哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2022 秋季

课程名称：数字逻辑设计（实验）

实验名称：密码锁设计

实验性质：综合设计型

实验学时：6 地点：T2615

学生班级：计科 6 班

学生学号：210110620

学生姓名：李松霖

评阅教师：

报告成绩：

实验与创新实践教育中心制

2022 年 12 月

设计的功能描述
概述基本功能、详细描述自行扩展的功能 基本功能：实现了 3 位数字的密码锁。 按键： • 按 s1 复位进入初始状态。 • 按 s2 设置密码（若已解锁或未输入密码）。 • 按 s3 验证密码（已经有设置密码）。 • 按 s5 确认（满三位输入）。 数码管显示： • 初始状态显示 00000000。 • 设置状态数码管全灭，随后根据 4*4 小键盘输入依次使能第一、二、三位数码管并显示其内容。 • 确认状态显示 00000000。 • 验证状态数码管全灭，随后根据 4*4 小键盘输入依次使能第一二三位数码管并显示其内容。 • 输入累计失败三次状态，显示 ffffffff。 信号灯显示： • 输入确认后 GLD0 亮。 • 根据累计失败次数显示对应数量红灯，最高显示 RLD5、RLD6、RLD7 三位（此时系统已经锁住）。 • 验证密码状态输入正确密码 GLD7 亮。

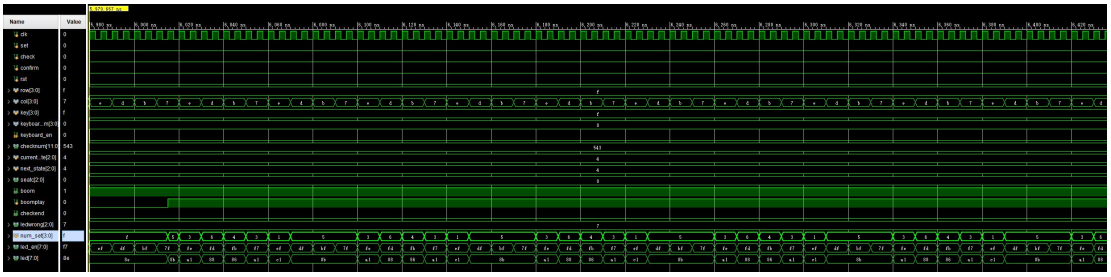
自行拓展功能：

系统锁住显示“ffffffff”短暂间隔后显示“hhhUdead”字幕。

详细介绍：

在系统锁住后本会显示“ffffffff”字样，为增加趣味性，模拟拆弹输错三次密码后不仅锁住还会爆炸，添加 boom 模块，内置计时器和另一套字母表，在系统锁住后采用另一套字母表，在经过计时器短暂计时后将“ffffffff”替换成“hhhUdead”字样并保持。

