

Laboratory 11

軟硬體共同實驗 (三): 有限狀態機之實作與文字型LCD之應用



Department of Electrical Engineering
National Cheng Kung University

實驗目的

- 瞭解有限狀態機(finite-state machine, FSM)
- 利用HDL實作有限狀態機與驗證
- 利用LCD實作文字顯示器

使用器材

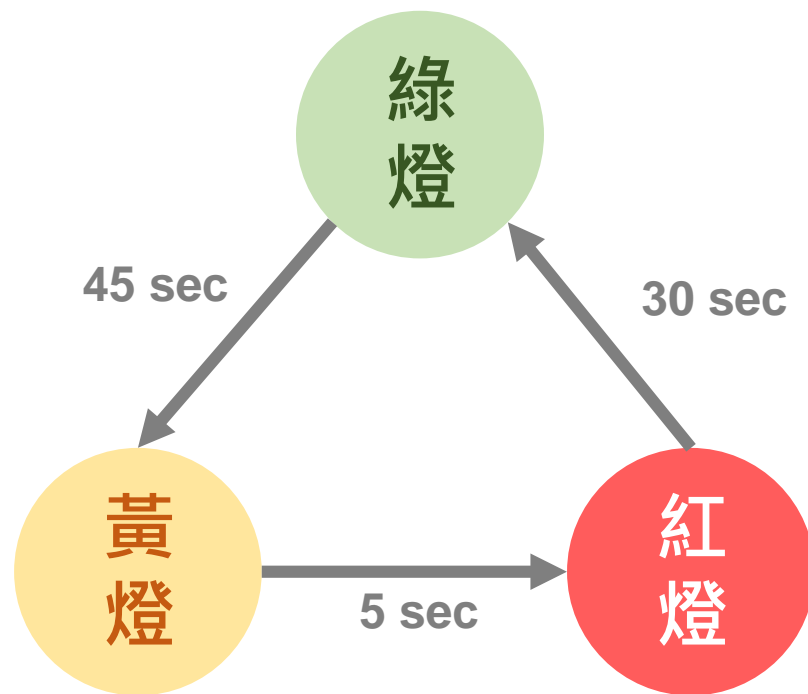
- 桌上型電腦
- Xilinx FPGA 板

Finite State Machine (1/4)

- 有限狀態機是由一組狀態、一個起始狀態、輸入、將輸入與現在狀態轉換為下一個狀態的轉換函數所組成。
- 在現實中，有許多事情可以用有限狀態機來表達，如：紅綠燈、自動販賣機 ... 等。
- 在資訊領域中，很多事情是由有限狀態所組成，再由不同的輸入而衍生出近乎無限的變化。下面將以兩個例子來說明有限狀態機。

Finite State Machine (2/4)

