Computer-Aided VLSI System Design
Midterm Examination
2018. 5. 8

Name _____

Student ID _____

Instructions

This is a ***CLOSED*** book exam. The exam is to be completed in **160** minutes. If you need scratch paper, just use the blank parts of these pages; show all of your work on these pages. Before you start writing, please check if you have all 15 pages of the exam.

Score Board (to be filled by TAs)

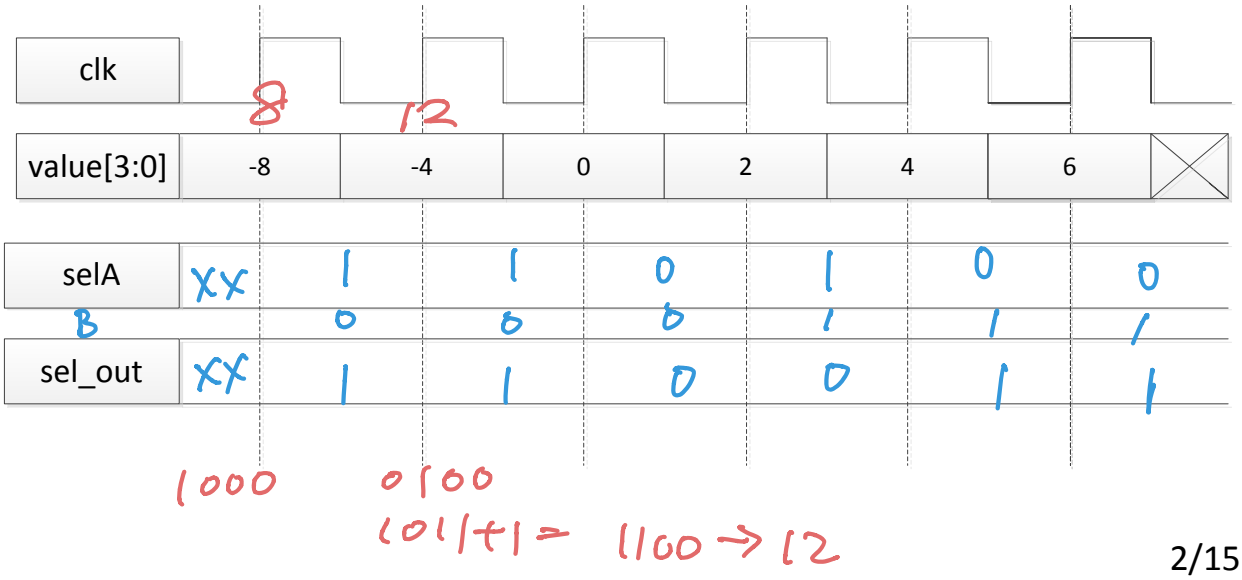
	Points	Score		Points	Score
Problem 1	10		Problem 6	30	
Problem 2	10		Problem 7	10	
Problem 3	10		Problem 8	10	
Problem 4	15				
Problem 5	15		Total	110	

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

A. (6pts) Identify syntax error (total 12) and inappropriate code (or semantics error), correct, and explain: 0.5pts for each

```
module 2value_selector (sel_out, clk, rst, value[3:0])

input clk;
input rst; /* Active High Asynchronous Reset
input [3:0] value;

output sel_out;

reg selA, selB;

assign selA = (value > 4'd10 || value < 4'd4)? 1'b1: 1'b0;
assign selB = (value > 4'd0 && value < 4'd8)? 1'b1: 1'b0;

always (posedge clk or rst);
begin
    if(rst)
        sel_out=1'b0;
    else
        sel_out=selA ^ selB;
end

endmodules
```

