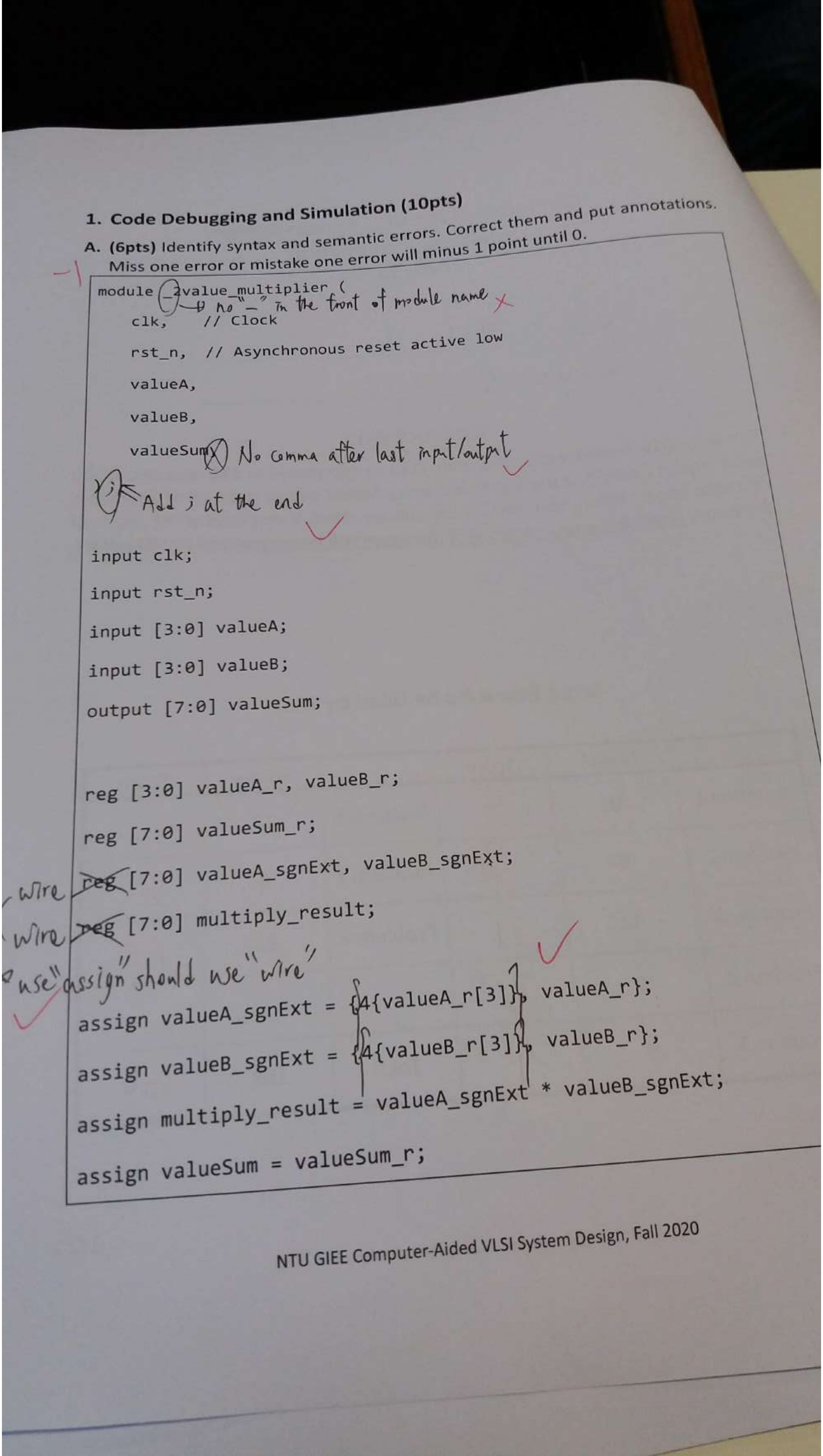


1. Code Debugging and Simulation (10pts)

A. (6pts) Identify syntax and semantic errors. Correct them and put annotations. Miss one error or mistake one error will minus 1 point until 0.

-1

```
module _value_multiplier (
       → no "_" in the front of module name ✗
    clk,    // Clock
    rst_n,  // Asynchronous reset active low
    valueA,
    valueB,
    valueSum,  No comma after last input/output ✓
);  ⇐ Add ; at the end ✓

input clk;
input rst_n;
input [3:0] valueA;
input [3:0] valueB;
output [7:0] valueSum;


reg [3:0] valueA_r, valueB_r;
reg [7:0] valueSum_r;
wire ~~reg~~ [7:0] valueA_sgnExt, valueB_sgnExt;
wire ~~reg~~ [7:0] multiply_result;
use "assign" should use "wire" ✓ ✓
assign valueA_sgnExt = {{4{valueA_r[3]}}, valueA_r};
assign valueB_sgnExt = {{4{valueB_r[3]}}, valueB_r};
assign multiply_result = valueA_sgnExt * valueB_sgnExt;
assign valueSum = valueSum_r;
```

