四位数字密码锁设计

姓名：田丰

专业：2012级通信工程

邮箱：dtianf@163.com

摘要

由于电子密码锁具有安全性高、功耗低、易操作等优势，使得其越来越受到人们的青睐。本设计利用QuartusII9.0软件平台和大规模可编程逻辑器件FPGA，完成了电子密码锁的功能设计与仿真，简化了密码锁的结构，提高了密码锁的保密性和可靠性。

关键词：密码锁，FPGA ，QuartusII 9.0，仿真

目录

1 引言

随着电子技术的发展，具有防盗报警功能的电子密码锁越来越收到人们的青睐，用其代替密码量少、安全性差的机械密码锁已是必然趋势。电子密码锁与普通机械锁相比，具有无可比拟的优越性，如保密性好、防盗性强、可以不用钥匙记住密码即可开锁等。目前使用的电子密码锁大部分是基于单片机技术，以单片机为主要器件，以软件的方式生成编码器与解码器。而在实际应用中，由于程序容易跑飞，使得系统的可靠性比较差。基于 FPGA 器件的电子密码锁，所有算法完全由硬件电路实现，使得系统的可靠性大为提高。

2 设计的主要内容和要求

设计一个简单的数字电子密码锁，密码为 4 位。要求具备如下功能：

(1)如果输入数字键，第一个数字会从数码管的最右端开始显示，此后每按下一个数字键，数码管上的数字必须往左移动一格，以便将新的数字显示出来。

(2)本密码锁为四位密码锁，当输入的数字超过四个时，不会显示第四个以后的数字。

(3)按下密码清零键，清除所有输入的数字，清除成为“0000”，即做归零动作。

(4)按下解锁键，检查输入的密码是否正确，若解锁指示灯（绿色LED灯)闪烁一次，即表示密码正确（开锁）。

(5)按下改密键，将当前输入的数字设置成新密码，且上锁指示灯（红色LED灯）闪烁一次，即密码锁已上锁。

3 整体设计方案

本系统采用模块化的设计，整个系统分为数字按键输入、时钟输入、功能按键输入、数字译码块、功能译码模块、核心处理模块、输出处理模块、显示译码电路八个模块。整体电路如图3.1所示。

