

數位系統技術



Traffic Lights

Ren-Der Chen (陳仁德)

Department of Computer Science and
Information Engineering

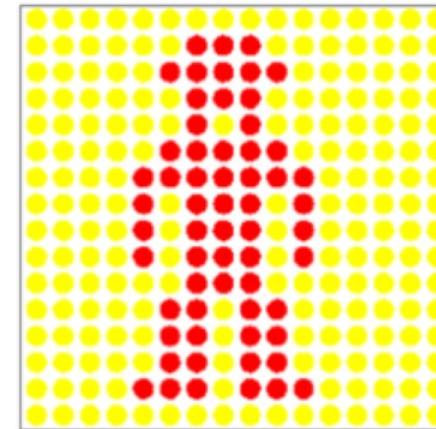
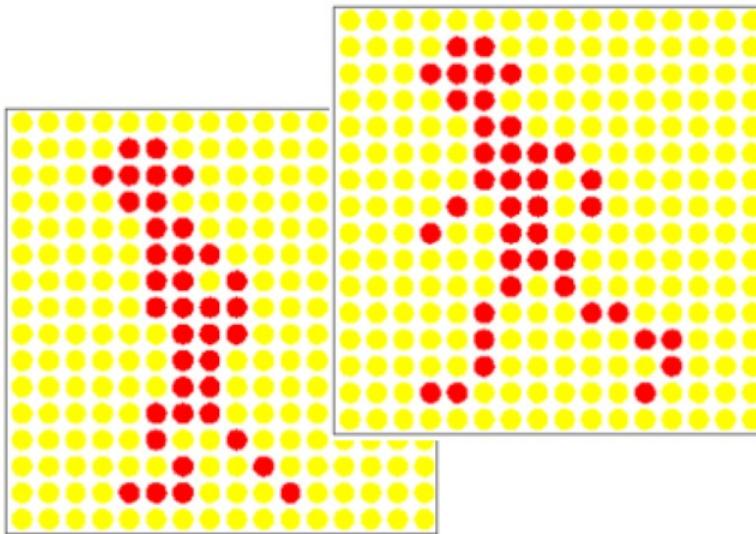
National Changhua University of Education

