西安电子科技大学

电子线路实验 课程实验报告

实验名称 大规模集成数字电路设计

机电工程 学院 2004031 班

姓名 李灏轩 学号 20049200166

同作者 徐驰翔

实验日期 2021 年 11 月 21 日

成 绩

指导教师评语： 指导教师： 年 月 日
实验报告内容基本要求及参考格式 一、实验目的 二、实验所用仪器（或实验环境） 三、实验基本原理及步骤（或方案设计及理论计算） 四、实验数据记录（或仿真及软件设计） 五、实验结果分析及回答问题（或测试环境及测试结果）

自评成绩：

徐驰翔：90

李灏轩：100

钱俊鑫：80

杨文华：70

蒋君睿：70

闫鹏举：70

一、实验目的

1、熟悉PLD设计流程的基本步骤。

为达到以上目的，学生应具备基本的自主学习能力和对实验结果数据进行处理和分析的能力。

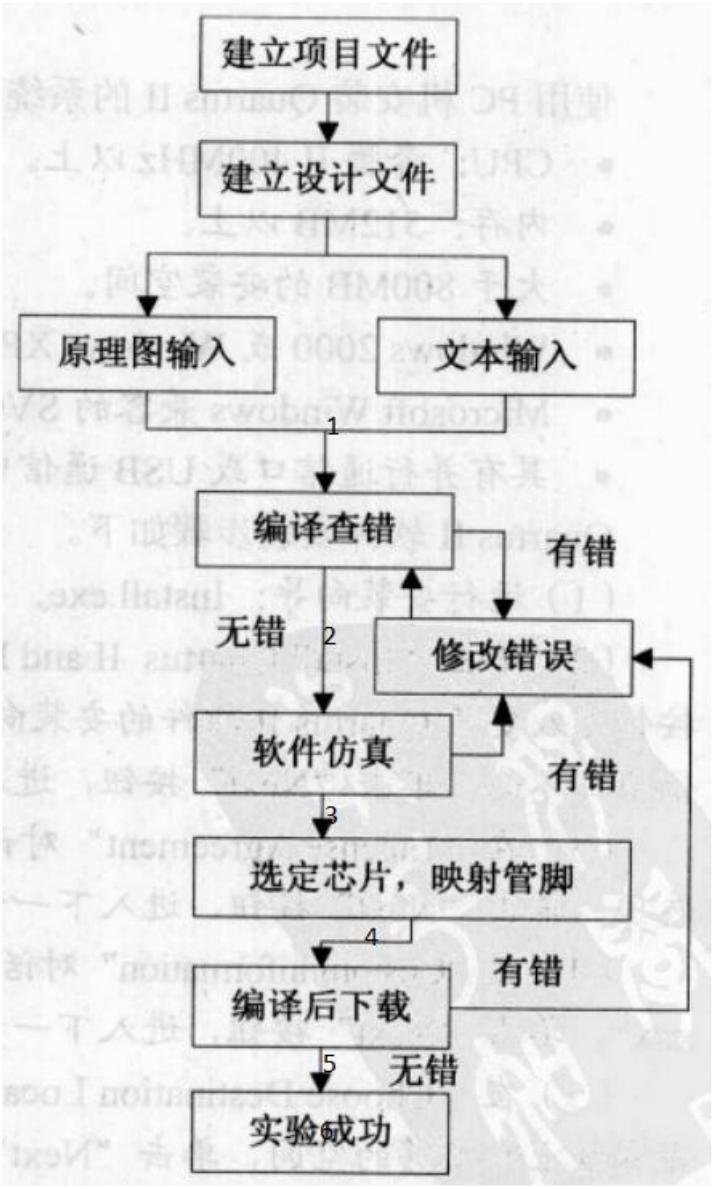
二、实验原理

第一周:崭新的实验课程，需要崭新的电路设计。请设计一个你认为最崭新的数字电路(必须与实验五、六和其他小组不同，使用芯片EPM7128SLC84-15!)，提交设计报告，并选派一个实验组携纸质版参加筛选试做。扣除试做满分后的设计分根据第二周选做结果评定，选做一组扣1分直至0分，筛选未过组取筛选通过设计的最低分。具体要求如下:以小组为单位，标准设计组成员必须是5~6人，自由组合，每人只能参加一个组，最多接受一个非标准组。

