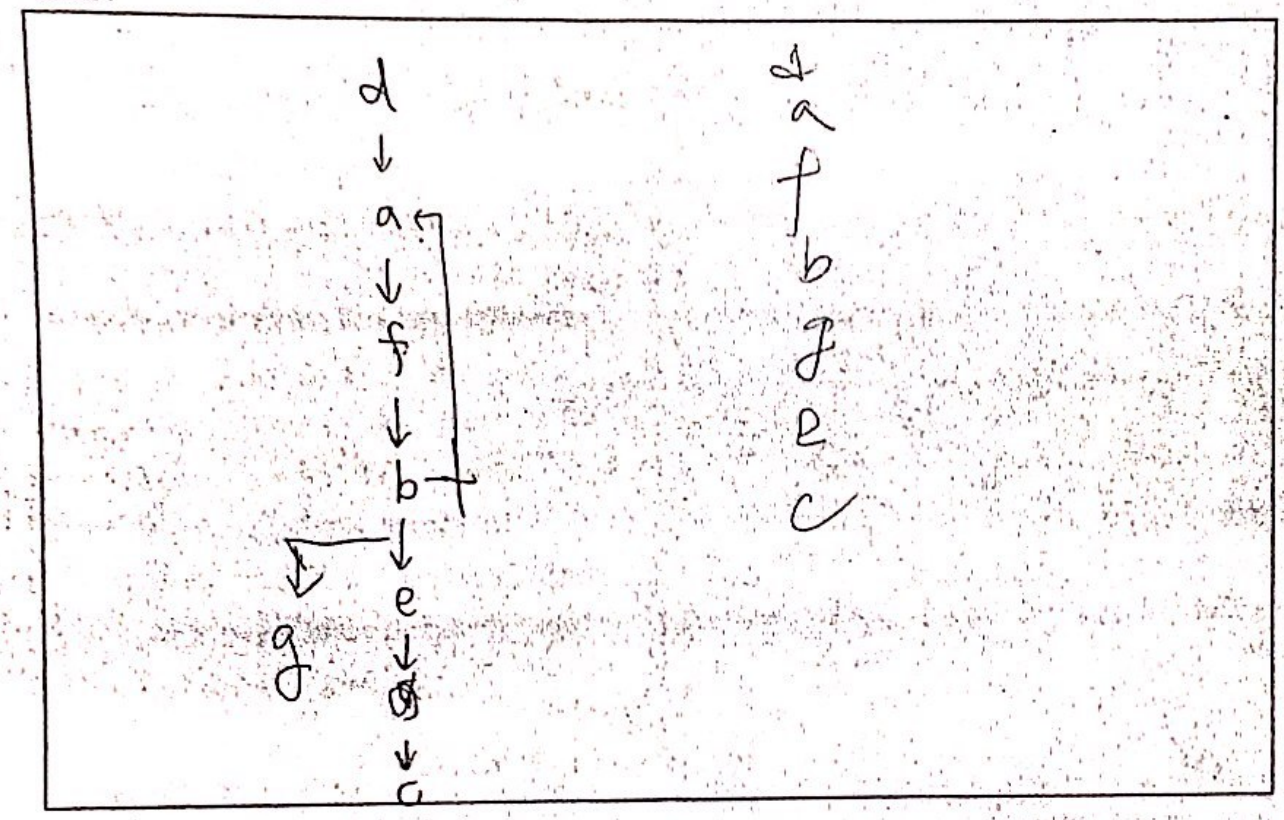
Problem 1 <Basic Concept>

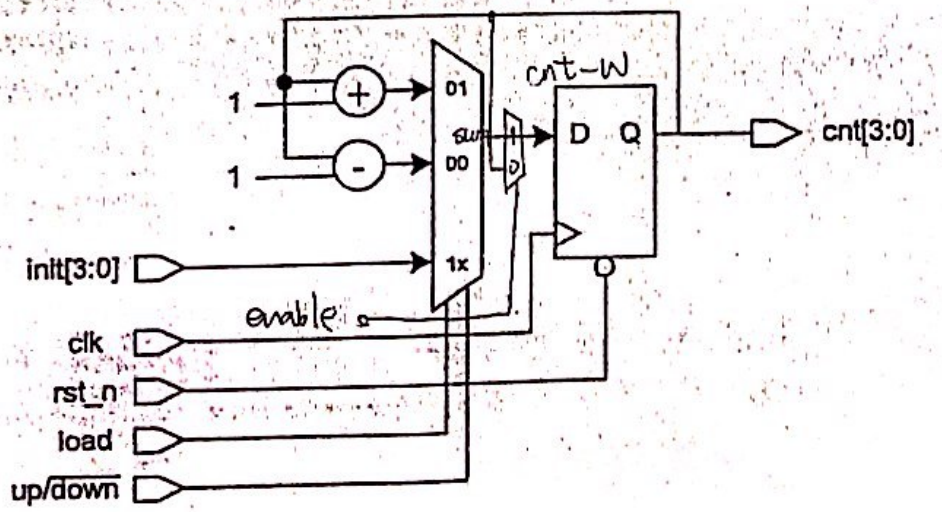
(6%) The following items are design flow steps. Please draw a design flow chart with correct connection between each step:
(a)RTL Coding; (b)Dft Insertion; (c)Tape Out; (d)Specification; (e)Place & Route; (f)Synthesis; (g)ATPG

Ans:



Problem 2 <Verilog Debug and Coding>

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



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

Verilog code with bugs

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

input clk, rst_n;
output [3:0] cnt;
input [3:0] init;
input load;
input updown;

wire [3:0] cnt_w;
reg [3:0] cnt;
reg [3:0] init;

always@(cnt or init)
begin
  case({load, updown})
    2'b00: cnt_w = cnt - 1'b1;
    2'b01: cnt_w = cnt + 1'b1;
    2'b10: cnt_w = init;
  endcase
end

always@(posedge clk or posedge
