西安电子科技大学

电子线路实验（II） 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 杨超 学号 20049200221

同作者 赵明宇 20049200176

周汉栋 20049200135

梁梦垚 20049200416

朱永怡 20049200499

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）

一、实验目的

1、熟悉大规模集成数字电路的设计方法

2、熟悉数字系统调试及故障排除方法

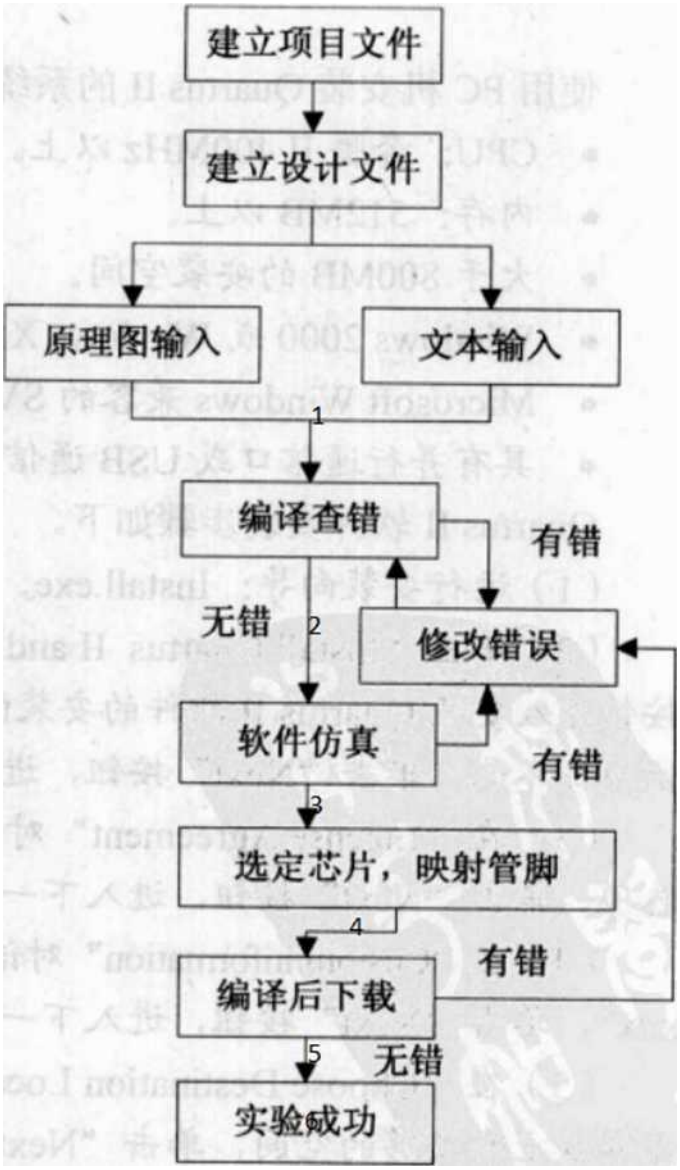
二、实验原理

早期的可编程逻辑器件，即 PLD,只有 PROM、EPROM 和 EEPROM 三种。由于结构限制,它们只能完成简单的数字逻辑；其后，出现了一类结构上稍复杂的可编程芯片，它能够完成 各种数字逻辑功能，这一阶段的产品主要有可编程陈列逻辑（PAL）和通用阵列逻辑（GAL）。这些早期的 PLD 器件的一个共同特点是可以实现速度特性较好的逻辑功能，但其过于简单 的结构也使它们只能实现规模较小的电路；为弥补这一缺陷，20 世纪 80 年代中期，Alta-a 和 Xilinx 公司分别推出了现场可编程门阵列（Field Programmable Gate Array, FPGA）,它 们都具有体系结构和逻辑单元灵活、集成度高以及适用范围宽等特点。

FPGA 的结构基于查找表技术。查找表（Looking Up Table, LUT）本质上就是一个 RAM, 当用户通过原理图或 HDL 代码描述了一个逻辑电路后，FPGA 软件会自动计算逻辑电路所 有结果，并把结果事先写入 RAM,这样，每输入一个信号进行逻辑运算就等于输入一个地 址进行查表，找出地址对应的内容，然后输出即可。FPGA 的编程单元基于 SRAM 结构， 从理论上讲，具有无限次重复编程的能力。

虽然内容实现的功能比较简单，但对于初学者来说，是不错的入门实例。简化的 Quaitusn 设计过程如下图所示。因为代码功能不正确或代码的编写风格不好对后期的设计会有很大的 影响，所以需要花很多时间在设计过程的仿真上。电路仿真的目的是确认功能正常，所以不 必完全照实际信号设置。例如 12M 分频，在用小模值仿真确认正确后，即可修改参数编译 下载。每次刚开始接触一个新东西的时候都会碰到困难，所以

