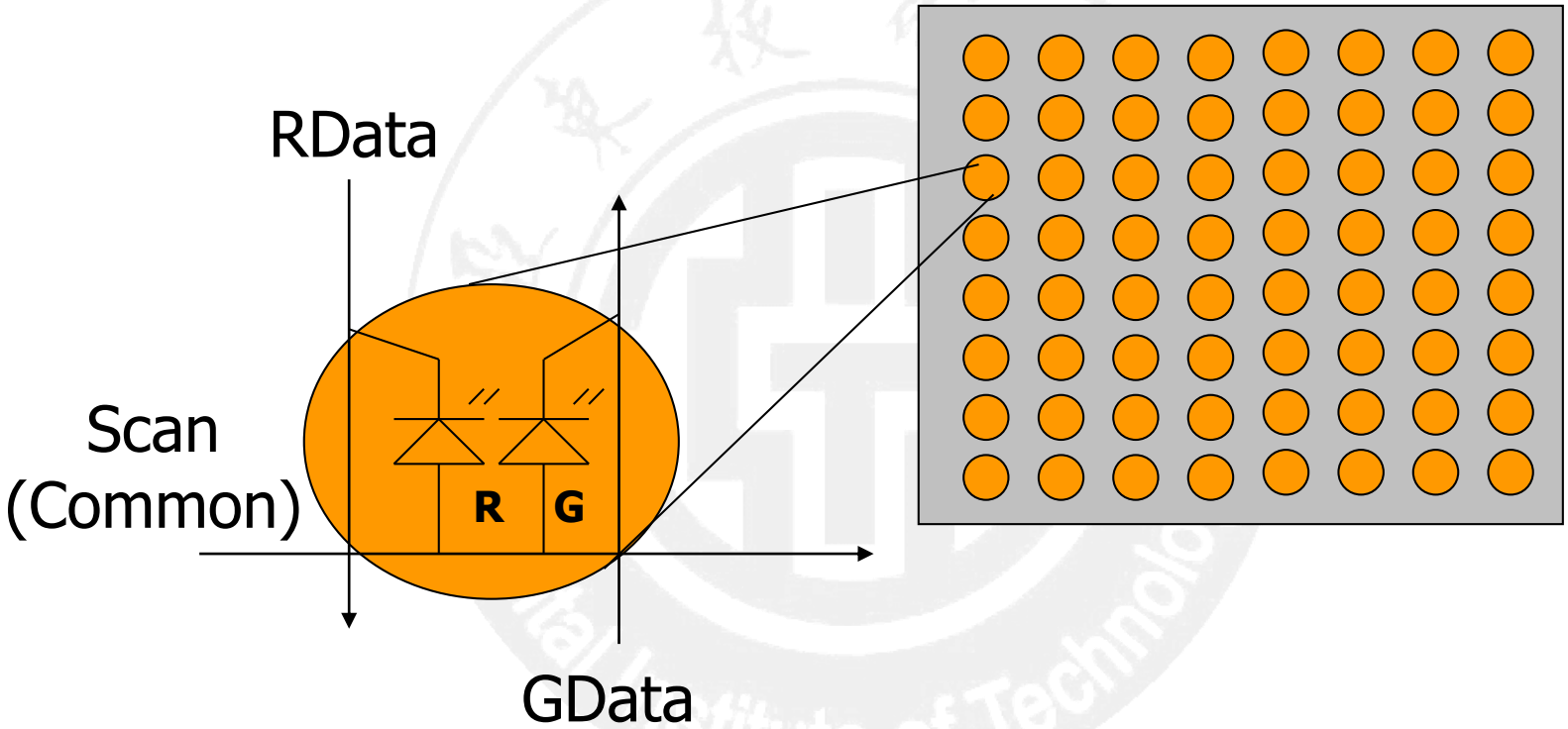
Design a 8x8 Dot Matrix Display

陳韋達

Outline

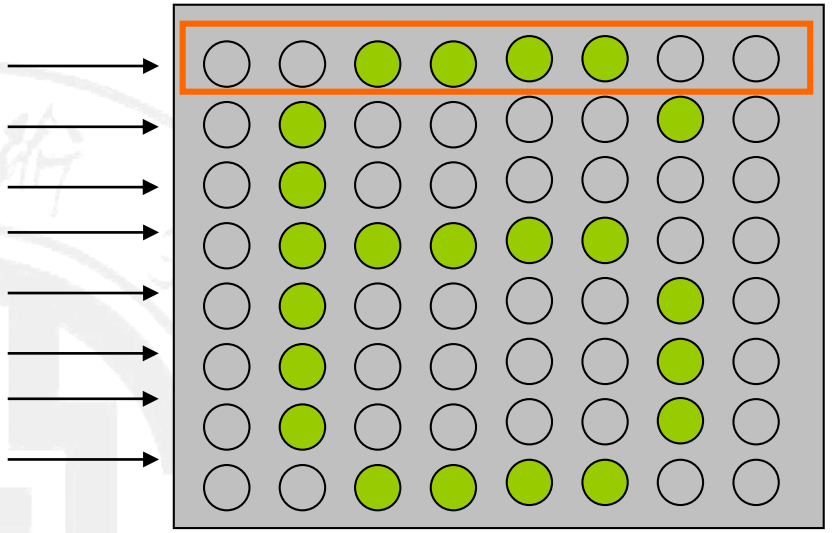
- Introduction to 8x8 Dot Matrix Display
  - How the dot matrix works
  - Scan controller and frame controller
  - Data mapping and data selected
  - Architecture block diagram
- Homework
  - Developing a simple traffic light for pedestrian
    - Combining 7-segment and dot matrix
  - sharpen your coding skills
  - More familiar with simulation tool

