

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机逻辑设计基础 |
| 姓 名： | 汪珉凯 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | 人工智能 |
| 邮 箱： | 904738557@qq.com |
| QQ 号： | 904738557 |
| 电 话： | 18157421318 |
| 指导教师： | 王总辉 |
| 报告日期： | 2023年 11月 22 日 |

**浙江大学实验报告**

课程名称： 计算机逻辑设计基础 实验类型： 综合

实验项目名称： 常用电子仪器的使用

学生姓名： 汪珉凯 学号：3220100975 同组学生姓名： 岳晟稼

实验地点： 紫金港东四509室 实验日期： 2023 年 9 月 21 日

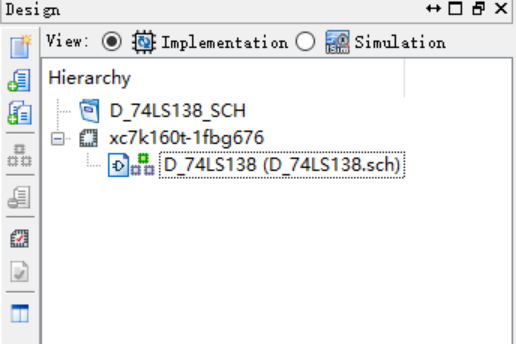
1. **操作方法与实验步骤**

**（一）原理图设计实现 74LS138 译码器模块**

**1、新建工程**

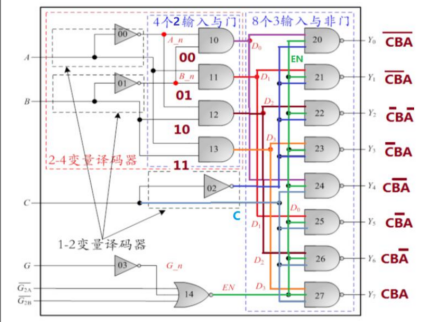
新建工程，工程名命名为 D\_74LS138\_SCH。

**2、新建 Schematic 源文件**

新建 Schematic 源文件，文件命名为 D\_74LS138。新建好的工程和源文件如下：

**3、绘制 74LS138 译码器原理图**

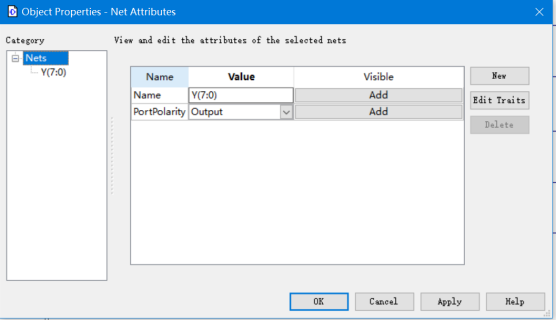
**（1）74LS138 译码器原理**



**（2）总线结构的绘制**

①先用 ADD\_WIRE 工具画一条线。

②双击该条电线，编辑这条线的名称，如下：

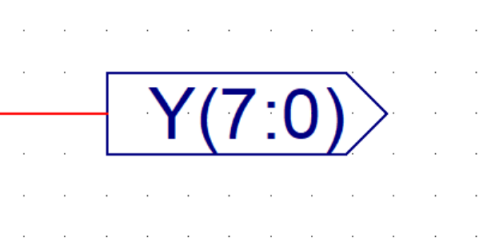


其中，Name 的格式为 name(n:0)，“name”为这条线的名字，“(n:0)”表示这条线为

（n+1）条单条导线的总线，单条导线的编号依次为 0~n。

③总线数据流的输入输出

要将总线所有数据一起输入或输入，只需要在导线的一头使用 I/O Device 即可。



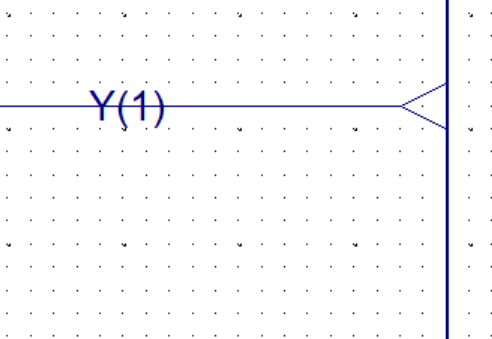
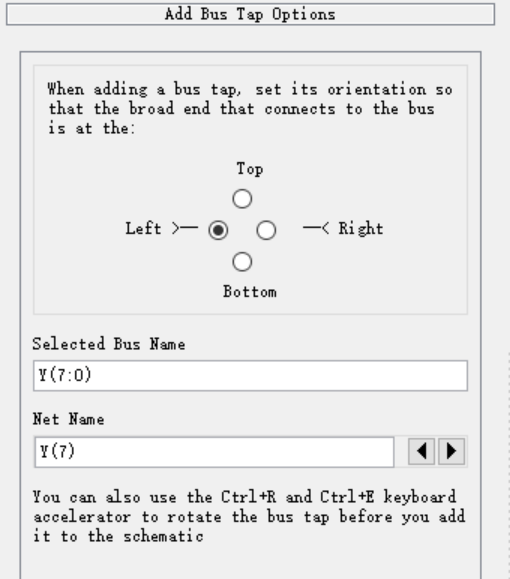
④单独导线的输入输出

