串行密码锁实验报告

2017011341, 陈旭

2019 年 5 月

1 实验目的

- 学习使用状态机控制电路工作，在不同状态下完成相应的功能；
- 进一步掌握时序逻辑电路的基本分析和设计方法；
- 学会利用仿真软件实现对数字电路的逻辑功能进行验证和分析。

2 实验内容

- 设计一个 4 位 16 进制串行密码锁，支持：设置密码、验证密码解锁。
- 提高部分：管理员万用密码、错误次数警报。

3 代码与分析

3.0.1 代码

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  use ieee.std_logic_unsigned.all;
5
6  entity lock is
7    port(
8      clock : in std_logic;
9      code : in std_logic_vector(3 downto 0);
10     mode : in std_logic_vector(1 downto 0);
11     clk, rst : in std_logic;
12     unlock, setting : out std_logic;
13     alarm, error : buffer std_logic;
14     lights : out std_logic_vector(3 downto 0)
```

```
15    );
16    type T is array (3 downto 0) of integer;
17  end lock;
18
19  architecture lock of lock is
20    signal password : T;
21    signal manager : T := (9, 6, 8, 4);
22    signal state : integer := 0;
23    signal error_count : integer := 0;
24    signal click : std_logic;
25  begin
26    process(clock)
27    begin
28      if clock'event and clock='1' then
29        click <= clk;
30      end if;
31    end process;
32
33    process(click, rst)
34    begin
35      if click'event and click='1' then
36        if mode="00" then
37          case state is
38            when 0 => password(0) <= CONV_INTEGER(code); state <= 1;
39            when 1 => password(1) <= CONV_INTEGER(code); state <= 2;
40            when 2 => password(2) <= CONV_INTEGER(code); state <= 3;
41            when 3 => password(3) <= CONV_INTEGER(code); state <= 4;
42            when 4 =>
43              unlock <= '0';
44              if CONV_INTEGER(code) = manager(0) then
45                state <= 5;
46              else
47                state <= 4;
48              end if;
49            when 5 =>
50              if CONV_INTEGER(code) = manager(1) then
51                state <= 6;
52              else
53                state <= 4;
```

```
54              end if;
55            when 6 =>
56              if CONV_INTEGER(code) = manager(2) then
57                state <= 7;
58              else
59                state <= 4;
60              end if;
61            when 7 =>
62              if CONV_INTEGER(code) = manager(3) then
63                state <= 18;
64                alarm <= '0';
65                error_count <= 0;
66              else
67                state <= 4;
68              end if;
69            when 18 =>
70              state <= 0;
71              when others => state <= 4; unlock <= '0'; error <= '0';
72          end case;
73        elsif mode="01" then
74          case state is
75            when 0 => password(0) <= CONV_INTEGER(code); state <= 1;
76            when 1 => password(1) <= CONV_INTEGER(code); state <= 2;
77            when 2 => password(2) <= CONV_INTEGER(code); state <= 3;
78            when 3 => password(3) <= CONV_INTEGER(code); state <= 8;
79            when 8 =>
80              unlock <= '0';
81              if alarm='0' then
82                if CONV_INTEGER(code) = password(0) and
83                    CONV_INTEGER(code) = manager(0) then
84                  state <= 9;
85                elsif CONV_INTEGER(code) = password(0) then
86                  state <= 12;
87                elsif CONV_INTEGER(code) = manager(0) then
88                  state <= 15;
89                else
90                  state <= 8;
91                  error <= '1';
92                  if error_count + 1 = 3 then
```

```
93                  alarm <= '1';
94                  error_count <= 0;
95                else
96                  error_count <= error_count + 1;
97                end if;
98              end if;
99            end if;
100         when 9 =>
101           if CONV_INTEGER(code) = password(1) and
102               CONV_INTEGER(code) = manager(1) then
103             state <= 10;
104           elsif CONV_INTEGER(code) = password(1) then
105             state <= 13;
106           elsif CONV_INTEGER(code) = manager(1) then
107             state <= 16;
108           else
109             state <= 8;
110             error <= '1';
111             if error_count + 1 = 3 then
112               alarm <= '1';
113               error_count <= 0;
114             else
115               error_count <= error_count + 1;
116             end if;
117           end if;
118         when 10 =>
119           if CONV_INTEGER(code) = password(2) and
120               CONV_INTEGER(code) = manager(2) then
121             state <= 11;
122           elsif CONV_INTEGER(code) = password(2) then
123             state <= 14;
124           elsif CONV_INTEGER(code) = manager(2) then
125             state <= 17;
126           else
127             state <= 8;
128             error <= '1';
129             if error_count + 1 = 3 then
130               alarm <= '1';
131               error_count <= 0;
```

