FPGA Lab-01

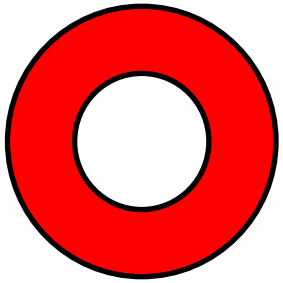
Running Led Light Design

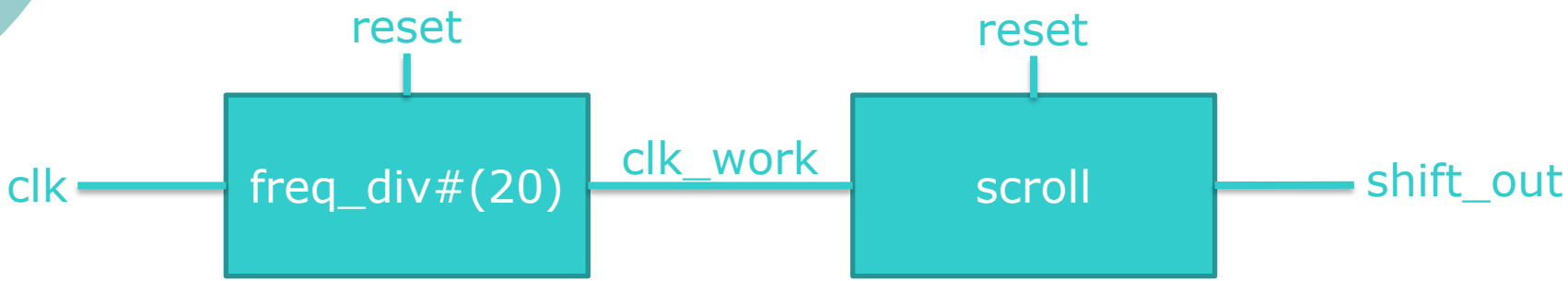
Coding issues !

- Pin Connection
- Missing Register
- Missing or Wrong Pin connection
- Missing semicolon or symbol from code

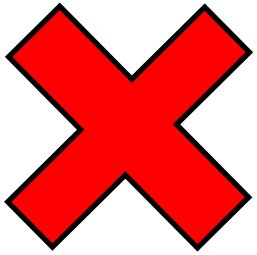
Pin Connection

```
freq_div#(20) M1 (clk, reset, clk_work);
scroll M2 (clk_work, reset, shift_out);
```



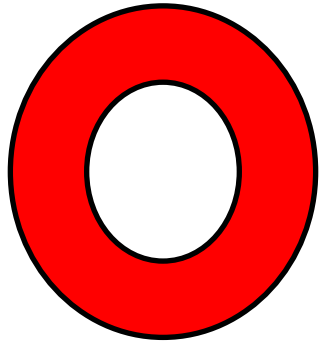



```
freq_div#(20) M1 (clk, reset, clk_work);
scroll M2 (clk, reset, shift_out);
```

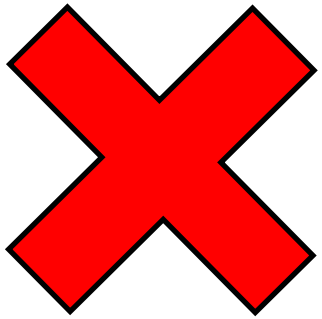


Variable is not declared properly

```
assign ctl_bit= 1'b1;
```



```
assign ctl_bit= 1;
```



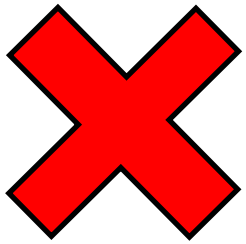
Correct Symbol notation

ppt講義上是.....

```
shift_out= 8'b1100_0000;
```

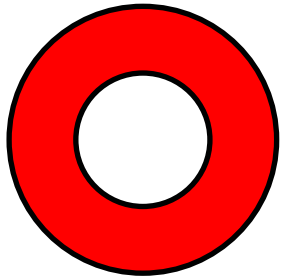
複製貼上後變成.....

```
shift_out= 8’b1100_0000;
```