要把总线的单个或者部分数据输入/输出，需要用到 Bus Tap。



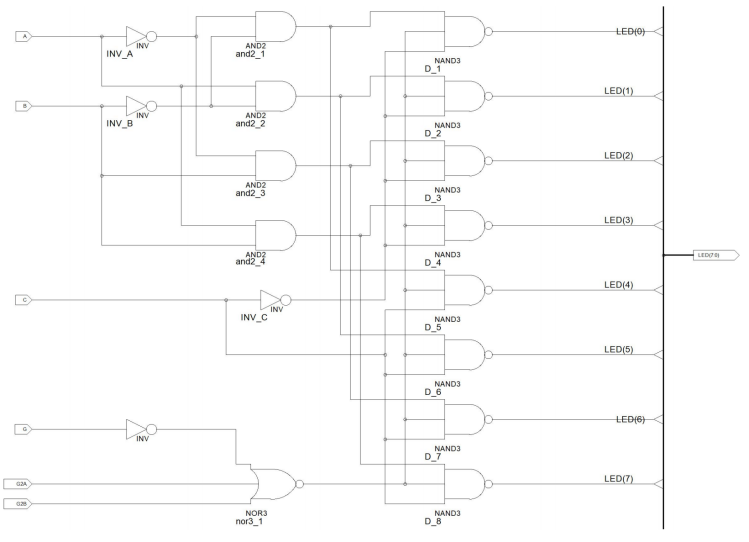
点击总线，左侧资源框里即该条总线的 Bus Tap 放置模式。上半界面

是bus tap 的朝向，下半部分是总线的哪条线的单独输入/输出端口。放好后的样式如下。



**（3）输入、输出端口的重命名**

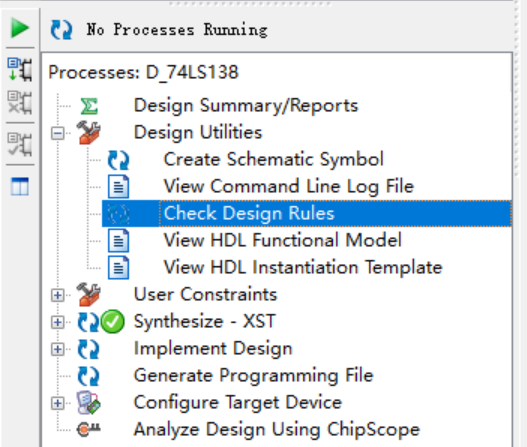
绘制好的原理图如下：



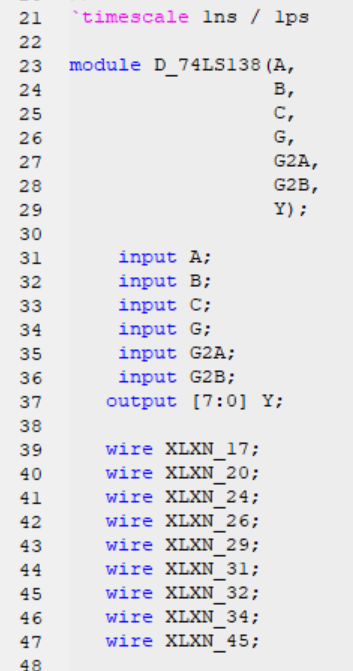
**4、编译**

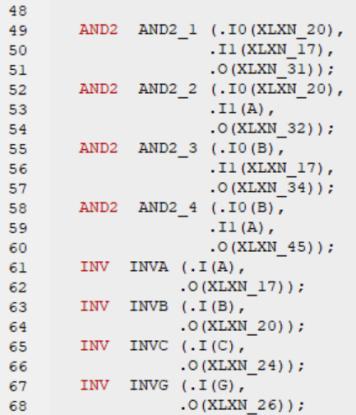
依次对该 schematic 文件运行 Check Design Rules、Synthesize。运行 View HDL Functional

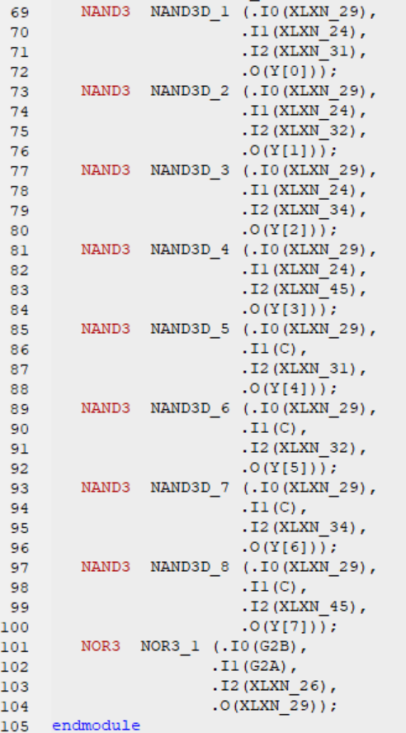
Module，查看并学习 Verilog 代码。



原理图的 Verilog 代码如下：



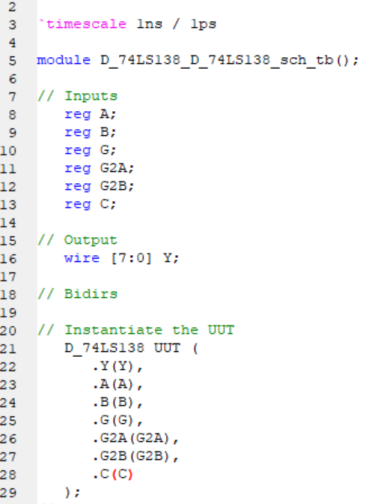


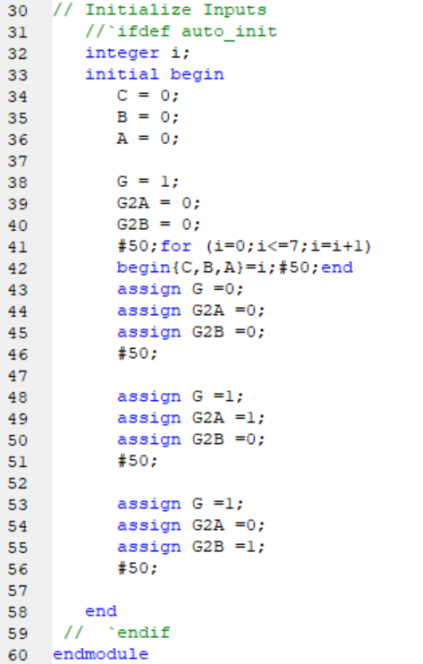


**5、仿真**

对该原理图进行仿真，检查逻辑关系是否正确。

仿真激励代码如下：





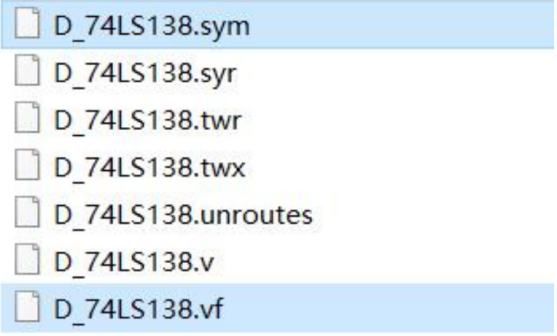
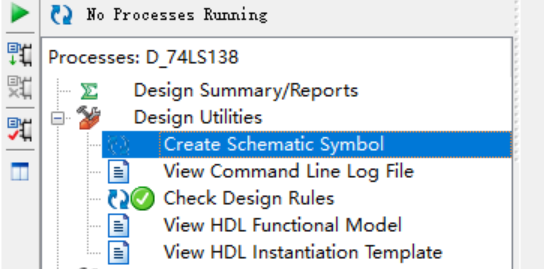
**6、生成符号图和 VF 文件**

