电子科技大学电子技术应用实验
实 验 报 告
（FPGA 部分测试）

学生姓名： 张前锋 **学号：** 2020020910019 **报告评分：** _____

实验地点： 科 A **实验时间：** 第 15 周一第 9、10 节 **指导老师：** 王丁

一、实验项目名称： 简易电梯控制系统的设计

二、实验学时： 6

三、项目要求：

1、实现 2 层楼的简易电梯控制系统。

2、电梯有 4 个按键。

1 楼外只有向上按键（KEY0），2 楼外只有向下按键（KEY1），电梯内还有 2 个按键分别为 1 楼按键（KEY2）和 2 楼按键（KEY3）。所有楼层外和电梯内的按键产生的信号作为给电梯的运行请求信号。

3、电梯有 4 个指示灯（LED0、LED1、LED2、LED3）。

LED0：按下 KEY0 键，若电梯不在 1 楼，则 LED0 亮。

LED1：按下 KEY1 键，若电梯不在 2 楼，则 LED1 亮；

LED2：电梯在 1 楼，按 KEY3 键，则 LED3 亮，电梯到 2 楼后 LED3 灭

LED3：电梯在 2 楼，按 KEY2 键，则 LED2 亮，电梯到 1 楼后 LED 灭。

4、有 2 个数码管，分别显示当前运行状态及楼层。

（1）1 个数码管显示当前运行状态，电梯有三个运行状态：待机、上行、下行。

待机：电梯停在 1 楼或 2 楼且无请求信号时均为待机状态。

上行状态：电梯停在 1 楼，有 KEY1 或 KEY3 被按下，进入上行状态。

下行状态：电梯停在 2 楼，有 KEY0 或 KEY2 被按下，进入下行状态。

（2）1 个数码管显示所在楼层，显示 1 或 2；每一层楼之间的运行时间间隔为 5 秒。

5、有 2 个拨码开关。

（1）复位开关。向下拨动后，电梯复位回到一楼。

（2）启动开关。向上拨动后，按键有效，电梯正常工作。

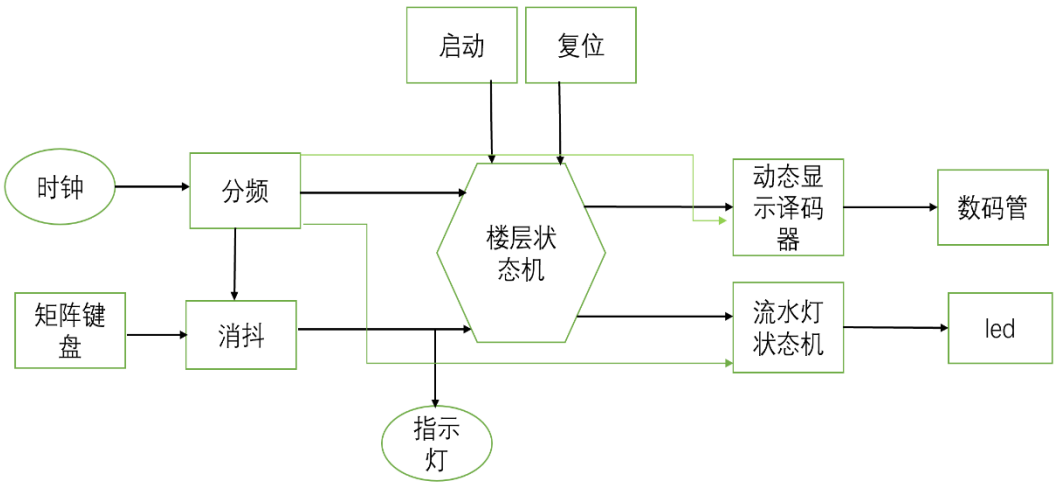
6、增加其他功能可自由发挥。

（1）电梯上行时，LED11 至 LED7 五个指示灯从左向右每隔一秒点亮一个；电梯下行时，LED7 到 LED11 五个指示灯从右到左每隔一秒亮一个。

（2）电梯上行时，楼层显示数码管前 4 秒显示 1,后 1 秒显示 2；

电梯下行时，楼层显示数码管前 4 秒显示 2，后 1 秒显示 1。

四、设计方案：



总共设计 7 个模块，顶层模块 elevatortop、分频模块 divclk、矩阵键盘检测模块 keyboard、消抖模块 ajxd、楼层状态模块 mainstate、数码管显示模块 dynamic_led2 和流水灯模块 led_5。

