规格严格　　功夫到家

HARBIN INSTITUTE OF TECHNOLOGY

哈 尔 滨 工 业 大 学

数 字 逻 辑 设 计 大 作

项 目 名 称：电梯控制器设计

姓 名：　学 号：

小 组 成	员　　所 在 院 系	学 号
	国 家 示 范 性|软 件 学 院	
	国 家 示 范 性|软 件 学 院	
	国 家 示 范 性|软 件 学 院	

目　录

一、　实验设计

1.1　设计过程

1.1.1　设计目的

当今社会，随着城市建设的不断发展，高层建筑的不断增多，电梯作为高层建筑中垂直运行的交通工具已与人们的日常生活密不可分。电梯控制器的数字系统正适合我们学习研究。

本次大作业，我们小组将设计一个 10 层楼的电梯控制器。功能包括：呼叫电梯、取消呼叫、电梯上升下降、电梯停（开门），并在开发板上模拟电梯运行状态。

1.1.2 设计要求（附加创新功能为红色）

（1） 电梯最少可以往返于 0-9 层，设计电梯控制器每秒运行一层。

（2） 用 10 个输入信号模拟呼叫电梯按钮，乘客可以手动点击输入要去的楼层数 A，同时也可以按键取消错误的输入。显然，课程所要求设置的取消按钮，在多楼层同时呼叫电梯的情况下，无法如愿取消自己想取消的楼层，故本次设计使用重复输入来取消输入的办法。

（3） 用 10 个 LED 排成直线，分别表示某层的呼叫电梯状态，某一层的 LED 灯亮了表示该层有人呼叫电梯，电梯应当按规划前往该层，并停下搭载乘客。

（4） 可自动显示当前电梯所在的楼层数 B，当 B 与 A 不同时，电梯应当前往楼层 A。

（5） 可以自动显示电梯每一次启停之间的运行时间。

（6） 任何时候按下复位键，电梯回到 1 层。

（7） 按键防抖设计，轻触按键（输入信号延续时间小于阈值）时输入无效。

（8） 可以同时**多个楼层呼叫电梯**，此时电梯将按照设定的优先级先后前往各个楼层，一般来讲，电梯上升时优先处理上层信号，下降时，优先处理下层信号。

1.2　工作原理

（1） 系统输入/输出变量

对于一个系统首先要有一个时钟输入，设为 clk；按键输入，设为 btn；数码管显示输入，设为 seg；叫楼梯层状态输出，设为 nfloor。

（2） 按键设计

本实验用开发板上 5 个按钮以及 5 个开关模拟 0-9 层楼的叫梯按键（按钮数量不足，所以以快速波动开关替代），这些按钮分别是：S0，S1，S2，S3，S4SWOSW1，SW2，SW3，SW4。

本实验使用 SW7 按钮作为复位开关。在任意时刻点击该按钮将会使数据清零，电梯回到 1 层。

（3）内部变量

定义了一个 10 位的按键寄存器 btn_pre_re 用于存储按键信息，即呼叫电梯的楼层信息，在防抖设计中，设定为每 200ms 读取一次叫梯信息，因此需要一个周期为 200ms 的时钟信号。

定义了一个电梯位置标识 lift_num，表示电梯当前所在楼层。

定义了一个电梯开门关门显示信号 lift_open（当 lift_open 时表示开门，door=0 表示关门）。

定义了一个电梯当前运行状态变量 lift_state（当 lift_state=1 时表示电梯处于上升状态，当 lift_state=2 时表示电梯处于下降状态）。

（4）显示设计

电梯控制器包括三种显示，其中，一个数码管显示电梯所在楼层，一个数码管显示本次启停之间运行时间，同时有 10 个 LED 灯显示每个楼层呼叫电梯的状态。

其中需要讲解的是，当某楼层没有叫梯时，该楼层对应的 LED 指示灯处于熄灭状态；当有叫梯时，按键对应的 LED 指示灯处于点亮状态；当某一层已经呼叫电梯，但由于某种原因发现不是自己想去的楼层时，再次点击该楼层的叫梯按钮可以取消叫梯。

（5）电梯控制器使用说明

开启电梯控制器或者重启控制器时，默认电梯在楼层 1。每个楼层各与一个 LED 灯相连。

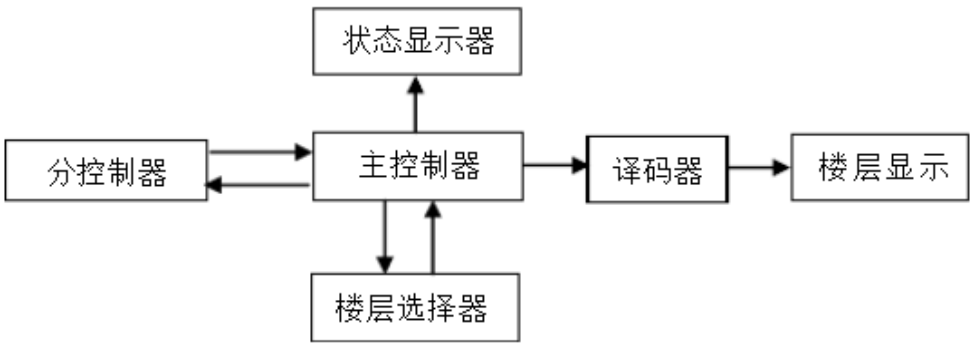
接下来，系统等待用户在任意时刻，点击一至多个楼层的叫梯按钮，这些楼层的叫梯状态按钮随即点亮，电梯按照优先级依次前往这些楼层。

