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 6 月 27 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

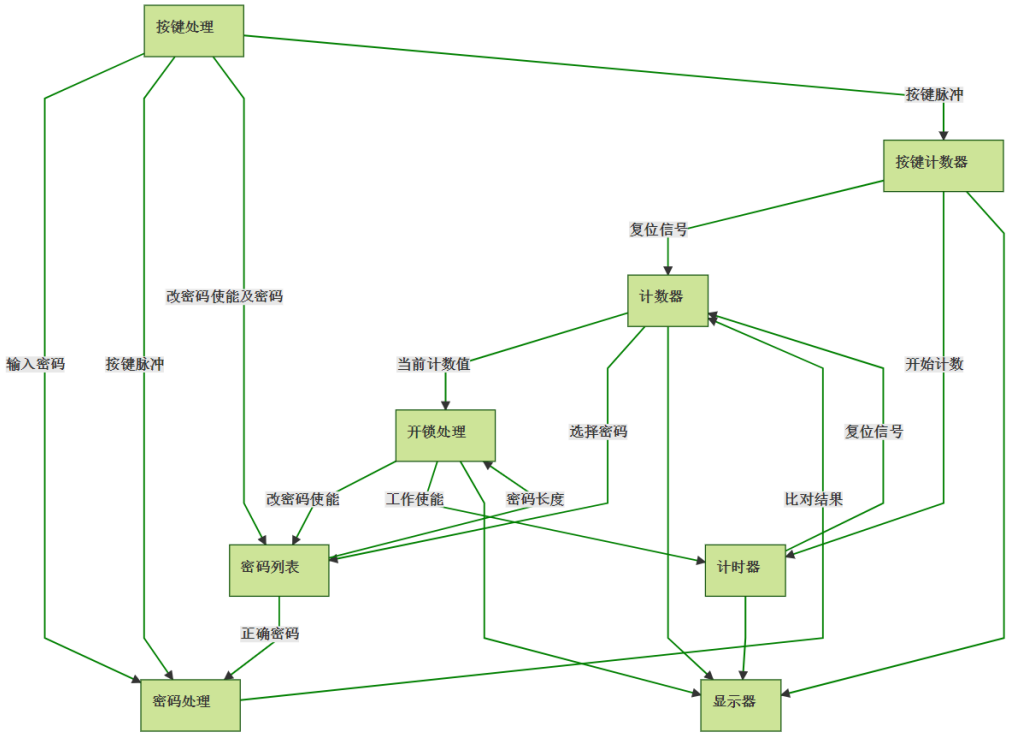
目录

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. 当开锁按扭开关（可设置 8 位或更多，其中只有 4 位有效，其余位为虚设）的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 4 位密码可以修改。
5. 记录按键次数并显示。
6. 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

密码处理　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。

密码表　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。

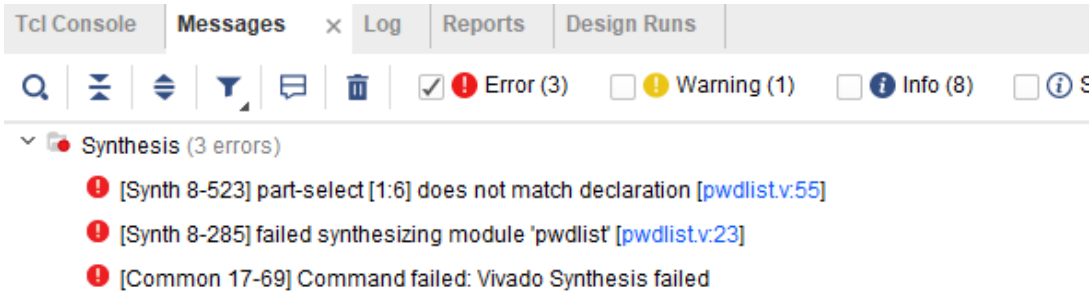
密码计数器　串行输入密码，每输对一位密码，计一次数，可以通过按键计数器与计时器所给的复位信号进行复位 (同步清零)。

