Basic of Timing Analysis in Physical Design

Lots of people asked me to write over timing analysis. Though a lot of material is present but still most of the people are not 100% sure about all these concepts. I am trying to put few of the things here in a simpler language and hope that it will help (beginner and professional).

Please let me know if anybody thinks that I should add few more things here. It's difficult to put everything in one blog so just consider this as the first part of Timing analysis.

<u>What is Timing Analysis??</u>

Before we start anything at least we should know what exactly we mean by Timing Analysis. Why these days it's so important?

<u>There are a couple of reasons for performing timing analysis</u>.

- We want to verify whether our circuit meet all of its timing requirements (Timing Constraints)
  - There are 3 types of design constraints- timing, power, area. During designing there is a trade-offs between speed, area, power, and runtime according to the constraints set by the designer. However, a chip must meet the timing constraints in order to operate at the intended clock rate, so timing is the most important design constraint.
- We want to make sure that circuit is properly designed and can work properly for all combinations of components over the entire specified operating environment. **"Every Time".**
- Timing analysis can also help with component selection.
  - An example is when you are trying to determine what memory device speed, you should use with a microprocessor. Using a memory device that is too slow may not work in the circuit (or would degrade performance by introducing wait states), and using one that is too fast will likely cost more than it needs to.

So I can say Timing analysis is the methodical analysis of a digital circuit to determine if the timing constraints imposed by components or interfaces are met. Typically, this means that you are trying to prove that all set-up, hold, and pulse-width times are being met.

<u>Note:</u> Timing analysis is integral part of ASIC/VLSI design flow. Anything else can be compromised but not timing!

<u>Types of Timing Analysis:</u>

There are 2 type of Timing Analysis-

- Static Timing Analysis:
  - Checks static delay requirements of the circuit without any input or output vectors.
- Dynamic Timing Analysis.
  - verifies functionality of the design by applying input vectors and checking for correct output vectors

Basic Of Timing Analysis:

The basis of all timing analysis is the clock and the sequential component (here we will discuss with the help of Flip-flop) . Following are few of the things related to clock and flip-flop which we usually wants to take care during Timing analysis.

<u>Clock related:</u>

- It must be well understood parametrically and glitch-free.
- Timing analysis must ensure that any clocks that are generated by the logic are clean, are of bounded period and duty cycle, and of a known phase relationship to other clock signals of interest.
- The clock must, for both high and low phases, meet the minimum pulse width requirements.
- Certain circuits, such as PLLs, may have other requirements such as maximum jitter. As the clock speeds increase, jitter becomes an increasingly important parameter.
- When "passing" data from one clock edge to the other, ensure that the worst-case duty cycle is used for the calculation. A frequent source of error is the analyst assuming that every clock will have a 50% duty cycle.

Flip-Flop related:

- All of the flip-flops parameters are always met. The only exception to this is when synchronizers are used to synchronize asynchronous signals
- For asynchronous presets and clears, there are two basic parameters that must be met.
- All setup and hold times are met for the earliest/latest arrival times for the clock.
- Setup times are generally calculated by designers and suitable margins can be demonstrated under test. Hold times, however, are frequently not calculated by designers.
- When passing data from one clock domain to another, ensure that there is either known phase relationships which will guarantee meeting setup and hold times or that the circuits are properly synchronized

Now Lets talk about Each type of Timing analysis One by one in the next few blogs.

As we have discussed in our last blog (about Basic of Timing analysis) that there are 2 types of timing analysis.

- Static Timing Analysis
- Dynamic Timing Analysis.

Note: There is one more type of Timing analysis: "Manual Analysis". But now a days nothing is 100% Manual. Evey thing is more automated and less manual. So that we are not discussing right now.

In this Blog (and few next as a part of this) we will discuss about the Static Timing Analysis. We will discuss Dynamic Timing Analysis later on.

Static Timing analysis is divided into several parts:

- Part 6c -> Continue of How to solve Setup and Hold Violation (more advance examples)

Static Timing Analysis:

Static timing analysis is a method of validating the timing performance of a design by checking all possible paths for timing violations under worst-case conditions. *It considers the worst possible delay through each logic element, but not the logical operation of the circuit*.

In comparison to circuit simulation, static timing analysis is

- Faster - It is faster because it does not need to simulate multiple test vectors.
- More Thorough - It is more thorough because it checks the worst-case timing for all possible logic conditions, not just those sensitized by a particular set of test vectors.

Once again Note this thing : Static timing analysis checks the design only for proper timing, not for correct logical functionality.

Static timing analysis seeks to answer the question, "Will the correct data be present at the data input of each synchronous device when the clock edge arrives, under all possible conditions?"

In static timing analysis, the word static alludes to the fact that this timing analysis is carried out in an input-independent manner. It locates the worst-case delay of the circuit over all possible input combinations. There are huge numbers of logic paths inside a chip of complex design. The advantage of STA is that it performs timing analysis on all possible paths (whether they are real or potential false paths).
However, it is worth noting that STA is not suitable for all design styles. It has proven efficient only for fully synchronous designs. Since the majority of chip design is synchronous, it has become a mainstay of chip design over the last few decades.

The Way STA is performed on a given Circuit:
To check a design for violations or say to perform STA there are 3 main steps:

- Design is broken down into sets of timing paths,
- Calculates the signal propagation delay along each path
- And checks for violations of timing constraints inside the design and at the input/output interface.

The STA tool analyzes ALL paths from each and every startpoint to each and every endpoint and compares it against the constraint that (should) exist for that path. All paths should be constrained, most paths are constrained by the definition of the period of the clock, and the timing characteristics of the primary inputs and outputs of the circuit.

Before we start all this we should know few key concepts in STA method: timing path, arrive time, required time, slack and critical path.
Let's Talk about these one by one in detail. In this Blog we will mainly Focus over Different Types of Timing Paths.

Timing Paths:

Timing paths can be divided as per the type of signals (e.g clock signal, data signal etc).

Types of Paths for Timing analysis:

- Data Path
- Clock Path
- Clock Gating Path
- Asynchronous Path

Each Timing path has a "Start Point" and an "End Point". Definition of Start Point and End Point vary as per the type of the timing path. E.g for the Data path- The startpoint is a place in the design where data is launched by a clock edge. The data is propagated through combinational logic in the path and then captured at the endpoint by another clock edge.

Start Point and End Point are different for each type of paths. It's very important to understand this clearly to understand and analysing the Timing analysis report and fixing the timing violation.

- Data path
  - Start Point
    - Input port of the design (because the input data can be launched from some external source).
    - Clock pin of the flip-flop/latch/memory (sequential cell)
  - End Point
    - Data input pin of the flip-flop/latch/memory (sequential cell)
    - Output port of the design (because the output data can be captured by some external sink)
- Clock Path
  - Start Point
    - Clock input port
  - End Point
    - Clock pin of the flip-flop/latch/memory (sequential cell)
- Clock Gating Path
  - Start Point
    - Input port of the design
  - End Point
    - Input port of clock-gating element.
- Asynchronous path
  - Start Point
    - Input Port of the design
  - End Point
    - Set/Reset/Clear pin of the flip-flop/latch/memory (sequential cell)

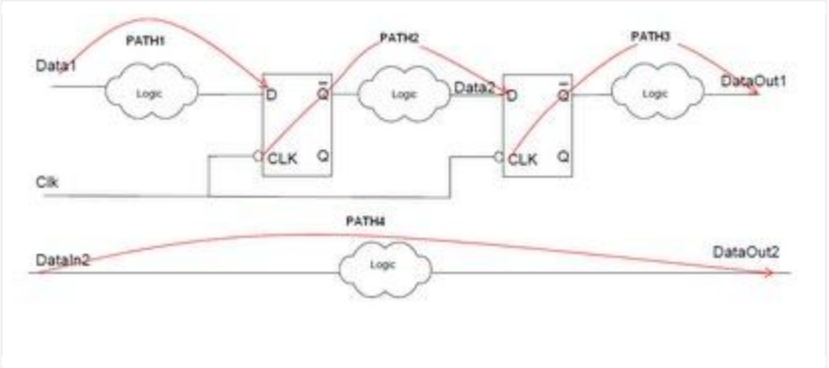
Data Paths:

If we use all the combination of 2 types of Starting Point and 2 types of End Point, we can say that there are 4 types of Timing Paths on the basis of Start and End point.

- Input pin/port to Register(flip-flop).
- Input pin/port to Output pin/port.
- Register (flip-flop) to Register (flip-flop)

- Register (flip-flop) to Output pin/port

Please see the following fig:

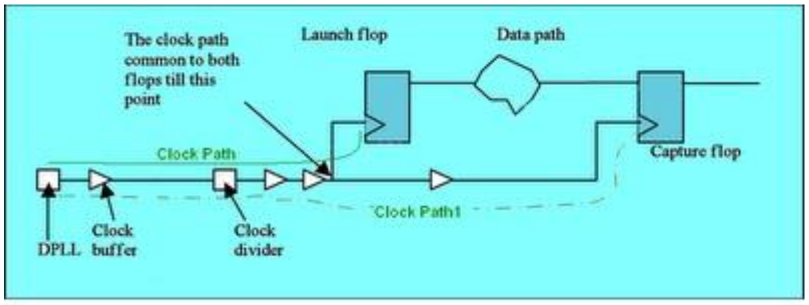


Timing Path- 4 types of Data Path

PATH1- starts at an input port and ends at the data input of a sequential element. (Input port to Register)
PATH2- starts at the clock pin of a sequential element and ends at the data input of a sequential element. (Register to Register)
PATH3- starts at the clock pin of a sequential element and ends at an output port.(Register to Output port).
PATH4- starts at an input port and ends at an output port. (Input port to Output port)

<u>Clock Path:</u>

Please check the following figure



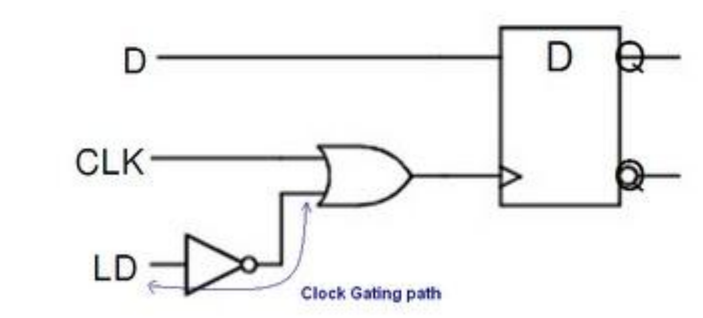
Timing Paths- Clock Paths

In the above fig its very clear that for clock path the starts from the input port/pin of the design which is specific for the Clock input and the end point is the clock pin of a sequential element. In between the Start point and the end point there may be lots of Buffers/Inverters/clock divider.

<u>Clock Gating Path:</u>

Clock path may be passed trough a “gated element” to achieve additional advantages. In this case, characteristics and definitions of the clock change accordingly. We call this type of clock path as “gated clock path".

As in the following fig you can see that



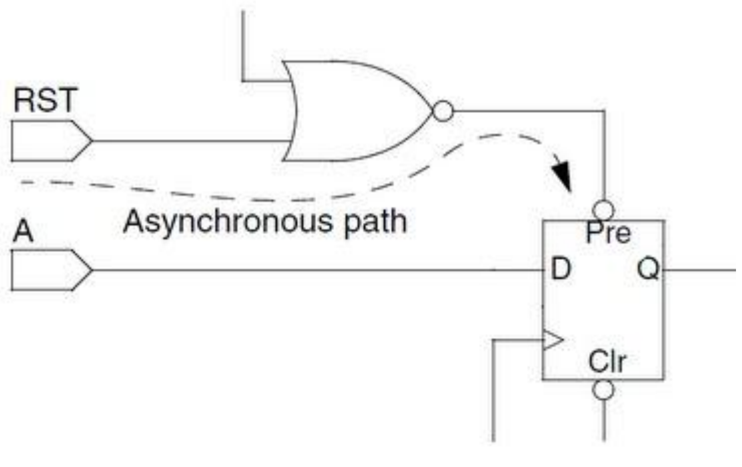
Timing Path- Clock Gating path.

LD pin is not a part of any clock but it is using for gating the original CLK signal. Such type of paths are neither a part of Clock path nor of Data Path because as per the Start Point and End Point definition of these paths, its different. So such type of paths are part of Clock gating path.

Asynchronous path:

A path from an input port to an asynchronous set or clear pin of a sequential element.

See the following fig for understanding clearly.



Timing Path- Asynchronous Path

As you know that the functionality of set/reset pin is independent from the clock edge. Its level triggered pins and can start functioning at any time of data. So in other way we can say that this path is not in synchronous with the rest of the circuit and that's the reason we are saying such type of path an Asynchronous path.

Other types of Paths:

There are few more types of path which we usually use during timing analysis reports. Those are subset of above mention paths with some specific characteristics. Since we are discussing about the timing paths, so it will be good if we will discuss those here also.

Few names are

- Critical path
- False Path
- Multi-cycle path
- Single Cycle path
- Launch Path
- Capture Path
- Longest Path (also know as Worst Path , Late Path , Max Path , Maximum Delay Path)
- Shortest Path (Also Know as Best Path , Early Path , Min Path, Minimum Delay Path)

Critical Path:

In short, I can say that the path which creates Longest delay is the critical path.

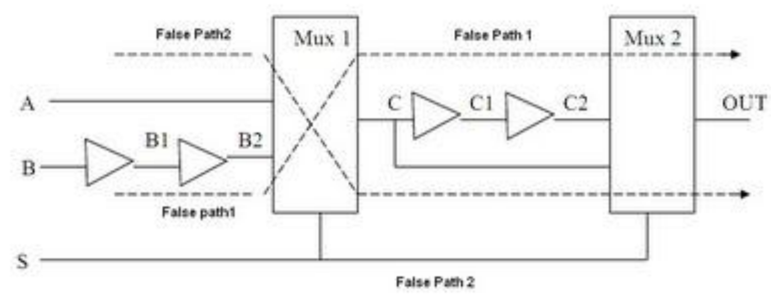
- Critical paths are timing-sensitive functional paths. because of the timing of these paths is critical, no additional gates are allowed to be added to the path, to prevent increasing the delay of the critical path.
- Timing critical path are those path that do not meet your timing. What normally happens is that after synthesis the tool will give you a number of path which have a negative slag. The first thing you would do is to make sure those path are not false or multicycle since it that case you can just ignore them.

Taking a typical example (in a very simpler way), the STA tool will add the delay contributed from all the logic connecting the Q output of one flop to the D input of the next (including the CLK->Q of the first flop), and then compare it against the defined clock period of the CLK pins (assuming both flops are on the same clock, and taking into account the setup time of the second flop and the clock skew). This should be strictly less than the clock period defined for that clock. If the delay is less than the clock period, then the "path meets timing". If it is greater, than the "path fails timing". The "critical path" is the path out of all the possible paths that either exceeds its constraint by the largest amount, or, if all paths pass, then the one that comes closest to failing.

False Path:

- Physically exist in the design but those are logically/functionally incorrect path. Means no data is transferred from Start Point to End Point. There may be several reasons of such path present in the design.

- Some time we have to explicitly define/create few false path with in the design. E.g for setting a relationship between 2 Asynchronous Clocks.
- The goal in static timing analysis is to do timing analysis on all “true” timing paths, these paths are excluded from timing analysis.
- Since false path are not exercised during normal circuit operation, they typically don't meet timing specification,considering false path during timing closure can result into timing violations and the procedure to fix would introduce unnecessary complexities in the design.
- There may be few paths in your design which are not critical for timing or masking other paths which are important for timing optimization, or never occur with in normal situation. In such case , to increase the run time and improving the timing result , sometime we have to declare such path as a False path , so that Timing analysis tool ignore these paths and so the proper analysis with respect to other paths. Or During optimization don't concentrate over such paths. One example of this. e.g A path between two multiplexed blocks that are never enabled at the same time. You can see the following picture for this.



False Path

Here you can see that False path 1 and False Path 2 can not occur at the same time but during optimization it can effect the timing of another path. So in such scenario, we have to define one of the path as false path.

Same thing I can explain in another way (Note- Took snapshot from one of the forum). As we know that, not all paths that exist in a circuit are "real" timing paths. For example, let us assume that one of the primary inputs to the chip is a configuration input; on the board it must be tied either to VCC or to GND. Since this pin can never change, there are never any timing events on that signal. As a result, all STA paths that start at this particular startpoint are false. The STA tool (and the synthesis tool) cannot know that this pin is going to be tied off, so it needs to be told that these STA paths are false, which the designer can do by telling the tool using a "false_path" directive. When told that the paths are false, the STA tool will not analyze it (and hence will not compare it to a constraint, so this path can not fail), nor will a synthesis tool do any optimizations on that particular path to make it faster; synthesis tools try and improve paths until they "meet timing" - since the path is false, the synthesis tool has no work to do on this path.
Thus, a path should be declared false if the designer KNOWS that the path in question is not a real timing path, even though it looks like one to the STA tool. One must be very careful with declaring a path false. If you declare a path false, and there is ANY situation where it is actually a real path, then you have created the potential for a circuit to fail, and for the most part, you will not catch the error until the chip is on a board, and (not) working. Typically, false paths exists

- from configuration inputs like the one described above

- from "test" inputs; inputs that are only used in the testing of the chip,and are tied off in normal mode (however, there may still be some static timing constraints for the test mode of the chip)
- from asynchronous inputs to the chip (and you must have some form of synchronizing circuit on this input) (this is not an exhaustive list, but covers the majority of legitimate false paths).

So we can say that false paths should NOT be derived from running the STA tool (or synthesis tool); they should be known by the designer as part of the definition of the circuit, and constrained accordingly at the time of initial synthesis.

MultiCycle Path:

- A multicycle path is a timing path that is designed to take more than one clock cycle for the data to propagate from the startpoint to the endpoint.

A multi-cycle path is a path that is allowed multiple clock cycles for propagation. Again, it is a path that starts at a timing startpoint and ends at a timing endpoint. However, for a multi-cycle path, the normal constraint on this path is overridden to allow for the propagation to take multiple clocks.
In the simplest example, the startpoint and endpoint are flops clocked by the same clock. The normal constraint is therefore applied by the definition of the clock; the sum of all delays from the CLK arrival at the first flop to the arrival at the D of the second clock should take no more than 1 clock period minus the setup time of the second flop and adjusted for clock skew.
By defining the path as a multicycle path you can tell the synthesis or STA tool that the path has N clock cycles to propagate; so the timing check becomes "the propagation must be less than N x clock_period, minus the setup time and clock skew". N can be any number greater than 1.

