1. **< Code Debugging and Simulation > (10pts)**

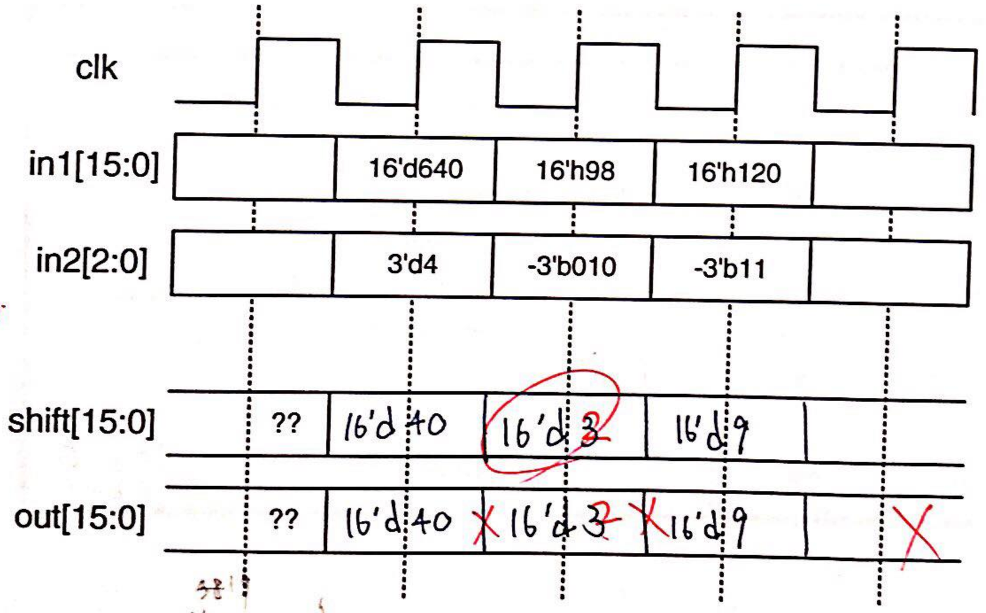
A. (5pts) Identify syntax error, correct, and explain: 0.5pts for each. Identify inappropriate code (or semantics error), correct, and explain: 0.5pts for each.

-1.5

```
module 2%shifter (out,clk, rst, in1, in2) ; (沒分号)
       └不能數字開頭
input clk, rst;
input [15:0] in1;
input [2:0] in2;
                reg
output [15:0] out;
wire
reg[15:0] shift;
assign shift = in1 >> in2;
/* variable shifter */

                                   or
always @(posedge clk and posedge rst) begin
if(!rst) out  =  16'd0;
else   out  =  shift;
end           <= (non-blocking)
endmodule (沒 endmodule)
```

B. (5pts) Finish the waveform below based on the circuit in part A. Note that you should use the decimal number representation to answer (such as 16'd0). Use "xx" to indicate values that cannot be determined from the information given. (In this period, signal rst is always 0)

-3

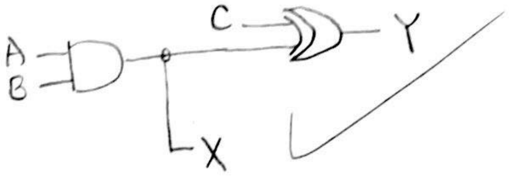


010
144
640/2^4 = 40
8+16×9
152/6
64
101
110

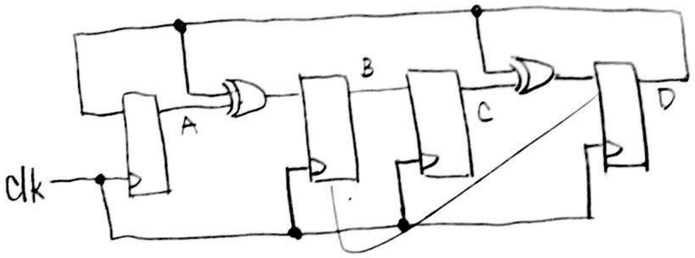
2. < Logic Synthesis + Blocking & Non-Blocking > (10pts)

In the following table, the left column show some pieces of Verilog RTL code. Please draw the corresponding circuits in the right column. You can use AND, OR, NAND, NOR, XOR, XNOR, NOT, MUX, D Flip-Flop, Latch in the circuit diagram.

