LAB03

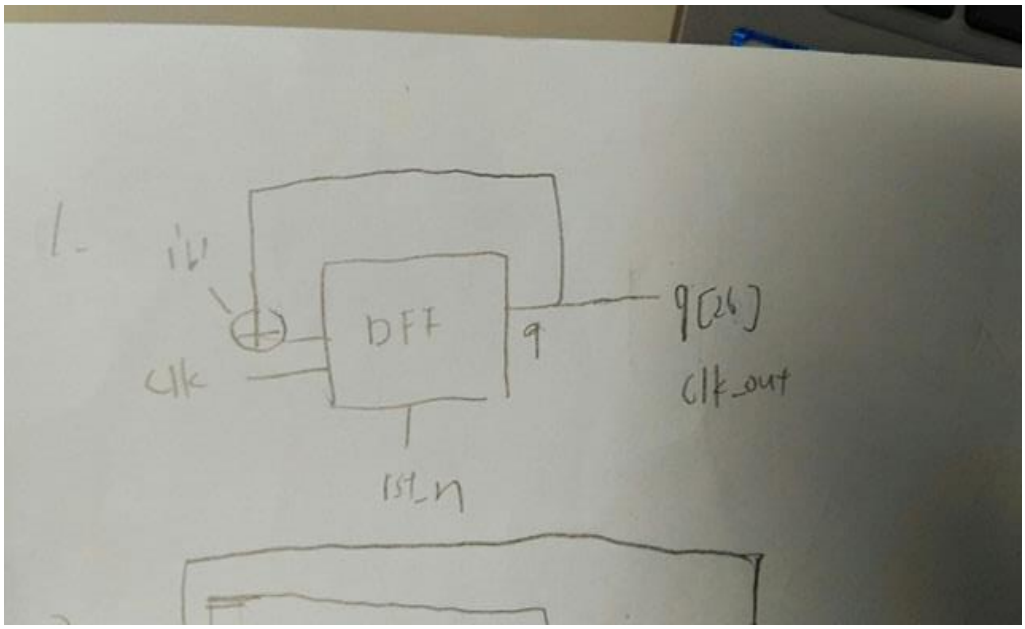
1.

Design Specification

```
reg [26:0]q;
input clk;          // global clock
input rst_n;        //  如果 res_n == 1，歸零
output clk_out;     // output  的結果
```

Design Implementation

Diagram:



Logic equation:

```
always @(posedge clk or negedge rst_n) begin
        if (~rst_n) q <= 0;
        else q <= q + 1;
    end
    assign clk_out = q[26];
```

