What is physical design ?

Physical design

Physical design means --->> netlist (.v) converted into GDSII form(layout form)
logical connectivity of cells converted into physical connectivity.
During physical design, all design components are instantiated with their geometric representations. In other words, all macros, cells, gates, transistors, etc., with fixed shapes and sizes per fabrication layer, are assigned spatial locations (placement) and have appropriate routing connections (routing) completed in metal layers.

Physical design directly impacts circuit performance, area, reliability, power, and manufacturing yield. Examples of these impacts are discussed below.

1. **Performance**: long routes have significantly longer signal delays.
2. **Area**: placing connected modules far apart results in larger and slower chips.
3. **Reliability:** A large number of vias can significantly reduce the reliability of the circuit.
4. **Power:** transistors with smaller gate lengths achieve greater switching speeds at the cost of higher leakage current and manufacturing variability; larger transistors and longer wires result in greater dynamic power dissipation.
5. **Yield**: wires routed too close together may decrease yield due to electrical shorts occurring during manufacturing, but spreading gates too far apart may also undermine yield due to longer wires and a higher probability of opens

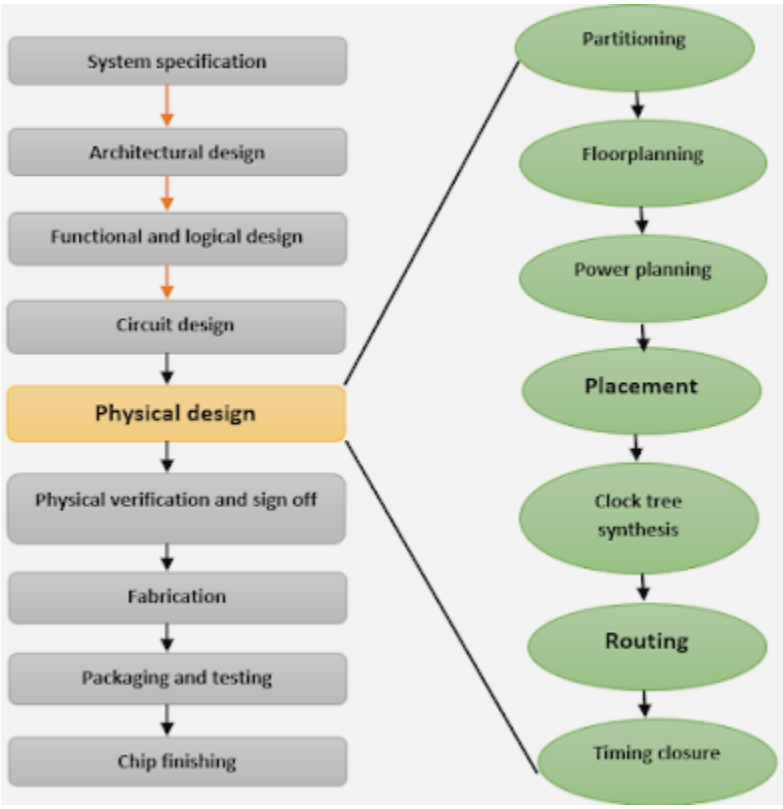
Due to its high complexity, physical design is split into several key steps (Fig. 1).

1. **Partitioning:** breaks up a circuit into smaller sub-circuits or module which can each be designed or analyzed individually.
2. **Floorplanning:** determines the shapes and arrangement of sub-circuits or modules, as well as the locations of external ports and IP or macro-blocks.
3. **Power and ground routing (power planning)**: often intrinsic to floorplanning, distributes power (VDD) and ground (GND) nets throughout the chip.
4. **Placement:** finds the spatial locations of all cells within each block.
5. **Clock network synthesis:** determines the buffering, gating (e.g., for power management) and routing of the clock signal to meet prescribed skew and delay requirements
6. **Global routing:** allocates routing resources that are used for connections; example resources include routing tracks in the channel and in the switch box
7. **Detailed routing**: assigns routes to specific metal layers and routing tracks within the global routing resources.

8. **Timing closure:** optimizes circuit performance by specialized placement or routing techniques .

**GDS- Graphical Data system

1. The *physical design* is the process of transforming a circuit description into the physical layout, which describes the position of cells and routes for the interconnections between them.
2. The main concern is the physical design of VLSI-chips is to find a layout with minimal area, further the total wire length has to be minimized. For some critical nets there are hard limitations for the maximal wire length.



ASIC FLOW AND PHYSICAL DESIGN FLOW

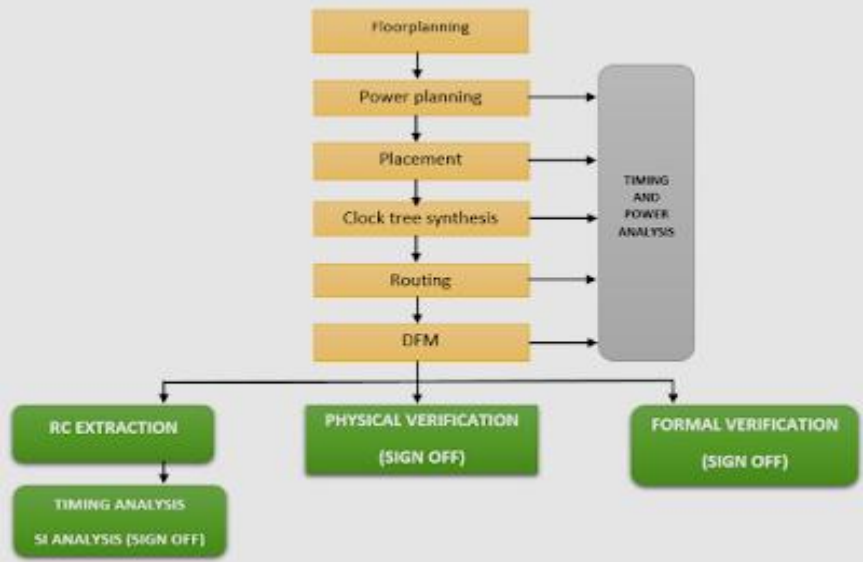
The main steps in the ASIC PHYSICAL DESIGN flow are:

1. Design netlist (after synthesis)
2. Floorplanning
3. Power planning
4. placement
5. Clock Tree Synthesis(CTS)
6. Routing
7. Physical verification
8. GDSII generation

Physical verification:

After physical design is completed, the layout must be fully verified to ensure correct electrical and logical functionality. Some problems found during physical verification can be tolerated if their impact on chip yield is negligible. Therefore, at this stage, layout changes are usually performed manually by experienced design engineers.

- **Design rule checking (DRC):** verifies that the layout meets all technology-imposed constraints. DRC also verifies layer density for chemical-mechanical polishing (CMP).
- **Layout vs. schematic (LVS):** checking verifies the functionality of the design. From the layout, a netlist is derived and compared with the original netlist produced from logic synthesis or circuit design.
- **Parasitic extraction:** derives electrical parameters of the layout elements from their geometric representations; with the netlist, these are used to verify the electrical characteristics of the circuit.
- **Antenna rule checking:** seeks to prevent antenna effects, which may damage transistor gates during manufacturing plasma-etch steps through the accumulation of excess charge on metal wires that are not connected to PN junction node
- **Electrical rule checking (ERC):** verifies the correctness of power and ground connections, and that signal transition times (slew), capacitive loads and fan-outs are appropriately bounded



EDA Tools:

1. **P&R**

   Synopsys: ICCI, ICCII, DC COMPILER
   cadence: Encounter, innovus

2. **TIMING ANALYSIS**

   Synopsys: Primetime
   cadence: tempus

3. **physical verification:**

   Synopsys: Hercules
   cadence: Assura
   mentor: calibre(mostly used)

4. **RC Extraction:**

   Synopsys: StarRCXT

5. **formal verification:**

   Synopsys: formality

Inputs for physical design

Name of Inputs	File format	Given by
Netlist	.v (Verilog)	synthesis team
Synopsys Design Constraints (SDC)	.sdc (written in TCL)	synthesis team
Timing library/logical library	.lib(liberty file)	vendors
Physical library	.lef (layout exchange format)	vendors
Technology file	.techlef /.tf	foundry
TLU+(Table Look Up)	.tlup	foundry

Description of all inputs

Netlist :

- Textual description of circuits components (like logic gates, combinational circuits, sequential circuits), so netlist is a collection of gates.
- It contains the logical connectivity of all the cells.
- It can also be a collection of resistors, capacitors or transistors.

example of netlist:

```
module and_gate(y,a,b);
input a,b;
output y;
AND2 U1(.Y(y), .A(a), .B(b));
Endmodule
```

Synopsys Design Constraints(SDC) :

These are timing constraints and used to meet the timings.
constraints are :

- create clock definition
- generated clock definition
- Virtual clock
- input delay

- output delay
- max delay
- min delay
- max transition
- max capacitance
- max fanout
- clock latency
- clock uncertainty etc..
  and clock exceptions are also present in SDC
- Multicycle Path
- False Path
- Half Cycle Path
- disable timing arcs
- case analysis
  will explain each constraints in detail

Timing library/logical library (.lib):

- It contains timing information of standard cells, soft macros, hard macros.
- It contains Functionality information of standard cells and soft macros.
- Timing information like cell delay setup, hold, recovery, removal are present.
- Design rules like max tran, max cap, max fanout, min cap are present
- Contain power information.
- PVT corners are also present. for every PVT corner the timing of cells is different. hence for every PVT corner there is a .lib file present
  cell delay is a function of input transition and output load and is calculated based on lookup tables.
  cell delays are calculated by Nonlinear Delay Model(NLDM) and composite current source (CCS) models.

CCS(Composite current source)	**NLDM (Non-Linear Delay Model)**
It's like Norton equivalent circuit	It's like Thevenin's equivalent circuits
current source used for driver modeling	Voltage source used for driver modeling
It 20 variables to account input transition and output load	It has only 2 variables
CCS is more accurate	Less accurate
CCS file is 10x times larger than NLDM because of more numbers of variables	Smaller than CCS file
Runtime for CCS is more	Runtime is less

in .lib file following units are present,

- time units
- voltage unit
- leakage power unit
- capacitive load unit
- slew rate
- rise and fall time

for each cell following attributes are present,

- area of cell
- leakage power
- capacitance
- rise and fall capacitance
- for each pin direction and their capacitance

Lookup tables are defined for different parameters like cell delay, hold, setup, recovery, removal with different matrix

```
cell_fall (delay_template_6x6) {
index_1 (“0.015, 0.04, 0.08, 0.2, 0.4, 0.8”);
index_2 (“0.06, 0.18, 0.42, 0.6, 1.2, 1.8”);
    values ( \
        “0.0606, 0.0624, 0.0744, 0.0768, 0.09, 0.098”, \
         “0.1146, 0.1152, 0.1164, 0.1212, 0.1314, 0.1514”, \
         “0.201, 0.2004, 0.2052, 0.2058, 0.2148, 0.2168”, \
         “0.48, 0.4806, 0.4812, 0.4824, 0.4866, 0.4889”, \
         “0.9504, 0.9504, 0.9504, 0.951, 0.9534, 0.975” \
         “0.6804, 0.6820, 0.6836, 0.6851, 0.6874, 0.6895" \);
```

index_1 represents input transition.

#index_2 represents output load i.e output net capacitance.

Ques: what would be the cell_fall time if input_net_transition is 0.08 and the output load is 0.6?

Ans: 0.2058

example of library:

```
cell(OR2_3) {
    area : 6.00
    power:
    rise_time:
    fall_time:
    pin (O) {
       direction : output;
        timing () {
        related_pin : "A";
        rise_propagation() }
        rie_transition()}
        function : "(A||B)";   #functionality
```

```
        max_cap:
        min_cap: }

    pin (A) {
       direction: input;
       cap: ;}
```

Physical Library(.lef) :

- It contains physical information of standard cells, macros, pads.
- Contain the name of the pin, pin location, pin layers, direction of pin(in, out, inout), uses of pin (Signal, Power, Ground) site row, height and width of the pin and cell.
- Contain the height of standard cell placement rows.
- Macros information like cell name, size, dimensions, layout, blockages and capacitance are defined.
- Design rules, via definitions, metal layers and metal capacitances are defined.
- For every technology the via and layer definition are different, so in physical library defined the type of layer(routing/master slice/overlap), width/pitch and spacing, direction, resistance, capacitance, and antenna factor are defined
- Contain preferred routing Directions, minimum width of the resolution

example of lef:

```
layer M2
type routing
 width 0.50;
 end M2
 layer via
   type cut
 end via
 macro AND_1
      origin 0.000
      size 4.5 by 12
      symmetry x y;
      site core;
  pin A
     dir input;
      port
      layer M2
  end
```

lef contains two types of views

1. **CELL view**: it is a full layout of the block and used at the time of tape out.
2. **FRAM view:** this is an abstract view that has only the pins, metals, via and blockages that are used in Placement & Route stages. this makes sure that the interconnection between the pins can be routed automatically and the routing tool will not route over existing metal/via areas otherwise any shorts will come into the picture.

Technology File :

- Contain the number of metal layers and vias and their name and conventions.
- Design rules for metal layers like the width of metal layer and spacing between two metal layers.
- Metal layers resistance and capacitance as well as routing grid.
- Unit, precision, color, and pattern of metal layer and via.
- Maximum current density is also present in the tech file.
- Contains ERC rules, Extraction rules, LVS rules.
- Physical and electrical characteristics of each layer and via.
- It contains nwell, pwell, metal pitch.

tech file should be compatible with both physical & timing libraries

example of tech file:

```
/* specify units and unit values*/
        technology {
                name
                unit
                operating conditions
                routing_rule_models
                        }
 /* define six basic color used to create display colors*/
                [ primary color {
                                primary_color_attributes
                                 } ]
/* define layer specific characteristics including display*/
/* characteristics and layer specific routing design rules*/
/* define layer and data types*/
/* defining vias used in the design */
/* define inter layer ruting design rules */
/* defining cell rows spacing rules */
/* defining density rules*/
/* defining via and slot rules*/
/* defining capacitance ,resistance and temperature coeff. of the layer*/
```

TLU+(Table Lookup) :

- It is a table containing wire cap at diffrent net length and spacing.
- contain RC coeficients for specific technology.
- TLU+ files are extracted or generated from ITF(contains interconnect details) file results.
- The main function of this files are—

[a] . Extracted R, C parasitics of metal per unit length
[b]. These RC parasitics are used for calculating net delays.
[c]. If TLU+ files are not present these R,C parasitics extracted from.ITF files
[d]. For loading of TLU+ we have to load 3 files: 1. TLU+ 2. Min TLU+ 3. Max TLU+
[e]. Map file maps the .itf file and .tf files of the layer and via names.
Milkyway.tf also contain parasitics model of wire as TLU+ contains. If you specify in ICC the TLU+ files then ICC used TLU+ files and did not read parasitics from .tf. if not specified by default ICC will use .tf.

advantage of TLU+
1.more accurate
2.different TLU+ for different RC corners and scenario.
disadvantage of Milkyway.tf ---It used only for one RC corner.

What are the sanity checks before going to start physical design flow

Sanity checks:

To ensure that the input received from the library team and synthesis team is correct or not. If we are not doing these checks then it creates problems in later stages of design.
Basically, we are checking following input files: and make sure that these files are complete and not erroneous.

1. design/netlist checks
2. SDC checks
3. Library checks

Design checks:

Check if current design is consistent or not
It checks the quality of netlist and identifies:
1. Floating pins
2. Multidriven nets
3. Undriven input ports
4. Unloaded outputs

5. Unconstrained pins
6. Pin mismatch counts between an instance and its reference
7. Tristate buses with non-tristate drivers
8. Wire loops across hierarchies

ICC command: **check_design:**

Checks for multi driven nets, floating nets/pins, empty modules.
Pins mismatch, cells or instances without I/O pins/ports etc.

SDC Checks:

1. If any unconstrained paths exist in the design then PNR tool will not optimize that path, so these checks are used to report unconstrained paths
2. Checks whether the clock is reaching to all the clock pin of the flip-flop.
3. Check if multiple clock are driving same registers
4. Check unconstrained endpoints
5. Port missing input/output delay.
6. Port missing slew/load constraints

ICC command: **check_timing**

Library checks:

It validate the library i.e. it checks the consistency between logical and physical libraries.
It checks the qualities of both libraries.
check_library: This command shows the name of the library, library type & its version, units of time, capacitance, leakage power, and current. It shows the number of cells missing, the number of metal or pins missing in the physical and logical library.

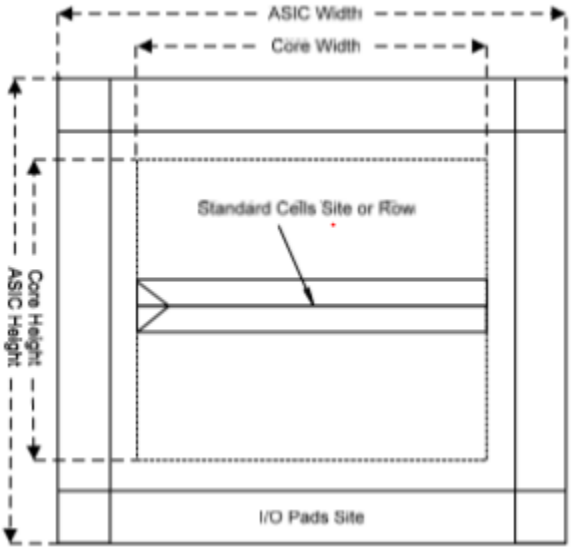
FloorPlan

Floor planning:

Floorplanning is the art of any physical design. A well and perfect floorplan leads to an ASIC design with higher performance and optimum area.
Floorplanning can be challenging in that, it deals with the placement of I/O pads and macros as well as power and ground structure.
Before we are going for the floor planning to make sure that inputs are used for floorplan is prepared properly.

Inputs for floorplan:

1. Netlist (.v)
2. Technology file (techlef)
3. Timing Library files (.lib)
4. Physical library (.lef)
5. Synopsys design constraints (.sdc)
6. Tlu+



After physical design database creation using imported netlist and corresponding library and technology file, steps are

1. Decide core width and height for die size estimation.
2. IO pad sites are created for placement of IO pad placement.
3. Placement of macros.
4. The standard cell rows created for standard cell placement.
5. Power planning (pre routing)

6. Adding physical only cells apart from this aspect ratio of the core, utilization of core area, cell orientation, and core to IO clearance are also taken care of during the floorplan stages.

Floorplan control parameter:

Core area depends upon **:**

1. **Aspect ratio:** Aspect ratio will decide the size and shape of the chip. It is the ratio between horizontal routing resources to vertical routing resources (or) ratio of height and width. **Aspect ratio = width/height**

2. **Core utilization:-** Utilization will define the area occupied by the standard cells, macros, and other cells.If core utilization is 0.8 (80%) that means 80% of the core area is used for placing the standard cells, macros, and other cells, and the remaining 20% is used for routing purposes.

core utilization = (macros area + std cell area +pads area)/ total core area

Pad placement:

In ASIC design three types of IO Pads. Generally pad placement and pin placement is done by Top-Level people. It is critical to the functional operation of an ASIC design to ensure that the pads have adequate power and ground connections and are placed properly in order to eliminate electro-migration and current-switching related problems.

1. Power
2. Ground
3. Signal

What is electromigration?

Electro-migration is the movement or molecular transfer of electrons in the metal from one area to another area that is caused by a high density electric current inflows in the metal. High density current can create voids or hillocks, resulting in increased metal resistance or shorts between wires and it can degrade the ASIC performance.

One can determine the minimum number of ground pads required to satisfy the current limit, and the required number of power pads equal to the number of ground pads.

Where

Ngnd = Itotal/Imax

Ngnd = number of ground pads
Itotal = total current in design (sum of static and dynamic currents)

Imax= max EM current
Current switching noise is generated when there is a transition between states on metal layers. This will cover in crosstalk topic.

Macro placement:

Macros may be memories, analog blocks. Proper placement of macros has a great impact on the quality and performance of the ASIC design. Macro placement can be manual or automatic. Manual macro placement is more efficient when there are few macros to be placed. Manual macro placement is done based on the connectivity information of macros to IO pin/pads and macro to macro. Automatic macro placement is more appropriate if the number of macros is large.

Types of macros:

- **Hard macros**: The circuit is fixed. We can't see the functionality information about macros. Only we know the timing information.

- **Soft macros:** The circuit is not fixed and we can see the functionality and which type of gates are using inside it. Also we know the timing information.

Guidelines to place macros:

- Placement of macros are the based on the fly-lines (its shows the connectivity b/w macro to macro and macro to pins) so we can minimize the interconnect length between IO pins and other cells.
- Place the macros around to the boundary of the core, leaving some space between macro to core edge so that during optimization this space will be used for buffer/inverter insertion and keeping large areas for placement of standard cells during the placement stage.
- Macros that are communicating with pins/ports of core place them near to core boundary.
- Place the macros of same hierarchy together.
- Keep the sufficient channel between macros

channel width = (number of pins * pitch)/ number of layers either horizontal or vertical

Eg. Let's assume If there are two macros having 50 pins and the pitch values are 0.6 and the total number of horizontal and vertical layers are 12. Means M0 M2 M4 M6 M8 M10 are horizontal layers and M1 M3 M5 M7 M9 M11are vertical layers.

Channel width = ((50+50)*0.6)/6
= 10

- Avoids notches while placing macros, if anywhere notches is present then use hard blockages in that area.
- Avoid crisscross connection of macro placement.

- Keep keep-out margin around the four sides of macros so no standard cells will not sit near to Macro pins. This technique avoids the congestion.
- Keep placement blockages at the corners of macros.
- For pin side of macros keep larger separation and for non-pin side, we can abut the macros with their halo so that area will be saved and Halo of two macros can abut so that no standard cells are placed in between macros.
- Between two macros at least one pair of power straps (power and Ground) should be present.

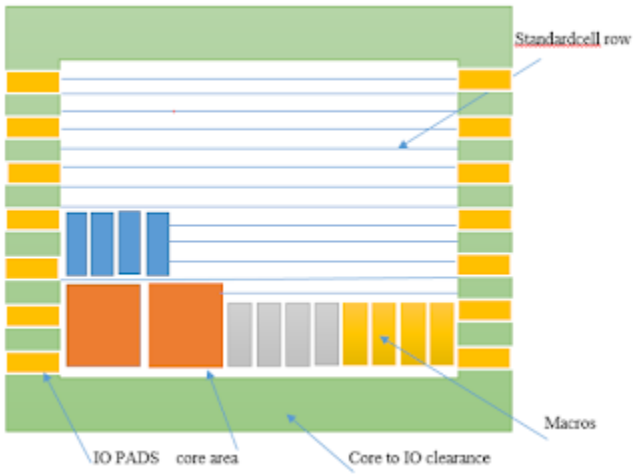


Fig: placement of IO pads, standard cells and macros hierarchy wise

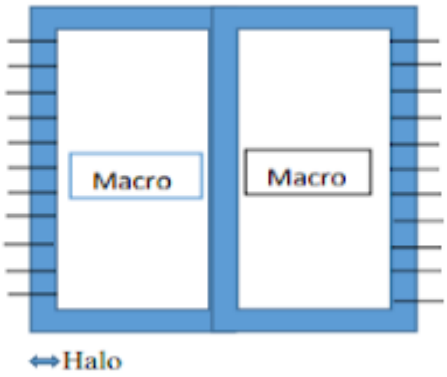


Fig. Abutted non-pin side macros with abutted halo

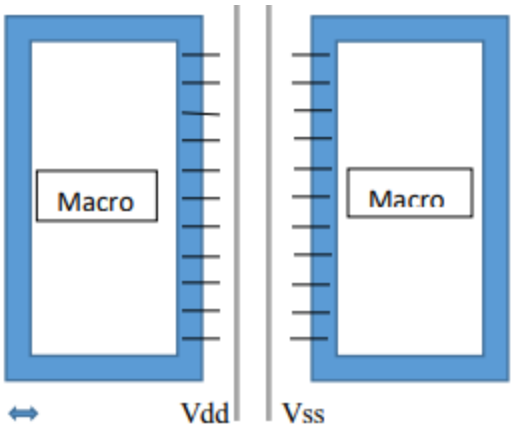


fig: pin side macros with halo

lots of iterations happen to get optimum floorplan. the designer takes care of the design parameter such as power, area, timing and performance during floorplanning.

types of floorplan techniques:

1. **Abutted:-** When the chip is divided into blocks in the abutted design there is no gap between the blocks.
2. **Non abutted:-** In this design there is a gap between blocks. The connection between the blocks is done through the routing nets.
3. **The mix of both:** This design is a combination of abutted and non- abutted.

outputs of floorplan:

1. get core and boundary area
2. IO ports/pins placed
3. macros placement done
4. floorplan def file

<u>key terms related to floorplan:</u>

standard cell row:

- The area allotted for the standard cells on the core is divided into rows where standard cells are placed.
- The height of the row is equal to the height of the standard cell and width varies. The height varies according to multiple standard cell row height. there may be double-height cells, triple-height cells, etc.
- The standard cells will sit in the row with proper orientation.

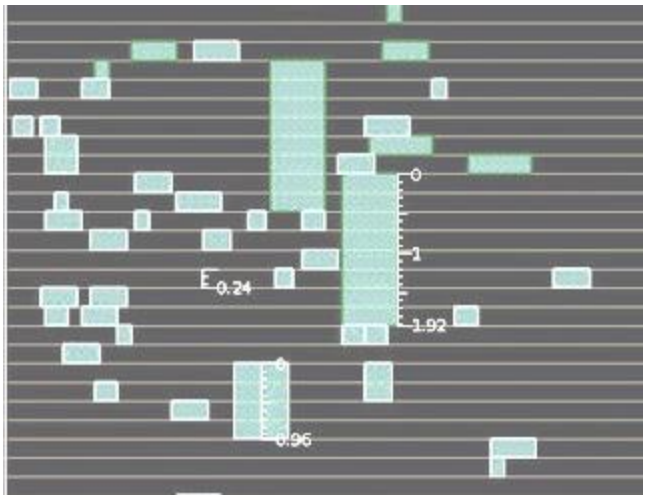


Fig: placement of standard cells in row

fly lines:

- macros are placed manually using fly lines. fly lines are a virtual connection between macros and macros to IO pads.
- This helps the designer about the logical connection between macros and pads.
- Fly lines act as guidelines to the designer to reduce the interconnect length and routing resources.
- fly lines are of two types:

macros to IO pin:

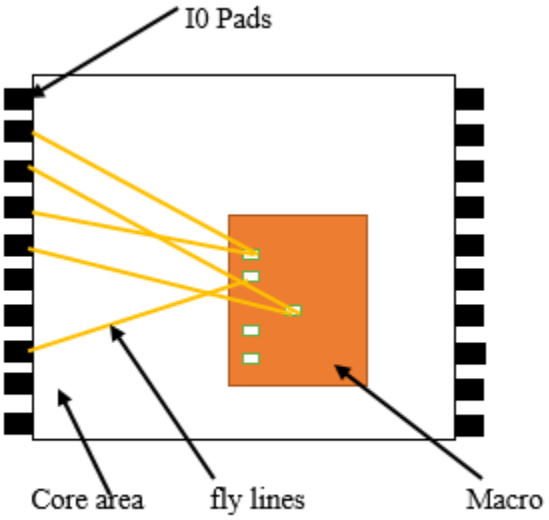


Fig a: macro to IO ports fly lines

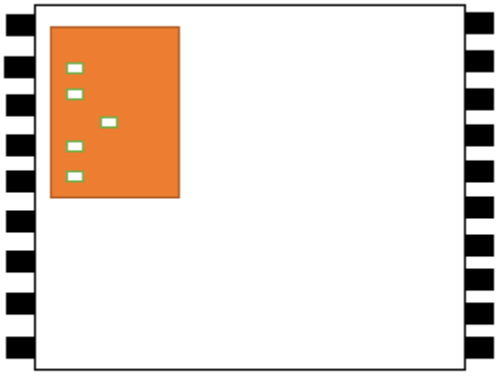


Fig b: macro placed at the core boundary

macros to macros fly lines:

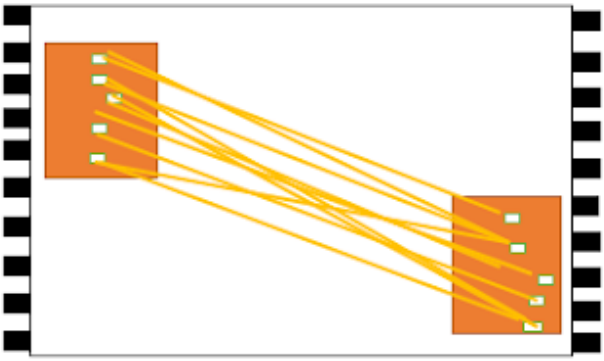


Fig a: macro to macro fly lines

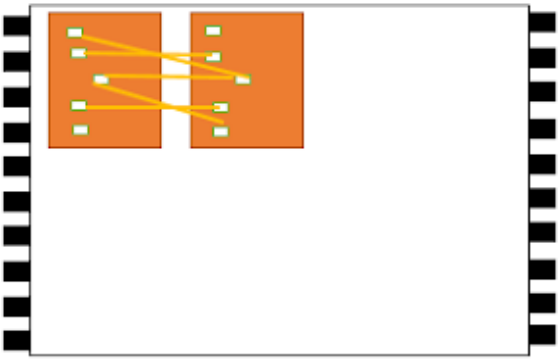
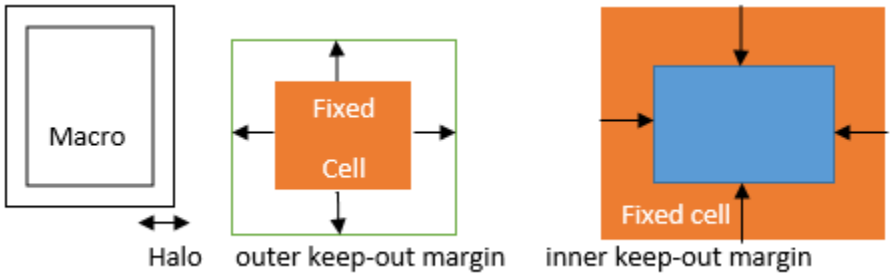


fig b: macros placed near to each other

halo(keep out margin):

- This is the region around the fixed macros so that no other macros and standard cell can be placed near to macros boundary
- the width of the keep out margin on each side of the fixed cell can be the same or different depending on how you define keepout margin.
- keeping the placement of cells out of such regions avoids congestion and produce better qor.
- Halo of two adjacent macros can be overlap.
    - If the macros moved from one place to another place, the halo **will also move**.



ICC command for keepout margin:

create_keepout_margin -outer {10 10 10 10} macro_name -type soft/hard
create_keepout_margin -inner {10 10 10 10} macro_name -type soft/hard

Blockages :

1. Blockages are the specific location where the placing of cells is blocked.
2. Blockages will not guide the tool but it allows not to place the standard cells buffer and inverters at some particular area i.e by using blockages we blocked the area so no standard cells and other cells won't be placed.
3. If the macros moved from one place to another place, blockages **will not move.**
4. Blockages are of three types. a) soft b) hard c)partial

soft blockages:

- only buffer can be placed.
- prevents from the placement of std cell and hard macro within the specified area during coarse placement but allows placement of buffer/inv during optimization, legalization and clock tree synthesis.

