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：计科五班

学生学号：210110504

学生姓名：廖雨泰

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：

1.按 S1 复位进入初始状态，数码管显示 00000000。

2.若已解锁，或未设置密码，按 S2 设置密码，通过 4*4 小键盘输入 3 位初始密码，输入的密码显示在显示屏上，且其余位不显示数字，输入满三位后按 S5 确定，成功后 GLD0 亮；

3.若已设置密码，按 S3 可以验证密码，通过小键盘逐位输入密码，输入的密码显示在显示屏上，且其余位不显示数字，输入满三位后按 S5 确定，如果正确则进入解锁状态，累计失败次数清零，不正确则累计失败次数+1，显示对应数量红灯，若累计 3 次则系统锁住，数码管显示 ffffffff，所有功能无效（除复位）。

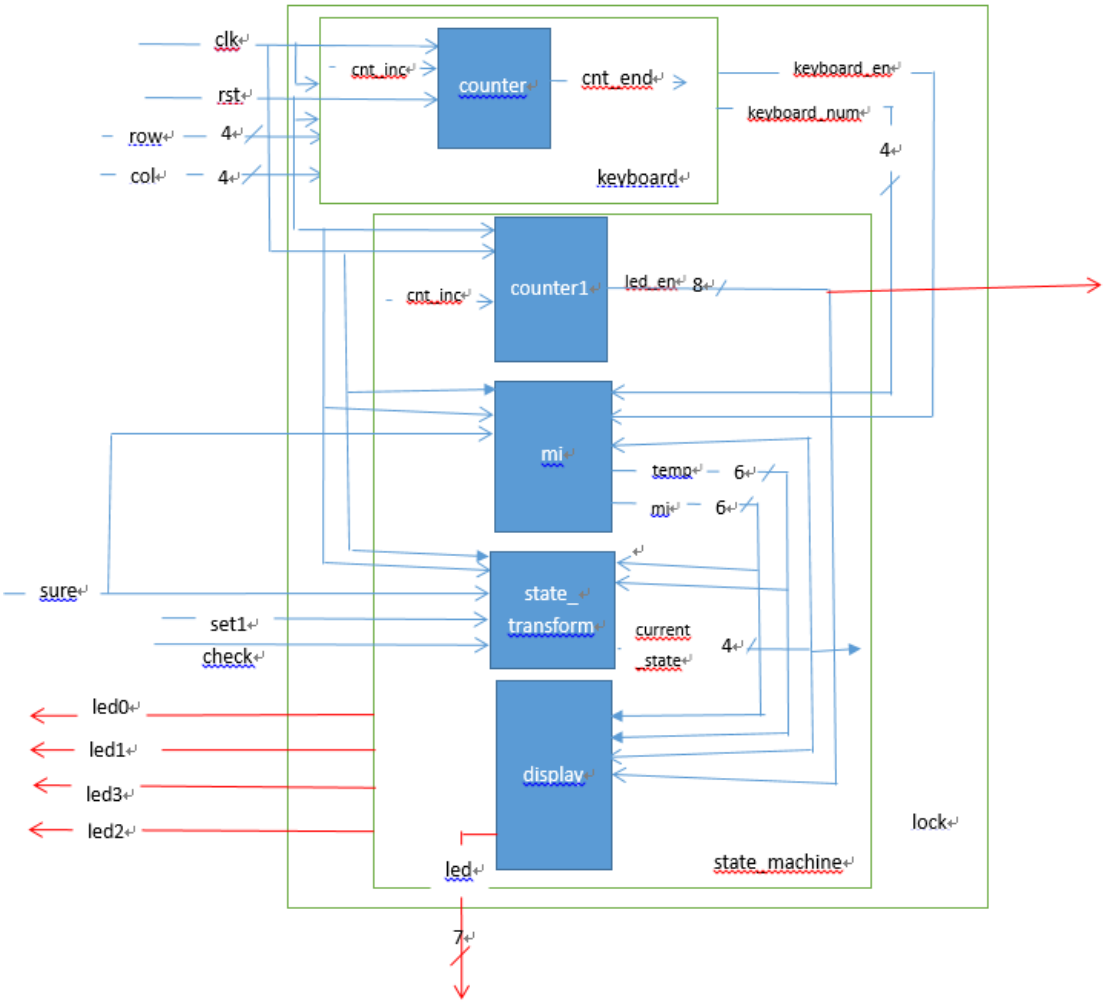
4.每次进入新状态，数码管清除上个状态的输入。

系统功能详细设计

用硬件框图描述系统主要功能及各模块之间的相互关系

要求有信号名、位宽、模块说明，可以参考下面的框图（仅为示例），须有密码处理模块、

数码管显示处理模块、按键处理模块，其他模块不限，不可用 vivado 的 RTL 截图。



