**数字逻辑与数字系统设计**

实验大作业报告

（2022年）

课程名称：数字逻辑与数字系统设计

任课教师：

作业题目：电子密码锁的设计

完成人：

学号：

班级：

报告日期： 2022年 11 月 15 日

|  |  |
| --- | --- |
| **报告成绩** | □**优秀: 4分;** □**良好: 3分;** □**中等：2.5分;**  □**及格及其他：< 2.5分;** □**雷同报告：0分** |
| **评判标准** | □**报告格式规范；（1分）**  □**各部分原理讲述清晰，逻辑图准确，仿真结果正确；（1分）**  □**有调试过程的说明，尤其是有对存在问题及解决方法的详细说明（要有截图）；（1分）**  □**设计结论客观准确，参考文献、设计心得与总结及附录等内容齐全。小组成员分工明确；（1分）** |
| **评阅人** |  |

目录

[一、 设计要求 3](#_Toc119962200)

[1.1 作业要求 3](#_Toc119962201)

[1.2 附加功能 3](#_Toc119962202)

[二、 工程原理和系统方框图 3](#_Toc119962203)

[2.1 工程原理 3](#_Toc119962204)

[2.1.1 显示 3](#_Toc119962205)

[2.1.2 状态设置 3](#_Toc119962206)

[2.1.3 输入 4](#_Toc119962207)

[2.1.4 原理说明 4](#_Toc119962208)

[2.2 端口功能与引脚约束 5](#_Toc119962209)

[2.3 系统方框图 7](#_Toc119962210)

[三、 各部分模块具体功能及设计思路 7](#_Toc119962211)

[3.1 七段数码管显示模块 7](#_Toc119962212)

[3.1.1 具体功能 7](#_Toc119962213)

[3.1.2 实际思路 7](#_Toc119962214)

[3.2 模10计数器模块 8](#_Toc119962215)

[3.2.1 具体功能 8](#_Toc119962216)

[3.2.2 实际思路 8](#_Toc119962217)

[3.3 重置密码模块 9](#_Toc119962218)

[3.3.1 具体功能 9](#_Toc119962219)

[3.3.2 实际思路 9](#_Toc119962220)

[3.4 倒计时模块 10](#_Toc119962221)

[3.4.1 具体功能 10](#_Toc119962222)

[3.4.2 实际思路 10](#_Toc119962223)

[3.5 按键除抖模块 12](#_Toc119962224)

[3.5.1 具体功能 12](#_Toc119962225)

[3.5.2 实际思路 12](#_Toc119962226)

[四、 调试过程 14](#_Toc119962227)

[五、 设计心得与总结 15](#_Toc119962228)

[5.1 本次实验总结： 15](#_Toc119962229)

[5.2 设计心得： 16](#_Toc119962230)

[六、 参考文献 17](#_Toc119962231)

[七、 附录 17](#_Toc119962232)

[7.1 附录1 17](#_Toc119962233)

[7.2 附录2：模块仿真截图： 17](#_Toc119962234)

[7.3 附录3：小组成员工作说明 18](#_Toc119962235)

# 设**计**要求

## 作业要求

（1）设计一个开锁密码至少为 4 位数字的密码锁。

（2）当开锁按钮开关（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，用 K1 灯亮表示开锁状态。

（3）从第一个按钮触动后的 9 秒内若未能将锁打开，则电路自动复位，同时用F1灯灭表示关锁状态。

（4）开锁倒计时要求用数码管显示出来，当输入密码正确时停止倒计时。

## 附加功能

（1）开始按钮时倒计时显示，9秒

（2）可以记录按键次数

（3）密码锁中4位密码可修改

（4）能够显示当前所输入密码

（5）按键除抖

# 工程原理和系统方框图

## 工程原理

### 显示

（1）左侧第一个显示倒计时（9秒），每一个数码管需要7个输出来确定字形，同时需要一个位置寄存器来确定位置。

（2）右侧四个数码管显示密码，每一个数码管需要7个输出来确定字形，同时需要一个位置寄存器来确定位置。

### 状态设置

（1）设置四个状态：

设置密码（set\_pw\_state）——表示是否设置密码，置为1则动态保存新设置的密码，置为0保存密码

正在解锁状态（lock\_state）——表示开锁状态，状态为0时锁关闭，不可更改密码，同时右侧K2提示灯灭；状态为1时表示已开锁，可修改密码，同时右侧K2提示灯亮。

reset\_signal——复位状态

input\_start, input\_reset——密码输入的开始、复位状态

（2）设置四个常量pw[0:3]来记录密码、设置四个变量in[0:3]来记录输入、当两个部分相同时开锁、当处于设置密码状态(set\_pw\_state)时，pw[3:0]等于in[3:0]。