- **create_placement_blockages -boundary {{10 20} {100 200}} -name PB1 -type soft**

hard blockages:

- No standard cells, macros and buffer/inv can be placed within the specified area during coarse placement, optimization, and legalization.
- Used to avoid routing congestion at macros corners.
- Control power rails generation at the macros.

c**reate_placement_blockages -boundry {{10 20} {100 200}} -name PB1 -type hard**

partial blockages:

- Partial blockages limit the cell density in the specified area.
- By default the blockage factor is 100% so no cells can be placed in that region but if we want to reduce density without blocking 100% area, we can change the blockage factor.

create_placement_blockages -Boundry {10 20 100 200} -type partial blocked_percentage 40

Partial blockages with a maximum allowed cell density of 60% (blocked % is 40) enclosed by a rectangle with corners (10 20) & (100 200)

- To allow unlimited usage of a partial blockage area specify a blockage percentage to zero.

ques. how can you say the floorplan is good?

ans: a good floorplan should meet the following constraints

1. minimize the total chip area
2. routing should be easy

---

1. what is floorplanning? how can you say that your floorplan is good?
2. what are the input and outputs of the floorplan?
3. what are the floorplan control parameters?
4. how will you determine the distance between the two macros?
5. what are the guidelines for placing the macros?
6. what are the steps needed to be taken care of while during floorplanning?
7. what are the issues you will face if floorplan is bad? ... congestion and timing
8. How can you estimate the area of the block?
9. how will you decide the pin location in block-level design?
10. what are blockages, halo, and keepout margins?

PHYSICAL ONLY CELLS

PHYSICAL ONLY CELLS:

These cells are not present in the design netlist. if the name of a cell is not present in the current design, it will consider as physical only cells. they do not appear on timing paths reports.they are typically invented for finishing the chip.

Tap cells:

- A tap cell is a special nonlogic cell with a well tie, substrate tie, or both.
- Tap cells are placed in the regular intervals in standard cell row and distance between two tap cells given in the design rule manual.
- These cells are typically used when most or all of the standard cells in the library contain no substrate or well taps.
- Generally, the design rules specify the maximum distance allowed between every transistor in a standard cell and a well or substrate tap.
- Before global placement (during the floorplanning stage), you can insert tap cells in the block to form a two-dimensional array structure to ensure that all standard cells placed subsequently comply with the maximum diffusion-to-tap distance limit.

command in ICC: create_tap_cells, specify the name of the library cell to use for tap cell insertion (-lib_cell option) and the maximum distance, in microns, between tap cells (-distance option)

- icc2_shell>**create_tap_cells -lib_cell myreflib/mytapcell -distance 30**

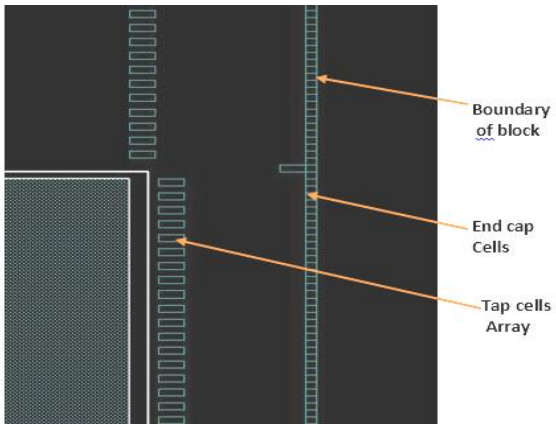


fig: TAP Cells and END CAP Cells

What is latch-up the problem:

It is the condition when low impedance path gets formed between VDD and GND terminal and there is direct current flow from VDD to GND which might result in a complete failure of chip. while the formation of CMOS INVERTER we saw the formation of PN junctions and because of these PN junctions there may be formation of parasitics elements like diode and transistors.

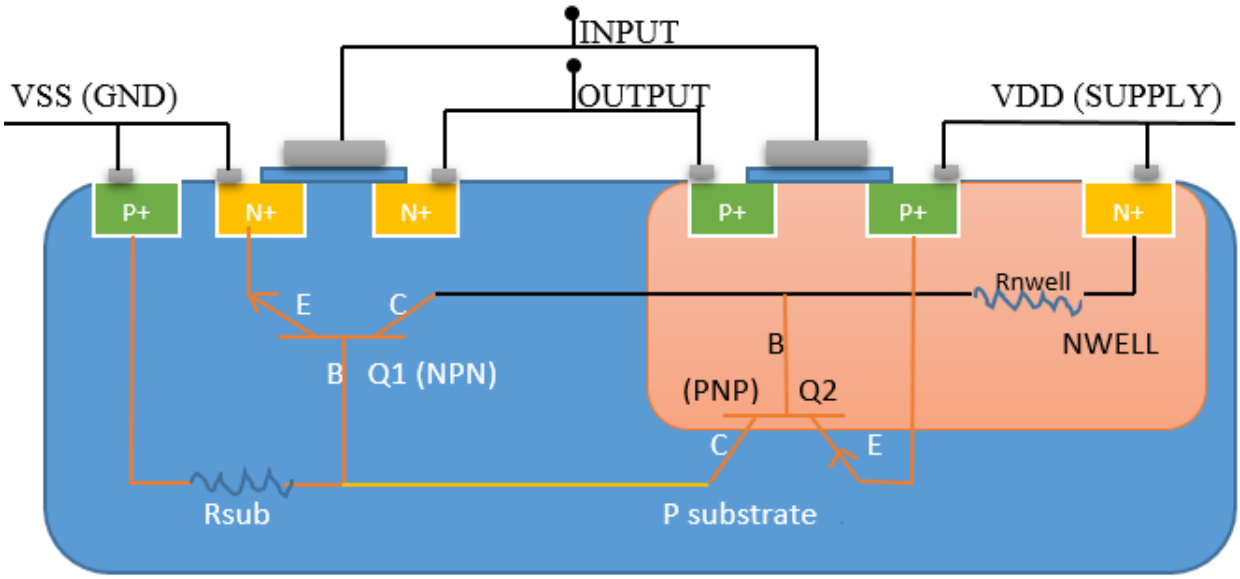


Fig: latch-up phenomenon

Transistors Q1(NPN) and Q2(PNP) are parasitics transistors that are getting formed during the manufacturing of CMOS inverter. If these two parasitic transistors are in on condition then current starts flowing from VDD to VSS and creates a short circuit. while manufacturing these devices the designer made sure that all PN junction should be in reverse bias so that no parasitic transistor will turn on and hence the normal operation will not be affected. but sometimes what happened because of external elements (like input and output) the parasitic transistors get turned on. for parasitics transistor gets turned on there are two scenarios::

1. **when the input and output > VDD: PNP transistor in ON condition:** because now P region is more positive than N region in Nwell, therefore Base-Emitter junction of PNP (Q2) transistor is in Forward biased and now this transistor will turn on. Now if we see in the fig the collector of PNP transistor is connected to the base of NPN transistor, because of this connection the current is flowing from collector (PNP) to base (NPN) and then because of this base current the NPN transistor gets turn on and the current flowing from VDD to VSS through these two parasitics transistors. This current is flowing even if we removed the external inputs and outputs and parasitic transistors make a feedback path in which current is latched up and creates a short circuit path.

2. **when input and output <VSS: NPN transistor in ON condition: N**ow N region is more negative than P region in P substrate, therefore Base-Emitter junction of NPN (Q1)

transistor is in Forward biased and now this transistor will turn on. Now if we see in the fig the Base of NPN transistor is connected to the Collector of PNP transistor, because of this connection the current is flowing from Base (NPN) to Collector (PNP) and then because of this Collector current the PNP transistor gets turn on and current flowing from VDD to VSS through these two parasitics transistors. This current is flowing even if we removed the external inputs and outputs and parasitic transistors make a feedback path in which current is latched up and creates a short circuit path.
In the fig shown above, the value of Rnwell and Rpsub resistance is quite high, if the values of these resistances will reduced then what will happen? The current flowing from the collector of PNP transistor will flow from these resistance paths, i.e current find the low resistance path to reach from VDD to VSS and NPN transistor will never get turn on and in this way, latchup problem will not occur.

Solution for latch-up problem:
Reducing the resistance values: tap the Nwell to VDD and Psubstrate to GND externally.

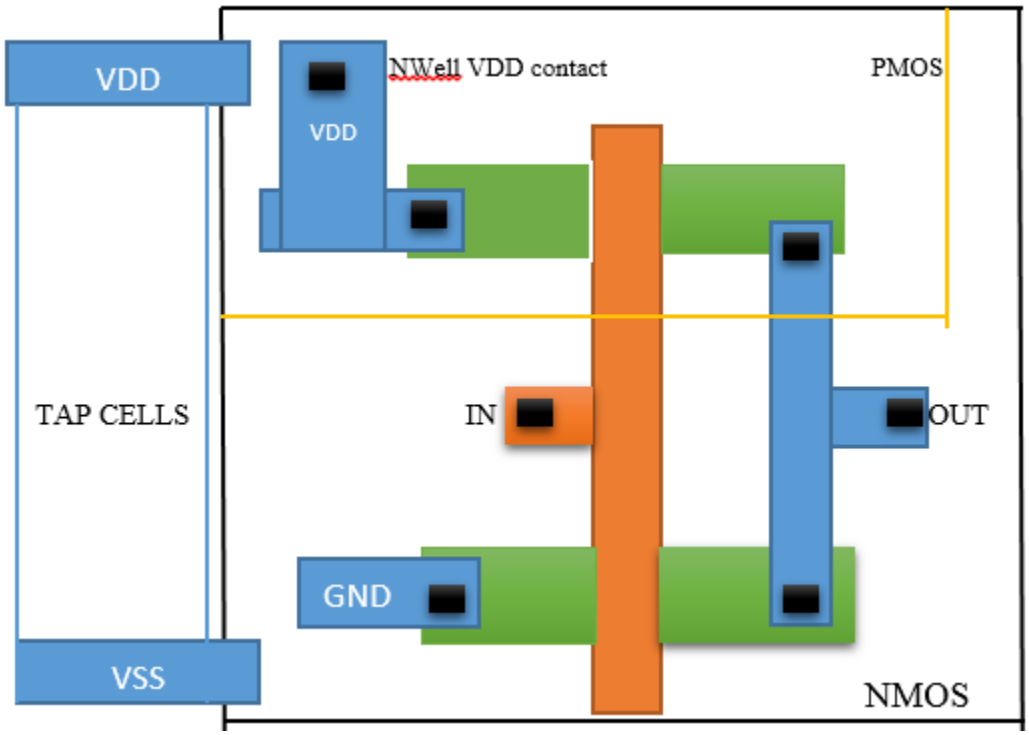


fig: tapping the VDD with Nwell and VSS with Psub externally

Tie cells:

- These are special-purpose cells whose output is constant high or low. The input needs to be connected to the gate of the transistor and there are only two types of input logic 1 and logic 0, but we do not connect them directly to gate of the transistor as with supply glitches can damage the transistor so we used tie high and tie low cells (these are nothing but resistors to

make sure that PG network connected through them) and output of these cells are connected to the gate of the transistor.

- There will be floating nets because of unused inputs they should be tie with some value either low or high to make them stable.

- Insert the tie cells manually also by command connect_tie_cells, this command insert tie cells and connect them to specified cell ports.

why tie cells are inserted?

The gate oxide is very thin and it is very sensitive to voltage fluctuations. If the Gate oxide is directly connected to the PG network, the gate oxide of the transistor may get damaged due to voltage fluctuations in the power supply. To overcome this problem tie cells are used.

How the circuit looks like and how it will work:

Tie high cells: initially we directly connect VDD to the gate of transistor now we connect the output of these cells to the gate of the transistor if any fluctuations in VDD due to ESD then PMOS circuit pull it back to the stable state. PMOS should be ON always, the input of the PMOS transistor is coming from the output of NMOS transistor and here in NMOS gate and drain are shorted and this is the condition of saturation (NMOS) and NMOS will act as pull-down and always give a low voltage at the gate of PMOS. now PMOS will on and gives stable high output and this output is connected to the gate of transistor

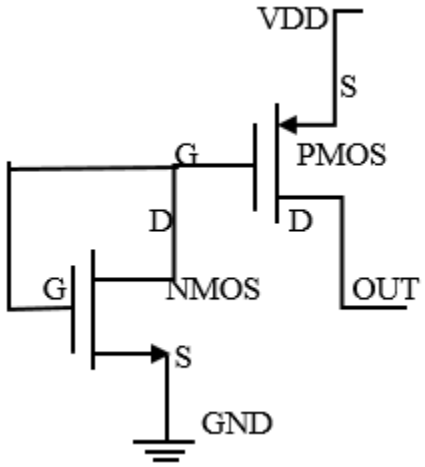


fig: TIE high cells

Tie low cells: initially we directly connect VDD to the gate of transistor now we connect the output of these cells to the gate of the transistor, if any fluctuations in VDD due to ESD (electrostatic discharge) then NMOS circuit pull down it back to the constant low stable state. in fig the gate and drain of PMOS transistor are shorted and hence this is on saturation region and it acts as pull up resistor and it always gives high voltage to the gate of NMOS transistor and because of this high voltage the NMOS transistor will be ON all the time and we get stable low output because it acts as pull-down transistor and this output is connected to the gate of the transistor.

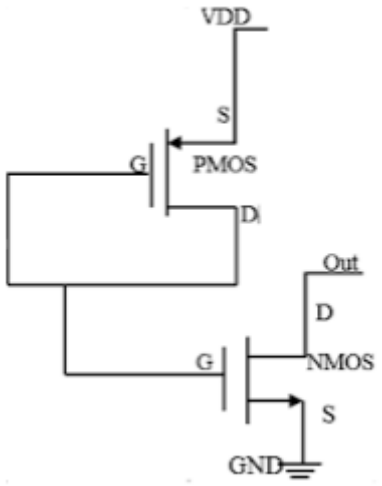


fig: TIE low cells

Filler Cells

- To ensure that all power nets are connected, you can fill empty space in the standard-cell rows with filler cells.
- Filler cells have no logical connectivity. these cells are provided continuity in the rows for VDD and VSS nets and it also contains substrate nwell connection to improve substrate biasing.
- Filler cell insertion is often used to add decoupling capacitors to improve the stability of the power supply and discontinuity in power.
- The IC Compiler II tool supports filler cells with and without metal and supports both single-height and multi-height filler cells.

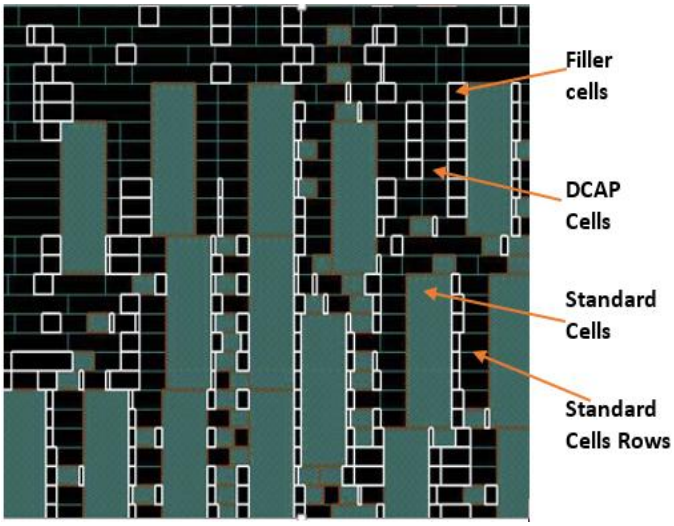


fig2: Filler cells, Decap cells

why we need continuity for nwell and implantation :

If there is continuity b/w nwell and implant layer it is easier for foundry people to generate them and the creation of a mask is a very costly process so it is better to use only a single mask. If nwell is discontinuous the DRC rule will tell that place cells further apart i.e maintain the minimum spacing because there is a **well proximity effect.**

we know nwell is tap to VDD and P substrate is tap to VSS to prevent latchup problem. now if there is a discontinuity in nwell it will not find well tap cells, so we have placed well tap cells explicitly, therefore it will increase the area explicitly, hence we have filler cells so no need to place well tap cells.

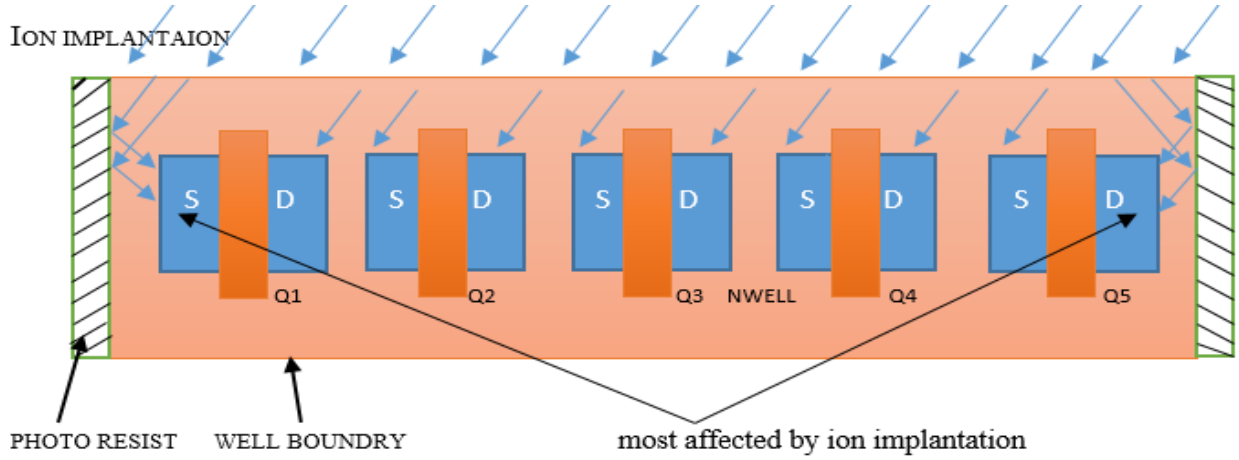
when we placed filler cells in the design:

when optimization of clock tree synthesis is completed i.e after timing has been met because let's say if we want to place buffer/inv for optimization purpose we can't place these cells because there is already placed filler cells, and enough area is not there to present buffer/inv, so after timing has been met and routing optimization is done then only placed the filler cells to fill the emptyspace.

Well proximity effect:

During the manufacturing of chips, we will get these types of problems, that's why they are second-order effects. In this, the transistors that are close to the well edge have different performance than ideally placed transistors, because of this effect the transistor speed can vary by+-10%.The transistor placed to the well boundary so it will get many problems during ion implantation. Implanted ion is coming to the well boundary and reflected/scatter from the well

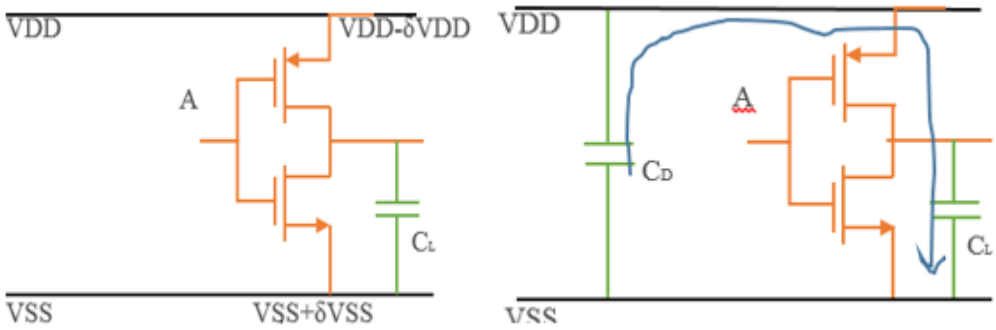
boundary to transistors Q1 & Q5 boundary and ions are deposited on the Q1 Q5 boundary. Ion particles are scattered/reflected due to photoresist on both side of nwell wall. these ions are deposited only those transistors who are near to the well boundary, so any one of the terminals of transistors gets affected by ion implantation and the rest of the transistor will get uniform ions.



well proximity effect

Decap cells:

- These are typically poly gate transistors where source and drain are connected to the ground rails and the gate is connected to the power rails. **(fig2)**
- The gate in a transistor consumes most of the power(dynamic) only at the active edge of the clock at which most of the sequential elements are switching.
- No voltage source is perfect hence glitches are produced on the power line due to this huge current drawn from the power grid at the active edge of the clock.
- We already know that power and ground rails are made up of metal and metal have its own resistance and inductance, when current is flowing through these metal wires there is some voltage drop.
- If the power source is far away from a flip flop the chances are that this flop can go into the metastable state due to this IR drop because it will not get sufficient voltage to work properly.
- Decap filler cells are small capacitors which are placed between VDD and GND all over the layout when the logic circuit draw a high amount of current, this capacitor provides extra charge to that circuit. when logic circuit not drawing any current, the de-cap is charged up to maximum capacitance.
  we know V= IR + L di/dt.
  V= IR (voltage drop due to the metal layer)
  V= Ldi/dt (voltage drop due to change in current with respect to time)



now if the voltage drop is greater than Noise margin(NM) of the inverter then it might lead to improper functionality. Let's say NM is 0.4 now 1-0.4=0.6v (anything less than 0.6 will consider as 0 by this inverter)

now let's say voltage drop is 0.5 i.e 1-0.5=0.5v now this will consider as 0 therefore it will lead to improper functionality. now we have seen voltage drop > NM so here de-cap cells comes into picture so these cells are capacitor that holds the reservoir of charges and these cells are placed near to the power-hungry cells, so whenever these cells switch the de-cap cells starts discharging themselves and provided the required voltage to the particular cells and keep the power supply constant to all the elements in the design.

With increasing clock frequency and decreasing supply voltage as technology scales, maintaining the quality of power supply becomes a critical issue. Typically, decoupling capacitors (de-caps) are used to keep the power supply within a certain percentage (e.g., 10%) of the nominal supply voltage. De-caps holds a reservoir of charge and are placed close to the power pads and near any large drivers. When large drivers switch, the de-caps provide instantaneous current to the drivers to reduce IR drop and Ldi/dt effects, and hence keep the supply voltage relatively constant. A standard de-cap is usually made from NMOS transistors in a CMOS process. At the 90nm technology node, the oxide thickness of a transistor is reduced to roughly 2.0nm. The thin oxide causes two new problems: possible electrostatic discharge (ESD) induced oxide breakdown and gate tunneling leakage. Potential ESD oxide breakdown increases the likelihood that an integrated circuit (IC) will be permanently damaged during an ESD event and hence raises a reliability concern. Higher gate tunneling leakage increases the total static power consumption of the chip. As technology scales further down, with a thinner oxide, the result is an even higher ESD risk and more gate leakage. The standard de-cap design experiences these two problems for more please visit the link below.

Disadvantages of Decap cells:

- These are very leaky.
- The interaction of de-cap cells with package RLC network .since the die is essentially a capacitor with small R, L and package having huge RL network, more de-cap cells placed the more chances of tunning the circuit is into its resonance frequency, since both VDD and GND will be oscillating, many of the designers are fails because of this.

$$w_o = \frac{1}{\sqrt{LC}}$$

Boundary cells (End cap cells):

- Before placing the standard cells, we can add boundary cells to the block. Boundary cells consist of end-cap cells, which are added to the ends of the cell rows and around the boundaries of objects such as the core area, hard macros, blockages, and voltage areas, and corner cells, which fill the empty space between horizontal and vertical end-cap cells. **(fig1)**
- End-cap cells are typically nonlogic cells such as a decoupling capacitor for the power rail. Because the tool accepts any standard cell as an end-cap cell, ensure that you specify suitable end-cap cells.
- Boundary cells include both end-cap cells placed on the left, right, top, and bottom boundaries, and inside and outside corner cells and.
- To know the end of the row and at the edges, endcap cells are placed to avoid the cell's damages at the end of the row to avoid the wrong laser wavelength for correct manufacturing.
- Boundary cells protect your design from external signals.
- These cells ensure that gaps do not occur between the well and implant layer and to prevent from the DRC violations.
- When you insert these at the end of the placement row, these will make sure that these cells properly integrated to the design and will have a clean unwell, other DRC clean, hence next block will abut without any issues.

fig: end cap cells

POWER PLANNING

POWER PLANNING

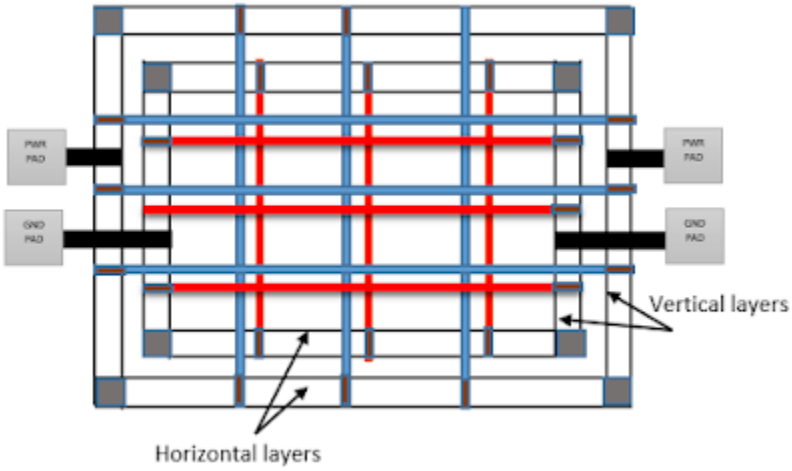
Before 2000 area, delay and performance were the most important parameters, if anyone design circuit the main focus was on how much less area is occupied by the circuit on the chip and what the speed is. Now situation is changed, the performance and speed is a secondary concern. In all nanometer (deep sub-micron) technology power becomes the most important parameter in the design. Almost all portable devices run on battery power. Power consumption is a very big challenge in modern-day VLSI design as technology is going to shrinks Because of

1. Increasing transistors count on small chip
2. Higher speed of operations
3. Greater device leakage currents

Grid structure:

Power planning means to provide power to the every macros, standard cells, and all other cells are present in the design. Power and Ground nets are usually laid out on the metal layers. In this create power and ground structure for both IO pads and core logic. The IO pads power and ground buses are built into the pad itself and will be connected by abutment.

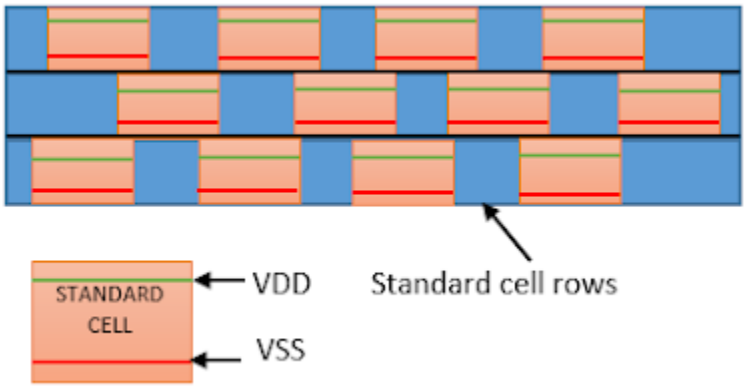
For core logic there is a core ring enclosing the core with one or more sets of power and ground rings. The next consideration is to construct cell power and ground that is internal to core logic these are called power and ground stripes that repeat at regular intervals across the logic or specified region, within the design. Each of these stripes run both vertically and horizontally at regular interval then this is called power mesh.



Ring and core power ground

The total number of stripes and interval distance is solely dependent on the ASIC core power consumption. As power consumption (static and dynamic) increases the distance of power and

ground straps interval increase to reduce overall voltage drop, thereby improving performance. In addition to the core power and ground ring, macro power and ground rings need to be created using vertical and horizontal metal layers. A macro ring encloses one or more macros, completely or partially with one or more sets of power and ground rings.
It is strongly recommended to check for power and ground connectivity after construction of the entire PG network.
Standard cell are placed in a row since they are placing side by side they are touching each other, and horizontal routing across the cells already done so you don't have to anything it's like a continuous line. That's why we are placing filler cells if any gap is thereafter optimization.



standard cells are placed in the rows

NOTE: as you are going to deep sub-micron technologies the numbers of metal layers are increased like in 28 nm technology we have only 7 metal layer and for 7nm technology, we have 13 metal layers, and one more thing as we are going to lower metal layers to higher metal layers the width of the layers is increasing. So better to use higher layers for the power line. They can carry more current.
Power planning involves

- Calculating number of power pins required
- Number of rings, stripes
- Width of rings and stripes
- IR drop

The calculation depends on the power consumed by chip, total current drawn, current rating of power pad (A/m), current rating of metal layers, and the percentage of current flowing through metal layers.

Inputs of power planning

- Netlist(.v)
- SDC
- physical and logical libraries (.lef & .lib)
- TLU+

- UPF

Level of power distribution

1. **Rings**: carries VDD and VSS around the chip
2. **Stripes**: carries VDD and VSS from the ring
3. **Rails**: connect VDD and VSS to the standard cell's VDD and VSS

POWER PLANNING MANAGEMENT

Core cell power management:

- In this power rings are formed around the core.
- If any macros/IP is power critical than create separate power ring for particular macro/IP.
- Number of power straps are created based on the power requirement.

I/O cell power management:

In this power rings are formed for I/O cells and trunks are created between core and power rings and power pads.

Ideal power distribution network has the following properties:

- Maintain a stable voltage with little noise
- Avoids wear out from EM and self-heating
- Consume little chip area and wiring
- Easy to layout

Real networks must balance these competing demands, meeting targets of noise and reliability. The noise goal is typically ±10%; for example, a system with a nominal voltage Vdd= 1.0 V may guarantee the actual supply remains within 0.9-1.1V. The two fundamental sources of power supply noise are IR drop and L di/dt noise.

IR Drop:

The power supply in the chip is distributed uniformly through metal layers across the design and these metal layers have their finite amount of resistance. When we applied the voltage the current starts flowing through these metal layers and some voltage is dropped due to that resistance of a metal wire and current. This drop is called IR drop. Because of IR drop, delay will increase and it violates the timing. And this will increase noise and performance will be degraded.

The value of an acceptable IR drop will be decided at the start of the project and it is one of the factors used to determine the derate value. If the value of the IR drop is more than the acceptable value, it calls to change the derate value. Without this change, the timing calculation becomes optimistic.

For example, a design needs to operate at 1.2 volts and it has tolerance (acceptable IR drop limit) of ±0.4 volts, so we ensure that the voltage across VDD and VSS should not fall below 0.8v and

should not rise above 1.6v. If the drop will be in this limit it does not affect timing and functionality

Power routes generally conduct a lot of current and due to this IR drop will become into the picture so we are going to higher layers to make these routes less resistive. For power routing top metal layers are preferred because they are thicker and offer less resistance.

For example, if there are 15 metal layers in the project then top two or three layers used by hierarchy people(top level or full chip) and metal 11 and metal 12 used for power planning purposes.

There are two types of IR drop

1. Static IR drop
2. Dynamic IR drop

Static IR drop:

This drop is independent of cell switching. And this is calculated with the help of metal own resistance.

Methods to improve static IR drop

1. We can go for higher layers if available
2. Increase the width of the straps.
3. Increase the number of wires.
4. Check if any via is missing then add more via.

Dynamic IR drop:

This drop is calculated with the help of the switching of cells. When a cell is switching at the active edge of the clock the cell requires large current or voltage to turn on but due to voltage drop sufficient amount of voltage is not reached to the particular cell and cell may be goes into metastable state and affect the timing and performance.

Methods to improve dynamic IR drop

1. Use de-cap cells.
2. Increase the number of straps.

Tools used for IR drop analysis:

1. Redhawk from Apache
2. Voltage storm from cadence

Electromigration:

When a high density of current is flowing through metal layers, the atoms (electron) in the metal layers are displaced from their original position causing open and shorts in the metal layers. Heating also accelerates EM because higher temperature cause a high number of metal ions to diffuse.

In higher technology nodes we saw the EM on power and clock metal layers but now in lower nodes the signal metal layers are also needed to be analyzes due to an increased amount of current density.
Clock nets are more prone to EM because they have high switching activity, because of this only we are avoiding to use high drive strength clock buffers to build the clock tree.

Methods to solve EM:

1. Increase the width of wire
2. Buffer insertion
3. Downsize the driver
4. Switch the net to higher metal layers.
5. Adding more vias
6. Keep the wire length short
7.

Checklist after power planning:

1. All power/Ground supplies defined properly?
2. Is IR drop acceptable?
3. Is the power supply is uniform?
4. Are specials cell added? (Tap cells and end cap cells)
5. Any power related shorts/opens in the design?