将电梯运行分为四个状态，使用状态变量 state，当 state=0 时电梯为一楼待机状态；state=1 时，一楼上行状态;state=2 时为二楼待机状态；state=3 为二楼下行状态。keyboard 模块通过矩阵键盘扫描获得四个按键经过 ajxd 模块键盘消抖后 key0-key3 送入楼层状态模块 mainstate，在 mainstate 模块中，首先判断复位信号是否为低电平，如果复位信号为低电平 reset=0 且电梯在二楼的话，令 state=3，将状态置为二楼下行状态，以回到一楼。当启动信号有效时，执行下面逻辑：一楼按下 key0 且电梯处于二楼待机状态时，指示灯 led[0]=1 且 state=3，状态变为二楼下行状态。二楼按下 key1 且电梯处于一楼待机，led[1]=1，state=1,电梯处于一楼上行状态。按下 key3 且电梯处于一楼，led[3]=1,state=1，电梯置于一楼上行状态。按下 key2 且电梯处于二楼，led[2]=1,state=3 电梯置于二楼下行状态。之后对 state 信号进行检测，当 state=1 且 led[1]=1 时，电梯在五秒后变为 state=2 二楼待机状态且 led[1]=0;当 state=1 且 led[3]=1 时，电梯五秒后变为 state=2 且 led[3]=0。对 state=3 时对应变化同理，见源码。在相应的 state 监测部分语句中实现连续按下两个不同按键的逻辑，同样对键盘状态进行检测，当键盘有效时将下一状态保存到 nextstate，在 5 秒后,将 nextsate 变为当前状态以实现返回逻辑。在 mainstate 中将上行和下行再细分为 4 秒加 1 秒来实现发挥功能：电梯上行时，楼层显示数码管前 4

秒显示 1,后 1 秒显示 2；电梯下行时，楼层显示数码管前 4 秒显示 2，后 1 秒显示 1,使用 ledfloor 来记录 4s 时楼层和 1s 时楼层。

state 状态和 ledfloor 送入 dynamic_led2 模块，进行动态显示译码，最后在开发板右面两个数码管显示运行状态和楼层信息。

同时，state 被送入 led_5 模块来是流水灯根据状态变化，从而实现发挥功能电梯上行时，LED11 至 LED7 五个指示灯从左向右每隔一秒点亮一个；电梯下行时，LED7 到 LED11 五个指示灯从右到左每隔一秒亮一个。

五、设计内容与代码

(1)顶层模块：elevatortop

```
module elevatortop(
  input clk,
  input set,     //T9 启动信号
  input reset,  //f3 复位信号
  input [3:0]col,//列键盘
  output  [3:0]row,//行键盘
  output  [3:0]led,//P9,R8,R7,T5
  output [4:0]led5,//E3,H3,G5,R1,T2,led11-led7
  output [1:0]dig,//位选右边两个
  output [7:0]seg//段选
    );
```

```
  wire clk_1k;//1k
  wire clk_50;//50Hz
  wire clk_1s;//1s

divclk u1(.clk(clk),
        .clk_1k(clk_1k),
        .clk_50(clk_50),
        .clk_1s(clk_1s));
        wire k0;
        wire k1;
        wire k2;
        wire k3;
keyboard u2(
        .clk_1k(clk_1k),//键盘扫描
        .clk_50(clk_50),//按键消抖
        .col(col),//列
        .row(row),//行
        .upButton(k0),//对应一楼按键
        .downButton(k1),//对应二楼按键
        .upstair(k2),
        .downstair(k3)//对应电梯内 1 2 按键
        );
```

```
          wire [1:0]state;
           wire [1:0]ledfloor;
mainstate u3(
            .clk(clk),
            .set(set),
             .reset(reset),
            .key0(k0),
            .key1(k1),
            .key2(k2),
            .key3(k3),
            .led(led),//指示灯
            .state(state),//00 待机状态 01 上行状态 10 下机状态
            .ledfloor(ledfloor)
            );
 dynamic_led2 u4(
             .state(state),
              .ledfloor(ledfloor),
             .clk(clk_1k),
             .seg(seg),
             .dig(dig)
             );
```

```
    led_5 u5(
        .clk_1s(clk_1s),
        .state(state),
        .led_o(led5)
    );
Endmodule
```