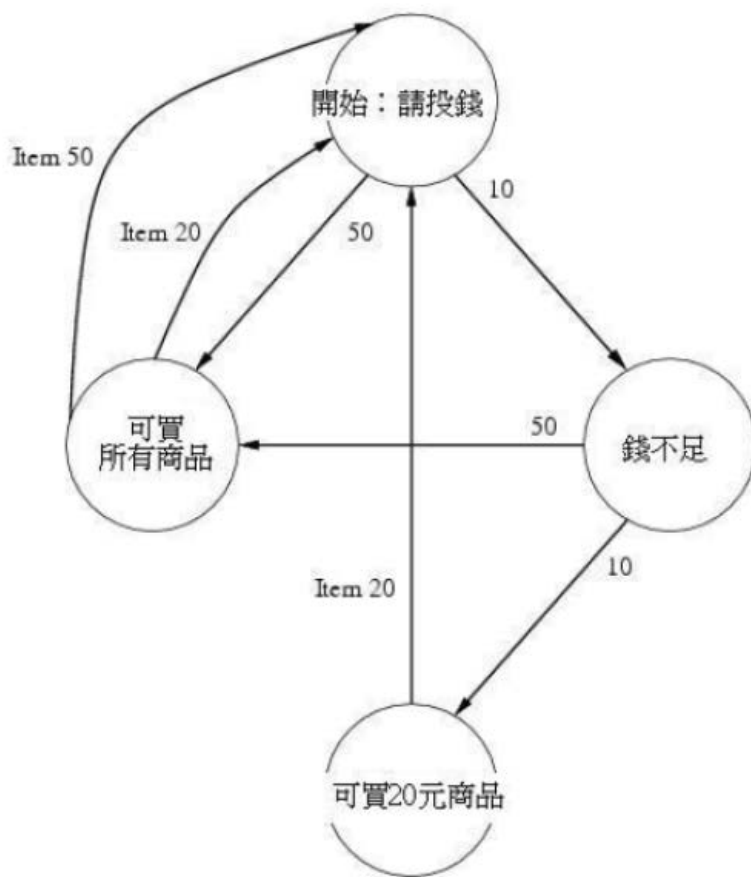
- 紅綠燈：從一開始的綠燈，經過一段時間後，將變為黃燈，再隔一會兒，就會變成紅燈，如此不斷反覆。其狀態圖如下



Finite State Machine (3/4)

- 自動販賣機指販賣兩類商品，一類售價20元，另一類售價50元。
- 假設該販賣機只能辨識10元與50元硬幣。
- 一開始機器處於“請投錢”的狀態，當投入10元時，機器會進入“錢不足”的狀態，直到投入的金額大於20元為止。
- 如果錢已投入50元以上金額，則可以選擇所有的產品，反則只能選擇20元的產品。
- 完成選擇後，將會賣出商品並且找回剩餘的零錢，隨後機器又將返回初始狀態。
下頁表示狀態圖。

Finite State Machine (4/4)



初始狀態：

開始：請投錢

輸入：

1. 10元
2. 50元
3. 選擇商品

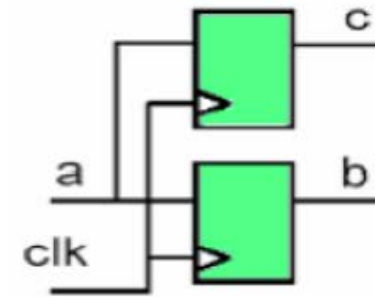
輸出：

1. 50元商品
2. 20元商品
3. 找零錢

Blocking and non-blocking

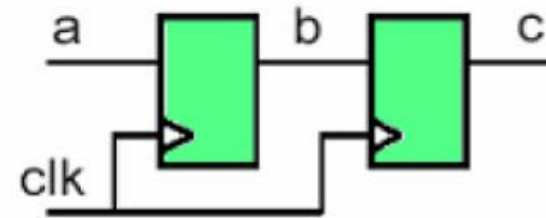
Blocking

```
always @(posedge clk)
begin
    b = a;
    c = b;
end
```



Non-blocking

```
always @(posedge clock)
begin
    b <= a;
    c <= b;
end
```



Finite State Machine (5/7)

Moore Machine

- Moore 有限狀態機的輸出只與當前狀態有關，與當下的輸入信號無關。
- EX:紅綠燈

Mealy Machine

- Mealy 有限狀態機的輸出不只與當前狀態有關，且與當下的輸入信號有關。
- EX:自動販賣機

因為Mealy比較複雜，本實習只介紹 moore machine

Finite State Machine (6/7)

