**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。

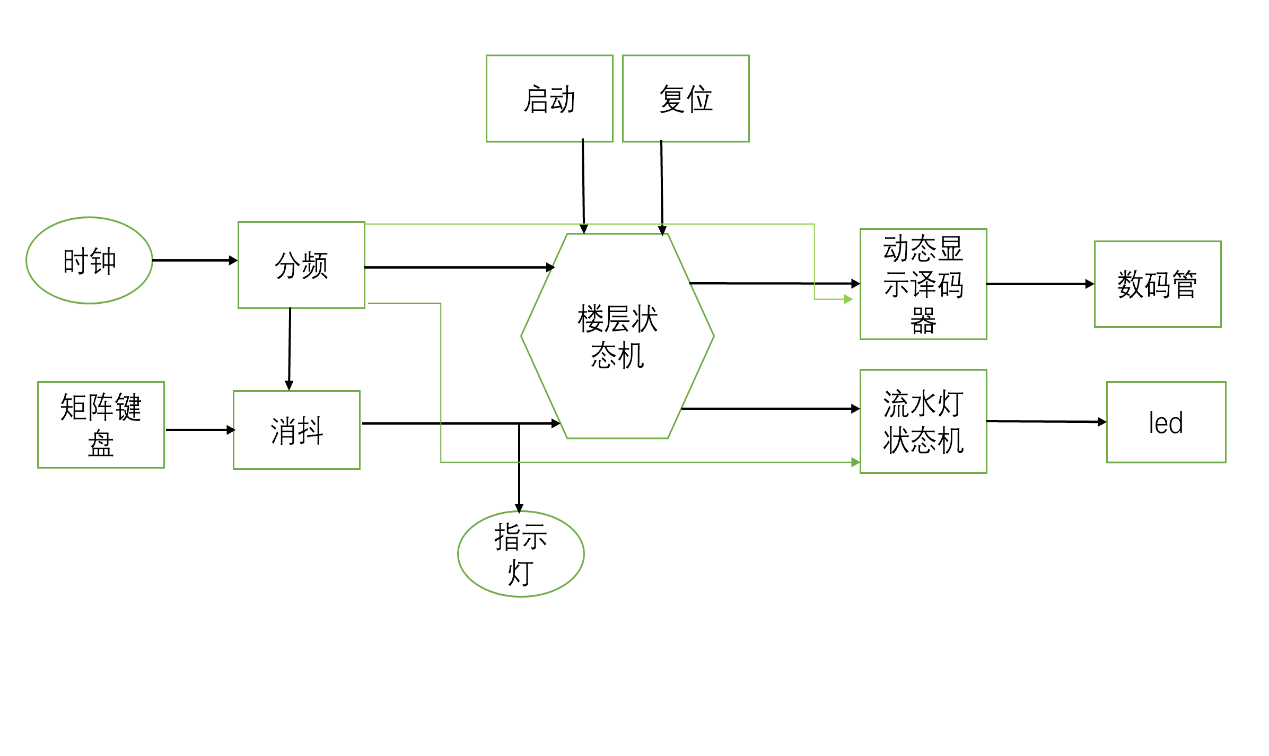
6、增加其他功能可自由发挥。

（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]}]