### 输入

（1）通过左边第三个按键P3，表示是否设置密码，置为1则动态保存新设置的密码，置为0保存密码

（2）通过左边第二个按键P4，表示是否输入并动态验证密码，置为1则表示输入密码状态，同时对当前输入的密码进行验证；置为0表示重置输入。

（2）通过四个按键R17,R11,U4,V1提供信号，将其作为密码输入的数字+1上升沿，使用模10计数器模块对其进行计数，并对四个键进行按键除抖操作。

### 原理说明

（1）采用“按一下，增加1”的方式实现数字的输入。这里的实现方式是利用mod10计数器控制输入数字，借助拨码按键模拟时钟上升沿，并对按键进行消抖，将消抖后的信号传递给计数器。计数器每接收到一个上升沿，就让计数器+1，数字9加一后变为数字0，这样就完成了十进制0~9数码的输入。

（2）电路不同状态的控制：由于密码锁需要实现“开锁”和“密码重置”两个主要功能，电路需要定义两种状态，一是开锁状态，二是重置密码状态。通过两个拨码按键来控制这两种状态。

a)当其中一个拨码按键（P4）输出高电平时，让电路处于输入密码状态，此时指示灯处于熄灭状态，表示锁未被打开。可以通过输入四位十进制数码来检测验证密码，当输入正确时开锁成功，对应指示灯K2亮，否则K2保持熄灭状态。

b)当重置密码对应的拨码（P3）按键输出高电平时，如果同时锁处于打开状态，就允许通过修改数码来重置密码，否则修改数码无效。

（3）七段数码管的显示：单独编写一个数码管模块，将输入的数字(8421BCD码)译成七段数码管的驱动信号(7位)数据输出到数码管位选端口，另用一个复位信号控制数码管输出置0。4位数码需要动态显示时，不断扫描四个端口，改变位选信号完成4位数码的动态显示。

（4）开锁倒计时：通过一个计数器不断记录系统时钟上升沿数量，这里定义100000000个上升沿对应一秒钟。当记录到0个上升沿时，倒计时剩余9秒；当记录到100000000个上升沿时，倒计时剩余8秒；当记录到200000000个上升沿时，倒计时剩余7秒；当记录到300000000个上升沿时，倒计时剩余6秒……当记录到900000000个上升沿时，倒计时剩余0秒，电路需要复位，此时指示灯不亮，倒计时结束提示灯亮，表示开锁没有成功且倒计时结束。

（5）按键除抖：由于按键所用开关为机械弹性开关，当机械触点断开、闭合时，由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下断开。因此在闭合及断开的瞬间均会伴有一连串的抖动，出现多个输入信号，会对正常功能造成一定的干扰。对此，通过编写debounce模块，对提供的原始按键信号进行处理，使用处理后的信号作为真正传入电路的时钟信号，确保了“按一下，只增加1”。



图1 按键抖动传入的原始信号

## 端口功能与引脚约束

表1 端口功能说明

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | 类型 | 位宽 | 功能 |
| clk | input | 1 | 系统的时钟信号 |
| clk0 | input | 1 | 按键信号 |
| clk1 | input | 1 | 按键信号 |
| clk2 | input | 1 | 按键信号 |
| clk3 | input | 1 | 按键信号 |
| reset | input | 1 | 复位信号 |
| set\_pw\_state | input | 1 | 密码设置状态 |
| judge | input | 1 | 开锁命令信号 |
| lock\_state | output | 1 | 密码开锁状态 |
| reset\_signal | output | 1 | 输入复位信号 |
| pos\_countdown | output | 1 | 倒计时数字显示位置 |
| led | output | 7 | 密码数字的数码管显示 |
| led\_countdown | output | 7 | 倒计时数字的数码管显示 |
| pos | output | 4 | 密码数字的显示位置 |

