# 实验五--变量译码器设计与应用实验报告

姓名：黄炯睿 学号：3170103455 专业：信息安全

课程名称：逻辑与计算机设计基础实验 同组学生姓名：无

实验时间：2018-10-18 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 译码器定义

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）

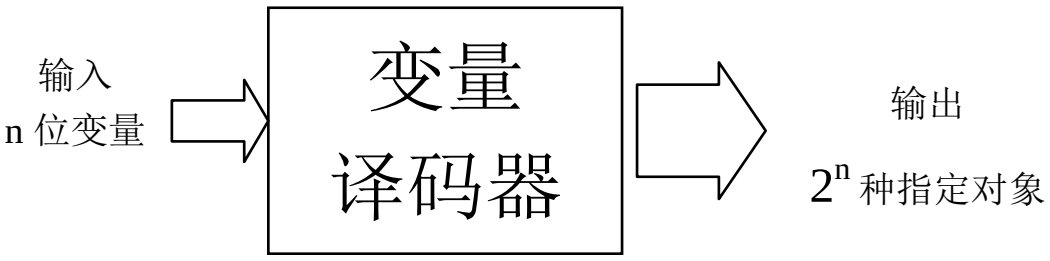
译码可分为：变量译码、显示译码

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 用 74LS138 译码器实现楼道灯控制

变量译码器是一个将 n 个输入变为 2n 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。



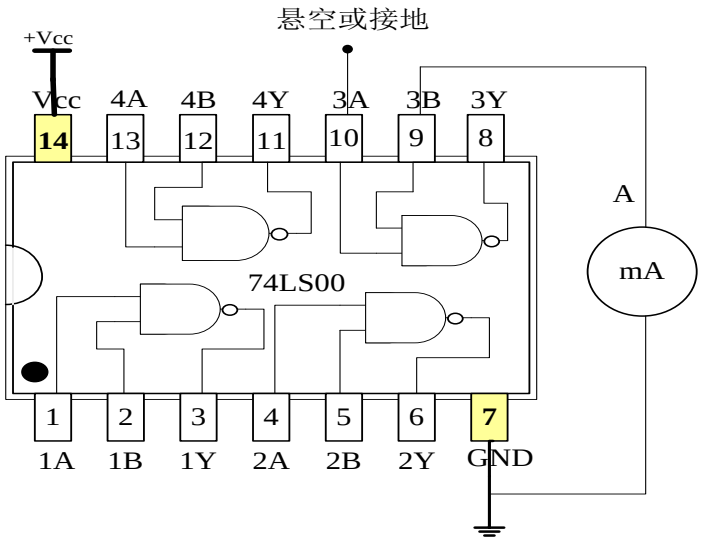

图表 1 变量译码器功能图

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，74LS138 变量译码器功能表以及引脚如下图所示。

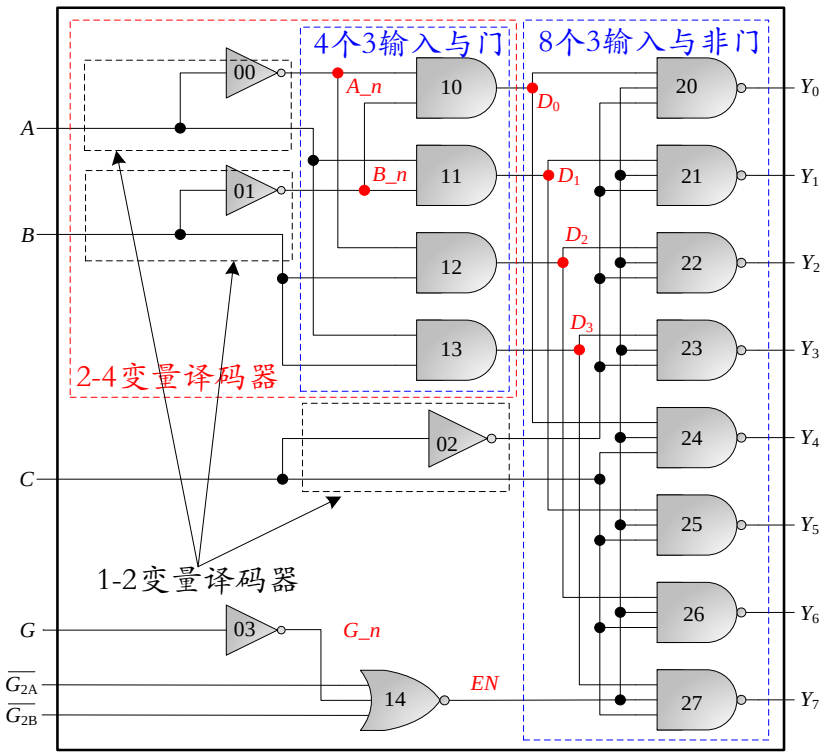
| 输入 | | 译码器输出（低电平有效） | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 使能 | 变量 | | | | | | | | |
| $GG_{2A}G_{2B}$ | CBA | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ | $Y_4$ | $Y_5$ | $Y_6$ | $Y_7$ |
| ×11 | ××× | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0×× | ××× | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 100 | 000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

图表 2 74LS138 变量译码器功能表




图表 3 74LS138 变量译码器引脚

2.2.4 74LS138 连线明细图，如下图所示：




图表 4 74LS138 连线明细图

2.2.5 用 74LS138 实现楼道灯功能

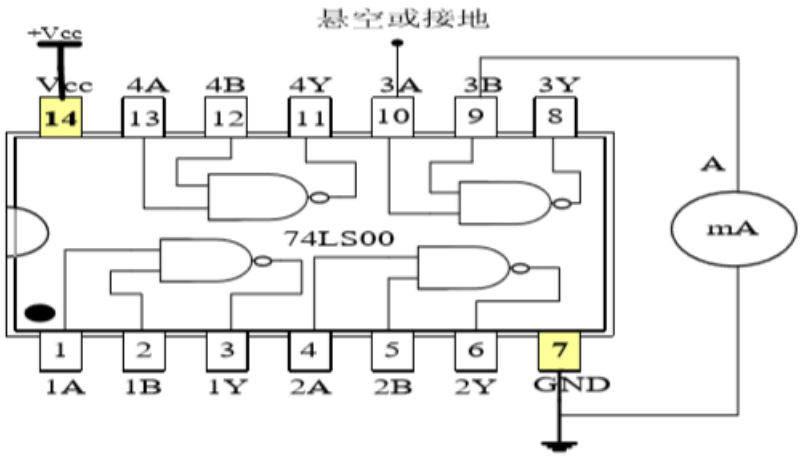
变量译码器的输出对应所有输入变量的最小项组合，如果将函数转换成最小项 和的形式，则可以用变量译码器实现函数的组合电路。 要实现楼道灯的功能，应实现如下的真值表以及逻辑表达式：

$$F = S_3\overline{S_2}\overline{S_1} + S_3\overline{S_2}S_1 + S_3S_2\overline{S_1} + S_3S_2S_1$$

| $S_3$ | $S_2$ | $S_1$ | $F$ |
|---|---|---|---|
| 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 楼道灯功能真值表及逻辑表达式

