实验五—变量译码器设计与应用实验报告

姓名：程梦烨 学号：3180103733 专业：软件工程

课程名称：逻辑与计算机设计基础实验 同组学生姓名：

实验时间：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

1.1 掌握变量译码器的逻辑构成和逻辑功能。
1.2 用变量译码器实现组合函数
1.3 掌握变量译码器的典型应用（地址译码的具体方法）
1.4 了解存储器编址的概念
1.5 采用原理图设计电路模块
1.6 进一步熟悉 ISE 平台及下载实验平台物理实验

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

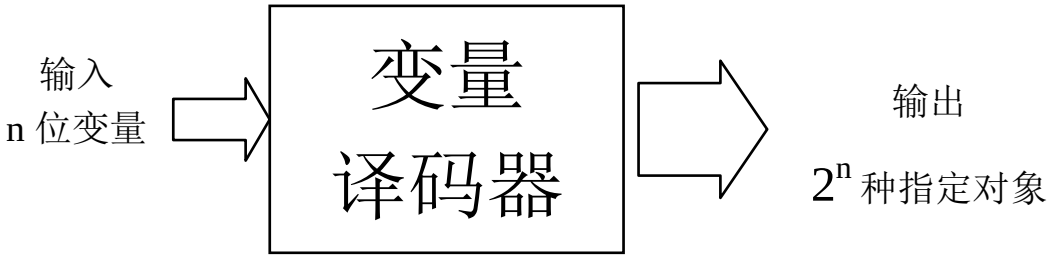
2.2 实验原理

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 2^n 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。



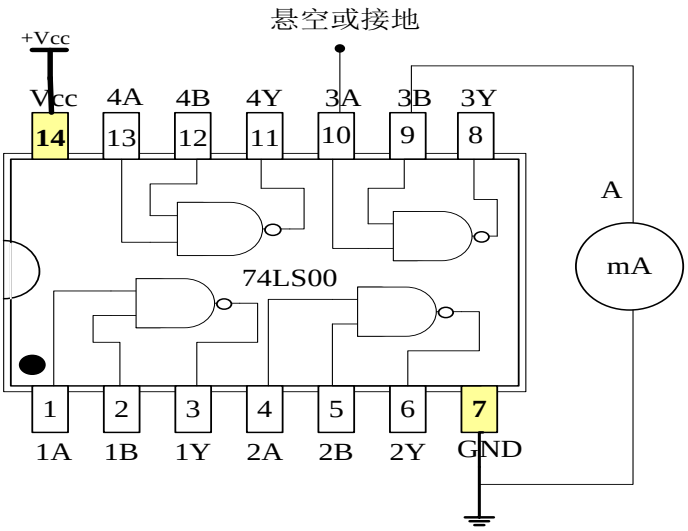
图表 1 变量译码器功能图

2.2.3 74LS138 变量译码器

- 74LS138 变量译码器是一个带 3 个使能端的 3-8 译码器，其逻辑结构由三级门电路构成，输出低电平有效。

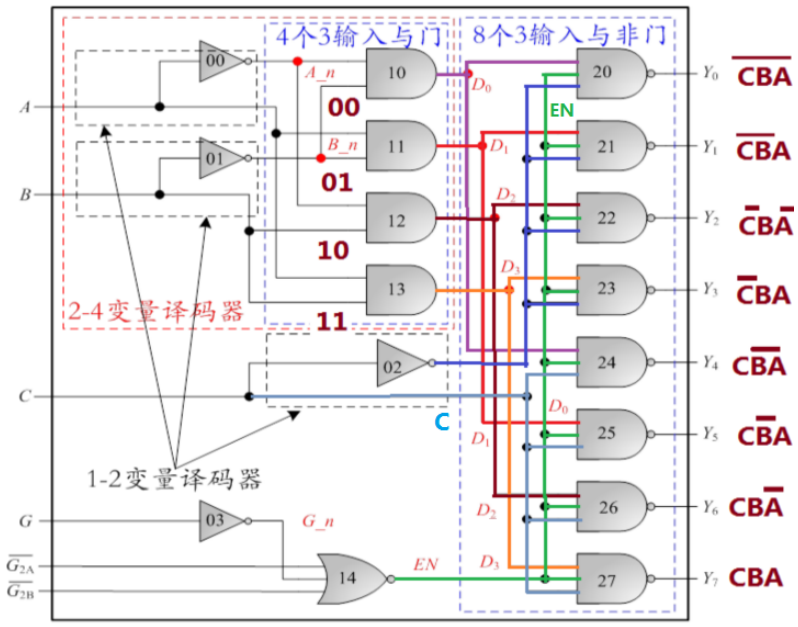
输入		译码器输出（低电平有效）							
使能	变量								
$G\overline{G_{2A}}\overline{G_{2B}}$	CBA	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$
×11	×××	1	1	1	1	1	1	1	1
0××	×××	1	1	1	1	1	1	1	1
100	000	**0**	1	1	1	1	1	1	1
100	001	1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

图表 2 74LS138 变量译码器功能表



图表 3 74LS138 变量译码器引脚

- 74LS138 变量译码器连线明细图



图表 4 变量译码器 74LS138 连线明细图

2.2.4 用变量译码器实现组合函数

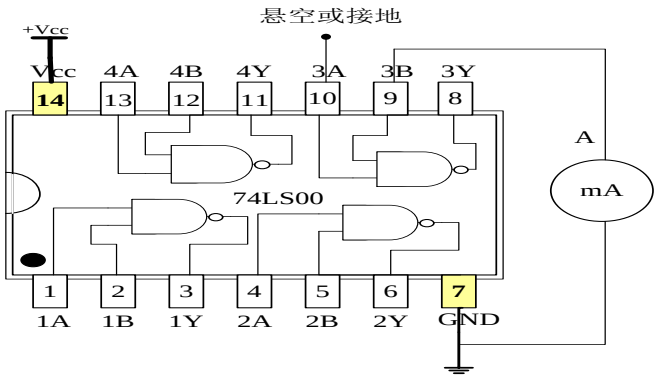
- 变量译码器的输出对应所有输入变量的最小项组合，如果将函数转换成最小项和的形式，则可以用变量译码器实现函数的组合电路:

  F = S3S2S1+S3S2S1+S3S2S1+S3S2S1

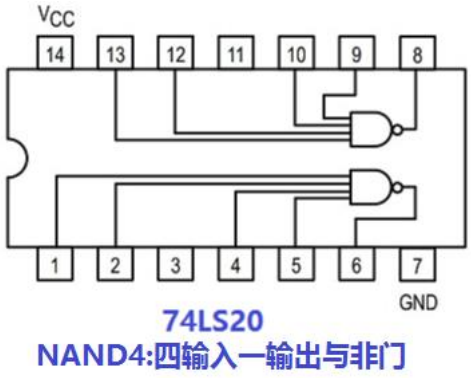
  F = 001 + 010 + 100 + 111

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

图表 5 楼道灯功能真值表



图表 6 用 74LS138 实现组合函数

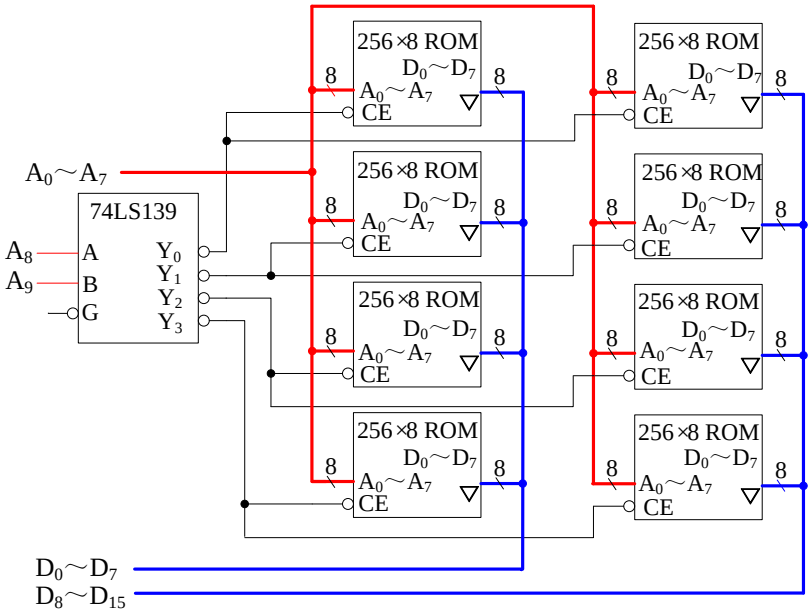


图表 7 用 74LS138 实现组合函数

2.2.5 变量译码器实现存储器地址译码

- 存储器电路中地址译码的意义：
  - 在容量扩展时，将不同的芯片分配到不同地址段，来达到更大的存储容量（寻址范围）
  - 在容量扩展时，将不同的芯片分配在同一地址段，来达到更大的存储单元（存储字）
- 地址译码原理：将访问存储器的地址线高位作为译码器的输入，译码器的输出控制各存储器的片选信号。
- 字扩展：译码器的不同输出连接到不同存储芯片的片选端。
- 位扩展：译码器的同一输出连接到不同存储芯片的片选端。