Few examples are

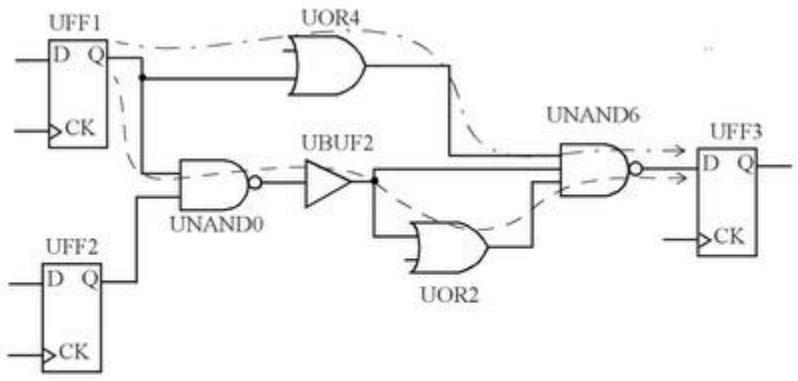
- When you are doing clock crossing from two closely related clocks; ie. from a 30MHz clock to a 60MHz clock,
  - Assuming the two clocks are from the same clock source (i.e. one is the divided clock of the other), and the two clocks are in phase.
  - The normal constraint in this case is from the rising edge of the 30MHz clock to the nearest edge of the 60MHz clock, which is 16ns later. However, if you have a signal in the 60MHz domain that indicates the phase of the 30MHz clock, you can design a circuit that allows for the full 33ns for the clock crossing, then the path from flop30 -> to flop60 is a MCP (again with N=2).
  - The generation of the signal 30MHZ_is_low is not trivial, since it must come from a flop which is clocked by the 60MHz clock, but show the phase of the 30MHz clock.
- Another place would be when you have different parts of the design that run at different, but related frequencies. Again, consider a circuit that has some stuff running at 60MHz and some running on a divided clock at 30MHz.
  - Instead of actually defining 2 clocks, you can use only the faster clock, and have a clock enable that prevents the clocks in the slower domain from updating every other clock,
  - Then all the paths from the "30MHz" flops to the "30MHz" flops can be MCP.
  - This is often done since it is usually a good idea to keep the number of different clock domains to a minimum.

Single Cycle Path:

A Single-cycle path is a timing path that is designed to take only one clock cycle for the data to propagate from the startpoint to the endpoint.

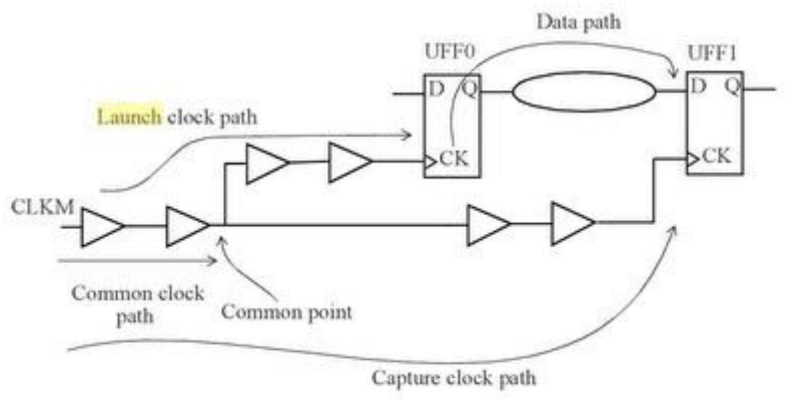
Launch Path and Capture Path:

Both are inter-related so I am describing both in one place. When a flip flop to filp-flop path such as UFF1 to UFF3 is considered, one of the flip-flop launches the data and other captures the data. So here UFF1 is referred to "launch Flip-flop" and UFF3 referred to "capture flip-flop".



These Launch and Capture terminology are always referred to a flip-flop to flip-flop path. Means for this particular path (UFF1->UFF3), UFF1 is launch flip-flop and UFF3 is capture flip-flop. Now if there is any other path starting from UFF3 and ends to some other flip-flop (lets assume UFF4), then for that path UFF3 become launch flip-flop and UFF4 be as capture flip-flop.

The Name "Launch path" referred to a part of clock path. Launch path is launch clock path which is responsible for launching the data at launch flip flop.
And Similarly Capture path is also a part of clock path. Capture path is capture clock path which is responsible for capturing the data at capture flip flop.
This is can be clearly understood by following fig.



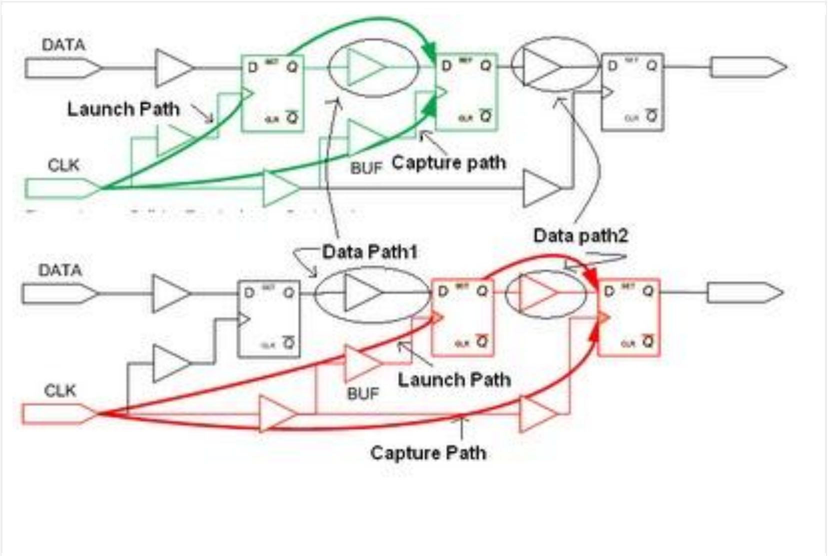
Launch Clock Path (Launch Path) and Capture Clock Path (Capture path)

Here UFF0 is referred to launch flip-flop and UFF1 as capture flip-flop for "Data path" between UFF0 to UFF1.So Start point for this data path is UFF0/CK and end point is UFF1/D.

One thing I want to add here (which I will describe later in my next blog- but its easy to understand here)-

- Launch path and data path together constitute arrival time of data at the input of capture flip-flop.
- Capture clock period and its path delay together constitute required time of data at the input of capture register.

Note: Its very clear that capture and launch paths are correspond to Data path. Means same clock path can be a launch path for one data path and be a capture path for another datapath. Its will be clear by the following fig (source of Fig is From Synopsys).

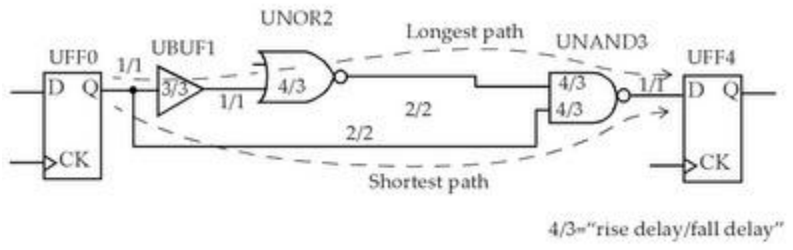


Same clock path behave like Capture and Launch path for different Data path.

Here you can see that for Data path1 the clock path through BUF cell is a capture path but for Data path2 its a Launch Path.

<u>Longest and Shortest Path:</u>

Between any 2 points, there can be many paths.
Longest path is the one that takes longest time, this is also called worst path or late path or a max path.
The shortest path is the one that takes the shortest time; this is also called the best path or early path or a min path.



In the above fig, The longest path between the 2 flip-flop is through the cells UBUF1,UNOR2 and UNAND3. The shortest path between the 2 flip-flops is through the cell UNAND3.

I have tried my best to capture all the important points related to the Timing Paths. Please Let me know If anything is missing here.

PART 2

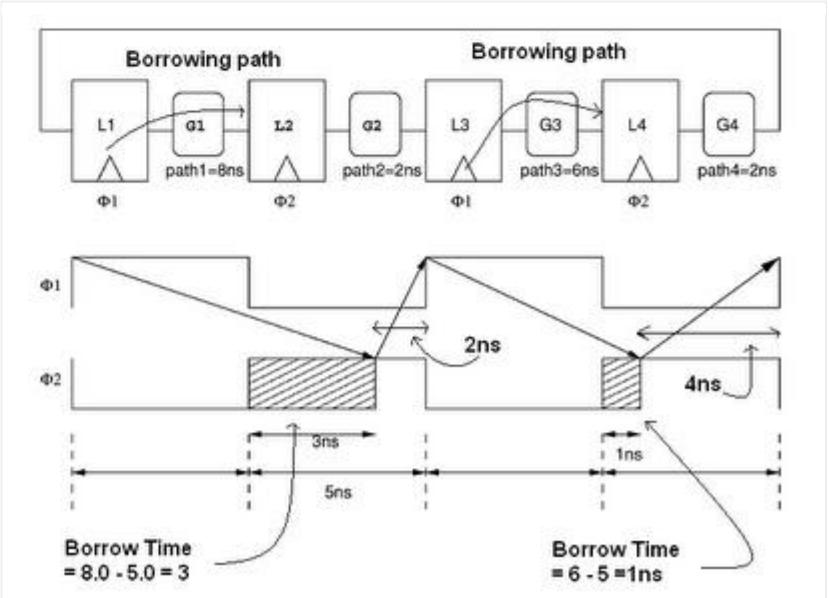
In a ASIC there are majorly two types of component. Flip-flop and other is Latches. Basically Here we will discuss about Latched based timing analysis.
Before this we should understand the basic differences between the latch based design and flip-flop based design.

- Edge-triggered flip-flops change states at the clock edges, whereas latches change states as long as the clock pin is enabled.
- The delay of a combinational logic path of a design using edge-triggered flip-flops cannot be longer than the clock period except for those specified as false paths and multiple-cycle paths. So the performance of a circuit is limited by the longest path of a design.
- In latch based design longer combinational path can be compensated by shorter path delays in the sebsequent logic stages.So for higher performance circuits deisgner are turning to latched based design.

Its true that in the latched based design its difficult to control the timing because of multi-phase clockes used and the lack of "hard" clock edges at which events must occur.

The technique of borrowing time from the shorter paths of the subsequent logic stages to the longer path is called ***time borrowing or cycle stealing***.

Lets talk about this. Please See the following figure.

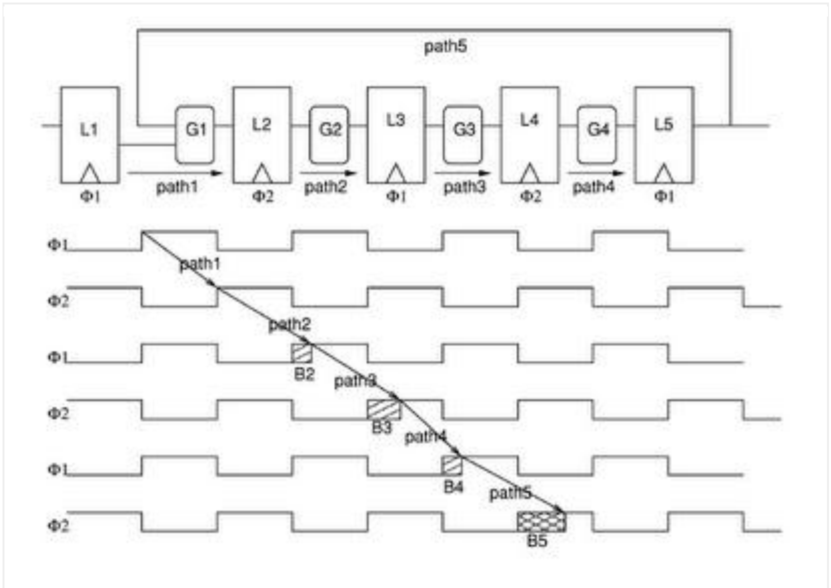


Example of Latched based design.

There are 4 latches (positive level sensitive). L1 and L3 are controlled by PH1 and L2 and L4 are controlled by PH2. G1, G2, G3 and G4 are combinational logic paths. For now assume a library setup time is zero for the latches and zero delay in latch data-path in the transparent mode.
Now if assume that if designs using edge-triggered flip-flops, the clock period has to be at least 8 ns because the longest path in G1 is 8 ns. Now as the clock pulse is 5ns , there is a voilation at L2. On the other hand, if the design uses latches , L2 latch is transparent for another 5ns and since the eighth (8th) ns is within the enabled period of L2, the signal alongpath1 can pass through L2 and continue on path2. Since the delay along path2 is 2 ns, which is short enough to compensate for the overdue delay of path1, this design will work properly. In other word we can say that path1 can borrow sometime (3ns) from the path2. Since the sum of path1 and path2 is 10ns, which is the required time of L3, there will be no voilation in either of the Latches.
For the same reason, path3 can borrow some time (1ns) from path4 without any timing violation.

Note: A latch-based design completes the execution of the four logic stages in 20 ns, whereas an edge-triggered based design needs 32 ns.

Lets see this in a more complex design. Its self explanatory.



Example Of Timing Borrowing

Just wanted to convey here that this Timing borrowing can be multistage. Means we can easily say that for a latched based design, each executing path must start at a time when its driving latch is enabled, and end at a time when its driven latch is enabled.

Few Important things:

- Time borrowing occur with in the same cycle. Means launching and capturing latches be using the same phase of the same clock. when the clocks of the launching and capturing latches are out of phase, time borrowing is not to happen. Usually it was disabled by EDA tools.
-

Time borrowing typically only affects setup slack calculation since time borrowing slows data arrival times. Since hold time slack calculation uses fastest data, time-borrowing typically does not affect hold slack calculation.

Few Important terminology:

Maximum Borrow time:

Maximum Borrow time is the clock pulse width minus the library setup time of the latch. Usually to calculate the maximum allowable borrow time, start with clock pulse width and then substract clock latency , clock reconvergence pessimism removal , library setup time of the endpoint latch.

Negative Borrow time:

If the arrival time minus the clock edge is a negative number, the amount of time borrowing is negative (in other way you can say that no borrowing). This amount is know as Negative Borrow time.

PART 3A

Its been long time, people are asking about Setup and Hold time blog. Finally time come for that. :)

The way we will discuss this concept in the following manner

1. What is SetUp and Hold time?
2. Definition of Setup and Hold.
3. Setup and Hold Violation.
4. How to calculate the Setup and Hold violation in a design?

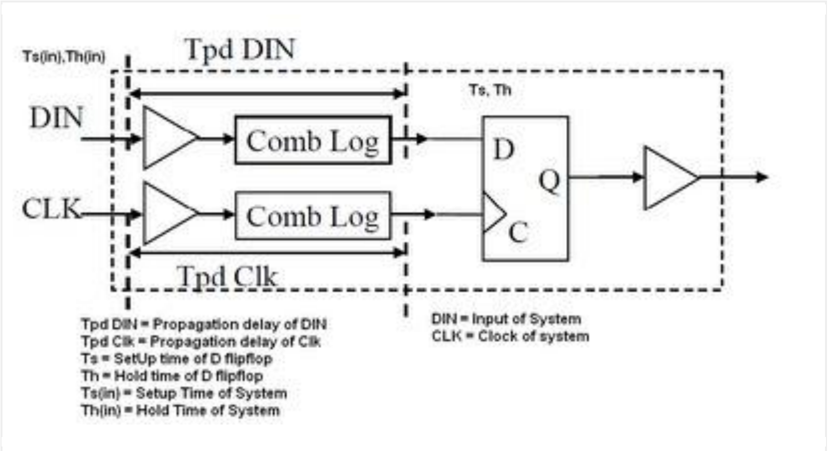
I saw that lots of people are confused with respect to this concept. And the reason of this are

1. They know the definition but don't know the origin or say concept behind Setup and Hold timing.
2. They know the formula for calculating setup and hold violation but don't know how this formula come in picture.
3. They become confuse by few of the terminology like capture path delay, launch path delay, previous clock cycle, current clock cycle, data path delay, slew, setup slew, hold slew, min and max concept, slowest path and fastest path, min and max corner, best and worst case etc during the explanation of Setup and Hold Timings/Violation.

I hope I can clarify your confusion. Let me explain this and if you face any problem let me know.

What is Setup and Hold time?

To understand the origin of the Setup and Hold time concepts first understand it with respect to a System as shown in the fig. An Input DIN and external clock CLK are buffered and passes through combinational logic before they reach a synchronous input and a clock input of a D flipflop (positive edge triggered). Now to capture the data correctly at D flip flop, data should be present at the time of positive edge of clock signal at the C pin (to know the detail just read basis of D flipflop).
Note: here we are assuming D flip flop is ideal so Zero hold and setup time for this.


Ts(in),Th(in)
Tpd DIN
Ts, Th
DIN
Comb Log
D
Q
CLK
Comb Log
C
Tpd Clk
Tpd DIN = Propagation delay of DIN
Tpd Clk = Propagation delay of Clk
Ts = SetUp time of D flipflop
Th = Hold time of D flipflop
Ts(in) = Setup Time of System
Th(in) = Hold Time of System
DIN = Input of System
CLK = Clock of system

SetUp and Hold Time of a System

There may be only 2 condition.

- **Tpd DIN > Tpd Clk**
  - For capture the data at the same time when Clock signal (positive clock edge) reaches at pin C, you have to apply the input Data at pin DIN "Ts(in)=(Tpd DIN) - (Tpd Clk)" time before the positive clock edge at pin CLK.

- In other word, at DIN pin, Data should be stable "Ts(in)" time before the positive clock edge at CLK pin.
- **This Time "Ts(in)" is know as Setup time of the System.**

- **Tpd DIN < Tpd Clk**
  - For capture the data at the same time when clock signal (positive clock edge) reaches at pin C, input Data at pin DIN should not change before "Th(in)= (Tpd Clk) - (Tpd DIN)" time. If it will change, positive clock edge at pin C will capture the next data.
  - In other word, at DIN pin, Data should be stable "Th(in)" time after the positive clock edge at CLK pin.
  - **This time "Th(in)" is know as Hold Time of the System.**

From the above condition it looks like that both the condition can't exist at the same time and you are right. But we have to consider few more things in this.

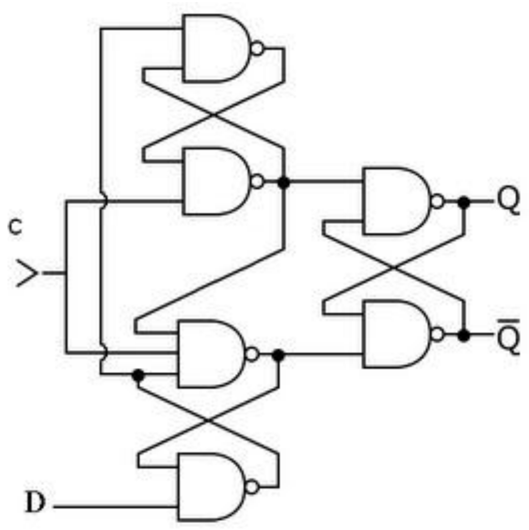
- Worst case and best case (Max delay and min delay)
  - Because of environment condition or because of PVT, we can do this analysis for the worst case (max delay) and best case (min delay) also.
- Shortest Path or Longest path (Min Delay and Max delay)
  - If combinational logic has multiple paths, the we have to do this analysis for the shortest path (min delay) and longest path (max delay) also.