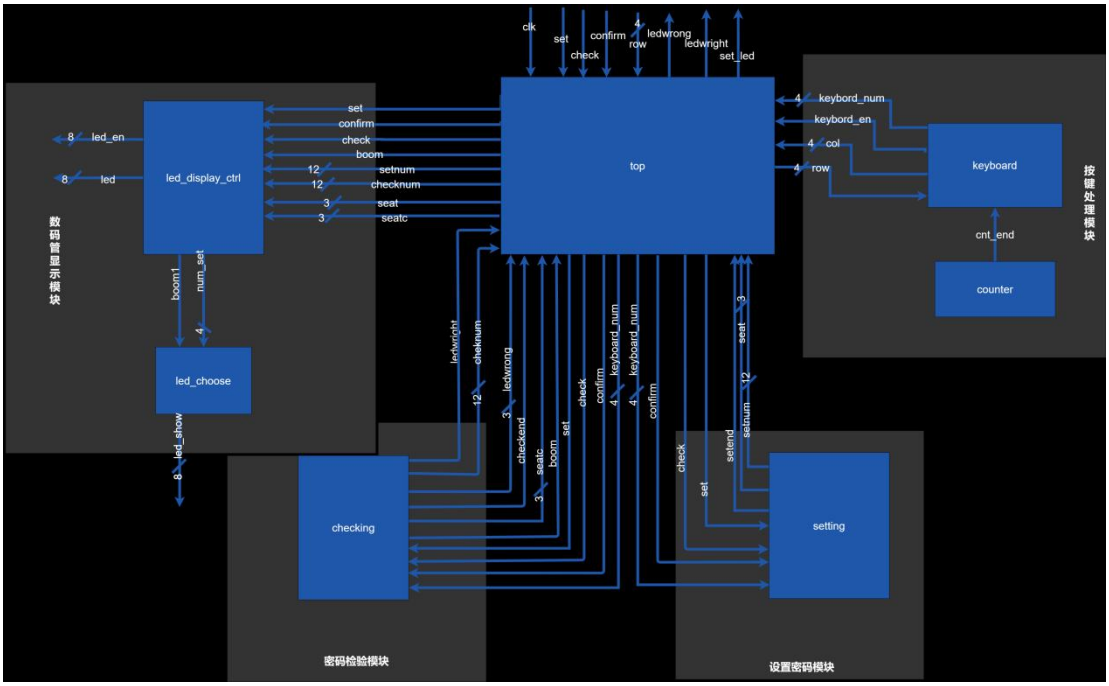
仿真图：



分析：在 boom 出现一段时间后（计数器控制）boomplay = 1，切换 led_diaplay_ctrl 模块下 led_choose 模块中的字母表，num_set 选择每个数码管位显示数字，即控制 led，随 led_en 轮播使能在数码管上显示“hhhUdead”8 位字符。

系统功能详细设计

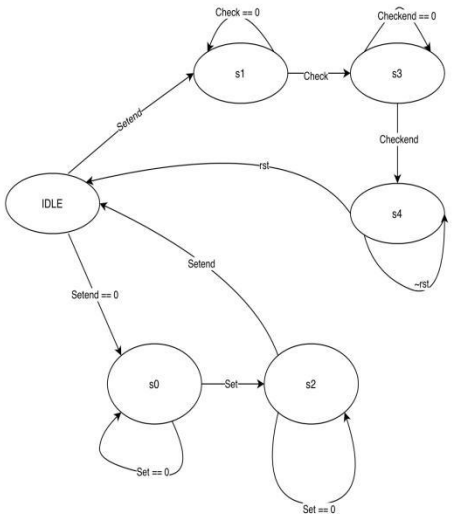
硬件框图：



注：密码检验模块与设置密码模块即为密码控制模块；由于线条数较多较密，故略去了 clk 和 rst 等两条线在模块间的连接，实则所有模块共用 clk 和 rst 两信号。

Top 模块（功能）：

状态编码	状态描述
IDLE = 111	初始状态
S0 = 000	已解锁或者未设置密码
S1 = 001	已设置密码
S2 = 010	进入设置
S3 = 011	进入验证密码
S4 = 100	解锁状态



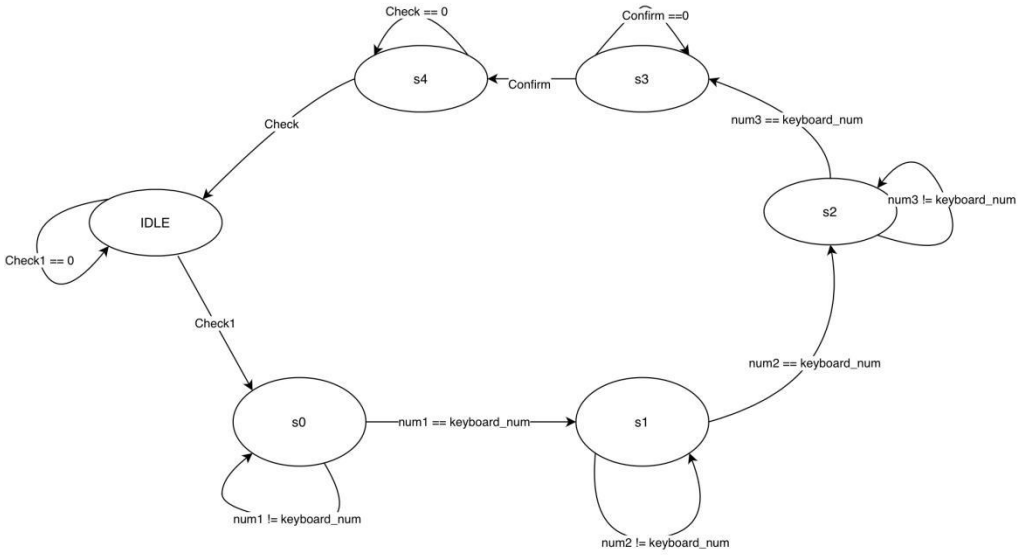
Top（功能）信号描述

信号	信号描述
rst	复位信号
set	进入设置信号
setend	已设置密码信号
check	进入验证密码信号
checkend	密码检测正确信号

密码匹配状态描述、信号描述：

状态编码	状态描述
IDLE = 111	初始状态
S0 = 000	已进入检查
S1 = 001	第一位正确
S2 = 010	第二位正确
S3 = 011	第三位正确
S4 = 100	已确认结果