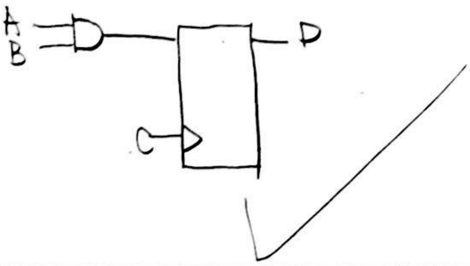
(a) Verilog Code (2pts)	Circuit Diagram

always @(*) begin
    X = A&B;
    Y = X^C;
end
	

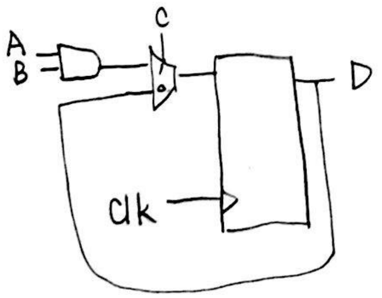
(b) Verilog Code (2pts)	Circuit Diagram

always @(posedge clk)
begin
    A <= D;
    B <= A ^ D;
    C <= B;
    D <= C ^ D;
end
	

(c) Verilog Code (3pts)	Circuit Diagram

always@(A or B or C)
begin
  if (C)
    D = A & B;
end
	

(d) Verilog Code (3pts)	Circuit Diagram

always@(posedge clk)
begin
  if (C)
    D <= A & B;
end
	

3. < Finite State Machine and Simulation > (10pts)

Given below is a Finite-State-Machine (FSM).

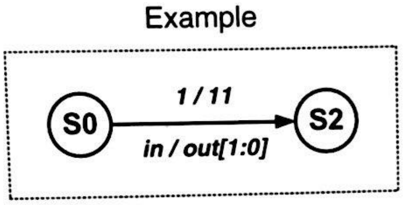
```
module FSM    (clk, rst, in, out_r);
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

input    clk, rst, in;
output [1:0] out_r;

reg [1:0] out_r, out;
reg [1:0] current_state, next_state;
// Next State Logic
always@(*) begin
    case(current_state)
    S0: next_state = (in == 1'b0)? S0 : S2;
    S1: next_state = (in == 1'b0)? S0 : S2;
    S2: next_state = (in == 1'b0)? S1 : S3;
    S3: next_state = (in == 1'b0)? S1 : S3;
    default: next_state = 2'b00;
    endcase
end
// Current State Memory & Output Register
always@(posedge clk or posedge rst)
begin
    if(rst) begin
        current_state <= 0;
        out_r <= 0;
    end
    else begin
        current_state <= next_state;
        out_r <= out;
    end
end
// Output Logic
always@(*) begin
    out[1] = in ^ current_state[0];
    out[0] = in ^ current_state[0] ^ current_state[1];
end

endmodule
```