- 同时进行字扩展和位扩展



图表 8 变量译码器实现存储器地址译码

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 1 套

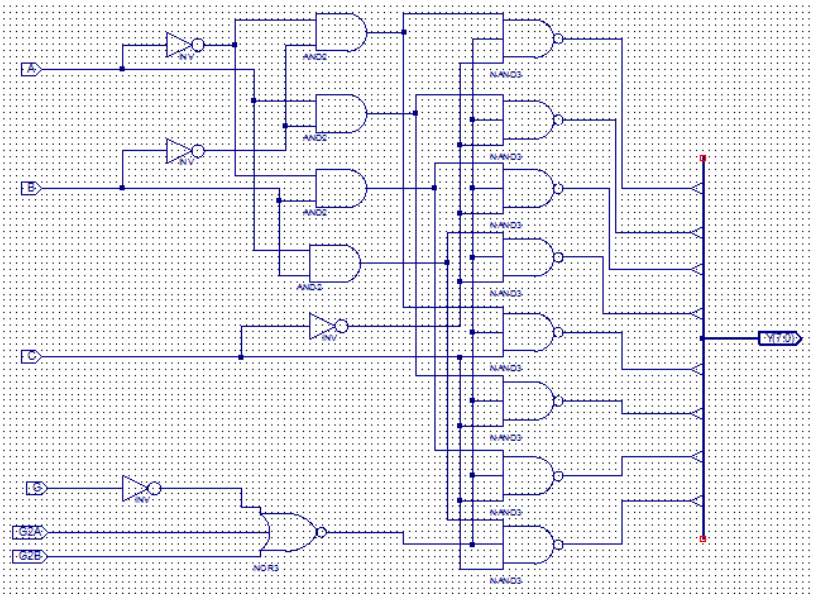
四、操作方法与实验步骤

4.1 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

4.1.2 新建 Schematic 源文件，文件名称用 D_74LS138

4.1.3 按照原理图设计 D_74LS138



图表 9 D_74LS138 在 ISE 中的连线图

4.1.4 Check Design Rules，检查错误

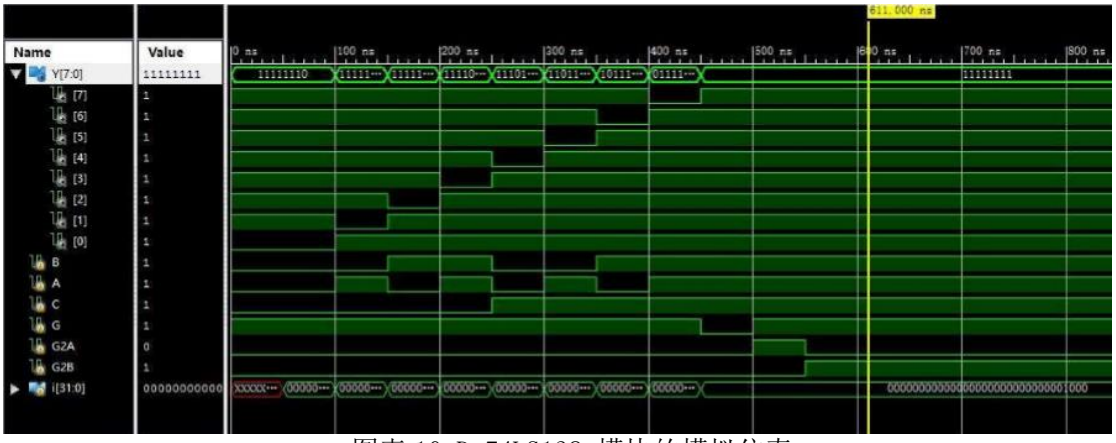
4.1.5 View HDL Functional Model，查看并学习 Verilog HDL 代码

4.1.6 模拟仿真

对 D_74LS138 模块进行仿真，激励代码如下（initial 主要代码）

```
integer i;
initial begin
B = 0;
A = 0;
C = 0;
G = 1;
G2A = 0;
G2B = 0;
#50;

for (i=0;i<=7;i=i+1) begin
{C,B,A}=i;
#50;
end

assign G=0;
assign G2A=0;
assign G2B=0;
#50;

assign G=1;
assign G2A=1;
assign G2B=0;
#50;

assign G=1;
assign G2A=0;
assign G2B=1;
#50;
End
```

仿真结果如下：



图表 10 D_74LS138 模块的模拟仿真

在模拟仿真中，当 C、B、A 的值从 000 依次变化到 111 时，对应的 LED 灯依次点亮，其他的 LED 灯灭，符合预期，成功实现了 D_74LS138 的功能。

4.1.7 生成逻辑符号图和 VF 文件

- Create Schematic Symbol，系统生成 D_74LS138 模块的逻辑符号图文件，文件后缀.sym
- 符号图位于工程根目录
  - 自动生成的符号可修改：可以用 Tools 菜单的 Symbol Wizard，也可以打开.sym 文件直接修改
  - 使用时，把.sym 和.vf（或.v）复制到对应工程目录

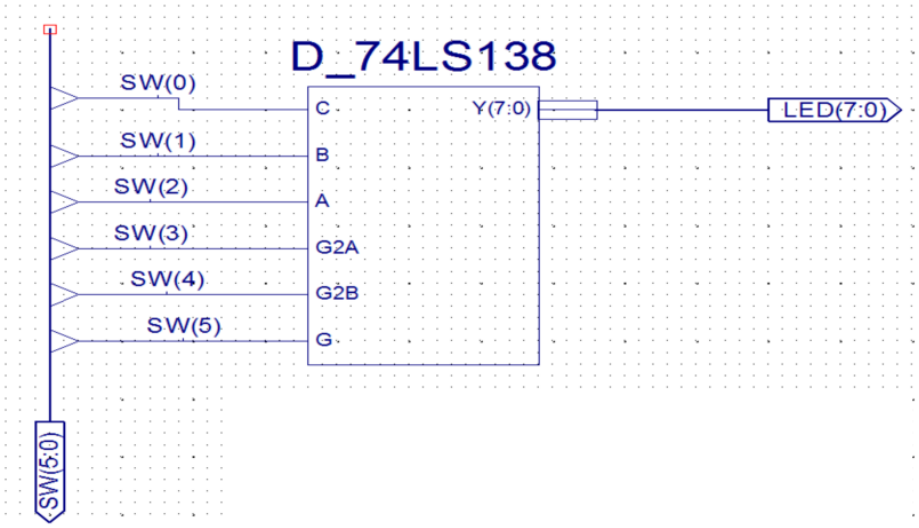
4.2 验证 D_74LS138

4.2.1 新建工程“D_74LS138_Test”（第二个工程）

4.2.2 新建 Schematic 文件“D_74LS138_Test”

4.2.3 复制 D_74LS138.sym 和.vf 到工程目录。

4.2.4 按照原理图设计 D_74LS138_Test



图表 11 D_74LS138_Test 原理图

4.2.5 下载验证

UCF 引脚定义：

输入用 6 个开关

3 个译码输入：SW[2:0]　AA13,AB10,AA10

3 个使能控制：SW[5:3]　Y12,Y13,AA12

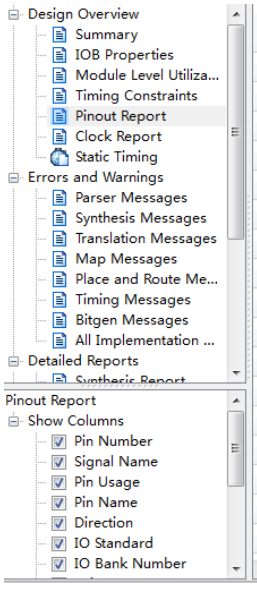
输出用 8 个 LED 灯：LED[7:0]

引脚约束代码如下：

```
NET"SW[2]"LOC=AA10 | IOSTANDARD=LVCMOS15;#A
NET"SW[1]"LOC=AB10 | IOSTANDARD=LVCMOS15;#B
NET"SW[0]"LOC=AA13 | IOSTANDARD=LVCMOS15;#C
NET"SW[5]"LOC=AA12 | IOSTANDARD=LVCMOS15;#G
NET"SW[4]"LOC=Y13 | IOSTANDARD=LVCMOS15;#G2B
NET"SW[3]"LOC=Y12 | IOSTANDARD=LVCMOS15;#G2A

NET"LED[0]"LOC=W23  | IOSTANDARD=LVCMOS33 ;#D1
NET"LED[1]"LOC=AB26 | IOSTANDARD=LVCMOS33 ;#D2
NET"LED[2]"LOC=Y25  | IOSTANDARD=LVCMOS33 ;#D3
NET"LED[3]"LOC=AA23 | IOSTANDARD=LVCMOS33 ;#D4
NET"LED[4]"LOC=Y23  | IOSTANDARD=LVCMOS33 ;#D5
NET"LED[5]"LOC=Y22  | IOSTANDARD=LVCMOS33 ;#D6
NET"LED[6]"LOC=AE21 | IOSTANDARD=LVCMOS33 ;#D7
NET"LED[7]"LOC=AF24  | IOSTANDARD=LVCMOS33 ;#D8
```

4.2.6 设计实现并检查约束结果

在 Sources 窗口中选择 Synthesis / Implementation，选中 lamp_ctrl；在 Processes 窗口下选择 Implement Design，进行物理转换、平面布图、映射、物理布线等 FPGA 目标格式实现文件生成。最后在设计摘要文档中有如下结果：



	Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termir
1	AF24	Buzzer	IOB33	IO_L20P_T3_12	OUTPUT	LVCMO...	12	12	SL...	
2	AB26	LED<0>	IOB33	IO_L9P_T1_DQS_12	OUTPUT	LVCMO...	12	12	SL...	
3	W24	LED<1>	IOB33	IO_L8N_T1_12	OUTPUT	LVCMO...	12	12	SL...	
4	W23	LED<2>	IOB33	IO_L8P_T1_12	OUTPUT	LVCMO...	12	12	SL...	
5	AB25	LED<3>	IOB33	IO_L7N_T1_12	OUTPUT	LVCMO...	12	12	SL...	
6	AA25	LED<4>	IOB33	IO_L7P_T1_12	OUTPUT	LVCMO...	12	12	SL...	
7	W21	LED<5>	IOB33	IO_L6N_T0_VREF_12	OUTPUT	LVCMO...	12	12	SL...	
8	V21	LED<6>	IOB33	IO_L6P_T0_12	OUTPUT	LVCMO...	12	12	SL...	
9	W26	LED<7>	IOB33	IO_L5N_T0_12	OUTPUT	LVCMO...	12	12	SL...	
10	AA10	S1	IOB	IO_L14P_T2_SRCC_33	INPUT	LVCMO...	33			
11	AB10	S2	IOB	IO_L14N_T2_SRCC_33	INPUT	LVCMO...	33			
12	AA13	S3	IOB	IO_L16P_T2_33	INPUT	LVCMO...	33			
13	AA12	S4	IOB	IO_L16N_T2_33	INPUT	LVCMO...	33			
14	Y13	S5	IOB	IO_L18P_T2_33	INPUT	LVCMO...	33			
15	Y12	S6	IOB	IO_L18N_T2_33	INPUT	LVCMO...	33			
16	A1			GND						
17	A2			GND						

图表 12 Pinouts Report

4.2.7 根据真值表验证

根据真值表，操作实验板，验证功能。

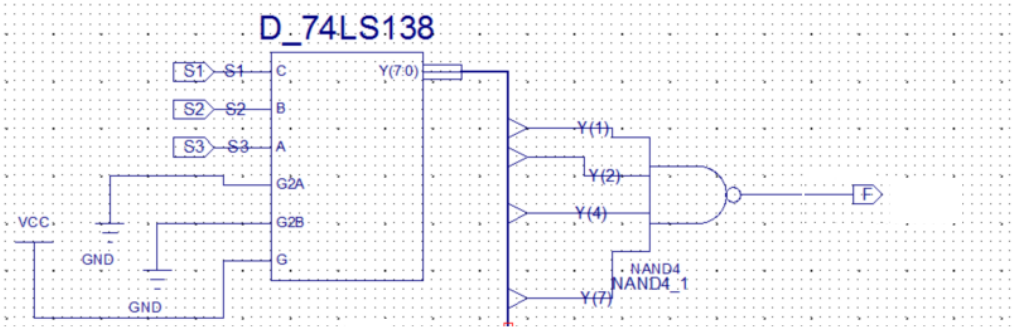
4.3 用 74LS138 译码器实现楼道灯控制

4.3.1 新建工程 LampCtrl138（第三个工程）

4.3.2 复制 D_74LS138.sym 和.vf 文件到工程目录。

只有加入到了工程目录中，在设计实现楼道灯控制时，才可以直接使用之前完成的 74LS138 元件进行设计。

4.3.3 按照原理图设计楼道灯控制逻辑电路



图表 13 楼道灯控制逻辑电路原理图

4.3.4 模拟仿真

对于楼道灯控制逻辑电路进行仿真，激励代码如下（initial 主要代码）：

```
integer i;
initial begin
    for(i=0;i<=8;i=i+1)begin
        {S3,S2,S1} <= i;
        #50;
    end
end
```

仿真结果如下：

图表 14 楼道灯控制逻辑电路模拟仿真

在模拟仿真中，只有一个 S1、S2、S3 中全部为 0 时，F 输出 0，LED 灯灭；只 有一个为 1 时，F 输出 1，LED 亮；有两个为 1 时，F 输出 0，LED 灯灭；全部为 1 时，F 输出位 1，LED 灯亮；符合楼道灯的功能要求。

4.3.5 下载验证

UCF 引脚定义：

输入用 3 个开关：S1,S2,S3

输出用 8 个 LED,蜂鸣器：F,LED(6:0),Buzzer

引脚约束代码如下：

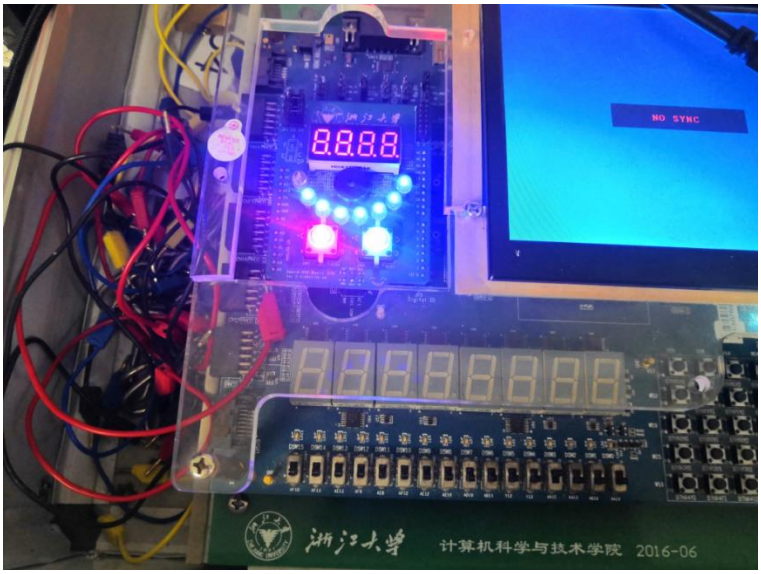
```
NET "S1" LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "S2" LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "S3" LOC = AA13 | IOSTANDARD = LVCMOS15;
NET "F"  LOC = W26 | IOSTANDARD = LVCMOS33;
NET "Buzzer" LOC = AF24 | IOSTANDARD = LVCMOS33;
NET "LED[0]"  LOC = AB26 | IOSTANDARD = LVCMOS33;
NET "LED[1]"  LOC = W24 | IOSTANDARD = LVCMOS33;
NET "LED[2]"  LOC = W23 | IOSTANDARD = LVCMOS33;
NET "LED[3]"  LOC = AB25 | IOSTANDARD = LVCMOS33;
NET "LED[4]"  LOC = AA25 | IOSTANDARD = LVCMOS33;
NET "LED[5]"  LOC = W21 | IOSTANDARD = LVCMOS33;
NET "LED[6]"  LOC = V21 | IOSTANDARD = LVCMOS33;
```

4.3.6 在 SWORD 开发板上验证是否实现了楼道灯控制的功能

根据真值表，操作实验板，验证功能。

五、实验结果与分析

5.1 设计实现 74LS138 译码器模块

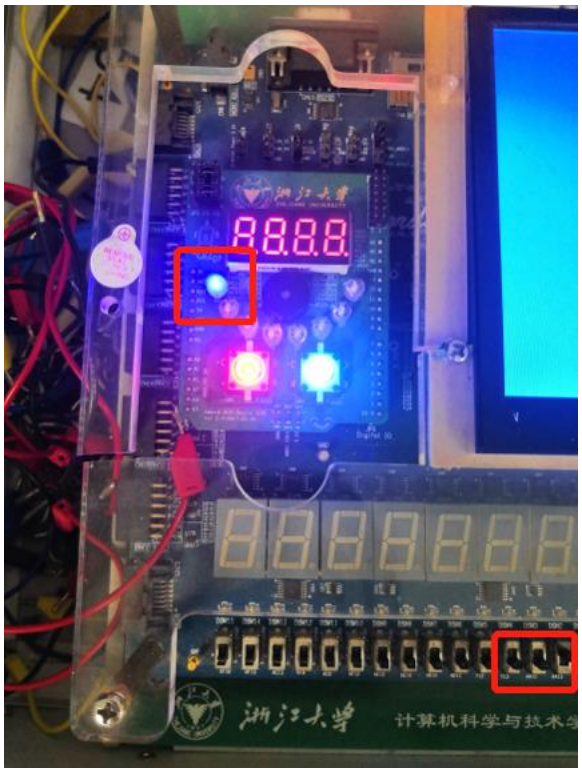


图表 15 验证 74LS138 译码器模块

经过实验验证，成功实现了 74LS139 译码器模块的功能，当开关从 000 依次变化到 111 时，对应的一盏 LED 灯点灭，其他七盏 LED 灯均亮，符合预期。

