We strive to be the most trusted partner for the world's hardest engineering problems.

DOCUMENTATION OF PLACEMENT AND CTS

Date: 6-Feb-25

Prepared by: Asif Ikbal

We strive to be the most trusted partner for the world's hardest engineering problems.

Table of Contents

We strive to be the most trusted partner for the world's hardest engineering problems.

We strive to be the most trusted partner for the world's hardest engineering problems.

PLACEMENT

Indroduction:

Placement is the step where we place all the standard cells in the core area of the design. The standard cells are picked for the gate level verilog and the timing and layout information it will get from lib file and stdcell lef file. The tool tries to place standard cell in such a way that the design should have minimal congestions and the best timing.During placement trail routing also can be done by the tool.Placement can be done by the tool in 3 different techniques, which are

1. Timing driven placement- Placement happens with timing as priority
2. Congestion driven placement- Placement happens with routing as a priority
3. Power driven Placement- Placement happens with power as a priority

Goals of Placement:

1. Timing, area and power optimization
2. Minimize congestion and congestion happens
3. Minimize Cell density, pin density
4. No timing DRV's(Max transition, Max cap, Max fanout, Max length)

Inputs of Placement:

- Netlist(.v)
- Physical libraries(all the lef files : technology lef, std cell lef, macro lef(if any)) – (.techlef or .tf)
- MMMC file- timing lib file (multi mode multi corner)
  - Timing libraries(.lib)
  - RC corners
  - SDC file
- Timing lib(macro lib, std lib)-)-(.lib(cadence) of .db(synapse)
- SDC file (timing constraint: create clock, set_input_delay,set_false_path)- (.sdc)
- UPF file - (.upf | .cpf) – (if any)

Output of placement:

- Placement DEF
- Congestion and timing reports
- Design with all std cell

Checks before Placement:

- Netlist should be clean
- Floorplan DEF should be good
  - Proper pin placement
  - Macros and pre-placed cells should be fix
  - Power routes should be free of DRCs
- Checks for Don't use, Don't touch cells and routes should be applied properly

Commands:

- check_design -checks pre_placement_stage (synopsys command)

We strive to be the most trusted partner for the world's hardest engineering problems.

- check_design -type place -out_file pre_place_checks.tcl (cadence command)
- sdc checks: check_design(ICC) , checkDesign -netlist (cadence LUI)
- Library checks: Cadence LUI(checkDesign –physicalLibrary, CheckDesign -timingLibrary, checkDesign -all),   ICC (Check_library)

Placment steps:

- Pre-placement
- Global placement
- Legalization
- High fanout net synthesis
- Scan chain reordering
- Post placement

Pre-placement:

Before doing the placement of std cells present in the netlist, we first need to insert some pre-place cells like end-cap cells, well-tap cells , spare cells. Some of this cells are place in floorplan like endcap are placed at floorplan surrending the macros.if everything is okay, then we do for the main placement. we have to provide all the correct placement and optimization settings that we want to be applied while the tool does the placement and optimization. These settings could be like partial placement blockage or density screen setting, bound or region creation, cell/instance padding, path_groups and effort, enabling the early clock flow (ECF) in case of innovus, enabling the extreme flow, enabling the useful skew, global congestion effort, global timing effort, power effort, Multibit flop conversion and many more.

Global placement:

During this stage, tool will places all the standard cells into the core area.But will not follow any DRCs such as std cells overlapping, miss alignment with std cells rows. In this stage tool also can do trial routing. Trial routijng is tool divides the entire design into metal layers into number of GCELLS.Each GCELLS carries set of H and V routing layers.

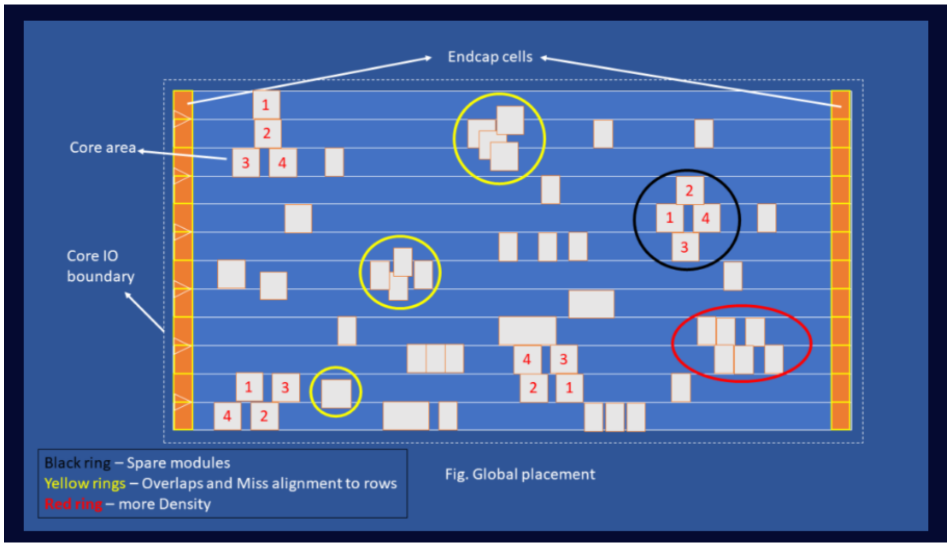


Fig.Global placement

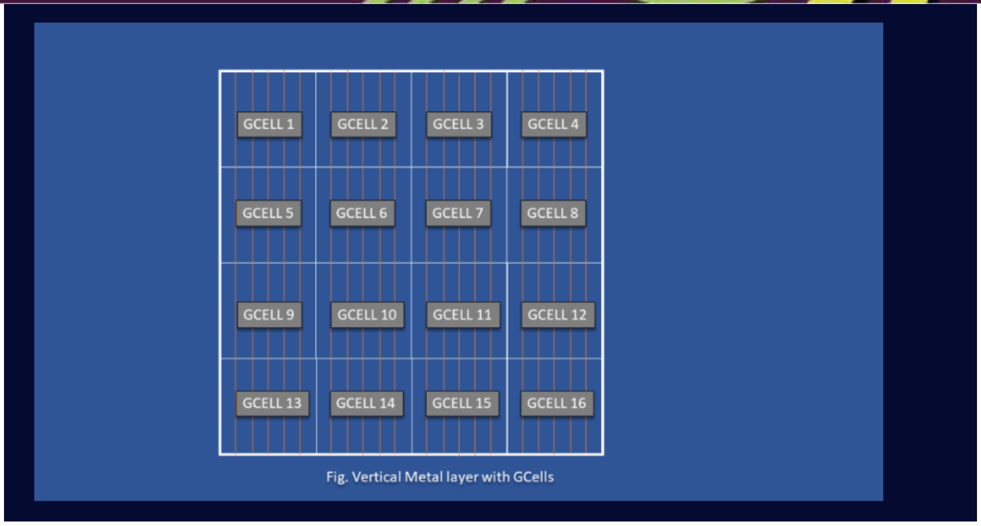


Fig.trail routing

Ref: [PLACEMENT - VLSI TALKS]

Legalization:

During legalization tool tries to spead the standard cells. If the cells density is more in a particular area in order to get rid of power issues, tool will distribute the cells . In this step the tool aligns all the overlapped std cells into the std cells rows by following DRC rutes. And make ure there are no std cells placed out of core area.

