

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：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

一、实验目的和要求

1. 掌握数据选择器的工作原理和逻辑功能
2. 掌握数据选择器的使用方法
3. 掌握 4 位数码管扫描显示方法
4. 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

(a) 根据事件简化真值表：

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 2.1 真值表

(b) 输出是控制信号全部最小项与或结构

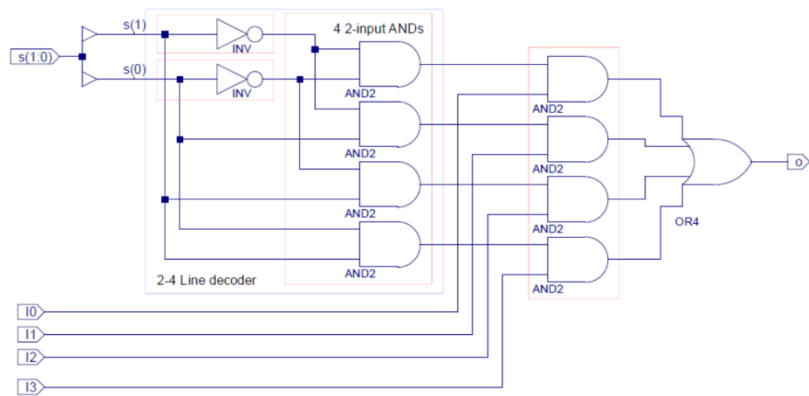


图 2.2 MUX4to1

(c) 控制结构不变，每路输入向量化

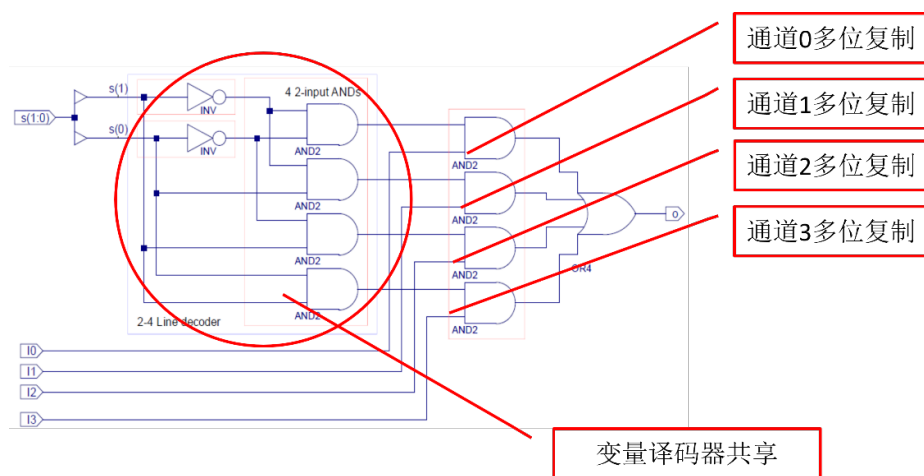


图 2.3 多路选择器位扩展

(d) 4 位四选一扩展：MUX4to1b4

