2025

Documentation of FloorPlanning

DATE: 6TH FEBRURUARY, 2025

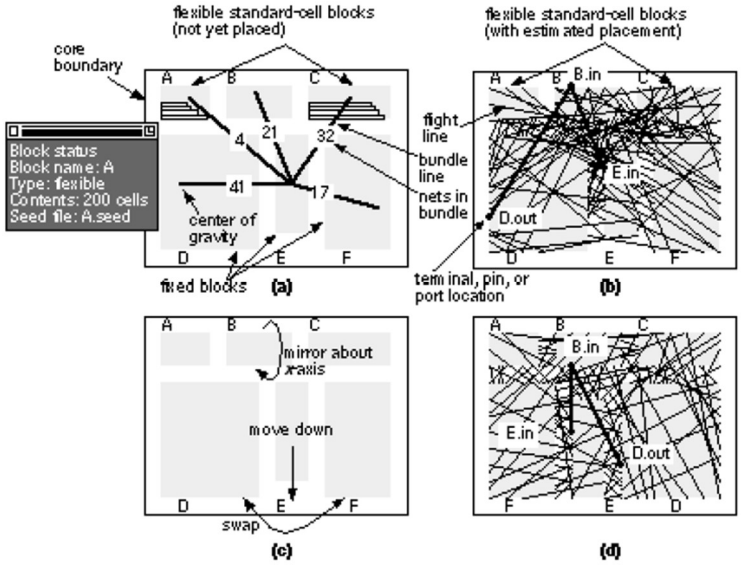
PREPARED BY AMIT SAHA

Table of Contents

Floorplanning

Introduction

Floorplan is a mapping between the logical description (Hierarchical Netlist) and the physical description (the floorplan).



Initial random floorplan generated by a floorplanning tool.

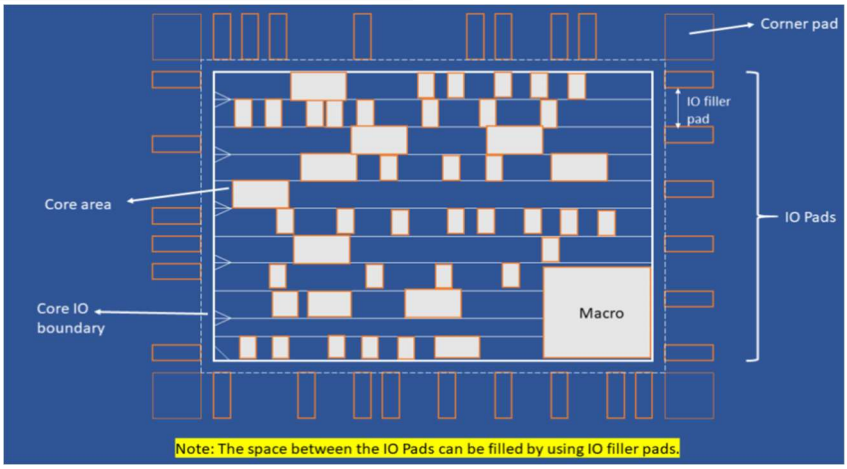
Blocks are moved to reduce congestion.

Types of Design

There are two types of designs:

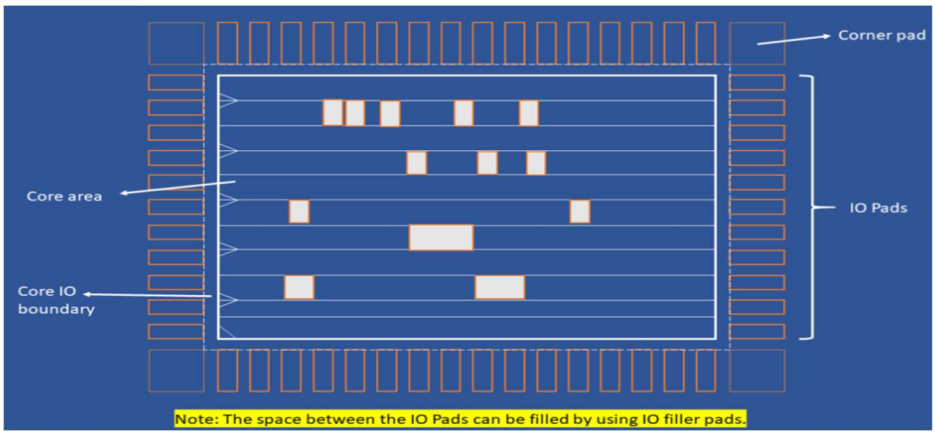
Core limited design

In this type, the core area is densely packed with standard cells, macros, and other components. The size of the chip is primarily determined by the core area. This design aims to maximize the utilization of the core area while ensuring there is enough space for routing.



Pad limited design

In this type, the size of the chip is determined by the number of I/O pads rather than the core area. The I/O pads are densely packed around the periphery of the chip, and the core area is adjusted to fit within the constraints set by the pads.



The goals of floorplan

- Arrange the blocks on a chip,
- Decide the location of the I/O pads,
- Decide the location and number of the power pads,
- Decide the type of power distribution.
- Decide the location and type of clock distribution.

Objective of floorplan

- Minimize the chip area.
- Minimize delay.
- Minimize routing congestion.

Input files for floorplanning

Netlist (.v)

A **netlist** is a crucial component in electronic design automation (EDA) for VLSI. It is essentially a description of the electrical connections between various components in a circuit. Here are the key points:

- **Components**: Lists all the components (like transistors, resistors, capacitors) used in the circuit.
- **Connections**: Specifies how these components are interconnected, detailing the nodes and nets.
- **Hierarchy**: Can include hierarchical information, showing how sub-circuits are connected to form the overall design.

Techfile or tlef

It contains detailed information about the technology used in the design, including:

- **Layer Information**: Details about different layers like metal layers, polysilicon layers, and via layers, including their names, types, and properties such as width, spacing, and resistance.
- **Design Rules**: Specifications for placement and routing, including minimum widths, spacing rules, and other constraints.
- **Manufacturing Grids**: Information about the grid used for aligning the geometry during manufacturing.
- **Via Definitions**: Details about the vias used to connect different layers.
- **Other Process Information**: Additional details related to the manufacturing process and design rules.

The Technology LEF file helps ensure that the design adheres to the specific manufacturing process and design rules, facilitating efficient and accurate chip fabrication.

Timing library files (.lib)

A **LIB file**, short for Liberty Timing File, is a critical component in VLSI design. It provides detailed information about the timing, power, and area characteristics of the standard cells used in a design. Here are the key aspects:

- **Timing Information**: Includes cell delay, transition times, setup and hold times, which are essential for timing analysis.
- **Power Information**: Contains data on power consumption, including leakage power and dynamic power.
- **Cell Characteristics**: Details about the cell's area, pin names, pin directions, and capacitance values.
- **Operating Conditions**: Specifies the conditions under which the cell operates, such as process, voltage, and temperature variations.