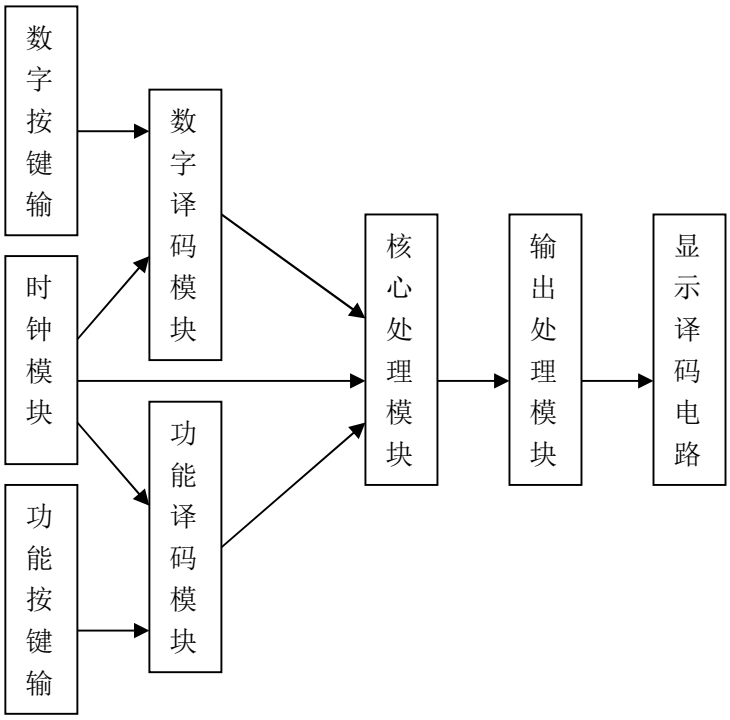

图3.1 数字密码锁总方框图

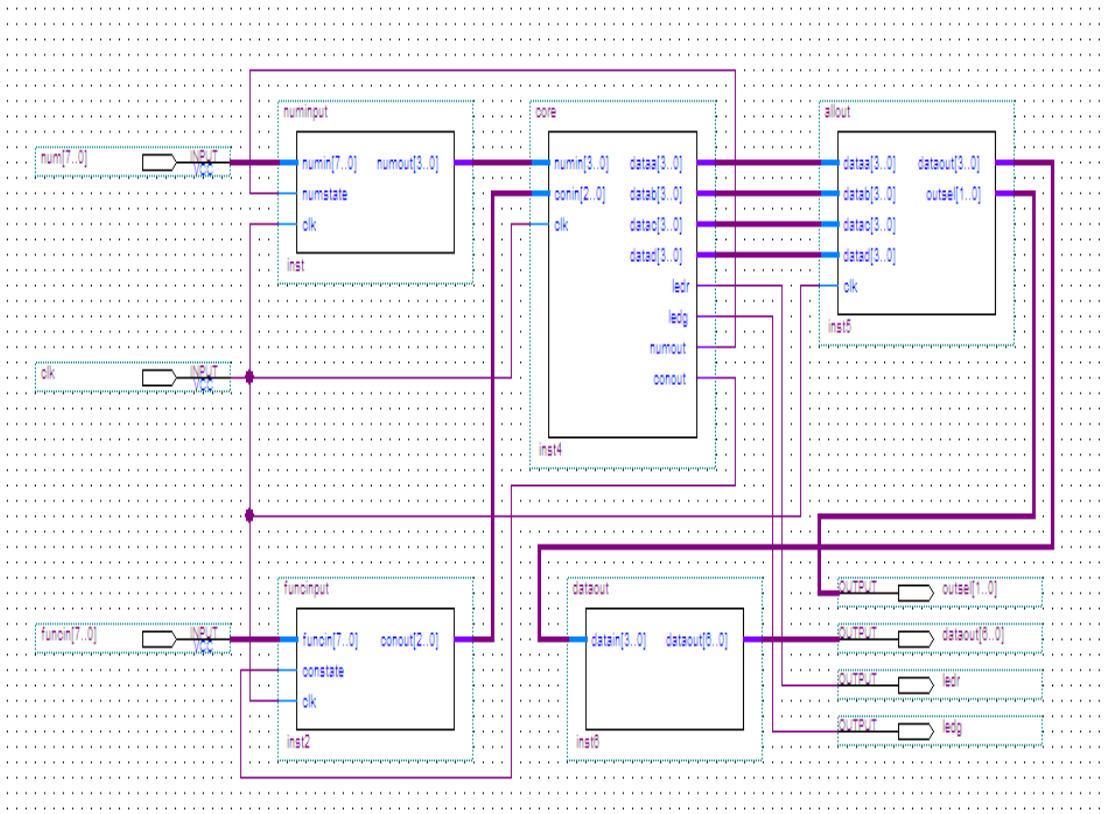

图3.2 四位密码锁的电路设计原理图

4 硬件电路的设计

4 位数字密码锁包括五个基本逻辑模块，分别为：数字按键输入模块（numinput）、功能按键输入模块（funcinput）、核心处理模块（core）、输出处理模块（allout）、七段译码器模块（dataout）。

4.1 数字按键输入--numinput

说明：读取数字键 0～9。按键为矩阵形式，高电平表示按键未按下，低电平表示按键按下。经数字按键输入模块处理后输出 4 位二进制代码，“0000”～“1001”分别表示 0～9，用“1010”表示其他无效输入。如表 4.1 所示。

表 4.1 数字按键输入模块（numinput）的数据输入输出

按键数字	按键扫描输出	Numinput 二进制输出	对应十进制数字
0	11011110	0000	0
1	01111101	0001	1
2	01111110	0010	2
3	10110111	0011	3
4	10111011	0100	4
5	10111101	0101	5
6	10111110	0110	6
7	11010111	0111	7
8	11011011	1000	8
9	11011101	1001	9
其他按键	其他	1010	10

（1）numinput--数字按键输入模块程序：

```
library ieee;
use ieee.std_logic_1164.all;
entity numinput is
```

```
port(numin        :IN std_logic_vector(7 downto 0);
numstate,clk   :IN std_logic;
numout       :OUT std_logic_vector(3 downto 0));
end numinput;
architecture one of numinput is
signal state    :std_logic;
signal mem    :std_logic_vector(7 downto 0);
begin
process(clk)
begin
if clk'event and clk='1' then
if state/=numstate then
if mem/=numin then
case numin is
when "11011110" => numout<="0000";---"0"
when "01111101" => numout<="0001";---"1"
when "01111110" => numout<="0010";---"2"
when "10110111" => numout<="0011";---"3"
when "10111011" => numout<="0100";---"4"
when "10111101" => numout<="0101";---"5"
when "10111110" => numout<="0110";---"6"
when "11010111" => numout<="0111";---"7"
when "11011011" => numout<="1000";---"8"
when "11011101" => numout<="1001";---"9"
when others => numout<="1010";
end case;
state<=numstate;
else numout<="1010";
end if;
```

```
mem<=numin;
end if;
end if;
end process;
end one;
```