(2)分频模块：divclk

```
module divclk(clk,clk_1k,clk_50,clk_1s);
      inout clk;
      output reg clk_1k=0;//后面需要改变数值，所以使用 reg 类型
      output reg clk_50=0;
      output reg clk_1s=0;

      integer clk_div_cnt0=0;//计数个数保存
      integer clk_div_cnt1=0;
      integer clk_div_cnt2=0;
      //1khz
      always @ (posedge clk)
              begin
                  if (clk_div_cnt0==24999)////判断是否达到最数,
                  begin
```

```
                clk_div_cnt0<=0;
                clk_1k=~clk_1k;//达到分频并反转
            end
            else
                clk_div_cnt0<=clk_div_cnt0+1;//未达到，继续计数
        end
    //50hz  ,20ms
      always @ (posedge clk)
        begin
            if (clk_div_cnt1==499999)//分频
                begin
                clk_div_cnt1<=0;
                  clk_50=~clk_50;
                 end
            else
                clk_div_cnt1<=clk_div_cnt1+1;
          end
     //1s
       always @ (posedge clk)
            begin
            if (clk_div_cnt2==24999999)//分频
```

```
                        begin
                        clk_div_cnt2<=0;
                         clk_1s=~clk_1s;
                         end
                         else
                          clk_div_cnt2<=clk_div_cnt2+1;
                    end
endmodule
```

(3)矩阵键盘检测模块：keyboard

```
module keyboard(
input clk_1k,//键盘扫描时钟
input clk_50,//按键消抖时钟 50Hz
input [3:0]col,//4 列输入
output  [3:0]row,//4 行输出
output  upButton,//对应一楼电梯外按键
output  downButton,//对应二楼电梯外按键
output  upstair,//一楼电梯内按键
output  downstair);//二楼电梯内按键
assign row[3:0]=4'b0001;//固定一行 row3
reg [3:0] btn=0;//初始化为 0
always @ (negedge clk_1k)
```

```
  begin
      btn[3:0]=col;
  end
 //对四个按键进行消抖
ajxd u0(
      .btn_in(btn[3]),
      .clk(clk_50),
      .btn_out(downstair)
      );
 ajxd u1(
       .btn_in(btn[2]),
       .clk(clk_50),
       .btn_out(upstair)
        );
ajxd u2(
       .btn_in(btn[1]),
       .clk(clk_50),
       .btn_out(downButton)
         );
ajxd u3(
       .btn_in(btn[0]),
       .clk(clk_50),
```

```
          .btn_out(upButton)
            );
Endmodule
```

(4)键盘消抖模块：ajxd

```
module ajxd(
    input btn_in, //未消抖信号
    input clk,//50Hz 时钟
    output btn_out//消抖后按键输出
    );
    reg  btn0=0;//定义了 btn0 寄存器
    reg  btn1=0;//定义了 btn1 寄存器
    reg  btn2=0;//定义了 btn2 寄存器
 always@ (posedge clk)
      begin
          btn0<=btn_in;
          btn1<=btn0;
         btn2<=btn1;
      end
      assign
btn_out=(btn2&btn1&btn0)|(~btn2&btn1&btn0);//btn_out 信号，每按
下一次，只产生一个上升沿
```

```
endmodule
```

