

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机逻辑设计基础 |
| 姓 名： | 龙永奇 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 本系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3220105907 |
| 指导教师： | 董亚波 |

2023年11月10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： 多路选择器设计及应用

学生姓名： 龙永奇 专业： 计算机科学与技术 学号： 3220105907\_\_\_

同组学生姓名： 贾一多 指导老师： 董亚波\_\_\_\_\_\_\_\_

实验地点： 东4-509实验日期： 2023 年 11 月 2 日

1. 实验目的和要求
2. 掌握数据选择器的工作原理和逻辑功能
3. 掌握数据选择器的使用方法
4. 掌握4位数码管扫描显示方法
5. 4位数码管显示应用—记分板设计

二、实验内容和原理

内容：

1. 数据选择器设计
2. 记分板设计

原理：

1. **4选1多路选择器：MUX4to1**
2. 根据事件简化真值表：

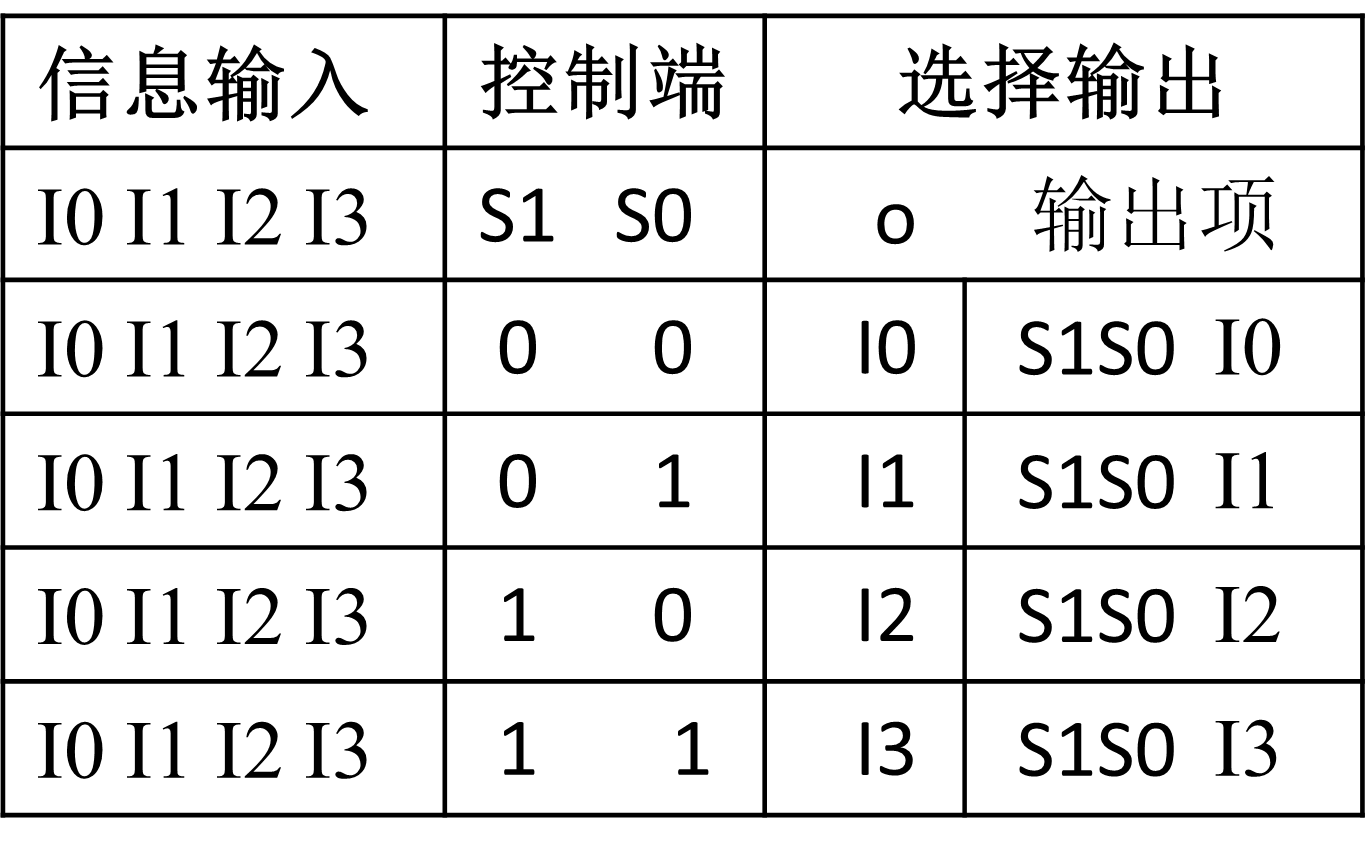


图2.1 真值表

1. 输出是控制信号全部最小项与或结构

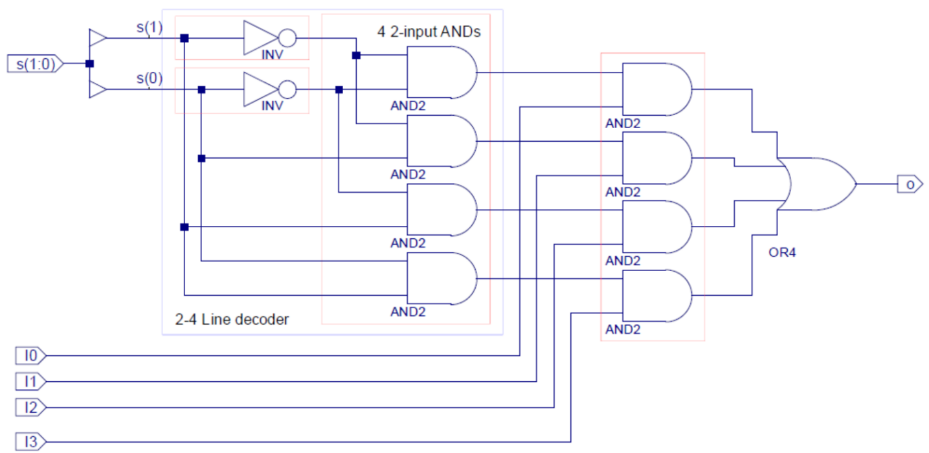


图2.2 MUX4to1

1. 控制结构不变，每路输入向量化

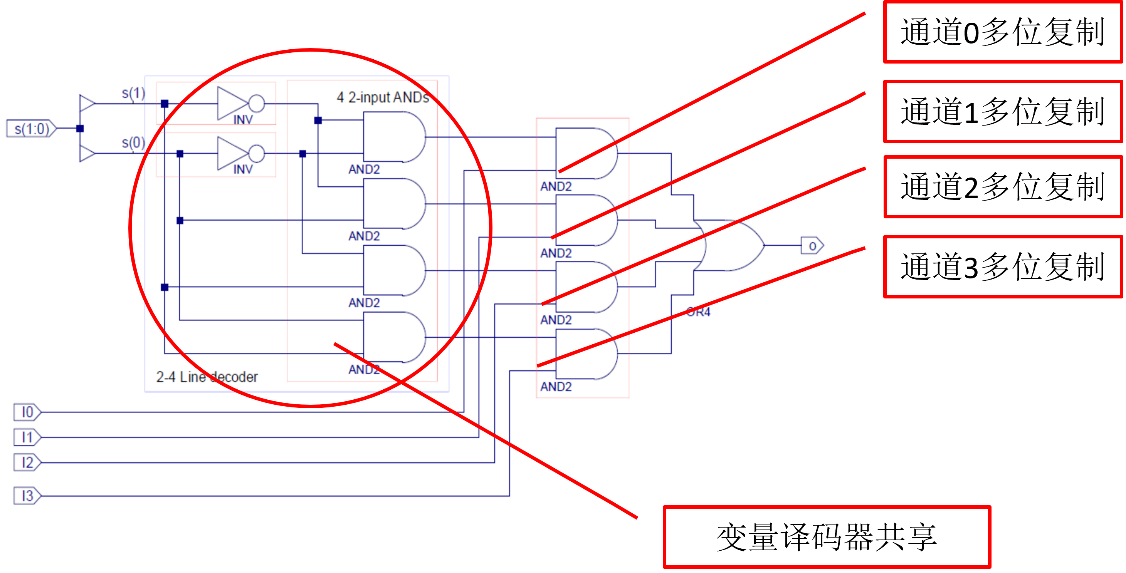


图2.3 多路选择器位扩展

1. 4位四选一扩展：MUX4to1b4

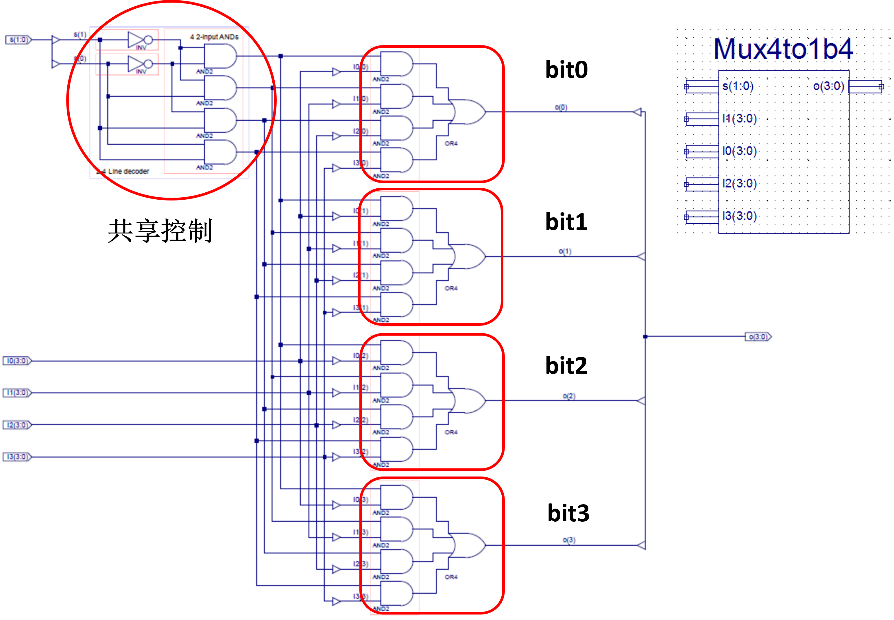


图2.4 4位四选一扩展

1. **计分板设计**

* **实现4位7段数码管动态扫描显示**

1. 扫描信号来自时钟计数分频器：**时序转化为组合电路**
2. 由板载时钟clk(100MHz)作为计数器时钟，分频后的高两位信号（clk\_div[18:17]）作为扫描控制信号Scan[1:0]，其数据为从0、1、2、3、0、……，输入2-4译码器产生数码管位选信号，控制哪个数码管显示（位选择），同时输入4选1多路复用器选择需要显示哪个数据（段码选择）
3. 计数器的分频系数要适当，几ms切换一次，眼睛舒适即可每个7 段码对应一个显示译码电路