（2）数字按键输入模块仿真图：

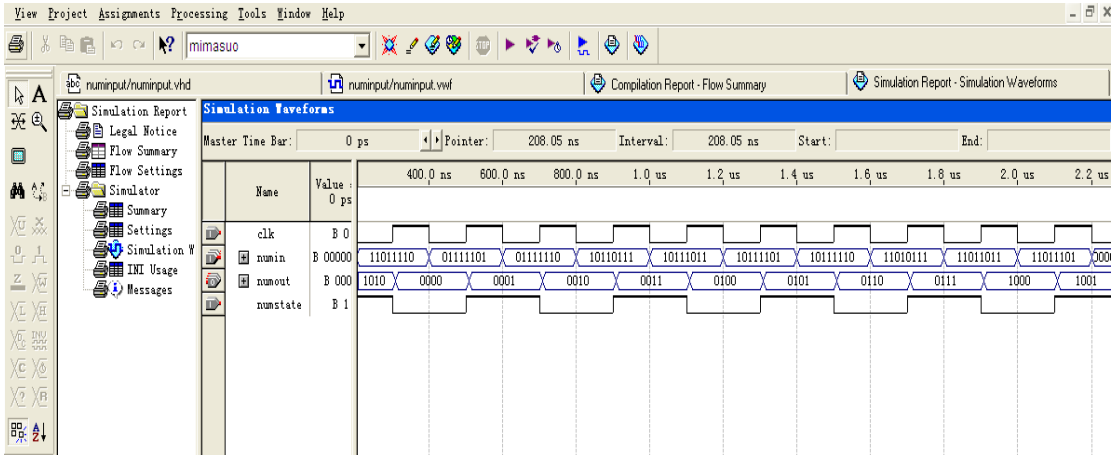

图 4.1 数字按键输入模块仿真图

由图可知，当数字按键输入模块的输入依次为"11011110"、"01111101"、"01111110"、"10110111"、"10111011"、"10111101"、"10111110"、"11010111"、"11011011"、"11011101"时，numout 输出依次输出“0000”、“0001”、“0010”、“0011”、“0100”、“0101”、“0110”、“0111”、“1000”、“1001”；当为其他按键输入时，numout 输出均为“1010”。

（3）数字按键输入--numinput 符号文件：

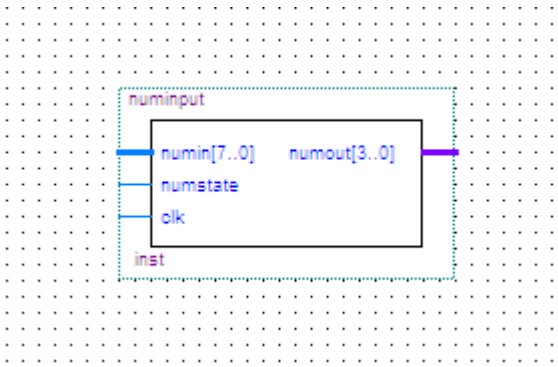

图 4.2 数字按键输入--numinput 符号文件

4.2 功能按键输入模块 -- funcinput

读取矩阵按键区控制功能按键 一清除键、改密键、上锁键、解锁键。高电平表示按键未按下，低电平表示按键按下。按照“清除，改密，上锁，解锁”顺序读取按键时，只能输出一位控制信号。输出的信号为 3 位二进制代码,“001”～“100”，依次表示“清除按键、改密按键、上锁，解锁”，用“000”表示输入不为功能按键信号。

表 4.2 控制功能按键输入模块（funcinput）的输出输入数据

功能按键	功能按键扫描输出	Funcinput 二进制输出	对应十进制数字
清除键	11100111	001	1
改密键	11101101	010	2
上锁键	11101110	011	3
解锁键	11101011	100	4
其他按键	其他	000	0

（1）Funcinput--功能按键输入模块程序：

```
library ieee;
use ieee.std_logic_1164.all;
entity funcinput is
    port( funcin:    IN std_logic_vector(7 downto 0);
    constate,clk:    IN std_logic;
conout:OUT std_logic_vector(2 downto 0));
end funcinput;
architecture one of funcinput is
    signal state:    std_logic;
    signal mem:      std_logic_vector(7 downto 0);
begin
process(clk)
begin
```

```
    if clk'event and clk='1' then
        if constate/=state then
        state<=constate;
    if mem/=funcin then
with  funcin  select
conout<="001"   when "11100111" , --清除键
        "010"   when "11101101",--改密键
        "011"   when "11101110", --上锁键
        "100"   when "11101011" , --解锁键
        "000"   when others;
        mem<=funcin;
        else conout<="000";
        end if;
    end if;
end if;
end process;
end one;
```

