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.

What is Timing Analysis??

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

There are a couple of reasons for performing timing analysis.

- 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.

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

Types of Timing Analysis:

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.

Clock related:

- 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

---

Static Timing analysis is divided into several parts:

- Part1 -> Timing Paths
- Part2 -> Time Borrowing
- Part3a -> Basic Concept Of Setup and Hold
- Part3b -> Basic Concept of Setup and Hold Violation
- Part3c -> Practical Examples for Setup and Hold Time / Violation
- Part4 -> Delay

Note: Part 4, 5 and 6 are still under development.

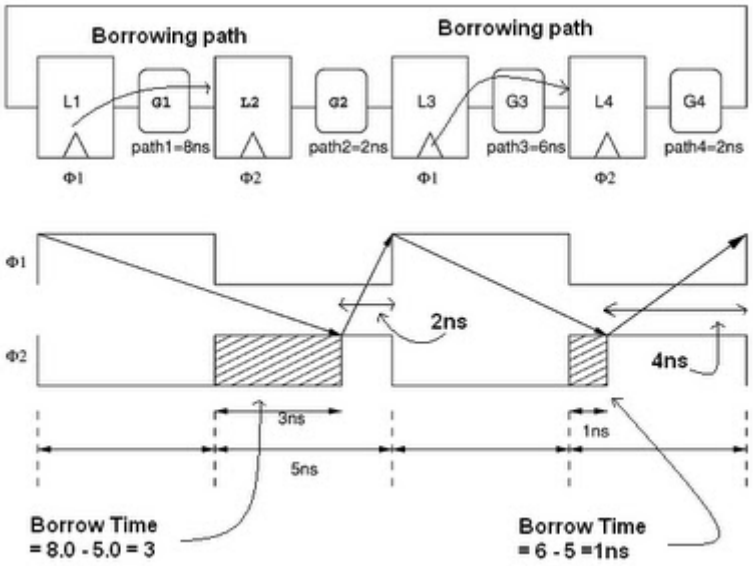
In a ASIC there are majorly two types of component. Flip-flop and other is Latches. Basically Here we will discuss about Latched based timing analysis.
Before this we should understand the basic differences between the latch based design and flip-flop based design.

- 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.



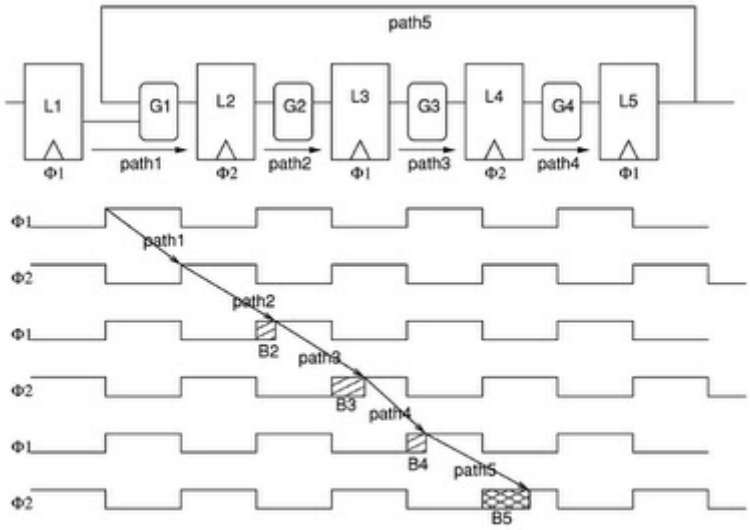
Example of Latched based design.

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 along path1 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.

++

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:

- Part1 -> Timing Paths
- Part2 -> Time Borrowing
- Part3a -> Basic Concept Of Setup and Hold
- Part3b -> Basic Concept of Setup and Hold Violation
- Part3c -> Practical Examples for Setup and Hold Time / Violation
- Part4 -> Delay

Note: Part 4, 5 and 6 are still under development.

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.

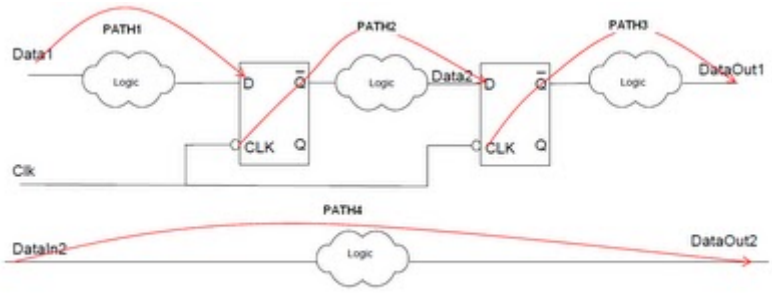
- Data path
  - Start Point
    - Input port of the design (because the input data can be launched from some external source).
    - Clock pin of the flip-flop/latch/memory (sequential cell)
  - End Point
    - Data input pin of the flip-flop/latch/memory (sequential cell)
    - Output port of the design (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)