(5)楼层状态模块：mainstate

```
module mainstate(
input clk,//系统时钟
input set,//启动
input reset,//复位
input key0,//一楼外按键
input key1,//二楼外按键
input key2,//一楼内按键
input key3,//二楼内按键
output reg [3:0]led=0,//四个 led 指示灯
output reg [1:0]state=0,//00，一楼待机    01，一楼上行状态  10
二楼待机状态  11 二楼下行状态
output reg[1:0] ledfloor=1
   );
   integer clk_count0=0;
   integer clk_count1=0;
   reg[1:0] nextstate=0;
always@(posedge clk)
begin
if(!reset)
begin
```

```
led=4'b0000;//led 清零
if(state==2)//如果在第二层的话
begin
state=3;//置于二楼下行状态
end
end
else if(set)//启动了,接下来按键才有效果

begin
 if((key0)&&state==2)  //一楼按下 key0 且电梯二楼待机
                 begin
                          led[0]=1; //一楼电梯外灯亮
                          state=3;  //2 楼下行状态
                  end
else if((key1)&&state==0)//二楼按下且电梯在一楼待机
                         begin
                                 led[1]=1;//二楼电梯外灯亮
                                 state=1;//1 楼上行状态
                         end

else if((key3)&&state==0)  //在一楼 电梯内按下
                            begin
```

```
                                led[3]=1;
                                state=1; //电梯一楼上行状态


                            end
else if((key2)&&state==2)  //在二楼，电梯内按
                              begin
                              led[2]=1;
                              state=3; //电梯二楼下行状态
                              end
end



if(state==1&&led[1]==1)//key1 按下引发上行状态
begin
if(clk_count0==249999999)
            begin
                clk_count0=0;
                 state=2;  //2 楼待机状态
                led[1]=0;//led 清零
            end
```

```
        else
            clk_count0=clk_count0+1;
end

if(state==1&&led[3]==1)//key3 按下引发上行状态
begin
//接下来先实现按下 key3 后立刻按下 key2 逻辑,不能写在上面的逻辑中判断，顺序执行太快，不能执行 key2 逻辑
 if(key2)//如果按下 key2 进入逻辑
                begin
                led[2]=1;
                nextstate=3;//到二楼后下一个状态为二楼下行
                end
 //实现立刻按下 key0 逻辑
 else if(key0)//如果按下 key2 进入逻辑
                begin
                led[0]=1;
                nextstate=3;//到二楼后下一个状态为二楼下行
                end

if(clk_count0==249999999)//计数 5s
            begin
```

```
                clk_count0=0;
                 state=2;  //2 楼待机状态
                led[3]=0;//led 清零

                        //立刻按下逻辑
                         //立刻按下 led2 逻辑
                        if(led[2])//led2 亮说明接着按下了 key2
                        begin
                        state=nextstate;//改变当前状态为 2 楼下
行
                        end
                        //立刻按下 key0
                        else if(led[0])//led0 亮说明接着按下了
key0
                         begin
                         state=nextstate;//改变当前状态为 2 楼下
行
                         end

            end
        else
            clk_count0=clk_count0+1;
```

```
end



if(state==3&&led[0]==1)//key0 按下引发 2 楼下行状态
begin
 if(clk_count1==249999999)
            begin
                clk_count1=0;
                 state=0;  //1 楼待机状态
                led[0]=0;//led 清零
            end
        else
            clk_count1=clk_count1+1;
end



if(state==3&&led[2]==1)//key2 按下引发 2 楼下行状态,
begin

//接下来实现立刻按下逻辑
 if(key3)//如果按下 key3 进入逻辑
                begin
```

```
                led[3]=1;
                nextstate=1;//到一楼后下一个状态为一楼上行
                end
 //实现立刻按下 key1 逻辑
 else if(key1)//如果按下 key1 进入逻辑
                begin
                led[1]=1;
                nextstate=1;//到一楼后下一个状态为一楼上行，以
返回二楼
                end

if(clk_count1==249999999)
              begin
                clk_count1=0;
                 state=0;  //1 楼待机状态
                led[2]=0;//led 清零

                        //立刻按下逻辑
                         if(led[3])//led3 亮说明接着按下了 key3
                           begin
                           state=nextstate;//改变当前状态
                           end
```

```
                                //立刻按下 key0
                                else if(led[1])//led1 亮说明接着按下
了 key1
                                 begin
                                 state=nextstate;//改变当前状态
                                 end


            end
         else
            clk_count1=clk_count1+1;
end
if(state==3&&!reset)//复位动作引发 2 楼下行状态
begin
if(clk_count1==249999999)//计数第 5s
            begin
                clk_count1=0;//清零
                state=0;//1 楼待机状态
            end
else
      clk_count1=clk_count1+1;
end
```

```
end
//下行 4s+1s 显示与上行 4+1s 显示，当 state 状态变化时执行
always@(state)
begin
if(state==3&&clk_count1==199999999)
begin
ledfloor=1;
end
else if(state==1&&clk_count0==199999999)
begin
ledfloor=2;
end
end
endmodule
```

