

# 自主设计：两种方式实现密码锁

林莉淇 PB22051128

## 1 实验内容

编写 verilog 程序，使用矩阵键盘、LED 灯、七段数码管、拨码开关设计两种方式实现的密码锁：1. 直接将设置的密码和输入的密码存入锁存器，利用逻辑关系判断密码是否正确；2. 先内置一个密码，使用序列发生器、序列检测器检查密码是否正确。

## 2 设计分析

需要针对序列产生器、序列检测器、七段数码管、分频器、按键检测器编写不同模块。设置的密码、待验证的密码都从矩阵键盘获得，用 LED 的亮暗情况表示它们，同时在七段数码管显示。密码验证结果也用 LED 的亮暗情况表示。其中，CLK 信号由分频器分频。

1. 直接将设置的密码和输入的密码存入锁存器，利用逻辑关系判断密码是否正确：从矩阵键盘获取设置的密码和待验证的密码，分别存入两个寄存器，再比较这两个寄存器存放数据是否相同。若相同，则验证成功，表示正确的 LED 亮；若不同，则验证失败，表示错误的 LED 亮。

2. 先内置一个密码，使用序列发生器、序列检测器检查密码是否正确：从矩阵键盘获取待验证的密码 1 和密码 2，用有限状态机的方法编写成序列产生器 1 和序列产生器 2。内置密码写入序列检测器中，检测产生的序列是否包含内置密码序列即可完成密码验证。通过片选信号选择验证密码 1 还是密码 2，并将序列产生器与按键检测器相连。若序列检测器检测到序列产生器表示的密码正确，则表示正确的 LED 亮；否则不亮。

具体模块如下：

- 列扫描信号产生器
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）；
  - 输出：数据序列；
- 按键检测器
  - 输入：时钟信号 clk（实际输入是经过分频后的 clkD）、行信号 row、列信号 col；
  - 输出：按键二进制编码；
- 数码管显示器
  - 输入：4 位二进制信号 q；
  - 输出：段码信号 A,B,C,D,E,F,G；
- 分频器
  - 输入：原始时钟 clk、复位信号 reset；
  - 输出：分频后 clkD；
- 序列产生器 1
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）、待验证的第一个密码 sq1；
  - 输出：数据序列；
- 序列产生器 2
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）、待验证的第二个密码 sq2；
  - 输出：数据序列；
- 序列检测器
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）、输入数据；
  - 输出：检测结果；
- 选择器
  - 输入：2 个数据 A,B、选择信号 select；
  - 输出：选择结果 Q；

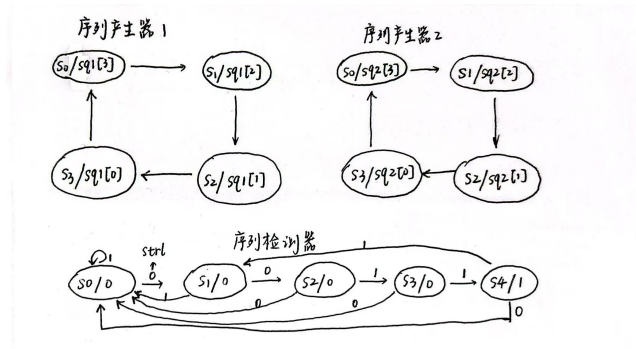


图 1: 状态转换图

### 3 Verilog 源代码

```

1 module scanout_llq(clk,reset,Moore);
2     input clk,reset;
3     output [3:0] Moore;
4     reg [3:0] Moore;
5
6     reg [1:0] st;
7     parameter S0=0,S1=1,S2=2,S3=3;
8
9     always @(posedge clk or negedge reset) begin
10         if (!reset) begin
11             st <= S0;
12         end
13         else begin
14             case (st)
15                 S0:begin st <= S1;end
16                 S1:begin st <= S2;end
17                 S2:begin st <= S3;end
18                 S3:begin st <= S0;end
19                 default st <= S0;
20             endcase
21         end
22     end
23
24     always @(st) begin
25         case (st)
26             S0:Moore <= 4'b1110;
27             S1:Moore <= 4'b1101;
28             S2:Moore <= 4'b1011;
29             S3:Moore <= 4'b0111;
30             default Moore <= 4'b1110;
31         endcase
32     end
33 endmodule