I/O PINS

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN U16 [get_ports clk_out]
set_property PACKAGE_PIN V17 [get_ports rst_n]
```

因為引出的是 q 的第 27bit，所以要數完 2^26 次 output 才會變一次，這樣頻率就會變 1/2^27

Conclusion

這一題是利用 flip-flop 的小觀念，因為每個 flip-flop 的輸出頻率都會是輸入的 clk 的 1/2，然後下一級的輸入(clk)又是上一級的輸出，所以就會一直除(1/2)下去。利用這個小特性，很快就能寫出 verilog code 了。

2.

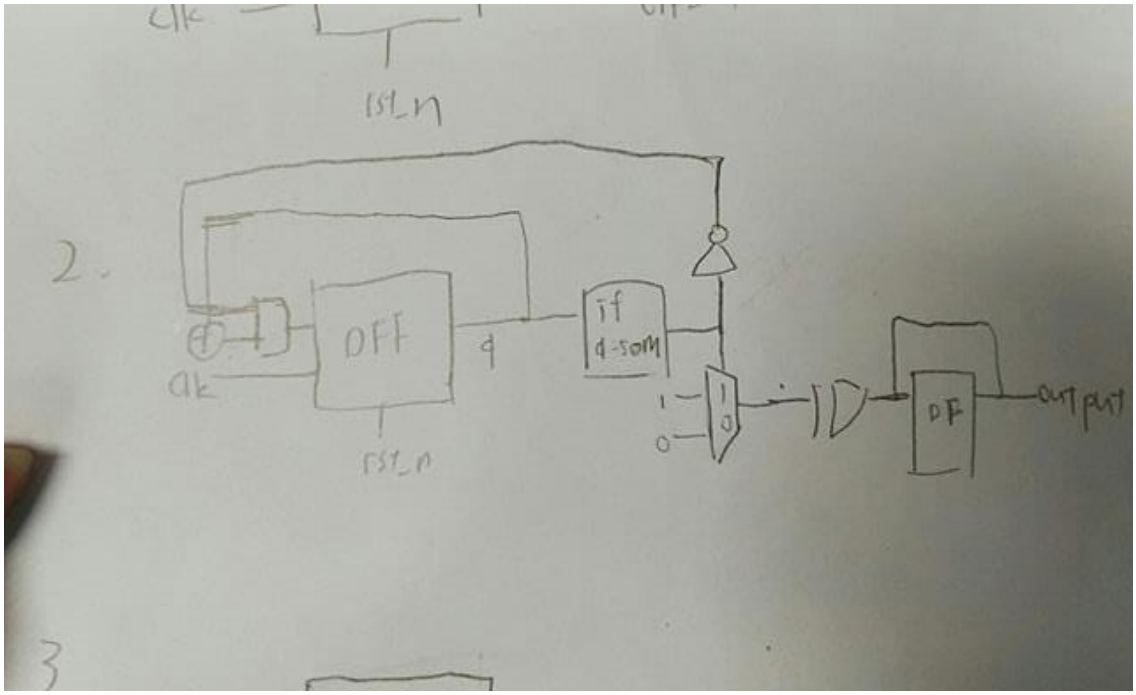
Design Specification

```
input clk;          // global clock
input rst_n;        // rst_n == 1, 重製
output reg q = 0;   // output

reg s;              // if (Q == 50M - 1) s = 1;
reg [26:0]Q;        // 計數
reg [26:0]Q_temp;
```

Design Implementation

Diagram:



Logic equation:

```
always @*
    Q_temp = Q + 1'b1;
always @(posedge clk or negedge rst_n)
    if (~rst_n || s == 1) Q <= 27'd0;
    else Q <= Q_temp;
```

```
always @(posedge clk or negedge rst_n)
    if (~rst_n)
        q <= 0;
    else
        q <= q ^ s;

always @*
    if (Q == 27'd50000000 - 1) begin
        s = 1;
    end
    else begin
        s = 0;
    end
```

I/O PINS

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN U16 [get_ports q]
set_property PACKAGE_PIN V17 [get_ports rst_n]
```

每當數到 50000000-1 時，clk 就會反向一次，輸出 0 或 1 的直的頻率會是原本 input clk 的 1/100000000 倍。

Disscussion:

這次遇到最大的問題就是我同時在 2 個 always block 裡 assign Q 的值，後來去查錯誤訊息的意思才發現，所以多設了一個變數 s 來控制 Q 的歸 0 及 output 值的改變。

Conclusion:

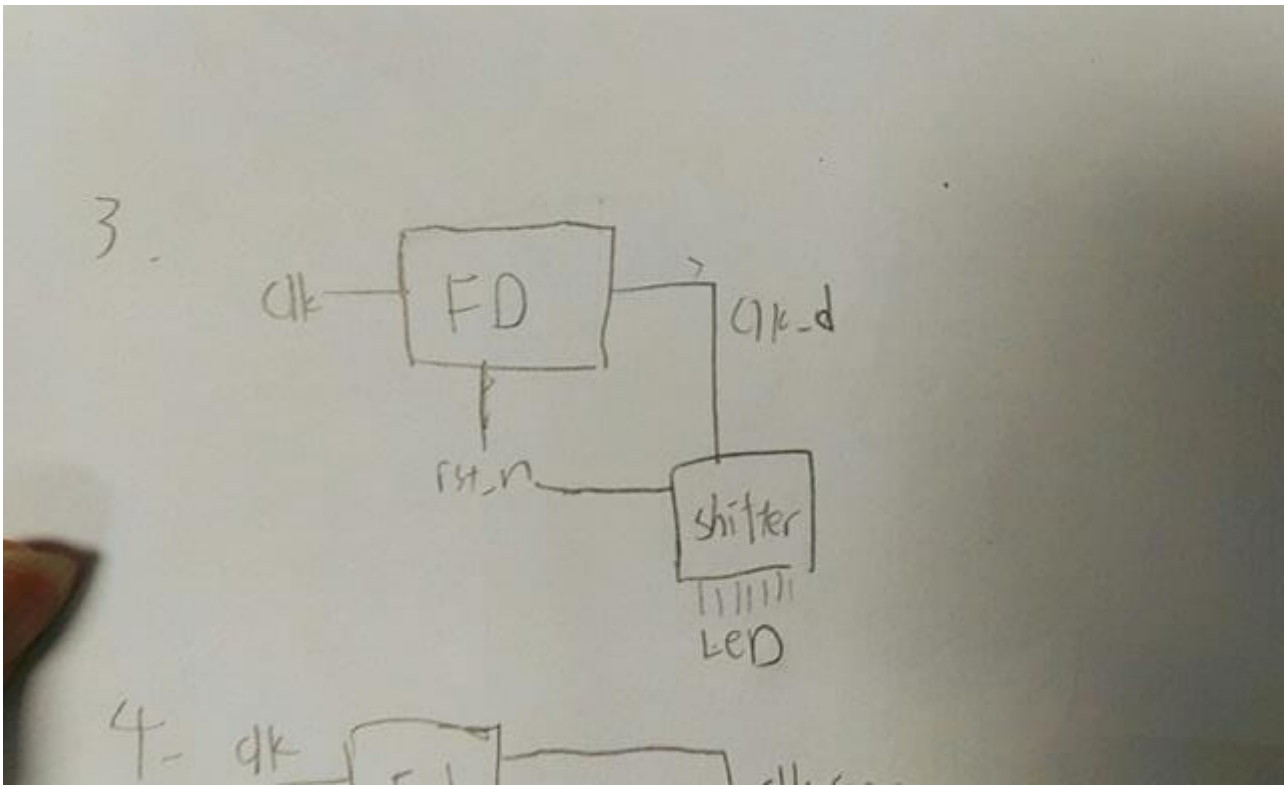
這一題的概念跟第一題很像，但是不同的地方在於，50000000 不事 2 的次 方數，所以不能藉由數到完整的次方數來做除頻的效果。因此採用每數 50000000 就讓輸出的 clk_out 反向一次，也就是每數 50000000，就會 0 跟 1 變化，如此 完整的一周期，需要數 10000000，頻率自然為原本了 1/100000000。跟第一題有點不一樣要注意一下，但大致上還相似。 但要注意的一點是，因為是從 0 開始數，所以要數到 50000000-1，而非 50000000

3.

Design Specification

```
output reg [7:0]q;          // this is the LED output
input clk;                  // global clock
input rst_n;                // reset
wire clk_d;                 // the clock generated by frequent divider
```

Design implementation




```
// Frequency shifter
FD U0(clk, rst_n, clk_d);       // FD is frequency divider at 3-2
always @(posedge clk_d or negedge rst_n)
    if (~rst_n)
        q <= 8'b01010101;       // the initial value
    else begin                  // shift
        q[0] <= q[7];
        q[1] <= q[0];
        q[2] <= q[1];
        q[3] <= q[2];
        q[4] <= q[3];
        q[5] <= q[4];
```

```
            q[6] <= q[5];
            q[7] <= q[6];
        end
```

I/O PINS

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN V17 [get_ports rst_n]
set_property PACKAGE_PIN U16 [get_ports {q[0]}]
set_property PACKAGE_PIN E19 [get_ports {q[1]}]
set_property PACKAGE_PIN U19 [get_ports {q[2]}]
set_property PACKAGE_PIN V19 [get_ports {q[3]}]
set_property PACKAGE_PIN W18 [get_ports {q[4]}]
set_property PACKAGE_PIN U15 [get_ports {q[5]}]
set_property PACKAGE_PIN U14 [get_ports {q[6]}]
set_property PACKAGE_PIN V14 [get_ports {q[7]}]
```

Discussion:

這次遇到的問題是出在 shifter 那裡，我本來用 C 寫法