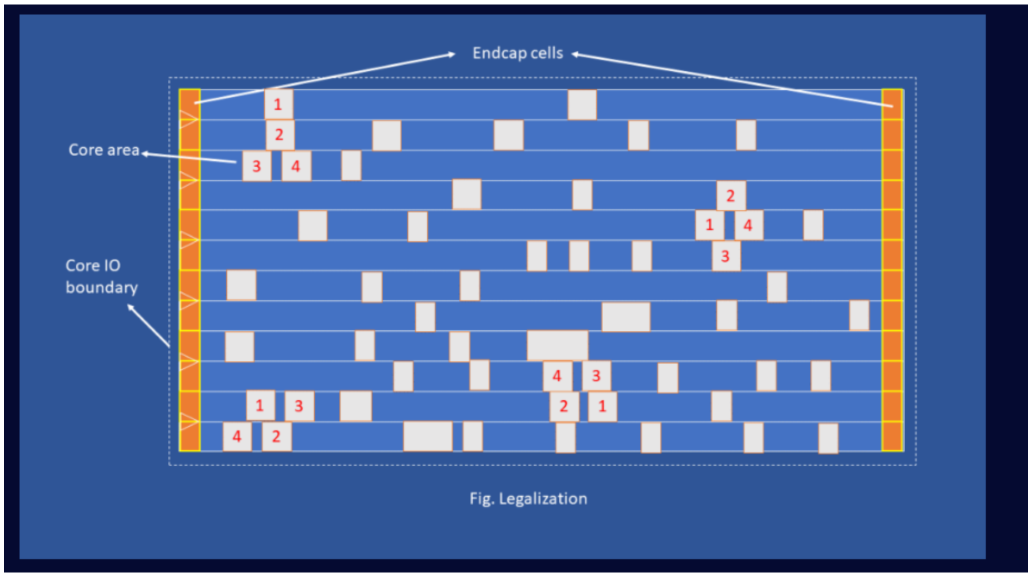
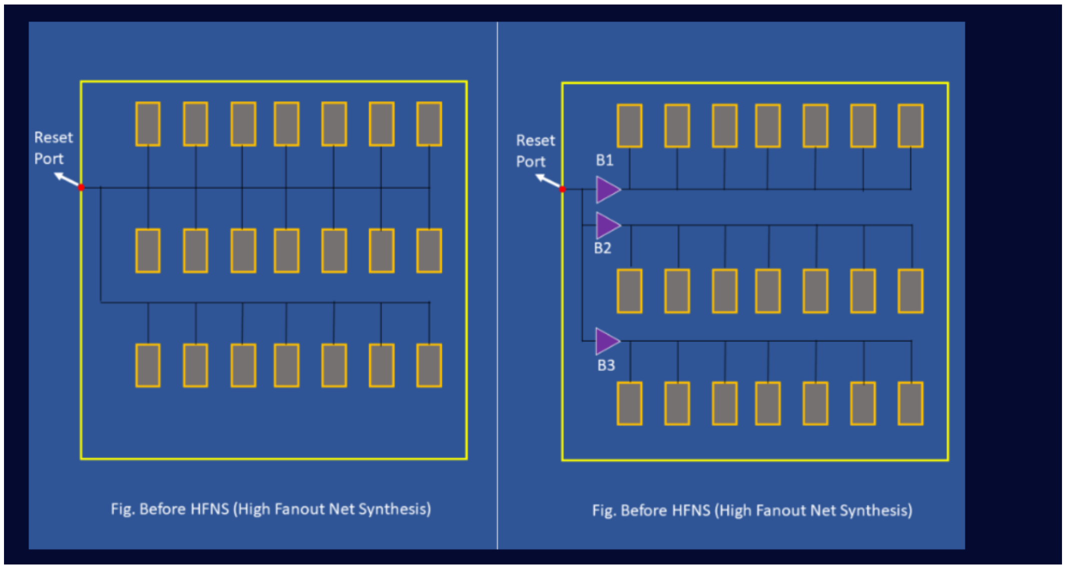


Fig. Legalization

High Fanout Net synthesis:

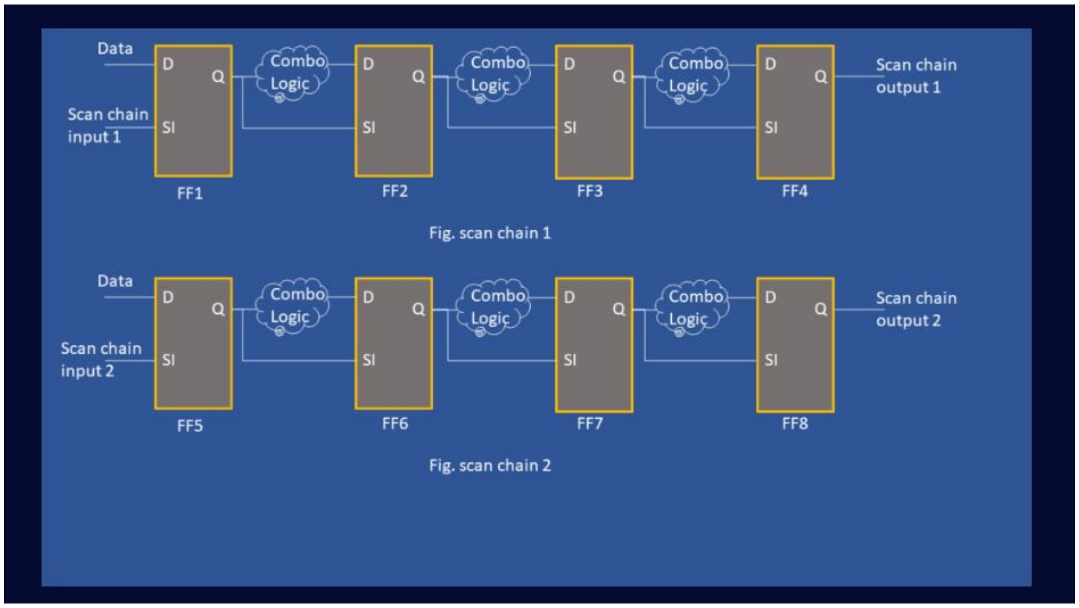
After legalization, there are some nets will have high number of fanouts like reset, scan enable etc. But, there is a restriction for maximum fanout in timing constraint. Even though it is a soft constraint high fanout will increase load capacitance. Based on the fanout constraint tool identifies the high fanout nets in the design. Then tool optimizes these nets by picking the right drivers or adding buffers to drive the load, adding repeaters to reduce on delay and signal integrity and balancing the nets capacitance and resistance to strength the signal integrity. By doing optimization tool meet the signal quality and timing requirements of the design while minimizing delay, power consumption and noise on high fanout nets. Following figure shows on example for HFNS.

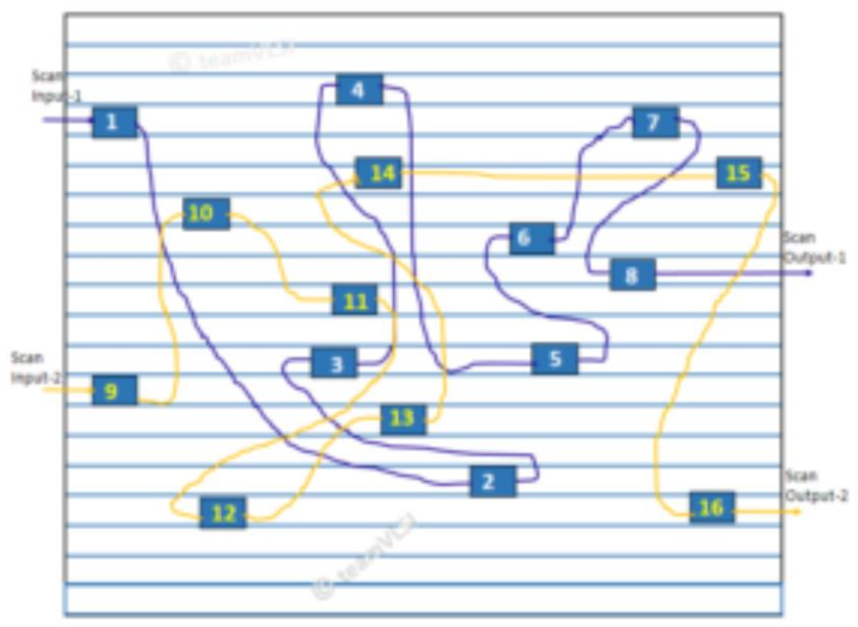


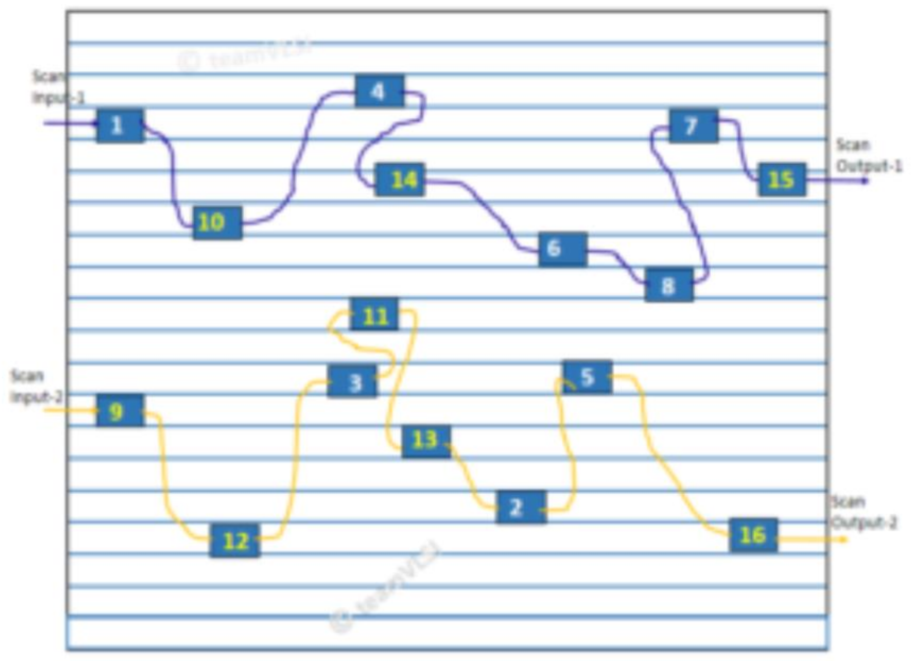
Reset Port
Fig. Before HFNS (High Fanout Net Synthesis)
Reset Port
B1
B2
B3
Fig. Before HFNS (High Fanout Net Synthesis)

Scan Chain reordering

During the dft stage, all the flipflops are connects in the form of shift registers pattern like the circuit shown in below image. DFT team create multiple scan chains. They divide all avaiable flipflops in the design into equal number of sets and stitches scan chain, each set represents one scan chain.For example, consider a design have 12 flipflops then dft team creates 3 scan chains, each chain will contain 4 flopflops and all are connected to each other in shift register pattern.