这一步的目的是将我们的电路封装为一个元件，以后可以直接取用而不用从逻辑门开始。

运行 Create Schematic Symbol，产生该元件的.sym 文件。同时，在上述的 View HDL

Functional Module 这一步，自动生成了该元件的.vf 文件。如果在其他工程中要使用该元

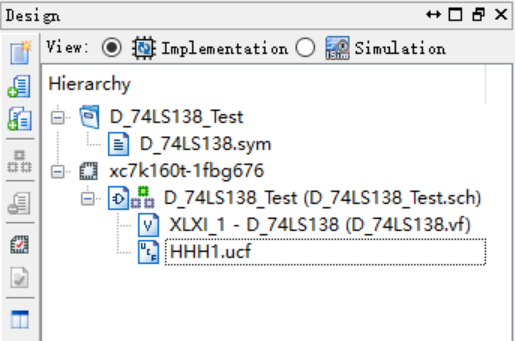
件，只需要把.sym 和.vf 文件加入到其他工程。



**（二）验证 74LS138 译码器的逻辑功能**

**1、新建工程 D\_74LS138\_TEST。**

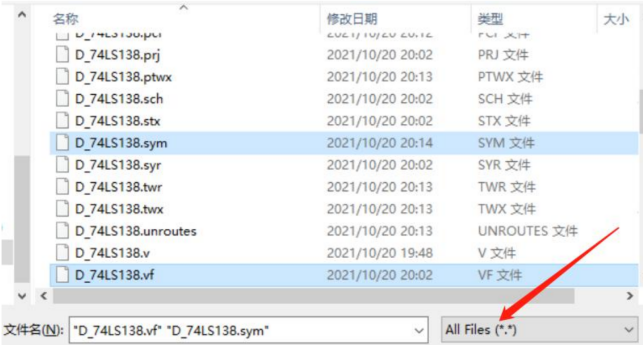
**2、新建 Schematic 文件“D74LS138TEST”。**



**3、在资源区域右键，选择“Add Copy of Source”，将第一个工程目录下的 D74LS138.vf**

**和 D74LS138.sym 文件加入到工程。**

注：（1）在文件目录下不能直接找到.vf 和.sym 后缀的文件，需要将文件类型切换到 All File。



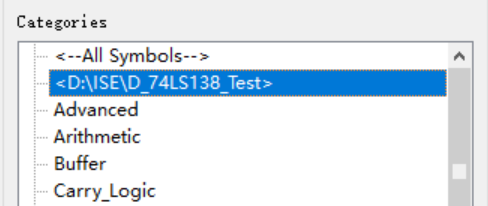
（2）.sym 文件是运行“Create Schematic Symbol”后生成的元器件符号封装文件。

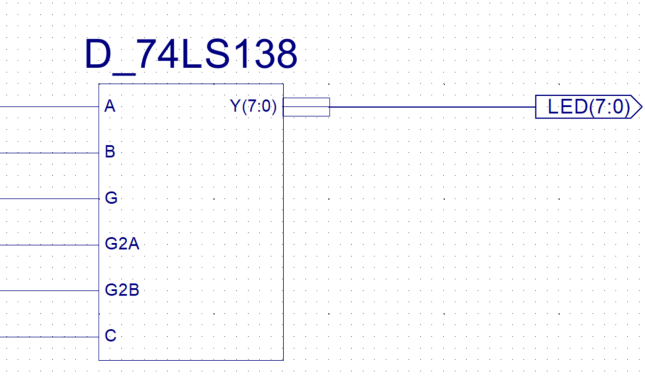
.vf 文件是运行“View HDL Functional Model”后生成的 schematic 文件的 verilog 代码文件，

描述元器件的内部结构，控制逻辑功能的实现。

4、**利用 D\_74LS138 元件绘图**。

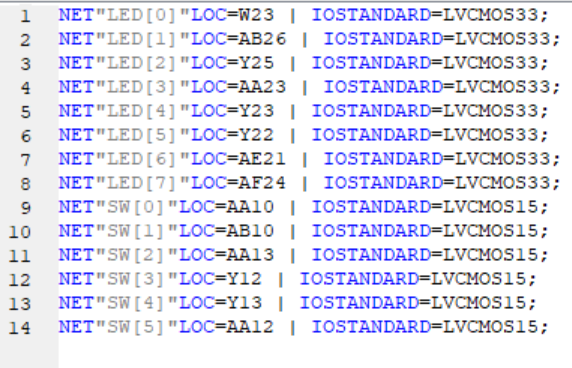
导入元件后，即可在 symbols 中找到自建的元件。





**5、引脚约束**

在原理图编译通过后，新建 ucf 文件，进行引脚约束。引脚约束代码如下：



**6、功能验证**

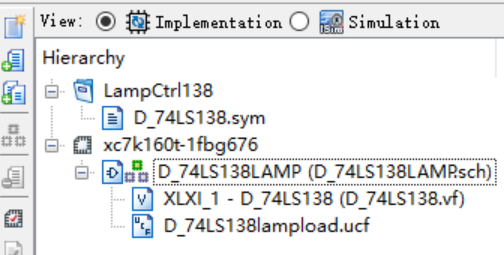
利用和实验 4 相同的操作，生成 bit 文件，导入到 sword 板中，对照真值表，验证

D\_74LS138 译码器的逻辑功能。

**（三）实现楼道灯的控制**

**1、新建工程 LampCtrl。**

**2、新建 Schematic 文件 Lamp。**



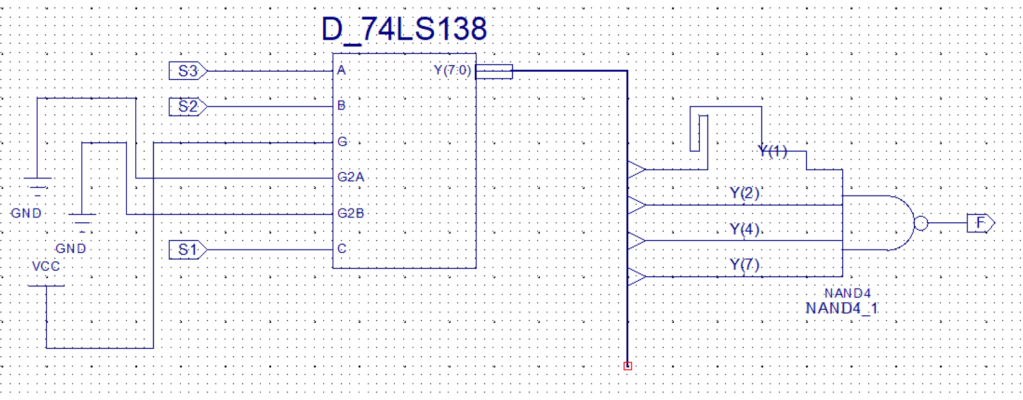
**3、将 D\_74LS138 译码器的.vf 和.sym 文件添加到此工程中。**

**4、绘制楼道灯的原理图**

1 用 VCC 表示，0 用 GND 表示。

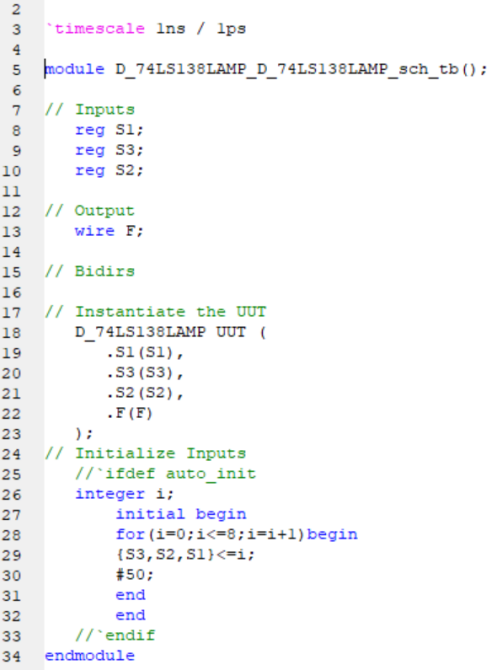
利用此电路可实现：当 ABC 中任意一个的高低电平状态改变时，输出信号都会改变，

