106010006 黃詩瑜 電機 21

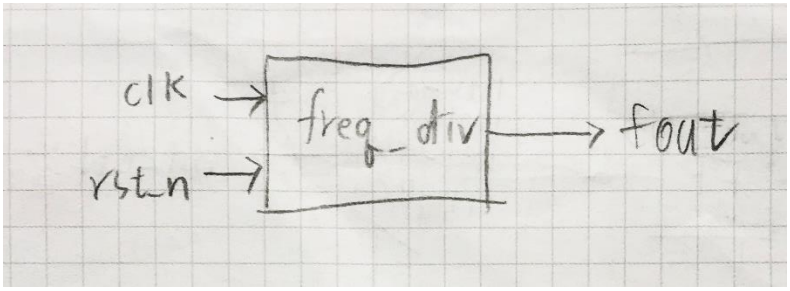
Lab 3: Counters and Shift Registers I

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 (f_{out}) of the original clock ($f_{crystal}$, 100MHz). Construct a frequency divider of this kind.

1.1 Write the specification of the frequency divider.

Output : fout

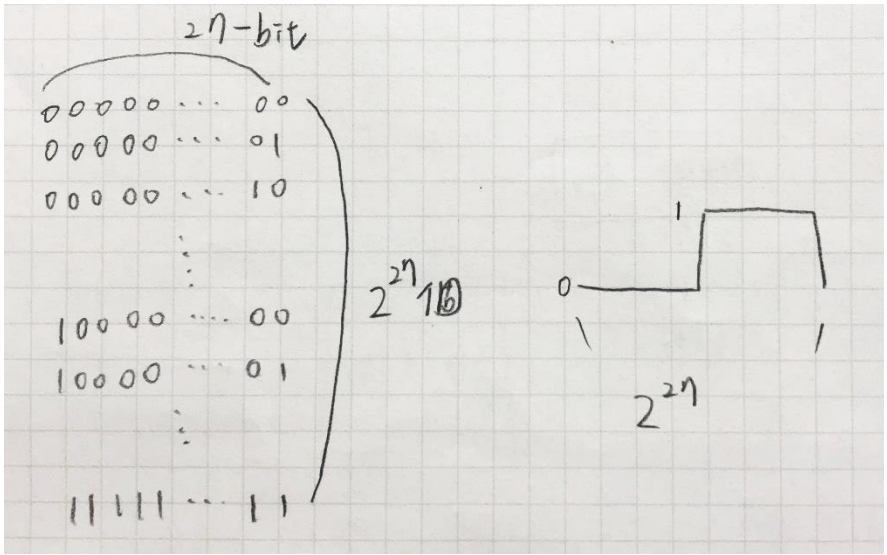
Input : clk,rst_n



1.2 Draw the block diagram of the frequency divider.

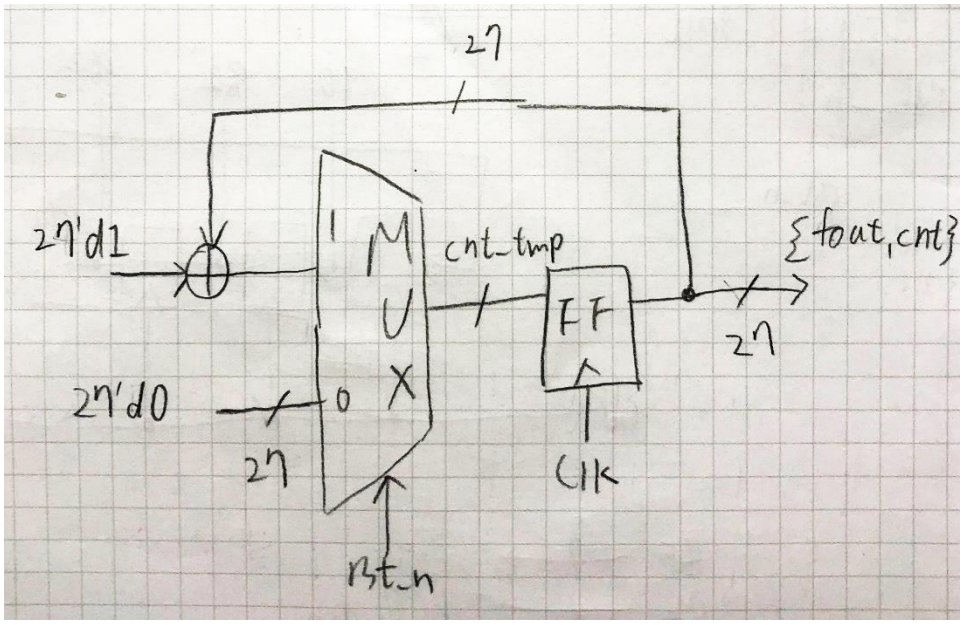
1.3 Implement the frequency divider with the following parameters.

