目录

“密码锁”系统设计报告

通信 1701　　胡成成　41724260

一． 引言

移动通信网络实验课程以软硬件结合的方式进行实践教学。完成 FPGA 基础实验后并要求自主设计一个完整的程序，并在硬件平台上实现逻辑功能。本次课程报告的设计以“密码锁”展开，在 Ego1 硬件平台上实现可操作的逻辑。具体将包括到基本的时钟及复位按钮，调节按钮及相应的 LED 灯来改变当前模式，一组 LED 流水灯倒计时设计，七段数码管分段显示及实时修改数字密码并显示当前锁的状态等。

实验需要预先学习的知识包括 Verilog HDL 语法学习，Vivado 软件使用学习，Ego1 板卡相关端口学习。

通过本次课程的学习，下面将对“密码锁”逻辑程序的具体设计流程展开说明。具体包括实验环境，设计原理，设计方案，实验测试与结果分析，设计总结几个部分。

二． 实验环境

操作系统：Window 10；

设计软件：Vivado 2017.4；

硬件平台：Ego1；

三． 设计原理

1. Verilog 基础

- 硬件描述语言(HDL)是一种用形式化方法来描述数字电路和设计数字逻辑系统的语言。它可以使数字逻辑电路设计者利用这种语言来描述自己的设计思

想，然后利用电子设计自动化(EDA)工具进行仿真，再自动综合到门级电路，再用 ASIC 或 FPGA 实现其功能。

- HDL 设计流程：如图 1 所示

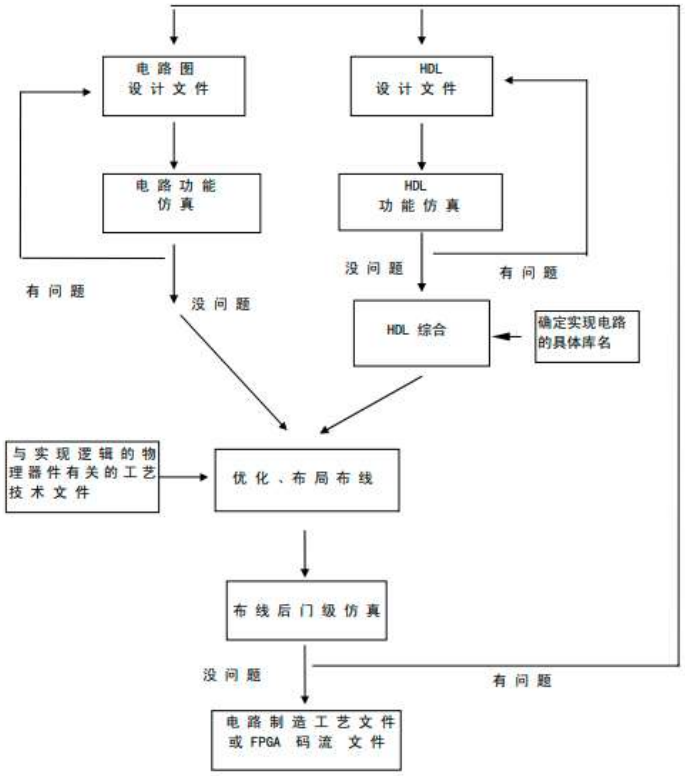

图 1 HDL 设计流程图

- Verilog 基本语法：一个复杂电路系统的完整 Verilog HDL 模型是由若干个 Verilog HDL 模块构成的，每一个模块又可以由若干个子模块构成。其中有些模块需要综合成具体电路，而有些模块只是与用户所设计的模块交互的现存电路或激励信号源。利用 Verilog HDL 语言结构所提供的这种功能就可以构造一个模块间的清晰层次结构来描述极其复杂的大型设计，并对所作设计的逻辑电路进行严格的验证。Verilog HDL 行为描述语言作为一种结构化和过程性的语言，其语法结构非常适合于算法级和 RTL 级的模型设计。这种行为描述语言具有以下功能：

