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

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity lock is
  port(
    clock : in std_logic;
    code : in std_logic_vector(3 downto 0);
    mode : in std_logic_vector(1 downto 0);
    clk, rst : in std_logic;
    unlock, setting : out std_logic;
    alarm, error : buffer std_logic;
    lights : out std_logic_vector(3 downto 0)
  );
  type T is array (3 downto 0) of integer;
end lock;

architecture lock of lock is
  signal password : T;
  signal manager : T := (9, 6, 8, 4);
  signal state : integer := 0;
  signal error_count : integer := 0;
  signal click : std_logic;
begin
  process(clock)
  begin
    if clock’event and clock=’1’ then
      click <= clk;
    end if;
  end process;

  process(click, rst)
  begin
    if click’event and click=’1’ then
      if mode=”00” then
        case state is
          when 0 => password(0) <= CONV_INTEGER(code); state <= 1;
          when 1 => password(1) <= CONV_INTEGER(code); state <= 2;
          when 2 => password(2) <= CONV_INTEGER(code); state <= 3;
          when 3 => password(3) <= CONV_INTEGER(code); state <= 4;
          when 4 =>
            unlock <= ’0’;
            if CONV_INTEGER(code) = manager(0) then
              state <= 5;
            else
              state <= 4;
            end if;
          when 5 =>
            if CONV_INTEGER(code) = manager(1) then
              state <= 6;
            else
              state <= 4;
            end if;
          when 6 =>
            if CONV_INTEGER(code) = manager(2) then
              state <= 7;
            else
              state <= 4;
            end if;
          when 7 =>
            if CONV_INTEGER(code) = manager(3) then
              state <= 18;
              alarm <= '0';
              error_count <= 0;
            else
              state <= 4;
            end if;
          when 18 =>
            state <= 0;
            when others => state <= 4; unlock <= '0'; error <= '0';
        end case;
      elsif mode="01" then
        case state is
          when 0 => password(0) <= CONV_INTEGER(code); state <= 1;
          when 1 => password(1) <= CONV_INTEGER(code); state <= 2;
          when 2 => password(2) <= CONV_INTEGER(code); state <= 3;
          when 3 => password(3) <= CONV_INTEGER(code); state <= 8;
          when 8 =>
            unlock <= '0';
            if alarm='0' then
              if CONV_INTEGER(code) = password(0) and
                  CONV_INTEGER(code) = manager(0) then
                state <= 9;
              elsif CONV_INTEGER(code) = password(0) then
                state <= 12;
              elsif CONV_INTEGER(code) = manager(0) then
                state <= 15;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
```

```
                    alarm <= '1';
                    error_count <= 0;
                  else
                    error_count <= error_count + 1;
                  end if;
                end if;
              end if;
            when 9 =>
              if CONV_INTEGER(code) = password(1) and
                  CONV_INTEGER(code) = manager(1) then
                state <= 10;
              elsif CONV_INTEGER(code) = password(1) then
                state <= 13;
              elsif CONV_INTEGER(code) = manager(1) then
                state <= 16;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 10 =>
              if CONV_INTEGER(code) = password(2) and
                  CONV_INTEGER(code) = manager(2) then
                state <= 11;
              elsif CONV_INTEGER(code) = password(2) then
                state <= 14;
              elsif CONV_INTEGER(code) = manager(2) then
                state <= 17;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 11 =>
              if CONV_INTEGER(code) = password(3) or
                  CONV_INTEGER(code) = manager(3) then
                state <= 8;
                unlock <= '1';
                error_count <= 0;
                error <= '0';
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 12 =>
              if CONV_INTEGER(code) = password(1) then
                state <= 13;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 13 =>
              if CONV_INTEGER(code) = password(2) then
                state <= 14;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 14 =>
              if CONV_INTEGER(code) = password(3) then
                state <= 8;
                unlock <= '1';
                error_count <= 0;
                error <= '0';
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 15 =>
              if CONV_INTEGER(code) = manager(1) then
                state <= 16;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 16 =>
              if CONV_INTEGER(code) = manager(2) then
                state <= 17;
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when 17 =>
              if CONV_INTEGER(code) = manager(3) then
                state <= 8;
                unlock <= '1';
                error_count <= 0;
                error <= '0';
              else
                state <= 8;
                error <= '1';
                if error_count + 1 = 3 then
                  alarm <= '1';
                  error_count <= 0;
                else
                  error_count <= error_count + 1;
                end if;
              end if;
            when others => state <= 8; unlock <= '0'; error <= '0';
          end case;
        end if;
      end if;
    end process;

    process(state)
    begin
      if state < 4 then
        setting <= '1';
      else
        setting <= '0';
      end if;
      case state is
        when 0 =>
          lights(0) <= '1'; lights(1) <= '0';
          lights(2) <= '0'; lights(3) <= '0';
        when 1 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '0'; lights(3) <= '0';
        when 2 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '1'; lights(3) <= '0';
        when 3 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '1'; lights(3) <= '1';
        when 4 =>
          lights(0) <= '1'; lights(1) <= '0';
          lights(2) <= '0'; lights(3) <= '0';
        when 5 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '0'; lights(3) <= '0';
        when 6 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '1'; lights(3) <= '0';
        when 7 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '1'; lights(3) <= '1';
        when 8 =>
          lights(0) <= '1'; lights(1) <= '0';
          lights(2) <= '0'; lights(3) <= '0';
        when 9 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '0'; lights(3) <= '0';
        when 10 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '1'; lights(3) <= '0';
        when 11 =>
          lights(0) <= '1'; lights(1) <= '1';
          lights(2) <= '1'; lights(3) <= '1';
        when 12 =>
```

```
        lights(0) <= '1'; lights(1) <= '1';
        lights(2) <= '0'; lights(3) <= '0';
      when 13 =>
        lights(0) <= '1'; lights(1) <= '1';
        lights(2) <= '1'; lights(3) <= '0';
      when 14 =>
        lights(0) <= '1'; lights(1) <= '1';
        lights(2) <= '1'; lights(3) <= '1';
      when 15 =>
        lights(0) <= '1'; lights(1) <= '1';
        lights(2) <= '0'; lights(3) <= '0';
      when 16 =>
        lights(0) <= '1'; lights(1) <= '1';
        lights(2) <= '1'; lights(3) <= '0';
      when 17 =>
        lights(0) <= '1'; lights(1) <= '1';
        lights(2) <= '1'; lights(3) <= '1';
      when 18 =>
        lights(0) <= '0'; lights(1) <= '0';
        lights(2) <= '0'; lights(3) <= '0';
      when others => NULL;
    end case;
  end process;
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。