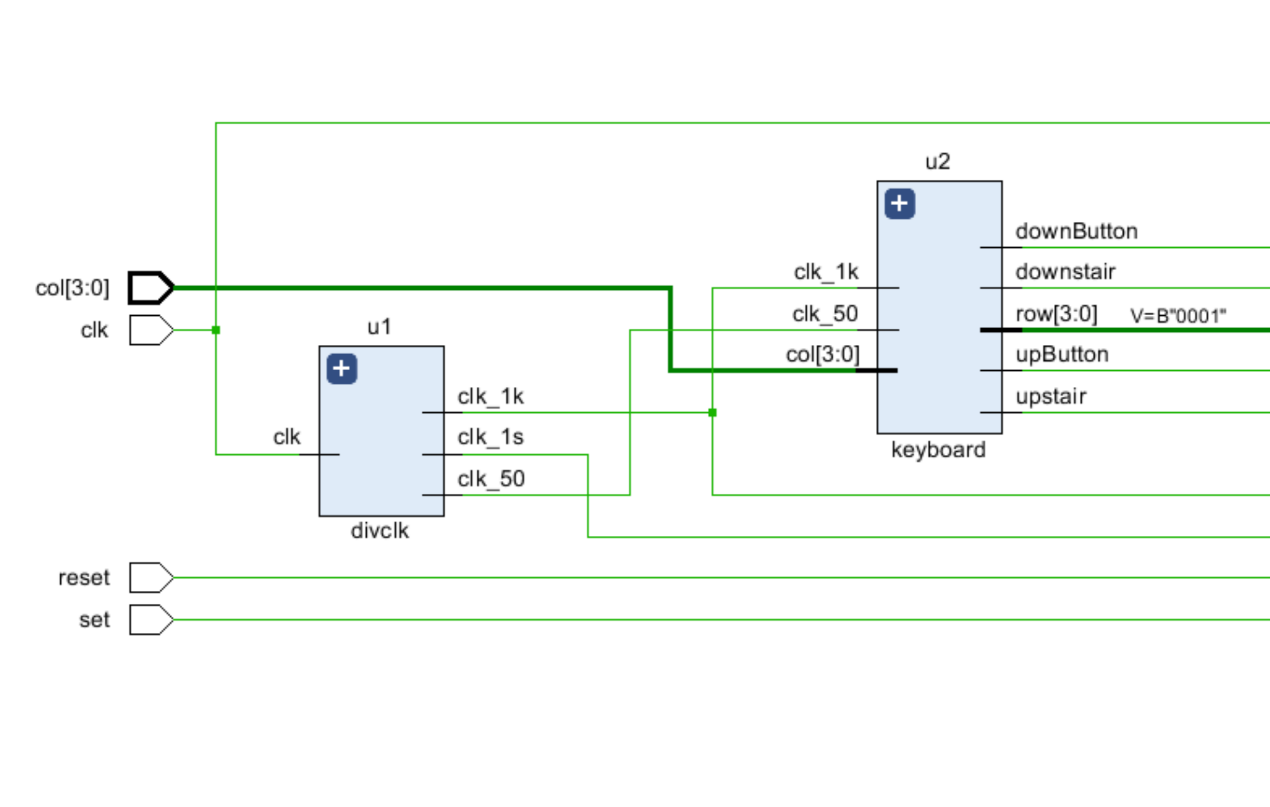
set\_property IOSTANDARD LVCMOS33 [get\_ports {led5[1]}]

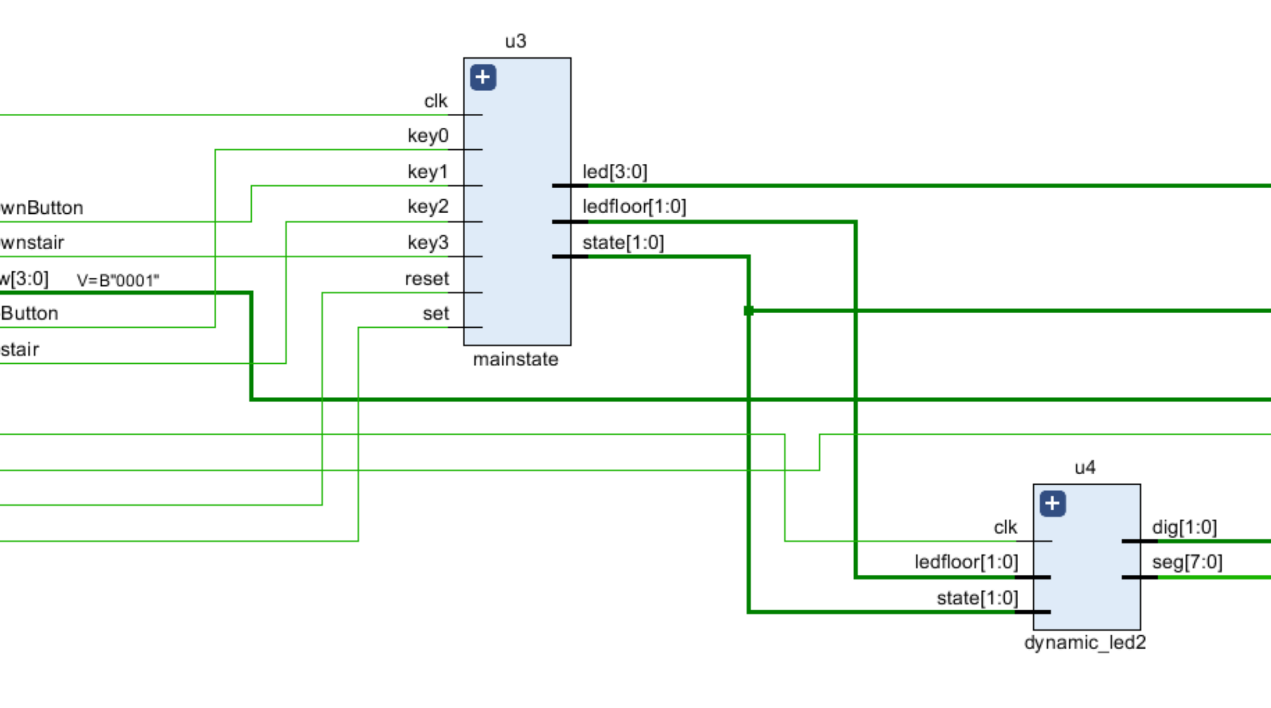
set\_property PACKAGE\_PIN R1 [get\_ports {led5[1]}]