开锁控制器　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。

第 4 章　调试过程

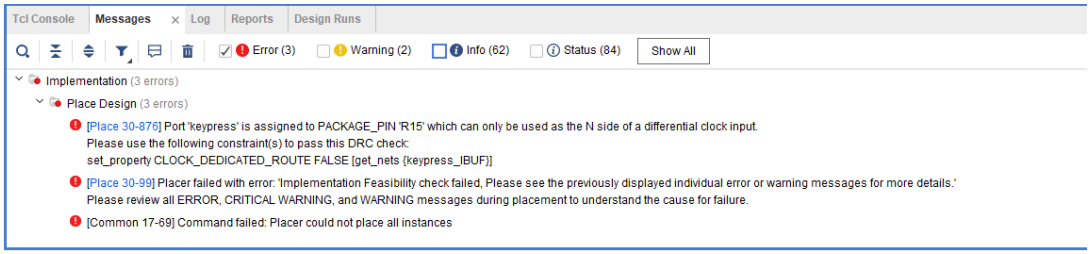
4.1　密码表中出现的问题

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。

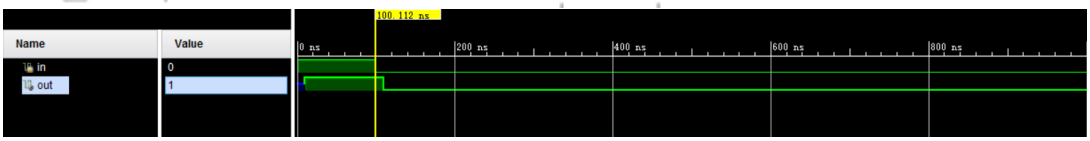
4.2　按键处理模块问题

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：

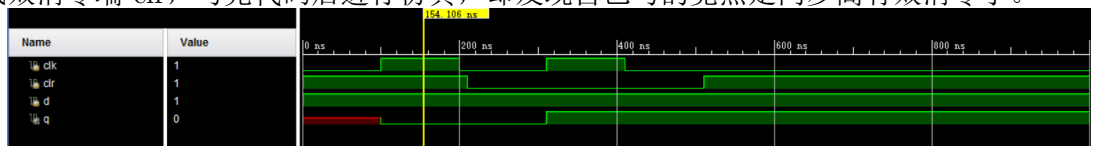
在 in 从高电平转为低电平之后的 10ns 内，out 并没有延续 in 之前高电平信号，而是变成高阻态，过了 10ns 才成为低电平状态，延迟器代码出现问题，发现其实只需要在最初始的 10ns 内为高阻态即可，需要初始化。按照下图修改之后，波形图为想要的图：

```
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。

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。可以修改密码，通过按键计数和时间限制进行报警。

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

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 入门教程

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

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

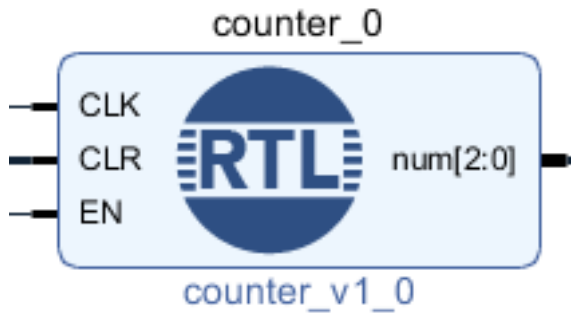


图 1: 密码长度计数器

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

