Harbin Institute of Technology

数字逻辑大作业

成员 *A:* 冯云龙
学号*:*1160300202
成员 *B:* 赖　昕
学号*:*1160300203

2017 年 7 月 5 日

摘要

大作业是在学完本门课程后，对所学知识的综合性考察. 知识覆盖面宽，实验所需时间长。要求学生灵活运用学过的计数器、触发器、译码电路等方面的知识，独立完成从设计、选片、连线、调试、排除故障到实现一个数字系统的全过程，详细书写项目报告。通过综合设计性实验，培养学生灵活运用所学知识解决比较复杂的实际问题的能力。

关键词：电子密码锁

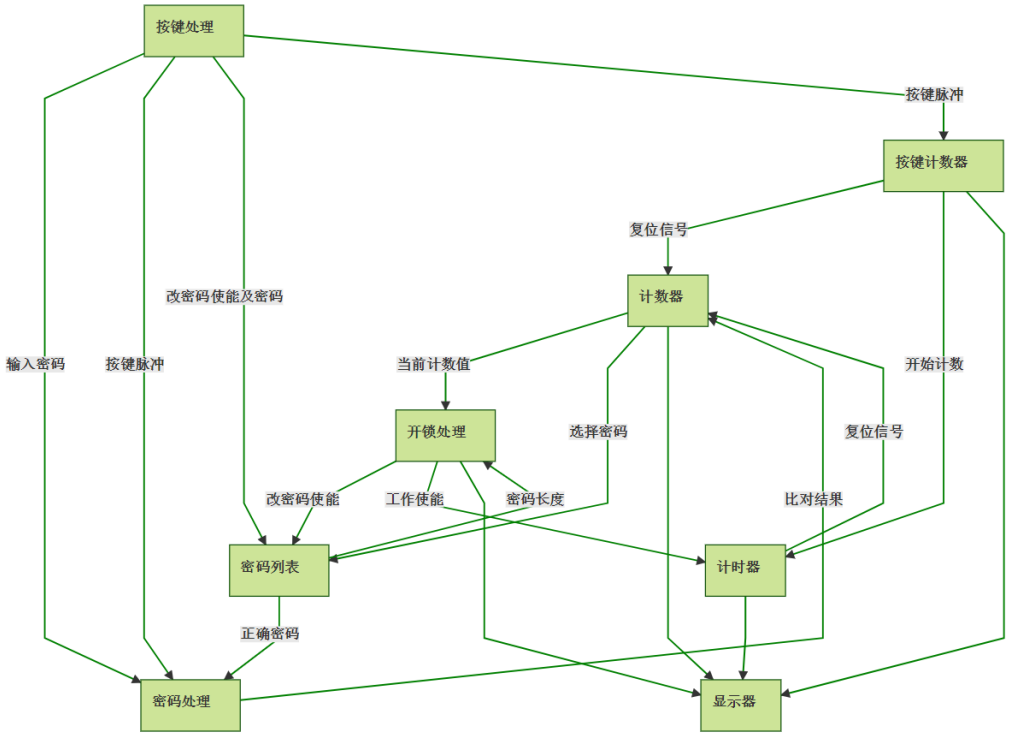
目录

第一部分　正文

第 1 章　设计目的及要求

1. 设计一个开锁密码至少为 4 位数字（或更多）的密码锁。
2. 当开锁按键输入代码等于所设密码时启动开锁控制电路，并且用绿灯亮、红灯灭表示开锁状态。
3. 从第一个按扭触动后的 5 秒内若未能将锁打开，则电路自动复位并发出报警信号，同时用绿灯灭、红灯亮表示关锁状态。
4. 密码锁中的密码可以修改，最高 7 位。
5. 记录按键次数并显示。
6. 输入密码倒计时。

第 2 章　工作原理、系统方框图



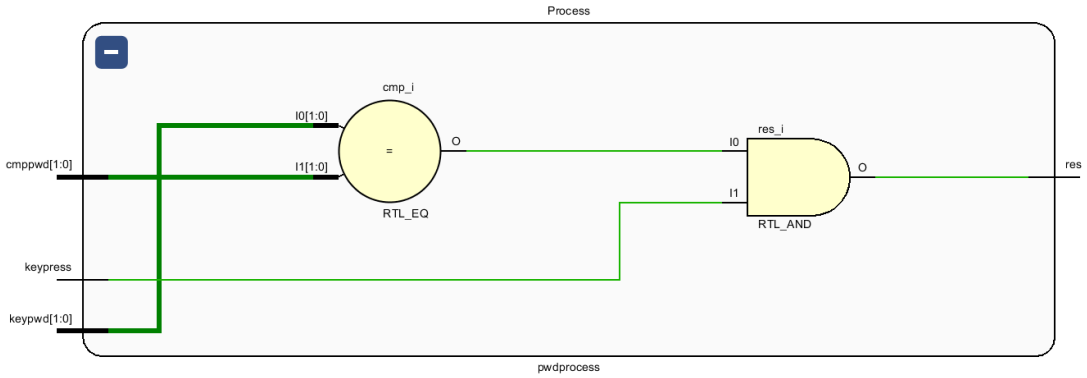
第 3 章　各部分选定方案及电路组成、相关器件

3.1　密码处理

3.1.1　选定方案

这部分我们使用的是一个等值比较器, 对密码进行比对, 当输入的密码与存储列表中的密码相等时，输出给计数器，使计数器计数。

3.1.2　电路组成



3.1.3　相关器件

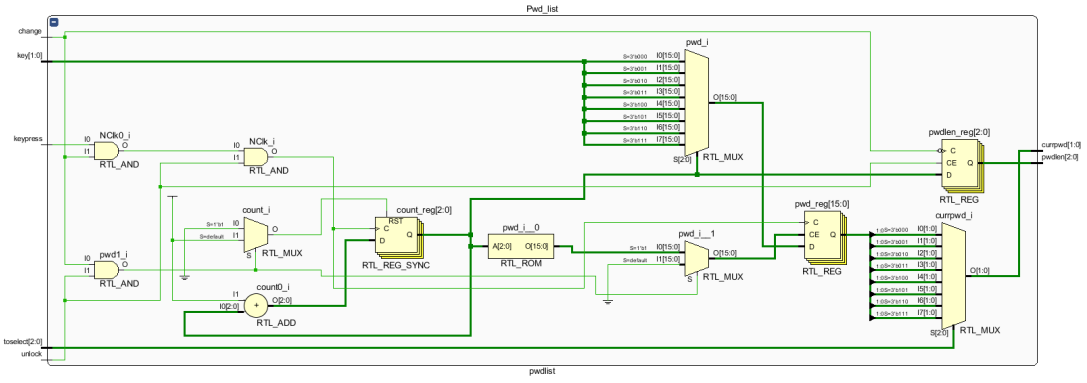
等值比较器，逻辑与门。

3.2　密码表

3.2.1　选定方案

首先我们使用了 16 位寄存器以储存密码和密码长度，使用数据选择器对数据进行选择, 在开锁情况下，给予修改信号可以对密码进行修改，当使用计数器对每一位进行修改，可以通过关闭修改信号以停止修改，当不再修改密码时，自动复位。

