浙江大学

本科实验报告

课程名称：	计算机逻辑设计基础
姓　　名：	王晓宇
学　　院：	竺可桢学院
专　　业：	计算机
邮　　箱：	1657946908@qq.com
QQ　　号：	1657946908
电　　话：	19550222634
指导教师：	洪奇军
报告日期：	2023 年 11 月 2 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：多路选择器设计与应用

学生姓名：王晓宇 学号：3220104364 同组学生姓名：

实验地点：紫金港东四 509 室 实验日期：2023 年 11 月 2 日

一、操作方法与实验步骤

（一）数据选择器设计

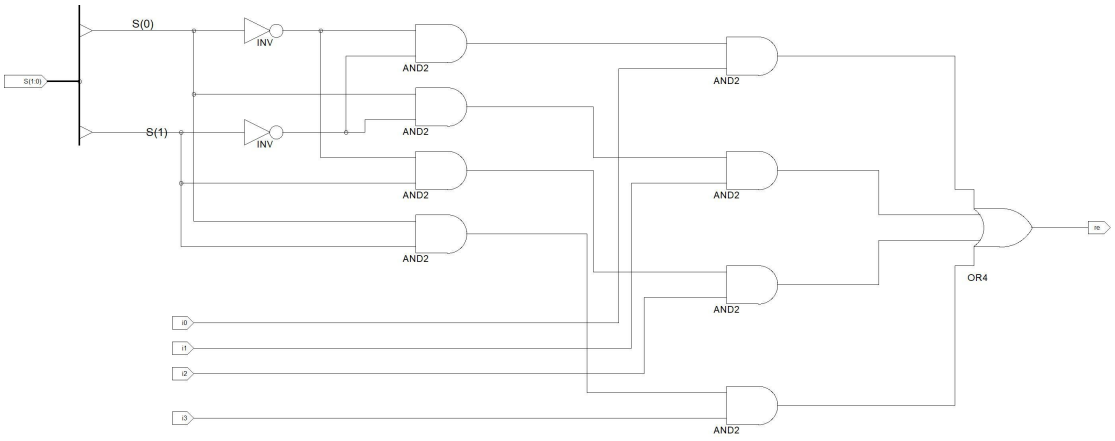
1、新建工程 Mux4to1b4。

2、设计元件 Mux4to1。

（1）新建 Schematic 源文件 MUX4to1.sch。

（2）按照原理图绘制。

此元件实现了 4 路选择器的功能。



（3）生成元件

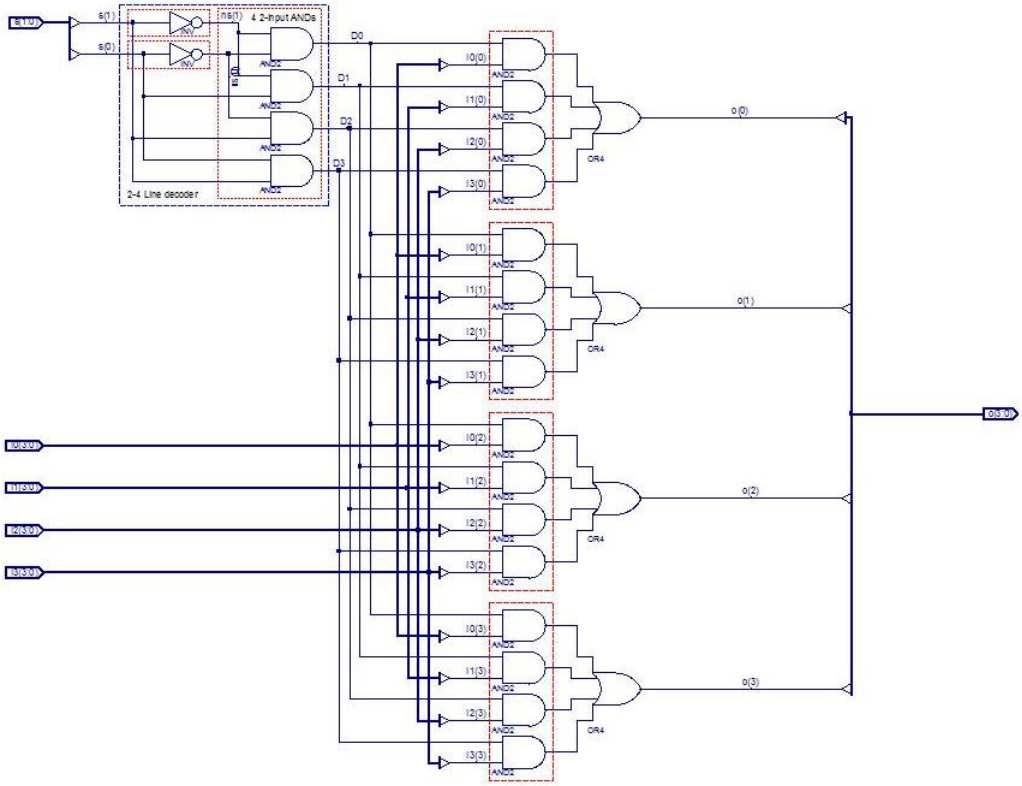
先 Check Design Rules，检查原理图有无绘制错误，接着分别运行 View HDL Functional Model 和 Create Schematic Symbol，生成此原理图的.vf 文件和.sym 文件，为之后使用该元件做准备。