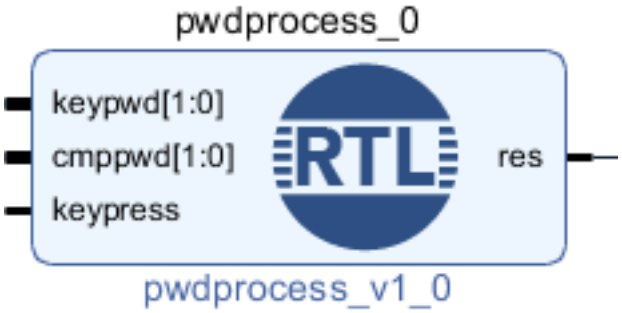


图 3: 密码处理

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

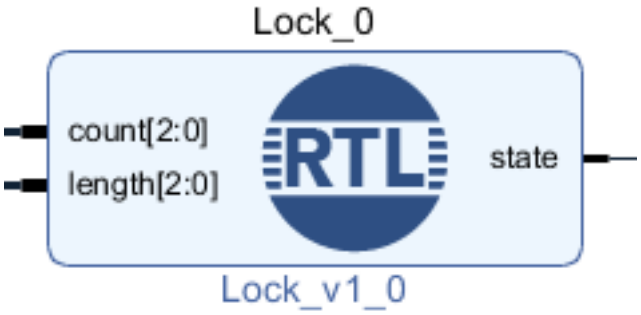


图 5: 开锁控制器

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

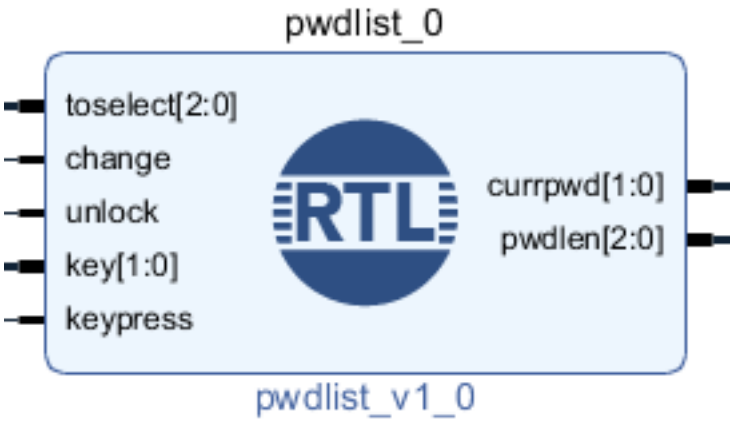


图 7: 密码表

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 按键部分

图 9: 按键部分

管脚名	功能
Clk1	给密码处理的脉冲信号
Clk2	给按键计数器的按键脉冲
En2	修改密码输出使能
Out1	输入密码输出
Out2	修改密码输出
S0 s3	按键输入

图 10: 功能表

B 总体设计图

C 仿真结果

C.1 A 部分仿真

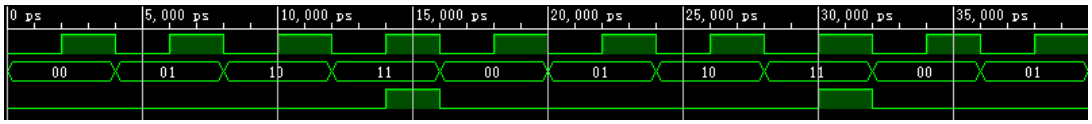
仿真激励代码

```
1 module Test();
2 reg Press;
3 reg [1:0]key_in;
4 wire out;
5
6 pwdprocess u0(key_in,3,Press,out);   //预设密码为3
7
```

```
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

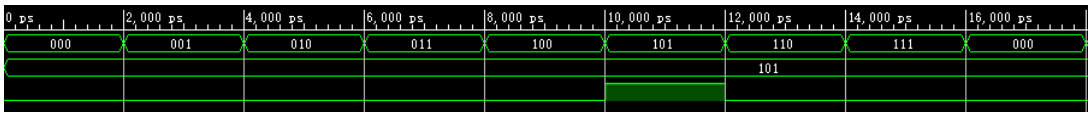
```
module Test();
reg [2:0]count;
reg [2:0]length;
wire state;

Lock u0(count,length,state);

initial begin
count=0;
length=5;
end

always #2 count=count+1;

endmodule
```