LIB files are used throughout the design and verification process to ensure that the design meets performance and power requirements.

Synopsys Design Constraint (.sdc)

An **SDC (Synopsys Design Constraints)** file is a crucial part of VLSI design. It is used to specify the design constraints for timing, power, and area, ensuring that the design meets its performance requirements. Here are some key aspects:

- **Clock Definitions**: Specifies the clock sources, periods, duty cycles, and edge times.
- **Timing Constraints**: Includes setup and hold times, input and output delays, and clock uncertainties.
- **Power Constraints**: Defines power-related constraints to manage power consumption.
- **Design Rules**: Sets rules for maximum fanout, transition times, and capacitance.
- **Operating Conditions**: Specifies the conditions under which the design operates, such as voltage and temperature.

SDC files are written in a format based on TCL (Tool Command Language) and are supported by most EDA tools for synthesis, place and route (PnR), and timing analysis.

RC coefficient file (Tlu+ or qrctech file)

An **RC coefficient file** in VLSI design contains information about the resistance (R) and capacitance (C) values associated with the interconnects and components in an integrated circuit. Here are the key points:

- **Resistance (R)**: Represents the opposition to the flow of electric current through the interconnects.
- **Capacitance (C)**: Represents the ability of the interconnects to store charge.

These files are crucial for accurate RC extraction and estimation, which are essential for timing analysis and signal integrity verification.

MMMC

MMMC (Multi-Mode Multi-Corner) analysis is a critical aspect of VLSI design. In this file need to put the lib and sdc file. It ensures that the chip functions correctly under various operating conditions and modes. Here are the key points:

- **Multi-Mode**: Refers to analyzing the design in different functional modes, such as normal operation, test mode, and low-power mode. Each mode may have different timing constraints and requirements.
- **Multi-Corner**: Involves evaluating the design across different process, voltage, and temperature (PVT) corners. This helps ensure the chip performs reliably under all possible manufacturing variations and environmental conditions.

MMMC analysis helps in optimizing the design for all these scenarios simultaneously, ensuring robust performance and reliability.

MMMC Flow

Innovus Flow

First, create a library set using the lib file. Next, establish the RC corner. Then, generate the delay corner by combining the library set and RC corner. After that, create a constraint mode using the SDC file. Following this, create an analysis view by utilizing the delay corner and constraint mode. Finally, specify which analysis view will perform the setup, hold, and power checks.

Example:

```
create_library_set -name ssgnp0p765v125c \
        -timing [list \
$LIB/tcbn03e_bwp143mh169l3p48cpd_base_svt_110b/tcbn03e_bwp143mh169l3p48cpd_base_svtssgnp_0p765v_125c_cworst_CCworst_T_ccs.lib.gz]
create_rc_corner -name rcworst_CCworst_T_0\
   -preRoute_res 1\
   -postRoute_res 1\
   -preRoute_cap 1\
   -postRoute_cap 1\
   -postRoute_xcap 1\
   -preRoute_clkres 1\
   -preRoute_clkcap 1\
   -qx_tech_file $QRC_MAX\
   -T 0
```

```
create_delay_corner -name setup_ssgnp_0p765v_125c_cworst_CCworst -
library_set ssgnp0p765v125c -rc_corner rcworst_CCworst_T_0

create_constraint_mode -name mission -sdc_files [list ${SDC_FILE}]

create_analysis_view -name func_setup_ssgnp_0p765v_125c_cworst_CCworst -
constraint_mode mission -delay_corner setup_ssgnp_0p765v_125c_cworst_CCworst

set_analysis_view \
        -setup [list func_setup_ssgnp_0p765v_125c_cworst_CCworst]
```

ICC 2 Flow

In ICC 2 Flow, there's no need to define any timing library information in MMMC. First, you create a mode, followed by creating a corner. Next, you read the parasitic tech file and create a scenario using the mode and corner. Then, you set the parasitic parameters, process label, voltage, and temperature, and define the operating condition. After that, you populate the mode and corner constraint file for the scenario. Finally, you set the status of the scenario to specify its purpose, such as setup, hold, or power analysis.

Example:

```
remove_modes -all; remove_corners -all; remove_scenarios -all
create_mode pmm_functional
current_mode pmm_functional
create_corner rcworst_CCworst_T
current_corner rcworst_CCworst_T

read_parasitic_tech -name maxtlu -tlup
/data/SynX060123/PDK/N7/tn07clbl045b2_1_2p1a/RC_TLUplus_cln7_1p11m_2x1ya3y2y
y2r_mim_ut-
alrdl_DPT_5corners_1.2p1a/rcworst/Tech/rcworst_CCworst_T/cln7_1p11m_2x1ya3y2
yy2r_mim_ut-alrdl_rcworst_CCworst_T.tluplus

create_scenario -mode pmm_functional -corner rcworst_CCworst_T -name
func_setup_timing_pmm_functional_rcworst_CCworst_T

set_parasitic_parameters -late_spec maxtlu -early_spec maxtlu
set_process_label ssgnp
set_voltage 0.765000
set_temperature 0
set_operating_conditions -analysis_type on_chip_variation
current_mode
current_corner
current_scenario
report_pvt
read_sdc $MODIFIED_SDCS/inputs/design_sdc/ananke_core.pmm_functional.sdc
```

```
source -v -e
$MODIFIED_SDCS/inputs/design_sdc/ananke_core_sdc_constraints_none_any.sdc.tc
l

set_scenario_status func_setup_timing_pmm_functional_rcworst_CCworst_T -
setup true -hold false -dynamic_power false -leakage_power true -
max_capacitance true -max_transition true -min_capacitance true -active true
```

Output of Floorplan

- Get core and boundary area
- IO ports / pins placed
- Macro placement done
- Floorplan def file

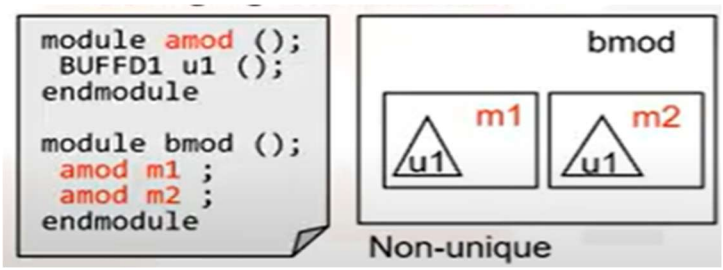
Sanity Checks

To ensure the quality of imported input files, they must be clean, accurate, and consistent with each other. Sanity checks are performed to verify these attributes. If the input files pass these checks, we can proceed to the next stage. Otherwise, any issues identified must be resolved before moving forward.