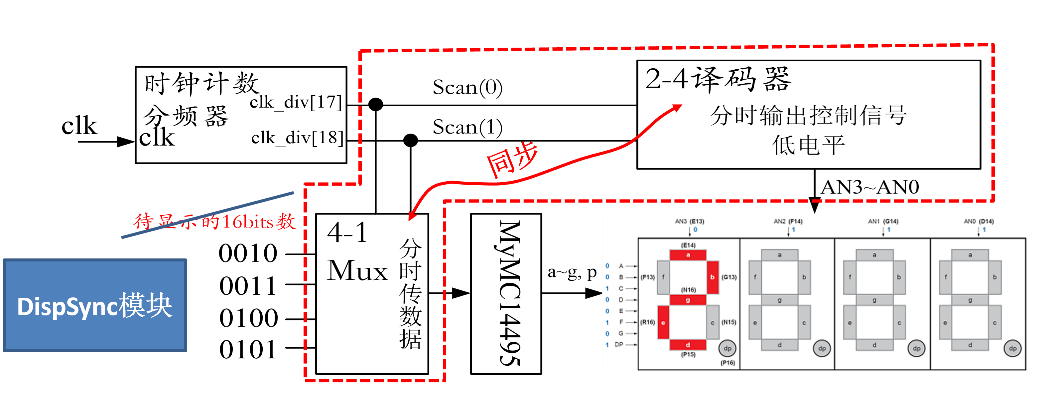


图2.5 4位7段数码管动态扫描显示

1. 对应波形图如下

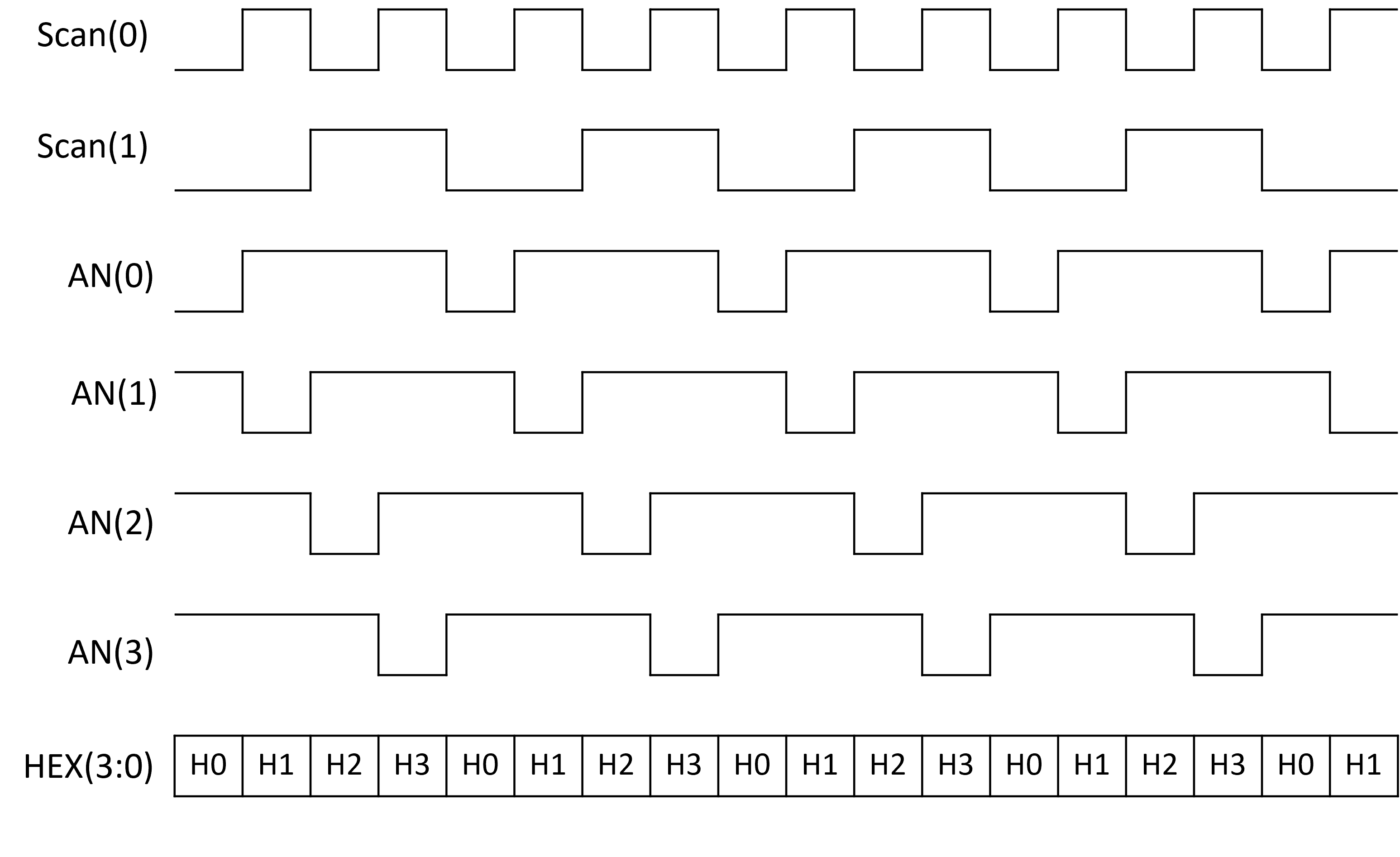


图2.6 波形

* **位扫描控制**

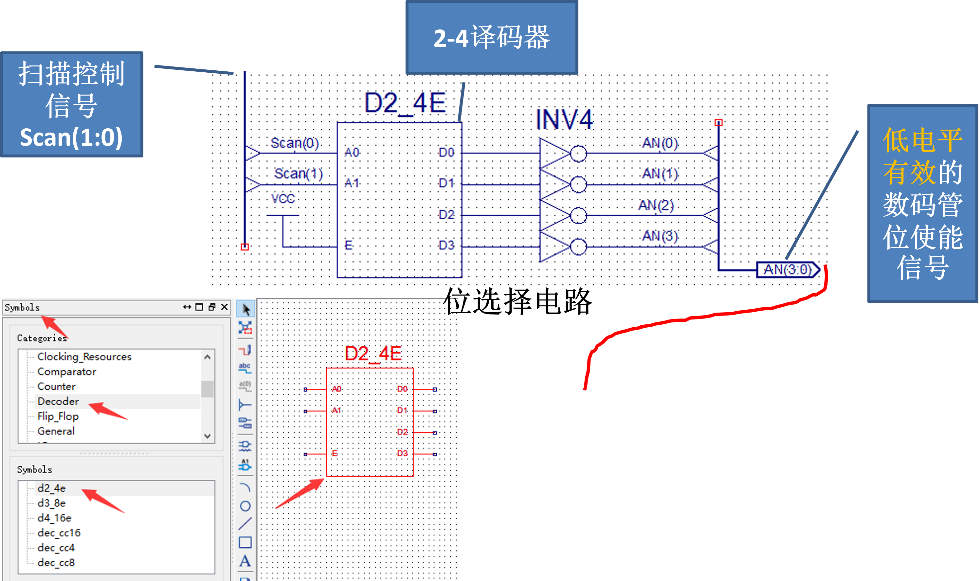


图2.7 位扫描控制

* **段码选择**

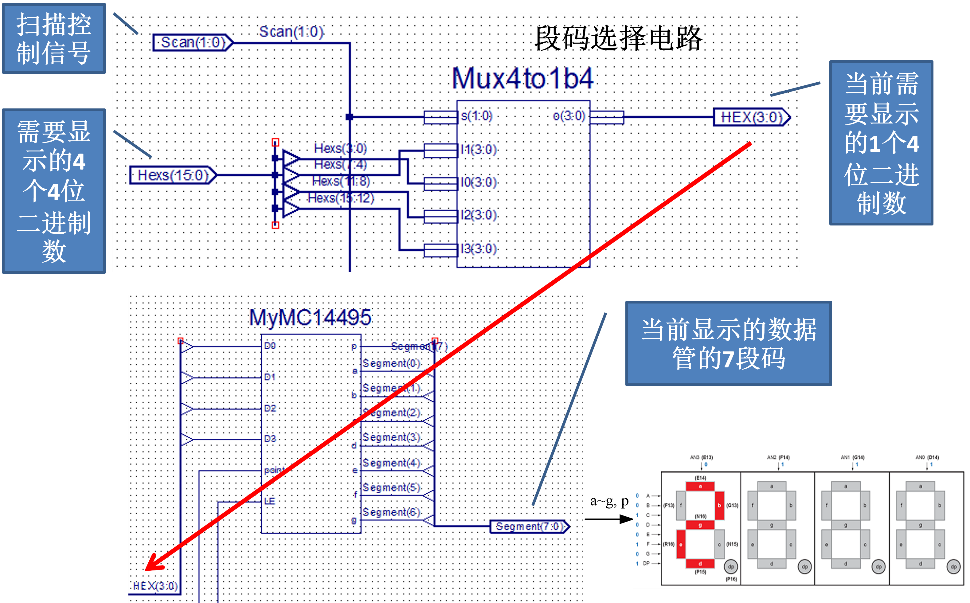
****

图2.8 段码选择

* **小数点与消隐选择**

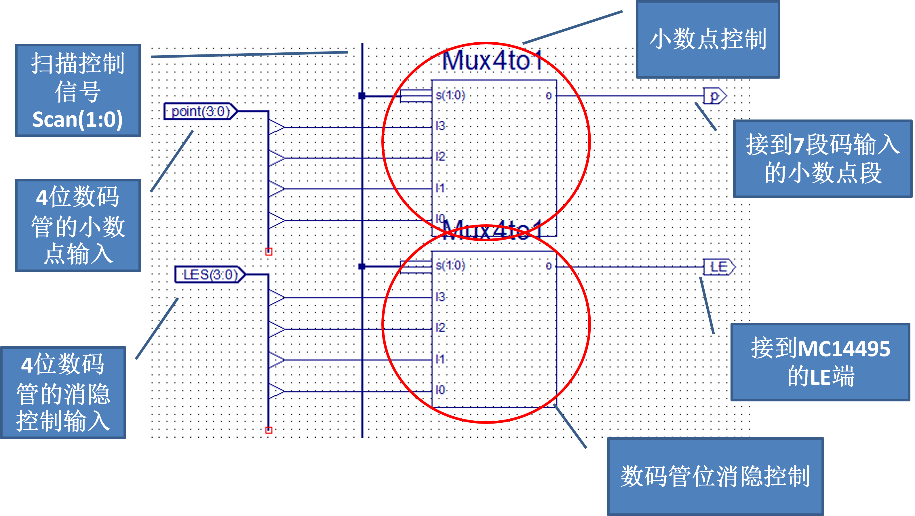
****

图2.9 小数点与消隐选择

1. **DisplaySync模块设计**
2. 用原理图形式设计

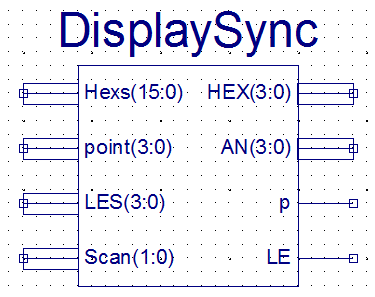


图2.10 Display模块

模块名：DisplaySync.sch

用原理图设计

制作逻辑符号并修改：DisplaySync.sym

输入：

Hexs(15:0)：需要显示的4个4位二进制数

point(3:0)：每位数码管的小数点

LES(3:0)：每位数码管是否需要消隐

Scan(1:0)：扫描控制信号

输出：

HEX(3:0)：当前要显示的4位二进制数

AN(3:0)：4位数码管的位选择信号（低电平有效）

P、LE：小数点和消隐控制

1. 用Case语句实现，代码如下：

module dispsync(input   [15:0] Hexs, //端口变量说明与定义合并

         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   //信号变化触发 (组合电路不用时钟触发)

    case (Scan)

