Problem 1 <Design Flow>

The following items are the steps in the cell-base design flow.

(a) DRC & LVS Verification; (b) DFT Insertion; (c) RTL Coding; (d) Place & Route; (e) Tape Out; (f) Synthesis; (g) Specification.

(6%) A. Please give the correct order of these steps.

Ans:

6

g
↓
c
↓
f
↓
b
↓
d
→ a
↓
e

(4%) B. Besides taping out, which ones belong to the back-end part of the design flow?

Ans:

d, a, e

2

Problem 2 <Basic Concept>

4 (4%) A. Give one feature of SystemVerilog for design verification.

Ans:

Object-oriented programming, 提供class

3 (3%) B. What kind of information does an SDF file give?

Ans:

timing information

(4%) C. What is a critical path?

3 **Ans:**

path delay 最長的 path

Input → FF

FF → FF

FF → output

(4%) D. What is setup time violation of a flip-flop?

Ans:

4

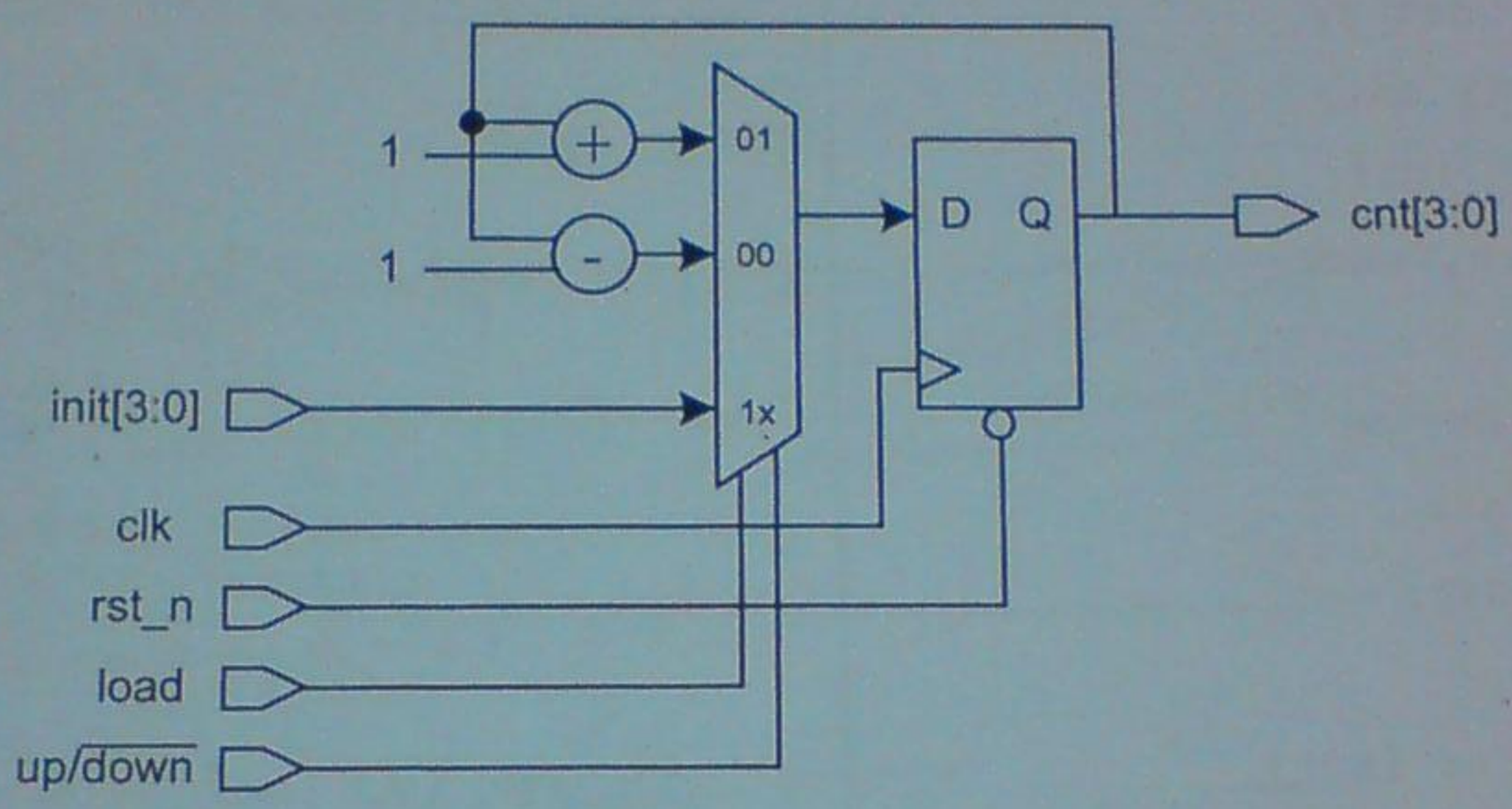
setup-time is the minimum time before the clocking event by which the input must be stable.