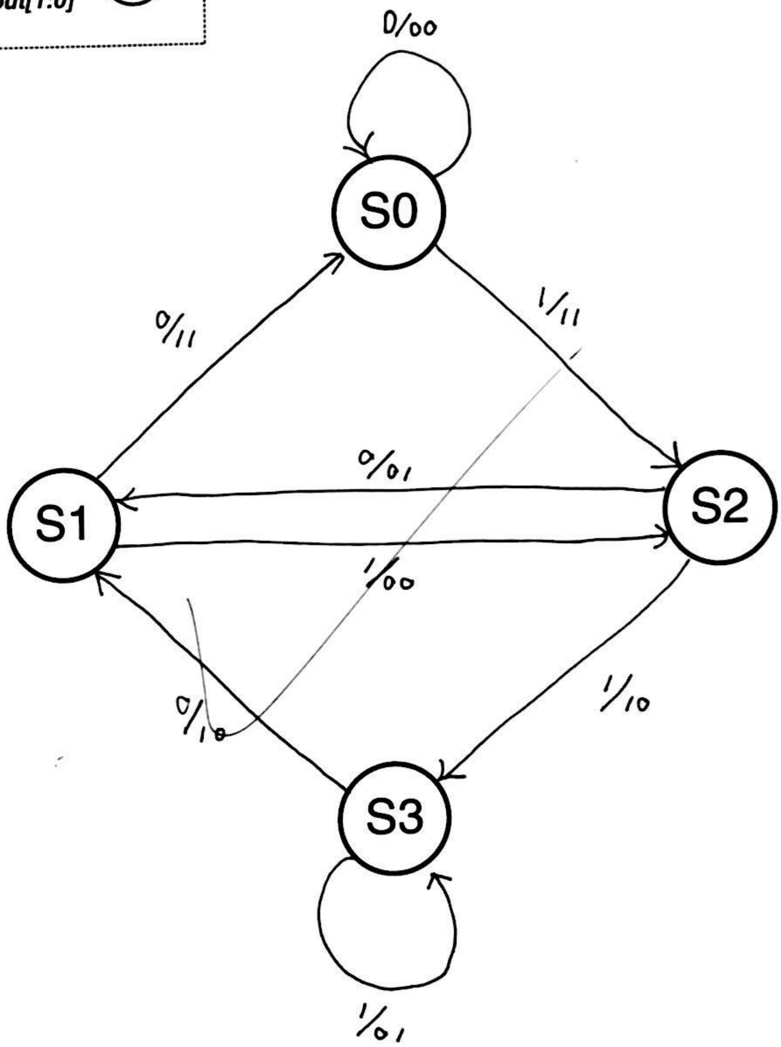
0×1 + 1×0 = 0

0× ×y

C xor

(a) **(5pts)** Please draw a state transition graph below for this FSM.





(b) (5pts) We have put this module in our testbench as a Design-Under-Test (DUT). After the simulation, the command window has printed response from monitor. Please finish the output results below based on this FSM and given information.

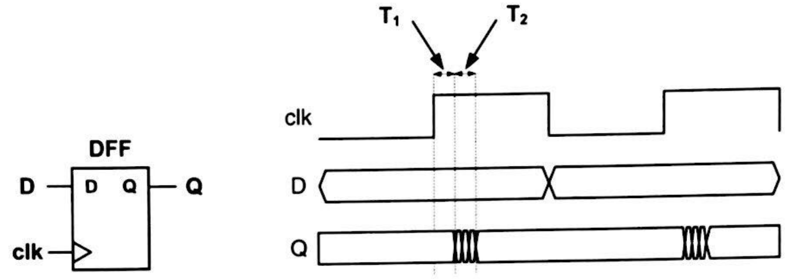
```
`timescale 1ns/1ns

module testbench;
reg    clk, rst;
reg    in;
wire [1:0] out;

FSM DUT(.clk(clk), .rst(rst), .in(in), .out_r(out));

// APPLY STIMULUS
$monitor("%t %b %b %b %b", $time, clk, rst, in, out);
endmodule
```

Monitor Output Response:

Time	clk	reset	in	out	
0	0	0	0	xx	
1	1	1	0	00	S0
2	0	0	1	00	S0
3	1	0	1	1 1	S1
4	0	0	1	1 1	S2
5	1	0	1	1 0	S3
6	0	0	1	1 0	S3
7	1	0	1	0 1	S3
8	0	0	0	0 1	S3
9	1	0	0	1 0	S1
10	0	0	1	1 0	S1
11	1	0	1	0 0	S2
12	0	0	0	0 0	S2
13	1	0	0	0 1	S1
14	0	0	0	0 1	S1
15	1	0	0	1 1	S0
16	0	0	0	1 1	S0

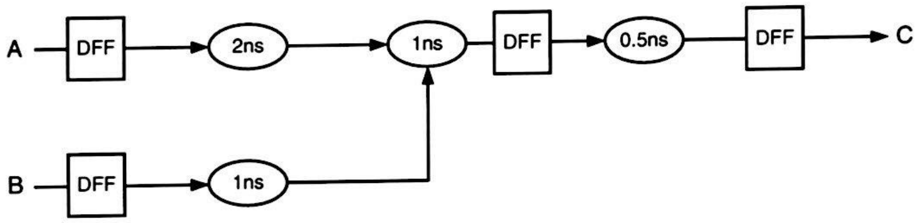
4. < Important Timing Parameters > (15pts)

Suppose that the timing characteristics of the flip-flops in the circuit are the same. Their timing diagrams and parameters can be described as follows:



T_1=0.2ns T_2= 0.3ns

The circuit below operates at the clock frequency of **250MHz.** Suppose that the rise, fall, and turn-off delays for each combinational element are the same.