            2‘b00 : begin Hex <= Hexs[3:0];     AN <= 4’b 1110; …   //同步输出

            2'b01 : begin Hex <= Hexs[7:4];     AN <= 4’b 1101; …   //同步输出

            2'b10 : begin Hex <= Hexs[11:8];   AN <= 4’b 1011; …    //同步输出

            2'b11 : begin Hex <= Hexs[15:12]; AN <= 4’b 0111; … //同步输出

    endcase

   end

endmodule

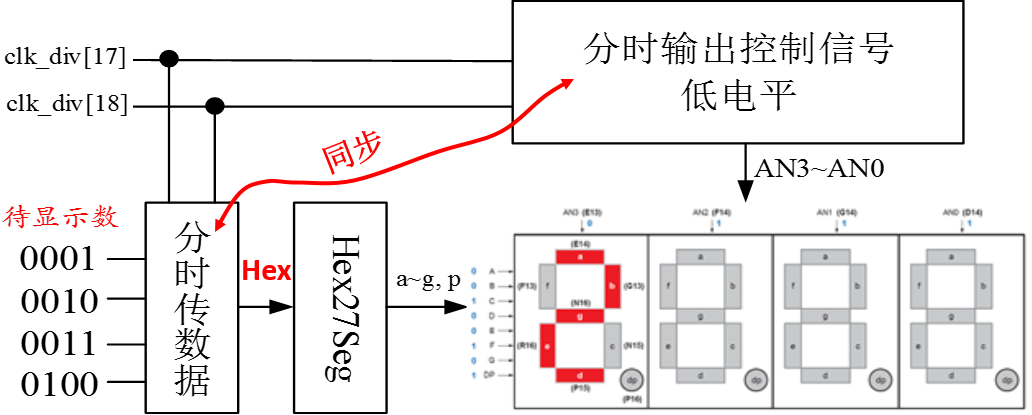


图2.11 Display模块

1. **DispNum模块内部结构**

输入4路4位信号，并根据选择信号输出，控制七段数码管显像

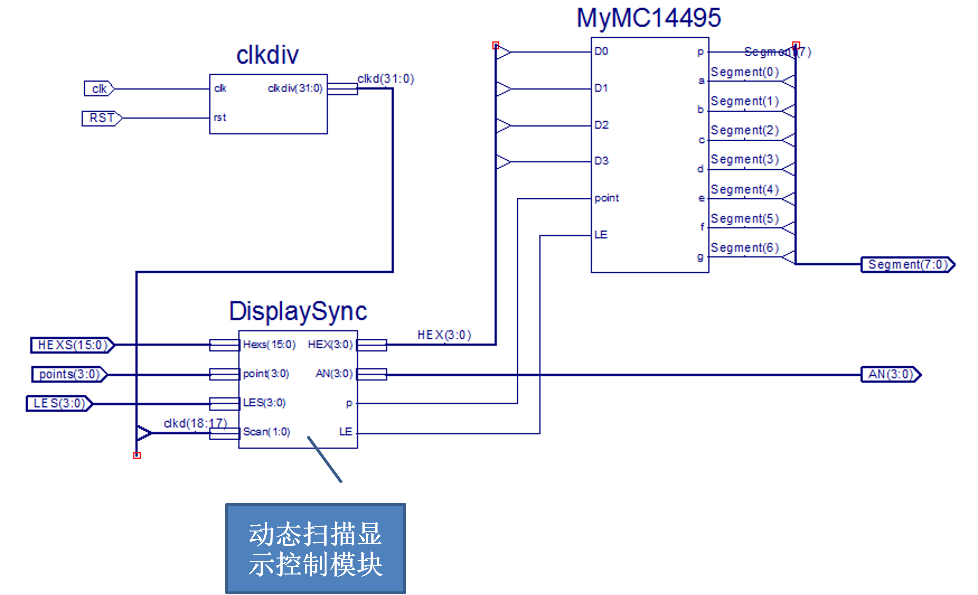


图 2.12 DispNum模块

1. **辅助模块：时钟计数分频器**



图2.13 时钟计数分频器

1. 设计32位时钟计数分频器

模块名：clkdiv.v

用Verilog HDL设计

制作逻辑符号并修改：clkdiv.sym

输入：

clk：实验板主时钟

rst：复位信号

输出：clkdiv(31:0)：分频时钟输出

1. 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

延时较高，要求不高的时钟也可以用

本实验中用clkdiv(18:17)作为扫描控制信号，控制4位数码管的动态扫描，每一位显示切换时间为  
217 / 100M = 1.3ms

module clkdiv(input   clk,      //端口变量说明与定义合并

              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

1. **设计CreateNumber按键数据输入模块**

使用行为描述设计：四个按键各按一下，4个4位2进制数分别加1

`timescale 1ns / 1ps

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; //display"AbCd"

    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

1. **计分板操作方法与应用设计**
2. 用BTNX4Y3～BTNX4Y0这4个按钮，每个按钮按下一次，对应的数码管的值加1
3. 用SW0～SW3这4个开关控制每个数码管的小数点
4. 用SW4～SW7这4个开关控制每个数码管的消隐

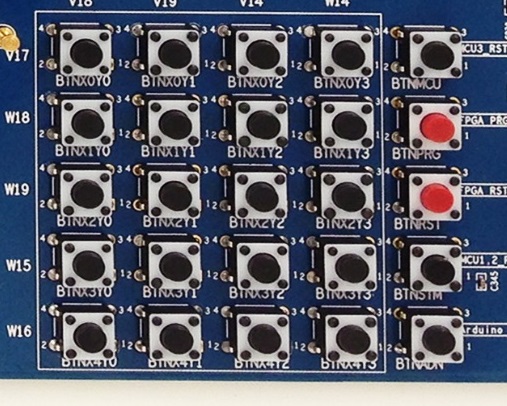
****

图2.14 开关控制

1. 新建工程

工程名称用ScoreBoard。

Top Level Source Type用HDL

设计动态扫描同步输出模块、通用计数分频模块

1. 输出输出引脚功能

输入

时钟：clk

使能控制：sw[7:4]

小数点输入：sw[3:0]

按键输入数字：BTNX4Y0~BTNX4Y3为btn[3:0]

输出

七段数码管段码输出线：segment[7:0}，包括a~g，p

七段数码管位选择线：an[3:0]

按键使能控制线：BTNX4，需要输出为0

根据设计修改UCF

1. **设计顶层模块**

新建源文件top，在右键菜单里设为“Top Module”，代码如下：

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 wire BTNX4

);

    wire [15:0] num;

    CreateNumber c0(btn,num);

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

    assign BTNX4 = 1'b0;    //Enable button inputs

endmodule

三、实验过程和数据记录

**3.1数据选择器设计**

1. **设计元件Mux4to1**
2. 在ISE中点击File选项卡，点击New Project，工程名为Mux4to1b4
3. 在Sources窗口中右键选择New Sources新建源文件向导中选择源文件类型为Schematic，输入文件名Mux4to1，勾选Add to Project
4. 使用Symbols和Schematic Editor输入原理图如下：

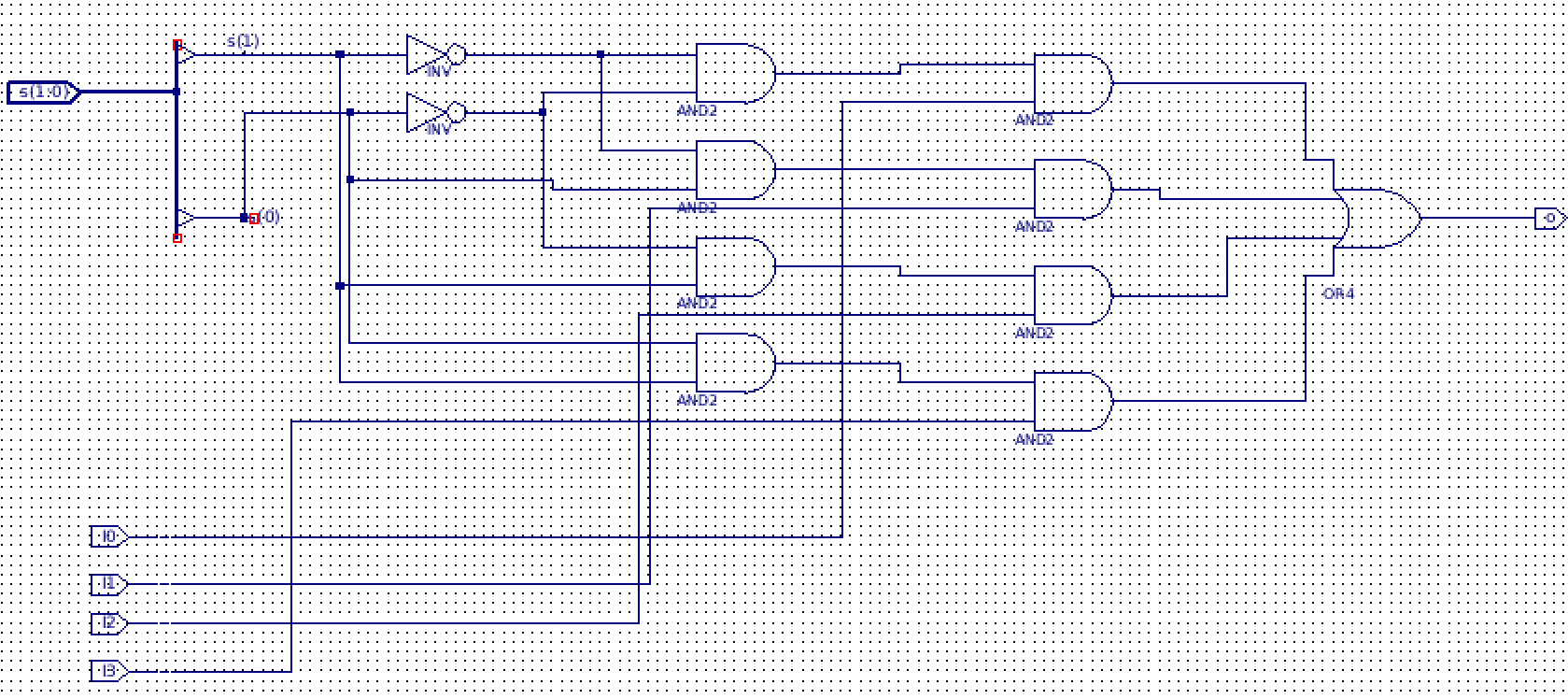


图3.1 Mux4to1

1. 点击Check Design Rules检查错误并查看电路的硬件描述代码
2. 点击View HDL Functional Model和Create Schematic Symbol，生成逻辑符号图.sym

硬件描述代码如下：

`timescale 1ns / 1ps

module Mux4to1\_sch(I0,

                   I1,

                   I2,

                   I3,

                   s,

                   o);

    input I0;

    input I1;

    input I2;

    input I3;

    input [1:0] s;

   output o;

   wire XLXN\_7;

   wire XLXN\_8;

   wire XLXN\_24;

   wire XLXN\_25;

   wire XLXN\_26;

   wire XLXN\_27;

   wire XLXN\_29;

   wire XLXN\_30;

   wire XLXN\_31;

   wire XLXN\_32;

   AND2  XLXI\_1 (.I0(XLXN\_8),

                .I1(XLXN\_7),

                .O(XLXN\_24));

   AND2  XLXI\_2 (.I0(s[0]),

                .I1(XLXN\_7),

                .O(XLXN\_25));

   AND2  XLXI\_3 (.I0(s[1]),

                .I1(XLXN\_8),

                .O(XLXN\_26));

   AND2  XLXI\_4 (.I0(s[1]),

                .I1(s[0]),

                .O(XLXN\_27));