故 setup time violation 就是運算的 delay 太久，無法滿足

是 $t_{cq} + t_{logic} + t_{su} < T_{clk}$.

Problem 3 <Verilog Debug and Coding>

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



(10%) The following Verilog-RTL code for the counter is not correct. Please write the correct version in the right column.

8

Verilog code with bugs	Corrected Code
`module counter(clk, rst_n, cnt,`	module counter (clk, rst_n, cnt,
`init, load, updown)`	init, load, updown);
`input clk, rst_n;`	
`output [3:0] cnt;`	
`input [3:0] init;`	
`input updown, load;`	
`wire [3:0] cnt_w;`	reg [3:0] cnt_w;
`reg [3:0] cnt;`	
`reg [3:0] init;`	wire [3:0] init;
`always@(cnt or init)`	always@(cnt or init or load or updown)
`begin`	
`  case({load, updown})`	
`  2'b00: cnt_w = cnt - 1'b1;`	
`  2'b01: cnt_w = cnt++;`	2'b01: cnt_w = cnt + 1'b1;
`  2'b10: cnt_w = init;`	default: cnt_w = cnt;
`  endcase`	
`end`	
`always@(posedge clk or rst_n)`	always@(posedge clk or negedge rst_n)
`begin`	
`  if(~rst_n)`	
`  cnt <= 4'd0;`	
`  else`	
`  cnt <= cnt_w;`	
`end`	
`endmodule`	

There are totally ~~7~~ 8 bugs.

Problem 4 <Verilog Simulation>

(12%) Given the following code written with Verilog-2001, please show the outputs if this code is simulated via a Verilog simulator.

```
`timescale 1ns/1ns
module test;
reg signed [7:0] x, a;
reg [7:0] y, b;

initial begin
x= -8'd6; y=8'd6;
 #5 y=8'd255;
 fork
  #15 a = x >>> (-2'b1);
  #5  b = x >> (-2'b1);
  begin
   #10 y <= x;
       x <= y;
  end
 join
 #20 x[4 +: 4] = 0;
  #5 $finish;
 end

 initial begin
  $monitor("%t %d %d %d %d",$time, x, y, a, b);
 end
 endmodule
```

110

1010

2

4 5 6 7

11111111

8421

Ans:

2

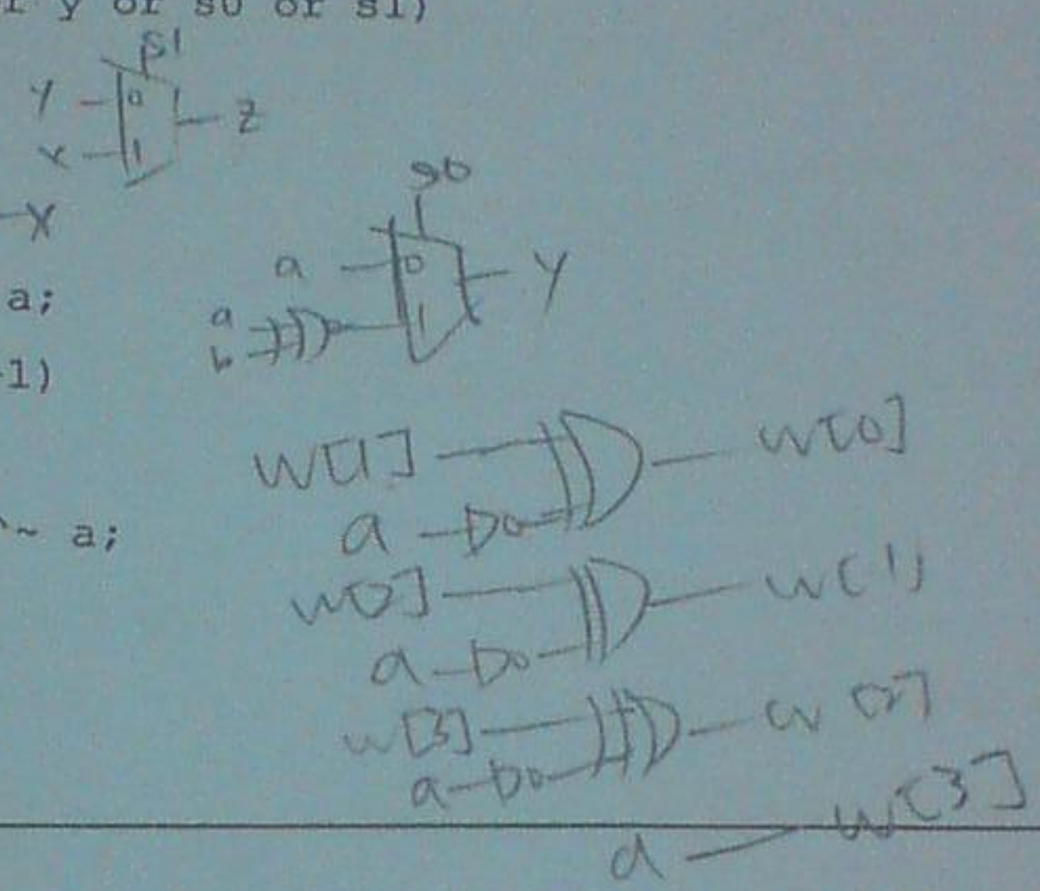
$time	x	y	a	b
0	~~6~~	6	x	x
5	~~6~~	255	x	x
10	~~6~~	255	x	~~1~~
15	~~255~~	~~6~~	x	~~1~~
20	~~255~~	~~6~~	~~255~~	~~1~~
40	15	~~6~~	~~255~~	1

