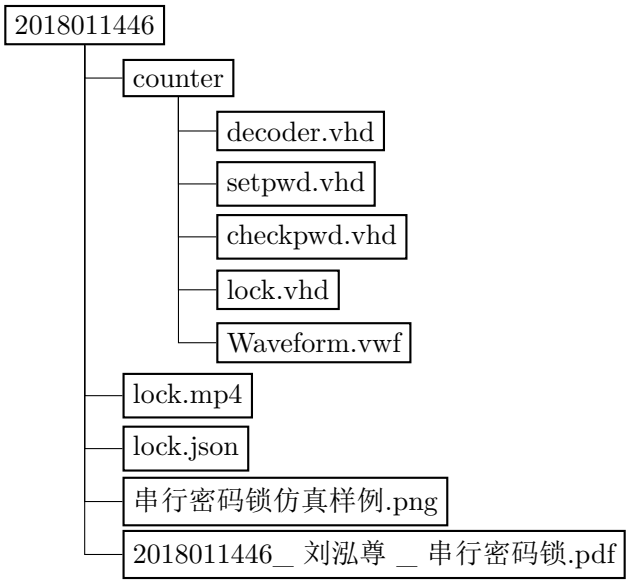
串行密码锁：实验报告

刘泓尊　2018011446　计 84

2020 年 4 月 1 日

目录

1 File Structure



2 实验目的

(1) 学习使用状态机来控制电路工作，在不同的状态下完成相应的功能。

(2) 进一步掌握时序逻辑电路的基本分析和设计方法。

(3) 学会利用软件仿真实现对数字电路的逻辑功能进行验证和分析.

3 实验任务

(1) 设计一个 4 位 16 进制串行电子密码锁，功能包括: 设置密码，验证密码。

(2) 实现密码预置 (管理员密码) 和系统报警功能。

4 代码及注释

在 mode=“00”时是设置密码状态，mode=“01”时是验证密码状态。mode=“1x”时仍是验证密码，但是验证管理员密码。管理员密码我在程序中设置为“0000”。每次更换操作后均应按下“rst”，每输入一位之后均应按下“clk”.

我使用状态机实现了串行密码锁的“设置密码”与“验证密码”。其流程与实验说明中的流程一致。我将程序模块分为“输入密码 (setpwd)”与“验证密码 (checkpwd)”两个模块，每个模块内的核心逻辑就是状态机。同时，为了便于直观显示，我将输入的每个数字通过“显示译码器”输出到 7 段数码管。

4.1 设置密码

设置密码的状态机与课本给出的一致。

setpwd.vhd

```
entity setpwd is
    port(
        clk, rst: in std_logic;
        mode: in std_logic_vector(1 downto 0);
        code: in std_logic_vector(3 downto 0);
        pwd0, pwd1, pwd2, pwd3: out std_logic_vector(3 downto 0);
        currentnum: out std_logic_vector(3 downto 0)
    );
end setpwd;

architecture bhv_set of setpwd is
    signal state: integer := 0;
begin
    currentnum <= code;
    process(clk, rst) begin
        if rst = '1' then --异步复位
            state <= 1;
```

```
        elsif clk'event and clk = '1' then
            if mode = "00" then
                case state is--设置密码状态机
                when 1 =>
                    pwd0 <= code; state <= 2;
                when 2 =>
                    pwd1 <= code; state <= 3;
                when 3 =>
                    pwd2 <= code; state <= 4;
                when 4 =>
                    pwd3 <= code; state <= 0;
                when others => null;
                end case;
            end if;
        end if;
    end process;
end bhv_set;
```