rst_n)
begin
  if(!rst_n)
    cnt <= 4'd0;
  else
    cnt <= cnt_w;
end

endmodule
```

Corrected Code

There are totally _____ bugs.

B. (10%) Rewrite the code, use *if...else...* instead of *case*. Please also add one more input pin "enable." When enable is 1, the counter can function as usual, but when enable is 0, the counter will hold the final cnt value and stopped with ignoring all the other input signals of load, init, and updown.

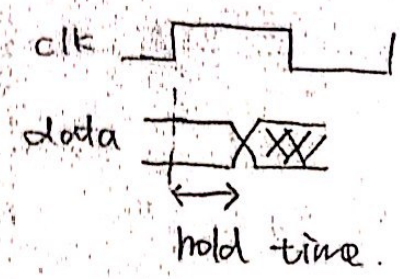
Ans:

```
module counter (clk, rst_n, cnt, initial, load, init,
                load, updown, enable);

input clk, rst_n, enable;
input load;
input updown;
input [3:0] init;
output [3:0] cnt;

reg [3:0] sum;
reg [3:0] cnt_w;

always @(load or updown or init or cnt)
begin
if (load == 1'b1)
sum = init;
else if (load==0 && updown == 1'b1)
sum= cnt+1'b1;
else
sum= cnt - 1'b1;
end