UPF & special cells used for power planning

UPF is an IEEE standard and developed by members of Accellera.UPF is designed to reflect the power intent of a design at a relatively high level.
UPF scripts describe which power rails should be routed to individual blocks, when blocks are expected to be powered up or shut down, how voltage levels should be shifted as signals cross from one power domain to another and whether measures should be taken to retain register and memory-cell contents if the primary power supply to a domain is removed.

The backbone of UPF (Synopsys), as well as the similar Common Power Format (CPF) (cadence), is the Tool Control Language (TCL). The TCL command "**create_power_domain**", used to define a power domain and its characteristics. for example, if this command is used by UPF-aware tools to define a set of blocks in the design that are treated as one power domain that is supplied differently to other blocks on the same chip. The idea behind this type of command is that power-aware tools read in the description of which blocks in a design can be powered up and down independently.

Content in UPF:

- **Power domains:** some time design have more than one power domain like multi Vdd design and sometimes only a single power domain design. In this group of elements share a common set of power supply.
- **Supply rails:** distribution of power in supply nets, ports, supply sets, power state
- **Additional Protection By special cells**: level shifters, isolation cells, power switches, retention registers

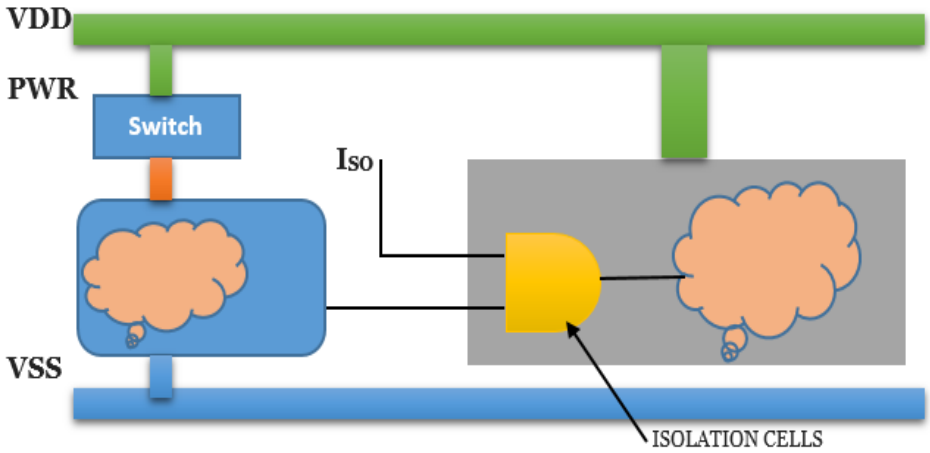
Isolation cells:

Isolations cells always used between two domains active mode domains and shut down mode domain.

Consider there are two power domains one D1 is in shutdown mode and D2 is in active mode, if data is passed through D1 to D2 this will not a valid data received at D2. To prevent this condition we insert an isolation cell b/w these two power domains to clamp a known value at the D2 otherwise it will give unknown value.

Shut down domain outputs may be floating outputs and this could be a problem when other active domains get these floating outputs as an input. This could affect the proper functioning of the active domain.

These cells are also called "clamp" because these cells convert the invalid or floating outputs to a known value or clamp it to some specified known value.

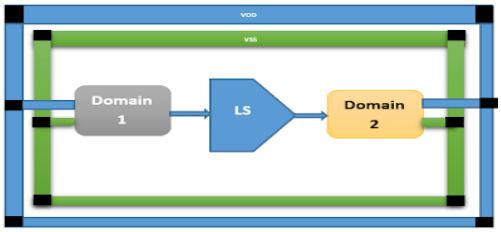


Level shifter:

The chip may be divided into multiple voltage domains, where each domain is optimized for the needs of certain circuits. For example a system on a chip might use a high supply voltage for memories to ensure cell stability, a medium voltage for a processor and low voltage for IO peripherals. Some of the challenges in using voltage domain include voltage levels for signals that cross domains, selecting which circuits belongs in which domain

Whenever a signal is going from low domain to high domain there will not be full output swing available at the output of high domain or vice versa. In this condition we required level shifter to shift up or down the voltage level according to the requirement.

let us consider one example, there are two designs with different power domains, one design is working on 1.5 v and the other is working on 1 v power supply. Now if the signal is passed from 1.5v domain to 1 v domain then wrong logic is interpreted by 1v domain. So to prevent this level sifter are inserted b/w two domains. The main function of the level shifter is to shift the voltage level according to the requirement of voltage levels



Retention Registers:

To reduce power consumption when the devices are not in use, the power domain is switched off for those devices. When the design blocks are in switched off or sleep mode, data in all flip-flops contained within a block will be lost. If the designer desires to retain this state then retention registers are used. Retention register requires D flip-flop and latch and required always-on supply to retain the data. Using the retention register area required more as compared to normal flop because here we are using flip-flop with always-on supply. So the area required more when we are using retention registers.

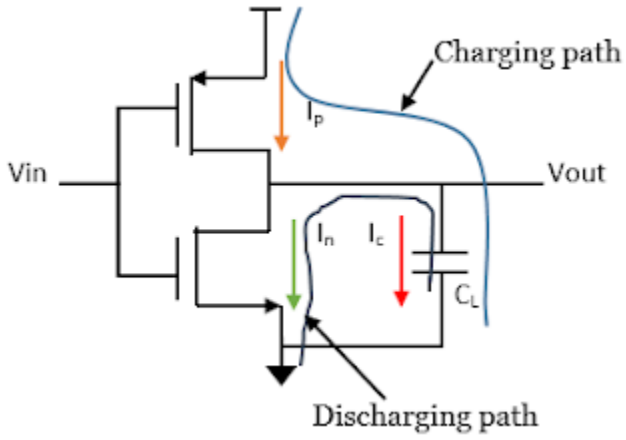
Always On cells:

These cells are always on irrespective of where they are placed. Generally buffer and inverters are used for always ON cells. Always cells are also present in UPF. They can be a special cell or regular buffer. If they are special cells they have their own secondary power supply and placed anywhere in the design.

If they are regular buffer/inverters they require always-on cells and restrict the placement of these types of cells in the specific region
For example if data needs to be routed through or from the sleep block domain to active block domain and distance between both domains is very long or driving load is also very large then buffer might be needed to drive the long nets from one domain to another.

SOURCES OF POWER DISSIPATION IN CMOS

Power dissipation in CMOS circuits comes from two components:



Dynamic dissipation due to:

- charging and discharging load capacitance as gate switch.
- "short circuit" current while both PMOS and NMOS are partially ON.

Pdynamic = Pswitching + Pshort circuit.....(1)

Static dissipation due to:

- subthreshold leakage through OFF transistors

- gate leakage through the gate dielectric
- junction leakage from source /drain diffusions

Pstatic = (Isub + Igate + Ijunction)Vdd.....(2)

So,

Ptotal =Pdynamic +Pstatic.....(3)

Dynamic power:

Whenever input signals in the circuit are changed its state with time, and it causes some power dissipation. Dynamic power is required to charging and discharging the load capacitance when transistor input switches

When the input switches from 1 to 0 the PMOS transistor (PULL UP network) turns ON and charges the load to VDD. And that time energy stored in the capacitance is

$$\mathbf{E_C = ½\ C_L\ V^2_{DD}}$$

The energy delivered from the power supply is

$$\mathbf{E_C = C_L\ V^2_{DD}}$$

Observed that only half of the energy from the power supply is stored in the capacitor. The other half is dissipated (converted to heat) in the PMOS transistor because transistor has a voltage across it at the same time current flows through it. The dissipated power depends on only the load capacitance not on transistor or speed at which the gate switches.

When the input switches from 0 to 1, the PMOS transistor turns off and the NMOS transistor is turned ON, and discharging the capacitor. The energy stored in the capacitor is dissipated in the NMOS transistor. No energy is drawn from the power supply in this case.
Depending upon the inputs at the gate of the transistor one gate is on the other is off because of charging and discharging of the capacitor. The inverter is sized for equal rise and fall times so we know that in one cycle we have rising and falling transition.

On rising edge output change $Q = CV_{DD}$ is required to charge the output node to V_{DD} (i.e. cap is charged to V_{DD}) and on falling edge the load capacitance is discharged to GND.
Now suppose gate switches at some average frequency f_{sw} (switching frequency). Over the time period T, the load is charging and discharging $T*f_{sw}$ times. So average power dissipation is

$$\mathbf{P_{switching} = CV^2_{DD}\ f_{sw}}$$

This is called dynamic power because it arises from the switching of the load. Because most gates don't switch every clock cycle, so it is convenient to express switching frequency as an activity factor(α) times the clock frequency f, now power dissipation written as

$$\mathbf{P_{switching} = \alpha\ CV^2_{DD}\ f}$$

Activity factor:

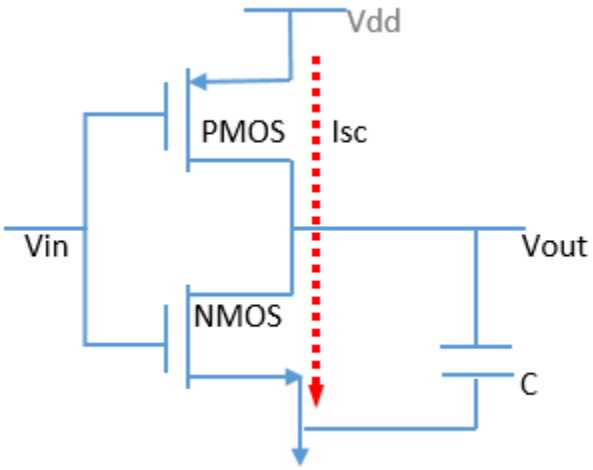
The activity factor is the probability that the circuit node transitions from 0 to 1 because that is only the time the circuits consume power. A cock has an activity factor $\alpha = 1$ because it rises and falls every cycle. The activity factor is powerful and easy to use lever for reducing power.
If a circuit can be turned off entirely, the activity factor and dynamic power go to zero. When a block is on the activity factor is 1. Glitches in the circuit can increase the activity factor.

Techniques to reduce dynamic power:

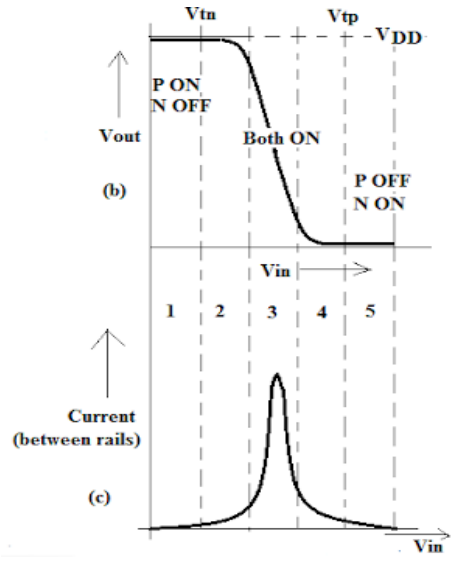
we know **$P_{switching} = \alpha C V_{DD}^2 f$**

- **Reduce α** (clock gating, sleep mode)
- **Reduce C** (small transistors, short wires, smaller fan-out)
- **Reduce Vdd** (reduce supply voltage up to which circuit works correctly)
- **Reduce f** (reduce the frequency to the extent if possible power-down mode, without sacrificing the performance beyond acceptable level)

Short circuit power:
Dynamic power includes a short circuit power component. It occurs in CMOS when input of gate switches. When both pullup and pulldown networks are conducting for a small duration and there is a direct path b/w VDD to VSS. during this scenario **spikes** will be generated momentarily in the current as shown in fig below. The current is flowing from VDD to VSS is also called cross-bar current.
This is normally less than 10% of the whole power so it can be estimated by adding 10% to the switching power. This power is directly related to the frequency of switching, as clock frequency increases the frequency of transition is also increased thus short circuit power dissipation also increases.



SHORT CIRCUIT CURRENT PATH IN CMOS

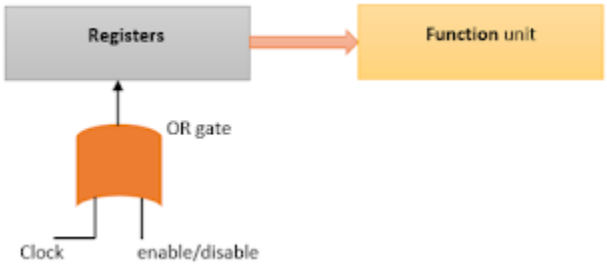


when both transistor are ON momentarily the short circuit comes in the form of spike

Clock gating:

Power dissipation is highly dependent on the signal transition activity in a circuit and the clock is mainly responsible for signal activities. Clock is the brain of the entire system so wherever clock transition takes place all the circuit works synchronously. Sometimes we have not required that clock to some blocks if we disable this clock to that particular block then switching activity reduced and activity factor α will also reduce and power dissipation also reduced.

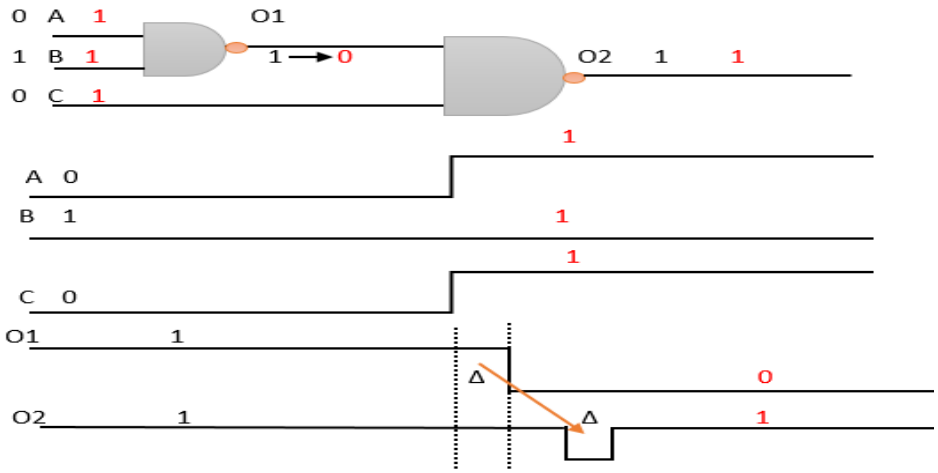
The inactive Registers are loaded by clock not directly but loaded through the OR gate using enable signal. When we know we don't require a functional unit we will set the enable to 1 so that the output of the OR gate will be constant 1, and the value of register will not change and therefore there will be no signal transition into a functional unit. When we are switching off the clock signal from the functional unit additional logic is required depending on the scenario that the functional unit required or not. But clock signal might add some delay in the critical path due to additional circuitry (OR gate and for enable different circuitry) then skew analysis is required.



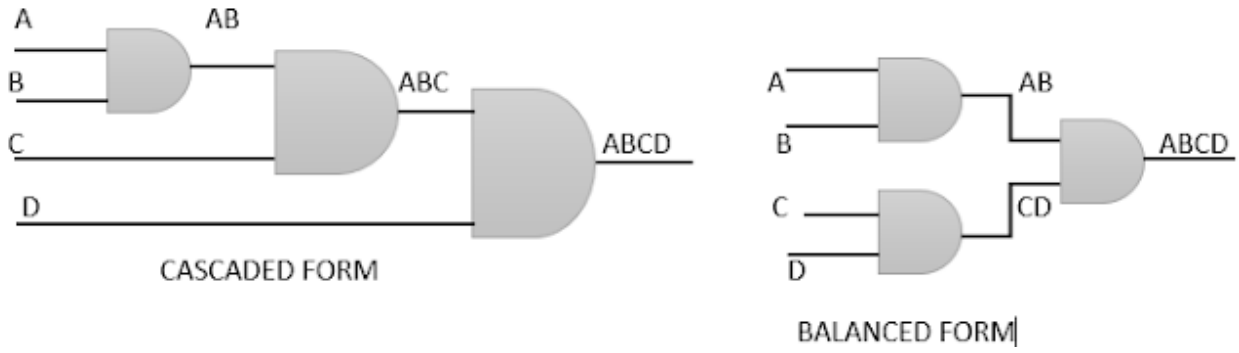
CLOCK GATING

Glitches:

We know in reality every gate has a finite amount of delay because of that delay only glitch will occur.
Let input A B C changes from 010 to 111.
When input 0 1 0 then output O1=1 and O2= 1
Now the input is changed to 111 then output O1 = 0 but O2 =1 remains the same. These both the ideal case when gate delays are zero.
Now what happened when we will consider the gate delays also?
The gate delay will be added on to both output because of the small amount of delay the glitch will appear.



so how it will be reduced?



The output is the same for both and the number of gates is also the same and we achieved reduced glitch power because the signal will reach at the same time in figure 2 (output of first and second gate) so there is no delay difference. Another advantage that the critical path delay is also reduced (in first figure it takes 3 gates to get the outputs and in second figure output takes

only 2 gates. So we can reduce the glitch power dissipation by the realization of the circuit in balanced form instead of cascaded form.

HOW TO REDUCE SUPPLY VOLTAGE (VDD) TO REDUCE DYNAMIC POWER:

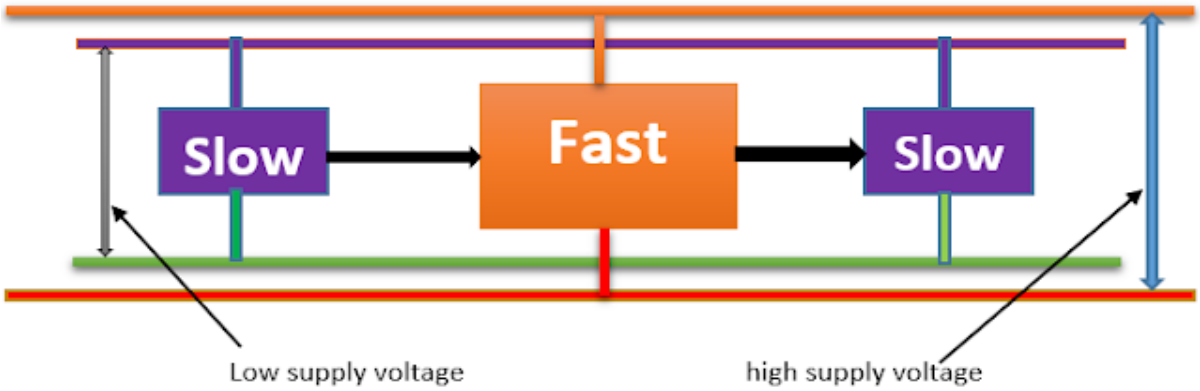
Weknow two design approach

Static approach: where the distribution of power supply voltage is fixed a priori among various functional blocks.

Dynamic approach: where the power supply voltage is changed as required.

Static voltage reduction approach:

First, you analyze the circuit and we can reduce the supply voltage that will make your circuit consumes less power but the circuit becomes may be slower so you have to analyze the circuit and find out which part of the circuit is not critical in terms of delay, you can possibly make that part little slower without touching the overall performances. So identify those parts and reduce the power supply for that.



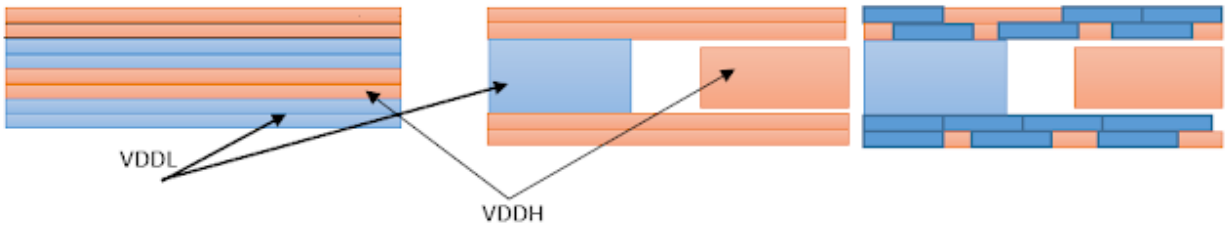
Suppose I have a circuit having three functional modules, let's assume we required the central block to be running fast but the other two blocks are running slower. So we use pair of voltage rails one is for low supply voltage and the other is for high supply voltage so the block which is supposed to run faster they are fed by the high supply voltage and the other two feed by low supply voltages rails, so our voltage is saved.
Here the distribution of the voltages is always fixed and here additional circuitry is required between different power domains because signals are sending from slow block to fast block and vice versa, some voltage translation is required.

Dynamic approach:
We adjust the operating voltage and frequency dynamically to match the performance requirements.

The modern-day processors can have different power modes like our laptops have different mode standby, sleep, hibernate, etc. Previously power is not the issue now power becomes more important because everyone wants high-performance mode with high battery mode i.e. system should run for a longer time.
This approach provides flexibility & doesn't limit the performance. The penalty of transition between two states can be high, it will take some time from one mode to another mode so it should not be done frequently.
High-performance mode: higher VDD and f
Power saving mode: lower VDD and f

VOLTAGE ISLANDS (MULTI-VDD)

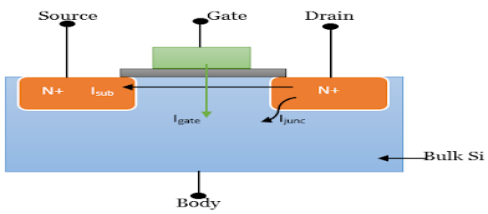
Cells are arranged in a row, there are two different voltage domains high and low. So all the cells who are supposed to give the high performance will be placed in high voltage domain and lower performance cells are sits in the low voltage domain. This is allowed for both macros and standard cell voltage alignment. sometimes times voltage island is in the same circuit row like some circuit has performance and some have high but if we are doing this the problem of power routing will become more difficult.



DIFFERENT POWER DOMAINS

Static power:

Static power is consumed even when a chip is not switching they leak a small amount of current. CMOS has replaced NMOS processes because contention current inherent to NMOS logic limited the number of transistors that could be integrated on one chip. Static CMOS gates have no contention current.
In processes with feature size above 180nm was typically insignificant except in very low power applications. In 90 and 65 nm processes, threshold voltage has reduced to the point that subthreshold leakage reaches levels of 1 sec to 10 sec of nA per transistor, which is significant when multiplied by millions of transistors on a chip.
In the 45 nm process, oxide thickness reduces to the point that gate leakage becomes comparable to subthreshold leakage unless high-k dielectric is employed. Overall, leakage has become an important design consideration in nanometer processes.



Static power sources:

1. Subthreshold leakage current: b/w source and drain. This is the dominant source of static power
2. Gate leakage current: from gate to body.
3. Junction leakage current: from source to body and drain to the body.

Static power reduction techniques:

1. Power gating
2. Multiple threshold voltages
3. Variable threshold voltages

Subthreshold leakage current:
It is caused by the thermal emission of carriers over the potential barrier set by a threshold. In a real transistor, current does not abruptly cut off below the threshold but rather drops off exponentially as shown in fig:
When the gate voltage is high (Vgs>Vt), the transistor is strongly ON. When the gate falls below Vt (Vgs < Vt) the exponential decline in current appears as a straight line. This current increase with temperature. It also increased as Vt is scaled-down along with the power supply for better performance.
Various mechanism affects the subthreshold leakage current

1. DIBL effect
2. Body effect
3. Narrow width affect
4. Effect of channel length
5. Effect of temperature

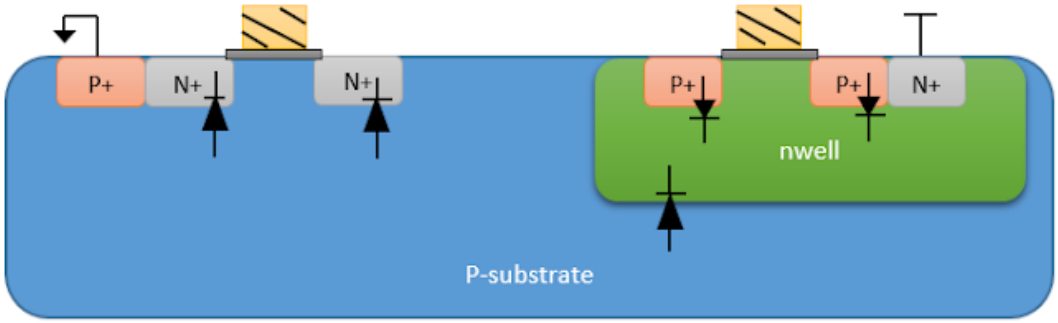
Gate leakage current:
It is a quantum mechanical effect caused by tunneling through an extremely thin gate dielectric. Gate leakage occurs when carriers tunnel through a thin gate dielectric when a voltage is applied across the gate (when the gate is ON). Gate leakage is an extremely strong function of the dielectric thickness. Gate leakage also depends on the voltage across the gate. For gate oxides thinner than 15-20 Å, there is a nonzero probability that an electron in the gate will find itself on the wrong side of the oxide, and it will get away through the channel. This effect of carriers crossing a thin barrier is called tunneling and because of this leakage current is flowing through the gate to the body.

Two physical mechanism for gate tunneling are:

1. **Fowler- Nordheim tunneling**: most important at high voltage and moderate oxide thickness
2. **Direct tunneling:** most important at lower voltage with thin oxides and it is the dominant leakage component.

Junction leakage current:

The p-n junction between diffusion and the substrate or well form diode as shown in fig: The well-to-substrate junction is another diode. The substrate and well are tied to GND or VDD to ensure these diodes do not become forward biased in normal operation. However, reversed – biased diode still conducts a small amount of current I_D. Junction leakage occurs when a source or drain diffusion region is at a different potential from the substrate.

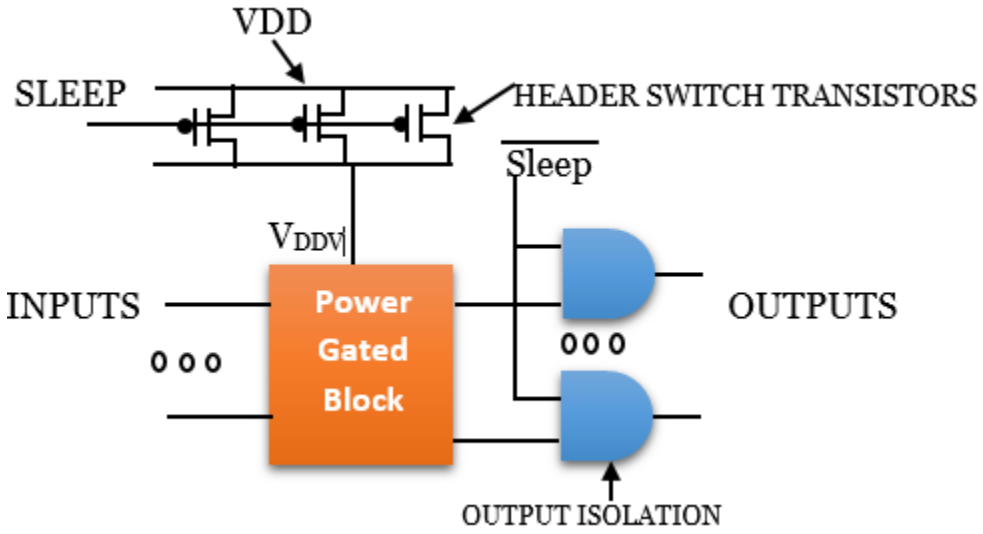


SUBSTRATE TO DIFFUSION DIODES IN CMOS

Static power reduction techniques:

POWER GATING:

The easier way to reduce static current during sleep mode is to turn off the power supply to the sleeping blocks. This technique is called power gating as shown in fig:



POWER GATING

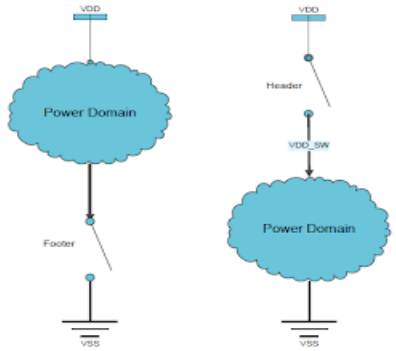
The logic block receives its power from a virtual V_{DD} rail, $V_{DDV.}$ When the block is active, the header switch transistors are ON and connecting V_{DDV} to V_{DD}.

When the block goes to sleep, the header switch turns OFF, allowing VDDV to float and gradually sink towards zero (0). As this occurs, the output of the block may take on voltage level in the forbidden or unknown state. The output of **isolations gates forces the output to a valid level during sleep** so that they do not cause problems in downstream logic.
Power gating introduces a number of design issues like header switch requires careful sizing, it should add minimal delay to the circuit during active operation and should have low leakage during sleep mode.

The transition between active and sleep modes takes some time and energy, so power gating is only effective when a block is turned off long enough. When a block is gated, the state must either saved or reset upon power-up. State retention registers use a second power supply to maintain the state. The important registers can be saved to memory so the entire block can be power-gated. The register must be reloaded from memory when power is restored.

Power gating was originally proposed as **multiple threshold CMOS** (MTCMOS) because it used low-vt transistors for logic and high-vt for header and footer switch.
Power gating can be done **externally** with disable input to a voltage regulator **or internally** with high–vt header and footer switches. External power gating completely eliminates the leakage during sleep, nut it takes a long time and energy because the power network may have 100s of nF decoupling capacitance to discharge. On-chip power gating can use a PMOS header switch transistor or NMOS footer switch transistors. NMOS transistors deliver more current per unit width so they can be smaller.

On the other hand, if both internal and external power gating is used, it is more consistent for both methods to cut off $V_{DD.}$



Power gating types:

1. Fine –grained power gating
2. Coarse-grained power gating

Fine-grained power gating: it is applied to individual logic gates, but placing this in every cell has enormous area overhead.

Coarse-grained power gating: in this the switch is shared across an entire block.

PLACEMENT

checks before doing placement:

After completion of floorplanning, power planning and placement of physical only cells Endcap cells and Tap cells, we check the base DRC and errors related to floorplanning like vertical spacing error, horizontal spacing error, min site row, vertical site row, and alignment. After inserting the tap cells, check to ensure that a standard cell peaceable area is protected by tap cells. Tap cells are placed correctly or not. Tap cells are typically used when most or all of the standard cells in the library contains no substrate or well taps. Generally, the design rule specifies the maximum distance allowed between every transistor in a standard cells and a well or substrate tap.

Advanced nodes often requires the insertion of additional tap cells to manage the substrate and well noise. Before placing the standard cells we add boundary cells (Endcap cells) also, which are added to the ends of the cell rows and around the boundaries of objects such as core and hard macros and we checked that endcap cells are placed or not.

After you have done the floorplanning i.e. created the core area, placed the macros and decided the power network structure of your design, it is time to let the tool do standard cell placement.

Placement:

Placement is the process of finding a suitable physical location for each cell in the block.
Tool only determine the location of each standard cell on the die.
Placement does not just place the standard cell available in the synthesized netlist, it also optimized the design.

The tool determines the location of each of the standard cell on the core. Various factors come into play like the timing requirement of the system, the interconnect length and hence the connections between cells, power dissipation, etc. the interconnect length depends on the placement solution used, and it is very important in determining the performance of the system as the geometries shrink.

Placement will be driven based on different criteria like timing driven, congestion driven, power optimization.

Placement is performed in two stages: coarse placement and legalization.

