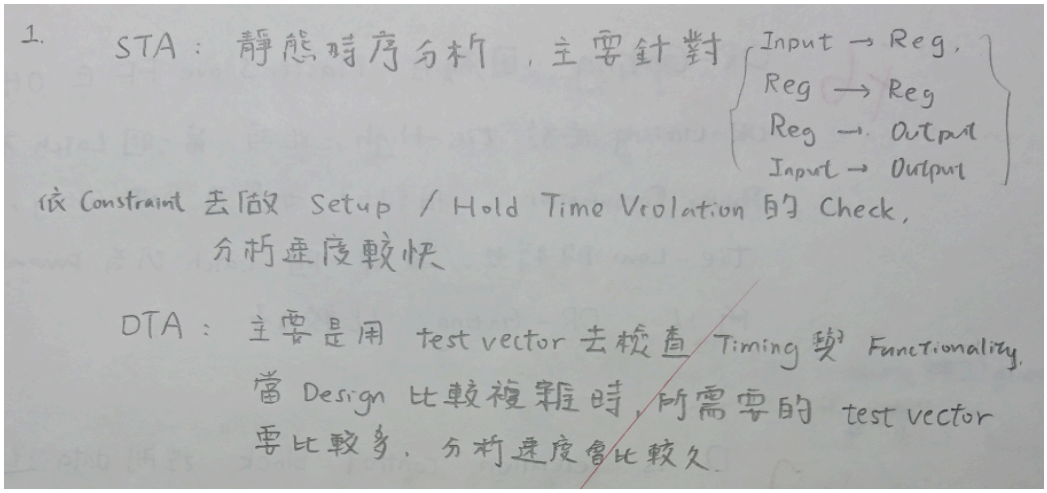
Timing

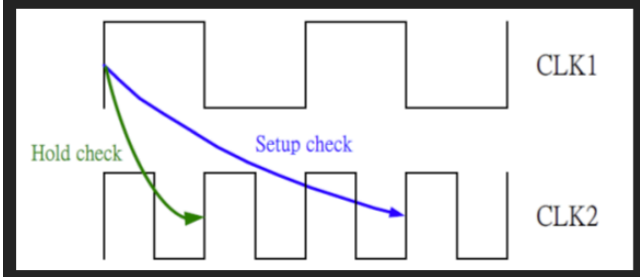
1. [10%] Please explain STA (static timing analysis) and DTA (dynamic timing analysis) briefly and compare them.



(c) If we want to set multicycle path as the following waveform, what value should be filled in the parentheses?

set_multicycle_path ( ) -from CLK1 -to CLK2

set_multicycle_path ( ) -hold -end -from CLK1 -to CLK2



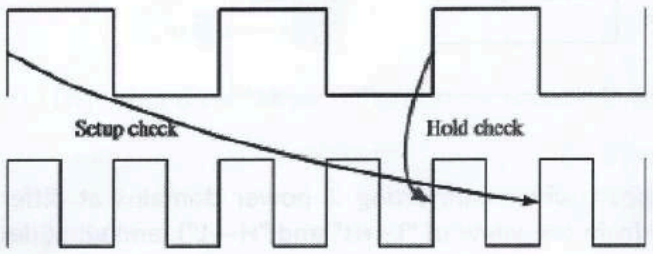
Ans:

(c) 3, 1. (1pts)

https://aijishu.com/a/1060000000206169

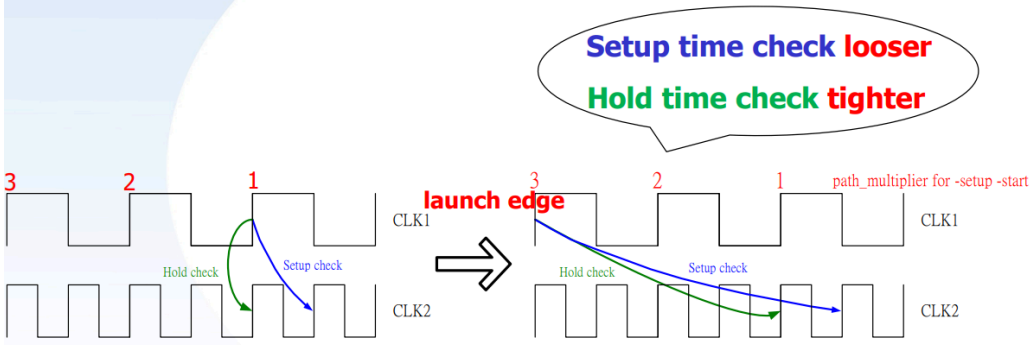
1. **[4%] Lab07:**

Please write two commands relative to "start point" to let the setup time check and hold time check like the picture below :
Hint : set_multicycle_path path_multiplier –from $start_point –to $end_point [-setup] [-hold][-start][-end]



✓ **Specify multiple path for setup time check relative to start point**

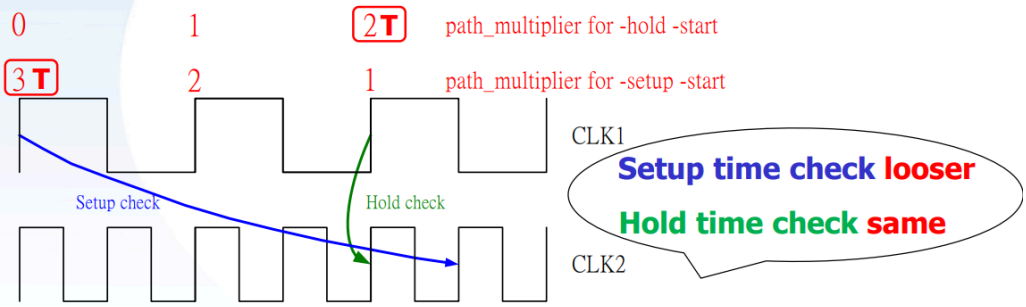
set_multicycle_path 3 -start –from CLK1 –to CLK2



✓ **Specify multiple path for hold time check relative to start point**

set_multicycle_path 3 -start –from CLK1 –to CLK2

set_multicycle_path 2 -hold -start –from CLK1 –to CLK2



Name: setup 從1開始算 hold 從0開始算 **Student ID:**

1. [6%] Please draw the diagram of setup and hold time check if a register in CLK1 domain is writing signal into a register in CLK2 domain under the set_multicycle_path timing constrain:

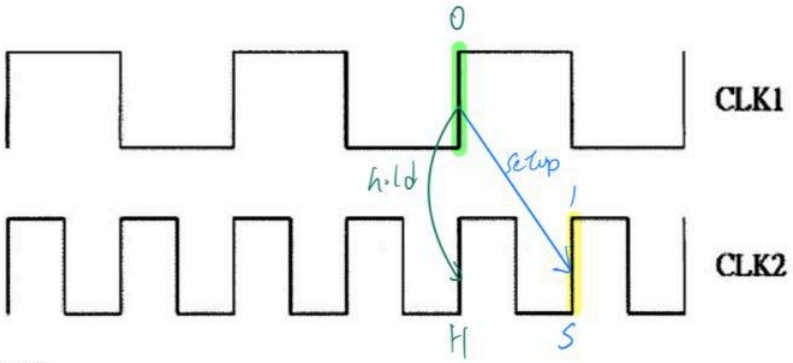
(1) (2 points)

step1 **set_multicycle_path 1 -end –from CLK1 –to CLK2**

step2 **set_multicycle_path 0 -hold -start –from CLK1 –to CLK2**

Given that the waveform relationship of clk1 and clk2 is:

(if the first rising edge of CLK1 is perfectly aligned to the first rising edge of CLK2 and the period of CLK1 is twice longer than the period of CLK2)



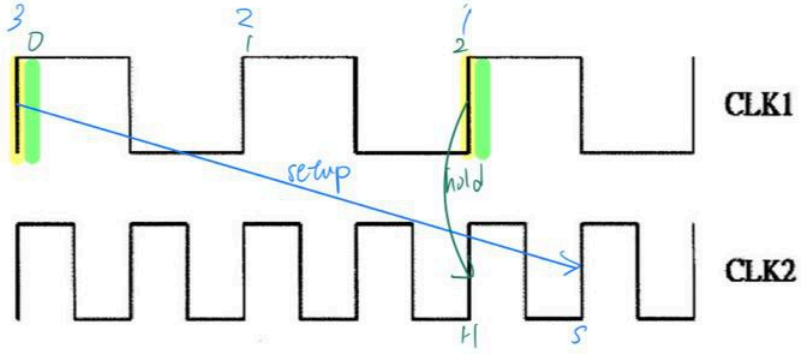
(2) (4 points)

step1 **set_multicycle_path 3 -start –from CLK1 –to CLK2**

step2 **set_multicycle_path 2 -hold -start –from CLK1 –to CLK2**

Given that the waveform relationship of clk1 and clk2 is:

(if the first rising edge of CLK1 is perfectly aligned to the first rising edge of CLK2 and the period of CLK1 is twice longer than the period of CLK2)



CDC

2. [9%] Please explain the main difference between double flop synchronizer and XOR double flop synchronizer if we use them in a CDC (clock domain crossing) design and why do we need to use synchronizer in a CDC design.

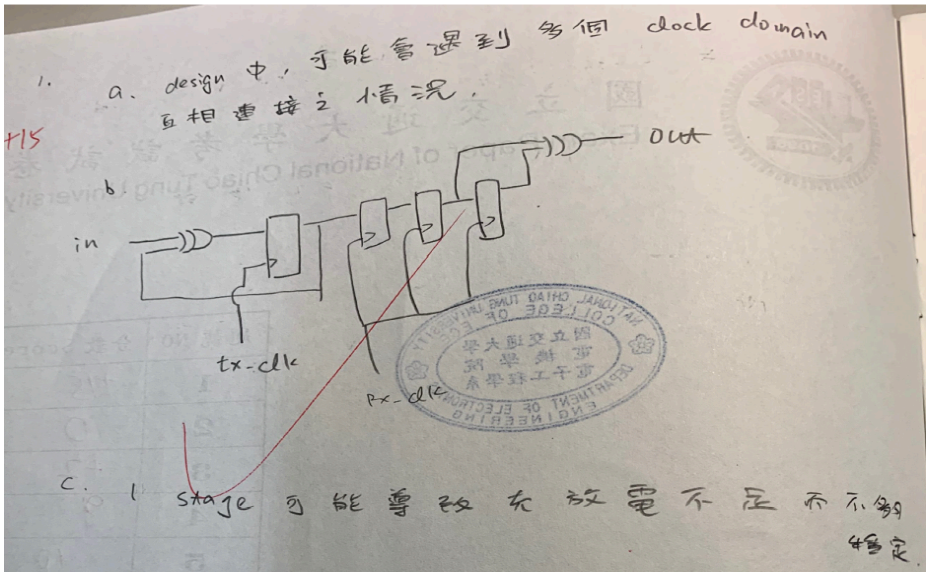
2.

double flop synchronizer: pass signal through different clock domain

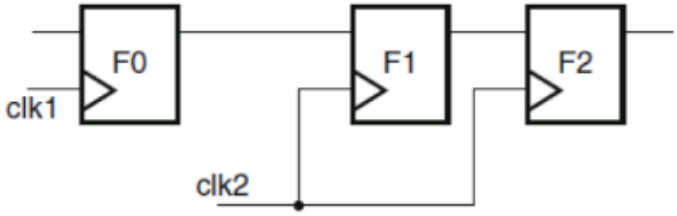
XOR double flop synchronizer: pass a pulse signal to different clock domain

We use synchronizers to transfer signals between different clock domains and avoid signal integrity problems during the transfer.

1. [15%] Please answer the following questions:
   a. [3%] Please explain CDC.
   b. [8%] In order to solve CDC problem, you need to use Synchronizer with XOR. Please draw the "2-stage Synchronizer with XOR" with Flip-Flop and XOR.
   c. [4%] And if we use 1-stage Synchronizer with XOR, what problem may occur?

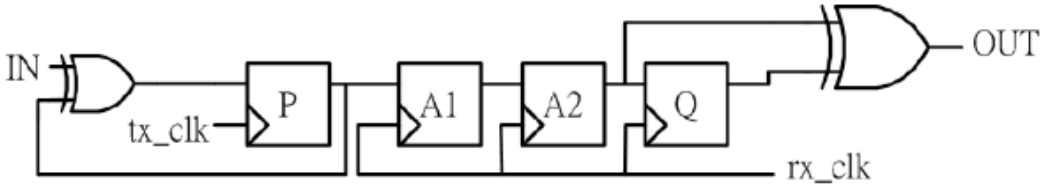


1. [10%]:
   Please clearly explain how double flop synchronizer and XOR double flop synchronizer work in our design if we use them in a CDC (clock domain crossing) design and why do we need to use synchronizer in a CDC design.



F1 samples the asynchronous input signal into the destination clock domain, probably a meta stable value. With a full clk2 cycle to permit any meta stability on the F1 output signal to decay. Then the signal is sampled by the same clock into F2, which we expect to be a stable and valid signal.

XOR Double flop synchronizer :



The signal we expected (Ex: a pulse) might be missed in double flop synchronizer, so we add XOR gate to both tx_clk domain and rx_clk domain to get the signal we expected. The XOR gate with flip-flop (Toggle Flop) of tx_clk domain turn the pulse signal to level signal, then double flop synchronizer samples the signal. Finally the XOR gate with flip-flop (Toggle Flop) of rx_clk domain turn the level signal to pulse signal.

(a) What is CDC (Clock Domain Crossing)?

(b) What is metastability? Why we may meet this problem in CDC cases?

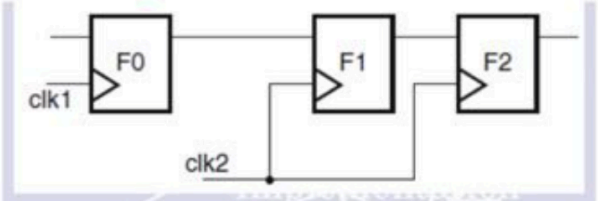
(c) Please draw the block diagram of Double Flop Synchronizer, and give an example of using it.

Ans:

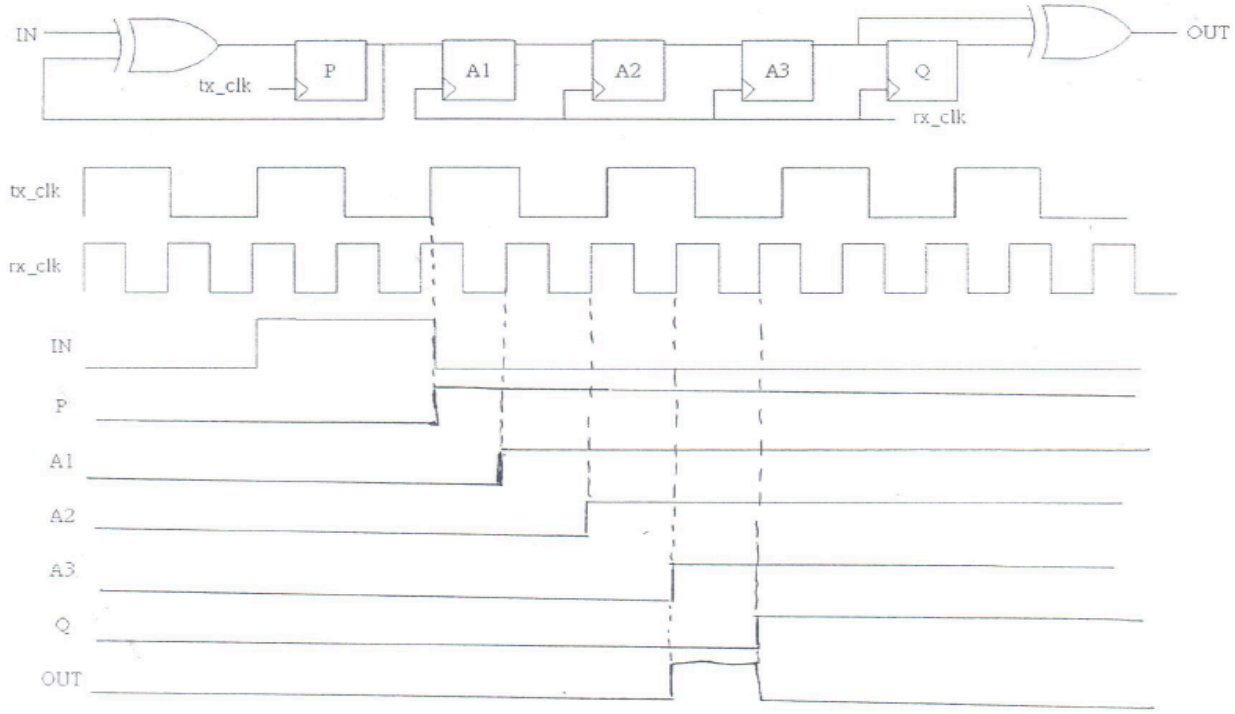
(a) When the data launched and captured by different (asynchronous) clock domain, this case is called CDC. (2pts)

(b) The unstable status due to non-ideal data transition is called metastability. In the CDC cases, sometimes the input of a clock triggered flip flop will come from another clock domain, and the time between the edges of different clocks may be uncertain, this may cause the metastability. (2pts)

(c) We can use it as a flag to determine whether the data is stable or not. (2pts)



2. [4%] (a) Please explain why we need a synchronizer in our design and its functionality.
[4%] (b) What may happened if we don't use XOR gates in a synchronizer?
[6%] (c) Complete the waveform below

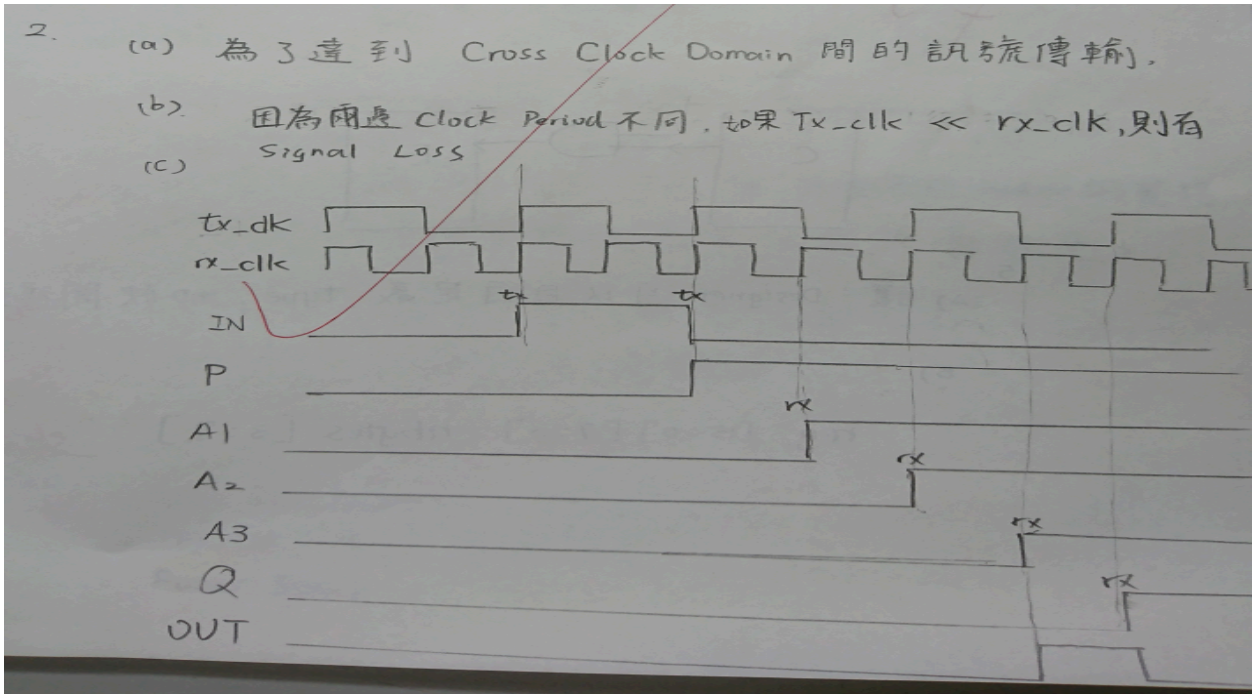


2.

(a) 為了達到 Cross Clock Domain 間的訊號傳輸.

(b) 因為兩邊 Clock Period 不同，如果 Tx_clk ≪ rx_clk，則有 Signal Loss

(c)



2. [4%]:
   We inserted synchronizers between multi clock domains. However, without further settings, the slack will be negative in synthesis and timing violation will occur in gate level simulation. Express what action must be done to remove the negative slack issue in synthesis (2%) and timing violation issue in gate level simulation (2%).