應該要是.....

```
shift_out= 8'b1100_0000;
```



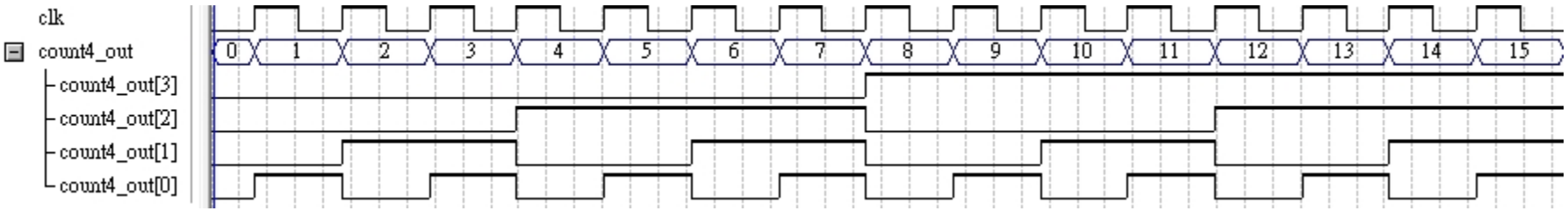
Other Issues

- always should come with reg
- Assign should use with wire
- begin must end
- Case must have endcase
- Case should have a default

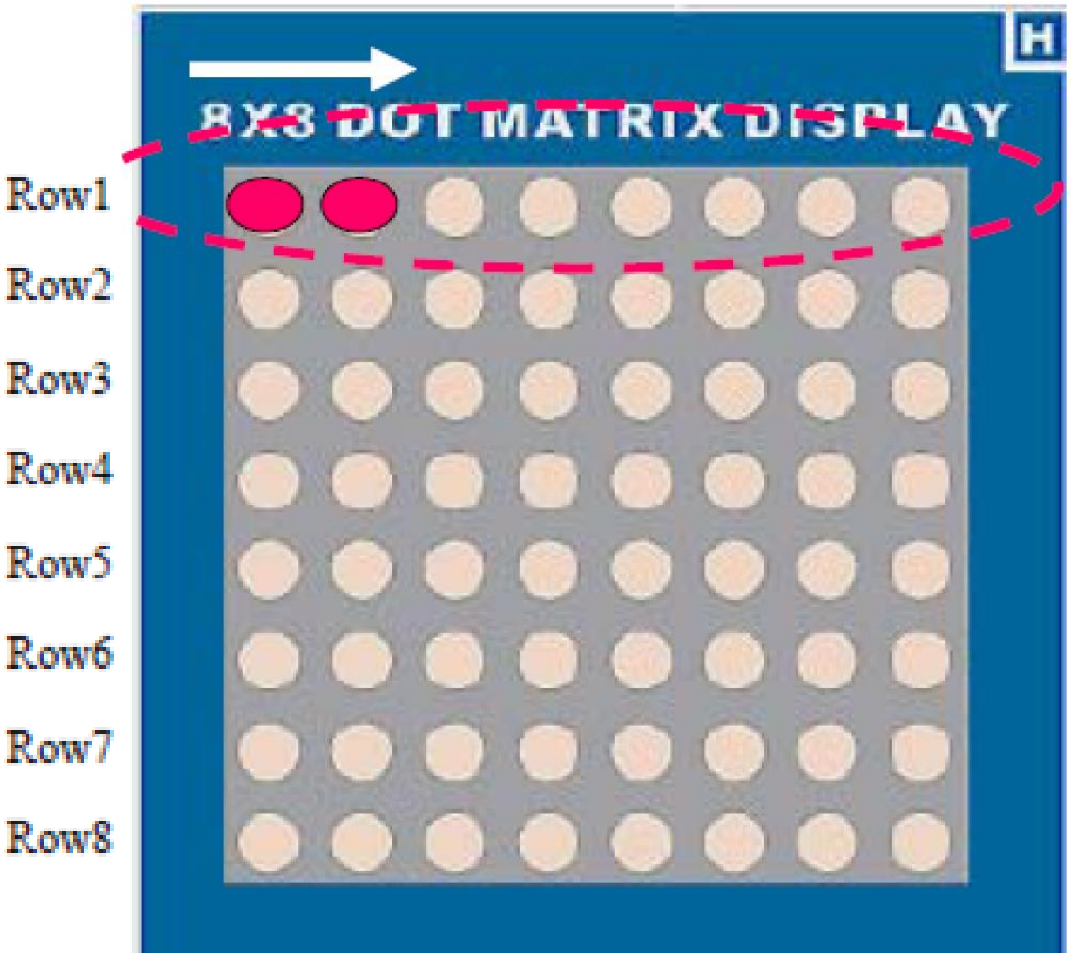
Frequency Division

```
module freq_div(clk_in, reset, clk_out);
parameter exp = 20;
input clk_in, reset;
output clk_out;
reg[exp-1:0] divider;
integer i;
assign clk_out= divider[exp-1];
always@ (posedge clk_in or posedge reset)      //正緣觸發
begin
if(reset)
for(i=0; i < exp; i=i+1)
divider[i] = 1'b0;
else
divider = divider+ 1'b1;
end
endmodule
```

