STATIC TIMING ANALYSIS

- Hema Naga Vaishnavi

CONTENTS

CHAPTER 1 INTRODUCTION

This chapter focuses on Static timing analysis, its procedures for nanometer designs.

1.1 Nanometer designs

As the technology node**[1]** is shrinking the effect of interconnect parasitics affect the performance of design in nanometer**[2]** or deep submicron designs**[3]**.

There are many factors that affect the performance of a design which we shall discuss in upcoming chapters.

Note [1]: Technology node refers to the physical size of the transistor. Moore's law states that for every 18 months the density of the die should be double the previous technology node's die density. This is possible by reducing the gate length of a MOSFET half the dimension with respect to the previous node so the number of transistors on a die doubles. So if we are using a technology node of 0.25μ it means that the channel length of the CMOS is 0.25μm.

Note [2] & [3]: Deep submicron refers to process technologies with feature size 0.25μm or lower. Nanometer refers to process technologies with feature size 0.1μm.

1.2 Static timing analysis and Statistical static timing analysis

This is the method to verify the timing in our design. If a data is sent to a register the time it takes to reach the final output has to reach in time before another input data may reach its output. No input vectors are given in static timing analysis. All the timing is done statistically ie. considering fixed delay in the complete design.

The functional verification is done by giving set input vectors to the design and checking if the output is reached correctly for the respective input. This is carried out by the front end team. This verifies only on the stimulus paths so this is not exhaustive but exhaustive only due to the input vectors that are provided.

The static timing analysis does timing checks on all the paths of the design and scenarios thus STA is exhaustive.

The static timing analysis has been deterministic as we have considered fixed delays on all the timing arcs considering the variations due to interconnect models and the process.

Statistical static timing analysis (SSTA) predicts the true operating frequency. SSTA has reduced and eliminated the need for broad guard - banding**[4]**. A statistical timer may properly identify and optimise the sensitivities with respect to variations, thus allowing statistical optimization methods.

Note [4]: Guard - banding refers to making sure that one part of the design fails and still the die continues to operate correctly.

1.2.1 SSTA vs STA

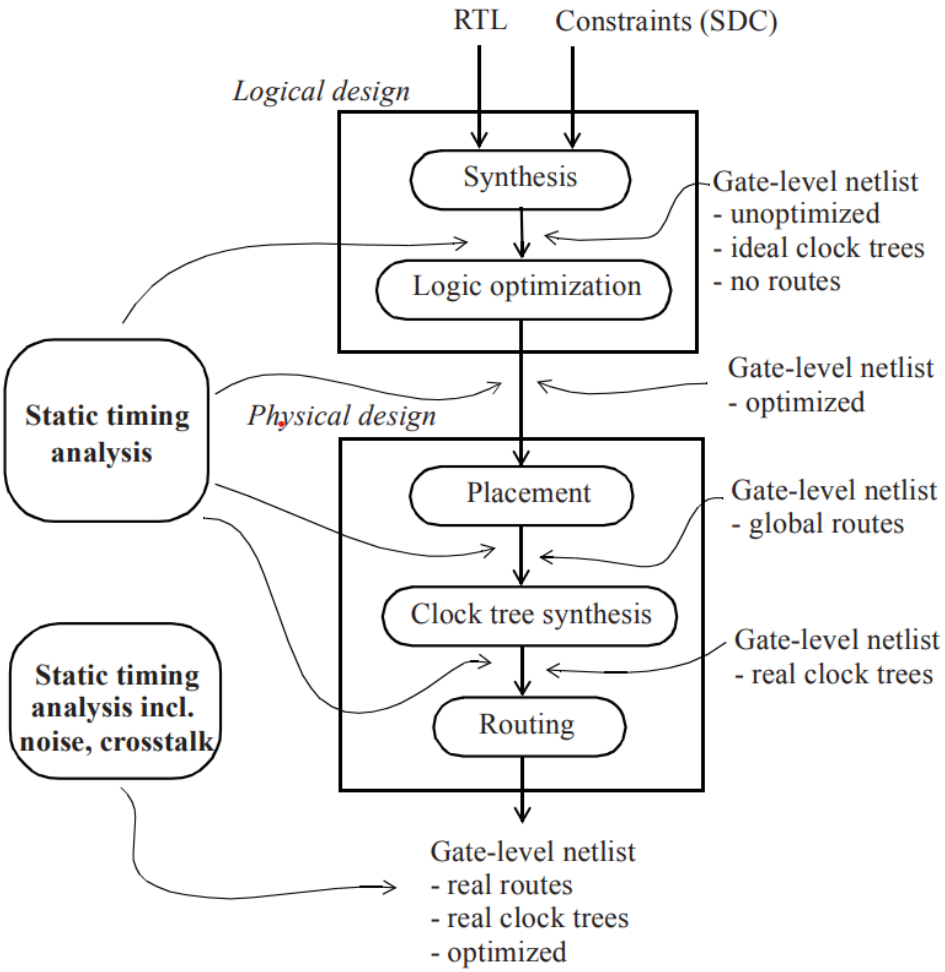
Many critical issues which cannot be addressed using STA can be resolved using SSTA tools. But there are few parameters which let SSTA not be used fully due to difficulty in the current process which may exploit the SSTA process. SSTA can detect where the slack is negative whereas the traditional STA does not.

The traditional STA does not provide accurate results of design slack**[5]**. In traditional STA each slack is just a number so design slack is equal to critical slack. In SSTA the slacks are probability distributions and min operations on these distributions are performed to obtain another probability distribution. As the slacks do not correlate accurately the min distribution is less than the min slack distribution in this kind of case STA tends to provide optimistic results.

Note [5]: Design slack refers to the minimum of the overall paths.

1.2.2 Observations during deriving the slacks with STA and SSTA

1. The slack using SSTA has oscillating curves which mean that the critical paths computed using SSTA and STA are different.
2. The 3σ slacks obtained using SSTA are larger than obtained from STA, this means that the SSTA reduces the level of pessimism.
3. For smaller number of paths the SSTA produces the small 3σ slack.



CMOS digital design flow

1.2.3 STA vs DTA

The simulation of dynamic timing analysis(DTA) the cell delays, net delays are back annotated**[6]** to netlist. This method is precise but takes a long time for simulation which DTA can not handle. DTA uses input vectors to check the functionality if the inputs are not proper then DTA simulation will take longer time.
To overcome the drawback of DTA, Static timing analysis(STA) has been developed which reduces the simulation time and does not use any input vectors. It calculates the cell and path delay and computes the maximum delay and minimum delay.

Note[6]: Back annotation: The propagation delays of cells and paths in netlist are overridden by delay values specified in SDF file. This process

of putting delays from a given source for cells in netlist during netlist simulation is called back annotation.

1.3 Design flow

The STA is rarely carried out after RTL design because here the main focus is on the functional verification as still many blocks will be in behavioural model. Once the design is synthesised then the design will be ready for performing timing analysis.

Initially the STA is done by considering 0 clock, 0 interconnect effect as we do not give any clock signal till clock tree is built and routing is not done at initial phase so zero interconnect delay is considered.

STA is done at every stage of physical design flow by considering worst case analysis.

Once the clock tree is built the STA is carried again by considering the delay. When the physical connection between cell ie. interconnect is built then the effect of RC, crosstalk noise and timing analysis is done.

1.4 STA along the design flow

At gate level (after synthesis when no physical connection is done)

1. With 0 interconnect parasitic by considering wire load model.
2. With an ideal clock.

During physical design phase

1. With routing estimation using global route congestion analysis.
2. When the real clock is applied.
3. With and without the effect of crosstalk.

1.5 Limitations of STA

There are some aspects of STA which cannot be captured and verified.

1. **Reset sequence:** STA cannot check if the reset pin has regained its logical status.

2. **X-handling:** STA can handle inputs logic 0 and logic 1. If the input is undetermined then it becomes hard for STA to analyse.
3. **PLL settings:** There may be some errors in PLL clock generations which STA cannot handle.
4. **Asynchronous clock domain crossings:** The signal in digital circuits travel from one clock domain to another and may cause metastability**[7]** so to overcome this, correct clock synchronizers should be present in asynchronous clock domain crossing paths.
5. **IO interface timing:** It may not be possible to specify the IO interface requirements in terms of STA.
6. **Interface between analog and digital blocks:** STA does not deal with analog blocks**[8]** so verification should be carefully done to ensure there is correct connectivity between analog and digital blocks.
7. **False paths:** When a false path is declared the timing on that path is completely removed and if any delay is propagating due to that path the STA cannot analyse and this leads to failure.
8. **FIFO pointers out of synchronisation:** STA cannot analyse if two FSMs are out of synchronous.
9. **Clock synchronisation logic:** STA assumes that the generated clock provides the same waveform as of the master clock if this doesn't happen then STA cannot handle such a situation.
10. **Functional behaviours across clock cycles**: STA cannot analyse signals that change its functional behaviour across clock cycles.

Note[7] : Metastability refers to the undefined state of an input signal.
Note[8]: STA is not applicable for analog blocks because the analog circuits do not have synchronising clock signals so we cannot test on setup and hold time requirements. In digital circuits we have a synchronous clock circuit which forms a reference signal to analyse setup and hold checks where the analog circuits do not have the reference signal so STA limits itself in analog circuits.

1.6 Power and area considerations

The timing,area and power go hand in hand. A designer should meet timing with minimum power consumption & dissipation and minimum area occupation in a chip.

CHAPTER 2 STA CONCEPTS

2.1 Standard cells

All the functionality in a design is made using the basic building blocks such as nand, nor, inverters, flip flops which are further built using the CMOS structure are known as standard cells.

2.2 Modelling CMOS

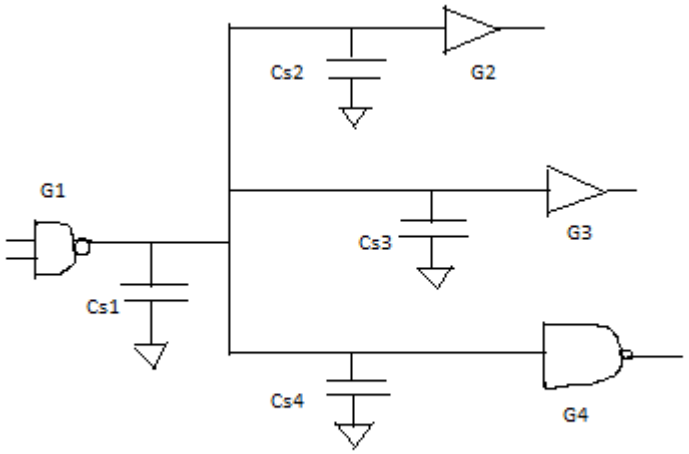
The CMOS structure has pull up and pull down circuits ie. When the input is high then the CMOS discharges due to the ON state of pull down structure and when the input is low the CMOS charges due to the ON state of pull up structure. CMOS sees resistance effects during charging and discharging.

If the CMOS has high output drive then charging of CMOS will be faster due to less resistance and if the CMOS has low output drive then charging of CMOS will be slow due to more resistance.

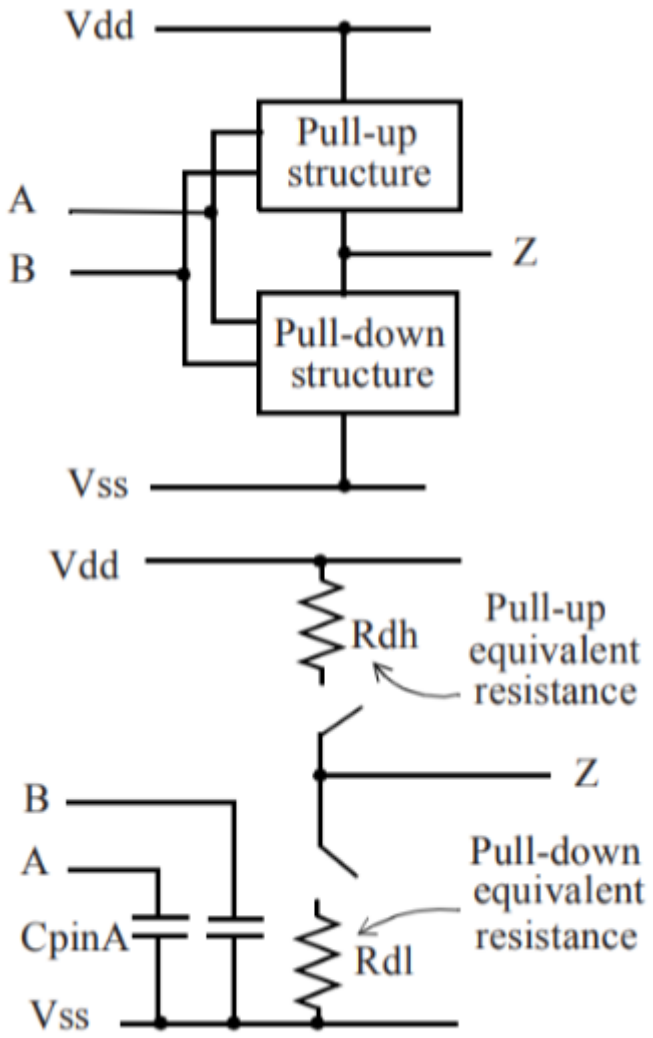
The larger the drive the more the area thus reducing the resistance of CMOS.

CMOS sees capacitive load because with varying input voltage the depletion region at the drain varies creating a capacitive effect between drain and substrate which is seen as output load for a CMOS.

When a cell has multiple fanouts then the total load will be the sum of all the capacitances on all the fanouts of that cell.



Capacitance on a net

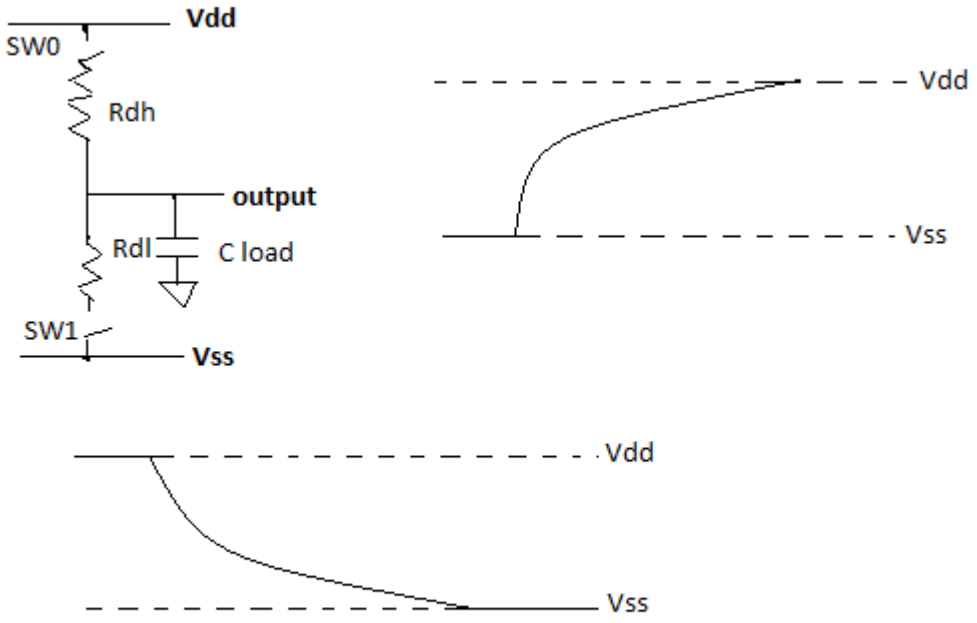


Pull up & Pull down Resistance

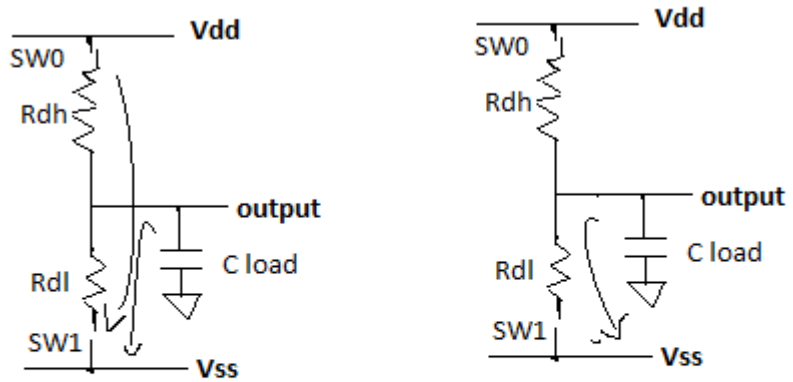
2.2.1 Switching waveform of CMOS

When we apply low input to the device then the capacitor will be charged and output sees a rise waveform and for high input the capacitor discharges and output sees a fall waveform. But the switching of input from low to high and vice versa cannot change instantaneously there will be a short span where both NMOS and PMOS will be ON and in this case the switching waveform sees an exponential rise or fall due to RC effect.
The voltage transition is given as

$$V = Vdd * [1 - e^{-t/(Rdh * Cload)}]$$



RC Charging and Discharging Waveforms



a) Cell is switching,
(Pull-up,Pull-down both on)

b) Cell is discharging to logic-0
(Pull-up off, Pull -down on)

Current flow for a CMOS cell output stage

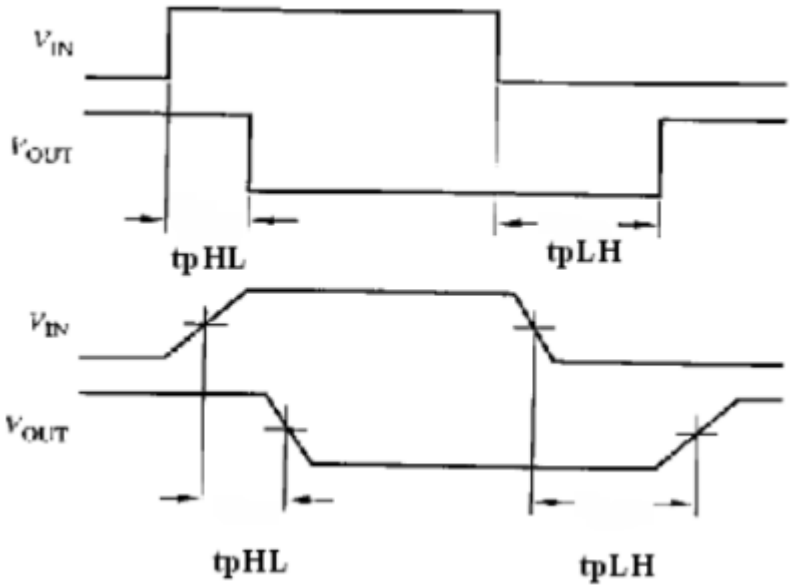
2.3 Propagation delay

The time required by the signal to propagate from input to output. The range of clock is considered with respect to 50% of input and 50% of output voltage transitions.

When a logic 0 is applied to the CMOS then the load capacitor starts charging. And when logic 1 is applied to the CMOS then the load capacitor starts discharging. This charging and discharging time of a capacitor is considered as propagation delay.

Considering 1st order Rc network the time taken by CL to charge to 50% of its final voltage is given as:
VOUT = VIN(1-e(-t/τ))
VOUT = VIN/2
VIN/2 = VIN(1-e(-t/τ))
½ - 1 = -e(-t/τ) ; Let τ = RC
Ln(1/2) = -t/RC
T = 0.69RC

The amount of time required for the load capacitor to fully charge by the CMOS is 5τ(5RC).



2.3.1 Slew of waveform

In timing analysis the rate of change from rise to fall or from fall to rise is called as slew. The change from fall to rise or rise to fall is called transition.

The transition is time inverse of slew rate. If slew is fast then the transition is less and vice versa.

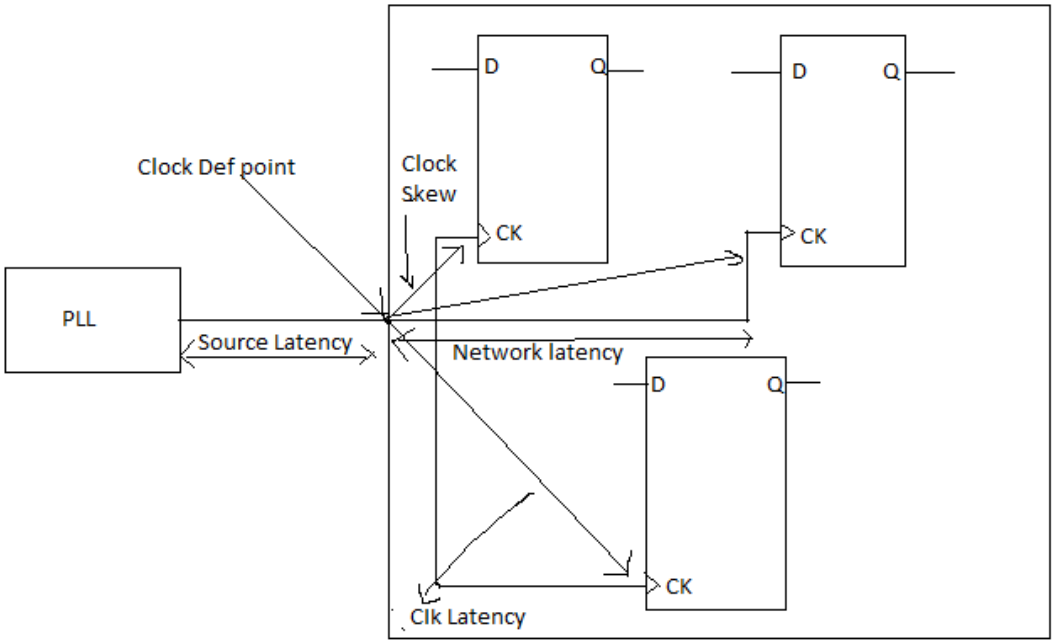
Example: The slew rate settings can be done by
Slew_lower_threshold_pct_fall 30;
Slew_upper_threshold_pct_fall 70;
Slew_lower_threshold_pct_rise 30;
Slew_upper_threshold_pct_rise 70;

These 30 and 70 are the percent of VDD. The threshold values specify that the falling slew difference between falling edges reaches 70% and 30% of VDD. Similarly the threshold values specify that the rising slew difference between rising edges reaches 70% and 30% of VDD.

The threshold value can even be 20% - 80% or 10% - 90%.

2.3.2 Skew between signals

Skew is the difference in arrival times of two or more signals. Clock skew refers to the difference in arrival times of clock signals at the end points of registers. When a clock tree has multiple end points and skew value is constrained where the value informs that the difference between longest clock path and shortest clock path is the skew.
Example: A clock tree with 100 end points has a skew of 10ps which means the difference between shortest clock path and longest clock path is 10ps.



During logical STA we consider the clock to be ideal ie. There is no delay between the cells. An ideal clock tree has 0 skew but when the real clock tree comes into picture zero skew is not possible because having zero skew means that all the cells are switching at the same time this leads to more power consumption.