4.2 验证密码

验证密码的流程与课本给出的一致，只是加了 mode=”01” 还是 mode=”1x” 的判断。

checkpwd.vhd

```
entity checkpwd is
    port(
        clk, rst: in std_logic;
            mode: in std_logic_vector(1 downto 0);
            code: in std_logic_vector(3 downto 0);
            unlock, err, alarm: buffer std_logic;
            pwd0, pwd1, pwd2, pwd3: in std_logic_vector(3 downto 0);--4位16进
                制密码，由setpwd提供
            currentnum: out std_logic_vector(3 downto 0) -- 数码管输出当前数位
    );
    type fourbitpwd is array(3 downto 0) of integer;
end checkpwd;

architecture bhv_check of checkpwd is
    signal state: integer := 0; --状态机当前状态
    signal cnt: integer := 0;--输入错误的次数
    constant adminpwd: fourbitpwd := (0, 0, 0, 0);--admin密码设为0000
begin
    currentnum <= code;
    process(clk, rst) begin
        if rst = '1' then --reset时alarm不清零
            state <= 1;--状态机:i状态开始接受第i位密码
```

```
            unlock <= '0'; err <= '0';
            if alarm = '1' then
                cnt <= 0;
            end if;
        elsif clk'event and clk='1' then
            if mode = "01" then --user验证密码模式
                case state is--状态机
                when 1 =>
                    if code = pwd0 then
                        state <= 2;
                    else
                        err <= '1'; cnt <= cnt + 1; state <= 0;--输入错误
                    end if;
                when 2 =>
                    if code = pwd1 then
                        state <= 3;
                    else
                        err <= '1'; cnt <= cnt + 1; state <= 0;
                    end if;
                when 3 =>
                    if code = pwd2 then
                        state <= 4;
                    else
                        err <= '1'; cnt <= cnt + 1; state <= 0;
                    end if;
                when 4 =>
                    if code = pwd3 then --正确
                        cnt <= 0; err <= '0'; unlock <= '1'; state <= 0;
                    else
                        err <= '1'; cnt <= cnt + 1; state <= 0;
                    end if;
                when others => null;
                end case;
                if cnt > 1 then
                    alarm <= '1';
                end if;
            elsif (mode = "10" or mode = "11") then --admin模式
                case state is
                when 1 =>
                    if CONV_INTEGER(code) = adminpwd(3) then
                        state <= 2;
                    else
                        err <= '1'; state <= 0;--输入错误
                    end if;
```

```
                when 2 =>
                    if CONV_INTEGER(code) = adminpwd(2) then
                        state <= 3;
                    else
                        err <= '1'; state <= 0;
                    end if;
                when 3 =>
                    if CONV_INTEGER(code) = adminpwd(1) then
                        state <= 4;
                    else
                        err <= '1'; state <= 0;
                    end if;
                when 4 =>
                    if CONV_INTEGER(code) = adminpwd(0) then
                        cnt <= 0; err <= '0'; unlock <= '1';
                        alarm <= '0';--警报关闭
                        state <= 0; --正确
                    else
                        err <= '1'; state <= 0;
                    end if;
                when others => null;
                end case;
                if cnt > 1 then
                    alarm <= '1';
                end if;
            end if;
        end if;
    end process;
end bhv_check;
```

4.3 串行密码锁

串行密码锁的实现即例化了上述的“设置密码”与“验证密码”模块，同时对输入数字做了译码处理，用于七段数码管输出。

lock.vhd

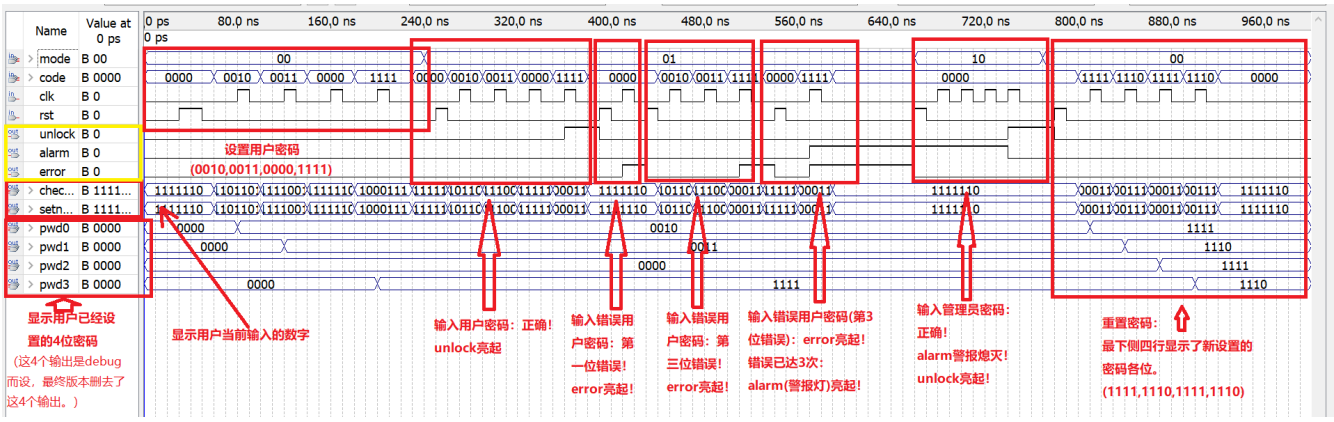
```
-- 串行密码锁:支持设置密码、验证密码、三次输入限制、管理员模式 --
entity lock is
    port(
        code: in std_logic_vector(3 downto 0); --only for one digit
        mode: in std_logic_vector(1 downto 0); --mode: 1x:Admin, 00:set
            password, 01:check password
        clk, rst: in std_logic;
        unlock_out, err_out, alarm_out: out std_logic;
```

```
8           curnum: out std_logic_vector(6 downto 0)--Show the input number, for
                debug;
9       );
10  end lock;
11
12  architecture bhv of lock is
13      component checkpwd
14          port(
15              clk, rst: in std_logic;
16              mode: in std_logic_vector(1 downto 0);
17              code: in std_logic_vector(3 downto 0);
18              unlock, err, alarm: buffer std_logic;
19              pwd0, pwd1, pwd2, pwd3: in std_logic_vector(3 downto 0);--4位16进
                    制密码，由setpwd提供
20              currentnum: out std_logic_vector(3 downto 0) -- 数码管输出当前数位
21          );
22      end component;
23      component setpwd
24          port(
25              clk, rst: in std_logic;
26              mode: in std_logic_vector(1 downto 0);
27              code: in std_logic_vector(3 downto 0);
28              pwd0, pwd1, pwd2, pwd3: out std_logic_vector(3 downto 0);
29              currentnum: out std_logic_vector(3 downto 0)
30          );
31      end component;
32      component decoder
33          port(
34              bit_4_vec: in std_logic_vector(3 downto 0);
35              bit_7_vec: out std_logic_vector(6 downto 0)
36          );
37      end component;
38      signal pwd0, pwd1, pwd2, pwd3: std_logic_vector(3 downto 0);
39      signal setnum: std_logic_vector(3 downto 0);
40      signal checknum: std_logic_vector(3 downto 0);
41      signal tempnum: std_logic_vector(3 downto 0);
42      signal alarm, err, unlock: std_logic;
43  begin
44      cpwd: checkpwd port map(clk=>clk, rst=>rst, mode=>mode, code=>code, unlock
            =>unlock, err=>err, alarm=>alarm, pwd0=>pwd0, pwd1=>pwd1, pwd2=>pwd2,
            pwd3=>pwd3, currentnum=>checknum);--验证密码元件例化
45      spwd: setpwd port map(clk=>clk, rst=>rst, mode=>mode, code=>code, pwd0=>
            pwd0, pwd1=>pwd1, pwd2=>pwd2, pwd3=>pwd3, currentnum=>setnum);--设置密
            码元件例化
```

```
46      tempnum <= setnum when mode = "00" else checknum;--显示的数字
47      alarm_out <= alarm; err_out <= err; unlock_out <= unlock;
48      de: decoder port map(bit_4_vec=>tempnum, bit_7_vec=>curnum);--显示译码
49  end bhv;
```