表2 引脚约束列表

|  |  |  |
| --- | --- | --- |
| 端口 | 方向 | 管脚号 |
| clk | input | P17 |
| clk0 | input | R17 |
| clk1 | input | R11 |
| clk2 | input | V1 |
| clk3 | input | U4 |
| reset | input | P5 |
| set\_pw\_state | input | P3 |
| judge | input | P4 |
| lock\_state | output | K2 |
| reset\_signal | output | F6 |
| pos\_countdown | output | G2 |
| led[6:0] | output | D4, E3, D3, F4, F3, E2, D2 |
| led\_countdown[6:0] | output | B4, A4, A3, B1, A1, B3, B2 |
| pos[3:0] | output | G1, F1, E1, G6 |

## 系统方框图

图2 各模块功能关系图

# 各部分模块具体功能及设计思路

## 七段数码管显示模块

### 具体功能

在时钟控制下，将七段数码管显示的字形与输入的数字对应，完成数码管的显示。

输入：clk（时钟），data（要显示的数字，4位二进制数表示），reset（复位信号）

输出：show（数码管对应的字形）

### 实际思路

|  |
| --- |
| module represent\_number(  input clk,  input reset,  input [3:0] data,  output reg[6:0] show  );  always @(posedge clk or posedge reset)  begin  if(reset)  show <= 7'b1111110;//复位，输出为零  else  begin  case(data)//0-9九个数对应的七段数码管取值  4'b0000: show <= 7'b1111110;  4'b0001: show <= 7'b0110000;  4'b0010: show <= 7'b1101101;  4'b0011: show <= 7'b1111001;  4'b0100: show <= 7'b0110011;  4'b0101: show <= 7'b1011011;  4'b0110: show <= 7'b1011111;  4'b0111: show <= 7'b1110000;  4'b1000: show <= 7'b1111111;  4'b1001: show <= 7'b1111011;  default:  show <= 7'b0000000;//其他情况不亮灯  endcase  end  end  endmodule |

## 模10计数器模块

### 具体功能

十进制数字的输入：由于开发板按键资源数量有限，无法实现数字0-9与按键的一一对应，只能采用“按一下，增加1”的方式实现数字的输入。这里的实现方式是利用mod10计数器控制输入数字，借助拨码按键模拟时钟上升沿，计数器每接收到一个上升沿(即拨码按键拨动了一下)，就让计数器+1，数字9加一后变为数字0，这样就完成了十进制0~9数码的输入。

输入：clk（按键上升沿），reset（复位信号），start（开始信号）

输出：out（输出当前数字）

### 实际思路

|  |
| --- |
| module mod10\_counter(  input clk,  input reset,  input start,  output reg [3:0] out  );  always @(posedge clk or posedge reset) //reset为1时，进行置0操作  begin  if(reset)  out <= 4'b0000;  else if(start)  begin  if(out == 4'b1001)  out <= 4'b0000; //模10  else  out <= out + 4'b0001;  end  end  endmodule |

## 重置密码模块

### 具体功能

当处于开锁状态(lock\_state)和设置密码状态(set\_pw\_state)时，将密码更改为输入的值；

在按键控制下，将密码更改为输入的密码。

输入：in0、in1、in2、in3（按键）

输出：pw0、pw1、pw2、pw3（四位开关要更改的密码）

### 实际思路

|  |
| --- |
| if(lock\_state == 1 && set\_pw\_state == 1) //锁打开时，若密码设置  begin  pw0 <= in0;  pw1 <= in1;  pw2 <= in2;  pw3 <= in3;  end |

## 倒计时模块

### 具体功能

1)在输入密码时，开启倒计时，共9秒。

2)当开始输入第一位密码后，此时的计时器开始工作，并且将计时器工作状态置为1，这样保证在倒计时过程中计时器持续触发，在倒计时完成后计时器会发出超时信号，并将计时器工作状态置为0。在倒计时过程中，剩余时间将会实时显示。

