42/47(針對比較大的設計如2 需針對每個block的功能做文字上的描述)
+5/10(助教自由給分)
=47/57

邏輯設計實驗 Lab04 結報

104060012 邱怡庭

prelab1 Cascade eight DFFs together as the figure shown below with the input bits D=(d), output Q=(q7q6q5q4q3q2q1q0), and tie all their Cs together as clk (the divided clock). This is a serial shift register circuit. By adding a multiplexer before the first DFF, we can choose the input from outside or from q0. As the clk changes 0,1,0,1,…, let the output of LEDs be (01010101), (10101010), (01010101), (10101010), ….

1.1 Construct the Verilog RTL representation for the logics with verification

1 Implement pre-lab1 with the following pin assignments.

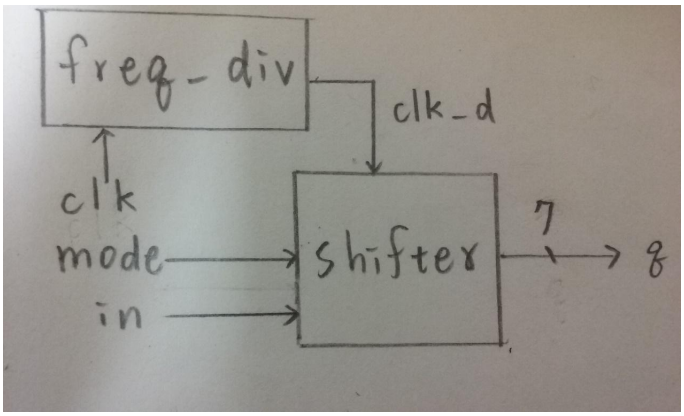
Design Specification

Input: clk, rst_n, mode, in;

Output: [7:0]q;

Wire: mode, in, [7:0]q, [1:0]clk_ctl, clk_d;

block diagram:



Design Implementation

logic function:

shifter

always @(posedge clk or negedge rst_n)

if (~rst_n)

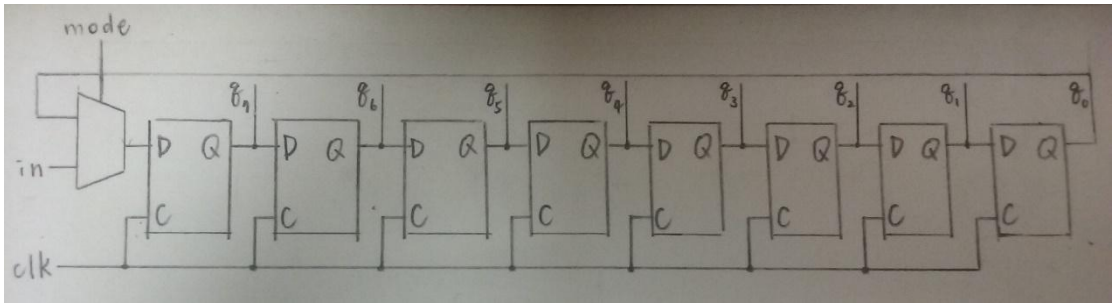
```
        begin
        q<=8'b01010101;
        end
    else if(~mode)
        begin
        q[0]<=q[1];
        q[1]<=q[2];
        q[2]<=q[3];
        q[3]<=q[4];
        q[4]<=q[5];
        q[5]<=q[6];
        q[6]<=q[7];
        q[7]<=q[0];
        end
    else
        begin
        q[0]<=q[1];
        q[1]<=q[2];
        q[2]<=q[3];
        q[3]<=q[4];
        q[4]<=q[5];
        q[5]<=q[6];
        q[6]<=q[7];
        q[7]<=in;
        end
```

有了logic diagram 後上面的code 就不需要了。
但要加一點文字敘述描述這個logic 電路的behavior

freq_div:

```
cnt_tmp = {clk_out,cnt_h,clk_ctl,cnt_l} + 1'b1;
```

logic diagram:



I/O pin assignment:

```
NET "q[7]" LOC = K6;
```

```
NET "q[6]" LOC = L7;
NET "q[5]" LOC = H3;
NET "q[4]" LOC = H4;
NET "q[3]" LOC = K5;
NET "q[2]" LOC = L5;
NET "q[1]" LOC = K3;
NET "q[0]" LOC = K4;
NET "clk" LOC = R10;
NET "rst_n" LOC = P2;
NET "mode" LOC = T1;
NET "in" LOC = P1;
```

Discussion:

用 always 裡的邏輯去架構出一個 8 個 1bit 的 flip-flops，以及透過 mode 的控制來選擇是否得到 in 的值。

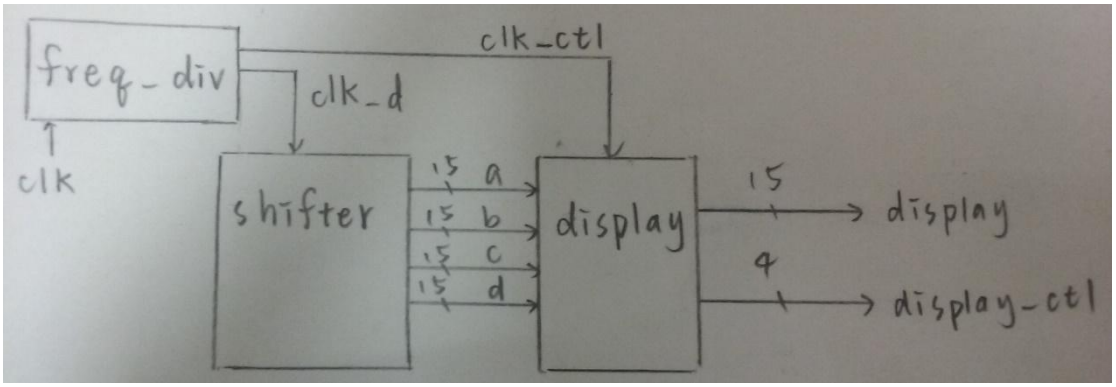
2 Use the idea from pre-lab1. We can do something on the seven-segment display. Assume we have the pattern of E, H, N, T, U for seven-segment display as shown below. Try to implement the scrolling pre-stored pattern NTHUEE with the four seven-segment displays.