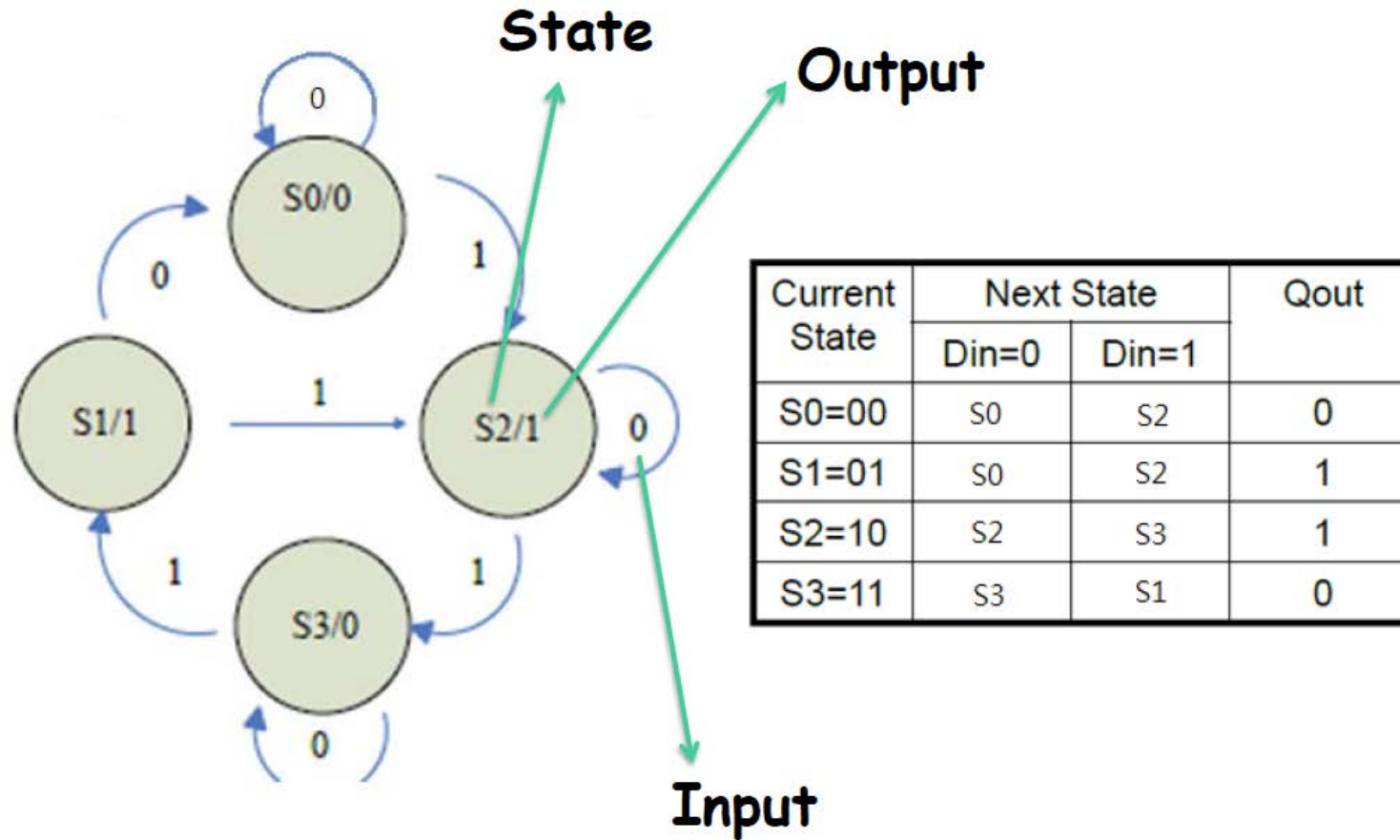
➤ Moore Machine



➤ Mealy Machine



Finite State Machine (7/7)