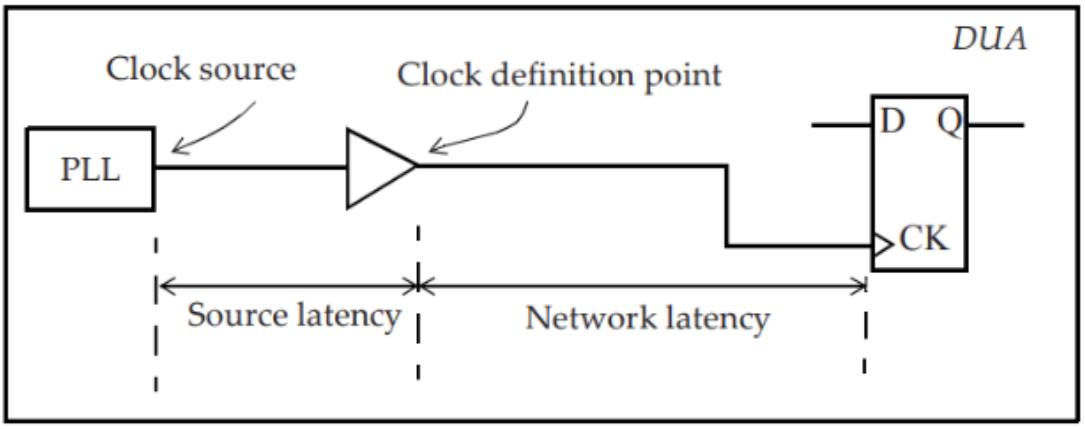
During logical STA as we do not know how much skew is present between clock signals we estimate some percentage of skew and consider it as uncertain value and constraint.

Types of skew:

a. **Global skew:** The skew seen between two non - communicating registers.
b. **Local skew:** The skew seen between two communicating registers.
c. **Positive skew:** This occurs when the capture clock is late with respect to the launch clock.
d. **Negative skew:** This occurs when the launch clock is late with respect to the capture clock.
e. **Useful skew:** This utilises the clock skew to meet the timing.

2.3.3 Clock Latency

Clock latency defines the total time it takes for a clock to reach from its start point to end point. As we have some delay in clock insert delay on the cell it is also called insertion delay.



☐ **There are two components of latency:**

1. **Source latency**: The time a clock signal takes to reach the clock definition point from clock source point.
2. **Network latency:** The time a clock signal takes to reach the register clock pin from clock definition point.

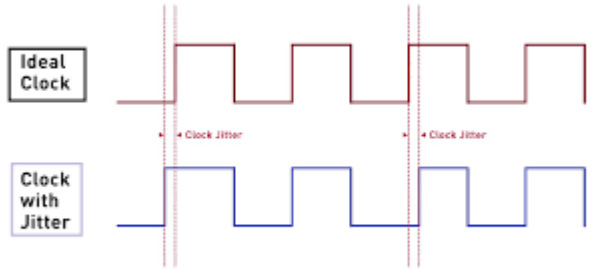
2.3.4 Jitter

The clock to a block is from the PLL(phase locked loop) which generates clock signals for the entire chip. There may be some errors in the clock signal ie. let's say the rise edge of my chip should be at 10ns but due to some errors we may have ±x value ie. I might get my signal rising at 9ns or 11ns which is an error that affects setup and hold requirements. If the transition of a signal is bad then the minimum pulse width is violated and adds jitter to the signal.

As this variation in clock signal is unknown so an estimated value is constrained as a factor of uncertainty.

☐ **There are two components of jitter:**

1. **Random:** It is unbounded and hard to analyse.
2. **Deterministic:** It is periodic and bounded and can be analysed.



2.3.5 Margin

To maintain extra pessimism the designer can add some delay value to the design as a component of uncertainty.

2.3.6 Interconnect

The cells when connected with metal wires exhibit RC effect due to which the signal may have some delay and this cannot be estimated till the physical connections are done in our design. So till the connection is made the STA does estimated values of RC as a component of uncertainty.

2.4 Timing arcs and timing paths

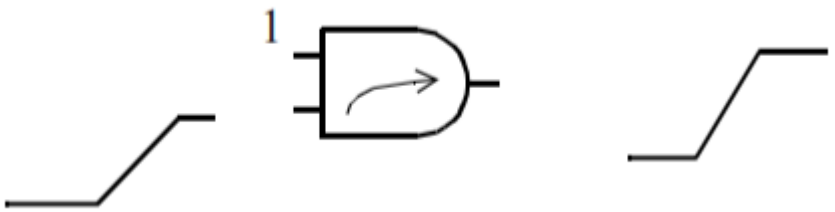
2.4.1 Timing arcs

Every cell exhibits a timing arc ie. with a given input transition how the output transition varies.

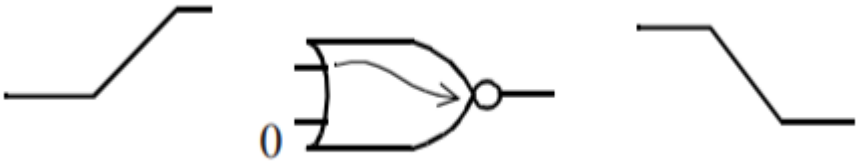
☐ **There are three types of unate:**

1. **Positive unate:** The input rise transition leads to output rise transition.

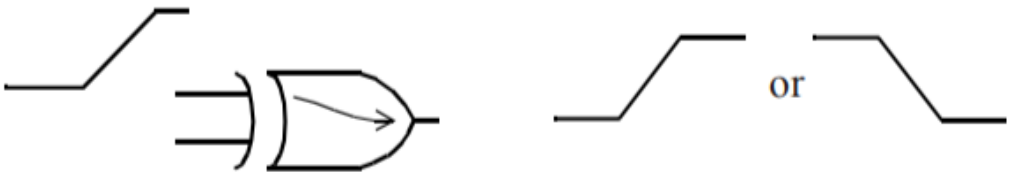
2. **Negative unate:** The input rise transition leads to output fall transition.
3. **Non-unate:** The output's rise or fall transition is not dependent on input's rise or fall transition.



Positive Unate Arc



Negative Unate Arc



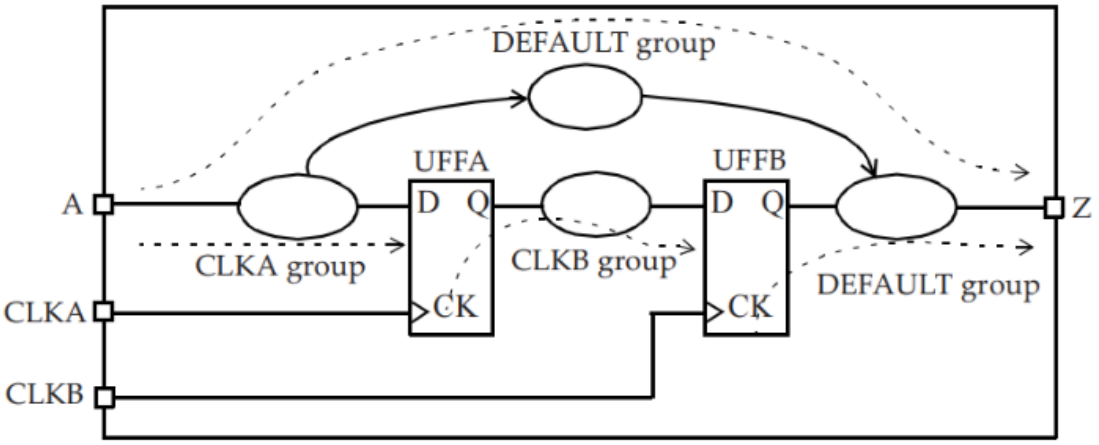
Non-Unate Arc

2.4.2 Timing paths

The timing analysis is done from a start point to an end point; these start and end points can be categorised into four types.

The different timing paths are:

1. Input to output path
2. Input to Register path
3. Register to Register path
4. Register to output path



2.5 Clock domains

The design can have one or more clock domains but it depends on the data path if these clock domains are communicating or not. If the data path starts at one clock domain and ends at another clock domain then the designer sets one clock domain as real and other as false path and no constraint is done on that path.

When different clock domains are considered and analysed it helps designers to validate the real timing paths.

2.6 Operating conditions

The operating conditions are a combination of process, voltage and temperature.
The STA is done considering the operating conditions.

As a chip has to work in all climatic conditions, voltage variations; any external changes should not affect the chip performance so STA is considering the changes in process, voltage and temperature considering worst case delay and if the timing is met in all the conditions then the chip is ready to work in any climatic condition.

☐ **Process variation:**

During fabrication there may occur some variations in etching, gate oxide thickness and doping concentration variation which leads to slow/fast/typical process of CMOS.

1. Slow process: Delay is more
2. Fast process: Delay is less
3. Typical process: Delay is nominal

☐ **Voltage:**

During fabrication due to gate oxide thickness, doping concentration difference there be threshold variation of CMOS.

1. Hvt: High threshold voltage which means the delay is more and power is less.
2. Lvt: Low threshold voltage which means the delay is less and power is more.
3. Svt/Rvt: Standard threshold voltage which means the delay and power is nominal.

☐ **Temperature:**

a) Higher technology nodes:

With increase in temperature the mobility of carriers decreases as collisions become less the current flow across the channel decreases thus increasing delay.
With decrease in temperature the mobility of carriers increases as bond breaks and increases the collisions there by increasing the current flow across the channel thus having less delay.

b) Lower technology nodes:

In a lower technology node, as temperature increases the threshold voltage decreases so overdrive voltage**[1]** and drain current increase which leads to decrease in cell delay. Here overdrive voltage is dominating over the mobility factor. But in higher technology nodes, overdrive voltage is not much dominating, and delay of the cell varies as

per variation in carrier mobility and as temperature increases mobility decreases and so drain current decreases which lead to increase in cell delay.

Note[1]: Overdrive voltage is Vgs - Vt; In lower nodes the Vgs value is almost near to Vt so slight variation in Vt impacts overdrive voltage thus causing Id to increase even with decrease in mobility as Id is directly proportional to square times of overdrive voltage. The impact of overdrive voltage is dominant over decrease in mobility due to temperature.

CHAPTER 3 Synchronous clock and Asynchronous clock timing analysis.

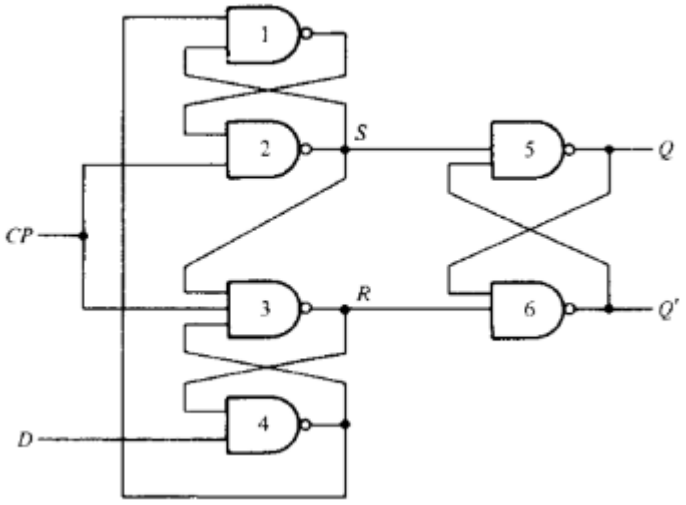
As we are talking about the timing analysis of our design. It deals with how fast or slow our data arrives at output for a given input.

So to analyse this timing we need to check how much the input should be stable upon giving a clock so that it does not affect the output data. We have setup and hold analysis to ensure that our output data is safe enough with respect to given input in synchronous circuits as we have a common clock to all the registers.

We have recovery and removal analysis to ensure that my reset pin is reset after and before data is given as there is no common clock to all the registers in asynchronous circuits.

3.1 Setup and hold analysis

Let's consider a D flip flop circuit.



Case 1: cp = 0; D = 0;
The FF-4 and FF-3 see output 1, FF-2 see output 1 and FF-1 see output 0.
Thus SR = 11.

Case 2: cp = 0; D =1 ;
The FF- 4 sees output 0, FF-3 sees output 1, FF-2 sees output 1 and FF-1 sees output 1.
Thus SR = 11

∴ When clock is 0 my data is held constant.

Case 3: cp = 1; D = 0;
The FF-4 sees output 1, FF-3 sees output 0, FF-2 sees output 1, FF-1 sees output 0.
Thus SR = 10

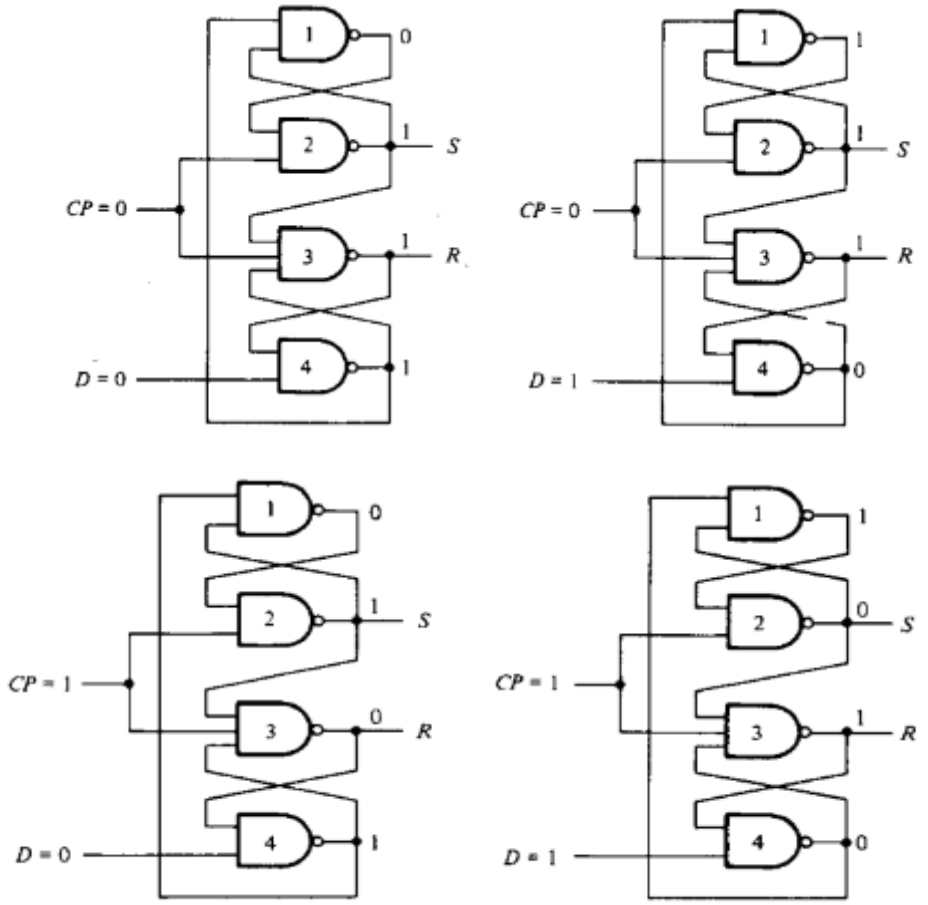
Case 4: cp = 1; D = 1;
The FF-4 sees output 1, FF-3 sees output 0, FF-2 sees output 1, FF-1 sees output 0.
Thus SR = 10

So with this 4 case analysis we can tell that when my D pin input is stable before the clock edge rises so that my previous input's output is reached at SR which is considered as setup time and upon changing the clock the output should not be changed with the new input so the amount of time the previous input to be stable after next clock so that the previous input's output is reached is considered as hold.

So with this analysis for the data to reach before clock next edge FF-4 and FF-1 should be calculated in setup analysis ie. FF-4 and FF-1 should be stable. For the data to be stable after reaching the output upon giving the clock edge the FF-2 or FF-3 has to be calculated for hold analysis ie. FF-2 or FF-3 should be stable.

Setup: The amount of time input should be stable before the arrival of a clock edge.
Hold: The amount of time the input is stable after the arrival of the clock edge.

A setup value and hold value can individually be negative ie. Even if setup comes after the clock edge the timing is met and similarly if hold comes before the clock edge and still meet timing but setup and hold both together can not be negative if so then it will be timing violation. If setup is negative and hold is positive the hold can compensate with setup and meet with timing. So the addition of setup and hold value should be a positive value.



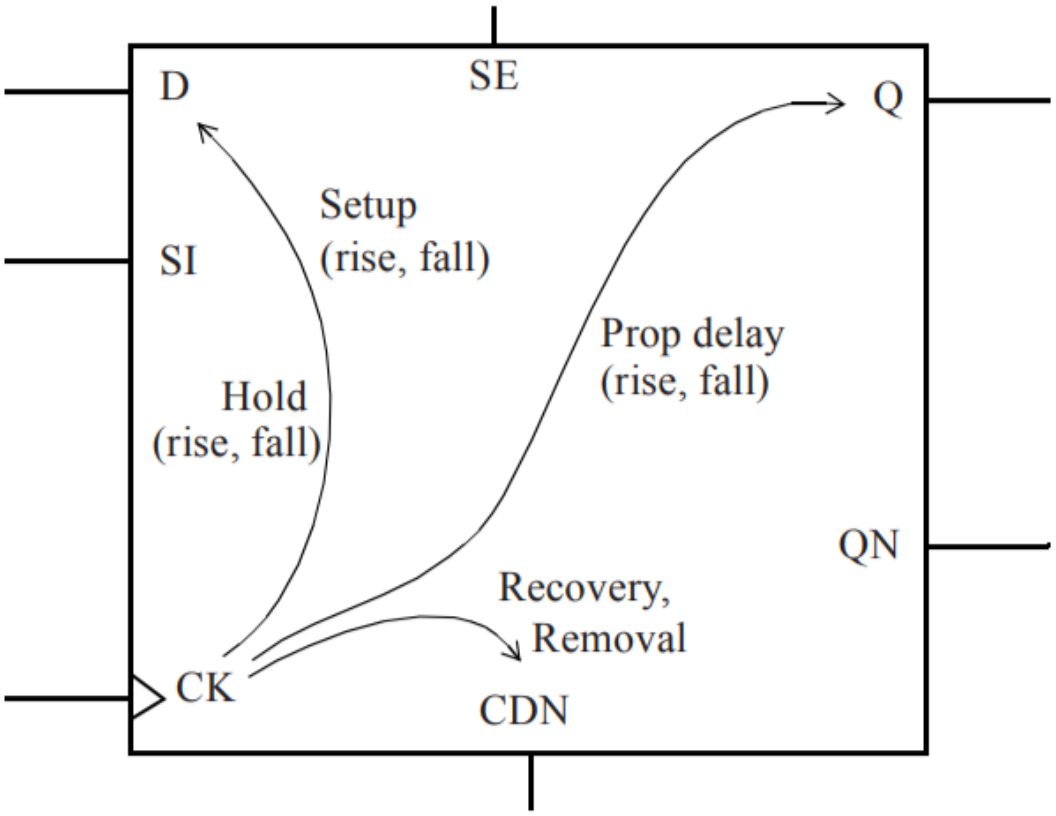
3.2 Recovery and removal analysis

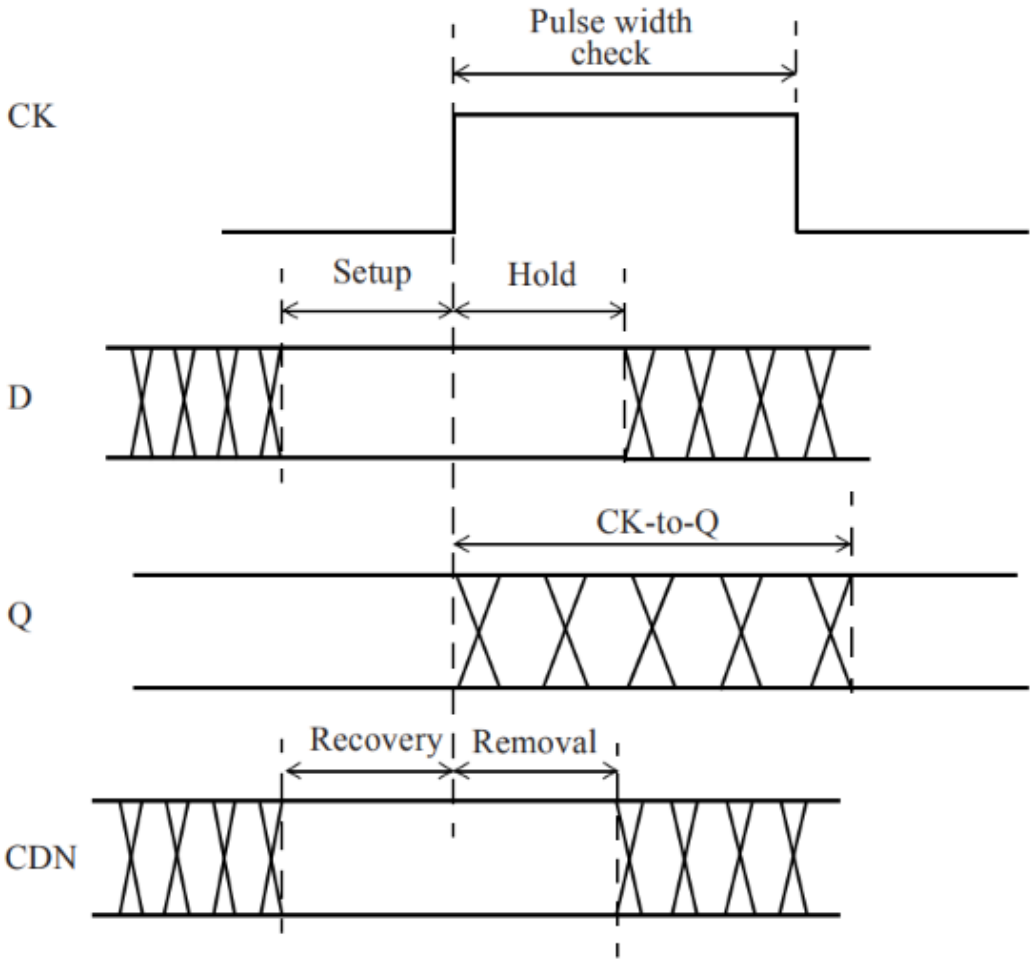
A removal timing check ensures that there is adequate time between an active clock edge and the release of an asynchronous control signal. The check ensures that the active clock edge has no effect because the asynchronous control signal remains active until removal time after the active clock edge.

A recovery timing check ensures that there is a minimum amount of time between the asynchronous signal becoming inactive and the next active clock edge.

Removal timing analysis: The asynchronous control signal is released (becomes inactive) well after the active clock edge so that the clock edge can have no effect

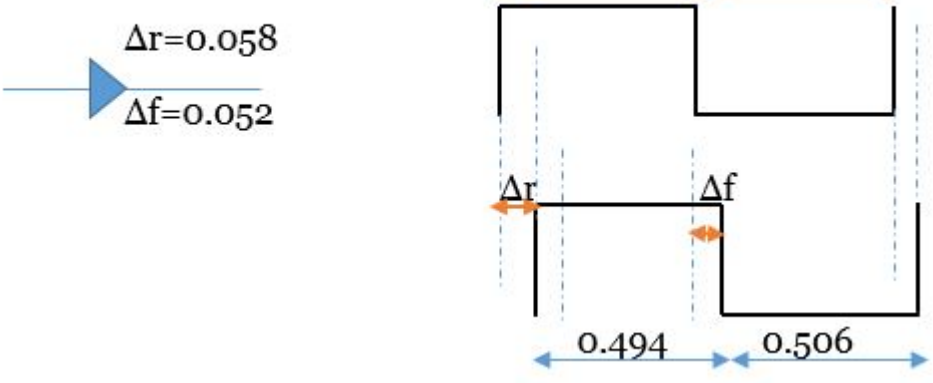
Recovery timing analysis: This check ensures that after the asynchronous signal becomes inactive, there is adequate time to recover so that the next active clock edge can be effective.