从而实现楼道灯（房间灯）按任意一个开关都能改变灯的暗灭状态的功能。



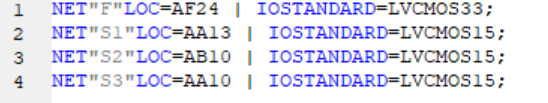
**5、楼道灯的仿真**

新建仿真激励文件 lamp\_sim.v，加入信号灯的仿真代码（如下），进行仿真。

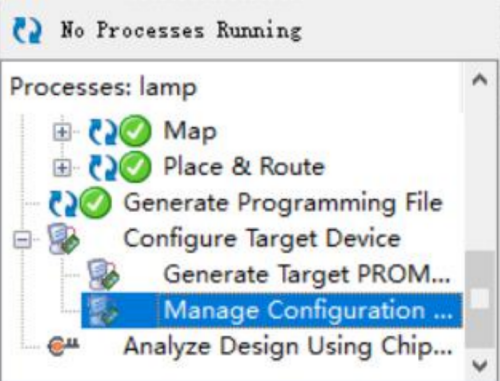


**6、引脚约束**

对楼道灯和开关进行引脚约束，代码如下：



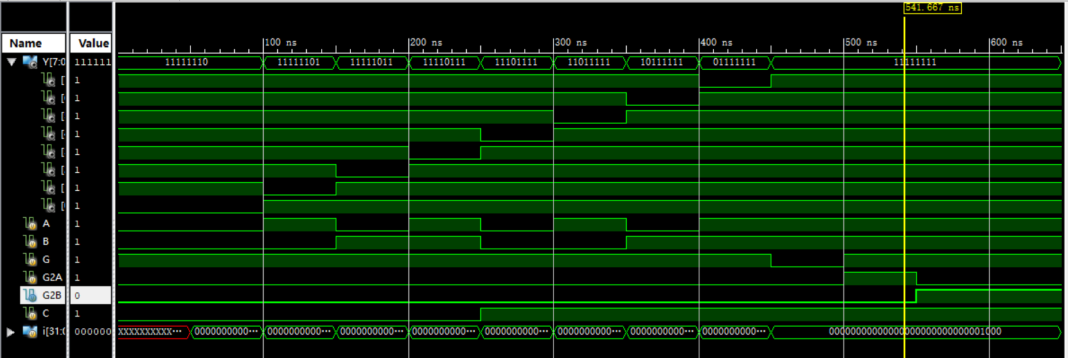
**7、生成 bit 文件，将 bit 文件下载到 sword 板上，进行功能测试。**



1. **实验结果与分析**

**（一）74LS138 译码器的设计**

**1、译码器的仿真结果**

****

CBA 组成的二进制数的十进制值对应的 LED 的输出为 0，其他 LED 输出为 1，说明译

码功能成功实现。

1. **bit 文件在 sword 板上的运行结果**

**不同输入对应开关如下所示：**

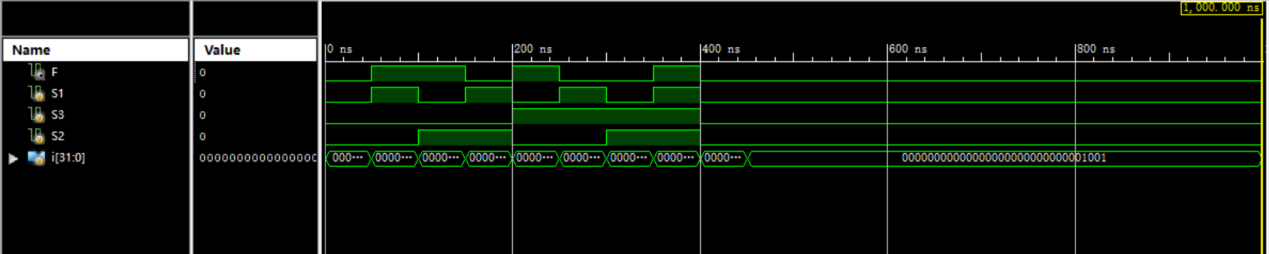
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **G** | **G2A** | **G2B** | **A** | **B** | **C** |
| **AA12** | **Y12** | **Y13** | **AA13** | **AB10** | **AA10** |

**具体实验结果如下所示：**

|  |  |  |
| --- | --- | --- |
| **G G2A G2B** | **CBA** | **结果** |
| **100** | **111** | **0a40e9d5a0e13546fb4fb37c30df52a** |
| **100** | **110** | **f5155fa4912372be34be85749c4ab27** |
| **100** | **010** | **213163443cd9368f3f2a53b96af0920** |
| **000** | **000** | **9efe96f78add13d472e74e0549f0b60** |
| **100** | **000** | **181f4dd253e337078111c9f27025604** |
| **100** | **101** | **c68cc8457a23311a792f7fc4b48bd94** |

**（二）楼道灯功能的实现**

**1、楼道灯的仿真结果**

****

按照格雷码的顺序写出真值表。

|  |  |  |  |
| --- | --- | --- | --- |
| **S3** | **S2** | **S1** | **F** |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **0** |
| **1** | **0** | **0** | **1** |

可以看到，每当我们改变其中一个输入，输出结果总是会改变，说明楼道灯功能可以实现。

**2、bit 文件在 sword 板上的运行结果**

|  |  |  |  |
| --- | --- | --- | --- |
| S3 | S2 | S1 | 结果 |
| 1 | 1 | 0 | 2cc075f3efc995703c9f65f4879b8e3 |
| 1 | 0 | 0 | cc077ff7ce9d667446e36fcb4e671fd |
| 0 | 0 | 1 | c28c5da5b134feaf43e249471ecfbc1 |
| 0 | 1 | 1 | a2b251fc0465e588e62231dbd0f2cb7 |
| 0 | 0 | 0 | 6b0ad7bac36c953cbd88a91d3ec2ede |

1. **讨论、心得**

本次实验内容可以说是实验4的扩展，我一开始以为会比较基础，但越到后面越发现还是有很多需要注意的小细节。比如画图过程中，可能会存在断线，但图片一经过放缩就很难发现，只有最后在运行的时候会出现error。我也慢慢意识到实验四作为基础的重要性。

本次实验主要学习了总线BUS的使用以及一些新的symbol的用法，以及如何自己画一个元器件并进行封装。此外，对给类器件的重命名（尤其是导线）的意义在本次实验尤为突出，可以让我避免混淆电路。

总体来说，我在实验四以后复习了该平台的基本用法，因此做实验5还是比较顺利的。

**浙江大学实验报告**

课程名称： 计算机逻辑设计基础 实验类型： 综合

实验项目名称： 七段数码管显示译码器设计与应用

学生姓名： 汪珉凯 学号： 3220100975 同组学生姓名： 岳晟稼

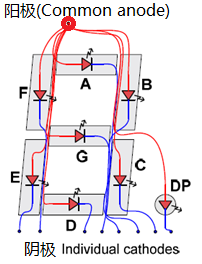
实验地点： 紫金港东四509室 实验日期： 2023 年 10 月 26 日

# 操作方法与实验步骤

1. 七段数码管简介

七段数码管一共由a~g 七个数码管组成，再加上一个小数点。

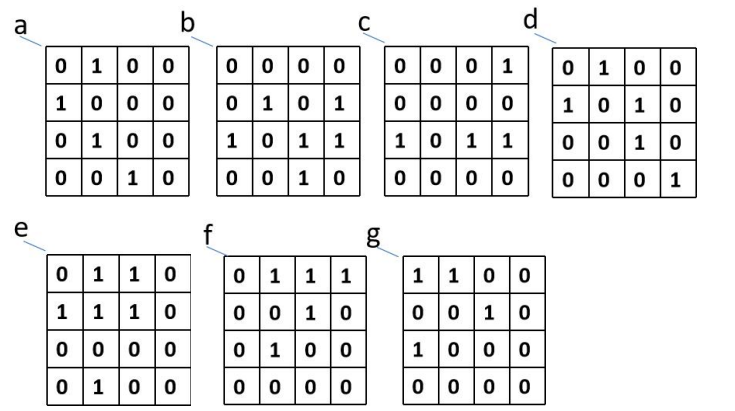
值得注意的是，当输入信号为 0 时，对应数码管亮；输入信号为 1 时，对应数码管灭。 与我们的一般习惯相反，因为输入信号是从阴极输入的。