3、设计元件 Mux4to1b4。

（1）设计思路

刚刚绘制的 Mux4to1 仅能实现一位信号的选择，但我们的七段数码管可以显示十六进制数的一位，故需要四位的选择器才能满足 0~F 的数值显示。

这个原件的原理图可以参考如下：

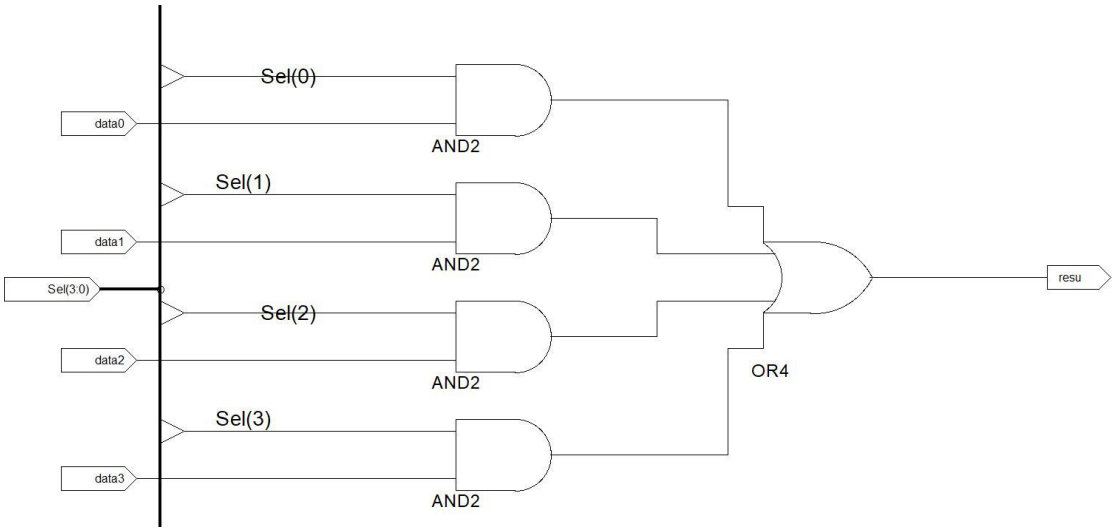


可以看到，对每一位信号我们都绘制的相同的逻辑电路：4 个与门和 1 个或门。故我们可以将其封装为一个原件，这样就只用绘制一次。

（2）绘制原件 AND4OR

①新建 Schematic 文件 AND4OR.sch。

②按照原理图实现此元件。



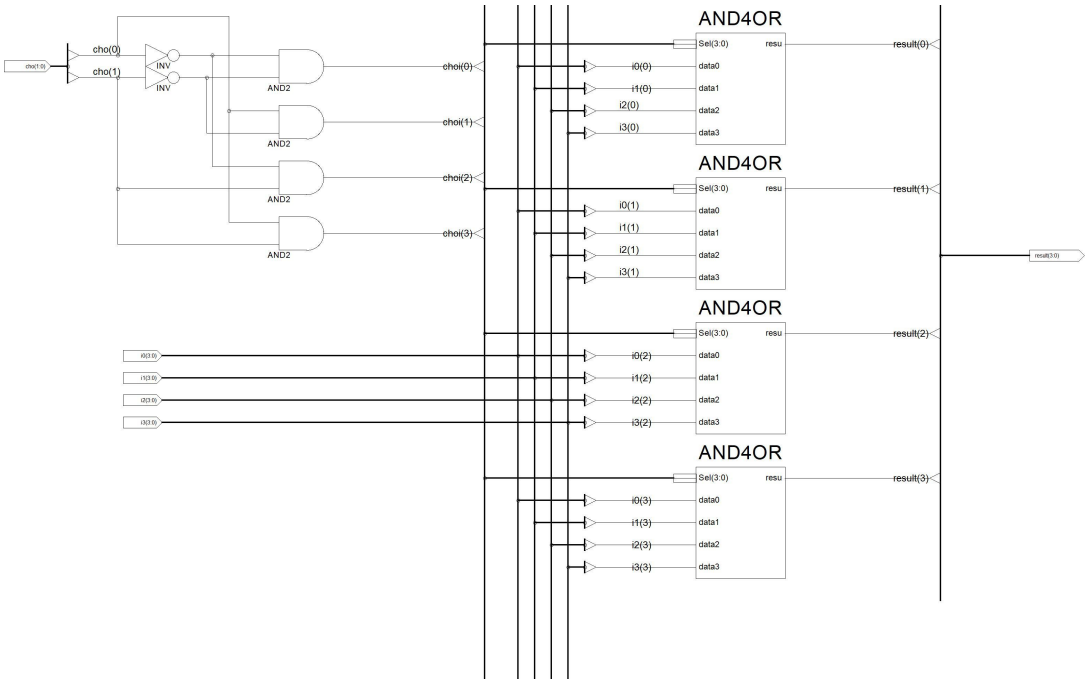
③生成元件

先 Check Design Rules，检查原理图有无绘制错误，接着分别运行 View HDL Functional Model 和 Create Schematic Symbol，生成此原理图的.vf 文件和.sym 文件。

（3）绘制原件 MUX4to1b4

①新建 Schematic 文件 MUX4to1b4.sch。

②利用刚刚生成的元件 AND4OR，绘制原理图。



（4）生成元件

先 Check Design Rules，检查原理图有无绘制错误，接着分别运行 View HDL Functional Model 和 Create Schematic Symbol，生成此原理图的.vf 文件和.sym 文件，为之后使用该元件做准备

4、对 4 位多路选择器进行仿真

（1）新建 Verilog Test Fixture 文件，命名为 test。

（2）本次实验我们自己编写了仿真激励代码。我编写的代码如下：

```
`timescale 1ns / 1ps

module MUX4to1b4_MUX4to1b4_sch_tb();

// Inputs
   reg [1:0] cho;
   reg [3:0] i0;
   reg [3:0] i2;
   reg [3:0] i1;
   reg [3:0] i3;

// Output
   wire [3:0] result;

// Bidirs

// Instantiate the UUT
   MUX4to1b4 UUT (
        .cho(cho),
```

```
        .i0(i0),
        .i2(i2),
        .i1(i1),
        .i3(i3),
        .result(result)
    );
// Initialize Inputs
    integer i, j;
     initial begin
        cho=0;
        i1=0;
        i2=0;
        i3=0;
        i0=0;

     for(j=0;j<=3;j=j+1)begin
            cho=j;
            i1=0;
            i2=0;
            i3=0;
            i0=0;
        for(i=0;i<=15;i=i+1)begin
            #50;
            if(j==0) i0 = i;
            else if(j==1) i1 = i;
            else if(j==2) i2 = i;
            else if(j==3) i3 = i;
        end
     end
     end
endmodule
```