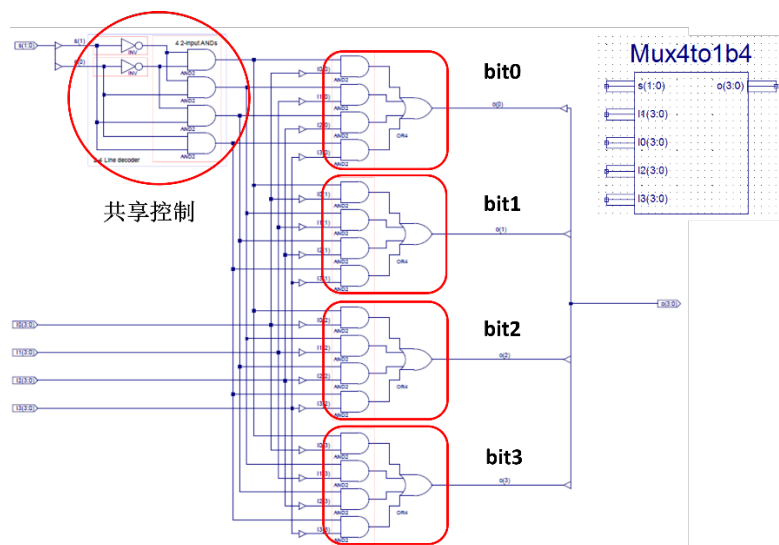


图 2.4 4 位四选一扩展

2. 计分板设计

- 实现 4 位 7 段数码管动态扫描显示

(a) 扫描信号来自时钟计数分频器：**时序转化为组合电路**

(b) 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）

(c) 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

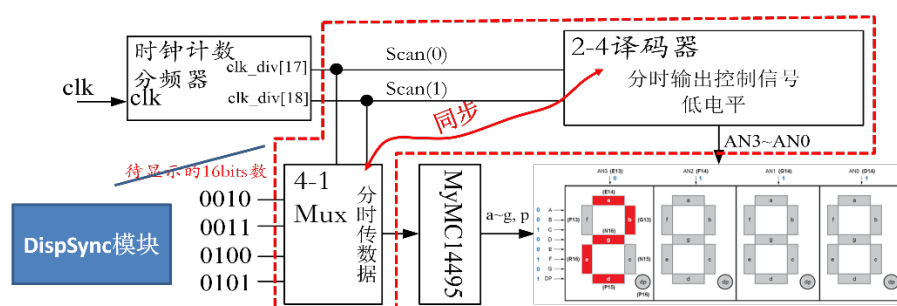


图 2.5 4 位 7 段数码管动态扫描显示

(d) 对应波形图如下

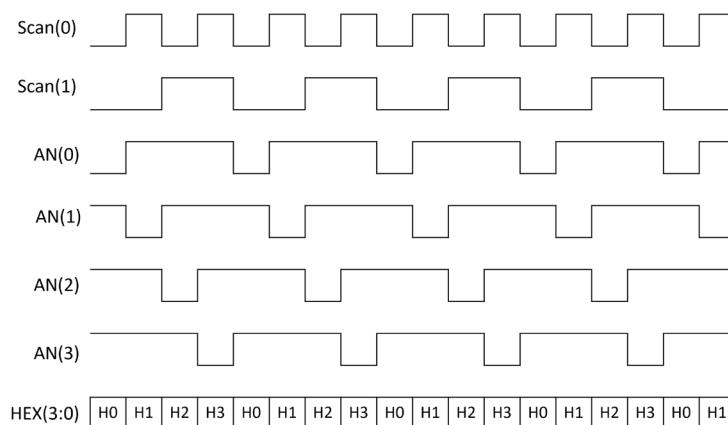


图 2.6 波形

- 位扫描控制

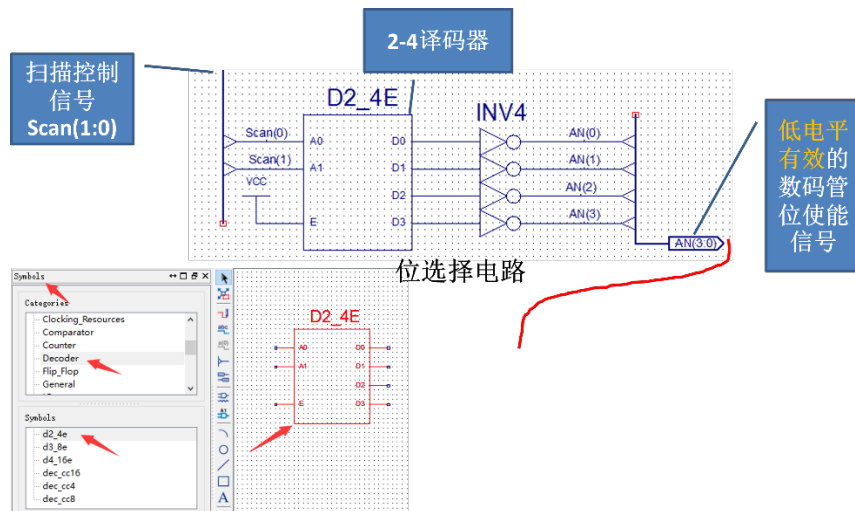


图 2.7 位扫描控制

● 段码选择

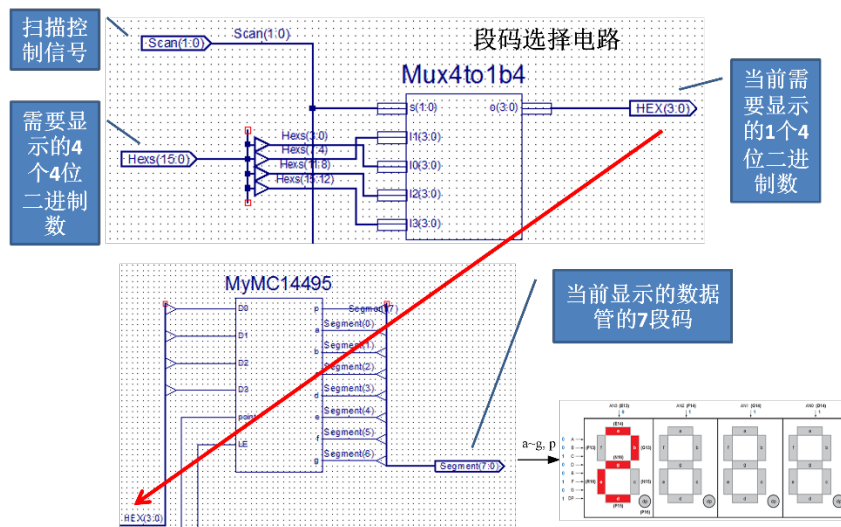