Cadence	ICC	Description
Reorder_scan	Optimize_dft	Reorders the scan cells after routing placement

Post Placement:

- During post placement tool tries to meet all design timing requirements like DRVs then WNS and TNS only for setup checks
- Finally tie cells also can be placed by the tool for required input pins

Ref: [PLACEMENT - VLSI TALKS]

We strive to be the most trusted partner for the world's hardest engineering problems.

Ref: [Placement Steps in Physical Design - Team VLSI]

Physical cells at pre-placement:

- Well tap cells
- Endcap cells
- IO buffers
- Antenna diodes
- Spare cells

Function	Cadence command	ICC command
Endcap cells	Add_endcaps	Create_boundary_cells
Welltap cells	Add_well_taps	Create_tap_cells
Antenna checks	Create_diodes	Create_diodes
Spare cells	Add_spare_insts	Add_spare_cells

Checks after placement:

- Check legalization
- Check PG connections for all the cells
- Check congestion, Density screen and pin density
- DRVs (transition, fanout, capacitance, max length)
- Timing(setup only)
- Check whether don't touch cells and nets are preserved or not
- Total utilization of the design after placement

Placement commands

Cadence	ICC	Commands
Place_design	Create_placement	Places std cells based on the global setting for placement,RC extraction, Timing analysis and early global routing
Place_detail	Legalize_placement	Corrects flawed cell locations and reports corrections or instance overlap problems
Place_fix_congestion	Refine_placement	Performs incremental placement with congestion optimization
Place_opt_design	Place_opt	Place and optimize the current design
Place_opt_design -icremental	Refine_opt	Refine the placement and the netlist of the current design

Ref: [Differences between refinePlace, ecoPlace, place_opt_design -incremental and place_opt_design -incremental_timing]

Placement Constraints:

There are main two type of placement constraints

1.Blockages

We strive to be the most trusted partner for the world's hardest engineering problems.

2.Placement bounds

We will discuss details about this. User needs to know this because ,this can be used to solve some qor issues as well

1.Blockages:

There are three types of blockages:

- Hard blockages: Prevents std cells to be placed in the area
- soft blockages: during optimization, only buffer and inverter can be placed in that area
- partial blockages: limits the cells denstiy in any particular area. 40% means only 40% area is blocked for the palcement

2.Placement bounds:

Controls the placement of group of leaf and hierarcical cells. Groups the cells connected to each other to minimize wire length. Helps to ensure appropiate location for the cell.

From cadence support, we can see the command details of create_boundary_constraint.

There are 4 types of bounds.

- Soft Guide: A soft guide constraint is similar to a guide constraint except that there are no fixed locations. This provides a stronger grouping for the instances under the same soft guide. The soft guide constraint is not as restrictive as a fence or a region constraint, so some instances might be placed further away if these have connections to other modules.
- Guide: The guide constraint is the loosest of all floorplan constraints in Innovus. It roughly defines an area within which you instruct the tool to place the cells of a given module.

  The guide constraint is automatically assigned to a module when it is moved or placed inside the core area. Cells from modules that are not part of the guide may be placed inside the guide if the area is underutilized. Likewise, cells from the modules defining the guide may be placed outside if the area is overutilized. To summarize, a guide is a soft indicant for placement of cells that may let some cells in and let some cells out.

- Region: The region constraint too is a loose constraint. However, it is stronger than the guide. When a region is defined, it requires all the cells of the modules contained in the defined region to be placed in the specified region area. If there is an extra space in the region, other cells from other modules may be brought inside. To summarize, a region is a stronger indicant for the placement of cells that may let some cells in and does not let anything out.
- Fence: The fence constraint is the strongest floorplan constraint in Innovus. The fence constraint does not allow cells from other modules to be placed inside even if the area is underutilized. In addition, the cells in a fenced area cannot be placed outside the defined fence.

  A fence is meant to be exclusive, although not completely. In some cases, especially when the fence is too small or there are buffers in the design. To summarize, a fence is the strongest indicant for the placement of cells that neither lets cells in nor lets anything out.

Description	Cadence	ICC
How to create bounds	create_boundary_constraint -group group_name -type {fence|region|guide|cluster} -rects [{x1 y1 x2 y2} …]	Create bound –name b1 –type soft|hard|exclusive –boundary {10 10 20 20} instance_2

We strive to be the most trusted partner for the world's hardest engineering problems.

Note: First need to create one instance group, this instance group will containt the cells which we want to bound, then we will include that group into a bound.

Ref: PLACEMENT - VLSI- Physical Design For Freshers

Ref: How to keep a group of cells close to one another during placement

ref: Recommendations for resolving congestion or local hot spots seen after placement

Ref: Bounds in Placement

Chanllenges in placement:

Congestion fixing after placement:

First we can see some reasons why there can be a congestion:

1.high standard cell denstiy in a small area

2.Placement of standard cells near the macros

3.High pin density at the edge of the macros due to high fan in cells like AOI,OAI

4.bad floorplan

5.Macro/std cells might have used all the metal layers inside and routing resources will be less

6.Placing the macros at the center instead of the boundary

7.During IO optimization, tool does IO buffering , so lots of cells placed in a core area

How to solve congestion:

1.we face congestion we can use three type of blockages,

- Soft blockage – in between macro channel, this blockage allows only buffer, inverter insertion for optimization only, as in between macro the channel is very critical, placing std combo cell will many routing there.
- Partial blockages: if the congestion issue is for, pin density, we can use partial blockage (50,60,70%) , and if the issue is for io buffer near port, we can use small small partial blockages together
- Hard blockages- we use hard blockage in the corner or notch of a macro .

2. Apply cell padding, specifyCellPad cellName 6 (CUI: set_cell_padding -cells <cellName> -padding 6)

3. Opt for congestion-driven placement with high effort

4.To fix IO buffer congestion one big partial blockage will not work, we had to use small partial blockages grouped together. This is spread the cells.

5.We can use module padding, setPlaceMode -modulePadding module factor

6. Run the congRepair (CUI: place_fix_congestion) command wil repair congestion

Ref: Congestion & Timing Optimization Techniques at 7nm Design

Ref: Recommendations for resolving congestion or local hot spots seen after placement

We strive to be the most trusted
partner for the world's hardest
engineering problems.

How to solve IO timing violation:

1.Creating placement bounds

By default tool tries to give more piority on reg2reg paths. So there can be some reg2out violations.So meet the internal timing , tool place those flops a little away from output port. These violations need to be addressed at the placement stage itself.It was found that there was enough margin in previous timing paths (reg2reg). so we can create a flopbound (create_bound) near the output ports to fix the reg2out setup violatons.

To create one placement bound for the inst which we want to place close to each other.

We need to identify:-

1.Location of the bound

2.Size of the bound

3.Which group should be created out of soft guide, guide, region and fence

location of the region bound has to carefull selected in the design. It makes sure that internal timing should not get violated while fixing reg2out timing violations.

Cadance lui:

```
createInstGroup GroupName
addInstToInstGroup GroupName {inst_name_list}
createSoftGuide GroupName
setPlaceMode -place_global_soft_guide_strength {low|medium|high}
place_opt_design
```

Cadence cui:

```
create_group -name group_name -type {fence|region|guide|cluster}

update_group -name group_name -add -objs object_name

create_boundary_constraint -group group_name -type {fence|region|guide|cluster} -rects [{x1 y1 x2
y2} …]
set_db place_global_soft_guide_strength {low|medium|high}
place_opt_design
```

2.Using magnet placement:

This command pulls the command standard cells closer to the hard macros or IOs with the legal location in a pre-place stage. With this command, you can

- Specify the target macro/IOs for attracting instances
- Specify the attracted objects with logic or sequential level, certain cell type
- Specify whether to honor soft blockage
- Mark the pulled cells as placed

The place_connected command can be used to pull the test logic close to the memories using simple scripting.

The recommended flow is to run this command before place_opt_design, as follows:

3.Using net weight

Use the following command to specify a net weight on the net connecting these.
The place_opt_design command gives a higher priority to the nets with higher weights when minimizing the net lengths.

Legacy Mode: specifyNetWeight <netName> <netWeightValue>

Common UI Mode: set_db <netName> .weight <netWeightValue>

netName: The name of the net between the instances to be placed closer.

netWeightValue: The value is between 1 to 10. The default value is 2.

Note: If the net weights are used to place the clock gating components close together, ensure that the following option is set to false:

Legacy Mode: setPlaceMode -place_global_clock_gate_aware false

Common UI Mode: set_db place_global_clock_gate_aware false

Some reference from cadence:

Cadence support: How to keep a group of cells close to one another during placemen t

We strive to be the most trusted partner for the world's hardest engineering problems.

Clock tree synthesis

CTS is the process of connection the design clock port to flop flop clk pin routing but in this path tool insert buffer or inverter to maintain less skew.

