# Computer-Aided VLSI System Design

## Midterm Examination

2019. 11. 21

Name 吳牧庭

Student ID R08943094

## Instructions

This is a ***CLOSED*** book exam. The exam is to be completed in **180** 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 **18** pages of the exam. **Note that your raw score of this exam will be normalized to 100 points.**

### Score Board (to be filled by TAs)

| | Points | Score | | Points | Score |
|---|---|---|---|---|---|
| Problem 1 | 10 | 6 | Problem 6 | 20 | 17 |
| Problem 2 | 10 | 10 | Problem 7 | 7 | 6 |
| Problem 3 | 10 | 10 | Problem 8 | 35 | 30+1 |
| Problem 4 | 10 | 10 | | | |
| Problem 5 | 3 | 3 | Total | 105 | 92+1 |

## 1. (10pts) Code Debugging and Simulation

A. (6pts) Identify syntax and semantic errors. Correct them and put annotations.

+6 Miss one error or mistake one error will minus 1 point until 0.

```
1  module 2value multiplier (      >cannot contain white space
                                      need to start with a letter
2      clk,       /Clock  //
3      rst_n,  /Asynchronous reset active low
                //
       valueA,
4      valueB, comma
       valueSum
5  ); semicolon


   input clk;
   input rst_n;
   input [3:0] valueA;
   input [3:0] valueB;

   reg [3:0] valueA_r, valueB_r;
   reg [7:0] valueSum_r;
6  reg [7:0] valueA_sgnExt, valueB_sgnExt;
   wire
       output
7  wire [7:0] valueSum;
   wire [7:0] multiply_result:

8  assign valueA_sgnExt = {4{valueA_r[3]}, valueA_r};
9  assign valueB_sgnExt = {4{valueB_r[3]}, valueB_r};
```

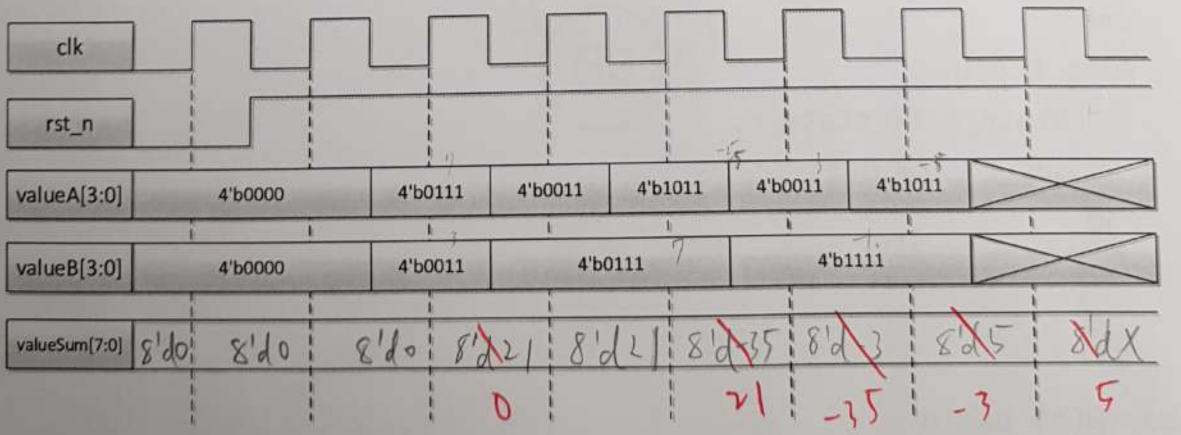
11111011
00000111

11111011
11111011

NTU GIEE Computer-Aided VLSI System Design, Fall 2019

```
assign multiply_result = valueA_sgnExt * valueB_sgnExt;

assign valueSum = valueSum_r;


always@ (posedge clk or negedge rst_n) begin
10      if (!rst_n) begin
            valueA_r <= 0;
            valueB_r <= 0;
11          valueSum_r <= 0; end
12      else begin
            valueA_r <= valueA;
            valueB_r <= valueB;
13          valueSum_r <= multiply_result; end
end
14 endmodule
```

+0 **B. (4pts)** Please draw the waveform for the circuit in part A. You can write value in signed decimal or binary format. Use "xx" to indicate values that cannot be determined given the provided information.