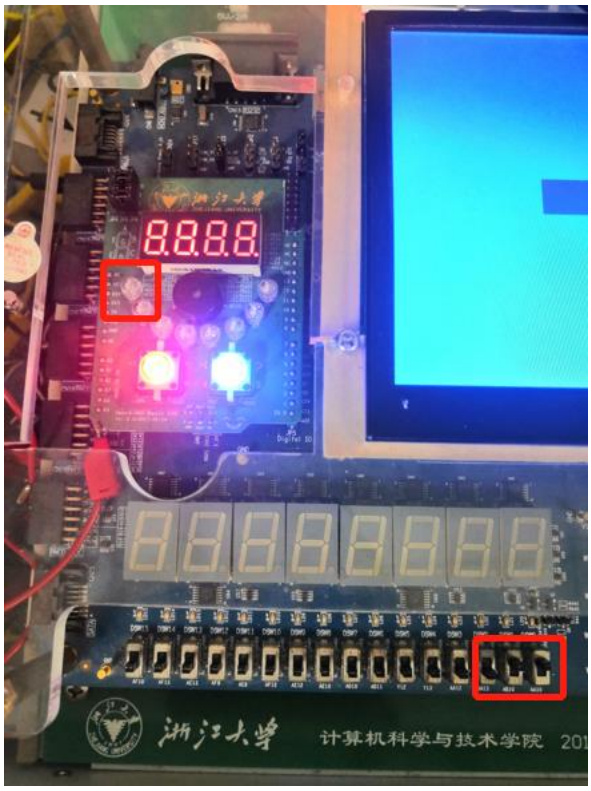
5.2 用 74LS138 译码器实现楼道灯控制

- 当只有一个开关打开时，楼道灯亮



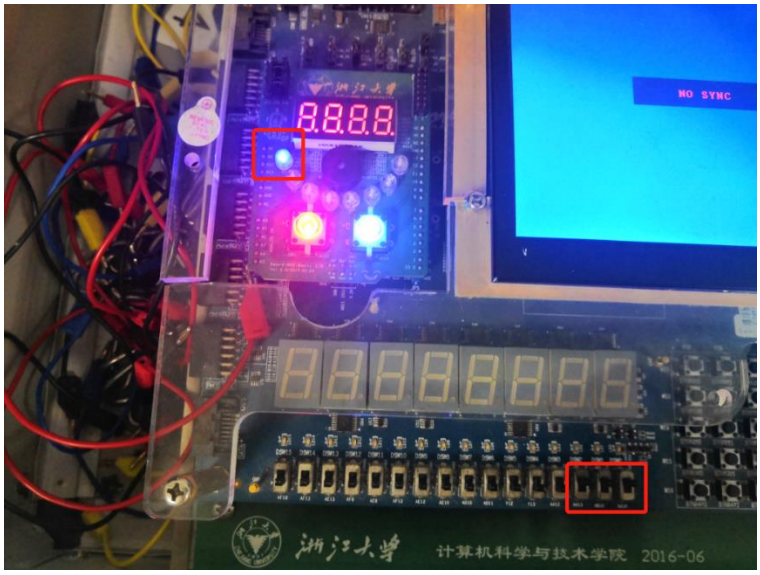
图表 16 验证楼道灯逻辑功能

- 当有两个开关打开时，楼道灯灭



图表 17 验证楼道灯逻辑功能

- 当三个开关全部打开的时候，楼道灯亮



图表 18 验证楼道灯逻辑功能

经过实验验证，成功利用 74LS138 变量译码器实现了楼道灯的功能，根据表达式 F = 001 + 010 + 100 +111，只有奇数个开关打开的时候，楼道灯会亮，而当偶数个开关打开的时候，楼道灯熄灭。

六、讨论、心得

这次实验是在第四次实验初步了解 ISE 基本操作后，第一次利用 ISE 建立工程并在 sword 板上实践验证的实验。通过这次实验，既复习了前次实验中练习的画图、模拟仿真、引脚约束等操作，又通过逻辑电路实现了变量译码器并以此实现了楼道灯的功能。虽然因为各种操作练习的还不够多，在开头画图、模拟仿真的时候出现了一些错误，但最终还是通过仿真的结果在连线图中找到并改正了错误，也更加熟悉了引脚约束和下载到 sword 板验证的操作。希望之后通过更多的练习，在接下来的实验里更加顺利的用逻辑电路实现功能。

实验六—7 段数码管显示译码器设计与应用

实验报告

姓名：程梦烨　　学号：3180103733　　专业：软件工程

课程名称：逻辑与计算机设计基础实验　　同组学生姓名：

实验时间：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

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

二、实验内容和原理

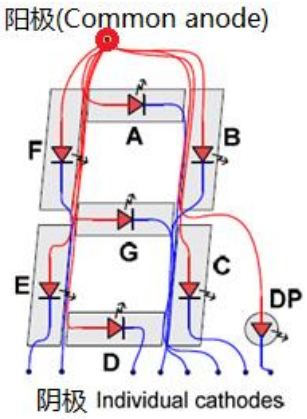
2.1 实验内容

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

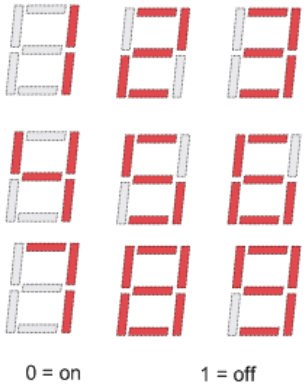
2.2.1 单位数码管

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



图表 1 单位数码管结构图

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



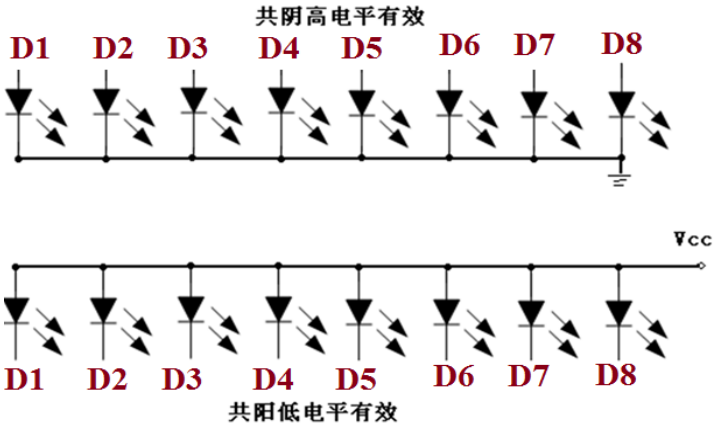
图表 2 单位数码管真值表

2.2.2 共阴（阳）控制

正负逻辑关系说明：

（负逻辑）共阳连接：8 个 LED 正极连在一起，负极低电平时

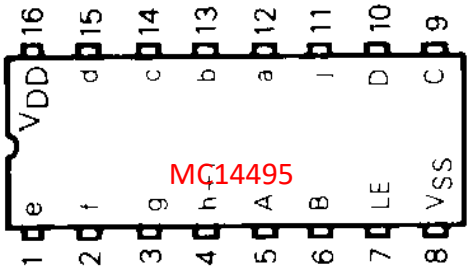
（正逻辑）共阴连接：8 个 LED 负极连在一起，正极高电平时点亮



图表 3 共阴（阳）控制

2.2.3 七段数码管 MC14495 结构

在本实验中，显示数字的七段数码管 MC14495 模块采用了共阳控制，LED 的正极连在一起，另一端作为点亮的控制，当负极=0 的时候，灯被点亮。



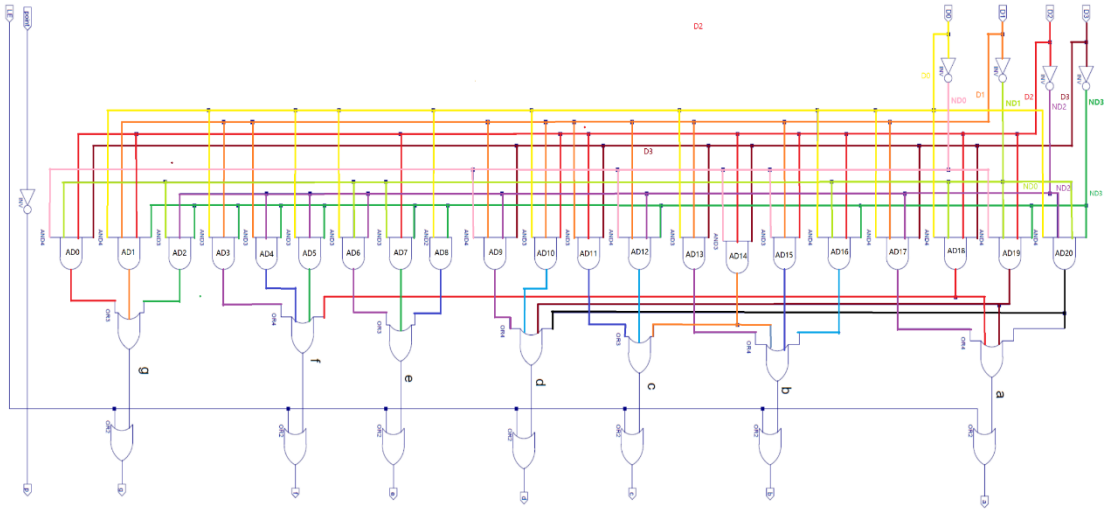
图表 4 MC14495 结构图

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

图表 5 MC14495 真值表

2.2.4 七段数码管连线图

通过卡诺图可以得到每段 LED 所对应的逻辑电路，整合到一起，可得到七段数码管 MC14495 的连线图。



图表 6 七段数码管 MC14495 连线图

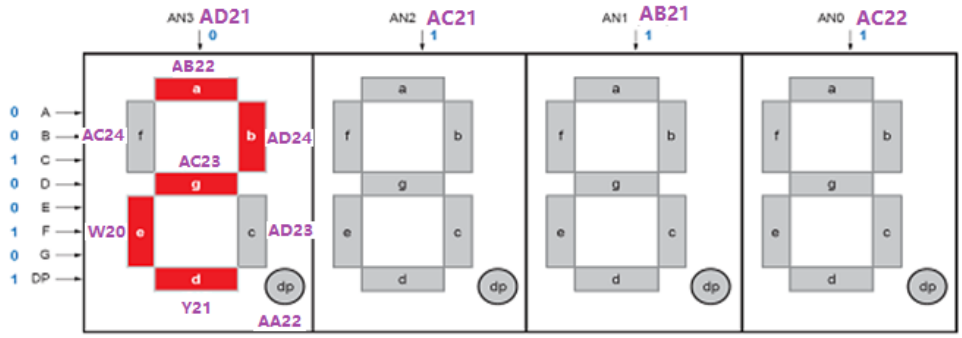
2.2.5 多位七段数码管显示原理

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：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