信号	信号描述
rst	复位信号
set	进入设置信号
setend	已设置密码信号
check	进入验证密码信号
checkend	密码检测正确信号



各模块描述

●主模块：

功能：

1. 连接各个底层模块，接收传输模块间信号。
2. 承载功能状态机，根据接收的模块信号驱动状态机，从而使能相关模块。

端口：

输入	输出
时钟信号端口	数码管使能端口
密码设置端口	数码管 A-DP 段使能端口
检测密码端口	
键盘输入端口	密码信号灯端口(含正确与失误)
数码管显示端口	

变量	含义
clk	时钟
set	s2 设置按下
check	S3 检测按下
confirm	s5 确认按下
rst	复位
row	行信号
col	列信号
set_led	设置完成灯信号
led_en	数码管位使能
led	数码管 A-DP 段使能
ledwrong	密码错误灯
keyboard_num	键盘按下数字
ledwright	密码正确使能
current_state	状态机当前状态
next_current	状态机下一状态
keyboard_en	键盘是否按下
setend	设置完成
checkend	检测完成
boom	连续输错三次信号
seat	设置时已输入数字个数
seatc	验证时已输入数字个数
setnum	设置时用于承接键盘输入
checknum	验证时已输入数字个数

主要设计代码：

```
always @(*)
begin//只判断状态！
  if(rst) next_state = IDLE;
    else begin
    case (current_state)
       IDLE : if(setend) next_state = s1;//已设置密码
              else  next_state = s0;//已经解锁或者未设置密码
       s0 : if(set) next_state = s2;//进入设置
            else next_state = s0;
       s1 : if(check) next_state = s3;//进入验证密码
            else next_state = s1;
       s2 : if(setend) next_state =IDLE;//设置完的信号
            else next_state = s2;
       s3 : if(checkend) next_state = s4;//正确信号
            else next_state = s3;
       s4 : if(~rst) next_state = s4;//解锁状态
            else next_state = IDLE;
       default: next_state = IDLE;
    endcase
  end
end
```

●数码管显示模块：

功能：

1. 负责接收主模块传入的灯控信号。
2. 根据接收灯控信号调控 led_choose 模块选择字符，轮播显示在数码管位上。
3. 锁机显示“ffffffff”后短暂间隔后切换字母表，显示“hhhUdead”。

端口：

输入	输出
时钟信号端口	数码管使能端口
主模块控制端口	数码管字符输出端口

变量	含义
seat	设置时已输入数字个数
seatc	验证时已输入数字个数
setnum	设置时用于承接键盘输入
checknum	验证时已输入数字个数
led	显示字符控制
led_en	数码管使能

主要设计代码（其中一个数码管位）：

```
        8'b01111111:begin//
                if(rst)begin
                 num_set = 4'b1110;
                 end
                 else if(set1)begin
                         if(seat[2]==1'b0)begin
                         num_set = setnum[11:8];
                         end
                         else
                         num_set = 4'b1110;
                 end
                 else if(check1)begin
                         if(seatc[2]==1'b0)begin
                         num_set = checknum[11:8];
                         end
                         else
                         num_set = 4'b1110;
                 end
                 else if(boom1)begin
                       if(boomplay)
                                begin
                                    num_set = 4'b0101;
                                end   else
                 num_set = 4'b1111;
                 end
                 else
                    num_set = 4'b0000;
              end
        endcase
end
```

```
always @(posedge clk or posedge rst )//数码管轮播，即轮流使能
begin
                if(rst) begin
                   counter3 <=27'd0;
                   led_en <= 8'b11111110;
                end
                else if (counter3 == 27'd1) begin//原为100000
                   counter3 <=27'd0;
                   led_en <= {led_en[6:0],led_en[7]};
                end
                else begin
                   counter3 <= counter3 +27'd1;
                 end
end

always @(posedge clk  )//“hhhUdead"字幕倒计时
begin
                if(rst) begin
                   counterplay <=27'd0;
                   boomplay<= 1'b0;
                end
                else if (counterplay == 27'd1000) begin//原
为100000000
                   boomplay<= 1'b1;
                end
                else if(boom1) begin
                   counterplay <= counterplay +27'd1;
                   boomplay <= boomplay;
                 end
           else;
end
```

● 设置密码模块：

功能：

1. 检测到 set 信号后启动密码设置状态机。
2. 将由主模块传递的 keyboard_num 收集整理
3. 将收集密码传输到数码管显示（逐位使能逐位显示）

端口：

输入	输出
时钟信号端口	密码端口
主模块控制端口	灯控信号端口

变量	含义
clk	时钟
set	s2 设置按下
check	S3 检测按下
confirm	s5 确认按下
rst	复位
row	行信号
col	列信号
set_led	设置完成灯信号
led_en	数码管位使能
led	数码管 A-DP 段使能
keyboard_num	键盘按下数字
ledwright	密码正确使能
current_state	状态机当前状态
next_current	状态机下一状态
keyboard_en	键盘是否按下
setend	设置完成
seat	设置时已输入数字个数
setnum	设置时用于承接键盘输入