## 2. Finite State Machine and Simulation (10pts)

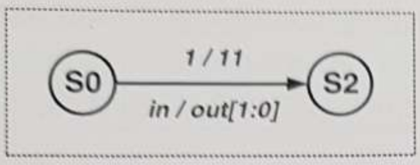
Given a Finite-State-Machine (FSM) as below.

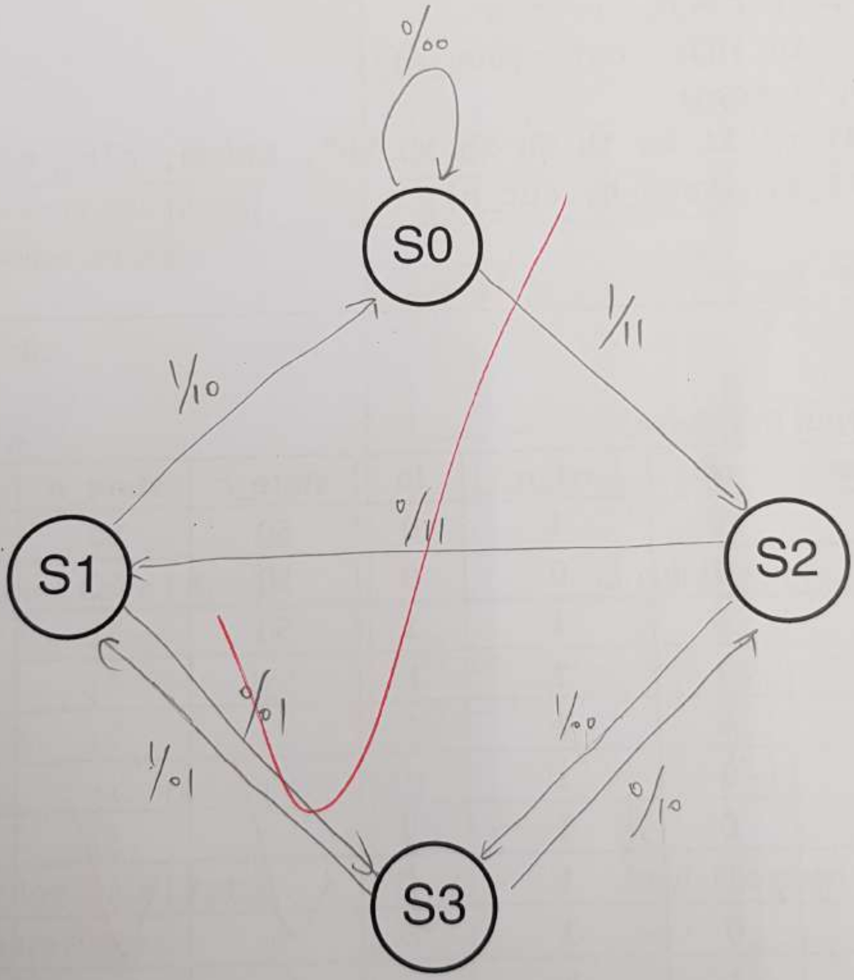
```
module FSM (clk, rst_n, in, out_r);
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

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

reg [1:0] out_r, out;
reg [1:0] state_c, state_n;

always @(*) begin
    case(state_c)
        S0: state_n = (in == 1'b1)? S2 : S0;
        S1: state_n = (in == 1'b1)? S0 : S3;
        S2: state_n = (in == 1'b0)? S1 : S3;
        S3: state_n = (in == 1'b0)? S2 : S1;
        default: state_n = 2'b01;
    endcase
end


always@(posedge clk or negedge rst_n) begin
    if(~rst_n) begin
        state_c <= S0;
        out_r   <= 2’b00;
    end
    else begin
        state_c <= state_n;
        out_r   <= out;
    end
end


always@(*) begin
    out[1] = in ^ state_c[1];
    out[0] = in ^ state_c[0] ^ state_c[1];
end
```

+5

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

Example





T5

(b) **(5pts)** The FSM is included as a design under test (DUT) in the testbench. After simulation, the command window shows the outputs. Please design a DUT (including input pattern) to test **all the possible conditions** based on operations of the FSM **within the given period**.