<u>Data Paths:</u>

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:

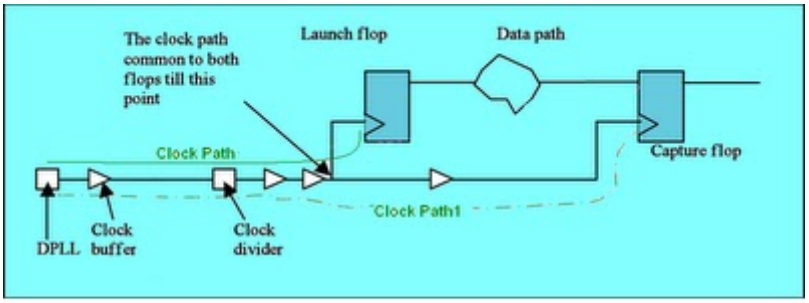


Timing Path- 4 types of Data Path

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)

Clock Path:

Please check the following figure



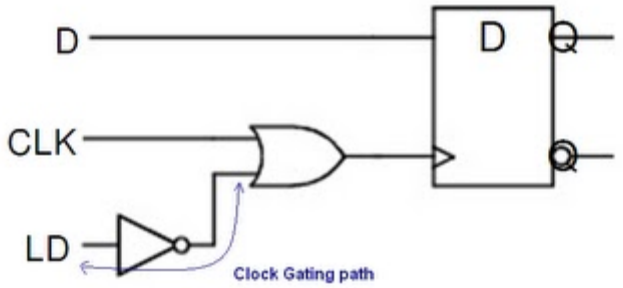
Timing Paths- Clock Paths

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.

Clock Gating Path:

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



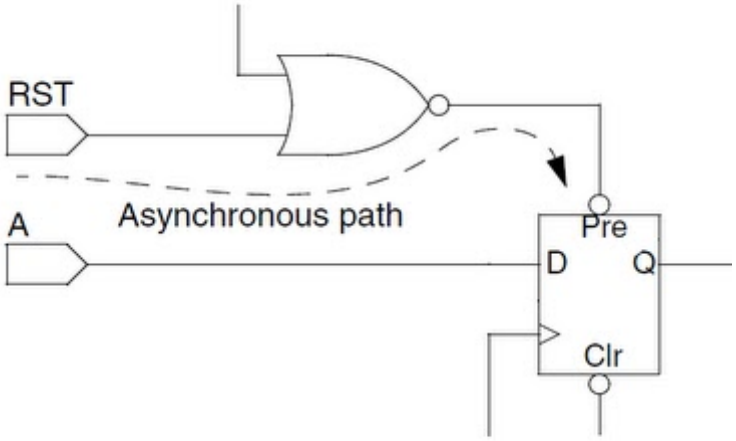
Timing Path- Clock Gating path.

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.

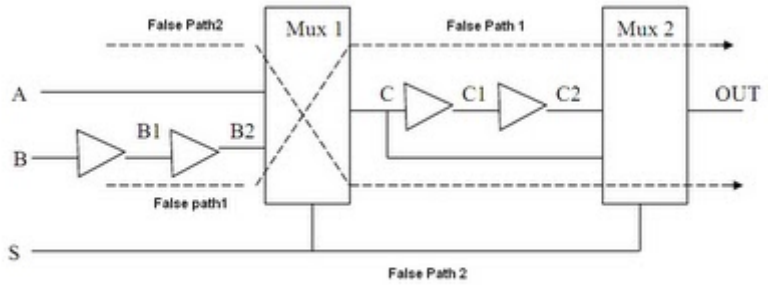
- Critical paths are timing-sensitive functional paths. because of the timing of these paths is critical, no additional gates are allowed to be added to the path, to prevent increasing the delay of the critical path.
- Timing critical path are those path that do not meet your timing. What normally happens 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,

