Computer-Aided VLSI System Design
Midterm Examination
2021. 11. 30

Name 陳昱仁

Student ID R10943117

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 **22** 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	24	23	Problem 6	22	22
Problem 2	10	10	Problem 7	12	12
Problem 3	12	12	Problem 8	26	25
Problem 4	12	12			
Problem 5	12	9	Total	130	125

1. Verilog (24pts)

1. You are given 4 different files and part of them are shown below. All of them represent a 7-bit adder design with 3 input **'in_1'**, **'in_2'**, **'clk'** and an output **'out'**. Determine the level of the following files. (Each block contains only one answer) **(3pts)**

	Register transfer level	Soft macro level	Gate level
File number	2	3	1

File 1:

```
......
DFFQX1 out_reg_0_ ( .D(N0), .CK(clk), .Q(out[0]) );
ADDFXL U8 ( .A(in_1[2]), .B(in_2[2]), .CI(n11), .CO(add_x_1_n2), .S(N2) );
......
```

File 2:

```
......
always @(posedge clk) out <= in_1 + in_2;
......
```

File 3:

```
......
  \**SEQGEN**  \out_reg[0]
( .clear(1'b0), .preset(1'b0), .next_state(N0), .clocked_on(clk), .data_in(
1'b0), .enable(1'b0), .Q(out[0]), .synch_clear(1'b0), .synch_preset(1'b0),
.synch_toggle(1'b0), .synch_enable(1'b1) );
  ADD_UNS_OP add_6_S2 ( .A(in_1), .B(in_2), .Z({N3, N2, N1, N0}) );
......
```

2. Please complete the following table for different **unsigned** representation <size>'<base format><number>. **(5pts)**

	size	base format	number
7'b001_1010	7	d	26
16'hx1	16	b	XXXXXXXXXXXXXXX1
9'o666	9	d	438
13	32	o	15
19'hbeaf	19	b	000 1011 1110 1010 1111

3. Please write down the corresponding Verilog code (in the left column) according to the register array description (in the right column) **(3pts)**

Verilog code	description
A register array named **MEM0** with 2048 1byte words	reg [7:0] MEM0 [0:2047];
A register array named **MEM1** with 256 1bit words	reg MEM1 [0:255];
Define a net named **A_w** whose width is 1 byte. Fetch a word in **MEM0** whose address is 345 and assign the value to **A_w**.	wire [7:0] A_w; assign A_w = MEM0[345];

4. Please explain the definitions of **1ns** and **100ps** in the following Verilog code. **(2pts)**

```
`timescale 1ns/100ps
```

1ns：simulation 時用到的 delay 相關時間資訊的單位，如：#2 為 delay 2ns

100ps：最小的時間精準度，即最小的一單位為 100ps 不能再更小

Simulation Period

5. Which of the following codes are synthesizable? Fill T if synthesizable or F if not. (5pts)

Codes	Synthesizable

`wire [7:0] A;`
`wire [2:0] B,C;`
`assign A = B * (C + 1'b1);`
	T

`reg [3:0] A;`
`initial begin`
`  A = 4'd3;`
`end`
	F

`reg [2:0] A,B;`
`always @(posedge clk) begin`
`  #(5) B = ~A;`
`end`
	F

`reg [2:0] A,B;`
`always @(*) begin`
`  B = A << 2;`
`end`
	T

`wire [2:0] A;`
`reg  B;`
`always @(*) begin`
`  B = &A;`
`end`
	T

6. What kinds of the data type of the variables in the following Verilog code can be? You can ONLY answer with **reg**, **wire** or **reg/wire**. **(2pts)**

```
// variable declaration
... A;
... B;
... C;
... D;

assign A = B;

always @ (*) begin
    C = D;
end
```

A: wire
B: reg/wire
C: reg
D: reg/wire

7. The following codes are implementing a signed adder with a 4-bit signed input, 3-bit signed input and an 8-bit signed output. Please complete the two coding styles below. **(2pts)**

Explicit sign extension

```
wire [3:0] A;    4 bits
wire [2:0] B;    3 bits.
wire [7:0] C;  8 bits

assign C = {{4{A[3]}}, A} + {{5{B[2]}}, B};
```

$signed()

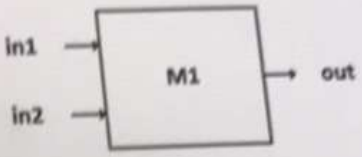
```
wire [3:0] A;
wire [2:0] B;
wire [7:0] C;

assign C = $signed(A) + $signed(B);
```

8. Please complete th
according to the
circuit behavior.

```
module M1 (
    input
    inpu
    ou
```

8. Please complete the module instantiation of the following Verilog code according to the specified port declaration method and description of the circuit behavior. **(2pts)**

```
module M1 (in2, in1, out);
    input  [1:0] in1;
    input  [1:0] in2;
    output [3:0] out;

    assign out = (in1 * 2) + in2;

endmodule
```



