8.密码锁

8.1 实验目的

利用盘古 PGX-Nano 板卡上的按键，拨码开关以及数码管实现一种简单的密码锁；

8.2 实验要求

利用拨码开关设置密码，使用按键输入开锁密码。当开锁密码与设定密码相同时开锁成功，数码管显示 8888，密码错误时显示 7777。

SW0~SW3 设置 2 位数密码，每两位设置一位密码，SW[1:0]设置第一位数据对应的二进制数值，SW[3:2]设置第二位数据对应的二进制数值。所以密码是由 0，1，2，3 组成的四位数。

S1-S0 按键作为密码输入，按键按一下数字加 1，数字由数码管显示，数字在 0，1，2，3 中循环。

S2 作为确认按键，按下 S2，输入的密码与设置的密码比对，如相同则显示 8888，若不同则显示 7777。 按下 S3 清零，按下后数码管显示 0000，可以重新输密码。

8.3 实验原理

原理上与前一个章节的序列检测是类似的，在前一个实验的基础上有了一些延伸；

序列对比的位宽发生改变，单个数据占 2bit，一个按键控制输入密码数据设置为 2bit 即可；对比与重新开始在此实验用两个按键实现，一个确认对比，一个清空结果；

8.4 实验源码（完整源码查看 demo 源文件）

根据需求我们需要如下三个子模块：

①按键控制模块；

1、对 4 个按键输入信号均做消抖处理

2、S3 和 S2 取下降沿输出

3、S[1：0]以下降沿来变更各自的输入密码，每次数字加 1（0～3 循环，2bit 即可）

②数码管显示模块；

显示状态有两种：

密码输入状态：

1、上电默认状态； 2、S3 下降沿触发进入重置状态；3、实时显示 2 位输入密码；

密码验证状态：

1、S2 下降沿触发进入；

2、显示密码验证结果，正确则显示 8888，错误则显示 7777；

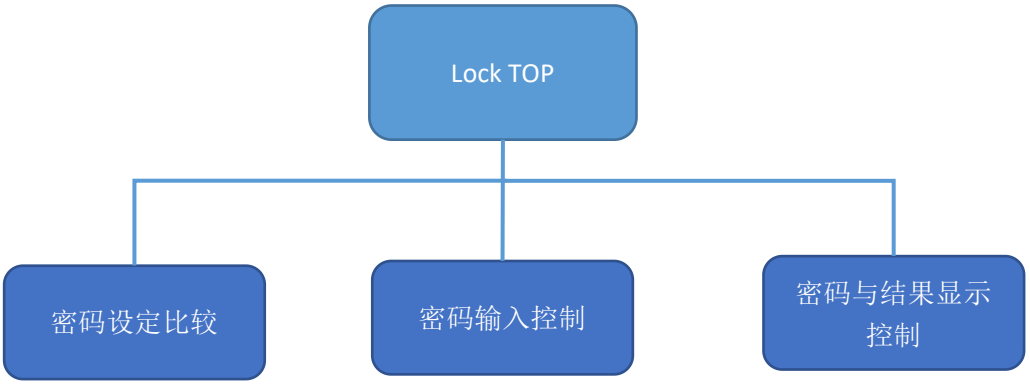
③密码验证模块；

S2 下降沿触发使能工作； S2 下降沿触发所存输入密码，并与拨码开关设置的密码进行比较；

输出密码比较结果，提供个数码管显示模块。

8.4.1 顶层模块设计

顶层模块与上述三个模块之间的关系如下图：



输入输出信号如下表：

信号	位宽	方向	描述
clk	1	输入	外部输入时钟，输入时钟为 50MHz
key	2	输入	轻触按键输入信号， K0~K1 输入
enter	1	输入	密码确认比对信号， K2 输入
init	1	输入	密码确认重新输入信号， K3 输入
sw	4	输入	密码设置输入信号， SW0~3 输入
smg	8	输出	密码对比结果显示数码管段选信号输出
dig	4	输出	密码对比结果显示数码管位选信号输出

Module 设计如下：

```
`timescale 1ns / 1ps
```

```
`define UD #1
module lock_top(
    input               clk,
    input   [1:0]       key,
    input               enter,
    input               init,
    input   [3:0]       sw,

    output [7:0]        smg,
    output [3:0]        dig
);

    wire                enter_trig;
    wire                init_trig;
    wire [3:0]          ctrl;
    wire                com_result;

    key_ctl key_ctl(
        .clk            (  clk          ),//input             clk,
        .key            (  key          ),//input       [1:0] key,
        .enter          (  enter        ),//input             enter,
        .init           (  init         ),//input             init,

        .enter_trig     (  enter_trig   ),//output            enter_trig,
        .init_trig      (  init_trig    ),//output            init_trig,
        .ctrl           (  ctrl         ) //output      [3:0] ctrl
    );

    compare compare(
        .clk            (  clk          ),//input             clk,
        .sw             (  sw           ),//input [3:0]       sw,
        .ctrl           (  ctrl         ),//input [3:0]       ctrl,
        .enter_trig     (  enter_trig   ),//input             enter_trig,
        .com_result     (  com_result   ) //output            com_result
    );

    seq_display seq_display(
        .clk            (  clk          ),//input             clk,
        .enter_trig     (  enter_trig   ),//input             enter_trig,
        .init_trig      (  init_trig    ),//input             init_trig,
        .com_result     (  com_result   ),//input             com_result,
        .ctrl           (  ctrl         ),//input       [3:0] ctrl,

        .smg            (  smg          ),//output reg  [7:0] smg,
```

```
        .dig            (   dig             ) //output reg   [3:0]     dig
    );

endmodule
```