3.3 Pulse width checks

There is a check to ensure that minimum pulse width at input is maintained or not. If the min pulse width is violated then the clock may not latch data properly. We use a clock buffer to overcome this pulse width violation.

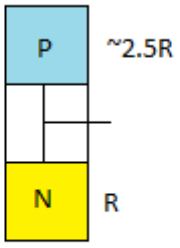


3.4 Buffers and inverters

3.4.1 Normal buffers

We have buffers in our design to improve the transition time of a signal The internal circuitry of a buffer is built with two different sized inverters.

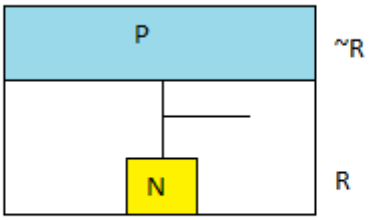
The first inverter has low drive strength and small size having another inverter as its load with high size and high drive strength thereby improving the transition of a signal.



Normal Buffer

3.4.2 Clock buffers

We have clock buffers which are placed in the clock path so that the cells have equal rise and fall time. This symmetry is obtained by sizing the PMOS of our inverter by 2.5 times the size of the NMOS so that the resistance of PMOS is reduced and gives the same transition time at the output as of the input.



Clock Buffer

3.4.3 Inverters

The inverters replace the buffers where the transition of a signal is the main concern in our design as the inverters have high drive strength and produce almost equal rise and fall transition. Two inverters can replace a buffer with less interconnect distance between the two.

CHAPTER 4 STANDARD CELL LIBRARY

Standard cell library is one of the input files for synthesis, STA and PD flow. This file contains the standard cell information like its drive strength, cell name, pin capacitance, wireload model, transition, area, functionality, library setup and hold time, load value etc;.

The information in the library is used for timing and even useful during synthesis for optimization.
In this chapter we shall learn how the standard cell library is used in timing analysis.

4.1 Pin capacitance

Pin capacitance is used to calculate total capacitive load at each node in the circuit. Every input and output pin of a cell has pin capacitance. In standard libraries the output pin capacitance is taken as 0 and only input pin capacitance is considered.

```
pin (INP1) {
  capacitance: 0.5;
  rise_capacitance: 0.5;
  rise_capacitance_range: (0.48, 0.52);
  fall_capacitance: 0.45;
  fall_capacitance_range: (0.435, 0.46);
  . . .
}
```

The units of resistance, voltage, capacitance are mentioned at the very starting of a standard cell library.

4.2 Timing model

The delay of a cell is defined by the input transition and output load. Output load is considered as capacitance ie. the time taken by the load to charge and discharge decides the delay of a cell.
The input transition of the cell is a slew of a signal from rise to fall or from fall to rise.

If output load is large the charging and discharging time increases and input transition of a cell is more and thus the delay is more. If output load is small the charging and discharging time decreases and input transition of a cell is less and thus the delay is less.
Thus with output load the input transition also changes.

4.2.1 Linear timing model

The cell delay depends on input transition and output load are represented as linear functions of a cell. This model does not give accurate delay values in submicron technologies.

4.2.2 Non linear delay model (NLDM)

The cell delay is based on input transition and output load so the NLDM considers a look up table of the input transitions and output load of a cell as index values and their combinations are characterised in a table.

The image below shows the index_1 input transition values and index_2 shows the output load.

```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*  0.16      0.35     1.43 */ \
        /* 0.1 */   "0.0513,  0.1537,  0.5280", \
        /* 0.3 */   "0.1018,  0.2327,  0.6476", \
        /* 0.7 */   "0.1334,  0.2973,  0.7252");
      }
    cell_fall(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*  0.16      0.35      1.43 */ \
        /* 0.1 */   "0.0617,  0.1537,   0.5280", \
        /* 0.3 */   "0.0918,  0.2027,   0.5676", \
        /* 0.7 */   "0.1034,  0.2273,   0.6452");
    }
```

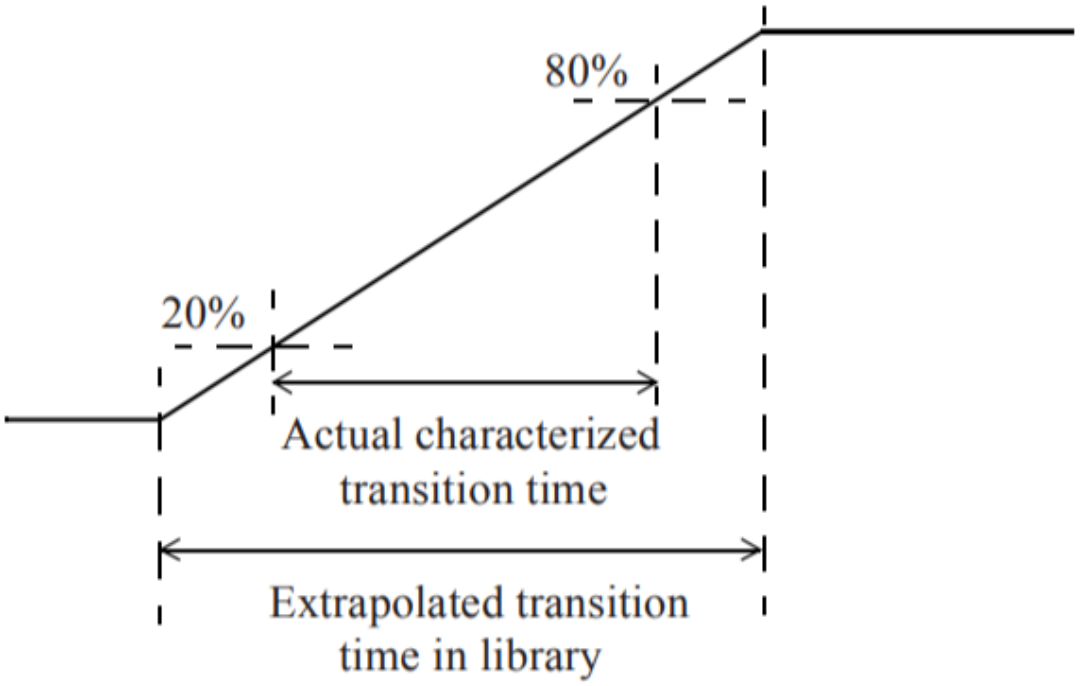
Since non-linear variations of delay are calculated based on transition and load it is called a non-linear delay model which is a 2-dimensional representation of delay models.

4.2.3 Slew deration

The library is characterised with a slew derate value based on the lower and upper threshold transition value. In our design the transition cannot be as exact as specified by the library there will be marginal variations in the slew so the library is characterised with slew derate value calculated based on the defined slew threshold values.

```
/* Threshold definitions */
slew_lower_threshold_pct_fall : 30.0;
slew_upper_threshold_pct_fall : 70.0;
slew_lower_threshold_pct_rise : 30.0;
slew_upper_threshold_pct_rise : 70.0;
input_threshold_pct_fall : 50.0;
input_threshold_pct_rise : 50.0;
output_threshold_pct_fall : 50.0;
output_threshold_pct_rise : 50.0;
slew_derate_from_library : 0.5;
```

Most of the previous generations have 10%-90% as the threshold limit for a slew. In our characterisation of slew if it is taken as 30%-70% while calculating the derate the reference is taken as 10%-90% as it was the previous generation's threshold.
Slew derate: (70-30)/(90-10) = 40/80 = 0.5



4.2.4 composite current source model

The NLDM model has only input transition and output load in its lookup table but as the technology node shrinks the drive value of a cell is becoming important to compute the delay of a cell and which is not possible with the NLDM model.

We have CCS model which has another index value ie. time. The time in the index refers to the time when the input waveform crosses the delay threshold. So the CCS deals with input transition, output load and time. The driver delay is modelled by the time varying and voltage dependent current source . This current source depends upon receiver pin capacitance and output charging currents so the name composite current source.

This model provides accurate delay values compared to NLDM.

4.2.5 Models for crosstalk noise

These models are described as CCSN(CCS for noise). To analyse the impact of crosstalk on signal ie. The amount of glitch that adds on the signal while propagating is obtained from the CCSN model. A two dimensional table is drawn in the library which has the DC currents and output voltage characterised to compute the delay.
When there is a crosstalk affect on a signal then the noise adds on the waveform causing glitch so the CCSN models contains the table of glitches ie.

1. Input glitch magnitude
2. Input glitch width
3. CCB[1] output net capacitance
4. Time

Models for DC margin: The DC margin refers to the largest DC variation allowed at the input pin of the cell which would keep the cell in its steady condition, that is, without causing a glitch at the output.

Models for noise immunity: The noise immunity models specify the glitch magnitude that can be allowed at an input pin. These are normally described in terms of two-dimensional tables with the glitch width and output capacitance as the two indices.

Different variations of the noise immunity models can be specified such as:

i. Noise_immunity_high
ii. Noise_immunity_low
iii. Noise_immunity_above_high (overshoot)
iv. Noise_immunity_below_low (undershoot).

Note[1]: CCB - source drain channel connected portion of a cell.

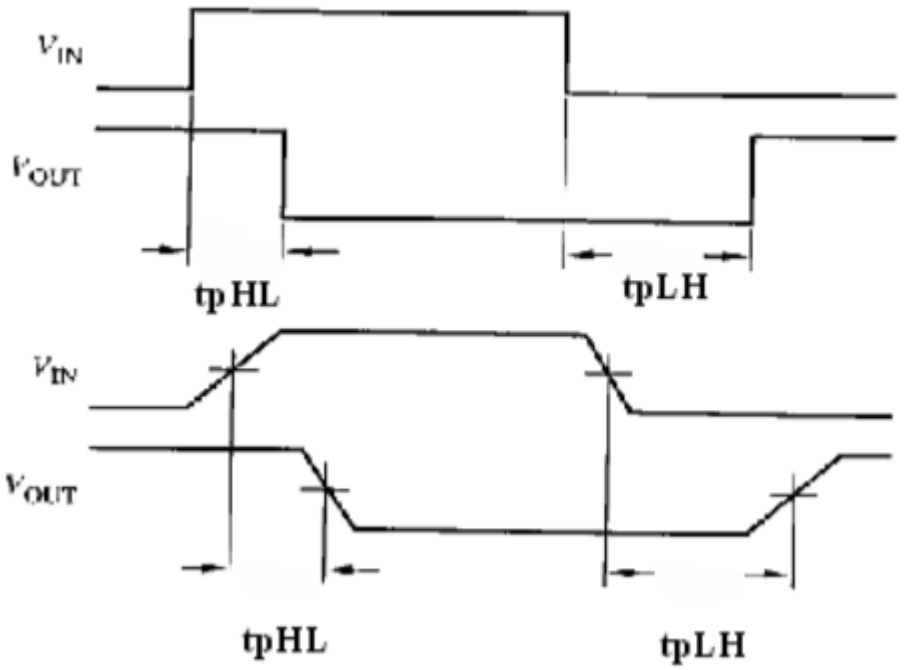
4.3 Propagation delay

The time required by the signal to propagate from input to output. The range of clock is considered with respect to 50% of input and 50% of output voltage transitions.

When a logic 0 is applied to the CMOS then the load capacitor starts charging. And when logic 1 is applied to the CMOS then the load capacitor starts discharging. This charging and discharging time of a capacitor is considered as propagation delay.

Considering 1st order Rc network the time taken by CL to charge to 50% of its final voltage is given as:
VOUT = VIN(1-e(-t/τ))
VOUT = VIN/2
VIN/2 = VIN(1-e(-t/τ))
½ - 1 = -e(-t/τ) ; Let τ = RC
Ln(1/2) = -t/RC
T = 0.69RC

The amount of time required for the load capacitor to fully charge by the CMOS is 5τ(5RC).



4.4 Power dissipation

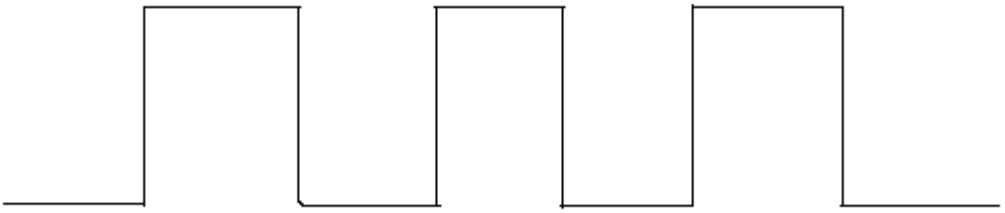
The power dissipation of a CMOS circuit is instead dominated by the dynamic dissipation resulting from charging and discharging capacitances.

4.4.1 Dynamic power dissipation(switching power):

Each time the capacitor CL gets charged through the PMOS transistor, its voltage rises from 0 to VDD, and a certain amount of energy is drawn from the power supply. Part of this energy is dissipated in the PMOS device, while the remainder is stored on the load capacitor. During the high-to-low transition, this capacitor is discharged, and the stored energy is dissipated in the NMOS transistor.

Dynamic power dissipation is seen only when PMOS is on and NMOS is off.

PPD = VCC * IL = CPD * VCC^2 * fIN



Computing the dissipation of a complex circuit is complicated by the f0 ->1 factor, also called the switching activity. Switching activity is the number of times the CMOS switches from 0 to 1 transition.

Reducing VDD has a quadratic effect on dynamic power, reducing VDD might help in reducing performance but it increases the power dissipation so when the design is power critical reducing VDD does not help. In such cases, reducing effective capacitance will improve performance and power dissipation. Reducing switching times will also reduce power dissipation but the logic of the design is given by the architectural team so having a change in effective capacitance is the main tool given in the designer's hand.

4.4.2 Direct path power dissipation(internal power dissipation):

When both NMOS and PMOS are ON we get to see direct path power dissipation as there exists a direct path from VDD to GND.

As both NMOS and PMOS are ON there exists a short circuit current and is called Ipeak.

$P_{DP} = t_{sc}.VDD.I_{peak}.f$

4.4.3 Static power dissipation (leakage power):

When both NMOS and PMOS are OFF we get to see static power dissipation.
In every device though it is in off state there exists a leakage current, the same is seen in OFF CMOS leading to static power dissipation.
There are 3 factors affecting the leakage current they are:

a. The reverse bias connection of PN junction leading to reverse leakage current.
b. Gate induced drain leakage.
c. Due to sub – threshold conduction.

Pstatic = VDD. $I_{leakage}$

4.5 Wire load model

Prior to routing there is no physical connection between the cells and the tool estimates the interconnect parasitic effect to be 0.
During logical STA the wire load model present in the standard cell library is used to estimate the RC value.
Based on the area of block and the design the tool chooses various wire load models defined in the library.
The net length increases with increase in area of block. So wire load estimates the RC with respect to fanout and length of a cell.

In the library the wire load model is written in the format :

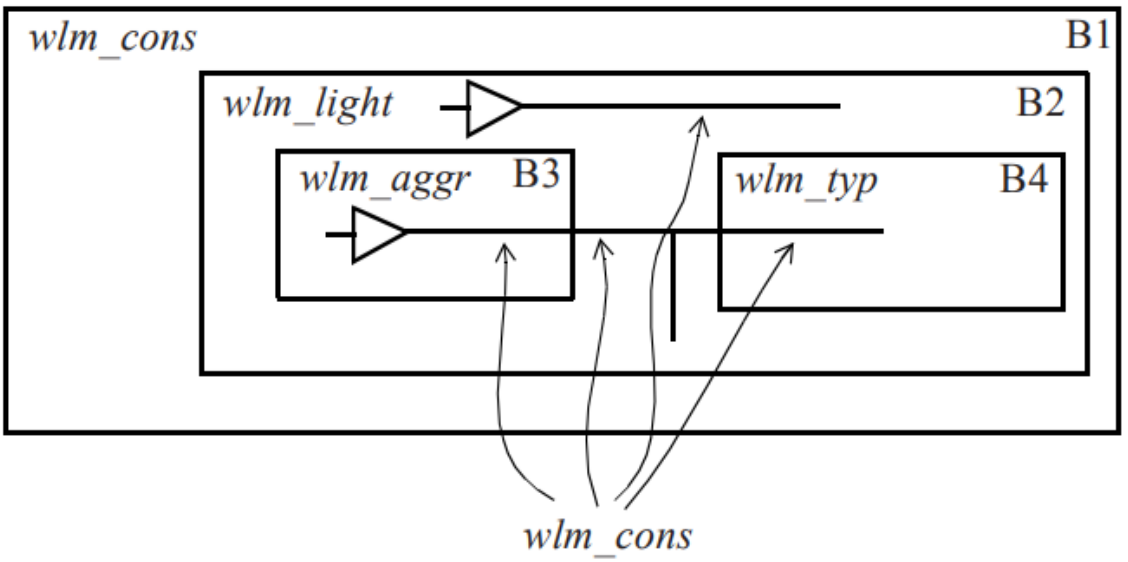
```
wireload(“wlm_conservative”) {
Resistance: 5.0;
Capacitance: 1.1;
Area: 0.05;
Slope: 0.5;
fanout(1,2.6);
fanout(2,2.9);
fanout(3,3.2);
fanout(4,3.6);
}
```

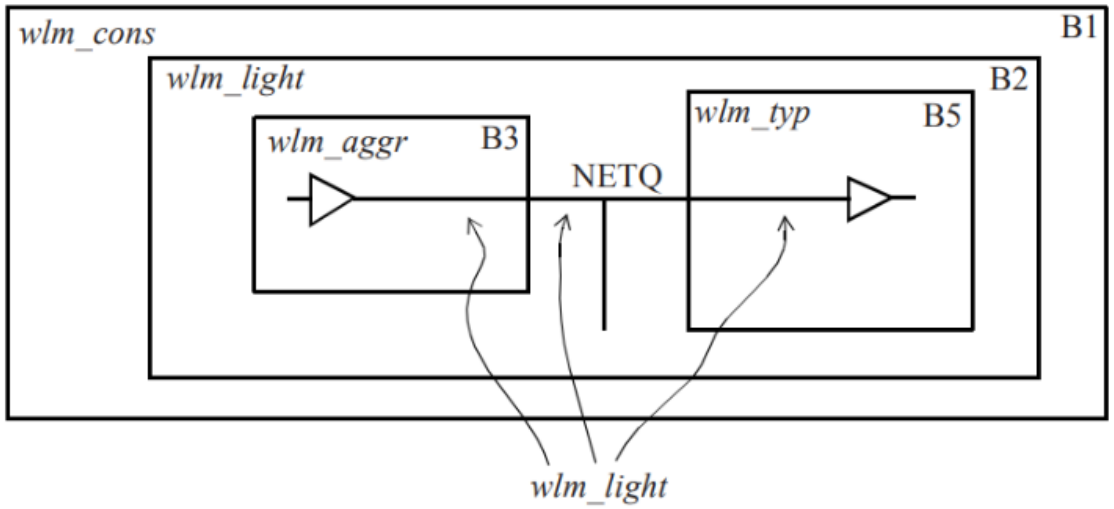
The slope is to calculate the extrapolation. The representation fanout(1,2.6) means fanout(fanout_number, fanout_length).
For a specified fanout the RC is calculated and estimated for delay analysis.

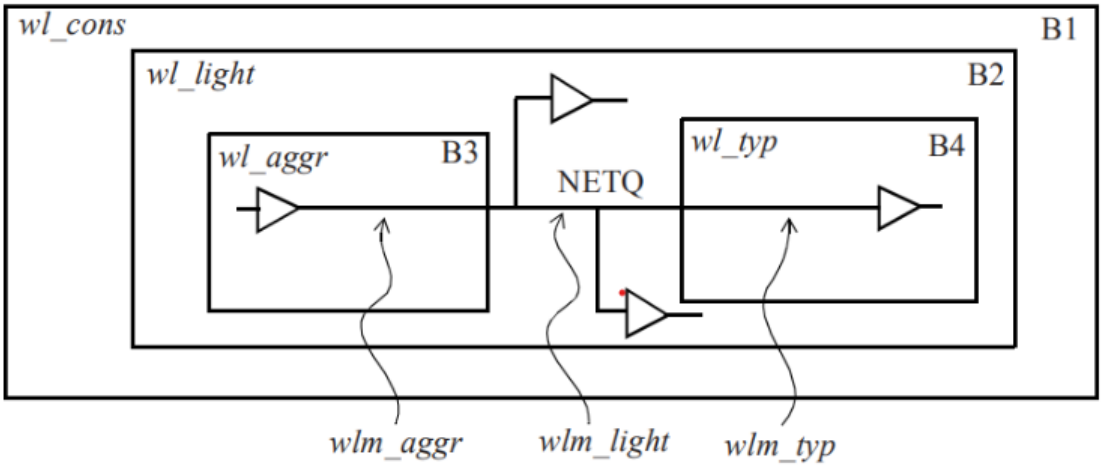
A zero wire load model specifies that there is no estimation of RC and delay of net is considered to be 0.
When net crosses hierarchical boundary then different wireload models are declared such as;

1. Top: In this any wireload inside a hierarchy is ignored and only the top level wireload is considered.
2. Enclosed: The net used for that block is completely estimated in wire load model.
3. Segmented: Every portion of net in a block is segmented and RC at every portion is calculated for wire load.

The wire load model can be specified using a command:
set_Wire_load_mode top/enclosed/segmented





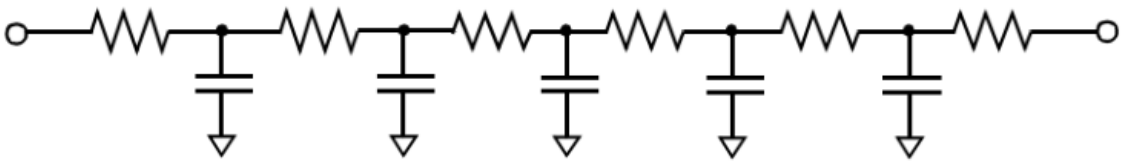


CHAPTER 5 INTERCONNECT AND DELAY CALCULATION

This chapter deals with the interconnect between the cells and its internal effects during timing verification.

5.1 RLC for interconnect

As the interconnection is made of metal wire which internally has resistance. In this context we consider the resistance between the output pin of the cell and input pin of fanout cells.
The inductance effect is seen in current loops. The inductance effect in the chip can be ignored at the chip level but is important at the board level. In chip level the loops are narrow and short thus making the inductance effect negligible.
So we discuss RC effects in the metal interconnect throughout this book.