I/O	$f_{crystal}$	f_{out}
Site	W5	U16



原本的clock頻率是100MHz，要將頻率變成$1/2^{27}$倍，就是利用27個

bits，讓clock從0數到27個bits都為1後又歸零，這樣一直循環，再把27個bits的最左邊的數字拉出來，會發現從0~2^{27}-1前半段它都是0，後半段都是1，然後0-1-0…一直循環，這就可以形成一個頻率變成1/2^{27}倍的clock了。



將filp-flop接上原本100MHz的clock，出來是27個bits，再把它拉到前面加1，出來是27個bits的cnt_tmp，然後又進入flip-flops一直循環，FF出來的27個bits，將最左邊的bit設為fout，就是我們的output，剩下的26個bits是cnt，當input rst_n為0時，counter將會歸零重0開始數，所以做一個MUX判斷rst_n的值，若是1，則繼續數，一個reset的功能。

Input clk接到FPGA板的W5，那就是原本的100MHz clock，將fout接到U16，它閃爍的頻率就是100M/2^{27}，週期就是1.3…，接近1秒，rst_n接到V17 switch，向上撥為1，向下為0，可以將counter reset。

Module code:

```
`define FREQ_DIV_BIT 27
module lab3_1(
fout, // divided clock output
clk, // global clock input
rst_n // active low reset
);
output fout;
input clk;
input rst_n;
reg fout; // clk output (in always block)
reg [`FREQ_DIV_BIT-2:0] cnt; // remainder of the counter
reg [`FREQ_DIV_BIT-1:0] cnt_tmp; // input to dff (in always block)
// Combinational logics: increment, neglecting overflow
always @*
cnt_tmp = {fout,cnt} + `FREQ_DIV_BIT'd1;
// Sequential logics: Flip flops
always @(posedge clk or negedge rst_n)
if (~rst_n)
{fout, cnt}<=`FREQ_DIV_BIT'd0;
else
{fout,cnt}<=cnt_tmp;
Endmodule
```

討論:

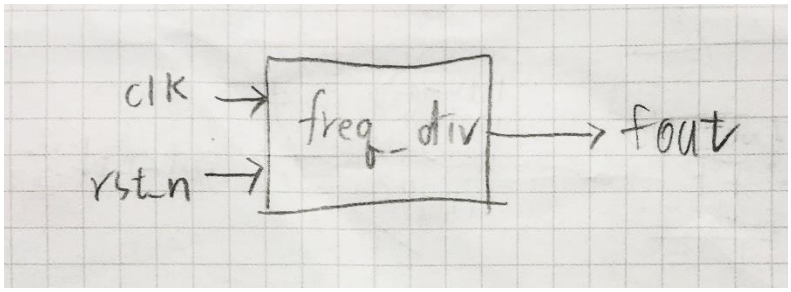
有點忘記 FF 和 counter，所以有上網看相關的教學影片找回記憶，

在 FF 的 always 中，是要使用 negative 還是 positive，想了蠻久的。

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.

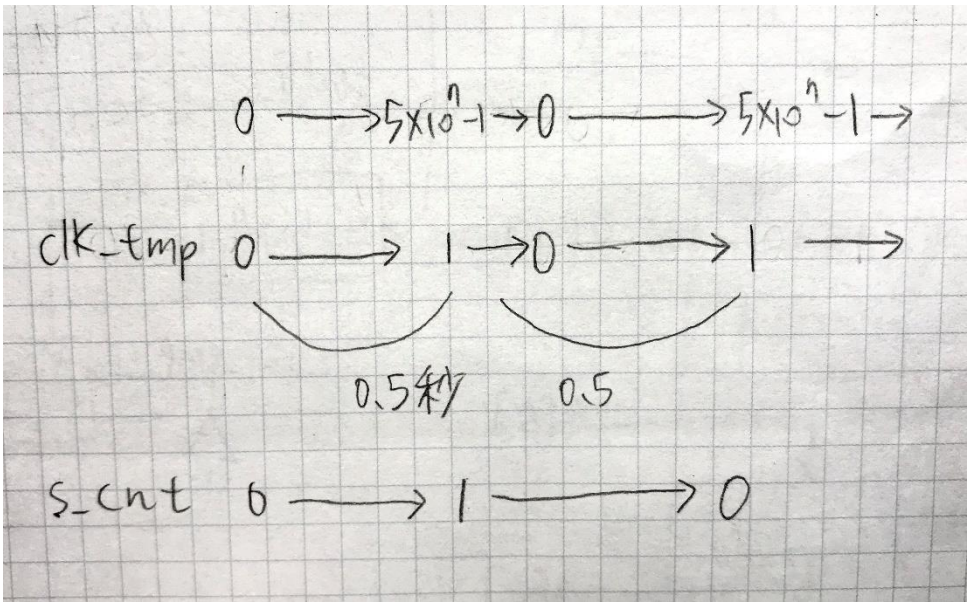
2.1 Write the specification of the frequency divider.