always @(enable or sum or cnt)
begin
if (enable)
cnt_w = sum;
else
```

C. (10%) After synthesis, if the simulation of the netlist shows there are some hold time violations on the output register. (a) What does it mean? (b) If we want to fix it manually, please draw the corresponding circuits.

Ans:

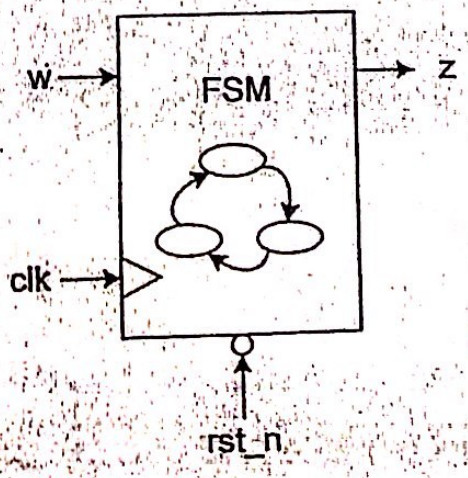
(a) 當 data 在 clk 來之後要 hold 住一段時間才能變動，沒到達 need 的時間.

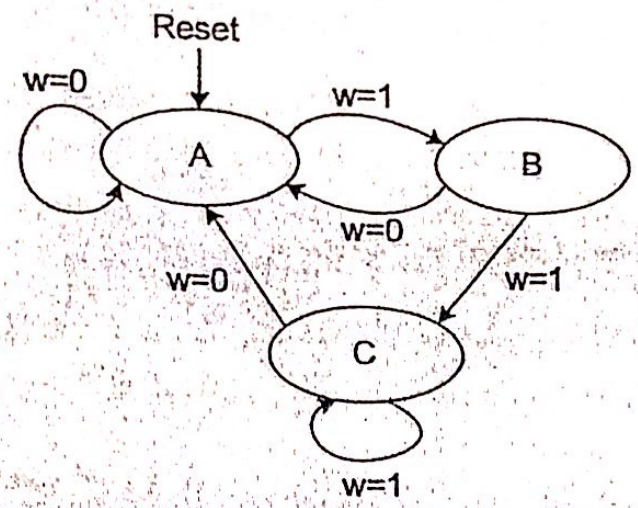


(b) 增加 hold time.

Problem 3 <Finite State Machine>

(10%) A state diagram of a finite state machine is shown below.



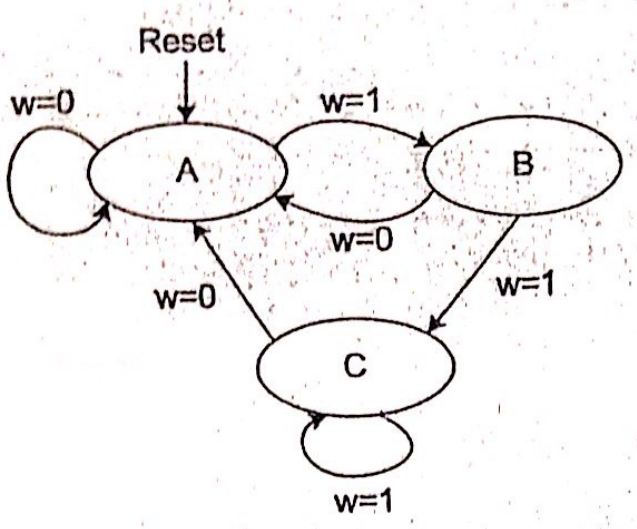


State	Output z
A	0
B	0
C	1

Please complete the following Verilog code:

Ans:

```
module FSM(clk, rst_n, z, w);

input clk, rst_n;
output z;
input w;

parameter A=2'b01, B=2'b10, C=2'b11;

reg [1:0] state; //current state
reg [1:0] next_state; //next state logic output
reg z;

//next state logic
always@(posedge clk or negedge Reset )
begin
 if(Reset)
   CS=A;
 else
   CS=NS;
end

always @(CS or
begin          A or B
               or C)
case(CS)
A: begin
   z=0;
   if(w)
   NS=B;
   else
   NS=A;
   end
B: begin
   z=0;
   if(w)
   NS=C;
   else
   NS=A;
   end
C: begin
   z=1;
   if(w)
default: begin
   Z=0;
   NS=A;
   end
endcase
end
```



State	Output z
A	0
B	0
C	1

Please complete the following Verilog code:

Ans:

```
module FSM(clk, rst_n, z, w);

input clk, rst_n;
output z;
input w;

parameter A=2'b01, B=2'b10; C=2'b11;

reg [1:0] state; //current state
reg [1:0] next_state; //next state logic output
reg z;

