北京郵電大學

数字系统设计实验报告

题目：简易快递自提柜系统（用户端）的设计与实现

姓　　名　廖云菱

学　　院　信息与通信工程学院

班　　级　2020211116

学　　号　2020210482

2021 年 12 月

摘要

本实验使用Verilog HDL语言，利用FPGA数字实验电路板设计实现简易快递自提柜用户端系统，模拟包裹自取的基本功能。主要实现了发光二极管指示货箱状态、4×4小键盘输入取件码、数码管显示输入的取件码、8×8点阵显示开关门动画、蜂鸣器提示相关操作五大功能。

关键词：Verilog FPGA 简易快递自提柜 用户端

目录

一、任务要求

1、基本要求：

1、本快递自提柜的容量为 8 个货箱（分别用发光二极管 LD0 - LD7 表示），用发光二极管 LD0 - LD7 的亮灭来指示 8 个货箱的状态，对应的发光二极管点亮表示该货箱已被占用，熄灭表示货箱是空置的；

2、 BTN0 - BTN7 分别代表数字 0~7，收货人自取货物时，用 BTN0 - BTN7 输入取件码（假设 8 个货箱的取件码分别为 000~007），并显示在数码管 DISP2 - DISP0 上。取件码输入正确则对应的发光二极管 LD（i）闪烁且对应的货箱开门（点阵演示货箱开门动画），3 秒后自动关闭货箱门（点阵演示货箱关门动画），对应的发光二极管 LD（i）熄灭表示该货箱已空出，可以再次被使用。如果取件码输入有误，对应的货箱不会开门，同时在数码管 DISP0 上显示 E 提示取件码输入有误，需要重新输入取件码；

3、 货箱的开门和关门动画自拟，尽可能形象；

4、 系统上电时，8 个货箱应都是满的，对应的发光二极管点亮；

5、 系统工作流程合理，工作稳定。

2、提高要求：

1、 用 4*4 小键盘输入取件码；

2、 为相关操作设计音效；

3、 自拟其他功能。

二、系统设计

（一）设计思路

本系统采用模块化电路设计的思路，尽量将电路板上的每一个功能模块单独编写一个文件，以做到代码清晰、可读性强，且不容易产生always块间的逻辑冲突。其中，由于4×4键盘需要进行状态机扫描、按键消抖和输出判断，操作较多，代码量较大，因此最后编写分为三个文件，即按键消抖模块（debounce）、键盘状态机扫描模块（Keyboard）和接收输入信息并向其他模块发出控制信号的操作中心（Operator）。

最后，共编写了顶层（ParcelLocker）、时钟分频（ClockDivide）、8×

8点阵（Screen）、数码管（DISP）、LED（LD）、4×4键盘（Keyboard）、按键消抖（debounce）、操作中心（Operator）、蜂鸣器（Beep）九个模块，然后在顶层模块通过一次或多次例化，完成总体系统的设计。

（二）总体框图

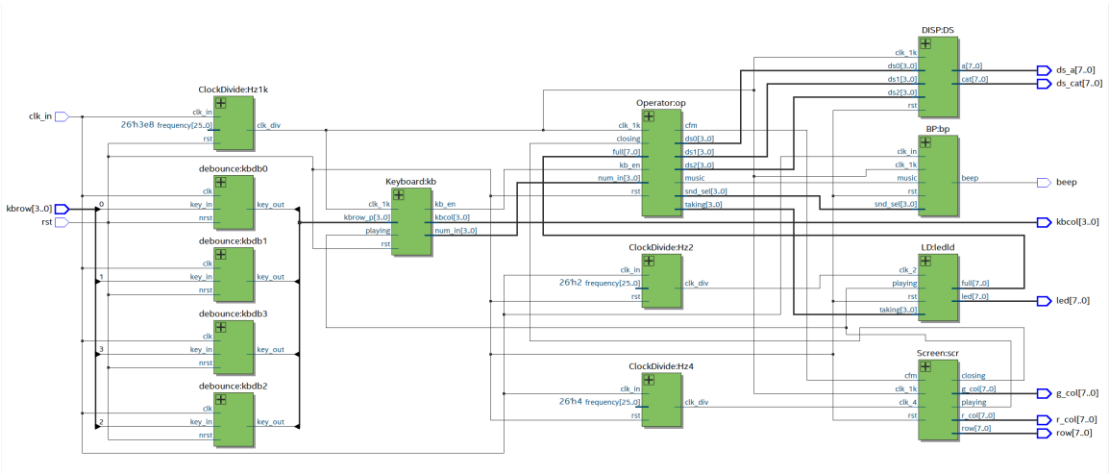

图 1 总体设计框图

（三）分块设计

1、顶层模块

顶层模块仅进行输入输出端口的定义、wire 型变量的定义以及各模块的例化，除此之外，不进行任何操作，相当于一个用于放置组件和连线的空白面包板。这样设计可以极大程度地简化仿真过程，仅需将 input 和 output 改为 reg 和 wire，再对原 input 改成的 reg 进行赋值，即可对所有模块进行总体仿真。

2、4×4 键盘模块

该模块完成键盘的状态机扫描，行信号输入的是消抖后的信号。键盘采用 1kHz 时钟进行列扫描，在某列扫描到行信号按下后，立即停止扫描，锁定键盘，计算输出的值，并通过使能脉冲使信号在按键按下一次后仅输出一次。

3、蜂鸣器模块

模块使用系统时钟频率 50MHz 来进行音高计数，保证音高的准确。使用 1kHz 进行时值的计数，使音符可以持续不同长度的时间。这样就能演奏出不同的音乐。

4、操作中心模块

操作中心类似 CPU，是连接用户输入和其他输出模块的中间模块。主要负责接收用户输入的数据，并判断相应的操作，再对数码管、蜂鸣器、点阵的值进行修改。

5、8×8 点阵模块

点阵使用 1kHz 时钟动态扫描，利用人眼视觉暂留效果，闪烁显示每一行的图案，来达到显示整个画面的目的；4Hz 时钟画面切换，使其能播放动画。

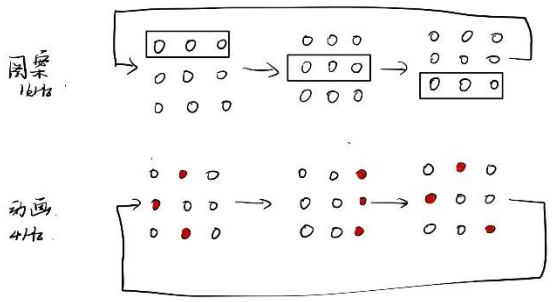

图 2 点阵显示原理图

6、数码管模块

数码管使用 1kHz 时钟动态扫描，同样是利用人眼的视觉暂留，闪烁显示三个数码管。

7、时钟分频模块

时钟分频模块内部设置计数器，并设定默认系统时钟频率为 50MHz，设置一个输入端口输入分频后的时钟频率，通过多次例化来获得不同频率的时钟。时钟分频的原理是利用时钟在计数器某个值时进行翻转，达到降低时钟频率的目的

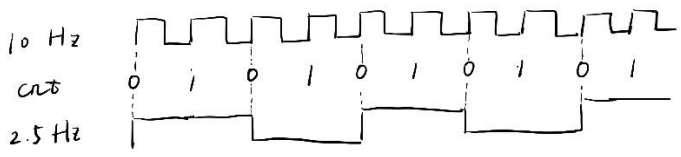

图 3 时钟分频原理图

8、LED 模块

LED 模块控制 LED 在快递柜满时常亮、动画播放时闪烁、播放完成后熄灭且不会再亮起，直到重置信号到来。

9、按键消抖模块

由于按键会存在抖动现象，所以需要延迟 20ms 左右再次检测按键的状态来判断按键是否按下。20ms 通过计数器实现，不同的时钟对应不同的大小，通过 parameter 型设置了默认值，可在例化时修改。

三、仿真波形及波形分析

1、顶层模块仿真

使用以下代码进行顶层模块仿真，模拟以下操作：1、从初始状态，将 rst 拨码开关由关变为开；2、按下某个键盘按键后抬起。得到图 4 所示的波形。由于整体仿真代码过长，已略去输入输出和变量定义和各模块例化，仅保留了主要操作部分。

```
1.  initial  begin
2.    clk_in = 0;
3.    #1_000_000;
4.    rst=1;
5.    #100_000_000;
6.    kbrow=4'b1101;
7.    #100_000_000;
8.    kbrow=4'b1111;
9.    #1_000_000;
10.   $stop;
11. end
12. always #1 clk_in = ~clk_in;
```

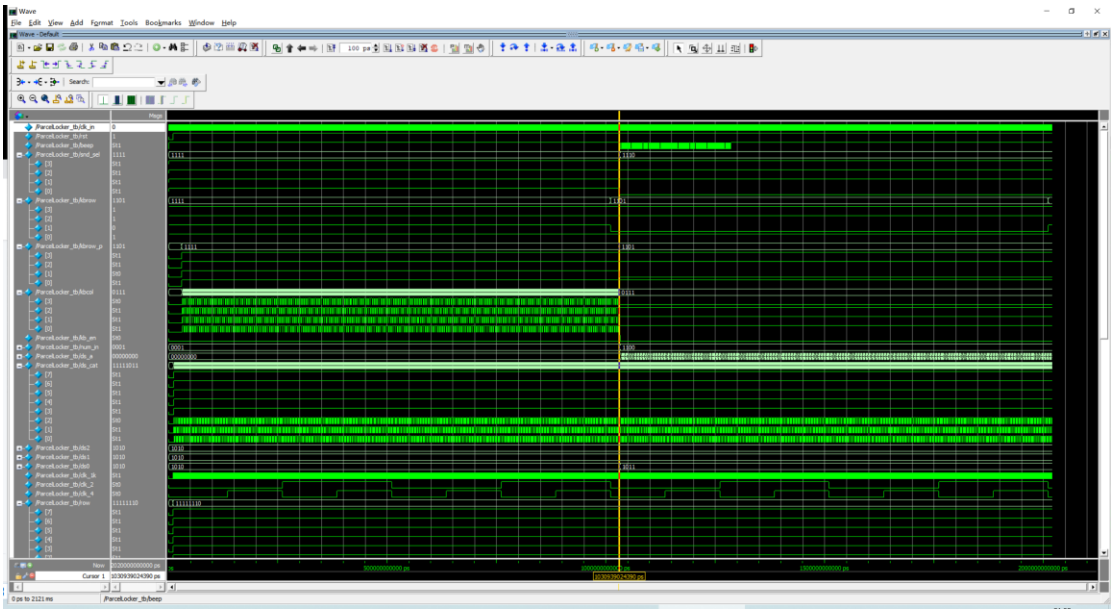

图 4 顶层模块仿真总体结果

对仿真结果进行分析可以发现，按键消抖后的 kbrow_p 略滞后于 kbrow 出现。而蜂鸣器 beep 的振荡、音乐选择值 snd_sel 的变化、数码管显示 ds_a 的变化、键盘列扫描 kbcol 的停止等，均出现在 kbrow_p 的变化之后，而非 kbrow 变化之后，成功地完成了按键消抖和对各组件的控制。