- 可描述顺序执行或并行执行的程序结构。
- 用延迟表达式或事件表达式来明确地控制过程的启动时间。
- 通过命名的事件来触发其它过程里的激活行为或停止行为。
- 提供了条件、if-else、case、循环程序结构。
- 提供了可带参数且非零延续时间的任务(task)程序结构。
- 提供了可定义新的操作符的函数结构(function)。
- 提供了用于建立表达式的算术运算符、逻辑运算符、位运算符。
- Verilog HDL 语言作为一种结构化的语言也非常适合于门级和开关级的模型设计。

2. Vivado 软件设计平台

- 概述：Vivado 设计分为 Project Mode 和 Non-project Mode 两种模式，一般简单设计中，我们常用的是 Project Mode。
- 使用 Vivado 设计的基本流程：如图 2 所示

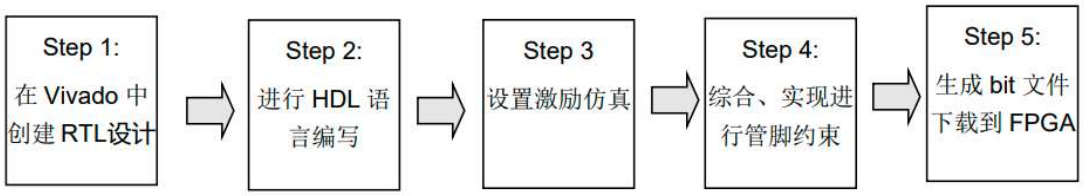

图 2 Vivado 设计流程

3. Ego1 硬件实验平台

EGO1 是依元素科技基于 Xilinx Artix-7 FPGA 研发的便携式数模混合基础教学平台。EGO1 配备的 FPGA (XC7A35T-1CSG324C)具有大容量高性能等特点，能实现较复杂的数字逻辑设计；在 FPGA 内可以构建 MicroBlaze 处理器系统，可进行 SoC 设计。该平台拥有丰富的外设，以及灵活的通用扩展接口。

- 系统时钟：EGO1 搭载一个 100MHz 的时钟芯片，输出的时钟信号直接与 FPGA 全局时钟输入引脚（P17）相连。若设计中还需要其他频率的时钟，可以采用 FPGA 内部的 MMCM 生成。
- 按键：两个专用按键分别用于逻辑复位 RST（S6）和擦除 FPGA 配置 PROG（S5），当设计中不需要外部触发复位时，RST 按键可以用作其他逻辑触发功能。
- 开关：开关包括 8 个拨码开关和一个 8 位 DIP 开关，管脚约束如图 3 所示

名称	原理图标号	FPGA IO PIN
SW0	SW_0	P5
SW1	SW_1	P4
SW2	SW_2	P3
SW3	SW_3	P2
SW4	SW_4	R2
SW5	SW_5	M4
SW6	SW_6	N4
SW7	SW_7	R1
SW8	SW_DIP0	U3
	SW_DIP1	U2
	SW_DIP2	V2
	SW_DIP3	V5
	SW_DIP4	V4
	SW_DIP5	R3
	SW_DIP6	T3
	SW_DIP7	T5

图 3 开关管脚约束

- LED 灯：LED 在 FPGA 输出高电平时被点亮。管脚约束如图 4 所示

名称	原理图标号	FPGA IO PIN	颜色
D0	LED0	F6	Green
D1	LED1	G4	Green
D2	LED2	G3	Green
D3	LED3	J4	Green
D4	LED4	H4	Green
D5	LED5	J3	Green
D6	LED6	J2	Green
D7	LED7	K2	Green
D8	LED8	K1	Green
D9	LED9	H6	Green
D10	LED10	H5	Green
D11	LED11	J5	Green
D12	LED12	K6	Green
D13	LED13	L1	Green
D14	LED14	M1	Green
D15	LED15	K3	Green

图 4 LED 管脚约束

- 七段数码管：数码管为共阴极数码管，即公共极输入低电平。共阴极由三极管驱动，FPGA 需要提供正向信号。同时段选端连接高电平，数码管上的对应位置才可以被点亮。因此，FPGA 输出有效的片选信号和段选信号都应该是高电平。管脚约束如图 5 所示