每个组交一个pdf 格式的设计报告，用满分组员名+简短的设计名命名，如陈某某组时钟.pdf，其它要求同实验报告。此外，只要认为有用的内容，都可写入设计报告，例如，崭新点，根据特定要求对需求、研制成本和性价比的分析等。抄袭零分。

为鼓励竞争，设计报告中必须包含自评成绩:最少一个100分，n人总分等于80n。时间:一周。之后，课代表将公布设计报告（要筛除的话，我会联系课代表)，未看到的同学请与课代表联系。注意时间节点。

第二周:在公布的全部设计中选做一个，根据实验结果评定实验分。



设计流程（分六步记分）

三、 实验仪器

1、数字逻辑电路实验箱+CPLD 开发板　　1块

四、实验内容及步骤

1、数字密码锁设计

实验代码如下：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY dl_top IS
PORT(
--时钟输人
clk    : IN    STD_LOGIC;
```

```
--安锁
lock  : IN     STD_LOGIC;
--输人密码
start  :IN     STD_LOGIC;
--报警复位
off_al  : IN    STD_LOGIC;
--修改密码
ps_ch  :IN      STD_LOGIC;
--密码确认
enter  : IN      STD_LOGIC;
-- 0~9按键输入
key_in : IN      STD_LOGIC_VECTOR(9 DOWNTO 0);
--钥匙信号
key    :OUT   STD_LOGIC;
--报警信号
warn    :OUT   STD_LOGIC
);
END dl_top;
ARCHITECTURE structural of dl_top IS
--元件说明
COMPONENT dl_control
PORT (
--时钟输入
clk     :IN    STD_LOGIC;
--安锁
lock :IN       STD_LOGIC;
--输人密码
start : IN      STD_LOGIC;
--报警复位
off_al  : IN     STD_LOGIC;
--修改密码
ps_ch  :IN      STD_LOGIC;
--密码确认
enter  : IN          STD_LOGIC;
--密码脉冲
ps_i  : IN           STD_LOGIC;
--比较结果
cmp_r  : IN         STD_LOGIC;
--完成密码输人指示
cin  :IN              STD_LOGIC;
--密码输人使能
code_en : OUT      STD_LOGIC;
--计数器清零
cnt_clr  : OUT     STD_LOGIC;
```

```
--计数器时钟
cnt_clk  : OUT     STD_LOGIC;
--寄存器读/写信号
reg_wr  : OUT    STD_LOGIC;
--钥匙信号
key  : OUT    STD_LOGIC;
--报警信号
warn  : OUT    STD_LOGIC
);
END COMPONENT;

COMPONENT  dl_counter

PORT(
             --清零
              clr          :IN      STD_LOGIC;
              --时钟
              clk         :IN      STD_LOGIC;
              --计数满指示
              cout       :OUT    STD_LOGIC;
              --计数值输出
                addr       :OUT    STD_LOGIC_VECTOR(1 DOWNTO 0)
             );
END COMPONENT;

COMPONENT dl_reg
   PORT(
            --读写时钟
            clk         :IN      STD_LOGIC;
             --读写信号
             reg_wr      :IN      STD_LOGIC;
             --使能
            en         :IN      STD_LOGIC;
            --地址
             addr          :IN      STD_LOGIC_VECTOR(1 DOWNTO 0);
             --数据输入
             data_in      :IN      STD_LOGIC_VECTOR(3 DOWNTO 0);
             --数据输出
             data_out   :OUT       STD_LOGIC_VECTOR(3 DOWNTO 0)
            );
 END COMPONENT;
COMPONENT dl_cmp
   PORT(
             a        :IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
```