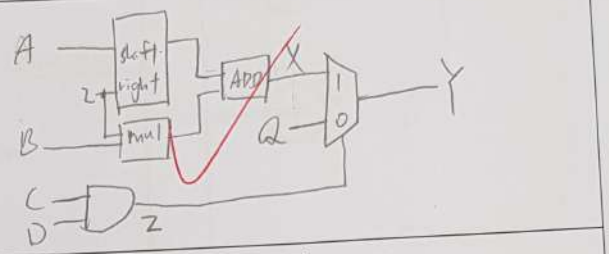
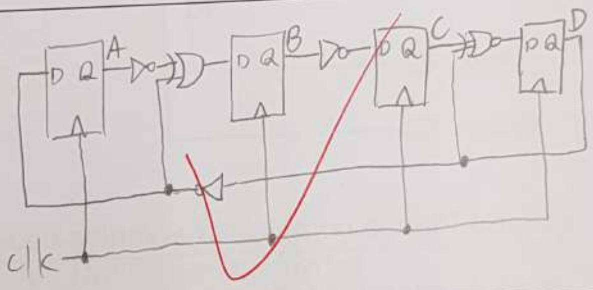
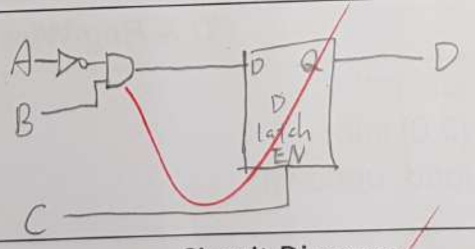
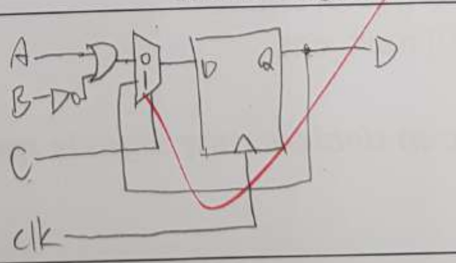
```
module testbench;
reg  clk, rst_n;
reg  in;
wire [1:0] out_r;
FSM DUT(.clk(clk), . rst_n
(rst_n), .in(in), .out_r(out_r));
always @(*) begin
   $monitor("%t %b %b %b %b %b", $time, clk, rst_n, in,
    state_c, state_n, out_r);
end
endmodule
```

**Monitor Output Response:**

| Time | clk | rst_n | in | state_c | state_n | out_r |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | S0 | S0 | XX |
| 1 | 1 | 0 | 0 | S0 | S0 | 00 |
| 2 | 0 | 1 | 1 | S0 | S2 | 00 |
| 3 | 1 | 1 | 1 | S2 | S3 | 11 |
| 4 | 0 | 1 | 0 | S2 | S1 | 11 |
| 5 | 1 | 1 | 0 | S1 | S3 | 11 |
| 6 | 0 | 1 | 0 | S1 | S3 | 11 |
| 7 | 1 | 1 | 0 | S3 | S2 | 01 |
| 8 | 0 | 1 | 0 | S3 | S2 | 01 |
| 9 | 1 | 1 | 0 | S2 | S1 | 10 |
| 10 | 0 | 1 | 1 | S2 | S3 | 10 |
| 11 | 1 | 1 | 1 | S3 | S1 | 00 |
| 12 | 0 | 1 | 1 | S3 | S1 | 00 |
| 13 | 1 | 1 | 1 | S1 | S0 | 01 |
| 14 | 0 | 1 | 1 | S1 | S0 | 01 |
| 15 | 1 | 1 | 0 | S0 | S0 | 10 |
| 16 | 0 | 1 | 0 | S0 | S0 | 10 |

