數位矽智產設計導論

期末專題結果報告

-基於FPGA的4-bit簡易電子密碼鎖

學生：李昀易　學號：7112064112
學生：郭奕承　學號：4108064114

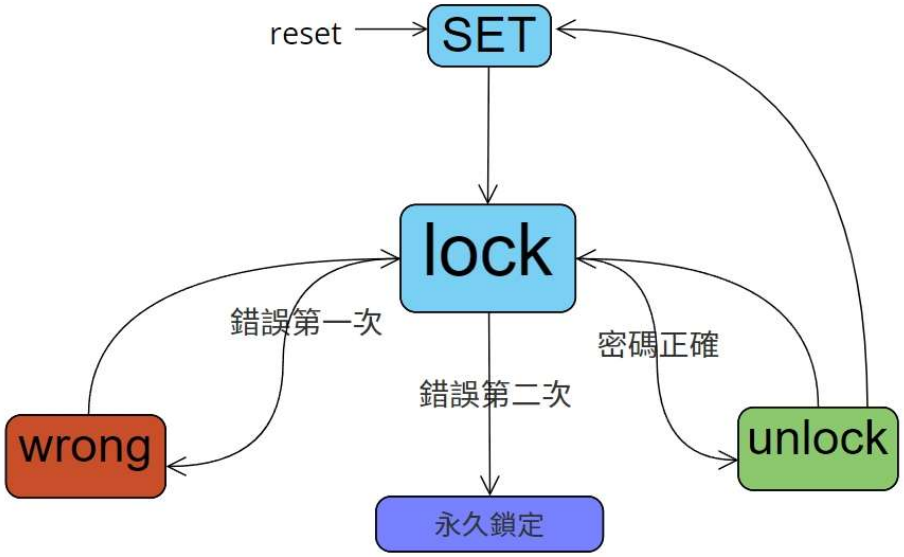
一、概述:

這是一個可以輸入 4 個十進位數字的電子密碼鎖(ex:1234, 7115.....)，使用者可以設置一組自訂的密碼，並將其鎖上；當密碼輸入正確時，可將其解開，並可以選擇重設密碼或再次鎖上；當密碼連續輸入錯誤兩次以上，系統將自動鎖定，無法再輸入密碼，只能透過 reset 重置系統。

二、詳細功能

1. 密碼設置：允許使用者使用 FPGA 設定一組由 0~9 組成的四位固定密碼
2. 密碼輸入：提供用戶通過四個(4bit, 二進制)指撥開關(switch)輸入密碼，輸入的密碼會轉成十進制顯示在 LED 上
3. 密碼驗證：用於驗證用戶輸入的密碼是否與設定的密碼相匹配
4. 開門信號：如果密碼輸入正確，FPGA 點亮四個 LED，並顯示”p”以模擬解鎖的動作
5. 密碼錯誤: 如果密碼輸入錯誤，會顯示”n”代表沒有成功解鎖，需要再輸入一次密碼
6. 安全機制：添加安全機制，輸入錯誤次數超過兩次，則系統會自動鎖定，只能透過 reset 重製整個密碼鎖
7. 密碼重設：當為解鎖狀態時，密碼可以被重設

三、流程圖



說明:

SET:

一開始取得電子鎖時為全新、未設置密碼狀態。在此狀態時可以設置四個十進制(0~9)的密碼 ex: 1234, 9819...

LOCK:

此狀態七段顯示器會顯示"In" 代表鎖定狀態，可以開始輸入密碼解鎖

Unlock:

當密碼輸入正確時，顯示"p"，此時可以選擇要重設密碼(SET)，或是重新鎖定電子鎖(lock)

Wrong:

當密碼第一次輸入錯誤時，顯示"n"，此時需要重新輸入密碼(lock)

永久鎖定:

當密碼第二次輸入錯誤時，顯示"lock"，此時電子鎖永久鎖定無法解開，除非系統重置

四、操作流程

1. 重置系統:按下reset重置系統，進入到設定密碼的狀態
2. 設定密碼:

