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：刘晨

学　　院：计算机科学与技术学院

系：

专　　业：图灵班

学　　号：3190104666

指导教师：董亚波

2020 年 11 月 5 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：刘晨 专业：图灵 1901 学号：3190104666

同组学生姓名：林初涵 指导老师：董亚波

实验地点：东 4-509 实验日期：2020 年 10 月 29 日

一、实验目的和要求

掌握数据选择器的工作原理和逻辑功能

掌握数据选择器的使用方法

掌握 4 位数码管扫描显示方法

4 位数码管显示应用—记分板设计

二、实验内容和原理

内容：

任务 1：数据选择器设计

任务 2：记分板设计

原理：

4 选一多路选择器 MUX4to1，是一个根据事件简化真值表，其输出是控制信号全部最小项的与或结构。

其输出真值表如下：

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 1 输出真值表

其中输出项是一个变量译码器。

对于四选一多路选择器进行位扩展，进行每路输入项量化，对通道多位复制并共享控制：

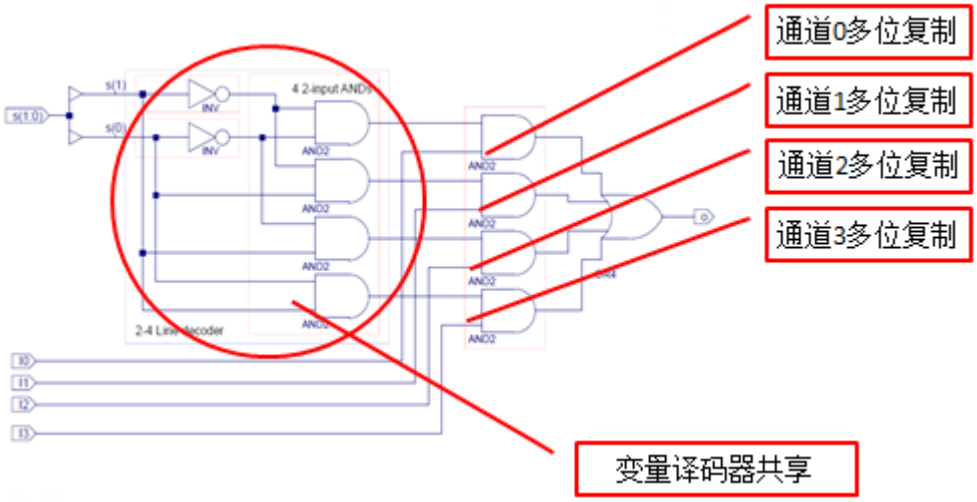


图 2 位扩展示意图

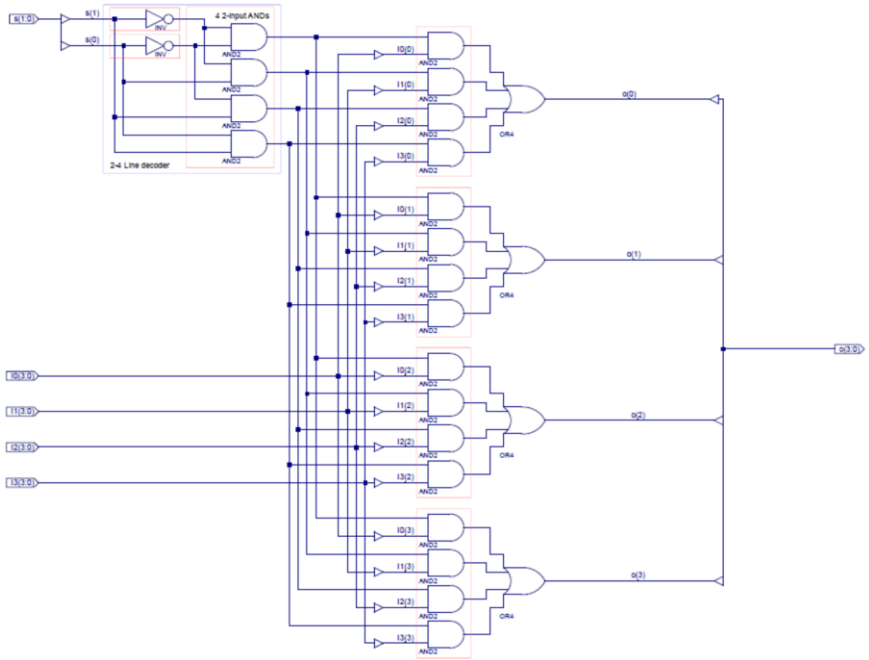