模块说明：keyboard 为按键处理模块，其中的 counter 为计数器模块，

state_machine 为状态机模块，其中的 counter 为利用计数器制成的数码管使能信号发生模块，

mi 为处理密码和待验证的输入数字的密码处理模块，state_transform 为状态转换模块，display 为数码管显示处理模块

状态描述、状态转移图及状态编码，包括功能状态转移图、密码匹配的状态转移图；

状态描述：

IDLE：初始态，解锁状态，0000

s0：设置密码，0001

s1：密码设置完成，0010

s2：校验密码，0011

s3：密码错误一次，0100

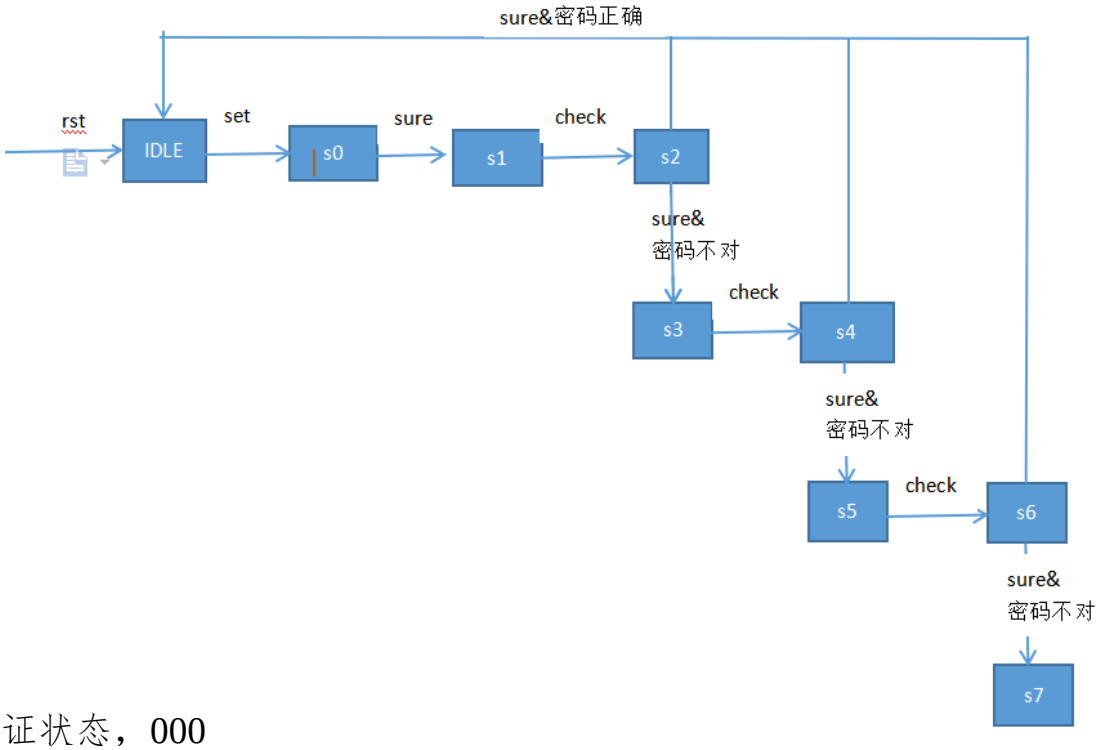
s4：校验密码，0101

s5：密码错误二次，0110

s6：校验密码，0111

s7：密码错误三次，1000

功能状态转移图



IDLE:初始态，未开始验证状态，000

A0:进入验证状态，001

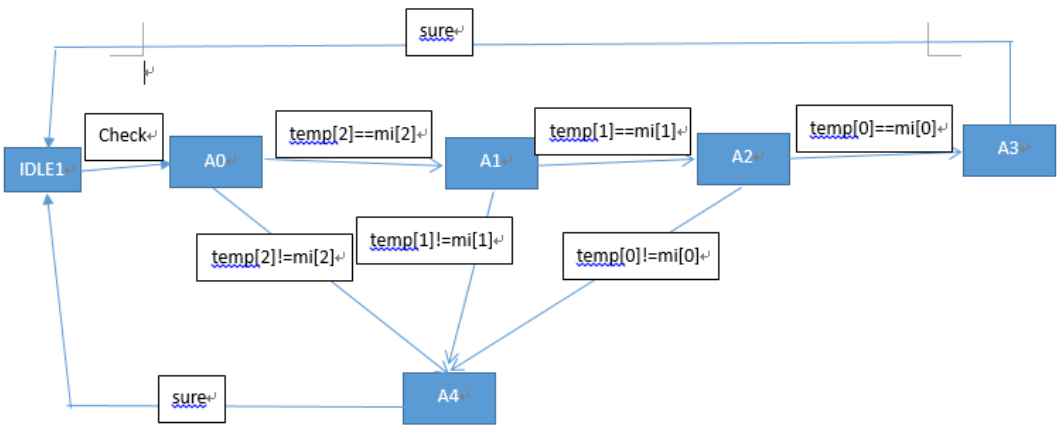
A1:匹配成功最高位，010

A2:匹配成功第二位，011

A3:匹配成功第三位，100,

A4:匹配失败，101

密码匹配状态转移图



各模块描述

包括模块功能，输入、输出端口、变量含义及主要设计代码

硬件框图有的模块都需体现

counter:

功能：计数器

输入：clk（时钟信号）,rst（复位信号）,cnt_inc（计数使能信号）

输出：cnt_end（一轮计数结束信号）

主要设计代码：

```
always @(posedge clk, posedge reset) begin
  if (reset) cnt <= 0;
  else if (cnt_end) cnt <= 0;
  else if (cnt_inc) cnt <= cnt + 1;
end
```

keyboard：

功能：按键处理

输入：clk（时钟信号）,rst（复位信号）

row(读取行输入信号)

输出：col(输出列扫描信号)keyboard_en(按键值有效信号)

keyboard_num(按键值)

主要设计代码：

```
always @(posedge clk, posedge reset) begin
```

```
    if (reset == 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 reset) begin
    if (reset == 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 reset) begin
    if (reset == 1)
        keyboard_led <= 0;
    else
        keyboard_led <= key;
end

always @(posedge clk, posedge reset) begin
    if (reset == 1)             col <= 4'b1111;
    else if (col ==  4'b1111) col <= 4'b1110;
    else if (cnt_end)           col <= {col[2:0], col[3]};
```

```
end


always @(posedge clk, posedge reset) begin
    if (reset == 1) key_r <= 0;
    else key_r <= key;
end


always @(posedge clk, posedge reset) begin
    if (reset == 1) key <= 0;
    else if (cnt_end) begin
        if (col[0] == 0) key[3:0]   <= ~row;
        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
```

counter1：

功能：产生数码管使能信号

输入：clk（时钟信号）,rst（复位信号）,cnt_inc（计数使能信号）

输出：led_en（数码管使能信号）

主要设计代码：

counter 源代码+

```
always @(posedge clk or posedge rst)
begin
if(rst) led_en <= 8'b11111110;
else if(cnt_end) led_en<={led_en[6:0],led_en[7]};
end
```

mi 模块：

功能：读入设定密码和验证密码

输入：rst（复位信号）,sure(确认信号),keyboard_num（按键值），clk（时钟信号）

keyboard_en（按键值有效信号）,current_state（当前状态）

输出：temp（待验证密码），mi(设定密码)

主要设计代码：

```
always @(posedge clk or posedge rst)                    //计数
begin
if(rst) n<=2'b10;
else if(sure)n<=2'b10;
else
if(((current_state==s2)|(current_state==s6)|(current_state==s4)|(current_state==s0))&keyboard_en)
n<=n-2'b01;
end
```

```
always @(posedge clk or posedge rst)
begin
if(rst) temp<=6'd0;
else if((current_state==s2)|(current_state==s6)|(current_state==s4))
        begin
        if((n==2'b10)&keyboard_en)
        begin
        case(keyboard_num)
        4'd1:temp[5:4]<=2'b01;
        4'd2:temp[5:4]<=2'b10;
        4'd3:temp[5:4]<=2'b11;
        default:temp[5:4]<=2'b00;
        endcase
        end
        else if((n==2'b01)&keyboard_en)
        begin
        case(keyboard_num)
        4'd1:temp[3:2]<=2'b01;
        4'd2:temp[3:2]<=2'b10;
        4'd3:temp[3:2]<=2'b11;
        default:temp[3:2]<=2'b00;
        endcase
```

```
        end
        else if((n==2'b00)&keyboard_en)
        begin
        case(keyboard_num)
        4'd1:temp[1:0]<=2'b01;
        4'd2:temp[1:0]<=2'b10;
        4'd3:temp[1:0]<=2'b11;
        default:temp[1:0]<=2'b00;
        endcase
        end
        end
else if(!((current_state==s2)|(current_state==s6)|(current_state==s4))) temp<=6'd0;
end

always @(posedge clk or posedge rst)
begin
if(rst) mi<=6'd0;
else if(current_state==s0)
        begin
        if((n==2'b10)&keyboard_en)
        begin
        case(keyboard_num)
```

```
4'd1:mi[5:4]<=2'b01;
4'd2:mi[5:4]<=2'b10;
4'd3:mi[5:4]<=2'b11;
default:mi[5:4]<=2'b00;
endcase
end
else if((n==2'b01)&keyboard_en)
begin
case(keyboard_num)
4'd1:mi[3:2]<=2'b01;
4'd2:mi[3:2]<=2'b10;
4'd3:mi[3:2]<=2'b11;
default:mi[3:2]<=2'b00;
endcase
end
else if((n==2'b00)&keyboard_en)
begin
case(keyboard_num)
4'd1:mi[1:0]<=2'b01;
4'd2:mi[1:0]<=2'b10;
4'd3:mi[1:0]<=2'b11;
default:mi[1:0]<=2'b00;
```

```
        endcase
        end
        end
else if(current_state==IDLE)mi<=6'd0;
end
```

state_transform 模块：

功能：状态转换

输入：mi(设定密码)，temp（验证密码），rst（复位信号），clk(时钟信号)

sure（确认按键），set（设置密码信号），check（检验密码信号）

输出：current_state(当前状态)