   用指撥開關輸入數字，按下S0確定數字，按下S1結束第一位的輸入

   用指撥開關輸入數字，按下S0確定數字，按下S2結束第二位的輸入

   用指撥開關輸入數字，按下S0確定數字，按下S3結束第三位的輸入

   用指撥開關輸入數字，按下S0確定數字，按下S4結束第四位的輸入，進入到鎖定狀態
3. 鎖定狀態: 畫面顯示"ln"，按下S0進入到輸入密碼狀態
4. 輸入密碼狀態:

   用指撥開關輸入數字，按下S0確定數字，按下S1結束第一位的輸入

   用指撥開關輸入數字，按下S0確定數字，按下S2結束第二位的輸入

   用指撥開關輸入數字，按下S0確定數字，按下S3結束第三位的輸入

   用指撥開關輸入數字，按下S0確定數字，按下S4結束第四位的輸入

   系統會驗證密碼是否正確
5. 輸入正確:

   畫面顯示"p"，按下S0回到設定密碼狀態，按下S1重新鎖上密碼鎖(回到鎖定狀態)
6. 輸入錯誤:

   畫面顯示"n"，按下S1重新輸入密碼狀態，若:

   輸入正確:回到5.輸入正確

   再次輸入錯誤:畫面顯示"lock"密碼鎖鎖死只能透過reset重置系統

五、code

```
`define CYCLE 10000 // 單一 cycle 長度

module lock(
    input clk,
    input rst_n,
    input [7:0] switch,
    input s0_r11, s1_r17, s2_r15, s3_v1, s4_u4, // 按鈕輸入
    output [7:0] seg7,
    output [3:0] seg7_sel,
    output [3:0] led,  //大 LED 燈的右四顆
    output [3:0] led2, //大 LED 燈的左四顆
    output [3:0] led_small_left, //小 LED 燈的左四顆
    output [3:0] led_small_right //小 LED 燈的右四顆

    );
    parameter TEST=1, WRONG=2, RIGHT=3, LOCK=4, READY=14;
    parameter INPUT1=5, INPUT2=6, INPUT3=7, INPUT4=8;
    parameter SET1=0, SET2=11, SET3=12, SET4=13;

    //暫存器宣告
    reg [7:0] seg7;
    reg [3:0] seg7_sel;
    reg [3:0] seg7_temp [0:3];
    reg [1:0] seg7_count;
    reg state_show;
    reg [7:0] state_show_seg7;

    reg [3:0] led;
    reg [3:0] led2;
    reg [3:0] led_small_left, led_small_right;

    reg [29:0] count;
    wire d_clk;

    reg [3:0] password_1;
    reg [3:0] password_2;
    reg [3:0] password_3;
    reg [3:0] password_4;

    reg [3:0] put_in_1;
    reg [3:0] put_in_2;
```

```
    reg [3:0] put_in_3;
    reg [3:0] put_in_4;

    reg wrong_flag;

    //FSM
    reg [4:0] state, nx_state;
    always @(posedge clk or negedge rst_n) begin
        if(!rst_n) begin
            state <= SET1;
        end
        else state <= nx_state;
    end
    always @(posedge clk or negedge rst_n) begin
        if(!rst_n) nx_state <= SET1;
        else
            case (state)
                SET1: nx_state <= (s1_r17) ? SET2 : SET1;
                SET2: nx_state <= (s2_r15) ? SET3 : SET2;
                SET3: nx_state <= (s3_v1) ? SET4 : SET3;
                SET4: nx_state <= (s4_u4) ? READY : SET4;

                READY: nx_state <= (s0_r11) ? INPUT1 : READY;

                INPUT1: nx_state <= (s1_r17) ? INPUT2 : INPUT1;
                INPUT2: nx_state <= (s2_r15) ? INPUT3 : INPUT2;
                INPUT3: nx_state <= (s3_v1) ? INPUT4 : INPUT3;
                INPUT4: begin
                    if (password_1==put_in_1 && password_2==put_in_2 &&
password_3==put_in_3 && password_4==put_in_4 && s4_u4) nx_state <= RIGHT;
                    else if((password_1!==put_in_1 || password_2!==put_in_2 ||
password_3!==put_in_3 || password_4!==put_in_4 ) && s4_u4) begin
                        if(wrong_flag) nx_state <= LOCK;
                        else nx_state <= WRONG;
                    end
                    else nx_state <= INPUT4;
                end
                RIGHT: begin
                    if(s0_r11) nx_state <= SET1;
                    else if(s1_r17) nx_state <= READY;
                    else nx_state <= RIGHT;
                end
```

```
            WRONG: nx_state <= s1_r17  ? READY : WRONG;
            LOCK: nx_state <= LOCK;

            default: nx_state <= SET1;
        endcase
end

// state_show
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) led <= 4'b1111;
    else
        case (state)
            SET1, SET2, SET3, SET4: led <=4'b0001;