转换后得到 F =001+010+100+111，因此利用 74LS138 实现楼道灯功能的连线明细图：




图表 6 用 74LS138 实现楼道灯功能连线明细图

# 三、主要仪器设备

1. SWORD 开发板 1 套
2. 装有 Xilinx ISE 14.7 的计算机 1 台

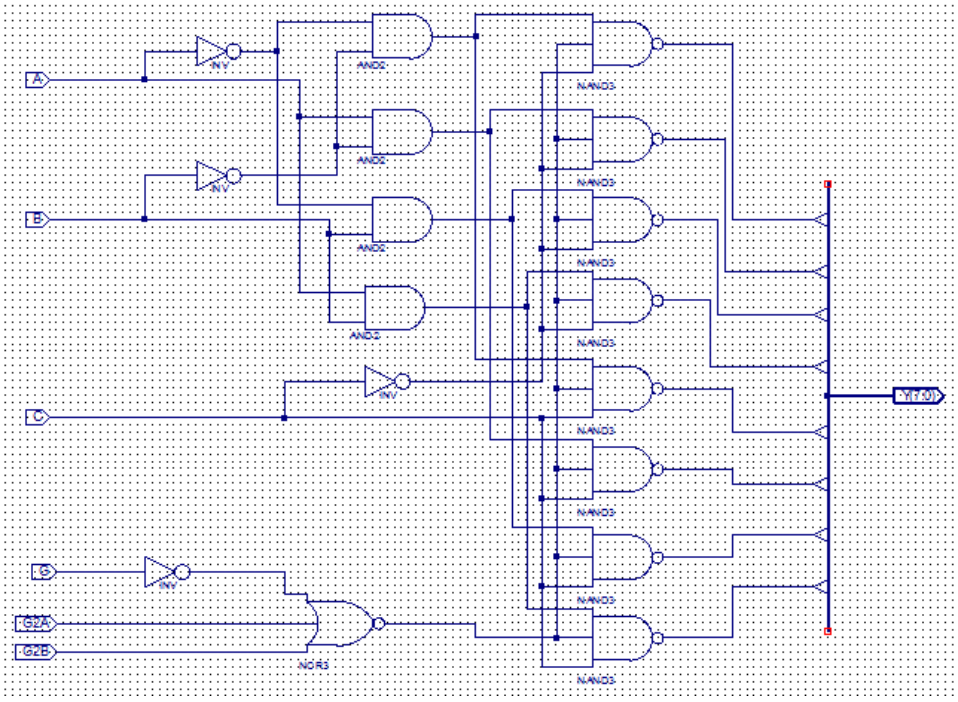
# 四、操作方法与实验步骤

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。

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，连线图如下图所示：




图表 7 D_74LS138 在 ISE 中的连线图

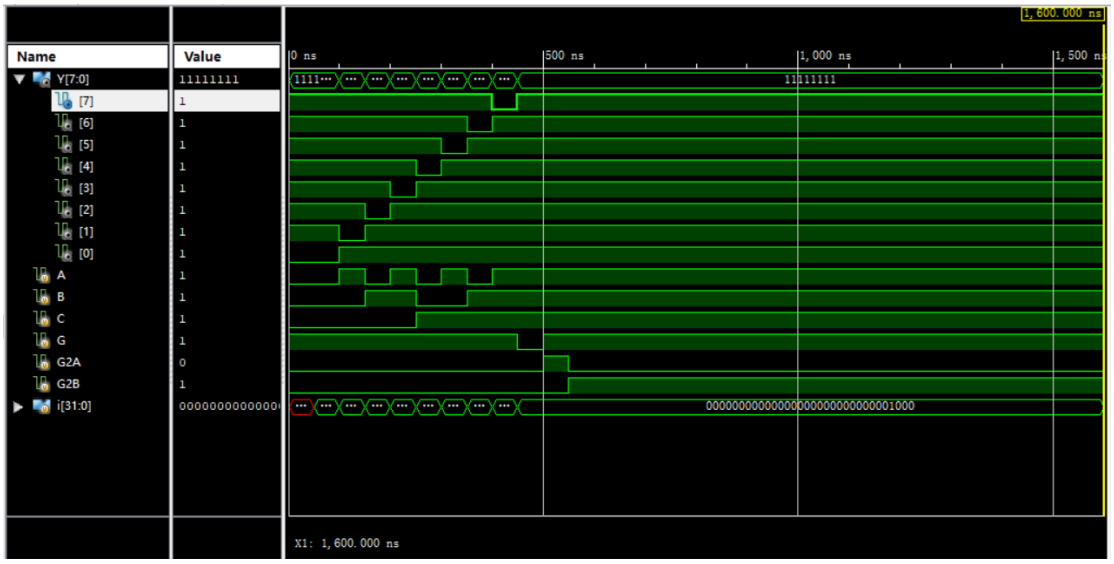
4.1.4 Check Design Rules，检查错误。

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 8 D_74LS138 模块的模拟仿真

模拟仿真汇总，随着输入的变化，8 种输出一次出现，仿真成功。

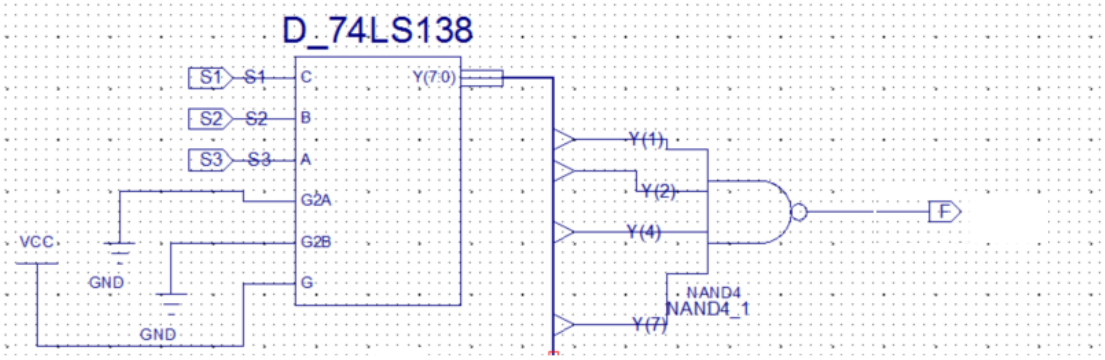
4.1.7 Create Schematic Symbol，系统生成 D_74LS138 模块的逻辑符号图文件，D_74LS138.sym

4.2 用 74LS138 译码器实现楼道灯控制

4.2.1 新建工程 LampCtrl138.