2、蜂鸣器模块仿真

操作中心对蜂鸣器的控制主要有两个部分，音乐选择 snd_sel 和音乐播放 music。图 5 中可以看到，music 仅为检测到输入键盘有效信号 kb_en 后持续 1ms 的小脉冲，当蜂鸣器检测到这个信号后，音符计数器将由停止处跳转到第一个音符处，音乐就开始播放了。蜂鸣器振动频率是由系统最高频的 50MHz 时钟算出的，因为使用较高频率的时钟可以振动频率更接近音高频率，音高也就更准。

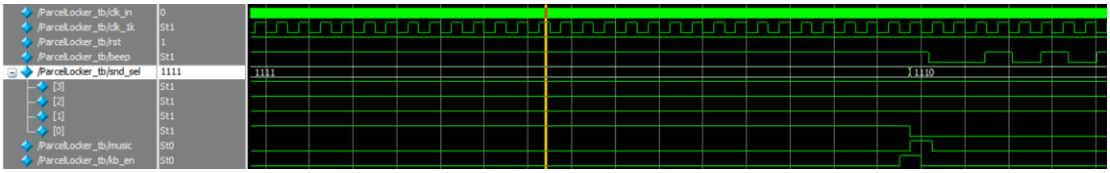

图 5 蜂鸣器模块仿真结果

3、4×4 键盘模块仿真

如图 6，4×4 键盘使用了状态机扫描的方法，使用 1kHz 的时钟对键盘进行列扫描。在检测到按键按下后，为保证信号的稳定，需要立即停止列扫描，标记当前按下的键的坐标，计算按下的数值，然后输出。同时，由于用户不一定会在

按下后立马将手抬起，为保证信号不会重复输出，需要键盘产生一个使能脉冲 kb_en 标识一次输入的值有效，而其他重复的输入都无效。该使能脉冲仅持续一个时钟周期，所以数据仅会在该时钟周期内被外部接收端接收。

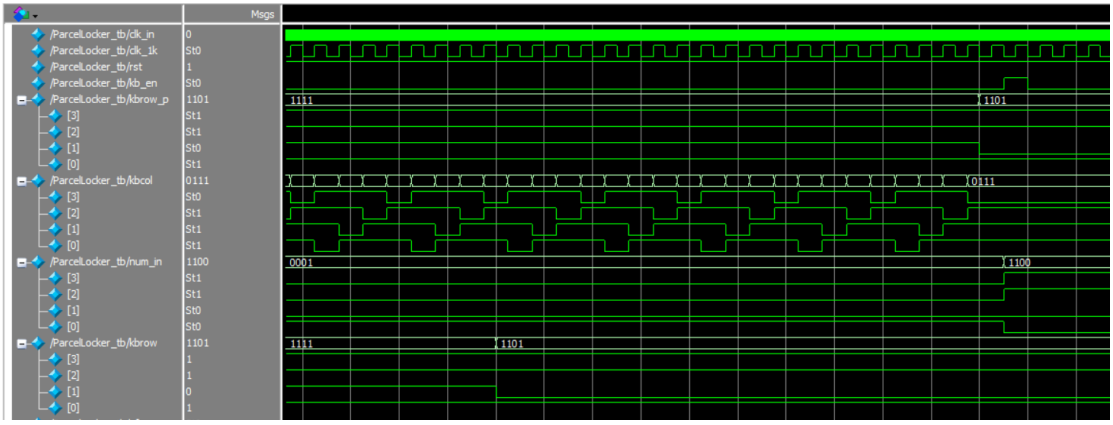

图 6 4×4 键盘模块仿真结果

4、数码管模块仿真

如图 7 是数码管模块仿真结果。数码管输入为 ds2、ds1、ds0，分别对应 2、1、0 位的数码管数值。图中，num_in 是操作中心 Operator 接收到来自键盘输入的数据，经逻辑判断，操作中心会更改 ds2、ds1、ds0 的数值，使数码管显示的内容发生变化。此处可以发现，num_in 的值要滞后于 ds0，这是由于 num_in 采用了时钟上升沿检测来赋值，而 ds0 采用了时钟下降沿。这么处理的原因是多位数据的变化会产生非常短时的不稳定现象，如果分别使用上升沿和下降沿赋值，将会使其中的赋值过程延缓半个时钟周期，使赋值更为稳定，不会出错。

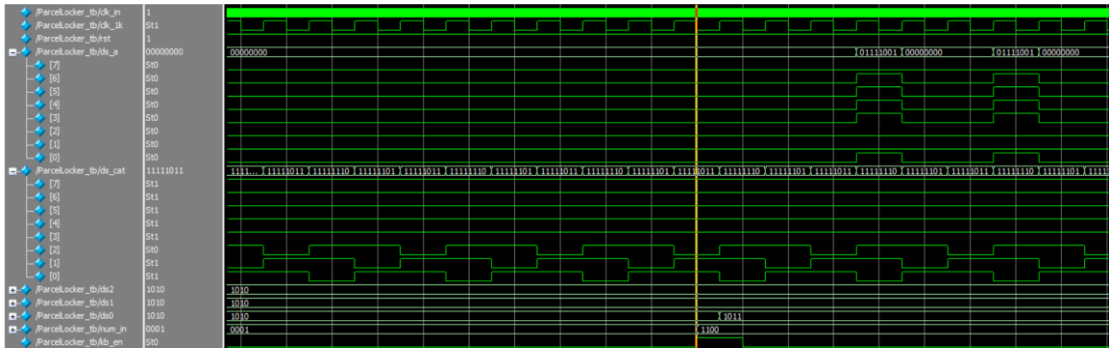

图 7 数码管模块仿真结果

5、8×8 点阵模块仿真

如图 8、图 9 为 8×8 点阵的仿真结果。8×8 点阵采用行扫描的方式，利用人的视觉暂留现象，以 1kHz 的频率逐行闪烁，以达到在整个点阵显示图案的目的。再采用 4Hz 时钟切换图案，达到显示动画的效果。同时，点阵还需要使正在播放信号 playing 变为高电平来锁定键盘 Keyboard，防止在播放动画时用户再进行输入操作，以及需要在播放关门动画时发出一个时钟周期的信号指示蜂鸣器播放关门音乐。