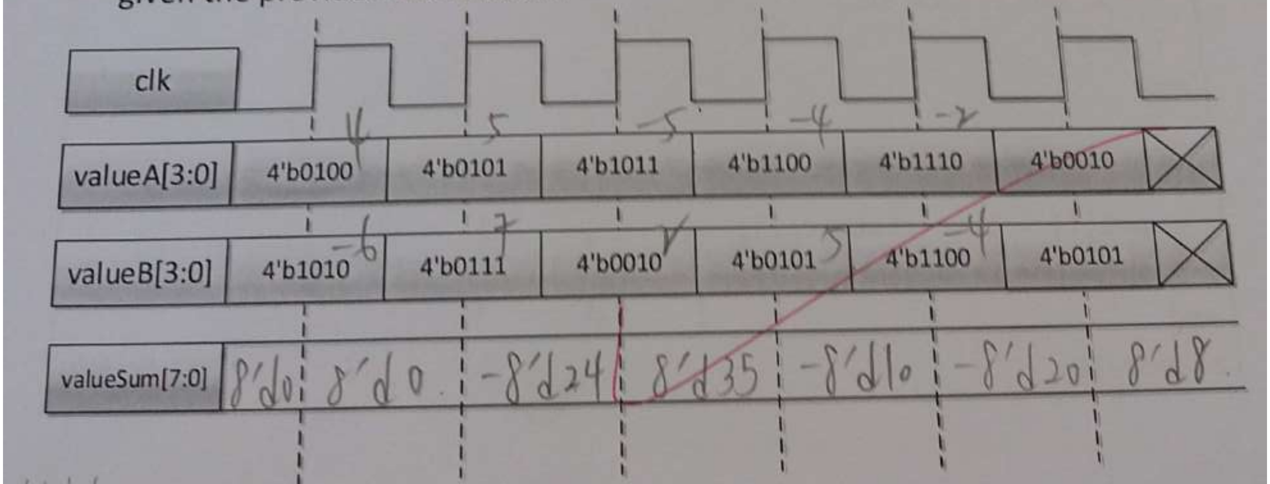
漏了@

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

active low

少了 always block 的 end

B. (4pts) Please show the waveform for the circuit in part A. Suppose that the signal **valueSum** is reset to zero in the beginning and all the syntax error and functionality are correct. Use "xx" to indicate values that cannot be determined given the provided information.



1111
1110
1101
1100
1011
1010
1001

1000 -8

11111111 -1
x 18 x2
11111110 -2

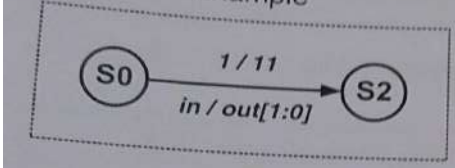
2. Finite State Machine and Simulation (8pts)

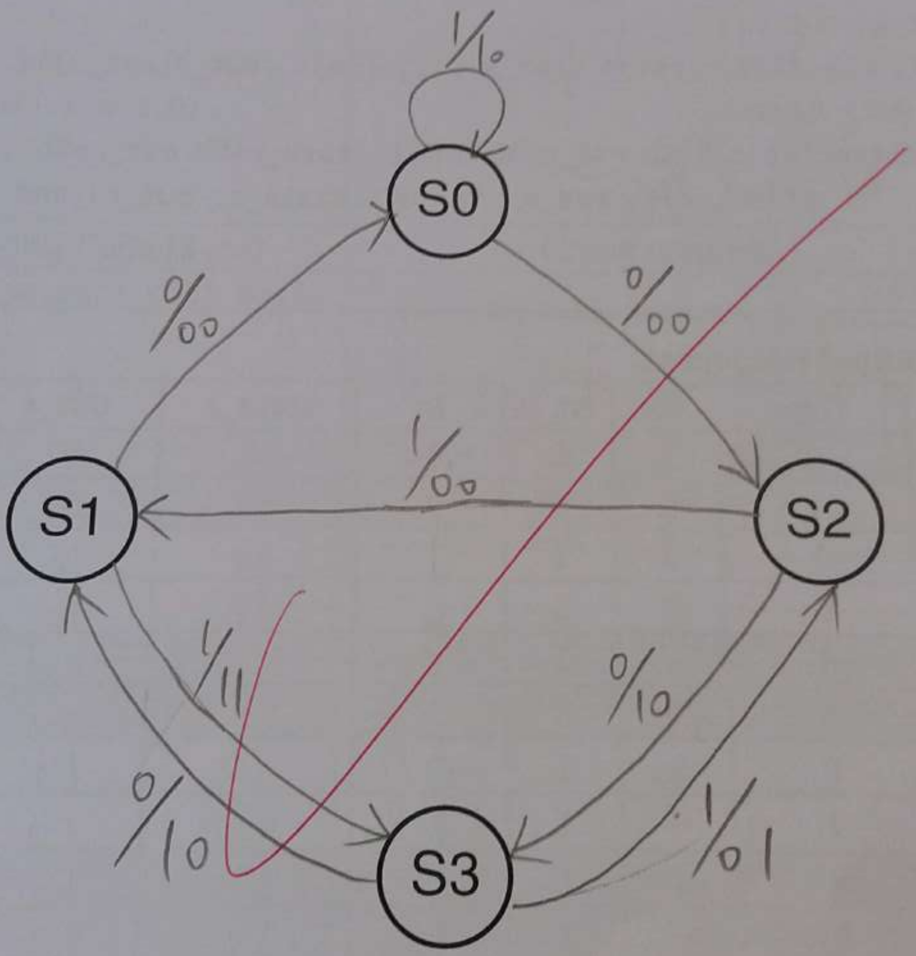
Given a Finite-State-Machine (FSM) as below.

```
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'b0) ? S2 : S0;
        S1: state_n = (in == 1'b0) ? S0 : S3;
        S2: state_n = (in == 1'b1) ? S1 : S3;
        S3: state_n = (in == 1'b1) ? S2 : S1;
    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                    S1   S2
    out[1] = in ^ state_c[1];        0    1    1
    out[0] = in & state_c[0];        1    0    1
end
endmodule
```

(a) (3pts) Please draw a Mealy state transition graph below for this FSM.

Example





(b) (5pts) The FSM is included as a design under test (DUT) in the testbench. After simulation, the terminal shows the outputs. Please design a DUT and input pattern to test **all the possible paths (include self-loops)** based on operations of the FSM **within the given period**.
(You should only change the value of **`in`** on **`negedge clk`**)

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

Monitor Output Response:

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

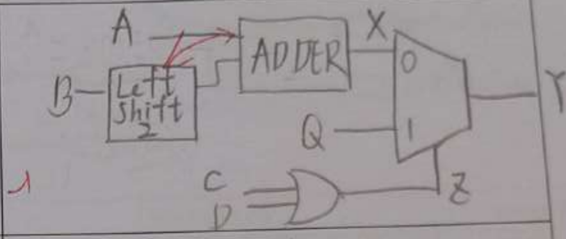
3. Logic Synthesis + Blocking & Non-Blocking (12 pts)

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 **(3 pts)**

```
always @(*) begin
    X = A + B << 2;
    Z = C | D;
    Y = (~Z) ? X : Q;
end
```

Circuit Diagram

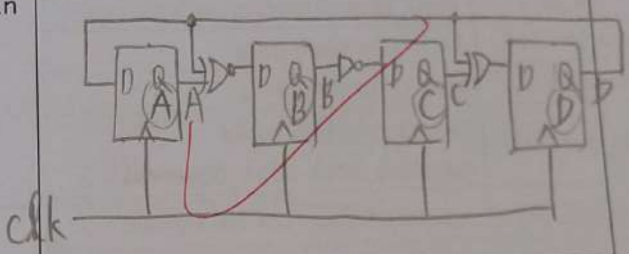


1

(b) Verilog Code **(3 pts)**

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

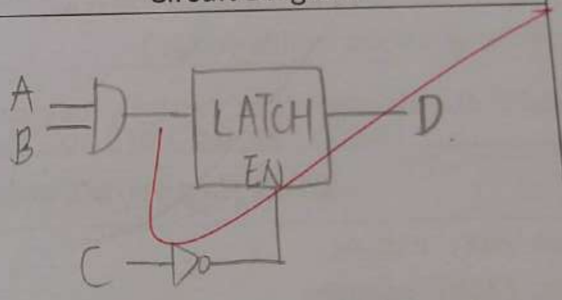
Circuit Diagram



(c) Verilog Code **(3pts)**

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

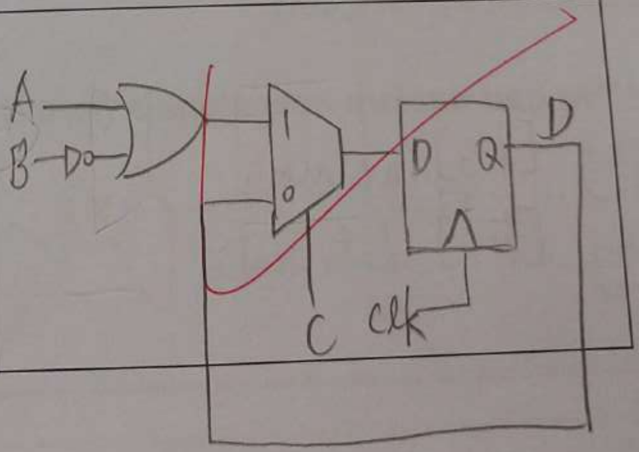
Circuit Diagram



(d) Verilog Code **(3pts)**

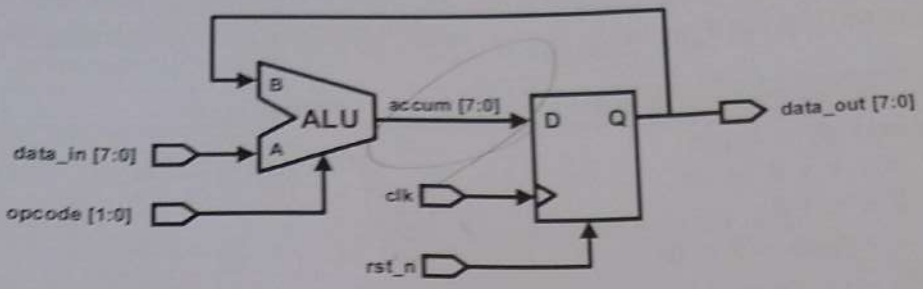
```
always @(posedge clk) begin
  if (C)
    D <= A | ~B;
end
```

Circuit Diagram



4. Verilog Design (6pts)

A deign with an ALU unit and its description of functions are shown below.



Signal	Description
clk	Input clock
rst_n	Input asynchronous negative reset
data_in [7:0]	Input unsigned data
accum [7:0]	Result from ALU
data_out [7:0]	Output unsigned data (positive clock edge triggered)
opcode [1:0]	Input operation control signals

opcode	ALU operation
2'b00	A + B
2'b01	A - B
2'b10	A XOR B
2'b11	A NOR B

(6pts) Please complete the Verilog code of this counter (you **don't** have to consider overflow in this design).

```
module alu_design (clk, rst_n, opcode, data_in, data_out);
```

I/O & Reg/Wire Declaration (1pt)

```
input clk, rst_n;
input [1:0] opcode;
input [7:0] data_in;
output [7:0] data_out;
```

// You can declare new signals here if you need

```
reg [7:0] accum;
reg [7:0] data-out;
```

✓

Combinational Logic for the Counter (3pts)

```
always @ (*) begin
  case(opcode)
    2'b00: accum = data_in + data_out;

    2'b01: accum = data_in - data_out;

    2'b10: accum = data_in ^ data_out;

    2'b11: accum=~(data_in | data_out);

  endcase
end
```

Sequential Logic for the Counter (2pts)

```
always @ ( posedge clk or negedge rst_n) begin
  if (!rst_n) begin
     data_out <= 8'b0;

  end else begin
     data_out <= accum;

  end.
end
```

```
endmodule
```

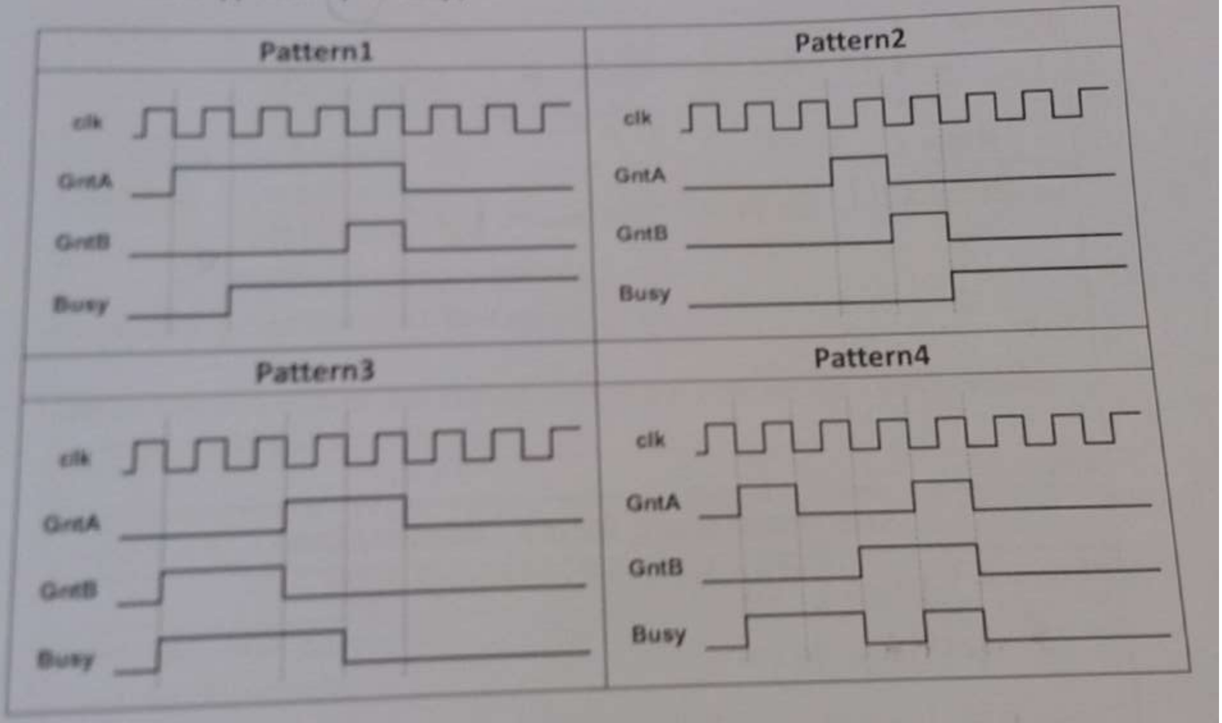
5. Formal Verification (9pts)

Consider 3 assertions and 4 patterns below. Please identify the patterns which are violated for each assertion.

(a) `assert_0: assert property (@(posedge clk)`
`!(GntB && !Busy))`

(b) `assert_1: assert property (@(posedge clk)`
`GntA && !GntB |-> ##[0:2] Busy)`

(c) `assert_2: assert property (@(posedge clk)`
`GntA || GntB |=> Busy)`

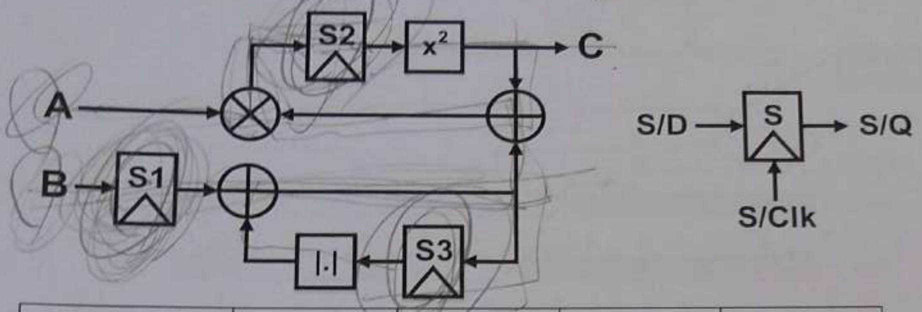


(9 pts) Fill in **Pass/Fail** in each block.

	Pattern1	Pattern2	Pattern3	Pattern4
(a)	Pass	Fail	Pass	Fail
(b)	Pass	Pass	Fail	Pass
(c)	Pass	Fail	Fail	Fail

6. Timing path & Setup/hold time (22pts)

Consider the circuit and the delay information below. Answer the following questions.



Operation	$\oplus$	$\otimes$	$\boxed{\lvert\cdot\rvert}$	$\boxed{x^2}$
Min Delay (ns)	0.5	3.0	0.2	1.5
Max Delay (ns)	1.0	4.0	0.5	2.0

A. (5pts) List all timing paths. How many timing paths in total? (e.g. A -> S1/D)

A → S2/D

S2/clk → C

S2/clk → S2/D

B → S1/D

S1/clk → S2/D.

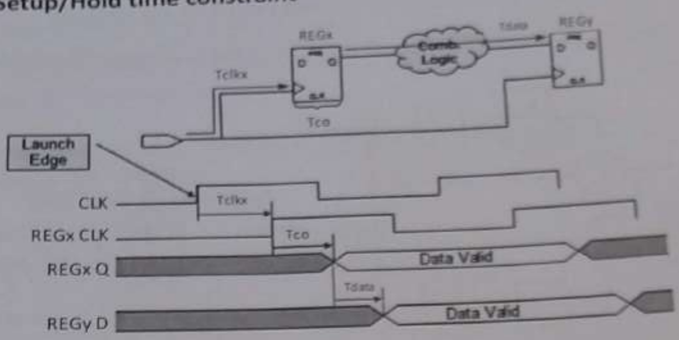
S1/clk → S3/D

S3/clk → S3/D

S3/clk → S2/D.

8 paths

B. Setup/Hold time constraint



Following the circuit in **6.A**. The registers' timing diagrams are shown above. T_{co} **= 0.5ns**, and T_{clk} **= 0.3ns** are the same for every register. Assuming that T_{setup} **(Setup Time) = 0.4ns**, T_{hold} **(Hold Time) = 0.1ns**, and $D_A = D_B = D_C =$ **1.5ns**. The circuit operates at the clock frequency of **125MHz**. 8 ns

B1. (5pts) Check that whether there are setup time violations or not by showing if the related inequalities for all timing paths are satisfied. If setup time violation(s) occur, identify the violated path(s).

RT > AT (Cycle time + T_{clk} − T_{setup} > T_{clk} + T_{co} + T_{pd})

(Met) A → S2/D : $8 + 0.3 - 0.4 > 1.5 + 4$ ✓

(Met) B → S1/D : $8 + 0.3 - 0.4 > 1.5$ ✓

(Met) S2/clk → C : $8 - 1.5 > 0.3 + 0.5 + 2$ ✓

(Met) S2/clk → S2/D : $8 + 0.3 - 0.4 > 0.3 + 0.5 + 2 + 1 + 4$ ✓

(Met) S1/clk → S2/D : $8 + 0.3 - 0.4 > 0.3 + 0.5 + 1 + 1 + 4$ ✓

(Met) S1/clk → S3/D : $8 + 0.3 - 0.4 > 0.3 + 0.5 + 1$ ✓

(Met) S3/clk → S3/D : $8 + 0.3 - 0.4 > 0.3 + 0.5 + 0.5 + 1$ ✓

(Met) S3/clk → S2/D : $8 + 0.3 - 0.4 > 0.3 + 0.5 + 0.5 + 1 + 1 + 4$ ✓

No setup-time violation

B2. (5pts) Check that whether there are hold time violations or not by showing if the related inequalities for all timing paths are satisfied. If hold time violation(s) occur, identify the violated path(s).

RT < AT   ($T_{clk} + T_{hold} < T_{clk} + T_{co} + T_{cd}$)

(Met) A → S2/D : 0.3 + 0.1 < 1.5 + 3 ✓

(Met) B → S1/D : 0.3 + 0.1 < 1.5 ✓

(Met) S2/clk → C : −1.5 < 0.3 + 0.5 + 1.5 ✓

(Met) S2/clk → S2/D : 0.3 + 0.1 < 0.3 + 0.5 + 1.5 + 0.5 + 3 ✓

(Met) S1/clk → S2/D : 0.3 + 0.1 < 0.3 + 0.5 + 0.5 + 0.5 + 3 ✓

(Met) S1/clk → S3/D : 0.3 + 0.1 < 0.3 + 0.5 + 0.5 ✓

(Met) S3/clk → S3/D : 0.3 + 0.1 < 0.3 + 0.5 + 0.2 + 0.5. ✓

(Met) S3/clk → S2/D : 0.3 + 0.1 < 0.3 + 0.5 + 0.2 + 0.5 + 0.5 + 3 ✓ ⇒ No hold time violat

B3. (7pts) Assuming that there is **time skew** between T_{clk3} and other T_{clk}, the relationship between those parameters is: **$T_{clk3} = T_{clki}$ + 0.1*Clk_cycle, where i = 1, 2. T_{clk1}, T_{clk2} and other conditions are the same as A2**. Explain if there are timing violations in this circuit. If yes, identify the violated path(s).

T_{clk1} , T_{clk2} = 1.1

setup time :

(Met) S1/clk → S3/D : 8 + 1.1 − 0.4 > 0.3 + 0.5 + 1 ✓

(Met) S3/clk → S3/D : 8 + 1.1 − 0.4 > 1.1 + 0.5 + 0.5 + 1

(Violated) S3/clk → S2/D : 8 + 0.3 − 0.4 > 1.1 + 0.5 + 0.5 + 1 + 1 + 4 ⇒ 7.9 > 8 (X

⇒ S3/clk → S2/D has setup time violation ✓

hold time :

(Met) S1/clk → S3/D : 1.1 + 0.1 < 0.3 + 0.5 + 0.5 ✓

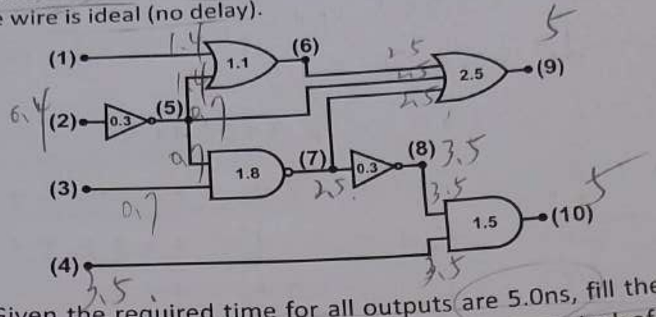
(Met) S3/clk → S3/D : 1.1 + 0.1 < 1.1 + 0.5 + 0.2 + 0.5

(Met) S3/clk → S2/D : 0.3 + 0.1 < 1.1 + 0.5 + 0.2 + 0.5 + 0.5 + 3. ✓

No hold time violation

7. Slack graph (8pts)

The figure below shows a combinational circuit with primary inputs. The delay of each input is 0.3ns and the delay of every gate is given on the gate (unit: ns). Assume the wire is ideal (no delay).



C1. (5pts) Given the required time for all outputs are 5.0ns, fill the blanks on the form with corresponding arrival time, required time, and slack of every node.

AT = Arrival Time, RT = Required Time.

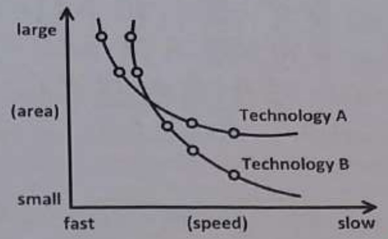
Node	(1)	(2)	(3)	(4)	(5)
AT (ns)	0.3	0.3	0.3	0.3	0.6
RT (ns)	1.4	0.4	0.7	3.5	0.7
Slack (ns)	1.1	0.1	0.4	3.2	0.1
Node	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**
AT (ns)	1.7	2.4	2.7	4.9	4.2
RT (ns)	2.5	2.5	3.5	5.0	5.0
Slack (ns)	0.8	0.1	0.8	0.1	0.8

C2. (3pts) Please indicate the critical path and show its slack values.

(2)→(5)→(7)→(9) : slack = 0.1 at all nodes.

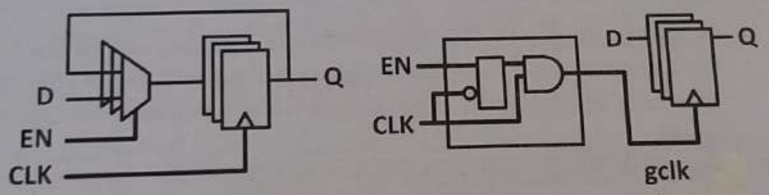
8. Synthesis (33pts)

1. Please explain why synthesis is **constraint-driven** and what is the meaning of the **tradeoff between area and speed** according to the following image. **(2pts)**

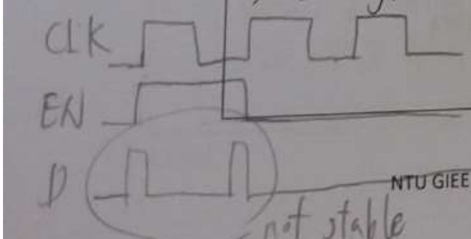


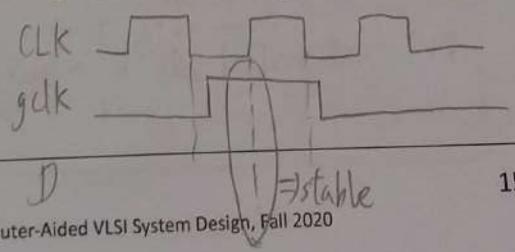
① Constraint-driven 代表由 designer 決定希望得到較好 area/timing 或 power 的 performance 而在 sdc 或 tcl 檔中下相对應的指令(constraint) 合成時是依據這些 constraint 來針对 area/power/timing 進行優化。② 若希望 timing (speed) fast 則 area 勢必有所犧牲而變大 (反之若希望 area small 則 timing 勢必有所犧牲而變慢 (speed)

2. Why we preferred the architecture on the right hand side instead of left hand side design in clock gating? **(2pt)**



為了讓 EN 使用falling edge trigger 的 cycle 製造出的gclk能stable為1或0. 否則易產生glitch

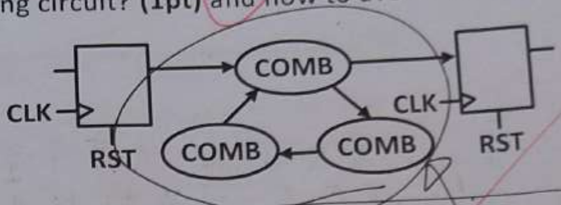




3. Please explain why should we consider ***PVT condition*** when synthesis **(1pt)** and what does ***PVT*** stand for? **(1pt)**

PVT 代表 operating condition, 根據製程.電壓.溫度
不同而造成 gate/cell 的 timing delay/area 有所差異
→而能定義出
P: process 製程 (越小越 fast)
V: voltage 電壓 (越大越 fast)
T: temperature 溫度 (越低越 fast)
max_delay : check setup time
min_delay : check hold time

4. What is the problem we may encounter when performing logic synthesis for the following circuit? **(1pt)** and how to avoid this situation? **(1pt)**



Combinational-loop may cause STA to have race-condition
⇒無法做 timing analysis
To avoid : Break the combinational loop here

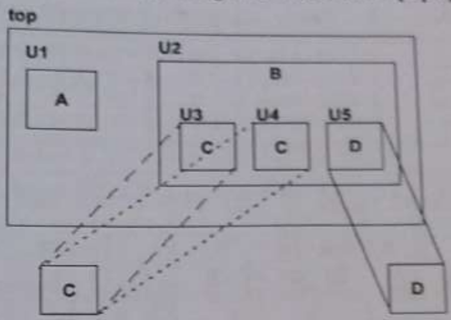
5. In synthesis stage for low power circuit design, we will use the following command to include both HVT (high-threshold voltage) and RVT (regular-threshold voltage) cells, please explain the advantage for this process. **(2pts)**

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

HVT : Let non-critical path to save power

RVT : Let critical path to improve timing

6. In a hierarchical design, submodules are sometimes referenced by more than one cell instance. What method is required to solve this situation? **(1pt)** And what is the effect after applying this method? **(1pt)**



uniquify：將 reference 到同一個 submodule 的 cell instance 分開，當作是不同的 submodule 處理

能對於 timing、area 各自進行優化得到更好結果 (performance)

~~但會執行~~ 且能保留其 hierarchy

7. Please explain the difference between ***RTL Level Netlist*** and ***Gate Level Netlist***. **(1pt)**

RTL 使用 assign、always block 描述電路

gate-level 使用 AND、OR 等邏輯閘及FF等描述電路.

RTL 經 synthesis 後可得 gate-level netlist

8. Please explain why we should add the following code into the testbench when performing gate level simulation. **(1pt)**

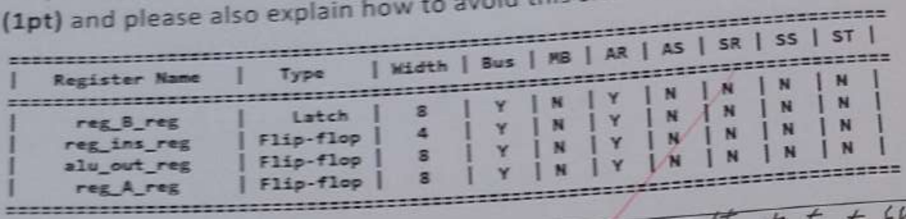
```
$sdf_annotate ("SDF_FILE_NAME", top_module_instance_name);
```

否則執行 gate-level simulation 時會使用 cell_model (tsmc13.

default 的 timing delay 而導致錯誤

sdf 裡定義 cell/gate 的 timing delay 用來做 simulation 才正

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



```
============================================================================================
|    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  |
============================================================================================
```

在 combinational part 中的 reg-B 的 next state 值沒有在所有情況下皆 assign 值, 例如 if-else 或 case 中某些 branch 未定義該值, 或未使用 default branch
solution: 在 combinational part 一開始就給予 default 值. 例如: reg-B-w =reg-B-r;
或是所有 if-else branch 及 case 中給予 default 值, 且寫滿所有 branch

10. Synopsys Design Constraints File (.sdc) specifies the timing constraints for synthesis. Please write the <u>***commands***</u> might be included in the ***.sdc*** file to meet the following specification. **(10pts, 1pt for each)**

2×10^8 $=0.5\times10^{-8}=5\times10^{-9}$

A. This design is an ALU with clock port **i_clk** with **200 MHz** operating frequency (1%)
B. **All flip-flops should meet the hold-time constraints** (1%)
C. Clock tree synthesis will be built in the place and route stage, so the **clock network cannot be re-buffered** (1%)
D. Clock tree synthesis will be built in the place and route stage, so the **clock network is considered with no delay** (1%)
E. The input delay is considered as **0.1*clock_cycle** (1%)
F. The output delay is considered as **0.1*clock_cycle** (1%)
G. The operating condition for synthesis should include both ***slow*** and ***fast*** library for ***max*** and ***min*** condition (1%)
H. The clock latency is considered as **0.05*clock_cycle** (1%)
I. The clock transition is considered as **0.01*clock_cycle** (1%)
J. The clock skew and jitter are considered as **0.1 ns** and **0.2ns, respectively** (1%)

These are <u>***only example commands***</u> you may use, please <u>***choose the correct ones and modify them***</u> to meet the abovementioned specification. **<u>(you do not have to consider the commands for including library)</u>**

```
1.  set period 10.0
2.  set_operating_conditions -max_library slow -max slow
3.  create_clock -name clk -period 10.0
```

```
4.  set_dont_touch_network [get_clocks clk]
5.  set_ideal_network [get_clocks clk]
6.  set_fix_hold [get_clocks clk]
7.  set_clock_latency 0.5 [get_clocks clk]
8.  set_clock_transition 0.1 [get_clocks clk]
9.  set_max_transition 0.1
10. set_input_delay 0.5 -clock clk [all_inputs]
11. set_output_delay 0.5 -clock clk [all_outputs]
12. set_clock_uncertainty 0.1 [get_ports clk]
13. set_load   1 [all_outputs]
14. set_drive   1 [all_inputs]
15. set_false_path -from {A} -through {C} -to {OUT}
16. set_max_delay 1 from [all_inputs] -to [all_outputs]
```