将电梯所在的楼层记为 B，而呼叫电梯的楼层记为 A1，A2…∈集合 A，则该优先次序可表达为：当电梯处于上升时，将优先前往 A_{up}={Ai|Ai∈A 且 A>B}这些楼层，对于这些楼层 A_{up}，将按照由小到大的顺序依次到达。反之，当电梯下降时，将优先前往 A_{down}={Ai|Ai∈A 且 A<B}这些楼层，对于这些楼层 A_{down}，将按照由大到小的顺序依次到达。

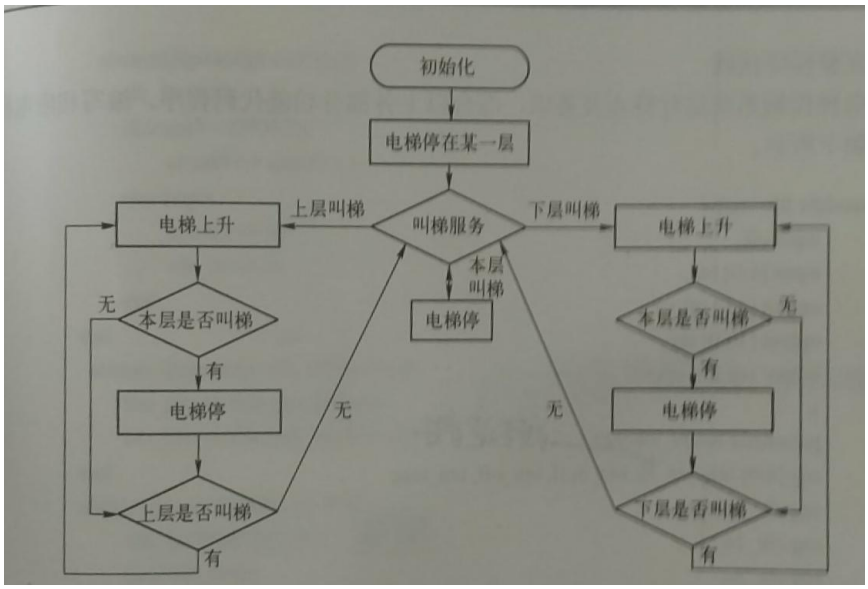
当电梯到达指定楼层时，电梯稍停，开门指示灯亮起，随后开门指示灯熄灭时，电梯将去往其他呼叫电梯的楼层。当没有其他楼层呼叫电梯时，电梯停留在该层，等待用户下一步输入。

1.3　系统方框图

1.3.1 电梯控制器模块结构图



1.3.1 电梯控制器运行流程图



1.4　各部分选定方案、说明和代码

1.4.1 分频器

（1）200ms 时钟信号

1）方案：分频一个 200ms 的时钟，用于按键防抖。

2）代码如下：

```
reg clk_200ms;
parameter N=99_999999;
     always@(posedge clk)begin
         clk_200ms<=0;
         if(count<N/5)
            count<=count+1;
         else begin
            count<=0;
            clk_200ms<=1;
        end
     end
```

3）说明：

在防抖设计中，设定为每 200ms 读取一次叫梯信息，以避免使用时因抖动而出现短时间内连续输入，因此需要一个周期为 200ms 的时

钟信号。我们从分频器中分离得到它。

（2）1s 时钟信号

1） 方案：分频一个 1s 的时钟，用于电梯运行时钟控制。

2） 代码如下：

```
reg clk_1s;
    always@(posedge clk)begin
        clk_1s<=0;
        if(count1<N)
            count1<=count1+1;
        else begin
            count1<=0;
            clk_1s<=1;
        end
    end
```

3） 说明：

为了控制电梯每秒运行一层楼梯，并且记录每次启停间电梯运行的时间，我们需要一个周期为 1s 的时钟信号。我们从分频器中分离得到它。

1.4.2 译码器

（1）显示电梯所在楼层的显示译码器

1）方案：根据当前电梯所在楼层 lift_num 编写译码器，使 7 段数码管显示电梯所在楼层。

2）代码如下：

```
reg clk_1s;
reg [10:0] dout;
reg [4:0] lift_num;
    always@(posedge clk_1s)begin
        case(lift_num-1)
            0:dout=11'b0100_1111110;
            1:dout=11'b0100_0110000;
            2:dout=11'b0100_1101101;
```

```
            3:dout=11'b0100_1111001;
            4:dout=11'b0100_0110011;
            5:dout=11'b0100_1011011;
            6:dout=11'b0100_1011111;
            7:dout=11'b0100_1110000;
            8:dout=11'b0100_1111111;
            9:dout=11'b0100_1111011;
            default:dout=11'b0100_1111110;
        endcase
    end
```

3）说明：

为了将当前电梯所在的楼层用数码管显示出来，我们编写一个显示译码器：用 switch 语句判断当前数字，并输出显示字形所需的各个信号，点亮数码管对应段。为了配合电梯的每秒运行一层的速度，我们还应该每秒刷新一次该模块的数据，显示电梯到达的新楼层。

（2）显示本次电梯启停间运行时间的显示译码器

1）方案：根据当前记录下的启停时间编写译码器，使 7 段数码管显示本次电梯启停间运行时间。

2）代码如下：

```
    reg clk_1s;
    reg [10:0] dout1;
    reg [4:0] clk_num;
       always@(posedge clk_1s)begin
                  case(clk_num)
                      0:dout1=11'b0100_1111110;
                      1:dout1=11'b0100_0110000;
                      2:dout1=11'b0100_1101101;
                      3:dout1=11'b0100_1111001;
                      4:dout1=11'b0100_0110011;
                      5:dout1=11'b0100_1011011;
                      6:dout1=11'b0100_1011111;
                      7:dout1=11'b0100_1110000;
                      8:dout1=11'b0100_1111111;
                      9:dout1=11'b0100_1111011;
```

```
                default:dout1=11'b0100_1111110;
            endcase
        end
```