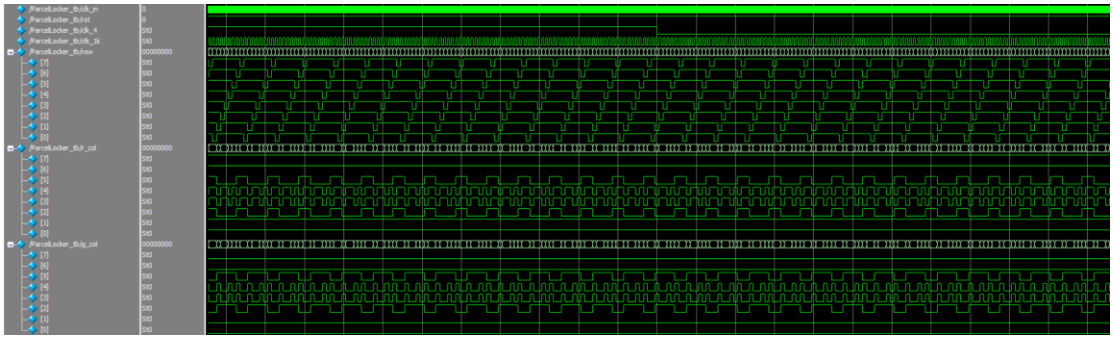

图 8 8×8 点阵模块仿真结果 1

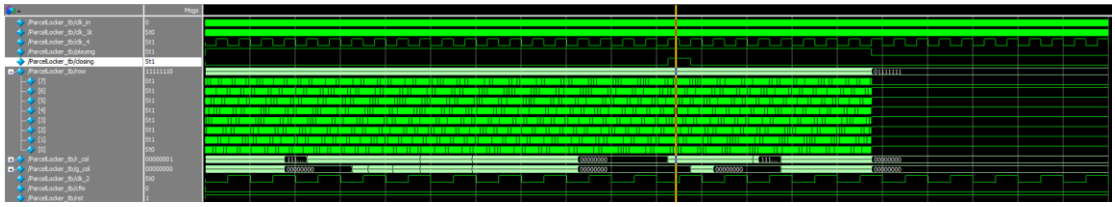

图 9 8×8 点阵模块仿真结果 2

6、时钟分频模块仿真

如图 10 为时钟分频模块仿真结果。本次设计使用了 1kHz、4Hz、2Hz 的时钟。

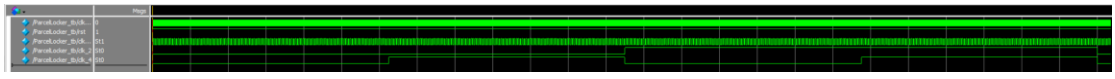

图 10 时钟分频模块仿真结果

四、代码

1、顶层模块文件

```
1. module ParcelLocker(
2.                 input     clk_in,  //系统时钟，默认为50MHz
3.                 input     rst,     //计数器重置
4.                 input [3:0] kbrow,  //4×4 键盘行信号
5.
6.                 output     beep,       //蜂鸣器
7.                 output [3:0]  kbcol,       //4×4 键盘列信号
```

```
8.              output [7:0]  row,r_col,g_col, //8×8 点阵行信号，红色 r 列信号，绿色 g 列信号
9.              output [7:0]  ds_a,        //数码管段选端
10.             output [7:0]  ds_cat,       //数码管位选端，仅用 3 个位
11.             output [7:0]  led          //快递柜指示 LD
12.            );
13.
14. wire clk_1k,clk_2,clk_4; //分频时钟，下划线后为对应频率
15.
16. wire     cfm;     //操作中心 Operator>>其他，确认信号，键盘输入的数字完成后点击确认，若成功则发出这个信号，使其他器件进行某些操作
17. wire     music;    //操作中心 Operator>>蜂鸣器 BP，音乐播放信号，是持续一个时钟周期的脉冲，接收到该信号后音乐开始播放
18. wire [3:0]  taking;    //操作中心 Operator>>快递柜指示 LD，由操作中心向快递柜标识目前正在取用的快递柜号码
19. wire [3:0]  ds2,ds1,ds0; //操作中心 Opera>>数码管，由操作中心控制数码管的显示值
20. wire [3:0]  snd_sel;   //操作中心 Operator>>蜂鸣器 BP，音乐选择信号，向蜂鸣器发送需要播放的音乐号码
21.
22. wire     kb_en;   //键盘 Keyboard>>操作中心 Operator，使能信号，检测到上升沿时操作中心才会进行操作
23. wire [3:0]  kbrow_p; //消抖后的 4×4 键盘行信号
24. wire [3:0]  num_in;  //键盘 Keyboard>>操作中心 Operator，根据键盘的输入值进行不同的操作
25.
26. wire     closing;  //点阵 Screen>>操作中心 Operator，关门标记，接收到该信号后操作中心指示蜂鸣器播放关门音乐
27. wire     playing;  //点阵 Screen>>其他，正在播放指示，为 1 时表明点阵正在播放开关门动画
28. wire [7:0]  full;    //快递柜指示 LD>>其他，为 1 时指示某个快递柜中存放了物品
29.
30.
31.
32.//分频时钟=======================================
33.   ClockDivide Hz1k(           //1kHz 的分频时钟
34.           .clk_in(clk_in),
35.           .frequency(1_000),
36.           .clk_div(clk_1k),
37.           .rst(rst)
38.           );
39.   ClockDivide Hz2(            //2Hz 的分频时钟
40.           .clk_in(clk_in),
```

```
            .frequency(2),
            .clk_div(clk_2),
            .rst(rst)
            );
    ClockDivide Hz4(              //4Hz 的分频时钟
            .clk_in(clk_in),
            .frequency(4),
            .clk_div(clk_4),
            .rst(rst)
        );
//=============================================


//8×8 点阵=====================
    Screen scr(
          .clk_1k(clk_1k),
          .clk_4(clk_4),
          .row(row),
          .r_col(r_col),
          .g_col(g_col),
          .rst(rst),
          .cfm(cfm),
          .playing(playing),
          .closing(closing)
          );
//============================


//七段数码管 DISP=========
  DISP DS(
      .a(ds_a),
      .cat(ds_cat),
      .clk_1k(clk_1k),
      .rst(rst),
      .ds2(ds2),
      .ds1(ds1),
      .ds0(ds0)
     );
//==================

```

```
85.
86. //4×4 矩阵键盘============
87.  Keyboard kb(
88.       .kbrow_p(kbrow_p),
89.       .kbcol(kbcol),
90.       .clk_1k(clk_1k),
91.       .rst(rst),
92.       .num_in(num_in),
93.       .kb_en(kb_en),
94.       .playing(playing)
95.       );
96. //====================
97.
98.
99.
100.   //操作中心===============
101.    Operator op(
102.        .clk_1k(clk_1k),
103.        .num_in(num_in),
104.        .ds2(ds2),
105.        .ds1(ds1),
106.        .ds0(ds0),
107.        .kb_en(kb_en),
108.        .cfm(cfm),
109.        .taking(taking),
110.        .full(full),
111.        .snd_sel(snd_sel),
112.        .rst(rst),
113.        .music(music),
114.        .closing(closing)
115.        );
116.   //====================
117.
118.
119.
120.   //快递柜指示==============
121.   LD ledld(
122.        .led(led),
123.        .clk_2(clk_2),
124.        .taking(taking),
125.        .playing(playing),
126.        .full(full),
127.        .rst(rst)
128.        );
```

```
129.   //====================
130.
131.
132.
133.   //蜂鸣器================
134.   BP bp(
135.       .clk_in(clk_in),
136.       .clk_1k(clk_1k),
137.       .beep(beep),
138.       .snd_sel(snd_sel),
139.       .music(music),
140.       .rst(rst)
141.       );
142.   //====================
143.
144.
145.
146.   //按键消抖===============
147.    debounce kbdb0(
148.        .clk(clk_in),
149.        .nrst(rst),
150.        .key_in(kbrow[0]),
151.        .key_out(kbrow_p[0])
152.        );
153.    debounce kbdb1(
154.        .clk(clk_in),
155.        .nrst(rst),
156.        .key_in(kbrow[1]),
157.        .key_out(kbrow_p[1])
158.        );
159.    debounce kbdb2(
160.        .clk(clk_in),
161.        .nrst(rst),
162.        .key_in(kbrow[2]),
163.        .key_out(kbrow_p[2])
164.        );
165.    debounce kbdb3(
166.        .clk(clk_in),
167.        .nrst(rst),
168.        .key_in(kbrow[3]),
169.        .key_out(kbrow_p[3])
170.        );
171.  //======================
172.
```

```
173.
174.
175.  endmodule
```

2、时钟分频模块文件

```
1. //时钟分频
2. module ClockDivide(
3.              input      clk_in,     //系统时钟
4.              input      rst,       //计数器重置信号
5.              input [25:0] frequency,  //分频后的时钟频率，例化时直接输入一个数字就可以，为保证 50MHz 能分频成 1Hz，需要 26 位
6.
7.              output reg  clk_div=0  //分频后的时钟
8.             );
9.
10.  parameter SYS_FREQUENCY=26'd50_000_000; //设置系统时钟的频率，默认为 50MHz，该值可在例化时修改
11.
12.  reg [24:0] cnt=0;  //25 位计数器，最小能将 50MHz 分成 1Hz
13.
14.  //计数器控制
15.  always @(posedge clk_in or negedge rst)
16.   begin
17.    if(!rst)begin                        //重置信号到来时直接置零
18.     cnt<=0;
19.    end
20.    else if(cnt == (SYS_FREQUENCY/2/frequency-1)) //该计算公式可以保证分频后时钟频率与 input 的 frequency 值相等
21.       begin
22.        cnt <= 0;
23.        clk_div <= ~clk_div;
24.    end
25.    else                                 //正常计数
26.       cnt <= cnt+1'b1;
27.   end
28.
29.endmodule
```

3、按键消抖模块文件

```
1. //按键消抖
2. module debounce(clk, nrst,key_in,key_out);
3.   input clk, nrst;
4.     input key_in;
5.     output reg key_out;
```

```
6.
7.
8.      parameter TIME_20MS = 1_000_000;
9.
10.     //变量
11.     reg [20:0] cnt=0;
12.     reg key_cnt=0;
13.
14.     // 消抖时间结束后检查按键值
15.     always @(posedge clk or negedge nrst) begin
16.         if(nrst == 0)
17.             key_out <= 0;
18.         //在 20ms 结束，才将 key_in 赋值给 key_out
19.         else if(cnt == TIME_20MS - 1)
20.             key_out <= key_in;
21.     end
22.
23.     //消抖状态时计数，否则为 0
24.     always @(posedge clk or negedge nrst) begin
25.         if(nrst == 0)
26.             cnt <= 0;
27.         else if(key_cnt)
28.             cnt <= cnt + 1'b1;
29.         else
30.             cnt <= 0;
31.     end
32.
33.      always @(posedge clk or negedge nrst) begin
34.             if(nrst == 0)
35.                 key_cnt <= 0;
36.             else if(key_cnt == 0 && key_in != key_out)
37.                 key_cnt <= 1;
38.             else if(cnt == TIME_20MS - 1)
39.                 key_cnt <= 0;
40.      end
41. endmodule
```