For synthesis, you have to release the timing check and inform the synthesizer, you can either directly write in syn.tcl, or write in syn.sdc and read that file in syn.tcl. Both setting multicycle path and setting false path make the timing check are OK, but setting multicycle path is recommended. Setting multicycle path make the timing check looser in certain paths, whereas setting false path just does not check the paths in certain path. Both methods make the timing check in synthesis level. (You can get full score with any method mentioned above)

For gate level simulation, you have to generate the sdf file that the first flip-flop in received clock domain have no setup and hold timing check. As mentioned in problem 1, this flip-flop will easily face the problem of setup / hold time violation and enter the metastable state. It is OK since the 2nd flip-flop will not. However, the simulator does not know and if setup / hold time violation occur, the output will be unknown. Thus the setup / hold timing check of the first flip-flop in received clock domain should be set to 0 in .sdf file, that's what the major part done in 02_run_pt in Lab07.

in syn.tcl or syn.sdf: set_multicycle_path, set_false_path, set_clock_groups

in pt.tcl: set_annotated_check

1. (a) Fig. 1 shows the basic 2 Flip-Flop synchronizer block diagram. Explain why a synchronizer is required in clock domain crossing (CDC) circuit. (5%)

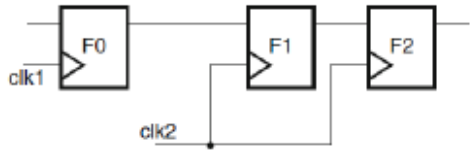


Fig. 1 2FF synchronizer

1.

When the frequency of clk1 and clk2 are different, the clock skew between clocks varies with time. The skew leads to a problem that sometimes setup / hold check will be violated, and sometimes not. This will cause the receiving flip-flop in clk2 clock domain (In Fig.1, it is F1) enter metastable state, that the output of the flip-flop is not ideal VDD or GND. If this non-ideal signal is used by the following circuits, the speed will be slower and the power consumption will be large. Even worse, the functionality will be wrong. With 2 Flip-Flop architecture in Fig. 1, even when the flip-flop F1 is possibility in metastable state, the flip-flop F2 will not. Thus the previous problem can be avoided.

Note: You have to mention the timing issue problem and keyword metastability, or else you will get some deducted points.

Note2: The synchronizer is to avoid the phenomenon of metatability, however when passing multiple bits, you cannot guarantee all bits are passed at the same cycle. Therefore usually only 1 bit is passed to inform the receiving end that all data is ready, then all data will be passed; or the gray coding will be applied to the transmitted signal (signal changes each time with only 1 bit varies).

(b) Fig. 2 shows the pulse synchronizer which we used in Lab07. Different from the synchronizer in Fig. 1 which can pass a value, the pulse synchronizer can pass a pulse. Try to complete the following timing diagrams to verify it can pass a pulse between two clock domains either when clk1 is faster than clk2 (5%) and when clk1 is slower than clk2 (5%).

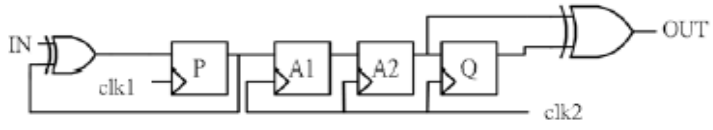
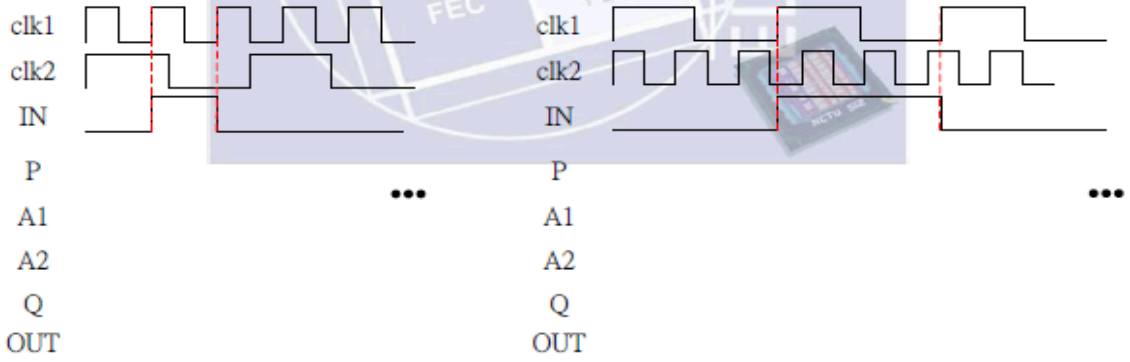
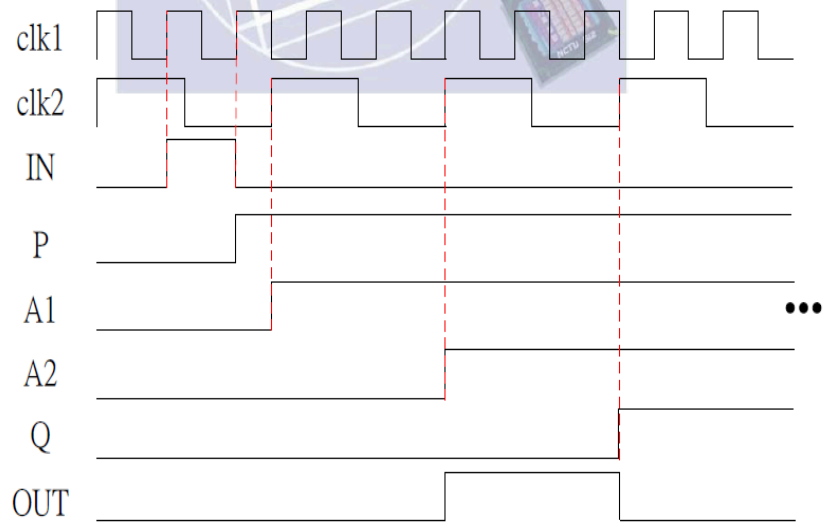


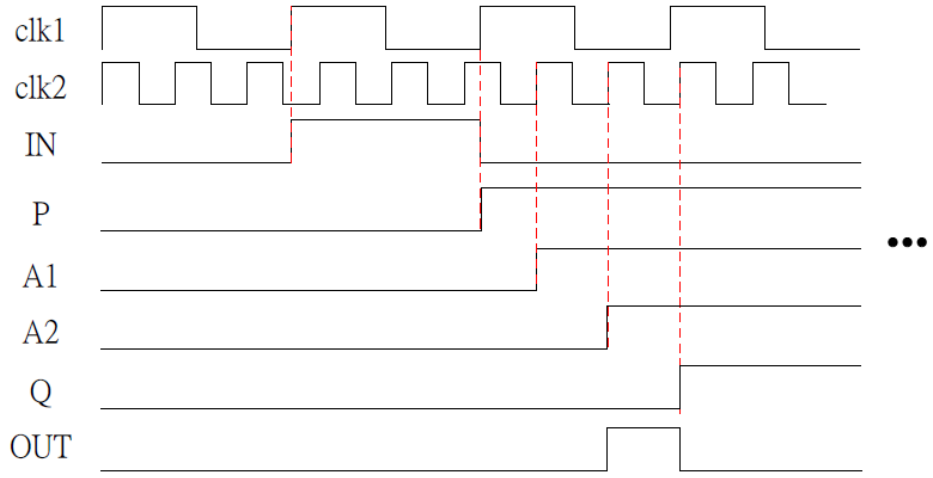
Fig. 2 pulse synchronizer



(Complete these two timing diagrams)

2. (a)





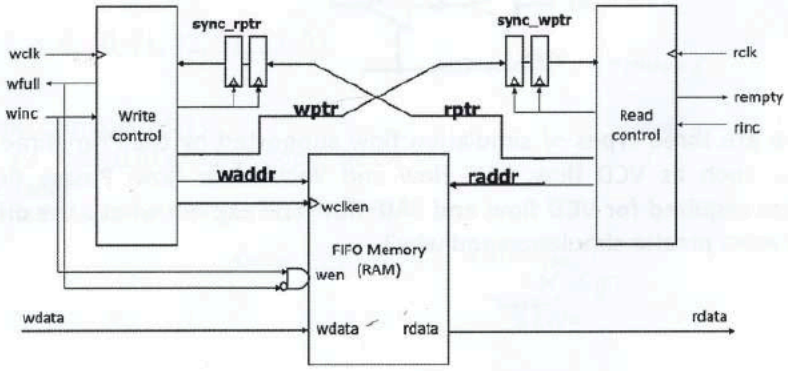
Note: each signal with 1 point. If you get a pulse at the end (OUT) with correct width but with wrong intermediate signals, you can get 1 point.

2. [11%] Lab07:

(a) [5%] In Dual Clock FIFO synchronizers, why do we need two D-FFs for write pointer? Now given a write action, what is the time relation between the data store in RAM and the read control responsing to this action? Please specify number of cycles.

(b) [3%] Why do we use gray encoding for pointers in Dual Clock FIFO synchronizers ?

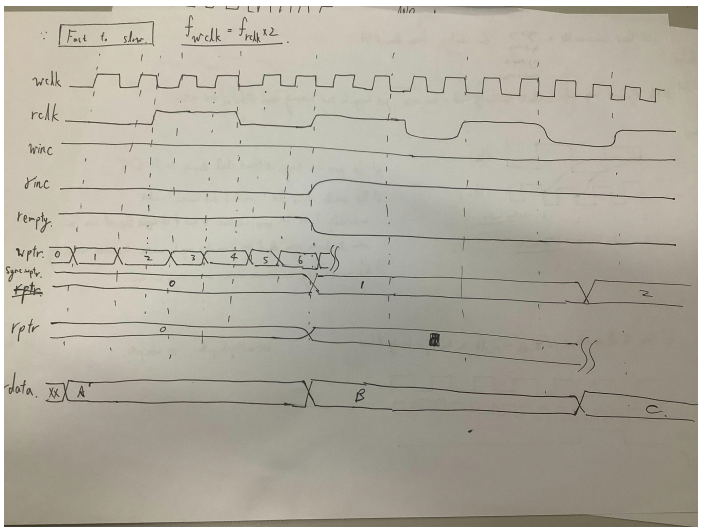
(c) [3%] What is the role of XOR in XOR synchronizer? Please consider both input and output.



a)

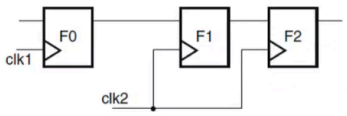
✓ Dual Clock FIFO synchronizers (Asynchronous FIFO)

- Data is written into a dual-port RAM block from the source clock domain and the RAM is read in the destination clock domain.
- Gray-coded read and write pointers are passed into the alternate clock domain to generate full and empty status flags.
- High throughput but difficult hardware.



✓ Double Flop(2-FF) Synchronizer

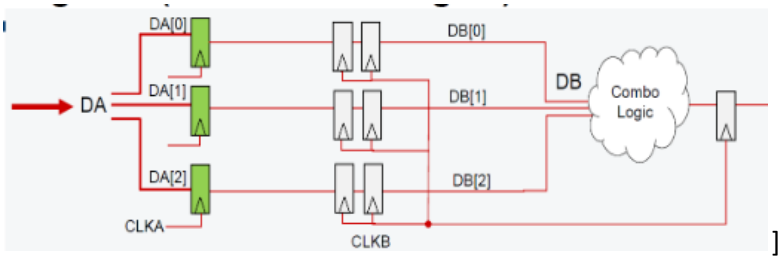
- F1 samples the asynchronous input signal into the destination clock domain, probably a meta-stable value.
- any meta-stability on the F1 output signal to decay. With a full clk2 cycle to permit
- clock into F2, which we expect to be a stable and valid signal. Then the signal is sampled by the same.
- Often used in 1-bit flag transmission.



(a) wptr and rptr are passed into the alternate clock domain to generate full and empty status flags by double FFs. To prevent data loss during passing, only one bit can be changed at one time.

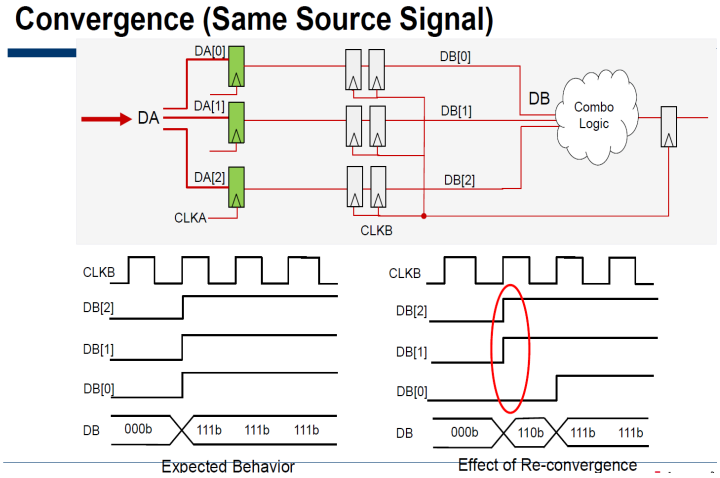
(b) XOR is used to ensure a pulse at the source domain will not be lost before it is synchronized at the input stage. At the output stage, XOR is used to generate a pulse at the destination clock domain.

Q: If DA is originally 3'b000, after passing through the synchronizer, the value should become 3'b011, yet the value is not the same as expected. What might the value of DB be? Draw out the waveform.How do we call this problem?



]

A:



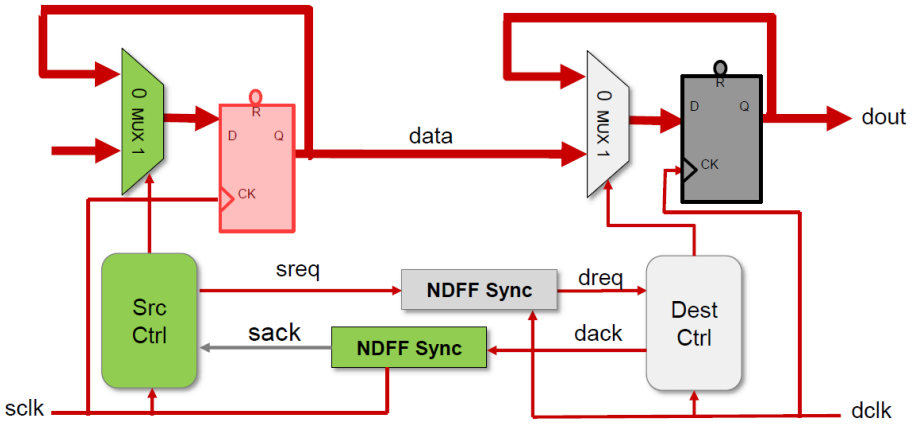
Q: Identify the incorrect structure of the wrong handshake synchronizer and draw the correct one out.

錯的圖, 好像在data bus上多了一個 NDFF BUS SYN

A: Correct Handshake syn.

Handshake Synchronizers

- Source and destination modules use a simple request-acknowledge protocol
- Closed loop synchronization that requires acknowledgment of receipt of the signal that crosses the CDC boundary
- High latency



Low Power

9. please list four ways to reduce the power consumption, and briefly explain why.(10%)

Dynamic Power Reduction: Switching Power = C * Vdd^2 * f

1. **Multi-Voltage**: reduce Vdd at non-critical block to reduce power consumption
2. **Power Gating**: shut down power to block of logic when it is not working
3. **RTL and Architecture Design Techniques**: pipeline increases area (capacitance), but the shallower logic reduces required Vdd
4. **Clock Gating**: reduce the power consumption of registers by turn off the unused registers / reduce the clock switching power

2. [3%] Describe 3 different Voltage Scaling Strategies (SVS, MVS, DVFS).

✓ Static Voltage Scaling (SVS)

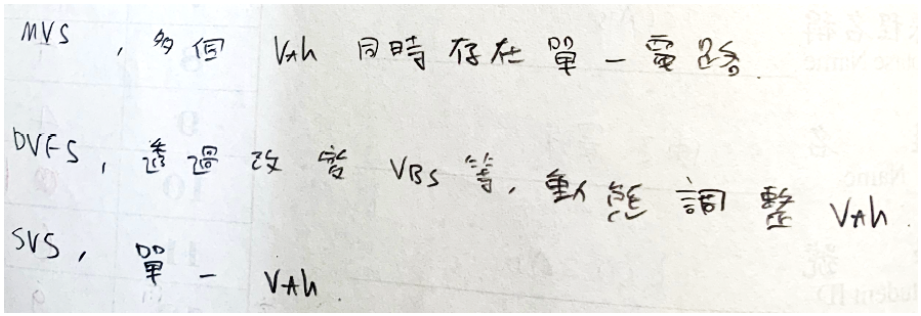
- Different blocks or subsystems are given different, fixed supply voltages

✓ Multi-level Voltage Scaling (MVS)

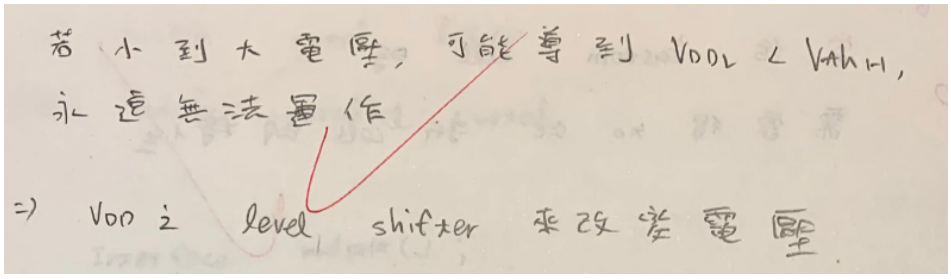
- A block or subsystem is switched between tow or more voltage levels
- Only a few, fixed, discrete level are supported for different operating modes

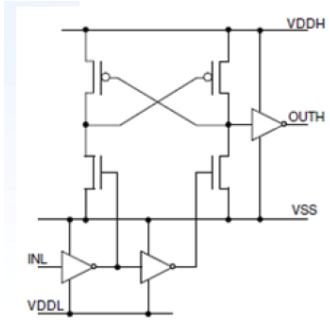
✓ Dynamic Voltage and Frequency Scaling (DVFS)

- An extension of MVS where a larger number of voltages levels are dynamically switched to follow changing workloads



3. [2%] What problems might occur when 2 power domains at different voltage levels are connected together (list one)? And how to solve it?





4. [6%] Please list two problems which will happen when a signal propagates between two power domains at different voltage levels. What kind of circuits should we use to deal with these problems?

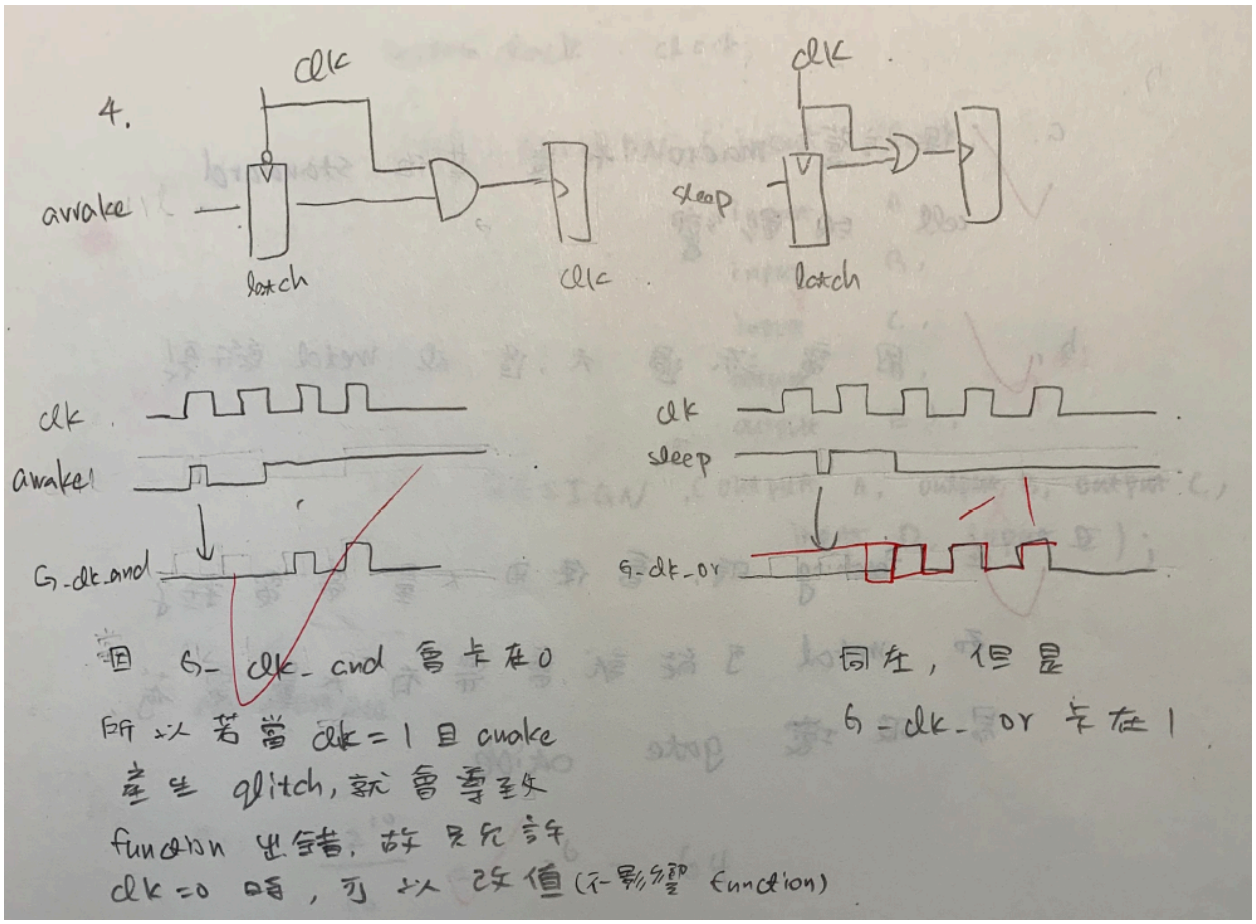
4.