So we can say that above condition can be like this.

- **Tpd DIN (max) > Tpd Clk (min)**
  - **SetUp time == Tpd DIN (max) - Tpd Clk (min)**
- **Tpd DIN (min) < Tpd Clk (max)**
  - **Hold time == Tpd Clk (max) - Tpd DIN (min)**

<u>For example</u> for combinational logic delays are
Data path (max, min) = (5ns, 4 ns)
Clock path (max, min) = (4.5ns, 4.1ns)
Then Setup time= 5-4.1=0.9ns
Hold time is = 4.5-4=0.5ns

Now similar type of explanation we can give for a D flip flop. There is a combinational logic between C and Q , between D and Q of the Flipflop. There are different delays in those conbinational logic and based on there max and min value , a flipflop has Setup and Hold time. One circuitry of the positive edge triggered D flip is shown below.



Positive Edge Triggered D flip-flop

There are different ways for making the D flip flop. Like by JK flipflop, master slave flipflop, Using 2 D type latches etc. Since the internal circuitry is different for each type of Flipflop, the Setup and Hold time is different for every Flipflop.

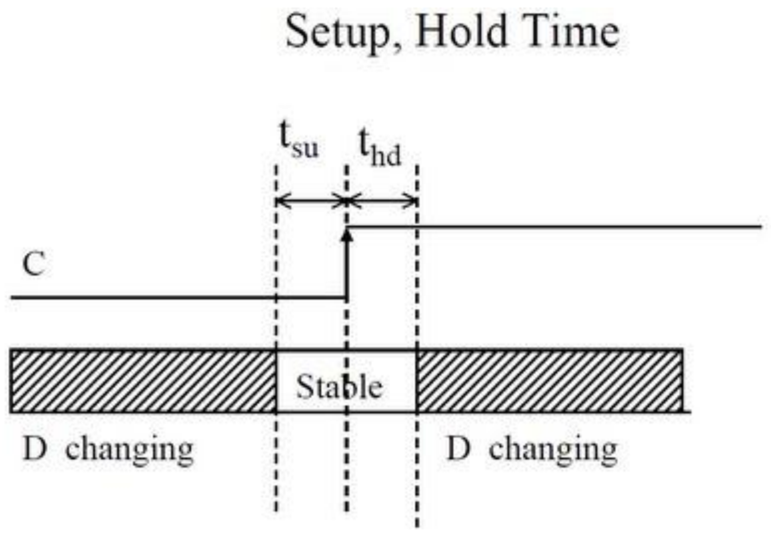
Definition:

Setup Time:

- **Setup time** is the minimum amount of time the data signal should be held steady **before** the clock event so that the data are reliably sampled by the clock. This applies to synchronous circuits such as the flip-flop.
- Or In short I can say that the amount of time the Synchronous input (D) must be stable **before** the active edge of the Clock.
- The Time when input data is available and stable **before** the clock pulse is applied is called Setup time.

Hold time:

- **Hold time** is the minimum amount of time the data signal should be held steady **after** the clock event so that the data are reliably sampled. This applies to synchronous circuits such as the flip-flop.
- Or in short I can say that the amount of time the synchronous input (D) must be stable **after** the active edge of clock.
- The Time **after** clock pulse where data input is held stable is called hold time.



Setup and Hold Violation:

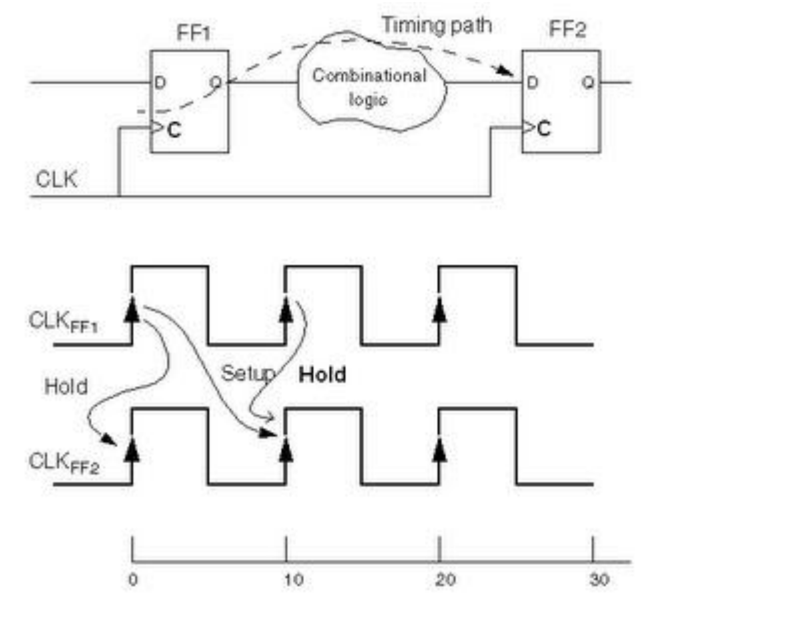
In simple language-
If Setup time is Ts for a flip-flop and if data is not stable before Ts time from active edge of the clock, there is a Setup violation at that flipflop. So if data is changing in the non-shaded area (in the above figure) before active clock edge, then it's a Setup violation.
And If hold time is Th for a flip flop and if data is not stable after Th time from active edge of the clock , there is a hold violation at that flipflop. So if data is changing in the non-shaded area (in the above figure) after active clock edge, then it's a Hold violation.

How to calculate the setup and hold violation in a design.. please see the next blog.

PART 3B

Here we will discuss how to calculate the Setup and Hold Violation for a design.

Till now we have discussed setup and hold violation with respect to the single flipflop, now lets extend this to 2 flip flop. In the following fig there are 2 flipflops (FF1 and FF2).



Single-Cycle Setup and Hold For Flip-Flops

Few important things to note down here-

- Data is launching from FF1/D to FF1/Q at the positive clock edge at FF1/C.
- At FF2/D , input data is coming from FF1/Q through a combinational logic.
- Data is capturing at FF2/D, at the positive clock edge at FF2/C.
- So I can say that Launching Flip-Flop is FF1 and Capturing Flip-Flop is FF2.
- So Data path is FF1/C --> FF1/Q --> FF2/D
- For a single cycle circuit- Signal has to be propagate through Data path in one clock cycle. Means if data is launched at time=0ns from FF1 then it should be captured at time=10ns by FF2.

So for **Setup analysis at FF2**, Data should be stable "Ts" time before the positive edge at FF2/C. Where "Ts" is the Setup time of FF2.

- If Ts=0ns, then , data launched from FF1 at time=0ns should arrive at D of FF2 before or at time=10ns. If data takes too long (greater then 10ns) to arrive (means it is not stable before clock edge at FF2) , it is reported as Setup Violation.
- If Ts=1ns, then, data launched from FF1 at time=0ns should arrive at D of FF2 before or at time=(10ns-1ns)=9ns. If data takes too long (greater then 9ns) to arrive (means it is not stable before 1ns of clock edge at FF2), it is reported as Setup Violation.

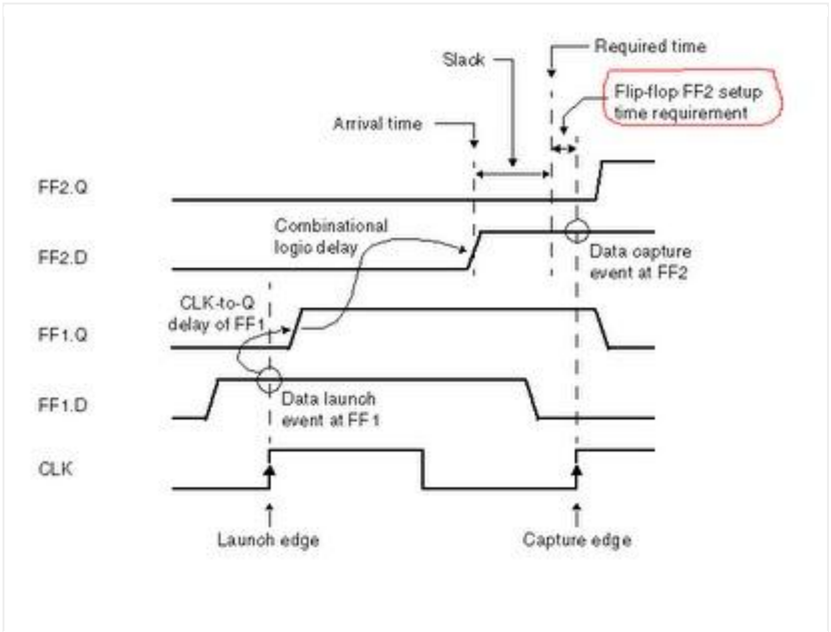
For **Hold Analysis at FF2**, Data should be stable "Th" time after the positive edge at FF2/C. Where "Th" is the Hold time of FF2. Means there should not be any change in the Input data at FF2/D between positive edge of clock at FF2 at Time=10ns and Time=10ns+Th.

- To satisfy the Hold Condition at FF2 for the Data launched by FF1 at 0ns, the data launched by FF1 at 10ns should not reach at FF2/D before 10ns+Th time.
- If Th=0.5ns, then we can say that the data launched from FF1 at time 10ns does not get propagated so soon that it reaches at FF2 before time (10+0.5)=10.5ns (Or say it should reach from FF1 to FF2 with in 0.5ns). If data arrive so soon (means with in 0.5ns from FF1 to FF2, data can't be stable at FF2 for time=0.5ns after the clock edge at FF2), its reported Hold violation.

With the above explanation I can say 2 important points:

1. *Setup is checked at next clock edge.*
2. *Hold is checked at same clock edge.*

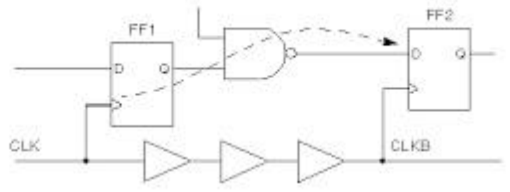
Setup Check timing can be more clear for the above Flip-flop combination with the help of following explanation.

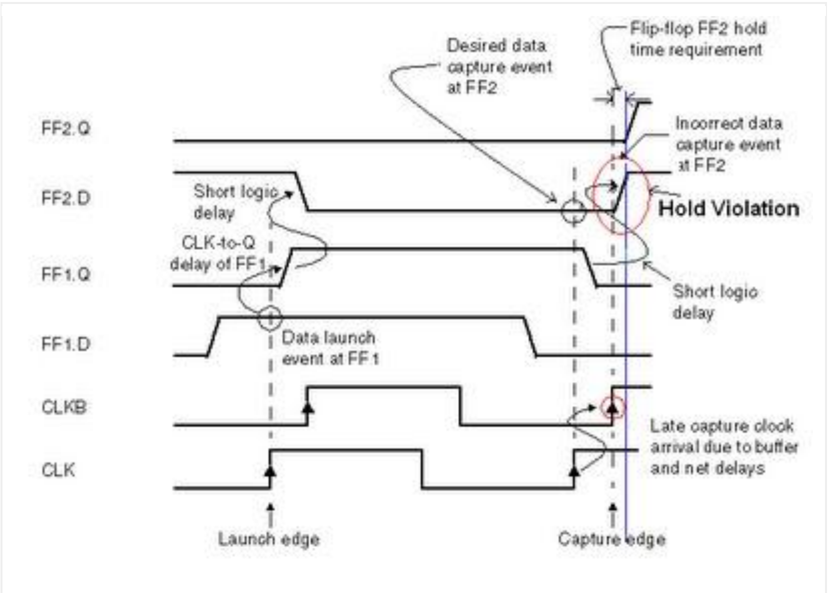


Setup Check Timing

In the above fig you can see that the data launched by FF1/D (at launch edge) reaches at FF2/D after a specific delay (CLK-to-Q delay + Conminational Logic Delay) well before the setup time requirement of Flip-Flop FF2, so there is no setup violation.
From the Fig its clear that if Slack= Required Time - Arrival time < 0 (-ive) , then there is a Setup violation at FF2.

Hold Check timing can be more clear with the help of following circuit and explanation.




Flip-flop FF2 hold time requirement
Desired data capture event at FF2
FF2.Q
Incorrect data capture event at FF2
FF2.D
Short logic delay
Hold Violation
CLK-to-Q delay of FF1
FF1.Q
Short logic delay
FF1.D
Data launch event at FF1
CLKB
Late capture clock arrival due to buffer and net delays
CLK
Launch edge
Capture edge

Hold Check Timing

In the above fig you can see that there is a delay in the CLK and CLKB because of the delay introduced by the series of buffer in the clock path. Now Flip-flop FF2 has a hold requirement and as per that data should be constant after the capture edge of CLKB at Flip-flop FF2. You can see that desired data which suppose to capture by CLKB at FF2.D should be at Zero (0) logic state and be constant long enough after the CLKB capture edge to meet hold requirement but because of very short logic delay between FF1/Q and FF1/D, the change in the FF1/Q propagates very soon. As a result of that there occurs a Hold violation.
This type of violation (Hold Violation) can be fixed by shortening the delay in the clock line or by increasing the delay in the data path.

Setup and Hold violation calculation for the single clock cycle path is very easy to understand. But the complexity increases in case of multi-cycle path ,Gated clock, Flip-flop using different clocks, Latches in place of Flip-Flop. We will discuss all these later sometime.

PART 3C

Till now we have discussed a lot of theory about setup and hold time (with and without Example). Now it's time to discuss the practical implementation of that. Means in a circuit

- How will you calculate the setup and hold values?
- How will you analyze setup and hold violation in a circuit?
- If you have to improve timing of a circuit then what can you do?

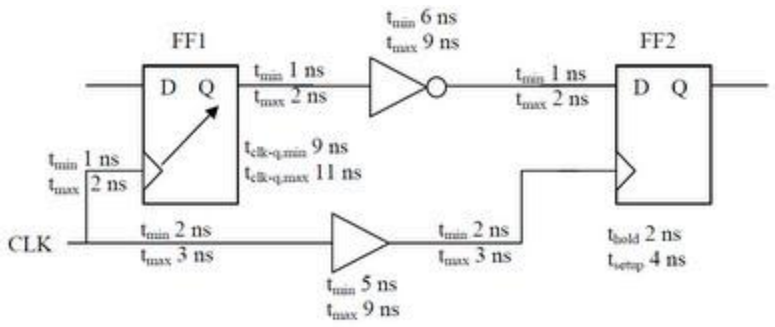
There are few formulas to calculate different parameter (Theory of those I already explained in my previous blogs). I am not going to explain those right now. First we will solve few examples which will give you an basic idea about these formulas, then in the last I will summarize all those in one place.

I saw a lot of confusion with respect to setup and hold timing calculation. Actually there are two things.

- Timing Specification of a Block/Circuit/Library:
  - You have a block with input A and output Y. Some combinational logic is there between A and Y. Now you have to calculate following parameters for that block
    - Setup Time Value at input A
    - Hold Time value at input A.
    - Maximum operating Clock Frequency or Time Period for that block.
    - Clock To Y delay value
    - Input A to Output Y delay value.
- Timing Violation of a circuit:
  - You have to operate a circuit at a particular clock frequency and now you have to find out whether this circuit has any setup or Hold Violation.

So in second case all the parameters are given and you have to find out whether this circuit has any violation or not and In first case you have to find out all the parameters keeping in mind that there should not be any violation.
Lets Discuss in the reverse order.
****
****

Problem1: In the following Circuit, Find out whether there is any Setup Or Hold Violation?



Solution:
<u>Hold Analysis:</u>
When a hold check is performed, we have to consider two things-

- Minimum Delay along the data path.
- Maximum Delay along the clock path.

If the difference between the data path and the clock path is negative, then a timing violation has occurred. (Note: there are few Exceptions for this- We will discuss that some other time)

Data path is: CLK->FF1/CLK ->FF1/Q ->Inverter ->FF2/D

Delay in Data path
= min(wire delay to the clock input of FF1) + min(Clk-to-Q delay of FF1) +min(cell delay of inverter) + min(2 wire delay- "Qof FF1-to-inverter" and "inverter-to-D of FF2")

=Td = 1+9+6+(1+1)=18ns

Clock path is: CLK-> buffer -> FF2/CLK

Clock path Delay
= max(wire delay from CLK to Buffer input) + max(cell delay of Buffer) + max(wire delay from Buffer output to FF2/CLK pin) + (hold time of FF2)
=Tclk = 3+9+3+2 = 17 ns

Hold Slack = Td - Tclk = 18ns -17ns = 1ns
Since Hold Slack is positive-> No hold Violation.

Note: If the hold time had been 4 ns instead of 2 ns, then there would have been a hold violation.
Td=18ns and Tclk = 3+9+3+4=19ns
So Hold Slack=Td - Tclk = 18ns - 19ns = -1ns (Violation)

Setup Analysis:
When a setup check is performed, we have to consider two things-

- Maximum Delay along the data path.
- Minimum Delay along the clock path.

If the difference between the clock path and the data path is negative, then a timing violation has occurred. (Note: there are few Exceptions for this- We will discuss that some other time)

Data path is: CLK->FF1/CLK ->FF1/Q ->Inverter ->FF2/D

Delay in Data path
= max(wire delay to the clock input of FF1) + max(Clk-to-Q delay of FF1) +max(cell delay of inverter) + max(2 wire delay- "Qof FF1-to-inverter" and "inverter-to-D of FF2")
=Td = 2+11+9+(2+2) = 26ns

Note: The first part of the clock path delay (during setup calculation) is the clock period, which has been set to 15 ns. Hope You remember in last blog, I have mentioned very clearly that ***Setup is checked at the next clock cycle.*** *That's the reason for clock path delay we have to include clock period also.*

Clock path is: CLK-> buffer -> FF2/CLK

Clock path Delay
= (Clock period) + min(wire delay from CLK to Buffer input) + min(cell delay of Buffer) + min(wire delay from Buffer output to FF2/CLK pin) - (Setup time of FF2)
=Tclk = 15+2+5+2-4=20ns

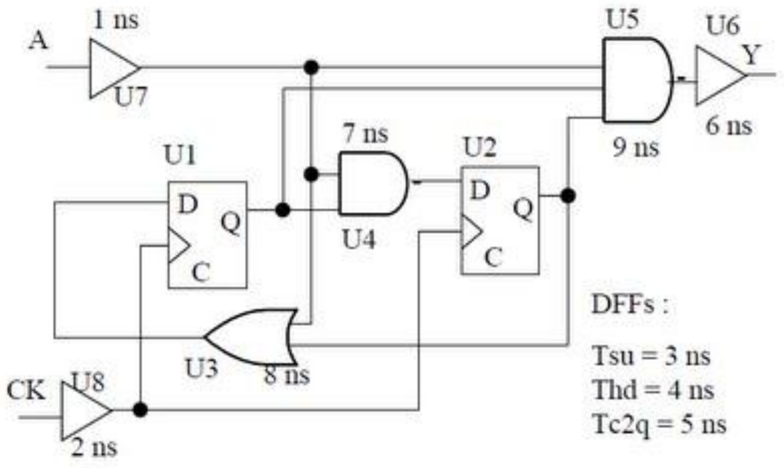
Setup Slack = Tclk - Td = 20ns - 26ns = -6ns.
Since Setup Slack is negative -> Setup violation.