```
            b          :IN    STD_LOGIC_VECTOR(3 DOWNTO 0);
            c              :OUT     STD_LOGIC
             );
 END COMPONENT;
COMPONENT dl_coder
  PORT(
         --时钟输入
         clk    :IN    STD_LOGIC;
          --使能
          en    :IN    STD_LOGIC;
          --按键输入
           key_in     :IN    STD_LOGIC_VECTOR(9 DOWNTO 0);
          --密码脉冲
           ps_i        :OUT   STD_LOGIC;
          --编码输出
          code_out       :OUT   STD_LOGIC_VECTOR(3 DOWNTO 0)
           );
 END COMPONENT;
--信号定义
SIGNAL    code_en         :STD_LOGIC;
SIGNAL     cnt_clr         :STD_LOGIC;
SIGNAL     cnt_clk         :STD_LOGIC;
SIGNAL     reg_wr          :STD_LOGIC;
SIGNAL     cin          :STD_LOGIC;
SIGNAL      ps_i          :STD_LOGIC;
SIGNAL      cmp_r          :STD_LOGIC;
SIGNAL      addr           :STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL       data_out        :STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL       code_out         :STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN

--元件例化
CONTROL: dl_control
   PORT MAP(
              --时钟输人

                clk       => clk,
               --安锁

                 lock     => lock,
               --输人密码

                   start      => start,
               --报警复位
```

```
                off_al      => off_al,
                --修改密码

                  ps_ch       => ps_ch,
                --密码确认
                enter        => enter,
                 --密码脉冲

               ps_i           => ps_i,
                --比较结果


                cmp_r        =>cmp_r,
                 --完 成密码输人指示
                 cin            =>cin,
                --密码输入使能
                 code_en            => code_en,
                --计数器清零
                 cnt_clr             => cnt_clr,
               --计数器时钟
                 cnt_clk            => cnt_clk,
                 --寄存器读/写信号
                reg_wr              =>reg_wr,
                -- 钥匙信号

                   key                    => key,
               --报警信号
                 warn          =>  warn
            );
COUNTER: dl_counter
PORT MAP(
                  --清零
                    clr         =>cnt_clr,

                   --时钟
                    clk       => cnt_clk,

                   --计数满指示
                   cout              => cin,
                  --计数值输出
                  addr                => addr
                 );
REG: dl_reg
 PORT MAP(
```

```
                --读/写时钟
                clk                 => clk,
                -- 读/写信号
               reg_wr           => reg_wr,
               --使能

                en                   =>cnt_clk,

          --地址
               addr               => addr,
                  --数据输人
                data_in            => code_out,
                  --数据输出

                 data_out          => data_out

               );
COMPARATOR: dl_cmp
PORT MAP(

                  a             => code_out,
                  b                => data_out,
                   c               => cmp_r
                );
 CODER: dl_coder
PORT MAP(
                    -- 时钟输入
                    clk        => clk,
                    --使能
                       en         => code_en,
                    --按键输入
                      key_in          => key_in,
                     -- 密码脉冲
                        ps_i             => ps_i,
                      -- 编码输出
                     code_out           => code_out
                  );
END structural;

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY dl_control IS
```

```
PORT(
          --时钟输人
          clk    : IN    STD_LOGIC;
           --安锁
           lock  : IN     STD_LOGIC;
           --输人密码
           start  :IN     STD_LOGIC;
          --报警复位
           off_al  : IN    STD_LOGIC;
           --修改密码
            ps_ch  :IN      STD_LOGIC;
             --密码确认
             enter  : IN      STD_LOGIC;
           --密码脉冲
            ps_i       :IN      STD_LOGIC;
           --比较结果
           cmp_r  : IN         STD_LOGIC;
--完成密码输人指示
cin  :IN              STD_LOGIC;
--密码输人使能
code_en : OUT      STD_LOGIC;
--计数器清零
cnt_clr  : OUT     STD_LOGIC;
--计数器时钟
cnt_clk  : OUT     STD_LOGIC;
--寄存器读/写信号
reg_wr  : OUT    STD_LOGIC;
--钥匙信号
key  : OUT    STD_LOGIC;
--报警信号
warn  : OUT    STD_LOGIC
);
END dl_control;
ARCHITECTURE behave of dl_control IS
--常量定义
CONSTANT  KEY_ACTIVE             : STD_LOGIC := '1';
--状态机定义
--共七个状态，分别是：
--错误、报警及修改密码

type state_type IS (OUTLOCK, INLOCK, PS_INPUT,PS_RIGHT,PS_WRONG,ALARM,PS_CHANGE);

SIGNAL state    :state_type;
BEGIN
```

```
  --密码脉冲作为计数时钟
 cnt_clk <= ps_i;