4、8×8 点阵模块文件

```
1. //8×8 点阵，行扫描
2. module Screen(
3.          input clk_1k,clk_4,  //时钟
4.          input rst,       //计数器重置
5.          input cfm,      //确认信号，操作中心 Operator>>，接到后播放动
   画
```

```
6.
7.          output reg    playing=0,  //正在播放动画指示，用于锁定键盘等操
    作
8.          output reg    closing=0,  //>>蜂鸣器BP，关门标识
9.          output reg [7:0]  row  =0,  //行信号，低电平有效
10.         output reg [7:0]  r_col  =0,  //红列信号，高电平有效
11.         output reg [7:0]  g_col =0  //绿列信号，高电平有效
12.         );
13.
14.
15.  reg [2:0] cnt_scan= 0;  //动态扫描计数器
16.  reg [4:0] cnt_playing= 0; //图案切换计数器
17.
18.  //图案切换计数
19.  always @(posedge clk_4 or negedge rst or posedge cfm)
20.    begin
21.    if(!rst) begin
22.     cnt_playing<=0;
23.     playing<=0;
24.    end
25.    else if(cfm)begin            //接到cfm信号，计数器由0变为1，后正
    常计数，正在播放指示打开
26.     cnt_playing<=5'b00001;
27.     playing<=1;
28.    end
29.    else if(cnt_playing==5'b11110) begin //计数器到该点后回到0，关闭正
    在播放指示
30.     cnt_playing<=0;
31.     playing<=0;
32.    end
33.    else if(cnt_playing==0)
34.     cnt_playing<=cnt_playing;      //计数器为0时若没有外部触发，则不
    会播放
35.    else begin                  //正常计数
36.     cnt_playing<=cnt_playing+1'b1;
37.     playing<=playing;
38.    end
39.    end
40.
41.  //动态扫描计数器
42.    always @(posedge clk_1k or negedge rst)
43.      begin
44.    if(!rst)begin            //接到rst信号，将计数器置零
45.     cnt_scan<=0;
```

```
46.     end
47.     else if(cnt_playing==0)       //此时播放停止，为安全考虑，将动态扫描
    计数器也停止
48.       cnt_scan<=cnt_scan;
49.       else if(cnt_scan ==3'b111)     //此时 8 行的动态扫描已经完成了 1
    次，将置零再次扫描
50.       cnt_scan <= 3'b000;
51.     else                  //正常计数
52.       cnt_scan <= cnt_scan + 1'b1;
53.     end
54.
55.    //动态扫描控制
56.   always @(posedge clk_1k)begin
57.    case(cnt_scan)
58.    3'b111:row<=8'b0111_1111;
59.    3'b110:row<=8'b1011_1111;
60.    3'b101:row<=8'b1101_1111;
61.    3'b100:row<=8'b1110_1111;
62.    3'b011:row<=8'b1111_0111;
63.    3'b010:row<=8'b1111_1011;
64.    3'b001:row<=8'b1111_1101;
65.    3'b000:row<=8'b1111_1110;
66.    default:row<=0;
67.    endcase
68.   end
69.
70.
71. //图案切换控制=========================================================
72.    always @(negedge rst or posedge clk_1k)
73.    begin
74.      if(!rst) begin
75.     r_col<=0;
76.     g_col<=0;
77.     end
78.     else begin
79.      case(cnt_playing)
80.
81.      //开门，一拍一个图案，与关门时区别，中心会有一个黄色的方块表示包裹在
    里面=====
82.       5'b00000:begin
83.         case(cnt_scan)
84.         3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
```

```
85.         3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
86.         3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
87.         3'b100: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
88.         3'b011: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
89.         3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
90.         3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
91.         3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; en
    d
92.         default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000; e
    nd
93.         endcase
94.        end
95.       5'b00001:begin
96.         case(cnt_scan)
97.         3'b111: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000; en
    d
98.         3'b110: begin  r_col<=8'b1110_0111; g_col<= 8'b0001_1000; en
    d
99.         3'b101: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100; en
    d
100.           3'b100: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
     end
101.           3'b011: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
     end
102.           3'b010: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
     end
103.           3'b001: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
     end
104.           3'b000: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
     end
105.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
    ; end
106.           endcase
107.          end
108.        5'b00010:begin
109.           case(cnt_scan)
110.           3'b111: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
     end
```

```
111.           3'b110: begin  r_col<=8'b1110_0111; g_col<= 8'b0001_1000;
    end
112.           3'b101: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
113.           3'b100: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
    end
114.           3'b011: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
115.           3'b010: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
116.           3'b001: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
    end
117.           3'b000: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
118.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
119.           endcase
120.         end
121.        5'b00011:begin
122.           case(cnt_scan)
123.           3'b111: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
124.           3'b110: begin  r_col<=8'b1111_1011; g_col<= 8'b0000_0100;
    end
125.           3'b101: begin  r_col<=8'b1111_1011; g_col<= 8'b0000_0100;
    end
126.           3'b100: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
    end
127.           3'b011: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
128.           3'b010: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
129.           3'b001: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
    end
130.           3'b000: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
131.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
132.           endcase
133.         end
134.       5'b00100:begin
135.           case(cnt_scan)
136.           3'b111: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
```

```
137.           3'b110: begin  r_col<=8'b1111_1001; g_col<= 8'b0000_0110;
    end
138.           3'b101: begin  r_col<=8'b1111_1010; g_col<= 8'b0000_0101;
    end
139.           3'b100: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
    end
140.           3'b011: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
141.           3'b010: begin  r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
    end
142.           3'b001: begin  r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
    end
143.           3'b000: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
144.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
145.           endcase
146.         end
147.        5'b00101:begin
148.           case(cnt_scan)
149.           3'b111: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
150.           3'b110: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
151.           3'b101: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
152.           3'b100: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
153.           3'b011: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
154.           3'b010: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
155.           3'b001: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
156.           3'b000: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
157.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
158.           endcase
159.         end
160.       5'b00110:begin
161.           case(cnt_scan)
162.           3'b111: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
```

```
163.           3'b110: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
164.           3'b101: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
165.           3'b100: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
166.           3'b011: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
167.           3'b010: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
168.           3'b001: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
169.           3'b000: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
170.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
171.           endcase
172.         end
173.        5'b00111:begin
174.           case(cnt_scan)
175.           3'b111: begin  r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
    end
176.           3'b110: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
177.           3'b101: begin  r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
    end
178.           3'b100: begin  r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
    end
179.           3'b011: begin  r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
    end
180.           3'b010: begin  r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
    end
181.           3'b001: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
182.           3'b000: begin  r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
    end
183.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
184.           endcase
185.         end
186.        5'b01000:begin
187.           case(cnt_scan)
188.           3'b111: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
```

```
189.           3'b110: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
190.           3'b101: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
191.           3'b100: begin  r_col<=8'b0001_1111; g_col<= 8'b0001_0000;
    end
192.           3'b011: begin  r_col<=8'b0001_1111; g_col<= 8'b0001_0000;
    end
193.           3'b010: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
194.           3'b001: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
195.           3'b000: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
196.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
197.           endcase
198.         end
199.       5'b01001:begin
200.           case(cnt_scan)
201.           3'b111: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
202.           3'b110: begin  r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
    end
203.           3'b101: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
204.           3'b100: begin  r_col<=8'b0001_1111; g_col<= 8'b0001_1000;
    end
205.           3'b011: begin  r_col<=8'b0001_1111; g_col<= 8'b0001_1000;
    end
206.           3'b010: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
207.           3'b001: begin  r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
    end
208.           3'b000: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
209.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
210.           endcase
211.         end
212.
213.
214.       //柜门完全打开，以下空几拍保证3秒后柜门才关闭，中心黄色包裹图案显
    示2秒=====
```

```
215.         5'b01010:begin
216.           case(cnt_scan)
217.           3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
218.           3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
219.           3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
220.           3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
221.           3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
222.           3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
223.           3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
224.           3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
225.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
226.           endcase
227.         end
228.       5'b01011:begin
229.           case(cnt_scan)
230.           3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
231.           3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
232.           3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
233.           3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
234.           3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
235.           3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
236.           3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
237.           3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
238.           default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
239.           endcase
240.         end
```

```
241.       5'b01100:begin
242.          case(cnt_scan)
243.          3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
244.          3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
245.          3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
246.          3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
247.          3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
248.          3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
249.          3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
250.          3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
251.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
252.          endcase
253.         end
254.       5'b01101:begin
255.          case(cnt_scan)
256.          3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
257.          3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
258.          3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
259.          3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
260.          3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
261.          3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
262.          3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
263.          3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
264.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
265.          endcase
266.         end
```

```
267.      5'b01110:begin
268.          case(cnt_scan)
269.          3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
270.          3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
271.          3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
272.          3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
273.          3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
274.          3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
275.          3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
276.          3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
277.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
278.          endcase
279.         end
280.      5'b01111:begin
281.          case(cnt_scan)
282.          3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
283.          3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
284.          3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
285.          3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
286.          3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
287.          3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
288.          3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
289.          3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
290.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
291.          endcase
292.         end
```

```
293.      5'b10000:begin
294.          case(cnt_scan)
295.          3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
296.          3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
297.          3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
298.          3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
299.          3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
300.          3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
301.          3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
302.          3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
303.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
304.          endcase
305.         end
306.      5'b10001:begin
307.          case(cnt_scan)
308.          3'b111: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
309.          3'b110: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
310.          3'b101: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
311.          3'b100: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
312.          3'b011: begin  r_col<=8'b0001_1000; g_col<= 8'b0001_1000;
    end
313.          3'b010: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
314.          3'b001: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
315.          3'b000: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
316.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
317.          endcase
318.         end
```

```
319.
320.       //包裹消失，再过一秒关柜门，此时屏幕不亮，default 已包含该情况
   =========
321.
322.
323.       //关门，一拍一个图案==================================
324.       5'b10110:begin
325.          closing<=1;
326.          case(cnt_scan)
327.          3'b111: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
328.          3'b110: begin  r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
    end
329.          3'b101: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
330.          3'b100: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
331.          3'b011: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
332.          3'b010: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
333.          3'b001: begin  r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
    end
334.          3'b000: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
335.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
336.          endcase
337.         end
338.       5'b10111:begin
339.         closing<=0;
340.          case(cnt_scan)
341.          3'b111: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
342.          3'b110: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
343.          3'b101: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
344.          3'b100: begin  r_col<=8'b0001_1111; g_col<= 8'b0001_0000;
    end
345.          3'b011: begin  r_col<=8'b0001_1111; g_col<= 8'b0001_0000;
    end
346.          3'b010: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
```

```
347.          3'b001: begin  r_col<=8'b0000_0111; g_col<= 8'b0000_0000;
    end
348.          3'b000: begin  r_col<=8'b0000_0001; g_col<= 8'b0000_0000;
    end
349.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
350.          endcase
351.         end
352.       5'b11000:begin
353.          case(cnt_scan)
354.          3'b111: begin r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
   end
355.          3'b110: begin r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
   end
356.          3'b101: begin r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
   end
357.          3'b100: begin r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
   end
358.          3'b011: begin r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
   end
359.          3'b010: begin r_col<=8'b0011_1111; g_col<= 8'b0000_0000;
   end
360.          3'b001: begin r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
   end
361.          3'b000: begin r_col<=8'b0000_0011; g_col<= 8'b0000_0000;
   end
362.          default: begin r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
363.          endcase
364.         end
365.       5'b11001:begin
366.          case(cnt_scan)
367.          3'b111: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
368.          3'b110: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
369.          3'b101: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
370.          3'b100: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
371.          3'b011: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
372.          3'b010: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
```

```
373.          3'b001: begin  r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
    end
374.          3'b000: begin  r_col<=8'b0000_1111; g_col<= 8'b0000_0000;
    end
375.          default: begin  r_col<=8'b0000_0000; g_col<= 8'b0000_0000
   ; end
376.          endcase
377.         end
378.      5'b11010:begin
379.          case(cnt_scan)
380.          3'b111: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
381.          3'b110: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
382.          3'b101: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
383.          3'b100: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
384.          3'b011: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
385.          3'b010: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
386.          3'b001: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
387.          3'b000: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
388.          default: begin r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
389.          endcase
390.         end
391.      5'b11011:begin
392.          case(cnt_scan)
393.          3'b111: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
394.          3'b110: begin r_col<=8'b1111_1001; g_col<= 8'b0000_0110;
   end
395.          3'b101: begin r_col<=8'b1111_1010; g_col<= 8'b0000_0101;
   end
396.          3'b100: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
397.          3'b011: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
398.          3'b010: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
```

```
399.           3'b001: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
400.           3'b000: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
401.           default: begin r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
402.           endcase
403.         end
404.       5'b11100:begin
405.           case(cnt_scan)
406.           3'b111: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
407.           3'b110: begin r_col<=8'b1111_1011; g_col<= 8'b0000_0100;
   end
408.           3'b101: begin r_col<=8'b1111_1011; g_col<= 8'b0000_0100;
   end
409.           3'b100: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
410.           3'b011: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
411.           3'b010: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
412.           3'b001: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
413.           3'b000: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
414.           default: begin r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
415.           endcase
416.         end
417.       5'b11101:begin
418.           case(cnt_scan)
419.           3'b111: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
420.           3'b110: begin r_col<=8'b1110_0111; g_col<= 8'b0001_1000;
   end
421.           3'b101: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
422.           3'b100: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
423.           3'b011: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
424.           3'b010: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
```

```
425.          3'b001: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
426.          3'b000: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
427.          default: begin r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
428.          endcase
429.         end
430.      5'b11110:begin
431.          case(cnt_scan)
432.          3'b111: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
433.          3'b110: begin r_col<=8'b1110_0111; g_col<= 8'b0001_1000;
   end
434.          3'b101: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
435.          3'b100: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
436.          3'b011: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
437.          3'b010: begin r_col<=8'b1101_1011; g_col<= 8'b0010_0100;
   end
438.          3'b001: begin r_col<=8'b1100_0011; g_col<= 8'b0011_1100;
   end
439.          3'b000: begin r_col<=8'b1111_1111; g_col<= 8'b0000_0000;
   end
440.          default: begin r_col<=8'b0000_0000; g_col<= 8'b0000_0000;
    end
441.          endcase
442.         end
443.
444.      //关门完成，计数器回到零停止，点阵全部熄灭
   ========================
445.
446.
447.      //出现任何意外状况都保证点阵全部熄灭============================
448.      default:begin r_col<=0; g_col<= 0; end
449.
450.      endcase
451.      end
452.     end
453.    //图案切换控制结束
   ================================================
454.
```

```
455.
456.
457.   endmodule
```