```
Temp <= q[7];
for (i = 7; i > 0; i = i + 1;) q[i] <= q[i - 1];
q[0] = temp;
```

但結果出問題了，就只能一個一個打

Conclusion

這次的就只是把 prelab 跟 3-2 接在一起而已，結果就是 4 個 led 交錯著閃

4.

Design specification

Main module

```
input clk;                  // global clock
input rst_n;                // reset
output [7:0]ssd;            // seven segment display
output [3:0]light;          // lights of seven segment display
wire [2:0]word;             // words
wire [1:0]clk_scan;         // clock of scan
wire clk_d;                 // clock for shifter
wire [2:0]in0;              // four words
wire [2:0]in1;
wire [2:0]in2;
wire [2:0]in3;
```

fd module

```
reg [25:0]q;
input clk;
input rst_n;
output reg clk_out;
output reg [1:0]clk_out2;
reg [26:0]q_temp;
```

rshifter module

```
input clk;
input rst_n;
output reg [2:0]q0;
output reg [2:0]q1;
output reg [2:0]q2;
output reg [2:0]q3;
reg [2:0]q4;
reg [2:0]q5;
```

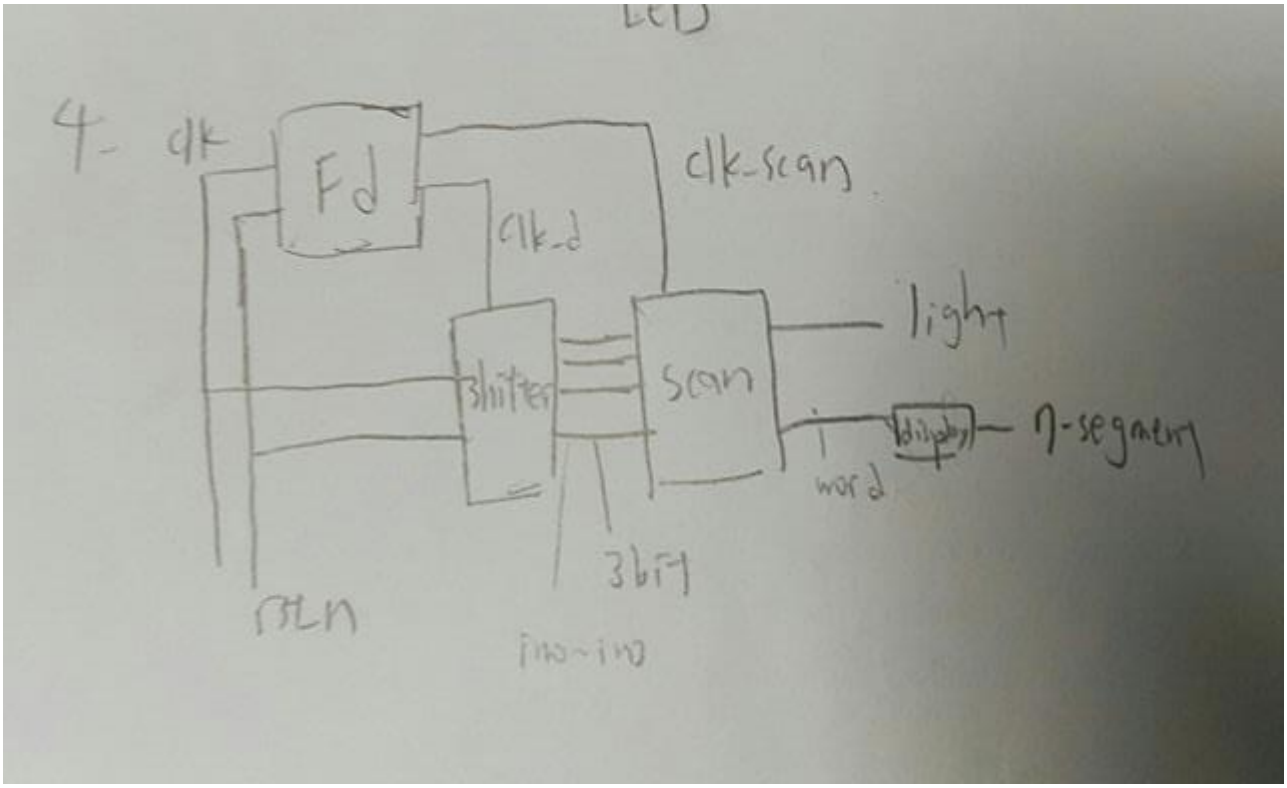
scan module