Goal of placement:
Timing, power, area optimization
Routable design
Minimum cell density and pin density(Reduce the congestion due to cells and pins)
Minimum timing DRC's

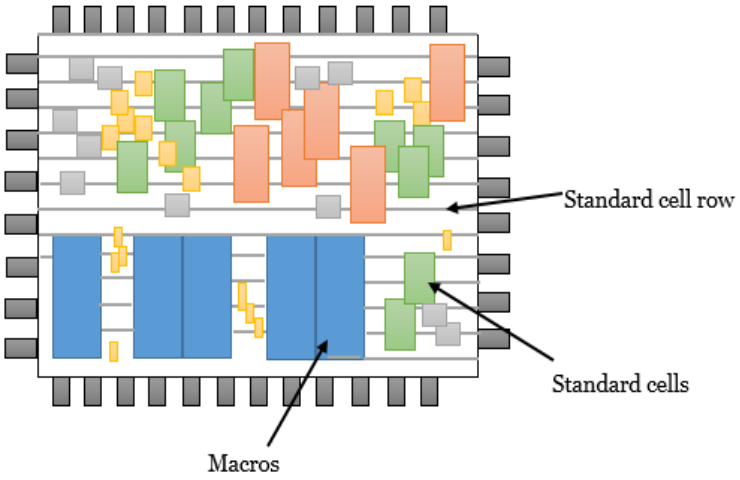
Inputs and output of placement:
Inputs
Netlist
Floorplan def
Logical and physical library
Design constraint
Technology file

Output:
Placement def

Before the start of placement optimization all wire load models are removed. Placement uses RC values from the virtual route to calculate timing. The virtual route is the shortest Manhattan distance between two pins. Virtual route RC values are more accurate than WLM RC's.

Coarse placement:

During the coarse placement, the tool determines an approximate location for each cell according to the timing, congestion and multi-voltage constraints. The placed cells don't fall on the placement grid and might overlap each other. Large cells like RAM and IP blocks act as placement blockages for standard cells. Coarse placement is fast and sufficiently accurate for initial timing and congestion analysis.



Legalization:

During legalization, the tool moves the cells to legal locations on the placement grid and eliminate any overlap between cells. These small changes to cell location cause the lengths of the wire connections to change, possibly causing new timing violations. Such violations can often be fixed by incremental optimization, for example: by resizing the driving cells.

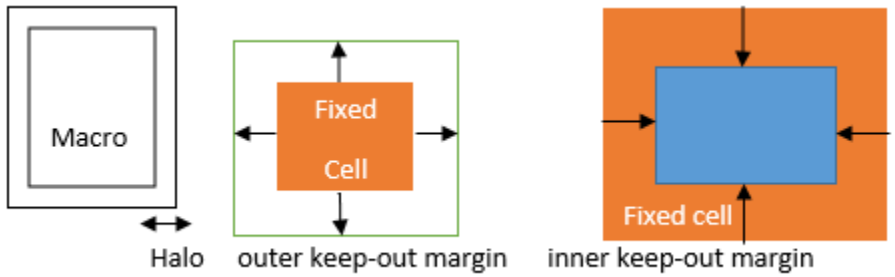
Placement constraints provide guidance during placement and placement optimization and legalization so that congestion and timing violations will be reduced.

1. Placement blockages
2. Placement bounds
3. Density constraint
4. Cell spacing constraint

Placement blockages:

It is the area where the cells must avoid during placement, optimization and legalization.
It can be hard and soft.
ICC tools supports two types of placement blockages
Keep-out margin
Area-based placement blockage: soft, hard, partial

Keep-out margin: it is a region around the boundary of fixed cells in a block in which no other cells are placed. The width of the keep-out margin on each side of the fixed cell can be the same or different. Keeping the placement of cells out of such regions avoids congestion and net detouring and produces better QOR (quality of results).



Create_keepout_margin –outer {10 10 10 10} my_lib_macro

By default hard keepout margin (lx by rx ty) [left bottom right top]

Create_keepout_margin –outer {10 10 10 10} my_lib_macro –type hard –name xyz

Area-based placement blockage:

It is a rectangular region in which cells can be placed or not or we can limit the number of cells.

Hard blockages: It prevents the placement of standard cells being placed in the blockage area.

Create_placement_blockage -boundary {10 20 100 200} –name pb0. By default it is hard blockages.

Soft blockage: during optimization buffer/inv can be placed in that blockage area.

Create_placement_blockage -boundary {10 20 100 200} –name pb1 –type soft

Partial placement blockages: it limits the cell density in a particular region.

Create_placement_blockage –boundary {10 20 100 200} –type partial –blocked_percentage 40 (it means 40 % area is blocked for placement of standard cells and rest of the 60% available for placement of standard cells)

Note: if you have both blockages are present at the same place then hard blockages take priority over the soft placement blockages.

Placement bounds:

It is a constraint that controls the placement of groups of leaf cells and hierarchical cells. It allows you to group cells to minimize wire length and place the cells at most appropriate locations. When our timing is critical during placement then we create bounds in that area where two communicating cells are sitting far from another. It is a fixed region in which we placed a set of cells. It comprises of one or more rectangular or rectilinear shapes which can be abutted or disjoint. In general we specify the cells and ports to be included in the bound. If a hierarchical cell is included, all cells in the sub-design belong to the bound.

Types of bounds:

1. Soft move bound
2. Hard move bound
3. Exclusive move bound

Soft move bound:
In this tool tries to place the cells in the move bound within a specified region, however, there is no guarantee that the cells are placed inside the bounds.

Create bound –name b0 –type soft –boundary {10 10 20 20} instance_1 #define softbound for instance_1 with its left corner at (10 10) and its upper-right corner at (20 20).

Hard move bound:

In this tool must place the cells in the move bound within a specified region.

Create bound –name b1 –type soft –boundary {10 10 20 20} instance_2

Exclusive move bound:

In this tool tries to place the cells in the group bound within a floating region, however, there is no guarantee that the cells are placed inside the bounds

Create bound –name b2 –exclusive –boundary {10 10 20 20} instance_1

Density controls:
It means how the density of cells can be packed. We can control the overall placement density for the block or the cell density for specific regions. To control the cell density for specific regions we can also use partial placement blockages.

Power optimization:
Dynamic power: This is the energy dissipated due to the voltage or logic transitions in the design objects, such as cells, pins and nets. The dynamic power consumption is directly proportional to the number and frequency of transition in the design.

Static (leakage) power: This is the energy dissipated even when there is no transition in the circuit? This is also known as leakage power and depends upon device characteristics. The main contributor to the leakage power is the sub-threshold-voltage leakage in the device. At lower technology nodes, leakage power consumption contributes significantly to the total power consumption of the circuit

In a library contains multiple-threshold- voltage cells,

The LVT cells have higher leakage current but better performance.
The HVT cells have lower leakage current but worst performance.
Percentage low threshold voltage optimization tries to find a balance between power and performance goals restricted the use of LVT cells.

During low power placement, the tool tries to minimize the length of high switching nets to improve the power QOR.
During Dynamic power-driven placement, the tool tries to improve both the timing and power of the critical nets and the power QOR without affecting the timing QOR.

Via ladder insertion:
A via ladder is a stacked via that starts from the pin layer and extends into an upper layer where the router connects to it. Via ladder reduces the via resistance which can improve the performance and Electromigration robustness. The tool can automatically insert the via ladder for cell pins on timing critical paths.

After initial placement and optimization by using place_opt command we perform incremental placement and optimization by using refine_opt command.

Place_opt: This command performs coarse placement, HFNS, optimization and legalization. In the **place_opt** command, the **–congestion** option causes the tool to apply **–high** effort to congestion removal for better routability, this will require more runtime and cause area utilization to be less uniform across the available placement area.

Refine_opt: if congestion is found to be a problem after placement and optimization. It can improve incrementally with the refine_opt command.

Refine_opt: perform 5 stages

1. **Initial path optimization**: it incrementally moves registers along timing paths to improve timing.
2. **Incremental placement**: to reduce congestion and improve routability.
3. **Incremental optimization**: perform incremental timing, area, congestion and leakage power optimization.
4. **Final placement:** final phase of path optimization to improve timing
5. **Legalization:** tool legalize the placement.

Magnet placement:

To improve congestion for a complex floorplan or to improve timing for the design we can use magnet placement to specify fixed object as a magnet and have the tool place all the standard cells connected to the magnet object close to it. We can fix macrocells, pins of fixed macro or IO ports as the magnet object.

For best results perform magnet placement before standard cell placement.

Command: magnet_placement

Timing driven placement:

Tool tries to place the standard cells along timing critical path close together to reduce net RC and meet setup timing.

Congestion driven placement:

Tool tries to spread the cells where the density of cells are more for the reduction of congestion.

Different task during placement:

- Placement of standard cells
- Optimization of area, power, congestion and timing.
- Legalization
- HFNS
- Scan chain reordering

High Fan-out Net Synthesis (HFNS):

In the placement stage we do this process. The process of buffering the high fan-out to balance the load because if design has too many loads then it affects delay and transition time. We know delay is load is directly proportional to the delay. By buffering the HFN the load can be balanced and this process is called the HFNS.

High fanout nets are mainly reset, preset, scan enable etc. these nets are not synthesized in the synthesis stage, also make sure you set an appropriate fan-out limit for your library using the command **set_max_fanout 20 [design_name]**

Used ideal clock in placement stage:

Clock net is also a high fan-out nets but in placement stage we set **set_ideal_clock or set_dont_touch** commands on the clock signal. If we don't take the ideal clock here the clock constraints like skew, insertion delay of clock buffers are not used and it affects the clock tree building. The clock network is ideal and does not have a clock buffer tree available for accurate clock network timing analysis. In ICC we use the following command to make sure that the clock is ideal not propagated in the placement stage.

Set_ideal_network [all_fanout –flat –clock_tree]

DFT optimization:

If block contains scan chains by default create_placement, place_opt and clock_opt commands perform DFT optimization. During initial placement, the tool focuses on the QOR for the function nets by ignoring the scan chains. After initial placement, the tool further improves the QOR by repartitioning and reordering the scan chains based on the initial placement.

Scan chains reordering reduces wire length so timing will improve.

Scan chains reordering minimize congestions and improves routability
The scan chain information (SCANDEF) from synthesis can be transferred to ICC compiler into two ways:
By loading the netlist in DDC format
By loading a SCANDEF file.

TIE cells insertion:

Sometimes in the netlist some unused inputs are tied to VDD/VSS (logic1/logic0). It is not recommended to connect a gate directly to the power network, so we use TIEHI or TIELO cells if available in the library. These are single pin cells that effectively ties the pin it connects high or low.

Optimization techniques:

Netlist constructing only changes existing gates, does not change functionality.

1. Cloning
2. Duplicates gates
3. Gate sizing
4. Swapping of pins that can change the final delay
5. Fan-out splitting

Congestion:

Congestion occurs when the number of available routing resources is less than the required routing resources. This condition we can see in global routing. A congestion map can help us to visualize the quality of placement. The congestion map shows the borders between global routing cells highlighted with different colors that represent the different levels of overflow. The overflow and underflow of the all selected layers. For example, if a highlighted light blue on the edge global routing cells shows 10/9 that means there are 9 available wire tracks and the required tracks are 10.

What are the reasons for congestion?

- High standard cell density in a small area
- Placement of standard cells near the macros
- High pin density at the edges of macros due to high fan in cells like AOI, OAI
- Bad floorplan (no proper blockages, halos are present)
- Macros/standard cell might have used all the metal layers inside and routing resources will be less

- Placing macros at the center instead of the boundary.
- During IO optimization tool does buffering so lot of cells placed in the core area

How to control the congestion:

High cell density can cause the congestion.by default the cell density can be up to 95%. We can reduce the cell density in congested areas by using coordinate option.

Set_congestion_options –max_util 0.45 –coordinate {x1 y1 x2 y2}

Here we set the maximum cell density upto 45% and given the coordinates for the particular area.

If the design is congested, we rerun the place_opt with the –congestion and –effort high options. During congestion driven placement, the cells which are sitting together and caused the congestion are spread apart.
Place_opt –congestion_driven –effort high

Reduce the local cell density using partial placement blockages
Create_placement_blockage –boundary {10 20 100 200} –type partial –blocked_percentage 40 (it means 40 % area is blocked for placement of standard cells and the rest of the 60% available for placement of standard cells)

If we have more pin density, which can be reduced by adding cell padding to the cells which is causing congestion. Cell padding can be applied by setting the keepout margin command.
Create_keepout_margin –type soft –outer {10 10 10 10} my_lib_macro

Macro padding or placement halos, soft blockages and hard blockages around the macros are like placement blockages around the edges of the macros. This makes sure that no standard cells are placed near the pins of macros and corners of macros, thereby giving extra breathing space for the macro pin connections to standard cells.
Change the floorplan (macros placement, macro spacing and pin orientation)
Reordering the scan chains to reduce the congestion

Checks after placement:
- Check legalization
- Check PG connections for all the cells.
- Check congestion, density screens & pin density maps all these should be under control
- Timing QOR, there should not be any high WNS violations.
- Minimum max Tran and max cap violations.
- Check whether all don’t touch cells & nets are preserved.
- Check the total utilization of design after placement.

CTS

CLOCK TREE SYNTHESIS (CTS)

Clock is not propagated before CTS so after clock tree build in CTS stage we consider hold timings and try to meet all hold violations**.**

After placement we have position of all standard cells and macros and in placement we have ideal clock (for simplicity we assume that we are dealing with a single clock for the whole design). At the placement optimization stage buffer insertion and gate sizing and any other optimization techniques are used only for data paths but in the clock path nothing we change.

CTS is the process of connecting the clocks to all clock pin of sequential circuits by using inverters/buffers in order to balance the skew and to minimize the insertion delay. All the clock pins are driven by a single clock source. Clock balancing is important for meeting all the design constraints.

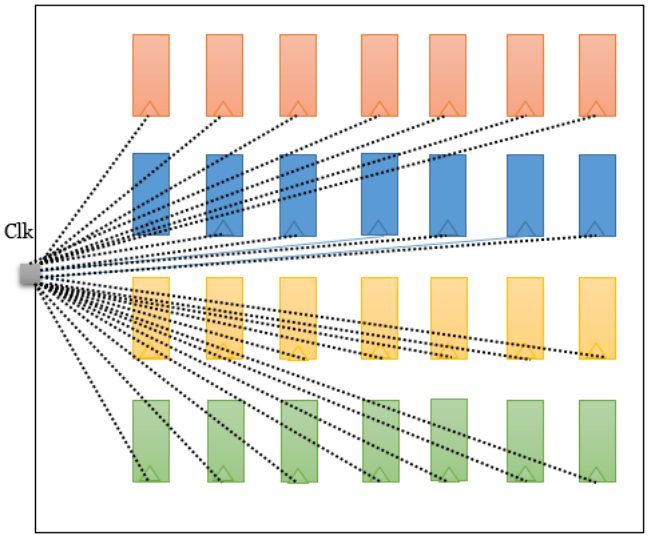


fig: before the clock tree is not build

Checklist before CTS:

- Before going to CTS it should meet the following requirements:
- The clock source are identified with the create_clock or create_generated_clock commands.
- The placement of standard cells and optimization is done. {NOTE: use check_legality –verbose command to verify that the placement is legalized. If cells are not legalize the qor is not good and it might have long run time during CTS stage}
- Power ground nets- pre-routed
- Congestion- acceptable

- Timing – acceptable
- Estimated max tran/cap – no violations
- High fan-out nets such as scan enable, reset are synthesized with buffers.

Inputs required for CTS:

- Placement def
- Target latency and skew if specify (SDC)
- Buffer or inverters for building the clock tree
- The source of clock and all the sinks where the clock is going to feed (all sink pins).
- Clock tree DRC (max Tran, max cap, max fan-out, max no. of buffer levels)
- NDR (Nondefault routing) rules (because clock nets are more prone to cross-talk effect)
- Routing metal layers used for clocks.

Output of CTS:

- CTS def
- Latency and skew report
- Clock structure report
- Timing Qor report

CTS target:

- Skew
- Insertion delay

CTS goal:

- Max Tran
- Max cap
- Max fan-out
- A buffer tree is built to balance the loads and minimize skew, there are levels of buffer in the clock tree between the clock source and clock sinks.

Effect of CTS:

Clock buffers are added congestion may increase non-clock cells may have been moved to less ideal locations can introduce timing and tran/cap violations.

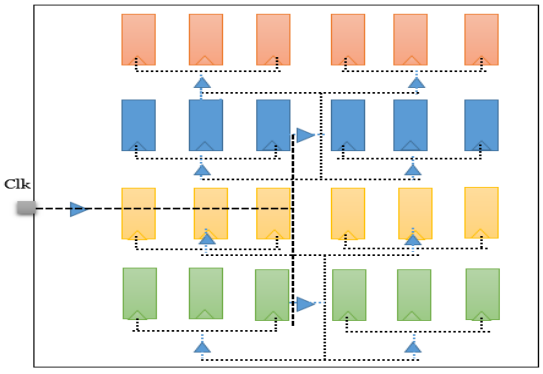


fig: CTS structure after clock tree build

Checks after CTS:

- In latency report check is skew is minimum? And insertion delay is balanced or not.
- In qor report check is timing (especially HOLD) met, if not why?
- In utilization report check Standard cell utilization is acceptable or not?
- Check global route congestion?
- Check placement legality of cells.
- Check whether the timing violations are related to the constrained paths or not like not defining false paths, asynchronous paths, half-cycle paths, multi-cycle paths in the design.

Clock Endpoints types:
When deriving the clock tree, the tool identifies two types of clock endpoints:

Sink pins (balancing pins):
Sink pins are the clock endpoints that are used for delay balancing. The tool assign an insertion delay of zero to all sink pins and uses this delay during the delay balancing.
During CTS, the tool uses sink pins in calculations and optimizations for both design rule constraints for both design rule constraints and clock tree timing (skew & insertion delay).
Sink pins are:
A clock pin on a sequential cell
A clock pin on a macro cell

Ignore pins:
These are also clock endpoints that are excluded from clock tree timing calculations and optimizations. The tool uses ignore pins only in calculation and optimizations for design rule constraints.

During CTS the tool isolate ignore pins from the clock tree by inserting a guide buffer before the pin. Beyond the ignore pins the tool never performs skew or insertion delay optimization but it does perform design rule fixing.

Ignore pins are:
Source pins of clock trees in the fanout of another clock
Non clock inputs pins of sequential cells
Output ports

Float pins: it is like stop pins but delay on the clock pin, macro internal delay. **Exclude pins:** CTS ignores the targets and only fix the clock tree DRC (CTS goals). **Nonstop pin**: by these pin clock tree tracing the continuous against the default behavior. Clock which are traversed through divider clock sequential elements clock pins are considered as non-stop pins.

Why clock routes are given more priority than signal nets:

Clock is propagated after placement because the exact location of cells and modules are needed for the clock propagation for the estimation of accurate delay, skew and insertion delay. Clock is propagated before routing of signals nets and clock is the only signal nets switches frequently which act as sources for dynamic power dissipation.

CTS Optimization process:

- By buffer sizing
- Gate sizing
- Buffer relocation
- Level adjustment
- HFN synthesis
- Delay insertion
- Fix max transition
- Fix max capacitance
- Reduce disturbances to other cells as much as possible.
- Perform logical and placement optimization to all fix possible timing.

NOTE: Mainly try to improve setup slack in preplacement, in placement and postplacement optimization before cts stages and in these stages neglecting the hold slack. in post placement optimization after cts stages the hold slack is improved. as a result of cts lot of buffers are added.

1. Difference between clock buffer and normal buffer?
2. what is pulse width violation?
3. what are the clock tree optimization process in detail?
4. what is useful skew and clock pushing and pulling in detail?
5. what is the difference between HFNS and CTS?
6. what are clock tree types?
7. what are the NDR rules applied to the clock net?
8. crosstalk noise and delay and it affects on set up and hold timing.
9. how positive skew is good for setup and negative skew is good for hold.
10. how to reduce the crosstalk effect.

Skew, latency, uncertainty, jitter

Skew:
This phenomenon in synchronous circuits. The Difference in arrival of clock at two consecutive pins of a sequential element.

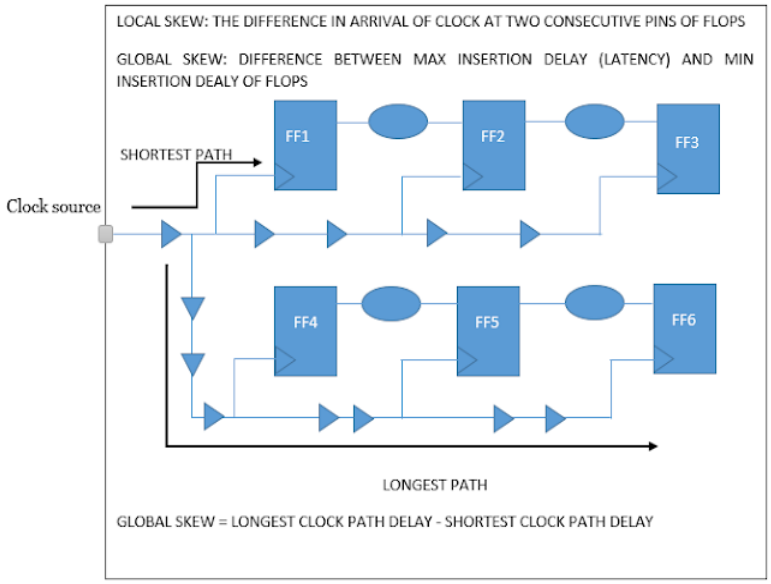


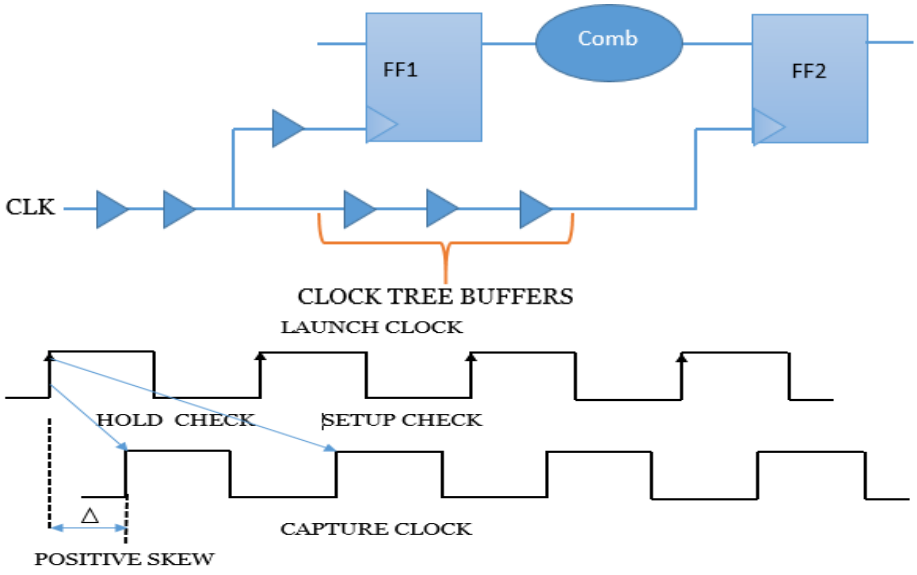
fig: skew representation

Sources of skew:

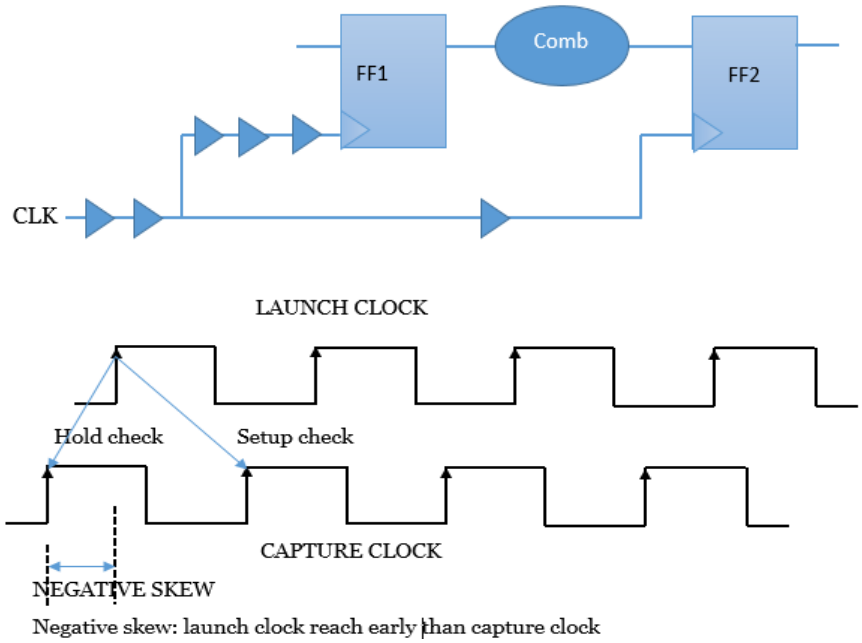
- Wire interconnect length
- Capacitive loading mismatch
- Material imperfections
- Temperature variations
- Differences in input capacitance on the clock inputs

Types of clock skew:

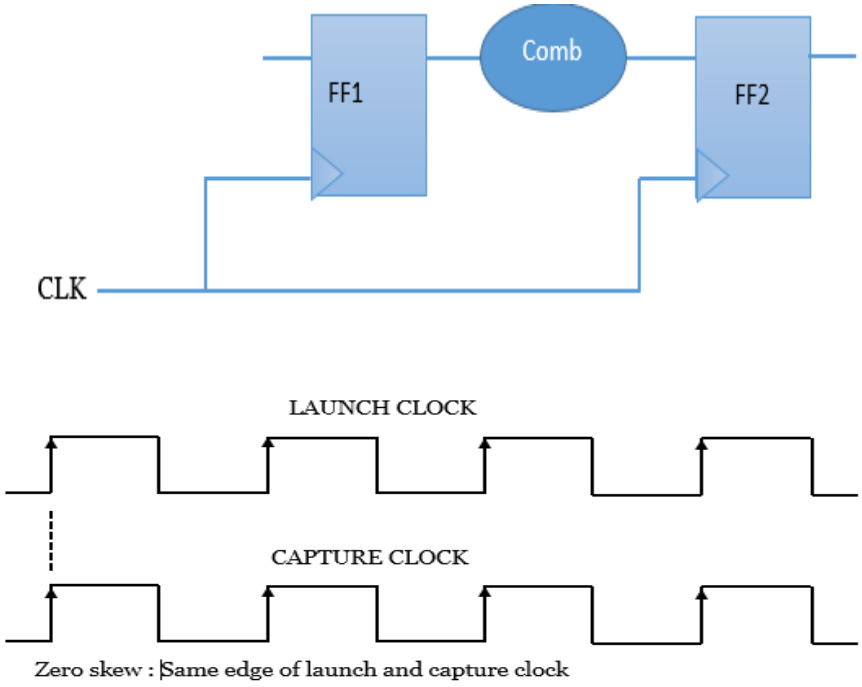
- **Positive skew:** if the capture clock comes late than the launch clock.



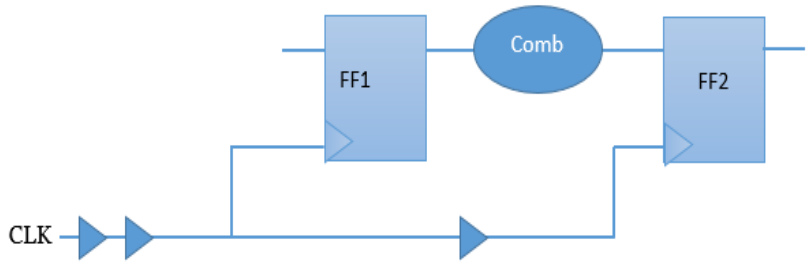
- **Negative skew:** if the capture clock comes early than the launch clock.



- **Zero skew:** when the capture clock and launch clock arrives at the same time. (ideally, it is not possible)



- **Local skew:** difference in arrival of clock at two consecutive pins of sequential element.it can be positive and negative local skew also.

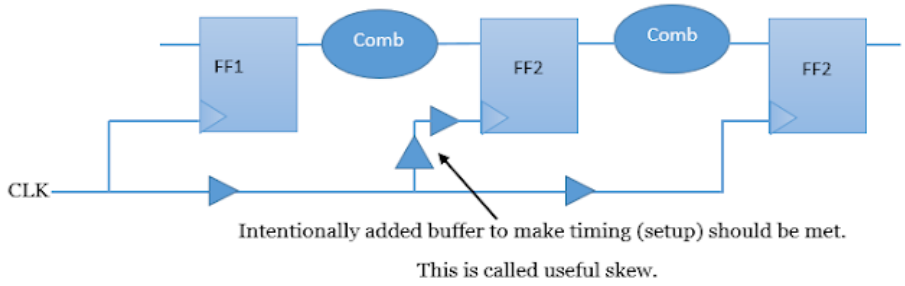


- **Global skew:** the difference between max insertion delay and the min insertion delay.it can be positive and negative local skew also.

max insertion delay: delay of the clock signal takes to propagate to the farthest leaf cell in the design.

min insertion delay: delay of the clock signal takes to propagate to the nearest leaf cell in the design.

- **Useful skew**: if the clock is skewed intentionally to resolve setup violations.



Latency:

The delay difference from the clock generation point to the clock endpoints.

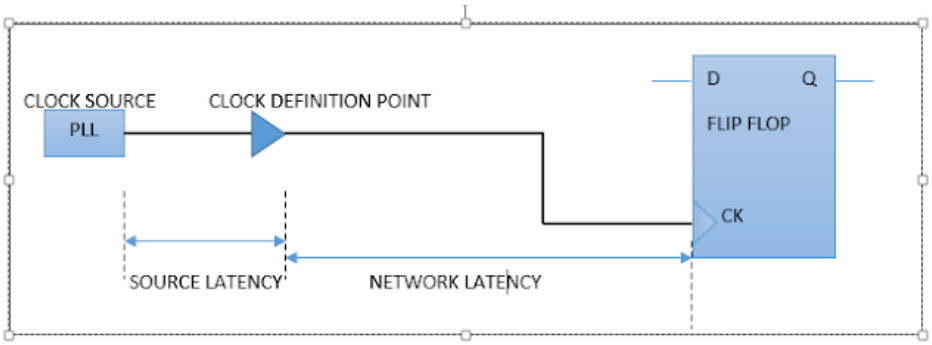


FIG: ON-CHIP CLOCK SOURCE

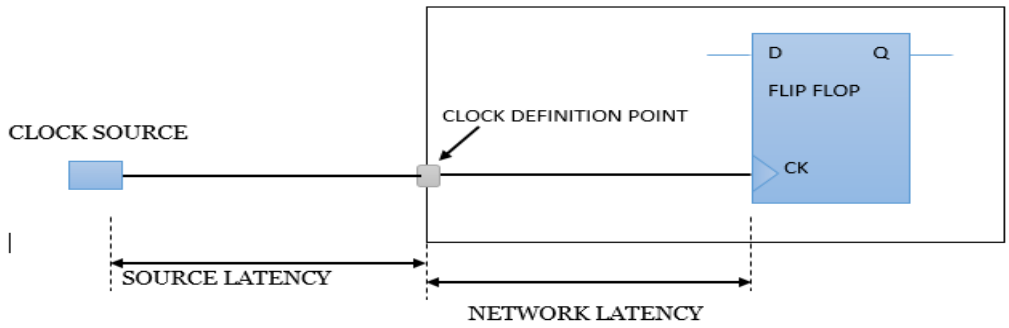


FIG: OFF CHIP CLOCK SOURCE

There are two types of latency:

Source latency: Source latency is also called insertion delay. The delay from the clock source to the clock definition points. Source latency could represent either on-chip or off-chip latency.

Network latency: The delay from the clock definition points (create_clock) to the flip-flop clock pins.

The total clock latency at the clock in a flip flop is the sum of the source and network latencies.

Set_clock_latency 0.8 [get_clocks clk_name1] ----> network latency
Set_clock_latency 1.9 –source [get_clocks clk_name1] -------> source latency
Set_clock_latency 0.851 –source –min [get_clocks clk_name2] -----> min source latency
Set_clock_latency 1.322 –source –max [get_clocks clk_name2] ------> max source latency

One important distinction to observe between source and network latency is that once a clock tree is built for the design, the network latency can be ignored. However the source latency remains even after the clock tree is built.

The network latency is an estimate of the delay of the clock tree before clock tree synthesis. After clock tree synthesis, the total clock latency from the clock source to a clock in of a flip flop is the source latency plus actual delay of the clock tree from the clock definition point to the flip flop.