1. 原理图设计实现MyMC14495模块
2. 设计目标

利用该模块，能把输入的 4 位二进制数的信号转换为对应的数码管亮暗的信号输出

2、利用卡诺图设计逻辑关系



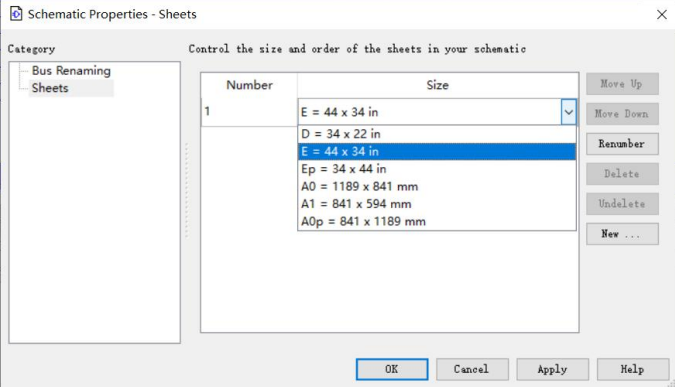
3、新建工程，命名为MyMC14495。

4、新建 Schematic 源文件，命名为 MyMC14495。

5、绘制该模块的原理图

（1）画布尺寸的改变

由于该模块的门数量较多，故预置的画布尺寸可能不够。双击画布空白处，即可在弹出的对话框中修改画布的尺寸。

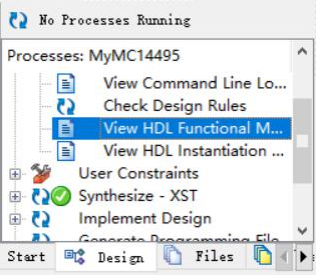


（2）原理图



6、编译

运行Check Design Rules，通过后再运行Synthesize - XST。之后运行View HDL Functional Model，生成 vf 文件，学习 verilog 代码，为之后编程做准备。



7、仿真

对该模块进行仿真激励，检验该模块功能是否正确。

由于原理图使用的逻辑门较多，因此这一步非常重要，需要认真检验。

仿真激励代码如下：

`timescale 1ns / 1ps

module MyMC14495\_MyMC14495\_sch\_tb();

// Inputs

    reg D3;

    reg D2;

    reg D0;

    reg D1;

    reg LE;

    reg point;

// Output

    wire a;

    wire b;

    wire c;

    wire d;

    wire e;

    wire f;

    wire g;

    wire p;

// Instantiate the UUT

MyMC14495 UUT (

    .D3(D3),

    .D2(D2),

    .D0(D0),

    .a(a),

    .b(b),

    .c(c),

    .d(d),

    .e(e),

    .f(f),

    .D1(D1),

    .g(g),

    .LE(LE),

    .point(point),

    .p(p)

);

// 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

        end

        #50

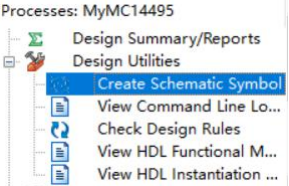
        LE = 1;

    end

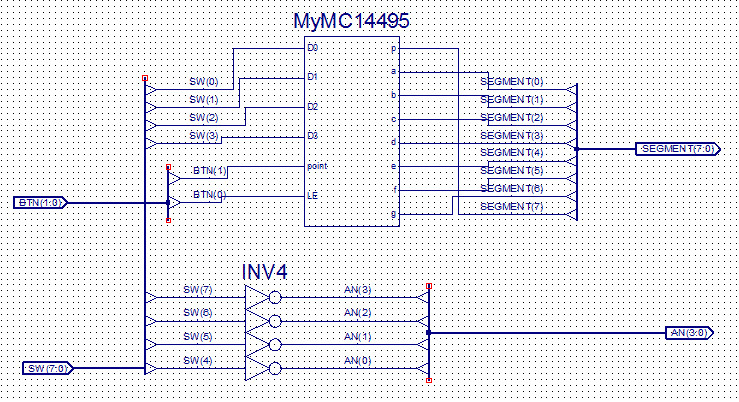
endmodule

8.生成元器件符号（symbol）

运行Create Schematic Symbol，生成MyMC14495模块的元器件。



1. 实现数码管显示
2. 新建工程DispNumber。
3. 新建Schematic文件DispNumber。
4. 将第一个工程生成的MyMC14495模块中的.vf和.sym文件导入。
5. 绘制原理图
   1. 调用 MyMC14495 模块，绘制数码管显示的原理图。



图中 AN 的作用：sword 板上该区域一共有四位七段数码管，因此我们利用 AN，来控 制每一个数码管是否亮。

1. 编译
2. 引脚约束

对输入输出进行引脚约束。

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;

- 23 -

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;

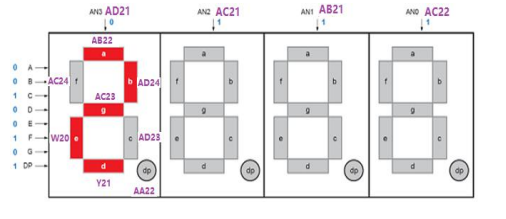
net "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;

net "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;

net "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;

net "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;

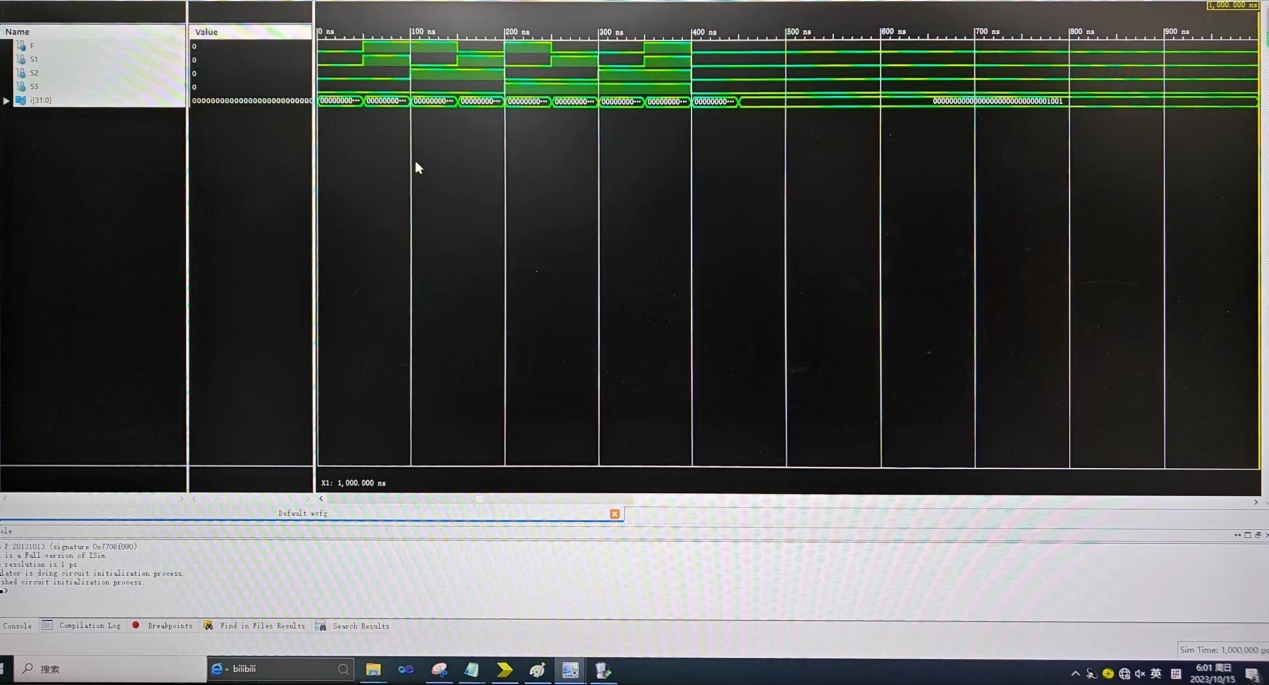
MOS33 对应 sword 板上数码管的区域，MOS15 对应按钮区域。应注意四位七段数码管 共用一套阴极，而不是每位数码管都有自己的阴极，这为下一个实验我们实现每位数码管独 立显示数字做出铺垫。