3)在规定时间内输入正确密码，则倒计时停止，倒计时结束后不再允许输入密码。

4)由于开发板上系统频率为100MHz，这里定义100000000个上升沿对应一秒钟。

### 实际思路

|  |
| --- |
| if(judge && countdown != 4'b0000 && lock\_state == 0) //此时进行倒计时，进行分频  begin  if(clock\_countdown != 900000000)  begin  clock\_countdown <= clock\_countdown + 1;  end  case(clock\_countdown)  900000000: countdown <= 4'b0000;  800000000: countdown <= 4'b0001;  700000000: countdown <= 4'b0010;  600000000: countdown <= 4'b0011;  500000000: countdown <= 4'b0100;  400000000: countdown <= 4'b0101;  300000000: countdown <= 4'b0110;  200000000: countdown <= 4'b0111;  100000000: countdown <= 4'b1000;  0 : countdown <= 4'b1001;  endcase  end  else if(~judge)  begin //未处于开锁状态  countdown <= 4'b1001;  clock\_countdown <= 0;  end |

在倒计时结束后，计时器会发出超时信号，并将计时器工作状态置为0。

|  |
| --- |
| if(countdown == 4'b0000)  begin  input\_start <= 0;  input\_reset <= 1; //清零  reset\_signal <= 1;  end  else  reset\_signal <= 0;  input\_start <= set\_pw\_state | judge; //处于输入状态  input\_reset <= reset | ((!judge) & (!set\_pw\_state)); //关闭judge和保存密码开关，则重置输入 |

在规定时间内输入正确密码，lock\_state置为1,表示开锁，倒计时结束后不再允许输入密码，lock\_state置为0。

|  |
| --- |
| always @(posedge timeout)  begin  if(pos == 4'b0001)  begin  pos <= 4'b0010;  num <= in1;  end  else if(pos == 4'b0010)  begin  pos <= 4'b0100;  num <= in2;  end  else if(pos == 4'b0100)  begin  pos <= 4'b1000;  num <= in3;  end  else if(pos == 4'b1000)  begin  pos <= 4'b0001;  num <= in0;  end  end  always @(\*) //检查输入状态  begin  if(judge)  begin  if(pw3==in3&&pw2==in2&&pw1==in1&&pw0==in0&&countdown!=4'b0000)  lock\_state <= 1;  else  lock\_state <= 0;  end  end |

## 按键除抖模块

### 具体功能

首先我们定义状态值hold\_on，用来表示当前是否已经接受到信号并正在进行延时判断。当检测到按键输入有变化时，若hold\_on为0，则将其置为1，且开始进行20ms的计时，这段时间内的所有输入信号的改变都将被忽略；20ms后，将hold\_on置为0，同时对输入值再次进行判断，若输入值仍与原输入状态不同，则判断为有效，对该输入值进行输出；否则认为之前的输入变化由按键抖动引起，不改变输出值。

输入：clk（按键上升沿），key\_in（按键输入）

输出：key\_out（按键输出）

### 实际思路

|  |
| --- |
| `timescale 1ns / 1ps module debounce(     input clk,     input key\_in,     //给1个按钮除颤     output reg key\_out     );     integer count;     //如果reset为0，则一直保持计时状态，当计时20ms后检查key\_in是否与key\_out相等      reg hold\_on;    //当前是否是处于延时状态     reg check;          //为1时，检查key\_in是否与key\_out相等，若不相等则key\_in按键有效，若相等说明是颤动     initial begin     count = 0;     end          always@ (posedge clk) begin     if(hold\_on == 1) begin         if(count >= 2\_000\_000 && count < 2\_001\_000) begin             count <= count + 1;             check <= 1;         end         else if(count >= 2\_001\_000 && hold\_on == 1) begin             count <= 0;             hold\_on <= 0;             check <= 0;         end         else begin             count <= count + 1;         end     end     else if(hold\_on == 0) begin         count <= 0;     end     end          always @ (posedge clk) begin          if(hold\_on == 0 && key\_in != key\_out)  //说明没有开始计时，检测到当前key与保存的key值不同，开始计时             begin             hold\_on <= 1;             end     end      always @ (posedge clk)   //进行检查     begin     if(check == 1 && key\_out != key\_in) begin         key\_out <= key\_in;         end     end       endmodule |