3）说明：

为了将本次电梯启停间运行时间用数码管显示出来，我们编写一个显示译码器：用 switch 语句判断当前数字，并输出显示字形所需的各个信号，点亮数码管对应段。为了配合电梯的每秒运行一层的速度，我们还应该每秒刷新一次该模块的数据，显示计时器时间。

1.4.3 主体控制模块

（1）按键取消和复位功能

1）方案：使用异或门用于实现按两次按键时进行取消的操作，按复位键使电梯回到楼层一。

2）代码如下：

```
reg [9:0] btn_pre_re, btn_off;
reg btn1_pre_re,btn1_off;
reg clk_200ms;
always@(posedge clk_200ms)begin
      btn_pre_re=btn_pre_re^btn;
      btn_pre_re=btn_pre_re&btn_off;
      btn1_pre_re=btn1_pre_re^btn1;//复位按键
      btn1_pre_re=btn1_pre_re&btn1_off;
      if(btn1_pre_re==1)
      begin
      btn_pre_re<=9'b000000010;
      btn1_pre_re=0;
      end
 end
```

3）说明：

为了避免电梯乘客因为某些原因错按了楼层，发现输入的楼层并不是自己想去的楼层时，可以通过取消呼叫电梯挽回失误，电梯控制器应当有取消的功能。因为课程所要求设置的取消按钮，在多楼层同时呼叫电梯的情况下，无法如愿取消自己想取消的楼层，故本次实验采用重复输入取消的办法，更加贴合实际，实用方便。设置复位键重启电梯控制

器的目的则无需赘述。

（2）寄存器数据操作模块（核心功能）

1）方案：设置 10 位寄存器 btn_pre_re 存储呼叫电梯的楼层，按下按键时将信息传入。到达楼层时用 coul 计数器延时 3s。lift_state 存储当前电梯执行的动作，即上升下降。如此可构建一个模块，使电梯连续上升或连续下降通往呼叫电梯的楼，并在到达每个目的地时，停滞 3 秒。用 lift_num 记录当前楼层。

2）代码如下：

```
reg [9:0] btn_pre_re;
reg [1:0] lift_state;
reg [4:0] lift_num;
reg clk_1s;
initial begin
    btn_off<=9'b111111111;
    btn1_off<=1'b1;
    btn_pre_re<=0;
    lift_num<=2;
    lift_state<=0;
    lift_open<=0;
end
  always@(posedge clk_1s)begin
      btn_buff=btn_pre_re;
      case(lift_state)
          0:begin
              if((btn_buff>>lift_num)>0)begin
              if(cou1<3)//类似 C 语言的思想，实现 3s 的延时
                begin
                cou1=cou1+1;
              end
              if(cou1==3)
                begin
                lift_num=lift_num+1;
                clk_num=clk_num+1;
                lift_state=1;//上层有人叫电梯
              cou1=0;
              end
```

```
                end

                if((btn_buff&(1<<(lift_num-1)))>0)begin
                  btn_buff=btn_buff&(~(1<<(lift_num-1)));
//本层有人叫电梯
                  btn_off=~(1<<(lift_num-1));
                  lift_open=1;
                  if(cou2<3)
                  begin
                  cou2=cou2+1;
                end
                if(cou2==3)
                  begin
                  clk_num=0;
                  lift_open=0;
                  lift_state=0;
                  cou2=0;
                  end
                end

                if((1<<(lift_num-1))>btn_buff)begin
//下层有人叫电梯
                  if(btn_buff>0)begin
                    if(cou3<3)
                  begin
                  cou3=cou3+1;
                end
                if(cou3==3)
                  begin
                    lift_num=lift_num-1;
                    clk_num=clk_num+1;
                    lift_state=2;
                    cou3=0;
                    end
                  end
                end
              end
```

```
1:begin
    if((btn_buff>>lift_num)>0)begin
      if((btn_buff&(1<<(lift_num-1)))>0)begin
          btn_buff=btn_buff&(~(1<<(lift_num-1)));
          btn_off=~(1<<(lift_num-1));
          lift_open=1;
          if(cou4<3)
      begin
      cou4=cou4+1;
    end
    if(cou4==3)
      begin
          clk_num=0;
          lift_open=0;
          lift_num=lift_num+1;
          clk_num=clk_num+1;
          cou4=0;
          end
      end
      else
          lift_num=lift_num+1;
          clk_num=clk_num+1;
    end
    else begin
          btn_buff=btn_buff&(~(1<<(lift_num-1)));
          btn_off=~(1<<(lift_num-1));
          lift_open=1;
          if(cou5<3)
      begin
      cou5=cou5+1;
    end
    if(cou5==3)
      begin
          clk_num=0;
          lift_open=0;
          lift_state=0;
          cou5=0;
          end
```

```
                end
            end

            2:begin
                btn_test=(btn_buff<<(10-lift_num));
                if(btn_test>0)begin
                  if((btn_buff&(1<<(lift_num-1)))>0)begin
//解决下楼命令的其中一个
                      btn_buff=btn_buff&(~(1<<(lift_num-1)));
                      btn_off=~(1<<(lift_num-1));
                      lift_open=1;
                      if(cou6<3)
                  begin
                  cou6=cou6+1;
                end
                if(cou6==3)
                  begin
                      clk_num=0;
                      lift_open=0;
                      lift_num=lift_num-1;
                      clk_num=clk_num+1;
                      cou6=0;
                      end
                  end
                  else
                      lift_num=lift_num-1;
                      clk_num=clk_num+1;
                end
                else begin//解决下楼命令的最后一个
                      btn_buff=btn_buff&(~(1<<(lift_num-1)));
                      btn_off=~(1<<(lift_num-1));
                      lift_open=1;
                      if(cou7<3)
                  begin
                  cou7=cou7+1;
                end
                if(cou7==3)
                  begin
```

```
                    clk_num=0;
                    lift_open=0;
                    lift_state=0;
                    cou7=0;
                    end
                end
            end
        endcase
    end
```