图 3 共享复制示意图

三、实验过程和数据记录

任务 1：数据选择器设计

1.设计实现数据选择器

新建工程，工程名称用 Mux4to1b4_sch；

新建源文件，类型是 Schematic，文件名称用 Mux4to14b；

按照如下原理图方式设计 4 位 4 选 1 数据选择器。

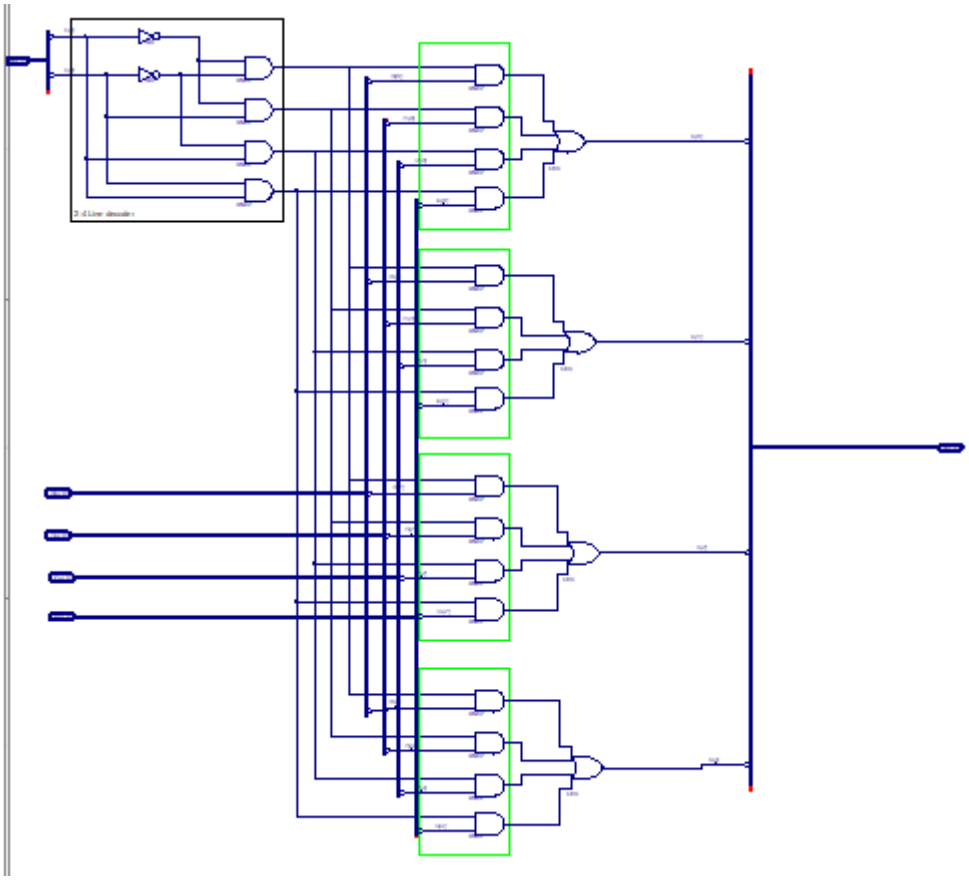


图 4 原理图

Check Design Rules，检查错误

View HDL Functional Model，查看并学习 Verilog HDL 代码

代码如下：

```
                    .O(XLXN_71));
    AND2  XLXI_25 (.I0(I2[3]),
                   .I1(XLXN_16),
                   .O(XLXN_72));
    AND2  XLXI_26 (.I0(I3[3]),
                   .I1(XLXN_20),
                   .O(XLXN_73));
    OR4   XLXI_43 (.I0(XLXN_82),
                  .I1(XLXN_83),
                  .I2(XLXN_84),
                  .I3(XLXN_85),
                  .O(o[0]));
```