Note: A bigger clock period or a less maximum delay of the inverter solve this setup violations in the circuit.

E.g
If Clock period is 22ns then
Tclk = 22+2+5+2-4=31-4=27ns   AND Td = 26ns
Setup Slack = Tclk - Td = 27-26=1ns (No Violation)

****
****

Problem2: In order to work correctly, what should be the Setup and Hold time at Input A in the following Circuit. Also find out the maximum operating frequency for this circuit. (Note: Ignore Wire delay). Where Tsu- Setup time; Thd-Hold Time; Tc2q- Clock-to-Q delay



Solution:
Step1: Find out the maximum Register to register Delay.

Max Register to Register Delay
= (clk-to-Q delay of U2) + (cell delay of U3) + (all wire delay) + (setup time of U1)
= 5 + 8 + 3 = 16 ns.

Note:
- *There are 2 register to register paths*
  - *U2 -> U3 ->U1 (Delay=5+8+3=16ns)*
  - *U1 -> U4 -> U2 (Delay=5+7+3=15ns)*
- *We have to pick maximum one.*

Step2: Find Out Setup Time:

A setup time = Setup time of Flipflop + Max (Data path Delay) - min(Clock path Delay)
= (Setup time of Flipflop + A2D max delay) - (Clk path min delay)

= Tsu + (Tpd U7 + Tpd U3 + wire delay) - Tpd U8
= 3 + (1+8) - 2 = 10 ns

Note:

- *Here we are not using the Clock period. Because we are not suppose to calculate the Setup violation. We are calculating Setup time. Please refer the part3a for the referance.*
- *All the wire dealy is neglected. If Wire delay present, we have to consider those one.*
- *There are 2 Data path*
    - *A -> U7 -> U4 -> D of U2 (Data path Delay = 1+7 =8ns)*
    - *A -> U7 -> U3 -> D of U1 (Data path Delay = 1+8 =9ns)*
- *Since for Setup calculation we need maximum Data path delay, we have choosen 2nd for our calculation.*

<u>Step3: Find Out Hold Time:</u>

A hold time = Hold time of Flipflop + max(Clock path Delay) - min(Data path delay)
=(Hold time of Flipflop + Clk path max delay) - (A2D max delay)
= Thd + Tpd U8 - (Tpd U7 + Tpd U4+wire delay)
= 4 + 2 - (1+7) = -2 ns

Note: Same explanation as for Setup time. For hold time we need minimum data path , so we have picked first Data path.

<u>Step4: Find out Clock to Out Time:</u>

Clock to Out
= Cell delay of U8 + Clk-to-Q delay of FlipFlop+ Cell delay of U5+ Cell delay of U6+ (all wire delay)
= Tpd U8+ U2 Tc2q + U5 Tpd + U6 Tpd
= 2 + 5 + 9 + 6 = 22 ns

Note:

- *There are 2 Clock to Out path- one from Flip flop U1 and other from U2.*
- *Since in this case the Clk-to-Q path for both Flipflop is same, we can consider any path. But in some other Circuit where the delay is different for both the paths, we should consider Max delay path.*

<u>Step5: Find Pin to Pine Combinational Delay (A to Y delay)</u>

Pin to Pin Combinational Delay (A to Y)
= U7 Tpd + U5 Tpd + U6 Tpd
= 1 + 9 + 6 = 16 ns

<u>Step5: Find Out Max Clock Frequency:</u>

Max Clock Freq = 1/ Max (Reg2reg, Clk2Out, Pin2Pin)
= 1/ Max (16, 22, 16)
= 45.5 Mhz

So summery is:

Parameter	**Description**	**Min**	**Max**	**Units**
Tclk	Clock Period	22		ns

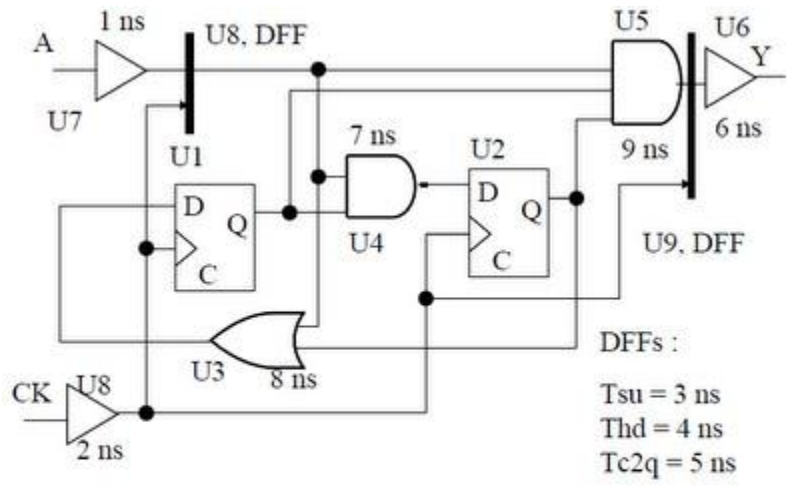
Fclk	Clock Frequency		45.5	Mhz
Atsu	A setup time	10		ns
Athd	A hold time	-2		ns
A2Y	A to Y Tpd		16	ns
Ck2Y	Clock to Y tpd		22	ns

Note: Negative hold times are typically specified as 0 ns.

****
****

Problem3: In the above Circuit, Try to improve the timing by adding any "buffer" or "Register".

Solution:
Best way of doing this is "Register all Input and Output". We are adding DFF so same specification (as U2 and U1).



Now follow all those 5 Steps onn by one.
Step1:

Max Register to Register Delay
U2 Tc2q + U5 Tpd + U9 Tsu = 5 + 9 + 3 = 17 ns

Note:

- *A lot of Register to Register path*
  - *U8 -> U5 -> U9 (Delay = 5+9+3=17ns)*
  - *U8 -> U4 -> U2 (Delay = 5+7+3=15ns)*
  - *U8 -> U3 -> U1 (Delay = 5+8+3=16ns)*
  - *U1 -> U4 -> U2 (Delay= 5+7+3=15ns)*
  - *U1 -> U5 -> U9 (Delay= 5+9+3=17ns)*
  - *U2 -> U5 -> U9 (Delay = 5+9+3=17ns)*

- *U2 -> U3 -> U1 (Delay = 5+8+3=16ns)*

- *Maximum delay is 17ns, Just picked anyone.*

Step2:
A setup time = Tsu + A2D Tpd max - Clk Tpd min
= Tsu + (Tpd U7) - Tpd U8
= 3 + (1) - 2 = 2 ns

Note: Only One path between A and D of FF(i.e U8)

Step3:
A hold time = Thd + Clk Tpd max - A2D Tpd min
= Thd + Tpd U8 - (Tpd U7)
= 4 + 2 - (1) = 5 ns

Note: Only One path between A and D of FF(i.e U8)

Step4:
Clock to out:
=Tpd U8+ U9 Tc2q + U6 Tpd
=2+5+6 = 13 ns

Step5:
No direct link between A and Y. So Not Applicable.

Step6:
Max Clock Freq = 1/ Max (Reg2reg, Clk2Out, Pin2Pin)
= 1/ Max (17, 13)
=58.8 Mhz

Parameter	**Description**	**Min**	**Max**	**Units**
Tclk	Clock Period	17		ns
Fclk	Clock Frequency		58.8	Mhz
Atsu	A setup time	2		ns
Athd	A hold time	5		ns
Ck2Y	Clock to Y tpd		13	ns

****
****

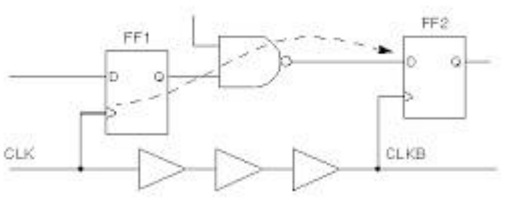
I hope This much will help you. Now its the time to summarize all the important things and formulas.

Points to remember:

1. *Setup is checked at next clock edge.*
2. *Hold is checked at same clock edge.*
3. *For Hold Check (Checking of hold Violation)*

- ***Minimum Delay** along the **data path***.
- ***Maximum Delay** along the **clock path***.

4. *For SetUp Check (Checking of Setup Violation)*
   1. ***Maximum Delay** along the **data path***.
   2. ***Minimum Delay** along the **clock path.***



Basic 2 FlipFlop circuit.

__Calculation of Setup Violation Check__: *C*onsider above circuit of 2 FF connected to each other.

Setup Slack = Required time - Arrival time (since we want data to arrive before it is required)

Where:

Arrival time (max) = clock delay FF1 (max) +clock-to-Q delay FF1 (max) + comb. Delay(max)
Required time = clock adjust + clock delay FF2 (min) - set up time FF2
Clock adjust = clock period (since setup is analyzed at next edge)

__Calculation of Hold Violation Check__: *C*onsider above circuit of 2 FF connected to each other.

Hold Slack = Arrival Time - Required time (since we want data to arrive after it is required)

Where:

Arrival time (min) = clock delay FF1 (min) +clock-to-Q delay FF1 (min) + comb. Delay(min)
Required time = clock adjust + clock delay FF2 (max) + hold time FF2

Clock adjust = 0 (since hold is analyzed at same edge)

<u>Calculation of Maximum Clock Frequency:</u>

Max Clock Freq = 1/ Max (Reg2reg delay, Clk2Out delay, Pin2Pin delay)

Where:

Reg2Reg Delay = Clk-to-Q delay of first FF (max) + conb delay (max) + setup time of 2nd FF.
Clk2Out Delay = Clock delay w.r.t FF (max) + clock-to-Q delay of FF1 (max) + comb. delay (max)
Pin2Pin delay = Comb delay between input pin to output pin (max)

PART 4A

This particular post is inspired by a question asked by Lalit. And Frankly speaking I am not able to resist myself to write a blog on this. I was thinking to capture all this since long but every time because of work I have to drop my thoughts.. But today after reading his question.. I am not able to control myself. :)

So the Question is: (original question)

I have a doubt regarding how delay is calculated along a path.i think there are two ways
1) to calculate max delay and min delay, we keep adding max delays and min delays of all cells(buffer/inverter/mux) from start point to end point respectively.
2)in other way, we calculate path delay for rising edge and falling edge separately. we apply a rise edge at start point and keep adding cell delay. cell delay depends upon input transition and output fanout. so now we have two path delay values for rise edge and falling edge. greater one is considered as Max delay and smaller one is min delay.
which one is correct ?

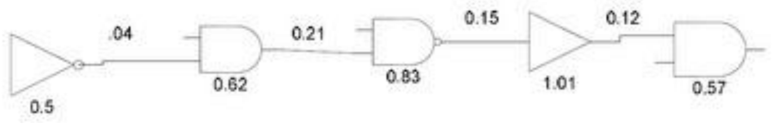
Short Ans is .. both are correct and you have to use both. May be you all become confuse, so let me give you few details.

As I have mention that for Setup and Hold calculation , you have to calculate the Delay of the Timing path (capture path or launch path). Now in a circuit there are 2 major type of Delay.

1. CELL DELAY
   - Timing Delay between an input pin and an output pin of a cell.
   - Cell delay information is contained in the library of the cell. e.g- .lef file
2. NET DELAY.
   - Interconnect delay between a driver pin and a load pin.
   - To calculate the NET delay generally you require 3 most important information.
     - Characteristics of the Driver cell (which is driving the particular net)
     - Load characteristic of the receiver cell. (which is driven by the net)
     - RC (resistance capacitance) value of the net. (It depends on several factor- which we will discuss later)

Both the delay can be calculated by multiple ways. It depends at what stage you require this information with in the design. e.g During pre layout or Post layout or during Signoff timing. As per the stage you are using this, you can use different ways to calculate these Delay. Sometime you require accurate numbers and sometime approximate numbers are also sufficient.

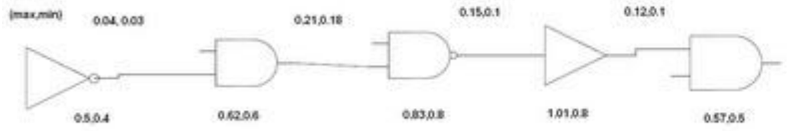
Now lets discuss this with previous background and then we will discuss few new concepts.



Now in the above fig- If I will ask you to calculate the delay of the circuit, then the delay will be

Delay=0.5+0.04+0.62+0.21+0.83+0.15+1.01+0.12+0.57=4.05ns (if all the delay in ns)

Now lets add few more value in this. As we know that every gate and net has max and min value, so in that case we can find out the max delay and min delay. (on what basis these max delay and min delay we are calculating .. we will discuss after that)



So in the above example, first value is max value and 2nd value is min value. So

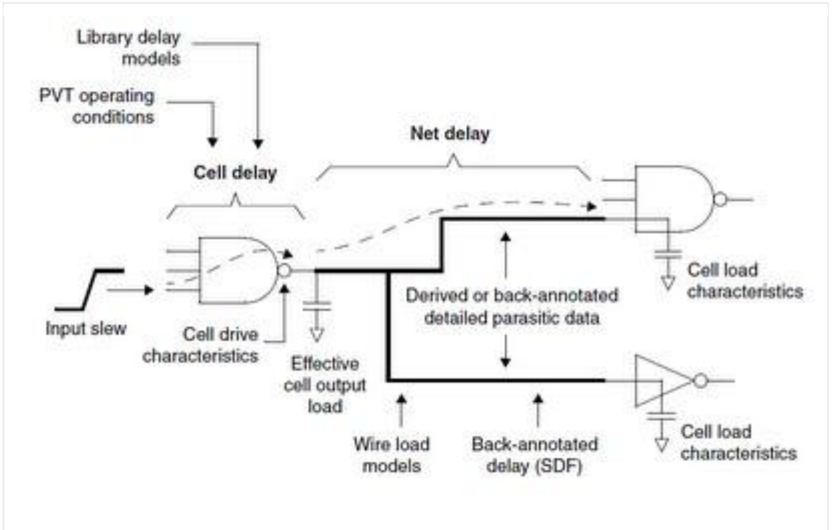
Delay(max)= 0.5+0.04+0.62+0.21+0.83+0.15+1.01+0.12+0.57=4.05ns
Delay(min)= 0.4+0.03+0.6+0.18+0.8+0.1+0.8+0.1+0.5=3.51ns

Till now every one know the concept. Now lets see what's the meaning of min and max delay.

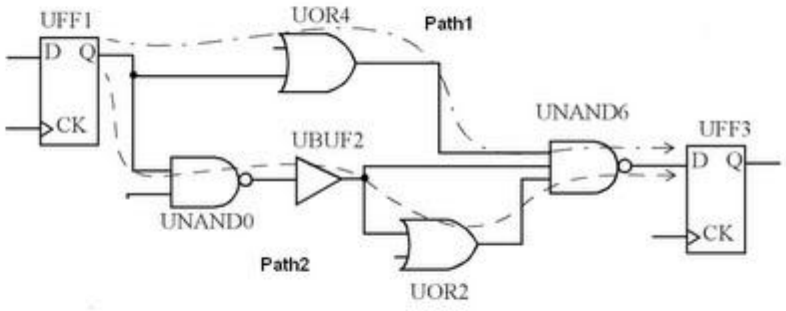
The delay of a cell or net depends on various parameters. Few of them are listed below.

- Library setup time
- Library delay model
- External delay
- Cell load characteristic
- Cell drive characteristic
- Operating condition (PVT)
- Wire load model
- Effective Cell output load
- Input skew
- Back annotated Delay

If any of these parameter vary , the delay vary accordingly. Few of them are mutually exclusive. and In that case we have to consider the effect of only one parameter at a time. If that's the case , then for STA, we calculated the delay in both the condition and then categorize them in worst (max delay) condition or the best condition (min delay). E.g- if a cell has different delay for rise edge and fall edge. Then we are sure that in delay calculation we have to use only one value. So as per their value , we can categorize fall and rise delay of all the cell in the max and min bucket. And finally we come up with max Delay and min delay.


Library delay models
PVT operating conditions
Cell delay
Net delay
Input slew
Cell drive characteristics
Effective cell output load
Derived or back-annotated detailed parasitic data
Cell load characteristics
Wire load models
Back-annotated delay (SDF)
Cell load characteristics
Information used in Cell and net delay calculation (Picture Source - Synopsys)

The way delay is calculated also depends which tool are you using for STA or delay calculation. Cadence may have different algorithm from Synopsys and same is the case of other vendor tools like mentor,magma and all. But in general the basic or say concepts always remain same.
I will explain about all these parameter in detail in next of few blogs, but right now just one example which can help you to understand the situation when you have a lot of information about the circuit and you want to calculate the delay.



In the above diagram, you have 2 paths between UFF1 and UFF3. So when ever you are doing setup and hold analysis, these path will be the part of launch path (arrival time). So lets assume you want to calculate the max and min value of delay between UFF1 and UFF2.

Information1:

	UOR4	UNAND6	UNAND0	UBUF2	UOR2
DELAY(ns)	5	6	6	2	5

Calculation:
Delay in Path1 : 5+6=11ns,
Delay in Path2: 6+2+5+6=19ns,
So

Max Delay = 19ns - Path2 - Longest Path - Worst Path
Min Delay = 11ns - Path1 - Smallest Path - Best Path

Information2:

	UOR4	UNAND6	UNAND0	UBUF2	UOR2
Rise Delay (ns)	5	6	4	1	1
Fall Delay (ns)	6	7	3	1	1

Calculation:
Delay in Path1 : Rise Delay : 5+6=11ns, Fall Delay: 6+7=13ns
Delay in Path2: Rise Delay : 4+1+1+6=12ns, Fall Delay: 3+1+1+7=12ns
So
Max Delay = 13ns -Path1 (Fall Delay)
Min Delay = 11ns - Path1 (Rise Delay)

<u>Note:</u> here there are lot of more concepts which can impact the delay calculation sequence, like unate. We are not considering all those right now. I will explain later.