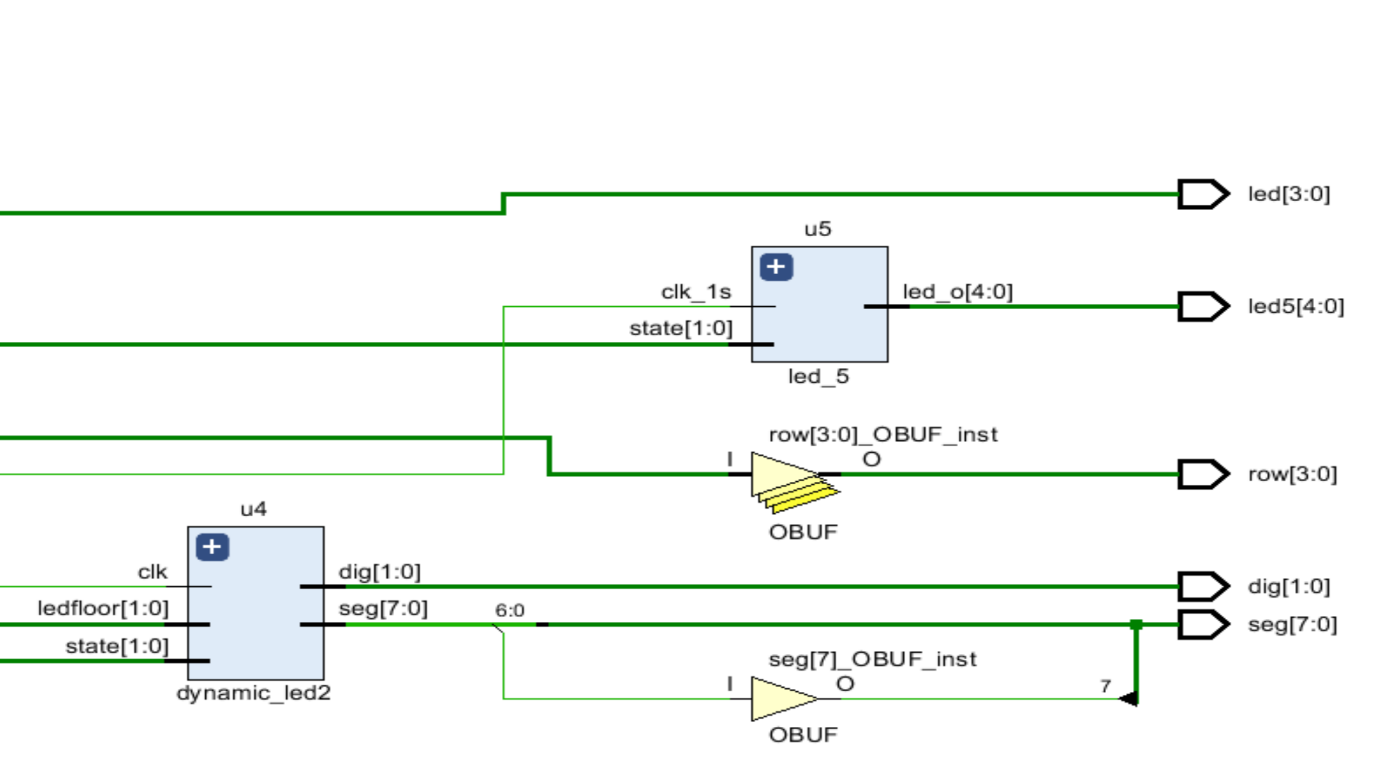
set\_property IOSTANDARD LVCMOS33 [get\_ports {led5[0]}]