利用这个仿真代码我们顺序遍历了每一种输出情况，用最保险的方法保证了结果没有差错。

（二）计分板应用设计

1、功能描述

（1）设计目标

用 Sword 板上的 4 位七段数码管实现一个计分板，并一一对应使用四个按钮，每按一次按钮，其对应的七段数码管示数就会加 1。

（2）模块说明

①Display Number 模块

输入 4 路的 4 位信号，并根据选择信号输出，控制七段数码管的显像。

②clkdiv 模块

计时器模块。在上一个实验中我们已经知道，4 个七段数码管公用一套 abcdefgp 输入接口，因此我们需要用计时器，在极短时间内快速切换，每切换到一位输出其对应的信号。由于人眼有视觉停留效应，看起来就成了 4 位不同的数字。

③Create Number 模块

由于我们需要实现按钮按一下，数字加一，因此需要这样的模块来进行运算。

2、新建工程 ScoreBoard

3、CreateNumber 模块的设计

（1）新建 Verilog 代码文件 CreateNumber.v

（2）输入 Verilog 代码

```
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
```

（3）点击 Check Syntax，编译该模块

4、clkdiv 计时器的设计

（1）新建 Verilog 代码文件 CreateNumber.v

（2）输入 Verilog 代码

```
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
```

（3）点击 Check Syntax，编译该模块

5、DisplaySync 模块的设计

（1）新建 Schematic 代码文件 DisplaySync.sch

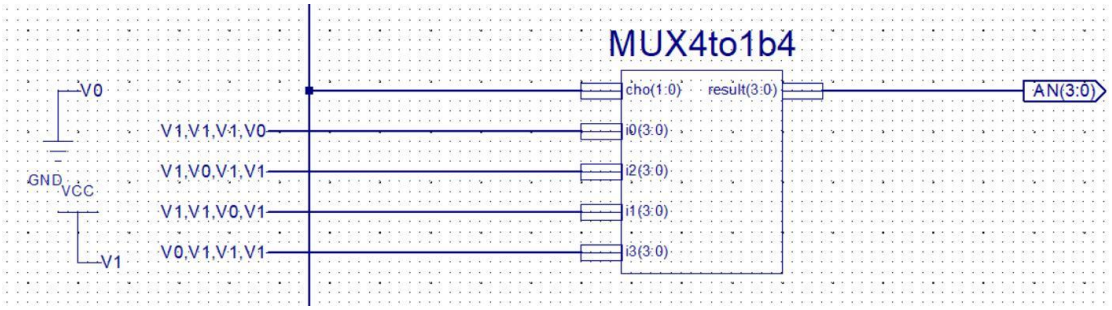
（2）导入第一个工程中生成的 MUX4to1 和 MUX4to1b4 元器件的.vf 和.sym 文件

我们会发现 MUX4to1B4 不能正常使用，原因是我们在该元器件设计时还使用了 AND4OR 元器件，只导入 MUX4to1b4 的.vf 文件，其 Verilog 代码中只写了调用了某个 AND4OR 的元器件，并未描述该元器件的内部结构。故我们需要把 AND4OR 的.vf 和.sym 文件一起导入。