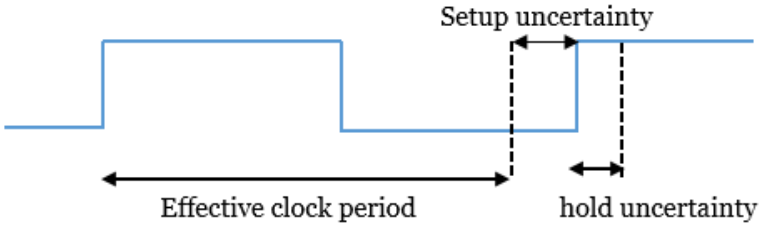
Clock Uncertainty: clock uncertainty is the difference between the arrivals of clocks at registers in one clock domain or between domains. it can be classified as static and dynamic clock uncertainties.

Timing Uncertainty of clock period is set by the command set_clock_uncertainty at the synthesis stage to reserve some part of the clock period for uncertain factors (like skew, jitter, OCV, CROSS TALK, MARGIN or any other pessimism) which will occur in PNR stage. The uncertainty can be used to model various factors that can reduce the clock period.

It can define for both setup and hold.

Set_clock_uncertainty –setup 0.2 [get_clocks clk_name1]
Set_clock_uncertainty –hold 0.05 [get_clocks clk_name1]

Clock uncertainty for setup effectively reduces the available clock period by the specified amount as shown in fig. and the clock uncertainty for hold is used as an additional margin that needs to be satisfied.



Pre CTS uncertainty = clock skew + jitter + margin
CTS uncertainty = jitter + margin

Static clock uncertainty: it does not vary or varies very slowly with time. Process variation induced clock uncertainty. An example of this is clock skew.

Sources of static clock uncertainty

- Intentional and unintentional mismatch in design
- On-chip variation (OCV)
- Load variation at every stage in clock distribution

Dynamic clock uncertainty: it varies with time. Dynamic power supply induced delay variation and clock jitter is the example of this

Sources of dynamic clock uncertainty:

- Voltage droop and dynamic voltage variations
- Temperature variations
- Clock generator jitter

Jitter: Jitter is the short term variations of a signal with respect to its ideal position in time. It is the variation of the clock period from edge to edge.it can vary +/- jitter value. From cycle to cycle the period and duty cycle can change slightly due to the clock generation circuitry. This can be modeled by adding uncertainty regions around the rising and falling edge of the clock waveform.

Sources of jitter:

- Internal circuitry of the PLL
- Thermal noise in crystal oscillators
- Transmitters and receivers of resonating devices

NOTE :

The first important point is that there are two phases in the design of when we are using a clock signal. In the first stage i.e. during RTL design, during synthesis and during placement the clock is ideal. The ideal clock has no distribution tree, it is directly connected at the same time to all flip flop clock pins.

The second phase comes when CTS inserts the clock buffer to build the clock tree into the design that carries the clock signal from the clock source pin to the all flip flops clock pins. After CTS is finished clock is called "propagated clock".

Clock latency term we are using when the clock is in ideal mode. It is the delay that exists from the clock source to the clock pin of the flip flop. This delay is specified by the user (not a real value or measured value).

When the clock is in propagated mode the actual delay comes into the picture then this delay is called as insertion delay. Insertion delay is a real and measured delay path through a tree of buffers. Sometimes the clock latency is interpreted as a desired target value for insertion delay.

Clock uncertainty> in the ideal mode we assume the clock is arriving at all the flip flop at the same time but ideally, we did not get the clock at the same time, maybe the clock will arrive at different times at different clock pins of a flip flop so in ideal mode clock assume some uncertainty . for example a 1ns clock with 100 ps clock uncertainty means that next clock pulse will occur after 1ns±50ps (either + or -).

The question of why the clock does bit always arrive exactly after one clock?
The reasons are:

1. The insertion delay to the launching flip flop's clock pin is different than the insertion delay of capturing clock (like maybe capture clock is coming before then the launch clock or capture clock is coming after the launch clock that difference is called skew)
2. The clock period is not constant. Some clock cycles may are longer or shorter than others in a random fashion. This is called clock jitter.
3. Even if the capture clock path and launch clock path are identical may be their path delays are different because different derate are applies on the path because the chip having different delay properties across the die due to process voltage and temperature variation i.e. called OCV (on-chip variation). This essentially increases the clock skew.

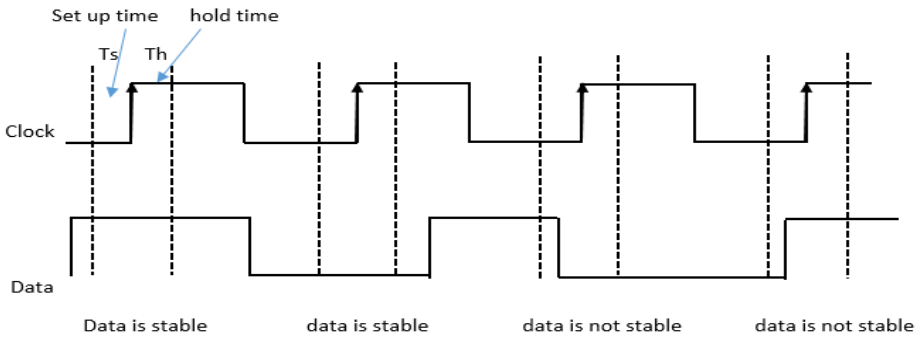
CTS (crosstalk and useful skew)

crosstalk and useful skew

For crosstalk and useful skew we have to know the basics of setup and hold timing. Here I am going to write here some small concepts related to timing that will be used for crosstalk and useful skew.

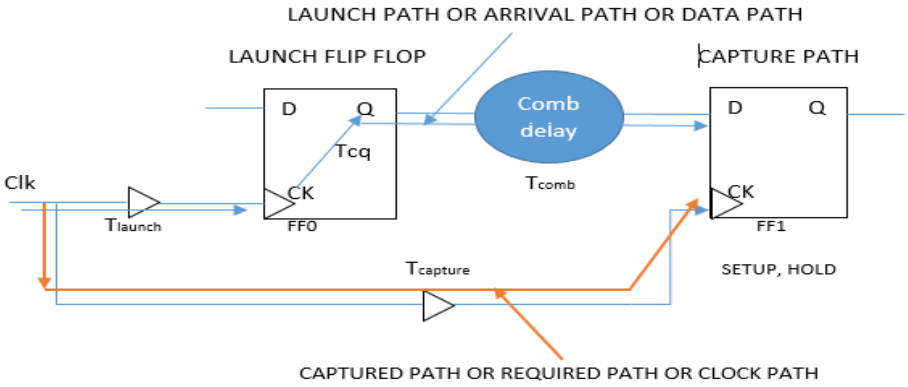
Setup time: The minimum time before the active edge of the clock, the input data should be stable i.e. data should not be changed at this time.

Hold time: The minimum time after the active edge of the clock, the input data should be stable i.e. data should not be changed at this time.



Capture edge: the edge of the clock at which data is captured by a captured flip flop.

Launch edge: the edge of the clock at which data is launched by a launch flip flop.



Launch flip flop: this is flop from where data is launched.

Captured flip flop: this is flop on which data is captured.

For setup check
Setup slack check = (required time) min – (arrival time) max
Arrival time = Tlaunch + Tcq+ Tcomb
Required time = Tclk+Tcapture-Tsu

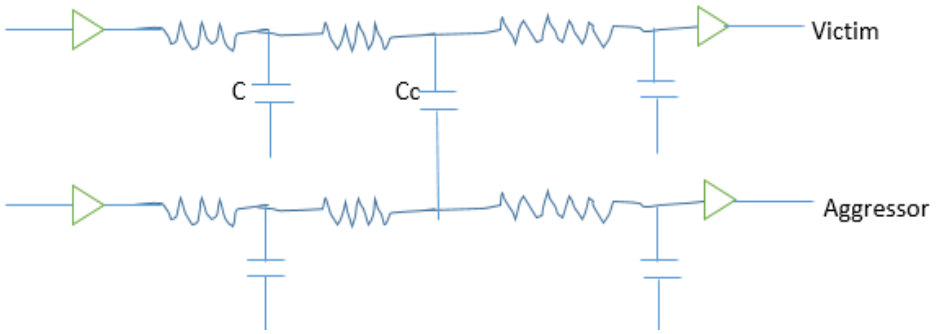
For setup time should not violate the required time should be greater than arrival time.

For hold check
Hold slack check = (arrival time) min – (required time) max
Arrival time = Tlaunch + Tcq + Tcomb
Required time = Tcapture + Thold

For hold time should not violate the arrival time should be greater than the required time.
#AT= arrival time
#RT = required time

Crosstalk noise: noise refers to undesired or unintentional effect between two or more signals that is going to affect the proper functionality of the chip. It is caused by capacitive coupling between neighboring signals on the die. In deep submicron technologies noise plays an important role in terms of functionality or timing of device due to several reasons.

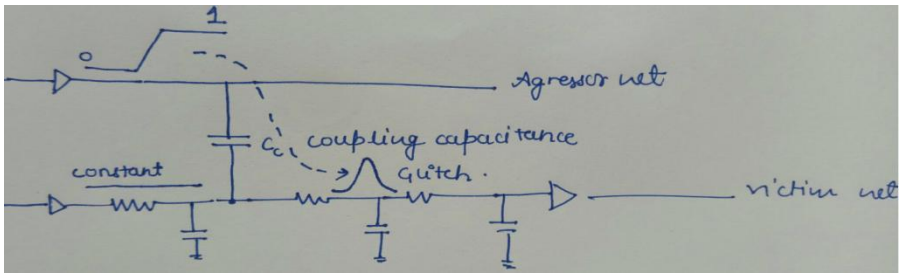
- Increasing the number of metal layers. For example, 28nm has 7 or 8 metal layers and in 7nm it's around 15 metal layers.
- Vertically dominant metal aspect ratio it means that in lower technology wire are thin and tall but in higher technology the wire is wide and thin, thus a greater proportion of the sidewall capacitance which maps into wire to wire capacitance between neighboring wires.
- Higher routing density due to finer geometry means more metal layers are packed in close physical proximity.
- Large number of interacting devices and interconnect.
- Faster waveforms due to higher frequencies. Fast edge rates cause more current spikes as well as greater coupling impact on the neighboring cells.
- Lower supply voltage, because the supply voltage is reduced it leaves a small margin for noise.
- The switching activity on one net can affect on the coupled signal. The effected signal is called the victim and affecting signals termed as aggressors.



There are two types of noise effect caused by crosstalk

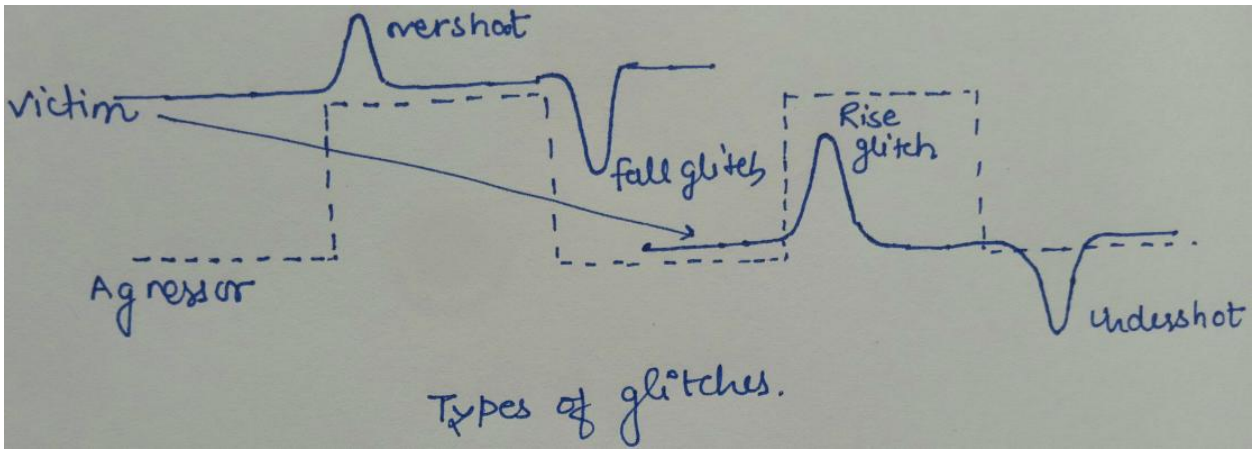
Glitch: when one net is switching and another net is constant then switching signal may cause spikes on the other net because of coupling capacitance (Cc) occur between two nets this is called crosstalk noise.

In fig the positive glitch is induced by crosstalk from rising edge waveform at the aggressor net. **The magnitude of glitch depends on various factors.**



- Coupling capacitance between aggressor and victim net: greater the coupling capacitance, larger the magnitude of glitch.
- Slew (transition) of the aggressor net: if the transition is more so magnitude of glitch also more. And we know the transition is more because of high output drive strength.
- If Victim net grounded capacitance is small then the magnitude of glitch will be large.
- If Victim net drive strength is small then the magnitude of glitch will be large.

Types of glitches:

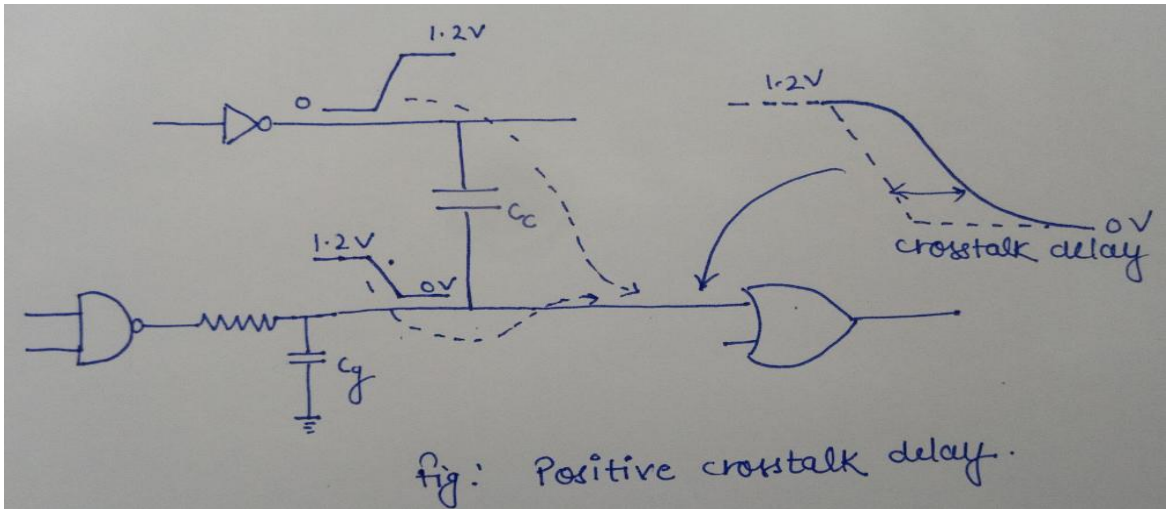


- **Rise:** when a victim net is low (constant 0) and the aggressor net is at a rising edge.
- **Fall:** when a victim net is low (constant 1) and the aggressor net is at the falling edge.
- **Overshoot:** when a victim net is high (constant 1)) and the aggressor net is at a rising edge.
- **Undershoot:** when a victim net is low (constant 0) and the aggressor net is at falling edge.

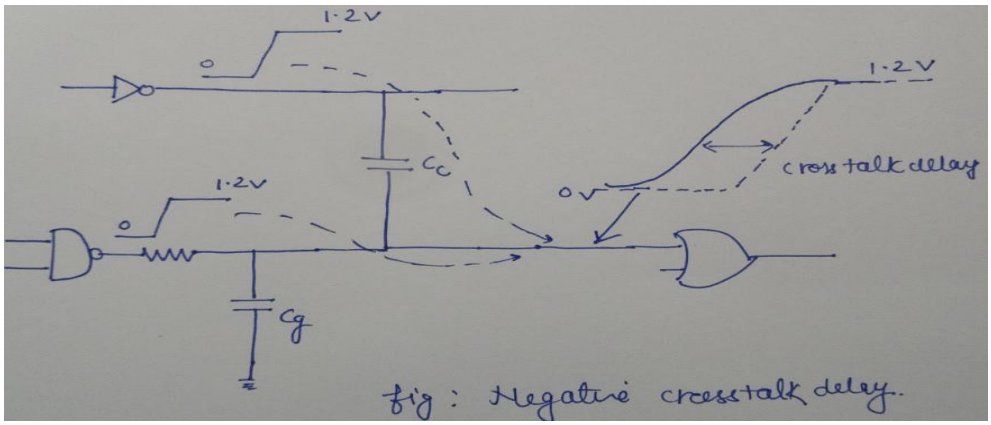
Crosstalk delay: when both nets are switching or in transition state then switching signal at the victim signal may have some delay or advancement in the transition due to coupling capacitance (Cc) occur between two nets this is called crosstalk delay.
Crosstalk delay depends on the switching direction of aggressor and victim net because of this either transition is slower or faster of the victim net.

Types of crosstalk:

1. **Positive crosstalk:** the aggressor net has a rising transition at the same time when the victim net has a falling transition. The aggressor net switching in the opposite direction increases the delay for the victim. The positive crosstalk impacts the driving cell, as well as the net, interconnect - the delay for both gets increased because charge required for the coupling capacitance Cc is more.



2. **Negative crosstalk:** the aggressor net is a rising transition at the same time as the victim net. The aggressor net switching in the same direction decrease delay of the victim. The positive crosstalk impacts the driving cell, as well as the net, interconnect - the delay for both gets decreased because the charge required for the coupling capacitance Cc is less.

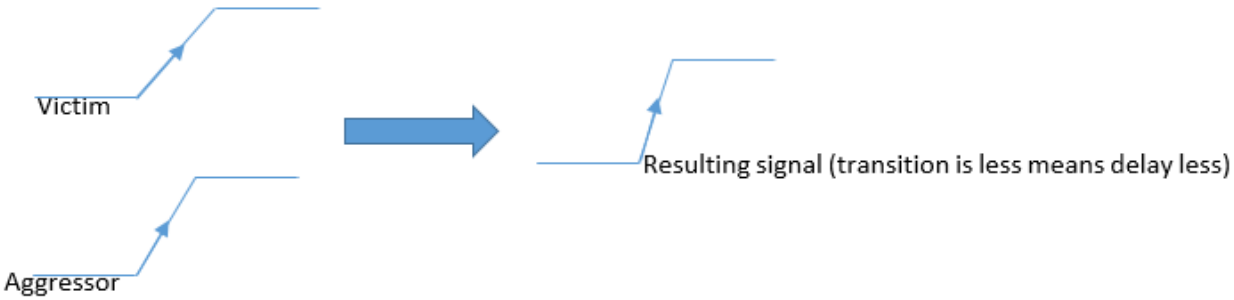


Crosstalk effect on timing analysis:

Consider crosstalk in data path:

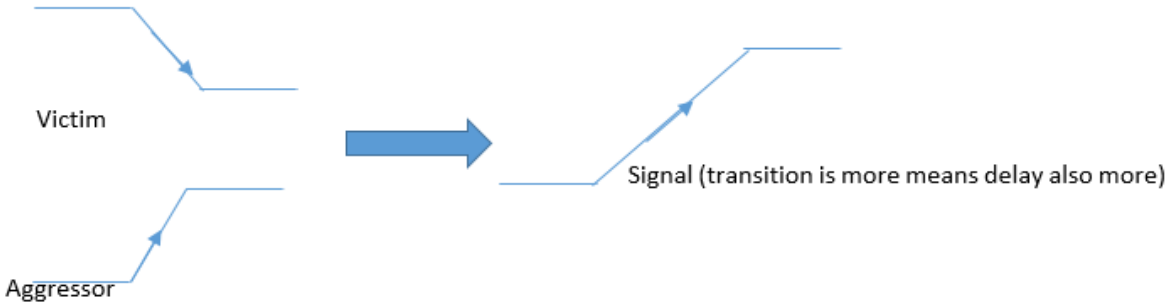
If the aggressor transition in the **same direction** as victim then victim transition becomes fast because of this data will be arrive early means arrival time will be less.

Setup = RT – AT(dec) this is good for setup #dec- decrease
Hold = AT(dec) – RT this is bad for hold



If the aggressor transition in a **different direction** as a victim then victim transition becomes slow because of this data will be 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



Consider crosstalk in clock path:

If the aggressor transition in the **same direction** as victim then victim transition becomes fast because of this data will be arrive early means arrival 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 transition in a **different direction** as a victim then victim transition becomes slow because of this data will be arrive late means arrival time will be more.
Setup = RT(inc) – AT this is good for setup
Hold = AT – RT(inc) this is not good for hold

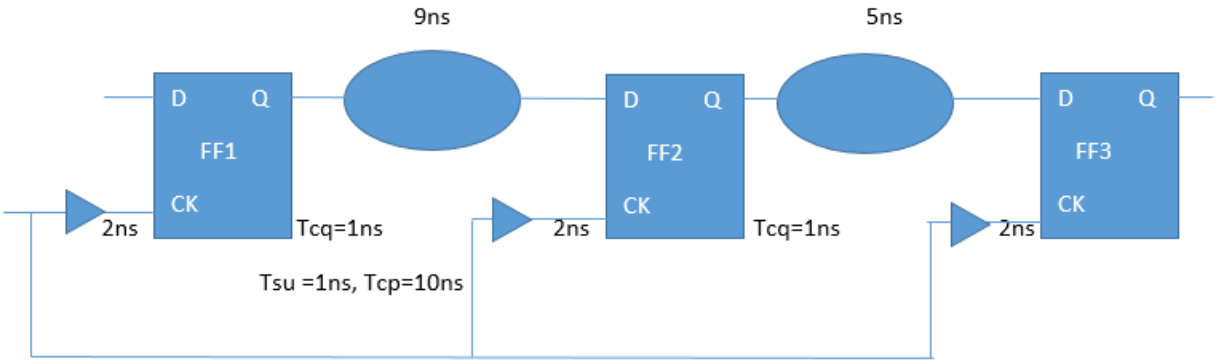
How to reduce the crosstalk:

- Wire spacing (NDR rules) by doing this we can reduce the coupling capacitance between two nets.
- Increased the drive strength of victim net and decrease the drive strength of aggressor net

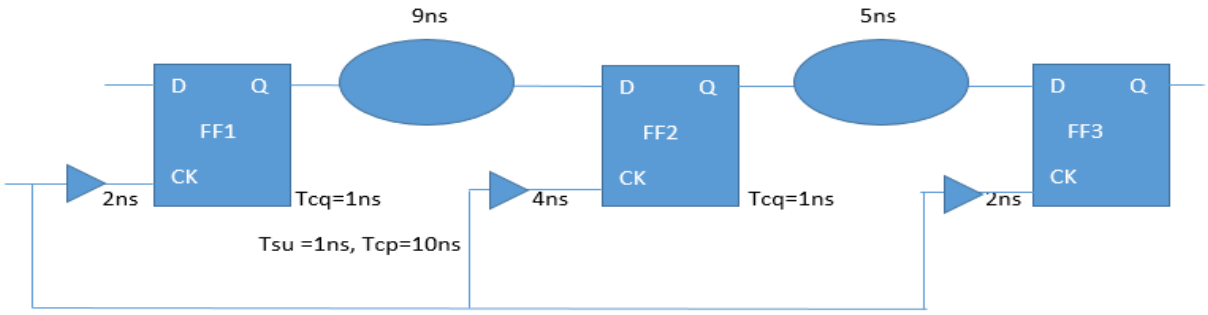
- Jumping to higher layers (because higher layers have width is more)
- Insert buffer to split long nets
- Use multiple vias means less resistance then less RC delay
- Shielding: high-frequency noise is coupled to VSS or VDD since shielded layers are connects to either VDD or VSS. The coupling capacitance remains constant with VDD or VSS.

Useful skew:

When clock skew is intentionally add to meet the timing then we called it useful skew.



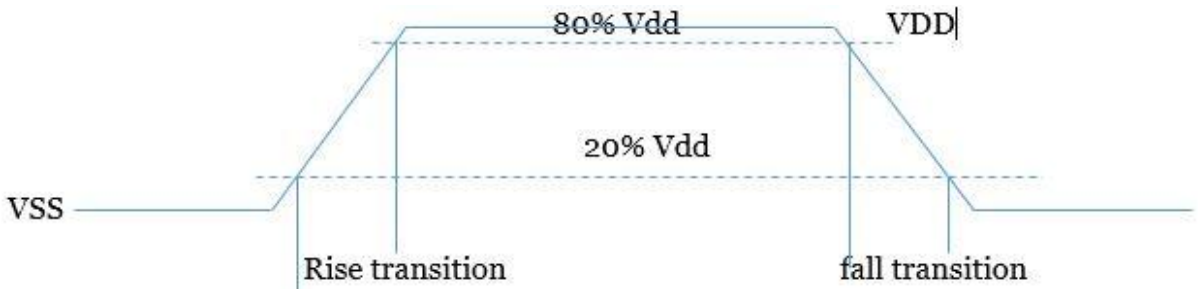
In this fig the path from FF1 to FF2
Arrival time = 2ns + 1ns + 9ns = 12ns
Required time = 10 ns (clock period) + 2ns - 1ns = 11ns
Setup slack = required time – arrival time
= 11ns -12ns
= -1ns (setup violated)
In this fig the path from FF2 to FF3
Arrival time = 2ns + 1ns + 5ns = 8ns
Required time = 10 ns (clock period) + 2ns - 1ns = 11ns
Setup slack = required time – arrival time
= 11ns -8ns
= 3ns (setup not violated)
Let's introduce some clock skew to path ff1 to ff2 to meet the timing. Here we add 2ns extra skew in clock path but we have to make sure about the next path timing violation.



In this fig the path from FF1 to FF2
Arrival time = 2ns + 1ns + 9ns = 12ns
Required time = 10 ns (clock period) + 4ns - 1ns = 13ns
Setup slack = required time – arrival time
= 13ns -12ns
=1ns (setup not violated)
In this fig the path from FF2 to FF3
Arrival time = 4ns + 1ns +5 ns = 10ns
Required time = 10 ns (clock period) + 2ns - 1ns = 11ns
Setup slack = required time – arrival time
= 11ns -10ns
= 1ns (setup not violated)

CTS CLOCK BUFFER AND MINIMUM PULSE WIDTH VIOLATION

Transition (slew): A slew is defined as a rate of change. In STA analysis the rising or falling waveforms are measured in terms of whether the transition(slew) is fast or slow. Slew is typically measured in terms of transition time, i.e. the time it takes for a signal to transition between two specific levels (1 to 0 or 0 to 1/ low to high or high to low). Transition time is inverse of the slew rate- the larger the transition time, the slower the slew and vice-versa.



In lib these transition is defined as:
#rising edge threshold:
Slew_lower_threshold_pct_rise : 20.0;
Slew_upper_threshold_pct_rise : 80.0;
#falling edge threshold:
Slew_upper_threshold_pct_fall : 80.0;
Slew_lower_threshold_pct_fall : 20.0;
These values are specified as a % of Vdd.

Rise time: The time required for a signal to transition from 20% of its (VDD) maximum value to 80% of its maximum value.

Fall time: The time required for a signal to transition from 80% of its (VDD) maximum value to 20% of its maximum value.

Propagation delay: The time required for the signal to change the inputs to its state like 0 to 1 or 1 to 0.

Clock buffer and normal buffer

Clock net is a high fan-out net and most active signal in the design. Clock buffer mainly used for clock distribution to make the clock tree. The main goal of CTS to meet skew and insertion delay, for this we insert buffer in the clock path. Now if the buffer has different rise and fall time it will affect the duty cycle with this condition tool can do skew optimization but complicates the whole optimization process as a tool has to deal with a clock with duty cycle at different flop paths. If buffer delays are the same only thing the tool has to do balance the delay by inserting buffer.

The clock buffers are designed with some special property like high drive strength, equal rise and fall time, less delay and less delay variation with PVT and OCV. Clock buffer has an equal rise and fall time. This prevents the duty cycle of clock signal from changing when it passes through a chain of clock buffers.

A perfect clock tree is that gives minimum insertion delay and 50% duty cycle for the clock. The clock can maintain the 50% duty cycle only if the rise and the fall delays and transition of the tree cells are equal.

How to decide whether we need to used buffer or inverter for building a clock tree in the clock tree synthesis stage. This decision totally depends on the libraries which we are using. The main factors which we consider to choose inverter or buffer are rise delay, fall delay, drive strength and insertion delay (latency) of the cell. In most of the library files, a buffer is the combination of two inverters so we can say that inverter will be having lesser delay than buffer with the same drive strength. Also inverters having more driving capacity than a buffer that's why most of the libraries preferred inverter over buffer for CTS.

Clock buffers sometimes have input and output pins on higher metal layers much fewer vias are needed in the clock distribution root. Normal buffer has pins on lower metal layers like metal1. Some lib also has clock buffers with input pins on high metal layers and output pins on lower metal layers. Normally clock routing is done into higher metal layers as compared to signal routing so to provide easier access to clock pins from these layers clock buffer may have pins in higher metal layers. And for normal buffer pins may be in lower metal layers.

Clock buffer are balanced i.e. rise and fall time almost the same. If these are not equal then duty cycle distortion in the clock tree will occur and because of this minimum pulse width violation comes into the picture. In clock buffer the size of PMOS is greater than NMOS.

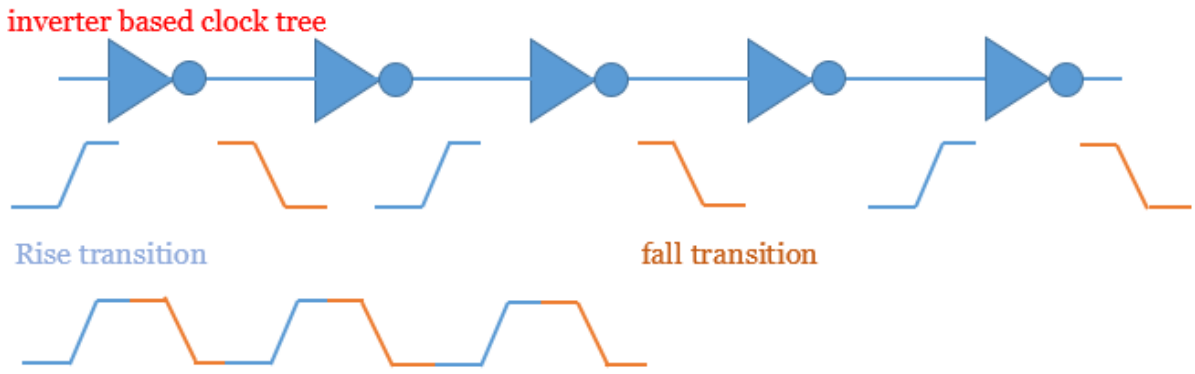
On the other hand normal buffer have not equal rise and fall time. In other words they don't need to have PMOS/NMOS size to 2:1 i.e. size of PMOS don't need to be bigger than the NMOS, because of this normal buffer is in a smaller size as compared to clock buffer and clock buffer consumes more power.

The advantage of using an inverter-based tree is that it gives equal rise and fall transition so due to that jitter (duty cycle jitter) get canceled out and we get symmetrical high and low pulse width.