E-mail: rdchen@cc.ncue.edu.tw

Spring, 2025

實驗內容

- 設計紅綠燈電路，綠燈時產生小綠人行走動畫，紅燈時產生小紅人靜止圖形。

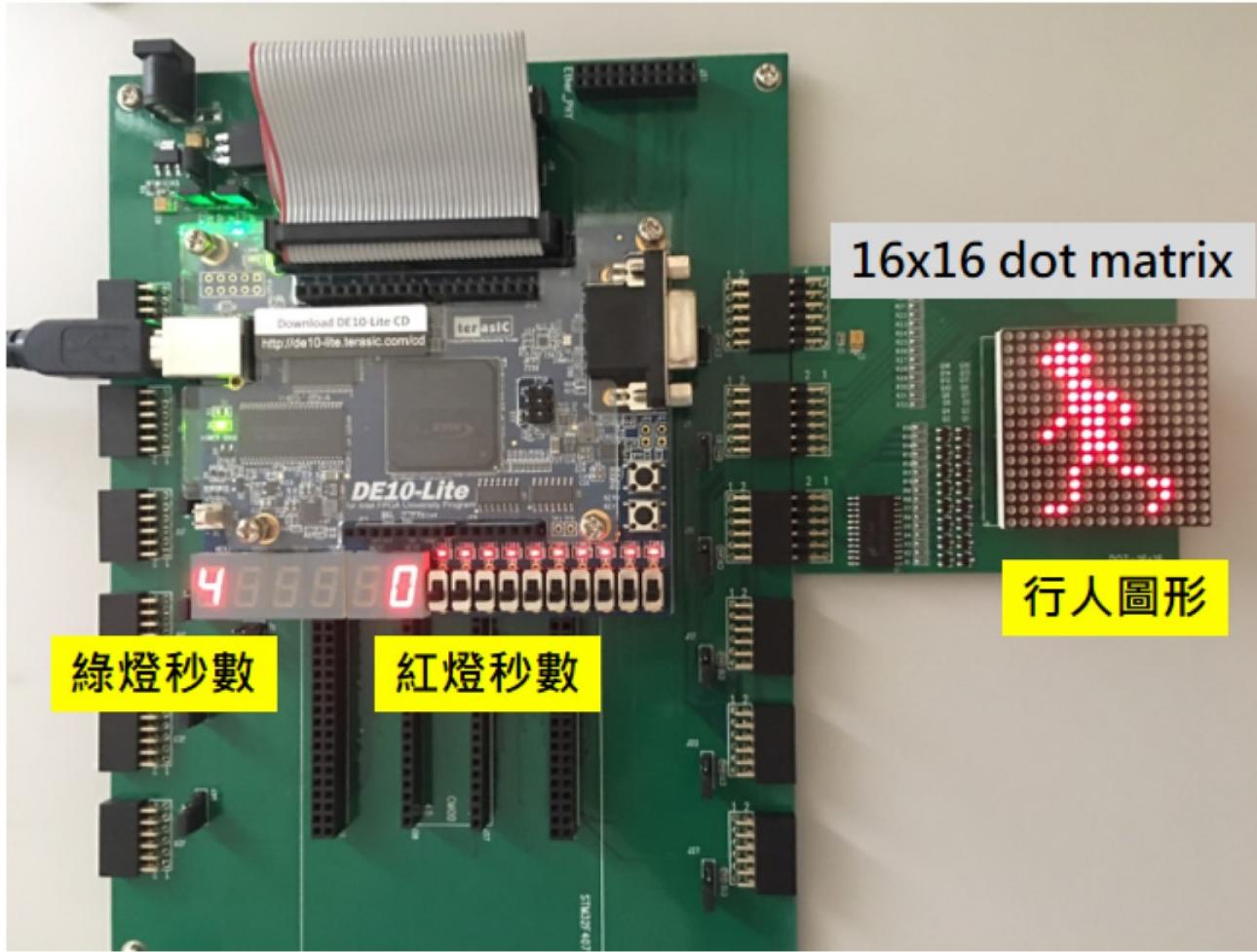


綠燈倒數秒數: 9, 8, 7,
6, 5, 4, 3, 2, 1, 0, 一開
始9秒時綠燈動畫開始,
0秒時切換紅燈倒數。