（3）绘制该模块的原理图

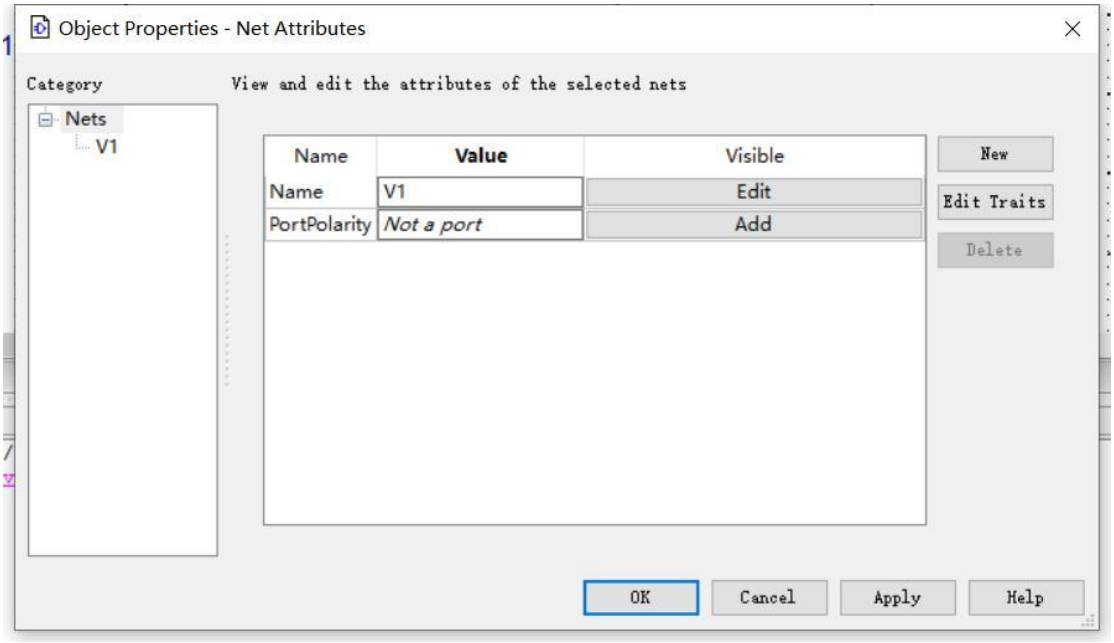
原理图的的 AN 部分控制 4 个七段数码管的亮暗情况，故需要依据 choice 的选择，在一个时刻只亮该时刻对应的数码管。

我们利用了 VCC 和 GND 来控制。若 choice 选择 00，则 i0 被输出，为 V1,V1,V1,V0，第 0 号数码管亮，此时上方同时输出第 0 号数码管的数字，即可达到相应数码管输出相应数字的目的