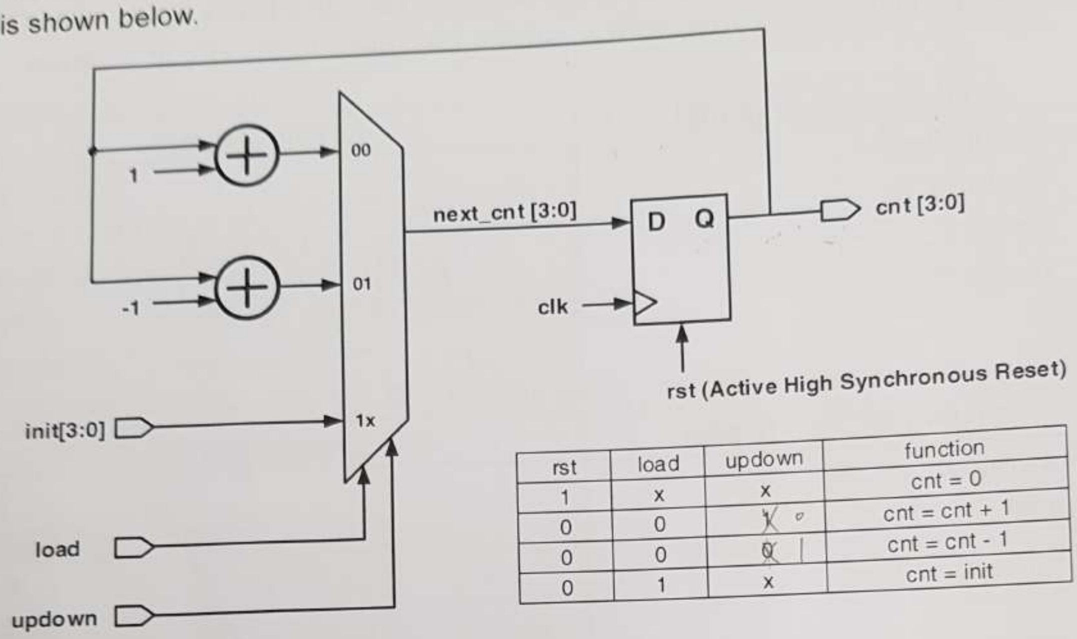
## 3. (10 +10 pts) Logic Synthesis + Blocking & Non-Blocking

Please draw the corresponding circuits (in the right column) according to the Verilog codes (in the left column). You can use AND, OR, NAND, NOR, XOR, XNOR, NOT, MUX, D Flip-Flop, Latch, Shifter, Adder, Multiplier in the circuit diagram.

| **(a)** Verilog Code **(2 pts)** | Circuit Diagram |
|---|---|
| always @(*) begin<br>X = A/4+B*2;<br>Z = C & D;<br>Y = (Z)? X: Q;<br>end |  A<br>shift right<br>2<br>B<br>mul<br>ADD<br>X<br>Q<br>1<br>0<br>Y<br>C<br>D<br>Z |
| **(b)** Verilog Code **(2 pts)** | Circuit Diagram |
| always @(posedge clk) begin<br>A <= ~D;<br>B <= ~A ^ ~D;<br>C <= ~B;<br>D <= C ~^ D;<br>end |  |
| **(c)** Verilog Code **( 3pts)** | Circuit Diagram |
| always@(A or B or C) begin<br>if(C)<br>D = ~A & B;<br>end |  |
| **(d)** Verilog Code **( 3pts)** | Circuit Diagram |
| always@(posedge clk) begin<br>if (!C)<br>D <= A \| ~B;<br>end |  |

## 4. (10pts) Verilog Design

A counter with synchronized initial value loading and up count/down count function is shown below.



| rst | load | updown | function |
|---|---|---|---|
| 1 | x | x | cnt = 0 |
| 0 | 0 | ~~1~~ 0 | cnt = cnt + 1 |
| 0 | 0 | ~~0~~ 1 | cnt = cnt - 1 |
| 0 | 1 | x | cnt = init |

**(10pts)** Please complete the Verilog code of this counter:

```
module counter (clk, rst, init, load, updown, cnt);
```

***I/O & Reg/Wire Declaration***

```
input clk, rst;
input [3:0] init;
input load, updown;

output reg [3:0] cnt;
reg [3:0] next_cnt;

// You can declare new signals here if you need
```

### Combinational Logic for the Counter (5pts)

```
always @ (*) begin
   if (load)
      next_cnt = init;
   else if (updown)
      next_cnt = cnt - 1'b1;
   else
      next_cnt = cnt + 1'b1;
```

+10

```
end
```

### Sequential Logic for the Counter (5pts)

```
always @ ( posedge clk                                  ) begin
if (rst)
     cnt <= 4'b0 ;
else
    cnt <= next_cnt ;
end
endmodule
```