//next state logic
always@(posedge clk or negedge Reset )
begin
 if(Reset)
   CS = A;
 else
   CS=NS;
end

always @(CS or
 begin
 case(CS)
 A: begin
```

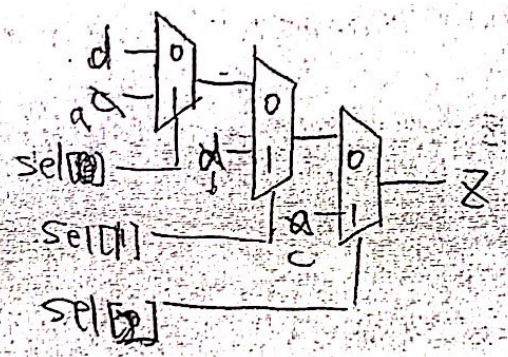
```
 end
B: begin
   z=0;
   if(w)
   NS=C;
   else
    NS=A;
   end
C: begin
   z=1;
   if(w)
```

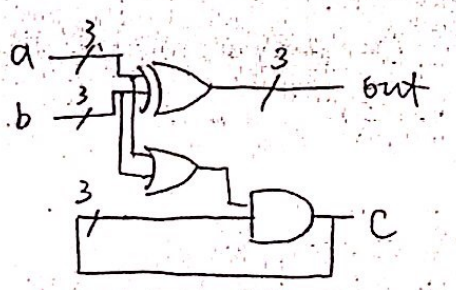
```
  end
default : begin
          Z=0;
          NS=A;
          end
endcase
end
```

Problem 4 <Manually Synthesis from Verilog RTL Code>

(10%) 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 in the circuit diagram.

Verilog Code	Circuit Diagram

`always @(a or b or c or d or sel)`
`begin`
`  z = d;`
`  if (sel[2])`
`  z = c;`
`  else if (sel[1])`
`  z = b;`
`  else if(sel[0])`
`  z = a;`
`end`
	

`always @(a or b or c)`
`begin`
`for( k=0; k<=2; k=k+1 )`
`begin`
`  out[k]=a[k]^b[k];`
`  c=(a[k]|b[k])&c;`
`end`
`end`
	

out[0] = a[0] ^ b[0]
out[1] = a[1] ^ b[1]
out[2] = a[2] ^ b[2]

C = (a[0] | b[0]) & (a[1] | b[1]) & (a[2] | b[2]) & C

a[0] →

Problem 5 (16%) <Synthesis with Design Vision>

A. (6%) In Design Vision, the warning message "multiple design instance" results from that you use the same HDL description to represent more than one design instance, how do we to handle it? There maybe several methods to handle it, what is the property of each method?

Ans:

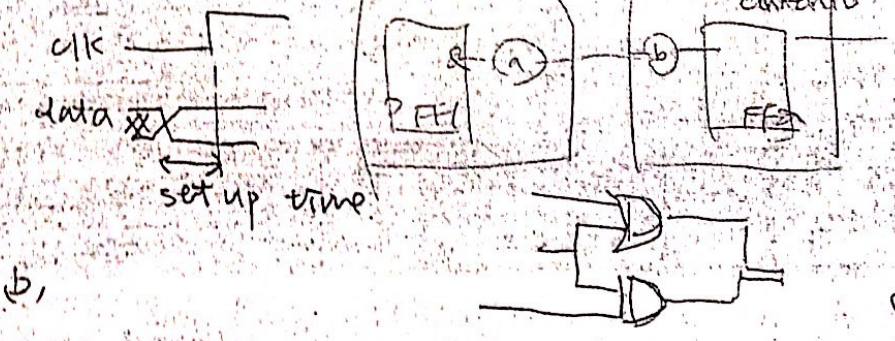
