

## 邏輯系統實驗

Lab 10 PYNQ+七段顯示器

2022/05/18

第 5 組	
組員姓名	學號
林珮玉	E24084096
廖本恩	E24102179
蘇冠誠	E24084143

# 基礎題一：

## 1. 簡述題目：4 bits 輸入七段顯示器

顯示 ( 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F ) 。

## 2. 實現方式：

BTN[3:0] – 數字輸入， BTN[3] 為最高位元。

SW[0] – reset，非同步的 reset 訊號，在高準位時會重置目前的數字。

LED[3:0] – 數字顯示，當相應的 BTN 按下時， LED 的亮暗會被切換。

ar[6:0] – 七段顯示器輸出，分別對應七端顯示器的 g、f、e、d、c、b、a 。

seg.v

```
`timescale 1ns / 1ps
///////////////////////////////
///////////////////
// Company:
// Engineer:
//
// Create Date: 2022/04/16 18:40:38
// Design Name:
// Module Name: seg7
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
```

```
//////////  
//////////  
  
module seg7(clk,rst,a,out);  
  
input clk;  
input rst ;  
input [3:0]a ;  
output reg [6:0] out;  
  
always@(*)begin  
    case(a)  
        4'b0000: out = 7'b1000000;  
        4'b0001: out = 7'b1111001;  
        4'b0010: out = 7'b0100100;  
        4'b0011: out = 7'b0110000;  
        4'b0100: out = 7'b0011001;  
        4'b0101: out = 7'b0010010;  
        4'b0110: out = 7'b0000010;  
        4'b0111: out = 7'b1111000;  
        4'b1000: out = 7'b0000000;  
        4'b1001: out = 7'b0010000;  
        4'b1010: out = 7'b0001000;  
        4'b1011: out = 7'b0000011;  
        4'b1100: out = 7'b1000110;  
        4'b1101: out = 7'b0100001;  
        4'b1110: out = 7'b0000110;  
        4'b1111: out = 7'b0001110;  
        default: out = 7'b1111111;  
    endcase  
end  
endmodule
```