Some basic terms what should be discussed is

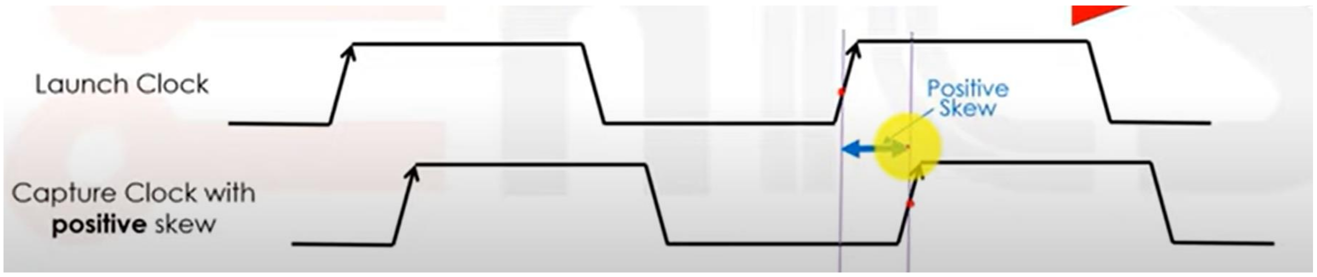
- Skew: the difference between clock arrival time at two different registers
- Jitter: difference in clock period between different cycles
- Slew: transition of clock signal. Trise/Tfall.
- Insertion delay: delay from clock source until registers
- Uncertainty: Clock uncertainty is the time difference between the arrivals of clock signals at registers in one clock domain or between domains. Pre CTS uncertainty is clock skew, clock Jitter and margin. After cts, the post cts uncertainty is clock jitter and margin. So, post cts uncertainty is less than the pre-cts uncertainty, because we exclude the skew value in post cts uncertainty

Ref: Clock Uncertainity - VLSI Master

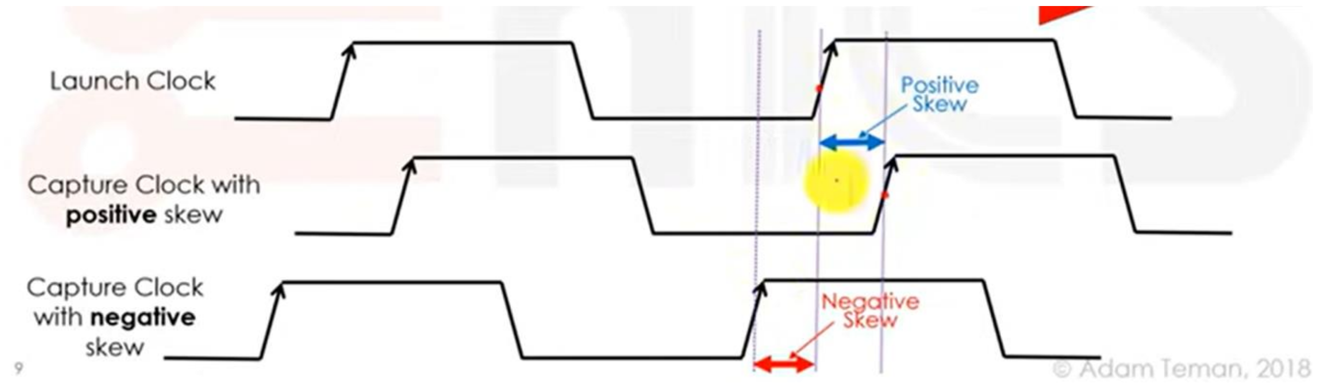
Skew:

There are two types of skew:

1.Positive skew: the capture clock edge comes late than launch clock edge



2.Negative skew: capture clock comes before the launch clock edge comes

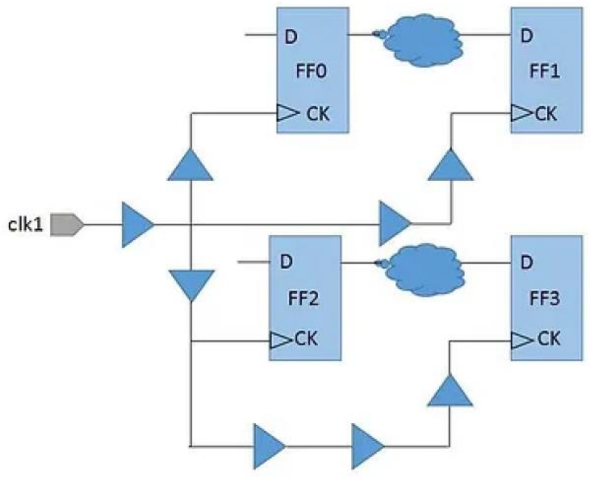

Launch Clock
Capture Clock with positive skew
Capture Clock with negative skew
Positive Skew
Negative Skew
© Adam Teman, 2018

Note: Positive skew is useful in reducing setup timing violations while negative skew ishelpful for reducing hold timing violations.

3.Global skew:

Global skew is the difference in arriabl time of clock signal between the shortest and longest clock path in the same skew group. Timing path may or may not exist between these registers.in below pic, skew between FFO and FF3 is global skew

4.Local clock skew: the difference in arrival time of clock signal between any two registers which shares a data path in a skew group.in below pic, skew between FF0 and FF1 is called the local skew



<u>Note:</u> Target of CTS is to build clock network in such a manner that clock skew should be as minimum as possible so that all the registers in the block cantrigger at the same time. To achieve zero skew is not possible practically due to different delay offered by nets. This means even if net length and metallayers used is same for two different paths, they exhibit different delays owning to on-chip variations (OCV). Also it is not advisable to have zero skew.Reason being, zero skew means all the flops will turn on at the same time thus consuming power from the PG rail simultaneously. This increases theload on PG rails to supply power to all the flops at the same time. This can lead to major IR drop and can affect the chip functionality. Thus idealpractice is to maintain skew as minimum as possible.

In a nut shell we can say that zero skew is not possible and advisable as well. Keeping skew constraint very high can lead to insertion of more numberof clock buffers in clock path. This can again lead to more power drop and higher area requirement. Also too much tighter skew constraint can lead to more timing violations. Thus, it is highly advisable to keep skew at the optimum value so that desirable PPA parameters can be easily achieved.

The reason behind the more timing violations for tighter skew , if that if the skew is less, then capture clock edge is coming a little eariler, because it will come close to launch edge. And is the capture edge is coming little eariler, then it will be bad for setup violation and good for hold violations

At before cts stage, set_ccopt_property -skew_group <> target_skew <> to set target skew

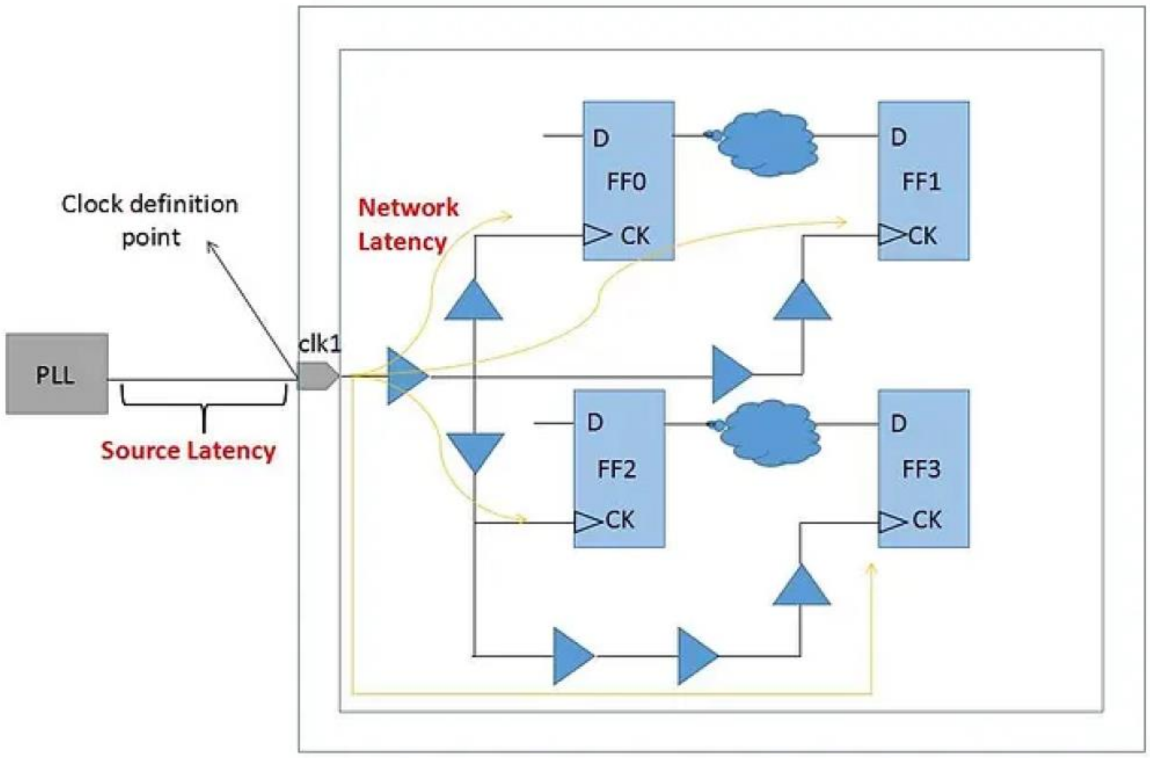
Some factors that can affect clock skew are as below.