```

代码 1: 列扫描信号产生器

```

1 module get_key_llq(clk,col,row,Moore);
2     input clk;
3     input [3:0] col;
4     input [3:0] row;
5     output [3:0] Moore;
6     reg [3:0] Moore;
7
8     always @(posedge clk) begin
9         case(row)
10             4'b1110:
11                 case(col)
12                     4'b1110:Moore<=4'b0001;
13                     4'b1101:Moore<=4'b0010;
14                     4'b1011:Moore<=4'b0011;
15                     4'b0111:Moore<=4'b1010;
16                 endcase
17             4'b1101:
18                 case(col)
19                     4'b1110:Moore<=4'b0100;
20                     4'b1101:Moore<=4'b0101;
21                     4'b1011:Moore<=4'b0110;
22                     4'b0111:Moore<=4'b1011;

```

```

23         endcase
24         4'b1011:
25             case(col)
26                 4'b1110:Moore<=4'b0111;
27                 4'b1101:Moore<=4'b1000;
28                 4'b1011:Moore<=4'b1001;
29                 4'b0111:Moore<=4'b1100;
30             endcase
31         4'b0111:
32             case(col)
33                 4'b1110:Moore<=4'b1110;
34                 4'b1101:Moore<=4'b0000;
35                 4'b1011:Moore<=4'b1111;
36                 4'b0111:Moore<=4'b1101;
37             endcase
38         endcase
39     end
40 endmodule

```

代码 2: 读取按键模块

```

1 module Seg7Led(q,A,B,C,D,E,F,G);
2     input [3:0] q;
3     output A,B,C,D,E,F,G;
4     reg [6:0] ATOG;
5     always@(q)
6         begin
7             ATOG = 7'b0000000;
8             case(q)
9                 4'b0000:ATOG = 7'b0000001;
10                4'b0001:ATOG = 7'b1001111;
11                4'b0010:ATOG = 7'b0010010;
12                4'b0011:ATOG = 7'b0000110;
13                4'b0100:ATOG = 7'b1001100;
14                4'b0101:ATOG = 7'b0100100;
15                4'b0110:ATOG = 7'b0100000;
16                4'b0111:ATOG = 7'b0001111;
17                4'b1000:ATOG = 7'b0000000;
18                4'b1001:ATOG = 7'b0001000;
19                4'b1010:ATOG = 7'b0001000;
20                4'b1011:ATOG = 7'b1100000;
21                4'b1100:ATOG = 7'b0110001;
22                4'b1101:ATOG = 7'b1000010;
23                4'b1110:ATOG = 7'b0110000;
24                4'b1111:ATOG = 7'b0111000;
25                //default:ATOG = 7'b1111111;
26            endcase
27        end
28
29        assign A = ATOG[6];
30        assign B = ATOG[5];
31        assign C = ATOG[4];
32        assign D = ATOG[3];
33        assign E = ATOG[2];
34        assign F = ATOG[1];
35        assign G = ATOG[0];
36
37 endmodule

```

代码 3: 数码管显示

```

1 module FD_11q(clk,clkD,reset);
2     input clk;
3     input reset;
4     output clkD;
5     reg [20:0] cnt = 0;
6     reg clkD;
7     always @(posedge clk) begin
8         if(reset == 1'b0) begin
9             cnt <= 20'd0;
10            clkD <=0;
11        end
12        else begin
13            if(cnt==20'd100000) begin
14                cnt <=20'd0;
15                clkD <=~clkD;
16            end
17            else begin
18                cnt <=cnt+1;