top.v

```
'timescale 1ns / 1ps  
//////////  
// Company:  
// Engineer:
```

```
//  
// Create Date: 2022/03/27 19:49:21  
// Design Name:  
// Module Name: top  
// Project Name:  
// Target Devices:  
// Tool Versions:  
// Description:  
//  
// Dependencies:  
//  
// Revision:  
// Revision 0.01 - File Created  
// Additional Comments:  
//  
///////////////////////////////  
  
module top(  
    input sysclk,//used  
    input [1:0] sw,//used  
    input [3:0] btn,//used  
    output reg [3:0] led,//used  
    output wire led4_b,  
    output wire led4_g,  
    output wire led4_r,  
    output wire led5_b,  
    output wire led5_g,  
    output wire led5_r,  
    //gpio  
    output wire [7:0] ar //used  
);  
    reg [3:0]sel;  
    wire [3:0] btn_out ;  
  
    always@(posedge btn[3] or posedge sw[0])begin  
        if(sw[0])begin  
            led[3] <= 1'd0 ;  
            sel[3] <= 1'd0 ;
```

```

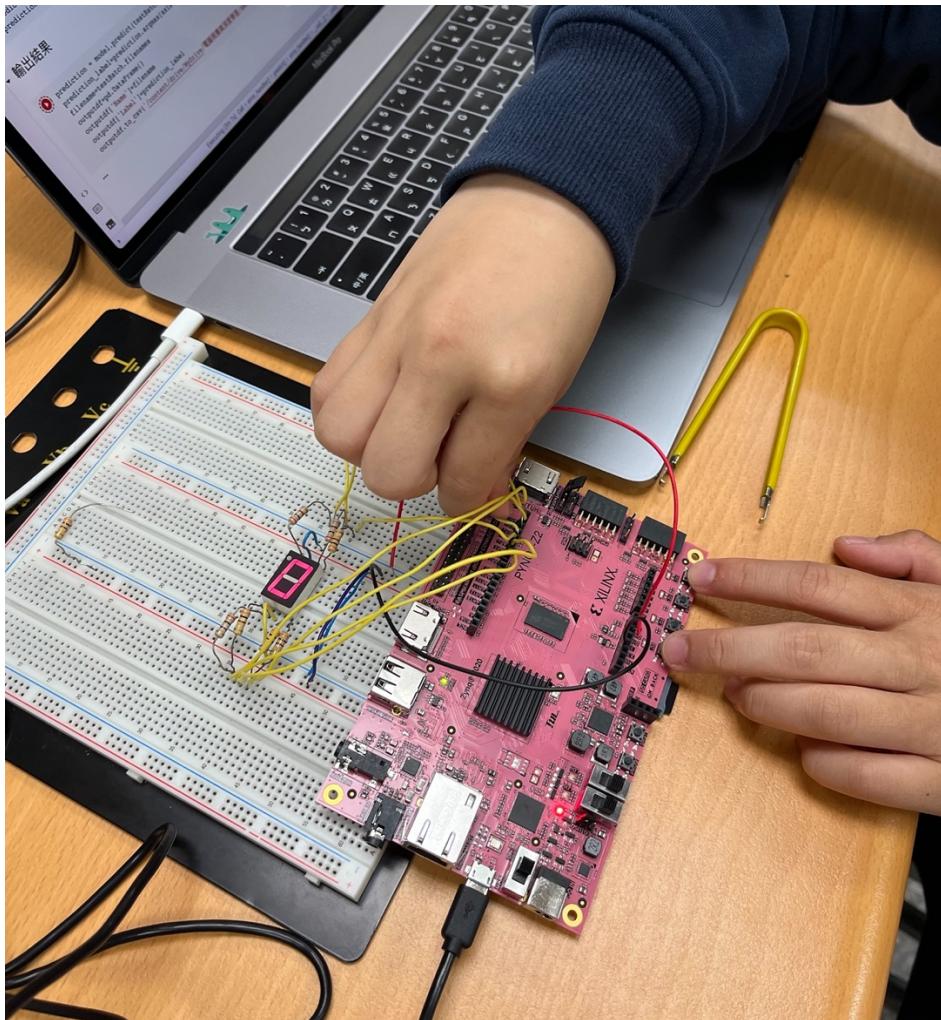
        end
    else begin
        led[3] <= ~led[3] ;
        sel[3] <= ~sel[3] ;
    end
end
always@(posedge btn[2] or posedge sw[0])begin
if(sw[0])begin
    led[2] <= 1'd0 ;
    sel[2] <= 1'd0 ;
end
else begin
    led[2] <= ~led[2] ;
    sel[2] <= ~sel[2] ;
end
end
always@(posedge btn[1] or posedge sw[0])begin
if(sw[0])begin
    led[1] <= 1'd0 ;
    sel[1] <= 1'd0 ;
end
else begin
    led[1] <= ~led[1] ;
    sel[1] <= ~sel[1] ;
end
end
always@(posedge btn[0] or posedge sw[0])begin
if(sw[0])begin
    led[0] <= 1'd0 ;
    sel[0] <= 1'd0 ;
end
else begin
    led[0] <= ~led[0] ;
    sel[0] <= ~sel[0] ;
end
end
end

debounce debounce_1(.din(btn[3]),.dout(btn_out[3]),.clk(sysclk),.rst(sw[0])) ;

```

```
debounce debounce_2(.din(btn[2]),.dout(btn_out[2]),.clk(sysclk),.rst(sw[0])) ;  
debounce debounce_3(.din(btn[1]),.dout(btn_out[1]),.clk(sysclk),.rst(sw[0])) ;  
debounce debounce_4(.din(btn[0]),.dout(btn_out[0]),.clk(sysclk),.rst(sw[0])) ;  
  
seg7 seg7_(.a(sel),.out(ar[6:0]));  
  
endmodule
```

### 3. 結果呈現



影片連結：

<https://drive.google.com/file/d/1Jl9pdmg67cE96u19rNoFCdPTKgr6uYg0/view?usp=sharing>

## 基礎題二：

1. 簡述題目：4 bits 輸入七段顯示器

使用 `BTN[3:0]` 作為輸入，並且幫按鈕加上 debounce circuit。

2. 實現方式：

`seg.v` 以及 `top.v` 皆與上題相同

```
seg.v
`timescale 1ns / 1ps
///////////////////////////////
///////////////////
// Company:
// Engineer:
//
// Create Date: 2022/04/17 12:50:50
// Design Name:
// Module Name: debounce
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////
///////////////////

module debounce(din,dout,clk,rst) ;
  input din,clk,rst ;
  output reg dout;
```