(a) (3pts) Write the timing inequality for setup time and hold time.

$T_{setup} < T_{cycle} - T_{cq} - T_{logic}$ $\nearrow$ $\underline{T_{setup} < 0.5\,ns}$

$T_{cycle} = \frac{1000}{250}\,ns = 4ns$ $T_{cq} = T_1 + T_2 = 0.5ns$ $T_{logic} = 2 + 1 = 3ns$

$T_{hold} < T_{cq.cd} + T_{logic.cd}$ $\nearrow$ $\underline{T_{hold} < 0.7\,ns}$

$T_{cq.cd} = T_1 = 0.2ns.$ $T_{logic.cd} = 0.5ns$

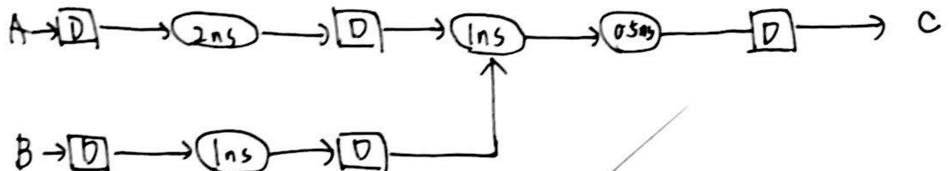
(b) (2pts) If **T_{setup} (Setup Time) = 1ns T_{hold} (Hold Time) = 1ns**

Are there setup time and hold time violations in this circuit? Use the timing inequalities in (a) for setup/hold time at the clock frequency to check them.

T_{setup} : 1ns > 0.5ns → setup time violation

T_{hold} : 1ns > 0.7ns → hold time violation

(c) (5pts) Followed by part (b), if there are setup/hold time violations in this circuit, how to perform "retiming" to solve these issues? Suppose that all of combinational elements cannot be further separated. Please draw your circuit and write the timing inequalities of the modified circuits after retiming.

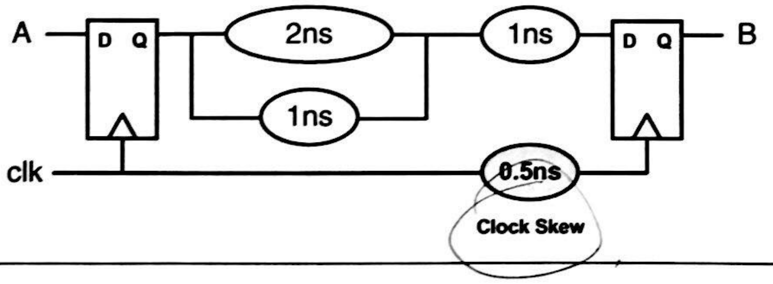


$T_{setup} < 4 - 0.5 - 2 = 1.5$ ns ($T_{logic} = 2$ns)

$T_{hold} < 0.2 + 1 = 1.2$ ns ($T_{logic.cd} = 1$ns)

(d) (5pts) If there is clock skew in this circuit, as shown in bellow. Please write the timing inequality for setup time and hold time at the clock frequency of **200MHz without** any timing violation.

$\frac{1000}{200} = 5$ ns



-5

$T_{setup} < T_{cycle} - T_{cq} - T_{logic}$

$T_{cycle} = \frac{1000}{200} = 5$ns $T_{cq} = 0.5$ns $T_{logic} = 2 + 1 = 3$ ns

$\rightarrow \underline{T_{setup} < 1.5 \text{ ns}}$

$T_{hold} < T_{cq.cd.} + T_{logic.cd}$

$T_{cq.cd} = 0.5$ns . $T_{logic.cd} = 2$ ns

$\rightarrow \underline{T_{hold} < 2.5 \text{ ns}}$

5. **< Synthesis Issues > (30pts)**

A. < Important files related to Design Compiler >

Please explain the meaning of the following terminologies and where to use them:

16

(a) (2pts) Technology library (e.g: slow.db/fast.db)

(b) (2pts) Standard Delay File (e.g: CHIP_syn.sdf)

(c) (2pts) tsmc13.v

a. conditions used in DC.

b. Timing information of synthesized gate-level code.
output of synthesis.

c. 跑 gate-level 模擬時的製程參數.

B. < Synopsys Design Constraints File(SDC) >

Please explain the meaning of the following command and why we use them in Design Compiler:

(a) (2pts) *set_dont_touch_network* *[get_clocks_clk]*
set_ideal_network *[get_ports clk]*

(b) (2pts) *set_clock_uncertainty 0.1 [get_clocks clk]*

a. 在 multiple module instance 時用
別改變設計.

-2

b.

-2

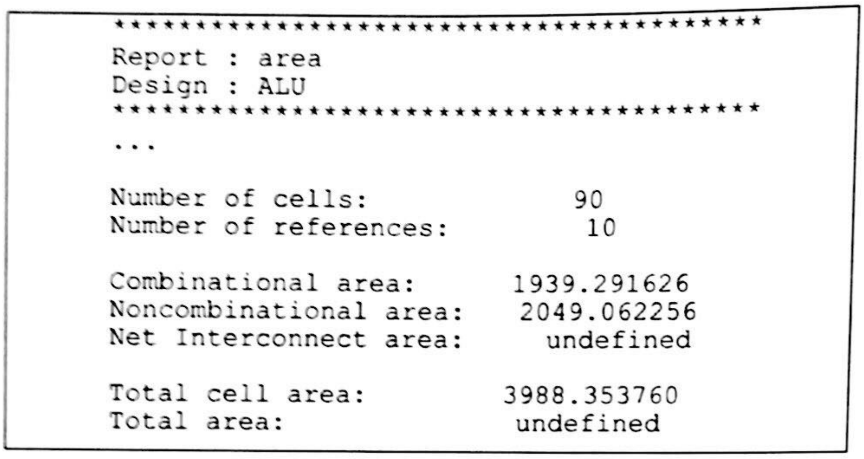
C. (3pts) < STA & Post-sim >

If we specify the clock to be 5.0ns during synthesis, the timing report shows that the constraints has been **met**. However, the gate-level simulation passed at 4.0ns with one set of test data. Is this possible? Why or why not?

有可能. 因為 synthesis 時是看 critical path, constraints met 代表 critical path 過了 5ns. 所以在模擬時. 若是跑 other path 有可能到 4ns.

D. < Area Report >

The following figure is the area report after synthesis.



```
****************************************
Report : area
Design : ALU
****************************************
...

Number of cells:                90
Number of references:            10

Combinational area:        1939.291626
Noncombinational area:      2049.062256
Net Interconnect area:        undefined

Total cell area:           3988.353760
Total area:                 undefined
```

(a) **(2pts)** It sometimes shows "undefined" in total area. Please explain it and describe how to fix it.

(b) Followed by part(a),

- I. **(2pts)** In cell-based IC design flow, we will focus on total cell area instead of total area, please explain why.
- II. **(4pts)** The total cell area will be underestimated in this situation. Please briefly explain why.

(c) **(3pts)** With the same RTL code, if we reduce the clock cycle, what part in the report will increase? Please briefly explain why.

a. 沒吃到製程參數. 所以沒有 wire ... 的面積.(tsmc13.v)

−2 加入參數

b. 1. 線不佔面積(可以重疊)

−6 II.

C. Combiational area 會 increase.

因為為了達到 clock cycle. DC會用更大的

logic gate.

E. < External IP issue >

If there's a memory module in DUT, and we generate several files from Memory Generator. Such as **rom_1024x4_t13_slow_syn.db, rom_1024x4_t13.v**.

(a) **(2pts)** Please explain what's the purpose of these files and what will happen if we don't have these files.

① rom cell 的 worst case，沒辦法模擬出完整結果

②.rom 在 T13 的製程參數.無法模擬.

(b) **(2pts)** Please modify Design Compiler setting file .synopsys_dc.setup as shown below. **(JUST NEED TO POINT OUT WHERE TO MODIFY)**

```
set search_path "Your_path/CBDK_TSMC013_Arm/CIC/SynopsysDC $search_path "
set target_library "slow.db fast.db"
set link_library    " * $target_library dw_foundation.db"
set symbol library "tsmc13.sdb generic.sdb"
set synthetic_library "dw_foundation.sldb"
```

-2

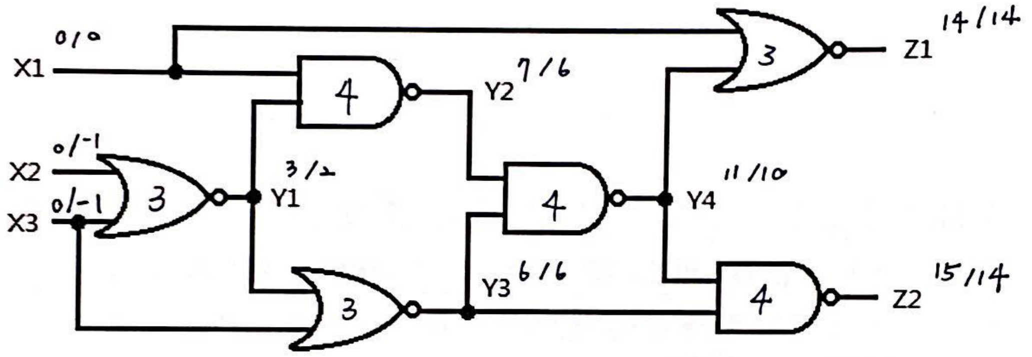
(c) **(2pts)** Should we synthesis **rom_1024x4_t13.v** with DUT.v ? Please explain why.

no. 記憶体有專門在設計的.自己合不能達到最好的效果.

6. **< Timing Analysis > (10pts)**

Calculate the arrival time, required time, and slack at each gate output, and find a critical path from primary input to primary output. Assume the delays of NAND gates and NOR gates are **4ns** and **3ns**, respectively. The arrival time at primary inputs is **0ns** and the required time at primary outputs is **14ns**.

9



X1: Arrival 0 ; Required 0 ; Slack 0

X2: Arrival 0 ; Required -1 ; Slack -1

X3: Arrival 0 ; Required -1 ; Slack -1

Y1: Arrival 3 ; Required 2 ; Slack -1

Y2: Arrival 7 ; Required 6 ; Slack -1

Y3: Arrival 6 ; Required 6 ; Slack 0

Y4: Arrival 11 ; Required 10 ; Slack -1

Z1: Arrival 14 ; Required 14 ; Slack 0

Z2: Arrival 15 ; Required 14 ; Slack -1

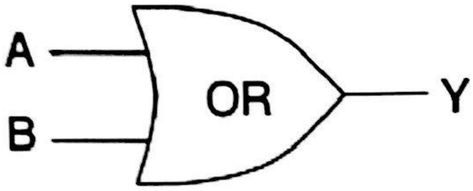
Critical path:

X_2
X_3 > $Y_1 - Y_2 - Y_4 - Z_2$

7. **< Design for Testability > (15pts)**

(a) **(5pts)** Given one OR gate for your reference below. Answer the following questions.

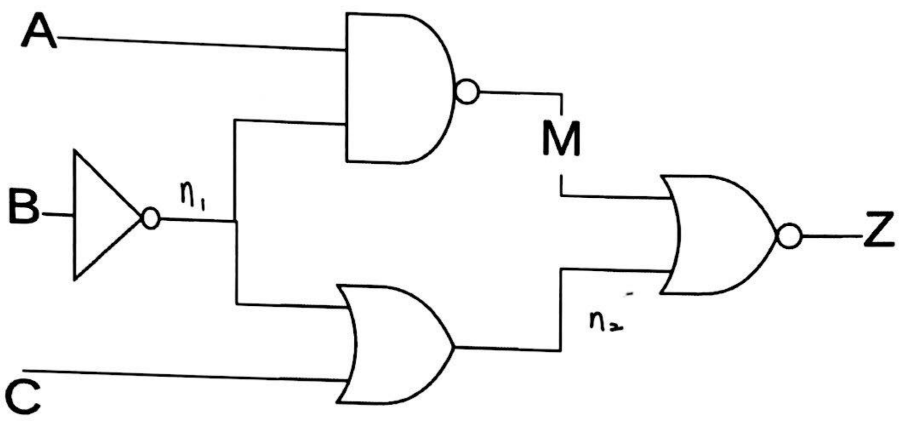
10



Complete the single stuck-at-fault (SSF) table of two-input OR gate below for the output value with SSF. By *signal*/*logic_value* we mean single stuck-at-*logic_value* fault on *signal*, e.g. *A*/0 means stuck-at-0 fault on signal *A*. **Please also mark the output value at Y differing from fault free one with "*" character (such as "1*").**

