

串行密码锁

计 93 王哲凡 2019011200

一、实验内容与要求

用状态机设计一个串行密码锁，包括以下功能：

- 密码预置：创建管理员万能密码（4 位十六进制）以备管理，它在任何时候都能开锁。
- 设置密码：用户可以设置 4 位十六进制密码，验证密码错误后锁定此功能直到再次开锁。
- 验证密码：用户串行输入密码，密码符合用户密码或管理员密码（报警灯亮时只能使用管理员密码）则亮开锁灯，均不符合则点亮错误灯。
- 系统报警：连续开锁三次失败后点亮报警灯，此时锁定设置密码与用户密码验证功能，只有输入管理员密码才可以解锁。

输入输出端包括：

1. `code[3:0]`：输入，设置密码或验证密码时用户输入的一位十六进制密码。
2. `clk`：输入，确认按钮，在对应模式下按下一次 `clk` 设置或验证一位密码，上边沿触发。
3. `rst`：输入，复位和初始化按钮，设置或验证密码开始前需按一次，设置或验证密码时也可按此按钮回到初始状态。
4. `mode[1:0]`：输入，高位为 1 代表不动，否则当低位为 0 时为设置密码模式，低位为 1 时为验证密码模式。
5. `unlock`：输出，开锁灯。
6. `alarm`：输出，报警灯。
7. `error`：输出，错误灯。

二、状态机设计

通过 `parameter` 设置 `admin_pwd` 表示固定的管理员密码（代码中设置为 `0xFFFF`）；`user_pwd` 存储用户密码。

`user_state` 和 `admin_state` 分别表示当前用户密码检测和管理员密码检测是否可用，条件为：

- `user_state == 1'b1`: `alarm == 1'b0` 且之前位的输入与用户密码相同。
- `admin_state == 1'b1`: 之前位的输入与管理员密码相同。

通过 `state` 表示状态：

- `3'd0`:
 - 等待状态，也即首位验证密码或首位设置密码状态。

- 当 `mode == 2'b00` 且设置密码可用时，设置第一位用户密码，切换至 `3'd1` 状态；
- 当 `mode == 2'b01` 时，验证第一位密码，验证成功则切换至 `3'd4` 状态，否则进行失败处理。
- `3'd1, 3'd2, 3'd3`:
 - 设置状态，`mode == 2'b00` 且设置可用时，设置第二到四位密码，并切换到下一个状态。
 - 对于 `3'd1, 3'd2`，`state` 加一。
 - 对于 `3'd3`，`state` 变为 `3'd7`。
- `3'd4, 3'd5, 3'd6`:
 - 验证状态，`mode == 2'b01` 时，验证第二到四位密码。
 - 如果验证成功则 `state` 加一。
 - 如果验证失败则进行失败处理。
- `3'd7`:
 - 解锁状态，此时开锁灯亮，报警灯熄灭。

当按下 `rst` 键即 `rst == 1'b1` 时：

- 无论当前状态如何，`state` 切换至 `3'd0`。
- 错误灯和开锁灯熄灭。

失败处理：

- `state` 切换至 `3'd0`。
- 如果 `cnt` 大于 1 则报警灯亮，`cnt` 清零。
- 否则 `cnt` 加一。

`cnt` 表示错误计数器，正常为 0 ~ 2 的整数，表示连续错误次数，当 `cnt` 大于 1 时再次发生错误，则亮报警灯。

三、实验代码及简要原理分析

```

1  module CipherTrunk(
2      input wire [3:0] code,
3      input wire [1:0] mode,
4      input wire clk,
5      input wire rst,
6      output reg unlock,
7      output reg alarm,
8      output reg error
9  );
10
11  parameter [15:0] admin_pwd = 16'hffff;
12  reg [15:0] user_pwd;
13  reg admin_state;
14  reg user_state;
15  reg [2:0] state;
16  reg [1:0] cnt;