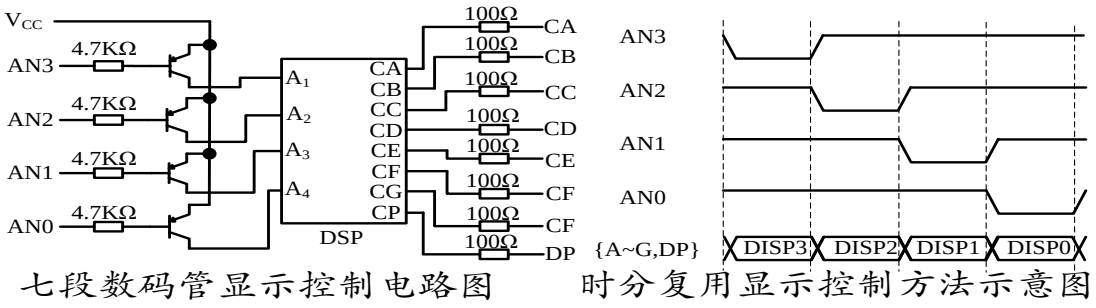
4 位七段码结构：正极为公共端，七段信号并联。



图表 7 4 位七段码结构图

2.2.6 分时控制示意

通过动态扫描，低电平与输入显示对应，分时送 a~g，p，用序列信号控制



图表 8 分时控制示意

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　1 套

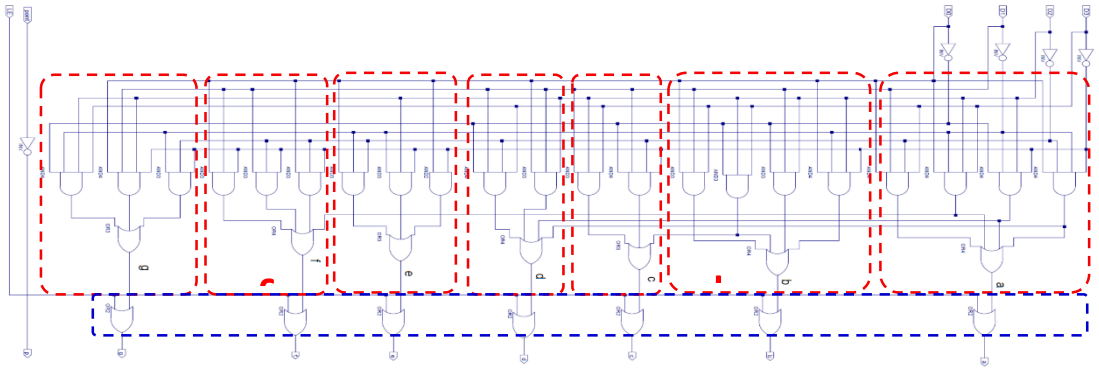
四、操作方法与实验步骤

4.1 设计实现显示译码 MyMC1449 模块

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 按照原理图设计 MyMC14495 模块



图表 9 MyMC14495 模块连线图

4.1.4 Check Design Rules，检查错误

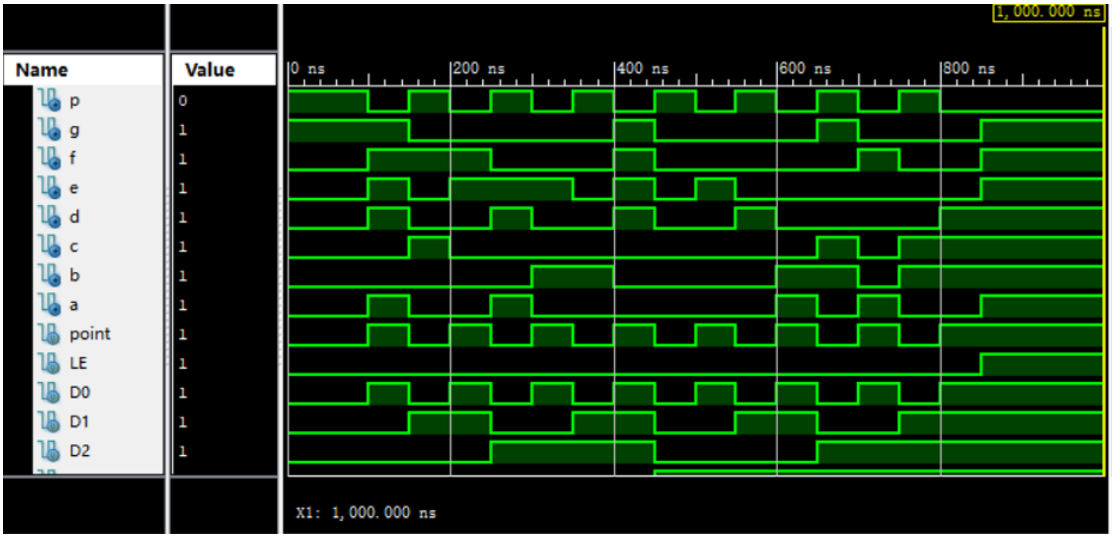
4.1.5 View HDL Functional Model，查看并学习 Verilog HDL 代码

4.1.6 模拟仿真

对 MyMC1449 模块进行仿真，激励代码如下（initial 主要代码）：

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

#50;
LE=1;
end
```

仿真结果如下：



图表 10 MyMC14495 模块的模拟仿真

在模拟仿真中，当 D3D2D1D0 为 0 时，只有 g 段为 1，即只有 g 段 LED 灯不亮，在数码管上显示为数字“0”；当 D3D2D1D0 为 1 时，只有 b、c 段为 0，即只有 b、c 段 LED 灯亮，在数码管上显示为数字“1”；当 D3D2D1D0 为 2 时，只有 c、f 段为 1，即只有 c、f 段 LED 灯不亮，在数码管上显示为数字“2”；并以此分别就检验其他数字，均和真值表相符，成功实现了 MyMC14495 模块的功能。

4.1.7 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，MyMC14495.sym

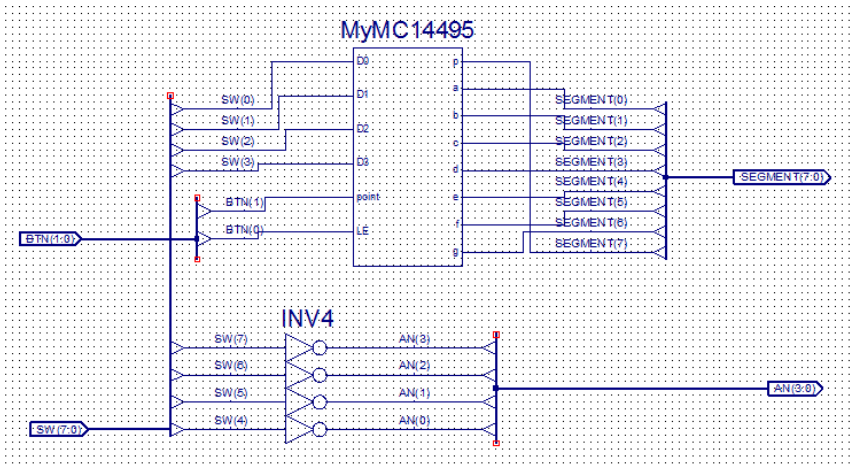
4.2 用 MyMC14495 模块实现数码管显示

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 按照原理图设计，并调用用 MyMC14495 模块



图表 11 数码管的显示连线图

4.2.5 下载验证

UCF 引脚定义

输入：

SW[7:4]=AN[3:0]

SW[3:0]=D3D2D1D0

SW[14]=LE

SW[15]=point

输出：a~g，p

引脚约束代码如下：