-- 状态机
PROCESS(clk)
BEGIN

        IF rising_edge(clk)  THEN
            CASE state IS
             --
             WHEN OUTLOCK =>
                          --

                   key<= '0';
                         IF lock = KEY_ACTIVE THEN

                           state <= INLOCK;
                         ELSIF ps_ch = KEY_ACTIVE THEN

                           state <=  PS_CHANGE;
                         ElSE

                           state <= OUTLOCK;
                         END IF;
                    WHEN INLOCK  =>
                           key          <='1';
                           code_en   <='0';
                           cnt_clr      <='1';
                           reg_wr     <='0';
                           warn          <='0';
                     IF start = KEY_ACTIVE THEN
                          state <= PS_INPUT;
                     ELSE
                           state <= INLOCK;
                     END IF;
                      --输人密码状态
                    WHEN PS_INPUT =>
                        --允许输人密码
                        code_en <='1';
                        cnt_clr <='0';
                       --寄存器读使能
                         reg_wr    <='0';
                       --密码位数和内容均正确
                       IF cin='1'  AND ps_i='1'  AND cmp_r='1'  THEN
```

```
                code_en<='0';
                cnt_clr <= '1';
                 state <= PS_RIGHT;
                --密码出现错误
            ELSIF  ps_i='1'  AND cmp_r='0' THEN
                 code_en<='0';
                 cnt_clr<= '1';
                 state<= PS_WRONG;
                    --在密码位数不足时,按Enter
             ELSIF enter = KEY_ACTIVE AND cin ='0' THEN
                  code_en<='0';
                 cnt_clr<= '1';
                 state<= ALARM;
             ELSE
                   state <= PS_INPUT;
             END IF;
              WHEN PS_RIGHT =>

                       IF enter = KEY_ACTIVE THEN
                              state <= OUTLOCK;
                        ELSE
                              state <= PS_RIGHT;
                        END IF;

                 WHEN PS_WRONG =>
                       IF enter = KEY_ACTIVE THEN
                                              state  <= ALARM;
ELSE
     state <= PS_WRONG;
    END IF;

 WHEN ALARM =>

    IF off_al = KEY_ACTIVE THEN

     warn <= '0';

   state <= INLOCK;
     ELSE

              warn<='1';
           state <= ALARM;
     END IF;
```

```
WHEN PS_CHANGE =>
                                code_en <='1';
                                cnt_clr <= '0' ;
                                reg_wr <= '1';
                                IF cin = '1'  THEN

code_en <= '0';

                                          cnt_clr   <='1';
                                          state  <=  OUTLOCK ;
                                 END IF;
                     WHEN OTHERS =>
                              state <= INLOCK;
                    END CASE;
                END IF ;

            END PROCESS;

END behave;



LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY dl_counter IS
PORT(

clr
            :IN               STD_LOGIC;
           clk
            :IN               STD_LOGIC;
           cout          :OUT
            STD_LOGIC;
           addr           :OUT
            STD_LOGIC_VECTOR(1 DOWNTO 0)
           )
;

END dl_counter;
ARCHITECTURE behave of dl_counter IS
CONSTANT RESET_ACTIVE  : STD_LOGIC  :='1' ;
SIGNAL cnt  : STD_LOGIC_VECTOR(1 DOWNTO 0);
```

```
BEGIN

addr <=  cnt;
       PROCESS(clk,clr)
       BEGIN

IF clr = RESET_ACTIVE THEN
                      cnt <= "00";
                     cout <= '0';

ELSIF   rising_edge(clk) THEN

IF cnt  = "10" THEN
                           cnt  <= "11";

cout <= '1';
                      ELSE

cnt <= cnt + '1';
                     END IF;
                END IF;
            END PROCESS;
 END behave;




LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY dl_reg IS
   PORT(

                     clk            :IN       STD_LOGIC;
                     reg_wr     :IN        STD_LOGIC;
                     en           :IN        STD_LOGIC;

                       addr              :IN        STD_LOGIC_VECTOR(1 DOWNTO 0);
                       data_in           :IN        STD_LOGIC_VECTOR(3 DOWNTO 0);

                      data_out              : OUT        STD_LOGIC_VECTOR(3 DOWNTO 0)
              );
 END dl_reg;
```

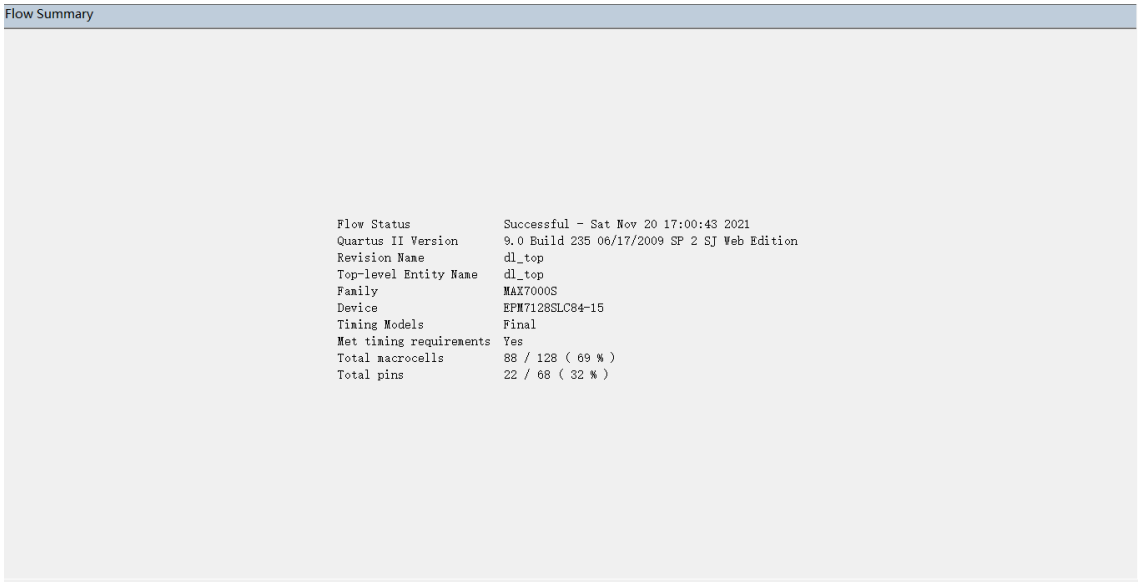
```
ARCHITECTURE     rtl of dl_reg     IS
            SIGNAL   m0          :STD_LOGIC_VECTOR(3 DOWNTO 0);
            SIGNAL   m1          :STD_LOGIC_VECTOR(3 DOWNTO 0);
            SIGNAL   m2          :STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
 PROCESS(clk)
  BEGIN

IF falling_edge(clk) THEN
            --密碼脉冲作内使能信号
           IF en='1' THEN
             CASE addr IS
                WHEN "01" =>
                           IF reg_wr = '1' THEN
                                     m0<= data_in;
                           ELSE

                                    data_out <= m0;
                           END IF;

                      WHEN "10" =>
                           IF reg_wr = '1'  THEN
                                m1 <= data_in;
                           ELSE
                                  data_out <= m1;
                            END IF ;
                     WHEN "11" =>
                            IF reg_wr = '1' THEN

                             m2 <= data_in;
                             ELSE

                               data_out <= m2;
                             END IF ;

                   WHEN OTHERS =>

                              NULL;
                                  END CASE;

                              END IF;

                         END IF;
```

```
                    END PROCESS;
  END rtl;


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;

ENTITY dl_cmp IS
  PORT(
              a                :IN          STD_LOGIC_VECTOR(3 DOWNTO 0);
              b                :IN          STD_LOGIC_VECTOR(3 DOWNTO 0);
              c                :OUT          STD_LOGIC

);
END dl_cmp;
ARCHITECTURE behave of dl_cmp IS
BEGIN
           c<='1'  WHEN a = b ELSE
                     '0';
END behave
;


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY dl_coder IS
     PORT(
                clk               :IN          STD_LOGIC;
                en               :IN          STD_LOGIC;
                   key_in             :IN          STD_LOGIC_VECTOR(9 DOWNTO 0);
                  ps_i               :OUT         STD_LOGIC;

                  code_out               :OUT          STD_LOGIC_VECTOR(3 DOWNTO 0)
           );
END dl_coder;
ARCHITECTURE rtl of dl_coder IS

