

(FPGA 部分测试)

实验地点: 科 A339 实验时间: 第 16 周四第七八节 指导老师: **

二、实验学时: 6

1. 基础要求部分

电梯处于1楼，按KEY3，LED3亮，电梯处于上行状态，楼层显示1；运行5秒至2楼后LED3灭，2楼待机。
电梯处于2楼，按KEY2，LED2亮，电梯处于下行状态，楼层显示2；运行5秒至1楼后LED2灭，1楼待机。
电梯处于1楼，按KEY1，LED1亮，电梯处于上行状态，楼层显示1；电梯运行至2楼后LED1灭，2楼待机。
电梯处于2楼，按KEY0，LED0亮，电梯处于下行状态，楼层显示2；运行至1楼后LED0灭，1楼待机。
电梯处于1楼，按KEY0或KEY2，指示灯均不应该亮，电梯1楼待机。
电梯处于1楼，按KEY3，LED3亮，电梯处于上行状态时，立刻按KEY2，LED2亮，电梯继续运行至2楼，LED3灭；然后电梯自动返回1楼待机，LED2灭。
电梯处于1楼；按KEY3，LED3亮，电梯处于上行状态时，立刻按KEY0，LED0亮，电梯继续运行至2楼，LED3灭；然后电梯自动返回1楼待机，LED0灭。
电梯处于1楼，按KEY1或KEY3，让电梯运行至2楼待机；再按KEY1/KEY3，指示灯均不应该亮，电梯继续待机。
电梯处于2楼，按KEY2，LED2亮，电梯处于下行状态时，立刻按KEY3，LED3亮，电梯继续运行至1楼，LED2灭；然后电梯自动返回2楼待机，LED3灭。
电梯处于2楼，按KEY2，LED2亮，电梯处于下行状态时，立刻按KEY1，LED1亮，电梯继续运行至1楼，LED2灭；然后电梯自动返回2楼待机，LED1灭。
将复位开关SW11拨至下方，电梯运行至1楼（运行时间5S）。
将启动开关SW0拨至下方，按任意按键均无反应。

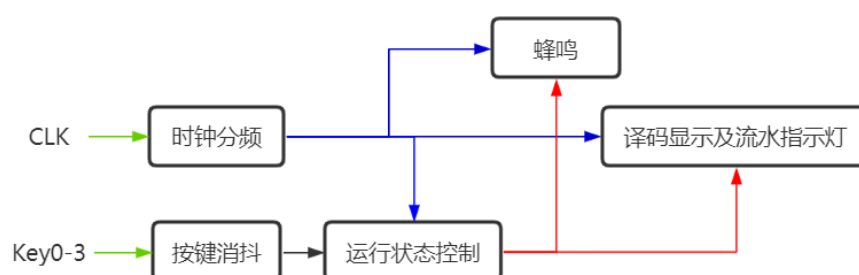
2. 发挥部分

电梯上行时，LED11至LED7五个指示灯从左到右每隔一秒点亮一个；
电梯下行时，LED7至LED11五个指示灯从右到左每隔一秒点亮一个。
电梯运行到达新楼层时，蜂鸣器发出一声清晰“嘀”声。
电梯开始上行或下行时，在最左边两个数码管上倒计时显示运行时间4.9~0.0（秒），精度为0.1秒。到达新楼层时显示0.0（秒）。
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。