Connect module ports by order list

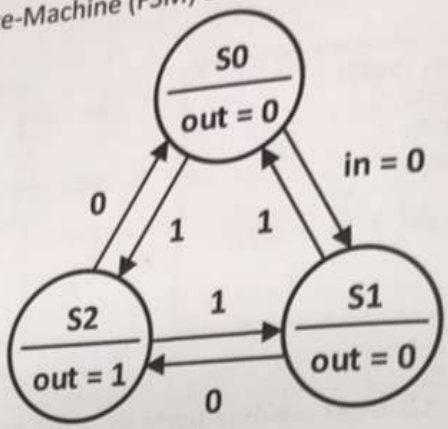
```
wire [1:0] A, B;
wire [3:0] C;

// circuit behavior: C = 2A + B
M1 inst_0( B , A , C );
```

Connect module ports by name

```
wire [1:0] A, B;
wire [3:0] C;

// circuit behavior: C = 2B + A
M1 inst_1( .in2( A ), .in1( B ), .out( C ));
```

2. Finite State Machine and Simulation (10pts)

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

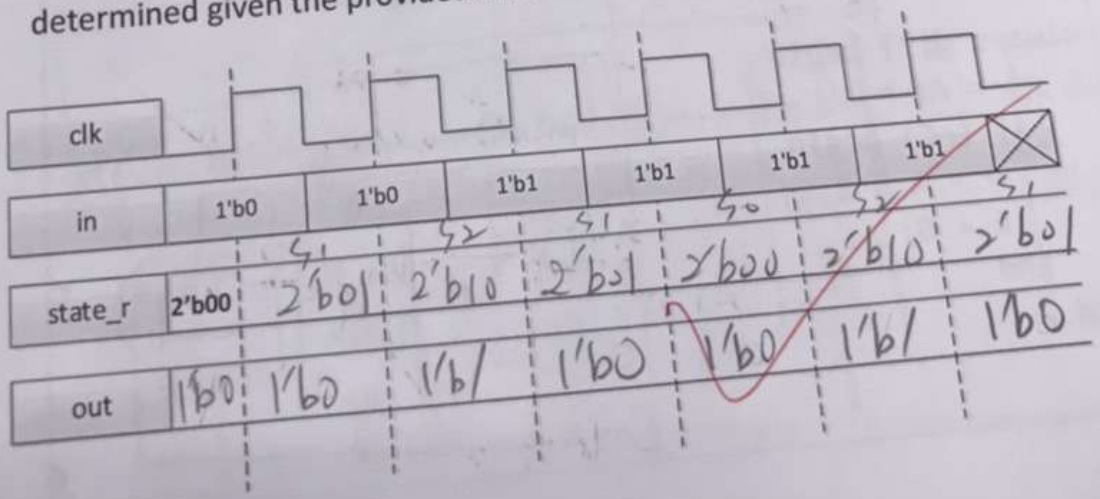


(a) Please complete the Verilog code below according to the FSM in the above figure. **Your code should be synthesizable and the inferred latch is not allowed. (8pts)**

```
module FSM (clk, rst_n, in, out);
    input  clk, rst_n, in;
    output out;
    parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;

    reg [1:0] state_r, state_w;

    // (1) next state logic
    always @ (*) begin
        case (state_r)
            S0 : begin
                 if(in) state_w = S2;
                 else   state_w = S1; end
            S1 : begin
                 if(in) state_w = S0;
                 else   state_w = S2; end
            S2 : begin
                 if(in) state_w = S1;
                 else   state_w = S0; end
            default: state_w = S0;
        endcase
    end
```

```
// (2) output logic
assign out = (state_r == S0) ? 0 :
             (state_r == S1) ? 0 :
             (state_r == S2) ? 1 : 0;

// (3) current state
// please use synchronous reset
always @( posedge clk ) begin
    if (~rst_n) begin
        state_r <= S0;
    end
    else begin
        state_r <= state_w;
    end
end

endmodule
```

(b) Please show the waveform for the circuit in part A. Suppose that the signal value **state_r** is reset to **"S0"** 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. **(2pts)**