3	2	1	0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

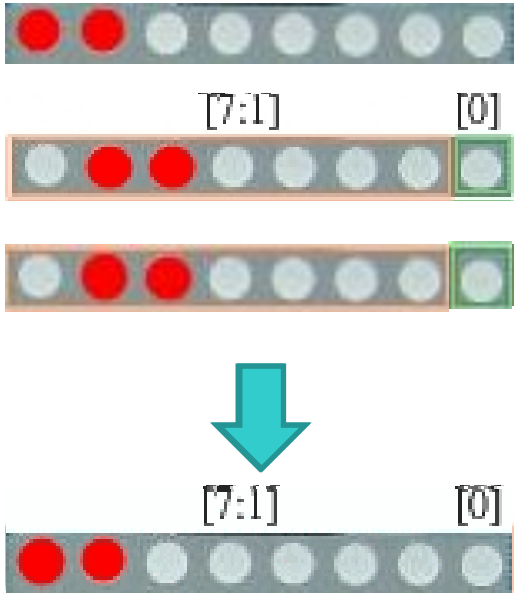


Lab 1 : **2 bits move from left to right**



Scroll Module

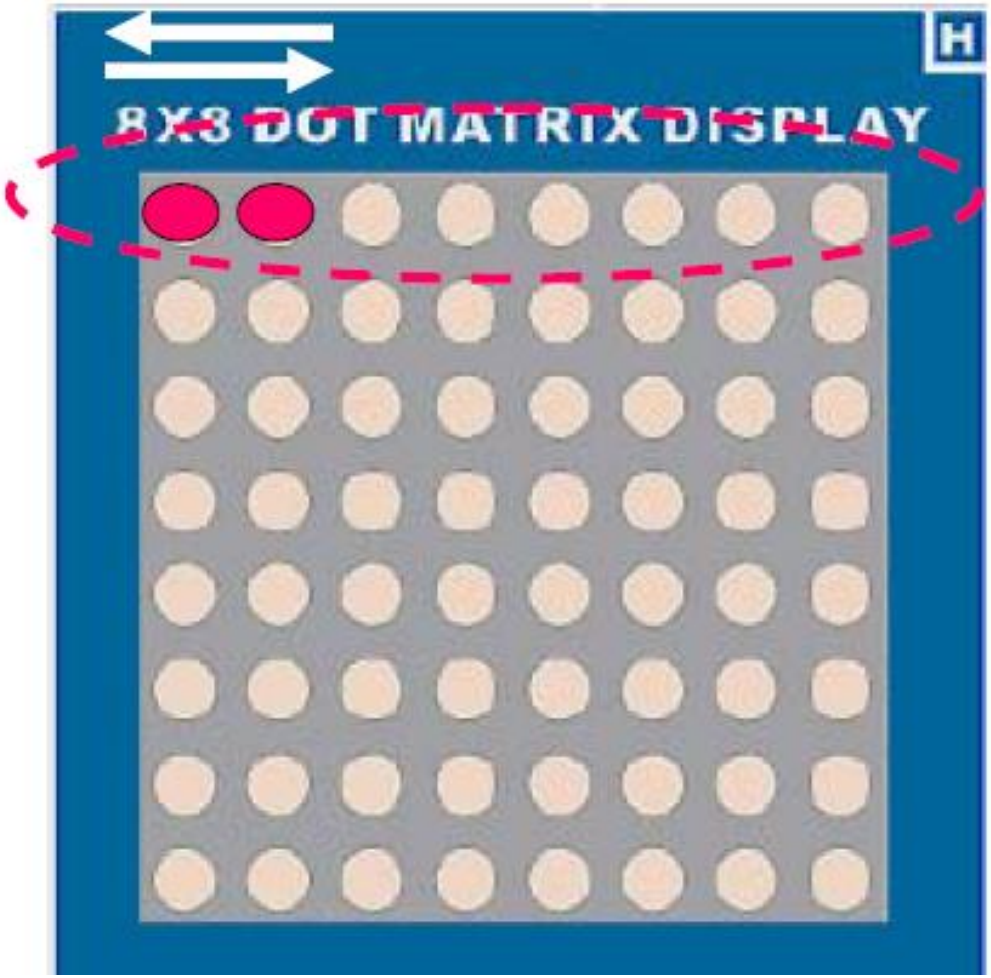
```
module scroll (clk, reset, shift_out);
input          clk, reset;
output         [7:0]shift_out;
reg            [7:0]shift_out;
always@ (posedge clk or posedge reset)
begin
if(reset)
shift_out= 8'b1100_0000;
else
shift_out= {shift_out[0], shift_out[7:1]};
end
endmodule
```