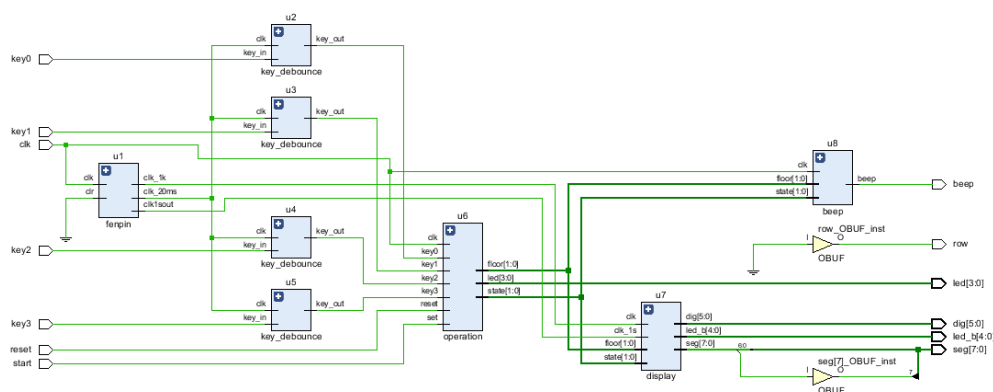
四、实现方案框图

在本次电梯项目中一共分为了五个模块：时钟分频、按键消抖、状态控制、蜂鸣、译码显示及流水指示灯。其模块的作用分别是：

- **时钟分频:** 将高频系统时钟通过分频得到不同合适频率的时钟频率作为不同模块的输入时钟 clk。
- **按键消抖:** 四个按钮 key0~3 的输入, 其中包含了按键消抖的板块防止误触。
- **状态控制:** 处理不同按钮输入的模块, 根据不同的情况分析出电梯该执行的运行状态、楼层情况以及特殊的情况。
- **蜂鸣:** 作为完成附加功能的板块, 在电梯到达新楼层发出嘀声。
- **译码显示及流水指示灯:** 作为显示电梯不同变量的板块, 其中包括位选和段选。完成不同情况的位选和对应的数字的段选。例如电梯的楼层和运行状态, 同时还完成上下行的 LED 依次点亮的附加功能。