3）说明：为了实现多楼层同时呼叫电梯时，电梯的运行顺序，可以设置一个变量记录当前电梯的运行状态，当电梯下降时，倾向于下降，上升时，倾向于上升。并按顺序经历每个应该到达的楼层。到达时，输出开门信号，该层的呼叫状态随之清零即可。每次上升或下降一层楼梯时，lift_num 随之加减 1。值得注意的是，连续上楼/下楼的状态需要在没有更高/低层的楼层呼叫电梯，或者到达最高、最低层楼时改变。

1.5　调试过程

（1） always 语句中时间延迟问题

第一次做 verilog 程序时，使用#作为延迟的手段。但是#只可以在仿真过程中使用，无法实现综合并下载至开发板上实现的问题。

例如：

```
case(lift_state)
            0:begin
                if((btn_buff>>lift_num)>0)begin
                  #(5*N);
                  lift_num=lift_num+1;
                  lift_state=1;
                end
```

```
if((btn_buff&(1<<(lift_num-1)))>0)begin
```

```
btn_buff=btn_buff&(~(1<<(lift_num-1)));
                    btn_off=~(1<<(lift_num-1));
                    lift_open=1;
                    #(5*N);
                    lift_open=0;
                    lift_state=0;
                  end

                  if((1<<(lift_num-1))>btn_buff)begin
                    if(btn_buff>0)begin
                      #(5*N);
                      lift_num=lift_num+1;
                      lift_state=2;
                    end
                  end
            end
```

修改的手段：采用了 C 语言类似的思想，设计一个 cou1 做计数功能，比如在 1s 的时钟下延迟 3s,使用 if(cou<3),if(cou==3)两个判断语句来实现。例如：

```
    case(lift_state)
                0:begin
                    if((btn_buff>>lift_num)>0)begin
                    if(cou1<3)//网上没有的方法，类似 C 语言的
```

思想，实现 3s 的延时，后面有类似代码

```
                    begin
                    cou1=cou1+1;
                  end
                  if(cou1==3)
```

```
                    begin
                    lift_num=lift_num+1;
                    clk_num=clk_num+1;
                    lift_state=1;//上层有人叫电梯
                  cou1=0;
                  end
                  end


if((btn_buff&(1<<(lift_num-1)))>0)begin

btn_buff=btn_buff&(~(1<<(lift_num-1)));//本层有人叫电梯
                    btn_off=~(1<<(lift_num-1));
                    lift_open=1;
                    if(cou2<3)
                    begin
                    cou2=cou2+1;
                  end
                  if(cou2==3)
                    begin
                    clk_num=0;
                    lift_open=0;
                    lift_state=0;
                    cou2=0;
                    end
                end
```

（2） 译码器显示问题

由于一开始对译码器原理不甚了解，导致高低电平和片选接反。

例如：case(lift_num)

```
0:dout=11'b1110000_0001;
1:dout=11'b1110_1001111;
2:dout=11'b1110_0010010;
3:dout=11'b1110_0000110;
4:dout=11'b1110_1001100;
5:dout=11'b1110_0100100;
6:dout=11'b1110_0100000;
7:dout=11'b1110_0001111;
8:dout=11'b1110_0000000;
9:dout=11'b1110_0000100;
default:dout=11'b1110_0000001;
endcase
```

修改后：

```
case(clk_num)
0:dout1=11'b0100_1111110;
1:dout1=11'b0100_0110000;
2:dout1=11'b0100_1101101;
3:dout1=11'b0100_1111001;
4:dout1=11'b0100_0110011;
5:dout1=11'b0100_1011011;
6:dout1=11'b0100_1011111;
7:dout1=11'b0100_1110000;
8:dout1=11'b0100_1111111;
9:dout1=11'b0100_1111011;
default:dout1=11'b0100_1111110;
endcase
```