(6)流水灯模块:led_5

```
module led_5(
input clk_1s,//1s 时钟
input[1:0] state,
output reg[4:0] led_o
    );
    reg[2:0] s_present=0;
```

```
reg[2:0] s_next;
always@(s_present,state)
if(s_present==4|state==0|state==2)
    begin
    s_next=-1;//保证从 state 变化后开始显示第一个
    end
   else
    s_next=s_present+1;
 always@(negedge clk_1s)
 begin
        s_present<=s_next;
 end
 always@(state,s_present)
 begin
 if(state==1)
    case(s_present)
        0:led_o=5'b10000;
        1:led_o=5'b01000;
        2:led_o=5'b00100;
        3:led_o=5'b00010;
        4:led_o=5'b00001;
        default:led_o=5'b00000;
```

```
          endcase
        else if(state==3)
         case(s_present)
               0:led_o=5'b00001;
               1:led_o=5'b00010;
               2:led_o=5'b00100;
               3:led_o=5'b01000;
               4:led_o=5'b10000;
               default:led_o=5'b00000;
          endcase
         else
             led_o=5'b00000;
     end
endmodule
```

(7)数码管动态显示模块:dynamic_led

```
module dynamic_led2(
input [1:0]state,//状态
input [1:0]ledfloor,
input clk,       //1kHz 信号
output reg [7:0] seg,//段码
output reg [1:0] dig//位码，最右边两个
```

```
);
reg [1:0] num=0;
    always @ (posedge clk)
    begin
        if (num>=1)
            num=0;
        else
            num=num+1;
    end

    //译码器
    always @ (num)
    begin
        case(num)
        0:dig=2'b10;//右边第一个
        1:dig=2'b01;//右边第二个
        default: dig=0;
        endcase
    end

    //数据选择器，确定显示数据
    reg [3:0] disp_data;
```

```
always @ (num)
begin
    case(num)
    0:
          case(state)
          0:disp_data=1;
          1:disp_data=ledfloor;
          2:disp_data=2;
          3:disp_data=ledfloor;
          endcase
     1:
          case(state)
                    0:disp_data=3;
                    1:disp_data=4;
                    2:disp_data=3;
                    3:disp_data=5;
             endcase
    default: disp_data=0;
    endcase
end
//显示译码器
always@(disp_data)
```

```
    begin
        case(disp_data)
        4'h1: seg=8'h06;//1 楼
        4'h2: seg=8'h5b;//2 楼
        4'h3: seg=8'h40;//待机
        4'h4: seg=8'h01;//上行
        4'h5: seg=8'h08;//下行
        default: seg=0;
        endcase
    end
endmodule
```