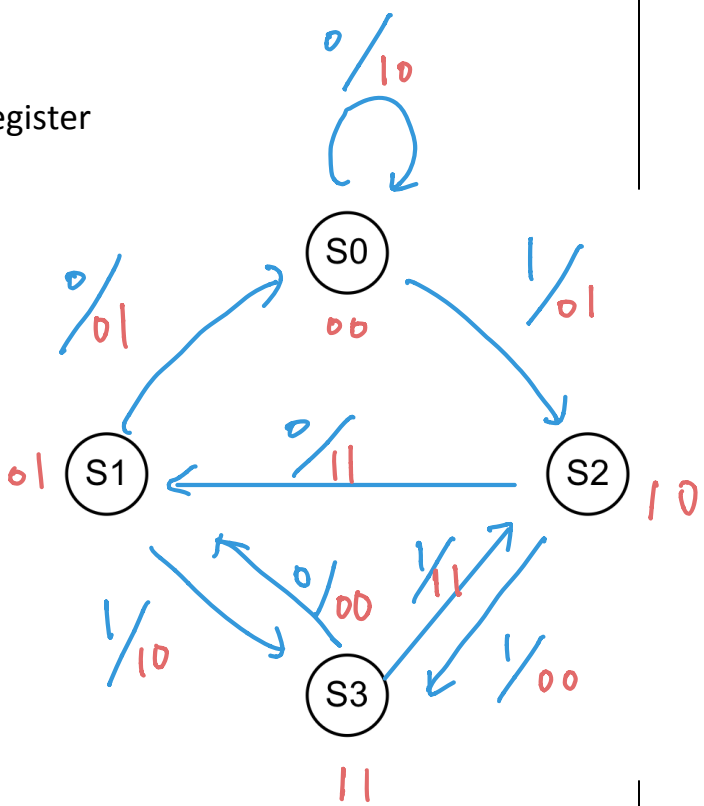
B. (4pts) Draw the waveform below based on the circuit in part A. Suppose that the signal **sel_out** has been reset to zero before the upcoming input. Use "xx" to indicate values that cannot be determined from the given information.



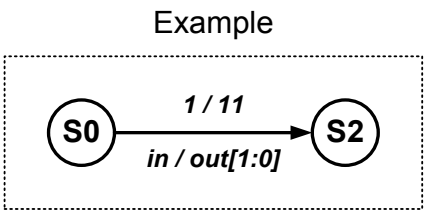
2. < Finite State Machine and Simulation > (10pts)

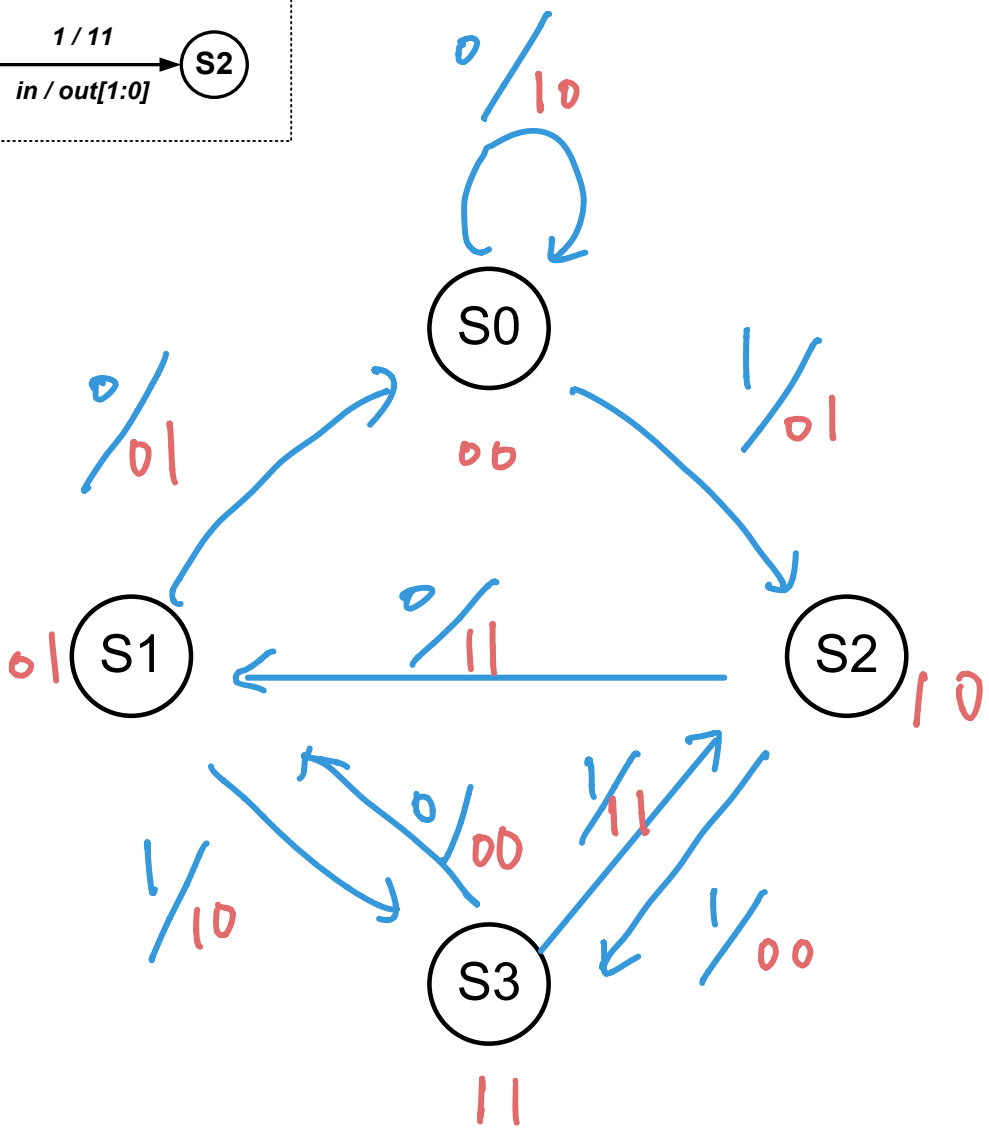
Given a Finite-State-Machine (FSM) as below.

```
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 : S3;
    S2: next_state = (in == 1'b0)? S1 : S3;
    S3: next_state = (in == 1'b0)? S1 : S2;
    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
```