图 5 硬件描述代码（部分）

2.仿真

对 MyMC14495 模块进行仿真，参考激励代码如下：

```
28  // Initialize Inputs
29  //   `ifdef auto_init
30  integer i,lch;
31         initial begin
32
33        s = 0;
34        I0 = 0;
35        I1 = 0;
36        I2 = 0;
37        I3 = 0;
38        for(lch=0;lch<=3;lch=lch+1)begin
39        #50;
40        s=lch;
41           for (i=0; i<=65535;i=i+1) begin
42           #50;
43           {I3,I2,I1,I0}=i;
44           end
45        end
46     end
47  endmodule
```

图 6 参考激励代码

得到的仿真波形如下：

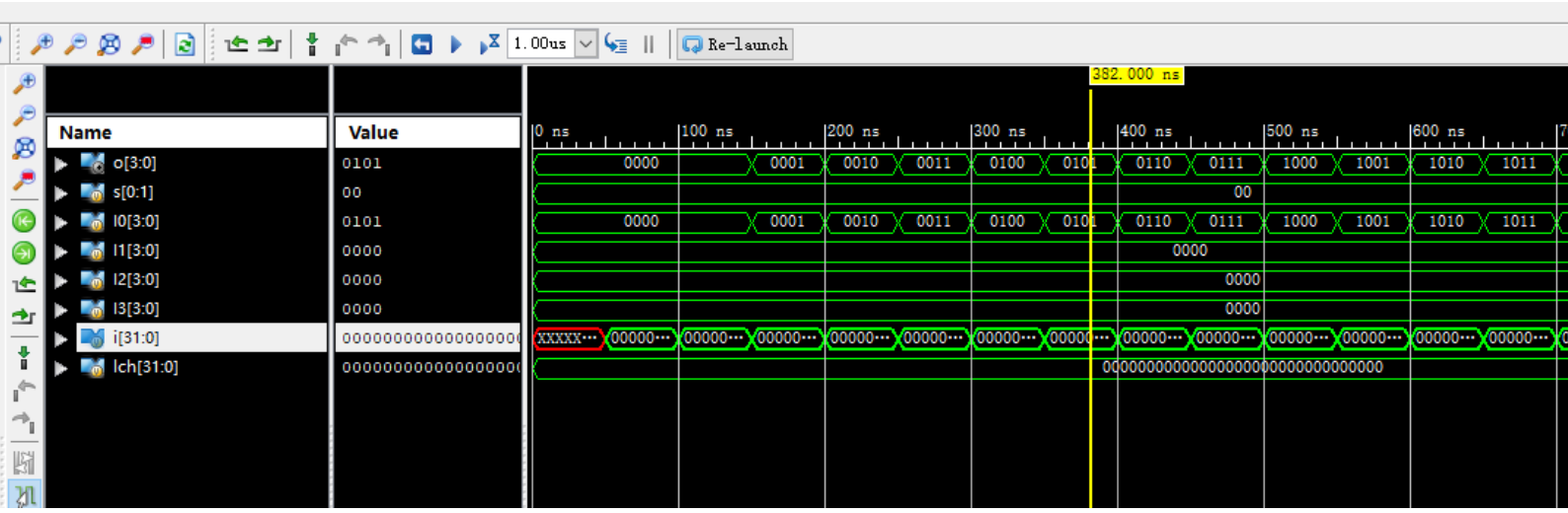


图 7 仿真波形

任务 2：记分板设计

1. 调用 MyMC14495

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以及通用计数分频模块。

使用 verliog 代码编写按键输入模块的代码。

其中所用到的按键输入模块代码分别为：

```
20  //////////////////////////////////////////////////////////////////
21  module clkdiv(
22      input clk,
23      input rst,
24      output reg[31:0]clkdiv
25      );
26    always @ (posedge clk or posedge rst) begin
27        if (rst) clkdiv <= 0;
28        else clkdiv <= clkdiv + 1'b1;
29    end
30  endmodule
31
```