Finite State Machine : Good coding style

```
module moore_good(Clk,
  Reset, In_Data, Out_Data);

input Clk, Reset, In_Data;
output [1:0] Out_Data;
reg [1:0] Out_Data;
reg [1:0] State, NextState;
parameter S0=2'b00, S1=2'b01,
  S2=2'b10, S3=2'b11;
```

```
always @(posedge Clk or
  posedge Reset)
begin
  if(Reset)
    State = S0;
  else
    State = NextState;
end
State register (flip-flops)
```

```
always @(In_Data or State)
begin
  case(State)
    S0: begin
      if(In_Data == 1)
        NextState = S2;
      else
        NextState = S0;
      end
    S1: begin
      if(In_Data == 1)
        NextState = S2;
      else
        NextState = S0;
      end
    S2: begin
      if(In_Data == 1)
        NextState = S3;
      else
        NextState = S2;
      end
    S3: begin
      if(In_Data == 1)
        NextState = S1;
      else
        NextState = S3;
      end
    endcase
  end
Next state logic
end
```

```
always @(State)
begin
  case(State)
    S0: Out_Data = 0;
    S1: Out_Data = 1;
    S2: Out_Data = 1;
    S3: Out_Data = 0;
  endcase
end
Output logic
endmodule
```

Finite State Machine : Example (1/2)

```
`timescale 1ns/10ps
module moore (Qout, clk, rst, Din);
output Qout;
input clk, rst, Din;
parameter [1:0] S0=2'b00, S1=2'b01, S2=2'b10, S3=2'b11;
reg Qout;
reg [1:0] CS, NS; // CS: current state; NS: next state
```

Input, Output, Reg, Wire

```
.....

always @ (posedge clk or posedge rst)
begin
if (rst==1'b1) CS=S0;
else CS = NS;
end
```

State register (Flip-flop)

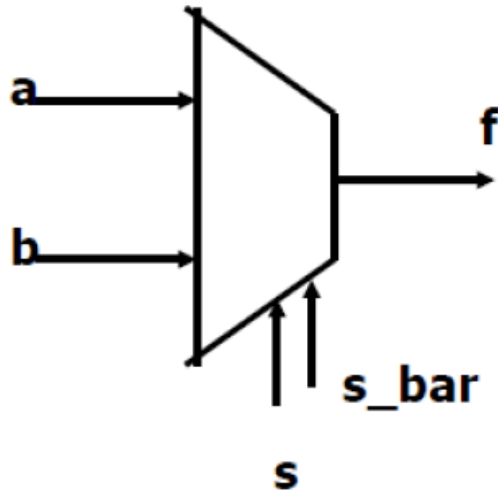
Finite State Machine : Example(2/2)

```
always @ (CS or Din)
begin
case (CS)
  S0: begin
      Qout=1'b0;
      if (Din==1'b0) NS=S0;
      else NS = S2;
    end
  S1: begin
      Qout=1'b1;
      if (Din==1'b0) NS=S0;
      else NS = S2;
    end
end
```

Next logic (Combinational circuit)

```
  S2: begin
      // fill the rest of the code
    end
  S3: begin
      // fill the rest of the code
    end
endcase
end // always(cs or din)
endmodule
```

MUX : Gate level



```
module mux2x1(f, s ,s_bar, a, b);
```

```
output f;
```

```
input s, s_bar;
```

```
input a, b;
```

```
wire nand1_out, nand2_out;
```

```
// boolean function
```

```
nand(nand1_out, s_bar, a);
```

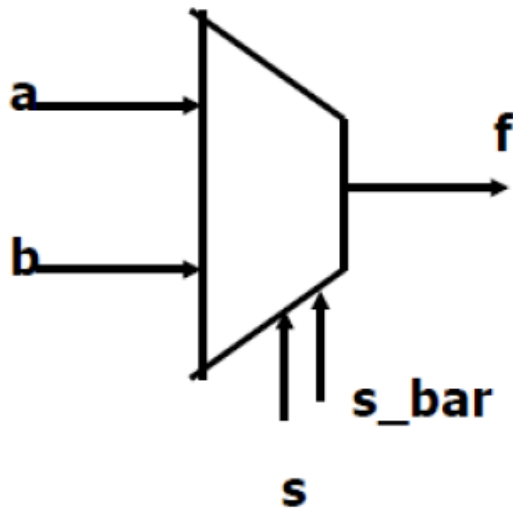
```
nand(nand2_out, s, b);
```

```
nand(f, nand1_out, nand2_out);
```

```
endmodule
```