（2）功能按键输入模块仿真图：

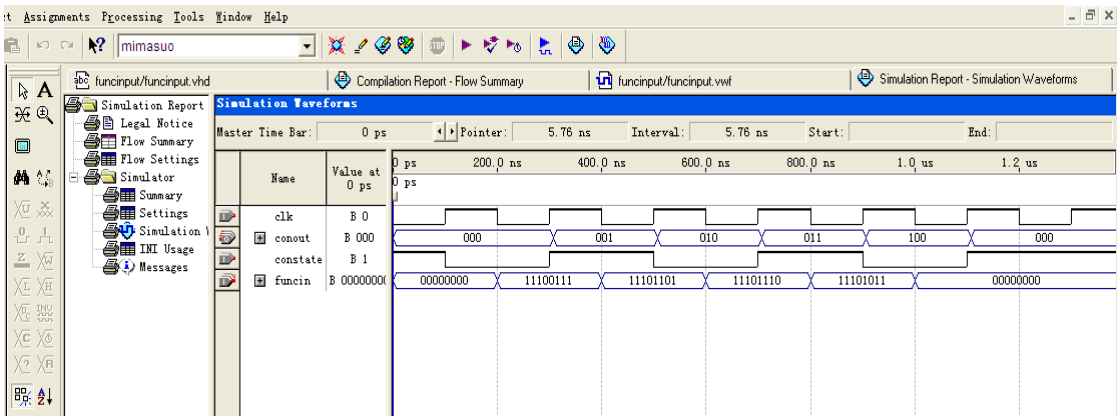

图 4.3 功能按键输入模块仿真图

由图可知：当功能按键输入模块的输入依次为“11100111”、“ 11101101”、“ 11101110”、“ 11101011”时，conout 输出依次为“001”、“010”、“011”、“100”， 当为其他按键输入时，conout 输出均为“000”。

（3）功能按键输入模块--funcinput 符号文件：

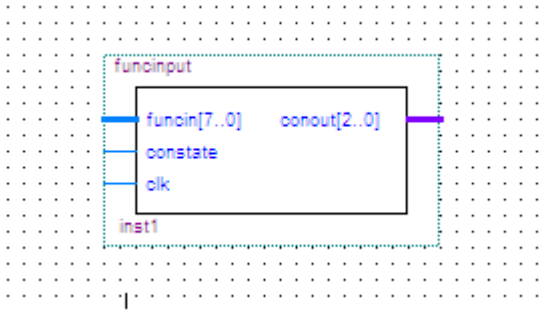

图 4.4　功能按键输入模块 --funcinput 符号文件

4.3 核心处理模块--core

核心处理模块将根据输入（数字按键输入以及功能按键输入）来改变存储器状态、数码管显示以及LED灯显示（红灯亮为上锁，绿灯亮为解锁）。

（1）Core--核心处理程序：

```
library ieee;
use ieee.std_logic_1164.all;
entity core is
    port( numin:     in std_logic_vector(3 downto 0);
    conin:        in std_logic_vector(2 downto 0);
    clk:             in std_logic;
    dataa,datab,datac,datad:       out  std_logic_vector(3  downto
0);
    ledr,ledg,numout,conout:       out std_logic);
end entity;
architecture one of core is
    type lockstate is (unlock,locked);
    signal
numa,numb,numc,numd,codea,codeb,codec,coded:std_logic_vector(3
downto 0);
    signal numstate,constate:      std_logic;
    signal locksta:            lockstate;
begin
```

```
process(clk,numin,conin)
begin
    if clk'event and clk='1' then--上升沿
           if numin/="1010" then
               numd<=numc;
               numc<=numb;
               numb<=numa;
               numa<=numin;
           end if;
    if conin/="000" then
           if conin="001" then--清除键按下
               numa<="0000";--全部清零
               numb<="0000";
               numc<="0000";
               numd<="0000";
    elsif conin="010" then --改密键按下
           if locksta/=locked then--锁并不是上锁状态
               codea<=numa;
               codeb<=numb;
               codec<=numc;
               coded<=numd;
           end if;
    elsif conin="011" then  --上锁键按下
           if locksta/=locked then
                  numa<="0000";
               numb<="0000";
               numc<="0000";
               numd<="0000";
               locksta<=locked;  --锁定密码锁
```

```
        end if;
elsif conin="100" then  --解锁键按下
        if locksta=locked then
            if numa=codea and numb=codeb then  --输入正确密码
                if numc=codec and numd=coded then
                    locksta<=unlock; --锁开
                end if;
            end if;
        end if;
    end if;
end if;
if locksta=locked then  --若锁锁定
        ledr<='1';      --led等高电平，红灯闪烁
        ledg<='0';
    else
        ledr<='0';
        ledg<='1';
    end if;
        dataa<=numa;
        datab<=numb;
        datac<=numc;
        datad<=numd;
    if numstate='1' then numstate<='0';
            else numstate<='1';
        end if;
    if constate='1' then constate<='0';
        else constate<='1';
        end if;
            numout<=numstate;
```

```
                conout<=constate;
            end if;
    end process;
    end one;
```

（2）核心处理模块仿真图：

当数字按键输入为有效输入（即输入的按键为数字按键 0～9 时），此时虽输入 9 位数字，但只有前四位有效，所以密码应为“1234”。 若 conin 输入为“011”（即按下上锁键），此时可以看到 ledr 为高电平，所以此时红灯闪烁一次，表示密码锁已上锁。仿真波形图如下：