Input		Fault-free Output	Output Value on Y with SSF					
A	**B**	**Y**	**A/0**	**A/1**	**B/0**	**B/1**	**Y/0**	**Y/1**
0	**0**	0	0	1*	0	1*	0	1*
0	**1**	1	1	1	0*	1	0*	1
1	**0**	1	0*	1	1	1	0*	1
1	**1**	1	1	1	1	1	0*	1

(b) (10pts) Given the circuit below, please generate one test pattern to detect the faults given as below. You may use D-Algorithm to generate the pattern. Please write down your pattern in the form of {abc}, e.g. {01X}, where X is don't-care bit.



1. (5pts) Z s-a-1

Activation : $Z = 0 \rightarrow \{M, n_2\} = \overset{①}{\{1, 0\}} . \overset{②}{\{0 . 1\}} . \overset{③}{\{1 . 1\}}$

① $\{A . n_1, C\} = \{X . 0 . 0\} \rightarrow \{A . B . C\} = \underline{\underline{\{X . 1 . 0\}}}$

② $\{A . n_1 . C\} = \{1 . 1 . X\} \rightarrow \{A . B . C\} = \underline{\underline{\{1 . 0 . X\}}}$

③ $\{A . n_1 . C\} = \{0 . 1 . 1\} \rightarrow \{A . B . C\} = \underline{\underline{\{0 . 0 . 1\}}}$ #

2. (5pts) M s-a-1

Activation : $M = 0 \rightarrow \{A . n_1\} = \{1 . 1\} \rightarrow \{A . B\} = \{1 . 0\}$

propagation : $n_2 = 0 \rightarrow C = 0$

$\Rightarrow \{A . B . C\} = \{1 . 0 . 0\}$ #

DUT

```
module DUT( clk, rst, in_en, A, B, O_ready, O);
   input            clk;
   input            rst;
   input            in_en;
   input   [1:0]    A;
   input   [1:0]    B;
   output           O_ready;
   output [3:0]     O;

   reg          O_ready, O_ready_nxt;
   reg [3:0] O;
   reg [3:0] A_sqr, A_sqr_nxt;
   reg [3:0] B_sqr, B_sqr_nxt;
always@(*) begin
   A_sqr_nxt = A*A;
   B_sqr_nxt = B*B;

   if(in_en)
      O_ready_nxt = 1'b1;
   else
      O_ready_nxt = 1'b0;

     O = A_sqr + B_sqr;
end

always@(posedge clk or posedge rst) begin
   if(rst) begin
      A_sqr <= 2'd0;
      B_sqr <= 2'd0;
      O_ready <= 1'b0;
   end
   else begin
      A_sqr <= A_sqr_nxt;
      B_sqr <= B_sqr_nxt;
      O_ready <= O_ready_nxt;
   end
end
endmodule
```

```
initial begin
   clk              = 1'b0;
   rst              = 1'b0;
   in_en            = 1'b0;
   exp_num          = 0;
   err              = 0;
   over             = 0;
end

always #(`CYCLE/2) clk = ~clk;

// data input
initial begin
   @(negedge clk) rst = 1'b1;
   #(`CYCLE);       rst = 1'b0;

   @(negedge clk) i=0;
   while (i <= N_PAT) begin
      in_en = 1'b1;
      A        = in_mem[i][3:2];
      B        = in_mem[i][1:0];
      i = i +1;
      @(negedge clk);
   end
end

// result compare
always @(negedge clk) begin
   O_exp = exp_mem[exp_num];
   if(O_ready) begin
      if(O != O_exp) begin
         $display("ERROR at %5d:O %4h !=O_exp %4h " ,exp_num, O, O_exp);
         err = err + 1;
      end
      exp_num = exp_num + 1;
   end
   if(exp_num == N_EXP) over = 1;
end
```

```
initial begin
  #(`CYCLE * `End_CYCLE);
  $display("-----------------------------------------------------\n");
  $display("Error!!! Somethings' wrong with your code ...!\n");
  $display("-------------------------FAIL------------------------\n");
  $display("-----------------------------------------------------\n");
  $finish;
end

initial begin
  @(posedge over)
  if((over) && (exp_num!='d0)) begin
    $display("-----------------------------------------------------\n");
    if (err == 0)   begin
      $display("All data have been generated successfully!\n");
      $display("-------------------------PASS-------------------------------\n");
    end
    else begin
      $display("There are %d errors!\n", err);
      $display("-----------------------------------------------------\n");
    end
  end
  #(`CYCLE/2); $finish;
end

endmodule
```