# 调试过程

1. 四位密码无法在数据板上显示：

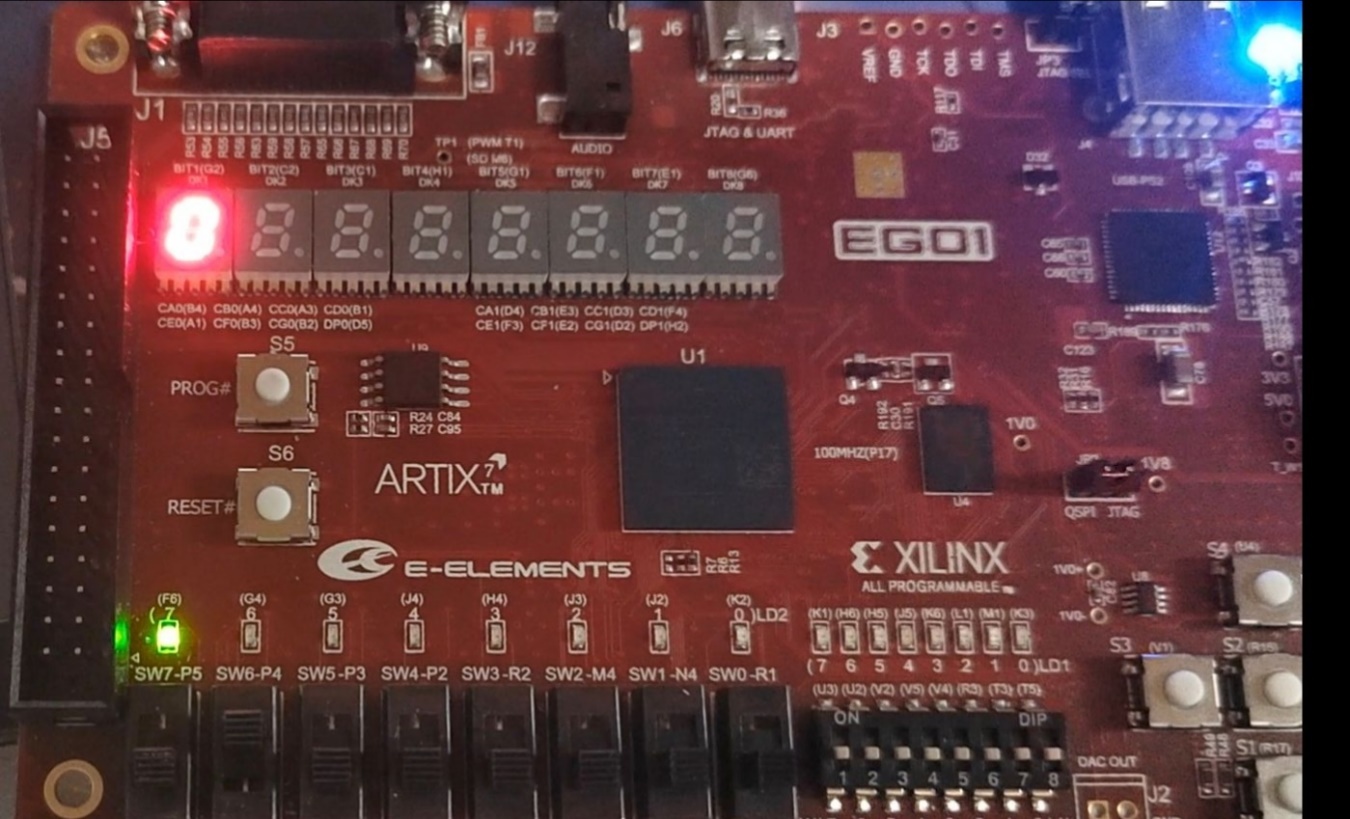


图3 四位密码无法在数字板显示

原因一：

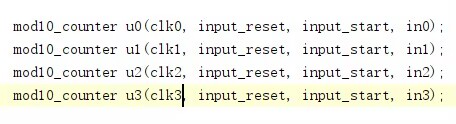


图4

我们最开始是将上述clk0, clk1, clk2, clk3都写成了clk，但是这里不能是让clk提供信号，clk0到clk3对应三个拨码开关，是我们手动给的上升沿。