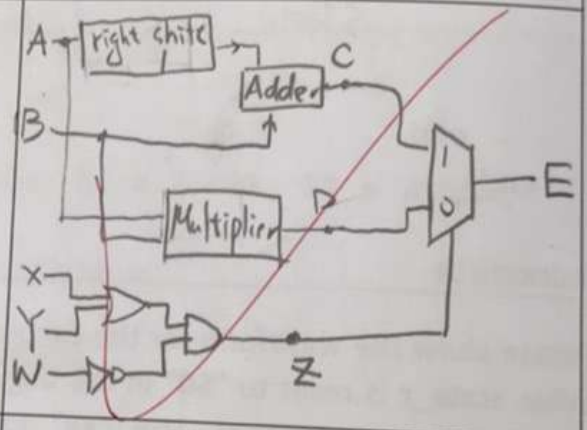


3. Translating RTL to Circuits (12 pts)

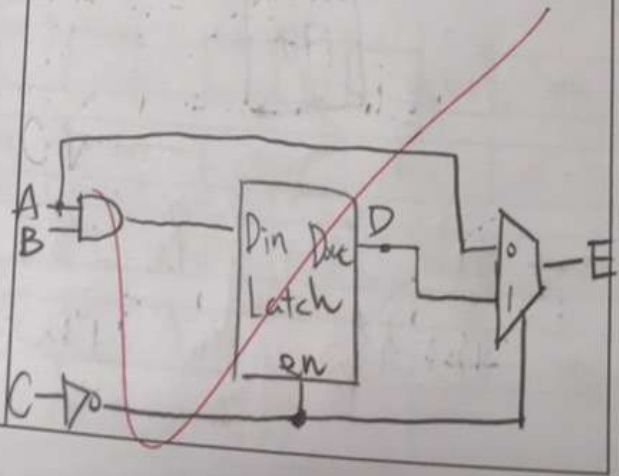
Please draw the corresponding circuits (in the right column) according to the Verilog codes (in the left column). You can use AND, OR, NAND, NOR, XOR, XNOR, NOT, MUX, D Flip-Flop, Latch, Shifter, Adder, Multiplier in the circuit diagram.

Please specify the input, output, and clock if **DFF** is used. And the input, output and enable if **Latch** is used. For example:

(a) Verilog Code (3 pts)	Circuit Diagram

`always @(*) begin`
`  C = (A >> 1) + B;`
`  D = A * B;`
`  Z = (X | Y) & (~W);`
`  E = Z ? C : D;`
`end`
	

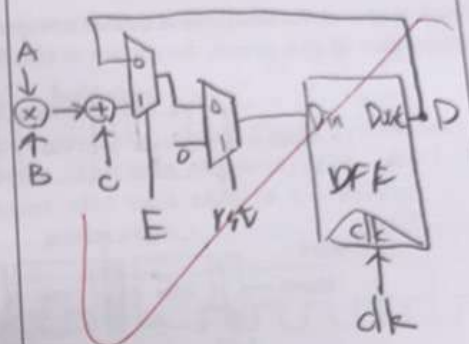
(b) Verilog Code (3 pts)	**Circuit Diagram**

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

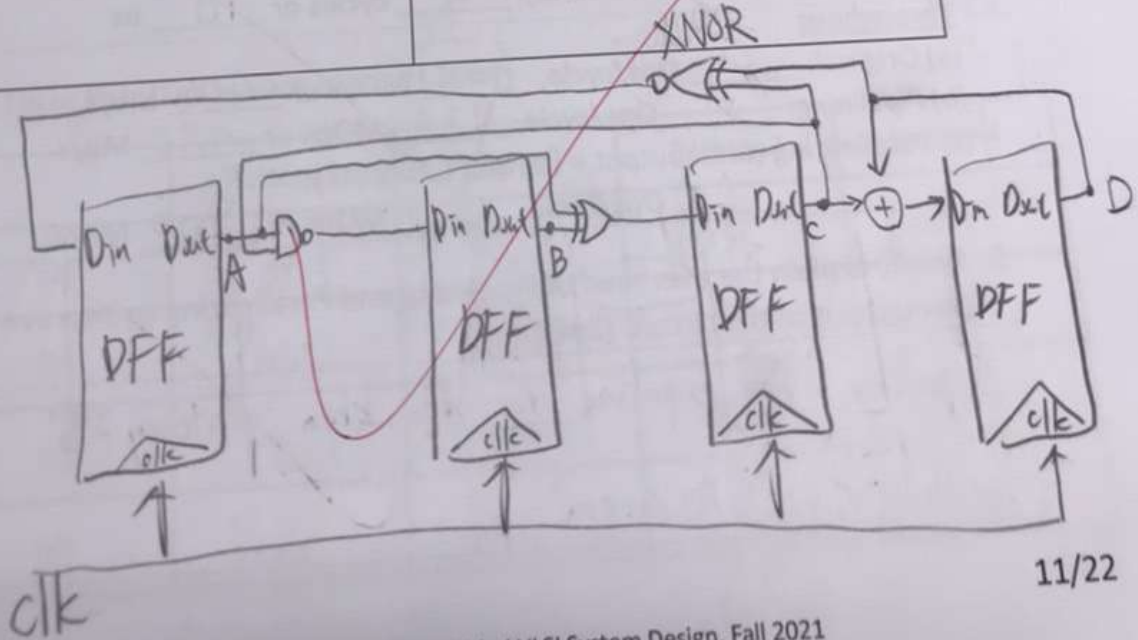
right column) according to t
OR, NAND, NOR, XOR, XNO
n the circuit diagram.

