LAB3_pre_1

Consider a 4-bit synchronous binary up counter (q3q2q1q0).

1.1 Draw the logic diagram

1.2 Construct Verilog RTL representation for the logics with verification.

specification:

output:

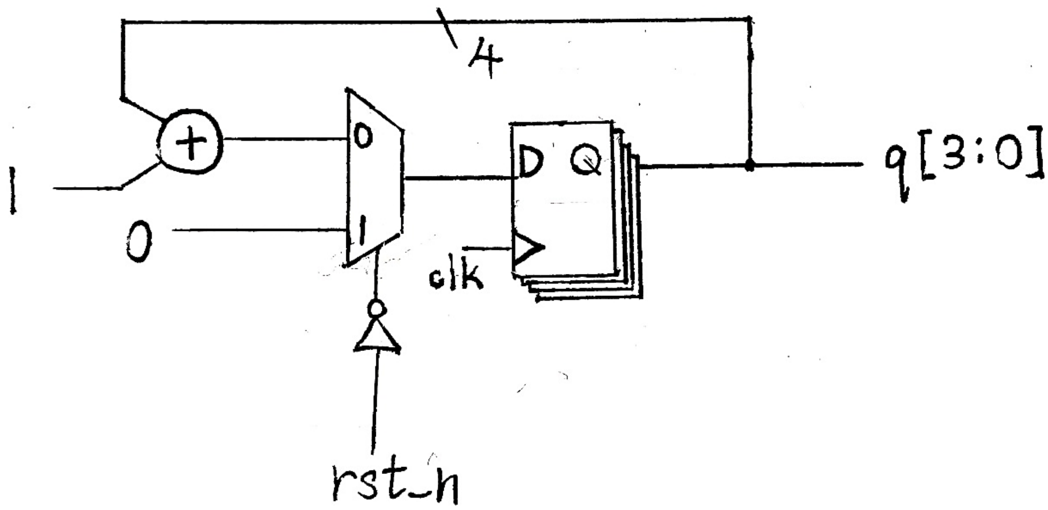
q[3:0]

input:

clk // clock

rst_n // active low reset

block-diagram/FSM:

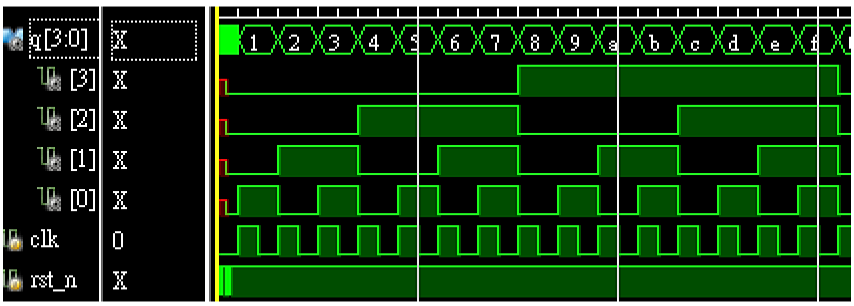


RTL representation:

```
module counter_up_binary_4bit(
    output reg [3:0] q,
    input            clk,   // clock
    input            rst_n  // active low reset
    );

    // 4-bit binary up counter
    always @ (posedge clk or negedge rst_n)
    begin
        if (~rst_n)
            q <= 0;         // reset
        else
            q <= q + 1;     // add 1
    end

endmodule                   // counter_up_binary_4bit
```

verification:

```
module counter_up_binary_4bit_test();
    // output
    wire [3:0] q;

    // input
    reg         clk;    // clock
    reg         rst_n;  // active low reset

    // instantiate
    counter_up_binary_4bit DUT0(
        .q(q),
        .clk(clk),      // clock
        .rst_n(rst_n)   // active low reset
        );

    // stimulus (initialization)
    initial
    begin
        #0 clk = 0;
        #2 rst_n = 1;
        #2 rst_n = 0;
        #2 rst_n = 1;
    end

    // stimulus (clock)
    always
        #10 clk = ~clk;

endmodule               // counter_up_binary_4bit_test
```



discussion:

這個 prelab 要做的是一個 4-bit 的二進位上數計數器，設計思路如下：用四個 FF 儲存當下數到多少，並將輸出 Q 連接到一個加法器。加一之後連到 FF 的 D 前面，等過一個時脈 clk 再存入 FF 更新數值。另，加法器與 FF 之間連接一個 MUX，用來 reset。

LAB3_pre_2

Cascade eight DFFs together as a shift register. Connect the output of the last DFF to the input of the first DFF as a ringer counter. Let the initial value of DFF output after reset be 01010101. Construct the Verilog RTL representation for the logics with verification.

specification:

output:

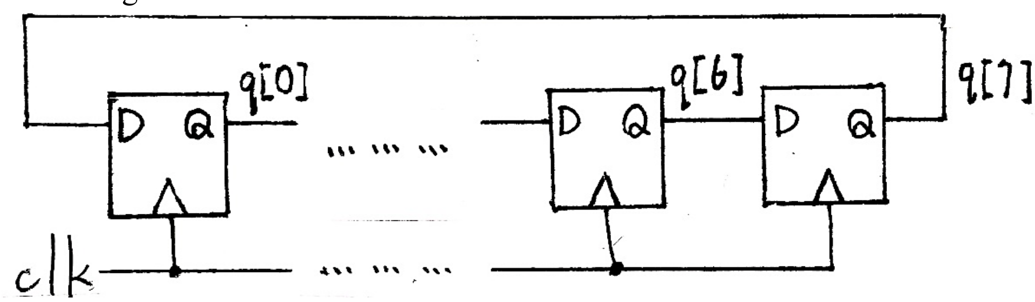
q[7:0]

input:

clk // clock

rst_n // active low reset

block-diagram/FSM:

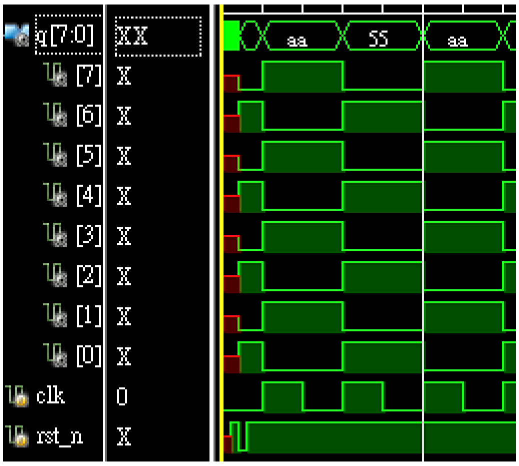


RTL representation:

```
module counter_ring_8bit(
    output reg [7:0] q,
    input            clk,    // clock
    input            rst_n   // active low reset
    );

    // 8-bit ring counter
    always @ (posedge clk or negedge rst_n)
    begin
        if (~rst_n)
        begin
            q <= 8'b01010101;   // reset
        end
        else
        begin
            q <= (q << 1);      // shift
            q[0] <= q[7];
        end
    end

endmodule                    // counter_ring_8bit
```

verification:

```
module counter_ring_8bit_test();
    // output
    wire [7:0] q;

    // input
    reg         clk;     // clock
    reg         rst_n;   // active low reset

    // instantiate
    counter_ring_8bit DUT0(
        .q(q),
        .clk(clk),       // clock
        .rst_n(rst_n)    // active low reset
        );

    // stimulus (initialization)
    initial
    begin
        clk = 0;
        #2 rst_n = 1;
        #2 rst_n = 0;
        #2 rst_n = 1;
    end

    // stimulus (clock)
    always
        #10 clk = ~clk;

endmodule                // counter_ring_8bit_test
```



discussion:

這個 prelab 要做的是 8-bit 的環形計數器，設計思路如下：拿八個 FF 來儲存要 shift 的內容，將這八個 FF 頭尾相連（上一個 FF 的 Q 連接到下一個 FF 的 D)，最後一個 FF 的 Q 則拉回第一個 FF 的 D 前面。如此一來，每一個時脈 clk 都會使這八個 FF 儲存的內容環繞移動一個單位。

LAB3_1

Frequency Divider: Construct a 27-bit synchronous binary counter. Use the MSB of the counter, we can get a frequency divider which provides a $1/2^{27}$ frequency output (fout) of the original clock (fcrystal, 100MHz). Construct a frequency divider of this kind.