原因二：

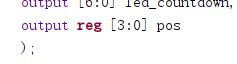


图5

经过仔细的筛查，我们发现用来选择右侧数字显示位置的pos没有写成output，不定义为output的话对pos的约束就无效，自然就没法显示数字了。

在将上述错误更正后，数字板成功的显示了数字。

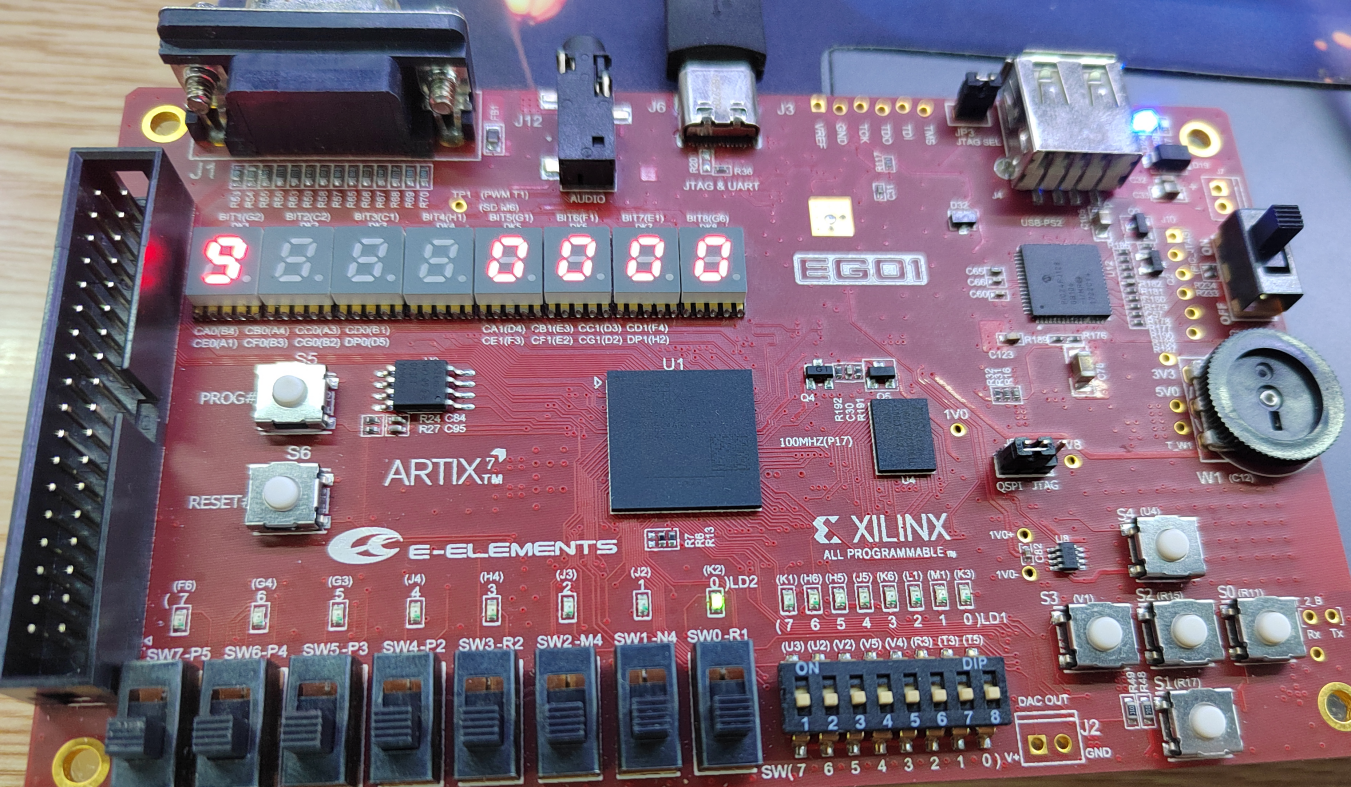


图 6 修改错误后的数字板

1. 手动按下按钮，有时候按一下记了2次甚至3次，于是我们编写相关模块，增加了按键除抖的功能，使其按一次仅加一，在按键20ms后检查key\_in是否与key\_out相等检查key\_in是否与key\_out相等，若不相等则key\_in按键有效，若相等则认为是按键抖动。

# 设计心得与总结

## 本次实验总结：

优点：

（一）正确实现电子密码锁的设计项目预期的各种要求

（二）实现了多项附加功能

（三）交互性好，用户体验感好设计结果成功

创新：使用按键除抖功能解决按键处理问题。

不足：密码输入位数只有4位。

需要改进的地方：密码输入位数。

## 设计心得：

这次密码锁的设计让我们明白了verilog的强大之处，由于它在其门级描述的底层,也就是晶体管开关的描述方面比VHDL等各种其它的HDL语言有更强的功能。所以在复杂数字逻辑电路和系统的设计仿真时更有优势;描述的设计思想、电路结构和逻辑关系清晰明了,并且设计语言简练、易学易用;其模块化分层结构在大规模设计时更能体现出优势。