名称	原理图标号	FPGA IO PIN
A0	LED0_CA	B4
B0	LED0_CB	A4
C0	LED0_CC	A3
D0	LED0_CD	B1
E0	LED0_CE	A1
F0	LED0_CF	B3
G0	LED0_CG	B2
DP0	LED0_DP	D5
A1	LED1_CA	D4
B1	LED1_CB	E3
C1	LED1_CC	D3
D1	LED1_CD	F4
E1	LED1_CE	F3
F1	LED1_CF	E2
G1	LED1_CG	D2
DP1	LED1_DP	H2
DN0_K1	LED_BIT1	G2
DN0_K2	LED_BIT2	C2
DN0_K3	LED_BIT3	C1
DN0_K4	LED_BIT4	H1
DN1_K1	LED_BIT5	G1
DN1_K2	LED_BIT6	F1
DN1_K3	LED_BIT7	E1
DN1_K4	LED_BIT8	G6

图 5 七段数码管管脚约束

四． 设计方案

1. 密码锁端口预设

- 对模块管脚定义与预设：

管脚	类型	位宽	功能
number1[3:0]	input	4	前四位数码管数字信息
number2[3:0]	input	4	后四位数码管数字信息
numbit[7:0]	input	8	数字显示位信息
ledbit[7:0]	output	8	数码管显示位信息
ledshow[6:0]	output	7	前四位数码管每段信息
ledshow2[6:0]	output	7	后四位数码管每段信息
sys_clk	input	1	总时钟
rst_n	input	1	重置清零信号
left_button	input	1	左键选中情况
right_button	input	1	右键选中情况
up_button	input	1	上键选中情况
down_button	input	1	下键选中情况
middle_button	input	1	中键选中情况
lock_status	input	1	是否为锁住状态
read_status	input	1	是否为检验密码状态
load_status	input	1	是否为修改密码状态
is_jmp[7:0]	input	8	当前选中位（实现闪烁）

- 创建 lock.v 的 source 文件, 创建 lock 模块初始化硬件输入输出需要的端口:

```
module lock(
input sys_clk,   //总时钟
input rst_n,      //reset 重置
//五个按钮：上下左右中
input up_button,
input down_button,
input left_button,
input right_button,
input middle_button,
input mode,     //SW7 开关控制模式
output wire ledlow,         //SW7 对应的 LED 灯
output wire [7:0] ledbit,       //数码管显示位信息
output wire [6:0] ledshow,     //前四位数码管每段信息
output wire [6:0] ledshow2,   //后四位数码管每段信息
output reg [7:0] leddown         //8 个小 LED 灯：流水计时
);
```

- 定义状态端口等逻辑端口:

```
reg lock_status;       //是否为锁住状态
wire load_status;     //是否为修改密码状态
wire read_status;     //是否为检验密码状态
reg succ_status;       //是否成功解锁
reg [7:0] is_jmp;     //当前选中位（实现闪烁）
reg [7:0] choose;     //指针
```

- 定义密码与显示变量:

```
reg [7:0] numbit;       //数字显示位信息
reg [3:0] number1;     //前四位数码管数字信息
reg [3:0] number2;     //后四位数码管数字信息
```

```
//锁密码字段 4 位
reg [3:0] pwd1;
reg [3:0] pwd2;
reg [3:0] pwd3;
reg [3:0] pwd4;
reg [3:0] pwds;          //当前输入密码
reg [3:0] number_1;
reg [3:0] number_2;
reg [3:0] number_3;
reg [3:0] number_4;
reg [3:0] number_5;
reg [3:0] number_6;
reg [3:0] number_7;
reg [3:0] number_8;
```

2. 密码锁状态分类

- 建立模式与状态联系：

```
assign ledlow = mode;
assign read_status = ~mode;
assign load_status = mode;
```