```

```

19         end
20     end
21 end
22 endmodule

```

代码 4: 分频器

```

1 module FSM1_1lq(clk,reset,Moore,sq1);
2     input clk,reset;
3     input [3:0] sq1;
4     output [1:0] Moore;
5     reg [1:0] Moore;
6     parameter S0=0,S1=1,S2=2,S3=3;
7     reg [3:0] st;
8
9     always @(posedge clk or negedge reset) begin
10         if (!reset) begin
11             st <= S0;
12         end
13         else begin
14             case (st)
15                 S0:begin st <= S1;end
16                 S1:begin st <= S2;end
17                 S2:begin st <= S3;end
18                 S3:begin st <= S0;end
19                 default st <= S3;
20             endcase
21         end
22     end
23
24     always @(st) begin
25         case (st)
26             S0:Moore <= sq1[3];
27             S1:Moore <= sq1[2];
28             S2:Moore <= sq1[1];
29             S3:Moore <= sq1[0];
30             default Moore <= 1'b0;
31         endcase
32     end
33 endmodule

```

代码 5: 序列产生器 1

```

1 module FSM2_1lq(clk,reset,Moore,sq2);
2     input clk,reset;
3     input [3:0] sq2;
4     output [1:0] Moore;
5     reg [1:0] Moore;
6     parameter S0=0,S1=1,S2=2,S3=3;
7     reg [3:0] st;
8
9     always @(posedge clk or negedge reset) begin
10         if (!reset) begin
11             st <= S0;
12         end
13         else begin
14             case (st)
15                 S0:begin st <= S1;end
16                 S1:begin st <= S2;end
17                 S2:begin st <= S3;end
18                 S3:begin st <= S0;end
19                 default st <= S3;
20             endcase
21         end
22     end
23
24     always @(st) begin
25         case (st)
26             S0:Moore <= sq2[3];
27             S1:Moore <= sq2[2];
28             S2:Moore <= sq2[1];
29             S3:Moore <= sq2[0];
30             default Moore <= 1'b0;
31         endcase
32     end
33 endmodule

```

代码 6: 序列产生器 2

```

1 module seq_detector_llq(clk,reset,Moore,str1);
2     input clk,reset,str1;
3     output [1:0] Moore;
4     reg [1:0] Moore;
5     parameter S0=0,S1=1,S2=2,S3=3,S4=4;
6     reg [3:0] st;
7
8     always @(posedge clk or negedge reset) begin
9         if (!reset) begin
10             st <= S0;
11         end
12         else begin
13             case (st)
14                 S0:begin
15                     if(str1 == 1'b0) st <= S1;
16                     else st <= S0; end
17                 S1:begin
18                     if(str1 == 1'b0) st <= S2;
19                     else st <= S0; end
20                 S2:begin
21                     if(str1 == 1'b1) st <= S3;
22                     else st <= S0; end
23                 S3:begin
24                     if(str1 == 1'b1) st <= S4;
25                     else st <= S0; end
26                 S4:begin
27                     if(str1 == 1'b1) st <= S1;
28                     else st <= S0; end
29                 default st <= S0;
30             endcase
31         end
32     end
33
34     always @(st) begin
35         case (st)
36             S0:Moore <= 1'b0;
37             S1:Moore <= 1'b0;
38             S2:Moore <= 1'b0;
39             S3:Moore <= 1'b0;
40             S4:Moore <= 1'b1;
41             default Moore <= 1'b0;
42         endcase
43     end
44 endmodule

```

代码 7: 序列检测器

```

1 module MUX2T01_llq(A,B,select,Q);
2     input A,B,select;
3     output Q;
4     reg Q;
5
6     always @(A or B or select) begin
7         begin
8             case (select)
9                 1'b0:Q <= A;
10                1'b1:Q <= B;
11            endcase
12        end
13    end
14 endmodule