Introduction of Dot-matrix(1)



Introduction of Dot-matrix(2)

0	0	0	0	0	0	0	1	c1
0	0	0	0	0	0	1	0	c2
0	0	0	0	0	1	0	0	c3
0	0	0	0	1	0	0	0	c4
0	0	0	1	0	0	0	0	c5
0	0	1	0	0	0	0	0	c6
0	1	0	0	0	0	0	0	c7
1	0	0	0	0	0	0	0	c8



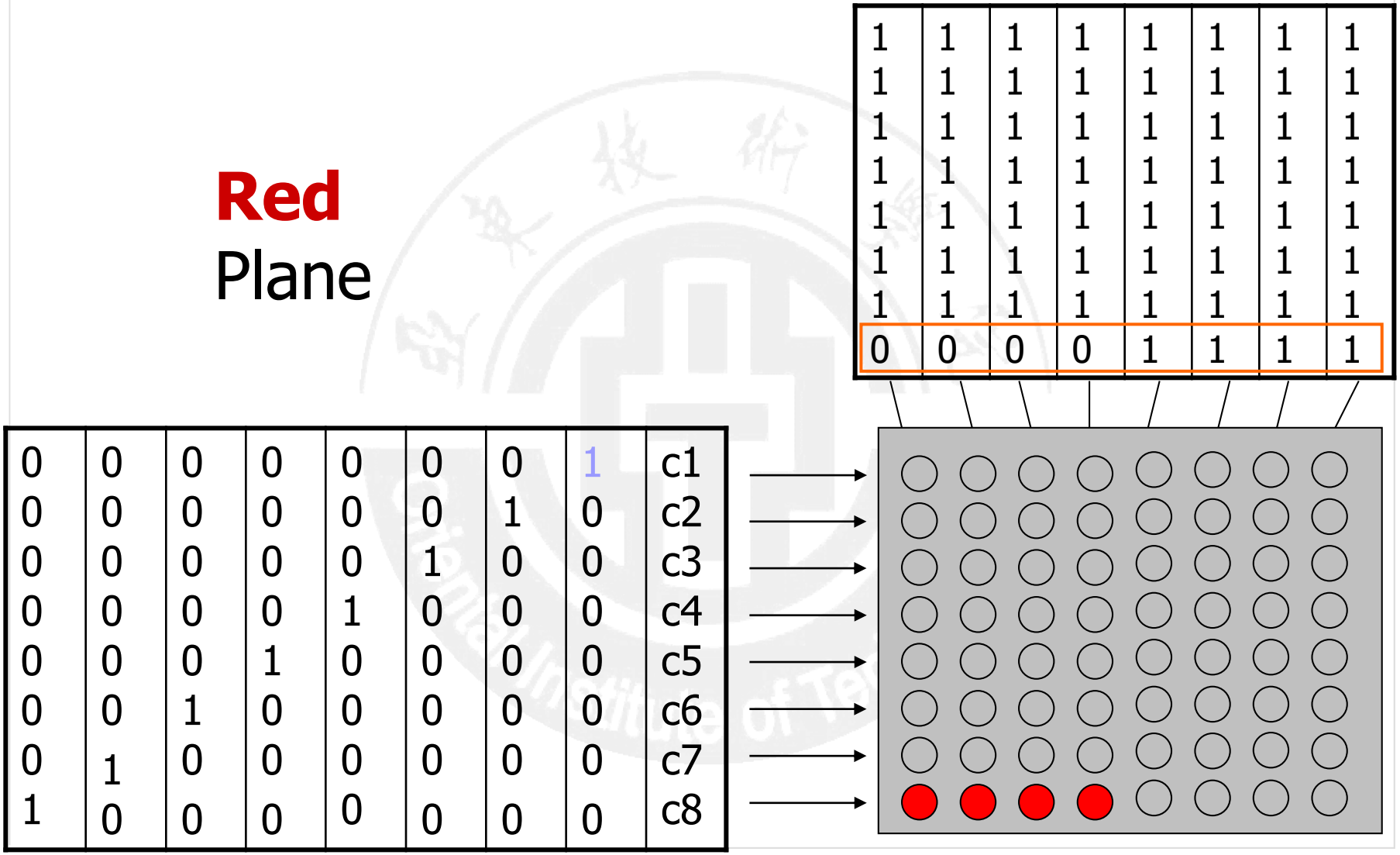
Green

Plane

1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	1
1	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1
1	0	1	1	1	1	0	1
1	0	1	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1