图 2.8 段码选择

● 小数点与消隐选择

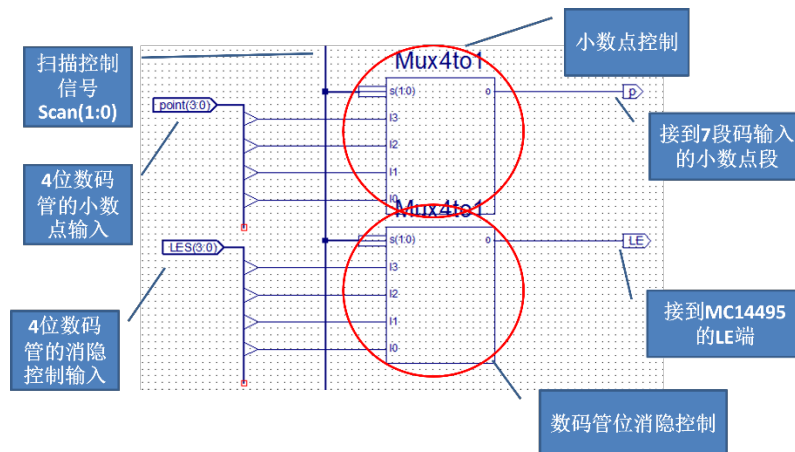


图 2.9 小数点与消隐选择

3. DisplaySync 模块设计

(a) 用原理图形式设计

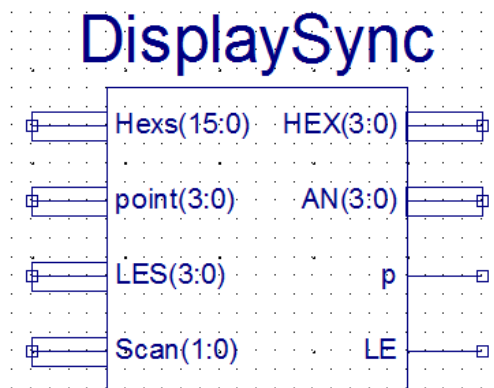


图 2.10 Display 模块

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

(b) 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

```

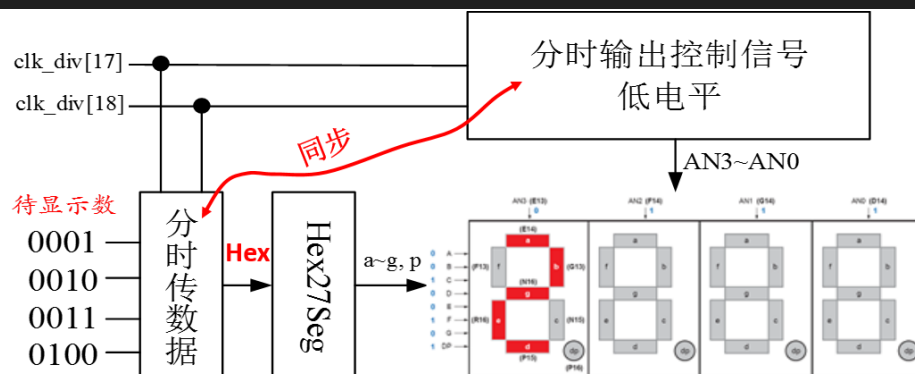


图 2.11 Display 模块

4. 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

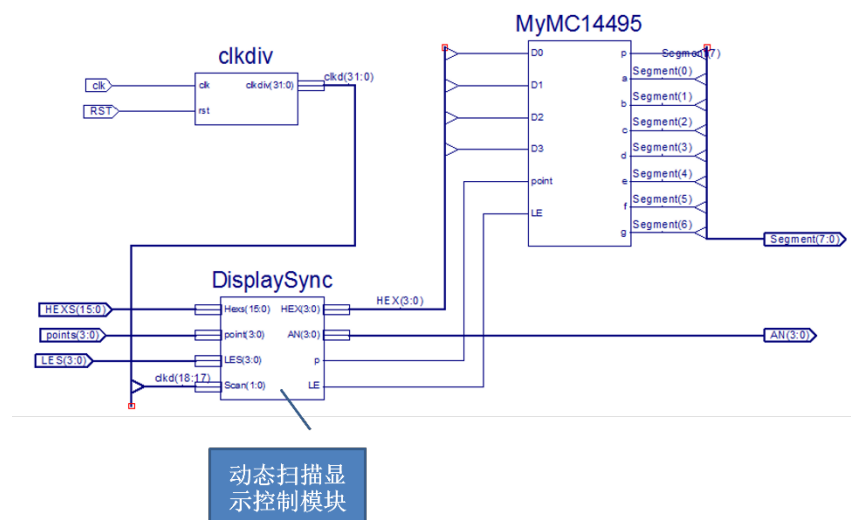


图 2.12 DispNum 模块

5. 辅助模块：时钟计数分频器



图 2.13 时钟计数分频器

(a) 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

(b) 32 位时钟计数分频器

可输出 $2 \sim 2^{32}$ 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 $2^{17} / 100M = 1.3ms$

```
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
```

6. 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

```

7. 计分板操作方法与应用设计

- (a) 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
- (b) 用 SW0~SW3 这 4 个开关控制每个数码管的小数点
- (c) 用 SW4~SW7 这 4 个开关控制每个数码管的消隐

