NCTU-EE IC LAB – Fall 2018

Final Exam Reference Answer

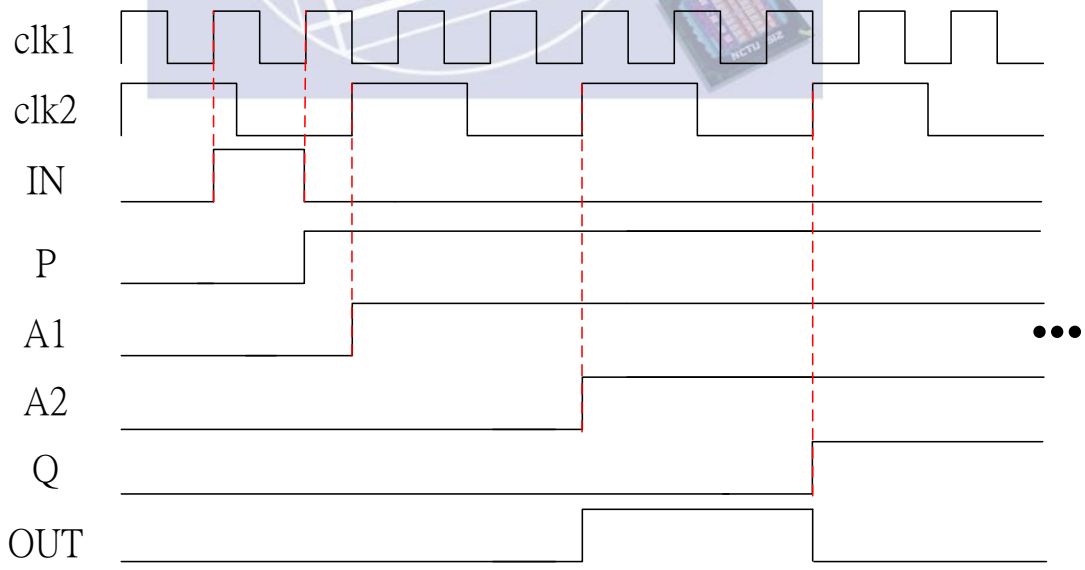
1.

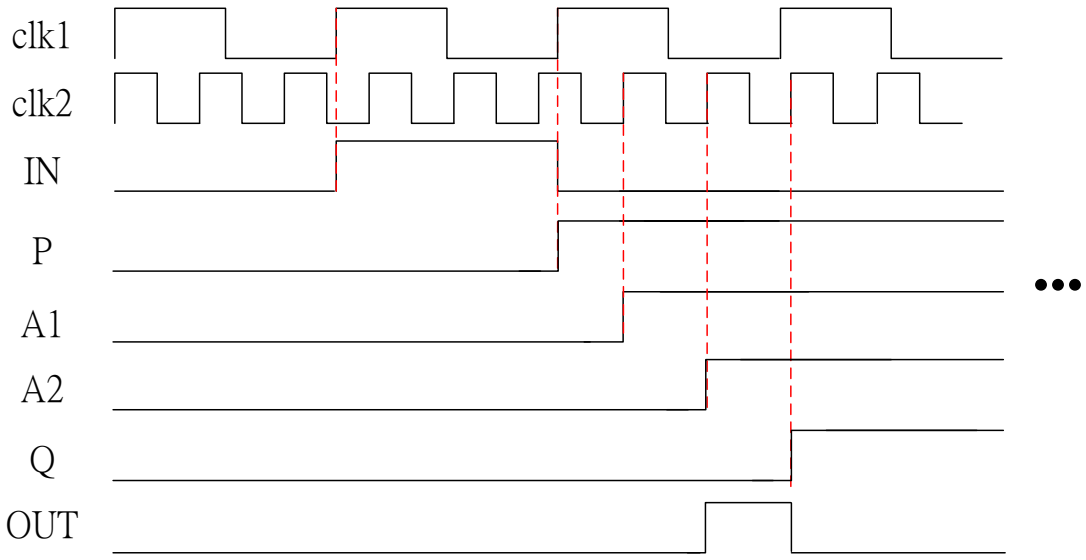
When the frequency of clk1 and clk2 are different, the clock skew between clocks varies with time. The skew leads to a problem that sometimes setup / hold check will be violated, and sometimes not. This will cause the receiving flip-flop in clk2 clock domain (In Fig.1, it is F1) enter metastable state, 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).

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.

(b)

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.

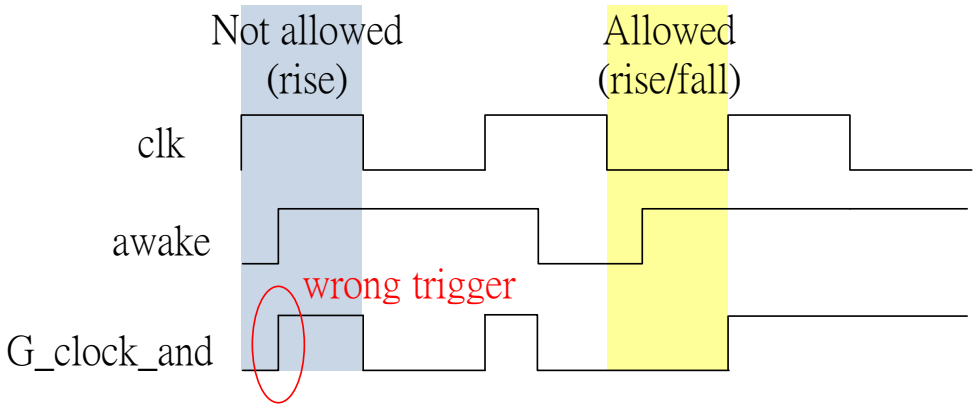
3.