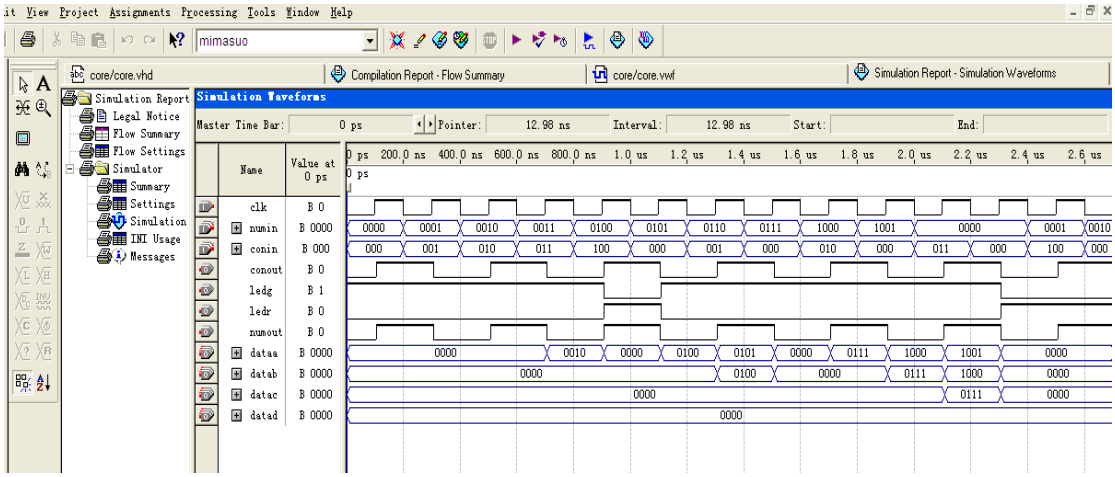

图 4.5　核心处理模块仿真图（一）

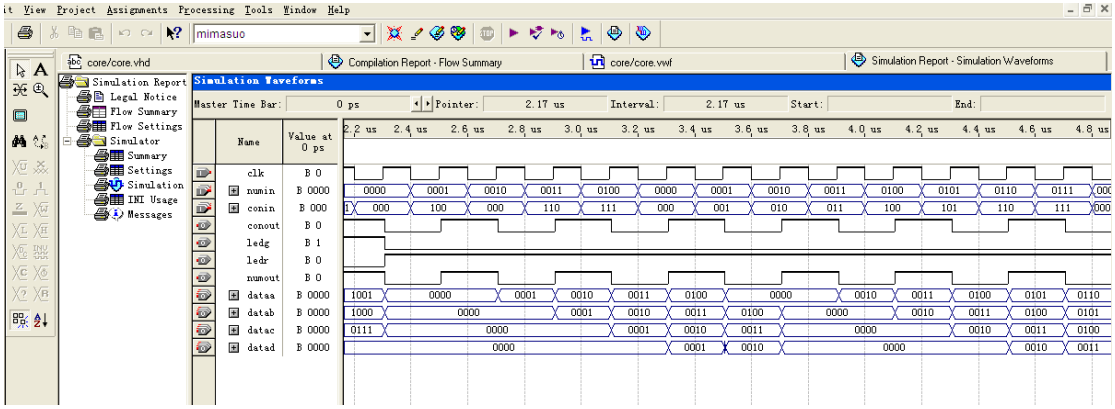

图 4.6　核心处理模块仿真图（二）

由图可知：若 conin 输入为“100”时（解锁时），当再次输入密码“1234”，并此时 ledg 输出为低电平（绿灯亮）。Dataa 输出密码“1234”，则表示此时密码锁已解锁。

（3）核心处理模块--core 符号文件：

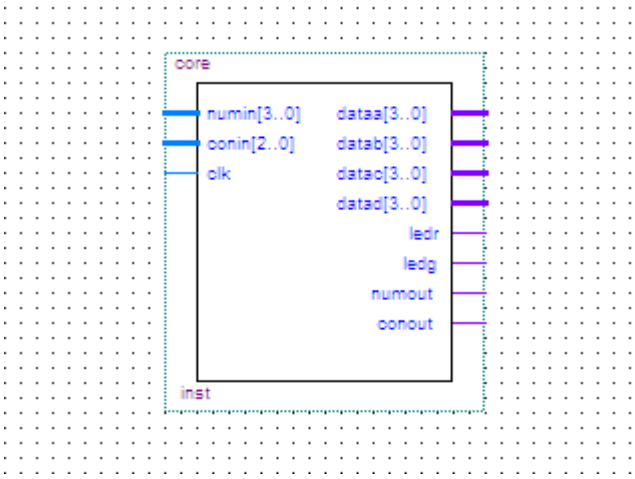

图 4.7 处理核心模块--core 符号文件

4.4 输出处理模块—allout

对处理核心模块—core 输出的数据进行刷新，使数码管及时显示刷新的数字。

（1）Allout--输出处理程序：

```
library ieee;
use ieee.std_logic_1164.all;
entity allout is
    port( dataa,datab,datac,datad:     in std_logic_vector(3 downto
0);
                clk:                   in std_logic;
                      dataout:         out  std_logic_vector(3  downto
                   0);
                          outsel:      out      std_logic_vector(1
                       downto 0));
end allout;
architecture one of allout is
    signal timer:    std_logic_vector(1 downto 0);
begin
process(clk)
```

```
begin
    if clk'event and clk='1' then--上升沿
            if timer="00" then
                    dataout<=dataa;
                    outsel<="00";
                    timer<="01";
            elsif timer="01" then
                    dataout<=datab;
                    outsel<="01";
                    timer<="10";
            elsif timer="10" then
                    dataout<=datac;
                    outsel<="10";
                    timer<="11";
            else
                    ataout<=datad;
                    outsel<="11";
                    timer<="00";
        end if;
    end if;
end process;
end one;
```