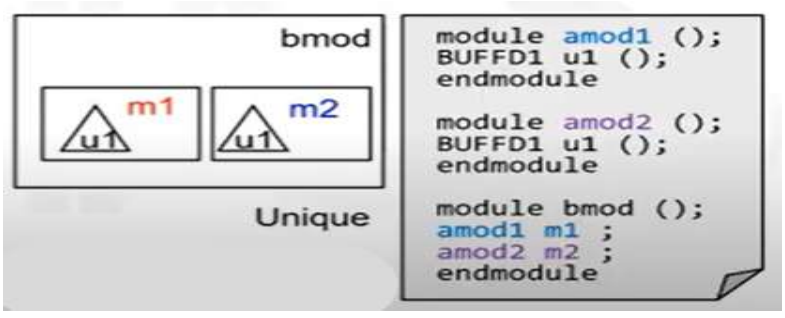
Sanity checks to perform on input files are listed below:

Netlist Check

When moving to the physical domain, the netlist must be unique. A unique netlist means that each sub-module is only referenced once.
Non-unique example:



Unique example:



If netlist is not unique then it will also do optimize m2 when it is optimizing the m1.

Design Check

Sanity checks must be performed on the netlist to ensure it is free of design issues. Some common design issues to look for include:

- Combinational loops
- Floating inputs
- Multi-driven inputs
- Unintended latches
- Tri-stateable gates
- Empty modules in the netlist
- Assign statements

Innovus tool command for design check is “checkDesign -netlist”

Library Checks

Sanity checks must be performed to verify the correlation between the netlist and the libraries. Specifically, the cells present in the netlist should also be found in the timing and standard cell libraries. This ensures that all components used in the design are properly defined and characterized, facilitating accurate timing analysis and synthesis.

Innovus tool command for library checks is “checkDesign -physicalLibrary” and “checkDesign -timingLibrary”.

Timing Checks

This sanity check ensures the quality of the SDC file by verifying that every path in the design is constrained. Unconstrained paths are identified as timing paths that lack constraints. Additionally, this check ensures consistency among the constraints, confirming that all timing requirements are properly defined and aligned throughout the design.

Innovus tool command for timing check is “check_timing”

Floorplan Creation Command

Innovus Command	ICC 2 Command
floorplan	initialize_floorplan

Example of Floorplan Creation

Innovus Example

```
set default_area 0
foreach cellarea [ dbget head.topCells.insts.area ] {
        set default_area [ expr $default_area + $cellarea]
}
set UTIL 0.15
```

```
set core_area [ expr $default_area / $UTIL]
set WH [expr sqrt($core_area)]
set coreW 320.904
set coreH 307.567
set boundary_offset_x   1.2
set boundary_offset_y   1.5145
floorPlan -coreMarginsBy die -fplanOrigin llcorner -flip s -s ${coreW}
${coreH} ${boundary_offset_x} ${boundary_offset_y} ${boundary_offset_x}
${boundary_offset_y} -noSnapToGrid
initCoreRow
```

ICC 2 Example

```
set boundary_offset_x 1.2
set boundary_offset_y 1.2285
set ROW_PATTERN "H117_H169"
set default_area 1800
set UTIL  80
set core_area [ expr $default_area / [expr $UTIL/100.00]]
set Width [ expr sqrt($core_area/10)]
set Height [expr 2*$Width]
set coreBoundaryX2 [ expr [expr int($Width/0.192)*0.192 + 0.144 ] +
$boundary_offset_x ]
set coreBoundaryY2 [ expr [ expr int($Height/0.572)*0.572 -0.169] +
$boundary_offset_y]
initialize_floorplan -control_type core -core_offset [list
$boundary_offset_x $boundary_offset_y] -boundary " {$boundary_offset_x
$boundary_offset_y} {$coreBoundaryX2 $coreBoundaryY2}" -row_pattern
$ROW_PATTERN
```

Types of Floorplans

In VLSI design, there are several types of floorplans, each with its own characteristics and applications. Here are the main types:

- **Abutted Floorplan**: In this type, there is no spacing between the blocks (macros). The blocks are placed directly next to each other, which can help in reducing the overall area but may lead to routing congestion.
- **Non-Abutted Floorplan**: This type includes spacing (channels) between the blocks. The channels provide space for routing and can help in reducing congestion, but they may increase the overall area of the design.
- **Mixed Floorplan**: A combination of abutted and non-abutted floorplans. Some blocks are placed directly next to each other, while others have spaced between them. This approach aims to balance the benefits of both types.

These floorplan types help designers optimize the layout for performance, area, and routing efficiency.

Floorplan Control Parameters

Before implementing a floorplan in a tool, it's important to understand some key parameters that contribute to an effective floorplan:

Core Area

This is the part of the die (full chip) where all types of cells (such as standard cells, macros, blockages, and physical-only cells) can be placed. All cells must be placed within the core area.

- **Core Size**:
  Core size = (Standard cell area + Macro area + Blockage's area) / (Standard cell utilization)

Core utilization

Utilization defines the area occupied by standard cells, macros, and other cells within the core. If the core utilization is 0.8 (80%), it means that 80% of the core area is used for placing these cells, while the remaining 20% is reserved for routing purposes. This balance ensures there is enough space for routing interconnections without congestion.

Core utilization = (Macro area + std cell area + pads area) / (Total core area)

Aspect Ratio

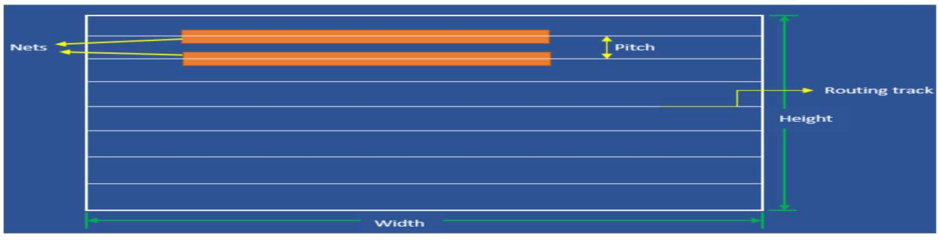
This is the ratio between the height and width of the core. For a square core area, the aspect ratio is 1.0. If the aspect ratio is 0.5, the width of the core is twice the height. Conversely, if the aspect ratio is 2.0, the height of the core is twice the width.

Aspect ratio = (Height of the core area) / (Width of the core area)
Or
Aspect ratio = (Horizontal routing tracks) / (Vertical routing tracks)

Terminologies

Routing Tracks

The entire metal layer can be divided into a number of horizontal and vertical lines, each of which is considered a routing track. These tracks are virtual and created solely for routing purposes; they do not physically exist in the design. A net or route middle is automatically aligned to the routing track, as illustrated in the picture below. The distance between these tracks is referred to as the pitch.