五、RTL 分析图



六、实现情况描述

- (1) 完成基础要求任务的全部内容
- (2) 完成附加发挥任务：在上下行的时候流水灯按正确的顺序依次点亮，且在到达楼层后发出嘀的响声。

七、完整呈现及必要注释

见后页附录

代码附录

```
1 //1: 顶层模块=====
2 module top_module(
3     input clk,
4     input start,
5     input reset,
6     input key0,key1,key2,key3,
7     output row,
8     output [3:0]led,
9     output [5:0]dig,
10    output [7:0]seg,
11    output beep,
12    output [4:0]led_b);
13    assign row=1'b0;
14    wire clk_1k;
15    wire clk_1s;
16    wire clk_20ms;
17
18    fenpin u1(.clk(clk),
19             .clk_1k(clk_1k),
20             .clk1sout(clk_1s),
21             .clk_20ms(clk_20ms));
22
23    wire k0;
24    wire k1;
25    wire k2;
26    wire k3;
27    key_debounce u2(
28        .clk(clk_20ms),
29        .key_in(key0),
30        .key_out(k0));
31    key_debounce u3(
32        .clk(clk_20ms),
33        .key_in(key1),
34        .key_out(k1));
35    key_debounce u4(
```

```
36         .clk(clk_20ms),
37         .key_in(key2),
38         .key_out(k2));
39 key_debounce u5(
40     .clk(clk_20ms),
41     .key_in(key3),
42     .key_out(k3));
43 wire [1:0] state;
44 wire [1:0] floor;
45 operation u6(
46     .clk(clk),
47     .set(start),
48     .reset(reset),
49     .key0(k0),
50     .key1(k1),
51     .key2(k2),
52     .key3(k3),
53     .led(led),
54     .state(state),
55     .floor(floor));
56 display u7(
57     .floor(floor),
58     .state(state),
59     .clk(clk_1k),
60     .seg(seg),
61     .clk_1s(clk_1s),
62     .led_b(led_b),
63     .dig(dig));
64 beep u8(
65     .clk(clk),
66     .state(state),
67     .floor(floor),
68     .beep(beep));
69 endmodule
```

```
70
71 //2.分频器=====
72 module fenpin(
73     input clk,clr,
74     output reg clk_1k=0, //1kHz 信号
75     output reg clk1sout=0, //1Hz 信号
76     output reg clk_20ms=0, //50Hz 信号
77     integer    clk1s_cnt=0
78 );
79 reg[24:0] cnt=0;
80 always @ (posedge clk) //分频得到1kHz 信号
81     begin
82         if (cnt==25000)
83             begin
84                 clk_1k=~clk_1k; cnt=0;
85             end
86         else cnt=cnt+1;
87     end
88
89 reg [31:0] temp = 0;
90 always@(posedge clk) //分频得到50Hz 信号
91     begin
92         if(temp == 999999)
93             begin
94                 temp <= 0; clk_20ms <= ~clk_20ms;
95             end
96         else temp <= temp+1;
97     end
98
99 always@(posedge clk) //分频得到1Hz 信号
100 begin
101     if(clr)
102     begin
103         clk1s_cnt<=0; clk1sout<=0;
```

```
104     end
105     else if(clk1s_cnt==24999999)
106         begin
107             clk1s_cnt<=0; clk1sout<=~clk1sout;
108         end
109     else    clk1s_cnt<=clk1s_cnt+1;
110 end
111 endmodule
112
113 //3. 按键消抖模块=====
114 module key_debounce(
115     input key_in,
116     input clk,
117     output key_out);
118     reg btn0=0;
119     reg btn1=0;
120     reg btn2=0;
121
122     always@(posedge clk)
123     begin
124         btn0<=key_in;
125         btn1<=btn0;
126         btn2<=btn1;
127     end
128
129     assign key_out=((btn0&btn1)&(~btn2) | (btn0&btn1&btn2) | ((~btn0)
130         &btn1&btn2));
131 endmodule
132
133 //4. 状态控制模块=====
134 module operation(
135     input clk,
136     input set,
137     input reset,
```

```
137     input key0,
138     input key1,
139     input key2,
140     input key3,
141     output reg [3:0] led=0, //指示灯
142     output reg [1:0] state=0, //0待机状态; 1上行; 2下行
143     output reg [1:0] floor=1 //1一楼; 2二楼
144 );
145 reg [27:0] cnt=0;
146 reg [27:0] clk_count=0;
147 reg k0=0;
148 reg k1=0;
149 reg k2=0;
150 reg k3=0;
151 reg [1:0] first=0;
152 always@(posedge clk)
153 begin
154     if(!reset) //复位开关被拨回
155         begin
156             if(floor==2) //如果电梯在2楼, 则运行五秒后回到一楼
157                 if(clk_count==24999999) //5s计时
158                     begin
159                         clk_count=0;
160                         floor=1; //1楼
161                         state=0; //待机状态
162                         led=4'b0000; //指示灯不亮
163                     end
164                 else
165                     begin
166                         clk_count=clk_count+1;
167                         state=2; //下行状态
168                         led=4'b0000; //指示灯不亮
169                     end
170             end
171         end
172     end
```

```
171     else if(set)//复位开关未被拨下, 且启动start有效
172     begin
173         if(led)//当led非0, 即电梯处于运行状态时
174         begin
175             if(clk_count==249999999)//5s计时
176             begin
177                 clk_count=0;
178                 if(floor==1)
179                     begin floor=floor+1;
180                     end//计时五秒后从一楼切换到二楼
181                 else
182                     begin floor=floor-1;
183                     end//计时五秒后从二楼切换到一楼
184
185                 //key3, key2接连被按下, 到二楼后保持led2亮起, 切换到
186                 下行状态
187                 if((k3&k2)&&first==1)
188                     begin state=2;k3=0;k2=0;led[3]=0;first=0;
189                     end
190
191                 //key3, key0接连被按下, 到二楼后保持led0亮起, 切换到