希望大家提早准备，熟悉软件。



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

设计的出发点基于我们所掌握的基础知识，在日常的数字电子技术的学习中，我们掌握了很多的设计样例，有密码锁，洗衣机，数字时钟，乒乓球游戏机等等，还有新学习到的 ROM 存储器。根据日常生活，我们假想设计一个加密的云端存储器，可以实现我们在云端通过正确的数字密码来得到我们想要的信息，来方便我们的平时生活。基于我们要使用的芯片 EPM7128SLC84-15，我们依照想法进行了初

步的原理设计，满足的功能有输入正确的密码会输出数字，连续输入 3 次的错误密码会报错，可以修改密码，可以对存进去的信息进行修改和保存。

实现一个硬件控制的加密云端存储器，要求包含以下几部分：密码锁部分，信息存储部分。

密码锁部分功能包括：输入密码，密码正确则打开，密码错误则警报，修改密码。

信息存储部分功能包括：可以在输入密码后获得所需信息，并将其输出。

具体实验内容：

1、密码箱

代码:

顶层设计 dlock：

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

entity dlock is
    port(
    clk:in std_logic;
    lock:in std_logic;
    start:in std_logic;
    off_al:in std_logic;
    ps_ch:in std_logic;
    enter:in std_logic;
    key_in:in std_logic_vector(9 downto 0);
    key:out std_logic;
    warn:out std_logic
    );
end dlock;

architecture behave of dlock is

component dl_control
    port(
    clk:in std_logic;
    lock:in std_logic;
    start:in std_logic;
    off_al:in std_logic;
    ps_ch:in std_logic;
    enter:in std_logic;
    wro_count:in std_logic;
```

```
        ps_i:in std_logic;
        cmp_r:in std_logic;
        cin:in std_logic;
        code_en:out std_logic;
        cnt_clr:out std_logic;
        cnt_clr2:out std_logic;
        cnt_clk2:out std_logic;
        reg_wr:out std_logic;
        key:out std_logic;
        warn:out std_logic
        );
end component;

component dl_counter
        port(
        clr:in std_logic;
        clk:in std_logic;
        cout:out std_logic;
        addr:out std_logic_vector(1 downto 0)
        );
end component;

component dl_counter2
port(
        clk:in std_logic;
        clr:in std_logic;
        wro_count:out std_logic
        );
end component;

component dl_reg
        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 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(
        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 component;

signal code_en:std_logic;
signal cnt_clr:std_logic;
signal cnt_clr2:std_logic;
signal cnt_clk2:std_logic;
signal reg_wr:std_logic;
signal wro_count: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
        CONTROL:dl_control
        port
map(clk=>clk,lock=>lock,start=>start,off_al=>off_al,ps_ch=>ps_ch,enter=>enter,wro_count=>
wro_count,
                ps_i=>ps_i,

        cmp_r=>cmp_r,cin=>cin,code_en=>code_en,cnt_clr=>cnt_clr,cnt_clr2=>cnt_clr2,cnt_clk2
=>cnt_clk2,
                reg_wr=>reg_wr,key=>key,warn=>warn
                );

        CONUTER:dl_counter
        port map(clr=>cnt_clr,clk=>ps_i,cout=>cin,addr=>addr);

        COUNTER2:dl_counter2
```

```
    port map(clr=>cnt_clr2,clk=>cnt_clk2,wro_count=>wro_count);

    REG:dl_reg
    port
map(clk=>clk,reg_wr=>reg_wr,en=>ps_i,addr=>addr,data_in=>code_out,data_out=>data_out);

    COMPARATOR:dl_cmp
    port map(a=>code_out,b=>data_out,c=>cmp_r);

    CODER:dl_coder
    port map(clk=>clk,en=>code_en,key_in=>key_in,ps_i=>ps_i,code_out=>code_out);

end behave;
```

reg 模块代码：

```
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 entity;

architecture behave 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
            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 behave;
```

counter 模块代码：

```
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;
```

counter2 模块代码：

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

entity dl_counter2 is
port(
    clk:in std_logic;
    clr:in std_logic;
    wro_count:out std_logic
    );
end entity;

architecture behave of dl_counter2 is
signal cnt:std_logic_vector(1 downto 0);
begin
    process(clk,clr)
    begin
        if clr='1' then
        cnt<="00";
        wro_count<='0';
        elsif rising_edge(clk) then
            if cnt="10" then
                cnt<="11";
                wro_count<='1';
            else
                cnt<=cnt+'1';
            end if;
        end if;
```

```
    end process;
end behave;
```