主要设计代码：

```
always @(*) //
begin
  if(rst) next_state = IDLE;
    else begin
    case (current_state)
       IDLE : if(set) next_state = s0;//进入第一位设置
              else  next_state = IDLE;
       s0 : if(keyboard_en) next_state = s1;//进入第二位设置
            else next_state = s0;
       s1 : if(keyboard_en) next_state = s2;//进入第三位设置
            else next_state = s1;
       s2 : if(keyboard_en) next_state = s3;//进入第三位设置
            else next_state = s2;
       s3 : if(confirm) next_state = s4;//设置完的信号
            else next_state = s3;
       s4 : if(rst) next_state = IDLE;
            else next_state = s4;//成功亮灯篆
       default: next_state = IDLE;
   endcase
  end
end


always @(posedge clk or posedge rst )
begin
     if(rst) seat<= 3'b111;
     else  if(next_state == s1)  seat[2] <= 1'b0;
     else  if(next_state == s2)  seat[1] <= 1'b0;
     else  if(next_state == s3)  seat[0] <= 1'b0;
     else seat <= seat ;
end
```

```
always @(posedge clk or posedge rst )
begin
     if(rst) setnum<= 12'b111111111111;
     else  if(keyboard_en&&next_state ==
s1)  setnum =
{keyboard_num,setnum[11:4]};
     else  if(keyboard_en&&next_state ==
s2)  setnum =
{setnum[11:8],keyboard_num,setnum[3:0]};
     else  if(keyboard_en&&next_state ==
s3)  setnum =
{setnum[11:4],keyboard_num};
     else setnum <= setnum ;
end
```

●密码检验模块：

功能：

1.检测到 check 信号后启动密码设置状态机。

2.将由主模块传递的 keyboard_num 收集整理

3.将收集密码传输到数码管显示（逐位使能逐位显示）

2.根据输入判断密码是否正确，若正确则输出正确信号，若错误则输出错误信号并根据错误次数判断是否锁机。

端口：

输入	输出
时钟信号端口	密码端口
主模块控制端口	灯控信号端口

变量	含义
seatc	输入密码占位
ledwrong	错误灯控
ledwright	正确灯控
current_state/next_state	密码匹配状态机
current_state1/next_state1	密码输入位数状态机
boom	锁机控制信号

主要设计代码：

```
always @(*) //
begin
  if(rst) next_state = IDLE;
    else begin
    case (current_state)
       IDLE : if(check1) next_state = s0;//check 已经被按下
              else  next_state = IDLE;
       s0 : if(num1 == keyboard_num) next_state = s1;//第一位正确
            else next_state = s0;
       s1 : if(num2 == keyboard_num) next_state = s2;//第二位正确
            else next_state = s1;
       s2 : if(num3 == keyboard_num) next_state = s3;//第三位正确
            else next_state = s2;
       s3 : if(confirm) next_state = s4;//按下 confirm 判断结果
            else next_state = s3;
       s4 : if(check) next_state = IDLE;
            else if (set)next_state = IDLE;
            else next_state = s4;
       default: next_state = IDLE;
    endcase
   end
end

always @(*) //
begin
  if(rst) next_state1 = IDLE1;
     else begin
     case (current_state1)
        IDLE1 : if(check) next_state1 = s01;//开始检测
              else  next_state1 = IDLE1;
        s01 : if(keyboard_en) next_state1 = s11;//第一位输入
             else next_state1 = s01;
        s11 : if(keyboard_en) next_state1 = s21;//第二位输入
             else next_state1 = s11;
        s21 : if(keyboard_en) next_state1 = s31;//第三位输入
             else next_state1 = s21;
        s31 : if(confirm) next_state1 = s41;//确认按钮按下
             else next_state1 = s31;
        s41 : if(check) next_state1 = IDLE1;//再次检测
             else next_state1 = s41;//
        default: next_state1 = IDLE1;
    endcase
   end
end
always @(posedge clk or posedge rst )
begin
      if(rst) checknum<= 12'b111111111111;
      else if(check)checknum<= 12'b111011101110;
//      else if(check)checknum <= {num1,num2,num3};
      else  if(keyboard_en&&next_state1 == s11)  checknum <=
{keyboard_num,checknum[11:4]};
      else  if(keyboard_en&&next_state1 == s21)  checknum <=
{checknum[11:8],keyboard_num,checknum[3:0]};
      else  if(keyboard_en&&next_state1 == s31)  checknum <=
{checknum[11:4],keyboard_num};
      else checknum <= checknum ;
end
```