Information3:

Library	Delay	UOR4	UNAND6	UNAND0	UBUF2	UOR2
Min	Rise Delay (ns)	5	6	4	1	1
	Fall Delay (ns)	6	7	3	1	1
Max	Rise Delay (ns)	5.5	6.5	4.5	1.5	1.5
	Fall Delay (ns)	5.5	6.5	2.5	0.5	0.5

Calculation:
For Min Library:
Delay in Path1 : Rise Delay : 5+6=11ns, Fall Delay: 6+7=13ns
Delay in Path2: Rise Delay : 4+1+1+6=12ns, Fall Delay: 3+1+1+7=12ns
For Max Library:
Delay in Path1 : Rise Delay : 5.5+6.5=12ns, Fall Delay: 5.5+6.5=14ns
Delay in Path2: Rise Delay : 4.5+1.5+1.5+6.5=14ns, Fall Delay: 2.5+0.5+0.5+6.5=10ns
So
Max Delay = 14ns- Path1(Fall Delay)/Path2(Rise Delay)
Min Delay = 10ns - Path2(Fall Delay)

As we have calculated above, STA tool also uses similar approach for finding the Max delay and Min Delay. Once Max and Min delay is calculated then during setup and hold calculation, we use corresponding value.

Once again I am mentioning that all these values are picked randomly. So it may be possible that practically the type/amount of variation in value is not possible.

In next part we will discuss these parameter in detail one by one.

PART 4B

In the previous post we have discussed about the way tool calculate the max and min delay in a circuit. Now we will discuss other basics of the Delay and delay calculation. During your day to day work (in Semiconductor Field) or say in different Books, you come across different terminology related to the delays. There is a long list of that.

- Input Delay
- Output Delay
- Cell Delay
- Net Delay
- Wire Delay
- Slope Delay
- Intrinsic Delay
- Transition Delay
- Connect Delay
- Interconnect Delay
- Propagation Delay
- Min/Max Delay
- Rising/Falling Delay
- Gate Delay
- Stage delay

Fortunately or say luckily out of the above mention long list few are just synonym of other and few are interrelated to each other . Like Net delay also know as Wire Delay , Interconnect delay. Broadly we can divide this Long List into 2 type of delay. Net Delay (Wire delay) and Cell Delay. (**Note :** Stage Delay = Net delay + Cell Delay.)

So let’s discuss these one by one. In digital design, a wire connecting pins of standard cells and blocks is referred to as a NET. A net

- Has only one driver
- Has a number of fanout cells or blocks.
- Can travel on multiple metal layers of the chip.

“Net Delay” refers to the total time needed to charge or discharge all of the parasitic (Capacitance / Resistance / Inductance) of a given Net. So we can say that Net delay is a function of

- Net Resistance

- Net Capacitance
- Net Topology

Now to calculate the Net delay, the wires are modeled in different ways and there are different way to do the calculation. Practically, when you are applying a particular delay model in a design , then you have to apply that to all cells in a particular library. You cannot mix delay models within a single library. There are few recommendations provided by experts or say experienced designer regarding the application of a particular Delay model in a design and that depends on

- Technology of design.
- At what stage you are ? Or say at what stage you want to apply a delay model.
- How accurately you want to calculate the delay.

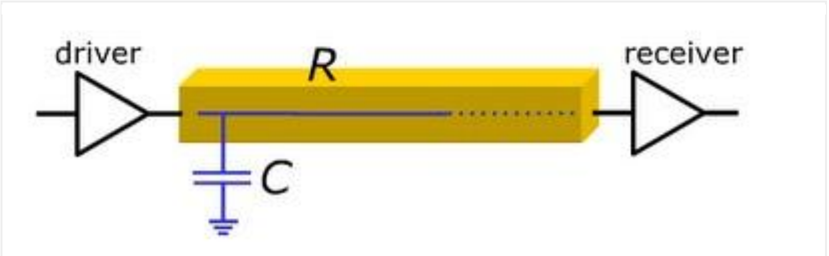
Note : Ideally Till the physical wire is not present in you design, you cannot calculate the Net delay. Reason is ... If wire is not present , you have no idea about the Length/Width of the wires. SO YOU CANN'T CALCULATE THE ACCURATE VALUES OF PARASITIC OR SAY DELAY VALUE OF THE WIRE. But here main point is accurate value, means there is possibility of inaccurate or say approximate value of delay value before physical laying of wire in a design.

There are several delay models. Those which can provide more accurate result, takes more runtime to do the calculation and those which are fast provides less accurate value of delay. Lets discuss few of them. Most popular delay models are -

- Lumped Capacitor Model
- Lumped RC model
- Distributed RC model
  - Pi RC network
  - T RC network
- RLC model
- Wire Load model
- Elmore Delay model
- Transmission Line Model

Lumped Capacitor Model.

- Model assume that wire resistance is negligible.
- Source driver sees a single loading capacitance which is the sum of total capacitance of the interconnect and the total loading capacitance at the sink.
- In past (higher technology-350nm and so), capacitor was dominating and that’s the reason in the model we have only capacitance.
  - Older technology had wide wires,
  - More cross section area implies less resistance and more capacitance.
  - So Wire model only with capacitance.
- In the Fig R=0

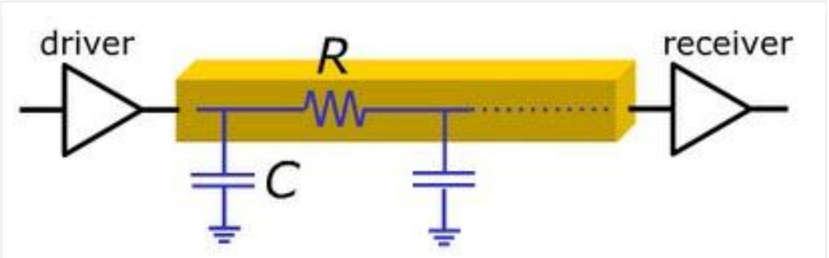


Lumped Capacitor Model

Lumped RC (Resistance Capacitance) model:

- As the feature size decreases to the submicron dimensions, width of the wire reduced.
- Resistance of wire is no longer negligible.
- Have to incorporate the resistance in our model. And that’s the reason Lumped RC model (or say RC tree) comes into picture.

In lumped RC model the total resistance of each wire segment is lumped into one single R, combines the global capacitive into single capacitor C.



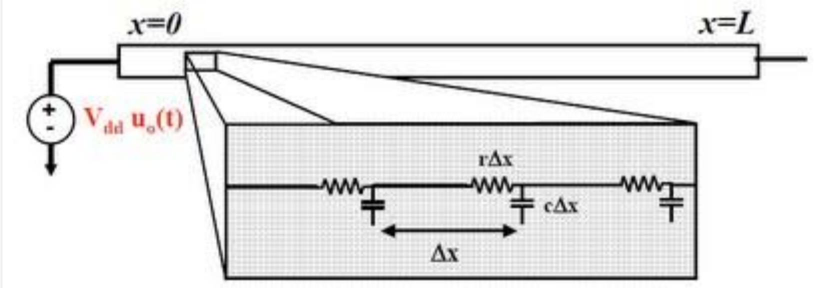
Lumped RC Model

Distributed RC model:

Distributed means RC is distributed along the length of the wire. The total resistance (Rt) and capacitance (Ct) of a wire can be expressed as
Rt = Rp * L
Ct = Cp * L

Where
Cp and Rp are Capacitance and Resistance per unit length.
L is the length of the wire.

Ideally, distributing the resistance and capacitance of a wire in very small portion of the wire (say delta) give you the better performance. Now to find out the total capacitance and resistance we use the differential equation.Distributed RC model provides better accuracy over lumped RC model. But this type of model is not practically possible.



Distributed RC Model

The distributed RC model can be configured by 2 ways based on the structure or say shape (pi and T). Following is the pictorial view.

<u>T model:</u>

- Ct is modeled as a half way of the resistive tree.
- Rt is broken into 2 sections (each being Rt/2)

<u>Pi Model:</u>

- Ct is broken into 2 sections (each being Ct/2) are connected on either side of the resistance.
- Rt is in between the capacitances.

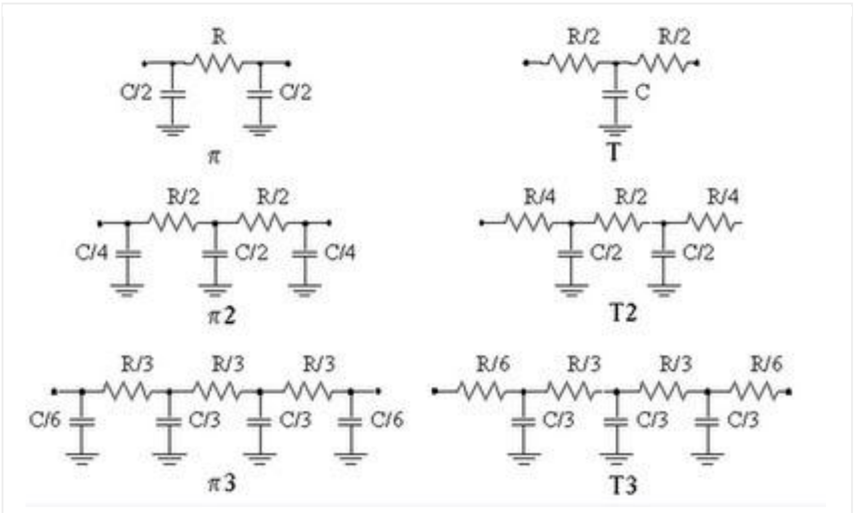
For practical purpose, wire-models with 5-10 elements/nodes are used to model the wire. It will provide the more accurate result. For N element section

<u>For T network:</u>

- Intermediate section of resistance are equal to Rt/N.
- Intermediate section of Capacitance are modeled by Ct/N
- **End section of Resistance are equal to Rt/(2N).**
- This T Network is represented as TN model.

<u>For Pi network:</u>

- Intermediate section of resistance are equal to Rt/N.
- Intermediate section of Capacitance are modeled by Ct/N
- **End section of Capacitance are equal to Ct/(2N).**
- This Pi Network is represented as PiN model.

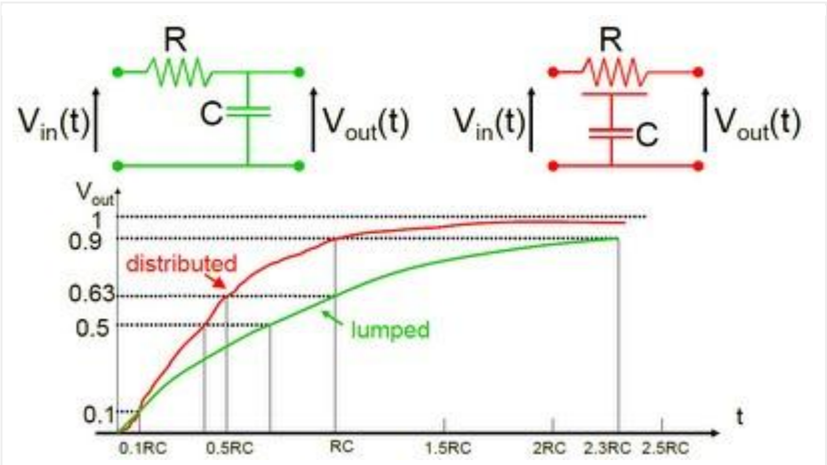


2 Types of Distributed RC Model (Pi- Model and T- Model)

Note: Lumped Vs Distributed RC wire:

Following is the comparison between the Lumped and distributed RC network. It will help you to understand in terms of uses of the both type of network in terms of accuracy and runtime.

Following is the Step Response of Lumped Vs Distributed RC line.



Step Response Of Lumped and Distributed RC Network

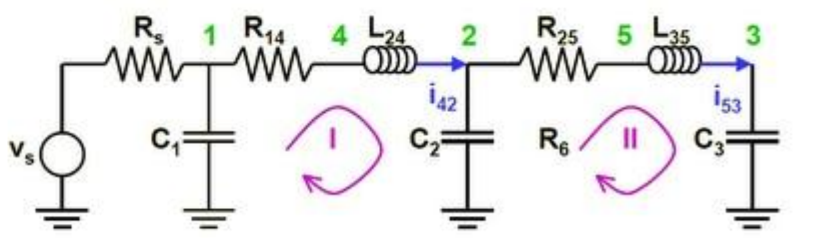
Below comparison Table will give you more accurate picture.

Output Potential range	**Time Elapsed**	
	Distributed RC Network	**Lumped RC network**
0 to 90%	1.0RC	2.3RC
10% to 90% (rise time)	0.9RC	2.2RC

0 to 63%	0.5RC	1.0RC
0 to 50%	0.4RC	0.7RC
0 to 10%	0.1RC	0.1RC

RLC model

In the past since the design frequency was low so the impedance (wL) was dominated by Resistance (wL << R). So we are not caring “L”. However if you are operating at higher frequency and use the wider wire that reduce the resistivity then we have to take account the inductance into our modeling.



Distributed RLC Model

In next part we will discuss Wire Load Delay Model...

PART 4C

Now the question is: What is Wire Load Models (WLM).

Wire loading models

- Used to estimate the interconnect wire delay during pre-layout in a design cycle.
- Wire load information is based on statistics from physical layout parasitic
  - Information from the statistics is used in both conservative and aggressive tables.
  - The conservative tables are based on “mean value” plus 3-sigma; the aggressive tables on “mean value” plus 1-sigma.
- Different for different technology.

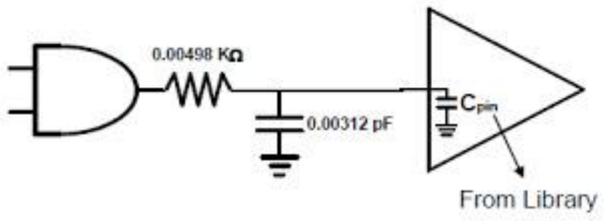
- Wire load models are approximated from one technology to another based on scaling factors. Due to these approximations, the accuracy of these models diminish over multiple technology nodes

- Describes effect of wire length and fanout on
  - Resistance
  - Capacitance
  - Area of the nets.
- All attributes (R, C and Area) are given per unit length wire.
- Slope value is used to characterize linear fanout.
- Basically a set of tables
  - Net fanout vs load
  - Net fanout vs resistance
  - Net fanout vs area

One example of such type of table is:

Net Fanout	Resistance KΩ	Capacitance pF
1	**0.00498**	**0.00312**
2	0.01295	0.00812
3	0.02092	0.01312
4	0.02888	0.01811

As per this



In above circuit - The RC value is estimated and represented as per WLM.

The following are few snapshot of the different format of wire load model.

```
wire_load("WLM1")     {
  resistance  :       0.0006        ;------>R per unit length
  capacitance :       0.0001        ;------> C per unit length
  area :                0.1          ;------> Area per unit length
  slope       :         1.5          ;------> Used for linear extrapolation
  fanout_length(1,  0.002)             ; ------> at fanout "1" length of the wire is 0.002

fanout_length(2,  0.006);
fanout_length(3,  0.009);
fanout_length(4,  0.015);
fanout_length(5,  0.020);

fanout_length(7,  0.028);             ------> at fanout "7" length of the wire is 0.028
fanout_length(8,  0.030);
fanout_length(9,  0.035);
fanout_length(10, 0.040);
}

wire_load("WLM2") {
        fanout_length(  1, 1 );
        fanout_length(  2, 2 );

         fanout_capacitance( 1, 0.002 );
         fanout_capacitance( 2, 0.004 );
         fanout_capacitance( 3, 0.006 );
         fanout_capacitance( 4, 0.008 );
         fanout_capacitance( 5, 0.010 );
         fanout_capacitance( 6, 0.013 );
         fanout_capacitance( 7, 0.015 );
         fanout_capacitance( 8, 0.019 );
         fanout_capacitance( 9, 0.023 );
         fanout_capacitance( 10, 0.027);


         fanout_resistance( 1, 0.01 );
         fanout_resistance( 2, 0.015 );
         fanout_resistance( 3, 0.022 );
         fanout_resistance( 4, 0.026 );
         fanout_resistance( 5, 0.030 );
         fanout_resistance( 6, 0.035 );
         fanout_resistance( 7, 0.039 );
         fanout_resistance( 8, 0.048 );
         fanout_resistance( 9, 0.057 );
         fanout_resistance( 10, 0.06 );
        fanout_area(  1, 0.11 );
        fanout_area( 20, 2.20 );
}
```

Here --
Area, Resistance and Capacitance are in per unit length of the interconnect.
The slope is the extrapolation slop to be used for data points that are not specified in the fan-out length table.

In general, not all fanouts are mentioned in a given WLM lookup table. For example, in above WLM1 and WLM2 lookup table, capacitance and resistance values for fanouts 1, 2, 3, 4, 5, 7, 8, 9, 10 is given. If we want to estimate the values at fanouts in the gaps (e.g. from 6) or outside the fanout range specified in the table (e.g Fanout 20), we have to calculated those value using (linear) interpolation and extrapolation.

<u>For WLM1</u>
For Fanout=20

Since its more than the max value of Fanout available in table (i.e 10) , so we have to perform extrapolation.

Net length = <length of net at fanout 10> + (20-10) x Slope
Resistance = <new calculated Net length at fanout 6> x Resistance or Capacitance value per unit length
Capacitance = <new calculated Net length at fanout 6> x Capacitance value per unit length

Net length = 0.040 + 10 x 1.5 (slope) = 15.04 ----------> length of net with fanout of 20
Resistance = 15.04 x 0.0006 = 0.009024 units
Capacitance = 15.04 x 0.0001 = 0.001504 units

For Fanout=6

Since it’s between 5 and 7 and corresponding fanout Vs length is available, we can do the interpolation.

Net length = ((net length at fanout 5) + (net length at fanout 7)) / 2

Resistance = <new calculated Net length at fanout 20> x Resistance value per unit length
Capacitance = <new calculated Net length at fanout 20> x Capacitance value per unit length

Net length = (0.0020 + 0.0028)/2=0.0048/2=0.0024 ----------> length of net with fanout of 6
Resistance = 0.0024 x 0.0006 = 0.00000144 units

Capacitance = 0.0024 x 0.0001 = 0.00000024 units
In the similar way we can calculate the WLM for any no of fanout value.

WLMs are often used in pre-placement optimization to drive speedups of critical paths. Since timing-driven placement plausibly makes nets on critical paths shorter than average, some optimism may be incorporated into the WLM. Thus, a WLM may actually consist of more than one lookup table, with each table corresponding to a different optimism level. There are several ways to incorporate the optimism level. If we use the WLMs that come from the (ASIC vendor's) design library, usually there are several tables from which we can select. We can also increase the optimism level of a WLM by multiplying all values in the WLM by some factor less than 1.2 For example, we can use 0.25, 0.5, or 0.75.