Main Module

```
module lab01_01 (clk, reset, shiftR_out, shiftG_out, ctl_bit);
input       clk;            // pin W16(10MHz)
input       reset;          // pin C16
output      [7:0]shiftR_out;
// pin D7, D6, A9, C9, A8, C8, C11, B11
output      [7:0]shiftG_out;
// pin A10 ,B10 ,A13 ,A12 ,B12 ,D12 ,A15 ,A14
assign shiftG_out =0;
output      ctl_bit;        // pin T22
assign      ctl_bit= 1'b1;
wire        clk_work;
freq_div    #(20) M1 (clk, reset, clk_work);
scroll M2 (clk_work, reset, shiftR_out);
endmodule
```

Lab 2：**2 bit move from left to right and right to left simultaneously**



Scroll Module

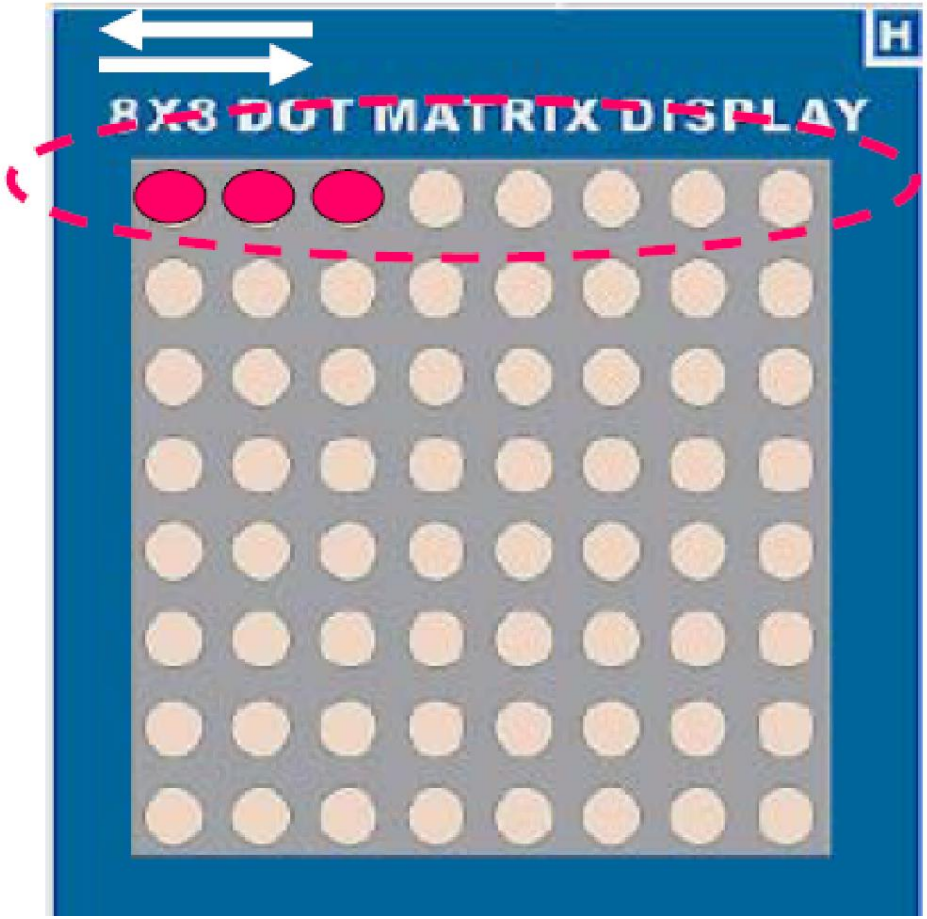
```
module scroll (clk, reset, shift_out);
input          clk, reset;
output         [7:0]shift_out;
wire           [7:0]shift_out;
reg            [8:0]pattern;
assign         shift_out= pattern[7:0];
always@ (posedge clk or posedge reset)
begin
if(reset)
pattern = 9'b0_1100_0000;
else
```

- case(pattern)
- 9'b0_11000000:pattern = 9'b0_01100000;
- 9'b0_01100000:pattern = 9'b0_00110000;
- 9'b0_00110000:pattern = 9'b0_00011000;
- 9'b0_00011000:pattern = 9'b0_00001100;
- 9'b0_00001100:pattern = 9'b0_00000110;
- 9'b0_00000110:pattern = 9'b0_00000011;