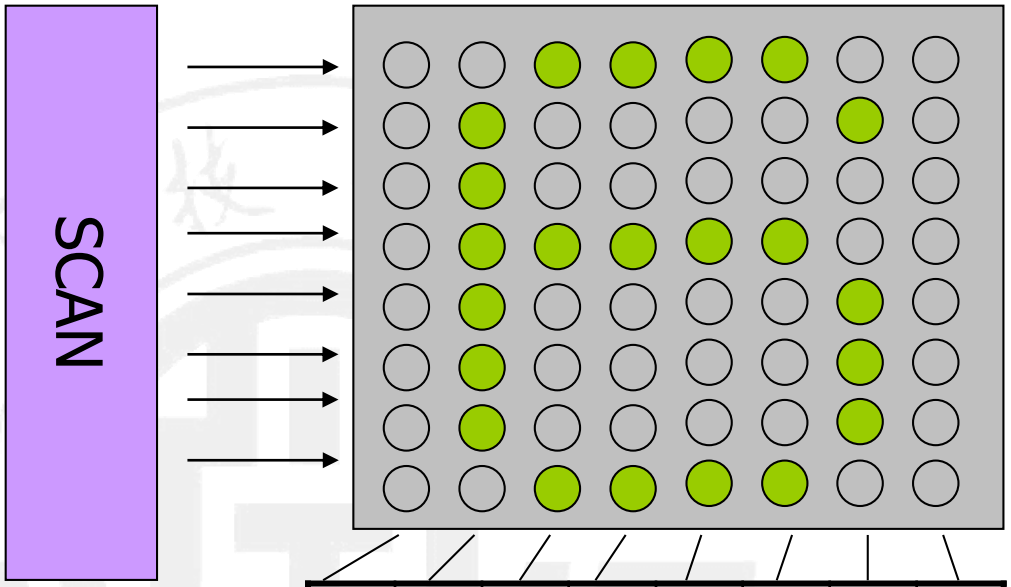
Introduction of Dot-matrix(3)

Red
Plane



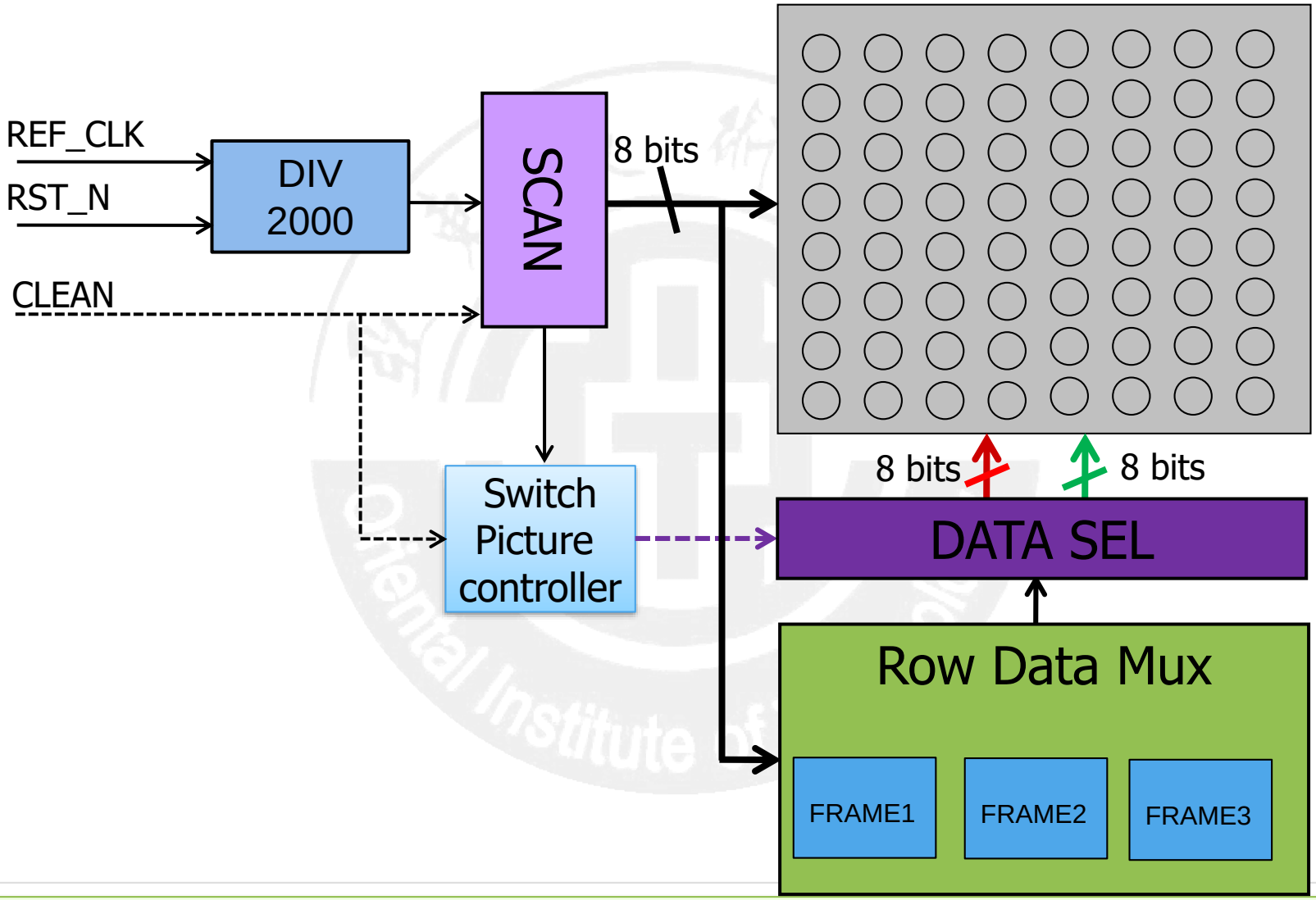
Introduction of Dot-matrix(4)

