NYCU – 2022 DCS 數位電路與系統 期中考

考試時間: 2022/04/07 15:30~16:20

姓名: 劉祐銃　　學號: 109511117

工作站帳號: dcs076

1. 語法觀念 (15%) +13

5 (1) 請解釋 blocking, non-blocking 的差異 (5%)

	等號	用法	用途
blocking	=	通常在 Combinational Logic 中	依次序執行
non-blocking	<=	通常在 Sequential Logic 中	同步執行

5 (2) 指出誰是 blocking, non-blocking (5%)

(a)

```
always@( posedge clk) begin
    q1  <= in ;
    q2  <= q1 ;
    out <= q2;
end
```

↳ non-blocking

(b)

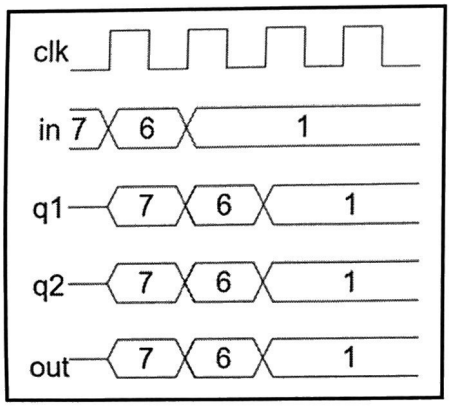
```
always@( posedge clk) begin
    q1  = in ;
    q2  = q1 ;
    out = q2;
end
```

↳ blocking

(因為要同時在觸發條件下一起更值)

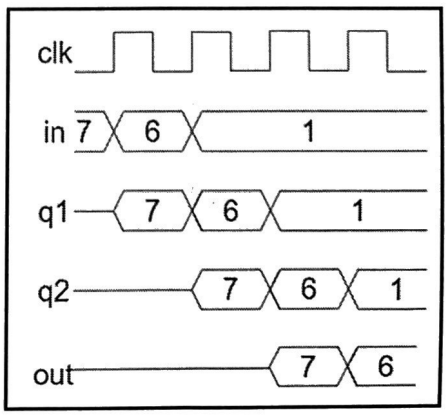
3 (3) 請問下方的哪個波型圖分別對應到(2)(a)或(2)(b)，並請解釋原因 (5%)

(a)



2(b) -1

(b)

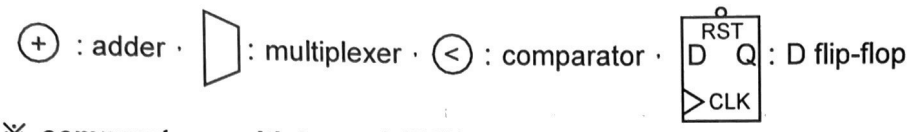


2(a) -1

2. Block diagram (14%)

14

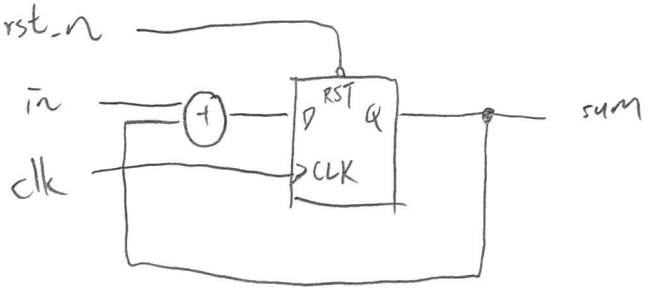
請使用 adder, multiplexer, comparator, D flip-flop 的圖例，
畫出以下兩個電路的 Block diagram。



※ comparator, multiplexer 之訊號順序請標示清楚。
※ 所有訊號不論 multi-bit bus 或 1-bit signal 都用一條線表示即可。

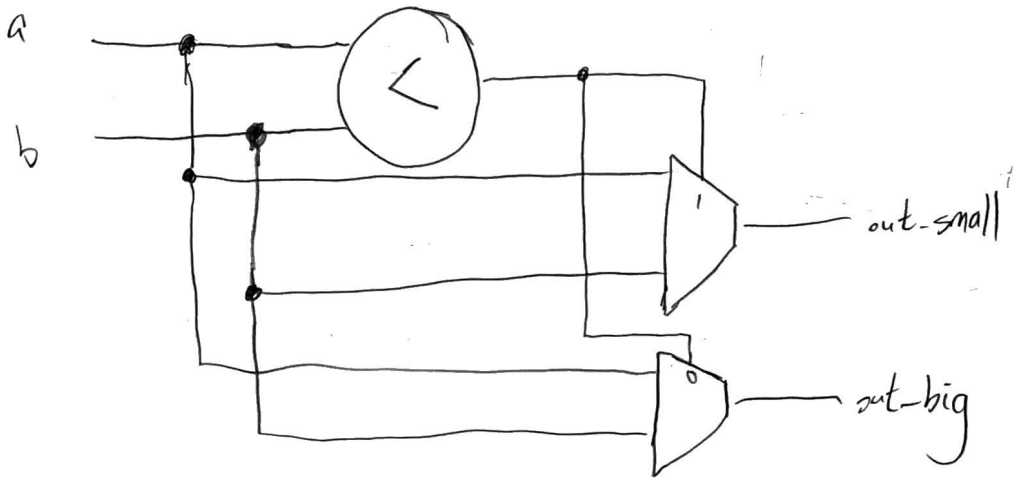
(1)

```
always@(posedge clk, negedge rst_n) begin
    if(~rst_n) begin
        sum <= 0;
    end
    else sum <= sum + in;
end
```