Output : fout
Input : clk,rst_n



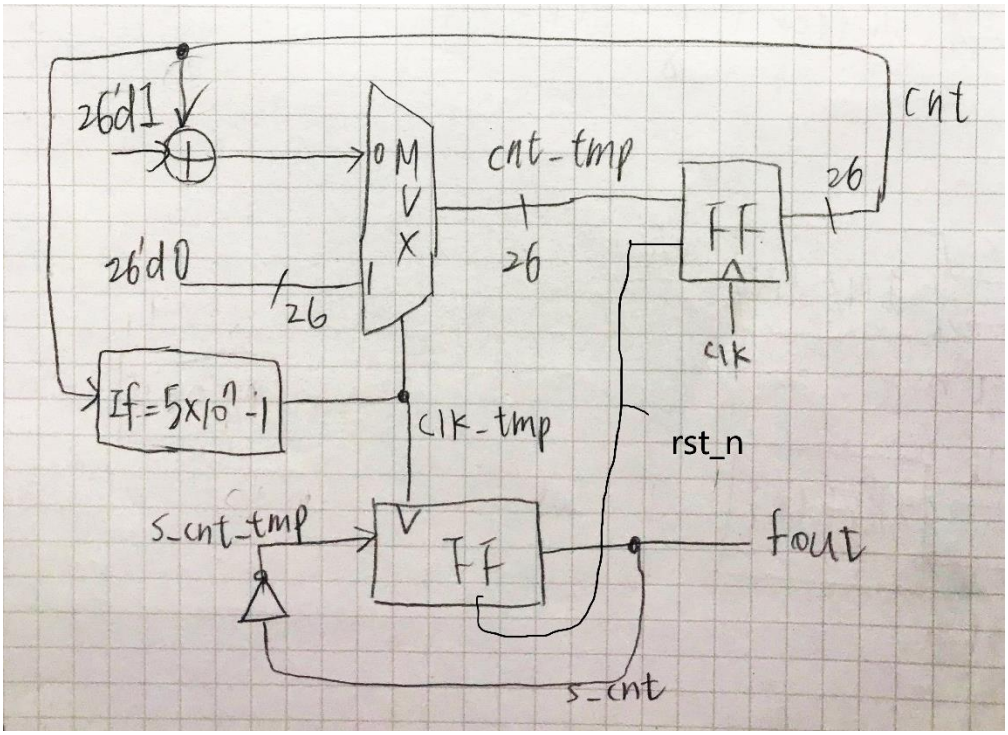
2.2 Draw the block diagram of the frequency divider.
2.3 Implement the frequency divider with the following parameters.

I/O	$f_{crystal}$	f_{out}
Site	W5	U16



這題是要做出 1Hz 的 clock，讓原本的 100MHz 的 clock 從 0 數到 $5x10^7$-1，這樣就是數了 50M，再歸零，一直循環，當數到 $5x10^7$-1 的時候，clk_tmp 從 0 變 1，100M 的 counter 歸零，clk_tmp 又會變成 0，所以 clk_tmp 每隔 0.5 秒就會變成 1，就這樣一直循還，再設計一個 s_cnt，可以透過 clk_tmp 的每 0.5 秒變化，讓 s_cnt 從 0 過 0.5 秒後變 1，再過 0.5 秒又變 0，這樣一直循還，就是一個 1Hz 的