(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 the given information.

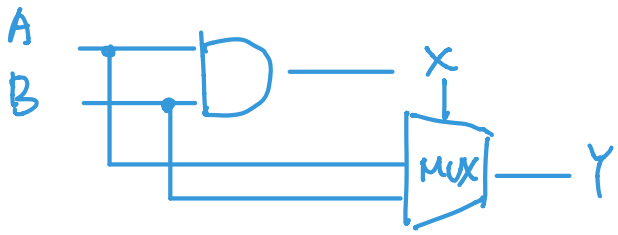
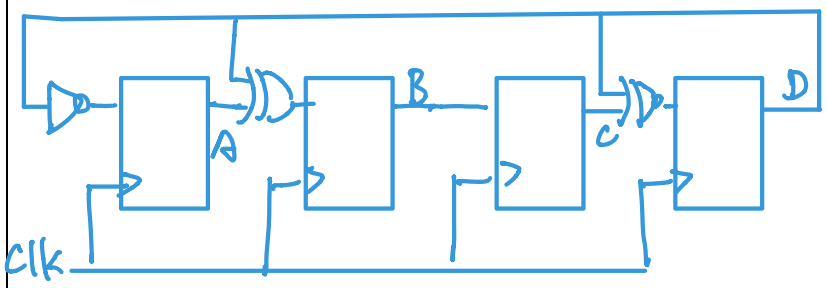
```
module testbench;
reg    clk, rst;
reg    in;
wire [1:0] out;

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

always @(*)begin
      $monitor("%t %b %b %b %b", $time, clk, rst, in, out);
End

endmodule
```

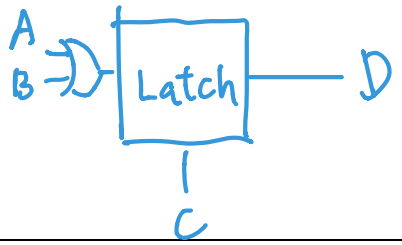
Monitor Output Response:

Time	clk	reset	in	out
0	0	0	0	xx
1	1	1	0	00
2	0	0	1	00
3	1	0	1	01
4	0	0	1	01
5	1	0	1	00
6	0	0	1	00
7	1	0	1	__ __
8	0	0	0	__ __
9	1	0	0	__ __
10	0	0	1	__ __
11	1	0	1	__ __
12	0	0	0	__ __
13	1	0	0	__ __
14	0	0	0	__ __
15	1	0	0	__ __
16	0	0	0	__ __

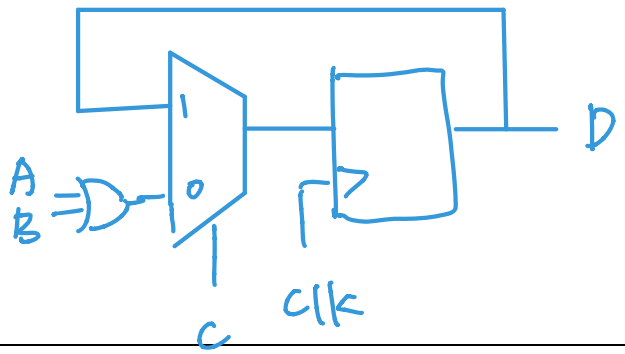
3. < Logic Synthesis + Blocking & Non-Blocking > (10pts)

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)? A: B; 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