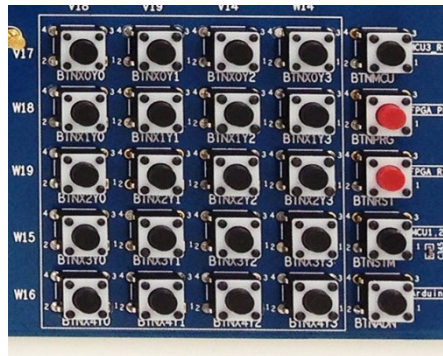


图 2.14 开关控制

- (d) 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

- (e) 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

8. 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

(a) 在 ISE 中点击 File 选项卡, 点击 New Project, 工程名为

Mux4to1b4

(b) 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

(c)使用 Symbols 和 Schematic Editor 输入原理图如下：

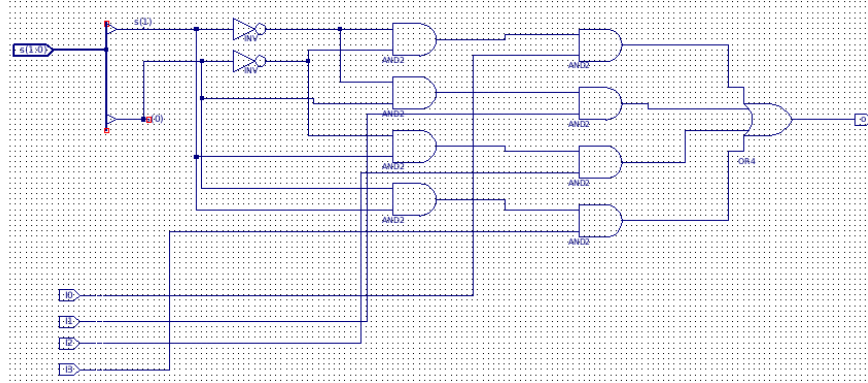


图 3.1 Mux4to1

(d)点击 Check Design Rules 检查错误并查看电路的硬件描述代码

(e)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

```

2. 对 Mux4to1 进行仿真

(a) 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_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

```

(b) 仿真结果如下：

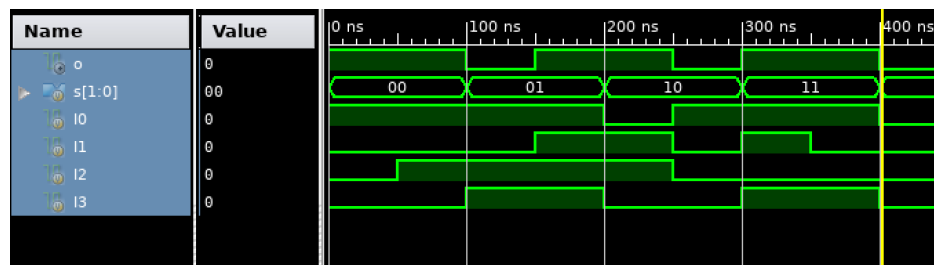


图 3.2 仿真波形

3. 设计元件 Mux4to1b4

- (a) 在 New Source 中创建文件名为 Mux4to1b4，勾选 Add to Project
- (b) 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
- (c) 绘制原理图如下：