5、数码管模块文件

```
1. //七段数码管
2. module DISP(
3.         input    clk_1k,     //时钟
4.         input    rst,        //计数器重置
5.         input [3:0] ds2,ds1,ds0,  //分别选择数码管2、1、0 位需要显示的数字
6.
7.         output reg [7:0] a  =0,  //数码管段码A~G，最后一位小数点
8.         output reg [7:0] cat =0  //数码管位选，其实只用3 位，但只写三位别的会亮
9.         );
10.
11. reg [7:0] seg [11:0];  //设置数组，存储不同数字需要打开的数码管段
12.
13. reg    seg_write =1;  //初始化 seg 数组的开关，seg 数组只写一次，写完就置零
14. reg [1:0] cnt_cat    =0;  //位选端的动态扫描计数器
15.
16. localparam   NUM0 =4'h0,
17.        NUM1 =4'h1,
18.        NUM2 =4'h2,
19.        NUM3 =4'h3,
20.        NUM4 =4'h4,
21.        NUM5 =4'h5,
22.        NUM6 =4'h6,
23.        NUM7 =4'h7,
24.        NUM8 =4'h8,
25.        NUM9 =4'h9,
26.        NONE =4'hA, //空屏
27.        ERROR =4'hB; //字符E
28.
29. //初始化 seg 数组
30. always@(posedge clk_1k) begin
31.  if(seg_write) begin
32.   seg[NUM0] <=8'b00111111; //显示0
33.   seg[NUM1] <=8'b00000110; //显示1
34.   seg[NUM2] <=8'b01011011; //显示2
35.   seg[NUM3] <=8'b01001111; //显示3
36.   seg[NUM4] <=8'b01100110; //显示4
```

```
37.    seg[NUM5] <=8'b01101101; //显示5
38.    seg[NUM6] <=8'b01111101; //显示6
39.    seg[NUM7] <=8'b00000111; //显示7
40.    seg[NUM8] <=8'b01111111; //显示8
41.    seg[NUM9] <=8'b01101111; //显示9
42.    seg[NONE] <=8'b00000000; //显示空
43.    seg[ERROR] <=8'b01111001; //显示E
44.    seg_write<=0;    //seg 初始化完毕，该信号置零
45.   end
46. end
47.
48. //数码管扫描计数器控制
49.    always @(posedge clk_1k or negedge rst)
50.    begin
51.    if(!rst)            //重置信号到来直接置到一个不会亮屏的值
52.      cnt_cat <= 2'b11;
53.    else if(cnt_cat ==2'b10) //只用三段数码管，到3 就变回0
54.      cnt_cat <= 2'b00;
55.    else              //正常计数
56.      cnt_cat <= cnt_cat + 1'b1;
57.    end
58.
59.   // 数码管位选动态扫描
60. always @(posedge clk_1k)
61. begin
62.   case(cnt_cat)
63.     2'b00:  begin cat<=8'b11111110;  a<=seg[ds0];  end //0 位
64.     2'b01:  begin cat<=8'b11111101;  a<=seg[ds1]; end //1 位
65.     2'b10:  begin cat<=8'b11111011;  a<=seg[ds2];  end //2 位
66.     default: begin cat<=0;         a<=0;       end
67.   endcase
68. end
69.
70.endmodule
```

6、4×4 键盘模块文件

```
1. //4×4 键盘，列扫描
2. module Keyboard(
3.             input    clk_1k,    //时钟
4.             input    rst,      //计数器重置
5.             input    playing,  //8×8 点阵 Screen>>，用于锁定键盘
6.             input [3:0] kbrow_p,  //消抖后的键盘行信号
7.
```

```
8.            output reg    kb_en =0,      //键盘输入有效信号，检测到上升沿后才能让值进入操作中心，使用原因是键盘是动态扫描的，直接输出不稳定，使用使能信号保证稳定输出
9.            output reg [3:0] kbcol  =4'b0000,  //列信号
10.           output reg [3:0] num_in        //>>操作中心Operator，向操作中心输入一个值
11.         );
12.
13. reg    col_ch   =0;    //已标识到列信号，用该值控制键盘的输入是否有效
14. reg [1:0] col_index =0;    //选中列标识，当检测到有效输入后对列进行标识，最后计算输出值时使用
15. reg [1:0] row_index =0;    //选中行标识，当检测到有效输入后对行进行标识，最后计算输出值时使用
16. reg [2:0] cnt     =0;    //动态列扫描计数器
17.
18. localparam   NUM0   =4'h0,   //数字0
19.        NUM1   =4'h1,   //数字1
20.        NUM2   =4'h2,   //数字2
21.        NUM3   =4'h3,   //数字3
22.        NUM4   =4'h4,   //数字4
23.        NUM5   =4'h5,   //数字5
24.        NUM6   =4'h6,   //数字6
25.        NUM7   =4'h7,   //数字7
26.        NUM8   =4'h8,   //数字8
27.        NUM9   =4'h9,   //数字9
28.        DELETE  =4'hD,   //删除键
29.        EMPTY   =4'hE,   //清空键
30.        CONFIRM =4'hC, //确认键
31.        NONE   =4'hA;   //无效键，按下无操作
32.
33. //状态机
34. always@(posedge clk_1k or negedge rst or posedge playing) begin
35.  if(!rst)begin
36.   cnt<=0;
37.   col_ch<=0;
38.   col_index<=0;
39.  end
40.  else if(playing) begin
41.   col_ch<=0;
42.   cnt<=0;
43.  end
44.  else begin
45.   case(cnt)
46.    3'd0:begin  //停止扫描状态
```

```
47.     col_ch<=0;     //标识到列信号置0，由于采用时钟上升沿扫描，该脉冲会持续一个时钟周期
48.         if(kbrow_p==4'b1111 && !playing)begin //确认当前无按键按下，且8×8 点阵 Screen 没有在播放动画时才进行扫描
49.         cnt<=cnt+1'b1;  //从第一列开始扫描
50.         kbcol<=4'b1110; //提前准备好列信号
51.         end
52.        end
53.     3'd1:begin  //扫描第一列
54.         if(kbrow_p!=4'b1111)begin //该列检测到按键按下
55.          col_index<=2'd0; //标识该列，当前是 0 列
56.          col_ch<=1;     //列信号已标识完成
57.          cnt<=0;       //回到停止扫描状态
58.         end
59.         else begin       //该列没有检测到按键按下
60.          cnt<=cnt+1'b1;  //扫描下一列
61.          kbcol<=4'b1101; //提前准备好列信号
62.         end
63.        end
64.     3'd2:begin  //扫描第二列
65.         if(kbrow_p!=4'b1111)begin
66.          col_index<=2'd1;
67.          col_ch<=1;
68.          cnt<=0;
69.         end
70.         else begin
71.          cnt<=cnt+1'b1;
72.          kbcol<=4'b1011;
73.         end
74.        end
75.     3'd3:begin  //扫描第三列
76.        if(kbrow_p!=4'b1111)begin
77.          col_index<=2'd2;
78.          col_ch<=1;
79.          cnt<=0;
80.         end
81.         else begin
82.          cnt<=cnt+1'b1;
83.          kbcol<=4'b0111;
84.         end
85.        end
86.     3'd4:begin  //扫描第四列
87.        if(kbrow_p!=4'b1111)begin
88.          col_index<=2'd3;
```

```
          col_ch<=1;
          cnt<=0;
        end
        else begin
          cnt<=3'd1;
          kbcol<=4'b1110;
        end
      end
    endcase
  end
end

  //行信号标识
  always@(posedge clk_1k or negedge rst)begin
   if(!rst) begin
    row_index<=0;
   end
   else begin
    case(kbrow_p)
     4'b1110:row_index<=2'd3;
     4'b1101:row_index<=2'd2;
     4'b1011:row_index<=2'd1;
     4'b0111:row_index<=2'd0;
     default:row_index<=row_index;
    endcase
   end
  end

  //通过行列标识，指示键盘输出一个值，让操作中心判断进行什么操作
  always@(negedge clk_1k)begin
  //由于行列标识是多位，且在时钟上升沿改变的，若此处使用时钟上升沿检测，发现输出的信号为上一状态，所以使用时钟下降沿检测，当行列信号在上升沿改变完成并稳定后，延迟半个时钟周期判断输出值
   case({row_index,col_index})
     4'b1101: num_in<=NUM0;   //输入数字0
     4'b0000: num_in<=NUM1;   //输入数字1
     4'b0001: num_in<=NUM2;   //输入数字2
     4'b0010: num_in<=NUM3;   //输入数字3
     4'b0100: num_in<=NUM4;   //输入数字4
     4'b0101: num_in<=NUM5;   //输入数字5
     4'b0110: num_in<=NUM6;   //输入数字6
     4'b1000: num_in<=NUM7;   //输入数字7
     4'b1001: num_in<=NUM8;   //输入数字8
     4'b1010: num_in<=NUM9;   //输入数字9
```

```
      4'b0011: num_in<=DELETE;  //删除键 Delete
      4'b0111: num_in<=EMPTY;   //清空键 Empty
      4'b1011: num_in<=CONFIRM; //确认键 Confirm
      default: num_in<=NONE;    //空键
     endcase
    if(col_ch)begin //检测到列信号被标识后发出键盘有效信号
     kb_en<=1; //由于采用时间下降沿检测，该脉冲会持续一个时钟周期
    end
    else begin
     kb_en<=0;
    end
   end

endmodule
```