   INV  XLXI\_5 (.I(s[1]),

               .O(XLXN\_7));

   INV  XLXI\_6 (.I(s[0]),

               .O(XLXN\_8));

   AND2  XLXI\_7 (.I0(I0),

                .I1(XLXN\_24),

                .O(XLXN\_29));

   AND2  XLXI\_8 (.I0(I1),

                .I1(XLXN\_25),

                .O(XLXN\_30));

   AND2  XLXI\_9 (.I0(I2),

                .I1(XLXN\_26),

                .O(XLXN\_31));

   AND2  XLXI\_10 (.I0(I3),

                 .I1(XLXN\_27),

                 .O(XLXN\_32));

   OR4  XLXI\_11 (.I0(XLXN\_32),

                .I1(XLXN\_31),

                .I2(XLXN\_30),

                .I3(XLXN\_29),

                .O(o));

endmodule

1. **对Mux4to1进行仿真**
2. 新建Verilog Test Fixture文件，名称为Mux4to1\_sim

仿真激励代码如下：

`timescale 1ns / 1ps

module Mux4to1\_sch\_Mux4to1\_sch\_sch\_tb();

   reg [1:0] s;

   reg I0;

   reg I1;

   reg I2;

   reg I3;

   wire o;

   Mux4to1\_sch UUT (

        .s(s),

        .I0(I0),

        .I1(I1),

        .I2(I2),

        .I3(I3),

        .o(o)

   );

initial begin

        s = 0;

        I0 = 1;

        I1 = 0;

        I2 = 0;

        I3 = 0;

        #50;

        I0 = 1;

        I1 = 0;

        I2 = 1;

        I3 = 0;

        #50;

        s = 1;

        I0 = 1;

        I1 = 0;

        I2 = 1;

        I3 = 1;

        #50;

        I0 = 1;

        I1 = 1;

        I2 = 1;

        I3 = 1;

        #50;

        s = 2;

        I0 = 0;

        I1 = 1;

        I2 = 1;

        I3 = 0;

        #50;

        I0 = 1;

        I1 = 0;

        I2 = 0;

        I3 = 0;

        #50;

        s = 3;

        I0 = 1;

        I1 = 1;

        I2 = 0;

        I3 = 1;

        #50;

        I0 = 1;

        I1 = 0;

        I2 = 0;

        I3 = 1;

        #50;

        s = 0;

        I0 = 0;

        I1 = 0;

        I2 = 0;

        I3 = 0;

        end

endmodule

1. 仿真结果如下：

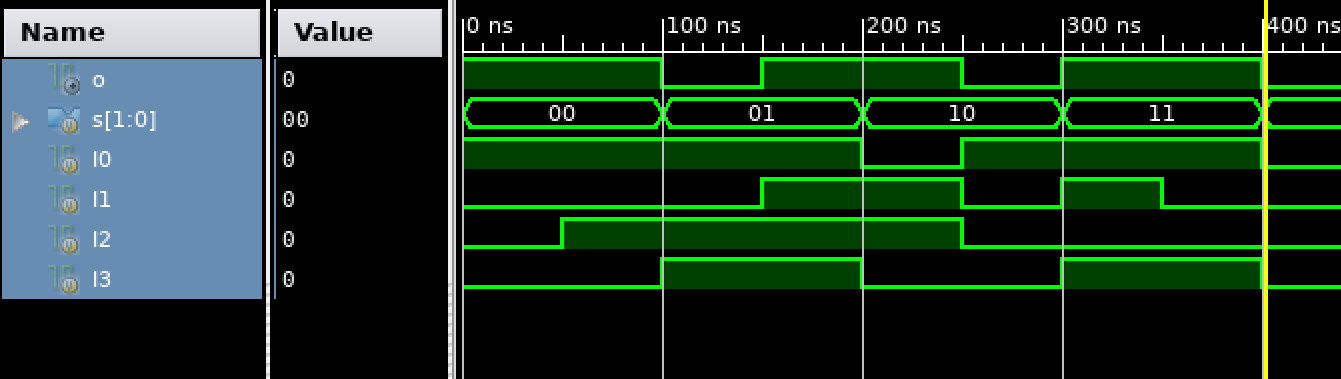


图3.2 仿真波形

1. **设计元件Mux4to1b4**
2. 在New Source中创建文件名为Mux4to1b4，勾选Add to Project
3. 之前绘制的Mux4to1仅能实现一位信号的选择，但七位数码管可以显示十六进制数的一位，因此需要四位选择器满足0-F的数值显示
4. 绘制原理图如下：

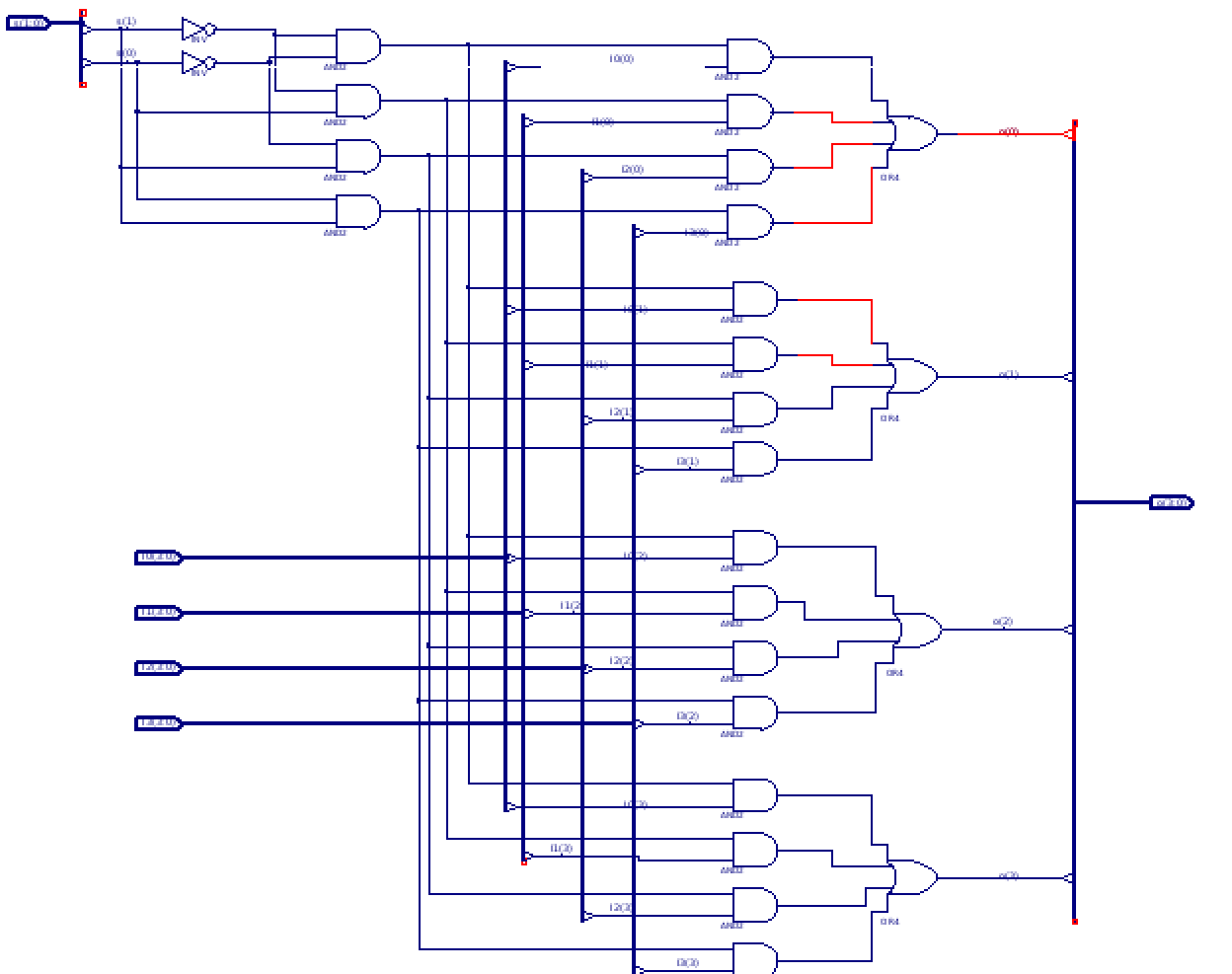


图3.3 Mux4to1b4

1. 点击Check Design Rules检查错误并查看电路的硬件描述代码
2. 点击View HDL Functional Model和Create Schematic Symbol，生成逻辑符号图.sym
3. **对Mux4to1b4进行仿真**
4. 新建Verilog Test Fixture文件，名称为Mux4to1b4\_sim
5. 仿真激励代码如下：