Manufacturing Grid

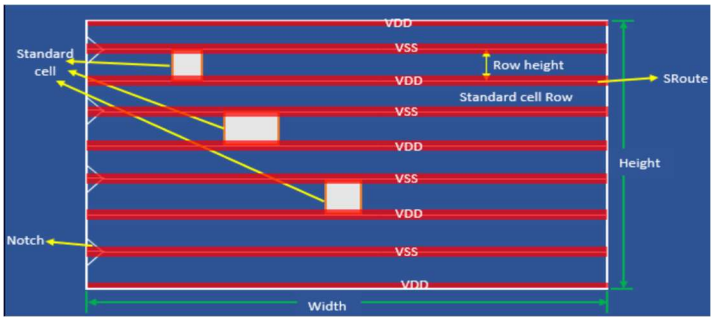
The minimum manufacturable area for a given technology node is defined by the technology node itself. This area represents the smallest feature size that can be reliably produced by the foundry for that specific technology. It sets the limits for how small the components and interconnections can be, ensuring that the design can be manufactured with high yield and reliability. It is defined in technology node.

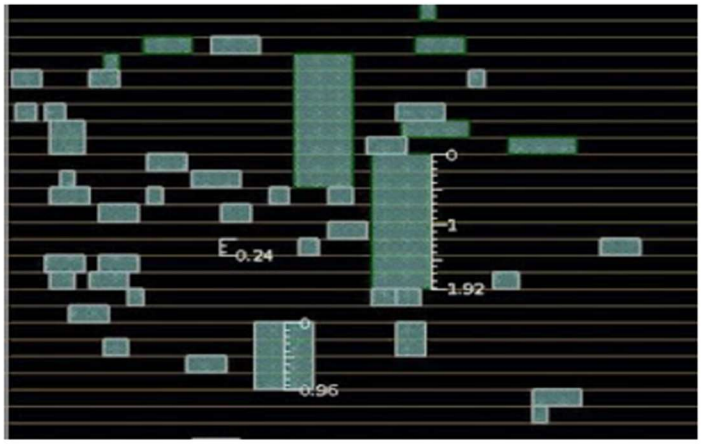
Standard Cell Site

The minimum width and height of a cell that can occupy space in the core area are defined by the standard cell site. All cells in the core area are multiples of these standard cell sites. The smallest unit, known as a filler cell, occupies only one standard cell site. This ensures that the layout is organized and that there are no gaps between cells, facilitating efficient placement and routing.

Standard Cell Rows

Standard cell rows are created in core area to place standard cells. During the placement stage all the standard cells in core area will have to align with in these rows. 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.
By seen the below figure you can understand that every standard cell height is fixed but width may vary



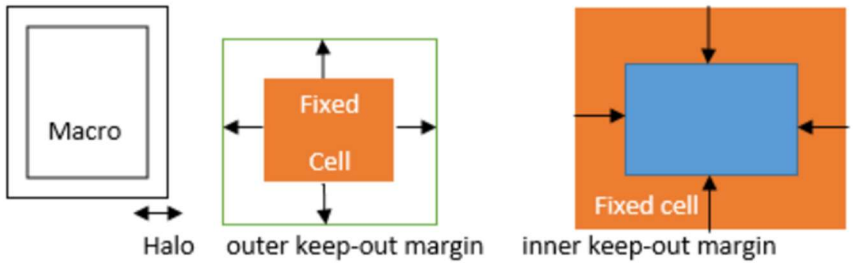


Inverted Rows

In the upper figure, you can see that every standard cell row has VDD and VSS. In the second row, this arrangement is inverted to VSS and VDD. This inversion allows the first and second row cells to share the same VSS, saving more area. Without this inverted row concept, a significant amount of area would be wasted.

Halo (Keep out margin)

This region around the fixed macros ensures that no other macros or standard cells are placed near the macro boundaries. The width of the keep-out margin on each side of the fixed cell can vary based on how you define it. Keeping cells out of these regions helps avoid congestion and improves the quality of results (QoR). The halos of two adjacent macros can overlap. If the macros are moved, their halos will move as well.



ICC command for keep out 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

Innovus command:
addHaloToBlock 10 10 10 10 macro_name

Innovus command	ICC 2 Command
addHaloToBlock	create_keepout_margin

Blockages

Blockages are specific locations where cell placement is restricted. They do not guide the tool but prevent standard cells, buffers, and inverters from being placed in certain areas. By using blockages, we ensure that no standard cells or other cells are placed in these designated areas. If macros are moved, the blockages will remain in their original positions. There are three types of blockages: soft, hard, and partial.

- **Soft blockages:**
  Only buffers can be placed in the specified area. This restriction prevents the placement of standard cells and hard macros during coarse placement but allows the placement of buffers and inverters during optimization, legalization, and clock tree synthesis.
  ICC 2 command:
  create_placement_blockages -boundary {{10 20} {100 200}} -name PB1 -type soft

Innovus command:
createPlacementBlockage -box (10 20 100 200) -name PB1 -type soft

- **Hard blockages:**
  No standard cells, macros, or buffers/inverters can be placed within the specified area during coarse placement, optimization, and legalization. This restriction helps avoid routing congestion at the corners of macros and controls the generation of power rails at the macros.

  ICC 2 command:
  create_placement_blockages -boundary {{10 20} {100 200}} -name PB1 -type hard

  Innovus command:
  createPlacementBlockage -box (10 20 100 200) -name PB1 -type hard

- **Partial blockages:**
  Partial blockages limit cell density in a specified area. By default, the blockage factor is set to 100%, preventing any cells from being placed in that region. However, if we want to reduce density without completely blocking the area, we can adjust the blockage factor accordingly.
  ICC 2 command:
  create_placement_blockages -boundary {{10 20} {100 200}} -name PB1 -type partial -blocked_percentage 40
  Partial blockages can limit cell density to a maximum of 60% (with 40% blocked) within a rectangular area defined by the corners (10, 20) and (100, 200). To allow unlimited usage of a partial blockage area, set the blockage percentage to zero.

  Innovus command:
  createPlacementBlockage -box (10 20 100 200) -name PB1 -type partial -density 60
  Here it will allow 60% cell placement in that region and it will block 40%.