(2)

```
always_comb begin
    out_big   = (a>b?a:b);
    out_small = (a>b?b:a);
end
```



3. 時序觀念 (20%)　+18

(1) 請解釋 01_RTL、02_SYN 和 03_GATE 各自的用途，並解釋不同之處。(12%)

RTL：不考慮時間，只驗證邏輯的正確性，測試寫出的 module 是否功能性正確。

SYN：將時間納入考量，包含 delay 等，較接近實際狀況，也會大略地估算出合成面積。

GATE：更綜合的全面考量，尤其在初始化值的重要性增加，不能使得線出現 high-Z 或是 X 的情況，否則與實際電路不符。✗

+10

Difference ⇒

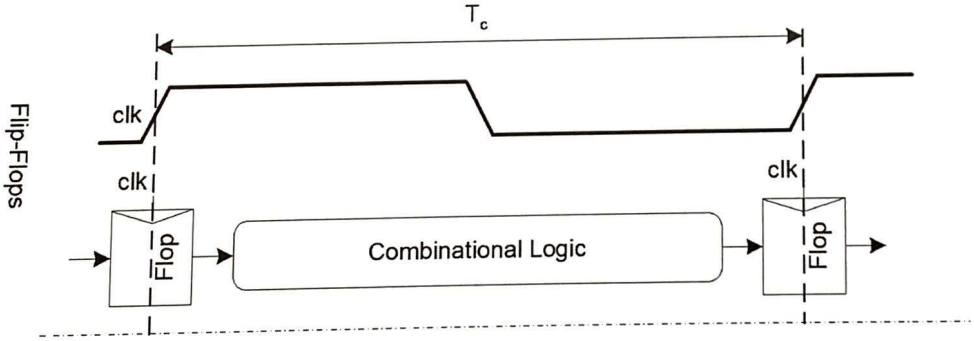
RTL	SYN	GATE
"理想" function正確性	考慮時間	考慮時間+運算合理性(不得空接或衝突)

(2) 如下圖，

假設 T_c 為 5ns，DFF 的 T_{setup} 為 1ns，combinational logic 需要 7ns 的時間，

假設 clk 訊號是理想的，沒有 clock skew、DFF 不需考慮 T_{cq}、T_{hold}，

請問會在 01、02、03 哪個階段先發生問題？原因是？(4%)　要如何解決？(4%)



+8

① 02 SYN 會產生問題

② 合成時會考慮 timing，而此設計超出了 cycle time，會有 setup time violation.

③ - 將 combinational logic 的所需時長降低 ⇒ 改變演算法或使用 pipeline 架構

- 將 cycle time 提高

4. FSM (14%) +14

一個名為曾帥的工程師正在開發一款自動咖啡沖煮機器人，當他完成 Finite state machine 時，被他的寵物貓 Tom 咬斷了電腦電源線導致一些資料損失。
請同學根據以下流程圖以及程式碼幫忙他恢復他的 FSM。

Ans: (1) f_need = 1

(2) (cnt = 28 & bean = 0)

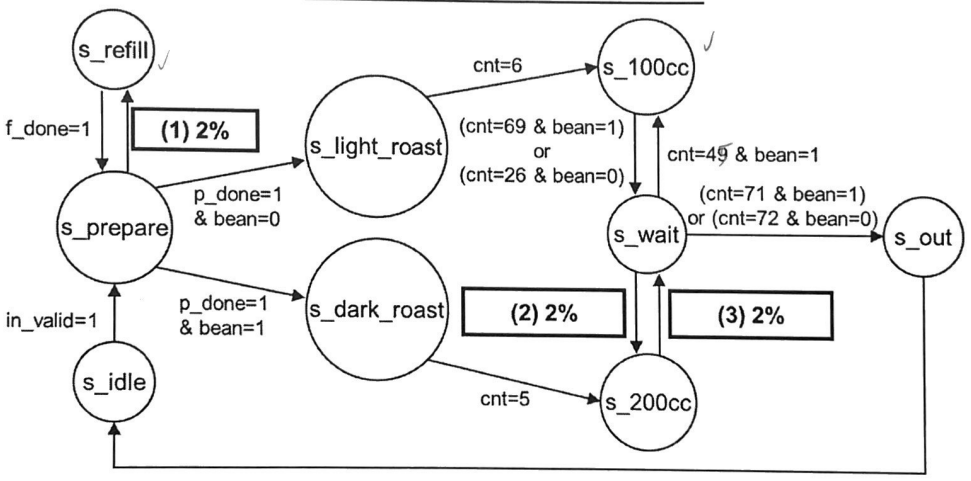
(3) (cnt = 45 & bean = 1) or (cnt = 68 & bean = 0)