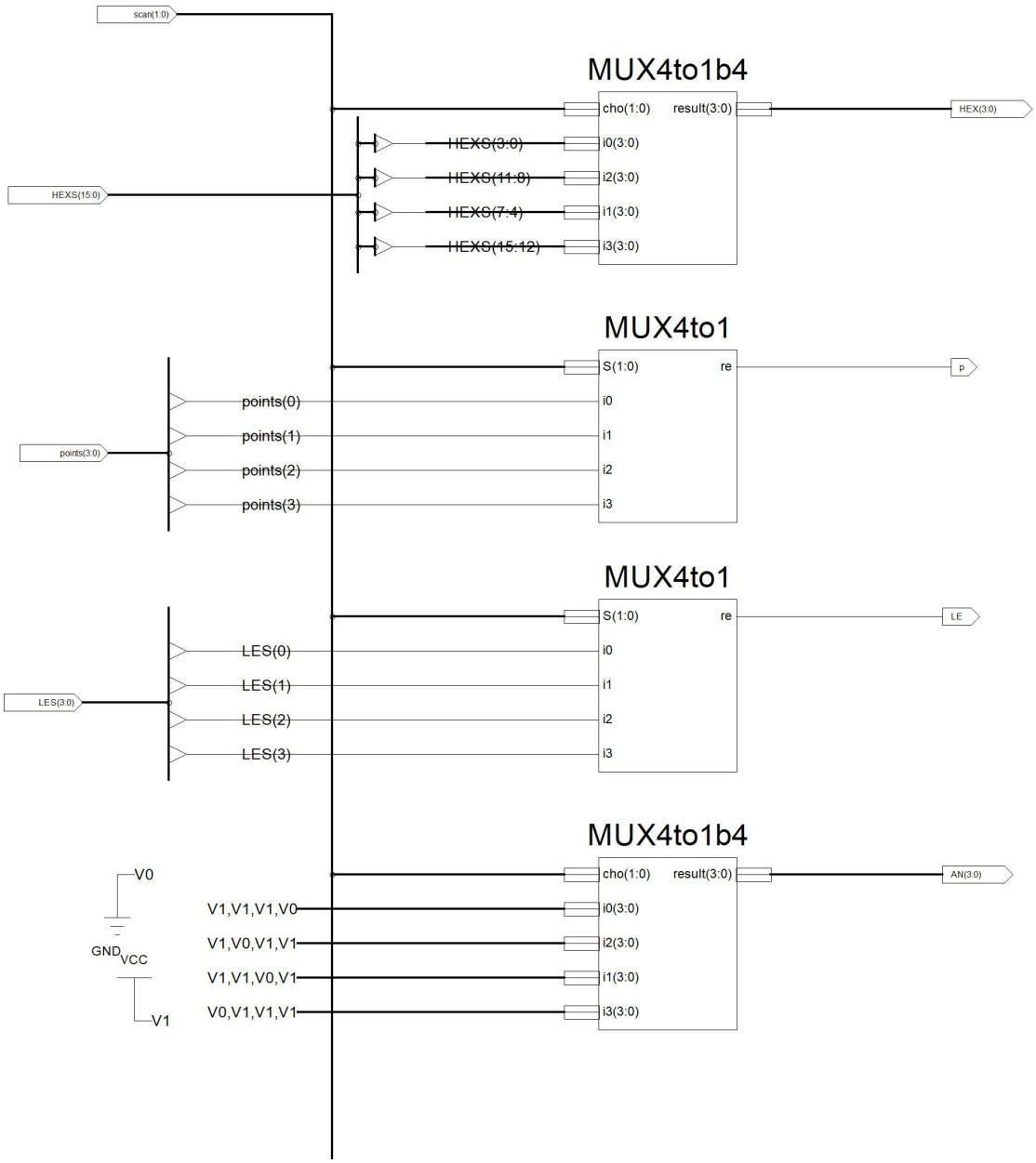


那么我们如何实现这一连接呢？一个方法是用 3 个 VCC，1 个 GND，做成一个总线结构，调节不同的顺序，分别输入到 i0~i3。但这样显得冗杂，需要花费较大的输入成本。

事实上，这里隐含着导线命名的第二个应用（前一个是(3:0)这样定义为总线）。



我们先双击 VCC 的接口的导线，命名为 V1。用同样的方法把 GND 的导线命名为 V0。



（4）编译并生成对应的元器件 DisplaySync。

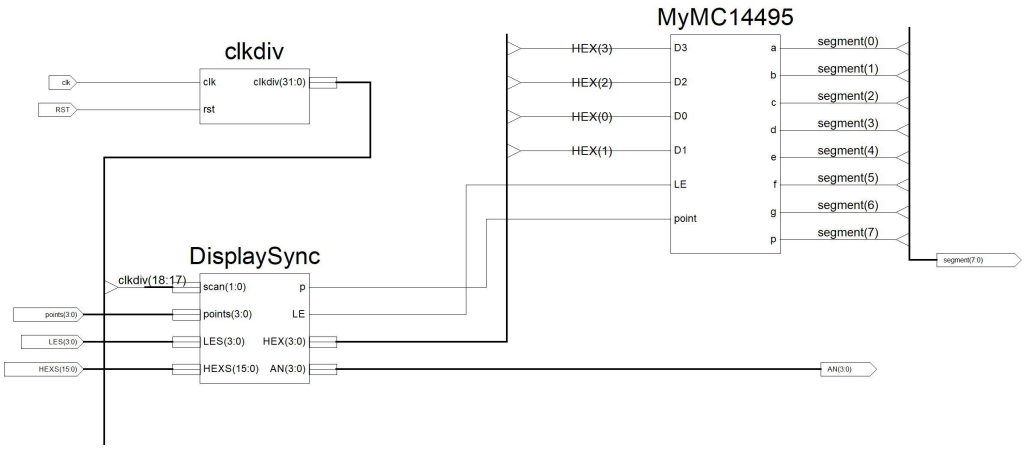
6、DisplayNumber 模块的设计

（1）新建 Schematic 源文件 disp_num.sch

（2）将上一个实验的元器件 MyMC14495 导入

（3）绘制 DisplayNumber 模块的原理图

注：clkdiv 输出的 32 位时钟信号中，我们选择了(18: 17)作为了 scan 扫描的信号输入。实际上这里是随意选择的，差异在刷新速率上，如果我们选择(1:0)，那么就是时钟每次＋1，数码管就会刷新一次。(18:17)是时钟每增加 2^17 后，数码管就刷新一次，这个数字比较符合我们肉眼看不出刷新的间隙，且不至于因刷新过快影响仪器寿命。