主要设计代码：

```
always @(posedge clk or posedge rst)
begin
if(rst) current_state<=IDLE;
else current_state<=next_state;
end

always @(*)
begin
case(current_state)
      IDLE:if(set) next_state=s0;
```

```
            else next_state=IDLE;
      s0:if(sure) next_state=s1;
            else next_state=s0;
      s1:if(check)next_state=s2;
            else next_state=s1;
      s2:if(sure&(temp==mi)) next_state=IDLE;
            else if(sure) next_state=s3;
            else next_state=s2;
      s3:if(check)next_state=s4;
            else next_state=s3;
      s4:if(sure&(temp==mi)) next_state=IDLE;
            else if(sure) next_state=s5;
            else next_state=s4;
      s5:if(check)next_state=s6;
            else next_state=s5;
      s6:if(sure&(temp==mi)) next_state=IDLE;
            else if(sure) next_state=s7;
            else next_state=s6;
      default:next_state=s7;
endcase
end
```

display 模块:

功能：产生数码管显示信号

输入：led_en（数码管使能信号）,current_state（当前状态），mi(密码),

temp（待验证密码）

输出：led（数码管显示信号）

主要设计代码：

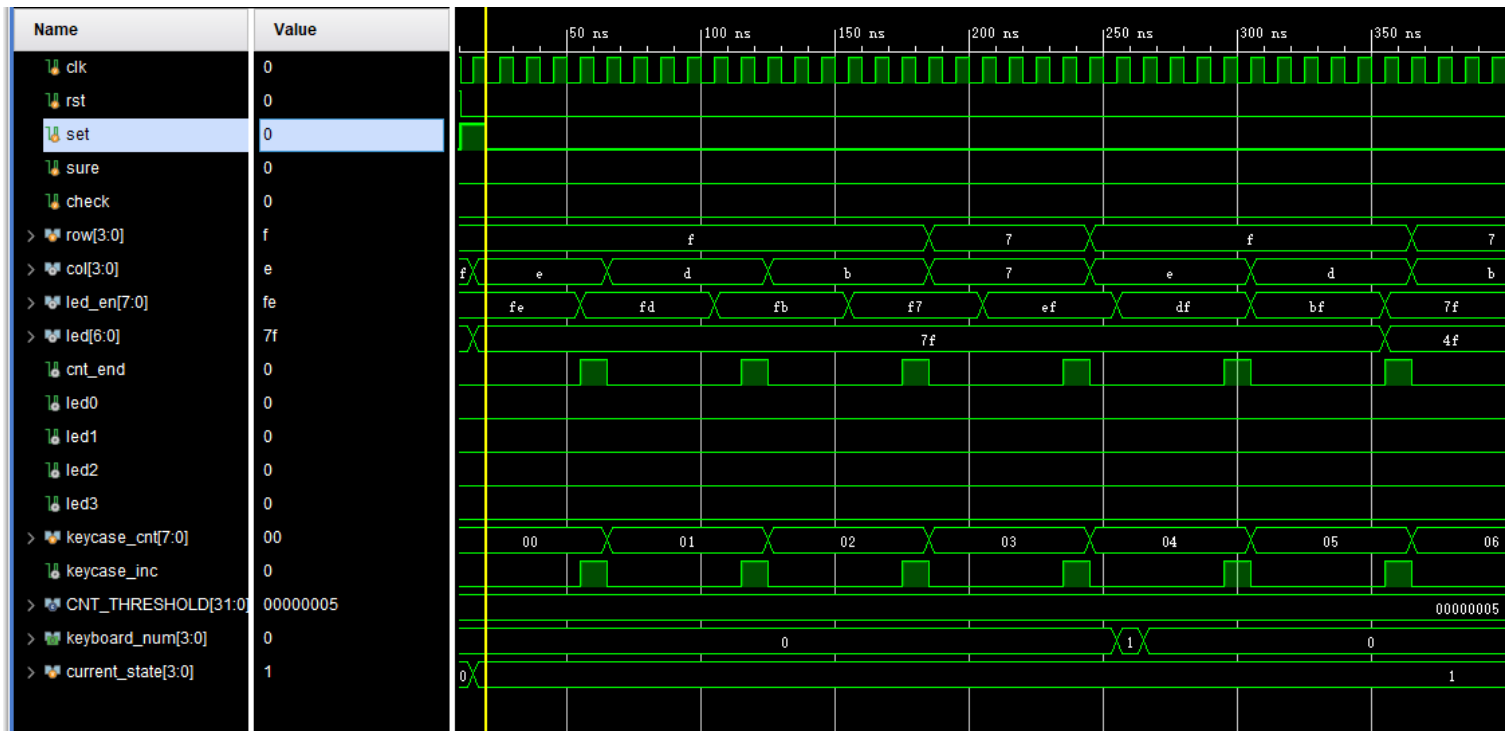
```
always@(*)
begin
case(current_state)
    IDLE:led=7'b0000001;
    s0:case(led_en)
        8'b11011111:case(mi[1:0])
                    2'b01:led=7'b1001111;
                    2'b10:led=7'b0010010;
                    2'b11:led=7'b0000110;
                    default:led=7'b1111111;
                    endcase
        8'b10111111:case(mi[3:2])
                    2'b01:led=7'b1001111;
                    2'b10:led=7'b0010010;
                    2'b11:led=7'b0000110;
                    default:led=7'b1111111;
```