3.2.2　电路组成



3.2.3　相关器件

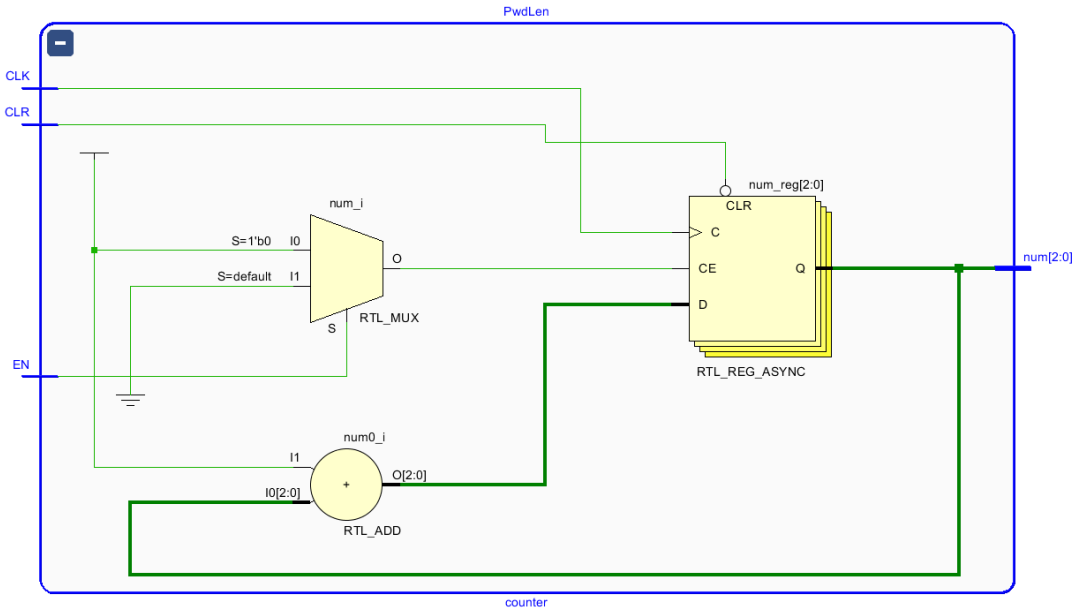
逻辑与门，逻辑非门，加法器，D 触发器，数据选择器。

3.3　密码计数器

3.3.1　选定方案

串行输入密码，每输对一位密码，计一次数，可以通过按键计数器与计时器所给的复位信号进行复位 (同步清零)。

3.3.2　电路组成



3.3.3　相关器件

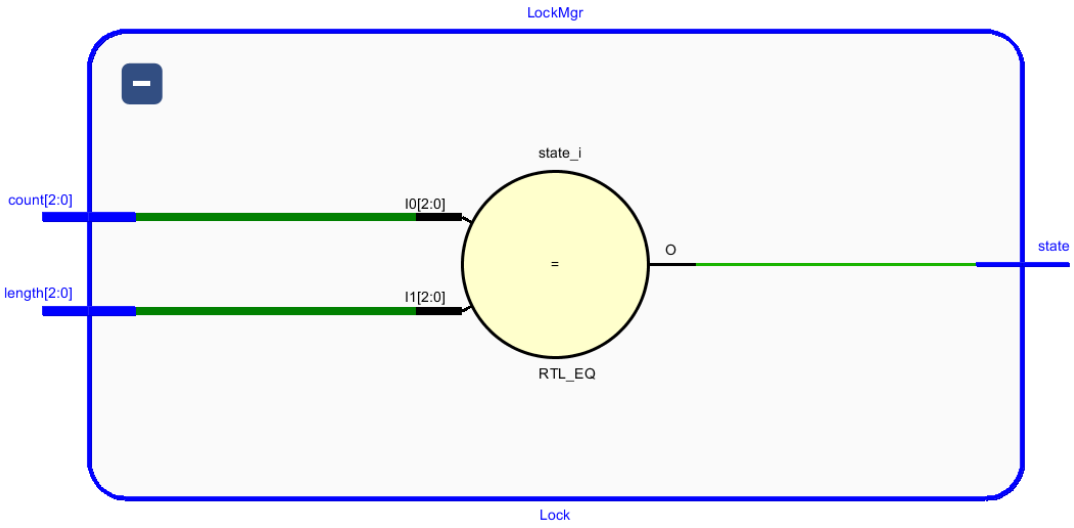
D 触发器，加法器，数据选择器。

3.4　开锁控制器

3.4.1　选定方案

实际上是一个等值比较器，将密码计数器的值与设置的密码长度进行比较，若相等，则输出为 1。

3.4.2　电路组成



3.4.3　相关器件

等值比较器。

3.5　核心功能

就是对开锁，解锁，修改密码的整合。

3.6　计时器

产生倒计时并可以输出倒计时归零信号。

3.6.1　选定方案

带有计数进位，清零端的计数器。

3.6.2　相关器件

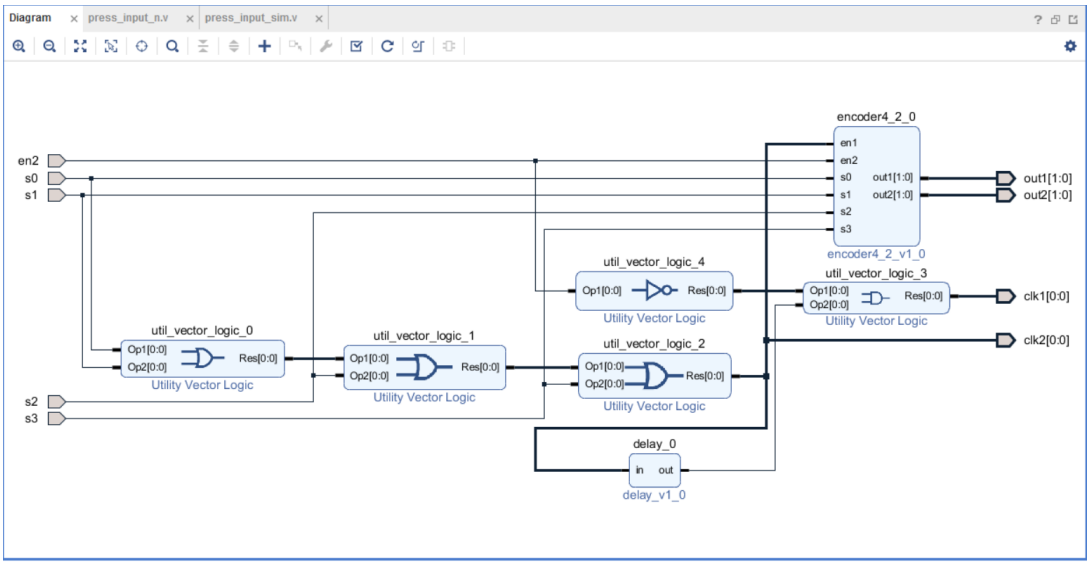
七段数码管。

3.7　按键处理

3.7.1　选定方案

以一个 4-2 编码器为基础进行改进。通过一个控制端对输出进行选择出口；并且设计了 2 个时钟输出端，其中一个给按键计数器，另外一个给密码处理；由于存在险象竞争，故给密码处理的时钟稍微比按键脉冲稍微延迟一些。延迟器可利用一个移位寄存器实现。

3.7.2　电路组成



3.7.3　相关器件

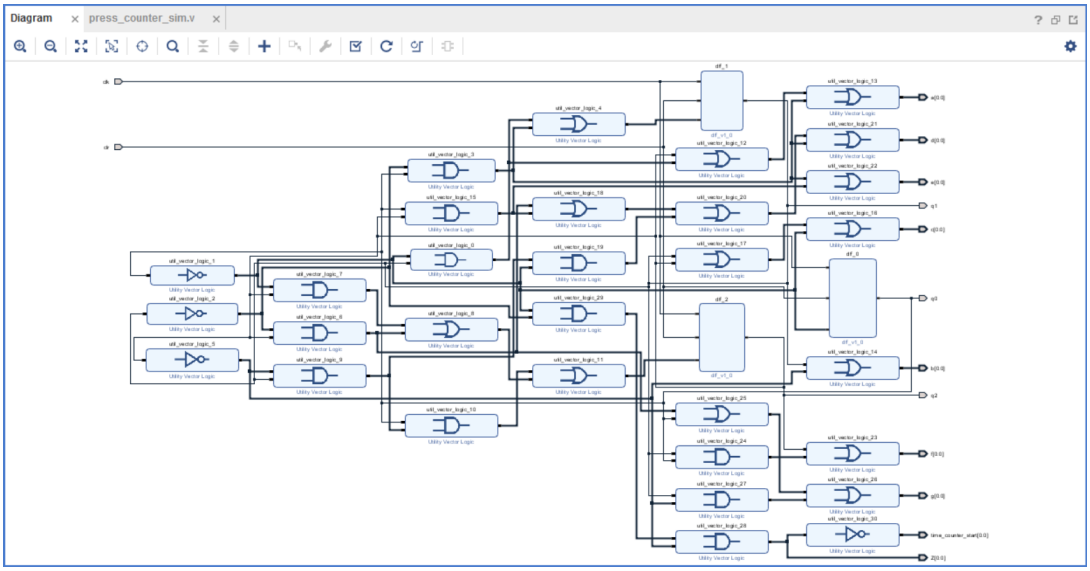
4-2 编码器，移位寄存器。

3.8　按键计数器

3.8.1　选定方案

以一个模 8 计数器为基础进行改进。计数器当前数值即当前输入密码的位数。并在此基础上设计了 2 个输出端。其中一个输出信号当按下第一个键后一直为 1，这个信号用来作为计时器的计时开始输入端；另一个当计数满 8 时为 1，这个信号用来作为给计数器的复位信号。