5 仿真结果

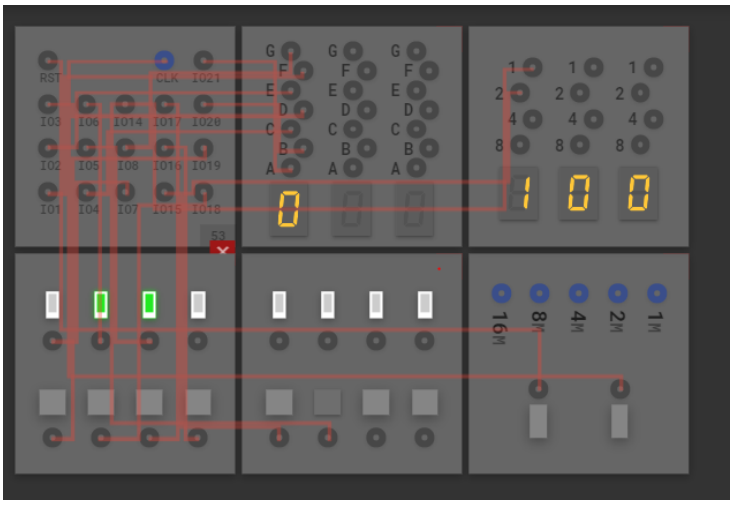
使用 Quartus 的 ModelSim 进行仿真 (附"**./lock/Waveform.vwf**" 文件). 下图详细说明了电路功能，包括“设置密码”，“验证密码”，“正确亮起 unlock”，“错误亮起 err”，“输入三次错误密码后亮起 alarm”，“管理员模式下输入 0000 解除 alarm”，“重置密码”等功能。



6 JieLabs 运行结果 (附录屏)

我在 JieLabs 上进行了硬件验证，并将实验结果录屏保存在./**lock.mp4** 下，也随附平台导出的 **lock.json** 文件。

该视频流程为: “设置密码为 137F”→“输入正确密码 137F，unlock 亮起”→“输入错误密码，err 亮起”→“输入 3 次错误密码，alarm 亮起”→“输入管理员密码 0000 解锁，消除 alarm”。下面是实验过程截图.



说明: 左下角从左至右 3 个 LED 分别为“unlock, err, alarm”. 左下角 4 个开关代表输入的 1 位 16 进制数，该数字也经过译码显示在 7 段数码管上。下层中部模块最左边两个开关为 mode 的控制，左侧高位右侧低位，为了直观，我将其显示在右上角最左侧数码管处。右下角模块左侧为 rst，右侧为 clk。

7 实验总结

本次实验是我第一次遇到较为实用性的功能电路，我学习了状态机在时序逻辑电路中的应用，学到了 buffer, case, integer 等语句的使用，以及 CONV_INTEGER 函数 (用于将 std_logic_vector 转换为 integer)。综合练习了所学知识，设计代码更加得心应手，收获良多。最后，感谢老师和助教在微信群的耐心答疑与帮助!