192                 下行状态
193                 else if(k3&k0)
194                     begin state=2;k3=0;k0=0;led[3]=0;
195                     end
196
197                 //key2, key3接连被按下, 到一楼后保持led3亮起, 切换到
198                 上行状态
199                 else if((k2&k3)&&first==2)
200                     begin state=1;k2=0;k3=0;led[2]=0;first=0;
201                     end
202
203                 //key2, key1接连被按下, 到一楼后保持led1亮起, 切换到
204                 上行状态
```



```
201         else if(k2&k1)
202             begin state=1;k2=0;k1=0;led[2]=0;
203             end
204
205         //回到待机状态
206         else
207             begin state=0;led=4'b0000;k0=0;k1=0;k2=0;k3=0;
208             end
209         end
210
211         else clk_count=clk_count+1;
212         //在运行过程中保持指示灯正常亮起，并且在按键在电梯运行过程中
213         //被按下时进行记录
214         if(key2|k2)
215             begin k2=1;k0=0;led[2]=1;
216             end
217         if(key0|k0)
218             begin k0=1;k2=0;led[0]=1;
219             end
220         if(key3|k3)
221             begin k3=1;k1=0;led[3]=1;
222             end
223         if(key1|k1)
224             begin k1=1;k3=0;led[1]=1;
225             end
226         end
227
228         else if((key0)&&floor==2) //当前在二楼，一楼按key0
229             begin
230                 led[0]=1;
231                 state=2; //下行
232             end
233
234         else if((key1)&&floor==1) //当前在一楼，二楼按key1
```

```
234         begin
235             led[1]=1;
236             state=1; //上行
237         end
238
239     else if((key2)&&floor==2) //当前在二楼 电梯内按下
240         begin
241             led[2]=1;
242             state=2; //下行
243             k2=1;
244             first=2; //用于表明key2先于key3被按下
245         end
246
247     else if((key3)&&floor==1) //当前在一楼 电梯内按下
248         begin
249             led[3]=1;
250             state=1; //上行
251             k3=1;
252             first=1; //用于表明key3先于key2被按下
253         end
254     end
255 end
256 endmodule
257
258 //5. 译码显示及流水指示灯模块=====
259
260 module display(
261     input [1:0] floor, //楼层
262     input [1:0] state, //状态
263     input clk,
264     input clk_1s,
265     output reg [4:0] led_b=0,
266     output reg [7:0] seg,
267     output reg [5:0] dig
```

```
268 );
269 reg num=0;
270     always@(posedge clk)
271     begin
272         if (num==1) num=0;
273         else num=num+1;
274     end
275 //位选
276     always@(num)
277     begin
278         case(num)
279             0:dig=6'b111110;
280             1:dig=6'b111101;
281             default: dig=0;
282         endcase
283     end
284
285 always@(posedge clk_1s or posedge state or negedge state)
286 begin
287     if(state==0)
288         led_b[4:0]=5'b00000;
289     else if (state ==1)//电梯处于上行状态
290     begin
291         if (led_b[4:0]==5'b00000)
292             led_b[4:0]=5'b10000;
293         else
294             led_b[4:0]=led_b[4:0]>>1;
295         //向右移一位。10000-01000-00100-00010-00001-00000
296     end
297     else if (state ==2)//电梯处于下行状态
298     begin
299         if (led_b[4:0]==5'b00000)
300             led_b[4:0]=5'b00001;//到了00000,下一个是00001
301         else
```

```
302         led_b[4:0]=led_b[4:0]<<1;
303         //向左移一位。00001-00010-00100-01000-10000-00000
304         end
305         else led_b[4:0]=5'b00000;
306         end
307
308     //选择器，确定显示数据
309     reg [3:0] disp_data;
310     always@(num)
311     begin
312         case(num)
313             0:disp_data=floor;
314             1:disp_data=state+3'b011;//3待机 4上行 5下行
315             default: disp_data=0;
316         endcase
317     end
318     //显示译码器
319     always@(disp_data)
320     begin
321         case(disp_data)
322             4'h1: seg=8'h06;//1楼
323             4'h2: seg=8'h5b;//2楼
324             4'h3: seg=8'h40;//待机
325             4'h4: seg=8'h01;//上行
326             4'h5: seg=8'h08;//下行
327             default: seg=0;
328         endcase
329     end
330 endmodule
331
332 //6.蜂鸣=====
333 module beep(
334     input clk,
335     input [1:0]state,
```

```
336     input [1:0] floor,
337     output reg beep
338 );
339 reg beep_n; //控制蜂鸣声时长
340 reg [24:0] beep_cnt=0;
341 reg [24:0] clk_500ms_cnt=0;
342 reg [24:0] clk_500ms_cnt2=0;
343     always @ (posedge clk)
344     begin
345         if(floor==1)
346             begin
347                 clk_500ms_cnt2=0;
348                 if (clk_500ms_cnt==24999999) //计数0.5s, 在此区间
349                     发出蜂鸣声
350                     begin
351                         beep_n=0;
352                         clk_500ms_cnt=24999999;
353                     end
354                 else if (clk_500ms_cnt!=24999999 && clk_500ms_cnt
355                     !=0)
356                     begin
357                         clk_500ms_cnt<=clk_500ms_cnt+1;
358                         beep_n=1;
359                     end
360                 else if (clk_500ms_cnt==0)
361                     begin