Clock 了。



將 filp-flop 接上原本 100MHz 的 clock，出來是 26 個 bits，再把它拉到前面加 1，出來是 26 個 bits 的 cnt_tmp，然後又進入 flip-flops 一直循環，設計一個 if，判斷是否從 0 數到 $5x10^7$-1 了，如果是 true，clk_tmp 會變成 1，讓 MUX 選擇 26 個 bits counter 歸零，然後 clk_tmp 會接到另一個 FF，在 clk_tmp 變 1 之前，s_cnt 為 0，s_cnt_tmp 為 1，當 clk_tmp 變 1，s_cnt_tmp 的值就會傳出去，s_cnt 變 1，fout 就會由 0 變 1，s_cnt_tmp 變 0，當又過 0.5 秒，clk_tmp 又變 1，s_cnt_tmp 的值又會傳出去，然後 fout 就會形成每 0.5 秒 0-1-0-1….的變化，就是 1Hz 的 clock。

把Input clk接到FPGA板的W5，那就是原本的100MHz clock，將fout接

到U16，它就會1秒閃一次，rst_n接到V17 switch，向上撥為1，向下為0，可以將counter reset。

Module code:

```
module lab3_2(
fout, // divided clock output
clk, // global clock input
rst_n // active low reset
);
output fout; // divided output
input clk; // global clock input
input rst_n; // active low reset
reg [26:0] cnt_tmp,cnt;
reg clk_tmp,s_cnt,s_cnt_tmp; // input to dff (in always block)
always @(cnt)
	if(cnt == 49999999) cnt_tmp <= 0;
	else cnt_tmp <= cnt + 27'd1;
//clk_temp 0.5s
always @(cnt)
	if(cnt == 49999999) clk_tmp <= 1;
	else clk_tmp <= 0;

always @(s_cnt)
	s_cnt_tmp <= ~s_cnt ;

assign fout = s_cnt;

//Flip flop with 0.5s freq
always @(posedge clk_tmp or negedge rst_n)
	if (~rst_n) s_cnt <= 0;
	else s_cnt <= s_cnt_tmp;
// Flip flops
always @(posedge clk or negedge rst_n)
	if (~rst_n) cnt <= 27'b0;
	else cnt <= cnt_tmp;
```

endmodule

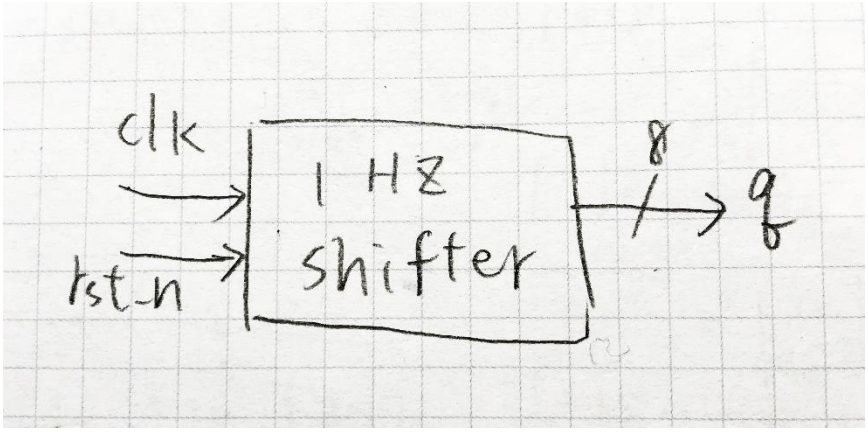
討論:

這題遇到的困難是一開始如何設計出來，有這想法之後再把電路畫出來其實就比較沒什麼問題了，主要是一開始要怎麼設計才是重點。

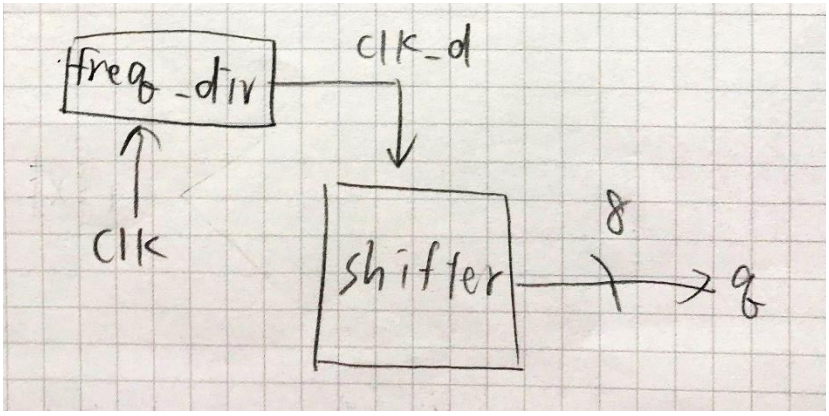
3. Implement pre-lab2 with clock frequency of 1 Hz.