Fly Lines

Macros are manually positioned using fly lines, which serve as virtual connections between macros and IO pads. These lines assist designers in understanding the logical connections between macros and pads. Fly lines act as guidelines, helping to minimize interconnect length and optimize routing resources.
Fly lines are two types:

- Macro to IO pin:

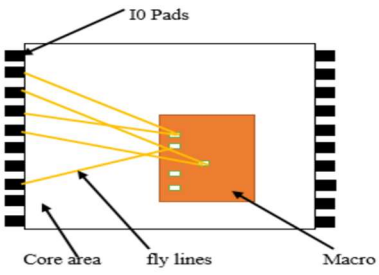


Fig a: macro to IO ports fly lines

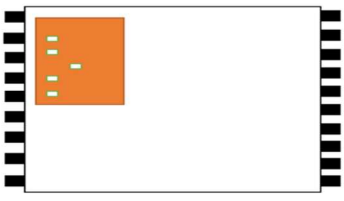


Fig b: macro placed at the core boundary

- Macro to macro fly lines:

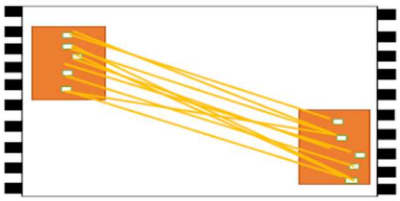


Fig a: macro to macro fly lines

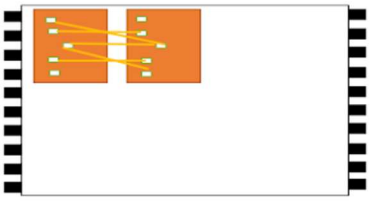


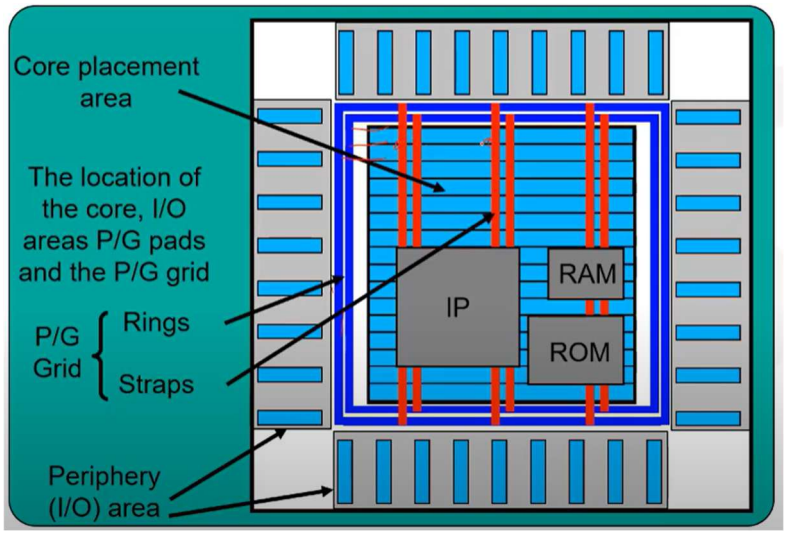
fig b: macros placed near to each other

Stages

To start floorplan first of all, required input files have to load into the PnR tool. Floorplanning in VLSI design involves several key stages to ensure an efficient and high-performance chip layout. Here are the main stages:

1. **Define Core and Die Size**: Determine the width and height of the core and the overall die size.
2. **I/O Pad Placement**: Place the input/output pads around the periphery of the die.
3. **Macro Placement**: Position the large, pre-designed blocks (macros) within the core area.
4. **Standard Cell Row Creation**: Create rows for placing standard cells, which are the basic building blocks of the design.
5. **Physical only cell Placement**: Place the Endcap and tap cells.
6. **Power Planning**: Design the power distribution network to ensure adequate power delivery to all parts of the chip.
7. **Placement Blockages**: Define areas where standard cells cannot be placed to avoid congestion and ensure proper routing.

After finishing the floorplan, we will get the following figure:



Core and Die Size

The **Core** is the central part of the chip where the main logic elements are placed. This includes:

- **Standard Cells**: These are the basic building blocks of the digital circuit.
- **Macros**: Larger functional blocks like memory, ALUs (Arithmetic Logic Units), etc.
- **Blockages**: Areas where no cells can be placed, often used to reserve space for routing or other purposes.

The core is essentially the active area of the chip where all the functional components are integrated. The layout of the core is crucial as it directly impacts the performance, power consumption, and overall efficiency of the chip.

The **Die** is the piece of semiconductor material on which the core and other components are fabricated. The die size includes:

- **Core Area**: The area occupied by the core.
- **I/O Pads**: These are the input/output pads that connect the chip to the external world.
- **Margins**: Space between the core and the edges of the die, often used for routing and other purposes.

The die size is determined by the core size and additional space required for I/O pads and margins. It is a critical parameter as it affects the cost and yield of the semiconductor manufacturing process.

IO Pad Placement

I/O (Input/Output) pads are crucial components in VLSI design as they serve as the interface between the chip and the external world. Proper placement of these pads is essential for ensuring signal integrity, minimizing noise, and optimizing the overall performance of the chip. Below are the key aspects of I/O pad placements:

Pad Ring Design

- **Pad ring**:
  The I/O pads are typically placed around the periphery of the chip, forming a pad ring. This ring provides a structured way to connect the internal circuitry of the chip to the external pins.
- **Types of pads**:
  There are different types of I/O pads, including power pads, ground pads, signal pads, and special function pads (e.g., ESD protection pads).

Placement Strategies

- **Signal integrity**:
  To maintain signal integrity, high-speed signal pads are placed close to each other and away from noisy power and ground pads.

- **Power distribution**:
  Power and ground pads are distributed evenly around the chip to ensure uniform power delivery and minimize voltage drops.
- **Noise isolation**:
  Analog and digital pads are often separated to reduce noise interference. Special pads with noise isolation features may be used for sensitive analog signals.

ESD Protection

- **Electrostatic Discharge (ESD):**
  ESD protection pads are placed strategically to protect the chip from electrostatic discharge, which can damage the internal circuitry.
- **ESD rings**:
  ESD protection is often implemented using ESD rings that surround the core and provide a path for discharging static electricity.

Routing Considerations

- **Routing paths**:
  The placement of I/O pads affects the routing paths of signals. Efficient routing minimizes the length of interconnections, reducing delay and power consumption.
- **Bonding techniques**:
  Different bonding techniques, such as wire bonding and flip-chip bonding, influence the placement of I/O pads. For example, flip-chip bonding allows for more flexible pad placement compared to wire bonding.

Package Constraints

- **Package type:**
  The type of package (e.g., BGA, QFP) used for the chip influences the placement of I/O pads. The package must accommodate the pad layout and provide reliable connections to the external pins.
- **Thermal considerations:**
  Pads are placed considering thermal management to ensure that heat generated by the chip is effectively dissipated.

The steps in I/O pad placement

They are as follow:

Initial Placement: Based on the design specifications, an initial placement of I/O pads is done. This involves defining the pad ring and placing the pads according to their function and connectivity requirements.

Optimization: The initial placement is optimized to improve signal integrity, reduce noise, and ensure efficient power distribution. This may involve iterative adjustments and simulations.

Verification: The final placement is verified through simulations to ensure that it meets all design constraints and performance requirements.

By carefully planning the placement of I/O pads, designers can enhance the performance, reliability, and manufacturability of the chip.

Pin Placement

Pin placement is a crucial step that involves determining the optimal locations for input/output pins on a chip. It typically begins during the floorplanning stage. Proper pin placement helps in maintaining signal integrity by minimizing the length of critical signal paths and reducing crosstalk.

Innovus Command	ICC 2 Command
edit_pin	place_pin

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.

