一、带安全锁定功能的电子密码锁

设计一个支持串行输入与退格修改功能的电子密码锁。该锁具有输入缓冲、错误计数及安全锁定机制，要求如下：

（1）输入与编辑逻辑：密码锁采用串行输入方式，正确密码固定为序列 1-2-3-4。系统内部需维护一个最大容量为 4 位的缓冲区。当 key_valid 有效时，将输入的 key_val 存入缓冲区；当缓冲区已满（4 位）时，忽略新的数字输入。当 backspace 信号有效时，若缓冲区非空，则删除最后输入的一位数字；

（2） 比对与开锁逻辑：当 confirm 信号有效时，对比缓冲区内的密码。若密码正确（且长度为 4），unlock 信号拉高并持续 10 个时钟周期，随后自动回到待机状态，同时将剩余重试次数 retry_cnt 重置为 2。无论比对结果如何，确认后必须立即清空缓冲区；

（3） 错误与锁定逻辑：系统复位后初始重试次数 retry_cnt 为 2。每提交一次错误密码，retry_cnt 减 1。当 retry_cnt 减为 0 时，系统进入锁定状态（LOCKED）；

（4） 锁定状态行为：进入锁定状态后，alarm 报警信号拉高。在此期间，忽略任何按键输入（数字、退格、确认均无效）。锁定持续 50 个时钟周期，结束后自动解除锁定，回到待机状态，并将 retry_cnt 自动恢复为 2。

程序头：

```
module serial_lock(
    input           clk,
    input           rst_n,
    input  [3 : 0]   key_val,
    input           key_valid,
    input           backspace,
    input           confirm,
    output          unlock,
    output          alarm,
    output [1 : 0]   retry_cnt,
    output [2 : 0]   input_len
);
endmodule
```

Benchmark：

```
`timescale 1ns/1ns
module tb_serial_lock();
    reg clk = 0;
    reg rst_n = 0;
    reg [3:0] key_val = 0;
    reg key_valid = 0;
    reg backspace = 0;
    reg confirm = 0;

    wire unlock, alarm;
    wire [1:0] retry_cnt;
    wire [2:0] input_len;

    // 实例化待测模块
    serial_lock u_dut(
        .clk(clk), .rst_n(rst_n),
        .key_val(key_val), .key_valid(key_valid),
        .backspace(backspace), .confirm(confirm),
        .unlock(unlock), .alarm(alarm),
        .retry_cnt(retry_cnt), .input_len(input_len)
    );

    // 时钟生成
    always #5 clk = ~clk;

    initial begin
        // 1. 系统初始化
        rst_n = 0;
        key_val = 0; key_valid = 0; backspace = 0; confirm = 0;
        #15 rst_n = 1;
        #10;

        // Case 1: 正常输入正确密码测试 (序列: 1-2-3-4)
        // 输入 1
        @(posedge clk); key_val = 4'd1; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk); // 间隔

        // 输入 2
        @(posedge clk); key_val = 4'd2; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk);
```

```
// 输入 3
@(posedge clk); key_val = 4'd3; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 输入 4
@(posedge clk); key_val = 4'd4; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 提交确认
@(posedge clk); confirm = 1;
@(posedge clk); confirm = 0;
@(posedge clk);

// 等待开锁信号拉高再拉低
wait(unlock == 1);
wait(unlock == 0);
#20;

// Case 2: 编辑与退格功能测试 (序列: 1-2-9-DEL-3-4)
// 输入 1
@(posedge clk); key_val = 4'd1; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 输入 2
@(posedge clk); key_val = 4'd2; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 输入 9 (错误位)
@(posedge clk); key_val = 4'd9; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

#10;
// 按下退格键
@(posedge clk); backspace = 1;
@(posedge clk); backspace = 0;
@(posedge clk);
#10;
```

```
// 输入 3
@(posedge clk); key_val = 4'd3; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 输入 4
@(posedge clk); key_val = 4'd4; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 提交确认
@(posedge clk); confirm = 1;
@(posedge clk); confirm = 0;
@(posedge clk);

// 检查是否成功开锁
wait(unlock == 1);
wait(unlock == 0);
#20;


// Case 3: 错误尝试与次数扣减测试 (初始重试机会=2)

// 输入 1
@(posedge clk); key_val = 4'd1; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 输入 2 (长度不足)
@(posedge clk); key_val = 4'd2; key_valid = 1;
@(posedge clk); key_valid = 0;
@(posedge clk);

// 提交确认
@(posedge clk); confirm = 1;
@(posedge clk); confirm = 0;
@(posedge clk);

// 此时观察波形 retry_cnt 应变 1
#20;

// Case 4: 耗尽机会触发锁定测试 (再错一次)

// 输入 0
```

```
        @(posedge clk); key_val = 4'd0; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk);
        // 输入 0
        @(posedge clk); key_val = 4'd0; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk);
        // 输入 0
        @(posedge clk); key_val = 4'd0; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk);
        // 输入 0
        @(posedge clk); key_val = 4'd0; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk);

        // 提交确认
        @(posedge clk); confirm = 1;
        @(posedge clk); confirm = 0;
        @(posedge clk);

        // 此时应触发 alarm
        wait(alarm == 1);

        // 锁定期间尝试按键 (应无效)
        #20;
        @(posedge clk); key_val = 4'd1; key_valid = 1;
        @(posedge clk); key_valid = 0;
        @(posedge clk);

        // 等待锁定自动解除
        wait(alarm == 0);

        // 此时观察波形 retry_cnt 应自动恢复为 2
        #20;
        $stop;
    end

endmodule
```