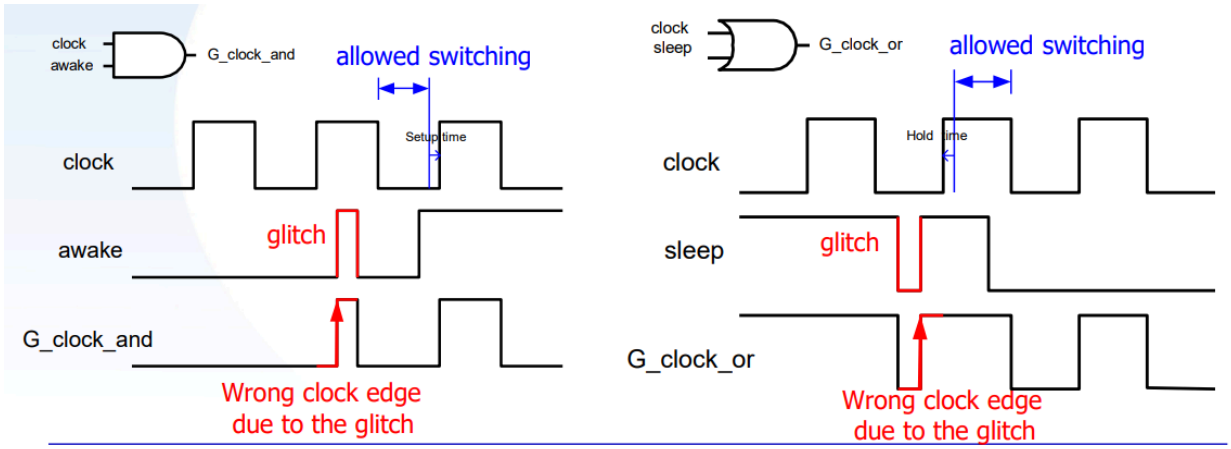
If a signal propagates from high voltage level to low voltage level, the high voltage may break the circuit.

If a signal propagates from low to high voltage level, it may not supply enough voltage and the data may be lost.

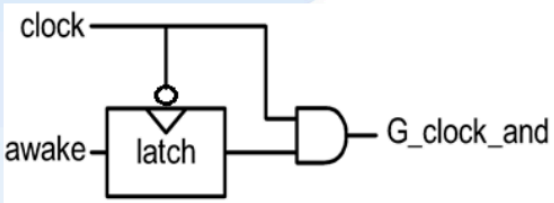
Use a level shifter to deal with this problem.

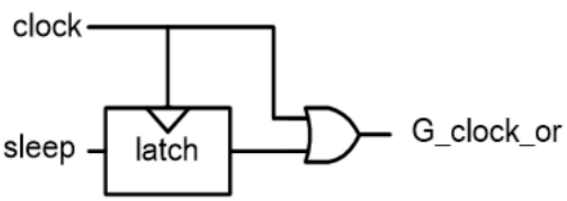
4. [10%] In Dynamic Clock Gating Technique, And-gating and Or-gating both suffer from glitch problem. Please draw the glitch-free Dynamic Clock Gating block diagram with input signals: clock, awake(sleep) and output signal: G_clk_and(G_clk_or) [6%] and their waveform (please also draw glitch in awake(sleep) signal and explain how the problem can be avoided? [4%]

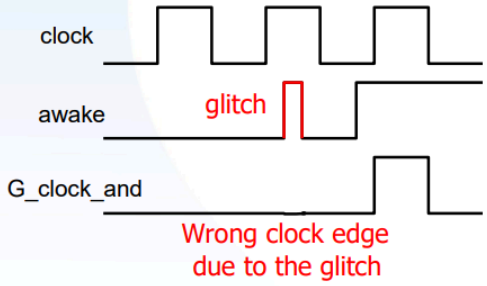


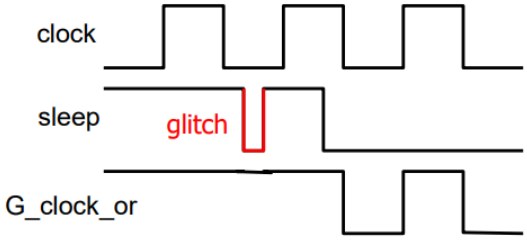


✓ The method of avoid glitch









4. Fig. 3 shows the AND-based and OR-based clock gating cells:

Fig. 3 AND-based and OR-based clock gating cells

(a) awake signal and sleep signal should be changed in a certain region (according to clock signals), or else the gated clock output will get glitches. Please specify the regions for both cells that awake / sleep should not change and use timing diagram to show that changing may result in glitches. (7%)
(Hint: you can let inputs: sleep / awake also have glitches that it will be easier to find the answer region)

(b) Latch based clock gating cells as shown in Fig. 4 solve the glitch problems, which becomes the famous clock gating cells nowadays. Please use timing diagrams again that the glitches problem in 4.(a) can be fixed. (7%)
(Your timing diagram should contain node t in both cases)

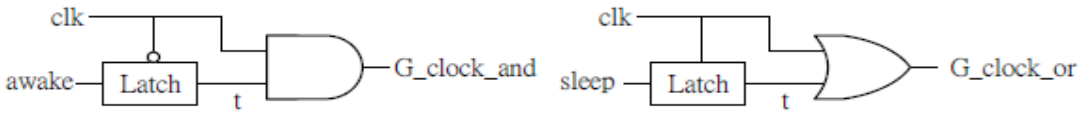
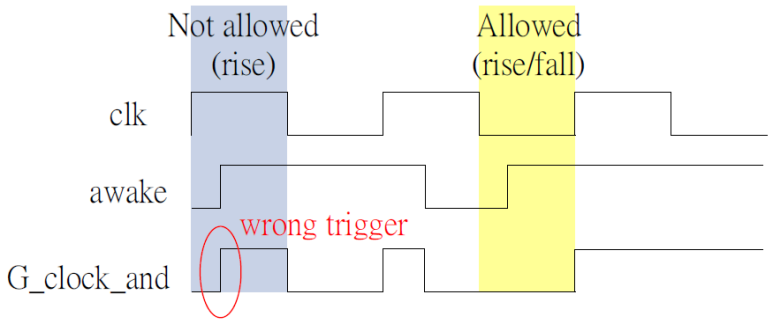
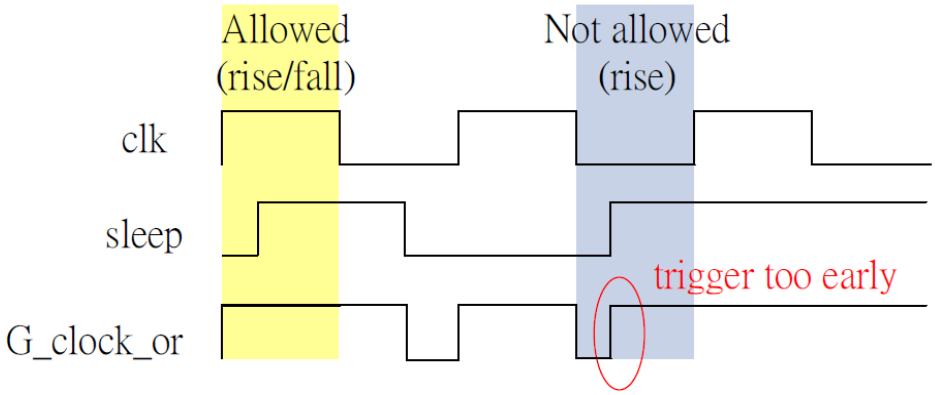


Fig. 4 Latch-based clock gating cells

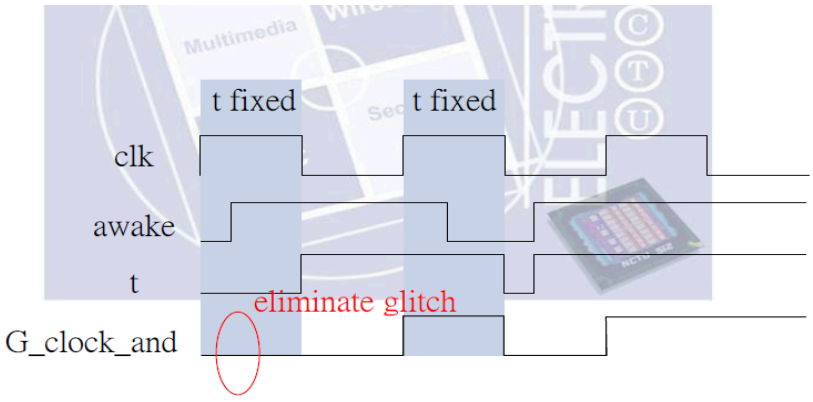
4. (a)

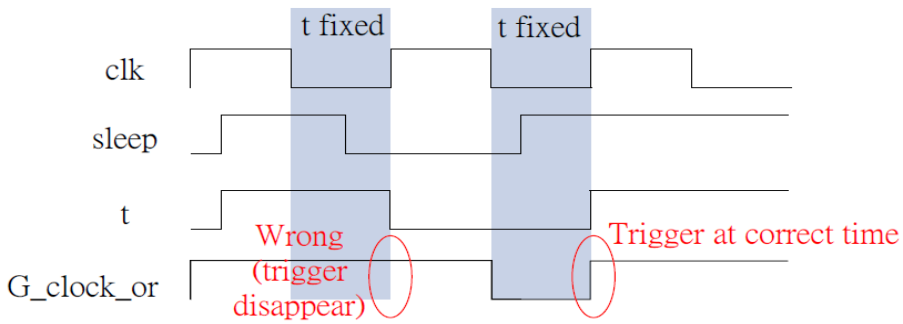




As shown in the above figures: for AND-based clock gating cells, the awake signal cannot rise when clk = 1. Since AND-base clock gating cells try to tie the output to 0, the awake signal passes when clk = 1 and when awake is rising, the flip-flop will be triggered. (Falling is actually OK. But when signal changes you cannot guarantee the combinational circuit output has no glitches. The glitch contains both rising and falling edge, thus glitch is also not allowed when clk =1). On the other hand, for OR-based clock gating cells, the output is tied to 1, thus the sleep signal cannot rise when clk = 0. (Again falling is OK, but any glitch is not allowed) If you draw the timing diagram but without specifying the allowed changing region, you will get 1 deducted point.

(b)



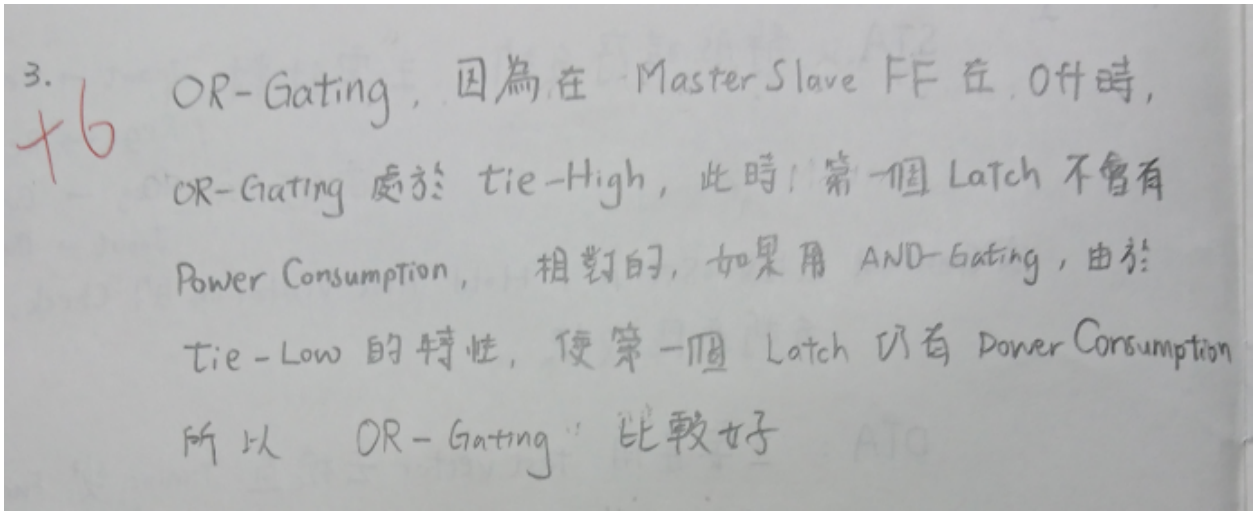


As shown in the above figures. Latches will gate the awake / sleep signals when clk=1 / clk=0, which are the non-allowable changing regions in part (a). Thus the trigger can be eliminated or occurs at correct time.

However, notice that in the bottom figure, which is the Latch version of Or-based clock gating cells, the problem remains: sleep signals still cannot change when clk = 0. Even though the glitch problem is fixed by Latch, the change of sleep signal cannot pass also. That's the reason in Lab08 practice if you used in_valid as the sleep signal, the result may be fail since in_valid changes in the latter cycle, which is clk = 0. The Latch version of AND-gated clock gating cells will not face this problem. Even that the awake logic cannot pass when clk=1, it can pass later when clk = 0, before the next clock rising edge. Thus keep in mind that in future if you want to use Latch version clock gating cells, AND-based clock gating cells will be better. (In slides, if clock gating cells are without Latch, or clock gating is better due to the power issue. And if the sleep/awake logic is calculated from registers instead of input signals, the glitch may occur when clk =1 so Or-based will be more preferred. However with latch version cells, above problems will not occur, and AND-based can perform the awake calculation every time (before next rising edge), but OR-based can only perform in the former half cycle, thus AND-based will be better).

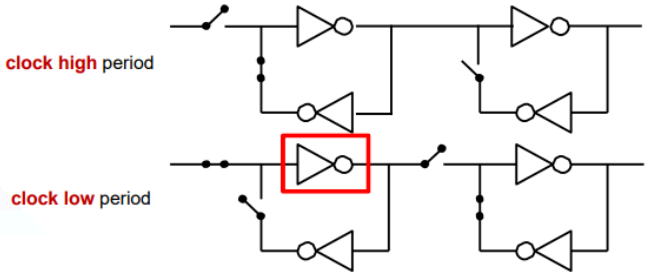
Note: if you do not draw t signal in timing diagram, you will get 1 deducted point.

3. [6%] When using clock gating, in turns of power consumption, which gating is better? AND-Gating or OR-Gating? Please justify your answer.



✓ OR-gating is better

- **OR-gating consumes less power** than AND-gating
  - For OR-gating, the gated clock is tied at high when the register is turned off
  - No matter data is toggling, the first latch circuits will not be toggled
  - For AND-gating, the gated clock is tied at low when the register is turned off
  - The first latch circuits will consume power as data input is switching
- The gating control signals should be generated from clock rising edge
- Flip-flops → stable gating the clock as clock high period
  - Consistent with original clock rising edge trigger registers design
  - It is easier for timing control and analysis



3. Shift register is a common method to save sequential input signal without making too much area (as shown in the following Verilog code).

```
always@(posedge clk) begin
      q1 <= in;
end
always@(posedge clk) begin
      q2 <= q1;
end
always@(posedge clk) begin
      q3 <= q2;
end
...
always@(posedge clk) begin
      q25 <= q24;
end
```

However, it is not suitable for clock gating design since all flip-flops should work when input is coming. If the percentage of inputting signal time is small, it is ok since flip-flop can be shut off when input is not coming. However when inputting signals dominate the design working cycle, more power will be consumed. How will you try to make less power consumption for saving long-term input signal? (Simply describe your method and simply write down the Verilog code) (6%)

A:

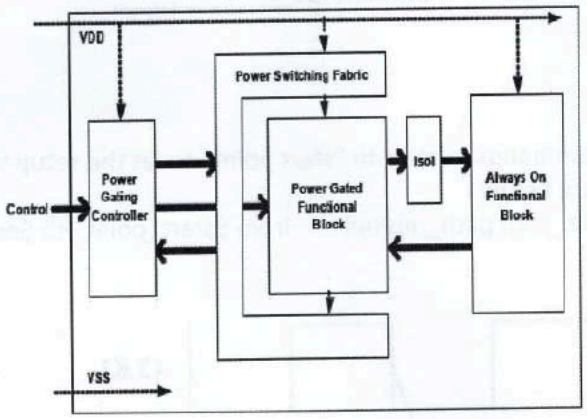
3.

Create a counter and make it increase when the inputs are coming. Store the input to the certain register according to the counter value. Then all flip-flops can be clock gated when the counter is corresponded to other storing flip-flops.

```
reg [4:0] cnt;
always@(posedge clk) begin
      if(!rst_n)
            cnt <= 5'b0;
      else if (in_valid)
            cnt <= cnt+1'b1;
      else
            …
end

always@(posedge clk) begin
      if(cnt==24)
            q1 <= in;
end
always@(posedge clk) begin
      if(cnt==23)
            q2 <= in;
end
always@(posedge clk) begin
      if(cnt==22)
            q3 <= in;
end
…
always@(posedge clk) begin
      if(cnt==0)
            q25 <= in;
end
```

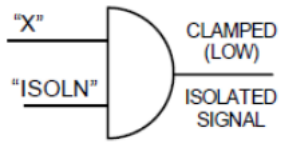
3. [12%] Lab08:

(a) [4%] Below is a classical architecture for power gating design.
(1) Which logic gate we can use to implement the "isolation cell", how it works and explain why we need it?
(2) Why we need to implement "retention register" in some situations?

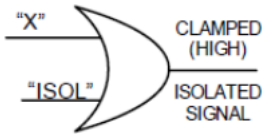


(a) AND gate or OR gate

Isolation



	0	1
0	0	0
1	0	1



	0	1
0	0	1
1	1	1

AND gate: when ISOLN=0, output=0; when ISOLN=1, output follows X.

OR gate: when ISOL=1, output=1; whenISOL=0, output follows X.

(b)

✓ States of some registers in shut-down mode need to be preserved

✓ Retention registers can **store data while in shut-down mode**

4. [8%] Down below is the figure of power gating.

A is “Power gating controller”

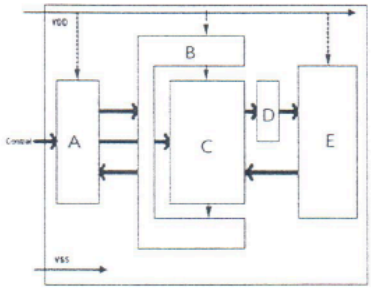
B is “Power switching fabric”

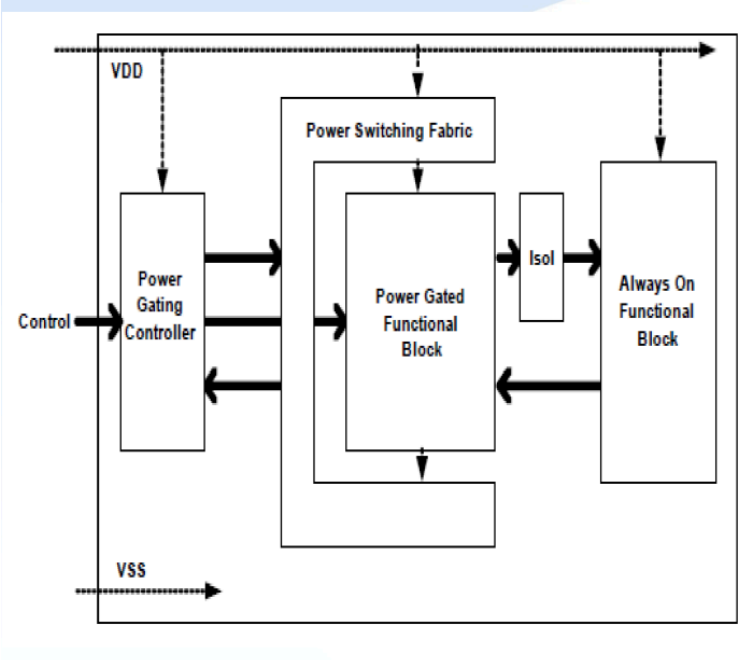
C is “Power gated block”

E is “Always on block”

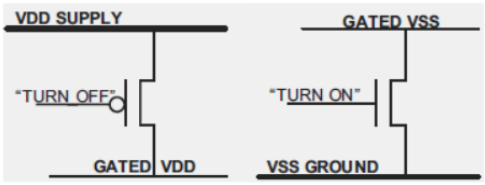
What is block “D”?