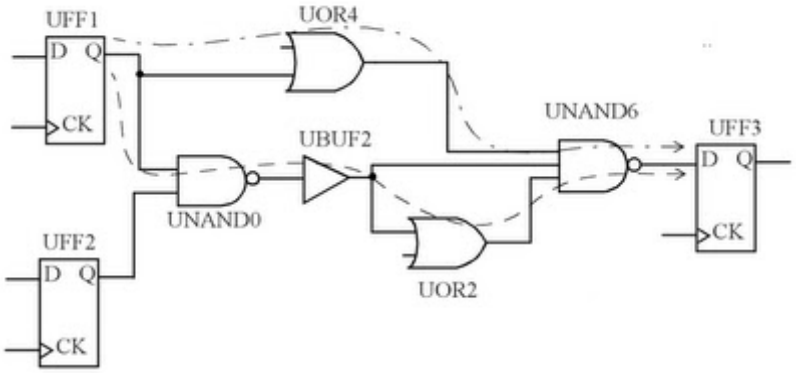
- Then all the paths from the "30MHz" flops to the "30MHz" flops can be MCP.
- This is often done since it is usually a good idea to keep the number of different clock domains to a minimum.

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.

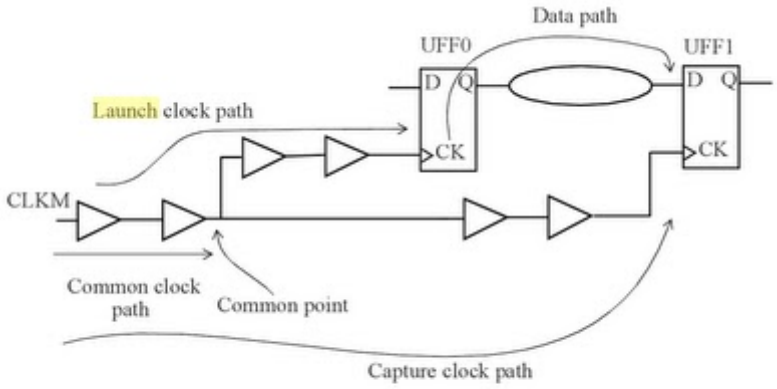
Launch Path and Capture Path:

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.



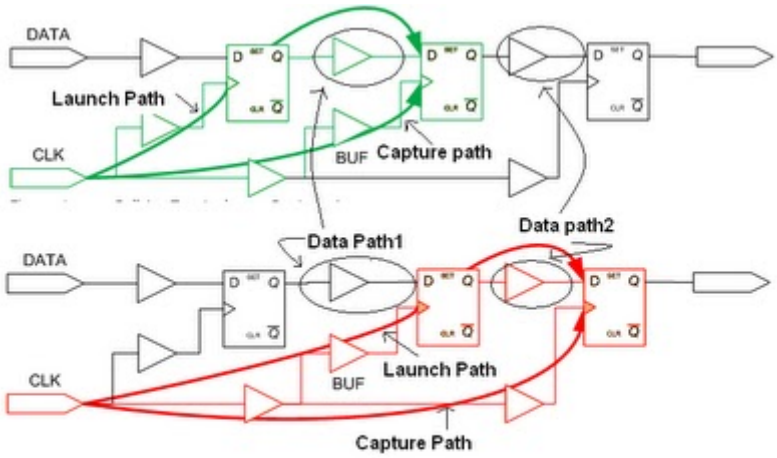
Launch Clock Path (Launch Path) and Capture Clock Path (Capture path)

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).



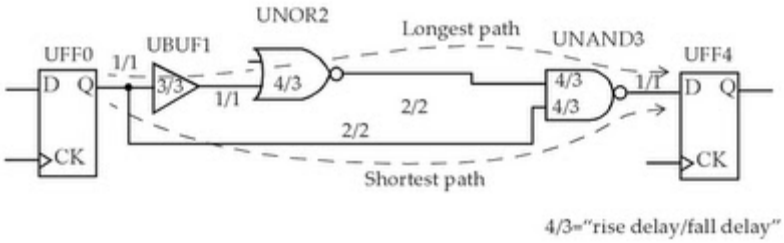
Same clock path behave like Capture and Launch path for different Data path.

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.

Longest and Shortest Path:

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.

