|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Nguyen Le Thao Vy | No. | 3341 | Div/Dept | DSD/ACD/ACT3 | Job  Date：Jan 09th 2024  Title | Engineer |
| Please tick  the period | First Month | □W1 □W2 □W3 □W4 | | | | | |
| Second Month | □W1 □W2 🗹W3 □W4 | | | | | |
| Third Month | □W1 □W2 □W3 □W4 | | | | | |

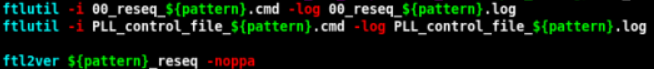
1. The weekly report aims to help accelerate member’s workspace integration and should be reviewed by mentor on the last working day of the week.
2. The new weekly report should be reviewed and signed by mentor and direct supervisor.

|  |
| --- |
| Work Experience Record |
| 1. Please describe the tasks and achievements you learned/executed:   All of the tasks I learned during this first week are based on the training plan of 2023, below are a brief description of topics and tasks I was working on:   * Topics of interest:   + Clock;   + SDC constraints;   + Testchip workflow. * Tasks that I was working on:   + Read materials – self-learn;   + Work on testchip FSFGFS016A. * Outcomes:   + Learnt basic concepts that aid in working process;   + Got familiar with running design kit and working in project. |
| 1. What are the problems encountered this week? Any actions taken? Any help needed?  * Problem encountered:   + Being unfamiliar with the testchip workflow. * Actions taken:   + Went through the flow in [FTV\_ACD\_TestChip\_Flow\_202303](file:///T:\FTV\DSD\DSD_ACD\ACT4\02_Internal_Tech_Shared\03_Testchip\FTV_ACD_TestChip_Flow_202303) under mentor’s guidance. |
| 1. What are the tasks for next week? Any preparation needed in advance?  * Work on assigned project; * Make time for completing items listed in training plan. |
|  |

|  |  |  |
| --- | --- | --- |
| Name  (Date) | Mentor | Direct Supervisor |
| Nguyen Le Thao Vy  (2024/01/21) | (Signature/Date) | (Signature/Date) |

***Questions from previous session***

1. Why do we need to resequence pattern?

Patterns are resequenced to avoid mismatch between Faraday Test Language and the format used by Xcelium.

1. What is add\_seq\_delay?

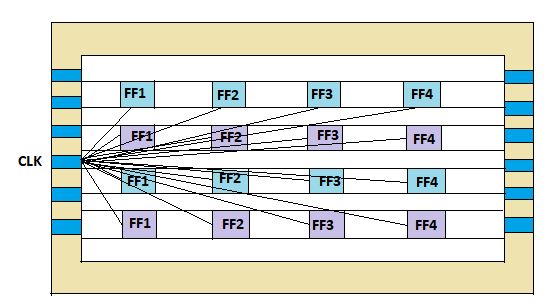
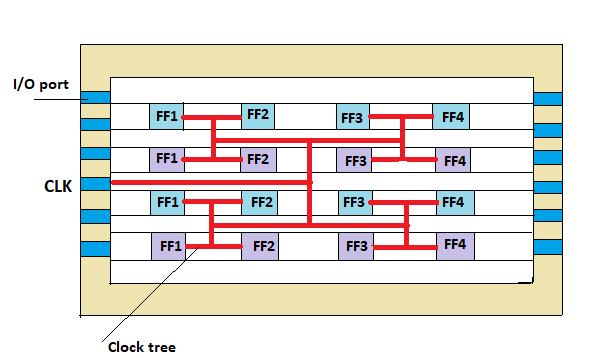
Simulation with no timing delays, i.e no SDF delay annotation, is prone to race conditions. With no proper clock delay balancing, the clock and data signals can propagate instantaneously, making the data from one stage available at downstream stages before the appropriate clock edge is available. Therefore, add\_seq\_delay should be used to apply a delay\_value to the input/output path of all sequential FFs that do not already have a path delay provided in the instantiation or to apply a delay\_value to an explicit instance.

1. Which type of timing violation is more critical?

Hold violation is more critical, because it is independent from the clock frequency (more difficult to fix), whereas setup violation can be fixed by adjusting clock signal.