Please also briefly describe how to implement the block D using one gate as an example.

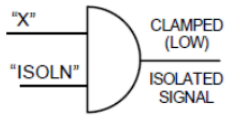




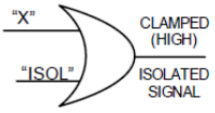
Power switching fabric



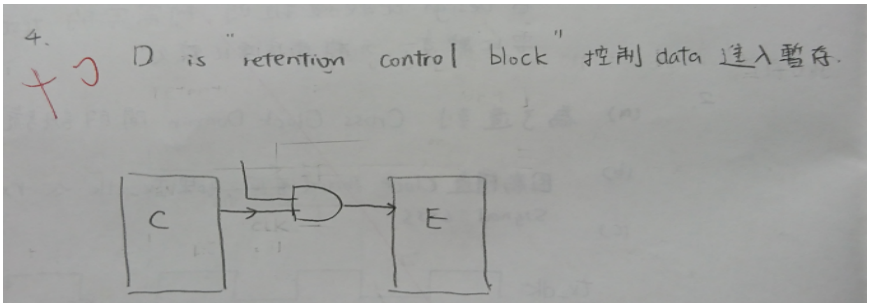
Isolation



	0	1
0	0	0
1	0	1

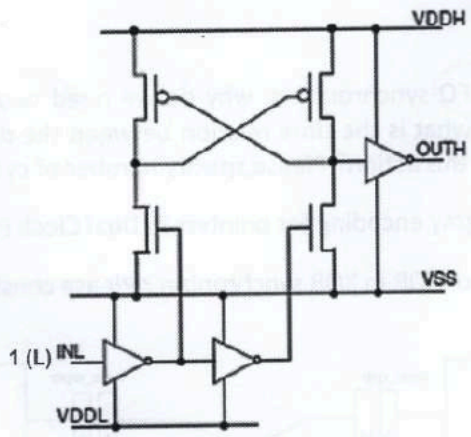


	0	1
0	0	1
1	1	1



(b) [4%] Please describe what will happen when connecting 2 power domains at different voltage levels ? (hint: please discuss from the wiew of "L→H" and "H→L"), and what device we will use to deal with this issue?

level shifter

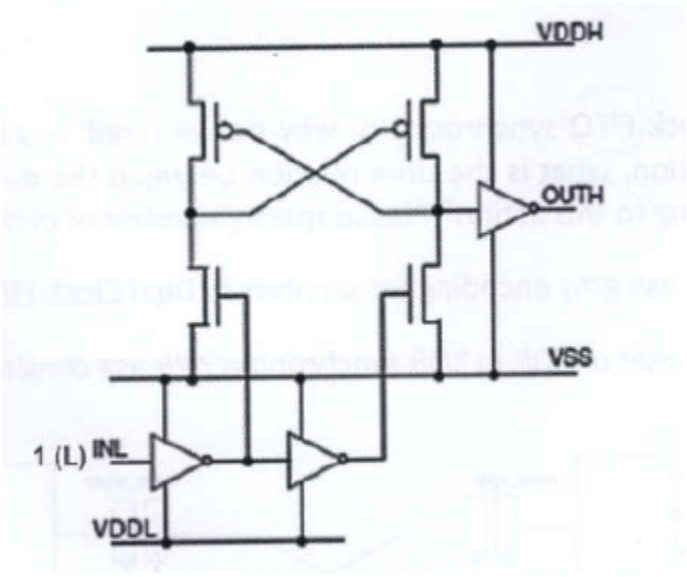


✓ **Connecting 2 power domains at different voltage levels can cause design issues**
- Timing inaccuracy
- Signals are not propagated

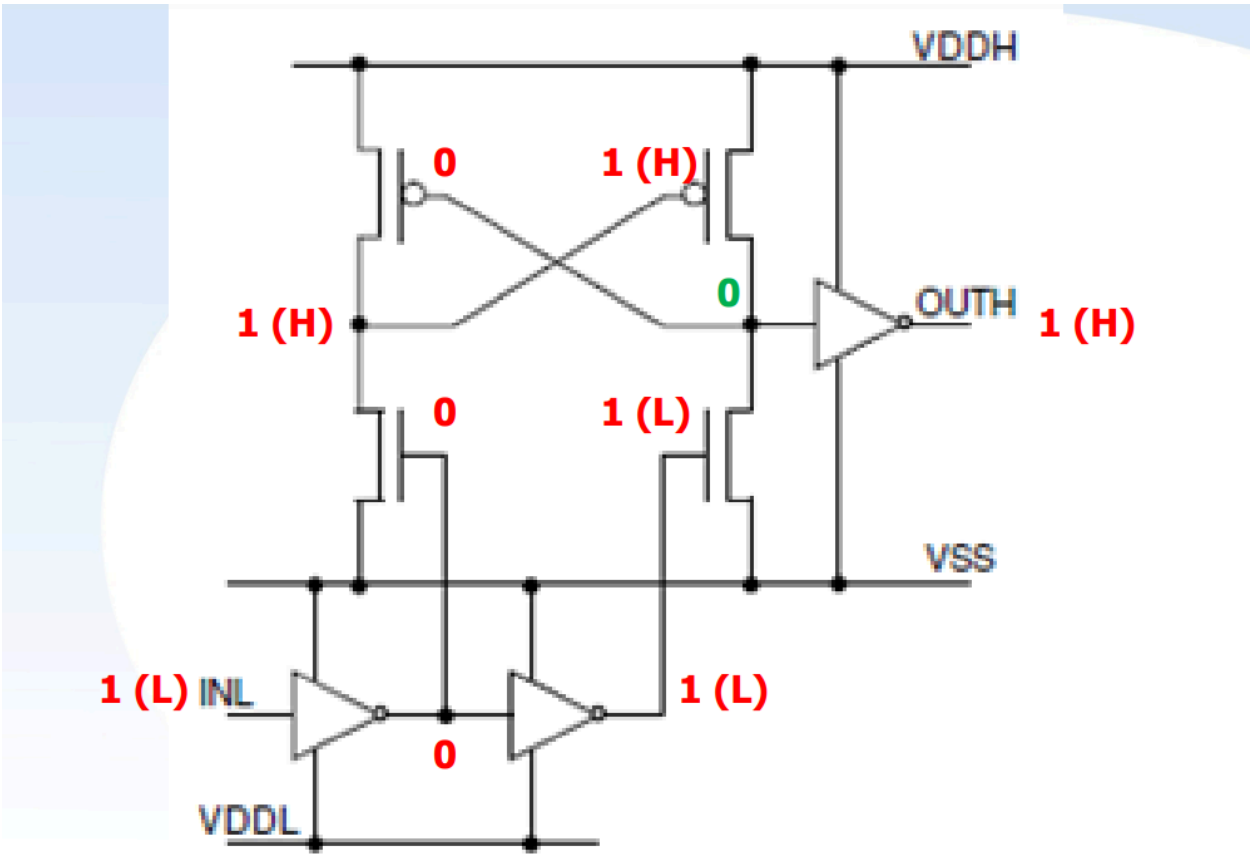
✓ **A level shifter is required**

(Lec08_Low_Power_Design P.20~28 discuss the view of "L->H")

2023 fall (Q): Given an empty level shifter, mark the voltage value 0,1(L),1(H) onto each possible node.



A:



Power analyze

3. [13%]:
There are three types of simulation flow supported by the Primetime tool for power simulation. Please explain all of them. You need to describe the **name, files of information required for simulation**, and **rank of precision** (write "1" for the most precise simulation, write "3" for the least precise simulation). Also, there are two types of signal activity files for power simulation. Please describe them respectively.

Name of the flow (3%)	Required information (3%)	Rank of precision (3%)

Describe the signal activity files (4%):

Name of the flow (3%)	Required information (3%)	Rank of precision (3%)
VCD Flow	1. Power Models 2. Netlist & Parasitics 3. Signal Activity (VCD)	1
SAIF Flow	1. Power Models 2. Netlist & Parasitics 3. Signal Activity (SAIF)	2
Vector-free Flow	1. Power Models 2. Netlist & Parasitics	3

(c) [4%] There are three types of simulation flow supported by the Primetime tool for power simulation, such as VCD flow, SAIF flow and Vector-free flow. **Please describe files of information required for VCD flow and SAIF flow and explain what's the difference.** Which one is the most precise simulation and why?

VCD flow: power models, netlists and parasitics, signal activity(VCD)

SAIF flow: power models, netlists and parasitics, signal activity(SAIF)

VCD is the most precise simulation because VCD file contains the most precise value changes of signals.

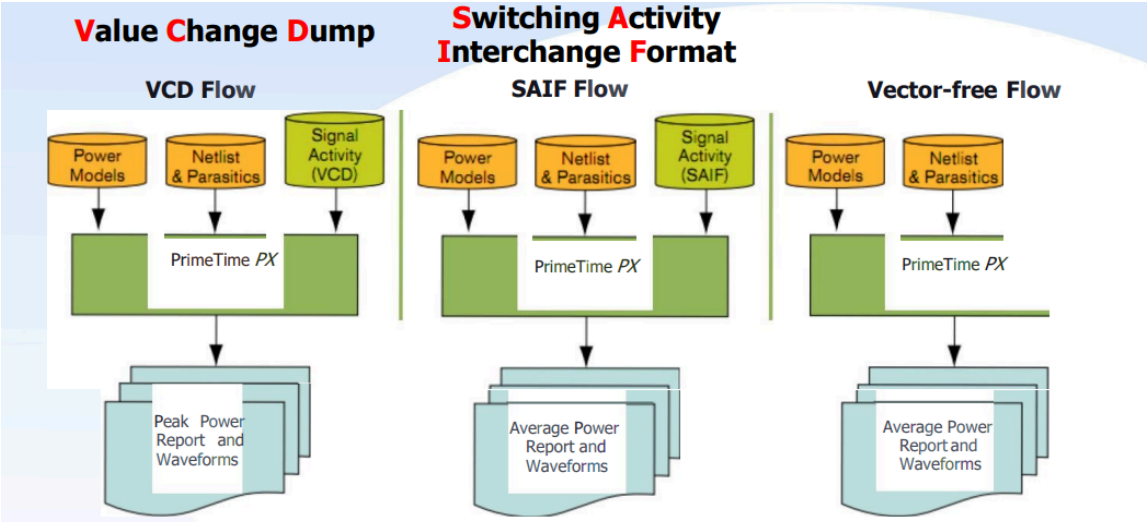


Figure 4:Power Analysis Flows.

VCD file : Contains value changes of a signal. i.e. at what times signals changes their values.

SAIF file : contains toggle counts and time information like how much time a signal was in 1 state, 0 state , x state. It contains cumulative information of vcd.

3. [10%] Actually, we can use report_power to get our power report in the design compiler. However, we usually get our power report by using PrimeTime. Why should we using PrimeTime to get the power report? What kind of file should we prepare if we want to get the power report by using PrimeTime?

3.

PrimeTime analyzes power based on signal toggling files, considering more details of the circuit, and is more precise. Design compiler only generates the power report for our reference because it doesn't consider the real signals.

We have to prepare power models, netlist & parasitics, and signal activity(for example: FSDB) for PT.

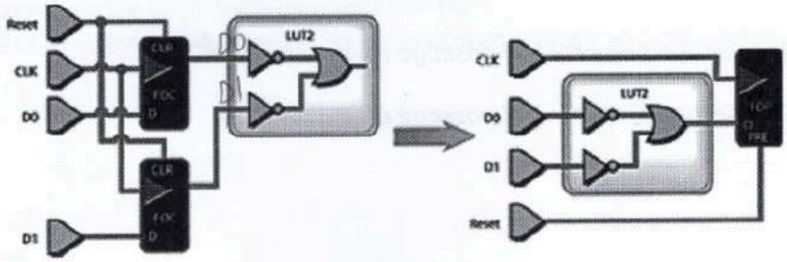
Lower-power SEC

4. [9%] Lab08:

(a) [4%] Please describe the difference between **SEC** (Sequential Equivalence Checking) and **CEC** (Combinational Equivalence Checking)?

(b) [2%] Please answer you will use SEC or CEC in each situation below:
(1): Compare design after retiming, design after gating.
(2): Compare RTL design, Gate-level netlist and layout netlist

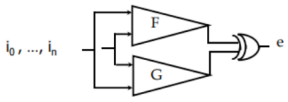
(c) [2%] If we use the SEC to check the design in figure, will the SEC report equivalent or not?



(d) [1%] What does "**bbox_m**" mean and when will we use this command?

(a)

✓ **Combinational Equivalence Checking (CEC, LEC)**
- Same functional behavior at all external ports and internal state element?
- Requires state-matching design
- Performs EC only on the remaining logical cones



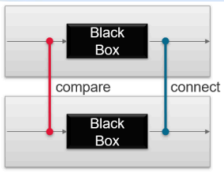
✓ **Sequential Equivalence Checking (SEC)**
- Same functional behavior at all external ports?
- Applies to state-matching and non-state-matching designs
- Ignores internal sequential differences

(b) (1) use SEC; (2) use CEC

(c) It will be equivalent.

(d) If we are sure that the internal logic of a block is equivalent, we can use -bbox_m to black-box it. SEC will compare the black-box inputs and connect the black-box outputs.

✓ Black-Box hides functionality



- Internal logic of block is removed
- Done via the elaborate stage (-bbox_m or –bbox_i)

✓ Meanings for SEC

- Black-Box outputs are considered inputs, and need to be connected
- Black-Box inputs are considered outputs, and need to be compared (assertions)
- Black-Box ports are automatically mapped by SEC

✓ Notes

- Black-Box modules must be guaranteed to be equivalent
- If Black-Box input assertions fail, the Black-Box output connections are no longer valid
- Output connections are theoretically valid up to the cycle in which the input assertions fail

System Verilog - Basic

(13) List at least two enhancement in design using system verilog , and explain what is the advantage. (4%)

1. use **logic** to replace reg and wire: no input/output restriction, no continuous or block procedure restriction
2. **typedef**: allow users to create their own names for type definitions to make code clear
3. **struct**: group related signals to enhance readability and clearly convey
4. **enum**: defines a set of named values, which provide built-in assertion. Often use to represent state machine
5. **union**: allow a single piece of storage to be represented different ways using different named member types
6. **package**: enable sharing a user-defined type definition across multiple modules
7. **always_comb, always_ff, always_latch** to replace always block: show designers intent
8. **interface**: encapsulate communication btw design blocks and btw design and verification blocks

(14) List at least two enhancement in verification using system verilog, and explain what is the advantage. (4%)

1. **OOP**: Each object should do what it should do, rather than several roles. Interaction between object and the object should be reduced as much as possible.
2. **Randomization**: Verify with sufficient set of vectors to gain a level of confidence that product will ship with a tolerable field-failure rate. The best known mechanism is randomization – randomization of data and using constraints to narrow the scope.

(15) Please check if there are syntax errors in the following code. Correct the error if there are existed errors.(8%)

```
1. enum {IDLE, XX='x, S1=2'b01, S2=2'b10} state, next;
2. enum integer {IDLE, XX='x, S1='b01, S2='b10} state, next;
3. enum integer {IDLE, XX='x, S1, S2} state, next;
4. enum {a=3, b=7, c} alpha;
5. enum bit [0:0] {a,b,c} alphabet;
6. enum {a=0, b=7, c, d=8} alpha;
7. enum bit [3:0] {bronze='h3, silver, gold='h5} medal4;
8. enum bit [3:0] {bronze=4'h3, silver, gold=4'h5} medal4;
```

1. default data type int is 2-state, cannot have "x" value

   -> enum logic [1:0] {IDLE=2'b00, XX=2'b01, S1=2'b10, S2=2'b11} state, next;

2. no error
3. An enumeration constant with X or Z assignments (XX) is followed by an unassigned enumeration constant.

   -> enum integer {IDLE, XX='x, S1=2'b01, S2} state, next;

4. no error
5. [0:0] should be replaced by [1:0]
6. c and d has the same value

   -> enum {a=0, b=7, c=8, d=9} alpha;

7. no error
8. no error

✓ Create enumerated data types:

- Defines a set of named values, which provides built-in assertion!!
- Data type defaults to ***int*** (32-bit, 2state, signed int)
- Variable initialized to 0 if initial values aren't specified

5. [4%]:
   (1) Please describe the major functionality of “typedef” syntax.
   (2) Please describe the major functionality of “enum” syntax.

typedef: allow users to create their own names for type definitions to make code clear.
Enum: defines a set of named values, which provide built-in assertion. Often use to represent state machine.

5.

A) [6%] Why is Typedef used in systemVerilog?
B) [6%] Write a representation of hBytes if “reg” is used as a datatype instead of hexOctet

```
typedef reg [7:0] octet;

typedef octet [15:0] hexOctet;

hexOctet hBytes [3:7];
```

5a) 讓Designer 可以自行定義Type、讓程式碼可讀性更高、加快開發速度

5b) reg [15:0][7:0] hBytes [3:7]

5. [20%] For (a) to (g), please fill in the blank
For (h) to (k), please answer the following question

(h) : how many packed array is declared in ALU.sv? how many bits is used to generate these packed array?

(i) : please indicate the problem of type FSM_STATE.

(j) : please indicate the problem of type STAGE.

(k) : please rewrite Piece.v so that we can put it inside the program of PATTERN.sv

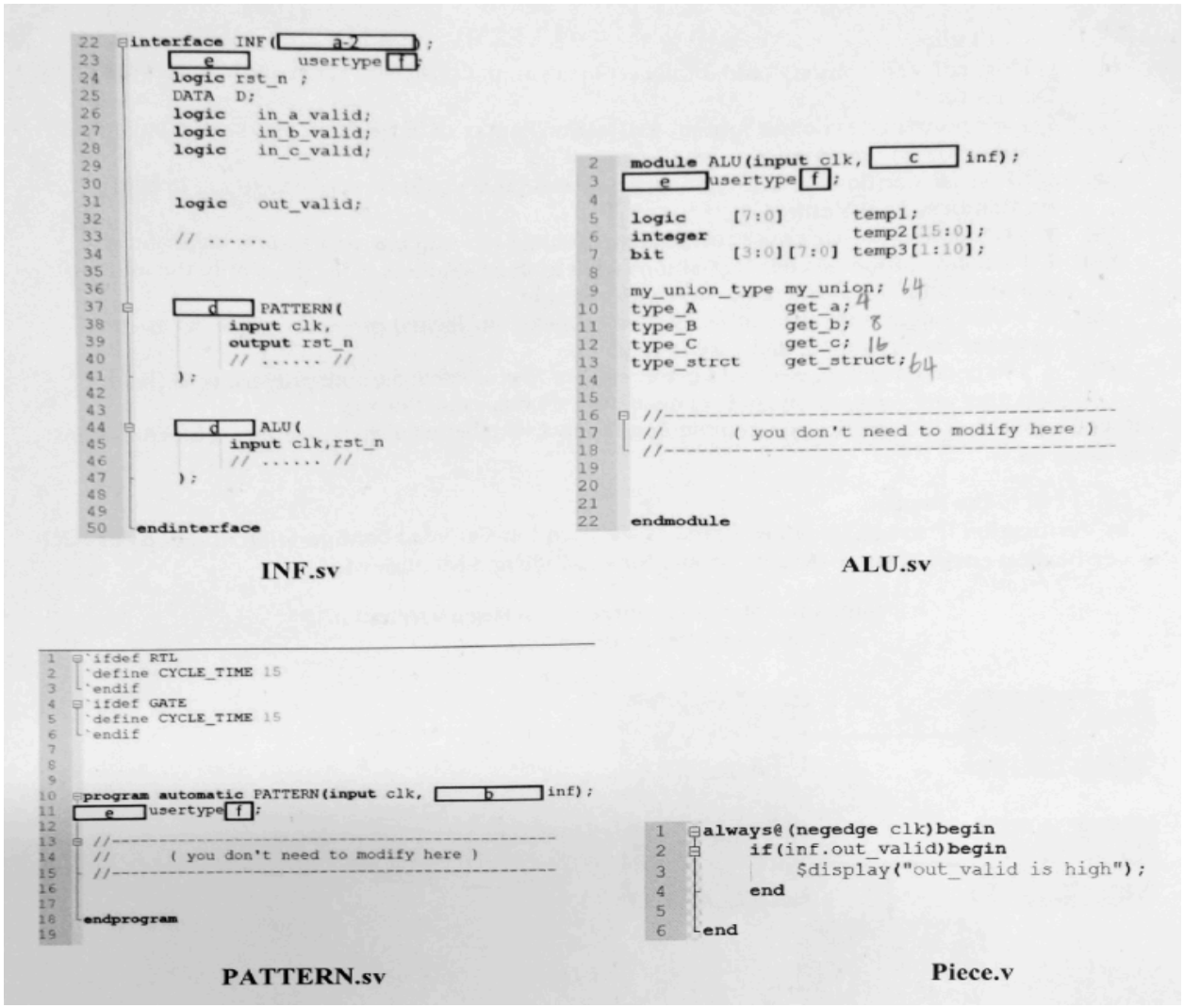
Note: a-1 and a-2 are related to each other. Please write them both on your answer sheet. Please clearly label your answer from (a-1), (a-2) to (k). (a)-(g), (h)(k) [2%] (i), (j) [1%]

```
1  `timescale 1ns/100ps
2
3  `include "Usertype_PKG.sv"
4  `include "INF.sv"
5  `include "PATTERN.sv"
6  `include "ALU.sv"
7
8
9  module TESTBED;
10
11   parameter simulation_cycle = 20;
12   reg  SystemClock;
13
14   INF      inf( a-1 );
15   PATTERN test_p(.clk(SystemClock), .inf( b ));
16   ALU      dut_b (.clk(SystemClock), .inf( c );
17
18 //--------------- Generate Clock ----------------
19 //      some clk generated block definitioin
20 //       ( you don't need to modify )
21 //-----------------------------------------------
22
23 endmodule
```