- Unbalanced clock structure
- On-Chip Variation (OCV) parameters
- Registers belonging to same skew group are placed far
- Unequal delay balancing at sink pins

Clock latency:

Clock latency is defined as the time taken by the clock signal to reach to the sink pin from its source. There are two types of clock latency i.e. Sourceand Network Latency.

- Clock source latency: is the time taken by clock signal to reach the clock defination point in the block from the clock source
- Clock network latency: is the time taken by clock signal to reach the sink pin from the clock defination point in the block.



There is a tradeoff between latency and skew. So, to control skew, more buffers are required in clock paths which means more insertion delay.Latency target provided to CTS is same for the longest and shortest path.

In short, more insertion delay means more number of clock buffer or inverters will be required which in return increases area, routing resources and power consumption

Command: set_clock_latency -max 5 {clkB clkD}

OCV mode

	Launch Path (Setup)	Capture Path (Setup)	Launch Path (Hold)	Capture Path (Hold)
Network Latency	-max	-min	-min	-max
Source Latency	-max -late	-min -early	-min -early	-max -late

Some parameters affecting the insertion delay are as below.

- Scattered registers
- Long routes increasing RC delay of net

We strive to be the most trusted partner for the world's hardest engineering problems.

- High clock fanout
- More number of buffers/inverters added to meet skew target
- Improper clock structure

Ref: [Clock Tree Synthesis - Part 2 : Clock Skew, Latency, and Uncertainty]

Ref: adi teman youtube: [DVD - Lecture 8: Clock Tree Synthesis - YouTube]

Before cts, the clock nets are ideal.

But to match the insertion delay with respect to cts, we add one insertion delay at flooplan,powerplan, placement stage. By using the sdc command "set_clock_latency". This command is used to specify ideal clock latency

After cts, tool will ignore this value of command set_clock_latency,

And will use real propagation delay and calculate the actual clock latency value.

The main goal of cts is to maintain less skew and minimum insertion delay/clock latency. lock latency refers to the delay that exists between the clock source and the flip-flop clock pin. It represents the time taken by the clock signal to propagate from its ideal waveform origin point to the clock definition point in the design.

Clock latency consists of two main components: source latency and network latency. Source latency measures the time it takes for the clock signal to propagate from the clock source to the clock definition point. On the other hand, network latency measures the time it takes for the clock signal to propagate from the clock definition point to the sinks, which are the registers that receive the clock signal.

Slew: the Clock tree needs to have one target slew value , which tool will try to achieve
Command: set_ccopt_property target_max_trans <value> -skew_group <skew_group_name>

Timing derates:
We have PVT condition applied, with using mmmc file with different lib files (ss,ff,tt).
In real world, PVT condition verify(die to die varaiation),
So we use timing derate to account those varaiation.
Timing derating factors model the effects of varying operating conditions by adjusting the delay values calculated for the individual timing arcs of a block
The set derating factors, use the *"set_timing_derate"* command.
To reset the derating factors, use the *"reset_timing_derate".*
To report the derating factores, use the *"report_timing_derates"*

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.

We strive to be the most trusted
partner for the world's hardest
engineering problems.

Clock tree Structures:

Clock mesh: Clock mesh structure can be divided into three sections namely - Global tree or Mesh drivers, Clock meshnetwork, and Local tree. An example of clock mesh is shown in figure 1. In clock mesh structure, clock path iscommon till global tree thus making this structure more robust against on-chip variations.

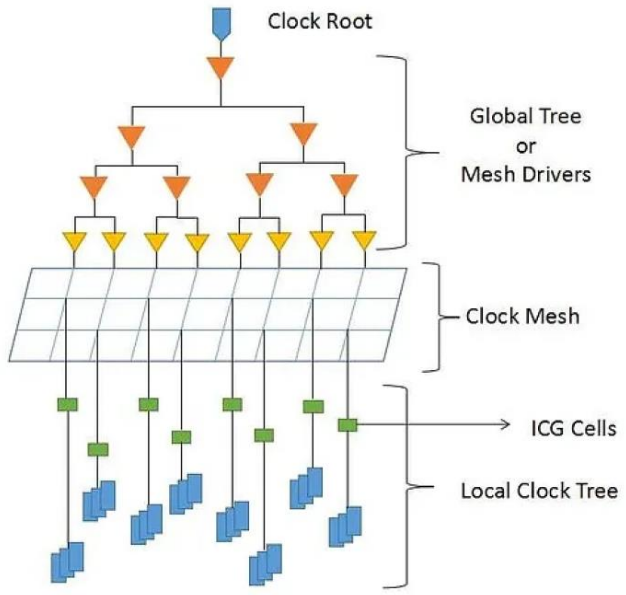


Figure 1 : Clock Mesh Structure

Note: the buffer driver the mesh, and the mesh is routing grid, and the flops are directly connected to the mesh routing grid

Advantage of Clock mesh:

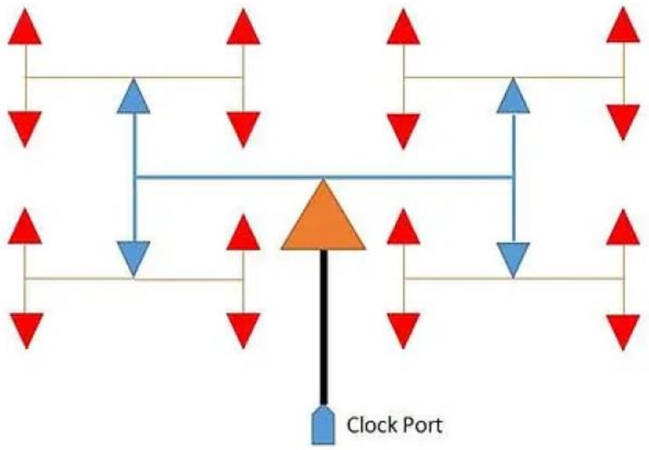
1.Most accurate skew and delay balancing
2.all sink pins has almost same route lengths
3.More rebust to OCV as common clock paths is longer (from mesh driver to clock mesh routing is same.so long common clock paths)
4.Skew is determined by grid density and not overly sensitive to load position ,so we don't need to add buffer on everyzone , the clock signal is avaiable already on everyzone, but h-tree is pretty denpendent on position of flops(because the h-structure need to go to every zone of design, which may be difficult)
5.So this is tolerant to process variations

Disadvantage of Clock mesh:

1.it requires relatively more routing resources, increases congestion in the design
2.Power consumption is more in clock mesh as lots of mesh buffers(in mesh drivers) are required to drive the mesh
3.Huge of wiring and power
Wirecap large
Need really strong drivers –so this pre-drivers has cap large
4.So,to solve this,we solve this, we can keep the distance between routing high,but this iscrease more skew, which is lossing the main advantage. So we need to let the skew as large as possible that the design can take

H-Tree clock structure:

This is the most popular clock structure used in the industry. The connection of clock cell forms a shape of H.



As it can be seen from above figure, there is a central huge buffer which drives the tree further. Central buffer isconnected to four other buffers and tree goes on like this till sink pin. In H-tree structure, sub-tree isapproximately equal to the square root of the length of its parent tree. Also the width of metal layer is more nearthe central buffer and it reduces as we move towards the sink pin.

It is impractical to achieve an ideal or pure H-tree structure because registers are not evenly placed in reality. Alsothere are routing blockages and route guides used in the design which makes it impossible to create a pure H-tree. By pure H-tree it means all the branches of a tree must be of equal length and should have the same numberof buffers in each sub-branch.

Advantage of HTREE:

1.A well balanced tree structure
2.A pure H-tree with evenly balanced load can achieve zero skew
3.Require relatively less routing resources compared to clock mesh structure
4.Improved power consumption as compared to clock mesh structure

Disadvantage of HTREE:

1.Less resistant of H-tree variations
2.if tree size grows longer then insertion delay increased and skew is likely to degrade
3.Huge central driver requires more area and power

The drawbacks of H-tree can be overcome up to a great extent by integrating fish-bone structure along with H-tree structure. In a fish-bone structure, a single buffer can drive multiple registers in place of a single buffer drivinga single register as in case of H-tree

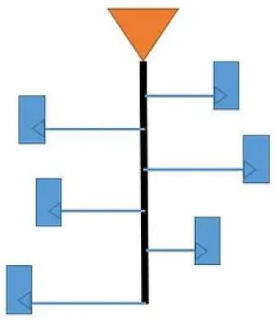


Figure 3: Fish-bone Clock Structure

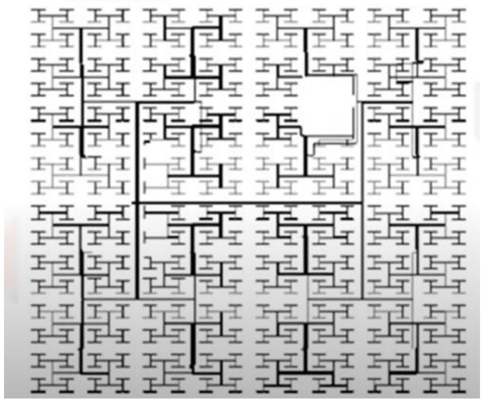
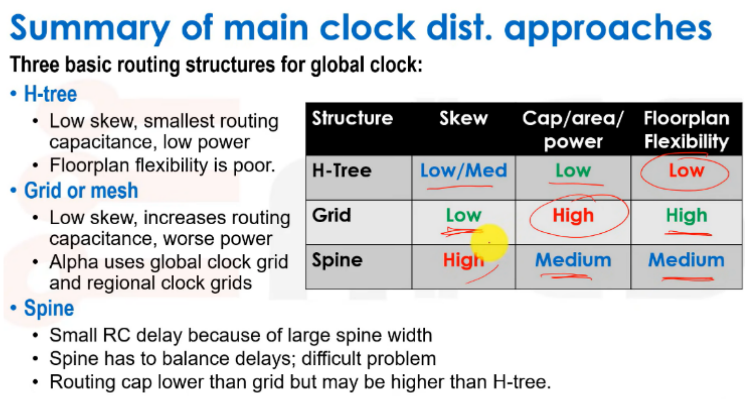


