROUTE = FALSE;

NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;#a_out
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b_out
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;#c_out
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;#d_out
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;#e_out
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;#f_out
```

```
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g_out
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;
```

4.2.7 下载验证。

五、实验结果与分析

SWORD开发板下载验证结果

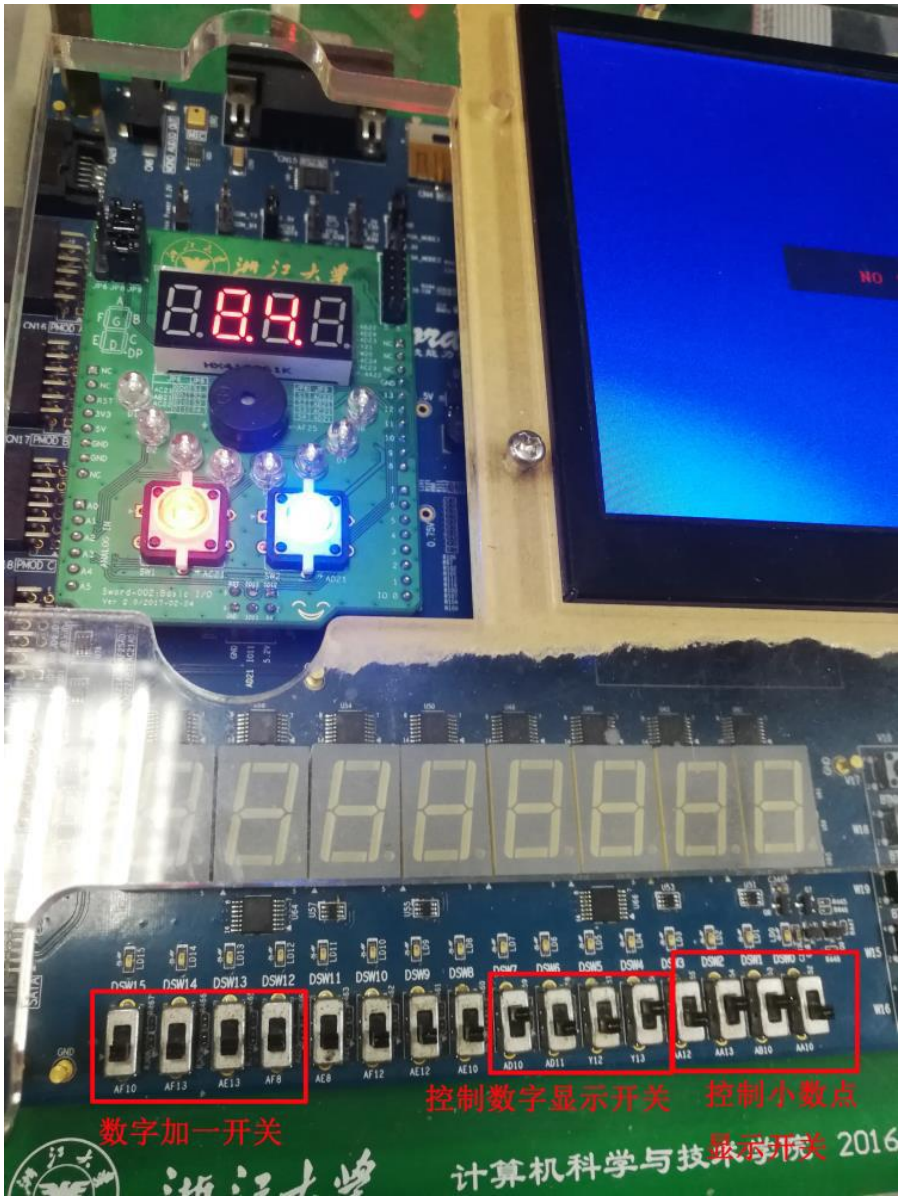
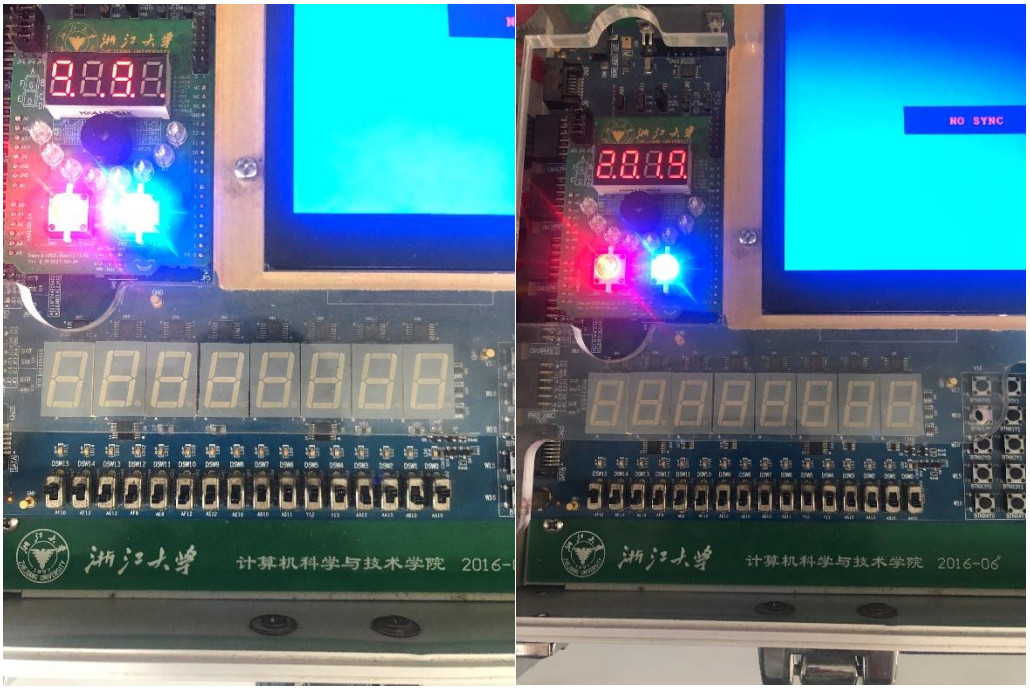


图10 实验结果照片节选

由引脚文件可知，最右侧的四个开关为控制小数点显示的开关，此时中间两个开关为高电平，两侧开关为低电平，与数码管中间两个小数点亮起相对应。控制小数点的开关左侧的四个开关为控制数字显示的开关，为低电平有效，此时两侧开关为高电平，对应数码管中只有中间两个数码管亮起。符合电路的功能逻辑。



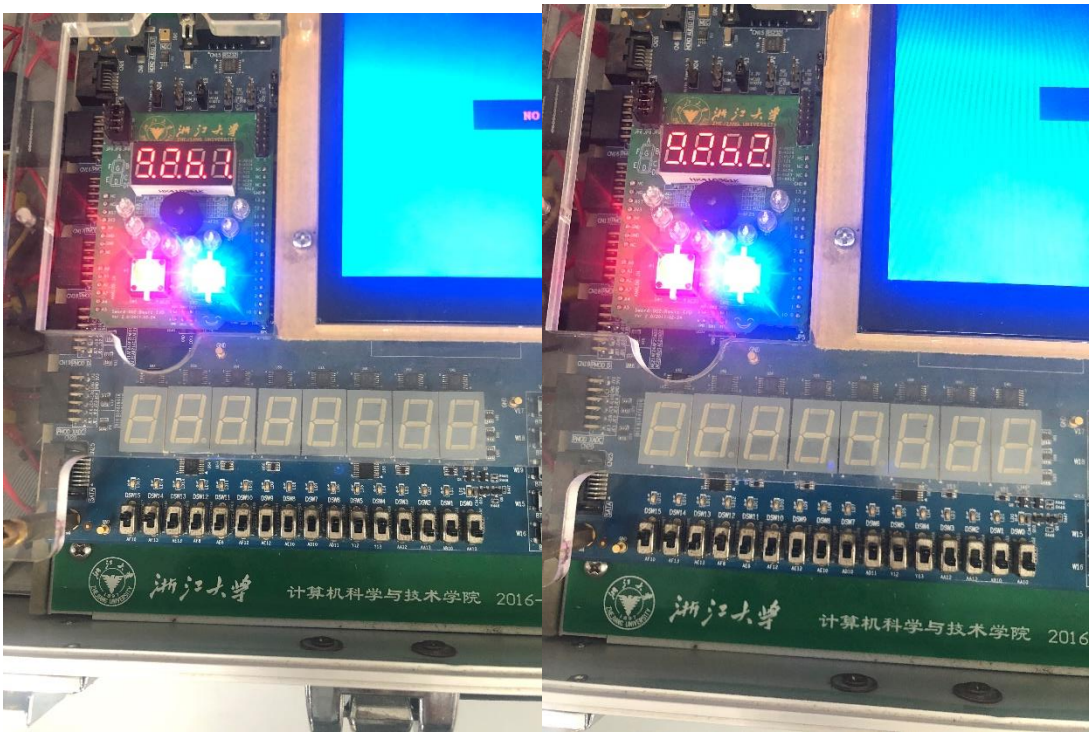


图11 实验照片节选

实验结果分析：数字能正常显示并能正常进行加一操作。与设计需求符合。

六、讨论与心得

本次实验运用了大量的模块，让我一开始也觉得十分难以入手，但当我着手完成时，我逐渐发现了这样自顶而下的思想的优越之处。我的大脑中有清晰的结构，进行模块化设计时能清楚地知道我在做什么，这些输入输出端应该怎么留，留在那。所以虽然说这次电路比前面几次都复杂得多，但是我做的明显比前边几次做得快。