Problem 5 <Manually Synthesis from Verilog RTL Code>

(12%) The following block shows a procedural block of Verilog RTL code. Please draw the corresponding circuits in the bottom block. You can use AND, OR, NAND, NOR, XOR, XNOR, NOT, MUX in the circuit diagram.

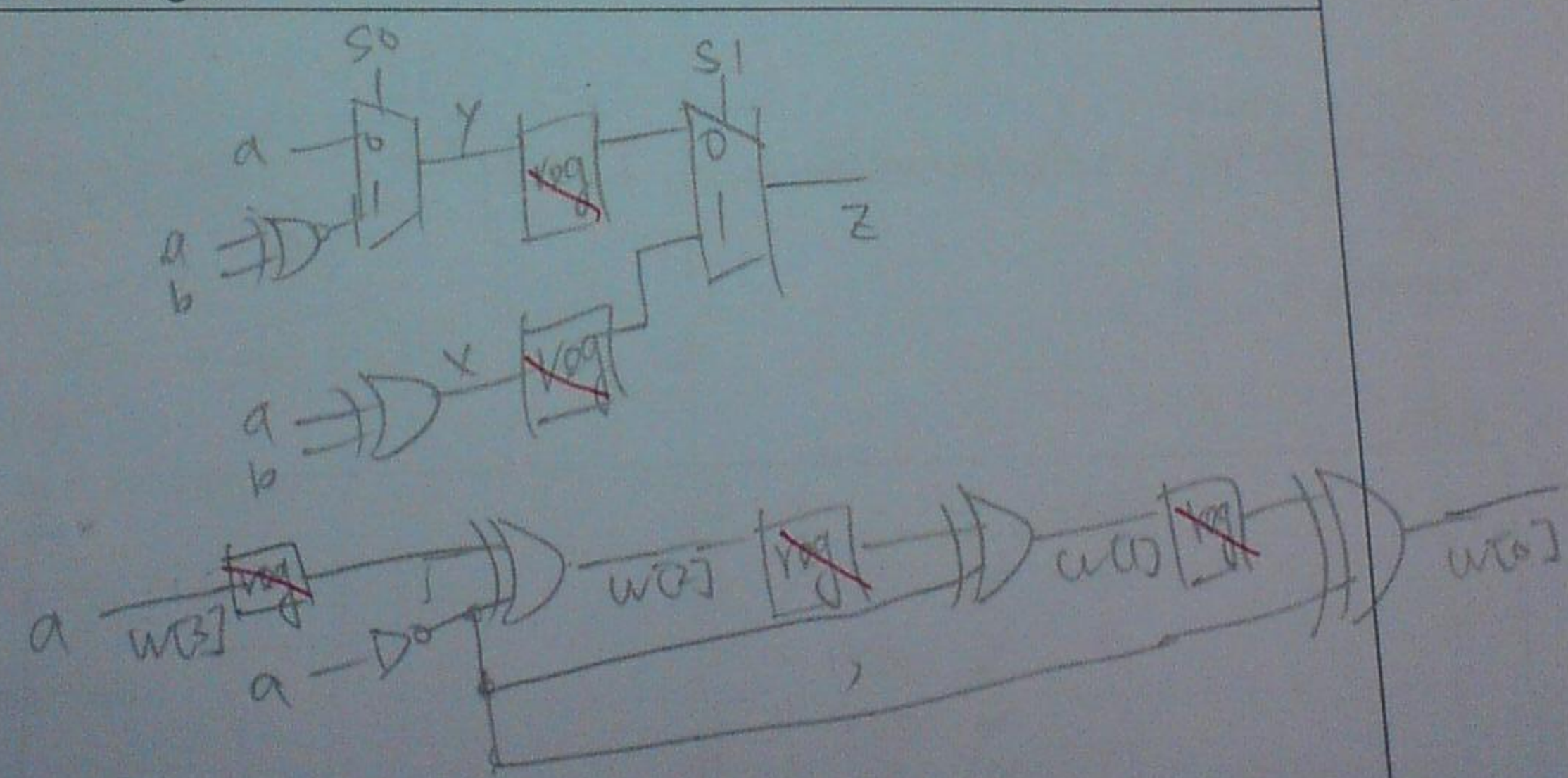
Verilog Code

```
wire a, b, s0, s1;
reg [3:0] w;
reg x, y, z;
integer i;
always@(a or b or x or y or s0 or s1)
begin
   z = (s1)? x: y;
   x = a + b;
   y = (s0)? (a==b): a;
   for(i=0; i<3; i=i+1)
   begin
      w[i] = w[i+1] ^~ a;
   end
   w[3] = a;
end
```