4.2.2 复制 D_74LS138.sym 和.vf 文件到工程目录。

4.2.3 按照原理图设计楼道灯控制逻辑电路，连线图如下图所示：



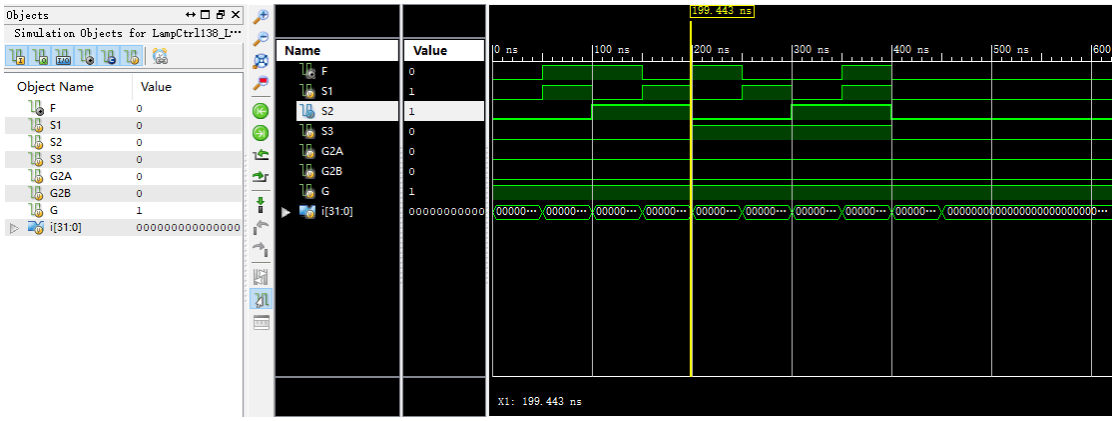

图表 9 楼道灯控制逻辑电路连线图

4.2.4 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 10 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。