图 9 时钟模块代码

```
21  module DispSync(
22      input [15:0]Hexs,
23      input [1:0]Scan,
24      input [3:0]Point,
25      input [3:0]Les,
26      output reg[3:0]Hex,
27      output reg p,LE,
28      output reg[3:0]AN
29      );
30    always @* begin
31        case(Scan)
32            2'b00 : begin Hex <= Hexs[3:0]; AN <= 4'b 1110; LE <= Les[0]; p <= Point[0];end
33            2'b01 : begin Hex <= Hexs[7:4]; AN <= 4'b 1101; LE <= Les[1]; p <= Point[1];end
34            2'b10 : begin Hex <= Hexs[11:8]; AN <= 4'b 1011; LE <= Les[2]; p <= Point[2];end
35            2'b11 : begin Hex <= Hexs[15:12]; AN <= 4'b 0111; LE <= Les[3]; p <= Point[3];end
36        endcase
37    end
38  endmodule
39
```

图 10　DisplaySync 模块代码

按照原理图设计 DispNum 模块：

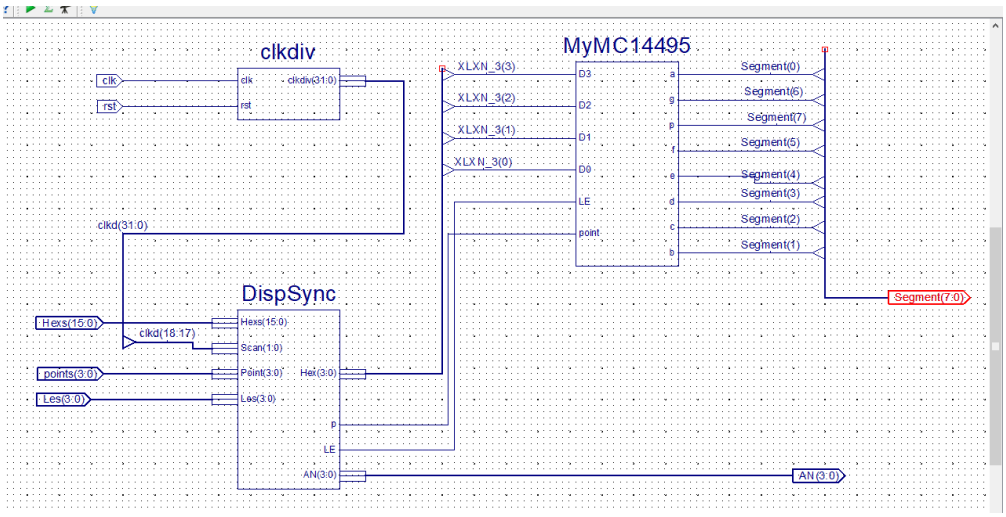


图 11　DispNum 模块的原理图

设计 CreateNumber：

```
21  module CreateNumber(
22      input wire [3:0] btn,
23      output reg [15:0] num
24      );
25      wire [3:0] A,B,C,D;
26      
27      initial num <= 16'b1010_1011_1100_1101;
28  
29      assign A = num[3:0] + 4'd1;
30      assign B = num[7:4] + 4'd1;
31      assign C = num[11:8] + 4'd1;
32      assign D = num[15:12] + 4'd1;
33  
34      always@(posedge btn[0]) num[3:0] <= A;
35      always@(posedge btn[1]) num[7:4] <= B;
36      always@(posedge btn[2]) num[11:8] <= C;
37      always@(posedge btn[3]) num[15:12] <= D;
38  
39  endmodule
```

图 12　CreateNumber 模块代码

设计顶层模块：

```
21  module top(input wire clk,
22     input wire [7:0] SW,
23     input wire [3:0] btn,
24     output wire [3:0] AN,
25     output wire [7:0] SEGMENT,
26     output wire BTNX4
27  );
28     wire [15:0] num;
29  
30     CreateNumber c0(btn,num);
31     
32     dispnum d0(clk, num, SW[7:4], SW[3:0], 1'b0, AN, SEGMENT);
33  
34     assign BTNX4 = 1'b0;
35  
36  endmodule
```

图 13 顶层模块代码

2.下载验证

下载验证，其中所用 ucf 文件内容为：

```
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;

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 "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;
```

图 14　ucf 文件

3.生成 bit 文件并下载到开发板

四、实验结果分析

如图是开发板导入 bit 文件后的初始状态. 调试结果如下：

图 15 初始状态

可以看到，初始状态，所有开关都保持为状态 0 时，数字板数字指向 1314.