●按键处理模块：

功能：

1. 接收按键输入信号，生成输入数字传输给主模块。
2. 承载功能状态机，根据接收的模块信号驱动状态机，从而使能相关模块。

端口：

输入	输出
时钟信号端口	键盘输入数字端口
主模块控制端口	键盘输入使能

变量	含义
keyboard_num	键盘按下数字
keyboard_en	键盘是否按下

主要设计代码：

```
counter #(CNT_THRESHOLD, 24) u_counter(//上板用
    .clk(clk),
    .rst(rst),
    .cnt_inc(1),
    .cnt_end(cnt_end)
);
always @(posedge clk, posedge rst) begin
    if (rst == 1) begin
        keyboard_num <= 0;
    end else if (key_posedge) begin
        if (key_posedge[0]) keyboard_num <= 'hd;
        else if (key_posedge[1]) keyboard_num <= 'hc;
        else if (key_posedge[2]) keyboard_num <= 'hb;
        else if (key_posedge[3]) keyboard_num <= 'ha;
        else if (key_posedge[4]) keyboard_num <= 'hf;
        else if (key_posedge[5]) keyboard_num <= 'h9;
        else if (key_posedge[6]) keyboard_num <= 'h6;
        else if (key_posedge[7]) keyboard_num <= 'h3;
        else if (key_posedge[8]) keyboard_num <= 'h0;
        else if (key_posedge[9]) keyboard_num <= 'h8;
        else if (key_posedge[10]) keyboard_num <= 'h5;
        else if (key_posedge[11]) keyboard_num <= 'h2;
        else if (key_posedge[12]) keyboard_num <= 'he;
        else if (key_posedge[13]) keyboard_num <= 'h7;
        else if (key_posedge[14]) keyboard_num <= 'h4;
        else if (key_posedge[15]) keyboard_num <= 'h1;
    end else begin
        keyboard_num <= 0;
    end
end
```

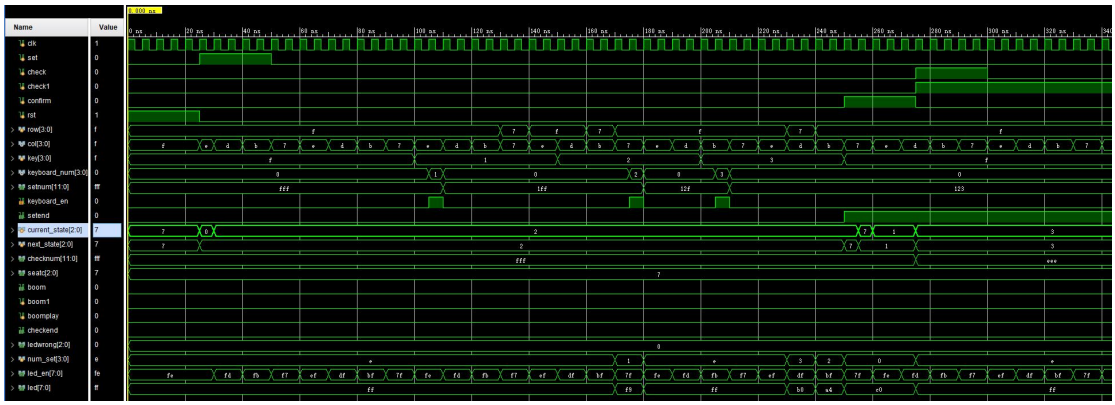
```
always @(posedge clk, posedge rst) begin
    if (rst == 1) begin
        keyboard_en <= 0;
    end else if (key_posedge) begin
        keyboard_en <= 1;
    end else begin
        keyboard_en <= 0;
    end
end
always @(posedge clk, posedge rst) begin
    if (rst == 1) key <= 0;
    else if (cnt_end) begin
        if (col[0] == 0) key[3:0]   <= ~row;//行输入信号；每列对应四
个 key 位
        if (col[1] == 0) key[7:4]   <= ~row;
        if (col[2] == 0) key[11:8]  <= ~row;
        if (col[3] == 0) key[15:12] <= ~row;
    end
End

assign cnt_end = (cnt == END);//每 15 个节拍一次？



always @(posedge clk, posedge rst) begin

    if (rst) cnt <= 0;

    else if (cnt_end) cnt <= 0;//结束就重来，

    else if (cnt_inc) cnt <= cnt + 1;//每个节拍加 1

End
```