4.2.5 下载验证 UCF 引脚定义：

输入用 3 个开关：S1,S2,S3

输出用 1 个 LED 灯

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;
```

4.2.6 在 SWORD 开发板上验证是否实现了楼道灯控制的功能

# 五、实验结果与分析

5.1 实验结果

5.2 实验分析

经过实验验证，成功利用 74LS139 变量译码器实现了楼道灯的功能，主要还是根据表达 式 F = 001 + 010 + 100 + 111，来实现，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

由于是初次使用，对于整套器材和流程都不是很清楚，我在完成实验的过程中遇到了很多的问题。

首先是在如何导入其他文件夹中的模块，开始我只是在 design 工作栏里 add the copy of source 然后发现在 symbol 里无法找到所需要的 symbol 原件。后来在同学的帮助下才知道也要在外部将.sym 和.vf 文件加入 project 所在文件夹。

还有一个小意外发生在最后的物理验证时，我手快，没有发现当时导入的.bit 文件并不是我刚刚完成的实验 5 的.bit 文件，导致物理验证的错误。

这一次的实验我花费了大量时间在仿真，引脚约束上，这可能主要还是因为对于软件不太熟悉，但经过这次的实验，对于画图以及模拟仿真、引脚约束这些常规的操作变得熟练了许多。

# 实验六--七段数码管显示译码器设计与应用

# 实验报告

姓名：黄炯睿 学号：3170103455 专业：信息安全

课程名称：逻辑与计算机设计基础实验 同组学生姓名：无

实验时间：2018-10-25 实验地点：紫金港东 4-509 指导老师：洪奇军

# 二、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

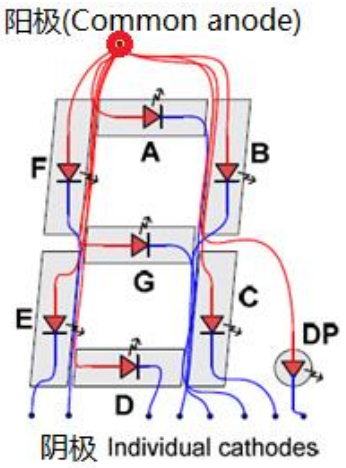
## 2.1 实验内容

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

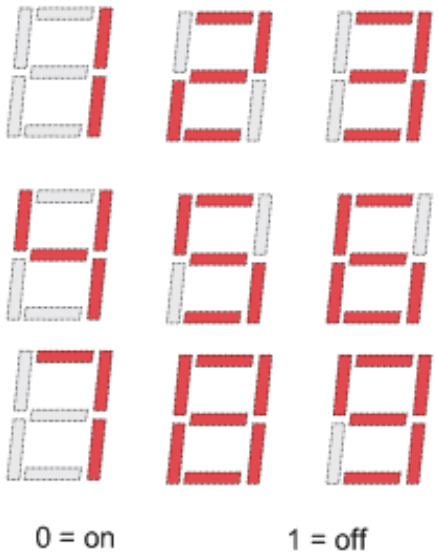
2.2.1 单位数码管

由 7+1 个 LED 构成的数字显示器件，如下图图表 1 所示，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 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。因此 MC14495 的结构以及该模块的真值表如下图所示。

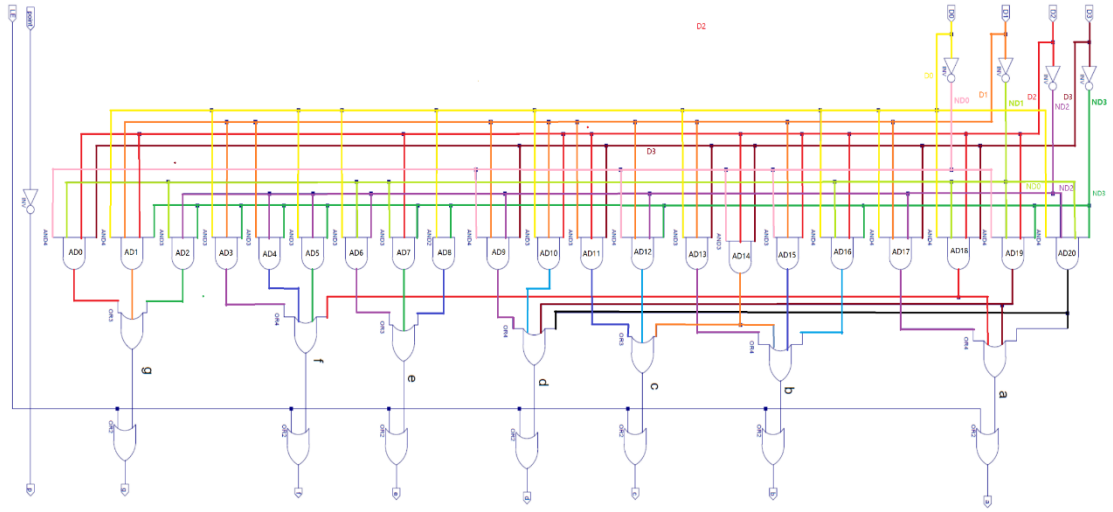

图表 3 MC14495 结构图

| Hex | $D_3D_2D_1D_0$ | BI/LE | a | b | c | d | e | f | g | p |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | p |
| 1 | 0001 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | p |
| 2 | 0010 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | p |
| 3 | 0011 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | p |
| 4 | 0100 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | p |
| 5 | 0101 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | p |
| 6 | 0110 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | p |
| 7 | 0111 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | p |
| 8 | 1000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P |
| 9 | 1001 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | P |
| A | 1010 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | P |
| B | 1011 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | P |
| C | 1100 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | P |
| D | 1101 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | P |
| E | 1110 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | P |
| F | 1111 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | P |
| X | xxxx | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

图表 4 MC14495 真值表

2.2.3 七段数码管连线图

根据以上的分析，再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，七段数码管 MC14495 的连线图如下图所示：



图表 5 七段数码管 MC14495 连线图

2.2.4 多位七段数码管显示原理

多位数码管的显示原理分为两种：静态显示和动态显示。

静态显示是每个 7 段码对应一个显示译码电路。动态扫描显示是时分复用显示，利用人眼视觉残留，将一个 7 段码译码电路分时为每个 7 段码提供译码。而本实验上 的开发板上的数码管只有一个显示译码电路，并且不采用动态扫描，所以最后 4 位数字都只能够显示同一个数字。

# 三、主要仪器设备

3. SWORD 开发板 1 套
4. 装有 Xilinx ISE 14.7 的计算机 1 台

# 四、操作方法与实验步骤

4.1 设计实现显示译码 MyMC14495 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。

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

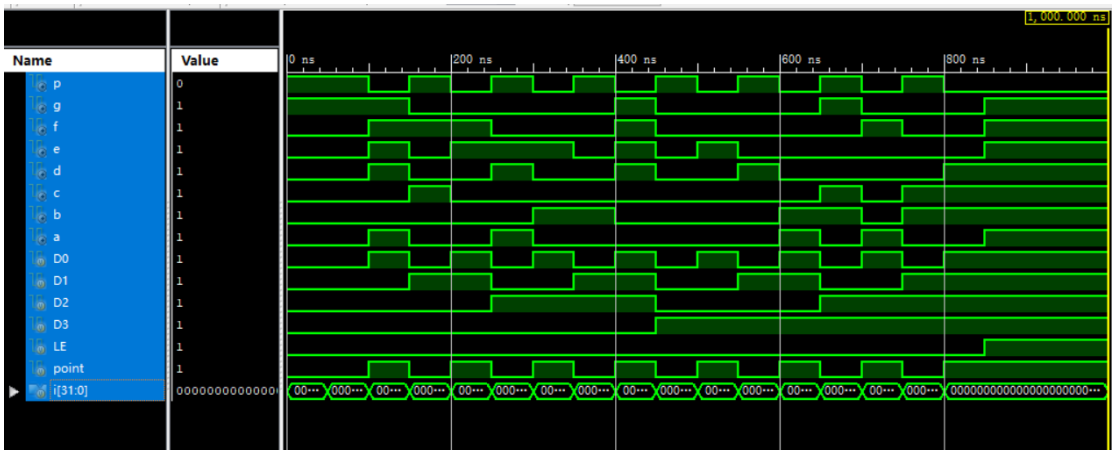
4.1.6 模拟仿真

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 6 MyMC14495 模块的模拟仿真

在仿真过程中，，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”；以此类推，可以验证该模块成功实现了预期的功能。

4.1.7 Create Schematic Symbol，系统生成 MyMC14495 模块的逻辑符号图文件，MyMC14495.sym

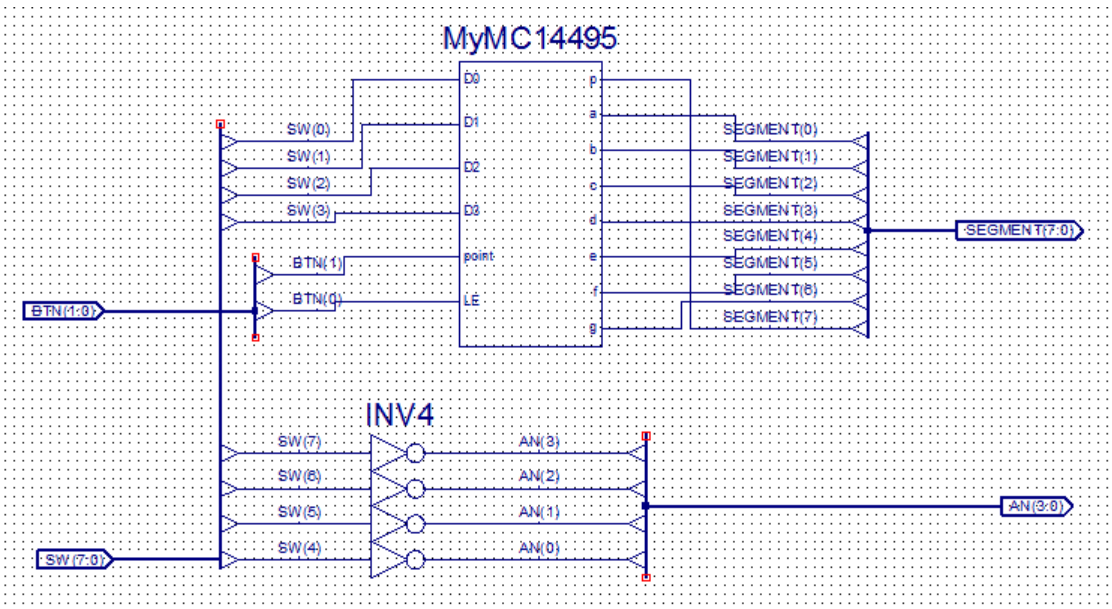
4.2 用 MyMC14495 模块实现数码管显示

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，利用 MyMC14495 模块来实现数码管的功能，连线图如下图所示：




图表 7 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;
NET "BTN[1]" LOC = AF10 | IOSTANDARD = LVCMOS15;

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;
```