Synopsys Design Constraints (SDC)

```
set cycle    2.0
create_clock -name clk -period $cycle [get_ports clk]
set_fix_hold                                  [get_clocks clk]
set_dont_touch_network                        [get_clocks clk]
set_ideal_network                             [get_ports clk]
set_clock_uncertainty                   0.1   [get_clocks clk]
set_clock_latency                       0.5   [get_clocks clk]

set_max_fanout 6 [all_inputs]

set_operating_conditions -min_library fast -min fast -max_library slow -max slow
set_drive          1        [all_inputs]
set_load           1        [all_outputs]
set_input_delay   0.1 -clock clk [remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 0.1 -clock clk [all_outputs]
set_wire_load_model -name tsmc13_wl10 -library slow
```

(a) **(5pts)** Please explain what message might show in terminal and why?

set up violation.

+3 : input delay & output delay are too

(b) **(5pts)** If we can modify the SDC file, please explain what's wrong in it and how to fix it.

+5 set-input-delay $cycle/2

set-output-delay $cycle/2

(c) **(5pts)** If we can **only** modify RTL code in DUT module, please describe what's wrong and how to fix it. **(YOU DON'T NEED TO WRITE MODIFIED CODE)**

+0

Synopsys Design Constraints (SDC)

```
set cycle    2.0
create_clock -name clk -period $cycle [get_ports clk]
set_fix_hold                                  [get_clocks clk]
set_dont_touch_network                        [get_clocks clk]
set_ideal_network                             [get_ports clk]
set_clock_uncertainty                   0.1   [get_clocks clk]
set_clock_latency                       0.5   [get_clocks clk]

set_max_fanout 6 [all_inputs]

set_operating_conditions -min_library fast -min fast -max_library slow -max slow
set_drive           1        [all_inputs]
set_load            1        [all_outputs]
set_input_delay   0.1 -clock clk [remove_from_collection [all_inputs] [get_ports clk]]
set_output_delay 0.1 -clock clk [all_outputs]
set_wire_load_model -name tsmc13_wl10 -library slow
```

(a) (5pts) Please explain what message might show in terminal and why?

+1

Error!!! Something's wrong with your code ... !

- - - - - FAIL - - - -

(b) (5pts) If we can modify the SDC file, please explain what's wrong in it and how to fix it.

把 clock cycle constraint 調高 (調鬆)

+1

(c) (5pts) If we can **only** modify RTL code in DUT module, please describe what's wrong and how to fix it. (**YOU DON'T NEED TO WRITE MODIFIED CODE**)

Bonus!! < STA & Post-sim inconsistent issue > (15pts)

The following is the RTL code from testbench (tb) and Design Under Test (DUT) module. The RTL simulation with those files already passed. However, if we use the SDC (Synopsys Design Constraints) file shown below, the timing report shows that the constraints is **met**, but the post-sim will fail.

Testbench (tb)

```
`timescale 1ns/10ps
`define CYCLE 2.0
`define SDFFILE "./DUT_syn.sdf"
`define End_CYCLE 100

`define PAT "./pattern.dat"
`define EXP "./golden.dat"

module tb;

parameter N_EXP = 10;
parameter N_PAT = N_EXP;

integer        i, exp_num, err;
reg            over;

reg    [3:0]   in_mem   [0:N_PAT-1];
reg    [3:0]   exp_mem [0:N_EXP-1];

reg            clk;
reg            rst;
reg            in_en;
reg    [1:0]   A, B;
wire           O_ready;
wire   [3:0]   O;
reg    [3:0]   O_exp;

DUT DUT( .clk(clk), .rst(rst), .in_en(in_en), .A(A), .B(B), .O_ready(O_ready), .O(O));

initial $sdf_annotate(`SDFFILE, DUT);

initial $readmemh(`PAT, in_mem);
initial $readmemh(`EXP, exp_mem);
```