## 5. (3pts) Operating condition

+3

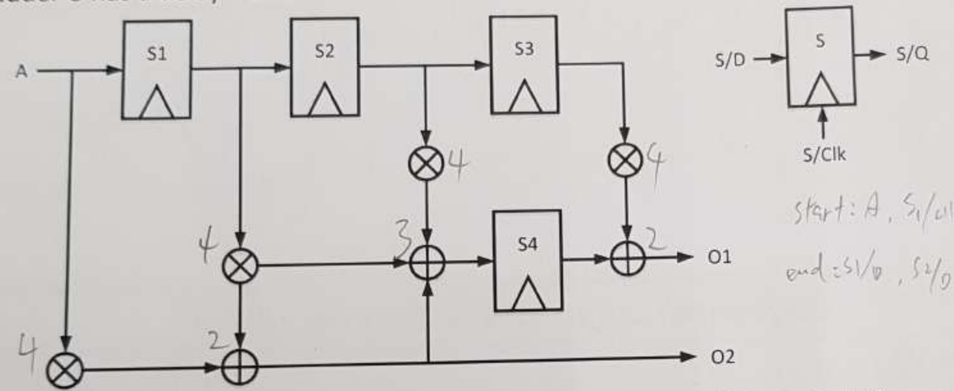
Fill in High/Low in each block ()

| Operational condition | Vt | supply voltage | temperature |
|---|---|---|---|
| fast | low | high | low |
| slow | high | low | high |

## 6. (20pts) Timing path & Setup/hold time

+17

Consider the circuit below, all multiplier has a delay = 4ns, adder-2 has a delay = 2 ns, adder-3 has a delay = 3 ns



start: A, S1/clk, S2/clk, S3/clk, S4/clk

end: S1/D, S2/D, S3/D, S4/D, O1, O2

### A. (7pts) List all timing path. How much timing path in total? (e.g. A -> S4/D)

+7

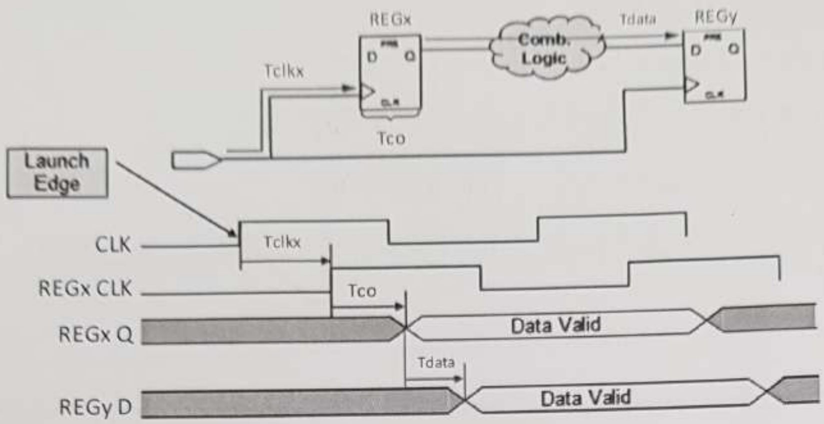
A → S1/D, S1/clk → S2/D, S2/clk → S3/D, S3/clk → O1,

A → S4/D, S1/clk → S4/D, S2/clk → S4/D, S4/clk → O1

A → O2, S1/clk → O2,

10 paths #

**B. (13pts) Setup/Hold time constraint**



$T_{cq} = 0.5$

$T_{clkp} = 10ns$

Following the circuit in 5.A. The registers' timing diagrams are shown above, and **Tco= 0.5ns** is same for every register. For register Si, it's Tclk is denoted as **Tclki**. The circuit in 5.A. operates at the clock frequency of **100MHz**. Suppose that the rise, fall delays for each combinational element are the same, and the input delay of A is $D_A$, output delay of O1 and O2 are $D_{O1}$ and $D_{O2}$.

**B1. (3pts)** Write the timing inequality for setup time in terms of Tclk1, Tclk2, Tclk3, Tclk4, $D_A$, $D_{O1}$, $D_{O2}$.

$T_{setup} = 0.5$
$T_{i/o} = 0.5$
$T_{clk1} = 0.3$
$T_{clk2} = 0.15$
slack :

setup check : RT > AT

+3

$\Rightarrow$ $10ns - T_{setup} + T_{clk4} > D_a + 4 + 2 + 3$ $(A \to S4/D)$

$10ns - T_{setup} + T_{clk1} > D_a$ $(A \to S1/D)$

$10ns - T_{setup} + T_{clk2} > T_{clk1} + 0.5$ $(S1/clk \to S2/D)$

$10ns - T_{setup} + T_{clk4} > T_{clk1} + 0.5 + (4+2+3)$ $(S1/clk \to S4/D)$

$10ns - D_{O2} > T_{clk1} + 0.5 + 4 + 2$ $(S1/clk \to O2)$

$10ns - T_{setup} + T_{clk3} > T_{clk2} + 0.5$ $(S2/clk \to S3/D)$

$10ns - T_{setup} + T_{clk4} > T_{clk2} + 0.5 + 4 + 3$ $(S2/clk \to S4/D)$

$10ns - D_{O1} > T_{clk3} + 0.5 + 4 + 2$ $(S3/clk \to O1)$

$10ns - D_{O1} > T_{clk4} + 0.5 + 2$ $(S4/clk \to O1)$

**B2. (3pts)** Write the timing inequality for hold time in terms of Tclk1, Tclk2, Tclk3, Tclk4, $D_A$, $D_{O1}$, $D_{O2}$.