`timescale 1ns / 1ps

module Mux4to1b4\_Mux4to1b4\_sch\_tb();

// Inputs

   reg [1:0] s;

   reg [3:0] I1;

   reg [3:0] I2;

   reg [3:0] I3;

   reg [3:0] I0;

// Output

   wire [3:0] o;

// Bidirs

// Instantiate the UUT

   Mux4to1b4 UUT (

        .s(s),

        .I1(I1),

        .I2(I2),

        .I3(I3),

        .I0(I0),

        .o(o)

   );

// Initialize Inputs

   initial begin

        s = 0;

        I1 = 0;

        I2 = 0;

        I3 = 0;

        I0 = 0;

       #50;

        s = 2'b01;

       #50;

        s = 2'b10;

       #50;

        s = 2'b11;

       #50;

        s = 0;

    end

    initial begin

         I0 = 4'b0001;

         I1 = 4'b0010;

         I2 = 4'b0100;

         I3 = 4'b1000;

    end

endmodule

1. 仿真结果如下：

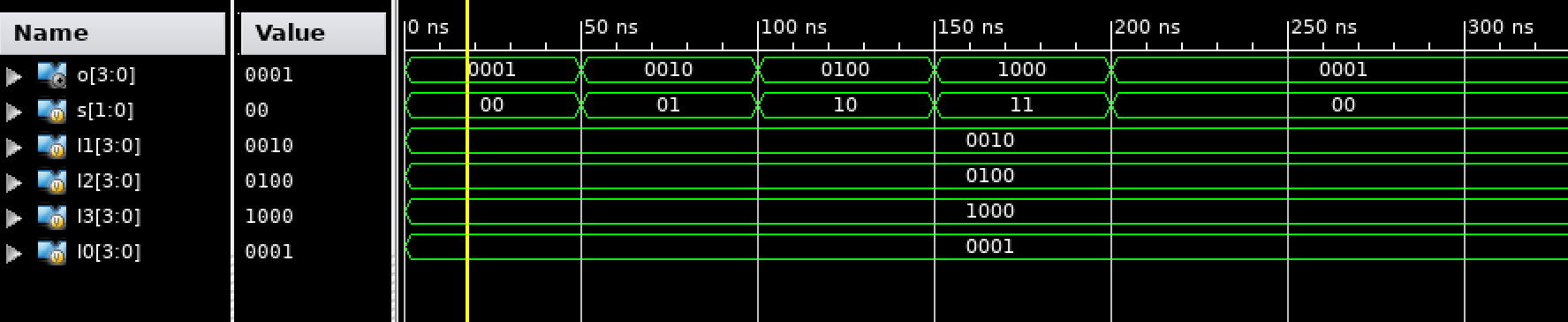


图3.4 仿真波形

**3.2 计分板应用设计**

1. **CreateNumber模块**
2. 新建工程ScoreBoard
3. 新建Verilog源代码文件CreateNumber,代码如下：

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

1. 点击Check Syntax，检查语法错误
2. **clkdiv计时器设计**
3. 新建Verilog代码文件clkdiv，输入以下代码：

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

1. 点击Check Syntax，编译该模块，检查语法错误
2. **DisplaySync模块设计**
3. 新建Schematic代码文件DisplaySync
4. 到导入Mux4to1和Mux4to1b4元器件的.sch和.sym文件