(4) curr_state <= next_state ;

(5) (f_done) ? s_prepare : s_refill ;

(6) ((cnt == 68 && bean == 1) || (cnt == 26 && bean == 0)) ? s_wait : s_100cc ;

(7) s_out : next_state = s_idle ;




```
always_ff@(posedge clk or negedge rst_n) begin
    if (!rst_n)
        curr_state <= s_idle;
    else
        (4) 2%
end

always_comb begin
    case(curr_state)
        s_idle          : next_state = (in_valid)? s_prepare : s_idle;
        s_prepare       : next_state = (f_need)  ? s_refill :
                                       (p_done && !bean)? s_light_roast :
                                       (p_done &&  bean)? s_dark_roast : s_prepare;
        s_refill        : next_state = (5) 2%
        s_light_roast   : next_state = (cnt==6)? s_100cc : s_light_roast;
        s_dark_roast    : next_state = (cnt==5)? s_200cc : s_dark_roast;
        s_100cc         : next_state = (6) 2%
        s_200cc         : next_state = ((cnt==45 && bean==1)||(cnt==68 && bean==0))? s_wait : s_200cc;
        s_wait          : next_state = (cnt==45 && bean==1)? s_100cc :
                                       (cnt==28 && bean==0)? s_200cc :
                                       ((cnt==71 && bean==1)||(cnt==72 && bean==0))? s_out : s_wait;
        (7) 2%
        default         : next_state = curr_state;
    endcase
end
```

5. Debug (21%)

找出下方 SystemVerilog RTL code 的語法錯誤、不可合成以及會產生 latch 的地方，並將其圈起來在一旁寫出正確的語法，如果是產生 latch 的地方就寫「latch」即可。

提示：總共 7 個錯誤，相同的語法錯誤算 1 個。(各 3%)

※ 把正確的改成錯誤的倒扣 1 分，把正確的改成正確的不扣分。

範例：第 27 行 elif 更正 else if

```
1  module BUG(
2  // input signals
3  clk,
4  rst_n,
5  mode,
6  in_valid,
7  in_data_0,
8  in_data_1,
9  in_data_2,
10 // output signals
11 result_0,
12 result_1,
13 result_2
14 );
15
16 input clk, rst_n, in_valid;
17 input [1:0] mode;
18 input [3:0] in_data_0, in_data_1, in_data_2;
19 output logic [5:0] result_0, result_1, result_2;
20
21 logic [2:0] counter_reg, [2:0] counter_nxt;
22
23 assign result_0 <= in_data_0 << 2;
24
25 always_comb begin
26     if (mode == 0) result_1 = in_data_1 - in_data_0;
27     elif (mode == 1) result_1 = in_data_1 - 1;
28     else result_1 = in_data_1 + 1;
29 end
30
31 always_comb begin
32     case (mode)
33     2b00: result_2 = in_data_2 + in_data_1;
34     2b01: result_2 = in_data_2 - in_data_1;
35     2b10; result_2 = counter_reg;
36 end
37
38 always_comb begin
39     counter_nxt = (in_valid) ? ((counter_reg == 7) ? 0 : (counter_reg + 1)) : counter_reg;
40     result_2 = counter_reg + 1;
41 end
42
43 always_ff @(posedge clk or negedge rst_n) begin
44     if (!rst_n) counter_reg <= 0;
45     else begin
46         counter_reg = 0;
47         if (in_valid) counter_reg <= counter_nxt;
48     end
49 end
50
51 endmodule
52
```

範例:
else if

6. 波型圖 (16%) ~~+12~~ +10

1-bit signal 與 multi-bit bus 的畫法請參照下圖範例：

請依照以下的 Design 與 clk、rst_n、in 輸入波型，
畫出 a、b、c、d 訊號的輸出波型圖 (各 4%)。

Design

```
1  logic [2:0] in;
2  logic [2:0] a, b;
3  logic c, d;
4
5  always_ff@(posedge clk or negedge rst_n) begin
6      if (!rst_n)
7          a <= 0;
8      else
9          a <= b;
10 end
11
12 always_comb begin
13     b = a + in;
14     c = in[2];
15     d = c;
16 end
```

Waveform