MUX : Behavior

Assign $f = (s)?a:b;$



or

```
always(a or b or s) begin
    if(s==1)
        f=a;
    else
        f=b;
end
```

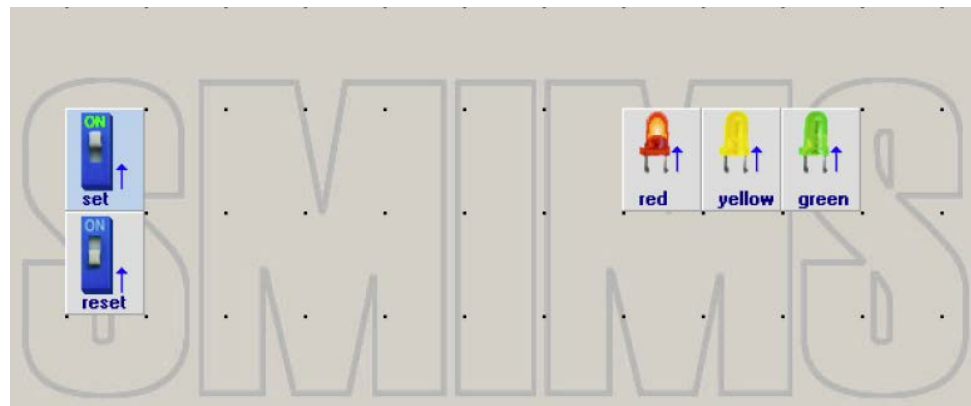

實作題

本次實作分為兩個基本實作與一個挑戰題

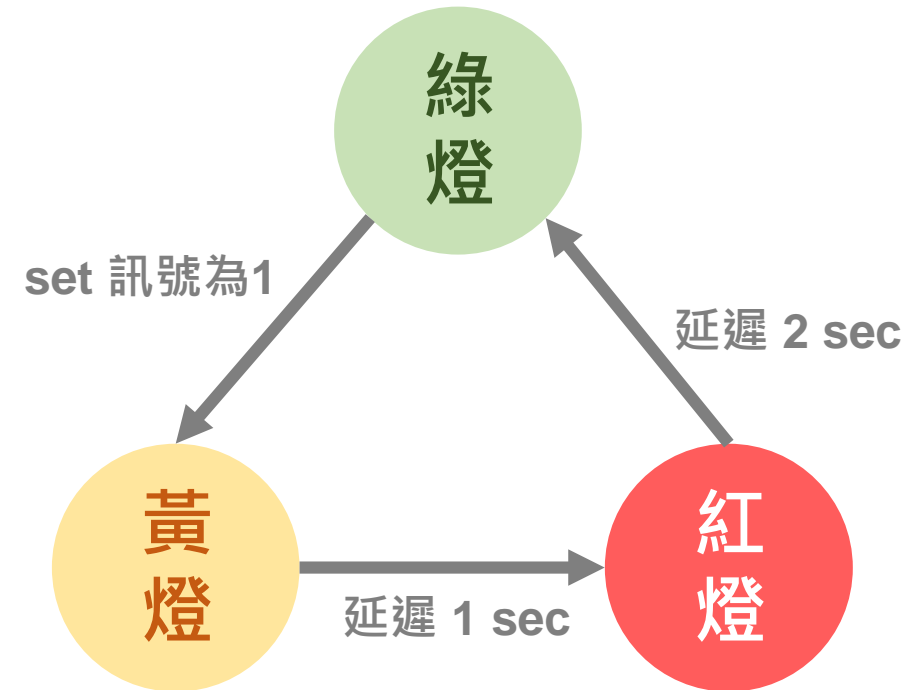
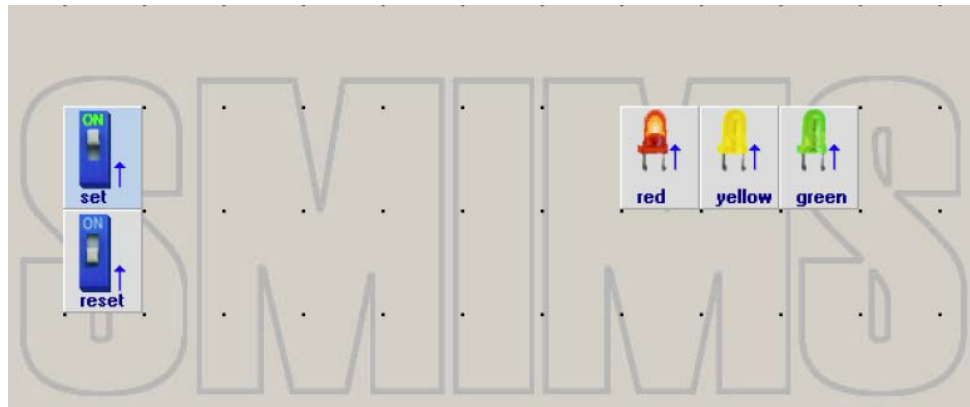
實作題(一) Moore Machine (1/4)