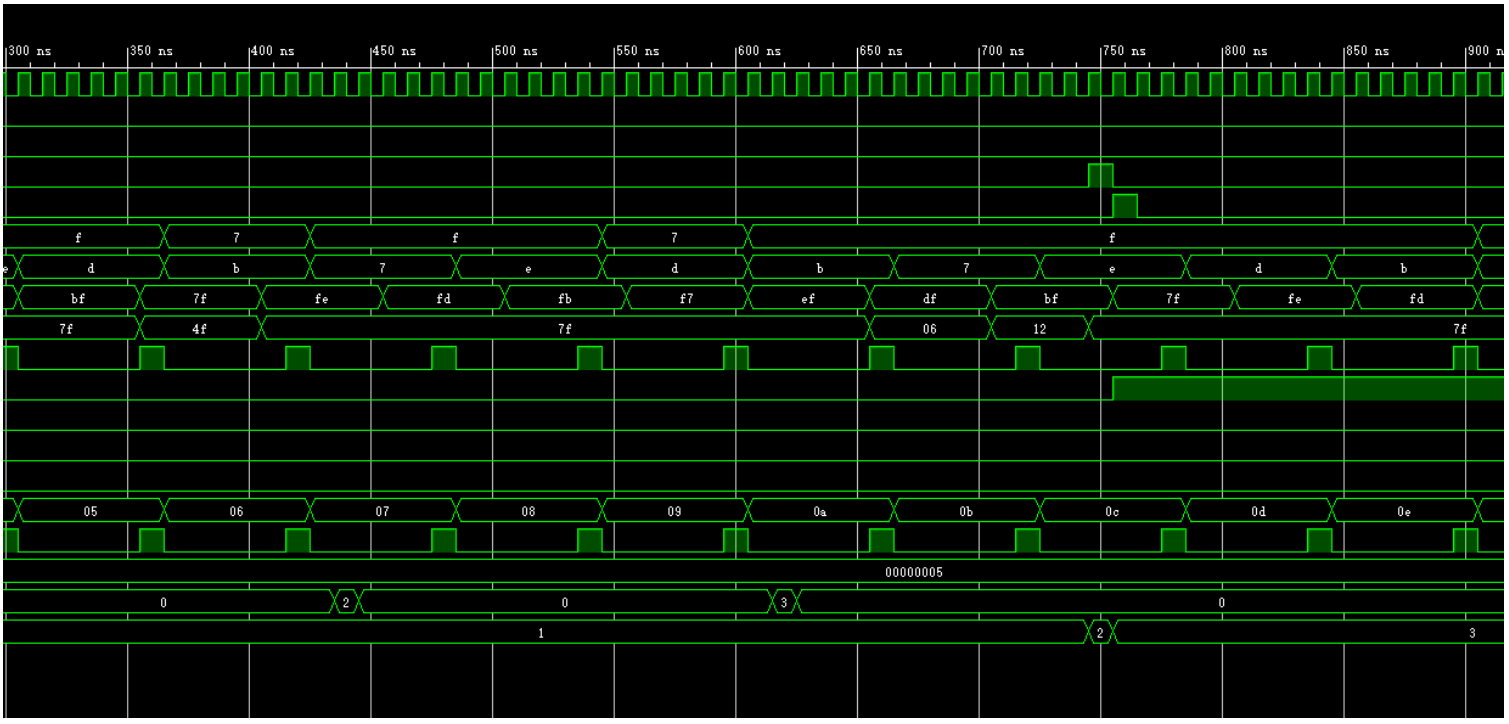
```
                endcase
        8'b01111111:case(mi[5:4])
                2'b01:led=7'b1001111;
                2'b10:led=7'b0010010;
                2'b11:led=7'b0000110;
                default:led=7'b1111111;
                endcase
        default:led=7'b1111111;
        endcase
s1:led=7'b1111111;
s5:led=7'b1111111;
s3:led=7'b1111111;
s7:led=7'b0111000;
default:case(led_en)
        8'b11011111:case(temp[1:0])
                2'b01:led=7'b1001111;
                2'b10:led=7'b0010010;
                2'b11:led=7'b0000110;
                default:led=7'b1111111;
                endcase
        8'b10111111:case(temp[3:2])
                2'b01:led=7'b1001111;
```

```
                2'b10:led=7'b0010010;
                2'b11:led=7'b0000110;
                default:led=7'b1111111;
                endcase
        8'b01111111:case(temp[5:4])
                2'b01:led=7'b1001111;
                2'b10:led=7'b0010010;
                2'b11:led=7'b0000110;
                default:led=7'b1111111;
                endcase
        default:led=7'b1111111;
        endcase
endcase
end
```