（2）输出处理模块仿真图：

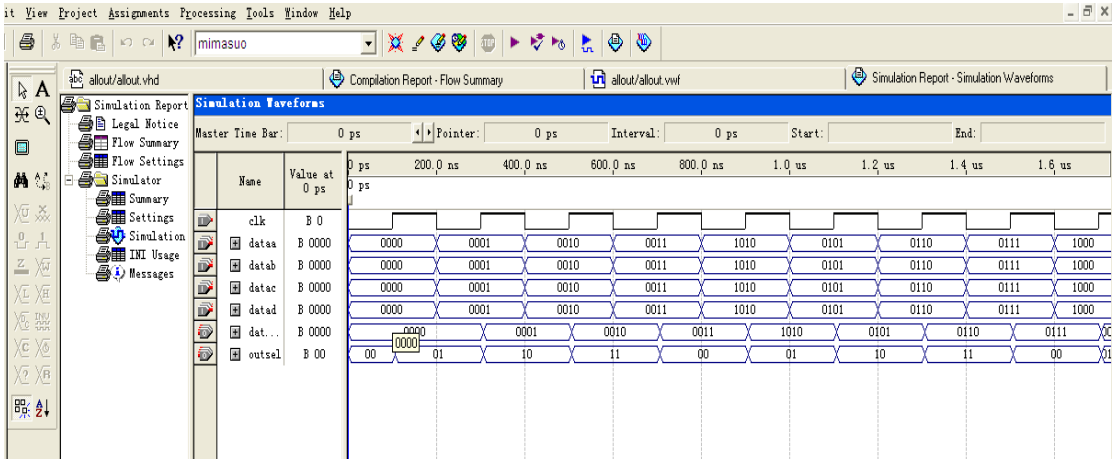

图 4.8 输出处理模块仿真图

当输出密码为“0123”时，经过输出处理模块的处理后，能刷新输出密码“0123”对应的二进制“0000”、“0001”、“0010”、“0011”，当密码改为“5678”时，该模块也能对应刷新出新密码对应二进制“0101”、“0110”、“0111”、“1000”。

（3）输出处理模块—allout 符号文件：

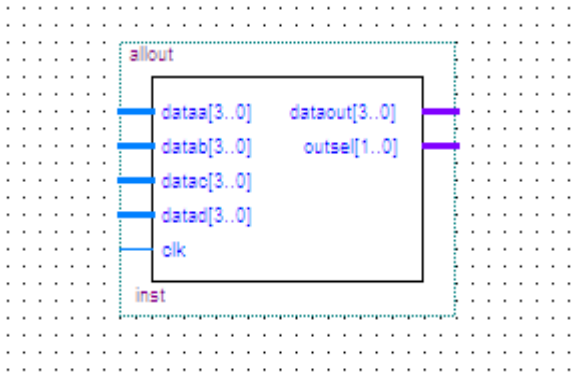

图 4.9 输出处理模块—allout 符号文件

4.5 七段译码器模块—dataout

把 4 位 2 进制数翻译成数码管代码，使输入的密码或修改的密码能及时显示在数码管上。

（1）Dataout--七段译码器模块程序：

```
library ieee;
use ieee.std_logic_1164.all;
entity dataout is
port( datain:IN std_logic_vector(3 downto 0);
```

```
dataout:OUT std_logic_vector(6 downto 0));
end dataout;
architecture one of dataout is
begin
process(datain)
begin
case datain is
      when "0000" => dataout<="1111110"; --数码管显示 0
      when "0001" => dataout<="0110000"; --数码管显示 1
      when "0010" => dataout<="1101101"; --数码管显示 2
      when "0011" => dataout<="1111001"; --数码管显示 3
      when "0100" => dataout<="0110011"; --数码管显示 4
      when "0101" => dataout<="1011011"; --数码管显示 5
      when "0110" => dataout<="1011111"; --数码管显示 6
      when "0111" => dataout<="1110000"; --数码管显示 7
      when "1000" => dataout<="1111111"; --数码管显示 8
      when "1001" => dataout<="1111011"; --数码管显示 9
      when others => dataout<="0000000"; --不显示
      end case;
end process;
end one;
```