7、操作中心模块文件

```
//操作中心
module Operator(
          input clk_1k,      //1kHz 时钟
          input rst,        //重置信号
          input kb_en,      //键盘 Keyboard>>，键盘使能，为了保证键盘扫描的稳定设置的，检测到上升沿后键盘的值成功输入到操作中心
          input closing,     //8×8 点阵 Screen>>，关门信号
          input [3:0] num_in, //键盘 Keyboard>>，输入一个数，让操作中心判断需要进行什么操作
          input [7:0] full,    //快递柜指示 LD>>，指示快递柜目前是否被占用

          output reg    cfm   =0,     //>>其他，表明键盘输入的三个数是有效的，指示其他器件进行打开柜门的操作
          output reg    music  =0,     //>>蜂鸣器 BP，操作蜂鸣器开始播放音乐
          output reg [3:0]  taking  =4'hA,  //>>快递柜指示 LD，指示目前某个快递柜正在被取用，让 LED 闪烁
          output reg [3:0]  snd_sel =4'hF,  //>>蜂鸣器 BP，指示蜂鸣器目前应该播放的音乐

          output reg [3:0] ds2=4'hA, //>>七段数码管 DISP，七段数码管 2 位显示的数字选择
          output reg [3:0] ds1=4'hA, //>>七段数码管 DISP，七段数码管 1 位显示的数字选择
          output reg [3:0] ds0=4'hA //>>七段数码管 DISP，七段数码管 0 位显示的数字选择
          );
```

```
19.
20. localparam  DS_NONE =4'hA,     //数码管显示空
21.         DS_ERROR =4'hB,     //数码管显示 E
22.
23.         BP_NONE    =4'hF,  //蜂鸣器不选择音乐
24.         BP_ALLCLEAR =4'hA,  //蜂鸣器选择清空音乐
25.         BP_COLSING =4'hB,  //蜂鸣器选择关门音乐
26.         BP_OPENING =4'hC,  //蜂鸣器选择开门音乐
27.         BP_DELETE  =4'hD,  //蜂鸣器选择删除音乐
28.         BP_ERROR   =4'hE,  //蜂鸣器选择错误音乐
29.
30.         KB_NONE    =4'hA,  //键盘按下无效键
31.         KB_CONFIRM =4'hC,  //键盘按下确认键
32.         KB_DELETE  =4'hD,  //键盘按下删除键
33.         KB_EMPTY   =4'hE,  //键盘按下清空键
34.
35.         LD_NONE    =4'hA;  //LED 未选中任何快递柜
36.
37. //具体操作，检测到键盘使能上升沿后启动
38.  always@(posedge clk_1k or negedge rst or posedge closing)begin
39.   if(!rst) begin
40.    ds2  <=DS_NONE;
41.    ds1  <=DS_NONE;
42.    ds0  <=DS_NONE;
43.    snd_sel<=BP_NONE;
44.    taking <=LD_NONE;
45.    cfm  <=0;
46.    music <=0;
47.   end
48.   else if(closing)begin
49.    snd_sel<=BP_COLSING;
50.    music <=1;
51.   end
52.   else begin
53.    if(kb_en)begin
54.     case(num_in)
55.      KB_NONE:    begin  //键盘传入这个值时，操作中心判断为按下了一个空
   键
56.                ds2<=ds2;
57.                ds1<=ds1;
58.                ds0<=ds0;
59.               end
60.      KB_CONFIRM:  begin  //键盘传入这个值时，操作中心判断为按下了确认
   键
```

```
61.                 if({ds2,ds1,ds0}<8) begin //仅有 8 个快递柜，且号码为
    000 到 007，所以只有这几个数据有效
62.                  if(full[ds0]) begin //对应的箱子里有快递，则判断传入数
    据成功
63.                   ds2  <=DS_NONE;    //数码管清空
64.                   ds1  <=DS_NONE;
65.                   ds0  <=DS_NONE;
66.                   snd_sel<=BP_OPENING; //向蜂鸣器 BP 模块指示播放开门音
    乐
67.                   taking <=ds0;       //向快递柜指示 LD 模块指示目前取用
    的快递箱号
68.                   cfm  <=1;          //发出确认信号，在代码末尾处 else 后
    会再次置零，逻辑上表明 cfm 是仅持续一个时钟周期的脉冲
69.                   music <=1;
70.                  end
71.                  else begin      //对应的箱子空，传入数据失败，显示 E
72.                   ds2  <=DS_NONE;
73.                   ds1  <=DS_NONE;
74.                   ds0  <=DS_ERROR;  //此号码对应显示 E 字符，详见 DISP
    模块
75.                   snd_sel<=BP_ERROR;  //向蜂鸣器 BP 模块指示播放错误音
    乐
76.                   music <=1;
77.                  end
78.                end
79.                else begin       //输入的是一个无效号码，传入数据失败
80.                 ds2  <=DS_NONE;
81.                 ds1  <=DS_NONE;
82.                 ds0  <=DS_ERROR;  //此号码对应显示 E 字符，详见 DISP 模
    块
83.                 snd_sel<=BP_ERROR;  //向蜂鸣器 BP 模块指示播放错误音乐
84.                 music <=1;
85.                end
86.              end
87.      KB_DELETE: begin //键盘传入这个值时，操作中心判断进行删除操作，数
    据退一格
88.              ds2  <=DS_NONE;
89.              ds1  <=ds2;
90.              ds0  <=ds1;
91.              snd_sel<=BP_DELETE; //向蜂鸣器 BP 模块指示播放删除音乐
92.              music <=1;
93.             end
94.      KB_EMPTY: begin  //键盘传入这个值时，操作中心判断进行清空操作，数
    码管清空
```

```
95.                ds2  <=DS_NONE;
96.                ds1  <=DS_NONE;
97.                ds0  <=DS_NONE;
98.                snd_sel<=BP_ALLCLEAR; //向蜂鸣器 BP 模块指示播放清空音乐
99.                music <=1;
100.                end
101.         default:  begin //键盘传入的其他值都判断为输入了一个数字，且输入三位后就停止输入
102.                if(ds0==DS_ERROR)begin     //显示 E 字符后输入，错误字符不占位，第 0 位数码管直接由 E 变为输入的数
103.                 ds2  <=DS_NONE;
104.                 ds1  <=DS_NONE;
105.                 ds0  <=num_in;
106.                 snd_sel<=num_in; //向蜂鸣器 BP 指示播放输入对应数字的音乐
107.                 music <=1;
108.                end
109.                else if(ds2==DS_NONE)begin  //最高位空时才可以输入，保证输入的数字只有三位
110.                 ds2  <=ds1;
111.                 ds1  <=ds0;
112.                 ds0  <=num_in;
113.                 snd_sel<=num_in; //向蜂鸣器 BP 指示播放输入对应数字的音乐
114.                 music <=1;
115.                end
116.               end
117.       endcase
118.      end
119.      else begin
120.       cfm  <=0;  //确认信号在一个时钟周期后置零
121.       music <=0;
122.      end
123.     end
124.    end
125.
126.  endmodule
```