➤ 請利用狀態機寫出紅綠燈控制器

- 包含一個啟動按鈕(set)，按下按鈕時紅綠燈會從綠燈轉為黃燈，紅燈，再轉回綠燈，另有一初始化按鈕(reset)會將紅綠燈初始到綠燈的狀態。
- 使用VeriInstrument燒錄至FPGA，在虛擬儀表板上產生2 switch與三個LED燈
- 需要完成下列*.v檔
 - clock_div.v
 - lab11_1.v



實作題(一) Moore Machine (2/4)



實作題(一) Moore Machine (3/4)

```
module clock_div(clock, clock_4Hz);
    input clock;
    output clock_4Hz;

    reg [39:0] count_4Hz;
    reg clock_4Hz;

    always@(posedge clock)
    begin
        if(count_4Hz<40'd50000000)
            count_4Hz <= count_4Hz + 20'd1;
        else
            count_4Hz <= 20'd1;
    end
    always@(posedge clock)
    begin
        if(count_4Hz <= 40'd25000000)
            clock_4Hz <= 1'd1;
        else
            clock_4Hz <= 1'd0;
    end
end
endmodule
```

clock_div.v

實作題(一) Moore Machine (4/4)

```

module lab11_1(clock, reset, set, green, yellow, red);

    input  clock;
    input  reset;
    input  set;

    output green;
    output yellow;
    output red;

    reg [1:0]  state, next_state;
    reg       green, yellow, red;
    reg [4:0]  count;
    reg       reset_count;

    clock_div  clock_div(
        .clock(clock),
        .clock_4Hz(clock_4Hz)
    );

    always@(posedge clock_4Hz)
        state <= next_state;

```

```

always@(state or reset or set or count)
begin
    if(reset == 1'b1)
        begin
            next_state = 2'd0;
        end
    else
        begin
            case(state)
            2'd0:
                if(set == 1'b1)
                    next_state=2'd1;
                else
                    next_state=2'd0;
            2'd1:
                if( count==5'd4 )
                    next_state=2'd2;
                else
                    next_state=2'd1;
            2'd2:
                if( count==5'd8 )
                    next_state=2'd0;
                else
                    next_state=2'd2;
            default:
                next_state = 2'd0;
            endcase
        end
    end
end

```

```

always@(state)
begin
    case(state)
    2'd0:  begin
            green  =1'b1;
            yellow =1'b0;
            red    =1'b0;
        end
    2'd1:  begin
            green  =1'b0;
            yellow =1'b1;
            red    =1'b0;
        end
    2'd2:  begin
            green  =1'b0;
            yellow =1'b0;
            red    =1'b1;
        end
    default:begin
            green  =1'b0;
            yellow =1'b0;
            red    =1'b0;
        end
    endcase
end