Green
Plane



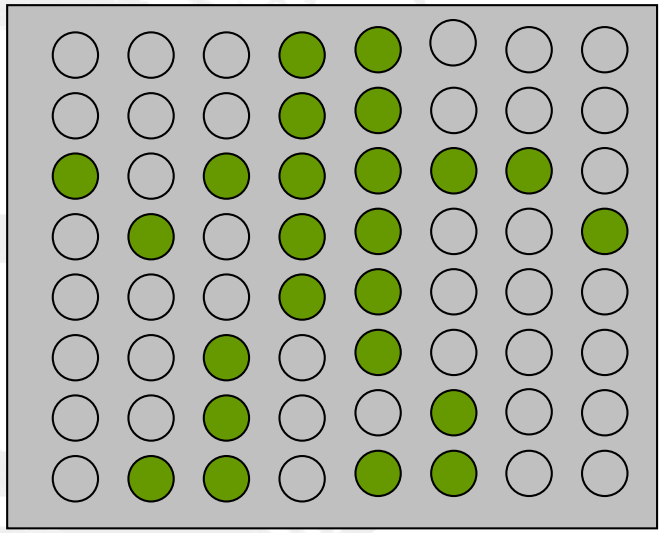
C3	1	1	0	0	0	0	1	1
BD	1	0	1	1	1	1	0	1
BF	1	0	1	1	1	1	1	1
83	1	0	0	0	0	0	1	1
BD	1	0	1	1	1	1	0	1
BD	1	0	1	1	1	1	0	1
BD	1	0	1	1	1	1	0	1
C3	1	1	0	0	0	0	1	1

Architecture



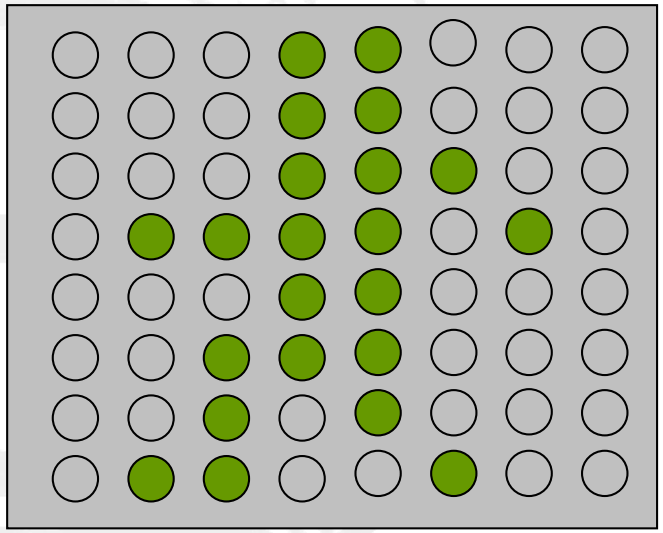
小綠人(1)

- FRAME #1 data
  - E7H
  - E7H
  - 41H
  - A6H
  - E7H
  - D7H
  - DBH
  - 93H



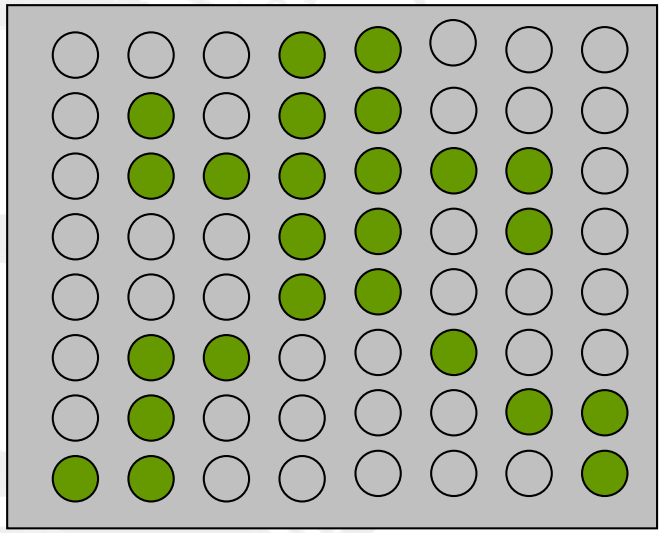
小綠人(2)