8、蜂鸣器模块文件

```
1. //蜂鸣器
2. module BP(
3.         input    clk_in,clk_1k, //时钟信号
4.         input    rst,       //重置信号
5.         input    music,     //操作中心 Operator>>，音乐播放信号
```

```
6.          input [3:0] snd_sel,    //操作中心 Operator>>，音乐选择信号，播放某一段音乐
7.
8.          output reg beep=1 //蜂鸣器电平，按一定频率高低电平切换时会发出响声
9.        );
10.
11. parameter DIVIDE=2;  //用音高计数器，即周期除这个数，再响应调节占空比计数器，可以调节占空比，为 2 的时候占空比 50%
12.
13. reg [8:0]  note_duration =0;  //当前音符的时值，灵敏度 1ms
14. reg [17:0] note_pitch;      //当前音符的音高
15.
16. reg [3:0]  cnt_note     =0;  //音符切换计数器，+1 后播放下一个音符
17. reg [8:0]  cnt_duration  =0;  //时值计数器，控制某个音响多久
18. reg [17:0]  cnt_pitch    =0;  //音高计数器，计数器置零时该周期结束，beep 由低电平变为高电平
19. reg [17:0]  cnt_tone     =0;  //占空比调节计数器，计数器置零时 beep 由高电平变为低电平
20.
21. localparam   //音高标识
22.         c1_PITCH =18'd190839,   //不同音符对音高计数器的置零时的取值要求，控制着该音高的周期长度
23.         d1_PITCH =18'd170068,
24.         e1_PITCH =18'd151515,   //音符表示方法示例：c1 为小字一组的 do，即中央 C
25.         f1_PITCH =18'd143266,
26.         g1_PITCH =18'd127511,
27.         a1_PITCH =18'd113636,
28.         b1_PITCH =18'd101215,
29.         c2_PITCH =18'd95602,
30.         d2_PITCH =18'd85179,
31.         e2_PITCH =18'd75873,
32.         f2_PITCH =18'd71633,
33.         g2_PITCH =18'd63776,
34.         a2_PITCH =18'd56818,
35.         b2_PITCH =18'd50607,
36.
37.         //音乐选择
38.         NUM0    =4'h0,  //数字 0
39.         NUM1    =4'h1,  //数字 1
40.         NUM2    =4'h2,  //数字 2
41.         NUM3    =4'h3,  //数字 3
42.         NUM4    =4'h4,  //数字 4
```

```
        NUM5    =4'h5,  //数字5
        NUM6    =4'h6,  //数字6
        NUM7    =4'h7,  //数字7
        NUM8    =4'h8,  //数字8
        NUM9    =4'h9,  //数字9
        ALLCLEAR =4'hA,  //清空键
        CLOSING =4'hB,  //关门
        CONFIRM =4'hC,  //开门
        DELETE  =4'hD,  //删除
        ERROR   =4'hE,  //错误

        //停止标识，对时值进行标识
        STOP_NOTE=9'b1_1111_1111; //停止

//占空比计数器的计算
always@(posedge clk_in or negedge rst) begin
 if(!rst)
  cnt_tone<=0;
 else if(cnt_tone==note_pitch/DIVIDE-1 || cnt_pitch==note_pitch-
1) //置零的两种条件：1、高电平结束；2、周期结束
  cnt_tone<=0;
 else cnt_tone<=cnt_tone+1'b1;                    //正常计数
end

//音高计数器的计算
always@(posedge clk_in or negedge rst) begin
 if(!rst)
  cnt_pitch<=0;
 else if(cnt_pitch==note_pitch-1)  //置零条件：周期结束
  cnt_pitch<=0;
 else                //正常计数
  cnt_pitch<=cnt_pitch+1'b1;
end

//蜂鸣器电平的切换
always@(posedge clk_in) begin
 if(note_pitch==0)
  beep<=1;
 else begin
  if(cnt_tone==0 && beep==1)     //占空比计数器结束，beep 为1>>0
   beep<=0;
  else if(cnt_pitch==0 && beep ==0)  //音高计数器结束，beep 为0>>1
   beep<=1;
   else beep<=beep;          //其余情况不变
```

```
86.   end
87.  end
88.
89.  //音符计数器的计算
90.  always@(posedge clk_1k or posedge music or negedge rst)begin
91.   if(!rst)
92.    cnt_note<=0;
93.   else if(music)
94.    cnt_note<=4'b0001;
95.   else if(note_duration==STOP_NOTE)
96.    cnt_note<=0;
97.   else if(cnt_duration==note_duration)
98.    cnt_note<=cnt_note+1'b1;
99.  end
100.
101.   always@(posedge clk_1k or negedge rst or posedge music)begin
102.    if(!rst)
103.     cnt_duration<=0;
104.    else if(music)
105.     cnt_duration<=0;
106.    else if(cnt_duration==note_duration)
107.     cnt_duration<=0;
108.    else if(note_duration==STOP_NOTE)
109.     cnt_duration<=cnt_duration;
110.    else cnt_duration<=cnt_duration+1'b1;
111.   end
112.
113.   //判断音乐选择和音高选择，播放音乐
114.   always@(posedge clk_in)begin
115.    case(snd_sel)
116.     NUM0:   case(cnt_note)              //0 到9 为按下相应数字按键的
    音乐
117.             4'b0001: begin note_pitch<=e2_PITCH; note_duration<=9'
    d125;    end
118.             default: begin note_pitch<=0;     note_duration<=STOP_
    NOTE; end
119.           endcase
120.     NUM1:   case(cnt_note)
121.             4'b0001: begin note_pitch<=c1_PITCH; note_duration<=9'
    d125;    end
122.             default: begin note_pitch<=0;     note_duration<=STOP_
    NOTE; end
123.           endcase
124.     NUM2:   case(cnt_note)
```

```
125.             4'b0001: begin note_pitch<=d1_PITCH; note_duration<=9'
   d125;    end
126.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
127.           endcase
128.     NUM3:   case(cnt_note)
129.             4'b0001: begin note_pitch<=e1_PITCH; note_duration<=9'
   d125;    end
130.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
131.           endcase
132.     NUM4:   case(cnt_note)
133.             4'b0001: begin note_pitch<=f1_PITCH; note_duration<=9'
   d125;    end
134.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
135.           endcase
136.     NUM5:   case(cnt_note)
137.             4'b0001: begin note_pitch<=g1_PITCH; note_duration<=9'
   d125;    end
138.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
139.           endcase
140.     NUM6:   case(cnt_note)
141.             4'b0001: begin note_pitch<=a1_PITCH; note_duration<=9'
   d125;    end
142.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
143.           endcase
144.     NUM7:   case(cnt_note)
145.             4'b0001: begin note_pitch<=b1_PITCH; note_duration<=9'
   d125;    end
146.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
147.           endcase
148.     NUM8:   case(cnt_note)
149.             4'b0001: begin note_pitch<=c2_PITCH; note_duration<=9'
   d125;    end
150.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
151.           endcase
152.     NUM9:   case(cnt_note)
153.             4'b0001: begin note_pitch<=d2_PITCH; note_duration<=9'
   d125;    end
```

```
154.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
155.           endcase
156.     ALLCLEAR:  case(cnt_note)                //清空（All）音乐
157.             4'b0001: begin note_pitch<=g1_PITCH; note_duration<=9'
   d125;    end
158.             4'b0010: begin note_pitch<=a1_PITCH; note_duration<=9'
   d125;    end
159.             4'b0011: begin note_pitch<=f1_PITCH; note_duration<=9'
   d125;    end
160.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
161.           endcase
162.     CLOSING:  case(cnt_note)//关门
163.             4'b0001: begin note_pitch<=c2_PITCH; note_duration<=9'
   d125;    end
164.             4'b0010: begin note_pitch<=b1_PITCH; note_duration<=9'
   d125;    end
165.             4'b0011: begin note_pitch<=a1_PITCH; note_duration<=9'
   d125;    end
166.             4'b0100: begin note_pitch<=g1_PITCH; note_duration<=9'
   d125;    end
167.             4'b0101: begin note_pitch<=f1_PITCH; note_duration<=9'
   d125;    end
168.             4'b0110: begin note_pitch<=e1_PITCH; note_duration<=9'
   d125;    end
169.             4'b0111: begin note_pitch<=d1_PITCH; note_duration<=9'
   d125;    end
170.             4'b1000: begin note_pitch<=c1_PITCH; note_duration<=9'
   d250;    end
171.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
172.           endcase
173.     CONFIRM:  case(cnt_note)                 //确认（Comfirm，开门）音
   乐
174.             4'b0001: begin note_pitch<=c1_PITCH; note_duration<=9'
   d375;    end
175.             4'b0010: begin note_pitch<=e1_PITCH; note_duration<=9'
   d125;    end
176.             4'b0011: begin note_pitch<=g1_PITCH; note_duration<=9'
   d125;    end
177.             4'b0100: begin note_pitch<=0;     note_duration<=9'd12
   5;    end
```

```
178.             4'b0101: begin note_pitch<=c2_PITCH; note_duration<=9'
   d125;    end
179.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
180.           endcase
181.     DELETE:   case(cnt_note)              //删除（Delete）音乐
182.             4'b0001: begin note_pitch<=a2_PITCH; note_duration<=9'
   d125;    end
183.             4'b0010: begin note_pitch<=f2_PITCH; note_duration<=9'
   d125;    end
184.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
185.           endcase
186.     ERROR:   case(cnt_note)              //错误（Error，显示 E 字符）
   音乐
187.             4'b0001: begin note_pitch<=g1_PITCH; note_duration<=9'
   d125;    end
188.             4'b0010: begin note_pitch<=g1_PITCH; note_duration<=9'
   d125;    end
189.             default: begin note_pitch<=0;     note_duration<=STOP_
   NOTE; end
190.           endcase
191.     default:         begin note_pitch<=0;      note_duration<=STOP
   _NOTE;  end
192.    endcase
193.   end
194.
195.  endmodule
```

9、LED 模块文件

```
1. //快递柜指示
2. module LD(
3.         input    clk_2,  //时钟
4.         input    rst,    //重置信号，让快递柜回到全部被占用的状态
5.         input    playing, //8×8 点阵 Screen>>，表明当前正在播放动画
6.         input [3:0] taking,  //操作中心 Operator>>，表明某个快递柜正在被
   打开，该值最初是大于快递柜号 0 到 7 的，所以以下 always 块不会进行任何操作
7.
8.         output reg [7:0] led=8'hFF, //快递柜的 LED 灯光
9.         output reg [7:0] full=8'hFF //快递柜是否被占用，为 1 表示快递柜有
   东西
10.        );
11.
12. //LED 灯开关控制
```

```
always @(posedge clk_2 or negedge rst or negedge playing)
    begin
    if(!rst) begin
     full<=8'hFF;
     led<=8'hFF;
    end
    else if(!playing) begin//动画未在播放
     full[taking]<=0;    //表明快递柜已空
     led[taking]<=0;    //led 熄灭
    end
    else                //动画正在播放
     led[taking]<=~led[taking]; //LED 按时钟频率闪烁
    end

endmodule
```

五、功能说明及资源利用情况

1. 功能说明

本实验设定拨码开关 SW7 为 rst 信号控制端，相当于系统的总开关。SW7 输出低电平时系统为初始状态，包裹全满，LED 全亮，点阵和数码管熄灭，且用户不能进行任何操作。将 SW7 拨至高电平，系统可以正常使用。用户在右侧 4×4 键盘输入数值并按下对应功能键进行相应操作，按键分布如图 11，每一个按键都有对应的音效。

1	2	3	删除
4	5	6	清空
7	8	9	确认
	0		

图 11 4×4 键盘按键分布图

若输入的值在 000 到 007 之间，并按下确认，系统将判断输入了一个有效值，点阵显示开门动画，蜂鸣器播放开门音乐，数码管清空，LED 闪烁。开门动画结束后过三秒将显示关门动画，蜂鸣器播放关门音乐。开关门期间键盘锁定，用户无法进行任何操作。动画播放结束后，此前取过包裹的 LED 灯将熄灭，并且对应的号码将不能使系统判断为有效，其他常亮 LED 对应的快递柜则仍可以用以上方法打开。在任何时候 SW7 拨至低电平都将使系统立即进入初始状态。