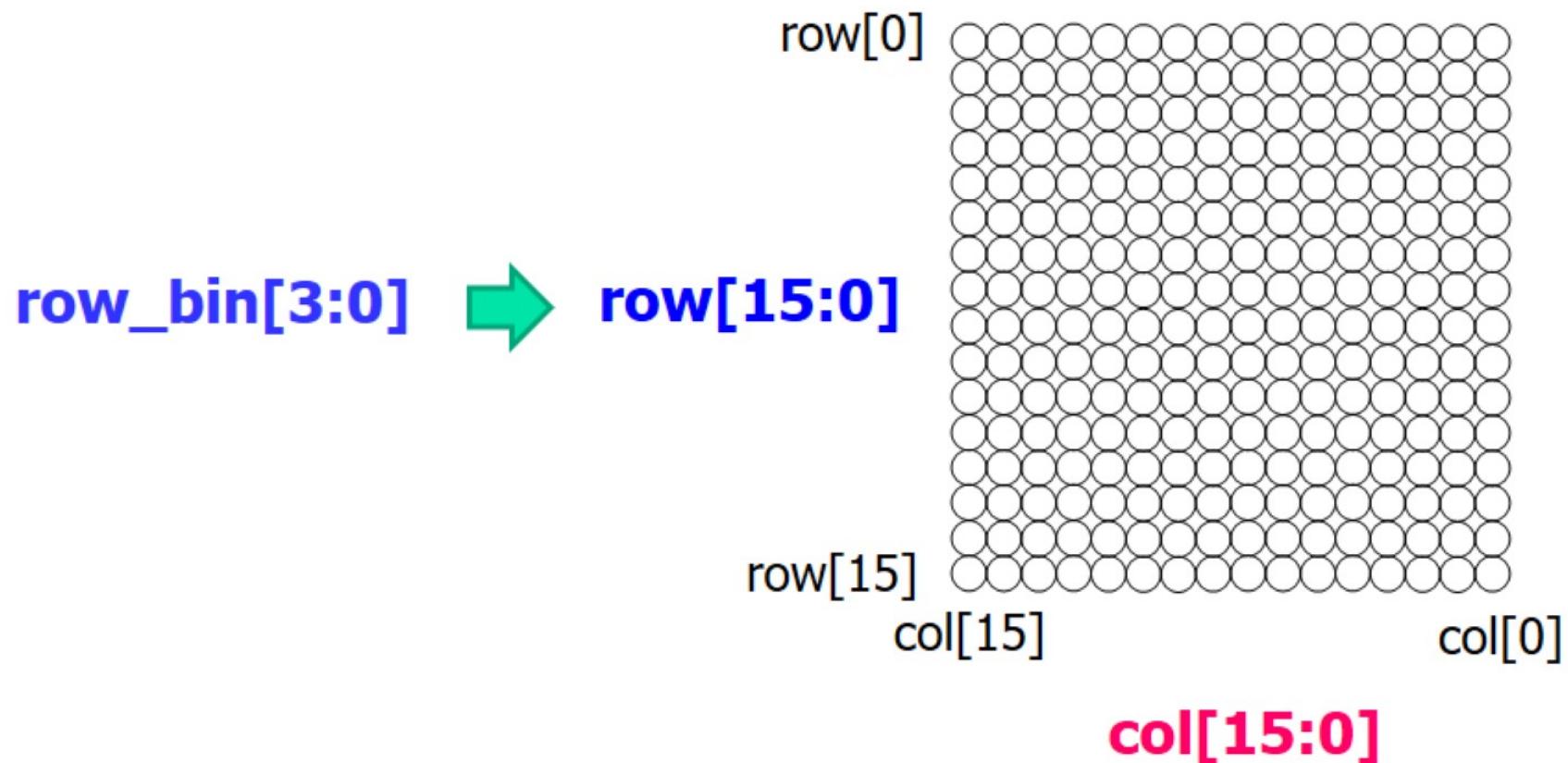


紅燈倒數秒數: 5, 4, 3,
2, 1, 0, 5秒紅燈開始
時呈現靜止圖案, 0秒
時切換綠燈倒數。

預期結果



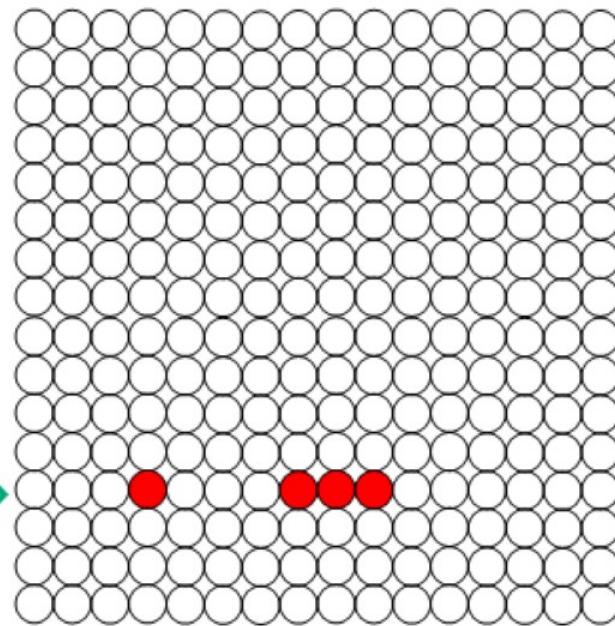
16x16 Dot Matrix (1/2)



16x16 Dot Matrix (2/2)