3.8.2　电路组成



3.8.3　相关器件

模 8 计数器，D 触发器。

3.9　显示器

3.9.1　选定方案

在计时器和按键计数器上有七段数码管输出端，制作好管脚约束即可显示。

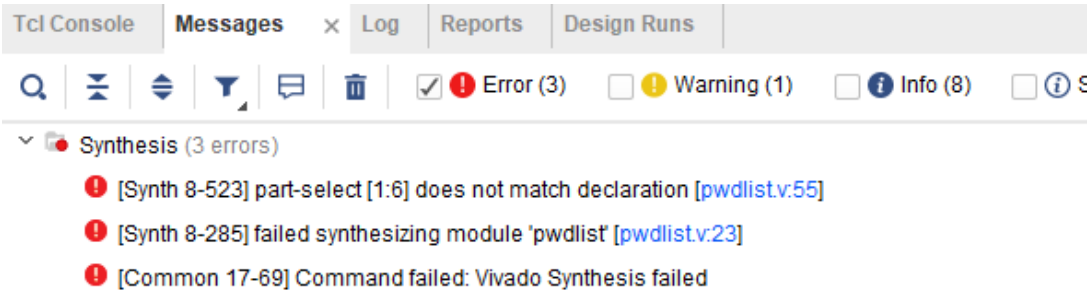
3.9.2　相关器件

七段数码管。

第 4 章　调试过程

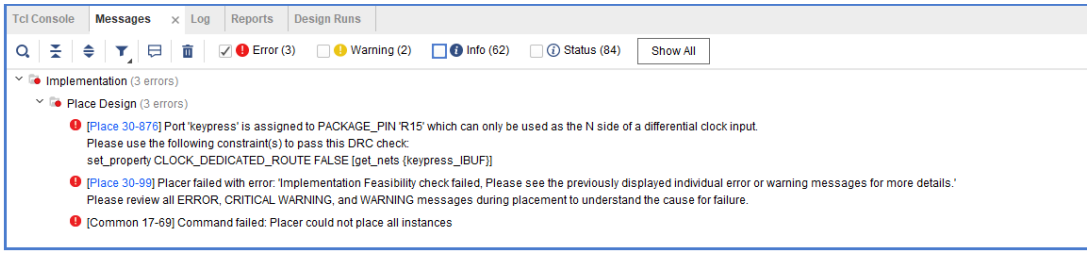
4.1　密码表中出现的问题

1. 变量未声明，经检查是变量名写错了，经过简单的修改就又可以正常工作了。



2. 这个错误很棘手，在百度时发现是由于在两个 *always* 块里修改了同一个 *reg* 变量，我对修改变量的部分做了综合，使用按键的上升沿进行触发，结果又出现了下面的错误。这也导致其无法自动复位。后来经过代码修改，使得在不修改密码时，密码长度计数值自动复位，又成功实现了自动复位。

3. 这个错误中提到 Pin$R15$ 不能当作 CLK 使用，多次研究后，将上升沿触发取消掉，成功。



4. 中间调试过程中发现改了一位密码就会改变开锁状态，导致密码修改失败，经过修改，使得密码长度变更发生在密码更改完毕后，解决问题。

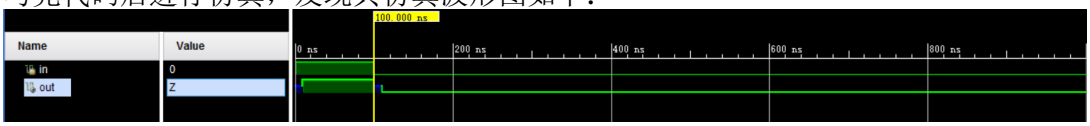
5. 仿真时完全正确，当下载综合到开发板上时，完全不正确。经研究还是密码表的问题，进行大修，重写，测试通过。

4.2　数码管

1. 数码管会有抖动，查找资料，修改代码，将显示与更改显示值的时间错开，完成数码管消隐。

4.3　按键处理模块问题

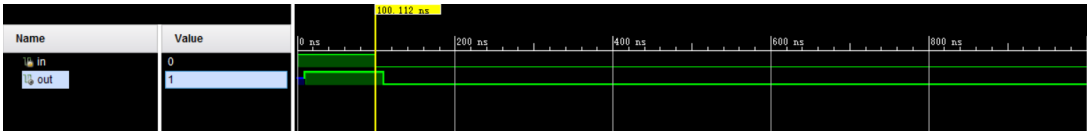
在按键处理模块中，需要编写延迟器使得按键脉冲能够为密码处理模块提供上升沿时钟信号。写完代码后进行仿真，发现其仿真波形图如下：



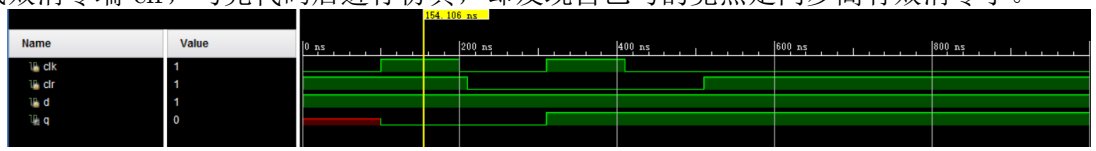
在 in 从高电平转为低电平之后的 10ns 内，out 并没有延续 in 之前高电平信号，而是变成高阻态，过了 10ns 才成为低电平状态，延迟器代码出现问题，发现其实只需要在最初始的 10ns 内为高阻态即可，需要初始化。按照下图修改之后，波形图为想要的图：

```
1  module delay(in, out);
2  input in;
3  output out;
4  reg out;
5  always@(*)
6  begin
7      out=1'bz;
8      #10
9      out=in;
10 end
11 endmodule
```

```
1  module delay(in, out);
2  input in;
3  output out;
4  reg out;
5  initial
6  begin
7      out=1'bz;
8  end;
9  always@(*)
10 begin
11     #10
12     out=in;
13 end
14 endmodule
```



在按键计数器模块中，欲用 D 触发器拼装成一个模 8 计数器。在编写 D 触发器时需要异步低效清零端 clr，写完代码后进行仿真，却发现自己写的竟然是同步高有效清零了。