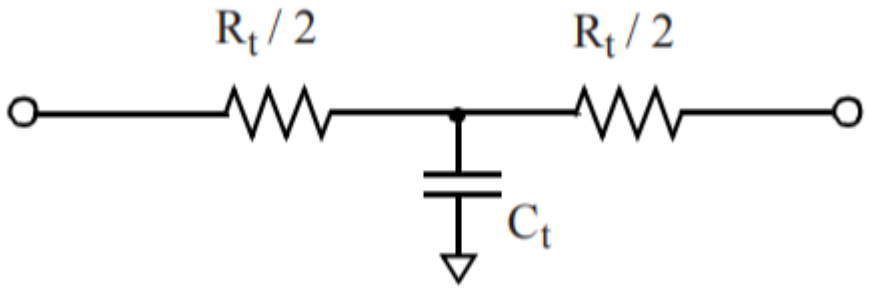


Distributed RC Tree

The interconnect RC can be modelled in different ways like:

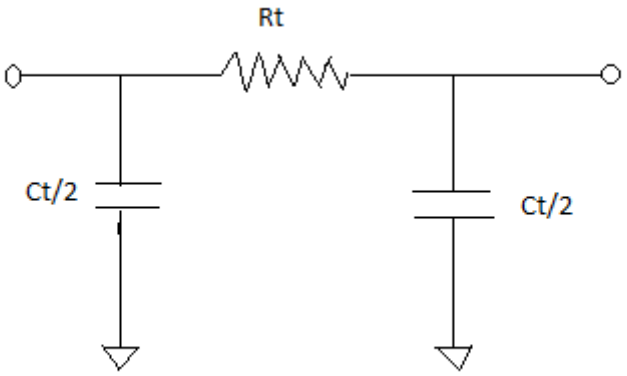
1. **Tree model:**

In this model the capacitance is connected halfway to the resistance. The resistance is divided 2.



2. **Pi model:**

The total capacitance is broken into two sections and a resistor in between them.



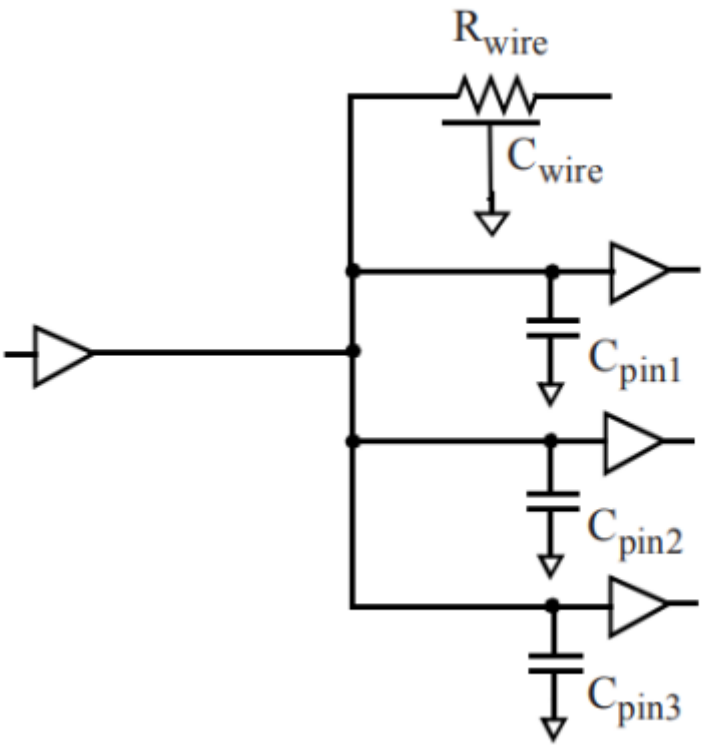
5.2 Post layout interconnect

In previous chapter we have discussed how interconnect is estimated in pre-layout stage ie. logical STA. Now we shall know how interconnect is considered in the post layout stage. We have input files which are used for RC extraction once physical connections between cells are created.

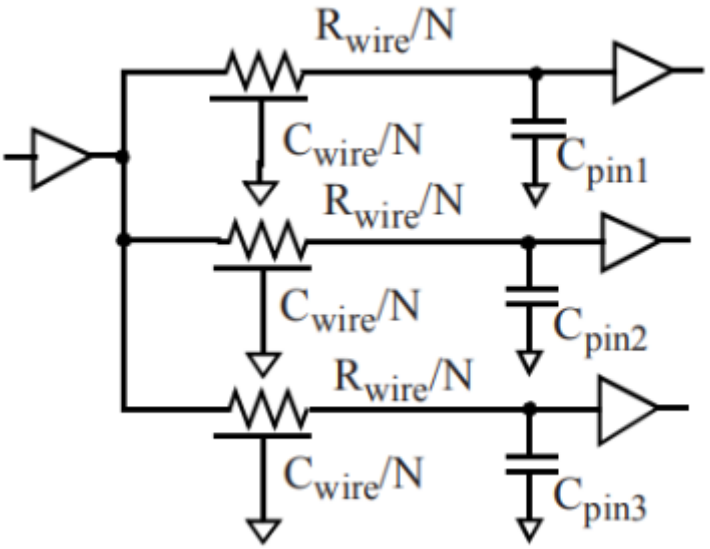
5.2.1 Interconnect trees

We have to check how our interconnect is connected with respect to the drive cell.
In pre-layout we estimate the interconnect from driver is connected in 3 possible ways:

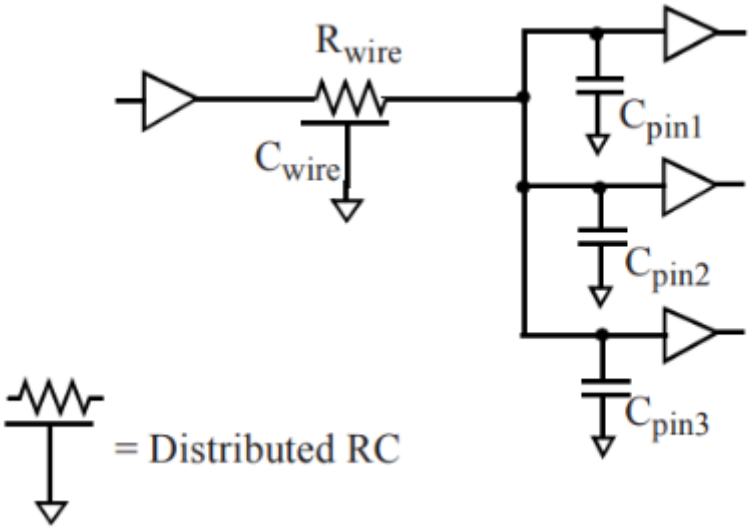
1. ***Best-case tree:*** The load pin is considered to be near the drive cell so reducing interconnect parasitic effect.
2. ***Balanced tree:*** Each destination is assumed to be on a separate portion of interconnect wire. So each path sees an equal portion of RC on the wire.
3. ***Worst-case tree*****:** In this case all destination pins are assumed to be far.



Best Case Tree



Balanced Tree



Worst-case Tree

5.2.2 Standard parasitics

In post layout STA the RC estimation will no longer come into picture. The real RC extraction is done using RC input files.
There are 3 formats of the extraction parasitics:

1. **Detailed standard parasitic format (DSPF)**

In this the detailed parasitics are represented in SPICE format.This file is too large as it is verbose for a total block so this is not used much. This calculates the RC effect for every net and provides accurate results.

2. **Reduced standard parasitic format (RSPF)**

This format uses the format of DSPF and reduces the parasitics but not the file format so even this holds drawbacks in usage. It considers only near end and far end capacitance and resistance effects with those wires which are in the pi model. The DSPF gives more accurate results of RC extraction.

3. **Standard parasitic extraction format(SPEF)**

It has reduced form of parasitics and as well the file format so this is most used. The modelling of RC varies with every net based on its fanout cell's drive and length of interconnect. Thus SPEF is more used.

5.2.3 Capacitances

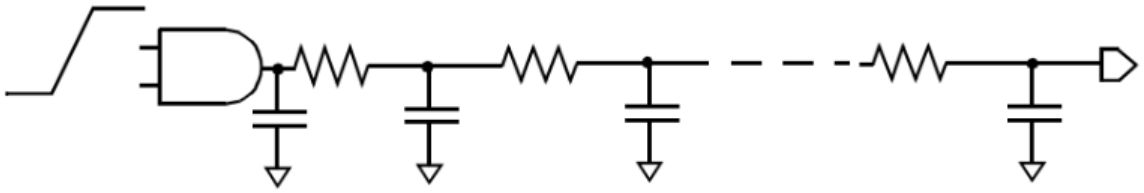
The interconnect sees capacitance effects in nanometer technology as sidewall capacitance**[1]**, coupling capacitance**[2]**, and ground capacitance**[3]**.
Note[1]: The sidewall capacitance/fringe capacitance is the capacitance seen between the side wall of CMOS with the surface of another CMOS.
Note[2]: The coupling capacitance is formed when two wires are connected in parallel with less distance between.
Note[3]: The ground capacitance is seen when a wire is grounded.
The representation of coupling capacitance is not described in DSPF and RSPF but is well defined in the SPEF.

5.3 Reducing interconnect resistance

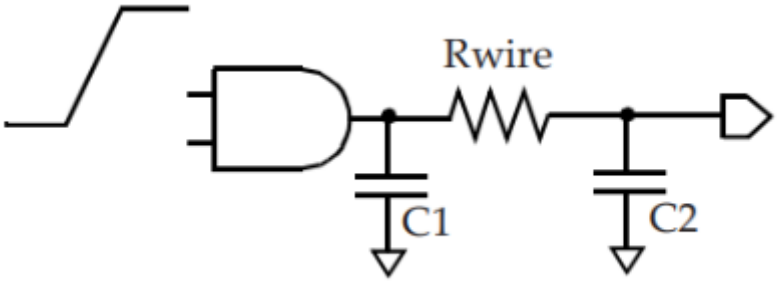
1. ***Wide trace:*** Having wire with more than the width can reduce the resistance effect without much increase in capacitance.
2. ***Higher metals:*** The higher metal layers in physical design have low resistance as it deals with power routing.

5.4 Effective capacitance

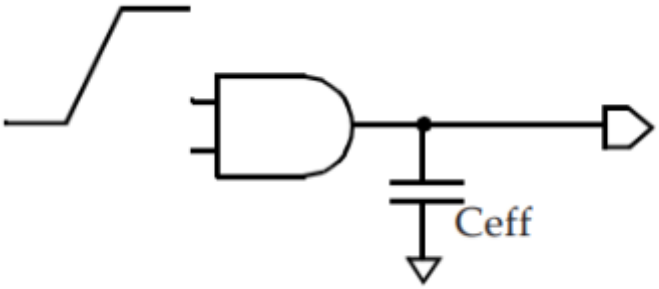
The interconnect sees a distributed RC network throughout the wire. The total capacitances on the wire can be replaced by a single capacitance ie. effective capacitance assuming that the original capacitance wire model and the effective capacitance wire model by considering resistance is mostly negligible sees the same delay.



RC Interconnect



PI model



Effective Capacitance

The effective capacitance is the function of:
1. The driving cell
2. The input impedance of the load as seen from the driving cell.

Note: For a given interconnect the cell with low output drive will see larger effective capacitance than with the high drive. So the effective capacitance will be between the minimum value of capacitances and maximum value which is the same as total capacitance when the interconnect resistance is negligible.

The phenomenon of near end capacitor charging faster than far end capacitor is called the resistive shielding effect of interconnect.

5.5 Interconnect delay

The effective capacitance approach deals with the output drive cell and the equivalent thevenin source at the output of the drive cell to compute the delay.

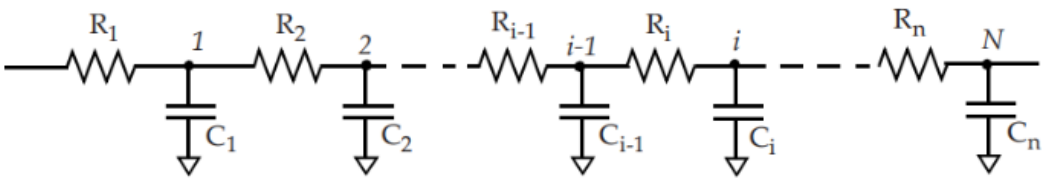
In pre layout timing analysis the RC interconnect type is determined by the tree models. There are three tree models: worst case for maximum delay, best case for minimum delay and the balanced tree for nominal delay.

5.5.1 Delay calculations on tree models

Elmore delay:

An RC tree meets the following three conditions:

1. Has a single input (source) node.
2. Does not have any resistive loops.
3. All capacitances are between a node and ground



The delays to various intermediate nodes are represented as:

$T_{d1} = C_1 * R_1;$
$T_{d2} = C_1 * R_1 + C_2 * (R_1 + R_2);$
. . .
$T_{dn} = \Sigma(i=1,N)\, C_i\, (\Sigma(j=1,i)\, R_j);$ # Elmore delay equation

For the best case model the entire tree sees the same R and C and can be c distributed into T or Pi models.

$$R_{wire} * (C_{wire} / 2 + C_{load})$$

For balanced model the R on wire is equal on all branches:

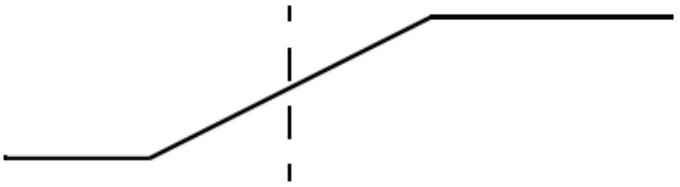
$$\text{net delay} = (R_{wire} / N) * (C_{wire} / (2 * N) + C_{pin})$$

In worst case model the RC on each wire is different:

$$\text{Net delay} = R_{wire} * (C_{wire} / 2 + C_{pins})$$

5.6 Slew merging

When there are multiple slews there are chances for the slew to merge with each other when they reach a common point. In such cases it is difficult for a static timing analysis environment to compute delay based on worst or best slew propagation so the timing tool should enable an option to calculate the timing during slew merge points.



(a) Slew at pin Z due to *A*->Z arc.

(b) Slew at pin Z due to *B*->Z arc.

The slew can be correct either in the worst case or best case ie. by max path propagation and min path propagation.
There are two possibilities when doing max path analysis:

i. Worst slew propagation:

This mode selects the worst slew at the merge point to propagate. For a timing path that goes through pins A->Z it is the worst case and can be selected, but for any timing path that goes through pins B->Z it is pessimistic.

ii. Worst arrival propagation:

This mode selects the worst arrival time at the merge point to propagate. The slew chosen in this case is exact for a timing path that goes through pins B->Z but is optimistic for a timing path that goes through pins A->Z.

There are two possibilities when doing min path analysis:

i. Best slew propagation:

This mode selects the best slew at the merge point to propagate. For a timing path that goes through pins B->Z it is best case, for any timing path that goes through pins A->Z it is pessimistic. For the paths going through A->Z, the path delays are smaller than the actual values and are thus pessimistic for min path analysis.

ii. Best arrival propagation:

This mode selects the best arrival time at the merge point to propagate. The slew chosen in this case is exact for a timing path that goes through pins A->Z but the selection is larger than the actual values for a timing path that goes through pins B->Z. For the paths going through B->Z, the path delays are larger than the actual values and are thus optimistic for min path analysis.

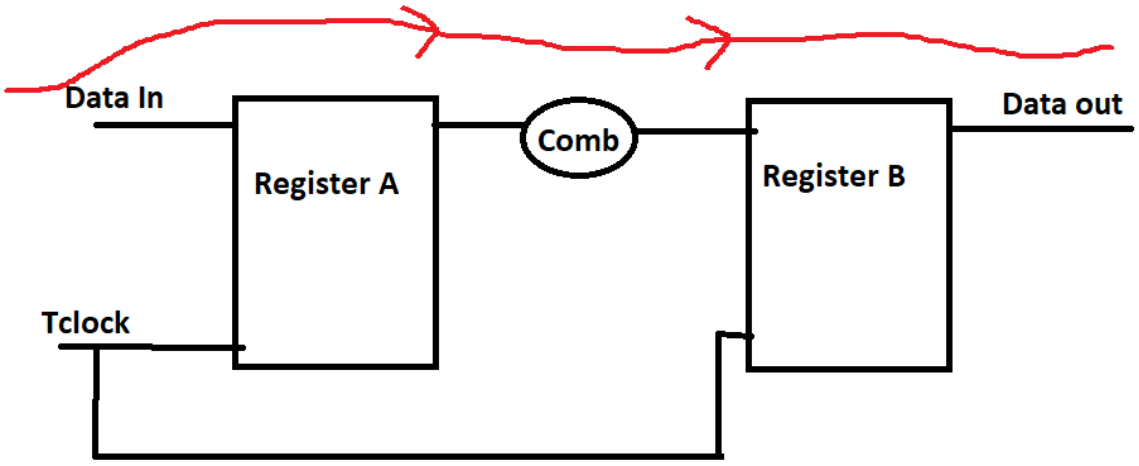
Note:

Due to RC effect the slew of a waveform will see a slew derate.

5.7 Path delay calculation

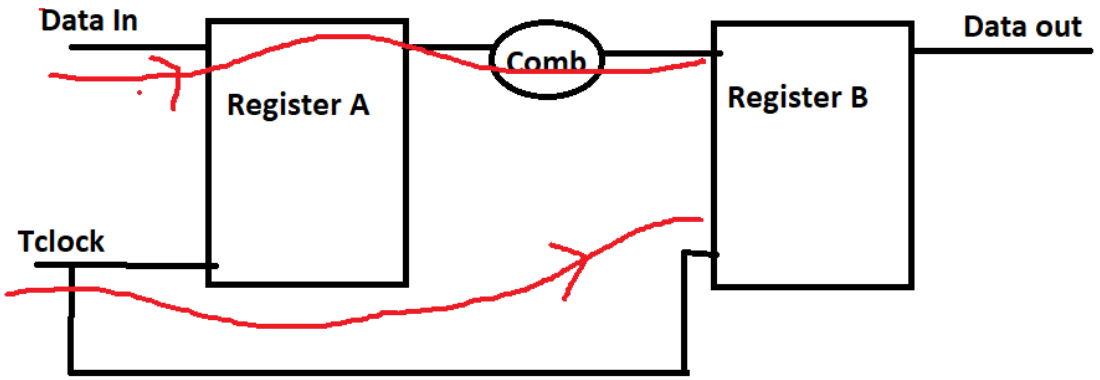
As discussed earlier, we have four timing path groups while computing delay in our design.

5.7.1 Input - output path



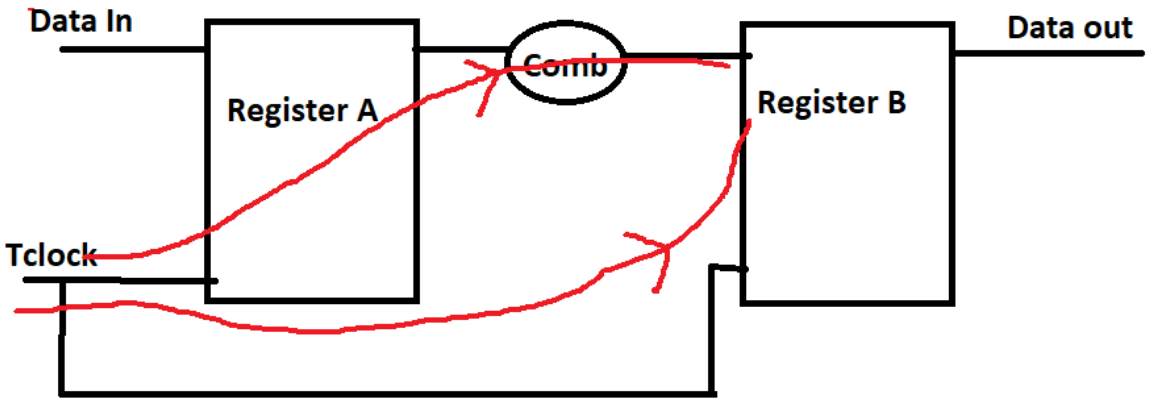
All the delays from input point to output are calculated.

5.7.2 Input to flip flop path



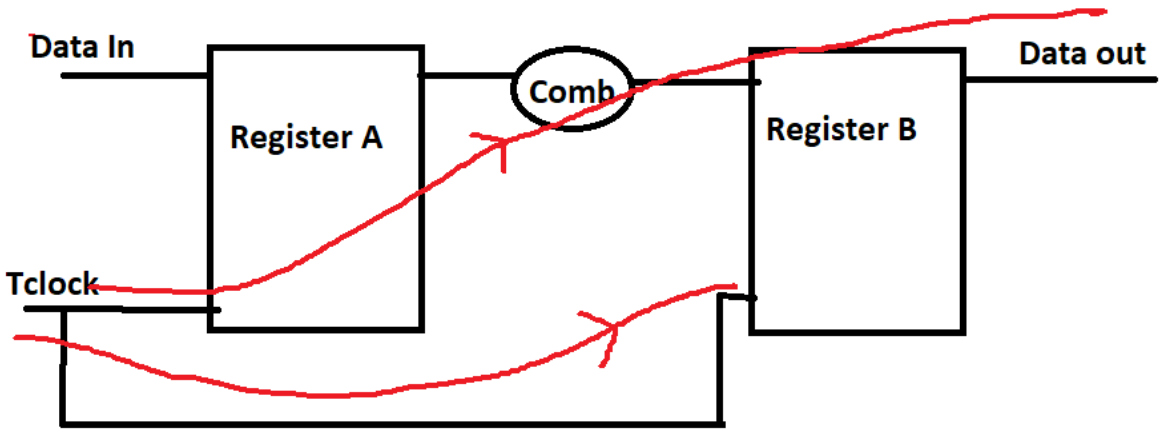
All the delays from input to flip flop are added.

5.7.3 Flip flop to flip flop path



All the delays from one flip flop start point to another flip flop end point are added.

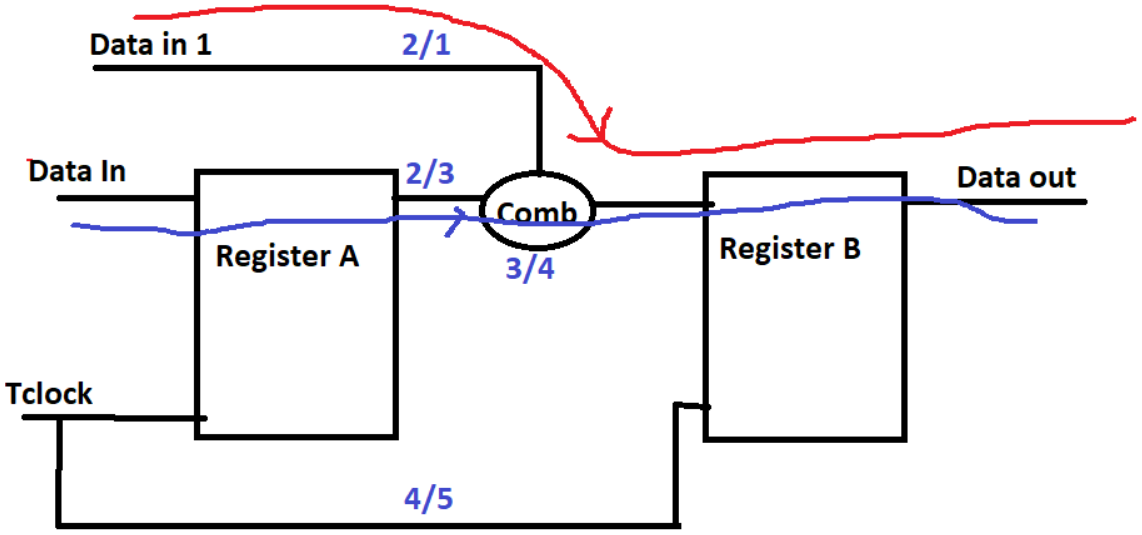
5.7.4 Flip flop to output path



All the delays from one flip flop start point to output are added.