Testbed.sv

```
21 `ifndef USERTYPE
22 `define USERTYPE
23
24 g usertype;
25
26 typedef enum { S_IDLE   =  3'd0,
27                S_a      =  3'd1,
28                S_b      =  3'd2,
29                S_c      =  3'd3,
30                S_d      =  3'd4
31              } FSM_STATE;
32
33 typedef enum { S_e = 2, S_f = 3, S_g, S_h = 4 } STAGE;
34
35 typedef logic [3:0] type_A;
36 typedef logic [7:0] type_B;
37 typedef type_B[1:0] type_C;
38
39 typedef struct packed {
40     integer     temp_int;
41     type_C      c;
42     logic[15:0] d;
43 } type_strct ;
44
45 typedef union packed{
46     type_strct    mem_struct;
47     type_A        [15:0] mem_a;
48     type_B        [7:0]  mem_b;
49     type_C        [3:0]  mem_c;
50 }my_union_type;
51
52 /*  some other type definition */
53
54 end g
55
56 `endif
```

Usertype_PKG.sv



```
interface INF( a-2 );
    e      usertype f;
    logic rst_n ;
    DATA  D;
    logic   in_a_valid;
    logic   in_b_valid;
    logic   in_c_valid;

    logic   out_valid;

    // ......

    d  PATTERN(
        input clk,
        output rst_n
        // ..... //
    );

    d  ALU(
        input clk,rst_n
        // ..... //
    );

endinterface
```

INF.sv

```
module ALU(input clk, c inf);
e usertype f;

logic    [7:0]       temp1;
integer              temp2[15:0];
bit      [3:0][7:0]  temp3[1:10];

my_union_type my_union;
type_A        get_a;
type_B        get_b;
type_C        get_c;
type_strct    get_struct;

//------------------------------------------
//      ( you don't need to modify here )
//------------------------------------------


endmodule
```

ALU.sv

```
`ifdef RTL
`define CYCLE_TIME 15
`endif
`ifdef GATE
`define CYCLE_TIME 15
`endif


program automatic PATTERN(input clk, b inf);
e usertype f;

//------------------------------------------
//      ( you don't need to modify here )
//------------------------------------------


endprogram
```

PATTERN.sv

```
always@(negedge clk)begin
    if(inf.out_valid)begin
        $display("out_valid is high");
    end

end
```

Piece.v

5.

a-1: SystemClock,a-2: input clk ,b: inf.PATTERN; c: inf.ALU, d: modport, e: import, f: ::*, g: package

h: temp1, temp3, my_union, get_a, get_b, get_c, get_struct

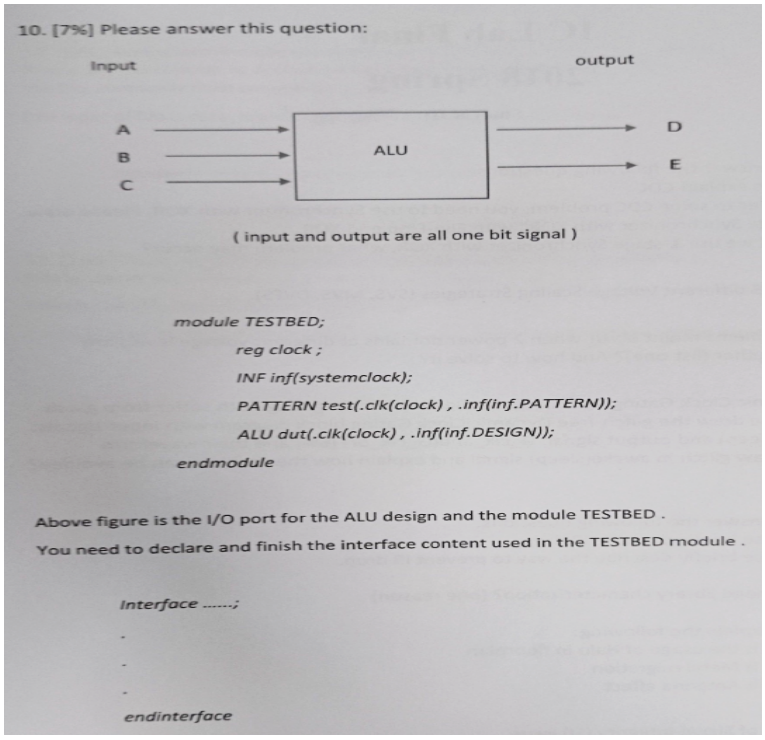
i: enum default type is int (32 bits, signed, 2-state)

j: S_g=4 and S_h=4

k:

```
initial begin
        forever@(negedge clk) begin
                if(inf.out_valid) $display("out_valid is high");
        end
```

end


10. [7%] Please answer this question:

Input output

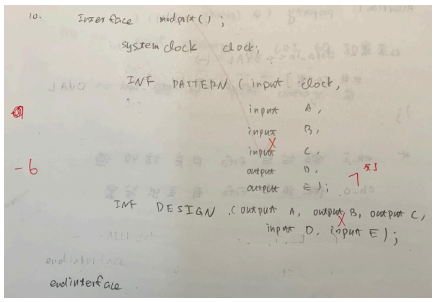
(input and output are all one bit signal)

```
module TESTBED;
    reg clock ;
    INF inf(systemclock);
    PATTERN test(.clk(clock) , .inf(inf.PATTERN));
    ALU dut(.clk(clock) , .inf(inf.DESIGN));
endmodule
```

Above figure is the I/O port for the ALU design and the module TESTBED .
You need to declare and finish the interface content used in the TESTBED module .

```
Interface ......;
.
.
.
endinterface
```

```
interface INF(input bit clock);
        logic A, B, C, D, E;
        modport PATTERN(
                output A, B, C, clock,
                input D, E );
        modport DESIGN(
                input A, B, C, clock,
                output D, E );
endinterface
```



6. [10%]:
   (1) Please explain what are pre_randomize() and post_randomize() function in every class? How to use these functions?
   (2) Please write the corresponding code if we want to create a class called "random_val" which contains a random member variable called "rand_val", which is a 32-bits variable. The possible values of this variable are 32'h10000000, 32'h01000000 and 32'h00000001 and these values need to be cyclic randomized without repeating.

pre_randomize(): set/correct constraints before randomization
Post_randomize(): make corrections after randomization
The pre_randomize is auto called before the randomize() , we can use it to check the variable used in random function correct or not.
The post_ranomize is auto called after randomize(), we can use it to check the variable value or doing some data processing after randomize.

```
class random_val;
        randc rand_val_1;
        logic [31:0] rand_val;
        constraint range{
                rand_val1 inside{16'h1000, 16'h0100, 16'h0001};
        }
        always_comb @(*) begin
                rand_val = {rand_val_1[15:8], 16'h0000, rand_val_1[7:0]};
        end
endclass
```

應該是我寫的那樣吧寶

看上面

但我不知道怎麼call function吐了

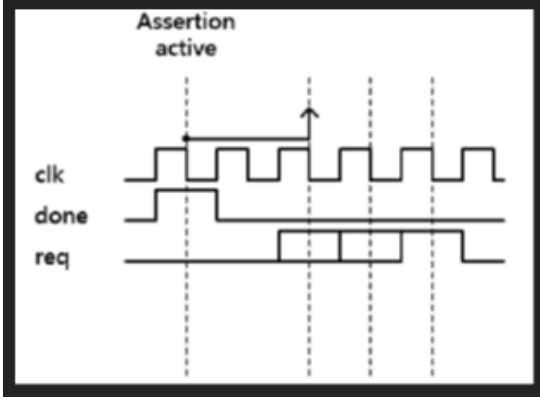
SystemVerilog Verification & Assertions

[2pts] Please explain the meaning of the following assertion (better with waveform):

```
property next_req;
   @(negedge clk) done === 1 |=> ##[1:3] req == 1;
Endproperty
```

Ans:

This assertion will check at negative edge of *clk*. For non-overlapped implication, the consequent sequence expression is evaluated on the next clock tick. So, this assertion checks that whenever *done* is asserted, *req* must be asserted after two cycles, or the following 2 cycles.



8. [6%]:

Given the verilog code shown below, calculate the total toggle coverage percentage of register 'data' after simulation. Please briefly explain your reasons.

```
reg  [4:0] data;

initial
begin
        data = 8;
        #50;
        data = 18;
        #50;
        data = 10;
        #50
        data = 27;
        #50
        data = 19;
end
```

Chapter 2. Coverage Metrics (sourceforge.net)

Lec10 P.5

Data = 5'b0_1000;

Data = 5'b1_0010;

Data = 5'b0_1010;

Data = 5'b1_1010;

Data = 5'b1_0011;

Data[0]: 1/2, Data[1]: 1/2, Data[2]: 0/2, Data[3]: 2/2, Data[4]:2/2

⇨ 6/10 = 60%

⇨ The register called "a", after being simulated, would have achieved a total toggle percentage of 50% (or 3 out of 6). Can you see which toggles are missing?

⇨ Bit 0 has never toggled to 1 (and has never been at a value of 1 and toggled to 0); therefore, bit-0 has toggled 0% (0 out of 2). Bit 1 has toggled from a value of 0 to 1, but has not toggled from a value of 1 to 0; therefore, bit-1 has toggled 50% (0 out of 2). Bit 2 is the only bit that has fully toggled, achieving a toggle percentage of 100% (2 out of 2). If you add all of the possible toggles to the number of achieved toggles, you end up with 3 out of 6 (or 50% total toggle coverage).

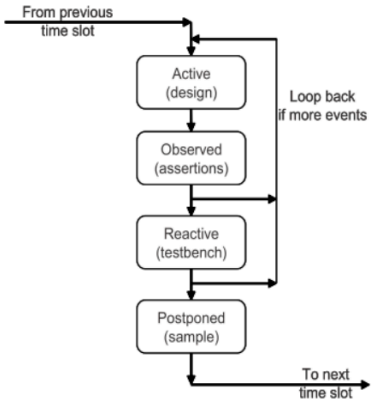
⇨

5. [6%] Lab09:

In SystemVerilog, why do we generate the pattern by using "**program**" block instead of "**module**"? Please also provide an example to illustrate your answer.

- **Purpose: Identifies verification code**
- **A program differs from a module**
  - Only initial blocks allowed
  - Execute in reactive region

```
program name (<port_list>);
  <declarations>; // type, func, class, clocking...
  <continuous_assign>
  initial <statement_block>
endprogram
```



One of the key features of the `program` construct is the ability to use concurrent sequences and randomization for stimulus generation. Concurrent sequences are sequences of events that can occur concurrently, allowing you to model complex scenarios more naturally than with the strictly procedural code that you would use inside a `module`.

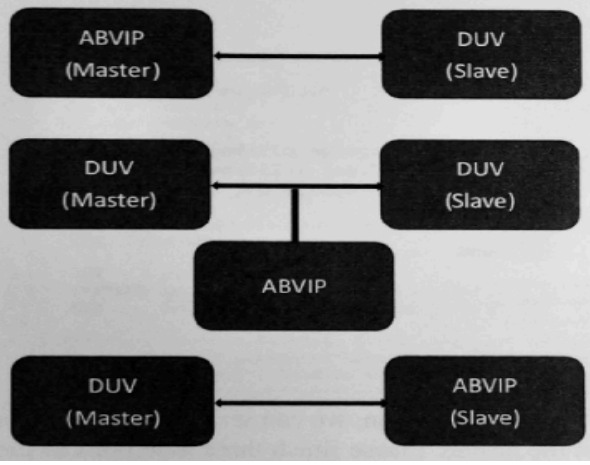
12. [7%] True or False:
    (1) ( ) Formal Verification could totally replace simulation-based verification in current IC Design flow.
    (2) ( ) We normally write our formal verification syntax in our Pattern.sv to setup our verification environment.
    (3) ( ) Formal Verification is a verification method that could be applied only in design written in system Verilog
    (4) ( ) Generally, we use cover , assert , and assume to setup our verification environment.
    (5) ( ) Assume properties tell formal tool what kind of stimulus is legal, so only the matched stimulus will be considered to verify our design.
    (6) ( ) Glue Logic is an alternative method to make our formal properties (such as assume, assert or cover) simple and easy to read.
    (7) ( ) In code coverage, we could comment out few of some assume properties to check whether our verification environment is over-constrained or not.

Hint: code coverage is a verification methodology to test whether each statement/branch/expression ha be covered.

(1) F (2) F (3) F (4) T? (5) T? (6) T (7) F

13. [3%] Fill in the Blanks:
In Verification IP usage, there are 3 kind of Verification-Oriented configuration to set up our AXI-verification environments. (Master Mode, Slave Mode and Monitor Mode)

Hint: ABVIP stands for Abbreviation Based Verification IP
DUV stands for Design Under Test



1. ____ Mode

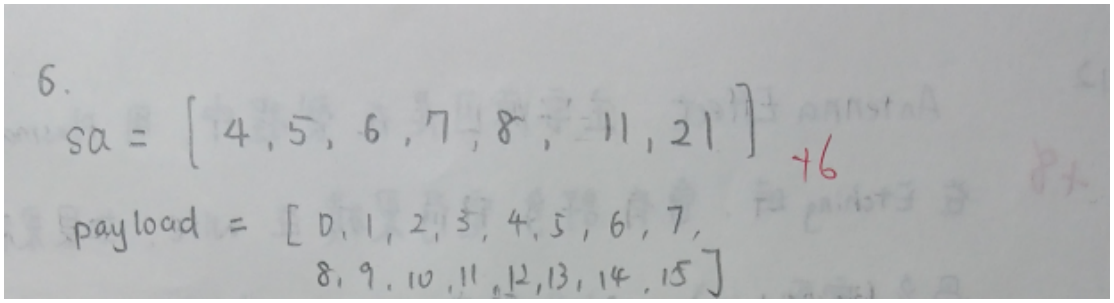
2. ____ Mode

3. ____ Mode

1.Master Mode, 2.Monitor Mode, 3.Slave Mode

6. [6%] What values are constrained by “Limit”? (List them out)

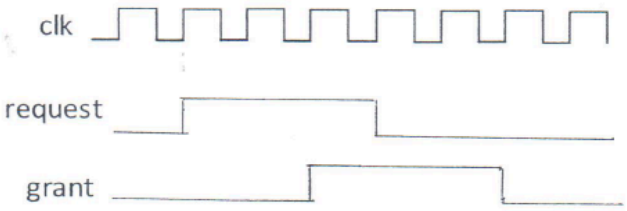
```
constraint Limit{
sa inside { [4:8], 11, 21};
payload.size() <= 4;
}
```

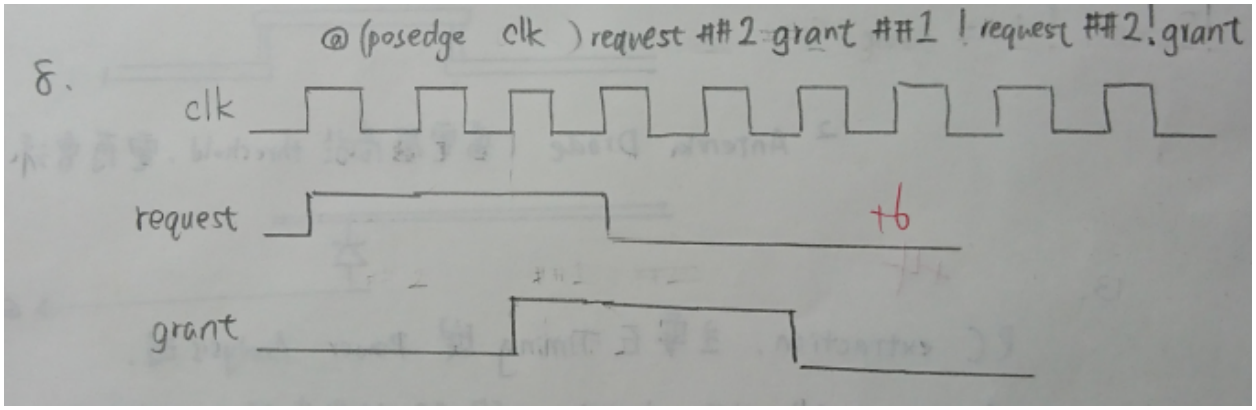


The payload is a 4 bits variable, thus there is 2^4 possible value for variable payload.

8. [6%] Fill in the waveform below given the property:

```
property p_request_grant;
        @(posedge clock) request ##2 grant ##1 !request ##2 !grant
endproperty
```





6. [10%] Lab09:

(a) [4%] What is the advantage of using System Verilog? Please list 1 advantage and give a eample.

(b) [6%] Please complete the code such that the class random_obj has an **8-bit** member variable named mem whose value is limited within the range below and the random function of mem will exhaust all values before repeating any individual value.

Range = 7, 8, 9, 10, 11, 12, 31, 35, 37

constraint{ inside }

```
class random_obj;
    // complete the code here
endclass
```

randc

(a) **enumerate**: defines a set of named values, which provide built-in assertion. Often use to represent state machine

Verilog:

parameter IDLE=2'b00; parameter IN_DATA=2'b01; EXE=2'b10; OUT=2'b11;

reg [1:0] state, n_state;

SystemVerilog:

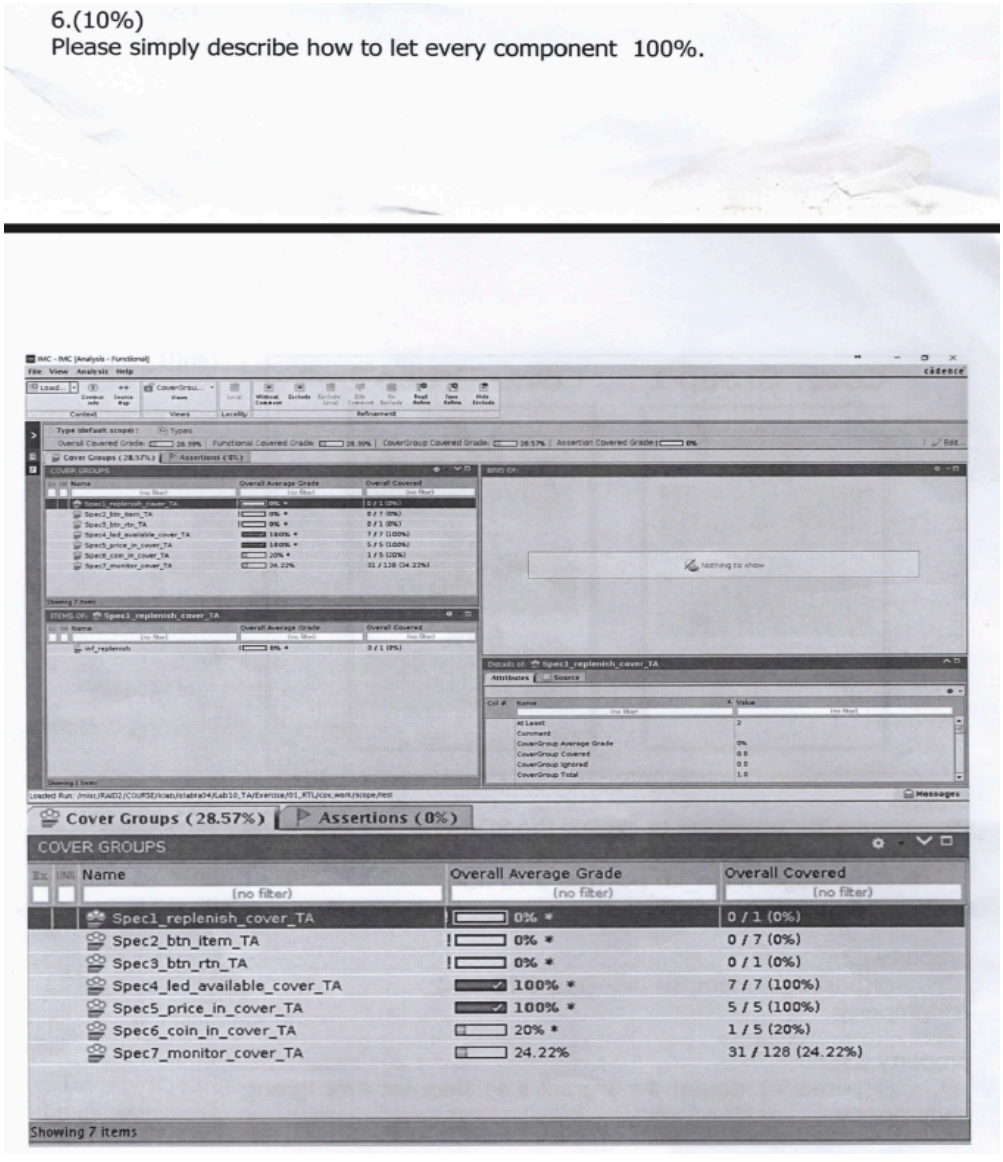
typedef enum logic[1:0] {IDLE, IN_DATA, EXE, OUT} state_t;

state_t state, n_state;

b)

```
class random_obj;
        randc logic[7:0] mem;
        function new (int seed);
            this.srandom(seed);
        endfunction
        constraint limit{ mem inside {7,8,9,10,11,12,31,35,37}; }
endclass
```