Specification:

Input : clk,rst_n
Output : q



Implementation:



這題只要將prelab第2題的shifter與第2題做的1Hz clock接起來就行

了，原本100MHz的clock，接到第二題做的freqency divider的input clk，freq_div的output是1Hz的clock，將它設為clk_d，並接到prelab2的shifter中的input clk，當作shifter的clock，shifter的output 8-bit q，就會每秒都shift一次。

照著講義的方式，將freq_div.v和shifter.v的檔案叫進來，把Input clk接到FPGA板的W5，那就是原本的100MHz clock，將q接到8個LED燈，clock還未啟動時是10101010，1為亮0為不亮，rst_n接到V17 switch，向上撥為1，向下為0，可以將counter reset，當clock啟動時，8-bit q就會每1秒由101010101變01010101一直循環，對應的LED燈就會照著閃爍。

Module code:

```
module lab3_3shiftreg(
q, // LED output
clk, // global clock
rst_n // active low reset
);
output [7:0] q; // LED output
input clk; // global clock
input rst_n; // active low reset
wire clk_d;
wire [7:0] q;
// Insert frequency divider (freq_div.v)
lab3_2 U_FD(
.fout(clk_d), // divided clock output
.clk(clk), // clock from the crystal
.rst_n(rst_n) // active low reset
);
```

```
// Insert shifter (shifter.v)
prelab3_2 U_D(
.q(q), // shifter output
.clk(clk_d), // clock from the frequency divider
.rst_n(rst_n) // active low reset
);

Endmodule
```

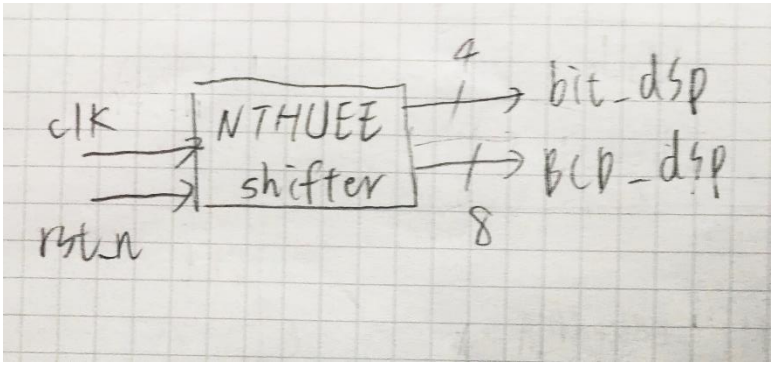
討論:

這題只要之前的code都完成後，照著講義的打法，將兩個.v檔串在一起就行了。

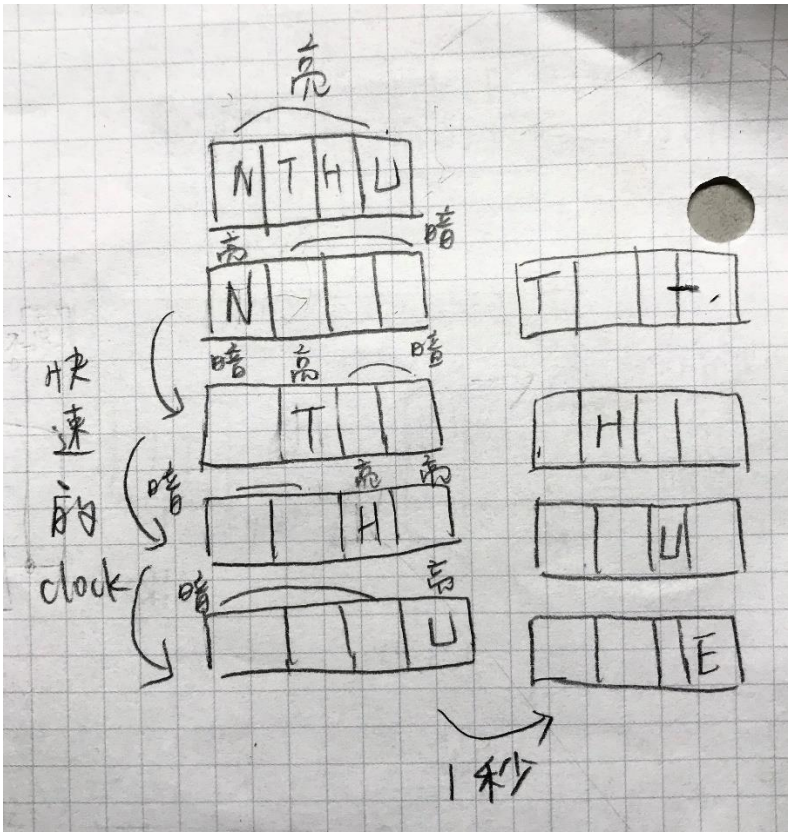
4. Use the idea from pre-lab2. We can do something on the seven-segment display. Assume we have the pattern of E, H, N, T, U for seven-segment display as shown below. Try to implement the scrolling pre-stored pattern NTHUEE with the four seven-segment displays.

Specification:

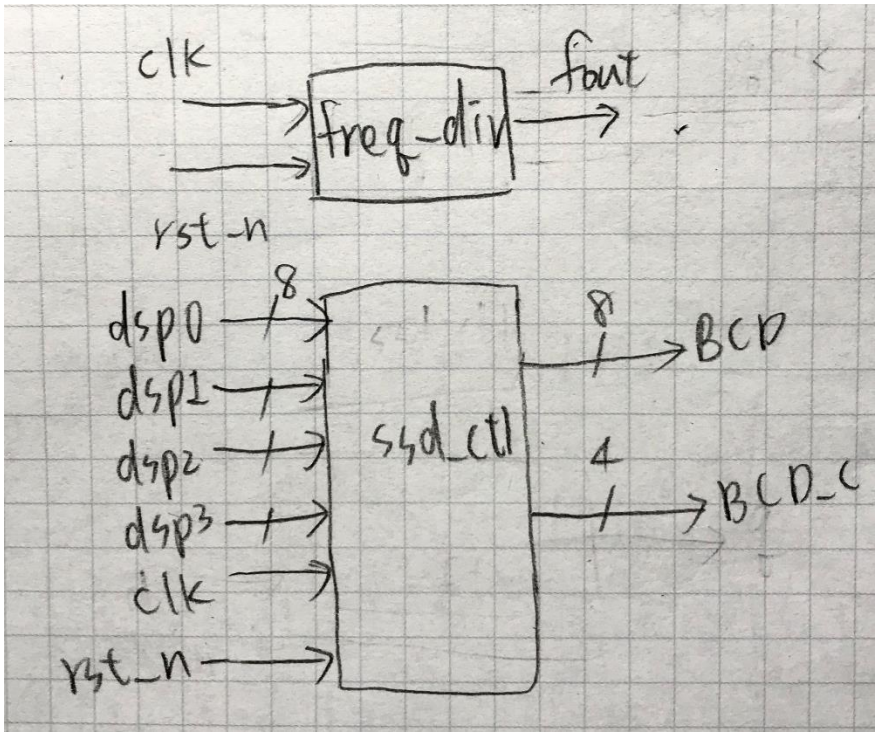
Input : clk,rst_n
Output : bit_dsp,BCD_dsp



Implementation:

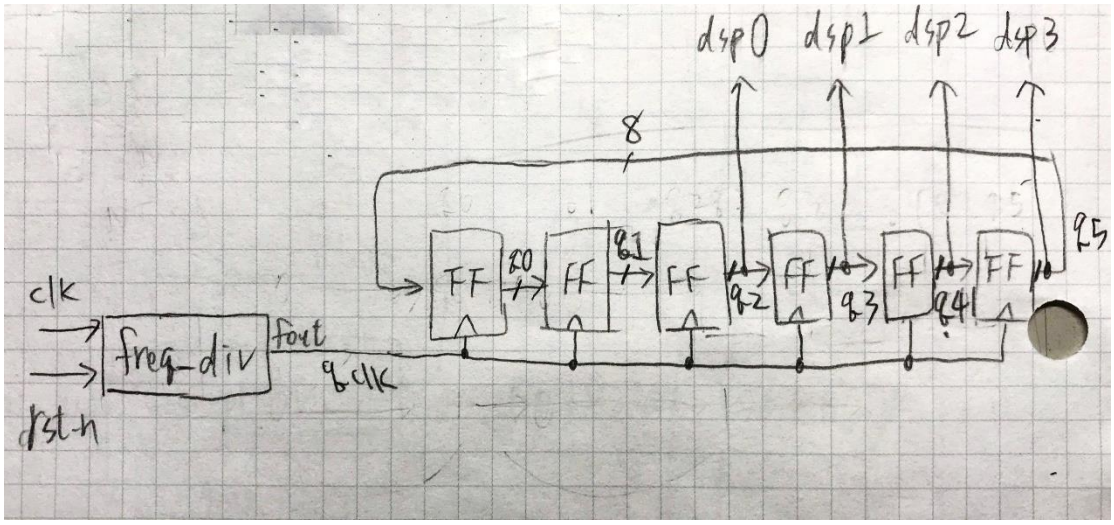


因為7-seg一次只能顯示相同的字母，所以要利用LED燈快速閃爍與視覺停留呈現出4個不同字母的感覺，設定最左邊的燈亮，其他暗，顯示一個英文字N，再讓第二個燈亮其他暗，顯示另一個英文字T，當快速閃爍，就可以看到4個不同的英文字NTHU，當過了一秒，將英文字NTHU切換成THUE，且一直循環，就可以看到NTHUEE的跑馬燈。

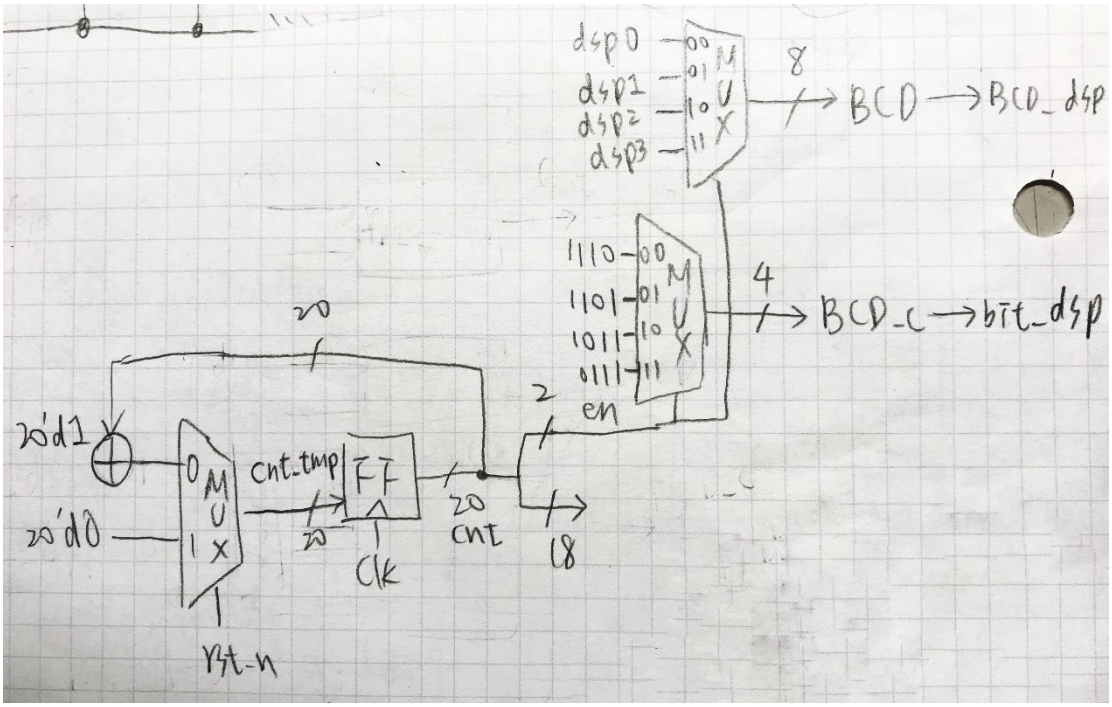


我分成3個.v檔，一個是frequncy divider，是一個1Hz的counter，使用第2題做出來的檔案，input是原本的100MHz clock和用來控制reset的rst_n，output fout是1Hz的clock。

一個是ssd_ctl，是用來控制7-seg要亮哪個燈與亮甚麼英文字，input是原本的100MHz clock和用來控制reset的rst_n，4個8 bits的input dsp是要亮的數字，output 8-bit BCD是要亮的數字，output 4-bit BCD_c是要亮7-seg的哪個燈。