```
132                 else
133                   error_count <= error_count + 1;
134                 end if;
135               end if;
136             when 11 =>
137               if CONV_INTEGER(code) = password(3) or
138                   CONV_INTEGER(code) = manager(3) then
139                 state <= 8;
140                 unlock <= '1';
141                 error_count <= 0;
142                 error <= '0';
143               else
144                 state <= 8;
145                 error <= '1';
146                 if error_count + 1 = 3 then
147                   alarm <= '1';
148                   error_count <= 0;
149                 else
150                   error_count <= error_count + 1;
151                 end if;
152               end if;
153             when 12 =>
154               if CONV_INTEGER(code) = password(1) then
155                 state <= 13;
156               else
157                 state <= 8;
158                 error <= '1';
159                 if error_count + 1 = 3 then
160                   alarm <= '1';
161                   error_count <= 0;
162                 else
163                   error_count <= error_count + 1;
164                 end if;
165               end if;
166             when 13 =>
167               if CONV_INTEGER(code) = password(2) then
168                 state <= 14;
169               else
170                 state <= 8;
```

```
171                 error <= '1';
172                 if error_count + 1 = 3 then
173                   alarm <= '1';
174                   error_count <= 0;
175                 else
176                   error_count <= error_count + 1;
177                 end if;
178               end if;
179             when 14 =>
180               if CONV_INTEGER(code) = password(3) then
181                 state <= 8;
182                 unlock <= '1';
183                 error_count <= 0;
184                 error <= '0';
185               else
186                 state <= 8;
187                 error <= '1';
188                 if error_count + 1 = 3 then
189                   alarm <= '1';
190                   error_count <= 0;
191                 else
192                   error_count <= error_count + 1;
193                 end if;
194               end if;
195             when 15 =>
196               if CONV_INTEGER(code) = manager(1) then
197                 state <= 16;
198               else
199                 state <= 8;
200                 error <= '1';
201                 if error_count + 1 = 3 then
202                   alarm <= '1';
203                   error_count <= 0;
204                 else
205                   error_count <= error_count + 1;
206                 end if;
207               end if;
208             when 16 =>
209               if CONV_INTEGER(code) = manager(2) then
```

```
210               state <= 17;
211             else
212               state <= 8;
213               error <= '1';
214               if error_count + 1 = 3 then
215                 alarm <= '1';
216                 error_count <= 0;
217               else
218                 error_count <= error_count + 1;
219               end if;
220             end if;
221           when 17 =>
222             if CONV_INTEGER(code) = manager(3) then
223               state <= 8;
224               unlock <= '1';
225               error_count <= 0;
226               error <= '0';
227             else
228               state <= 8;
229               error <= '1';
230               if error_count + 1 = 3 then
231                 alarm <= '1';
232                 error_count <= 0;
233               else
234                 error_count <= error_count + 1;
235               end if;
236             end if;
237           when others => state <= 8; unlock <= '0'; error <= '0';
238         end case;
239       end if;
240     end if;
241   end process;
242
243   process(state)
244   begin
245     if state < 4 then
246       setting <= '1';
247     else
248       setting <= '0';
```