SIGNAL key_in_1 :          STD_LOGIC_VECTOR(9 DOWNTO 0);
SIGNAL key_in_2 :          STD_LOGIC_VECTOR(9 DOWNTO 0);
```

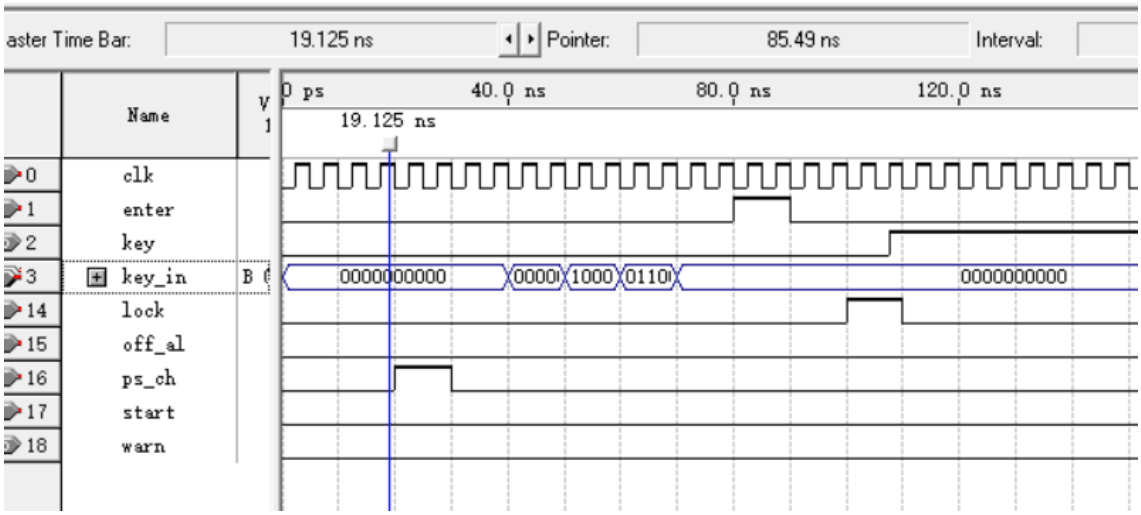
```
SIGNAL temp           : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
U1 : PROCESS (clk)
BEGIN
IF rising_edge(clk) THEN
                IF en = '1' THEN
                 key_in_2 <= key_in_1;
                          key_in_1 <= key_in;
                    END IF;
            END IF;
END PROCESS;

ps_i <='1'  WHEN    key_in_2 /= key_in_1  ELSE
 '0';

-- 10

U2 : PROCESS(clk)
BEGIN

       IF rising_edge(clk) THEN

             IF en = '1' THEN

                  CASE key_in IS

                        WHEN "0000000001" => temp <= "0000" ;
                        WHEN "0000000010" => temp <= "0001";
                        WHEN "0000000100" => temp <= "0010";
                        WHEN "0000001000" => temp <= "0011";
                        WHEN "0000010000" => temp <= "0100";
                        WHEN "0000100000" => temp <= "0101";
                        WHEN "0001000000" => temp <= "0110";
                        WHEN "0010000000" => temp <= "0111";
                        WHEN "0100000000" => temp <= "1000";
                        WHEN "1000000000" => temp <= "1001";
                        WHEN OTHERS  => temp <= "0000";

      END CASE;
               END IF;
           END IF;

END PROCESS;
```

```
code_out <= temp WHEN en = '1' ELSE
          "0000" ;