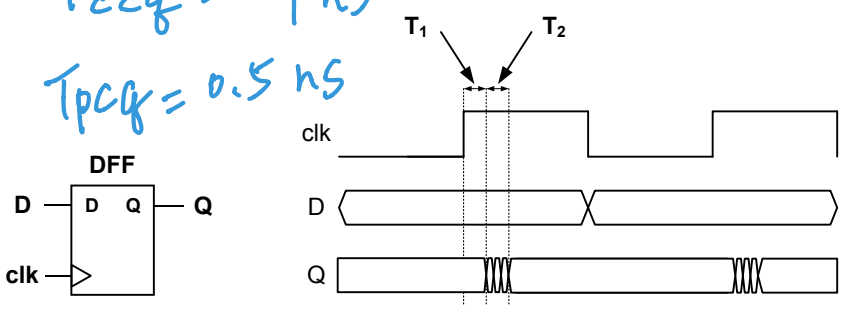
always@(posedge clk)
begin
if (!C)
D <= A | B;
end
	

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:

$T_{ccq} = 0.4\,ns$

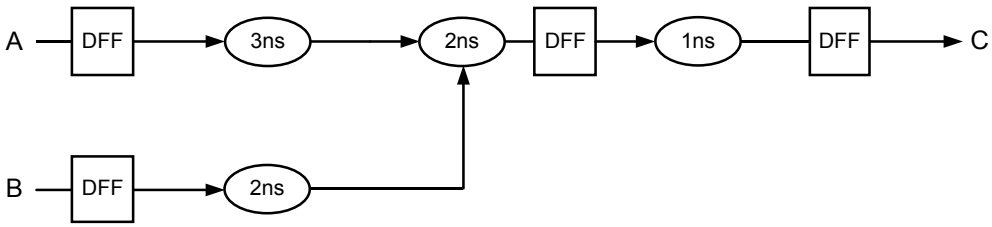
$T_{pcq} = 0.5\,ns$



$\frac{1}{200 \times 10^6} = \frac{10}{2}\,ns = 5ns$

T_1=0.4ns T_2= 0.1ns

The circuit below operates at the clock frequency of **200MHz.** 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.

$T_{clk} > T_{pcq} + \begin{pmatrix} 3+2 \\ 2 \\ 1 \end{pmatrix} + T_{setup} \Rightarrow T_{setup} < 2.5$ (−0.5 → 2.5 → 3.5) (?)

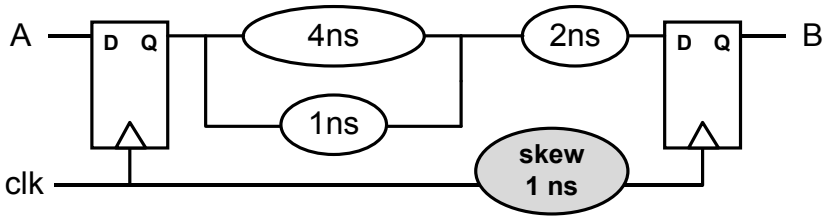
$T_{hold} < 0.4 + \begin{pmatrix} 5 \\ 4 \\ 1 \end{pmatrix}$ $T_{hold} < 1.4$

(b) (2pts) If **T_{setup} (Setup Time) = 1ns** **T_{hold} (Hold Time) = 1.5ns**

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.

(c) (5pts) Following 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.

(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.



5. <Coding for Synthesis> (15%)

A. (8pts) Loadable, Up-Down Counter

Create a module that can count in both the up and down directions. The preset is to be set to decimal value **4**. Also, upon asserting the load signal **load**, the register value should be set to the input value **data_in**. Both the **reset** and the **load** are **synchronous** and the module should count on the rising edge of the **clock**. The following are the ports of the module:

clk: 1-bit clock input, all actions performed on rising edge
rst 1-bit reset, causes reset on '1' (synchronous)
up_down 1-bit input (if '1', then count up, if '0', then count down)
load 1-bit load enable input, loads synchronized with CLK rising edge
data_in 3-bit input data for loading counter value
Q 3-bit result

```
module counter (clk, rst, up_down, load, data_in, Q);
input clk, rst, updown, load;
input [2:0] data_in;
output [2:0] Q;
    always @(posedge clk) begin
      if (rst)
          Q <= 4;
      else if (load)
          Q      <= data_in
      else if (up-down)
          Q <= Q + 1;
      else  Q <= Q - 1;
    end

endmodule
```

B. (7pts) Gray Counter

Create a three-bit gray counter with positive edge reset. The preset is to be set to value 000. Recall that a Gray Counter changes only one-bit at a time. For example, a 2-bit Gray counter has a count sequence 00, 01, 11, 10 corresponding to a decimal count values of 0, 1, 2, 3 respectively. In the Gray count sequence, only one bit changes between adjacent count values, and the right-most bit is changed as long as it does not result in a code word that has been visited earlier.

clk: 1-bit clock input, all actions on positive edge
rst: 1-bit reset, causes reset on positive edge (asynchronous)
gray_out: 3-bit result