```

```

17
18 initial cnt = 2'b00;
19
20 always @(posedge clk or posedge rst) begin
21     if (rst) begin
22         unlock <= 1'b0;
23         error <= 1'b0;
24         state <= 3'b0;
25     end
26     else if (mode == 2'b00 && alarm == 1'b0 && cnt == 2'b00) begin
27         case (state)
28             3'd0: begin
29                 user_pwd[3:0] <= code;
30                 state <= 3'd1;
31             end
32             3'd1: begin
33                 user_pwd[7:4] <= code;
34                 state <= 3'd2;
35             end
36             3'd2: begin
37                 user_pwd[11:8] <= code;
38                 state <= 3'd3;
39             end
40             3'd3: begin
41                 user_pwd[15:12] <= code;
42                 state <= 3'd7;
43                 unlock <= 1'b1;
44             end
45             default;;
46         endcase
47     end
48     else if (mode == 2'b01) begin
49         case (state)
50             3'd0:
51                 if ((code == user_pwd[3:0] && alarm == 1'b0) || code ==
admin_pwd[3:0]) begin
52                     if (code == user_pwd[3:0] && alarm == 1'b0)
53                         user_state <= 1'b1;
54                     else
55                         user_state <= 1'b0;
56                     if (code == admin_pwd[3:0])
57                         admin_state <= 1'b1;
58                     else
59                         admin_state <= 1'b0;
60                     state <= 3'd4;
61                     error <= 1'b0;
62                 end
63             else begin

```

```

64         error <= 1'b1;
65         if (cnt > 2'd1) begin
66             alarm <= 1'b1;
67             cnt <= 2'd0;
68         end
69         else
70             cnt <= cnt + 1'b1;
71     end
72     3'd4:
73         if ((code == user_pwd[7:4] && user_state == 1'b1) || (code ==
admin_pwd[7:4] && admin_state == 1'b1)) begin
74             if (state == 3'd6) begin
75                 unlock <= 1'b1;
76                 alarm <= 1'b0;
77                 cnt <= 2'd0;
78             end
79             state <= 3'd5;
80             if (code != user_pwd[7:4])
81                 user_state <= 1'b0;
82             if (code != admin_pwd[7:4])
83                 admin_state <= 1'b0;
84         end
85         else begin
86             error <= 1'b1;
87             state <= 1'b0;
88             if (cnt > 2'd1) begin
89                 alarm <= 1'b1;
90                 cnt <= 2'd0;
91             end
92             else
93                 cnt <= cnt + 1'b1;
94         end
95     3'd5:
96         if ((code == user_pwd[11:8] && user_state == 1'b1) || (code ==
admin_pwd[11:8] && admin_state == 1'b1)) begin
97             if (state == 3'd6) begin
98                 unlock <= 1'b1;
99                 alarm <= 1'b0;
100                 cnt <= 2'd0;
101             end
102             state <= 3'd6;
103             if (code != user_pwd[11:8])
104                 user_state <= 1'b0;
105             if (code != admin_pwd[11:8])
106                 admin_state <= 1'b0;
107         end
108         else begin
109             error <= 1'b1;

```

```

110         state <= 1'b0;
111         if (cnt > 2'd1) begin
112             alarm <= 1'b1;
113             cnt <= 2'd0;
114         end
115         else
116             cnt <= cnt + 1'b1;
117         end
118     3'd6:
119         if ((code == user_pwd[15:12] && user_state == 1'b1) || (code ==
admin_pwd[15:12] && admin_state == 1'b1)) begin
120             if (state == 3'd6) begin
121                 unlock <= 1'b1;
122                 alarm <= 1'b0;
123                 cnt <= 2'd0;
124             end
125             state <= 3'd7;
126             if (code != user_pwd[15:12])
127                 user_state <= 1'b0;
128             if (code != admin_pwd[15:12])
129                 admin_state <= 1'b0;
130         end
131         else begin
132             error <= 1'b1;
133             state <= 1'b0;
134             if (cnt > 2'd1) begin
135                 alarm <= 1'b1;
136                 cnt <= 2'd0;
137             end
138             else
139                 cnt <= cnt + 1'b1;
140         end
141         default;;
142     endcase
143 end
144 end
145
146 endmodule
147

```

具体寄存器的含义均可见状态机设计部分。

四、仿真与延迟探究



上述仿真包括了：

- 密码设置；
- 用户密码解锁；
- 管理员密码解锁；
- 连续三次错误报警；
- 管理员密码解除报警并解锁。

具体流程为：

1. 初始先设置用户密码为 $0x1248$ ，此时开锁灯亮；
2. 清零验证用户密码 $0x1248$ ，开锁灯亮；
3. 清零验证管理员密码 $0xFFFF$ ，开锁灯亮；
4. 连续 3 次输入 2 失败，报警灯亮，并且每次出错导致错误灯亮；
5. 清零验证管理员密码 $0xFFFF$ ，报警灯灭，开锁灯亮。

五、实验收获

1. 学习了状态机的使用，利用状态机可以更好表示电路抽象含义。
2. 尝试练习了更复杂的时序电路，对于现态、次态概念有了更深的认识。