二、带冷却中断的自动交通灯控制系统

设计背景：

模拟智能十字路口交通灯。系统在默认情况下按固定周期自动循环（主干道红/绿/黄交替）。同时设有行人过街按钮，该按钮作为“中断源”，在主干道绿灯期间可触发中断，强制系统提前进入黄灯变灯流程。为防止滥用，中断触发后具有冷却保护期。

功能要求：

（1） 信号定义：

输入：clk, rst_n, ped_btn（行人按钮，脉冲）。

输出：main_light（主干道指示灯 0:绿, 1:黄, 2:红），ped_light（人行道指示灯 0:红, 1:绿）。

复位 (rst_n=0) 结束后，系统必须从主干道红灯（行人绿灯）状态开始运行。

*人行道指示灯仅在主干道指示灯为红灯时为绿灯。

（2） 自动循环流程（基准时序）：

• 主干道绿灯（行人红灯）：默认持续 10 个时钟周期。若无中断，时间到后主干道进入黄灯。

• 主干道黄灯（行人红灯）：持续 3 个时钟周期。时间到后主干道进入红灯。

• 主干道红灯（行人绿灯）：持续 10 个时钟周期。时间到后主干道回到绿灯。

（3） 中断与冷却机制：

中断触发：

仅在主干道绿灯状态下有效。

当 ped_btn 有效且不在冷却期时，无视主干道绿灯的剩余时间，立即 (ped_btn 有效的下一个时钟上升沿) 跳转到主干道黄灯状态，开始变灯流程。

冷却保护：

当 ped_btn 按下后会进入冷却期，之后的 18 个时钟周期内，系统必须忽略任何 ped_btn 信号（即不能触发中断）。

程序头：

```
module traffic_button_ctrl(
    input           clk,
    input           rst_n,
    input           ped_btn,
    output reg [1:0] main_light, // 0:Green, 1:Yellow, 2:Red
    output reg [1:0] ped_light   // 0:Red, 1:Green
);
endmodule
```

Benchmark (Testbench):

```
`timescale 1ns/1ns
module tb_traffic();
    reg clk = 0;
    reg rst_n = 0;
    reg ped_btn = 0;

    wire [1:0] main_light;
    wire [1:0] ped_light;

    traffic_button_ctrl u_dut(
        .clk(clk), .rst_n(rst_n),
        .ped_btn(ped_btn),
        .main_light(main_light), .ped_light(ped_light)
    );

    always #5 clk = ~clk;

    initial begin
        rst_n = 0; #10 rst_n = 1;
        #340;
        ped_btn = 1;//行人红灯按下
        #10;
        ped_btn = 0;
        #270
        ped_btn = 1;//行人绿灯按下
        #10;
        ped_btn = 0;
        #150
        ped_btn = 1;//行人红灯按下
        #10;
        ped_btn = 0;
        #150;
        ped_btn = 1;//行人连续按下
        #10;
        ped_btn = 0;


        #50;
        $stop;
    end
    endmodule
```