二、 实验结论与总结

2.1 设 计 结 论

本次设计完成了全部要求的LED灯功能模，拟即电电梯梯达可到控楼层，计时功能，时间显示功能加清除。等功能功能

2.2 设 计 心 得 与 总 结

通过参与这次大作业的实验过程，让我了解了数我对数电的操作产生了浓厚的兴趣。

同时，掌握了对逻辑器件的操作，对复杂逻辑关作，增加了我的动手能力，团队合作能力。

最大的收vivado获是的对使用操作，熟悉其代码控制流程线上平台操控开发版实现所需功能。

三、 参 考 文 献

[1]Charles H.Roth, Jr. 著,解 晓,黎萌 永,王志 坤 等译.逻辑设计基础（原书第版）,机械工业,2005.版社

[2]邓 元.数字电路与系统设计（第二版）,西安电子

[3]康 玉 欣 、侯 博 雅 .基于Xilinx Vivado的云数鹏字逻辑,实验教程北京：电子工2018.08出版社，

[4]王 玉.数字逻辑实用教程，2012.9华大学出版社，

[5]李 世 雄，丁 康 源。数字逻辑电子技1993教程。北京

[6]邱 寄 帆。数字逻辑电路。2000京2：清华大学出版社

[7]唐 德 洲。数字电子技术。2002庆8：重庆大学出版社

四、 附录

附录一：含注释的代码

```
module lift_stu(
    input clk,
    input [9:0] btn,
    input btn1,
    output [9:0] nfloor,
    output [10:0] seg,
    output [10:0] seg1,
    output reg lift_open
    );
    parameter N=99_999999;
    reg [9:0] btn_pre_re,btn_buff,btn_off,btn_test;
    reg btn1_pre_re,btn1_off;
    reg clk_200ms;
    reg clk_1s;
    reg clk_3s;
    reg [31:0] count,count1,count3;
    reg [1:0] cou1,cou2,cou3,cou4,cou5,cou6,cou7;
    reg [10:0] dout,dout1;
    reg [1:0] lift_state;
    reg [4:0] lift_num,clk_num;
    initial begin
       btn_off<=9'b111111111;
       btn1_off<=1'b1;
       btn_pre_re<=0;
```

```
        lift_num<=3;
        lift_state<=0;
        lift_open<=0;
    end
     always@(posedge clk)begin   //分频一个 200ms 的时钟，用于按键防抖
         clk_200ms<=0;
         if(count<N/5)
            count<=count+1;
         else begin
            count<=0;
            clk_200ms<=1;
        end
     end
      always@(posedge clk)begin   //分频一个 1s 的时钟，用于电梯运行时钟控制
         clk_1s<=0;
         if(count1<N)
            count1<=count1+1;
         else begin
            count1<=0;
            clk_1s<=1;
        end
     end
      always@(posedge clk_200ms)begin
          btn_pre_re=btn_pre_re^btn;
```

```
        btn_pre_re=btn_pre_re&btn_off;//创新设计：异或门用于
```

按两次按键时进行取消的操作

```
        btn1_pre_re=btn1_pre_re^btn1;//复位按键
        btn1_pre_re=btn1_pre_re&btn1_off;
        if(btn1_pre_re==1)
        begin
        btn_pre_re<=9'b000000010;
        btn1_pre_re=0;
        end
    end
     always@(posedge clk_1s)begin
        btn_buff=btn_pre_re;//btn_pre_re 寄存按键信息
        case(lift_state)
            0:begin
                if((btn_buff>>lift_num)>0)begin
                if(cou1<3)//网上没有的方法，类似 C 语言的思想，实
```

现 3s 的延时，后面有类似代码

```
                  begin
                  cou1=cou1+1;
                end
                if(cou1==3)
                  begin
                  lift_num=lift_num+1;
                  clk_num=clk_num+1;
                  lift_state=1;//上层有人叫电梯
```

```
cou1=0;
end
end

if((btn_buff&(1<<(lift_num-1)))>0)begin
  btn_buff=btn_buff&(~(1<<(lift_num-1)));//
```

本层有人叫电梯

```
  btn_off=~(1<<(lift_num-1));
  lift_open=1;
  if(cou2<3)
  begin
  cou2=cou2+1;
end
if(cou2==3)
  begin
  clk_num=0;
  lift_open=0;
  lift_state=0;
  cou2=0;
  end
end

if((1<<(lift_num-1))>btn_buff)begin//下层有
```

人叫电梯

```
  if(btn_buff>0)begin
    if(cou3<3)
```

```
                begin
                cou3=cou3+1;
              end
              if(cou3==3)
                begin
                  lift_num=lift_num-1;
                  clk_num=clk_num+1;
                  lift_state=2;
                  cou3=0;
                  end
                end
              end
          end

          1:begin
              if((btn_buff>>lift_num)>0)begin
                if((btn_buff&(1<<(lift_num-1)))>0)begin

btn_buff=btn_buff&(~(1<<(lift_num-1)));
                  btn_off=~(1<<(lift_num-1));
                  lift_open=1;
                  if(cou4<3)
                begin
                cou4=cou4+1;
              end
              if(cou4==3)
                begin
```

```
                clk_num=0;
                lift_open=0;
                lift_num=lift_num+1;
                clk_num=clk_num+1;
                cou4=0;
                end
            end
            else
                lift_num=lift_num+1;
                clk_num=clk_num+1;
        end
        else begin

btn_buff=btn_buff&(~(1<<(lift_num-1)));
                btn_off=~(1<<(lift_num-1));
                lift_open=1;
                if(cou5<3)
            begin
            cou5=cou5+1;
        end
        if(cou5==3)
            begin
                clk_num=0;
                lift_open=0;
                lift_state=0;
                cou5=0;
                end
```

```
                end
            end

            2:begin
                btn_test=(btn_buff<<(10-lift_num));
                if(btn_test>0)begin
                  if((btn_buff&(1<<(lift_num-1)))>0)begin//
```

解决下楼命令的其中一个

```
btn_buff=btn_buff&(~(1<<(lift_num-1)));
                    btn_off=~(1<<(lift_num-1));
                    lift_open=1;
                    if(cou6<3)
                  begin
                  cou6=cou6+1;
                end
                if(cou6==3)
                  begin
                      clk_num=0;
                      lift_open=0;
                      lift_num=lift_num-1;
                      clk_num=clk_num+1;
                      cou6=0;
                      end
                  end
                  else
                      lift_num=lift_num-1;
```