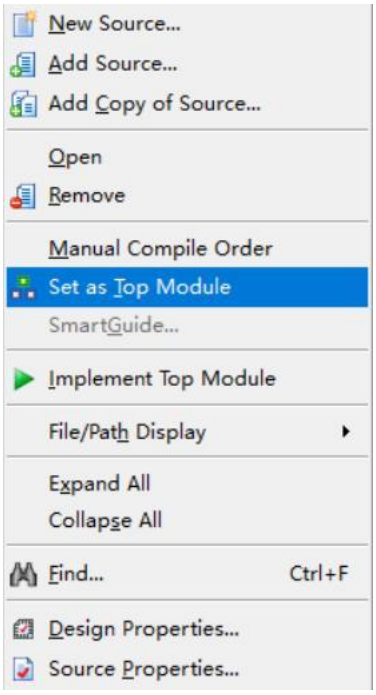


（4）编译并生成对应的 Verilog 代码

7、各模块的汇总

（1）新建 Verilog 代码文件，命名为 top

右键将该文件设置为 Top Module，只有最顶层 module 才能进行引脚约束等操作。



8、引脚约束

（1）七段数码管引脚约束

```
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;
```

（2）数码管等开关的引脚约束

```
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;
```

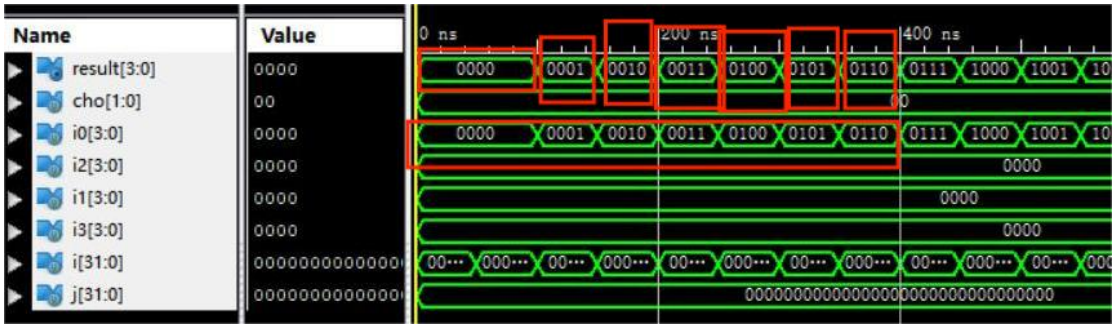
（3）按钮和时钟的引脚约束

```
NET "clk" LOC=AC18    |   IOSTANDARD = LVCMOS18;
NET "btn[0]" LOC=W14  |   IOSTANDARD = LVCMOS18;
NET "btn[0]" clock_dedicated_route = false;
NET "btn[1]" LOC=V14  |   IOSTANDARD = LVCMOS18;
NET "btn[1]" clock_dedicated_route = false;
NET "btn[2]" LOC=V19  |   IOSTANDARD = LVCMOS18;
NET "btn[2]" clock_dedicated_route = false;
NET "btn[3]" LOC=V18  |   IOSTANDARD = LVCMOS18;
NET "btn[3]" clock_dedicated_route = false;
NET "BTNX4" LOC=W16   |   IOSTANDARD = LVCMOS18;
```