he input, output and

(c) Verilog Code (3pts)	Circuit Diagram

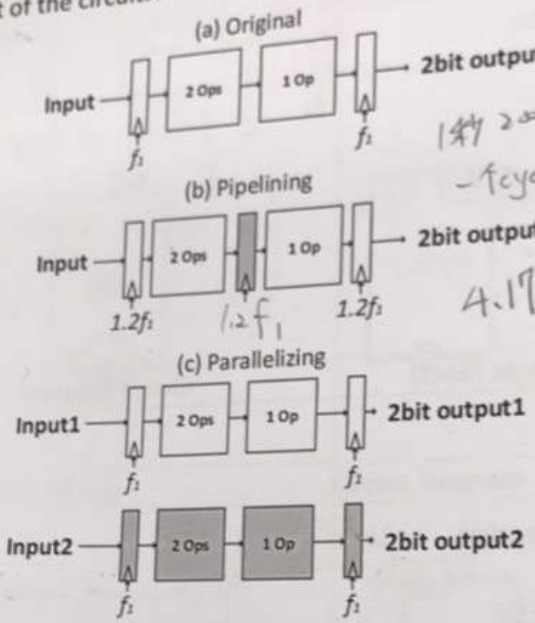
always @(posedge clk)
begin
if(rst) begin
D <= 0;
else begin
if (E)
D <= A * B + C;
end
end
	

(d) Verilog Code (3pts)	Circuit Diagram
always @(posedge clk) begin A <= D ~^ C; B <= ~(A & C); C <= B ^ A; D <= D + C; end	



4. Architecture Improvement (12pts)

Pipelining and Parallelizing are commonly used design techniques to improve the throughput of the circuit. As shown in the figure below:



A. Please determine the **latency** and **throughput** for (a) Original, (b) Pipelining and (c) Parallelizing. (Assuming f_1 = 200 MHz) **(10pts)**

Latency

(a) Original: 2 cycles or 10 ns

(b) Pipelining: 3 cycles or 12.51 ns

(c) Parallelizing (for each output): 2 cycles or 10 ns

Throughput

(a) Original: 3 Ops/cycle, 600 MOps or 400 Mb/s

(b) Pipelining: 3 Ops/cycle, 720 MOps or 480 Mb/s

(c) Parallelizing (total output = {output1,output2}):
6 Ops/cycle, 1200 MOps or 800 Mb/s

B. Briefly explain the overhead of Pipelining and Parallelizing to improve the throughput of the circuit. **(2pts)**

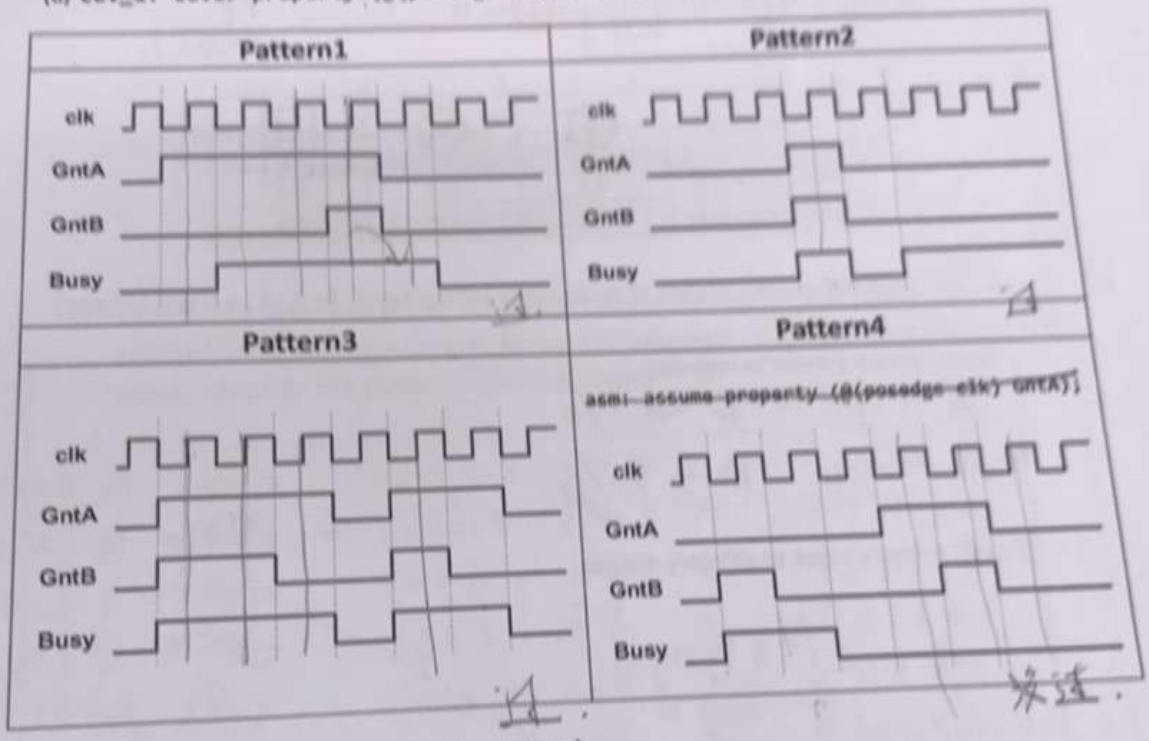
Pipeline: 增 pipeline register area, latency 增