进行调试：

（1）打开 SW[3]—SW[0]

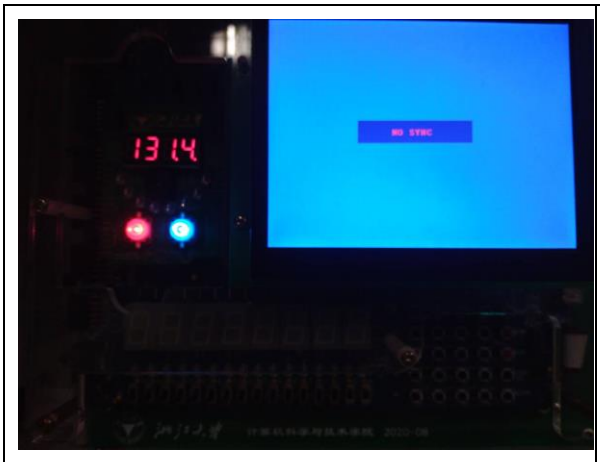
开关	SW[3]	SW[2]	SW[1]	SW[0]	SW[7]	SW[6]	SW[5]	SW[4]
状态	1	1	1	1	0	0	0	0

图 15 状态 1

可以看到，数字之间添加了小数点。

（2）保持 SW[7]—SW[4]不懂，对于 SW[3]—SW[0]多次调试。

开关	SW[3]	SW[2]	SW[1]	SW[0]	SW[7]	SW[6]	SW[5]	SW[4]
状态 2	0	0	1	1	0	0	0	0
状态 3	0	1	0	0	0	0	0	0



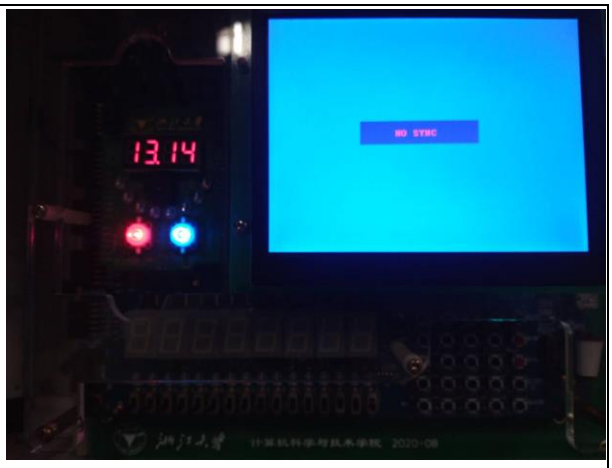


图 10 状态 2 状态 3

可以看出，SW[3]—SW[0]分别控制从左向右的四个小数点。

（3）对于 SW[7]—SW[0]多次调试

开关	SW[3]	SW[2]	SW[1]	SW[0]	SW[7]	SW[6]	SW[5]	SW[4]
状态 4	0	1	0	0	0	1	1	0
状态 5	1	1	1	1	1	1	1	1
状态 6	1	1	1	1	1	0	0	0
状态 7	1	1	1	1	0	0	0	1