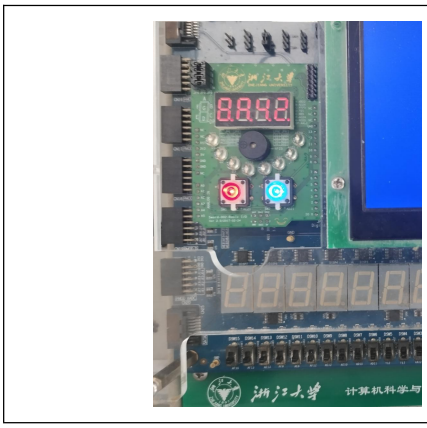
9、生成 bit 文件并下载到 sword 板上实验

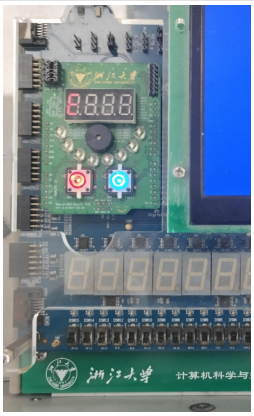
二、实验结果与分析

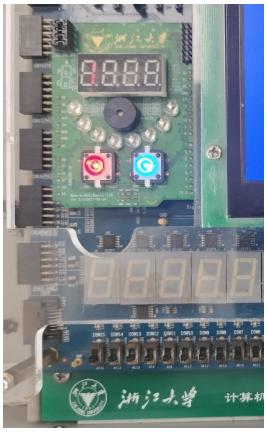
仿真实验



上板验证





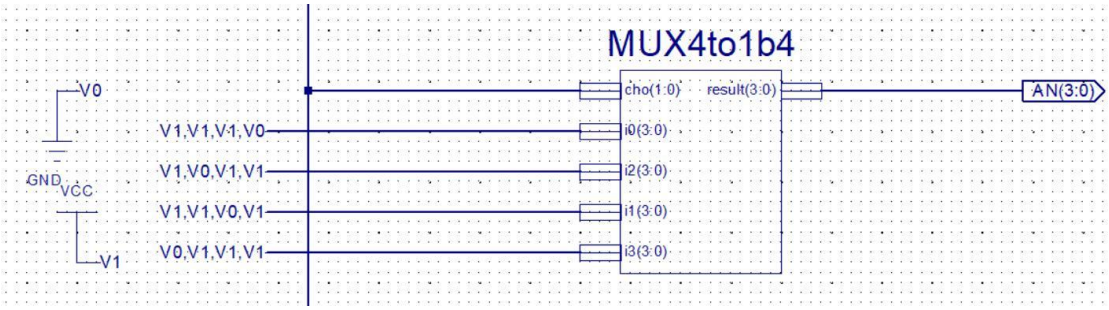


三、讨论、心得

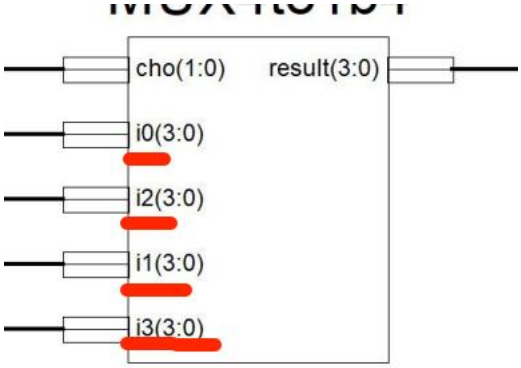
这次实验总体是利用计时器的刷新原理去快速选择要显示的那位七段数码管，但是如果只是按照 PPT 去照搬的话会出很多 BUG：

例如调用函数时可能会出现传入参数与函数定义架构参数类型不匹配导致调用错误（这是因为每个人在编写代码或者画图产生的函数不同，调用函数也自然需要服从该函数的要求）；

在画如下图时，显示 V1,V1,V1,V0 时只是修改了名称并没有使该名字显示，即没有点击 Visiable 选项导致图一直画不对；



另外在选择元器件接图时可能会出现接口不单调的情况，例如可能出现 I0,I2,I1,I3 这种情况。



所以每次实验应该去理解 PPT 上面的原理而不是简单的 Copy，这样才能正确上板验证。