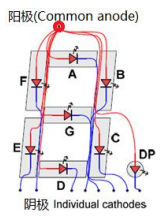


7、生成 bit 文件，并下载到 sword板上检验效果。

# 二、实验结果与分析

1、MyMC14495的仿真激励





依据七段数码管的分布图来检查仿真结果是否出错。

1. 七段数码管的显示

|  |  |  |  |
| --- | --- | --- | --- |
| BTN | SW(7:4) | SW(3:0) | 效果图 |
| 10 | 0000 | 1000 |  |
| 00 | 0000 | 1000 |  |
| 01 | 0000 | 1000 |  |
| 01 | 0001 | 0000 |  |
| 01 | 0001 | 0001 |  |
| 01 | 0010 | 0010 |  |
| 01 | 0011 | 0011 |  |
| 01 | 0100 | 0100 |  |
| 01 | 0101 | 0101 |  |
| 01 | 0110 | 0110 |  |
| 01 | 0111 | 0111 |  |
| 01 | 1000 | 1000 |  |
| 01 | 1001 | 1001 |  |
| 01 | 1010 | 1010 |  |
| 01 | 1011 | 1011 |  |
| 01 | 1100 | 1100 |  |
| 01 | 1101 | 1101 |  |
| 01 | 1110 | 1110 |  |
| 01 | 1111 | 1111 |  |

# 三、讨论、心得

画图时保持耐心，连对每一根线，连完一根之后要检查一下，然后发现在左上角可以选择删除部分线，这让我连错线之后修改时更加方便，还有需要注意文件类型，中间有一次sch文件选错之后不能使用，尽量保持细心。

以上是本次的报告与心得。

**浙江大学实验报告**

课程名称： 计算机逻辑设计基础 实验类型： 综合

实验项目名称： 多路选择器设计与应用

学生姓名： 汪珉凯 学号：3220100975 同组学生姓名： 岳晟稼

实验地点： 紫金港东四509室 实验日期： 2023 年 11 月 2 日

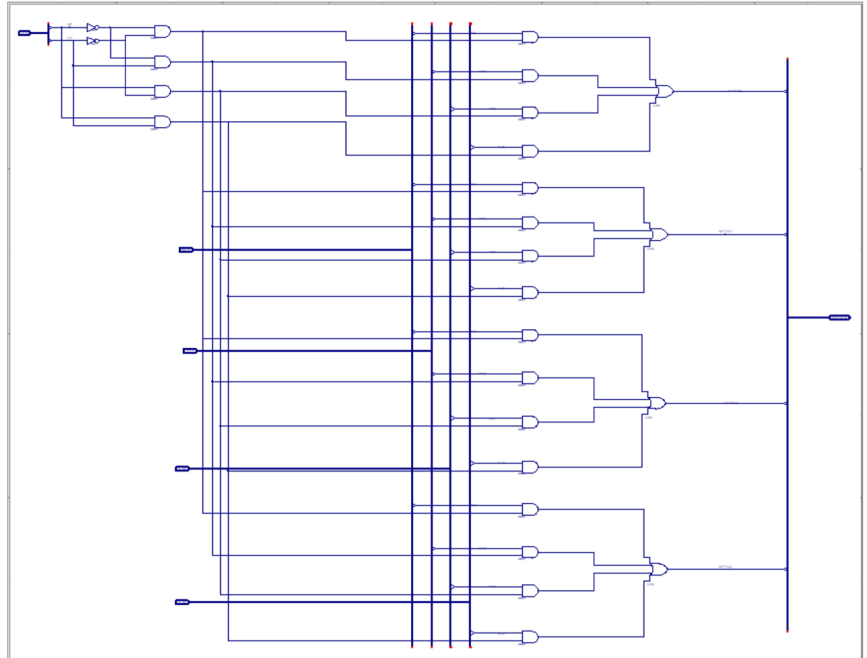
1. **操作方法与实验步骤**

**1、4 位四选一多路选择器的设计**

⑴新建工程 Mux4to1b4\_sch

⑵新建文件 Mux4to1b4

⑶绘制逻辑电路图



⑷对电路进行仿真

⑸生成.sim 和.vf 文件

Check Design Rules，在原理图无误情况下运行 View HDL Functional Model 和 Create Schematic Symbol，生成此原理图的.vf 文件和.sym 文件。