- FRAME #2 data
  - E7H
  - E7H
  - E3H
  - 85H
  - E7H
  - C7H
  - D7H
  - 9BH



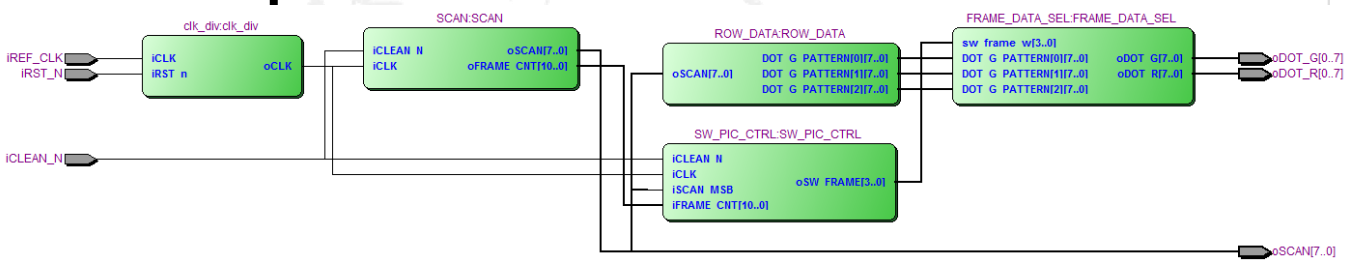
小綠人(3)

- FRAME #3 data
  - E7H
  - A7H
  - 81H
  - E5H
  - E7H
  - 9BH
  - BCH
  - 3EH



Top module- Dot matrix

```
module dot_matrix(
                  input iREF_CLK,iRST_N,iCLEAN_N,
                  output reg [0:7] oDOT_G,oDOT_R,
                  output     [7:0] oSCAN
                  );
 parameter DIV_CNT    = 1000;//freq=(50Mhz/(DIV_CNT*2))
 parameter FRAME_CNT  = 500;
 parameter PIC_CNT    = 3-1;
 wire clk_kHz;
 wire [0:7] DOT_G_PATTERN[2:0];
 wire [10:0] frame_cnt_w;
 wire [3:0] sw_frame_w;
 wire [7:0] scan_w;
 assign oSCAN=scan_w;
 clk_div #(.DIV_CNT(DIV_CNT))
clk_div(
          .iCLK(iREF_CLK),
          .iRST_n(iRST_N),
          .oCLK(clk_kHz)
        );

 SCAN #(.FRAME_CNT(FRAME_CNT))
SCAN(
       .iCLK(clk_kHz),
       .iCLEAN_N(iCLEAN_N),
       .oSCAN(scan_w),
       .oFRAME_CNT(frame_cnt_w)
    );

SW_PIC_CTRL #(.FRAME_CNT(FRAME_CNT),
              .PIC_CNT(PIC_CNT))
SW_PIC_CTRL(
             .iCLK(clk_kHz),
             .iCLEAN_N(iCLEAN_N),
             .iFRAME_CNT(frame_cnt_w),
             .iSCAN_MSB(scan_w[7]),
             .oSW_FRAME(sw_frame_w)
             );

FRAME_DATA_SEL FRAME_DATA_SEL(
                      .sw_frame_w(sw_frame_w),
                      .DOT_G_PATTERN(DOT_G_PATTERN),
                      .oDOT_G(oDOT_G),
                      .oDOT_R(oDOT_R)
                      );

ROW_DATA ROW_DATA(
                  .oSCAN(scan_w),
                  .DOT_G_PATTERN(DOT_G_PATTERN)
                );
 endmodule
```



Clk div and SCAN modules

```
module clk_div(
                iCLK,iRST_n,
                oCLK
                );

input       iCLK,iRST_n;
output reg  oCLK;

parameter DIV_CNT = 500;

reg [31:0] div;


always@(posedge iCLK)
   if(iRST_n==1'd0)
      begin
         oCLK<=1'd0;
         div<=32'd0;
      end
   else
      begin
         if(div==DIV_CNT)
            begin
               oCLK<=~oCLK;
               div<=32'd0;
            end
         else
            begin
               div<=div+32'd1;
            end
      end

endmodule
```

```
module SCAN(
            input iCLK,iCLEAN_N,
            output reg [7:0] oSCAN,
            output     [10:0] oFRAME_CNT
            );

parameter FRAME_CNT = 1;

reg [10:0] frame_cnt;

assign oFRAME_CNT = frame_cnt;

always@(posedge iCLK)
   if(!iCLEAN_N)
      begin
         oSCAN<=8'd1;
         frame_cnt<=11'd0;
      end
   else
      begin
         if(oSCAN==8'h80)
            begin
               oSCAN<=8'd1;
               if(frame_cnt==FRAME_CNT)
                  frame_cnt<=11'd0;
               else
                  frame_cnt<=frame_cnt+11'd1;
            end
         else
            oSCAN<=oSCAN << 1'd1;
      end
endmodule
```