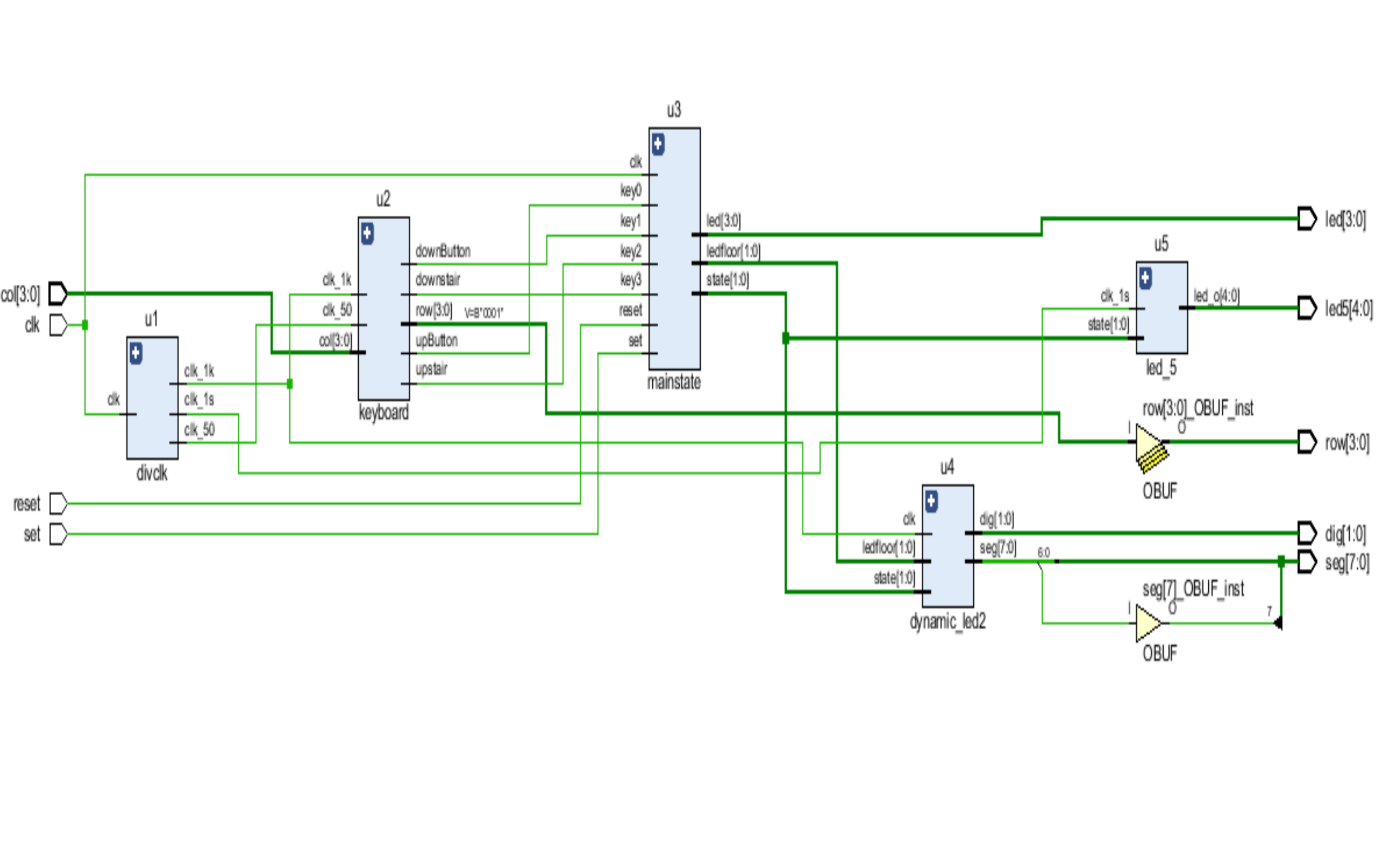
set\_property PACKAGE\_PIN T2 [get\_ports {led5[0]}]

**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。