Lab 4: Shift Registers

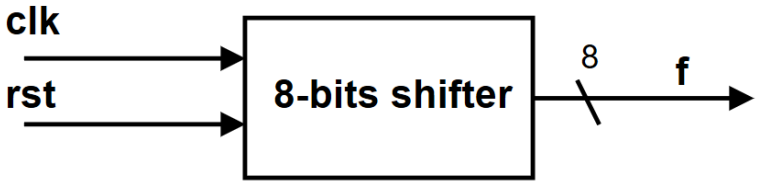
1. Pre-lab and implementation.

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

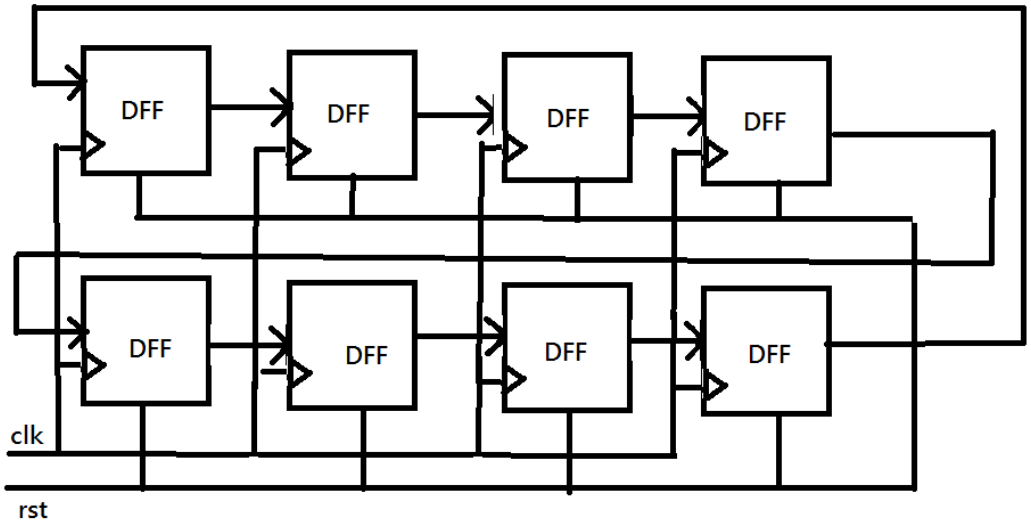
(2) Block Diagram



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 1 Hz 的訊號。



(2) Verilog codes

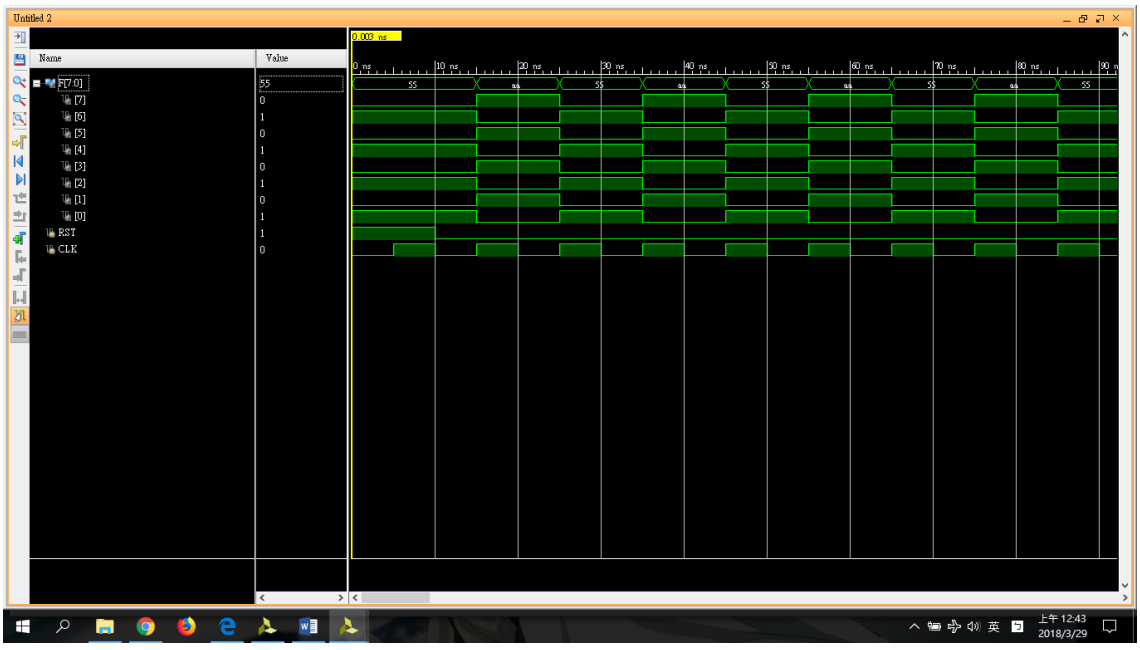
將 prelab(左側)與 lab4_1 的 code 比較一下。

```
`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
```

```
`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
```

(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

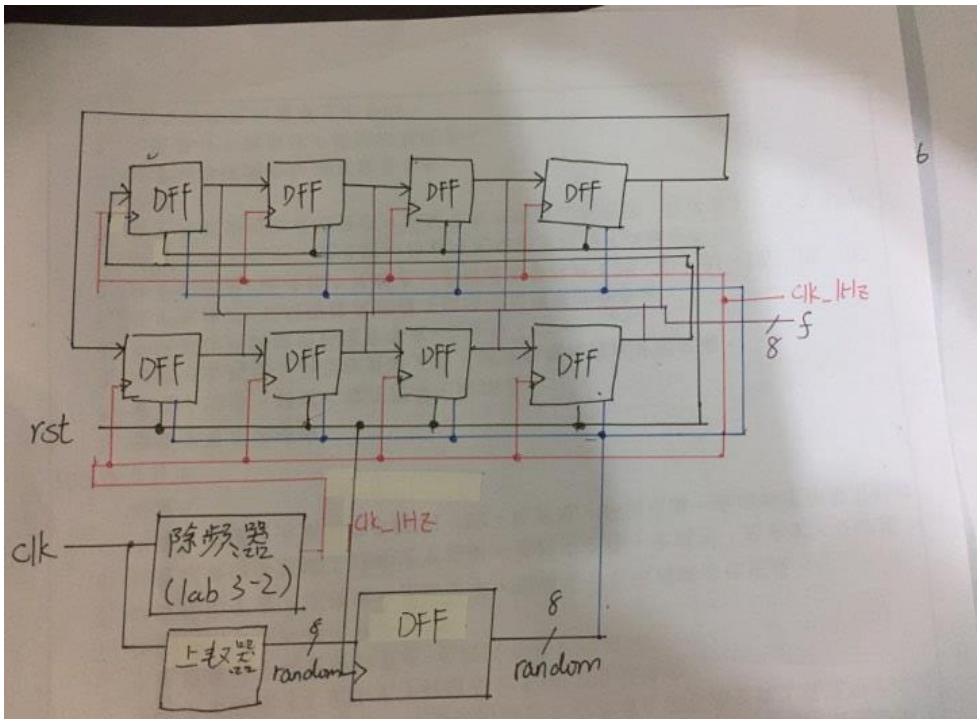
input clk　　共同時脈，100MHz，採正緣觸發。
input rst　　開關，採負緣觸發。
output [7:0] f　　七個 DFF 的輸出。