经过检查代码，发现在 always 块的敏感列表除 posedge clk 外未加 clr，导致清零端同步时序。另外再将 clr 的有效值更改为 0 即可。

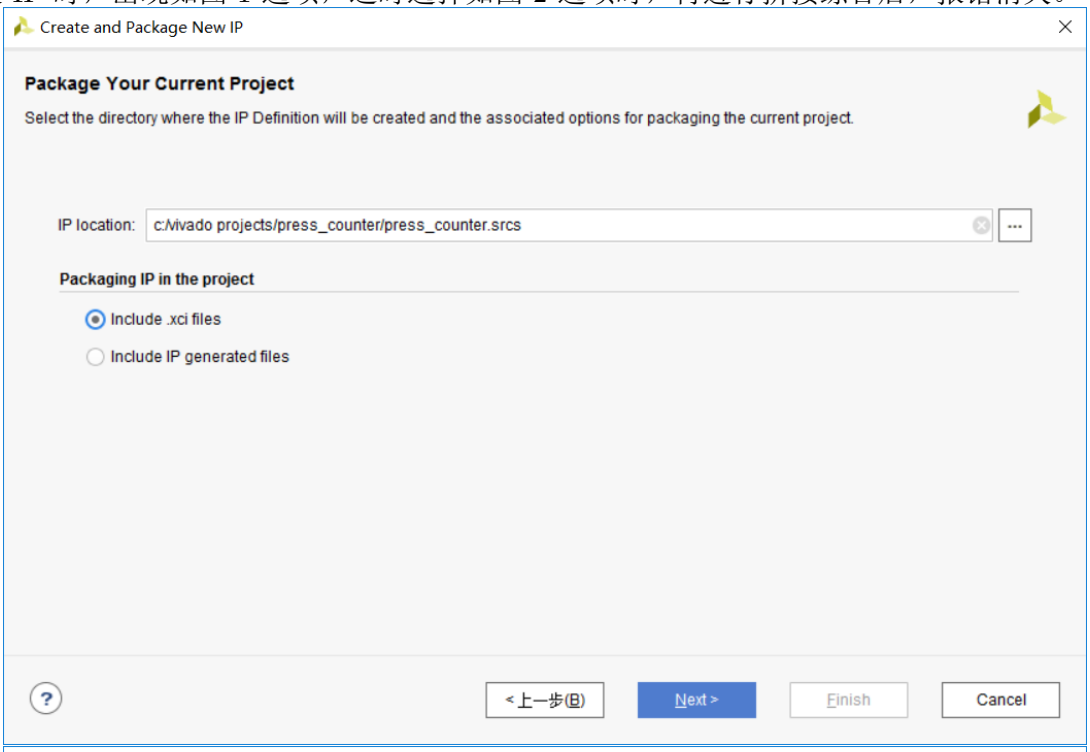
4.4　计时器模块问题

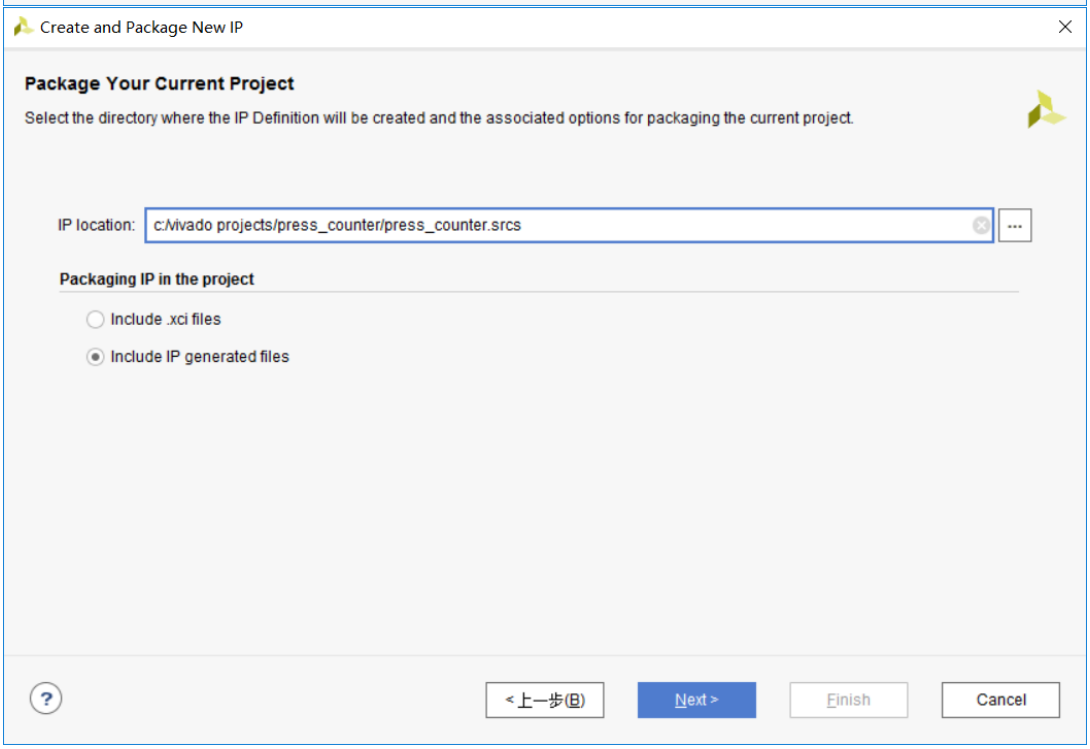
在计时器写好代码，仿真、综合、约束管脚之后，发现仿真正确，但是在开发板中调试却得不到结果。起初是不断给时钟上升沿，但就是没有反应，预测时钟端出现不可综合问题。将代码中的“=”改成“<=”后，再次进行仿真、综合，这次发现时钟端有效，但异步清零端无效。最后迫不得已把计时器模块的任务交给了队友冯云龙，得以解决。

```
10  always@(posedge clk or negedge clr)
11  begin
12      if(~clr)
13      begin
14          x<=0;
15          time_c_out<=0;
16      end
17      else if(en) x<=x+1;
18      case({x})
19              0:s<=7'b1011011;
20              1:s<=7'b0110011;
21              2:s<=7'b1111001;
22              3:s<=7'b1101101;
23              4:s<=7'b0110000;
24              5:time_c_out<=1;
25              endcase
26  end
```

4.5　组装出现的问题

在将各组员的工作组装时，拼接 block 完成之后综合时报错，封装 IP 时出现差错；再重新封装 IP 时，出现如图 1 选项，这时选择如图 2 选项时，再进行拼接综合后，报错消失。





第 5 章　设计结论

5.1　项目成果

通过团队的分工与合作，我们完成了带有三个附加功能的密码锁。可以修改真实密码数值以及位数，并且可以记录当前按键次数，以及输入密码时的时间倒计时进行保障安全。

5.2　项目团队

我们的团队一共有两名成员，地位是等同的，分工协作完成了我们的项目。期间我们进行了各种讨论以敲定最终的接口。

第 6 章　设计心得与总结

6.1　冯云龙

大作业的设计是曲折的，这次设计让我体会到设计接口的必要性，在设计整个项目的时候，我仿照着计算机界协同工作的一般形式，即先设计接口，再实现功能。很多时候一个大的工程都不是一个人完成的，而是许许多多的人分工合作完成的，如何协调任务，分配任务是一个很重要的问题。计算机领域的解决方案就是设计接口，这样就隐藏了具体实现，只关注功能的可用性。每一个模块在设计之初如果就完成了整合，那么每个模块只要按照文档实现，那么实现后一定也能很好的整合。这次的大作业，我们只有两个人，设计的重要性就已经体现出来，更大的工程项目肯定对于设计的要求会更高。团队的合作，设计是极为重要的。

6.2　赖昕

在这次大作业的完成过程中，我深切地感受到工程实际实现与理论之间的差距。团队从设计开始，一直到最终的实现，都是坎坷曲折的。最初，我本以为设计出各个模块分好工后就能够一帆风顺地完成作业。然而，实际上我们只花费了半天时间在理论的设计上，在具体操作和调试过程中花费的时间和精力远远大于理论设计上。当设计出具体用什么器件来实现某项功能并写好代码时，vivado 会报错；处理好之后写好激励代码进行仿真时，又会发现一系列的错误；在处理好之后进行综合，又会出现不可预知的错误；最后就算历经千辛万苦在 vivado 上仿真综合全都正确之后，连接到开发板上后又会遇到无效的问题。并且在各组员分工完成进行组装的阶段，也会出现各种各样意想不到的错误。另外，通过这次大作业，我对 vivado 更加熟悉了，使用 verilog 语言编写程序也更加熟练了，对 FPGA 编程的经验也更加丰富，对各个器件的功能具体实现也有了清晰的认识。此外，我更是接触到了延迟器、分频器等等之前不甚了解的器件的具体实现方法。其中延迟器可以运用移位寄存器来实现，分频器可以运用计数器来进行实现。最终，整个密码器的实现与整个团队的合作也是密不可分的。正是团队组员之间的合理分工与协作，才能让最终的组装得到满意的结果。也正是这种合理分工，才使得合作有了意义。