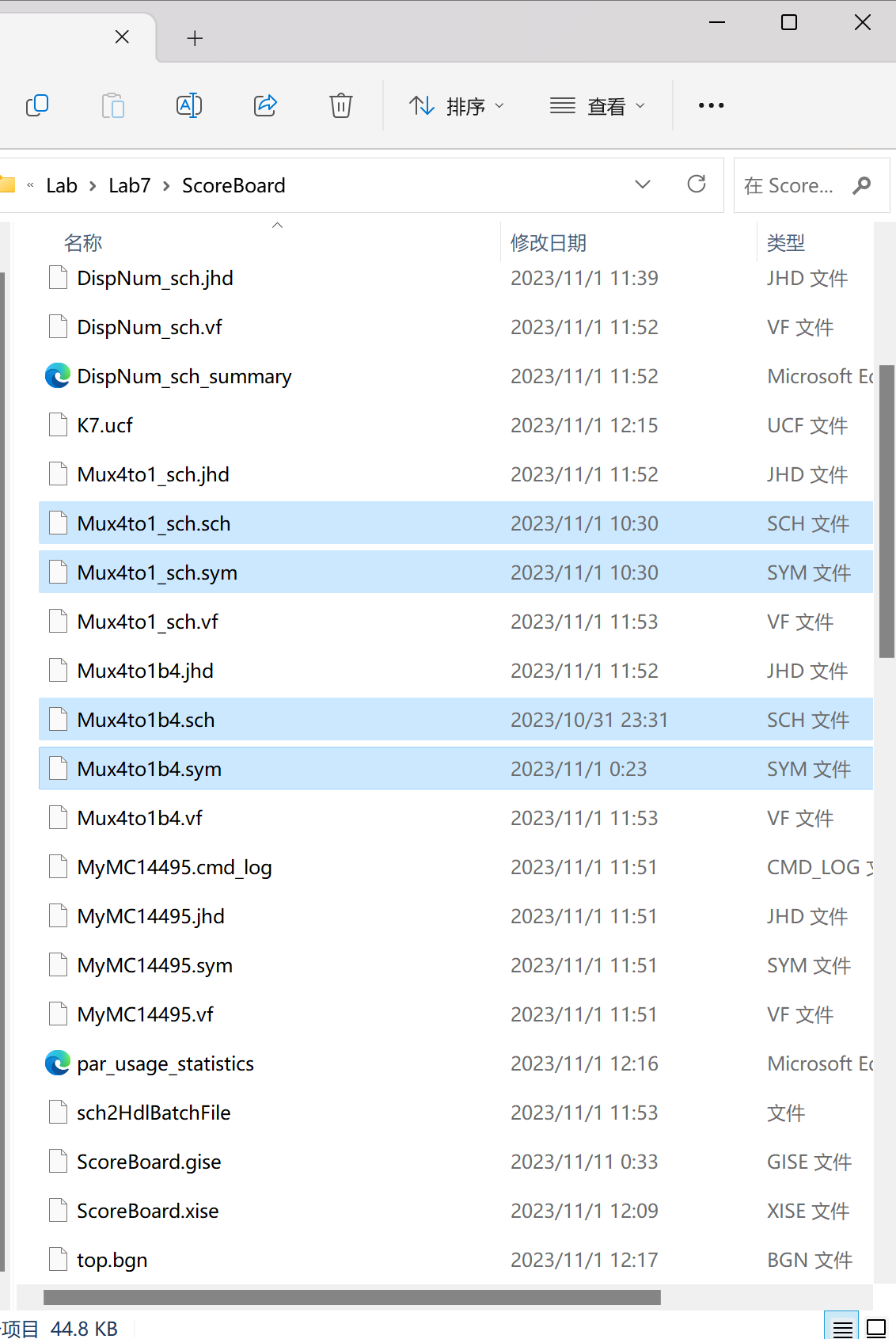


图3.5 导入文件

1. 绘制原理图：

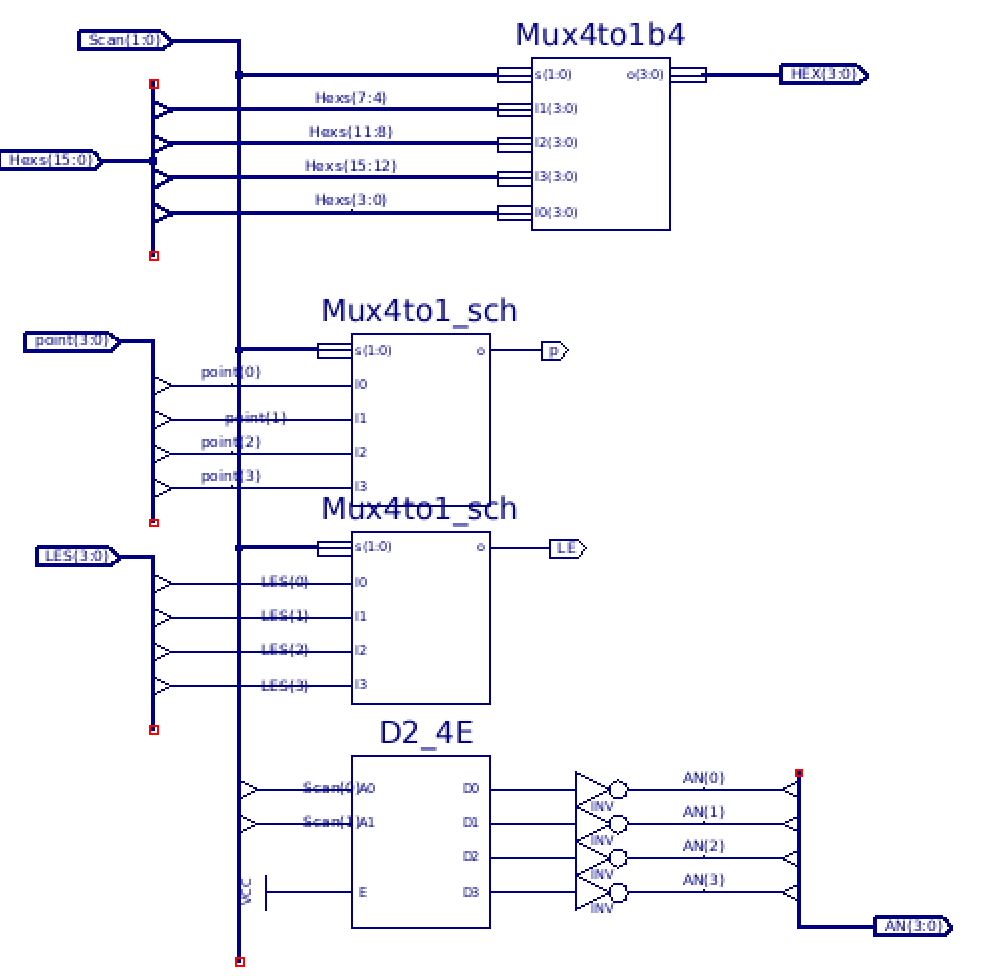


图3.6 DisplaySync原理图

1. 编译并生成对应元器件DisplaySync
2. **DisplayNumber模块设计**
3. 新建Schematic源文件DispNum
4. 将实验6绘制的MyMC14495文件导入
5. 绘制原理图如下：

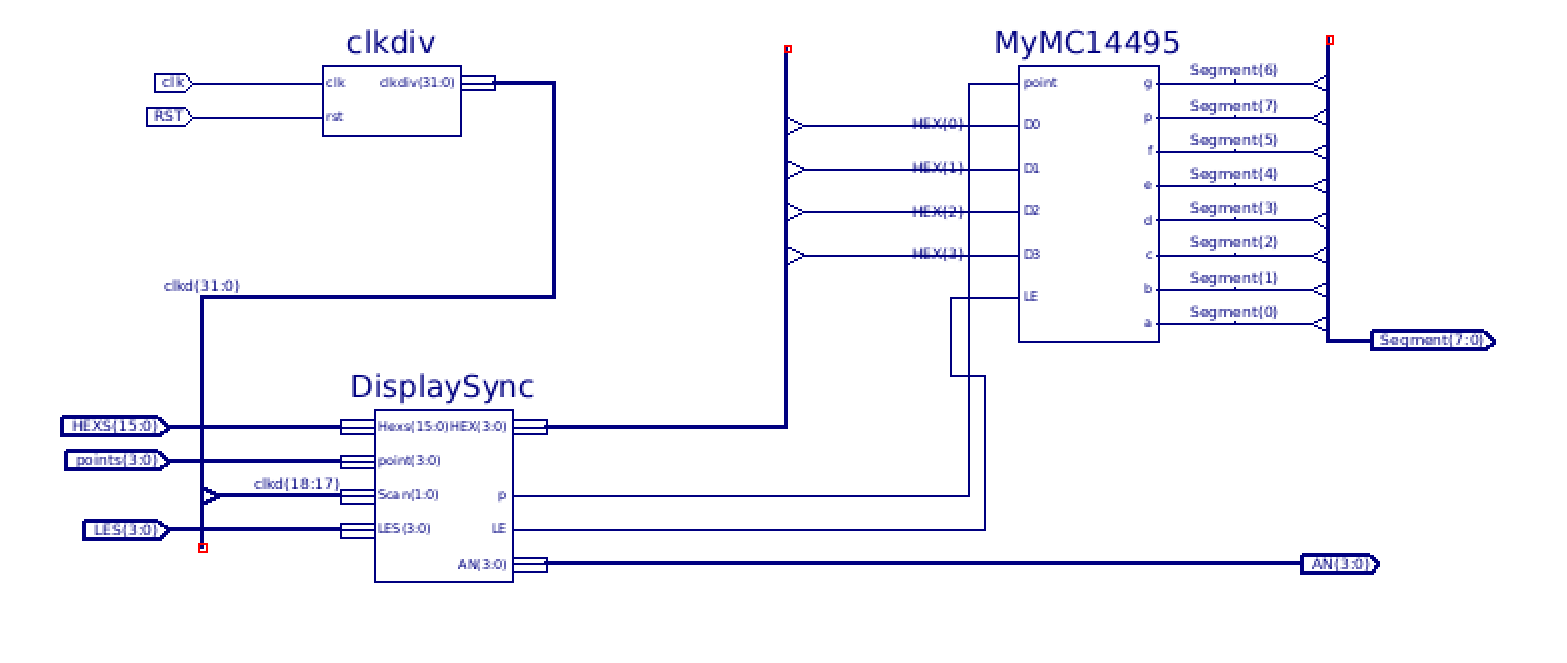


图3.7 DispNum原理图

1. 编译生成对应的Verilog代码
2. **Top模块设计**
3. 新建Verilog源代码文件top
4. 由于只有最顶层的Module才能进行引脚约束操作，因此需要右键将该文件设置为Top Module

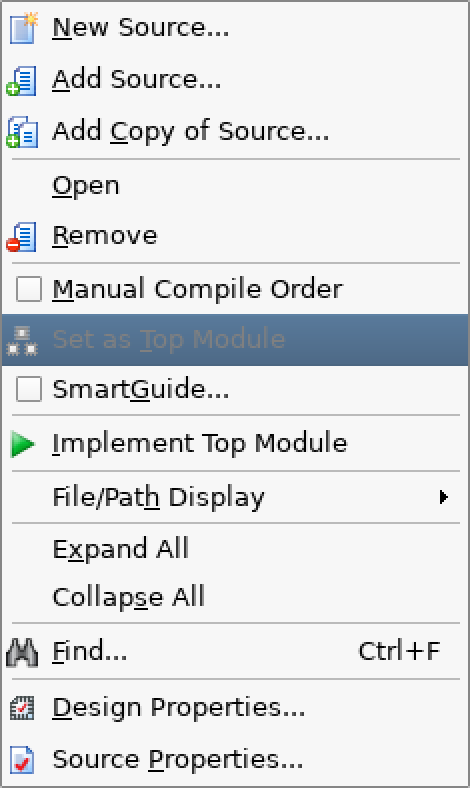


图3.8 top module

1. 输入Verilog代码如下：

`timescale 1ns / 1ps

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 wire BTNX4

    );

    wire [15:0] num;

    CreateNumber c0(btn,num);

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

    assign BTNX4 = 1'b0;

endmodule

1. **建立引脚约束文件**
2. 七段数码管引脚约束：

NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;//a

NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;//b

NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;//c

NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;//d

NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;//e

NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;//f

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;

1. 数码管开关引脚约束：

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;

1. 时钟和各按钮引脚约束：7

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;

1. **下载到SWORD板上进行验证**
2. 点击Generate Programming Files, 通过后点击Configure Target Device -> Manage Configuration Project。
3. 根据电路原理图和引脚约束文件，实验结果如下：