++

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

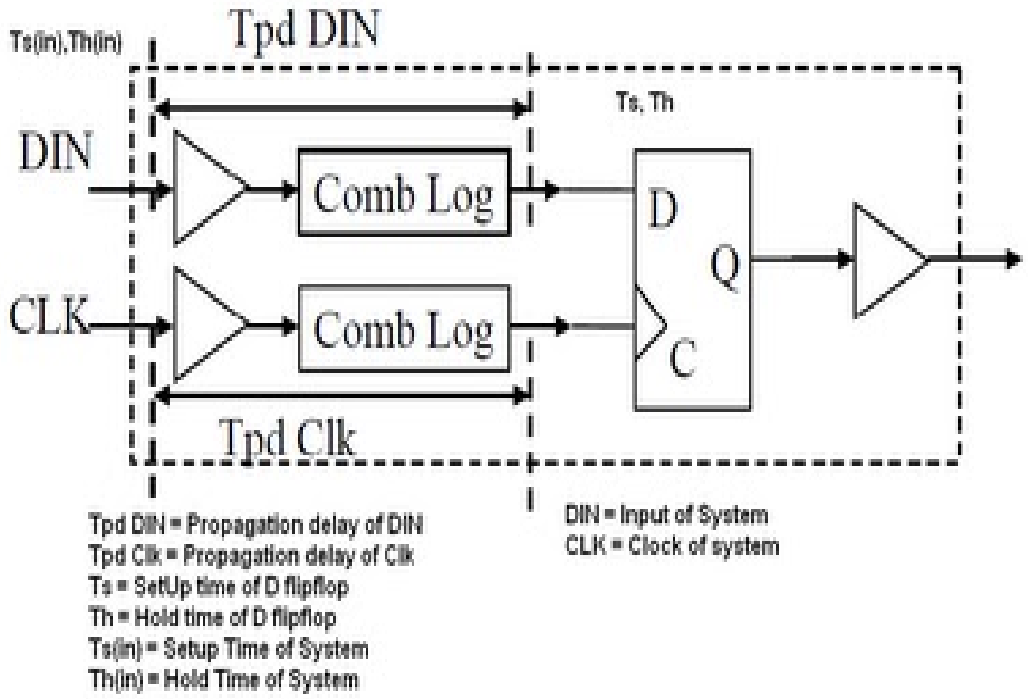
1. They know the definition but don't know the origin or say concept behind Setup and Hold timing.
2. They know the formula for calculating setup and hold violation but don't know how this formula come in picture.

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.



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.

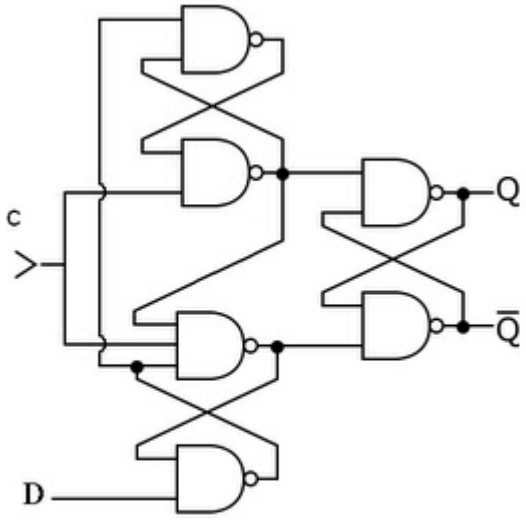
- Worst case and best case (Max delay and min delay)
  - Because of environment condition or because of PVT, we can do this analysis for the worst case (max delay) and best case (min delay) also.
- Shortest Path or Longest path (Min Delay and Max delay)
  - If combinational logic has multiple paths,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

For example 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

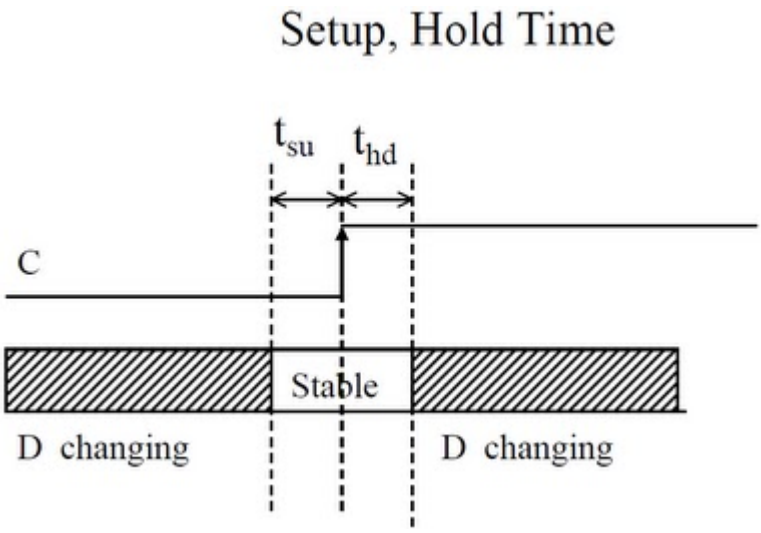
There are different ways for making the D flip flop. Like by JK flipflop, master slave flipflop, Using 2 D type latches etc. Since the internal circuitry is different for each type of Flipflop, the Setup and Hold time is different for every Flipflop.

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.