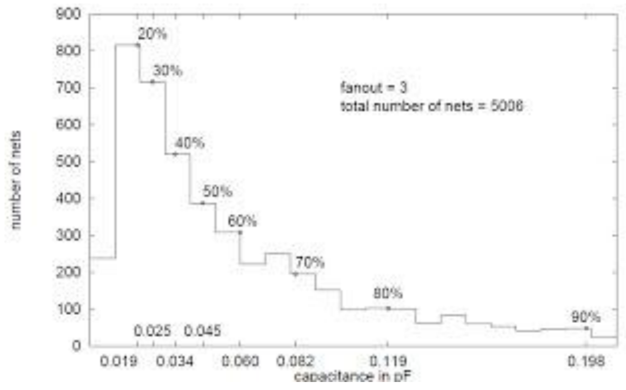
WLM Types
For flows that run timing-based logic optimization before placement, there are three basic types of WLMs that can be used:

1. Statistical WLMs
    - Are based on averages over many similar designs using the same or similar physical libraries.
2. Structural WLMs
    - Use information about neighboring nets, rather than just fanout and module size information.
3. Custom WLMs
    - Are based on the current design after placement and routing, but before the current iteration of preplacement synthesis.

Now the Question is: Where do the wire load models come from?

Normally the semiconductor vendors will develop the models.

ASIC vendors typically develop wireload models based on statistical information taken from a variety of example designs. For all the nets with a particular fanout, the number of nets with a given capacitance is plotted as a histogram. A single capacitance value is picked to represent this fanout value in the wireload model. If a very conservative wireload model is desired, the 90% decile might be picked (i.e. 90% of the nets in the sample have a capacitance smaller than that value).



In this example 90% of nets have a capacitance smaller then 0.198pf. So in the WLM table, you will notice that fanout_capacitance(3, 0.198).
Similar statistics are gathered for resistance and net area.
Usually the vendor supplies a family of wireload models, each to be used for a different size design. This is called *area-based wireload selection*

Few Advance concepts:

Till now we have discussed that for a particular Net you can estimate the RC value as per the WLM. Let me ask you one question. What if your design is hierarchical? Do you think even in that case you can use the same WLM for a particular net which is crossing the hierarchical boundaries? Short ANS is: you can use it but you will lose the accuracy.
Just to solve this problem, Vendors usually supplies multiple WLMs. There are different Modes for WLM analysis- few important are:

WLM analysis has three modes:

1. Top:
   - Consider the design as it has no hierocracy and use the WLM for the top module to calculate delays for all modules.
   - Any low level WLM is ignored.
2. Enclosed:

- Use the WLM of the module which completely encloses the net to compute delay for that net.

3. Segmented:

- If a net goes across several WLM, use the WLM that corresponds to that portion of the net which it encloses only.

PART 5A

This is a general question in most of the interview, what's the maximum clock frequency for a particular circuit? Or Interviewer will provide some data and they will repeat the same question. Many of us know the direct formula and after applying that we can come across the final "Ans" but if someone twist the question. Some -time we become confuse. I motivation of this blog is the same. Several people asked me how to calculate the max-clock frequency. So I thought that it's best if I can write something over this.

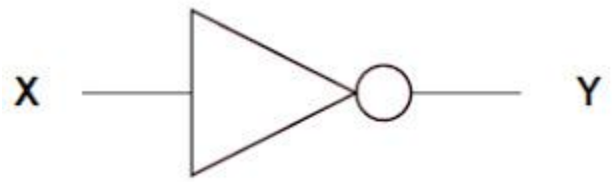
Here I will discuss the same but from basic point of view. It has 3 major sections.

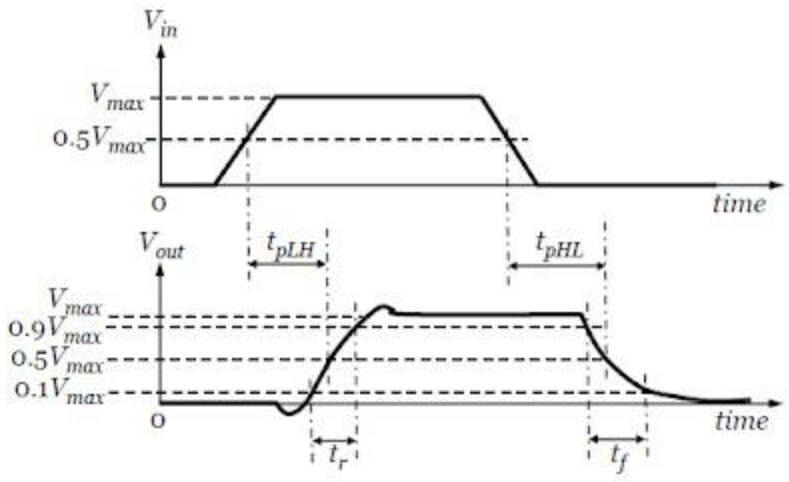
1. In 1st section, we will discuss different definitions with respect to Sequential and combinational Circuits.
2. 2nd Section contains the basics of "Maximum Clock Frequency". I will explain why and how you can calculate the max Clock frequency.
3. I will take few examples and try to solve them. I will make sure that I can capture at least 2-4 examples from easy one to difficult one.

As we know that now a days all the chips has combinational + sequential circuit. So before we move forward, we should know the definition of "Propagation delay" in both types of circuits. Please read it once because it will help you to understand the "Maximum Clock Frequency" concepts.

<u>Propagation Delay in the Combinational circuits:</u>

Let's consider a "NOT" gate and Input/output waveform as shown in the figure





From the above figure, you can define

- **Rise Time (tr):** The time required for a signal to transition from 10% of its maximum value to 90% of its maximum value.
- **Fall Time (tf):** The time required for a signal to transition from 90% of its maximum value to 10% of its maximum value.
- **Propagation Delay (tpLH, tpHL):** The delay measured from the time the input is at 50% of its full swing value to the time the output reaches its 50% value.

I want to rephrase above mention definition as

- This value indicates the amount of time needed to reflect *a permanent change* at an output, if there is any change in logic of input.
- Combinational logic is guaranteed not to show any further output changes in response to an input change after tpLH or tpHL time units have passed.

So, when an input X change, the output Y is not going to change instantaneous. Inverter output is going to maintain its initial value for some time and then it's going to change from its initial value. After the propagation delay (tpLH or tpHL - depends on what type of change- low to high or high to low), the inverter output is stable and is guaranteed not to change again until another input change (here we are not considering any SI/noise effect).

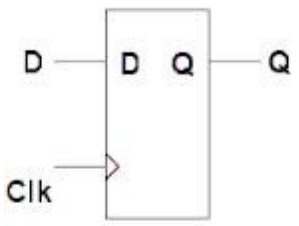
Propagation Delay in the Sequential circuits:

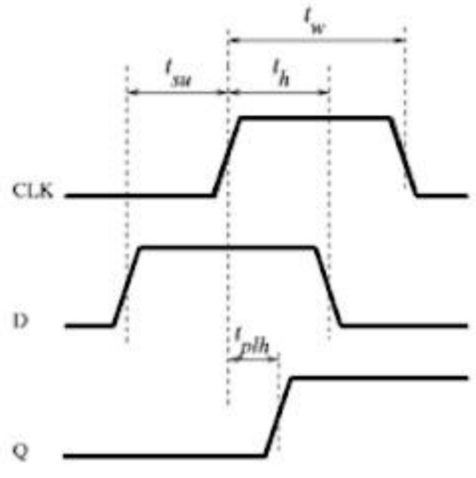
In the sequential circuits, timing characteristics are with respect to the clock input. You can correlate it in this way that in the combinational circuit every timing characteristic/parameter are with respect to the data input change but in the sequential circuits the change In the "data input" is important but change in the clock value has higher precedence. E.g in a positive-edged-triggered Flip-flop, the output value will change only after a presence of positive-edge of clock whether the input data has changed long time ago.

So flip-flops only change value in response to a change in the clock value, timing parameters can be specified in relation to the rising (for positive edge-triggered) or falling (for negative-edge triggered) clock edge.

Note: Setup and hold time we have discussed in detail in the following blogs. Setup and Hold part1; Setup and Hold part2; Setup and Hold part3 . But just to refresh your memories :) , I have captured the definition here along with “propagation delay”.

Let’s consider the positive-edge flip-flop as shown in figure.





D Flip-flop (edge-triggered)
(positive edge triggering)

Propagation delay, tpHL and tpLH , has the same meaning as in combinational circuit – beware propagation delays usually will not be equal for all input to output pairs.

Note: In case of flip-flop there is only one propagation delay i.e tclk-Q (clock→Q delay) but in case of Latches there can be two propagation delays: tClk-Q (clock→Q delay) and tD-Q (data→Q delay). Lation delay we will discuss later.
So again let me rephrase the above mention definition

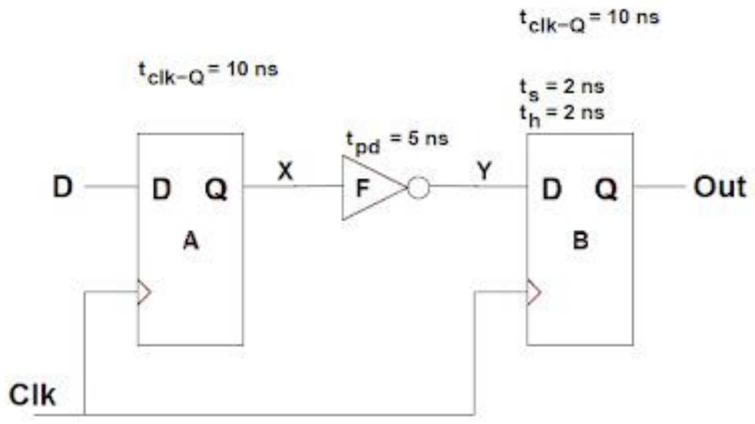
- This value indicates the amount of time needed for a permanent change at the flip-flop output (Q) with respect to a change in the flip flop-clock input (e.g. rising edge).
- When the clock edge arrives, the D input value is transferred to output Q. After tClk−Q (here which is equivalent to tpLH), the output is guaranteed not to change value again until another clock edge trigger (e.g. rising edge) arrives and corresponding Input also.

Setup time (*tsu*) - This value indicates the amount of time **before** the clock edge that data input *D* **must** be stable.
Hold time (*th*) - This value indicates the amount of time **after** the clock edge that data input *D* **must** be held stable.

The circuit must be designed so that the *D* flip flop input signal arrives at least "*tsu"* time units **before** the clock edge and does not change until at least "*th"* time units **after** the clock edge. If either of these **restrictions are violated for any of the flip-flops in the circuit, the circuit will not operate correctly. These** restrictions limit the maximum clock frequency at which the circuit can operate (that's what I am going to explain in the next section J)

The Maximum Clock Frequency for a circuit:

I hope you may be asking that why there is a need of explaining the combinational circuit propagation delay here. Combinational circuit is always independent of clock, so why combination circuit here. J
Now the point is combinational circuit plays a very important role in deciding the clock frequency of the circuit. Let's first discuss an example and try to calculate the circuit frequency, and then we will discuss rest of the things in details. J
Note: Following diagram and numbers, I have copied from one of the pdf downloaded by me long time back.



Now let's understand the flow of data across these Flip-flops.

- Let's assume data is already present at input D of flip-flop A and it's in the stable form.
- Now Clock pin of FF (Flip-Flop) A i.e Clk has been triggered with a positive clock edge (Low to high) at time "0ns".
- As per the propagation delay of the sequential circuit (tclk-Q), it will take at least 10ns for a valid output data at the pin X.
  - Remember- If you will capture the output before 10ns, then no one can give you the guarantee for the accurate/valid value at the pint X.
- This data is going to transfer through the inverter F. Since the propagation delay of "F" is 5ns, it means, you can notice the valid output at the pin Y only after 10ns+5ns=15ns (with reference to the positive clock edge- 10ns of FF A and 5 ns of inverter)
  - Practically this is the place where a more complex combinational circuit are present between 2 FFs. So in a more complex design, if a single path is present between X and Y, then the total

time taken by the data to travel from X to Y is equal to the sum of the propagation delay of all the combinational circuits/devices. (I will explain this in more detail in the next section with more example)

- Now once valid data reaches at the pin Y, then this data supposed to capture by FF B at the next clock positive edge (in a single cycle circuit).
  - We generally try to design all the circuit in such a way that it operates in a single clock cycle. Multiple clock cycle circuit are special case and we are not going to discuss that right now (as someone says – it's out of scope of this blog J)
- For properly capturing the data at FF B, data should be present and stable 2ns (setup time) before the next clock edge as part of setup definition).

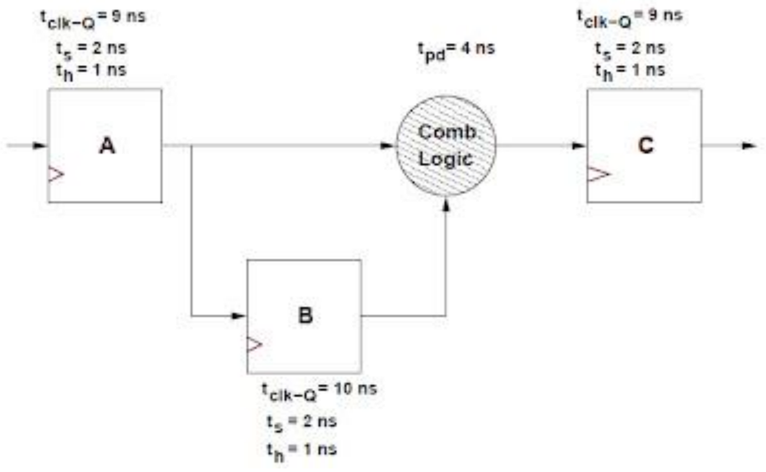
So it means between 2 consecutive positive clock edge, there should be minimum time difference of 10ns +5ns +2ns = 17ns. And we can say that for this circuit the minimum clock period should be 17ns (if we want to operate the circuit in single clock cycle and accurately).
Now we can generalize this
Minimum Clock Period = tclk-Q (A) + tpd (F) + ts (B)
And "**Maximum Clock Frequency = 1/(Min Clock Period)"**

Now at least we have some idea how to calculate the Max clock frequency or Min Clock Period. So even if we will forget the formula then we can calculate our self and we can also prove the logic behind that. Let me use the same concept in few of the more complex design circuit or you can say the practical circuit.

PART 5B

Example 1: Multiple FF's Sequential Circuit

In a typical sequential circuit design there are often millions of flip-flop to flip-flop paths that need to be considered in calculating the maximum clock frequency. This frequency must be determined by locating the longest path among all the flip-flop paths in the circuit. Consider the following circuit.



There are three flip-flop to flip-flop paths (flop A to flop B, flop A to flop C, flop B to flop C). Using an approach similar to whatever I have explained in the last section, the delay along all three paths are:

- TAB = tClk−Q(A) + ts(B) = 9 ns + 2 ns = 11 ns
- TAC = tClk−Q(A) + tpd(Z) + ts(C) = 9 ns + 4 ns + 2 ns = 15 ns
- TBC = tClk−Q(B) + tpd(Z) + ts(C) = 10 ns + 4 ns + 2 ns = 16 ns

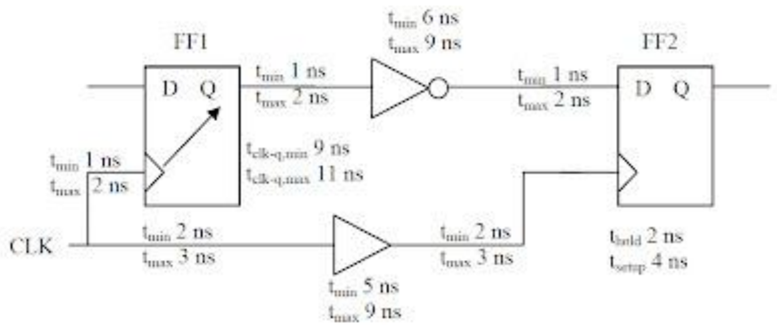
Since the TBC is the largest of the path delays, the minimum clock period for the circuit is Tmin = 16ns and the maximum clock frequency is 1/Tmin = 62.5 MHz.

Example 2: Circuit with min and max delay Specification

Let's consider following circuit. Now this circuit is similar to the normal FF circuitry, only differences are

- Every specification has 2 values (Min and Max).
- There is a combinational circuit in the clock path also.

Note: if you are wondering why there are min and max value (or like from where these values are coming, then you have to refer another blog).



Now let's understand the flow/circuit once again.

- Every interconnect wire also has some delay, so you can see clock CLK will take some time to reach the clock pin of the FF1.
- That's means with reference to original clock edge (let's assume at 0ns), clock edge will take minimum 1ns and maximum 2ns to reach the clock pin of the FF1.
- So in the similar fashion, if we will calculate the total minimum delay and maximum delay.
    - In data path : max delay = (2+11+2+9+2)ns=26ns
    - In data path : min delay = (1+9+1+6+1)ns=18ns
    - In clock path: max delay= (3+9+3)ns=15ns
    - In clock path : min delay = (2+5+2)ns=9ns
- In the last 2 example, there were no delays in the clock path, so it was easy to figure out the minimum clock period. But in this example we have to consider the delay in the clock path also.
- So for minimum clock period, we just want to make sure that at FF2, data should be present at least "tsetup" time before positive clock edge (if it's a positive edged triggered flipflop) at the FF2.

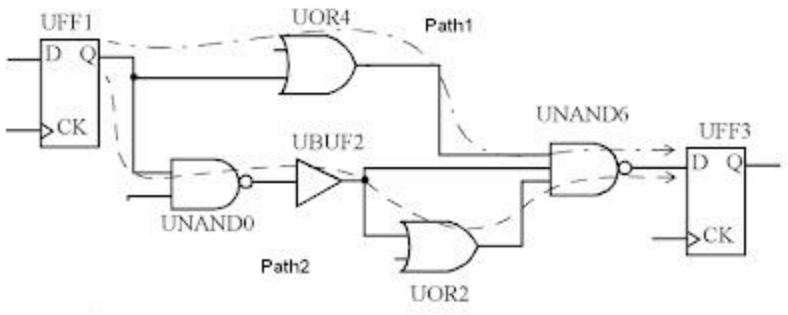
- So Clock edge can reach at the FF2 after 9ns/15ns (min/max) with the reference of original clock edge.
- And data will take time 18ns/26ns (min/max) with the reference of original clock edge.
- So clock period in all the 4 combinations are
  - Clock period (T1)= (Max data path delay)-(max clock path delay)+tsetup=26-15+4=15ns
  - Clock period (T2)= (Min data path delay)-(max clock path delay)+tsetup=18-15+4=7ns
  - Clock period (T3)= (Max data path delay)-(min clock path delay)+tsetup=26-9+4=21ns
  - Clock period (T4)= (Min data path delay)-(min clock path delay)+tsetup=18-9+4=11ns

- Since we want that this circuit should work in the entire scenario (all combination of data and clock path delay), so we have to calculate the period on the basis of that.
  - Now if you will see all the above clock period, you can easily figure out that if the clock period is less than 21ns, then either one or all of the scenarios/cases/combinations fail.
  - **So we can easily conclude that for working of the entire circuit properly**
    - **Minimum Clock Period = Clock period (T3) = (Max data path delay)-(min clock path delay)+tsetup=26-9+4=21ns**

So in general:
Minimum Clock Period = (Max data path delay)-(min clock path delay) + tsetup

And **"Maximum Clock Frequency = 1/(Min Clock Period)"**

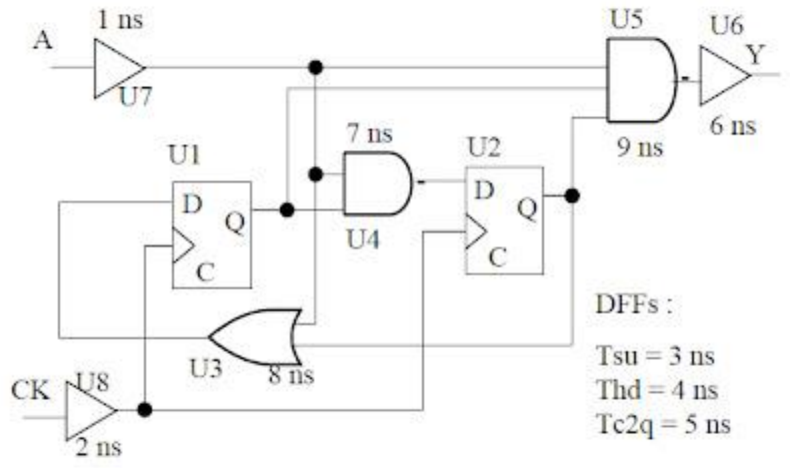
<u>Example 3: Circuit with multiple Combinational paths between 2 FFs:</u>



Now same scenario is with this example. I am not going to explain much in detail. Just it's like that if you have multiple paths in between the 2-flipflops, then as we have done in previous examples, please calculate the delays.