1.1 Write the specification of the frequency divider.

1.2 Draw the block diagram of the frequency divider.

1.3 Implement the frequency divider with the following parameters.

I/O	fcrystal	fout
Site	W5	U16

specification:

output:

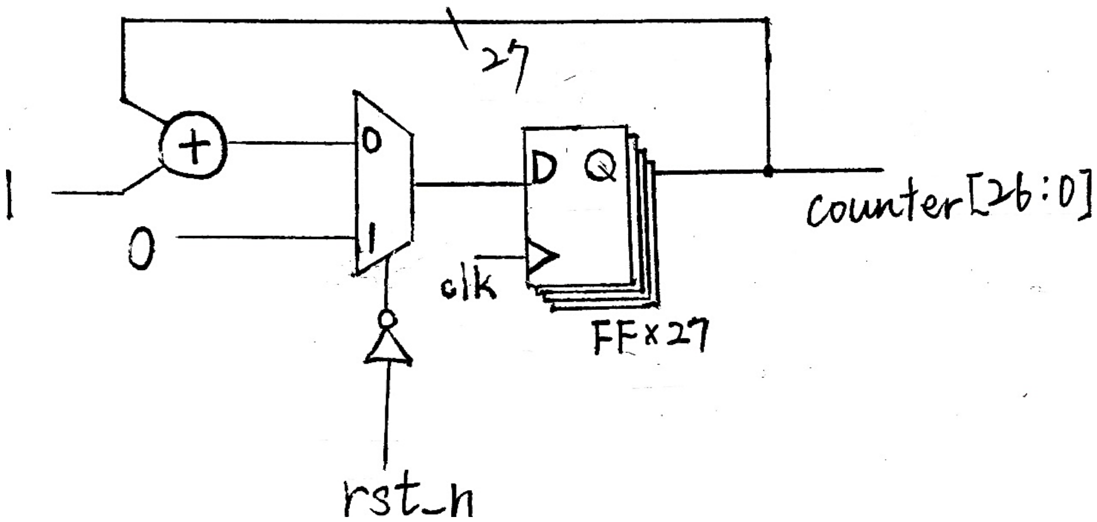
fout // around 1Hz but slightly slower

input:

fcrystal // 100M Hz global clock

rst_n // active low reset

block-diagram/FSM:



Verilog code:

```verilog
module frequency_divider_2e27(
    output fout,            // output frequency
    input  fcrystal,        // crystal frequency
    input  rst_n            // active low reset
    );
    reg [26:0] counter;     // 2^27 counter

    assign fout = counter[26];

    // 2^27 counter
    always @ (posedge fcrystal or negedge rst_n)
    begin
        if (~rst_n)
            counter <= 0;               // reset
        else
            counter <= counter + 1;     // add 1
    end

endmodule               // frequency_divider_2e27
```

implement:

fcrystal	IN	W5	LVCMOS33*
fout	OUT	U16	LVCMOS33*
rst_n	IN	V17	LVCMOS33*

discussion:

這個 LAB 要做的是一個除頻器，要將 FPGA 的十萬赫茲時脈的頻率除以二的二十七次方。設計思路如下：因為二進位計數器天生就是除頻器，所以直接參考 prelab1 的做法，將上數計數器中 FF 的數量增加到 27 個，再把最高位的 bit 當作輸出即可。

LAB3_2

Frequency Divider: Use a count-for-50M counter and some glue logics to construct a 1 Hz clock frequency. Construct a frequency divider of this kind.

2.1 Write the specification of the frequency divider.

2.2 Draw the block diagram of the frequency divider.

2.3 Implement the frequency divider with the following parameters.

I/O	**fcrystal**	**fout**
Site	**W5**	**U16**

specification:

output:

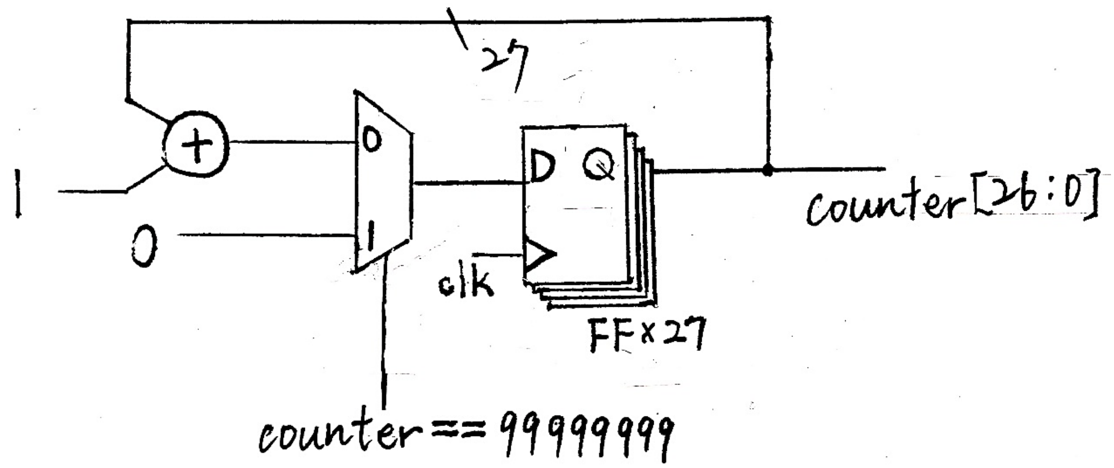
fout // 1Hz 50% duty

input:

fcrystal // 100M Hz clock

rst_n // active low reset

block-diagram/FSM:



Verilog code:

```
23 module frequency_divider_100M(
24     output fout,        // output frequency
25     input  fcrystal,    // crystal frequency
26     input  rst_n        // active low reset
27     );
28     reg [26:0] counter;
29     parameter DIVISOR = 100_000_000;
30 
31     assign fout = (counter < DIVISOR / 2) ? (0) : (1);   // 50% duty
32 
33     always @ (posedge fcrystal or negedge rst_n)
34     begin
35         if (~rst_n)
36             counter <= 0;   // reset
37         else
38             counter <= (counter < DIVISOR - 1) ? (counter + 1) : (0);
39     end
40 
41 endmodule                    // frequency_divider_100M
```

implement:

fcrystal	IN	W5	LVCMOS33*
fout	OUT	U16	LVCMOS33*
rst_n	IN	V17	LVCMOS33*

discussion:

這題跟上一題一樣要做除頻器，但這次要將 FPGA 的十萬赫茲時脈處理成剛剛好一赫茲。我沿用上一題的概念，並稍加修改：由於二的二十七次方比十萬稍大，所以這題須要另外多拿一個比較器來用。如果那 27 個 FF 的數值數到十萬減一，就讓 MUX 輸出零，使 FF 的數值在下一個時脈歸零。然而，這麼做雖然成功地製造出了一赫茲的輸出，但卻不是完美的一半一半（50% duty）。所以，我最後再多拿一個比較器來用，如果那 27 個 FF 的數值小於十萬的一半，就輸出零，如果大於則輸出一，如在 Verilog code 的 line 31 處所示。

LAB3_3

Implement pre-lab1 with clock frequency of 1 Hz. Use the following I/O to demonstrate the counter results.

I/O	fcrystal	q3	q2	q1	q0
Site	W5	V19	U19	E19	U16

specification:

output:

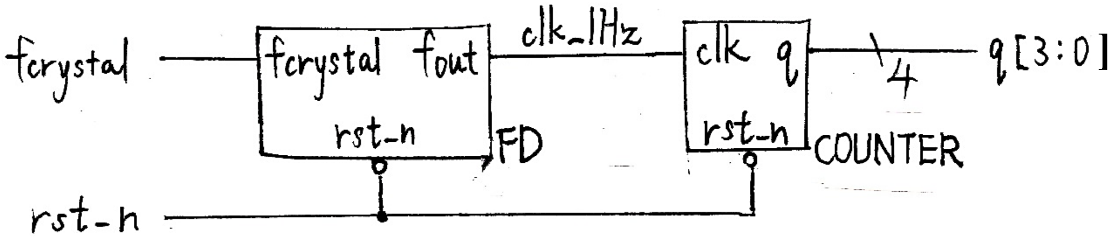
q[3:0] // prelab1 output

input:

fcrystal // 100M Hz global clock

rst_n // active low reset

block-diagram/FSM:



TOP:

```
module LAB3_3(
    output [3:0] q,
    input fcrystal,      // global clock
    input rst_n          // active low reset
    );
    wire clk_1Hz;

    frequency_divider_100M FD(
        .fout(clk_1Hz),           // 1 Hz clock
        .fcrystal(fcrystal),      // crystal frequency (global clock)
        .rst_n(rst_n)             //  active low reset
        );

    counter_up_binary_4bit COUNTER(
        .q(q),
        .clk(clk_1Hz),            // 1 Hz clock
        .rst_n(rst_n)             // active low reset
        );

endmodule                         // LAB3_3
```

implement:

q (4)	OUT		LVCMOS33*
q[3]	OUT	V19	LVCMOS33*
q[2]	OUT	U19	LVCMOS33*
q[1]	OUT	E19	LVCMOS33*
q[0]	OUT	U16	LVCMOS33*
Scalar ports (2)			
fcrystal	IN	W5	LVCMOS33*
rst_n	IN	V17	LVCMOS33*

discussion:

我在上一題寫的除頻器性質不夠好，所以不能直接拿我上一題的除頻器作為 prelab1 的時脈。將上一題的除頻器部分修改重寫後的 code 如下圖：

```
23 module frequency_divider_100M(
24     output reg fout,       // output frequency
25     input      fcrystal,   // crystal frequency
26     input      rst_n       // active low reset
27     );
28     reg [26:0] counter;
29
30     // count for 50M times
31     always @ (posedge fcrystal or negedge rst_n)
32     begin
33         if (~rst_n)
34             counter <= 1;  // reset
35         else
36             counter <= (counter < 50_000_000) ? (counter + 1) : (1);
37     end
38
39     // creat 50% duty output
40     always @ (posedge fcrystal or negedge rst_n)
41     begin
42         if (~rst_n)
43             fout <= 0;     // reset
44         else begin
45             if (counter == 1)
46                 fout <= ~fout;       // 50% duty
47     end end
48
49 endmodule                  // frequency_divider_100M
```

我認為上一題的除頻器的問題在於把輸出變成 50% duty 的方法不妥。這一題我改變方法，直接拿一個 TFF（line 39~47），將 FF 的最高位與 T 相接。並修改比較器的 code（line 36），讓 FF 的數值數到十萬的一半時就重設成一。透過這樣的方式所產生的 1Hz 時脈，與上一題一樣都是完美的 50% duty，而且可以直接連接到 prelab1 當作四位元計數器的時脈。

LAB3_4

Implement pre-lab2 with clock frequency of 1 Hz. The I/O pins can be assigned by yourself.

specification:

output:

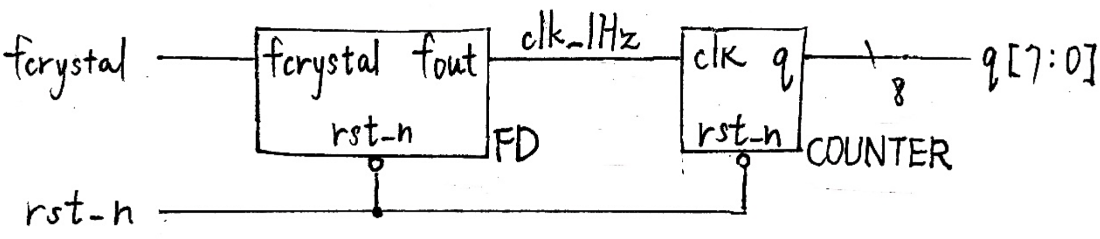
q[7:0]	// output from prelab2

input:

fcrystal	// 100M Hz global clock

rst_n	// active low reset

block-diagram/FSM:



TOP:

```
module LAB3_4(
    output [7:0] q,
    input        fcrystal,     //global clock
    input        rst_n         //active low reset
    );

    frequency_divider_100M FD(
        .fout(clk_1hz),        //1 Hz clock
        .fcrystal(fcrystal),   //crystal frequency (global clock)
        .rst_n(rst_n)          // active low reset
        );

    counter_ring_8bit COUNTER(
        .q(q),
        .clk(clk_1hz),         //1 Hz clock
        .rst_n(rst_n)          //active low reset
        );

endmodule                      //LAB3_4
```

implement:

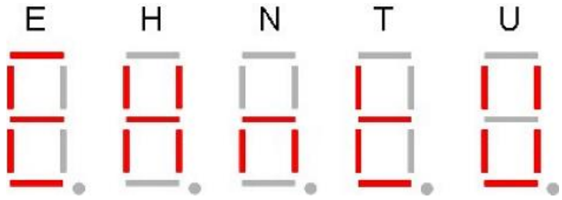
q (8)	OUT		LVCMOS33*
q[7]	OUT	V14	LVCMOS33*
q[6]	OUT	U14	LVCMOS33*
q[5]	OUT	U15	LVCMOS33*
q[4]	OUT	W18	LVCMOS33*
q[3]	OUT	V19	LVCMOS33*
q[2]	OUT	U19	LVCMOS33*
q[1]	OUT	E19	LVCMOS33*
q[0]	OUT	U16	LVCMOS33*
Scalar ports (2)			
fcrystal	IN	W5	LVCMOS33*
rst_n	IN	V17	LVCMOS33*

discussion:

這一題跟上一題差不多，只差在這題是把一赫茲的時脈連接到 prelab2 的環形計數器。但由於題目訂定環形計數器的八個 bits 儲存的內容是 01010101，所以實作之後無法從 FPGA 板的 LED 上看出儲存的 bits 在繞圈（環狀移動），只感覺它們在交替明滅。必須如下一題一般，設置不重複的 bits 才能看出循環效果。

LAB3_5

Use the idea from pre-lab2. 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.



specification:

output:

digit_active[3:0] // common anode activation

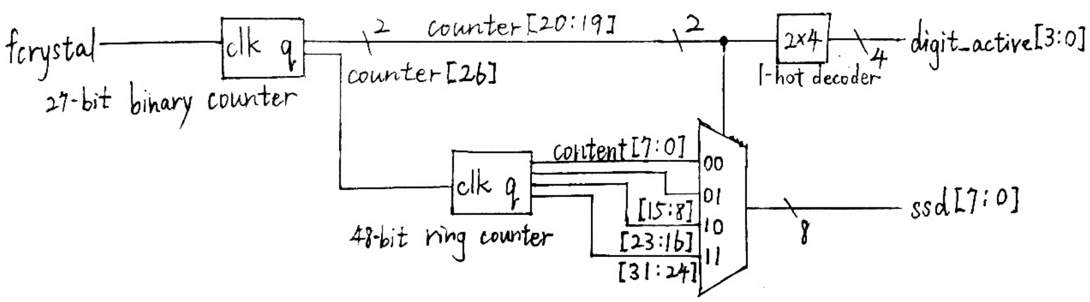
ssd[7:0] // 7 segment display

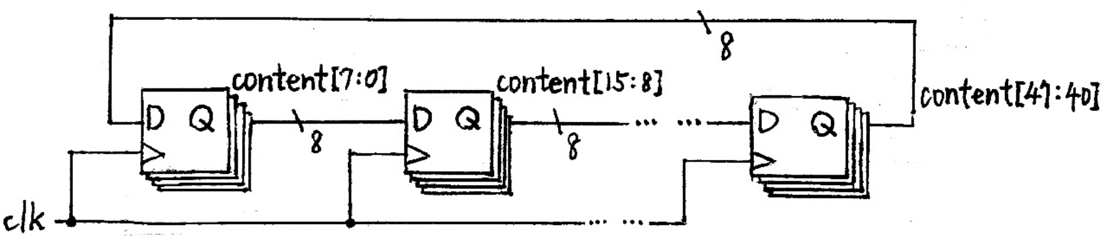
input:

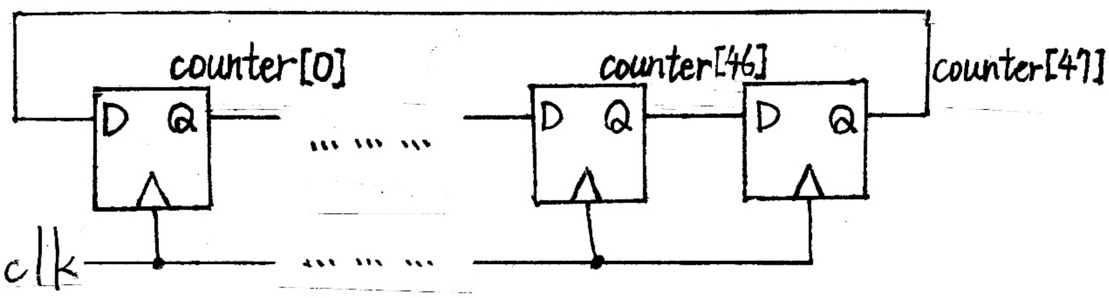
fcrystal, // 100M Hz global clock

rst_n // active low reset

block-diagram/FSM:







```
module LAB3_5(
    output reg [3:0] digit_active,  // common anode
    output reg [7:0] ssd,           // 7 segment display
    input fcrystal,                 // 100M Hz clock
    input rst_n                     // active low reset
    );
    reg [26:0] counter;             // control scrolling speed
    reg [47:0] content;             // pre-stored pattern NTHUEE

    // timer to control the scrolling speed of "NTHUEE" displayed
    always @ (posedge fcrystal or negedge rst_n)
    begin
        if (~rst_n)
            counter <= 0;           // reset
        else
            counter <= counter + 1; // add 1
    end

    // a 8-shifting-step ring-counter for pre-stored pattern "NTHUEE"
    always @ (negedge counter[26] or negedge rst_n)
    begin
        if (~rst_n)
            // letter            N        T        H        U        E        E
            content <= 48'b11010101_11100001_10010001_10000011_01100001_01100001;
        else begin
            content[ 7:0] <= content[47:40];  // shift 8 steps
            content[47:8] <= content[39: 0];  // shift 8 steps
    end end

    // a 2-to-4 1-hot decoder for common anode activation
    always @ (counter[20:19])
        case (counter[20:19])              // take a suitable refresh frequency
            2'b00:   digit_active <= 4'b1110; // 0
            2'b01:   digit_active <= 4'b1101; // 1
            2'b10:   digit_active <= 4'b1011; // 2
            2'b11:   digit_active <= 4'b0111; // 3
            default: digit_active <= 0;
        endcase

    // sending the corresponding signal pattern to the SSD at the right timing
    always @ (counter[20:19])
        case (counter[20:19])              // take a suitable refresh frequency
            2'b00:   ssd <= content[ 7: 0];   //  first digit (letter)
            2'b01:   ssd <= content[15: 8];   // second digit (letter)
            2'b10:   ssd <= content[23:16];   //  third digit (letter)
            2'b11:   ssd <= content[31:24];   //   last digit (letter)
            default: ssd <= 0;
        endcase

endmodule                              // LAB3_5
```

implement:

digit_acti...	OUT		LVCMOS33*
digit_a...	OUT	W4	LVCMOS33*
digit_a...	OUT	V4	LVCMOS33*
digit_a...	OUT	U4	LVCMOS33*
digit_a...	OUT	U2	LVCMOS33*
ssd (8)	OUT		LVCMOS33*
ssd[7]	OUT	W7	LVCMOS33*
ssd[6]	OUT	W6	LVCMOS33*
ssd[5]	OUT	U8	LVCMOS33*
ssd[4]	OUT	V8	LVCMOS33*
ssd[3]	OUT	U5	LVCMOS33*
ssd[2]	OUT	V5	LVCMOS33*
ssd[1]	OUT	U7	LVCMOS33*
ssd[0]	OUT	V7	LVCMOS33*
Scalar ports (2)			
fcrystal	IN	W5	LVCMOS33*
rst_n	IN	V17	LVCMOS33*

discussion:

這題要做的是一個能顯示「NTHUEE」的七段顯示器。然而，FPGA 上的七段顯示器只有四個，塞不下六個英文字母，因此需要用計時捲動的方式輪流顯示。本題的設計思路如下：沿用第一題的二十七位二進位上數計數器，取第十九與二十位的頻率轉成 1-hot 並接到七段顯示器的 common anode。利用人眼的視覺暫留使七段顯示器看起來同時顯示四個不同的字母。另拿一個四十八位元的環形計數器，用二進位計數器的第二十六位來當作時脈，一次位移八個位元。如此一來，該環形計數器就能將預先儲存好的字母樣式定時捲動，再透過 MUX 連接到七段顯示器上。