-

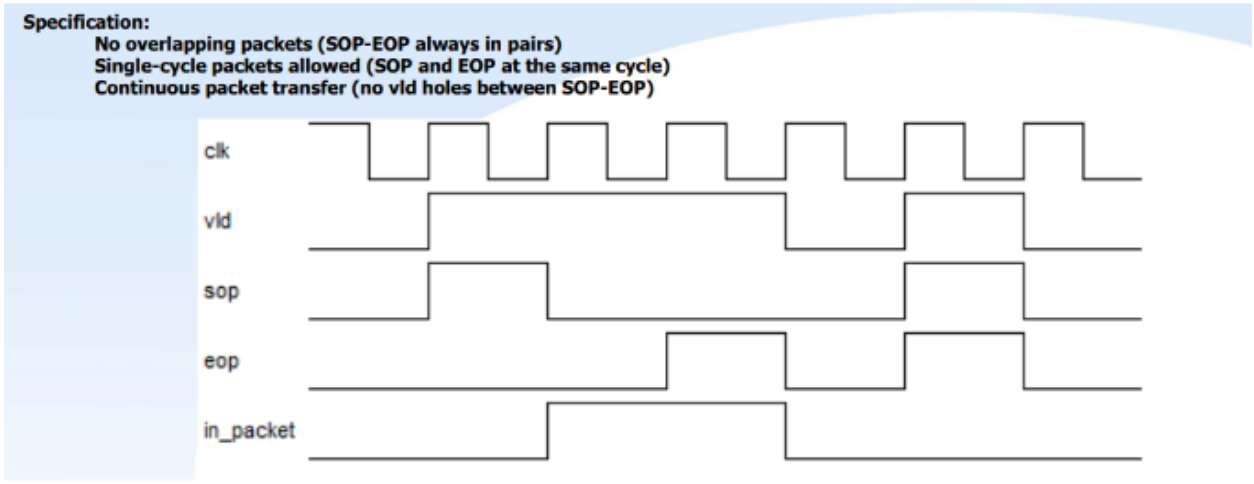


(?)

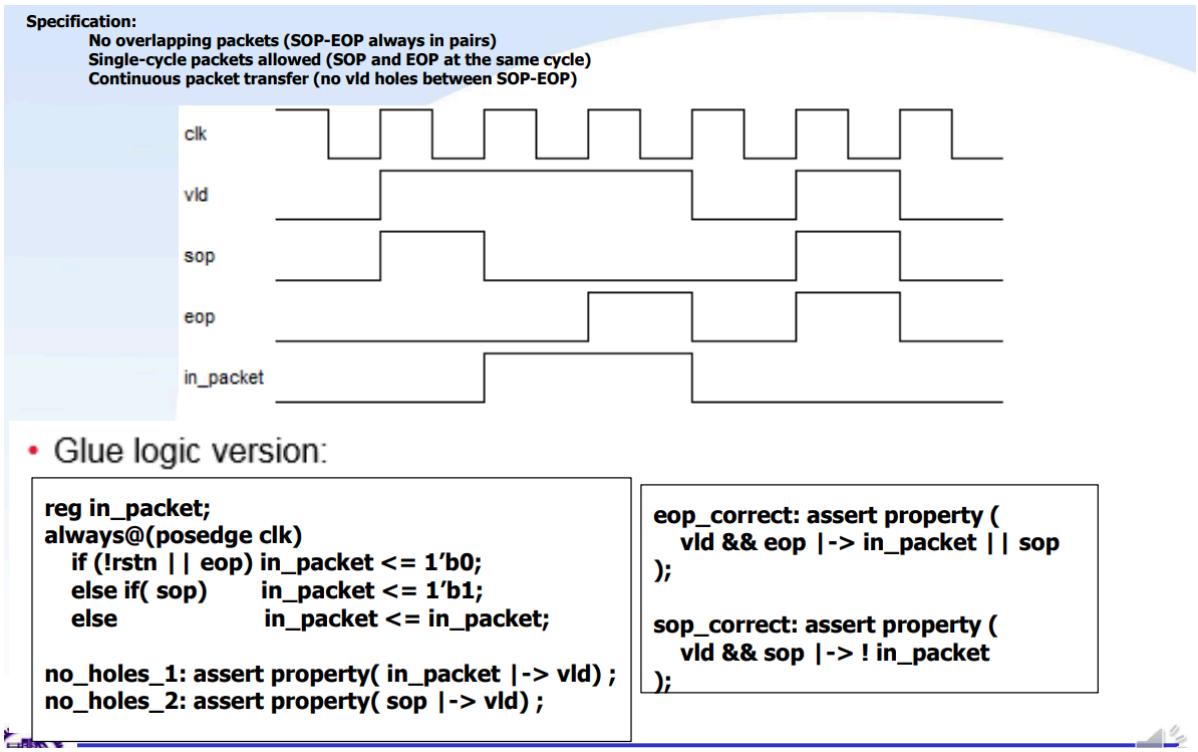
increase pattern coverage, e.g. increase pattern numbers or using randc, to hit the coverage more

check if the design is possible to hit all the coverage spec

2023 Fall(Q): Complete the glue logic and assertions for SOP/EOP interface~



A:



趁快忘記前記起來ww造福學弟妹XD by Jacky Yeh

System Verilog - Assertion

7. [8%] Lab10:

If we want to check whether our design hits the specification, we can write the assertion in the checker.sv file. There are two signals, A and B in the design, and two signals are triggered when posedge clk. Please finish the two assertions below. Notice that you cannot use always block here.

(a) [4%] If A changes from 0 to 1 between one rising clock, then 2-5 cycles later, B should be high for 5 cycles. (Hint: *$rose()* could be used)

(b) [4%] If A is 0, B should be high 3 clock cycles ago.

Assertion1: assertion property (@(posedge clk)) ..... ;

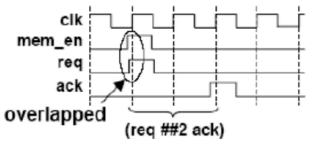
Assertion2: assertion property (@(posedge clk)) ..... ;

(a) Assertion1: assert property(@(posedge clk) $rose(A) |-> ##[2:5] B[*5]);
(b) Assertion2: assert property(@(posedge clk) !A |-> ($past(B,3) == 1));

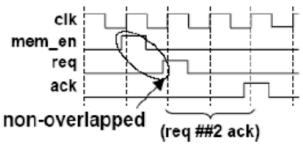
5.(9%)
Please draw the waveform to explain the following two assertions

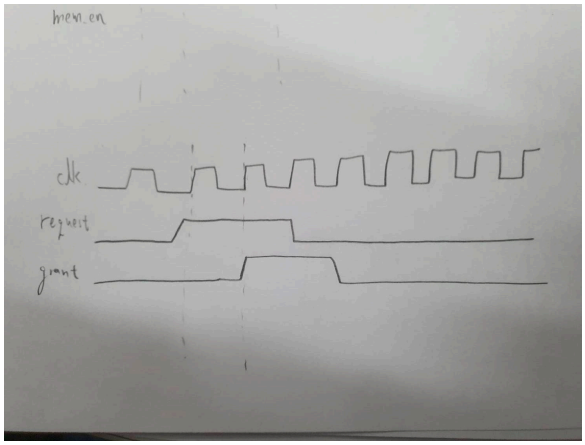
```
property q1;
        @(posedge) mem_en |-> (req ##3 ack);
endproperty

property q2;
        @(posedge) mem_en |=> (req ##3 ack);
endproperty

property q3;
        @(posedge) request ## 1 grant ##1 !request ##1 !grant;
endproperty
```

```
property p_req_ack;
 @(posedge clk) mem_en |-> (req ##2 ack);
endproperty: p_req_ack
```




```
property p_req_ack;
 @(posedge clk) mem_en |=> (req ##2 ack);
endproperty: p_req_ack
```





7. [9%] If we want to check whether our design hits the specification, we can write the assertion in the check file. There are two signals, A and B, in the design. Please finish three assertions as the below.
   (1) A happens then 2 or more cycles later B happens
   (2) If A happens, then B must happen at the same cycle.
   (3) If A happens, then B must happen at the next cycle.

```
Assertion1: assertion property (@( posedge clk )) ..... ;
Assertion2: assertion property (@( posedge clk )) ..... ;
Assertion3: assertion property (@( posedge clk )) ..... ;
```

(1) assert1: assert property(@(posedge clk) B |-> ##[2:$] A);
(2) assert2: assert property(@(posedge clk) A |-> B);
(3) assert3: assert property(@(posedge clk) A |=> B);

7. [12%]:

If we want to check whether our design hits the specification, we can write the assertion in the checker.sv file. There are three signals, A, B, C, in the design, and three signals will only be high consecutive unfixed cycles. Please finish the three assertions below. Notice that you cannot use always block here.

(1) If A happens , 1 or more cycles later B should be 0 when C happens.

(2) If A changes from 1 to 0 between one rising clock, then B must happen 10 times in a row after 2-10 cycles.

(3) if A is 0, B was high before 2 clock cycles.

Assertion1: assertion property (@(posedge clk)) ..... ;
Assertion2: assertion property (@(posedge clk)) ..... ;
Assertion3: assertion property (@(posedge clk)) ..... ;

Lec 10 P.26 P30. P.31

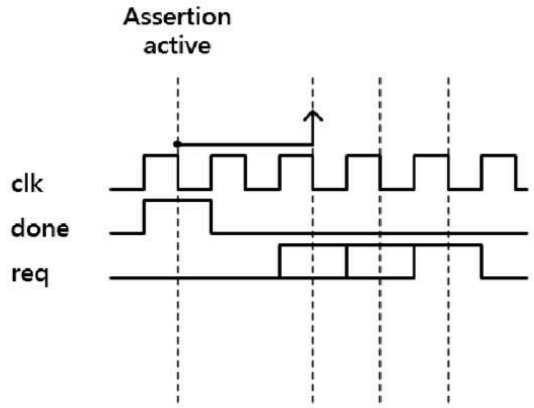
1. Assertion 1: A |-> ##[1:$] (C && !B);
2. Assertion 2: $fell(A) |-> ##[2:10] B[*10]
3. Assertion 3: !A |-> ($past(b,2) == 1);

[2pts] Please explain the meaning of the following assertion (better with waveform):

```
property next_req;
    @(negedge clk) done === 1 |=> ##[1:3] req == 1;
endproperty
```

Ans:

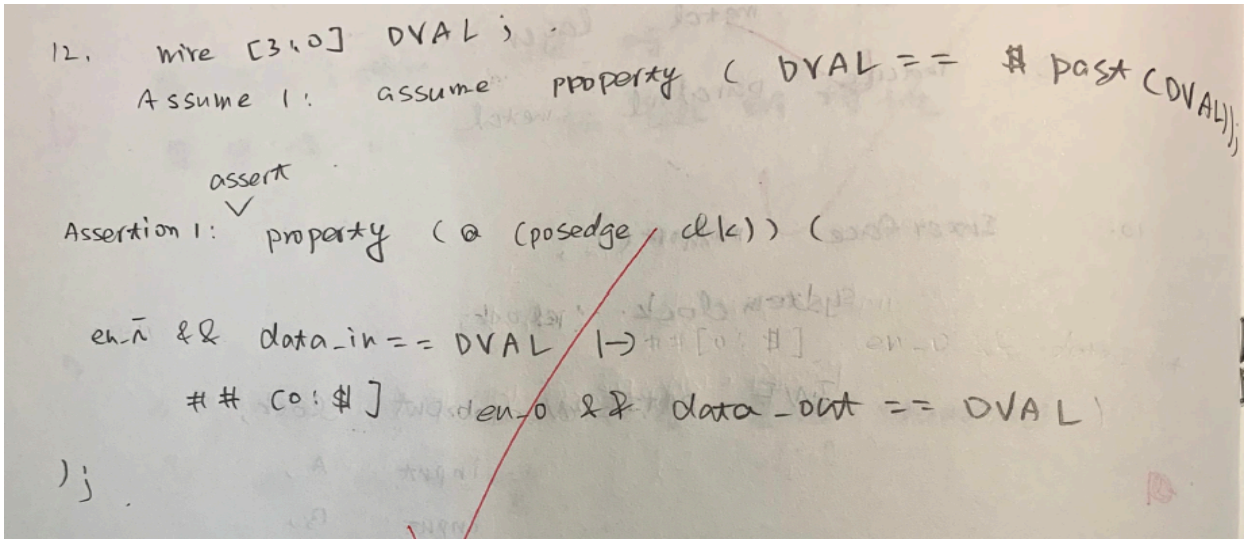
This assertion will check at negative edge of *clk*. For non-overlapped implication, the consequent sequence expression is evaluated on the next clock tick. So, this assertion checks that whenever *done* is asserted, *req* must be asserted after two cycles, or the following 2 cycles.



12. [9%] Nondeterminnistic constant is very useful during formal testing.
Please use this concept to declare a 4 bits signal DVAL and write an assertion to check if data comes in the fifo, eventually must come out .

(the input of fifo is data_in and output of fifo is data_out , they are also 4 bits)

Assertion1: assertion property (@(posedge clk)) ..... ;



System Verilog - Coverage

6. [6%] What are code coverage and functional coverage?

Coverage Models

- **Function Coverage**
  - Property related covers (Lab09)
  - Covergroups (Lab09)

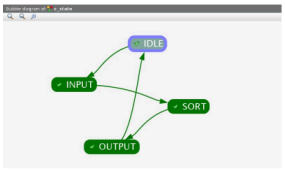
Pros:
Realizable!
Cons:
Corner case!
Time consuming!

- **Code Coverage**
  - Branch
  - Statement
  - Expression

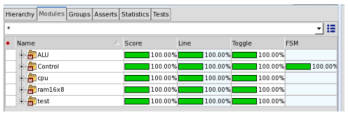
Pros:
Automatically gen.!
Fully covered
Cons:
False alarm issue

Code Coverage

- ✓ **Statement (line) coverage**
- ✓ **Block coverage**
- ✓ **Conditional/Expression coverage**
- ✓ **Branch/Decision coverage**
- ✓ **Toggle coverage**
- ✓ **FSM coverage**



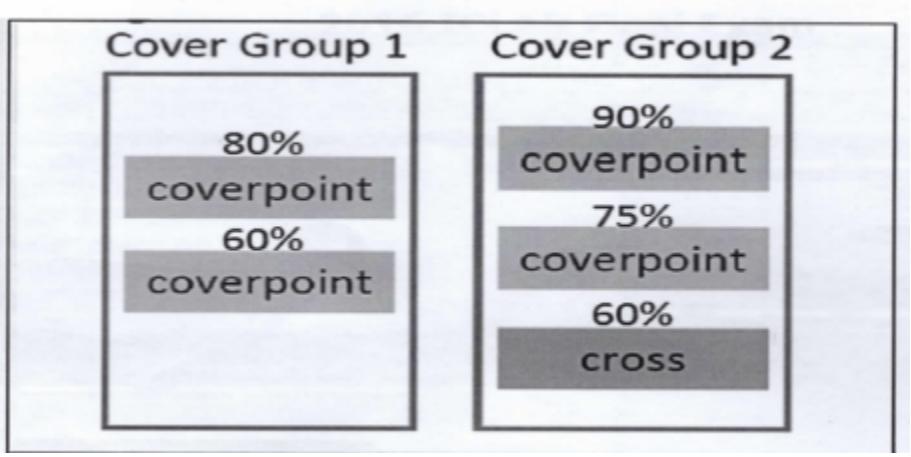

```
always @(posedge clock)
  begin                    => Block 1 [always block]
    if(x == y) begin       => Block 2 [If block]
      out1 = x+y;
      out2 = x^2 + y^2;
    else                   => Block 3 [Else block]
      out1 = x;
      out2 = y;
  end
```



Functional Coverage

- ✓ **Sample points are known as cover point**
- ✓ **A cover point can be an integral variable or an integral expression.**
- ✓ **Multiple cover points sample at the same time are placed together in a cover group**
- ✓ **A cover group can sample any visible variable directly such as program variables, signals from an interface, or any signal in the design (using a hierarchical reference). (see Appendix A.)**

4. (10%)
Calculate the coverage of the whole design, the coverage of coverpoints or cross is specified as shown on the picture



covergroup1: 0.5*80% + 0.5*60%

covergroup2: 0.33*90% + 0.33*75% + 0.33*60%

Total = 72.5%

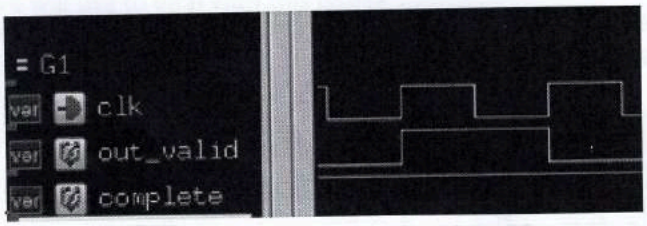
7. [6%] How many state bins are created:
   a. bins s0 = {[0:4]}
   b. bins s1 [] = {[3:9]}

7a) 1個bin

7b) 7個bins

8. [8%] Lab10:

(a) [6%] If we want to create a covergroup *cg1* and sample the value of signal "*complete*" at negedge clk when the out_valid if high, where the waveforms of out_valid and "*complete*" are shown below. What is the problem when using the code below? How can we correct it?




```
covergroup cg1@(negedge clk && out_valid);
      ......
endgroup
```

(b) [2%] How can we get the coverage of *cg1* during the simulation (without using other EDA tools like imc)?

(a) covergroup cg1 @(negedge clk iff(out_valid));

(b) $get_coverage()

✓ **$get_coverage() returns testbench coverage percentage as a *real* value**

```
repeat (10) begin
  addr = $urandom_range(0,7);
  // Sample the covergroup
  my_cov.sample();
  #10 ;
end
// Stop the coverage collection
my_cov.stop();
// Display the coverage
$display("Instance coverage is %e",my_cov.get_coverage());
```

13. [10%] There are different kind of coverage report in the jaspergold .
Please specify the concept of COI and proof coverage ?

What is the difference between them ?

If we want to get the COI or proof coverage report , do we need to prove all the assertions ?