(2) Block Diagram

Design Implementation

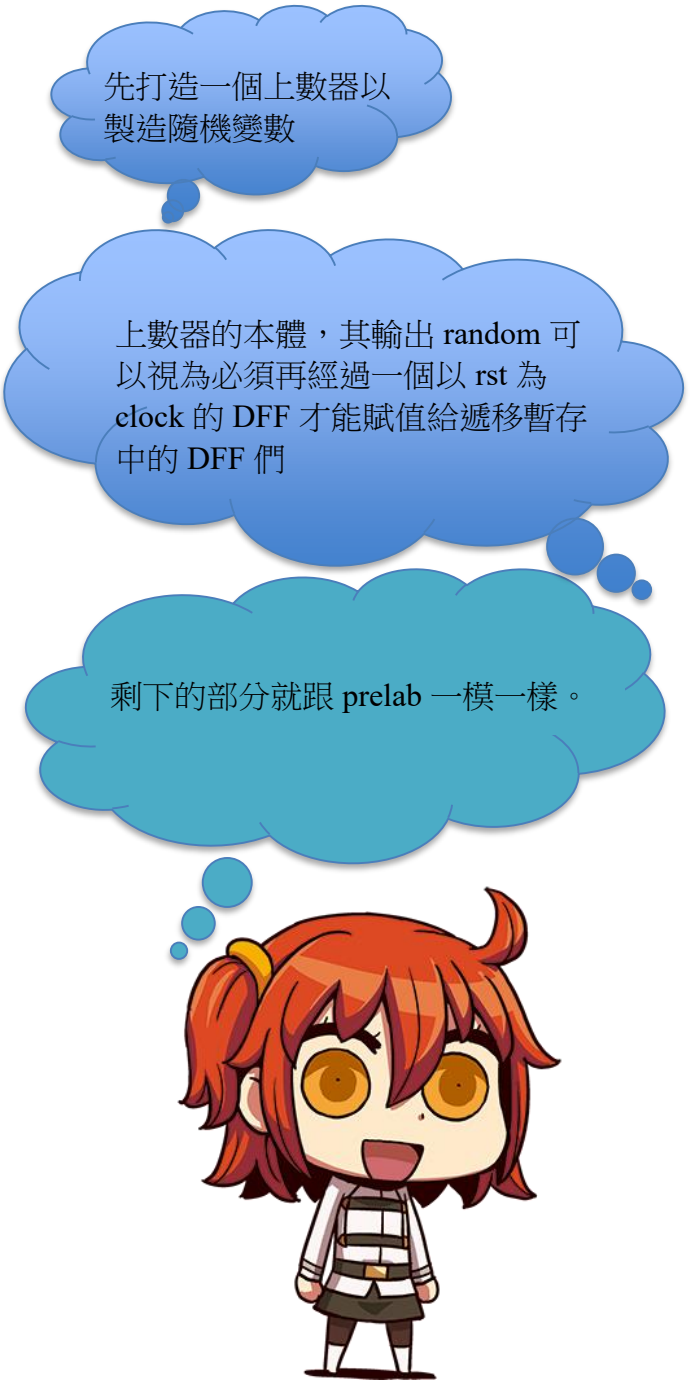
我習慣把 1 Hz 的 Clock 也接成 Output 方便觀察。

(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
```



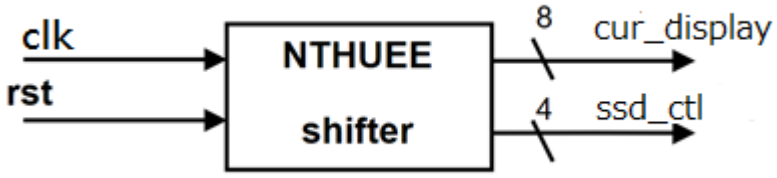
3. NTHUEE Displayer.

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

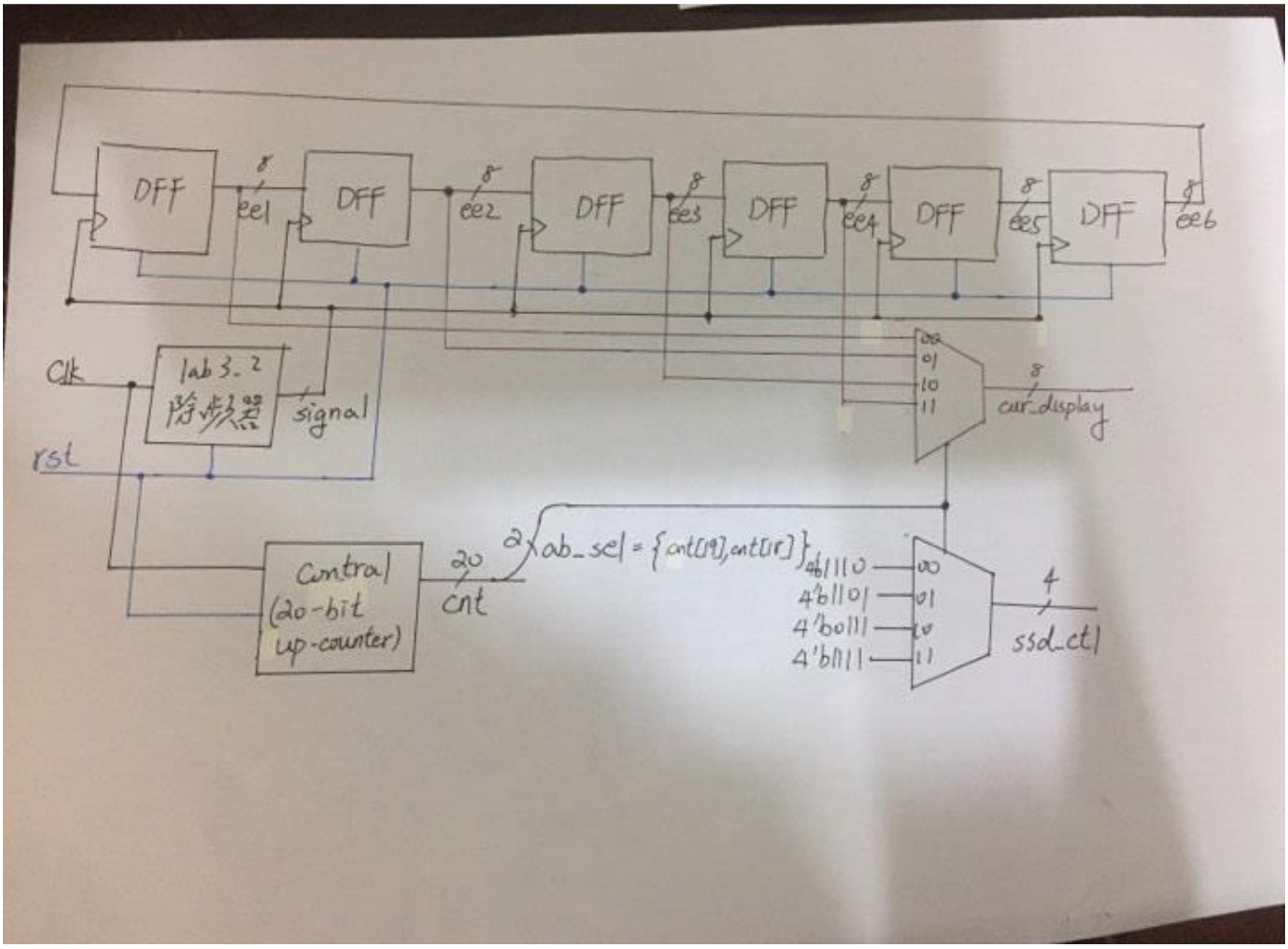
(2) Block Diagram



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

```
`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
```

六個字母

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

6 個 DFF 的預設值

開關一旦打開，就開始遞移的程序

```
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
```

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」。

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)。