**Lab 4: Shift Registers**

**1. Pre-lab and implementation.**

**Design Specification**

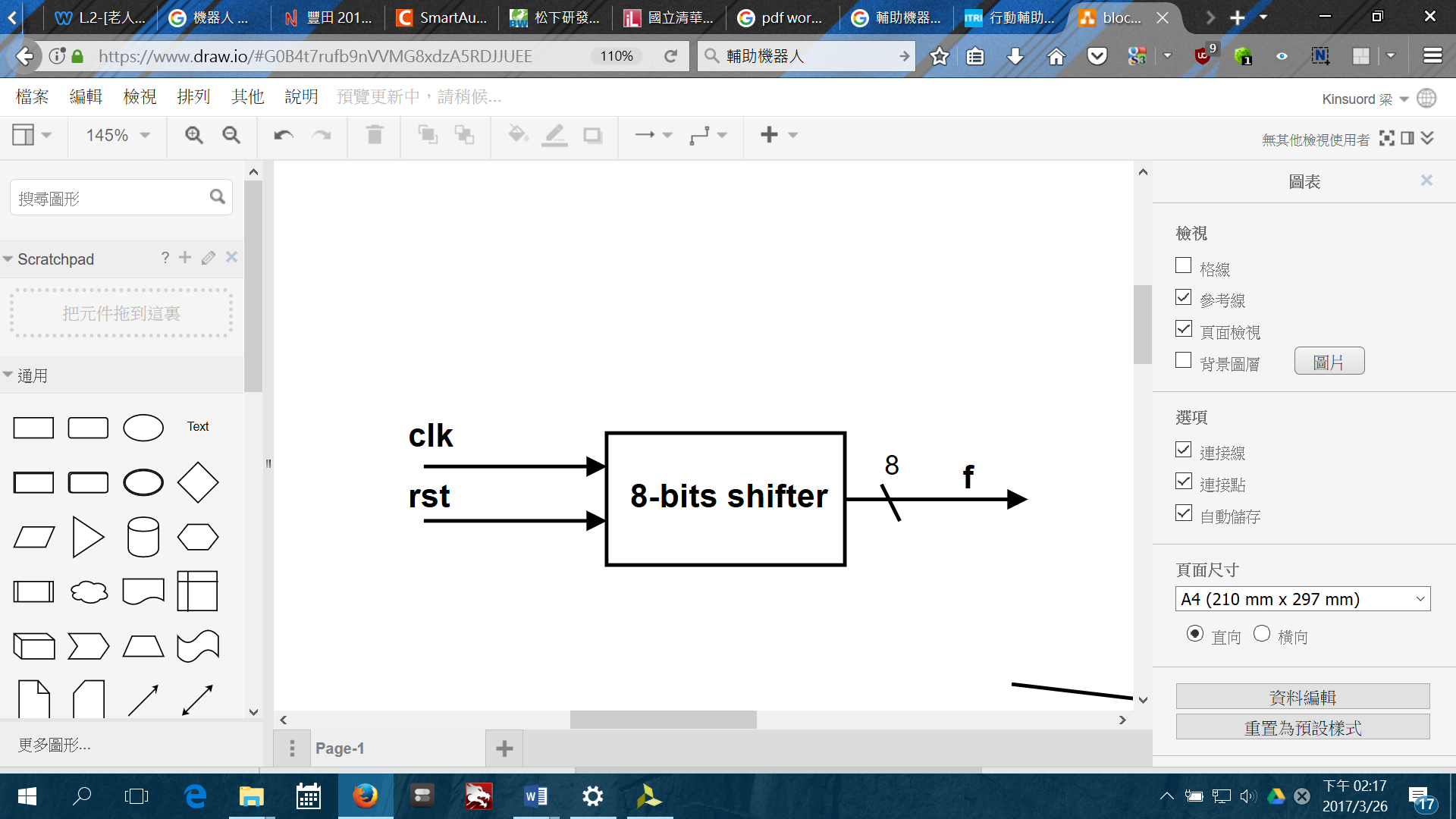
(1) I/O ports

input rst 等同於此暫存器的開關，採正緣觸發

input clk 時脈，採正緣觸發

output reg [7:0] f; 所有DFF裡面暫存的值

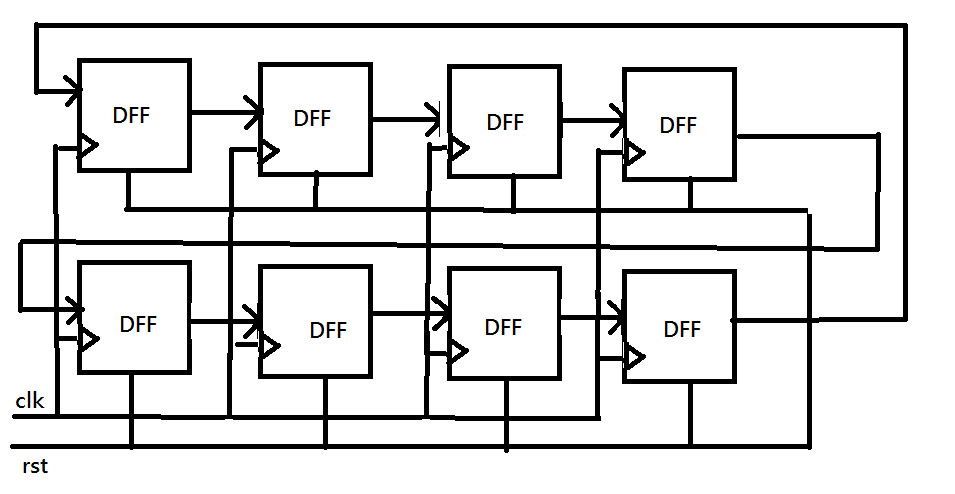
(2) Block Diagram



**Design Implementation**

(1) Logic Diagram

　　Ring Counter 顧名思義，只要把 8 個 DFF 頭尾串聯起來就是了。然而實際燒到FPGA板子上時所接的clock會是由lab3中所做的除頻器產生的１Hz的訊號。



(2) Verilog codes

將prelab(左側)與lab4\_1的code比較一下。

`timescale 1ns / 1ps

module lab4\_1(rst, f, clk, clk\_1hz);

input clk;

input rst;

output wire clk\_1hz;

output reg [7:0] f;

lab3\_2 U0(.rst\_n(rst), .signal(clk\_1hz), .clk(clk));

always@ (posedge clk\_1hz or negedge rst)

if (~rst)

f <= 8'b01010101;

else

begin

f[0]<=f[7];

f[1]<=f[0];

f[2]<=f[1];

f[3]<=f[2];

f[4]<=f[3];

f[5]<=f[4];

f[6]<=f[5];

f[7]<=f[6];

end

endmodule

`timescale 1ns / 1ps

module prelab04(rst, clk, f);

input rst;

input clk;

output reg [7:0] f;

always@ (posedge clk or posedge rst)

if (rst)

f <= 8'b01010101;

else

begin

f[0]<=f[7];

f[1]<=f[0];

f[2]<=f[1];

f[3]<=f[2];

f[4]<=f[3];

f[5]<=f[4];