SW_PIC_CTRL and ROW DATA modules

```
module SW_PIC_CTRL(
                  input iCLK,iCLEAN_N,iSCAN_MSB,
                  input [10:0] iFRAME_CNT,
                  output reg [3:0] oSW_FRAME
                  );


parameter FRAME_CNT = 1;
parameter PIC_CNT = 1;

always@(posedge iCLK)
   if(!iCLEAN_N)
      begin
         oSW_FRAME<=4'd0;
      end
   else
      begin
         if((oSW_FRAME==PIC_CNT)&&(iFRAME_CNT==FRAME_CNT))
            oSW_FRAME<=4'd0;
         else if((iFRAME_CNT==FRAME_CNT) &&(iSCAN_MSB))
            oSW_FRAME<=oSW_FRAME+4'd1;

      end

endmodule
```

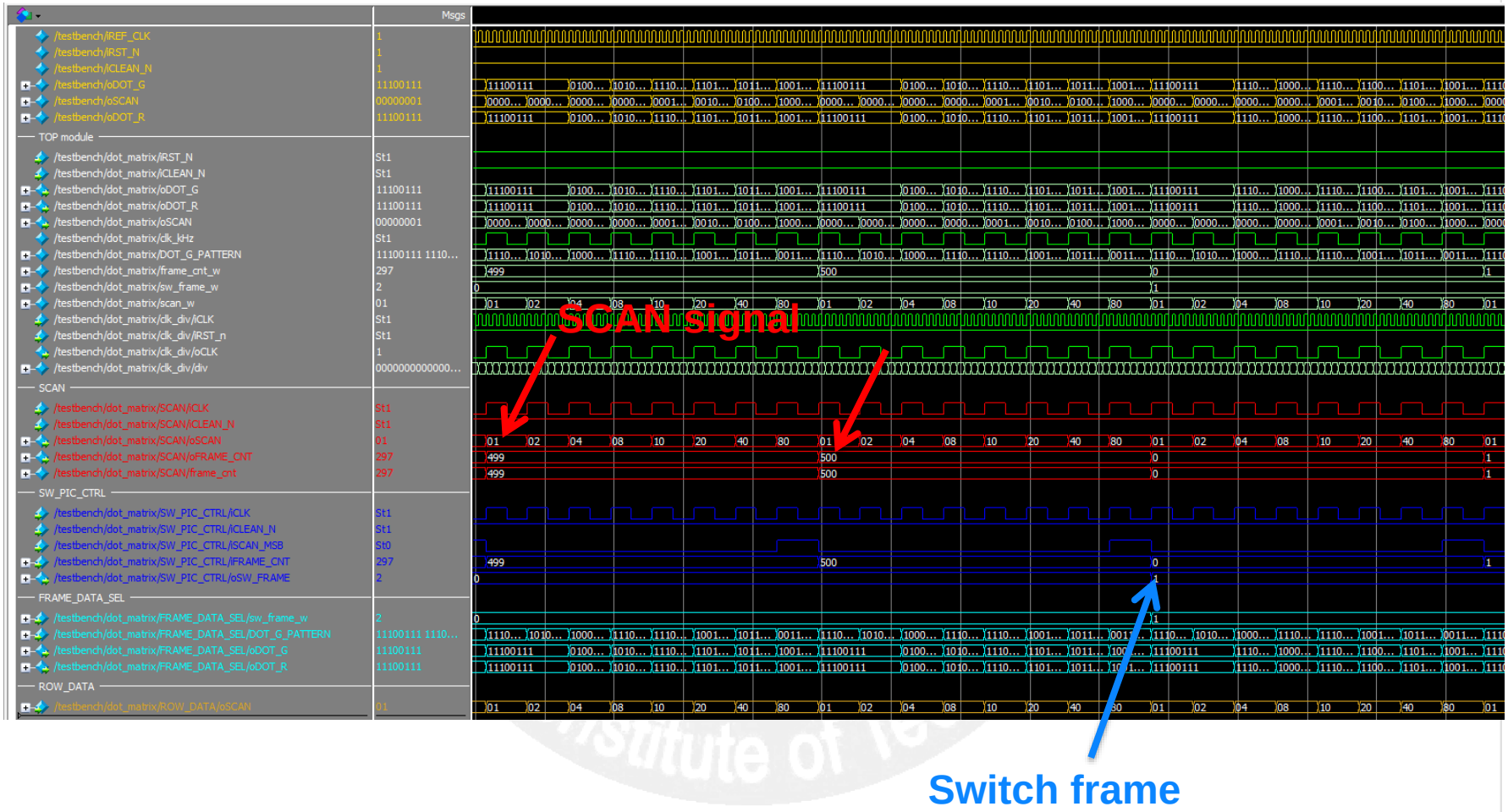
```
module ROW_DATA(
                  input  [7:0] oSCAN,
                  output reg [7:0] DOT_G_PA
                  );
always@(*)
   case(oSCAN)
   8'b0000_0001:DOT_G_PATTERN[0]=8'hE7;
   8'b0000_0010:DOT_G_PATTERN[0]=8'hE7;
   8'b0000_0100:DOT_G_PATTERN[0]=8'h41;
   8'b0000_1000:DOT_G_PATTERN[0]=8'hA6;
   8'b0001_0000:DOT_G_PATTERN[0]=8'hE7;
   8'b0010_0000:DOT_G_PATTERN[0]=8'hD7;
   8'b0100_0000:DOT_G_PATTERN[0]=8'hBD;
   8'b1000_0000:DOT_G_PATTERN[0]=8'h96;
   default:DOT_G_PATTERN[0]=8'hFF;
   endcase
always@(*)
   case(oSCAN)
   8'b0000_0001:DOT_G_PATTERN[1]=8'hE7;
   8'b0000_0010:DOT_G_PATTERN[1]=8'hE7;
   8'b0000_0100:DOT_G_PATTERN[1]=8'hE3;
   8'b0000_1000:DOT_G_PATTERN[1]=8'h85;
   8'b0001_0000:DOT_G_PATTERN[1]=8'hE7;
   8'b0010_0000:DOT_G_PATTERN[1]=8'hC7;
   8'b0100_0000:DOT_G_PATTERN[1]=8'hD7;
   8'b1000_0000:DOT_G_PATTERN[1]=8'h9B;
   default:DOT_G_PATTERN[1]=8'hFF;
   endcase
always@(*)
   case(oSCAN)
   8'b0000_0001:DOT_G_PATTERN[2]=8'hE7;
   8'b0000_0010:DOT_G_PATTERN[2]=8'hA7;
   8'b0000_0100:DOT_G_PATTERN[2]=8'h81;
   8'b0000_1000:DOT_G_PATTERN[2]=8'hE5;
   8'b0001_0000:DOT_G_PATTERN[2]=8'hE7;
   8'b0010_0000:DOT_G_PATTERN[2]=8'h9B;
   8'b0100_0000:DOT_G_PATTERN[2]=8'hBC;
   8'b1000_0000:DOT_G_PATTERN[2]=8'h3E;
   default:DOT_G_PATTERN[2]=8'hFF;
   endcase
endmodule
```