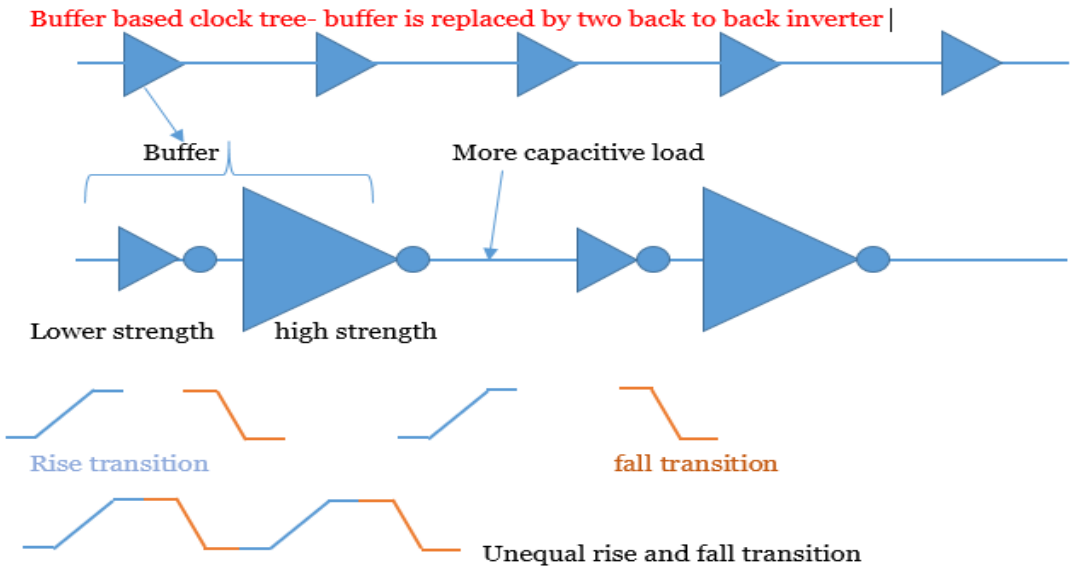
Buffer contain two inverters with unequal size in area and unequal drive strength. First inverter is of small size having low drive strength and the second buffer is of large size having high drive, strength are connected back to back as shown in figure below.

So a load of these two inverters are unequal. The net length b/w two back to back inverter is small so small wire capacitance will present here we can neglect that but for the next stage the net length is more and because of net length the capacitance is more by wire capacitance and next inverter input pin capacitance and we get unequal rise and fall time so jitter will get added in clock tree with an additional cost of more area than an inverter.

So mainly we are preferred inverter-based trees instead of the buffer based.



inverter based tree having equal rise and fall time



buffer based tree having unequal rise and fall time

Why PMOS is having bigger size than NMOS?

We know NMOS have majority charge carriers are electrons and PMOS have majority charges carriers are holes. And we also know that electrons are very much faster than holes.
Since electron mobility is greater than the hole mobility, so PMOS width must be larger to compensate and make the pull-up network more stronger. If W/L of PMOS is the same as NMOS the charging time of the output node would be higher than the discharging time because discharging time is related to the pulldown network.
So we make PMOS is of big size so that we can get equal rise and fall time.
Normal buffer are designed with W/L ratio such that sum of rise and fall time is minimum.

Normally **(R) PMOS > (R) NMOS**

(R) PMOS =3*(R) NMOS
For making equal resistance of both transistor the size of PMOS is bigger than the NMOS.

$$\mu p.\frac{Wp}{Lp}=\mu n\frac{Wn}{Ln}$$

Consider the same length for both transistor Ln=Lp

$$\mu p\ Wp = \mu n\ Wn$$

$$Wp = \left(\frac{\mu n}{\mu p}\right) Wn$$

And we know $\mu n \gg \mu p$

So $Wp \gg Wn$

The duty cycle of clock:

It is the fraction of one period of the clock during which clock signal is in the high (active) state. A period is the time it takes for a clock signal to complete an on-and-off state. Duty cycle (D) is expressed in percentage (%).

$$D = \frac{\text{the at which clock is high}}{\text{total period of clock}} * 100\%$$

Minimum Pulse width violation:

It is important for the clock signal to ensure the proper functionality of sequential and combinational cells. Ensure that the width of the clock signal is wide enough for the cell, internal operation i.e. minimum pulse width of the clock has to be maintain for proper output otherwise, the cell will go into metastable state and we will not get the correct output.

In other words clock pulse into the flop/latch must be wide enough so that it does not interfere with the correct functionality of the cells.

Minimum pulse width violation checks are to ensure that the pulse width of the clock signal for the high and low duration is more than the required value.

Basically this violation is based on what frequency of operation and Technology we are working. If the frequency of design is 1 GHz then the time period for each high and low pulse will be 0.5ns as if we consider the duty cycle is 50%.

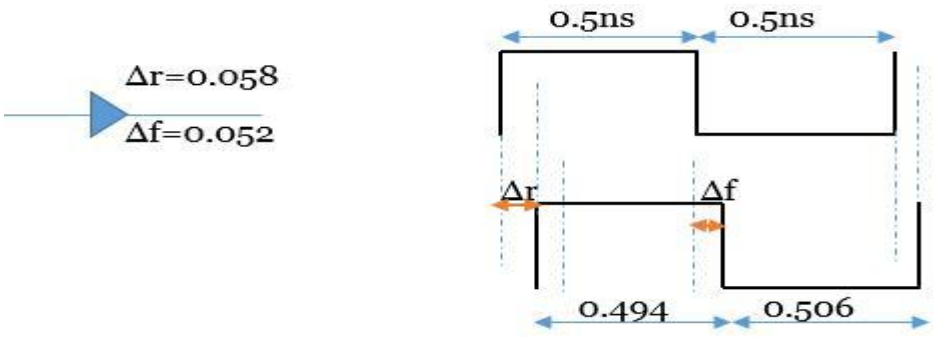
Normally we saw that in most of design duty cycle always keep 50% for the simplicity otherwise designer can face many issues like clock distortion and minimum pulse width violation. If in our design is using half-cycle path means data is launch at the positive edge and capturing at the negative edge and again minimum pulse width as rising level and fall level will not be the same and if lots of inverter and buffer will be in chain then it is possible that pulse can completely vanish.

Normally for the clock path, we use clock buffer because they have equal rise and fall delay of these buffer as compare to normal buffer having unequal delay that's why we have to check minimum pulse width.

Why the minimum pulse width violation occurs:

Due to unequal rise and fall delay of combinational cell. Let's take an example of buffer and clock signal having 1 GHz frequency (1ns period) is entering into a buffer. So for example, if the

rise delay is more than the fall delay than the output of clock pulse width will have less width for high level than the input clock pulse.



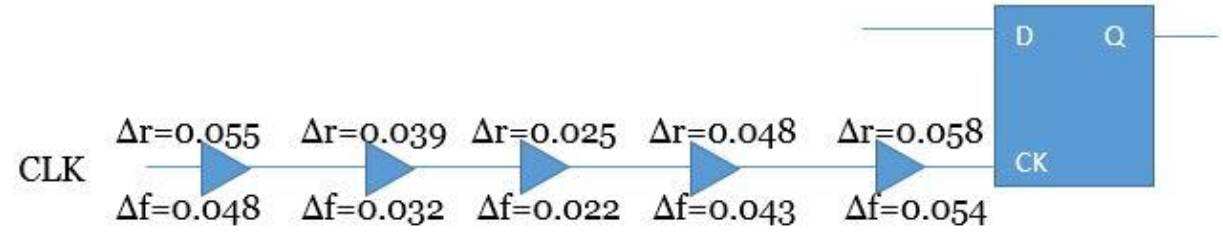
The difference b/w rise and fall time is: 0.007

High pulse: 0.5-0.006=0.494

Low pulse: 0.5+0.006=0.506

We can understand it with an example:-

Let's there is a clock signal which is pass through more numbers of buffers with different rise and fall delay time. We can calculate how it effects to the low or high pulse of the clock signal. The width of clock signal is decreasing when buffer delay is more than the pulse width.



Δr = rise delay of buffer
Δf = fall delay of buffer

As we know every buffer in the chain is taking more time to charge than to discharge. When the clock signal is propagating through a long chain of buffers, the pulse width is reduced as shown below.

We can understand by the calculation:-

High pulse width = half pulse width of clock signal– (rise delay –fall delay)
= 0.5 - (0.055-0.048) - (0.039-0.032) - (0.025-0.022) - (0.048-0.043) - (0.058-0.054) = 0.474ns

Low Pulse width = half pulse width of clock signal + (rise delay –fall delay)
= 0.5 + (0.055–0.048) + (0.039–0.032) + (0.025–0.022) + (0.048 – 0.043) + (0.058 – 0.054) = 0.526ns

Let's required value of Min pulse width is 0.410ns, Uncertainty = 90ps
Then high pulse width = 0.474-0.090 = 0.384ns
The slack is 0.384-0.410= - 0.026ns
here we can see that we are getting min pulse width violation for high pulse as total high pulse width is less than the required value.

If uncertainty we did not consider then violation will not occur in this scenario.

How to correct if violations are present in design:
We need to change the clock tree cells which have equal rise and fall delay time or use those cells they have less difference between rise and fall delays.

What are the problems occurs if pulse width violation occurs:

- Sequential data might not be captured properly, and flop can go into a metastable state.
- In some logic circuits the entire pulse could disappear and does not capture any new data.

So it is required to ensure every circuit element always gets a clock pulse greater than minimum pulse width required then only violation will not occur in the design.

There are two types of minimum pulse width checks are performed:
Clock pulse width check at sequential devices
Clock pulse width check at combinational circuits

How to report:

report_timing –check_type pulse_width

```
pt_shell> report_min_pulse_width

****************************************
Report : min pulse width
         -path_type summary
Design : middle
Version: Z-2006.12-Beta4-DEV
Date   : Wed Nov  8 09:06:01 2006
****************************************

  sequential_clock_pulse_width

                          Required        Actual
Pin                       pulse width     pulse width     Slack
------------------------------------------------------------------
ff1/CP                      10.20           10.00          -0.20  (VIOLATED)
ff2/CP                      10.20           10.04          -0.16  (VIOLATED)
ff3/CP                       2.10            2.00          -0.10  (VIOLATED)
ff3/CP                       2.10            2.00          -0.10  (VIOLATED)
ff2/CP                      10.00            9.96          -0.04  (VIOLATED)

  clock_tree_pulse_width

                          Required        Actual
Pin                       pulse width     pulse width     Slack
------------------------------------------------------------------
nand1/Z                      10.00           9.58          -0.42  (VIOLATED)
or1/B                        10.00           9.58          -0.42  (VIOLATED)
nand1/A                      10.20          10.00          -0.20  (VIOLATED)
or1/Z                        10.20          10.04          -0.16  (VIOLATED)
or1/Z                        10.00           9.96          -0.04  (VIOLATED)
```

How to define pulse width:

By liberty file (.lib):

By default all the registers in the design have a minimum pulse width defined in .lib file as this is the format to convey the std cell requirement to the STA tool.

By convention min pulse width is defined for the clock signal and reset pins.
Command name: **min_pulse_width**
In SDC file (.sdc):
set_min_pulse_width -high 5 [get_clock clk1]
set_min_pulse_width -low 4 [get_clock clk1]
If high or low is not specified then constraints applied to both high and low pulses.
NOTE:
Balanced buffers means buffer having equal rise and fall time.
Unbalanced buffers means buffer having unequal rise & fall time

Clock Tree routing Algorithms

The main idea behind using these algorithms to minimize the skew. So how we minimize the skew by using these algorithms. Distribute the clock signal in such a way that the interconnections (routing wires) carrying the clock signal to the other sub-blocks that are equal in length.

Several algorithms exist that are trying to achieve this goal (to minimize the skew).

- H-Tree
- X-Tree
- Method of Mean and Median
- Geometric Matching Algorithms
- Zero skew clock routing

The first to four algorithm techniques are trying to make minimize the length and the last one is to use the actual interconnect delay in making the skew is zero.

H-Tree

In this algorithm Clock routing takes place like the English letter H.

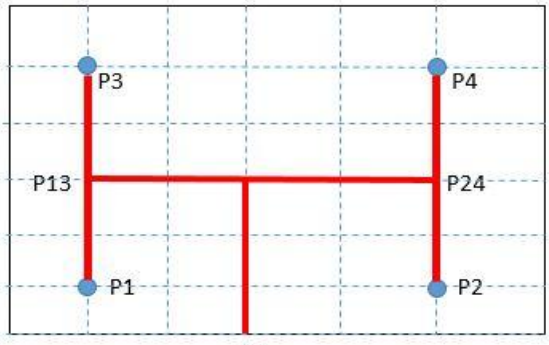
It is an easy approach that is based on the equalization of wire length.

In H tree-based approach the distance from the clock source points to each of the clock sink points are always the same.

In H tree approached the tool trying to minimize skew by making interconnection to subunits equal in length.

This type of algorithm used for the scenario where all the clock terminal points are arranged in a symmetrical manner like as in gate array are arranged in FPGAs.

In fig (a) all the terminal points are exactly 7 units from the reference point P0 and hence skew is zero if we are not considering interconnect delays.



The distance from Po to each terminal point is 7

It can be generalized to 4i. When we are going to up terminals are increased like 4, 16, and 64…and so on and regularly placed across the chip in H structure.

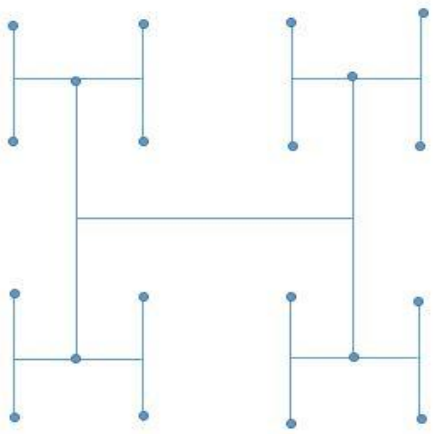


fig: H tree with 16 sink points

In this routing algorithm all the wires connected on the same metal layers, we don't need to move horizontal to vertical or vertical to horizontal on two layers.
H tree do not produce corner sharper than 900 and no clock terminals in the H tree approach in close proximity like X tree.

Advantages:
Exact zero skew in terms of distance (here we are ignoring parasitic delay) due to the symmetry of the H tree.
Typically used for very special structures like top-level clock level distribution not for the entire clock then distributed to the different clock sinks.

Disadvantages:
Blockages can spoil the symmetry of the H tree because sometimes blockages are present on the metal layers.
Non-uniform sink location and varying sink capacitance also complicate the design of the H tree.

X-tree

If routing is not restricted to being rectilinear there is an alternative tree structure with a smaller delay we can use. The X tree also ensures to skew should be zero. X-tree routing algorithm is similar to H-tree but the only difference is the connections are not rectilinear in the X tree-based approach.

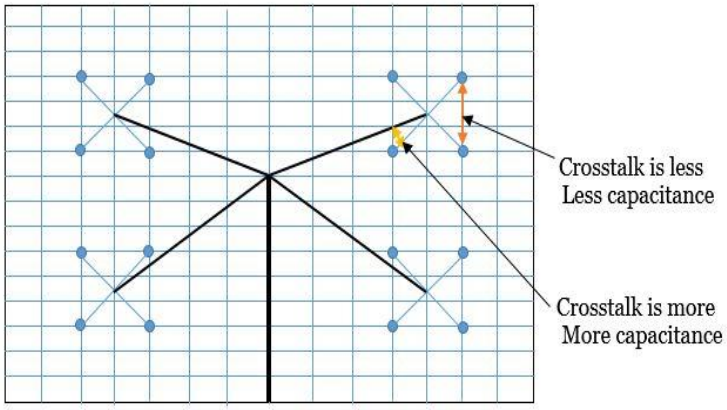


Fig: X tree with 16 points

Although it is better than the H tree but this may cause crosstalk due to close proximity of wires. Like H tree this is also applicable for top-level tree and then feeding to the next level tree.

Disadvantages:

Cross Talk due to adjacent wires

Clock Routing is not rectilinear

Both of the H Tree and X tree approach basically designed for a four array tree structure. Each of the 4 nodes connected to the other 4 nodes in the next stages so the number of terminal points or sink will grow as the power of 4 like 4,16 and 64 and so on.

These two methods basically did not consider the exact location of the clock terminals it independently create the clock tree and produce a regular array of sink locations across the surface of the chip.But in other approaches, we did not ignore the exact location of the actual clock terminal points so now the question is how we what these approaches will do for exact location.They look at where we required the clocks to be sent w.r.t location and try to build the tree systematically and that tree does not look like the H tree and X tree.

Method of mean & median (MMM) algorithm:

Method of mean and median follows the strategy similar to the H-tree algorithm, but it can handle sink location anywhere we want.

Step 1: It continuously partitions the set of terminals into two subsets of equal parts (median) (As Fig.)
Step2: connects the center of mass of the whole set (module) to the center of masses of the two partitioned subset (mean).

How the partitioning is done?

Let Lx denoted the list of clock points sorted accordingly to their x-coordinates
Let Px be the median in Lx

-assign points in the list to the left of Px and Lx

-assign the remaining points to Pr.

Next, we go for a horizontal partition where we partition a set of points into two sets Pb &Pt
This process is repeated iteratively.

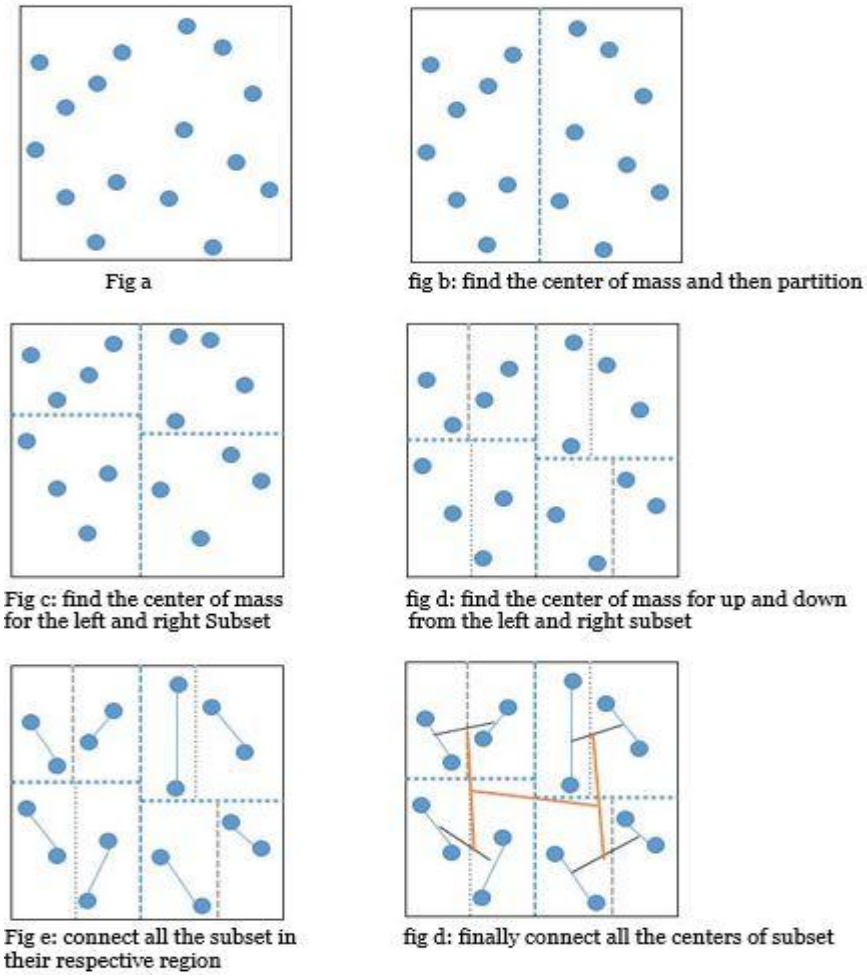


Fig a

fig b: find the center of mass and then partition

Fig c: find the center of mass for the left and right Subset

fig d: find the center of mass for up and down from the left and right subset

Fig e: connect all the subset in their respective region

fig d: finally connect all the centers of subset

fig: MMM algorithm

This algorithm ignores the blockages and produces a non-rectilinear (not regularly spaces) tree. Here some wire may also interact with each other.
It is a top-down approach as we are partitioning till each partition consist of a single point.

Recursive geometric Matching Algorithm (RGM)

This is another binary tree-based routing algorithm in which clock routing is achieved by constructing a binary tree using exclusive geometry matching.

Unlike the Method of mean & median (MMM) algorithm which is top-down and this is bottom-up fashion. Here we used the concept of recursive matching.

To construct a clock tree by using recursive matching determines a minimum cost geometric matching of n sink nodes.
The Center of each segment is called tapping point and the clock signal is provided at this point then the signal will arrive at the two endpoints of the segment with zero skew.

Find a set of n/2 line segments that match n endpoints and minimum total length. After each matching step a balance or tapping point is found on each matching segment to maintain zero skew to the related sinks. These set of n/2 tapping point then forms the input to the next matching step.

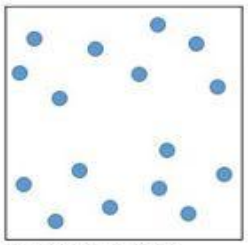


Fig a: set of n sinks

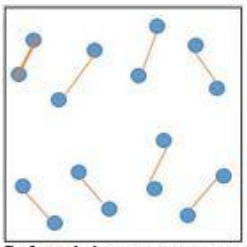


fig b: minimum cost geometric matching

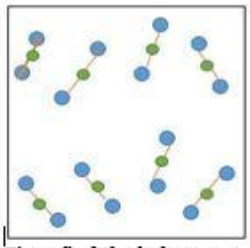


Fig c: find the balance or
Tapping points (the point that
Achieve zero skew in the
Sub tree not always a mid-point)

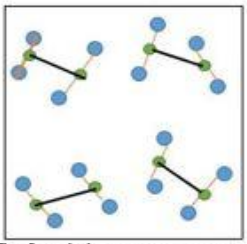


fig d; minimum cost geometric matching

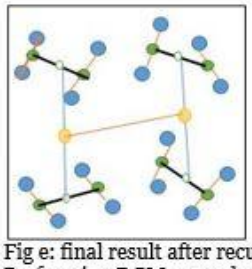


Fig e: final result after recursively
Performing RGM on each subset

Fig:- RGM algorithm

This bottom-up approach gives a better result than a top-down approach.

STA - STA,DTA,TIMING ARC, UNATENESS

What is Timing Analysis?

We checked whether the circuit meets all its timing requirements. The timing analysis is used to refer to either of these two methods - static timing analysis, or the timing simulation (dynamic timing analysis).

What is static Timing Analysis?

STA is the technique to verify the timing of a digital design. The STA analysis is the static type and in this analysis of the design is carried out statically and does not depend upon the data values being applied at the input pins.

The more important aspect of static timing analysis is that the entire design (typically specified in hardware descriptive languages like VHDL or VERILOG) is analyzed once and the required timing checks are performed for all possible timing paths and scenarios related to the design. Thus, STA is a complete and exhaustive method for verifying the timing of a design.
In STA whole design is divided into a set of timing paths having start and endpoints and calculate the propagation delay for each path and check whether there is any violation in the path and report it.

What is Dynamic Timing Analysis?

DTA is a simulation-based timing analysis where a stimulus is applied on input signals, and resulting behavior is observed and verified using the Verilog test bench, then time is advanced with new input, the stimulus applied, and the new behavior is observed and verified and so on. It is an approach used to verify the functionality as well as the timing of the design.

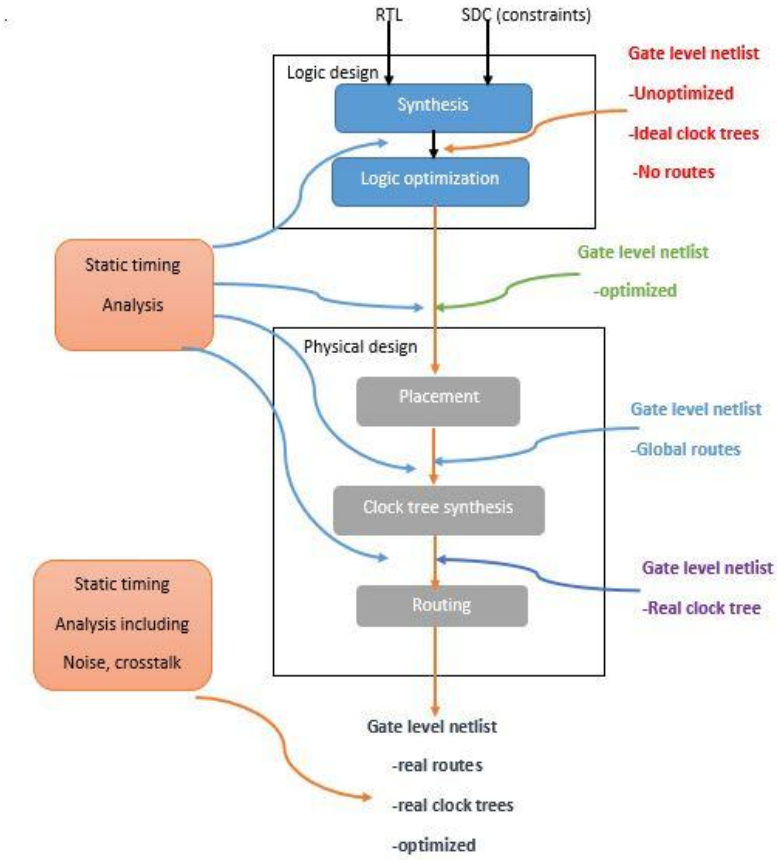
This analysis can only verify the portions of the design that get exercised by stimulus (vectors). Verification through timing simulation is only as exhaustive as the test vectors used. To simulate and verify all the timing paths and timing conditions of a design with 10-100 million gates are very slow and the timing cannot be verified completely. Thus, it is very difficult to do exhaustive verification through simulation.

Why Static Timing Analysis?

- STA is a complete and exhaustive verification of all timing checks of a design.
- STA provides a faster and simpler way of checking and analyzing all the timing paths in a design for any timing violations.
- Day by day the complexity of ASIC design is increasing, which may contain 10 to 100 million gates, the STA has become a necessity to exhaustively verify the timing of a design.

Design flow for Static timing Analysis:

In ASIC design, the static timing analysis can be performed at many stages of the implementation. STA analysis first done at RTL level and at this stage more important is to verify the functionality of the design not timing.



Once the design is synthesized from RTL to Gate – level, then STA analysis is used for verifying the timing of the design. STA is also performing logic optimization to identify the worst/critical timing paths. STA can be rerun after logic optimization to see whether there are failing paths are still remaining that need to be optimized or to identify the worst paths in the design.

At the start of physical design (PD) stages like floorplan and placement, the clock is considered as an ideal which means the delay from clock to all the sink pins of the flip flop is zero (i.e. clock is reaching to all the flip flop at the same time). After placement, in the CTS stage clock tree is built and STA can be performed to check the timing. During physical design, STA can be performed at each and every stage to identify the worst paths.

In the logic design phase, interconnect is ideal since there is no physical information related to the placement of Macros and standard cells. In this stage, to estimate the length of interconnect we used WLM (wire load model) which provides estimated RC interconnect length based on the fan-out of the cell.

In the physical design stage, we have the information about the placement of macros and standard cells and these cells are connected by interconnect metal traces. The parasitic RC of the metal affects the delay and power dissipation in the design.
Before the routing is finalized this phase is called the Global route phase, the implementation tool used to estimate the routing length and the routing estimates are used to determine resistance and capacitance parasitic that are needed to calculate the wire delays. Before the routing stages we are not focused on the effect of coupling. After the detailed routing complete, actual RC values obtained from the extraction tool (used to extract the detailed parasitic from the design) and the effect of coupling also analyzed.

Limitations of STA:

1. If all the flip-flops are in reset mode into their required values after applying synchronous or asynchronous rest this condition cannot be checked using static timing analysis.
2. STA is dealing with only known values like logic-0 and logic 1 (or we can say low and high). If any unknown value X in the design comes then this value will not check by using STA.
3. Ensure that correct clock synchronizer is present whenever there are asynchronous clock domain crossing is present in the design otherwise STA does not check if the correct clock synchronizer is being used.
4. If the design having digital and analog blocks then the interface between these two blocks will not handle by STA because STA does not deal with analog blocks. Some verification methodologies are used to ensure the connectivity between these kinds of blocks.
5. STA verifies the all the timing paths included that timing path also which does not meet all the requirements and even though logic may never be able to propagate through the path these timing paths are false paths. So we have to give proper timing constraints for false path and multicycle paths then only STA qor result will be better.

Standard cells:

Most of the complex functionality in the chip is designed using basic blocks of AND, OR, NAND, NOR AOI, OAI cells and flip flops. These blocks are predesigned and called standard cells.
The functionality and timing of these standard cells are pre-characterized and available to the designer in the form of standard cell libraries and use these blocks according to the requirement.

Propagation delay: **https://www.physicaldesign4u.com/2020/03/cts-part-iii-clock-buffer-and-minimum.html**

Transition (slew): **https://www.physicaldesign4u.com/2020/03/cts-part-iii-clock-buffer-and-minimum.html**

Timing arcs and unateness:

Timing arcs:

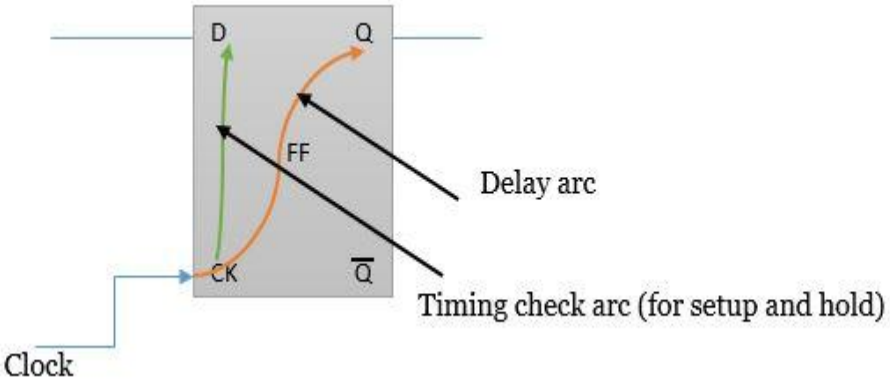
The timing arc means a path from each input to each output of the cell. Every combinational logic cell has multiple timing arcs. Basically, it represents how much time one input takes to reach up to output (eg. A to Y and B to Y). Like if we see AND, OR, NAND, and NOR cell as shown in the figure. In sequential cells such as flip flop have timing arcs from clock to the outputs and clock to data input.

Timing arcs can be further divided into two categories – cell arcs and net arcs.

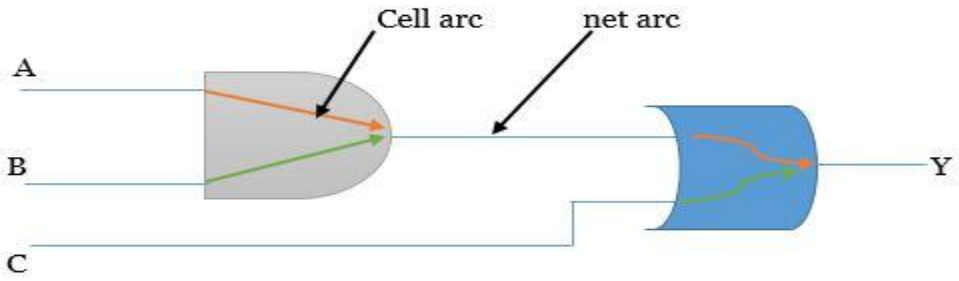
Cell arcs: This arc is between an input pin and an output pin of a cell i.e. source pin is an input pin of a cell and sink pin is the output pin of the same cell. Cell arcs can be further divided into sequential and combinational arcs.

Combinational arcs are between an input and output pin of a combinational cell or block.

Sequential arcs are between the clock pin and either input or output pin. Setup and hold timing arcs are between the input data pin and clock pin of flip flop and are termed as timing check arcs as they constrain a form of the timing relationship between a set of signals. Sequential delay arc is between clock pin and output Q pin of FF. An example of a sequential delay arc is clk to q is called delay arc and clk to D input is called timing check arcs in sequential circuits.



Net arcs: These arcs are between driver (cell) pin of a net and load pin of a net i.e. the source pin is output pin of one cell and the sink pin is input pin of another cell. Net arcs are always a delay timing arcs.