- 9'b0_00000011:pattern = 9'b1_00000110;
- 9'b1_00000110:pattern = 9'b1_00001100;
- 9'b1_00001100:pattern = 9'b1_00011000;
- 9'b1_00011000:pattern = 9'b1_00110000;
- 9'b1_00110000:pattern = 9'b1_01100000;
- 9'b1_01100000:pattern = 9'b1_11000000;
- 9'b1_11000000:pattern = 9'b0_11000000;
- default:pattern = 9'b0_11000000;endcase
- end
- endmodule

Main Module

```
module lab01_2_top (clk, reset, shiftR_out,
shiftG_out, ctl_bit);
input       clk; // pin W16
input       reset; // pin C16
output      [7:0]shiftR_out;
// pin D7, D6, A9, C9, A8, C8, C11, B11
output      [7:0]shiftG_out;
// pin A10 ,B10 ,A13 ,A12 ,B12 ,D12 ,A15 ,A14
assign shiftG_out =0;
output      ctl_bit; // pin T22
assign      ctl_bit= 1'b1;
freq_div#(20)M1 (clk, reset, clk_work);
scroll M2 (clk_work, reset, shiftR_out);
endmodule
```

Lab 3：**3 bit move from left to right and right to left simultaneously**



Scroll Module

```
module scroll (clk, reset, shift_out);
input          clk, reset;
output         [7:0]shift_out;
wire           [7:0]shift_out;
reg            [8:0]pattern;
assign         shift_out= pattern[7:0];
always@ (posedge clk or posedge reset)
begin
if(reset)
pattern = 9'b0_1110_0000;
else
```