```
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 ;
NET "SEGMENT[3]"LOC = Y21 | IOSTANDARD = LVCMOS33 ;
NET "SEGMENT[4]"LOC = W20 | IOSTANDARD = LVCMOS33 ;
NET "SEGMENT[5]"LOC = AC24 | IOSTANDARD = LVCMOS33 ;
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 ;
```

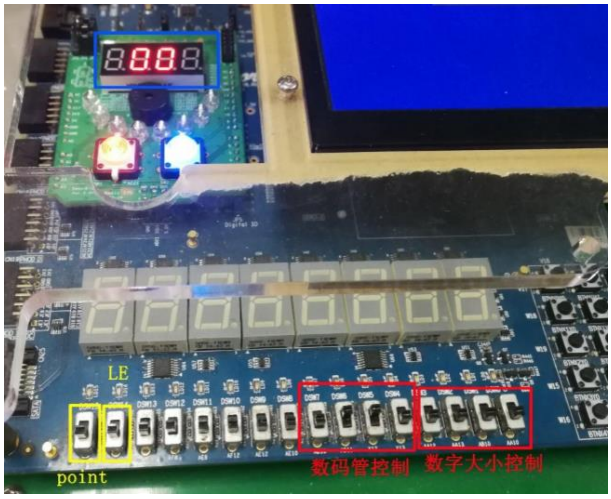
4.2.6 下载到板上验证

根据真值表，在 SWORD 开发板上验证是否实现了 7 段数码管的功能

五、实验结果与分析

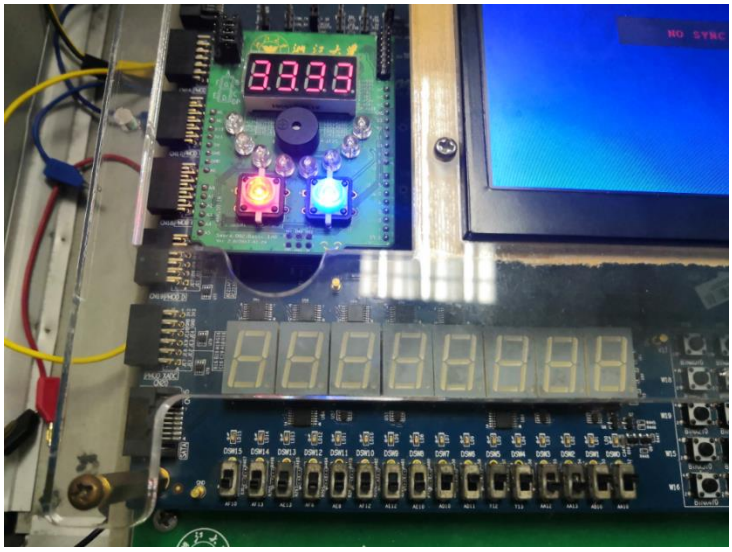
5.1 实验结果

5.1.1 通过数码管控制的 4 个开关可以控制显示数字的数码管的个数，例如下图只有中间两个数码管显示 0 的情况：



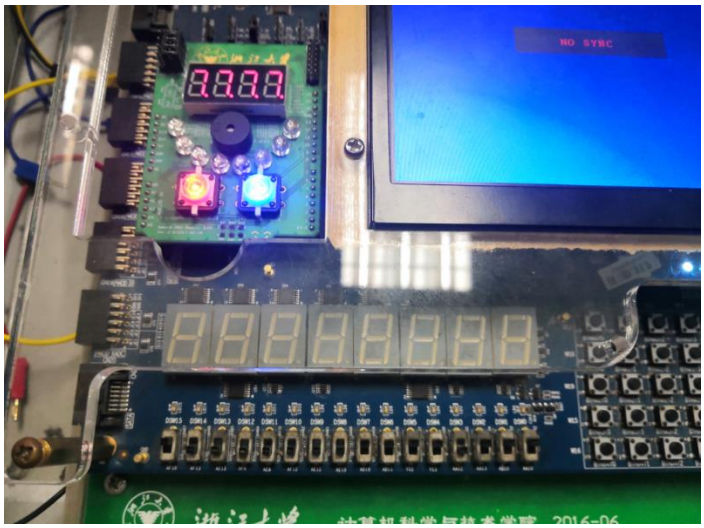
图表 12 实验结果图

5.1.2 通过数字大小控制的 4 个开关控制数码管显示不同的数，如下图所示：



图表 13 实验结果图

5.1.2 在数码管显示数字的同时通过控制 point 开关显示小数点，如下图所示：



图表 14 实验结果图

5.2 实验分析

经过实验结果的验证分析，成功实现了 7 段数码管显示译码器的功能，可以成功控制显示的数字、显示数字的位数和小数点的现实。但由于开发板上使用的是四位七段信号共联，所以四位只能显示相同的数字，如果要实现不同的数字，要使用动态扫描，将会在下一个实验中实现。

六、讨论、心得

这次实验因为有了上次实验的经验和基础，提前画图并做了仿真模拟等等，虽然 MyMC14495 模块的连线非常复杂，但在理清了逻辑关系后画图也不那么困难，十分顺利的通过了模拟仿真的验证。而在引脚约束的部分，由于课件上没有给出全部的代码，因此引脚约束的代码还是需要理解了输入输出后仿照前一次的写法来写。且这次实验因为提前预习做好了大部分工作，因此在正式上课时老师讲解完实验知识点后，就成功实现了下载到 sword 板上验证结果，顺利的完成了这次实验。因此之后的实验还是需要提前预习，并把能完成的部分提前完成。

实验七—多路选择器设计及应用实验报告

姓名：程梦烨　　学号：3180103733　　专业：软件工程

课程名称：逻辑与计算机设计基础实验　　同组学生姓名：

实验时间：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

1.1　掌握数据选择器的工作原理和逻辑功能
1.2　掌握数据选择器的使用方法