```

代码 8: 选择器

```

1 module FPGA_EXP_FINAL(keyset,keyinput,check,key_set,key_input,reset,clk,row,col,rightled,wrongled,
2 A,B,C,D,E,F,G,LED,sq1,sq2,sqin1,sqin2,sqrightled,select,sqcheck);
3     input reset,clk,keyset,keyinput,check,sqin1,sqin2,select,sqcheck;
4     input [3:0] row;
5     output [3:0] col;
6
7     output [3:0] key_input;
8     reg [3:0] key_input;
9     output [3:0] key_set;
10    reg [3:0] key_set;
11
12    output [3:0] sq1;
13    reg [3:0] sq1;
14    output [3:0] sq2;

```

```

15     reg [3:0] sq2;
16
17     output A,B,C,D,E,F,G,LED;
18     output reg rightled,wrongled,sqrightled;
19     wire [3:0] Moore;
20     wire x1,x2,x,sqright,clkD;
21
22     FD_llq FD(clk,clkD,reset);
23     scanout_llq scanout(clkD,reset,col);
24     get_key_llq get_key(clkD,col,row,Moore);
25     Seg7Led Seg7Led(key_input,A,B,C,D,E,F,G);
26     FSM1_llq FSM1(clkD,reset,x1,sq1);
27     FSM2_llq FSM2(clkD,reset,x2,sq2);
28     MUX2T01_llq MUX2T01(x1,x2,select,x); //select=0,验证密码1
29     seq_detector_llq seq_detector(clkD,reset,sqright,x);
30
31     always @(posedge clkD) begin
32         if(reset==0) begin //复位
33             key_set <= 4'b0;
34             key_input <= 4'b0;
35             sq1 <= 4'b0;
36             sq2 <= 4'b0;
37         end
38         else if(reset == 1) begin
39             if(keyset) begin //设置密码
40                 key_set <= Moore;
41             end
42             else begin
43                 key_set <= key_set;
44             end
45
46             if(keyinput) begin //输入待验证密码
47                 key_input <= Moore;
48             end
49             else begin
50                 key_input <= key_input;
51             end
52
53             if(check) begin //方式1比较寄存器密码值
54                 if(key_set[3]==key_input[3] && key_set[2]==key_input[2] && key_set[1]==key_input[1] &&
55                    key_set[0]==key_input[0]) begin
56                     rightled <= 1;
57                 end
58                 else begin
59                     wrongled <= 1;
60                 end
61             end
62
63             if(sqin1) begin //输入待验证密码1
64                 sq1 <= Moore;
65             end
66             else begin
67                 sq1 <= sq1;
68             end
69
70             if(sqin2) begin //输入待验证密码2
71                 sq2 <= Moore;
72             end
73             else begin
74                 sq2 <= sq2;
75             end
76
77             if(sqcheck) begin //验证输入的密码
78                 sqrightled <= sqright;
79             end
80         end
81     end
82     assign LED=1;
83 endmodule
84

```

代码 9: 顶层函数（仿真时去掉分频器，直接给仿真代码中设置的时钟信号）

```

1  `timescale 10ns/1ns
2  module FPGA_EXP_FINAL_tb();
3      reg clk,clkD,reset,keyset,keyinput,check,sqin1,sqin2,select,sqcheck;
4      reg [3:0] row;
5