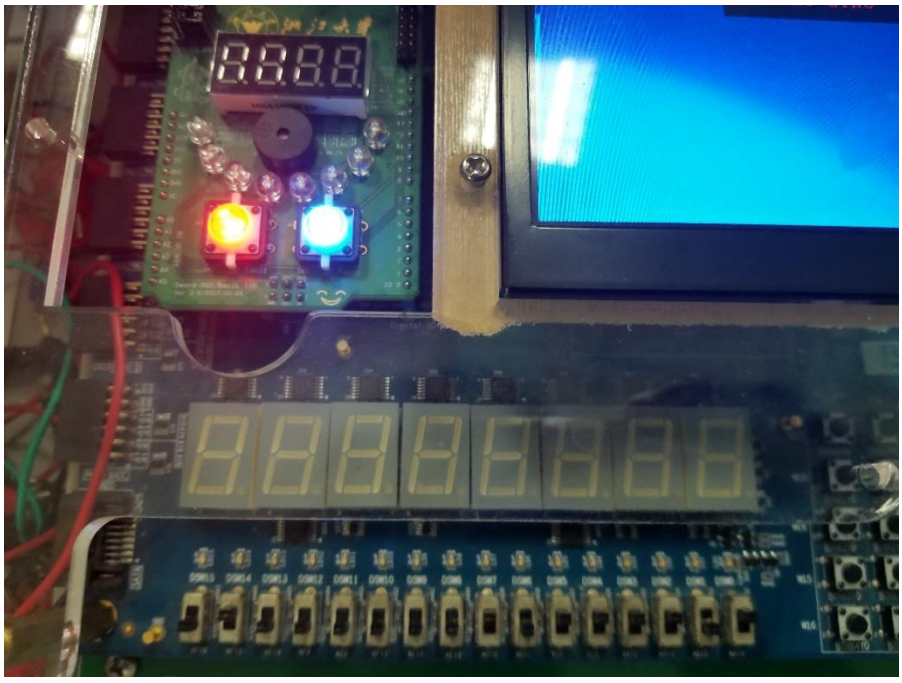
```
  NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;

NET "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 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

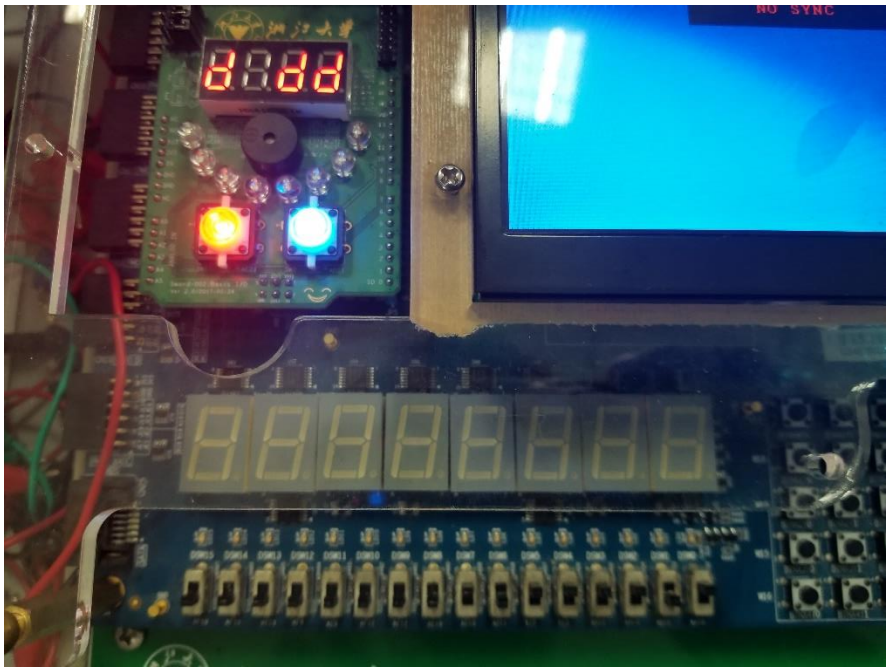
5.1.1



图表 8 实验结果图 1

最右边四个开关为 1101，且右数 5-8 个开关为 1011，小数点开关（左数第一个）为 0，但屏幕上不显示数字。因为使能开关（右数第 9 个）为 0，此时不会有任何输出，满足逻辑条件。

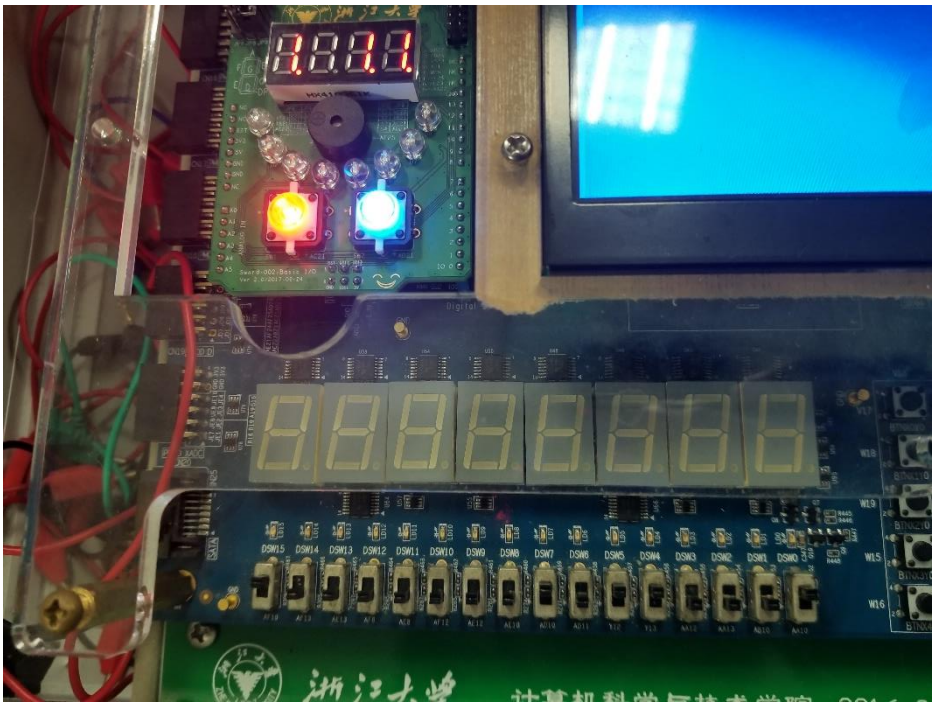
5.1.2



图表 9 实验结果图 2

最右边四个开关为 1101，且右数 5-8 个开关为 1011，使能开关（右数第 9 个）为 1，小数点开关为 0（左数第一个），此时屏幕上显示 1101 的十六进制数，即 d，满足逻辑条件。

5.1.3



图表 10 实验结果图 3

最右边四个开关为 0001，且右数 5-8 个开关为 1011，使能开关（右数第 9 个）为 1，小数点开关为 1（左数第一个），此时屏幕上显示 0001 的十六进制数和小数点，即 1. ，满足逻辑条件。

5.2 实验分析

经过实验结果的分析，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。七段数码管的 AN 控制信号可以决定该个七段数码管是否显示，从而产生只有两个或三个七段数码管显示数字的效果。 在此次实验七段数码管显示数字的显示方式中，仍采用的是静态显示。四个七段数码管只能显示相同的数字，在下次实验中我们将使用动态显示来显示不同的数字。

# 六、讨论、心得

此次实验难度有所提高，主要在于原理图的复杂和引脚代码的学习。这次实验我大量的时间花在了引脚约束上，由于 ppt 上并没有给出全部的代码，所以引脚约束的代码还是先理解了一会儿引脚约束的写法，最后在理解的基础上再开始写。

经过这次实验，我对于硬件语言开始有了一定的了解，走出了不再照葫芦画瓢的第一步。

# 实验七--多路选择器设计及应用实验报告

姓名：黄炯睿　　学号：3170103455　　专业：信息安全

课程名称：逻辑与计算机设计基础实验　　同组学生姓名：　　无

实验时间：2018-11-3　　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

1.11 掌握变量译码器的的逻辑构成和逻辑功能。
1.12 掌握数据选择器的使用方法
1.13 掌握 4 位数码管扫描显示方法