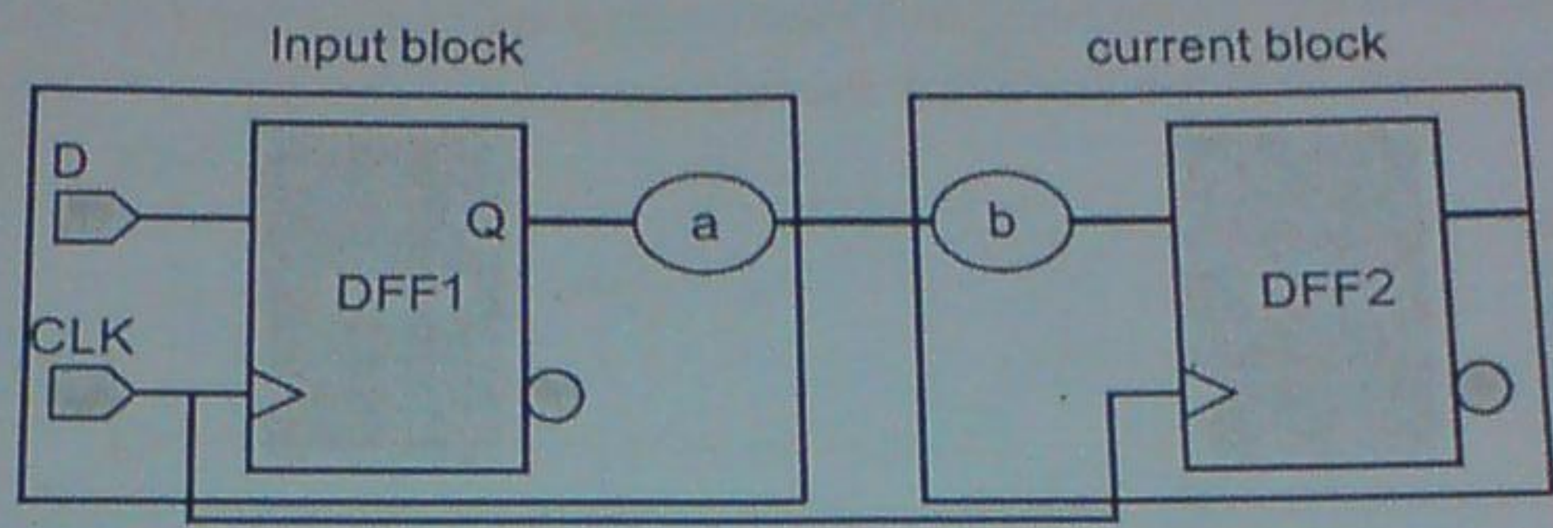
Circuit Diagram

10



Problem 6 <Input Delay>

(5%) B. Please specify the **input delay** of the **current block** in the following circuit.



5

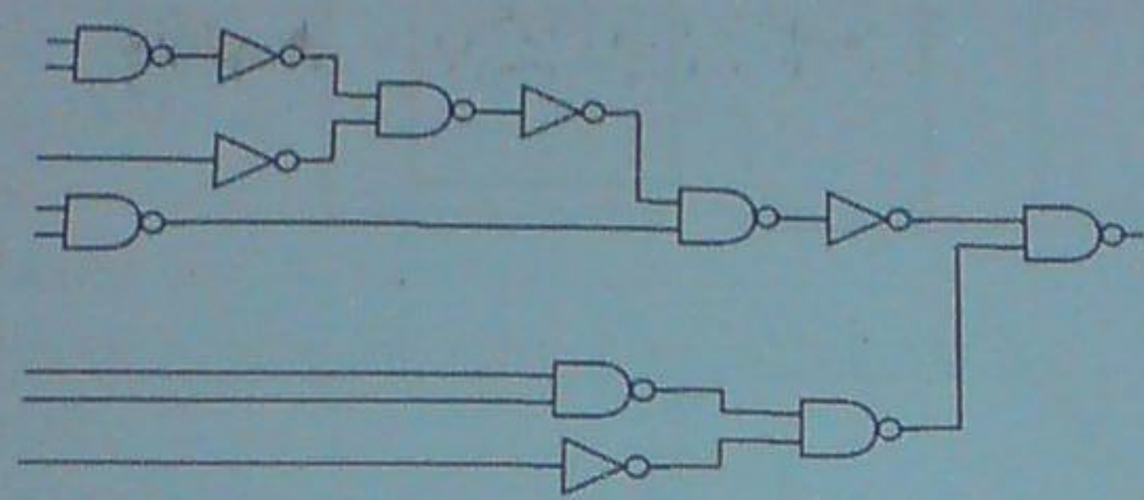
Ans:

$t_{cq} + t_a$

Problem 7 <DAGON algorithm>

(12%) Perform technology mapping using the DAGON algorithm to compute a *minimum-cost cover* for the following subject graph using the given library.
(In Phase 1, write down the optimal costs and their corresponding gate types for each gate in the subject graph. In Phase 2, identify the minimum-cost cover.)