FRAME_DATA_SEL module and test-bench

```
module FRAME_DATA_SEL(
                      input [3:0] sw_frame_w,
                      input [7:0] DOT_G_PATTERN[2:0],
                      output reg [7:0] oDOT_G,oDOT_R
                      );


always@(*)
   case(sw_frame_w)
   4'd0:begin
            oDOT_G=DOT_G_PATTERN[0];
            oDOT_R=DOT_G_PATTERN[0];
        end
   4'd1:begin
            oDOT_G=DOT_G_PATTERN[1];
            oDOT_R=DOT_G_PATTERN[1];
        end
   4'd2:begin
            oDOT_G=DOT_G_PATTERN[2];
            oDOT_R=DOT_G_PATTERN[2];
        end
   default:
        begin
            oDOT_G=8'hff;
            oDOT_R=8'hff;
        end
   endcase
endmodule
```

```
`timescale 1ns/100ps
module testbench;

reg iREF_CLK,iRST_N,iCLEAN_N;
wire [7:0] oDOT_G,oDOT_R,oSCAN;


dot_matrix #(.DIV_CNT(2))
dot_matrix(
            .iREF_CLK(iREF_CLK),
            .iRST_N(iRST_N),
            .iCLEAN_N(iCLEAN_N),
            .oDOT_G(oDOT_G),
            .oDOT_R(oDOT_R),
            .oSCAN(oSCAN)
          );

initial
   begin
      iREF_CLK=1'd0;
      iRST_N=1'd0;
      iCLEAN_N=1'd0;
   end

initial
forever
   # 10 iREF_CLK = ~iREF_CLK;

initial
   begin
   # 100 iRST_N=1'd1;
   # 1000 iCLEAN_N=1'd1;
   end
endmodule
```