```

```

always@(posedge clock_4Hz)
begin
    if(state!=next_state)
        reset_count <= 1'b1;
    else
        reset_count <= 1'b0;
end

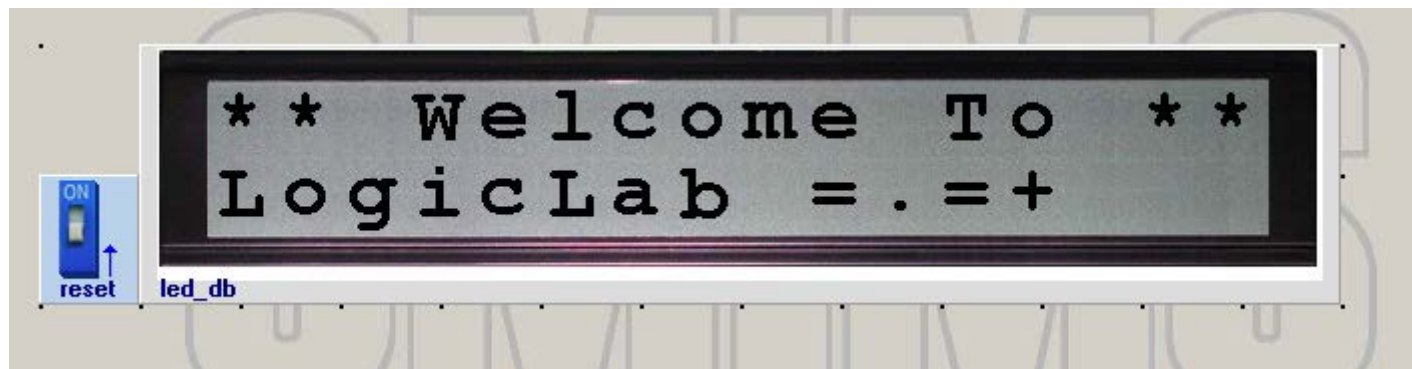
//計數器
always@(posedge clock_4Hz or posedge reset_count)
    if(reset_count == 1'b1)
        count <= 5'd0;
    else
        count <= count + 5'd1;
endmodule

```

lab11_1.v

實作題(二) LCD顯示器 (1/3)

- 請利用虛擬儀表版上的文字形LCD顯示模組逐字顯示一段字元
- 需使用一按鈕代表Reset，能清空LCD螢幕並初始化系統。
 - 使用VerilInstrument在虛擬儀表板上產生1個switch與1個文字形LCD顯示模組
 - 需要完成下列*.v檔
 - clock_div.v
 - lab11_2.v



實作題(二) LCD顯示器 (2/3)

接腳編號	訊號名稱	功能說明
1~8	DB0~DB7	資料匯流排(Data Bus)
9	RS	暫存器選擇訊號： 0：選擇指令暫存器(Write) 1：選擇資料暫存器(Write)
10	R/W	讀寫信號線： 0：寫入(Write) 1：無作用
11	EN	資料致能(Enable)訊號： 1→0：由High變Low時寫入資料才能有效轉移
12	RST	重置訊號 1：將LCD 畫面清空