            INPUT1, INPUT2, INPUT3: led <=4'b0010;
            TEST, INPUT4: led <=4'b0100;
            RIGHT: led <=4'b1111;
            WRONG: led <=4'b1000;
            LOCK: led <= 4'b1111;
            READY: led <= 4'b1100;
            default: led <= 4'b0000;
        endcase
end

// 確認有輸入進去
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) led2 <= 4'b1111;
    else begin
        if(s0_r11) led2 <=4'b1111;
        else led2 <= 0;
    end
end

// 顯示目前輸入第幾位
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) led_small_right <= 4'b0;
    else
        case (state)
            SET1, INPUT1: led_small_right <=4'b0001;
            SET2, INPUT2: led_small_right <=4'b0010;
            SET3, INPUT3: led_small_right <=4'b0100;
            SET4, INPUT4: led_small_right <=4'b1000;
```

```
            default: led_small_right <= 4'b0000;
        endcase
end

// 錯誤顯示次數
always @(posedge clk or negedge rst_n) begin
    if(!rst_n||state==RIGHT) led_small_left <= 4'b0;
    else if(wrong_flag&&state==LOCK) led_small_left <= 4'b1111; // 鎖定時全亮
    else if(wrong_flag) led_small_left <=4'b0001; // 錯誤一次亮一顆
    else led_small_left <= led_small_left;
end

// 關於儲存設定的密碼
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        password_1 <= 0;
        password_2 <= 0;
        password_3 <= 0;
        password_4 <= 0;
    end
    else if(state==SET1 && s0_r11)
        password_1 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else if(state==SET2 && s0_r11)
        password_2 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else if(state==SET3 && s0_r11)
        password_3 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else if(state==SET4 && s0_r11)
        password_4 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else begin
        password_1 <= password_1;
        password_2 <= password_2;
        password_3 <= password_3;
        password_4 <= password_4;
    end
end


// 關於儲存輸入的密碼
always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        put_in_1 <= 0;
        put_in_2 <= 0;
        put_in_3 <= 0;
```

```
        put_in_4 <= 0;
    end
    else if(state==INPUT1 && s0_r11)
        put_in_1 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else if(state==INPUT2 && s0_r11)
        put_in_2 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else if(state==INPUT3 && s0_r11)
        put_in_3 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else if(state==INPUT4 && s0_r11)
        put_in_4 <= switch[3:0]<4'b1010 ? switch[3:0] : 0 ;
    else begin
        put_in_1 <= put_in_1;
        put_in_2 <= put_in_2;
        put_in_3 <= put_in_3;
        put_in_4 <= put_in_4;
    end
end


// wrong_flag
always @(posedge clk or negedge rst_n)begin
    if(!rst_n) wrong_flag <= 0;
    else if(state==WRONG)  wrong_flag <= 1'b1;
    else if(state==RIGHT)  wrong_flag <= 0;
    else wrong_flag <= wrong_flag;
end

//七段顯示器顯示
always @(posedge d_clk or negedge rst_n)begin
    if(!rst_n)begin
        seg7_count <= 0;
    end
    else begin
        seg7_count <= seg7_count + 1;
    end
end

always @(posedge d_clk or negedge rst_n)begin
    if(!rst_n)begin
        seg7_sel <= 0;
        seg7 <= 0;
    end
    else begin
```

```
        case (state)
            SET1, SET2, SET3, SET4, INPUT1, INPUT2, INPUT3, INPUT4: begin // 顯示十進制
                case(seg7_count)
                    0:  seg7_sel <= 4'b0001;
                    1:  seg7_sel <= 4'b0010;