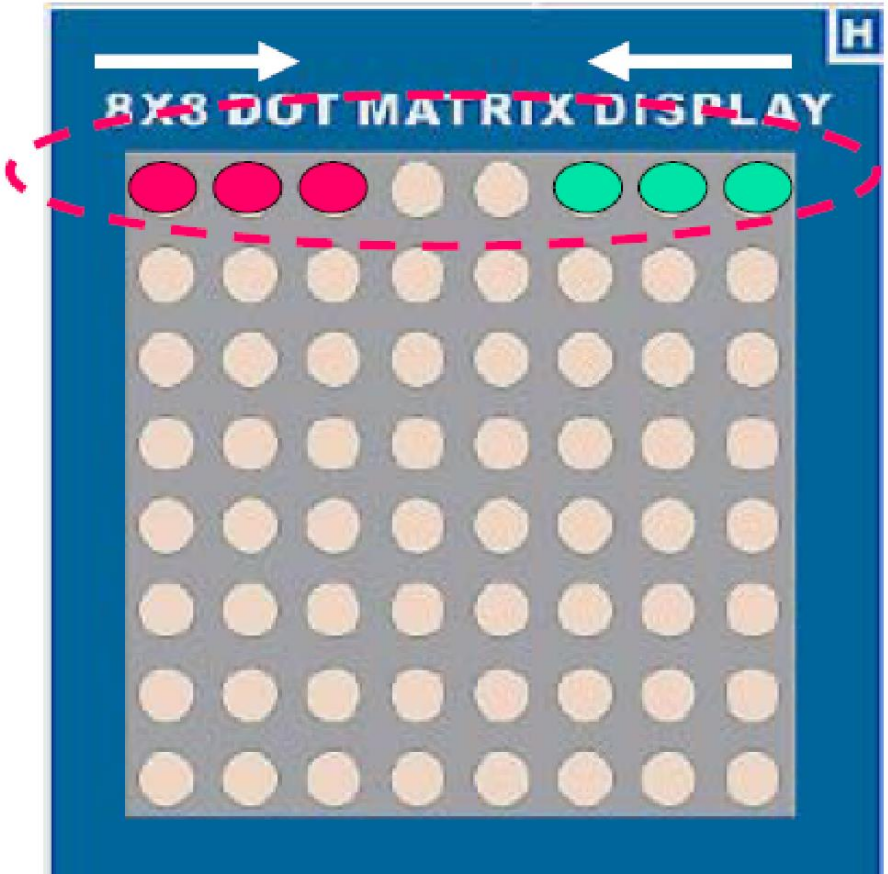
```
case(pattern)
 9'b0_11100000:pattern = 9'b0_01110000
 9'b0_01110000:pattern = 9'b0_00111000;
 9'b0_00111000:pattern = 9'b0_00011100;
 9'b0_00011100:pattern = 9'b0_00001110;
 9'b0_00001110:pattern = 9'b0_00000111;
 9'b0_00000111:pattern = 9'b1_00001110;
 9'b1_00001110:pattern = 9'b1_00011100;
 9'b1_00011100:pattern = 9'b1_00111000;
 9'b1_00111000:pattern = 9'b1_01110000;
 9'b1_01110000:pattern = 9'b1_11100000;
 9'b1_11100000:pattern = 9'b0_01110000;
default:pattern = 9'b0_11100000;endcase
end
endmodule
```

Main Module

```
module lab01_03 (clk, reset, shiftR_out,
shiftG_out, ctl_bit);
input       clk; // pin W16
input       reset; // pin C16
output      [7:0]shiftR_out;
// pin D7, D6, A9, C9, A8, C8, C11, B11
output      [7:0]shiftG_out;
// pin A10 ,B10 ,A13 ,A12 ,B12 ,D12 ,A15 ,A14
assign shiftG_out =0;
output      ctl_bit; // pin T22
assign      ctl_bit= 1'b1;
freq_div#(20) M1 (clk, reset, clk_work);
scroll M2 (clk_work, reset, shiftR_out);
endmodule
```

Lab 2：3 bit move from left to right(red light) and right to left(green light) simultaneously

- The red light and the green light use different pins



Scroll Module

```
module scroll (clk, reset, shift_red,shift_green);
input          clk, reset;
output         [7:0]shift_red,shift_green;
wire           [7:0]shift_red,shift_green;
reg            [8:0]pattern;
assign shift_red=pattern[8]?{8'b0000000}:(pattern[7:0]);
assign shift_green=pattern[8]?(pattern[7:0]):{8'b00000000};
always@ (posedge clk or posedge reset)
begin
if(reset)
pattern = 9'b0_1110_0000;
else
```