LCD 的輸入接腳功能表

代碼	字型	代碼	字型	代碼	字型	代碼	字型	代碼	字型
0x00	空白	0x01	!	0x02	“	0x03	#	0x04	\$
0x05	%	0x06	&	0x07	‘	0x08	(0x09)
0x0A	*	0x0B	+	0x0C	,	0x0D	-	0x0E	.
0x0F	/	0x10	0	0x11	1	0x12	2	0x13	3
0x14	4	0x15	5	0x16	6	0x17	7	0x18	8
0x19	9	0x1A	:	0x1B	;	0x1C	<	0x1D	=
0x1E	>	0x1F	?	0x20	@	0x21	A	0x22	B
0x23	C	0x24	D	0x25	E	0x26	F	0x27	G
0x28	H	0x29	I	0x2A	J	0x2B	K	0x2C	L
0x2D	M	0x2E	N	0x2F	O	0x30	P	0x31	Q
0x32	R	0x33	S	0x34	T	0x35	U	0x36	V
0x37	W	0x38	X	0x39	Y	0x3A	Z	0x3B	[
0x3C	\	0x3D]	0x3E	^	0x3F	_	0x40	`
0x41	a	0x42	b	0x43	c	0x44	d	0x45	e
0x46	f	0x47	g	0x48	h	0x49	i	0x4A	j
0x4B	k	0x4C	l	0x4D	m	0x4E	n	0x4F	o
0x50	p	0x51	q	0x52	r	0x53	s	0x54	t
0x55	u	0x56	v	0x57	w	0x58	x	0x59	y
0x5A	z	0x5B	{	0x5C		0x5D	}	0x5E	~
0x5F	0x5F 到 0xFF 為空白								

Ascii 編碼對照表

實作題(二) LCD顯示器 (3/3)

```

module lab112(clock, reset, lcd_db, lcd_rs, lcd_rw, lcd_en, lcd_rst);

    input        clock;
    input        reset;

    output [7:0] lcd_db;
    output       lcd_rs;
    output       lcd_rw;
    output       lcd_en;
    output       lcd_rst;

    reg  [4:0]    count;
    reg  [7:0]    lcd_db;

    //請加入一除頻電路clock_div模組，並修改之

    assign  lcd_rst  = reset;
    assign  lcd_rs   = 1'b1;
    assign  lcd_rw   = 1'b0;
    assign  lcd_en   = clock_2Hz;

    always @(posedge clock_2Hz)
    begin
        if(reset)
            count <= 5'd0;
        else
            count <= count+5'd1;
    end
end

```

```

// LCD的輸出的字元訊號 (用counter控制)
always @(count)
begin
    case(count[4:0])
        'h0 : lcd_db = 'h0A;    // *
        'h1 : lcd_db = 'h0A;    // *
        'h2 : lcd_db = 'h00;    // space
        'h3 : lcd_db = 'h37;    // W
        'h4 : lcd_db = 'h45;    // e
        'h5 : lcd_db = 'h4C;    // l
        'h6 : lcd_db = 'h43;    // c
        'h7 : lcd_db = 'h4F;    // o
        'h8 : lcd_db = 'h4D;    // m
        'h9 : lcd_db = 'h45;    // e
        'hA : lcd_db = 'h00;    // space
        'hB : lcd_db = 'h34;    // T
        'hC : lcd_db = 'h4F;    // o
        'hD : lcd_db = 'h00;    // space
        'hE : lcd_db = 'h0A;    // *
        'hF : lcd_db = 'h0A;    // *
    endcase
end

```

```

'h10 : lcd_db = 'h2C; // L
'h11 : lcd_db = 'h4F; // o
'h12 : lcd_db = 'h47; // g
'h13 : lcd_db = 'h49; // i
'h14 : lcd_db = 'h43; // c
'h15 : lcd_db = 'h2C; // L
'h16 : lcd_db = 'h41; // a
'h17 : lcd_db = 'h42; // b
'h18 : lcd_db = 'h00; // space
'h19 : lcd_db = 'h1D; // =
'h1A : lcd_db = 'h0E; // .
'h1B : lcd_db = 'h1D; // =
'h1C : lcd_db = 'h0B; // +
'h1D : lcd_db = 'h00; // space
'h1E : lcd_db = 'h00; // space
default : lcd_db = 'h00;
endcase
end
endmodule

```

lab11_2.v

挑戰題

- 修改Lab11_2，改為使用按鈕控制LCD 文字顯示，每按一次就切換顯示文字。
- 新增以下兩個verilog檔
 - one_shot.v
 - lab11_3.v