5.7.5 Multiple paths

When there are multiple paths for worst case analysis the longest path is considered and for best cases analysis the shortest path is considered in delay calculation.



5.8 Slack calculation

Slack is the difference between the required time of a signal and the arrival time of that signal.

When data is launched by the launch flop it takes one cycle for data to be captured by the capture flop so the setup check is done after a clock cycle of the launch cycle.

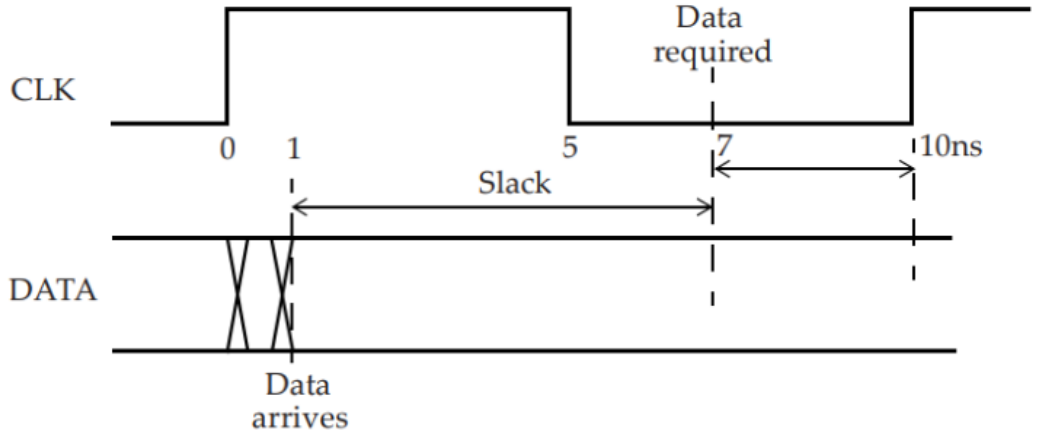
If the launch path is late and capture path is early and the setup slack meets in such a worst case condition then it is considered that the timing can be met at any other point also.

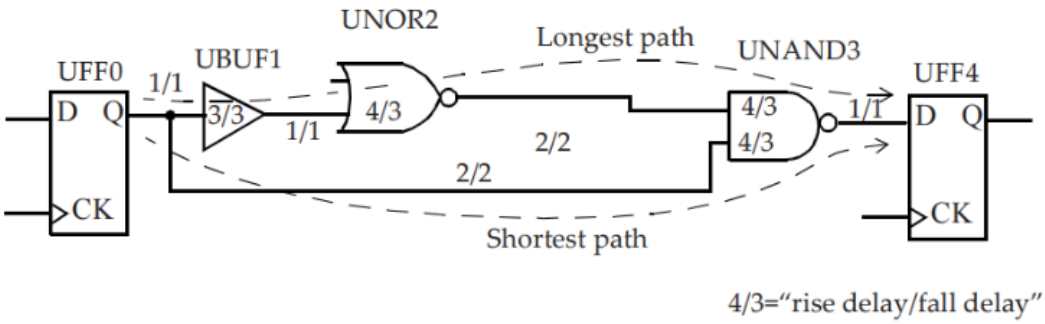
If the launch path is early and capture path is late and the hold slack meets in such a worst case condition then it is considered that the timing can be met at any other point also.

Launch path is the path from clock launch point to flop's clock point and data path is from clock to q path plus the combinational delay between launch flop and capture flop. Required path is the path from clock launch point to capture flop's clock point plus the propagation delay along the clock path minus/plus the setup/hold time respectively.

We will discuss an example of calculating setup and hold slack for worst case in chapter 9.

Setup slack = required time - arrival time
Hold slack = arrival time - required time





4/3="rise delay/fall delay"

CHAPTER 6 CROSSTALK AND NOISE

In the previous chapter we have discussed the interconnect and how delay is calculated. In this chapter we shall see how the interconnect disturbs our signal due to crosstalk. In deep submicron technologies as the interconnect distance reduces the coupling effect between signals increases which creates a huge impact in timing analysis.
The crosstalk causes noise and as well the timing issues. Crosstalk is unintentional coupling activity between two or more signals.

6.1 Noise and Signal integrity

Noise is the unintentional effects that affect the operation of a chip. In deep submicron technologies the impact of noise becomes more significant.

Let's see why noise is so important in deep submicron technologies.

1. **Increased metal layers:** The number of metal layers in lower technology nodes is more compared to the higher technology nodes. In lower technology nodes the number of cells in a chip increases thereby increasing the requirement of metal connections.
2. **Vertical dominant metal aspect ratio:** As the wires are getting thin and tall the effect of sidewall capacitance increases which is minimum in previous technology nodes.
3. **Larger number of interacting devices:** As number of cells in the chip increases the interacting devices increase creating more denser wire connections leading to coupling effect.
4. **Faster waveforms:** As frequency of chip increases the waveforms switch fast creating current spikes leading to distortion in signal.

6.2 Crosstalk

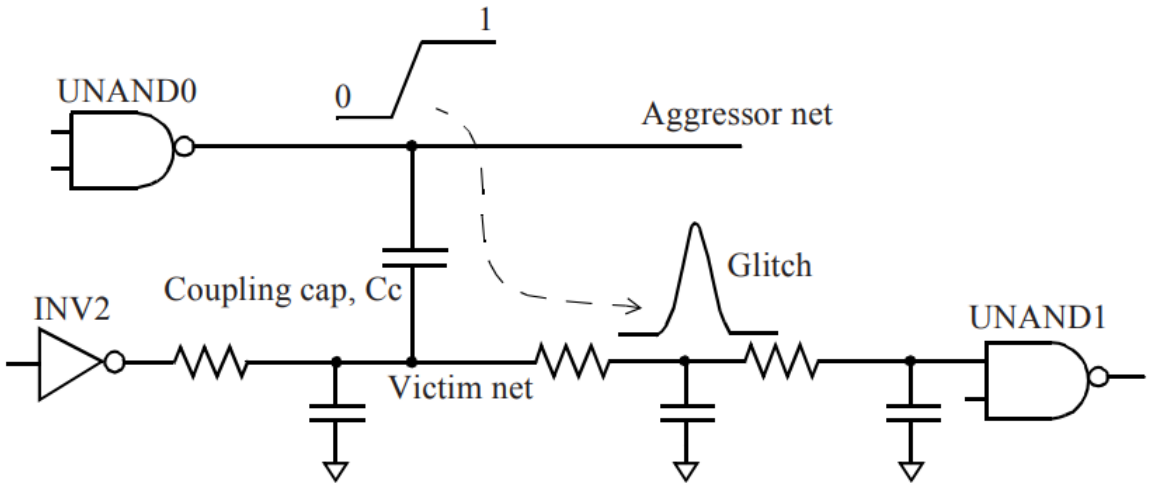
The unintentional coupling between two or more signals is called crosstalk. When a signal is being impacted by another signal it is called victim net and the net that is causing the other signal to distort is called aggressor net.

The crosstalk causes two types of effects:

1. Glitch: It is due to switching aggressor and the steady state victim.
2. Timing: This is due to switching of both aggressor and victim.

6.3 Crosstalk glitch analysis

The glitch occurs when the aggressor net is switching and the victim net is steady. When the aggressor is switching the coupling capacitance seen between the aggressor net and the victim net gets charged and creates a bump in the victim net;s signal which is a glitch. The glitch could be positive or negative.

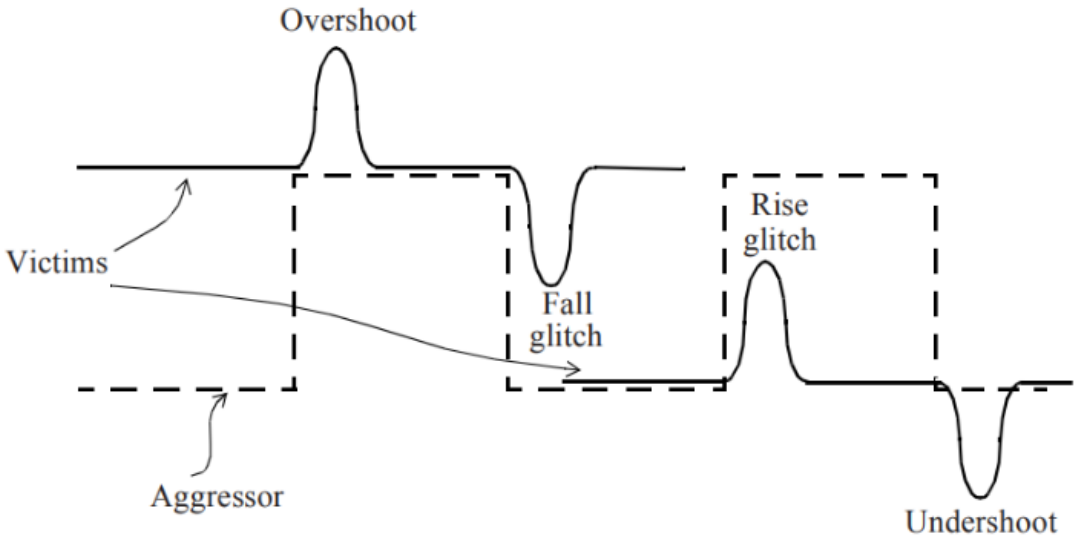


The magnitude of glitch is decided by several factors:

1. **Slew of the aggressor:** If the aggressor is switching fast then the amount of charge that coupling capacitor gets is more and creates larger glitch on the victim net.
2. **Coupling capacitance between the nets:** If the coupling capacitance is more then the amount charge it takes increases and creates larger glitches.
3. **Ground capacitance:** If the victim has good ground capacitance then the charges from the coupling capacitor that is creating glitch on the victim net gets grounded through the ground capacitor.
4. **Victim net drive strength:** If the drive strength of victim net is less then the output transition time will increase thereby the magnitude of glitch on the victim net increases.

6.3.1 Types of glitches

1. **Rise glitch:** This is due to steady low victim and high aggressor.
2. **Fall glitch:** This is due to steady high victim and low aggressor.
3. **Undershoot:** When both the aggressor and victim are below steady low.
4. **Overshoot:** When both the aggressor and victim are above steady high.

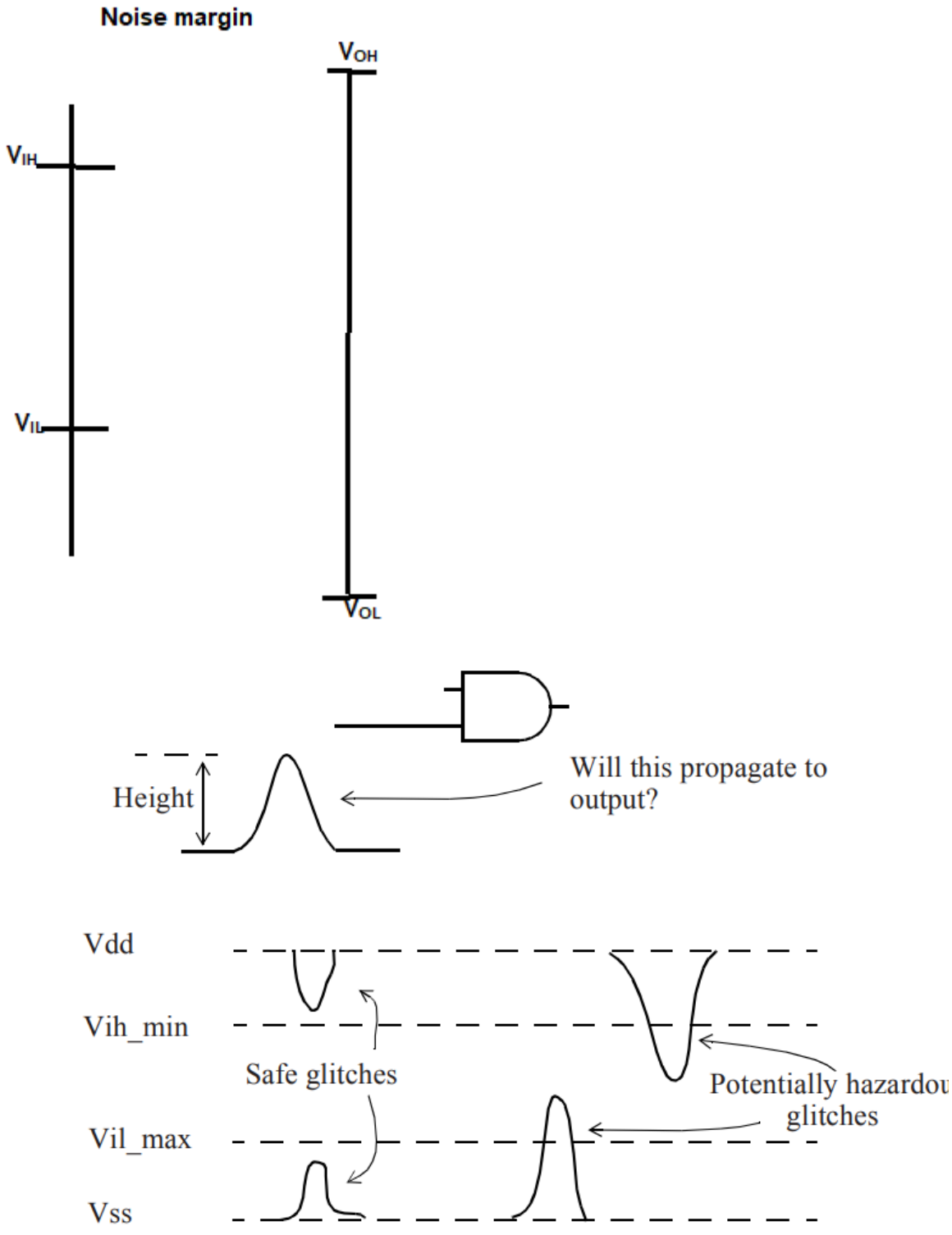


6.3.2 Glitch threshold and propagation

If a glitch occurs within the DC and AC noise margin limit then the glitch will not create much impact on the signal.

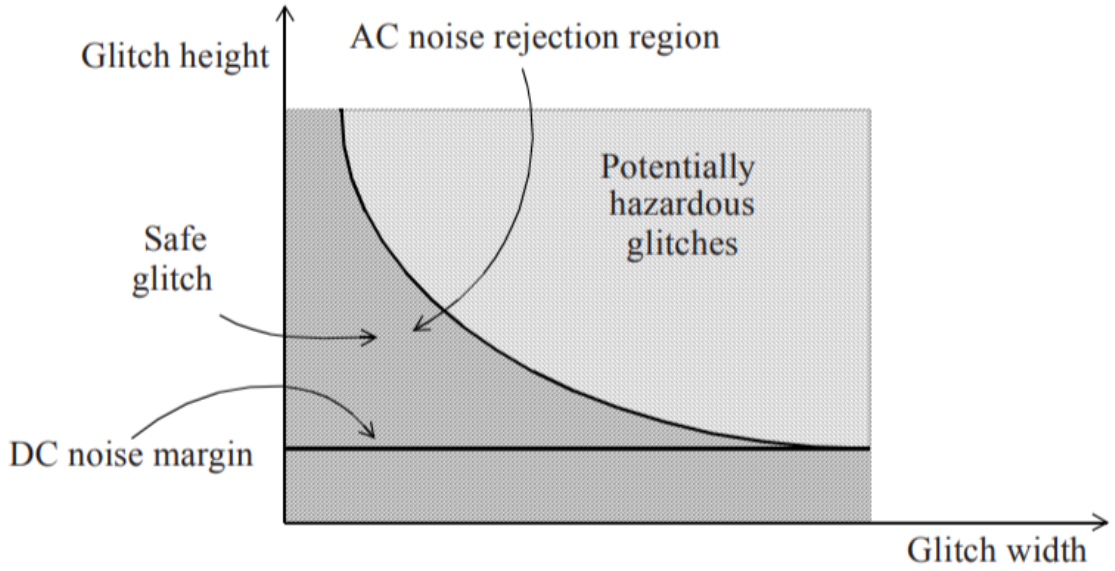
A) DC threshold

The amount of noise added to input could hold the output at logic 1 or 0 for the applied input without distortion is called noise margin.
NML = VIL – VOL Low noise margin.
NMH = VOH – VIH High noise margin.
NM = (NML + NMH)/2
Any glitch that is above Vil and below Vih will create a huge impact on the signal and has to be reduced.
The DC threshold decides the height of a glitch.



B) AC threshold

If the output capacitance of a cell is more then the glitch will be narrow as the cell delay increases. The narrow glitch does not create much impact on the signal.



So we can have three cases:

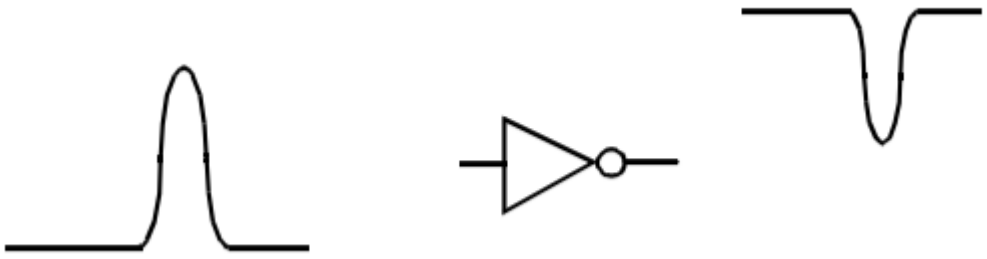
Case 1: When no output load, with positive input glitch creates a taller output glitch.

Case 2: When there is a load with the same positive input glitch the glitch will be small compared to the glitch at no output load.

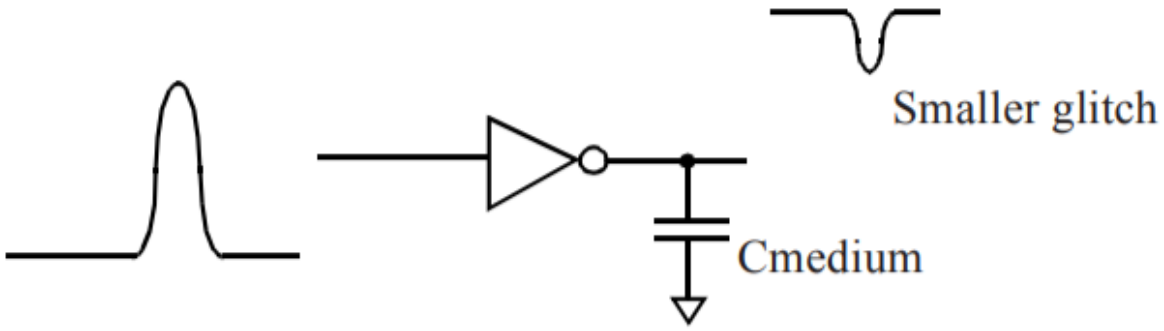
Case 3: With high load and the same positive input glitch then there will be 0 output glitch.

Thus increasing the output can reduce the glitch.

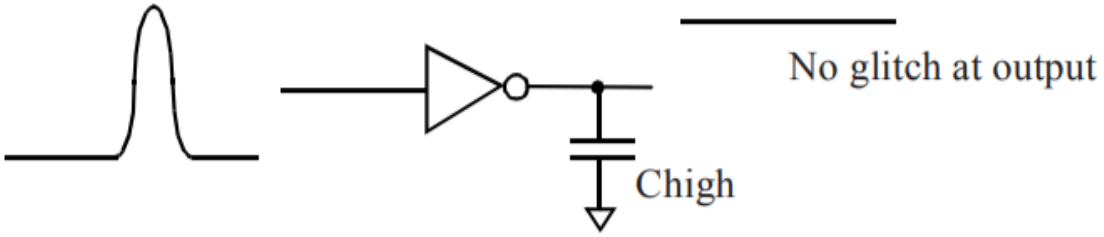
The amount of glitch that is accepted by a design is defined in the library.



No output load



Medium load Capacitance

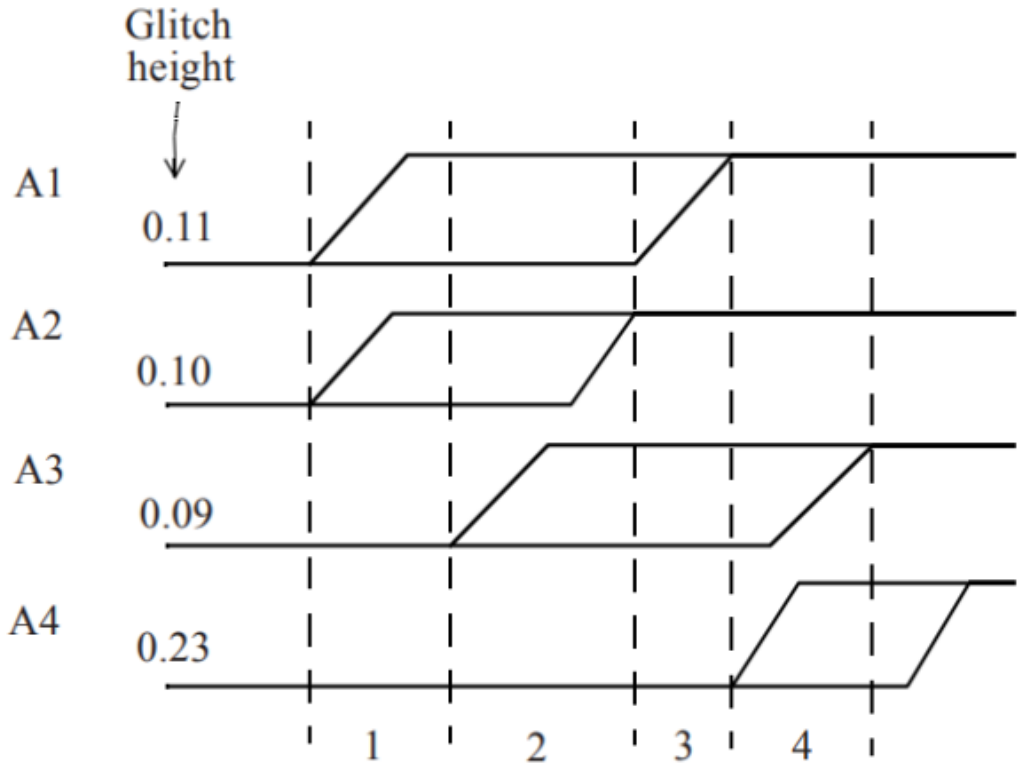


High load Capacitance

Input glitch is same in all 3 cases

6.3.3 Multiple switching aggressors

All the switching aggressors delay is computed and added on the victim net.
When a victim sees multiple switching aggressor; based on the concurrency in the aggressors, the glitches due to individual aggressors are added on the victim net. The glitches due to individual aggressor is calculated by computing all the 4 types of glitches(rise, fall, undershoot,overshoot). Then all the glitch contributors are combined and added on to the victim net.