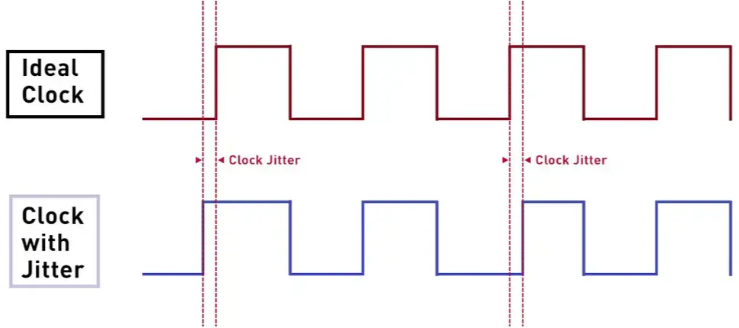
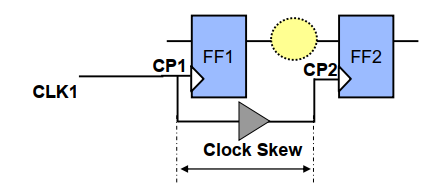
***Clock***

Clocks are periodic signals that are broadcast and therefore are generated either on the chip or outside of the chip. Common features of a clock include clock period, which defines at what interval the waveform repeats itself, its rising and falling edges, and pulse width that is determined based on how much the clock is high.

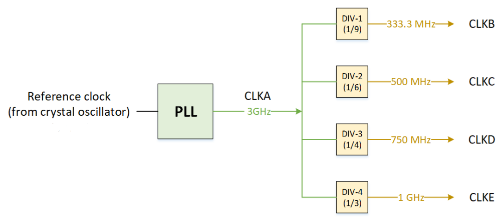
**Clock propagation:** Clock signals need to traverse across the entire design to reach all the flip-flops and latches. To make sure that all clock signals are distributed uniformly to all sequential elements, buffers and/or inverters can be added along the clock paths to balance skew and minimize insertion delay.

* Ideal mode: At stages prior to CTS, there is no clock buffer in the design, so the effect of a clock must be explicitly modeled in SDC file(s).
* Propagated mode: After CTS, the modeled clock information can now be replaced with actual clock information.

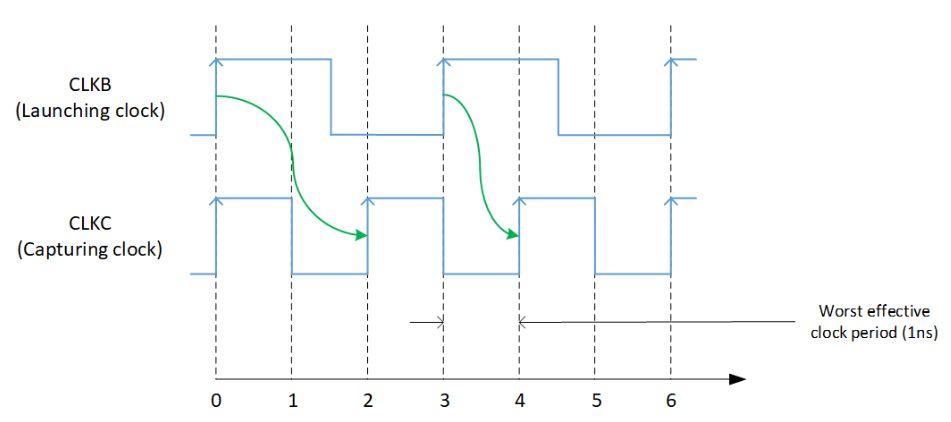
**Clock uncertainty** = Clock jitter + clock skew, in which:

* Clock jitter = deviation of a clock edge from its ideal location; refers to uncertainty in period and duty cycle of the clock that typically originate from the circuitry that is generating the clock signals.
* Clock skew: Differences in clock signal arrival times across the chip are called clock skew. Ideally, CTS is performed to reduce variance of the clock arrival at the different leaf flops, but it is inevitable because it is a real design where data transfers between flops happen across various flops in the design.

**Multiple synchronous clocks scenario**

Below is an example of a scenario where we have multiple clocks: PLL is generating a main clock named CLKA of frequency 3 GHz, and there are 4 dividers generating CLKB, CLKC, CLKD and CLKE of frequency 333.3 MHz, 500 MHz, 750 MHz and 1 GHz respectively, from the main clock. Since all these clocks are derived from the same clock source, they are all synchronous to each other.

Consider a (setup) timing path that involves two clocks CLKB (launch clock) and CLKC (capture clock). To determine the worst case data transfer in this case, there are certain steps to follow:

1. Find the base period: Base period is defined as the least common multiple of the clock periods involved. In this case, the least common multiple is 6ns. Basically, this means that the clocks waveform between 0-6ns will look the same as between 6-12ns, 12-18ns, so on; therefore, the tool only needs to find the worst case timing situation between 0-6ns and that will define the worst case timing for all the clock cycles.
2. Align the leading edges of waveform and determine the worst case data transfer.
   * First launching edge: 0ns; first capturing edge: 2ns 🡪 effective clock period = 2ns.
   * Second launching edge: 3ns; second capturing edge: 4ns 🡪 effective clock period = 1ns.