```
input [2:0]in0;
input [2:0]in1;
input [2:0]in2;
input [2:0]in3;
```

input [1:0]clk_scan;
output reg [3:0]light;
output reg [2:0]word;

display module
input [2:0]in;
output reg [7:0]D_ssd;

design implementation



FD is freauency divider
Shifter can shift nthuee
Scan is scan control for seven segment display

Main module

```
f U0(.clk(clk),
    .rst_n(rst_n),
    .clk_out(clk_d),
    .clk_out2(clk_scan)
    );
rshifter U1(.clk(clk_d),
            .rst_n(rst_n),
            .q0(in0),
            .q1(in1),
            .q2(in2),
            .q3(in3)
            );
scan U2(.in0(in0),
        .in1(in1),
        .in2(in2),
        .in3(in3),
        .clk_scan(clk_scan),
        .light(light),
        .word(word)
        );
display U3(.in(word), .D_ssd(ssd));
```

Fd module

```
reg [25:0]q;
input clk;
input rst_n;
output reg clk_out;
output reg [1:0]clk_out2;
reg [26:0]q_temp;

always @*
    q_temp = {clk_out, q} + 1'b1;
always @(posedge clk or negedge rst_n) begin
    if (~rst_n) q <= 0;
    else {clk_out, q} <= q_temp;
end
always @(posedge clk or negedge rst_n) begin
    if (~rst_n) clk_out2 <= 2'b00;
    else clk_out2 <= {q[15], q[14]};
end
```

Rshifter module

```
always@(posedge clk or negedge rst_n)
    if (~rst_n) begin
        q0 <= 3'd0;
        q1 <= 3'd1;
        q2 <= 3'd2;
        q3 <= 3'd3;
        q4 <= 3'd4;
        q5 <= 3'd4;
    end
    else begin
        q0 <= q1;
        q1 <= q2;
        q2 <= q3;
        q3 <= q4;
        q4 <= q5;
        q5 <= q0;
    end
```

Scan module

```
case (clk_scan)
2'b00: begin
    light = 4'b0111;
    word = in0;
end
2'b01: begin
    light = 4'b1011;
    word = in1;
end
2'b10: begin
    light = 4'b1101;
    word = in2;
end
2'b11: begin
    light = 4'b1110;
    word = in3;
end
default: begin
    light = 1111;
    word = 3'b111;
end
endcase
```

Display module

```
always@*
    case(in)
        3'd0:D_ssd = 8'b11010101;   //N
        3'd1:D_ssd = 8'b11100001;   //T
        3'd2:D_ssd = 8'b10010001;   //H
        3'd3:D_ssd = 8'b10000011;   //U
        3'd4:D_ssd = 8'b01100001;   //E
        default:D_ssd=8'b00000000;
    endcase
```

Discussion:

這次的真的很難，我是去偷看老是第四張的講義才打出來的，中間 de 了很多 bug 像是忘記用陣列宣告，還有 case 忘記 default，但我花醉酒找的錯誤是[3:0]in0, in1, in2, in3，不知道為何我的 verilog 不接紹這種寫法，導致我只有第一個燈會跑

Conclusion:

由於這次真的比較複雜，我決定按部就班，真的照著我畫的圖上打，我設了一個 main module，main module 裡甚麼也不做，只負責連接各 module，我發現這樣邏輯比較清楚，也方便 debug，以後我都會採用此種寫法