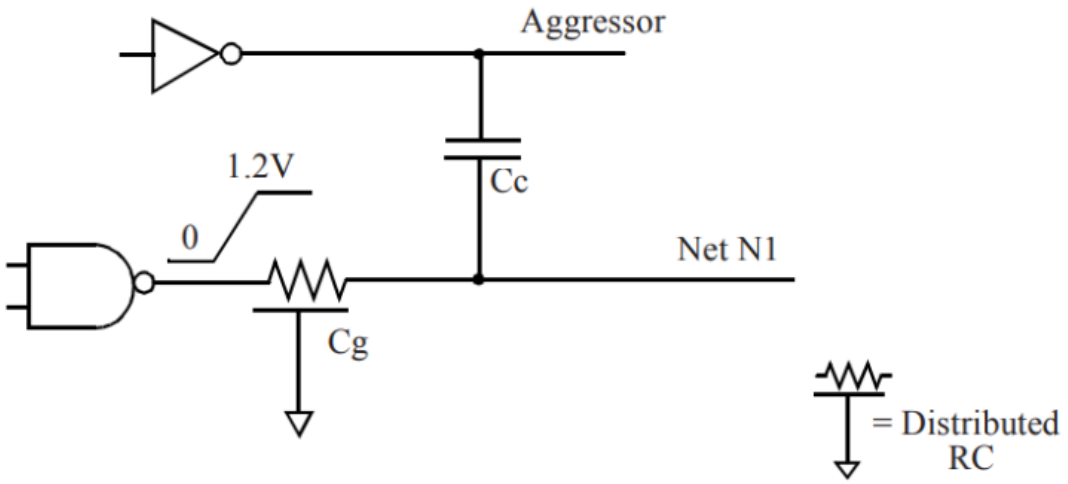


6.3.4 Aggressor functional correlation

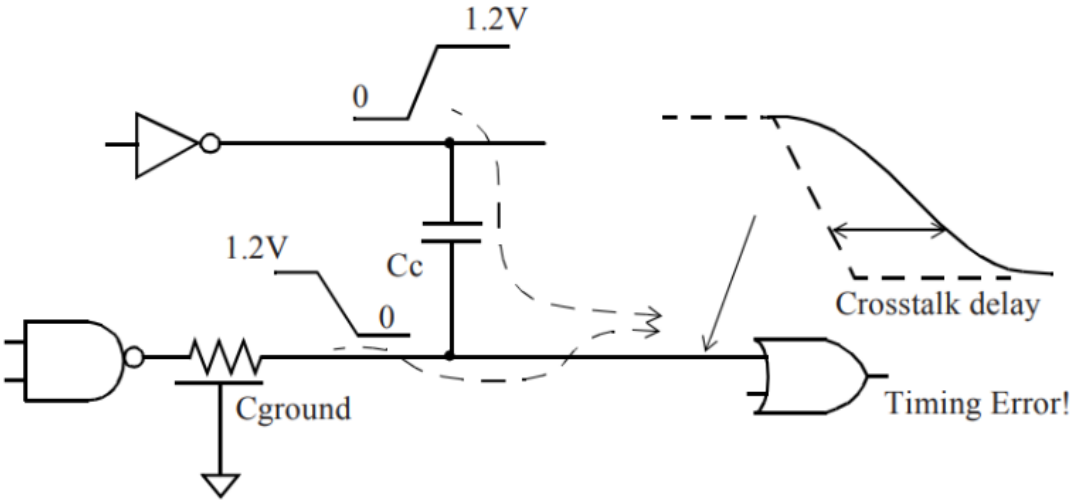
When we consider a scan control signal it switches only during scan mode and does not create crosstalk impact in functional mode. So the scan net will be aggressor only in scan mode.

6.4 Crosstalk timing analysis

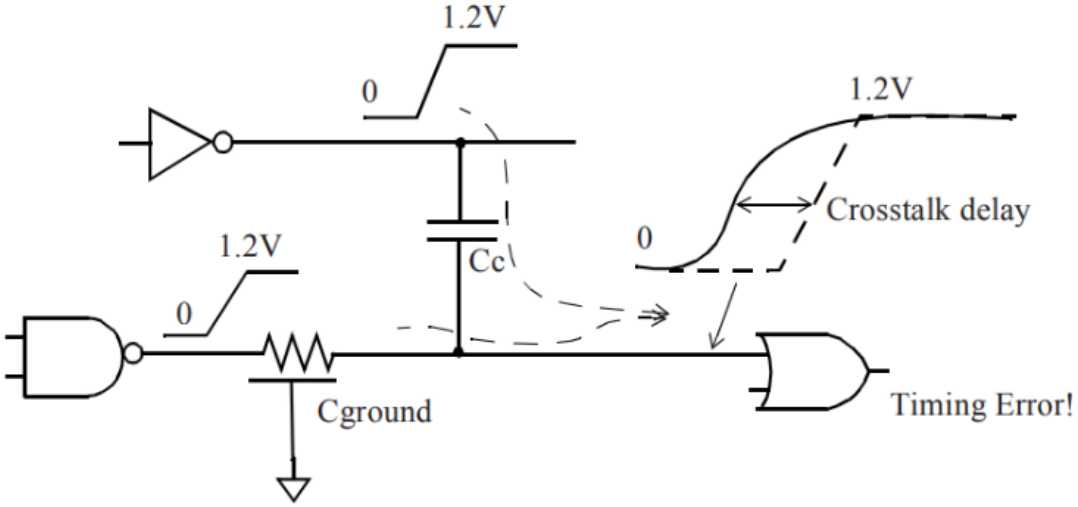
The capacitance effect in nanometer designs are more prominent as the interconnect spacing is very large which sees coupling capacitance. If the neighbouring net of the switching net is steady then all the charges from switching net are grounded through steady net as the steady net is considered to be a ground capacitance. But if the neighbouring net is also switching then with respect to the direction of switching on both the nets the delay is calculated.



Case 1: Aggressor net is steady: If the aggressor is in steady high then coupling capacitor is charged and if the aggressor is steady low then coupling capacitor is discharged.
Case 2: Aggressor switching in the same direction: If the aggressor and victim are switching in the same direction let's say both are rising signals then the charges from aggressor nets charge the coupling capacitor thereby making the victim net more easier to form a rise signal thus reducing the delay. So this is called negative crosstalk delay.
Case 2: Aggressor switching in the different direction: If the aggressor and victim are switching in the different direction let's say aggressor is rising signal and victim is falling signal, then the charges from aggressor nets charge the coupling capacitor thereby making the victim to form a rise signal instead of falling signal thus increases the work on victim to make its output signal to fall, this leads to increase in the delay. So this is called positive crosstalk delay.



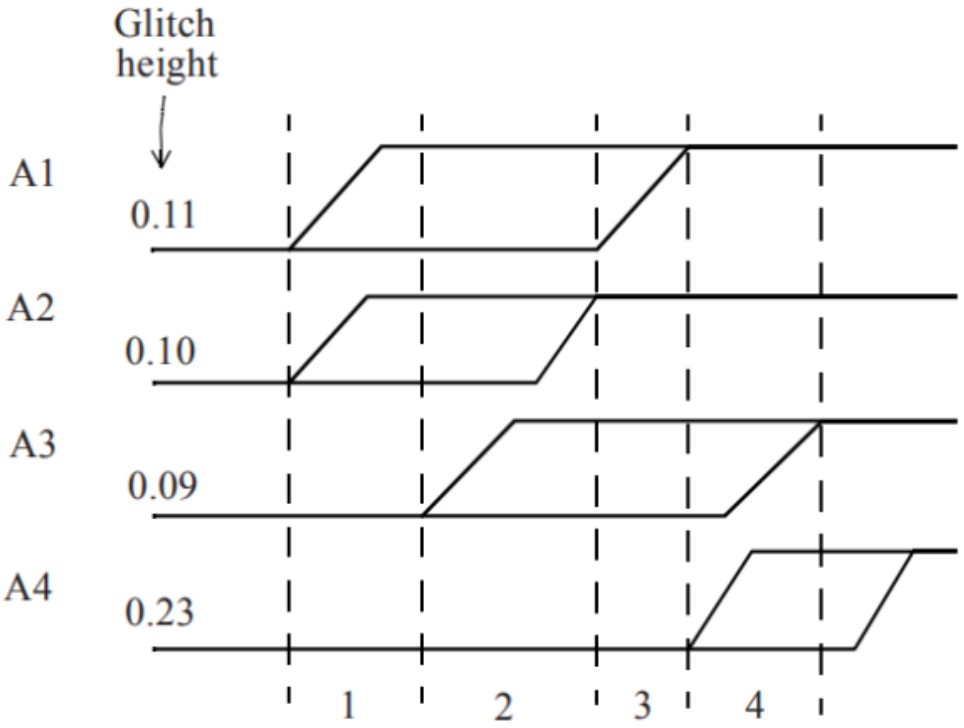
Positive Crosstalk delay



Negative Crosstalk delay

6.4.1 Multiple switching aggressors

The effect on the victim due to multiple aggressors holds the same concept as glitches with multiple aggressors. All the switching aggressors delay is computed and added on the victim net.
When a victim sees multiple switching aggressor; based on the concurrency in the aggressors, the glitches due to individual aggressors are added on the victim net. The glitches due to individual aggressor is calculated by computing all the 4 types of glitches(rise, fall, undershoot,overshoot). Then all the glitch contributors are combined and added on to the victim net. But the aggressor net which has worst case delay mainly affects the victim net.



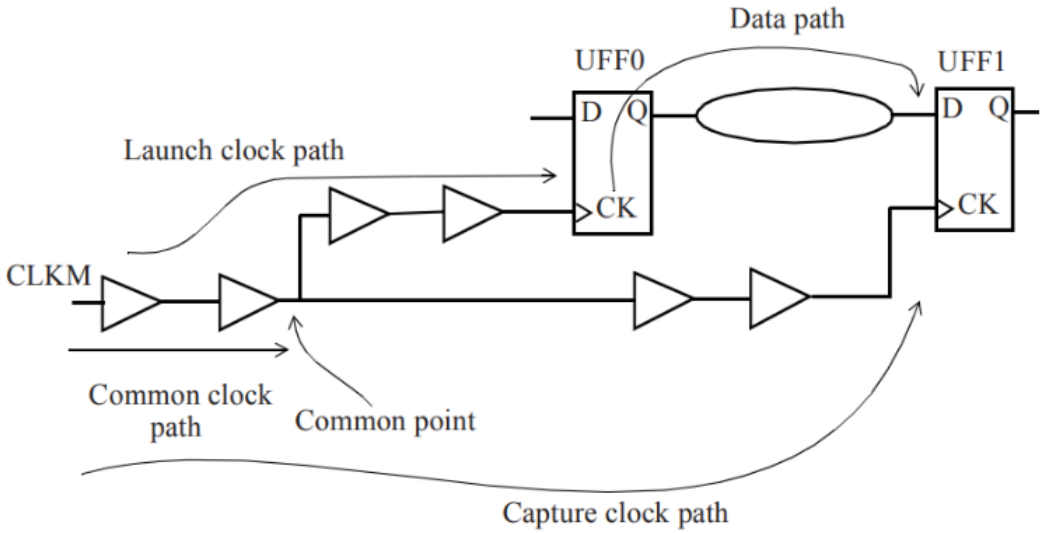
6.4.2 Aggressor functional correlation

When we consider a scan control signal it switches only during scan mode and does not create crosstalk impact in functional mode. So the scan net will be aggressor only in scan mode.

6.4.3 Timing verification using crosstalk delay

1. Positive rise delay: rise edge moves forward in time
2. Positive fall delay: fall edge moves forward in time
3. Negative rise delay: rise edge moves backward in time
4. Negative fall delay: fall edge moves backward in time

6.4.4 Setup and hold analysis using crosstalk delay



Crosstalk in datapath & clock path

A) Setup analysis

For setup time to meet the requirement the launch path plus the data path should be minimum and capture path should be maximum. But for the worst case analysis if the launch path plus the data path is maximum and capture path is minimum and still setup is met then it means setup can be met at any other case also.

So to meet setup in the worst case we have to make launch and data path late by positive crosstalk analysis and capture path in negative crosstalk analysis. If setup is met then the timing verification is clear.

B) Hold analysis

For hold time to meet the requirement the launch path plus the data path should be maximum and capture path should be minimum. But for the worst case analysis if the launch path plus the data path is minimum and capture path is maximum and still hold is met then it means hold can be met at any other case also.

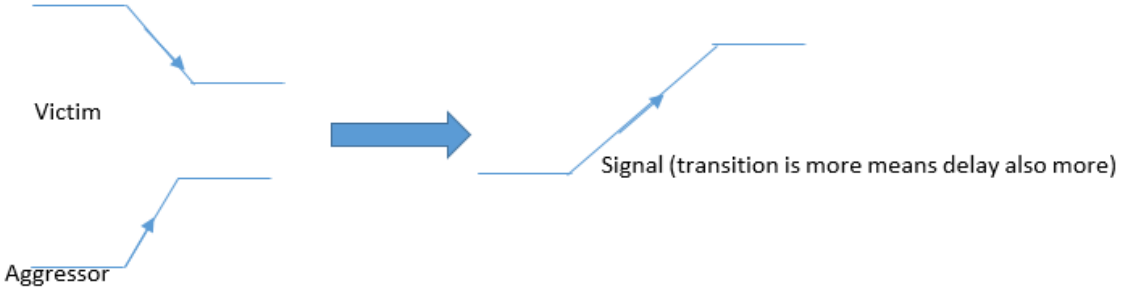
So to meet hold in the worst case we have to make launch and data path late by negative crosstalk analysis and capture path in positive crosstalk analysis. If hold met then the timing verification is clear.

Best case setup and hold analysis scenarios:

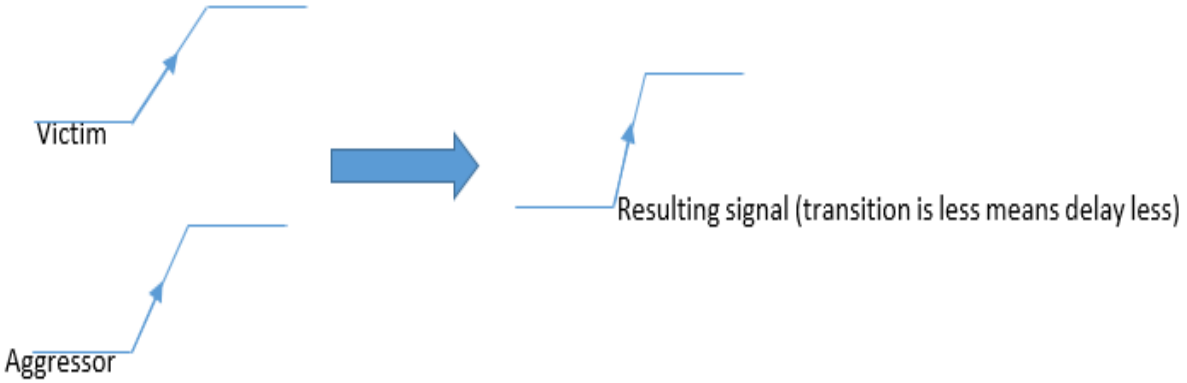
1. **Setup and hold analysis on data path**

- If the aggressor transitions in a different direction as a victim then victim transition becomes slow because this data will arrive late means arrival time will be more.

Setup = RT – AT(inc) this is not good for setup
Hold = AT(inc) – RT this is good for hold #inc- increase



If the aggressor transitions in the same direction as the victim then victim transition becomes fast because this data will arrive early means arrival time will be less.
Setup = RT – AT(dec) this is good for setup #dec- decrease
Hold = AT(dec) – RT this is bad for hold



2. Setup and hold analysis on clock path

If the aggressor transitions in the same direction as the victim then the victim transition becomes fast because this clock will arrive early means Required time will be less.
Setup = RT(dec) – AT this is not good for setup
Hold = AT – RT(dec) this is good for hold

If the aggressor transitions in a different direction as a victim then the victim transition becomes slow because this clock will arrive late means the Required time will be more.

Setup = RT(inc) – AT this is good for setup
Hold = AT – RT(inc) this is not good for hold

6.5 Crosstalk avoidance techniques

1. **Shielding:** The shield wires are placed on both sides of the critical signal nets. The power or ground nets are used for shielding.

The crosstalk effect in adjacent metal layers is quite less as compared with the same metal layers because the adjacent metal layers are orthogonal to each other which sees less coupling capacitance. But the same metal wires see more coupling capacitance as they run in parallel.

2. **Spacing:** Increasing the spacing between metal layers will reduce the effect of coupling capacitance and so thus the crosstalk effect.
3. **Fast slew rate:** If the slew rate is fast then the transition of the signal is less which gives less impact on crosstalk noise.
4. **Guard ring:** A guard ring in the substrate helps in reducing crosstalk effect.

CHAPTER 7 STA ENVIRONMENT

To make a proper verification on timing the STA environment has to be set up correctly. The setting up of the STA environment means creating a clock,setting maximum and minimum delays on the paths, specifying I/O delays, false paths, multicycle paths etc.
This chapter gives a little introduction on the configuration of the STA environment.

7.1 clocks

We know a clock has a duty cycle, period, rise and fall edge. All these have to be defined in the EDA tool so the tool understands the properties of the clock signal in our design.

1. **Create_clock** : This command is used to define the master clock of our block.

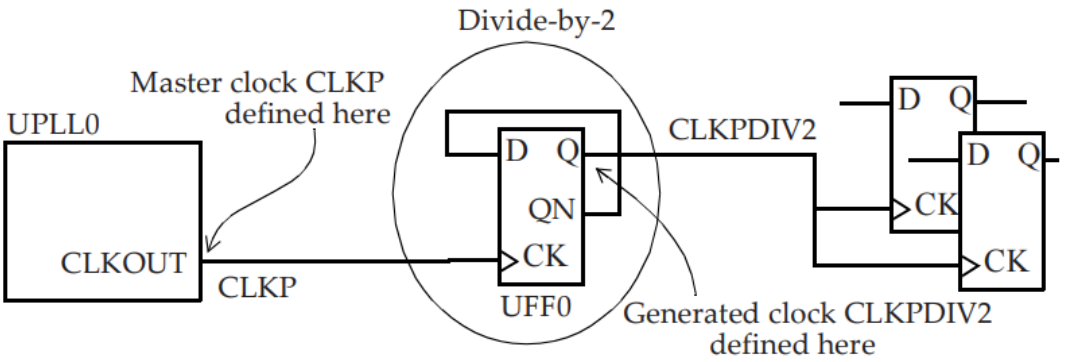
It has a few switches to define the master clock in detail.
[get_ports <port list>] helps the tool identify the start point of the clock.
Create_clock -name <user defined name> -period <value> [get_ports <ports list>]
If ports are not defined then the tool considers it as a virtual clock.

2. **Virtual clock:** This clock exists but is not associated with any pin/port of the design. This is used to specify input and output delays relative to the clock. This uses the same command as the master clock but does not have port declaration.

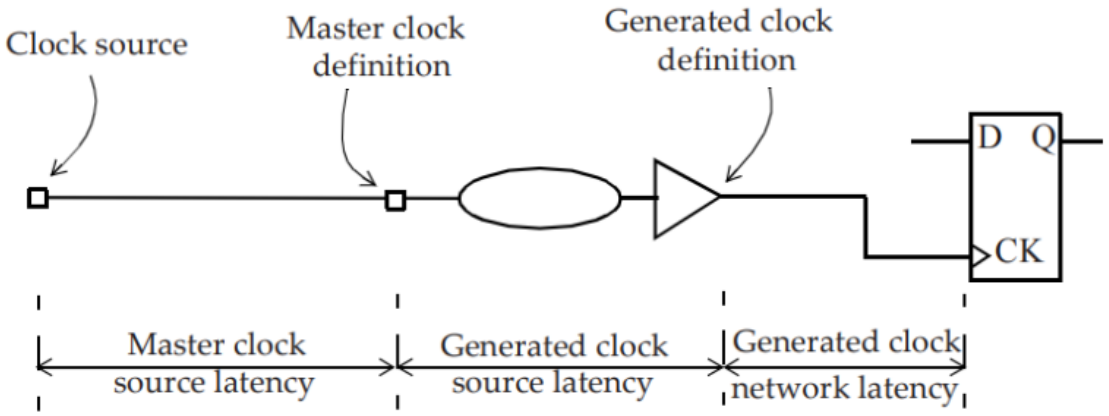
Create_clock -name <user defined name> -period <value>

3. **Generated clock:** The generated clock is the component of the master clock. If any other flop's output pin is acting as a clock then the designer can define it as a generated clock. One can even declare it as master clock instead of generated clock, in such case the new master clock has to be constrained by all the STA constraints so declaring such net as generated clock causes no harm as additional constraints are not required.

The generated clock is in phase with the master clock.
Create_generated_clock -divide_by 2 -name <user defined name> -source <source of the clock point> [get_pins <pin where the generated clock is generated>]



The generated clock can be constrained with clock latency. From the clock source point to clock definition point and from clock definition point to generated clock point can be computed as source latency and from generated clock point to register clock point is constrained as network latency.



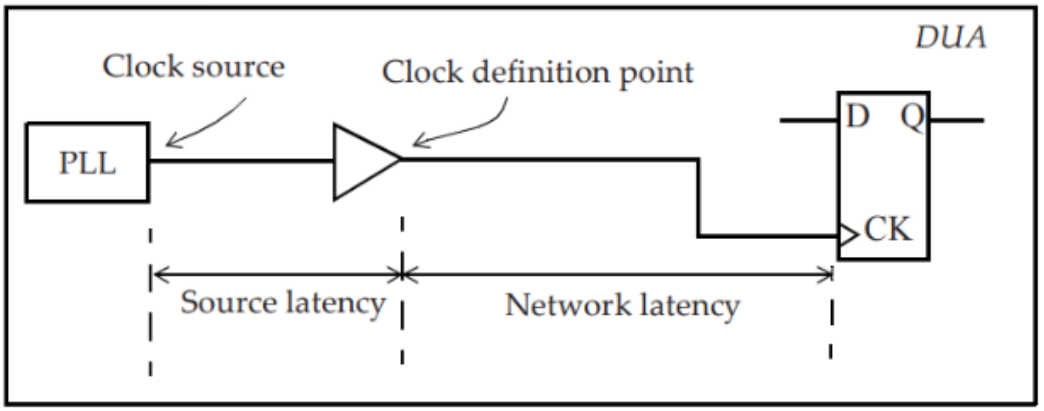
4. **Clock_uncertainty:** This is used to define if there are any uncertainties in the timing. This increases the clock period. The uncertainty in the timing can be due to many factors like jitter, skew, margin and crosstalk which have been discussed in earlier chapters.