通过大作业，使我们更加了解了FPGA的相关设计与实现的操作。使我体会到了：清晰的思路十分重要，框图是硬件设计的中心，原则上不包含任何实际的元器件，采用模块或者IP核的方式构成系统，主要描述每个模块涉及到的输入和输出，以及模块之间的联系，即描述系统各部分连接关系的结构图；设计激励判别Verilog源程序是否符合功能要求，用来检测Verilog编码的正确性，但应注意，完全的正确性验证是不现实的，模拟测试只能验证设计的不正确性，波形模拟结果是由设计者判定的。在Vivado中通过约束文件限定管脚，即每个端口都可与相应的管脚对应，要注意正确对应管脚，不要出现管脚冲突；波形模拟与实际在开发板上运行始终存在差异，最后一定要对实际电路进行测试，而且原来的波形模拟的输入信号不再是人为输入，而是来源于电路板上的其他芯片，很可能出现时序问题，抗干扰问题，速度问题等新问题需要解决。

总结来说，本次大作业并且在设计过程中使我们更加深刻理解了集成电路的设计和运行原理，基本掌握了verilog语言和开发板的使用，尤其是学会了FPGA开发板上七段数码管的使用，以及如何对按键进行除抖，理解了原理并实现了该功能。我们设计的密码锁既有优点，也存在着一些不足。

在过程中我们感觉到了数字逻辑的奇妙，更加热爱了数字逻辑课程。

# 参考文献

[1]数字逻辑与数字系统设计实验大作业——4位密码锁，2020年 12月 13日，

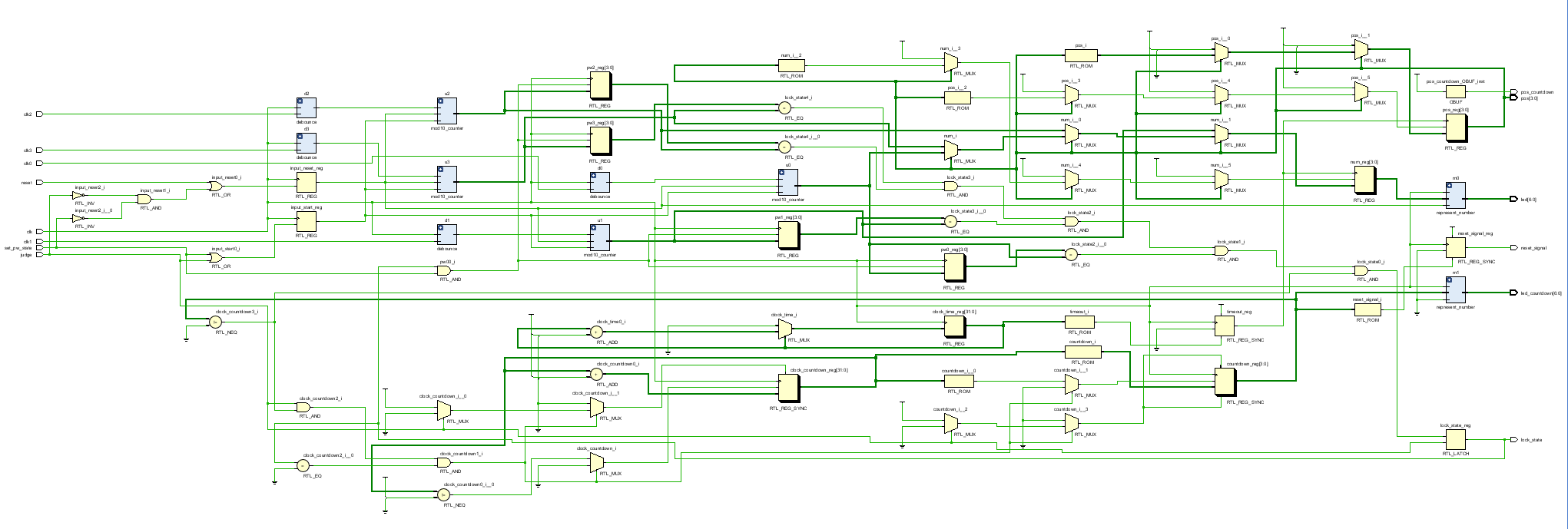
原文链接：https://blog.csdn.net/Carrot\_kexin/article/details/120676663

[2]汪国强,李尚甫,王飞,谢丽丽,王钦,孙柏.基于FPGA的四位电子密码锁设计与实现[J].无线电通信技术,2016,42(04):95-98.