- 密码锁状态分为待解锁，解锁超时失败和解锁成功三个状态，分别用字符 CALL，FAIL 和-HCC（个人姓名缩写，胡成成）三个字符在数码管上展示。其中字符代号在后面可视化设计中给出。

```
//成功解锁  -HCC
    if(succ_status == 1'b1)
    begin
        number_5 <= 4'd14;
        number_6 <= 4'd12;
```

```
            number_7 <= 4'd13;
            number_8 <= 4'd13;
        end
        else
        begin
             //未在规定时间内解锁    Fail
            if(lock_status == 1'b1)
            begin
                number_5 <= 4'd15;
                number_6 <= 4'd10;
                number_7 <= 4'd1;
                number_8 <= 4'd11;
            end
            else
            //解锁中  CALL
            begin
                number_5 <= 4'd13;
                number_6 <= 4'd10;
                number_7 <= 4'd11;
                number_8 <= 4'd11;
            end
        end
```

3. 密码锁开锁逻辑

- 左右选择相应的数码管位置

```
if(choose == 8'd0)
        begin
            if(right_timer == 64'd9_999_999)
            begin
                choose = 8'd1;
```

```
        end
    end
    else if(choose == 8'd1)
    begin
        if(left_timer == 64'd9_999_999)
        begin
            choose = 8'd0;
        end
        else if(right_timer == 64'd9_999_999)
        begin
            choose = 8'd2;
        end
    end
    else if(choose == 8'd2)
    begin
        if(left_timer == 64'd9_999_999)
        begin
            choose = 8'd1;
        end
        else if(right_timer == 64'd9_999_999)
        begin
            choose = 8'd3;
        end
    end
    else if(choose == 8'd3)
    begin
        if(left_timer == 64'd9_999_999)
        begin
            choose = 8'd2;
        end
```

```
    end
```

- 上下调节数码管对应位置的数字（0~9），以数码管四位密码第一位为例

```
if(choose == 4'd0)
        begin
            if(up_timer == 64'd9_999_999)
            begin
                if(number_1 < 4'd9)
                begin
                    number_1 = number_1 + 1'b1;
                end
            end
            if(down_timer == 64'd9_999_999)
            begin
                if(number_1 > 4'd0)
                begin
                    number_1 = number_1 - 1'b1;
                end
            end
        end
```

- 初始化和默认密码设置（初始密码 2020）

```
begin
        pwd1 <= 4'd2;
        pwd2 <= 4'd0;
        pwd3 <= 4'd2;
        pwd4 <= 4'd0;
        pwds <= 4'b0000;
        succ_status <= 1'b0;
    end
```

- 密码正确解锁，此时显示“-HCC”

```
if(lock_status == 1'b0)
        begin
            //读取状态  密码正确解锁
            if(read_status == 1'b1)
            begin
                if(middle_timer == 64'd9_999_999)
                begin
                    if(pwds == 4'b1111)
                    begin
                        succ_status <= 1'b1;
                    end
                    else
                    begin
                        succ_status <= 1'b0;
                    end
                end
            end
```

-

4. 密码锁修改密码设计

- 修改密码逻辑

```
//非读取状态
            else
            begin
                if(load_status == 1'b1)
                begin
                    //加载状态下  设置密码----------------------------------------
                    if(middle_timer == 64'd9_999_999)
```

```
                    begin
                        pwd1 <= number_1;
                        pwd2 <= number_2;
                        pwd3 <= number_3;
                        pwd4 <= number_4;

                        succ_status <= 1'b0;
                    end
                end
            end
```