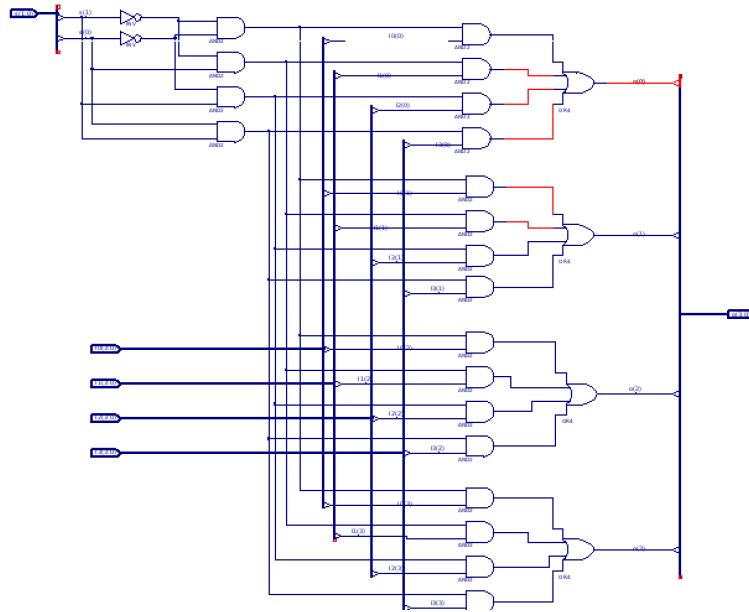


图 3.3 Mux4to1b4

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

4. 对 Mux4to1b4 进行仿真

- (a) 新建 Verilog Test Fixture 文件, 名称为 Mux4to1b4_sim
- (b) 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

```

(c) 仿真结果如下：

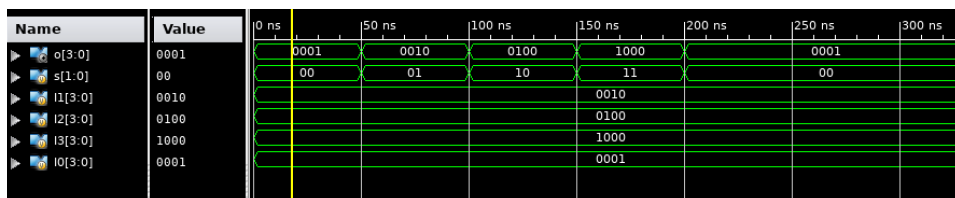


图 3.4 仿真波形

3.2 计分板应用设计

1. CreateNumber 模块

(a) 新建工程 ScoreBoard

(b) 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

```

(c) 点击 Check Syntax, 检查语法错误

2. clkdiv 计时器设计

(a) 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

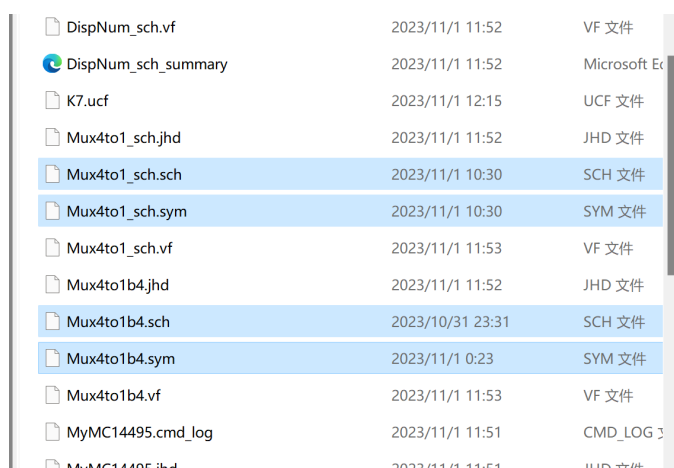
```

(b) 点击 Check Syntax, 编译该模块, 检查语法错误

3. DisplaySync 模块设计

(a) 新建 Schematic 代码文件 DisplaySync

(b) 到导入 Mux4to1 和 Mux4to1b4 元器件的 .sch 和 .sym 文件



File Name	Date/Time	File Type
DispNum_sch.vf	2023/11/1 11:52	VF 文件
DispNum_sch_summary	2023/11/1 11:52	Microsoft Excel 文件
K7.ucf	2023/11/1 12:15	UCF 文件