```

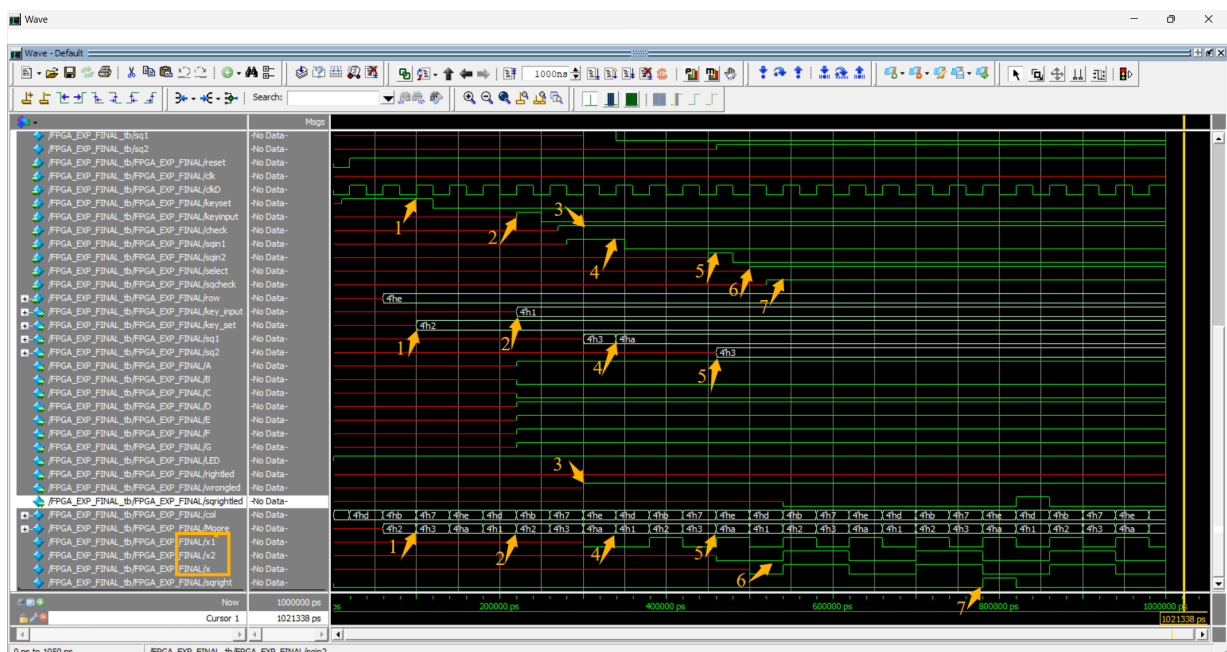
```

6 wire rightled,wrongled,sqrighbled;
7
8 initial #0 clkD=0;
9 initial #0 reset=0;
10 initial #2 clkD=0;
11 initial #2 reset=1;
12
13
14 always #2 clkD=~clkD;
15
16 always begin
17     #1 keyset=1;
18     #5 row=4'b1110;
19     #6 keyset=0;
20     #10 keyinput=1;
21     #2 row=4'b1110;
22     #1 keyinput=0;
23     #2 check=1'b1;
24
25     #1 sqin1=1;
26     #5 row=4'b1110;
27     #2 sqin1=0;
28     #10 sqin2=1;
29     #2 row=4'b1110;
30     #1 sqin2=0;
31
32     #2 select=1'b1;
33     #2 sqcheck=1'b1;
34
35     #70 $stop;
36 end
37
38 FPGA_EXP_FINAL FPGA_EXP_FINAL(keyset,keyinput,check,key_set,key_input,reset,clk,
39 row,col,rightled,wrongled,A,B,C,D,E,F,G,LED,sq1,sq2,sqin1,sqin2,sqrighbled,select,sqcheck);
40 endmodule

```

## 4 仿真结果记录

1. 设置密码信号 keyset=1 且 clkD 上升沿时, 矩阵键盘读取的值 Moore 为 2, 相应的 key\_set 寄存器存入 2 的二进制编码 0010, 同时 key\_set (key\_set[0]-key\_set[3]) 分别连接四个 LED 灯, 对应可现实亮暗情况以直观显示 key\_set