第 7 章　参考文献

1. verilog 语言实现任意分频

http://blog.csdn.net/ywhfdl/article/details/7641288

2. verilog 语言关于如何处理一个变量在多个 always 块里赋值不可综合问题

   http://bbs.csdn.net/topics/390752257

3. 如何解决 vivado 对时钟端有关问题的报错

   http://blog.csdn.net/cnkuaike123/article/details/48403067

4. FPGA 中的延时处理

   https://www.mianbaoban.cn/blog/post/64632

5. 实现信号延时的方法

   http://blog.csdn.net/xiangyuqxq/article/details/7255967

6. verilog 入门教程

7. Tobias Oetiker,Hubert Partl, Irene Hyna and Elisabeth Schlegl,lshort-zh-cn

8. 单片机数码管显示消隐

   http://www.eeworld.com.cn/mcu/article_2016122332568.html

第二部分　附录

A　总体器件表及相关器件的功能表、管脚分布

A.1　密码长度计数器

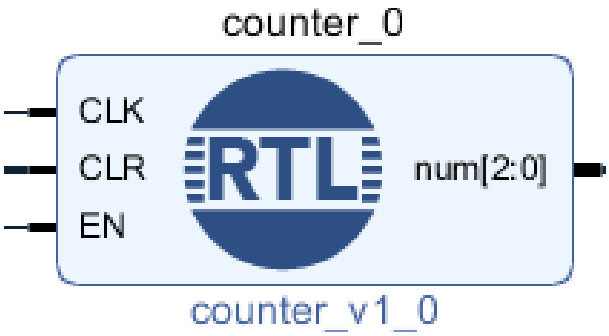


图 1: 密码长度计数器

管脚名	功能
CLK	上升沿触发计数
CLR	异步清零端
EN	使能端
num	当前密码输对的个数

图 2: 功能表

A.2　密码处理

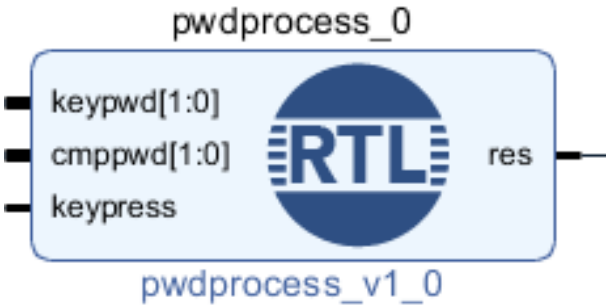


图 3: 密码处理

管脚名	功能
keypwd	按键输入密码
cmppwd	正确密码输入端
keypress	按键脉冲，有按键
res	比对结果

图 4: 功能表

A.3　开锁控制器

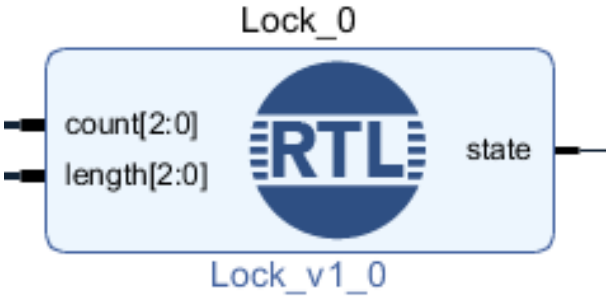


图 5: 开锁控制器

管脚名	功能
count	当前输入正确密码的位数
length	正确密码总长度
state	当前开锁状态

图 6: 功能表

A.4 密码表

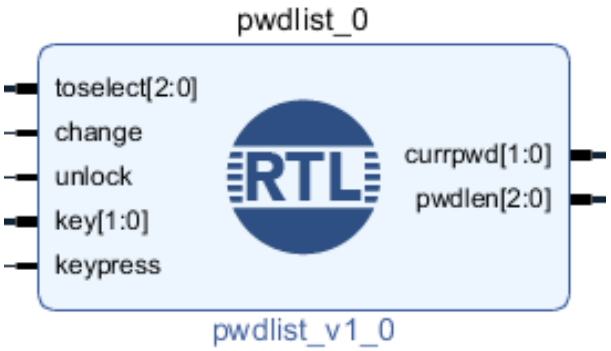


图 7: 密码表

管脚名	功能
toseclect	选择的密码
change	是否修改密码
unlock	是否已开锁
key	按键值
keypress	按键脉冲
currpwd	当前正确密码
pwdlen	密码长度

图 8: 功能表

A.5 核心综合

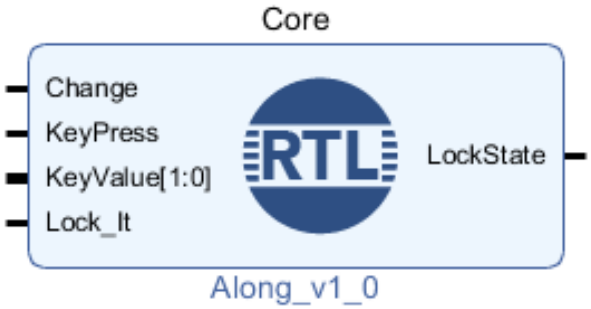


图 9: 核心综合

管脚名	功能
Change	修改密码使能
KeyPress	给按键计数器的按键脉冲
KeyValue	按键值
LockIt	上锁信号
LockState	开锁状态

图 10: 功能表

A.6 按键部分

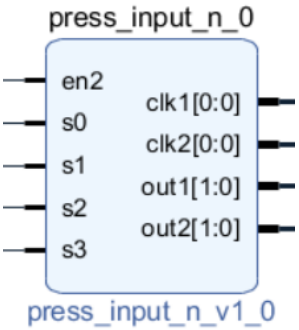


图 11: 按键部分

管脚名	功能
clk1	脉冲信号（给密码处理）
clk2	按键脉冲（给按键计数器）
en2	修改密码输出使能
out1	输入密码输出（给密码处理）
out2	修改密码输出
s0 s3	按键输入

图 12: 功能表

A.7 按键计数器部分

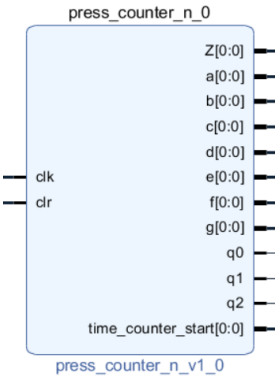
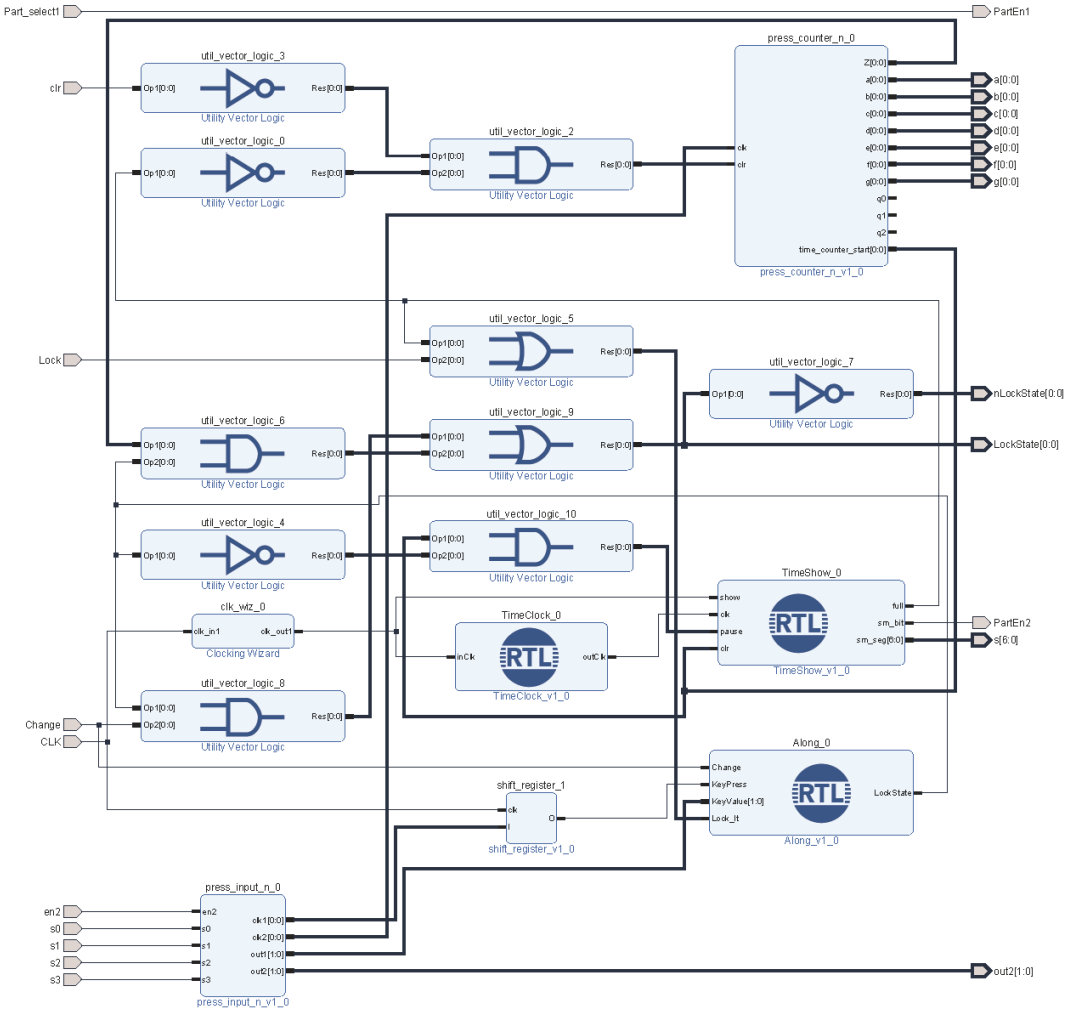