调试报告

仿真波形截图及仿真分析

注：以下状态机若只提到一种操作改变状态则其余操作状态机状态不变

功能状态转移：



分析：

在按下 rst 即重置后，状态机初始为 IDLE 状态；

1. if（setend&&ledwright! = 1’b1）即设置完成且并非检测完成，状态机进入 s1 检验状态
2. else 进入 s0 状态；

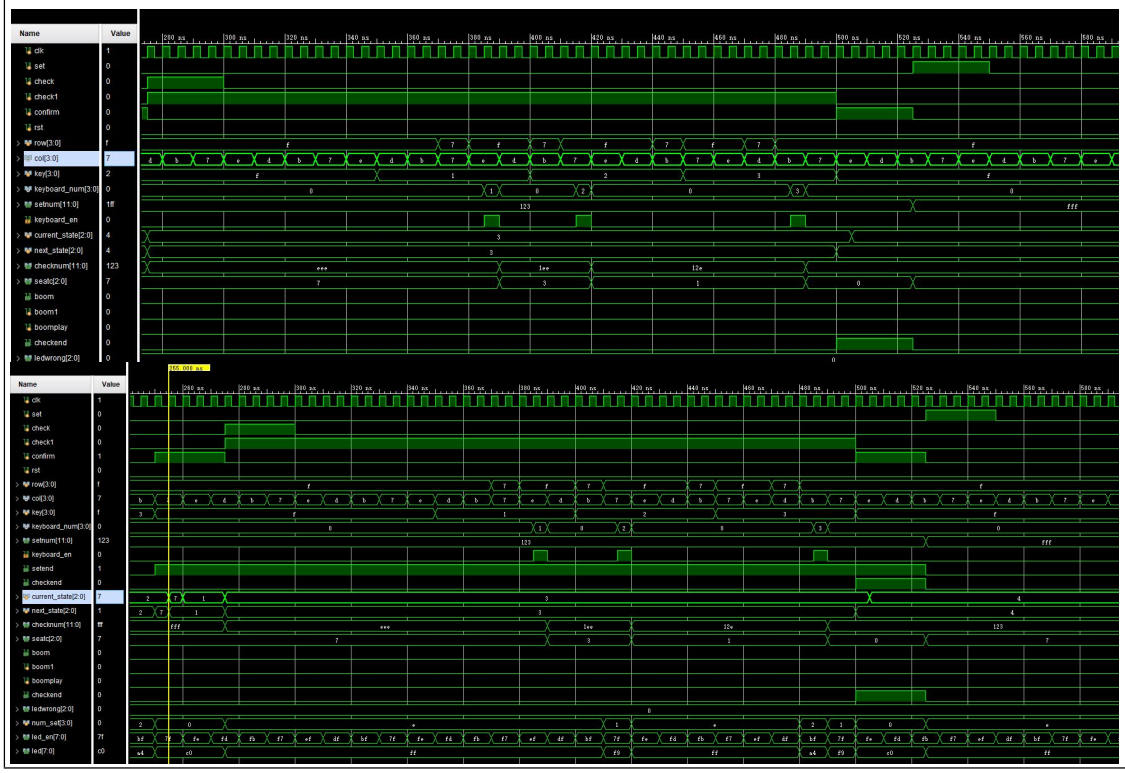
s0 状态，按下 set 进入 s2 密码设置状态；

s1 状态，按下 check 进入 s3 密码匹配判断状态（判断是否正确）；

s2 状态，setend = 1 即设置完成后进入 IDLE 初始状态；

s3 状态，checkend = 1 即密码匹配成功密码正确后进入 s4 解锁状态；

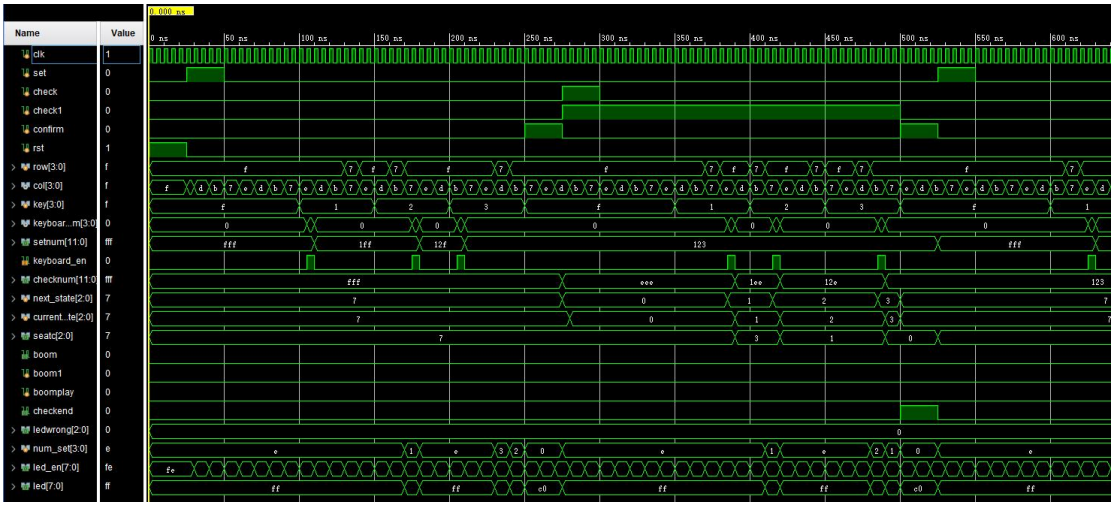