control 模块代码：

```
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(
    clk:in std_logic;
    lock:in std_logic;
    start:in std_logic;
    off_al:in std_logic;
    ps_ch:in std_logic;
    enter:in std_logic;
    wro_count:in std_logic;
    ps_i:in std_logic;
    cmp_r:in std_logic;
    cin:in std_logic;
    code_en:out std_logic;
    cnt_clr:out std_logic;
    cnt_clr2:out std_logic;
    cnt_clk2:out std_logic;
    reg_wr:out std_logic;
    key:out std_logic;
    warn:out std_logic
    );
end dl_control;

architecture behave of dl_control is
CONSTANT KEY_ACTIVE:STD_LOGIC:='1';
type state_type is
(OUTLOCK,INLOCK,PS_INPUT,PS_RIGHT,PS_WRONG,ALARM,PS_CHANGE);
signal state:state_type;

begin

    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;
    when PS_INPUT=>
        code_en<='1';
        cnt_clr<='0';
        reg_wr<='0';
        if cin='1' and ps_i='1' and cmp_r='1' then
            code_en<='0';
            cnt_clr<='1';
            cnt_clr2<='1';
            state<=PS_RIGHT;
        elsif ps_i='1' and cmp_r='0' then
            code_en<='0';
            cnt_clr<='1';
            cnt_clr2<='0';
            cnt_clk2<='1';
            state<=PS_WRONG;
        elsif enter=KEY_ACTIVE and cin='0' then
            code_en<='0';
            cnt_clr<='1';
            cnt_clr2<='0';
            cnt_clk2<='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 and wro_count='1' then
                        cnt_clk2<='0';
                        state<=ALARM;
                    elsif enter=KEY_ACTIVE then
                        cnt_clk2<='0';
                        state<=INLOCK;
                    else
                        state<=PS_WRONG;
                    end if;
                when ALARM=>
                    if off_al=KEY_ACTIVE then
                        warn<='0';
                        state<=INLOCK;
                    else
                        cnt_clk2<='0';
                        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;
```

coder 模块代码：

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

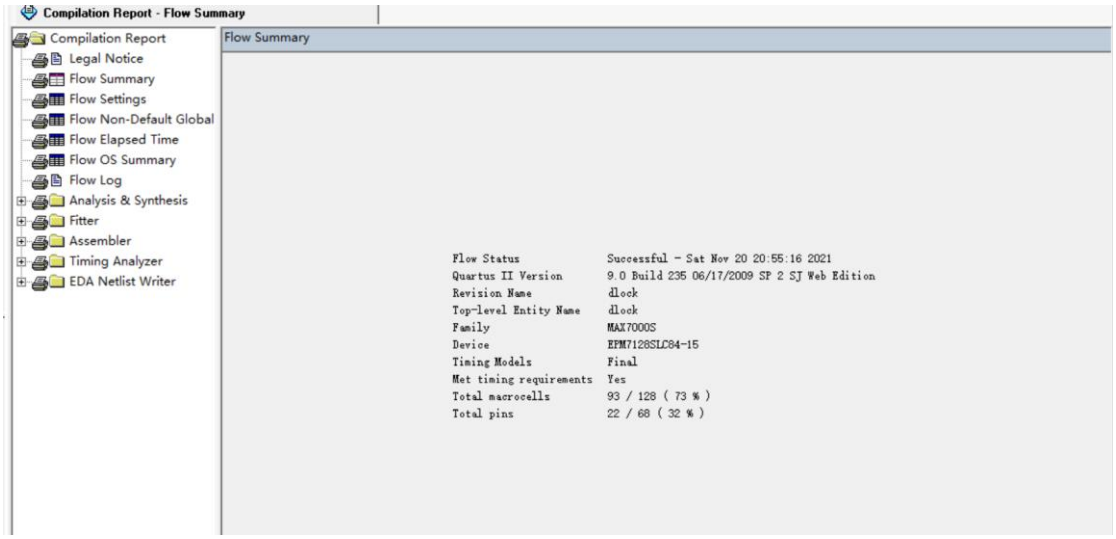
```
use ieee.std_logic_arith.all;

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 behave of dl_coder is
signal key_in_1:std_logic_vector(9 downto 0);
signal key_in_2:std_logic_vector(9 downto 0);
begin
	U1:process(clk)
	begin
		if rising_edge(clk) then
			if en='1' then
				if key_in="0000000000" then
				key_in_1<=key_in;
				key_in_2<=key_in;
				ELSE
				key_in_2<=key_in_1;
				key_in_1<=key_in;
				end if;
			end if;
		end if;
	end process;

	ps_i<='1' when key_in_2/=key_in_1 else '0';

	U2:process(clk)
	begin
		if rising_edge(clk) then
			if en='1' and key_in/="000000000" then
				case key_in is
				when "0000000001"=>code_out<="0000";
				when "0000000010"=>code_out<="0001";
				when "0000000100"=>code_out<="0010";
				when "0000001000"=>code_out<="0011";
				when "0000010000"=>code_out<="0100";
				when "0000100000"=>code_out<="0101";
```

```
                    when "0001000000"=>code_out<="0110";
                    when "0010000000"=>code_out<="0111";
                    when "0100000000"=>code_out<="1000";
                    when "1000000000"=>code_out<="1001";
                    when OTHERS=>code_out<="0000";
                    end case;
                end if;
            end if;
        end process;
end behave;
```

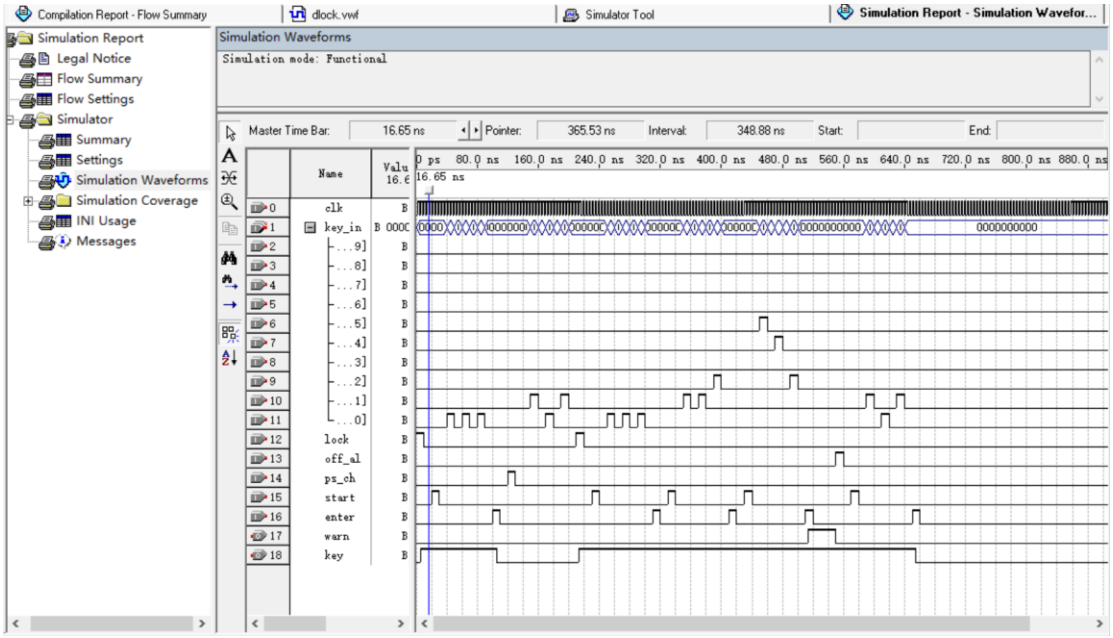
cmp 模块代码：

```
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;
```

Flow summary 截图



仿真实验结果截图



仿真结果说明：lock 为上锁，key 为高电平表示上锁状态，初始密码 000，ps-ch 为高电平时表示修改密码，如图所示我们将密码修改为 101，接下来我们分别输入 000，112，542，key 始终为高电平表示解锁失败，同时 warn 在三次输入密码错误之后报警，在让 off-al 高电平使得报警消除，之后我们再输入正确密码 101，此时 key 变成低电平表示解锁成功。

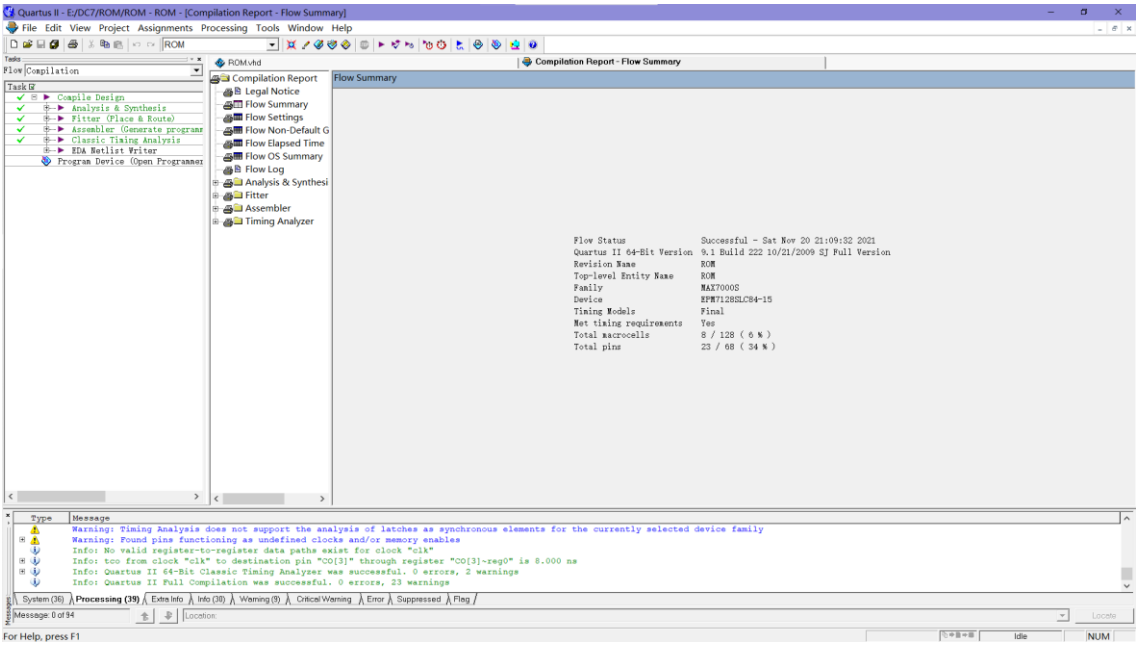
2、信息存储器

代码:

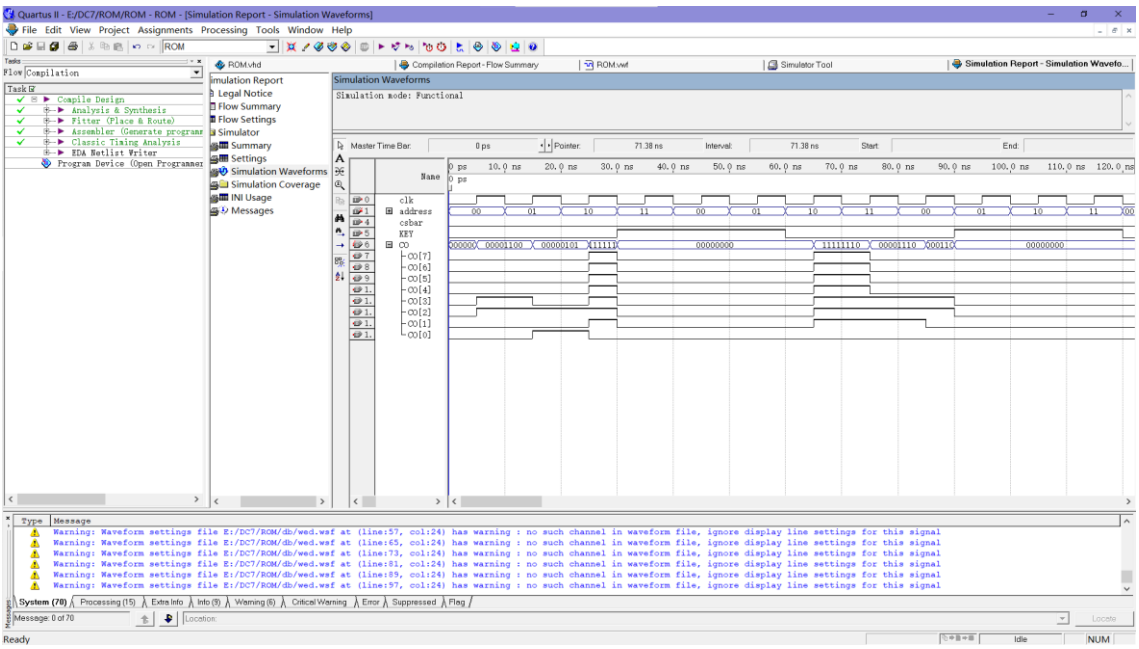
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY ROM IS
  PORT(  address : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
```

```
        csbar,clk ,KEY: IN STD_LOGIC;
        CO : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END ROM;

ARCHITECTURE version1 OF ROM IS
 BEGIN
PROCESS(csbar,address,clk)

TYPE rom_array IS ARRAY (0 TO 3) OF BIT_VECTOR(7 DOWNTO 0);
VARIABLE index : INTEGER := 0;
CONSTANT rom_store : rom_array :=
 (0 =>   X"0c",
 1 => X"05",
                2 => X"fe",
 3 =>X"0e",
                OTHERS => X"00");
BEGIN
index := 0;
IF csbar = '0' THEN
If(address = "00")THEN
index :=0;
elsif(address = "01")THEN
index :=1;
elsif(address = "10")THEN
index :=2;
elsif(address = "11")THEN
index :=3;
end if;
IF(KEY = '0')then
IF clk'event and clk='1' THEN
co <=TO_StdlogicVector(rom_store(index));
END IF;
ELSE CO<=X"00";
END IF;
ELSE
CO<=X"00";
END IF;
END PROCESS;
END version1;
```

Flow summary 截图:



仿真结果截图：




```
CONSTANT rom_store : rom_array :=
 (0 =>  X"0c",
 1 => X"05",
 2 => X"fe",
 3 =>X"0e",
 OTHERS => X"00");
```

仿真结果说明：KEY=0,cabar=0,在 clk 的上升沿，CO 输出对应的存储的信息。

address=00，输出 00001100；adress=01，00000101；adress=10，输出 11111110；adress=11，输出 00001110。

3、云端加密存储器

顶层设计：

代码：library ieee;

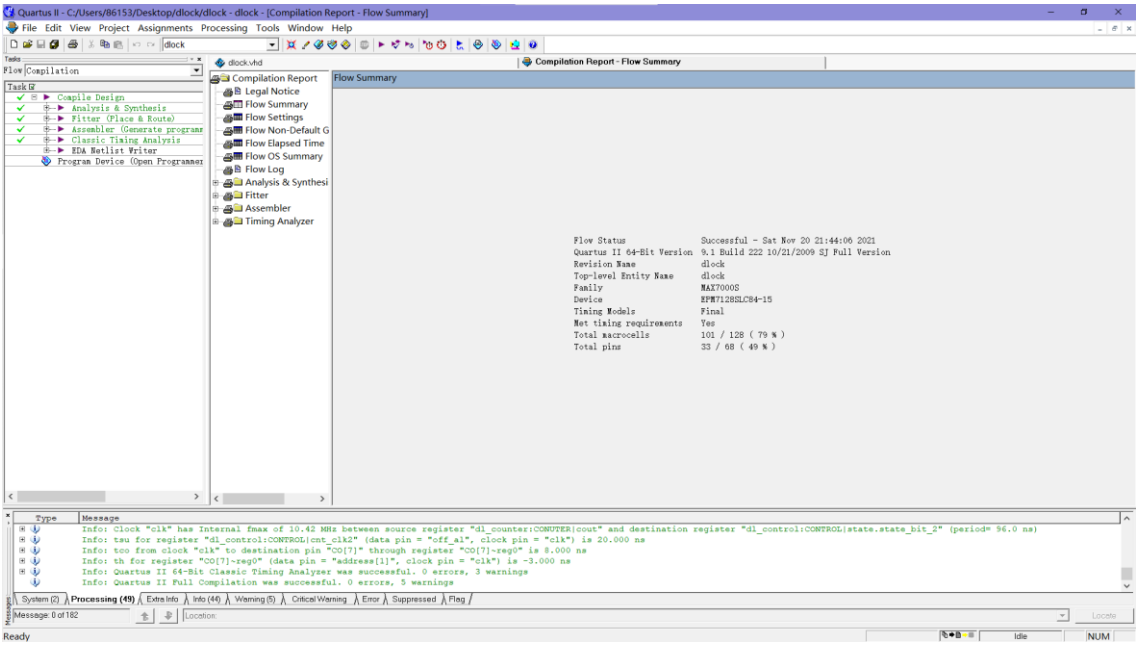
```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity dlock is
    port(
    clk:in std_logic;
    lock,csbar:in std_logic;
    start:in std_logic;
    off_al:in std_logic;
    ps_ch:in std_logic;
    enter:in std_logic;
    key_in:in std_logic_vector(9 downto 0);
    address:in std_logic_vector(1 downto 0);
    key:buffer std_logic;
    warn:out std_logic;
    CO:OUT std_logic_VECTOR(7 downto 0)
    );
end dlock;

architecture behave of dlock is


component dl_control
    port(
    clk:in std_logic;
    lock:in std_logic;
    start:in std_logic;
    off_al:in std_logic;
    ps_ch:in std_logic;
    enter:in std_logic;
    wro_count:in std_logic;
    ps_i:in std_logic;
    cmp_r:in std_logic;
    cin:in std_logic;
    code_en:out std_logic;
    cnt_clr:out std_logic;
    cnt_clr2:out std_logic;
    cnt_clk2:out std_logic;
    reg_wr:out std_logic;
    key:out std_logic;
    warn:out std_logic
```

```
        );
end component;

component dl_counter
        port(
        clr:in std_logic;
        clk:in std_logic;
        cout:out std_logic;
        addr:out std_logic_vector(1 downto 0)
        );
end component;

component dl_counter2
port(
        clk:in std_logic;
        clr:in std_logic;
        wro_count:out std_logic
        );
end component;

component dl_reg
        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 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(
        clk:in std_logic;
        en:in std_logic;
        key_in:in std_logic_vector(9 downto 0);
```

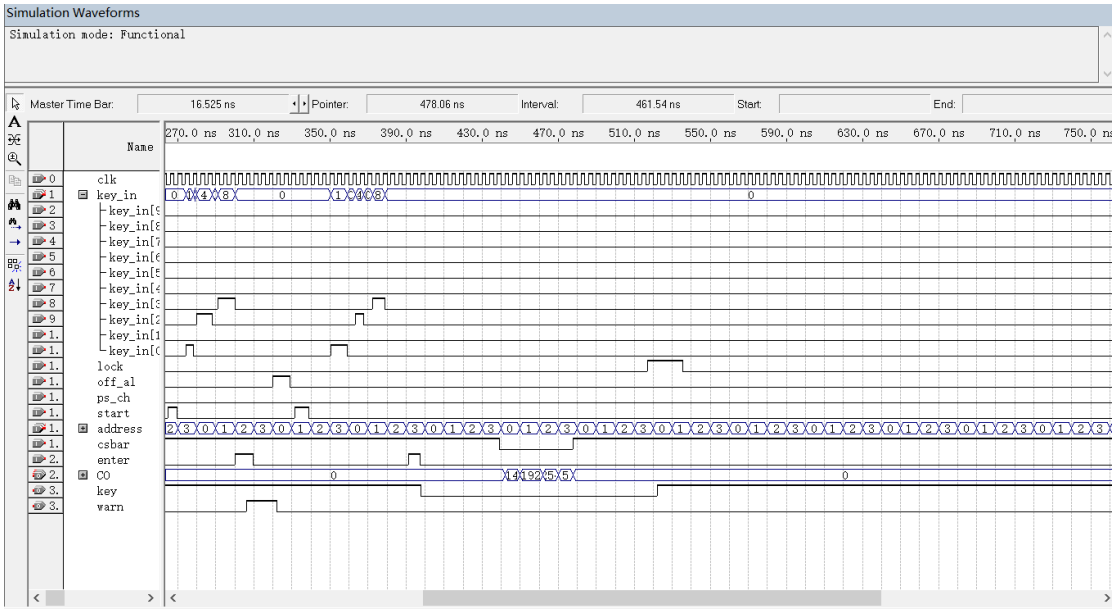
```
        ps_i:out std_logic;
        code_out:out std_logic_vector(3 downto 0)
        );
end component;

signal code_en:std_logic;
signal cnt_clr:std_logic;
signal cnt_clr2:std_logic;
signal cnt_clk2:std_logic;
signal reg_wr:std_logic;
signal wro_count: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);
SIGNAL CNT:std_logic_VECTOR(7 DOWNTO 0);

begin
        CONTROL:dl_control
        port
map(clk=>clk,lock=>lock,start=>start,off_al=>off_al,ps_ch=>ps_ch,enter=>enter,wro_count=>wro_c
ount,
                        ps_i=>ps_i,

        cmp_r=>cmp_r,cin=>cin,code_en=>code_en,cnt_clr=>cnt_clr,cnt_clr2=>cnt_clr2,cnt_clk2=>cnt
_clk2,
                        reg_wr=>reg_wr,key=>key,warn=>warn
                        );

        CONUTER:dl_counter
        port map(clr=>cnt_clr,clk=>ps_i,cout=>cin,addr=>addr);

        COUNTER2:dl_counter2
        port map(clr=>cnt_clr2,clk=>cnt_clk2,wro_count=>wro_count);

        REG:dl_reg
        port
map(clk=>clk,reg_wr=>reg_wr,en=>ps_i,addr=>addr,data_in=>code_out,data_out=>data_out);

        COMPARATOR:dl_cmp
        port map(a=>code_out,b=>data_out,c=>cmp_r);
```

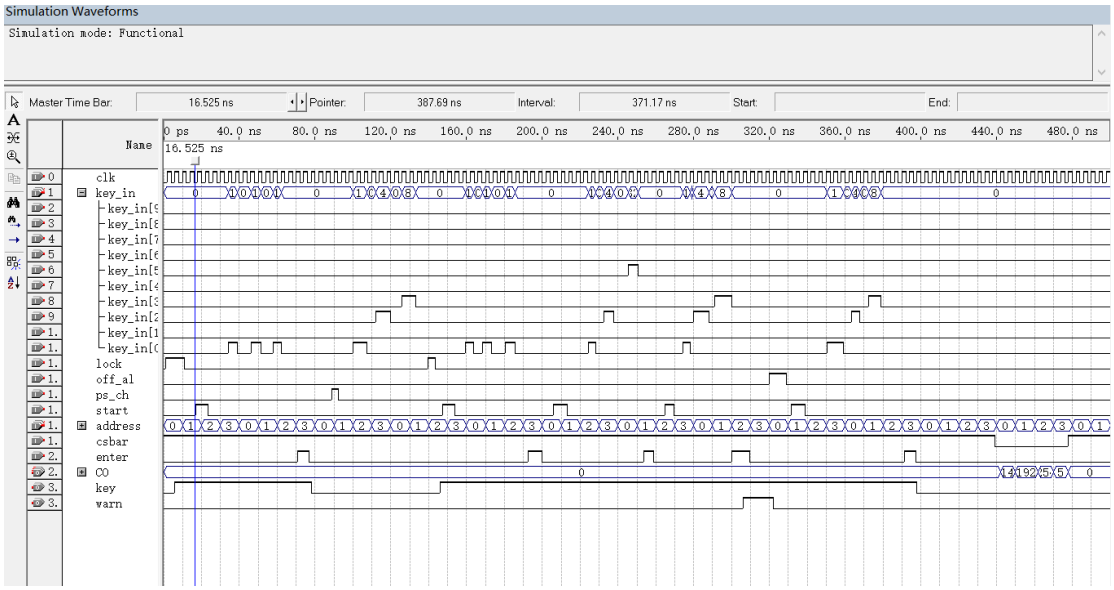
```
    CODER:dl_coder
    port map(clk=>clk,en=>code_en,key_in=>key_in,ps_i=>ps_i,code_out=>code_out);


        PROCESS(address, csbar)
        TYPE ram_array IS ARRAY (0 TO 3) OF BIT_VECTOR(7 DOWNTO 0);
        VARIABLE index : INTEGER := 0;

        CONSTANT ram_store : ram_array :=
            (0 => X"0e",
            1 => X"c0", --lda $FE
            2 => X"fe",
            3 => X"05", --ldb $FF
            OTHERS => X"00");

        BEGIN
        index := 0;
            IF csbar = '0' THEN
 --calculate address as an integer

                if(address = "00")then
                index :=0;
                elsif(address = "01")then
                index :=1;
                elsif(address = "10")then
                index :=2;
                elsif(address = "11")then
                index :=3;
                end if;
            if key = '0'then
                if(clk'event and clk= '1')then
                co<=to_stdlogicvector(ram_store(index));
                end if;
            else
                co <= x"00";
            end if;
            else
                co <= x"00";
            end if;

        end process;

end behave;
```

Flow summary 截图:


Quartus II - C:/Users/86153/Desktop/dlock/dlock - dlock - [Compilation Report - Flow Summary]
File Edit View Project Assignments Processing Tools Window Help
Compilation Report - Flow Summary
Flow Summary
Flow Status Successful - Sat Nov 20 21:44:06 2021
Quartus II 64-Bit Version 9.1 Build 222 10/21/2009 SJ Full Version
Revision Name dlock
Top-level Entity Name dlock
Family MAX7000S
Device EPM7128SLC84-15
Timing Models Final
Met timing requirements Yes
Total macrocells 101 / 128 (79 %)
Total pins 33 / 68 (49 %)
Info: Clock "clk" has Internal fmax of 10.42 MHz between source register "dl_counter:COMUPER|cout" and destination register "dl_control:CONTROL|state.state_bit_2" (period= 96.0 ns)
Info: tsu for register "dl_control:CONTROL|cnt_clk2" (data pin = "off_al", clock pin = "clk") is 20.000 ns
Info: tco from clock "clk" to destination pin "CO[7]" through register "CO[7]~reg0" is 8.000 ns
Info: th for register "CO[7]~reg0" (data pin = "address[1]", clock pin = "clk") is -3.000 ns
Info: Quartus II 64-Bit Classic Timing Analyzer was successful. 0 errors, 3 warnings
Info: Quartus II Full Compilation was successful. 0 errors, 5 warnings
Ready

仿真实验结果截图：


Simulation Waveforms
Simulation mode: Functional
Master Time Bar: 16.525 ns Pointer: 478.06 ns Interval: 461.54 ns Start: End:
clk
key_in
lock
off_al
ps_ch
start
address
csbar
enter
CO
key
warn



仿真功能说明：

lock 为上锁，key 为高电平表示上锁状态，初始密码 000，ps-ch 为高电平时表示修改密码，如图所示我们将密码修改为 023，接下来我们分别输入 000，025，023，key 始终为高电平表示解锁失败，同时 warn 在三次输入密码错误之后报警，在让 off-al 高电平使得报警消除，之后我们再输入正确密码 023，key 为低电平表示解锁成功。对于云端数据加密存储器而言，不仅需要密码锁打开，同时需要在 csbar 为低电平时才可以读取数据，保存的 4 个 8 位数据为按照顺序为 00001100，00000101，11111110，00001110，其余任何情况 co 输出都为 0，即数据不输出的情况。

五、实验中的经验与教训

实验过程中出现的问题以及解决方法

由于是一项较为复杂与庞大的工程，并且我们把自己的设计与想法加入了想要完成的项目中，所以在诸多方面遇到了问题，

有语法不精通导致的语法问题：

1. CASE 语句赋值条件不完整而导致的错误。
2. 在多个 if 语句嵌套使用时，缺少对应的 end if 而导致错误。
3. 对变量的赋值语句错写成<=或者=。
4. 标点符号的中英文输入导致程序编译不成功。
5. 没有养成良好的代码编写习惯，没有习惯缩进，导致代码报错时寻找错误异常麻烦。

还有波形仿真时遇到的问题：

1. 上升沿和下降沿的长短，选取的位置不同而导致的错误。
2. 还有功能较多时如何完整的把功能全部演示出来的问题。

等等诸如此类因为语法不熟导致的错误和设计过程中想法不成熟而导致的问题，在我们勤翻语法书，查找各种资料和小伙伴的互帮互助中解决了问题。

最后从想法，讨论，设计到代码，波形的仿真，我们大约持续了 3 天的时间，加上学习程序，我们大致花费了一周的时间来完成我们的这个想法，最后的结果是我们感到非常满意。

实验中的经验教训

在实验时，常遇到波形仿真结果和预期不符的情况，有的是因为设计的问题，有的则是因为代码的问题。经过这次的设计实验，让我们深刻的意识到了从想法到一个完整的程序并

不是一个简单的事情，需要遇到很多的问题与困难，在我们的不懈努力之下，较好的完成了我们的想法。经过此次设计实验，我们对 VHDL 的掌握得到了极大的提高，对数字电子技术也有了更加深刻的理解。

六、自评

杨超	100
赵明宇	75
周汉栋	75
朱永怡	75
梁梦垚	75