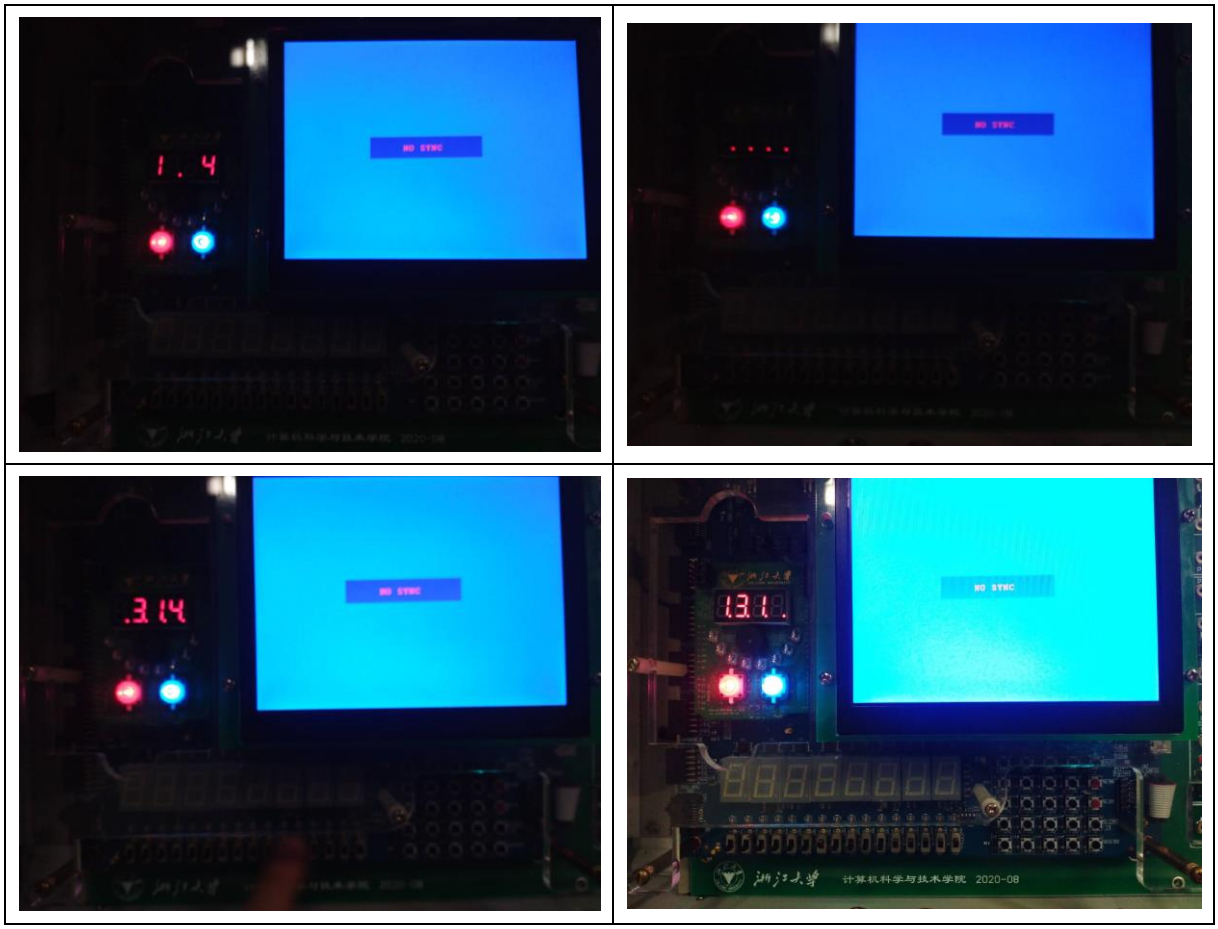


图 10 状态 4 5 6 7

可知，SW[7]—SW[4]分别控制着四位数字的消隐。

（4）保持开关为全闭状态，依次按下四个按钮。

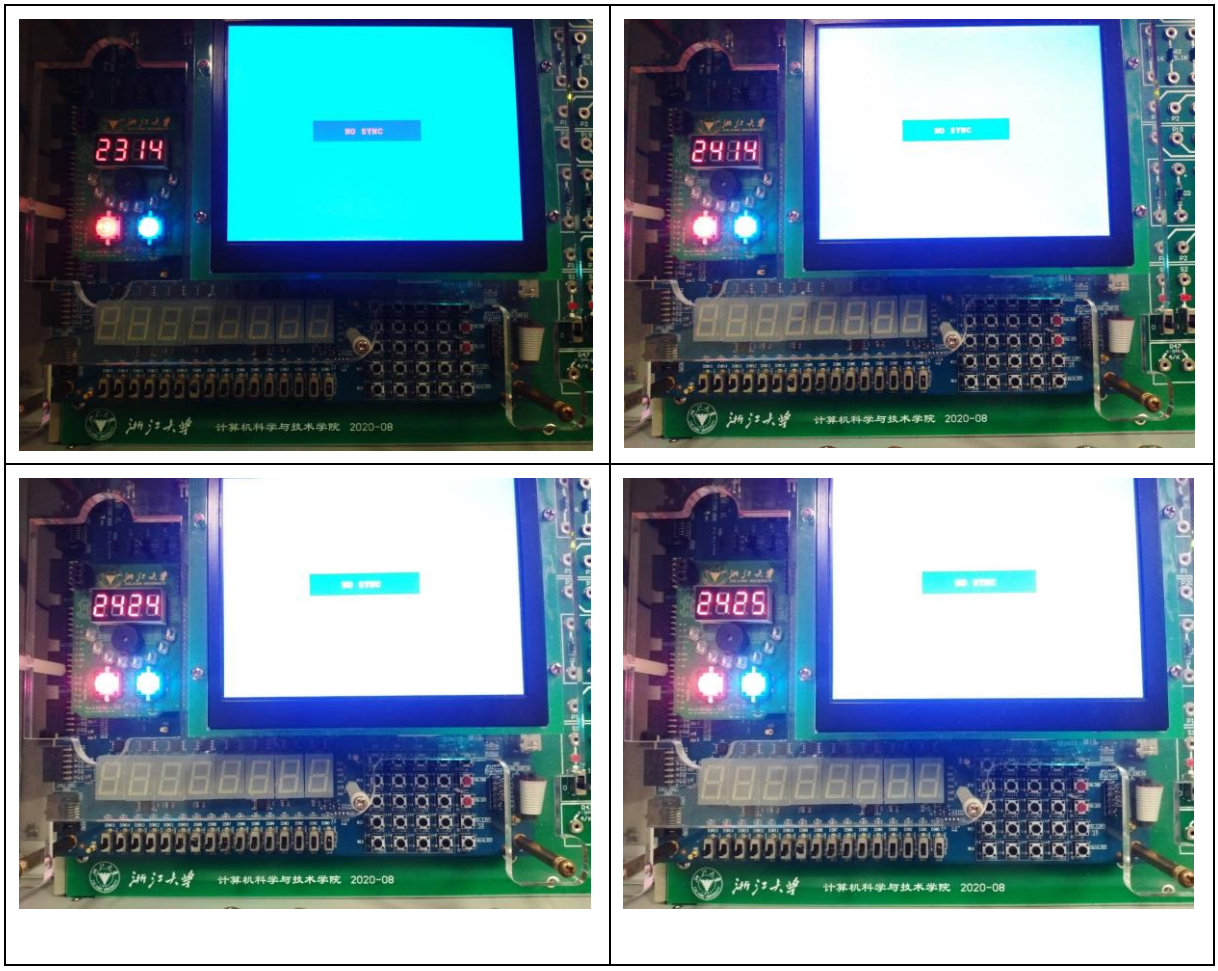


图 11 状态 8 9 10 11

可以看出，每按下一个按钮，其所对应的数字板上面的数字增大一位。

按下从左到右第一个按钮，数字变化为 2314；

按下第二个按钮，数字变化为 2414；

按下第三个按钮，数字变化为 2424；

按下第四个按钮，数字变化为 2425.

四、实验结果分析

① 分析生成的硬件描述代码

硬件描述代码符合预期，并能输出正确结果。

② 分析仿真结果