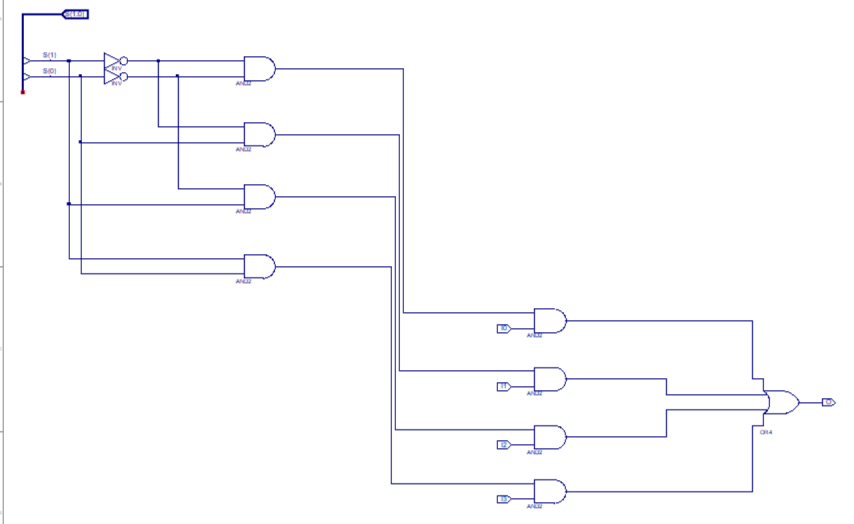
**2、计分板应用设计**

⑴新建工程 ScoreBoard

⑵添加 Mux4to1b4\_sch 的.vf 和.sim 文件

⑶新建 Mux4to1.sch 文件

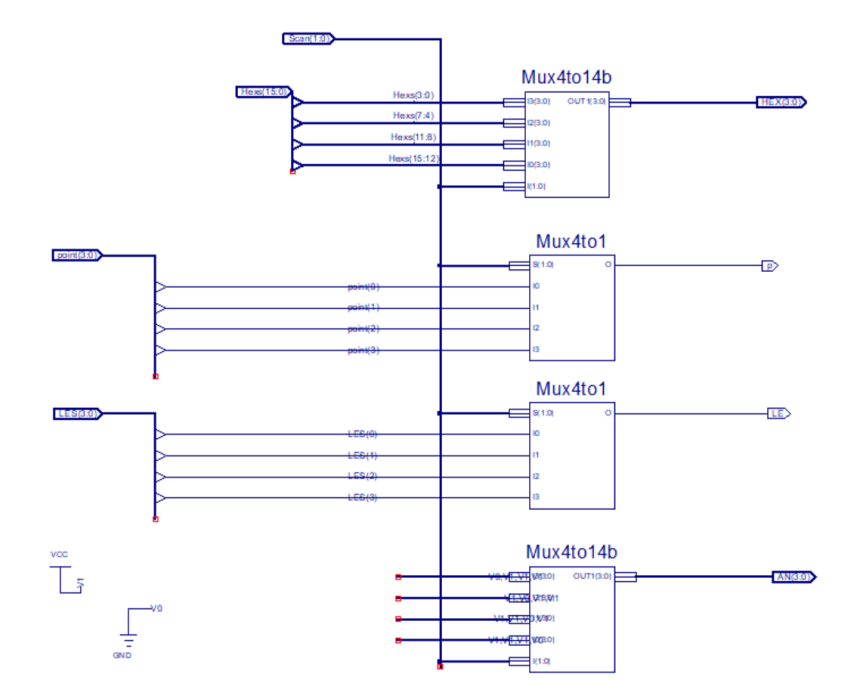
⑷绘制逻辑图



⑸生成.sim 和.vf 文件

⑹新建 DisplaySync.sch **文件**

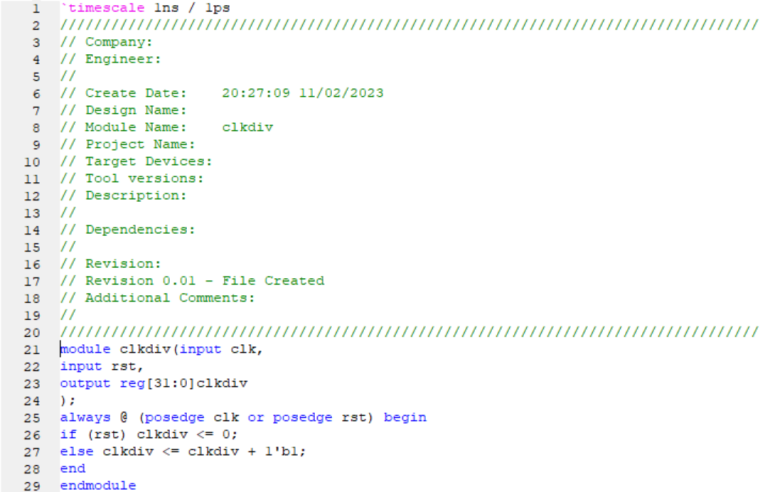
⑺绘制逻辑图



⑻生成.sim 和.vf 文件

⑼新建 clkdiv.v 文件

⑽写入代码

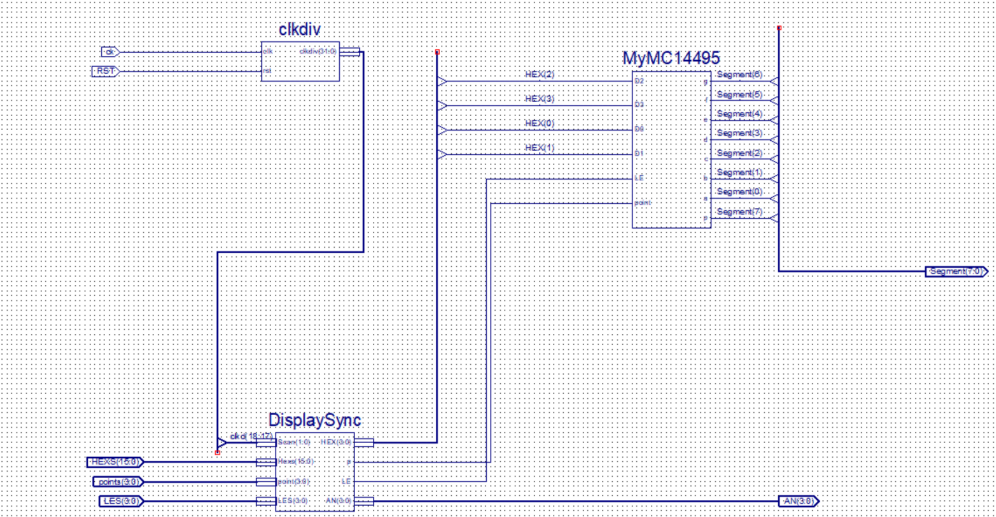


⑾生成.sym 文件

⑿导入 MyMC14495 的.sim 和.vf 文件

⒀新建 disp\_num 文件

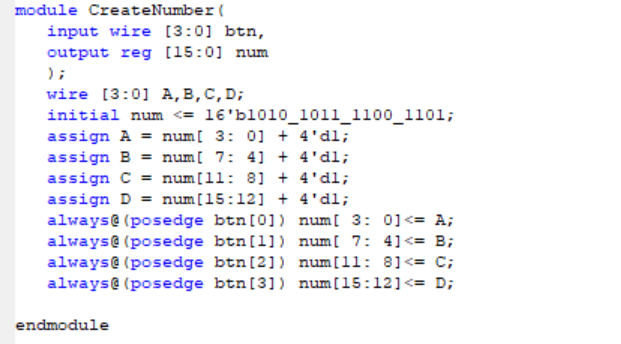
⒁绘制逻辑图



⒂生成.sym 和.vf 文件

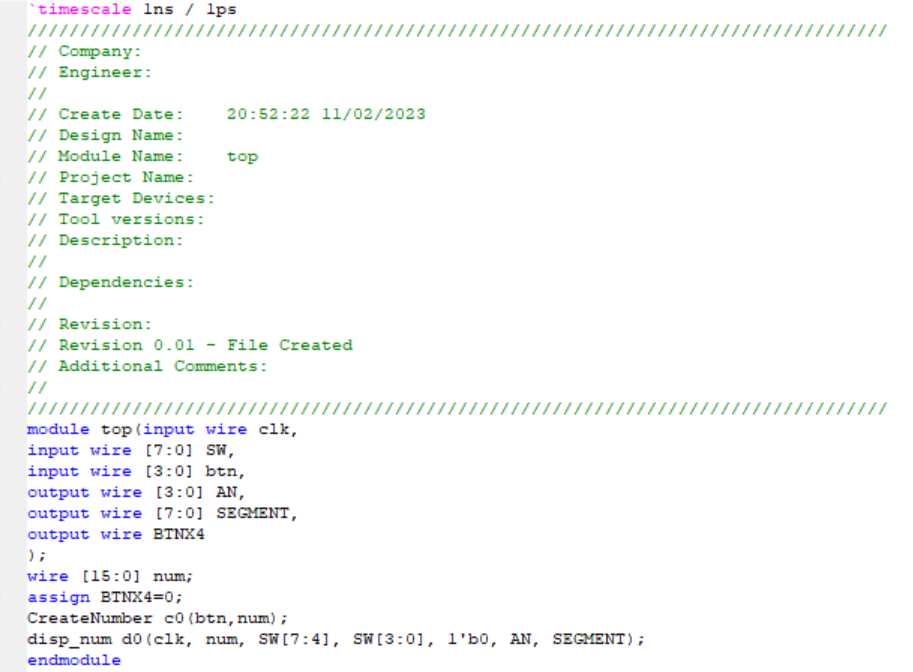
⒃新建 CreateNumber 模块

⒄写入代码



⒅新建顶层模块

⒆写入代码

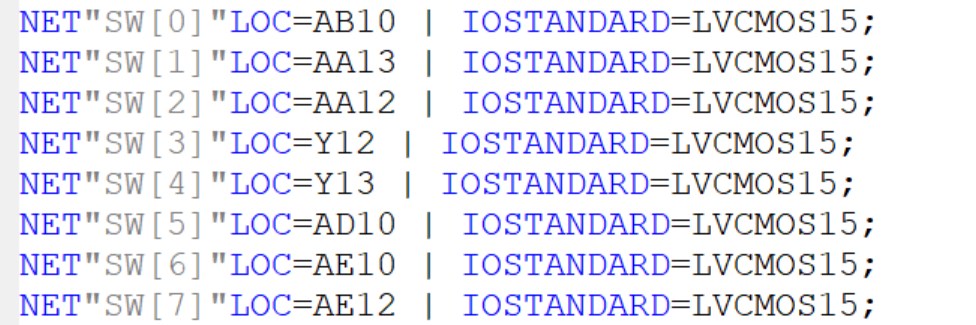


⒇引脚约束

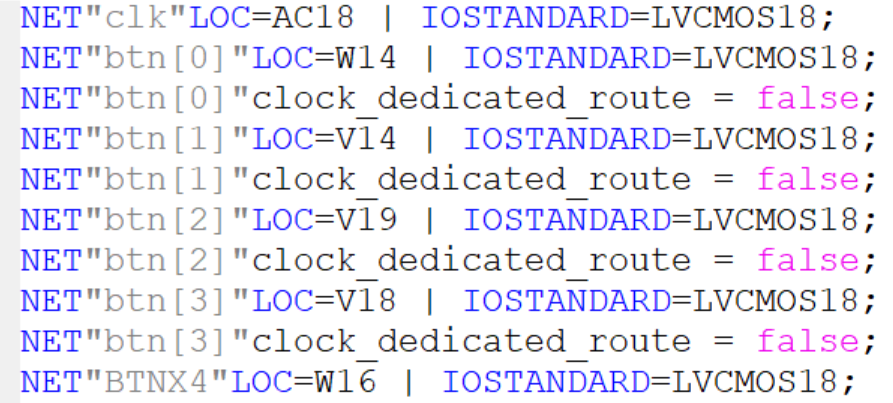
1.**七段数码管引脚约束**



2.**数码管等开关的引脚约束**



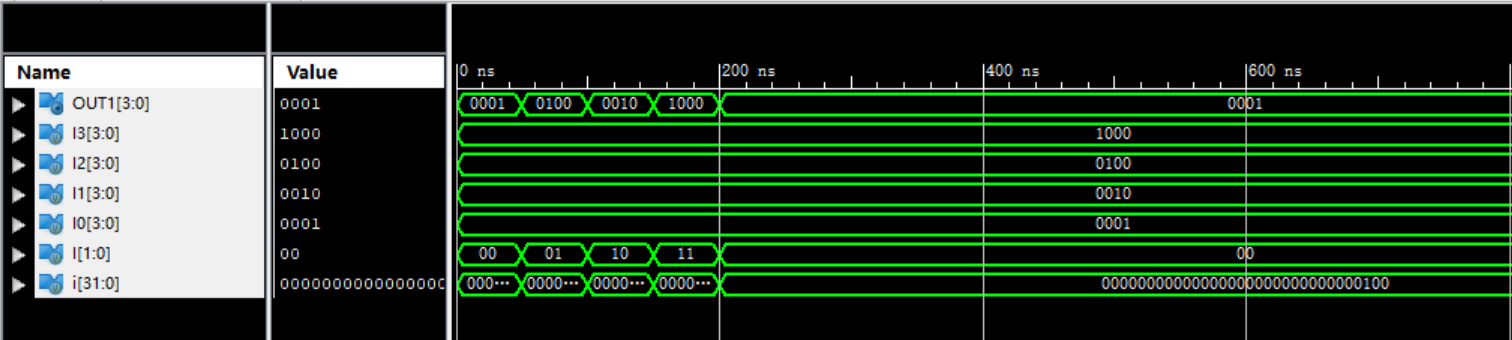
3.**按钮和时钟的引脚约束**



⒇生成 bit 文件并下载到 sword 板上实验

1. **实验结果与分析**

**2.1 关于数据选择器**

****

在 0ns 到 50ns 过程中 I=00，选择 l 输出 OUT1(3:0)=0001；