waveform


Msgs
/testbench/iREF_CLK
/testbench/iRST_N
/testbench/iCLEAN_N
/testbench/oDOT_G
/testbench/oSCAN
/testbench/oDOT_R
TOP module
/testbench/dot_matrix/iRST_N
/testbench/dot_matrix/iCLEAN_N
/testbench/dot_matrix/oDOT_G
/testbench/dot_matrix/oDOT_R
/testbench/dot_matrix/oSCAN
/testbench/dot_matrix/clk_kHz
/testbench/dot_matrix/DOT_G_PATTERN
/testbench/dot_matrix/frame_cnt_w
/testbench/dot_matrix/sw_frame_w
/testbench/dot_matrix/scan_w
/testbench/dot_matrix/clk_div/iCLK
/testbench/dot_matrix/clk_div/iRST_n
/testbench/dot_matrix/clk_div/oCLK
/testbench/dot_matrix/clk_div/div
SCAN
/testbench/dot_matrix/SCAN/iCLK
/testbench/dot_matrix/SCAN/iCLEAN_N
/testbench/dot_matrix/SCAN/oSCAN
/testbench/dot_matrix/SCAN/oFRAME_CNT
/testbench/dot_matrix/SCAN/frame_cnt
SW_PIC_CTRL
/testbench/dot_matrix/SW_PIC_CTRL/iCLK
/testbench/dot_matrix/SW_PIC_CTRL/iCLEAN_N
/testbench/dot_matrix/SW_PIC_CTRL/iSCAN_MSB
/testbench/dot_matrix/SW_PIC_CTRL/iFRAME_CNT
/testbench/dot_matrix/SW_PIC_CTRL/oSW_FRAME
FRAME_DATA_SEL
/testbench/dot_matrix/FRAME_DATA_SEL/sw_frame_w
/testbench/dot_matrix/FRAME_DATA_SEL/DOT_G_PATTERN
/testbench/dot_matrix/FRAME_DATA_SEL/oDOT_G
/testbench/dot_matrix/FRAME_DATA_SEL/oDOT_R
ROW_DATA
/testbench/dot_matrix/ROW_DATA/oSCAN
SCAN signal
Switch frame

Homework

- 矩陣小綠人和小紅人須與七段顯示器結合，條件如下:
  - 1.七段顯示器初始值為20，小綠人走路為一般速度
  - 2.七段顯示器數到10時，小綠人加速走路
  - 3.七段顯示器值為0時，顯示小紅人並持續三秒
  - 4.反覆1~3步驟

CSEP Development Kit

❖ Pin assignments

CLK_50M	PIN_N2	指撥開關_SW1(最左邊)	PIN_K6	按鈕開關_PB1	PIN_K22	點矩陣_Red[7]	PIN_E22
七段_a	PIN_D2	指撥開關_SW1	PIN_K5	按鈕開關_PB2	PIN_K23	點矩陣_Red[6]	PIN_D26
七段_b	PIN_E1	指撥開關_SW1	PIN_K4	按鈕開關_PB3	PIN_K24	點矩陣_Red[5]	PIN_D25
七段_c	PIN_E2	指撥開關_SW1	PIN_K3	按鈕開關_PB4	PIN_K25	點矩陣_Red[4]	PIN_D23
七段_d	PIN_F1	指撥開關_SW1	PIN_K2			點矩陣_Red[3]	PIN_C25
七段_e	PIN_F2	指撥開關_SW1	PIN_K1			點矩陣_Red[2]	PIN_C24
七段_f	PIN_F3	指撥開關_SW1	PIN_J6			點矩陣_Red[1]	PIN_B25
七段_g	PIN_F4	指撥開關_SW1(最右邊)	PIN_J5			點矩陣_Red[0]	PIN_B24
七段_dot	PIN_G1	指撥開關_SW2(最左邊)	PIN_M5			點矩陣_Green[7]	PIN_F26
七段掃描_1(最左邊)	PIN_G2	指撥開關_SW2	PIN_M4			點矩陣_Green[6]	PIN_F25
七段掃描_2	PIN_G3	指撥開關_SW2	PIN_M3			點矩陣_Green[5]	PIN_F24
七段掃描_3	PIN_G4	指撥開關_SW2	PIN_M2			點矩陣_Green[4]	PIN_F23
七段掃描_4	PIN_G5	指撥開關_SW2	PIN_L6			點矩陣_Green[3]	PIN_E26
七段掃描_5	PIN_G6	指撥開關_SW2	PIN_L4			點矩陣_Green[2]	PIN_E25
七段掃描_6(最右邊)	PIN_H1	指撥開關_SW2	PIN_L3			點矩陣_Green[1]	PIN_E24
		指撥開關_SW2(最右邊)	PIN_L2			點矩陣_Green[0]	PIN_E23
LED1	PIN_H2					點矩陣_掃描[0]	PIN_G21
LED2	PIN_H3	語音IC_RDY	PIN_H25			點矩陣_掃描[1]	PIN_G22
LED3	PIN_H4	語音IC_PWM	PIN_J20			點矩陣_掃描[2]	PIN_G23
LED4	PIN_H6	語音IC_SDI	PIN_H24			點矩陣_掃描[3]	PIN_G24
LED5	PIN_J1	語音IC_SCLK	PIN_H26			點矩陣_掃描[4]	PIN_G25
LED6	PIN_J2	語音IC_RES	PIN_J21			點矩陣_掃描[5]	PIN_G26
LED7	PIN_J3					點矩陣_掃描[6]	PIN_H21
LED8	PIN_J4					點矩陣_掃描[7]	PIN_H23