input clk, rst_n

wire clk_d, [1:0]clk_ctl, [14:0]a, [14:0]b, [14:0]c, [14:0]d;

output [14:0] display, [3:0] display_ctl;

block diagram:



Design Implementation

logic function:

display:

always @(posedge clk or negedge rst_n)

if (~rst_n)

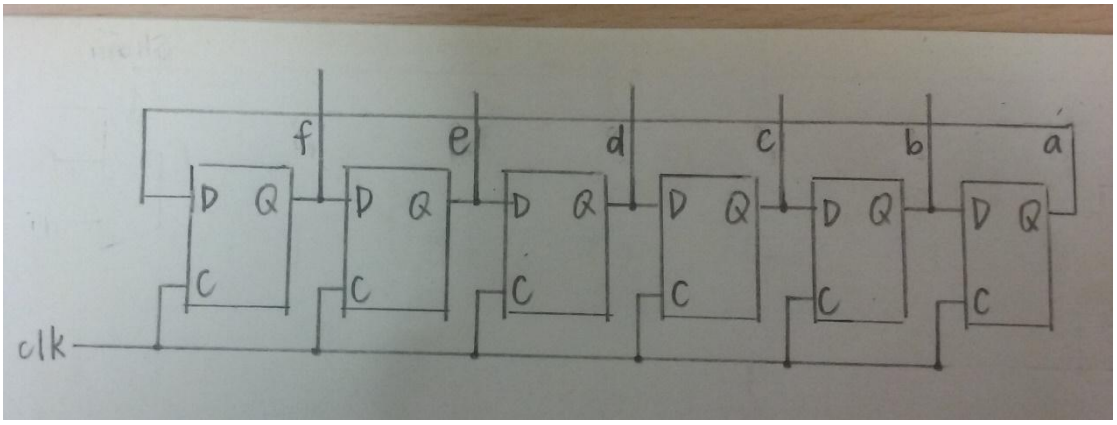
logic function 盡量不要將code 貼上來 可以畫簡單logic diagram 如decoder 、flip flop 等等(以這個層次的block為unit)，再底下做電路behavior的文字敘述。

```
        begin
        a<=15'b 10010011_011110_1; //N
        b<=15'b 01111111_101101_1; //T
        c<=15'b 10010000_111111_1; //H
        d<=15'b 10000011_111111_1; //U
        e<=15'b 01100000_111111_1; //E
        f<=15'b 01100000_111111_1; //E
        end
        else
        begin
        a[14:0]<=b[14:0];
        b[14:0]<=c[14:0];
        c[14:0]<=d[14:0];
        d[14:0]<=e[14:0];
        e[14:0]<=f[14:0];
        f[14:0]<=a[14:0];
        end
freq_div:
        cnt_tmp = {clk_out,cnt_h,clk_ctl,cnt_l} + 1'b1;
ssd_ctl:
        // display value selection
always @(clk_ctl or display0 or display1 or display2 or display3)
        case(clk_ctl)
        2'b00: display = display0;
        2'b01: display = display1;
        2'b10: display = display2;
        2'b11: display = display3;
        default : display = 15'b11111111_111111_1;
        endcase

// 7-segment control
always @(clk_ctl)
        case(clk_ctl)
        2'b00: display_ctl = 4'b1110;
        2'b01: display_ctl = 4'b1101;
        2'b10: display_ctl = 4'b1011;
        2'b11: display_ctl = 4'b0111;
        default : display_ctl = 4'b1111;
```

endcase

logic diagram:



I/O pin assignment:

```
NET "display_ctl[3]" LOC = "T6";
NET "display_ctl[2]" LOC = "V6";
NET "display_ctl[1]" LOC = "U8";
NET "display_ctl[0]" LOC = "V8";
NET "display[14]" LOC = "P6";
NET "display[13]" LOC = "N4";
NET "display[12]" LOC = "V5";
NET "display[11]" LOC = "T5";
NET "display[10]" LOC = "U7";
NET "display[9]" LOC = "R3";
NET "display[8]" LOC = "N5";
NET "display[7]" LOC = "R5";
NET "display[6]" LOC = "T3";
NET "display[5]" LOC = "T4";
NET "display[4]" LOC = "V4";
NET "display[3]" LOC = "V7";
NET "display[2]" LOC = "R7";
NET "display[1]" LOC = "T7";
NET "display[0]" LOC = "U5";
NET "clk" LOC = "R10";
NET "rst_n" LOC = "T1";
```

Discussion:

設置 6 個 15bits 的暫存器，然後使用邏輯帶出 a<=b, b<=c, c<=d, d<=e, e<=f,

f<=a，最後只要顯示出 a,b,c,d 即可，另外由於是顯示 4 個 14 段顯示器，因此需再使用 ssd control。

Conclusion:

實驗的步驟雖然越來越繁雜，但當完成時總是相當有成就感，像是這次成功看到 NTHUEE 在顯示器上移動時，覺得很開心，希望未來能學會更多控制的功能!