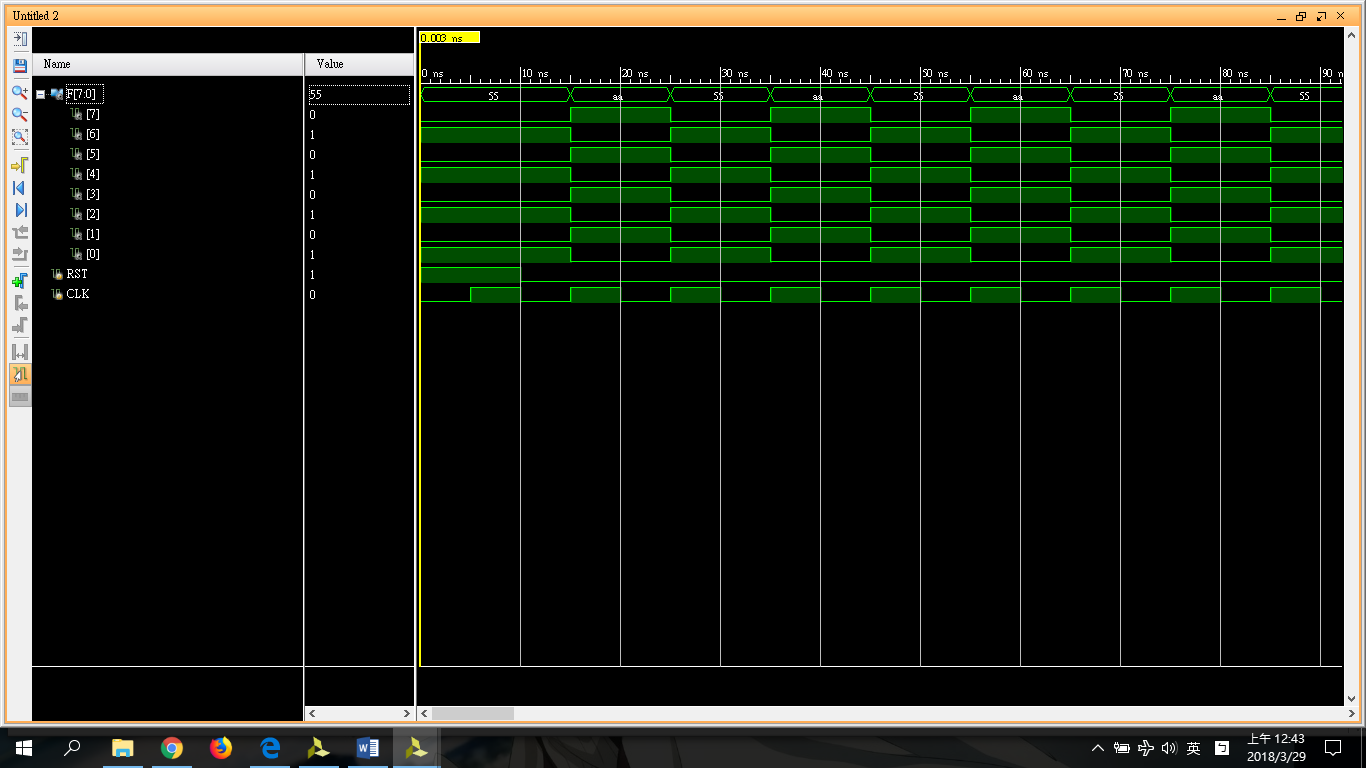
f[6]<=f[5];

f[7]<=f[6];

end

endmodule

(3) Verification

模擬結果如下：

**2. A ring counter that its initial value can be set randomly.**

**Design Specification**

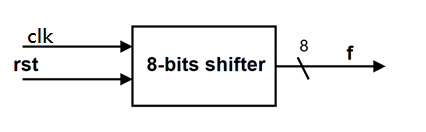
(1) I/O ports

input clk 共同時脈，100MHz，採正緣觸發。

input rst 開關，採負緣觸發。

output [7:0] f 七個DFF的輸出。

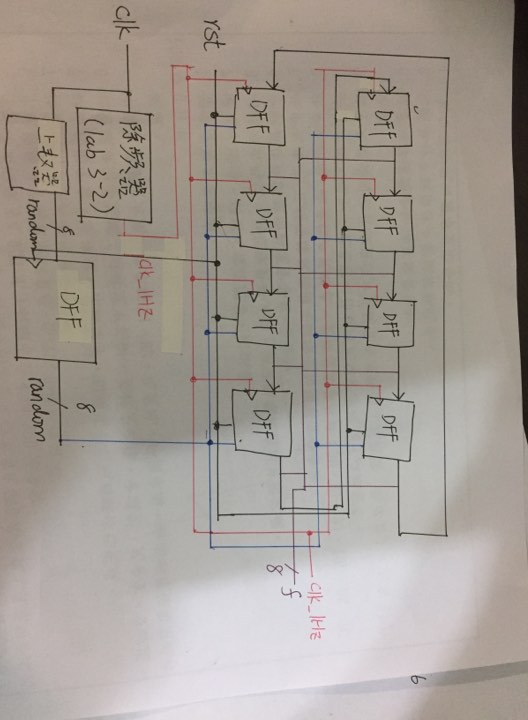
(2) Block Diagram



**Design Implementation**

　　我習慣把１Hz的Clock也接成Output方便觀察。

(1) Logic Diagram



(2) Verilog codes

`timescale 1ns / 1ps

module lab4\_2(rst, f, clk, clk\_1hz);

input clk;

input rst;

output wire clk\_1hz;

output reg [7:0] f;

reg [7:0] random;

reg [7:0] count;

reg [7:0] count\_temp;

lab3\_2 U0(.rst\_n(rst), .clk(clk), .signal(clk\_1hz));

always@(count)

count\_temp <= count + 1;

always@ (posedge clk or negedge rst)

if (~rst)

random <= count\_temp;

else

count <= count\_temp;

always@ (posedge clk\_1hz or negedge rst)

if (~rst)

f <= random;

else

begin

f[0]<=f[7];

f[1]<=f[0];

f[2]<=f[1];

f[3]<=f[2];

f[4]<=f[3];

f[5]<=f[4];

f[6]<=f[5];

f[7]<=f[6];

end

endmodule



上數器的本體，其輸出random可以視為必須再經過一個以rst為clock的DFF才能賦值給遞移暫存中的DFF們

剩下的部分就跟prelab一模一樣。

先打造一個上數器以製造隨機變數

**3. NTHUEE Displayer.**

**Design Specification**

(1) I/O ports

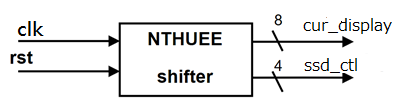
input clk; 共同時脈

input rst; 開關

output reg [7:0] cur\_display 單個七段顯示器所顯示的字母

output reg [3:0] ssd\_ctl 控制七段顯示器各別的亮暗

(2) Block Diagram

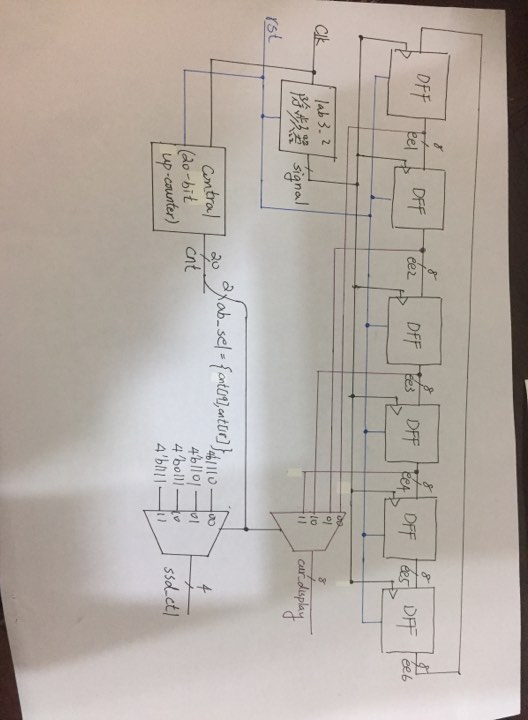


**Design Implementation**

　　這一題共需要三個模組。第一是控制遞移器與顯示字母的lab4\_3.v，第二是除頻器lab3\_2.v，第三是一個20bit的加法器contral.v。



(1) Logic Diagram



(2) Verilog codes

開關一旦打開，就開始遞移的程序

6個DFF的預設值

Lab3\_2是除頻器，signal為1Hz的clock

Contral是一個20bit的加法器，取其輸出末兩位為多工器的input

六個字母

`timescale 1ns / 1ps

`define SS\_N 8'b10101011

`define SS\_T 8'b10000111

`define SS\_H 8'b10001001

`define SS\_U 8'b11000001

`define SS\_E 8'b10000110

`define SS\_F 8'b10001110

module lab4\_3(clk, rst, ssd\_ctl, cur\_display);

input clk;

input rst;

output reg [7:0] cur\_display;

output reg [3:0] ssd\_ctl;

wire [19:0] cnt;

wire [1:0] ab\_sel;

wire signal;

reg [7:0] ee1, ee2, ee3, ee4, ee5, ee6;

lab3\_2 U0(.rst\_n(rst), .clk(clk), .signal(signal));

contral U1(.rst(rst), .clk(clk), .cnt(cnt));

assign ab\_sel = {cnt[19], cnt[18]};

always@ (posedge signal or negedge rst)

if (~rst)

begin

ee1 <= `SS\_N;

ee2 <= `SS\_T;

ee3 <= `SS\_H;

ee4 <= `SS\_U;

ee5 <= `SS\_E;

ee6 <= `SS\_E;

end

else

begin

ee6 <= ee1;

ee1 <= ee2;

ee2 <= ee3;

ee3 <= ee4;

ee4 <= ee5;

ee5 <= ee6;

end

整個程式中最關鍵的地方。ab\_sel這個控制信號同時決定「要亮什麼」和「現在誰該亮」。

always @(ab\_sel)

case (ab\_sel)

2'b00: cur\_display = ee1;

2'b01: cur\_display = ee2;

2'b10: cur\_display = ee3;

2'b11: cur\_display = ee4;

default: cur\_display = `SS\_F;

endcase

always @(ab\_sel)

case (ab\_sel)

2'b00: ssd\_ctl = 4'b1110;

2'b01: ssd\_ctl = 4'b1101;

2'b10: ssd\_ctl = 4'b1011;

2'b11: ssd\_ctl = 4'b0111;

default: ssd\_ctl = 4'b1111;

endcase

endmodule

附上contral.v的程式碼。它只是一個普通的up-counter。

`timescale 1ns / 1ps

module contral(clk, rst, cnt);

reg [19:0] cnt\_tmp;

output reg [19:0] cnt;

input clk;

input rst;

always @(posedge clk or negedge rst)

if (~rst)

cnt <= 19'b00;

else

cnt <= cnt\_tmp;

always @ (cnt)

cnt\_tmp <= cnt+1;

endmodule

**Discussion**

　　第二題最大的困難點就是要釐清怎麼找出隨機賦予初始值的方法。Verilog不像C語言具有產生亂數的函數庫。但經過思考發現這只是在玩文字遊戲。只要使用一套無法被掌握的規則產生的值就可以稱為亂數了。在這裡採用以100MHz為時脈的上數器的輸出來實作。由於頻率太快，將開關壓下時根本無法確定上數器的輸出，故可以視為隨機變數。

第三題的code中，有兩個以ab\_sel控制的always敘述。其上半部是控制「要亮什麼」。控制單一一個七段顯示器的字母；下半部是「現在誰該亮」。藉由一次只亮單一一個七段顯示器並且快速的切換的過程中達成視覺暫留。以開關剛打開的狀況為例，也就是重複「第一個亮n」「第二個亮t」「第三個亮H」「第四個亮U」直到signal反相，完成遞移後再重複「第一個亮t」「第二個亮H」「第三個亮U」「第四個亮E」……

　　第三題還有遇到一個麻煩。由於我沒寫lab2的bounus，所以也不知道要用20bits的上數器來使七段顯示器產生視覺暫留。於是我一開始只用2個bits，就造成七段顯示器上顯示很怪的圖形。事實上20bits的方法是問室友的，但我們討論不出為何2bits的就不行。

**Conclusion**

1. 遞移器即為一連串DFF的排列。寫Verilog可以不必像C一樣在遞移過程中需要另外另一個暫存器。

2. 要控制七段顯示器使之產生視覺暫留，需要一20bit的up-counter的最高位的兩個bit來控制。

**References**

上課的講義

《Verilog硬體描述語言數位電路》，鄭信源，儒林圖書公司(2016)。