调试报告

仿真波形截图及仿真分析





密码设定：

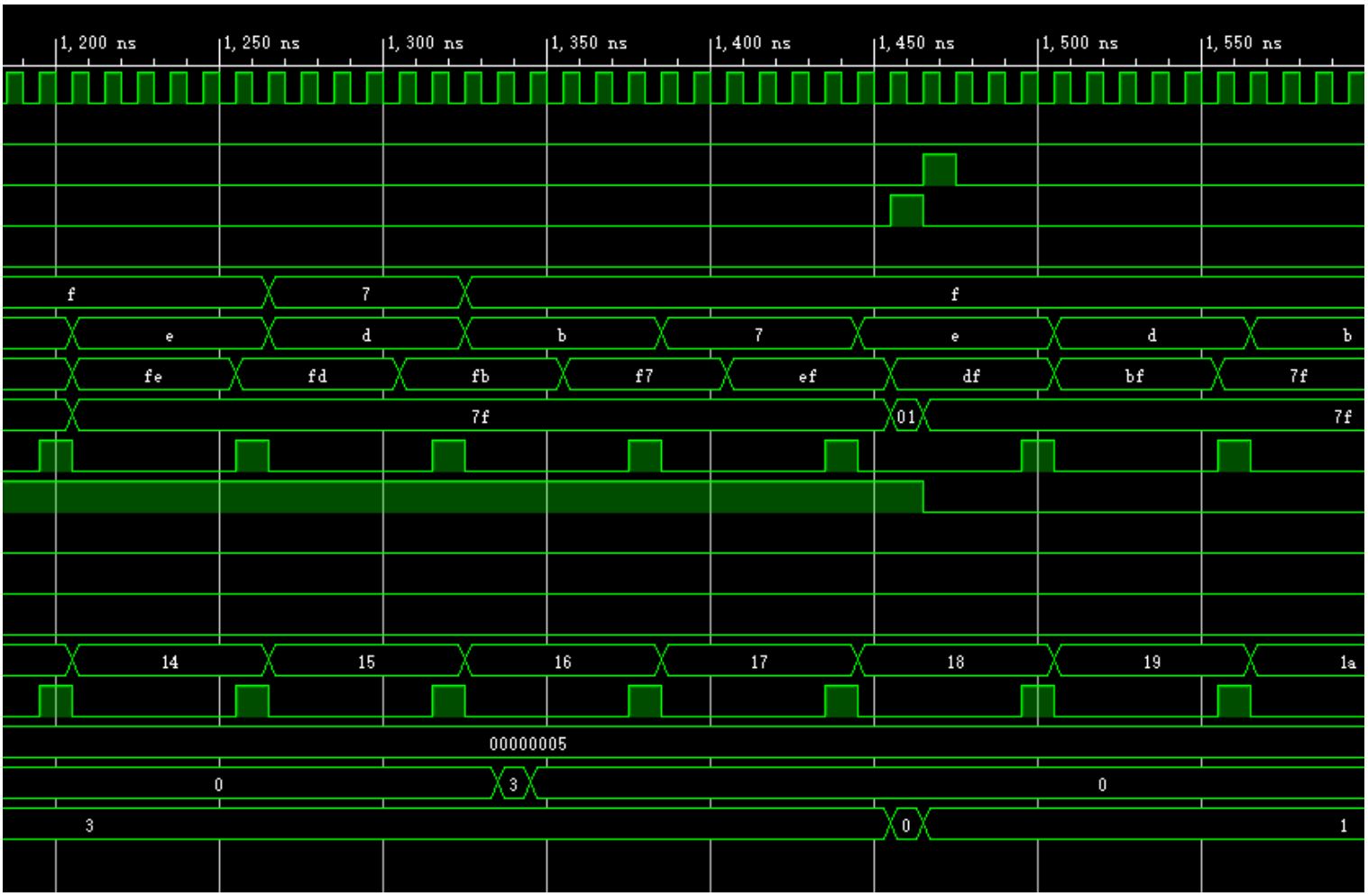
初始态：clk 为 0，rst 为 1，sure 为 0，set 为 0，check 为 0，current_state 为 0，符合预期