1.3　掌握 4 位数码管扫描显示方法
1.4　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 记分板设计

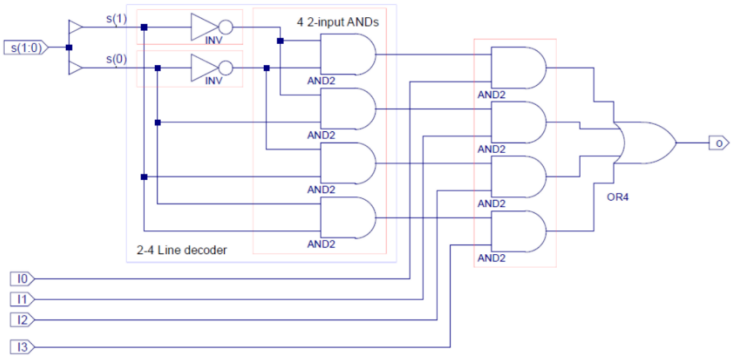
2.2 实验原理

2.2.1 四选一多路选择器：MUX4to1

根据事件简化真值表，输出是控制信号全部最小项与或结构

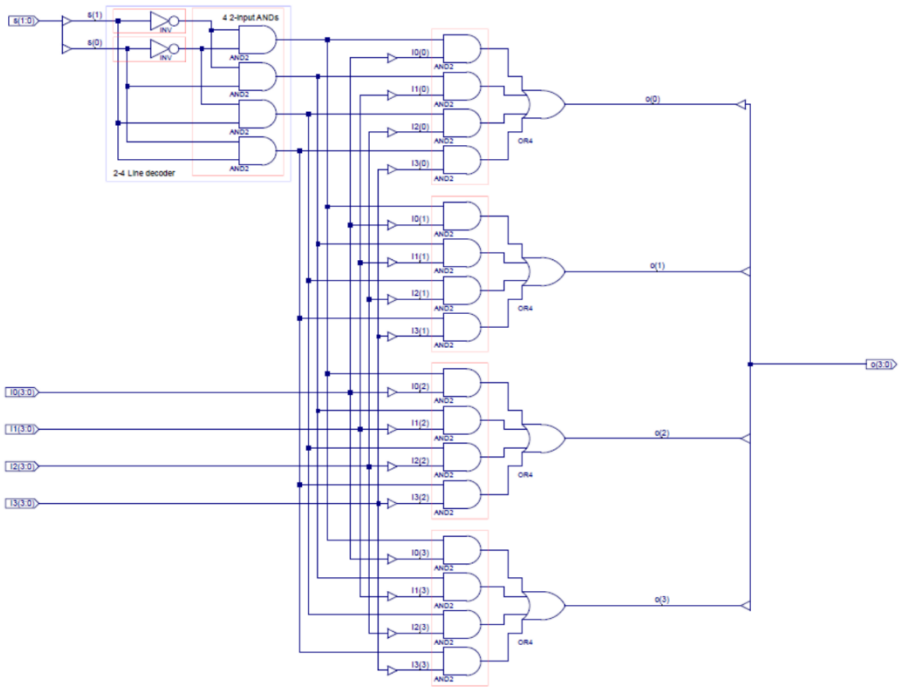
信息输入	控制端	选择输出	
I0 I1 I2 I3	S1 S0	o	输出项
I0 I1 I2 I3	0 0	I0	S1S0 I0
I0 I1 I2 I3	0 1	I1	S1S0 I1
I0 I1 I2 I3	1 0	I2	S1S0 I2
I0 I1 I2 I3	1 1	I3	S1S0 I3

图表 1 四选一多路选择器真值表



图表 2 四选一多路选择器连线图

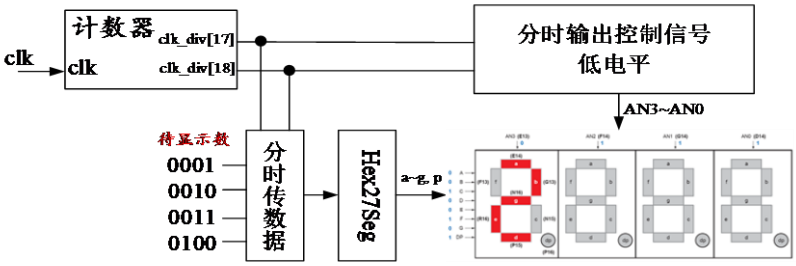
2.2.2 4 位四选一扩展：MUX4to1b4



图表 3 MUX4to1b4 模块连线图

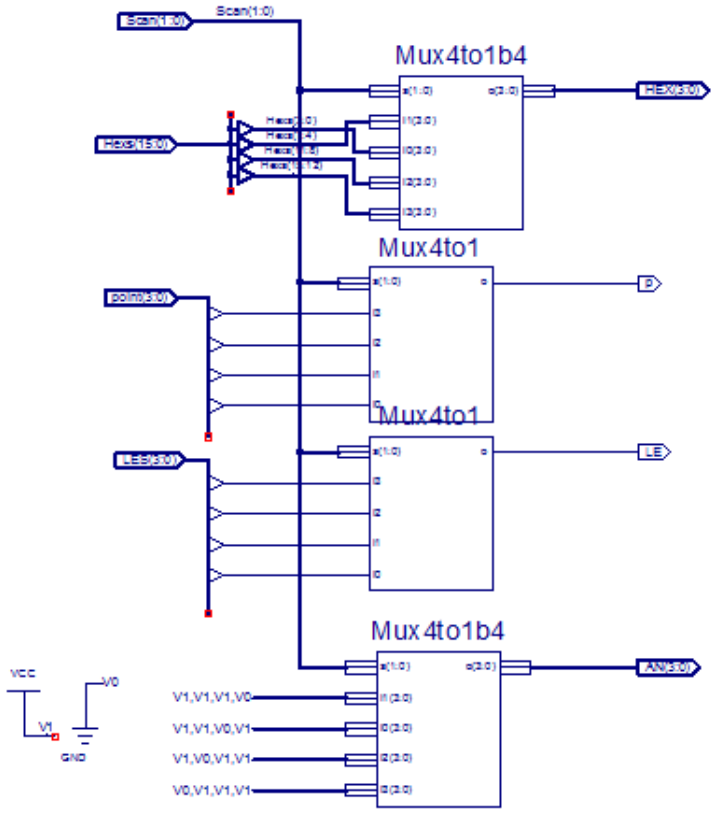
2.2.3 动态扫描显示

扫描信号来自计数器：将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。利用视觉延时，计数器的分频系数要适当，眼睛舒适，无法分辨出来即可。 使用 if_then 或 case 语句实现条件输出电路。



图表 4 动态扫描流程图

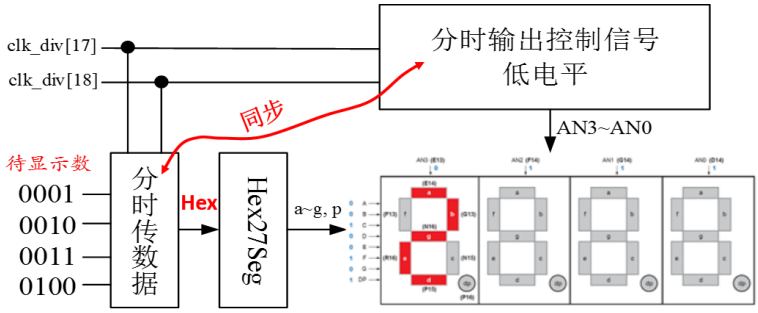
2.2.4 多路选择器应用：4 位七段显示扫描控制



图表 5 DisplaySync 模块原理图

2.2.5 辅助模块：时钟计数分频器

本实验使用的是 32 位时钟计数分频器，可输出 2-232 分频信号，可用于一般非同步类时钟信号，但延时较高，要求不高的时钟也可以用。而在多位七段显示器动态扫描的实验中可用。



图表 6 时钟计数分频器

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 1 套

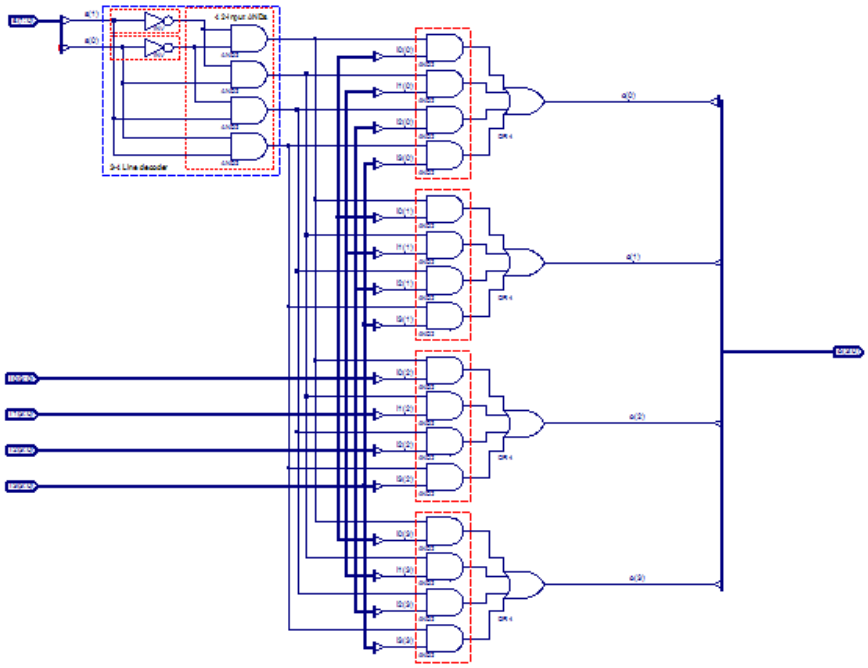
四、操作方法与实验步骤

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 Mux4to14b。

4.1.3 按照原理图设计 Mux4to14b 模块，连线图如下图所示



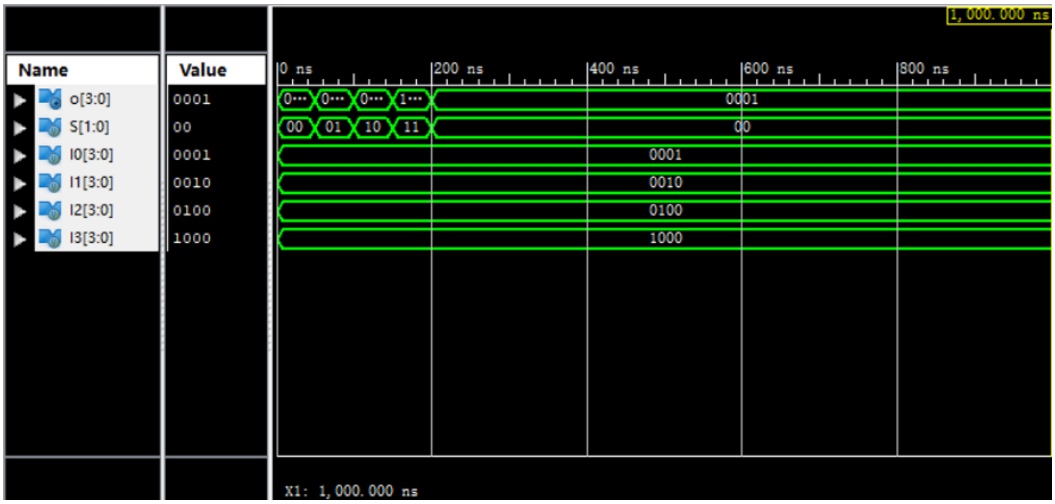
图表 7 Mux4to14b 模块连线图

4.1.1 模拟仿真

对 Mux4to14b 模块进行仿真，激励代码如下（initial 主要代码）：