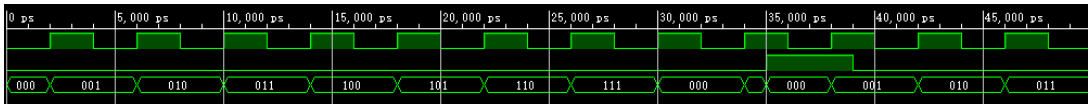


预设的密码长度为 5，经过仿真，输入的正确密码为 5 是才能获得开锁状态。

密码计数器

```
module Test();
reg clk;
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
```

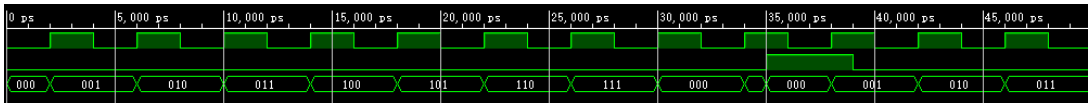


获得一次 clk 记一次数，上升沿有效，异步清零。

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
8  reg [1:0]key;             //修改密码所给的值
9  reg keypress;             //按键脉冲
10
11 pwdlist u0(select,beSelect,len,change,unlock,key,keypress);
12
13 initial begin
14 select = 0;
15 change = 1;
16 unlock = 1;
17 key = 0;
18 keypress = 0;
```

```
19 end
20
21 always #2 select=select+1;
22 always begin
23 #2 keypress = ~keypress;
24 key = key + 1;
25 end
26
27 endmodule
```

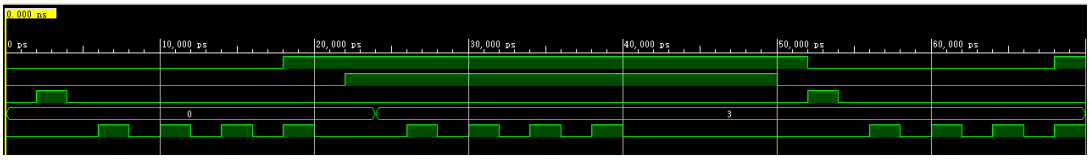


可以修改密码，最多 7 位，初始密码为 0000。

整体仿真

```
1  module Test();
2  wire LockState;
3  reg Change;                  //是否修改密码
4  reg Lock_It;                  //当前开锁状态
5  reg [1:0]KeyValue;                 //修改密码所给的值
6  reg Keypress;                //按键脉冲
7
8  Along u0(
9       Change,
10      Keypress,
11      KeyValue,
12      Lock_It,
13      LockState
14      );
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

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，实验报告。

D.1 密码处理

```
module pwdprocess(
    input [1:0] keypwd,                 //按键输入密码
    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 CLK;
    assign CLK = keypress && change && unlock;

    wire Enchange;
    assign Enchange = change && unlock;

    initial begin
    pwd = 0;
```

```
21      count = 0;
22      pwdlen = 4;
23      end
24
25      always @(toselect,pwd)
26      begin
27      case(toselect)
28      3'b000 : currpwd = pwd[1:0];
29      3'b001 : currpwd = pwd[3:2];
30      3'b010 : currpwd = pwd[5:4];
31      3'b011 : currpwd = pwd[7:6];
32      3'b100 : currpwd = pwd[9:8];
33      3'b101 : currpwd = pwd[11:10];
34      3'b110 : currpwd = pwd[13:12];
35      3'b111 : currpwd = pwd[15:14];
36      endcase
37      end
38
39      always @(posedge CLK or negedge Enchange)
40      begin
41      if(Enchange) begin
42      case(count)
43      3'b000 : pwd[1:0] = key;
44      3'b001 : pwd[3:2] = key;
45      3'b010 : pwd[5:4] = key;
46      3'b011 : pwd[7:6] = key;
47      3'b100 : pwd[9:8] = key;
48      3'b101 : pwd[11:10] = key;
49      3'b110 : pwd[13:12] = key;
50      3'b111 : pwd[15:14] = key;
51      endcase
52      count = count + 1;
53      end
54      else begin
55      if(unlock)
56      pwdlen = count;
57      count = 0;
58      end
59      end
```

```
60
61 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
```

赖昕 计时器，按键计数器，按键译码处理器, 显示器。