Then calculate the time period and see which one is satisfying all the condition. Or directly I can say that we can calculate the Clock period on the bases of the delay of that path which has big number.
Min Clock Time Period = Tclk-q (of UFF1) + max(delay of Path1,delay of Path2) +Tsetup (of UFF3)

Example 4: Circuit with Different kind of Timing paths:



Since I have mentioned that it has different kind of timing path, so you should know about the timing paths. For that you can refer the (Post link) post. After reading the Timing path, you can easily figure out that in the above circuit there are 4 types of data paths and 2 clock paths

Data path:

1. *Register to register Path*
   - *U2 -> U3 ->U1 (Delay=5+8=13ns)*
   - *U1 -> U4 -> U2 (Delay=5+7=12ns)*
2. *Input pin/port to Register(flip-flop)*
   - *U7 -> U4 -> U2 (Delay=1+7=8ns)*
   - *U7 -> U3 -> U1 (Delay=1+8=9ns)*
3. *Input pin/port to Output pin/port*
   - *U7 -> U5 -> U6 (Delay=1+9+6=16ns)*
4. *Register (flip-flop) to Output pin/port*
   - *U1 -> U5 -> U6 (Delay=5+9+6=20ns)*
   - *U2 -> U5 -> U6 (Delay=5+9+6=20ns)*

Clock path:

- *U8 -> U1 (Delay = 2ns)*
- *U8 -> U2 (Delay =2ns)*

Now few important points- This is not a full chip circuit. In general, recommendation is that you use registers at every input and output port. But for the time being, we will discuss this circuit, considering this as full chip circuit. And you will how much analysis you have to do in this case. Next example, I will add the FFs (registers) at input and output port and then you come to know the difference.

Now let's Study this circuit in more details.

- In this circuit, we have to do the analysis in such a way that if we will apply an input at Port A, then how much time it will take to reach at output Port Y. It will help us to find out the time period of clock.
- Output pin Y is connected with a 3input NAND gate. So if we want a stable out at Y, we have to make sure that all 3 Inputs of NAND gate should have stable data.
- One input of NAND gate is connected with Input pin A with the help of U7.
    - Time take by data to reach NAND gate is 1ns (gate delay of U7)
- Second input pin of NAND gate is connected with output pin Q of Flip flop U2.
    - Time take by data which is present at input D of FF –U2 to reach NAND gate:
        - 2ns(delay of U8)+5ns(Tc2q of FF U2)=7ns
- Third input pin of NAND gate is connected with the output pin Q of Flip Flop U1.
    - Time take by data which is present at input D of FF –U2 to reach NAND gate:
        - 2ns(delay of U8)+5ns(Tc2q of FF U1)=7ns

Note:

- I know you may have doubt that why delay of U8 comes in picture.
    - With reference to the clock edge at CLK pin, we can receive the data at NAND pin after 7ns only (Don't ask me- why we can't take reference in negative?)
- May be you can ask why we haven't consider the setup time of FF in this calculation.
    - If in place of NAND gate, any FF would there then we will consider the setup. We never consider the setup and Tc2q (Clk-2-Q) values of same FF in the delay calculation at the same time. Because when we are considering Clk-2-Q delay, we assume that Data is already present at input Pin D of the FF.

So Time required for the data to transfer from input (A) to output (Y) Pin is the maximum of:

Pin2Pin Delay = U7+U5+U6 = 1+9+6=16ns
Clk2Out (through U1) delay = U8 +U1+U5+U6=2+5+9+6=22ns
Clk2Out (through U2) delay = U8 +U2+U5+U6=2+5+9+6=22ns.

So out of this Clk2Out Delay is Maximum.

From the above Study, you can conclude that data can be stable after 7ns at the NAND gate and maximum delay is 22ns. And you can also assume that this much data is sufficient for calculating the Max Clock Frequency or Minimum Time Period. But that's not the case. Still our analysis is half done in calculating the Max-clock-frequency.

As we have done in our previous example, we have to consider the path between 2 flip-flops also. So the paths are:

- From U1 to U2 (Reg1Reg2)
  - Path delay= 2ns (Delay of U8) + 5ns (Tclk2Q of U1)+7ns (Delay of U4)+3ns (Setup of U2) – 2ns (Delay of U8)=17ns-2ns=15ns
- From U2 to U1 (Reg2Reg1)
  - Path delay = 2ns (Delay of U8) + Tclk2Q of U2 (5ns) + Delay of U3 (8ns) + setup of U1 (3ns) – Delay of U8 (2ns) =18ns -2ns = 16ns.

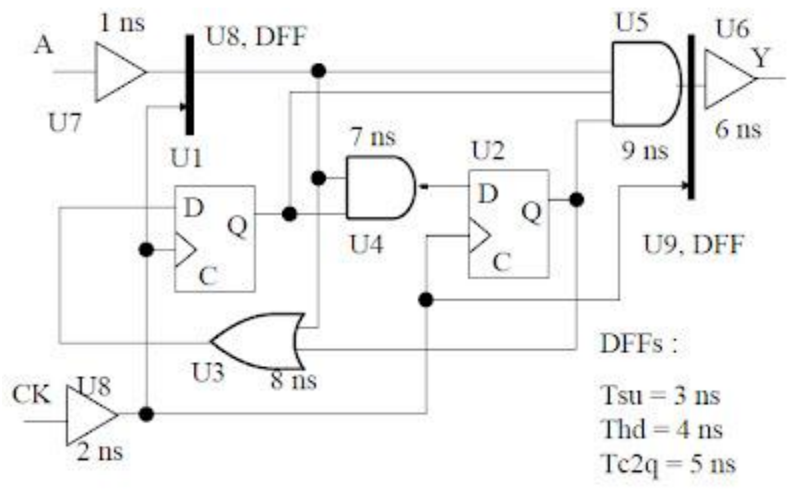
Note:

- I am sure you will ask why did I subtract "Delay of U8" from the above calculation :) because Delay of U8 is common to both the launch and capture path (In case you want to know what's Launch and capture path please follow this post). So we are not supposed to add this delay in our calculation. But just to make it clear, I have added as per the previous logic and then subtracted it to make it clear.

So now if you want to calculate the maximum clock frequency then you have to consider all the delay which we have discussed above.

So
Max Clock Freq = 1/ Max (Reg1Reg2, Reg2Reg1, Clk2Out_1, Clk2Out_2, Pin2Pin)
= 1/ Max (15, 16, 22, 22, 16)
=1/22 =45.5MHz

<u>Example 5: Circuit with Different kind of Timing paths with Register at Input and output ports:</u>



In this example, we have just added 2 FFs U8 at Input pin and U9 at output pin. Now for this circuit, if we want to calculate the max clock frequency then it's similar to example 1.
There are 7 Flip flop to flipflop paths

1. *U8 -> U4 -> U2*
   - *Delay = 5ns+7ns+3ns=15ns*
2. *U8 -> U3 -> U1*
   - *Delay = 5ns+8ns+3ns=16ns*
3. *U8 -> U5 -> U9*
   - *Delay = 5ns+9ns+3ns=17ns*
4. *U1 -> U4 -> U2*
   - *Delay = 5ns +7ns +3ns = 15ns*
5. *U1 -> U5 -> U9*
   - *Delay= 5ns+9ns+3ns=17ns*
6. *U2 -> U5 -> U9*
   - *Delay=5ns+9ns+3ns=17ns*
7. *U2 -> U3 -> U1*
   - *Delay=5ns+8ns+3ns=16ns*

Since the maximum path delay is 17ns,
The Minimum clock period for the circuit should be Tmin = 17 ns
And the Maximum clock frequency is 1/Tmin = 58.8 MHz.

PART 6A

We have discussed few basics about the “Setup and Hold violation” in last few posts. Once designer’s figured out the number of setup and hold violation then their next challenge is: “How to fix these violations”.

EDA tools usually take care but still you have to provide the input (or say proper switch) to fix these violations. That means I can say that “Timing/Routing Tools are enough intelligent to solve most of the timing violation, but still Tools never be more intelligent than the human brain”. ☺

There are different ways to fix these issues and every way has the reason for that. So designers should know what exactly the reason of Issue and what are the different methods (priority wise) or at least different EDA’s switch for fixing those violation.

In this series we will discuss the following things one by one.

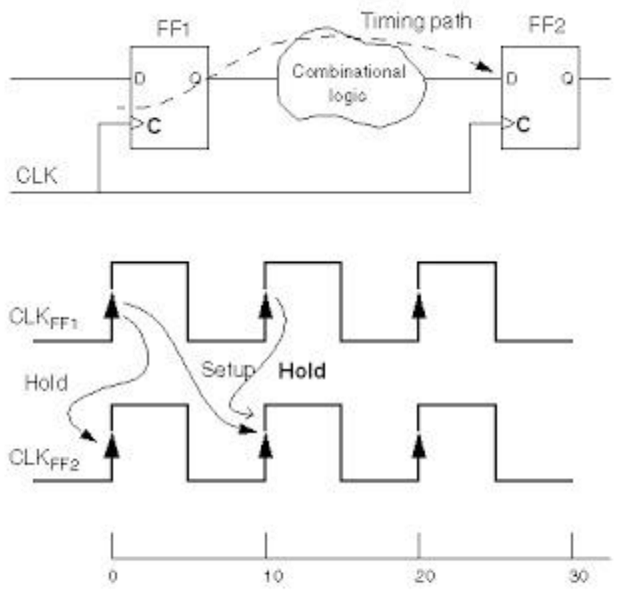
- Basic of Fixing the SETUP and HOLD violations.
  - More Examples here. Very less theory.
  - Few shortcuts/formula/tricks to find out whether these violations are fixable or not. And If fixable, then a rough idea where and how.
- Different ways to fix.
  - Their basics or say physics/Engineering of using that method for fixing.
  - Which method is good and in what scenario you can use them.

Before that- If still you have any doubt regarding the Setup and Hold then please refer following post.
What exactly is the setup and hold – please refer the previous blog.

What are setup and hold violation – please refer the previous blog.

Basics of Fixing the “SETUP and HOLD” violations.

Let’s start with following Diagram and consider this as common for next few examples.



In the following examples we will pick different values of Setup/Hold values of Capture FF and Combinational Path delay. Through these example we will study - How the setup and hold violations are dependent to each other and on the delay of the circuit. If these things are clear then it's very easy for you to understand -- how can we fix the violations and if we are using any particular methods, then why?

Example 1:

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
2ns	1ns	10ns	0ns (Ideal)	0ns (Ideal)	0.5ns

Let's discuss the flow of the data from FF1 to FF2

- Data is going to launch from FF1 at +ive Clock Edge at 0ns and it will reach to FF2 after 0.5ns (combinational logic delay only).
- This data is going to capture at FF2 at +ive Clock Edge at 10ns.

- **As per the Setup definition**, data should be stable 2ns (Setup time of FF2) at FF2 before the +ive Clock Edge (which is at 10ns)
  - In the above case – data become stable 9.5ns before the Clock edge at 10ns (10ns – 0.5ns). That means it satisfy the Setup condition. **NO SETUP VIOLATION.**
- At the FF1 – second set of data is going to launch at t=10ns and it will reach the FF2 in another 0.5ns, means at t=10.5ns.
- This second set of data is going to update/override the first set of data.
- **As per the Hold Definition**, data should be stable till 1ns (Hold time of FF2) at FF2 after the clock edge (which is at t=10ns)
  - In the above case – first set of data is going to override by second set of data at 10.5ns (means just after 0.5ns of the +ive Clock edge at FF2). This means it is not satisfying the hold condition. **HOLD VIOLATION.**

To fix this Hold violation – we have to increase the delay of the Data path so that the second set of data should not reach before t=11ns (10ns+1ns). <u>That means the minimum delay of the Combinational Logic Path should be 1ns for NO HOLD VIOLATION.</u>

That means if you want to fix the HOLD violation, you can increase the Delay of the Data path by any method (we will discuss all those methods in detail – Just keep small patience·💻☽·).

But it doesn't mean that you can increase the Delay by any Value. Let's assume that you have increased the delay of combinational path by adding extra buffer (with delay of 8.5ns). Now new specifications are

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
2ns	1ns	10ns	0ns (Ideal)	0ns (Ideal)	=0.5ns+8.5ns=9ns

As per the Setup definition, data should be stable 2ns (Setup time of FF2) before the Clock Edge (at FF2 which is at 10ns) and with the updated specification – data will be stable at t=9ns, just 1ns before the Clock edge at t=10ns at FF2. That means it is not satisfying the Setup condition. **SETUP VIOLATION**.

Since Data path delay is more than 1ns, there is **NO HOLD VIOLATION** (just we have discussed few paragraph above)

So it means that if we want to fix the setup violation, the Delay of the combinational path should not be more then 8ns (10ns – 2ns). Means 8ns is the maximum value of the Delay of the Combinational Logic path for NO SETUP VIOLATION.

So we can generalize this –

For NO HOLD and SETUP VIOLATION, the delay of the path should be in between 1ns and 8ns.

OR

For Violation free Circuit:

Min delay of Combinational path > Hold time of Capture FF.

Max delay of Combinational path < Clock Period - Setup time of Capture FF.

Example 2:

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
6ns	5ns	10ns	0ns (Ideal)	0ns (Ideal)	0.5ns

Flow of the data from FF1 to FF2:

- Data is going to launch from FF1 at Clock Edge at 0ns and it will reach to FF2 after 0.5ns (combinational logic delay only).
- This data is going to capture at FF2 at Clock Edge at 10ns.
- **As per the Setup definition**, data should be stable 6ns (Setup time of FF2) before the Clock Edge (which is at 10ns)
  - In the above case – data become stable 9.5ns before the Clock edge at 10ns (10ns – 0.5ns). That means it satisfy the Setup condition. **NO SETUP VIOLATION**.
- At the FF1 – second set of data is going to launch at t=10ns and it will reach the FF1 in another 0.5ns, means at t=10.5ns.
- This second set of data is going to update/override the first set of data.

- **As per the Hold Definition**, data should be stable till 5ns (Hold time of FF2) after the clock edge (which is at t=10ns) at FF2
  - In the above case – first set of data is going to override by second set of data at 10.5ns (means just after 0.5ns of the Clock edge at FF2). This means it is not satisfying the hold condition. **HOLD VIOLATION**.

To fix this Hold violation – (As per the previous example) we may increase the delay of the Data path, so that the second set of data should not reach before t=15ns (10ns+5ns). That means the minimum delay of the Combinational Logic Path should be 5ns for NO HOLD VIOLATION.

But Now if you will verify the Setup condition once again (with combination delay of 5ns- which we have assumed for fixing the hold violation) then you come to know that data is going to stable only after 5ns (means 10ns-5ns = 5ns before the clock edge at t-10ns). But as per the setup condition data should be stable before 6ns. So it means now it's not satisfying Setup Condition. Means SETUP VIOLATION.

<u>So in this scenario, we can't fix the setup and hold violation at the same time by adjusting the delay in the combinational logic.</u>

You can also see it directly with the help of minimum and maximum value of combinational delay.

Min delay > Hold time of Capture FF (means 5ns)

Max Delay < Clock Period – Setup time of capture FF (Means 10ns – 6ns = 4ns)

So Min delay > 5ns and Max Delay < 4ns which is not possible.

<u>Now the point is how to fix these violations? Actually this is a non-fixable issue until you just change the clock frequency or replace the FF with lesser setup/hold value. ☺</u>

Let me explain this.

Min delay has dependence only on Hold time, which is fixed for a particular FF.

Max delay has dependence on 2 parameters – Clock Period and Setup time - where Setup time is fixed for a particular FF.

So if you can change the FF with lower setup/hold violation, then you can fix this issue. But in case if that’s not possible then we have to change the Clock period.

<u>In case we are changing the Clock period:</u>

Keep -- Min delay >= 5ns (No HOLD Violation)

Setup violation is by 6ns-5ns =1ns (6ns= Setup time and 5ns = combinational delay). What if we will increase the Clock period by 1ns. Means New clock period should be > 11ns.

So for Clock Period 11ns:

Max delay <= Clock period (11ns) – Setup time (6ns) =5ns.

Now - Max Delay=Min Delay = 5ns. (Neither Hold nor Setup Violation.)

We can generalize-

<u>For Violation Free Circuit</u>

Clock Period >= Setup time + Hold time.

<u>Summary of this Post:</u>

Min delay of Combinational path > Hold time of Capture FF.

Max delay of Combinational path < Clock Period - Setup time of Capture FF.

Clock Period >= Setup time + Hold time.

In the next part we will discuss few more examples with more restrictions. Like-

- What if we can’t reduce the Delay in the Data path?

PART 6B

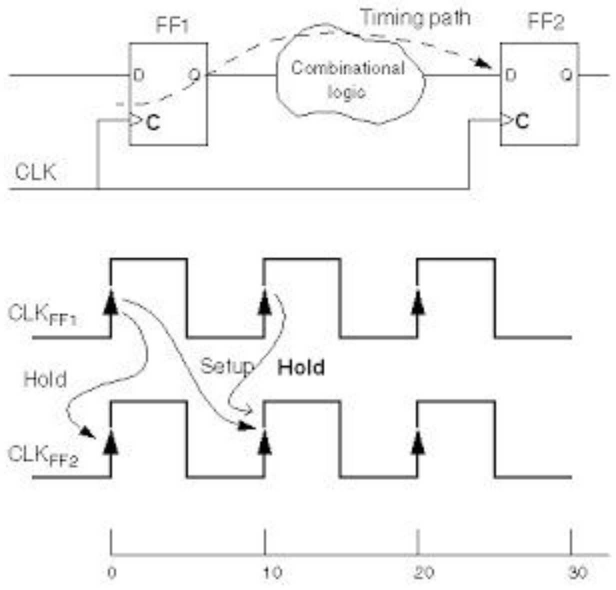
n the last part/post we have discussed 2 examples with different specifications (Both net delay and Tck2Q were ideal means 0ns) and come to know that for Violation free Circuit, following conditions should be satisfied.