                    2:  seg7_sel <= 4'b0100;
                    3:  seg7_sel <= 4'b1000;
                endcase
                case(seg7_temp[seg7_count])
                    0:seg7 <= 8'b0011_1111;
                    1:seg7 <= 8'b0000_0110;
                    2:seg7 <= 8'b0101_1011;
                    3:seg7 <= 8'b0100_1111;
                    4:seg7 <= 8'b0110_0110;
                    5:seg7 <= 8'b0110_1101;
                    6:seg7 <= 8'b0111_1101;
                    7:seg7 <= 8'b0000_0111;
                    8:seg7 <= 8'b0111_1111;
                    9:seg7 <= 8'b0110_1111;
                endcase
            end

            TEST: begin
                seg7_sel <= 4'b0001;
                seg7 <= 8'b0011_1111; //顯示 0
            end

            RIGHT:begin
                seg7_sel <= 4'b0001;
                seg7 <= 8'b0111_0011; //密碼正確以 P 表示
            end

            WRONG:begin
                seg7_sel <= 4'b0001;
                seg7 <= 8'b0101_0100; //密碼錯誤以 N 表示
            end

            READY: begin
                case(seg7_count)
                    2:  begin
                        seg7 <= 8'b0101_0100; //顯示 n
                        seg7_sel <= 4'b0001;
```

```
                    end
                    3:  begin
                        seg7 <= 8'b0000_0110;
                        seg7_sel <= 4'b0010; //顯示 1
                    end
                endcase
            end

            LOCK: begin
                case(seg7_count)
                    0:  begin
                        seg7_sel <= 4'b0001;
                        seg7 <= 8'b0111_0110; //顯示 K
                    end
                    1:  begin
                        seg7_sel <= 4'b0010;
                        seg7 <= 8'b0011_1001; //顯示 C
                    end
                    2:  begin
                        seg7_sel <= 4'b0100;
                        seg7 <= 8'b0011_1111; //顯示 O
                    end
                    3:  begin
                        seg7_sel <= 4'b1000;
                        seg7 <= 8'b0011_1000; //顯示 L
                    end
                endcase
            end

            default: begin
                seg7_sel <= 4'b0001;
                seg7 <= 8'b0011_1111;
            end
        endcase

    end
end

//switch 二進位轉十進位
always @(posedge clk or negedge rst_n)begin
    if(!rst_n)begin
        seg7_temp[0] <= 0;
```

```
            seg7_temp[1] <= 0;
            seg7_temp[2] <= 0;
            seg7_temp[3] <= 0;
        end
        else begin
            if(switch[3:0]<4'b1010)
                case (state)
                    SET1, SET2, SET3, SET4, INPUT1, INPUT2, INPUT3, INPUT4: begin //
SET1ting
                        seg7_temp[3] <= 0;
                        seg7_temp[2] <= (switch[3:0] % 1000) / 100;
                        seg7_temp[1] <= (switch[3:0] % 100) / 10;
                        seg7_temp[0] <= switch[3:0] % 10;
                    end
                endcase
            else begin
                seg7_temp[0] <= 0;
                seg7_temp[1] <= 0;
                seg7_temp[2] <= 0;
                seg7_temp[3] <= 0;
            end

        end
    end

    //除頻
    always @(posedge clk or negedge rst_n)begin
        if(!rst_n)
            count <= 0;
        else if (count >= `CYCLE)
            count <= 0;
        else
            count <= count + 1;
    end
    assign d_clk = count > (`CYCLE/2) ? 0 : 1;

endmodule
```