2. 输入验证密码信号  $keyinput=1$  且  $clkD$  上升沿时，矩阵键盘读取的值  $Moore$  为 1，相应的  $key\_input$  寄存器存入 1 的二进制编码 0001，同时  $key\_input$  ( $key\_input[0]-key\_input[3]$ ) 分别连接四个 LED 灯，对应可现实亮暗情况以直观显示  $key\_input$
3. 验证信号  $check=1$  且  $clkD$  上升沿时，比较  $key\_set$  寄存器和  $key\_input$  寄存器的值，二者不相等， $wrongled=1$ ， $rightled$  无变化。同时  $wrongled$  和  $rightled$  分别连接两个 LED 灯，对应可现实亮暗情况以直观显示密码验证正确与否
4. 输入第一个密码信号  $sqin1=1$  且  $clkD$  上升沿时，矩阵键盘读取的值  $Moore$  为 a，相应的  $sq1$  寄存器存入 a 的二进制编码 1010，同时  $sq1$  ( $sq1[0]-sq1[3]$ ) 分别连接四个 LED 灯，对应可现实亮暗情况以直观显示  $sq1$ 。序列产生器 1 产生 1010 的序列  $x1$
5. 输入第二个密码信号  $sqin2=1$  且  $clkD$  上升沿时，矩阵键盘读取的值  $Moore$  为 3，相应的  $sq2$  寄存器存入 3 的二进制编码 0011，同时  $sq2$  ( $sq2[0]-sq2[3]$ ) 分别连接四个 LED 灯，对应可现实亮暗情况以直观显示  $sq2$ ，序列产生器 2 产生 0011 的序列  $x2$
6. 片选信号  $select=1$  且  $clkD$  上升沿时，选中第二个密码，二选一选择器输出结果  $x$  与  $x2$  相同
7. 验证密码信号  $sqcheck=1$  且  $clkD$  上升沿时，用序列检测器验证选中的密码（第二个密码，为 3）与内置密码 3 是否相同。完整验证 0011 之后，密码相同， $sqright=1$ ，同时  $sqright$  连接 LED 灯，对应可现实亮暗情况以直观显示密码验证正确与否

## 5 RTL 结构图

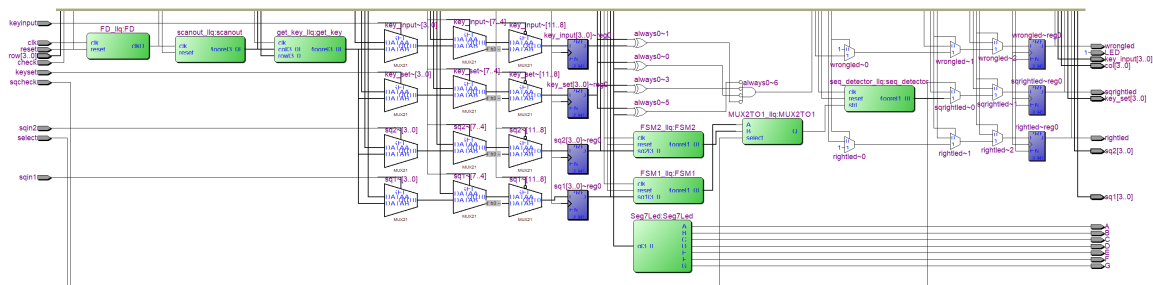


图 3: 总程序

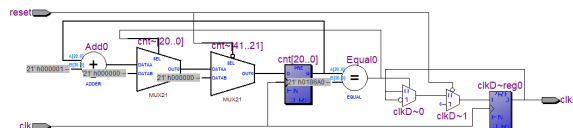


图 4: 分频器

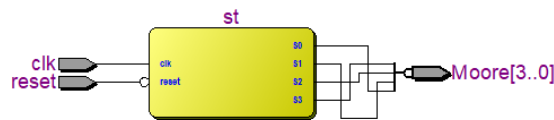


图 5: 列扫描信号产生器

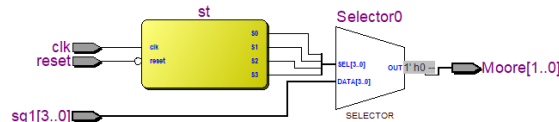


图 6: 序列产生器 1（待验证第一个密码）



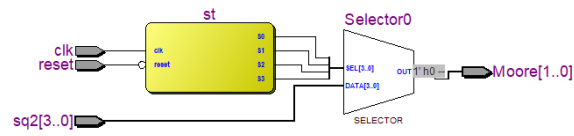


图 7: 序列产生器 2 (待验证第二个密码)