362                         clk_500ms_cnt<=clk_500ms_cnt+1;
363                         beep_n=0;
364                     end
365             end
366         else
367             begin
368                 clk_500ms_cnt=0;
369                 if (clk_500ms_cnt2==24999999)
```

```
368         //计数0.5s, 在此区间发出蜂鸣声
369         begin
370             beep_n=0;
371             clk_500ms_cnt2=24999999;
372         end
373     else if(clk_500ms_cnt2!=24999999&&clk_500ms_cnt2
374             !=0)
375         begin
376             clk_500ms_cnt2<=clk_500ms_cnt2+1;
377             beep_n=1;
378         end
379     else if(clk_500ms_cnt2==0)
380         begin
381             clk_500ms_cnt2<=clk_500ms_cnt2+1;
382             beep_n=0;
383         end
384     end
385 if(beep_n==1)
386     begin
387         if(beep_cnt==49000)
388             //向蜂鸣器输入一个约500Hz的脉冲, 使蜂鸣器发声
389             begin
390                 beep=~beep;
391                 beep_cnt=0;
392             end
393         else
394             begin beep_cnt=beep_cnt+1; end
395         end
396     else
397         begin
398             beep=0;
399             beep_cnt=0;
400         end
end
```

```
401         end
402     endmodule
403
404     // 约束文件=====
405     set_property IOSTANDARD LVCMOS33 [get_ports {dig[5]}]
406     set_property IOSTANDARD LVCMOS33 [get_ports {dig[4]}]
407     set_property IOSTANDARD LVCMOS33 [get_ports {dig[3]}]
408     set_property IOSTANDARD LVCMOS33 [get_ports {dig[2]}]
409     set_property IOSTANDARD LVCMOS33 [get_ports {dig[1]}]
410     set_property IOSTANDARD LVCMOS33 [get_ports {dig[0]}]
411     set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
412     set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
413     set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
414     set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
415     set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
416     set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
417     set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
418     set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
419     set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
420     set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
421     set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
422     set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
423     set_property IOSTANDARD LVCMOS33 [get_ports beep]
424     set_property IOSTANDARD LVCMOS33 [get_ports clk]
425     set_property PACKAGE_PIN T5 [get_ports {led[3]}]
426     set_property PACKAGE_PIN R7 [get_ports {led[2]}]
427     set_property PACKAGE_PIN R8 [get_ports {led[1]}]
428     set_property PACKAGE_PIN P9 [get_ports {led[0]}]
429     set_property PACKAGE_PIN D4 [get_ports clk]
430     set_property PACKAGE_PIN L2 [get_ports beep]
431     set_property PACKAGE_PIN N11 [get_ports {dig[5]}]
432     set_property PACKAGE_PIN N14 [get_ports {dig[4]}]
433     set_property PACKAGE_PIN N13 [get_ports {dig[3]}]
434     set_property PACKAGE_PIN M12 [get_ports {dig[2]}]
```

```
435 set_property PACKAGE_PIN H13 [get_ports {dig[1]}]
436 set_property PACKAGE_PIN G12 [get_ports {dig[0]}]
437 set_property PACKAGE_PIN L13 [get_ports {seg[7]}]
438 set_property PACKAGE_PIN M14 [get_ports {seg[6]}]
439 set_property PACKAGE_PIN P13 [get_ports {seg[5]}]
440 set_property PACKAGE_PIN K12 [get_ports {seg[4]}]
441 set_property IOSTANDARD LVCMOS33 [get_ports reset]
442 set_property IOSTANDARD LVCMOS33 [get_ports start]
443 set_property PACKAGE_PIN F3 [get_ports reset]
444 set_property PACKAGE_PIN T9 [get_ports start]
445 set_property PACKAGE_PIN K13 [get_ports {seg[3]}]
446 set_property PACKAGE_PIN L14 [get_ports {seg[2]}]
447 set_property PACKAGE_PIN N12 [get_ports {seg[1]}]
448 set_property PACKAGE_PIN P11 [get_ports {seg[0]}]
449 set_property PACKAGE_PIN K3 [get_ports row]
450 set_property IOSTANDARD LVCMOS33 [get_ports row]
451 set_property IOSTANDARD LVCMOS33 [get_ports key0]
452 set_property IOSTANDARD LVCMOS33 [get_ports key1]
453 set_property IOSTANDARD LVCMOS33 [get_ports key2]
454 set_property IOSTANDARD LVCMOS33 [get_ports key3]
455 set_property PACKAGE_PIN R12 [get_ports key0]
456 set_property PACKAGE_PIN T12 [get_ports key1]
457 set_property PACKAGE_PIN R11 [get_ports key2]
458 set_property PACKAGE_PIN T10 [get_ports key3]
459 set_property IOSTANDARD LVCMOS33 [get_ports {led_b[4]}]
460 set_property IOSTANDARD LVCMOS33 [get_ports {led_b[3]}]
461 set_property IOSTANDARD LVCMOS33 [get_ports {led_b[2]}]
462 set_property IOSTANDARD LVCMOS33 [get_ports {led_b[1]}]
463 set_property IOSTANDARD LVCMOS33 [get_ports {led_b[0]}]
464 set_property PACKAGE_PIN T2 [get_ports {led_b[0]}]
465 set_property PACKAGE_PIN R1 [get_ports {led_b[1]}]
466 set_property PACKAGE_PIN G5 [get_ports {led_b[2]}]
467 set_property PACKAGE_PIN H3 [get_ports {led_b[3]}]
468 set_property PACKAGE_PIN E3 [get_ports {led_b[4]}]
```