（2）Dataout--七段译码器模块仿真图：

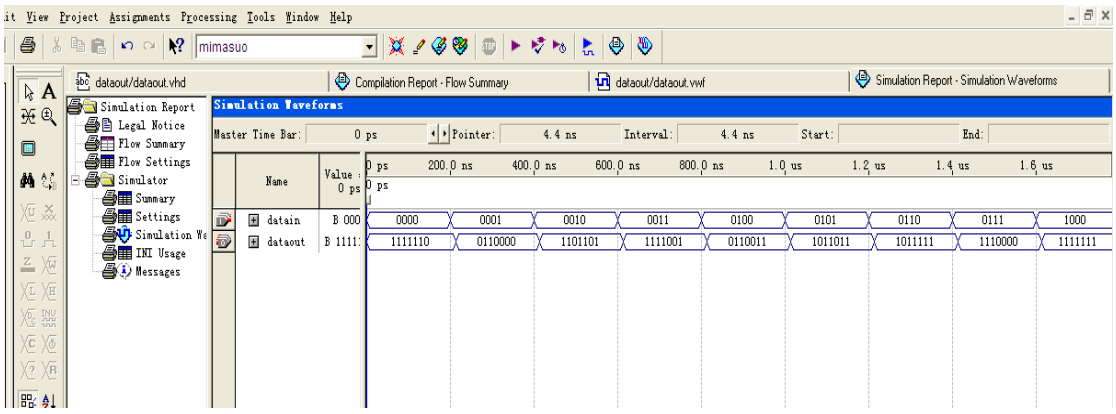

图 4.10 七段译码器模块仿真图

由图可知：当七段译码器的输入信号（datain）为“0000”、“0001”、“0010”、“0011”、“0100”、“0101”、“0110”、“0111”、“1000”、“1001”，输出信号（dataout）依次为:“1111110”、“ 0110000”、“ 1101101”、“ 1111001”、“ 0110011”、“ 1011011”、“ 1011111”、“ 1110000”、 “1111111”、“ 1111011”，当 conin 输入信号为其他时，译码输出为“0000000”。

表 4.3　七段译码器的输入及译码对照表

二进制译码输入	二进制译码输出	数码管显示数字
0000	1111110	0
0001	0110000	1
0010	1101101	2
0011	1111001	3
0100	0110011	4
0101	1011011	5
0110	1011111	6
0111	1110000	7
1000	1111111	8
1001	1111011	9
其他输入	0000000	无显示