- **Each assertion affected by some cover items**
  - The region will be called Cone-Of-Influence (COI)
- **Finds the union of the all assertion COIs**
  - The remaining Out of COI cover items indicate holes in the assertion set – code that is not checked by any asserts
  - Fast measurement – no formal engines are run

COI

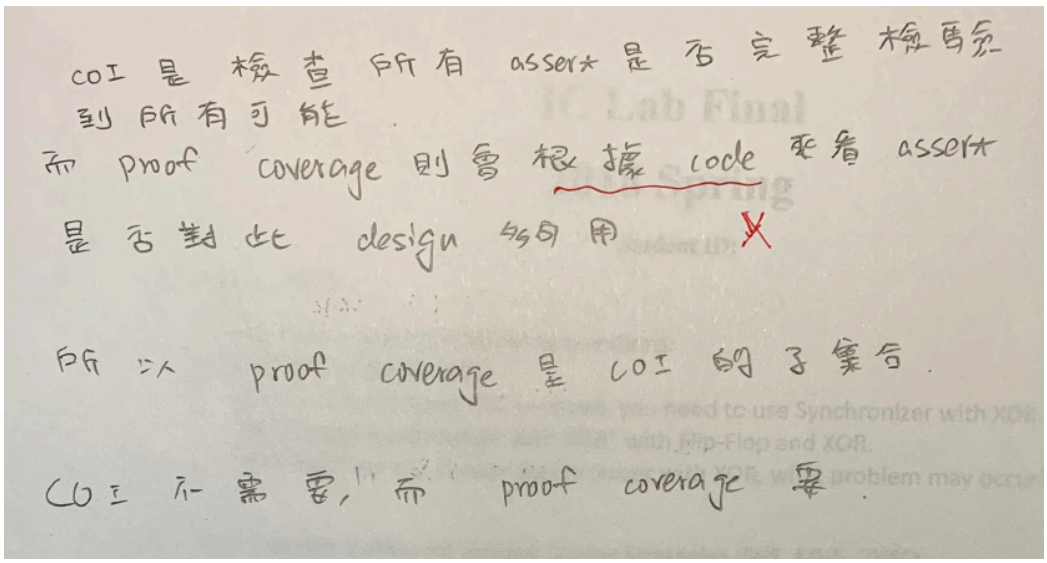
- **Find the region cannot truly influence assertion status**
  - Example:
  - assign a = b+c; } COI
  - assign m = a; } coverage } Proof coverage
  - **assert** property (@(posedge clk) a == m);
- **Represents the portion of the design verified by formal engines**
  - Subset of the COI – COI represents the maximum potential of proof coverage

Proof

Proof vs. COI analysis

- **Proof coverage is a subset of the COI**
  - COI represents the maximum potential of proof coverage
- **COI doesn't require a proof to take place, while proof coverage does.**

Difference



report – COI: No, Proof: Yes

5. Following is a covergroup declaration:

```
covergroup final @(posedge clk);
    option.per_instance = 1;
    option.at_least = 100;
    signal0: coverpoint inf.signal_0{
    bins signal_0 [] = ([0:4] => [0:4]);
    }
    signal1: coverpoint inf.signal_1{
    bins signal_1 = 1 ;
    }
    signal2: coverpoint inf.signal_2{
    option.auto_bin_max = 16;
    }
    signal3: coverpoint inf.signal_3{
    bins s3_0 = 0;
    bins s3_1 = 1;
    bins s3_2 = 2;
    }
    cross signal1 , signal3 ;
endgroup
```

(a) If signal_2 is a 10-bit signal, how many numbers will contain in one bin (in coverpoint signal2)? (3%)

(b) In what condition that this covergroup will achieve 100% coverage? (You need to write down each condition clearly) (15%)

(a) $2^{10} = 1024$ numbers will be divided into 16 groups, thus each group has 1024 / 16 = 64 numbers.

(b) To make the covergroup have 100% coverage, you have to have 100% coverage for all coverpoints and coverpoint crosses.
All the following are checked at every clock rising edges:

signal0: inf.signal_0 changes from 0→0, 0→1, 0→2, … 4→2, 4→3, 4→4, totally 25 changes (each change corresponds to one bin) should occur at least 100 times for each transition respectively.

signal1: inf.signal_1 should equal to 1 at least 100 times.

signal2: inf.signal_2 should belongs to 0~63, 64~127, … 960~1023, totally 16 groups (bins) at least 100 times for each group respectively.

signal3: inf.signal_3 should equal to 1 at least 100 times, 2 at least 100 times and 3 at least 100 times.

cross signal1, signal3: inf.signal_1 should equal to 1 and inf.signal_3 should equal to 1 at the same clock rising edge at least 100 times; inf.signal_1 should equal to 1 and inf.signal_3 should equal to 2 at the same clock rising edge at least 100 times and inf.signal_1 should equal to 1 and inf.signal_3 should equal to 3 at the same clock rising edge at least 100 times.

Note: if you give the correct bin numbers but without description, you will get 1 deducted point for each coverpoint / croverpoint cross.

6. For using JasperGold in Lab10, you can choose to show the COI results and proof core results with the provided assertions, describe the difference between these two results. (5%)

COI (cone of influence) results will highlight the cover items which may affect the assertion results based on code tracing. In real case however, not all cover items in COI contribute to the asserted expressions when proving. Only part of them does. The cover items which help proving the assertion correctness are called proof core. Therefore the proof core will be a subset of COI.

For the proof core result shown in Lab10, is the proof core result of the coverage, or called proof core coverage, proof coverage since the result is shown in Coverage tab. This will only highlight the expressions in proof core which make the assertion correct.

Note 1: COI and COI coverage usually refer to the same item. Whereas proof core refers to expressions which help verifying the assertion correctness; but proof core coverage refers to expressions which make assertion correct. When all assertions are correct, the proof core and proof core coverage refers to the same item.

Note 2: You have to mention (1) which kind of cover items (or the conditions) will be highlighted in COI and (2) the reason why the proof core is a subset of COI. Or else you will get some deducted points.

APR

9. [12%] Lab11:

(a) [6%] The following files: LEF, captbl, CDB, GDS are required for APR, please explain the content or purpose of these files.

(b) [4%] Please explian what is antenna effect, and list 1 solution for this.

✔ **LIB Library : timing information**
- Calculate cell delay according to different combinations of input slew rate and output loading capacitance
- Using SignalStorm Library Characterizer

✔ **LEF Library : process and cell information**
- Metal layer information, routing pitch, default direction
- Cell size, pin position, pin metal layer

✔ **RC Extraction**
- RC extraction for further power analysis, simulation etc

✔ **CeltIC Library**
- cdB model (noise model for each cell which will be use in SI optimization phase)

✔ **GDSII**
- planar geometric shapes and other information about the layout in hierarchical form

✔ **Antenna Effect**
- In a chip manufacturing process, metal is initially deposited so it covers the entire chip
- Then, the unneeded portions of the metal are removed by etching, typically in plasma (charged particles).
- The exposed metal collect charge from plasma and form voltage potential.
- If the voltage potential across the gate oxide becomes large enough, the current can damage the gate oxide.

✔ **Antenna Ratio =** $\dfrac{\text{Area of process antennas on a node}}{\text{Area of gates to the node}}$

✔ **Problem Repair**
- Add jumper (changer metal layer)
- Add antenna cell (diode)
- Add dummy transistor

7. In APR in Lab11. Different from .db files used in synthesis. We need to provide
   (a) .lib files (Hint: this one is the same as .db files, just with different format)
   (b) .cdb files
   (c) .lef files
   (d) .captbl files and .tch files
   (e) .gds files: These files contain the physical information like metals / polys / substrates of each cell. At the end of APR, we need to merge the current layout with the .gds files of all cells to generate the .gds file for whole layout.
   Please describe what these files (a)~(d) are and the purposes of providing these files. (12%)

(a) .lib files contain timing, driving strength, capacitance loading and power information, which are used to perform static timing and power analysis.
(b) .cdb files are the noise models. They are required to consider the SI (signal integrity) issues when performing the static timing analysis. (Note that innovus will perform the post-Route timing analysis with SI consideration by default, so the files should be provided)
(c) .lef files contain the geometric information, such as metal layer sizes and pin locations.. It also provides simple DRCs like limitations of width and spacing. The files are provided to help the APR tool plan the correct routing paths, avoiding the routing wires violate DRCs or short with the metals in each cell.

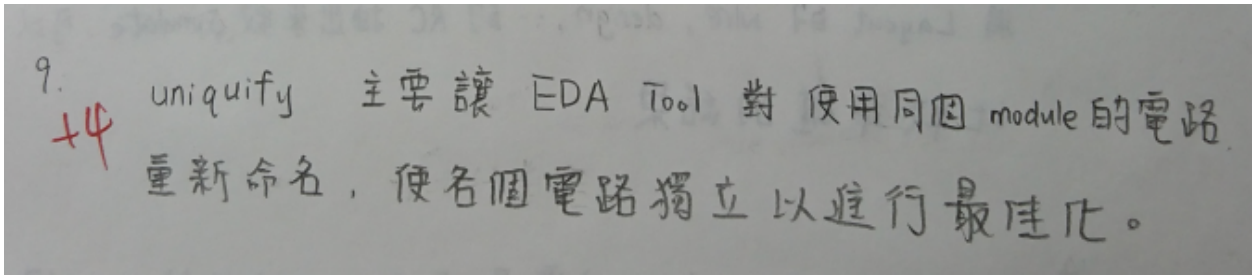
lef: process and cell information -> Metal layer information, routing pitch, default direction, Cell size, pin position, pin metal layer

(d) .captbl and .tch files both provide some rules of the RC (resistance and capacitance) extraction. .captbl files just describe the capacitance table affected by the metal shape such as width, length and thickness. Where .tch files (usually generated by the tool FireIce) provide more accurate RC extractions rule

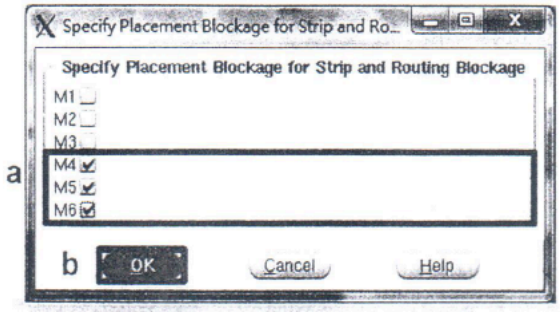
(e) .gds, a more detailed .lef for fabrication

- It is a binary file format that represents layout data in a hierarchical format.
- Data such as labels, shapes, layer information and other 2D and 3D layout geometric data.
- This file is then provided to the fabrication plant that uses this file to etch the chip based on the parameters provided in the file.

9. [4%] What is purpose of the "uniquify" command in the design flow?



11. [4%] In APR flow there is a step before we place the standard cell. **Descript its intention**.
The step: **Place -> Specify -> Placement Blockage**



- ◇ M2
- ◇ M3
- ◆ M4
- ◆ M5
- ◆ M6

Choose the metal layers of stripe. Then there will be no cell placed under stripe.

10. [4%] Lab11:

Explain what is LVS, DRC, and DRV, separately.

- ✔ **LVS: layout verse schematic**
  - The signals are connect correctly
- ✔ **DRC: design rule check**
  - Metal density
  - Size of poly, metals, vias
- ✔ **DRV: design rule violation**
  - Fan-out
  - Load
  - Driving capability
  - Wire length

11. [9%] Lab12:

Please answer the following questions:

(a) [3%] Please briefly explain what IR drop is.

(b) [3%] Please briefly explain at least 3 reasons why IR drop may cause the chip to fail.

(c) [3%] Please describe at least 3 methods to prevent IR drop.

(a) Since metals are not ideal conductors. Even though the resistance is small, when the current is large, the drop of voltage (V=IR) will not be neglectable. Especially in the middle of the chip since the current needs to pass through long paths to arrive there.

(b) With the IR drop, the power supplied to cells will not be ideal, that power may be less than VDD, and ground may be higher than 0V. The less |Vgs-Vt| make the cell run slower, and may violate the timing check (setup/hold). The allowable input range will also be smaller, implying the noise margin becomes less. Also the slower switching activity may increase the internal power.

Note: You will get full scores with mentioning two of the above issues.

(c) Just to make the equivalent R less when transmitting current to the middle of the chip, therefore adding more stripes can solve the problems.

1. Adding stripes to avoid IR drop on cell's power line • 2. Change the position of power pad to avoid long power line • 3. Use square-shaped Chip to avoid long power line • 4. Add decoupling cell 5. Uses Interleaving, wire group power rail 6. Stack as many power rail as you can to reduce the equivalent R

8. [8%] Why do we need to add pad filler and filler cell in APR flow? Please explain the reason respectively.

• Provide power to pad • There should be no spacing between pads. Pad filler is necessary for core limited design. Filler cell is used to prevent N-well DRC.

8. What's information are in the technology ~~LDF~~ LEF file and Cell library LEF file?(10%)

Technology LEF file

- ✓ **Process Technology**
  - Layers
    - Poly
    - Metal
    - Via
  - Design rule
    - Net width
    - Net spacing
    - Area
  - /*Parasitic
    - Resistance
    - Capacitance*/
- ✓ **APR Technology**
  - Site
  - Pitch
  - Default direction
  - Via rule

```
LAYER met1
  TYPE
  ROUTING ;
  WIDTH 0.240 ;
  MAXWIDTH 9.0 ;
  AREA 0.1764 ;
  THICKNESS 0.528 ;
   ....
   ....
END met1

SPACING
  SAMENET met1 met1 0.240 ;
  SAMENET met2 met2 0.280
  STACK ;
   ...
END SPACING

SITE umc6site
  SYMMETRY
  y ;  CLASS
  CORE ;
  SIZE 0.660 BY 5.040 ;
END umc6site
```

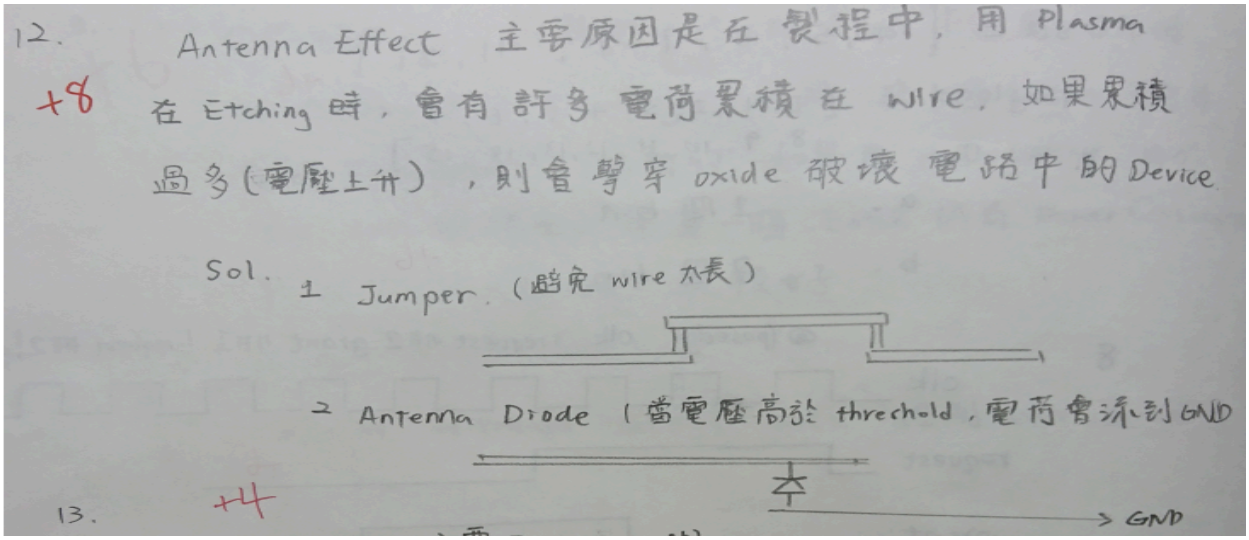
```
[VERSION statement]
[BUSBITCHARS statement]
[DIVIDERCHAR statement]
[UNITS statement]
[MANUFACTURINGGRID statement]
[USEMINSPACING statement]
[CLEARANCEMEASURE statement ;]
[PROPERTYDEFINITIONS statement]
[LAYER (Nonrouting) statement
| LAYER (Routing) statement] ...
[SPACING statement ]
[MAXVIASTACK statement]  [VIA
statement] ...
[VIARULE statement] ...
[VIARULE GENERATE statement] ...
[NONDEFAULTRULE statement] ...
[SITE statement] ...
[BEGINEXT statement]
...  [END LIBRARY]
```

Cell library LEF file

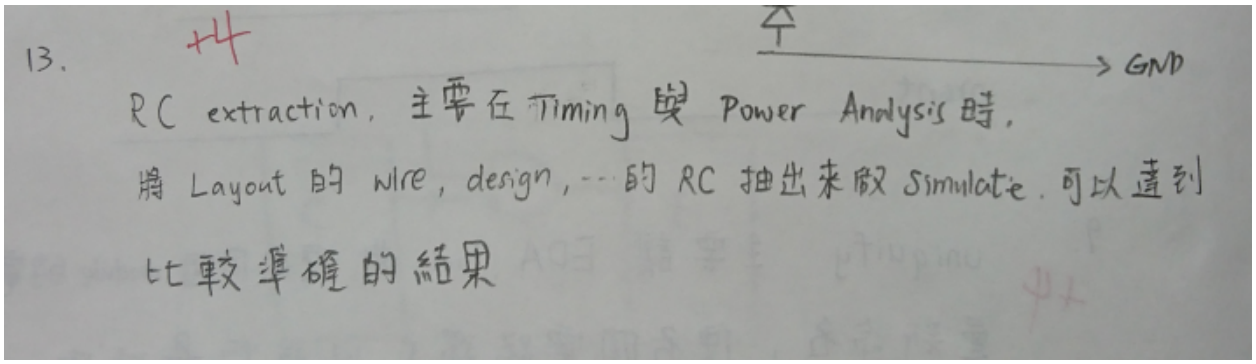
- ✓ **Define physical data for**
  - Standard cells
  - I/O pads
  - Memories
  - Other hard macros
- ✓ **Describe abstract shape**
  - A cell library LEF file can include any of the statements as shown in right hand side

```
MACRO
macroName  [
CLASS
  { COVER [BUMP]
  | RING
  | BLOCK [BLACKBOX | SOFT]
  | PAD [INPUT | OUTPUT |INOUT | POWER | SPACER | AREAIO]
  | CORE [FEEDTHRU | TIEHIGH | TIELOW | SPACER | ANTENNACELL | WELLTAP]
  | ENDCAP {PRE | POST | TOPLEFT | TOPRIGHT | BOTTOMLEFT | BOTTOMRIGHT}
  } ;]
[FOREIGN foreignCellName [pt [orient]] ;] ...
[ORIGIN pt ;]
[EEQ macroName ;]
[SIZE width BY height ;]
[SYMMETRY {X | Y | R90} ... ;]
[SITE siteName [sitePattern] ;] ...
[PIN statement] ...
[OBS statement] ...
[DENSITY statement] ...
[PROPERTY propName propVal ;] ...
END macroName
```

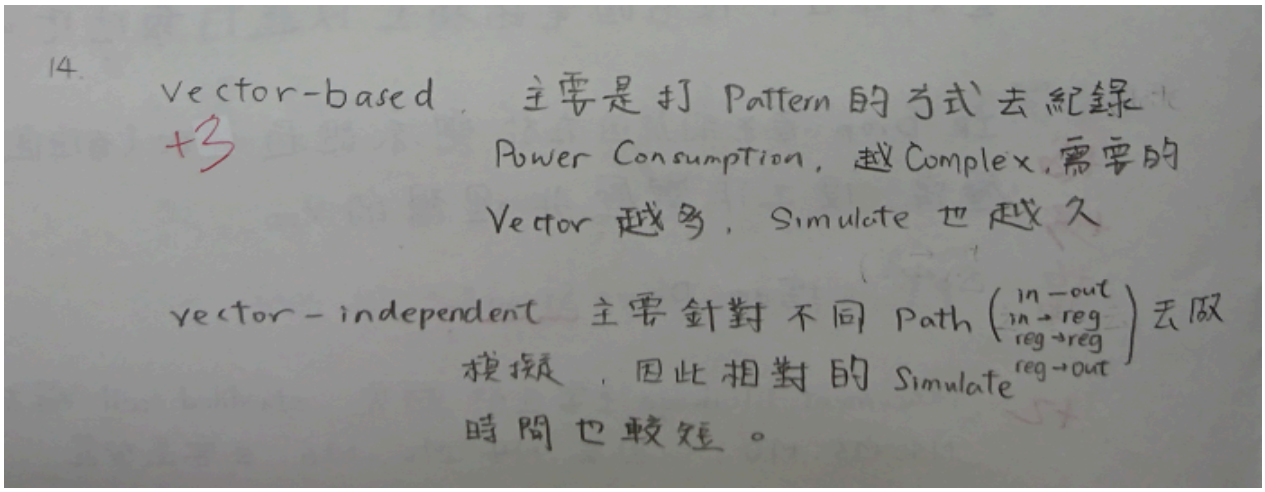
12. [8%] What is **Antenna Effect**? List 2 repair method for Antenna Effect.



13. [4%] Please simply describe why **RC extraction** is needed.



14. [6%] Describe the powered simulation methods **vector-based** and **vector-independent**.



✓ **Vector-based**
- Uses VCD (waveform file)
  - VCD is waveform file, like fsdb
- Requires:
  - Simulation is possible at the full-chip level
  - Simulation provides sufficient functional coverage of design
  - Vectors include those cause highest power consumption

✓ **Vector-independent**
- Uses TWF (timing window file, generated by Static Timing Analysis)
  - Clock domain, skews, slack, arrival times of each pin.
- Requires:
  - A good estimation of input switching frequency

9. [8%] Please explain what is crosstalk, and list at least three solutions.

兩條平行導線之間會有耦合效應(寄生電容)導致彼此電壓改變時會互相影響

✔ **Crosstalk prevention – Placement solution**
- Insert buffer in lines
- Upsize driver
- Congestion optimization

✔ **Crosstalk prevention – Routing solution**
- Limit length of parallel nets
- Wider routing grid
- Shield special nets

11. [4%] (a) IR drop can cause the chip to fail due to (at least 3)?
[4%] (b) What can we do to solve IR drop?