（8）管脚约束文件

```
set_property PACKAGE_PIN D4 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
##switch
set_property PACKAGE_PIN T9 [get_ports set]
set_property IOSTANDARD LVCMOS33 [get_ports set]
set_property PACKAGE_PIN F3 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
##Buttons 约束
set_property PACKAGE_PIN K3 [get_ports {row[0]}]
```

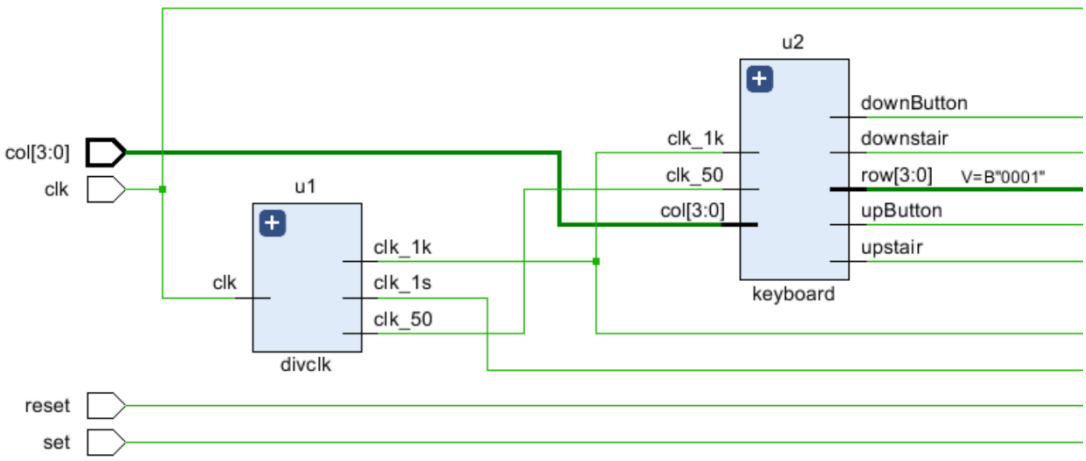
```
set_property IOSTANDARD LVCMOS33 [get_ports {row[0]}]
set_property PACKAGE_PIN M6 [get_ports {row[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[1]}]
set_property PACKAGE_PIN P10 [get_ports {row[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[2]}]
set_property PACKAGE_PIN R10 [get_ports {row[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[3]}]
##按键
set_property PACKAGE_PIN R12 [get_ports {col[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[0]}]
set_property PACKAGE_PIN T12 [get_ports {col[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[1]}]
set_property PACKAGE_PIN R11 [get_ports {col[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[2]}]
set_property PACKAGE_PIN T10 [get_ports {col[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[3]}]

##下拉列线到低电平
set_property PULLDOWN true [get_ports {col[3]}]
set_property PULLDOWN true [get_ports {col[2]}]
set_property PULLDOWN true [get_ports {col[1]}]
set_property PULLDOWN true [get_ports {col[0]}]
```