Mux4to1_sch.jhd	2023/11/1 11:52	JHD 文件
Mux4to1_sch.sch	2023/11/1 10:30	SCH 文件
Mux4to1_sch.sym	2023/11/1 10:30	SYM 文件
Mux4to1_sch.vf	2023/11/1 11:53	VF 文件
Mux4to1b4.jhd	2023/11/1 11:52	JHD 文件
Mux4to1b4.sch	2023/10/31 23:31	SCH 文件
Mux4to1b4.sym	2023/11/1 0:23	SYM 文件
Mux4to1b4.vf	2023/11/1 11:53	VF 文件
MyMC14495.cmd_log	2023/11/1 11:51	CMD_LOG 文件
MyMC14495.jhd	2023/11/1 11:51	JHD 文件

图 3.5 导入文件

(c) 绘制原理图:

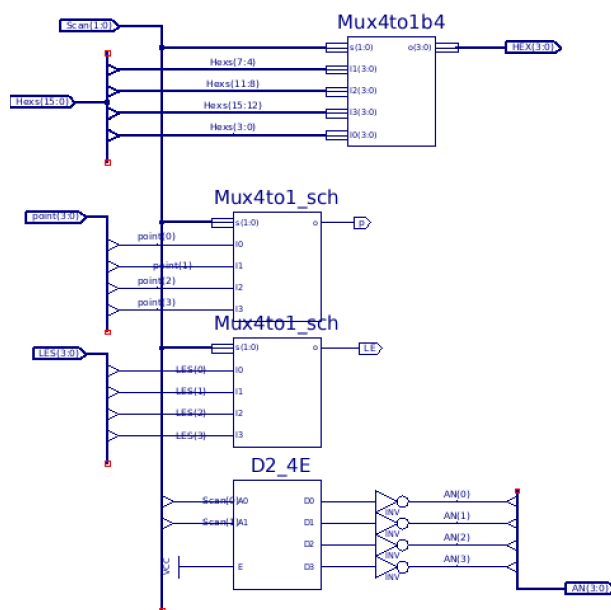


图 3.6 DisplaySync 原理图

(d) 编译并生成对应元器件 DisplaySync

4. DisplayNumber 模块设计

(a) 新建 Schematic 源文件 DispNum

(b) 将实验 6 绘制的 MyMC14495 文件导入

(c) 绘制原理图如下：

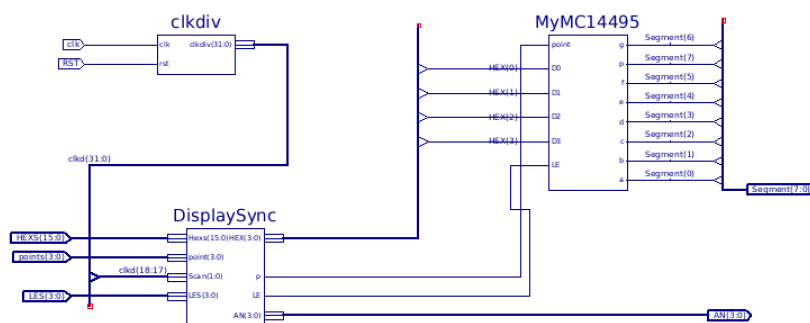


图 3.7 DispNum 原理图

(d) 编译生成对应的 Verilog 代码

5. Top 模块设计

(a) 新建 Verilog 源代码文件 top

(b) 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

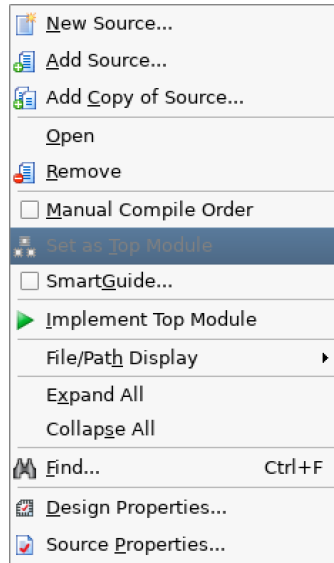


图 3.8 top module

(c) 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
```