As shown above, the worst case happens at the second clock cycle. In order to present this worst case in the first cycle, the capture clock edge can be shifted a certain amount, called “*phase shift*”. The equation used for phase shift calculation is:

(CE – LE) – (C1 – L1), in which:

* + CE = worst-case capturing clock edge
  + LE = worst-case launching clock edge
  + C1 = capturing edge in first cycle
  + L1 = launching edge in first cycle

Example: (4 – 3) – (2 – 0) = 1 – 2 = -1 🡪 if CLKC is shifted left 1 unit, the worst case data transfer will show up in the first cycle.

***SDC***

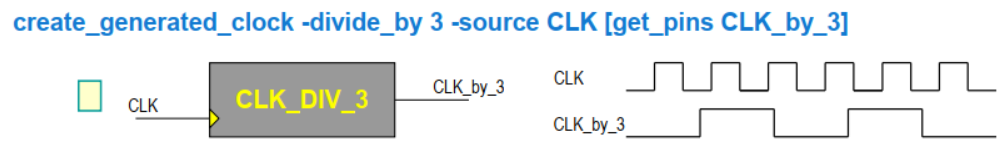
Below are a summary of design objects that exist in a design:

* Design (current\_design) = container for the entire circuit;
* Cell / Block (get\_cells) = can be an instance of a design / library component;
* Port (get\_ports / all\_inputs / all\_outputs) = signal entry / exit point;
* Pin (get\_pins) = a physical connection for a single net;
* Clock (get\_clocks / all\_clocks) = port / pin that drives sequential cells;
* Net (get\_nets) = interconnect between cell pins and design ports.

**Some commonly used commands**

* Timing
  + **create\_clock** period\_value [-name clock\_name]

[-waveform edge\_list] [-add] [source\_objects]

* + **create\_generated\_clock** [-name clock\_name]-source master\_pin [-edges edge\_list][-divide\_by factor] [-multiply\_by factor][-duty\_cycle percent] [-invert][-edge\_shift shift\_list] [-add] [-master\_clock clock][-combinational]
  + **set\_clock\_latency** [-rise] [-fall] [-min] [-max][-source] [-late] [-early] [-clock clock\_list] object\_list
  + **set\_clock\_transition** [-rise] [-fall] [-min] [-max]

transition clock\_list

* + **set\_disable\_timing** [-from from\_pin\_name] [-to to\_pin\_name] cell\_pin\_list
  + **set\_propagated\_clock** object\_list
  + **set\_clock\_uncertainty** [-from from\_clock][-rise\_from rise\_from\_clock][-fall\_from fall\_from\_clock] [-to to\_clock][-rise\_to rise\_to\_clock] [-fall\_to fall\_to\_clock][-rise] [-fall] [-setup] [-hold] object\_list
  + **set\_input\_delay** [-clock clock\_name] [-clock\_fall][-rise] [-fall] [-max] [-min] [-add\_delay][-network\_latency\_included] [-source\_latency\_included]
  + **set\_output\_delay** [-clock clock\_name] [-clock\_fall][-level\_sensitive] [-rise] [-fall] [-max] [-min] [-add\_delay][-network\_delay\_included] [-source\_latency\_included]
* Exception
  + **set\_false\_path** [-setup] [-hold] [-rise] [-fall][-from from\_list] [-to to\_list] [-through through\_list][-rise\_from rise\_from\_list] [-rise\_to rise\_to\_list][-rise\_through -rise\_through\_list][-fall\_from fall\_from\_list] [-fall\_to fall\_to\_list][-fall\_through fall\_through\_list]
  + **set\_max\_delay** [-rise] [-fall][-from from\_list] [-to to\_list] [-through through\_list][-rise\_from rise\_from\_list] [-rise\_to rise\_to\_list][-rise\_through rise\_through\_list][-fall\_from fall\_from\_list] [-fall\_to fall\_to\_list][-fall\_through fall\_through\_list]
  + **set\_multicycle\_path** [-setup] [-hold] [-rise] [-fall][-start] [-end] [-from from\_list] [-to to\_list][-through through\_list] [-rise\_from rise\_from\_list][-rise\_to rise\_to\_list][-rise\_through rise\_through\_list][-fall\_from fall\_from\_list] [-fall\_to fall\_to\_list]

[-fall\_through fall\_through\_list] path\_multiplier

* Environmental
  + **set\_load** [-min] [-max] [-subtract\_pin\_load] [-pin\_load][-wire\_load] value objects
  + **set\_fanout\_load** value port\_list
  + **set\_input\_transition** [-rise] [-fall] [-min] [-max][-clock clock\_name] [-clock\_fall] port\_list
  + **set\_port\_fanout\_number** value port\_list
* Design rules
  + **set\_max\_capacitance** value object\_list
  + **set\_max\_fanout** value object\_list
  + **set\_max\_transition** [-clock\_path][-data\_path] [-rise] [-fall] value object\_list