There are two types of macros:

- **Hard Macros**: These are pre-designed and fixed circuits. We cannot see the internal functionality or the types of gates used within them. However, we do have access to their timing information.
- **Soft Macros**: These are flexible circuits where the internal functionality and the types of gates used are visible. We also have access to their timing information. Since they are not fixed, they can be modified and optimized as needed.

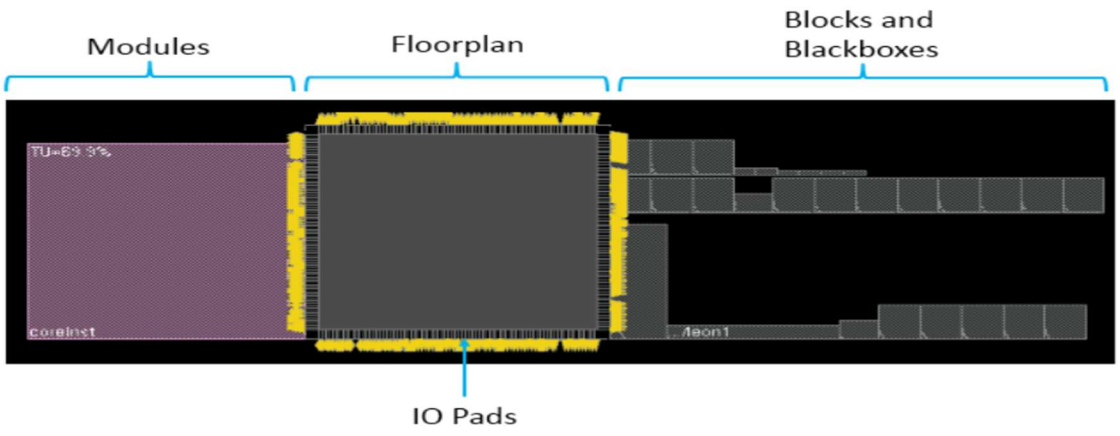
Macro placement is a crucial aspect of PnR due to its significant impact on the overall performance, power consumption, and area efficiency of the chip. Here are some key reasons why macro placement is important:

1. **Performance Optimization**: Proper macro placement helps minimize the critical path delays, ensuring that signals travel the shortest possible distance. This is essential for achieving high-speed performance in the design.
2. **Power Efficiency**: By strategically placing macros, designers can reduce power consumption. This is achieved by minimizing the switching activity and reducing the capacitance of interconnects.
3. **Area Utilization**: Efficient macro placement maximizes the utilization of the available silicon area. This helps in fitting more functionality into a smaller chip area, which is crucial for cost-effective manufacturing.
4. **Routing Congestion**: Proper placement of macros helps in reducing routing congestion. This ensures that there are enough routing resources available for signal interconnections, which is vital for maintaining signal integrity and avoiding timing violations.
5. **Thermal Management**: Strategic placement of macros can help in managing the thermal profile of the chip. By avoiding hotspots and ensuring even distribution of heat-generating components, designers can enhance the reliability and longevity of the chip.
6. **Design Complexity**: As VLSI designs become more complex, the importance of macro placement increases. It helps in managing the complexity by logically grouping related components and ensuring that the design is easier to manage and verify.

When placing macros, it's essential to keep in mind several important steps:

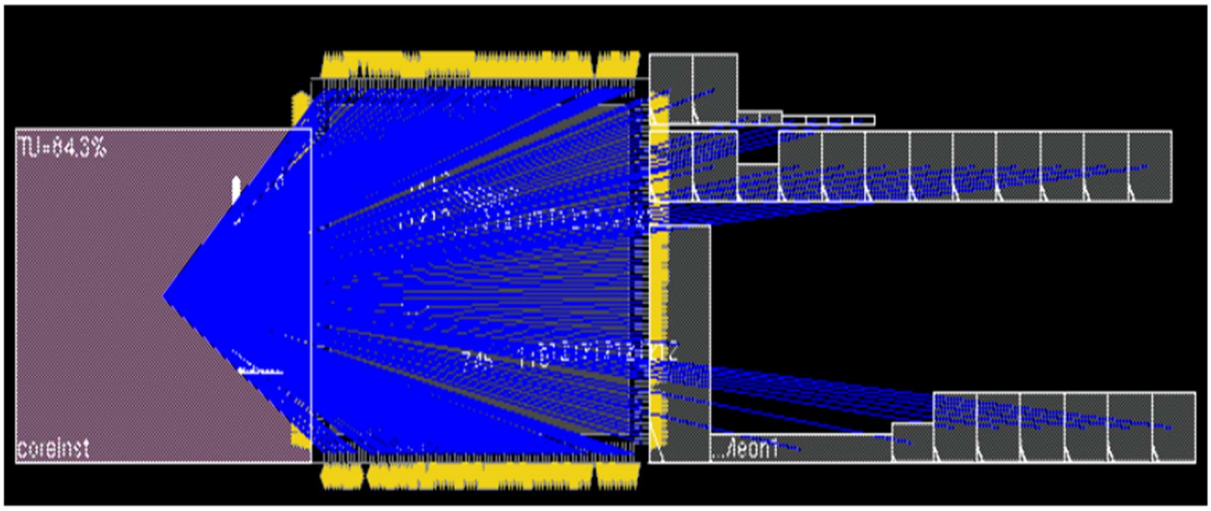
1. **Position Macros on the Periphery**: Whenever possible, place macros around the edge of the core area to reduce routing congestion and facilitate power supply.
2. **Group Logically**: Place macros that interact closely together to minimize signal routing distances and enhance performance.
3. **Orient Pins Properly**: Ensure that macro pins face towards the core or each other to reduce detour routing and improve timing and congestion management.
4. **Allocate Routing Space**: Leave sufficient space around macros for net routing and power grid, using congestion maps to identify and address hotspots.
5. **Avoid Criss-Crossing**: Optimize placement to prevent crisscrossing of macros, which can complicate routing and increase delays.
6. **Use Halos and Blockages**: Implement halos around macros to prevent congestion and use blockages to manage gaps between macros.
7. **Verify Connections**: Utilize flight lines to check connections between macros and fixed elements like I/O pins and pre-placed macros for optimal placement.
8. **Minimize Open Fields**: Reduce open fields to maximize the utilization of the available area.
9. **Consider Power Domains**: Ensure macros are placed within their respective power domain fences to maintain power integrity.
10. **Optimize Channels**: Optimize channels to avoid congestion near ports and ensure smooth routing. channel width = (number of pins*pitch)/number of layers either horizontal or vertical.
11. Avoid notches while placing macros, if anywhere notches is present then use hard blockages in that area.
12. 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.
13. 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.
14. Between two macros at least one pair of power straps (power and Ground) should be present.