结合仿真代码，仿真给的激励是依次改变 I3，I2，I1，I0 的值，在这种激励下得到输出的仿真结果是不同的数码管的仿真结果，这个结果符合预期。这样的结果在开发板上体现如下表。

③ 分析开发板结果

开发板结果符合设计要求。

开发板调试结果汇总：

开关 0 1 2 3 分别控制四个小数点；

开关 4 5 6 7 分别控制四个数字是否亮；

按钮每按一下，数字加一。

五、讨论与心得

在仿真代码时，发现如果采用#50，那么由于数据过多，后面 I2，I3 的值都始终保持在 0000 没有改变，于是我把#50 改变成为了#20，#5，仿真图像变密了且仿真出来的值也变多了。

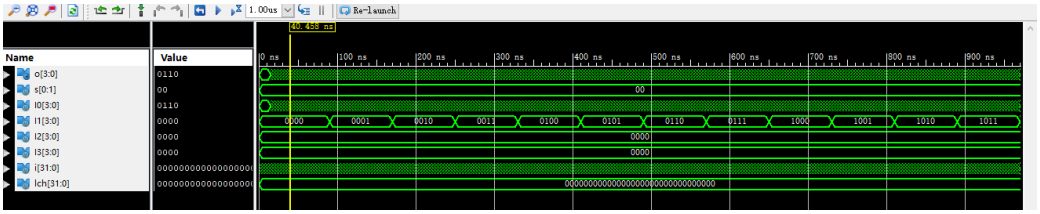


图 12　改成#20 后的图像