- IR drop can cause the chip to fail due to
  - ◆ Performance (circuit running slower than specification)
  - ◆ Functionality problem (setup or hold violations)
  - ◆ Unreliable operation (less noise margin)

- Prevention:
  - 1. Adding stripes to avoid IR drop on cell's power line
  - 2. Change the position of power pad to avoid long power line
  - 3. Use square-shaped Chip to avoid long power line
  - 4. Add decoupling cell

8. Briefly describe what pad limited design and core limited design are in APR. (5%)

✔ **Core limited design or Pad limited design**
- Die size determination
  - When pad width > (core width + core margin), die size is decided by pads. And it is called pad limited design
  - When pad width < (core width + core margin), die size is decided by core. And it is called core limited design

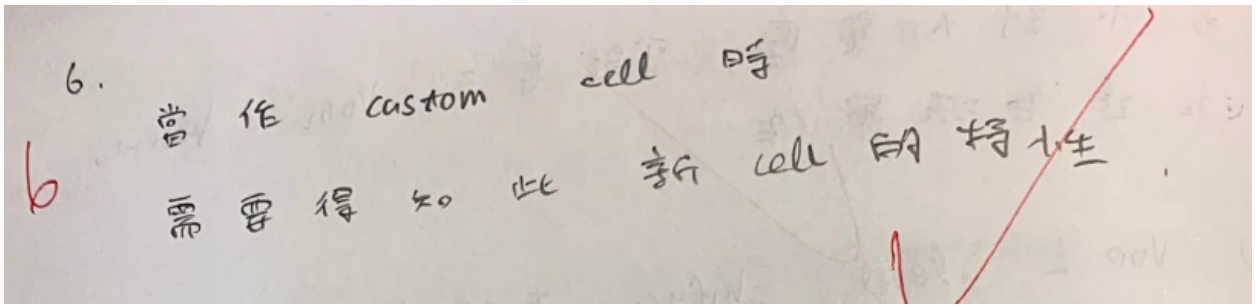
Pad Width

12. [7%] Lab12:

(a) [3%] Please describe what is electromigration (EM).
(b) [4%] Please describe the difference between power analysis and rail analysis.

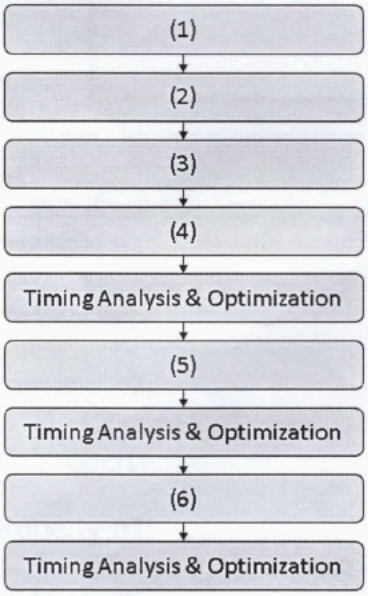
(a) 電路使用很久後，導線會有侵蝕和堆積的現象，導致電路出現斷路或短路的情形
(b) power analysis: 電路大概消耗多少能量；rail analysis: 電路的power delay是否足夠供應到電路中的每個cell

6. [6%] Why we need library characterization? (one reason)

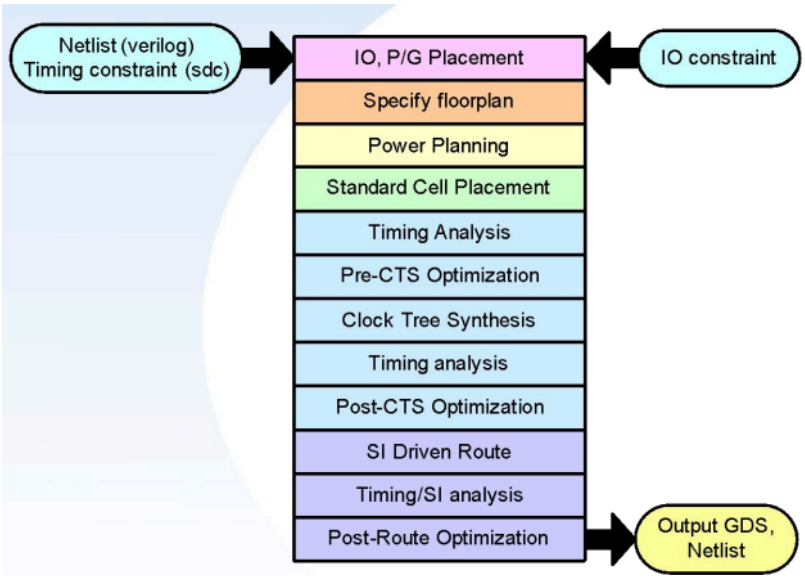


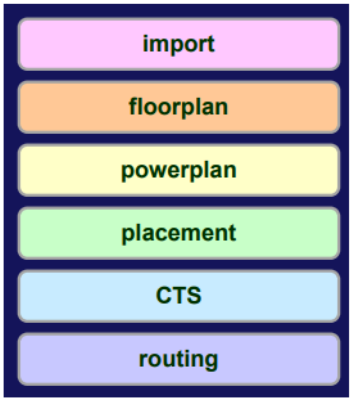
1.Please finish the APR flow as below. (12%)
(Example:(1)a (2)b (3)c (4)d (5)e (6)f)

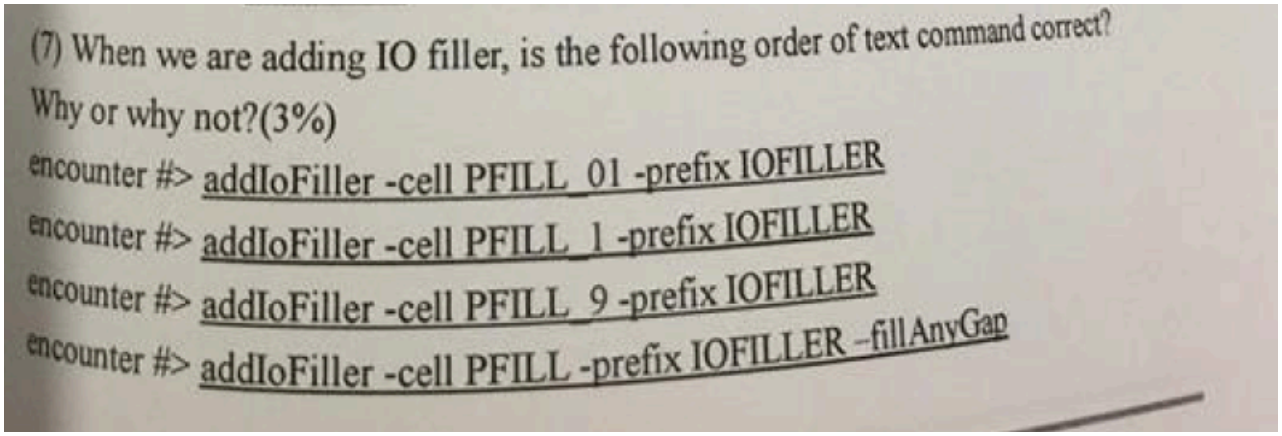
a.Standard Cell Placement
b.Specify floorplan
c.Clock Tree Synthesis
d.Power Planning
e.SI Driven Route
f.IO, P/G Placement

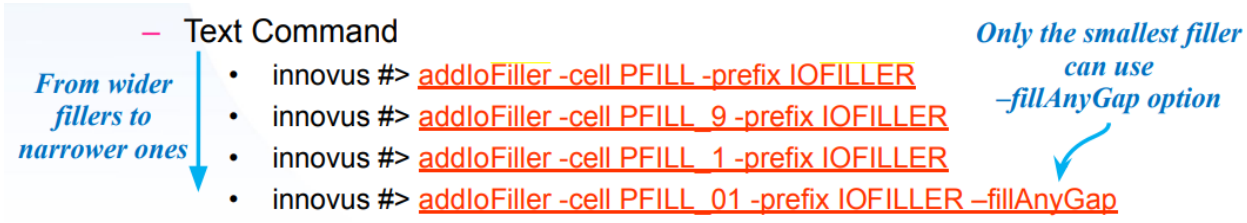


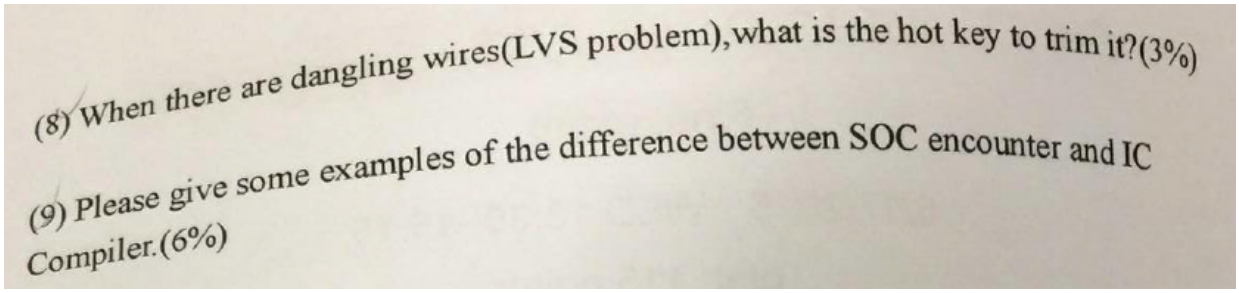
(1)f (2)b (3)d (4)a (5)c (6)e







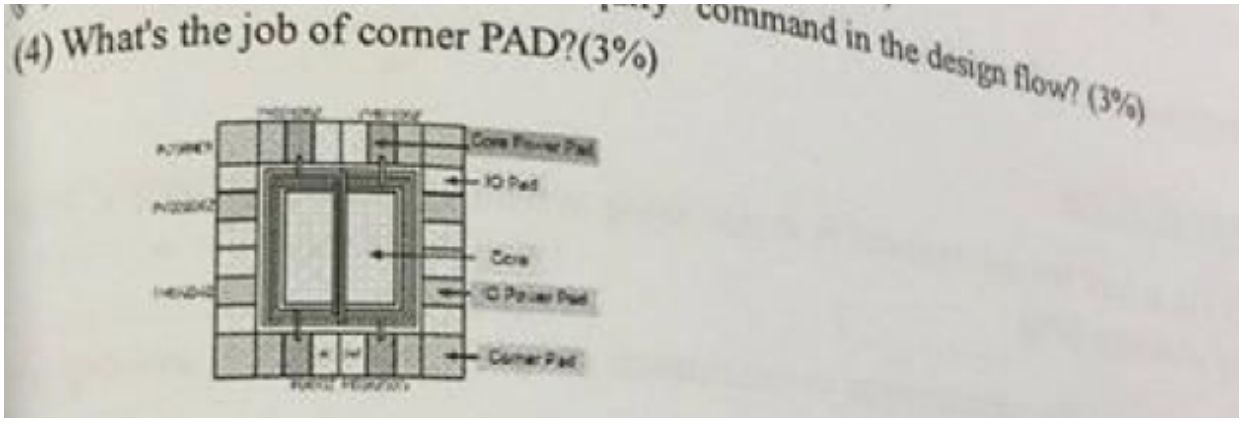




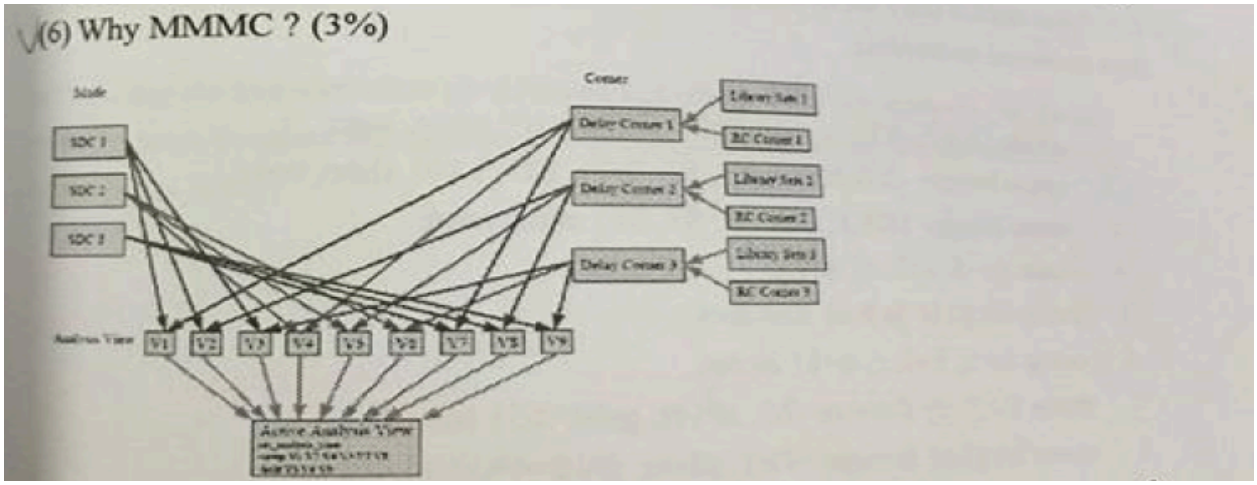
(8) shift-T

(9)

IC compiler is a place & route tool from Synopsis, which works in conjunction with design compiler (DC for synthesis).
SOC Encounter is a platform from Cadence for RTL to GSDII flow which uses RTL compiler (RC for synthesis).
Ultimately both are doing same thing by different EDA vendors.



Filling up the gap to prevent DRC violation.

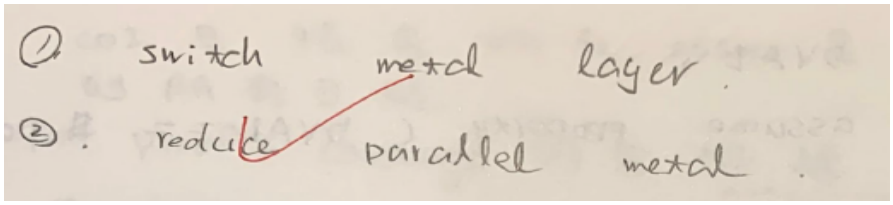


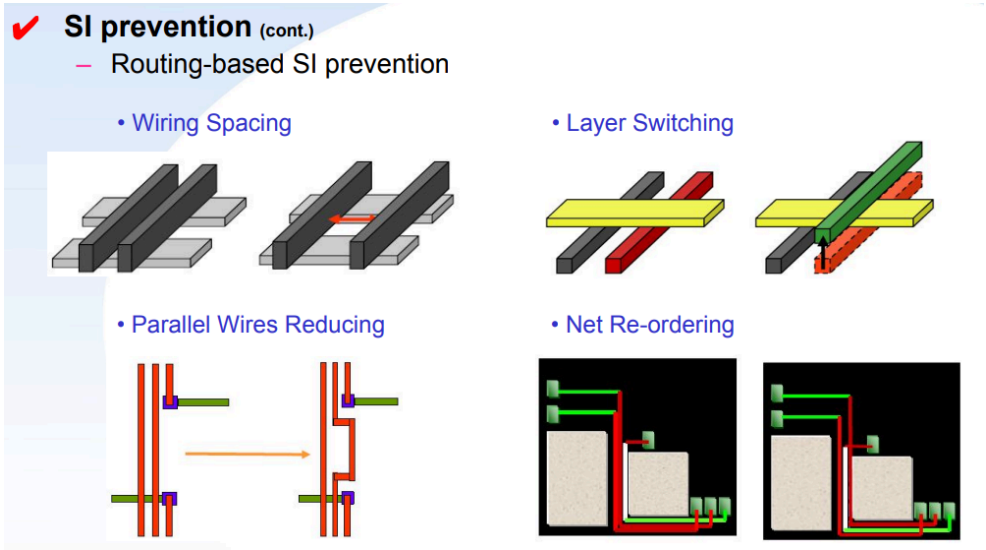
multi-mode-multi-corner: to analyze the design under different corner cases, ensure the design can work under different conditions

8. [6%] List three of Signal Integrity (SI) Issue

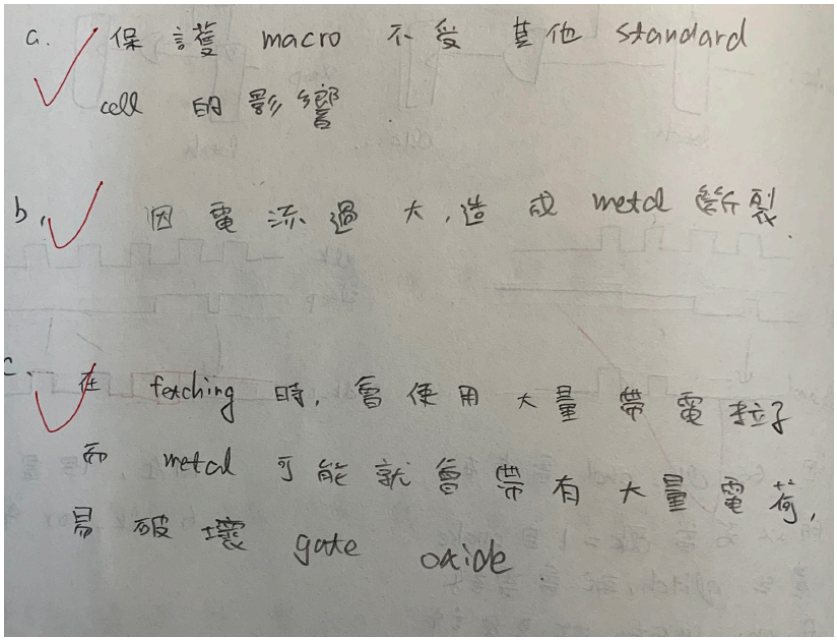
✓ **Signal Integrity (SI) Issue**
- Crosstalk
- Charge sharing
- Supply noise
- Leakage
- Propagated noise
- Overshoot
- Under shoot

9. [4%] List two routing method to prevent Signal Integrity (SI) Issue

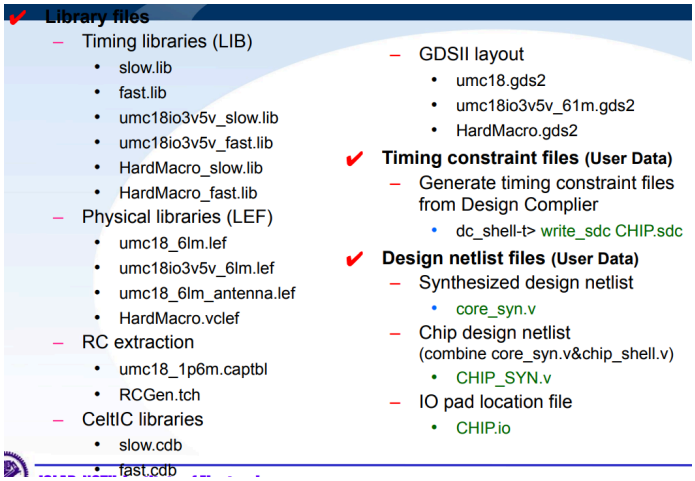




7. [10%] Please explain the following:
   a. [4%] What is the usage of Halo in floorplan
   b. [3%] What is Metal migration
   c. [3%] What is Antenna effect



9. [10%]:
   What libraries and files are used in APR. Please explain the purpose of including these files.



- ✓ **LIB Library : timing information**
  - Calculate cell delay according to different combinations of input slew rate and output loading capacitance
  - Using SignalStorm Library Characterizer
- ✓ **LEF Library : process and cell information**
  - Metal layer information, routing pitch, default direction
  - Cell size, pin position, pin metal layer
- ✓ **RC Extraction**
  - RC extraction for further power analysis, simulation etc
- ✓ **CeltIC Library**
  - cdB model (noise model for each cell which will be use in SI optimization phase)
- ✓ **GDSII**
  - planar geometric shapes and other information about the layout in hierarchical form

Please describe the benefits of using **wire group** and **interleaving** on power ring.

wire group: via數量增加 減少電阻

interleaving: 穩壓, prevent the noisy voltage source from causing signal integrity issue.

12. [12%]:
    a. [4%] What is the difference between power/rail analysis?
    b. [4%] What input files do we need when running vector-based power analysis and rail analysis respectively? What's the content in the files?
    c. [4%] If we use different input files to do the vector-based power analysis, will the result of rail analysis be different? Why?

a. The power analysis is to analyze the power consumption of your design which include dynamic power and static power. While rail analysis is to analyze if the powerplan is enough to provide whole chip.

b. Vector-based power analysis:
   - VCD : waveform file, like .fsdb file.

   Rail analysis:
   - dynamic_VDD.pitavg : instance current file.
   - dynamic_GND.pitavg : instance current file.

c. Yes. Because the vector-based power analysis use VCD file to generate instance current file, if we choose different interval of VCD, that might be different path that electron go through, which means there may be different instantaneous

current generate. So the instance current file will be different of original one and cause different analysis result.

Else (未分類)

```
3. (10%)
interface INF(input blk)
        logic data,ready,done,addr;
        clocking cb @(posedge clk)
                default input #2ns output #1ns
                input data , ready;
                output done;
                output #3ns addr;
        endclocking

        modport cb(clocking cb);
endinterface
```