```
initial begin
    S = 0;
    I0 = 0;
    I1 = 0;
    I2 = 0;
    I3 = 0;

    I0[0] = 1;
    I1[1] = 1;
    I2[2] = 1;
    I3[3] = 1;
    #50;

    S[0] = 1;  #50;
    S[0] = 0; S[1] = 1;  #50;
    S[0] = 1;  #50;
    S[0] = 0;  S[1] = 0;
end
```

仿真结果如下：



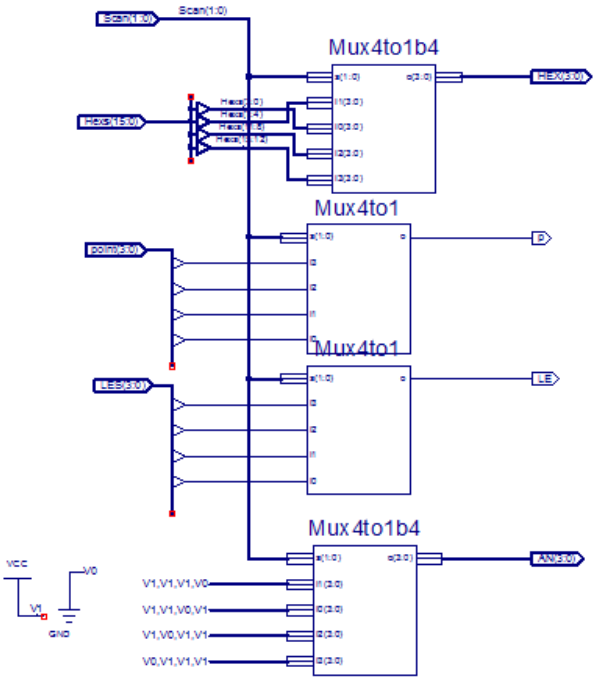
图表 8 Mux4to14b 模块的模拟仿真

在模拟仿真中，当输入 s 为 00 时，只有 O[0]为 1，当输入 s 为 01 时，只有 O[1]为 1，当输入 s 为 10 时，只有 O[2]为 1，当输入 s 为 11 时，只有 O[3]位 1， 符合预期结果，成功实现了 Mux4to14b 模块的功能。

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。

4.2.2 根据原理设计动态扫描同步输出模块，利用之前已经完成的 Mux4to14b 模块和 Mux4to1 来实现，具体连线图如下图所示：

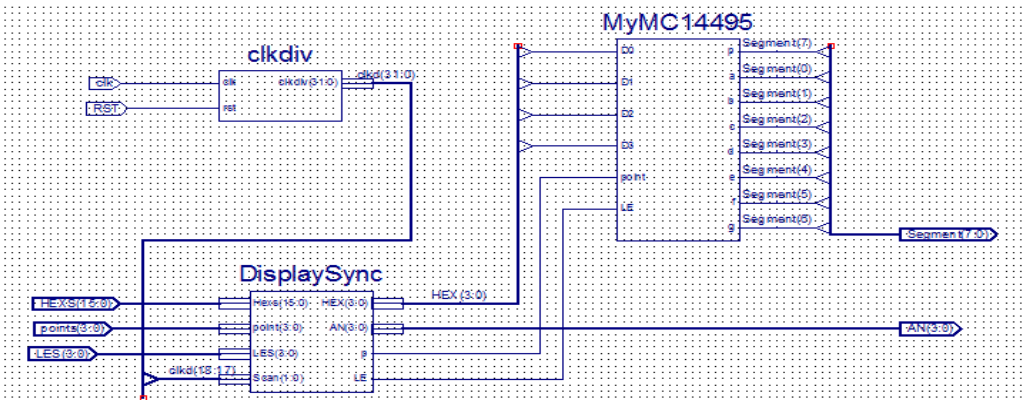


图表 9 动态扫描同步模块设计连线图

4.2.3 根据原理设计通用计数分频模块，代码如下所示：

```
module clkdiv(input clk,
    input rst,
    output reg[31:0]clkdiv
);

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

4.2.4 新建源文件，类型是 Schematic，文件名称用 disp_num，并根据原理图方式进行设计，具体连线图如下图所示：



图表 10 显示数字 disp_num 模块原理连线图

4.2.5 新建源文件 top，并右键设为“Top Module”，作为顶层文件，代码如下所示：

```
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,
    output Buzzer
);
    wire [15:0] num;
    CreatNumber c0(btn,num);
    disp_num d0(clk,num,SW[7:4],SW[3:0],1'b0,AN,Buzzer,SEGMENT);
endmodule
```

4.2.6 使用行为描述设计，即为 top 模块中的 CreateNumber 函数，四个按键各按一下，4 个 4 位 2 进制数分别加 1。根据原理代码如下所示：

```
module CreatNumber(
    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
```

4.2.7 下载验证

UCF 引脚定义：

输入：

时钟：clk

使能控制：sw[7:4]为 les[3:0]

小数点输入：sw[3:0]为 point[3:0]

按键输入数字：sw[15:12]为 btn[3:0]

输出：

a~g，p=segment

an[3:0]

引脚约束代码如下：

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

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;

NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
```

```
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;

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 = AF10 | IOSTANDARD = LVCMOS15;
NET "btn[1]" LOC = AF13 | IOSTANDARD = LVCMOS15;
NET "btn[2]" LOC = AE13 | IOSTANDARD = LVCMOS15;
NET "btn[3]" LOC = AF8 | IOSTANDARD = LVCMOS15;
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 "Buzzer" LOC = AF24 | IOSTANDARD = LVCMOS33;
```

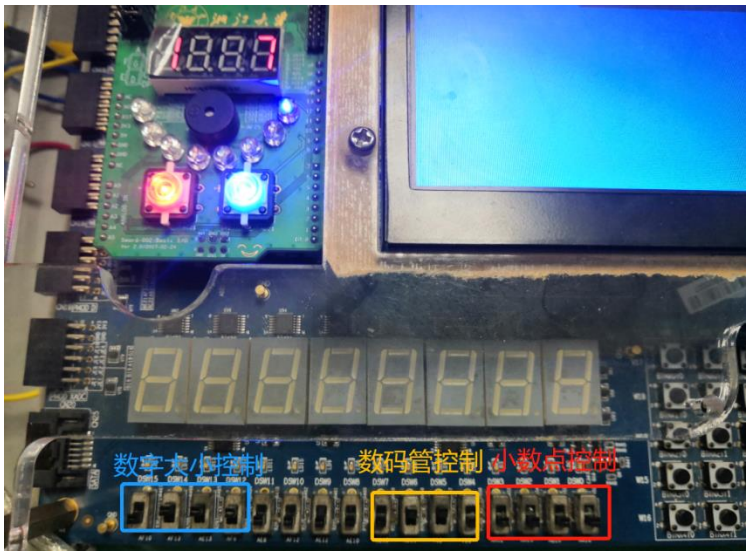
4.2.8 下载到板上验证

在 SWORD 开发板上验证是否实现了计数板的功能

五、实验结果与分析

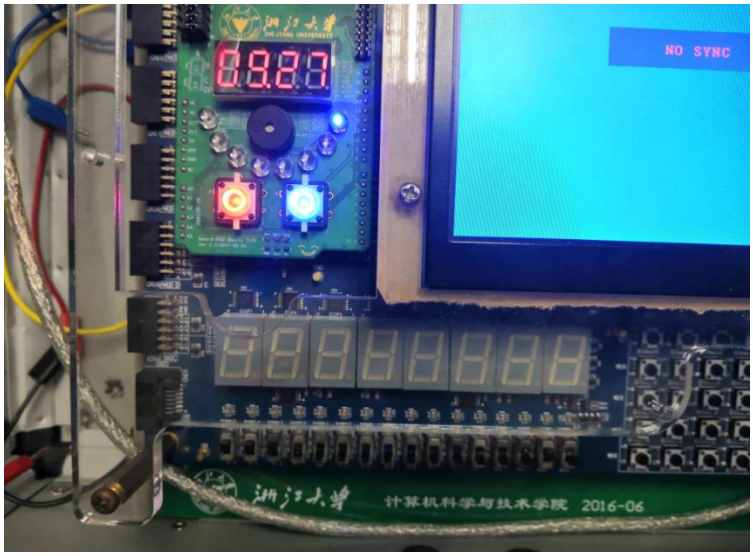
5.1 实验结果

5.1.1 通过开关，可以在四位七段数码管显示数字的基础上，每一位显示不同的数字，如下图所示：

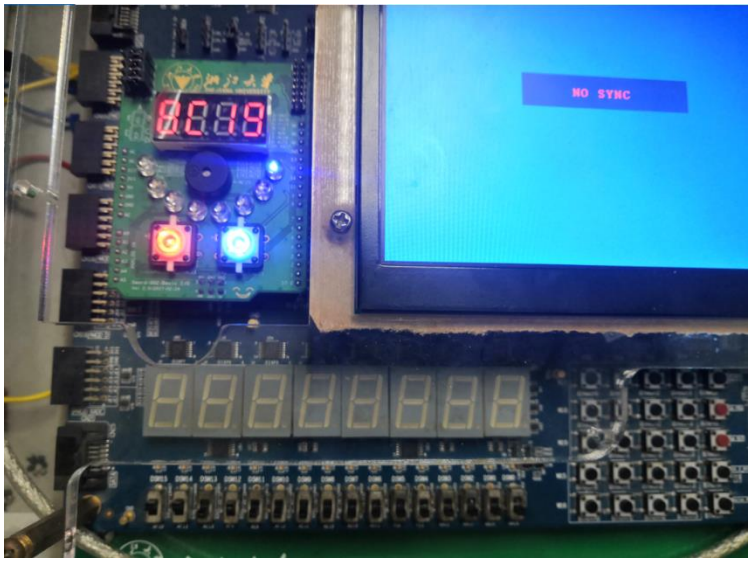


图表 11 实验结果测试图

5.2.2 通过开关的调整，能够显示任意数字和控制每一位小数点有无，如下图所示：



图表 12 实验结果测试图



图表 13 实验结果测试图

5.2 实验分析

经过实验结果的验证分析，成功实现了计数板的基本功能，即实现了用动态扫描来实现四位七段数码管显示不同的数字，并且能够利用开关使开放板上显示任意的数字。

六、讨论、心得

本次实验是在实验六的基础上，从四位数码管只能显示同一个数字，到通过动态扫描实现了显示不同的数字。和前两次实验相比，这次实验步骤更加复杂，模块的原理图设计也更多，在提前预习进行原理图设计的过程中，也是花了一定时间阅读课件才弄明白了整个设计流程，但经过前几次的练习，画图部分已经熟练了很多。本次实验中是第一次用 verilog 语

言而不是原理图设计实现顶层模块，也使我对 Verilog 语言有了进一步的理解，希望在以后实验的过程中加强对于 Verilog 语言的理解。