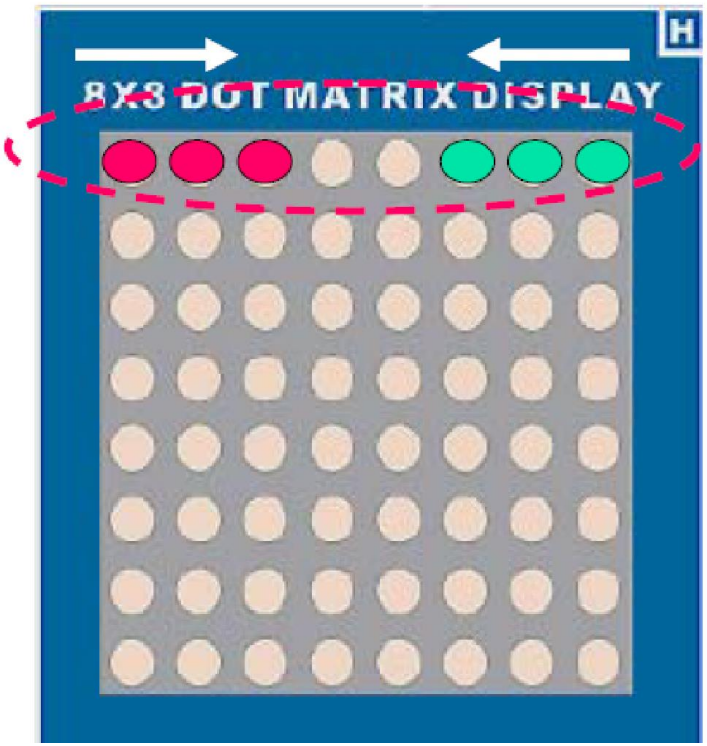
```
case(pattern)
  9'b0_11100000:pattern = 9'b0_01110000;
  9'b0_01110000:pattern = 9'b0_00111000;
  9'b0_00111000:pattern = 9'b0_00011100;
  9'b0_00011100:pattern = 9'b0_00001110;
  9'b0_00001110:pattern = 9'b0_00000111;
  9'b0_00000111:pattern = 9'b1_00001110;
  9'b1_00001110:pattern = 9'b1_00011100;
  9'b1_00011100:pattern = 9'b1_00111000;
  9'b1_00111000:pattern = 9'b1_01110000;
  9'b1_01110000:pattern = 9'b1_11100000;
  9'b1_11100000:pattern = 9'b0_01110000;
default:pattern = 9'b0_11100000;endcase
end
endmodule
```