Basic Example of Floorplan

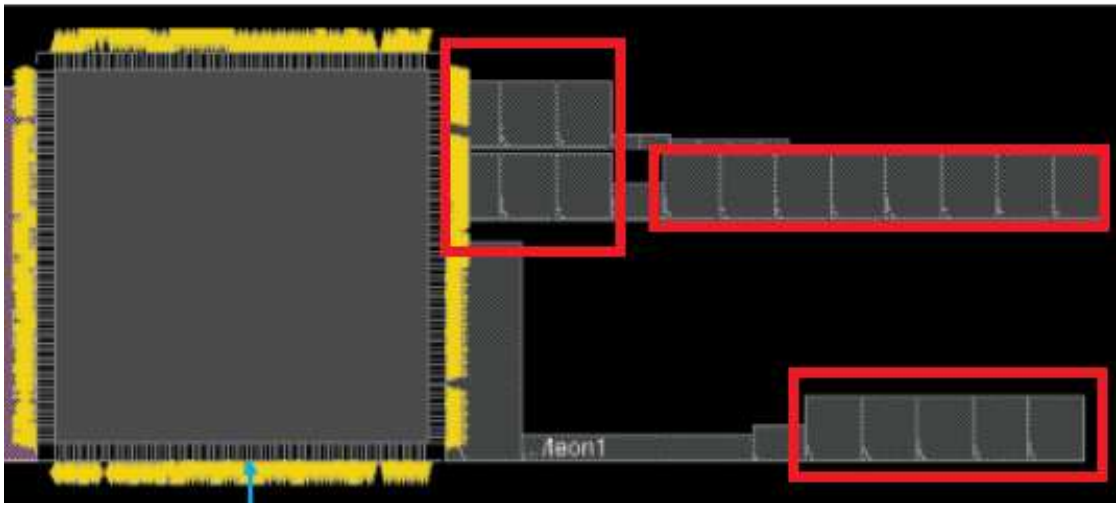


After importing the design, a basic floorplan is displayed. The IO pads are placed according to the IO placement file provided during Design Import. If no file is provided, they are placed randomly. On the left side of the floorplan, pink modules represent the modules in the Verilog netlist. On the right side, hard macros (blocks) and any defined black-boxes are displayed.

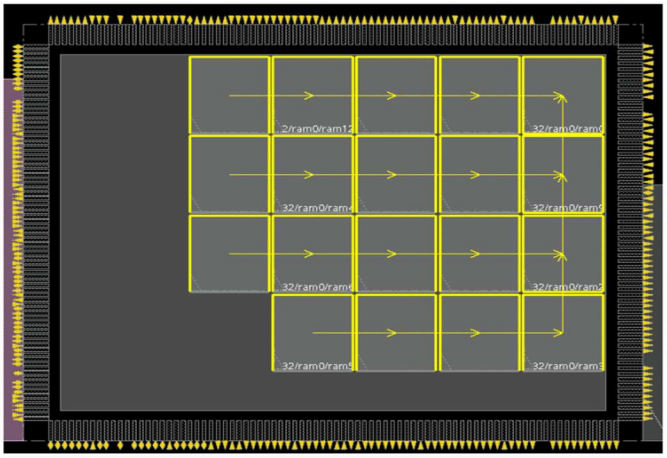
First, it's important to understand which blocks are related. The initial task is to determine which macros belong to the same hierarchy. Additionally, it's important to consider routing congestion and the orientation of the macros. Manually placing macros is often more effective than automatic placement. By selecting the block and using the flight lines, we can identify the relationships between blocks. In the following figure, you can see the flight lines after selecting all macros:



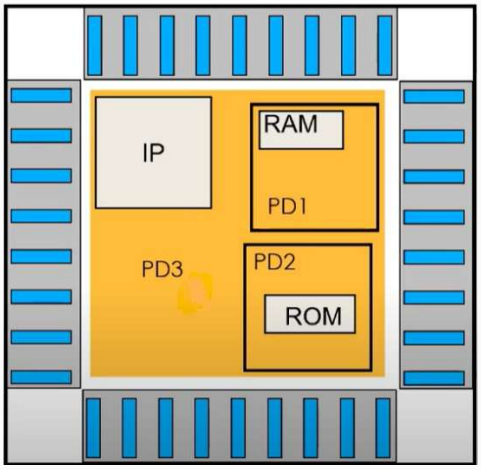
In the figure below, the red-marked blocks are related to each other:



Therefore, it's necessary to place these blocks close to each other to facilitate easy communication and avoid routing congestion. As shown in the following figure, they should be placed accordingly:



From the figure above, the related macros are placed together. It's also important to leave space between the macros to allow for standard cell placement. Controlling the utilization of the channel between two macros is a good practice. Also, if there are some macros which are connected to the ports need to place those macros near the port. If there are multiple voltage domains, ensure that the specific macros are placed within their respective voltage domains.



From the figure above, there are three power domains: PD1, PD2, and PD3. The RAM is in the PD1 domain, the ROM is in the PD2 domain, and the IP is in the PD3 domain. Therefore, these components need to be placed in their respective power domains.

Standard Cell Row Creation

Standard cells are the fundamental building blocks used in digital VLSI design. These cells are pre-designed and pre-characterized to perform specific logic functions, such as AND, OR, NAND, NOR gates, flip-flops, and more. The process of creating standard cell rows involves arranging these cells in a structured manner to facilitate efficient placement and routing.

Defining Cell Height: Standard cells have a uniform height, ensuring they fit neatly in rows.
Power Rails: Each row has horizontal power (VDD) and ground (VSS) rails at the top and bottom.

N-Well and P-Well Regions: These regions accommodate PMOS and NMOS transistors, respectively.
Cell Placement: Cells are placed in rows with aligned power rails and input/output pins for easy routing.
Abutment: Cells are placed next to each other without gaps to reduce area and simplify routing.
Routing Channels: Spaces between rows used for interconnecting cells.
Design Rule Checks (DRC): Ensures the layout meets manufacturing constraints.

This structured approach optimizes space, performance, and power efficiency in VLSI designs.

Physical Only Cell Placement

Physical only cells are special cells used in VLSI design that do not perform any logical functions but serve important physical purposes. These cells include filler cells, decoupling capacitors, ESD cells, DTCD cells, tap cells and boundary or endcap cells. Proper placement of these cells is crucial for maintaining the physical integrity and performance of the chip.

For floorplan stage only ESD cells, DTCD cells, Endcap and Tap cells are needed.