Fig: H-Tree

This is still hard to implement.As in the entire design, every zone may be have same number of flops, so this buffer(above one) may not drive same number of register everytime, which may create difference. And design may have many blocked area because of macro and blockages



Summary of main clock dist. approaches

Three basic routing structures for global clock:

- H-tree
  - Low skew, smallest routing capacitance, low power
  - Floorplan flexibility is poor.
- Grid or mesh
  - Low skew, increases routing capacitance, worse power
  - Alpha uses global clock grid and regional clock grids
- Spine
  - Small RC delay because of large spine width
  - Spine has to balance delays; difficult problem
  - Routing cap lower than grid but may be higher than H-tree.

Structure	Skew	Cap/area/ power	Floorplan Flexibility
H-Tree	Low/Med	Low	Low
Grid	Low	High	High
Spine	High	Medium	Medium

Some basic Understanding of CTS:

What a clock is made of:

- A clock port or output pin of an IP or output pin of a gate(clock mux, clock gate) or the output pins of the lock generator, if the clock is a generated clock, it will have any clock devider Q pin as a clock source.
- Lead/end points/sinks: clock pins of registers, memories, clock pin of macro etc (it can be one output port as well, if the clock is driven outside of block)
- Buffer and inverters: required to meet clock transitions and meet skew requirements
- Special logics: multiplexers, integrated clock gates, clock devider, this logics exist before cts as well, this logics comes from rtl

How CTS is defined?

After loading the placement database, the clock tree related informations are taken from databased and sdc. There is a important file called clock spec file, which can be generated by command from database, and we can see that are the information of the clock that we are building

Clock spef file:

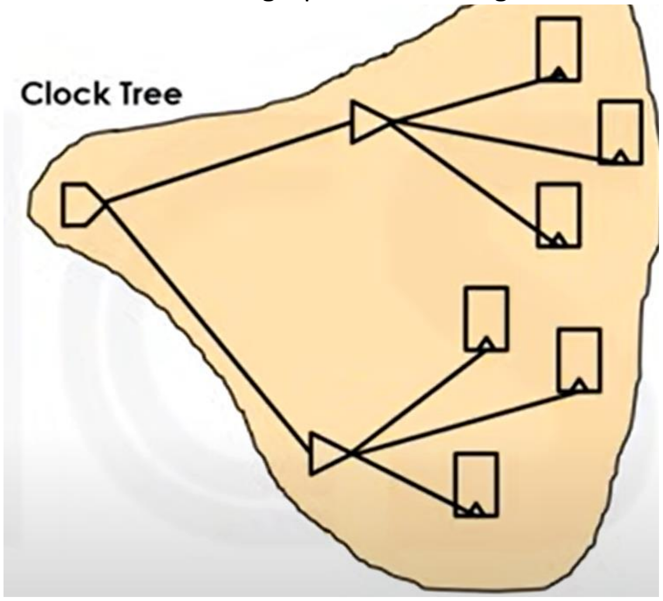
CTS spec file contains the information like NDR rules, Clock buffers and clock inverters, skew and latency target, leaf nets and trunk nets , max /min transitions. One can find the value of clock skew and latency

targets, max tran and max cap, uncertainty set during CTS stage either in CTS file or in clock specification file. Tool will consider values specified in this file as ideal value during optimisation and report generation.
There are two major thing that is inside a clock spec file:

1. Clock tree
2. Clock skew group

1.Clock tree:

the root of a circuit graph for buffering.



Create_clock_tree -name clk -source [get_ports clk] -no_skew_groups
Note: if -no_skew_groups is used, then command will not create any default skew group for now, we will can create skew group manually using create_skew_group commands.

2.Clock skew group:

a group of flip-flop(clock sinks) which have the same skew target. By default all the sinks of a clock tree are in the same skew group. But there will be many skew groups in one single clock tree . there can be overlap between skew groups as well.

By default skew groups are created by tool, and tool can create skew groups of different clocks sink pins. And one sink can even belong to more than one skew group(in that case, lowest skew is taken as a target skew for that one sink).

Create_skew_groups -name clk -sources [get_ports CLK] -auto_sinks
Note: -auto_sinks will automatically take all the sinks that is connected from clock sources

Update_skew_group -skew_group clk -add_sinks [get_pins FF1/CK]

Note:
You can get the detailed information regarding the skew group using the following command:
Legacy UI: report_ccopt_skew_groups -file <filename>
Common UI: report_skew_groups -out_file <filename>
Ref: Understanding various steps in clock tree synthesis using CCOpt-CTS log file

CTS Definitions/Exceptions:

There are different type of pins that cts consider for buffering, which is defined in clock spec file

We strive to be the most trusted
partner for the world's hardest
engineering problems.

- Stop pins:
  - A leaf of a clock tree(clk pin of flops,macro). The clock net will be buffered up to the stop pin but not beyond it
  - All clock sinks are implicit stop pins and therefore will be considered for skew banacing /analysis till stop pins
  - If user want to define any additional pins as stop pins in ccopt, use:
    - Set_db pin:INV1/A .cts_sink_type stop

    So clock will not propagate beyond the inverter INV1/A pin
- Ignore pins:
  - Ignore pins are pins on clock net that will not be considered a sink in any skew group
  - Suppose, if any sink is a primary output ports(which is a sink for this design), this output port will go to another block,taking clock signal, in that case, we don't want to balanced skew this port, we can define ignore pin on this. But we will fix DRV(max_transition,Max_cap) requirements here with buffers, so a few buffers are expected
  - Simply, we can say, ignore pins will not be balanced with respect to any skew target, but it will insert buffer to balance any DRV requirements. And buffering will happened uppto ignore pin, not beyond it
  - Command: set_db pin:INV1/A .cts_sink_type ignore
  - You can check the 'ignored sinks' with the following command:

```
foreach pin [get_ccopt_clock_tree_sinks *] {
    if { [get_ccopt_property sink_type -pin $pin] == "ignore" } {
        puts "global ignore pin found : $pin"
    }
}
```

- Exclude pins:
  - Exclude pins are similar to ignore pins but the clock net will not be buffered up to an exclude pin. DRV is not going to be balaced and skew is also not going to be balanced for this pins.
  - Command: set_db pin:div_ff1/ck .cts_sink_type exclude
- Through pins:
  - Through pins are pins which we want to propagate through them, and insert buffer beyond those pins and add those new pins to skew groups
  - Suppose, the clock gate CLK pins, as clock gates are created from latch, it has CLK pin as well, so tool may consider them as stop pins. But we want to consider the paths behind those clock gates as well, and do balancing.
- Insertion delay pin/Float pin:
  - This is same as sync or stop pin but internal clock latency of that pin is taken into consideration while building the clock tree. Its a clock entry pin of hard macros and it needs to be considered while building the clock tree
  - In some cases, we would like to provide the clock to a certain stop pin earlier or later than the average insertion delay.
  - For example , if a macro block has some internal insertion delay(means, inside macro,from the macro pins there is a internal delay to reach the flop), we can should provide the clock early for skew balancing, as after that pin, there is a internal delay ,

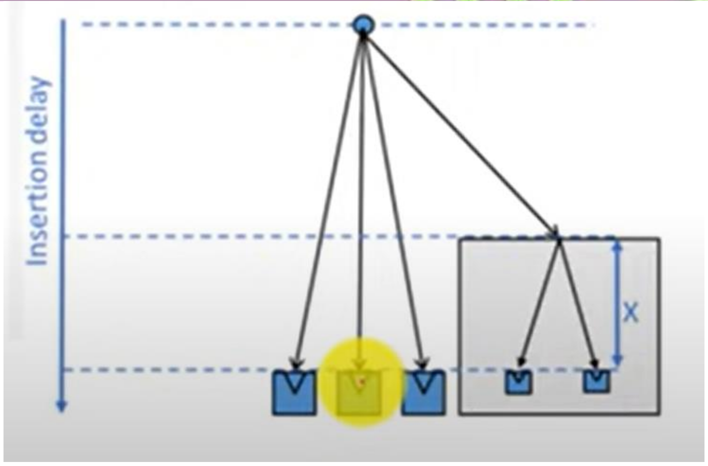


Fig: macro with internal insertion delay (x is internal insertion delay)