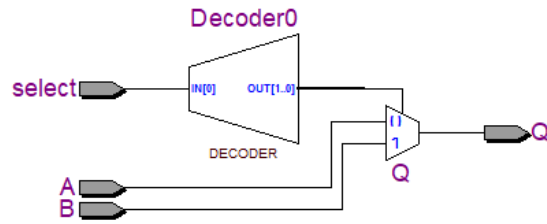


图 8: 二选一选择器



图 9: 序列检测器

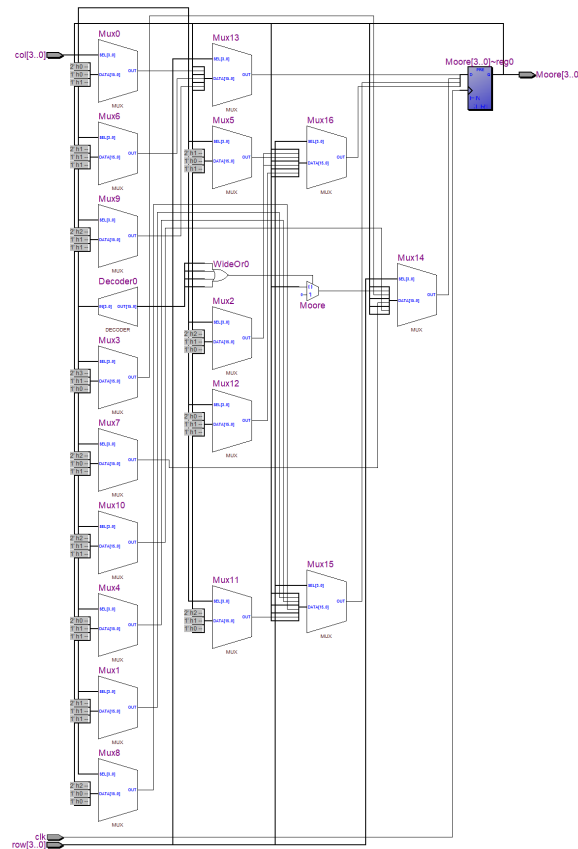


图 10: 获取按键模块

## 6 FPGA 验证结果记录

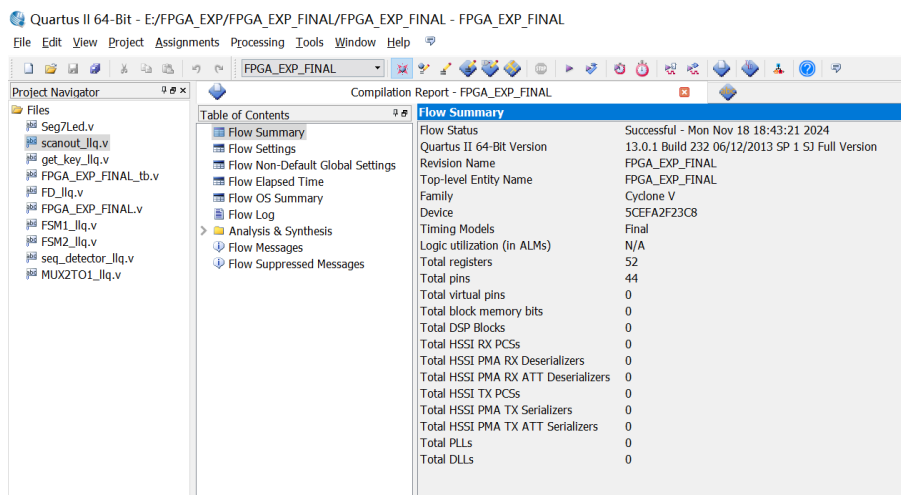


图 11: 资源占用

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
A[0]	Output	PIN_X19	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
B[0]	Output	PIN_H18	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
C[0]	Output	PIN_K20	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
D[0]	Output	PIN_M18	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
E[0]	Output	PIN_E16	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
F[0]	Output	PIN_G13	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
G[0]	Output	PIN_G17	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
LED	Output	PIN_E12	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
check	Input	PIN_U12	3B	B3B_NO	2.5 V ...fault		12mA ...ault	1 (default)
clk	Input	PIN_W16	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
col[3]	Output	PIN_AB22	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
col[2]	Output	PIN_AB20	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
col[1]	Output	PIN_AA20	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
col[0]	Output	PIN_Y17	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_input[3]	Output	PIN_C15	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_input[2]	Output	PIN_H13	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_input[1]	Output	PIN_F14	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_input[0]	Output	PIN_J13	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_set[3]	Output	PIN_D12	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_set[2]	Output	PIN_G12	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_set[1]	Output	PIN_B13	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
key_set[0]	Output	PIN_C13	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
keyinput	Input	PIN_R11	3B	B3B_NO	2.5 V ...fault		12mA ...ault	1 (default)
keyset	Input	PIN_Y11	3B	B3B_NO	2.5 V ...fault		12mA ...ault	1 (default)