1. **ESD cells:**
   Electrostatic Discharge (ESD) cells are used to protect the chip from electrostatic discharge events, which can cause damage to the internal circuitry. These cells are strategically placed to provide a path for discharging static electricity, thereby safeguarding the chip. Generally, ESD protection is critical for lower technology nodes, they are more susceptible to damage from electrostatic discharge, making ESD cells essential for ensuring the reliability and longevity of the chip.
2. **DTCD cells:**
   DTCD (Dummy Test Chip Design) cells are used for testing and calibration purposes. They help in verifying the manufacturing process and ensuring that the chip meets the required specifications. These cells do not perform any logical functions but are essential for process validation. This type of cell is crucial for lower technology nodes for maintaining the physical integrity and reliability of the chip, even though they do not contribute to the logical operations.
3. **Endcap cells**:
   Endcap cells, also known as boundary cells, are special physical-only cells used in VLSI design. They do not perform any logical functions but serve important physical purposes, particularly at the boundaries of the chip. These cells are crucial for maintaining the physical integrity and reliability of the chip.
   Key functions of standard cells:
   - **Protection of standard cells:**
     - **Gate protection:**
       Endcap cells protect the gates of standard cells placed near the boundary from damage during the manufacturing process. This is especially important to prevent issues like gate oxide damage.
     - **DRC compliance:**
       They help avoid design rule check (DRC) violations related to the base layers (e.g., N-well and implant layers) at the boundaries.

- **Alignment and Continuity:**
  - **Row termination:**
    Endcap cells are placed at the ends of each placement row to terminate the row properly. This ensures that the rows are aligned and continuous, which is essential for maintaining the integrity of the power and ground rails.
  - **Block integration:**
    They are also placed at the top and bottom rows at the block level to facilitate integration with other blocks.
- **Isolation and Stability:**
  - **Isolation:**
    Endcap cells often include dummy poly layers that help isolate different domains and the core, preventing electrical interference.
  - **Mechanical stability:**
    These cells provide mechanical stability to the chip, especially at the corners, ensuring that the chip structure remains intact during and after manufacturing.

Placements of endcap cells:

Endcap cells are placed just after the macro placement and site row creation. They are considered pre-placed cells, meaning they are positioned before the placement of standard cells.
Place-and-route (P&R) tools can automatically insert endcap cells at the ends of rows and around the periphery of the core. This ensures that the placement is consistent and adheres to design rules.
In some cases, manual adjustments may be needed to ensure that the endcap cells are correctly positioned, especially in complex designs with multiple blocks and power domains.

Innovus Command	ICC 2 Command
addEndCap	create_boundary_cells

4. **Tap cells**:

Tap cells, also known as well tap cells, are special physical-only cells used in VLSI design to maintain the integrity of the substrate and prevent latch-up issues in CMOS circuits. These cells do not perform any logical functions but are crucial for ensuring the reliable operation of the chip.
Key functions of tap cells:

- **Latch-Up Prevention:**
  - **Latch-Up:**
    Latch-up is a condition where a parasitic structure within the CMOS circuit creates a low-impedance path between the power supply and ground, leading to high current flow and potential chip failure.
  - **Prevention:**
    Tap cells connect the n-well to the power supply (VDD) and the p-substrate to the ground (VSS), preventing the formation of parasitic paths that can cause latch-up.
- **Substrate Biasing:**
  - **N-well and P-substrate:**
    Tap cells ensure that the n-well and p-substrate are properly biased. This is essential for maintaining the correct operation of PMOS and NMOS transistors.

- **Biasing:**
  By providing a direct connection to VDD and VSS, tap cells help maintain a stable substrate bias, which is crucial for the reliable operation of the transistors.

Placements of Tap cells:

- **Pre-Placement Stage:**
  - **Initial placement:**
    Tap cells are placed after the macro placement and endcap placement. This is known as the pre-placement stage.
  - Regular intervals:
    Tap cells are distributed at regular intervals throughout the layout to ensure uniform substrate biasing and effective latch-up prevention.
- **Pattern:**
  - Tap cells are often placed in a checkerboard pattern, where they are positioned in alternating rows and columns. This pattern provides maximum coverage and ensures that the entire substrate is properly biased.
  - **Adjustments:**
    If a macro or other large block is in the path of the tap cells, the placement pattern is adjusted to maintain coverage without interfering with the macro.
- **Design Rule Compliance:**
  - **DRC rules:**
    The placement of tap cells must comply with design rule checks (DRC) specific to the technology node being used. This includes maintaining the maximum allowable distance between tap cells to ensure effective biasing.

Innovus Command	ICC 2 Command
addWellTap	create_tap_cells

Power Planning

Power planning is a crucial stage in VLSI design, aimed at ensuring that all parts of the chip receive adequate power while minimizing power-related issues such as IR drop and electromigration. This process involves creating a robust power distribution network that delivers power efficiently to every macro, standard cell, and other components on the chip.

Define Power Requirements: Calculate total power and current needs.
Create Power Rings: Surround core and I/O areas with VDD and VSS rings.
Design Power Straps and Mesh: Lay out horizontal and vertical power lines to form a grid.
Place Power Rails: Add power rails to standard cell rows and macros.

IR Drop Analysis: Simulate and adjust to minimize voltage drops.
Electromigration Checks: Ensure current density is within safe limits.
Power Grid Verification: Verify connectivity and run simulations to meet design specs.
Effective power planning ensures stable power distribution, optimizing performance and efficiency.

Placement Blockages

Placement blockages are used in VLSI design to control where cells can and cannot be placed on the chip. They help manage the layout, avoid congestion, and ensure that critical areas are reserved for specific purposes. There are three types of placement blockages: hard, soft and partial.

Key functions of placement blockages:

- Defining Standard Cells and Macro Areas:
  - Purpose:
    Ensure that standard cells and macros are placed in designated areas, maintaining an organized layout.
  - Implementation:
    Blockages are created around macros to prevent standard cells from being placed too close, which could lead to routing difficulties.
- Reserving Channels for Buffer Insertion:
  - Purpose:
    Reserve space for buffers and inverters needed during the CTS process.
  - Implementation:
    Soft blockages are used to ensure that there is enough room for these elements, which are crucial for maintaining clock signal integrity.
- Preventing Congestion Near Macros:
  - Purpose:
    Avoid routing congestion around large blocks like RAM or IP blocks.
  - Implementation:
    Hard blockages are placed around these macros to ensure that standard cells are not placed too close, allowing for better routing paths.
- Managing Power Distribution:
  - Purpose:
    Control the placement of cells to ensure efficient power distribution.
  - Implementation:
    Blockages can be used to guide the placement of cells in a way that optimizes power rail usage and minimizes IR drop.

Innovus Command	ICC 2 Command
createPlaceBlockage -type hard/soft/partial	create_placement_blockage -type <blockage type>

Major Issues

- If utilization is too high, it can make it difficult to close a design. High utilization can lead to routing congestion, which negatively impacts the optimization and legalization stages.
  Ensuring an appropriate utilization level is crucial to avoid these issues and achieve a successful design closure.
- If macros are not properly oriented and macros of the same hierarchy are not placed close to each other, it can lead to routing congestion, which will negatively impact timing.
- Tap cells should be placed properly; otherwise, we could end up with latch-up issues, which are critical for the chip.

References

1. https://www.physicaldesign4u.com/2019/12/floorplanning-floor-planning-is-art-of.html
2. https://vlsitalks.com/physical-design/floorplan/
3. Basic Floorplanning in Innovus Implementation System
4. DVD - Lecture 6c: Floorplanning - YouTube