- If the avg insertion delay for all the sinks are T. then for this macro pin we should give insertion delay of T-X
- Command: set_db pin:mem/CK .cts_pin_insertion_delay 150ps
- Note: this means the value of x is define 150. That means, this 150ps will subtrated from the avg insertion delay . so the new insertion delay will be (T-150ps)

Summary:

	Part of the clock tree	Considered for DRV fixing	Considered for delay balancing	Clock propagates beyond it
Stop pin	Yes	Yes	Yes	No
Ignore pin	Yes	Yes	No	No
Exclude pin	No	No	No	No
Float pin	Yes	Yes	Yes	No

Clock net Routing:

Clock nets are prone to signal integrity if not properly routed or balanced

1. A glitch on a clock net wil cause an additional clock edge
2. Slow transitions will cause deteriorated setup hold times of registers
3. Fast transitions will strong aggressors to neighbour nets

For that,

- We will pre-route the clocks nets, first choice for routing tracks
- Use higher, thicker metals for clock routing
  - Low resistance, less capacitance
- Apply shielding to clock nets, if we have enough routing resources
- Add de-caps next to clock buffers which will provide extra charge , if there is current shortage
- We can use cell padding on clock buffers, then use those empty space to for adding decap cells
- We should NDR (double width, double spacing)
- In routing type, we can define pre-ferred layers and shielding
- In Innouvs, we differentiate between three types of clock nets:
  - Top- the initial branch of the clock tree, Very wide and High layer
  - Trunks- the main branches of the clock tree, Wide and high layer
  - Leaf- the bottom levels of the clock tree. Closer to the logic

How to apply NDR:

First, Define Non-default routing rules (NDR):

- Double width and double spacing NDR:

We strive to be the most trusted
partner for the world's hardest
engineering problems.

        - Create_route_rule -name **CTS_2W2S** – spacing_multiplier 2 -width_multiplier 2
    - Then create a routing type for cts with metal layers mentioned
        - Create_routing_type – name cts_trunk – non_default_rule **CTS_2W2S** -top_preferred_layer M7 -bottom_preferred_layer M6 -shield_net VSS -bottom_shielding_layer M6
    - Now assign nets to this rule
        - Set_db cts_route_type_trunk cts_trunk
        - Cts_route_type_trunk is a build in command in innovus, which takes all the trunk nets of clock network

Clock tree steps:

The Clock tree synthesis will be build mainly in 4 steps:

- Clustering
- Balancing
- Routing of clock tree
- Post conditioning

Clustering:

During Clustering, Tool will build only a DRV-aware clock tree and will not balance the clocks. At the start of this step, tool will print the maximum driver distance and the unit delay for the delay for the clock buffer/inverter from a user-provided list or from the library.

Balancing:

During this step,Tool will balanced the design per the skew group constraint. Look for the pattern balancing in the log file. This pattern will repeat multiple times in the log file

Routing of clock tree:

During this step, tool will route all the clock tree nets using a nanoroute engine

Post conditioning:

This step is run to clean up any minor degradation after the clock routing

Ref: Understanding various steps in clock tree synthesis using CCOpt-CTS log file
Ref: Understanding CCOpt prerequisite checks and common error/warning messages seen during this stage
Ref: How to debug clock latency / insertion delay QOR issues post CCOpt CTS

CTS issues and fixes:

After cts to check for any errors and warning when the clock tree was build. We can check that with " check_design -type cts".

Setup Violation :

1. reduce the amount of delay in data path, this ca be done by reducing the necessary buffers
2.VT swapping, means you swap HVT cell with SVT or LVT cells
3. upsizing the cells can also prevent the setup violation
4. Use of two inverters in place of buffer
5. readjusting the position of cell
6.Using useful skew, one thing to remember, positive skew helps setup timing
7. Inst group or bound to reduce the in2reg , reg2out violations

Hold timing violation fixing:

1. introducing the delay in data path, you can do this by adding the buffers in data path,if the magnite of the hold value is small, using small driving buffer cell can solve small hold violations
2. Downsizing the cells in data path.before doing that need to check the setup margin

We strive to be the most trusted partner for the world's hardest engineering problems.

3. VT swapping, you can swap the the LVT buffers to SVT or HVT buffers, to increase the data path delay, it would not distrub the layout as well.
4.Using useful skew, we can solve
5. Lock up latches, if we some how try to separate the launching edge and capturing edge by a phase then it will relax the timing path and improves the chances of hold timing being met.
6.If the magnitude of the hold violation is large. We need to go for the delay cells into the data paths solve large hold violations, again we need to check for setup margin
7.Using slow lapture register to increase the Tcq delay of launch path
8.Use swap the capture flop to decrease the hold time of the capture register to solve small number of hold violations
9.One thing to remember, negative skew helps in hold timing violations, so we can push the launch clock or pull the capture clock.lets see some picture below:

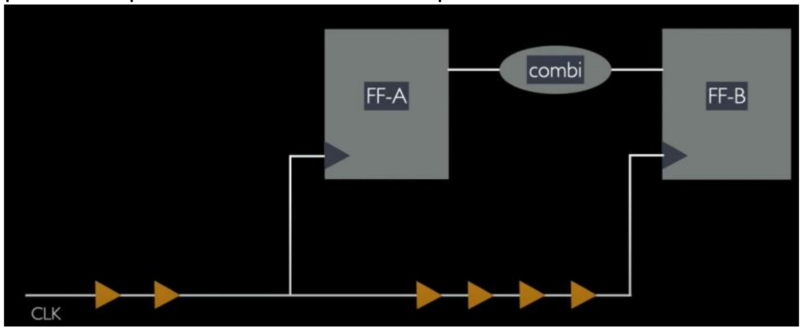


Fig: ideal condition

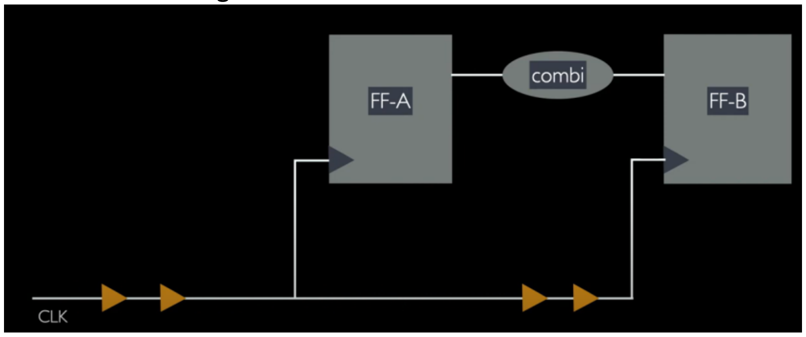


Fig: decreasing latency of capture flop(negative skew)/ Pulling the capture flop edge

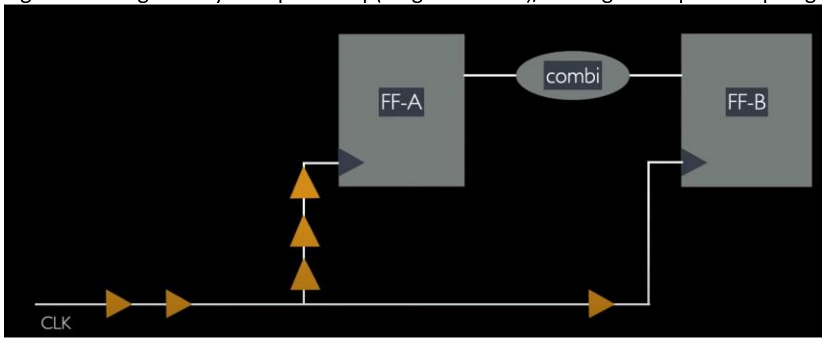


Fig: increaset the latency of launch clock flop/pushing the launch flop edge
Ref: How do I avoid setup and hold time violation? | by Agnathavasi | Medium

Max transition violation:

Normally, the each cell has max_transition limit in the library.if the load of cell goes high, and if the value goes output of lookup table, the extrapolation value give high transition value.
In sdc file, using set_max_transition value, user over constraint the limit, we can create issues.
Normally, the max_transition violation issues in on load cells input pins, we can filter unique driver cells which is driving those violationing loads.
If the design has max transition issues, we can use below procedure:

1.Increase the driver size, It will enhance the output transition, upsizing means increasing W/L, and current , I propotional to W\L.
2.Decrease the net length by moving cells nearer
3.adding buffers
4.Increase the width of the route
5.Break the nets if there is long nets
6.Break the large fanout by duplicating drivers or with buffering
7.Split Fanout
8.Swap Hvt to Lvt

Min pulse width violations:

1.min pulse width violation mainly occurs because of unequal rise time and fall time of cells, we can solve this issue by using cells that are equal rise time and fall time
2. remove some buffer from the buffer chain, because long buffer can give worst transition, that worst transition can effect the rise time of the cells.
3. Convert HVT to LVT.

How to define pulse width?

1.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

2.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.

How to report min pulse width?

Command: report_timing -check_type pulse_width
Command: report_min_pulse_width

Ref: CTS (PART -III) CLOCK BUFFER AND MINIMUM PULSE WIDTH VIOLATION - VLSI- Physical Design For Freshers

Signal integrity issues:

Before going into solving the cross-talk,first we need to do some sanity checks:

1.check run logs for errors/Warning

2.Check if coupling cap is extracted in SPEF

3.Review parasitics annotation summary, check is the annotation is 100% or not

4.review crosstalk settings (Noise thershold for hight and bump, check is the analysis tool report noise on source or the load

We strive to be the most trusted
partner for the world's hardest
engineering problems.