Thold=0.1
Ti/o=0.5
Tclk1=0.3
Tclk2=0.15

+3

hold check : AT > RT

Da > Thold + Tclk1 (A → S1/D)

Da + 4+2+3 > Thold + Tclk4 (A → S4/D)

Tclk1 + 0.5 > Thold + Tclk2 (S1/clk → S2/D)

Tclk1 + 0.5 + 4+3 > Thold + Tclk4 (S1/clk → S4/D)

Tclk2 + 0.5 > Thold + Tclk3 (S2/clk → S3/D)

Tclk2 + 0.5 + 4+3 > Thold + Tclk4 (S2/clk → S4/D)

**B3. (3pts)** If **Tsetup (Setup Time) = 0.5ns, Thold (Hold Time) = 0.1ns,** $D_A$=$D_{O1}$=$D_{O2}$= **0.5ns, Tclk1=0.3ns, Tclk2=0.15ns, Tclk3=Tclk4=0ns.**

Explain if there are setup time and hold time violations in this circuit. If yes, identify the violated path(s).

+1

setup violations :

S1/clk → S4/D Slack = RT − AT = (10 − 0.5 + 0) − (0.3 + 0.5 + 4 + 2 + 3) = − 0.3 (violate)

↑ critical path

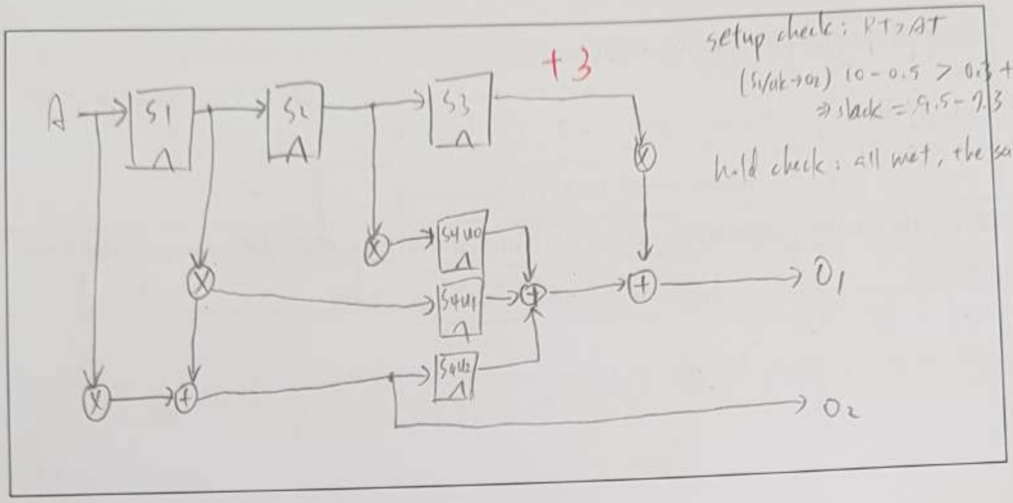
hold violations

A → S1/D slack = AT − RT = (0.5) − (0.1 + 0.3) = 0.1 (met) ✗

no

not hold time bottleneck.

**B4. (4pts)** Following part **B2**, if there are setup/hold time violations in this circuit, how to perform **retiming** to solve these issues (Assume the added registers have the same Tclk as the removed one)? Please draw your circuit and show that no setup and hold time violations exists.



setup check: RT > AT
(S1/clk→O2) 10 − 0.5 > 0.3 + 0.5 + 4 + 2 + 0.5
⇒ slack = 9.5 − 7.3 = 2.2 (met)
hold check: all met, the same before retiming