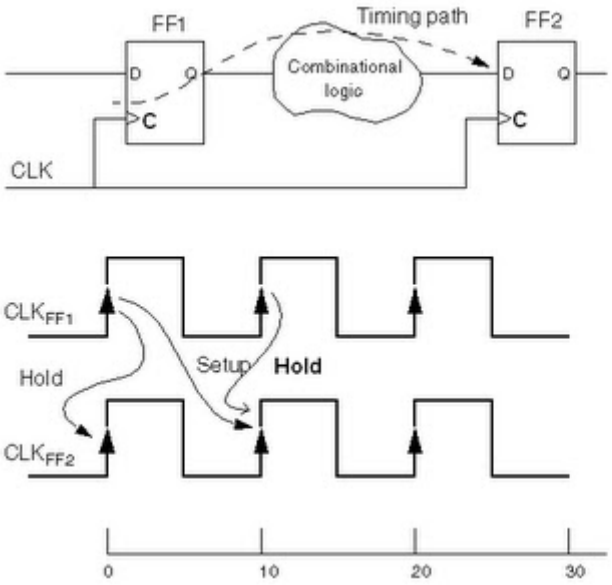


Setup and Hold Violation:

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.

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.

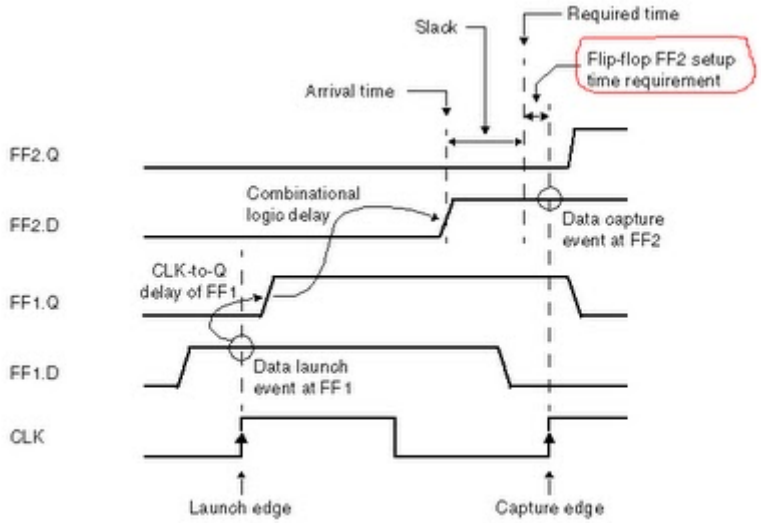
For **Hold Analysis at FF2**, Data should be stable "Th" time after the positive edge at FF2/C. Where "Th" is the Hold time of FF2. Means there should not be any change in the Input data at FF2/D between positive edge of clock at FF2 at Time=10ns and Time=10ns+Th.

- 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.*

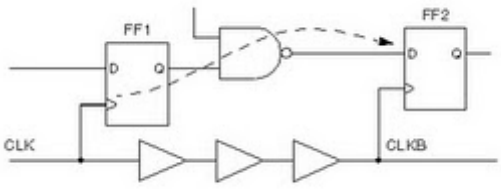
Setup Check timing can be more clear for the above Flip-flop combination with the help of following explanation.

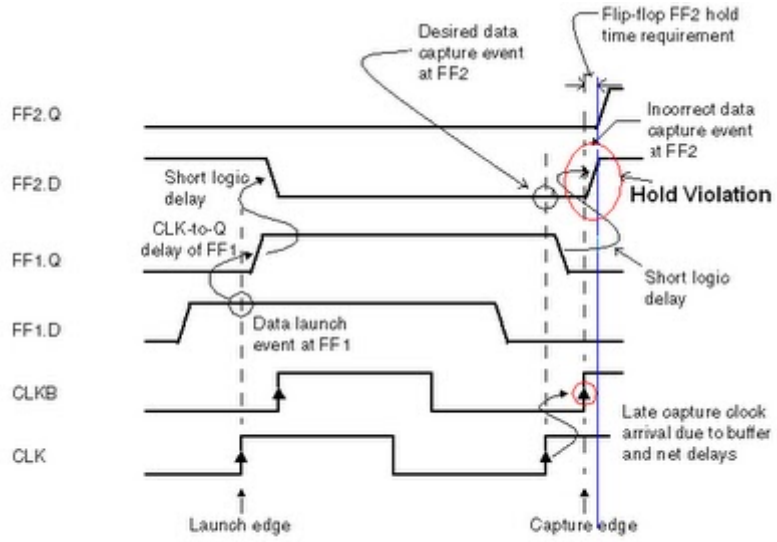


Setup Check Timing

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 + Conmintional 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.





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.