reset	Input	PIN_T19	5A	B5A_NO	2.5 V ...fault		12mA ...ault	1 (default)
rightled	Output	PIN_H15	7A	B7A_NO	2.5 V ...fault		12mA ...ault	1 (default)
row[3]	Input	PIN_V21	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
row[2]	Input	PIN_W21	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
row[1]	Input	PIN_Y19	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
row[0]	Input	PIN_Y21	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
select	Input	PIN_Y15	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq1[3]	Output	PIN_K17	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq1[2]	Output	PIN_L22	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq1[1]	Output	PIN_L18	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq1[0]	Output	PIN_L17	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq2[3]	Output	PIN_M20	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq2[2]	Output	PIN_M22	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq2[1]	Output	PIN_P19	5A	B5A_NO	2.5 V ...fault		12mA ...ault	1 (default)
sq2[0]	Output	PIN_N20	5B	B5B_NO	2.5 V ...fault		12mA ...ault	1 (default)
sqcheck	Input	PIN_Y14	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
sqin1	Input	PIN_Y13	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
sqin2	Input	PIN_AB13	4A	B4A_NO	2.5 V ...fault		12mA ...ault	1 (default)
srightled	Output	PIN_R21	5A	B5A_NO	2.5 V ...fault		12mA ...ault	1 (default)
wrongled	Output	PIN_T20	5A	B5A_NO	2.5 V ...fault		12mA ...ault	1 (default)

图 12: 管脚锁定情况

动态结果的文字描述：先拨动 DIP15=0，复位。再拨动 DIP15=1 并保持不变，使能列扫描信号产生器和分频器、序列产生器、检测器。

随后按照仿真结果的 1-7 步骤分别拨动开关并在矩阵键盘输入数字，七段数码管显示输入的密码，同时 LED 灯也会以亮暗情况直观显示方式一的设置密码、输入密码、验证结果和方式二的密码一、密码二、验证结果。

## 7 实验总结

通过这次实验，我明白了分模块调用的重要性。针对不同器件写不同模块再进行调用，可以使代码结构更加清晰。

例化语句不能写在 always 内，在外例化后，只要有触发信号变化就会使得例化语句的输出变化，进而获得目标输出。

同时，例化模块的输出只能用 wire 类型，不能用 reg 类型，否则编译报错。

在仿真的时候，仿真信号的设置对系统功能验证非常重要。为了得到内置密码 3 的矩阵按键输入，我多次计算对应的信号延时，加深对仿真信号的理解。

我还碰到上板时按下按键，数码管和 LED 一直没有反应的情况。经检查，分频器过度分频，在按下的时候没遇到上升沿，检测不到，需要长按或修改分频系数。

在这次实验中，我收获良多。之后的应用中我会更注意让代码结构更清晰，仿真更加全面，以便于最后的硬件验证。