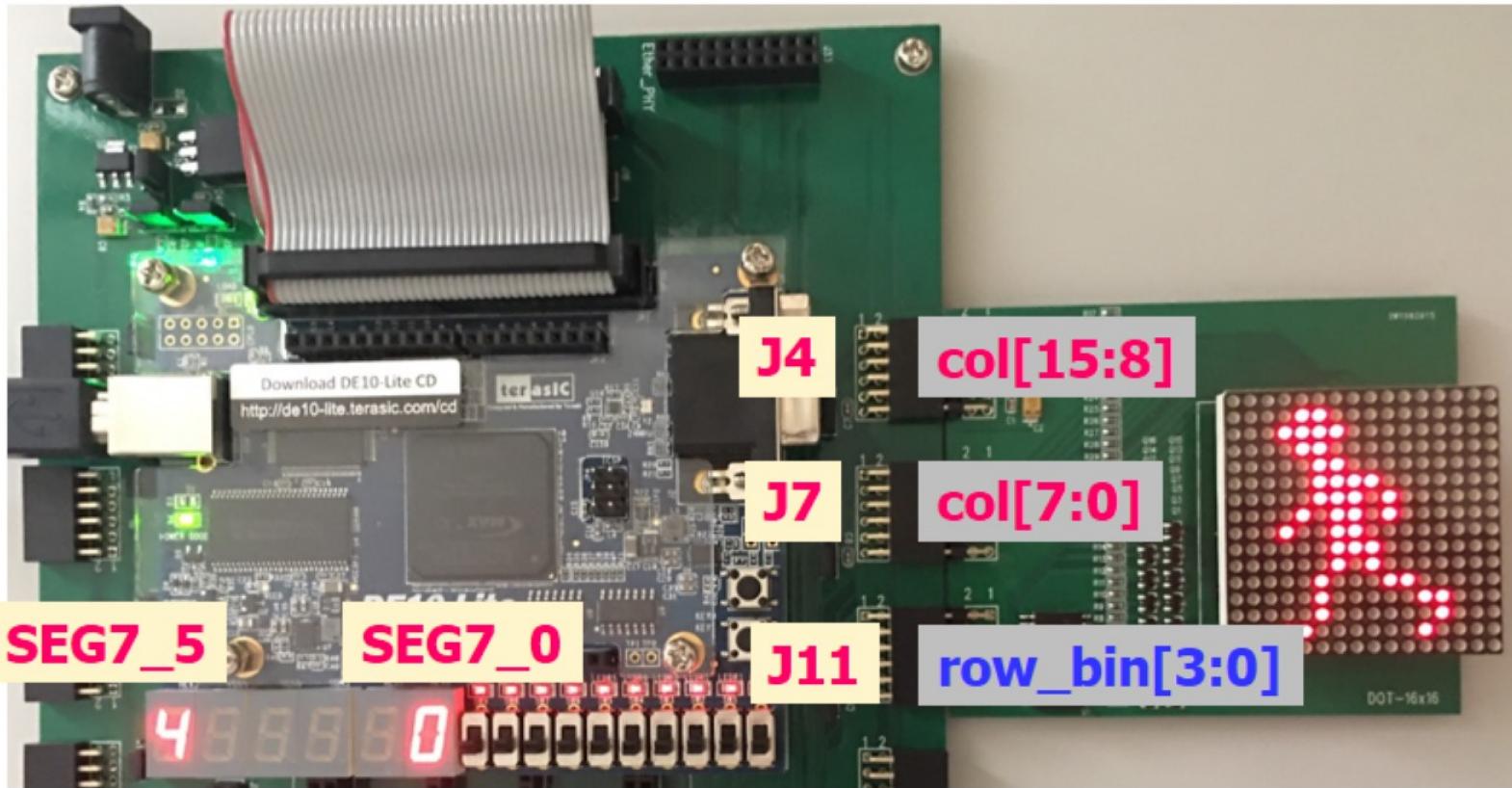
row_bin[3:0] = 4'b1100

row[12] →



col = 16'b0001_0001_1100_0000

Port Mapping



Design 1: row_sel.v

- 設計row掃描電路row_sel.v，此時之clk為row掃描時脈，使用時要先除頻。

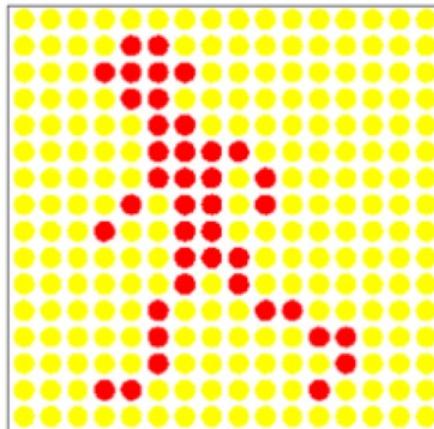
```
module row_sel(clk, rst, row_bin);
    input clk, rst;
    output [ ] row_bin;

    always@(posedge clk or posedge rst) begin
        if (rst)
            row_bin = [ ] // top-most 初始值，選取最上方row
        else // shift down
            row_bin = [ ];
    end
endmodule
```

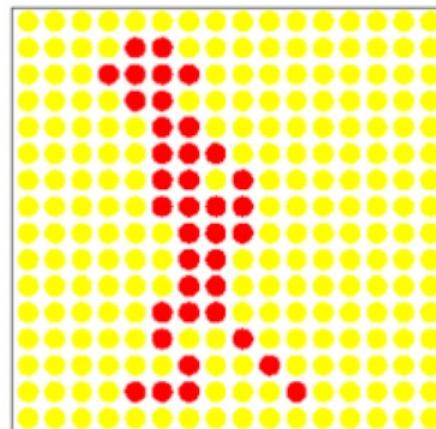
row_bin之值遞增(0~15)，依序選取下方row

Design 2

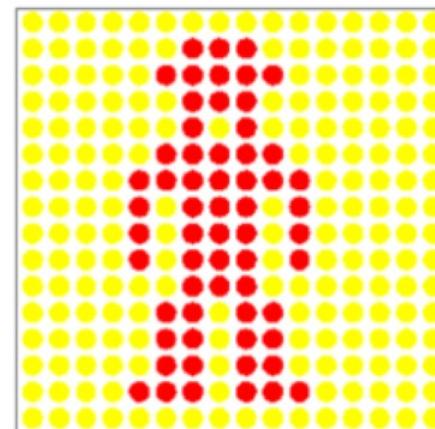
- 設計小綠人行走圖形編碼pattern.v與pattern2.v，及小紅人靜止圖形編碼pattern3.v。



pattern



pattern2



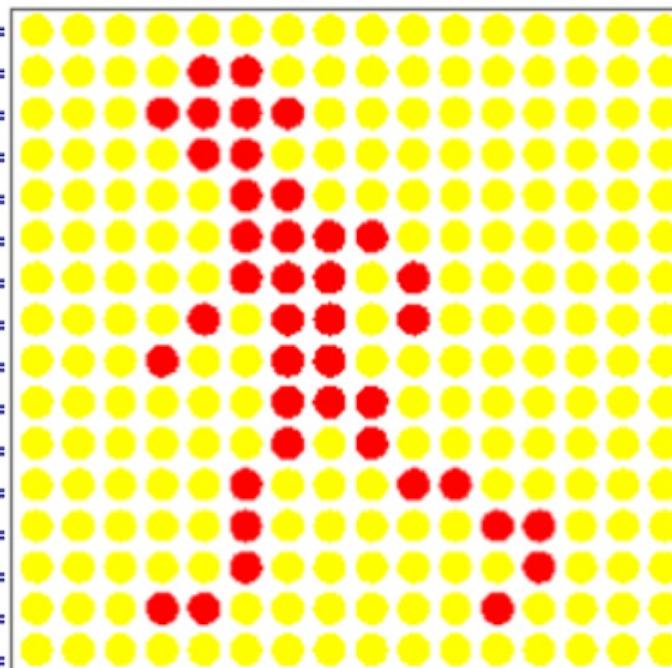
pattern3

pattern.v

```
module pattern(row_bin, col);
    input [3:0] row_bin;
    output reg [15:0] col;

    always@(*) begin
        case(row_bin)
            4'd0: col= 4'b0000;
            4'd1: col= 4'b0001;
            4'd2: col= 4'b0010;
            4'd3: col= 4'b0011;
            4'd4: col= 4'b0100;
            4'd5: col= 4'b0101;
            4'd6: col= 4'b0110;
            4'd7: col= 4'b0111;
            4'd8: col= 4'b1000;
            4'd9: col= 4'b1001;
            4'd10: col= 4'b1010;
            4'd11: col= 4'b1011;
            4'd12: col= 4'b1100;
            4'd13: col= 4'b1101;
            4'd14: col= 4'b1110;
            4'd15: col= 4'b1111;
        default: col=16'b0000000000000000;
        endcase
    end
endmodule
```