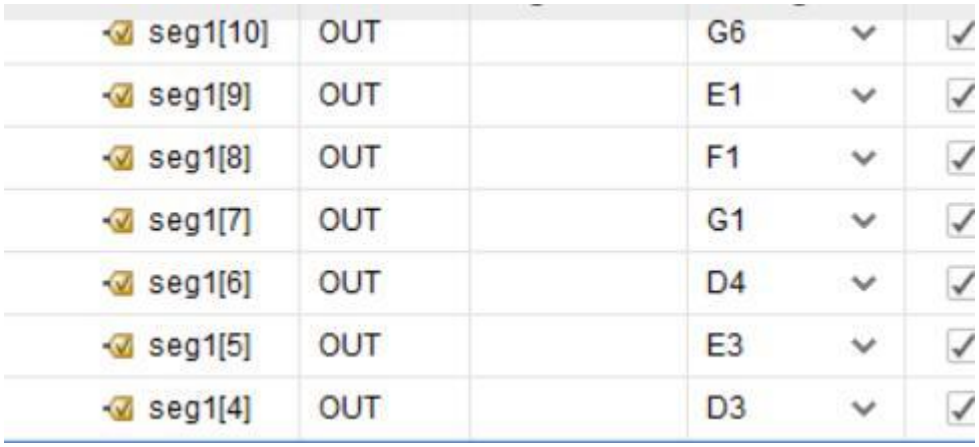
```
                    clk_num=clk_num+1;
                end
                else begin//解决下楼命令的最后一个

btn_buff=btn_buff&(~(1<<(lift_num-1)));
                    btn_off=~(1<<(lift_num-1));
                    lift_open=1;
                    if(cou7<3)
                  begin
                  cou7=cou7+1;
                end
                if(cou7==3)
                  begin
                    clk_num=0;
                    lift_open=0;
                    lift_state=0;
                    cou7=0;
                    end
                  end
              end
           endcase
       end
       always@(posedge clk)begin
       case(lift_num-1)              //记录当前楼层的数码管显
示电路
           0:dout=11'b0100_1111110;
```

```
            1:dout=11'b0100_0110000;
            2:dout=11'b0100_1101101;
            3:dout=11'b0100_1111001;
            4:dout=11'b0100_0110011;
            5:dout=11'b0100_1011011;
            6:dout=11'b0100_1011111;
            7:dout=11'b0100_1110000;
            8:dout=11'b0100_1111111;
            9:dout=11'b0100_1111011;
            default:dout=11'b0100_1111110;
        endcase
    end
    always@(posedge clk)begin           //记录启停时间的时钟的
```

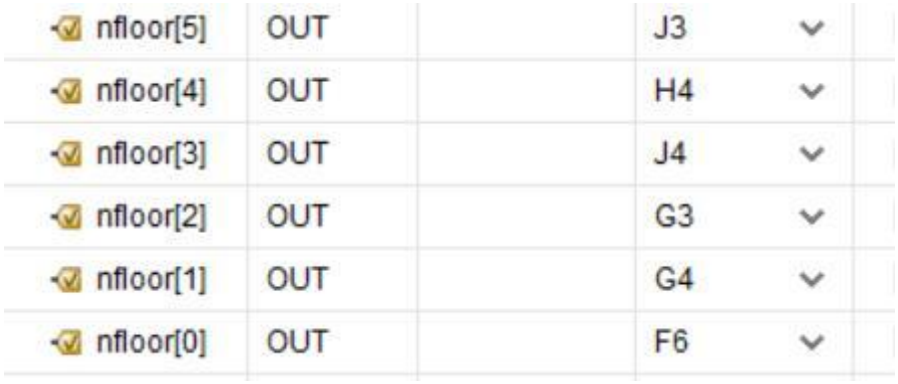
数码管显示电路

```
            case(clk_num)
                0:dout1=11'b0100_1111110;
                1:dout1=11'b0100_0110000;
                2:dout1=11'b0100_1101101;
                3:dout1=11'b0100_1111001;
                4:dout1=11'b0100_0110011;
                5:dout1=11'b0100_1011011;
                6:dout1=11'b0100_1011111;
                7:dout1=11'b0100_1110000;
                8:dout1=11'b0100_1111111;
                9:dout1=11'b0100_1111011;
                default:dout1=11'b0100_1111110;
```

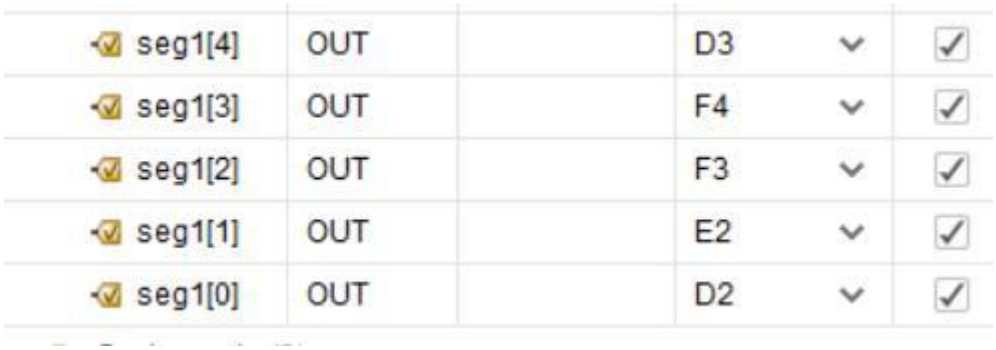
```
            endcase
            end
    assign nfloor=btn_pre_re;
    assign seg=dout;
    assign seg1=dout1;
endmodule
```



seg1[10]	OUT	G6	✓
seg1[9]	OUT	E1	✓
seg1[8]	OUT	F1	✓
seg1[7]	OUT	G1	✓
seg1[6]	OUT	D4	✓
seg1[5]	OUT	E3	✓
seg1[4]	OUT	D3	✓



nfloor[5]	OUT	J3	
nfloor[4]	OUT	H4	
nfloor[3]	OUT	J4	
nfloor[2]	OUT	G3	
nfloor[1]	OUT	G4	
nfloor[0]	OUT	F6	



seg1[4]	OUT	D3	✓
seg1[3]	OUT	F4	✓
seg1[2]	OUT	F3	✓
seg1[1]	OUT	E2	✓
seg1[0]	OUT	D2	✓

btn[3]	IN		V1	
btn[2]	IN		R15	
btn[1]	IN		R17	
btn[0]	IN		R11	
nfloor (10)	OUT			
nfloor[9]	OUT		H6	

seg[5]	OUT		A4	✓
seg[4]	OUT		A3	✓
seg[3]	OUT		B1	✓
seg[2]	OUT		A1	✓
seg[1]	OUT		B3	✓
seg[0]	OUT		B2	✓