5. 可视化设计

- 密码锁开锁倒计时采用流水灯式倒计时，倒计时完毕未成功解锁，密码锁将被锁死，显示 FAIL

```
if(~rst_n)
begin
    leddown <= 8'b11111111;
    lock_status = 1'b0;
end
//流水灯计时
else if(read_status == 1'b1)
begin
    if(led_timer == 64'd199_999_999) //49_999_999
        begin
            leddown <= 8'b11111110;
        end
        else if(led_timer == 64'd399_999_999)//99_999_999
        begin
            leddown <= 8'b11111100;
```

```
end
else if(led_timer == 64'd599_999_999)//149_999_999
begin
    leddown <= 8'b11111000;
end
else if(led_timer == 64'd799_999_999)//199_999_999
begin
    leddown <= 8'b11110000;
end
else if(led_timer == 64'd999_999_999)//249_999_999
begin
    leddown <= 8'b11100000;
end
else if(led_timer == 64'd1_199_999_999)//299_999_999
begin
    leddown <= 8'b11000000;
end
else if(led_timer == 64'd1_399_999_999)//349_999_999
begin
    leddown <= 8'b10000000;
end
else if(led_timer == 64'd1_599_999_999)//399_999_999
begin
    leddown <= 8'b00000000;
end
else if(led_timer == 64'd1_799_999_999)//449_999_999
begin
    leddown <= 8'b11111111;
    lock_status = 1'b1;       //超时锁住
end
```

```
        end
        //read_status=0 ,设置密码
        else
        begin
            leddown <= 8'b10101010;
        end
```

- 数码管字符设计：其中 0-9 字符不在这里列出，给出自定义字符的设计代号和显示模式如下

```
if(num == 4'd10)
    begin
        digital = 7'b1111110;//A
    end
    if(num == 4'd11)
    begin
        digital = 7'b0100101;//L
    end
    if(num == 4'd12)
    begin
        digital = 7'b0111110;//H
    end
    if(num == 4'd13)
    begin
        digital = 7'b1100101;//C
    end
    if(num == 4'd14)
    begin
        digital = 7'b0001000;//-
    end
    if(num == 4'd15)
```

```
    begin
        digital = 7'b1101100;//F
    end
```

- 选中数码管字符闪烁

```
begin
    if(choose == 8'd0)
    begin
        is_jmp <= 8'b10000000;
    end
    else if(choose == 8'd1)
    begin
        is_jmp <= 8'b01000000;
    end
    else if(choose == 8'd2)
    begin
        is_jmp <= 8'b00100000;
    end
    else if(choose == 8'd3)
    begin
        is_jmp <= 8'b00010000;
    end
end
```

- 时钟设计，代码过长在这里不在仔细阐述。

6. 管脚约束

端口名称	输入输出方向	管脚号
sys_clk	input	P17
rst_n	input	P15

up_button	input	U4
down_button	input	R17
left_button	input	V1
right_button	input	R11
middle_button	input	R15
mode	input	P5
ledlow	output	F6
ledbit[0]	output	G6
ledbit[1]	output	E1
ledbit[2]	output	F1
ledbit[3]	output	G1
ledbit[4]	output	H1
ledbit[5]	output	C1
ledbit[6]	output	C2
ledbit[7]	output	G2
ledshow[0]	output	B1
ledshow[1]	output	A3
ledshow[2]	output	A1
ledshow[3]	output	B2
ledshow[4]	output	A4
ledshow[5]	output	B3
ledshow[6]	output	B4

ledshow2[0]	output	F4
ledshow2[1]	output	D3
ledshow2[2]	output	F3
ledshow2[3]	output	D2
ledshow2[4]	output	E3
ledshow2[5]	output	E2
ledshow2[6]	output	D4
leddown[0]	output	K3
leddown[1]	output	M1
leddown[2]	output	L1
leddown[3]	output	K6
leddown[4]	output	J5
leddown[5]	output	H5
leddown[6]	output	H6
leddown[7]	output	K1