Parallelizing: 增加 area

5. Formal Verification (12pts)

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

(a) `assert_0: assert property (@(posedge clk) GntA == 0 |-> !Busy);`
(b) `assert_1: assert property (@(posedge clk) GntA || GntB |-> ##[0:2] Busy);`
(c) `cov_0: cover property (@(posedge clk) GntA && !GntB);`
(d) `cov_1: cover property (@(posedge clk) GntA && GntB |=> Busy);`

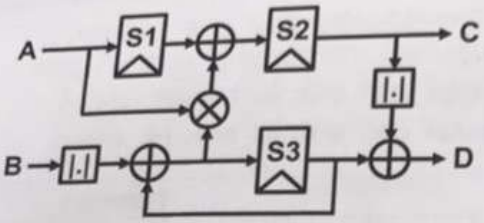


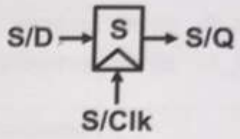
Fill in **Pass/Fail** in each block. **(12pts)**

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

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

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





Operation	⊕	⊗	|·|
Min Delay (ns)	0.5	3.0	0.2
Max Delay (ns)	1.0	4.0	0.5

A. List all timing path under the corresponding type. (e.g. A -> S1/D) **(5pts)**

Type1: primary input to register

A → S1/D B → S2/D

A → S2/D B → S3/D

Type2: primary input to primary output

Nothing.

Type3: register to register

S1/clk → S2/D S3/clk → S2/D

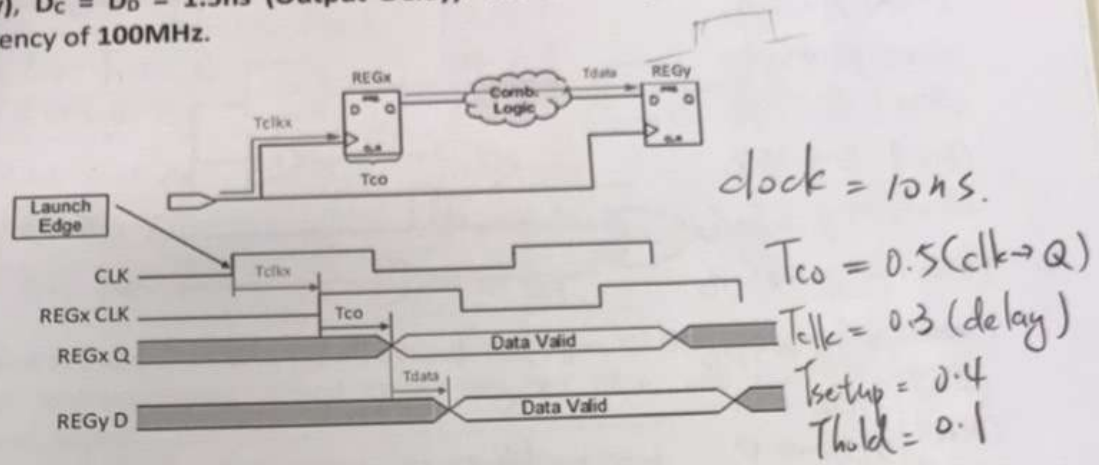
S3/clk → S3/D

Type4: register to primary output

S2/clk → C S3/clk → D

S2/clk → D

Following the circuit in **6.A**, the registers' timing diagrams are shown below. **T_{co} = 0.5ns**, and **T_{clkx} = 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 = 1.5ns (Input Delay)**, **D_C = D_D = 1.5ns (Output Delay)**. The circuit operates at the clock frequency of **100MHz**.



clock = 10ns.

T_{co} = 0.5 (clk → Q)

T_{clk} = 0.3 (delay)

T_{setup} = 0.4

T_{hold} = 0.1

B. Check that whether there are setup time violations or not by showing if the related inequalities of each paths are satisfied. If setup time violation(s) occur, identify the violated path(s). **(5pts)**

		RT		AT
(Met)	A → S1/D	10 + 0.3 − 0.4	>	1.5
(Met)	A → S2/D	10 + 0.3 − 0.4	>	1.5 + 4 + 1
(Met)	B → S2/D	10 + 0.3 − 0.4	>	1.5 + 0.5 + 1 + 4 + 1
(Met)	B → S3/D	10 + 0.3 − 0.4	>	1.5 + 0.5 + 1
(Met)	S1/clk → S2/D	10 + 0.3 − 0.4	>	0.3 + 0.5 + 1
(Met)	S2/clk → S2/D	10 + 0.3 − 0.4	>	0.3 + 0.5 + 1 + 4 + 1
(Met)	S3/clk → S3/D	10 + 0.3 − 0.4	>	0.3 + 0.5 + 1
(Met)	S2/clk → C	10 + 1.5	>	0.3 + 0.5
(Met)	S2/clk → D	10 − 1.5	>	0.3 + 0.5 + 0.5 + 1
(Met)	S3/clk → D	10 − 1.5	>	0.3 + 0.5 + 1

No setup violation time.

C. Check that whether there are hold time violations or not by showing if the related inequalities of each paths are satisfied. If hold time violation(s) occur, identify the violated path(s). **(5pts)**

		RT		AT
(Met)	A→S1/D	0.3+0.1	<	1.5
(Met)	A→S2/D	0.3+0.1	<	1.5+3+0.5
(Met)	B→S2/D	0.3+0.1	<	1.5+0.2+0.5+3+0.5
(Met)	B→S3/D	0.3+0.1	<	1.5+0.2+0.5
(Met)	S1/clk→S2/D	0.3+0.1	<	0.3+0.5+0.5
(Met)	S1/clk→S2/D	0.3+0.1	<	0.3+0.5+0.5+3+0.5
(Met)	S1/clk→S3/D	0.3+0.1	<	0.3+0.5+0.5
(Met)	S2/clk→C	-1.5	<	0.3+0.5
(Met)	S2/clk→D	-1.5	<	0.3+0.5+0.2+0.5
(Met)	S3/clk→D	-1.5	<	0.3+0.5+0.5

ans here

☆ No hold time violation #1

D. If the circuit operates at the clock ~~frequency of **125MHz**, and other~~ **conditions are remained the same**. Explain if there are timing violations in this circuit. If yes, identify the violated path(s). **(7pts)**

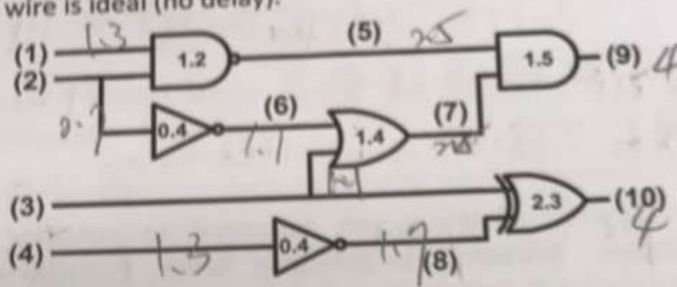
clock 10ns → 8ns, clock 只影響 setup time, AT不改

		RT		AT
(Met)	A→S1/D	8+0.3-0.4	>	1.5
(Met)	A→S2/D	8+0.3-0.4	>	1.5+4+1
(violate)	B→S2/D	8+0.3-0.4	>	1.5+0.5+1+4+1
(Met)	B→S3/D	8+0.3-0.4	>	1.5+0.5+1
(Met)	S1/clk→S2/D	8+0.3-0.4	>	0.3+0.5+1
(Met)	S1/clk→S2/D	8+0.3-0.4	>	0.3+0.5+1+4+1
(Met)	S1/clk→S3/D	8+0.3-0.4	>	0.3+0.5+1
(Met)	S2/clk→C	8-1.5	>	0.3+0.5
(Met)	S2/clk→D	8-1.5	>	0.3+0.5+0.5+1
(Met)	S3/clk→D	8-1.5	>	0.3+0.5+1

☆ B→S2/D violate setup time violation, no hold time violation

7. Slack graph (12pts)

The figure below shows a combinational circuit with primary inputs. The delay of each input is 0.2ns and the delay of every gate is given on the gate (unit: ns). Assume the wire is ideal (no delay).



(a) Given the required time for all outputs are 4.0ns, fill the blanks on the form with corresponding arrival time, required time, and slack of every node.**(10pts)**

AT = Arrival Time, RT = Required Time.

Node	(1)	(2)	(3)	(4)	(5)
AT (ns)	0.2	0.2	0.2	0.2	1.4
RT (ns)	1.3	0.7	1.1	1.3	2.5
Slack (ns)	1.1	0.5	0.9	1.1	1.1
Node	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**
AT (ns)	0.6	2.0	0.6	3.5	2.9
RT (ns)	1.1	2.5	1.7	4	4
Slack (ns)	0.5	0.5	1.1	0.5	1.1

(b) Please indicate the critical path and show its slack values. **(2pts)**

(2) → (6) → (7) → (9)

slack = 0.5 (ns)

8. Synthesis (26pts)

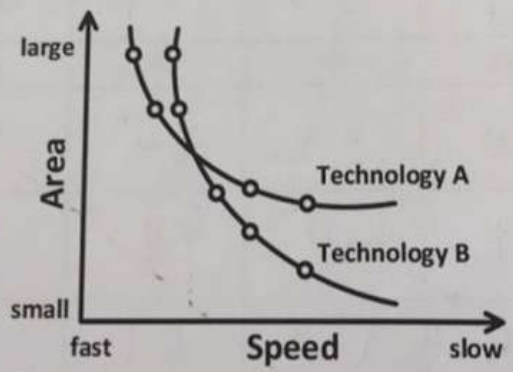
1. Please explain what is **PVT condition** considered in the synthesis. **(1pt)**

在不同的 process, voltage, temperature, cell 的速度不同，需將欲設定之 condition (PVT)，告訴 DC 現在合成的 operation condition

2. For synthesis, designers are supposed to consider different **PVT conditions**. Assuming under the same P, please choose the corresponding V and T states for slow and fast corners, respectively. **(2pts)**

Condition	V	T
Slow	High / (Low)	(High) / Low
Fast	(High) / Low	High / (Low)

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

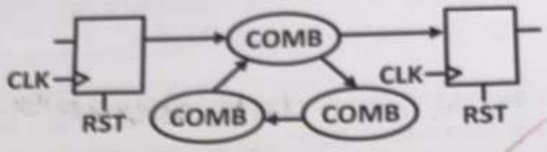


constraint-driven 是指給定製程下，依 designer 要 area 或 timing 或 power 優先，告知 DC，DC 會往此設定進行合成

給定製程下，速度越快，面積越大，速度越慢，面積越小

面積和速度為負相關

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



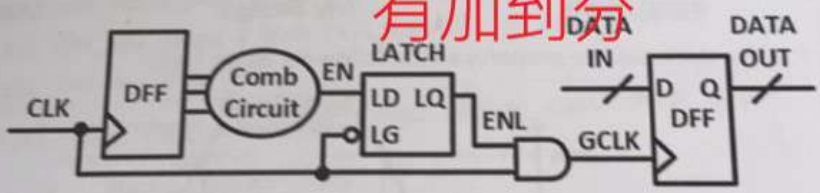
AT有此情形

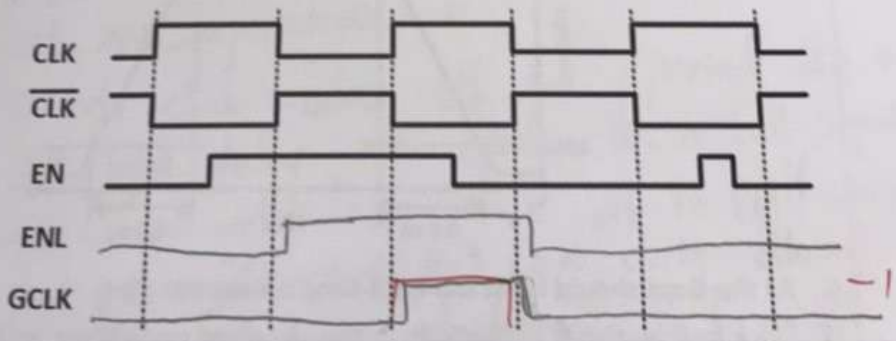
Combinatial loop無法使STA分析之AT穩定，會持續上升

在RTL時，就不能寫出combinatial loop，可用register切開此loop

5. Clock gating is a common measure for reducing dynamic power dissipation. Please explain what is the function of the **LATCH** in the following circuit figure. **(1pt)** And also **plot the waveforms of ENL and GCLK right on the figure**. **(2pts)**

有加到分





-1

EN是combinatial loop之output，故可能會在運算時產生glitch，而使用反向clk的Latch，可以過濾EN，使這種不想要的glitch消除，以免GCLK有glitch

6. Synopsys Design Constraints File (.sdc) specifies the design rule and optimization constraints for synthesis. Please write the **commands** might be included in the **.sdc** file to meet the following specification. The default unit for time is ***ns***, for capacitance is ***pf***. **(13pts)**