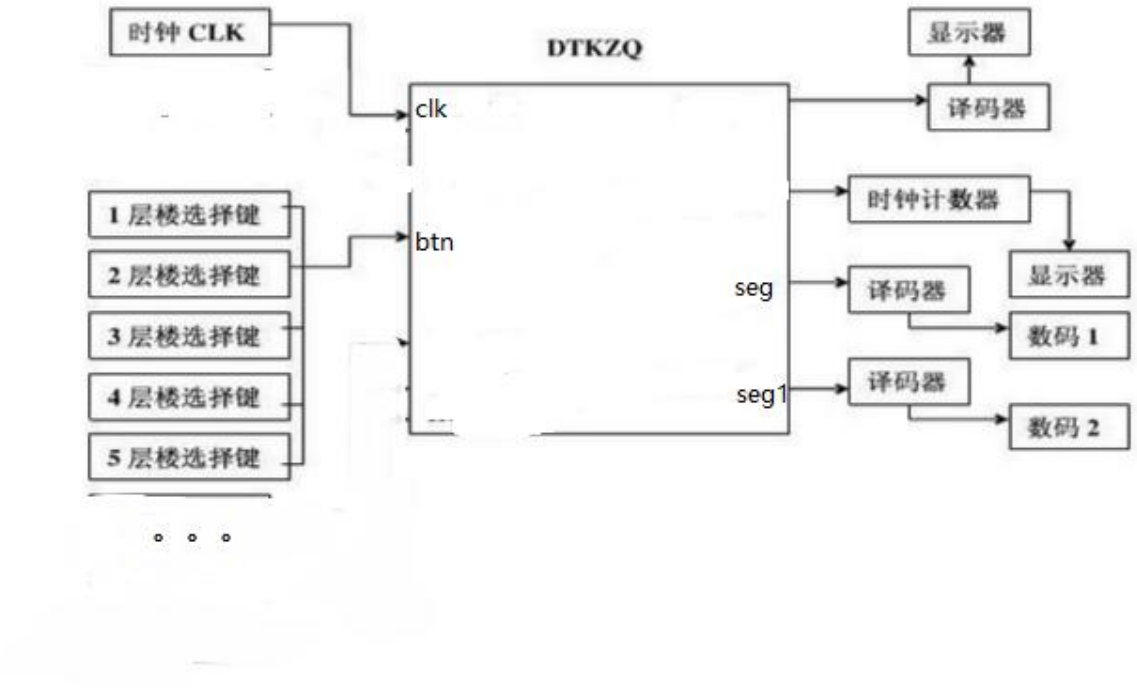
seg[10]	OUT		H1
seg[9]	OUT		C1
seg[8]	OUT		C2
seg[7]	OUT		G2
seg[6]	OUT		B4

nfloor[9]	OUT		H6	✓
nfloor[8]	OUT		K1	✓
nfloor[7]	OUT		K2	✓
nfloor[6]	OUT		J2	✓

Scalar ports (3)			
btn1	IN		R1
clk	IN		P17
lift_open	OUT		K3

btn[9]	IN		R2	✓
btn[8]	IN		P2	✓
btn[7]	IN		P3	✓
btn[6]	IN		P4	✓
btn[5]	IN		P5	✓
btn[4]	IN		U4	✓