8.4.2 按键控制设计

```
`timescale 1ns / 1ps
`define UD #1
module key_ctl(
    input               clk,
    input       [1:0] key,
    input               enter,
    input               init,

    output              enter_trig,
    output              init_trig,
    output      [3:0] ctrl
);

    wire [3:0] btn_deb;
    // 按键消抖
    btn_deb#(
        .BTN_WIDTH    (   4'd4      ) , //parameter     BTN_WIDTH = 4'd8
        .MS_20        (   20'd1_000_000     )
    ) btn_deb_key
    (
        .clk          (   clk           ),//input                 clk,
        .btn_in       (   {enter,init,key} ),//input [BTN_WIDTH-1:0] btn_in,
        .btn_deb      (   btn_deb ) //output reg [BTN_WIDTH-1:0]btn_deb
    );

    reg [1:0]   S1_push_cnt=2'd0;
    reg [1:0]   S2_push_cnt=2'd0;

    reg btn1_deb_1d,btn2_deb_1d;
    reg enter_deb_1d,init_deb_1d;

    assign enter_trig = ~btn_deb[3] & enter_deb_1d;
    assign init_trig = ~btn_deb[2] & init_deb_1d;

    always @(posedge clk)
```

```
    begin
        btn1_deb_1d   <= `UD btn_deb[0];
        btn2_deb_1d   <= `UD btn_deb[1];
        init_deb_1d   <= `UD btn_deb[2];
        enter_deb_1d <= `UD btn_deb[3];
    end

    always @(posedge clk)
    begin
        if(~btn_deb[2] & init_deb_1d)
            S1_push_cnt <= `UD 2'd0;
        else if(~btn_deb[0] & btn1_deb_1d)
        begin
            S1_push_cnt <= `UD S1_push_cnt + 2'd1;
        end
    end

    always @(posedge clk)
    begin
        if(~btn_deb[2] & init_deb_1d)
            S2_push_cnt <= `UD 2'd0;
        else if(~btn_deb[1] & btn2_deb_1d)
        begin
            S2_push_cnt <= `UD S2_push_cnt + 2'd1;
        end
    end

    assign ctrl = {S2_push_cnt,S1_push_cnt};

endmodule
```

8.4.3 按键消抖设计

```
`timescale 1ns / 1ps
`define UD #1
module btn_deb#(
    parameter                       BTN_WIDTH = 4'd8,
    parameter                       MS_20 = 20'd1_000_000
)
(
    input                           clk,//50MHz
    input       [BTN_WIDTH-1:0] btn_in,
```

```
        output reg [BTN_WIDTH-1:0] btn_deb
    );
    //=============================================================
============================
        reg [19:0] time_cnt= 20'd0;
        always@(posedge clk)
        begin
            if(time_cnt == MS_20 - 1'b1)
                time_cnt <= 20'd0;
            else
                time_cnt <= time_cnt + 1'd1;
        end

        always @(posedge clk)
        begin
            if(time_cnt == MS_20 - 1'b1)
                btn_deb <= btn_in;
        end

    endmodule
```

8.4.4 对比模块设计

```
    `timescale 1ns / 1ps
    `define UD #1
    module compare(
        input           clk,
        input [3:0]   sw,
        input [3:0]   ctrl,
        input           enter_trig,
        output          com_result
    );
    //=============================================================
=====================
        //锁存当前的输入密码;
        reg [3:0] ctrl_1d;
        always @(posedge clk)
        begin
            if(enter_trig)
                ctrl_1d <= `UD ctrl;
        end
        assign com_result = (ctrl_1d == sw);

    endmodule
```

8.4.5 显示模块设计

此模块设计需要注意数码管显示的两种模式：密码输入模式与密码对比结果显示模式；两种模式的切换由 enter_trig 与 init_trig 触发进入；

对于数码管的显示控制模块这里就不重复描述了；

8.5 实验现象

验证步骤：

1、 调整输入序列，更改拨码开关的输入值（SW[3：0]）；

2、 调整固定序列，通过轻触按键 S1~S0 调整输入密码，数码管实时显示输入密码；

3、 按下轻触按键 S2，触发进行密码比对，并且数码管显示比对结果；

4、 按下轻触按键 S3，进入重新输入密码状态，重新执行前面三个步骤；

实验现象

当 SW[3:0]=8’b1010;当输入密码状态时显示 0022 时，按下 S2 后数码管显示数字 8888；当输入密码状态时显示不是 0022 时，按下 S2 后数码管显示数字 7777；按下 S3 后重新调整密码，进入输入密码状态；

当 SW[3:0]=8’b1100;当输入密码状态时显示 0030 时，按下 S2 后数码管显示数字 8888；当输入密码状态时显示不是 0030 时，按下 S2 后数码管显示数字 7777；按下 S3 后重新调整密码，进入输入密码状态；