若 checkend != 1 即密码匹配失败，则验证失败，保持 s3 状态；



s4 状态，若 rst = 1，则进入 IDLE 初始状态；

对于密码锁定状态，在 s3 到 s4 间完成，详见后续密码匹配失败；

密码匹配成功：



分析：

在密码设置完成按下 confirm 后（黄线），此时密码匹配状态处于 IDLE 状态；

按下 check 使得 check1 信号为 1，状态机进入 s0 状态；

num1（setnum[11:8]）== keyboard_num 即第一位输入正确，状态机进入 s1 状态；

num2（setnum[7:4]） == keyboard_num 即第二位输入正确，状态机进入 s2 状态；

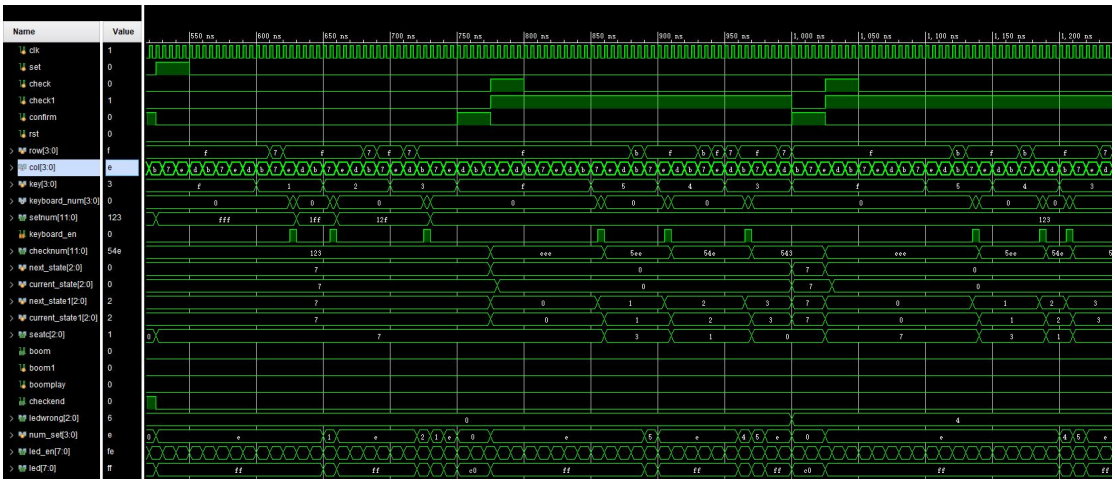
num3（setnum[3:0]） == keyboard_num 即第三位输入正确，状态机进入 s3 状态；