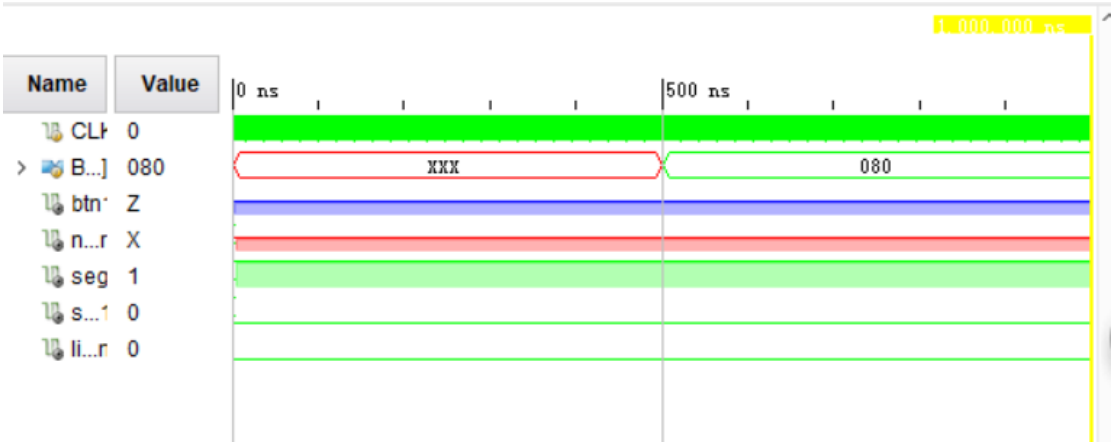
附录二：总体设计图



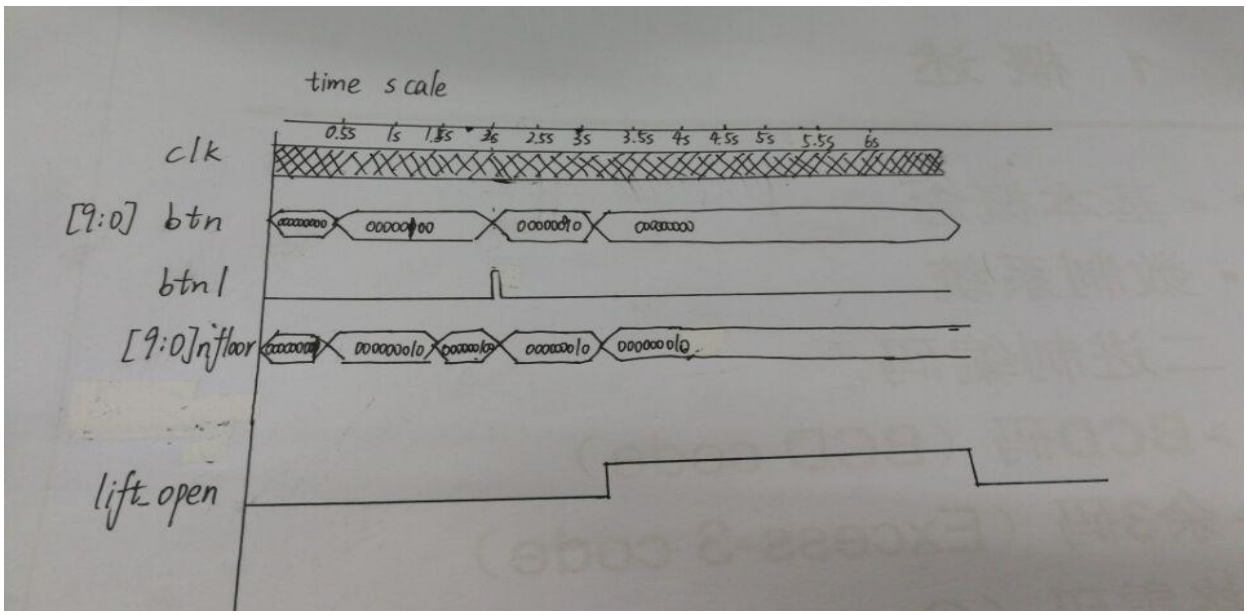
附录三：仿真结果

假设一开始在 0 层在 500ms 的时候按了一下 3 楼按键，然后 2s 之后按复位键，完成复位操作。（此时正在上楼但是立刻变成回到一楼的状态）

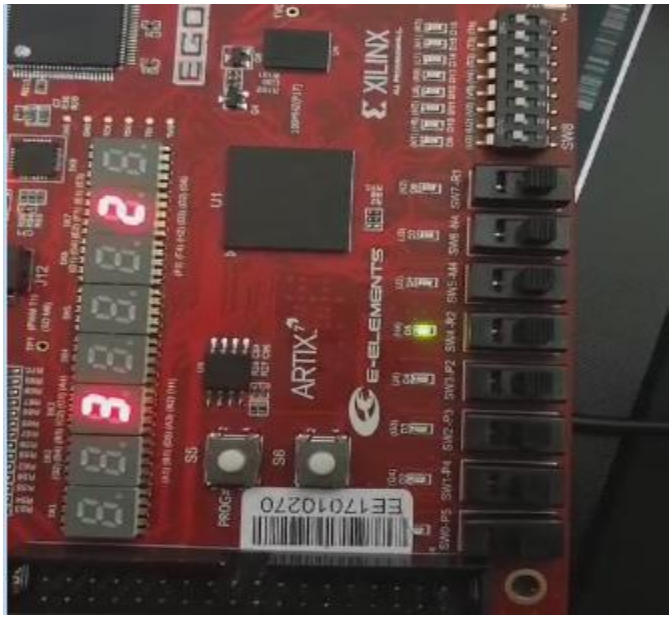
我们先尝试在 vivado 的波形仿真下进行操作，可惜的是，vivado 只能给出 1000ns 以内的波形。



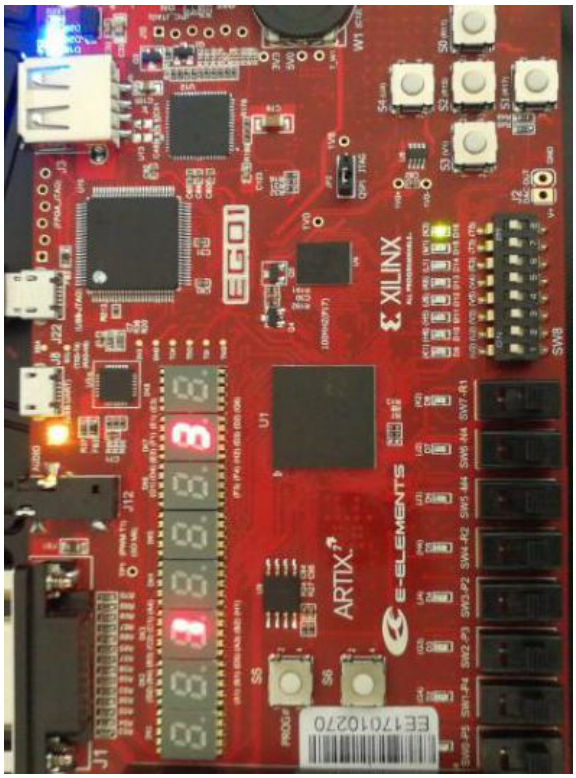
于是，在老师的建议下，我们用手画的方式完成了波形仿真。



下面给出一些开发板运行的照片，作为仿真结果的一部分。



这是正在上楼的操作，可以看到 4 楼的指示灯亮了，电梯正在上升。



这是电梯到达目标楼层，电梯显示了起停时间。开门指示灯亮了，持续 3s。

附 录 四 ： 成 员 工 作 说 明

作 为 小 组 成 员 ， 我 提 出 了 其 中 一 个 创 新 思 路 ，
呼 叫 电 梯 的 处 理 ， 并 撰 写 了 部 分 报 告 （ 即 报 告 的