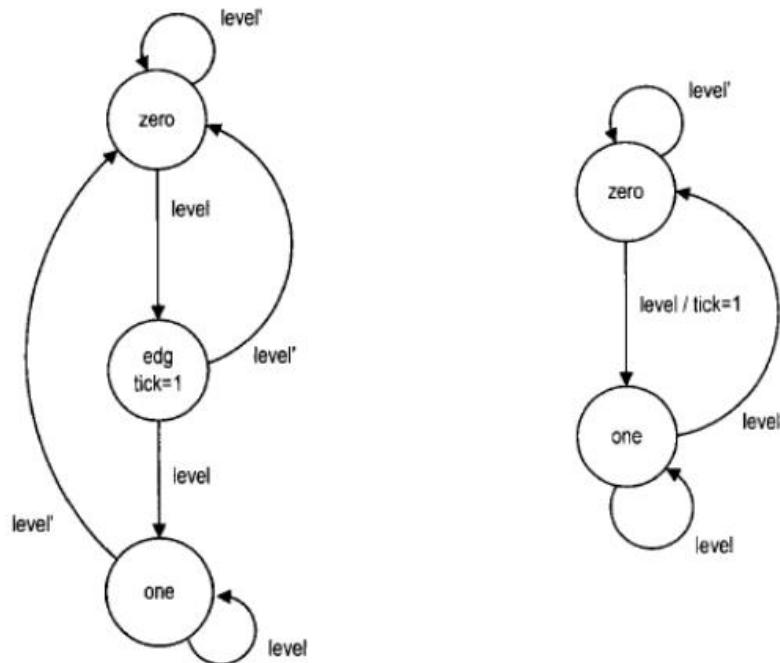
```

reg [2:0] next_state,current_state ;

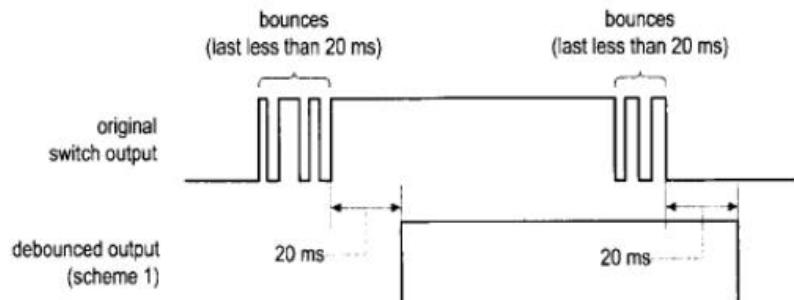
always@(*)begin
    case(current_state)
        3'd0 : next_state = (din)?3'd0:3'd1;
        3'd1 : next_state = (din)?3'd0:3'd2;
        3'd2 : next_state = (din)?3'd0:3'd3;
        3'd3 : next_state = (din)?3'd0:3'd4;
        3'd4 : next_state = (din)?3'd0:3'd5;
        3'd5 : next_state = (din)?3'd0:3'd5;
        default : next_state = 3'd0 ;
    endcase
end
always@(posedge clk or posedge rst)begin
    if(rst)begin
        current_state <= 3'd0;
    end
    else begin
        current_state <= next_state ;
    end
end
always@(*)begin
    case(current_state)
        3'd0 : dout = 1'b1 ;
        3'd1 : dout = 1'b1 ;
        3'd2 : dout = 1'b1 ;
        3'd3 : dout = 1'b1 ;
        3'd4 : dout = 1'b1 ;
        3'd5 : dout = 1'b0 ;
        default : dout = 1'b1 ;
    endcase
end
endmodule