小綠人行走圖形編碼



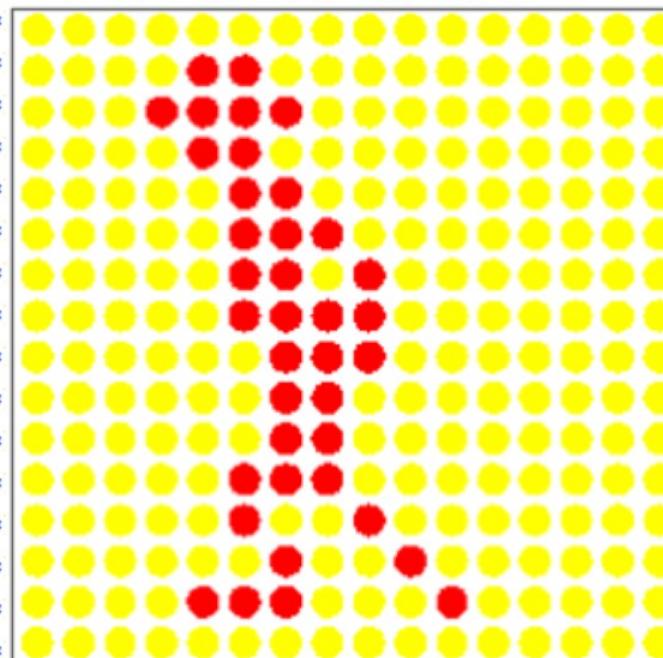
紅色為1
黃色為0

pattern2.v

```
module pattern2(row_bin, col);
    input [3:0] row_bin;
    output reg [15:0] col;

    always@(*) begin
        case(row_bin)
            4'd0: col= 0;
            4'd1: col= 0;
            4'd2: col= 0;
            4'd3: col= 0;
            4'd4: col= 0;
            4'd5: col= 0;
            4'd6: col= 0;
            4'd7: col= 0;
            4'd8: col= 0;
            4'd9: col= 0;
            4'd10: col= 0;
            4'd11: col= 0;
            4'd12: col= 0;
            4'd13: col= 0;
            4'd14: col= 0;
            4'd15: col= 0;
        default: col=16'b0000000000000000;
    endcase
end
endmodule
```