2. 资源利用情况

Flow Status	Successful - Sun Dec 26 22:30:19 2021
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ParcelLocker
Top-level Entity Name	ParcelLocker
Family	MAX II
Device	EPM1270T144C5
Timing Models	Final
Total logic elements	873 / 1,270 (69 %)
Total pins	59 / 116 (51 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

图 12 编译报告

Node Name	Direction	Location	I/O Bank	itter Locatior	I/O Standard	Reserved	urrent Streng	ict Preservati
beep	Output	PIN_60	4	PIN_60	3.3-V LVTTL		16mA ...ault)	
clk_in	Input	PIN_18	1	PIN_18	3.3-V LVTTL		16mA ...ault)	
ds_a[7]	Output	PIN_51	4	PIN_51	3.3-V LVTTL		16mA ...ault)	
ds_a[6]	Output	PIN_52	4	PIN_52	3.3-V LVTTL		16mA ...ault)	
ds_a[5]	Output	PIN_53	4	PIN_53	3.3-V LVTTL		16mA ...ault)	
ds_a[4]	Output	PIN_55	4	PIN_55	3.3-V LVTTL		16mA ...ault)	
ds_a[3]	Output	PIN_57	4	PIN_57	3.3-V LVTTL		16mA ...ault)	
ds_a[2]	Output	PIN_58	4	PIN_58	3.3-V LVTTL		16mA ...ault)	
ds_a[1]	Output	PIN_59	4	PIN_59	3.3-V LVTTL		16mA ...ault)	
ds_a[0]	Output	PIN_62	4	PIN_62	3.3-V LVTTL		16mA ...ault)	
ds_cat[7]	Output	PIN_31	1	PIN_31	3.3-V LVTTL		16mA ...ault)	
ds_cat[6]	Output	PIN_30	1	PIN_30	3.3-V LVTTL		16mA ...ault)	
ds_cat[5]	Output	PIN_70	4	PIN_70	3.3-V LVTTL		16mA ...ault)	
ds_cat[4]	Output	PIN_69	4	PIN_69	3.3-V LVTTL		16mA ...ault)	
ds_cat[3]	Output	PIN_68	4	PIN_68	3.3-V LVTTL		16mA ...ault)	
ds_cat[2]	Output	PIN_67	4	PIN_67	3.3-V LVTTL		16mA ...ault)	
ds_cat[1]	Output	PIN_66	4	PIN_66	3.3-V LVTTL		16mA ...ault)	
ds_cat[0]	Output	PIN_63	4	PIN_63	3.3-V LVTTL		16mA ...ault)	
g_col[7]	Output	PIN_38	4	PIN_38	3.3-V LVTTL		16mA ...ault)	
g_col[6]	Output	PIN_39	4	PIN_39	3.3-V LVTTL		16mA ...ault)	
g_col[5]	Output	PIN_40	4	PIN_40	3.3-V LVTTL		16mA ...ault)	
g_col[4]	Output	PIN_41	4	PIN_41	3.3-V LVTTL		16mA ...ault)	
g_col[3]	Output	PIN_42	4	PIN_42	3.3-V LVTTL		16mA ...ault)	
g_col[2]	Output	PIN_43	4	PIN_43	3.3-V LVTTL		16mA ...ault)	
g_col[1]	Output	PIN_44	4	PIN_44	3.3-V LVTTL		16mA ...ault)	
g_col[0]	Output	PIN_45	4	PIN_45	3.3-V LVTTL		16mA ...ault)	
kbcol[3]	Output	PIN_120	2	PIN_120	3.3-V LVTTL		16mA ...ault)	
kbcol[2]	Output	PIN_119	2	PIN_119	3.3-V LVTTL		16mA ...ault)	
kbcol[1]	Output	PIN_118	2	PIN_118	3.3-V LVTTL		16mA ...ault)	
kbcol[0]	Output	PIN_117	2	PIN_117	3.3-V LVTTL		16mA ...ault)	
kbrow[3]	Input	PIN_114	2	PIN_114	3.3-V LVTTL		16mA ...ault)	
kbrow[2]	Input	PIN_113	2	PIN_113	3.3-V LVTTL		16mA ...ault)	
kbrow[1]	Input	PIN_112	2	PIN_112	3.3-V LVTTL		16mA ...ault)	
kbrow[0]	Input	PIN_111	2	PIN_111	3.3-V LVTTL		16mA ...ault)	
led[7]	Output	PIN_73	3	PIN_73	3.3-V LVTTL		16mA ...ault)	
led[6]	Output	PIN_74	3	PIN_74	3.3-V LVTTL		16mA ...ault)	
led[5]	Output	PIN_75	3	PIN_75	3.3-V LVTTL		16mA ...ault)	
led[4]	Output	PIN_76	3	PIN_76	3.3-V LVTTL		16mA ...ault)	
led[3]	Output	PIN_77	3	PIN_77	3.3-V LVTTL		16mA ...ault)	
led[2]	Output	PIN_78	3	PIN_78	3.3-V LVTTL		16mA ...ault)	
led[1]	Output	PIN_79	3	PIN_79	3.3-V LVTTL		16mA ...ault)	
led[0]	Output	PIN_80	3	PIN_80	3.3-V LVTTL		16mA ...ault)	
r_col[7]	Output	PIN_11	1	PIN_11	3.3-V LVTTL		16mA ...ault)	
r_col[6]	Output	PIN_12	1	PIN_12	3.3-V LVTTL		16mA ...ault)	
r_col[5]	Output	PIN_13	1	PIN_13	3.3-V LVTTL		16mA ...ault)	
r_col[4]	Output	PIN_14	1	PIN_14	3.3-V LVTTL		16mA ...ault)	
r_col[3]	Output	PIN_15	1	PIN_15	3.3-V LVTTL		16mA ...ault)	
r_col[2]	Output	PIN_16	1	PIN_16	3.3-V LVTTL		16mA ...ault)	
r_col[1]	Output	PIN_21	1	PIN_21	3.3-V LVTTL		16mA ...ault)	
r_col[0]	Output	PIN_22	1	PIN_22	3.3-V LVTTL		16mA ...ault)	
row[7]	Output	PIN_1	1	PIN_1	3.3-V LVTTL		16mA ...ault)	
row[6]	Output	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
row[5]	Output	PIN_3	1	PIN_3	3.3-V LVTTL		16mA ...ault)	
row[4]	Output	PIN_4	1	PIN_4	3.3-V LVTTL		16mA ...ault)	
row[3]	Output	PIN_5	1	PIN_5	3.3-V LVTTL		16mA ...ault)	
row[2]	Output	PIN_6	1	PIN_6	3.3-V LVTTL		16mA ...ault)	
row[1]	Output	PIN_7	1	PIN_7	3.3-V LVTTL		16mA ...ault)	
row[0]	Output	PIN_8	1	PIN_8	3.3-V LVTTL		16mA ...ault)	
rst	Input	PIN_125	2	PIN_125	3.3-V LVTTL		16mA ...ault)	
<<new node>>								

图 13 管脚分配图

六、故障及问题分析

1、4×4 键盘的状态机扫描问题

这是一个令我几近崩溃的问题。最开始，我以为只需要将列扫描的输出信号和行输入信号用大括号合并起来，再用一个 case 语句就能够完美利用好键盘。但事实证明我想得太过简单，因为我的键盘输入会让数码管不断闪烁。我首先把问题锁定在了按键消抖，编写了一个简单的程序，即一个按键亮一个 LED 的方式来进行故障排查。排查后，我发现独立按键和键盘在同样的消抖模块下输出情况并不相同，因此我知道我的方向错了。但我发现键盘的输出是有规律的，同一行键盘控制的 LED 灯会不断闪烁，而其他行键盘控制的不会，就好像我按下了一排的键盘而不只是一个。此时我大概知道了问题所在，可能是按键高扫频导致键盘实际接收到的值不能跟扫频同步，而且不断扫描会导致键盘在被按下期间不断向外输出相同的值。但是降低键盘扫频是不可取的，因为这会导致极大的输入延迟，需要用户长时间按下按键，降低了使用体验。上网查阅资料后，我了解了状态机的思想，即在扫描到某一列时，若检测到该列有按键按下，就立马停止扫描，锁定键盘，标记按键坐标，并输出一个使能脉冲，使键盘只能将值输出一次。最后，我完成了键盘代码的编写。

2、快递柜开启时的输入问题

此前，我并未在点阵播放时试图锁定其它模块，导致用户在播放期间若输入了有效值，会使当前开启快递柜的操作直接结束，并立即开始下一次开启柜门的操作。这个问题并不棘手，但是我前几个方案并不太完美，因为我总是漏掉了某几个模块的锁定，导致一些小小的漏洞，比如数码管锁住了，但蜂鸣器却响了。最后我想到键盘是用户的唯一输入端，不论是哪一个操作，都需要从键盘开始，所以我只要把键盘锁定做好就能够完美完成所有模块的锁定。

3、资源节约问题

此前，每次编译后控制台都会显示很多警告。耐心检查并查阅资料后，我发现以警告数字位数过多的居多，尤其是蜂鸣器模块中记录音高的 18 位数，系统自动分配了 32 位。最后，我在每个数字前都会加上位数的限定，极大地节省板载资源。

七、总结和结论

1、总结

本次实验对我来说很痛苦也很快乐。痛苦在于每一次 Debug 都需要耐下心来，一坐几小时还不一定有结果，比如我上的倒数第二节课，本来是打算去验收的，但由于两行代码写错了，3 小时也没能检查出来，而几天后的最后一节课上只花了 10 分钟就解决了。快乐在于过程中我跟同学相互帮助拉近了关系，和完成那一刻我充满的成就感，让我感到我真的能做出一些很好玩的东西。数电实验课同时促进了我理论课的学习，让我对于同步、异步时序逻辑有了更深的理解。

2、结论

本次实验我完成了简易快递自提柜系统用户端设计的要求，学习了 Verilog 硬件开发语言，并加深了对数电理论课内容的理解。

参考文献

[1]于斌等.Verilog HDL 数字系统设计及仿真（第一版）电子工业出版社，2018.

[2]刘培植等.数字电路与逻辑设计北京：北京邮电大学出版社，2019.