# 附录

## 附录1

总体设计图：

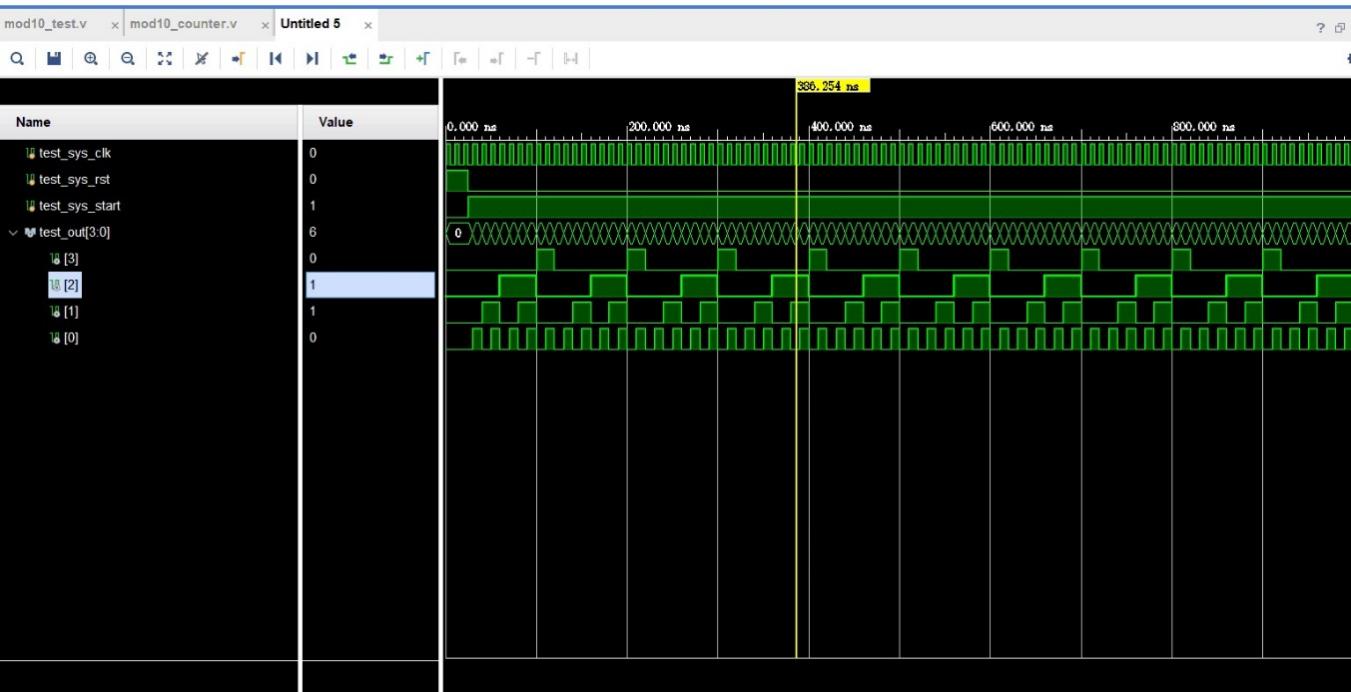


## 附录2：模块仿真截图：

模10仿真文件：

|  |
| --- |
| module mod10\_test();     reg test\_sys\_clk;     reg test\_sys\_rst;     reg test\_sys\_start;     wire [3:0] test\_out;          initial begin         test\_sys\_clk = 1&apos;b1;     end          always #5 test\_sys\_clk = ~test\_sys\_clk;          initial begin         test\_sys\_rst <= 1&apos;b1;         test\_sys\_start <= 1&apos;b0;         repeat(3)         @(negedge test\_sys\_clk);         test\_sys\_rst <= 1&apos;b0;         test\_sys\_start <= 1&apos;b1;     end          mod10\_counter mod10(test\_sys\_clk, test\_sys\_rst, test\_sys\_start, test\_out);  endmodule |

模10的波形图：



## 附录3：小组成员工作说明