五． 实验测试与结果分析

1. 密码锁开锁测试

- 准备解锁时，显示 CALL 字符等待输入，如图 6 所示

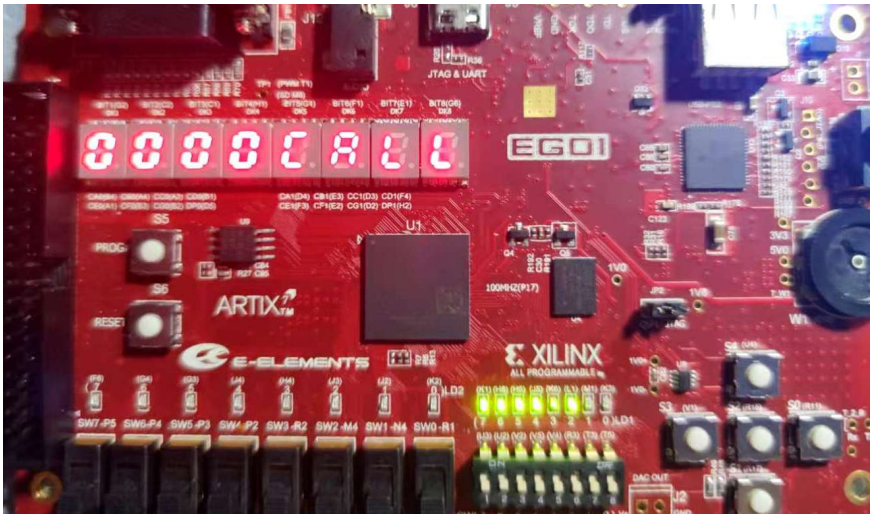

图 6 准备解锁状态

- 在规定 20 秒内，输入正确密码并确认，成功解锁，返回-HCC，如图 7 所示

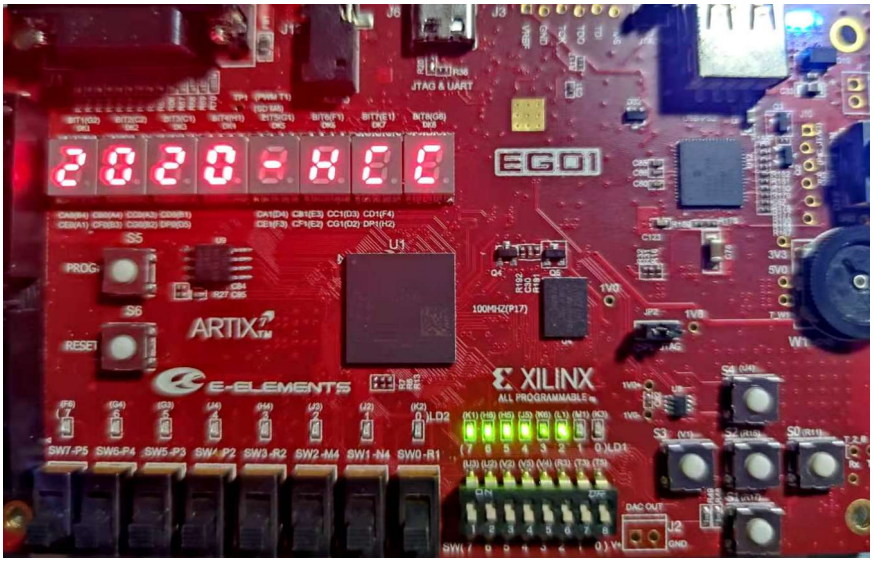

图 7 输入正确密码状态

2. 修改密码测试

- 调节按钮 SW7（左下角按钮上拉，相应 LED 点亮，流水灯切换为间隔亮灯），此时调节每位密码数值设置锁的密码，如图 8 所示设为 1234。

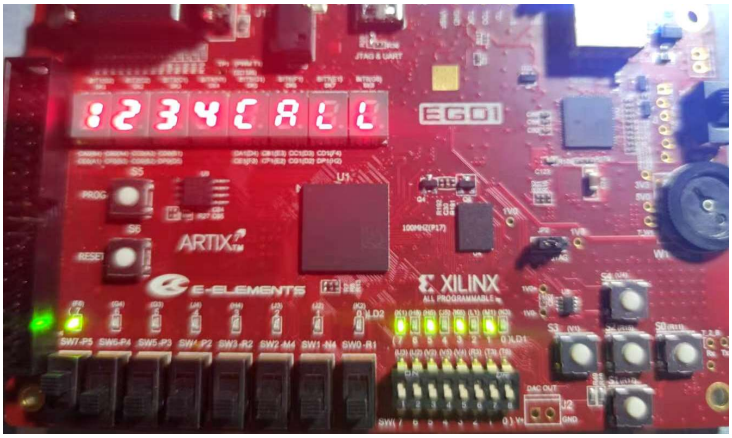

图 8 修改密码状态

- 设置完后返回正常解锁模式测试，输入 1234 成功解锁，返回-HCC

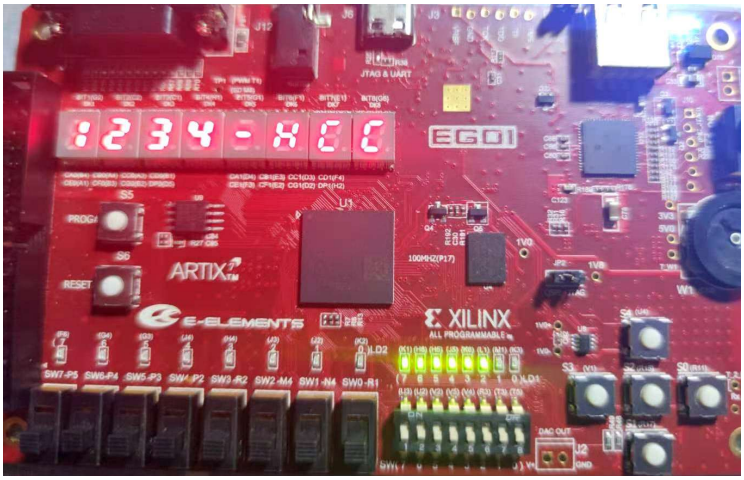

图 9 修改密码后解锁状态

3. 密码锁超时锁死测试

- 超过 20 秒未输入正确密码，密码锁将锁死，返回 FAIL，如图 10 所示