图 13: 按键计数器部分

管脚名	功能
clk	按键脉冲（来自按键处理）
clr	异步清零端（低有效）
Z	满八进一的输出（给计数器提供的复位信号）
a g	七段数码管输出
q0 q2	当前计数的三位 2 进制表示数
start	开始计数信号（给计时器提供）

图 14: 功能表

B 总体设计图



C 仿真结果

C.1 A 部分仿真

仿真激励代码

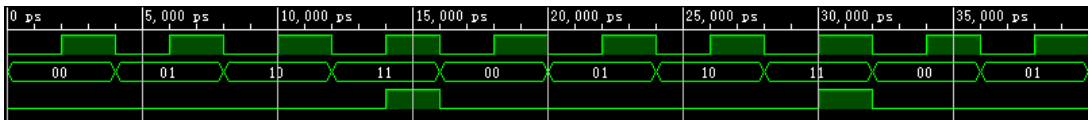
```
module Test();
reg Press;
reg [1:0]key_in;
wire out;

pwdprocess u0(key_in,3,Press,out);  //预设密码为3

initial begin
Press=0;
key_in=0;
end

always #2 Press=~Press;
always #4 key_in = key_in +1;

endmodule
```



预设的密码为 3，经过仿真，输入为 3 是才能获得正确的脉冲，否则无效。

开锁控制器

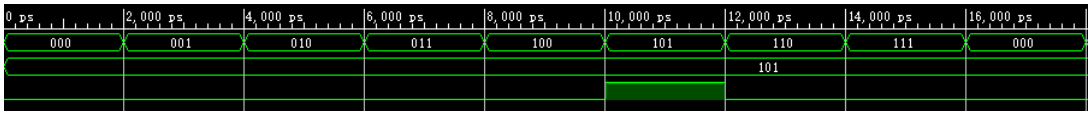
```
module Test();
reg [2:0]count;
reg [2:0]length;
wire state;

Lock u0(count,length,state);

initial begin
count=0;
length=5;
end

always #2 count=count+1;
```

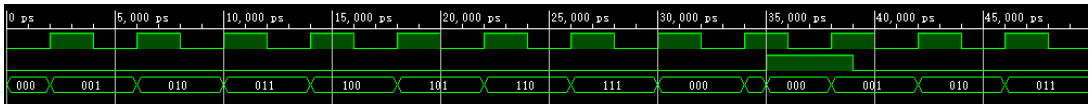
```
14
15 endmodule
```



预设的密码长度为 5，经过仿真，输入的正确密码为 5 是才能获得开锁状态。

密码计数器

```
1  module Test();
2  reg clk;
3  reg clr;
4  wire [2:0]state;
5
6  counter u0(clk,clr,state);
7
8  initial begin
9  clk=0;
10 clr=0;
11 end
12
13 always #2 clk=~clk;
14
15 always begin
16 #35 clr = 1;
17 #4 clr = 0;end
18
19 endmodule
```



获得一次 clk 记一次数，上升沿有效，异步清零。

密码表

```
1  module Test();
2  reg [2:0]select;          //选择下一位的正确密码
3  wire [1:0]beSelect;       //被选择出来的密码
4  wire [2:0]len;            //存储的密码长度
5
6  reg change;               //是否修改密码
7  reg unlock;               //当前开锁状态
```

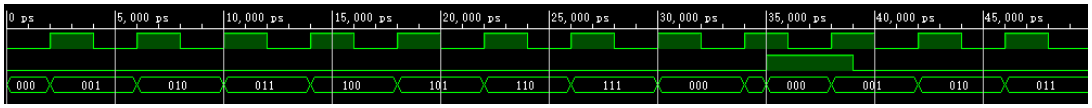
```
reg [1:0]key;              //修改密码所给的值
reg keypress;              //按键脉冲

pwdlist u0(select,beSelect,len,change,unlock,key,keypress);

initial begin
select = 0;
change = 1;
unlock = 1;
key = 0;
keypress = 0;
end

always #2 select=select+1;
always begin
#2 keypress = ~keypress;
key = key + 1;
end

endmodule
```



可以修改密码，最多 7 位，初始密码为 0000。

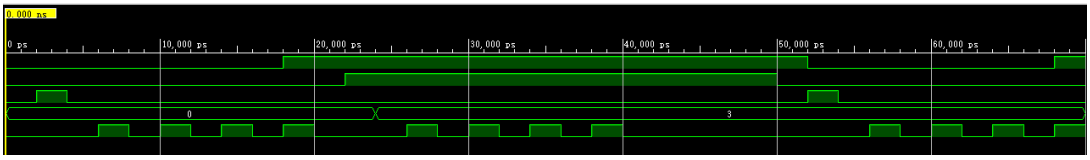
核心功能仿真

```
module Test();
wire LockState;
reg Change;                //是否修改密码
reg Lock_It;               //当前开锁状态
reg [1:0]KeyValue;              //修改密码所给的值
reg Keypress;              //按键脉冲

Along u0(
    Change,
    Keypress,
    KeyValue,
    Lock_It,
    LockState
    );
```

```
15 initial begin
16     Change=0;
17     Lock_It=0;
18     KeyValue=0;
19     Keypress=0;
20     //上锁
21     #2 Lock_It = 1;
22     #2 Lock_It = 0;
23     //开锁
24     #2 Keypress = ~Keypress;
25     #2 Keypress = ~Keypress;
26     #2 Keypress = ~Keypress;
27     #2 Keypress = ~Keypress;
28     #2 Keypress = ~Keypress;
29     #2 Keypress = ~Keypress;
30     #2 Keypress = ~Keypress;
31     #2 Keypress = ~Keypress;
32     //修改密码
33     #2 Change = 1;
34     #2 KeyValue = 3;
35     #2 Keypress = ~Keypress;
36     #2 Keypress = ~Keypress;
37     #2 Keypress = ~Keypress;
38     #2 Keypress = ~Keypress;
39     #2 Keypress = ~Keypress;
40     #2 Keypress = ~Keypress;
41     #2 Keypress = ~Keypress;
42     #2 Keypress = ~Keypress;
43     //密码修改完成，上锁
44     #10 Change = 0;
45     #2 Lock_It = 1;
46     #2 Lock_It = 0;
47     //使用新密码开锁
48     #2 Keypress = ~Keypress;
49     #2 Keypress = ~Keypress;
50     #2 Keypress = ~Keypress;
51     #2 Keypress = ~Keypress;
52     #2 Keypress = ~Keypress;
53     #2 Keypress = ~Keypress;
```

```
54        #2 Keypress = ~Keypress;
55        #2 $finish;
56  end
57  endmodule
```