0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0

```
module gray_counter (clk, rst, gray_out);
input clk, rst;
output [2:0] gray_out;
reg gray_out;
always @(posedge clk) begin
    if(rst)
        gray_out <= 3'b000;
    else begin
        case(gray_out)
            3'b000: gray_out <= 3'b001;
                    :
            3'b101: gray_out <= 3'b100;
        endcase
    end

endmodule
```

6. < Synthesis Issues > (30pts)

A. < Important files related to Design Compiler >

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

(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) 記錄在不同環境下的 cell 及 timing 資訊
(提供給 dc 做合成用)
(b) .sdf 記錄合成後的 delay 資訊
(c) 為 library 檔，合成後 verilog netlist 內部 cell 資訊

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]*

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

If we specify the clock to period 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?

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) Following 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.

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 these files are for and what will happen if we do not include these files.

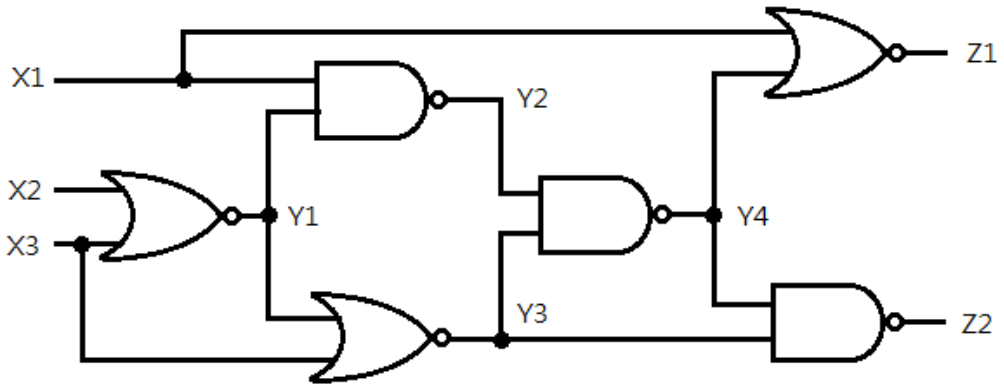
(b) (2pts) Please modify Design Compiler setting file .synopsys_dc.setup as shown below. **(JUST NEED TO POINT OUT WHERE TO MODIFY)**

```
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"
…
```

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

7. < Timing Analysis > (10pts)

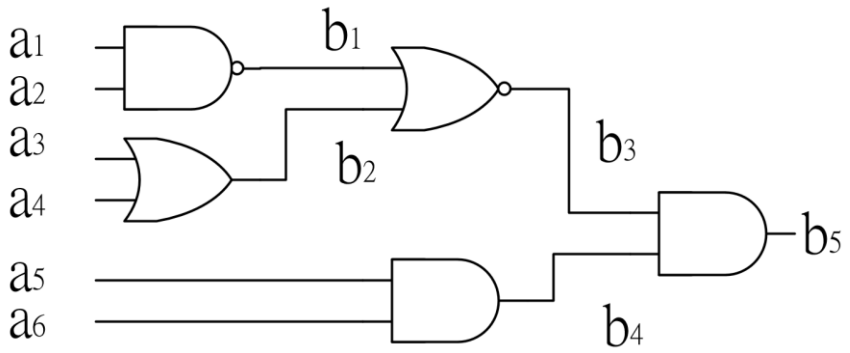
Calculate the arrival time, required time, and slack at each gate output, and find a critical path from primary input to primary output. Assume the delays of NAND gates and NOR gates are **5ns** and **2ns**, respectively. The arrival time at primary inputs is **0ns** and the required time at primary outputs is **16ns**.



X1: Arrival _____; Required _____; Slack _____
X2: Arrival _____; Required _____; Slack _____
X3: Arrival _____; Required _____; Slack _____
Y1: Arrival _____; Required _____; Slack _____
Y2: Arrival _____; Required _____; Slack _____
Y3: Arrival _____; Required _____; Slack _____
Y4: Arrival _____; Required _____; Slack _____
Z1: Arrival _____; Required _____; Slack _____
Z2: Arrival _____; Required _____; Slack _____
Critical path: _____

8. < Design for Testability > (10pts)

Consider the following circuit.



Q1. How many SSF in the circuit? (2pts)

Q2. Find the input vector (a1, a2, a3, a4, a5, a6) to test for the following stuck-at-faults. (2pts for each)

(a) b1 stuck at 1.

(b) b2 stuck at 0.

(c) b3 stuck at 0.

(d) b4 stuck at 1.

ANS