A. create-clock -name i-clk -period 5.0
B. set-fix-hold [get-clocks i-clk]
C. set-dont-touch-network [get-clocks i-clk]
D. set-ideal-network [get-clocks i-clk]
E. set-input-delay 0.5 -clock i-clk [all-inputs]
F. set-output-delay 0.5 -clock i-clk [all-outputs]
G. set-operating-conditions -max-library slow -max slow -min-library fast
-min fast
H. set-clock-latency 0.25 [get-clocks i-clk]
I. set-clock-transition 0.05 [get-clocks i-clk]
J. set-clock-uncertainty 0.3 [get-clocks i-clk]

11. Two situations are considered for clock uncertainty: clock skew and clock jitter. Please explain the definition of ***clock skew*** (**1pt**) and ***clock jitter*** (**1pt**).

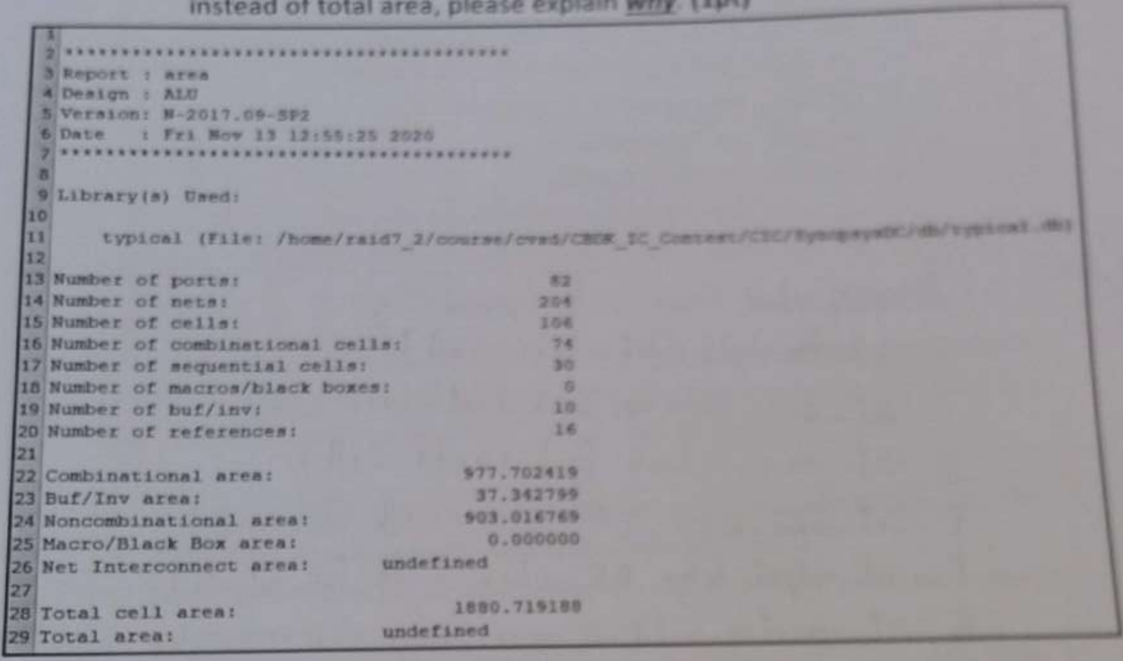
skew: (On-chip) spatial variation of the arrival time of clock
(不同 FF 間 clock 的 arrival time 可能有差異)
jitter: (off-chip) temporal variation of a clock-period in the same
Flip-Flop (同一 FF 的 cycle time 長度 可能有差異)

12. Please read the area synthesis report below and answer the related questions:

a. How to fix the ***undefined interconnect area***? (1pt)

b. If Macro/Black Box area is not zero, what might be indicated? (1pt)

c. In cell-based IC design flow, we will mainly focus on total cell area instead of total area, please explain __why__. (1pt)



```

****************************************
Report : area
Design : ALU
Version: N-2017.09-SP2
Date   : Fri Nov 13 12:55:25 2020
****************************************

Library(s) Used:

    typical (File: /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db/typical.db)

Number of ports:                              82
Number of nets:                              204
Number of cells:                             106
Number of combinational cells:                74
Number of sequential cells:                   30
Number of macros/black boxes:                  0
Number of buf/inv:                            10
Number of references:                         16

Combinational area:                   977.702419
Buf/Inv area:                          37.342799
Noncombinational area:                903.016769
Macro/Black Box area:                   0.000000
Net Interconnect area:      undefined

Total cell area:                     1880.719188
Total area:                 undefined
```

a、use "set-wire-load-model" to 估計 net/wire 的 area

b. 有使用 external IP，例如 SRAM

c、在合成階段所估計的 net/wire area 相當不準確，因此只先考慮 total cell area

13. 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. **(1pt)**

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

Path1

```
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
```

Path2

```
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      5.54 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
```

Path 1 is critical path

Since slack = required time - arrival time

slack 越小代表 timing 越緊 ⇒ 1.49 < 1.64

⇒ Path 1 is critical path