* **控制小数点开关**

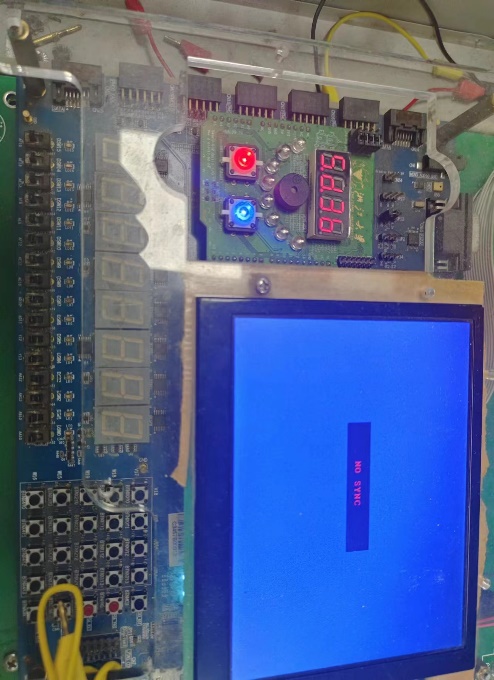
 

图3.9 图3.10

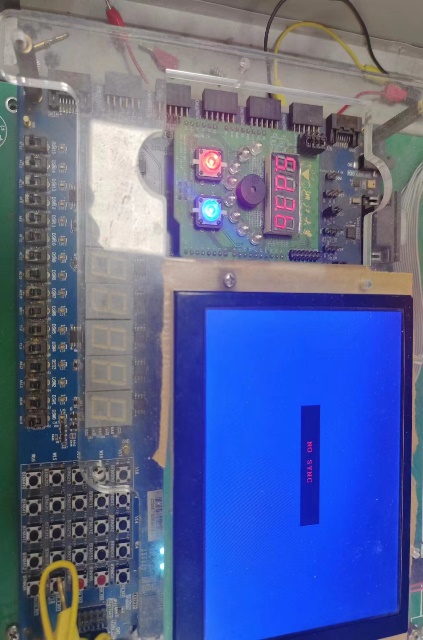
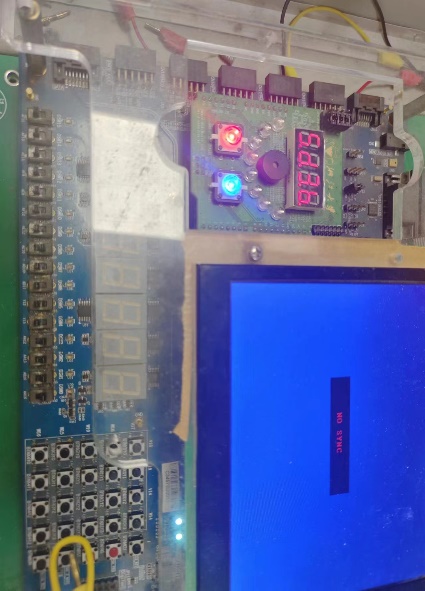
 

图3.11 图3.12

* **控制数码管开关**

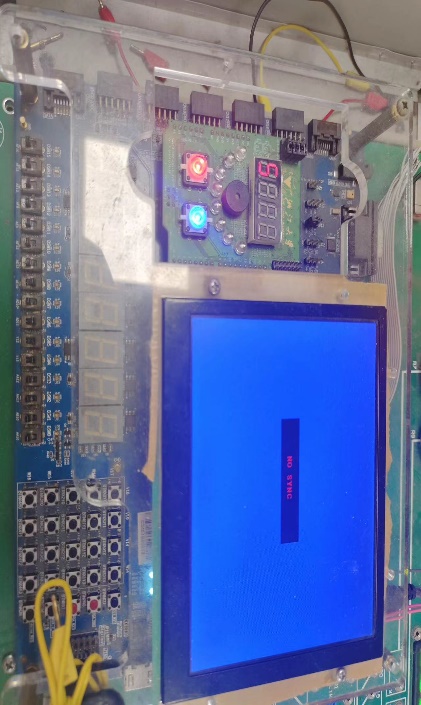
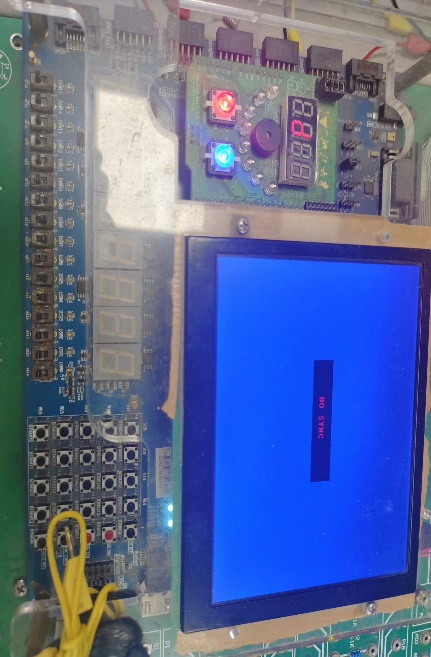
** **

图3.13 图3.14



图3.15 图3.16

* **用按钮使对应数码管数字增加**

**** **** **** **** 图3.17 图3.18 图3.19 图3.20

1. 实验结果分析

本次实验从七位数码管控制出发，通过变量编码器的应用，依次完成硬件电路图设计、仿真模拟、开发板测试三个步骤

1. **电路图绘制与硬件描述代码**

本次实验中所使用的电路图来自课程PPT，通过电路图生成硬件描述代码后，将代码与图中各组件依次对应，使用了多个逻辑门和反相器（INV）来实现开关信号和LED输出之间的逻辑关系：

1. INV用于对输入信号取反
2. AND3 用于实现三个输入信号的与逻辑运算
3. OR4用于实现四个输入信号的或逻辑运算。
4. **仿真模拟**
5. 仿真激励输入的代码分别测试了Mux4to1和Mux4to1b，仿真波形图中对应输出结果与预期相同：

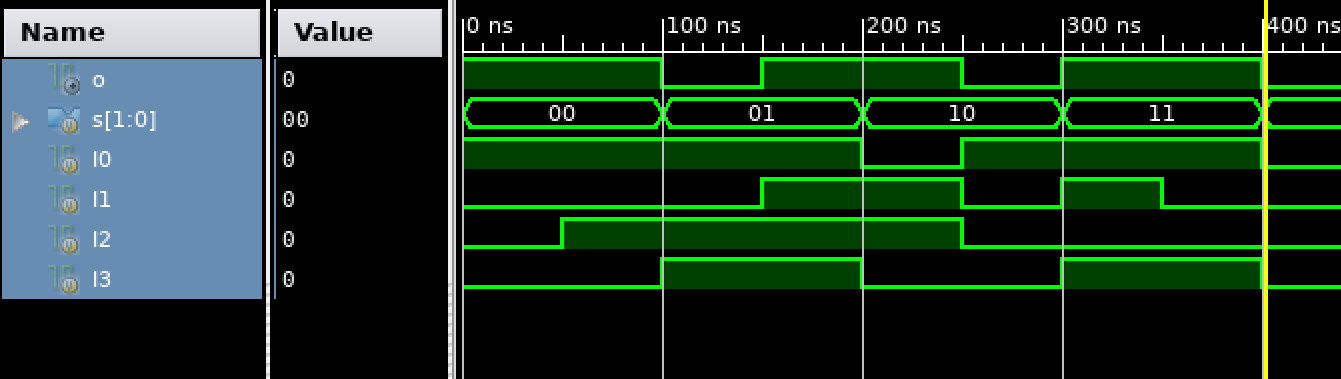


图4.1 仿真波形图1

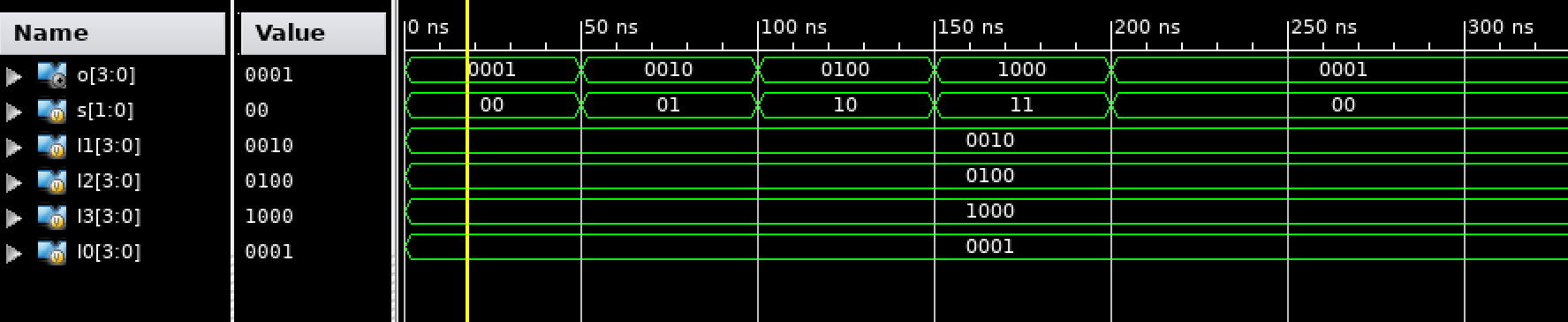


图4.2 仿真波形图2

1. **开发板验证**

在开发板上，使用按钮增加数字时会出现点击一下按钮，但数字改变多次的情况，在老师解释后了解到是因为机械抖动导致的，会在下一节实验课上进行改进。

五、讨论与心得

Lab7的实验原理图以及流程相对复杂，本次实验的难点主要在于绘制复杂的电路图，以及将不同模块整合，如果其中一个出错则整体都会有问题，其他原理和概念都已经学过，在电路图的绘制和仿真过程中，细心和耐心是非常重要的，确保一切都按照预期进行，以获得准确的实验结果。同时感谢助教老师以及同组同学的帮助！