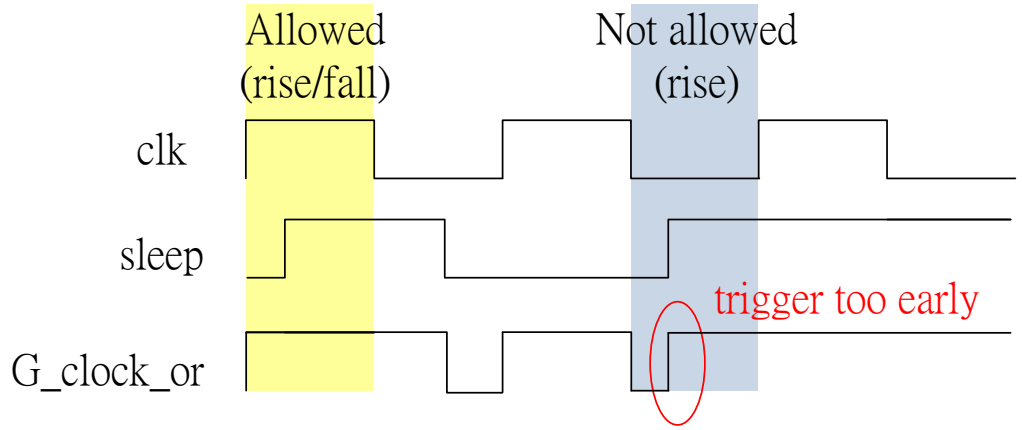
Create a counter and make it increase when the inputs are coming. Store the input to the certain register according to the counter value. Then all flip-flops can be clock gated when the counter is corresponded to other storing flip-flops.

```
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
```

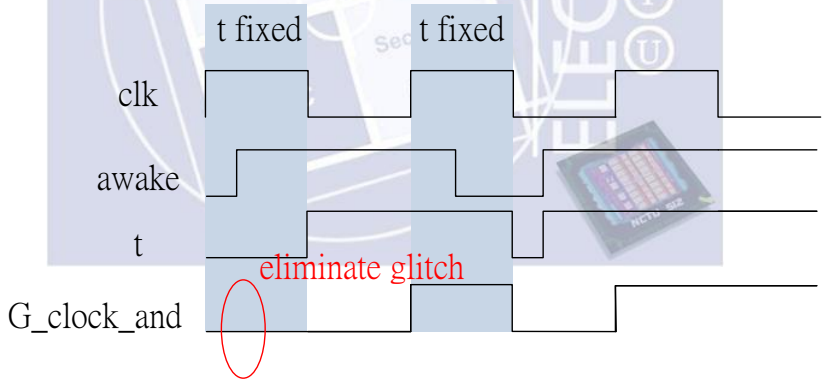
4. (a)

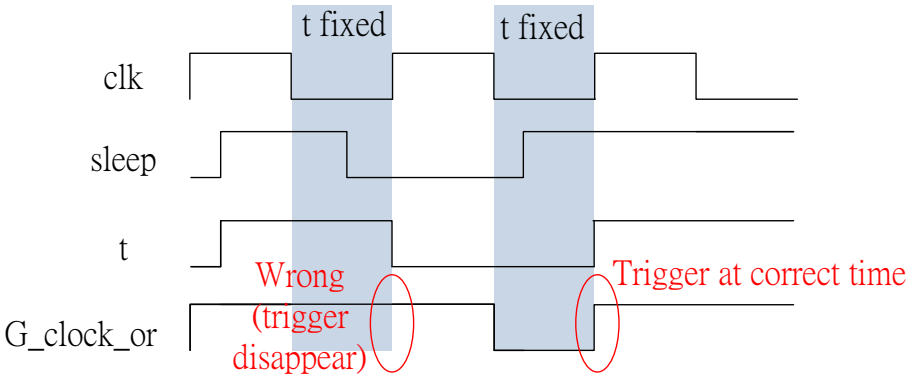




As shown in the above figures: for AND-based clock gating cells, the awake signal cannot rise when clk = 1. Since AND-base clock gating cells try to tie the output to 0, the awake signal passes when clk = 1 and when awake is rising, the flip-flop will be triggered. (Falling is actually OK. But when signal 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.

5.

(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.

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.

7.

(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.

(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

8.

When the area before layout (which may contain standard cells area and macros area) is small but the number of pads is large. The chip size will be determined by the pad numbers and is called pad limited design. On the other hand, if the area before layout is large and the number of pads is small, the chip size will be determined by the core area (area before layout / utilization) and the core to io boundary, which is called core limited design.

Note 1: Aspect ratio of macros, utilization and core to io boundary also determine whether it is pad limited or core limited design, but here they are assumed to be in reasonable ranges. So the major causes are the area before layout and the number of pads

Note 2: You have to mention the pad number issues. If you write an unclear description like pad area > core area so that it is pad limited design, you will get some deducted points. Drawing pictures to help description will be also OK.

9.

(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.