dont_touch
ungroup
uniquify

B. (10%) Describe the following terms in synthesis (you can draw the diagram to explain it)

(a)Setup Time (b)Input Delay (c)False Path (d)Flatten (e)Slack

Ans:

(a) 在 clk 來之前，有多的時間 data 不可更動



(b)

Problem 6 <Design for Testing>

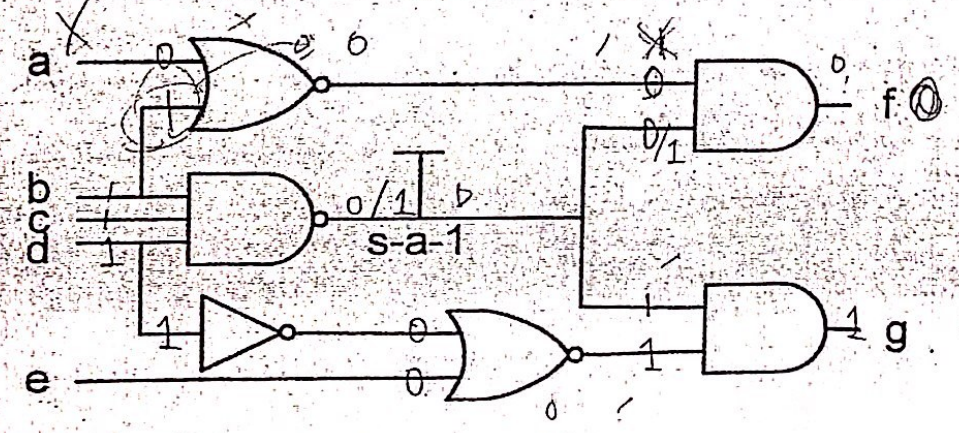
A. (8%) It is easy to confuse testing and verification. Please fill out the following table to show whether the left-hand-side tasks should belong to testing or verification.

Ans.

Task	Testing or Verification
Find out functional error of the circuits	T V
Find out manufacturing defects of the chip	X T
Find out package defects	X T
Find out timing error of the circuits	T V
To check if the design can meet the target specification	T V
FPGA prototyping	T V
Should be done repeatedly for all chips	X T
If the error is found, it can be fixed	T V

B. (10%)Please employ D-algorithm to find out the test pattern for the following fault. (please derive the pattern directly at the following figure)

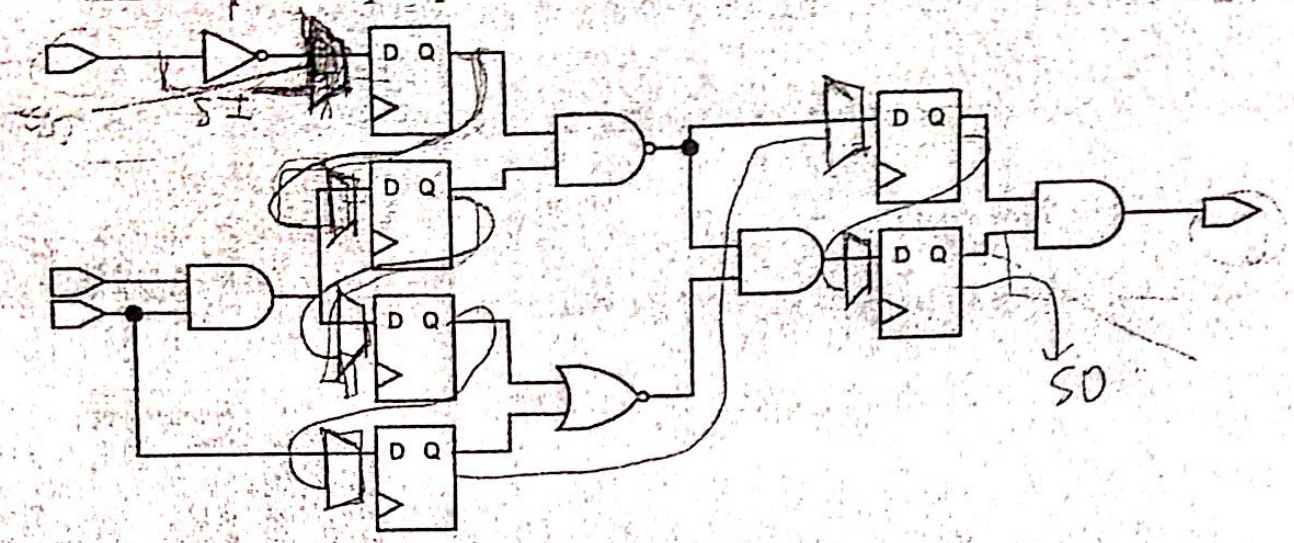
Ans.



Test input vector (a, b, c, d, e) = (X 1 1 1 1)

Expected output vector (f, g)= (0, 0)

C. (10%) For the following circuits, please draw the associated circuits with full-scan with multiplexed flip-flop. What input and output pins should be added?



Ans.

tsen

The added input and output pins are: