

# 四位密码锁.实验报告

2018011324

计82

尤艺霖

## 1.实验要求&目的

### 1.1.实验目的

学习使用状态机来控制电路工作，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

学会利用软件仿真实现对数字逻辑电路的逻辑功能进行验证和分析

### 1.2.实验要求

设计一个四位十六进制密码锁，需求具有设置密码和验证密码的功能，并能在错误/正确时点亮对应的灯。

## 2.思路

使用9个状态来完成这个实验，分别设为p0-p8。

提供三个输入：时钟信号clk，重置信号rst，模式选择mode；状态只有在时钟上升沿才会发生改变

p7、p8分别为成功验证/设置的状态和发生错误的状态，需要重置才能回到开始状态。

p0作为开始状态，模式为设置密码时读入第一位，然后走向p1；模式为验证密码时验证第一位，错误则走向p8，否则走向p4。

p1、p2、p3为设置密码用的状态，分别输入对应的位数，如果在这一过程中改变了模式，则走向p8，否则走向下一个，其中p3成功时走向p7。

p4、p5、p6为验证密码的状态，在过程中改变模式或者验证出错则走向p8，否则走向下一个，其中p6成功时走向p7。

## 3.代码

### 3.1.locker.vhd

输入和输出采用教材上的要求，所以mode是两位的。

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

entity locker is
port(
    clk,rst:in std_logic;
    mode:in std_logic_vector(1 downto 0); --00设置密码, 01输入密码
    code:in std_logic_vector(3 downto 0);
```

```

        unlock,err:out std_logic
    );
end locker;

architecture lock of locker is
    type state is(p0,p1,p2,p3,p4,p5,p6,p7,p8);
    signal cur:state;
    signal pwd0,pwd1,pwd2,pwd3: std_logic_vector(3 downto 0);
begin
    process(clk,rst)
    begin
        if(rst='1') then --重置
            cur<=p0;
            unlock<='0';
            err<='0';
        elsif(rising_edge(clk)) then --状态机部分
            case cur is
                when p0=>
                    if(mode="00") then
                        pwd0<=code;
                        cur<=p1;
                    elsif(mode="01") then
                        if(pwd0=code) then
                            cur<=p4;
                        else
                            cur<=p8;
                            err<='1';
                        end if;
                    end if;
                when p1=> --p1,p2,p3是设置密码部分
                    if(mode="00") then
                        pwd1<=code;
                        cur<=p2;
                    elsif(mode="01") then
                        cur<=p8;
                        err<='1';
                    end if;
                when p2=>
                    if(mode="00") then
                        pwd2<=code;
                        cur<=p3;
                    elsif(mode="01") then
                        cur<=p8;
                        err<='1';
                    end if;
                when p3=>
                    if(mode="00") then
                        pwd3<=code;
                        cur<=p7;
                        unlock<='1';
                    elsif(mode="01") then
                        cur<=p8;
                        err<='1';
                    end if;
            end case;
        end if;
    end process;
end lock;

```

```

        end if;
when p4=> --p4,p5,p6是输入密码部分
    if(mode="00") then
        cur<=p8;
        err<='1';
    elsif(mode="01") then
        if(pwd1=code) then
            cur<=p5;
        else
            cur<=p8;
            err<='1';
        end if;
    end if;
when p5=>
    if(mode="00") then
        cur<=p8;
        err<='1';
    elsif(mode="01") then
        if(pwd2=code) then
            cur<=p6;
        else
            cur<=p8;
            err<='1';
        end if;
    end if;
when p6=>
    if(mode="00") then
        cur<=p8;
        err<='1';
    elsif(mode="01") then
        if(pwd3=code) then
            cur<=p7;
            unlock<='1';
        else
            cur<=p8;
            err<='1';
        end if;
    end if;
    when others=>NULL;
end case;
end if;
end process;
end lock;

```

### 3.2.locker\_tb.vhd

由教材中网站生成的TestBench模板修改而来，包括了重置、设置密码、正确和两次错误、再次设置密码等测试项目，具体图像由实验结果给出。

```

-- Testbench created online at:
--   www.doulos.com/knowhow/perl/testbench_creation/
-- Copyright Doulos Ltd
-- SD, 03 November 2002

```

```

library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity locker_tb is
end;

architecture bench of locker_tb is

    component locker
    port(
        clk,rst:in std_logic;
        mode:in std_logic_vector(1 downto 0);
        code:in std_logic_vector(3 downto 0);
        unlock,err:out std_logic
    );
    end component;

    signal clk,rst: std_logic;
    signal mode: std_logic_vector(1 downto 0);
    signal code: std_logic_vector(3 downto 0);
    signal unlock,err: std_logic ;

    constant clock_period: time := 1000 ns;
    signal stop_the_clock: boolean;

begin

    uut: locker port map ( clk    => clk,
                           rst     => rst,
                           mode    => mode,
                           code     => code,
                           unlock  => unlock,
                           err      => err );

    stimulus: process
    begin

        -- Put initialisation code here
        rst<='0';
        mode<="00";
        code<="0000";
        wait for 400ns;
        rst<='1';
        wait for 1000ns;
        rst<='0';
        code<="1000";
        wait for 1000ns;
        code<="0100";
        wait for 1000ns;
        code<="0010";
        wait for 1000ns;
    end process;
end architecture bench;

```

```
code<="0001";
wait for 1000ns;

rst<='1';
wait for 1000ns;
rst<='0';
code<="1000";
mode<="01";
wait for 1000ns;
code<="0100";
wait for 1000ns;
code<="0010";
wait for 1000ns;
code<="0001";
wait for 1000ns;

rst<='1';
wait for 1000ns;
rst<='0';
code<="1000";
mode<="01";
wait for 1000ns;
code<="0100";
wait for 1000ns;
code<="0010";
wait for 1000ns;
code<="1001";
wait for 1000ns;

rst<='1';
wait for 1000ns;
rst<='0';
code<="1000";
mode<="01";
wait for 1000ns;
code<="0101";
wait for 1000ns;

rst<='1';
wait for 1000ns;
rst<='0';
code<="0000";
mode<="00";
wait for 1000ns;
code<="0000";
wait for 1000ns;
code<="0000";
wait for 1000ns;
code<="0000";
wait for 1000ns;

rst<='1';
wait for 1000ns;
```

```

rst<='0';
code<="0000";
mode<="01";
wait for 1000ns;
code<="0000";
wait for 1000ns;
code<="0000";
wait for 1000ns;
code<="0000";
wait for 1000ns;

-- Put test bench stimulus code here

stop_the_clock <= true;
wait;
end process;

clocking: process
begin
    while not stop_the_clock loop
        clk <= '0', '1' after clock_period / 2;
        wait for clock_period;
    end loop;
    wait;
end process;

end;

```

## 4.实验结果

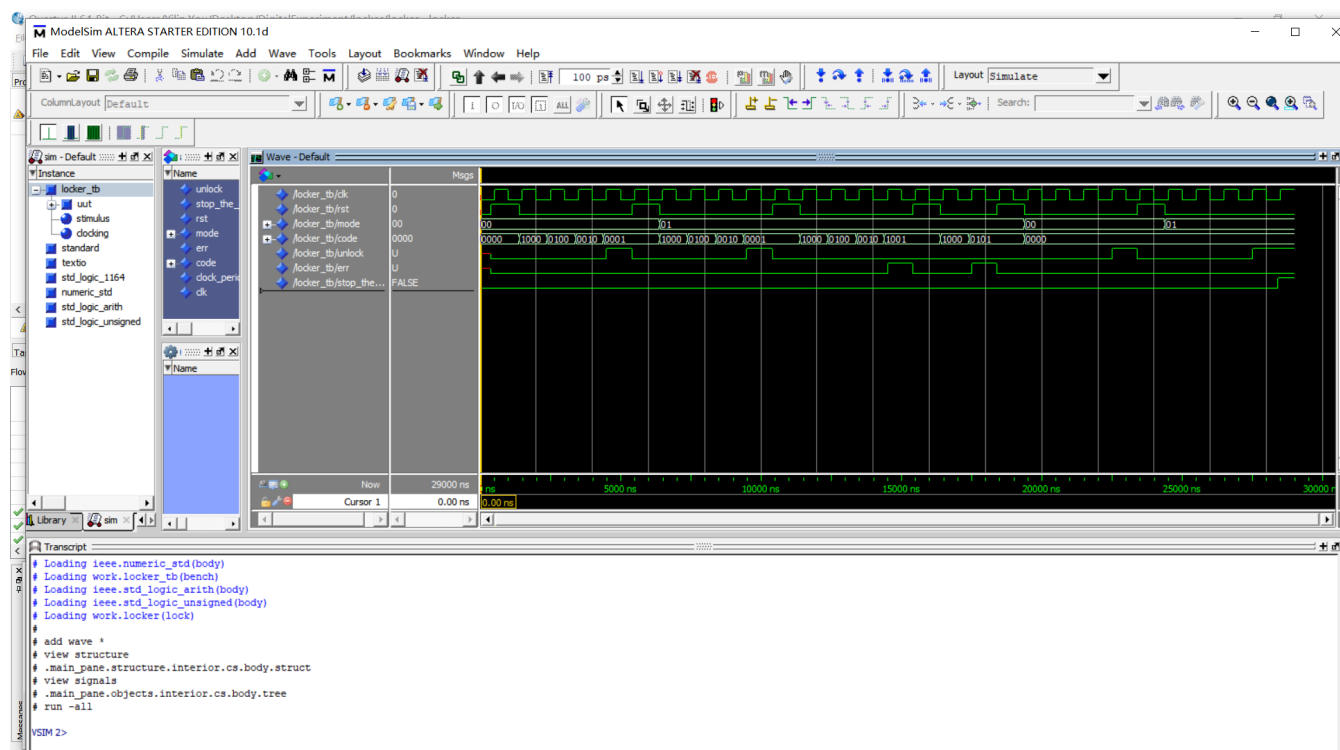
### 4.1.状态机

通过Quartus自带的Status Machine Viewer可以获取对应的状态机：



### 4.2.本地仿真结果

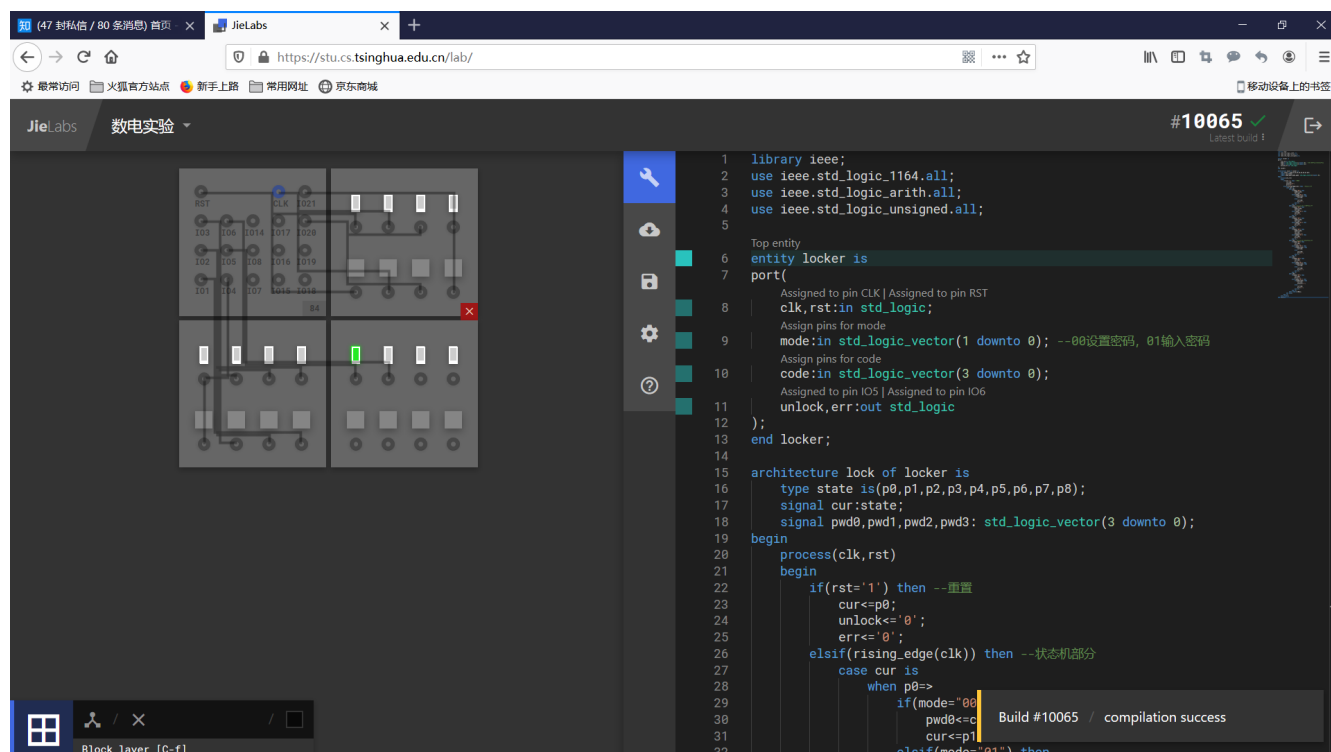
使用ModelSim-Altera进行仿真，TestBench代码已经放在上面了。测试中包含了重置、设置密码、成功验证、两次错误验证（位置不一样）、重新设置密码、重新验证正确这些测试项目，并能正确工作。



### 4.3.Jielab实验结果

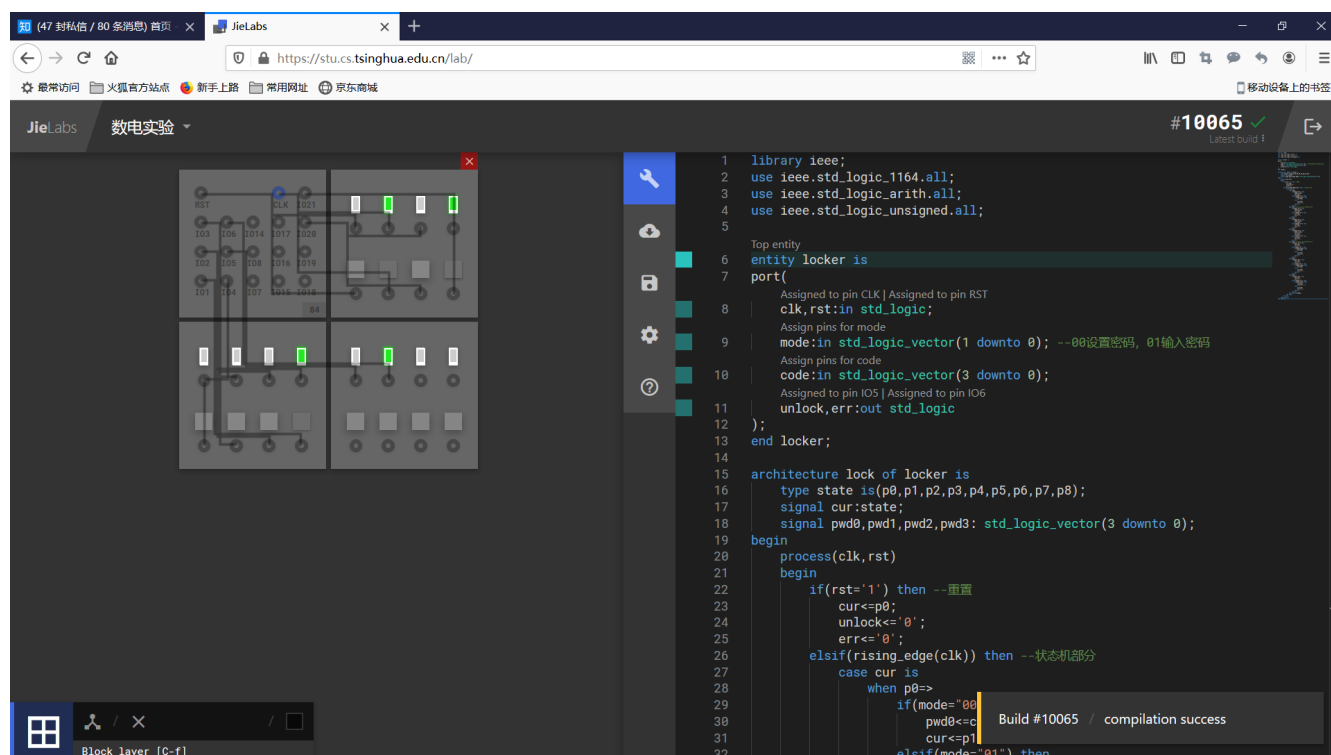
接线方式如图所示。

三张图分别展现了设置密码为0000，验证错误和验证正确三种情况，其他情况经测试也是正确的。



The screenshot shows the JieLab web interface for a digital logic experiment. On the left, a circuit diagram is displayed with various components and connections. On the right, the Verilog code for the 'locker' entity is shown. The code defines the entity's ports and internal logic, including a state machine for the locker's operation. The code is as follows:

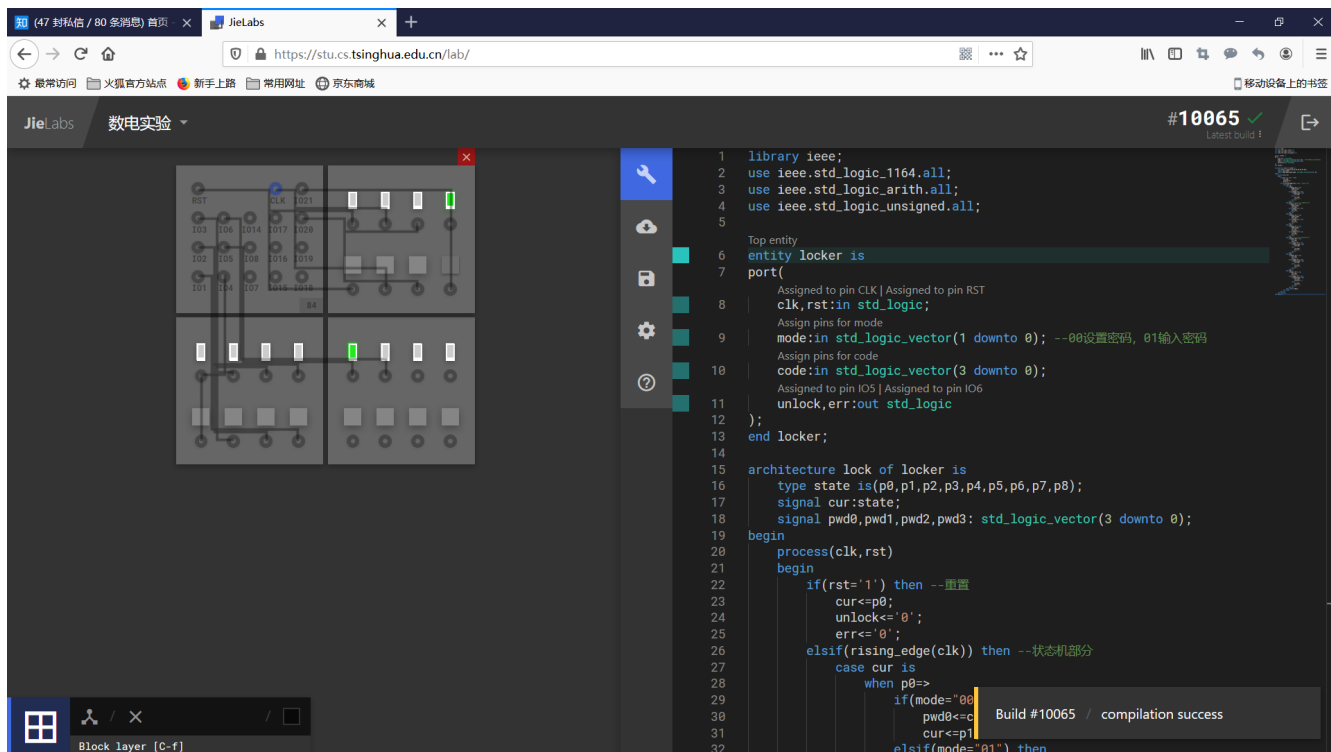
```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity locker is
7 port(
8     Assigned to pin CLK | Assigned to pin RST
9     clk,rst:in std_logic;
10    Assign pins for mode
11    mode:in std_logic_vector(1 downto 0); --00设置密码, 01输入密码
12    Assign pins for code
13    code:in std_logic_vector(3 downto 0);
14    Assigned to pin IO5 | Assigned to pin IO6
15    unlock,err:out std_logic
16 );
17 end locker;
18
19 architecture lock of locker is
20 type state is(p0,p1,p2,p3,p4,p5,p6,p7,p8);
21 signal cur:state;
22 signal pwd0,pwd1,pwd2,pwd3: std_logic_vector(3 downto 0);
23 begin
24 process(clk,rst)
25 begin
26     if(rst='1') then --重置
27         cur<=p0;
28         unlock<='0';
29         err<='0';
30     elsif(rising_edge(clk)) then --状态机部分
31         case cur is
32             when p0=>
33                 if(mode="00"
34                     if(pwd0<=c
35                         cur<=p1
36                     elsif(mode="01") then
```



This screenshot is identical to the one above, showing the JieLab web interface with the circuit diagram and Verilog code for the 'locker' entity. The code is as follows:

```
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 locker is
7 port(
8     Assigned to pin CLK | Assigned to pin RST
9     clk,rst:in std_logic;
10    Assign pins for mode
11    mode:in std_logic_vector(1 downto 0); --00设置密码, 01输入密码
12    Assign pins for code
13    code:in std_logic_vector(3 downto 0);
14    Assigned to pin IO5 | Assigned to pin IO6
15    unlock,err:out std_logic
16 );
17 end locker;
18
19 architecture lock of locker is
20 type state is(p0,p1,p2,p3,p4,p5,p6,p7,p8);
21 signal cur:state;
22 signal pwd0,pwd1,pwd2,pwd3: std_logic_vector(3 downto 0);
23 begin
24 process(clk,rst)
25 begin
26     if(rst='1') then --重置
27         cur<=p0;
28         unlock<='0';
29         err<='0';
30     elsif(rising_edge(clk)) then --状态机部分
31         case cur is
32             when p0=>
33                 if(mode="00"
34                     if(pwd0<=c
35                         cur<=p1
36                     elsif(mode="01") then
```





## 5.总结

这次实验之前的那节讲状态机的课我有事没能来听，相关的知识都是从网络中学习得到。在这次试验中，我成功使用状态机的方法完成了一个四位十六进制密码锁，支持设置密码和验证密码，并能在正确/错误是提供对应信号。

通过这次实验，我学会了利用状态机进行设计，增加了对VHDL和仿真工具的熟练度。但代码仍然显得笨重，有很多类似的代码段，这还有待后续的改进。