6. 建立引脚约束文件

(a) 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;
```

(b) 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;
```

(c) 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 "BTN4" LOC = W16 | IOSTANDARD = LVCMOS18;
```

7. 下载到 SWORD 板上进行验证

(a) 点击 Generate Programming Files, 通过后点击 Configure Target Device -> Manage Configuration Project。

(b) 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

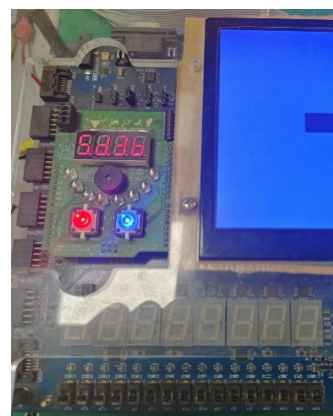


图 3.10

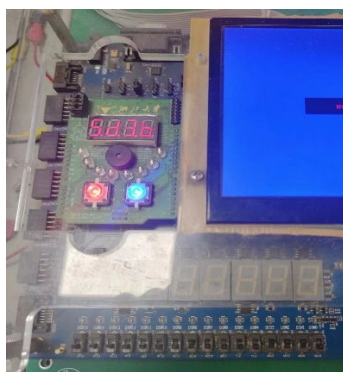


图 3.11

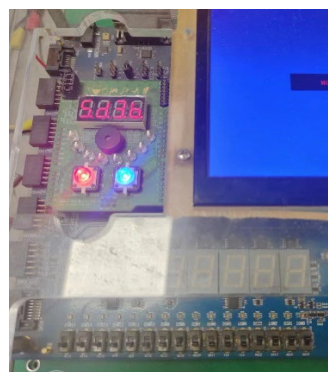


图 3.12

➤ 控制数码管开关

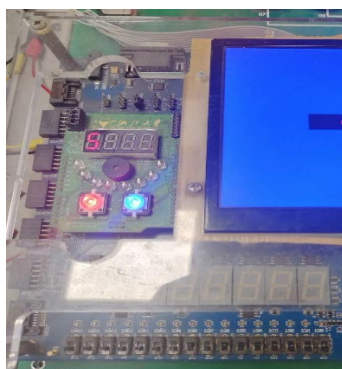


图 3.13

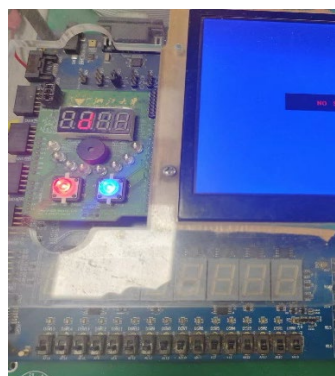


图 3.14

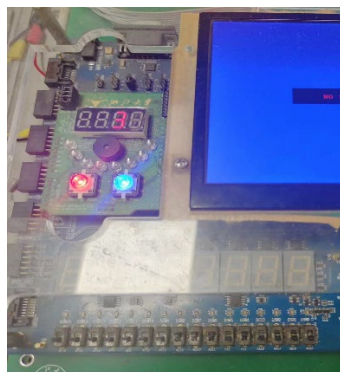


图 3.15



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

四、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：

- (a) INV 用于对输入信号取反
- (b) AND3 用于实现三个输入信号的与逻辑运算
- (c) OR4 用于实现四个输入信号的或逻辑运算。

2. 仿真模拟

(a) 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：

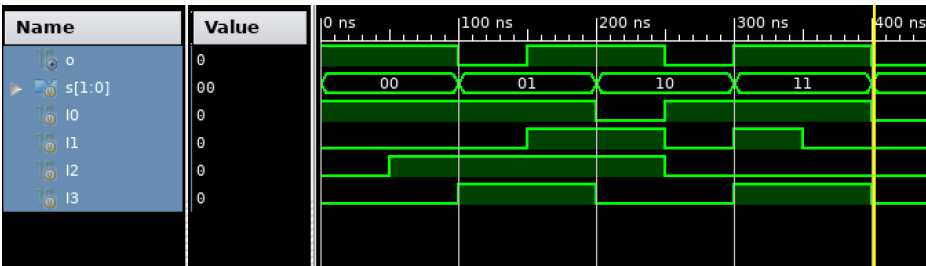


图 4.1 仿真波形图 1

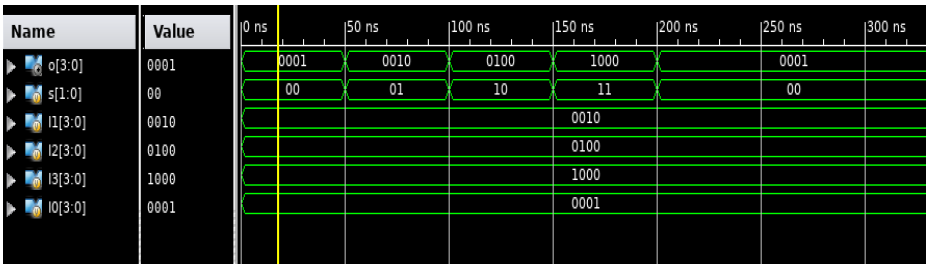


图 4.2 仿真波形图 2

3. 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！