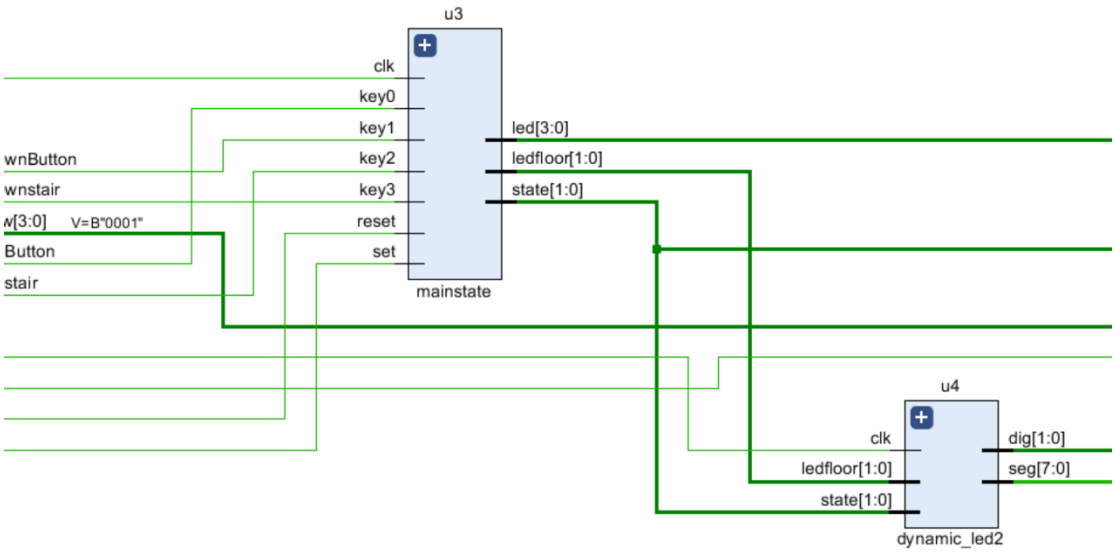
```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets col_IBUF]
##set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets col_IBUF]
##显示
set_property PACKAGE_PIN G12 [get_ports {dig[0]}]
set_property PACKAGE_PIN H13 [get_ports {dig[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {dig[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dig[1]}]

set_property PACKAGE_PIN L13 [get_ports {seg[7]}]
set_property PACKAGE_PIN M14 [get_ports {seg[6]}]
set_property PACKAGE_PIN P13 [get_ports {seg[5]}]
set_property PACKAGE_PIN K12 [get_ports {seg[4]}]
set_property PACKAGE_PIN K13 [get_ports {seg[3]}]
set_property PACKAGE_PIN L14 [get_ports {seg[2]}]
set_property PACKAGE_PIN N12 [get_ports {seg[1]}]
set_property PACKAGE_PIN P11 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
```

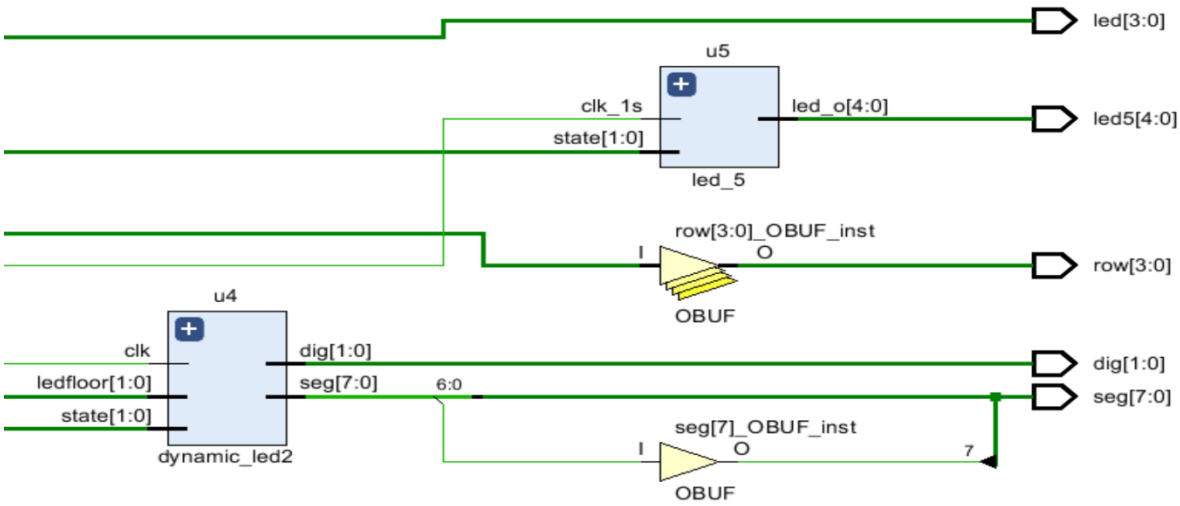
```
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
##灯
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]

set_property PACKAGE_PIN T5 [get_ports {led[3]}]
set_property PACKAGE_PIN R7 [get_ports {led[2]}]
set_property PACKAGE_PIN R8 [get_ports {led[1]}]
set_property PACKAGE_PIN P9 [get_ports {led[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {led5[4]}]
set_property PACKAGE_PIN E3 [get_ports {led5[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {led5[3]}]
set_property PACKAGE_PIN H3 [get_ports {led5[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {led5[2]}]
set_property PACKAGE_PIN G5 [get_ports {led5[2]}]
```