Min delay of Combinational path > Hold time of Capture FF.
Max delay of Combinational path < Clock Period - Setup time of Capture FF.

Clock Period >= Setup time + Hold time.

In this post we will discuss few more examples with more restrictions. Like
- What if we can’t reduce the Delay of Data path?

Let’s consider the following figure common to all examples until unless it’s specified



Example 3:

Specification of the FF Circuit

Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
3ns	2ns	10ns	0ns (Ideal)	0ns (Ideal)	5ns (can’t be further reduced).

On the basic of last post …let’s start with checking few conditions directly.
<u>Clock Period Condition: (Satisfied)</u>
Setup time +Hold time = 5ns
Clock period = 10ns
Clock Period > Setup time +Hold time (10> 5)

<u>Min delay / Hold Condition: (Satisfied)</u>
Combinational Delay (5ns) > Hold time.
Means - NO HOLD VIOLATION

<u>Max Delay / Setup Condition: (Satisfied)</u>
Combinational delay (5ns) < Clock period (10ns) – Setup (3ns)
Means - NO SETUP VIOLATION.

This example is just to refresh your memories about the previous post.

Example 4:

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
4ns	3ns	10ns	0ns (Ideal)	0ns (Ideal)	8ns (can’t be further reduced).

<u>Clock Period Condition: (Satisfied)</u>
Setup time +Hold time = 4ns+3ns = 7ns
Clock period = 10ns
Clock Period > Setup time + Hold time (10 > 7)

<u>Min delay / Hold Condition: (Satisfied)</u>
Combinational Delay (8ns) > Hold time (3ns)
Means - NO HOLD VIOLATION

<u>Max Delay / Setup Condition: (Not Satisfied)</u>
Combinational delay (8ns) **Is Not Less Than** “Clock period (10ns) – Setup (4ns)”
Means - SETUP VIOLATION.

Since we can’t change this Combinational delay and also Setup time for the FF, so we have to think something else. ☺. Since we can’t touch the data path, we can try with clock path.

Flow of the data from FF1 to FF2:
- Let’s assume that you have added one buffer of T_capture delay in the clock path between the FF1 and FF2.

- Data is going to launch from FF1 at Clock Edge at 0ns and it will reach to FF2 after 8ns (combinational logic delay only).
- This data is going to capture at FF2 at Clock Edge at 10ns+T_capture. (because of Delay added by Buffer).
- **As per the Setup definition**, data should be stable 4ns (Setup time of FF2) before the Clock Edge at FF2 and in the above case clock edge is at t=T_capture+10ns.

So, for No Setup violation:

=> 8ns (Combinational Delay) < T_capture+10ns (clock period) – 4ns (Setup Time of FF2)
=> 12ns – 10ns < T_capture
=> **T_capture > 2ns**.

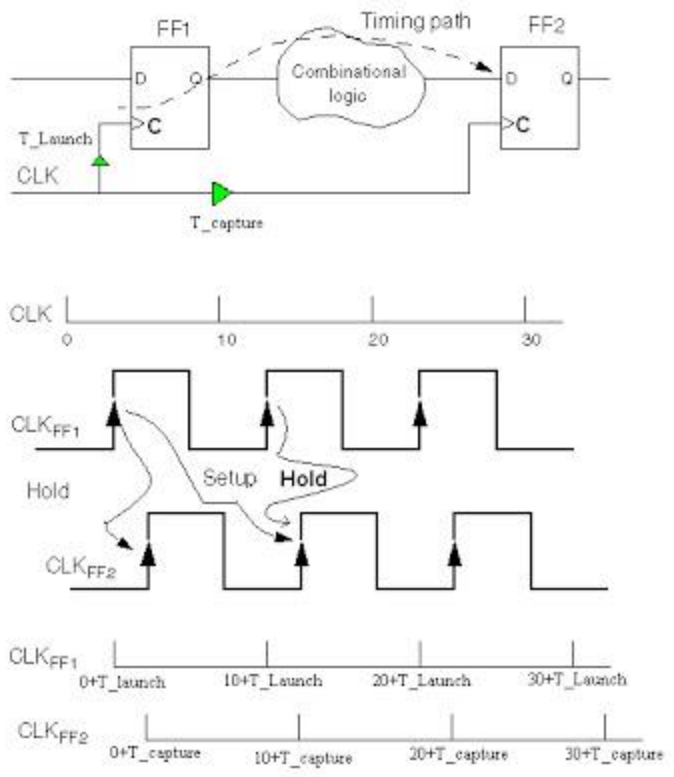
Let's assume if my T_capture = 3ns. Then NO SETUP VIOLATION.

<u>Now, recheck the Hold violation.</u>

- At the FF1 – second set of data is going to launch at t=10ns and it will reach the FF2 in another 8ns, means at t=18ns.
- This second set of data is going to update/override the first set of data present at FF2.
- **As per the Hold Definition**, data should be stable till 3ns (Hold time of FF2) after the clock edge at FF2 (Which is at t=10ns+3ns=13ns – where 3ns is the T_capture).
- That means Data should be remain stable till t=13ns+3ns=16ns.
  - In the above case the second set of data is going to override only after t=18ns. <u>That means first set of data remain Stable till 16ns</u>. Means **NO HOLD VIOLATION.**

Let me Generalize this concept:

I am sure, few people may have question that what will happen if we will add the buffer in the Launch path. Let's discuss that. Please consider the following Diagram for this. In this Launch Clock path has a buffer with a delay of "T_launch" and Capture clock path has another buffer of delay "T_capture".



Specification of the FF Circuit	
Setup	T_setup
Hold	T_hold
Clock Period	Clk_period
Tck2q Delay	0 (Ideal)
Net Delay	0 (Ideal)
Combinational Logic Delay (b/w 2FFs)	Td
Launch Clock path Delay	T_launch
Capture Clock path Delay	T_capture

Let's understand the data flow from FF1 to FF2

- Data is going to launch from FF1 at Clock Edge at T_launch and it will reach to FF2 after Td (combinational logic delay only) that means t= "Td + T_launch".
- This data is going to capture at FF2 at Clock Edge at "Clk_period + T_capture"
- **As per the Setup definition**, data should be stable "T_setup" (Setup time of FF2) time before the Clock Edge at FF2
  - Means data should reach at FF2 before t= "Clk_period + T_capture – T_setup".

So For NO SETUP VIOLATION:

=> T_launch + Td < Clk_period + T_capture – T_setup
=> **Td < Clk_Period + (T_capture - T_launch) – T_setup**

- At the FF1 – second set of data is going to launch at t= “Clk_Period + T_launch” and it will reach the FF2 in another Td, means at t=” Clk_Period + Td + T_launch”.
- This second set of data is going to update/override the first set of data present at FF2.
- **As per the Hold Definition**, data should be stable till “T_hold” (Hold time of FF2) time after the Clock edge (which is at t= “Clk_Period + T_capture”).
  - Means Next set of data should not reach FF2 before t= “Clk_Period + T_capture + T_hold”

So For NO HOLD VIOLATION:

=> Clk_Period + Td + T_launch > Clk_Period + T_capture + T_hold
=> **Td > (T_capture - T_launch) + T_hold**

<u>Summary of this post:</u>

<u>Clock Period Condition:</u>
Clock period > Setup time + Hold Time

<u>Max Delay/ Setup Condition:</u>
Td < Clk_Period + (T_capture - T_launch) – T_setup

<u>Min Delay / Hold Condition:</u>
Td > (T_capture - T_launch) + T_hold

In the next part we will discuss

- More examples which will explain the above conditions in more details.
- How to fix the Setup and hold violation, if we can neither decrease nor increase the Delay in the Data path?

PART 6C

In the last part/post we have discussed 2 more examples with different specifications with more restrictions (Both net delay and Tck2Q were ideal means 0ns) and figure out that if you want to fix the violation by increasing/decreasing the delay in the data path then following condition should be satisfied.

Min delay of Combinational path > Hold time of Capture FF.
Max delay of Combinational path < Clock Period - Setup time of Capture FF.

Clock Period >= Setup time + Hold time.

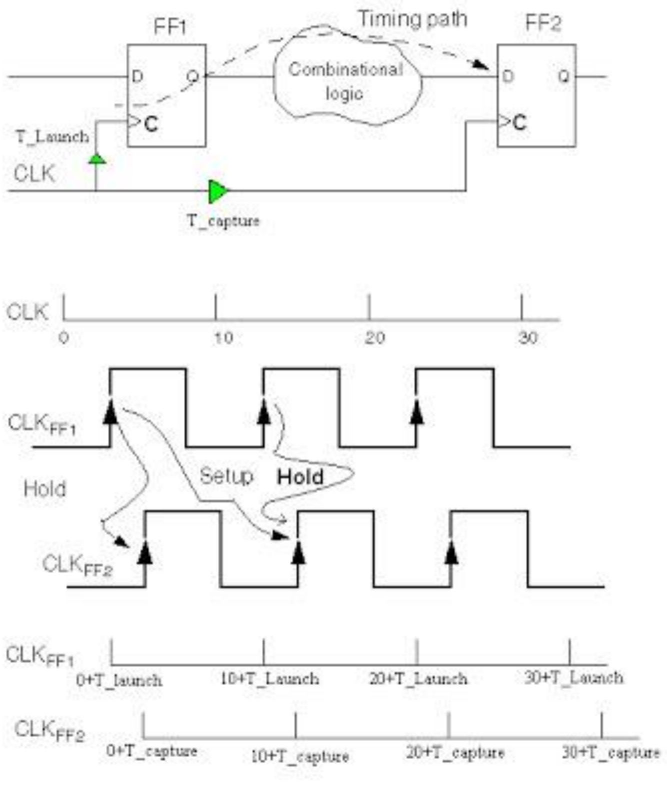
But in case if you can't touch the data path and you have to increase/decrease the delay in the clock path (means between "Clk pin to Launch FF clock pin" Or between "Clk pin and capture FF clock pin"), then following conditions should satisfied.

<u>Max Delay/ Setup Condition:</u>
Td < Clk_Period + (T_capture - T_launch) – T_setup

<u>Min Delay / Hold Condition:</u>
Td > (T_capture - T_launch) + T_hold

Where:
Td -> Combinational path delay (between the 2 FFs)
T_capture -> Delay of circuit present between "Clk pin and capture FF clock pin"
T_launch -> Delay of circuit present between "Clk pin to Launch FF clock pin"

In this post we will discuss few more examples with more restrictions.
Let's consider the following figure common to all examples until unless it's specified.



Example 5:

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
3ns	2ns	10ns	0ns (Ideal)	0ns (Ideal)	11ns (can't be further reduced)

On the basic of last post …let's start with checking few conditions directly.

Clock Period Condition: (Satisfied)
Setup time +Hold time = 5ns
Clock period = 10ns
Clock Period > Setup time +Hold time (10> 5)

Min delay / Hold Condition: (Satisfied)
Combinational Delay (11ns) > Hold time.
Means - NO HOLD VIOLATION

Max Delay / Setup Condition:
Combinational delay (11ns) **Is Not Less Than** "Clock period (10ns) – Setup (3ns)"
Means - SETUP VIOLATION.

Since adding delay in the data path is not going to fix this violation and we can't reduce the combinational delay. So as we have discussed in our last post, we will try with Clock path.

From the last post, if T_capture is the delay of buffer which is inserted between the CLK and Capture's FF and T_launch is the delay of buffer which is inserted between the CLK and Launch's FF, then

Max Delay /Setup condition is :
Td < Clock Period + (T_capture - T_launch) – T_setup
=> 11ns < 10ns – 3ns + (T_capture - T_launch)
=> 11ns < 7ns + (T_capture - T_launch)
=> 4ns < (T_capture - T_launch)

Now we can choose any combination of T_capture and T_launch such that their difference should be less than 4ns.
Note: Remember in the design if you are fixing the violation by increasing or decreasing the delay in the Clock path then always prefer not to play too much with this path.

I never prefer to use T_launch in this case (For setup fixing, I ignore to use T_launch).
So let's assume T_launch =0ns and T_capture = 5ns

Then

11ns < 7ns + 5ns **means no Setup Violation.**

Check once again the Hold condition.
Min delay / Hold Condition:
Td > (T_capture - T_launch) + T_hold
=> 11ns > (T_capture - T_launch) + T_hold
=> 11ns > 5ns + 2ns
=> 11ns > 7ns – **Means No Hold Violation.**

Example 6:

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
3ns	5ns	10ns	0ns (Ideal)	0ns (Ideal)	2ns (can't be further reduced and we can't increase the delay in the data path by any methods)

Let's check the conditions directly.

<u>Clock Period Condition (Satisfied):</u>
Setup time +Hold time = 8ns
Clock period = 10ns
Clock Period > Setup time +Hold time (10ns > 8ns)
Means we can fix violations, if there is any.

<u>Max Delay/ Setup Condition (Satisfied):</u>
Td < Clk_Period + (T_capture - T_launch) – T_setup
Combinational Delay = 2ns
There is no delay in the clock path till now, so T_capture=T_launch=0ns
=> Td (2ns) < Clk_period (10ns) + 0ns – T_setup (3ns)
=> 2ns < 7ns – **Means NO SETUP Violations**

<u>Min Delay / Hold Condition (Not Satisfied):</u>
Td > (T_capture - T_launch) + T_hold
Combinational Delay = 2ns
There is no delay in the clock path till now, so T_capture=T_launch=0ns
=> Td (2ns) **is not greater than** 0ns + T_hold (5ns)
Means HOLD VIOLATION

Since we can't make change in the delay path, so we have to touch the clock path.
For Hold fixing -
=> Td > (T_capture - T_launch) + T_hold
=> 2ns > (T_capture - T_launch) + 5ns
=> -3ns > (T_capture - T_launch)

For Satisfying the above equation T_launch should have more value in comparison to T_capture. We can choose any combination of T_capture and T_launch.

Note: Remember in the design if you are fixing the violation by increasing or decreasing the delay in the Clock path then always prefer not to play too much with this path.

I will never prefer to use T_capture in this case (For Hold fixing, I ignore to use T_capture).
So let's assume T_capture =0ns and T_launch = 4ns

Then

T_launch + Td > 5ns (hold time)
=> 4ns +2ns > 5ns **NO HOLD Violation.**
Check once again the Setup Condition:
Td < Clock Period + (T_capture - T_launch) – T_setup
=> 2ns < 10ns + 0ns -4ns – 3ns
=> 2ns < 3ns **Means No Setup Violation.**

Note: (T_capture - T_launch) also known as CLOCK SKEW. I will explain this later in this blog. Right now, it's Just for your info.

Example 7:

Specification of the FF Circuit					
Setup	Hold	Clock period	Tck2q delay	Net Delay	Combinational Logic Delay
6ns	5ns	10ns	0ns (Ideal)	0ns (Ideal)	0.5ns

Note: this is the same example which we have discussed in the part-6a. Let's check all the conditions one by one.

<u>Clock Period Condition (Not Satisfied):</u>
Setup time +Hold time = 11ns
Clock period = 10ns
Clock Period **is not greater than** Setup time +Hold time
Means we can't fix violations, if there is any.

But still we will try once again with all other conditions, just to prove that above mention condition should be true for fixing the violations.

<u>Max Delay/ Setup Condition (Satisfied):</u>
Td < Clk_Period + (T_capture - T_launch) – T_setup
Combinational Delay = 0.5ns
There is no delay in the clock path till now, so T_capture=T_launch=0ns
=> Td (0.5ns) < Clk_period (10ns) + 0ns – T_setup (6ns)
=> 0.5ns < 4ns – **Means NO SETUP Violations**

<u>Min Delay / Hold Condition (Not Satisfied):</u>
Td > (T_capture - T_launch) + T_hold
Combinational Delay = 0.5ns
There is no delay in the clock path till now, so T_capture=T_launch=0ns
=> Td (0.5ns) **is not greater than** 0ns + T_hold (5ns)
Means HOLD VIOLATION

If you want to fix the Hold violation, then we have already seen that by increasing/decreasing the delay in the data path it can't be fixed. Even if this will fixed, then Setup violation will occur. Let's Try with T_capture or T_launch. Means by adding delay in the clock circuit.

As per the above equations/conditions and corresponding values:
Max Delay/ Setup Condition :
Td < Clock Period + (T_capture - T_launch) – T_setup
=> Td < 10ns -6ns + (T_capture - T_launch)
=> Td < 4ns + (T_capture - T_launch)

Min Delay / Hold Condition:
Td > (T_capture - T_launch) + T_hold
=> Td > (T_capture - T_launch) + 5ns

Remember all 3 variable Td,T_capture,T_launch are positive number.
Possible values of (T_capture - T_launch) = +/-A (where A is a positive number)

Case 1: (T_capture - T_launch) = +A

=> Td < 4ns+A - Condition (a)
=> Td> 5ns+A – Condition (b)
Satisfying both the conditions ("a" and "b") not possible for any +ive value of A.

Case 1: (T_capture - T_launch) = -A

=> Td< 4ns-A => Td+A < 4ns - Condition (a)
=> Td> 5ns-A => Td +A > 5ns - Condition (a)
Satisfying both the conditions ("a" and "b") not possible for any +ive value of A.

That means, I am successfully able to prove that if following condition is not satisfied then you can't fix any type of violation by increasing/decreasing delay in either data_path or clock_path.

Clock Period > Setup time + Hold time.

Summary of this post:

Clock Period Condition:
Clock period > Setup time + Hold Time
For fixing any type of violation (without changing Clock period) - This condition should be satisfied.

Max Delay/ Setup Condition:
Td < Clk_Period + (T_capture - T_launch) – T_setup
For Fixing the Setup Violation – Always prefer T_capture over T_launch

Min Delay / Hold Condition:
Td > (T_capture - T_launch) + T_hold
For Fixing the hold Violation – Always prefer T_launch over T_capture.

Till now we have discussed almost all the necessary basic of fixing the violation of Setup and Hold time. You have been noticed that everywhere I have talked about the increasing/decreasing the delay. If I have mentioned anywhere adding/removing the buffer, that also mean increasing/decreasing the delay.

There are several other ways through which you can increase/decreasing the delay of the circuit. In the next post we will discuss

- Different methods for increasing/decreasing the delay in a circuit/path.
- Also try to capture the basics behind above said methods one by one