（3）输出处理模块—allout 符号文件：

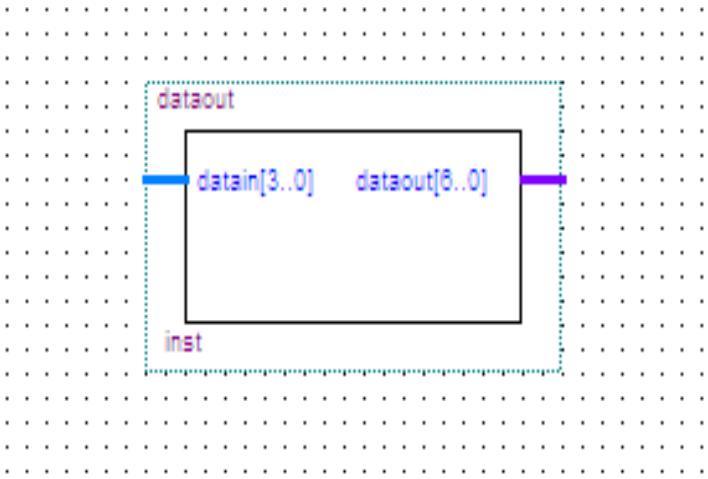

图 4.11 输出处理模块—allout 符号文件

5 仿真结果

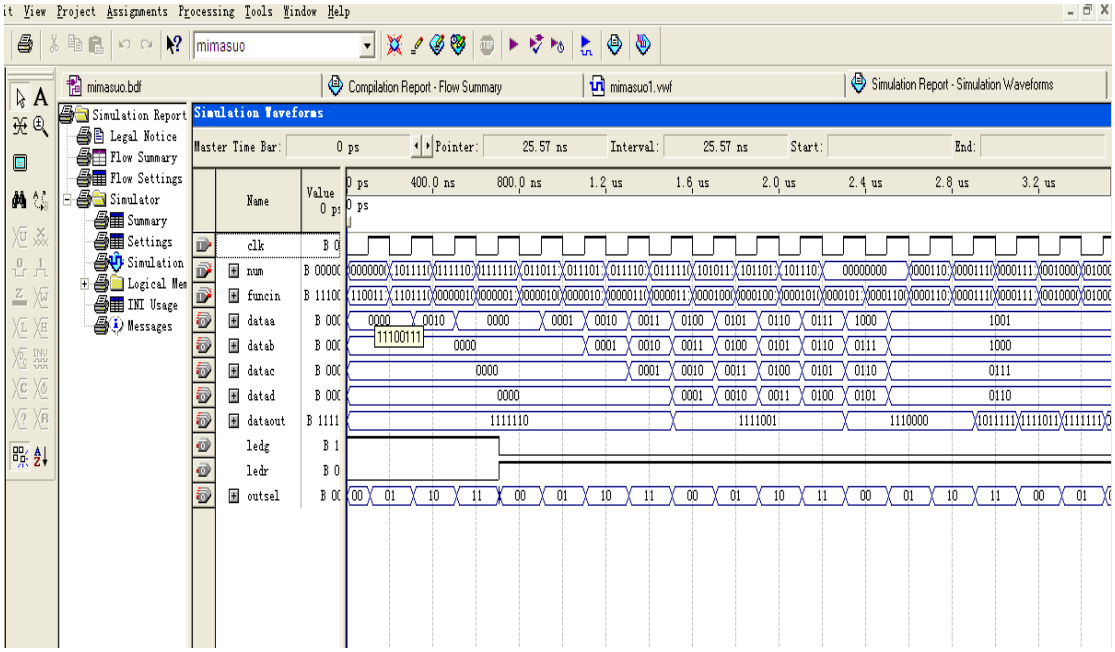

图5.1 系统仿真图（一）

由图可知，当功能按键的输入（funcin）为“11100111”（即按下功能按键“清除键”）时，系统输出（dataa，datab，datac，datad）均为“0000”，表示密码锁的密码已清除，数码管显示输出为“1111110”，即此时4个数码管均显示数字“0”。

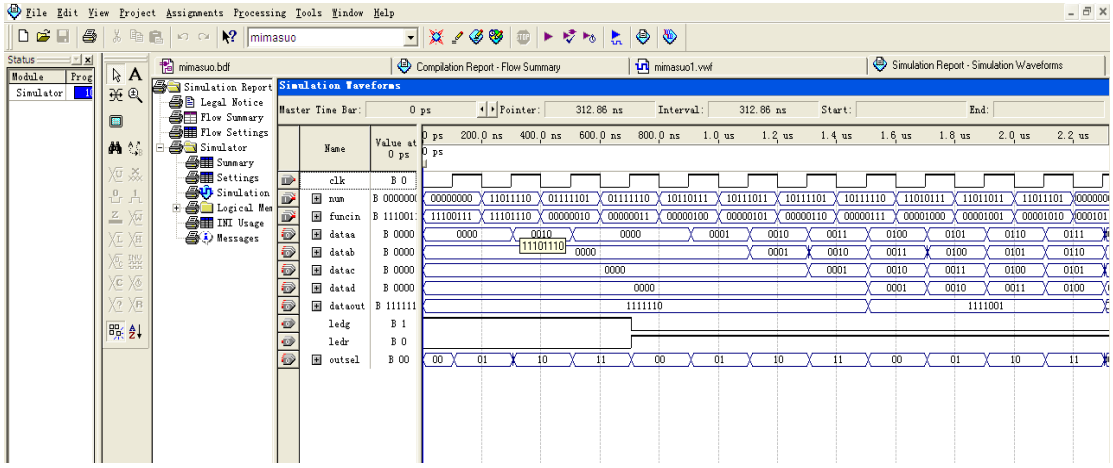

图5.2 系统仿真图（二）

由图可知，功能按键的输入（funcin）先为“11100111”（即按下功能按键“清除键”），后为“11101110”（即按下功能按键“改密键”）后，系统输出

（dataa，datab，datac，datad）为“0001”、“0010”、“0011”、“0100”，表示密码已修改为“1234”，且先ledg输出为低电平，表示为在开锁状态下修改密码，后ledr输出为高电平，表示密码已接受，数字密码锁已上锁。

6 设计总结

该密码锁的各项功能在仿真中通过了验证，密码位数也可根据实际需求来改变。实验表明，此密码锁运行稳定可靠，可满足日常生活基本要求，在实际应用中还可将 LED 灯显示替换为译码电路及七段数码管实现，进一步提高电路使用的人性化。

通过本次课程设计的学习，我深深的体会到 VHDL 设计的重要性和目的性所在。本次设计课不仅仅培养了我们的实际操作能力，也培养了我们灵活运用课本知识，理论联系实际，独立自主的进行设计的能力。它不仅仅是一个学习新知识新方法的好机会，同时也是对我所学知识的一次综合的检验和复习，使我明白了自己的缺陷所在，从而查漏补缺。

参考文献

[1] 黄仁欣.EDA 技术实用教程[M].北京:清华大学出版社，2006

[2] 张庆双主编.实用电子电路 200 例[M].北京:机械工业出版社， 2005

[3] 邹彦、庄严等编.EDA 技术与数字系统设计[M].北京:电子工业出版社，2008

[4] 华中科技大学电子技术课程组编.康华光主编.电子技术基础 [M].第五版.北 京:高等教育出版社，2006

致谢

在此次课程设计中，非常感谢老师给我们一次这样的展示机会以及鼓励，此外，对各位同学给予我的帮助，我表示最诚挚的谢意！