1.14　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

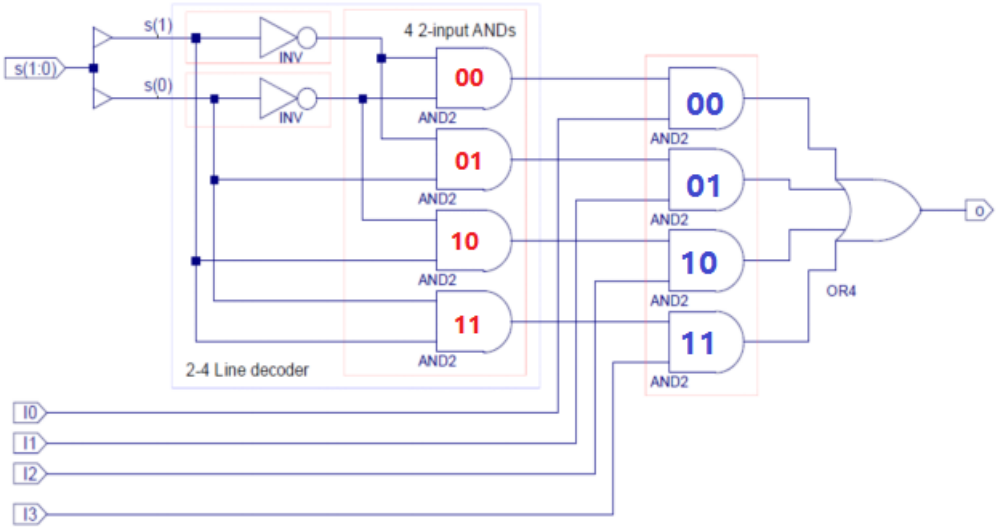
## 2.2 实验原理

2.2.1 四选一多路选择器：MUX4to1

根据事件简化真值表，输出是控制信号全部最小项与或结构，真值表以及连线图 如下图所示：

| 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 |

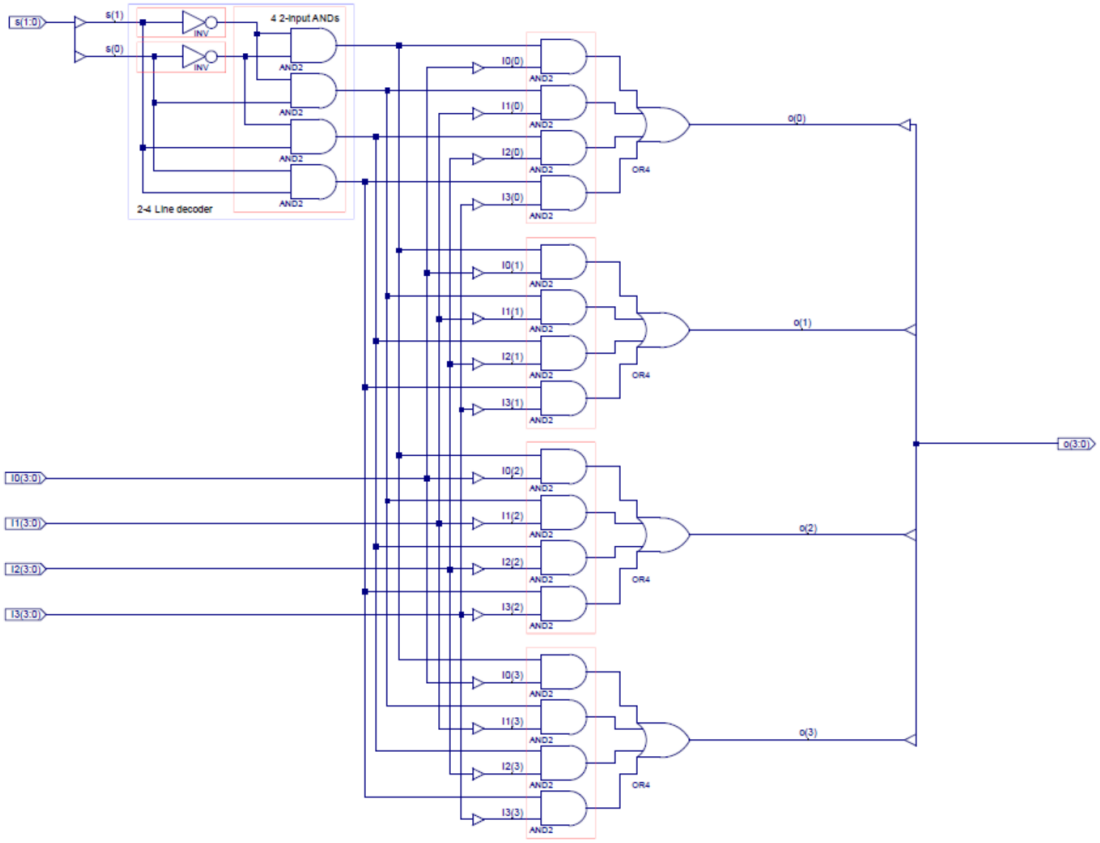
图表 1 四选一多路选择器真值表




图表 2 四选一多路选择器连线图

2.2.2 MUX4to1b4 模块

MUX4to1b4 模块为四位四选一多路选择器，是一种拓展，以下为该模块的连线图：

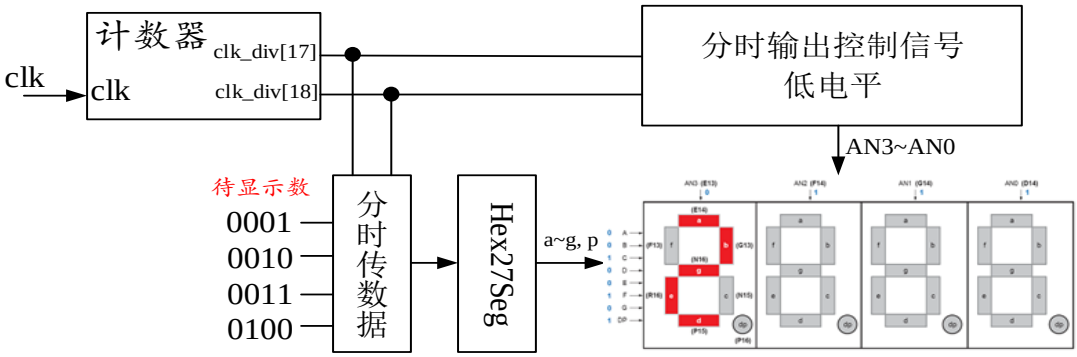



图表 3 MUX4to1b4 模块连线图

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。流程图如下图所示：




图表 4 动态扫描流程图

2.2.4 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以用。

# 三、主要仪器设备

5. SWORD 开发板 1 套
6. 装有 Xilinx ISE14.7 的计算机 1 台

# 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。

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

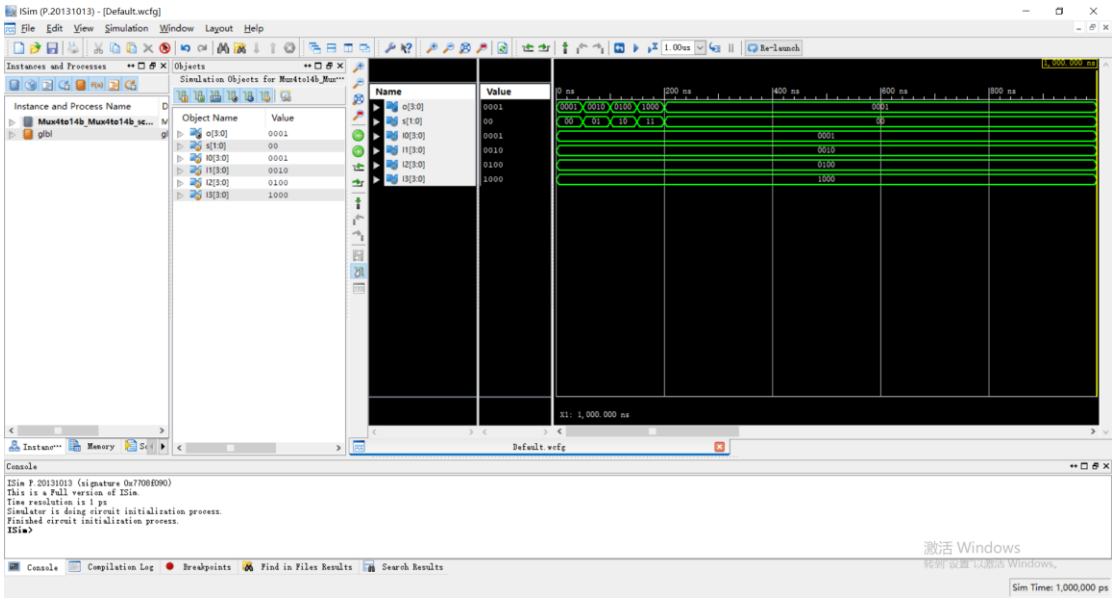
4.1.6 模拟仿真

对 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 5 Mux4to14b 模块的模拟仿真

在仿真过程中，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 模块的功能在数码管上显示为数字“1”；以此类推，可以验证该模块成功实现了预期的功能。

4.1.7 Create Schematic Symbol，系统生成 Mux4to14b 模块的逻辑符号图文件，Mux4to14b.sym

## 4.2 记分板应用设计

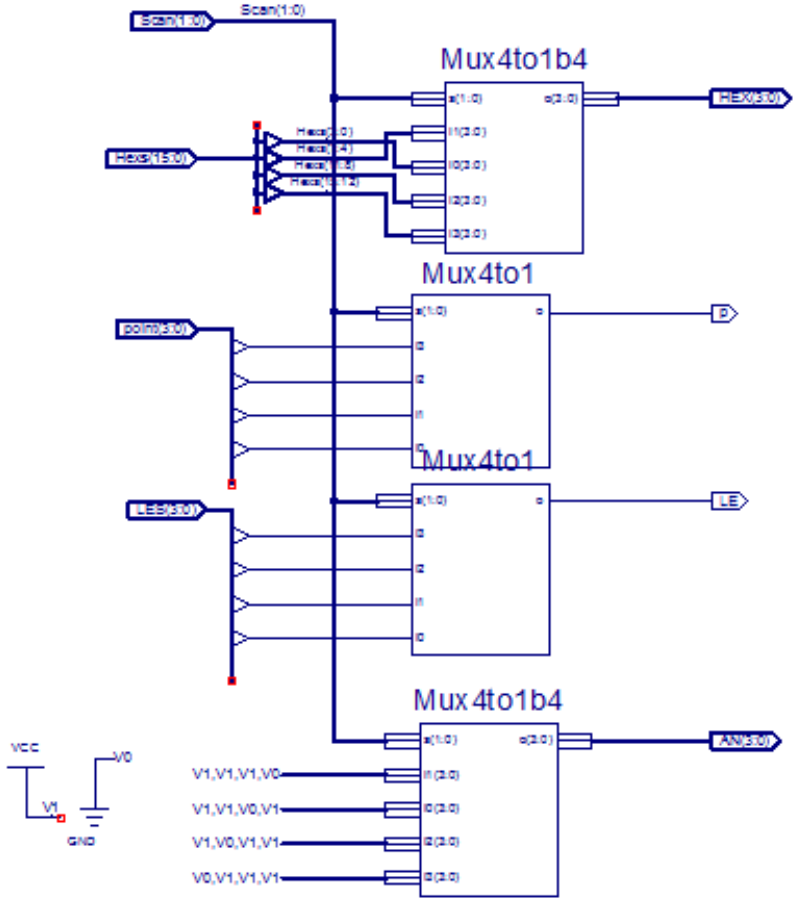
4.2.1 新建工程，工程名称用 ScoreBoard，Top Level Source Type 用 HDL。

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
```