Main Module

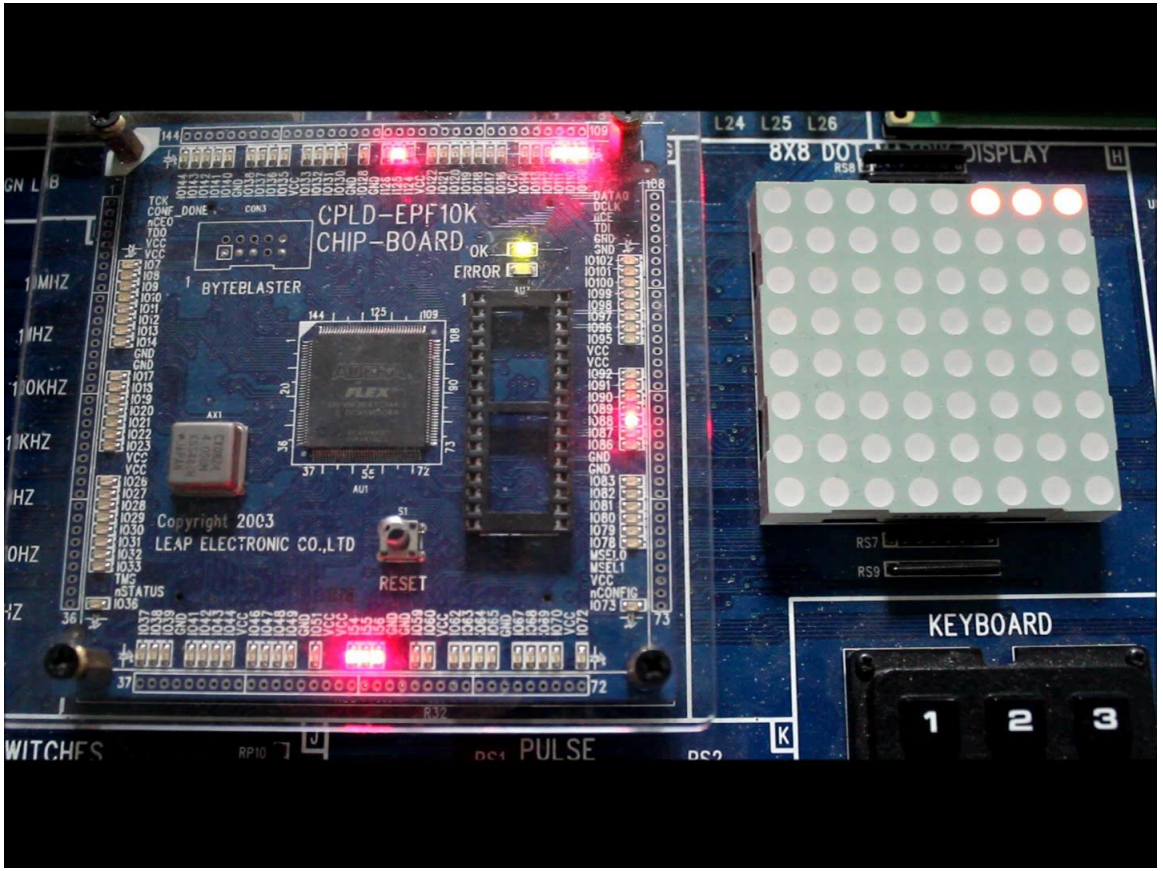
```
module lab01_04 (clk, reset, shift_red,shift_green,
ctl_bit);
input          clk;       // pin W16
input          reset;     // pin C16
output         [7:0]shift_red;
// pin D7, D6, A9, C9, A8, C8, C11, B11
output         [7:0]shift_green;
// pin A10, B10, A13, A12, B12, D12, A15, A14
output         ctl_bit;  // pin T22
assign         ctl_bit= 1'b1;
freq_div#(20)M1 (clk, reset, clk_work);
scroll M2 (clk_work, reset, shift_red,shift_green);
endmodule
```

Extra Question : 3 bit move from left to right(red light, fast) and right to left(green light, slow) simultaneously

- Use two frequency divider modules to generate two different speed clocks
- Use these two clocks according to different speeds



3 bit move from left to right(red light, fast) and right to left(green light, slow) simultaneously



Scroll Module

```
module scroll (clk, reset, shift_red,shift_green,direction);
input           clk, reset;
output          [7:0]shift_red,shift_green;
output          direction;
wire            [7:0]shift_red,shift_green;
reg             [8:0]pattern;
assign shift_red=pattern[8]?{8'b0000000}:(pattern[7:0]);
assign shift_green=pattern[8]?(pattern[7:0]):{8'b00000000};
assign direction = pattern[8];
```

Main Module

```
module lab01_04 (clk, reset, shift_red,shift_green, ctl_bit);
input           clk;       // pin W16
input           reset;     // pin C16
output          [7:0]shift_red;
// pin D7, D6, A9, C9, A8, C8, C11, B11
output          [7:0]shift_green;
// pin A10, B10, A13, A12, B12, D12, A15, A14
output          ctl_bit;  // pin T22
wire            clk_slow,clk_fast,clk_work,direction;
assign          ctl_bit= 1'b1;
assign          clk_work = (direction)?clk_slow : clk_fast;
freq_div#(20)M1 (clk, reset, clk_slow);
freq_div#(18)M2 (clk, reset, clk_fast);
scroll M3 (clk_work, reset, shift_red,shift_green,direction);
endmodule
```