小綠人行走圖形編碼2



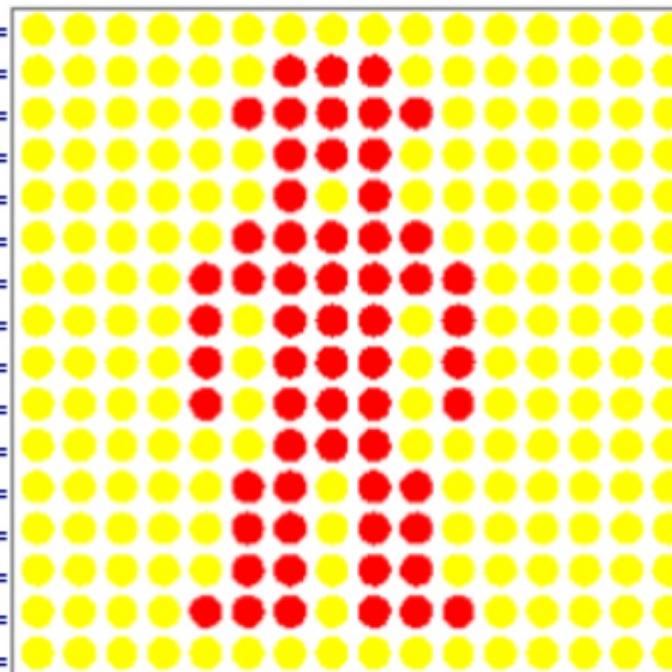
紅色為1
黃色為0

pattern3.v

```
module pattern3(row_bin, col);
    input [3:0] row_bin;
    output reg [15:0] col;

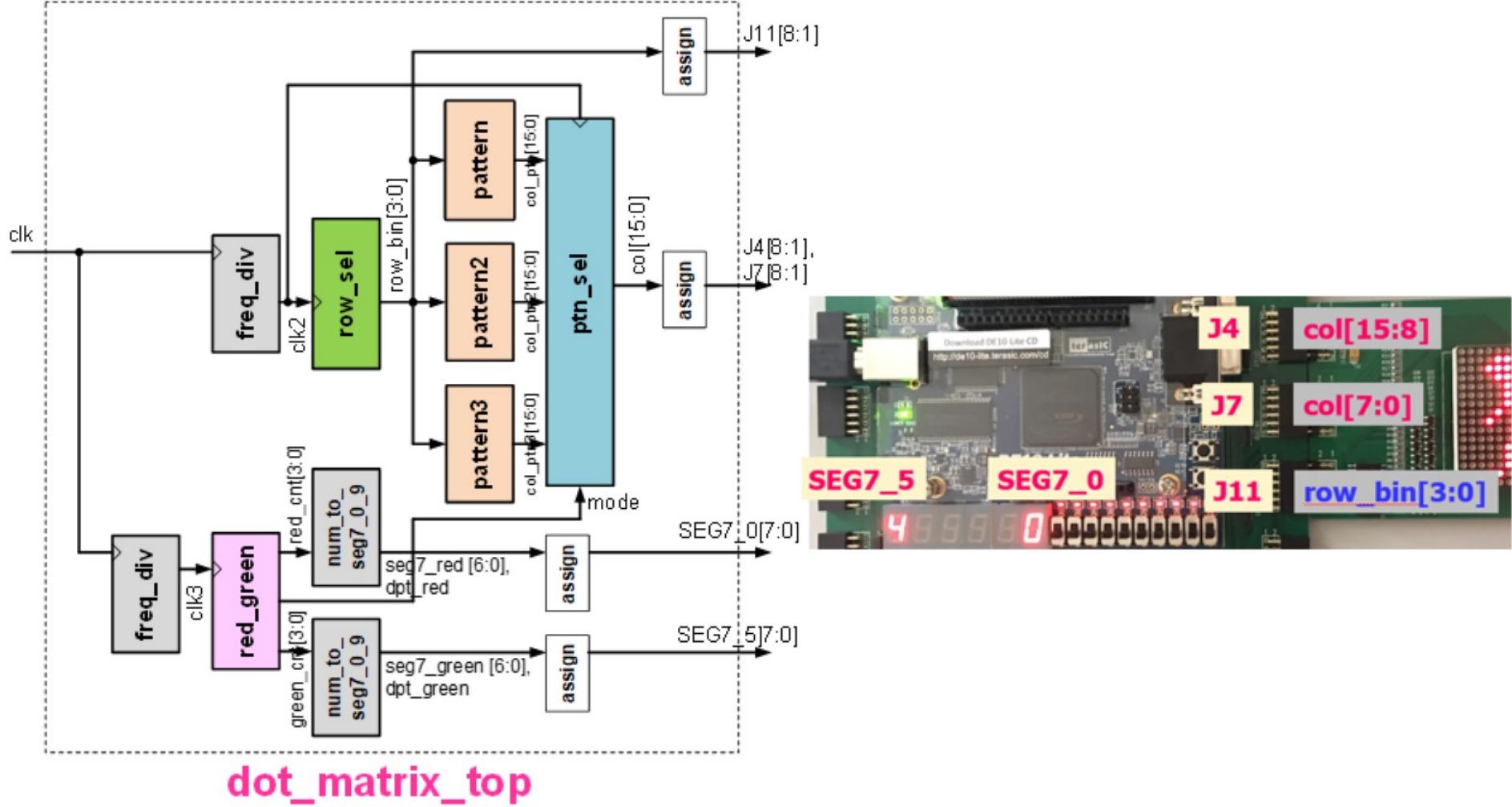
    always@(*) begin
        case(row_bin)
            4'd0: col= 4'b0000;
            4'd1: col= 4'b0001;
            4'd2: col= 4'b0010;
            4'd3: col= 4'b0011;
            4'd4: col= 4'b0100;
            4'd5: col= 4'b0101;
            4'd6: col= 4'b0110;
            4'd7: col= 4'b0111;
            4'd8: col= 4'b1000;
            4'd9: col= 4'b1001;
            4'd10: col= 4'b1010;
            4'd11: col= 4'b1011;
            4'd12: col= 4'b1100;
            4'd13: col= 4'b1101;
            4'd14: col= 4'b1110;
            4'd15: col= 4'b1111;
        default: col=16'b000000000000000000000000;
    endcase
end // always
endmodule
```

小紅人靜止圖形編碼



紅色為1
黃色為0

dot_matrix_top 電路方塊圖



Design 3: ptn_sel.v (1/2)

- 依據目前紅綠燈mode，產生對應圖形之col值，此時之clk為row掃描時脈，使用時要先除頻。

```
module ptn_sel(clk, rst, mode, col_ptn, col_ptn2, col_ptn3, col);
    input clk, rst;
    input mode;
    input [15:0] col_ptn, col_ptn2, col_ptn3;

    output [15:0] col;
    reg [8:0] ptn_cnt; ptn_cnt: 紀錄綠燈時, 每一圖形之停留時間(0~255), 以row掃描週期為計數單位
    reg ptn_flag;
```

mode=0: 綠燈模式
mode=1: 紅燈模式
col_ptn: 綠燈行走圖形
col_ptn2: 綠燈行走圖形2
col_ptn3: 紅燈停止圖形

```
always@ (posedge clk or posedge rst) begin
    if (rst) begin
        ptn_cnt = 8'b0;
        ptn_flag = 1'b0;
        col = col_ptn;
    end
end
```

ptn_flag: 紀錄綠燈時, 目前所輸出之圖形
ptn_flag=0: 送出col_ptn
ptn_flag=1, 送出col_ptn2

Design 3: ptn_sel.v (2/2)

```
else begin
    if (mode == 1'b0) begin // green mode
        ptn_cnt = [ ]           // from 0 to 255
                                圖形之停留時間(0~255)遞增1
        if (ptn_cnt == 8'b0)
            ptn_flag = [ ]      當圖形之顯示時間
                                到達255, 切換圖形

        col = (ptn_flag==1'b0) [ ]
    end
    else if (mode == 1'b1) // red mode
        col = [ ]             依據目前ptn_flag之值,
                                產生對應圖形之col值,
                                col_ptn or col_ptn2?
end
end
endmodule
```

紅燈時, 僅顯示一種圖形col_ptn3

Design 4: red_green.v (1/3)

- 設計具2種倒數模式之紅綠燈電路red_green.v，此時之clk為燈號倒數時脈。
 - mode=0: 綠燈倒數模式 (秒數: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)
 - mode=1: 紅燈倒數模式 (秒數: 5, 4, 3, 2, 1, 0)