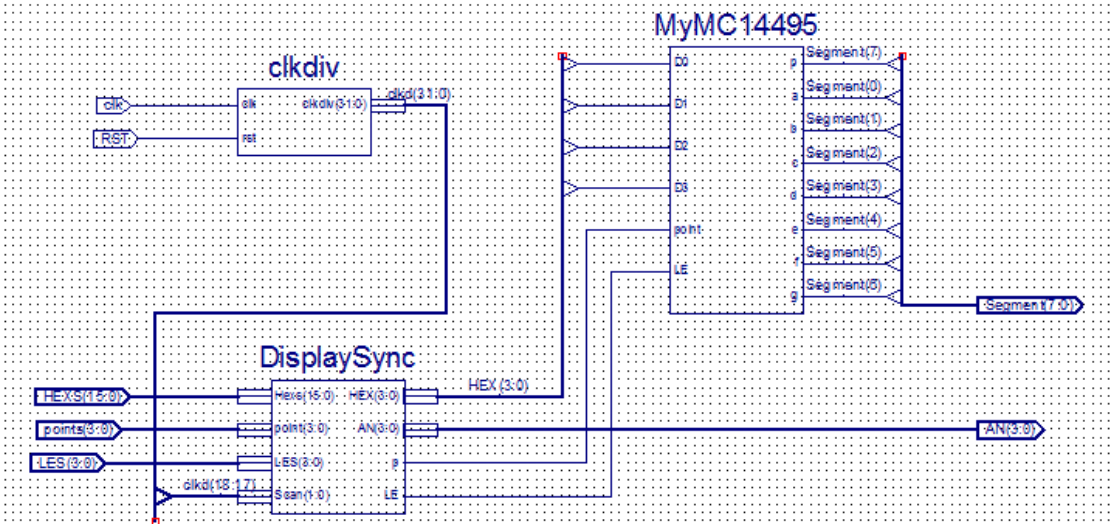
4.2.3 根据原理设计动态扫描同步输出模块，利用之前已经完成的 Mux4to14b 模块和