在 50ns 到 100ns 过程中 I=01，选择 l1 输出 OUT1(3:0)=0010；

在 100ns 到 150ns 过程中 I=10，选择 l2 输出 OUT1(3:0)=0100；

在 150ns 到 200ns 过程中 I=11，选择 l3 输出 OUT1(3:0)=1000。

**2.2 关于计分板**

以下四张图是通过改变按键改变小数点的位置：

|  |  |
| --- | --- |
| fa0697037a9e46b0431d33c5e5b6661 | 8867a31eb12c13b45078ebec1cca114 |
| 412778494b87d6af346e7c6cb83c84b | 9a16daada748859d0041b1fe68dac07 |

以下四张图是通过改变按键改变显示的位数：

|  |  |
| --- | --- |
| 6584db715282079832a67441ef0f356 | 2684718444eb5544dee93883773087f |
| 9ad3c873604bdc8e9ed9be44f0999cc | 541637ba952924588ed223d18414c03 |

以下三张图是通过按下加分键来改变显示：

|  |  |
| --- | --- |
| **0446fc19989eab81165704c8959dfec** | **63c9329fd4fe02fe2af8ce79df8326f** |
| **3bc456bc1663fb66a31cadd8cca2762** | **5c5ac9b712be3eb9961a4a4e7a52955** |

1. **讨论、心得**

本次实验可以说是前几次实验的综合，总体来说难度并不大。但中途有一个地方花了我比较多的时间是我没有想到“OUT”居然也可以作为关键字，因此花费了比较多的时间来排查这个错误。此外，本工程运用了较多的自造模块，这和c语言自己定义函数的思想非常类似。以后我会继续做好预习工作，确保实验进行更加顺利。

1. **个人生活照片展示**