此时三位输入已完成，按下 confirm 进行结果判断，状态机进入 s4 状态；

若按下 set（更改密码）或者 check（错误时进行下一次验证）或 rst，状态机回到 IDLE 状态；

若进行其他操作，则状态机保持 s4 状态，灯控显示匹配成功；

密码匹配失败：



分析：

密码匹配失败机制采用双状态机：

状态机①：

在密码设置完成按下 confirm 后（黄线），此时密码匹配状态处于 IDLE 状态；

按下 check 使得 check1 信号为 1，状态机进入 s0 状态；

num1（setnum[11:8]）== keyboard_num 即第一位输入正确，状态机进入 s1 状态；

num2（setnum[7:4]） == keyboard_num 即第二位输入正确，状态机进入 s2 状态；

num3（setnum[3:0]） == keyboard_num 即第三位输入正确，状态机进入 s3 状态；

此时三位输入已完成，按下 confirm 进行结果判断，状态机进入 s4 状态；

若按下 set（更改密码）或者 check（错误时进行下一次验证）或 rst，状态机回到 IDLE 状态；

若进行其他操作，则状态机保持 s4 状态，灯控显示匹配成功；

状态机②：

在密码设置完成按下 confirm 后（黄线），此时密码匹配状态处于 IDLE 状态；

按下 check 使得 check1 信号为 1，状态机进入 s01 状态；

keyboard_en = 1 即第一位输入，状态机进入 s11 状态；

keyboard_en = 1 即第二位输入，状态机进入 s21 状态；

keyboard_en = 1 即第三位输入，状态机进入 s31 状态；

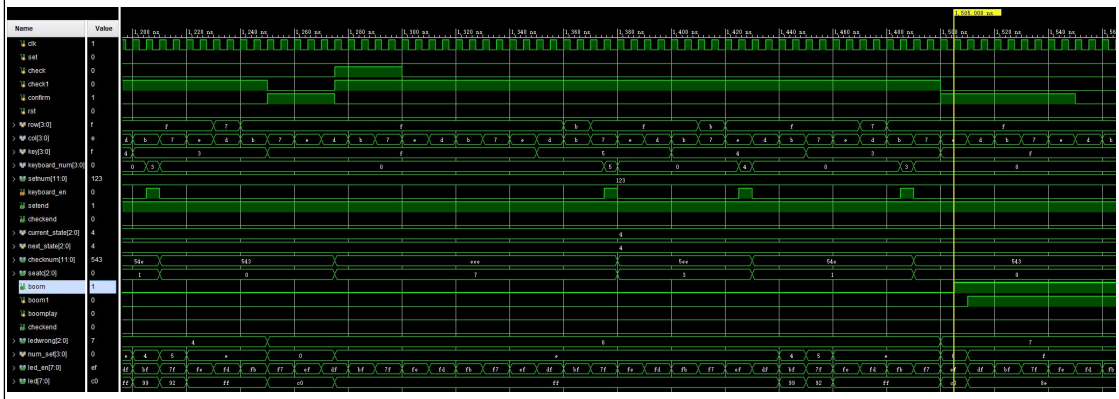
此时三位输入已完成，按下 confirm 进行结果判断，状态机进入 s41 状态；

若此时状态机①同步处于 s4 状态，则证明每位输入都正确，密码验证成功，若状态机①不同步处于 s4 状态，则输入了错误密码；

按下 set（更改密码）或者 check（错误时进行下一次验证）或 rst，状态机回到 IDLE 状态；

若进行其他操作，则状态机保持 s41 状态，灯控显示匹配失败及失败次数；

锁定：



在三次错误后 ledwrong = 3’d7 触发 boom = 1 即状态机锁定；

设计过程中遇到的问题及解决方法

●问题 1：密码如何同时验证正确和错误

描述：

原本 checking 模块中只有一个状态机，面临着若采用单位字符输入验证正确信号来驱动状态机，则输入错误时会滞留在当前状态，而这就造成了无法只在最后一个输入结束后通过 confirm 信号驱动下一步状态机，即状态机无法循环，又由于错误的密码组也有可能有几位正确，所以无法预判会滞留在哪个状态来防止 confirm 信号。

解决方法：

采用双状态机的形式，一个采用单位字符输入验证正确的信号来驱动，一个采用 keyboard_en 即键盘输入使能来驱动。这样做的好处是只需要检验双状态机是否同时处于最后一个输入结束即可，若同步，则按下 confirm 双状态机同时进入第五个状态即输入正确，若不认同不则只会有一个状态机进入第五个状态易于检测，解决上述问题。

●问题 2：想实现自加功能但 led_choose 码表中字符不足且 led_display_ctrl 难以切换显示

描述：

原本 led_choose 模块中只有一个码表，case 采用 4 位的 num_set 控制，故最高只有 16 位，然而被数字输入已经占有了 10 位，仅剩余 6 位十分有限且还有一位要用于表示空输入，再者切换显示字符也难以通过简单信号控制。

解决方法：

采用双码表的形式，仍旧利用 num_set 选择，但 led_choose 中，根据 boom 信号衍生的 boom1 信号选择码表，若 boom1 = 1 则 num_set 此时选择的是自定义的字符，有 16 位（除去空输入 15 位）可利用。

led_diaplay_ctrl 中则采用多层判断的信号来判断是否切换输入，若检测到 boom1（锁机信号）稍后又检测到 boomplay(计时器用以产生间隔)则切换码表，成功解决上述问题。

课程设计总结

总结：

此次设计着重看重模块化设计，通过前面的学习发现，数字逻辑设计实验对模块化的程度要求更高，语言编程中模块函数大多是为了简洁方便，但数字逻辑设计实验里的模块化决定了信号复杂程度，信号传输交互是否保持高效率，甚至决定了是否能够简单实现目标，当然也决定了代码的易操作和简介程度。

仍需改进的内容：

对于灯控显示模块设计的不够**简易**，仍存在大量相似代码。

收获：

在完成数字逻辑实验后，最大的收获就是对模块和接线这两种抽象概念有了更深的理解，明白了先设计再写的思维模式能够大幅节省时间和避免代码复杂，另外也对硬件没有那么陌生，了解了底层架构之后不再会觉得硬件深不可测，仍是可以通过简单信号堆叠构建成庞大的硬件板块，再轻松利用代码去控制。