將1Hz freq_div的fout接到qclk，qclk是6個filp-flops的clock，6個FF做成shifter，q5~q0一開始是對應到NTHUEE這5個英文字，這樣就能每秒shift一次英文字，再將q5,q4,q3,q2接到dsp3,dsp2,dsp1,dsp0，就是將要顯示再7-seg上的4個英文字。



ssd_ctl內部有一個frequency divider，將原本100MHz的clock頻率變小，FF出來是20-bit cnt，會一直加1從0數到2^{20}-1然後又歸零，將最

左邊的2個bits取出來設為en，就是頻率比100MHz小的clock了，這個en會接到兩個MUX，一個是選擇亮7-seg的4個燈的哪一個，另一個是選擇亮的那個燈要顯示的英文字，出來的output是BCD_c和BCD，再把他們接給bit_dsp與BCD_dsp，4 bits的output bit_dsp會接到FPGA板的U2,U4,V4,W4，0為亮1為不亮，就可以控制7-seg亮的燈，8 bits的output BCD_dsp是控制要顯示什麼英文字，0為亮1為不亮。

把3個.v檔接在一起，就是一個NTHUEE的跑馬燈了。

Module code:

ssd_ctl:

```
module lab3_4ssd_ctl(
    input [7:0]dsp0,
    input [7:0]dsp1,
    input [7:0]dsp2,
    input [7:0]dsp3,
    input clk,
    output reg [7:0]BCD,
    output reg [3:0]BCD_c,
    input rst_n
    );

    //counter
    reg [19:0] cnt,cnt_tmp;
    //+1
    always @(cnt)
        cnt_tmp <= cnt+1;
    // Flip flops
    always @(posedge clk or negedge rst_n)
        if (~rst_n) cnt <= 19'b0;
        else cnt <= cnt_tmp;
```

```
    reg [7:0]BCD_tmp;
  /*抓出最左邊的2個bits*/
    wire [1:0]en;
    assign en={cnt[19],cnt[18]};
   /*選擇要亮的英文字*/
    always @(en)
    case(en)
        2'b00: BCD = dsp0;
        2'b01: BCD = dsp1;
        2'b10: BCD = dsp2;
        2'b11: BCD = dsp3;
        default: BCD_tmp = 8'b11111111;
    endcase

      /*選擇要亮哪個燈，0為亮1為不亮*/
    always @(en)
    case(en)
        2'b00: BCD_c = 4'b1110;
        2'b01: BCD_c = 4'b1101;
        2'b10: BCD_c = 4'b1011;
        2'b11: BCD_c = 4'b0111;
        default: BCD_tmp = 4'b0000;
    endcase
endmodule

/*8個bits控制7-seg要顯示的英文字，0為亮1為不亮，先將它們define給對應的
SS_N~E*/
`define SS_N 8'b11010101
`define SS_T 8'b11100001
`define SS_H 8'b10010001
`define SS_U 8'b10000011
`define SS_E 8'b01100001

module nthuee(
    input clk,
    input rst_n,
    output [7:0]BCD_dsp,
    output [3:0]bit_dsp
```

```
    );
    wire qclk; //1Hz的clock

    /*接上1Hz的frequency divider和ssd_ctl*/
    lab3_4fre U0 (.clk(clk),.rst_n(rst_n),.fout(qclk));
    lab3_4ssd_ctl U1 (.dsp3(q5),.dsp2(q4),.dsp1(q3),.dsp0(q2),
                      .clk(clk),.BCD(BCD_dsp),.BCD_c(bit_dsp),.rst_n(rst_n));
    reg [7:0]q0,q1,q2,q3,q4,q5;

/*一開始將NTHUEE分別設定給q5~q0*/
    always @(posedge qclk or negedge rst_n)
        if (~rst_n) begin
            q5<=`SS_N;
            q4<=`SS_T;
            q3<=`SS_H;
            q2<=`SS_U;
            q2<=`SS_E;
            q0<=`SS_E;

        end

   /*將這6個英文字每1秒shift一次*/
        else begin
            q0<=q5;
            q1<=q0;
            q2<=q1;
            q3<=q2;
            q4<=q3;
            q5<=q4;

        end

endmodule
```

討論:

這題花了我最久的時間，一開始完全搞不懂怎麼利用視覺佔留設計

出跑馬燈，向助教詢問了很久之後才搞懂，一開始出現很多bug，像是case的最後一項要是default，或是閃爍的clock太慢或太快，都會造成實驗失敗，後來修改很多次之後，才成功。

結論:

這次實驗可以感覺到難度越來越難了，之前都是實驗課的前一天才開始做，之後就要平均分配不要積到最後一天才開始做，不然又要熬夜了。