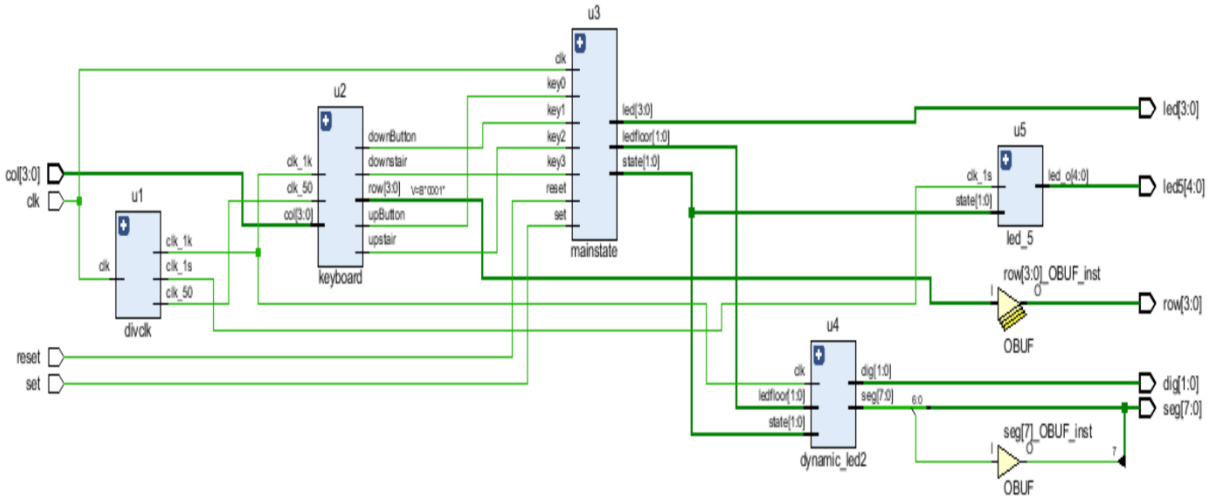
```
set_property IOSTANDARD LVCMOS33 [get_ports {led5[1]}]

set_property PACKAGE_PIN R1 [get_ports {led5[1]}]


set_property IOSTANDARD LVCMOS33 [get_ports {led5[0]}]

set_property PACKAGE_PIN T2 [get_ports {led5[0]}]
```

六、RTL 分析截图









七、实现情况总结

经过验证测试，本设计实现了全部的基础功能部分，同时实现了两项发挥功能，“电梯上行时，LED11 至 LED7 五个指示灯从左向右每隔一秒点亮一个；电梯下行时，LED7 到 LED11 五个指示灯从右到左每隔一秒亮一

个。电梯上行时，楼层显示数码管前 4 秒显示 1,后 1 秒显示 2；电梯下行时，楼层显示数码管前 4 秒显示 2，后 1 秒显示 1。”验收表要求的设计任务全部完成。

通过本次 FPGA 课程综合设计，独立完成并实现了简易电梯控制系统，在实现一个个功能的同时获得很大的成就感。完全掌握了 FPGA 的几个基础功能，对 Verilog 语言的使用更加熟悉，编程能力和 FPGA 模块设计能力有很大的提高，对数字电路知识及其应用有了更加深入的理解。