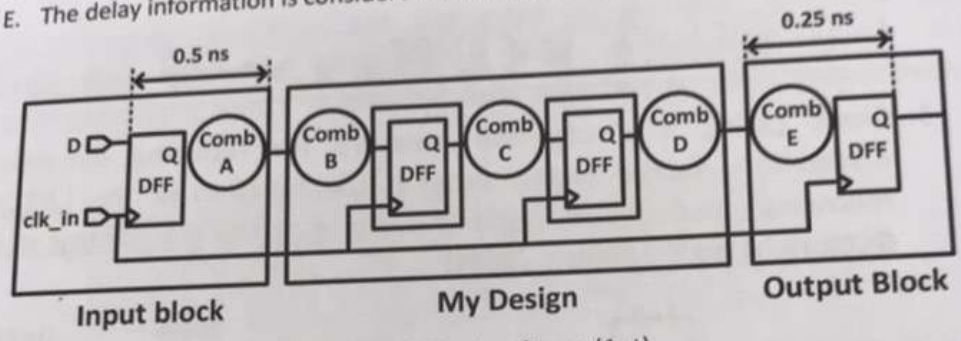
A. The operating condition for synthesis should include both ***slow*** and ***fast*** library for ***max*** and ***min*** condition (1pt)

B. The longest time to drive a pin and change its value is **0.2 ns** (1pt)

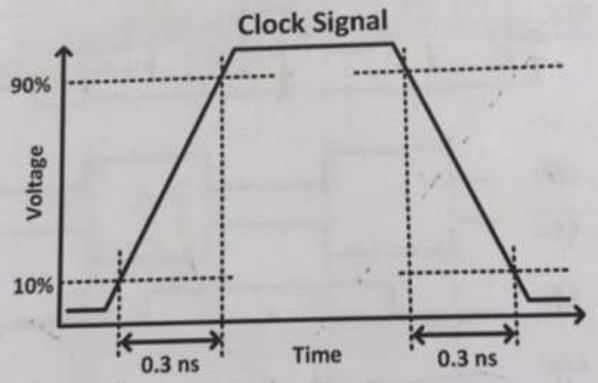
C. The maximum capacitive load that an output pin can drive is **0.3 pf** (1pt)

D. This design is an ALU with a clock port **clk_in** with **50 MHz** operating frequency (1pt)

E. The delay information is considered as the following figure (2pts)



F. The clock has the property as the following figure (1pt)



G. All **flip-flops should meet the hold-time constraints** (1pt)

H. Clock tree synthesis will be built in the place and route stage, so the **clock network cannot be re-buffered** (1pt)

I. Clock tree synthesis will be built in the place and route stage, so the **clock network is considered with no delay** (1pt)

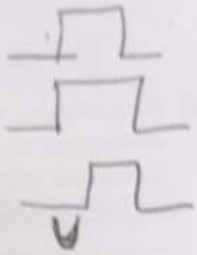
J. The clock latency is considered as **0.02*clock_cycle** (1pt)

K. The clock skew and jitter are considered as **0.2 ns** and **0.3 ns** (1pt)

L. The optimization goal is to **reduce the area as small as possible** (1pt)

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

```
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_max_capacitance 0.1
11. set_input_delay 0.5 -clock clk [all_inputs]
12. set_output_delay 0.5 -clock clk [all_outputs]
13. set_clock_uncertainty 0.1 [get_ports clk]
14. set_load    0.5  [all_outputs]
15. set_drive   0.5  [all_inputs]
16. set_false_path -from {A} -through {C} -to {OUT}
17. set_max_delay 1 from [all_inputs] -to [all_outputs]
18. set_max_area 10
```



A. set_operating_conditions -max_library slow -max slow -min_library fast -min fast
B. set_max_transition 0.2
C. set_max_capacitance 0.3
D. create_clock -name clk_in -period 20.0
E. set_input_delay 0.5 -clock clk_in [all_inputs]
   set_output_delay 0.5 -clock clk_in [all_outputs]
F. set_clock_transition 0.3 [get_clocks clk_in]
G. set_fix_hold [get_clocks clk_in]
H. set_dont_touch_network [get_clocks clk_in]
I. set_ideal_network [get_clocks clk_in]
J. set_clock_latency 0.4 [get_clocks clk_in]
K. set_clock_uncertainty 0.5 [get_ports clk_in]
L. set_max_area 0

7. Please explain the meaning of **standard delay file (.sdf) (1pt)**, and 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);
```

.sdf 為合成後，你的RTL design的gate level檔的時間資訊(.sdf是timing檔，net list在.v)

因要把正確 timing info. 加入 simulation 才正確，否則他會拿 tsmc.v 的 timing，但 tsmc.v 的 timing 都是 default，非你 design 的 timing

8. 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.db → 使 non-critical path power 減少

rvt.db → 用在 critical path，使 timing 跑較快