Mux4to1 来实现其功能，具体连线图如下图所示




图表 7 动态扫描同步模块设计连线图

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 8 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,
);
    wire [15:0] num;
    CreatNumber c0(btn,num);
    disp_num d0(clk,num,SW[7:4],SW[3:0],1'b0,AN,SEGMENT);
endmodule
```

4.2.6 使用行为描述设计，即为 top 模块中的 CreateNumber 函数，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 "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;
```

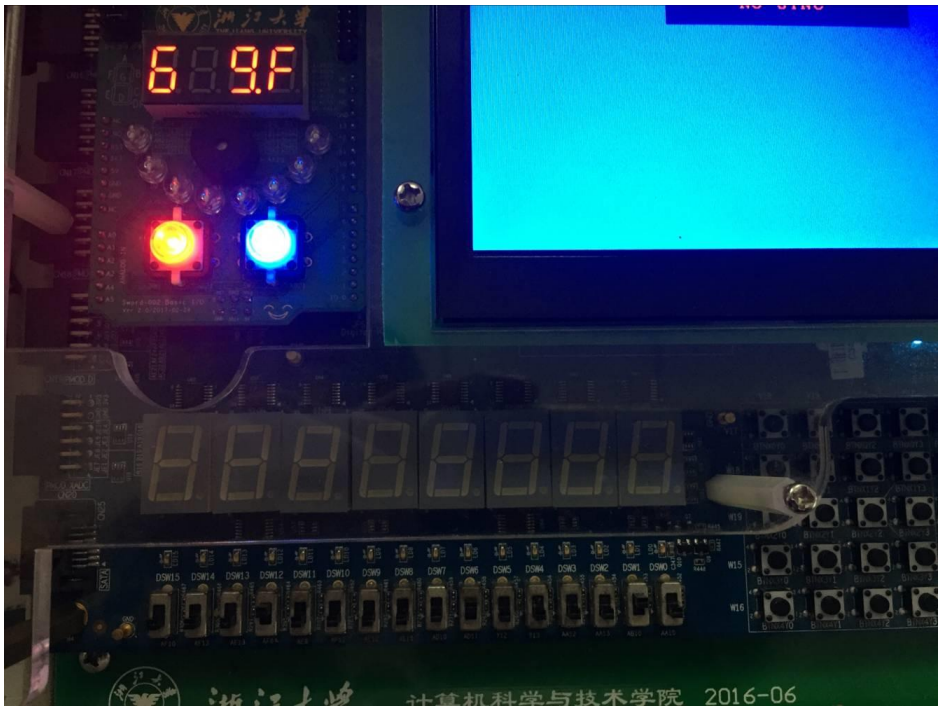
```
NET "SW[4]" loc=Y13 | IOSTANDARD=LVCMOS15;
NET "SW[5]" loc=Y12 | IOSTANDARD=LVCMOS15;
NET "SW[6]" loc=AD11 | IOSTANDARD=LVCMOS15;
NET "SW[7]" loc=AD10 | IOSTANDARD=LVCMOS15;

NET "BTN[0]" LOC=AF10 | IOSTANDARD=LVCMOS15;#SW[14]
NET "BTN[1]" LOC=AF13 | IOSTANDARD=LVCMOS15;#SW[15]
NET "BTN[2]" LOC=AE13 | IOSTANDARD=LVCMOS15;#SW[15]
NET "BTN[3]" LOC=AF8 | 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
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;
```

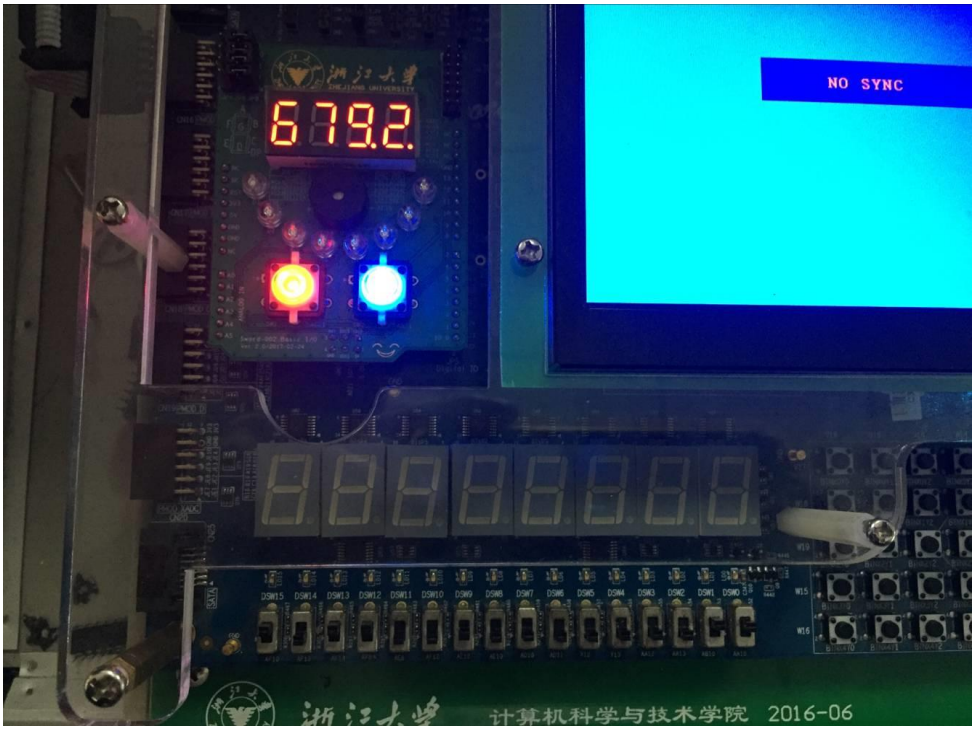
4.2.8 在 SWORD 开发板上验证是否实现了计数板的功能

# 五、实验结果与分析

5.1 实验结果

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




最右四个开关控制四个 points，右数 5-8 个开关控制数字的开关，左数第 1-4 个控制数字的加减增加。如图，实验能完美地做到四个不同数字的显示和开关。

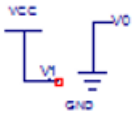
5.2 实验分析

该实验是在实验六的基础做出的优化，现在屏幕上能显示的数字不在是完全一致的了。动态扫描的存在让我们能用开关控制，看到四个不同的数字出现在屏幕之上。

# 六、讨论、心得

实验七是在实验六上做出的优化，这个实验给我带来了很大的困难。对于原理图的运用我现在已经是很熟练了，但对于实验中第一次出现的 Verilog 语言书写，我感到很大的理解困难。另一大困难是对于时序电路的理解，我一直不知道它的原理，这也给我当时写 top.v 带来了很大困难。

值得一提的是我在这次实验中的一个小 bug，花了我大约 4 小时找到它。我在




DisplaySync 模块中因为疏忽少画了 ，直接导致我过了编译却在物理验证方面始终错误。太惨痛了这教训，我以后一定要认真画图，争取一遍过。

我觉得之后的实验中，我要加强对于 Verilog 语言的理解，并且能努力理解 ppt 上的内容。