Design 4: red_green.v (2/3)

```
module red_green(clk_light_cnt, rst,
                  red_cnt, green_cnt, mode);
  // default time for each mode
  parameter GREEN_TIME = 4'd9; 綠燈開始倒數秒數
  parameter RED_TIME = 4'd5;   紅燈開始倒數秒數

  input clk_light_cnt, rst;
  output reg [3:0] red_cnt, green_cnt;
  output reg mode;

  always@(posedge clk_light_cnt or posedge rst) begin
    if (rst) begin
      mode = 1'b0; // green mode
      green_cnt = GREEN_TIME; // initial time for green
      red_cnt = 4'd0; // initial time for red
    end
  end
```

clk_light_cnt: 燈號倒數時脈,
決定倒數的速度
red_cnt: 目前紅燈秒數
green_cnt: 目前綠燈秒數

mode=0: 綠燈模式
mode=1: 紅燈模式

綠燈由
GREEN_TIME開始
倒數

Design 4: red_green.v (3/3)

```
else begin
    if (mode == 1'b0) begin // green mode
        green_cnt = [REDACTED]
        if (green_cnt == [REDACTED]) begin
            mode = [REDACTED] // to red mode
            red_cnt = [REDACTED] // set time for red
        end
    end
    else if (mode == 1'b1) begin // red mode
        red_cnt = [REDACTED]
        if (red_cnt == [REDACTED]) begin
            mode = [REDACTED] // to green mode
            green_cnt = [REDACTED] // set time for green
        end
    end
end
endmodule
```

1. 綠燈秒數遞減
2. 判斷是否倒數結束
3. 若是則切換至紅燈模式
4. 設定紅燈秒數值為RED_TIME

1. 紅燈秒數遞減
2. 判斷是否倒數結束
3. 若是則切換至綠燈模式
4. 設定綠燈秒數值為GREEN_TIME

Design 5: dot_matrix_top.v (1/4)

- 設計dot_matrix_top.v，整合所有電路模組。

```
module dot_matrix_top (clk, rst, J4, J7, J11, SEG7_5, SEG7_0);
    input clk, rst;
    output [8:1] J4, J7;
    output [8:1] J11;
    output [7:0] SEG7_5, SEG7_0;

    wire [15:0] col;
    wire [3:0] row_bin;
```

Design 5: dot_matrix_top.v (2/4)

```
// col: J4 & J7
assign J4[1] = ~col[15];
assign J4[3] = ~col[14];
assign J4[5] = ~col[13];
assign J4[7] = ~col[12];
assign J4[2] = ~col[11];
assign J4[4] = ~col[10];
assign J4[6] = ~col[9];
assign J4[8] = ~col[8];

assign J7[1] = ~col[7];
assign J7[3] = ~col[6];
assign J7[5] = ~col[5];
assign J7[7] = ~col[4];
assign J7[2] = ~col[3];
assign J7[4] = ~col[2];
assign J7[6] = ~col[1];
assign J7[8] = ~col[0];

// row: J11
assign J11[1] = row_bin[0];
assign J11[3] = row_bin[1];
assign J11[5] = row_bin[2];
assign J11[7] = row_bin[3];
```

Design 5: dot_matrix_top.v (3/4)

```
wire clk2, clk3;  
wire [15:0] col_ptn, col_ptn2, col_ptn3;  
wire [3:0] red_cnt, green_cnt;  
wire mode;
```



Design 5: dot_matrix_top.v (4/4)

```
freq_div #(25) m6  
red_green m7
```

```
wire [6:0] seg7_red;  
wire dpt_red;  
wire [6:0] seg7_green;  
wire dpt_green;
```

```
num_to_seg7_0_9 m8  
num_to_seg7_0_9 m9
```

```
assign SEG7_0 =  
assign SEG7_5 =
```