```

Debounce 概念描述：



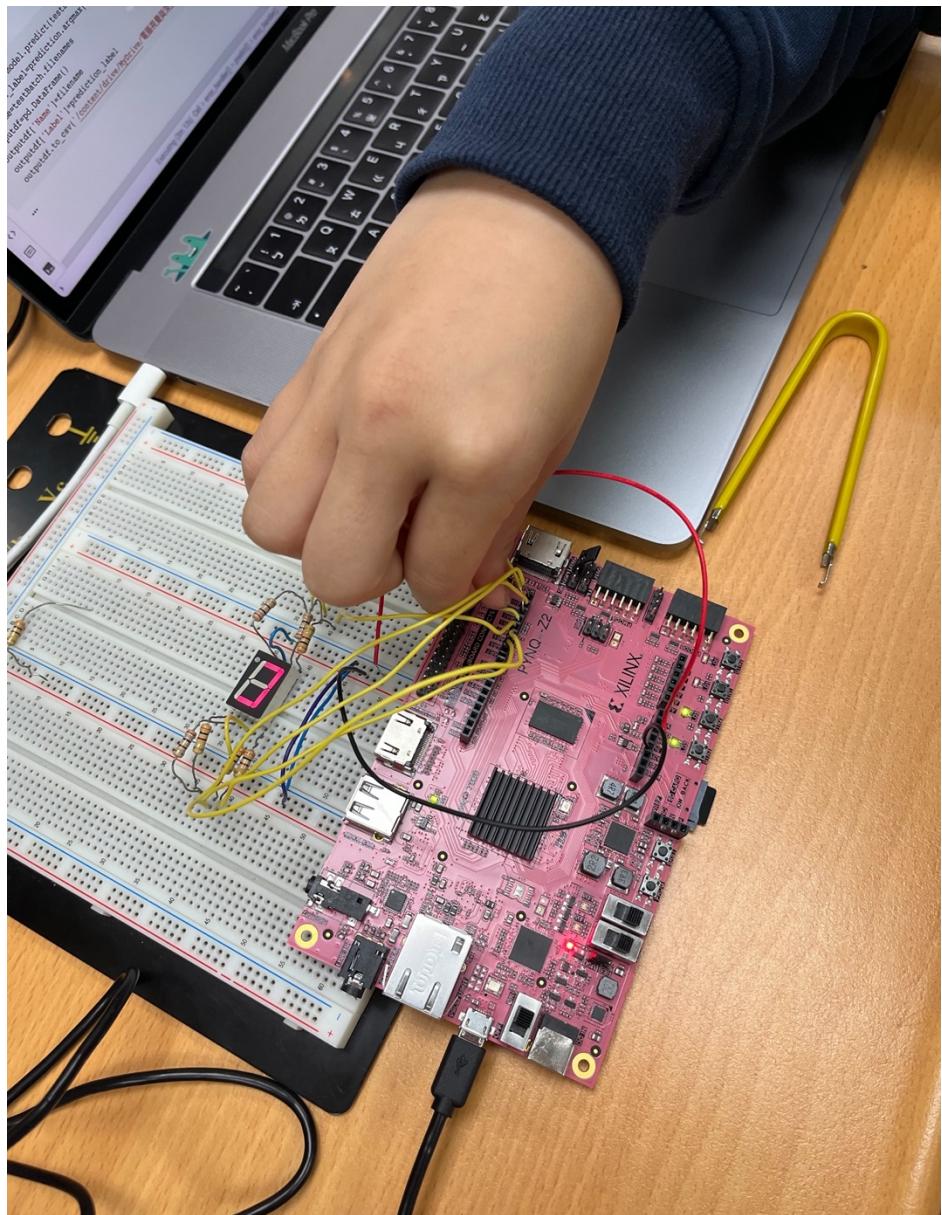
Moore and Mealy Model FSM State Diagrams



Original and debounced outputs

利用 Moore Machine 達到 Debounce 效果，也就是其 Next step 只與 Current Step 有關，若經過幾個 Moore 機之後，當 Next step = Current Step，表示狀態達穩定，才會觸發後續程式。

### 3. 結果呈現



影片連結：

[https://drive.google.com/file/d/1NGLiMPMwfkQGfbpGL\\_cz9f8E  
QSr0FLO3/view?usp=sharing](https://drive.google.com/file/d/1NGLiMPMwfkQGfbpGL_cz9f8EQSr0FLO3/view?usp=sharing)

# 心得

## 組員一 林珮玉 E24084096

這次實驗主要是學習 state machine 的流程，原本以為這次要完蛋了，不過很感謝助教放寬要求，code 部分沒有出太難，甚至可以在挑戰提找到答案哈哈。其中我最印象深刻的是 debounce 的概念，其實之前做 Arduino 的時候有稍微接觸過，但那時候還不太清楚 debounce 的運作方式，只知道要加 delay 即可免除其不穩定的問題。透過這次實驗的體驗之後，我才知道原來背後的硬體邏輯涉及 Moore 狀態機問題！雖然整題來說實驗不是很順利，但是學到很多，也很感謝組員特地回成大的一起補做並完成 Demo。

## 組員二 廖本恩 E24102179

這次應該是整學期實驗課最不順利的一次，連第一個 demo 都沒做出來，而且燒到板子的過程也不太順利。但是因為這些挫折，也讓我學會從零開始，從拉檔案到產生 block、拉板子的 I/O、合成為產生 bit stream 完整的跑過一遍，也是有很大的收穫。在補做的時候也更仔細看程式碼的邏輯，了解自己的盲點，說不定也是這學期學到最多東西的一堂課。

## 組員三 蘇冠誠 E24084143

這次實驗我幾乎完全幫不上忙，我就只能相信其他組員能成功後來我有看了助教給的檔案，有關控制訊號的敘述只有在板子夠理想時才可能正常運作，所以我可能會用計數器與電路內降頻的方式（frequency divider）來嘗試避開控制訊號不穩的情況。