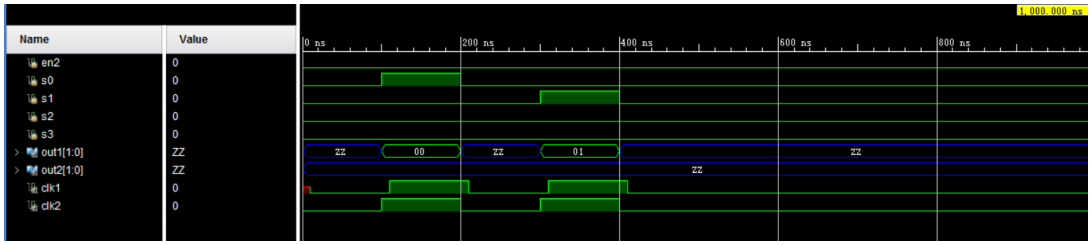


运行效果良好，开锁上锁，修改密码都做的相当不错。

C.2 B 部分仿真

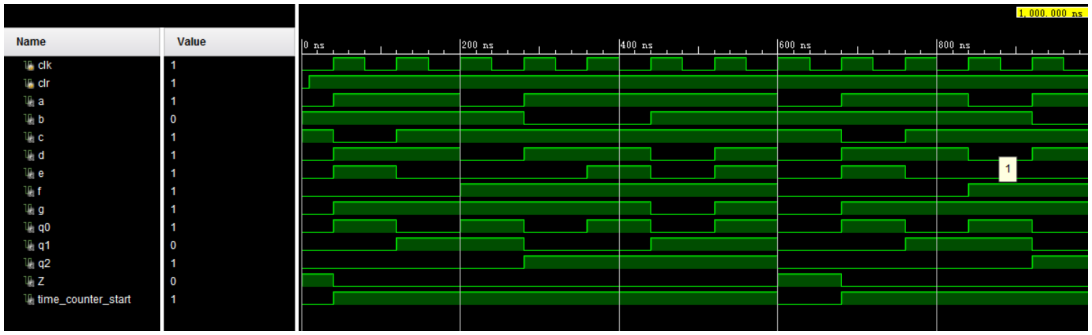
按键处理

```
1   module press_input_sim;
2   reg en2,s0,s1,s2,s3;
3   wire [1:0]out1,out2;
4   wire clk1,clk2;
5   press_input_n a(clk1,clk2,en2,out1,out2,s0,s1,s2,s3);
6   initial
7   begin
8         en2=0;
9         {s3,s2,s1,s0}=4'b0000;
10        #100
11        {s3,s2,s1,s0}=4'b0001;
12        #100
13        {s3,s2,s1,s0}=4'b0000;
14        #100
15        {s3,s2,s1,s0}=4'b0010;
16        #100
17        {s3,s2,s1,s0}=4'b0000;
18  end
19  endmodule
```



按键计数器

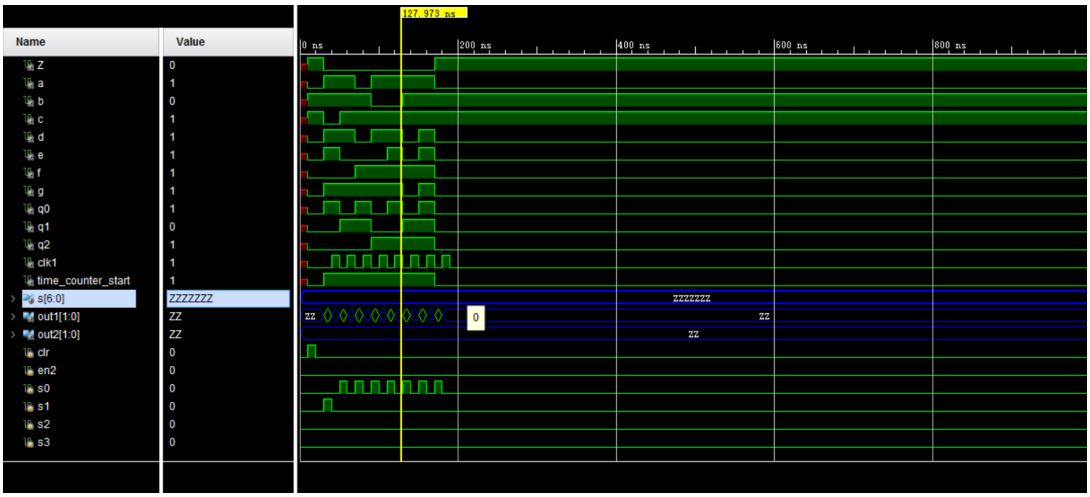
```
module press_counter_sim;
reg clk,clr;
wire a,b,c,d,e,f,g,q0,q1,q2,Z,time_counter_start;
press_counter_n my(Z,a,b,c,clk,clr,d,e,f,g,q0,q1,q2,time_counter_start);
initial
begin
      clr=0;
      clk=0;
      #10 clr=1;
end
always #40 clk=~clk;
endmodule
```



整体仿真

```
module lai_sim;
wire Z,a,b,c,d,e,f,g,q0,q1,q2,clk1,time_counter_start;
wire [6:0]s;
wire [1:0]out1,out2;
reg clr,en2,s0,s1,s2,s3;
lai lx
  (Z,a,b,c,clk1,clr,d,e,
    en2,f,g,out1,out2,q0,q1,
    q2,s0,s1,s2,s3,time_counter_start);
    initial
    begin
        clr=0;
        en2=0;
        s0=0;
        s1=0;
        s2=0;
        s3=0;
        #10 clr=1;
```

```
19            #10 clr=0;
20            #10 s1=1;
21            #10 s1=0;//1
22            #10 s0=1;
23            #10 s0=0;//0
24            #10 s0=1;
25            #10 s0=0;//0
26            #10 s0=1;
27            #10 s0=0;//0
28            #10 s0=1;
29            #10 s0=0;//0
30            #10 s0=1;
31            #10 s0=0;//0
32            #10 s0=1;
33            #10 s0=0;//0
34            #10 s0=1;
35            #10 s0=0;//0
36        end
37    endmodule
```



D 工作说明

冯云龙 密码处理，密码列表，密码计数器，开锁控制器，计时器，实验报告。

D.1 密码处理

```
1  module pwdprocess(
2      input [1:0] keypwd,          //按键输入密码
3      input [1:0] cmppwd,          //正确密码
```

```
    input keypress,                    //按键脉冲，上升沿触发
    output res                         //输出计数脉冲
    );
    wire cmp;
    assign cmp=keypwd==cmppwd;
    assign res = cmp && keypress;
endmodule
```

D.2 密码表

```
module pwdlist(
    input [2:0]toselect,           //计数选择
    output reg[1:0]currpwd,        //输出密码
    output reg[2:0]pwdlen,         //密码长度
    input change,                  //是否修改密码
    input unlock,                  //是否已解锁
    input [1:0]key,                //数据输入
    input keypress                 //按键脉冲
    );
    reg [15:0]pwd;                 //每两位是一组密码.
    reg [2:0]count;

    wire NClk;
    assign NClk = keypress && change && unlock;

    initial begin
    count = 0;
    pwd = 0;                       //初始密码
    pwdlen = 4;                    //初始密码长度
    end

    always @(toselect,pwd)         //密码选择部分
    begin
    case(toselect)
    3'b000 : currpwd = pwd[1:0];
    3'b001 : currpwd = pwd[3:2];
    3'b010 : currpwd = pwd[5:4];
    3'b011 : currpwd = pwd[7:6];
    3'b100 : currpwd = pwd[9:8];
    3'b101 : currpwd = pwd[11:10];
```

```
31      3'b110 : currpwd = pwd[13:12];
32      3'b111 : currpwd = pwd[15:14];
33      endcase
34      end
35
36      always @(posedge NClk)
37      begin
38      if(change && unlock) begin//已经解锁并且给了密码修改使能
39      case(count)
40      3'b000 : pwd[1:0] = key;
41      3'b001 : pwd[3:2] = key;
42      3'b010 : pwd[5:4] = key;
43      3'b011 : pwd[7:6] = key;
44      3'b100 : pwd[9:8] = key;
45      3'b101 : pwd[11:10] = key;
46      3'b110 : pwd[13:12] = key;
47      3'b111 : pwd[15:14] = key;
48      endcase
49      count = count + 1;
50      end
51      else count = 0;
52      end
53
54      always @(negedge change)begin          //密码长度修改
55      if(unlock) pwdlen = count;
56      end
57
58  endmodule
```