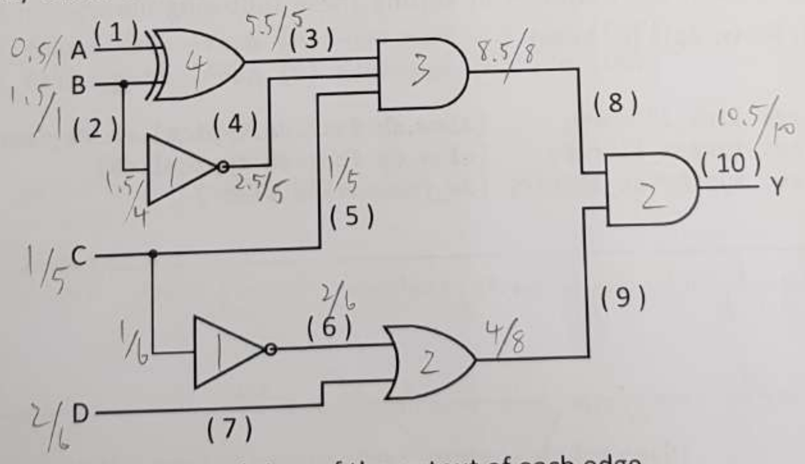
## 7. (7pts) Slack graph

+6

The following figure shows a combinational circuit with primary inputs A, B, C, and D. Arrival time of these primary inputs are 0.5ns, 1.5ns, 1ns, and 2ns respectively. The delay of each gate is listed below:

| XOR | INV | AND3 | AND2 | OR |
|---|---|---|---|---|
| 4ns | 1ns | 3ns | 2ns | 2ns |

Assume delay values of wires are zero.



**C1. (2%)** Write down the arrival time of the output of each edge.

+2

1:10 = [0.5, 1.5, 5.5, 2.5, 1, 2, 2, 8.5, 4, 10.5]

**C2. (3pts)** Given the require time at Y to be 10ns, compute require time and slack

+3

RT[1:10] = [1,1,5,5,5,6,6,8,8,10]

slack[1:10] = [0.5, -0.5, -0.5, 2.5, 4, 4,4, -0.5, 4, -0.5]

**C3. (2pts)** Please indicate the critical path and show the critical path by required time values

+1

(2) → (3) → (8) → 10

RT: 1.5 → 5.5 → 8.5 → 10.5

## 8. (35pts) Synthesis

1. Please explain the purposes of setting these following libraries in logic synthesis stage. **(4pts, 2pts for each)**

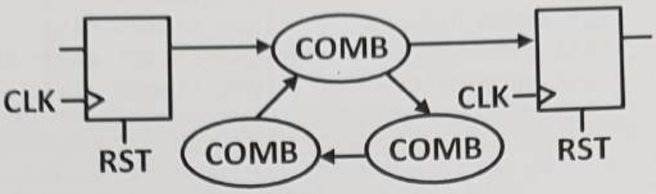
```
a. set link_library      {slow.db fast.db typical.db dw_foundation.sldb}
   set target_library    {slow.db fast.db typical.db}
b. set synthetic_library {dw_foundation.sldb }
```

a. Specify technology libraries, containing timing/area info for synthesis.

b. Provide timing/area info for designware IPs.

2. What is the problem we may encounter when performing logic synthesis for the following circuit? **(3pts)**

-2

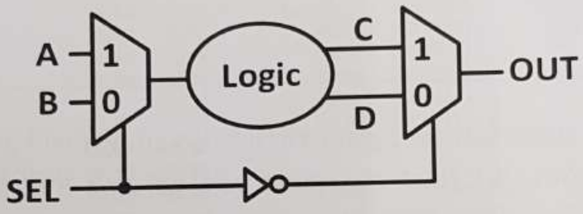


Combinational loop, hard to perform STC.

STA ?

3. Please explain the meaning of commands for the corresponding circuit. **(2pts)**

-1



```
set_false_path -from {A} -through {C} -to {OUT}
set_false_path -from {B} -through {D} -to {OUT}
```

Don't report timing violations for the 2 paths.

analyze

4. Please explain the clock skew and jitter. **(4pts, 2pts for each)**

Skew: the time difference of clk to arrive at different flops.

jitter: the time difference of clk period among cycles.