Pin Assignments

clk

Clocks: 50 MHZ

CLK1: P11

CLK2: N14

rst

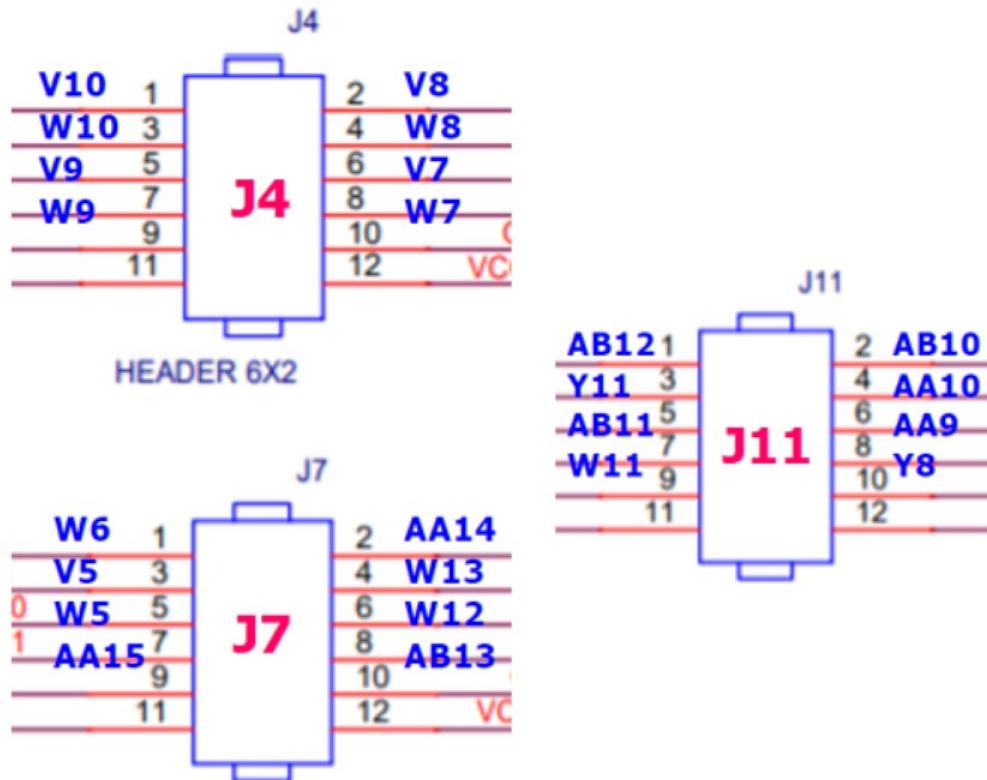


KEY0: B8

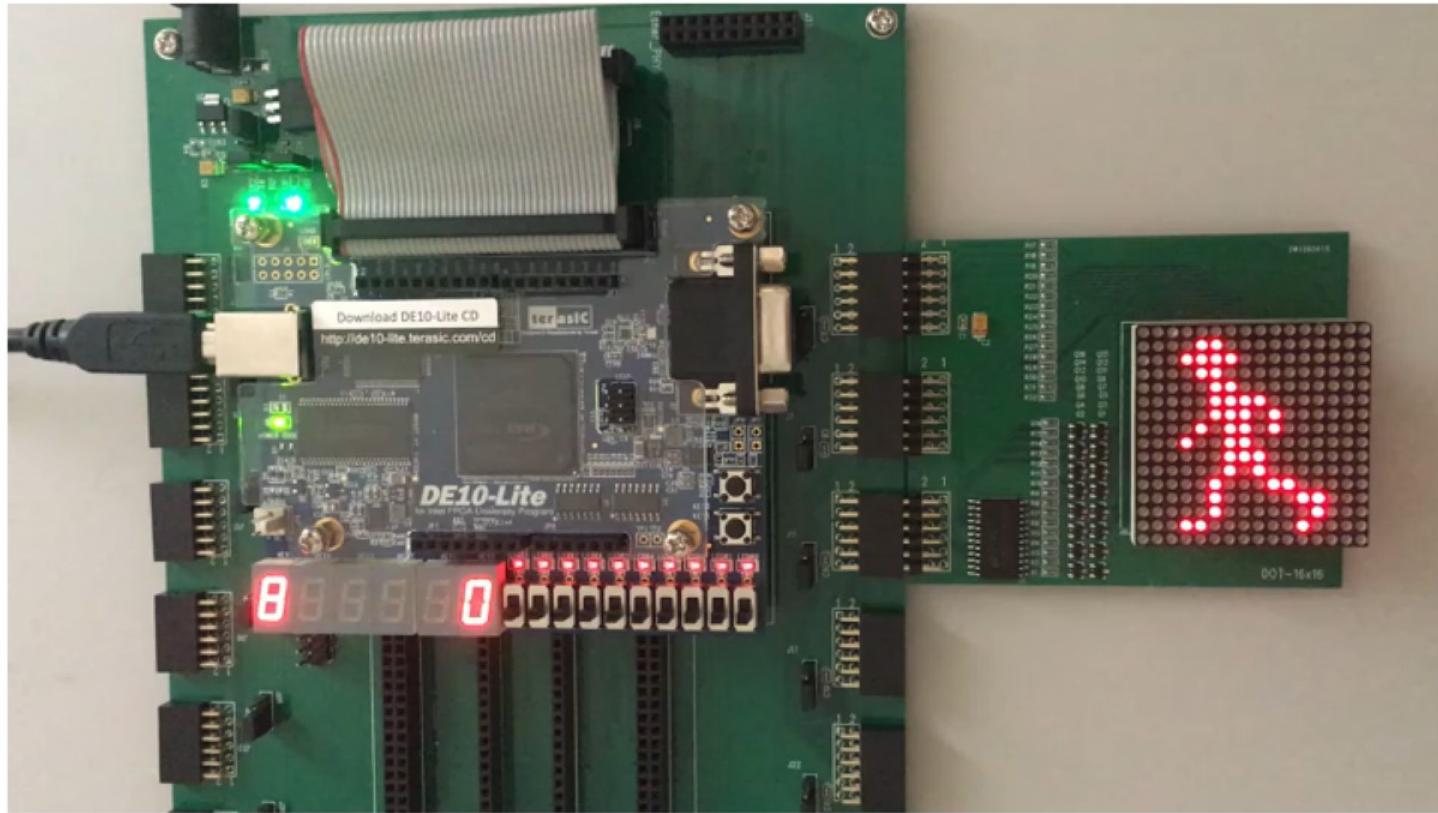
Push: 0

KEY1: A7

Not push: 1



執行結果



File List

■ Verilog files

- freq_div.v (二的次方之除頻電路)
- num_to_seg7_0_9.v (bcd轉SEG7編碼)
- row_sel.v (產生row_bin)
- pattern.v (小綠人行走圖形編碼)
- pattern2.v (小綠人行走圖形編碼2)
- pattern3.v (小紅人靜止圖形編碼)
- ptn_sel.v (產生col值)
- red_green.v (紅綠燈倒數秒數)
- dot_matrix_top.v (最上層top電路)

實驗結果驗收

- 請老師或助教驗收 dot_matrix_top 電路於實驗板顯示之圖形