END rtl;
```

Flow Summary截图如下：



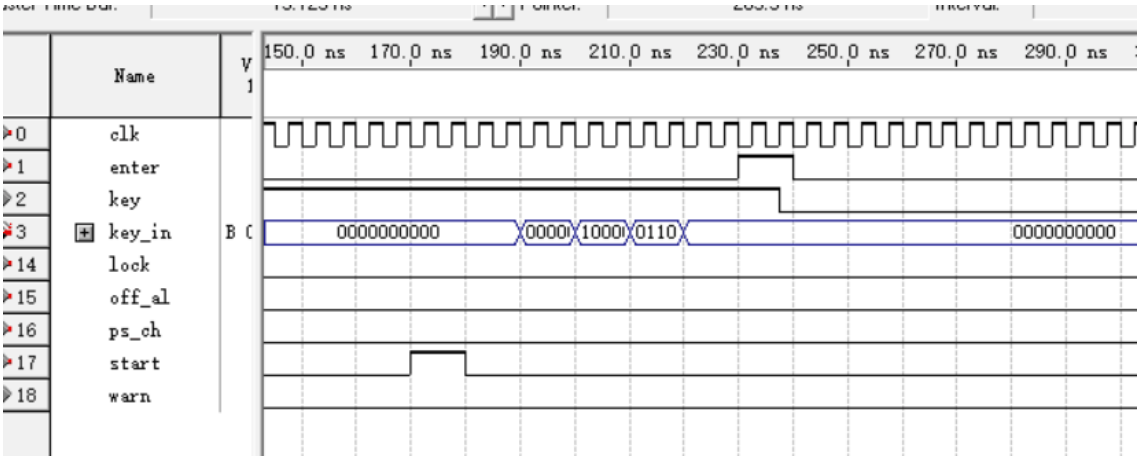
波形截图如下：



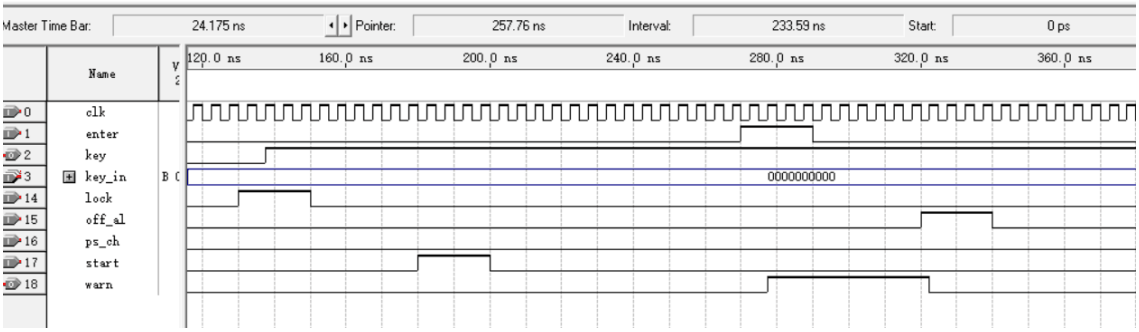
解释说明如下：

ps_ch为修改密码键高电平有效，输入key_in为密码输入端，输入新密码后，点击确认键enter高电平有效，此时新密码被输入。

Lock为锁定键，高电平有效，锁闭合准备开锁，start为输入密码键，高电平有效，此时允许输入密码，输入完毕后再按“enter”密码确认。



解释说明如下：先按“start”准备输入密码，密码输入完毕后再按“enter”确认密码输入完毕，此波形中密码正确，key由高电平变为低电平解锁。



解释说明如下：此波形图中密码输入错误，故enter确认输入完毕后，“warn”为报警端，高电平有效，密码错误进行报警，“off_al”为复位端，高电平有效，对报警端进行复位，警报解除。

五、实验注意事项

（1）两人一实验组（12分钟）接力完成，课前在课代表处排号并登记实验题目；请课代表注意：排号表中姓名前加学号后三位，并请第一位同学带到实验室。

（2）要求完全使用VHDL语言设计，不得使用其它设计输入方式（例如，画原理图）。

（3）VHD文件课前写好，设计流程其它步骤需到课演示，项目要求建在U盘上。

（4）CPLD开发板上使用的芯片是MAX7000S系列EPM7128SLC84-15。

（5）锁定管脚后要再次执行全程编译（必要步骤）。可以使用自己的电脑，但不能保证编程器兼容。

六、团队工作分配

项目设计：

徐驰翔 蒋君睿 李灏轩 钱俊鑫 闫鹏举 杨文华

进行预实验：

徐驰翔 李灏轩

报告撰写：

钱俊鑫

提高团队工作有效性的方法：

①对工作任务提前进行合理科学分配。

②团队成员间及时沟通，相互帮助解决问题。

③团队成员之间要相互激励，互相关怀。

七、实验心得

本次实验中我们感受到了VHDL语言的神奇和仿真实验功能的强大，只有将书本知识和实际实验相结合才有可能做出成功的成果！