```
249       end if;
250       case state is
251         when 0 =>
252           lights(0) <= '1'; lights(1) <= '0';
253           lights(2) <= '0'; lights(3) <= '0';
254         when 1 =>
255           lights(0) <= '1'; lights(1) <= '1';
256           lights(2) <= '0'; lights(3) <= '0';
257         when 2 =>
258           lights(0) <= '1'; lights(1) <= '1';
259           lights(2) <= '1'; lights(3) <= '0';
260         when 3 =>
261           lights(0) <= '1'; lights(1) <= '1';
262           lights(2) <= '1'; lights(3) <= '1';
263         when 4 =>
264           lights(0) <= '1'; lights(1) <= '0';
265           lights(2) <= '0'; lights(3) <= '0';
266         when 5 =>
267           lights(0) <= '1'; lights(1) <= '1';
268           lights(2) <= '0'; lights(3) <= '0';
269         when 6 =>
270           lights(0) <= '1'; lights(1) <= '1';
271           lights(2) <= '1'; lights(3) <= '0';
272         when 7 =>
273           lights(0) <= '1'; lights(1) <= '1';
274           lights(2) <= '1'; lights(3) <= '1';
275         when 8 =>
276           lights(0) <= '1'; lights(1) <= '0';
277           lights(2) <= '0'; lights(3) <= '0';
278         when 9 =>
279           lights(0) <= '1'; lights(1) <= '1';
280           lights(2) <= '0'; lights(3) <= '0';
281         when 10 =>
282           lights(0) <= '1'; lights(1) <= '1';
283           lights(2) <= '1'; lights(3) <= '0';
284         when 11 =>
285           lights(0) <= '1'; lights(1) <= '1';
286           lights(2) <= '1'; lights(3) <= '1';
287         when 12 =>
```

```
288           lights(0) <= '1'; lights(1) <= '1';
289           lights(2) <= '0'; lights(3) <= '0';
290         when 13 =>
291           lights(0) <= '1'; lights(1) <= '1';
292           lights(2) <= '1'; lights(3) <= '0';
293         when 14 =>
294           lights(0) <= '1'; lights(1) <= '1';
295           lights(2) <= '1'; lights(3) <= '1';
296         when 15 =>
297           lights(0) <= '1'; lights(1) <= '1';
298           lights(2) <= '0'; lights(3) <= '0';
299         when 16 =>
300           lights(0) <= '1'; lights(1) <= '1';
301           lights(2) <= '1'; lights(3) <= '0';
302         when 17 =>
303           lights(0) <= '1'; lights(1) <= '1';
304           lights(2) <= '1'; lights(3) <= '1';
305         when 18 =>
306           lights(0) <= '0'; lights(1) <= '0';
307           lights(2) <= '0'; lights(3) <= '0';
308         when others => NULL;
309       end case;
310     end process;
311   end lock;
```

3.0.2 分析

状态介绍:

- 0: 开始设置密码
- 1: 设置第二位密码
- 2: 设置第三位密码
- 3: 设置第四位密码
- 4: 管理模式，开始输入管理员密码
- 5: 输入第二位管理员密码
- 6: 输入第三位管理员密码
- 7: 输入第四位管理员密码

- 8: 解锁模式，开始输入解锁密码
- 9: 输入第二位解锁密码，可接受管理员密码与用户密码
- 10: 输入第三位解锁密码，可接受管理员密码与用户密码
- 11: 输入第四位解锁密码，可接受管理员密码与用户密码
- 12: 输入第二位解锁密码，可接受用户密码
- 13: 输入第三位解锁密码，可接受用户密码
- 14: 输入第四位解锁密码，可接受用户密码
- 15: 输入第二位解锁密码，可接受管理员密码
- 16: 输入第三位解锁密码，可接受管理员密码
- 17: 输入第四位解锁密码，可接受管理员密码
- 18: 按下按键开始设置密码，或切换到验证密码状态

启动时开始设置密码，设置密码后根据 Mode 进入管理员模式（状态 4）或验证模式（状态 8）；管理员模式下，成功验证管理员模式将进入状态 18，确认后可修改密码，也可以不进行确认直接转到验证模式；验证模式下，可输入管理员密码与用户密码进行解锁，若解锁失败累计失败次数，且回到状态 8，失败次数过多则无法再解锁，只能切换到管理员模式用管理员密码清除失败次数。

4 实验小结

本次实验给我最大的收获就是用 VHDL 来实现状态机，通过这个实验，让我更加熟悉状态机的理论，用 VHDL 实现状态机来解决具体问题。

一开始没有考虑到用管理员密码解锁的情况，到了检查的时候才进行修改，浪费了一些时间。因此在定义问题时就需要把问题考虑清楚，避免再次遇到这这样的情况。