10ns 时，set 变为 1，current_state 变为 1，符合预期。

后续直到 725ns，row 和 col 不断变化，设定密码输入，745ns 时，sure 变为 1，current_state 变为 2，led0 变为 1，符合预期

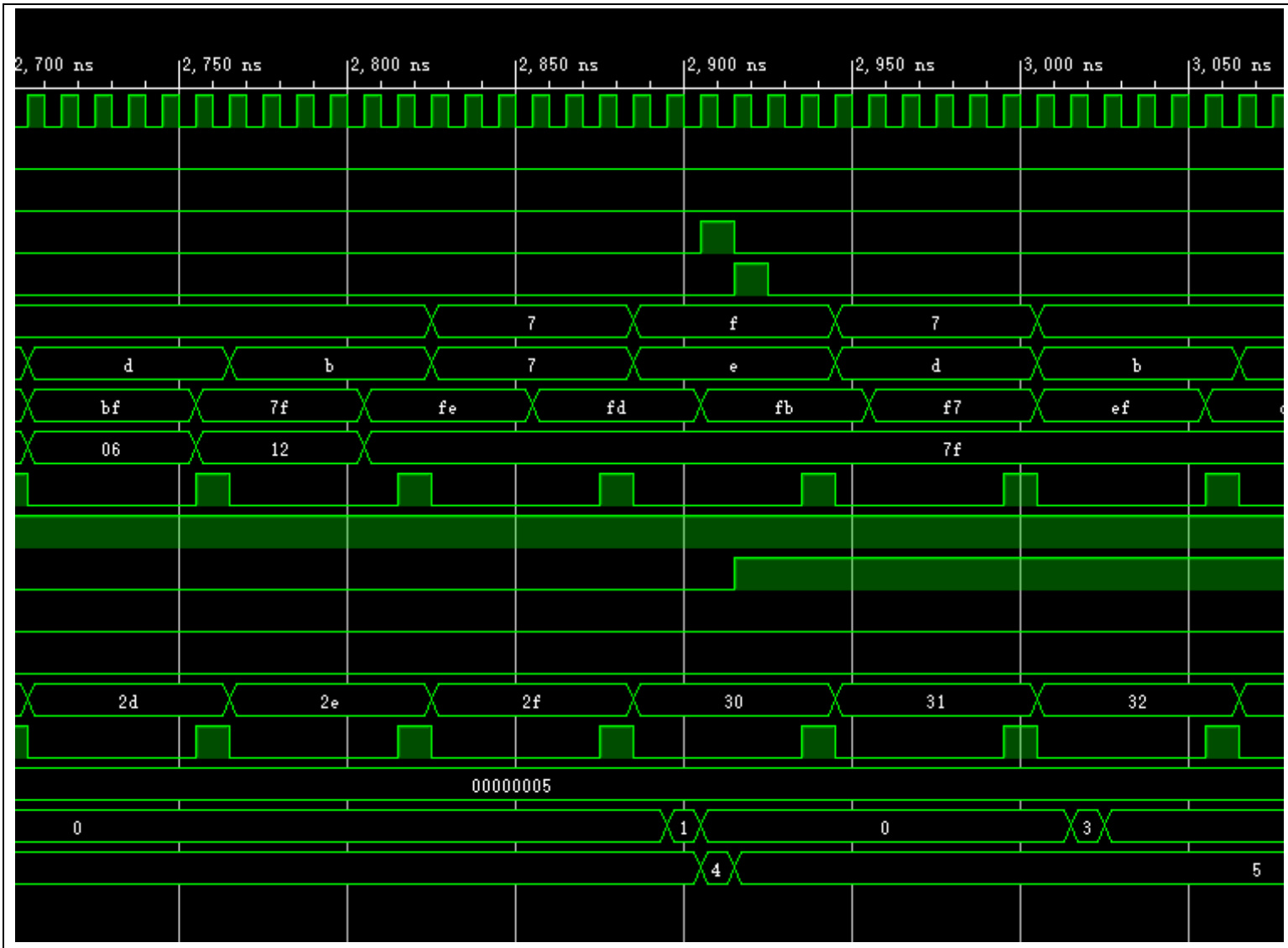
密码匹配：

755ns 时，check 变为 1，current_state 变为 3，符合预期



密码匹配成功：

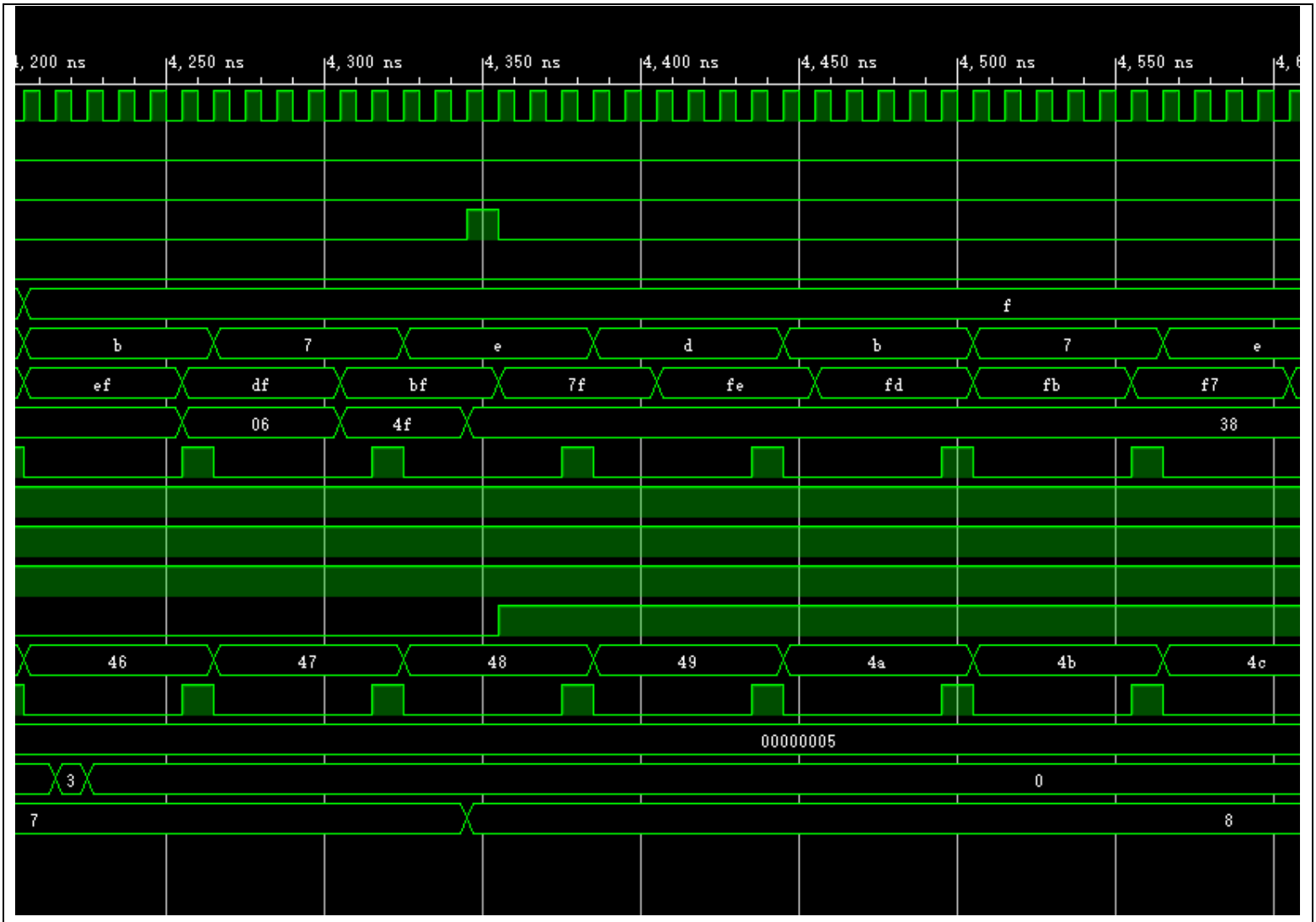
后续直到 1445ns,row 和 col 不断变化，检验密码输入，1455ns 时，sure 变为 1，解锁成功，current_state 变为 0，led0 也变为 0，符合预期



密码匹配失败：

2800ns 时，current_state 为 3。

2905ns 时，sure=1,密码错误，current_state 变为 4，led1 变为 1，符合预期



密码锁定：

4250ns 时，current_state 为 7。

4345ns 时，sure 为 1，密码错误，current_state 变为 8，led3 变为 1，锁定，符合预期。

设计过程中遇到的问题及解决方法

1.按键与 keyboard_led 对应不上，一个一个试验后对应好

2.按键输入时一个数字会输入很多次，将 keyboard_led 改为用 keyboard_en 和 keyboard_num

3.led1，led2,led3 进入另一状态时自动熄灭，改组合逻辑为时序逻辑

4.密码匹配未用状态机实现，重新写密码匹配

课程设计总结

包括设计的总结和还需改进的内容以及收获

总结：

1.时序逻辑和组合逻辑要分清什么时候用哪个

2.只有两个信号能以边沿触发的形式出现在敏感信号列表里

需要改进的内容：

对我来说工程量较大，后续几次作业几乎每次都要课下来机房 2-4 次，除了一次作业外，其他作业都对我有较大提升，所以我认为可以删掉中间一次较为简单的项目，将时间分配给其他几次作业，这样并不会影响学生的阶梯型提升

收获：

verilog 代码能力上升，对组合逻辑和时序逻辑有了更深的理解，代码更加规范。