Unateness:

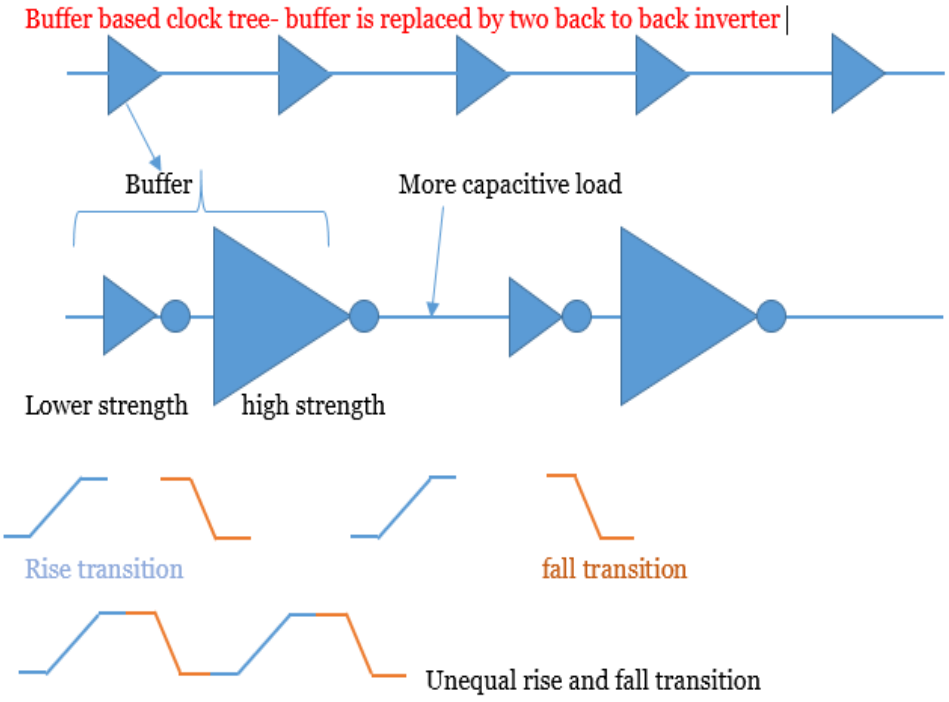
Each timing arcs has a timing sense that means how the output changes for different types of transitions on input this is called unateness. Unateness is important for timing as it specifies how the output is responding for the particular input and how much time it will take.

Timing arc unateness are of three types:

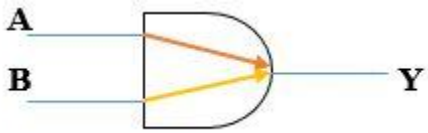
Positive unate: If a rising transition on the input gives the output to rise and falling transition on the input gives the output to fall i.e. there is no change in transition of input and output then that timing arc is called positive unate.

Example: Buffer, AND, OR gate

Buffer: There are two-timing arc in buffer. First is rising input A to Y which gives rising output Y and second is for falling input A to Y that gives falling output Y i.e. what type of edge is giving to the input we got same at the output (output is constant).



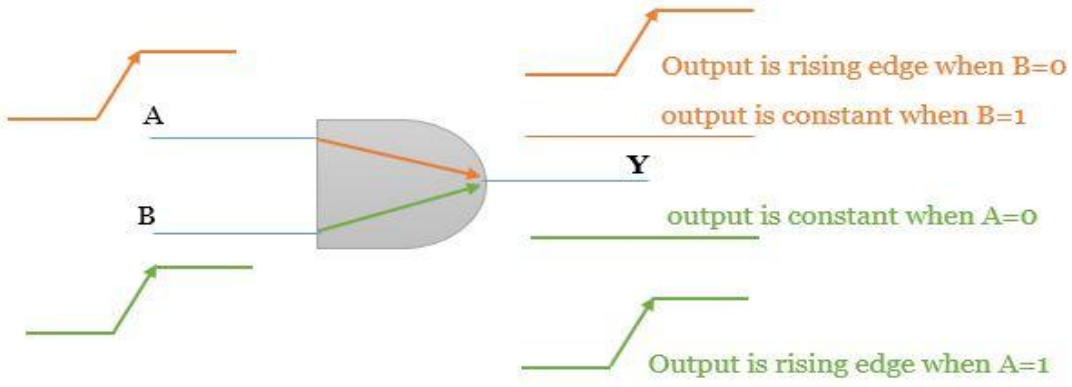
AND gate: There are four timing arcs.
Input A to output Y for rising edge
Input B to output Y for rising edge
Input A to output Y for falling edge
Input B to output Y for falling edge



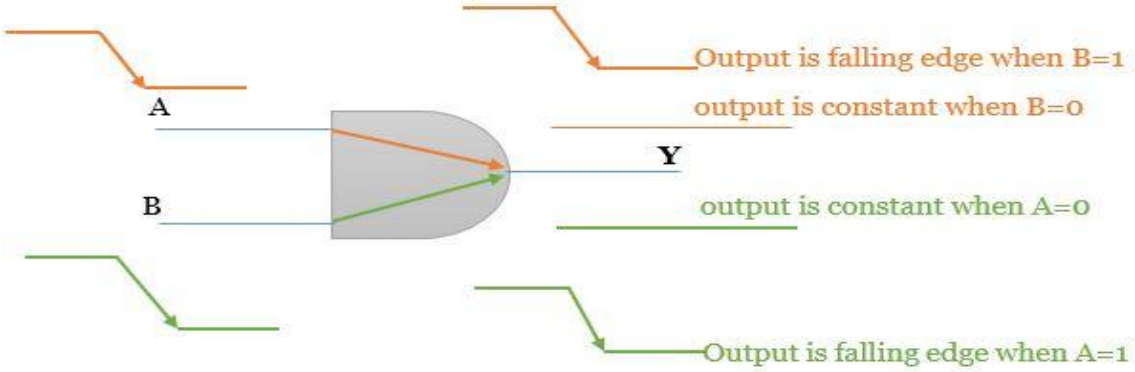
The truth table of AND gate is shown in figure.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Check for **rising edge,** from the truth table we can see that
If A = 0, and B (is going from 0 to 1): Y is constant at 0 (no change)
If A = 1, and B (is going from 0 to 1): Y is changed from 0 to 1
If B = 0, and A (is going from 0 to 1): Y is constant at 0 (no change)
If B = 1, and A (is going from 0 to 1): Y is changed from 0 to 1.



Check for the **falling edge,** from the truth table we can see that
If A = 0, and B (is going from 1 to 0): Y is constant at 0 (no change)
If A = 1, and B (is going from 1 to 0): Y is changed from 1 to 0
If B = 0, and A (is going from 1 to 0): Y is constant at 0 (no change)
If B = 1, and A (is going from 1 to 0): Y is changed from 1 to 0.



Negative unate: If a rising transition on the input gives the output to fall and falling transition on the input gives the output to rise i.e. there is a change in transition of input and output then that timing arc is called negative unate.

Example: inverter, NAND, NOR gate

Inverter: There are two-timing arc in inverter. First is rising input A to Y which gives falling output Y and second is for falling input A to Y that gives rising output Y i.e. what type edge is giving to the input we got opposite of that at the output (output is inverted).

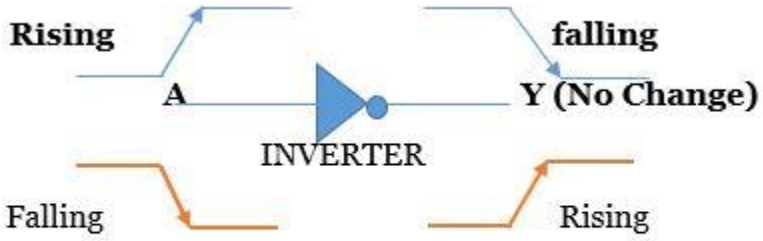
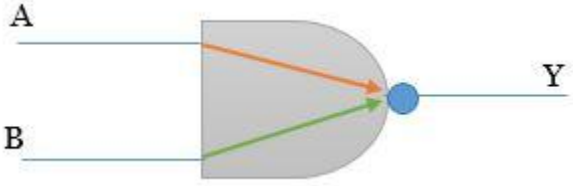


Fig: negative unate

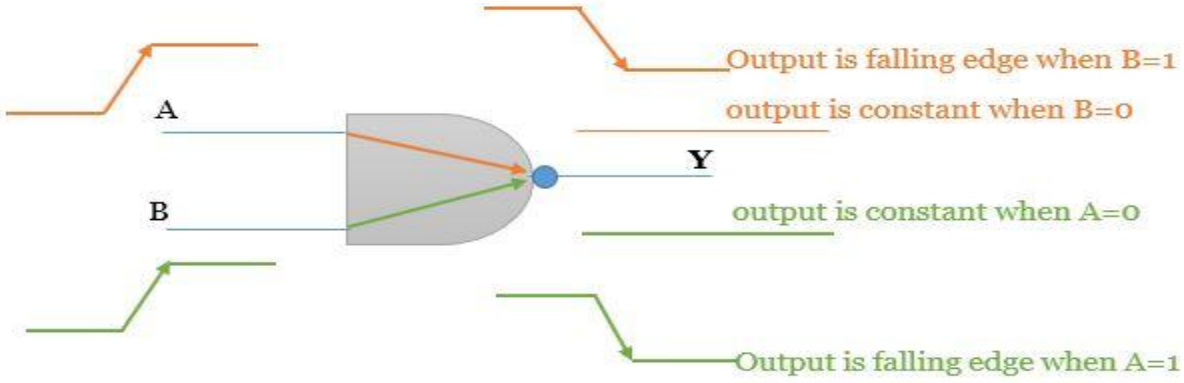
NAND gate: There are four timing arcs.
Input A to output Y for rising edge
Input B to output Y for rising edge
Input A to output Y for falling edge
Input B to output Y for falling edge



The truth table of NAND gate is shown in figure.

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

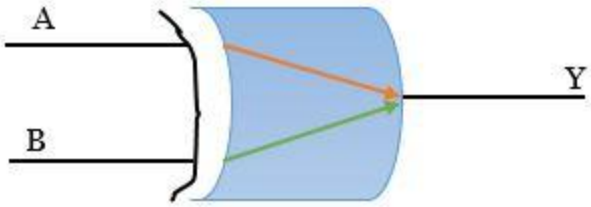
Check for the **rising edge**, from the truth table we can see that
If A = 0, and B (is going from 0 to 1): Y is constant at 1 (no change)
If A = 1, and B (is going from 0 to 1): Y is changed from 1 to 0
If B = 0, and A (is going from 0 to 1): Y is constant at 1 (no change)
If B = 1, and A (is going from 0 to 1): Y is changed from 1 to 0.



Check for the **falling edge**, from the truth table we can see that
If A = 0, and B (is going from 1 to 0): Y is constant at 1 (no change)
If A = 1, and B (is going from 1 to 0): Y is changed from 0 to 1
If B = 0, and A (is going from 1 to 0): Y is constant at 1 (no change)
If B = 1, and A (is going from 1 to 0): Y is changed from 0 to 1.

Non-unate: The output transition cannot be determined by not only the direction of an input but also depends on the state of the other inputs.

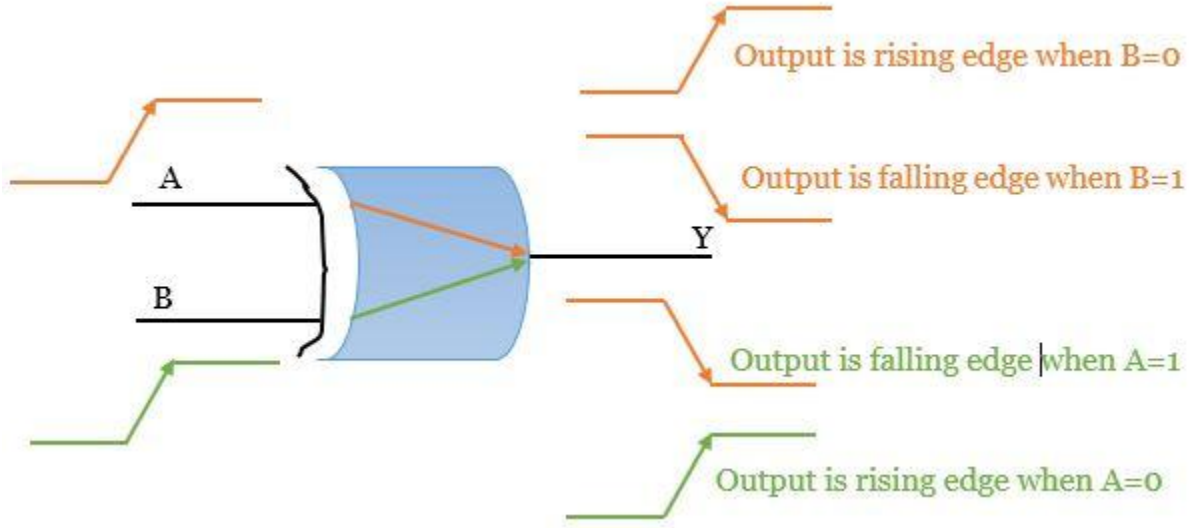
Example: XOR, XNOR gate **XOR gate:** There are four timing arcs.
Input A to output Y for rising edge
Input B to output Y for rising edge
Input A to output Y for falling edge
Input B to output Y for falling edge



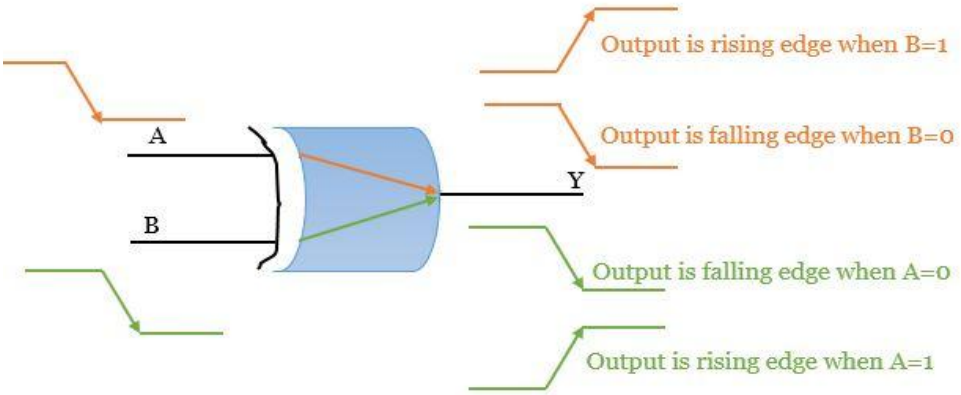
The truth table of XOR gate is shown in figure

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Check for the rising edge, from the truth table we can see that
If A = 0, and B (is going from 0 to 1): Y is changed from 0 to 1.
If A = 1, and B (is going from 0 to 1): Y is changed from 1 to 0.
If B = 0, and A (is going from 0 to 1): Y is changed from 0 to 1.
If B = 1, and A (is going from 0 to 1): Y is changed from 1 to 0.



Check for the falling edge, from the truth table we can see that
If A = 0, and B (is going from 1 to 0): Y is changed from 1 to 0.
If A = 1, and B (is going from 1 to 0): Y is changed from 0 to 1.
If B = 0, and A (is going from 1 to 0): Y is changed from 1 to 0.
If B = 1, and A (is going from 1 to 0): Y is changed from 0 to 1.

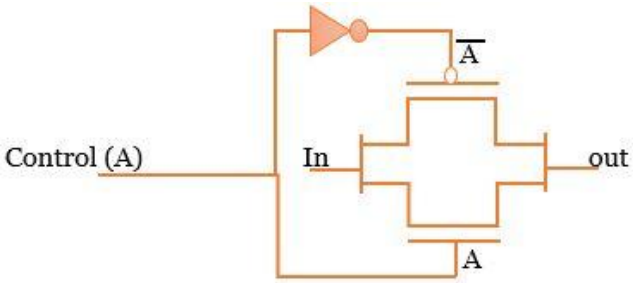


From where we can get timing arcs:

We know that all the cell information related to timing functionality is present in the library (.lib). The cell is a function of the input to output. For all the combinations of input and outputs and rising and falling conditions of inputs are defined in the .lib file. In case you have to read SDF, this delay is picked from SDF (Standard Delay Format) file. The net arcs taken from the parasitic values that are given in SPEF (Standard Parasitic Exchange Format) file, or SDF.

STA- TRANSMISSION GATE,D LATCH, DFF,SETUP &HOLD

Before going to understand the setup and hold timing we should have to know about D latch and D FF and D latch and D FF is made up of transmission gate and inverters. So in this post, I will cover transmission gate, D LATCH, D FF,
setup up and hold time.



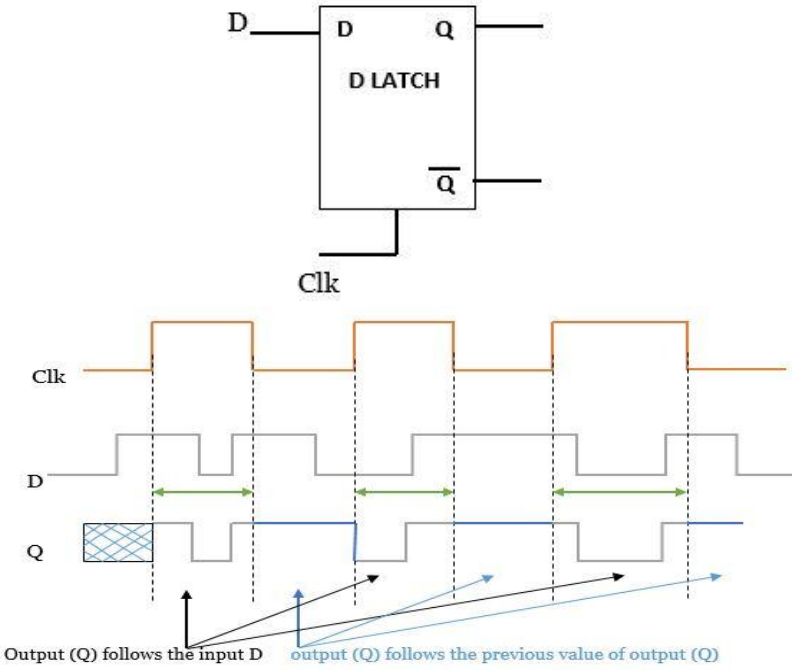
Truth table

Control (A)	PMOS	NMOS	IN	OUT
1	0	1	0	0
			1	1
0	1	0	0	Z
			1	

NOTE: PMOS is on when gate input is 0.
NMOS is on when gate input is 1.

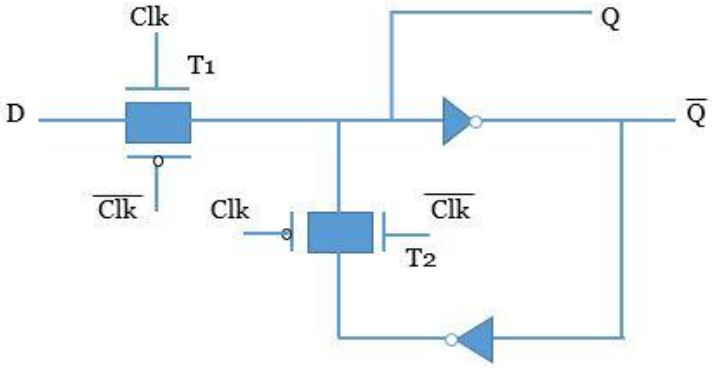
Working: When control is high (1) from the truth table we can see both transistors are ON at the same time and whatever is applied to the input we got at the output.
When control is low (0) from the truth table we can see both transistors are OFF at the same time and whatever is applied to the input is not reached to the output so we got high impedance (Z) at the output.

D latch:
The latch is a level-sensitive device and it is transparent when the clock is high if it is a positive level-sensitive latch and when the clock is low it is called negative level-sensitive latch.

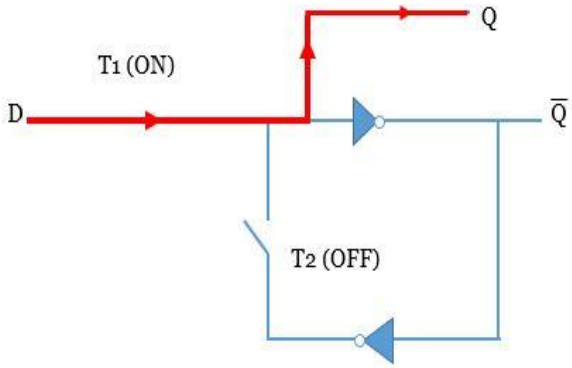


Positive triggered D latch waveform

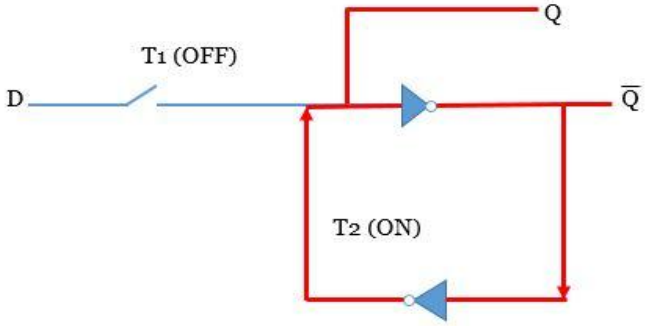
Positive D latch using transmission Gate:
It consists of two transmission gates and two inverters.



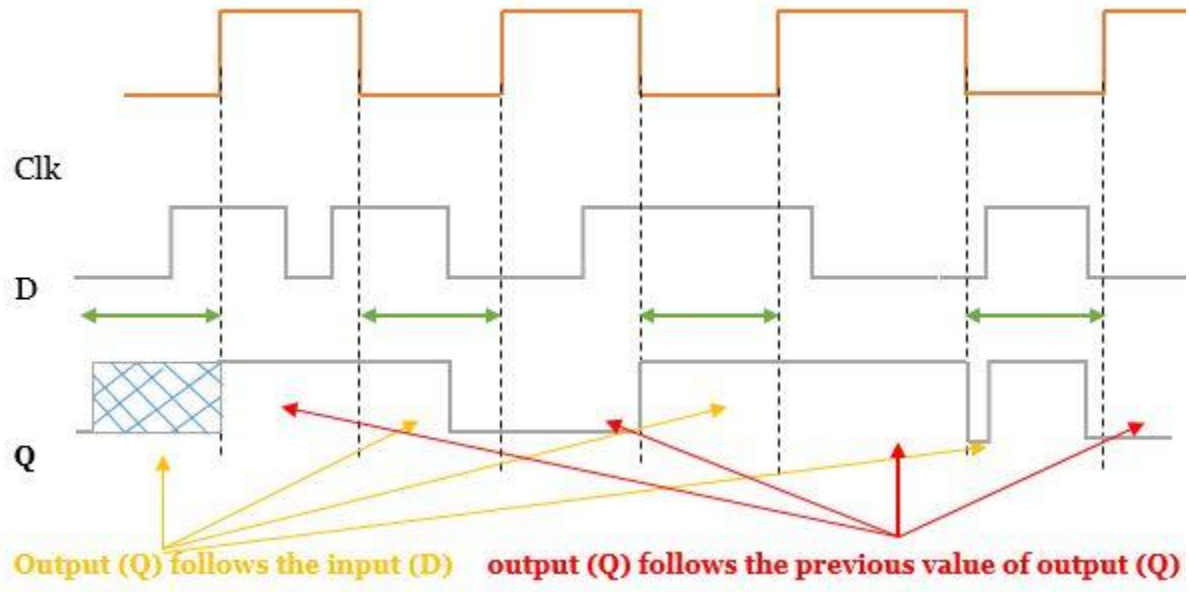
When Clk = high (1) T1 is ON and T2 is OFF, so output (Q) directly follows the input (D).



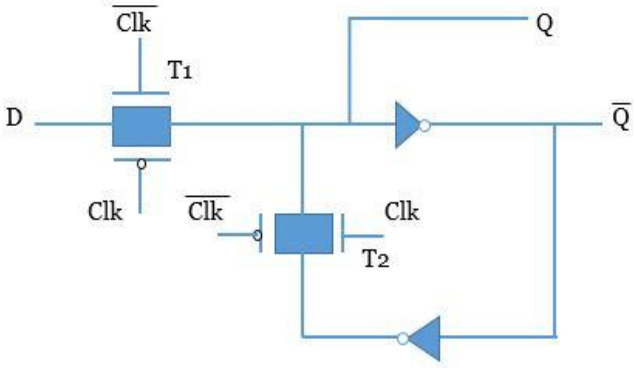
When Clk = low (0) T1 is OFF and T2 is ON, now new data entering into the latch is stopped and we get only previously-stored data at the output.



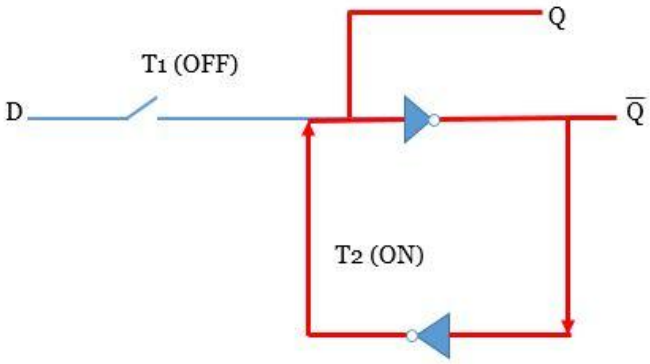
Negative D latch using transmission Gate:
It is also consist of two transmission gate and two inverters. It is working in an exactly opposite manner of the positive level-sensitive D latch.



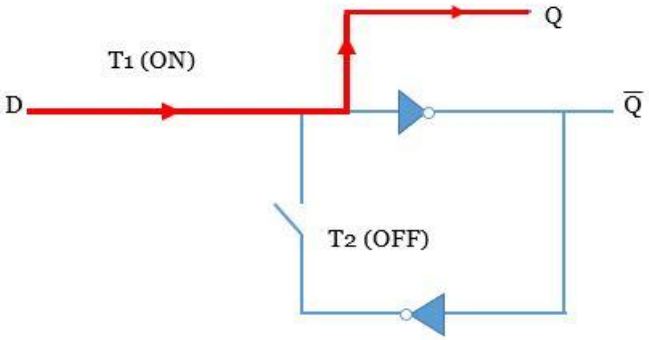
Negative triggered D latch waveform



When Clk = low (0) T1 is ON and T2 is OFF, so output (Q) directly follows the input (D).

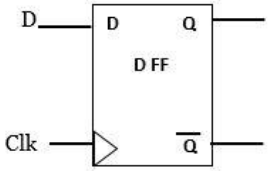


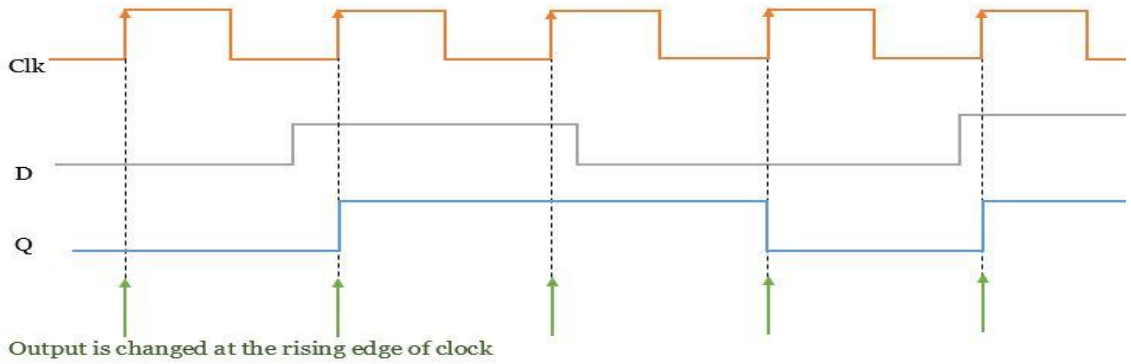
When Clk = high (1) T1 is OFF and T2 is ON, now new data entering into the latch is stopped and we get only previously-stored data at the output.



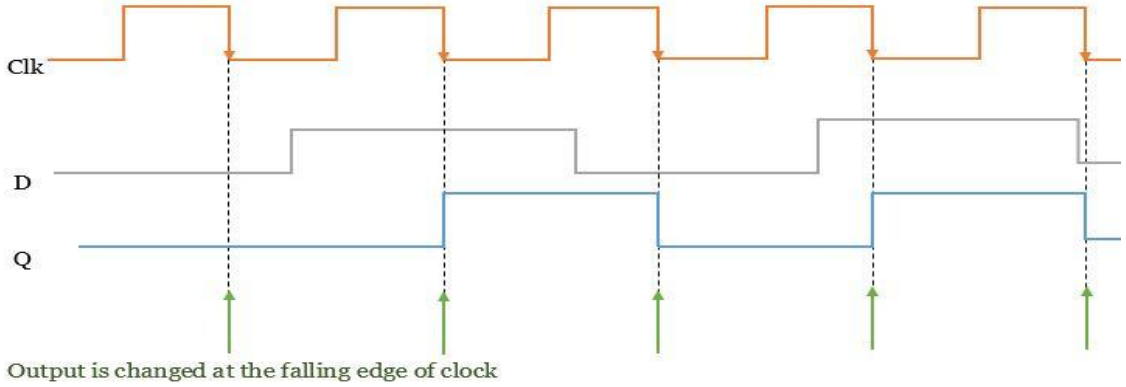
D Flip flop:

A D flip flop is an edge-triggered device which means the output (Q) follows the input (D) only at the active edge (for positive rising edge) of the clock (for the positive edge-triggered) and retain the same value until the next rising edge i.e. output does not change between two rising edges, it should be changed only at the rising edge.





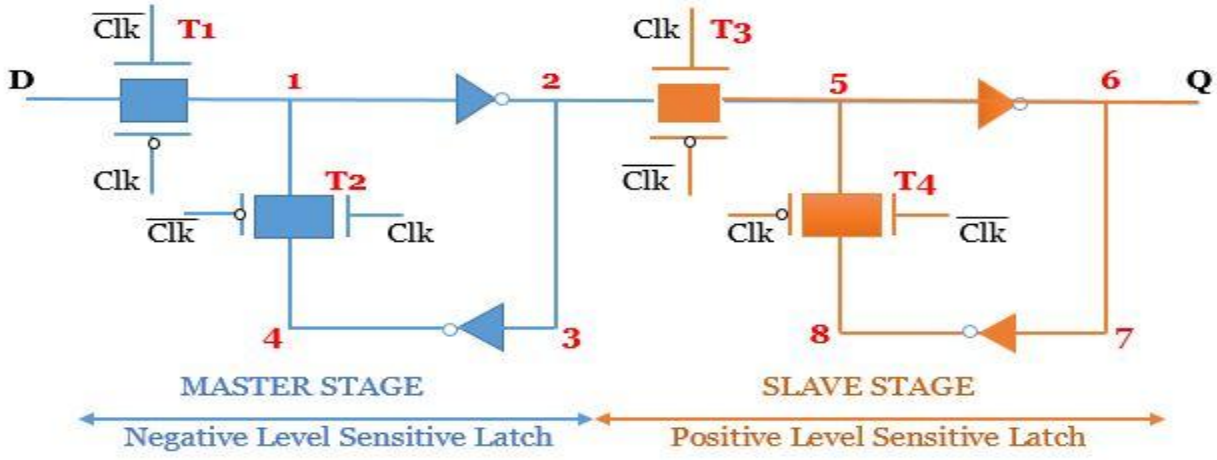
Positive edge-triggered D FF waveform



Negative edge-triggered D FF waveform

D Flip flop using a transmission gate:

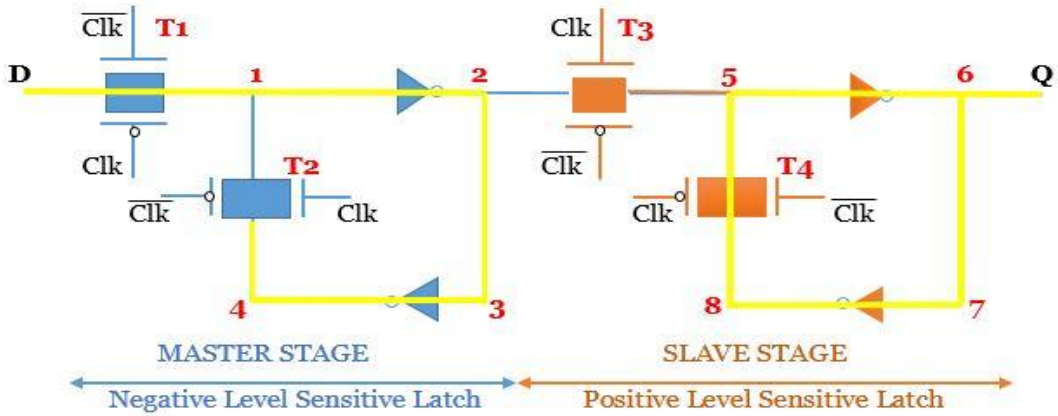
It is a combination of negative level-sensitive latch and positive level-sensitive latch that giving an edge-sensitive
device. Data is change only at the active edge of the clock.