5. Please explain the following commands for clock setting. **(10pts, 2pts for each)**

```
a. create_clock -name clk -period 10 clk
b. set_fix_hold                  [get_clocks clk]
c. set_dont_touch_network        [get_clocks clk]
d. set_ideal_network             [get_ports  clk]
e. set_clock_uncertainty 0.1     [get_clocks clk]
```

a. specify clk port and set timing constraint for clk period

b. fix hold time violations via buffer insertion,

c. don't optimize clk network since clk tree should not be generated here,

d. set clk delay to 0

e. consider clk uncertainty of 0.1 ns since clk arrival time at flops varies,

6. In synthesis stage for low power circuit design, we will use the following command to include both HVT (high-threshold voltage) and RVT (regular-threshold voltage) cells, please explain the advantage for this process. **(3pts)**

```
set_target_library "rvt.db hvt.db"
```

For long paths, use RVT to meet timing constraint,
For short paths, use HVT to reduce power consumption.

7. When performing timing analysis with the following command, if there are **only** two paths in the timing report ***Design.timing*** shown as the following table, please indicate which path is the critical path **(1pt)** and explain why. **(2pts)**

```
report_timing -path full -delay max > Design.timing
```

**Path1**

```
Startpoint: Y[1] (input port clocked by clk)
Endpoint: FB_A[10] (output port clocked by clk)
Path Group: clk
Path Type: max

Point                          Incr    Path
-------------------------------------------------
clock clk (rise edge)          0.00    0.00
clock network delay (ideal)    0.50    0.50
input external delay           5.00    5.50 r
Y[1] (in)                      0.00    5.50 r
U620/Y (CLKINVX1)              0.04    5.55 f
U619/Y (MXI2X1)                0.12    5.66 r
U293/CO (ADDFXL)               0.61    6.27 r
U294/CO (ADDFXL)               0.33    6.60 r
U430/Y (AND2X1)                0.19    6.79 r
U429/Y (XOR2X1)                0.16    6.95 f
U702/Y (MXI2X1)                0.56    7.51 r
U363/Y (INVX12)                0.75    8.26 f
FB_A[10] (out)                 0.00    8.26 f
data arrival time                      8.26

clock clk (rise edge)          10.00   10.00
clock network delay (ideal)     0.50   10.50
clock uncertainty              -0.10   10.40
output external delay          -0.50    9.90
data required time                      9.90
-------------------------------------------------
data required time                      9.90
data arrival time                      -8.26
-------------------------------------------------
slack (MET)                             1.64
```

**Path2**

```
Startpoint: Y[0] (input port clocked by clk)
Endpoint: FB_A[11] (output port clocked by clk)
Path Group: clk
Path Type: max

Point                          Incr    Path
-------------------------------------------------
clock clk (rise edge)          0.00    0.00
clock network delay (ideal)    0.50    0.50
input external delay           5.00    5.50 f
Y[0] (in)                      0.00    5.50 f
U622/Y (CLKINVX1)              0.04    5.54 r
U621/Y (MXI2X1)                0.09    5.63 f
U432/Y (AND2X1)                0.21    5.84 f
U293/CO (ADDFXL)               0.36    6.19 f
U294/CO (ADDFXL)               0.39    6.58 f
U430/Y (AND2X1)                0.21    6.79 f
U428/Y (AND2X1)                0.18    6.97 f
U427/Y (XOR2X1)                0.15    7.12 f
U701/Y (MXI2X1)                0.54    7.66 r
U368/Y (INVX12)                0.74    8.41 f
FB_A[11] (out)                 0.00    8.41 f
data arrival time                      8.41

clock clk (rise edge)          10.00   10.00
clock network delay (ideal)     0.50   10.50
clock uncertainty              -0.10   10.40
output external delay          -0.50    9.90
data required time                      9.90
-------------------------------------------------
data required time                      9.90
data arrival time                      -8.41
-------------------------------------------------
slack (MET)                             1.49
```

Path 2.

lower slack.

8. Please explain the reason why we should add the following code in initial block in the test bench for gate level simulation. **(3pts)**

```
$sdf_annotate(sdf_file.sdf, test_bench_module.top_instance);
```

Sdf files contain timing info. of the cells in netlist, without which the timing check in simulation will not be accurate.

-1

9. Why the synthesis report shows register with **latch** type when loading design? **(1pt)** and please also explain how to avoid this situation. **(2pts)**

```
===============================================================================
|   Register Name    |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     reg_B_reg      |   Latch    |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|    reg_ins_reg     | Flip-flop  |   4   |  Y  | N  | Y  | N  | N  | N  | N  |
|    alu_out_reg     | Flip-flop  |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|     reg_A_reg      | Flip-flop  |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
```

Combinational loop is found.

Avoid partially specified condition coding style.
eg. if without else, case without default