How to solve Crosstalk analysis
1.Using NDR rules: Double spacing,more spacing, less capacitance,less cross talk
2.Multiple vias:less resistance, less RC delay
3.Shielding: constant cross coupling capacitance , known value of crosstalk
4.Buffer insertion: boost the victim strength, but one extra buffer will consum more power and area
5.downsize: Downsize the aggessor, or upsize the victim cell
6.Moving aggresor nets in different location, reassigning tracks for aggressor , if in the new location, the aggressor may not effect the another new victim nets, if they do not overlap in the switching window
7.Breaking the long victim nets, and switch the part of victim net to another layer

How to fix IR:
1.Spead cells , this is reduce IR
2.Downsize cells
3.split Output load capacitance, because the amount of current drawn from the PG is directly proportional to output load
4.Add PG stripes
5.Increase stripes width
6.Use multicut vias
7.Cell padding, to reduce peak current demand
8.Insert Decap cells, decaps acts as charge bank, when it is needed, it provides charge, but decaps are more prone to leakge current

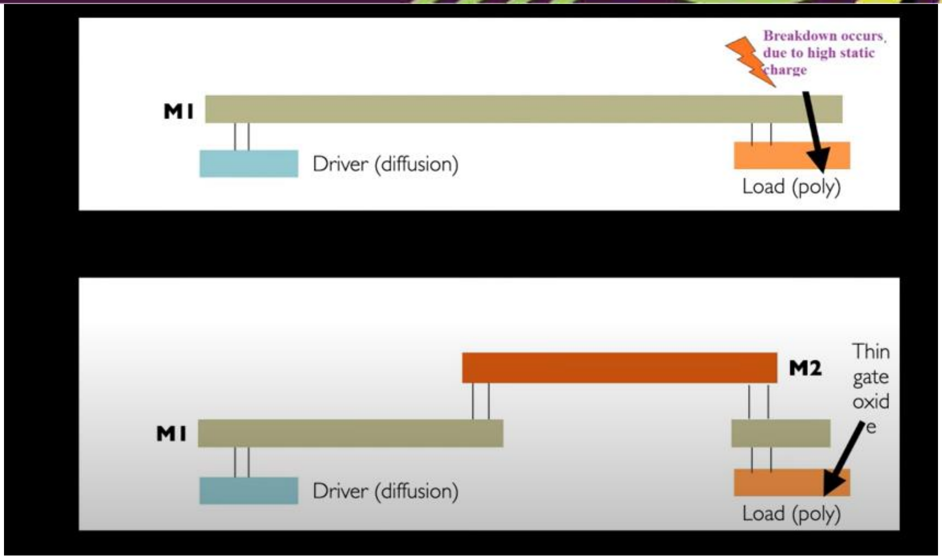
Electro-migration issues:
Wires are getting thiner, so the resistance is getting higher. But the current density requirement is high for lower node.

1.downsize the driver cell will reduce the current density at the output of the driving pin and nets, but make sure that the downsize is not effecting the timing and max_transition
2.we can apply NDR on victim nets to increase the nets width, which will reduce the resistance of the net. Make sure that there are no DRC or congestion due to NDR. We can apply NDR on a net segment or a part of net
3.insert buffer on long nets, inserting buffer ,will reduce the resistance of long nets, make sure that the em effected net has enough setup margin
4.Route on Higher layer , high layer has less resistance because of high width, make sure that there are no congestion on higher layer
5.Replace single cut vias with multi cut via, multi cut via will increase the area, but the resistance will be less, so it will help to reduce EM
6.High fanout break by split or buffer: high fanout drains high current form cell. If we reduce the fanout that would reduce the output current of the driver cell.Generally we add same size buffer or one level up buffer, so that timing should not effected.
Note: Hold fixes includes, buffer insertion , which breaks long nets, if the longs nets are break, then the resistance will less. So hold fixes can fix EM as well.

Antenna effect issue:
Foundary provide antenna rules, which must be met.antenna rules mainly is antenna ratio,which is the ratio between metal to gate area

1.Change the order of the routing layers. If the gate(s) immediately connects to the highest metal layer, no antenna violation will normally occur. This is also called layer hopping

We strive to be the most trusted partner for the world's hardest engineering problems.



Note: this layer hopping should be done close to load pin.
Mainly in manufacuring process, first the low layer will be fabricated. Then the upper one. So the lower one length will short. And it will accumale less charge

2. Add diode(s) to the net. A diode can be formed away from a MOSFET source/drain, for example, with an n+ implant in a p-substrate or with a p+ implant in an n-well. If the diode is connected to metal near the gate(s), it can protect the gate oxide. This can be done only on nets with violations, or on every gate (in general by putting such diodes in every library cell). The "every cell" solution can fix almost all antenna problems with no need for action by any other tools. However, the extra capacitance of the diode makes the circuit slower and more power hungry.

Q.Why Hold timing violation is fixed after CTS?
Ans: Before CTS, the clock is ideal. It means that the values of the skew and slew are not accurate. They are just the approximate values. But after CTS, the clock is propagated. It means that the skew is balanced. Also, the transition times of the clock(s) along with the input and output rise and fall times can be known. The balancing of the clock skew across the network is achieved by inserting buffers (not ordinary buffers, but special clock buffers) in the clock path.
Also it is known that the hold violation is more dangerous than setup violation. This is because setup violation can be fixed by decreasing the frequency of operation of the circuit. But this is not the case with hold. Even if the hold fix is done before the CTS stage, the additional buffers in the clock path add up (or in some cases reduce) certain amount of delay. Hence, it is always recommended to fix the hold violations after the CTS stage.

Both of setup and hold fixes requires the exact path delay as well as the exact clock insertion delay, so in theory, both should be fixed after CTS. The only reason we fix the setup time before CTS is that it's a bit too late to do so after CTS. After CTS, all the cells are placed and you don't have much flexibility in fixing the setup time. On the other hand, fixing hold is just adding the buffers and is easily done even after placement.
If the tool becomes sophisticated enough to handle the setup time fix even after placement, we don't need to fix the setup before CTS. Basically, it's mainly due to the tool limitation.

Both the setup and hold violation analysis is a pessimistic analysis i.e. we want to find the worst failing paths. For this reason when we are analyzing setup time we want to make are clock as fast as possible and data as slow as possible (we use WC corner for setup analysis). From the point of view of the clock network, the clock delays are minimum before CTS when the clocks are considered ideal and propagation delay of clock network is zero. Thus setup can be analysed for the worst failing path before CTS.

We strive to be the most trusted
partner for the world's hardest
engineering problems.

When we are analyzing hold time we want to make clock as slow as we can and data as fast as we can (BC corner for hold analysis). Again from the point of view of the clock network, the clock delays are maximum after CTS when the clocks are propagated and buffer cells are added in the clock network to balance skew. This hold can be analysed for the worst failing path after CTS

But I think, we should check it, Because the hold timing violation can came if the hold timing of capture flip-flop is giving high abnormal value.
Another thing, in half cycle path , the hold timing violation can occur, because, there can be a sdc issue that the false path is not accurately define for those paths.

Q.skew constraint is very tight, what will happeneed?
Ans: with tight skew constraint, it will add many buffer in the clock paths, which will increase the insertion delay, and high number of clock buffer will increase power and area. \

Q.Clock tree should be created with buffer or inverter?

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.

Ref: CTS (PART -III) CLOCK BUFFER AND MINIMUM PULSE WIDTH VIOLATION - VLSI- Physical Design For Freshers

Q.Some basic of CTS:
1.Dynamic power:
P=a x F x C x VDD,
around 30% of power is used by clock nets,
2.signal integrity:

- if the coupling capacitance is high, then it can increase or decrease delay,

What the main problem is , is the rise or fall increase much, then the value can be get less than the nldm table and create interpolation, and also vise-varse

- if the slew or transition is slow in clock, that will create more prone to noise ,also poor flipflop functional(worst Tcq,Tsetup,Thold)

3.if the fast clock,means overconstraint, so big driver, so more power and area. And it is a potential bigger aggressor to other nets
4. so best practise is to keep transition time to 10% of the clock period
5.Target skew is 5% of clock period
6.Insertion delay is 30% of clock period
7.One focus point in cts, is main goal is to meet timing and DRV constraints
8.Minimizing skew was just to correlate post-cts and pre-cts timing.
9.if there are lots of inverter or buffer in a clock path, the min pulse width violation can occur., we should remove the buffer which is giving bad transition. Min pulse width violation mainly occurs due to unequal rise and fall time of comb cell.

Ref: Enhancing VLSI Design Efficiency: Tackling Congestion and Shorts with Practical Approaches and PnR Tool (ICC2)

Youtube video(adi_teman): DVD - Lecture 8: Clock Tree Synthesis - YouTube

Ref: CTS (PART -III) CLOCK BUFFER AND MINIMUM PULSE WIDTH VIOLATION - VLSI- Physical Design For Freshers