Positive edge-triggered D FF using Transmission gate

when **Clk= LOW (0)** T1, T4 is ON and T2, T3 is OFF.

New data (D) is continuously entering through T1 and getting stored till the edge of T2 (path is D-1-2-3-4 and at node 4 it stops) it cannot pass through T2 and T3 transmission gate because they are off. This operation for the master latch. For slave latch it keeps retaining the previously stored value of output (Q) (path is 5-6-7-8-5).



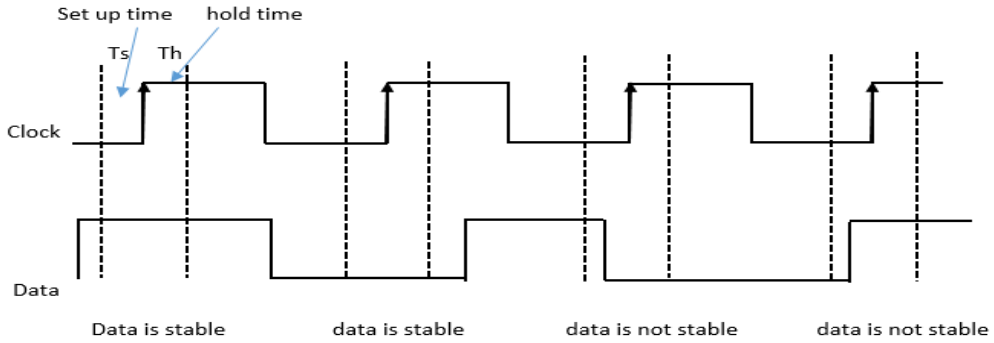
when Clk is high

Again if Clk is low the master latching circuit is enabled and there is no change in the output. Any changes in input is reflected at node 4 which is reflected at the output at the next positive edge of the clock.

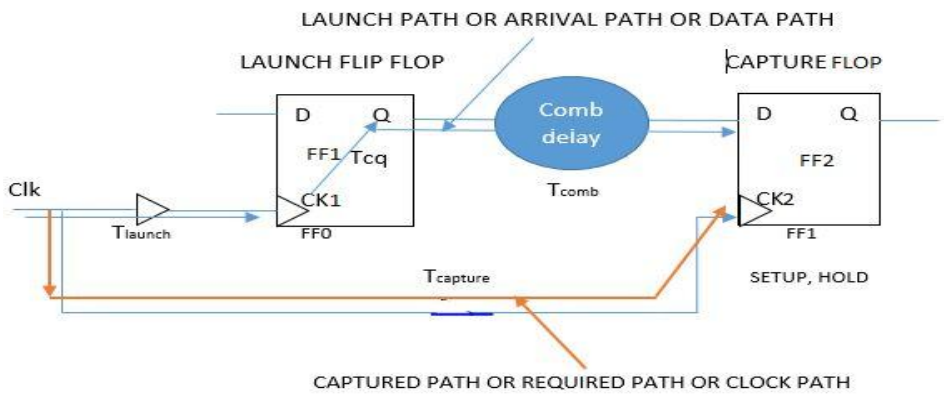
So we can say that if D changes, the changes would reflect only at node 4 when the clock is low and it will appear at the output only when the Clk is high.

Setup time:

The minimum time for which the data (D) should be stable at the input before the active edge of clock arrival.



The data is launched from FF1 and captured at the FF2 at the next clock edge. The launched data should be present at the D pin of capture flop at least setup time before the next active edge of the clock arrives.



So total time to propagate the data from launch to capture flop = one time period (T) –Tsu
This is the required time for the data travel from launch to capture flop.

And how much time it does take data to arrive at the D pin of capture flop is =Tcq (clock to Q delay of FF1) + Tcomb (combinational delay). This is called arrival time.
So condition for setup timing to not violate

$$\underbrace{Tcq + Tcomb}_{\text{Arrival time}} < \underbrace{T - Tsu}_{\text{Required time}}$$

The time which is taken by data to actually Arrive at the capture flop.

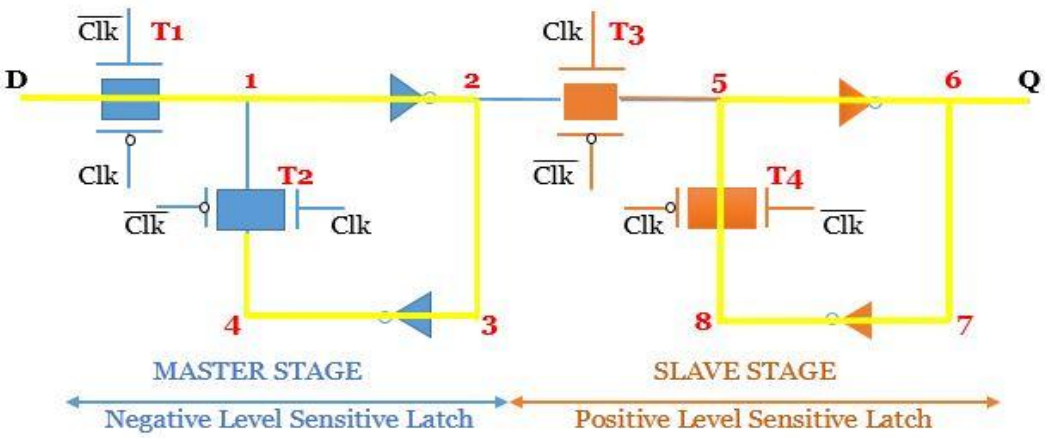
within this time data should arrive at the capture flop

RT > AT
Slack = RT –AT
Slack =+ve (no violation)
=-ve (setup violation)

Now, what is Tsu (setup time)? How do we determine and how much should be the setup time of flip flop and from where we can find this.

Now, what is Tsu (setup time)? How do we determine and how much should be the setup time of flip flop and from where we can find this.

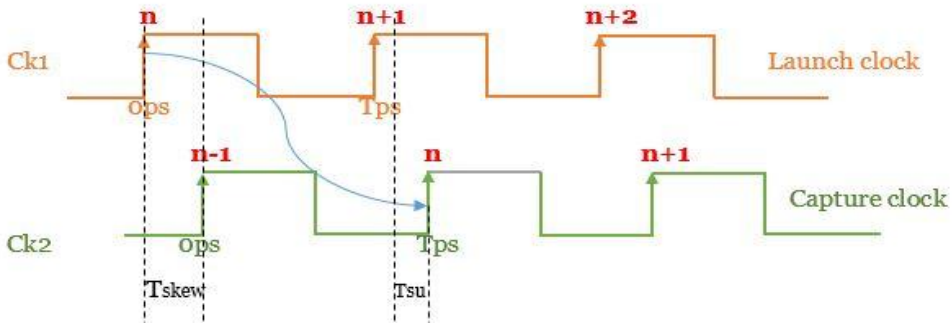
When the CLK is low the input (D) is following the path D-1-2-3-4 and it will take some time to reach at the node 4 that time we will call setup time.



What happens if data (D) is not stable for the setup time before the next active edge of the clock arrives?

So now when the clock turns high the data which has to be launched should be present at node 4 but since the data is slow it would not get enough time to travel till node 4 and the data (D) is still be present somewhere between node 2 and 3 (let's say) so we don't know which data will be launched at the rising edge and output will be indeterminate because data is not reached at node 4 yet i.e. data is late.

If skew is present in the design:



The required time (RT) is = T – Tsu + Tskew

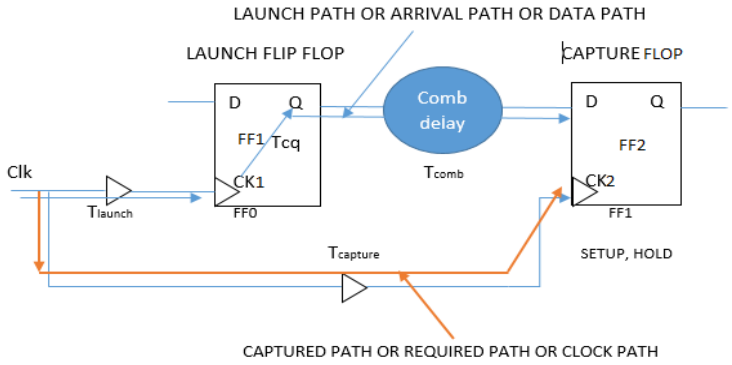
If there is a positive skew it means we are giving more time to data to arrive at D pin of capture FF. so positive skew is good for setup but bad for hold.

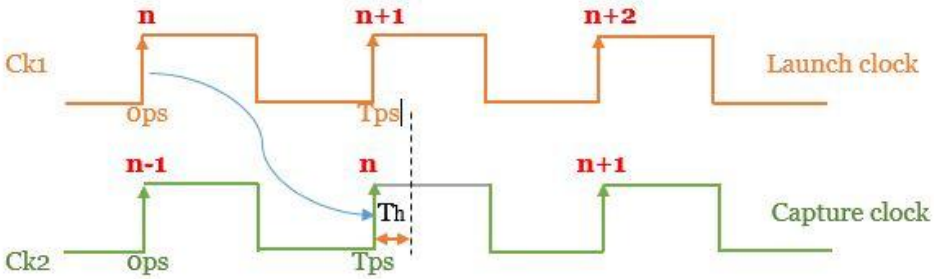
Tskew is positive or negative depending on the capture clock it comes fast or slow than the launch clock.

Positive skew: if the capture clock comes late than the launch clock.
Negative skew: if the capture clock comes early than the launch clock.
for more information about skew follow the below link
https://www.physicaldesign4u.com/2020/04/skew-latency-uncertaintyjitter.html

Hold time:

The minimum time for which the data (D) should be stable at the input after the active edge of clock has arrived.





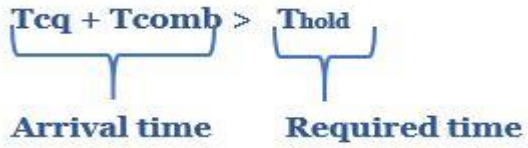
Data is launched from FF1 at 0sec and this data should be captured at FFF2 after one time period (T). The hold time is to make sure that the current data (n) which is being captured at the FF2 should not be corrupted by the next data (n+1) which has been launched from the launch flop at the same edge.

Why do we check to hold at the same edge itself?

Because this same edge is going to both the flip flops if at this edge The capturing flop FF2 is capturing the current data (n) at this same edge itself the launch flop FF1 is launching the next data (n+1) so the whole check is to make sure that this new data (n+1) which is being launched at the same edge from the launch flop FF1 should not come so fast that it corrupts the current data (n) which is being captured at the capture flop at the same edge.

The arrival time of this $(n+1)^{th}$ data should at least be greater

than the Thold time of capture flop FF2. Basically this current data (n) should be held for enough time for it to be captured reliably, that enough time is called hold time.

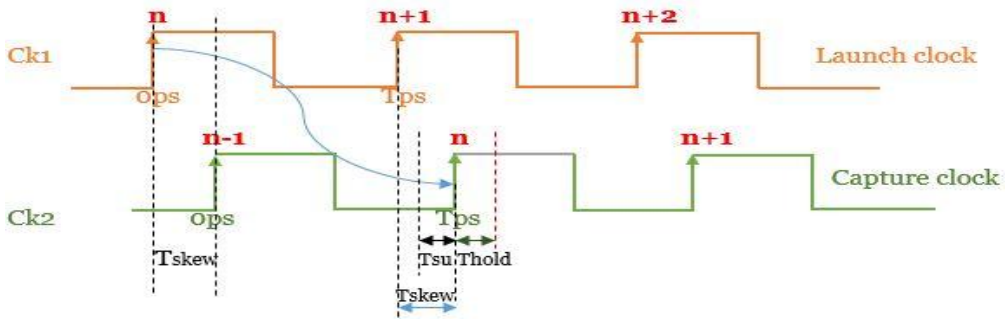


Arrival time Required time

Hold slack = AT – RT
If hold slack = +ve (No violation)
= -ve (hold violation)
If arrival time is less that means data coming is very fast (or early) so hold violation occurs.

If positive skew is present:

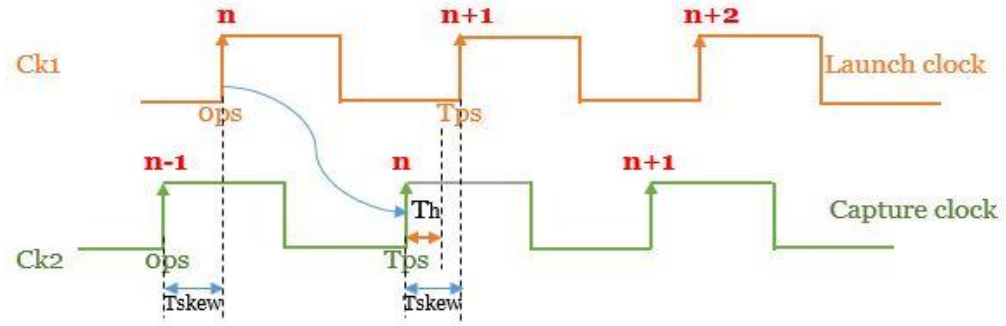
It means the next data (n+1) will be launched early from the launch flop FF1 and till now the capture clock is not reached to the capture flop FF2 so the data (n) also did not have to capture yet, but this nth data has to be stable at the capture clock for Tskew+ Thold time otherwise data n will be corrupted. So we can say +ve skew is bad for hold.



Positive skew

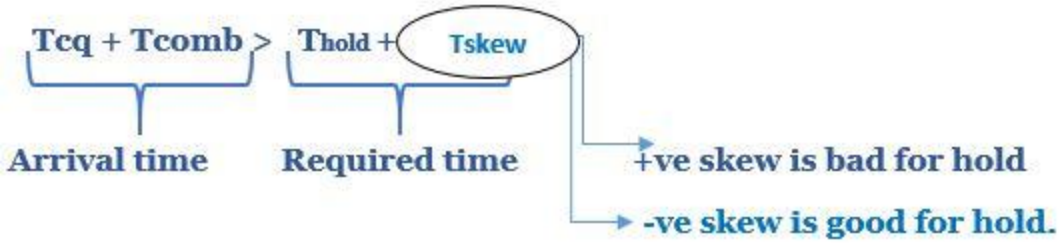
If negative skew is present:

it means the data (n) is being captured at captured flop FF2 early but by the time (n+1) data will not be getting launched from the launched flop FF1, so the data (n) got enough time to be held at the input for it to be captured reliability but till now the launch flop did not launch (n+1) data. So negative skew is good for hold.



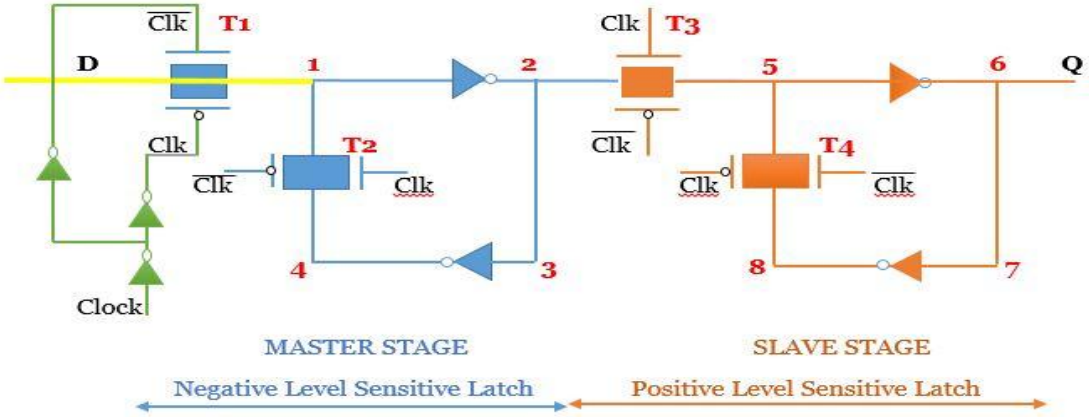
Negative skew

so the condition for hold time with skew,



Hold time (Thold) of flip flop:

For working of this structure please read DFF using the transmission gate.



The Clock is turning from low to high the T1 and T4 transmission gate is turned off and stop entering the new data (D) into the device but transmission gate T1 does not turn off the immediately it will take some time to become OFF because the clock (Clk) is passes through many buffer and inverter then reached to the transmission gate T1 so we can say that transmission gate also take some time to get turned OFF so during the time when it is turning OFF from ON during that time new data should not come and disturbed the current data which is being captured.

Basically new data should not enter into the devices during that time also so the hold time will be the time is to take the transmission gate to turn off completely after the clock edge has arrived. if there is any combination delay in the data path then the hold requirement will change.

Sometimes we saw setup and hold time is negative in library file what is the significance of that? Can setup and hold time be negative?

https://www.physicaldesign4u.com/2020/04/sta-iii-global-setup-and-hold-time-can.html

STA - Global setup and hold time. Can setup and hold time of FF be negative??

Global setup and hold time:
Sometime we saw setup and hold time is negative in library file what is the significance of that? Can setup and hold time be negative?

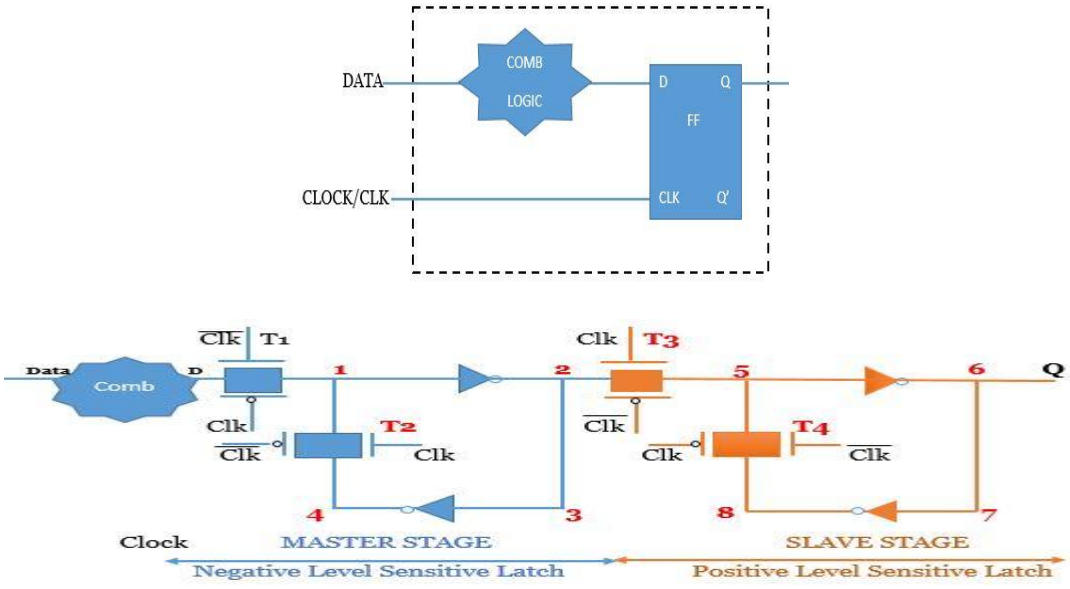
for understanding the working of D FF using transmission gate please read this post.
https://www.physicaldesign4u.com/2020/04/sta-ii-transmission-gated-latch-dffsetup.html

Data path delay (Tcomb) = The Time taken for the data to reach at the input of transmission gate or D pin of FF.

Clock pathdelay ($Tclk_{int}$) = The Time taken for the clock

to reach from clock generation point (clock) to the clk pin (clk) of transmission gate or Clk pin of FF.
Both setup and hold time are measured with respect to the active edge of the clock. For a pure flip flop (containing no extra gate delays) setup and hold time will always be a positive number and fixed once the chip is fabricated it can't be changed. But if we put some glue logic around the FF data path and clock path then setup and hold requirement will be changed and we called this **global setup time** and these components are available as a part of the standard cell library. Setup and hold time can be negative also depending on where we measure the setup and hold. If we want to measure setup and hold time at the component level then it may be negative.

First scenario: If some combinational logic is present between Data pin and transmission gate T1 (data path) or D input of flip flop.



For setup:

Some delay will be induced due to combo logic now the data will take more time to reach from D pin of FF to node 4. So we can say that setup time will be increased because of combo logic. Set up time is the time, when data reach to node 4 and now combinational logic is present so the setup Time requirement will be increased.

Let us take delay of comb logic (Tcomb) is 1ns and the original setup time (Tsu) is 3ns.

Then **new setup time (Tsu_{new}) = Tsu + Tcomb**

= 3ns +1 ns =4ns

Now data will take 4ns to reach at node 4.

NOTE: IF COMB DELAY IS PRESENT IN THE DATA PATH THEN THE SETUP REQUIREMENT IS INCREASED.

For hold:

There is some comb logic sitting between the Data input & transmission gate pin D so now what will happen during this time, the transmission gate T1 is turning OFF till clock is not high and the new data comes at the input of the FF it will have to travel through the comb logic delay before it goes inside and disturbed our original data which is being captured so can we say that if the delay is added in the data path our hold time will be decreased and new data have to wait until the new active edge of the clock will not arrive at the transmission gate T1.

Let us take original hold time of FF (Thold) is 2ns it means the clock takes 2ns to reach from the generation point (clock) to the clock pin (clk) of FF or transmission gate to turn off so during that time new data should not come into the device and data should be stable for 2ns at the input after the rising edge of the clock has arrived.

Then New hold time ($Thold_{new}$) = Thold - Tcomb

= 2ns - 1 ns =1ns (positive hold time)

If Tcomb = 2ns

$Thold_{new}$ = Thold – Tcomb

= 2ns – 2ns = 0 ns (zero hold time)

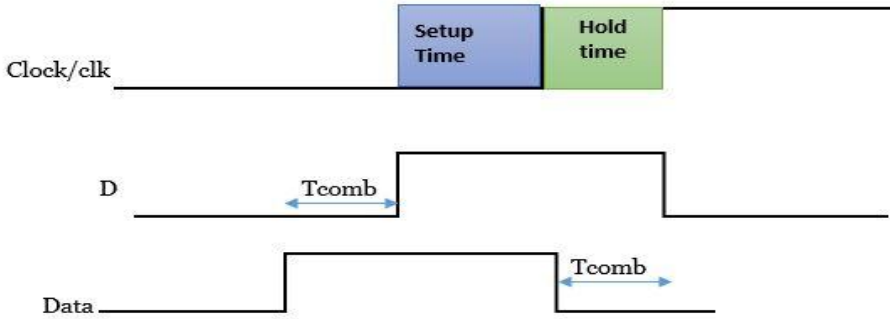
If the comb logic is equal to internal clock delay then our hold time will be zero if hold time is zero it means no need to hold the data after the clock edge has arrived.

If Tcomb = 3ns

$Thold_{new}$ = Thold – Tcomb

= 2ns – 3ns = -1 ns (negative hold time)

NOTE: IF COMB DELAY IS PRESENT IN THE DATA PATH THEN THE HOLD REQUIREMENT IS DECREASED.

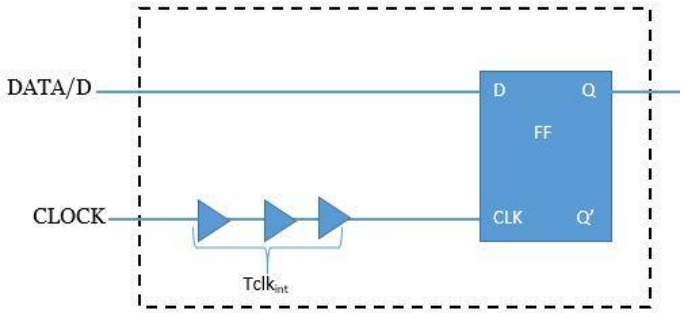


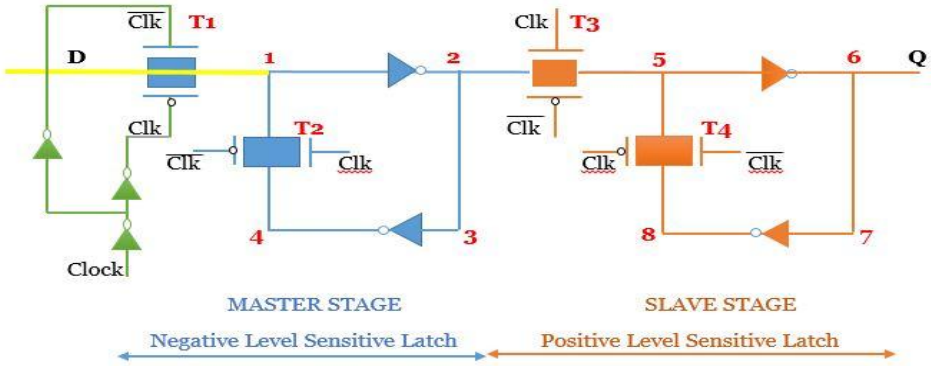
Summary:
If comb logic present between data pin to the D pin of flop then,

Tsu_{new} = Tsu + Tcomb
$Thold_{new}$ = Thold – Tcomb

If Thold > Tcomb => Positive hold time
Thold < Tcomb => Negative hold time
Thold = Tcomb => Zero hold time

Second scenario: if clock delay is present in clock path (the clock is passed through some buffer and inverter to reach at the T1)





For setup:
It means the active edge of the clock will take more time to reach at the transmission gate from the clock generation point (clock) this is called internal clock delay. So what will happen if this delay is present?
So we know that as soon as the active edge of clock has arrived, the transmission gate T1 is turned off and it stops the data entering into the device. We also know that clock is not reaching to the T1 instantly because of the internal clock delay even though the active edge of clock arrived at the generation point and the transmission gate T1 is still ON and the data still have some time to enter into the device and reached at node 4, then we can say that if internal clock delay is present in the design, the setup requirement will be decreased because the data will get some extra time to enter into the device and stabilize at node 4 even though the clock has arrived at the reference point of the flip flop.

Let us take internal clock delay ($Tclk_{int}$) is 2ns and Tsu is 3ns.
Then **new setup time (Tsu_{new}) = Tsu - $Tclk_{int}$**
= 3ns -2ns
= 1ns (positive setup time)

if internal clock delay ($Tclk_{int}$) is 3ns,
Then **new setup time (Tsu_{new}) = Tsu - $Tclk_{int}$**
= 3ns - 3ns
= 0ns (zero setup time)

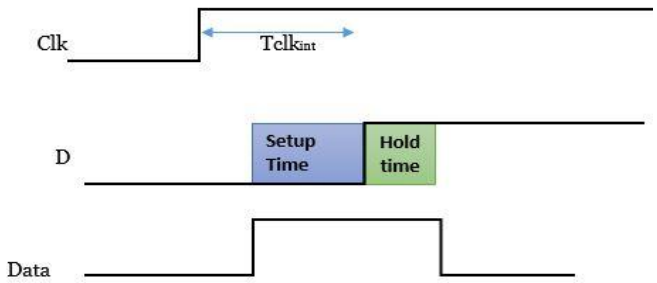
if internal clock delay ($Tclk_{int}$) is 4ns
Then **new setup time (Tsu_{new}) = Tsu - $Tclk_{int}$**
= 3ns -4ns
= -1ns (negative setup time)

It means the data can reach 1ns later than the active edge of clock because internally clock will take more time to reach the transmission gate T1 and this data even though it arrives 1 ns late it will still be able to make it to node 4 and get stabilized.

For hold:
So we know that as soon as the active edge of clock has arrived, the transmission gate T1 is turned off and it stops the data entering into the device and data have to wait at the D pin of transmission gate T1 till the clock is going to low.
Then we can say that if internal clock delay is present in the design the hold requirement will be increased because the data will have to wait at the D pin of T1 to enter into the device
let take **if internal clock delay ($Tclk_{int}$) is 2ns and hold time (Thold) of FF is 2ns**

Then

New hold time ($Thold_{new}$) = Thold + $Tclk_{int}$
= 2ns +2 ns =4ns

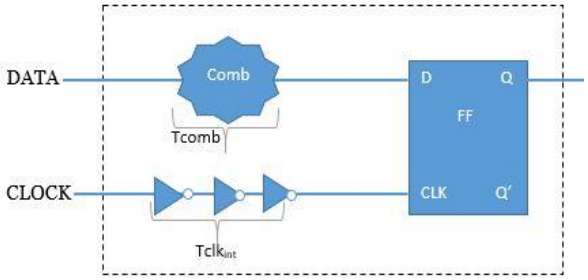


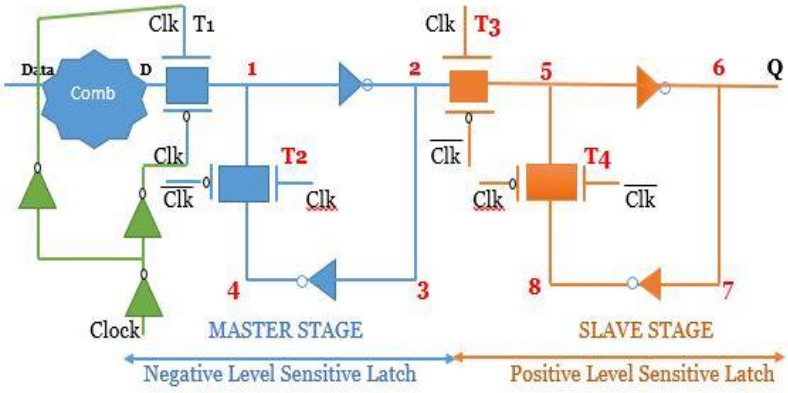
Summary:

$Thold_{new}$ = Thold + $Tclk_{int}$
Tsu_{new} = Tsu - $Tclk_{int}$

If Tsu > $Tclk_{int}$ => Positive setup time
Tsu < $Tclk_{int}$ => Negative setup time
Tsu = $Tclk_{int}$ => Zero setup time

Third scenario: if some comb logic is sitting b/w Data pin and the D pin of FF and clock internal delay is present b/w clock generation point to the clk pin of FF.





New setup time (Tsu_{new}) = Tsu +Tcomb - $Tclk_{int}$
New hold time ($Thold_{new}$) = Thold –Tcomb + $Tclk_{int}$

more precise value:
when the chip is fabricating all the components on the chip have different propagation delay because of the PVT (process - voltage - temperature) conditions so Every cell has three types of delay max, min, and typical delay.

for setup we consider worst (max) data path and min clock path so,
New setup time **(Tsu_{new}) = Tsu +Tcomb(max) - $Tclk_{int}$(min)**

For hold we consider worst (max) clock path and max clock path so,
New hold time **($Thold_{new}$) = Thold – Tcomb(min) + $Tclk_{int}$(max)**

Setup and hold values can not be negative simultaneously but individually they may be negative. so for the setup and hold checks to be consistent, the sum of setup and hold values should be positive.

from where got the setup and hold values: library file
so the next post is related to how the setup and hold are defined for rise and fall constraints in the library file (.lib).