This is constrained using the command: set_clock_uncertainty -setup/hold <value> [get_clocks <clock on which uncertainty has to be added>]

We reduce uncertainty in setup and add in hold. In setup we require additional clock period to meet setup time so we reduce uncertainty from the clock but for hold the minimum clock period is required so addition of uncertainty helps in hold.

5. **Clock_latency:** As the clock inserts delay on the signal it is insertion delay which has two components; source latency and network latency which have been discussed in earlier chapters.

Set_clock_latency -rise/fall <value> -max/min <value> [get_clock <clock on which latency is added>]
If -source is specified then the tool considers it as source latency if not it will be taken as network latency.
The source latency remains the same even after building the clock tree. But the network latency is the estimated delay of the clock prior to CTS.



On-chip clock source

7.2 Constraining I/O paths

If any path is not constrained the STA cannot verify the timing correctly so all the paths of our block have to be constrained.
We have an input and output signal to and from our block respectively where the current block owner does not know the delay of the input path.

Input path:

As there is an input path to our block from another block and delay is not known to the designer, the STA engineer has to constrain this input path with some delay. In this case the virtual clock plays a role where the input path's clock is not known; the designer can still create a clock signal as a virtual clock and define the input delay received by the block.
Set_input_delay -clock <master clock> -name <user defined name> [get_ports [remove_from_collection all_inputs] <clock name>]
We need to remove clock ports from the input delay as input delay is only on the data path.

Output path:

As there is an output path from our block to another block so delay has to be set by the designer, the STA engineer has to constrain this output path with some delay. In this case the virtual clock plays a role where the output path's clock is not known; the designer can still create a clock signal as a virtual clock and define the output delay received by the block.
Set_output_delay -clock <master clock> -name <user defined name> [get_ports all_outputs]

7.3 Modelling external attributes

Defining a clock, IO delays are more than enough for the STA but to know the accurate timing results there are still more factors which play an important role to decide the delay of the cell.
For input paths the slew decides the timing and for output paths the load decides the timing so the factors which affect the slew and output load have to be constrained.

1. **Set_drive:** It is used to drive the input ports of the block. Drive is the resistance of the cell. If the drive is more then the transition is more.

Set_drive 50 [all_inputs] : which means the cell has resistance of 50 units.

2. **Set_driving_cell:** This indicates the drive strength of a cell ie. inverse of resistance. If drive strength is more then the transition is less. It is the driving capability of a cell.

Set_driving_cell -lib_cell <cell name> [all_inputs]

3. **Set_load**: To model external load driven by the output set_load adds a load in the output port. By default capacitive load on all outputs are considered to be 0.

Set_load <value> [all_outputs]
The load has to be specified to calculate the delay of the design. If load is more then the delay is also high.

7.4 Design rule checks

These rules check that the design's port/pin meets the design rule checks within the specified limits of transition and capacitance. Any violations in these rules are reported in the slack so it is also known as design rule violations.
Set_max_transition and set_max_capacitance are two rules to constrain in case of violation.
DRV: Design rule violation: This includes max_transition, max_capacitance, max_fanout, max_length. Among these max_trans and max_cap are hard constraints as this violation must be fixed and max_fanout and max_length are soft constraints as this violation can be ignored.

7.5 Timing exceptions

7.5.1 set_case_analysis

In the design there are few signals whose value is constant in a certain mode and if this is not constrained then STA will consider it as one of the timing paths and may create violation. So set_case_analysis helps in fixing the value on the path and tells the tool that this path is not for timing analysis.

Set_case_analysis 0/1 [get_ports <ports on which case analysis should be given>]

7.5.2 Set_disable_timing

In certain timing paths there are chances that the path can not continue from a particular cell and the STA tries to verify the timing requirement on this path so to avoid such a scenario we can break the timing path by disabling that particular path.

Set_disable_timing -from <from pin> -to <to pin> [get_ports <ports on which time has to disabled>]

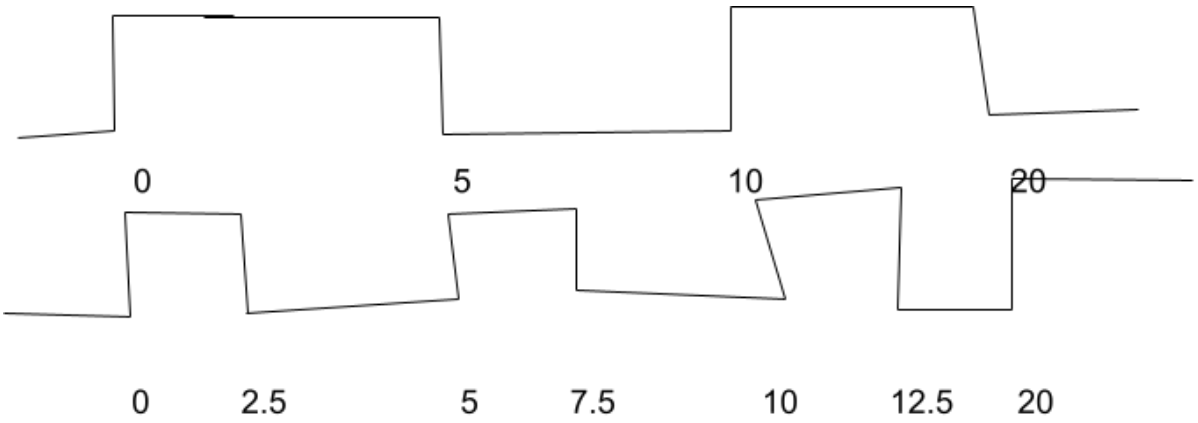
7.5.3 Multicycle path

In some cases the combinational block between two flip flops can take more than one clock cycle to propagate through the logic, in such cases the combinational path is called a multicycle path.

Though the capture flop in such a multicycle path domain captures data in every edge the STA redirects the capture edge based on the number of cycles the combination logic takes to propagate the data.

Set_multicycle_path <value> -setup/hold -from [get_ports <port list>] -to [get_ports <port list>]

In general we know the setup check is done next clock edge but the clock takes more than one cycle to propagate the data then the number of cycles the clock takes to reach the capture flop should be pushed to have a setup check. The hold check can be done one edge before the setup or at the same edge as of the launch.



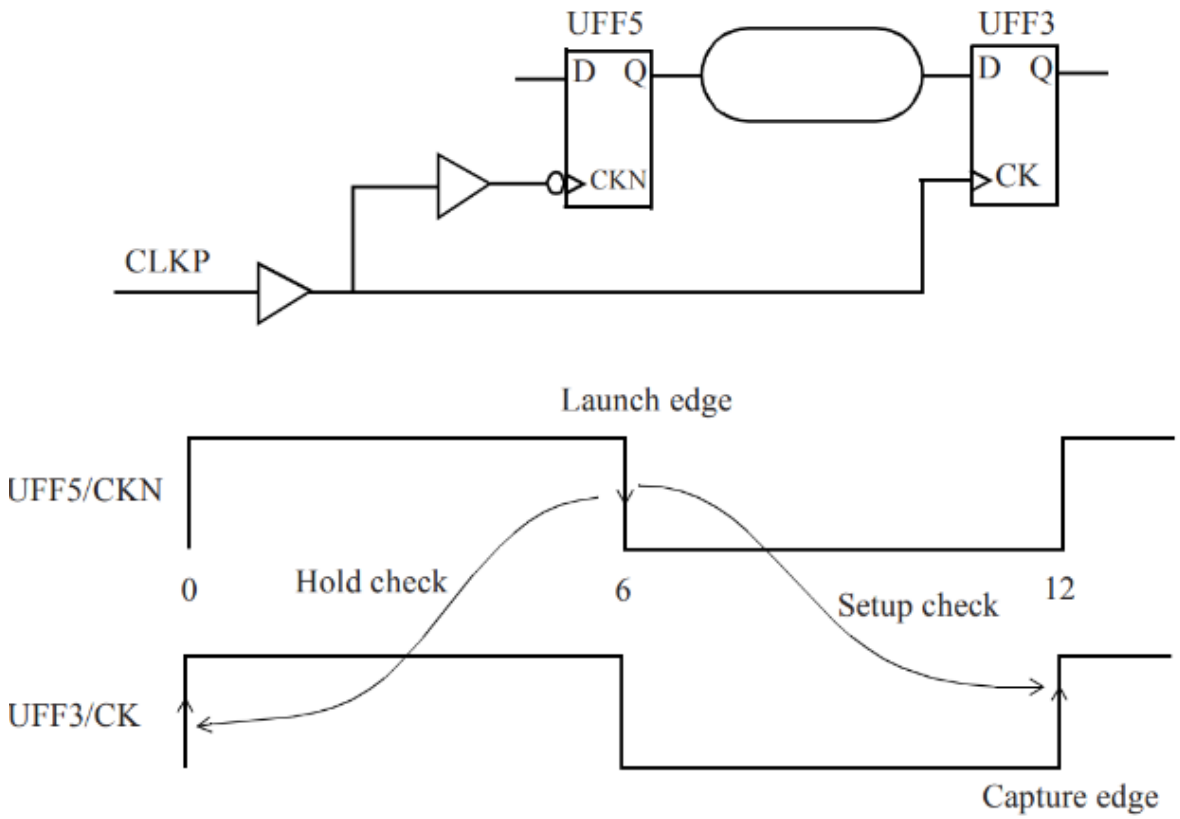
So for setup check by default tool checks at rise edge 5 but this will lead to loss in data so we have to push the setup check to 10 and hold by default checks at rise edge 5 ie. one edge prior to setup check. The hold check can be checked at rise edge 0 or rise edge 5.

7.5.4 False path

In the design certain timing paths are not possible and these paths can be turned off by using the set_false_path feature. This feature lets STA to ignore these paths for timing validation.
Setting a false path and multicycle path on the same port is not valid because the false path ignores the crosstalk analysis and may lead to timing violations.
Set_false_path -from [get_ports <port list>] -to [get_ports <port list>]

7.5.5 Half cycle path

When a launch edge is positive/negative edge and capture edge is at negative/positive edge respectively then the design contains a half cycle path.
In the below timing diagram we can conclude that the launch is at fall edge 6ns and capture is at rising edge 12ns which actually has to be 18ns, so due to the existing of half cycle path the capture is at 12ns which is at half way the total cycle and so thus the setup check at 12ns. The hold check is by default done at one edge prior to setup check so it can be done 6ns or 0ns which gets another half cycle.



7.6 Timing across clock domains

7.6.1 Slow to fast clock domain

In such slow to fast clock domains the clock which is slow can be expanded with the base period as of the fast clock. The setup check can be pushed by the number of cycles the fast clock is ahead of the slow clock using a multicycle path.

In the below shown fig. Let's consider the slow clock with 5ns and fast clock with 20ns periods. So now to have setup check at every slow clock rise edge we have to push the setup check by 4ns. The hold check has to be done either at the same edge as of launch or one edge before the setup.

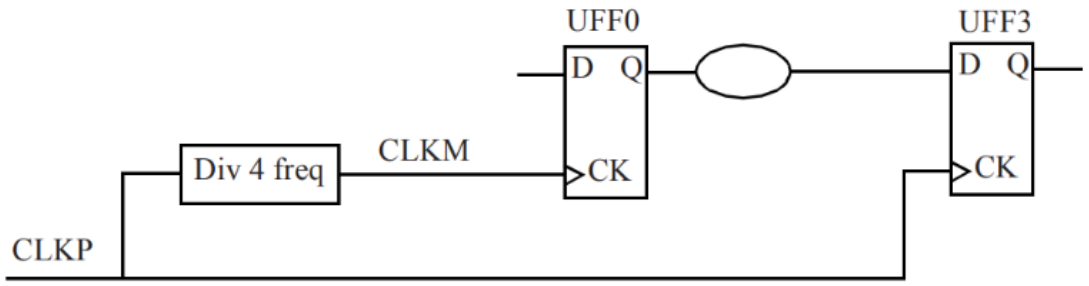
We need to create two clocks for each path.

Create_clock -name CLKM -period 20 -waveform {0 10}

Create_clock -name CLKP -period 5 -waveform {0 2.5}

Set_multicycle_path 4 -setup -from [get_clocks CLKM] -to [get_clocks CLKP]

Set_multicycle_path 3 -hold -from [get_clocks CLKM] -to [get_clocks CLKP]



7.6.2 Fast to slow clock domain

In such fast to slow clock domains the clock which is slow can be expanded with the base period as of the fast clock. The setup check can be pushed by the number of cycles the fast clock is ahead of the slow clock using a multicycle path.

In the below shown fig. Let's consider the slow clock with 5ns and fast clock with 20ns periods. So now to have setup check at every slow clock rise edge we have to push the setup check by 2ns. The hold check has to be done either at the same edge as of launch or one edge before the setup.

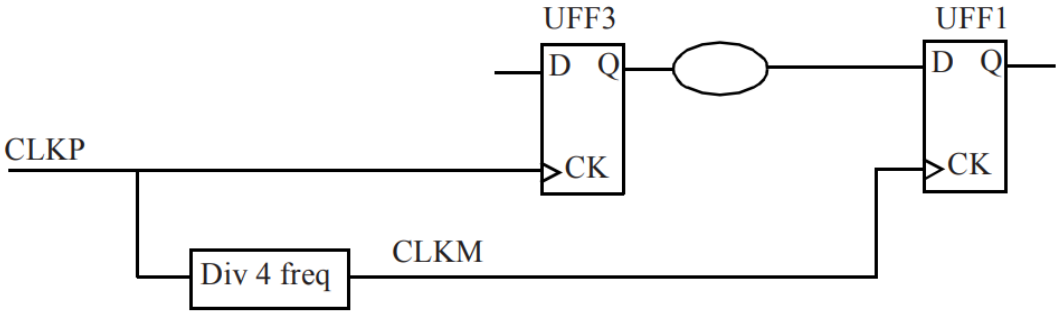
We need to create two clocks for each path.

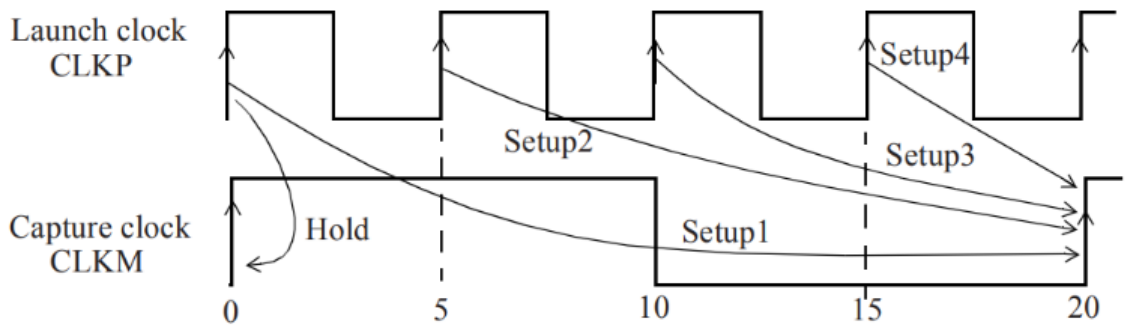
Create_clock -name CLKM -period 5 -waveform {0 2.5}

Create_clock -name CLKP -period 20 -waveform {0 10}

Set_multicycle_path 2 -setup -from [get_clocks CLKM] -to [get_clocks CLKP]

Set_multicycle_path 1 -hold -from [get_clocks CLKM] -to [get_clocks CLKP]





7.7 Multiple clocks

In our design we have multiple clocks and STA checks if all those clocks are inter multiples or not and the analysis varies.

7.7.1 Integer multiples

The STA computes checks the periods of all the clock domains and if they hold common multiple integers of period then the base period is expanded to have synchronous among all the clock domains.

7.7.2 Non integer multiples

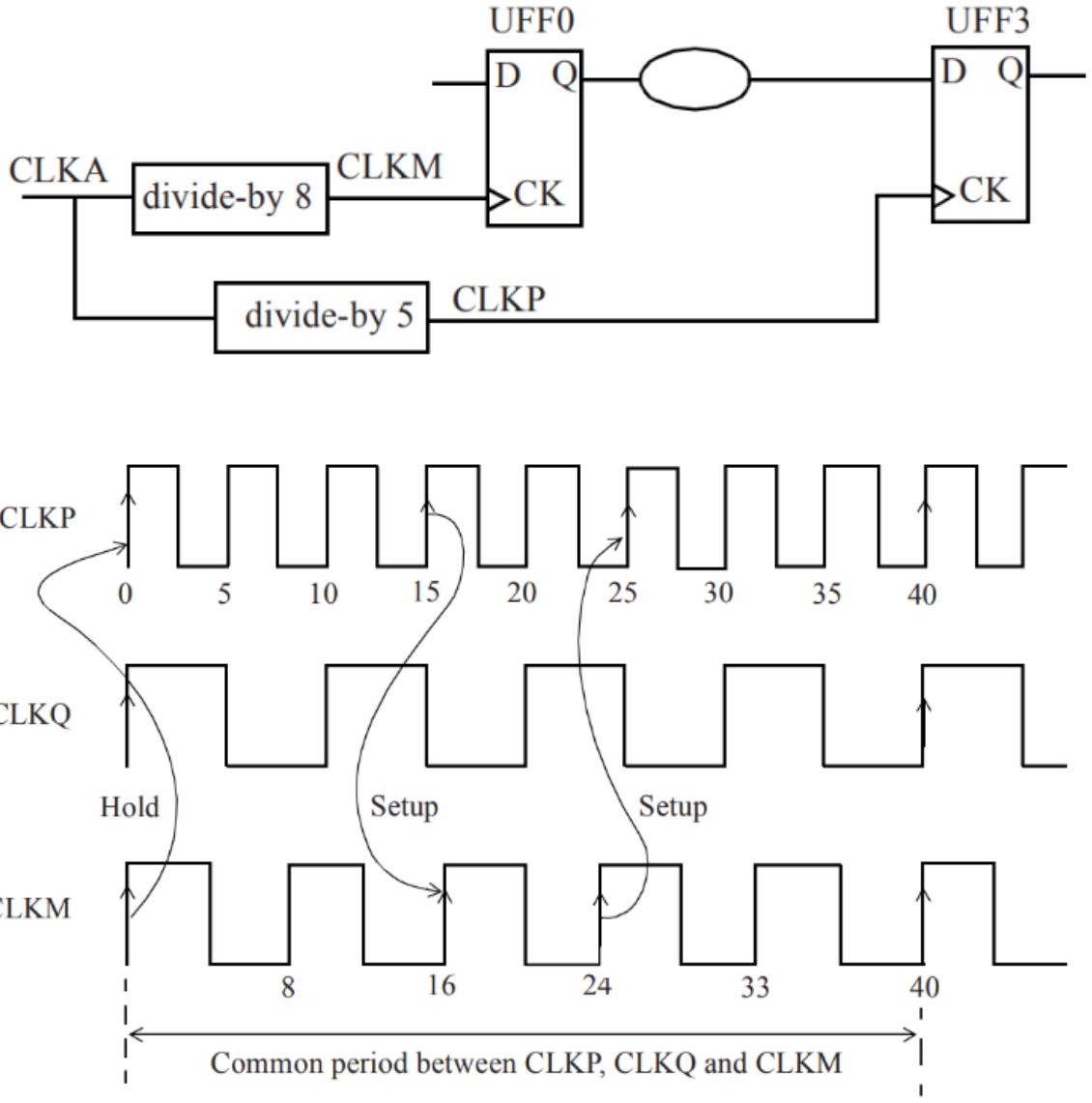
Since the clock domains do not have any relation with their periods, in such cases the STA adds a number of cycles so that both the clock domains can have common relation.

Create_clock -name CLKM -period 8 -waveform {0 4}

Create_clock -name CLKP -period 5 -waveform {0 2.5}

Expanding both the clock domains till 40ns will give relation with both the clocks.

The setup and hold checks are done at the minimum timing between launch edge and capture edge.


UFF0
UFF3
D Q
D Q
CLKA
divide-by 8
CLKM
CK
CK
divide-by 5
CLKP
CLKP
0 5 10 15 20 25 30 35 40
CLKQ
Hold
Setup
Setup
CLKM
8 16 24 33 40
Common period between CLKP, CLKQ and CLKM

This chapter discusses the advanced STA concepts like OCV, clock gating and non-sequential timing checks.

8.1 On - chip variations (OCV)

In our chapter 1 we have discussed the PVT corner of a chip. Due to some process variations the chip sees variation in the characteristics of MOSFET. This variation could be due to internal affect(process, voltage variations) or external parameters (climatic conditions etc). OCV gave correct results till 65nm technology. Any variation below 65nm did not have correct results so this situation led to two other concepts AOCV and POCV.

For the optimal performance of the chip, a lower temperature, higher voltage, and a fast process are necessary. A high-temperature, lower voltage, and slow process are the remarks of the worst chip.

Two types of variations:

1. Local process variations: variations seen on a die
2. Global process variations: variations seen across multiple manufacturing lots.

The chip does not only affect due to process variations it even depends on the voltage and temperature.

The PVT arises due to these following factors:

1. IR drop variation
2. Threshold voltage of PMOS and NMOS
3. Channel length variation of PMOS and NMOS
4. Temperature variation due to local hotspots (junction temperature variation and ambient temperature variation).
5. Interconnect metal etch or thickness variation

The PVT which impacts the complete chip and which arises due to above discussed factors is called OCV.

Due to above mentioned factors the cells in the chip may have some variation in voltage which may increase or decrease the delay of the cell which could lead to bad timing. So to overcome this STA constraints have a feature to derate the cell by a marginal factor.

The derate value is given by the foundry based on the voltage variation a cell sees. For the worst case setup maximum derate and for the best case hold minimum derate is given to meet the timing.

Set_timing_derate -early/late <maxvalue/minvalue>

If there is the presence of the common clock path for both launch and capture then the maximum and minimum are subtracted and multiplied with the derate value to remove the common path pessimism. This process is called Common Path Pessimism Removal(CPPR) or Common Reconvergence Pessimism Removal (CRPR).
The derate should be added in setup analysis and removed in hold analysis.