Subject graph:

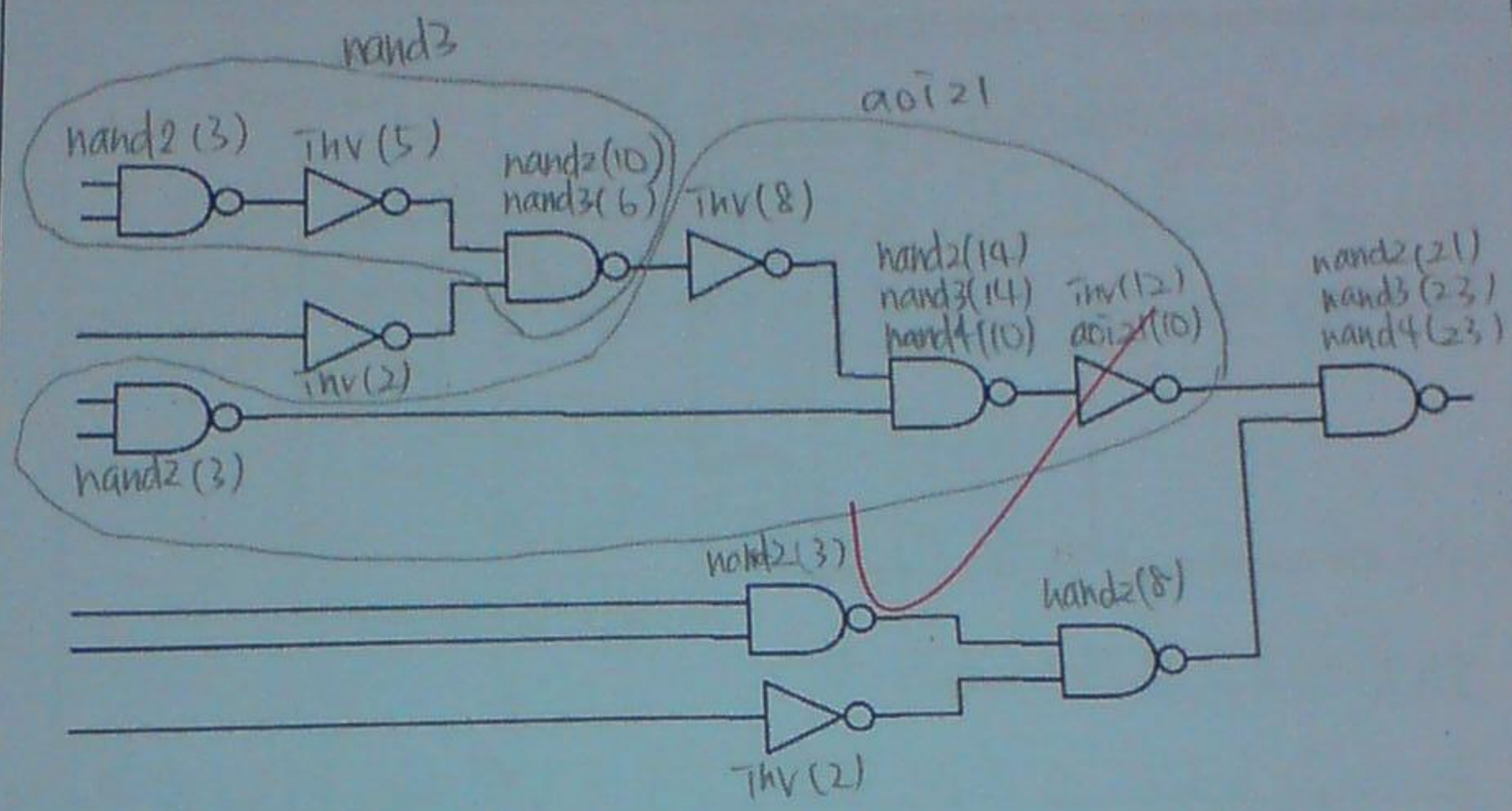


Library:

cost	gate type		pattern graph
2		inv	
3		nand2	
4		nand3	
5		nand4	
4		aoi21	
5		aoi22	

Ans:

12


nand3
aoi21
nand2 (3)
Inv (5)
nand2(10)
nand3(6)
Inv(8)
nand2(14)
nand3(14)
nand4(10)
Inv(12)
aoi21(10)
nand2(21)
nand3(23)
nand4(23)
Inv(2)
nand2(3)
nand2(3)
nand2(8)
Inv(2)

Problem 8 <Design for Testability>

(9%) A. Concepts of DfT. Please answer the following questions briefly.

9

1) Explain the meaning of "Fault Simulation".

Ans:

Insert fault 後做 logic simulation，
藉由檢查 faulty 的 output 有沒有等於 fault-free 的 output 來
得知 imply 的 test vector 能不能 detect 到該 insert 的 fault.

2) Explain the meaning of "Fault Modeling".

Ans:

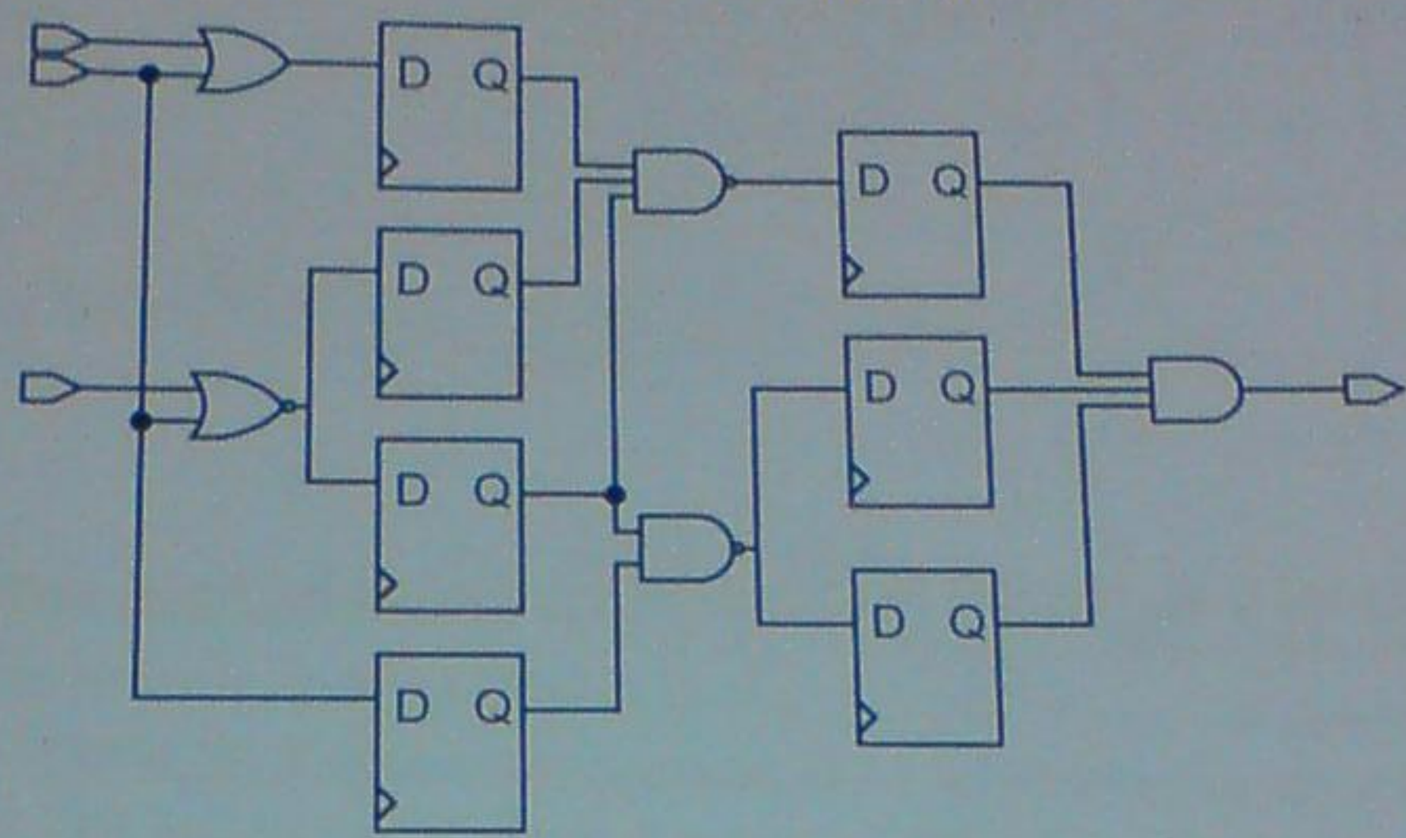
描象化表示實体的 defect，
如 single stuck-at fault 用來表示實体电路 short 到 V_d 或地，
其它如 bridging fault 則是用來表示實体电路兩條線 short 在一起

3) Explain the meaning of "Fault Collapsing".

Ans:

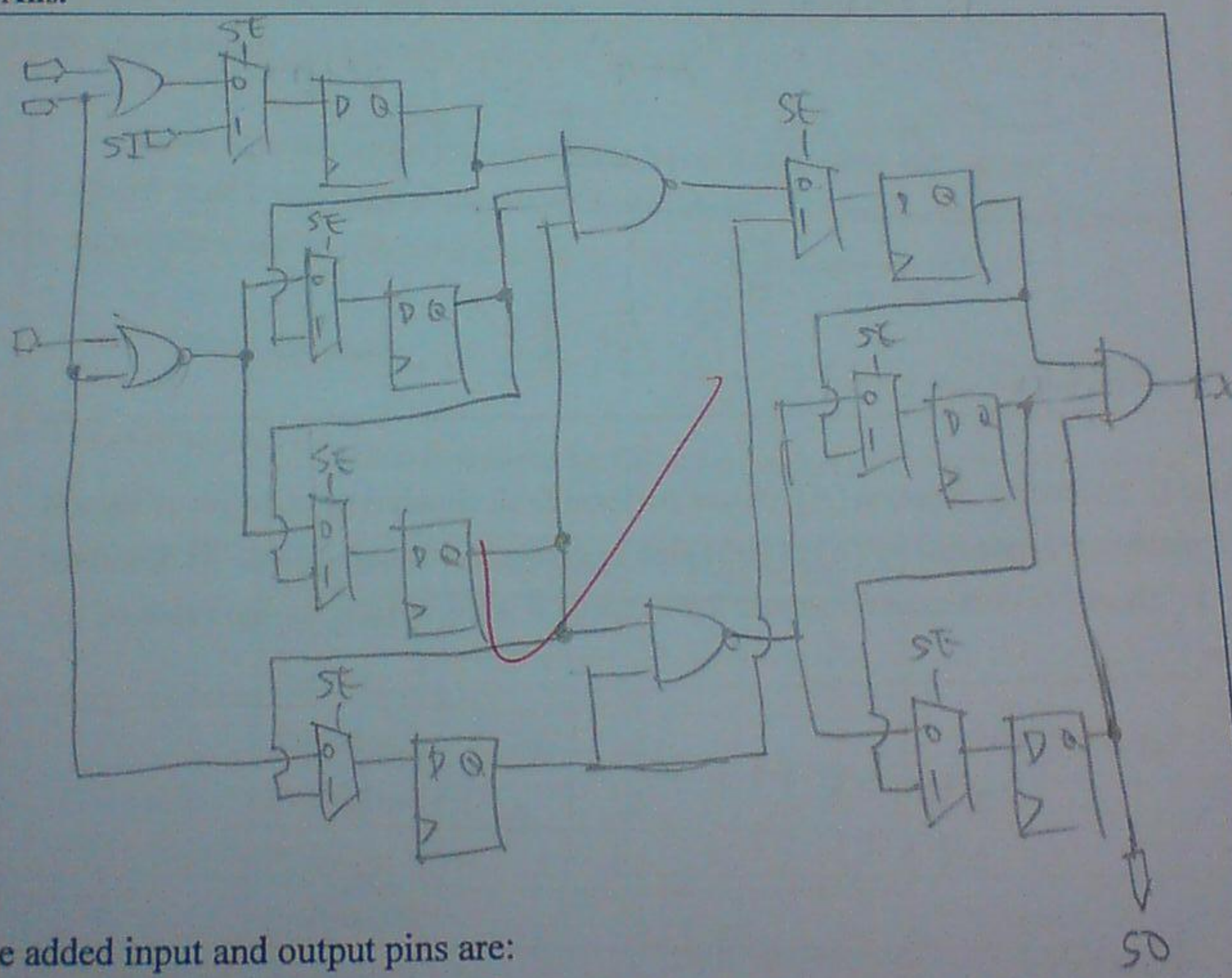
將 equivalent fault 刪掉，
藉此可減少 fault simulation 或 ATPG 的 run time

(5%) B. For the following circuits, please draw the associated circuits with full-scan with multiplexed flip-flop. Show the input and output pin added with full-scan.



5

Ans.

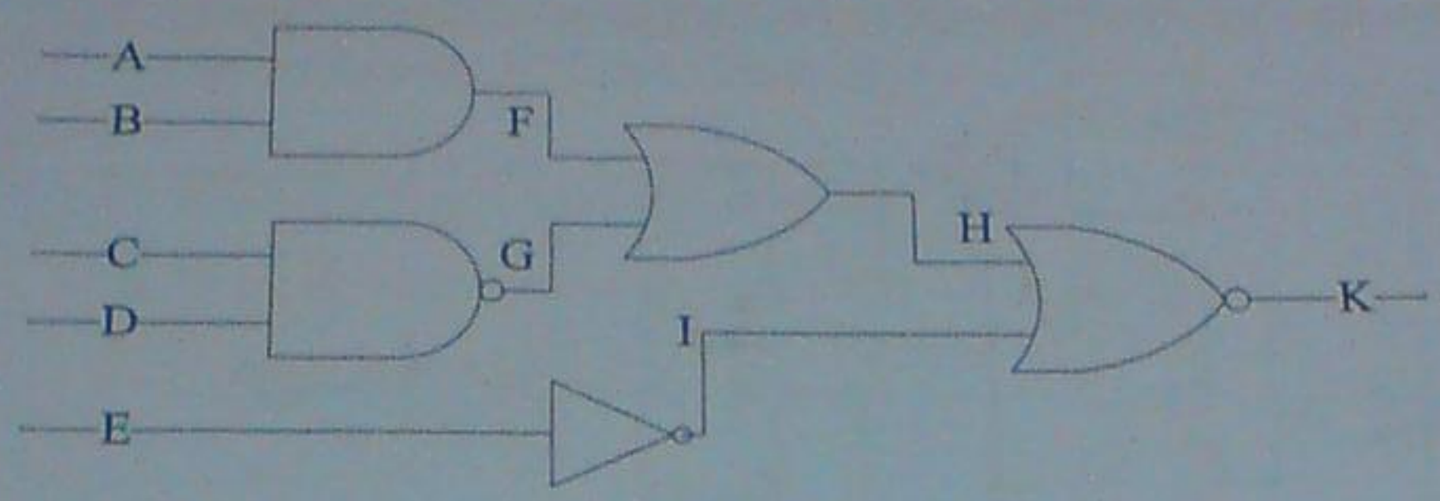


The added input and output pins are:

Input pin: SI, SE

output pin: SO

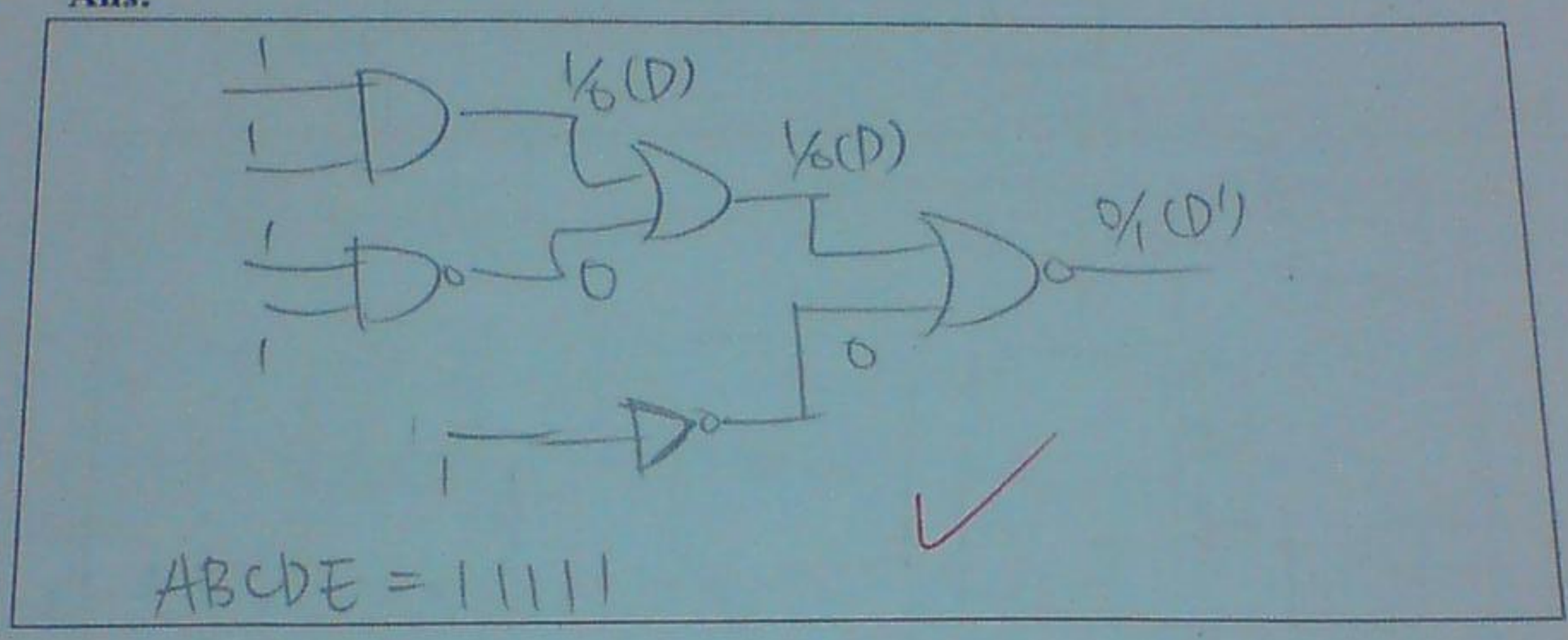
Problem 9 <Fault Simulation & ATPG>



(5%) A. Apply the D-algorithm to find out the test pattern for the "stuck-at 0" fault at net F

5

Ans:



(5%) B. Following question (A), please perform fault simulation to decide if the test pattern(s) generated by (A) is (are) able to detect the bridging fault "G dominant F." (hint: "G dominant F" means the value of F is dominated by the value of G)

5

Ans:

fault-free:

ABCDE	FGHI	K
11111	1010	0

faulty (G dominant F):

ABCDE	$F_f G H_f I$	K_f
11111	0000	1

$K \neq K_f$ = detect

```
endmodule
```

Ans:

	0	-6	6	x	x
	5	-6	255	x	127
107	15	255 -6	-6 255	-3 x	125 31
203		-1	-6 250	x	125 31
	0	-1	-6 250	-31	125 31
	40	0 15	-6 250	-3 -1	125 31