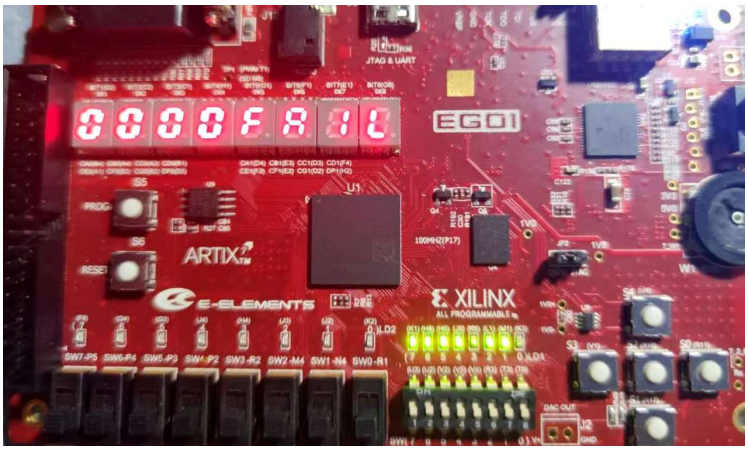

图 10 解锁超时状态

六． 设计总结

本次实践通过对 FPGA 的设计，实现了密码锁应用的逻辑，包括输入密码时的 CALL 状态，正确输入密码-HCC 状态，超时未正确输入密码 FAIL 状态，修改密码切换 MODE 等。学习 Verilog 的基本语法，软件操作使用 Vivado，在实现密码锁设计的工程中，也遇到了一些难题，例如实现按键延时响应，输入密码位闪烁，自定义字符的设计等。通过查阅资料，调试最终的程序，亲身操作实践，同时还学到了很多的课外知识，不仅仅局限于课本的理论，将实践与理论结合起来的时候，才是真的掌握了所学的知识。

课内课外我们都应该有自己的一些思考和想法，作为通信工程的一名大学生，应该严格要求自己，扩展自己的视野，多多实践才是王道。

七． 致谢

感谢杨裕亮老师 4 周的移动通信网络实验课程的教诲；让我们知道身为通信人，不能满足于课本知识的学习，要更多地去实践，通过编程验证自己的想法，设计自己的点子。