D.3 密码计数器

```
1   module counter(
2       input CLK,                  //计数时钟
3       input CLR,                  //复位
4       output reg[2:0]num              //当前计数
5       );
6       initial num = 0;
7
8       always @(posedge CLK) begin
9       if(CLR)
```

```
10          num = 0;
11        else
12          num = num + 1;
13        end
14  endmodule
```

D.4 开锁控制器

```
1   module Lock(
2       input [2:0]count,                   //当前输入正确的密码个数
3       input [2:0]length,                  //密码长度
4       output state                    //当前解锁状态，1为开锁
5       );
6       assign state = (count == length);
7   endmodule
```

D.5 核心功能封装

```
1   module Along(
2       input Change,                       //密码修改使能
3       input KeyPress,                     //按键脉冲
4       input [1:0]KeyValue,                //按键值
5       input Lock_It,                      //上锁信号
6       output LockState                    //开锁状态
7       );
8
9         wire toLock;
10        assign toLock=~Lock_It;
11
12        wire [2:0]currlen;
13        wire [1:0]currpwd;
14        wire [2:0]pwdlen;
15        wire isRight;
16
17        Lock LockMgr
18                (.count(currlen),
19                 .length(pwdlen),
20                 .state(LockState));
21          counter PwdLen
22                (.CLK(isRight),
```

```
23                  .CLR(toLock),
24                  .EN(LockState),
25                  .num(currlen));
26          pwdlist Pwd_list
27                  (.change(Change),
28                  .currpwd(currpwd),
29                  .key(KeyValue),
30                  .keypress(KeyPress),
31                  .pwdlen(pwdlen),
32                  .toselect(currlen),
33                  .unlock(LockState));
34          pwdprocess Process
35                  (.cmppwd(currpwd),
36                  .keypress(KeyPress),
37                  .keypwd(KeyValue),
38                  .res(isRight));
39  endmodule
```

D.6 计时器

```
1  module TimeClock(
2      input inClk,
3      output outClk
4      );
5      reg [31:0]timeclk;
6
7      assign outClk = timeclk[21];
8      initial begin
9      timeclk = 0;
10     end
11
12     always@(posedge inClk)begin
13     timeclk = timeclk +1;
14     end
15 endmodule
```

```
1  module TimeShow(
2      input show,                    //显示切换信号
3      input clk,                     //时钟信号
4      input pause,                   //暂停计时
```

```
5       input clr,                          //清零段
6       output full,                        //进位信号
7       output reg sm_bit,                  //片选信号
8       output reg[6:0]sm_seg               //显示数码管
9       );
10
11      reg [3:0]timesec0;
12      assign full = timesec0 == 4;
13
14      initial begin
15      sm_bit=1;
16      sm_seg=1;
17      timesec0=0;
18      end
19
20      always @(posedge clk or negedge clr) begin
21      if(~clr)
22        timesec0 = 0;
23      else
24      if(pause)begin
25          if(timesec0==4)
26          timesec0=0;
27          else
28          timesec0=timesec0+1;
29          end
30      end
31
32      always@(posedge show)
33      begin
34      case(timesec0)
35                  0:sm_seg= 7'b0110011;                   //显示4
36                  1:sm_seg= 7'b1111001;                    //显示3
37                  2:sm_seg= 7'b1101101;                   //显示2
38                  3:sm_seg= 7'b0110000;                   //显示1
39                  4:sm_seg= 7'b1111110;                   //显示0
40                  default:
41                  sm_seg= 7'b0000000;                     //不显示
42      endcase
43      end
```

```
44 endmodule
```

赖昕 按键计数器，按键译码处理器，显示器，最终组装。

D.7 D 触发器

```
1  module dff(clk,clr,d,q);//d触发器
2  input clk,clr,d;
3  //clk: 时钟沿（上升沿触发）
4  //clr: 异步清零端（低电平有效）
5  //d: 触发器输入值端
6  output q;//q: 触发器输出端
7  reg q;
8  always@(posedge clk or negedge clr)//clk上升沿触发，clr异步
9  begin
10 if(~clr) q<=0;//clr低电平有效
11 else if(clk) q<=d;//来时钟沿时，q=d;
12 end
13 endmodule
```

D.8 4-2 编码器

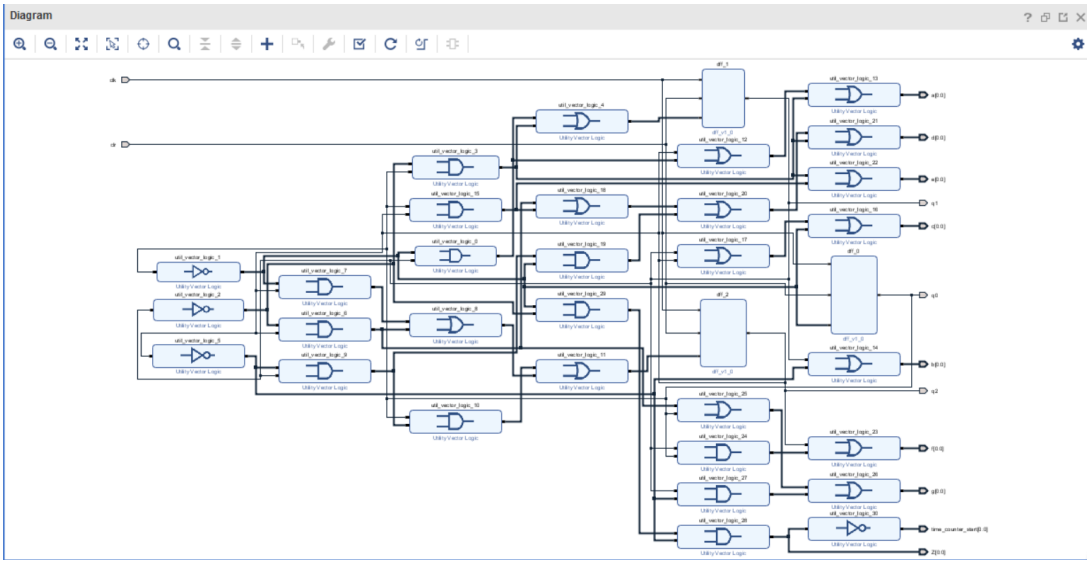
```
1  module encoder4_2(en1,en2,s0,s1,s2,s3,out1,out2);
2  input en1,en2;//en1:译码器使能端；en2:修改密码使能端
3  input s0,s1,s2,s3;//s0~s3;按键输入
4  output [1:0]out1;//out1: 用于输入密码的输出
5  output [1:0]out2;//out2: 用于修改密码的输出
6  reg [1:0]out1;
7  reg [1:0]out2;
8  always@(*)
9  begin
10     if(~en1)//如果en1为低电位，则输出均为高阻态
11     begin
12         out1=2'bzz;
13         out2=2'bzz;
14     end
15     else if(~en2)
16     //如果en1为高电位且en2为低电位，
17     //则输入密码out1输出为编码输出，修改密码out2输出为高阻态
18     begin
```

```
19          out2=2'bzz;
20          case({s3,s2,s1,s0})
21              4'b0001:out1=2'b00;
22              4'b0010:out1=2'b01;
23              4'b0100:out1=2'b10;
24              4'b1000:out1=2'b11;
25          endcase
26      end
27      else
28      //如果en1为高电位且en2为高电位,
29      //则修改密码out2输出为编码输出, 输入密码out1输出为高阻态
30      begin
31          out1=2'bzz;
32          case({s3,s2,s1,s0})
33              4'b0001:out2=2'b00;
34              4'b0010:out2=2'b01;
35              4'b0100:out2=2'b10;
36              4'b1000:out2=2'b11;
37          endcase
38      end
39  end
40  endmodule
```

D.9 移位寄存器

```
1  module shift_register(clk,I,O);
2  input I,clk;
3  output O;
4  reg [3:0]b;
5  wire O;
6  always@(posedge clk)//当时钟上升沿来到时, 寄存器移位。
7  begin
8      b<={b[2:0],I};
9  end
10 assign O=b[3];
11 endmodule
```

D.10 按键处理



D.11 按键计数器