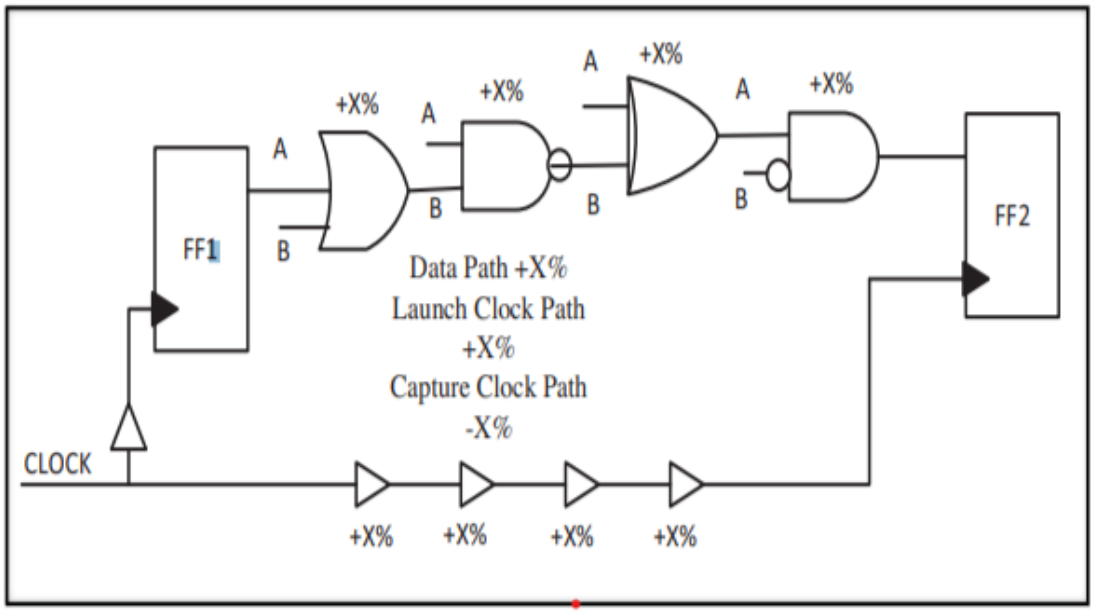
8.1.1 AOCV - Advanced On Chip Variation

The OCV has been using the most pessimistic derate value on the complete chip but when the technology node is shrinking this pessimistic value no longer gives correct results in timing closure. So to overcome this AOCV has been developed which does distance and depth based analysis.

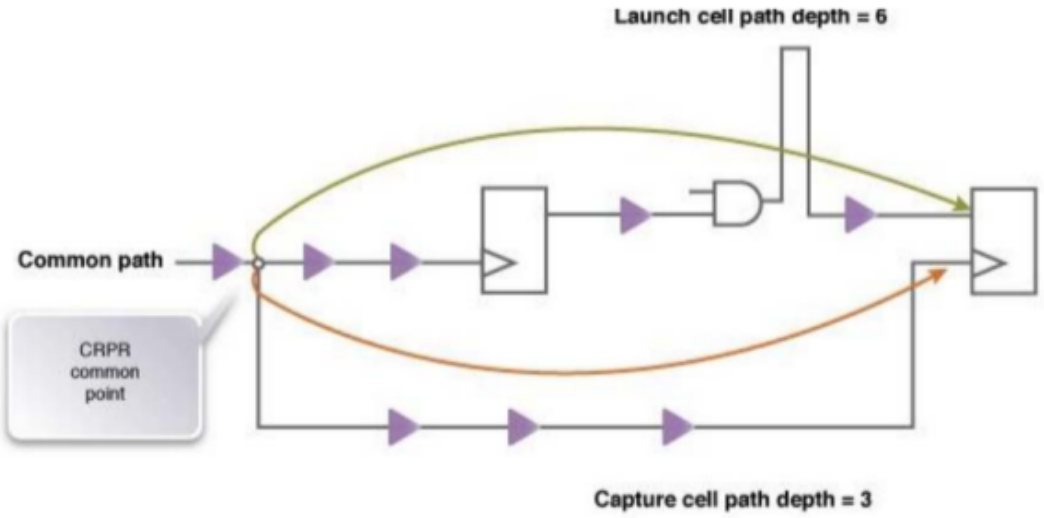
In AOCV we do not have fixed derate value instead based on distance and depth the derate value changes on the chip.

The derate factor increases with increase in the distance as the variations increase along the distance. AOCV based on distance is due to global variations.
The AOCV based on depth is due to local variations. The local variations seem to have better results compared to global variations. So in AOCV as depth increases the derate factor decreases. As path depth increases the AOCV derate decreases due to cancellation of random variations. AOCV has less pessimism than OCV.

Distance based AOCV



Process of advanced on-chip Variations



Depth based AOCV

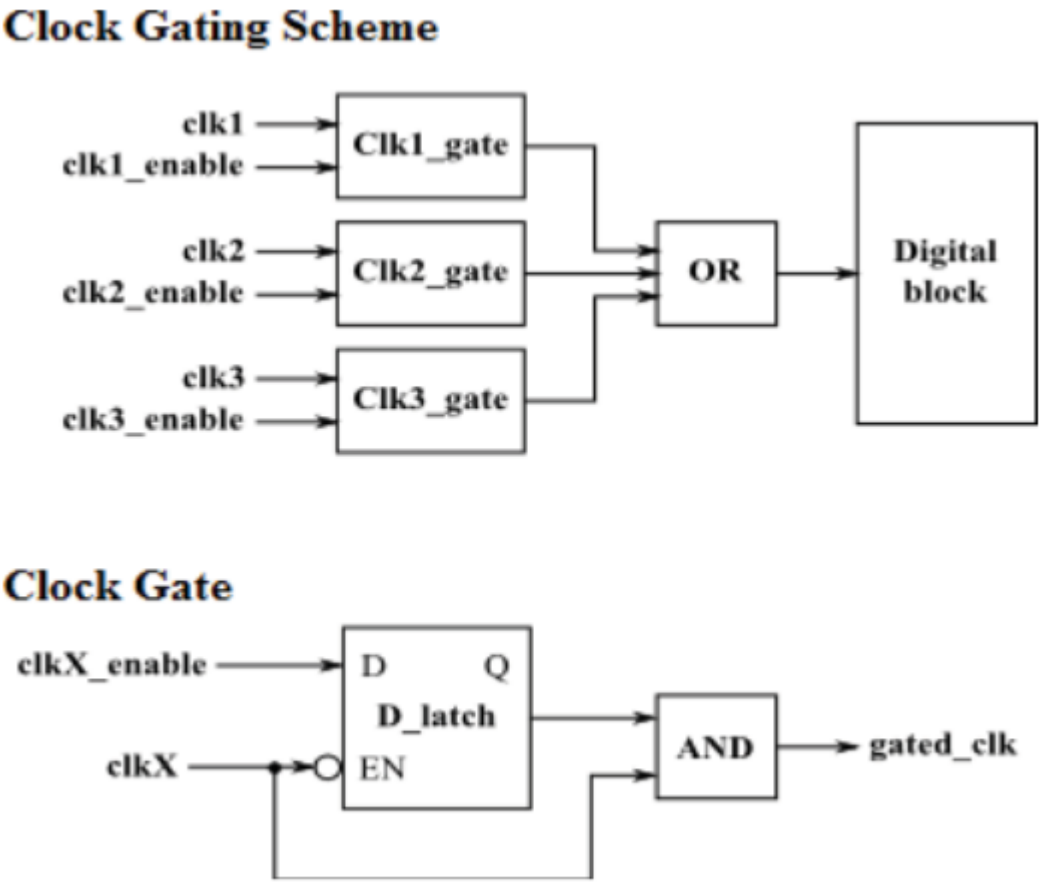
8.1.2 Parametric On Chip Variation (POCV)

In POCV, the cell delay is determined based on the delay variation of the cell, instead of applying a specific derating factor to the cell.
Based on the variation in parameters on a cell the derate value is calculated with the help of normal distribution curve.

POCV uses a normal distribution curve to establish the delay in each path. At the centre of the normal deviation the data is denser and provides the accurate results but at other parts of the deviation the data is less dense and gives less accurate results. Any deviation from the normal distribution curve is obtained using the standard deviation. POCV is less pessimistic compared to OCV and AOCV.

8.2 Clock gating

The clock consumes more power as it switches continuously in the chip. To reduce this power consumption we have a clock gating technique which uses a special circuitry to switch off the switching activity of the clock signal when the flops are in OFF state.
In the below circuit the clock gate cells have a clock signal and as well enable signal so for each gated clock signal it uses one clock gate cell which increases area and power consumption and all these signals are fed to an OR gate. This can be overcome by replacing all the clock gate cells by an AND gate which is fed by a clock signal and an enable signal. When both the inputs are high the signal starts propagating through AND gate else the gate will be in OFF state this reduces the area and power consumption.

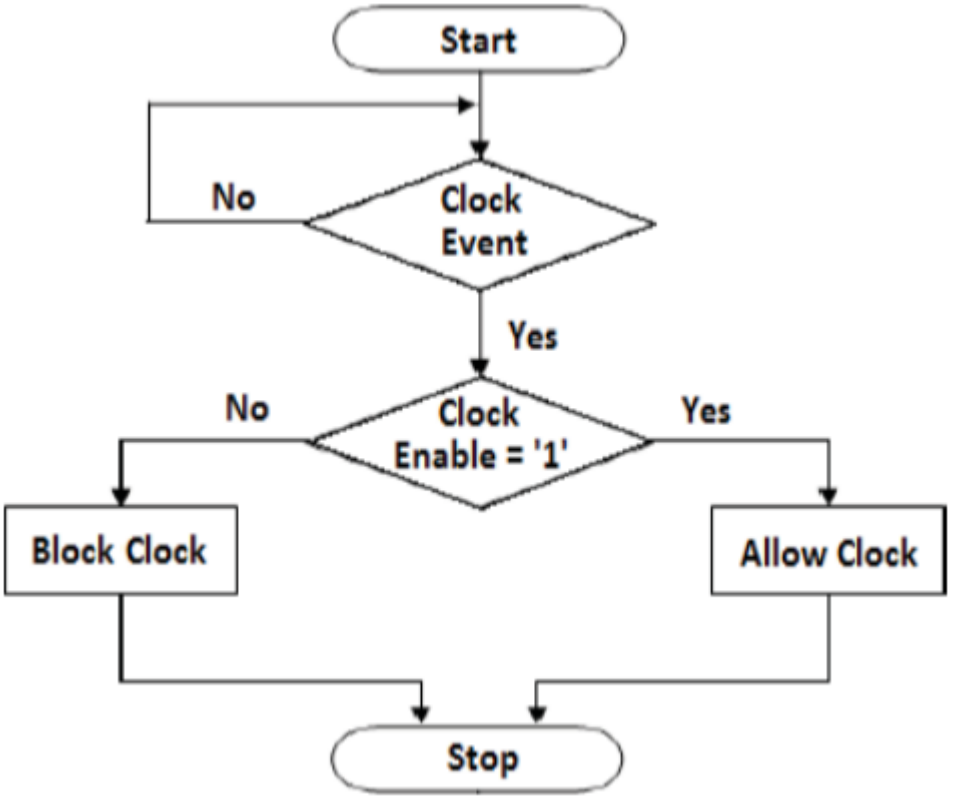


Clock Gating Technique

Some of the clock gating techniques used for dynamic power reduction is:

- Gate based clock gating
- Latch based clock gating
- Flip-flop based clock gating
- Synthesis based clock gating
- Data driven based clock gating
- Auto gated clock gating

8.2.1 Flow chart for clock gating



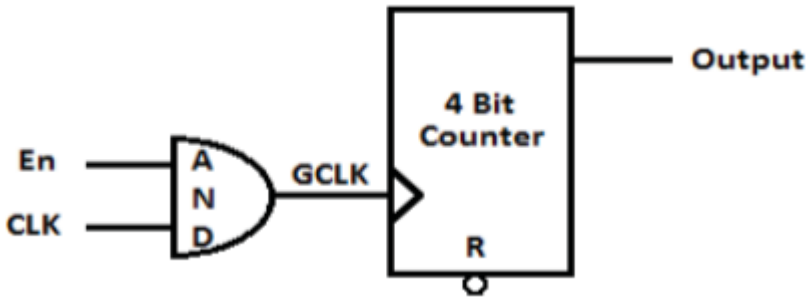
8.2.2 Clock gating techniques

1. **Gate based clock gating**

The gate based clock gating using gates like AND, OR, NOR.

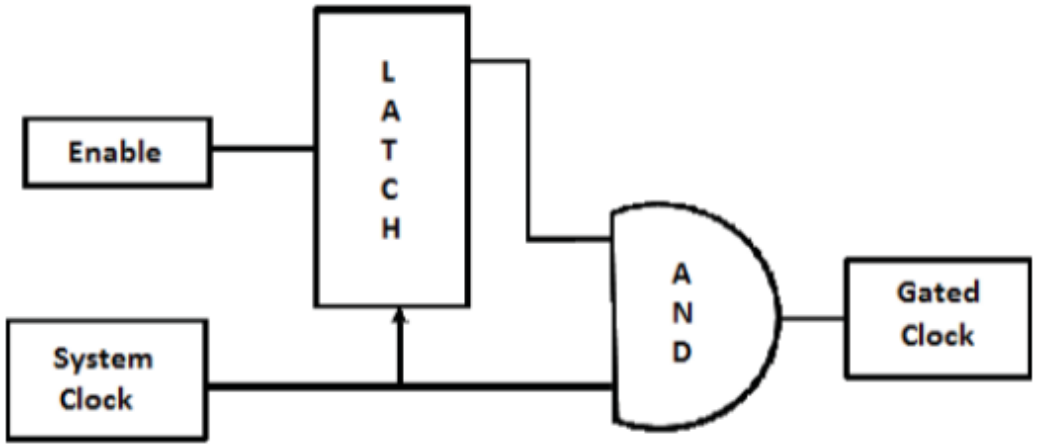
AND gate :

The clock gate cells an AND gate which is fed by a clock signal and an enable signal. When both the inputs are high the signal starts propagating through AND gate else the gate will be in OFF state this reduces the area and power consumption. The AND gate technique can give some errors due to glitches that are obtained due to continuous ON - OFF activity of the cell. The AND gate is mostly used in negative edge triggering. Any changes in level the AND gate cannot identify as the clock edge is considered for any changes to be accepted by an AND gate and this causes glitches.



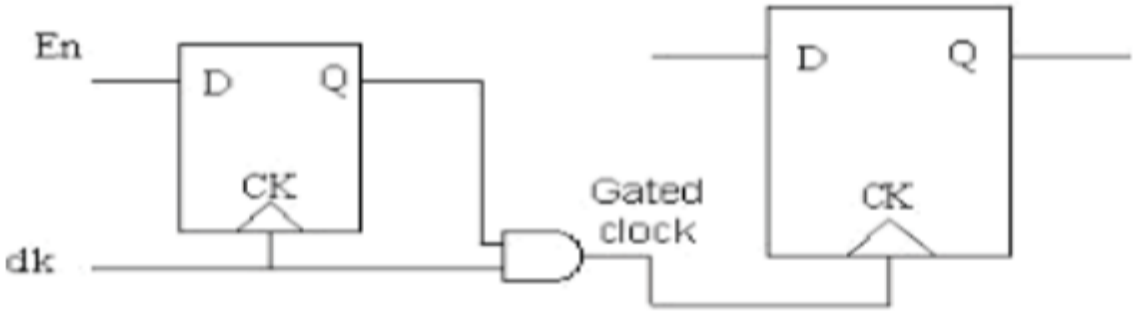
2. **Latch based clock gating**

The drawback of using AND gate can be overcome by latch based clock gating. As the latch is level sensitive any changes in the level is accepted by the AND gate which is followed by a latch.



3. **Flip flop based clock gating**

This uses a D flip flop and an AND gate. The flop is edge triggered and any changes in edge are only captured which results in glitches. The sleep time for flop is more so the charging and discharging time of output capacitor will be more this leads to increase in area and power consumption. So this method is not preferred.



8.2.3 Clock gating checks

Clock gating check occurs when a gating control signal can control the path of a clock signal at a logic cell. If a clock is not acting as a clock after gating the cell then no clock gating check is required. A clock gating check is done only if the clock after the gating cell acts as clock downstream**[1].**

There are two types of clock gating checks:
• Active-high clock gating check: Occurs when the gating cell has an and or a nand function.
• Active-low clock gating check: Occurs when the gating cell has an or or a nor function.

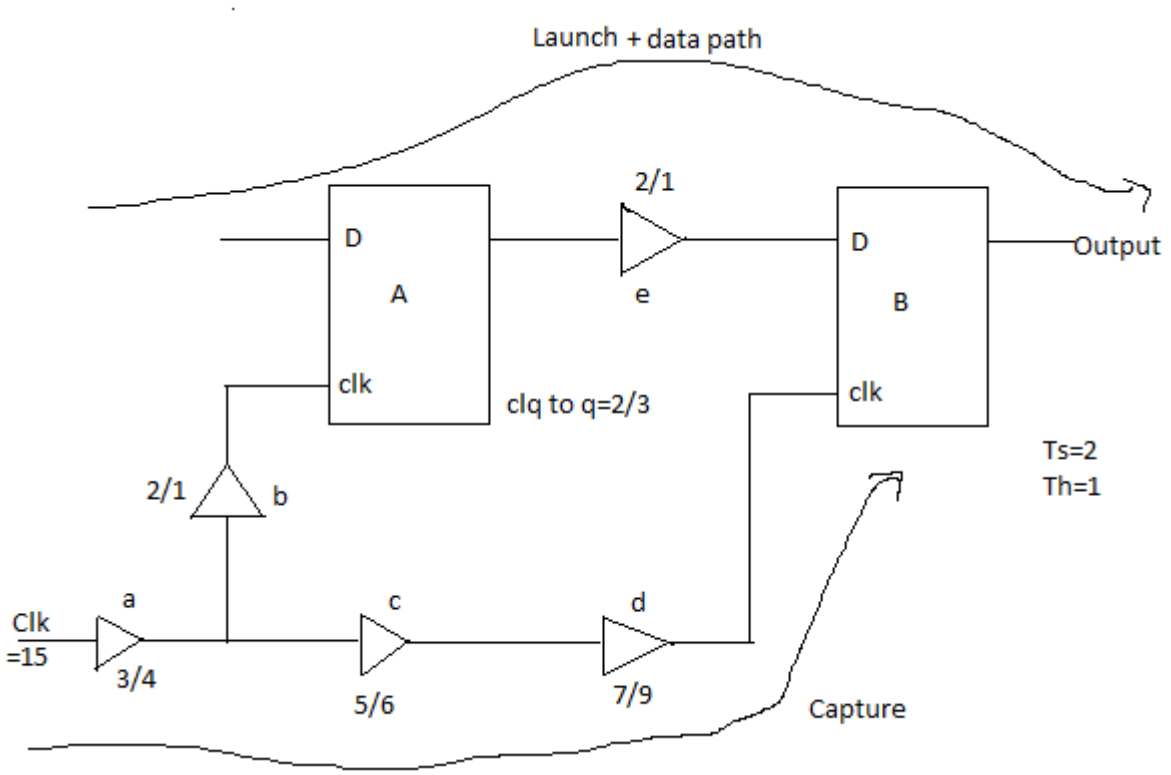
Clock gating check is done using a command “set_clock_gating_check”

Note[1]: Downstream clock usage can be either as a FF clock or it can fanout to an output port or as a generated clock that refers to the output of the gating cell as its master.

Chapter 9 Problems

9.1 Setup and hold slack

Q) For the above given figure find the worst setup slack and worst hold slack?



Sol) Tsetup = datapath + launch path - clockpath

Thold = clock path-[launch path+datapath]

Tcapture = clock propagation delay

Tlaunch = clock period +clock propagation delay on launch path + clock-q delay + combinational delay -Tsetup(+Thold)

Note: For setup take maximum delay values in launch path and minimum delay values in capture path for worst case if Tlaunch-Tcapture meets then it means in all other cases setup slack will meet and the same goes with hold slack.

For hold slack take minimum delay values in launch path and maximum delay values in capture path for worst case.

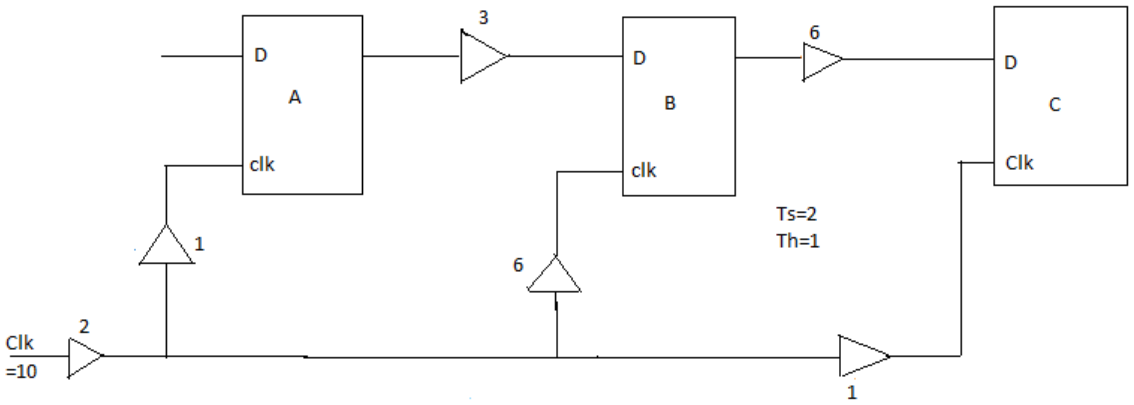
Setup slack: Tlaunch - Tcapture

[15+4+2+3+2-2 = 24ns] - [3+5+7 = 15ns]

Tsetup = 24 - 15 = 9ns

Hold slack: Tcapture - Tlaunch
[4+6+9=19] - [15+3+1+2+1+1= 23ns]
Thold = 19-23 = -4ns

9.2 Useful Skew

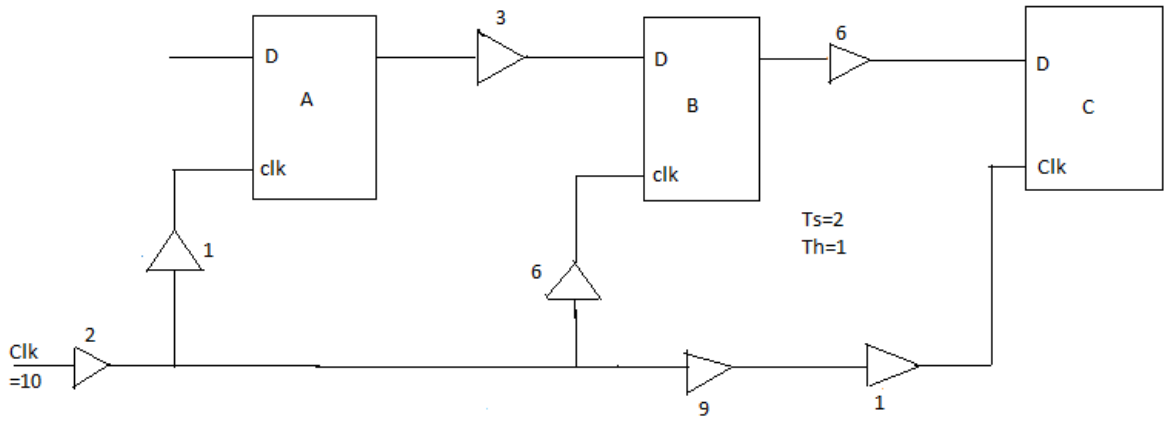


Q) **Adjust the Clock skew in the clock path to meet the timing.**

A) For A to B:
Tl=2+1+3=6
Tc=10+2+6-2=16
Ts=Tc-Tl=16-6=10ns

For B to C:
Tl=2+6+6=14
Tc=10+2+1=13
Ts=Tc-Tl=13-14=-1ns

The difference is 9ns to meet the setup slack from B to C.
So we add a buffer of 9ns in the clock path of C Register.



For A to B:

Tl=2+1+3=6

Tc=10+2+6-2=16

Ts=Tc-Tl=16-6=10ns

For B to C:

Tl=2+6+6=14

Tc=10+9+2+1=22

Ts=Tc-Tl=22-14=8ns